IBM WebSphere Commerce

Guida per il programmatore

Versione 5.4

GC13-3051-00

IBM WebSphere Commerce

Guida per il programmatore

Versione 5.4

GC13-3051-00

Nota:
Prima di utilizzare queste informazioni e il prodotto supportato, consultare il paragrafo Informazioni particolari.

Prima edizione (marzo 2002), Rilascio 2

Questa versione si riferisce ai seguenti prodotti:

- IBM WebSphere Commerce Business Edition per Windows NT e Windows 2000, Versione 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Business Edition per AIX, Versione 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Business Edition per software Solaris Operating Environment, Versione 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Business Edition per Linux, Versione 5.4 (Programma 5724-A18)
- IBM WebSphere Commerce Studio, Business Developer Edition per Windows NT e Windows 2000, Versione 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Professional Edition per Windows NT e Windows 2000, Versione 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Professional Edition per AIX, Versione 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Professional Edition per software Solaris Operating Environment, Versione 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Professional Edition per Linux, Versione 5.4 (Programma 5724-A18)
- IBM WebSphere Commerce Studio, Professional Developer Edition per Windows NT e Windows 2000, Versione 5.4 (Program 5724-A18)

e a tutti i successivi rilasci e modifiche di questo prodotto, se non diversamente specificato in nuove edizioni. Accertarsi di utilizzare l'edizione appropriata per il livello del prodotto.

Richieste di ulteriori copie di questo prodotto o informazioni tecniche sullo stesso vanno indirizzate ad un rivenditore autorizzato IBM o ad un rappresentante commerciale IBM.

Come ultima pagina del manuale è stato predisposto un foglio riservato ai commenti del lettore. Se il foglio è stato rimosso, i commenti su questa pubblicazione possono essere inviati al seguente indirizzo:

SELFIN S.p.A.
Translation Assurance
Via F. Giordani, 7
80122 Napoli
ITALY

Tutti i commenti e suggerimenti inviati potranno essere utilizzati liberamente dall'IBM o dalla Selfin e diventeranno esclusiva delle stesse.

Prima di iniziare

Il manuale *IBM WebSphere Commerce - Guida per il programmatore* fornisce informazioni sull'architettura e sul modello di programmazione di WebSphere Commerce. In particolare, contiene dettagli relativi ai seguenti argomenti:

- Interazioni tra i componenti
- Modelli di progettazione
- Modello oggetti permanente
- Controllo dell'accesso
- Messaggi e gestione degli errori
- Implementazione dei comandi
- Strumenti per lo sviluppo
- Sviluppo del codice personalizzato

Inoltre, in questo manuale sono inclusi i seguenti supporti didattici:

Creazione di nuova logica aziendale
Questo supporto didattico descrive come creare nuovi comandi, bean di dati ed enterprise seguendo il modello di programmazione di WebSphere Commerce. Inoltre viene illustrato come integrare la logica in un negozio esistente e distribuire il codice per un WebSphere Commerce Server di destinazione.

Modifica ed estensione della logica aziendale esistente
Questo supporto didattico si suddivide in due sezioni. La prima sezione mostra come aggiungere nuova logica a un comando di controller esistente. La seconda sezione illustra come modificare un comando di attività esistente e un bean entità di WebSphere Commerce. Inoltre, mostra come integrare le modifiche in un negozio esistente e distribuire codice per un WebSphere Commerce Server di destinazione.

Convenzioni utilizzate in questa pubblicazione

Questa pubblicazione utilizza le seguenti convenzioni tipografiche:

Grassetto: indica i comandi o i controlli della GUI (Graphical User Interface) come i nomi dei campi, i pulsanti o le scelte di menu.

`Monospazio`: indica gli esempi di testo da immettere esattamente come sono mostrati e i percorsi per le directory.

Corsivo: utilizzato per evidenziare delle parti e per indicare le variabili cui l'utente attribuisce dei valori.

Questa icona indica un Suggerimento, vale a dire ulteriori informazioni che aiutano a completare un'attività.

Windows indica informazioni specifiche di WebSphere Commerce per Windows NT e Windows 2000.

AIX indica informazioni specifiche per WebSphere Commerce per AIX.

Solaris indica informazioni specifiche per WebSphere Commerce per software Solaris™ Operating Environment.

400 indica informazioni specifiche per WebSphere Commerce per IBM @server iSeries 400 (noto in passato come AS/400)

Linux indica le informazioni specifiche per WebSphere Commerce per Linux.

DB2 indica informazioni specifiche DB2 Universal Database

Oracle indica informazioni specifiche di Oracle®.

Professional indica informazioni specifiche di WebSphere Commerce Professional Edition.

Business indica informazioni specifiche di WebSphere Commerce Business Edition.

Conoscenze richieste

Questo manuale si rivolge agli sviluppatori di negozio che devono essere in grado di personalizzare un'applicazione WebSphere Commerce. Gli sviluppatori di negozio che eseguono le estensioni programmatiche devono avere conoscenze nelle seguenti aree:

- Java
- Architettura del componente Enterprise JavaBeans (EJB)
- Tecnologia di pagina JavaServer
- HTML

- Tecnologia di database
- VisualAge per Java, Enterprise Edition, Versione 3.5

Dove reperire informazioni

Questo manuale potrà essere aggiornato in futuro. Per aggiornamenti, visitare i seguenti siti Web di WebSphere Commerce:

Business

`http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html`

Professional

`http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html`

Gli aggiornamenti possono includere nuove informazioni, supporti didattici integrativi o aggiornati e codice di esempio correlato ai supporti didattici.

Indice

Parte 1. Concetti e architettura

Capitolo 1. Panoramica

Componenti software di WebSphere Commerce

Prima di esaminare il funzionamento di WebSphere Commerce Server, è opportuno analizzare l'insieme dei componenti software correlati al processo di personalizzazione. Il diagramma seguente illustra una visualizzazione semplificata di questi prodotti software:

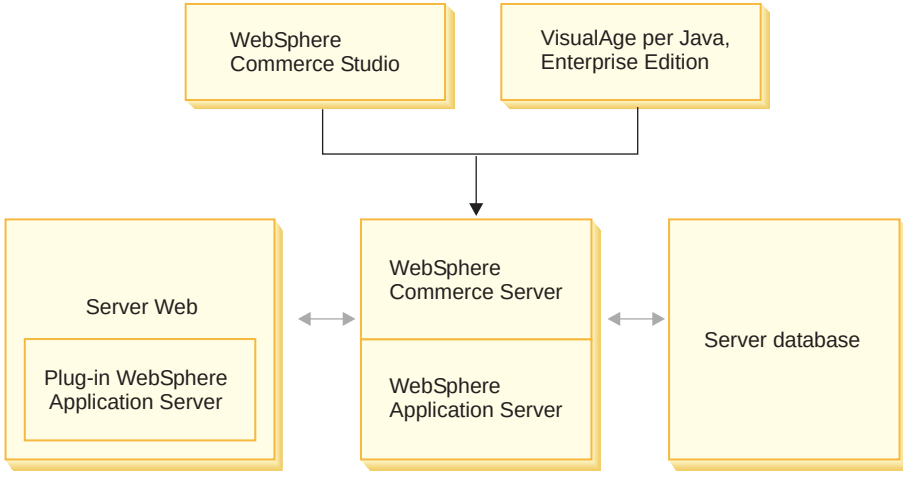


Figura 1.

Il server Web è il primo punto di contatto per le richieste HTTP in entrata per l'applicazione e-commerce. Per disporre di un'interfaccia efficace con WebSphere Application Server, utilizza il plug-in WebSphere Application Server.

WebSphere Commerce Server viene eseguito in WebSphere Application Server e può, quindi, sfruttare i vantaggi offerti dalle funzioni del server di applicazione. Il server del database contiene la maggior parte dei dati dell'applicazione, compresi dati relativi ai prodotti e agli acquirenti. In generale, le estensioni all'applicazione vengono eseguite modificando o estendendo il codice di WebSphere Commerce Server. Inoltre, è possibile che sia necessario memorizzare nel proprio database i dati che si trovano all'esterno dello schema del database WebSphere Commerce.

Per creare logica aziendale personalizzata, gli sviluppatori utilizzano due strumenti: WebSphere Commerce Studio e VisualAge per Java, Enterprise Edition. WebSphere Commerce Studio viene utilizzato per creare e gestire le

attività di fronte negozio quali le maschere JSP. VisualAge per Java, Enterprise Edition viene utilizzato per creare nuova logica aziendale in Java, che estenda le funzionalità esistenti o crei nuove funzioni. Se l'applicazione che si utilizza richiede estensioni allo schema del database, gli sviluppatori di database devono utilizzare strumenti per la creazione di nuove tabelle.

Architettura dell'applicazione WebSphere Commerce

Dopo aver esaminato la compatibilità dei vari componenti software in merito alla personalizzazione, è importante comprendere l'architettura dell'applicazione. In tal modo, sarà possibile conoscere gli elementi che costituiscono i principali livelli e le parti che possono essere modificate. Il diagramma riportato di seguito indica i vari livelli che costituiscono l'architettura dell'applicazione:

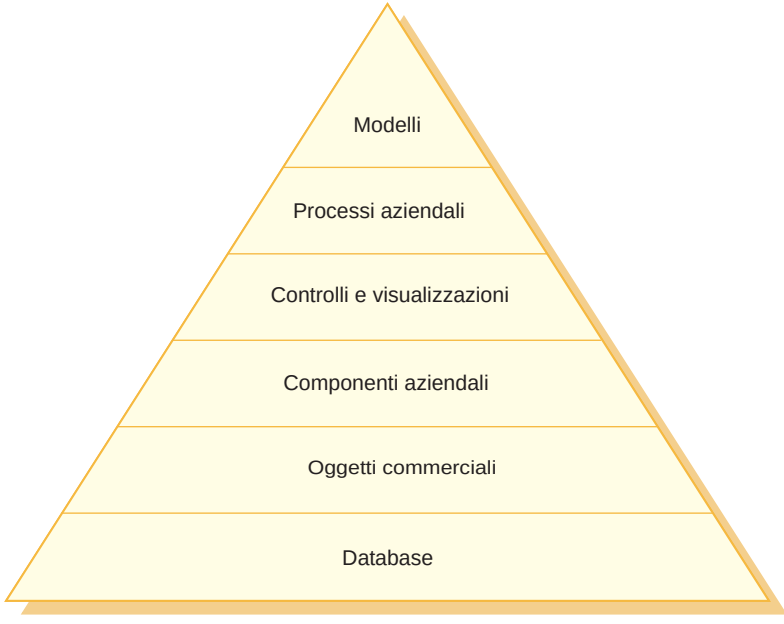


Figura 2.

Di seguito viene riportata la descrizione dei singoli livelli dell'architettura:

Database

WebSphere Commerce utilizza uno schema del database progettato specificamente per applicazioni e-commerce e per i relativi requisiti di dati. Di seguito sono riportati esempi di tabelle in uno schema:

- Utente
- Ordine
- Prodotto

Oggetti aziendali

Gli oggetti aziendali sono entità all'interno del dominio aziendale e incorporano la logica basata sui dati, necessaria per estrarre o interpretare informazioni contenute all'interno del database. Queste entità sono conformi alle specifiche di Enterprise JavaBeans.

I bean entità funzionano come interfaccia tra l'applicazione aziendale e il database. Inoltre, i bean entità sono più facilmente comprensibili delle complesse relazioni tra le colonne nelle tabelle del database.

Componenti aziendali

I componenti aziendali sono unità della logica aziendale. Eseguono una logica aziendale di procedura generica. La logica viene implementata utilizzando il modello WebSphere Commerce dei comandi di controller e di quelli di attività. Un esempio di questo tipo di componente è il comando di controller OrderProcess. Questo particolare comando incorpora tutta la logica aziendale necessaria per eseguire un ordine tipico. L'applicazione e-commerce richiama il comando OrderProcess, che a sua volta richiama una serie di altri comandi per eseguire singole unità di lavoro. Ad esempio, i singoli comandi dell'attività assicurano la disponibilità di inventario sufficiente a soddisfare i requisiti dell'ordine, a elaborare i pagamenti, ad aggiornare lo stato dell'ordine e, al completamento del processo, a diminuire la disponibilità dell'inventario della quantità appropriata.

Controlli e viste

Un controller Web determina l'implementazione e la vista appropriata del comando di controller da utilizzare. Le implementazioni possono essere specifiche del negozio.
Le viste riportano i risultati dei comandi e delle azioni dell'utente. Vengono implementate con le maschere JSP. Esempi di queste viste comprendono ProductDisplay (riproduce una pagina di prodotto con informazioni fondamentali per il prodotto dell'acquirente selezionato) e OrderPrepare (presenta all'acquirente un modulo per inoltrare le informazioni necessarie per l'ordine).

Processi aziendali

Gli insiemi di componenti aziendali e di viste consentono di creare processi di flusso di lavoro e flusso di sito conosciuti come processi aziendali. Alcuni esempi di processi aziendali comprendono:

Registrazione utente

Questo processo aziendale comprende i componenti aziendali, ad esempio, il comando UserRegistrationAdd che crea un record di registrazione per un nuovo utente, e le viste relative a tutte le fasi che costituiscono il processo di registrazione utente.

Navigazione nei cataloghi
Questo processo aziendale comprende i componenti aziendali, ad esempio i comandi StoreCatalogDisplay e CategoryDisplay, che mostrano rispettivamente i cataloghi di un negozio e le categorie in essi contenute, e le viste relative a tutte le fasi che costituiscono il processo di navigazione in un catalogo.

Modelli
Se raggruppati insieme, i livelli bassi del diagramma formano i modelli aziendali e-commerce. Un esempio di modello aziendale e-commerce è il modello azienda-a-consumatore utilizzato dal negozio di esempio InFashion. Un altro esempio è il modello business-to-business utilizzato dal negozio di esempio ToolTech.

Architettura runtime di WebSphere Commerce

Nella sezione precedente è stata introdotta l'architettura dell'applicazione, nella quale sono stati descritti i vari livelli dell'applicazione WebSphere Commerce da un punto di vista dell'applicazione aziendale. Questa sezione descrive come implementare l'architettura di runtime.

I principali componenti dell'architettura di runtime di WebSphere Commerce sono:

- Motore servlet
- Listener di protocollo
- Gestore degli adattatori
- Adattatori
- Controller Web
- Comandi
- Bean entità
- Bean di dati
- Gestore dei bean di dati
- Pagine di visualizzazione
- File XML

Le interazioni tra i componenti di WebSphere Commerce sono illustrate nel diagramma seguente. Ulteriori dettagli su ciascun componente vengono forniti nelle sezioni successive.

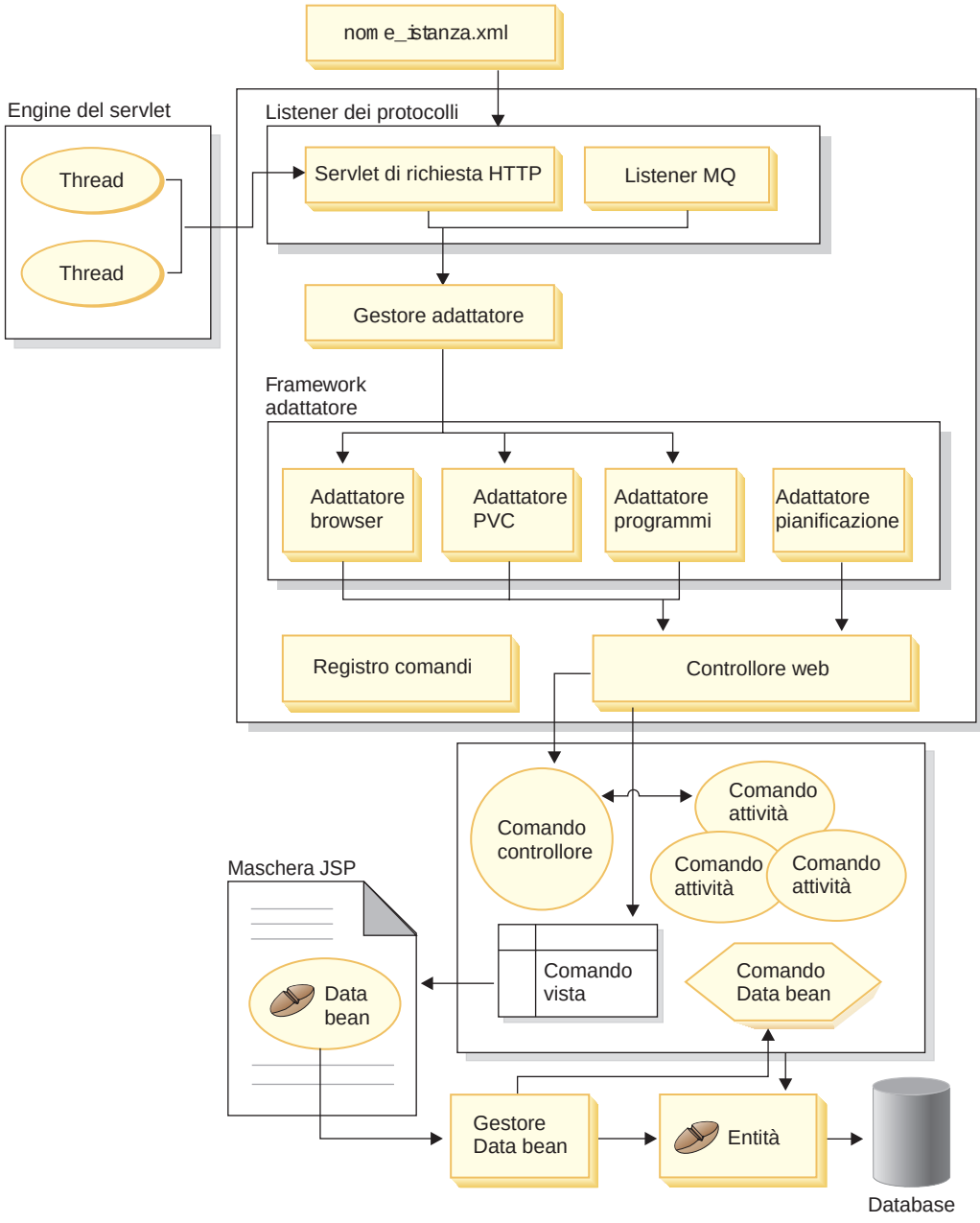

nome_istanza.xml
Engine del servlet
Thread
Thread
Listener dei protocolli
Servlet di richiesta HTTP
Listener MQ
Gestore adattatore
Framework adattatore
Adattatore browser
Adattatore PVC
Adattatore programmi
Adattatore pianificazione
Registro comandi
Controllore web
Comando controllore
Comando attività
Comando attività
Comando attività
Maschera JSP
Data bean
Comando vista
Comando Data bean
Gestore Data bean
Entità
Database

Figura 3.

Motore servlet

Il motore servlet fa parte dell'ambiente di runtime di WebSphere Application Server e opera come programma di distribuzione per le richieste URL in entrata. Questo motore gestisce un pool di thread per controllare le richieste. Ciascuna richiesta in entrata viene eseguita in un thread separato.

Le versioni precedenti di WebSphere Commerce utilizzavano un server delle applicazioni C++ che implementava il programma per la distribuzione delle attività per le richieste URL gestendo un insieme di processi di sistema preassegnati. Questo modello obsoleto, che utilizzava i processi di sistema, richiedeva una maggiore quantità di risorse rispetto al nuovo modello che, invece, utilizza i thread Java. Sfruttando il motore servlet, la nuova architettura di runtime di WebSphere Commerce fornisce una soluzione aziendale più scalabile.

Gestore degli adattatori

Il gestore degli adattatori stabilisce quale adattatore è in grado di gestire la richiesta, quindi inoltra la richiesta a questo adattatore.

Listener di protocollo

I comandi di WebSphere Commerce possono essere richiamati da vari dispositivi. Esempi di tali dispositivi sono:

- Browser Internet
- Telefoni cellulari che utilizzano i browser Internet
- Applicazioni per le relazioni tra le aziende che inviano messaggi XML utilizzando MQSeries
- Sistemi di acquisizione che inviano richieste utilizzando XML su HTTP
- Lo scheduler di WebSphere Commerce che esegue un processo in background

I dispositivi possono utilizzare diversi protocolli di comunicazione. Un listener di protocollo è un componente di runtime che riceve richieste in entrata dal trasporto e le distribuisce agli adattatori appropriati in base al protocollo utilizzato. Esempi di listener di protocollo sono:

- Servlet delle richieste
- Listener MQSeries

Quando il servlet delle richieste riceve una richiesta URL dal motore servlet, invia la richiesta al gestore degli adattatori che, a sua volta, esegue una query per scegliere il tipo di adattatore in grado di elaborare la richiesta. Una volta individuato l'adattatore, la richiesta viene inviata all'adattatore.

Una volta inizializzato, il servlet delle richieste è in grado di leggere il file di configurazione *nome_istanza*.xml. Uno dei blocchi di configurazione nel file XML definisce tutti gli adattatori. Il metodo init() del servlet di richiesta inizializza tutti gli adattatori definiti.

Il listener MQSeries riceve messaggi MQSeries basati su XML da programmi remoti e distribuisce le richieste a gestori degli adattatori non HTTP.

Lo scheduler del processo non richiede un listener di protocollo.

Adattatori

Gli adattatori di WebSphere Commerce sono componenti specifici del dispositivo che eseguono funzioni di elaborazione prima di inviare una richiesta al controller Web. Esempi di attività di elaborazione eseguite da un adattatore sono:

- Istruzioni a un controller Web per l'elaborazione di una richiesta con modalità specifiche del tipo di unità. Ad esempio, un adattatore di unità PvC (Pervasive Computing) può istruire il controller Web affinché ignori i controlli HTTPS nella richiesta originale.
- Conversione del formato del messaggio della richiesta in entrata in un set di proprietà che i comandi di WebSphere Commerce possono analizzare.
- Garantire la permanenza della sessione specifica dell'unità.

Il seguente diagramma mostra la gerarchia delle classi di implementazione relativa alla struttura dell'adattatore di WebSphere Commerce.

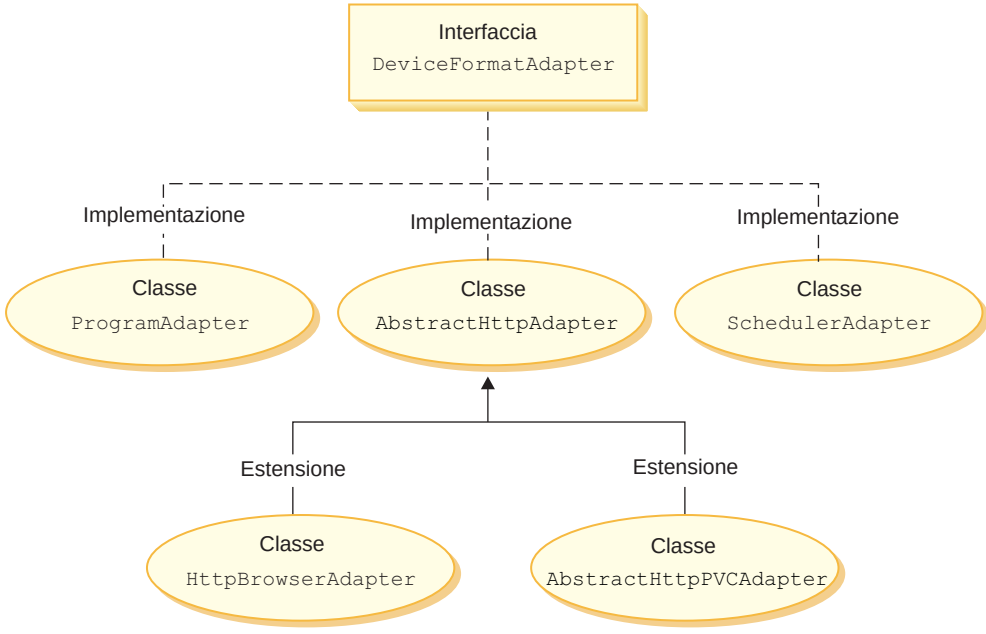


Figura 4.

Come mostrato nel precedente diagramma, tutti gli adattatori implementano l'interfaccia DeviceFormatAdapter. Di seguito vengono riportati gli adattatori utilizzati dall'ambiente di runtime di WebSphere Commerce.

Adattatore di programma
L'adattatore di programma fornisce il supporto per i programmi remoti richiamando i comandi di WebSphere Commerce. L'adattatore di programma riceve richieste e utilizza un programma di utilità per l'associazione dei messaggi per convertire la richiesta in un oggetto CommandProperty. Dopo aver eseguito la conversione, l'adattatore del programma utilizza l'oggetto CommandProperty ed esegue la richiesta.

Adattatore del programma di pianificazione
L'adattatore del programma di pianificazione fornisce il supporto per i comandi di WebSphere Commerce che vengono eseguiti come processi in background.

Adattatore browser HTTP
L'adattatore browser HTTP fornisce supporto per le richieste per richiamare i comandi di WebSphere Commerce ricevuti dai browser HTTP.

Adattatore PvC HTTP
Si tratta di una classe di adattatore astratta che può essere utilizzata

per sviluppare specifici adattatori di unità PvC. Ad esempio, se è necessario sviluppare un adattatore per una particolare applicazione di telefono cellulare, è possibile estendere questo adattatore.

Se necessario, è possibile estendere la struttura dell'adattatore in uno dei seguenti modi:

- Creare un adattatore per un'unità PvC specifica (ad esempio, creare una classe HttpIModePVCAdapterImpl per fornire il supporto alle unità i-mode). Un adattatore di questo tipo deve estendere la classe AbstractHttpAdapterImpl.
- Creare un nuovo adattatore che si colleghi a un nuovo listener di protocollo. Questo nuovo adattatore deve implementare l'interfaccia DeviceFormatAdapter.

Controller Web

Il controller Web di WebSphere Commerce è un contenitore di applicazione che segue un modello di progettazione simile a quello di un contenitore EJB. Tale contenitore semplifica il ruolo dei comandi garantendo servizi come la gestione della sessione (basata sulla permanenza della sessione stabilita dall'adattatore), il controllo della transazione, il controllo dell'accesso e l'autenticazione.

Inoltre, il controller Web partecipa al miglioramento del modello di programmazione per l'applicazione di commercio. Ad esempio, il modello di programmazione definisce i tipi di comando che un'applicazione deve scrivere. Ciascun tipo di comando ha un determinato obiettivo. La logica aziendale deve essere implementata nei comandi di controller e la logica di visualizzazione nei comandi di visualizzazione. Il controller Web si aspetta che il comando di controller restituisca un nome di visualizzazione. In caso tale nome non venga restituito, viene inviata un'eccezione.

Per le richieste HTTP, il controller Web effettua le seguenti attività:

- Avvia la transazione utilizzando l'interfaccia UserTransaction dal pacchetto javax.transaction.
- Richiama i dati di sessione dall'adattatore.
- Stabilisce se l'utente deve essere collegato prima di richiamare il comando. Se necessario, reindirizza il browser dell'utente a un URL di collegamento.
- Controlla se l'URL richiede un HTTPS sicuro. Se richiesto ma la richiesta corrente non utilizza HTTPS, reindirizza il browser Web a un URL HTTPS.
- Richiama il comando di controller e lo invia al contesto di comando e agli oggetti proprietà di immissione.
- Se si verifica un'eccezione rollback della transazione ed è possibile eseguire di nuovo il comando di controller, effettua questo tentativo.

- In genere, nel caso di un comando di visualizzazione da inviare nuovamente al client, il comando di controller restituisce un nome di visualizzazione. Il controller Web richiama il comando di visualizzazione per la visualizzazione corrispondente. Esistono diversi metodi per creare una visualizzazione di risposta, ad esempio il reindirizzamento a un URL diverso, l'inoltro a una maschera JSP o la scrittura di un documento HTML all'oggetto di risposta.
- Salva i dati di sessione.
- Esegue il commit dei dati di sessione.
- Esegue il commit della transazione corrente se eseguita correttamente.
- Esegue il rollback della transazione corrente in caso di errore (in base alle circostanze).

Comandi

I comandi di WebSphere Commerce sono bean che contengono la logica di programmazione associata alla gestione di una particolare richiesta.

Esistono quattro tipi principali di comando WebSphere Commerce:

Comando del controller
Un comando del controller contiene la logica relativa a un determinato processo aziendale. Esempi di comandi del controller comprendono il comando *OrderProcessCmd* per l'elaborazione degli ordini e il comando *UserRegistrationAddCmd* per la creazione di nuovi utenti registrati. In genere, un comando del controller contiene le istruzioni di controllo (ad esempio, if, then, else) e richiama i comandi di attività per l'esecuzione di determinate attività nel processo aziendale. Una volta completato, un comando del controller restituisce un nome visualizzazione. Quindi, il controller Web sceglie la classe di implementazione appropriata per il comando della visualizzazione e la esegue.

Comando di attività
Un comando di attività implementa una specifica unità della logica di applicazione. In generale, un comando del controller e una serie di comandi di attività implementano la logica dell'applicazione per una richiesta URL. Un comando di attività viene eseguito nello stesso contenitore del comando del controller.

Comando del bean di dati
Un comando del bean di dati viene richiamato da una maschera JSP quando viene eseguita l'istanza di un bean di dati. La funzione principale di un comando del bean di dati consiste nel riempire il bean con dei dati.

Comando di visualizzazione

Un comando di visualizzazione crea una visualizzazione come risposta a una richiesta client. Esistono tre tipi di comando di visualizzazione:

Comando di visualizzazione di reindirizzamento

Questo comando invia la visualizzazione tramite un protocollo di reindirizzamento, ad esempio il reindirizzamento di un URL. Un comando del controller dovrebbe restituire un comando di visualizzazione in questo tipo di visualizzazione, per restituire una visualizzazione tramite un protocollo di reindirizzamento. Quando si utilizza un protocollo di reindirizzamento, vengono modificati gli stack URL del browser. Quando viene immessa una chiave di ricarica, l'URL reindirizzato viene eseguito in sostituzione dell'URL originale.

Comando di visualizzazione diretto

Questo comando invia la visualizzazione di risposta direttamente al client.

Comando di visualizzazione di inoltro

Questo comando inoltra la richiesta di visualizzazione a un altro componente Web, ad esempio a una maschera JSP.

Esistono tre modi per richiamare un comando di visualizzazione:

- Un comando del controller specifica un nome di comando di visualizzazione una volta completata correttamente la richiesta.
- Un client richiede direttamente una visualizzazione.
- Un comando rileva un errore e decide che è necessario eseguire un'attività per elaborarlo. Il comando invia un'eccezione con un nome di comando di visualizzazione. Quando l'eccezione giunge al controller Web, viene eseguito il comando di visualizzazione dell'errore e viene restituita una risposta al client.

Bean entità di WebSphere Commerce

I bean entità sono gli oggetti permanenti, di tipo transazionale, forniti da WebSphere Commerce. Per coloro che conoscono l'ambito commerciale, i bean entità rappresentano i dati di WebSphere Commerce in modo intuitivo, ossia invece di analizzare lo schema del database, è possibile accedere ai dati da un bean entità che riproduce meglio i concetti e gli oggetti dell'ambito commerciale. È possibile estendere i bean entità esistenti. Inoltre, per alcuni requisiti aziendali specifici dell'applicazione, è possibile sviluppare dei bean entità completamente nuovi.

I bean entità vengono implementati in base al modello del componente EJB (Enterprise JavaBeans).

Per ulteriori informazioni sui bean entità, consultare “Implementazione dei bean entità di WebSphere Commerce” a pagina 49.

Bean di dati

I bean di dati Java sono bean utilizzati principalmente dai Web designer. In genere, forniscono accesso a un'entità di WebSphere Commerce. Un Web designer può inserire questi bean in una maschera JSP, in modo che le informazioni dinamiche vengano diffuse nella pagina al momento della visualizzazione. Il Web designer deve conoscere soltanto i dati che il bean è in grado di fornire e quelli che il bean richiede di immettere. Cercando di tenere separata la visualizzazione dalla logica aziendale, non è necessario che il Web designer conosca il funzionamento del bean.

Gestore dei bean di dati

Quando un bean di dati di WebSphere Commerce viene inserito in una maschera JSP utilizzando WebSphere Studio Page Designer, viene generata una riga di codice che popola il bean di dati, durante il runtime, richiamando il gestore dei bean di dati.

Di seguito è riportato un esempio di codice da Page Designer.

```
com.ibm.commerce.beans.DataBeanManager.activate(bean_di_dati, request)
```

Maschere JSP

Le maschere JSP sono servlet specializzati generalmente utilizzati per motivi di visualizzazione. Dopo aver completato una richiesta URL, il controller Web può richiamare un comando di visualizzazione che a sua volta richiama una maschera JSP. Un client può inoltre richiamare un modello JSP direttamente dal browser senza un comando associato. In questo caso, l'URL per la maschera JSP deve contenere nel proprio percorso il servlet delle richieste, in modo che tutti i bean di dati richiesti da una maschera JSP possano essere attivati all'interno di una singola transazione. Il servlet delle richieste può inoltrare una richiesta URL alla maschera JSP ed eseguire questa maschera all'interno di una singola transazione.

Il gestore dei bean di dati rifiuta qualunque URL per una maschera JSP che non contenga nel proprio percorso il servlet delle richieste. Per ulteriori informazioni sulla protezione delle maschere JSP e su altre risorse, consultare Capitolo 4, “Controllo accessi” a pagina 91.

File di configurazione *Nome_istanza*.xml

Il file di configurazione *Nome_istanza*.xml imposta le informazioni di configurazione per l'istanza. Viene letto al momento dell'inizializzazione del servlet delle richieste.

Riepilogo per una richiesta

Questa sezione fornisce un riepilogo del flusso di interazione tra i componenti quando si crea una risposta a una richiesta.

Dopo il diagramma, viene riportata una descrizione di ciascuna fase.

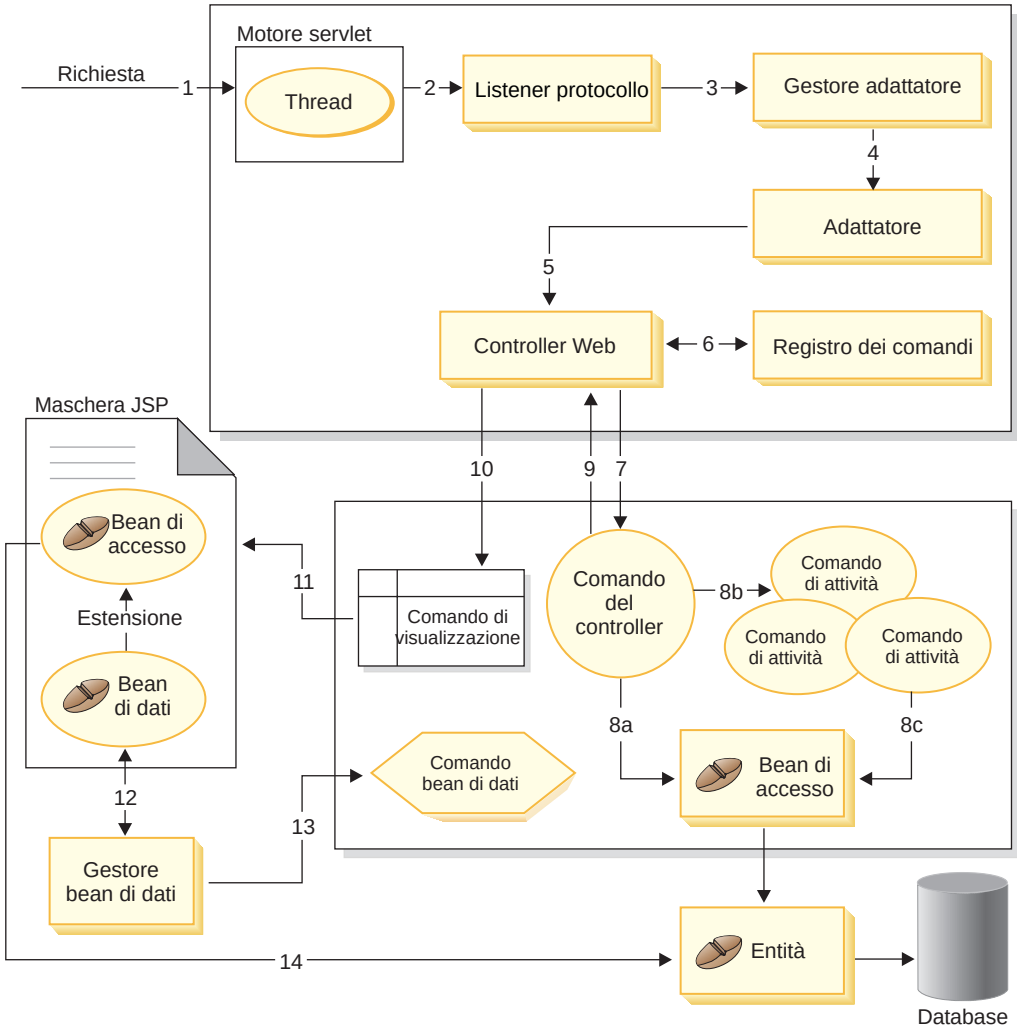


Figura 5.

Le informazioni seguenti si riferiscono al diagramma sopra riportato.

1. La richiesta viene indirizzata al motore servlet dal plug-in di WebSphere Application Server.

2. La richiesta viene eseguita nel thread a cui appartiene. Il motore del servlet distribuisce la richiesta al listener del protocollo. Il listener può essere un servlet di richiesta HTTP oppure MQ Listener.
3. Il listener del protocollo invia la richiesta al gestore degli adattatori.
4. Il gestore degli adattatori determina quale adattatore è in grado di gestire la richiesta, quindi la inoltra al relativo adattatore. Ad esempio, se la richiesta proviene da un browser Internet, il gestore degli adattatori la inoltra all'adattatore di un browser HTTP.
5. L'adattatore inoltra la richiesta al controller Web.
6. Il controller Web sceglie il comando da richiamare eseguendo una query il registro di comando.
7. Se la richiesta necessita di un comando di controller, il controller Web richiama il comando di controller appropriato.
8. Una volta che il comando di controller ha avviato l'esecuzione, sono disponibili le seguenti opzioni:
   a. Il comando di controller può accedere al database utilizzando un bean di accesso e il corrispondente bean entità.
   b. Il comando di controller può richiamare uno o più comandi di attività. Successivamente, i comandi di attività possono accedere al database utilizzando i bean di accesso e i corrispondenti bean entità (illustrati in 8(c)).
9. Una volta completato, il comando di controller restituisce un nome di visualizzazione.
10. Il controller Web cerca il nome di visualizzazione nella tabella VIEWREG. Richiama l'implementazione del comando di visualizzazione registrato per il tipo di unità del richiedente.
11. Il comando di visualizzazione invia la richiesta a una maschera JSP.
12. In una maschera JSP, un bean di dati è necessario per richiamare le informazioni dinamiche dal database. Il gestore dei bean di dati attiva il bean di dati.
13. Il gestore dei bean di dati richiama un comando del bean di dati, se necessario.
14. Il bean di accesso dal quale è stato esteso il bean di dati, accede al database utilizzando il corrispondente bean entità.

Differenze fondamentali tra la personalizzazione nei rilasci precedenti

A partire da WebSphere Commerce Suite Version 5.1, il WebSphere Commerce Server è stato interamente scritto in linguaggio Java. Pertanto, è necessario utilizzare Java per personalizzare le funzioni. Ciò risulta molto diverso dal modello che è stato utilizzato in WebSphere Commerce Suite, Versione 4.1 (e

versioni precedenti di Net.Commerce), in cui le macro C++ e Net.Data venivano utilizzate per le operazioni di personalizzazione.

La tabella seguente indica i cambiamenti principali.

	Versione 4.1 (e precedenti) in C++	**Versione 5.1 (e successive versioni) in Java**
Modelli di applicazione	Comandi	Comandi di controller
	Attività	Comandi di attività
	Funzioni sostituibili	Comandi di attività
	Attività di visualizzazione	Comandi di visualizzazione
	Attività di errore	Comandi di visualizzazione
Modelli di visualizzazione	Macro Net.Data	Maschere JSP, bean di dati, comandi del bean di dati e comandi di visualizzazione
Modelli di permanenza	Net.Data e ODBC	Bean entità
Dispatcher URL	Dispatcher URL C++ di WebSphere Commerce	Motore servlet di WebSphere
Modelli di attività	Processi di sistema	Thread Java
Adattatore comandi	Nessuno	Controller Web e adattatori

In questo manuale non viene descritto il processo di migrazione. Per ulteriori informazioni sulla migrazione, fare riferimento al manuale *WebSphere Commerce Guida alla migrazione*.

Parte 2. Modello di programmazione

Capitolo 2. Modelli di progettazione

Per sviluppare la struttura di WebSphere Commerce, è possibile utilizzare vari modelli e meccanismi di progettazione. WebSphere Commerce fornisce un modello di progettazione di livello elevato che ciascuna applicazione WebSphere Commerce dovrebbe utilizzare. In questo capitolo, saranno analizzati i modelli di progettazione riportati di seguito:

- Modello di progettazione modello-visualizzazione-controller
- Modello di progettazione dei comandi
- Modello di progettazione delle visualizzazioni

Modello di progettazione modello-visualizzazione-controller

Il modello di progettazione modello-visualizzazione-controller (MVC) specifica che un'applicazione è formata da un modello di dati e da una serie di informazioni sulla presentazione e sul controllo. Questo modello prevede la separazione di ciascuno di questi elementi in diversi oggetti.

Il *modello* (ad esempio, le informazioni sui dati) contiene solo i dati di applicazione puri e non contiene alcuna logica per la descrizione delle modalità di presentazione dei dati a un utente.

La *visualizzazione* (ad esempio, le informazioni di presentazione) presenta i dati del modello all'utente. La visualizzazione conosce le modalità di accesso ai dati del modello ma non conosce il significato dei dati o il modo in cui l'utente li può utilizzare.

Infine, il *controller* (ad esempio, le informazioni per il controllo) si pone tra la visualizzazione e il modello. Il controller esamina gli eventi avviati da una visualizzazione (o da un'altra origine esterna) e risponde in maniera appropriata a tali eventi. In molti casi, l'azione di risposta consiste nel richiamare un metodo del modello. Poiché la visualizzazione e il modello sono collegati da un meccanismo di notifica, questa azione viene automaticamente riflessa nella visualizzazione.

Molte delle più recenti applicazioni seguono questo modello, alcune con lievi variazioni. Ad esempio, alcune applicazioni combinano la visualizzazione e il controller in una sola classe perché sono già strettamente legate. Tutte le variazioni tendono decisamente verso la separazione tra i dati e la relativa presentazione. In tal modo, non solo la struttura di un'applicazione viene semplificata ma è possibile anche riutilizzare il codice.

Poiché esistono già molti manuali che descrivono questo modello, nonché molti esempi, questo manuale non si soffermerà sui dettagli del modello.

Il seguente diagramma mostra come il modello di progettazione MVC si applica a WebSphere Commerce.

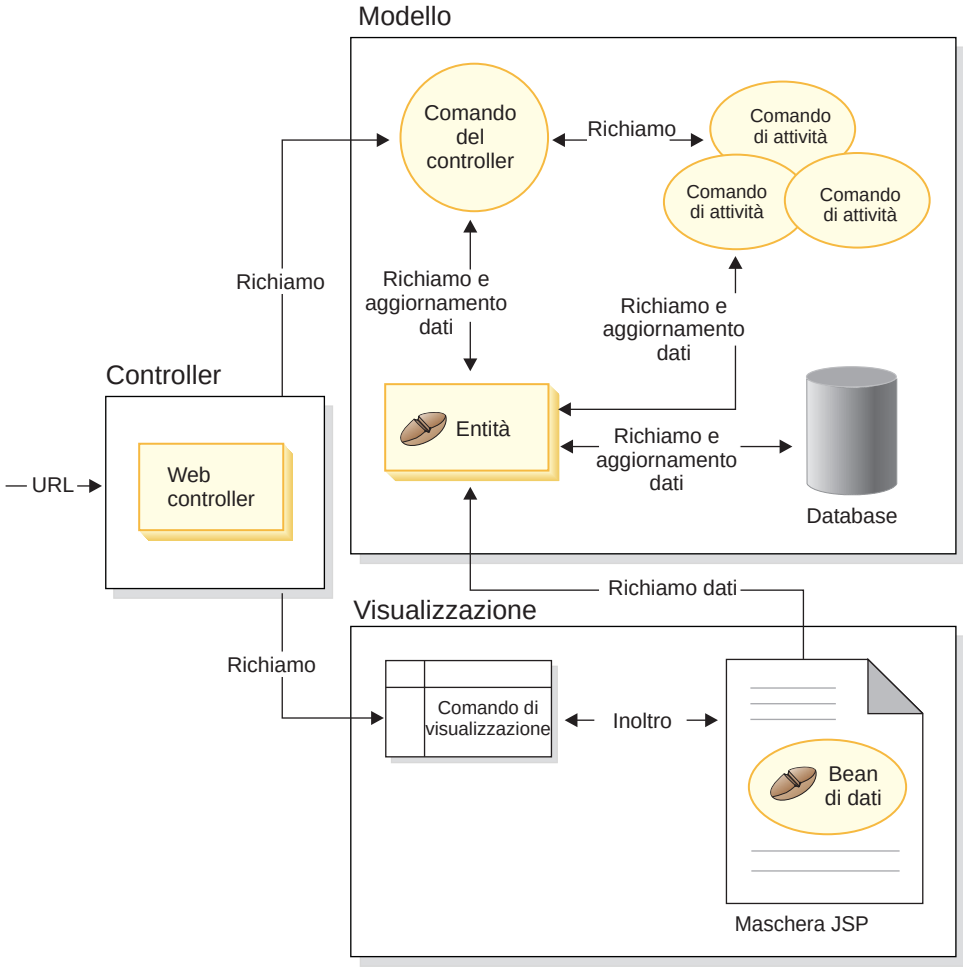


Figura 6.

Modello di progettazione dei comandi

WebSphere Commerce Server accetta richieste da applicazioni client thin basate su browser, nonché da altre applicazioni remote. Ad esempio, è possibile che una richiesta venga inoltrata da un sistema di acquisizione remoto oppure da un altro server commerce.

I vari formati delle richieste vengono convertiti in un formato comune dagli adattatori che costituiscono la struttura di adattatori. Una volta eseguita questa conversione, le richieste possono essere interpretate dai comandi di WebSphere Commerce.

I comandi sono bean che eseguono la logica aziendale. Essi rappresentano la logica procedurale sotto forma di attività di logica di elaborazione di livello elevato o di logica aziendale discreta. Un comando basato sull'elaborazione agisce come controller che coinvolge varie entità e altri comandi, mentre un comando di attività esegue una specifica attività e può accedere a un singolo oggetto.

Struttura del comando

I bean del comando seguono un modello di progettazione specifico. Ciascun comando comprende una classe di interfaccia (ad esempio `CategoryDisplayCmd`) e una classe di implementazione (ad esempio `CategoryDisplayCmdImpl`). Dal punto di vista del chiamante, la logica di richiamo prevede l'impostazione delle proprietà di immissione, l'utilizzo del metodo `execute()` e il recupero delle proprietà di emissione.

Dal punto di vista del programma di implementazione dei comandi, questi seguono la struttura dei comandi di WebSphere, che implementa il modello di progettazione comandi standard, consentendo la creazione di un livello di riferimenti indiretti tra il chiamante e l'implementazione. I meccanismi chiave tra questi livelli di riferimenti indiretti includono:

1. La possibilità di richiamare il gestore delle politiche del controllo accessi che consente di stabilire se l'utente è autorizzato a richiamare il comando.
2. La possibilità di eseguire un'implementazione del comando diversa per ciascun negozio in base all'identificativo del negozio.
3. La possibilità di eseguire un'implementazione delle visualizzazioni diversa basata sul tipo di unità del richiedente.

Il seguente diagramma mostra una panoramica concettuale delle interfacce per i quattro tipi principali di comando:

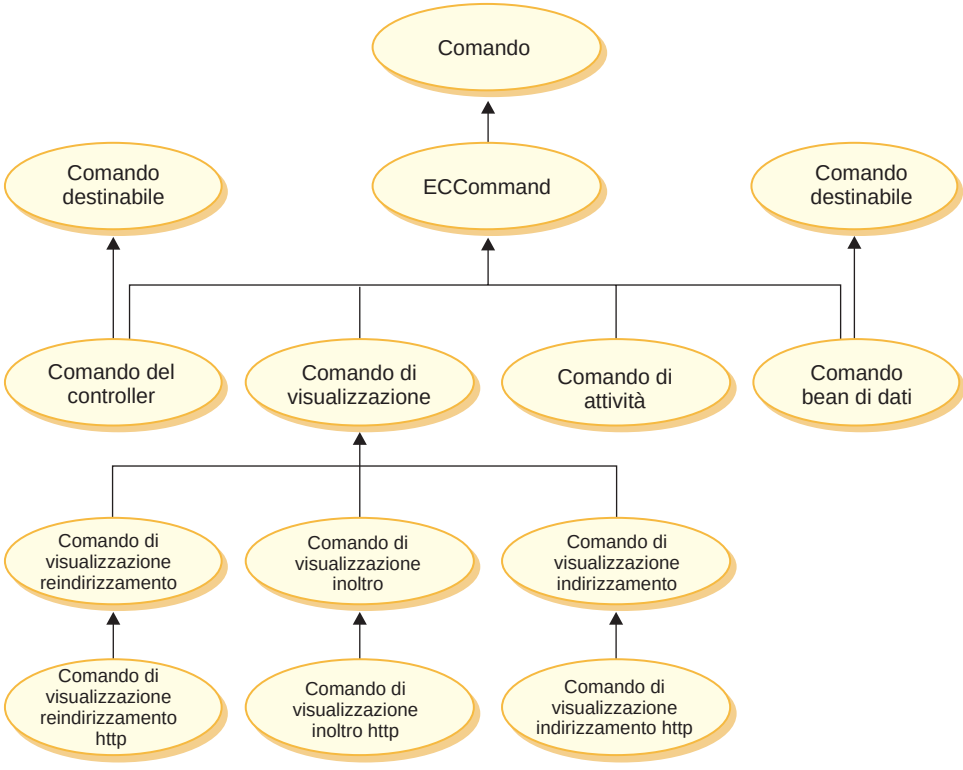


Figura 7.

Comando di controller

Un comando di controller contiene la logica relativa a un determinato processo aziendale. Esempi di comandi di controller comprendono il comando *OrderProcessCmd* per l'elaborazione degli ordini e il comando *UserRegistrationAddCmd* per la creazione di nuovi utenti registrati. In generale, un comando di controller contiene le istruzioni di controllo (ad esempio, if, then, else) e richiama i comandi di attività per l'esecuzione di determinate attività nel processo aziendale. Una volta completato, un comando di controller restituisce un nome visualizzazione. Quindi, il controller Web sceglie la classe di implementazione appropriata per l'attività della visualizzazione e la esegue.

Anche se un comando di controller è un comando indirizzabile, viene supportata solo una destinazione locale.

Comando di attività

Un comando di attività implementa una specifica unità della logica di applicazione. In generale, un comando di controller e una serie di

comandi di attività implementano la logica dell'applicazione per una richiesta URL. Un comando di attività viene eseguito nello stesso contenitore del comando di controller.

Comando del bean di dati

Un comando del bean di dati viene richiamato da una pagina JSP quando viene eseguita l'istanza di un bean di dati. La funzione principale di un comando del bean di dati consiste nel riempire i campi del bean di dati.

Anche se un comando dei dati è un comando indirizzabile, viene supportata solo una destinazione locale.

Comando di visualizzazione

Un comando di visualizzazione crea una visualizzazione come risposta a una richiesta client. Esistono tre modi per richiamare un comando di visualizzazione:

- Un comando di controller specifica un nome di comando di visualizzazione dopo il completamento della richiesta
- Un client può richiedere direttamente una visualizzazione.
- Un comando di controller o di attività rileva un errore e decide che è necessario eseguire un'attività di errore per elaborarlo, quindi invia un'eccezione con un nome comando di visualizzazione. Quando l'eccezione giunge al controller Web, viene eseguito il comando di visualizzazione e viene restituita una risposta al client.

Esistono tre tipi di comando di visualizzazione:

Comando di visualizzazione di reindirizzamento

Questo comando invia la visualizzazione tramite un protocollo di reindirizzamento, per esempio il reindirizzamento di un URL. Quando si richiede un protocollo di reindirizzamento, un comando di controller dovrebbe restituire un comando di visualizzazione di questo tipo. Quando si utilizza un protocollo di reindirizzamento, vengono modificati gli stack URL del browser. Quando viene immessa una chiave di ricarica, l'URL reindirizzato viene eseguito in sostituzione dell'URL originale.

Comando di visualizzazione diretto

Questo comando invia la visualizzazione di risposta direttamente al client.

Comando di visualizzazione di inoltro

Questo comando inoltra la richiesta di visualizzazione a un altro componente Web, ad esempio a una maschera JSP.

Comando Factory

Per creare nuovi oggetti comando, il chiamante del comando utilizza il *comando factory*. Il comando factory è un bean utilizzato per eseguire l'istanza dei comandi. Si basa sul modello di progettazione di factory che separa l'esecuzione dell'istanza di un oggetto da una classe di richiamo e la assegna alla classe factory che è in grado di individuare la classe di implementazione di cui eseguire l'istanza.

Factory fornisce un metodo efficace per eseguire l'istanza di nuovi oggetti. In tal caso, il comando factory consente di determinare la classe di implementazione corretta durante la creazione di un nuovo oggetto comando, in base ai singoli negozi. Il nome dell'interfaccia comando e l'identificativo del singolo negozio vengono inviati al nuovo oggetto comando, in base all'istanza.

Esistono due modi per specificare la classe di implementazione di un comando. È possibile specificare una classe di implementazione predefinita direttamente nel codice per l'interfaccia del comando utilizzando la variabile `defaultCommandClassName`. Ad esempio, il codice riportato di seguito consente di uscire dall'interfaccia CategoryDisplayCmd:

```
String defaultCommandClassName =
     "com.ibm.commerce.catalog.commands.CategoryDisplayCmdImpl"
```

Un altro modo per specificare la classe di implementazione consiste nell'utilizzo del registro comandi di WebSphere Commerce. Il registro comandi deve essere sempre utilizzato quando la classe di implementazione varia in base al negozio. Ulteriori informazioni sul registro comandi sono disponibili alla pagina 28.

Nel caso in cui una classe di implementazione predefinita venga specificata nel codice per l'interfaccia e una diversa classe di implementazione viene specificata nel registro comandi, il registro comandi ha la precedenza.

La sintassi per l'utilizzo del comando factory è la seguente:

```
cmd = CommandFactory.createCommand(Nomeinterfaccia, commandContext.getStoreId())
```

dove *Nomeinterfaccia* è il nome dell'interfaccia per il nuovo bean dei comandi e il metodo getStoreId determina il negozio per il quale utilizzare il comando.

Nota: La sintassi per l'utilizzo dei comandi factory per la creazione dei comandi delle politiche aziendali è differente dalla precedente sezione di codice. Per ulteriori informazioni sui comandi factory per la creazione dei comandi delle politiche aziendali, fare riferimento a "Richiamo della nuova politica aziendale" a pagina 180.

Flusso dei comandi

In questa sezione viene fornita una panoramica sul flusso logico che intercorre tra i comandi e il database di WebSphere Commerce. Il diagramma e le descrizioni seguenti illustrano il flusso.

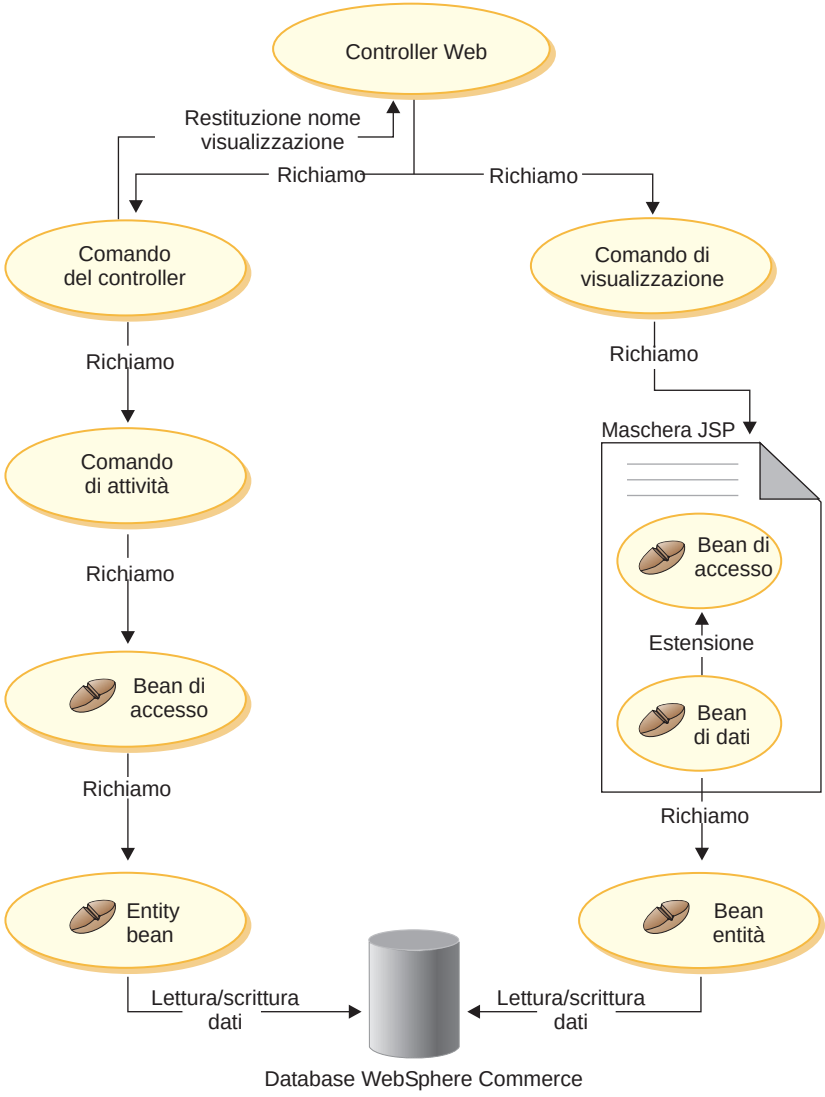


Figura 8.

Quando il controller Web riceve una richiesta, stabilisce se è necessario richiamare un comando di controller o un comando di visualizzazione. In entrambi i casi, il controller Web sceglie anche la classe di implementazione per il comando e la richiama.

Si analizzi innanzitutto la parte sinistra del diagramma. Poiché i comandi di controller comprendono la logica per un processo aziendale, richiamano spesso i comandi delle singole attività per eseguire specifiche unità di lavoro nel processo aziendale. I bean di accesso vengono utilizzati per richiamare o aggiornare le informazioni del database. Sia il comando di attività che il comando di controller possono richiamare il bean di accesso. Le richieste quindi passano dai bean di accesso ai bean entità che possono effettuare operazioni di lettura e scrittura all'interno del database di WebSphere Commerce.

Si osservi adesso la parte destra del diagramma. Un comando di visualizzazione viene richiamato dal controller Web quando un comando di controller completa l'elaborazione e restituisce il nome del comando di visualizzazione da richiamare oppure quando si verifica un errore e deve essere riportata la relativa visualizzazione.

Di solito, i comandi di visualizzazione richiamano una maschera JSP per visualizzare la risposta al client. Nella maschera JSP, i bean di dati vengono utilizzati per inserire le informazioni dinamiche nella pagina. Il gestore dei bean di dati attiva i bean di dati. Il bean di dati (che si estende dal bean di accesso) richiama il corrispondente bean entità. Se vi si accede indirettamente da una maschera JSP, un bean entità di solito richiama le informazioni dal database piuttosto che scrivere le informazioni nel database.

Struttura del registro comandi

I comandi di attività e di controller di WebSphere Commerce vengono registrati nel registro comandi. Il registro comandi è formato dalle seguenti tre tabelle:

- URLREG
- CMDREG
- VIEWREG

Nota: Business Le informazioni contenute in questa sezione non sono valide per la registrazione dei comandi delle politiche aziendali. Per ulteriori informazioni sulla registrazione di nuovi comandi delle politiche aziendali, consultare "Registrazione della nuova politica aziendale e del relativo comando" a pagina 164.

Tabella URLREG

La tabella URLREG associa gli URI (Universal Resource Indicator) alle interfacce del comando di controller. Gli URI forniscono un meccanismo semplice e flessibile per l'identificazione della risorsa. Un identificativo URI è una stringa relativamente breve di caratteri utilizzata per identificare una

risorsa astratta o fisica. In WebSphere Commerce, l'URI contiene solo informazioni relative al comando. Nel seguente URL, la sezione URI viene riportata in grassetto:

```
http://nomehost/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=Id_negozio&catalogId=Id_catalogo&langId=-1
```

Anche se la corrispondenza tra un URI e un nome di interfaccia è diretta, ciascun negozio può specificare se il comando richiede HTTPS o AUTHENTICATION. Per ciascuna richiesta URL in entrata, il controller Web cerca il nome dell'interfaccia per il comando di controller e utilizza il nome individuato per determinare la classe di implementazione appropriata, come registrato nella tabella CMDREG.

La seguente tabella descrive le informazioni contenute nella tabella di database URLREG.

Nome colonna	Descrizione	Commenti
URL	Nome URI	Ad esempio `MyNewCommand` o `ProductDisplay`
STOREENT_ID	Identificativo entità negozio	Questo valore può essere impostato su `0` per utilizzare il comando per tutti i negozi oppure su un identificativo del negozio univoco per indicare che il comando viene utilizzato solo per un determinato negozio.
INTERFACENAME	Nome interfaccia comando di controller	Ad esempio, `com.ibm.commerce.catalog.commands.GetProductDisplay.TemplateCmd`
HTTPS	HTTP sicuro necessario per questa richiesta di URL	Utilizzare `1` quando HTTPS è obbligatorio e `0` quando non lo è.
DESCRIPTION	Descrizione dell'URI	Ad esempio, `Questo comando viene utilizzato per scopi di verifica.`
AUTHENTICATED	Un registro utente è obbligatorio per questa richiesta URL	Utilizzare `1` quando l'autenticazione è obbligatoria e `0` quando non lo è.
INTERNAL	Indica se il comando è interno o esterno a WebSphere Commerce	Utilizzare `1` quando il comando è interno e `0` quando è esterno.

Quando un controller Web riceve una richiesta URL, richiama il nome dell'interfaccia relativo al comando di controller richiesto e lo utilizza per cercare il nome della classe di implementazione dalla tabella CMDREG. Stabilisce anche se è necessario un HTTPS per la richiesta URL controllando la colonna HTTPS nella tabella URLREG.

Solo i comandi richiamati tramite le richieste URL devono essere registrati nella tabella URLREG. Pertanto, in questa tabella devono essere registrati solo i comandi di controller e non i comandi di attività o di visualizzazione.

La seguente istruzione SQL crea una voce per MyNewControllerCommand utilizzata da un determinato negozio (il cui identificativo è 5):

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCommand',
5,'com.ibm.commerce.commands.MyNewControllerCommand',0,
'This is a test command.',null)
```

La sintassi generale per l'istruzione insert è la seguente

```
insert into nome_tabella (nome_colonna1,nome_colonna2, ... ,nome_colonna)
values (valore_colonna1,valore_colonna2,...,valore_colonna)
```

I valori della stringa devono essere racchiusi tra virgolette.

Tabella CMDREG

CMDREG è la tabella di registrazione del comando. Questa tabella fornisce il meccanismo per la creazione di corrispondenze tra l'interfaccia del comando e la relativa classe di implementazione. Più implementazioni di un'interfaccia consentono di personalizzare il comando per ciascun negozio.

Solo i comandi di controller e di attività vengono registrati nella tabella CMDREG. I comandi di visualizzazione vengono registrati nella tabella VIEWREG.

La seguente tabella descrive le informazioni contenute nella tabella di database CMDREG.

Nome colonna	Descrizione	Commenti
STOREENT_ID	Identificativo entità negozio	Questo valore può essere impostato su 0 per utilizzare il comando per tutti i negozi oppure su un identificativo del negozio univoco per indicare che il comando viene utilizzato solo per un determinato negozio.

Nome colonna	Descrizione	Commenti
INTERFACENAME	Nome interfaccia comando	Definisce l'interfaccia; utilizzare lo stesso nome utilizzato nella tabella URLREG.
DESCRIPTION	Descrizione di questo comando	Ad esempio, `Questo comando viene utilizzato per scopi di verifica`.
CLASSNAME	Nome della classe di implementazione del comando	Di solito, il nome dell'interfaccia che termina con ″Impl″.
PROPERTIES	Coppia predefinita nome-valore impostata come proprietà di immissione al comando	Il formato è lo stesso della stringa di richiesta dell'URL. Ad esempio, ″parm1=val1&parm2=val2″
LASTUPDATE	Ultimo aggiornamento a questa voce di comando	
TARGET	Nome destinazione del comando. Posizione in cui verrà eseguito il comando.	Sono supportate solo destinazioni locali.

In generale, quando si crea un nuovo comando di controller o di attività, è necessario creare la voce corrispondente nella tabella CMDREG. Ad esempio, la seguente istruzione SQL crea una voce per MyNewCommand utilizzata da un determinato negozio (il cui identificativo è 5):

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION, CLASSNAME,
PROPERTIES, LASTUPDATE, TARGET) values
(5,'com.ibm.commerce.catalog.commands.MyNewCommand', 'This is a test
command', 'com.ibm.commerce.catalog.commands.MyNewCommandImpl',
'myDefaulParm1=myDefaultVal1', '0000-12-01', 'Local')
```

La sintassi generale per l'istruzione insert è la seguente

```
 insert into nome_tabella (nome_colonna1,nome_colonna2, ... ,nome_colonna)
values (valore_colonna1,valore_colonna2,...,valore_colonna)
```

I valori della stringa devono essere racchiusi tra virgolette.

Se il comando che si sta scrivendo utilizza sempre la stessa classe di implementazione, non è necessario registrare il comando nella tabella CMDREG. In tal caso, è possibile utilizzare l'attributo defaultCommandClassName nell'interfaccia per specificare la classe di implementazione. Ad esempio, nel codice per l'interfaccia, includere la seguente stringa:

```
String defaultCommandClassName =
     "com.ibm.commerce.command.MyNewCommandImpl"
```

Se si specifica la classe di implementazione in questo modo, non è possibile inviare le proprietà predefinite alla classe di implementazione ed è necessario utilizzare la stessa classe di implementazione per tutti i negozi.

Esempio di un comando di controller registrato

Si supponga di avere nel proprio sito due negozi, il NegozioA e il NegozioB. Ciascun negozio presenta requisiti di sicurezza differenti per il comando di controller MyUrl, nonché differenti implementazioni del comando. Questa sezione illustra come utilizzare il registro comandi per abilitare questa personalizzazione.

La seguente tabella mostra le voci relative al NegozioA e al NegozioB nella tabella URLREG:

Nome colonna	Voce del NegozioA	Voce del NegozioB
URL	MyUrl	MyUrl
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce. mycommands.myUrl	com.ibm.commerce. mycommands.myUrl
HTTPS	1	1
DESCRIPTION	Voce di esempio nella tabella URLREG.	Voce di esempio nella tabella URLREG.
AUTHENTICATED	1	0
INTERNAL	null	null

Nota: Gli spazi nei valori di INTERFACENAME sono stati inseriti unicamente per motivi di visualizzazione. In realtà, ciascun valore è un'unica stringa senza spazi.

In base alle voci della tabella URLREG, il controller Web stabilisce che il nome dell'interfaccia per MyURL URI è com.ibm.commerce.mycommands.MyUrl. Inoltre, stabilisce che il NegozioA richiede l'esecuzione del comando utilizzando sia HTTPS che AUTHENTICATION, ma che il NegozioB richiede solo HTTPS. I valori di HTTPS e AUTHENTICATION vengono utilizzati dal controller Web ma non dall'interfaccia.

Il seguente diagramma illustra questo flusso:

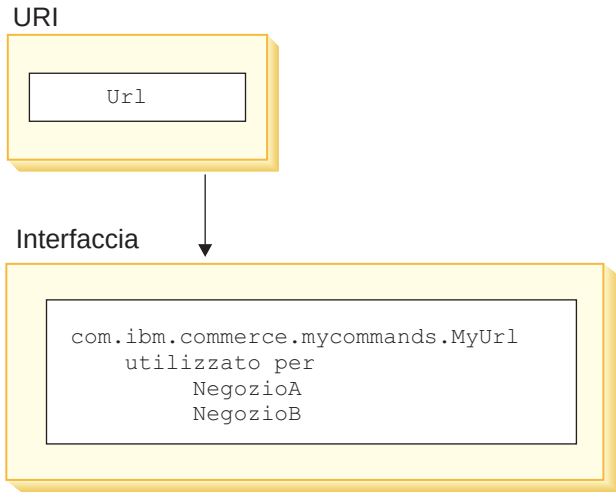


Figura 9.

La seguente tabella mostra le voci nella tabella CMDREG. Vengono visualizzate solo le colonne necessarie per questo esempio.

Nome colonna	Voce del NegozioA	Voce del NegozioB
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce. mycommands.myUrl	com.ibm.commerce. mycommands.myUrl
CLASSNAME	com.ibm.commerce. mycommands. myUrlStoreAImpl	com.ibm.commerce. mycommands.myUrlStoreBImpl

Nota: Gli spazi nei valori di INTERFACENAME e CLASSNAME sono stati inseriti unicamente per motivi di visualizzazione. In realtà, ciascun valore è un'unica stringa senza spazi.

In base alle voci della tabella CMDREG, il controller Web stabilisce che per il NegozioA la classe di implementazione per l'interfaccia com.ibm.commerce.mycommands.MyUrl è com.ibm.commerce.mycommands.MyUrlStoreAImpl. Inoltre, stabilisce che per il NegozioB la classe di implementazione per la stessa interfaccia è com.ibm.commerce.mycommands.MyUrlStoreBImpl. Il seguente diagramma illustra questo flusso:

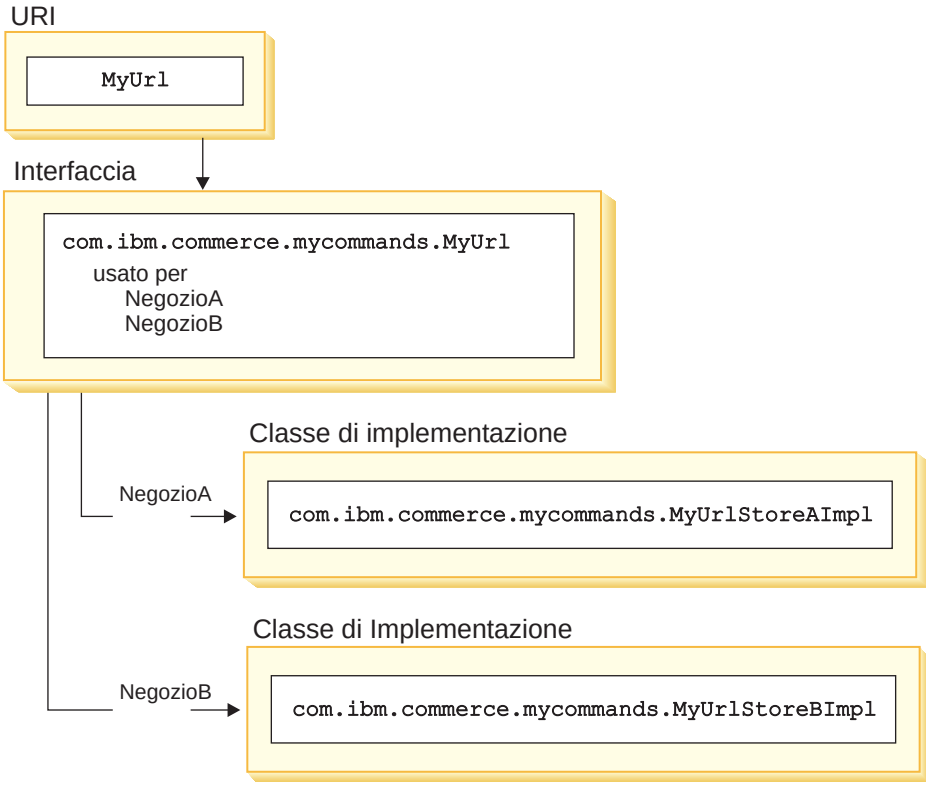


Figura 10.

Tabella VIEWREG

La tabella VIEWREG consente la registrazione delle implementazioni del comando di visualizzazione specifiche di un'unità. Utilizzando questa tabella, è possibile registrare più implementazioni di una visualizzazione. Quindi, la struttura dei comandi è in grado di restituire diverse visualizzazioni ai vari client.

Quando un nome di un comando di visualizzazione viene restituito da un comando del controller o specificato in un'eccezione, il controller Web sceglie la classe del comando di visualizzazione dalla tabella VIEWREG. È possibile associare più nomi di comando di visualizzazione alla stessa classe di implementazione.

Nome colonna	Descrizione	Commenti
VIEWNAME	Nome visualizzazione	Ad esempio, `AddressForm`

Nome colonna	Descrizione	Commenti
DEVICEFMT_ID	Identificativo tipo unità.	Le opzioni disponibili includono: • BROWSER (valore predefinito) • I_MODE • E-mail • MQXML • MQNC
STOREENT_ID	Identificativo entità negozio	Questo valore può essere impostato su `0` per utilizzare il comando per tutti i negozi oppure su un identificativo del negozio univoco per indicare che il comando viene utilizzato solo per un determinato negozio.
INTERFACENAME	Nome interfaccia comando di visualizzazione	Le opzioni predefinite sono ForwardView, DirectView e RedirectView.
CLASSNAME	Nome della classe di implementazione del comando di visualizzazione	È possibile utilizzare l'implementazione predefinita.
PROPERTIES	Coppia predefinita nome-valore impostata come proprietà di immissione al comando	Se viene visualizzata sempre la stessa pagina, impostare il nome file JSP in questa proprietà (`docname=`*`nome_jsp`*`.jsp`). Se viene utilizzata sempre la stessa maschera JSP per tutti i negozi, impostare `storeDir=no` per impedire di utilizzare una directory specifica di un negozio. Se un utente generico può utilizzare il comando, impostare `isGeneric=true`.
DESCRIPTION	Descrizione di questo comando	
HTTPS	HTTP sicuro necessario per questa richiesta di URL	Utilizzare `1` quando HTTPS è obbligatorio e `0` quando non lo è.
LASTUPDATE	Ultimo aggiornamento di questa voce	

Nome colonna	Descrizione	Commenti
INTERNAL	Indica se il comando è interno o esterno a WebSphere Commerce	Utilizzare 1 quando il comando è interno e 0 quando è esterno.

Quando si crea un nuovo comando di visualizzazione, è possibile che sia necessario creare una voce corrispondente nella tabella VIEWREG. Se si verifica uno dei seguenti eventi, è necessario registrare il comando di visualizzazione nella tabella VIEWREG:

- Il comando di visualizzazione viene eseguito con il controllo di accesso
- Esistono più implementazioni del comando di visualizzazione
- Le proprietà vengono impostate nella colonna PROPERTIES

È possibile accedere ai comandi di visualizzazione registrati dal registro comandi utilizzando il nome di visualizzazione o direttamente utilizzando il nome file di visualizzazione. È possibile accedere alle visualizzazioni non registrate nella tabella VIEWREG solo quando un client utilizza il nome file di visualizzazione.

Si supponga, ad esempio, che una visualizzazione denominata *MyView* presenti la seguente voce VIEWREG:

Nome colonna	Voce
VIEWNAME	MyView
DEVICEFMT_ID	BROWSER
STOREENT_ID	0
INTERFACENAME	com.ibm.commerce.commands.ForwardViewCommand
CLASSNAME	com.ibm.commerce.commands.HTTPForwardViewCommandImp
PROPERTIES	docname=MyView.jsp
DESCRIPTION	An example for calling a JSP template using either the view name or directly from a URL.
HTTPS	0
LASTUPDATE	2000–11–30
INTERNAL	0

Poiché MyView è una visualizzazione registrata, un client può accedere alla visualizzazione utilizzando il nome del comando oppure sostituendo il nome del file di visualizzazione attuale con il nome del comando. Utilizzando il nome di visualizzazione, un URL di esempio potrebbe essere:

`http://`*`nomehost`*`.com/webapp/wcs/stores/servlet/MyView`

e utilizzando il nome file, un URL di esempio potrebbe essere:
`http://nomehost.com/webapp/wcs/stores/servlet/MyView.jsp`

Se esiste una possibilità che un client richiami una visualizzazione registrata direttamente (utilizzando il nome file di visualizzazione), è necessario registrare il comando utilizzando il nome del file di visualizzazione per la visualizzazione, come mostrato in questo esempio (MyView e MyView.jsp).

Una visualizzazione non registrata nella tabella può essere richiamata solo utilizzando il nome file di visualizzazione. Quindi, se esiste una visualizzazione non registrata che utilizza il file MyUnregisteredView.jsp, l'URL di accesso a tale visualizzazione è il seguente:
`http://nomehost.com/webapp/wcs/stores/servlet/MyUnregisteredView.jsp`

La seguente istruzione SQL di esempio crea una voce per *MyNewViewCommand* utilizzata da un determinato negozio:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID,
INTERFACENAME, CLASSNAME, PROPERTIES, DESCRIPTION,HTTPS, LASTUPDATE,
INTERNAL) values ('MyNewViewCommand', 'BROWSER', 5,
'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commmerce.command.HttpForwardViewCommandImpl',
'docname=MyNewViewCommand.jsp', 'A test view command.', 0,
'0000-12-01', 0)
```

Di seguito viene riportato un altro esempio di tabella VIEWREG con le informazioni chiave:

COMMAND - NAME	DEVICE - FMT_ID	INTERFACE - NAME	CLASSNAME	PROPERTIES
ProductDisplayView	BROWSER	Forward View Command	HttpForwardView CommandImpl	
InterestItemAddView	BROWSER	Redirect View Command	HttpRedirectView CommandImpl	docname =item.jsp
InterestItemDeleteView	BROWSER	Redirect View Command	HttpRedirectView CommandImpl	docname =item.jsp
GenericApplication Error	BROWSER	Redirect View Command	HttpRedirectView CommandImpl	docname =usererr.jsp
GenericSystemError	BROWSER	Redirect View Command	HttpRedirectView CommandImpl	docname =syserr.jsp

COMMAND - NAME	DEVICE - FMT_ID	INTERFACE - NAME	CLASSNAME	PROPERTIES
Logon	BROWSER	Forward View Command	HttpForwardView CommandImpl	docname= logon.jsp & storeDir=no

Nota: Gli spazi nei valori di COMMANDNAME, INTERFACENAME, CLASSNAME e PROPERTIES sono stati inseriti unicamente per motivi di visualizzazione. In realtà, ciascun valore è un'unica stringa senza spazi. Anche i trattini nei nomi delle colonne sono stati inseriti solo per motivi di visualizzazione.

La tabella precedente illustra i seguenti scenari:

- Il nome di visualizzazione *ProductDisplayView* viene restituito al controller Web da un comando di controller; in questo caso, ProductDisplay. Il controller Web sceglie l'interfaccia del comando di visualizzazione e i nomi classe utilizzando il nome comando di visualizzazione ProductDisplayView e il relativo identificativo di unità. Un comando di visualizzazione può presentare diverse classi di implementazione per diversi negozi e identificativi di unità. Il nome interfaccia, tuttavia, deve essere lo stesso poiché esso definisce il tipo di comando di visualizzazione.
- I comandi InterestItemAdd e InterestItemDelete restituiscono al controller Web, rispettivamente, i nomi di visualizzazione *InterestItemAddView* e *InterestItemDeleteView*. Entrambi i comandi richiedono visualizzazioni di reindirizzamento; quindi, il nome interfaccia del comando di visualizzazione per entrambe le visualizzazioni è RedirectViewCommand. Esiste una maschera JSP comune registrata per entrambe le visualizzazioni. Il controller Web estrae le proprietà (docname=item.jsp) e le invia al comando di visualizzazione (HttpRedirectViewCommandImpl).
- Se un comando di controller o di attività invia un'eccezione ECApplication a causa di un parametro utente errato, possono verificarsi i seguenti eventi:
  - Se nel comando di controller è stata specificata una visualizzazione che deve essere richiamata quando si verifica un'eccezione dell'applicazione, la voce relativa a quella visualizzazione viene richiamata dalla tabella VIEWREG ed elaborata di conseguenza.
  - Se non viene specificata alcuna visualizzazione, viene richiamato il comando GenericApplicationError e viene visualizzata la maschera JSP registrata nel database. Utilizzando la tabella precedente come esempio, viene visualizzata la maschera `usererr.jsp`.
- Se un comando di controller o di attività invia un'eccezione ECSystem a causa di un'eccezione di sistema, possono verificarsi i seguenti eventi:

- Se nel comando di controller è stata specificata una visualizzazione che deve essere richiamata quando si verifica un'eccezione di sistema, la voce relativa a quella visualizzazione viene richiamata dalla tabella VIEWREG ed elaborata di conseguenza.
- Se non viene specificata alcuna visualizzazione, viene richiamato il comando GenericSystemError e viene visualizzata la maschera JSP registrata nel database. Utilizzando la tabella precedente come esempio, viene visualizzata la maschera `syserr.jsp`.

- Gli utenti del browser possono richiamare la pagina per il collegamento immettendo l'URL di collegamento. Poiché la proprietà storeDir è impostata su "no", le informazioni specifiche di un negozio non sono incluse nel percorso per la maschera JSP. Di conseguenza, viene visualizzata la stessa pagina di collegamento per i clienti di tutti i negozi.

Modello di progettazione di visualizzazione

Le pagine di visualizzazione restituiscono una risposta a un client. Di solito, le pagine di visualizzazione vengono implementate come maschere JSP (metodo consigliato); tuttavia, possono essere scritte direttamente come servlet.

Per supportare più tipi di unità, l'accesso URL a un comando di visualizzazione deve utilizzare il nome di visualizzazione e non il nome del file JSP.

La giustificazione logica principale per questo livello di riferimenti indiretti è che la maschera JSP rappresenta una visualizzazione. La capacità di selezionare la visualizzazione appropriata (ad esempio, in base alla locale, al tipo di unità o ad altri dati del contesto della richiesta) è di fondamentale importanza, soprattutto perché una singola richiesta spesso dispone di più possibili visualizzazioni. Si supponga, ad esempio, che due acquirenti richiedano la home page di un negozio; uno di essi utilizza un normale browser Web mentre l'altro un telefono cellulare. Ovviamente, non può essere visualizzata la stessa home page per entrambi. Il controller Web ha la responsabilità di accettare la richiesta, quindi in base alle informazioni contenute nella struttura del registro comandi, deve determinare la visualizzazione per ciascun acquirente.

Maschere JSP e bean di dati

Un bean di dati è un bean Java utilizzato in una maschera JSP per fornire un contenuto dinamico. In genere un bean di dati fornisce una rappresentazione semplice di bean entità di WebSphere Commerce. Il bean di dati contiene proprietà che possono essere richiamate da o impostate in un bean entità. Per questo motivo, un bean di dati semplifica l'attività di inserimento dei dati dinamici nelle maschere JSP.

Un bean di dati dispone di una classe *BeanInfo* che definisce le proprietà da utilizzare nella pagina di visualizzazione. La classe BeanInfo abilita anche l'utilizzo dei bean di dati in siti in varie lingue fornendo i nomi di proprietà in tutte le lingue supportate da WebSphere Commerce.

Un bean di dati viene attivato dalla seguente chiamata:

```
com.ibm.commerce.beans.DataBeanManager.activate(DataBean, HTTPServletRequest)
```

WebSphere Studio Page Designer genera automaticamente la precedente riga di codice quando si inserisce un bean di dati di WebSphere Commerce in una maschera JSP.

Durante lo sviluppo delle maschere JSP, gli sviluppatori di negozio devono considerare le proprietà del negozio e i problemi di attivazione delle varie lingue. Per ulteriori informazioni sulle attivazioni di varie lingue, consultare la guida in linea di WebSphere Commerce.

Considerazioni sulla sicurezza delle maschere JSP e dei bean di dati

Un codice particolare per l'utilizzo delle maschere JSP e dei bean di dati riduce al minimo la possibilità di accessi di utenti non autorizzati. L'inserimento, la cancellazione, l'aggiornamento di parti di istruzioni SQL dovrebbero essere eseguiti in fase di sviluppo. Utilizzare il parametro inserts per raccogliere le informazioni di immissione runtime.

Un esempio di utilizzo di un parametro insert per raccogliere le informazioni di immissione runtime viene descritto di seguito:

```
select * from Order where owner =?
```

Inoltre, non dovrebbero essere utilizzate stringhe di immissione per comporre l'istruzione SQL. Segue un esempio di utilizzo di una stringa di immissione:

```
select * from Order where owner = "input_string"
```

Tipi di bean di dati

Un bean di dati è un bean Java utilizzato principalmente per fornire dati dinamici nelle maschere JSP. Esistono due tipi di bean di dati: smart e command.

Per richiamare i propri dati, un bean di dati smart utilizza il metodo di *acquisizione lenta*. Questo tipo di bean di dati può garantire prestazioni migliori quando non è necessario estrarre tutti i dati dal bean di accesso, in quanto acquisisce solo i dati richiesti. I bean di dati smart che richiedono accesso al database devono estendersi dal bean di accesso per il bean entità corrispondente e implementare l'interfaccia com.ibm.commerce.SmartDataBean. Ad esempio, il bean di dati ProductData estende il bean di accesso ProductAccessBean che corrisponde al bean entità Product.

Alcuni bean di dati smart non richiedono l'accesso al database. Ad esempio, il bean di dati smart PropertyResource richiama i dati da un bundle di risorse piuttosto che dal database. Quando non è richiesto l'accesso al database, il bean di dati smart deve estendere la classe SmartDataBeanImpl.

Un bean di dati command utilizza un comando per richiamare i propri dati ed è più leggero. Il comando richiama tutti gli attributi del bean di dati contemporaneamente, anche se la maschera JSP non lo richiede. Di conseguenza, per le maschere JSP che utilizzano solo una parte degli attributi di un bean di dati, un bean di dati command può richiedere tempi troppo lunghi. Per le maschere JSP che richiedono gran parte o tutti gli attributi, il bean di dati command è molto conveniente.

I bean di dati command possono anche estendersi dai bean di accesso corrispondenti e implementare l'interfaccia com.ibm.commerce.CommandDataBean.

Interfacce del bean di dati

I bean di dati implementano uno o tutte le seguenti interfacce Java:

- com.ibm.commerce.SmartDataBean.
- com.ibm.commerce.CommandDataBean
- com.ibm.commerce.InputDataBean (facoltativa)

Ciascuna interfaccia Java descrive l'origine dati utilizzata per popolare il bean di dati. Implementando più interfacce, il bean di dati può accedere ai dati da varie origini. Di seguito vengono fornite ulteriori informazioni su ciascuna interfaccia.

Interfaccia SmartDataBean: Un bean di dati che implementa l'interfaccia SmartDataBean può richiamare i propri dati, senza un comando di bean di dati associato. Di solito, un bean di dati smart si estende da un bean di accesso di un bean entità corrispondente. Quando un bean di dati smart è attivo, il gestore dei bean di dati richiama il metodo di riempimento del bean di dati. Utilizzando il metodo di riempimento, il bean di dati può richiamare tutti gli attributi, tranne quelli degli oggetti associati. Ad esempio, se il bean di dati si estende da una classe di bean di accesso per un bean entità, il bean di dati richiama il metodo refreshCopyHelper. Tutti gli attributi del bean entità corrispondente vengono automaticamente riempiti nel bean di dati smart. Tuttavia, se il bean entità dispone di oggetti associati, gli attributi di questi oggetti non vengono richiamati. I principali vantaggi offerti dall'utilizzo dei bean di dati smart sono i seguenti:

- La procedura di implementazione è semplice e non è necessario scrivere un comando del bean di dati.
- Quando si aggiungono nuovi campi al bean entità, non è necessario apportare le modifiche al bean di dati. Dopo aver modificato il bean entità,

è necessario generare nuovamente il bean di accesso utilizzando gli strumenti in VisualAge per Java. Una volta rigenerato il bean di accesso, tutti i nuovi attributi sono automaticamente disponibili nel bean di dati smart.

- I bean entità spesso contengono attributi che rappresentano gli oggetti associati. Per motivi legati alle prestazioni, il bean di dati smart non richiama automaticamente tali attributi. È più opportuno, invece, eseguire la procedura di richiamo di tali attributi a quando richiesti, come mostrato nel seguente diagramma:

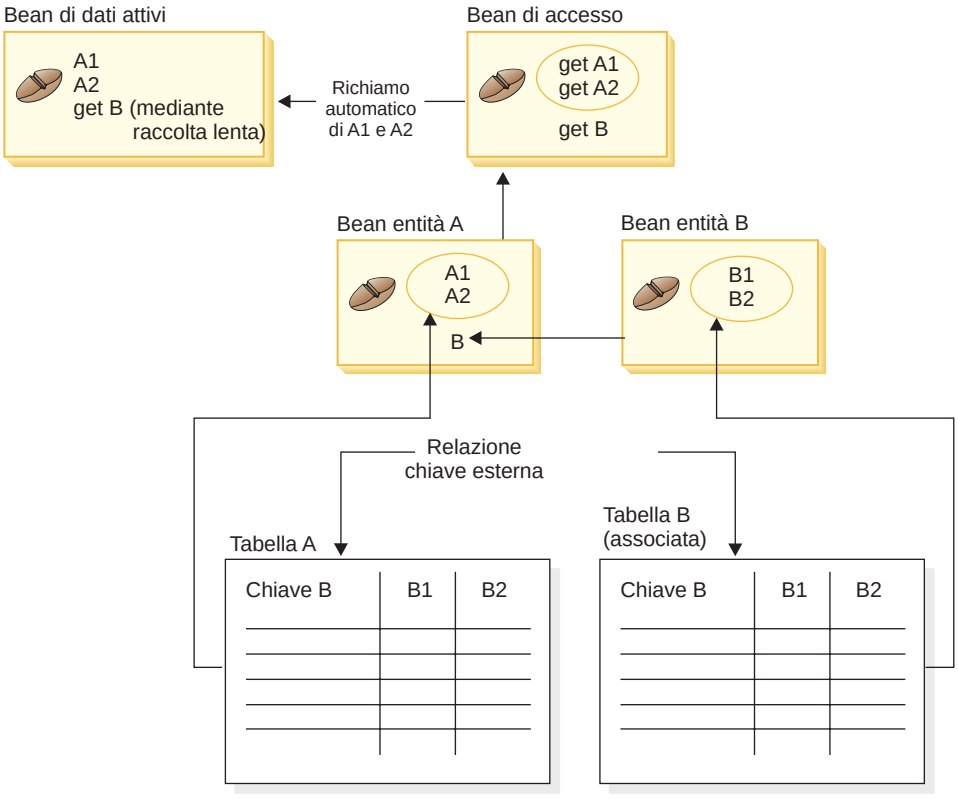


Figura 11.

Per ulteriori informazioni relative al richiamo tramite acquisizione lenta, fare riferimento alla sezione “Richiamo dei dati con acquisizione lenta” a pagina 44.

Interfaccia CommandDataBean: Un bean di dati che implementa l’interfaccia CommandDataBean richiama i dati da un comando dal bean di dati. Questo tipo di bean di dati è un oggetto leggero; per riempire i propri dati si affida a un comando del bean di dati. Il bean di dati deve implementare il metodo

getCommandInterfaceName() (come definito dall'interfaccia com.ibm.commerce.CommandDataBean) che restituisce il nome dell'interfaccia del comando dei bean di dati.

Interfaccia InputDataBean: Un bean di dati che implementa l'interfaccia InputDataBean richiama i dati dai parametri dell'URL o dagli attributi impostati dal comando di visualizzazione.

Gli attributi definiti in questa interfaccia possono essere utilizzati come campi di chiave principale per acquisire dati aggiuntivi. Quando viene richiamata una maschera JSP, il codice del servlet JSP generato riempie tutti gli attributi che corrispondono ai parametri dell'URL; quindi, attiva il bean di dati inviandolo al relativo gestore. Il gestore dei bean di dati richiama a sua volta il metodo setRequestProperties() del bean di dati (come definito dall'interfaccia com.ibm.commerce.InputDataBean) per inviare tutti gli attributi impostati dal comando di visualizzazione. WebSphere Studio genera il seguente codice per ciascun bean di dati inserito nelle pagine utilizzando Page Designer:

```
com.ibm.commerce.beans.DataBeanManager.activate(DataBean, HTTPServletRequest);
```

Classe BeanInfo

Un bean di dati non è completo senza una classe BeanInfo che implementi l'interfaccia java.lang.Object.BeanInfo. La classe BeanInfo viene utilizzata per fornire informazioni esplicite relative ai metodi e alle proprietà del bean di dati. Inoltre, consente di nascondere i metodi di runtime pubblici nella classe di implementazione del bean di dati dal Web designer oppure per impostare la stringa di visualizzazione appropriata per ciascuno attributo del bean di dati.

Per ulteriori informazioni sull'implementazione di una classe BeanInfo, fare riferimento alle specifiche JavaBeans di Sun Microsystems.

Attivazione del bean di dati

I bean di dati possono essere attivati utilizzando i metodi activate o silentActivate disponibili nella classe com.ibm.commerce.beans.DataBeanManager. Il metodo activate è un metodo di attivazione nel quale l'evento di attivazione viene eseguito con esito positivo solo se sono disponibili tutti gli attributi. Nel caso in cui uno solo degli attributi non sia disponibile, viene inviata un'eccezione per l'intero processo di attivazione.

Il metodo silentActivate non invia eccezioni quando non sono disponibili singoli attributi.

Richiamo dei comandi di controller da una maschera JSP

Talvolta, è necessario richiamare i comandi di controller da una maschera JSP: in questi casi, è possibile utilizzare `ControllerCommandInvokerDataBean`.

Utilizzando il bean di dati, è possibile specificare il nome interfaccia del comando da richiamare oppure è possibile impostare direttamente il nome del comando da richiamare. È anche possibile impostare le proprietà di richiesta per il comando.

Quando si attiva questo bean di dati mediante il gestore dei bean di dati, il comando di controller viene eseguito e le proprietà di risposta sono disponibili nella maschera JSP.

Dopo aver eseguito il comando di controller, è possibile eseguire la visualizzazione.

Richiamo dei dati con acquisizione lenta

Quando un bean di dati è attivo, è possibile riempirlo da un comando del bean di dati o dal metodo populate() del bean di dati. Gli attributi richiamati giungono al bean entità corrispondente al bean di dati. Un bean entità può anche disporre di oggetti associati che, a loro volta, dispongono di un dato numero di attributi.

Se, al momento dell'attivazione, gli attributi provenienti da tutti gli oggetti associati sono stati richiamati automaticamente, è possibile che si verifichi un problema delle prestazioni. Il livello delle prestazioni può diminuire con l'aumentare del numero degli oggetti associati.

Si supponga di disporre di un bean di dati del prodotto che contiene un elevato numero di vendite incrociate, vendite al rialzo o prodotti accessori (oggetti associati). È possibile riempire tutti gli oggetti non appena il bean di dati del prodotto viene attivato. Tuttavia, questa procedura di riempimento può richiedere varie query nel database. Nel caso in cui tutti gli attributi siano obbligatori per la pagina, le query nel database possono rivelarsi inefficaci.

In genere, non tutti gli attributi sono obbligatori per la pagina; quindi, un modello di progettazione migliore consente di eseguire un'acquisizione lenta, come illustrato di seguito:

