IBM WebSphere Commerce

프로그래머 안내서

버전 5.4

GA30-1614-00

IBM WebSphere Commerce

프로그래머 안내서

버전 5.4

GA30-1614-00

주!

이 책 및 이 책이 지원하는 제품을 사용하기 전에 주의사항의 일반 정보를 읽으십시오.

초판(2002년 3월) 릴리스 2

이 책은 다음 제품에 적용됩니다.

- Windows NT® 및 Windows® 2000용 IBM® WebSphere® Commerce Business Edition, 버전 5.4(프로그램 5724-A18)
- AIX®용 IBM WebSphere Commerce Business Edition, 버전 5.4(프로그램 5724-A18)
- Solaris Operating Environment Software용 IBM WebSphere Commerce Business Edition, 버전 5.4(프로그램 5724-A18)
- Linux용 IBM WebSphere Commerce Business Edition, 버전 5.4(프로그램 5724-A18)
- Windows NT 및 Windows 2000용 IBM WebSphere Commerce Studio, Business Developer Edition, 버전 5.4(프로그램 5724-A18)
- Windows NT 및 Windows 2000용 IBM WebSphere Commerce Professional Edition, 버전 5.4(프로그램 5724-A18)
- AIX용 IBM WebSphere Commerce Professional Edition, 버전 5.4 (프로그램 5724-A18)
- Solaris Operating Environment Software용 IBM WebSphere Commerce Professional Edition, 버전 5.4(프로그램 5724-A18)
- Linux용 IBM WebSphere Commerce Professional Edition, 버전 5.4(프로그램 5724-A18)
- Windows NT 및 Windows 2000IBM WebSphere Commerce Studio, Professional Developer Edition, 버전 5.4(프로그램 5724-A18)

새 개정판에 별도로 명시하지 않는 한, 위 제품의 모든 후속 릴리스와 수정에 적용됩니다. 제품 레벨에 맞는 올바른 버전을 사용하고 있는지 확인하십시오.

현지의 IBM 담당자나 IBM 지사를 통해 서적을 주문하십시오. 다음 주소에서는 책을 구비하고 있지 않습니다.

IBM에서는 귀하의 의견을 환영합니다. 다음 중 한 가지 방법으로 의견을 보내주십시오.

1. 전자 우편은 다음으로 보내주십시오.

   ibmkspoe@kr.ibm.com
2. 우편으로 보내실 경우에는 다음 주소로 우송해 주십시오.

   135-270
   서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
   한국 아이.비.엠 주식회사
   고객만족센터

독자가 IBM에 정보를 보낼 경우, 그 정보가 타당하면 IBM은 적절한 방식으로 이를 사용하거나 배포할 수 있으며, 제공한 독자는 이에 대해 책임을 지거나 사용에 제한을 받지 않습니다.

시작하기 전에

*IBM WebSphere Commerce 프로그래머 안내서*는 WebSphere Commerce 아키텍처 및 프로그래밍 모델에 대한 정보를 제공합니다. 특히 다음과 같은 주제에 대한 자세한 내용이 들어 있습니다.

- 구성요소 상호 작용
- 설계 패턴
- 지속 오브젝트 모델
- 액세스 제어
- 오류 처리 및 메시지
- 명령 구현
- 개발 도구
- 사용자 정의 코드 전개

또한 이 책에는 다음 학습 항목이 있습니다.

새 비즈니스 로직 작성

이 학습에서는 WebSphere Commerce 프로그래밍 모델에 따라 새 명령, 데이터 bean 및 엔터프라이즈 bean을 작성하는 방법을 보여줍니다. 또한 기존 상점에 로직을 통합하고 대상 WebSphere Commerce Server에 코드를 전개하는 방법을 보여줍니다.

기존 비즈니스 로직 수정 및 확장

이 학습은 두 부분으로 나누어 집니다. 첫 번째는 새 로직을 기존 제어기 명령에 추가하는 방식을 설명합니다. 두 번째는 기존 태스크 명령 및 WebSphere Commerce 엔티티 bean을 수정하는 방법을 보여줍니다. 또한 기존 상점에 수정사항을 통합하고 대상 WebSphere Commerce Server에 코드를 전개하는 방법을 보여줍니다.

이 책에 사용된 규칙

이 책은 다음과 같은 규칙을 사용합니다.

굵은체는 필드, 아이콘 또는 메뉴 선택사항의 이름과 같은 GUI(Graphical User Interface) 제어 또는 명령을 표시합니다.

모노체는 디렉토리 경로 및 정확하게 입력해야 하는 텍스트의 예를 표시합니다.

*기울임꼴*은 사용자가 값을 대체해야 하는 변수 및 강조에 사용됩니다.

이 아이콘은 태스크 완료에 도움을 주는 추가 정보를 표시합니다.

Windows Windows NT 및 Windows 2000용 WebSphere Commerce 2000 고유 정보를 나타냅니다.

AIX AIX용 WebSphere Commerce 고유 정보를 나타냅니다.

Solaris Solaris™용 WebSphere Commerce Operating Environment software 고유 정보를 나타냅니다.

400 IBM @server iSeries™ 400®용(이전에는 AS/400®WebSphere Commerce 고유 정보를 나타냅니다.

Linux Linux용 WebSphere Commerce 고유 정보를 나타냅니다.

DB2 DB2 Universal Database 고유 정보를 나타냅니다.

Oracle Oracle 고유 정보를 나타냅니다.

Professional WebSphere Commerce Professional Edition 고유 정보를 나타냅니다.

Business WebSphere Commerce Business Edition 고유 정보를 나타냅니다.

지식 요구사항

이 책은 WebSphere Commerce 응용프로그램을 사용자 정의하는 방법을 이해해야 하는 상점 개발자를 위한 책입니다. 프로그램 확장을 수행 중인 상점 개발자는 다음과 같은 영역에 대한 지식이 있어야 합니다.

- Java
- EJB 구성요소 아키텍처
- JSP 기술
- HTML
- 데이터베이스 기술
- VisualAge for Java™, Enterprise Edition, 버전 3.5

추가 정보

이 책은 앞으로 갱신될 수도 있습니다. 갱신사항은 다음 WebSphere Commerce 웹 사이트를 확인하십시오.

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`

갱신사항으로는 새 정보, 추가 학습서 또는 갱신된 학습서, 학습서와 관련된 견본 코드가 있습니다.

목차

제 1 부 개념 및 아키텍처

제 1 장 개요

WebSphere Commerce 소프트웨어 구성요소

WebSphere Commerce 서버가 작동하는 방법을 살펴보기 전에 사용자 정의 처리와 관련된 소프트웨어 구성요소에 대해 살펴보는 것이 좋습니다. 다음 도표는 이러한 소프트웨어 제품의 간단한 보기입니다.

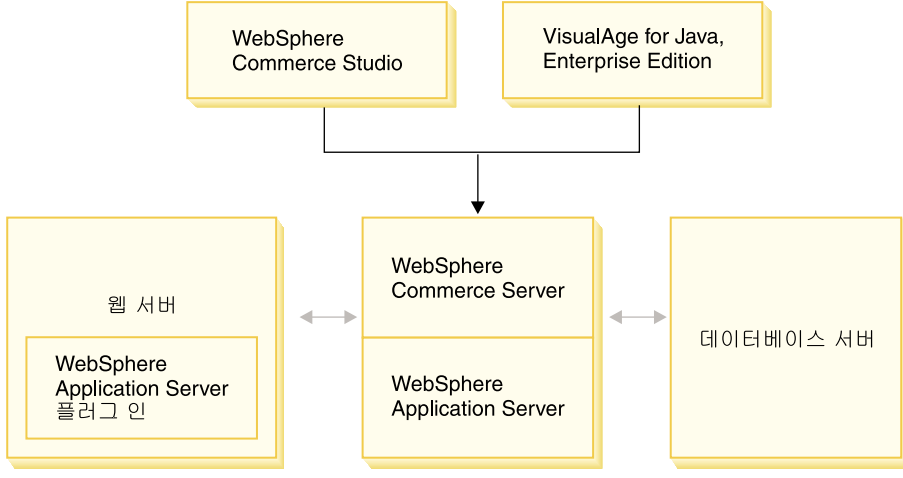


그림 1.

웹 서버는 전자 상거래 응용프로그램의 수신 HTTP 요청이 들어오는 첫 번째 지점입니다. WebSphere Application Server에 효과적으로 연결하기 위해 WebSphere Application Server 플러그인이 사용됩니다.

WebSphere Commerce Server는 WebSphere Application Server 내에서 실행되며 응용프로그램 서버의 여러 기능을 이용할 수 있습니다. 데이터베이스 서버는 상품 및 구매자 데이터를 포함한 대부분의 응용프로그램 데이터를 보관합니다. 일반적으로 WebSphere Commerce Server의 코드를 수정하거나 확장함으로써 응용프로그램을 확장할 수 있습니다. 또한 WebSphere Commerce 데이터베이스 스키마 영역 외부의 데이터를 데이터베이스에 저장해야 합니다.

개발자들은 사용자 정의된 비즈니스 로직을 작성하기 위해 두 가지 주요 도구 WebSphere Commerce Studio와 VisualAge® for Java, Enterprise Edition을 사용합니다. WebSphere Commerce Studio는 상점 입구 자원(예: JSP 템플리트)을 작성하고 관리하는 데 사용됩니다. VisualAge for Java, Enterprise Edition은 기존 기능을 확장하거나 완전히 새 기능을 작성하는 새 비즈니스 로직을 Java로 작성하는 데 사용됩니다. 응용프로그램에 데이터베이스 스키마 확장이 필요한 경우, 데이터베이스 개발자는 선호하는 데이터베이스 개발 도구를 사용하여 새 테이블을 작성해야 합니다.

WebSphere Commerce 응용프로그램 아키텍처

사용자 정의와 관련된 여러 가지 소프트웨어 구성요소들의 작동 방법을 살펴보았습니다. 이제 응용프로그램 아키텍처에 대해 살펴보고자 합니다. 이는 기초 계층이 되는 부분과 수정 가능한 부분을 이해하는 데 도움이 됩니다. 다음 도표는 응용프로그램 아키텍처를 형성하는 여러 계층을 보여줍니다.

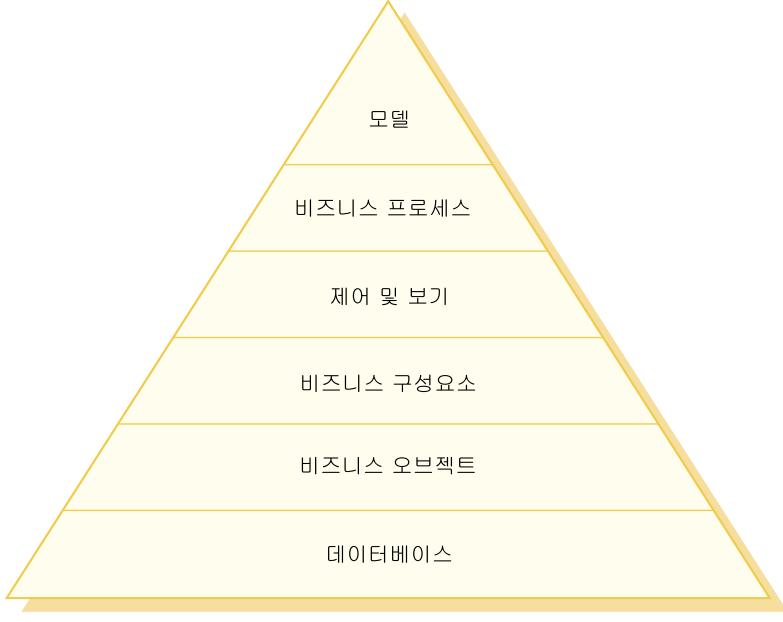


그림 2.