```
getCrossSellProducts () {
   if (crossSellDataBeans == null)
     crossSellDataBeans= getCrossSellDataBeans();
   return crossSellDataBean;
    }
```

Impostazione degli attributi JSP - Panoramica

Il modello di programma WebSphere Commerce propone il modello di progettazione MVC. Quindi, la presentazione di un risultato di una richiesta di URL è separata dai comandi di controller e di attività. Tali comandi sono indipendenti dall'unità: implementano la logica aziendale e producono i dati da restituire al client senza disporre di informazioni sul client. Al contrario, un comando di visualizzazione è specifico dell'unità.

Sebbene i comandi di controller e di attività non compongano direttamente la visualizzazione, essi inviano le informazioni alla visualizzazione. È importante comprendere il modo in cui le informazioni vengono inviate alla visualizzazione. Il seguente diagramma illustra il modo in cui le proprietà passano tra il controller Web, il registro comandi, il comando di controller e il comando di visualizzazione.

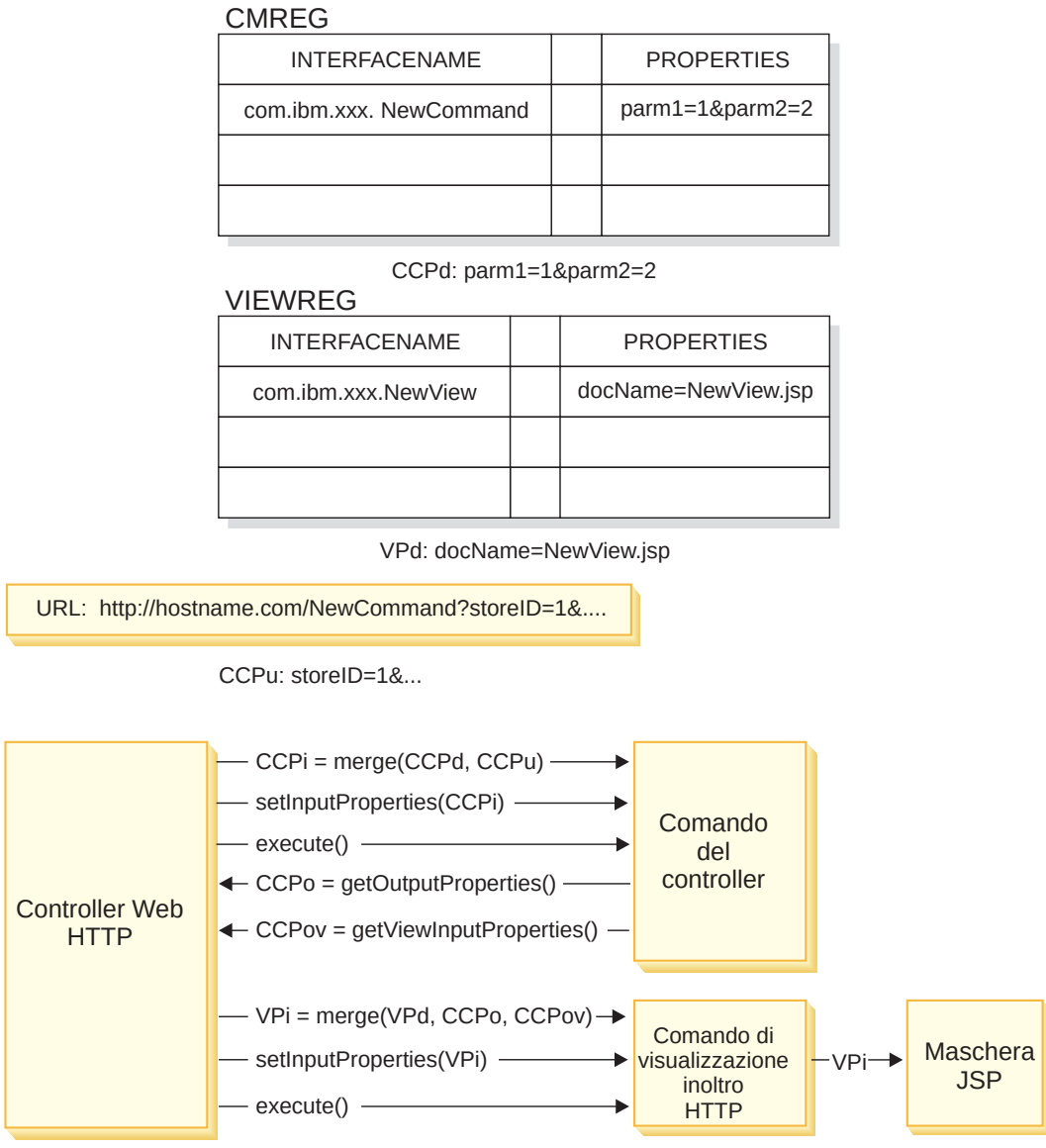


Figura 12.

Il diagramma precedente illustra le seguenti interazioni:

- Il controller Web unisce le proprietà di immissione dei parametri dell'URL (CCPu) e la voce della tabella CMDREG per il comando di controller (CCPd). Viene creato il CCPi.
- Il controller Web invia le proprietà unite (CCPi) al comando di controller ed esegue tale comando.

- Il comando di controller imposta le proprietà di emissione, come CCPo. Si tratta delle proprietà di emissione prodotte dal comando stesso. Una delle proprietà di emissione, viewCommandName, viene impostata sul nome comando di visualizzazione desiderato. Tali proprietà vengono richiamate dal controller Web utilizzando un metodo get.
- Il comando di controller imposta un'altra serie di proprietà di emissione, come CCPov. Per impostazione predefinita, tali proprietà vengono impostate sulle proprietà di immissione originali (CCPi). È possibile personalizzare tali proprietà. Ad esempio, è possibile che non sia necessario inviare tutti i parametri di immissione al comando di visualizzazione.
- Il controller Web unisce le tre serie di proprietà, CCPo, CCPov e VPd (le proprietà registrate nella tabella VIEWREG) alle proprietà per il comando di visualizzazione (VPi).
- Il controller Web imposta le proprietà unite, VPi, ed esegue il comando di visualizzazione.
- Il comando di visualizzazione imposta gli attributi nella maschera JSP dalle proprietà di immissione.

Durante la scrittura dei comandi, non è necessario eseguire esplicitamente la procedura di unione delle proprietà. Le classi di comando astratte includono un metodo mergeProperties. Per ulteriori informazioni su questo metodo, consultare la sezione "Riferimenti" della guida in linea di WebSphere Commerce.

Impostazione delle proprietà obbligatorie

Un comando di controller deve impostare le seguenti proprietà per ciascun tipo di comando di visualizzazione. Se le proprietà non vengono impostate dal comando, devono essere definite nella tabella VIEWREG.

- Se si utilizza il comando ForwardView, impostare `docname = nome_file_visualizzazione` dove *nome_file_visualizzazione* è il nome della maschera di visualizzazione. Ad esempio, `docname = productDisplay.jsp`.
- Se si utilizza il comando DirectView, eseguire una delle procedura riportate di seguito:
  - Impostare `textDocument = xxx` dove *xxx* è un oggetto java.io.InputStream che contiene il documento sotto forma di testo
  - Impostare `rawDocument = yyy` dove *yyy* è un oggetto java.io.InputStream che contiene il documento in forma binaria

  Se si utilizza il comando DirectView, è anche possibile impostare `contentType = ttt` dove *ttt* è il tipo di contenuto del documento
- Se si utilizza il comando RedirectView, impostare `url = uuu` dove *uuu* è l'URL di reindirizzamento.

Capitolo 3. Modello oggetti permanente

WebSphere Commerce utilizza una grande quantità di dati permanenti. Nello schema del database corrente esistono più di 520 tabelle definite; tuttavia, è possibile che si renda necessario estendere o personalizzare lo schema del database per soddisfare particolari esigenze aziendali.

WebSphere Commerce utilizza i bean entità che si basano sull'architettura del componente EJB (Enterprise JavaBeans) come livello di oggetto permanente. Questi bean entità rappresentano i dati di WebSphere Commerce in un modo da modellare i concetti e gli oggetti del dominio commerciale. Questo livello permanente fornisce una struttura estensibile.

VisualAge per Java, Enterprise Edition, fornisce sofisticati strumenti EJB e un ambiente di prova dell'unità che supporta lo sviluppo di questa struttura.

Implementazione dei bean entità di WebSphere Commerce

Bean entità di WebSphere Commerce - Panoramica

Come già specificato, il livello di permanenza dell'architettura di WebSphere Commerce viene implementato in base all'architettura del componente EJB. L'architettura EJB definisce due tipi di bean enterprise: i bean entità e i bean di sessione. I bean entità sono ulteriormente divisi in bean CMP (container-managed persistence) e bean BMP (bean-managed persistence).

Gran parte dei bean entità di WebSphere Commerce sono bean entità CMP. Un esiguo numero di bean di sessione privi di stato viene utilizzato per gestire le operazioni di database intensive, come la somma di tutte le righe di una determinata colonna. Uno dei vantaggi dell'utilizzo dei bean entità CMP è rappresentato dal fatto che gli sviluppatori possono utilizzare gli strumenti EJB disponibili in VisualAge per Java, Enterprise Edition. Questi strumenti consentono agli sviluppatori di definire gli oggetti Java e le relative corrispondenze della tabella di database. Gli strumenti generano automaticamente i programmi di permanenza per i bean entità. I programmi di permanenza sono oggetti Java che conservano campi Java nel database e popolano i campi Java con i dati del database.

VisualAge per Java fornisce due estensioni alle specifiche EJB correnti: eredità EJB e associazione. L'eredità EJB consente a un bean enterprise di ereditare proprietà, metodi e attributi del descrittore del controllo del livello del metodo da un altro bean enterprise che risiede nello stesso gruppo. Un'associazione è una relazione tra due bean entità CMP.

Alcuni dei bean entità di WebSphere Commerce utilizzano la funzione di eredità EJB. I bean entità di WebSphere Commerce non utilizzano la funzione delle associazioni fornita da VisualAge per Java. Durante lo sviluppo dei bean entità, non utilizzare la funzione di associazione di VisualAge per Java. In tal modo, il modello oggetti risulterà più semplice. Piuttosto che utilizzare la funzione di associazione fornita da VisualAge per Java, una relazione oggetto tra i bean enterprise può essere stabilita aggiungendo i metodi getter espliciti ai bean enterprise.

WebSphere Commerce fornisce due serie di bean enterprise: privati e pubblici. I bean enterprise privati vengono utilizzati dagli strumenti e dall'ambiente di runtime di WebSphere Commerce. *Non* utilizzare o modificare questi bean.

I bean enterprise pubblici, dall'altro lato, vengono utilizzati dalle applicazioni commerciali ed è possibile utilizzarli o estenderli. Questi bean sono organizzati nei seguenti gruppi EJB:

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction
- WCSCatalog
- WCSCommon
- WCSContract
- WCSCoupon
- WCSFulfillment
- WCSInventory
- WCSMessageExtensions
- WCSOrder
- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

Alcuni dei gruppi EJB elencati sopra contengono bean di sessione. Per semplificare la migrazione in futuro, non modificare la classe dei bean di sessione. Se necessario, è possibile creare un nuovo bean di sessione in un nuovo gruppo EJB. Per ulteriori informazioni sulla creazione di nuovi bean di sessione, fare riferimento a "Scrittura di nuovi bean di sessione" a pagina 77.

Descrittore di sviluppo per i bean enterprise di WebSphere Commerce

Un descrittore di sviluppo è una classe speciale serializzata che contiene impostazioni di runtime per un bean enterprise. I descrittori di sviluppo per i bean enterprise di WebSphere Commerce vengono impostati in un modo particolare e non possono essere modificati.

Quando si creano nuovi bean enterprise (bean entità o di sessione), impostare i descrittori di sviluppo in EJB Development Environment (di VisualAge per Java) facendo clic con il pulsante destro del mouse sul bean e selezionando **Proprietà**. I descrittori di sviluppo per i nuovi bean enterprise devono osservare le stesse convenzioni dei bean enterprise di WebSphere Commerce. In particolare, accertarsi di impostare gli attributi sui seguenti valori:

Attributo	Valore
Attributo di transazione	`TX_REQUIRED`
Livello isolamento	`TRANSACTION_READ_COMMITTED`
Modo Run-As	`SYSTEM_IDENTITY` o `CLIENT_IDENTITY`
Rientrante	Accertarsi che questa opzione non sia selezionata.

Spesso un bean enterprise contiene metodi che leggono soltanto le informazioni dal database ma non lo aggiornano mai. Si tratta di metodi di *sola lettura* e devono essere esplicitamente indicati come tali (fare clic con il pulsante destro del mouse sul metodo e selezionare **Attributi metodo EJB> Metodo di sola lettura**). In caso contrario, il contenitore EJB non tenta necessariamente di aggiornare il database alla fine della transazione e genera un errore di rollback nella transazione di sola lettura, compromettendo, così, le prestazioni.

Livelli di isolamento

Il livello di isolamento delle transazioni utilizzato per i bean enterprise di WebSphere Commerce all'interno di VisualAge per Java è `TRANSACTION_READ_COMMITTED`. Vi è una differenza tra l'implementazione di questo livello di isolamento per il driver JDBC per DB2 e il driver JDBC per Oracle. Per quanto concerne il livello di isolamento utilizzato durante la distribuzione nell'ambiente di WebSphere Application Server, questo varia a seconda del database utilizzato.

Il livello di isolamento utilizzato per i database DB2 quando la gestione avviene all'esterno del WebSphere Test Environment è `TRANSACTION_REPEATABLE_READ`. Il livello di isolamento utilizzato per i database Oracle quando la gestione avviene all'esterno del WebSphere Test Environment è `TRANSACTION_READ_COMMITTED`.

Se si esegue una distribuzione in un database DB2, non occorre modificare manualmente il livello di isolamento delle transazioni, in quanto viene modificato durante il processo di distribuzione quando si esegue il comando `modifyIsolationLevel`.

La seguente tabella riporta le associazioni tra i livelli di isolamento delle transazioni DB2 e Oracle e i corrispondenti livelli di isolamento delle transazioni JDBC.

JDBC	**DB2**	**Oracle**
Read Uncommitted	Uncommitted Read	Read Uncommitted
Read Committed	Cursor Stability	(Non applicabile)
Repeatable Read	Read Stability	Read Committed
Serializable	Repeatable Read	Serializable
Nessuno	(Non applicabile)	(Non applicabile)

Per quanto concerne il livello di isolamento Cursor Stability in DB2, verranno bloccate solo le righe che sono state aggiornate durante la transazione specificata. Se le colonne di una determinata riga non vengono aggiornate, anche se fanno parte di un gruppo di risultati restituito da un'istruzione SQL, il blocco applicato a questa riga specifica verrà rilasciato quando il cursore si sposterà su un'altra riga. In alcuni casi, come l'aggiornamento dell'inventario, un comportamento diverso risulterebbe più opportuno. Pertanto, modificando il livello di transazione in Read Stability in DB2, è possibile accrescere notevolmente il livello di integrità dei dati.

Se in Oracle è disponibile solo il livello Read Committed oppure l'equivalente JDBC, Repeatable Read, l'implementazione vera e propria viene eseguita in maniera molto simile a quella del livello Repeatable Read in DB2.

Estensione del modello oggetti di WebSphere Commerce

Il modello oggetti di WebSphere Commerce può essere esteso in uno dei seguenti modi:

- Estendere i bean enterprise pubblici di WebSphere Commerce.
- Scrivere un nuovo bean entità
- Scrivere un nuovo bean di sessione privo di stato

Nelle sezioni seguenti vengono forniti i dettagli relativi alle modalità di esecuzione di tali estensioni.

Metodologie di estensione del modello oggetti

I requisiti di applicazione possono comportare l'estensione del modello oggetti WebSphere Commerce esistente. Un esempio di un tale requisito è l'aggiunta di ulteriori attributi all'applicazione. Questa operazione può essere eseguita in uno dei seguenti modi:

Senza modificare un bean entità pubblico di WebSphere Commerce
: Creare una nuova tabella del database, quindi creare un nuovo bean entità per tale tabella. Aggiungere campi e metodi al bean entità per utilizzare il nuovo attributo, come necessario. Generare il codice di distribuzione e un bean di accesso per il nuovo bean entità. Quando l'applicazione richiede il nuovo attributo, avvia un oggetto bean di accesso e utilizza i relativi metodi per richiamare, impostare o utilizzare l'attributo.

Modificando un bean entità pubblico di WebSphere Commerce
: Creare una nuova tabella del database e creare un'unione di tabella tra la nuova tabella e quella esistente che corrisponde al bean enterprise esistente che viene modificato. Creare nuovi campi nel bean entità pubblico esistente di WebSphere Commerce e associare i campi e le corrispondenti colonne della nuova tabella utilizzando un'associazione di tabella secondaria. Aggiungere i metodi richiesti. Generare nuovamente il codice di distribuzione e il bean di accesso per il bean entità esistente. I nuovi attributi sono disponibili quando l'applicazione avvia l'oggetto bean di accesso.

È tuttavia necessario raggiungere un compromesso tra questi due approcci che, in genere riguarda il livello delle prestazioni e lo sforzo compiuto per la gestione del codice.

Esempio di estensione: Si consideri un esempio in cui l'applicazione richiede di catturare il tipo di abitazione di cui dispone un cliente. Si crea una tabella denominata USERRES contenente ID e tipo di residenza cliente, dove il tipo di residenza (resType) può essere una villetta, un condominio o un appartamento. Queste informazioni sono di tipo demografico e, come tali, sono correlate alla tabella USERDEMO Commerce Suite esistente. Esaminando il magazzino di codice di WebSphere Commerce, si riscontra che il gruppo WCSUser EJB contiene un bean enterprise ″Demographics″. Questo bean ha i getter e setter dell'informazione demografica memorizzati nella tabella USERDEMO.

Per eseguire la personalizzazione, vi sono due opzioni. È possibile creare un nuovo bean entità che interagisce con la tabella USERRES oppure è possibile aggiungere un nuovo campo (più i metodi getter e setter appropriati) al bean Demographics.

Utilizzando il primo approccio (creando interamente il nuovo codice), viene creato un nuovo bean entità Userres e i relativi campi vengono associati alle colonne della tabella USERRES. Quando l'applicazione richiede il tipo di residenza del cliente, deve avviare un oggetto bean di accesso Userres e richiamare i dati. Se l'applicazione richiede contemporaneamente altre informazioni demografiche, deve avviare anche un oggetto bean di accesso Demographics e richiamare eventuali altri attributi richiesti. Per avviare un nuovo bean di accesso o quello originale, è necessario modificare le parti della logica applicazione che cercano di richiamare una serie completa di informazioni demografiche di un cliente. Il seguente diagramma mostra questo approccio di estensione modello oggetti:

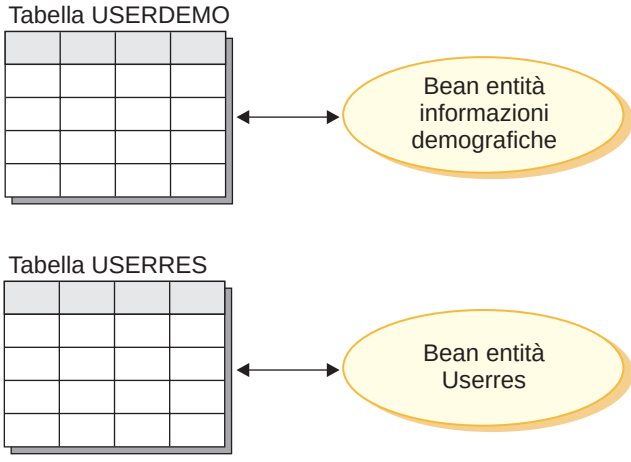


Figura 13.

Dal punto di vista della maschera di visualizzazione, un bean di data deve poter accedere al nuovo attributo, in modo che le informazioni siano disponibili alle maschere JSP. Per presentare una vista unificata allo sviluppatore Web che crea le maschere JSP, è necessario creare un nuovo bean di dati che estenda il bean di accesso per il bean entità originale, esistente. Il bean di dati deve utilizzare una delega per inserire gli attributi dal nuovo bean di accesso. Il seguente diagramma mostra questo scenario di implementazione bean:

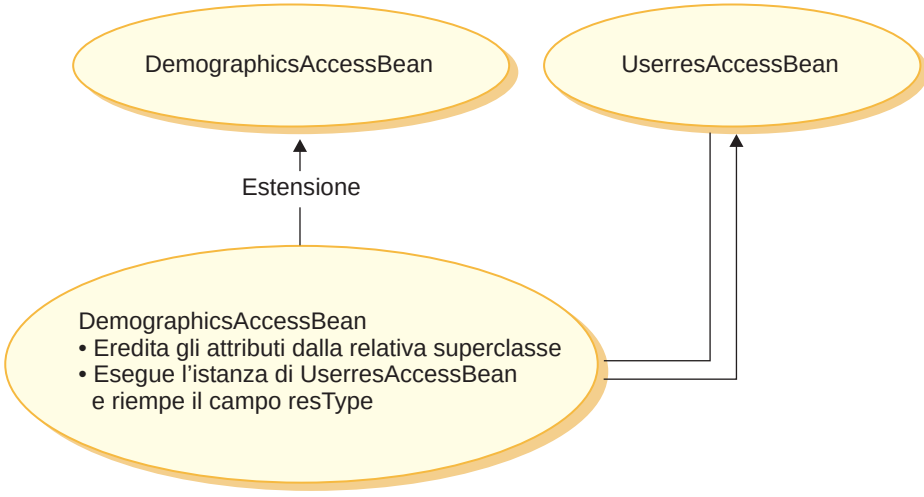


Figura 14.

Utilizzando il secondo approccio (modifica del codice esistente), viene aggiunto un nuovo campo al bean entità Demographics e viene creata una corrispondenza di tabella secondaria tra il nuovo campo e la colonna appropriata nella tabella USERRES. Quando l'applicazione richiede il tipo di residenza del cliente, avvia un oggetto bean di accesso Demographics e richiama il tipo di residenza. Se l'applicazione richiede altri tipi di informazioni demografiche sul cliente, queste sono disponibili nella stessa richiesta al bean. Il seguente diagramma mostra questo approccio per la modifica di bean enterprise:

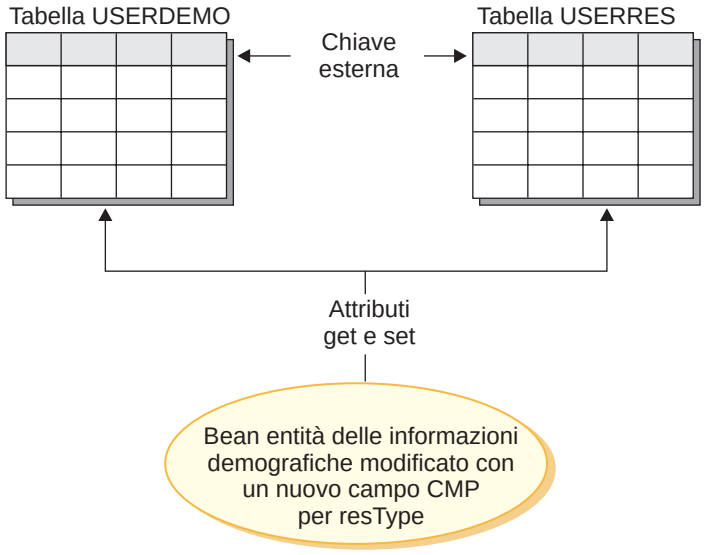


Figura 15.

Dal punto di vista della maschera di visualizzazione, il nuovo attributo (resType) diventa automaticamente disponibile nel bean di dati non appena viene generato nuovamente DemographicsAccessBean.

Si osservi che quando si estende il modello oggetti, *non* devono essere aggiunte nuove colonne alle tabelle di database di WebSphere Commerce esistenti. È necessario creare una nuova tabella per il nuovo attributo. Se si cerca di aggiungere nuove colonne alle tabelle esistenti, quando si eseguirà la migrazione ai successivi rilasci di WebSphere Commerce, il nuovo attributo andrà perso.

Implicazioni per la manutenzione e le prestazioni: Il secondo approccio garantisce migliori prestazioni di runtime. Ciò è dovuto al fatto che per richiamare o impostare il nuovo attributo è necessario solo avviare un singolo bean entità e viene utilizzato un singola acquisizione per richiamare tutti gli attributi richiesti.

Poiché il secondo approccio modifica il codice di WebSphere Commerce esistente, si verifica un problema di migrazione al momento del rilascio del nuovo magazzino di codice WebSphere Commerce. È necessario unire il codice personalizzato con il nuovo codice, ma quando si importa il nuovo magazzino codice, le informazioni sulla corrispondenza tra i campi aggiunti al bean enterprise e la nuova tabella non vengono salvate. Di conseguenza, quando si esegue una migrazione a un nuovo rilascio del magazzino di codice di WebSphere Commerce, è necessario effettuare le seguenti operazioni:

1. Versione del codice EJB personalizzato.

2. Importare la nuova versione del codice di WebSphere Commerce.
3. Utilizzando gli strumenti di VisualAge per Java, confrontare la versione personalizzata del codice con il nuovo rilascio del codice di WebSphere Commerce. Unire nuovamente il codice personalizzato nello spazio di lavoro.
4. Associare di nuovo manualmente gli attributi aggiunti ai bean enterprise di WebSphere Commerce pubblici alle colonne appropriate del database.
5. Generare nuovamente il codice di distribuzione e i bean di accesso per i bean enterprise modificati nel passo 4.

Per semplificare questa migrazione, è importante documentare in modo completo le estensioni del modello oggetti al momento dello sviluppo.

È possibile eseguire una selezione per utilizzare una combinazione dei due approcci quando si effettuano molte estensioni al modello oggetti. È possibile utilizzare il primo approccio per le aree del sistema meno suscettibili alla riduzione delle prestazioni e il secondo approccio quando in caso di problemi relativi alle prestazioni. In tal modo, si riduce il lavoro per le migrazioni future, garantendo al tempo stesso un buon livello delle prestazioni del sistema.

Utilizzo consigliato dei bean di sessione

Uno dei punti di forza di WebSphere Commerce è rappresentato dalla possibilità di utilizzare i bean entità CMP (Container-Managed Persistence). I bean entità CMP sono componenti Java lato server distribuiti, permanenti, transazionali che possono essere generati mediante gli strumenti forniti in VisualAge per Java. In molti casi, questi bean rappresentano una scelta valida per la permanenza degli oggetti e possono rivelarsi molto più efficaci di altre opzioni per l'associazione relazionale degli oggetti. Pertanto, WebSphere Commerce ha implementato gli oggetti commerce utilizzando i bean entità CMP.

Tuttavia, vi sono alcuni casi in cui è consigliabile utilizzare un helper JDBC di bean di sessione. Di seguito vengono illustrate le situazioni specifiche:

- Quando una query restituisce un gruppo di risultati ampio. In questa sezione, si parla di *gruppo di risultati ampio*.
- Quando una query recupera i dati da numerose tabelle. In questa sezione, si parla di *aggregazione di entità*.
- Quando un'istruzione SQL esegue un'operazione di database intensiva. In questa sezione, si parla di *istruzioni SQL arbitrarie*.

Ulteriori dettagli vengono forniti nelle sezioni successive.

Se il bean di sessione viene utilizzato come wrapper JDBC per richiamare informazioni dal database, l'implementazione del controllo accessi a livello

risorsa diventa più difficile. Quando un bean di sessione viene utilizzato in questo modo, lo sviluppatore di tale bean deve aggiungere le clausole "`where`" appropriate all'istruzione "`select`" per impedire che utenti non autorizzati accedano alle risorse.

Gruppo di risultati ampio: Vi sono casi in cui una query restituisce un gruppo di risultati ampio e i dati recuperati sono essenzialmente a scopo di lettura o di visualizzazione. In questo caso, è consigliabile utilizzare un bean di sessione privo di stato e creare all'interno di questo bean un metodo finder che esegua le stesse funzioni di un metodo finder di un bean entità. Il metodo finder creato in un bean di sessione privo di stato deve effettuare le seguenti operazioni:

- Eseguire un'istruzione select SQL
- Creare un'istanza del bean di accesso per ciascuna riga recuperata
- Impostare gli attributi nel bean di accesso per ogni colonna recuperata

Quando il bean di accesso viene restituito, il comando non è in grado di stabilire se questo bean è stato restituito da un metodo finder nel bean di sessione oppure da un metodo finder di un bean entità. Pertanto, l'utilizzo di un metodo finder di un bean di sessione non comporta modifiche al modello di programmazione. Solo il comando che ha eseguito la chiamata è in grado di stabilire se si tratta di un metodo finder del bean di sessione o del bean entità. Questa informazione è chiara a tutte le altre parti del modello di programmazione.

Aggregazione di entità: In questa situazione specifica, una visualizzazione combina diversi oggetti e una singola pagina di visualizzazione viene popolata con informazioni contenute in diverse tabelle di database. Ad esempio, si consideri il concetto "My Account". Può essere rappresentato da informazioni di tabella relative al cliente, (ad esempio, il nome del cliente, l'età o l'ID cliente) o da informazioni contenute nella tabella indirizzi (ad esempio, l'indirizzo in cui vengono specificate una strada e una città).

È possibile creare una semplice istruzione SQL per recuperare tutte le informazioni dalle varie tabelle eseguendo un join SQL. Questa operazione è nota come "acquisizione completa". Di seguito viene riportato un esempio di istruzione select SQL per "My Account", dove T1 rappresenta la tabella CUSTOMER e T2 quella ADDRESS:

```
select T1.NAME, T1.AGE, T2.STREET, T2.CITY
  from CUSTOMER T1, ADDRESS T2
  where (T1.ID=? and T1.ID=T2.ID)
```

Lo strumento EJB di VisualAge per Java non supporta questa nozione di acquisizione completa. Al contrario, esegue un'acquisizione lenta che risulta in un'istruzione select SQL per ciascun oggetto associato. Di solito, non è questa la procedura utilizzata per recuperare questo tipo di informazioni.

Per eseguire un'acquisizione completa, è consigliabile utilizzare un bean di sessione. In questo bean di sessione, creare un metodo finder per recuperare le informazioni richieste. Il metodo finder deve effettuare le seguenti operazioni:

- Eseguire un'istruzione select SQL per l'acquisizione completa
- Creare un'istanza di un bean di accesso per ciascuna riga nella tabella principale e per ciascun oggetto associato.
- Per ogni colonna acquisita e per ciascun oggetto associato acquisito, impostare l'attributo corrispondente nel bean di accesso.

Un bean di accesso non memorizza nella cache un metodo getter che lanci un'eccezione. In questo caso, è necessario creare una semplice classe wrapper per il bean di accesso utilizzando il seguente modello:

```
public class CustomerAccessBeanCopy extends CustomerAccessBean {
   private AddressAccessBean address=null;

   /* The following method overrides the getAddress method in
      the CustomerAccessBean.
   */
   public AddressAccessBean getAddress() {
      if (address == null)
         address = super.getAddress();
      return address;
      }

    /* The following method sets the address to the copy. */

    public void _setAddress(AddressAccessBean aBean) {
       address = aBean;
      }
}
```

Continuando con l'esempio di CUSTOMER e ADDRESS, il metodo finder del bean di sessione crea un'istanza di un CustomerAccessBean per ciascuna riga della tabella CUSTOMER e di un AddressAccessBean per ciascuna riga corrispondente della tabella ADDRESS. Quindi, per ogni colonna della tabella ADDRESS, imposta gli attributi in AddressAccessBean (strada e città). Per ogni colonna della tabella ADDRESS, imposta gli attributi in CustomerAccessBean (nome, età e indirizzo). Questa procedura viene illustrata nel seguente diagramma.

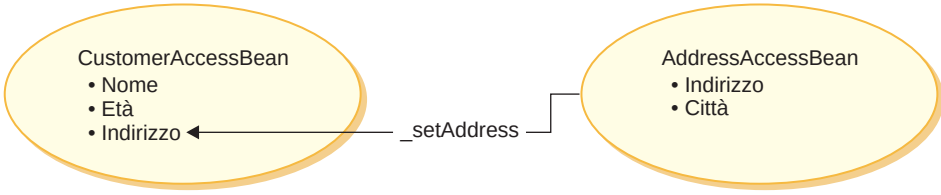


Figura 16.

Istruzioni SQL arbitrarie: In questo caso, vi è un gruppo di istruzioni SQL arbitrarie che eseguono operazioni intensive del database. Ad esempio, l'operazione per sommare tutte le righe di una tabella viene considerata un'operazione intensiva di database. È possibile che non tutte le righe selezionate corrispondano a un bean entità nel modello permanente.

Quando un cliente cerca di visualizzare un set di dati ampio, viene creata un'istruzione SQL arbitraria: ad esempio, il cliente desidera esaminare tutti i dispositivi di fissaggio in un negozio di ferramenta in linea oppure tutti i vestiti di un negozio di abbigliamento in linea. Questo tipo di ricerca genera un gruppo di risultati ampio e, probabilmente, sono richiesti solo pochi campi per ciascuna riga. È possibile che inizialmente venga presentato un riepilogo in cui vengano riportati il nome, l'immagine e il prezzo degli articoli.

In questo caso, creare un metodo helper del bean di sessione. Questo metodo esegue un'operazione di lettura o di scrittura. Quando esegue un'operazione di lettura, restituisce un oggetto di sola lettura utilizzato solo a scopo di visualizzazione.

Di solito, un data modelling appropriato riduce il numero di casi di istruzioni SQL arbitrarie.

Estensione dei bean entità pubblici

Questa sezione illustra il modello di progettazione dei bean entità pubblici di WebSphere Commerce. Questo modello consente di eseguire estensioni come l'aggiunta di nuovi campi permanenti, nuovi metodi aziendali o nuovi metodi finder.

Il seguente diagramma mostra le classi di implementazione del bean entità Catalog.

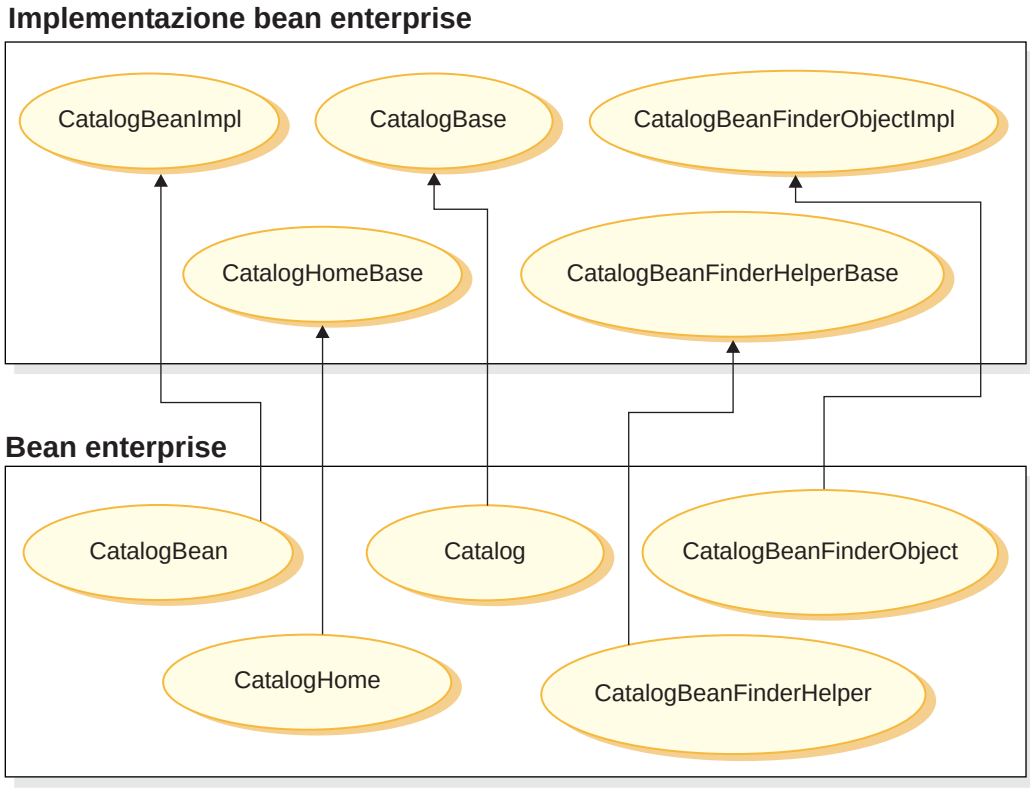


Figura 17.

Il diagramma precedente è valido anche per altri bean entità, in quanto questi sono strutturati in un formato simile e seguono le stesse regole di denominazione. Per applicare il diagramma a un altro bean entità, sostituire il nome del bean entità per "Catalog". Ad esempio, la classe InterestItemBean consente di estendere la classe InterestItemBeanImpl e l'interfaccia InterestItem consente di estendere l'interfaccia InterestItemBase.

Il diagramma mostra come la classe di implementazione o l'interfaccia per i bean enterprise sia stata separata in due parti, utilizzando l'eredità Java. La superclasse o l'interfaccia contengono il codice di implementazione di WebSphere Commerce. Tutte queste superclassi o interfacce sono definite in pacchetti e progetti separati dalle classi secondarie e dalle interfacce.

Ad eccezione delle classi *nome_bean*`BeanFinderHelperBase` e *nome_bean*`BeanFinderObjectImpl`, WebSphere Commerce contiene il codice binario per tutte le superclassi e le interfacce. Il codice di origine per le classi *nome_bean*`BeanFinderHelperBase` e *nome_bean*`BeanFinderObjectImpl` è

compreso, quindi è possibile accedere alla definizione dell'istruzione SQL per ciascun metodo finder definito. In ogni caso, queste classi non devono essere modificate.

Per esaminare il codice di origine per `CatalogBeanFinderHelperBase` e `CatalogBeanFinderObjectImpl`, aprire il pacchetto `com.ibm.commerce.catalog.objsrc`. Altri tipi di pacchetto utilizzano regole di denominazione simili.

Le modifiche possono essere eseguite nelle classi secondarie e nelle interfacce.

Se si aggiungono nuovi metodi finder ai bean enterprise pubblici, risulta necessario seguire una particolare convenzione di ridenominazione per i metodi. Ridenominare i nuovi metodi `findX`*`una_descrizione`* in cui *`una_descrizione`* è la descrizione della selezione effettuata. Alcuni esempi di nome sono `findXByOwnerId` e `findXByOrderStatus`. L'utilizzo di questa convenzione di ridenominazione consente di evitare il rischio di un conflitto tra nomi (nomi duplicati) con i metodi finder di WebSphere Commerce.

Un modo per modificare un bean entità pubblico di WebSphere Commerce esistente, consiste nell'aggiungere ulteriori campi. In questo caso, dopo aver aggiunto i nuovi campi, è necessario esaminare ciascun metodo finder nel bean. Se la parte della clausola `where` dei metodi finder contiene alias di database, (ad esempio, T1. o T2.), occorre rimuovere gli alias.

Creazione di un nuovo bean enterprise CMP

Se si dispone di un nuovo attributo che deve essere aggiunto al modello oggetti di WebSphere Commerce, è possibile creare una nuova tabella di database in cui è presente una colonna per l'attributo richiesto. È quindi necessario includere questo attributo in un bean enterprise, in modo che i comandi di WebSphere Commerce possano accedere alle informazioni.

Un modo per integrare il nuovo attributo nel modello oggetti di WebSphere Commerce è la creazione di un nuovo bean enterprise CMP. In questo bean, è quindi necessario creare un campo che corrisponda all'attributo nella nuova tabella del database.

Per creare un nuovo bean enterprise CMP, è necessario effettuare le seguenti operazioni in VisualAge per Java:

1. Accertarsi che il proprietario dello spazio di lavoro sia impostato come Sviluppatore WCS.
2. Creare un nuovo gruppo EJB per il bean.
3. Creare il nuovo bean enterprise CMP utilizzando la SmartGuide per la creazione dei bean enterprise.

4. Per ogni colonna della corrispondente tabella del database, aggiungere al bean un nuovo campo CMP.
5. Se necessario, creare una coppia di metodi getter e setter per ogni campo CMP creato.
6. Se necessario, definire i campi FinderHelper nell'interfaccia FinderHelper e aggiungere il nuovo metodo FinderHelper.
7. Creare un nuovo metodo ejbCreate, se richiesto, e promuovere il metodo ejbCreate sull'interfaccia principale del bean enterprise. Questa operazione è necessaria se il nuovo bean enterprise deve creare nuove voci all'interno della corrispondente tabella del database.
8. Associare i campi del bean enterprise alle colonne della tabella del database.
9. Creare il bean di accesso e il codice distribuito per il bean enterprise.

Nelle seguenti sezioni, vengono fornite ulteriori informazioni su questi passaggi.

Nota: Nel caso in cui sia necessario proteggere il nuovo bean mediante il sistema di controllo accessi di WebSphere Commerce, consultare Capitolo 4, "Controllo accessi" a pagina 91 per ulteriori dettagli. Il controllo accessi può essere aggiunto anche dopo aver creato il bean.

Si supponga di disporre di una nuova tabella denominata USERRES in cui vengono specificate informazioni sul tipo di residenza dell'utente. In questa tabella sono presenti tre colonne: una colonna IDUTENTE, una colonna CASA in cui si specifica il tipo di casa e una colonna CAMERE in cui viene specificato il numero di camere dal letto dell'abitazione.

Creazione di un nuovo gruppo EJB: Quando si creano nuovi bean entità, devono essere creati in un gruppo EJB separato dai gruppi EJB di WebSphere Commerce. Quando si lavora con i bean enterprise in VisualAge per Java, è necessario passare alla scheda EJB all'interno dello spazio di lavoro. Quindi, dal menu **EJB**, è possibile selezionare **Aggiungi > Gruppo EJB** per avviare la SmartGuide per l'aggiunta di gruppi EJB.

Durante la creazione di un gruppo EJB, è necessario specificare due dati importanti: il progetto nel quale è memorizzato il codice EJB e il nome del gruppo EJB.

È possibile visualizzare il progetto EJB dalla scheda Progetti all'interno dello spazio di lavoro. Quando si crea un nuovo gruppo EJB, è necessario specificare un progetto diverso dai progetti WebSphere Commerce. Ad esempio, potrebbe essere necessario effettuare una selezione in modo che il gruppo EJB utilizzi il progetto MyCustomEJB. Il progetto non deve essere presente prima della creazione del gruppo, in quanto può essere

automaticamente creato da VisualAge per Java. È necessario che il progetto venga utilizzato soltanto per il codice EJB e non per qualsiasi comando o codice di bean di dati. Questa diversificazione dei tipi di codice è necessaria per le esigenze di sviluppo. Separando il proprio codice personalizzato dal codice di WebSphere Commerce, l'impatto della migrazione per i rilasci futuri sarà minimo.

Per l'assegnazione del nome del progetto e del gruppo EJB, assicurarsi di utilizzare le convenzioni di denominazione dell'applicazione appropriate.

Creazione di un nuovo bean enterprise CMP: Per creare il nuovo bean enterprise CMP, è possibile utilizzare la SmartGuide per la creazione dei bean enterprise. Per avviare lo strumento, fare clic con il pulsante destro del mouse sul gruppo EJB al quale si desidera aggiungere il nuovo bean e selezionare **Aggiungi > Bean enterprise**.

Effettuare una selezione per creare un nuovo bean enterprise e specificare il relativo nome. Con le convenzioni di denominazione di WebSphere Commerce per i bean enterprise è necessario denominare il bean utilizzando lo stesso nome della tabella alla quale il bean corrisponde. Ad esempio, se la nuova tabella del database si chiama USERRES, è necessario che il bean enterprise venga denominato UserRes.

Il nome del progetto viene immediatamente aggiunto al campo Progetto. All'interno del progetto è necessario specificare un nome di pacchetto nel quale il codice deve essere memorizzato. Un esempio di codice da memorizzare nel pacchetto è il codice per il bean di accesso creato per il bean enterprise. Inoltre, durante la denominazione del pacchetto, assicurarsi di seguire le convenzioni di denominazione relative alla propria applicazione. Un nome di esempio per questo pacchetto è `com.mycompany.mycustombeans`.

Nella classe di bean, VisualAge per Java crea il campo privato denominato `EntityContext`. WebSphere Commerce fornisce il proprio campo del contesto dell'entità all'interno di ECEntityBean e il nuovo bean entità utilizza tale campo piuttosto che quello generato nella classe. Pertanto, è necessario rimuovere il campo `EntityContext` appena creato dal nuovo bean entità.

Il nuovo bean enterprise deve contenere un campo `serialVersionUID`. Se si utilizza la SmartGuide per creare il nuovo bean, VisualAge per Java genera questo campo. Se non si utilizza lo strumento per creare il bean, è necessario aggiungere questo campo.

Nel campo Superclasse è necessario specificare la classe `com.ibm.commerce.base.objects.ECEntityBean`. Il seguente codice di esempio mostra le funzioni fornite dalla superclasse:

```
public class myEJB extends com.ibm.commerce.base.objects.ECEntityBean {
   public void ejbLoad()throws java.rmi.RemoteException {
        super.ejbLoad();--the super method will add EJB trace
  --your logic --
   }
    public void ejbStore()throws java.rmi.RemoteException {
        super.ejbStore();--the super method will add EJB trace
        --your logic --
   }
}
```

È inoltre necessario rimuovere qualsiasi metodo ejbCreate() e ejbPostCreate() che non contiene argomenti. La mancata rimozione di questi metodi nel bean entità potrebbe provocare degli errori di runtime.

Per ogni colonna della tabella del database è necessario creare un nuovo campo CMP all'interno del bean (facendo clic su Avanti all'interno della SmartGuide in modo da visualizzare la relativa opzione per l'aggiunta di campi CMP). Per ogni campo, specificare un nome e il tipo di dati del campo. Selezionare la casella di controllo Campo chiave per qualsiasi colonna contenente la chiave principale o parte di essa. Per tutte le altre colonne, selezionare la casella di controllo dell'interfaccia remota Promuovi metodi getter e setter.

Una volta completati tutti i campi, fare clic su Fine e verrà creato il nuovo bean enterprise CMP in VisualAge per Java.

Creazione di nuovi campi FinderHelper nell'interfaccia *Nome_Bean*FinderHelper: L'interfaccia *Nome_Bean*FinderHelper contiene le clausole di ricerca SQL che corrispondono a tutti i metodi FinderHelper diversi dal metodo findByPrimaryKey.

Un nuovo bean enterprise utilizza i metodi findXBy*NomeArg* (dove *NomeArg* è il nome di un argomento) definiti nell'interfaccia FinderHelper del bean con i metodi FinderHelper per definire query SQL. Utilizzare la convenzione di denominazione "findXBy" per il nome del campo per assicurare che i nomi dei campi siano sempre univoci rispetto ai nomi dei campi di WebSphere Commerce.

Per creare i nuovi campi FinderHelper nel bean, è necessario selezionare l'interfaccia *Nome_Bean*_FinderHelper e modificare il codice di origine per stabilire come definire le istruzioni select. Ad esempio:

```
public interface UserResFinderHelper {
 public static final String findXByHomeAndRoomsWhereClause = "(T1.HOME = ?
      and T1.ROOMS = ?)";
}
```

L'interfaccia home potrebbe richiedere quindi un metodo denominato `findXByHomeAndRooms` che accetti parametri di immissione per ciascuna CASA e CAMERE che popolano i valori rappresentati dal carattere ?. Questo tipo di costruzione di query viene definito *parametro insert*.

Se i parametri di immissione fossero stati ″detached″ e ″3″, l'istruzione SQL generata sarebbe stata

```
select * from USERRES where HOME=detached and ROOMS=3
```

Per motivi di sicurezza, quando si creano i metodi FinderHelper per un nuovo bean entità, è necessario utilizzare parametri insert. Il motivo principale è che tali parametri consentono di proteggere la query da eventuali modifiche da parte di utenti. Un approccio alternativo potrebbe essere l'utilizzo di un costrutto simile al seguente:

```
public static final String
     findXByOwnerIdWhereClause = " (T1.OWNERID = stringa_immissione) ";
```

dove *stringa_immissione* è un valore di stringa inoltrato da un URL. Questa soluzione non è consigliata, poiché utenti non autorizzati potrebbero immettere il valore "'123' oppure 1=1", modificando, così, l'istruzione SQL. Se un utente può modificare l'istruzione SQL, è possibile che possano accedere a dati anche senza autorizzazione. Quindi, l'approccio consigliato è quello che prevede l'utilizzo del parametro insert.

Se non è possibile utilizzare un parametro insert e, quindi, occorre utilizzare una stringa di immissione per comporre l'istruzione SQL, è necessario implementare la verifica del parametro sulla stringa di immissione per evitare tentativi di accesso ai dati non desiderati.

Creazione di nuovi metodi FinderHelper nell'interfaccia *Nome_Bean*Home: Per ogni campo FinderHelper specificato nell'interfaccia *Nome_Bean*`FinderHelper`, è necessario creare un metodo FinderHelper nell'interfaccia home del bean. Il nome dei metodi FinderHelper deve corrispondere esattamente al nome del campo FinderHelper, tranne per la parte ″WhereClause″ che viene eliminata. Questo significa che per il nome del campo di esempio `findXByHomeAndRoomsWhereClause`, il nome del metodo corrispondente è `findXByHomeAndRooms`.

Per creare i nuovi metodi FinderHelper, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sull'interfaccia *Nome_Bean* **Home** e selezionare **Aggiungi > Metodo**.
   Si apre la SmartGuide per la creazione dei metodi.
2. Selezionare **Crea nuovo metodo** quindi **Avanti**.
3. Nel campo **Nome metodo** immettere un nome per il metodo FinderHelper. Questo nome deve corrispondere esattamente al nome del campo

FinderHelper, fatta eccezione per la parte "WhereClause" che può essere eliminata. Ad esempio, immettere `findXByHomeAndRooms`.

4. Nel campo **Tipo di risultato** immettere una delle seguenti opzioni:
   - Se il metodo FinderHelper utilizza la chiave principale per eseguire query nel database e se il metodo restituisce un record univoco, specificare come tipo di risultato l'oggetto EJB. Ad esempio, immettere `UserRes`.
   - Se il metodo FinderHelper restituisce un gruppo di risultati anziché un unico record, specificare il tipo restituito come `java.util.Enumeration`.
5. Fare clic sul pulsante **Aggiungi** accanto a **Quali parametri dovrebbe avere questo metodo?** per aggiungere i parametri appropriati. Ad esempio, è possibile aggiungere `argHome` di tipo `String` per specificare il tipo di residenza e `argRooms` di tipo `byte` per specificare il numero di camere.
6. Una volta aggiunti tutti i parametri, fare clic su **Avanti**.
7. Fare clic sul pulsante **Aggiungi** accanto a **Quali eccezioni devono essere attivate da questo metodo?** e aggiungere le seguenti eccezioni:
   - `java.rmi.RemoteException`
   - `javax.ejb.FinderException`

   Nota: Il precedente elenco di eccezioni riporta il gruppo minimo di eccezioni che il metodo dovrebbe lanciare. A seconda del codice, è possibile che sia necessario specificare altre eccezioni.
8. Fare clic su **Fine**.

Creazione di un nuovo metodo ejbCreate: Una volta creato il nuovo bean enterprise, viene automaticamente generato il metodo ejbCreate. Questo metodo viene promosso all'interfaccia remota in modo che sia disponibile nel bean di accesso. Il metodo predefinito ejbCreate contiene soltanto parametri che rappresentano la chiave principale o una parte di essa. Questo significa che viene eseguita l'istanza soltanto per quei valori in seguito all'esecuzione dell'istanza.

Se il bean enterprise contiene campi che non fanno parte della chiave principale e che sono campi non annullabili, è necessario creare un nuovo metodo ejbCreate in cui eseguire l'istanza per tali campi. Effettuando questa operazione, ogni volta che viene creato un nuovo record, tutti i campi non annullabili verranno completati con i dati appropriati.

Per creare un nuovo metodo ejbCreate, effettuare le seguenti operazioni:

1. Nel pannello Tipi espandere la classe ***Nome_Bean*Bean**. Ad esempio, selezionare **UserResBean**.
2. Fare clic sul metodo ejbCreate esistente per visualizzare il codice di origine. Potrebbe trattarsi di ejbCreate(String), ejbCreate(String, int) oppure

è possibile che vengano richiesti altri parametri di immissione, a seconda della chiave principale del bean enterprise.

3. È necessario modificare il codice di origine in modo che ogni campo CMP venga incluso come parametro di immissione nel metodo e che venga eseguita l'istanza con il valore appropriato ogni campo. Per l'esempio UserRes in cui UserId è la chiave principale, il codice di origine inizialmente è simile a:

```
public void ejbCreate(int argUserId)
   throws javax.ejb.CreateException, java.rmi.RemoteExeption {
    _initLinks();
   userId = argUserId;
}
```

   Tuttavia, è opportuno assicurarsi che venga eseguita l'istanza sia per il numero di camere che per il tipo di residenza. In questo caso, è necessario modificare il codice come riportato di seguito:

```
public void ejbCreate(int argUserId, String argHome, byte Rooms)
   throws javax.ejb.CreateException, java.rmi.RemoteExeption {
   _initLinks();
   // All CMP fields should be initialized here
   userId = argUserId;
   home = argHome;
   rooms = argRooms;
}
```

   Nota: Se si desidera utilizzare una chiave principale generata dal sistema, consultare "Chiavi principali" a pagina 82 per dettagli.

4. Salvare il metodo modificato. Quando si salva il codice, VisualAge per Java crea un nuovo metodo ejbCreate in cui vengono richiamati i nuovi parametri. Il metodo ejbCreate originale verrà conservato.
5. Eliminare il metodo ejbCreate originale (quello senza argomenti).
6. Fare clic con il pulsante destro del mouse sul nuovo metodo ejbCreate e selezionare **Aggiungi > Interfaccia home EJB**.
7. Creare un metodo ejbPostCreate corrispondente. (Non è necessario aggiungere questo metodo all'interfaccia home.)

Associazione tra la tabella del database e il nuovo bean enterprise: Una volta creato il nuovo bean enterprise, è necessario associare i campi CMP del bean e le colonne della tabella del database. Questa associazione è detta schema. Con VisualAge per Java vengono forniti degli strumenti che consentono di semplificare questa attività.

Per creare lo schema, effettuare le seguenti operazioni:

1. Dal menu **EJB** selezionare **Apri in > Schemi database**. Si apre la finestra Schema browser.

2. Dal menu **Schemi** selezionare **Importa/Esporta schemi > Importa schema dal database**.
3. Immettere un nome per il nuovo schema e fare clic su **OK**.
4. Nella finestra della connessione al database immettere le seguenti informazioni:

Attributo	Valore DB2	Valore Oracle
Tipo collegamento	`COM.ibm.db2.jdbc.app.DB2Driver`	`Oracle.jdbc.driver.OracleDriver`
Origine dati	`jdbc:db2:wc_database_name`	`jdbc:oracle:thin@`*`nomehost`*`:`*`porta`*`:`*`SID_Oracle`*
ID utente	*`nome_utente_db_wc`*	*`nome_utente_db_wc`*
Parola d'ordine	*`password_db_wc`*	*`password_db_wc`*

con i valori sostituiti come riportato di seguito:

- DB2 *`nome_database_wc`* è il nome del WebSphere Commerce database
- Oracle *`nomehost`* è il nome host del server Oracle.
- Oracle *`porta`* è il numero della porta del database Oracle.
- Oracle *`SID_Oracle`* è l'ID istanza Oracle.
- *`nome_utente_db_wc`* è il nome utente per il database.
- *`password_db_wc`* è la password del database.

5. Dall'elenco **Qualificatori**, selezionare il qualificatore del database (può trattarsi del nome utente database).
6. Fare clic su **Crea elenco tabelle**.
7. Selezionare *nuova_tabella* dall'elenco generato, quindi fare clic su OK per creare lo schema.
8. Una volta creato lo schema, fare clic su *nuova_tabella* nel pannello Schema.
9. Selezionare, quindi fare doppio clic su *nuova_tabella* nel pannello Tabella. Viene visualizzata la finestra Editor tabelle.
10. Rimuovere qualsiasi informazione dal campo **Qualificatori**. In tal modo, il bean enterprise può essere portato su altri computer.
11. Dal menu **Schemi**, selezionare **Salva schema**. Immettere i relativi nomi della classe, del pacchetto e del progetto.

In seguito, è necessario creare la corrispondenza schema. La corrispondenza dello schema crea un'associazione tra i campi e le colonne del database nel bean enterprise.

Per creare una corrispondenza schema, effettuare le seguenti operazioni:

1. Dal menu **EJB** selezionare **Apri in > Mappe schemi**.
   Si apre la finestra Mappe datastore.
2. Immettere il nome della corrispondenza. Per ulteriori informazioni sulla denominazione della nuova corrispondenza, fare riferimento a "Considerazioni sulla denominazione di oggetti dello schema del database" a pagina 85.
3. Selezionare lo schema e il gruppo EJB. Per ulteriori informazioni sulla denominazione del nuovo schema, fare riferimento a "Considerazioni sulla denominazione di oggetti dello schema del database" a pagina 85.
4. Nel pannello Mappe datastore selezionare la relativa corrispondenza.
5. Nel pannello Classi permanenti selezionare la classe.
6. Dal menu **Mappa di tabella**, selezionare **Nuova mappa di tabella > Aggiungi mappa di tabella senza eredità**.
7. Dall'elenco a discesa **Tabelle** selezionare la relativa tabella e fare clic su **OK**.
8. Nel pannello Mappe tabelle, evidenziare e fare clic con il pulsante destro del mouse sulla corrispondenza della tabella, quindi selezionare **Edita corrispondenza delle proprietà**.
   Si apre la finestra dell'editor di corrispondenze di proprietà.
9. Per ognuno degli attributi della classe (i campi CMP nel bean) è necessario specificare il tipo di corrispondenza e la colonna del database alla quale deve corrispondere. Ad esempio, si potrebbe creare una corrispondenza per l'attributo della classe UserId utilizzando un tipo di corrispondenza ″Simple″ per la colonna IDUTENTE della tabella del database.
10. Una volta creata una corrispondenza tra tutti i campi e le relative colonne del database, è necessario salvare la corrispondenza schema. Dal menu **Mappe datastore** selezionare **Salva corrispondenza datastore**. per salvare la corrispondenza, immettere i relativi nomi della classe, del pacchetto e del progetto. Fare clic su **Fine**.

Creazione di un bean di accesso e generazione del codice di distribuzione:
Un bean di accesso funge da wrapper per il bean enterprise che semplifica il modo in cui gli altri componenti interagiscono con il bean enterprise. È necessario creare un bean di accesso per il nuovo bean enterprise.

Quando si crea un nuovo codice distribuito, gli strumenti di VisualAge per Java analizzano il bean per accertarsi che le regole delle specifiche EJB Sun Microsystems e le regole particolari del server EJB corrispondano.

Per creare un bean di accesso per il nuovo bean enterprise, effettuare le seguenti operazioni:

1. Nel pannello Bean enterprise fare clic con il pulsante destro del mouse sul nuovo bean enterprise e selezionare **Aggiungi > Bean di accesso**. (Per visualizzare il bean, potrebbe essere necessario espandere prima il gruppo EJB che contiene il nuovo bean.)
   Si apre la SmartGuide per la creazione dei bean di accesso.
2. Nei campi relativi al **gruppo EJB**, al **bean enterprise** e al **nome del bean di accesso**, specificare il gruppo EJB, il nome bean e il nome del bean di accesso appropriati.
3. Selezionare **Copy Helper per un bean entità per il tipo di bean di accesso** e fare clic su **Avanti**.
4. Dall'elenco a discesa **Seleziona metodo locale per constructor senza argomenti**, selezionare **findByPrimaryKey**.
5. Dall'elenco a discesa **Programma di conversione**, selezionare **WCSStringConverter** per le proprietà iniziali e fare clic su **Avanti**.
6. Nella finestra Seleziona e personalizza proprietà bean per copy helper, selezionare **WCSStringConverter** per ciascun campo.
7. Fare clic su **Fine**.

Per generare il codice di distribuzione, effettuare le seguenti operazioni:

1. Nel pannello Bean enterprise fare clic con il pulsante destro del mouse sul nuovo bean enterprise e selezionare **Genera codice di distribuzione**.

Si noti che il codice di distribuzione generato utilizzando questo strumento è conforme alla specifica EJB 1.0 e viene utilizzato soltanto eseguendo il bean enterprise all'interno di VisualAge per Java. In una fase successiva, quando si distribuisce il bean enterprise in un'applicazione WebSphere Commerce in esecuzione all'interno di WebSphere Application Server V4.0, viene richiesto di generare un file JAR contenente un codice di distribuzione conforme alle specifiche EJB 1.1. Per ulteriori informazioni sulla creazione di questo file JAR EJB 1.1 di esportazione, consultare "Informazioni sul codice di distribuzione EJB" a pagina 194 e "Generazione del codice di distribuzione" a pagina 360.

Utilizzo del client di verifica per il test del bean enterprise: VisualAge per Java fornisce un client di verifica che può essere utilizzato per eseguire il test dei bean enterprise. Per utilizzare tale client per verificare il nuovo bean, effettuare le seguenti operazioni:

1. Avviare il server EJB che contiene il bean enterprise che si desidera verificare.
2. Fare clic con il pulsante destro del mouse sul bean enterprise e selezionare **Esegui client di prova**.
   Vengono aperte le finestre Client del test EJB e Ricerca EJB.
3. Nel campo **Nome JNDI** immettere il nome JNDI del bean enterprise e fare clic su **Ricerca**.

4. Fare clic con il pulsante destro del mouse sul metodo **findByPrimaryKey** che dispone degli argomenti e selezionare **Richiama**.

Pratiche di codifica: Si consiglia di utilizzare le seguenti pratiche di codifica dei bean enterprise:

- Non utilizzare i tipi di dati BLOB e CLOB.
- Nessun campo CMP del tipo di dati LONG (meglio conosciuto come LONG VARCHAR) deve essere il primo o l'ultimo membro dell'elenco dei campi CMP per il bean enterprise. Per verificare tale elenco, selezionare il metodo `EJSJDBCPersister._hydrate()` e accertarsi che il primo o l'ultimo elemento dell'elenco sia del tipo LONG VARCHAR. In questo caso, effettuare quanto segue:
  1. Rimuovere le informazioni del primo campo. Denominare tale campo `fieldA`.
  2. Rimuovere le informazioni da un altro campo che *non sia* del tipo LONG VARCHAR. Denominare tale campo `fieldB`.
  3. Reimpostare il campo `fieldA`.
  4. Reimpostare il campo `fieldB`.
  5. Aprire il browser della corrispondenza schemi e modificare la corrispondenza tabella in modo da confermare le modifiche apportate. Salvare la corrispondenza.
  6. Scrivere nuovamente il codice distribuito.
- Il codice dei bean enterprise non deve fare riferimento a niente altro che non sia presente all'esterno dei pacchetti dei bean enterprise. Ad esempio, non devono fare riferimento a comandi o bean di dati nel codice del bean enterprise.
- Per abilitare il controllo accessi per il bean enterprise, aggiungere l'interfaccia com.ibm.commerce.security.Protectable e/o l'interfaccia com.ibm.commerce.security.Groupable all'interfaccia remota del bean enterprise. Dopo aver aggiunto tali interfacce, creare nuovamente il codice distribuito del bean e il bean di accesso. È inoltre necessario creare un oggetto classe helper di accesso per il pacchetto objsrc.

Creazione di un bean di dati semplice

Un bean di dati è un bean che viene utilizzato nelle maschere JSP per recuperare informazioni dal bean enterprise. Un bean di dati semplice consente di estendere il relativo bean di accesso e di implementare l'interfaccia SmartDataBean. La maggior parte del codice del bean di dati viene generata automaticamente da VisualAge per Java.

Per creare un bean di dati semplice, effettuare le seguenti operazioni:

1. Creare un progetto e un pacchetto per memorizzare il codice del bean di dati.

2. Creare un bean di dati che consenta di estendere il bean di accesso corrispondente e di estendere l'interfaccia del bean di dati appropriata.
3. Creare i metodi setter per il bean di dati.
4. Creare i metodi getter per il bean di dati.

Creazione del progetto e del pacchetto per il codice del bean di dati: La creazione di un progetto e di un pacchetto consente di definire un'ubicazione in cui memorizzare il codice del bean di dati.

Per creare un nuovo progetto, effettuare le seguenti operazioni:

1. Selezionare la scheda **Progetti**.
2. Dal menu **Selezionato**, selezionare **Aggiungi > Progetto**.
   Si apre la SmartGuide per l'aggiunta di un progetto.
3. Assicurarsi che **Crea nuovo progetto denominato** sia selezionata e specificare un nome per il nuovo progetto. Ad esempio, immettere `My Data Beans`.
4. Fare clic su **Fine**.

Per creare un nuovo pacchetto, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sul progetto creato per il codice del bean di dati e selezionare **Aggiungi> Pacchetto**. Ad esempio, fare clic con il pulsante destro del mouse su **My Data Beans** e selezionare **Aggiungi > Pacchetto**.
   Si apre la SmartGuide per l'aggiunta dei pacchetti.
2. Assicurarsi che **Crea nuovo pacchetto denominato** sia selezionata e specificare un nome appropriato per il nuovo pacchetto bean di dati. Ad esempio, immettere `com.mycompany.mydatabeans`.
3. Fare clic su **Fine**.

Creazione di un bean di dati: Un bean di dati è un bean Java utilizzato in una maschera JSP per fornire contenuto dinamico alla pagina. Di solito, fornisce una rappresentazione semplice (indirettamente) di un bean entità estendendo un bean di accesso. Il bean di dati contiene proprietà che possono essere richiamate da o impostate in un bean entità.

Per creare un bean di dati, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sul pacchetto in cui memorizzare il bean di dati e selezionare **Aggiungi > Classe**.
   Si apre la SmartGuide per la creazione delle classi.
2. I campi del nome del progetto e del pacchetto hanno già dei valori.
3. Assicurarsi che l'opzione **Crea nuova classe** sia selezionata e fare clic su **Avanti**.

4. Nel campo **Nome classe** immettere un nome per il nuovo bean di dati. Ad esempio, per creare un bean di dati che consenta di estendere UserResAccessBean, immettere `UserResDataBean`.
5. Per specificare la superclasse, fare clic su **Sfoglia**, quindi nel campo modello immettere il nome del bean di accesso corrispondente. Ad esempio, immettere `UserResAccessBean` e fare clic su **OK**.
6. Fare clic su **Avanti**.
7. Per specificare le interfacce che il bean di dati deve implementare, fare clic su **Aggiungi**. Nella finestra Interfaccia, effettuare le seguenti operazioni:
   a. Nel campo **Modello**, immettere `com.ibm.commerce.beans.SmartDataBean`, quindi fare clic su **Aggiungi**.
   b. Nel campo **Modello**, immettere `com.ibm.commerce.beans.InputDataBean`, quindi fare clic su **Aggiungi**.
   c. Fare clic su **Chiudi**.
8. Fare clic su **Fine**.

Aggiunta di campi richiesti al bean di dati: Questa sezione descrive come aggiungere campi richiesti al nuovo bean di dati.

Per aggiungere il campo iCommandContext, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sul bean di dati (ad esempio, UserResDataBean) e selezionare Aggiungi > Campo.
   Si apre la SmartGuide per la creazione del campo.
2. Nel campo **Nome campo**, immettere `iCommandContext`.
3. Fare clic su **Sfoglia** per aggiungere il tipo di campo e immettere `com.ibm.commerce.command.CommandContext`. Fare clic su **OK**.
4. Per i modificatori di accesso, selezionare **Protetto**.
5. Fare clic su **Fine**.

Per aggiungere il campo iRequestProperties, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sul bean di dati (ad esempio, UserResDataBean) e selezionare Aggiungi > Campo.
   Si apre la SmartGuide per la creazione dei campi.
2. Nel campo **Nome campo**, immettere `iRequestProperties`.
3. Fare clic su **Sfoglia** per aggiungere il tipo di campo e immettere `com.ibm.commerce.datatype.TypedProperty`. Fare clic su **OK**.
4. Per i modificatori di accesso, selezionare **Protetto**.
5. Fare clic su **Fine**.

Modifica dei metodi set del bean di dati: Dopo aver creato il bean di dati, è necessario modificare il codice in alcuni metodi set generati.

Per aggiornare i metodi set, effettuare le seguenti operazioni:

1. Espandere il nuovo bean di dati per visualizzarne i campi e i metodi.
2. Selezionare il metodo **setCommandContext(CommandContext)** per visualizzarne il codice di origine.
   Viene visualizzato il pannello contenente il codice di origine:

   ```
   public void setCommandContext(com.ibm.commerce.comand.CommandContext arg1)
    {
    }
   ```

3. Modificare il codice di origine in modo che venga visualizzato come riportato di seguito:

   ```
   public void setCommandContext(com.ibm.commerce.comand.CommandContext arg1)
   {
      iCommandContext = arg1;
   }
   ```

   Salvare il proprio lavoro (Ctrl+S).
4. Selezionare il metodo **setRequestProperties(TypedProperty)** per visualizzare il codice di origine.
   Viene visualizzato il pannello contenente il codice di origine:

   ```
   public void setRequestProperties(
    com.ibm.commerce.datatype.TypedProperty arg1)
      throws Exception
      {
      }
   ```

5. È possibile modificare il codice di origine per popolare la chiave principale del bean di accesso corrispondente. A tal scopo, è consigliabile utilizzare il gestore dei bean di dati per impostare indirettamente questo valore. Questo metodo indiretto è stato progettato per assicurare che il valore di una chiave principale richiamato dalle proprietà URL non sovrascrivi la chiave principale nel caso in cui sia stata impostata precedentemente. Perché il metodo setRequestProperties segua questo modello, specificare per questo metodo la sezione di codice riportata di seguito. In questo esempio, la chiave principale è l'id utente. Questo valore può essere diverso a seconda delle situazioni: ad esempio, è possibile che il codice riportato di seguito non venga compilato immediatamente nell'applicazione:

   ```
   public void setRequestProperties(
      com.ibm.commerce.datatype.TypedProperty arg1)
      throws Exception
      {
         iRequestProperties = arg1;
         try {
             if (// check for nulls
                 getDataBeanKeyUserId() == null)
                {
                   super.setInitKey_UserId(aUserId);
                }
   ```

```
        } catch (com.ibm.commerce.exception.ParameterNotFoundException e)
             {
             }
    }
```

Esistono altri due modi in cui è possibile impostare la chiave principale per il bean di accesso. È possibile impostarla all'esterno del bean di dati, ad esempio, nella maschera JSP. In questo caso, prima di attivare il bean di dati nella maschera JSP in maniera esplicita, richiamare il metodo set del bean di dati per la chiave principale. Ad esempio, la maschera JSP potrebbe includere codice come quello riportato di seguito (dove db è l'oggetto bean di dati):

```
db.setInitKey_UserId(/*input parameter*/)
db.activate();
```

In alternativa, la chiave principale può essere impostata in maniera diretta, ossia solo la maschera JSP contiene il metodo db.activate, quindi il gestore dei bean di dati imposta in maniera esplicita la chiave principale nel bean di accesso. Ad esempio, il codice relativo al metodo setRequestProperties del bean di dati è simile a quello riportato di seguito:

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception
      {
         iRequestProperties = arg1;
         try
             {
                super.setInitKey_UserId(aUserId);
             }
       } catch (com.ibm.commerce.exception.ParameterNotFoundException e)
           {
           }
      }
```

Tuttavia, la procedura consigliata per l'impostazione della chiave principale è quella indiretta, illustrata in 5.

Modifica del metodo get del bean di dati: Dopo aver creato il bean di dati, è necessario modificare il codice in alcuni dei metodi get generati.

Per aggiornare i metodi get, effettuare le seguenti operazioni:

1. Espandere il nuovo bean di dati per visualizzarne i campi e i metodi.
2. Selezionare il metodo **getCommandContext()** per visualizzarne il codice di origine.
   Viene visualizzato il pannello contenente il codice di origine:

   ```
   public com.ibm.commerce.comand.CommandContext getCommandContext ()
   {
      return null;
   }
   ```

3. Modificare il codice di origine in modo che venga visualizzato come riportato di seguito:

```
public com.ibm.commerce.comand.CommandContext getCommandContext ()
{
   return iCommandContext;
}
```

   Salvare il proprio lavoro (Ctrl+S).

4. Selezionare il metodo **getRequestProperties()** per visualizzarne il codice di origine.
   Viene visualizzato il pannello contenente il codice di origine:

```
public com.ibm.commerce.datatype.TypedProperty setRequestProperties()
{
   return null;
}
```

5. Modificare il codice di origine in modo che venga visualizzato come riportato di seguito:

```
public com.ibm.commerce.datatype.TypedProperty setRequestProperties()
{
   return iRequestProperties;
}
```

   Salvare il proprio lavoro (Ctrl+S).

Modifica del metodo populate(): È necessario modificare il metodo populate completando la seguente procedura:

1. Espandere il nuovo bean di dati per visualizzarne i campi e i metodi.
2. Selezionare il metodo **`populate()`** per visualizzarne il codice di origine.
   Viene visualizzato il pannello contenente il codice di origine:

```
public void populate () throws Exception {}
```

3. Modificare il codice di origine in modo che venga visualizzato come riportato di seguito:

```
public void populate() throws Exception {
   super.refreshCopyHelper();
}
```

   Salvare il proprio lavoro (Ctrl+S).

Scrittura di nuovi bean di sessione

Quando si creano nuovi bean di sessione, devono essere creati in un gruppo EJB separato dai gruppi EJB di WebSphere Commerce. Questo nuovo gruppo EJB deve essere memorizzato in un nuovo progetto separato dai progetti WebSphere Commerce. Ad esempio, è possibile creare il gruppo EJB `MyCustomBeans` nel pacchetto `com.mycompany.mycustomcode`. Separando il proprio codice personalizzato dal codice di WebSphere Commerce, l'impatto della migrazione per i rilasci futuri sarà minimo.

Il nuovo bean di sessione deve estendere la classe com.ibm.commerce.base.helpers.BaseJDBCHelper. La superclasse fornisce metodi che consentono di ottenere un oggetto connessione JDBC dall'oggetto origine dati utilizzato dal server commerce in modo che il bean di sessione partecipi alla stessa transazione degli altri bean entità. Il seguente codice di esempio mostra le funzioni fornite dalla superclasse:

```
public class mySessionBean extends com.ibm.commerce.base.helpers.BaseJDBCHelper
   implements SessionBean {

   public Object myMethod () throws javax.naming.NamingException,
      java.rmi.RemoteException, SQLException {

       ////////////////////////////////////////////////
      //  -- your logic, such as initialization --    //
      ////////////////////////////////////////////////

         try {
             // get a connection from the WebSphere Commerce data source
             makeConnection();
             PreparedStatement stmt = getPreparedStatement(
                "your sql string");
             ////////////////////////////////////////////////////////////
             // -- your logic such as set parameter into the prepared  //
             // statement --                                            //
             ////////////////////////////////////////////////////////////
                  ResultSet rs = executeQuery(stmt, false);

             ////////////////////////////////////////////////
             // -- your logic to process the result set -- //
            ////////////////////////////////////////////////

                 }
          finally {
             // return the connection to the WebSphere Commerce data source
            closeConnection();
          }

        ////////////////////////////////////////////////////
       // -- your logic to return the result ---          //
       ////////////////////////////////////////////////////

   }

}
```

Nel precedente esempio di codice, il metodo `executeQuery` utilizza due parametri di immissione. Il primo è un'istruzione preparata mentre il secondo è un indicatore booleano associato ad un'operazione di ripulitura della cache. Impostare questo indicatore su `true` se si deve ripulire, tramite il contenitore, tutti gli oggetti entità per la transazione corrente dalla cache prima di eseguire il query. Questa operazione potrebbe essere richiesta se sono stati eseguiti aggiornamenti di alcuni oggetti entità ed è necessario avviare una ricerca in

questi oggetti aggiornati. Se l'indicatore è impostato su `false`, eventuali aggiornamenti agli oggetti entità non verranno memorizzati nel database fino a quando non sarà terminata la transazione.

Si consiglia di limitare l'uso di questa operazione di ripulitura e impostare l'indicatore su `false`, tranne per i casi in cui è effettivamente richiesta. L'operazione di ripulitura richiede un'intensa elaborazione delle risorse.

Cicli di vita dell'oggetto

I bean enterprise nel modello oggetti comprendono oggetti *indipendenti* e *dipendenti*. Un oggetto indipendente ha un proprio ciclo di vita, controllato direttamente dalle richieste di creazione o di rimozione della logica aziendale che richiama l'oggetto. Gli oggetti dipendenti hanno un ciclo di vita che dipende da un altro oggetto; tale oggetto, detto anche *oggetto proprietario*, può essere a sua volta un oggetto dipendente; tuttavia, a un livello superiore della gerarchia di associazione esiste un oggetto indipendente. Quando l'oggetto proprietario viene eliminato, vengono eliminati di conseguenza anche tutti gli oggetti dipendenti. Gli elementi che vengono realmente eliminati vengono controllati attraverso le specifiche di eliminazione a cascata all'interno del database.

Ad esempio, dato un oggetto utente che restituisce un oggetto rubrica e un elenco di oggetti ordine, se l'oggetto utente viene eliminato, anche l'oggetto rubrica sarà eliminato, in quanto appartenente dell'utente, oltre a tutti gli indirizzi in essa contenuti in quanto parte della rubrica. Tuttavia, gli oggetti ordine dell'utente non vengono eliminati perché il proprietario degli ordini è un oggetto negozio e non l'oggetto utente.

Esiste un modello specifico di progettazione per la creazione di oggetti dipendenti. Il metodo create di un oggetto dipendente deve fornire almeno un riferimento all'oggetto proprietario, quindi il proprietario deve essere già presente prima che venga creato un oggetto dipendente.

Transazioni

L'architettura di Enterprise JavaBeans V1.0 specifica tre opzioni alternative dell'ora del commit relativamente allo stato dell'istanza. Tali opzioni sono descritte come opzioni A, B e C nel documento delle specifiche. Per ulteriori dettagli su queste opzioni, fare riferimento al documento delle specifiche Enterprise JavaBean V1.0 di Sun Microsystem.

Sebbene WebSphere Application Server implementi le opzioni A e C, l'opzione A assume che il database non è condiviso.

Nell'opzione C, il contenitore del bean enterprise non memorizza nella cache un'istanza "pronta" tra le transazioni. Appena completata la transazione, l'istanza viene restituita al pool delle istanze disponibili. WebSphere

Commerce utilizza l'opzione C perché il database è condiviso tra più applicazioni di WebSphere Commerce. In questa implementazione, il contenitore carica i dati permanenti per i bean entità all'inizio di ciascuna transazione e i bean entità vengono memorizzati nella cache unicamente per la durata della transazione. Il contenitore attiva più istanze di un bean entità, una per ciascuna transazione dalla quale si potrà accedere all'entità. La sincronizzazione della transazione viene eseguita dal database.

L'attributo di transazione per ciascun bean enterprise è impostato su `TX_REQUIRED`. Poiché il controller Web avvia una transazione prima di eseguire un comando che accede a un bean enterprise (attraverso il bean di accesso corrispondente), i metodi aziendali del bean enterprise vengono richiamati nel contesto di questa transazione.

Altre considerazioni sui bean entità

Find for Update

Il caso in cui più applicazioni possono accedere alla stessa riga di un database per motivi di aggiornamento della riga, è noto come *aggiornamento simultaneo*. Vi sono situazioni in cui gli aggiornamenti simultanei sono consentiti e altre in cui non sono assolutamente consigliati.

Se l'aggiornamento del database comporta una sostituzione in cui il nuovo valore non ha relazioni con il valore corrente presente nel database, gli aggiornamenti simultanei sono consentiti. Se l'aggiornamento simultaneo è consentito e più applicazioni tentano di aggiornare la stessa riga di un database, l'ultimo tentativo è quello che aggiornerà il database.

Se l'aggiornamento del database dipende dal valore corrente nel database, un aggiornamento simultaneo non è consigliato. Ad esempio, se un'applicazione sta aggiornando l'inventario del prodotto, solo un'applicazione alla volta dovrebbe essere in grado di aggiornare l'inventario.

I fattori che influiscono sulla possibilità di aggiornamento simultaneo includono i blocchi al database e i livelli di isolamento del bean enterprise.

Per impedire che una seconda applicazione aggiorni simultaneamente una riga, la prima applicazione che accede alla riga deve utilizzare l'opzione "find for update" nella riga. Quando viene utilizzata l'opzione "for update", un *blocco di scrittura* (noto anche come un blocco esclusivo) viene applicato alla riga. Con il blocco di scrittura applicato alla riga, tutte le applicazioni che tentano di accedere alla riga utilizzando l'opzione "find for update" vengono bloccate.

Se la propria applicazione consente aggiornamenti simultanei, è possibile che possa richiamare i dati senza dover bloccare la riga.

Si consideri lo scenario OrderProcess in cui UpdateInventory deve trovare tutti i prodotti inclusi in un ordine e aggiornare l'inventario di conseguenza. Poiché gli stessi prodotti possono essere inclusi in molti altri ordini, è opportuno utilizzare *find for update*; inoltre, è consigliabile utilizzarlo appena possibile nella transazione per ridurre le possibilità di condizioni di stallo. Quindi, l'algoritmo UpdateInventory può essere rappresentato dal seguente pseudo codice:

```
UpdateInventory
 find all the order items in the order
 for each order item
  fetch its inventory using "find for update"
  ...
```

In uno scenario di transazioni prolungate Business-to-Business, dove un ordine può contenere molti articoli, Find for update deve essere utilizzata appena possibile. La logica può essere la seguente:

```
find for update the inventory of all the products in an order
for each product
 if (total quantity ordered for that product < inventory)
        deduct quantity from inventory
 else
        error
```

Metodo remoto di allineamento

Poiché WebSphere Application Server non scrive nel database le modifiche apportate ai bean entità prima del commit della transazione, il database può essere temporaneamente non sincronizzato con i dati memorizzati nella cache del contenitore del bean entità.

È disponibile un metodo remoto di allineamento (nella classe `com.ibm.commerce.base.helpers.BaseJDBCHelper`) che scrive tutte le modifiche apportate alla transazione corrente (ossia accetta le informazioni dalla cache del bean enterprise) e aggiorna il database. Questo metodo remoto può essere richiamato da un comando. Utilizzare questo metodo soltanto quando è assolutamente necessario, in quanto è molto dispendioso in termini di risorse e, quindi, può compromettere il livello delle prestazioni.

Si consideri un comando di collegamento che abbia il seguente codice:

```
UserAccessBean uab = ...;
 uab.setRegisteredTimestamp(currentTimestamp);
 uab.commitCopyHelper();
```

Prima di eseguire il commit della transazione, REGISTEREDSTAMP nella tabella USER non sarà aggiornato con la data/ora corrente. L'aggiornamento viene eseguito solo al momento dell'esecuzione del commit della transazione. Il metodo di allineamento deve essere utilizzato in modo che qualsiasi query

JDBC diretta (nella stessa transazione), ad esempio, *selezionato dall'utente dove ora registrata ...*, restituisca l'utente con la data/ora di registrazione specificata.

Sicurezza dei bean enterprise

Se si utilizza WebSphere Application Server per la sicurezza dei bean enterprise, è necessario assegnare i metodi di tutti i nuovi bean enterprise al gruppo di metodi per la sicurezza `WCSMethodGroup`, servendosi della Console di gestione di WebSphere Application Server. Eseguire questa procedura quando si impiegano nuovi bean enterprise. Inoltre, se si modificano i bean entità di WebSphere Commerce esistenti, è necessario assegnare i metodi di tutti i bean entità del gruppo EJB interessato al gruppo di metodi per la sicurezza `WCSMethodGroup`. Per una descrizione del processo di impiego del codice personalizzato, fare riferimento a "Sviluppo del codice" a pagina 193.

Chiavi principali

Una chiave principale è una chiave univoca che fa parte della definizione di una tabella. Essa può essere utilizzata per distinguere un record dagli altri. Tutti i record devono avere una chiave principale. Quando viene creato un nuovo record in una tabella, è necessario generare una chiave principale univoca per tale record.

Nel modello di programmazione di WebSphere Commerce il livello permanente comprende i bean entità che interagiscono con il database. Pertanto, è possibile creare i record del database quando viene eseguita l'istanza di un bean entità. Il metodo ejbCreate per l'esecuzione dell'istanza di un bean entità potrebbe perciò comprendere una logica utile a creare una chiave principale per i nuovi record.

Quando sono necessarie le informazioni dal database per un'applicazione, vengono utilizzati direttamente i bean entità eseguendo l'istanza del bean di accesso del corrispondente bean e quindi rilevando o impostando i vari campi. L'istanza per un bean di accesso viene eseguita per un particolare record in un database (ad esempio, per un particolare profilo utente) e viene utilizzata la chiave principale per selezionare le informazioni corrette dal database.

Nelle seguenti sezioni viene descritto come creare una chiave principale univoca e come effettuare le selezioni mediante tale chiave.

Creazione di chiavi principali: Il metodo `ejbCreate` viene utilizzato per eseguire le nuove istanze di un bean entità. Tale metodo viene generato automaticamente ma comprenderà soltanto la logica per eseguire l'istanza delle chiavi principali a un valore statico.

Assicurarsi che la chiave principale abbia un valore nuovo e univoco. In questo caso, è necessario disporre di un metodo ejbCreate simile alla seguente sezione di codice:

```
Public void ejbCreate(int argMyOtherValue)
   throws javax.ejb.CreateException,
   java.rmi.RemoteException {
       //Initialize CMP fields
      MyKeyValue = com.ibm.commerce.key.ECKeyManager.
         singleton().getNextKey("nome_tabella");
      MyOtherValue = argMyOtherValue;
}
```

Nella precedente sezione del codice, il metodo getNextKey genera un valore intero univoco per la chiave principale. Il parametro di immissione *nome_tabella* relativo al metodo deve corrispondere esattamente al valore TABLENAME definito nella tabella KEYS. Verificare che i caratteri corrispondano esattamente.

Oltre a includere il codice precedente nel metodo ejbCreate, è necessario creare una voce nella tabella KEYS. Di seguito viene riportato un esempio di istruzione SQL per la creazione della voce nella tabella KEYS:

```
insert into KEYS (TABLENAME, COUNTER, KEYS_ID)
   values ("nome_tabella", 0, 1)
```

Con l'istruzione SQL precedente vengono accettati i valori predefiniti per altre colonne della tabella KEYS. Il valore relativo a COUNTER indica il valore da cui iniziare il conteggio. Specificare per KEYS_ID un qualunque valore positivo.

Se la chiave principale viene definita come tipo di dati long (BIGINT per DB2 o NUMBER per Oracle), utilizzare il metodo `getNextKeyAsLong`.

Selezione mediante la chiave principale: All'interno di un bean di accesso è necessario selezionare il record del database utilizzando la chiave principale. La seguente sezione del codice illustra come eseguire questa selezione. Inoltre, include una logica aggiuntiva, che verrà illustrata di seguito.

```
UserProfileAccessBean abUserProfile = new UserProfileAccessBean();
abUserProfile.setInitKey_UserId(getUserId().toString());
abUserProfile.refreshCopyHelper();
```

Nella prima riga della sezione del codice precedente viene eseguita l'istanza di UserProfileAccessBean detta ″abUserProfile″. Nella seconda riga viene impostata la chiave principale nel bean di accesso. La convenzione di denominazione `setInitKey_`*xxx* (dove *xxx* indica il nome del campo della chiave principale) viene utilizzata da VisualAge per Java per denominare i metodi set per le chiavi principali. Quando si crea un'istanza del bean di accesso, è necessario assicurare che tutti i campi impostati da un metodo

setInitKey_xxx vengano inizializzati prima di utilizzare il metodo refreshCopyHelper. L'ordine in cui i metodi setInitKey_xxx vengono richiamati è irrilevante.

Una volta richiamati tutti i metodi setInitKey_xxx, sono stati inizializzati tutti i campi richiesti ed è quindi possibile utilizzare il metodo refreshCopyHelper per recuperare le informazioni dal database.

Se vengono aggiornati valori all'interno della cache locale del bean di accesso, è necessario includere anche una chiamata commitCopyHelper per aggiornare il database con le ultime informazioni. Ad esempio, nel caso in cui in seguito al recupero dei dati mediante il metodo refreshCopyHelper venga aggiornato il nome di un cliente (impostando il valore del nome), sarà necessario richiamare `abUserProfile.commitCopyHelper()` per aggiornare il database con le nuove informazioni.

Utilizzo dei bean entità

Un programma che si serve di bean enterprise deve utilizzare l'interfaccia JNDI (Java Naming and Directory Interface) nonché le interfacce home e remota dei bean enterprise. Per semplificare il modello di programmazione, viene generato un bean di accesso per ciascun bean enterprise. Quando si creano i propri bean enterprise, per generare questo bean di accesso utilizzare gli strumenti di VisualAge per Java.

I comandi di WebSphere Commerce interagiscono con i bean di accesso anziché direttamente con i bean entità. Come illustrato dal diagramma, l'utilizzo del bean di accesso offre i seguenti vantaggi:

- Un'interfaccia di programmazione più semplice. Il bean di accesso si comporta come un bean Java e nasconde dai client tutte le interfacce di programmazione specifiche del bean enterprise, come JNDI, le interfacce home e remota.
- Durante il runtime, il bean di accesso memorizza nella cache l'oggetto home del bean enterprise perché le ricerche nell'oggetto home sono dispendiose in termini di tempo e di utilizzo delle risorse.
- Il bean di accesso implementa un oggetto copyHelper che riduce il numero delle chiamate al bean enterprise quando utilizza gli attributi del bean enterprise dei comandi get e set. Di conseguenza, è necessaria una sola chiamata al bean enterprise durante la lettura o la scrittura di più attributi del bean enterprise.

Il seguente diagramma visualizza l'interazione tra i comandi, i bean di accesso, i bean entità e il database.

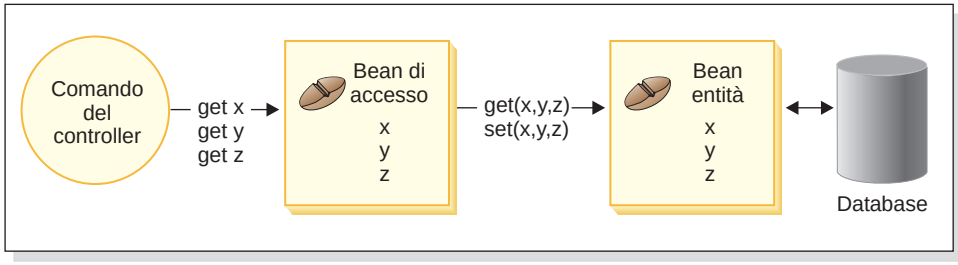


Figura 18.

Considerazioni sul database

Durante la personalizzazione della propria applicazione e-commerce, è possibile creare nuove tabelle del database. Quando si creano queste tabelle, si consiglia di seguire una serie di convenzioni in modo che le tabelle siano congruenti con le tabelle di WebSphere Commerce.

Considerazioni sulla denominazione di oggetti dello schema del database

Le sezioni successive forniscono una guida per la denominazione di oggetti di schema del database.

Convenzioni di denominazione per tabelle e viste

Il seguente elenco fornisce una guida per l'assegnazione di nomi alle nuove tabelle e viste:

- Per evitare conflitti tra nomi (nomi duplicati) con tabelle e viste di WebSphere Commerce per futuri rilasci, il primo carattere per il nome della tabella o della vista dovrebbe essere `X`. Ad esempio, `XMYTABLE`.
- Il nome della tabella o della vista non deve contenere più di 10 caratteri. Se il nome scelto supera questo limite, ridurre la lunghezza del nome rimuovendo le vocali dalla fine del nome, fino a raggiungere i 10 caratteri.
- Il nome della tabella o della visualizzazione non deve contenere caratteri speciali, come "_", "+", "$", "%" o spazi vuoti.
- Non utilizzare le parole riservate per il database come nomi di tabelle o di visualizzazioni.
- I nomi delle visualizzazioni devono terminare con `VW`.
- I nomi di tabelle o di visualizzazioni devono essere nomi singolari.

Convenzioni di denominazione per colonne

Il seguente elenco fornisce una guida per l'assegnazione dei nomi alle colonne presenti nella nuova tabella:

- Per evitare conflitti tra nomi (nomi duplicati) con le colonne presenti nelle tabelle WebSphere Commerce per futuri rilasci, il primo carattere per il nome della colonna dovrebbe essere `X`. Ad esempio, `XMYCOLUMN`.
- Il nome della colonna non deve contenere più di 18 caratteri. Se il nome scelto supera tale limite, ridurre la lunghezza del nome rimuovendo le vocali dalla fine del nome, fino a raggiungere i 18 caratteri.
- I nomi delle colonne (diversi dalla chiavi esterne) non deve contenere caratteri speciali, come "_", "+", "$", "%" o spazi vuoti.
- Non utilizzare le parole riservate per il database come i nomi delle colonne.
- Per i nomi delle colonne è possibile utilizzare parole combinate mediante una combinazione vocale attiva. Ad esempio, `COMBINERESULT`.
- Le colonne di chiavi principali generate dovrebbero essere ridenominate come *`id_tabella`*. Ad esempio, la chiave principale per la tabella USERS è `USERS_ID`.
- I nomi delle colonne di chiavi esterne generate non dovrebbero essere modificate.
- Se le colonne sono utilizzate per personalizzazioni future, dovrebbero essere denominate `field`*`x`* dove *`x`* è una cifra numerica che inizia da 1.

Convenzioni di denominazione per gli indici

Il seguente elenco fornisce una guida per l'assegnazione dei nomi agli indici delle nuove tabelle:

- Il nome dell'indice non deve contenere più di 18 caratteri.
- Il nome dell'indice non deve contenere spazi vuoti.
- Il nome dell'indice non deve contenere parole riservate per il database.
- Un indice non-univoco dovrebbe essere denominato come `I_`*`tabellax`* dove *`tabella`* è il nome della tabella e *`x`* è un numero, che comincia con 1. Ad esempio, un indice non-univoco per la tabella USERS è `I_USERS1`.
- Un indice univoco dovrebbe essere denominato come `UI_`*`tabellax`* dove *`tabella`* è il nome della tabella e *`x`* è un numero, che inizia da 1. Ad esempio, un indice univoco per la tabella USERS è `UI_USERS1`.
- Le dimensioni totali dell'indice non devono essere superiori ai 254 byte.
- Il nome dell'indice deve essere univoco all'interno dello schema del database.

Convenzioni di denominazione per chiavi principali

Il seguente elenco fornisce una guida per l'assegnazione dei nomi alle chiavi principali delle nuove tabelle:

- Il nome della chiave principale non deve contenere più di 18 caratteri.
- Il nome della chiave principale non deve contenere spazi vuoti.
- Il nome della chiave principale non deve contenere parole riservate per il database.

- La chiave principale dovrebbe essere denominata come P_*tabella* dove *tabella* è il nome della tabella. Ad esempio, la chiave principale per la tabella USERS è P_USERS.
- Il nome della chiave principale deve essere univoco all'interno dello schema del database.

Convenzioni di denominazione per chiavi esterne

Il seguente elenco fornisce una guida per l'assegnazione dei nomi per le chiavi esterne delle nuove tabelle:

- Il nome della chiave esterna non deve superare i 18 caratteri.
- Il nome della chiave esterna non deve contenere spazi vuoti.
- Il nome della chiave esterna non deve contenere parole riservate al database.
- La chiave esterna deve essere denominata come F_*tabella* dove *tabella* è il nome della tabella. Ad esempio, la chiave esterna per la tabella USERS dovrebbe essere F_USERS1.
- Il nome della chiave esterna deve essere univoco all'interno dello schema del database.

Convenzioni di denominazione per i trigger del database

Il seguente elenco fornisce una guida da seguire per la denominazione dei trigger del database:

- Il nome del trigger del database non deve superare i 18 caratteri.
- Il nome del trigger del database non deve contenere spazi vuoti.
- Il nome del trigger del database non deve contenere parole di database riservate.
- Il trigger del database deve essere denominato come T_*tabella* dove *tabella* è il nome della tabella. Ad esempio, il nome del trigger del database per la tabella USERS è T_USERS1.
- Il nome del trigger del database deve essere univoco all'interno dello schema del database.

Considerazioni sul tipo di dati di colonna del database

Questa sezione introduce i tipi di dati delle colonne che possono essere utilizzati quando si creano nuove tabella. Le descrizioni dei vari tipi di dati si basano sulla terminologia DB2. "Differenza di tipi di dati tra database" a pagina 89 descrive le differenze se si utilizza un database diverso.

BIGINT

È un numero intero a 64-bit compreso tra -9223372036854775807 e 9223372036854775807. Il valore di INTEGER al contrario è solo la metà di BIGINT.

INTEGER

È un numero intero a 32-bit compreso tra -2147483647 e 2147483647. In generale, anziché BIGINT il tipo di dati numerici finiti predefinito dovrebbe essere INTEGER. A meno che non vi sia una ragione particolare per l'utilizzo di BIGINT, per motivi legati alle prestazioni si consiglia di utilizzare INTEGER come tipo di dati numerici. Un genere, il tipo di dati BIGINT viene utilizzato da una chiave generata dal sistema.

Si consiglia di non utilizzare i tipi di dati SMALLINT o SHORT, in quanto vengono associati a tipi di dati Java non oggetto che possono causare problemi durante l'esecuzione di istanze di alcuni bean enterprise.

TIMESTAMP
È un valore costituito da sette parti (anno, mese, giorno, ora, minuto, secondo e microsecondi) che indica una data e un'ora, ad eccezione del fatto che l'ora include una specifica frazionata di microsecondi. Internamente, la data/ora è rappresentata da una stringa di 10 byte, ciascuno dei quali è composto da due cifre decimali. I primi 4 byte rappresentano la data, i successivi 3 byte l'ora e gli ultimi 3 i microsecondi.

CHAR
È una stringa di caratteri a lunghezza fissa di lunghezza INTEGER, che può contenere da 1 a 254 caratteri. Se non viene specificata la lunghezza viene assunto come valore 1 carattere. Poiché CHAR è una colonna di database a lunghezza fissa, qualsiasi spazio dei caratteri finali non utilizzati viene modificato in spazi bianchi. Per motivi di prestazioni, non si consiglia di utilizzare il tipo di dati CHAR poiché CHAR non è flessibile e la lunghezza non può essere modificata successivamente. Se la colonna della stringa contiene meno di 64 caratteri in lunghezza e viene regolarmente richiamata o aggiornata, utilizzare CHAR per ottenere prestazioni migliori.

VARCHAR
È una stringa di caratteri a lunghezza variabile, con un numero intero con lunghezza massima compresa tra 1 e 32672. Tuttavia, a differenza di CHAR in cui i dati della colonna sono memorizzati con la tabella, VARCHAR viene rappresentata internamente come un indicatore di riferimento all'interno di una pagina del database. Quindi, la lunghezza di una colonna VARCHAR può essere modificata in qualsiasi momento dopo la creazione.

LONG VARCHAR
È una stringa di caratteri di lunghezza variabile che può essere utilizzata se non è possibile creare VARCHAR all'interno della stessa pagina del database. LONG VARCHAR è molto simile a VARCHAR ad eccezione del fatto che può essere distribuito su più pagine del

database. Limitare l'uso del tipo di dati LONG VARCHAR solo a quei casi in cui è assolutamente necessario poiché gli oggetti LONG VARCHAR sono di norma molto dispendiosi in termini di prestazioni.

CLOB È una stringa di caratteri di lunghezza variabile che può essere utilizzata se la lunghezza della colonna richiede il superamento del limite di 32KB di LONG VARCHAR. La lunghezza di un oggetto CLOB può raggiungere 1 GB senza modificare la configurazione del database. I dati di testo che sono memorizzati come CLOB vengono convertiti in modo appropriato quando si spostano tra sistemi diversi.

BLOB È una stringa di caratteri binari a lunghezza variabile che memorizza i dati non strutturati nel database. Gli oggetti BLOB possono memorizzare fino a 4 GB di dati binari. In genere, si consiglia di non utilizzare BLOB come tipo di dati della colonna a meno che non risulti assolutamente necessario. In termini di prestazioni, un oggetto BLOB viene considerato uno dei più dispendiosi per qualsiasi tipo di database.

DECIMAL(20,5)
Questo tipo di dati viene definito in modo particolare per l'utilizzo con i numeri a cifre decimali fissi, quali le unità di valuta. Per altri numeri decimali a virgola mobile è possibile utilizzare FLOAT.

Differenza di tipi di dati tra database

La seguente tabella elenca i tipi di dati utilizzati nello schema del database WebSphere Commerce e mostra i tipi di dati corrispondenti per le implementazioni di database differenti.

Oggetto JDBC	Windows AIX Solaris Linux **DB2**	Windows AIX Solaris **Oracle®**	400 **DB2**
Hashtable	BLOB()	BLOB	BLOB()
Timestamp	TIMESTAMP	DATE	TIMESTAMP
Integer	INTEGER	INTEGER	INTEGER
BigDecimal	DECIMAL(,)	DECIMAL(,)	DECIMAL(,)
Long	BIGINT	NUMBER	BIGINT
Double	FLOAT	NUMBER	FLOAT
String	CHAR()	VARCHAR2()	GRAPHIC() CCSID 13488

Oggetto JDBC	Windows AIX Solaris Linux **DB2**	Windows AIX Solaris **Oracle®**	400 **DB2**
byte[]	CHAR() per dati di bit	RAW()	CHAR() per dati di bit
String	VARCHAR()	VARCHAR2()	VARGRAPHIC() CCSID 13488
String	LONG VARCHAR	VARCHAR2() (Per ulteriori dettagli, consultare la seguente tabella)	VARGRAPHIC(4000) ALLOCATE() CCSID 13488
byte[]	LONG VARCHAR per dati di bit	LONG RAW	VARCHAR(8000) ALLOCATE() for bit data
String	CLOB()	CLOB()	DBCLOB() CCSID 13488

Nota:

Poiché esiste una percentuale di successi non coerente quando il driver JDBC Oracle gestisce le informazioni di tipo di dati LONG, è consigliabile non utilizzare il tipo di dati LONG tutte le volte che è possibile. In questi casi, l'errore più comune è l'errore "Il flusso è stato già chiuso".

Se occorre utilizzare questo tipo di dati, solo una colonna per database può utilizzare il tipo LONG. Inoltre, quando si crea un'istruzione select, non inserire la colonna LONG come primo o ultimo elemento dell'istruzione. Inoltre, per le operazioni con un carico notevole, è consigliabile non associare questa colonna a un campo a CMP in un bean entità, ma utilizzare un bean di sessione per eseguire i recuperi e gli aggiornamenti in questa colonna.

Capitolo 4. Controllo accessi

Controllo accessi

Il modello del controllo accessi di un'applicazione WebSphere Commerce si basa su tre concetti fondamentali: utenti, azioni e risorse. Gli utenti sono le persone che utilizzano il sistema. Le risorse sono le entità gestite dalle applicazioni. Ad esempio, le risorse possono essere i prodotti, i documenti o gli ordini. Anche i profili utente che rappresentano persone sono risorse. Le azioni sono le attività che gli utenti possono eseguire sulle risorse. Il controllo accessi è il componente delle applicazioni e-commerce che determina se un utente può eseguire una determinata azione su una specifica risorsa.

In un'applicazione WebSphere Commerce vi sono due livelli di controllo accessi. Il primo livello del controllo accessi viene eseguito da WebSphere Application Server. In questo modo, WebSphere Commerce utilizza WebSphere Application Server per proteggere i servlet e i bean enterprise. Il secondo livello è il sistema di controllo accessi di WebSphere Commerce.

La struttura di controllo accessi di WebSphere Commerce utilizza le politiche del controllo accessi per determinare se un utente può eseguire una determinata azione su una specifica risorsa. Tale struttura per il controllo accessi garantisce un controllo accessi capillare. Questa struttura non sostituisce ma viene utilizzata insieme al controllo accessi fornito da WebSphere Application Server.

Panoramica sulla protezione delle risorse in WebSphere Application Server

Le seguenti risorse di WebSphere Commerce sono protette mediante il controllo accessi di WebSphere Application Server:

- Bean entità
  Questi bean modellano gli oggetti in un'applicazione e-commerce. Si tratta di oggetti distribuiti a cui è possibile accedere dai client remoti.
- Maschere JSP
  WebSphere Commerce utilizza le maschere JSP per le pagine di visualizzazione. Ciascuna maschera JSP può contenere uno o più bean di dati che richiamano i dati dai bean entità. I client possono richiedere le pagine JSP creando una richiesta URL.
- Comandi di controller e di visualizzazione
  I client possono richiedere i comandi di controller e di visualizzazione creando richieste URL. Inoltre, una pagina di visualizzazione può contenere un collegamento a un'altra pagina utilizzando il nome file JSP o il nome visualizzazione, come registrato nella tabella VIEWREG.

WebSphere Commerce Server è generalmente configurato per l'utilizzo dei seguenti percorsi Web:

- `/webapp/wcs/stores/servlet/*`
  Questo percorso viene utilizzato per le richieste da effettuare al servlet delle richieste.
- `/webapp/wcs/stores/*.jsp`
  Questo percorso viene utilizzato per le richieste da effettuare al servlet JSP.

Il diagramma riportato di seguito mostra il percorso che le richieste dovrebbero potenzialmente seguire per accedere alle risorse di WebSphere Commerce, in base alla configurazione del percorso Web sopra riportato.

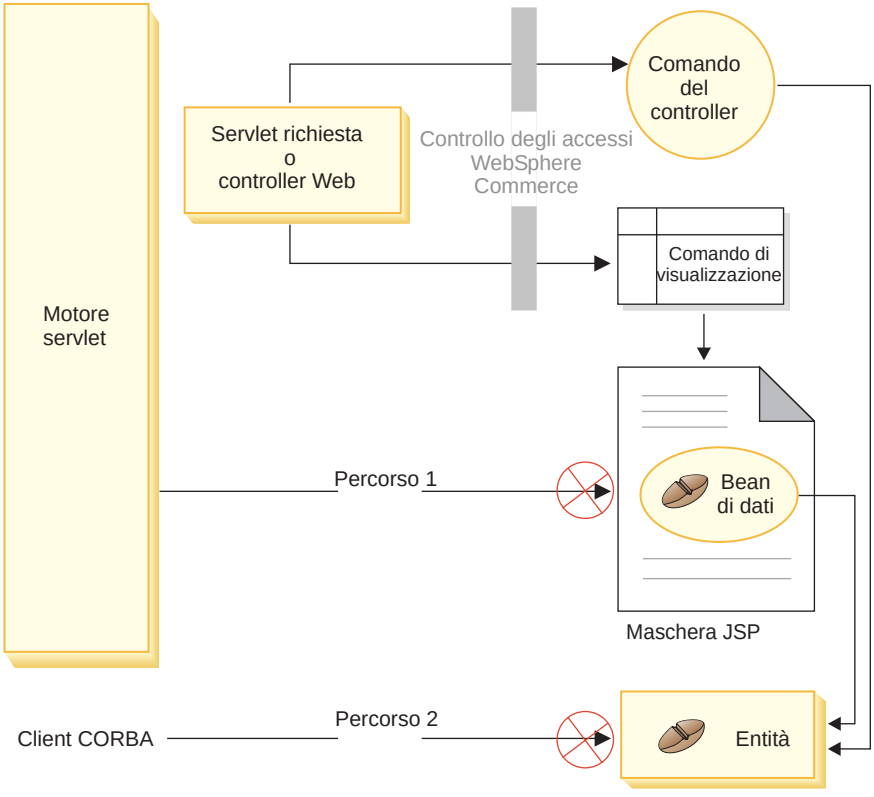


Figura 19.

Tutte le richieste valide devono essere indirizzate al servlet delle richieste, che a sua volta le indirizza al controller Web. Il controller Web implementa il controllo accessi per i comandi di controller e le visualizzazioni. I percorsi Web mostrati sopra, tuttavia, consentono a utenti malintenzionati di accedere direttamente alle maschere JSP (percorso 1) e ai bean entità (percorso 2). Per proteggersi da questi attacchi, è necessario rifiutarli durante il runtime.

Per impedire l'accesso diretto alle maschere JSP e ai bean entità, servirsi di uno dei seguenti approcci:

Sicurezza WebSphere Application Server

WebSphere Application Server fornisce alcune funzioni per la sicurezza. Utilizzando questo approccio, tutti i metodi del bean enterprise e le maschere JSP vengono configurate per essere richiamate solo da System Identity. Per accedere a tali risorse di WebSphere Commerce, una richiesta URL deve essere inoltrata al servlet delle richieste che imposta System Identity sul thread corrente prima di inviarla al controller Web. Il controller Web, quindi, si assicura che il chiamante disponga dell'autorizzazione richiesta prima di inviare la richiesta al comando di controller o della visualizzazione corrispondente. Tutti i tentativi di accedere direttamente alle maschere JSP e ai bean entità (vale a dire, senza utilizzare il controller Web) sono rifiutati dal componente per la sicurezza WebSphere Application Server.

Per ulteriori informazioni sulla configurazione di WebSphere Application Server in modo da proteggere le risorse di WebSphere Commerce, fare riferimento a *WebSphere Commerce - Guida all'installazione*. Per informazioni sulla sicurezza all'interno di WebSphere Application Server, fare riferimento alla sezione Responsabile di sistema nella documentazione WebSphere Application Server.

Per ulteriori informazioni sulla configurazione della sicurezza WebSphere Application Server per metodi contenuti nei bean enterprise personalizzati, consultare "Assemblaggio dei nuovi bean enterprise nell'enterprise application" a pagina 371 e "Assemblaggio dei bean enterprise modificati nell'enterprise application" a pagina 376.

Protezione firewall

Quando un WebSphere Commerce Server è in esecuzione protetto da firewall, i client Internet non sono in grado di accedere direttamente ai bean entità. Utilizzando questo approccio, la protezione per le maschere JSP viene fornita dal bean di dati incluso nella pagina. Il bean di dati viene attivato dal gestore dei bean di dati. Il gestore dei bean di dati rileva se la maschera JSP è stata inoltrata da un comando di visualizzazione. In caso di risposta negativa, viene inviata un'eccezione e la richiesta per la maschera JSP viene rifiutata.

Introduzione alle politiche di controllo accessi di WebSphere Commerce

Il modello di controllo accessi di WebSphere Commerce è basato sul rafforzamento delle politiche di controllo accessi. Le politiche di controllo accessi consentono alle regole di controllo accessi di poter essere estrapolate da un codice della logica aziendale, rimuovendo pertanto la necessità di

inserire istruzioni di controllo all'interno del codice. Ad esempio, non è necessario includere un codice simile a quello riportato di seguito:

```
if (user.isAdministrator())
   then {}
```

Le politiche di controllo accessi vengono applicate dal gestore delle politiche. In generale, quando un utente prova ad accedere a risorse protette, il gestore delle politiche del controllo accessi determina quale politica è possibile applicare a questa risorsa, quindi, in base alla politica applicabile, determina se l'utente può accedere alle risorse richieste.

Una politica di controllo accessi è una politica basata su quattro elementi e che viene memorizzata nella tabella ACPOLICY. Ogni politica di controllo accessi presenta questo formato:

```
AccessControlPolicy [UserGroup, ActionGroup, ResourceGroup, Relationship]
```

Gli elementi nella politica di controllo accessi a quattro elementi specificano che un utente membro di un determinato gruppo può effettuare operazioni all'interno del gruppo di azioni sulle risorse appartenenti ad un determinato gruppo di risorse fin quando vengono rispettate le condizioni specificate nelle relazioni con la risorsa stessa. Ad esempio, `[AllUsers, UpdateDoc, doc, creator]` specifica che tutti gli utenti possono aggiornare un documento se di questo documento sono i creatori.

Il gruppo utenti è un particolare tipo di gruppo membri definito nella tabella di database MBRGRP. Un gruppo utenti deve essere associato al tipo di gruppo membri `-2`. Il valore `-2` rappresenta un gruppo di accesso ed è definito nella tabella MBRGRPTYPE. L'associazione tra il gruppo utenti e il tipo di gruppo membri viene memorizzata all'interno della tabella MBRGRPUSG.

L'appartenenza di un utente a un particolare gruppo di utenti può essere dichiarata in maniera esplicita o implicita. Una dichiarazione esplicita viene eseguita se nella tabella MBRGRPMBR viene specificato che l'utente appartiene a un particolare gruppo di membri. Una dichiarazione di appartenenza implicita viene eseguita quando l'utente soddisfa una particolare condizione (ad esempio, tutti gli utenti ricoprono il ruolo di direttore prodotti), specificata nella tabella MBRGRPCOND. Inoltre si possono verificare condizioni combinate (ad esempio, tutti gli utenti che ricoprono il ruolo di direttori prodotti e che hanno sempre ricoperto questo ruolo negli ultimi 6 mesi) o esclusioni esplicite.

La maggior parte delle condizioni in base alle quali è possibile includere un utente all'interno di un gruppo, si basa sull'adempimento di un particolare ruolo da parte dell'utente. Ad esempio, è possibile applicare una particolare politica di controllo accessi che consenta agli utenti che ricoprono il ruolo di

direttori prodotti, di eseguire operazioni di gestione del catalogo. In questo caso, qualsiasi utente a cui è stato assegnato il ruolo di direttore prodotti nella tabella MBRROLE viene implicitamente incluso del gruppo utenti.

Per ulteriori dettagli sul sottosistema del gruppo membri, consultare la guida in linea di WebSphere Commerce.

L'elemento ActionGroup proviene dalla tabella ACACTGRP. Un gruppo di azioni fa riferimento a un gruppo di azioni specificato esplicitamente. L'elenco di tutte le azioni è memorizzato nella tabella ACACTION e la relazione che esiste tra ogni azione e il relativo gruppo (o gruppi) azione è memorizzata nella tabella ACACTACTGP. Un esempio di gruppo di azioni è ″OrderWriteCommands″. Questo gruppo di azioni include le seguenti azioni che vengono utilizzate per aggiornare gli ordini:

- com.ibm.commerce.order.commands.OrderDeleteCmd
- com.ibm.commerce.order.commands.OrderCancelCmd
- com.ibm.commerce.order.commands.OrderProfileUpateCmd
- com.ibm.commerce.order.commands.OrderUnlockCmd
- com.ibm.commerce.order.commands.OrderScheduleCmd
- com.ibm.commerce.order.commands.ScheduledOrderCancelCmd
- com.ibm.commerce.order.commands.ScheduledOrderProcessCmd
- com.ibm.commerce.order.commands.OrderItemAddCmd
- com.ibm.commerce.order.commands.OrderItemDeleteCmd
- com.ibm.commerce.order.commands.OrderItemUpdateCmd
- com.ibm.commerce.order.commands.PayResetPMCmd

Un gruppo di risorse è un meccanismo per raggruppare insieme particolari tipi di risorsa. L'appartenenza a un gruppo di risorse può essere specificata in uno dei seguenti modi:

- Utilizzando la colonna delle condizioni nella tabella ACRESGRP
- Utilizzando la tabella ACRESGPRES

In molti casi, è sufficiente utilizzare la tabella ACRESGPRES per associare risorse al gruppo risorse. Utilizzando questo metodo, le risorse vengono definite nella tabella ACRESCGRY mediante il loro nome classe Java. Quindi, queste risorse vengono associate ai relativi gruppi di risorse (tabella ACRESGRP) utilizzando la tabella di associazione ACRESGPRES. Nei casi in cui il solo nome classe Java non è sufficiente per definire i membri appartenenti ad un gruppo di risorse (ad esempio, se occorre limitare ulteriormente gli oggetti della classe sulla base di un attributo della risorsa), è possibile definire completamente il gruppo di risorse utilizzando la colonna

delle condizioni della tabella ACRESGRP. Si noti che per eseguire questo raggruppamento sulla base di un attributo, la risorsa deve anche implementare l'interfaccia Groupable.

Nel seguente diagramma viene illustrato un esempio di specifica di raggruppamento delle risorse. In questo esempio, il gruppo di risorse `10023` include tutte le risorse associate ad esso nella tabella ACRESGPRES. Il gruppo di risorse `10070` viene definito utilizzando la colonna del campo delle condizioni nella tabella ACRESGRP. Tale gruppo di risorse include istanze dell'interfaccia remota Orders il cui stato è `"Z"` (che specifica un elenco di requisiti condivisi).

Nota: È possibile ricevere maggiori dettagli sulle informazioni XML per la colonna Conditions della tabella ACRESGRP nel manuale *WebSphere Commerce Access Control Guide*.

ACRESGRP

AcResGrp_Id	GrpName	Conditions
10023	AccountRepresentatives CmdResourceGroup	null
10070	SharedRequisitionList ResourceGroup	<profile> <andListCondition> <simpleCondition> <variable name="Status"/> <operator name="="/> <value data="Z"/> </simpleCondition> <simpleCondition> <variable name="classname"/> <operator name="="/> <value data="com.ibm.commerce.order.objects.Order"/> </simpleCondition> </andListCondition> </profile>

ACRESGRPES

AcResGrp_Id	AcResCgry_Id
10023	10246
10023	10247
10023	10248
10023	10249
10023	10250

ACRESCGRY

AcResCgry_Id	ResClassname
10246	com.ibm.commerce.contract.commands.ContractCreateCmd
10247	com.ibm.commerce.contract.commands.ContractCreateCmd
10248	com.ibm.commerce.contract.commands.ContractCreateCmd
10249	com.ibm.commerce.contract.commands.ContractCreateCmd
10250	com.ibm.commerce.contract.commands.ContractCreateCmd

Figura 20.

La colonna MEMBER_ID delle tabelle ACACTGRP, ACRESGRP e ACRELGRP deve avere un valore di `-2001` (Organizzazione Root).

La politica di controllo accessi può includere facoltativamente un elemento Relationship oppure RelationshipGroup come quarto elemento.

Se la politica di controllo accessi utilizza un elemento Relationship, esso proviene dalla tabella ACRELATION. Se, dall'altro lato, include un elemento RelationshipGroup, esso proviene dalla tabella ACRELGRP. Non è richiesto di includere un elemento, ma se viene incluso uno non è possibile includere l'altro. Una specifica RelationshipGroup della tabella ACRELGRP assume una priorità sull'informazione Relationship della tabella ACRELATION.

Nella tabella ACRELATION vengono specificati i tipi di relazione esistenti tra gli utenti e le risorse. Alcuni esempi di tipi di relazione includono il creatore, l'utente che inoltra e il proprietario. Un esempio sull'utilità dell'elemento di relazione consiste nel garantire che il creatore di un ordine possa sempre aggiornare l'ordine.

Nella tabella ACRELGRP vengono specificati i tipi di gruppi di relazione che possono essere associati a risorse particolari. Un gruppo di relazione consiste nel raggruppamento di uno o più catene di relazioni. Una catena di relazioni è costituita da una serie di una o più relazioni. Un esempio di un gruppo di relazione può essere un utente definito come creatore della risorsa e anche appartenente all'entità organizzativa acquirente alla quale si fa riferimento nella risorsa.

La specifica del gruppo di relazione risulta una parte facoltativa della politica di controllo accessi. Viene generalmente utilizzata se sono stati creati dei comandi personalizzati e tali comandi non sono subordinati a determinati ruoli. In questi casi si potrebbe decidere di rafforzare la relazione tra l'utente e la risorsa. Di solito, se i comandi sono subordinati a determinati ruoli, ciò avviene tramite l'elemento UserGroup della politica del controllo accessi piuttosto che utilizzando l'elemento Relationship.

Un altro importante concetto correlato alle politiche del controllo accessi è quello del *proprietario* di una politica del controllo accessi. Quest'ultimo è l'entità organizzativa alla quale appartiene la politica del controllo accessi. È molto importante conoscere il proprietario di una politica del controllo accessi, in quanto una politica del controllo accessi può essere applicata unicamente alle risorse che appartengono al proprietario della politica.

Per ogni risorsa in questione, il gestore delle politiche del controllo accessi applica le politiche di controllo accessi che appartengono all'entità organizzativa oppure alle entità organizzative precedenti a una gerarchia di membri, fino a quando non viene individuata una politica che conceda un'autorizzazione oppure fino a quando non vengono verificate tutte le politiche e nessuna concede un'autorizzazione.

Si consideri il seguente diagramma che illustra una gerarchia di membri.

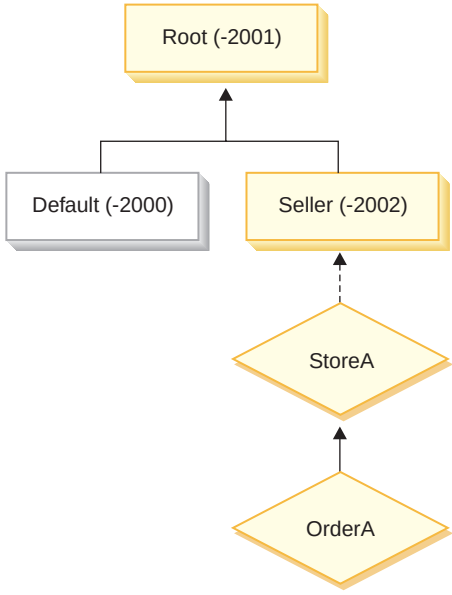


Figura 21.

Per la risorsa "OrderA", è possibile applicare qualsiasi politica del controllo accessi che appartenga all'organizzazione Seller o Root. Se il gestore delle politiche di controllo accessi individua una politica che appartiene ad una delle organizzazioni che concedono un'autorizzazione all'utente (basata sui quattro elementi della politica di controllo accessi), interrompe immediatamente la ricerca tra le politiche. Tuttavia, se non viene rilevata alcuna politica del controllo accessi che appartiene a quelle organizzazioni che consentano all'utente di eseguire un'operazione sulle risorse protette, l'accesso viene negato.

Gruppi di relazione

Un gruppo di relazione consente di specificare più relazioni. Una relazione può esistere direttamente tra un utente e la risorsa in questione oppure può essere una catena di relazioni che in maniera indiretta pone in relazione l'utente e la risorsa.

Nota: È importante sapere che per le seguenti sezioni relative ai gruppi di relazione le uniche organizzazioni disponibili in WebSphere Commerce Professional Edition sono: RootOrganization, DefaultOrganization e SellerOrganization. Esempi che fanno riferimento ad altre organizzazioni si applicano solo a WebSphere Commerce Business Edition.

Confronto tra relazioni e gruppi di relazione: Le politiche di controllo accessi possono stabilire che un utente debba soddisfare una particolare

relazione con riferimento alla risorsa da accedere oppure che debba soddisfare le condizioni specificate in un gruppo di relazione.

Nella maggior parte dei casi, quando si specifica una relazione vengono soddisfatti i requisiti del controllo accessi per la propria applicazione. Se tuttavia la politica richiede di specificare non una relazione diretta tra utente e risorsa ma una serie di relazioni tra l'utente e la risorsa, è necessario utilizzare un gruppo di relazioni.

Ad esempio, se occorre specificare un'associazione tra un utente e un'organizzazione acquirente nella quale la relazione richiede un particolare ruolo da parte dell'utente per quella organizzazione oppure che l'utente sia un membro dell'organizzazione acquirente, allora è necessario utilizzare un gruppo di relazione e una catena di relazioni.

Se occorre soltanto rafforzare un'associazione che sia direttamente tra l'utente e la risorsa in questione, si può utilizzare una semplice relazione. Ad esempio, questo potrebbe essere il caso in cui è necessario rafforzare la condizione che l'utente sia il creatore della risorsa.

Se vengono associate più relazioni semplici, ad esempio, l'utente deve essere il creatore *oppure* la persona che inoltra, allora ciò diventa una catena di relazioni e occorre utilizzare un gruppo di relazione. Quest'associazione di semplici relazioni potrebbe verificarsi quando si utilizza WebSphere Commerce Professional Edition o WebSphere Commerce Business Edition.

Informazioni generali sui gruppi di relazione: Una catena di relazioni è costituita da una serie di una o più relazioni. La lunghezza di una catena di relazioni viene determinata dal numero di relazioni che essa contiene. Tale numero può essere determinato analizzando il numero di elementi `<parameter name="aName" value="aValue" />` presenti nella rappresentazione XML della catena di relazioni.

Solo l'ultimo elemento `<parameter name="Relationship" value="aValue" />` deve essere gestito dal metodo fulfills() della risorsa. Gli altri attributi vengono gestiti internamente dal gestore della politica di controllo accessi.

Quando una catena di relazioni ha una lunghezza pari a 2, il primo elemento `<parameter name="aName" value="aValue" />` rappresenta la relazione tra un utente e un'entità organizzativa. L'ultimo elemento `<parameter name="aName" value="aValue" />` rappresenta la relazione tra l'entità organizzativa e la risorsa.

Se occorre definire i gruppi di relazione, ciò è possibile definendo le informazioni del gruppo di relazione presenti nel file XML. Si può modificare il file `defaultAccessControlPolicies.xml` oppure creare il proprio file XML.

Per ulteriori dettagli sulla creazione delle informazioni basate su XML, consultare il manuale *WebSphere Commerce Access Control Guide*.

Nelle seguenti sezioni vengono illustrati esempi di diversi tipi di gruppi di relazione.

Gruppi di relazione composti da una singola catena di relazioni: Business Come parte di una politica di controllo accessi, può essere richiesto di rafforzare la condizione che un utente appartenga ad una determinata entità organizzativa (come la BuyingOrganizationalEntity) della risorsa. Ciò richiede la creazione di un gruppo di relazione composto da una catena di relazioni con lunghezza pari a due. La catena di relazioni viene definita di lunghezza ″due″ poiché formata da due relazioni separate. La prima relazione è composta dall'utente e dalla sua entità organizzativa principale (parent). L'utente è considerato ″secondario″ (child) in tale relazione. Per la seconda relazione il gestore delle politiche di controllo accessi verifica se l'entità organizzativa principale (parent) soddisfa la relazione BuyingOrganizationalEntity con la risorsa. In altri termini, restituisce il valore ″true″ se risulta l'entità organizzativa acquirente della risorsa.

La seguente sezione XML viene estratta dal file `defaultAccessControlPolicies.xml` ed illustra come definire questo tipo di gruppo di relazione:

```
<RelationGroup Name="MemberOf->BuyerOrganizationalEntity"
      OwnerID="RootOrganization">
   <RelationCondition><![CDATA[
      <profile>
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="HIERARCHY" value="child"/>
            <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
         </openCondition>
      </profile>
   ]]></RelationCondition>
</RelationGroup>
```

Business Un altro esempio è quello che all'utente venga applicato il ruolo di Rappresentante account per l'entità organizzativa acquirente della risorsa in questione. Nuovamente si utilizza un gruppo di relazione composto da una catena di relazioni con lunghezza pari a due. La prima parte della catena individuerà tutte le entità organizzative per le quali l'utente dispone del ruolo di rappresentante per l'account. Pertanto, per questa serie di entità organizzative, il gestore delle politiche di controllo accessi verifica se almeno una di esse soddisfa la relazione BuyingOrganizationalEntity con la risorsa. In altri termini, restituisce il valore true se un'entità organizzativa risulta quella acquirente della risorsa.

La seguente sezione XML viene estratta dal file `defaultAccessControlPolicies.xml` ed illustra come definire questo tipo di gruppo di relazione:

```
<RelationGroup Name="AccountRep->BuyerOrganizationalEntity"
      OwnerID="RootOrganization">
   <RelationCondition><![CDATA[
      <profile>
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="ROLE" value="Account Representative"/>
            <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
         </openCondition>
      </profile>
   ]]></RelationCondition>
</RelationGroup>
```

Gruppi di relazione composti da più catene di relazioni: È possibile comporre un gruppo di relazione in modo che contenga più catene di relazioni. In tal caso, è necessario stabilire se l'utente deve soddisfare tutte le catene di relazioni (condizione di operatore logico *AND*) oppure se l'utente deve soddisfare almeno una delle catene di relazioni (condizione di operatore logico *OR*).

Business Per illustrare questo tipo di relazione, la seguente sezione XML viene utilizzata come esempio concreto che l'utente deve essere il creatore della risorsa e che deve anche appartenere all'entità BuyingOrganizationalEntity specificata nella risorsa. La prima catena, che stabilisce che l'utente è il creatore della risorsa, deve avere una lunghezza pari ad uno. La seconda catena, che stabilisce che l'utente deve appartenere all'entità BuyingOrganizationalEntity specificata nella risorsa, deve avere una lunghezza pari a due.

```
<RelationGroup Name="Creator_And_MemberOf->BuyerOrganizationalEntity"
      OwnerID="RootOrganization">
 <RelationCondition><![CDATA[
  <profile>
   <andListCondition>
    <openCondition name="RELATIONSHIP_CHAIN">
     <parameter name="RELATIONSHIP" value="creator" />
    </openCondition>
    <openCondition name="RELATIONSHIP_CHAIN">
     <parameter name="HIERARCHY" value="child"/>
     <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
    </openCondition>
   </andListCondition>
  </profile>
 ]]></RelationCondition>
</RelationGroup>
```

Se si richiede che l'utente non sia subordinato alla condizione dell'operatore logico *AND*, è necessario modificare la tag `<andListCondition>` in `<orListCondition>`.

Professional Business Per illustrare un gruppo di relazione che può essere utilizzato in WebSphere Commerce Professional Edition (nonché in WebSphere Commerce Business Edition), è possibile considerare un gruppo di relazione utilizzato per rafforzare la condizione che l'utente debba essere il creatore o la persona che inoltra la risorsa. Questa condizione è illustrata nella seguente sezione XML.

```
<RelationGroup Name="Creator_Or_Submitter"
     OwnerID="RootOrganization">
   <RelationCondition><![CDATA [
   <profile>
      <orListCondition>
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="RELATIONSHIP"value="creator"/>
         </openCondition>
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="RELATIONSHIP"value="submitter"/>
         </openCondition>
      </orListCondition>
   </profile>
   ]]></RelationCondition>
</RelationGroup>
```

Tipi di controllo accessi

Esistono due tipi di controllo accessi, entrambi basati sulle politiche: controllo accessi a livello di comando e controllo accessi a livello di risorsa.

Il controllo accessi a livello di comando (conosciuto anche come "basato sul ruolo") utilizza un tipo di politica esteso. È possibile specificare che tutti gli utenti con un ruolo particolare possono eseguire determinati tipi di comando. Ad esempio, è possibile specificare che gli utenti che ricoprono il ruolo di Rappresentante per l'account possono eseguire qualsiasi comando all'interno del gruppo di risorse AccountRepresentativesCmdResourceGroup. Oppure come riportato nel seguente diagramma, un altro esempio di politica è quando si specifica che tutti i responsabili di negozio possono eseguire una determinata azione nel gruppo ExecuteCommandAction su qualsiasi risorsa specificata da StoreAdminCmdResourceGrp.

Nota: Le informazioni XML per la colonna Conditions della tabella MBRGRPCOND vengono generate quando si utilizza la Console di gestione per impostare i gruppi di accessi. Per ulteriori informazioni sull'utilizzo della Console di gestione per impostare i gruppi di accessi, fare riferimento alla guida in linea di WebSphere Commerce.

ACPOLICY

PolicyName	Member_Id	MbrGrp_Id	AcActGrp_id	AcResGrp_Id	AcRelGrp_Id
StoreAdministrators ExecuteStoreAdmin CmdResourceGroup	-2001	-8	10052	10018	null

MBRGRP

MbrGrp_Id	MbrGrpName
-8	StoreAdministrators

MBRGRPCOND

MbrGrp_Id	Conditions
-8	<profile> <simpleCondition> <variable name="role"/> <operator name="="/> <value data="Store Administrator"/> </simpleCondition> </profile>

ACACTGRP

AcActGrp_Id	GroupName
10052	ExecuteCommandActionGroup

ACRESGRP

AcResGrp_Id	GrpName
10018	StoreAdminCmdResourceGroup

Figura 22.

Una politica di controllo accessi a livello di comando presenta sempre `ExecuteCommandActionGroup` come gruppo di azioni per i comandi di controller. Per le visualizzazioni, il gruppo di risorse è sempre `ViewCommandResourceGroup`.

Tutti i comandi del controller devono essere protetti dal controllo accessi a livello di comando. Inoltre, qualsiasi visualizzazione che è possibile chiamare direttamente o lanciare mediante un reindirizzamento da un altro comando (invece che mediante un inoltro alla visualizzazione) deve essere protetta dal controllo accessi a livello di comando.

Il controllo accessi a livello di comando non considera il tipo di risorsa sulla quale agisce il comando. Esso determina soltanto se l'utente può eseguire il comando specifico. In caso affermativo, è possibile applicare una politica del controllo accessi a livello di risorsa sequenziale per determinare se l'utente può accedere alla risorsa in questione.

Si consideri il caso in cui un responsabile del negozio prova a eseguire delle attività di gestione. Il primo livello del controllo accessi è determinare se l'utente può eseguire questo particolare comando di gestione del negozio. Dopo aver stabilito che l'utente è autorizzato a eseguire questo comando, in quanto i responsabili del negozio possono eseguire comandi nella politica controllo accessi in storeAdminCmds, è possibile richiamare una politica di controllo accessi a livello di risorsa. Con questa politica si dichiara che i responsabili del negozio sono autorizzati soltanto a gestire attività amministrative per negozi che appartengono all'organizzazione per la quale l'utente è il responsabile del negozio.

In sintesi, nel controllo accessi a livello di comando la "risorsa" è il comando stesso, mentre l'"azione" consiste l'esecuzione del comando, ossia creare un'istanza dell'oggetto comando. La verifica del controllo accessi consente di stabilire se l'utente è autorizzato a eseguire il comando. Al contrario, nel controllo di accesso a livello di risorsa, la "risorsa" è una qualunque risorsa che può essere protetta, alla quale accede il comando o il bean, mentre l'"azione" è rappresentata dal comando stesso.

Interazioni del controllo accessi

In questa sezione viene presentato il diagramma di interazione che illustra l'applicazione del controllo accessi all'interno della struttura delle politiche del controllo accessi di WebSphere Commerce.

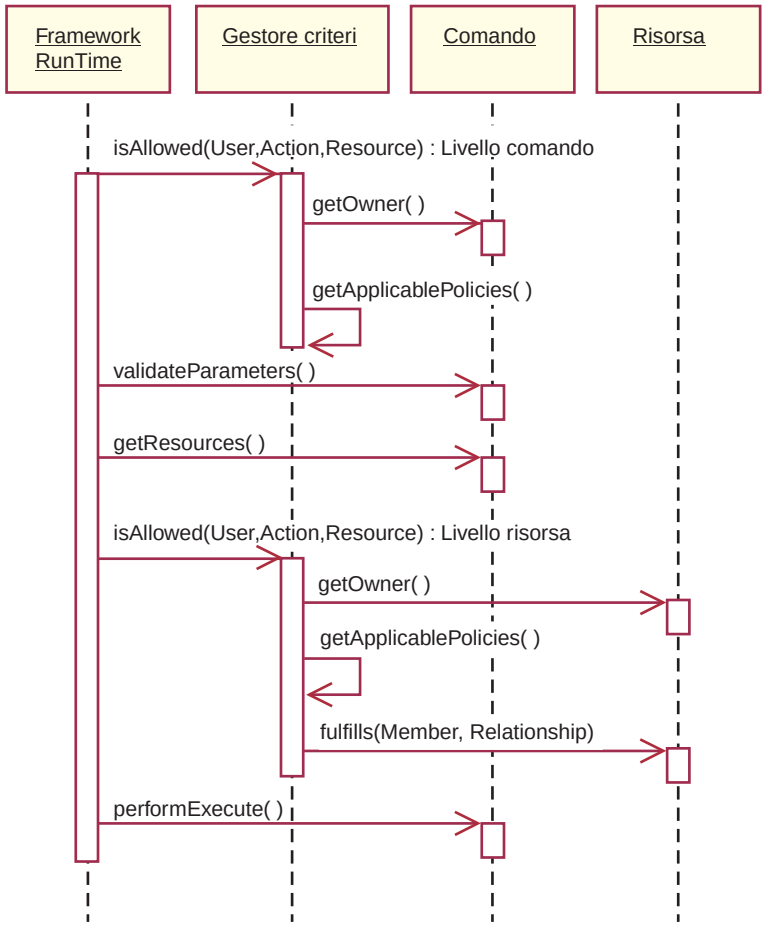


Figura 23.

Il precedente diagramma mostra le azioni eseguite dal *responsabile delle politiche* di controllo accessi. Il gestore delle politiche di controllo accessi è il componente del controllo accessi che determina se l'utente è autorizzato ad eseguire l'azione specificata su una particolare risorsa. Viene infatti effettuata una ricerca tra le politiche del proprietario delle risorse e delle relative organizzazioni. Se almeno una politica garantisce l'accesso, allora l'autorizzazione viene concessa.

Nel seguente elenco vengono descritte le azioni illustrate nel precedente diagramma di interazione. Sono ordinate dall'alto del diagramma verso il basso.

1. `isAllowed()`
   I componenti di runtime determinano se l'utente dispone dell'accesso a livello di comando per il comando di controller oppure per quello di visualizzazione.
2. `getOwner()`
   Il gestore delle politiche di controllo accessi determina il proprietario della risorsa a livello di comando. L'implementazione predefinita restituisce l'ID membro (`memberId`) del proprietario del negozio (`storeId`) nel contesto di comando. Nel caso in cui nel contesto di comando non sia presente alcun ID di negozio, viene restituita l'organizzazione root (`-2001`).
3. `getApplicablePolicies()`
   Il gestore delle politiche di controllo accessi rileva ed elabora le politiche applicabili, in base alla risorsa, all'azione e all'utente specificato.
4. `validateParameters()`
   Parametro iniziale di controllo e risoluzione.
5. `getResources()`
   Restituisce un vettore di accesso che è un vettore delle coppie risorsa-azione.

   Nel caso in cui non venga restituito nulla, la verifica del controllo accessi a livello di risorsa non viene eseguita. Se vi sono risorse da proteggere, viene restituito un vettore di accesso (costituito da coppie risorsa-azione).

   Ciascuna *risorsa* è un'istanza di un oggetto che è possibile proteggere (un oggetto che implementa l'interfaccia `com.ibm.commerce.security.Protectable`). In molti casi, la risorsa è un bean di accesso.

   Anche se il bean di accesso non implementa l'interfaccia `com.ibm.commerce.security.Protectable`, la verifica del controllo accessi può aver luogo fintantoché il corrispondente bean enterprise è protetto, sulla base delle informazioni incluse in "Implementazione del controllo accessi nei bean enterprise" a pagina 110.

   L'*azione* è una stringa che rappresenta l'operazione da eseguire sulla risorsa. Nella maggior parte dei casi, l'azione è il nome interfaccia del comando.
6. `isAllowed()`
   Il componente di runtime determina se l'utente dispone dell'accesso a livello di risorsa per tutte le coppie risorsa-azione specificate da `getResources()`.
7. `getOwner()`
   La risorsa restituisce il `memberId` del proprietario. Ciò determina le politiche da applicare. Vengono applicate soltanto le politiche del proprietario delle risorse e delle precedenti organizzazioni.

8. `getApplicablePolicies()`
   Il gestore delle politiche di controllo accessi ricerca le politiche applicabili e quindi le applica. Se viene rilevata almeno una politica per coppia risorsa-azione che concede l'autorizzazione all'utente ad accedere alla risorsa, allora l'accesso è garantito, altrimenti viene negato.
9. `fulfills()`
   Se una politica applicabile dispone di un gruppo di relazioni specificato, viene effettuato un controllo per accertarsi che il membro soddisfi la relazione oppure le relazioni specificate rispetto alla risorsa.
10. `performExecute()`
    La logica aziendale del comando.

Interfaccia protetta

Un fattore chiave per la protezione di una risorsa mediante le politiche del controllo accessi di WebSphere Commerce, è rappresentato dal fatto che le risorse devono implementare l'interfaccia `com.ibm.commerce.security.Protectable`. Tale interfaccia viene comunemente utilizzata con i bean enterprise e con i bean di dati, ma solo quei particolari bean che richiedono protezione devono implementare l'interfaccia.

Con l'interfaccia Protectable, una risorsa deve fornire due metodi chiave: `getOwner()` e `fulfills(Long member, String relationship)`.

Le organizzazioni o le entità organizzative possiedono le politiche di controllo accessi. Il metodo `getOwner` restituisce il memberId del proprietario della risorsa che può essere protetta. Una volta che il gestore delle politiche per il controllo accessi ha determinato il proprietario delle risorse, richiama il memberId di ogni elemento precedente per il proprietario nella gerarchia dei membri. Quindi vengono applicate tutte le politiche di controllo accessi che appartengono al proprietario dalla richiesta getOwner originale così come le politiche che appartengono agli elementi predecessori del proprietario.

Vengono quindi applicate le politiche di controllo accessi che si applicano a un proprietario particolare o a un predecessore di tale proprietario nella gerarchia dei membri.

Il metodo `fulfills` restituisce soltanto il valore true se il particolare membro soddisfa la relazione richiesta in riferimento alla risorsa. Di solito il membro è un utente singolo; tuttavia, potrebbe anche essere un'organizzazione. Il membro è un'organizzazione se si utilizza un gruppo di relazione nella politica di controllo accessi.

Interfaccia Groupable

L'applicazione di una politica di controllo accessi è specifica di un gruppo di risorse. I raggruppamenti di risorse possono essere effettuati in base ad attributi come il nome della classe, lo stato di un ordine o il valore storeId.

Se una risorsa viene raggruppata in base a un attributo diverso dal nome di classe allo scopo di applicare le politiche di controllo accessi, essa deve implementare l'interfaccia `com.ibm.commerce.grouping.Groupable`.

La seguente sezione di codice rappresenta l'interfaccia Groupable:

```
Groupable interface {
 Object getGroupingAttributeValue (String attributeName, GroupContext context)
}
```

Ad esempio, per implementare una politica che si applica soltanto agli ordini che si trovano in sospeso (`stato = P, pending, ovvero in sospeso`)), l'interfaccia remota del bean entità Order implementa l'interfaccia Groupable e il valore di attributeName viene impostato su `"status"`.

L'utilizzo dell'interfaccia Groupable (raggruppamento) non è molto frequente.

Ricerca di ulteriori informazioni sul controllo accessi

Per ricevere ulteriori informazioni sul modello di controllo accessi di WebSphere Commerce, consultare il manuale *WebSphere Commerce Access Control Guide*. Questa Guida fornisce una panoramica sul controllo degli accessi e descrive come utilizzare la Console di gestione per creare o modificare politiche, gruppi di azioni e gruppi di risorse.

Implementazione del controllo accessi

Questa sezione descrive come implementare il controllo accessi nel codice personalizzato.

Identificazione delle risorse che possono essere protette

In generale, i bean enterprise e i bean di dati sono risorse che si desidera proteggere. Tuttavia, non tutti i bean enterprise e i bean di dati devono essere protetti. All'interno dell'applicazione esistente di WebSphere Commerce, le risorse che richiedono protezione implementano già l'interfaccia di protezione. Quando si creano nuovi bean enterprise e nuovi bean di dati, è necessario stabilire cosa proteggere. Le risorse che vanno protette dipendono dall'applicazione utilizzata.

Se un comando restituisce un bean enterprise nel metodo `getResources`, allora il bean enterprise deve essere protetto in quanto il gestore delle politiche di controllo accessi richiama il metodo getOwner sul bean enterprise. Inoltre, verrà richiamato anche il metodo `fulfills` se è stata specificata una relazione nella corrispondente politica del controllo accessi a livello di risorsa.

Se si implementa l'interfaccia di protezione (e quindi si pongono sotto protezione le risorse) per tutti i bean enterprise e i bean di dati, allora potrebbe essere necessario applicare più politiche per l'applicazione. Con

l'aumentare del numero di politiche le prestazioni possono peggiorare e la gestione delle politiche può divenire più difficile.

Viene fatta una distinzione teorica tra risorse primarie e risorse dipendenti. Una *risorsa primaria* può esistere anche da sola. Una *risorsa dipendente* invece esiste soltanto se esiste la relativa risorsa primaria. Ad esempio, nel codice dell'applicazione di WebSphere Commerce, è possibile proteggere il bean entità Order ma non quello dell'entità OrderItem. Il motivo è che l'esistenza di un OrderItem dipende da un Order: Order è la risorsa primaria mentre OrderItem è una risorsa dipendente. Se un utente può accedere a un ordine, allora può accedere anche agli elementi dell'ordine.

Allo stesso modo, è possibile proteggere il bean entità User ma non il bean entità Address. In questo caso, l'esistenza dell'indirizzo dipende dall'utente, pertanto qualsiasi accesso all'utente vale anche come accesso all'indirizzo.

Le risorse primarie vanno protette, ma spesso non è necessario proteggere le risorse dipendenti. Se un utente può accedere a una risorsa primaria, allora ha un senso che lo stesso utente possa accedere alle relative risorse dipendenti.

Implementazione del controllo accessi nei bean enterprise

Se si creano nuovi bean enterprise che richiedono protezione mediante le politiche di controllo accessi, è necessario effettuare le seguenti operazioni:

1. Creare un nuovo bean enterprise, assicurando che si estenda da `com.ibm.commerce.base.objects.ECEntityBean`.
2. Assicurarsi che l'interfaccia remota del bean estenda l'interfaccia `com.ibm.commerce.security.Protectable`.
3. Se le risorse con le quali interagisce il bean sono raggruppate in base a un attributo diverso dal nome della classe Java della risorsa, l'interfaccia remota del bean deve estendere anche l'interfaccia `com.ibm.commerce.grouping.Groupable`.
4. La classe bean enterprise contiene le implementazioni predefinite per i seguenti metodi:
   - `getOwner`
   - `fulfills`
   - `getGroupingAttributeValue`

   Sovrascrivere qualsiasi metodo che non è necessario. È necessario almeno sovrascrivere il metodo `getOwner`.
   Le implementazioni predefinite di questi metodi sono riportate nelle seguenti sezioni di codice,

```
**********************************************************************
public Long getOwner() throws Exception, java.rmi.RemoteException
{
```

```
   return null;
}
*****************************************************************************
*****************************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException {
{
      return false;
}
*****************************************************************************
*****************************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception, java.rmi.RemoteException
{
      return null;
}
*****************************************************************************
```

Di seguito vengono riportate delle implementazioni di esempio di questi metodi basati su OrderBean:

```
*****************************************************************************
public Long getOwner() throws Exception, java.rmi.RemoteException
{
   com.ibm.commerce.common.objects.StoreEntityAccessBean storeEntAB = new
   com.ibm.commerce.common.objects.StoreEntityAccessBean();
   storeEntAB.setInitKey_storeEntityId(getStoreEntityId().toString());
   return storeEntAB.getMemberIdInEJBType();
}
*****************************************************************************
*****************************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException {
   {
      if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }
      else if (relationship.equalsIgnoreCase (
         com.ibm.commerce.base.helpers.EJBConstants.
         SAME_ORGANIZATIONAL_ENTITY_AS_CREATOR_RELATION)) {
            com.ibm.commerce.user.objects.UserAccessBean creator = new
               com.ibm.commerce.user.objects.UserAccessBean();
            creator.setInitKey_MemberId(getMemberId().toString());
            com.ibm.commerce.user.objects.UserAccessBean ab = new
               com.ibm.commerce.user.objects.UserAccessBean();
            ab.setInitKey_MemberId(member.toString());
            if (ab.getParentMemberId().equals(creator.getParentMemberId()))
               return true;
          }
      return false;
}
*****************************************************************************
```

```
*****************************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception
   {
      if (attributeName.equalsIgnoreCase("Status"))
         return getStatus();
   return null;
   }
*****************************************************************************
```

5. Creare (o ricreare) il codice generato e il bean di accesso del bean enterprise.

Implementazione del controllo accessi nei bean di dati

Se necessario, un bean di dati può essere protetto, direttamente o indirettamente, dalle politiche di controllo accessi. Se un bean di dati viene protetto direttamente, allora significa che esiste una politica di controllo accessi che si applica a quel particolare bean di dati. Se invece un bean di dati viene protetto indirettamente, allora la protezione viene delegata a un altro bean di dati per il quale esiste una politica di controllo accessi.

Se si crea un bean di dati che viene protetto direttamente da una politica di controllo accessi, allora è necessario che il bean di dati possa:

1. Implementare l'interfaccia `com.ibm.commerce.security.Protectable`. Pertanto, il bean deve fornire un'implementazione dei metodi `getOwner()` e `fulfills(Long member, String relationship)`. Tali metodi devono essere implementati sull'interfaccia remota del bean.

   Quando un bean di dati implementa l'interfaccia Protectable, il gestore dei bean di dati richiama il metodo `isAllowed` per determinare se l'utente dispone dei privilegi di controllo accessi appropriati, in accordo alla politica di controllo accessi corrente. Il metodo `isAllowed` viene descritto dalla seguente sezione di codice:

   ```
   IsAllowed(Context, "Display", protectable_databean);
   ```

2. Se le risorse con le quali interagisce il bean vengono raggruppate in base a un attributo diverso dal nome della classe Java della risorsa, il bean deve implementare l'interfaccia `com.ibm.commerce.grouping.Groupable`.
3. Implementare l'interfaccia `com.ibm.commerce.security.Delegator`. Questa interfaccia è descritta nella seguente sezione di codice:

   ```
   Interface Delegator {
    Protectable getDelegate();
   }
   ```

   Nota: Perché possa essere protetto direttamente, il metodo `getDelegate` deve restituire il bean di dati, ossia il bean di dati si autodelega per il controllo accessi.

La differenza tra la protezione diretta e quella indiretta dei bean di dati è molto simile alla differenza che esiste tra risorse primarie e risorse dipendenti. Se il bean di dati può esistere da solo, allora può essere protetto direttamente. Se invece l'esistenza del bean di dati dipende dall'esistenza di un altro bean di dati, allora è necessario che venga delegato l'altro bean di dati per la protezione.

Un esempio di bean di dati che può essere protetto direttamente è il bean di dati Order. Un esempio di bean di dati che invece deve essere protetto indirettamente è il bean di dati OrderItem.

Se viene creato un nuovo bean di dati che deve essere protetto indirettamente da una politica di controllo accessi, è necessario che il bean di dati possa:

1. Implementare l'interfaccia `com.ibm.commerce.security.Delegator`. Questa interfaccia è descritta nella seguente sezione di codice:

```
Interface Delegator {
 Protectable getDelegate();
}
```

   Nota: Il bean di dati restituito da `getDelegate` deve implementare l'interfaccia Protectable.

Se un bean di dati non implementa l'interfaccia Delegator, allora viene applicato senza alcuna protezione da parte della politica di controllo accessi.

Implementazione del controllo accessi nei comandi del controller

Quando si crea un comando del controller, la classe di implementazione del il nuovo comando deve estendere la classe `com.ibm.commerce.commands.ControllerCommandImpl` e la relativa interfaccia deve estendere l'interfaccia `com.ibm.commerce.command.ControllerCommand`.

Per le politiche a livello del comando, il nome dell'interfaccia del comando viene specificata come risorsa. Affinché una risorsa sia protetta, deve implementare l'interfaccia Protectable. A seconda del modello di programmazione di WebSphere Commerce, tale implementazione viene eseguita facendo in modo che l'interfaccia del comando sia un'estensione dell'interfaccia `com.ibm.commerce.command.ControllerCommand` e l'implementazione del comando un'estensione di `com.ibm.commerce.commands.ControllerCommandImpl`. L'interfaccia `ControllerCommand` estende l'interfacciacom.ibm.commerce.command.AccCommand, che a sua volta estende `Protectable`. L'interfaccia `AccCommand` è l'interfaccia minima che un comando deve implementare per essere protetto dal controllo accessi a livello del comando.

Se il comando accede alle risorse che devono essere protette, creare una variabile dell'istanza privata del tipo `AccessVector` per contenere le risorse.

Successivamente, sovrascrivere il metodo getResources in quanto l'implementazione predefinita di questo metodo prevede la restituzione di un valore null e quindi il controllo delle risorse non ha luogo.

Nel nuovo metodo getResources, è necessario restituire un vettore di risorse delle coppie risorsa-azione sul quale il comando può essere eseguito. Quando non specifica in maniera esplicita un'azione, per impostazione predefinita l'azione viene impostata sul nome interfaccia del comando in esecuzione.

Inoltre, è necessario che il metodo stabilisca se deve creare un'istanza della risorsa oppure se può utilizzare la variabile dell'istanza esistente che contiene il riferimento alla risorsa. Per migliorare il livello delle prestazioni del sistema, è consigliabile controllare se l'oggetto risorse già esiste. Se richiesto, è possibile utilizzare lo stesso metodo getResources, nel metodo performExecute del nuovo comando di controller.

Di seguito viene riportato un esempio del metodo getResources:

```
private AccessVector resources = null;

public AccessVector getResources() throws ECException {

   if (resources == null) {
      OrderAccessBean orderAB = new OrderAccessBean();
      orderAB.setInitKey_orderId(getOrderId().toString());
      resouces = new AccessVector(orderAB);
      }
   return resources;
}
```

Come esempio, si consideri il comando OrderItemUpdate. Il metodo getResources di questo comando restituisce gli oggetti proteggibili Order e User. Poiché l'azione non è specificata, si assume come valore predefinito quello del comando OrderItemUpdate.

Il metodo getResources può restituire più risorse. Quando si verifica ciò, se l'azione deve essere effettuata, deve essere rilevata una politica che conceda all'utente l'accesso a tutte le risorse specificate. Se, ad esempio, l'utente ha accesso solo a due delle tre risorse specificate, l'azione potrebbe non aver luogo (sono necessarie tutte e tre le risorse).

Se è necessario effettuare un ulteriore controllo dei parametri oppure risolvere i parametri nel comando di controller, è possibile utilizzare il metodo validateParameters(). Questa operazione è facoltativa.

Ulteriore controllo del livello delle risorse

Non sempre è possibile determinare tutte le risorse che devono essere protette nel momento in cui viene richiamato il metodo getResources del comando di controller.

Se necessario, un comando di attività può implementare il metodo getResources in modo che venga restituito un elenco di risorse sulle quali il comando può operare.

Un altro modo per effettuare il controllo del livello delle risorse è richiamare direttamente il gestore delle politiche di controllo accessi mediante il metodo checkIsAllowed(Object resource, String action). Questo metodo è disponibile per qualsiasi classe che rappresenti un'estensione della classe com.ibm.commerce.command.AbstractECTargetableCommand. Ad esempio, le classi di seguito riportate sono un'estensione della classe AbstractECTargetableCommand:

- com.ibm.commerce.command.ControllerCommandImpl
- com.ibm.commerce.command.DataBeanCommandImpl

Il metodo checkIsAllowed è disponibile anche per le classi che estendono la classe com.ibm.commerce.command.AbstractECCommand. Ad esempio,, la seguente classe viene estesa dalla classe AbstractECCommand:

- com.ibm.commerce.command.TaskCommandImpl

Di seguito è riportata la firma del metodo checkIsAllowed.

```
void checkIsAllowed(Object resource, String action)
   throws ECException
```

Questo metodo restituisce un'eccezione ECApplicationException se l'utente corrente non dispone dell'autorizzazione all'esecuzione dell'azione specificata sulla risorsa specificata. Se l'accesso è concesso, il metodo non restituisce eccezioni.

Controllo accessi per i comandi "create"

Dal momento che il metodo getResources viene richiamato prima del metodo performExecute in un comando, è necessario adottare un approccio differente per il controllo accessi alle risorse che non sono state ancora create. Ad esempio, se si dispone di WidgetAddCmd, il metodo getResources non può restituire la risorsa che sta per essere creata. In questo caso, il metodo getResources deve restituire il creatore della risorsa stessa. Ad esempio, un comando viene creato da un comando factory, un ordine viene creato all'interno di un negozio e un utente viene creato all'interno di un'organizzazione.

Implementazioni predefinite per il controllo accessi a livello di comando

Per quanto concerne il controllo accessi a livello di comando, l'implementazione predefinita del metodo getOwner() restituisce il memberId del proprietario del negozio nel caso in cui sia stato specificato lo storeId. Se lo storeId non è stato specificato, viene restituito il memberId dell'organizzazione root (memberId = -2001).

L'implementazione predefinita del metodo `getResources()` restituisce il valore `null`.

L'implementazione predefinita di `validateParameters()` non restituisce nulla.

Implementazione delle politiche di controllo accessi nelle visualizzazioni

Il controllo accessi a livello di risorsa per le visualizzazioni viene eseguito dal gestore dei bean di dati. Il gestore dei bean di dati viene richiamato nei seguenti casi:

1. Quando la maschera JSP include la tag `<useBean>` e il bean di dati non si trova nell'elenco attributi.
2. Quando la maschera JSP include il seguente metodo activate:

```
DataBeanManager.activate(xyzDatabean, request);
```

Nota: Qualsiasi bean di dati che deve essere protetto (direttamente o indirettamente) deve implementare l'interfaccia Delegator. Qualsiasi bean di dati che deve essere protetto direttamente attiverà l'autodelega e, quindi, deve anche implementare l'interfaccia Protectable. I bean di dati che sono protetti indirettamente devono delegare ad un bean di dati che implementa l'interfaccia Protectable.

Sebbene sconsigliata, nei seguenti casi si verifica una deviazione del controllo accessi:

1. Se la maschera JSP richiama direttamente i bean di accesso piuttosto che utilizzare i bean di dati.
2. Se la maschera JSP richiama direttamente il metodo populate() del bean di dati.

Se i risultati di un comando di controller devono essere inoltrati a una visualizzazione (utilizzando ForwardViewCommand), il controllo accessi a livello di comando non viene eseguito sulle visualizzazioni. Inoltre, se il comando di controller inserisce i bean dei dati popolati (che vengono utilizzati nella visualizzazione) nell'elenco attributi della proprietà della risposta e poi li inoltra alla visualizzazione, la maschera JSP può accedere ai dati senza dover ricorrere al gestore dei bean di dati. In questo caso, non è necessario utilizzare tag `<useBean>` nella maschera JSP. In questo modo, è possibile accrescere l'efficacia di una maschera JSP, in quanto può ignorare tutte le verifiche ridondanti dei controlli accessi a livello di risorsa sulle risorse (bean dei dati) alle quali l'utente può già accedere mediante il comando di controller.

Capitolo 5. Messaggi e gestione degli errori

Gestione degli errori di comando

WebSphere Commerce utilizza una struttura per la gestione degli errori di comando ben definita e semplice da utilizzare in un codice personalizzato. La progettazione prevede che la struttura gestisca gli errori in modo da supportare negozi multiculturali. Le sezioni successive descrivono i tipi di eccezione che un comando può inviare, il modo in cui le eccezioni vengono gestite, il modo in cui il testo del messaggio viene memorizzato e utilizzato, il modo il cui le eccezioni vengono registrate e le modalità di utilizzo della struttura fornita con i comandi.

Tipi di eccezione

Un comando può inviare una delle seguenti eccezioni:

ECApplicationException
Questa eccezione viene inviata se l'errore è relativo all'utente. Ad esempio, quando un utente immette un parametro non valido viene inviata l'eccezione ECApplicationException. Quando questa eccezione viene inviata, il controller Web non tenta di eseguire nuovamente il comando anche se il comando è specificato come comando che può essere eseguito più volte.

ECSystemException
Questa eccezione viene inviata se viene rilevata un'eccezione di runtime o un errore di configurazione di WebSphere Commerce. Esempi di questo tipo di eccezione includono eccezioni Pointer null ed eccezioni di rollback della transazione. Quando viene inviato questo tipo di eccezione, il controller Web tenta di nuovo di eseguire il comando se è un comando che prevede più esecuzioni e se l'eccezione è stata causata da una condizione di stallo del database o dal rollback del database.

Entrambe le eccezioni sopra elencate sono classi che si estendono dalla classe ECException disponibile nel pacchetto com.ibm.commerce.exception.

Per poter inviare una di queste eccezioni, specificare le seguenti informazioni:

- Nome visualizzazione errore
  Il controller Web cerca questo nome nella tabella VIEWREG.
- Oggetto ECMessage
  Questo valore corrisponde al testo del messaggio contenuto nel file delle proprietà.

- Parametri di errore
  Queste coppie nome-valore sono utilizzate per sostituire le informazioni in un messaggio di errore. Ad esempio, un messaggio può contenere un parametro che consente di trattenere il nome del metodo che ha inviato l'eccezione. Questo parametro viene impostato quando viene inviata l'eccezione; quindi, al momento della registrazione del messaggio di errore, il file di log riporta il nome effettivo del metodo.
- Dati di errore
  Si tratta di attributi facoltativi che possono essere resi disponibili nella maschera JSP tramite un bean di dati di errore.

La gestione delle eccezioni è strettamente collegata al sistema di registrazione. Quando un'eccezione viene inviata, viene automaticamente registrata.

File delle proprietà del messaggio di errore

Per semplificare la gestione dei messaggi di errore e per supportare i negozi che utilizzano più lingue, il testo dei messaggi di errore viene memorizzato nei file delle proprietà (properties). Il testo dei messaggi di WebSphere Commerce viene memorizzato nel file `ecServerMessages_XX_XX.properties` dove *_XX_XX* rappresenta la locale (ad esempio, `_en_US`).

Il contesto di comando restituisce un identificativo della lingua utilizzata dal client. Quando è necessario utilizzare un messaggio, il controller Web sceglie il file delle proprietà da utilizzare in base all'identificativo della lingua.

Esistono due tipi di messaggio definiti nel file `ecServerMessages`*XX_XX*`.properties`: i messaggi utente e i messaggi del sistema. I messaggi utente vengono visualizzati ai clienti nei relativi browser. Sia i messaggi utente che i messaggi di sistema vengono automaticamente catturati nel file di log dei messaggi.

Quando viene inviato un errore, uno dei parametri obbligatori è un oggetto messaggio. Per ECSystemExceptions, l'oggetto messaggio deve contenere due chiavi, una per il messaggio del sistema e una per il messaggio utente. Per ECApplicationExceptions, l'oggetto messaggio contiene la chiave per il messaggio utente (i messaggi di sistema non sono utilizzati).

Tutti i messaggi del sistema sono predefiniti. Non è possibile creare propri messaggi di sistema. Quindi, quando il codice personalizzato restituisce ECSystemException, deve specificare una chiave di messaggi per uno dei messaggi di sistema predefiniti. È possibile creare messaggi utente personalizzati. I messaggi utente nuovi devono essere memorizzati in un file proprietà separato.

Flusso di gestione dell'eccezione

Il seguente diagramma mostra il flusso di informazioni determinato dal rilevamento di un'eccezione. Alla fine del diagramma è riportata la descrizione di ciascuna fase.

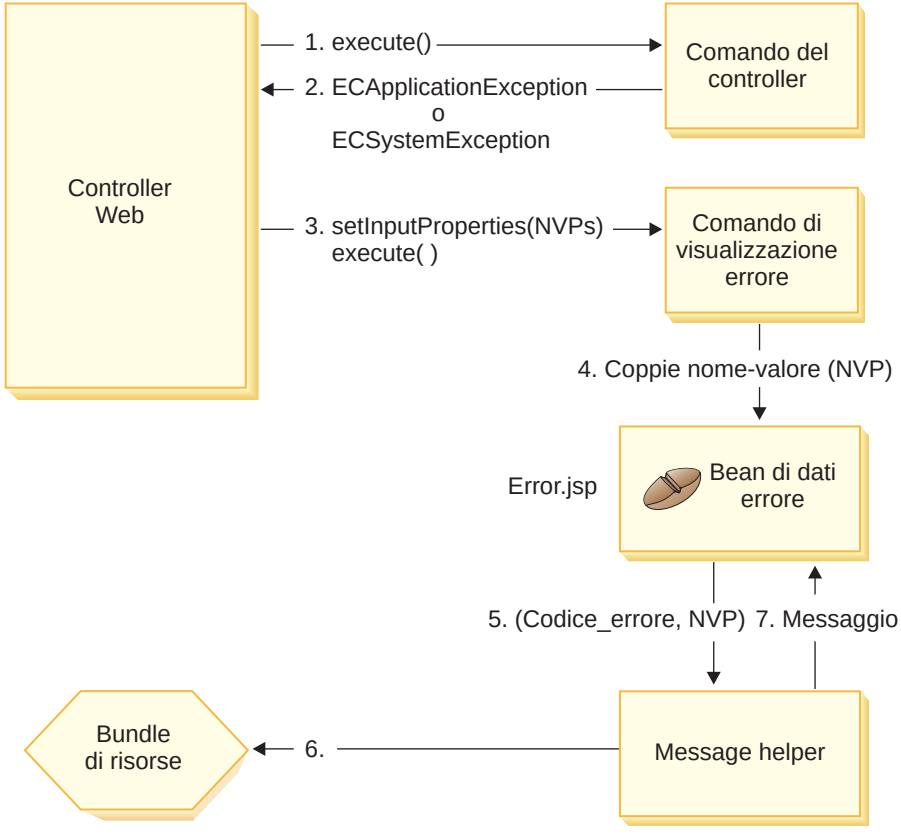


Figura 24.

1. Il controller Web richiama un comando di controller.
2. Il comando invia un'eccezione rilevata dal controller Web. Può essere ECApplicationException o ECSystemException. L'oggetto eccezione contiene le seguenti informazioni:
   - Nome visualizzazione errore
   - Oggetto ECMessage
   - Parametri dell'errore
   - Dati dell'errore (facoltativo)
3. Il controller Web definisce il nome di visualizzazione dell'errore dalla tabella VIEWREG e richiama il comando per la visualizzazione degli errori

specificati. Quando richiama il comando, il controller Web crea una serie di proprietà dall'oggetto ECException e le imposta nel comando di visualizzazione utilizzando il metodo setInputProperties del comando di visualizzazione.

4. Il comando di visualizzazione richiama una maschera JSP dell'errore (in questo caso, Error.jsp); quindi, viene inviata la coppia nome-valore alla maschera JSP.
5. ErrorDataBean invia i parametri di errore all'oggetto helper del messaggio.
6. L'oggetto helper del messaggio riceve il messaggio richiesto (utilizzando l'oggetto messaggio e i parametri di errore) dal file di proprietà appropriato.
7. Il bean di dati di errore restituisce il messaggio alla maschera JSP.

Gestione dell'eccezione nel codice personalizzato

Quando si creano nuovi comandi, è importante includere la gestione delle eccezioni appropriata. È possibile utilizzare la gestione degli errori e la struttura dei messaggi forniti con WebSphere Commerce specificando le informazioni richieste quando si rileva un'eccezione.

Per scrivere una propria logica della gestione delle eccezioni, è necessario completare le seguenti operazioni:

1. Individuare le eccezioni nel proprio comando che richiedono una particolare elaborazione.
2. Creazione di un'eccezione ECApplicationException o ECSystemException, in base al tipo di eccezione rilevata.
3. Se ECApplicationException utilizza un nuovo messaggio, definire il messaggio in un nuovo file delle proprietà.

Rilevamento e creazione delle eccezioni

Per illustrare le prime due fasi, la seguente sezione del codice mostra un esempio di rilevamento di un'eccezione in un comando:

```
try {
// your business logic
}
catch(FinderException e) {
     throw new ECSystemException (ECMessage._ERR_FINDER_EXCEPTION,
          className, methodName, new Object [] {e.toString()}, e);
}
```

Il precedente oggetto _ERR_FINDER_EXCEPTION ECMessage viene definito nel modo seguente:

```
public static final ECMessage _ERR_FINDER_EXCEPTION =
new ECMessage (ECMessageSeverity.ERROR, ECMessageType.SYSTEM,
     ECMessageKey._ERR_FINDER_EXCEPTION);
```

Il testo del messaggio _ERR_FINDER_EXCEPTION viene definito nel file ecServerMessages_*xx_XX*.properties (dove *_xx_XX* rappresenta la locale come *_en_US*), nel modo seguente:

```
_ERR_FINDER_EXCEPTION  =
     The following Finder Exception occurred during processing: "{0}".
```

Quando si rileva un'eccezione del sistema, è possibile utilizzare una serie predefinita di messaggi. La seguente tabella ne riporta la descrizione.

Oggetto messaggio	Descrizione
_ERR_FINDER_EXCEPTION	Ricevuto quando viene restituito un errore da una chiamata del metodo finder EJB.
_ERR_REMOTE_EXCEPTION	Ricevuto quando viene restituito un errore da una chiamata del metodo remoto EJB.
_ERR_CREATE_EXCEPTION	Ricevuto quando si verifica un errore durante la creazione di istanze EJB.
_ERR_NAMING_EXCEPTION	Ricevuto quando viene restituito un errore da un server dei nomi.
_ERR_GENERIC	Ricevuto quando si verifica un errore di sistema imprevisto. Ad esempio, un'eccezione pointer null.

Quando si rileva un'eccezione dell'applicazione, è possibile utilizzare un messaggio esistente specificato nel file ecServerMessages_xx_xx.properties appropriato, oppure creare un nuovo messaggio memorizzato in un nuovo file delle proprietà. Come già precisato, *non è possibile* modificare i file ecServerMessages_xx_XX.properties.

La seguente sezione di codice mostra un esempio di rilevamento di un'eccezione di applicazione in un comando:

```
try {
// your business logic

}
// catch some new type of application exception
catch(//your new exception)
 {
     throw new ECApplicationException (MyMessages._ERR_CUSTOMER_INVALID,
          className, methodName, errorTaskName, someNVPs);
}
```

Il precedente oggetto _ERR_CUSTOMER_INVALID ECMessage viene definito nel modo seguente:

```
public static final ECMessage _ERR_CUSTOMER_INVALID =
     new ECMessage (ECMessageSeverity.ERROR, ECMessageType.USER,
     MyMessagesKey._ERR_CUSTOMER_INVALID, "ecCustomerMessages");
```

Quando si creano nuovi messaggi utente, è necessario assegnare un tipo di USER nel modo seguente:

```
ECMessageType.USER
```

Il testo per il messaggio _ERR_CUSTOMER_INVALID si trova nel file `ecCustomerMessages.properties`. Questo file deve risiedere in una directory che si trova nel classpath. Il testo viene definito nel modo seguente:

```
_ERR_CUSTOMER_INVALID = Invalid ID "{0}"
```

Creazione di messaggi

Se il comando invia una ECApplicationException che utilizza un nuovo messaggio, è necessario creare questo nuovo messaggio. La creazione di un nuovo messaggio prevede le seguenti fasi:

1. Creazione di una nuova classe che contiene le chiavi dei messaggi.
2. Creazione di una nuova classe che contiene gli oggetti ECMessage.
3. Creazione di una raccolta di risorse.
4. Verifica messaggio.

Ulteriori dettagli relativi a ciascuna operazione vengono forniti nelle sezioni successive.

Creazione di una classe per le chiavi di messaggio

La prima fase del processo di creazione di nuovi messaggi utente consiste nella creazione di una classe che contenga le nuove chiavi di messaggio. Una chiave di messaggio è un indicatore univoco utilizzato dal servizio di registrazione per individuare il testo del messaggio corrispondente nel bundle delle risorse. Questa nuova classe deve essere creata nel proprio pacchetto e memorizzata in un progetto separato dai progetti di WebSphere Commerce.

Considerare un esempio, chiamato `MyNewMessages`, in cui si crea una nuova classe, chiamata `MyMessageKeys` che contiene le chiavi di messaggi `_ERR_CUSTOMER` e `_ERR_CUSTOMER_INVALID_ID` e che viene inserita nel pacchetto `com.mycompany.messages`. In questo caso, la definizione della classe è la seguente:

```
public class MyMessageKeys
{
    public static String _ERR_CUSTOMER="_ERR_CUSTOMER";
    public static String _ERR_CUSTOMER_INVALID_ID="_ERR_CUSTOMER_INVALID_ID";
}
```

I contenitori String per le chiavi di messaggio consentono al compilatore di controllare la propria validità.

Creazione di una classe per gli oggetti ECMessage

Nello stesso pacchetto in cui è stata creata la classe per le chiavi di messaggio, creare un'altra classe che contenga gli oggetti di ECMessage. La classe

ECMessage definisce la struttura di un oggetto messaggio. Viene utilizzato per richiamare e mantenere i messaggi di testo sensibili alla locale.

L'oggetto messaggio contiene i seguenti attributi: severità, tipo, chiave, bundle delle risorse e bundle delle risorse associate. Esistono diversi metodi constructor per questa classe. Per informazioni dettagliate, consultare la sezione "Riferimenti" della guida in linea di WebSphere Commerce.

Seguendo l'esempio MyNewMessages, creare una nuova classe chiamata `MyMessages` all'interno del pacchetto `com.mycompany.messages` nel modo seguente:

```
import com.ibm.commerce.ras.*;

public class MyMessages
    {
        static String myResourceBundle = "ecCustomerMessages";

        public static ECMessage _ERR_CUSTOMER = new ECMessage
             (ECMessageSeverity.ERROR,ECMessageType.USER,
                   MyMessageKeys._ERR_CUSTOMER, myResourceBundle);
        public static ECMessage _ERR_CUSTOMER_INVALID_ID = new ECMessage
             (ECMessageSeverity.ERROR, ECMessageType.USER,
                  MyMessageKeys._ERR_CUSTOMER_INVALID_ID,
                  myResourceBundle);

    }
```

Nella sezione di codice precedente, l'istruzione di importazione viene richiesta per la creazione dell'oggetto ECMessage. L'oggetto MyMessage._ERR_CUSTOMER è un messaggio dell'utente con severità `ERROR`. MyMessageKeys._ERR_CUSTOMER viene utilizzata dal servizio di registrazione di WebSphere Commerce per individuare il testo dei messaggi contenuti nel file delle proprietà *ecCustomerMessages*.

Creazione di una raccolta di risorse per i messaggi utente

È necessario creare un nuovo bundle delle risorse in cui vengono memorizzate le chiavi del messaggio con il testo corrispondente. Questo bundle di risorse può essere implementato come oggetto Java oppure come file delle proprietà. È consigliabile utilizzare i file delle proprietà in quanto sono più semplici da tradurre e da gestire. I file delle proprietà sono utilizzati per i messaggi di WebSphere Commerce.

Per continuare l'esempio MyNewMessages, creare un file di testo chiamato `ecCustomerMessages.properties`. Se i messaggi devono essere utilizzati da un singolo servlet di negozio, inserire il file nella seguente directory:

unità`:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_`*Nomeistanza*`.ear\wcstores.war\WEB-INF\classes`

Se i messaggi devono essere utilizzati da servlet di un singolo servlet di strumenti, inserire il file nella seguente directory:

unità:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_*Nomeistanza*.ear\
wctools.war\WEB-INF\classes

Se i messaggi devono essere utilizzati globalmente da tutti i servlet dell'enterprise application, inserire il file nella seguente directory:

unità:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_*Nomeistanza*.ear\
properties

Poiché il file delle proprietà contiene coppie di chiavi di messaggio e il testo del messaggio corrispondente, il file ecCustomerMessages.properties contiene le seguenti righe:

```
_ERR_CUSTOMER_MESSAGE = The customer message "{0}".
_ERR_CUSTOMER_INVALID_ID = Invalid ID "{0}".
```

Test unità di un messaggio

Dopo aver creato le classi contenenti le chiavi dei messaggi, le classi contenenti gli oggetti ECMessage e il bundle di risorse, è necessario verificare i nuovi messaggi.

Per eseguire il test dei nuovi messaggi descritti nelle sezioni precedenti, effettuare le seguenti operazioni:

1. Creare una nuova classe per l'esecuzione del test. In questo caso, creare MyTestingClass.
2. Aggiungere la seguente istruzione di importazione alla classe

   ```
   import com.ibm.commerce.ras.*;
   ```
3. Aggiungere un metodo main() alla classe.
4. Esaminare la seguente sezione di codice. Aggiornarlo in base alle proprie esigenze (ad esempio, accertarsi che il percorso della directory punti a un file di configurazione valido del proprio sistema) e inserirlo nel metodo main()

   ```
   // the fileName String variable should point
   // to a valid WebSphere Commerce configuration file:
     String fileName = "E:\\WebSphere\\CommerceServer\\instances
         \\demo\\xml\\demo.xml";

      LogConfiguration config = LogConfiguration.getUniqueInstance ();
      config.initialize (fileName, "testClone");

      ECMessageLog.out (MyMessage._ERR_CUSTOMER,
           "MyTestingClass", "main", "Hello");
   ```

   Le prime tre righe del codice inizializzano il servizio di registrazione di WebSphere Commerce. L'ultima riga di codice indica a ECMessageLog di stampare il messaggio MyMessage._ERR_CUSTOMER. MyTestingClass e

`main` fanno parte del formato di traccia del log. La stringa `Hello` viene posizionata nel segnaposto {0} del messaggio definito nel file `ecCustomerMessages.properties`.

5. Eseguire la classe. Nel file di log, la traccia del messaggio è simile alla seguente:

```
==============
 TimeStamp:      2000-11-29 16:41:42.5
 Thread ID:     <main>
 Class:          MyTestingClass
 Method:         main
 Severity:       1
 Message Text:   The customer message "Hello".
```

È possibile eseguire un test simile per visualizzare il secondo messaggio.

Traccia del flusso di esecuzione

WebSphere Commerce include la classe ECTrace che viene utilizzata per tenere traccia del flusso di esecuzione dei componenti in esecuzione sulla macchina WebSphere Commerce Server. La classe ECTrace fa parte del pacchetto com.ibm.commerce.ras.

Quando si crea una nuova logica aziendale, è possibile inserire una traccia nel codice per il metodo di debug. Le informazioni della traccia vengono catturate nel log di traccia. È possibile specificare un punto di entrata e di uscita della traccia. Inoltre, è possibile indicare che è possibile mantenere traccia di dati specifici tra questi due punti.

Per utilizzare la funzione di traccia, è necessario attivarla per il componente nel quale si desidera eseguire la traccia. Per attivare la traccia per un determinato componente, è possibile utilizzare la Console di gestione oppure gestore configurazione.

Quando si tiene traccia del codice personalizzato, *è necessario* utilizzare il componente EXTERN. Nel gestore configurazione, tale componente viene denominato *External*.

Per impostare il punto di entrata di una traccia nel proprio codice, utilizzare la seguente sintassi:

```
ECTrace.entry (ECTraceIdentifiers.COMPONENT_EXTERN, NomeClasse, Nomemetodo);
```

dove *NomeClasse* rappresenta la classe che contiene il metodo di cui è stata eseguita la traccia. Poiché questa stringa può essere utilizzata per tracciare l'analisi del file, dovrebbe contenere il nome della classe completo. Se il metodo di cui eseguire la traccia è statico, la dichiarazione di esempio di NomeClasse è

```
String NomeClasse = "com.mycompany.agrouping.MyTracedClass";
```

Se il metodo di cui eseguire la traccia non è statico, la dichiarazione di esempio di NomeClasse è

```
String NomeClasse = this.getClass().getName();
```

Per impostare il punto di traccia per eseguire la traccia dei dati di un metodo, utilizzare la seguente sintassi:

```
ECTrace.trace (ECTraceIdentifiers.COMPONENT_EXTERN, NomeClasse,
     NomeMetodo, Testo);
```

dove *Testo* è il testo da visualizzare nel log di traccia.

Per impostare il punto di uscita di una traccia nel proprio codice, utilizzare la seguente sintassi:

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, NomeClasse,
     NomeMetodo);
```

Se si desidera tenere traccia dell'oggetto restituito dal metodo di cui eseguire la traccia, impostare il punto di uscita come segue:

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, NomeClasse,
     NomeMetodo, Oggettorestituito);
```

dove `Oggettorestituito` rappresenta l'oggetto Java restituito dal metodo.

Ad esempio, occorre eseguire la traccia del metodo performExecute di un nuovo comando di controller denominato MyNewControllerCmd. La seguente sezione di codice mostra come utilizzare i metodi ECTrace all'interno del metodo performExecute.

```
public void performExecute() throws ECException {
     ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN,
          this.getClass().getName(), "performExecute");

     super.performExecute();


//////////////////////////////////////////
//   Some of your business logic          //
//////////////////////////////////////////


     ECTrace.trace(ECTraceIdentifiers.COMPONENT_EXTERN,
          this.getClass().getName(), "performExecute",
          "My code is great!");

//////////////////////////////////////////
//   Some more business logic             //
//////////////////////////////////////////
```

```
        ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN,
             this.getClass().getName(), "performExecute");

    }
```

Quando viene richiamato il precedente metodo performExecute, il file di log di traccia acquisisce le seguenti informazioni:

```
==============
TimeStamp:     2000-12-05 17:32:00.257
Thread ID:    <P=502832:O=0:CT>
Component:     EXTERN
Class:         com.mycompany.agrouping.MyNewControllerCmd
Method:        performExecute
Trace:          ENTRY POINT
==============
TimeStamp:     2000-12-05 17:32:00.257
Thread ID:    <P=502832:O=0:CT>
Component:     EXTERN
Class:         com.mycompany.agrouping.MyNewControllerCmd
Method:        performExecute
Trace:         My code is great!
==============
TimeStamp:     2000-12-05 17:32:00.258
Thread ID:    <P=502832:O=0:CT>
Component:     EXTERN
Class:         com.mycompany.agrouping.MyNewControllerCmd
Method:        performExecute
Trace:         EXIT POINT
```

Si consiglia di utilizzare la funzione di traccia solo per le funzioni principali. La funzione di traccia non è disponibile per varie lingue perché è stata progettata per essere utilizzata dagli sviluppatori del negozio. Al contrario, i messaggi sono abilitati per varie lingue in quanto i messaggi del sistema sono utilizzati per la gestione e i messaggi utente sono visualizzati ai clienti.

Gestione degli errori delle maschere JSP

La gestione degli errori per le maschere JSP può essere eseguita in vari modi:

- Gestione degli errori dall'interno della pagina
  Per i file JSP che richiedono una gestione e un ripristino errori più complessi, è possibile scrivere un file direttamente per la gestione degli errori dal bean di dati. Il file JSP può rilevare le eccezioni restituite dal bean di dati o controllare i codici di errore all'interno di ciascun bean di dati a seconda di come sono stati attivati i bean di dati. Il file JSP può, quindi, eseguire un'azione di ripristino appropriata in base all'errore ricevuto. Un file JSP può utilizzare qualsiasi combinazione dei seguenti ambiti di gestione degli errori.
- JSP di errori a livello di pagina
  Un file JSP può anche specificare la propria maschera JSP di errori

predefinita da un'eccezione che si è verificata all'interno del file JSP tramite la tag degli errori JSP. In tal modo, un programma JSP può specificare la propria gestione di un errore. Un file JSP che non specifica una tag di errori JSP incontrerà un errore nella maschera di errori JSP a livello di applicazione. Nella JSP di errore a livello di pagina, deve richiamare la classe helper JSP (com.ibm.server.JSPHelper) per eseguire il rollback della transazione corrente.

- JSP di errori a livello di applicazione
  Un'applicazione in WebSphere può specificare una maschera JSP di errori predefinita quando si verifica un'eccezione all'interno di uno dei servlet o dei file JSP. La maschera JSP di errori a livello di applicazione può essere utilizzata come un gestore di errori del centro commerciale o del negozio (per un singolo modello di negozio). Nella maschera JSP degli errori a livello di applicazione, deve essere effettuata una chiamata alla classe helper del servlet per eseguire il rollback della transazione corrente. Ciò accade perché il controller Web non si trova nel percorso di esecuzione per eseguire il rollback della transazione. Quando possibile, seguire i due tipi di gestione degli errori sopra riportati. Utilizzare la strategia di gestione degli errori a livello di applicazione solo se necessario.

Capitolo 6. Implementazione dei comandi

In questa sezione vengono fornite informazioni sulla modalità di scrittura di nuovi comandi di controller, nuovi comandi di attività e nuovi comandi del bean di dati. Inoltre, viene descritto come estendere le funzionalità dei comandi già esistenti.

Nota: Business In questo capitolo non vengono descritti i comandi delle politiche aziendali. Per ulteriori informazioni sui comandi relativi alle politiche aziendali, fare riferimento a Capitolo 7, "Accordi commerciali e politiche aziendali (Business Edition)" a pagina 153.

Nuovi comandi - Introduzione

Il modello di programmazione di WebSphere Commerce definisce quattro tipi di comandi: comandi di controller, di attività, di visualizzazione e comandi del bean di dati. Quando si crea nuova logica aziendale per la propria applicazione e-commerce, è possibile che si renda necessario creare nuovi comandi di controller, di attività e del bean di dati. Nuovi comandi di visualizzazione possono non essere necessari. Più avanti in questa sezione, verranno fornite ulteriori informazioni sui comandi di visualizzazione.

I nuovi comandi devono implementare la corrispondente interfaccia (che a sua volta deve estendersi da un'interfaccia esistente). Per semplificare la scrittura dei comandi, WebSphere Commerce comprende una classe di implementazione per ciascun tipo di comando. I nuovi comandi devono estendersi da tale classe.

La seguente tabella riporta la classe di implementazione dalla quale un comando deve estendersi e l'interfaccia che deve implementare.

Tipo di comando	**Esempio nome comando**	**Si estende da**	**Implementa interfaccia di esempio**
Comando di controller	MyControllerCmdImpl	com.ibm.commerce.command.ControllerCommandImpl	MyControllerCmd
Comando di attività	MyTaskCmdImpl	com.ibm.commerce.command.TaskCommandImpl	MyTaskCmd

Tipo di comando	Esempio nome comando	Si estende da	Implementa interfaccia di esempio
Comando del bean di dati	MyDataBeanCmdImpl	com.ibm.commerce.command.DataBeanCommandImpl	MyDataBean

Nota: Gli spazi nei nomi delle classi di implementazione sono solo esemplificativi.

Il seguente diagramma illustra la relazione tra l'interfaccia e la classe di implementazione di un nuovo comando di controller con la classe di implementazione astratta esistente e l'interfaccia. Sia la classe astratta che l'interfaccia si trovano nel pacchetto com.ibm.commerce.command.

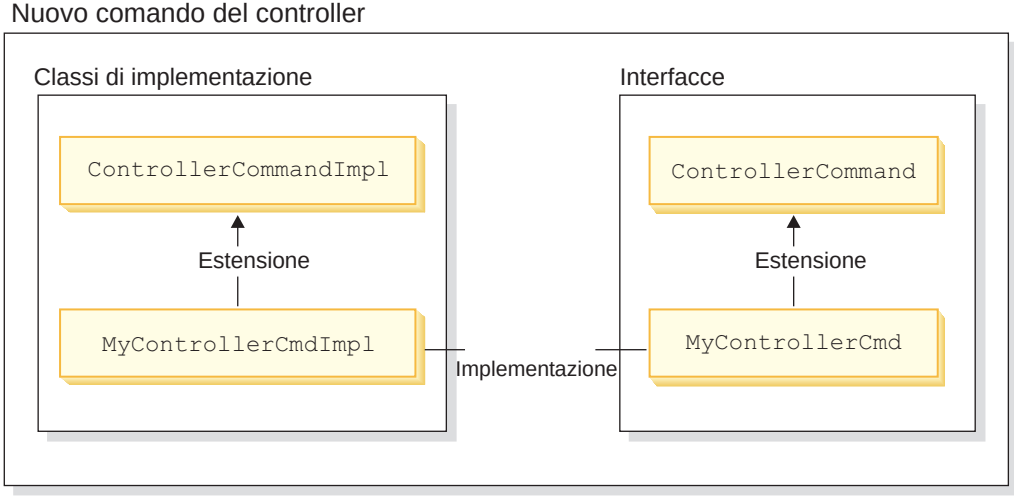


Figura 25.

Il seguente diagramma illustra la relazione tra l'interfaccia e la classe di implementazione di un nuovo comando di attività con la classe di implementazione astratta esistente e l'interfaccia. Sia la classe astratta che l'interfaccia si trovano nel pacchetto com.ibm.commerce.command.

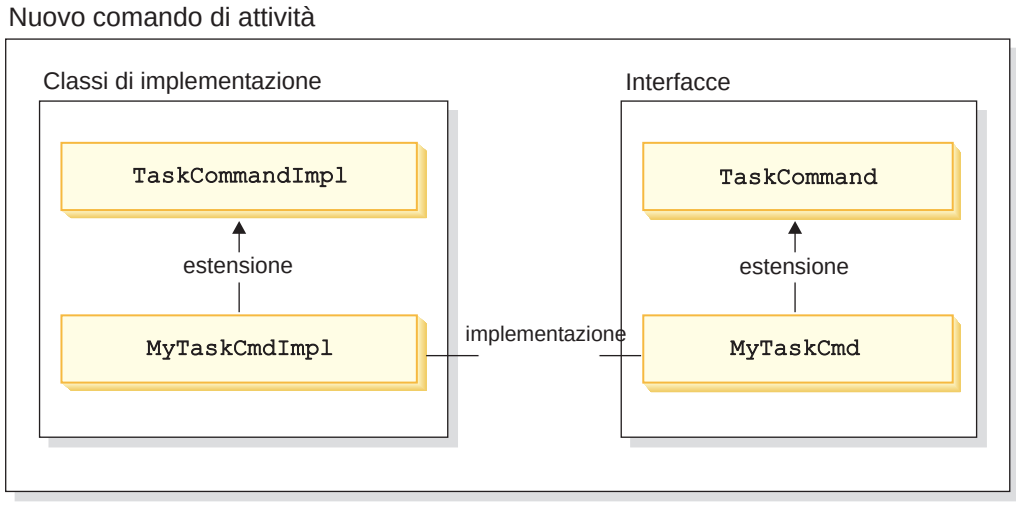


Figura 26.

Il seguente diagramma illustra la relazione tra l'interfaccia e la classe di implementazione di un nuovo comando dei bean di dati con la classe di implementazione astratta esistente e l'interfaccia. Sia la classe astratta che l'interfaccia si trovano nel pacchetto com.ibm.commerce.command.

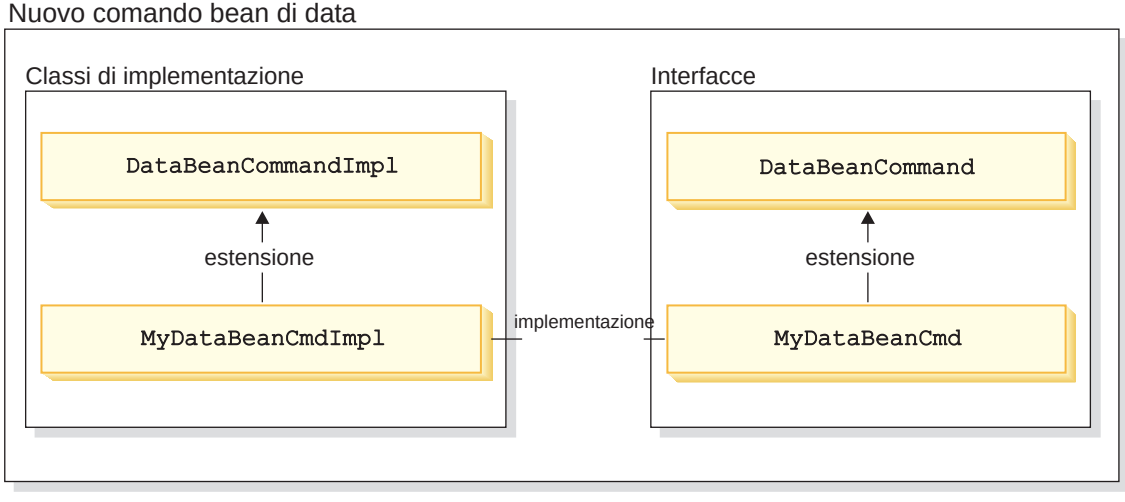


Figura 27.

Un comando di visualizzazione ha due funzioni principali: formattare una risposta e inviarla al client. Una serie di comandi di visualizzazione generici sono forniti per inviare la risposta ai client che utilizzano protocolli diversi. La funzione di formattazione viene di solito gestita dal comando di

visualizzazione che richiama la maschera JSP. Ad esempio, il comando di visualizzazione RedirectViewCommand indica al client l'URL per ricevere la risposta (la risposta viene quindi formattata da una specifica maschera JSP). Il comando di visualizzazione ForwardViewCommand invia la richiesta alla maschera JSP per la formattazione e la pagina viene visualizzata per il client.

Utilizzando questo modello di comando di visualizzazione, è possibile creare nuove *visualizzazioni* (la risposta al client) creando nuove maschere JSP. Tuttavia, la maschera JSP deve essere richiamata da uno dei comandi di visualizzazione esistenti.

Creazione di pacchetti del codice personalizzato

Quando si crea codice personalizzato, è necessario seguire una determinata struttura organizzativa del codice. In genere, il codice personalizzato viene memorizzato in pacchetti e progetti che sono separati da quelli inclusi in WebSphere Commerce.

Quando si creano nuovi comandi, inserirli in un nome pacchetto appropriato ai propri requisiti aziendali. Vale a dire, se i comandi sono destinati a un determinato negozio, inserirli in un pacchetto univoco per il negozio. Se sono destinati a più negozi, inserirli in vari pacchetti. Ad esempio, è possibile disporre dei seguenti pacchetti:

- `com.bigbusiness.storeA.commands`
- `com.bigbusiness.storeB.commands`
- `com.bigbusiness.commands`

La struttura di pacchetti sopra riportata consente di differenziare la logica aziendale a livello di negozio. Inoltre, devono essere memorizzati in un progetto separato dai progetti di WebSphere Commerce. Ad esempio, i precedenti pacchetti possono essere posizionati in un progetto denominato `BigBusinessCustomCode`.

Quando si creano nuovi bean di dati, tenerli in un pacchetto separato dalla logica di comando; tuttavia, questo pacchetto deve essere compreso nel progetto che memorizza i pacchetti di comando. Seguendo l'esempio riportato sopra, il pacchetto `com.bigbusiness.databeans` viene inserito nel progetto `BigBusinessCustomCode`.

Quando si creano nuovi bean entità, devono essere memorizzati in un progetto univoco. Quindi, è possibile disporre del progetto `BigBusinessCustomEntityBeans` che contiene il pacchetto `com.bigbusiness.objects`.

Questa modalità di creazione dei pacchetti è obbligatoria per lo sviluppo del codice.

Contesto di comando

Il contesto di comando è un identificativo di controller Web. I comandi possono ottenere informazioni dal controller Web utilizzando il contesto di comando. Esempi di informazioni disponibili sono l'ID utente, l'oggetto utente, l'identificativo della lingua e del negozio.

Quando si crea un comando, è possibile accedere al contesto di comando richiamando il metodo getCommandContext() della superclasse del comando. Il contesto di comando viene impostato sul comando di controller quando il comando viene richiamato dal controller Web. Un comando di controller deve propagare il contesto di comando a tutte le attività o comandi richiamati durante l'elaborazione. Dal contesto di comando, un comando può richiamare le seguenti informazioni chiave:

getUserId() e getUser()
Richiama l'ID utente corrente o l'oggetto utente. L'ID utente per la sessione corrente viene salvato in un contesto di sessione. Il contesto di sessione può essere tenuto inalterato in uno dei seguenti modi: utilizzando il cookie WebSphere Commerce o un oggetto sessione permanente di WebSphere Application Server. Il contesto di comando nasconde la complessità della gestione della sessione da un comando.

getStoreId(), getStore() e getStore(storeId)
Richiama il negozio associato alla richiesta corrente. Il controller Web restituisce l'ID del negozio nell'URL. Se l'ID del negozio non viene specificato nell'URL, può essere richiamato dall'oggetto sessione salvato dalla richiesta precedente. L'ambiente runtime di WebSphere Commerce gestisce una serie di oggetti a cui si accede frequentemente. Ad esempio, gestisce la serie di oggetti negozio. Un comando deve richiamare l'oggetto negozio sempre dal contesto di comando per usufruire della cache degli oggetti nel controller Web. È possibile richiamare il negozio corrente utilizzando il metodo getStore() oppure richiamare un determinato oggetto negozio utilizzando il metodo getStore(storeId) dal contesto di comando.

getLanguageId()
Restituisce l'ID della lingua da utilizzare per la richiesta corrente. Il controller Web implementa una struttura di globalizzazione. Il principio sul quale si basa questa struttura consiste nel determinare una lingua preferita dall'utente e supportata dal negozio. Se l'URL contiene un ID della lingua, il controller Web stabilisce se questa lingua è supportata dal negozio; in tal caso, è questo l'ID della lingua restituito dal metodo getLanguageId(). Nel caso in cui l'URL non contenga alcun ID della lingua, il controller Web esegue una ricerca in una struttura decisionale per stabilire se vi è un ID della lingua (supportato dal negozio) nell'oggetto sessione corrente, oppure

controlla le preferenze registrate dell'utente oppure restituisce l'ID della lingua predefinito relativo al negozio.

getCurrency()
Restituisce la valuta da utilizzare per la richiesta corrente. Dal momento che la valuta fa parte della struttura di globalizzazione, la logica di questo metodo è simile a quella del metodo getLanguageId().

getCurrentTradingAgreements() e getTradingAgreement(tradingAgreementId)
Restituisce il set di accordi commerciali utilizzati per la sessione corrente. Questo set può essere costituito da accordi commerciali concessi all'utente oppure può essere un sottoinsieme definito mediante il comando ContractSetInSession. Un comando deve richiamare sempre l'oggetto accordo commerciale dal contesto di comando per usufruire della cache degli oggetti del controller Web. È possibile richiamare l'accordo commerciale corrente utilizzando il metodo getCurrentTradingAgreements() oppure un oggetto accordo commerciale specifico utilizzando il metodo getTradingAgreement(tradingAgreementId) dal contesto di comando.

Il contesto di comando deve essere utilizzato come oggetto di sola lettura. I metodi di impostazione relativi a tale contesto non devono essere richiamati. Tali metodi sono riservati all'ambiente runtime di WebSphere Commerce e potrebbero non essere compresi nei rilasci futuri.

Per ulteriori dettagli sull'API (Application Programming Interface) dei comandi, consultare la sezione "Riferimenti" della guida in linea di WebSphere Commerce.

Nuovi comandi di controller

Come già specificato, un nuovo comando di controller deve estendersi dalla classe di comando di controller astratta (com.ibm.commerce.command.ControllerCommandImpl). Quando si scrive un nuovo comando di controller, è necessario sovrascrivere i seguenti metodi dalla classe astratta:

- isGeneric()
- isRetriable()
- setRequestProperties(com.ibm.commerce.datatype.TypedProperty reqParms)
- validateParameters()
- getResources()
- performExecute()

Ciascuno di questi metodi è descritto più dettagliatamente nelle sezioni seguenti.

Metodo isGeneric

Nell'implementazione standard di WebSphere Commerce esistono vari tipi di utente. Tra questi, gli utenti generici, gli ospiti e gli utenti registrati. Tra i gruppi di utenti registrati esistono clienti e responsabili.

L'utente generico dispone di un ID utente comune utilizzato in tutto il sistema. Questo ID utente comune supporta la navigazione generale nel sito in un modo che riduce al minimo l'utilizzo delle risorse di sistema. Si tratta di un tipo di navigazione molto più efficiente poiché il controller Web non deve richiamare un oggetto utente per i comandi che possono essere richiamati da un utente generico.

Il metodo isGeneric restituisce un valore booleano che specifica se il comando può essere richiamato da un utente generico. Il metodo isGeneric di una superclasse del comando di controller imposta il valore su `false` (sta ad indicare che il comando può essere richiamato da un utente registrato o da un utente ospite). Se il nuovo comando di controller creato può essere richiamato da utenti generici, sovrascrivere questo metodo per restituire `true`.

Se il nuovo comando non acquisisce o non crea le risorse associate a questo utente, sovrascrivere questo metodo in modo che restituisca `true`. Un esempio di un comando che può essere richiamato da un utente generico è il comando ProductDisplay. Questo comando consente agli utenti di visualizzare i prodotti. Un esempio di un comando per il quale un utente deve essere un utente non iscritto o registrato (e quindi isGeneric restituisce `false`) è il comando OrderItemAdd.