응용프로그램 아키텍처의 각 계층에 대한 설명은 다음과 같습니다.

데이터베이스

WebSphere Commerce는 전자 상거래 응용프로그램과 데이터 요구사항에 맞게 특별히 설계된 데이터베이스 스키마를 사용합니다. 다음은 이 스키마로 된 테이블 예입니다.

- 사용자
- 주문
- 상품

비즈니스 오브젝트

비즈니스 오브젝트는 상거래 도메인 내의 엔티티를 나타내고 데이터베이스 내에 들어 있는 정보를 가져오거나 해석하는 데 필요한 데이터 중심 로직을 캡슐화합니다. 이 엔티티는 Enterprise JavaBeans 스펙을 따릅니다.

이 엔티티 bean은 상거래 응용프로그램과 데이터베이스 사이의 인터페이스로 작동합니다. 또한 엔티티 bean은 각 데이터베이스의 테이블 열 간의 복잡한 관계보다 이해하기 쉽습니다.

비즈니스 구성요소

비즈니스 구성요소는 비즈니스 로직의 단위입니다. 비즈니스 구성요소는 각 단위로 나뉘어진 프로시저 비즈니스 로직을 수행합니다. 로직은 제어기 명령 및 태스크 명령의 WebSphere Commerce 모델을 사용하여 구현됩니다. 유형 구성요소의 예로는 OrderProcess 제어기 명령이 있습니다. 이 명령은 일반 주문을 처리하는 데 필요한 모든 비즈니스 로직을 캡슐화합니다. 전자 상거래 응용프로그램은 OrderProcess 명령을 호출하고, 이 명령은 차례대로 여러 태스크 명령을 호출하여 각 작업 단위를 수행합니다. 예를 들어, 각 태스크 명령은 주문 요구사항을 충족시킬 만큼 충분한 재고량이 있는지 확인, 지불을 처리 및 주문 상태를 갱신하고 처리가 완료되면 적절한 분량만큼 재고량을 감소시킵니다.

제어 및 보기

웹 제어기는 사용할 해당 제어기 명령 구현 및 보기를 판별합니다. 구현은 상점에 따라 다릅니다.

보기는 명령 및 사용자 조치의 결과를 표시합니다. 이 보기는 JSP 템플리트를 사용하여 구현됩니다. 보기의 예로는 ProductDisplay(구매자가 선택

한 상품의 관련 정보를 보여주는 상품 페이지 리턴) 및 OrderPrepare(구매자에게 해당 주문 정보를 제출하기 위한 양식 제시)가 있습니다.

비즈니스 프로세스

비즈니스 구성요소와 보기가 한데 모여 비즈니스 프로세스라고 하는 워크플로우 및 사이트플로우를 작성합니다. 비즈니스 프로세스의 예는 다음과 같습니다.

사용자 등록

이 비즈니스 프로세스에는 사용자 등록 프로세스를 구성하는 모든 단계와 관련된 비즈니스 구성요소(예: 새 사용자에 대한 등록 레코드를 작성하는 UserRegistrationAdd 명령) 및 보기가 있습니다.

카탈로그 탐색

이 비즈니스 프로세스에는 카탈로그 탐색 프로세스를 구성하는 모든 단계와 관련된 비즈니스 구성요소(예: 상점 카탈로그와 카탈로그 내의 카테고리를 보여주는 StoreCatalogDisplay 및 CategoryDisplay 명령) 및 보기가 있습니다.

모델 전체적으로 볼 때 도표의 하위 계층은 전자 상거래 비즈니스 모델을 구성합니다. 전자 상거래 비즈니스 모델의 한 예로는 InFashion 견본 상점에서 사용되는 B2C 모델이 있습니다. 다른 예로는 ToolTech 견본 상점에서 사용되는 B2B 모델이 있습니다.

WebSphere Commerce 런타임 아키텍처

앞 절에서는 응용프로그램 아키텍처를 소개하면서 비즈니스 응용프로그램 관점에서 WebSphere Commerce 응용프로그램의 여러 층에 대해 설명했습니다. 이 절에서는 런타임 아키텍처를 구현하는 방법에 대해 설명합니다.

WebSphere Commerce 런타임 아키텍처의 주요 구성요소는 다음과 같습니다.

- Servlet 엔진
- 프로토콜 리스너
- 어댑터 관리자
- 어댑터

- 웹 제어기
- 명령
- 엔티티 bean
- 데이터 bean
- 데이터 bean 관리자
- 표시 페이지
- XML 파일

WebSphere Commerce 구성요소 간의 상호 작용은 다음과 같은 도표로 표시됩니다. 다음 절에서는 각각의 구성요소에 대해 보다 자세히 설명합니다.

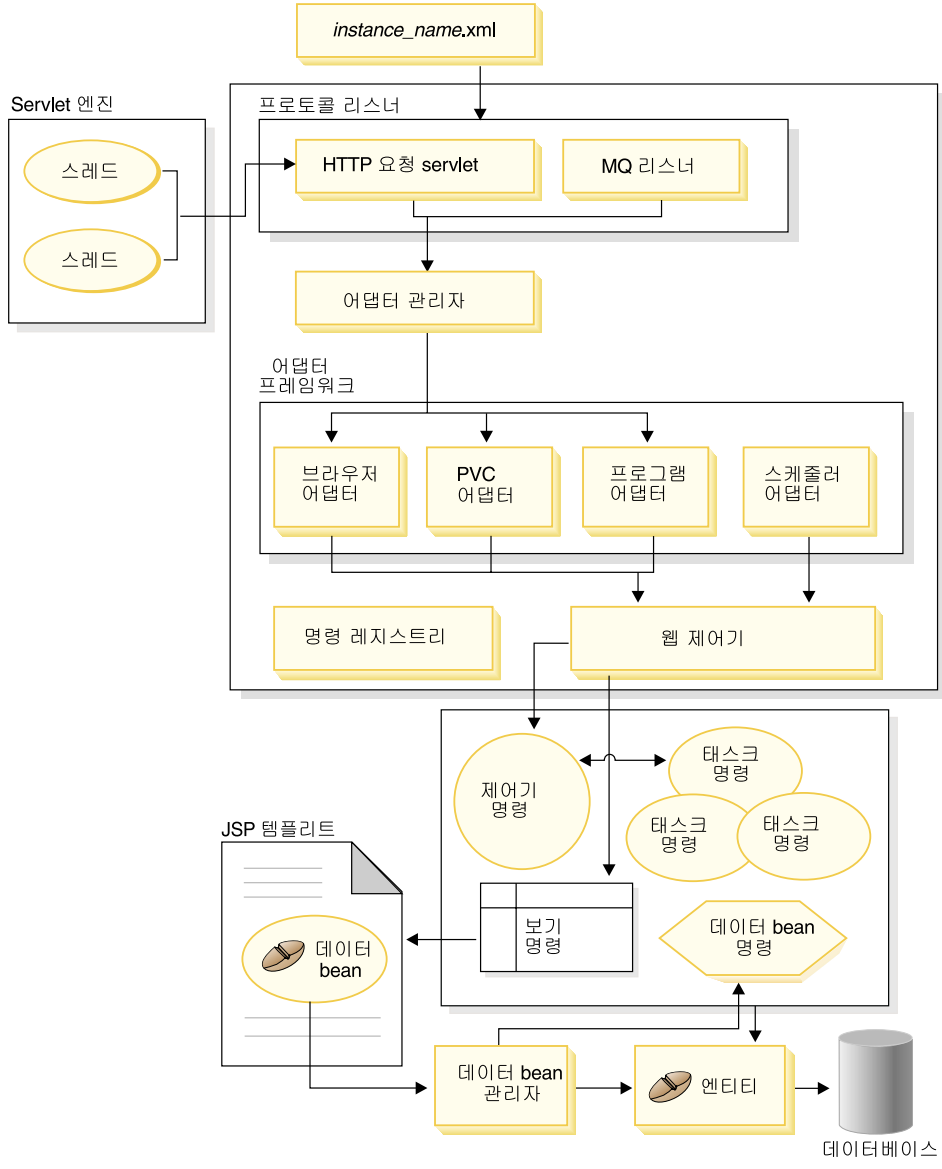

instance_name.xml
Servlet 엔진
스레드
스레드
프로토콜 리스너
HTTP 요청 servlet
MQ 리스너
어댑터 관리자
어댑터
프레임워크
브라우저
어댑터
PVC
어댑터
프로그램
어댑터
스케줄러
어댑터
명령 레지스트리
웹 제어기
제어기
명령
태스크
명령
태스크
명령
태스크
명령
JSP 템플리트
데이터
bean
보기
명령
데이터 bean
명령
데이터 bean
관리자
엔티티
데이터베이스

그림 3.

Servlet 엔진

Servlet 엔진은 인바운드 URL 요청의 요청 디스패처 역할을 하는 WebSphere Application Server 런타임 환경의 일부입니다. Servlet 엔진은 요청을 처리하기 위해 스레드 풀을 관리합니다. 각 인바운드 요청은 개별 스레드로 실행됩니다.

이전 버전의 WebSphere Commerce는 사전에 할당된 시스템 프로세스 세트를 유지하여 URL 요청용 자체 태스크 디스패처를 구현한 C++ 응용프로그램 서버를 사용하였습니다. 시스템 처리를 사용하는 이전 모델은 Java 스레드를 사용하는 새 모델보다 자원이 더 많이 소요됩니다. 새 WebSphere Commerce 런타임 아키텍처는 Servlet 엔진을 사용하여 보다 조정성이 뛰어난 상거래 솔루션을 제공합니다.

어댑터 관리자

어댑터 관리자는 어떤 어댑터가 요청을 핸들할 것인지를 결정한 후 요청을 해당 어댑터로 전달할 수 있는지 판별합니다.

프로토콜 리스너

WebSphere Commerce 명령은 다양한 장치에서 호출될 수 있습니다. 명령을 호출할 수 있는 장치의 예는 다음과 같습니다.

- 일반 인터넷 브라우저
- 인터넷 브라우저를 지원하는 이동 전화
- MQSeries를 사용하여 XML 메시지를 보내는 B2B 응용프로그램
- HTTP에서 XML을 사용하여 요청을 전송하는 조달 시스템
- 백그라운드 작업을 실행하는 WebSphere Commerce 스케줄러

장치는 다양한 통신 프로토콜을 사용합니다. 프로토콜 리스너는 전송으로부터 인바운드 요청을 수신한 후 사용된 프로토콜에 따라 해당 어댑터로 요청을 지정하는 런타임 구성요소입니다. 프로토콜 리스너에는 다음이 포함됩니다.

- 요청 Servlet
- MQSeries 리스너

요청 Servlet이 Servlet 엔진에서 URL 요청을 수신하면 어댑터 관리자에게 요청을 전달합니다. 그러면 어댑터 관리자는 요청을 처리할 수 있는 어댑터를 결정하기 위해 어댑터 유형을 조회합니다. 일단 특정 어댑터가 결정되면 요청은 해당 어댑터로 전달됩니다.

요청 Servlet이 초기화되면 *instance_name*.xml 구성 파일을 읽습니다. XML 파일의 구성 블록 중 하나에 모든 어댑터가 정의됩니다. 요청 Servlet의 init() 메소드는 정의된 모든 어댑터를 초기화합니다.

MQSeries® 리스너는 원격 프로그램에서 XML 기반 MQSeries 메시지를 받아 비 HTTP 어댑터 관리자로 메시지를 지정합니다.

작업 스케줄러에는 프로토콜 리스너가 필요하지 않습니다.

어댑터

WebSphere Commerce 어댑터는 웹 제어기로 요청을 전달하기 전에 처리 기능을 수행하는 장치별 구성요소입니다. 어댑터에서 수행되는 처리 태스크의 예는 다음과 같습니다.

- 웹 제어기에 장치 유형에 맞는 방식으로 요청을 처리하도록 지시. 예를 들어 PvC(Pervasive Computing) 장치 어댑터는 웹 제어기에 원래 요청한 HTTPS 검사를 무시하도록 지시할 수 있습니다.
- WebSphere Commerce 명령이 구문 분석할 수 있는 특성 세트로 인바운드 요청의 메시지 포맷 변환
- 장치 고유의 세션 지속성 제공

다음 도표는 WebSphere Commerce 어댑터 프레임워크의 구현 클래스 계층을 보여줍니다

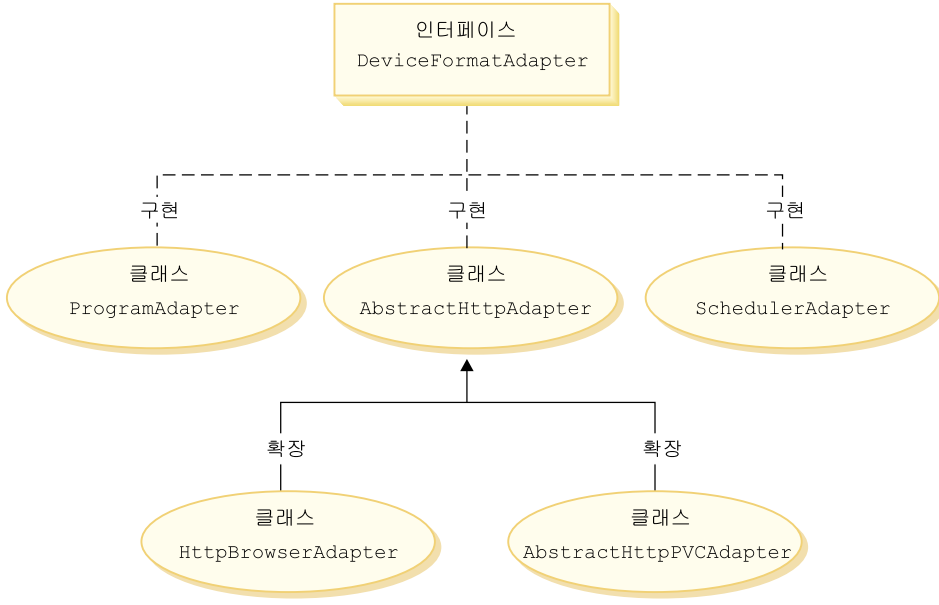


그림 4.

위의 도표에 표시된 대로 모든 어댑터는 DeviceFormatAdapter 인터페이스를 구현합니다. 다음은 WebSphere Commerce 런타임 환경에서 사용되는 어댑터입니다.

프로그램 어댑터

프로그램 어댑터는 WebSphere Commerce 명령을 호출하여 원격 프로그램을 지원합니다. 프로그램 어댑터는 요청을 받고 메시지 맵퍼를 사용하여 요청을 CommandProperty 오브젝트로 변환합니다. 변환 후에, 프로그램 어댑터는 CommandProperty 오브젝트를 사용하고 요청을 실행합니다.

스케줄러 어댑터

스케줄러 어댑터는 백그라운드 작업으로 실행되는 WebSphere Commerce 명령을 지원합니다

HTTP 브라우저 어댑터

HTTP 브라우저 어댑터는 HTTP 브라우저에서 수신되는 WebSphere Commerce 명령을 호출하는 요청을 지원합니다.

HTTP PvC 어댑터

특정 PvC 장치 어댑터를 개발하는 데 사용할 수 있는 추상 어댑터 클래스입니다. 예를 들어 특정 휴대폰 응용프로그램용 어댑터를 개발해야 하는 경우, 이 어댑터에서 확장합니다.

필요한 경우, 어댑터 프레임워크는 다음의 두 가지 방법으로 확장될 수 있습니다.

- 특정 PvC 장치용 어댑터를 작성하십시오(예를 들어 i 모드 장치를 지원하는 HttpIModePVCAdapterImpl 클래스를 작성하십시오). 이러한 유형의 어댑터는 AbstractHttpAdapterImpl 클래스를 확장해야 합니다.
- 새 프로토콜 리스너에 연결되는 새 어댑터를 작성하십시오. 새 어댑터는 DeviceFormatAdapter 인터페이스를 구현해야 합니다.

웹 제어기

WebSphere Commerce 웹 제어기는 EJB 컨테이너의 설계 패턴과 유사한 설계 패턴을 따르는 응용프로그램 컨테이너입니다. 이 컨테이너는 세션 관리(어댑터에서 설정한 세션 지속성에 따라), 트랜잭션 제어, 액세스 제어 및 인증과 같은 서비스를 제공하여 명령의 역할을 단순화합니다.

또한 웹 제어기는 상거래 응용프로그램의 프로그래밍 모델을 실행하는 역할을 수행합니다. 예를 들어 프로그래밍 모델은 응용프로그램이 작성할 수 있는 명령의 유형을 정의합니다. 각 유형의 명령은 특정 목적을 처리합니다. 비즈니스 로직은 제어기 명령에서 구현되어야 하며, 보기 로직은 보기 명령에서 구현되어야 합니다. 웹 제어기에서는 제어기 명령이 보기 이름을 리턴할 것으로 예상합니다. 보기 이름이 리턴되지 않으면 예외가 발생합니다.

HTTP 요청의 경우, 웹 제어기는 다음과 같은 태스크를 수행합니다.

- javax.transaction 패키지에서 UserTransaction 인터페이스를 사용하여 트랜잭션을 시작합니다.
- 어댑터에서 세션 데이터를 가져옵니다.
- 명령을 호출하기 전에 사용자가 로그온되어야 할지 여부를 판별합니다. 필요한 경우, 사용자의 브라우저를 로그온 URL로 경로 재지정합니다.
- 보안 HTTPS가 URL에 필요한지 확인합니다. 필요하지만 현재 요청이 HTTPS를 사용하지 않는 경우 웹 브라우저를 HTTPS URL로 경로 재지정하십시오.

- 제어기 명령을 호출하고 명령 컨텍스트 및 입력 특성 오브젝트를 전달합니다.
- 트랜잭션 롤백 예외가 발생하고 제어기 명령이 다시 시도될 수 있는 경우, 제어기 명령을 재시도합니다.
- 클라이언트로 돌려보낼 보기 명령이 있으면 일반적으로 제어기 명령은 보기 이름을 리턴합니다. 웹 제어기는 해당 보기의 보기 명령을 호출합니다. 응답 보기를 작성하는 방법은 다양합니다. 여기에는 다른 URL로 재지정, JSP 템플리트로 전달 또는 응답 오브젝트에 HTML 문서 작성과 같은 방법이 있습니다.
- 세션 데이터를 저장합니다.
- 세션 데이터를 확약합니다.
- 성공한 경우 현재 트랜잭션을 확약합니다.
- 실패한 경우 현재 트랜잭션을 롤백합니다(환경에 따라).

명령

WebSphere Commerce 명령은 특정 요청의 처리와 관련된 프로그래밍 로직이 들어 있는 bean입니다.

WebSphere Commerce 명령에는 다음 네 가지 기본 유형이 있습니다.

제어기 명령

제어기 명령은 특정 비즈니스 프로세스에 관련된 로직을 캡슐화합니다. 제어기 명령의 예로는 주문 프로세스를 위한 *OrderProcessCmd* 명령과 새로 등록된 사용자 작성을 위한 *UserRegistrationAddCmd* 명령이 있습니다. 일반적으로 제어기 명령은 제어 명령문(예: if, then, else)을 포함하며 비즈니스 프로세스에서 각각의 태스크를 수행할 태스크 명령을 호출합니다. 완료시, 제어기 명령은 보기 이름을 리턴합니다. 그런 후 웹 제어기는 보기 명령을 위한 해당 구현 클래스를 판별한 후 보기 명령을 실행합니다.

태스크 명령

태스크 명령은 특정 응용프로그램 로직 단위를 구현합니다. 일반적으로 제어기 명령과 한 세트의 태스크 명령이 함께 URL 요청에 대한 응용프로그램 로직을 구현합니다. 태스크 명령은 제어기 명령과 같은 컨테이너에서 실행됩니다.

데이터 bean 명령

데이터 bean 명령은 데이터 bean의 인스턴스가 생성될 때 JSP 템플리트에서 호출됩니다. 데이터 bean 명령의 기본 기능은 데이터를 가진 데이터 bean에 대량 자료 반입하는 것입니다.

보기 명령

보기 명령은 클라이언트 요청에 대한 응답으로 보기를 구성합니다. 보기 명령에는 다음 세 가지 유형이 있습니다.

경로 재지정 보기 명령

이 보기 명령은 URL 경로 재지정과 같은 경로 재지정 프로토콜을 사용하여 보기를 전송합니다. 제어기 명령은 경로 재지정 프로토콜을 사용하여 보기를 리턴하기 위해 이러한 보기 유형의 보기 명령을 리턴해야 합니다. 경로 재지정 프로토콜이 사용될 때는 브라우저에서 URL 스택을 변경합니다. 다시 로드 키가 입력되면 경로 재지정된 URL은 원래 URL 대신 실행됩니다.

직접 보기 명령

이 보기 명령은 응답 보기를 직접 클라이언트에 전송합니다.

전달 보기 명령

이 보기 명령은 보기 요청을 JSP 템플리트와 같은 다른 웹 구성 요소로 전달합니다.

보기 명령이 호출될 수 있는 세 가지 방식은 다음과 같습니다.

- 요청이 완료되면 제어기 명령은 보기 명령 이름을 지정합니다.
- 클라이언트에서는 보기를 직접 요청합니다.
- 명령에서는 오류를 검출하며 오류를 처리하기 위해 오류 태스크가 실행되어야 합니다. 명령은 보기 명령 이름 관련 예외를 발생시킵니다. 예외가 웹 제어기로 전달되면 해당 명령은 오류 보기 명령을 실행하고 클라이언트로 응답을 돌려보냅니다.

WebSphere Commerce 엔티티 bean

엔티티 bean은 WebSphere Commerce가 제공하는 지속적인 트랜잭션 상거래 오브젝트입니다. 상거래 도메인에 익숙한 경우, 엔티티 bean은 즉각적으로 WebSphere

Commerce 데이터를 나타냅니다. 즉, 데이터베이스 스키마를 이해하지 않고도 상거래 도메인에서 개념과 오브젝트를 보다 근접하게 모델링하는 엔티티 bean의 데이터에 액세스할 수 있습니다. 기존의 엔티티 bean을 확장할 수 있습니다. 또한 고유 응용프로그램별 비즈니스 요구사항에 맞추어 완전한 새 엔티티 bean을 전개할 수 있습니다.