Quando isGeneric restituisce il valore `true`, il controller Web non crea un nuovo oggetto utente per la sessione corrente. Quindi, i comandi che possono essere richiamati da un utente generico sono più veloci, in quanto il controller Web non deve richiamare un oggetto utente.

Di seguito è riportata la sintassi per utilizzare questo metodo per abilitare utenti generici a richiamare un comando:

```
public boolean isGeneric()
{
     return true;
}
```

Metodo isRetriable

Il metodo isRetriable restituisce un valore booleano che specifica se il comando può essere eseguito nuovamente su un'eccezione verificatasi durante il rollback della transazione. Il metodo isRetriable della superclasse del comando di controller restituisce un valore `false`. Se il comando può essere eseguito nuovamente in un'eccezione di rollback della transazione, questo metodo deve essere sovrascritto e deve essere restituito il valore `true`.

Un esempio di comando che non deve essere eseguito nuovamente in caso di eccezione di transazione è il comando OrderProcess. Questo comando richiama il processo di autorizzazione del pagamento di altri produttori. Non può essere eseguito nuovamente perché l'autorizzazione non può essere annullata. Un esempio di comando che può essere eseguito nuovamente è il comando ProductDisplay.

Di seguito è riportata la sintassi per l'abilitazione del comando da eseguire di nuovo nel caso in cui si verifichi un'eccezione di rollback:

```
public boolean isRetriable()
{
   return true;
}
```

Metodo setRequestProperties

Il metodo setRequestProperties viene richiamato dal controller Web per inviare tutte le proprietà di immissione al comando di controller. Il comando di controller deve analizzare le proprietà di immissione e impostare le singole proprietà esplicitamente in questo metodo. Questa impostazione esplicita delle proprietà da parte del comando di controller stesso introduce il concetto delle proprietà di sicurezza.

Di seguito viene riportata la sintassi di questo metodo:

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty reqParms)
{

     // parse the input properties and explicitly set each parameter

}
```

Metodo validateParameters

Il metodo validateParameters viene utilizzato per eseguire una verifica iniziale dei parametri e per eventuali attività di risoluzione dei parametri. Ad esempio, è possibile utilizzarlo per la risoluzione di `orderId=*`. Questo metodo viene richiamato prima dei due metodi getResources e performExecute. Per ulteriori dettagli su questa sequenza, consultare "Interazioni del controllo accessi" a pagina 105.

Metodo getResources

Questo metodo viene utilizzato per implementare il controllo accessi a livello di risorsa. Restituisce un vettore di coppie risorsa-azione con le quali funziona il comando. Nel caso in cui non venga restituito nulla, il controllo accessi a livello di risorsa non viene eseguito. Per ulteriori informazioni sul controllo accessi, consultare Capitolo 4, "Controllo accessi" a pagina 91.

Metodo performExecute

Il metodo performExecute contiene la logica aziendale per i comandi da creare. Deve richiamare il metodo performExecute della superclasse del comando prima che venga eseguita nuova logica aziendale. Alla fine, deve restituire un nome di visualizzazione.

Di seguito è riportato un esempio della sintassi del metodo performExecute in un nuovo comando di controller. In questo caso, la risposta utilizza un comando di visualizzazione di reindirizzamento; tuttavia, può utilizzare un comando di visualizzazione diretto o di inoltro.

```
public void performExecute() throws ECException
{
     super.performExecute();

//////////////////////////////////////////////
//  your business logic                     //
//////////////////////////////////////////////

// Create a new TypedProperty for response properties.
TypedProperty rspProp = new TypedProperty();

// set response properties
rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
//////////////////////////////////////////////////////
//  The following line is optional.  The VIEWREG   //
//  table can specify the redirect URL.            //
//////////////////////////////////////////////////////

rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

//////////////////////////////////////////////////////////////////////////
//   If you are using a forward view, you can set the                   //
//   response properties as follows:                                    //
//   TypedProperty rspProp = new TypedProperty();                       //
//   rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");                //
//   rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");              //
//                                                                      //
//   Again, it is optional to explicitly set the name of the JSP template.//
//   The VIEWREG table can specify the JSP template.                    //
//////////////////////////////////////////////////////////////////////////

setResponseProperties(rspProp);
}
```

Se si specifica l'URL di reindirizzamento nel metodo performExecute ed esiste una voce nella tabella VIEWREG, il valore specificato nel codice ha la precedenza sul valore della tabella VIEWREG. Lo stesso ordine di precedenza vale per la specificazione di una maschera JSP nel codice.

Comandi di controller con tempi di esecuzione lunghi

Se l'esecuzione di un comando di controller richiede tempi lunghi, è possibile suddividere il comando in due comandi. Il primo comando, che viene eseguito come risultato di una richiesta URL, aggiunge semplicemente il secondo comando al programma di pianificazione, in modo che venga eseguito come processo in background. Il seguente diagramma illustra questo processo:

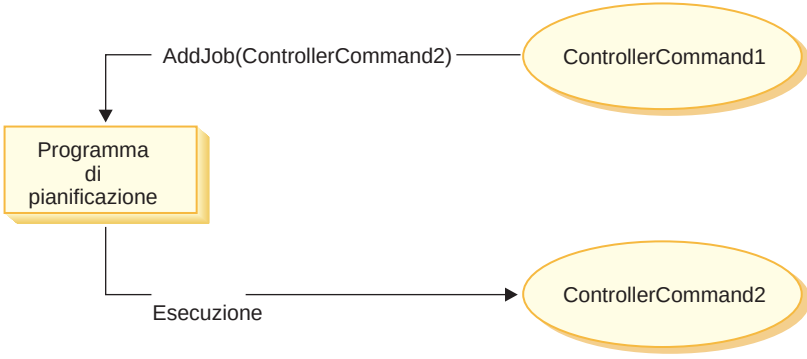


Figura 28.

Il processo mostrato nel diagramma precedente è il seguente:

1. ControllerCommand1 viene eseguito come risultato di una richiesta dell'URL.
2. ControllerCommand1 aggiunge un processo allo Scheduler. Il processo è ControllerCommand2. ControllerCommand1 restituisce una visualizzazione, immediatamente dopo l'aggiunta del processo allo Scheduler.
3. Lo Scheduler esegue ControllerCommand2 come processo in background.

In questo scenario, il client di solito esegue la scansione del risultato da ControllerCommand2. ControllerCommand2 scrive lo stato del processo nel database.

Formattazione delle proprietà di immissione per visualizzare i comandi

Quando un comando di controller viene eseguito completamente, restituisce il nome di una visualizzazione da eseguire. È possibile che sia necessario inviare a questa visualizzazione diverse proprietà di immissione. Esistono ben tre origini dei parametri di immissione, come specificato nell'elenco riportato di seguito:

- Le proprietà predefinite memorizzate nella colonna PROPERTIES della tabella CMDREG
- Le proprietà predefinite della colonna PROPERTIES della tabella VIEWREG

- Le proprietà di immissione dell'URL

Per ulteriori informazioni sulle modalità di unione e di impostazione di queste proprietà negli attributi relativi alla maschera JSP, consultare "Impostazione degli attributi JSP - Panoramica" a pagina 45. Questa sezione descrive le modalità di formattazione delle proprietà di immissione per visualizzare un comando.

Per quanto concerne il reindirizzamento dei comandi di visualizzazione, verranno illustrati i due seguenti argomenti:

- Ridimensionamento di una stringa di query per supportare il reindirizzamento dell'URL
- Gestione di un URL di reindirizzamento a lunghezza limitata

Per quanto riguarda i comandi di visualizzazione di inoltro, verranno esaminate l'enumerazione e l'impostazione dei parametri di immissione come attributi in HttpServletRequestObject.

Ridimensionamento dei parametri di immissione in una stringa di query per HttpRedirectView

Tutti i parametri di immissione inviati a un comando di visualizzazione di reindirizzamento vengono ridimensionati in una stringa di query per il reindirizzamento di un URL. Ad esempio, l'immissione per il comando di visualizzazione di reindirizzamento contiene le seguenti proprietà:

```
URL = "MyView?p1=v1&p2=v2";
ip1 = "iv1";   // input to orginal controller command
ip2 = "iv2" ;  // input to original controller command
op1 = "ov1";
op2 = "ov2";
```

In base ai parametri di immissione precedenti, l'URL finale è

```
MyView?p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2
```

Se il comando deve utilizzare SSL, i parametri vengono crittografati, quindi l'URL finale viene riportato come

```
MyView?krypto=encrypted_value_of"p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2"
```

Gestione di un URL di reindirizzamento a lunghezza limitata

Per impostazione predefinita, tutti i parametri di immissione del comando di controller vengono estesi al comando di visualizzazione di reindirizzamento. Se esiste un limite per il numero di caratteri contenuti nell'URL di reindirizzamento, ciò può rappresentare un problema. Un esempio di limite applicato alla lunghezza dell'URL lo si ha quando il client utilizza il browser Internet Explorer. In questo caso, l'URL non può superare i 2083 byte. In caso contrario, l'URL viene troncato. Pertanto, possono verificarsi problemi quando

vi sono numerosi parametri di immissione oppure quando si utilizza la crittografia; infatti, di solito una stringa di crittografia è due o tre volte più lunga di una stringa non crittografata.

Quando vi sono URL di reindirizzamento a lunghezza limitata, è possibile utilizzare due approcci:

1. Sovrascrivere il metodo getViewInputProperties nel comando di controller per restituire solo i set di parametri richiesti che devono essere inviati al comando di visualizzazione di reindirizzamento.
2. Utilizzare un carattere speciale nei parametri dell'URL per indicare i parametri da rimuovere dalla stringa dei parametri di immissione.

Per illustrare questi due approcci, si consideri il seguente set di parametri di immissione per il comando di controller:

```
URL="MyView";
// All of the following are inputs to the original controller command.
ip1="ipv1";
ip2="ipv2";
ip3="ipv3";
iq1="iqv1";
iq2="iqv2";
ir1="ipr1";
ir2="ipr2";
is="isv";
```

Sovrascrivendo il metodo getViewInputProperties, è possibile scrivere il nuovo metodo in modo che i seguenti parametri vengano inviati al nuovo comando:

```
ir2="ipr2";
is="isv";
```

Utilizzando il secondo approccio, è possibile richiamare il comando di visualizzazione utilizzando parametri speciali per indicare che determinati parametri devono essere rimossi. Ad esempio, è possibile ottenere lo stesso risultato specificando il testo riportato di seguito come parametro dell'URL:

```
URL="MyView?ip*=&iq*=&ir1="
```

Questo parametro indica alla struttura runtime WebSphere Commerce che:

- Tutti i parametri i cui nomi iniziano con `ip*=` devono essere rimossi.
- Tutti i parametri i cui nomi iniziano con `iq*=` devono essere rimossi.
- Tutti i parametri i cui nomi iniziano con `ir1=` devono essere rimossi.

Impostazione degli attributi nell'oggetto HttpServletRequest per HttpForwardView

L'HttpForwardViewCommandImpl predefinito enumera tutti i parametri inviati al comando e li imposta come attributi nell'oggetto HttpServletRequest.

Ad esempio, se l'oggetto requestProperties inviato al comando di visualizzazione di reindirizzamento contiene i seguenti parametri:

```
p1="pv1";
p2="pv2";
p3=pv3;   // pv3 is an object
```

alla maschera JSP vengono inviati i seguenti attributi utilizzando il metodo request.setAttribute():

```
request.setAttribute("p1", "pv1");
request.setAttribute("p2", "pv2");
request.setAttribute("p1", pv1);
request.setAttribute("RequestProperties", requestProperties);
request.setAttribute("CommandContext", commandContext);
```

dove `requestProperties` è l'oggetto TypedProperty inviato al comando, `commandContext` è l'oggetto contesto di comando inviato al comando e `p1`, `p2` e `p3` sono i parametri definiti nell'oggetto requestProperties.

Rollback e commit del database per i comandi di controller

I dati vengono spesso creati o aggiornati mediante l'esecuzione di un comando di controller. In molti casi, è necessario aggiornare il database con le nuove informazioni alla fine della transazione. La transazione viene gestita dal controller Web.

Il controller Web segnala l'inizio della transazione prima di richiamare il comando di controller. Una volta completata l'esecuzione, il comando di controller restituisce un nome visualizzazione al controller Web. Il controller Web è responsabile anche della segnalazione della fine della transazione. Il punto reale in cui termina la transazione (prima di richiamare la visualizzazione) dipende dal tipo di visualizzazione utilizzata.

Esistono tre tipi di comando di visualizzazione:

- Comando di visualizzazione di inoltro
- Comando di visualizzazione di reindirizzamento
- Comando di visualizzazione diretto

Il controller Web determina il comando da utilizzare per la visualizzazione, cercando il nome della visualizzazione all'interno della tabella VIEWREG.

Se la voce all'interno della tabella VIEWREG specifica l'utilizzo di ForwardViewCommand, allora il controller Web inoltra i risultati del comando di controller alla relativa classe di implementazione ForwardViewCommand (specificati anche in VIEWREG). Il comando di visualizzazione viene eseguito all'interno del contesto della transazione corrente. In questo caso, non si

verifica alcun commit o rollback del database fino a che l'esecuzione del comando di visualizzazione non viene completata.

Se la voce all'interno della tabella VIEWREG specifica l'utilizzo di RedirectViewCommand, allora il controller Web inoltra i risultati del comando di controller alla relativa classe di implementazione RedirectViewCommand. Il comando di visualizzazione quindi opera al di fuori dell'ambito della transazione corrente e il commit o il rollback del database si verifica prima che venga richiamato il comando di visualizzazione di reindirizzamento.

Se la voce all'interno della tabella VIEWREG specifica l'utilizzo di DirectViewCommand, allora il controller Web inoltra i risultati del comando di controller alla relativa classe di implementazione DirectViewCommand. Il comando di visualizzazione viene eseguito all'interno del contesto della transazione corrente. In questo caso, non si verifica alcun commit o rollback del database fino a che l'esecuzione del comando di visualizzazione non viene completata. Nota: ForwardViewCommand e DirectViewCommand sono simili. ForwardViewCommand inoltra i risultati a una maschera JSP. Per contro, DirectViewCommand riceve i risultati come flusso di immissione e li passa come flusso di emissione. Esso utilizza il metodo getRawDocument (che gestisce i dati come byte) oppure il metodo getTextDocument (che invece gestisce i dati come testo).

Nel caso in cui il comando di visualizzazione venga eseguito nello stesso ambito della transazione del comando di controller, un errore nel comando di visualizzazione provoca un rollback dell'intera transazione. Non è detto che questo sia il risultato desiderato: dipende dalla propria logica aziendale.

Esempio di un ambito della transazione con un comando di controller

Per illustrare le differenze all'interno dell'ambito della transazione per un comando di controller in base al tipo di comando di visualizzazione utilizzato, considerare i seguenti esempi.

Caso 1: Esecuzione della visualizzazione all'interno dell'ambito della transazione del comando di controller

Si supponga di aver creato un nuovo comando di controller denominato `YourControllerCmdA`. Il metodo performExecute del comando includerà quanto segue:

```
.
.
// Create a new TypedProperty object for output.
TypedProperty rspProp = new TypedProperty();

//////////////////////
// Business logic   //
//////////////////////
```

```
// Return the view
rspProp.put(ECConstants.EC_VIEWTASKNAME, "YourView");
SetResponseProperties(rspProp);
```

Nella precedente sezione di codice, il comando di controller restituisce "YourView" come visualizzazione. Questo valore viene registrato nella tabella VIEWREG. Di seguito viene riportato un esempio di istruzione per registrare YourView.

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView',  -1, XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView.jsp');
```

dove XX è l'identificativo del negozio. Dal momento che la visualizzazione utilizza la classe di implementazione `com.ibm.commerce.command.HttpForwardViewCommandImpl`, il controller Web utilizza il comando di visualizzazione di inoltro generico.

In base alla precedente registrazione del comando, il controller Web avvia il file `YourView.jsp` all'interno dell'ambito della transazione del comando di controller. Se si verifica un errore nel file `YourView.jsp`, la transazione non riesce e si verifica un rollback del database. Di conseguenza, l'esecuzione del comando di controller termina con esito negativo.

Caso 2: Esecuzione della visualizzazione all'esterno dell'ambito della transazione del comando di controller

Si supponga di desiderare di disporre delle informazioni relative al database, anche nel caso in cui si verifica un errore nella visualizzazione. Per consentire l'esecuzione della visualizzazione all'esterno dell'ambito della transazione del comando di controller, è necessario che la visualizzazione venga eseguita come reindirizzamento.

Per eseguire la visualizzazione come reindirizzamento, il metodo performExecute del comando di controller restituisce la visualizzazione come riportato di seguito:

```
.
.
// Create a new TypedProperty object for output.
TypedProperty rspProp = new TypedProperty();

//////////////////////
// Business logic   //
//////////////////////

// Return the view
rspProp.put(ECConstants.EC_VIEWTASKNAME, EC_GENERIC_REDIRECTVIEW);
rspProp.put(EC_Constants.EC_REDIRECTURL, "YourView2");
```

La seguente istruzione SQL di esempio supporta la strategia di reindirizzamento:

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView2',  -1, XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView2.jsp');
```

dove XX è l'identificativo di negozio.

Dal momento che il comando invia il valore EC_GENERIC_REDIRECTVIEW come parametro di proprietà della risposta, il controller Web utilizza il comando di visualizzazione di reindirizzamento generico. La visualizzazione di reindirizzamento generica viene registrata nella tabella VIEWREG con le seguenti informazioni:

- ViewName = RedirectView
- DeviceFmt_Id = -1
- InterfaceName = com.ibm.commerce.command.RedirectViewCommand
- ClassName = com.ibm.commerce.command.HttpRedirectViewCommandImpl

Il controller Web richiama il comando di visualizzazione di reindirizzamento generico, che rileva l'URL di reindirizzamento come proprietà dell'immissione. La risposta è il reindirizzamento all'URL. Una volta verificatosi il reindirizzamento, viene richiamato il valore YourView2. Quest'ultimo viene implementato come una visualizzazione di inoltro generica.

Nuovi comandi di attività

Un nuovo comando di attività deve estendersi dalla classe di comando di attività astratta (com.ibm.commerce.command.TaskCommandImpl). Come illustrato nel diagramma a pagina 131, il nuovo comando di attività deve essere definito nel modo seguente:

```
public class MyTaskCmdImpl extends com.ibm.commerce.command.TaskCommandImpl
     implements MyTaskCmd {

}
```

Tutte le proprietà di immissione ed emissione devono essere definite nell'interfaccia del comando, ad esempio MyTaskCmd. Il chiamante programma l'interfaccia del comando di attività piuttosto che la classe di implementazione del comando di attività. In tal modo è possibile avere più implementazioni del comando di attività (una per ciascun negozio) senza che il chiamante debba preoccuparsi delle classe di implementazione da chiamare.

Tutti i metodi definiti nell'interfaccia devono essere implementati nella classe di implementazione. Poiché il contesto di comando deve essere impostato dal chiamante (un comando di controller), il comando di attività non deve impostare il contesto di comando. Tuttavia, il comando di attività può ottenere informazioni dal controller Web utilizzando il contesto di comando.

Oltre all'implementazione dei metodi definiti nell'interfaccia del comando di attività, è necessario sostituire il metodo performExecute dalla classe com.ibm.commerce.command.TaskCommandImpl.

Il metodo performExecute contiene la logica aziendale per l'unità di lavoro eseguita dal comando di attività. Dovrebbe richiamare il metodo performExecute della superclasse del comando di attività, prima di eseguire qualsiasi logica aziendale. La seguente sezione di codice riporta un esempio del metodo performExecute per un comando di attività:

```
public void performExecute() throws ECException
{
     super.performExecute();

     // Include your business logic here.

     // Set output properties so the controller command
     // can retrieve the result from this task command.
}
```

La struttura di runtime richiama il metodo getResources del comando di controller per determinare le risorse che possono essere protette alle quali il comando accederà. Potrebbe essere il caso in cui un comando di attività è in esecuzione insieme a un comando di controller e provi ad accedere alle risorse che non sono state restituite dal metodo getResources del comando di controller. In questo caso, il comando di attività stesso è capace di implementare un metodo getResources in modo che sia fornito un controllo accessi alle risorse protette.

Per impostazione predefinita, getResources restituisce `null` per un comando di attività e la verifica del controllo accessi a livello di risorsa non viene eseguita. Pertanto, è necessario sovrascrivere questo metodo se il comando di attività accede a risorse che possono essere protette.

Personalizzazione di comandi esistenti

Questa sezione descrive le varie modalità con cui è possibile personalizzare un comando esistente di controller, di attività e del bean di dati.

Personalizzazione di comandi di controller esistenti

Un comando di controller contiene la logica aziendale relativa a un processo aziendale. I comandi possono eseguire singole unità di lavoro di un processo

aziendale. Esistono vari modi in cui è possibile personalizzare un comando di controller, alcuni dei quali coinvolgono i comandi di attività personalizzati.

Se viene personalizzato un comando di controller, è necessario:

- Aggiungere ulteriore logica ed elaborazione a un comando di controller esistente: prima o dopo la logica aziendale esistente oppure sia prima che dopo.
- Sostituire uno o più comandi di attività. Ciò consente di modificare la modalità di esecuzione di una determinata fase del processo aziendale.
- Sostituire la visualizzazione richiamata dal comando di controller.

Nelle seguenti sezioni vengono fornite informazioni dettagliate su come effettuare tali modifiche.

Aggiunta di nuova logica aziendale a un comando di controller esistente

Si supponga di disporre di un comando di controller di WebSphere Commerce esistente denominato `ExistingControllerCmd`. In base alle convenzioni di denominazione di WebSphere Commerce, questo comando presenta una classe di interfaccia denominata `ExistingControllerCmd` e una classe di implementazione denominata `ExistingControllerCmdImpl`. A questo punto, in base a una richiesta aziendale è necessario aggiungere nuova logica aziendale a questo comando. Parte della logica deve essere eseguita prima della logica del comando esistente, mentre l'altra parte deve essere eseguita dopo la logica del comando esistente.

Il primo passo per la creazione di nuova logica aziendale è la creazione di una nuova classe di implementazione che estenda la classe di implementazione originale. In questo esempio, viene creata una nuova classe `ModifiedControllerCmdImpl` che si estende alla classe `ExistingControllerCmdImpl`. Questa nuova classe dovrebbe implementare l'interfaccia originale (`ExistingControllerCmd`).

In questa nuova classe è necessario creare un nuovo metodo `performExecute` per sovrascrivere il metodo `performExecute` del comando esistente. Nel nuovo metodo `performExecute`, vi sono due modi per inserire nuova logica aziendale: è possibile includere il codice direttamente nel comando di controller oppure è possibile creare un nuovo comando di attività per eseguire la nuova logica. Se si crea un nuovo comando di attività, è necessario creare un'istanza del nuovo oggetto comando di attività dal comando di controller.

La seguente sezione di codice illustra come aggiungere logica aziendale a un comando di controller esistente includendo la logica direttamente nel comando:

```
public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
```

```
    public void performExecute()
       throws com.ibm.commerce.exception.ECException
       {

          /* Insert new business logic that must be
             executed before the original command.
          */

          // Execute the original command logic.
          super.performExecute();

          /* Insert new business logic that must be
             executed after the original command.
          */
        }
  }
```

La seguente sezione di codice illustra come aggiungere nuova logica aziendale all'inizio di un comando di controller esistente creando un'istanza del nuovo comando di attività dal comando di controller. Inoltre, è possibile creare la nuova interfaccia del comando di attività e la classe di implementazione e registrare il comando di attività nel registro dei comandi.

```
// Import the package with the CommandFactory
import com.ibm.commerce.command.*;

public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {

      public void performExecute()
         throws com.ibm.commerce.exception.ECException
         {
            MyNewTaskCmd cmd = null;
            cmd = (MyNewTaskCmd) CommandFactory.createCommand(
               "com.mycompany.mycommands.MyNewTaskCommand",
               getStoreId());

            /*
            Set task command's input parameters, call its
            execute method and retrieve output
            parameters, as required.
            */

            super.performExecute();
         }
   }
```

Indipendentemente dal fatto che si esclude la nuova logica aziendale del comando di controller oppure si crea un comando di attività per eseguire tale logica, è necessario aggiornare la tabella CMDREG nel registro dei comandi di WebSphere Commerce in modo da associare la classe di implementazione del

nuovo comando di controller all'interfaccia del comando di controller esistente. La seguente istruzione SQL illustra un aggiornamento di esempio:

```
aggiornare CMDREG
set CLASSNAME='ModifiedControllerCmdImpl'
where INTERFACENAME='ExistingControllerCmd'
```

Sostituzione dei comandi di attività richiamati da un comando di controller

Un comando di controller di solito richiama più comandi di attività che eseguono singole attività. Tutte insieme, queste attività costituiscono quindi il processo aziendale rappresentato dal comando di controller. Potrebbe essere necessario modificare il modo in cui viene eseguito un particolare passaggio di un processo piuttosto che aggiungere una nuova logica aziendale all'inizio o alla fine del comando di controller. In questo caso, è necessario sostituire l'esecuzione dell'istanza del comando di attività che si desidera modificare con l'esecuzione dell'istanza del nuovo comando di attività in modo che l'attività venga eseguita nella maniera desiderata.

In base al modello di programmazione di WebSphere Commerce, non è necessario creare una nuova classe di implementazione del comando di controller per sostituire il comando di attività. Il comando di controller esegue l'istanza del comando di attività richiamando il metodo `createCommand` del comando factory. Il comando factory utilizza il nome dell'interfaccia del comando di attività e quindi determina la classe di implementazione corretta, in base al registro comandi. Pertanto, per sostituire il comando di attività del quale si desidera eseguire l'istanza, è necessario creare una nuova classe di implementazione del comando di attività, quindi aggiornare il registro dei comandi in modo che il nome dell'interfaccia del comando di attività originale venga associato alla nuova classe di implementazione del comando di attività. Per ulteriori informazioni, fare riferimento a "Personalizzazione dei comandi di attività esistenti" a pagina 150.

Sostituzione della visualizzazione richiamata da un comando di controller

Per sostituire la visualizzazione richiamata da un comando di controller, occorre creare una nuova classe di implementazione per il comando di controller. Creare, ad esempio, una nuova classe `ModifiedControllerCmdImpl` che si estenda a `ExistingControllerCmdImpl` e implementi l'interfaccia `ExistingControllerCmd`.

Nella classe `ModifiedControllerCmdImpl`, sovrascrivere il metodo `performExecute`. Nel nuovo metodo `performExecute`, richiamare `super.performExecute` per accertarsi che tutti i comandi vengano elaborati. Dopo aver eseguito la logica del comando, è possibile utilizzare le proprietà della risposta per sovrascrivere la visualizzazione richiamata. La seguente sezione di codice mostra come sovrascrivere la visualizzazione quando questa viene eseguita come reindirizzamento:

```
// Import the packages containing TypedProperty, and ECConstants.
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute()
         throws com.ibm.commerce.exception.ECException
         {

            // Execute the original command logic.
            super.performExecute();

            // Create a new TypedProperty for response properties.
            TypedProperty rspProp = new TypedProperty();

           // set response properties
           rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
           ////////////////////////////////////////////////////
          //  The following line is optional.  The VIEWREG   //
          //  table can specify the redirect URL.            //
          ////////////////////////////////////////////////////

          rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

          setResponseProperties(rspProp);

         }
   }
```

La seguente sezione di codice mostra come sovrascrivere la visualizzazione quando questa viene eseguita come visualizzazione di inoltro:

```
// Import the packages containing TypedProperty, and ECConstants.
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute()
         throws com.ibm.commerce.exception.ECException
         {

            // Execute the original command logic.
            super.performExecute();

            // Create a new TypedProperty for response properties.
            TypedProperty rspProp = new TypedProperty();

           // set response properties
           rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");

           ////////////////////////////////////////////////
```

```
            //  It is optional to explicitly set the name   //
            //  of the JSP template. The VIEWREG table can  //
            //  specify the JSP template.                   //
            ////////////////////////////////////////////////

            rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");

            setResponseProperties(rspProp);

        }
    }
```

Per stabilire la visualizzazione utilizzata da un comando di controller esistente, consultare la sezione Riferimenti della guida in linea di WebSphere Commerce.

Personalizzazione dei comandi di attività esistenti

Esistono due approcci standard per modificare i comandi di attività esistenti di WebSphere Commerce. Questi tre approcci consentono di:

- Aggiungere elaborazione e logica a un comando di attività esistente. prima o dopo la logica aziendale esistente oppure sia prima che dopo.
- Sostituire completamente la logica aziendale esistente con la propria logica aziendale.

Per eseguire le precedenti modifiche, si crea una nuova classe di implementazione del comando di attività. Nella sezione successiva vengono fornite ulteriori informazioni.

Aggiunta di nuova logica aziendale a un comando di attività

Si supponga di disporre di un comando di attività esistente di WebSphere Commerce denominato `ExistingTaskCmd`. In base alle convenzioni di denominazione di WebSphere Commerce, questo comando dovrebbe presentare una classe di interfaccia denominata `ExistingTaskCmd` e una classe di implementazione denominata `ExistingTaskCmdImpl`. A questo punto, in base a una richiesta aziendale è necessario aggiungere nuova logica aziendale a questo comando. Parte della logica deve essere eseguita prima della logica del comando esistente, mentre l'altra parte deve essere eseguita dopo la logica del comando esistente.

Il primo passo per la creazione di nuova logica aziendale è la creazione di una nuova classe di implementazione che estenda la classe di implementazione originale. In questo esempio, viene creata una nuova classe `ModifiedTaskCmdImpl` che si estende alla classe `ExistingTaskCmdImpl`. Questa nuova classe dovrebbe implementare l'interfaccia originale (`ExistingTaskCmd`).

Nel nuovo comando, sovrascrivere il metodo `performExecute` esistente e includere la nuova logica prima e dopo aver richiamato il metodo`super.performExecute`.

Il seguente pseudo-code mostra come aggiungere nuova logica a un comando di attività esistente:

```
public class ModifiedTaskCmdImpl extends ExistingTaskCmdImpl
     implements ExistingTaskCmd {


         /* Insert new business logic that must be
            executed before the original command.
         */

         // Execute the original command logic.
         super.performExecute();

         /* Insert new business logic that must be
            executed after the original command.
         */
}
```

È inoltre necessario aggiornare la tabella CMDREG per associare la nuova classe di implementazione all'interfaccia esistente. La seguente istruzione SQL illustra un aggiornamento di esempio:

```
aggiornare CMDREG
set CLASSNAME='ModifiedTaskCmdImpl'
where INTERFACENAME='ExistingTaskCmd'
```

Sostituzione della logica aziendale di un comando di attività esistente

Per sostituire la logica aziendale di un comando di attività esistente, è necessario creare una nuova classe di implementazione per questo comando. Questa nuova classe di implementazione deve estendere il comando di attività esistente ma non implementare l'interfaccia esistente. Inoltre, nella nuova classe di implementazione, non richiamare il metodo `performExecute` della superclasse.

Anche se può sembrare strano, è consigliabile estendere il comando che si sostituisce per motivi di compatibilità con versioni future di WebSphere Commerce. Questo approccio protegge il codice da eventuali modifiche apportate alle interfacce del comando nelle versioni future di WebSphere Commerce.

Ad esempio, si supponga di voler sostituire la logica aziendale del comando di attività `OrderNotifyCmdImpl`. In questo caso, è necessario creare un nuovo comando di attività denominato `CustomizedOrderNotifyCmdImpl`. Tale comando estende `OrderNotifyCmdImpl`. Nel nuovo comando `CustomizedOrderNotifyCmdImpl` si crea la nuova logica aziendale, ma non si richiama il metodo `performExecute` per la superclasse. Se una futura versione di WebSphere Commerce introduce nell'interfaccia un nuovo metodo chiamato `newMethod`, la versione corrispondente del comando `OrderNotifyCmdImpl` comprenderà un'implementazione predefinita di tale metodo. Quindi, poiché

il nuovo comando estende `OrderNotifyCmdImpl`, il compilatore troverà l'implementazione predefinita del nuovo metodo nel comando `OrderNotifyCmdImpl` e il nuovo comando viene protetto dalla modifica dell'interfaccia.

Per assicurarsi che la nuova classe di implementazione fornisca le stesse prestazioni di quella esistente, consultare la sezione Riferimenti della guida in linea di WebSphere Commerce.

Personalizzazione del bean di dati

Un bean di dati di solito si estende da un bean di accesso. Il bean di accesso, che può essere generato da VisualAge per Java, fornisce un modo semplice per accedere alle informazioni di un bean entità. Quando vengono apportate modifiche a un bean entità (ad esempio, aggiungendo un nuovo campo, un nuovo metodo aziendale o un nuovo finder), anche il bean di accesso subisce le stesse modifiche appena viene rigenerato. Poiché il bean di dati estende il bean di accesso, eredita automaticamente i nuovi attributi. Come risultato di questa relazione, non è richiesto alcun codice per abilitare il bean di dati all'utilizzo dei nuovi attributi dal bean entità.

Se occorre aggiungere nuovi attributi ai bean di dati che non derivano da un bean entità, è possibile estendere il bean di dati esistente utilizzando l'eredità Java. Ad esempio, se si desidera aggiungere un nuovo campo a OrderDataBean, definire MyOrderDataBean come specificato di seguito:

```
public class MyOrderDataBean extends OrderDataBean
{
     public String myNewField () {
     //  implementare qui il nuovo campo
     }

}
```

Il nuovo bean di dati deve avere anche una classe BeanInfo. Di seguito è riportato un esempio della dichiarazione per questa classe:

```
public class MyOrderDataBeanInfo extends java.beans.SimpleBeanInfo
{

}
```

VisualAge per Java fornisce uno strumento che consente di generare questa classe BeanInfo.

Capitolo 7. Accordi commerciali e politiche aziendali (Business Edition)

Le informazioni contenute in questo capitolo si applicano soltanto al WebSphere Commerce Business Edition.

Introduzione

Uno degli elementi chiave del commercio B2B (business-to-business) è la gestione delle relazioni. Per gestire una relazione tra un'azienda acquirente e un'azienda venditrice, si utilizza un accordo commerciale. Il modello di accordo commerciale utilizzato da WebSphere Commerce Business Edition supporta vari tipi di accordo commerciale come, ad esempio, contratti e RFQ (Request For Quote).

Il principale elemento di un accordo commerciale è un gruppo di termini e condizioni. Ciascun termine e condizione definisce una regola aziendale specifica da utilizzare per l'operazione commerciale. Utilizzando WebSphere Commerce Business Edition, è possibile negoziare un gruppo di termini e condizioni utilizzando un processo RFQ in linea, oppure non in linea, quindi acquisire questo gruppo utilizzando le interfacce di gestione delle relazioni aziendali in WebSphere Commerce Accelerator.

Vi sono vari metodi per definire i termini e le condizioni:

- Termini e condizioni che selezionino una delle politiche aziendali predefinite come, ad esempio, il listino prezzi e la politica di restituzione. Oppure possono selezionare una politica aziendale che è stata creata. Un oggetto termine può anche riferirsi a più oggetti politica aziendale.
- Termini e condizioni che applichino una regolazione specifica alla politica aziendale come ad esempio, una regolazione del prezzo standard.
- Termini e condizioni che definiscano un gruppo di parametri che regolano un processo aziendale. Ad esempio, possono specificare che un particolare centro di evasione ordini deve essere utilizzato da un determinato contratto.

Un contratto è formato da un insieme di termini, come illustrato nel seguente diagramma:

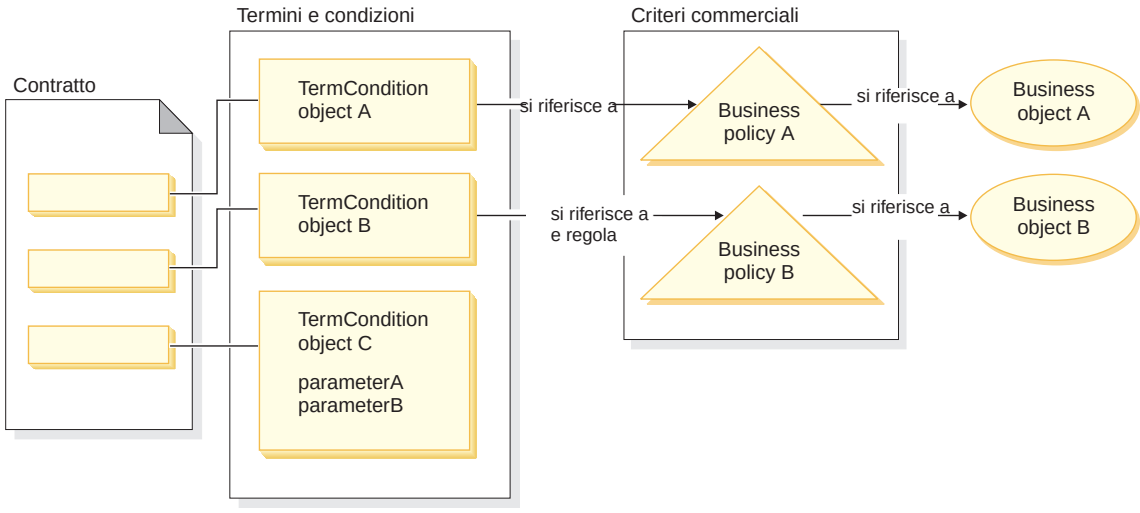


Figura 29.

Dal grafico precedente, notare quanto segue:

- Il termine "regolazione" si riferisce a una modifica apportata alla politica aziendale. Ad esempio, può essere utilizzato per applicare uno sconto al risultato di una politica aziendale in modo tale che viene applicato uno sconto del 10% sul prezzo standard. Può essere anche utilizzato per gestire la politica aziendale mediante una serie di parametri.
- Ad esempio, l'oggetto A di TermCondition può rappresentare un oggetto termini e condizioni di spedizione. In questo caso la politica aziendale A può rappresentare una modalità di spedizione e l'oggetto commerciale A la modalità di spedizione "A3" dello spedizioniere XYZ.
- Ancora, l'oggetto B di TermCondition può rappresentare un oggetto termini e condizioni di prezzo in base al quale viene applicato il 50% di sconto al prezzo definito dalla politica aziendale B. In questo caso, la politica aziendale B è la politica dei prezzi e l'oggetto commerciale B è un contenitore di posizioni commerciali che definisce la posizione della categoria principale.

Questo capitolo fornisce ai programmatori le istruzioni per la creazione di nuove politiche aziendali e nuovi termini e condizioni.

Il negozio di esempio ToolTech mostra un oggetto termini e condizioni di spedizione e un oggetto termini e condizioni di prezzo nel flusso commerciale. Per ulteriori informazioni sui dati di contratto che supportano questi esempi, fare riferimento a "Dati di contratto dell'esempio" a pagina 156.

Politiche aziendali: comandi e oggetti

Un oggetto politica aziendale contiene le seguenti informazioni:

- ID della politica
  Questa è la chiave principale per l'oggetto politica aziendale.
- Tipo di politica
  Definisce il tipo di politica aziendale. `Price` e `ProductSet` sono esempi di tipi di politica.
- Nome della politica
  Ogni politica aziendale deve avere un nome univoco.
- Entità negozio
  Il negozio o il gruppo di negozi all'interno del quale viene distribuita la politica aziendale.
- Proprietà
  Un gruppo di proprietà predefinite che possono essere inviate a un comando della politica aziendale. I comandi associati all'oggetto politica aziendale sono memorizzati nella tabella BusinessPolicyCmd.
- Periodo di validità
  Il periodo di tempo per il quale l'oggetto politica aziendale è valido.
- Comando della politica aziendale
  Nessuno o più comandi della politica aziendale che implementano la politica aziendale stessa. Di norma, un comando di politica aziendale viene richiamato da un processo commerciale che può essere un comando di attività o un comando di controller. Ad esempio, il comando `getContractPrice()` richiama termini e condizioni relativi al prezzo. Tali termini e condizioni si riferiscono a un particolare comando di politica dei prezzi utilizzato per calcolare il prezzo.

È possibile associare più comandi delle politiche aziendali a un unico oggetto politica aziendale. Ogni comando delle politiche aziendali deve implementare la stessa interfaccia definita dall'oggetto del tipo di politica aziendale. La struttura di un nuovo comando della politica aziendale viene riportata nel seguente diagramma:

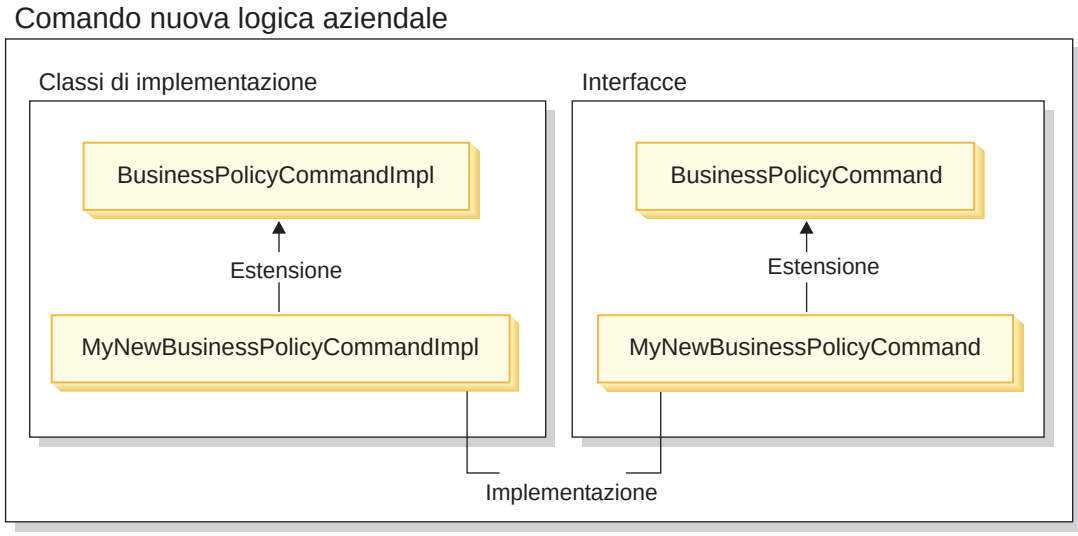


Figura 30.

Come mostrato nel precedente diagramma, per poter creare un nuovo comando della politica aziendale, è necessario creare una nuova classe di implementazione che estenda la classe di implementazione BusinessPolicyCmdImpl di WebSphere Commerce. È necessario inoltre creare una nuova interfaccia che estenda l'interfaccia BusinessPolicyCmd.

Dati di contratto dell'esempio

Questa sezione fornisce un'introduzione ad alcuni dei dati di contratto utilizzati nel negozio di esempio ToolTech.

I dati di esempio presenti nelle seguenti sezioni sono organizzati in una tabella di database. Vengono visualizzate soltanto le righe e le colonne rilevanti. Tenere presente anche che quando l'esempio viene installato gli identificativi (ad esempio CONTRACT_ID) possono avere valori diversi da quelli mostrati qui.

Dati di esempio della tabella CONTRACT

La tabella di seguito riportata illustra i dati di esempio rilevanti della tabella di database CONTRACT di ToolTech. Tener presente che per motivi di visualizzazione, le intestazioni di colonna del database appaiono nella prima colonna e la riga dei dati di esempio della tabella è mostrata nella seconda colonna.

Nome colonna	Dati di esempio
CONTRACT_ID	10007

Nome colonna	Dati di esempio
MAJORVERSION	1
MINORVERSION	0
NAME	ToolTechContractNumber 4567
MEMBER_ID	-2001
ORIGIN	0
STATE	3
USAGE	1
MARKFORDELETE	0

Dati di esempio della tabella TERMCOND

La tabella di seguito riportata illustra i dati di esempio rilevanti della tabella di database TERMCOND di ToolTech. Tener presente che per motivi di visualizzazione, le intestazioni di colonna del database appaiono nella prima colonna e le righe dei dati di esempio della tabella sono mostrate nella seconda e terza colonna.

Nome colonna	Riga di dati di esempio 1	Riga di dati di esempio 2
TERMCOND_ID	10025	10030
TCSUBTYPE_ID	PriceTCPriceListWith SelectiveAdjustment	ShippingTCShippingMode
TRADING_ID	10007	10007
STRINGFILED1	ProductSet2	
INTEGERFIELD2	10002	
INTEGERFIELD3	1	
BIGINTFIELD1	10051	
FLOATFIELD1	-50.0	
SEQUENCE	1	6

Dati di esempio della tabella POLICYTC

La tabella di seguito riportata illustra i dati di esempio rilevanti della tabella di database POLICYCT di ToolTech. Questa tabella stabilisce la relazione tra una politica e l'oggetto termini e condizioni.

	Nome colonna	
	POLICY_ID	TERMCOND_ID
Riga di dati di esempio 1	10053	10025
Riga di dati di esempio 2	10056	10030

Dati di esempio della tabella POLICY

La tabella di seguito riportata illustra i dati di esempio rilevanti della tabella di database POLICY di ToolTech.

Nome colonna	Riga di dati di esempio 1	Riga di dati di esempio 2
POLICY_ID	10053	10056
POLICYNAME	MasterCatalogPriceList	A3
POLICYTYPE_ID	Price	ShippingMode
STOREENT_ID	10051	10051
PROPERTIES	name=ToolTech& member_id=-2001	shippingMode=A3
STARTTIME	null	null
ENDTIME	null	null

Dati di esempio della tabella TRADEPOSCN

La tabella di seguito riportata illustra i dati di esempio rilevanti della tabella di database TRADEPOSCN di ToolTech.

	Nome colonna			
	READEPOSCN_ID	**MEMBER_ID**	**NAME**	**TYPE**
Riga di dati di esempio	10051	-2001	ToolTech	S

Dati di esempio della tabella SHIPMODE

La tabella di seguito riportata illustra i dati di esempio rilevanti della tabella di database SHIPMODE di ToolTech.

	Nome colonna			
	SHIPMODE_ID	**STOREENTITY_ID**	**CODE**	**CARRIER**
Riga di dati di esempio	10053	10051	A3	Spedizioniere XYZ

Estensione del modello di contratto esistente

Un contratto può essere costituito da uno o più oggetti termini e condizioni, ognuno dei quali si riferisce a una politica. Le seguenti sezioni descrivono le operazioni da eseguire per creare una nuova politica aziendale e per applicare tale politica al flusso commerciale.

Vengono di seguito riportate, come breve panoramica, le operazioni di alto livello necessarie per l'esecuzione di questa attività:

1. Creazione di una nuova politica aziendale.
   Le seguenti attività si riferiscono alla creazione di un nuovo comando di politica aziendale:
   a. Creazione di un nuovo tipo di politica aziendale (se richiesto).
      I tipi di politica aziendale forniti sono diversi, ma se quelli standard non si adattano alle proprie esigenze, crearne uno nuovo.
   b. Creazione dei un nuovo comando di politica aziendale.
   c. Registrazione della nuova politica aziendale e del nuovo comando di politica aziendale.
2. Collegamento di un oggetto termini e condizioni alla nuova politica aziendale
   È possibile collegare l'oggetto termini e condizioni esistente alla nuova politica aziendale o creare un nuovo oggetto termini e condizioni. Se si crea un nuovo oggetto termini e condizioni, è necessario effettuare le seguenti operazioni:
   a. Registrare i nuovi termini e condizioni nel database
   b. Registrare i nuovi termini e condizioni nella DTD (Document Type Definition) del contratto
   c. Creare un nuovo bean enterprise CMP per i termini e condizioni
   d. Aggiornare WebSphere Commerce Accelerator con i nuovi termini e condizioni
3. Richiamo della nuova politica aziendale durante il flusso commerciale.

Creazione di una nuova politica aziendale

La creazione di una nuova politica aziendale comporta la registrazione di una politica aziendale nel database oltre alla creazione di un nuovo comando di politica aziendale.

La creazione di un nuovo comando della politica aziendale prevede i seguenti passaggi:

1. Creazione di un nuovo tipo di politica aziendale (se richiesto).
2. Scrittura del nuovo comando della politica aziendale.
3. Registrazione della nuova politica aziendale e del relativo comando nel database.

Ognuno di questi passaggi verrà descritto in maniera più dettagliata nelle sezioni seguenti.

Creazione di un nuovo tipo di politica aziendale

In questa sezione viene descritto come creare un nuovo tipo di politica aziendale. Un tipo di politica aziendale indica l'ambito della transazione al quale si applica la politica. Tra i tipi di politica aziendale vi sono:

- Price
- ProductSet
- ShippingMode
- ShippingCharge
- Payment
- ReturnCharge
- ReturnApproval
- ReturnPayment
- InvoiceFormat

Se il tipo di politica aziendale esistente non soddisfa le proprie esigenze aziendali, è necessario creare un nuovo tipo di politica aziendale. La creazione di un nuovo tipo di politica aziendale consiste nella relativa definizione e registrazione.

Quando si definisce e si registra un nuovo tipo di politica aziendale, è necessario aggiornare le seguenti tabelle del database:

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

Nella tabella POLICYTYPE viene specificato il tipo di politica aziendale che si desidera creare. Tale tabella contiene un'unica colonna, POLICYTYPE_ID, che è la chiave principale. Un valore di esempio è `Price`. Se si crea un nuovo tipo di politica aziendale, accertarsi di disporre di un valore POLICYTYPE_ID univoco.

La tabella PLCYTYCMIF è la tabella in cui viene specificata la relazione tra il tipo di politica aziendale e l'interfaccia del comando. In altre parole, per ogni tipo di politica aziendale, specifica l'interfaccia del comando Java per l'oggetto politica aziendale. Mentre il numero di comandi della politica aziendale che implementano una politica aziendale è illimitato, ogni comando deve implementare l'interfaccia specificata.

Nella tabella PLCYTYPDSC viene descritto il tipo di politica aziendale. Essa include un identificativo della lingua della descrizione e la descrizione stessa del tipo di politica aziendale.

Per creare un nuovo tipo di politica aziendale, creare una voce in ognuna di queste tabelle. Le seguenti istruzioni SQL forniscono un esempio:

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('MyNewPolicyType');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('MyNewPolicyType',
   'com.mycompany.mybusinesspolicycommands.MyNewPolicy');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('MyNewPolicyType', -1,
   'My new policy type for example purposes.');
```

Come operazione conclusa del processo di creazione di un nuovo tipo di politica aziendale, è possibile codificare una o più nuove interfacce di tipo di politica aziendale. Tali interfacce vengono poi implementate da qualsiasi comando di politica aziendale appartenente al dominio di questo tipo di politica aziendale. Ad esempio, nel negozio di esempio ToolTech, Prezzo è definito come tipo di politica aziendale. Come tale, esistono le interfaccecom.ibm.commerce.price.commands.ResolvePriceListsCmd e com.ibm.commerce.price.commands.RetrievePricesCmd implementate fa tutti i comandi di politica aziendale relativi al prezzo.

Se non esiste un comando di politica aziendale che esegue le operazioni sul nuovo tipo di politica, non è necessario creare una nuova interfaccia. Questa situazione è piuttosto rara e nella maggior parte dei casi quando si crea un nuovo tipo di politica aziendale, è necessario creare anche una nuova interfaccia.

Quando si crea un'interfaccia di un tipo di politica aziendale, la nuova interfaccia deve estendere l'interfaccia com.ibm.commerce.command.BusinessPolicyCommand.

Scrittura del nuovo comando della politica aziendale

Per creare un nuovo comando di politica aziendale, è necessario creare un nuovo comando che implementa l'interfaccia del tipo di politica al quale il comando stesso si riferisce. Il nuovo comando deve estendere la classe di implementazione com.ibm.commerce.command.BusinessPolicyCommandImpl. Questa procedura è molto simile a quella per la creazione di un nuovo comando di controller o di attività.

Sono disponibili due approcci differenti mediante i quali è possibile inviare le proprietà di immissione a un comando della politica aziendale. Il primo modo consiste nello specificare le proprietà di immissione predefinite nella colonna PROPERTIES della tabella POLICY. Per ulteriori informazioni su questa tabella, fare riferimento alla sezione successiva.

Il secondo approccio consiste nel creare un nuovo campo nel comando per ciascuna proprietà di immissione. Per ogni campo, creare una nuova coppia di metodi getter e setter.

Impostazione di requestProperties nei comandi delle politiche aziendali

Esistono due modi in cui è possibile impostare requestProperties in un oggetto comando delle politiche aziendali. Il primo approccio prevede l'utilizzo della colonna PROPERTIES della tabella POLICY per impostare le proprietà predefinite. A tale scopo, occorre utilizzare il metodo setRequestProperties. In alternativa, è possibile fare in modo che il comando (di controller o di attività) che richiama il comando delle politiche aziendali, imposti in maniera esplicita altre proprietà richieste.

Quando si crea un nuovo comando per le politiche aziendali, è necessario sovrascrivere il metodo setRequestProperties predefinito per includere la logica e impostare in maniera esplicita ogni singolo parametro contenuto nell'oggetto requestProperties.

Si consideri l'esempio di un nuovo comando per le politiche aziendali il cui nome interfaccia è MyNewBusinessPolicyCmd e che presenta la classe di implementazione MyNewBusinessPolicyCmdImpl.

Si supponga che la voce della nella tabella POLICY relativa a questo comando includa i seguenti valori nella colonna PROPERTIES:

- defaultProperty1=apple
- defaultProperty2=orange
- defaultProperty3=banana

L'interfaccia relativa a questo nuovo comando è definita nel modo seguente:

```
public interface MyNewBusinessPolicyCmd extends
   com.ibm.commerce.command.BusinessPolicyCmd {
      java.lang.String defaultCommandClassName =
         'com.mycompany.mycommands.MyNewBusinessPolicyCmdImpl';
      public void setProperty1();
      public void setProperty2();
}
```

La classe di implementazione relativa a questo nuovo comando è definita nel modo seguente:

```
public class MyNewBusinessPolicyCmdImpl extends
   com.ibm.commerce.command.BusinessPolicyCmdImpl
   implements com.mycompany.mycommands.MyNewBusinessPolicyCmd {
      // Establish default properties that are stored in the POLICY table

      private java.lang.String defaultProperty1;
      private java.lang.String defaultProperty2;
      private java.lang.String defaultProperty3;

      // Begin to establish properties that must be set
     // by the calling command.

      //  *** property1   ***
```

```
      private java.lang.String property1;
      public java.lang.String getProperty1() {
         return property1;
         }
      public void setProperty1(java.lang.String newProperty1) {
         property1 = newProperty1;
      }

      //  *** property2   ***
      private java.lang.String property2;
      public java.lang.String getProperty2() {
         return property2;
         }
      public void setProperty1(java.lang.String newProperty2) {
         property2 = newProperty2;
         }

      // End establishing properties that must be set
      // by the calling command.

      /* Upon instantiation the business policy command sets all
         default properties from the POLICY table into the
         requestProperties object.  The calling command
         is responsible for setting any other required properties.
      */

      public void setRequestProperties(com.ibm.commerce.datatype.TypedProperty
         requestProperties) {
         //  Get the default properties defined in the POLICY table
         setDefaultProperty1(requestProperties.get("defaultProperty1"));
         setDefaultProperty2(requestProperties.get("defaultProperty2"));
         setDefaultProperty3(requestProperties.get("defaultProperty3"));
         }

}
```

Il comando che richiama il nuovo comando di politica aziendale deve essere definito in modo simile al seguente:

```
public class MyCallerCommandImpl
   extends com.ibm.commerce.command.TaskCommandImpl
   implements com.mycompany.mycommands.MyCallerCommand {

   /* Include all elements and processing required for the
      task command.
   */

   // Determine the policy ID and setPolicyId

   // Call the business policy command.

   cmd = (MyNewBusinessPolicyCmd) CommandFactory
            createPolicyCommand(policyId);

   // Set required properties
```

```
    cmd.setProperty1("Fruit salad");
    cmd.setProperty2("Favorite food");

  cmd.execute();
}
```

Registrazione della nuova politica aziendale e del relativo comando

Una volta creato il nuovo comando della politica aziendale, è necessario registrare sia la politica che il relativo comando all'interno del database.

Le politiche aziendali vengono registrate nella tabella POLICY. Questa tabella contiene le seguenti colonne:

- POLICY_ID
  La chiave principale. Si tratta dell'ID della politica.
- POLICYNAME
  Un nome univoco della politica.
- POLICYTYPE_ID
  L'ID del tipo di politica. Questa è la chiave esterna per la tabella POLICYTYPE.
- STOREENT_ID
  Il negozio o il gruppo di negozi al quale si applica la politica.
- PROPERTIES
  Le proprietà predefinite che possono essere impostate per il comando della politica aziendale. Queste vengono specificate come coppia nome-valore, ad esempio `parm1=val1&parm2=val2`.
- STARTDATE
  La data di inizio (specificata come formato orario) della politica. Se questo valore è NULL, l'inizio della politica è immediato.
- ENDDATE
  La data di fine (specificata come formato orario) della politica. Se il valore è NULL, non c'è alcuna data di fine.

Dopo aver registrato la nuova politica nella tabella POLICY, è necessario registrare una relazione tra la politica e il comando che implementa la politica aziendale stessa. A tale scopo, viene utilizzata la tabella POLICYCMD. Essa contiene le seguenti colonne:

- POLICY_ID
  La chiave esterna che fa riferimento alla tabella POLICY.
- BUSINESSCMDCLASS
  Il comando della politica aziendale che implementa la politica.
- PROPERTIES
  Le proprietà predefinite che possono essere impostate per il comando della politica aziendale. Queste vengono specificate come coppia nome-valore, ad esempio `parm1=val1&parm2=val2`.

Collegamento di un oggetto termini e condizioni a una nuova politica aziendale

Nella struttura delle politiche e dei contratti di WebSphere Commerce, i termini e le condizioni (oppure semplicemente *termini*) consentono di descrivere un accordo tra un acquirente e un rivenditore. Gli oggetti termini e condizioni possono essere utilizzati in vari tipi di accordo commerciale, ad esempio un contratto e una RFQ (Request For Quotation). Di solito, questi oggetti si riferiscono a politiche aziendali con una rettifica facoltativa. Ad esempio, l'oggetto termini prezzo viene creato scegliendo uno degli oggetti politica dei prezzi. Attraverso un termine dei prezzi, un gestore degli account può effettuare delle rettifiche ai prezzi standard del negozio, come ad esempio:

- Una percentuale di sconto sull'elenco dei prezzi standard
- Una percentuale di sconto su un particolare gruppo di prodotti

Le rettifiche vengono specificate come termini.

Quando si crea una nuova politica aziendale, almeno un oggetto termini e condizioni deve riferirsi a questa politica, se essa deve essere utilizzata nel contratto. È possibile collegare un oggetto termini e condizioni esistenti a una politica aziendale (catturando la relazione tra un oggetto termini e condizioni esistente e la nuova politica aziendale nel file `B2BTrading.dtd`), oppure è possibile creare un nuovo oggetto termini e condizioni correlato alla nuova politica aziendale.

Creazione di nuovi termini e condizioni

All'interno dell'architettura di WebSphere Commerce, i nuovi oggetti termini vengono creati effettuando le seguenti operazioni:

1. Aggiornamento dello schema del database per includere i nuovi termini.
2. Aggiornamento del file `B2BTrading.dtd` per confermare il nuovo oggetto termini.
3. Creazione di un nuovo bean enterprise per i termini.
4. Aggiornamento di WebSphere Commerce Accelerator per riportare i nuovi termini oppure utilizzo del comando di caricamento contratti per creare un nuovo contratto implementando i nuovi termini.

Nelle sezioni successive, l'esempio `MyTC` rappresenta il nuovo oggetto termini.

Registrazione dei nuovi termini e condizioni nel database

Quando si crea un nuovo oggetto termini, è necessario aggiornare lo schema del database per includere questo schema. In particolare, occorre aggiornare le tabelle TCTYPE e TCSUBTYPE.

La seguente istruzione SQL illustra un esempio di aggiornamento dello schema:

```
insert into TCTYPE (TCTYPE_ID) values ('MyTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME,
   DEPLOYCOMMAND)
values ('MySubTC, 'MyTC ',
   'com.ibm.commerce.contract.objects.MySubTCAccessBean',
   'packagename.MySubTCDeployCmd');
```

Registrazione di nuovi termini e condizioni nella DTD (Document Type Definition)

Il file B2BTrading.dtd è il file DTD (Documento Type Definition) che specifica i diversi termini e condizioni che è possibile utilizzare nelle politiche aziendali. Per rendere disponibili i nuovi termini e condizioni nei contratti, è necessario aggiornare il file in modo da includere i nuovi termini e condizioni.

Una volta creato un nuovo oggetto termini, è necessario aggiungerlo alla definizione TermCondition, quindi creare un nuovo elemento che lo descriva.

Per aggiornare il file B2BTrading.dtd, effettuare le seguenti operazioni:

1. Visualizzare la seguente directory:
   - Windows *unità*:\WebSphere\CommerceServer\xml\trading
   - AIX /usr/WebSphere/CommerceServer/xml/trading
   - Solaris /opt/WebSphere/CommerceServer/xml/trading
   - Linux /opt/WebSphere/CommerceServer/xml/trading
   - 400 /QIBM/ProdData/WebCommerce/xml/trading
2. Aprire il file B2BTrading.dtd.
3. Aggiornare la definizione TermCondition con il nuovo oggetto termini. Nell'esempio riportato di seguito, l'aggiornamento viene indicato in grassetto:

   ```
   <!ELEMENT TermCondition (TermConditionDescription?,Participant*,
   CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
   PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
   PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
   OrderTC|MyTC))>
   ```

   Le interruzioni di riga sono state inserite unicamente per agevolare la visualizzazione.
4. A questo punto, aggiungere il nuovo elemento al file B2BTrading.dtd. L'esempio riportato di seguito mostra l'aggiornamento per aggiungere l'elemento MyTC che si riferisce a una politica aziendale e ha due attributi richiesti.

```
<!ELEMENT MyTC (MySubTC)>
<!ELEMENT MySubTC (PolicyReference)>
<!ATTLIST MySubTC
   attr1 CDATA #REQUIRED
   attr2 CDATA #REQUIRED
>
```

5. Salvare il file.

Creazione di un nuovo bean enterprise CMP relativo ai termini

È necessario creare un nuovo bean enterprise CMP per l'oggetto termini e condizioni. Il bean viene creato per il sottotipo termini e condizioni.

Tener presente che di norma quando si creano nuovi bean enterprise, essi vengono inseriti nel proprio gruppo di EJB piuttosto che in uno dei gruppi di EJB che contengono i bean entità WebSphere Commerce. In questo caso, comunque, poiché tutti i nuovi bean entità per termini e condizioni devono ereditare dal bean TermCondition di WebSphere Commerce., è necessario inserire i nuovi bean per termini e condizioni nel gruppo di EJB del contratto WCS.

Per creare il nuovo bean enterprise CMP per l'oggetto termini e condizioni, effettuare le seguenti operazioni in VisualAge per Java:

1. Utilizzare una procedura guidata per creare il nuovo bean enterprise procedendo come segue:
   a. Nell'Area di lavoro di VisualAge per Java, selezionare la scheda EJB.
   b. Evidenziare, quindi fare clic con il pulsante destro del mouse sul gruppo EJB del **contratto WCS** e selezionare **Aggiungi > Enterprise Bean with Inheritance**.
      Si apre la SmartGuide di creazione di un bean enterprise con eredità.
   c. Nella SmartGuide, immettere le informazioni appropriate per il bean. La seguente tabella mostra valori di esempio.

Attributo	Valore
Nome bean	`MySubTC`
Eredita da	`TermCondition`
Pacchetto	`com.ibm.commerce.contract.objects`
Classe bean	`MySubTCBean`
Interfaccia remota	`MySubTC`
Interfaccia home	`MySubTCHome`

   d. Fare clic su **Aggiungi** per aggiungere i campi CMP al bean e creare nuovi campo per bean, se richiesto. Per questo esempio, due nuovi campo CMP vengono creati con le seguenti informazioni:

Attibuto	Valore
Nome campo	`attr1`
Tipo campo	`String`
Accedi mediante i metodi getter e setter	enable
Promuovi metodi getter e setter per l'interfaccia remota	enable

Attibuto	Valore
Nome campo	`attr2`
Tipo campo	`Integer`
Accedi mediante i metodi getter e setter	enable
Promuovi metodi getter e setter per l'interfaccia remota	enable

e. Fare clic su **Fine**.

2. L'operazione successiva consiste nel creare una corrispondenza tra i campi del nuovo bean e le colonne della tabella TERMCOND. Per creare queste informazioni di corrispondenza, procedere come segue:
   a. Dal menu **EJB**, selezionare **Apri in > Mappe schemi**.
      Si apre la finestra del browser di corrispondenze.
   b. Nel pannello Mappe datastore del browser di corrispondenze, fare doppio clic su **Contratto WCS**.
   c. Nel pannello Classi persistenti, fare doppio clic su **TermCondition** quindi selezionare **MySubTC**.
   d. Dal menu Mappe tabella, selezionare **Nuova mappa tabella > Aggiungi singola mappa tabella con eredità**.
      Si apre l'editor della singola mappa tabella con eredità.
   e. Nel campo **Valore discriminatore**, immettere il valore TCSUBTYPE_ID. Ad esempio, in questo caso immettere `'MySubTC'` (inclusi gli apici) e fare clic su **OK**.
   f. Assicurarsi che **MySubTC** sia ancora selezionato nel pannello Classi persistenti. Nel pannello Mappe tabella, evidenziare e fare clic sulla tabella fare clic con il pulsante destro del mouse TERMCOND. Selezionare Edita corrispondenze di proprietà.
      Si apre la finestra dell'editor delle corrispondenze di proprietà.
   g. In questo editor, impostare gli attributi come segue:

Attributo classe	Tipo corrispondenza	Colonna tabella
attr1	Simple	STRINGFIELD2
attr2	Simple	INTEGERFIELD1

e fare clic su **OK**.

h. Dal menu Mappe datastore, selezionare Salva corrispondenza datastore. Quando si salva la corrispondenza, immettere le seguenti informazioni:

Attributo	Valore
Progetto	IBM WCS Enterprise Beans
Pacchetto	WCSContract EJB Reserved
Nome classe	WCSContractMap

Fare clic su **Fine** e chiudere il browser di corrispondenze.

3. Nel nuovo bean enterprise (ossia, in MySubTCBean) creare un nuovo metodo `ejbCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement)`, come riportato di seguito:

```
public void ejbCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
   java.rmi.RemoteException, javax.naming.NamingException,
   javax.ejb.RemoveException {
    _initLinks();
    super.ejbCreate (argTradingId, argElement);
    this.attr1= null;
    this.attr2= null;
}
```

4. Create a new `ejbPostCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement)` methos, as follows:

```
public void ejbPostCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
   java.rmi.RemoteException, javax.naming.NamingException,
   javax.ejb.RemoveException
   {
    parseXMLElement(argElement);
   }
```

5. Sovrascrivere il metodo `parseXMLElement(org.w3c.dom.Element argElement)` in MySubTCBean, come riportato di seguito:

```
public void parseXMLElement(org.w3c.dom.Element argElement) throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException,
   javax.ejb.RemoveException
   {
      super.parseXMLElement(argElement);

      if (argElement == null)
         return;
```

```
        String nodeName = argElement.getNodeName();
        if (nodeName.equals("TCCopy"))
           return;

        // get element "MyTC"
        Element eMyTC = ContractUtil.getElementByTag(argElement,"MyTC");

        // get element "MySubTC" from element "MyTC"
        Element eMySubTC = ContractUtil.getElementByTag(eMyTC ,"MySubTC");
        this.attr1 = eMySubTC .getAttribute("attr1").trim();
        this.attr2 = new Integer (eMySubTC .getAttribute("attr2").trim());

        // get element "PolicyReference" from "MySubTC"
        Element ePolicyReference = ContractUtil.getElementByTag(eMySubTC,
           "PolicyReference");
        parseElementPolicyReference(ePolicyReference);

    }
```

6. Sovrascrivere il metodo `createNewVersion(Long argNewTradingId)` in MySubTCBean, come riportato di seguito:

```
public Long createNewVersion(Long argNewTradingId) throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException,
   javax.ejb.RemoveException,
   org.xml.sax.SAXException,
   java.io.IOException

   {

   // Contract a seqElement since tcSequence can not be null
   Element seqElement = ContractUtil.getSeqElementFromTCSequence(
      this.tcSequence);
   MySubTCAccessBean newTC = new MySubTCAccessBean(argNewTradingId,
     seqElement);

   Long newTCId = newTC.getReferenceNumberInEJBType();
   newTC.setInitKey_referenceNumber(newTCId.toString());
   newTC.setMandatoryFlag(this.mandatoryFlag);
   newTC.setChangeableFlag(this.changeableFlag);
   // set columns for this specific TC
   newTC.setAttr1(this.attr1);
   newTC.setAttr2(this.attr2);
   newTC.commitCopyHelper();

   return newTCId;
}
```

7. Sovrascrivere il metodo `getXMLString()` in MySubTCBean, come riportato di seguito:

```
public String getXMLString() throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
```

```
   java.rmi.RemoteException,
   javax.naming.NamingException
   {
      String xmlTC = "      <MyTC>"  +
                     "%XML_POLICYREFERENCE%" +
                     "         <MySubTC attr1=\"" + this.attr1 +
                     "\"  attr2=\"" + this.attr2.toString() + "\"/>"
                     "         '>" +
                     "      <MYTC></MYTC>" ;
      String xmlPolicy = getXMLStringForElementPolicyReference(
         "ProductSet") ;
      xmlTC = ContractUtil.replace( xmlTC, "%XML_POLICYREFERENCE%",
         xmlPolicy );

      return xmlTC;
}
```

8. Sovrascrivere il metodo `markForDelete()` in MySubTCBean, come riportato di seguito:

```
public void markForDelete() throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException
   {
      // code: remove entries from associated tables which
      // cannot be deleted though delete cascade
}
```

9. Assicurarsi che il metodo `ejbCreate` sia stato aggiunto all'interfaccia principale e che tutti gli altri metodi modificati siano stati aggiunti all'interfaccia remota.
10. L'operazione successiva consiste nel creare un bean accesso per il bean entità MySubTC procedendo come segue:
    a. Fare clic con il pulsante destro del mouse sul bean entità **MySubTC** e selezionare **Aggiungi > Access Bean**.
       Si apre la SmartGuide di creazione del bean di accesso.
    b. Assicurarsi di aver immesso le seguenti informazioni:

Tabella 1.

Attributo	**Valore**
Gruppo EJB	`WCSContract`
Bean enterprise	`MySubTC`
Nome bean di accesso	`MySubTCAccessBean`
Tipo bean di accesso	`CopyHelper per un bean entità`

       e selezionare **Avanti**.
    c. Da **Seleziona metodo locale per constructor senza argomenti** elenco a discesa, selezionare **findByPrimaryKey(TermConditionKey)**

d. Per **initKey_referenceNumber** (nella colonna delle proprietà iniziali), impostare il **programma di conversione** su `com.ibm.commerce.base.objects.WCStringConverter` e fare clic su **Avanti**.

e. Per tutti i nuovo campo aggiunti, assicurarsi che **CopyHelper** venga selezionato e impostare il valore del programma di conversione per ciascun campo su `com.ibm.commerce.base.objects.WCStringConverter`. Fare clic su **Fine**. Quando la creazione del codice è completa, è possibile visualizzare il nuovo codice passando alla scheda **Progetti**, espandendo il progetto **IBM WCS Enterprise Beans** e poi i pacchetto **com.ibm.commerce.contract.objects**.

11. Ritornare alla scheda EJB, fare clic con il pulsante destro del mouse sul bean enterprise **MySubTC** e selezionare **Genera codice di distribuzione**.

12. È anche necessario rigenerare il codice di distribuzione per il bean parent (il bean TermCondition) e tutti i bean sibling (tutti gli altri bean nel gruppo di EJB del contratto WCS che contengono nel nome "TC"). Tener presente che se è stato aggiunto un nuovo campo oppure è stata modificare l'interfaccia remota del bean TermCondition esistente, è necessario rigenerare anche i bean di accesso per il bean in questione e per tutti i relativi bean child.
    Per rigenerare il codice di distribuzione, procedere come segue:

    a. Evidenziare il bean TermCondition e tutti gli altri bean il cui nome contiene "TC" (ad esempio, DisplayCustomizationTC, FulfillmentTC e InvoiceTC sono alcuni bean sibling).

    b. Con tutti questi bean evidenziati, fare clic con il pulsante destro del mouse e selezionare **Genera codice di distribuzione**.

13. Il passo successivo consiste nel sovrascrivere i metodi che si trovano nel comando di attività `ValidateContractCmd`. In questo comando, ci sono tre metodi da sovrascrivere per supportare il nuovo oggetto termini. Essi sono:

    - `validateTCType()`
      Questo metodo controlla il tipo di oggetto termini che può essere specificato in un contratto. Ad esempio, InvoiceTC appartiene all'account, quindi non può essere contenuto in un contratto.
    - `validateTCOccurrence()`
      Questo metodo controlla la ricorrenza dei termini. Ad esempio, nell'implementazione predefinita di questo metodo è necessario che un contratto presenti almeno un PriceTC.
    - `otherValidateCheck()`
      L'implementazione predefinita di questo metodo è vuota. È possibile aggiungere una ulteriore convalida che non interessi i primi due metodi.

14. Nel caso in cui sia necessario distribuire i termini, creare un nuovo comando di distribuzione e registrarlo nel database. Se necessario, effettuare le seguenti operazioni:
    a. In questo esempio, l'interfaccia del nuovo comando di distribuzione si chiama MySubTCDeployedCmd, mentre la classe di implementazione si chiama MySubTCDeployedCmdImpl. Inoltre, il comando si trova nel pacchetto packagename. Per registrare questo comando, inoltrare il seguente comando SQL:

       ```
       insert into CMDREG (STOREENT_ID, INTERFACENAME, CLASSNAME, TARGET)
       values (0, 'packagename.MySubTCDeployCmd',
       'packagename.MySubTCDeployCmdImpl', 'Local');
       ```

    b. Nel pacchetto packagename, creare la nuova interfaccia MySubTCDeployedCmd. Quest'ultima deve estendersi all'interfaccia del comando com.ibm.commerce.contract.commands.DeployTCCmd. Di seguito viene descritta l'interfaccia del nuovo comando:

       ```
       public interface MySubTCDeployCmd extends
          com.ibm.commerce.contract.commands.DeployTCCmd
          {
              // customized code
          }
       ```

       È presente un parametro protetto abTC e un metodo chiamato getTargetStoreId() in DeployTCCmd. Il valore di abTC è MySubTCAccessBean e il metodo getTargetStoreId() restituisce l'identificativo del negozio al quale viene distribuito il contratto.
    c. Nello stesso pacchetto, creare la classe di implementazione MySubTCDeployCmdImpl. Questa classe di implementazione deve estendersi acom.ibm.commerce.contract.commands.DeployTCCmdImpl. Di seguito viene descritta la nuova classe di implementazione:

       ```
       public class MySubTCDeployCmdImpl
          extends com.ibm.commerce.contract.commands.DeployTCCmdImpl
          implements MySubTCDeployCmd
          {
            // customer code
          }
       ```

Aggiornamento di WebSphere Commerce Accelerator per l'uso di nuovi termini e condizioni

Dopo aver creato il nuovo oggetto termini, è possibile aggiornare WebSphere Commerce Accelerator in modo da utilizzarlo per la creazione dei nuovi contratti che includano i questi nuovi termini. Per aggiornare WebSphere Commerce Accelerator, effettuare le seguenti operazioni:

1. Creazione di un nuovo file JavaScript per i nuovi termini. Nell'esempio utilizzato in questa sezione, questo file viene chiamato Extensions.js.

2. Creazione di una nuova maschera JSP che includa una sezione HTML in cui l'utente possa immettere le informazioni richieste per i nuovi termini. Nell'esempio utilizzato in questa sezione, questo file viene chiamato `ContractMyTC.jsp`.
3. Creazione di un nuovo bean di dati per i nuovi termini. Nell'esempio utilizzato in questa sezione, questo file viene chiamato `MyTCDataBean`.
4. Registrazione della nuova vista nella tabella VIEWREG.
5. Aggiornamento del file `ContractRB_`*`locale`*`.properties` per includere le nuove risorse.
6. Modifica del file `ContractNotebook.xml` per includere la nuova pagina.

Queste singole operazioni verranno descritte in maniera più dettagliata nelle sezioni successive.

Creazione del nuovo file JavaScript: La prima operazione da eseguire per aggiornare l'WebSphere Commerce Accelerator per utilizzare i nuovi termini, è la creazione di un nuovo file JavaScript. In questa sezione, è possibile utilizzare il seguente file di esempio:

- Windows `unità:\WebSphere\CommerceServer\samples\contract\Extensions.js`
- Windows `unità:\WebSphere\CommerceServerDev\samples\contract\Extensions.js`
- AIX `/usr/WebSphere/CommerceServer/samples/contract/Extensions.js`
- Solaris `/opt/WebSphere/CommerceServer/samples/contract/Extensions.js`
- Linux `/opt/WebSphere/CommerceServer/samples/contract/Extensions.js`
- 400 `/QIBM/ProdData/WebCommerce/samples/contract/Extensions.js`

Per utilizzare questo file di esempio, copiarlo nella seguente directory:

- Windows `unità:\WebSphere\AppServer\installedApps\WC_Enterprise_App_Nomeistanza.ear\wctools.war\javascript\tools\contract`
- AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_Nomeistanza.ear/wctools.war/javascript/tools/contract`
- Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_Nomeistanza.ear/wctools.war/javascript/tools/contract`

- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wctools.war/javascript/tools/contract
- 400 /QIBM/UserData/WebASAdv4/*NomeIstanzaAmmin_Was*/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wctools.war/javascript/tools/contract

In questo nuovo file, è necessario creare un oggetto JavaScript per memorizzare i dati per i nuovi termini. Questa operazione viene riportata nella seguente sezione di codice:

```
function ContractMyTCModel() {

   this.tcReferenceNumber = "";
   this.policyReferenceNumber = "";

   this.attr1 = "";
   this.attr2 = "";

   this.policyList = new Array();
   this.selectedPolicyIndex = "0";
}
```

Inoltre, è anche necessario creare un nuovo oggetto JavaScript per inoltrare i nuovi termini. Questa operazione deve essere coerente con le estensioni applicate al file B2BTrading.dtd, così come Questa operazione viene riportata nella seguente sezione di codice:

```
function submitMyTC(termsAndConditions) {

   var tcModel = get("ContractMyTCModel");

   if (tcModel != null) {

      var myTC = new Object();
      myTC.MyTC = new Object();
      myTC.MyTC.MySubTC = new Object();
      myTC.MyTC.MySubTC.attr1 = tcModel.attr1;
      myTC.MyTC.MySubTC.attr2 = tcModel.attr2;

      myTC.MyTC.PolicyReference = new Object();
      myTC.MyTC.PolicyReference.policyName =
         tcModel.policyList[tcModel.selectedPolicyIndex].policyName;
      myTC.MyTC.PolicyReference.policyType = "ProductSet";
      myTC.MyTC.PolicyReference.storeIdentity =
         tcModel.policyList[tcModel.selectedPolicyIndex].storeIdentity;
      myTC.MyTC.PolicyReference.Member =
         tcModel.policyList[tcModel.selectedPolicyIndex].member;

      if (tcModel.tcReferenceNumber != "") {
         // Change the term and condition
         myTC.action = "update";
```

```
            myTC.referenceNumber = tcModel.tcReferenceNumber;
        }
        else {
            // Create a new term and condition
            myTC.action = "new";
        }

        termsAndConditions[termsAndConditions.length] = myTC;
    }

    return true;
}
```

Creazione della nuova maschera JSP: Successivamente occorre creare una nuova maschera JSP che includa una sezione HTML in cui l'utente possa immettere le informazioni richieste dai nuovi termini. In questa sezione, è possibile utilizzare il seguente file di esempio:

- Windows *unità*:\WebSphere\CommerceServer\samples\contract\ContractMyTC.jsp
- Windows *unità*:\WebSphere\CommerceServerDev\samples\contract\ContractMyTC.jsp
- AIX /usr/WebSphere/CommerceServer/samples/contract/ContractMyTC.jsp
- Solaris /opt/WebSphere/CommerceServer/samples/contract/ContractMyTC.jsp
- Linux /opt/WebSphere/CommerceServer/samples/contract/ContractMyTC.jsp
- 400 /QIBM/ProdData/WebCommerce/samples/contract/ContractMyTC.jsp

Per utilizzare questo file di esempio, copiarlo nella seguente directory:

- Windows *unità*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*Nomeistanza*.ear\wctools.war\tools\contract
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wctools.war/tools/contract
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wctools.war/tools/contract
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wctools.war/tools/contract

- 400 /QIBM/UserData/WebASAdv4/*NomeIstanzaAmmin_Was*/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wctools.war/tools/contract

La sezione di codice riportata di seguito illustra un esempio di sezione HTML di una maschera JSP che può essere utilizzata per MyTC.

```
<!--
////////////////////////////////////
// HTML SECTION
////////////////////////////////////
-->

<BODY onLoad="onLoad()" class="content">

   <H1>
   <%= contractsRB.get("MyTCHeading") %>
   </H1>

<FORM NAME="MyTCForm">

   <%= contractsRB.get("MyTCAttr1Label") %>
   <BR>
   <INPUT type=text name=Attr1 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCAttr2Label") %>
   <BR>
   <INPUT type=text name=Attr2 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCPolicyLabel") %>
   <BR>
   <SELECT NAME="PolicyList" SIZE="1">
   </SELECT>

</FORM>
```

Creazione del nuovo bean di dati: A questo punto, occorre creare un nuovo bean di dati che carichi i dati necessari dal bean di accesso MySubTC. Di seguito vengono riportate le principali sezioni di codice relative alla creazione del nuovo bean di dati:

```
public class MyTCDataBean extends MySubTCAccessBean
   implements SmartDataBean, Delegator {
      private java.lang.Long contractId;
      private boolean hasMyTC = false;
      private CommandContext iCommandContext;


      /**
       * MyTCDataBean default constructor.
       */
      public MyTCDataBean() {
```

```
        }
        /**
         * MyTCDataBean constructor.
         */
        public MyTCDataBean(Long newContractId) {
         contractId = newContractId;
        }


        /*
         * populate the attributes from TermConditionAccessBean
         */
        public void populate() throws Exception {

              Enumeration myTCEnum = new TermConditionAccessBean().
                       findByTradingAndTCSubType(contractId, "MySubTC");
                    if (myTCEnum != null) {
                          // assume a contract only has one MyTC for this example

                          setEJBRef(((TermConditionAccessBean)
                                myTCEnum.nextElement()).getEJBRef());
                          refreshCopyHelper();
                          hasMyTC = true;
                    }

}
```

Registrazione della nuova vista nella tabella VIEWREG: La vista appena creata deve essere registrata nella tabella VIEWREG. Di seguito viene riportata un'istruzione SQL di esempio per la registrazione della nuova vista:

```
insert into VIEWREG(VIEWNAME,DEVICEFMT_ID,STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, HTTPS, INTERNAL)
values ('ContractMyTCPanelView', -1, 0,
   'com.ibm.commerce.tools.command.ToolsForwardViewCommand',
   'com.ibm.commerce.tools.command.ToolsForwardViewCommandImpl',
   'docname=tools/contract/ContractMyTC.jsp', 1, 1)
```

Aggiornamento del file`ContractRB_`*locale*`.properties` file: Il seguente file delle proprietà deve essere aggiornato con le informazioni specifiche dei nuovi termini:

- Windows *unità*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*Nomeistanza*.ear\properties\com\ibm\commerce\tools\contract\properties\ContractRB_*locale*.properties
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*locale*.properties

- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*locale*.properties
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*locale*.properties
- 400 /QIBM/UserData/WebASAdv4/*NomeIstanzaAmmin_Was*/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*locale*.properties

Di seguito viene riportato un esempio di informazioni da aggiungere al file:

```
MyTCHeading=My TC
attr1Empty=Attribute One must be entered.
attr2Empty=Attribute Two must be entered.
attr1TooLong=Attribute One is too long.
attr2TooLong=Attribute Two is too long.
MyTCAttr1Label=Attribute One (required)
MyTCAttr2Label=Attribute Two (required)
MyTCPolicyLabel=Policy
```

Modifica del file ContractNotebook.xml: Infine, per aggiungere i nuovi termini all'WebSphere Commerce Accelerator, è necessario aggiornare il seguente file per includere la nuova pagina:

- Windows *unità*:\WebSphere\CommerceServer\xml\tools\contract\ContractNotebook.xml
- AIX /usr/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- Solaris /opt/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- Linux /opt/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- 400 /QIBM/UserData/WebCommerce/instances/*Nomeistanza*/xml/tools/contract/ContractNotebook.xml

Di seguito viene riportata una sezione di codice utilizzata per includere la nuova pagina in questo esempio:

```
<panel name="MyTCHeading"
    url="ContractMyTCPanelView"
    parameters="contractId,accountId"
    helpKey="MC.contract.MyTCPanel.Help" />
```

Importazione del nuovo contratto utilizzando i nuovi termini

Come alternativa all'aggiornamento degli strumenti di WebSphere Commerce per utilizzare i nuovi termini, è possibile utilizzare il comando per l'importazione dei comandi (per informazioni su questo comando, consultare la guida in linea di WebSphere Commerce) per importare un nuovo contratto che includa questo nuovo oggetto termini. Dopo aver eseguito l'importazione, la principale sezione del file `Contract.xml` viene visualizzata come segue:

```
<TermCondition>
   <MyTC>
      <MySubTC attr1="adc" attr2="123" />
         <PolicyReference policyName = "Product Set 1"
                          policyType = "ProductSet"
                          storeIdentity = "StoreGroup1" >
            <Member>
               <User distinguishName = "uid=wcsadmin,o=Root Organization"/>
            </Member>
         </PolicyReference>
      </MyTC>
 </TermCondition>
```

Richiamo della nuova politica aziendale

Una volta creata una nuova politica aziendale, dopo averla associata ad almeno un oggetto termini e condizioni, è necessario aggiornare la logica dell'applicazione in modo che richiami i nuovi comandi di politica aziendale.

I comandi della politica aziendale vengono richiamati all'interno dei comandi di attività e dei comandi di controller.

Per richiamare i comandi della politica aziendale, viene utilizzato il comando factory. Esistono due metodi `create` che possono essere utilizzati per richiamare i comandi della politica aziendale. Il primo metodo viene utilizzato per richiamare un comando della politica aziendale quando soltanto un comando è associato alla politica. Tale valore viene illustrato nella seguente sezione del codice:

```
CommandFactory createBusinessPolicyCommand(Long policyId);
```

Il secondo metodo, invece, viene utilizzato per richiamare un comando della politica aziendale quando sono presenti più comandi associati alla politica. Tale valore viene illustrato nella seguente sezione del codice:

```
CommandFactory createBusinessPolicyCommand(Long policyId, String cmdIfName);
```

Nell'esempio precedente, `cmdIfName` viene utilizzato per specificare il nome dell'interfaccia del comando della politica aziendale che deve essere creato.

Il comando factory ricerca l'oggetto politica nella tabella POLICYCMD per determinare il comando che implementa tale politica. Inoltre, acquisisce qualsiasi proprietà predefinita e la imposta come requestProperties nel comando della politica aziendale.

La seguente sezione del codice mostra un esempio di richiamo di una politica di rimborso:

```
RefundPolicyCmd  cmd;

   ////////////////////////////////////////////////////////
   // Get the refund policy id from the refundTC object     //
   //  and use it to create the policy command.             //
   ////////////////////////////////////////////////////////
 cmd = (RefundPolicyCmd) CommandFactory
         createPolicyCommand (refundTC.getRefundPolicy);

  cmd.execute()
```

Creazione di un contratto

L'operazione successiva da eseguire per applicare l'estensione del modello di contratto al processo commerciale consiste nel creare un contratto che includa i termini e le condizioni che si riferiscono alla nuova politica aziendale. Un contratto può essere creato mediante WebSphere Commerce Accelerator oppure utilizzando uno dei comandi dell'URL dei contratti (ContractImportApprovedVersion e ContractImportDraftVersion). Per ulteriori informazioni sulla creazione dei contratti, fare riferimento alla guida in linea di WebSphere Commerce.

Scenari di personalizzazione dei contratti

Questa sezione fornisce una panoramica delle operazioni relative al seguente scenario di personalizzazione del contratto:

- Abilitazione di un ribasso

Scenari di ribassi

In questo scenario di esempio, viene creato un ribasso forfettario. Poiché il negozio di esempio ToolTech non include né termini e condizioni né un tipo di politica che corrisponda allo scenario dei ribassi, essi devono essere creati. Inoltre, è necessario creare una nuova politica aziendale oltre a una tabella di database per memorizzare i codici dei ribassi.

L'implementazione di questo scenario di ribassi prevede l'esecuzione delle seguenti fasi di livello elevato:

1. Creazione della tabella di database XREBATECODE e di un bean entità XRebateCodeBean corrispondente utilizzato per accedere alle informazioni da questa tabella.

2. Creazione di una nuova politica aziendale new 5DollarRebate eseguendo le seguenti attività secondarie:
   a. Creazione del nuovo tipo di politica aziendale. Ciò definisce l'interfaccia (RebatePolicyCmd) che sarà implementata dal nuovo comando di politica aziendale.
   b. Creazione del nuovo comando di politica aziendale CalculateRebateCmdImpl.
   c. Registrazione del nuovo comando di politica aziendale e del nuovo tipo di politica aziendale nel database.
3. Creazione di un nuovo oggetto termini e condizioni (RebateTC) per il ribasso effettuando le seguenti attività secondarie:
   a. Registrazione del nuovo oggetto termini e condizioni RebateTC nel database
   b. Aggiornamento del file `B2BTrading.dtd` in modo da includere il nuovo oggetto RebateTC.
   c. Creazione di un nuovo bean enterprise per RebateTC.
   d. Aggiornamento di WebSphere Commerce Accelerator per riflettere il nuovo oggetto RebateTC.
4. Creazione di un nuovo contratto che utilizza RebateTC.
5. Integrazione della nuova politica aziendale nel flusso di acquisto.

Ognuna di queste operazioni viene descritta in dettaglio delle successive sezioni.

Passo 1: Creazione della nuova tabelle e del bean enterprise

Poiché lo schema di database esistente non include la specifica del codice e dell'importo del ribasso, è necessario creare una nuova tabella. In generale, quando viene creata una nuova tabella, viene creato anche un nuovo bean identità utilizzato nell'accesso alle informazioni contenuto in questa tabella.

In questo esempio, si supponga che venga creata la seguente tabella di database XREBATECODE.

Tabella 2. tabella di database XREBATECODE

	Nome colonna		
	REBATECODE_ID	AMOUNT	CURRENCY
Dati di esempio	201	5	CAD
	202	10	CAD

Inoltre, viene creato un nuovo bean entità CMP (XRebateCodeBean). Per informazioni dettagliate sulla creazione di questo bean, fare riferimento a "Creazione di un nuovo bean enterprise CMP" a pagina 62.

Passo 2: Creazione della politica aziendale “5DollarRebate”

Per creare questa nuova politica aziendale, procedere come segue:

1. Creare l’interfaccia del nuovo tipo di politica aziendale. Si tratta dell’interfaccia `RebatePolicyCmd` che sarà implementata da `CalculateRebateCmdImpl`
2. Creare il nuovo comando di politica aziendale `CalculateRebateCmdImpl`.
3. Registrare la nuova politica aziendale e il relativo comando nel database.