엔티티 bean은 EJB 구성요소 모델에 따라 구현됩니다.

엔티티 bean에 대한 자세한 내용은 53 페이지의 『WebSphere Commerce 엔티티 bean의 구현』을 참조하십시오.

데이터 bean

데이터 bean은 주로 웹 디자이너가 사용하는 Java bean입니다. 주로 이들은 WebSphere Commerce 엔티티에 대한 액세스를 제공합니다. 웹 디자이너는 이들 bean을 JSP 템플리트에 전개하여 표시할 때 페이지에 동적 정보로 대량 자료 반입 되도록 할 수 있습니다. 웹 디자이너는 bean이 제공할 수 있는 데이터와 bean이 입력으로 요구하는 데이터를 이해하기만 하면 됩니다. 비즈니스 로직에서 표시가 분리되므로 웹 디자이너가 bean 작동 방법을 알 필요는 없습니다.

데이터 bean 관리자

WebSphere Commerce 데이터 bean이 WebSphere Studio Page Designer를 사용하여 JSP 템플리트로 삽입될 때, 런타임시 데이터 bean 관리자를 호출하여 데이터 bean에 대량 자료 반입하는 한 행의 코드를 생성합니다.

다음은 Page Designer의 견본 코드입니다.

```
am.ibm.commerce.beans.DataBeanManager.activate(data_bean, request)
```

JavaServer Pages 템플리트

JSP 템플리트는 특별히 표시용으로 사용되는 특수한 Servlet입니다. URL 요청이 완료되면 웹 제어기는 JSP 템플리트를 호출하는 보기 명령을 호출합니다. 또한 클라이언트는 관련 명령없이 브라우저에서 직접 JSP 템플리트를 호출할 수 있습니다. 이 경우 JSP 템플리트에서 필요로 하는 모든 데이터 bean이 하나의 트랜잭션 내에서 활성화될 수 있도록 JSP 템플리트의 URL은 그 경로에 요청 servlet을 포

함시켜야 합니다. 요청 servlet은 URL 요청을 JSP 템플리트로 전달하여 하나의 트랜잭션 내에서 JSP 템플리트를 실행할 수 있습니다.

데이터 bean 관리자는 그 경로에 요청 servlet을 포함하지 않는 JSP 템플리트의 URL을 거부합니다. JSP 템플리트 및 기타 자원의 보호에 관한 자세한 내용은 101 페이지의 제 4 장『액세스 제어』를 참조하십시오.

Instance_name.xml 구성 파일

Instance_name.xml 구성 파일은 인스턴스의 구성 정보를 설정합니다. 요청 servlet이 초기화될 때 파일을 읽습니다.

요청 정보 요약

이 절은 요청에 대한 응답으로 발생하는 구성요소 간의 상호 작용 플로우의 정보를 요약합니다.

다음 도표는 각 단계에 대한 설명입니다.

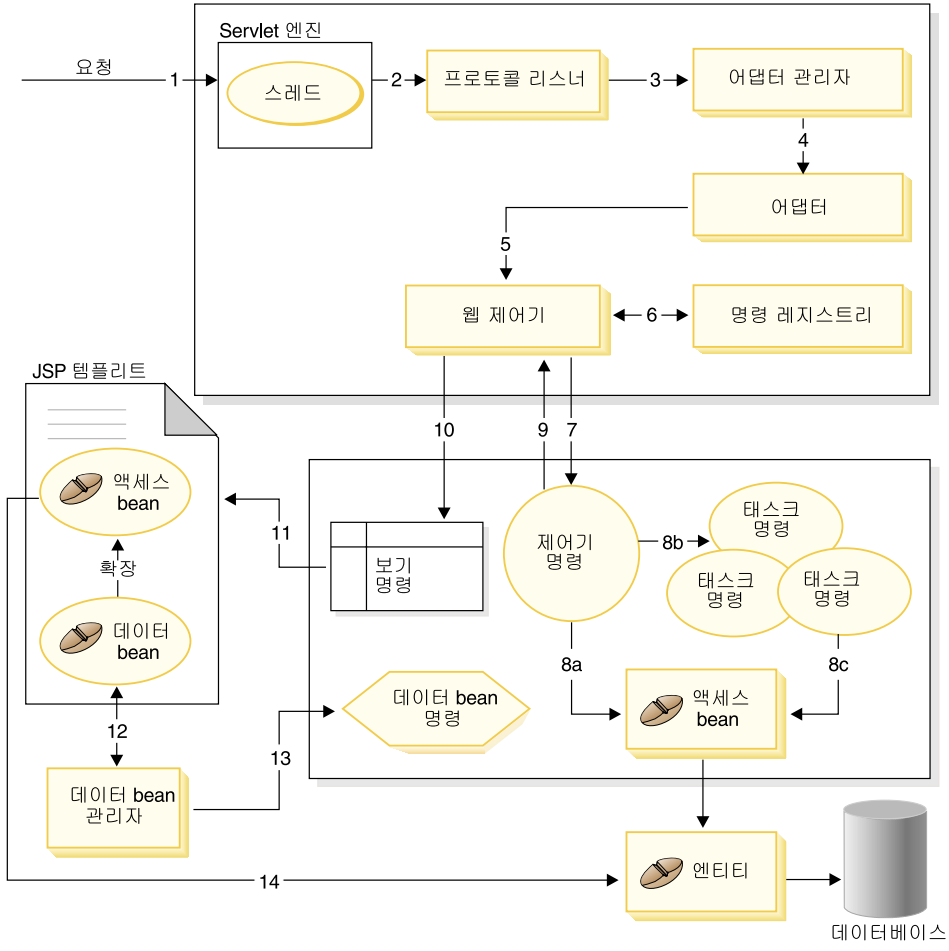


그림 5.

다음 정보는 위의 도표에 해당합니다.

1. 요청은 WebSphere Application Server 플러그인에 의해 Servlet 엔진으로 지정됩니다.
2. 요청이 자체 스레드에서 실행됩니다. servlet 엔진은 프로토콜 리스너로 요청을 지정합니다. 프로토콜 리스너는 HTTP 요청 Servlet이거나 MQ 리스너입니다.
3. 프로토콜 리스너가 요청을 어댑터 관리자에게 전달합니다.

4. 어댑터 관리자는 어떤 어댑터가 요청을 핸들한 다음 해당 어댑터로 요청을 전달할 수 있는지 판별합니다. 예를 들어 인터넷 브라우저에 요청이 제공되면 어댑터 관리자는 요청을 HTTP 브라우저 어댑터로 전달합니다.
5. 어댑터는 요청을 웹 제어기로 전달합니다.
6. 웹 제어기는 명령 레지스트리를 조회하여 호출할 명령을 판별합니다.
7. 요청의 제어기 명령 사용을 전제로 하면 웹 제어기는 해당 제어기 명령을 호출합니다.
8. 제어기 명령이 실행되면, 다음과 같은 몇 가지 경로가 사용 가능해집니다.
   a. 제어기 명령은 액세스 bean과 해당 엔티티 bean을 사용하여 데이터베이스에 액세스할 수 있습니다.
   b. 제어기 명령은 하나 이상의 태스크 명령을 호출할 수 있습니다. 그런 다음 태스크 명령은 액세스 bean 및 해당 엔티티 bean(8(c) 참조)을 사용하여 데이터베이스에 액세스할 수 있습니다.
9. 완료시, 제어기 명령은 웹 제어기로 보기 이름을 리턴합니다.
10. 웹 제어기는 VIEWREG 테이블에서 보기 이름을 찾고 요청자의 장치 유형에 대해 등록된 보기 명령 구현을 호출합니다.
11. 보기 명령은 요청을 JSP 템플리트로 전달합니다.
12. JSP 템플리트 내에서 데이터 bean은 데이터베이스로부터 동적 정보를 검색해야 합니다. 데이터 bean 관리자는 데이터 bean을 활성화합니다.
13. 데이터 bean 관리자는 필요한 경우 데이터 bean 명령을 호출합니다.
14. 데이터 bean이 확장된 액세스 bean은 해당 엔티티 bean을 사용하여 데이터베이스에 액세스합니다.

이전 릴리스의 사용자 정의와의 주요 차이점

WebSphere Commerce Suite 버전 5.1부터 WebSphere Commerce Server를 Java로 완전하게 작성했습니다. 따라서 기능을 사용자 정의하려면 Java를 사용해야 합니다. 이는 사용자 정의에 C++ 및 Net.Data® 매크로를 사용한 WebSphere Commerce Suite, 버전 4.1(및 Net.Commerce™ 이전 버전)에서 사용한 모델과는 다릅니다.

다음 표는 주요 변경사항이 요약되어 있습니다.

	C++로 작성된 버전 4.1(이전 버전)	**Java로 작성된 버전 5.1(및 이후 버전)**
응용프로그램 모델	명령	제어기 명령
	태스크	태스크 명령
	대체 가능한 함수	태스크 명령
	보기 태스크	보기 명령
	오류 태스크	보기 명령
표시 모델	Net.Data 매크로	JSP 템플리트, 데이터 bean, 데이터 bean 명령 및 보기 명령
지속 모델	Net.Data 및 ODBC	엔티티 bean
URL 디스패처	WebSphere Commerce C++ URL 디스패처	WebSphere servlet 엔진
태스크 모델	시스템 처리	Java 스레드
명령 어댑터	없음	웹 제어기 및 어댑터

이 책에서는 이주 프로세스에 대해서는 다루지 않습니다. 이주에 대한 자세한 내용은 *WebSphere Commerce 이주 안내서*를 참조하십시오.

제 2 부 프로그래밍 모델

제 2 장 설계 패턴

WebSphere Commerce 프레임워크를 개발하는 데에는 다양한 설계 패턴 및 메커니즘이 사용됩니다. WebSphere Commerce에는 각 WebSphere Commerce 응용프로그램에서 따라야 하는 고급 설계 패턴을 제공합니다. 이 장에서는 다음 설계 패턴에 대해 설명합니다.

- model-view-controller 설계 패턴
- 명령 설계 패턴
- 표시 설계 패턴

model-view-controller 설계 패턴

MVC(model-view-controller) 설계 패턴은 응용프로그램이 데이터 모델, 표시 정보 및 제어 정보로 구성되도록 지정합니다. 이 패턴에서는 이들 각각이 다른 오브젝트로 분리되어야 합니다.

모델(예: 데이터 정보)에는 순수한 응용프로그램 데이터만 들어 있으며 사용자에게 데이터 표시 방법을 설명하는 로직은 없습니다.

보기(예: 표시 정보)는 사용자에게 모델 데이터를 표시합니다. 보기를 통해서 모델 데이터에 액세스할 방법을 알 수 있지만 이 데이터의 의미나 사용자가 이를 조작하기 위해 수행할 수 있는 작업은 알 수 없습니다.

마지막으로 *제어기*(예: 제어 정보)는 보기와 모델 사이에 존재합니다. 제어기는 보기(또는 또다른 외부 소스)에 의해 트리거된 이벤트를 청취하고 이러한 이벤트에 대한 해당 반응을 실행합니다. 대부분의 경우, 반응은 모델에서 메소드를 호출하는 것입니다. 보기와 모델이 알림 메커니즘을 통해 연결되므로 이 조치의 결과는 보기에 자동으로 반영됩니다.

오늘날 대부분의 응용프로그램은 이러한 패턴을 따르며 조금씩 변형됩니다. 예를 들어 일부 응용프로그램에서는 보기와 제어기가 이미 매우 단단하게 결합되었으므

로 하나의 클래스로 결합됩니다. 모든 변형은 데이터와 해당 표시를 분리합니다. 이는 응용프로그램의 구조를 더 간단히 만들 뿐만 아니라 코드를 다시 사용할 수 있게 합니다.

견본과 패턴에 대해 참조할 만한 서적이 많으므로 이 문서에서는 패턴을 아주 자세하게 설명하고 있지는 않습니다.

다음 도표는 MVC 설계 패턴이 WebSphere Commerce에 적용되는 방법을 보여줍니다.

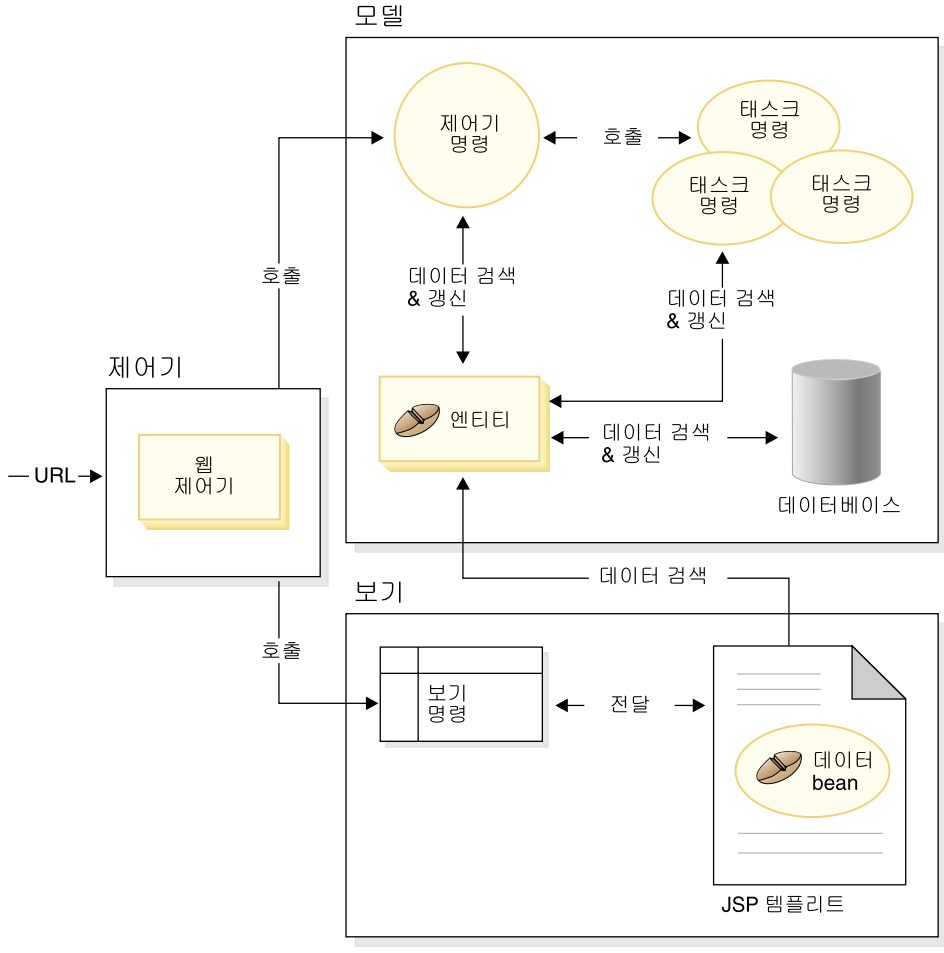


그림 6.

명령 설계 패턴

WebSphere Commerce Server는 다른 원격 응용프로그램에서는 물론 브라우저 기반 thin 클라이언트 응용프로그램의 요청도 승인합니다. 예를 들어 요청은 원격 조달 시스템이나 다른 commerce 서버로부터 제공될 수 있습니다.

다양한 포맷으로 된 모든 요청은 어댑터 프레임워크를 구성하는 어댑터에 의해 일반 포맷으로 변환됩니다. 한번 요청이 일반 포맷으로 되면 요청들은 WebSphere Commerce 명령으로 이해될 수 있습니다.

명령은 비즈니스 로직을 수행하는 bean입니다. 이들 명령은 고급 처리 로직 또는 분리 비즈니스 로직 태스크 형태로 프로시저 로직을 나타냅니다. 처리 기반 명령은 여러 엔티티 및 기타 명령을 연결하는 제어기의 역할을 하는 반면, 태스크 명령은 특정 태스크를 수행하고 하나의 오브젝트에만 액세스할 수 있습니다.

명령 프레임워크

명령 bean은 특정 설계 패턴을 따릅니다. 각 명령은 인터페이스 클래스(예: `CategoryDisplayCmd`)와 구현 클래스(예: `CategoryDisplayCmdImpl`)를 모두 포함하고 있습니다. 호출자의 관점에서 보면 호출 로직에는 입력 특성 설정, `execute()` 메소드 호출 및 출력 특성 검색이 포함됩니다.

명령 구현의 관점에서 보면 명령은 호출자와 구현 사이의 간접 레벨을 허용하는 표준 명령 설계 패턴을 구현하는 WebSphere 명령 프레임워크를 따릅니다. 이 간접 레벨 내에서 작동 가능한 주요 메커니즘은 다음과 같습니다.

1. 사용자가 명령을 호출하도록 허용되었는지 여부를 판별하는 액세스 제어 정책 관리자를 호출하는 기능
2. 상점 식별자에 따라 상점마다 서로 다른 명령 구현을 실행할 수 있는 기능
3. 요청자의 신용 카드 유형에 근거하여 여러 가지 보기 구현을 실행하는 기능

다음 도표는 네 가지 기본 명령 유형에 대한 인터페이스의 개념적인 개요를 보여줍니다.

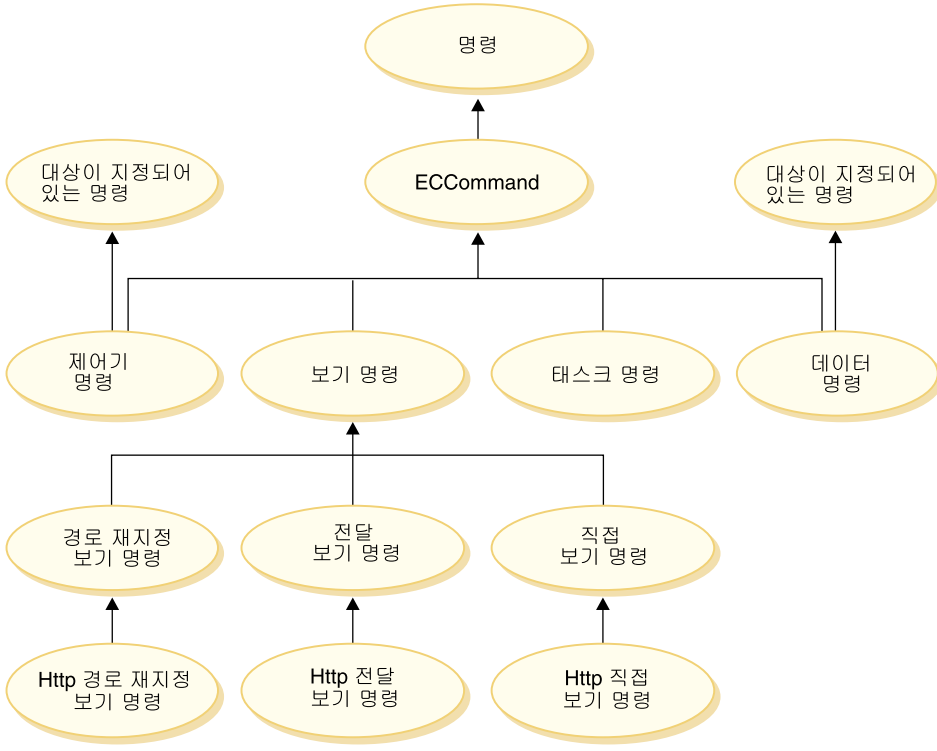


그림 7.

제어기 명령

제어기 명령은 특정 비즈니스 처리에 관련된 로직을 캡슐화합니다. 제어기 명령의 예로는 주문 처리를 위한 *OrderProcessCmd* 명령과 새로 등록된 사용자 작성을 위한 *UserRegistrationAddCmd* 명령이 있습니다. 일반적으로 제어기 명령은 제어 명령문(예: if, then, else)을 포함하며 비즈니스 처리에서 각각의 태스크를 수행할 태스크 명령을 호출합니다. 완료시, 제어기 명령은 보기 이름을 리턴합니다. 웹 제어기는 보기 태스크를 위한 해당 구현 클래스를 판별한 후 보기 태스크를 실행합니다.

제어기 명령은 대상을 지정할 수 있는 명령이긴 하지만 로컬의 대상 지정만 지원됩니다.

태스크 명령

태스크 명령은 특정 응용프로그램 로직 단위를 구현합니다. 일반적으로 제어기 명령과 한 세트의 태스크 명령이 함께 URL 요청에 대한 응용프로그램 로직을 구현합니다. 작업 명령은 제어기 명령과 같은 컨테이너에서 실행됩니다.

데이터 bean 명령

데이터 bean 명령은 데이터 bean의 인스턴스가 생성될 때 JSP 페이지에서 호출됩니다. 데이터 bean 명령의 기본 기능은 데이터 bean의 필드를 대량 자료 반입하는 것입니다.

데이터 bean 명령은 대상을 지정할 수 있는 명령이긴 하지만 로컬의 대상 지정만 지원됩니다.

보기 명령

보기 명령은 클라이언트 요청에 대한 응답으로 보기를 구성합니다. 보기 명령이 호출될 수 있는 세 가지 방식은 다음과 같습니다.

- 요청이 완료되면 제어기 명령은 보기 명령 이름을 지정합니다.
- 클라이언트에서는 보기를 직접 요청할 수 있습니다.
- 제어기 또는 태스크 명령은 오류를 검출 및 처리하기 위해 오류 태스크를 실행해야 하는지 결정하고 보기 명령 이름을 가진 예외를 발생시킵니다. 예외가 웹 제어기로 전달되면 보기 명령을 실행하고 클라이언트로 응답을 돌려보냅니다.

보기 명령에는 다음 세 가지 유형이 있습니다.

경로 재지정 보기 명령

이 보기 명령은 URL 경로 재지정과 같은 경로 재지정 프로토콜을 사용하여 보기를 전송합니다. 경로 재지정 프로토콜이 필요할 때 제어기 명령은 이 보기 유형의 보기 명령을 리턴해야 합니다. 경로 재지정 프로토콜이 사용될 때는 브라우저에서 URL 스택을 변경합니다. 다시 로드 키가 입력되면 경로 재지정된 URL은 원래 URL 대신에 실행됩니다.

직접 보기 명령

이 보기 명령은 응답 보기를 직접 클라이언트에 전송합니다.

전달 보기 명령

이 보기 명령은 보기 요청을 JSP 템플리트와 같은 다른 웹 구성 요소로 전달합니다.

명령 팩토리

새 명령 오브젝트를 작성하기 위해 명령 호출자는 *명령 팩토리*를 사용합니다. 명령 팩토리는 명령 인스턴스를 작성하는 데 사용되는 bean입니다. 명령 팩토리는 팩토리 설계 패턴에 따라 달라집니다. 이 설계 패턴은 호출 클래스가 아닌 구현 클래스(인스턴스 생성)의 정보가 있는 팩토리 클래스에서 오브젝트 인스턴스를 작성하도록 합니다.