Creazione del tipo di politica aziendale “Rebate”: Poiché non esiste un tipo di politica aziendale che corrisponde ai ribassi, è necessario crearne una nuova. La creazione di un nuovo tipo di politica aziendale comporta la definizione e la registrazione di un tipo di politica nel database. È necessario aggiornare le seguenti tabelle:

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

Per questo scenario, per creare il nuovo tipo di politica REBATE, occorre utilizzare le seguenti istruzioni SQL:

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('Rebate');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('Rebate',
   'com.mycompany.mybusinesspolicycommands.RebatePolicyCmd');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('Rebate', -1,
   'Rebate policy type.');
```

Di conseguenza, la seguente tabella mostra le colonne rilevanti della tabella PLCYTYCMIF che mostra la relazione tra il tipo di politica e il comando di politica aziendale al quale è correlato.

Tabella 3. Aggiornamenti effettuati nella tabella PLCYTYCMIF

	Nome colonna	
	POLICYTYPE_ID	BUSINESSCMDIF
Dati di esempio	Ribasso	com.mycompany.mybusinesspolicycommands.RebatePolicyCmd

È anche necessario codificare la nuova interfaccia `RebatePolicyCmd`. Questa interfaccia deve estendersi dall’interfaccia `com.ibm.commerce.command.BusinessPolicyCommand`. Come suggerito dalla tabella precedente, inserire questa interfaccia nel proprio pacchetto.

Creazione del comando di politica aziendale CalculateRebateCmdImpl: Per creare il nuovo comando di politica aziendale, è necessario creare un nuovo

comando chiamato CalculateRebateCmdImpl che estende la classe di implementazione com.ibm.commerce.command.BusinessPolicyCommandImpl. Questo comando deve implementare l'interfaccia RebatePolicyCmd creata al passo precedente.

Notare che in questo esempio, il nome interfaccia e in nome comando sono diversi. Questi nomi sono stati scelti con l'intenzione di mostrare che possono esistere comandi di politica aziendale che implementano il tipo di ribasso della politica aziendale. Ciascuna implementazione (ovvero ciascun comando di politica aziendale) implementa poi il ribasso in maniera univoca.

La logica del comando dipende dalla particolare implementazione di come il cliente deve selezionare la merce. Inoltre, questo comando CalculateRebateCmdImpl deve essere richiamato da un controller separato o da un comando di attività dell'applicazione.

Registrazione della nuova politica aziendale e del nuovo comando: La nuova politica aziendale deve essere registrata nel database. È anche necessario registrare la relazione tra la nuova politica aziendale e il nuovo comando.

Per registrare queste informazioni, è possibile utilizzare il comando com.ibm.commerce.contract.commands.PolicyAddCmd. Di seguito viene riportato un esempio dell'uso del comando PolicyAdd per questo scenario:

```
http://localhost:8080/webapp/wcs/stores/servlet/PolicyAdd?
   type=Rebate&name=5DollarRebate&plcyStoreId=-1
   &cmd_1=com.mycompany.mybusinesspolicycommands.CalculateRebateCmdImpl
   &startDate=2002-05-08%2000:00:00&endDate=2003-05-09%2000:00:00
   &commonProps=rebatecode_id%3D501&URL=aRedirectURL
```

Tener presente che i caratteri riservati dell'URL devono essere sostituiti dai relativi codici ASCII per le proprietà di immissione. Pertanto, il tipico simbolo = (uguale) viene sostituito da "%3D", il carattere & (E commerciale) da "%26" e lo spazio da "%20".Il formato data utilizzato nell'esempio precedente è aaaa-mm-gg hh:mm:ss, con il codice ASCII che sostituisce i caratteri riservati dell'URL.

Le seguenti tabelle mostrano le colonne rilevanti delle tabelle di database interessate dopo l'esecuzione degli aggiornamenti.

Tabella 4. Aggiornamenti effettuati nella tabella POLICY

	Nome colonna				
	POLICY _ID	POLICY NAME	POLICYTYPE _ID	STOREENT _ID	PROPERTIES
Dati di esempio	301	5Dollar Rebate	Rebate	-1	rebatecode_id= 201

Tener presente che si presume che i valori relativi a data di inizio e data di fine siano impostati su `null`.

Tabella 5. Aggiornamenti effettuati nella tabella POLICYCMD

	Nome colonna		
	POLICY _ID	BUSINESS CMDCLASS	PROPERTIES
Dati di esempio	301	com.mycompany. mybusinesspolicycommands. CalculateRebateCmdImpl	null

Il risultato è la presenza di una nuova politica aziendale denominata "5DollarRebate" collegata al comando CalculateRebateCmd.

Passo 3: Creazione di termini e condizioni "RebateTC"

Per creare termini e condizioni "RebateTC", effettuare le seguenti operazioni:

1. Registrazione del nuovo oggetto termini e condizioni RebateTC nel database
2. Aggiornamento del file B2BTrading.dtd in modo da includere il nuovo oggetto RebateTC.
3. Creazione di un nuovo bean enterprise per RebateTC.
4. Aggiornamento di WebSphere Commerce Accelerator per riflettere il nuovo oggetto RebateTC.

Registrazione di termini e condizioni "RebateTC" nel database: Quando si crea un nuovo oggetto termini, è necessario aggiornare lo schema del database per includere questo schema. In particolare, occorre aggiornare le tabelle TCTYPE e TCSUBTYPE.

La seguente istruzione SQL illustra un esempio di registrazione di RebateTC nel database:

```
insert into TCTYPE (TCTYPE_ID) values ('RebateTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME,
   DEPLOYCOMMAND)
values ('RebateTC, 'RebateTC ',
   'com.ibm.commerce.contract.objects.RebateTCAccessBean',
   null);
```

Le seguenti tabelle mostrano un estratto delle colonne rilevanti delle tabelle TCTYPE e TCSUBTYPE.

Tabella 6. Aggiornamenti effettuati nella tabella TCTYPE

	Nome colonna
	TCTYPE_ID

Tabella 6. Aggiornamenti effettuati nella tabella TCTYPE (Continua)

Dati di esempio	RebateTC

Tabella 7. Aggiornamenti effettuati nella tabella TCSUBTYPE

	Nome colonna			
	TCSUBTYPE _ID	TCTYPE _ID	ACCESSBEAN NAME	DEPLOY COMMAND
Dati di esempio	RebateTC	RebateTC	com.ibm.commerce. contract.objects. RebateTCAccessBean	null

Aggiornamento del file B2BTrading.dtd per riportare il nuovo RebateTC: Per rendere disponibili i nuovi termini e condizioni nei contratti, è necessario aggiornare il file `B2BTrading.dtd` in modo da includere i nuovi termini e condizioni. Quando si aggiorna questo file, è necessario aggiungere i nuovi termini e condizioni alla definizione TermCondition, quindi creare un nuovo elemento che li descriva.

Il testo in grassetto mostra un esempio di come aggiungere il nuovo RebateTC alla definizione TermCondition:

```
<!ELEMENT TermCondition (TermConditionDescription?,Participant*,
CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
OrderTC|RebateTC))>
```

Le interruzioni di riga sono state inserite unicamente per agevolare la visualizzazione.

A questo punto è necessario aggiungere una nuova stanza che descriva questi termini e condizioni nel file. Viene di seguito riportato un esempio per RebateTC:

```
<!ELEMENT RebateTC (PolicyReference?)>
```

Creazione di un nuovo bean entità per RebateTC: È necessario creare un nuovo bean enterprise per il nuovo RebateTC. Questo nuovo bean deve ereditare dal bean di WebSphere Commerce TermCondition.

Un nuovo bean enterprise per termini e condizioni in genere prende il nome dal sottotipo. In questo caso, il sottotipo di termini e condizioni corrisponde al tipo di termini e condizioni, pertanto il bean prende il nome dal tipo.

La tabella di seguito riportata mostra alcune informazioni generali sul nuovo bean da creare. Per ulteriori dettagli sul bean, incluso i metodi da

sovrascrivere, fare riferimento a “Creazione di un nuovo bean enterprise CMP relativo ai termini” a pagina 167.

Tabella 8.

Attributo	Valore
Nome bean	`RebateTC`
Eredita da	`TermCondition`
Pacchetto	`com.ibm.commerce.contract.objects`
Classe bean	`RebateTCBean`
Interfaccia remota	`RebateTC`
Interfaccia home	`RebateTCHome`

Aggiornamento di WebSphere Commerce Accelerator per l’inclusione di RebateTC: Dopo aver creato il nuovo oggetto termini, è possibile aggiornare WebSphere Commerce Accelerator in modo da utilizzarlo per la creazione dei nuovi contratti che includano i questi nuovi termini. Per informazioni sull’aggiornamento di questo strumento fare riferimento a “Aggiornamento di WebSphere Commerce Accelerator per l’uso di nuovi termini e condizioni” a pagina 173.

Passo 4: Creazione di un nuovo contratto

È necessario creare un nuovo contratto che includa termini e condizioni “RebateTC” e faccia riferimento alla politica aziendale “5DollarRebate”. Per creare un nuovo contratto, è possibile utilizzare WebSphere Commerce Accelerator o XML. Tali metodi sono descritti nella guida in linea di WebSphere Commerce.

Le tabelle di seguito riportate mostrano gli aggiornamenti apportati alle colonne rilevanti delle tabelle di database TERMCOND e POLICYTC, dopo la creazione del contratto.

Tabella 9. Aggiornamenti effettuati alla tabella TERMCOND

	Nome colonna		
	TRADING _ID	TERMCOND _ID	TCSUBTYPE _ID
Dati di esempio	25	901	RebateTC

Tabella 10. Aggiornamenti effettuati nella tabella POLICYTC

	Nome colonna	
	POLICY _ID	TERMCOND_ID

Tabella 10. Aggiornamenti effettuati nella tabella POLICYTC (Continua)

Dati di esempio	301	901

Passo 5: Integrazione della nuova politica aziendale nel flusso di acquisto

In questo scenario, si presume che al negozio venga aggiunta una nuova pagina che consente a tutti i clienti di registrare e dichiarare i ribassi. Quando il cliente fa clic per dichiarare un ribasso, richiama un comando che richiama a sua volta la nuova interfaccia `RebatePolicyCmd`. Ad esempio, un nuovo comando di controller ClaimRebateCmd che richiama l'interfaccia `RebatePolicyCmd`. La politica aziendale corretta viene rilevata e (in questo caso) viene applicata la politica aziendale "5DollarRebate".

Parte 3. Ambiente di sviluppo

Capitolo 8. Strumenti di sviluppo e distribuzione

Questo capitolo introduce gli strumenti di sviluppo principali utilizzati per la personalizzazione di un'applicazione di WebSphere Commerce. Descrive il processo di distribuzione del codice personalizzato da VisualAge per Java a un WebSphere Commerce Server. Descrive, inoltre, le modalità di distribuzione del codice in Commerce Studio, in modo da poter usufruire del codice personalizzato durante la distribuzione della maschere JSP.

Ambiente di sviluppo

Il pacchetto di sviluppo consigliato per la creazione di codice personalizzato da utilizzare con WebSphere Commerce Business Edition è WebSphere Commerce Studio, Business Developer Edition. Il pacchetto di sviluppo consigliato per la creazione di codice personalizzato da utilizzare con WebSphere Commerce Professional Edition è WebSphere Commerce Studio Professional Developer Edition. Entrambi questi pacchetti comprendono tutti gli strumenti necessari per creare codice personalizzato e per eseguire attività di sviluppo del Web.

Sia WebSphere Commerce Studio Business Developer Edition che WebSphere Commerce Studio Professional Developer Edition forniscono la possibilità di includere un negozio WebSphere Commerce di esempio che viene utilizzato nel componente WebSphere Test Environment di VisualAge per Java. Ciò semplifica la configurazione dell'ambiente di sviluppo, in quanto non è detto che uno sviluppatore utilizzi gli strumenti di WebSphere Commerce per creare un negozio e che sposti le risorse del negozio nell'ambiente di sviluppo.

Un altro componente chiave per l'ambiente di sviluppo è il magazzino del codice di WebSphere Commerce. Tale magazzino deve essere importato nello spazio di lavoro di VisualAge per Java in seguito al completamento dell'installazione del prodotto. Dopo aver importato tale magazzino, lo sviluppatore deve effettuare alcuni passi della configurazione relativi a WebSphere Test Environment.

Una volta completate le operazioni di installazione e di configurazione, lo sviluppatore dispone di una macchina di sviluppo autonoma, sulla quale è possibile creare e testare il codice personalizzato di WebSphere Commerce. Allo sviluppatore non viene richiesto di installareWebSphere Commerce sulla macchina di sviluppo.

WebSphere Commerce Studio

Sia WebSphere Commerce Studio Business Developer Edition che WebSphere Commerce Studio Professional Developer Edition includono VisualAge per Java, Enterprise Edition, Versione 4.0. Questa edizione di VisualAge per Java include una serie di strumenti per lo sviluppo e il debug delle maschere JSP avanzate che agevolano lo sviluppo delle attività di fronte negozio. Inoltre, l'integrazione avanzata con WebSphere Studio consente di aggiungere rapidamente il contenuto a queste maschere JSP, incrementando il livello di produttività per programmatori e sviluppatori Web. Per lo sviluppo del codice per applicazioni back-office, fornisce supporto per la tecnologia Enterprise JavaBeans e funzioni di connettività per supportare l'integrazione con altri sistemi, come CICS TS, MQSeries, SAP R/3 e altri ancora. Inoltre, il WebSphere Test Environment integrato di VisualAge per Java, Enterprise Edition, consente agli sviluppatori di eseguire le funzioni di WebSphere Commerce senza mai uscire da VisualAge per Java. Ciò vuol dire che è possibile eseguire il test del codice personalizzato di WebSphere Commerce senza dover distribuire il codice a un WebSphere Commerce Server.

Nota: Il componente WebSphere Test Environment di VisualAge per Java, Enterprise Edition, Versione 4.0 esegue le applicazioni all'interno di WebSphere Application Server V3.5.4. Si noti che sia WebSphere Commerce Business Edition che WebSphere Commerce Professional Edition utilizzano WebSphere Application Server V4.0. Il risultato di questa differenza di versioni consiste nel generare il codice di distribuzione all'esterno del VisualAge per Java utilizzando lo strumento EJBDeploy fornito con il WebSphere Application Server V4.0 prima di distribuire bean enterprise nuovi o modificati in un'istanza WebSphere Commerce in esecuzione su WebSphere Application Server V4.0. Infatti, tale codice di distribuzione deve essere generato su una macchina con il WebSphere Application Server V4.0 installato. Per ulteriori dettagli, consultare "Informazioni sul codice di distribuzione EJB" a pagina 194.

Per completare le attività descritte in Parte 4, "Supporti didattici" a pagina 205, è necessario installare WebSphere Commerce Studio Business Developer Edition oppure WebSphere Commerce Studio Professional Developer Edition. Per ulteriori informazioni sull'installazione di Commerce Studio e sulla configurazione di VisualAge per Java, fare riferimento al manuale *WebSphere Commerce Studio Business Developer Edition Guida all'installazione* o *WebSphere Commerce Studio Professional Developer Edition Guida all'installazione*.

Caratteristiche e funzioni di VisualAge per Java

Questa *Guida per il programmatore* non è stata concepita per illustrare le modalità di impiego di VisualAge per Java. In riferimento a questo prodotto, questa Guida illustra come eseguire un'attività in VisualAge per Java più come un modo per completare un'attività di programmazione per la creazione di un'applicazione e-commerce per WebSphere Commerce. Pertanto, i nuovi utenti di VisualAge per Java potranno apprendere le modalità di esecuzione di alcune attività in in VisualAge per Java eseguendo i supporti didattici contenuti in questo libro. Tuttavia, per acquisire maggiore dimestichezza nell'utilizzo di questo strumento, è necessario fare riferimento alla documentazione di VisualAge per Java, ai supporti didattici e ai corsi.

Ad esempio, l'argomento del supporto didattico (Facoltativo) Utilizzo del programma di debug in VisualAge per Java fornisce una rapida introduzione alle modalità di utilizzo del programma di debug per visualizzare il valore delle variabili di un comando di attività durante l'esecuzione del codice. Il componente Debugger di VisualAge per Java è uno strumento di grande efficacia e per ulteriori dettagli sulle funzioni fornite, consultare la documentazione relativa a VisualAge per Java.

Magazzino di codice WebSphere Commerce

Per creare il codice personalizzato per un'applicazione WebSphere Commerce, è necessario importare un magazzino del codice di WebSphere Commerce nello spazio di lavoro di VisualAge per Java. Il magazzino è disponibile sul CD di WebSphere Commerce Business Edition Disco 2 e sul CD di WebSphere Commerce Professional Edition Disco 2. Il magazzino corrente (al momento della pubblicazione di questa documentazione) si chiama `WC_54.dat`.

Sviluppo del codice

Durante la personalizzazione della propria applicazione e-commerce è possibile eseguire le seguenti operazioni:

- Creazione di nuovi comandi, di bean di dati e di bean entità
- Estensione di bean entità di WebSphere Commerce esistenti
- Modifica della logica dei comandi esistenti o dei bean di dati

Durante lo sviluppo del codice in VisualAge per Java, è possibile eseguire il test del codice in WebSphere Test Environment. A questo stesso punto, è necessario distribuire il codice a un WebSphere Commerce Server esterno all'ambiente di sviluppo.

Nelle sezioni successive, *WebSphere Commerce Server di destinazione* fa riferimento a WebSphere Commerce Server per il quale si utilizza il codice personalizzato. In alcuni scenari di verifica, è possibile impiegarlo su

WebSphere Commerce Server che è in esecuzione sulla stessa macchina di VisualAge per Java. In altre situazioni, WebSphere Commerce Server di destinazione è ubicato su un'altra macchina e può essere persino eseguito su una piattaforma diversa.

Le seguenti sezioni descrivono le procedure avanzate per l'impiego di vari tipi di codice personalizzato. È consigliabile utilizzare queste procedure per comprendere quelle descritte nel processo di distribuzione e per istruzioni dettagliate, fare riferimento a Appendice B, "Dettagli sulla configurazione" a pagina 349.

In aggiunta alle seguenti sezioni che descrivono la distribuzione del codice personalizzato, se nell'ambiente di distribuzione sono state create nuove politiche di controllo accessi, è necessario creare queste politiche sul WebSphere Commerce Server di destinazione. Per un esempio di questa procedura, fare riferimento a "Caricamento delle politiche di controllo accesso per le nuove risorse" a pagina 283 nei supporti didattici.

Informazioni sul codice di distribuzione EJB

È importante sapere che il componente WebSphere Test Environment di VisualAge per Java V4.0 utilizza il WebSphere Application Server V3.5.4. In questa versione di WebSphere Application Server, i bean enterprise sono supportati al livello della specifica Enterprise JavaBeans (EJB) V1.0. Pertanto, per eseguire i bean enterprise all'interno di WebSphere Test Environment, utilizzare gli strumenti di VisualAge per Java per generare il codice di distribuzione dei bean enterprise che sia conforme alla specifica EJB V1.0.

Per contro, sia WebSphere Commerce Business Edition che WebSphere Commerce Professional Edition utilizzano WebSphere Application Server V4.0. Il WebSphere Application Server V4.0 supporta la specifica EJB V1.1. Pertanto, il codice di distribuzione per i bean enterprise in esecuzione all'interno di WebSphere Test Environment è diverso dal codice di distribuzione per i bean enterprise in esecuzione all'interno di WebSphere Application Server V4.0.

Di seguito è riportato l'esito sullo sviluppo e sulla distribuzione:

- Per controllare i bean enterprise all'interno di WebSphere Test Environment, utilizzare gli strumenti di VisualAge per Java per generare il codice di distribuzione che corrisponda ai requisiti del WebSphere Application Server V3.5.4 utilizzato in WebSphere Test Environment. Per generare questo codice, fare clic con il pulsante destro del mouse sul bean enterprise e selezionare **Genera codice di distribuzione**. Tale operazione non crea un file JAR.
- Per distribuire i bean enterprise su un WebSphere Application Server V4.0, è necessario procedere come segue:

1. Utilizzare gli strumenti di VisualAge per Java per esportare i bean enterprise in un *file JAR EJB 1.1 di esportazione*, cosi' come viene definito in questo documento. Tale file JAR crea i pacchetti del codice in un formato utilizzato dallo strumento EJBDeploy. Per creare questo file, fare clic con il pulsante destro del mouse sul gruppo EJB che contiene il bean enterprise da distribuire e selezionare **Export > EJB 1.1 JAR**. Si noti che durante la creazione di questo file JAR, è necessario selezionare il tipo di database appropriato per la macchina sulla quale si desidera distribuire il codice.
2. Trasferire il file JAR EJB 1.1 di esportazione su un server di destinazione che esegue il WebSphere Application Server V4.0.
3. Sul server di destinazione, eseguire lo strumento EJBDeploy utilizzando il file JAR EJB 1.1 di esportazione come file di immissione per creare un nuovo file JAR del codice di distribuzione che sia conforme alla specifica Enterprise JavaBeans V1.1.

Esistono altre operazioni necessarie per la distribuzione dei bean enterprise. Per ulteriori informazioni consultare "Distribuzione di nuovi bean entità" a pagina 196 e "Distribuzione dei bean entità pubblici modificati di WebSphere Commerce" a pagina 199. Per ulteriori informazioni sull'utilizzo dello strumento EJBDeploy, fare riferimento a "Generazione del codice di distribuzione" a pagina 360.

Impiego di nuovi comandi e bean di dati

Quando si creano nuovi comandi e bean di dati, inserirli in un pacchetto con un nome appropriato alla propria applicazione. Ad esempio, è possibile creare un nuovo pacchetto e denominarlo `com.mycompany.mycommands` nel quale inserire i nuovi comandi. Questo pacchetto deve essere memorizzato in un progetto separato dai progetti di WebSphere Commerce (IBM WC Commerce Server e IBM WC Enterprise Beans). Ad esempio, creare un progetto e denominarlo `Nuovo Progetto`. Per garantire una corretta distribuzione del codice, è necessario organizzare il codice con attenzione.

Una volta memorizzati i nuovi comandi e i bean di dati nel nuovo progetto, per eseguire la distribuzione completare le seguenti fasi:

1. Creare un file JAR per il progetto utilizzando la macchina di sviluppo. Dalla pagina dei progetti di VisualAge per Java, utilizzare gli strumenti per esportare il progetto in un file JAR. Assegnare al file JAR un nome appropriato al codice, ad esempio, `CustomCommands.jar`. Inoltre, è necessario rejar il file JAR utilizzando strumenti esterni a VisualAge per Java per assicurare che tutte le informazioni di denominazione necessarie vengano incluse per la distribuzione in WebSphere Application Server. Per ulteriori informazioni, fare riferimento a "File JAR per comandi personalizzati e bean di dati" a pagina 349.

2. Copiare il file JAR, le maschere JSP e altre risorse del negozio nella directory appropriata di WebSphere Commerce Server di destinazione. Per ulteriori informazioni, fare riferimento a "Memorizzazione delle risorse su WebSphere Commerce Server di destinazione" a pagina 356.
3. Registrare il nuovo comando nel registro comandi sul WebSphere Commerce Server di destinazione. Per ulteriori informazioni, fare riferimento a "Struttura del registro comandi" a pagina 28.
4. Arrestare e riavviare l'enterprise application di WebSphere Commerce, utilizzando la console di gestione di WebSphere Application Server. Per ulteriori informazioni sull'avvio e sull'arresto di questa applicazione, fare riferimento al manuale *WebSphere Commerce Guida all'installazione*.

Distribuzione di nuovi bean entità

Quando si creano nuovi bean entità, occorre crearli in un gruppo EJB separato dai gruppi EJB che contengono i bean entità di WebSphere Commerce. Inoltre, è necessario utilizzare il proprio progetto e accertarsi che tale progetto non contenga il codice per comandi o bean di dati. Ad esempio, creare un nuovo gruppo EJB denominato `MyEntityBeans` e un progetto denominato `MyEntityBeansProject`. Se il progetto contiene un codice diverso da quello utilizzato per i nuovi bean entità, è possibile che la distribuzione non venga eseguita correttamente.

Quando il funzionamento del bean entità in WebSphere Test Environment è soddisfacente, eseguirne la distribuzione. Le seguenti informazioni forniscono una panoramica delle fasi di distribuzione.

1. Utilizzando la macchina di sviluppo, creare un nuovo file JAR EJB 1.1 di esportazione per il nuovo gruppo EJB. Dalla pagina EJB di VisualAge per Java, fare clic con il pulsante destro del mouse e selezionare il nuovo gruppo di EJB per esportare il codice in un file JAR di EJB 1.1. Assegnare al file un nome appropriato per il codice, ad esempio `MyEntityBeans_DT.jar`. Per ulteriori informazioni, fare riferimento a "Creazione di file JAR per nuovi bean entità" a pagina 351.
2. Creare un nuovo file JAR di implementazione per il nuovo progetto EJB. Per creare questo file JAR, selezionare la pagina Progetto, quindi fare clic con il pulsante destro del mouse sul nuovo progetto EJB e selezionarlo per esportare il codice in un file JAR. Assegnare al file un nome appropriato per il codice, ad esempio `MyEntityBeansImpl.jar`. Inoltre, è necessario rejar il file JAR di implementazione utilizzando strumenti esterni a VisualAge per Java per assicurare che tutte le informazioni di denominazione necessarie vengano incluse per la distribuzione in WebSphere Application Server. Per ulteriori informazioni, fare riferimento a "Creazione di file JAR per nuovi bean entità" a pagina 351.

3. Copiare i file JAR nella directory appropriata della macchina WebSphere Commerce Server di destinazione. Per ulteriori informazioni, fare riferimento a "Memorizzazione delle risorse su WebSphere Commerce Server di destinazione" a pagina 356.
4. Sul WebSphere Commerce Server di destinazione, utilizzare lo strumento per la distribuzione di EJB fornito con WebSphere Application Server per generare il codice di distribuzione per i nuovi bean enterprise. Questo strumento utilizza il file JAR creato al passo 1 come file di immissione creando il corrispondente file JAR contenente il codice di distribuzione per tutti i bean enterprise nel gruppo EJB. Per ulteriori informazioni, fare riferimento a "Generazione del codice di distribuzione" a pagina 360.
5. Sul WebSphere Commerce Server di destinazione, modificare il livello di isolamento transazione dei bean enterprise contenuti nel file JAR del codice di distribuzione. Utilizzare l'utility della riga comandi `modifyIsolationLevel` fornita con WebSphere Commerce per impostare in modo appropriato il livello di isolamento delle transazioni per il proprio tipo di database. Per ulteriori informazioni, fare riferimento a "Modifica del livello di isolamento delle transazioni dei bean entità" a pagina 363.
6. Sul WebSphere Commerce Server di destinazione, caricare le politiche di controllo accessi per una qualsiasi nuova risorsa creata. Utilizzare i comandi `acpload` e `acpnlsload` di WebSphere Commerce per caricare le informazioni sulle politiche. Consultare la sezione "Caricamento delle politiche di controllo accesso per le nuove risorse" a pagina 283 del supporto didattico Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale" per un esempio di caricamento delle politiche di controllo degli accessi per nuove risorse.
7. Sul WebSphere Commerce Server di destinazione, esportare l'enterprise application corrente di WebSphere Commerce da WebSphere Application Server. Dopo il completamento dell'esportazione, viene creato un file .ear che contiene l'intera applicazione. Per esportare l'applicazione corrente, aprire la WebSphere Application Server Console di gestione, selezionare l'application enterprise di WebSphere Commerce, quindi l'opzione di esportazione. Al termine, viene creato un file `WC_Enterprise_App_`*`Nomeistanza`*`.ear`. Per ulteriori informazioni, fare riferimento a "Esportazione dell'enterprise application di WebSphere Commerce corrente" a pagina 364.
8. Sul WebSphere Commerce Server di destinazione, esportare le informazioni sulla configurazione per i bean enterprise contenuti nell'enterprise application corrente di WebSphere Commerce in esecuzione su WebSphere Application Server. Queste informazioni vengono esportate in un file XML mediante l'opzione `-export` della funzione di riga comandi `XMLConfig` che viene fornita dal WebSphere Application Server. Il file XML generato (riferito qui come file `OutputFile.xml`) contiene una stanza di informazioni sulla configurazione per ciascun bean enterprise contenuto nell'enterprise application. È quindi necessario aggiungere una nuova

stanza a questo file per ciascun bean enterprise nuovo che si sta distribuendo. Per ulteriori informazioni, fare riferimento a "Esportazione delle informazioni sulla configurazione per i bean enterprise" a pagina 366.

9. Aggiungere i nuovi bean enterprise all'enterprise application utilizzando lo Strumento di assemblaggio delle applicazioni fornito con WebSphere Application Server. Utilizzando questo strumento è possibile aprire il file `WC_Enterprise_App_`*`Nomeistanza`*`.ear` per l'applicazione corrente e quindi importare i nuovi bean enterprise nell'applicazione. Inoltre, viene impostata la class path per i nuovi bean enterprise in modo da includere eventuali file JAR dipendenti ed aggiungere il file JAR di implementazione come file all'applicazione. Infine, questo strumento viene utilizzato per configurare la sicurezza di WebSphere Application Server per i metodi contenuti nei propri bean enterprise. Dopo aver completato questi passi, salvare l'applicazione creando un nuovo file `.ear` per il proprio enterprise application. Per ulteriori informazioni, fare riferimento a "Assemblaggio dei nuovi bean enterprise nell'enterprise application" a pagina 371.
10. Importare la nuova enterprise application in WebSphere Application Server. Questa operazione è costituita dalle tre seguenti attività secondarie:
    a. Utilizzando il WebSphere Application Server Console di gestione, arrestare e rimuovere l'enterprise application originale di WebSphere Commerce. Per ulteriori informazioni, fare riferimento a "Arresto e rimozione dell'enterprise application" a pagina 381.
    b. Utilizzando l'opzione `-import` dell'utility della riga comandi `XMLConfig` per importare la nuova enterprise application in WebSphere Application Server. Per ulteriori informazioni, fare riferimento a "Importazione dell'enterprise application" a pagina 382.
    c. Utilizzando il WebSphere Application Server Console di gestione, aggiornare le informazioni visualizzate in modo che appaia la nuova enterprise application, quindi avviare la nuova applicazione. Per ulteriori informazioni, fare riferimento a "Avvio dell'enterprise application" a pagina 383.

Impiego di estensioni per i comandi e i bean di dati esistenti

Il metodo per l'estensione dei comandi esistenti dipende dal tipo di modifica richiesta. I metodi di estensione sono esposti in "Personalizzazione di comandi esistenti" a pagina 145. In generale, la modifica della logica esistente prevede la creazione di una nuova classe che eredita dalla classe che richiede la personalizzazione. Sovrascrivere i metodi della superclasse come richiesto per sostituire o modificare la logica.

Quando si personalizzano i bean di dati, si crea anche una nuova classe che estende un bean di dati esistente. Nella nuova classe, apportare le modifiche necessarie.

Quando si creano queste nuove classi, assicurarsi che siano memorizzate in uno dei propri pacchetti, anch'esso memorizzato in uno dei progetti creati.

Poiché le estensioni sono efficacemente gestite dalle sottoclassi, la distribuzione delle estensioni ai comandi e ai bean di dati è la stessa di quella per i nuovi comandi e bean di dati. Per ulteriori informazioni, consultare "Impiego di nuovi comandi e bean di dati" a pagina 195.

Distribuzione dei bean entità pubblici modificati di WebSphere Commerce

Quando si modifica un bean enterprise pubblico di WebSphere Commerce, si modifica il codice WebSphere Commerce. Pertanto, la tecnica di distribuzione dei bean entità è leggermente diversa da quella utilizzata per i nuovi bean entità. Di seguito è riportata una panoramica sulle fasi di distribuzione.

1. Creare un file JAR EJB 1.1 di esportazione per il gruppo EJB di WebSphere Commerce che contiene i bean entità modificati. Con la scheda EJB selezionata nello spazio di lavoro VisualAge per Java, selezionare il gruppo EJB appropriato e quindi esportarlo in un file JAR EJB 1.1 di esportazione. Durante l'operazione di denominazione del file JAR, è possibile utilizzare la convenzione di denominazione `Cust_`*`NomeGruppoEJB`*`-ejb_DT.jar`, dove *`NomeGruppoEJB`* è il nome del gruppo EJB che è stato modificato, con il suffisso _DT accodato alla fine del nome del file JAR. Ad esempio, durante la denominazione del file JAR per il gruppo EJB di WCSUser, è possibile denominare il file come `Cust_WCSUser-ejb_DT.jar`. Il suffisso _DT viene utilizzato semplicemente come promemoria che occorre in un secondo momento passare questo file JAR allo strumento EJBDeploy per generare il codice di distribuzione dei bean contenuti nel gruppo EJB. Per ulteriori informazioni, fare riferimento a "Creazione di file JAR per i bean entità personalizzati di WebSphere Commerce" a pagina 353.
2. Creare un nuovo file client JAR che contiene il codice client per tutti i gruppi EJB di WebSphere Commerce. Con tutti i gruppi EJB di WebSphere Commerce selezionati (tutti i nomi cominciano con `WCS`), esportare ad un file JAR client. Per ulteriori informazioni, fare riferimento a "Creazione di file JAR per i bean entità personalizzati di WebSphere Commerce" a pagina 353.
3. Copiare i file JAR nella directory appropriata della macchina WebSphere Commerce Server di destinazione. Per ulteriori informazioni, fare riferimento a "Memorizzazione delle risorse su WebSphere Commerce Server di destinazione" a pagina 356.
4. Sul WebSphere Commerce Server di destinazione, utilizzare lo strumento EJBDeploy fornito con WebSphere Application Server per generare il codice di distribuzione dei bean enterprise contenuti nel gruppo EJB che si sta distribuendo. Questo strumento utilizza ilfile JAR creato al passo 1 come file di immissione creando il corrispondente file JAR contenente il

codice di distribuzione per tutti i bean enterprise nel gruppo EJB. Per ulteriori informazioni, fare riferimento a “Generazione del codice di distribuzione” a pagina 360.

5. Sul WebSphere Commerce Server di destinazione, modificare il livello di isolamento transazione dei bean enterprise contenuti nel file JAR del codice di distribuzione. Utilizzare l'utility della riga comandi `modifyIsolationLevel` fornita con WebSphere Commerce per impostare in modo appropriato il livello di isolamento delle transazioni per il proprio tipo di database. Per ulteriori informazioni, fare riferimento a “Modifica del livello di isolamento delle transazioni dei bean entità” a pagina 363.
6. Sul WebSphere Commerce Server di destinazione, esportare l'enterprise application corrente di WebSphere Commerce da WebSphere Application Server. Dopo il completamento dell'esportazione, viene creato un file .ear che contiene l'intera applicazione. Per esportare l'applicazione corrente, aprire la WebSphere Application Server Console di gestione, selezionare l'application enterprise di WebSphere Commerce, quindi l'opzione di esportazione. Al termine, viene creato un file `WC_Enterprise_App_`*`Nomeistanza`*`.ear`. Per ulteriori informazioni, fare riferimento a “Esportazione dell'enterprise application di WebSphere Commerce corrente” a pagina 364.
7. Sul WebSphere Commerce Server di destinazione, esportare le informazioni sulla configurazione per i bean enterprise contenuti nell'enterprise application corrente di WebSphere Commerce in esecuzione su WebSphere Application Server. Queste informazioni vengono esportate in un file XML mediante l'opzione `-export` della funzione di riga comandi `XMLConfig` che viene fornita dal WebSphere Application Server. Il file XML generato (riferito qui come file `OutputFile.xml`) contiene una stanza di informazioni sulla configurazione per ciascun bean enterprise contenuto nell'enterprise application. Per ulteriori informazioni, fare riferimento a “Esportazione delle informazioni sulla configurazione per i bean enterprise” a pagina 366. È necessario modificare, all'interno di questo file, la stanza che descrive il bean enterprise che è stato modificato, in modo che punti al nuovo file`Cust_`*`NomeGruppoEJB`*`-ejb.jar`.
8. Sul WebSphere Commerce Server di destinazione, utilizzare lo Strumento di assemblaggio delle applicazioni per integrare il gruppo EJB modificato nell'enterprise applicacation. Utilizzando questo strumento, aprire il file `.ear` per l'applicazione corrente. Una volta aperto il file, svolgere le seguenti attività:
   a. Prendere nota delle informazioni class path per la versione originale del gruppo EJB che è stato modificato. Ad esempio, se è stato modificato il bean User nel gruppo EJB di WCSUser, copiare le informazioni class path per il gruppo WCSUser in un file di testo.
   b. Cancellare la versione originale del gruppo EJB modificato (ad esempio, cancellare il gruppo WCSUser).

c. Importare la nuova versione del gruppo EJB modificato.
d. Applicare le informazioni class path originali al gruppo EJB appena importato.
e. Configurare la sicurezza WebSphere Application Server per i metodi contenuti in tutti i bean enterprise del gruppo EJB modificato.
f. Salvare l'applicazione in un nuovo `.ear`

Per ulteriori informazioni, fare riferimento a "Assemblaggio dei bean enterprise modificati nell'enterprise application" a pagina 376.

9. Importare la nuova enterprise application in WebSphere Application Server. Questa operazione è costituita dalle tre seguenti attività secondarie:
   a. Utilizzando il WebSphere Application Server Console di gestione, arrestare e rimuovere l'enterprise application originale di WebSphere Commerce. Per ulteriori informazioni, fare riferimento a "Arresto e rimozione dell'enterprise application" a pagina 381.
   b. Utilizzando l'opzione `-import` dell'utility della riga comandi `XMLConfig` per importare la nuova enterprise application in WebSphere Application Server. Per ulteriori informazioni, fare riferimento a "Importazione dell'enterprise application" a pagina 382.
   c. Utilizzando la WebSphere Application Server Console di gestione, aggiornare le informazioni visualizzate in modo che appaia la nuova enterprise application, quindi avviare la nuova applicazione. Per ulteriori informazioni, fare riferimento a "Avvio dell'enterprise application" a pagina 383.

Distribuzione dei nuovi bean di dati da utilizzare in Commerce Studio

Se si sta utilizzando Commerce Studio per la distribuzione delle maschere JSP, i nuovi bean di dati devono essere distribuiti in Commerce Studio. In particolare, deve essere creato un file JAR per i bean di dati.

Per creare questo file JAR, fare clic con il pulsante destro del mouse sul pacchetto dei bean di dati ed esportarli in un file JAR. Nelle opzioni, includere quanto segue:

- **classe**
- **risorsa**
- **bean**

Inoltre, selezionare **Seleziona tipi e risorse segnalate** per includere i comandi e le risorse necessarie per i bean di dati.

Una volta esportato il file JAR, aggiornare il classpath per Commerce Studio per includere questo file JAR.

Se si desidera modificare un bean di dati esistente di WebSphere Commerce, creare un nuovo bean di dati che estenda quello da personalizzare. In effetti, si crea un nuovo bean di dati che deve essere distribuito con la procedura descritta in questa sezione.

Distribuzione di bean entità pubblici personalizzati da utilizzare in Commerce Studio

Se si modifica un bean entità pubblico e si utilizza Commerce Studio per lo sviluppo delle maschere JSP, creare un nuovo file JAR client per tutti i bean entità pubblici e modificare il classpath per Commerce Studio immettendo il nome del nuovo file JAR prima del nome del file JAR originale. Commerce Studio utilizza il file JAR client quando i bean di dati vengono utilizzati in Page Designer.

Per creare questo file JAR, utilizzare gli strumenti di VisualAge per Java. Dalla pagina EJB in VisualAge per Java selezionare *tutti* i gruppi EJB di WebSphere Commerce (non solo quelli modificati) e esportarli in un file JAR del client. Una volta completata l'esportazione, assicurarsi di specificare questo nuovo file JAR all'inizio del classpath per Commerce Studio.

File di log

Durante il processo di installazione del prodotto, lo sviluppo e la distribuzione del codice, vengono generati file di log. In questa sezione verranno elencati alcuni file di log che potranno essere consultati durante queste fasi.

File di log di Commerce Studio

Durante il processo di installazione, Commerce Studio crea file di log. Per accedere a questi file di log, aprire il seguente file:

```
C:\Winnt\WCStudioInstall.log
```

File di log per la configurazione di Commerce Studio per WebSphere Test Environment

Se si seleziona l'opzione per eseguire attività di sviluppo di back-end, vengono eseguite numerose operazioni di configurazione per WebSphere Test Environment durante l'installazione di WebSphere Commerce Studio. Ad esempio, se si seleziona l'opzione che consente di includere un negozio di esempio eseguito nel componente WebSphere Test Environment di VisualAge per Java, vengono generati i file di log correlati al processo di caricamento di massa e alla creazione del database. Per accedere a questi file, passare alla seguente directory:

unità:\WebSphere\CommerceServerDev\instances*nome_istanza*\logs

File di log dei componenti di WebSphere Commerce in esecuzione all'interno di WebSphere Test Environment

Quando si esegue un negozio oppure si esegue la verifica di un singolo componente in WebSphere Test Environment, è possibile che vengano generati un file di traccia e un file di messaggi. L'ubicazione predefinita di questi file è la seguente directory:

unità:\WebSphere\CommerceServerDev\instances*nome_istanza*\logs

File di log del comando modifyIsolationLevel in esecuzione

Quando si distribuiscono i bean enterprise, si utilizza il comando modifyIsolationLevel per modificare il livello di isolamento transazione di ciascun bean enterprise nel file JAR. I file di log generati vengono memorizzati nel file di log che specificato durante l'esecuzione del comando. Per ulteriori informazioni, fare riferimento a "Modifica del livello di isolamento delle transazioni dei bean entità" a pagina 363.

Registrazione in VisualAge per Java

Quando si esegue il codice in WebSphere Test Environment, la finestra Console viene eseguita come log attivo. Inoltre, è possibile modificare il livello di traccia per un server EJB per aumentare la quantità di informazioni che possono essere ricercate nella finestra Console. Queste informazioni vengono specificate come livello di traccia, all'interno delle proprietà del server EJB.

Metodo di pagamento di prova

Per impostazione predefinita, il negozio di esempio in esecuzione in WebSphere Test Environment utilizza per un metodo di pagamento di prova. Questo metodo consente di completare il flusso di acquisti in WebSphere Test Environment, senza richiamare un Payment Manager remoto. Lo stato degli ordini collocati utilizzando questo metodo di pagamento è 'M', ossia il pagamento è stato avviato ed è in attesa di elaborazione.

Questo metodo consente di completare solo un acquisto e non permette l'elaborazione degli ordini inoltrati. Pertanto, questo metodo di pagamento può essere utilizzato solo in WebSphere Test Environment.

Le classi di implementazione di questo metodo sono tipi di comando per politiche aziendali. Pertanto, la selezione della classe di implementazione da utilizzare è regolata da una politica aziendale. Per impostazione predefinita, il negozio di esempio InFashion in esecuzione in WebSphere Test Environment include una politica aziendale (TestPaymentMethod) che utilizza le seguenti implementazioni dei comandi correlati al pagamento:

- com.ibm.commerce.payment.commands.DoPayment**Test**CmdImpl
- com.ibm.commerce.payment.commands.CheckPaymentAccept**Test**CmdImpl

- com.ibm.commerce.payment.commands.DoCancel**Test**CmdImpl
- com.ibm.commerce.payment.commands.DoDeposit**Test**CmdImpl
- com.ibm.commerce.payment.commands.DoRefund**Test**CmdImpl

È importante sottolineare che questi comandi vengono forniti per eseguire una verifica del solo processo di checkout. Mentre la precedente suite di comandi viene fornita per motivi di completezza, DoDepositTestCmdImpl è uno stub di comando che lancia un'eccezione se richiamato. Non utilizzare funzioni per gestione ordini con questi ordini. Se occorre verificare queste altre funzioni, è possibile configurare WebSphere Test Environment per utilizzare un Payment Manager remoto.

Durante la distribuzione del codice nel WebSphere Commerce Server di destinazione, è importante non copiare la politica aziendale relativo al metodo di pagamento di prova dalla macchina di sviluppo in quella di destinazione. Inoltre, è importante non registrare i comandi relativi al metodo di pagamento di prova nel WebSphere Commerce Server di destinazione.

Per ulteriori informazioni sulla disabilitazione del metodo di pagamento di prova, consultare il capitolo "Configurazione di VisualAge per Java" nel manuale *WebSphere Commerce Business Edition Guida all'installazione* o *WebSphere Commerce Professional Edition Guida all'installazione*.

Utilizzo di un Payment Manager remoto

Se il lavoro di sviluppo prevede l'utilizzo di ordini una volta eseguiti, ad esempio, modificando una procedura del processo di gestione ordini, è necessario configurare il negozio in esecuzione in WebSphere Test Environment per utilizzare un Payment Manager remoto. Per ulteriori informazioni sulle operazioni di configurazione, fare riferimento a *WebSphere Commerce Studio, Business Developer Edition - Guida all'installazione*. Questa documentazione fornisce anche informazioni sulla rimozione degli ordini eseguiti con il metodo di pagamento di prova dal proprio database.

Parte 4. Supporti didattici

Questa sezione contiene dei supporti didattici che permettono di conoscere meglio l'applicazione e-commerce. Sono disponibili i seguenti supporti didattici:

- Creazione di una nuova logica aziendale.
  In questo supporto didattico viene illustrato il processo di sviluppo per la creazione di una nuova logica aziendale. Esso include le seguenti attività secondarie:
  - Preparazione di un progetto di esempio
  - Scrittura di nuovi comandi
  - Creazione di un nuovo bean di dati e invio di proprietà su richiesta
  - Utilizzo di bean entità
  - Creazione di un nuovo bean enterprise
- Aggiornamento della logica aziendale attuale
  Questo supporto didattico mostra il processo di sviluppo per l'aggiornamento della logica aziendale esistente di WebSphere Commerce. Esso include le seguenti attività secondarie:
  - Estensione di un comando di controller di WebSphere Commerce esistente
  - Modifica di un bean entità pubblico di WebSphere Commerce esistente
  - Creazione di un nuovo comando di attività che estende un comando di attività di WebSphere Commerce esistente.

I supporti didattici contengono sezioni di codice da immettere in VisualAge per Java. È consigliabile scaricare una versione PDF di questa documentazione ai seguenti siti Web:

Business

```
http://www.ibm.com/software/webservers/commerce/wc_be
   /lit-tech-general.html
```

Professional

```
http://www.ibm.com/software/webservers/commerce/wc_pe
   /lit-tech-general.html
```

È possibile tagliare e incollare questa sezione di codice dalla versione PDF della Guida per il programmatore.

Capitolo 9. Supporto didattico: creazione di nuova logica aziendale

Ambiente dei supporti didattici

I supporti didattici contenuti in questo manuale richiedono l'installazione di WebSphere Commerce Business Edition V5.4 o WebSphere Commerce Professional Edition V5.4. Inoltre, quando si installa il prodotto, è necessario selezionare le seguenti opzioni:

- **Sviluppo di attività fronte negozio con WebSphere Studio**
- **Sviluppo di logica backend negozio con VisualAge per Java**
- **Crea database**
- **Includi negozio di esempio**

Per ulteriori informazioni sull'installazione e sulla configurazione, seguire le istruzioni riportate in *WebSphere Commerce Studio, Business Developer Edition Guida all'installazione* oppure*WebSphere Commerce Studio Professional Developer Edition Guida all'installazione*.

Prima di avviare i supporti didattici, è necessario che il negozio di esempio di WebSphere Test Environment sia completamente funzionante e che sia possibile effettuare un acquisto all'interno del negozio.

Operazioni di distribuzione del codice nel supporto didattico

I supporti didattici contenuti in questo manuale includono operazioni facoltative che descrivono la modalità di distribuzione di un codice personalizzato a un WebSphere Commerce Server di destinazione. Si assuma che il WebSphere Commerce Server di destinazione sia in esecuzione su Windows NT o su Windows 2000 su una macchina diversa da quella di sviluppo. Se si sta utilizzando WebSphere Commerce Studio Business Developer Edition come proprio ambiente di sviluppo, occorre eseguire la distribuzione in WebSphere Commerce Business Edition. Se si sta utilizzando WebSphere Commerce Studio Professional Developer Edition come proprio ambiente di sviluppo, occorre eseguire la distribuzione in WebSphere Commerce Professional Edition.

Inoltre, è necessario aver creato sul WebSphere Commerce Server di destinazione un negozio sulla base del negozio di esempio InFashion. All'interno del negozio, è necessario poter completare un acquisto. Per quanto riguarda l'elaborazione del pagamento, è possibile utilizzare un Payment Manager remoto o locale.

Se si effettua una distribuzione di WebSphere Commerce Server di destinazione che si trova sulla stessa macchina dell'ambiente di sviluppo oppure in esecuzione su un sistema operativo differente, consultare Capitolo 8, "Strumenti di sviluppo e distribuzione" a pagina 191 e Appendice B, "Dettagli sulla configurazione" a pagina 349 per le informazioni sulla distribuzione appropriate.

Preparazione del progetto di esempio

In questa sezione, viene importato il magazzino `WC_SAMPLE_54.dat` necessario per eseguire il supporto didattico per la "creazione di nuova logica aziendale".

Per preparare l'ambiente, effettuare le seguenti operazioni:

1. Assicurarsi di utilizzare il magazzino di `WC_54.dat` del codice WebSphere Commerce. Questo magazzino è disponibile sul CD di di WebSphere Commerce Business Edition V5.4 Disco 2 o di WebSphere Commerce Professional Edition V5.4 Disco 2.
2. Importare il progetto di esempio nello spazio di lavoro, nel modo seguente:
   a. Inserire il seguente CD nella relativa unità sulla macchina di sviluppo:
      - Business WebSphere Commerce Business Edition, V5.4 Disco 2
      - Professional WebSphere Commerce Professional Edition, V5.4 Disco 2
   b. Aprire VisualAge per Java.
   c. Dal menu **File** scegliere **Importa**
      Verrà visualizzata la SmartGuide di importazione.
   d. Scegliere un **magazzino** da importare e fare clic su **Avanti**.
   e. Nella finestra per l'importazione da un altro magazzino, procedere nel modo seguente:
      1) Selezionare **Magazzino locale**.
      2) Nel campo **Nome del magazzino**, immettere *unità_CD*`:\repository\samples\programguide\WC_SAMPLE_54.dat` dove *unità_CD* è l'unità CD.
      3) Selezionare **Progetti** e fare clic su **Dettagli**. Selezionare il progetto **_WCSamples**. Selezionare la versione **WC Sample 5.4** e fare clic su **OK**.
      4) Assicurarsi che l'opzione **Aggiungere l'edizione del progetto più recente allo spazio di lavoro** sia selezionata.
      5) Fare clic su **Fine** per iniziare l'importazione.
         L'importazione può richiedere alcuni minuti.

3. Accertarsi che il proprietario dello spazio di lavoro sia impostato come Sviluppatore WCS, attenendosi alle seguenti operazioni:
   a. Dal menu **Spazio di lavoro**, selezionare **Modifica proprietario spazio di lavoro**.
   b. Selezionare **WCS Developer** e fare clic su **OK**.
4. Copiare le maschere JSP per gli esercizi nella directory appropriata, per poterle utilizzare in WebSphere Test Environment. Per copiare questi file, effettuare le seguenti operazioni:
   a. Individuare i file `Sample.jsp` e `Sample_All.jsp` nella seguente directory
      unità_CD`:\repository\samples\programguide\`

      dove *unità_CD* è l'unità CD della seguente directory:
   b. Copiare questi due file nella seguente directory:
      unità_vaj`:\VAJava\ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web`

      dove *unità_vaj* è l'unità sulla quale è stato installato VisualAge per Java.
5. Copiare i file delle politiche dei controlli accessi nella directory appropriata. Questi file vengono utilizzati per caricare nuove politiche di controlli accessi per le nuove risorse create durante l'esecuzione dei supporti didattici. Per copiare questi file, effettuare le seguenti operazioni:
   a. Visualizzare la seguente directory:
      unità_CD`:\repository\samples\programguide\`
   b. Nella directory, individuare i seguenti file:
      - `SampleCmdACPolicy.xml`
        Questo file XML contiene la politica di controllo accessi utilizzata quando si crea un nuovo comando di controller.
      - `SampleACPolicy.xml`
        Questo file XML contiene la politica di controllo accessi utilizzata quando si crea un nuovo bean enterprise.
      - `SampleACPolicy_`*locale*`.xml`
        dove *locale* è l'identificativo della lingua. Questo file XML contiene la descrizione della politica del controllo accessi.
   c. Copiare i file precedenti nella seguente directory:
      unità`:\WebSphere\CommerceServerDev\xml\policies\xml`
      dove *unità* è l'unità sulla quale è stato installato WebSphere Commerce Studio.
6. Verificare che sia possibile avviare i supporti didattici nel proprio ambiente effettuando le seguenti operazioni:
   a. Avviare il server di nomi permanenti, come descritto a pagina 347.
   b. Avviare il server EJB, come descritto a pagina 348.

c. Avviare il motore servlet, come descritto a pagina 348

d. Aprire il browser e immettere il seguente URL:

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
   storeId=ID_negozio&catalogId=Id_catalogo&langId=-1
```

dove *ID_negozio* è l'identificativo del negozio di esempio (10001 è un valore di esempio) e *Id_catalogo* è l'identificativo del catalogo del negozio di esempio (10001 è un valore di esempio).
Quando viene visualizzata la home page del negozio di esempio, selezionare un prodotto e acquistarlo. Alla fine della procedura, viene visualizzata la pagina "Conferma ordine" del flusso degli acquisti.

Per verificare il valore storeId per il negozio, è possibile fare riferimento alla tabella STOREENT.

e. Chiudere tutti i browser e arrestare WebSphere Test Environment.

A questo punto, è possibile avviare i supporti didattici.

Scrittura dei comandi

Scrittura dei comandi di controller

Questa sezione espone le modalità di scrittura di un nuovo comando di controller. Dopo aver completato questo esercizio, verrà visualizzato un nuovo comando chiamato *MyNewControllerCmd*. Questo comando viene utilizzato da tutti i negozi e ciascuno di essi utilizza la stessa implementazione del comando.

La creazione di un nuovo comando di controller prevede tre fasi preliminari:

1. Registrazione del comando nella struttura del registro dei comandi.
2. Creazione dell'interfaccia per il comando.
3. Creazione della classe di implementazione per il comando

Per ulteriori informazioni sui comandi, consultare "Modello di progettazione dei comandi" a pagina 22.

Prima di eseguire questo supporto didattico
Completare tutte le procedure descritte in "Preparazione del progetto di esempio" a pagina 208.

Per scrivere un nuovo comando, effettuare le seguenti operazioni:

1. Innanzitutto, è necessario stabilire un nome da associare al comando. In questo esempio il comando viene definito *MyNewControllerCmd*.

2. Il comando deve essere registrato nella struttura del registro comandi. Il processo di registrazione per il nuovo comando di controller comprende la creazione di una voce nella tabella URLREG.

   DB2 Se si utilizza un database DB2, effettuare le seguenti operazioni per registrare il comando:

   a. Aprire il Centro comandi DB2 (**Start > Programmi >IBM DB2 > Centro di comandi**).
   b. Dal menu **Strumenti** selezionare **Impostazioni strumenti**.
   c. Selezionare la casella **Utilizza carattere fine istruzione** e assicurarsi che il carattere specificato sia un punto e virgola (**;**)
   d. Dopo aver selezionato la scheda Script, creare la voce richiesta nella tabella URLREG immettendo le seguenti informazioni nella finestra dello script:

      ```
      connettersi a nome_database;
      insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
          DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
          'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
          'This is a new controller command for test/education purposes.',
          null)
      ```

      dove *nome_database* è il nome del database e fare clic sull'icona di esecuzione.
      Questo comando viene utilizzato da tutti i venditori (indicato dal valore 0 per STOREENT_ID).

   Oracle Se si utilizza un database Oracle, effettuare le seguenti operazioni per registrare il comando:

   a. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
   b. Nel campo **Nome utente** immettere il nome utente Oracle.
   c. Nel campo **Password**, immettere la password per Oracle.
   d. Nel campo **Stringa host** immettere la propria stringa di collegamento.
   e. Nella finestra SQL Plus, immettere il testo riportato di seguito per creare la voce richiesta nella tabella URLREG:

      ```
      insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
          DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
          'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
          'This is a new controller command for test/education purposes.',
          null);
      ```

      e premere Invia per eseguire l'istruzione SQL.
   f. Immettere quanto segue per confermare le modifiche al database:

      ```
      commit;
      ```

e premere Invia per eseguire l'istruzione SQL.

Nota: Per semplicità, in questo esercizio il nuovo comando dispone di una sola classe di implementazione, ossia la stessa classe di implementazione utilizzata da tutti i negozi. Poiché questa classe viene specificata direttamente nel codice per l'interfaccia, non è necessario registrare l'associazione tra l'interfaccia e la classe di implementazione nella tabella CMDREG. Fuori dal supporto didattico, registrare il comando di controller nella tabella CMDREG, oltre che nella tabella URLREG.

3. I comandi di controller devono restituire una visualizzazione. Il nuovo comando di controller creato restituisce la visualizzazione SampleViewTask. È necessario registrare la visualizzazione SampleViewTask nella tabella VIEWREG.

   DB2 Se si utilizza un database DB2, effettuare le seguenti operazioni per registrare la visualizzazione:

   a. Creare una voce nella tabella VIEWREG immettendo quanto segue nella finestra dello script (è possibile che si renda necessario cancellare dapprima la precedente istruzione SQL dalla finestra):

      ```
      insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
         CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
      values ('SampleViewTask',-1, 0,
          'com.ibm.commerce.command.ForwardViewCommand',
          'com.ibm.commerce.command.HttpForwardViewCommandImpl',
          'docname=Sample.jsp','This is a sample view for the
          Bonus Point exercise', 0, null)
      ```

      e fare clic sull'icona di esecuzione.

   Oracle Se si utilizza un database Oracle, effettuare le seguenti operazioni per registrare la visualizzazione:

   a. Nella finestra SQL Plus, immettere la seguente istruzione SQL:

      ```
      insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
         CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
      values ('SampleViewTask',-1, 0,
         'com.ibm.commerce.command.ForwardViewCommand',
         'com.ibm.commerce.command.HttpForwardViewCommandImpl',
         'docname=Sample.jsp','This is a sample view for the
         Bonus Point exercise', 0, null);
      ```

      e premere Invia per eseguire l'istruzione SQL.

   b. Immettere quanto segue per confermare le modifiche al database:

      ```
      commit;
      ```

      e premere Invia per eseguire l'istruzione SQL.

4. Nella finestra Spazio di lavoro di VisualAge per Java, espandere il progetto **_WCSamples**.
5. Espandere il pacchetto **com.ibm.commerce.sample.commands**, fare clic con il pulsante destro del mouse sull'interfaccia **MyNewControllerCmd** e selezionare **Aggiungi > Campo**.
   Viene visualizzata la SmartGuide per la creazione dei campi.
6. Creare un campo che specifica la classe di implementazione predefinita per l'interfaccia, eseguendo le operazioni descritte:
   a. Nel campo **Nome campo** immettere `defaultCommandClassName`.
   b. Da elenco a discesa sul **Tipo campo**, selezionare `Stringa`.
   c. Nel campo Valore iniziale, immettere `"com.ibm.commerce.sample.commands.MyNewControllerCmdImpl"`. Assicurarsi di includere le doppie virgolette.
   d. Fare clic su **Fine**.
      Viene generato il codice per il nuovo campo.

In qualsiasi punto del supporto didattico, quando si sovrascrive un campo o un metodo che si trova nella superclasse, è possibile che venga visualizzato un messaggio in cui si indica che il nuovo campo o metodo nasconderà il campo o il metodo ereditato. In tal caso, fare clic su **Sì** per continuare.

7. Espandere la classe **MyNewControllerCmdImpl** e selezionare il relativo metodo **performExecute** per visualizzarne il codice di origine.
8. Nel codice di origine del metodo performExecute, eliminare il commento dalla prima e dalla quinta sezione. La prima sezione inserisce il seguente codice nel metodo:

```
// Create a new TypedProperties for output.
  TypedProperty rspProp = new TypedProperty();
```

   La quinta sezione inserisce il seguente codice per il metodo:

```
// see how controller command call a JSP

 rspProp.put(ECConstants.EC_VIEWTASKNAME, "SampleViewTask");
 setResponseProperties( rspProp );
```

   Salvare il proprio lavoro (**Ctrl + S**). La sezione di codice sopra riportata imposta il nome di visualizzazione da restituire al comando di controller.
9. Specificare il controllo accessi a livello di comando per questo nuovo comando di controller. In questo caso, la politica di controllo accessi a livello di comando specificherà che tutti gli utenti possono eseguire il comando. Questa politica viene specificata nel file `SampleCmdACPolicy.xml`. Per caricare la nuova politica di controllo accessi, effettuare le seguenti operazioni:

a. Alla richiesta comandi, passare alla seguente directory:
   unità:\WebSphere\CommerceServerDev\bin
b. È necessario caricare il comando acpload, il cui formato è:

   ```
   acpload nome_db utente_db password_db inputXMLFile
   ```

   dove
   - *nome_db* è il nome del database
   - *utente_db* è il nome dell'utente del database
   - *password_db* è la password del database
   - *FileXMLinput* è il nome del file XML che contiene la politica.

   Ad esempio, è possibile inviare il seguente comando:

   ```
   acpload VAJ_Demo user password SampleCmdACPolicy.xml
   ```

10. Aggiungere il nuovo progetto _WCSamples al classpath per il motore servlet effettuando le seguenti operazioni:
    a. Dal menu **Spazio di lavoro**, selezionare **Strumenti > WebSphere Test Environment**.
       Si apre WebSphere Test Environment Centro di controllo.
    b. Fare clic su **Motore servlet** e quindi su **Modifica classpath**
       Nella finestra Classpath motore servlet, fare clic su **Seleziona tutto**, quindi su **OK**.
11. Eseguire un test sul nuovo comando nel modo seguente:
    a. Avviare il server di nomi permanenti, come descritto a pagina 347.
    b. Avviare il server EJB, come descritto a pagina 348.
    c. Avviare il motore servlet, come descritto a pagina 348
    d. Aprire il browser e immettere il seguente URL:

       ```
       http://localhost:8080/webapp/wcs/stores/servlet/
          MyNewControllerCmd
       ```

       Dopo alcuni minuti il browser visualizza una pagina in cui è illustrato il "JSP di esempio", come riportato di seguito:

Figura 31.

12. Creare una versione del codice nello stato corrente. In tal modo, è possibile ripristinare lo stato corrente del codice in qualsiasi momento. Per creare la versione, procedere nel modo seguente:
    a. Selezionare i pacchetti **com.ibm.commerce.sample.commands** e **com.ibm.commerce.sample.databeans** (tenere premuto il tasto `Ctrl` mentre si evidenziano per selezionare più pacchetti), fare clic con il pulsante destro del mouse e selezionare **Gestisci > Crea edizione aperta**.
    b. Fare clic con il pulsante destro del mouse sul progetto **_WCSamples** e selezionare **Gestione > Crea edizione aperta**.
    c. Fare clic con il pulsante destro del mouse di nuovo sul progetto **_WCSamples** e selezionare **Gestione > Versione**.
    Verrà aperta la finestra relativa alla conversione degli elementi selezionati.
    d. Selezionare il pulsante di scelta **Nome unico**, immettere `mySample 1.2 Completed` e fare clic su **OK**.

A questo punto, viene aggiunto un nuovo comando e viene testato brevemente nell'ambiente integrato di verifica.

Modifica di MyNewControllerCmd

Nella sezione precedente è stato creato MyNewControllerCmd. In questo esercizio esaminare il contenuto del nuovo comando per comprendere meglio le modalità di creazione di un comando di controller personalizzato.

Innanzitutto, esaminare la struttura del codice creato. L'interfaccia MyNewControllerCmd amplia l'interfaccia ControllerCommand. Definisce anche la classe di implementazione da utilizzare come predefinita. Questa classe viene utilizzata quando il comando non è registrato nella tabella CMDREG oppure quando una classe di implementazione non è specificata in quella tabella.

È stata creata anche la classe MyNewControllerCmdImpl. La classe di implementazione è la classe che dovrà contenere la logica aziendale (o richiamare i comandi di attività per eseguire singole attività aziendali) da implementare. La classe di implementazione contiene i metodi seguenti:

Metodi in MyNewControllerCmdImpl	**Descrizione**
MyNewControllerCmdImpl()	Il metodo constructor.
validateParameters()	Utilizzato per la convalida di tipo server dei parametri di immissione del comando.
isGeneric()	Stabilisce se un utente generico può richiamare il comando.
isRetriable()	Stabilisce se il comando viene richiamato dopo un rollback del database.
performExecute()	Contiene la logica aziendale del comando.

Le sezioni seguenti contengono maggiori dettagli su come aggiornare il nuovo comando di controller.

Invio di variabili alla maschera JSP

In questa sezione, sono state apportate modifiche a MyNewControllerCmd in modo da inoltrare le variabili alla maschera JSP. Per visualizzare le variabili, utilizzare un nuovo bean di dati denominato DataBeanSampleBean. Per abilitare la visualizzazione delle variabili nella maschera JSP, effettuare le seguenti operazioni:

1. Nella finestra Spazio di lavoro di VisualAge per Java, espandere il progetto **_WCSamples**.
2. Espandere il pacchetto **com.ibm.commerce.sample.commands**, quindi la classe **MyNewControllerCmdImpl** e selezionarne il metodo **performExecute**.
3. Nel codice di origine per il metodo performExecute, eliminare il commento dalla sezione 2. In tal modo, viene inserito il seguente codice nel metodo:

```
// see how controller command pass in variables to JSP

 // Add additional parameters in controller command
 // to rspProp for response
```

```
//
rspProp.put("ControllerParm1", "Hello world");
rspProp.put("ControllerParm2", "Have a nice day!");
```

La sezione di codice riportata precedentemente crea due nuovi parametri che vengono inseriti nelle proprietà da inoltrare alla maschera JSP. Salvare il proprio lavoro(**Ctrl + S**).

4. Il bean di dati DataBeanSampleBean viene utilizzato dalla maschera JSP per visualizzare le variabili. Il bean è stato creato automaticamente ma è necessario modificare il codice di origine come segue:
   a. Espandere il pacchetto **com.ibm.commerce.sample.databeans**.
   b. Espandere la classe **DataBeanSampleBean**, quindi selezionare il metodo **setRequestProperties** per visualizzarne il codice di origine.
   c. Nel codice di origine del metodo setRequestProperties, eliminare il commento dalla prima sezione. In tal modo, sarà inserito nel metodo il seguente codice:

      ```
      // copy input TypedProperties to local

       requestProperties = aParam;
      ```

      Salvare il proprio lavoro. La sezione di codice precedente copia i valori da aParam (definiti nella superclasse) localmente. Viene utilizzato per richiamare le proprietà dall'oggetto richiesta.
5. Aggiornare il file Sample.jsp per utilizzare il nuovo bean di dati e visualizzare le variabili effettuando le seguenti operazioni:
   a. Utilizzando un editor di testo, aprire i file `Sample.jsp` e `Sample_All.jsp`. Questi file si trovano nella seguente directory: *unità_vaj*`:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host \default_app\web`
   b. Copiare la sezione 1 del codice da Sample_All.jsp a Sample.jsp tra `<!-- SECTION 1 -->` e `<!-- END OF SECTION 1 -->`. In tal modo, viene inserito il seguente codice nella maschera JSP:

      ```
      <!-- SECTION 1 -->
      <%
      DataBeanSampleBean testBean = new DataBeanSampleBean ();
      com.ibm.commerce.beans.DataBeanManager.activate (testBean, request);
      %>
      <!-- END OF SECTION 1 -->
      ```

      Questa sezione di codice avvia il bean di dati.
   c. Copiare la sezione 2 da Sample_All.jsp a Sample.jsp tra `<!-- SECTION 2 -->` e `<!-- END OF SECTION 2 -->`. In tal modo, viene inserito il seguente codice nella maschera JSP:

      ```
      <!-- SECTION 2 -->
      <%
      TypedProperty prop = testBean.getRequestProperties();
      ```

```
out.print("<B>List of name value pairs in TypedProperties
           object</B><P>");

// convert from request Properties to query string
for (Enumeration pns = prop.keys(); pns.hasMoreElements();) {
    String paramName = (String) pns.nextElement();
 // do not add the url parameter to the query string
 Object val = prop.get(paramName,null);
      if (val != null) {
            if (val.getClass().isArray()) {
   // flatten the array
   String[] oarray = (String[]) val;
             int len = java.lang.reflect.Array.getLength(val);
             for (int i = 0; i < len; i++) {
         out.print(paramName + "[" + i + "]=" + oarray[i] + "<br>");
         }
        } else {
         // assume that it is a String
    out.print(paramName + "=" + val.toString() + "<br>");
    }
  }
 }
%>
<P>
<!-- END OF SECTION 2 -->
```

Salvare questo file.
Questa sezione di codice utilizza il metodo getRequestProperties dell'oggetto bean di dati testBean. Esegue un ciclo delle proprietà e le visualizza nel browser.

6. Eseguire un test delle modifiche per verificare che le variabili vengano visualizzate nella maschera JSP effettuando le seguenti operazioni:
   a. Accertarsi che WebSphere Test Environment sia in esecuzione.
   b. In un browser, immettere quanto segue:

      ```
      http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd
      ```

      Viene visualizzato il file JSP di esempio che riporta le proprietà del comando di controller. Viene visualizzato quanto segue:

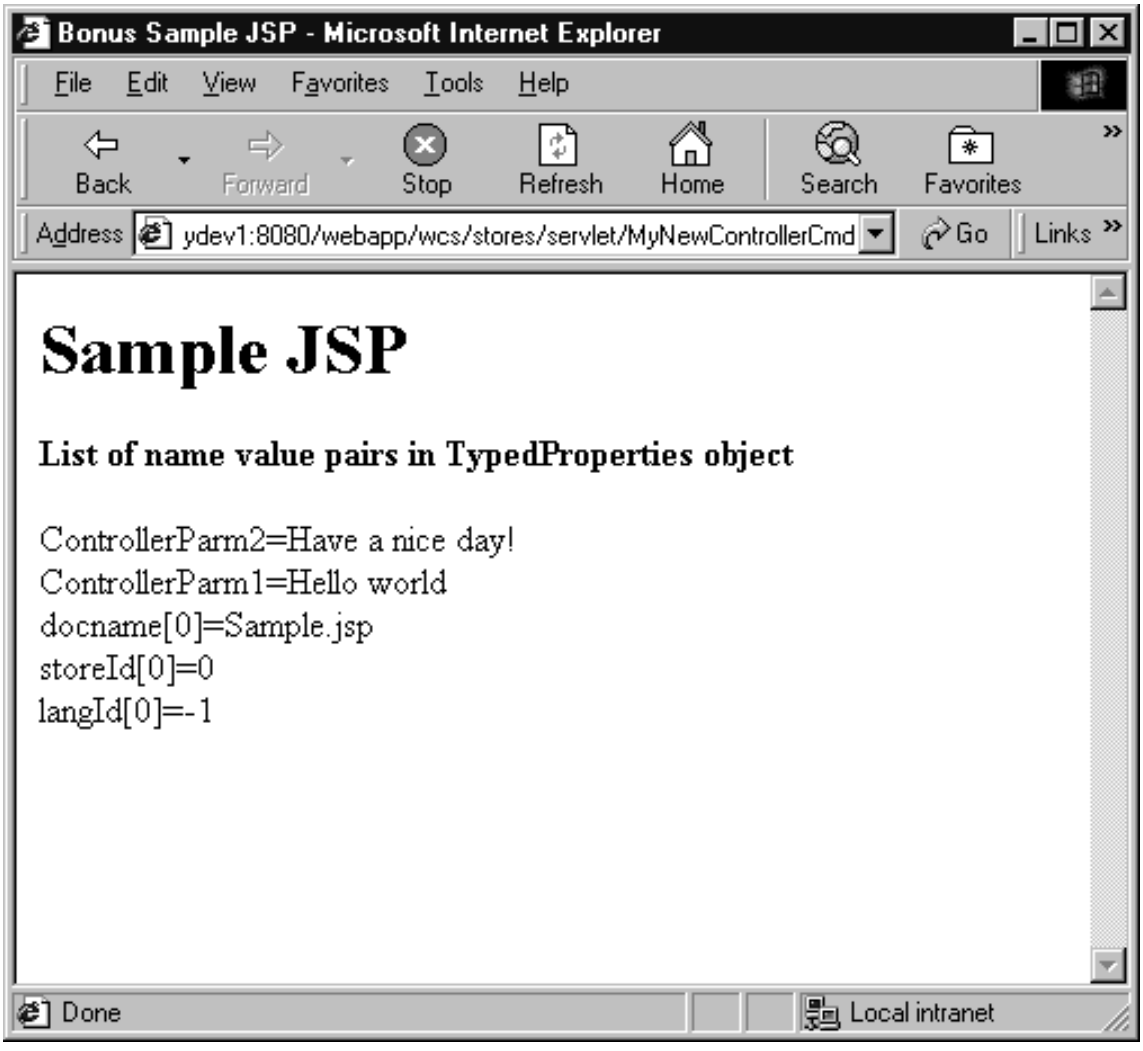


Figura 32.

7. Creare una versione del codice nello stato corrente. Ridenominare la versione `mySample 1.3 Completed`. Per ulteriori dettagli sul codice della versione, fare riferimento a 12 a pagina 215.

Modifica del metodo validateParameters

In realtà, il metodo validateParameters presente al momento nel nuovo comando è uno stub del comando, il cui codice è:

```
public void validateParameters() throws ECException {
     }
```

In questa sezione, verrà aggiunto il parametro personalizzato per la verifica del comando e quindi i parametri verranno inviati alla maschera JSP.

Quando si modifica il metodo validateParameters, si aggiungono alla classe nuovi campi utilizzati per i parametri.

Creazione di nuovi campi: Quando si aggiungono nuovi campi a un'interfaccia oppure a una classe esistente, è possibile utilizzare la SmartGuide per la creazione dei campi di VisualAge per Java. Questa sezione descrive le procedure generiche da eseguire per la creazione di un nuovo campo. Utilizzare queste informazioni nelle seguenti sezioni del supporto didattico quando è necessario aggiungere nuovi campi alle interfacce e alle classi.

Per creare un nuovo campo, effettuare le operazioni descritte:

1. Fare clic con il pulsante destro del mouse sull'interfaccia o la classe alla quale si sta aggiungendo il nuovo campo e selezionare **Aggiungi > Campo**.
   Viene visualizzata la SmartGuide per la creazione dei campi.
2. Nel campo del **Nome campo**, immettere il nome del nuovo campo.
3. Per specificare il tipo di campo, eseguire le operazioni descritte:
   - Dall'elenco a discesa del **Tipo campo** selezionare il tipo di campo.
   - Se il tipo di campo richiesto non è specificato nell'elenco, fare clic su **Sfoglia**. Nel campo **Modello**, immettere il nome (o il nome parziale) del tipo di campo e fare clic su **OK**.

   Nota: Nei supporti didattici, ogni volta che si specifica il tipo di campo `String`, viene richiamato dal pacchetto `java.lang`.
4. Nel campo del **valore iniziale**, immettere il valore iniziale del campo. Per i valori iniziali che corrispondono a string, assicurarsi di racchiudere il valore tra le doppie virgolette (" ").
5. Per il valore **Modifier accesso**, selezionare il pulsante di scelta appropriato (public, protected, nessuna o private), se necessario.
6. Selezionare la casella **Accedi mediante i metodi getter e setter** se si desidera generare i metodi getter e setter per il campo. Se si seleziona questa opzione, è necessario specificare anche le proprietà per i metodi getter e setter nel modo seguente:
   - Per il metodo getter, selezionare un'opzione tra public, protected, private o nessuna.
   - Per il metodo setter, selezionare un'opzione tra public, protected, private o nessuna.
7. Fare clic su **Fine**.

Per modificare il metodo validateParameters, effettuare le seguenti operazioni:

1. È necessario creare due nuovi campi nella classe MyNewControllerCommandImpl. Il primo viene utilizzato per le stringhe di immissione e il secondo per i numeri interi di immissione. Per creare questi campi, eseguire le operazioni descritte:
   a. Espandere il pacchetto **com.ibm.commerce.sample.commands**.
   b. Selezionare la classe **MyNewControllerCmdImpl**.
   c. Aggiungere un campo alla classe, utilizzando i seguenti valori. Per le procedure dettagliate su come creare un nuovo campo, fare riferimento a "Creazione di nuovi campi" a pagina 220.

Nome attributo	**Valore**
Nome campo	`inputString`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

   d. Creare un altro campo per i numeri interi di immissioni, utilizzando i seguenti valori:

Nome attributo	**Valore**
Nome campo	`inputInteger`
Tipo campo	`Integer` **Nota:** Fare clic su **Sfoglia** e immettere `Integer`. Non selezionare `int`.
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

2. Selezionare il metodo **performExecute** nella classe MyNewControllerCmdImpl.
3. Nel codice di origine del metodo performExecute, eliminare il commento dalla terza sezione per inserire il seguente codice nel metodo:

```
// see how controller command pass in input variables to JSP

     rspProp.put("ControllerInput1", getInputString());
     rspProp.put("ControllerInput2", getInputInteger().toString());
```

Questo codice invia le variabili dal comando di controller al bean di dati. Salvare il proprio lavoro.

4. Selezionare il metodo **validateParameters** nella classe MyNewControllerCmdImpl.
5. Eliminare il commento dalla sezione 1 nel codice di origine di validateParameters, per inserire il seguente codice nel metodo:

```
// uncomment to check parameters

      TypedProperty prop = getRequestProperties();

// retrieve required parameters
 //
 try {
          setInputString(prop.getString("input1"));
 } catch (ParameterNotFoundException e) {
          throw new ECApplicationException(
               ECMessage._ERR_CMD_MISSING_PARAM,
               "MyControllerCmdImpl","validateParameters",
                ECMessageHelper.generateMsgParms(e.getParamName()));
 }

 // retrieve optional Integer
 // set input2 = 0 if no input value
 //
 setInputInteger(prop.getInteger("input2", 0));
```

Salvare il proprio lavoro.
La sezione di codice precedente controlla i due parametri di immissione. Il blocco try controlla la presenza del primo parametro; nel caso in cui questo parametro non venga rilevato, viene lanciata un'eccezione. Poiché il secondo parametro è facoltativo, questo codice imposta il relativo valore su `0` se il parametro manca o appartiene a un tipo errato.

6. Verificare il comando nel modo seguente:
   a. Assicurarsi che il server dei nomi permanenti, il server EJB e il motore servlet siano in esecuzione.
   b. Dal proprio browser, immettere i seguenti URL:
      - Caso 1: Parametro mancante:
        Immettere

        ```
        http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd
        ```

        Poiché non è stato inoltrato alcun parametro al comando, viene visualizzato un errore di applicazione generico ad indicare che

manca un parametro. Il risultato è illustrato nella seguente schermata:

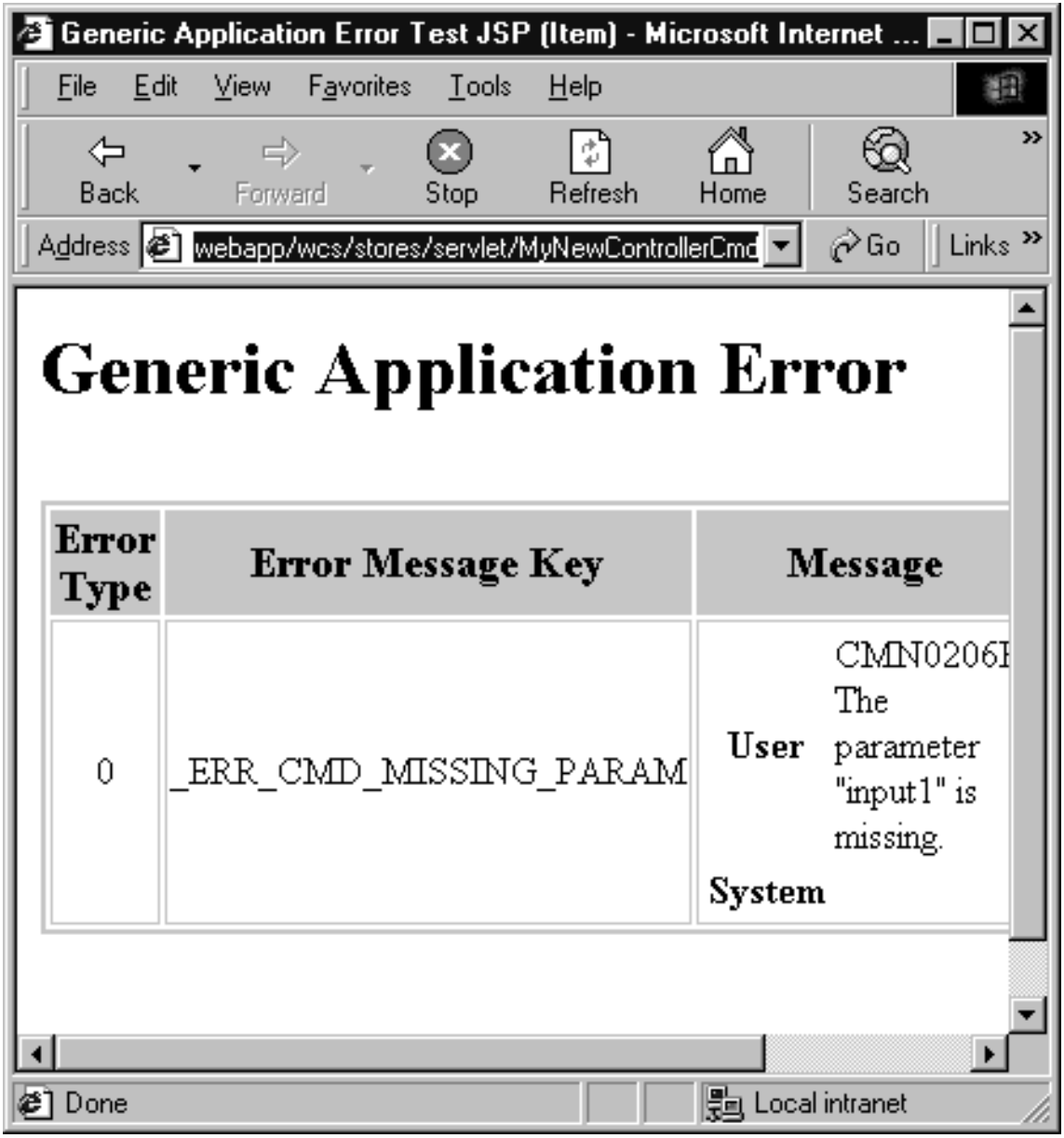


Figura 33.

Nota: Se viene visualizzato il messaggio di errore "Pagina non trovata", il motore servlet potrebbe essersi arrestato. Per

dettagli, consultare WebSphere Test Environment Centro di controllo Se viene visualizzata la pagina JSP di esempio e non la pagina di errore generico dell'applicazione, è necessario arrestare e riavviare il motore del servlet oppure ricaricare la pagina nel browser.

- Caso 2: Primo parametro valido, secondo parametro mancante: Immettere

  ```
  http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=abc
  ```

  Questo comando visualizza la pagina JSP di esempio, anche se il secondo parametro è stato omesso. La seguente immagine ne mostra il risultato. Quindi, segue una spiegazione del motivo per il quale non si è verificato alcun errore.

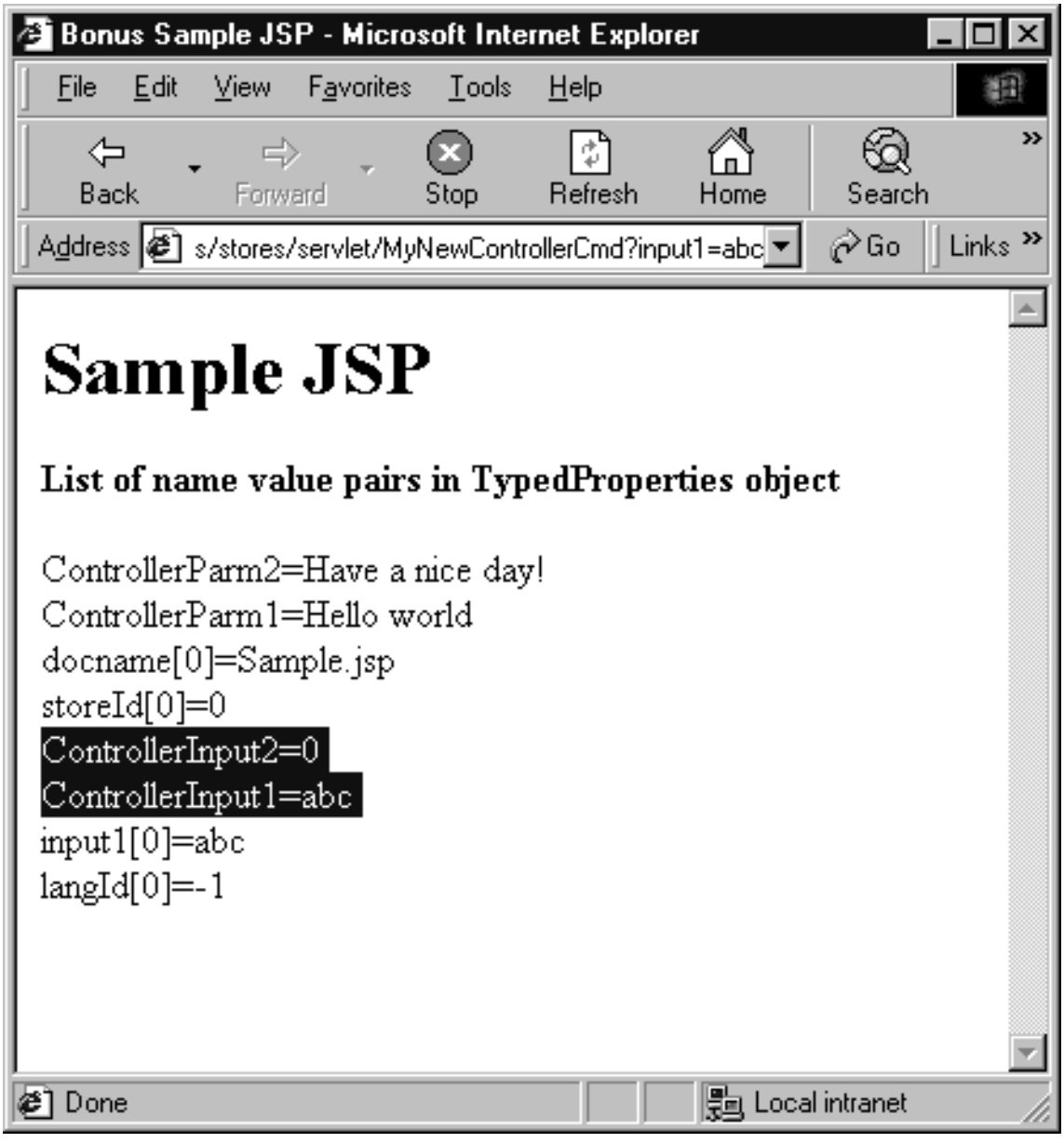


Figura 34.

Non è stato restituito alcun errore per il modo in cui è stato utilizzato il metodo `getInteger`. Infatti, la riga di codice `setInputInteger(prop.getInteger("input2", 0))` imposta il valore predefinito 0 per input2. Questo valore predefinito viene utilizzato quando il parametro risulta mancante o di tipo errato. Per eseguire

la verifica di questo parametro, modificare il codice in `setInputInteger (prop.getInteger("input2"))` e immettere di nuovo l'URL (assicurarsi di aggiornare il browser). Verrà visualizzata una pagina di errore generico dell'applicazione.

Nota: Se si esegue la verifica della modifica `setInputInteger(prop.getInteger("input2"))` nel codice utilizzato, ripristinare `setInputInteger(prop.getInteger("input2", 0))` prima di passare al testcase successivo.

- Caso 3: Primo parametro valido, secondo parametro non valido: Immettere

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=abc
```

Una volta eseguito questo comando, la pagina JSP di esempio viene visualizzata e il valore per il secondo parametro (un numero intero) è impostato sul valore predefinito `0`. Questo è il risultato del metodo `getInteger` utilizzato come descritto nella sezione precedente. Il risultato è illustrato nella seguente schermata:

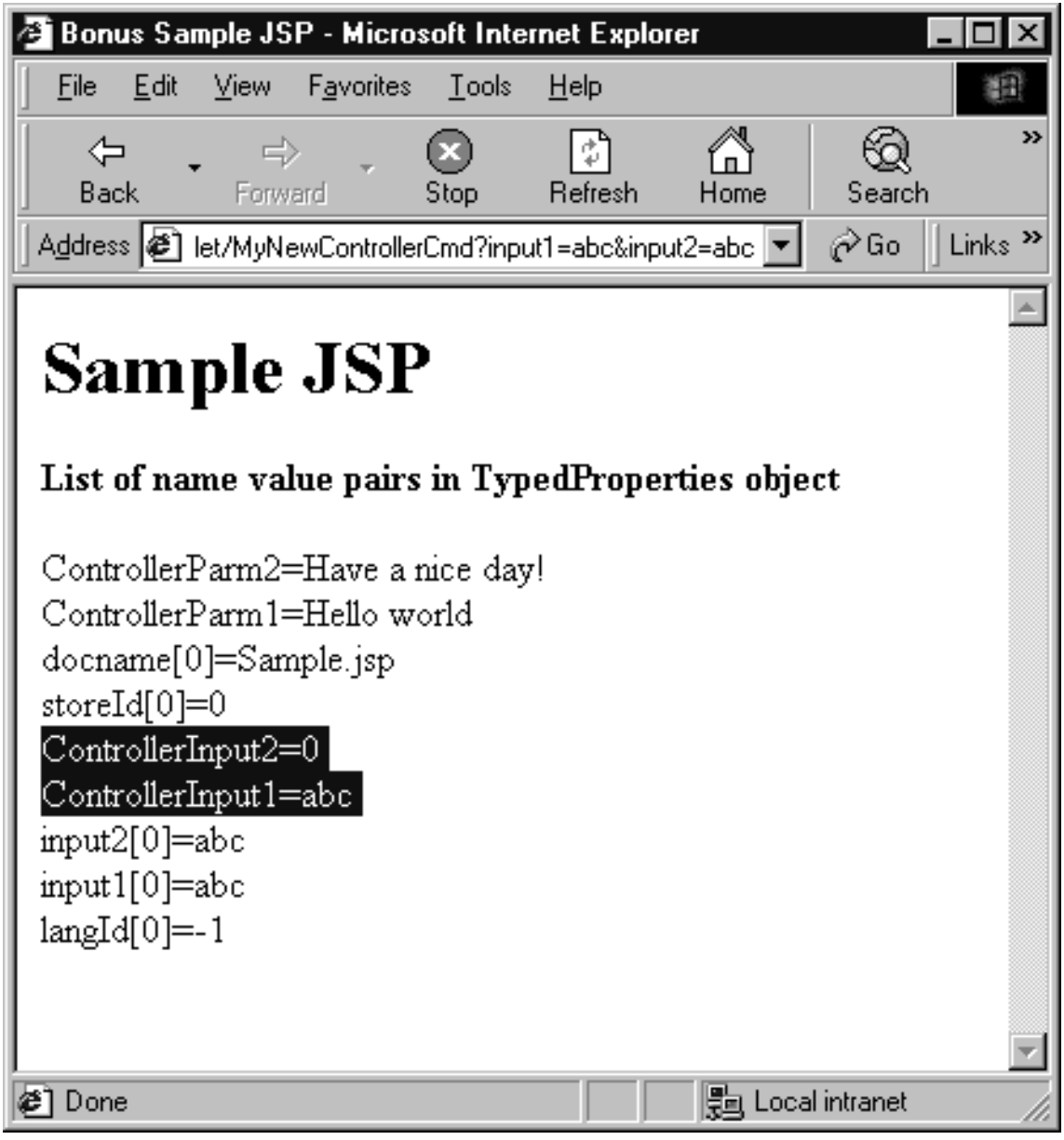


Figura 35.

- Caso 4: Due parametri validi:
  Immettere

  ```
  http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=abc&input2=1000
  ```

Questo comando visualizza la pagina JSP di esempio e i valori di immissione quando vengono immessi. Il risultato è illustrato nella seguente schermata:

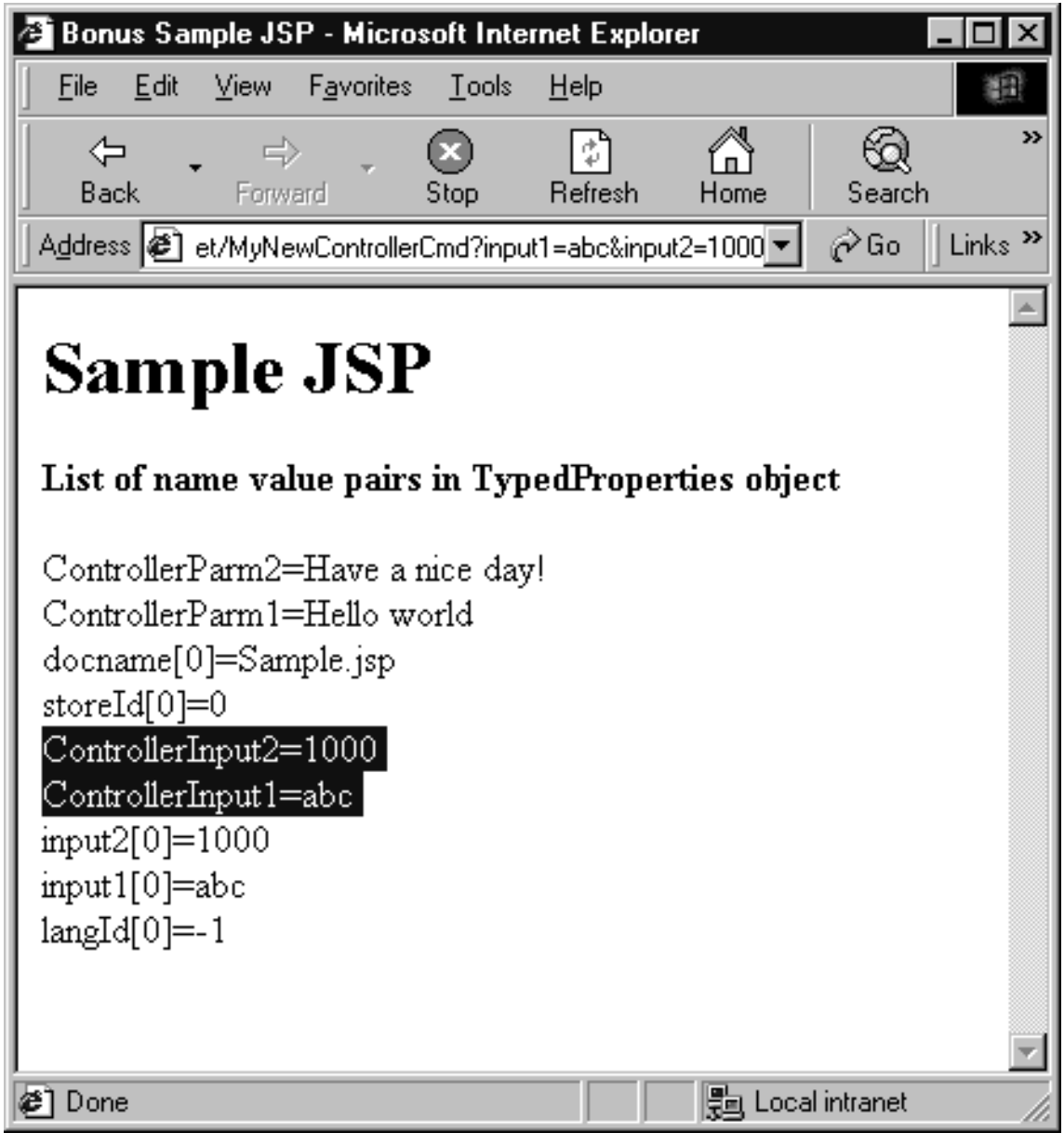


Figura 36.

7. Creare una versione del codice nello stato corrente. Ridenominare la versione `mySample 1.4 Completed`. Per ulteriori dettagli sulle procedure di assegnazione della versione al codice, fare riferimento a 12 a pagina 215.

Creazione di un comando di attività

Un comando di controller rappresenta in genere un processo aziendale o una funzione complessa. Ad esempio, tutta la logica aziendale relativa all'elaborazione di ordini si trova nel comando di controller OrderProcessCmd. È spesso possibile dividere il processo aziendale in attività specifiche più piccole. Ad esempio, all'interno del comando OrderProcessCmd, esistono diversi comandi di attività che vengono richiamati per eseguire singole unità di lavoro. Uno di essi, richiamato dal comando di controller OrderProcessCmd, è CalculateOrderTaxTotalCmd.

MyNewControllerCmdImpl non richiama al momento nessun comando di attività. Questo esercizio è diviso in due sezioni. Nella prima, viene creato un nuovo comando di attività. Nella seconda, il metodo performExecute del comando di controller viene modificato per richiamare il comando di attività.

La seguente sezione di codice mostra il metodo corrente performExecute dalla classe MyNewControllerCmdImpl, con tutti i commenti rimossi:

```
public void performExecute() throws ECException {

     super.performExecute();

      TypedProperty rspProp = new TypedProperty();
     rspProp.put("ControllerParm1", "Hello world");
     rspProp.put("ControllerParm2", "Have a nice day!");

     rspProp.put("ControllerInput1", getInputString());
     rspProp.put("ControllerInput2", getInputInteger().toString());

     rspProp.put(ECConstants.EC_VIEWTASKNAME, "SampleViewTask");
     setResponseProperties(rspProp);

}
```

Scrivere il codice del comando di attività: Questa sezione illustra come scrivere un nuovo comando di attività. La creazione di un comando di attività completamente nuovo prevede la creazione di un'interfaccia e di una classe di implementazione. Quando si crea un comando, l'interfaccia dovrebbe aprire `com.ibm.commerce.commands.TaskCommand`. La classe di implementazione deve estendere `com.ibm.commerce.command.TaskCommandImpl`.

Dopo aver completato questo esercizio, verrà visualizzato un nuovo comando chiamato *MyNewTaskCmd*. Questo comando viene utilizzato da tutti i negozi e ciascuno di essi utilizza la stessa implementazione del comando.

In questa parte del supporto didattico, è possibile aggiungere campi e metodi all'interfaccia del nuovo comando delle attività. Dopo la creazione di nuovi campi per la classe MyNewControllerCmdImpl, provare a creare i campi per questa interfaccia e se sono necessari ulteriori dettagli, fare riferimento a "Creazione di nuovi campi" a pagina 220.

Creazione di metodi: Questa sezione descrive le procedure generiche per l'aggiunta di metodi alle classi e alle interfacce esistenti. Leggere le istruzioni descritte di seguito, alle quali occorre fare riferimento quando il supporto didattico richiede di creare nuovi metodi.

Per creare un nuovo metodo, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sulla classe o sull'interfaccia alla quale si sta aggiungendo il metodo e selezionare **Aggiungi > Metodo**. Viene visualizzata la SmartGuide per la creazione dei metodi.
2. Assicurarsi che l'opzione **Crea nuovo metodo** sia selezionata e fare clic su **Avanti**.
3. Nel campo relativo al **nome del metodo**, immettere il nome per il nuovo metodo.
4. Specificare il tipo di risultato del metodo eseguendo una delle procedure descritte di seguito:
   - Dall'elenco a discesa **Tipo di risultato** selezionare il tipo di risultato appropriato. Ad esempio, selezionare `String`.
   - Se il tipo di risultato non è compreso nell'elenco, fare clic su **Sfoglia**. Quindi, nel campo **Modello**, immettere il tipo di risultato e fare clic su **OK**.

   Nota: Nei supporti didattici, ogni volta che si specifica il tipo di risultato `String`, viene richiamato dal pacchetto `java.lang`.
5. Se il metodo prevede parametri, fare clic su **Aggiungi**. Nella finestra Parametri, specificare il nome del parametro e altre informazioni richieste, quindi fare clic su **Aggiungi**. Dopo aver aggiunto i parametri, fare clic su **Chiudi**.
6. Fare clic su **Avanti**.
   Viene visualizzata la finestra degli attributi.
7. Se il metodo restituisce degli errori, fare clic su **Aggiungi** nella finestra degli attributi. Nel campo **Modello**, immettere il nome dell'eccezione, quindi fare clic su **Aggiungi**. Una volta aggiunte tutte le eccezioni, fare clic su **Chiudi**.
8. Fare clic su **Fine**.
   Viene generato il codice per il metodo.

Per creare il comando MyNewTaskCmd, eseguire le operazioni descritte:

1. Espandere il pacchetto **com.ibm.commerce.sample.commands**.
2. Fare clic con il pulsante destro del mouse sull'interfaccia **MyNewTaskCmd** e selezionare **Aggiungi > Campo**.
3. Creare un campo che specifica la classe di implementazione predefinita da utilizzare con l'interfaccia, utilizzando la SmartGuide per la creazione del campo. Utilizzare i valori nella seguente tabella. Per ulteriori dettagli sulla creazione di un campo, fare riferimento a "Creazione di nuovi campi" a pagina 220.

Nome attributo	Valore
Nome campo	`defaultCommandClassName`
Tipo campo	`String`
Valore iniziale	`"com.ibm.commerce.sample.commands.MyNewTaskCmdImpl"`

Il codice per il campo generato viene visualizzato nel modo seguente:

```
java.lang.String defaultCommandClassName =
     "com.ibm.commerce.sample.commands.MyNewTaskCmdImpl";
```

Dal momento che viene utilizzata la stessa classe di implementazione in tutto il sito e nessuna proprietà predefinita viene inoltrata al comando, è possibile specificare l'implementazione predefinita nel codice. Se esiste un comando con più implementazioni o con proprietà predefinite (contenute nella tabella CMDREG), è necessario registrare il comando nella tabella CMDREG per creare corrispondenze tra l'interfaccia e la classe di implementazione.

4. Aggiungere i nuovi metodi all'interfaccia MyNewTaskCmd, eseguendo le procedure descritte di seguito:
   a. Fare clic con il pulsante destro del mouse sull'interfaccia **MyNewTaskCmd** e selezionare **Aggiungi > Metodo**. Creare nuovi metodi utilizzando i valori specificati nelle procedure seguenti, utilizzando la SmartGuide per la creazione di metodi. Per ulteriori dettagli sulla creazione di metodi, fare riferimento a "Creazione di metodi" a pagina 230.
   b. Creare un nuovo metodo che richiama un totale di punti di bonus del cliente, utilizzando i seguenti valori:

Nome attributo	Valore
Nome metodo	`getOldBonusPoint`
Tipo di risultato	`String`
Parametri	nessuna
Eccezioni	nessuna

Nota: È possibile che venga visualizzato un errore in cui si indica che il metodo abstract ereditato creato non è stato implementato nella classe di implementazione MyNewTaskCmdImpl. L'errore verrà corretto in una fase successiva.

c. Creare un nuovo metodo che richiama l'emissione dell'ID utente dal comando delle attività. Per la creazione di questo metodo, utilizzare i seguenti valori:

Nome attributo	**Valore**
Nome metodo	`getTask_output_userId`
Tipo di risultato	`String`
Parametri	nessuna
Eccezioni	nessuna

d. Creare un nuovo metodo che richiama un valore di emissione dal comando delle attività. Per la creazione di questo metodo, utilizzare i seguenti valori:

Nome attributo	**Valore**
Nome metodo	`getTask_output1`
Tipo di risultato	`String`
Parametri	nessuna
Eccezioni	nessuna

e. Creare un nuovo metodo che imposta il primo valore di immissione sul comando delle attività. Per la creazione di questo metodo, utilizzare i seguenti valori:

Nome attributo	**Valore**
Nome metodo	`setTask_input1`
Tipo di risultato	`void`
Nome parametro	`newTask_input1`
Tipo di riferimento	`String`
Eccezioni	nessuna

f. Creare un nuovo metodo che imposta il secondo valore di immissione sul comando delle attività. Per la creazione di questo metodo, utilizzare i seguenti valori:

Nome attributo	**Valore**
Nome metodo	`setTask_input2`

Nome attributo	Valore
Tipo di risultato	`void`
Nome parametro	`newTask_input2`
Tipo primitivo	`int`
Eccezioni	nessuna

5. Aggiungere i nuovi campi alla classe MyNewTaskCmdImpl nel modo seguente:
   a. Fare clic con il pulsante destro del mouse sulla classe **MyNewTaskCmdImpl** e selezionare **Aggiungi > Campo**. Creare nuovi campi utilizzando i valori specificati nelle procedure seguenti, utilizzando la SmartGuide per la creazione dei campi. Se risulta necessario conoscere altre informazioni sulla creazione di nuovi campi, fare riferimento a "Creazione di nuovi campi" a pagina 220.
   b. Creare un nuovo campo nella classe di implementazione, utilizzando i seguenti valori:

Nome attributo	Valore
Nome campo	`task_input1`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

   c. Creare un nuovo campo nella classe di implementazione, utilizzando i seguenti valori:

Nome attributo	Valore
Nome campo	`task_input2`
Tipo campo	`int`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`

Nome attributo	Valore
Setter	`public`

d. Creare un nuovo campo nella classe di implementazione, utilizzando i seguenti valori:

Nome attributo	Valore
Nome campo	`task_output_userId`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

e. Creare un nuovo campo nella classe di implementazione, utilizzando i seguenti valori:

Nome attributo	Valore
Nome campo	`oldBonusPoint`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

f. Creare un nuovo campo nella classe di implementazione, utilizzando i seguenti valori:

Nome attributo	Valore
Nome campo	`task_output1`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.

Nome attributo	Valore
Accedi mediante i metodi getter e setter	checked
Getter	public
Setter	public

6. Selezionare il metodo **performExecute** della classe **MyNewTaskCmdImpl** per visualizzare il codice di origine.
7. Nel codice di origine, eliminare il commento dalla prima sezione per inserire il seguente codice nel metodo:

```
// modify the task_input1 and see it in the NVP list

     setTask_output1( "Hello ! " + getTask_input1() );
```

   Salvare il proprio lavoro.
   La sezione di codice precedente rende disponibili i nuovi attributi come emissione del comando.

Richiamare il comando di attività: Dopo aver creato il comando di attività, è necessario richiamarlo dal comando di controller. Nei passi successivi viene descritta la procedura per modificare il comando di controller.

1. Nello spazio di lavoro, selezionare il metodo **performExecute** della classe **MyNewControllerCmdImpl**.
2. Nel pannello di origine, eliminare il commento dalla quarta sezione per richiamare il comando di attività. In tal modo, viene inserito il seguente codice nel metodo:

```
// see how controller command call a task command

     MyNewTaskCmd cmd = null;
     try {
          cmd = (MyNewTaskCmd) CommandFactory.createCommand(
               "com.ibm.commerce.sample.commands.MyNewTaskCmd",
               getStoreId());
          // Set input parameters to task command
          cmd.setTask_input1(getInputString());
          cmd.setTask_input2(getInputInteger().intValue());
           // This is required for all commands
           cmd.setCommandContext(getCommandContext());
          // Invoke the command's performExecute method
           cmd.execute();
           // retrieve output parameter from task command

          rspProp.put("task_output1", cmd.getTask_output1());

          if (cmd.getTask_output_userId() != null) {
               rspProp.put("task_output_userId",
               cmd.getTask_output_userId());
          }
```

```
            if (cmd.getOldBonusPoint() != null) {
                 rspProp.put("task_output_oldBonusPoint",
                 cmd.getOldBonusPoint());
            }

    } catch (ECException ex) {
            // throw the exception as is
             throw (ECException) ex;
    }
```

Salvare il proprio lavoro.
La sezione di codice precedente crea un nuovo comando di attività utilizzando i comandi standard. Quindi, imposta il contesto di comando, richiama ″perform execute″ e i parametri di emissione del comando di attività.

3. Eseguire il test del comando immettendo l'URL per il comando di controller, come segue:

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

La pagina JSP di esempio visualizza l'elenco delle coppie di valori dei nomi nell'oggetto di richiesta, incluso i valori di emissione delle attività. Il risultato verrà visualizzato nel modo seguente:

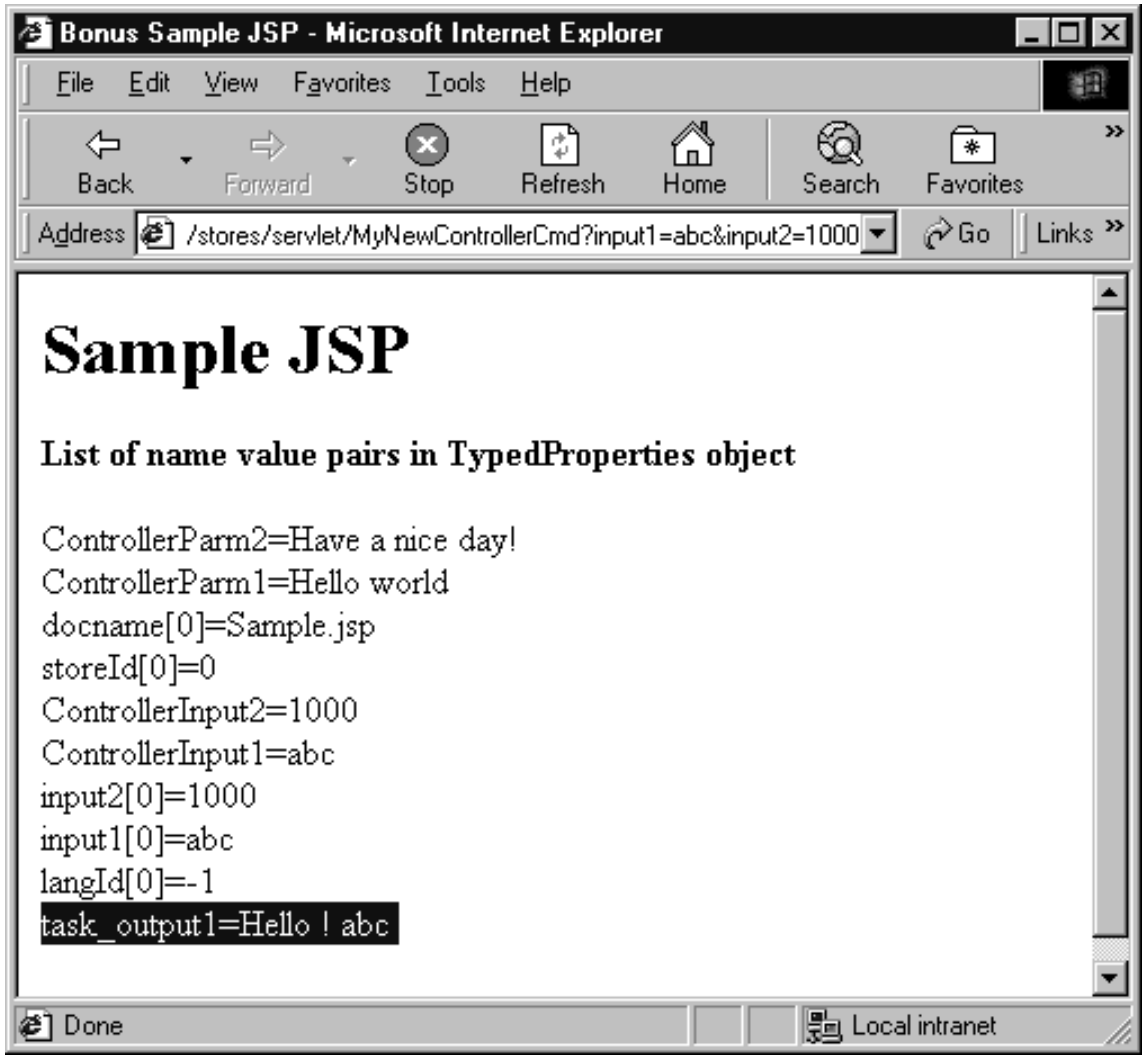


Figura 37.

4. Creare una versione del codice nello stato corrente. Ridenominare la versione `mySample 1.5 Completed`. Per ulteriori dettagli su come assegnare la versione al proprio codice, fare riferimento a 12 a pagina 215.

Convalida di un ID utente

La fase successiva è la modifica del comando di attività allo scopo di utilizzare UserRegistryAccessBean per confermare se il valore immesso dall'utente appartiene a un utente registrato. Oltre a modificare il comando di attività, è necessario modificare anche DataBeanSampleBean.

Modifica di MyNewTaskCmdImpl per la convalida dell'ID utente: È necessario modificare il metodo performExecute nella classe MyNewTaskCmdImpl in modo che convaliderà l'ID dell'utente utilizzando UserRegistryAccessBean. Per aggiungere questa nuova funzione al metodo performExecute, effettuare le seguenti operazioni:

1. Nello spazio di lavoro, selezionare il metodo **performExecute** della classe **MyNewTaskCmdImpl**.
2. Nel pannello di origine, eliminare il commento dalla seconda sezione del metodo performExecute. In tal modo, viene inserito il seguente codice nel metodo:

```
// use UserRegistryAccessBean to check member reference number

     String refNum;

     UserRegistryAccessBean rrb = new UserRegistryAccessBean();

     try {
          rrb = rrb.findByUserLogonId(getTask_input1());
          refNum = rrb.getUserId();

     } catch (javax.ejb.FinderException e) {

     return;

     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
          this.getClass().getName(), "performExecute");
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
          this.getClass().getName(), "performExecute");
     } catch (javax.ejb.CreateException e) {
          throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
          this.getClass().getName(), "performExecute");
     }

      setTask_output_userId(refNum);
```

   Salvare il proprio lavoro.

Modifica di DataBeanSampleBean: È necessario modificare DataBeanSampleBean per utilizzare i parametri di immissione predefiniti. A tale scopo, effettuare le seguenti operazioni:

1. Nello spazio di lavoro, espandere il pacchetto **com.ibm.commerce.sample.databeans**.
2. Aggiungere nuovi campi al bean di dati nel modo seguente:
   a. Fare clic con il pulsante destro del mouse sulla classe **DataBeanSampleBean** e selezionare **Aggiungi > Campo**. Creare nuovi campi utilizzando i valori specificati nelle procedure seguenti, utilizzando la SmartGuide per la creazione dei campi. Se risulta

necessario conoscere altre informazioni sulla creazione dei campi, fare riferimento a "Creazione di nuovi campi" a pagina 220.

b. Aggiungere un nuovo campo al bean di dati utilizzando i seguenti valori:

Nome attributo	**Valore**
Nome campo	`input1`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

c. Aggiungere un nuovo campo al bean di dati utilizzando i seguenti valori:

Nome attributo	**Valore**
Nome campo	`input2`
Tipo campo	`int`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

d. Aggiungere un nuovo campo al bean di dati utilizzando i seguenti valori:

Nome attributo	**Valore**
Nome campo	`task_output_userId`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`

Nome attributo	Valore
Getter	public
Setter	public

3. Selezionare il metodo **populate** della classe DataBeanSampleBean per visualizzarne il codice di origine.
4. Nel codice di origine, eliminare il commento dalla sezione 1A e 1B. In tal modo, viene inserito il seguente codice nel metodo:

```
//// Section 1A //////////
// set additional data fields

try {
   setInput1( getRequestProperties().getString("input1"));
   setInput2( getRequestProperties().getIntValue("input2", 0)  );
   setTask_output_userId(getRequestProperties().getString
     ("task_output_userId"));

//// End of Section 1A /////

   //// Section 2 ////////////
   /*
   // instantiate databean to BonusAccessBean
   setTask_output_oldBonusPoint(getRequestProperties().getString(
      "task_output_oldBonusPoint"));
   */
   //// End of Section 2 ////

//// Section 1A ///////////
// instantiate databean to BonusAccessBean and
// set additional data fields

   }
catch (ParameterNotFoundException e){}

//// End of Section 1B ////
```

   Salvare il proprio lavoro.
   La sezione di codice precedente acquisisce i valori del campo di dati dal comando di controller, utilizzando il metodo getRequestProperties.

Modifica di Sample.jsp per la convalida dell'ID utente: La maschera JSP corrente deve essere modificata per poter eseguire la convalida dell'ID utente. Per modificare il file Sample.jsp, eseguire le procedure descritte:

1. Aprire i file `Sample.jsp` e `Sample_All.jsp` con un editor di testo.
2. Copiare la terza sezione del codice da Sample_All.jsp a Sample.jsp (la parte compresa tra `<!-- SECTION 3 -->` e `<!-- END OF SECTION 3 -->`). Il codice riportato di seguito viene inserito nella maschera JSP:

```
<!-- SECTION 3 -->

<B>Your first input is &lt; <%=testBean.getInput1()%> &gt;</B>

<%
String userId = testBean.getTask_output_userId();

if (userId == null) {

%>

<UL>
     <LI> This is not a registered user id.
</UL>

<%

   } else {

%>

<B>
<UL>
     <LI> 'lt; <%=testBean.getInput1()%> &gt; is a registed user id.
     <LI> The member reference number of this user is <%=userId%>.
</UL>
</B>

<%

}

%>

<B>Your second input is < <%=testBean.getInput2()%> ></B> <P>

<!-- END OF SECTION 3 -->
```

Salvare il file Sample.jsp.

3. Aprire il browser e immettere il seguente URL:

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

Il browser visualizza il file JSP di esempio che indica che il valore di input1 non è un ID utente valido, come illustrato nella seguente schermata:

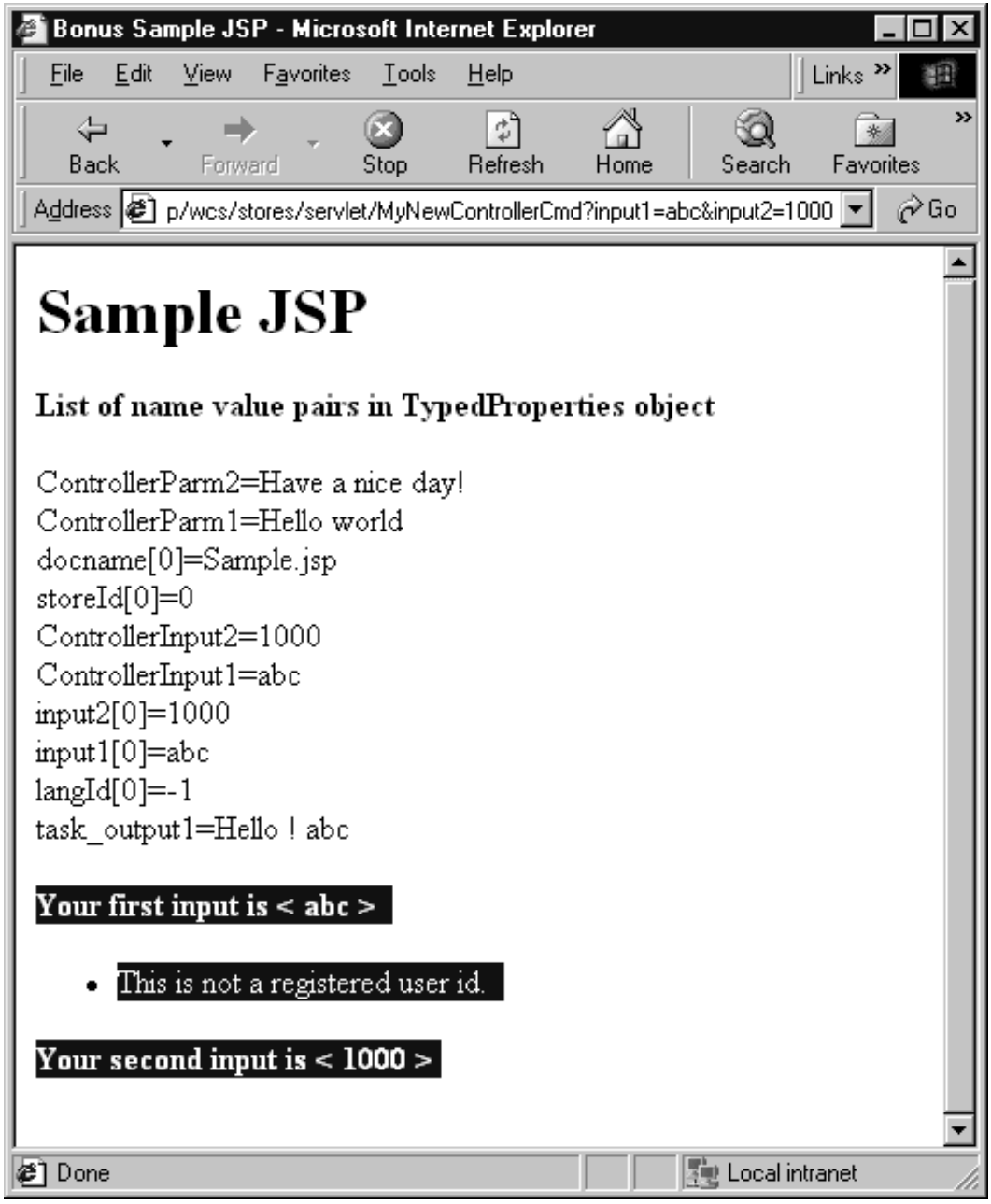


Figura 38.

4. Per visualizzare il risultato di un valore di input1 valido come ID utente, immettere il seguente URL:

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=wcsadmin&input2=1000
```

Il risultato è illustrato nella seguente schermata:

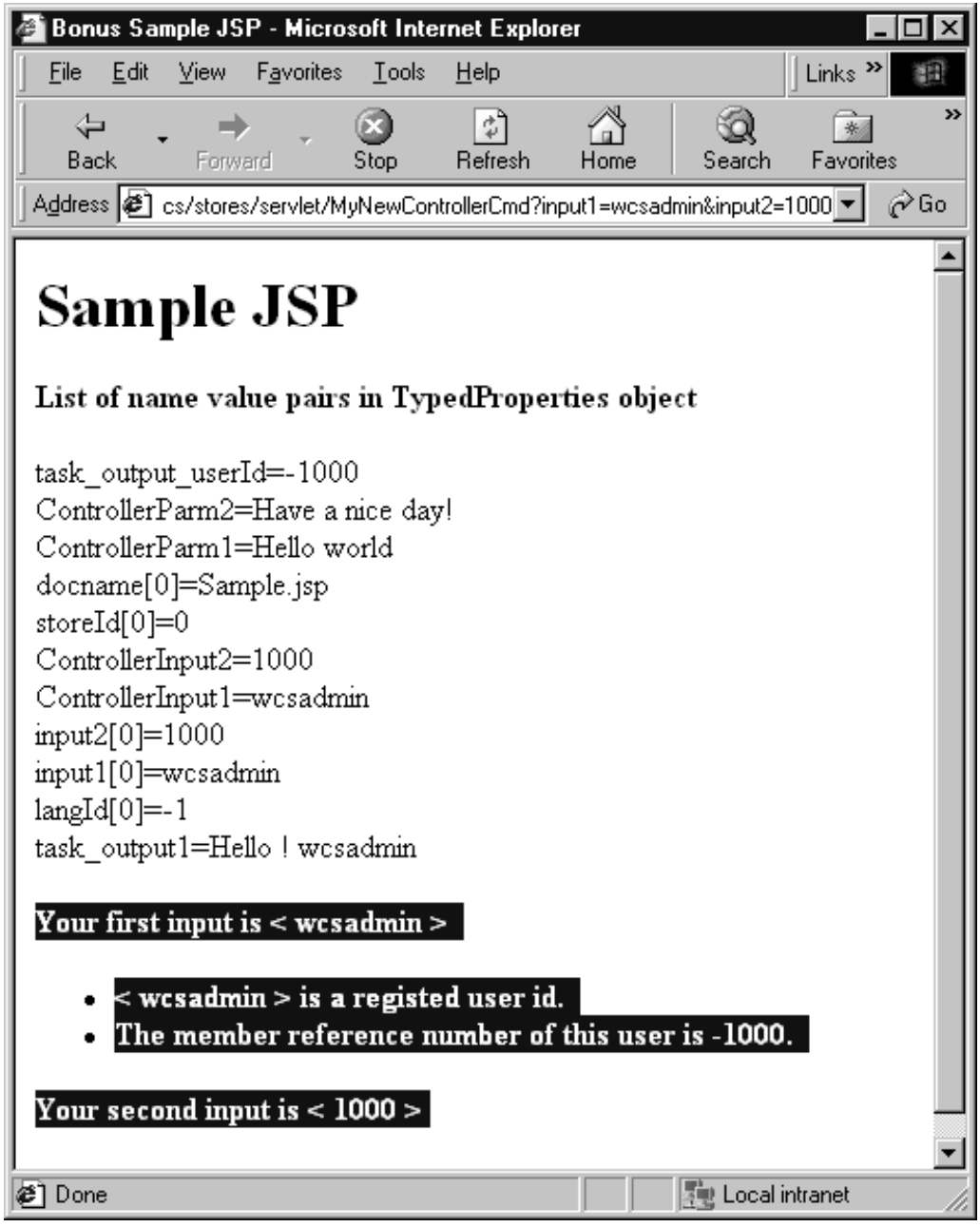


Figura 39.

5. Creare una versione del codice nello stato corrente. Ridenominare la versione `mySample 1.6 Completed`. Per ulteriori dettagli su come assegnare la versione al proprio codice, fare riferimento a 12 a pagina 215.

Creazione di un nuovo bean entità

Questa sezione descrive l'utilizzo di VisualAge per Java per la creazione di un nuovo bean entità. Nel contesto creato per questo esempio, uno dei requisiti aziendali richiede il conteggio di una serie di bonus per ciascun utente nell'applicazione commerciale. Poiché lo schema del database di WebSphere Commerce non contiene queste informazioni, è necessario creare una nuova tabella di database che le contenga. In base al modello di programmazione di WebSphere Commerce, una volta creata la tabella del database è necessario creare un bean entità (che è un bean enterprise) per accedere ai dati.

In questo esempio è possibile utilizzare una SmartGuide di VisualAge per Java per creare un bean entità.

Creazione di una nuova tabella database

Prima di procedere alla creazione del bean entità, creare prima la nuova tabella database. La tabella da creare si chiama Bonus.

DB2 Se si utilizza un database DB2, effettuare le seguenti operazioni per creare la tabella:

1. Aprire il Centro di comandiDB2 (**Start > Programmi > IBM DB2 > Centro di comandi**) e fare clic sulla scheda **Script**.
2. Nella finestra Script, immettere:

   ```
   connect to nome_database;
   CREATE TABLE Bonus (MEMBERID BIGINT NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (id_utente) on delete cascade)
   ```

   dove *nome_database* è il nome del proprio database. Fare clic sull'icona Execute.
   A questo punto la tabella Bonus viene creata.

   Nota: Prima di creare la tabella Bonus, immettere il seguente comando (se questo esempio è stato già utilizzato con questo database):

   ```
   drop table Bonus
   ```

Oracle Se si utilizza un database Oracle, effettuare le seguenti operazioni per creare la tabella:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password**, immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.

5. Nella finestra SQL Plus, immettere la seguente istruzione SQL:

```
CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   e premere Invia per eseguire l'istruzione SQL. Viene così creata la tabella BONUS.

   Nota: Prima di creare la tabella Bonus, immettere il seguente comando (se questo esempio è stato già utilizzato con questo database):

   ```
   drop table Bonus;
   ```

6. Immettere quanto segue per confermare le modifiche al database:

   ```
   commit;
   ```

   e premere Invia per eseguire l'istruzione SQL.

Creazione del bean entità BonusBean

Una volta creato il database, è possibile avviare il processo di creazione del nuovo bean entità. Le fasi successive prevedono l'utilizzo di VisualAge per Java. Per creare il nuovo bean entità, effettuare le seguenti operazioni:

1. Creare un gruppo EJB. Un gruppo EJB è un gruppo logico che consente di organizzare i bean enterprise. In un gruppo EJB è possibile eseguire operazioni globali che possono essere applicate su tutti i bean enterprise che risiedono in quel gruppo. Ad esempio, se si seleziona un gruppo EJB da esportare in un file JAR EJB, tutti i bean enterprise nel gruppo saranno esportati.
   In questo supporto didattico, un gruppo EJB viene creato per organizzare tutti i bean enterprise correlati alla personalizzazione della propria tabella Bonus.
   Per creare un proprio gruppo EJB, eseguire le procedure descritte:
   a. Nello spazio di lavoro, fare clic sulla scheda **EJB**.
   b. Dal menu **EJB**, selezionare **Add > EJB Group**.
      Si apre la Smartguide per l'aggiunta di un gruppo EJB.
   c. Nel campo **Progetto**, immettere `_WCSamplesEntityBeansProject`.

      Nota: Il codice del bean entità deve essere memorizzato all'interno di un proprio progetto per motivi di applicazione.

   d. Nel campo **Crea un nuovo EJB denominato**, immettere `WCSSamplesEntityBeans` e fare clic su **Fine**.
2. Creare un nuovo bean entità.
   Per creare un nuovo bean entità, eseguire le procedure descritte di seguito:
   a. Nel pannello Bean Enterprise, Fare clic con il pulsante destro del mouse sul gruppo EJB **WCSSamplesEntityBeans** e selezionare

Aggiungi > Bean Enterprise.
Si apre la SmartGuide per la creazione del bean enterprise.

b. Immettere le seguenti informazioni:

Attributo	Valore
Nome bean	`Bonus` **Nota:** In base alle convenzioni di denominazione, il bean entità deve essere chiamato con lo stesso nome della tabella a cui si accede.
Tipo bean	`Bean entità con campi CMP (container-managed persistence)`
Crea una nuova classe bean	`enable`
Progetto	`_WCSamplesEntityBeansProject`
Pacchetto	`com.ibm.commerce.sample.objects`
Nome classe	`BonusBean`
Superclasse	`com.ibm.commerce.base.objects.ECEntityBean`

e selezionare **Avanti**.

c. Fare clic sul pulsante **Aggiungi** accanto alla casella di testo **Aggiungi campi al bean** per aggiungere un campo per la colonna MEMBERID nella tabella BONUS.
Si apre la SmartGuide per la creazione dei campi CMP.

d. Immettere le seguenti informazioni:

Attributo	Valore
Nome campo	memberId
Tipo campo	Long **Nota:** È necessario utilizzare il tipo di dati *Long*, non *long*.
Campo chiave	enable

e fare clic su **Fine**.

e. Fare clic di nuovo su **Aggiungi** per aggiungere un campo per la colonna BONUSPOINT nella tabella BONUS.

f. Creare un altro campo con le seguenti informazioni

Attributo	Valore
Nome campo	bonusPoint
Tipo campo	Integer **Nota:** È necessario utilizzare il tipo di dati *Integer*, non *int*.

Attributo	Valore
Accedi mediante i metodi getter e setter	enable
Promuovi metodi getter e setter per l'interfaccia remota	enable

quindi fare clic su **Fine** nella finestra Create CMP.

g. Proteggere questo bean enterprise mediante il controllo accessi, effettuando le seguenti operazioni:
   1) Fare clic su **Aggiungi** accanto a **Quali interfacce deve estendere l'interfaccia remota?**.
   2) Immettere `com.ibm.commerce.security.Protectable` nel campo **Modello** e fare clic su **Aggiungi**. Fare clic su **Chiudi** per chiudere la finestra.

h. Fare di nuovo clic su **Fine**.

L'entità Bonus viene creata come un bean enterprise.

3. Impostare il livello di isolamento del bean entità, eseguendo le procedure indicate:
   a. Fare clic con il pulsante destro del mouse sul bean **Bonus** e selezionare **Proprietà**.
   b. Dall'elenco a discesa per il **livello di isolamento**, selezionare **TRANSACTION_READ_COMMITTED** e fare clic su**OK**.
4. Quando si crea un nuovo bean enterprise, VisualAge per Java genera un campo EntityContext e i corrispondenti metodi getEntityContext() e setEntityContext(EntityContext) nel bean. Seguendo il modello di programmazione di WebSphere Commerce, il nuovo bean estende la classe com.ibm.commerce.base.objects.ECEntityBean e ECEntityBean fornisce l'implementazione di questo campo e di questi metodi. Dal momento che EntityContext, getEntityContext e setEntityContext non devono essere sovrascritti, a questo punto è necessario eliminare il campo e i metodi dal bean.
   Per eliminare il campo EntityContext e i rispettivi metodi get e set, effettuare le seguenti operazioni:
   a. Nel pannello Tipi, selezionare la classe **BonusBean**.
      Il pannello Membri visualizza i campi e metodi per questa classe.

      Nota: Se questo pannello non viene visualizzato, fare clic sull'icona della classe/interfaccia nel pannello Proprietà. Si apre il pannello Tipi.

   b. Nel pannello Membri, effettuare le seguenti operazioni:
      1) Fare clic con il pulsante destro del mouse sul campo **entityContext** e selezionare **Elimina**.

2) Fare clic con il pulsante destro del mouse sul metodo **getEntityContext()** e selezionare **Elimina**.
3) Fare clic con il pulsante destro del mouse sul metodo **setEntityContext(EntityContext)** e selezionare **Elimina**.

c. Salvare il proprio lavoro (Ctrl+S).

5. Aggiungere un nuovo metodo getMemberId al bean enterprise, effettuando le seguenti operazioni:
   a. Fare clic con il pulsante destro del mouse sulla classe **BonusBean** e selezionare **Aggiungi > Metodo**.
   Si apre la SmartGuide per la creazione dei metodi.
   b. Creare il nuovo metodo utilizzando i valori specificati nella seguente tabella. Per informazioni più dettagliate sulla creazione di un nuovo metodo, consultare"Creazione di metodi" a pagina 230.

Tabella 11.

Nome attributo	**Valore**
Nome metodo	`getMemberId`
Tipo di risultato	`Long`
Parametri	nessuna
Eccezioni	nessuna

   c. Quando si crea il nuovo metodo, visualizzare il codice di origine.
   d. Per impostazione predefinita, il metodo contiene il seguente codice:
   ```
   return null;
   ```
   Modificare questo codice in
   ```
   return memberId;
   ```
   e. Aggiungere il nuovo metodo all'interfaccia remota facendo clic con il pulsante destro del mouse sul metodo **getMemberId** e selezionando **Aggiungi a > Interfaccia remota EJB**.

6. Aggiungere nuovi campi FinderHelper in BonusBeanFinderHelper. Questa interfaccia contiene una clausola di ricerca che corrisponde a un metodo FinderHelper creato nella fase successiva. Per aggiungere i campi FinderHelper, effettuare le seguenti operazioni:
   a. Nel pannello Tipi, fare clic sull'interfaccia **BonusBeanFinderHelper**.
   b. Modificare il codice nel pannello dell'origine nel seguente modo:
   ```
   public interface BonusBeanFinderHelper {
       public static final String
                       findByMemberIdWhereClause = " (MEMBERID = ?) ";
           }
   ```

Nota: La sintassi di "WhereClause" è fondamentale; deve corrispondere al nome del metodo utilizzato per il metodo FinderHelper. In questo caso, "findByMemberId" in `findByMemberIdWhereClause` corrisponde esattamente al nome del metodo creato nella fase successiva (`findByMemberId`).

7. Aggiungere nuovi metodi FinderHelper nell'interfaccia BonusHome
   Per aggiungere nuovi metodi FinderHelper, effettuare le seguenti operazioni:
   a. Nel pannello Tipi, fare clic con il pulsante destro del mouse sull'interfaccia **BonusHome** e selezionare **Aggiungi > Metodo**. Viene visualizzata la SmartGuide per la creazione dei metodi.
   b. Selezionare **Crea nuovo metodo** quindi **Avanti**.
   c. Nel campo **Nome metodo**, immettere `findByMemberId`.
   d. Nel campo **Tipo di risultato**, immettere `Bonus`.
   e. Fare clic su **Aggiungi** accanto a **Quali parametri deve avere questo metodo?**
      Si apre la finestra Parametri.
   f. Nel campo **Nome**, immettere `argMemberId`.
   g. Selezionare **Tipi di riferimento** e immettere `Long`. Fare clic su **Aggiungi** quindi su **Chiudi**.
   h. Fare clic su **Avanti**.
   i. Fare clic su **Aggiungi** accanto a **Quali eccezioni devono essere attivate da questo metodo?** , immettere `RemoteException` nel campo **Modello** e fare clic su **Aggiungi**. L'eccezione `java.rmi.RemoteException` viene aggiunta nella finestra Attributi. (La finestra Attributi può essere posizionata dietro la finestra Eccezioni.)
   j. Nel campo **Modello** della finestra Eccezioni immettere `FinderException`, fare clic su **Aggiungi** e, quindi, su **Chiudi**. L'eccezione `javax.ejb.FinderException` viene elencata nella finestra Attributi.
   k. Fare clic su **Fine**.
8. Aggiungere un nuovo metodo `ejbCreate` al gruppo EJB. Questo metodo viene promosso all'interfaccia home in modo che sia disponibile in un bean di accesso generato. Per creare questo metodo, effettuare le seguenti operazioni:
   a. Selezionare la classe **BonusBean** nel pannello Tipi.
   b. Fare clic su **ejbCreate(Long)** nel pannello Membri.
   c. Modificare il codice e renderlo uguale al seguente:

```
public void ejbCreate(java.lang.Long argMemberId,
  Integer argBonusPoint)
  throws javax.ejb.CreateException, java.rmi.RemoteException {
            _initLinks();
```

```
// All CMP fields should be initialized here.
 memberId=argMemberId;
        bonusPoint=argBonusPoint;
}
```

d. Salvare il codice; VisualAge per Java crea un nuovo metodo che si chiama **ejbCreate(Long, Integer)**, nel pannello Membri.

e. Fare clic con il pulsante destro del mouse sul metodo **ejbCreate(Long)** originale e selezionare **Elimina**.

9. Aggiungere il nuovo metodo all'interfaccia home. Il metodo sarà così disponibile nella classe del bean di accesso. A tale scopo, effettuare le seguenti operazioni:

   a. Fare clic con il pulsante destro del mouse sul metodo **ejbCreate(Long, Integer)** nella classe BonusBean e selezionare **Aggiungi a > Interfaccia home EJB**.

10. Aggiornare il metodo getOwner effettuando le seguenti operazioni:

    a. Selezionare la classe **BonusBean** nel pannello Tipi.

    b. Fare clic sul metodo **getOwner()** nel pannello Membri.

       Nota: Se è stata selezionata l'opzione per visualizzare i metodi ereditati, sullo schermo verranno riportati due metodi getOwner(). Uno di questi due metodi è quello ereditato dalla classe ECEntityBean e non deve essere selezionato. Assicurarsi di selezionare il metodo getOwner specifico della classe BonusBean.

    c. Di seguito viene riportato il codice di origine del metodo getOwner:

```
public Long getOwner()
   throws Exception, java.rmi.RemoteException {
   {
      return null;
   }
```

       È necessario modificare il valore restituito dal metodo: la parte da modificare viene indicata in grassetto nella seguente sezione di codice:

```
public Long getOwner()
   throws Exception, java.rmi.RemoteException {
   {
      return getMemberId();
   }
```

       Salvare il proprio lavoro.

    d. Fare clic sul metodo **fulfills(Long, String)** nel pannello Membri.

       Nota: Se è stata selezionata l'opzione per visualizzare i metodi ereditati, sullo schermo verranno riportati due metodi fulfills(Long, String,). Uno di questi due metodi è quello

ereditato dalla classe ECEntityBean e non deve essere selezionato. Assicurarsi di selezionare il metodo fulfills(Long, String,) specifico della classe BonusBean.

e. Di seguito viene riportato il codice di origine del metodo fulfills(Long, String,):

```
public boolean fulfills(Long member, String relationship)
  throws Exception, java.rmi.RemoteException {
  {
      return false;
  }
```

È necessario specificare la relazione che l'utente deve soddisfare. A tale scopo, modificare la parte indicata in grassetto nella seguente sezione di codice:

```
public boolean fulfills(Long member, String relationship)
  throws Exception, java.rmi.RemoteException
  {

      if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }      return false;
  }
```

Salvare il proprio lavoro.

11. Mettere in corrispondenza la tabella di database BONUS a BonusBean. La prima fase della procedura di corrispondenza dello schema di database all'entità BonusBean prevede l'utilizzo degli strumenti di VisualAge per Java per creare lo schema del database. Per creare lo schema, effettuare le seguenti operazioni:
    a. Nella finestra Spazio di lavoro, dal menu **EJB** selezionare **Apri in > Schemi di database**.
    b. Dal menu **Schemi**, selezionare **Importa/Esporta schemi > Importa schema dal database**.
    Si apre la finestra delle informazioni richieste.
    c. Nel campo per il **nome dello schema**, immettere `WCSSamples` e fare clic su **OK**.
    Si apre la finestra delle informazioni di collegamento al database.
    d. Immettere le seguenti informazioni:

Attributo	**Valore** DB2 **DB2**	Oracle **Valore Oracle**
Tipo collegamento	`COM.ibm.db2.jdbc.app.DB2Driver`	`Oracle.jdbc.driver.OracleDriver`
Origine dati	`jdbc:db2:`*`wcs_db_name`*	`jdbc:oracle:thin:@`*`nomehost`*`:`*`porta`*`:`*`SID`*

Attributo	Valore DB2 DB2	Oracle Valore Oracle
Nome utente	*wcs_db_user_name*	*wcs_db_user_name*
Password	*wcs_db_password*	*wcs_db_password*

con i valori sostituiti come riportato di seguito:

- DB2 *nome_database_wcs* è il nome del database WebSphere Commerce
- Oracle *nomehost* è il nome host Oracle
- Oracle *porta* è il numero di porta del database Oracle (ad esempio, 1521).
- *nome_utente_db_wcs* è il nome utente per il database.
- *password_db_wcs* è la password del database.

Fare clic su **OK**.
Si apre la finestra per la selezione delle tabelle.

e. Dall'elenco **Qualificatori**, selezionare il qualificatore del database (può trattarsi del nome utente database o del nome macchina) e fare clic su **Crea elenco tabelle**. Viene caricato un elenco delle tabelle di database disponibili.

f. Selezionare **Bonus** dal pannello Tabelle e fare clic su **OK**. Attendere alcuni secondi.

g. Nel browser dello schema, fare clic sull'ultima tabella aggiunta per assicurarsi che vengano visualizzate entrambe le colonne della tabella.

h. Fare clic con il pulsante destro del mouse sulla tabella **Bonus** e selezionare **Edita tabella**.
Viene visualizzata la finestra Editor tabelle.

i. Rimuovere qualunque voce dal campo **Qualificatore**. È buona norma rimuovere le informazioni sul qualificatore, in modo che il codice possa essere distribuito su altre macchine che utilizzano un database differente.

j. Oracle Modificare i tipi di dati delle colonne come segue:

1) Selezionare la colonna **MEMBERID**, quindi fare clic su **Modifica**. Dall'elenco a discesa Tipo, selezionare **BIGINT** e fare clic su **OK**.

2) Selezionare la colonna **BONUSPOINT**, quindi fare clic su **Modifica**. Dall'elenco a discesa Tipo, selezionare **INTEGER** e fare clic su **OK**.

k. Fare clic su **OK** per uscire dall'editor delle tabelle.

l. Dal menu **Schemi**, selezionare **Salva schema**.
Si apre la finestra Salva schema.

m. Immettere le seguenti informazioni:

Attributo	Valore
Progetto	_WCSamplesEntityBeansProject
Pacchetto	com.ibm.commerce.sample.objects
Nome classe	WCSSamplesSchema

quindi fare clic su **Fine** e chiudere il browser dello schema.

12. Una volta creato lo schema, è possibile creare le relative corrispondenze. Per creare la corrispondenza tra la tabella BONUS e l'entità BonusBean, effettuare le seguenti operazioni:
    a. Dal menu **EJB**, selezionare **Apri in > Mappe schemi**. Si apre la finestra Browser corrispondenze.
    b. Nel Browser corrispondenze, dal menu **Mappe datastore** selezionare **Nuova corrispondenza gruppi EJB**.
    Si apre la finestra Nuova corrispondenza datastore.
    c. Immettere le seguenti informazioni:

Attributo	Valore
Nome	WCS Samples
Gruppo EJB	WCSSamplesEntityBeans
Schema	WCSSamples

e fare clic su **OK**.

    d. Nel pannello Mappe datastore, fare clic su **WCS Samples**.
    e. Nel pannello Classi permanenti, fare clic su **Bonus**.
    f. Dal menu **Mappe di tabelle**, selezionare **Nuova corrispondenza di tabella > Aggiungi corrispondenza tabella senza eredità**.
    g. Da elenco a discesa **Tabella** selezionare **Bonus** e fare clic su **OK**.
    h. Nel pannello corrispondenze tabella, selezionare **Bonus**, fare clic con il pulsante destro del mouse e selezionare **Edit property maps**.
    Si apre l'editor delle corrispondenze delle proprietà.
    i. Impostare gli attributi nel seguente modo:

Attributo classe	Tipo corrispondenza	Colonna tabella
memberId	Simple	MEMBERID
bonusPoint	Simple	BONUSPOINT

e fare clic su **OK**.

    j. Dal menu **Mappe datastore**, selezionare **Salva corrispondenza datastore**.
    Si apre la finestra relativa.

k. Immettere le seguenti informazioni:

Attributo	Valore
Progetto	`_WCSamplesEntityBeansProject`
Pacchetto	`com.ibm.commerce.sample.objects`
Nome classe	`WCSSamplesMap`

e fare clic su **Fine**, quindi chiudere la finestra Browser corrispondenze.

13. Una volta creata l'entità BonusBean e una corretta corrispondenza dello schema, è necessario creare un bean di accesso per il bean entità. Tale bean di accesso facilita alle applicazioni l'accesso alle informazioni contenute nel bean entità Bonus. Per generare tale bean di accesso, è possibile utilizzare gli strumenti di VisualAge per Java, in base all'entità già creata (in particolare, il bean di accesso utilizzerà solo i metodi inviati all'interfaccia remota). Per creare il bean di accesso per il bean entità Bonus, effettuare le seguenti operazioni:
    a. Nello spazio di lavoro, dopo aver selezionato la scheda EJB, fare clic con il pulsante destro del mouse sul bean enterprise **Bonus** e selezionare **Aggiungi > Bean di accesso**.
    Si apre la finestra SmartGuide per la creazione del bean di accesso (questa operazione può richiedere vari secondi).
    b. Assicurarsi di aver immesso le seguenti informazioni:

Attributo	Valore
Gruppo EJB	`WCSSamplesEntityBeans`
Bean enterprise	`Bonus`
Nome bean di accesso	`BonusAccessBean`
Tipo bean di accesso	`CopyHelper per un bean entità`

e selezionare **Avanti**.

    c. Dall'elenco a discesa **Seleziona metodo locale per constructor senza argomenti** , selezionare **findByMemberId(Long)**.
    d. Per **init_argMemberId**, (nella colonna delle proprietà iniziali), impostare il **programma di conversione** su `com.ibm.commerce.base.objects.WCSStringConverter` e fare clic su **Avanti**.
    e. Per bonusPoint, assicurarsi che **CopyHelper** sia selezionato, impostare il valore del **programma di conversione** su `com.ibm.commerce.base.objects.WCSStringConverter` e fare clic su **Fine**.

    Nota: Non è necessario fare alcuna modifica nel campo memberId.

f. Fare clic su **OK** quando viene visualizzato il messaggio“Generazione codice completata”.

Per visualizzare il nuovo codice, selezionare la scheda**Progetti**, espandere il progetto **_WCSamplesEntityBeansProject**, quindi espandere **com.ibm.commerce.sample.objects**. Viene visualizzata nel pacchetto una nuova classe denominata `BonusAccessBean`.

14. Nella fase successiva è prevista la generazione del codice di distribuzione.
Il programma di utilità per la generazione del codice analizza i bean per assicurarsi che le specifiche EJB di Sun Microsystems vengano soddisfatte e che le regole specifiche del server EJB vengano rispettate. Inoltre, per ciascun bean enterprise selezionato, lo strumento di generazione del codice crea le implementazioni home e EJBObject (remota) e le classi di implementazione per le interfacce home e remota, oltre alle classi persister e finder JDBC per i bean CMP. Inoltre, genera anche l'ORB Java, gli stub e le classi tie necessarie per l'accesso RMI su IIOP, nonché gli stub per le interfacce home e remota.
Se è stato selezionato un gruppo EJB contenente un bean enterprise CMP, o se è stato selezionato un singolo bean enterprise CMP, vengono generati anche i seguenti elementi:
    - Una stringa per la creazione della tabella viene generata nella classe persister.
    - Un'implementazione Persister che stabilisce le corrispondenze da e verso la tabella

    Per generare il codice di distribuzione, effettuare le seguenti operazioni:
    a. Dopo aver selezionato la scheda EJB, dal pannello Bean Enterprise fare clic con il pulsante destro del mouse sul bean enterprise **Bonus** e selezionare **Generate Deployed Code**. La generazione del codice dura alcuni minuti.
15. Prima di verificare il bean enterprise Bonus, è necessario creare un nuovo server EJB che contiene tutti i gruppi EJB di WebSphere Commerce e il nuovo gruppo EJB WCSSamplesEntityBeans. Questi gruppi devono essere in esecuzione sullo stesso server per poter gestire gli ambiti delle transazioni. Una volta creato il nuovo server, è necessario riavviarlo.
Per creare e avviare il nuovo server EJB, eseguire le procedure descritte di seguito:
    a. Assicurarsi che tutti i gruppi EJB siano catalogati.
    b. Nel pannello Bean enterprise, selezionare tutti i gruppi EJB di WebSphere Commerce e il nuovo gruppo EJB (vale a dire, selezionare tutti i gruppi EJB che iniziano con `WCS`).

c. Una volta selezionati i gruppi EJB, fare clic con il pulsante destro del mouse e selezionare **Aggiungi a > Configurazione server** .
Si apre la finestra Configurazione server EJB (questa operazione può richiedere vari secondi).

d. Fare clic con il pulsante destro del mouse sul server EJB creato {ad esempio **Server EJB (server2)**}, selezionare **Proprietà** e inserire i seguenti valori:

Attributo	**Valore** DB2 **DB2**	Oracle **Valore Oracle**
Origine dati	*nome_istanza* WebSphere Commerce DB2 DataSource	*nome_istanza* WebSphere Commerce Oracle DataSource
Tipo collegamento	<DataSource>	<DataSource>
Nome utente	*wcs_db_user_name*	*wcs_db_user_name*
Password	*wcs_db_password*	*wcs_db_password*
Timeout transazione	1200	1200
Timeout inattività transazione	600000	600000

e fare clic su **OK**.

Nota: Il valore per l'origine dati deve corrispondere all'origine dati specificata nel file *nome_istanza*.xml.

A seconda dell'hardware installato sulla macchina di sviluppo (ad esempio, la velocità del processore) può essere necessario aumentare i valori per le proprietà del server EJB **Transaction Timeout** e **Transaction Inactivity**.

e. Se WebSphere Test Environment è in esecuzione, interromperlo insieme al server dei nomi permanenti e altri server EJB, come descritto in Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347.

f. Avviare il server dei nomi permanenti, come descritto in "Avvio e interruzione del server dei nomi permanenti" a pagina 347.

g. Avviare il nuovo server EJB, come descritto in "Avvio e interruzione del server EJB" a pagina 348.

16. Una volta avviato il server EJB, è possibile avviare il client del test. Utilizzando il client del test, è possibile creare un nuovo record all'interno del database. Per avviare il client del test e creare tale record, effettuare le seguenti operazioni:

a. Dopo aver selezionato la scheda EJB, fare clic con il pulsante destro del mouse sul bean enterprise **Bonus** e selezionare **Esegui client di test**.

b. Nella finestra Ricerca EJB, fare clic su **Ricerca**.
Si apre la finestra Bonus.

c. Fare clic su **create(Long, Integer)**. Nel pannello Dettagli,immettere quanto segue:

Attributo	Valore
Long	`–1000`
Integer	`100`

Fare clic sull'icona per il richiamo nella finestra del client del test EJB.

Nota: Il primo attributo deve corrispondere al memberId per gli utenti registrati. È possibile individuare i valori di memberId nella tabella USERS.

17. Verificare (effettuando direttamente una query al database) che il record del database sia stato creato correttamente.

DB2 Se si utilizza un database DB2, effettuare le seguenti operazioni:

a. Aprire il Centro comandi DB2 (**Start > Programmi > IBM DB2 > Centro comandi**)

b. Selezionare la scheda **Interattiva**.

c. Immettere `connect to` *`wcs_database_name`* e fare clic sull'icona Esegui.

d. Immettere `SELECT * FROM Bonus` e fare clic sull'icona Esegui.
Verranno restituiti i seguenti valori:

Colonna	Valore
MEMBERID	`-1000`
BONUSPOINT	`100`

Il record sopra riportato è stato creato dal client del test EJB.

Oracle Se si utilizza un database Oracle, effettuare quanto segue:

a. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle - OraHome81> Application Development > SQL Plus**).

b. Nel campo **Nome utente** immettere il nome utente Oracle.

c. Nel campo **Password** immettere la password per Oracle.

d. Nel campo **Stringa host** immettere la propria stringa di collegamento.

e. Nella finestra SQL Plus, immettere:

```
select * from BONUS;
```

e fare clic su Invia per eseguire l'istruzione SQL, che dovrà restituire i seguenti risultati:

Colonna	Valore
MEMBERID	-1000
BONUSPOINT	100

18. Utilizzare il client del test per verificare che il bean enterprise Bonus abbia effettuato correttamente l'accesso al record del database, completando le seguenti operazioni:
    a. Nella finestra Bonus, selezionare la scheda **Home**.
    b. Nel pannello Metodi, fare clic su **findByMemberId(Long)**.
    c. Nel campo **Long**, immettere –1000 e fare clic sull'icona Richiama.
    d. Con la scheda **Remoto** selezionata, espandere **Metodi**, fare clic su **getBonusPoint** e fare clic sull'icona Richiama.
    Il pannello Details mostra un valore intero pari a 100.
    e. Chiudere le finestra Bonus e EJB Test Client.

Integrazione del bean entità Bonus con MyNewControllerCmd

Nella precedente sezione, è stata descritta l'esecuzione del test del nuovo bean entità Bonus utilizzando il client del test generato con VisualAge per Java. In questa sezione, verrà esposta la modalità di integrazione del bean entità Bonus con la logica MyNewControllerCmd. Dopo aver aggiornato il codice Java, la maschera Sample.jsp viene aggiornata per creare un'interfaccia che consenta di aggiornare il saldo dei punti bonus dell'acquirente.

L'integrazione del bean entità Bonus prevedere l'esecuzione delle seguenti fasi di livello elevato:

1. Modifica del metodo performExecute della classe MyNewTaskCmdImpl per calcolare i nuovi punti di bonus e salvarli nella tabella BONUS.
2. Aggiunta di un metodo getResources alla classe MyNewControllerCmdImpl per restituire un elenco di risorse utilizzate dal comando. Questo metodo viene incluso per il controllo accessi.
3. Creazione di una nuova politica controllo accessi per le nuove risorse.
4. Modifica di DataBeanSampleBean per l'estensione dal bean di accesso per il bean entità Bonus. Se il bean di dati si estende dal bean di accesso, tutti gli attributi dal bean di accesso vengono ereditati dal bean di dati.
5. Modifica dei metodi in DataBeanSampleBean.
6. Modifica del classpath per il motore servlet in WebSphere Test Environment per includere il nuovo _WCSamplesEntityBeansProject.
7. Modifica della maschera Sample.jsp per consentire agli utenti di immettere i punti di bonus e di visualizzare i risultati.

Modifica di MyNewTaskCmdImpl per il calcolo dei punti di bonus:
MyNewTaskCmdImpl viene utilizzato come punto di integrazione per il bean entità Bonus e MyNewControllerCmd (in quanto MyNewControllerCmd richiama MyNewTaskCmd).

Per modificare MyNewTaskCmdImpl in modo che esegua il calcolo dei punti di bonus, effettuare le seguente operazioni:

1. Nella finestra Spazio di lavoro di VisualAge per Java, espandere il progetto **_WCSamples**.
2. Espandere il pacchetto **com.ibm.commerce.sample.commands**, quindi selezionare la classe **MyNewTaskCmdImpl** per visualizzarne il codice di origine.
3. Eliminare il commento dalla seguente istruzione di importazione:

   ```
   import com.ibm.commerce.sample.objects.*;
   ```

   Salvare (**Ctrl + S**).
4. Selezionare il metodo **performExecute** della classe **MyNewTaskCmdImpl**.
5. Nel codice di origine del metodo performExecute, eliminare il commento dalla terza sezione. In tal modo, viene inserito il seguente codice nel metodo:

   ```
   // use BonusAccessBean to update new bonus point

   String newBonusPoint = null;
   BonusAccessBean bb = new BonusAccessBean();
   try {
        if (refNum != null) {
             bb.setInit_argMemberId(refNum);
             bb.refreshCopyHelper();
             oldBonusPoint = bb.getBonusPoint();
        }
   } catch (javax.ejb.FinderException e) {
        try {
             bb = new BonusAccessBean(new Long(refNum),new Integer(0));
             oldBonusPoint = "0";
        } catch (javax.ejb.CreateException ec) {
             throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
                  this.getClass().getName(), "performExecute");
        } catch (javax.naming.NamingException ec) {
             throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
                  this.getClass().getName(), "performExecute");
        } catch (java.rmi.RemoteException ec) {
             throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
                  this.getClass().getName(), "performExecute");
        }
   } catch (javax.naming.NamingException e) {
        throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
             this.getClass().getName(), "performExecute");
   } catch (java.rmi.RemoteException e) {
   ```

```
     throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
          this.getClass().getName(), "performExecute");
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
          this.getClass().getName(), "performExecute");
}


try {
     if (oldBonusPoint != null) {
          int newBP =  Integer.parseInt(oldBonusPoint) + getTask_input2();
          newBonusPoint = Integer.toString( newBP );
          bb.setBonusPoint( newBonusPoint )   ;
          newBonusPoint=bb.getBonusPoint();
          bb.commitCopyHelper();
     }
} catch (javax.ejb.FinderException e) {
     throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
          this.getClass().getName(), "performExecute");
} catch (javax.naming.NamingException e) {
     throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
          this.getClass().getName(), "performExecute");
} catch (java.rmi.RemoteException e) {
     throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
          this.getClass().getName(), "performExecute");
} catch (javax.ejb.CreateException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
          this.getClass().getName(), "performExecute");
}
```

Salvare il proprio lavoro.

Aggiunta del metodo getResources alla classe MyNewControllerCmdImpl: In questa sezione, verrà aggiunto un nuovo metodo getResources alla classe MyNewControllerCmdImpl. Questo metodo restituisce un elenco di risorse che il comando utilizza durante l'elaborazione. Questo metodo viene incluso per implementare il controllo accessi a livello di risorsa.

Per aggiungere il metodo getResources, effettuare le seguenti operazioni:

1. Con la scheda **Progetti** selezionata, espandere il progetto **_WCSamples**.
2. Espandere il pacchetto **com.ibm.commerce.sample.commands**.
3. Selezionare la classe **MyNewControllerCmdImpl** per visualizzarne il codice di origine.
4. Nel codice di origine, eliminare i commenti relativi alla sezione controllo accessi. Questa sezione viene riportata nel seguente codice:

```
public AccessVector getResources() throws ECException {

   // use UserRegistryAccessBean to check member reference number

   String refNum;
   String methodName="getResources";
```

```
        com.ibm.commerce.user.objects.UserRegistryAccessBean rrb =
           new com.ibm.commerce.user.objects.UserRegistryAccessBean();

        try {
           rrb = rrb.findByUserLogonId(getInputString());
           refNum = rrb.getUserId();

        }
        catch (javax.ejb.FinderException e) {

           throw new ECSystemException(ECMessage._ERR_BAD_USER_NAME,
              this.getClass().getName(),methodName);

        }
        catch (javax.naming.NamingException e) {
           throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
              this.getClass().getName(), methodName);
        }
        catch (java.rmi.RemoteException e) {
           throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
              this.getClass().getName(), methodName);
        }
        catch (javax.ejb.CreateException e) {
           throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
           this.getClass().getName(), methodName);
        }


        //find the Bonus bean for this user
        String newBonusPoint = null;
        com.ibm.commerce.sample.objects.BonusAccessBean bb =
           new com.ibm.commerce.sample.objects.BonusAccessBean();
        try {
           if (refNum != null) {
              bb.setInit_argMemberId(refNum);
              bb.refreshCopyHelper();
           }
        }
      catch (javax.ejb.FinderException e) {

        // The user doesn't have a Bonus object so return the container that
        // will hold the bonus object when it's created

        return new AccessVector(rrb);

        }
        catch (javax.naming.NamingException e) {
           throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
              this.getClass().getName(), methodName);
        }

        catch (java.rmi.RemoteException e) {
           throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
           this.getClass().getName(), methodName);
```

```
        }

        catch (javax.ejb.CreateException e) {
           throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
           this.getClass().getName(), methodName);
        }

        return new AccessVector(bb);


    }
```

Salvare il proprio lavoro. (Ctrl+S).

Una volta salvata la sezione precedente di codice, VisualAge per Java separa i campi e gli accessi da questa visualizzazione. I campi e gli accessi vengono ora visualizzati nella classe MyNewControllerCmdImpl mentre il metodo è indicato con una M.

Nota: Per comodità, in questo supporto didattico gli oggetti risorsa vengono creati in questo metodo getResources. In un'applicazione reale, è preferibile creare gli oggetti risorsa nel metodo validateParameters e salvarli in variabili di istanza. Pertanto, l'oggetto può essere riutilizzato dai metodi getResources e performExecute.

Impostazione della politica del controllo accessi per la nuova risorsa: Viene fornito una politica del controllo accessi di esempio. Questa politica crea i seguenti oggetti controllo accessi:

Un'azione
L'azione creata è `com.ibm.commerce.sample.commands.MyNewControllerCmd`

Un gruppo di azioni
Il gruppo di azioni creato è `MyNewControllerCmdActionGroup`. Questo gruppo di azione contiene un'unica azione; `com.ibm.commerce.sample.commands.MyNewControllerCmd`

Una categoria di risorse
La categoria di risorse creata è `com.ibm.commerce.sample.objects.BonusResourceCategory`. Questa categoria è relativa al bean entità Bonus.

Un gruppo di risorse
Il gruppo di risorse creato è `BonusResourceGroup` e contiene la precedente categoria di risorse.

Una politica
La politica creata è `AllUsersUpdateBonusResourceGroup`. Questa politica consente all'utente di eseguire l'azione MyNewControllerCmd

sul bean bonus solo se l'utente è "proprietario" dell'oggetto bonus. Ad esempio, se l'utente si è collegato come wcsadmin, l'utente può modificare i punti di bonus solo per wcsadmin.

L'impostazione della politica AllUsersUpdateBonusResourceGroup prevede il completamento delle seguenti operazioni:

1. Caricamento del file SampleACPolicy.xml utilizzando il comando acpload.
2. Caricamento della descrizione SampleACPolicy_***locale***.xml utilizzando il comando acpnlsload.
3. Aggiornamento del registro delle politiche. Questa operazione è richiesta solo nel caso in cui il motore servlet sia in esecuzione al momento del caricamento della politica di controllo accessi.

Per impostare la politica AllUsersUpdateBonusResourceGroup, effettuare le seguenti operazioni:

1. Alla richiesta comandi, passare alla seguente directory:
   unità:\WebSphere\CommerceServerDev\bin
2. Per caricare il file SampleACPolicy.xml, è necessario inoltrare il comando acpload, il cui formato è:

   ```
   acpload nome_db utente_db password_db FileXMLinput
   ```

   dove
   - *nome_db* è il nome del database
   - *utente_db* è il nome dell'utente del database
   - *password_db* è la password del database
   - *inputXMLFile* è il nome del file XML contenente la politica

   Ad esempio, è possibile inviare il seguente comando:

   ```
   acpload VAJ_Demo user password SampleACPolicy.xml
   ```
3. Per caricare la descrizione della politica, è necessario inviare il comando acpnlsload, il cui formato è:

   ```
   acpnlsload nome_db utente_db password_db inputXMLFile
   ```

   Ad esempio, è possibile inviare il seguente comando:

   ```
   acpnlsload VAJ_Demo user password SampleACPolicy_en_US.xml
   ```
4. Se il motore servlet per WebSphere Test Environment è in esecuzione, arrestarlo e riavviarlo per aggiornare il registro delle politiche.

Modifica di DataBeanSampleBean per i punti di bonus: In questa sezione viene modificato DataBeanSampleBean per estendere BonusAccessBean, con le seguenti procedure:

1. Nello spazio di lavoro, espandere il pacchetto **com.ibm.commerce.sample.databeans**.

2. Aggiungere un nuovo campo al bean di dati nel modo seguente:
   a. Fare clic con il pulsante destro del mouse sulla classe **DataBeanSampleBean** e selezionare **Aggiungi > Campo**. Creare nuovi campi come descritto nelle seguenti procedure, utilizzando la SmartGuide per la creazione dei campi. Se risulta necessario conoscere altre informazioni sulla creazione dei campi, fare riferimento a "Creazione di nuovi campi" a pagina 220.
   b. Aggiungere un nuovo campo al bean di dati utilizzando i seguenti valori:

Nome attributo	Valore
Nome campo	`task_output_oldBonusPoint`
Tipo campo	`String`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`checked`
Getter	`public`
Setter	`public`

   Fare clic su **Fine**.
3. Fare clic sulla classe **DataBeanSampleBean** per visualizzare il proprio codice di origine. Nel codice di origine, decommentare la seguente istruzione per l'importazione:

```
import com.ibm.commerce.sample.objects.*;
```

   Salvare il proprio lavoro.
4. Dal codice di origine per la classe DataBeanSampleBean, eliminare il commento relativo a Section 1 e inserire un commento per Section 2. In tal modo, il bean si estenderà da BonusAccessBean. Dopo le modifiche, il codice sarà il seguente:

```
/// Section 1 ////////////////////////////////////////////////

// Extend the databean to BonusAccessBean

public class DataBeanSampleBean
   extends com.ibm.commerce.sample.objects.BonusAccessBean
   implements SmartDataBean {


//////////////////////////////////////////////////////////////

/// Section 2 ////////////////////////////////////////////////
/*
// Extend the databean to BonusAccessBean
```

```
     public class DataBeanSampleBean implements SmartDataBean {
*/

//
//////////////////////////////////////////////////////////
```

Salvare il proprio lavoro.

5. Selezionare il metodo **setTask_output_userId(String)** per visualizzarne il codice di origine. Individuare la seguente riga del codice:

```
public void setTask_output_userId(java.lang.String newTask_output_userId) {
  task_output_userId = newTask_output_userId;
```

Dopo la riga precedente, immettere il codice seguente per creare l'istanza del nuovo BonusAccessBean:

```
//////////////////////////////////////
// Section A : instantiate BonusAccessBean

if (task_output_userId != null)
     this.setInit_argMemberId(newTask_output_userId);

//////////////////////////////////////
```

Salvare il proprio lavoro.

6. Selezionare il metodo **populate()** per visualizzarne il codice di origine. Eliminare il commento dalla sezione 2 per creare l'istanza di BonusAccessBean. In tal modo, viene inserito il seguente codice nel metodo:

```
setTask_output_oldBonusPoint(getRequestProperties().getString(
    "task_output_oldBonusPoint"));
```

Modifica del classpath: Prima di eseguire il test del comando modificato in WebSphere Test Environment, è necessario modificare un classpath per includere il nuovo progetto creato per il nuovo bean entità Bonus. Per modificare il classpath, effettuare le seguenti operazioni:

1. Dal menu **Spazio di lavoro** in VisualAge per Java, selezionare **Strumenti > WebSphere Test Environment**.
   Si apre WebSphere Test Environment Centro di controllo.
2. Fare clic su **Motore servlet**.
3. Se Motore servlet è in esecuzione, fare clic su **Arresta motore servlet** e, quindi, su **Modifica Classpath**.
4. Fare clic su **Seleziona tutto**, quindi fare clic su **OK**.

Modifica della maschera Sample.jsp per i punti di bonus: Per modificare la maschera di visualizzazione, effettuare le seguenti operazioni:

1. Aprire i file `Sample.jsp` e `Sample_All.jsp` in un editor di testo.

2. Copiare la quarta sezione del codice da Sample_All.jsp a Sample.jsp (la parte compresa tra <!-- SECTION 4 --> e <!-- END OF SECTION 4 -->). Il codice riportato di seguito viene inserito nella maschera JSP:

```
<!-- SECTION 4 -->

<h1>
Bonus Administration
</h1>

<%
if (userId != null) {
%>

<B>
<UL>
   <LI> The bonus point before update is
      <%=testBean.getTask_output_oldBonusPoint()%>
   <LI> The bonus point after update is
      <%=testBean.getBonusPoint()%>
</UL>
</B>

<%
}
%>

<br>
<B>Input to command:</B><P>

<FORM NAME=Bonus ACTION="MyNewControllerCmd">
<TABLE>
   <TR>
        <TD>
             <B>Logon ID </B>
         </TD>
        <TD>
             <input type=text name=input1
                 value='<%=testBean.getInput1()%>'>
        </TD>
   </TR>
   <TR>
         <TD>
             <B>Bonus Point</B>
         </TD>
         <TD>
              <input type=text name=input2>
         </TD>
     </TR>
     <TR>
          <TD COLSPAN=2>
               <input type=submit>
          </TD>
     </TR>
```

```
</TABLE>
</FORM>

<!-- END OF SECTION 4 -->
```

Salvare il file `Sample.jsp`.

3. Dal momento che il nuovo bean Bonus è protetto mediante il controllo accessi e gli utenti possono eseguire solo l'azione MyNewControllerCmd sul bean di cui sono proprietari, l'utente deve collegarsi. Pertanto, verrà utilizzata la funzione login per consentire all'utente di accedere al negozio di esempio. A tale scopo, è necessario copiare il file Sample.jsp nella struttura di directory del negozio, poiché dopo aver effettuato il collegamento al negozio, il controller Web ricercherà il file Sample.jsp nella directory del negozio. Copiare il file Sample.jsp da:
   unità_vaj`:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web`
   a
   unità_vaj`:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web\`*directory:negozio*.

   Nota: Per questo negozio di esempio, il valore di *directory_negozio* è InFashion.
4. Assicurarsi che WebSphere Test Environment sia in esecuzione (fare riferimento a Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347).
5. Collegarsi come utente `wcsadmin` effettuando le seguenti operazioni:
   - Immettere il seguente URL nel browser:
     `http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?storeId=`*Id_negozio*`&catalogId=`*Id_catalogo*`&langId=-1`
   - Fare clic sul collegamento **Esegui la registrazione**.
     Viene visualizzata la pagina Registrazione o collegamento.
   - Nel campo **Indirizzo e-mail**, immettere `wcsadmin`.
   - Nel campo **Password**, immettere la password per l'utente `wcsadmin` e fare clic su **Collegamento**.

     Nota: La password originale era `wcsadmin`, ma è stata modificata quando è stato eseguito il comando `contractPublish` durante il processo di installazione. Questa operazione viene descritta nel seguente documento:

     – Business *WebSphere Commerce Studio Business Developer Edition Guida all'installazione*

     – Professional *WebSphere Commerce Studio Professional Developer Edition Guida all'installazione*

6. Dopo aver completato il collegamento, immettere nel browser il seguente URL:

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=wcsadmin&input2=1000
   ```

   Viene visualizzata una pagina che contiene tutti i parametri di emissione precedenti e un nuovo modulo che consente di aggiornare il bilancio dei punti di bonus per un utente. La pagina visualizzata è simile alla seguente figura:

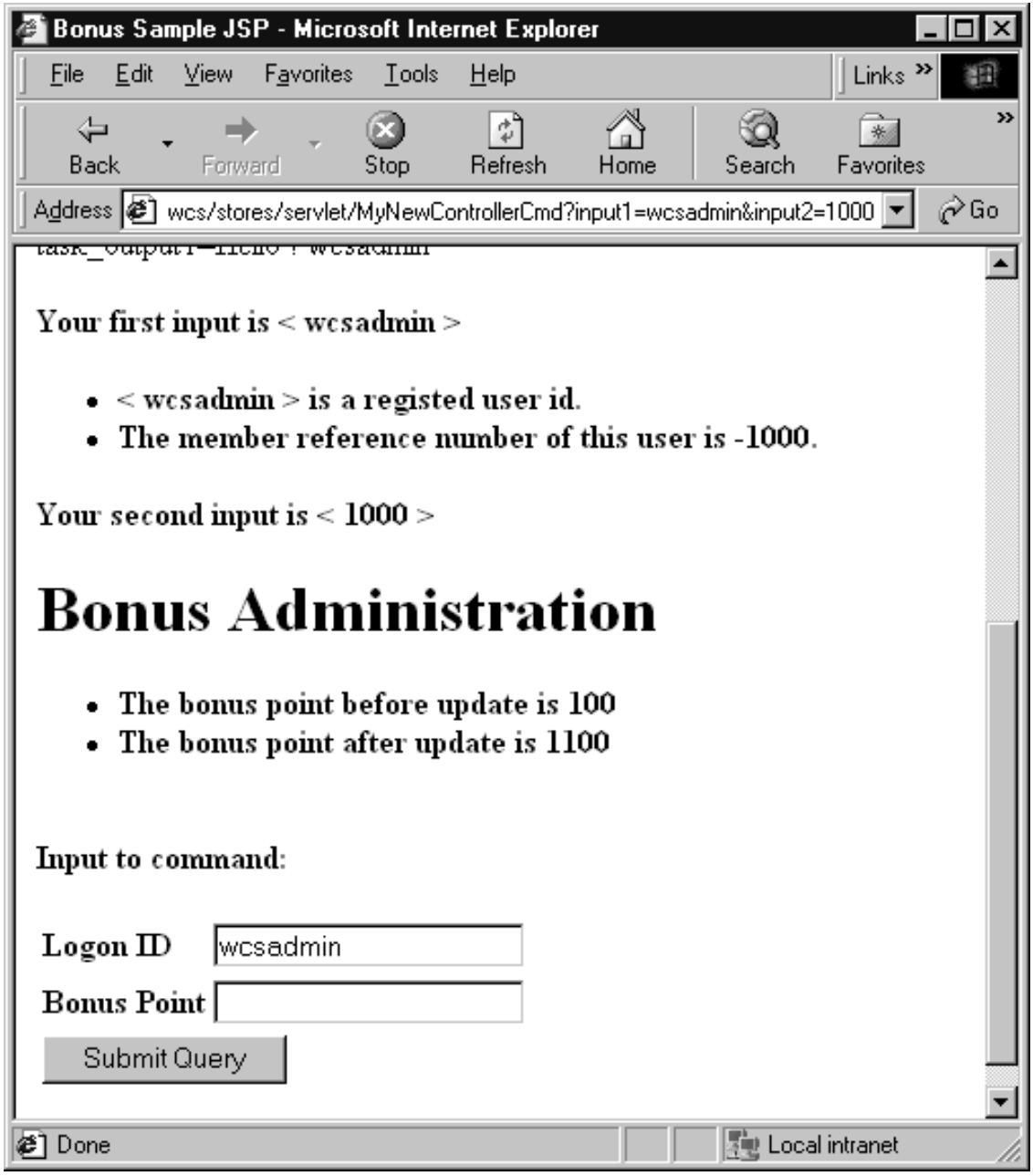


Figura 40.

7. Creare una versione del codice nello stato corrente. Ridenominare la versione `mySample 1.7 Completed`. Quando si assegna la versione al

codice, selezionare entrambi i progetti. Per ulteriori dettagli sull'assegnazione della versione al proprio codice, fare riferimento a 12 a pagina 215.

(Facoltativo) Utilizzo del programma di debug in VisualAge per Java

In questa sezione viene illustrato come aggiungere un punto di interruzione nel codice e avviare il programma di debug di VisualAge per Java. Questa sezione è stata inclusa per introdurre il programma di debug. Per ulteriori dettagli su questa efficace funzione, consultare la Guida in linea di VisualAge per Java.

Questa sezione è facoltativa in quanto non contiene nuovo codice necessario per il supporto didattico. Questa sezione illustra come creare un punto di interruzione, verificare i valori di alcune variabili del punto di interruzioni e come rimuoverlo.

Aggiunta di un punto di interruzione al codice

Per aggiungere un punto di interruzione al codice, effettuare le seguenti operazioni:

1. Assicurarsi che l'utilizzo dei punti di interruzione sia abilitato all'interno del proprio spazio di lavoro; a tale scopo, completare la seguente procedura:
   a. Dal menu **Finestra**, selezionare **Debug**. Assicurarsi che **Abilitazione globale punti di interruzione** sia selezionato.
2. Selezionare la scheda **Progetti**.
3. Espandere il progetto **_WCSamples**.
4. Espandere il pacchetto **com.ibm.commerce.sample.commands**.
5. Espandere la classe **MyNewTaskCmdImpl**.
6. Selezionare il metodo **performExecute** per visualizzarne il codice di origine.
7. Nel pannello Origine, posizionare il cursore (e fare clic) all'inizio della seguente riga di codice:

   ```
   setTask_output1( "Hello ! " + getTask_input1() );
   ```

   Lasciare il cursore in questa posizione.
8. Dal menu **Modifica**, selezionare **Punto di interruzione**.
9. Nella finestra Configurazione: punto di interruzione N.1, selezionare **In Thread selezionato** e fare clic su **OK**.
10. Assicurarsi che WebSphere Test Environment sia in esecuzione (fare riferimento a Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347).
11. Collegarsi come utente `wcsadmin` effettuando le seguenti operazioni:

- Immettere il seguente URL nel browser:

  ```
  http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
       storeId=Id_negozio&catalogId=Id_catalogo&langId=-1
  ```

- Fare clic sul collegamento **Esegui la registrazione**.
  Viene visualizzata la pagina Registrazione o collegamento.
- Nel campo **Indirizzo e-mail**, immettere `wcsadmin`.
- Nel campo **Password**, immettere la password per l'utente wcsadmin e fare clic su **Collegamento**.

12. Dopo aver completato il collegamento, immettere il seguente URL:

    ```
    http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=wcsadmin&input2=1000
    ```

13. La finestra Debugger viene visualizzata quando all'interno del codice si raggiunge il punto di interruzione.

Verifica dei valori delle variabili

Questa sezione illustra come verificare i valori delle variabili `task_input1` e `task_output1` nelle varie fasi dell'esecuzione del comando.

Quando si apre la finestra Debugger a causa del punto di interruzione nel metodo performExecute di MyNewTaskCmdImpl, passare a questa finestra ed effettuare le seguenti operazioni:

1. Nel pannello Variabile, espandere **this**.
2. Fare clic su **task_input1**.
   Nel pannello Valore, viene visualizzato il valore di questa variabile relativo a questa fase dell'esecuzione. Viene visualizzato `wcsadmin`.

Per verificare che il valore della variabile `task_output1` venga impostato come previsto, è importante che il programma di debug continui l'esecuzione del metodo performExecute per raggiungere il punto in cui la variabile viene impostata. A tale scopo, effettuare le seguenti operazioni:

1. Utilizzare la funzione Step over per spostarsi all'interno del codice. Per utilizzare questa funzione, utilizzare uno dei seguenti approcci:
   - Dal menu **Selezionato**, selezionare **Step Over**.
   - Fare clic su F6.
   - Fare clic sull'icona Step Over.

   Per questo esempio, fare clic su F6 quattro volte. In tal modo, viene eseguito il codice che imposta la variabile task_output1.
2. Nel pannello Variabile, fare clic su task_output1.
   Nel pannello Valore, viene visualizzato il valore di questa variabile relativo a questa specifica fase dell'esecuzione del comando. Viene visualizzato `Hello ! wcsadmin`.

3. È possibile terminare l'esecuzione del comando facendo clic sull'icona Riprendi.

Rimozione del punto di interruzione

Per rimuovere il punto di interruzione dal codice, effettuare le seguenti operazioni:

1. Ritornare alla finestra Spazio di lavoro.
2. Assicurarsi che sia possibile visualizzare il codice di origine del metodo`performExecute` della classe `MyNewTaskCmdImpl`.
3. Nel pannello Origine, posizionarsi sulla seguente riga di codice:

   ```
   setTask_output1( "Hello ! " + getTask_input1() );
   ```

   Sul margine sinistro del pannello, è visualizzato un punto blu che indica il punto di interruzione.
4. Fare doppio clic sul punto blu per rimuovere il punto di interruzione.

Il punto di interruzione viene rimosso dal codice.

Integrazione di MyNewControllerCmd con il negozio di esempio in WebSphere Test Environment

In questa sezione, viene descritta la procedura per l'aggiunta di un collegamento alla home page del negozio di esempio che richiama MyNewControllerCmd. Per eseguire questa fase di integrazione, effettuare le seguenti operazioni:

1. In un editor di testi, aprire il file `sidebar.jsp`. Questo file si trova nella seguente directory:
   unità_vaj`:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\`*directory_negozio*`\include`
   dove *unità_vaj* è l'unità sulla quale è stato installato VisualAge per Java e *directory_negozio* è il nome della directory del negozio di esempio.
2. Aggiungere un'altra riga con un link a MyNewControllerCmd nella tabella inserendo il seguente codice prima della tag finale `</table>`:

   ```
   <tr>
   <td>
   <a href = "MyNewControllerCmd?input1=wcsadmin&input2=1000">
        MyNewControllerCmd</a>
   </td>
   </tr>
   ```

   Salvare il proprio lavoro.
3. Assicurarsi che WebSphere Test Environment sia in esecuzione.
4. Eseguire il test dell'integrazione immettendo il seguente URL in un browser:

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=Id_negozio&catalogId=Id_catalogo&langId=-1
```

Fare clic sul collegamento **Esegui la registrazione**.
Viene visualizzata la pagina Registrazione o collegamento.

5. Nel campo **Indirizzo e-mail**, immettere `wcsadmin`.
6. Nel campo **Password**, immettere la password per l'utente wcsadmin e fare clic su **Collegamento**.
7. Una volta eseguito il collegamento, fare clic sul collegamento **MyNewControllerCmd** nel pannello di navigazione laterale. Viene visualizzato il JSP di esempio.

   Nota: Se il pannello di navigazione laterale non visualizza alcun collegamento, sidebar.jsp può essere memorizzato nella cache. Eliminarlo dalla cache e ricaricare la pagina. Per informazioni sull'eliminazione dei file JSP compilati, fare riferimento anche a Appendice C, "Suggerimenti per VisualAge per Java" a pagina 385. Una volta cancellati i file JSP compilati, può rendersi necessario riavviare il servlet engine.

(Facoltativo) Distribuzione di una nuova logica aziendale su un WebSphere Commerce Server remoto

In questa sezione viene descritto come distribuire la nuova logica aziendale in un negozio che viene eseguito su un WebSphere Commerce Server remoto. Prima di procedere con la distribuzione, è necessario aver creato un negozio (sulla base del negozio di esempio InFashion) sul WebSphere Commerce Server remoto.

Il processo di distribuzione include passaggi che vengono eseguiti sia sulla macchina di sviluppo che sulla macchina del WebSphere Commerce Server di destinazione.

Creazione del file JAR per la nuova logica di comando

È necessario creare un file JAR che contenga la nuova logica di comando e del bean di dati. Per creare questo file JAR, effettuare le seguenti operazioni:

1. Arrestare WebSphere Test Environment, come descritto in Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347.
2. Con la scheda Progetti selezionata, selezionare il progetto **_WCSamples**.
3. Con il progetto evidenziato, fare clic con il pulsante destro del mouse e selezionare **Esporta**.
   Si apre la SmartGuide di esportazione.
4. Selezionare **file Jar** e fare clic su **Avanti.**

5. Nel campo File jar immettere quanto segue:
   unità:\WebSphere\CommerceServerDev\mytemp\wcssamples_1.jar
   dove *unità* è l'unità sulla quale è installato Commerce Studio.
6. Selezionare gli attributi nel seguente modo:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
bean	Selezionato
Includi attributi di debug nei file .class	Selezionato

   Accettare i valori predefiniti per gli altri attributi.
7. Fare clic su **Fine**.
8. Se richiesto, confermare la creazione della nuova directory.

Poiché il file JAR creato non contiene le informazioni complete sulla denominazione dei pacchetti è necessario utilizzare un altro programma di utilità per i pacchetti (esterni a VisualAge per Java) per creare nuovi pacchetti del file JAR. Per creare un nuovo pacchetto di questo file, eseguire le procedure descritte:

1. Da una finestra comandi, passare alla seguente following directory:
   unità:\WebSphere\CommerceServerDev\mytemp
2. Immettere `mkdir temp1`.
3. Immettere `cd temp1`.
4. Impostare il percorso come specificato di seguito:
   `set PATH=%PATH%;`*unità*`:\WebSphere\WebSphereStudio4\bin;`
   dove *unità* è l'unità sulla quale è stato installato WebSphere Studio.
5. Immettere `jar xvf ../wcssamples_1.jar`
6. Immettere `jar cvf ../wcssamples.jar *` (si noti che la parte _1 è stata rimossa dal nome).

Creazione del file JAR per il nuovo gruppo EJB

È necessario creare un file JAR per il nuovo gruppo EJB. Per creare questo file, effettuare le seguenti operazioni:

1. Con la scheda EJB selezionata, fare clic con il pulsante destro del mouse sul gruppo EJB **WCSSamplesEntityBeans** e selezionare **Esporta > EJB 1.1 JAR**.
   Si apre la SmartGuide per l'esportazione su un file EJB 1.1 JAR.
2. Nel campo **File JAR**, immettere
   unità:\WebSphere\CommerceServerDev\mytemp\sampleEntityBeans_DT.jar

3. Selezionare gli attributi nel seguente modo:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
Database di destinazione	DB2 Se si sta distribuendo su un database DB2, selezionare **DB2 per NT, V7.1**. Oracle Se si sta distribuendo su un database Oracle, selezionare **Oracle, V8**.
Includi attributi di debug nei file .class	Selezionato

Accettare i valori predefiniti per gli altri attributi.

4. Fare clic su **Fine**.

Il file JAR è stato creato.

Il file JAR è stato denominato con il suffisso "_DT" per indicare che è necessario eseguire questo file JAR con lo strumento EJB Deploy fornito con WebSphere Application Server prima di distribuire il file nell'applicazione WebSphere Commerce.

Creazione del file JAR di implementazione per il nuovo bean enterprise

È necessario creare un file JAR che contenga il nuovo codice di implementazione per il bean enterprise Bonus. Per creare questo file, effettuare le seguenti operazioni:

1. Con la scheda del progetto selezionata, fare clic con il pulsante destro del mouse su **_WCSSamplesEntityBeansProject** e selezionare **Esporta**. Viene visualizzata la SmartGuide per l'esportazione.
2. Selezionare **file Jar** e fare clic su **Avanti.**
3. Nel campo **File JAR** immettere
   unità`:\WebSphere\CommerceServerDev\mytemp\sampleImpl_1.jar`
4. Selezionare gli attributi nel seguente modo:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
bean	Selezionato

Attributo	Valore
Includi attributi di debug nei file .class	Selezionato

Accettare i valori predefiniti per gli altri attributi.

5. Fare clic su **Fine**.

Il file JAR è stato creato. Chiudere VisualAge per Java.

Poiché il file JAR creato non contiene le informazioni complete sulla denominazione dei pacchetti è necessario utilizzare un altro programma di utilità per i pacchetti (esterni a VisualAge per Java) per creare nuovi pacchetti del file JAR. Per creare un nuovo pacchetto di questo file, eseguire le procedure descritte:

1. Da una finestra comandi, passare alla seguente following directory: *unità*:\WebSphere\CommerceServerDev\mytemp
2. Immettere `mkdir temp2`.
3. Immettere `cd temp2`.
4. Impostare il percorso come specificato di seguito: `set PATH=%PATH%;`*unità*`:\WebSphere\WebSphereStudio4\bin;` dove *unità* è l'unità sulla quale è stato installato WebSphere Studio.
5. Immettere `jar xvf ../sampleImpl_1.jar`
6. Immettere `jar cvf ../sampleImpl.jar *` (si noti che la parte _1 è stata rimossa dal nome).

Copia dei file JSP sul WebSphere Commerce Server di destinazione

È necessario copiare il file `Sample.jsp` e i file `sidebar.jsp` aggiornati nelle directory appropriate per il negozio in cui si distribuisce il codice.

Prima di copiare le maschere JSP aggiornate sul WebSphere Commerce Server di destinazione, può essere utile eseguire una copia di backup delle maschere JSP originali presenti su quella macchina. In altre parole, rinominare il file esistente come `sidebar.jsp.bak`.

Per copiare questi file, effettuare le seguenti operazioni:

1. Sulla macchina di sviluppo, passare alla directory: *unità_vaj*:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web*directory_negozio* dove *unità_vaj* è l'unità sulla quale è stato installato VisualAge per Java e *directory_negozio* è il nome della directory per il negozio. Copiare il file `Sample.jsp`
2. Incollare il file `Sample.jar` nella seguente directory del WebSphere Commerce Server di destinazione:

```
unità:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_nome_istanza.ear\
   wcstores.war\directory_negozio
```

dove *unità* è l'unità in cui WebSphere Commerce è installato, *directory_negozio* è il nome della directory per il negozio e *nome_istanza* è il nome dell'istanza WebSphere Commerce.

3. Sulla macchina di sviluppo, visualizzare la seguente directory:
   unità_vaj:\VAJava\Ide\project_resources\IBM WebSphere Test Environment
   \hosts\default_host\default_app\web*directory_negozio*\include
   Copiare il file sidebar.jsp
4. Incollare il file sidebar.jsp nella seguente directory del WebSphere Commerce Server di destinazione:

```
unità:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_nome_istanza.ear\
   wcstores.war\directory_negozio\include
```

Copia dei file JAR sul WebSphere Commerce Server di destinazione

È necessario copiare i file JAR dalla macchina di sviluppo alla appropriata directory sul WebSphere Commerce Server di destinazione.

Esistono inoltre altri due passi che devono essere eseguiti sul file JAR contenente il nuovo gruppo EJB. È necessario eseguire lo strumento di distribuzione EJB sul file da WebSphere Application Server. Quindi, è necessario eseguire sul file il comando modifyIsolationLevel. Nella fase di copia dei file JAR nel WebSphere Commerce Server di destinazione, questo particolare file JAR viene memorizzato in una directory temporanea in attesa delle fasi successive.

Per copiare questi file, effettuare le seguenti operazioni:

1. Sulla macchina di sviluppo, passare alla seguente directory:
   unità:\WebSphere\CommerceServerDev\mytemp e posizionare i seguenti file:
   - wcssamples.jar
   - sampleImpl.jar
   - sampleEntityBeans_DT.jar

   dove *unità* è l'unità sulla quale è stato installato WebSphere Commerce Studio, Business Developer Edition.

   Ciascuno dei file precedenti deve essere copiato in una determinata directory del WebSphere Commerce Server di destinazione. Leggere i passi di seguito riportati per verificare che ciascun file sia memorizzato nella corretta ubicazione.
2. Copiare il file wcssamples_DT.jar nella seguente directory del WebSphere Commerce Server di destinazione:

unità:\WebSphere\AppServer\InstalledApps\
WC_Enterprise_App_*nome_istanza*.ear\wcstores.war\WEB-INF\lib

dove *unità* è l'unità in cui è installato WebSphere Commerce Business Edition e *nome_istanza* è il nome dell'istanza (ad esempio, demo).

3. Creare una directory di libreria temporanea in cui si desidera collocare il file JAR. Ovvero, creare la seguente directory:

   unità:\WebSphere\CommerceServer\temp\lib

4. Copiare il file sampleImpl.jar nella seguente directory del WebSphere Commerce Server di destinazione:

   unità:\WebSphere\CommerceServer\temp\lib

5. Copiare il file sampleEntityBeans_DT.jar nella seguente directory del WebSphere Commerce Server di destinazione:

   unità:\WebSphere\CommerceServer\temp

Esecuzione dello strumento per la distribuzione di EJB

Per poter eseguire correttamente i bean enterprise su un server di verifica o di produzione, è necessario generare un codice di distribuzione per i bean enterprise. Lo strumento di distribuzione per Enterprise JavaBeans (chiamato anche strumento di distribuzione EJB) fornito con WebSphere Application Server, contiene un'interfaccia riga comandi da utilizzare per generare un codice di distribuzione di bean enterprise.

Per eseguire questo strumento, eseguire le seguenti operazioni:

1. Alla richiesta comandi, passare alla seguente directory:

   unità:\WebSphere\CommerceServer\temp

2. Aggiungere temporaneamente lo strumento al percorso di sistema immettendo il seguente comando:

   PATH=*unità*:\WebSphere\AppServer\deploytool;%PATH%

3. Immettere il comando ejbdeploy come segue:

   ejbdeploy *EJBGroupJARFile WorkingDir FileJAREmissione* -nowarn -keep -35 -cp *ClassPathFIleJARDip*

   dove:

   - *FileJARGruppoEJB* è il nome completo del file JAR dei bean enterprise per cui si desidera generare un codice distribuito. In questo caso, *unità*:\WebSphere\CommerceServer\temp\sampleEntityBeans_DT.jar.
   - *DirLavoro* è il nome della directory in cui sono memorizzati i file temporanei richiesti per la creazione del codice.
   - *FileJAREmissione* è il nome completo del file JAR di emissione. In questo caso, immettere *unità*:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar.

- `-nowarn` è un parametro facoltativo che elimina i messaggi informativi e di avvertenza.
- `-keep` è un parametro facoltativo per mantenere la directory in uso dopo aver eseguito il comando ejbdeploy.
- `-35` è un parametro obbligatorio che utilizzerà le stesse regole di corrispondenza dall'alto verso il basso per i bean entità utilizzate nello strumento di distribuzione EJB fornito con WebSphere Application Server, Versione 3.5.
- `-cp` *`ClassPathOfDepJARFiles`* è il class path dei file JAR dipendenti. In questo caso, immettere

  ```
  "unità:\WebSphere\CommerceServer\temp\lib\sampleImpl.jar;
  unità:\WebSphere\AppServer\installedApps\
     WC_Enterprise_App_Nomeistanza.ear\lib\wcsejbimpl.jar"
  ```

  Nota: È necessario racchiudere il valore della class path tra doppi apici (“”).

Modifica del livello di isolamento transazione per i bean Bonus

Questa procedura prevede l'utilizzo del comando `modifyIsolationLevel` per modificare il livello di isolamento transazione del bean Bonus. Questo strumento imposta anche il livello di isolamento del bean sul livello richiesto dal tipo di database utilizzato.

Per eseguire il comando `modifyIsolationLevel`, effettuare le seguenti operazioni:

1. Sul WebSphere Commerce Server di destinazione, aprire una finestra comandi.
2. Passare alla seguente directory:
   `unità``:\WebSphere\CommerceServer\bin`
3. È necessario utilizzare il comando `modifyIsolationLevel` la cui sintassi è riportata di seguito:

   ```
   modifyIsolationLevel -jarFile nome_file_jar.jar
      -logFile nome_file_log -dbType tipo_db
   ```

   dove

   - *`nome_file_jar.jar`* è il nome del file JAR che contiene il codice personalizzato
   - *`nome_file_log`* è il nome completo del file in cui registrare le informazioni
   - *`tipo_db`* è il tipo di database in uso. Immettere DB2 oppure ORACLE

   Di seguito viene riportato un esempio del comando `modifyIsolationLevel` con tutti i valori specificati:

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

Se la finestra comandi non visualizza eccezioni, il comando è stato eseguito correttamente. Al termine dell'esecuzione, è possibile notare che la data e l'orario del file JAR sono cambiati.

Nota: I nomi dei parametri operano una distinzione tra maiuscolo e minuscolo, ossia jar**F**ile non è lo stesso di jar**f**ile. Assicurarsi di immettere correttamente i nomi dei parametri.

Aggiornamento del database di destinazione

Poiché si sta effettuando la distribuzione di una nuova logica aziendale su un WebSphere Commerce Server di destinazione che utilizza un database diverso da quello utilizzato da WebSphere Test Environment, è necessario aggiornare il database di destinazione per confermare le modifiche effettuate al registro comandi e per creare la tabella BONUS.

DB2 Se si utilizza un database DB2, effettuare le seguenti operazioni per aggiornare il database di destinazione:

1. Aprire il Centro comandi DB2 (**Start > Programmi >IBM DB2 > Centro di comandi**).
2. Dal menu **Strumenti** selezionare **Impostazioni strumenti**.
3. Selezionare la casella **Utilizza carattere fine istruzione** e assicurarsi che il carattere specificato sia un punto e virgola (**;**)
4. Dopo aver selezionato la scheda Script, creare la voce richiesta nella tabella URLREG immettendo le seguenti informazioni nella finestra dello script:

   ```
   connettersi a nome_database;
   insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
       DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
       'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
       'This is a new controller command for test/education purposes.',
       null)
   ```

   dove *nome_database* è il nome del database e fare clic sull'icona di esecuzione.
   Questo comando viene utilizzato da tutti i venditori (indicato dal valore 0 per STOREENT_ID).
5. Creare una voce nella tabella VIEWREG immettendo quanto segue nella finestra dello script:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE) values
   ('SampleViewTask',-1, 0, 'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=Sample.jsp','This is a sample view for the Bonus Point
   exercise', 0, null)
```

e fare clic sull'icona di esecuzione.

6. Creare la tabella BONUS immettendo quanto segue nella finestra dello script:

```
CREATE TABLE Bonus (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (id_utente) on delete cascade)
```

Fare clic sull'icona di esecuzione.

I passaggi precedenti consentono quindi di registrare MyNewControllerCmd e SampleViewTask nel registro comandi e di creare la tabella BONUS.

Oracle Se si utilizza un database Oracle, effettuare le seguenti operazioni per aggiornare il database di destinazione:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password** immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.
5. Creare la voce richiesta nella tabella URLREG, immettendo le seguenti informazioni nella finestra SQL Plus:

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
    DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
    'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
    'This is a new controller command for test/education purposes.',
    null);
```

e premere Invia per eseguire l'istruzione SQL.

6. Creare una voce nella tabella VIEWREG, immettendo la seguente istruzione nella finestra SQL Plus:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE) values
   ('SampleViewTask',-1, 0, 'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=Sample.jsp','This is a sample view for the Bonus Point
   exercise', 0, null);
```

e premere Invia per eseguire l'istruzione SQL.

7. Creare la tabella BONUS immettendo la seguente istruzione nella finestra SQL Plus:

```
CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   e premere Invia per eseguire l'istruzione SQL.
8. Immettere quanto segue per confermare le modifiche al database:

```
commit;
```

   e premere Invia per eseguire l'istruzione SQL.

Caricamento delle politiche di controllo accesso per le nuove risorse

Nel supporto didattico, è stato creato un nuovo bean enterprise (il bean Bonus), che si tratta di una risorsa da proteggere. Pertanto, esiste una politica del controllo accessi correlata a questa risorsa. È stato creato anche un nuovo comando di controller che può essere eseguito da tutti gli utenti. Le informazioni sul caricamento della politica del controllo accessi sono state caricate sulla macchina di sviluppo. È ora necessario caricarle sul WebSphere Commerce Server di destinazione.

Per impostare le politiche di controllo accessi, effettuare le seguenti operazioni:

1. Inserire il seguente CD nella relativa unità:
   - Business CD di WebSphere Commerce Business Edition, V5.4 Disco 2
   - Professional CD di WebSphere Commerce Professional Edition, V5.4 Disco 2
2. Visualizzare la seguente directory:

   unità_CD:\repository\samples\programguide\
3. Nella directory, individuare i seguenti file:
   - SampleCmdACPolicy.xml
     Questo file XML contiene la politica di controllo accessi utilizzata dal nuovo comando di controller.
   - SampleACPolicy.xml
     Questo file XML contiene la politica di controllo accessi utilizzata quando si crea un nuovo bean enterprise.
   - SampleACPolicy_*locale*.xml
     dove *locale* è l'identificativo della lingua. Questo file XML contiene la descrizione della politica del controllo accessi.
4. Copiare i tre file precedenti nella seguente directory:
   unità:\WebSphere\CommerceServer\xml\policies\xml
5. Per caricare il file SampleCmdACPolicy.xml, utilizzare una richiesta comandi per passare alla seguente directory:

unità:\WebSphere\CommerceServer\bin
È necessario eseguire il comando acpload, il cui formato è:

```
acpload nome_db utente_db password_db inputXMLFile
```

dove

- *nome_db* è il nome del database
- *utente_db* è il nome dell'utente del database
- *password_db* è la password del database
- *FileXMLinput* è il nome del file XML che contiene la politica.

Ad esempio, è possibile inviare il seguente comando:

```
acpload mall user password SampleCmdACPolicy.xml
```

6. Caricare il file SampleACPolicy.xml, eseguendo il comando acpload specificando il file SampleACPolicy.xml come file di immissione. Ad esempio, è possibile inviare il seguente comando:

   ```
   acpload mall user password SampleACPolicy.xml
   ```

7. Per caricare la descrizione della politica, è necessario inviare il comando acpnlsload, il cui formato è:

   ```
   acpnlsload nome_db utente_db password_db inputXMLFile
   ```

   Ad esempio, è possibile inviare il seguente comando:

   ```
   acpnlsload mall user password SampleACPolicy_en_US.xml
   ```

In genere, per rendere effettiva la politica occorre aggiornare il registro. In questo caso, non occorre eseguire l'aggiornamento, in quanto sarà necessario arrestare e avviare di nuovo l'applicazione WebSphere Commerce Server in in WebSphere Application Server, una procedura prevista dal processo di distribuzione del bean enterprise. In caso contrario, è possibile utilizzare la Console di gestione di WebSphere Commerce per aggiornare il registro. Per ulteriori informazioni sulla Console di gestione, consultare la Guida in linea di WebSphere Commerce.

400 Se si distribuisce ad un'istanza di WebSphere Commerce in esecuzione su una macchia iSeries, utilizzare comandi diversi per caricare le informazioni sulla politica di controllo accessi. Utilizzare il comando LODWCSAC invece del comando acpload e il comando LODWCSACD invece del comando acpnlsload.

La sintassi del comando LODWCSAC è:

```
LODWCSAC DATABASE(Nomedb) SCHEMA(Nomeschema)
PASSWD(Passwordistanza) INSTROOT('Rootistanza')
INFILE('Fileimmissione')
```

dove

- *Nomedb* è il nome del database relazionale come definito nel comando WRKRDBDIRE.
- *Nomeschema* è il nome dello schema di database per l'istanza (stesso nome del nome dell'istanza).
- *Passwordistanza* è la password dell'istanza.
- *Rootistanza* è la root dell'istanza. Un esempio di root dell'istanza è

  /QIBM/UserData/WebCommerce/instances/*Nomeistanza*
- *Fileimmissione* è il nome completo del file XML di immissione che dispone delle politiche di accesso

La sintassi del comando LODWCSACD è:

```
LODWCSACD DATABASE(Nomedb) SCHEMA(Nomeschema)
PASSWD(Passwordistanza) INSTROOT('Rootistanza')
INFILE('Fileimmissione')
```

È possibile memorizzare i file XML per le politiche del controllo accessi nella seguente directory:

/QIBM/UserData/WebCommerce/instances/*Nomeistanza*

Inoltre, è necessario utilizzare il percorso completo per il DTD del controllo accessi all'interno dei file XML per le politiche di controllo accessi. I DTD per le politiche di controllo accessi sono memorizzati nella directory /QIBM/ProdData/WebCommerce/xml/policies/dtd.

Come esempio, se si distribuiscono le politiche del controllo accessi per i supporti didattici in un'istanza Websphere Commerce in esecuzione su una macchina iSeries, è necessario modificare la specifica DTD nei file XML per tali politiche da:

```
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">
```

a

```
<!DOCTYPE Policies SYSTEM "/QIBM/ProdData/WebCommerce/
xml/policies/dtd/accesscontrolpolicies.dtd">
```

Per ulteriori informazioni sul modello di controllo accessi di WebSphere Commerce, consultare il manuale *WebSphere Commerce Access Control Guide*.

Esportazione dell'enterprise application corrente da WebSphere Application Server

In questo passo, l'enterprise application corrente viene esportata da WebSphere Application Server in modo che sia possibile aprirla con lo Strumento di assemblaggio delle applicazioni.

Per esportare l'enterprise application corrente, attenersi alla seguente procedura:

1. Creare una directory in cui l'enterprise application corrente verrà esportato. Non chiamare questa directory "temp" per evitare che il file venga eliminato durante le procedure di routine di manutenzione del sistema, fino a quando non si è certi di essere soddisfatti del funzionamento del codice personalizzato una volta distribuito. Per creare questa directory, effettuare le seguenti operazioni ad una richiesta comandi:
   a. Visualizzare la seguente directory:
      unità:\WebSphere\CommerceServer\
   b. Immettere il seguente comando:
      ```
      mkdir working
      ```
      Viene creata la directory *unità*:\WebSphere\CommerceServer\working.
2. Aprire la WebSphere Application Server Console di gestione.
3. Espandere **Dominio di gestione WebSphere**.
4. Espandere **Enterprise Applications**.
5. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **demo** e selezionare **Esporta applicazione**.
6. Nel campo **Esporta directory**, immettere *unità*:\WebSphere\CommerceServer\working.
   Viene esportata l'intera applicazione, incluse tutte le risorse presenti nel file WC_Enterprise_App_*Nomeistanza*.ear (dove *Nomeistanza* è il nome dell'istanza di WebSphere Commerce).
7. Fare clic su **OK**. L'esportazione dell'applicazione potrebbe richiedere alcuni minuti.

Esportazione delle informazioni di configurazione XML per l'enterprise application

È inoltre necessario esportare le informazioni di configurazione XML per l'enterprise application. Per esportare queste informazioni, utilizzare l'utilità della riga comandi XMLConfig fornita da WebSphere Application Server.

Per esportare queste informazioni di configurazione, effettuare le seguenti operazioni:

1. Copiare il file `was.export.app.xml` dalla seguente directory:

   unità`:\WebSphere\CommerceServer\xml\config`

   nella seguente directory:

   unità`:\WebSphere\CommerceServer\working`

2. Aprire il file `was.export.app.xml` in un editor di testo. In questo file, sostituire tutte le occorrenze di`$Enterprise_Application_Name$` con

   `WebSphere Commerce Enterprise Application - `*Nomeistanza*

   dove *Nomeistanza* è il nome dell'istanza di WebSphere Commerce (ad esempio, demo). Salvare questo file.

   Nota: Il valore che si immette deve corrispondere al valore visualizzato per la propria istanza nella console di gestione avanzata di WebSphere.

3. Alla richiesta comandi, passare alla seguente directory:

   unità`:\WebSphere\CommerceServer\working`

4. Richiamare lo strumento XMLConfig per eseguire un'esportazione parziale immettendo il seguente comando:

   ```
   xmlConfig -export OutputFile.xml -partial was.export.app.xml
      -adminNodeName wasHostName
   ```

   dove *wasHostName* è il nome del nodo in WebSphere Application Server contenente l'enterprise application corrente. Inoltre, `OutputFile.xml` è il nome del file creato in seguito all'esecuzione di questo comando e `was.export.app.xml` è il file modificato al passo 2.

Dopo l'esportazione delle informazioni relative ai bean enterprise presenti nell'enterprise application corrente, è necessario aggiungere una nuova stanza al file XML con la descrizione del bean Bonus.

Per aggiungere la nuova stanza con la descrizione del bean Bonus, procedere nel modo seguente:

1. Visualizzare la seguente directory:

   unità`:\WebSphere\CommerceServer\working`

2. Aprire il file `OutputFile.xml` con un editor di testo.
3. Individuare la tag `<ear-file-name>` e sostituire il valore con quello di seguito riportato:

   ```
   unità:\WebSphere\CommerceServer\working\
      WC_Enterprise_App_Nomeistanza.ear
   ```

4. È necessario inoltre aggiungere una nuova stanza per il bean Bonus, come mostrato nell'esempio di seguito riportato:

```
<ejb-module name="WCSSamplesEntityBeans">
   <jar-file>sampleEntityBeans.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         WebSphere Commerce Server - demo/</application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/WebSphere Commerce DB2 DataSource demo</jndi-name>
         <default-user>utente</default-user>
         <default-password>password</default-password>
      </data-source>
      <enterprise-bean-binding name="Bonus_Binding">
         <jndi-name>democom/ibm/commerce/sample/objects/Bonus</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

   dove

   - *user* è il nome utente del database.
   - *password* è la password per l'utente del database.