팩토리는 새 오브젝트의 인스턴스를 작성하기 위한 수준 높은 방법을 제공합니다. 이 경우, 명령 팩토리는 각각의 상점에 따라 새 명령 오브젝트를 작성할 때 올바른 구현 클래스를 판별하는 방법을 제공합니다. 명령 인터페이스 이름 및 특정 상점 식별자는 인스턴스가 작성될 때 새 명령 오브젝트로 전달됩니다.

명령의 구현 클래스를 지정하는 데는 두 가지 방법이 있습니다.
`defaultCommandClassName` 변수를 사용하여 명령 인터페이스용 코드에 기본 구현 클래스를 직접 지정할 수 있습니다. 예를 들어 다음 코드는 CategoryDisplayCmd 인터페이스에 존재합니다.

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

구현 클래스를 지정하는 두 번째 방법은 WebSphere Commerce 명령 레지스트리를 사용하는 것입니다. 명령 레지스트리는 구현 클래스가 한 상점에서 다른 상점으로 변환할 때 항상 사용해야 합니다. 명령 레지스트리에 대한 자세한 내용은 31 페이지에 있습니다.

기본 구현 클래스가 인터페이스의 코드에 지정되고 다른 구현 클래스가 명령 레지스트리에 지정된 경우에 명령 레지스트리의 우선순위가 높습니다.

명령 팩토리 사용 구문은 다음과 같습니다.

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

여기서 *interfaceName*은 새 명령 bean의 인터페이스 이름이고 getStoreId 메소드는 명령이 사용될 상점을 결정합니다

주: 명령 팩토리를 사용하여 비즈니스 정책 명령을 작성하는 구문은 앞의 코드 부분과 다릅니다. 명령 팩토리를 사용한 비즈니스 정책 명령 작성에 대한 자세한 내용은 199 페이지의 『새 비즈니스 정책 호출』을 참조하십시오.

명령 플로우

이 절에서는 명령과 WebSphere Commerce 데이터베이스 사이의 논리 플로우 개요를 제공합니다. 다음 도표 및 설명은 이러한 플로우를 보여줍니다.

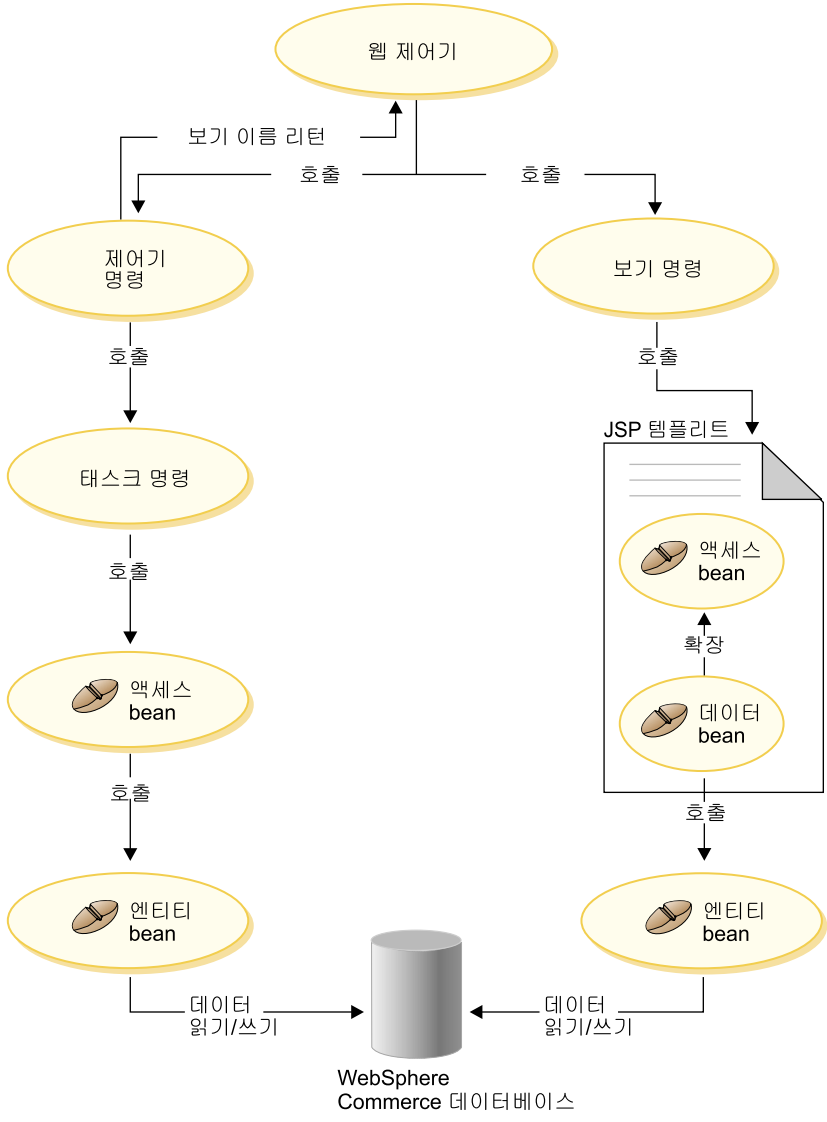


그림 8.

웹 제어기가 요청을 수신하면 요청 시 제어기 명령의 호출 여부를 결정합니다. 어떤 경우에도 웹 제어기는 명령의 구현 클래스를 결정한 후 해당 클래스를 호출합니다.

먼저 도표의 왼쪽 부분을 살펴보십시오. 제어기 명령이 비즈니스 처리의 로직을 캡슐화하므로 비즈니스 처리에서 특정 작업 단위를 수행하기 위해 각각의 작업 명령

을 자주 호출합니다. 액세스 bean은 데이터베이스의 정보가 검색되거나 갱신되어야 할 때 호출됩니다. 작업 명령 또는 제어기 명령은 액세스 bean을 호출할 수 있습니다. 그런 다음 요청은 액세스 bean에서 WebSphere Commerce 데이터베이스를 읽고 쓸 수 있는 엔티티 bean으로 이동합니다

이제 도표의 오른쪽 부분을 살펴보십시오. 보기 명령은 제어기 명령이 처리를 완료하고 호출할 보기 명령 이름을 리턴한 경우나 오류가 발생하여 오류 보기가 표시되어야 하는 경우에 웹 제어기에서 호출됩니다.

보기 명령은 일반적으로 클라이언트에 대한 응답을 표시하기 위해 JSP 템플리트를 호출합니다. JSP 템플리트 내에서 데이터 bean은 페이지에 동적 정보를 대량 자료 반입하는 데 사용됩니다. 데이터 bean 관리자가 데이터 bean을 활성화합니다. 데이터 bean(액세스 bean에서 확장)은 해당 엔티티 bean을 호출합니다. JSP 템플리트에서 간접적으로 액세스할 때 엔티티 bean은 일반적으로 데이터베이스에서 정보를 검색합니다(데이터베이스에 정보를 기록하지 않음).

명령 등록 프레임워크

WebSphere Commerce 제어기 및 태스크 명령은 명령 레지스트리에 등록됩니다. 다음 세 테이블은 명령 레지스트리를 구성합니다.

- URLREG
- CMDREG
- VIEWREG

주: Business 이 절은 비즈니스 정책 명령 등록에는 적용되지 않습니다. 새 비즈니스 정책 명령 등록에 대한 자세한 내용은 180 페이지의 『새 비즈니스 정책 및 비즈니스 정책 명령 등록』을 참조하십시오.

URLREG 테이블

URLREG 테이블은 URI(Universal Resource Indicator)를 제어기 명령 인터페이스에 맵핑합니다. URI는 자원 ID에 대해 간단하고 확장 가능한 메커니즘을 제공합니다. URI는 추상적 또는 물리적 자원을 식별하는 데 사용되는 상대적으로 간단한 문자열입니다. WebSphere Commerce에서 URI에는 명령 정보만 포함됩니다. 다음 URL에서 URI 부분은 굵은체로 표시됩니다.

```
http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=store_Id&catalogId=catalog_Id&langId=-1
```

URI와 인터페이스 이름이 일대일로 대응하는 경우, 각 상점에서는 HTTPS 또는 AUTHENTICATION이 명령에 필요한지 지정할 수 있습니다. 각 인바운드 URL 요청에서 웹 제어기는 제어기 명령의 인터페이스 이름을 찾아본 후, 해당 이름을 사용하여 CMDREG 테이블에 등록된 대로 올바른 구현 클래스를 판별합니다.

다음 표는 URLREG 데이터베이스 테이블에 포함된 정보에 대해 설명합니다.

열 이름	설명	주석
URL	URI 이름	예: MyNewCommand 또는 ProductDisplay
STOREENT_ID	상점 엔티티 식별자	0으로 설정하여 모든 상점의 명령을 사용하거나 고유 상점 식별자로 설정하여 명령이 특정 상점에만 사용되도록 표시할 수 있습니다.
INTERFACENAME	제어기 명령 인터페이스 이름	예: com.ibm.commerce.catalog.commands.GetProductDisplay.TemplateCmd
HTTPS	URL 요청에 보안 HTTP가 필요함	HTTPS가 필수인 경우에는 1을 사용하고 필수가 아닌 경우에는 0을 사용하십시오.
DESCRIPTION	URI 설명	예: This command is used for testing purposes.
AUTHENTICATED	이 URL 요청에 사용자 로그온이 필요함	인증이 필수인 경우에는 1을 사용하고 필수가 아닌 경우에는 0을 사용하십시오.
INTERNAL	명령이 WebSphere Commerce의 내부 명령인지 여부 표시	내부 명령이면 1을 사용하고 외부 명령이면 0을 사용하십시오.

웹 제어기가 URL 요청을 수신하면 요청된 제어기 명령의 인터페이스 이름을 검색하고 이 이름을 사용하여 CMDREG 테이블에서 구현 클래스 이름을 찾아봅니다. 또한 URLREG 테이블에서 HTTPS 열을 확인하여 HTTPS가 URL 요청에 필요한지 판별하기도 합니다.

URL 요청에 의해 호출되는 명령만이 URLREG 테이블에 등록되어야 합니다. 따라서 태스크 또는 보기 명령이 아닌 제어기 명령만 여기에 등록되어야 합니다.

다음 SQL 문은 특정 상점(상점 식별자가 5임)에서 사용되는 MyNewControllerCommand의 항목을 작성합니다.

```
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)
```

insert 문의 일반 구문은 다음과 같습니다.

```
insert into table_name (column_name1,column_name2, ... ,column_namen)
values (column1_value,column2_value,...,columnn_value)
```

문자열 값은 작은따옴표로 묶어야 합니다.

CMDREG 테이블

CMDREG는 명령 등록 테이블입니다. 이 테이블은 명령 인터페이스를 해당 구현 클래스에 맵핑하기 위한 메커니즘을 제공합니다. 상점별로 명령을 사용자 정의할 경우, 인터페이스의 복수 구현이 허용됩니다.

제어기 명령 및 작업 명령만 CMDREG 테이블에 등록됩니다. 보기 명령은 VIEWREG 테이블에 등록됩니다.

다음은 CMDREG 데이터베이스 테이블에 포함된 정보에 대해 설명합니다.

열 이름	설명	주석
STOREENT_ID	상점 엔티티 식별자	0으로 설정하여 모든 상점의 명령을 사용하거나 고유 상점 식별자로 설정하여 명령이 특정 상점에만 사용되도록 표시할 수 있습니다.
INTERFACENAME	명령 인터페이스 이름	인터페이스를 정의하며 URLREG 테이블에서 사용한 것과 동일한 이름을 사용합니다.
DESCRIPTION	해당 명령에 대한 설명	예: This command is used for testing purposes.
CLASSNAME	명령 구현 클래스 이름	일반적으로, 끝에 "Impl"이 추가된 인터페이스 이름
PROPERTIES	명령에 입력 특성으로 설정된 기본 이름-값 쌍	포맷은 URL 조회 문자열과 같습니다(예: "parm1=val1&parm2=val2").

열 이름	설명	주석
LASTUPDATE	이 명령 항목이 마지막으로 갱신된 날짜	
TARGET	명령 대상 이름. 명령이 실제로 실행되는 위치입니다.	로컬 대상이 지원됩니다.

일반적으로 새 제어기 또는 태스크 명령을 작성하면 CMDREG 테이블에 해당 항목을 작성해야 합니다. 예를 들어 다음 SQL 문은 특정 상점(상점 식별자가 5임)에서 사용되는 MyNewCommand의 항목을 작성합니다.

```
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')
```

insert 문의 일반 구문은 다음과 같습니다.

```
 insert into table_name (column_name1,column_name2, ... ,column_namen)
values (column1_value,column2_value,...,columnn_value)
```

문자열 값은 작은따옴표로 묶어야 합니다.

작성 중인 명령이 항상 동일한 구현 클래스를 사용할 경우, CMDREG 테이블에 명령을 등록해야 할 필요는 없습니다. 이 경우, 인터페이스에 defaultCommandClassName 속성을 사용하여 구현 클래스를 지정할 수 있습니다. 예를 들면 인터페이스 코드에서 다음을 사용합니다.

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

이 방법으로 구현 클래스를 지정하면 구현 클래스에 기본 특성을 전달할 수 없으며 모든 상점에 동일한 구현 클래스가 사용되어야 합니다.

등록된 제어기 명령의 예

사이트에 StoreA 및 StoreB의 두 상점이 있는 시나리오를 고려해 보십시오. 상점마다 명령 구현 방식이 다르며 MyUrl 제어기 명령에 대한 보안 요구사항도 다릅니다. 이 절에서는 다음과 같이 사용자 정의할 수 있도록 명령 레지스트리 사용 방법을 보여줍니다.

다음 표는 URLREG 테이블에 있는 StoreA 및 StoreB의 항목을 보여줍니다.

열 이름	**StoreA**의 항목	**StoreB**의 항목
URL	MyUrl	MyUrl
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce. mycommands.myUrl	com.ibm.commerce. mycommands. myUrl
HTTPS	1	1
DESCRIPTION	URLREG 테이블의 항목 예	URLREG 테이블의 항목 예
AUTHENTICATED	1	0
INTERNAL	널(NULL)값	널(NULL)값

주: INTERFACENAME의 값에 있는 공백은 표시용입니다. 각각의 값은 실제로 하나의 연속 문자열입니다.

URLREG 테이블의 항목에 따라 웹 제어기는 MyURL URI의 인터페이스 이름이 com.ibm.commerce.mycommands.MyUrl임을 판별합니다. 또한 StoreA에서는 명령이 HTTPS 및 인증을 모두 사용하여 실행되어야 하나 StoreB에서는 HTTPS만 필요합니다. HTTPS 및 인증에 대한 값은 인터페이스가 아니라 웹 제어기에서 사용됩니다.

다음 도표는 이러한 플로우를 보여줍니다.

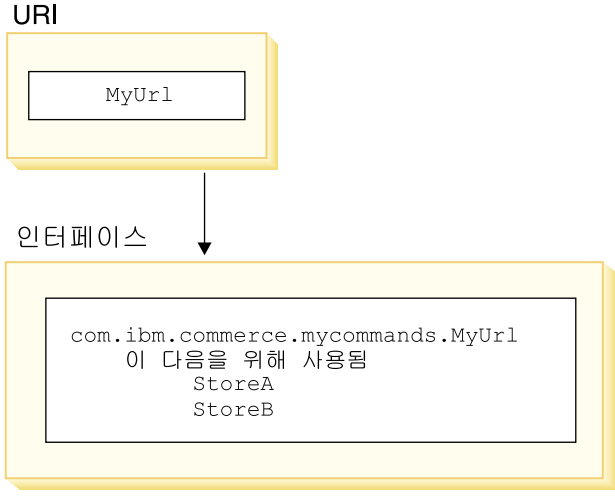


그림 9.

다음 표는 CMDREG 테이블의 항목을 보여줍니다. 이 예에 사용되는 열만 표시됩니다.

열 이름	**StoreA**의 항목	**StoreB**의 항목
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

주: INTERFACENAME 및 CLASSNAME의 값에 있는 공백은 표시용입니다. 각각의 값은 실제로 하나의 연속 문자열입니다.

CMDREG 테이블의 항목에 따라 웹 제어기는 StoreA의 경우 com.ibm.commerce.mycommands.MyUrl 인터페이스 구현 클래스가 com.ibm.commerce.mycommands.MyUrlStoreAImpl임을 판별합니다. 또한 StoreB의 경우 동일한 인터페이스의 구현 클래스가 com.ibm.commerce.mycommands.MyUrlStoreBImpl임을 판별합니다. 다음 도표는 이러한 플로우를 보여줍니다.

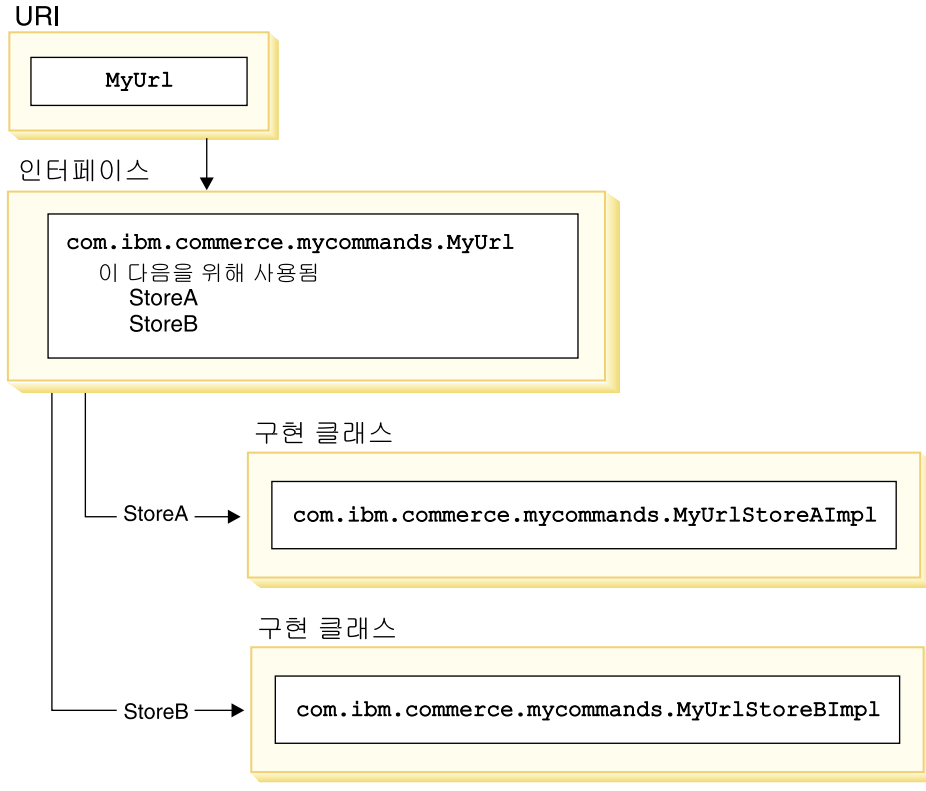


그림 10.

VIEWREG 테이블

VIEWREG 테이블은 장치 고유의 보기 명령 구현을 등록할 수 있도록 합니다. 이 테이블을 사용하여 여러 가지 보기 구현을 등록합니다. 그러면 명령 프레임워크가 다양한 클라이언트에 다른 보기를 리턴할 수 있습니다.

보기 명령 이름이 제어기 명령에서 리턴되거나 예외에 지정되면 웹 제어기가 VIEWREG 테이블의 보기 명령 클래스를 판별합니다. 여러 개의 보기 명령 이름이 동일한 구현 클래스에 맵핑될 수 있습니다.

열 이름	설명	주석
VIEWNAME	보기 이름	예: `AddressForm`
DEVICEFMT_ID	장치 유형 식별자	사용 가능한 옵션은 다음과 같습니다. • BROWSER(기본값) • I_MODE • E-mail • MQXML • MQNC
STOREENT_ID	상점 엔티티 식별자	`0`으로 설정하여 모든 상점의 명령을 사용하거나 고유 상점 식별자로 설정하여 명령이 특정 상점에만 사용되도록 표시할 수 있습니다.
INTERFACENAME	보기 명령 인터페이스 이름	기본 옵션은 ForwardView, DirectView 및 RedirectView입니다.
CLASSNAME	보기 명령 구현 클래스 이름	기본 구현을 사용할 수 있습니다.
PROPERTIES	명령에 입력 특성으로 설정된 기본 이름-값 쌍	동일한 페이지가 항상 표시되면 이 특성에 JSP 파일 이름을 설정하십시오(`docname=`*`jsp_name`*`.jsp`). 동일한 JSP 템플리트가 모든 상점에 사용되면 상점 고유의 디렉토리가 사용되지 않도록 `storeDir=no`로 설정하십시오. 일반 사용자가 명령을 호출할 수 있는 경우, `isGeneric=true`를 설정하십시오.
DESCRIPTION	해당 명령에 대한 설명	
HTTPS	URL 요청에 보안 HTTP가 필요함	HTTPS가 필수인 경우에는 1을 사용하고 필수가 아닌 경우에는 0을 사용하십시오.
LASTUPDATE	항목이 마지막으로 갱신된 날짜	
INTERNAL	명령이 WebSphere Commerce의 내부 명령인지 여부 표시	내부 명령이면 1을 사용하고 외부 명령이면 0을 사용하십시오.

새 보기 명령을 작성할 때 VIEWREG 테이블에 해당 항목을 작성해야 합니다. 다음 조건 중 하나가 충족되면 보기 명령은 VIEWREG 테이블에 등록되어야 합니다.

- 보기 명령이 액세스 제어하에 실행됩니다.
- 여러 개의 보기 명령 구현이 있습니다.
- 특성이 PROPERTIES 열에 설정되어 있습니다.

등록된 보기 명령은 보기 이름을 사용한 명령 레지스트리를 통해 액세스되거나 실제 표시 파일 이름을 사용하여 직접 액세스될 수 있습니다. VIEWREG 테이블에 등록되지 않은 보기는 클라이언트에서 실제 표시 파일 이름을 사용할 경우에만 액세스될 수 있습니다.

다음과 같이 VIEWREG 항목이 있는, *MyView* 보기를 고려해 보십시오.

열 이름	항목
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

MyView가 등록된 보기이므로 클라이언트는 명령 이름을 사용하거나 실제 표시 파일 이름을 명령 이름으로 대체하여 보기에 액세스할 수 있습니다. 보기 이름을 사용하는 견본 URL은 다음과 같습니다.