   Note:

   a. Le interruzioni di riga nell'esempio precedente sono state inserite unicamente per agevolare la visualizzazione.
   b. Verificare che il valore **$hostName$** corrisponda al nome server del nodo di amministrazione corrente. Inoltre, accertarsi che non vi siano caratteri di ritorno unitario su questa riga.
   c. La specifica `<application-server-full-name>` non può essere distribuita su più di una riga.
   d. Se si utilizza un database Oracle, è necessario modificare le informazioni di origine dati. Modificare la seguente riga estratta dalla precedente sezione di codice:

      ```
      <jndi-name>jdbc/WebSphere Commerce DB2 DataSource demo</jndi-name>
      ```

      nella seguente:

      ```
      <jndi-name>jdbc/WebSphere Commerce Oracle DataSource demo</jndi-name>
      ```

   e. Quando si esegue la distribuzione delle proprie applicazioni (ad esempio, all'esterno di questo supporto didattico), accertarsi che il nome JNDI dei bean enterprise specificato nel file XML corrisponda al nome JNDI utilizzato in VisualAge per Java e che abbia l'istanza WebSphere Commerce collegata.

5. Salvare il file `OutputFile.xml`.

Assemblaggio del nuovo gruppo EJB nell'enterprise application

In questa fase, l'enterprise application viene aperta con lo strumento di assemblaggio delle applicazioni. Una volta aperta con questo strumento, è possibile effettuare le operazioni di seguito riportate per aggiungere il nuovo bean Bonus nell'enterprise application:

1. Importare il nuovo bean Bonus. Il file JAR per il nuovo gruppo EJB è memorizzato nella sezione del modulo EJB dell'enterprise application.
2. Impostare il classpath per il bean Bonus per includere il file JAR di implementazione.
3. Aggiungere il file JAR di implementazione all'applicazione. Questo file JAR viene memorizzato nella sezione File dell'enterprise application.
4. Impostare la sicurezza di WebSphere Application Server per i metodi contenuti nel bean Bonus.

Per assemblare il nuovo gruppo EJB nell'enterprise application, effettuare le seguenti operazioni:

1. Ritornare all'enterprise application corrente, effettuando le seguenti operazioni:
   a. Alla richiesta comandi, passare alla seguente directory:
      unità:\WebSphere\CommerceServer\working
   b. Immettere il seguente comando:
      ```
      copy WC_Enterprise_App_Nomeistanza.ear
         WC_Enterprise_App_Nomeistanza.ear.bak
      ```
2. Aprire la Console di gestione di WebSphere Application Server.
3. Dal menu **File**, selezionare **Strumenti > Application Assembly Tool**.
4. Se si apre una finestra di benvenuti, selezionare **Annulla** per chiuderla.
5. Aprire l'enterprise application che si desidera utilizzare attenendosi alla seguente procedura:
   a. Dal menu **File**, selezionare **Apri**.
   b. Nel campo **Nome file**, immettere:
      ```
      unità:\WebSphere\CommerceServer\working\
         WC_Enterprise_App_Nomeistanza.ear
      ```
      e fare clic su **Apri**. Prima di continuare con i passi successivi, attendere l'apertura dell'applicazione. L'apertura può richiedere alcuni minuti.
6. Fare clic con il pulsante destro del mouse su **Moduli EJB** e selezionare **Importa**.
7. Nel campo **Nome file**, immettere:
   unità:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar

e fare clic su **Apri**. Nella finestra per la conferma dei valori, fare clic su **OK**.

8. Una volta importato il file `sampleEntityBeans.jar`, individuare il gruppo EJB **WCSSamplesEntityBeans** e selezionare questo gruppo.
   Le informazioni su questo gruppo sono visualizzate nel pannello a destra.
9. Nel campo del classpath per il nuovo bean enterprise, inserire i file JAR dipendenti. In questo caso, immettere

   ```
   lib/sampleImpl.jar lib\wcsejbimpl.jar
   ```
10. Fare clic su **Applica**.
11. Aggiungere il file `sampleImpl.jar` all'applicazione, effettuando le seguenti operazioni:
    a. Fare clic con il pulsante destro del mouse sul nodo **File** per l'enterprise application e selezionare**Aggiungi file**. Il nodo **File** per l'enterprise application si trova accanto al pulsante dell'albero gerarchico. Esistono altri nodi File per i componenti dell'enterprise application, ma basta selezionare il nodo File per l'intera applicazione.
    b. Nella finestra Aggiungi file fare clic su **Sfoglia**.
    c. Visualizzare la directory *unità*`:\WebSphere\CommerceServer\temp`.
    d. Con questa directory evidenziata, fare clic su **Seleziona**.
    e. Ritornare alla finestra Aggiungi file. Il contenuto della directory *unità*`:\WebSphere\CommerceServer\temp` risulta visualizzato. Evidenziare la directory `lib` directory.
       Il contenuto della directory `lib` viene visualizzato nel pannello di destra.
    f. Nel pannello di destra, selezionare il file `sampleImpl.jar` e fare clic su **Aggiungi**. Il file viene visualizzato nel pannello File selezionati.
    g. Fare clic su **OK**.
12. Configurare la sicurezza per il bean Bonus effettuando le seguenti operazioni:
    a. Con il nodo Moduli EJB espanso, individuare ed espandere il nodo **WCSSamplesEntityBeans**.
    b. Espandere **Entity Beans**.
    c. Espandere **Bonus**
    d. Fare clic su **Estensioni metodo**, quindi, nel pannello a destra, procedere come segue:
       1) Fare clic sul separatore **Avanzato**.
       2) Verificare che **Identità di sicurezza** sia selezionato.
       3) Per ciascun metodo, verificare che sia selezionato **Usa identità di server EJB**.
       4) Fare clic su **Applica** (se sono state effettuate delle modifiche).

e. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse **Ruoli sicurezza** nel gruppo EJB `WCSamplesEntityBeans` e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:
   1) Nel campo **Nome**, immettere `WCSecurityRole` e fare clic su **Applica**. Se questo ruolo già esiste, non è necessario eseguire questa operazione.

f. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse su **Autorizzazioni metodi** nel gruppo EJB `WCSamplesEntityBeans` e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:
   1) Nel campo **Nome autorizzazione metodo**, immettere `WCMethodPermission`
   2) Nell'area di selezione **Metodi**, fare clic su **Aggiungi**. Si apre la finestra
   Aggiungi metodo.
   3) Espandere **sampleEntityBeans.jar**, quindi **Bonus** e poi ciascuno degli elenchi di metodi **Home** e **Remote**.
   4) Tenere premuto il tasto Maius e selezionare tutti i metodi home, quindi fare clic su**OK**.
   5) Ripetere il processo di selezione del metodo per aggiungere anche i metodi remote (se esistono metodi remote).
   6) Nell'area di selezione Ruoli, fare clic su **Aggiungi**, selezionare `WCSecurityRole` e quindi fare clic su **OK**.
   7) Fare clic su **Applica** per ciascun aggiornamento.

13. Dal menu **File**, selezionare **Salva**.
14. Chiudere lo strumento di assemblaggio delle applicazioni.

Una volta completato questo passo, viene creata una nuova enterprise application contenente tutta la logica precedente e la nuova logica aziendale. Questi elementi sono contenuti nel file `WC_Enterprise_App_`*`instanceName`*`.ear` recentemente modificato.

Importazione della nuova enterprise application in WebSphere Application Server

Di seguito vengono riportati i passi principali per l'importazione della nuova enterprise application in WebSphere Application Server:

1. Arresto dell'enterprise application attualmente in esecuzione in WebSphere Application Server e successiva rimozione. Questi passi vengono eseguiti nella Console di gestione di WebSphere Application Server.
2. Importazione della nuova applicazione, mediante l'utility della riga comandi XMLConfig.
3. Aggiornamento della Console di gestione di WebSphere Application Server ed avvio della nuova enterprise application.

Queste singole operazioni verranno descritte in maniera più dettagliata nelle sezioni successive.

Arresto e rimozione dell'enterprise application corrente

Per arrestare e rimuovere l'enterprise application corrente da WebSphere Application Server, effettuare le seguenti operazioni:

1. Aprire la Console di gestione di WebSphere Application Server.
2. Espandere **WebSphere Administrative Domain**.
3. Espandere **Nodi**.
4. Espandere *Nomenodo* (dove *Nomenodo* è il nome del nodo).
5. Espandere **Server delle applicazioni**.
6. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Server -** *Nomeistanza* e selezionare **Arresta**.
7. Espandere **Enterprise Applications**.
8. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Enterprise Application - demo** e selezionare **Arresta**.
9. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Enterprise Application - demo** e selezionare **Rimuovi**.
10. Quando viene richiesto di indicare se l'applicazione deve essere esportata, selezionare **No**.

Importazione della nuova enterprise application mediante XMLConfig

Per importare la nuova enterprise application mediante l'utility della riga comandi XMLConfig, attenersi alla seguente procedura:

1. Visualizzare la seguente directory:

   ```
   unità:\WebSphere\CommerceServer\working
   ```

2. Alla richiesta comandi, immettere il comando di seguito riportato per importare l'enterprise application in WebSphere Application Server:

   ```
   xmlConfig -import OutputFile.xml -adminNodeName nomehost_was
   ```

   dove *nomehost_was* è il nome del nodo di WebSphere Application Server contenente l'applicazione corrente.

Nota: 400 Se si esegue la distribuzione di un'istanza di WebSphere Commerce in esecuzione su iSeries, occorre effettuare un ulteriore passo per modificare le autorizzazioni della directory dopo l'importazione

dell'applicazione. Consultare "Importazione dell'enterprise application" a pagina 382 per maggiori dettagli sulla modifica di tali autorizzazioni.

Avvio della nuova enterprise application

Una volta importata la nuova enterprise application mediante l'utility della riga comandi XMLConfig, è possibile utilizzare la Console di gestione di WebSphere Application Server per eseguire un aggiornamento e quindi avviare la nuova applicazione.

Per aggiornare la console e avviare la nuova applicazione, effettuare le seguenti operazioni:

1. Aprire la Console di gestione di WebSphere Application Server.
2. Espandere **Dominio di gestione WebSphere**
3. Evidenziare **Nodi**.
4. Fare clic sull'icona per l'**aggiornamento dell'albero secondario**.
5. Avviare l'applicazione WebSphere Commerce effettuando le seguenti operazioni:
   - Espandere **Server delle applicazioni**.
   - Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Server -** *Nomeistanza* e selezionare **Avvia**.

Test di MyNewControllerCmd

La successiva operazione è la verifica della nuova logica del negozio in esecuzione nell'ambiente di WebSphere Application Server. Per verificare MyNewControllerCmd, effettuare le seguenti operazioni:

1. Eseguire il test dell'integrazione immettendo il seguente URL in un browser:

   ```
   http://nomehost/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=Id_negozio&catalogId=Id_catalogo&langId=-1
   ```

   dove *Id_negozio* è l'ID del negozio e *Id_catalogo* è l'ID del catalogo del negozio.
2. Fare clic sul collegamento **Esegui la registrazione**.
   Viene visualizzata la pagina Registrazione o collegamento.
3. Nel campo **Indirizzo e-mail**, immettere `wcsadmin`.
4. Nel campo **Password**, immettere la password per l'ID wcsadmin utilizzata da questo sito, quindi fare clic su **Collegamento**.
5. Una volta eseguito il collegamento, fare clic sul collegamento **MyNewControllerCmd** nel pannello di navigazione laterale. Viene visualizzato il JSP di esempio.

Se non viene visualizzato il file aggiornato `sidebar.jsp` eseguire le operazioni descritte di seguito per ripulire la cache:

1. Eliminare i file memorizzati nella cache contenuti nella seguente directory: *unità*:\WebSphere\CommerceServer\instances*Nomeistanza*\cache
2. Eliminare i file cache importanti dalle seguenti directory:

   unità:\WebSphere\AppServer\temp*Nomehost*\
   WebSphere_Commerce_Server_*nomeIstanza*\
   WebSphere_Commerce_Enterprise_Application_-_*nomeIstanza*\
   wcstores.war*nomeNegozio*

   unità:\WebSphere\AppServer\temp*Nomehost*\
   WebSphere_Commerce_Server_*nomeIstanza*\
   WebSphere_Commerce_Enterprise_Application_-_*nomeIstanza*\
   wcstores.war
3. Eliminare il contenuto della cache del browser.

Se la compilazione della pagina JSP ha una durata eccessiva, è possibile che la pagina non venga visualizzata. In tal caso, ricaricare la pagina.

Capitolo 10. Modifica ed estensione della logica aziendale esistente

Nei seguenti supporti didattici viene illustrato come estendere e modificare la logica aziendale di WebSphere Commerce.

Estensione di un comando di controller esistente

In questa sezione viene descritto come estendere il comando di controller OrderProcess esistente in modo che i punti bonus totali accumulati durante gli acquisti vengano visualizzati nella pagina di conferma dell'ordine.

Nota: Lo scopo di questo supporto didattico è l'illustrazione del processo di modifica di un comando di controller esistente. Non è stato progettato per illustrare il metodo più efficace per modificare il processo di elaborazione di un ordine all'interno del flusso di acquisti. A questo scopo, con WebSphere Commerce viene fornito il comando di attività `ExtOrderProcess` che può essere utilizzato per modificare il processo di elaborazione di un ordine.

Prima di iniziare questo supporto didattico
Completare tutte le procedure descritte in Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale" a pagina 207.

Nel seguente elenco sono riportate le operazioni che consentono di estendere il comando OrderProcess:

1. Creazione di un nuovo pacchetto nel quale è memorizzato il codice personalizzato. Tutti i codici personalizzati (per i bean di dati e di comandi) devono essere memorizzati in progetti e pacchetti diversi da quelli del codice di WebSphere Commerce.
2. Creazione di una classe `OrderProcessCmdBonusImpl` che estende il comando `OrderProcessCmdImpl`.
3. Aggiunta di campi e metodi alla classe `OrderProcessCmdBonusImpl`.
4. Modifica del registro dei comandi per utilizzare la classe `OrderProcessCmdBonusImpl`
5. Modifica della maschera confirmation.jsp per visualizzare la nuova logica aziendale.
6. Test della nuova logica aziendale all'interno di WebSphere Test Environment.

7. (Facoltativo) Distribuzione della nuova logica aziendale in un negozio su un WebSphere Commerce Server remoto.

Creazione di un nuovo pacchetto per OrderProcessCmdBonusImpl

Per creare il nuovo pacchetto in cui memorizzare il comando OrderProcessCmdBonusImpl, effettuare le seguenti operazioni:

1. Nella finestra Spazio di lavoro di VisualAge per Java assicurarsi che il separatore relativo ai progetti sia selezionato.
2. Fare clic con il pulsante destro del mousesul progetto **_WCSamples** e selezionare **Aggiungi > Pacchetto**.
   Si apre la SmartGuide per l'aggiunta di un pacchetto.
3. Assicurarsi che il pulsante di selezione **Creare un nuovo pacchetto denominato** sia abilitato e immettere com.ibm.commerce.sample.order.
4. Fare clic su **Fine**.

Creazione della classe OrderProcessCmdBonusImpl

Per creare la nuova classe OrderProcessCmdBonusImpl, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sul pacchetto com.ibm.commerce.sample.order e selezionare **Aggiungi > Classe**.
   Si apre la SmartGuide per la creazione delle classi.
2. Assicurarsi che il pulsante di selezione **Crea una nuova classe** sia selezionato.
3. Nel campo **Nome classe** immettere OrderProcessCmdBonusImpl.
4. Per specificare la superclasse, fare clic su **Sfoglia**, quindi nel campo **Modello** immettere com.ibm.commerce.order.commands.OrderProcessCmdImpl e fare clic su **OK**.
5. Fare clic su **Avanti**.
6. Per specificare i pacchetti da importare, fare clic su **Aggiungi pacchetto**. Nel campo **Modello**, immettere i seguenti pacchetti:
   - com.ibm.commerce.datatype e fare clic su **Aggiungi**
   - com.ibm.commerce.exception e fare clic su **Aggiungi**
   - com.ibm.commerce.order.commands e fare clic su **Aggiungi**
   - com.ibm.commerce.order.objects e fare clic su **Aggiungi**
   - com.ibm.commerce.ras e fare clic su **Aggiungi**
   - com.ibm.commerce.server e fare clic su **Aggiungi**
   - com.ibm.commerce.sample.objects e fare clic su **Aggiungi**
   - javax.ejb e fare clic su **Aggiungi**
   - java.io e fare clic su **Aggiungi**
   - java.math e fare clic su **Aggiungi**, quindi su **Chiudi**.

7. Per specificare le interfacce che la classe deve implementare, fare clic su **Aggiungi**. Nel campo Modello immettere la seguente interfaccia:
   - `OrderProcessCmd` e fare clic su **Aggiungi**, quindi selezionare **Chiudi**.
8. Fare clic su **Fine**.

Aggiunta di campi e metodi a `OrderProcessCmdBonusImpl`

È necessario aggiungere alla classe due campi e un metodo performExecute.

Per aggiungere il campo `theOrder` alla classe `OrderProcessCmdBonusImpl`, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sulla classe `OrderProcessCmdBonusImpl` e selezionare **Aggiungi > Campo**. Si apre la SmartGuide per la creazione dei campi.
2. Aggiungere un campo alla classe utilizzando gli attributi riportati di seguito. Per maggiori dettagli sulla creazione di un nuovo campo, fare riferimento a "Creazione di nuovi campi" a pagina 220

Nome attributo	**Valore**
Nome campo	`theOrder`
Tipo campo	`OrderAccessBean`
Valore iniziale	Non specificare.
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`selezionato`
Getter	`public`
Setter	`public`

   e fare clic su **Fine**.

Per aggiungere il campo `bonusPercentAmount` alla classe `OrderProcessCmdBonusImpl`, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sulla classe `OrderProcessCmdBonusImpl` e selezionare di nuovo **Aggiungi > Campo**.
2. Aggiungere un campo alla classe utilizzando gli attributi riportati di seguito. Per maggiori dettagli sulla creazione di un nuovo campo, fare riferimento a "Creazione di nuovi campi" a pagina 220

Nome attributo	**Valore**
Nome campo	`bonusPercentAmount`
Tipo campo	`double`
Valore iniziale	`1000`

Nome attributo	Valore
Modifier di accesso	`private`
Altri modifier	Non selezionare alcuna opzione.
Accedi mediante i metodi getter e setter	`selezionato`
Getter	`public`
Setter	`public`

e fare clic su **Fine**.

Per aggiungere il metodo `performExecute` alla classe `OrderProcessCmdBonusImpl`, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sulla classe **`OrderProcessCmdBonusImpl`** e selezionare **Aggiungi > Metodo**.
   Si apre la SmartGuide per la creazione dei metodi.
2. Assicurarsi che il pulsante di selezione **Crea un nuovo metodo** sia selezionato quindi fare clic su **Avanti**.
3. Nel campo **Nome metodo** immettere `performExecute`.
4. Da **Tipo di risultato** elenco a discesa, selezionare `void`. Fare clic su **Avanti**.
5. Per specificare l'eccezione che potrebbe essere restituita dal metodo, fare clic su **Aggiungi**. Nel campo **Modello** immettere `ECException`, fare clic su **Aggiungi** e, quindi, su **Chiudi**.
6. Fare clic su **Fine**.
   Il metodo viene generato e viene visualizzato il relativo codice di origine.
7. È necessario modificare il codice di origine. Posizionare la seguente riga nel codice di origine del metodo performExecute:

   ```
   public void performExecute() throws
      com.ibm.commerce.exception.ECException {
   ```

   Dopo l'immissione di tale riga, immettere il seguente codice:

È possibile tagliare e incollare questo codice dalla versione PDF della Guida per il programmatore. Si consiglia inizialmente di copiare il codice nella finestra Scrapbook in VisualAge per Java (per ulteriori informazioni, fare riferimento alla guida in linea VisualAge per Java) e controllare il codice per assicurarsi che non siano stati persi o modificati caratteri durante l'operazione di taglia e incolla. Quindi, una volta controllato il codice, copiarlo nell'ubicazione di destinazione. La copia del testo in un altro editor può comportare la modifica di alcuni caratteri.

```
final String methodName = "performExecute";
ECTrace.entry(ECTraceIdentifiers.COMPONENT_ORDER,
   this.getClass().toString(), methodName);
```

```
// do all order processing as normal
super.performExecute();

// *** updating Bonus Point Information ***

// fetch order info
theOrder = new OrderAccessBean();
theOrder.setInitKey_orderId(getOrderRn().toString());

int bonusPt; // bonus points for this order
int bonusTotal; // total bonus points
BigDecimal subtotal;  // subtotal
BigDecimal bonusdeter;  // bonus determinant
BigDecimal ans;

// determine bonus points = subtotal * bonus determinant
try {
     subtotal = theOrder.getTotalProductPriceInEJBType();
     bonusdeter = new BigDecimal(bonusPercentAmount);
     ans = subtotal.multiply(bonusdeter);
     bonusPt = Math.round(ans.floatValue());

     System.out.println("subtotal is: " + subtotal +
        " bonus deter is: " + bonusdeter + " ans is: " + ans);
     System.out.println("Bonus Percent amount = " +
        bonusPercentAmount);
     System.out.println("Bonus calculated is: "+ bonusPt);
    }


    // Various Exceptions
   catch (Exception ex) {
   throw new ECSystemException(ECMessage._ERR_GENERIC,
      this.getClass().toString(), methodName,
      ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
   }

// *** Updating bonus points in BONUS table using bean
//     created in previous example ***

BonusAccessBean bonusBean = new BonusAccessBean();
bonusBean.setInit_argMemberId(
   getCommandContext().getUserId().toString());
try {
    //new bonus value = this order bonus points + Old Bonus Points
      bonusTotal = bonusPt + Integer.parseInt(bonusBean.getBonusPoint());
      bonusBean.setBonusPoint(String.valueOf(bonusTotal));
    bonusBean.commitCopyHelper();

    System.out.println("In try, BonusTotal calculated is: "+
           bonusTotal);

    }

 // Various exceptions
```

```
        catch (FinderException e) // user does not have points setup yet
        {
           // create a row in table bonus
             bonusTotal = bonusPt;

           try {
               BonusAccessBean bonusBeanNew = new
                  BonusAccessBean(getCommandContext().getUserId(),
                     new Integer(bonusTotal));

                  System.out.println("In catch, BonusTotal calculated is: "+
                     bonusTotal);

        }
        catch (Exception ex) {
           throw new ECSystemException(ECMessage._ERR_GENERIC,
              this.getClass().toString(), methodName,
              ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
           }
     }
        catch (Exception ex) {
            throw new ECSystemException(ECMessage._ERR_GENERIC,
              this.getClass().toString(), methodName,
              ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
        }

        // *** setting view details ***

        // Fetch setResponse properties and add bonus parameters
       //  needed by the JSP page
         TypedProperty resp = getResponseProperties();
         resp.put("bonus", new Integer(bonusPt).toString());
        setResponseProperties(resp);
        ECTrace.exit(ECTraceIdentifiers.COMPONENT_ORDER,
           this.getClass().toString(), methodName);
```

8. Salvare il proprio lavoro.

Modifica del registro dei comandi per utilizzare `OrderProcessCmdBonusImpl`

In questo esempio, si desidera utilizzare la nuova classe di implementazione per l'elaborazione degli ordini nel caso in cui fosse prevista. Affinché ciò si verifichi, è necessario aggiornare il registro dei comandi in modo da associare l'interfaccia OrderProcess originale alla nuova classe di implementazione `OrderProcessCmdBonusImpl`.

DB2 Se si utilizza un database DB2, effettuare le seguenti operazioni per aggiornare il registro comandi:

1. Aprire il Centro comandi DB2 (**Start > Programmi >IBM DB2 > Centro di comandi**).
2. Dopo aver selezionato la scheda Script, creare la voce richiesta nella tabella CMDREG immettendo le seguenti informazioni nella finestra dello script:

```
connettersi a nome_database;
aggiornare CMDREG
impostare CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0;
```

dove *nome_database* è il nome del database in uso; quindi, fare clic sull'icona dell'esecuzione
Questo comando viene utilizzato da tutti i venditori (indicati dal valore 0 per STOREENT_ID).

Oracle Se si utilizza un database Oracle, effettuare le seguenti operazioni per aggiornare il registro comandi:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password**, immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.
5. Creare la voce richiesta nella tabella URLREG, immettendo le seguenti informazioni nella finestra SQL Plus:

   ```
   aggiornare CMDREG
   impostare CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0;
   ```

   Premere Invia per eseguire l'istruzione SQL
   Questo comando viene utilizzato da tutti i venditori (indicati dal valore 0 per STOREENT_ID)
6. Immettere quanto segue per confermare le modifiche al database:

   ```
   commit;
   ```

   e premere Invia per eseguire l'istruzione SQL.

Modifica della maschera `confirmation.jsp`

Per visualizzare la nuova logica aziendale aggiunta al processo aziendale di elaborazione di un ordine, è necessario modificare la maschera `confirmation.jsp`. Per modificare la maschera di visualizzazione, effettuare le seguenti operazioni:

1. Spostarsi nella seguente directory:
   `unità_vaj:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\`*`directory_negozio`*.
2. Eseguire una copia di `confirmation.jsp` e denominarla `confirmation.jsp.bak`
3. Aprire il file `confirmation.jsp` con un editor di testo.

4. Dopo le istruzioni di importazione esistenti, aggiungere quanto segue:

```
<%@ page import="com.ibm.commerce.datatype.*" %>
```

5. Subito dopo la seguente riga nella maschera JSP:

```
String orderRn = jhelper.getParameter("orderId");
```

   aggiungere quanto segue:

```
String bonus = ((TypedProperty)request.getAttribute(
   ECConstants.EC_REQUESTPROPERTIES)).getString("bonus");
```

6. Individuare la seguente sezione nella maschera JSP:

```
<tr>
<td align="left" valign="middle">
<font class="product"><%=infashiontext.getString("GRAND_TOTAL")%>
   </font></td>
<td align="right" valign="middle">
<font class="strongprice"><%=orderBean.getGrandTotal()  %></font></td>
```

   e aggiungere quanto segue:

```
</tr>
<tr>
<td align="left" valign="middle">
<font class="text">Bonus Points</font></td>
<td align="right" valign="middle">
<font class="strongtext"><%=bonus  %></font></td>
```

7. Salvare il proprio lavoro.

Test di `OrderProcessCmdBonusImpl` all'interno di WebSphere Test Environment

È quindi possibile utilizzare WebSphere Test Environment per effettuare il test della nuova logica aziendale. Per controllare il comando `OrderProcessCmdBonusImpl`, effettuare le seguenti operazioni:

1. Avviare il WebSphere Test Environment, come descritto in Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347.
2. Avviare un browser e immettere l'URL del negozio. Ad esempio, immettere il seguente URL:

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=10001&catalogId=10001&langId=-1
```

3. Selezionare e acquistare un prodotto.
4. Una volta acquistato un prodotto, con la conferma dell'ordine viene visualizzato il numero di punti bonus guadagnati con l'ordine.

(Facoltativo) Distribuzione di una logica aziendale personalizzata su un WebSphere Commerce Server remoto

Una volta completato il test della logica aziendale in WebSphere Test Environment e confermato il proprio codice, è possibile distribuire tale codice in un negozio presente su un WebSphere Commerce Server remoto. In questo supporto didattico le opzioni di personalizzazione comprendono le seguenti:

- La nuova classe OrderProcessCmdBonusImpl.
- Il file della maschera confirmation.jsp aggiornato.
- Il registro comandi aggiornato.

La distribuzione del codice comprende i seguenti passaggi:

1. Creazione di un file JAR per la logica del comando mediante i tool di VisualAge per Java.
2. Copia del file JAR e della maschera JSP nelle relative directory sul WebSphere Commerce Server di destinazione.
3. Aggiornamento del registro comandi del WebSphere Commerce Server di destinazione.

Note sul metodo di pagamento di prova

Il negozio di esempio in esecuzione in WebSphere Test Environment utilizza per impostazione predefinita un metodo di pagamento di prova. Questo metodo consente di completare il flusso di acquisti in WebSphere Test Environment, senza richiamare un Payment Manager. Questo metodo consente di completare solo un acquisto e non permette l'elaborazione degli ordini inoltrati. Pertanto, questo metodo di pagamento può essere utilizzato solo inWebSphere Test Environment.

Accertarsi che sia possibile completare un acquisto nel negozio in cui si sta distribuendo questo codice personalizzato. Il processo di pagamento può essere eseguito utilizzando un Payment Manager locale o remoto.

Per ulteriori informazioni sul metodo di pagamento di prova, consultare "Metodo di pagamento di prova" a pagina 203.

Creazione del file JAR per la logica del comando

È necessario impacchettare la logica del comando in un file JAR in modo che questo possa essere distribuito al WebSphere Commerce Server di destinazione. Dal momento che `OrderProcessCmdBonusImpl` è memorizzato nel progetto _WCSamples, viene creato un file JAR relativo a questo progetto.

Per creare il file JAR, effettuare le seguenti operazioni:

1. Arrestare WebSphere Test Environment, come descritto in Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347.
2. Fare clic con il pulsante destro del mousesul progetto **_WCSamples** e selezionare **Esporta**.
   Si apre la SmartGuide per l'esportazione.
3. Selezionare **file Jar** e fare clic su **Avanti.**
4. Nel campo File jar immettere quanto segue:
   unità`:\WebSphere\CommerceServerDev\mytemp_b\wcssamplesb_1.jar`
   dove *unità* è l'unità sulla quale è installato Commerce Studio.

5. Selezionare gli attributi nel seguente modo:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
bean	Selezionato
Includi attributi di debug nei file .class	Selezionato

   Accettare i valori predefiniti per gli altri attributi.
6. Fare clic su **Fine**.

Poiché il file JAR creato non contiene le informazioni complete sulla denominazione dei pacchetti è necessario utilizzare un altro programma di utilità per i pacchetti (esterni a VisualAge per Java) per creare nuovi pacchetti del file JAR. Per creare un nuovo pacchetto di questo file, eseguire le procedure descritte:

1. Da una finestra comandi, passare alla seguente directory:
   unità:\WebSphere\CommerceServerDev\mytemp_b
2. Immettere `mkdir temp1`.
3. Immettere `cd temp1`.
4. Impostare il percorso come specificato di seguito:
   `set PATH=%PATH%;`*unità*`:\WebSphere\WebSphereStudio4\bin;`
   dove *unità* è l'unità sulla quale è stato installato WebSphere Studio.
5. Immettere `jar xvf ../wcssamplesb_1.jar`
6. Immettere `jar cvf ../wcssamplesb.jar *`

Memorizzazione delle attività sul WebSphere Commerce Server di destinazione

Il file JAR per la logica del comando e la maschera `confirmation.jsp` modificata devono essere posizionate nelle directory appropriate sul WebSphere Commerce Server di destinazione.

Per memorizzare il file JAR nella directory appropriata su un WebSphere Commerce Server remoto, effettuare le seguenti operazioni:

1. Sulla macchina di sviluppo, aprire una finestra di comandi e passare alla seguente directory:
   unità:\WebSphere\CommerceServerDev\mytemp_b
   e individuare il file `wcssamplesb.jar`.
2. Copiare questo file nella seguente directory sul WebSphere Commerce Server di destinazione:

```
unità:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_Nomeistanza.ear\wcstores.war\WEB-INF\lib
```

Per memorizzare la maschera confirmation.jsp nella directory appropriata sul WebSphere Commerce Server di destinazione, effettuare le seguenti operazioni:

1. Sulla macchina di sviluppo, spostarsi alla seguente directory:
   unità_vaj:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*directory_negozio*
2. Copiare la maschera confirmation.jsp nella seguente directory sul WebSphere Commerce Server di destinazione:

   ```
   unità:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_Nomeistanza.ear\wcstores.war\Dirnegozio
   ```

Aggiornamento del registro comandi

Se si distribuisce il comando OrderProcessCmdBonusImpl a un WebSphere Commerce Server di destinazione che utilizza un database diverso da quello utilizzato dal WebSphere Test Environment, è necessario aggiornare il database di destinazione in modo da confermare le modifiche effettuate nel registro comandi.

DB2 Se si utilizza un database DB2, effettuare le seguenti operazioni per aggiornare il database del WebSphere Commerce Server di destinazione:

1. Aprire il Centro comandi DB2 (**Start > Programmi >IBM DB2 > Centro di comandi**).
2. Dopo aver selezionato la scheda Script, creare la voce richiesta nella tabella CMDREG immettendo le seguenti informazioni nella finestra dello script:

   ```
   connettersi a nome_database_destinazione;
   aggiornare CMDREG
   impostare CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0
   ```

   dove *nome_database_destinazione* è il nome del database, quindi fare clic sull'icona di esecuzione

Oracle Se si utilizza un database Oracle, effettuare le seguenti operazioni per aggiornare il registro comandi:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password**, immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.

5. Creare la voce richiesta nella tabella CMDREG, immettendo le seguenti informazioni nella finestra SQL Plus:

   ```
   aggiornare CMDREG
   impostare CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0;
   ```

   Fare clic su Invia per eseguire l'istruzione SQL.
   Questo comando viene utilizzato da tutti i venditori (indicato dal valore 0 per STOREENT_ID)
6. Immettere quanto segue per confermare le modifiche al database:

   ```
   commit;
   ```

   e premere Invia per eseguire l'istruzione SQL.

Riavvio di enterprise application in WebSphere Application Server

Una volta aggiunta la logica dei comandi a enterprise application inserendo le risorse file nelle directory appropriate e aggiornando il registro comandi, è necessario arrestare e riavviare enterprise application per rendere effettive le modifiche.

Per arrestare e riavviare enterprise application, attenersi alla seguente procedura:

1. Aprire la WebSphere Application Server Console di gestione.
2. Espandere **Dominio di gestione WebSphere**.
3. Espandere **Nodi**.
4. Espandere *Nomenodo* (dove *Nomenodo* è il nome del nodo).
5. Espandere **Server delle applicazioni**.
6. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Server - demo** e selezionare **Arresta**.
7. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Server - demo** e selezionare **Avvia**.

Test della nuova logica nel negozio InFashion in esecuzione nel WebSphere Application Server

È quindi possibile verificare la nuova logica aziendale nel negozio InFashion in esecuzione nel WebSphere Application Server.

Per eseguire questa verifica finale, effettuare le seguenti operazioni:

1. Una volta avviata l'istanza WebSphere Commerce Server, avviare un browser ed immettere l'URL della home page del negozio. Ad esempio, immettere il seguente URL:

   ```
   http://nomehost/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=id_negozio&catalogId=Id_catalogo&langId=-1
   ```

   dove *Id_negozio* è l'ID del negozio e *Id_catalogo* è l'ID del catalogo del negozio.
2. Selezionare e acquistare un prodotto.
3. Una volta acquistato un prodotto, con la conferma dell'ordine viene visualizzato il numero di punti bonus guadagnati con l'ordine.

Modifica di un bean entità esistente ed estensione di un comando di attività esistente

Nota: Lo scopo di questo supporto didattico è di illustrare il processo di modifica dei bean entità esistenti e di estendere i comandi di attività esistenti. Non è stato progettato per illustrare il miglior modo per modificare il processo di assegnazione del prezzo a un prodotto. Per ulteriori informazioni sui prezzi da scontare, consultare il manuale *WebSphere Commerce Calculation Framework Guide*.

Nell'esercitazione Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale", è stata creata una nuova sezione di logica aziendale. Tale esercitazione comprendeva la creazione di un nuovo comando di controller, una nuova maschera JSP, una nuova tabella di database, un nuovo bean enterprise per l'accesso alla tabella e un bean di accesso corrispondente e di un bean di accesso ai dati. Tutti questi elementi di logica sono stati raggruppati per creare un'applicazione di punti di bonus semplificata, nella quale è possibile aggiornare i punti di bonus di n utente utilizzando una maschera JSP che viene attivata dal nuovo comando di controller.

Il modo in cui la tabella BONUS viene utilizzata in Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale" viene mostrato nel seguente diagramma:

Utilizzo della tabella BONUS nel supporto didattico ‘Creazione di una nuova logica aziendale”

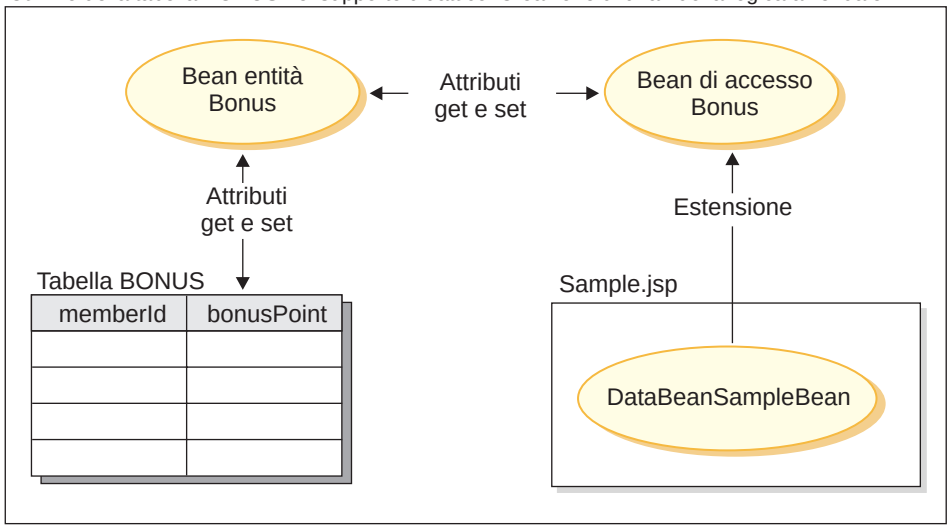


Figura 41.

Nel supporto didattico successivo, la tabella BONUS viene utilizzata in un modo diverso. Specificamente, il bean entità utente viene personalizzato in modo tale da venire visualizzato alle applicazioni con la colonna BONUSPOINT della tabella BONUS che è una colonna nella tabella USERS. Quando viene creato un nuovo record nella tabella USERS, nella tabella BONUS viene creato automaticamente un record corrispondente.

Per creare questa *unione di tabella*, deve essere aggiunto un nuovo campo CMP al bean entità utente. Questo campo CMP viene associato alla colonna BONUSPOINT nella tabella BONUS utilizzando funzione Mappa tabella secondaria nel browser di corrispondenze di VisualAge per Java.

Per integrare il bean entità utente modificato nel flusso di acquisto, viene creato un nuovo prezzo dei prodotti. I nuovi prezzi tengono in considerazione il bilancio di punti di bonus dell’acquirente. Creare il nuovo prezzo estendendo il comando di attività GetProductContractUnitPriceCmd. Quando si estende questo comando, viene creata una nuova interfaccia che estende l’interfaccia GetProductContractUnitPriceCmd. La nuova interfaccia aggiunge un ulteriore attributo (per il prezzo del bonus). Inoltre, viene creata una nuova classe di implementazione che estende la classe di implementazione GetContractUnitPriceCmdImpl. Questa nuova classe di implementazione si chiama “GetNewContractUnitPriceCmdImpl”. La nuova classe di implementazione richiama il metodo performExecute della propria superclasse e, quindi, aggiunge la logica aziendale per determinare il nuovo prezzo del bonus.

Il seguente diagramma mostra il modo in cui la tabella BONUS viene utilizzata nel successivo supporto didattico.

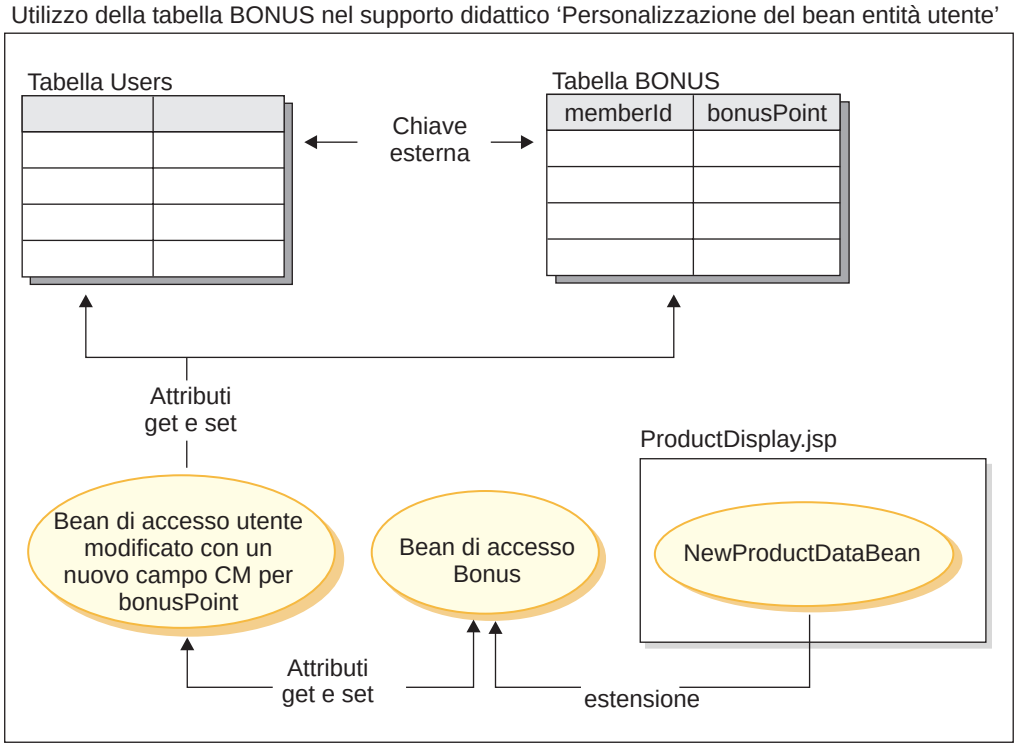


Figura 42.

Questo supporto didattico include i seguenti passi:

1. Aggiungere un nuovo CMP al nuovo bean entità utente.
2. Creare la tabella BONUS e inserirvi i dati.
3. Aggiornare lo schema di database WCSUser e l'associazione tabella in modo da includere la nuova tabella BONUS.
4. Creare la relazione di chiave esterna tra le tabelle BONUS e USER.
5. Creare l'associazione tabella BONUS.
6. Generare il codice di distribuzione e il bean di accesso per il bean entità utente.
7. Utilizzare il client di verifica per un controllo preliminare del bean entità aggiornato.
8. Creare una nuova interfaccia comandi di attività e una nuova classe di implementazione. Il nuovo comando attività estende `GetBaseUnitProductPriceCmd`. La funzione più importante in questo comando è la logica nel metodo performExecute() della classe di

implementazione. Questo metodo calcola un nuovo *prezzo bonus* per il prodotto. Tale prezzo bonus viene calcolato riducendo il prezzo in base al saldo dei punti di bonus dell'acquirente, fino a un massimo del 20% di riduzione sul prezzo calcolato. La seguente tabella mostra esempi di questa formula di riduzione del prezzo:

Prezzo calcolato	**Bilanciamento punti bonus**	**Riduzione massima (20% del prezzo calcolato)**	**Nuovo prezzo bonus**
£2.000.000	100	£400.000	£1.800.000 Poiché il saldo dei punti di bonus è inferiore alla riduzione massima, l'elenco dei prezzi viene ridotto in base ai punti di bonus.
£2.000.000	300	£400.000	£1.600.000 Poiché il saldo dei punti di bonus supera la riduzione massima, l'elenco dei prezzi viene ridotto in base alla riduzione massima di £400.000...

9. Creazione di `NewProductDataBean` che estende `ProductDataBean`. Aggiunta di un nuovo metodo a questo bean in modo da poter utilizzare facilmente il nuovo prezzo bonus in una pagina di visualizzazione del prodotto.
10. Aggiornamento della maschera JSP di visualizzazione del prodotto per il negozio InFashion per mostrare il prezzo bonus.
11. Test della logica aziendale del negozio InFashion in esecuzione all'interno di WebSphere Test Environment.
12. (Facoltativo) Distribuzione della logica aziendale aggiornata al WebSphere Commerce Server e test all'esterno di WebSphere Test Environment.

Prima di iniziare questo supporto didattico

Se non è stato completato Capitolo 9, “Supporto didattico: creazione di nuova logica aziendale” a pagina 207, prima di iniziare questo supporto didattico, è necessario eseguire i passi descritti in “Preparazione del progetto di esempio” a pagina 208.

Aggiunta di un nuovo campo bonusPoint al bean entità utente

In questa sezione vengono utilizzati gli strumenti EJB di VisualAge per Java per aggiungere il nuovo campo CMP al bean entità. Il nuovo campo viene denominato *bonusPoint* e può essere associato alla colonna BONUSPOINT della tabella BONUS.

Per aggiungere il nuovo campo CMP al bean entità utente, eseguire le seguenti operazioni:

1. Se sono in esecuzione, arrestare il motore servlet, il server EJB e il server nomi permanenti, come descritto in Appendice A, “Avvio e interruzione di WebSphere Test Environment” a pagina 347.
2. Nello spazio di lavoro, fare clic sulla scheda **EJB**.
3. Espandere il gruppo EJB **WCSUser**.
4. Fare clic con il pulsante destro del mouse sul bean **User** e selezionare **Aggiungi > campo CMP**.
   Si apre la SmartGuide per la creazione del campo CMP.
5. Creare un nuovo campo CMP con le seguenti proprietà:

Proprietà	Valore
Nome campo	`bonusPoint`
Tipo campo	`int`
Valore iniziale	`0`
Accedi mediante i metodi getter e setter	`enable`
Promuovi metodi getter e setter per l’interfaccia remota	`enable`
Getter	`public`
Setter	`public`

e fare clic su **Fine**.

Il campo bonusPoint viene visualizzato nel pannello Proprietà. Potrebbero venire visualizzati dei messaggi di avvertenza, ma questi vengono risolti quando si genera nuovamente il codice del bean entità.

Creazione e riempimento della tabella BONUS

In questo supporto didattico viene utilizzata una nuova tabella di database che registra i punti di bonus di un utente.

Se è stato completato Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale" a pagina 207, si conosce già questa tabella. In questo caso, è necessario aggiornare la tabella per aggiungere una riga per ogni utente nella tabella USERS. Se non viene completato Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale" a pagina 207, è necessario creare e riempire la tabella. Vengono fornite le istruzioni per ognuno di questi scenari.

DB2 Se si utilizza un database DB2 e occorre *aggiornare* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire il Centro di comandi DB2 (**Start > Programmi > IBM DB2 > Centro di comandi**) e fare clic sulla scheda **Script**.
2. Nel campo **Command** immettere

   ```
   connect to nome_database
   ```

   dove *nome_database* è il nome del database e fare clic sull'icona di esecuzione.
3. Nel campo **Command** immettere quanto segue, quindi, fare clic sull'icona di esecuzione:

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
   ```

   La tabella BONUS è quindi stata aggiornata.

DB2 Se si utilizza un database DB2 e occorre *creare e popolare* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire il Centro di comandi DB2 (**Start > Programmi > IBM DB2 > Centro di comandi**) e fare clic sulla scheda **Script**.
2. Nel campo **Command** immettere

   ```
   connect to nome_database
   ```

   dove *nome_database* è il nome del database e fare clic sull'icona di esecuzione.
3. Nel campo **Command** immettere quanto segue, quindi, fare clic sull'icona di esecuzione:

```
CREATE TABLE BONUS (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (id_utente) on delete cascade)
```

La tabella BONUS è stata creata.

4. Per riempire questa tabella, immettere quanto segue nel campo **Command**, quindi, fare clic sull'icona di esecuzione:

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```

5. Chiudere il Centro di comandi DB2.

Oracle Se si utilizza un database Oracle ed è necessario *aggiornare* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password**, immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.
5. Aggiornare la tabella BONUS, immettendo le seguenti informazioni nella finestra SQL Plus:

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
   ```

   Fare clic su Invia per eseguire l'istruzione SQL.
   La tabella BONUS è ora aggiornata.

6. Immettere quanto segue per confermare le modifiche al database:

   ```
   commit;
   ```

   e premere Invia per eseguire l'istruzione SQL.

Oracle Se si utilizza un database Oracle ed è necessario *creare e riempire* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password**, immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.
5. Creare la tabella BONUS, immettendo le seguenti informazioni nella finestra SQL Plus:

```
CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

Fare clic su Invia per eseguire l'istruzione SQL.
La tabella BONUS è stata creata.

6. Riempire la tabella BONUS immettendo quanto segue:

```
insert into BONUS (select USERS_ID, 0 from USERS);
```

7. Immettere quanto segue per confermare le modifiche al database:

```
commit;
```

e premere Invia per eseguire l'istruzione SQL.

Aggiornamento dello schema e della corrispondenza tabella

Nelle seguenti sezioni, saranno esposte le procedure per l'aggiornamento dello schema WCSUser con la nuova tabella BONUS, per la creazione della relazione della chiave esterna per la nuova tabella e della corrispondenza di tabella tra i campi del bean entità User e le colonne della tabella BONUS.

Creazione dello schema tabella BONUS

Per creare lo schema di tabella, effettuare le seguenti operazioni:

1. Fare clic con il pulsante destro del mouse sul bean entità **User** e selezionare **Apri in > Schemi di database**.
   Si apre la finestra Browser schema.
2. Dall'elenco **Schemi**, selezionare **WCS User**.
3. Dal menu **Tabelle**, selezionare **Nuova tabella**.
   Si apre l'editor delle tabelle.
4. Nel campo **Nome**, immettere `BONUS`.
5. Lasciare vuoti i campi **Qualificatore** e **Nome fisico**.
6. Nella sezione relativa alle colonne della tabella, fare clic su **Nuovo**. Si apre l'editor delle colonne. Creare una colonna effettuando le seguenti operazioni:

Attributo	**Valore**
Nome	`memberId`
Nome fisico	Lasciare questo campo vuoto.
Tipo	`BIGINT`
Dettagli tipo	`VapConverter`
Consenti valori null	non selezionato

e fare clic su **OK**.

7. Selezionare **memberId** nell'elenco delle colonne Tabella e fare clic su **>>** per utilizzare questa colonna come chiave principale.
8. Creare un'altra colonna (fare di nuovo clic su **Nuovo**) effettuando le seguenti operazioni:

Attributo	**Valore**
Nome	`bonusPoint`
Nome fisico	Lasciare questo campo vuoto.
Tipo	`INTEGER`
Dettagli tipo	`VapConverter`
Consenti valori null	selezionato

   e fare clic su **OK**.
9. Fare clic su **OK** nell'Editor tabelle.
   Dopo alcuni secondi la tabella **BONUS** viene elencata nell'elenco delle **Tabelle** della finestra del browser di schema.
10. Per una verifica, fare clic su **BONUS** nell'elenco **Tabelle** e assicurarsi che le due colonne contenute siano visualizzate nell'elenco **Colonne**.

Creazione della relazione di chiavi esterne

In questa sezione viene esposta la procedura di creazione della relazione chiave esterna tra le tabelle BONUS e USERS.

Per creare la relazione chiave esterna, effettuare le seguenti operazioni:

1. Nella finestra Browser schema, fare clic sullo schema **Utente WCS**.
   Vengono visualizzate le tabelle e le relazioni chiave esterna relative a questo schema.
2. Dal menu **Chiavi esterne**, selezionare **Nuova relazione chiave esterna**.
   Si apre l'editor della relazione chiave esterna.
3. Nel campo **Nome**, `F_User_Bonus`.
4. Assicurarsi che la casella di controllo **Limite nel database** sia contrassegnata.
5. Dall'elenco a discesa **Tabella chiave principale**, selezionare `USERS`
6. Dall'elenco a discesa **Tabella chiave esterna**, selezionare `BONUS`
7. Fare clic sul campo vuoto sotto l'intestazione **Chiave esterna**, in modo da visualizzare un elenco a discesa. Da questo elenco selezionare **memberId** e fare clic su **OK**.
   `F_User_Bonus` viene visualizzato nell'elenco delle relazioni di chiavi esterne.
8. Dal menu **Schemi**, selezionare **Salva schema**, quindi fare clic su **Fine**.
9. Chiudere Browser schema.

Creazione della corrispondenza della tabella BONUS

In questa sezione, viene creata la corrispondenza tra la colonna `BONUSPOINT` nella tabella BONUS e il campo `bonusPoint` nel bean entità utente.

Per creare la corrispondenza di tabella BONUS, effettuare le seguenti operazioni:

1. Assicurarsi che la scheda EJB sia selezionata quando si apre lo spazio di lavoro.
2. Dal menu **EJB**, selezionare **Apri in > Mappe schema**.
   Si apre il browser di corrispondenze.
3. Dall'elenco Mappe datastore, selezionare **Utente WCS**.
4. Nell'elenco **Classi permanenti** eseguire le seguenti operazioni:
   a. Fare doppio clic su **Membro**.
   b. Selezionare **Utente** (si osservi che **Utente** viene visualizzato sotto **Membro** solo dopo aver espanso Membro nel passo 4a.)
5. Dal menu **Mappe tabella**, selezionare **Nuova mappa tabella> Aggiungi mappa tabella secondaria**.
   Si apre la finestra Mappa tabella secondaria.
6. Dall'elenco a discesa **Tabella**, selezionare **BONUS**.
7. Dall'elenco a discesa Relazione chiave esterna, selezionare **F_User_Bonus** e fare clic su **OK**.
   Dopo alcuni secondi, viene visualizzata la corrispondenza `BONUS (secondaria)` nell'elenco delle di corrispondenze di tabella.
8. Evidenziare e, quindi, fare clic con il pulsante destro del mouse sulla corrispondenza di tabella**BONUS (secondaria)** e selezionare **Modifica corrispondenze di proprietà**.
   Si apre l'editor di corrispondenze di proprietà.
9. Scorrere fino alla riga che visualizza l'attributo di classe bonusPoint. Selezionare **bonusPoint**.
10. Fare clic sulla voce corrispondente nella colonna **Tipo di corrispondenza** e selezionare **Semplice** da elenco a discesa.
11. Fare clic sulla voce corrispondente nella colonna **Colonna tabella** e selezionare **bonusPoint** da elenco a discesa.
12. Lasciare tutti gli altri campi inalterati e fare clic su **OK**.
    Viene generata la nuova corrispondenza tabella e, dopo alcuni secondi viene visualizzato

    ```
    (a) bonusPoint (bonusPoint)
    ```

    nella colonna **corrispondenze proprietà** del browser di corrispondenze.
13. Fare clic con il pulsante destro del mouse sulla corrispondenza datastore **WCS User**, selezionare **Salva corrispondenza datastore**, quindi fare clic su **Fine**.

14. Chiudere il browser di corrispondenze.

A questo punto, il bean utente contiene errori nei quali viene indicato che alcune classi astratte non sono implementate. Questi errori vengono risolti quando viene generato il codice di distribuzione.

Generazione del codice di distribuzione e del bean di accesso

Poiché è stato modificato il codice per il bean entità User, è necessario generare nuovamente il codice di distribuzione e il relativo bean di accesso. Gli strumenti forniti da VisualAge per Java semplificano notevolmente la generazione del codice.

Per eseguire questa procedura, effettuare le seguenti operazioni:

1. Assicurarsi che la scheda EJB sia selezionata quando si apre lo spazio di lavoro.
2. Espandere il gruppo EJB **WCSUser**.
3. Fare clic con il pulsante destro del mouse sul bean **User** e selezionare **Genera Codice configurato**.
   se viene visualizzato un messaggio in cui viene chiesto se creare o meno un'edizione del pacchetto visualizzato, fare clic su **Sì**.
   Le attività di generazione codice richiedono alcuni minuti.
4. Una volta generato il codice di distribuzione, fare clic con il pulsante destro del mouse di nuovo sul bean **User** e selezionare **Aggiungi > Bean di accesso**. Viene aperta la SmartGuide per la creazione del bean di accesso.
5. Nella pagina Seleziona proprietà bean di accesso accettare i valori predefiniti e fare clic su **Avanti**.
6. Nella pagina Define Zero Argument Constructor accettare i valori predefiniti e fare clic su **Avanti**.
7. Nella pagina Seleziona e personalizza proprietà bean per copy helper, scorrere fino a **bonusPoint** nella colonna Bean enterprise. Selezionare **Copy Helper** per il bean e impostare il programma di conversione su `com.ibm.commerce.base.objects.WCSStringConverter`.
8. Fare clic su **Fine**.
9. Fare clic su **OK** quando viene visualizzato il messaggio"Generazione codice completata".

Verifica della modifica utilizzando il client del test

Per verificare il bean entità utente appena personalizzato è possibile utilizzare un client test. Per avviare il client test e verificare il bean entità, eseguire le seguenti operazioni:

1. Avviare il server dei nomi permanenti, come descritto in "Avvio e interruzione del server dei nomi permanenti" a pagina 347.

2. Avviare il server EJB sul quale si trova il gruppo EJB WCSUser, come descritto in "Avvio e interruzione del server EJB" a pagina 348.
3. Nel pannello Bean enterprise, espandere il gruppo EJB **WCSUser**. Fare clic con il pulsante destro del mouse sul bean entità **User** e selezionare **Esegui client test**.
4. Nella finestra Ricerca EJB fare clic su **Ricerca**.
5. Dall'elenco dei metodi, selezionare **findByPrimaryKey(MemberKey)**.
6. Nel pannello Dettagli, fare clic su **<null>**, quindi immettere `-1000` nella casella argomento e fare clic sull'icona Richiama nella finestra Client test EJB.

   Si osservi che il valore immesso deve corrispondere ad un valore presente nella colonna USERS_ID della tabella USERS.

   Una volta completata l'esecuzione di questo metodo, le informazioni per l'utente con l'ID di-1000 vengono visualizzate in una struttura ad albero nel pannello Metodi. In tale struttura, esiste un nodo denominato *methods*.
7. Espandere il nodo **Metodi**.
8. Fare clic su **getBonusPoint()**, quindi sull'icona Richiama.
   Questo metodo richiama il bilancio di punti bonus del cliente. Viene restituito il bilancio corrente dei punti bonus.
9. Fare clic su **setBonusPoint(int)**, inserire `100` nel pannello Dettagli e, quindi, fare clic sull'icona Richiama.
10. Fare clic su **getBonusPoint()** e, quindi, sull'icona Richiama.
    Viene restituito un valore di 100. Ciò mostra che il database è stato aggiornato correttamente dal bean enterprise utente.
11. Chiudere le finestre Client test EJB e User.

Creazione dell'interfaccia `GetNewProductContractUnitPriceCmd`

Come parte dell'esercizio di personalizzazione, viene creata la nuova logica aziendale per calcolare un prezzo scontato, in base al bilancio di punti bonus del cliente. Questo calcolo viene eseguito da un nuovo comando attività. In questa sezione, viene creata l'interfaccia per questo nuovo comando attività.

Per creare l'interfaccia per il nuovo comando attività, eseguire le seguenti operazioni:

1. Nello spazio di lavoro, con la scheda Progetti selezionata, espandere il progetto **_WCSamples**.
2. Fare clic con il pulsante destro del mouse sul pacchetto **com.ibm.commerce.sample.commands** e selezionare **Aggiungi > Interfaccia**.
3. Assicurarsi che **Crea una nuova interfaccia** sia selezionata e nel campo **Nome interfaccia**, immettere `GetNewProductContractUnitPriceCmd`.

4. Selezionare l'interfaccia da estendere facendo clic su **Aggiungi**, quindi nel campo **Modello** immettere `com.ibm.commerce.price.commands.GetProductContractUnitPriceCmd`, fare clic su **Aggiungi**, quindi su **Chiudi**.
5. Fare clic su **Avanti**.
6. Per aggiungere le istruzioni di importazione appropriate, fare clic su **Aggiungi pacchetto** e, quindi, eseguire le seguenti operazioni:
   a. Nel campo **Modello** immettere `com.ibm.commerce.price.utils` e fare clic su **Aggiungi**.
   b. Nel campo **Modello** immettere `com.ibm.commerce.exception` e fare clic su **Aggiungi**.
   c. Fare clic su **Chiudi**.
7. Assicurarsi che l'opzione **Rendi pubblica l'interfaccia** sia selezionata.
8. Fare clic su **Fine**.
   Il codice di origine per la nuova interfaccia viene visualizzato nel pannello Origine.
9. Creare una firma del metodo per il metodo `getBonusPrice()` effettuando le seguenti operazioni:
   a. Fare clic con il pulsante destro del mouse sull'interfaccia **GetNewProductContractUnitPriceCmd** e selezionare **Aggiungi > Metodo**.
      Si apre la SmartGuide per la creazione del metodo.
   b. Assicurarsi che l'opzione per la **creazione di un nuovo metodo** sia selezionata e fare clic su **Avanti**.
   c. Nel campo Nome metodo, immettere `getBonusPrice`.
   d. Per selezionare il tipo di risultato del metodo, fare clic su **Sfoglia**. Nel campo **Modello** immettere `MonetaryAmount` e fare clic su **OK**, quindi su **Avanti**.
   e. Per selezionare l'eccezione inviata dal metodo, fare clic su **Aggiungi**. Nel campo **Modello** immettere `ECSystemException` e fare clic su **Aggiungi**, quindi su **Chiudi**.
   f. Fare clic su **Fine**.
10. Aggiungere un nuovo campo all'interfaccia che specifichi la classe di implementazione predefinita per il comando effettuando le seguenti operazioni:
    a. Fare clic con il pulsante destro del mousesull'interfaccia **GetNewProductContractUnitPriceCmd** e selezionare **Aggiungi > Campo**.
       Si apre la SmartGuide per la creazione del campo.
    b. Nel campo **Nome campo** immettere `defaultCommandClassName`.
    c. Dall'elenco a discesa **Tipo campo** selezionare **Stringa**.

d. Nel campo **Valore iniziale**, immettere

```
"com.ibm.commerce.sample.commands.
   GetNewContractUnitPriceCmdImpl"
```

e fare clic su **Fine**.

Note:

1) Includere le doppie virgolette quando si immette questo valore.
2) Se viene visualizzato un messaggio di avvertenza indicante che un campo nella superclass verrà nascosta dalla creazione di questo nuovo campo, fare clic su **Sì** per procedere.

11. Aggiungere un nuovo campo all'interfaccia che specifichi il nome del comando effettuando le seguenti operazioni:
    a. Fare clic con il pulsante destro del mousesull'interfaccia **GetNewProductContractUnitPriceCmd** e selezionare **Aggiungi > Campo**.
    Si apre la SmartGuide per la creazione del campo.
    b. Nel campo **Nome campo** immettere `NAME`.
    c. Dall'elenco a discesa **Tipo campo** selezionare **Stringa**.
    d. Nel campo **Valore iniziale**, immettere

```
"com.ibm.commerce.sample.commands.
   GetNewProductContractUnitPriceCmd"
```

e fare clic su **Fine**.

Creazione della classe di implementazione `GetNewContractUnitPriceCmdImpl`

È necessario creare la classe di implementazione per il nuovo comandi di attività. Tale classe deve implementare la nuova interfaccia creata e deve contenere la logica aziendale del comando di attività.

Per creare la classe di implementazione `GetNewContractUnitPriceCmdImpl`, effettuare le seguenti operazioni:

1. Nello spazio di lavoro, con la scheda Progetti selezionata, espandere il progetto **_WCSamples**.
2. Fare clic con il pulsante destro del mouse sul pacchetto **com.ibm.commerce.sample.commands** e selezionare **Aggiungi > Classe**. Si apre la SmartGuide per l'aggiunta della classe.
3. Assicurarsi che l'opzione **Crea nuova classe** sia selezionata e creare la classe nel modo seguente:
    a. Nel campo **Nome classe**, immettere `GetNewContractUnitPriceCmdImpl`.
    b. Per specificare la superclasse, fare clic su **Sfoglia**, quindi nel campo **Modello** immettere `com.ibm.commerce.price.commands.GetContractUnitPriceCmdImpl` e fare clic su **OK**.

c. Fare clic su **Avanti**.
d. Per specificare i pacchetti da importare, fare clic su **Aggiungi pacchetto**. Nel campo **Modello**, immettere i seguenti pacchetti:
   - `com.ibm.commerce.command` e fare clic su **Aggiungi**
   - `com.ibm.commerce.exception` e fare clic su **Aggiungi**
   - `com.ibm.commerce.price.commands` e fare clic su **Aggiungi**
   - `com.ibm.commerce.price.utils` e fare clic su **Aggiungi**
   - `com.ibm.commerce.ras` e fare clic su **Aggiungi**
   - `com.ibm.commerce.server` e fare clic su **Aggiungi**
   - `java.math` e fare clic su **Aggiungi**, quindi su **Chiudi**.
e. Per specificare le interfacce che la classe deve implementare, fare clic su **Aggiungi**. Nel campo Modello immettere le seguenti interfacce:
   - `GetContractSpecialPriceCmd` e fare clic su **Aggiungi**.
   - `GetContractUnitPriceCmd` e fare clic su **Aggiungi**.
   - `GetProductContractUnitPriceCmd` e fare clic su **Aggiungi**.
   - `GetNewProductContractUnitPriceCmd` e fare clic su **Aggiungi**, quindi su **Chiudi**.
f. Fare clic su **Fine**.

4. Fare clic con il pulsante destro del mouse sulla classe **GetNewContractUnitPriceCmdImpl** e selezionare **Aggiungi > Campo**. Si apre la SmartGuide per la creazione del campo. Immettere le informazioni nel seguente modo:
   a. Nel campo **Nome campo** immettere `bonusPrice`.
   b. Per selezionare il tipo di campo, fare clic su **Sfoglia**. Nel campo Modello, immettere `MonetaryAmount` e fare clic su **OK**.
   c. Fare clic su **Fine**.
5. Aggiungere un nuovo metodo `performExecute()` alla classe. Le attività di questo metodo sono le seguenti:
   - richiama il metodo `performExecute()` della superclasse (GetContractUnitPriceCmdImpl)
   - imposta i valori `thisClass` e `methodName` che il meccanismo di gestione delle eccezioni deve utilizzare
   - crea istanze di un StoreAccessBean
   - crea istanze di un UserAccessBean
   - individua i prezzi originali dei prodotti
   - calcola lo sconto massimo applicabile
   - calcola il nuovo prezzo bonus (tale prezzo è del tipo *doppio*)
   - individua il tipo di valuta per il negozio

- arrotonda il prezzo del bonus e lo memorizza come quantità monetaria nella valuta corretta

Per aggiungere questo metodo, fare clic con il pulsante destro del mouse sulla classe **GetNewContractUnitPriceCmdImpl** e selezionare **Aggiungi > Metodo**. Si apre la SmartGuide per la creazione del metodo. Immettere le informazioni nel seguente modo:

a. Assicurarsi che l'opzione per la **creazione di un nuovo metodo** sia selezionata e fare clic su **Avanti**.
b. Nel campo **Nome metodo** immettere `performExecute`.
c. Dall'elenco a discesa **Tipo ritorno** selezionare **nullo** e fare clic su **Avanti**.
d. Per selezionare l'eccezione inviata dal metodo, fare clic su **Aggiungi**. Nel campo **Modello** immettere `ECException` e fare clic su **Aggiungi**, quindi su **Chiudi**.
e. Fare clic su **Fine**.
f. Dopo questa riga del codice di origine:

```
public void performExecute()
   throws com.ibm.commerce.exception.ECException
{
```

aggiungere il seguente codice:

È possibile tagliare e incollare questo codice dalla versione PDF della Guida per il programmatore. Si consiglia inizialmente di copiare il codice nella finestra Scrapbook in VisualAge per Java (per ulteriori informazioni, fare riferimento alla guida in linea VisualAge per Java) e controllare il codice per assicurarsi che non siano stati persi caratteri durante l'operazione di taglia e incollate. Quindi, una volta controllato il codice, copiarlo nell'ubicazione di destinazione. La copia del testo in un altro editor può comportare la modifica di alcuni caratteri.

```
 super.performExecute();

// Get and set this class name and method
// for use when exceptions occur.
final String thisClass =
   GetContractUnitPriceCmdImpl.class.getName();
final String methodName = "performExecute";

//get the store access bean
Integer storeId = getStoreId();
com.ibm.commerce.common.objects.StoreAccessBean storeAB =
     getCommandContext().getStore(storeId);

//get the user access bean
com.ibm.commerce.user.objects.UserAccessBean bonusAB =
     new com.ibm.commerce.user.objects.UserAccessBean();
```

```
// get the calculated price from the GetContractUnitPriceCmdImpl
MonetaryAmount priceOrg = super.getPrice();
double dblPriceOrg = priceOrg.getValue().doubleValue();

//calculate the maximum bonus that can apply to this product
double dblBonusPrice; // = dblPriceOrg;
double dblMaxBonusPoint = 0;

try {
bonusAB.setInitKey_MemberId(super.getUserId().toString());
bonusAB.refreshCopyHelper();
double dblMaxDed = dblPriceOrg * 0.2;
dblMaxBonusPoint =
     (new java.math.BigDecimal(
        bonusAB.getBonusPoint()).doubleValue());
if (dblMaxBonusPoint > dblMaxDed) dblMaxBonusPoint = dblMaxDed;
} catch (javax.ejb.CreateException ex) {
throw new ECSystemException(
     ECMessage._ERR_CREATE_EXCEPTION, thisClass, methodName, ex);
} catch (javax.ejb.FinderException ex) {

} catch (javax.naming.NamingException ex) {
throw new ECSystemException(
     ECMessage._ERR_GENERIC, thisClass, methodName, ex);
} catch (java.rmi.RemoteException ex) {
throw new ECSystemException(
     ECMessage._ERR_REMOTE_EXCEPTION, thisClass, methodName, ex);
}


//apply the maximum applicable bonus to this product price
dblBonusPrice = dblPriceOrg - dblMaxBonusPoint ;

//get the currency of this store
CommandContext context = getCommandContext();
String requestedCurrency = Helper.getCurrency( context, storeAB );

//round off and return the bonus price in MonetoryAmount type
bonusPrice = new MonetaryAmount(
     new BigDecimal(dblBonusPrice), requestedCurrency);
CurrencyManager.getInstance().roundCustomized(bonusPrice, storeAB);
```

Salvare il proprio lavoro.

6. Selezionare il metodo **getBonusPrice()** nella classe **GetNewContractUnitPriceCmdImpl** per visualizzarne il codice di origine. Nel codice di origine, modificare

```
return null;
```

in

```
return bonusPrice;
```

Salvare il proprio lavoro.

Creazione del bean di dati NewProductDataBean

Il metodo getCalculatedBonusPrice() deve essere aggiunto al ProdctDataBeanWebSphere Commerce esistente. Non è realmente necessario modificare il codice per il ProductDataBean, pertanto, è necessario creare un nuovo bean di dati che estende il ProductDataBean e, quindi, aggiungere il metodo al novo bean di dati.

Per creare il nuovo bean di dati, eseguire le seguenti operazioni:

1. Nello spazio di lavoro, con la scheda Progetti selezionata, espandere il progetto **_WCSamples**.
2. Fare clic con il pulsante destro del mouse sul pacchetto **com.ibm.commerce.sample.databeans** e selezionare **Aggiungi > Classe**. Si apre la SmartGuide per l'aggiunta della classe. Immettere le informazioni nel seguente modo:
   a. Nel campo **Nome classe** immettere NewProductDataBean.
   b. Per specificare la superclasse, fare clic su **Sfoglia**, quindi nel pannello **Modello** immettere com.ibm.commerce.catalog.beans.ProductDataBean. Fare clic su **OK** e, quindi, su **Avanti**.
   c. Aggiungere alla classe le istruzioni di importazione appropriate, facendo clic su **Aggiungi pacchetti**, quindi eseguire le seguenti operazioni:
      - Immettere com.ibm.commerce.beans e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.catalog.objects e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.command e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.datatype e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.exception e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.price.beans e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.ras e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.sample.commands e fare clic su **Aggiungi**.
      - Immettere com.ibm.commerce.server e fare clic su **Aggiungi**.
      - Immettere java.util e fare clic su **Aggiungi**, quindi su **Chiudi**.
   d. Fare clic su **Fine**.
3. Fare clic con il pulsante destro del mouse sulla classe **NewProductDataBean** e selezionare **Aggiungi > Metodo**. Si apre la SmartGuide per l'aggiunta del metodo.
4. Assicurarsi che l'opzione **Crea nuovo metodo** sia selezionata e fare clic su **Avanti**.

5. Nel campo **Nome metodo** immettere `getCalculatedBonusPrice`.
6. Per selezionare il tipo di risultato, fare clic su **Sfoglia**. Nel campo **Modello** immettere `PriceDataBean`, fare clic su **OK**, quindi su **Avanti**.
7. Per selezionare l'eccezione inviata dal metodo, fare clic su **Aggiungi**. Nel campo **Modello** immettere `ECSystemException` e fare clic su **Aggiungi**, quindi su **Chiudi**.
8. Fare clic su **Fine**.
9. Nel codice di origine, sostituire `return null;` con quanto segue:

```
PriceDataBean ibnPrice = null;
try {
     GetNewProductContractUnitPriceCmd comm =
        (GetNewProductContractUnitPriceCmd) CommandFactory.createCommand
        (GetNewProductContractUnitPriceCmd.NAME,
             getCommandContext().getStoreId());

     ECTrace.trace(ECTraceIdentifiers.COMPONENT_CATALOG,
     this.getClass().getName(), "getCalculatedBonusPrice",
     "Getting Price for CatalogEntry: " + getProductID());

     comm.setCatEntryId(new Long(getProductID()));
     comm.setCommandContext(getCommandContext());
     comm.execute();
     ibnPrice = new PriceDataBean(comm.getBonusPrice(),
          getCommandContext().getStore(),
          getCommandContext().getLanguageId());

} catch (Exception e) {
     throw new ECSystemException(ECMessage._ERR_RETRIEVE_PRICE,
          this.getClass().getName(), "getCalculatedBonusPrice",e);
}

return ibnPrice;
```

Salvare il proprio lavoro.

Aggiunta del nuovo prezzo bonus alla maschera di visualizzazione del prodotto

Il passo successivo è aggiungere il nuovo prezzo bonus alla maschera di visualizzazione del prodotto, in modo che gli acquirenti possano visualizzare il prezzo personalizzato. Una volta aggiornata la maschera di visualizzazione, viene visualizzato il nuovo prezzo scontato.

Il negozio di esempio utilizza la maschera `ProductDisplay.jsp` per la visualizzazione dei prodotti. Pertanto, è necessario aggiornare questa maschera con le informazioni per visualizzare il nuovo prezzo.

Per aggiornare la maschera di visualizzazione, eseguire le seguenti operazioni:

1. Passare alla seguente directory:
   unità_vaj:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*nome_negozio*
2. Effettuare una copia del file ProductDisplay.jsp e denominarla ProductDisplay.jsp.bak.
3. Aprire ProductDisplay.jsp in un editor di testo.
4. Dopo <%@ page import="com.ibm.commerce.common.beans.*" %>, aggiungere le seguenti istruzioni di importazione:
   ```
   <%@ page import="com.ibm.commerce.sample.commands.*" %>
   <%@ page import="com.ibm.commerce.sample.databeans.*" %>
   ```
5. Sostituire tutte le ricorrenze di ProductDataBean con NewProductDataBean.
6. Individuare la seguente riga:
   ```
   <font class="price"><%=product.getCalculatedContractPrice()%></font>
      <br><br>
   ```

   Dopo questa riga, inserire la seguente riga per richiamare e visualizzare il prezzo bonus del prodotto
   ```
   <font class="price"><%=product.getCalculatedBonusPrice()%>
         Bonus Price </font>
   <br><br>
   ```
7. Salvare questo file.

Nota: Se le sezioni vengono tagliate e incollate nella maschera di visualizzazione dalla versione PDF del manuale *WebSphere Commerce - Guida per il programmatore*, assicurarsi che i caratteri non subiscano alcuna variazione.

Verifica dell'estensione del bean enterprise

In questa sezione, è possibile verificare l'estensione realizzata per il bean enterprise, visualizzando un prodotto nel negozio di esempio InFashion. Viene visualizzato il nuovo prezzo bonus.

Si osservi che, per semplificare questo esempio, il prezzo bonus è visualizzabile a tutti gli acquirenti (registrati o ospiti). Per gli acquirenti che non hanno punti bonus, il prezzo bonus è uguale a quello normale.

Per verificare l'estensione del bean enterprise e visualizzare il prezzo bonus, eseguire le seguenti operazioni:

1. Verificare che il progetto _WCSamples sia incluso nel percorso per il motore servlet, effettuando le seguenti operazioni:
   a. Dal menu **Spazio di lavoro** in VisualAge per Java, selezionare **Strumenti > WebSphere Test Environment**.
      Si apre WebSphere Test Environment Centro di controllo.
   b. Fare clic su **Motore servlet**.

c. Se Motore servlet è in esecuzione, fare clic su **Interrompi Motore servlet** e, quindi, su **Modifica del class path**.

d. Se **_WCSamples** non è stato ancora selezionato, selezionarlo e fare clic su **OK**.

2. Avviare l'WebSphere Test Environment come indicato in Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347. Il server dei nomi permanenti e il server EJB potrebbero già essere in esecuzione, in questo caso, è necessario avviare solo il motore servlet.

3. Aprire un browser ed immettere il seguente URL

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
    storeId=Id_negozio&catalogId=Id_catalogo&langId=-1
```

4. Fare clic sul collegamento **Registra** sotto l'intestazione Servizi e, quindi, fare clic su **Registra** sotto l'intestazione Nuovo cliente.
Registrare un nuovo cliente utilizzando l'indirizzo e- `wctester@wc` e la password `wctester1`. Completare gli altri campi con i valori di test e fare clic su **Inoltra**. Lasciare il browser aperto.

5. DB2 Aprire il Centro di comandi DB2 ed effettuare le seguenti operazioni:

   a. Fare clic sulla scheda **Interactive**

   b. Nel campo **Command** eseguire le seguenti operazioni:

      1) Immettere

```
connect to nome_database
```

      dove *nome_database_utente* è il nome del database di WebSphere Commerce, quindi fare clic sull'icona Esegui.

      2) Immettere `select users_id from userreg where logonid = 'wctester@wc'` e fare clic sull'icona di esecuzione.

   c. La scheda Risultati interrogazione visualizza la voce per il cliente registrato nel passo 4. Annotare qui il valore USERS_ID del cliente:

      _____

   d. Aggiornare il bilancio di punti bonus del cliente appena registrato. Fare clic sulla scheda Interactive e nel campo **Command** immettere quanto segue:

```
update BONUS set BONUSPOINT = 1000 where MEMBERID = id_utenti
```

      dove *id_utenti* è il valore del passo 5c. Fare clic sull'icona di esecuzione.

6. Oracle Aggiornare il bilancio dei punti bonus dell'utente effettuando le seguenti operazioni:

   a. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).

   b. Nel campo **Nome utente** immettere il nome utente Oracle.

c. Nel campo **Password**, immettere la password per Oracle.
d. Nel campo **Stringa host** immettere la propria stringa di collegamento.
e. Immettere `select users_id from userreg where logonid = 'wctester@wc';`
f. Viene visualizzata la voce per il cliente registrato nel passaggio 4. Annotare qui il valore USERS_ID del cliente: _____
g. Aggiornare il bilancio di punti bonus del cliente appena registrato immettendo quanto riportato di seguito:

   ```
   update BONUS set BONUSPOINT = 1000 dove MEMBERID = id_utenti;
   ```

   dove *id_utenti* è il valore del passo 6f.
h. Immettere quanto segue per confermare le modifiche al database:

   ```
   commit;
   ```

   e premere Invia per eseguire l'istruzione SQL.

7. Nel browser, fare clic sul collegamento **Moda uomo** per visualizzare la sezione abbigliamento maschile del negozio.
8. Fare clic sul collegamento per l'offerta speciale per visualizzare la pagina del prodotto. Questa pagina visualizza il prezzo normale ed il prezzo scontato in base al bilancio dei punti bonus del cliente.

   Nota: Se una traccia stack viene visualizzata al posto del prezzo bonus, è necessario disabilitare la memorizzazione cache. Questa operazione può essere effettuata impostando il valore del componente CacheDaemon su false nel file *nome_istanza*.xml, come mostrato di seguito:

   ```
   <component compClassName=
         "com.ibm.commerce.cache.daemon.CacheDaemonComponent"
                  enable="false"
                  name="CacheDaemon" />
   ```

   Dopo aver modificato il valore nel file *nome_istanza*.xml, è necessario arrestare e riavviare il motore servlet in WebSphere Test Environment.

(Facoltativo) Distribuzione di una logica aziendale personalizzata su un WebSphere Commerce Server remoto

In questa sezione viene descritta la modalità di distribuzione del bean entità modificato e del nuovo comando attività a un negozio in esecuzione all'esterno di WebSphere Test Environment.

La distribuzione comprende la creazione di file JAR per i bean enterprise public WebSphere Commerce e la logica di bean di dati e comandi, il posizionamento di file JAR nelle directory appropriate sul server di

destinazione, l'arresto dell'istanza WebSphere Commerce, la modifica dei classpath, la distribuzione dei bean enterprise beans utilizzando il programma di utilità XMLConfig e il riavvio dell'istanza.

Creazione del file JAR per il nuovo comando prezzo

È necessario creare un file JAR per il progetto **_WCSamples** in modo che il nuovo comando di attività venga distribuito. Per creare questo file JAR, effettuare le seguenti operazioni sulla macchina di sviluppo:

1. Arrestare WebSphere Test Environment, come descritto in Appendice A, "Avvio e interruzione di WebSphere Test Environment" a pagina 347.
2. Con la scheda Progetti selezionata, selezionare il progetto **_WCSamples**.
3. Con il progetto evidenziato, fare clic con il pulsante destro del mouse e selezionare **Esporta**.
   Si apre la SmartGuide di esportazione.
4. Selezionare **file Jar** e fare clic su **Avanti.**
5. Nel campo **Jar file** immettere quanto segue:
   unità`:\WebSphere\CommerceServerDev\mytemp_c\wcssamplesc_1.jar`
   dove *unità* è l'unità sulla quale è stato installato WebSphere Commerce.
6. Selezionare gli attributi nel seguente modo:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
bean	Selezionato
Includi attributi di debug nei file .class	Selezionato

   Accettare i valori predefiniti per gli altri attributi.
7. Fare clic su **Fine**.

Poiché il file JAR creato non contiene le informazioni complete sulla denominazione dei pacchetti è necessario utilizzare un altro programma di utilità per i pacchetti (esterni a VisualAge per Java) per creare nuovi pacchetti del file JAR. Per creare un nuovo pacchetto di questo file, eseguire le procedure descritte:

1. Da una finestra comandi visualizzare la seguente directory:*unità*`:\WebSphere\CommerceServerDev\mytemp_c`
2. Immettere `mkdir temp3`.
3. Immettere `cd temp3`.

4. Impostare il percorso come specificato di seguito:
   `set PATH=%PATH%;`*`unità`*`:\WebSphere\WebSphereStudio4\bin;`
   dove *unità* è l'unità sulla quale è stato installato WebSphere Studio.
5. Immettere `jar xvf ../wcssamplesc_1.jar`.
6. Immettere `jar cvf ../wcssamplesc.jar *` (si noti che la parte _1 è stata rimossa dal nome).

Creazione di un file JAR per il gruppo EJB WCSUser

È necessario creare un file JAR EJB 1.1 di esportazione per il gruppo EJB contenente il bean enterprise modificato. È necessario selezionare il seguente gruppo durante la creazione del file JAR:

- WCSUser

Per creare il file JAR per il gruppo EJB WCSUser, effettuare le seguenti operazioni:

1. Con la scheda EJB selezionata, evidenziare il gruppo EJB **WCSUser**.
2. Fare clic con il pulsante destro del mouse sul gruppo EJB **WCSUser** e selezionare **Esporta > EJB 1.1 JAR**.
   Viene visualizzata la SmartGuide per l'esportazione su un file JAR EJB 1.1.
3. Nel campo **file JAR**,
   immettere`unità`*`:\WebSphere\CommerceServerDev\mytemp_c\`*
   `CustomizedWCSUserDeployed_DT.jar`
4. Selezionare gli attributi nel seguente modo:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
Database di destinazione	DB2 Se si sta distribuendo su un database DB2, selezionare **DB2 per NT, V7.1**. Oracle Se si sta distribuendo su un database Oracle, selezionare **Oracle, V8**.
Includi attributi di debug nei file .class	Selezionato

   Accettare i valori predefiniti per gli altri attributi.
5. Fare clic su **Fine**.

Il file JAR è stato creato.

Il file JAR è stato denominato con il suffisso "_DT" per indicare che è necessario eseguire questo file JAR con lo strumento EJB Deploy fornito con WebSphere Application Server prima di distribuire il file nell'applicazione WebSphere Commerce.

Creazione del file wcsejsclient.jar

Per creare il file client JAR, effettuare le seguenti operazioni:

1. Con la scheda EJB selezionata, evidenziare tutti i gruppi EJB di WebSphere Commerce (il nome comincia per WCS). Con i gruppi evidenziati, fare clic con il pulsante destro del mouse e selezionare **Esporta > Client JAR**. Si apre il manuale Export SmartGuide.
2. Nel campo **File JAR**, immettere *unità*:\WebSphere\CommerceServerDev\mytemp_c\wcsejsclient.jar
3. Selezionare gli attributi nel seguente modo:

Attributo	Valore
bean	Selezionato
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
Includi attributi di debug nei file .class	Selezionato

   Accettare i valori predefiniti per gli altri attributi.
4. Fare clic su **Fine**.

Il file JAR è stato creato.

Copia della maschera JSP aggiornata nella directory del negozio di destinazione

In "Aggiunta del nuovo prezzo bonus alla maschera di visualizzazione del prodotto" a pagina 325, è stata aggiornata la maschera di visualizzazione in modo da presentare il prezzo appena creato. In questo passo viene copiata la maschera JSP aggiornata dalla struttura di directory WebSphere Test Environment nella directory utilizzata dal negozio quando è in esecuzione all'esterno di WebSphere Test Environment.

1. Sulla macchina di sviluppo spostarsi alla seguente directory: *unità_vaj*:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web*directory_negozio* dove *unità_vaj* è l'unità sulla quale è stato installato VisualAge per Java e *directory_negozio* è il nome della directory per il negozio di esempio. Copiare il file ProductDisplay.jsp.

2. Incollare il file ProductDisplay.jsp nella seguente directory:

   ```
   unità:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_nome_istanza.ear\
      wcstores.war\directory_negozio
   ```

   dove *unità* è l'unità in cui WebSphere Commerce è installato, *directory_negozio* è il nome della directory per il negozio e *nome_istanza* è il nome dell'istanza WebSphere Commerce.

Copia dei file JAR sul WebSphere Commerce Server di destinazione

È necessario copiare i file JAR dalla macchina di sviluppo alla appropriata directory sul WebSphere Commerce Server di destinazione. Per copiare questi file procedere nel modo seguente:

1. Sulla macchina di sviluppo, passare alla directory:
   unità:\WebSphere\CommerceServerDev\mytemp_c
   e individuare i seguenti file:
   - wcssamplesc.jar
   - CustomizedWCSUserDeployed_DT.jar
   - wcsejsclient.jar

   dove *unità* è l'unità sulla quale è stato installato WebSphere Commerce Studio, Business Developer Edition.

   Ciascuno dei file precedenti deve essere copiato in una determinata directory del WebSphere Commerce Server di destinazione. Leggere i passi di seguito riportati per verificare che ciascun file sia memorizzato nella corretta ubicazione.
2. Copiare il file wcssamplesc.jar nella seguente directory del WebSphere Commerce Server di destinazione:

   ```
   unità:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_nome_istanza.ear\
      wcstores.war\WEB-INF\lib
   ```

   dove *unità* è l'unità in cui è installato WebSphere Commerce Business Edition e *nome_istanza* è il nome dell'istanza (ad esempio, demo).
3. Copiare il file wcsejsclient.jar nella seguente directory del WebSphere Commerce Server di destinazione:

   ```
   unità:\WebSphere\CommerceServer\temp\lib
   ```
4. Copiare il file CustomizedWCSUserDeployed_DT.jar nella seguente directory del WebSphere Commerce Server di destinazione:

   ```
   unità:\WebSphere\CommerceServer\temp
   ```

Esecuzione dello strumento per la distribuzione di EJB

È necessario eseguire lo strumento di distribuzione di EJB sul file JAR contenente il nuovo gruppo EJB. Questo strumento è fornito con WebSphere Application Server.

Per eseguire questo strumento, eseguire le seguenti operazioni:

1. Alla richiesta comandi, passare alla seguente directory:

   unità:\WebSphere\CommerceServer\temp

2. Aggiungere temporaneamente lo strumento al percorso di sistema immettendo il seguente comando:

   PATH=*unità*:\WebSphere\AppServer\deploytool;%PATH%

3. Immettere il comando ejbdeploy come segue:

   ejbdeploy *FileJARGruppoEJB DirLavoro FileJAREmissione* -nowarn -keep -35 -cp *ClasspathFileJARDip*

   dove:

   - *FileJARGruppoEJB* è il nome del file JAR per il gruppo EJB. In questo caso, CustomizedWCSUserDeployed_DT.jar.
   - *DirLavoro* è la directory utilizzata.
   - *FileJAREmissione* è il nome del file JAR di emissione. In questo caso, immettere CustomizedWCSUserDeployed.jar.
   - -nowarn è un parametro facoltativo che elimina i messaggi informativi e di avvertenza.
   - -keep è un parametro facoltativo per mantenere la directory in uso dopo aver eseguito il comando ejbdeploy.
   - -35 è un parametro obbligatorio che utilizzerà le stesse regole di corrispondenza dall'alto verso il basso per i bean entità utilizzate nello strumento di distribuzione EJB fornito con WebSphere Application Server, Versione 3.5.
   - -cp *ClassPathFileJARDip* è il class path dei file JAR dipendenti. Quando si modifica un enterprise di WebSphere Commerce esistente, è necessario includere i file wcsejsclient.jar, wcsejbimpl.jar e xml4j.jar nel class path dei file JAR dipendenti. effettuando le seguenti operazioni:

     "*unità*:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
     unità:\WebSphere\AppServer\InstalledApps\
     WC_Enterprise_App_*Nomeistanza*.ear\lib\wcsejbimpl.jar;
     unità:\WebSphere\AppServer\InstalledApps\
     WC_Enterprise_App_*Nomeistanza*.ear\lib\xml4j.jar;"

Modifica del livello di isolamento transazione per i bean entità

Questa procedura prevede l'utilizzo del comando modifyIsolationLevel per modificare il livello di isolamento transazione dei bean entità al livello richiesto per il proprio tipo di database specifico.

Per eseguire il comando modifyIsolationLevel, effettuare le seguenti operazioni:

1. Dal WebSphere Commerce Server di destinazione, utilizzare una richiesta comandi per passare alla seguente directory:

   unità:\WebSphere\CommerceServer\bin

2. È necessario utilizzare il comando modifyIsolationLevel la cui sintassi è riportata di seguito:

   ```
   modifyIsolationLevel -jarFile nome_file_jar.jar
      -logFile nome_file_log -dbType tipo_db
   ```

   dove

   - *nome_file_jar.jar* è il nome del file JAR che contiene il codice personalizzato
   - *nome_file_log* è il nome completo del file in cui registrare le informazioni
   - *tipo_db* è il tipo di database in uso. Immettere DB2 oppure ORACLE

   Di seguito viene riportato un esempio del comando modifyIsolationLevel con tutti i valori specificati:

   ```
   modifyIsolationLevel -jarFile
      D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
      -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
      -dbType DB2
   ```

Nota: I nomi dei parametri operano una distinzione tra maiuscolo e minuscolo, ossia jar**F**ile non è lo stesso di jar**f**ile. Assicurarsi di immettere correttamente i nomi dei parametri.

Se la finestra comandi non visualizza eccezioni, il comando è stato eseguito correttamente. Al termine dell'esecuzione, è possibile notare che la data e l'orario del file JAR sono cambiati.

Aggiornamento del database di destinazione

Se si esegue una distribuzione in un WebSphere Commerce Server di destinazione che utilizza un database differente da quello utilizzato dal WebSphere Test Environment, è necessario aggiornare il database di destinazione nel modo seguente:

- Se Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale" a pagina 207 è stata completata, è necessario aggiornare la tabella per aggiungere una riga per ogni utente nella tabella USERS.
- Se non viene completato Capitolo 9, "Supporto didattico: creazione di nuova logica aziendale" a pagina 207, è necessario creare e riempire la tabella.

Se non viene completato Capitolo 9, “Supporto didattico: creazione di nuova logica aziendale” a pagina 207, è necessario creare e riempire la tabella. Vengono fornite le istruzioni per ognuno di questi scenari.

DB2 Se si utilizza un database DB2 e occorre *aggiornare* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire il Centro di comandi DB2 (**Start > Programmi > IBM DB2 > Centro di comandi**) e fare clic sulla scheda **Script**.
2. Nel campo **Script**, immettere

   ```
   connect to nome_database
   ```

   dove *nome_database* è il nome del database e fare clic sull'icona Esegui.
3. Nel campo **Script**, immettere quanto segue, quindi fare clic sull'icona di esecuzione:

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
   ```

   La tabella BONUS è quindi stata aggiornata.

DB2 Se si utilizza un database DB2 e occorre *creare e popolare* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire il Centro di comandi DB2 (**Start > Programmi > IBM DB2 > Centro di comandi**) e fare clic sulla scheda **Script**.
2. Nel campo **Script**, immettere

   ```
   connect to nome_database
   ```

   dove *nome_database* è il nome del database e fare clic sull'icona Esegui.
3. Nel campo **Script**, immettere quanto segue, quindi fare clic sull'icona di esecuzione:

   ```
   CREATE TABLE BONUS (MEMBERID BIGINT NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (id_utente) on delete cascade)
   ```

   La tabella BONUS è stata creata.
4. Per riempire questa tabella, immettere quanto segue nel campo **Script**, quindi fare clic sull'icona di esecuzione:

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```
5. Chiudere il Centro di comandi DB2.

Oracle Se si utilizza un database Oracle ed è necessario *aggiornare* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password**, immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.
5. Aggiornare la tabella BONUS, immettendo le seguenti informazioni nella finestra SQL Plus:

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
   ```

   Fare clic su Invia per eseguire l'istruzione SQL.
   La tabella BONUS è ora aggiornata.
6. Immettere quanto segue per confermare le modifiche al database:

   ```
   commit;
   ```

   e premere Invia per eseguire l'istruzione SQL.

Oracle Se si utilizza un database Oracle ed è necessario *creare e riempire* la tabella BONUS, effettuare le seguenti operazioni:

1. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
2. Nel campo **Nome utente** immettere il nome utente Oracle.
3. Nel campo **Password**, immettere la password per Oracle.
4. Nel campo **Stringa host** immettere la propria stringa di collegamento.
5. Creare la tabella BONUS, immettendo le seguenti informazioni nella finestra SQL Plus:

   ```
   CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (users_id) on delete cascade);
   ```

   Fare clic su Invia per eseguire l'istruzione SQL.
   La tabella BONUS è stata creata.
6. Riempire la tabella BONUS immettendo quanto segue:

   ```
   insert into BONUS (select USERS_ID, 0 from USERS);
   ```
7. Immettere quanto segue per confermare le modifiche al database:

   ```
   commit;
   ```

e premere Invia per eseguire l'istruzione SQL.

Esportazione dell'enterprise application corrente da WebSphere Application Server

In questo passo, l'enterprise application corrente viene esportata da WebSphere Application Server in modo che sia possibile aprirla successivamente con lo Strumento di assemblaggio delle applicazioni.

Per esportare l'enterprise application corrente da WebSphere Application Server, effettuare le seguenti operazioni:

1. Aprire la WebSphere Application Server Console di gestione.
2. Espandere **Dominio di gestione WebSphere**.
3. Espandere **Enterprise Application**.
4. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, espandere l'applicazione **demo** e selezionare **Esporta applicazione**.
5. Nel campo **Esporta directory**, immettere *unità*:\WebSphere\CommerceServer\working.
   Viene esportata l'intera applicazione, incluse tutte le risorse presenti nel file WC_Enterprise_App_*Nomeistanza*.ear (dove *Nomeistanza* è il nome dell'istanza di WebSphere Commerce).

   Nota: Se già esiste un file WC_Enterprise_App_*Nomeistanza*.ear, è possibile ridenominare il vecchio file oppure sovrascriverlo.

Esportazione delle informazioni di configurazione XML per l'enterprise application

È inoltre necessario esportare le informazioni di configurazione XML per l'enterprise application. Per esportare queste informazioni, utilizzare l'utilità della riga comandi XMLConfig fornita da WebSphere Application Server.

Per esportare queste informazioni di configurazione, effettuare le seguenti operazioni:

1. Alla richiesta comandi, passare alla seguente directory:
   unità:\WebSphere\CommerceServer\working
2. Richiamare lo strumento XMLConfig per eseguire un'esportazione parziale immettendo il seguente comando:

```
xmlConfig -export OutputFileB.xml -partial was.export.app.xml
   -adminNodeName NomeHostwas
```

   dove *NomeHostwas* è il nome del nodo in WebSphere Application Server contenente l'enterprise application corrente. Inoltre, OutputFileB.xml è il nome del file creato in seguito all'esecuzione di questo comando.

Dopo l'esportazione delle informazioni relative ai bean enterprise presenti nell'enterprise application corrente, è necessario aggiornare il fileOutputFileB.xml in modo che punti al file JAR contenente il codice per il bean User modificato.

Per aggiornare il file OutputFileB.xml, eseguire le procedure descritte:

1. Passare alla seguente directory:

   unità:\WebSphere\CommerceServer\working

2. Aprire il file OutputFileB.xml con un editor di testo.
3. Individuare la tag <ear-file-name> e sostituire il valore con quello di seguito riportato:

   unità:\WebSphere\CommerceServer\working\
   WC_Enterprise_App_*Nomeistanza*.ear

4. Individuare la stanza <ejb-module> per il gruppo EJB di WCSUser e modificare il valore contenuto nelle tag <jar-file> in CustomizedWCSUserDeployed.jar
5. Salvare il file OutputFileB.xml.

Assemblaggio del gruppo EJB modificato nell'enterprise application

In questa fase, l'enterprise application viene aperta con lo strumento di assemblaggio delle applicazioni. Una volta aperta con questo strumento, è possibile effettuare le operazioni di seguito riportate per includere il bean Utente modificato nell'enterprise application:

1. Effettuare una copia del classpath per la versione esistente del gruppo EJB WCSUser.
2. Rimuovere la versione esistente del gruppo EJB WCSUser.
3. Importare la nuova versione del gruppo EJB WCSUser. Il file JAR per il nuovo gruppo EJB è memorizzato nella sezione del modulo EJB dell'enterprise application.
4. Impostare il classpath per il gruppo EJB WCSUser.

Per assemblare il nuovo gruppo EJB nell'enterprise application, effettuare le seguenti operazioni:

1. Ritornare all'enterprise application corrente, effettuando le seguenti operazioni:
   a. Alla richiesta comandi, passare alla seguente directory:

      unità:\WebSphere\CommerceServer\working

   b. Immettere il seguente comando:

      copy WC_Enterprise_App_*Nomeistanza*.ear
      WC_Enterprise_App_*Nomeistanza*.ear.bak2

2. Aprire la WebSphere Application Server Console di gestione.

3. Dal menu **Strumenti**, selezionare **Strumento di assemblaggio applicazioni**. (Se si apre una finestra di benvenuti, selezionare **Annulla** per accedere alla console).
4. Aprire l'enterprise application che si desidera utilizzare attenendosi alla seguente procedura:
   a. Dal menu **File**, selezionare **Apri**.
   b. Nel campo **Nome file**, immettere:

      unità:\WebSphere\CommerceServer\working\
      WC_Enterprise_App_*Nomeistanza*.ear

      e fare clic su **Apri**. Prima di continuare con i passi successivi, attendere l'apertura dell'applicazione. L'apertura può richiedere alcuni minuti.
5. Fare clic su **Moduli EJB**. Nel pannello a destra vengono visualizzati i moduli EJB presenti nell'enterprise application.
6. Fare clic sul modulo EJB **WCSUser**.
7. Fare clic sulla scheda Generale per visualizzare le informazioni sul class path per il modulo EJB WCSUser esistente. Copiare queste informazioni in un file di testo (ad esempio, WCSUser_path.txt).
8. Fare clic con il pulsante destro del mouse sul modulo EJB **WCSUser** e selezionare **Elimina**.
9. Fare clic con il pulsante destro del mouse su **Moduli EJB** e selezionare **Importa**.
10. Nel campo **Nome file**, immettere:

    unità:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar

    e fare clic su **Apri**. Nella finestra per la conferma dei valori, fare clic su **OK**.
11. Una volta importato il file CustomizedWCSUserDeployed.jar, individuare il gruppo EJB **WCSUser** e selezionare questo gruppo.
    Le informazioni su questo gruppo sono visualizzate nel pannello a destra.
12. Aprire il file di testo contenente le informazioni sul classpath per la versione precedente del gruppo EJB WCSUser. Selezionare e copiare il classpath.
13. Nel campo del classpath per il nuovo gruppo EJB WCSUser, incollare le informazioni sul classpath.
14. Fare clic su **Applica**.
15. Dal menu **File**, selezionare **Chiudi**.
16. Attendere la chiusura del file, quindi dal menu **File**, selezionare **Apri** per aprire nuovamente il file*unità*:\WebSphere\CommerceServer\working\ WC_Enterprise_App_*Nomeistanza*.ear.

17. Configurare la sicurezza per il bean User effettuando le seguenti operazioni:
    a. Con il nodo Moduli EJB espanso, individuare ed espandere il nodo **WCSUser**.
    b. Espandere **Entity Beans**.
    c. Espandere **Utente**
    d. Fare clic su **Estensioni metodo**, quindi, nel pannello a destra, procedere come segue:
        1) Fare clic sul separatore **Avanzato**.
        2) Verificare che **Identità di sicurezza** sia selezionato.
        3) Per ciascun metodo, verificare che sia selezionato **Usa identità di server EJB**.
        4) Fare clic su **Applica** (se sono state effettuate delle modifiche).
    e. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse su **Ruoli sicurezza** nel gruppo EJB `WCSUser` e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:
        1) Nel campo **Nome**, immettere `WCSecurityRole` e fare clic su **Applica**. Se questo ruolo già esiste, non è necessario eseguire questa operazione.
    f. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse su **Autorizzazioni metodi** nel gruppo EJB `WCSUser` e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:
        1) Nel campo **Nome autorizzazione metodo**, immettere `WCMethodPermission`
        2) Nell'area di selezione **Metodi**, fare clic su **Aggiungi**. Si apre la finestra
        Aggiungi metodo.
        3) Espandere il file **CustomizedWCSUserDeployed.jar** e selezionare tutti i bean enterprise (tenere premuto il tasto Maius durante la selezione). Fare clic su **OK**. Tutti i bean enterprise vengono quindi visualizzati nella colonna bean Enterprise mentre **Tutti i metodi** vengono visualizzati nella colonna Tipi.
        4) Nell'area di selezione Ruoli, fare clic su **Aggiungi**, selezionare `WCSecurityRole` e quindi fare clic su **OK**.
        5) Fare clic su **Applica**, quindi su **OK**.
18. Dal menu **File**, selezionare **Salva**.
19. Chiudere lo strumento di assemblaggio delle applicazioni.

Una volta completato questo passo, viene creata una nuova enterprise application contenente tutta la logica precedente e la nuova logica aziendale. Questi elementi sono contenuti nel file `WC_Enterprise_App_`*`nomeIstanza`*`.ear` recentemente modificato.

Importazione della nuova enterprise application in WebSphere Application Server

Di seguito vengono riportati i passi principali per l'importazione della nuova enterprise application in WebSphere Application Server:

1. Arresto dell'enterprise application attualmente in esecuzione in WebSphere Application Server e successiva rimozione. Questi passi vengono eseguiti in WebSphere Application Server Console di gestione.
2. Importazione della nuova applicazione, mediante l'utility della riga comandi XMLConfig.
3. Aggiornamento della Console di gestione di WebSphere Application Server ed avvio della nuova enterprise application.

Queste singole operazioni verranno descritte in maniera più dettagliata nelle sezioni successive.

Arresto e rimozione dell'enterprise application corrente: Per arrestare e rimuovere l'enterprise application corrente da WebSphere Application Server, effettuare le seguenti operazioni:

1. Aprire la Console di gestione di WebSphere Application Server.
2. Espandere **WebSphere Administrative Domain**.
3. Espandere **Nodi**.
4. Espandere *Nomenodo* (dove *Nomenodo* è il nome del nodo).
5. Espandere **Server delle applicazioni**.
6. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Server -** *Nomeistanza* e selezionare **Arresta**.
7. Espandere **Enterprise Applications**.
8. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Enterprise Server - demo** e selezionare **Arresta**.
9. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Enterprise Server - demo** e selezionare **Rimuovi**.
10. Quando viene richiesto di indicare se l'applicazione deve essere esportata, selezionare **No**.

Importazione della nuova enterprise application mediante XMLConfig: Per importare la nuova enterprise application mediante l'utility della riga comandi XMLConfig, attenersi alla seguente procedura:

1. Passare alla seguente directory:

unità:\WebSphere\CommerceServer\working

2. Alla richiesta comandi, immettere il comando di seguito riportato per importare l'enterprise application in WebSphere Application Server:

   ```
   xmlConfig -import OutputFileB.xml -adminNodeName nomehost_was
   ```

   dove *nomehost_was* è il nome del nodo di WebSphere Application Server contenente l'applicazione corrente.

Nota: 400 Se si esegue la distribuzione di un'istanza di WebSphere Commerce in esecuzione su iSeries, occorre effettuare un ulteriore passo per modificare le autorizzazioni della directory dopo l'importazione dell'applicazione. Consultare "Importazione dell'enterprise application" a pagina 382 per maggiori dettagli sulla modifica di tali autorizzazioni.

Avvio della nuova enterprise application: Una volta importata la nuova enterprise application mediante l'utility della riga comandi XMLConfig, è possibile utilizzare la Console di gestione di WebSphere Application Server per eseguire un aggiornamento e quindi avviare la nuova applicazione.

Per aggiornare la console e avviare la nuova applicazione, effettuare le seguenti operazioni:

1. Aprire la Console di gestione di WebSphere Application Server.
2. Espandere **Dominio di gestione WebSphere**
3. Evidenziare **Nodi**.
4. Fare clic sull'icona per l'**aggiornamento dell'albero secondario**.
5. Avviare l'applicazione WebSphere Commerce effettuando le seguenti operazioni:
   - Espandere **Server delle applicazioni**.
   - Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Server -** *Nomeistanza* e selezionare **Avvia**.

Verifica del nuovo codice nel negozio di destinazione

Per verificare il bean entità modificato e il nuovo comando di attività all'interno del negozio in esecuzione su WebSphere Application Server, effettuare le seguenti operazioni:

1. Aprire il browser e immettere il seguente URL:

   ```
   http://nomehost/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=Id_negoizo&catalogId=Id_catalogo&langId=-1
   ```

   dove *Id_negozio* è l'ID del negozio e *Id_catalogo* è l'ID del catalogo del negozio.

Nota: Se si riceve l'errore 500, è necessario riavviare il WebSphere Application Server per poter aggiornare il proprio enterprise application.

2. Fare clic sul collegamento **Registra** sotto l'intestazione Servizi, quindi selezionare **Registra** sotto l'intestazione Nuovo cliente.
Registrare un nuovo cliente utilizzando l'indirizzo e- `wctester@wc` e la password `wctester1`. Completare gli altri campi con i valori di test e fare clic su **Inoltra**. Lasciare il browser aperto.

3. DB2 Aprire il Centro di comandi DB2 ed effettuare le seguenti operazioni:
   a. Fare clic sulla scheda **Interactive**
   b. Nel campo **Command** eseguire le seguenti operazioni:
      1) Immettere
      ```
      connect to nome_database
      ```
      dove *nome_database_utente* è il nome del database di WebSphere Commerce, quindi fare clic sull'icona Esegui.
      2) Immettere `select users_id from USERREG where LOGONID = 'wctester@wc'` e fare clic sull'icona di esecuzione.
   c. La scheda Risultati interrogazione visualizza la voce per il cliente registrato nel passo 2. Annotare qui il valore USERS_ID del cliente:
      _____
   d. Aggiornare il bilancio di punti bonus del cliente appena registrato. Fare clic sulla scheda Interactive e nel campo **Command** immettere quanto segue:
      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = id_utenti
      ```
      dove *id_utenti* è il valore del passo 3c. Fare clic sull'icona di esecuzione.

4. Oracle Aggiornare il bilancio dei punti bonus dell'utente effettuando le seguenti operazioni:
   a. Aprire la finestra dei comandi Oracle SQL Plus (**Start > Programmi > Oracle > Application Development > SQL Plus**).
   b. Nel campo **Nome utente** immettere il nome utente Oracle.
   c. Nel campo **Password**, immettere la password per Oracle.
   d. Nel campo **Stringa host** immettere la propria stringa di collegamento.
   e. Immettere `select users_id from USERREG where LOGONID = 'wctester@wc';` per determinare l'ID dell'utente.
   f. Viene visualizzata la voce per il cliente registrato nel passaggio 2. Annotare qui il valore USERS_ID del cliente: _____
   g. Aggiornare il bilancio di punti bonus del cliente appena registrato immettendo quanto riportato di seguito:

```
update BONUS set BONUSPOINT = 1000 dove MEMBERID = id_utenti;
```

dove *id_utenti* è il valore del passo 4f.

h. Immettere quanto segue per confermare le modifiche al database:

```
commit;
```

e premere Invia per eseguire l'istruzione SQL.

5. Nel browser, fare clic sul collegamento **Moda uomo** per visualizzare la sezione abbigliamento maschile del negozio.
6. Fare clic sul collegamento per l'offerta speciale per visualizzare la pagina del prodotto. Questa pagina visualizza il prezzo normale ed il prezzo scontato in base al bilancio dei punti bonus del cliente.

Parte 5. Appendici

Appendice A. Avvio e interruzione di WebSphere Test Environment

Questa sezione descrive le fasi relative all'avvio di WebSphere Test Environment in VisualAge per Java. In genere, l'avvio di questo ambiente richiede l'esecuzione delle seguenti fasi:

1. Avvio del server dei nomi permanenti.
2. Avvio del server EJB.
3. Avvio di Servlet engine.

Ciascuna di queste fasi viene descritta nelle sezioni successive.

Per interrompere l'ambiente del test, invertire l'ordine delle fasi.

Nota: Per poter utilizzare tutte le funzioni di WebSphere Test Environment, assicurarsi che WebSphere Application Server non sia in esecuzione prima di avviare WebSphere Test Environment. Per informazioni sull'arresto di WebSphere Application Server, fare riferimento a *WebSphere Commerce Guida per l'installazione*.

Avvio e interruzione del server dei nomi permanenti

Il server dei nomi permanenti riceve richieste di ricerca dai client per i bean enterprise. Tutti i bean enterprise sono registrati con il server dei nomi permanenti.

Per avviare o interrompere il server dei nomi permanenti, effettuare le seguenti operazioni:

1. Dal menu **Spazio di lavoro** in VisualAge per Java, selezionare **Strumenti > WebSphere Test Environment**.
   Si apre WebSphere Test Environment Centro di controllo.
2. Fare clic su **Server nomi permanenti** e quindi su una delle opzioni:
   - **Avvia nome server**.
     Attendere che nella finestra della console venga visualizzato il messaggio relativo all'apertura del server. L'avvio di questo server può richiedere qualche minuto.
   - **Arresta server di nomi**.

Avvio e interruzione del server EJB

Il server EJB è un processo o applicazione di livello elevato che fornisce un ambiente runtime per il supporto dell'esecuzione dell'applicazione server che utilizza i bean enterprise.

Per avviare o interrompere il server EJB, effettuare le seguenti operazioni:

1. Aprire la finestra della configurazione del server EJB (fare clic sulla scheda EJB, quindi dal menu **EJB** selezionare **Apri in > Configurazione server**).
2. Fare clic con il pulsante destro del mouse sul server EJB che si desidera avviare o arrestare (ad esempio, **Server EJB {Server 1}**) ed eseguire una delle seguenti operazioni:
   - Selezionare **Avvia server**
     Attendere che nella finestra della console venga visualizzato il messaggio relativo all'apertura del server: è possibile che sia necessario selezionare **Server EJB** nella Console per visualizzare lo stato di questo server. L'avvio del server può impiegare 10/15 minuti a seconda del computer utilizzato.
   - Selezionare **Arresta server**

La Console è il dispositivo di immissione ed emissione standard per i programmi Java in IDE. Per visualizzare il risultato di un determinato programma, selezionare prima il programma nel pannello relativo a tutti i programmi; l'output del programma viene visualizzato nel relativo pannello.

Avvio e interruzione di Servlet engine

Servlet engine integra la funzionalità del server Web e di WebSphere Application Server per creare un ambiente per l'esecuzione del test.

Per avviare o interrompere Servlet engine, effettuare le seguenti operazioni:

1. Dal menu **Spazio di lavoro** in VisualAge per Java, selezionare **Strumenti > WebSphere Test Environment**.
   Si apre WebSphere Test Environment Centro di controllo.
2. Fare clic su **Motore servlet**, quindi su una delle seguenti opzioni:
   - **Avvia Motore servlet**.
     Quando il Servlet è in esecuzione, la console visualizza un messaggio informativo.
   - **Arresta Motore servlet**.

Appendice B. Dettagli sulla configurazione

Una volta creato il codice personalizzato in VisualAge per Java e averlo provato in WebSphere Test Environment, è necessario distribuirli su un'istanza WebSphere Commerce che è in esecuzione esternamente a WebSphere Test Environment. Questa istanza di WebSphere Commerce può essere eseguita in locale sulla macchina di sviluppo oppure su un'altra macchina (che utilizza lo stesso sistema operativo o uno diverso).

In quest'appendice sono descritte le operazioni richieste per la distribuzione del codice personalizzato in un'istanza WebSphere Commerce in esecuzione all'esterno del WebSphere Test Environment. Per una maggiore comprensione delle operazioni di livello avanzato relative al processo di distribuzione, fare riferimento alla sezione "Sviluppo del codice" a pagina 193 prima di consultare quest'appendice per i dettagli.

Associazione all'IFS (iSeries)

400 Le informazioni contenute in questa sezione si applicano solo se il proprio WebSphere Commerce Server di destinazione è in esecuzione su una piattaforma iSeries.

Se il WebSphere Commerce Server di destinazione è in esecuzione su una piattaforma iSeries, è necessario quindi associare un'unità locale della macchina di sviluppo all'IFS (Integrated File System) del server iSeries. Nelle sezioni successive di questo documento, *unità_iSeries* viene utilizzata come riferimento di questa unità locale associata all'IFS. Inoltre, *unità* definisce un'unità locale sulla macchina di sviluppo (unità non associata all'IFS).

File JAR per comandi personalizzati e bean di dati

Il codice per i comandi personalizzati e i bean di dati deve essere memorizzato in un progetto che è separato dal codice di WebSphere Commerce. Una volta completata la verifica all'interno di WebSphere Test Environment, è necessario creare un file JAR per il progetto che contiene il comando personalizzato e il codice del bean di dati e quindi posizionare il file JAR nella directory appropriata della macchina WebSphere Commerce Server di destinazione.

Per creare un file JAR per i comandi personalizzati e i bean di dati, eseguire le operazioni descritte:

1. Nello spazio di lavoro di VisualAge per Java, selezionare la scheda **Progetto**.
2. Fare clic con il pulsante destro del mouse sul progetto che contiene il codice del comando personalizzato e dei bean di dati e selezionare **Esporta**.
   Viene visualizzata la finestra SmartGuide per l'esportazione.
3. Selezionare **File JAR** e fare clic su **Avanti**.
4. Nel campo **file Jar**, immettere quanto segue:
   unità:\WebSphere\CommerceServerDev*directory_temp*\
   *nome_file_jar*_1.jar
   dove *unità*:\WebSphere\CommerceServerDev*directory_temp*\ è una directory temporanea che ha spazio sufficiente per il file JAR e *nome_file_jar*_1.jar è il nome del file JAR con _1 accodato.
5. Selezionare gli attributi per il file JAR nel modo seguente:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
bean	Selezionato
Includi attributi di debug nei file .class	Selezionato

   Accettare i valori predefiniti per gli altri attributi.
6. Fare clic su **Fine**.

Poiché il file JAR di implementazione non contiene le informazioni complete sulle convenzioni di denominazione dei pacchetti è necessario utilizzare un altro programma di utilità per la creazione dei pacchetti (esterno a VisualAge per Java) per creare nuovi pacchetti del file JAR. Per creare un nuovo pacchetto di questo file, eseguire le procedure descritte:

1. Da una finestra comandi, passare alla directory in cui è memorizzato *nome_file_jar*_1.jar nelle operazioni precedenti.
2. Immettere `mkdir temp1`.
3. Immettere `cd temp1`.
4. Impostare il percorso come specificato di seguito:
   `set PATH=%PATH%;`*unità*`:\WebSphere\WebSphereStudio4\bin;`
   dove *unità* è l'unità sulla quale è stato installato WebSphere Studio.
5. Immettere `jar xvf ../`*nome_file_jar*`_1.jar`.
6. Immettere `jar cvf ../`*nome_file_jar*`.jar *` (si noti che _1 è stato rimosso dal nome).
7. Passare alla directory *unità*:\WebSphere\CommerceDev*directory_temp*.

8. Se viene eseguita una distribuzione in un'istanza WebSphere Commerce locale, copiare *nome_file_jar*.jar nella seguente directory:

   unità:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_*Nomeistanza*.ear\wcstores.war\WEB-INF\lib

   . In caso contrario, lasciare il file JAR nella directory temporanea fino a quando non risulta necessario trasferire tutte le risorse dei file nella macchina WebSphere Commerce Server di destinazione.

Creazione di file JAR per nuovi bean entità

Quando si creano nuovi bean entità, è necessario memorizzare il codice in un progetto che è separato da qualsiasi codice di WebSphere Commerce. Inoltre, deve essere separato da qualsiasi codice dei comandi personalizzati e dei bean di dati. È necessario creare un nuovo bean entità in un gruppo EJB separato dai gruppi EJB che contengono i bean entità pubblici di WebSphere Commerce.

L'impiego di nuovi bean entità comporta la creazione di due file JAR. Il primo file JAR è il JAR EJB 1.1 di esportazione, e viene creato utilizzando strumenti disponibili quando è selezionata la scheda EJB. Il secondo file JAR contiene il codice di implementazione per il bean entità e viene creato dal progetto che contiene il codice del bean entità stesso. Questo secondo file JAR viene creato utilizzando strumenti disponibili quando la scheda Progetti è selezionata nello spazio di lavoro VisualAge per Java.

Creazione del file JAR EJB 1.1 di esportazione

Per creare il file JAR EJB 1.1 di esportazione per i nuovi bean entità, eseguire le procedure descritte:

1. Nello spazio di lavoro di VisualAge per Java, selezionare la scheda **EJB**.
2. Fare clic con il pulsante destro del mouse sul gruppo EJB che contiene il codice del bean entità personalizzato e selezionare **Esporta > JAR EJB 1.1**. Viene visualizzata la SmartGuide per l'esportazione in un file JAR EJB 1.1.
3. Nel campo **File Jar**, immettere quanto segue:
   unità:\WebSphere\CommerceServerDev\temp_directory*NomeFilejar*_DT.jar
   dove *unità*:\WebSphere\CommerceServerDev\temp_directory è una directory temporanea che ha spazio sufficiente per il file JAR e *NomeFilejar*_DT.jar è il nome del file JAR con il suffisso _DT accodato.

Il file JAR è stato denominato con il suffisso "_DT" per indicare che è necessario eseguire questo file JAR con lo strumento EJB Deploy fornito con WebSphere Application Server prima di distribuire il file nell'applicazione WebSphere Commerce.

4. Selezionare gli attributi per il file JAR nel modo seguente:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
Database di destinazione	DB2 Se si sta distribuendo su un database DB2, effettuare una delle scelte riportate: • Windows AIX Solaris Linux **DB2 per NT, V7.1** • 400 **DB2 per AS/400, V4** Oracle Se si sta distribuendo su un database Oracle, selezionare **Oracle, V8**
Includi attributi di debug nei file .class	Selezionato

Accettare i valori predefiniti per gli altri attributi.

5. Fare clic su **Fine**.

Il file JAR è stato creato.

Creazione del file JAR di implementazione

Per creare il file JAR di implementazione per i nuovi bean entità, eseguire le procedure descritte:

1. Nello spazio di lavoro di VisualAge per Java, selezionare la scheda **Progetto**.
2. Fare clic con il pulsante destro del mouse sul progetto che contiene il codice del bean entità personalizzato e selezionare **Esporta**.
   Viene visualizzata la SmartGuide per l'esportazione.
3. Selezionare **File JAR** e fare clic su **Avanti**.
4. Nel campo **File Jar**, immettere quanto segue:
   unità:\WebSphere\CommerceServerDev*directory_temp**NomeFilejar*_1.jar
   dove *unità*:\WebSphere\CommerceServerDev*directory_temp* è una directory temporanea che ha spazio sufficiente per il file JAR e *NomeFilejar*_1.jar è il nome del file JAR con _1 accodato.
5. Selezionare gli attributi per il file JAR nel modo seguente:

Attributo	Valore
classe	Selezionato

Attributo	Valore
java	Non selezionato
risorsa	Selezionato
bean	Selezionato
Includi attributi di debug nei file .class	Selezionato

Accettare i valori predefiniti per gli altri attributi.

6. Fare clic su **Fine**.

Poiché il file JAR di implementazione non contiene le informazioni complete sulle convenzioni di denominazione dei pacchetti è necessario utilizzare un altro programma di utilità per la creazione dei pacchetti (esterno a VisualAge per Java) per creare nuovi pacchetti del file JAR. Per creare un nuovo pacchetto di questo file, eseguire le procedure descritte:

1. Da una finestra comandi, passare alla directory in cui è memorizzato *NomeFilejar*_1.jar nelle operazioni precedenti.
2. Immettere `mkdir temp1`.
3. Immettere `cd temp1`.
4. Impostare il percorso come specificato di seguito:
   `set PATH=%PATH%;`*`unità`*`:\WebSphere\WebSphereStudio4\bin;`
   dove *unità* è l'unità sulla quale è stato installato WebSphere Studio.
5. Immettere `jar xvf ../`*`NomeFilejar`*`_1.jar`.
6. Immettere `jar cvf ../`*`NomeFilejar`*`.jar *` (si noti che la parte _1 è stata rimossa dal nome).
7. Passare alla directory
   unità`:\WebSphere\CommerceServerDev\`*`directory_temp`*.
8. Se viene eseguita una distribuzione in un'istanza WebSphere Commerce locale, copiare il file *NomeFilejar*_1.jar nella seguente directory:
   unità`:\WebSphere\CommerceServer\temp\lib`
   In caso contrario, lasciare il file JAR nella directory temporanea fino a quando non è necessario trasferire tutte le risorse file nel WebSphere Commerce Server di destinazione.

Creazione di file JAR per i bean entità personalizzati di WebSphere Commerce

È possibile estendere qualsiasi dei bean entità pubblici di WebSphere Commerce. Questi bean vengono trovati nei seguenti gruppi EJB:

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction

- WCSCatalog
- WCSCommon
- WCSContract
- WCSCoupon
- WCSFulfillment
- WCSInventory
- WCSMessageExtensions
- WCSOrder
- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

Se si desidera estendere un bean entità pubblico di WebSphere Commerce, è necessario creare un file JAR EJB 1.1 di esportazione per il gruppo EJB che contiene il bean entità pubblico di WebSphere Commerce .

Creazione del file JAR EJB 1.1 di esportazione

Per creare il file JAR EJB 1.1 di esportazione per il gruppo EJB che contiene il bean entità modificato, eseguire le procedure descritte:

1. Nello spazio di lavoro di VisualAge per Java, selezionare la scheda **EJB**.
2. Fare clic con il pulsante destro del mouse sul gruppo EJB che contiene il codice del bean entità personalizzato e selezionare **Esporta > JAR EJB 1.1**. Viene visualizzata la SmartGuide per l'esportazione in un file JAR EJB 1.1.
3. Nel campo **Jar file**, immettere quanto segue:
   unità:\WebSphere\CommerceServerDev*directory_temp*\Cust_*NomeGruppoEJB*-ejb_DT.jar
   dove *unità*:\WebSphere\CommerceServerDev*directory_temp* è una directory temporanea che ha spazio sufficiente per il file JAR e Cust_*NomeGruppoEJB*-ejb_DT.jar è il nome del file JAR con il suffisso _DT accodato.

Il file JAR è stato denominato con il suffisso "_DT" per indicare che è necessario eseguire questo file JAR con lo strumento EJB Deploy fornito con WebSphere Application Server prima di distribuire il file nell'applicazione WebSphere Commerce.

4. Selezionare gli attributi per il file JAR nel modo seguente:

Attributo	Valore
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
Database di destinazione	DB2 Se si sta distribuendo su un database DB2, effettuare una delle scelte riportate: • Windows AIX Solaris Linux **DB2 per NT, V7.1** • 400 **DB2 per AS/400, V4** Oracle Se si sta distribuendo su un database Oracle, selezionare **Oracle, V8**
Includi attributi di debug nei file .class	Selezionato

Accettare i valori predefiniti per gli altri attributi.

5. Fare clic su **Fine**.

Creazione del file JAR client

Per creare il file client JAR, effettuare le seguenti operazioni:

1. Con la scheda EJB selezionata, evidenziare tutti i gruppi EJB di WebSphere Commerce (il nome comincia per WCS). Con i gruppi evidenziati, fare clic con il pulsante destro del mouse e selezionare **Esporta > Client JAR**. Si apre il manuale Export SmartGuide.
2. Nel campo **File JAR**, immettere quanto segue:
   unità:\WebSphere\CommerceServerDev*directory_temp*\wcsejsclient.jar
3. Selezionare gli attributi nel seguente modo:

Attributo	Valore
bean	Selezionato
classe	Selezionato
java	Non selezionato
risorsa	Selezionato
Includi attributi di debug nei file .class	Selezionato

Accettare i valori predefiniti per gli altri attributi.

4. Fare clic su **Fine**.

Il file JAR è stato creato.

Memorizzazione delle risorse su WebSphere Commerce Server di destinazione

Le risorse correlate al proprio codice personalizzato devono essere copiate su WebSphere Commerce Server di destinazione. Queste risorse includono i file JAR per i comandi personalizzati, i bean di dati ed entità. È anche possibile avere nuove maschere JSP e grafici che ne supportano la personalizzazione.

AIX Solaris Linux Effettuare tutte le operazioni di distribuzione sul WebSphere Commerce Server di destinazione utilizzando l'utente creato durante l'esecuzione delle procedure descritte nella sezione *"Esecuzione dello script di post-installazione"* del manuale *WebSphere Commerce Guida all'installazione*. L'impostazione predefinita è `wasuser`. Inoltre, accertarsi che le risorse del file (ad esempio, i file JAR) e le directory nelle quali sono ubicate tali risorse abbiano le autorizzazioni in lettura, scrittura ed esecuzione per questo utente.

400 Accertarsi che le autorizzazioni per le directory `/QIBM/UserData/WebCommerce/instances/`*`Nomeistanza`* e `/QIBM/UserData/WebCommerce/instances/`*`Nomeistanza`*`/working` includano l'utente `QEJB`. Aggiungere questo utente ad entrambe le directory, impostando Data Authority su `*RWX`.

Nella seguente tabella vengono riportate le directory standard nelle quali le risorse sono memorizzate su un WebSphere Commerce Server di destinazione in esecuzione su Windows NT o Windows 2000.

Tabella 12.

Tipo di risorsa	Ubicazione directory sul WebSphere Commerce Server di destinazione
File JAR per comandi e logica databean	*`unità`*`:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*`Nomeistanza`*`.ear\wcstores.war\WEB-INF\lib`
JAR EJB 1.1 di esportazione creato utilizzando VisualAge per Java	*`unità`*`:\WebSphere\CommerceServer\temp` **Nota:** Questo file viene quindi passato come file di immissione allo strumento EJBDeploy per generare il codice di distribuzione che può essere utilizzato in WebSphere Application Server V4.0.
File JAR del codice client di EJB	*`unità`*`:\WebSphere\CommerceServer\temp\lib` **Nota:** Questo file viene solo utilizzato nel class path per lo strumento EJBDeploy.
File JAR del codice di implementazione EJB	*`unità`*`:\WebSphere\CommerceServer\temp\lib` **Nota:** Questo file viene quindi aggiunto all'enterprise application come risorsa file.

Tabella 12. (Continua)

Tipo di risorsa	Ubicazione directory sul WebSphere Commerce Server di destinazione
Maschere JSP	*unità*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*Nomeistanza*.ear\wcstores.war*dirNegozio*
Immagini	*unità*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*Nomeistanza*.ear\wcstores.war*dirNegozio*\images

Per memorizzare le risorse sulla macchina WebSphere Commerce Server di destinazione, eseguire le operazioni descritte di seguito:

1. Individuare i file JAR del comando e del codice del bean di dati. Essi si trovano in una delle seguenti directory sulla macchina di sviluppo:
   - Windows *unità*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*Nomeistanza*.ear\wcstores.war\WEB-INF\lib
   - Windows *unità*:\WebSphere\CommerceServerDev*directory_temp*
2. Copiare i file JAR del comando e del codice del bean di dati in una delle seguenti directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*Nomeistanza*.ear\wcstores.war\WEB-INF\lib
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/WEB-INF/lib
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/WEB-INF/lib
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/WEB-INF/lib
   - 400 /QIBM/UserData/WebASAdv4/*NomeIstanzaAmmin_WAS*/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/WEB-INF/lib
3. Individuare i file JAR EJB 1.1 di esportazione per i nuovi gruppi EJB, così come per i bean entità modificati di WebSphere Commerce, nella seguente directory della macchina di sviluppo:
   Windows *unità*:\WebSphere\CommerceServerDev*temp_directory*
4. Copiare i file JAR EJB 1.1 di esportazione in una delle seguenti directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp

- Solaris `/opt/WebSphere/CommerceServer/temp`
- Linux `/opt/WebSphere/CommerceServer/temp`
- 400 `/QIBM/UserData/WebCommerce/instances/Nomeistanza/temp`

5. Se sono stati creati nuovi bean enterprise, individuare il file JAR di implementazione per questi bean nella seguente directory della macchina di sviluppo:
   Windows *unità*`:\WebSphere\CommerceServerDev\`*directory_temp*
6. Copiare il file JAR di implementazione per i nuovi bean enterprise nella seguente directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*`:\WebSphere\CommerceServer\temp\lib`
   - AIX `/usr/WebSphere/CommerceServer/temp/lib`
   - Solaris `/opt/WebSphere/CommerceServer/temp/lib`
   - Linux `/opt/WebSphere/CommerceServer/temp/lib`
   - 400 `/QIBM/UserData/WebCommerce/instances/`*Nomeistanza*`/temp/lib`
7. Se sono stati modificati dei bean entità di WebSphere Commerce esistenti, individuare il file JAR client nella seguente directory della macchina di sviluppo:
   Windows *unità*`:\WebSphere\CommerceServerDev\`*temp_directory*
8. Copiare il file JAR client in una delle seguenti directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*`:\WebSphere\CommerceServer\temp\lib`
   - AIX `/usr/WebSphere/CommerceServer/temp/lib`
   - Solaris `/opt/WebSphere/CommerceServer/temp/lib`
   - Linux `/opt/WebSphere/CommerceServer/temp/lib`
   - 400 `/QIBM/UserData/WebCommerce/instances/`*Nomeistanza*`/temp/lib`
9. Copiare tutte le maschere JSP nelle seguenti directory sulla macchina WebSphere Commerce Server di destinazione:
   - Windows *unità*`:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*Nomeistanza*`.ear\wcstores.war\`*directory_negozio*
   - AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*Nomeistanza*`.ear/wcstores.war/`*directory_negozio*
   - Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*Nomeistanza*`.ear/wcstores.war/`*directory_negozio*

- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/*directory_negozio*
- 400 /QIBM/UserData/WebASAdv4/*NomeIstanzaAmmin_WAS*/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/*directory_negozio*

dove *directory_negozio* è la directory per il proprio negozio.

10. Copiare tutte le immagini nelle seguenti directory presenti sulla macchina WebSphere Commerce Server di destinazione:
    - Windows *unità*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*Nomeistanza*.ear\wcstores.war*directory_negozio*\images
    - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/*directory_negozio*/images
    - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/*directory_negozio*/images
    - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/*directory_negozio*/images
    - 400 /QIBM/UserData/WebASAdv4/*NomeIstanzaAmmin_WAS*/installedApps/WC_Enterprise_App_*Nomeistanza*.ear/wcstores.war/*directory_negozio*/images

    dove *directory_negozio* è la directory per il proprio negozio.

Aggiornamento del database di destinazione

Quando il WebSphere Commerce Server di destinazione utilizza un database diverso da quello della macchina di sviluppo, è necessario eseguire tutti gli aggiornamenti effettuati al database di sviluppo nel database utilizzato dal WebSphere Commerce Server di destinazione. Ciò include gli aggiornamenti per la registrazione di comandi nuovi o modificati e la creazione di tabelle aggiuntive e di politiche di controllo accessi per le nuove risorse create.

Per ulteriori informazioni sul caricamento delle politiche di controllo accessi (compresi la sintassi dei comandi per diverse piattaforme e i requisiti per le autorizzazioni delle directory), fare riferimento al manuale *WebSphere Commerce Access Control Guide*.

400 È necessario disporre di un programma di utilità per l'esecuzione di istruzioni SQL. È possibile utilizzare il Client Access Express V5R1 (installazione completa). Per aprire questo programma di utilità, effettuare le seguenti operazioni:

1. Aprire **Operations Navigator**.
2. Una volta aperto il pannello di navigazione delle operazioni, viene richiesto di accedere ad un particolare sistema. Accertarsi di selezionare la macchina iSeries di destinazione e di utilizzare la password e il profilo utente dell'istanza di WebSphere Commerce. Ciò assicura che tutte le tabelle create appartengono al profilo utente dell'istanza di WebSphere Commerce.
3. Fare clic sul proprio sistema nel pannello di destra, quindi fare clic con il pulsante destro del mouse su **DATABASE** e selezionare **Run SQL Scripts** dall'elenco a discesa.
   Viene visualizzata la finestra di esecuzione degli script SQL. Mediante l'utilizzo di questa finestra è possibile eseguire il taglia e incolla delle istruzioni SQL o di aprire uno script SQL. È possibile impostare il proprio schema predefinito utilizzando le opzioni di impostazione di Connection/JDBC.

Generazione del codice di distribuzione

In questa sezione vengono descritte le modalità di utilizzo dello strumento di distribuzione EJB per la generazione del codice di distribuzione dei bean enterprise contenuti nel file JAR EJB 1.1 di esportazione. Questo strumento è fornito con WebSphere Application Server.

Per generare il codice di distribuzione, effettuare le seguenti operazioni:

1. Alla richiesta comandi sul WebSphere Commerce Server di destinazione, passare alla seguente directory:
   - Windows `unità:\WebSphere\CommerceServer\temp`
   - AIX `/usr/WebSphere/CommerceServer/temp`
   - Solaris `/opt/WebSphere/CommerceServer/temp`
   - Linux `/opt/WebSphere/CommerceServer/temp`
2. 400 Alla richiesta comandi sul WebSphere Commerce Server di destinazione, immettere i seguenti comandi:
   ```
   STRQSH
   cd /QIBM/UserData/WebCommerce/instances/Nomeistanza/temp
   ```
3. Aggiungere temporaneamente lo strumento al percorso di sistema immettendo il seguente comando:
   - Windows `PATH=unità:\WebSphere\AppServer\deploytool;%PATH%`

- AIX `PATH=/usr/WebSphere/AppServer/deploytool:$PATH`
- Solaris `PATH=/opt/WebSphere/AppServer/deploytool:$PATH`
- Linux `PATH=/opt/WebSphere/AppServer/deploytool:$PATH`
- 400 `PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH`

  `CP=`*`ClassPathdiFileJARDip`*

  dove *ClassPathdiFileJARDip* è il class path dei file JAR dipendenti. Se si modifica un'enterprise di WebSphere Commerce esistente, è necessario includere i file `wcsejsclient.jar`, `wcsejbimpl.jar` e `xml4j.jar` nel class path dei file JAR dipendenti. Per questo motivo, il seguente esempio fornisce un class path per un caso in cui un bean enterprise di WebSphere Commerce esistente è stato modificato:

  ```
  CP=/QIBM/UserData/WebCommerce/instances/nomeistanza/temp/lib/
  wcsejsclient.jar:
  /QIBM/UserData/WebASAdv4/NomeIstanzaAmmin_WAS/installedApps/
  WC_Enterprise_App_nomeistanza.ear/lib/wcsejbimpl.jar:
  /QIBM/UserData/WebASAdv4/NomeIstanzaAmmin_WAS/installedApps/
  WC_Enterprise_App_nomeistanza.ear/lib/xml4j.jar
  ```

  Nota: Le interruzioni di riga nell'esempio di class path precedente sono state inserite unicamente per agevolare la visualizzazione. È necessario immettere le informazioni del class path su un'unica riga.

4. Solaris Linux Per evitare di superare il limite di caratteri nell'interfaccia della riga comandi, viene utilizzato uno script per richiamare il comando `ejbdeploy`. Per creare questo file script e richiamare il comando, effettuare quanto segue:
   a. Esplorare la seguente directory:

      `/opt/WebSphere/AppServer/bin`

   b. Creare un nuovo file e ridenominare lo script appropriatamente. Ad esempio, ridenominare il file `myejbd.sh`.
   c. Includere le seguenti informazioni nel file `myejbd.sh`:

      ```
      #!/bin/sh
      ./ejbdeploy.sh EJBGroupJARFile DirLavoro OutputJARFile
      -nowarn -keep -35 -cp ClassPathOfDepJARFiles
      ```

      dove le variabili dispongono delle stesse definizioni di quelle riportate nel passo 5 (tale passo non viene eseguito dall'utente). Di seguito è riportato un esempio delle parti da includere nel file script, con i valori per le variabili compresi:

      ```
      #!/bin/sh
      ./ejbdeploy.sh
      /opt/WebSphere/CommerceServer/temp/CustomizedWCSUserDeployed_DT.jar
      ```

```
. /opt/WebSphere/CommerceServer/temp/CustomizedWCSUserDeployed.jar
-nowarn -keep -35 -cp "/opt/WebSphere/CommerceServer/temp/lib/
wcsejsclient.jar:/opt/WebSphere/AppServer/installedApps/
WC_Enterprise_App_demo.ear/lib/wcsejbimpl.jar:/opt/WebSphere/
AppServer/installedApps/WC_Enterprise_App_demo.ear/lib/xml4j.jar"
```

Nota: Tutte le interruzioni di riga dopo il comando `./ejbdeploy.sh` sono state inserite unicamente per agevolare la visualizzazione. Non sono consentite interruzioni di riga dopo il comando `./ejbdeploy.sh` nel file.

Salvare il file script in modo che sia eseguibile.

d. Richiamare il file script immettendo quanto segue:

```
./myejbd.sh
```

5. Windows AIX 400 Immettere il comando `ejbdeploy` come segue:

Windows 400

```
ejbdeploy FileJARGruppoEJB_DT DirLavoro FileJAROutput -nowarn -keep -35 -cp
   ClasspathdiFileJARDip
```

AIX

```
/usr/WebSphere/AppServer/bin/ejbdeploy.sh EJBGroupJARFile_DT DirLavoro
OutputJARFile -nowarn -keep -35
-cp ClassPathOfDepJARFiles
```

dove:

- *EJBGroupJARFile*_DT è il nome del file JAR per il gruppo EJB. Ad esempio, se è stato modificato un bean nel gruppo EJB di WCSUser, un esempio d'uso su una piattaforma Windows è

  ```
  unità:\WebSphere\CommerceServer\temp\CustomizedWCSUser_DT.jar
  ```

  400 Un esempio d'uso per la piattaforma iSeries è

  ```
  /QIBM/UserData/WebCommerce/instances/Nomeistanza/temp/
  CustomizedWCSUser_DT.jar
  ```

- *DirLavoro* è il nome della directory in cui sono memorizzati i file temporanei richiesti per la creazione del codice.
- *OutputJARFile* è il nome completo del file JAR di emissione. Ad esempio, è possibile immettere `CustomizedWCSUserDeployed.jar`.
- `-nowarn` è un parametro facoltativo che elimina i messaggi informativi e di avvertenza.
- `-keep` è un parametro facoltativo per mantenere la directory in uso dopo aver eseguito il comando ejbdeploy.

- `-35` è un parametro obbligatorio che utilizzerà le stesse regole di corrispondenza dall'alto verso il basso per i bean entità utilizzate nello strumento di distribuzione EJB fornito con WebSphere Application Server, Versione 3.5.
- `-cp` *`ClasspathdiFileJARDip`* è il class path dei file JAR dipendenti. Quando si modifica un enterprise di WebSphere Commerce esistente, è necessario includere i file `wcsejsclient.jar`, `wcsejbimpl.jar` e `xml4j.jar` nel class path dei file JAR dipendenti. Per questo motivo, il seguente esempio fornisce un class path per un caso in cui un bean enterprise di WebSphere Commerce esistente è stato modificato:

  ```
  "unità:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
  unità:\WebSphere\AppServer\installedApps\
  WC_Enterprise_App_nomeistanza.ear \lib\wcsejbimpl.jar;
  unità:\WebSphere\AppServer\installedApps\
  WC_Enterprise_App_nomeistanza.ear\lib\xml4j.jar;"
  ```

  Nota: 400 Per i valori class path, immettere `-cp $CP` dove `CP` è stato precedentemente definito con le voci class path appropriate. Inoltre, le virgolette non sono richieste.

Modifica del livello di isolamento delle transazioni dei bean entità

In questa sezione vengono descritte le modalità d'uso dell'utility della riga comandi `modifyIsolationLevel` per l'impostazione del livello di isolamento delle transazioni dei bean entità al livello corretto per il tipo di database.

Per eseguire il comando `modifyIsolationLevel`, effettuare le seguenti operazioni:

1. Dal WebSphere Commerce Server di destinazione, utilizzare una richiesta comandi per passare alla seguente directory:
   - Windows *`unità`*`:\WebSphere\CommerceServer\bin`
   - AIX `/usr/WebSphere/CommerceServer/bin`
   - Solaris `/opt/WebSphere/CommerceServer/bin`
   - Linux `/opt/WebSphere/CommerceServer/bin`
2. 400 Immettere i seguenti comandi:

   ```
   STRQSH
   cd /QIBM/ProdData/WebCommerce/bin
   ```

3. È necessario utilizzare il comando `modifyIsolationLevel`, la cui sintassi è riportata di seguito:

   Windows AIX 400

   ```
   modifyIsolationLevel -jarFile nome_file_jar.jar
      -logFile nome_file_log -dbType tipo_db
   ```

Solaris Linux

```
./modifyIsolationLevel.sh -jarFile nome_file_jar.jar
   -logFile nome_file_log -dbType tipo_db
```

dove

- *nome_file_jar.jar* è il nome del file JAR che contiene il codice personalizzato
- *nome_file_log* è il nome completo del file in cui registrare le informazioni
- *tipo_db* è il tipo di database in uso. Immettere DB2 oppure ORACLE

Di seguito viene riportato un esempio del comando `modifyIsolationLevel` con tutti i valori specificati per l'utilizzo su una piattaforma Windows:

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

400 Di seguito viene riportato un esempio del comando`modifyIsolationLevel` con tutti i valori specificati per l'utilizzo su una piattaforma iSeries:

```
modifyIsolationLevel -jarFile
   /QIBM/UserData/WebCommerce/instances/Nomeistanza/temp/
   CustomizedWCSUserDeployed.jar -logFile /QIBM/UserData/WebCommerce/
   instances/Nomeistanza/logs/output.log -dbType DB2
```

Le interruzioni di riga nei precedenti esempi sono state inserite unicamente per agevolare la visualizzazione.

Nota: I nomi dei parametri operano una distinzione tra maiuscolo e minuscolo, ossia `jarFile` non è lo stesso di `jarfile`. Assicurarsi di immettere correttamente i nomi dei parametri.

Se la finestra comandi non visualizza eccezioni, il comando è stato eseguito correttamente. Al termine dell'esecuzione, è possibile notare che la data e l'orario del file JAR sono modificati.

Esportazione dell'enterprise application di WebSphere Commerce corrente

In questa sezione vengono descritte le modalità di utilizzo della WebSphere Application Server Console di gestione per l'esportazione dell'enterprise application di WebSphere Commerce corrente.

Per esportare l'enterprise application corrente da WebSphere Application Server, effettuare le seguenti operazioni:

1. Accertarsi di avere la seguente directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working
   - 400 /QIBM/UserData/WebCommerce/instances/*Nomeistanza*/working

   Se questa directory non esiste, crearla ora.

   Nota: AIX Solaris Linux Assicurarci che l'autorizzazione per la directory /working sia impostata sull'utente creato durante l'esecuzione delle operazioni riportate nella sezione *"Esecuzione dello script di post-installazione"* del manuale *WebSphere Commerce Guida all'installazione*.

   400 Accertarsi che le autorizzazioni per le directory /QIBM/UserData/WebCommerce/instances/*Nomeistanza* e /QIBM/UserData/WebCommerce/instances/*Nomeistanza*/working includano l'utente QEJB. Aggiungere questo utente ad entrambe le directory, impostando Data Authority su *RWX.
2. Aprire la WebSphere Application Server Console di gestione.
3. Espandere **Dominio di gestione WebSphere**.
4. Espandere **Enterprise Application**.
5. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, espandere l'applicazione **demo** e selezionare **Esporta applicazione**.
6. Nel campo **Esporta directory**, immettere quanto segue:
   - Windows *unità*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working
   - 400 /QIBM/UserData/WebCommerce/instances/*Nomeistanza*/working

   Viene esportata l'intera applicazione, incluse tutte le risorse presenti nel file WC_Enterprise_App_*Nomeistanza*.ear (dove *Nomeistanza* è il nome dell'istanza di WebSphere Commerce).

Esportazione delle informazioni sulla configurazione per i bean enterprise

In questa sezione vengono descritte le modalità di utilizzo dell'opzione `-export` dell'utility della riga comandi `XMLConfig` per l'esportazione delle informazioni sulla configurazione dei bean enterprise contenuti nell'enterprise application esistente.

Dopo aver esportato le informazioni sui bean dell'applicazione esistente, è necessario aggiungere manualmente le informazioni per tutti i nuovi bean che si sta aggiungendo all'applicazione.

Per esportare queste informazioni di configurazione, effettuare le seguenti operazioni:

1. Copiare il file `was.export.app.xml` dalla seguente directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*`:\WebSphere\CommerceServer\xml\config`
   - AIX `/usr/WebSphere/CommerceServer/xml/config`
   - Solaris `/opt/WebSphere/CommerceServer/xml/config`
   - Linux `/opt/WebSphere/CommerceServer/xml/config`
   - 400 `/QIBM/ProdData/WebCommerce/xml/config`

   nella seguente directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*`:\WebSphere\CommerceServer\working`
   - AIX `/usr/WebSphere/CommerceServer/working`
   - Solaris `/opt/WebSphere/CommerceServer/working`
   - Linux `/opt/WebSphere/CommerceServer/working`
   - 400 `/QIBM/UserData/WebCommerce/instances/`*Nomeistanza*`/working`

     dove
     - *Nomeistanza* è il nome dell'istanza di WebSphere Commerce.
2. Windows AIX Solaris Linux Aprire il file `was.export.app.xml` in un editor di testo. In questo file, sostituire tutte le occorrenze di`$Enterprise_Application_Name$` con

   `WebSphere Commerce Enterprise Application - `*Nomeistanza*

   dove *Nomeistanza* è il nome dell'istanza di WebSphere Commerce (ad esempio, `demo`). Salvare questo file.

Nota: Il valore che si immette deve corrispondere al valore visualizzato per la propria istanza nella WebSphere Application Server Console di gestione.

3. 400 Aprire il file `was.export.app.xml` in un editor di testo. In questo file, sostituire tutte le occorrenze di`$Enterprise_Application_Name$` con

   Nomeistanza - WebSphere Commerce Enterprise Application

   dove *Nomeistanza* è il nome dell'istanza di WebSphere Commerce (ad esempio, demo). Salvare questo file.

   Nota: Il valore che si immette deve corrispondere al valore visualizzato per la propria istanza nella WebSphere Application Server Console di gestione.

4. Windows AIX Solaris Linux Alla richiesta comandi, passare alla seguente directory:
   - Windows *unità*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working

5. 400 Ad una richiesta comandi, immettere quanto segue:

```
STRQSH
PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
cd /QIBM/UserData/WebCommerce/instances/Nomeistanza/working
```

6. Windows AIX Solaris Linux >Richiamare lo strumento XMLConfig per eseguire un'esportazione parziale immettendo il seguente comando:

   Windows

```
xmlConfig -export OutputFile.xml -partial was.export.app.xml
   -adminNodeName nomeHostWas
```

   AIX

```
/usr/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
   -partial was.export.app.xml -adminNodeName nomeHostWas
   -nameServiceHost nomeHostWas -nameServicePort PortaAmminWas
```

   Solaris Linux

```
/opt/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
   -partial was.export.app.xml -adminNodeName nomeHostWas
   -nameServiceHost nomeHostWas -nameServicePort PortaAmminWas
```

dove

- *nomeHostWas* è il nome del nodo in WebSphere Application Server contenente l'enterprise application corrente.
- OutputFile.xml è il nome del file creato in seguito all'esecuzione di questo comando e was.export.app.xml è il file modificato al passo 2.
- *PortaAmminWas* è la porta di gestione di WebSphere Application Server.

7. 400 Richiamare lo strumento XMLConfig per eseguire un'esportazione parziale immettendo il seguente comando:

```
xmlConfig -export OutputFile.xml -partial was.export.app.xml
-adminNodeName nomeHostWas -nameServiceHost nomeHostWas
-nameServicePort PortaAmminWas -instance NomeIstanzaWas
```

dove

- *nomeHostWas* è il nome del nodo in WebSphere Application Server contenente l'enterprise application corrente.

  Nota: Il valore per *nomeHostWas* è sensibile al maiuscolo/minuscolo e deve corrispondere al valore riportato nella configurazione TCP/IP. Utilizzare un processore per riga comandi per accedere a CFGTCP, opzione 12 per il controllo del nome host.
- OutputFile.xml è il nome del file creato in seguito all'esecuzione di questo comando e was.export.app.xml è il file modificato al passo 3.
- *PortaAmminWas* è la porta di gestione di WebSphere Application Server.
- *NomeIstanzaWas* è il nome dell'istanza di WebSphere Application Server.

Dopo l'esportazione delle informazioni sulla configurazione per ciascuno dei bean contenuti nell'enterprise application corrente, è necessario aggiungere una nuova stanza che descriva ciascun bean enterprise che si intende aggiungere all'applicazione. Ad esempio, se si dispone di un nuovo bean entità denominato "Bonus", è necessario aggiungere una stanza che descriva tale bean. Inoltre, occorre sostituire una variabile presente nel file di configurazione per specificare il nome esatto del file .ear per il proprio enterprise application.

Per effettuare questi aggiornamenti al file OutputFile.xml, effettuare le seguenti operazioni:

1. Passare alla seguente directory sul WebSphere Commerce Server di destinazione:
   - Windows *unità*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working

- 400 /QIBM/UserData/WebCommerce/instances/ *Nomeistanza*/working

2. Aprire il file OutputFile.xml con un editor di testo.
3. Individuare la tag <ear-file-name> e sostituire il valore con quello di seguito riportato:
   - Windows *unità*:\WebSphere\CommerceServer\working\ WC_Enterprise_App_*Nomeistanza*.ear
   - AIX /usr/WebSphere/CommerceServer/working/ WC_Enterprise_App_*Nomeistanza*.ear
   - Solaris /opt/WebSphere/CommerceServer/working/ WC_Enterprise_App_*Nomeistanza*.ear
   - Linux /opt/WebSphere/CommerceServer/working/ WC_Enterprise_App_*Nomeistanza*.ear
   - 400 /QIBM/UserData/WebCommerce/instances/*Nomeistanza*/ working/WC_Enterprise_App_*Nomeistanza*.ear
4. È necessario inoltre aggiungere una nuova stanza per ciascun nuovo bean che si sta aggiungendo all'enterprise application. Il seguente esempio mostra come aggiungere un nuovo bean, definito "Bonus".

Windows AIX Solaris Linux

```
<ejb-module name="yourEJBGroup">
   <jar-file>FileJARDistribuitoUtente.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         WebSphere Commerce Server - Nomeistanza/
      </application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/WebSphere Commerce DB2 DataSource Nomeistanza
            </jndi-name>
         <default-user>utente</default-user>
         <default-password>password</default-password>
      </data-source>
      <enterprise-bean-binding name="NomeBindingBean">
         <jndi-name>NomeJNDIdiBeanNomeIstanza</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

400

```
<ejb-module name="GruppoEJButente">
   <jar-file>FileJARDistribuitoUtente.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         Nomeistanza - WebSphere Commerce Server/
```

```
      </application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/Nomeistanza WebSphere Commerce DB2 DataSource
            </jndi-name>
         <default-user>utente</default-user>
         <default-password>password</default-password>
      </data-source>
      <enterprise-bean-binding name="NomeBindingBean">
         <jndi-name>NomeJNDIdiBeanNomeIstanza</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

dove

- *GruppoEJBUtente* è il nome del gruppo EJB che contiene il bean che si sta aggiungendo all'enterprise application.
- *FileJARDistribuitoUtente* è il nome del file JAR che contiene il codice di distribuzione per il gruppo EJB.
- *Nomeistanza* è il nome dell'istanza di WebSphere Commerce in cui viene distribuito il codice.
- *utente* è il nome utente del database.
- *password* è la password per l'utente del database.
- *NomeBindingBean* è il nome bind per il bean enterprise. Ad esempio, per il bean denominato Bonus, risulta `Bonus_Binding`.
- *NomeJNDIdiBeanNomeIstanza* è il nome JNDI del bean enterprise con collegata l'istanza di WebSphere Commerce. Tale nome JNDI deve corrispondere esattamente al nome del bean enterprise. Un valore di esempio è `democom/ibm/commerce/sample/objects/Bonus`. Nell'esempio, "`demo`" è il nome dell'istanza e "`com/ibm/commerce/sample/objects/Bonus`" è il nome JNDI del bean enterprise. Questo valore deve essere immesso su un'unica riga.
  È possibile controllare il nome JNDI utilizzando VisualAge per Java oppure lo Strumento di assemblaggio delle applicazioni.
  Per controllare il nome JNDI mediante VisualAge per Java, effettuare quanto segue:
  a. Fare clic con il pulsante destro del mouse sul bean e selezionare **Proprietà**.
     Viene visualizzato il nome JNDI.

Il nome JNDI può essere controllato utilizzando loStrumento di assemblaggio delle applicazioni; tuttavia, ciò richiede già la realizzazione del nuovo bean enterprise nell'enterprise application. Si accede allo Strumento di assemblaggio delle applicazioni dal menu Strumenti della WebSphere Application Server Console di gestione.

Per controllare il nome JNDI utilizzando lo Strumento di assemblaggio delle applicazioni, effettuare quanto segue:

a. Aprire il file .ear che contiene il bean.
b. Espandere il modulo EJB che contiene il bean.
c. Selezionare il bean.
d. Fare clic sulla scheda **Bindings**.
   Viene visualizzato il nome JNDI.

L'utilizzo di uno qualsiasi di questi metodi mostra il nome JNDI, ma è ancora necessario collegare il nome dell'istanza di WebSphere Commerce quando si aggiunge le informazioni al file XML.

Note:

a. Le interruzioni di riga nelle istruzioni delle stanze precedenti sono state inserite unicamente per agevolare la visualizzazione.
b. Verificare che il valore **$hostName$** corrisponda al nome server del nodo di amministrazione corrente. Inoltre, accertarsi che non vi siano caratteri di ritorno unitario su questa riga.
c. La specifica `<application-server-full-name>` non può essere distribuita su più di una riga.
d. Se si utilizza un database Oracle, è necessario modificare le informazioni di origine dati. Modificare la seguente riga estratta dalla precedente sezione di codice:

   ```
   <jndi-name>jdbc/WebSphere Commerce DB2 DataSource Nomeistanza
      </jndi-name>
   ```

   nella seguente:

   ```
   <jndi-name>jdbc/WebSphere Commerce Oracle DataSource Nomeistanza
      </jndi-name>
   ```

5. Se si esegue la distribuzione di bean entità di WebSphere Commerce modificati, è necessario aggiornare le stanze per tali bean in modo che vengano riportati i nomi dei file JAR che contengono il codice di distribuzione per i bean modificati.
6. Salvare il file `OutputFile.xml`.

Assemblaggio dei nuovi bean enterprise nell'enterprise application

In questa sezione vengono descritte le modalità di utilizzo dello Strumento di assemblaggio delle applicazioni per l'assemblaggio di nuovi bean enterprise in un'enterprise application esistente.

400 Poiché lo Strumento di assemblaggio delle applicazioni viene eseguito sulla piattaforma Windows, è necessario specificare l'unità che è stata associata all'IFS iSeries quando vengono richiesti i nomi percorso completi.

Tale unità viene definita *unità_iSeries*.

AIX Solaris Linux Si consiglia di aumentare il valore della dimensione heap della memoria nel file `assembly.sh` per evitare di esaurire la memoria quando si salvano file .ear nuovi o modificati.

Tale file si trova nella seguente directory:

- AIX `/usr/WebSphere/AppServer/bin`
- Solaris `/opt/WebSphere/AppServer/bin`
- Linux `/opt/WebSphere/AppServer/bin`

Per aumentare la dimensione heap della memoria, modificare la seguente riga:

```
$JAVA_HOME/jre/bin/java
```

nel modo seguente:

```
$JAVA_HOME/jre/bin/java -mx512M
```

In questa fase, viene aperto il file `.ear` dell'enterprise application creato nella sezione Esportazione dell'enterprise application di WebSphere Commerce corrente con lo strumento di assemblaggio dell'applicazione. Una volta aperta con questo strumento, effettuare le operazioni di seguito riportate per aggiungere il nuovo bean entità nell'enterprise application:

1. Importare il gruppo EJB contenente il nuovo bean entità. Il file JAR per il nuovo gruppo EJB sarà memorizzato nella sezione del modulo EJB dell'enterprise application.
2. Impostare il classpath per il nuovo bean entità per includere il file JAR di implementazione.
3. Aggiungere il file JAR di implementazione all'applicazione. Questo file JAR sarà memorizzato nella sezione File dell'enterprise application.
4. Impostare la sicurezza di WebSphere Application Server per i metodi contenuti nel nuovo bean entità.

Per assemblare il nuovo gruppo EJB nell'enterprise application, effettuare le seguenti operazioni:

1. Windows AIX Solaris Linux Eseguire una copia di backup dell'enterprise application corrente effettuando le operazioni di seguito riportate sul WebSphere Commerce Server di destinazione:
   a. Alla richiesta comandi, visualizzare la seguente directory:
      - Windows *unità*`:\WebSphere\CommerceServer\working`
      - AIX `/usr/WebSphere/CommerceServer/working`

- Solaris `/opt/WebSphere/CommerceServer/working`
- Linux `/opt/WebSphere/CommerceServer/working`

b. Effettuare una copia del file `WC_Enterprise_App_Nomeistanza.ear` esistente e ridenominarlo `WC_Enterprise_App_nomeIstanza.ear.bak`.

2. 400 Eseguire una copia di backup dell'enterprise application corrente, effettuando le seguenti operazioni:

   a. Ad una richiesta comandi, immettere quanto segue:

   ```
   STRQSH
   cd /QIBM/UserData/WebCommerce/instances/Nomeistanza/working
   cp WC_Enterprise_App_Nomeistanza.ear
   WC_Enterprise_App_Nomeistanza.ear.bak
   ```

   b. Per evitare inutili lunghe attese causate dal trasferimento di dati tra la macchina client locale e la macchina iSeries che esegue il WebSphere Application Server, creare la seguente directory sulla propria macchina client locale e poi copiare il file `WC_Enterprise_App_Nomeistanza.ear` nella seguente directory: `unità:\WebSphere\CommerceServer\working` dove *unità* è l'unità locale.

   Nota: Creare la directory `unità:\WebSphere\CommerceServer\working` se non esiste sulla propria macchina locale.

3. Aprire la WebSphere Application Server Console di gestione.
4. Dal menu **Strumenti**, selezionare **Strumento di assemblaggio applicazioni**.
5. Se si apre una finestra di benvenuti, selezionare **Annulla** per chiuderla.
6. Aprire l'enterprise application che si desidera utilizzare attenendosi alla seguente procedura:

   a. Dal menu **File**, selezionare **Apri**.

   b. Nel campo **Nome file**, immettere:

   - Windows `unità:\WebSphere\CommerceServer\working\WC_Enterprise_App_Nomeistanza.ear`
   - AIX `/usr/WebSphere/CommerceServer/working/WC_Enterprise_App_Nomeistanza.ear`
   - Solaris `/opt/WebSphere/CommerceServer/working/WC_Enterprise_App_Nomeistanza.ear`
   - Linux `/opt/WebSphere/CommerceServer/working/WC_Enterprise_App_Nomeistanza.ear`
   - 400 `unità:\WebSphere\CommerceServer\working\WC_Enterprise_App_Nomeistanza.ear`

e fare clic su **Apri**. Prima di continuare con i passi successivi, attendere l'apertura dell'applicazione. L'apertura può richiedere alcuni minuti.

7. Fare clic con il pulsante destro del mouse su **Moduli EJB** e selezionare **Importa**.
8. Nel campo **Nome file**, immettere quanto segue:
   - Windows *unità*:\WebSphere\CommerceServer\temp*FileJARDistribuitoUtente*.jar
   - AIX /usr/WebSphere/CommerceServer/temp/*FileJARDistribuitoUtente*.jar
   - Solaris /opt/WebSphere/CommerceServer/temp/*FileJARDistribuitoUtente*.jar
   - Linux /opt/WebSphere/CommerceServer/temp/*FileJARDistribuitoUtente*.jar
   - 400 *unità_iSeries*:\QIBM\UserData\WebCommerce\instances*Nomeistanza*\temp*FileJARDistribuitoUtente*.jar

   dove
   - *FileJARDistribuitoUtente* è il nome del file JAR che contiene il codice di distribuzione per il gruppo EJB
   - *unità_iSeries* è l'unità locale associata all'IFS iSeries.

   Fare clic su **Apri**, quindi nella finestra per la conferma dei valori, fare clic su **OK**.
9. Una volta importato il file *FileJARDistribuitoUtente*.jar, individuare il gruppo EJB *GruppoEJBUtente* (dove *GruppoEJBUtente* è il nome del gruppo EJB) e selezionare questo gruppo.
   Le informazioni su questo gruppo sono visualizzate nel pannello a destra.
10. Nel campo del classpath per il nuovo bean enterprise, inserire i file JAR dipendenti. Ad esempio, è possibile immettere il corrispondente file JAR di implementazione e il file JAR di implementazione dei bean entità di WebSphere Commerce, come di seguito illustrato:

    lib/*FileJarImplUtente*.jar lib/wcsejbimpl.jar
11. Fare clic su **Applicare**.
12. Aggiungere il file JAR di implementazione per il gruppo EJB all'applicazione, effettuando le seguenti operazioni:
    a. Fare clic con il pulsante destro del mouse sul nodo **File** per l'enterprise application e selezionare **Aggiungi file** (il nodo **File** per l'enterprise application si trova nella parte inferiore della struttura gerarchica. Esistono altri nodi File per i componenti dell'enterprise application, ma basta selezionare il nodo File per l'intera applicazione.

b. Nella finestra Aggiungi file fare clic su **Sfoglia**.
c. Esplorare la seguente directory:
   - Windows *unità*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp
   - Solaris /opt/WebSphere/CommerceServer/temp
   - Linux /opt/WebSphere/CommerceServer/temp
   - 400 *unità_iSeries*:\QIBM\UserData\WebCommerce\instances*Nomeistanza*\temp
d. Con questa directory evidenziata, fare clic su **Seleziona**.
e. Ritornare alla finestra Aggiungi file. Il contenuto della directory temporanea risulta visualizzato. Evidenziare la directory `lib` directory. Il contenuto della directory `lib` viene visualizzato nel pannello di destra.
f. Nel pannello di destra, selezionare il file *FileJarImplUtente* e fare clic su **Aggiungi**. Il file viene visualizzato nel pannello File selezionati.
g. Fare clic su **OK**.

13. Configurare la sicurezza per il bean entità effettuando le seguenti operazioni:
    a. Con il nodo Moduli EJB espanso, individuare ed espandere il nodo *GruppoEJBUtente*.
    b. Espandere **Entity Beans**.
    c. Espandere *BeanEntitàUtente* dove *BeanEntitàUtente* è il nome del bean entità.
    d. Fare clic su **Estensioni metodo**, quindi, nel pannello a destra, procedere come segue:
       1) Fare clic sul separatore **Avanzato**.
       2) Verificare che **Identità di sicurezza** sia selezionato.
       3) Per ciascun metodo, verificare che sia selezionato **Usa identità di server EJB**.
       4) Fare clic su **Applica** (se sono state effettuate delle modifiche).
    e. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse su **Ruoli sicurezza** nel gruppo EJB *GruppoEJBUtente* e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:
       1) Nel campo **Nome**, immettere `WCSecurityRole` e fare clic su **Applica**. Se questo ruolo già esiste, non è necessario eseguire questa operazione.
    f. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse su **Autorizzazioni metodi** nel gruppo EJB *GruppoEJBUtente* e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:

1) Nel campo **Nome autorizzazione metodo**, immettere `WCMethodPermission`
2) Nell'area di selezione **Metodi**, fare clic su **Aggiungi**. Si apre la finestra
   Aggiungi metodo.
3) Espandere *FileJARDistribuitoUtente*.**jar**, quindi **Bonus** e poi ciascuno degli elenchi di metodi **Home** e **Remote**.
4) Tenere premuto il tasto Maius e selezionare tutti i metodi home, quindi fare clic su**OK**.
5) Ripetere il processo di selezione del metodo per aggiungere anche i metodi remote (se esistono metodi remote).
6) Nell'area di selezione Ruoli, fare clic su **Aggiungi**, selezionare `WCSecurityRole` e quindi fare clic su **OK**.
7) Fare clic su **Applicare**.

14. Dal menu **File**, selezionare **Salva**.
15. Chiudere Strumento di assemblaggio delle applicazioni.
16. 400 Copiare il file `WC_Enterprise_App_`*`Nomeistanza`*`.ear` appena modificato dalla macchina locale nella macchina iSeries che esegue il WebSphere Application Server. Ovvero, copiare il file dalla seguente directory:

    `unità``:\WebSphere\CommerceServer\working`

    nella seguente directory:

    `unità_iSeries``:\QIBM\UserData\WebCommerce\instances\`*`Nomeistanza`*`\working`

    dove *`unità_iSeries`* è la lettera dell'unità associata al proprio iSeries IFS.

Una volta completato questo passo, viene creata una nuova enterprise application contenente tutta la logica precedente e la nuova logica aziendale. Questi elementi sono contenuti nel file `WC_Enterprise_App_`*`nomeIstanza`*`.ear` recentemente modificato.

Assemblaggio dei bean enterprise modificati nell'enterprise application

In questa sezione vengono descritte le modalità di utilizzo di Strumento di assemblaggio delle applicazioni per l'assemblaggio di bean enterprise di WebSphere Commerce modificati in un'enterprise application.

In questa fase, l'enterprise application viene aperta con lo strumento di assemblaggio delle applicazioni. Una volta aperta con questo strumento, è possibile effettuare le operazioni di seguito riportate per includere un bean enterprise di WebSphere Commerce modificato nell'enterprise application:

1. Effettuare una copia del class path per la versione esistente del gruppo EJB che è stato modificato.
2. Rimuovere la versione esistente del gruppo EJB che è stato modificato.
3. Importare la nuova versione del gruppo EJB modificato. Il file JAR per il nuovo gruppo EJB è memorizzato nella sezione del modulo EJB dell'enterprise application.
4. Impostare il classpath per il gruppo EJB modificato.
5. Impostare la sicurezza di WebSphere Application Server per i metodi contenuti nel bean entità modificato.

AIX Solaris Linux Si consiglia di aumentare il valore della dimensione heap della memoria nel file `assembly.sh` per evitare di esaurire la memoria quando si salvano file `.ear` nuovi o modificati.

Tale file si trova nella seguente directory:

- AIX `/usr/WebSphere/AppServer/bin`
- Solaris `/opt/WebSphere/AppServer/bin`
- Linux `/opt/WebSphere/AppServer/bin`

Per aumentare la dimensione heap della memoria, modificare la seguente riga:

```
$JAVA_HOME/jre/bin/java
```

nel modo seguente:

```
$JAVA_HOME/jre/bin/java -mx512M
```

Per assemblare il gruppo EJB modificato nell'enterprise application, effettuare le seguenti operazioni:

1. Windows AIX Solaris Linux Eseguire una copia di backup dell'enterprise application corrente effettuando le operazioni di seguito riportate sul WebSphere Commerce Server di destinazione:
   a. Alla richiesta comandi, visualizzare la seguente directory:
      - Windows *unità*`:\WebSphere\CommerceServer\working`
      - AIX `/usr/WebSphere/CommerceServer/working`
      - Solaris `/opt/WebSphere/CommerceServer/working`
      - Linux `/opt/WebSphere/CommerceServer/working`
   b. Effettuare una copia del file `WC_Enterprise_App_`*`Nomeistanza`*`.ear` esistente e ridenominarlo `WC_Enterprise_App_`*`nomeIstanza`*`.ear.bak`.

2. 400 Eseguire una copia di backup dell'enterprise application corrente, effettuando le seguenti operazioni:
   a. Ad una richiesta comandi, immettere quanto segue:
      ```
      STRQSH
      cd /QIBM/UserData/WebCommerce/instances/Nomeistanza/working
      cp WC_Enterprise_App_Nomeistanza.ear
      WC_Enterprise_App_Nomeistanza.ear.bak
      ```
   b. Per evitare inutili lunghe attese causate dal trasferimento di dati tra la macchina client locale e la macchina iSeries che esegue il WebSphere Application Server, creare la seguente directory sulla propria macchina client locale e poi copiare il file `WC_Enterprise_App_Nomeistanza.ear` nella seguente directory:
      unità`:\WebSphere\CommerceServer\working`
      dove *unità* è l'unità locale.

      Nota: Creare la directory *unità*`:\WebSphere\CommerceServer\working` se non esiste sulla propria macchina locale.
3. Aprire la WebSphere Application Server Console di gestione.
4. Dal menu **Strumenti**, selezionare **Strumento di assemblaggio applicazioni**.
5. Aprire l'enterprise application che si desidera utilizzare attenendosi alla seguente procedura:
   a. Dal menu **File**, selezionare **Apri**.
   b. Nel campo **Nome file**, immettere:
      - Windows *unità*`:\WebSphere\CommerceServer\working\WC_Enterprise_App_Nomeistanza.ear`
      - AIX `/usr/WebSphere/CommerceServer/working/WC_Enterprise_App_Nomeistanza.ear`
      - Solaris `/opt/WebSphere/CommerceServer/working/WC_Enterprise_App_Nomeistanza.ear`
      - Linux `/opt/WebSphere/CommerceServer/working/WC_Enterprise_App_Nomeistanza.ear`
      - 400 *unità*`:\WebSphere\CommerceServer\working\WC_Enterprise_App_Nomeistanza.ear`

      e fare clic su **Apri**. Prima di continuare con i passi successivi, attendere l'apertura dell'applicazione. L'apertura può richiedere alcuni minuti.
6. Fare clic su **Moduli EJB**. Nel pannello a destra vengono visualizzati i moduli EJB presenti nell'enterprise application.

7. Fare clic sul modulo EJB per il gruppo EJB che è stato modificato. Ad esempio, se è stato modificato un bean nel gruppo EJB di WCSUser, fare clic su **WCSUser**.
8. Fare clic sulla scheda Generale per visualizzare le informazioni sul class path. Copiare queste informazioni in un file di testo (ad esempio, `WCSUser_path.txt`).
9. Fare clic con il pulsante destro del mouse sul modulo EJB e selezionare **Elimina**.
10. Fare clic con il pulsante destro del mouse su **Moduli EJB** e selezionare **Importa**.
11. Nel campo **Nome file**, immettere quanto segue:
    - Windows *unità*`:\WebSphere\CommerceServer\temp\Cust_`*NomeGruppoEJB*`-ejb.jar`
    - AIX `/usr/WebSphere/CommerceServer/temp/Cust_`*NomeGruppoEJB*`-ejb.jar`
    - Solaris `/opt/WebSphere/CommerceServer/temp/Cust_`*NomeGruppoEJB*`-ejb.jar`
    - Linux `/opt/WebSphere/CommerceServer/temp/Cust_`*NomeGruppoEJB*`-ejb.jar`
    - 400 *unità_iSeries*`:\QIBM\UserData\WebCommerce\instances\`*Nomeistanza*`\temp\Cust_`*NomeGruppoEJB*`-ejb.jar`

    e fare clic su **Apri**. Nella finestra per la conferma dei valori, fare clic su **OK**.
12. Una volta importato il file *FileJARGruppoEJB*`.jar`, individuare il gruppo EJB modificato e selezionare questo gruppo.
    Le informazioni su questo gruppo sono visualizzate nel pannello a destra.
13. Aprire il file di testo contenente le informazioni sul classpath per la versione precedente del gruppo EJB. Selezionare e copiare il classpath.
14. Nel campo **Classpath** per il gruppo EJB modificato, incollare le informazioni sul class path.
15. Fare clic su **Applicare**.
16. Dal menu **File**, selezionare **Chiudi**.
17. Attendere la chiusura del file, quindi dal menu **File** selezionare **Apri** per aprire nuovamente il file `WC_Enterprise_App_`*Nomeistanza*`.ear`.
18. Configurare la sicurezza per il bean modificato effettuando le seguenti operazioni:
    a. Con il nodo Moduli EJB espanso, individuare ed espandere il nodo per il gruppo EJB modificato.

b. Espandere **Entity Beans**.
c. Espandere il gruppo EJB modificato.
d. Fare clic su **Estensioni metodo**, quindi, nel pannello a destra, procedere come segue:
   1) Fare clic sul separatore **Avanzato**.
   2) Verificare che **Identità di sicurezza** sia selezionato.
   3) Per ciascun metodo, verificare che sia selezionato **Usa identità di server EJB**.
   4) Fare clic su **Applica** (se sono state effettuate delle modifiche).
e. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse su **Ruoli sicurezza** nel gruppo EJB modificato e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:
   1) Nel campo **Nome**, immettere `WCSecurityRole` e fare clic su **Applica**. Se questo ruolo già esiste, non è necessario eseguire questa operazione.
f. Nel pannello di navigazione di sinistra, fare clic con il pulsante destro del mouse su **Autorizzazioni metodi** nel gruppo EJB modificato e selezionare **Nuovo**, quindi effettuare le seguenti operazioni:
   1) Nel campo **Nome autorizzazione metodo**, immettere `WCMethodPermission`
   2) Nell'area di selezione **Metodi**, fare clic su **Aggiungi**. Si apre la finestra
   Aggiungi metodo.
   3) Espandere *GruppoEJBModificato* e selezionare tutti i bean enterprise (tenere premuto il tasto Maius durante la selezione). Fare clic su **OK**. Tutti i bean enterprise vengono quindi visualizzati nella colonna bean Enterprise mentre tutti i metodi vengono visualizzati nella colonna Tipi.
   4) Nell'area di selezione Ruoli, fare clic su **Aggiungi**, selezionare `WCSecurityRole` e quindi fare clic su **OK**.
   5) Fare clic su **Applicare**.

19. Dal menu **File**, selezionare **Salva**.
20. Chiudere Strumento di assemblaggio delle applicazioni.
21. 400 Copiare il file `WC_Enterprise_App_`*`Nomeistanza`*`.ear` appena modificato dalla macchina locale nella macchina iSeries che esegue il WebSphere Application Server. Ovvero, copiare il file dalla seguente directory:

    `unità``:\WebSphere\CommerceServer\working`

    nella seguente directory:

    `unità_iSeries``:\QIBM\UserData\WebCommerce\instances\`*`Nomeistanza`*`\working`

dove *unità_iSeries* è la lettera dell'unità associata al proprio iSeries IFS.

Una volta completato questo passo, viene creata una nuova enterprise application contenente tutta la logica precedente e la nuova logica aziendale. Questi elementi sono contenuti nel file `WC_Enterprise_App_`*`nomeIstanza`*`.ear` recentemente modificato.

Arresto e rimozione dell'enterprise application

In questa sezione vengono descritte le modalità di utilizzo della WebSphere Application Server Console di gestione per arrestare un'enterprise application attualmente in esecuzione e quindi rimuoverla.

Per arrestare e quindi rimuovere l'enterprise application, effettuare le seguenti operazioni:

1. Aprire la Console di gestione di WebSphere Application Server.
2. Espandere **WebSphere Administrative Domain**.
3. Espandere **Nodi**.
4. Espandere ***Nomenodo*** (dove ***Nomenodo*** è il nome del nodo).
5. Espandere **Server delle applicazioni**.
6. Windows AIX Solaris Linux Fare clic con il pulsante destro del mouse sul server di applicazioni WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse su **WebSphere Commerce Server - *Nomenodo*** e selezionare **Arresta**.
7. 400 Fare clic con il pulsante destro del mouse sul server di applicazioni WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse su ***Nomeistanza* - WebSphere Commerce Server** e selezionare **Arresta**.
8. Espandere **Enterprise Applications**.
9. Windows AIX Solaris Linux Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Enterprise Application - *Nomeistanza*** e selezionare **Arresta**.
10. 400 Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione ***Nomeistanza* - WebSphere Commerce Enterprise Application** e selezionare **Arresta**.
11. Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Enterprise Application -*Nomeistanza*** (o ***Nomeistanza* - WebSphere Commerce Enterprise Application** per iSeries) e selezionare **Rimuovi**.

12. Quando viene richiesto di indicare se l'applicazione deve essere esportata, selezionare **No**.
13. Quando viene richiesto di indicare se l'applicazione deve essere rimossa, selezionare **Sì**.

Importazione dell'enterprise application

In questa sezione vengono descritte le modalità d'uso dell'utility della riga comandi XMLConfig per l'importazione di un'enterprise application.

Per importare la nuova enterprise application, attenersi alla seguente procedura:

1. Windows AIX Solaris Linux Richiamare lo strumento XMLConfig per importare l'enterprise application in WebSphere Application Server immettendo il seguente comando:

   Windows

   ```
   xmlConfig -import OutputFile.xml -adminNodeName nomeHostWas
   ```

   AIX

   ```
   /usr/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
    -adminNodeName nomeHostWas
    -nameServiceHost nomeHostWas -nameServicePort PortaAmminWas
   ```

   Solaris Linux

   ```
   /opt/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
    -adminNodeName nomeHostWas
    -nameServiceHost nomeHostWas -nameServicePort PortaAmminWas
   ```

   dove

   - *nomeHostWas* è il nome del nodo in WebSphere Application Server contenente l'enterprise application corrente.
   - *OutputFile.xml* è il file XML che descrive tutti i bean enterprise che si utilizzano.
   - *PortaAmminWas* è la porta di gestione di WebSphere Application Server.

2. 400 Richiamare lo strumento XMLConfig per importare l'enterprise application in WebSphere Application Server immettendo il seguente comando:

   ```
   STRQSH
   PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
   xmlConfig -import
   /QIBM/UserData/WebCommerce/instances/Nomeistanza/working/OutputFile.xml
   -adminNodeName nomeHostWas
    -nameServiceHost nomeHostWas -nameServicePort PortaAmminWas
   -instance NomeIstanzaWas
   ```

dove

- *OutputFile.xml* è il nome completo del file XML che descrive tutti i bean enterprise.
- *nomeHostWas* è il nome del nodo in WebSphere Application Server contenente l'enterprise application corrente.

  Nota: 400 Il valore per *nomeHostWas* è sensibile al maiuscolo/minuscolo e deve corrispondere al valore riportato nella configurazione TCP/IP. Utilizzare un processore per riga comandi per accedere a CFGTCP, opzione 12 per il controllo del nome host.
- *PortaAmminWas* è la porta di gestione di WebSphere Application Server.
- *NomeIstanzaWas* è il nome dell'istanza di WebSphere Application Server.

400 Mentre si tenta di eseguire il comando `XMLConfig -import`, si potrebbe ricevere un messaggio di errore simile al seguente:"Impossibile espandere il file ear in `/QIBM/UserData/WebAsAdv4/`*`NomeIstanzaWas`*`/installedApps/WC_Enterprise_App_`*`Nomeistanza`*`.ear`". Se si riceve questo messaggio, rimuovere o ridenominare la directory precedente ed eseguire nuovamente il comando.

3. 400 Dopo aver importato l'enterprise application, è necessario eseguire uno script per modificare le autorizzazioni di directory. Per eseguire questo script, eseguire le seguenti operazioni:
   - Ad una richiesta comandi, immettere quanto segue:

     ```
     STRSQH
     cd /QIBM/ProdData/WebCommerce/bin
     changeAuthority nomeIstanzaAmminWas Nomeistanza
     ```

     dove
     - *nomeIstanzaAmminWas* è il nome dell'istanza di gestione di WebSphere Application Server.
     - *Nomeistanza* è il nome dell'istanza di WebSphere Commerce.

Avvio dell'enterprise application

In questa sezione vengono descritte le modalità di utilizzo di WebSphere Application Server Console di gestione per aggiornare la visualizzazione e quindi avviare un'enterprise application.

1. Aprire la Console di gestione di WebSphere Application Server.
2. Espandere **Dominio di gestione WebSphere**, quindi **Nodi** e poi *Nomenodo*
3. Selezionare il nodo *Nomenodo*.

4. Fare clic sull'icona per l'**aggiornamento dell'albero secondario**.
5. Avviare l'applicazione WebSphere Commerce effettuando le seguenti operazioni:
   - Espandere **Server delle applicazioni**.
   - Windows AIX Solaris Linux Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione **WebSphere Commerce Server -** *Nomeistanza* e selezionare **Avvia**.
   - 400 Fare clic con il pulsante destro del mouse sull'applicazione WebSphere Commerce. Ad esempio, fare clic con il pulsante destro del mouse sull'applicazione *Nomeistanza* **- WebSphere Commerce Server** e selezionare **Avvia**.

Appendice C. Suggerimenti per VisualAge per Java

Questa sezione contiene alcuni suggerimenti relativi alla risoluzione dei problemi, a miglioramenti e semplificazioni all'ambiente di sviluppo.

Modifica delle proprietà per Servlet engine in WebSphere Test Environment

Le proprietà per Servlet engine in WebSphere Test Environment sono controllate dal file delle proprietà `default.servlet_engine`. In questo file, è possibile modificare la radice del documento per il server Web ed è possibile modificare la porta utilizzata da WebSphere Test Environment.

Se si desidera cambiare la porta quando WebSphere Test Environment non esegue più le sue funzioni e il riavvio non fa parte delle opzioni disponibili, effettuare le seguenti operazioni:

1. Aprire il file *percorso_installazione_VAJ*`\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine` in un editor di testo.
2. Nella stanza `<transport>`, sostituire il valore `8080` nella riga successiva con una porta disponibile.

   ```
   <arg name="port" value="8080"/>
   ```

3. Sostituire il valore `8080` nelle seguenti righe con la stessa porta specificata sopra.

   ```
   <hostname-binding hostname="localhost:8080" servlethost="default_host"/>
      <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
   ```

4. Salvare il file.
5. Aprire WebSphere Test Environment Centro di controllo e avviare Motore servlet.

Per impostazione predefinita, la radice del documento per WebSphere Test Environment è *percorso_installazione_VAJ*`\IDE\ProjectResources\IBM WebSphere Test Environment\hosts\default_host_app\web`. Per cambiare directory, procedere nel modo seguente:

1. Aprire WebSphere Test Environment Centro di controllo e interrompere Motore servlet.
2. Aprire il file *percorso_installazione_VAJ*`\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine` in un editor di testo.

3. Nella stanza <websphere-webgroup name="default_app">, sostituire <document-root>$approot$/web</document-root> con quanto segue:

   ```
   <document-root>radice_documento</document-root>
   ```

   dove *radice_documento* è la radice del documento desiderata.
4. Sostituire il valore 8080 nelle seguenti righe con la stessa porta specificata sopra.

   ```
   <hostname-binding hostname="localhost:8080" servlethost="default_host"/>
      <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
   ```
5. Salvare il file.
6. Aprire WebSphere Test Environment Centro di controllo e avviare Motore servlet.

Risoluzione dei problemi del server dei nomi permanenti

Se si verificano problemi con il server dei nomi permanenti, è possibile dover eliminare e ricreare il database del server dei nomi permanenti. Per dettagli sulla creazione di questo database, consultare *Commerce Studio Guida all'installazione*.

In alternativa, se un messaggio indica che la porta è correntemente in uso, assicurarsi che WebSphere Application Server non sia in esecuzione.

Eliminazione dei file JSP compilati

VisualAge per Java gestisce una cartella di progetto nella quale i file compilati vengono memorizzati per motivi relativi alle prestazioni. È possibile che in alcuni casi si renda necessario eliminare i file JSP compilati. Ad esempio, se si elimina un bean di dati da un file JSP, possono verificarsi errori alla successiva apertura del file stesso. In tal caso, è possibile eliminare il file JSP compilato.

Per eliminare i file JSP compilati, effettuare le seguenti operazioni:

1. In VisualAge per Java, selezionare la scheda Progetto.
2. Scorrere fino al progetto **JSP Page Compile Generate Code**.
3. Selezionare l'intero progetto (se si desidera eliminare molti file JSP compilati) oppure espandere il progetto ed eliminare solo quelli da ricompilare.

Informazioni particolari

Queste informazioni sono state sviluppate per i prodotti e i servizi offerti negli Stati Uniti d'America. IBM potrebbe non offrire ad altri paesi le funzioni, i prodotti o i servizi discussi in questo documento. Per informazioni sui prodotti o i servizi disponibili nella propria zona, contattare il rappresentante IBM locale. Qualsiasi riferimento a prodotti, programmi o servizi IBM non implica che l'IBM intenda renderli disponibili in tutti i paesi in cui opera. In sostituzione a quelli forniti dall'IBM, possono essere usati prodotti, programmi o servizi funzionalmente equivalenti che non comportino violazione dei diritti di proprietà intellettuale dell'IBM. È tuttavia responsabilità dell'utente valutare e verificare la funzionalità di tali prodotti, programmi e servizi non IBM.

L'IBM può avere brevetti o domande di brevetto in corso relativi a quanto trattato nella presente pubblicazione. La fornitura di questa pubblicazione non implica la concessione di alcuna licenza su di essi. Per eventuali domande sulle licenze, scrivere al seguente indirizzo:

IBM Director of Licensing
IBM Europe
Schoenaicher Str. 220
D-7030 Boeblingen
Deutschland

Per domande sulle licenze relative a informazioni DBCS, contattare IBM Intellectual Property Department del proprio paese oppure scrivere a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

Il seguente paragrafo non è valido per il Regno Unito o per tutti i paesi le cui leggi nazionali siano in contrasto con le disposizioni in esso contenute:

L'INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE QUESTA PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA", SENZA ALCUNA GARANZIA, ESPLICITA O IMPLICITA, IVI INCLUSE EVENTUALI GARANZIE DI COMMERCIABILITÀ ED IDONEITÀ AD UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni; quindi la presente dichiarazione potrebbe essere non essere a voi applicabile.

Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le informazioni incluse in questo documento vengono modificate su base periodica; tali modifiche verranno incorporate nelle nuove edizioni della pubblicazione. L'IBM si riserva il diritto di apportare miglioramenti e/o modifiche al prodotto o al programma descritto nel manuale in qualsiasi momento e senza preavviso.

Tutti i riferimenti a siti Web non dell'IBM contenuti in questo documento sono forniti solo per consultazione. I materiali disponibile presso i siti Web non fanno parte di questo prodotto e l'utilizzo di questi è a discrezione dell'utente.

Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente dall'IBM e dalla Selfin e diventeranno esclusiva delle stesse.

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue, Markham, Ontario L6G 1C7
Canada

Queste informazioni possono essere rese disponibili secondo condizioni contrattuali appropriate, compreso, in alcuni casi, il pagamento di un addebito.

Il programma su licenza descritto in questo manuale e tutto il materiale su licenza ad esso relativo sono forniti dall'IBM nel rispetto delle condizioni previste dalla licenza d'uso.

Tutti i dati relativi alle prestazioni contenuti in questa pubblicazione sono stati determinati in un ambiente controllato. Pertanto, i risultati ottenuti in ambienti operativi diversi possono variare in modo considerevole. Alcune misure potrebbero essere state fatte su sistemi di livello di sviluppo per cui non si garantisce che queste saranno uguali su tutti i sistemi disponibili. Inoltre, alcune misure potrebbero essere state ricavate mediante estrapolazione. I risultati possono quindi variare. Gli utenti di questa pubblicazione devono verififcare che i dati siano applicabili al loro specifico ambiente.

Le informazioni relative a prodotti non IBM sono state ottenute dai fornitori di tali prodotti. L'IBM non ha verificato tali prodotti e, pertanto, non può

garantirne l'accuratezza delle prestazioni. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti.

Tutte le dichiarazioni riguardanti la futura direzione o le intenzioni della IBM sono soggette a sostituzione o al ritiro senza preavviso, e rappresentano unicamente scopi e obiettivi della IBM stessa.

Tutti i prezzi IBM riportati sono prezzi al dettaglio suggeriti dalla IBM stessa, sono correnti e soggetti a cambiamenti senza preavviso. I prezzi al fornitore possono variare.

Queste informazioni hanno solo scopo di pianificazione. Queste informazioni possono essere soggette a variazioni prima che i prodotti descritti siano disponibili.

Questa pubblicazione contiene esempi di dati e prospetti utilizzati quotidianamente nelle operazioni aziendali, pertanto, può contenere nomi di persone, società, marchi e prodotti. Tutti i nomi contenuti nel manuale sono fittizi e ogni riferimento a nomi ed indirizzi reali è puramente casuale.

LICENZA SOGGETTA ALLE LEGGI SUL DIRITTO D'AUTORE:

Queste informazioni contengono esempi di programmi applicativi in lingua originale, che illustrano le tecniche di programmazione su diverse piattaforme operative. Potete copiare, modificare e distribuire questi esempi di programmi sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in modo conforme alle API (Application Programming Interface) a seconda della piattaforma operativa per cui gli esempi dei programmi sono stati scritti. Questi esempi non sono stati testati approfonditamente tenendo conto di tutte le condizioni possibili. La IBM, quindi, non può garantire o assicurare la affidabilità, la praticità o il funzionamento di questi programmi. Potete copiare, modificare e distribuire questi esempi di programmi sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in modo conforme alle API (Application Programming Interface) IBM.

Ogni copia o parte di tali programmi di esempio deve includere un'informazione sul copyright come questa di seguito indicata:

Se questa pubblicazione viene visualizzata in formato elettronico, Þ possibile che le fotografie e le illustrazioni a colori non vengano visualizzate.

Marchi

I seguenti termini sono marchi dell'IBM Corporation:

400	iSeries
AIX	MQSeries
AS/400	Net.Commerce
CICS	Net.Data
DB2	VisualAge
IBM	WebSphere

Windows NT è un marchio della Microsoft Corporation.

Oracle è un marchio della Oracle Corporation.

Solaris, Java, e tutti i marchi e i logo basati su Java sono marchi della Sun Microsystems, Inc.

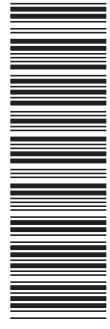
Altri nomi di prodotti, società o servizi potrebbero essere marchi o marchi di servizi di altre società.

Indice analitico

V

Numero parte: CT024IT

(1P) P/N: CT024IT



GC13-3051-00

Spine information:

IBM WebSphere Commerce Guida per il programmatore Versione 5.4