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

파일 이름을 사용하는 견본 URL은 다음과 같습니다.

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

클라이언트에서 등록된 보기를 직접 호출할 가능성이 있으면(표시 파일 이름을 사용) 이 예(MyView 및 MyView.jsp)에 표시된 대로 실제 표시 파일 이름과 동일한 보기의 이름을 사용하여 명령을 등록해야 합니다.

테이블에 등록되지 않은 보기는 표시 파일 이름을 사용해야 호출될 수 있습니다. 따라서 MyUnregisteredView.jsp 파일을 사용하는 미등록 보기가 있으면 보기에 액세스할 URL은 다음과 같습니다.

```
http://hostname.com/webapp/wcs/stores/servlet/MyUnregisteredView.jsp
```

다음 SQL 명령문 예는 한 특정 상점에서 사용하는 *MyNewViewCommand* 항목을 작성합니다.

```
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)
```

다음 표는 주요 정보가 있는 또다른 견본 VIEWREG 테이블을 제공합니다.

COMMAND - NAME	DEVICE - FMT_ID	INTERFACE - NAME	CLASSNAME	PROPERTIES
ProductDisplayView	BROWSER	전달 보기 명령	HttpForwardView CommandImpl	
InterestItemAddView	BROWSER	경로 재지정 보기 명령	HttpRedirectView CommandImpl	docname =item.jsp
InterestItemDeleteView	BROWSER	경로 재지정 보기 명령	HttpRedirectView CommandImpl	docname =item.jsp
GenericApplication 오류	BROWSER	경로 재지정 보기 명령	HttpRedirectView CommandImpl	docname =usererr. jsp
GenericSystemError	BROWSER	경로 재지정 보기 명령	HttpRedirectView CommandImpl	docname =syserr. jsp
Logon	BROWSER	전달 보기 명령	HttpForwardView CommandImpl	docname= logon.jsp & storeDir=no

주: COMMANDNAME, INTERFACENAME, CLASSNAME 및 PROPERTIES의 값에 있는 모든 공백은 표시용입니다. 각각의 값은 실제로 하나의 연속 문자열입니다. 열 이름의 하이픈 또한 표시용입니다.

앞에 나온 테이블은 다음과 같은 시나리오를 보여줍니다.

- *ProductDisplayView* 보기 이름은 제어기 명령(ProductDisplay)에서 웹 제어기로 리턴됩니다. 웹 제어기는 ProductDisplayView 보기 명령 이름과 해당 장치 식별자를 사용하여 보기 명령 인터페이스 및 클래스 이름을 판별합니다. 보기 명령은 상점 및 장치 식별자마다 서로 다른 구현 클래스를 가질 수 있습니다. 그러나 인터페이스 이름은 보기 명령 유형을 정의하므로 동일해야 합니다.
- InterestItemAdd 및 InterestItemDelete 명령은 각각 *InterestItemAddView* 및 *InterestItemDeleteView* 보기 이름을 웹 제어기로 리턴합니다. 두 명령 모두 경로 재지정 보기를 필요로 하므로 두 보기 모두의 보기 명령 인터페이스 이름은 RedirectViewCommand입니다. 두 보기 모두에 공통 JSP 템플리트가 등록되어 있습니다. 웹 제어기는 특성(docname=item.jsp)을 가져와 보기 명령(HttpRedirectViewCommandImpl)에 전달합니다.
- 제어기 명령 또는 태스크 명령에서 잘못된 사용자 매개변수에 대한 ECApplication 예외를 발생시킬 경우, 다음이 발생할 수 있습니다.
  - 제어기 명령(응용프로그램 예외의 경우 호출)내에 지정된 보기가 있는 경우, 해당 보기 항목이 VIEWREG 테이블에서 검색되어 이에 따라 처리됩니다.
  - 보기가 지정되어 있지 않은 경우, GenericApplicationError 명령이 호출되며 데이터베이스에 등록된 JSP 템플리트가 표시됩니다. 예를 들어 앞에 나온 테이블을 사용하면 `usererr.jsp` 템플리트가 표시됩니다.
- 제어기 명령 또는 작업 명령이 시스템 예외에 대한 ECSystem 예외를 발생시키는 경우, 다음과 같이 발생할 수 있습니다.
  - 제어기 명령(시스템 예외의 경우 호출) 내에 지정된 보기가 있는 경우, 해당 보기의 항목이 VIEWREG 테이블에서 검색되어 이에 따라 처리됩니다.
  - 보기가 지정되어 있지 않은 경우, GenericSystemError 명령이 호출되며 데이터베이스에 등록된 JSP 템플리트가 표시됩니다. 예를 들어 앞에 나온 테이블을 사용하면 `syserr.jsp` 템플리트가 표시됩니다.

- 브라우저 클라이언트는 로그온 URL을 입력하여 로그온 페이지를 호출할 수 있습니다. storeDir 특성이 “no”로 설정되어 있으므로 상점 고유 정보가 JSP 템플리트의 경로에 포함되어 있지 않습니다. 그러므로 모든 상점의 고객에게 동일한 로그온 페이지가 표시됩니다.

표시 설계 패턴

표시 페이지는 클라이언트에 응답을 리턴합니다. 일반적으로 표시 페이지는 JSP 템플리트로 구현할 수 있지만(권장되는 메소드) Servlet으로 직접 작성될 수 있습니다.

여러 장치 유형을 지원하기 위해 보기 명령에 액세스하는 URL은 실제 JSP 파일의 이름이 아니라 보기 이름을 사용해야 합니다.

이러한 간접 레벨 이면의 기본 원리는 JSP 템플리트가 보기를 표시하는 것입니다. 적절한 보기를 선택할 수 있는 기능은(예: 요청 컨텍스트의 로케일, 장치 유형 또는 기타 데이터 기준) 매우 바람직합니다(특히 하나의 요청에는 종종 여러 개의 사용 가능한 보기가 있을 경우). 예를 들어, 두 구매자가 상점 홈페이지 요청합니다(한 구매자는 일반 웹 브라우저 사용, 다른 구매자는 휴대폰 사용). 분명 동일한 홈페이지가 각 구매자에게 표시되어서는 안됩니다. 요청을 승인한 다음, 명령 등록 프레임워크의 정보에 따라 각 구매자가 수신하는 보기를 판별하는 것은 웹 제어기의 작업입니다.

JSP 템플리트 및 데이터 bean

데이터 bean은 동적 컨텐츠를 제공하기 위해 JSP 템플리트 내에서 사용되는 Java bean입니다. 일반적으로 데이터 bean은 WebSphere Commerce 엔티티 bean을 간단히 나타냅니다. 데이터 bean은 엔티티 bean 내에서 검색되거나 설정될 수 있는 특성을 캡슐화합니다. 이와 같이 데이터 bean은 JSP 템플리트로 동적 데이터를 통합하는 작업을 단순화합니다.

데이터 bean에는 표시 페이지에 사용될 수 있는 특성을 정의하는 *BeanInfo* 클래스가 있습니다. 또한 BeanInfo 클래스는 지원되는 모든 WebSphere Commerce 언어로 특성 이름을 제공함으로써 다국어 지원 사이트에서 데이터 bean을 사용할 수 있도록 합니다.

데이터 bean은 다음과 같은 호출로 활성화됩니다.

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

WebSphere Studio Page Designer는 WebSphere Commerce 데이터 bean이 JSP 템플리트로 삽입될 때 자동으로 코드 앞에 행을 생성합니다.

상점 개발자는 JSP 템플리트 개발시 상점 및 다국어 지원 작동 문제의 특성을 고려해야 합니다. 다국어 지원 작동에 대한 자세한 내용은 WebSphere Commerce 온라인 도움말을 참조하십시오.

JSP 템플리트 및 데이터 bean 보안 고려사항

JSP 템플리트 및 데이터 bean을 사용하는 특수한 코드 방식을 통해 권한이 없는 사용자가 데이터베이스에 액세스할 가능성을 최소할 수 있습니다. SQL 문의 insert, select, update 및 delete 부분은 개발시 작성되어야 합니다. 런타임 입력 정보를 수집하려면 매개변수 삽입을 사용하십시오.

매개변수 삽입을 사용한 런타임 입력 정보 수집의 예는 다음과 같습니다.

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

반면 입력 문자열을 사용하여 SQL 문을 작성할 수는 없습니다. 입력 문자열을 사용하는 예는 다음과 같습니다.

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

데이터 bean 유형

데이터 bean은 JSP 템플리트에서 동적 데이터를 제공하는 데 주로 사용되는 Java bean입니다. 데이터 bean에는 두 가지 유형인 스마트 데이터 bean 및 명령 데이터 bean이 있습니다.

스마트 데이터 bean에서는 *lazy fetch* 메소드를 사용하여 고유의 데이터를 검색합니다. 이 데이터 bean 유형은 필요한 만큼만 데이터를 검색하므로 액세스 bean의 모든 데이터가 필요하지 않은 상황에서 더 나은 성능을 제공할 수 있습니다. 데이터베이스에 액세스해야 하는 스마트 데이터 bean은 해당 엔티티 bean의 액세스 bean에서 확장되어 com.ibm.commerce.SmartDataBean 인터페이스를 구현해야 합니다. 예를 들어 ProductData 데이터 bean은 상품 엔티티 bean에 해당하는 ProductAccessBean 액세스 bean을 확장합니다.

일부 스마트 데이터 bean의 경우 데이터베이스에는 액세스할 필요가 없습니다. 예를 들어 PropertyResource 스마트 데이터 bean은 데이터베이스가 아니라 자원 번들에서 데이터를 검색합니다. 데이터베이스에 액세스할 필요가 없는 경우 스마트 데이터 bean은 SmartDataBeanImpl 클래스를 확장해야 합니다.

명령 데이터 bean은 작은 데이터 bean으로 해당 데이터를 검색합니다. 명령은 JSP 템플리트에 필요한지 여부에 상관없이 한번에 모든 데이터 bean 속성을 검색합니다. 결과적으로 데이터 bean에서 선택된 속성만 사용하는 JSP 템플리트의 경우 명령 데이터 bean의 속도가 느려질 수 있습니다. 대부분의 속성 또는 모든 속성이 필요한 JSP 템플리트의 경우 명령 데이터 bean은 매우 편리합니다.

명령 데이터 bean은 해당 액세스 bean에서 확장되어 com.ibm.commerce.CommandDataBean 인터페이스를 구현할 수도 있습니다.

데이터 bean 인터페이스

데이터 bean은 다음 Java 인터페이스 중 하나 또는 모두를 구현합니다.

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

각 Java 인터페이스는 데이터 bean에 대량 자료 반입되는 데이터 소스에 대해 설명합니다. 여러 인터페이스를 구현하면 데이터 bean은 다양한 소스의 데이터에 액세스할 수 있습니다. 각 인터페이스에 대한 추가 정보는 아래에 있습니다.

SmartDataBean 인터페이스: SmartDataBean 인터페이스를 구현하는 데이터 bean은 연관된 데이터 bean 명령없이 고유의 데이터를 검색할 수 있습니다. 스마트 데이터 bean은 보통 해당 엔티티 bean의 액세스 bean에서 확장합니다. 스마트 데이터 bean이 활성화되면 데이터 bean 관리자는 데이터 bean의 populate 메소드를 호출합니다. 데이터 bean은 populate 메소드를 사용하여 연관된 오브젝트의 속성을 제외하고 모든 속성을 검색할 수 있습니다. 예를 들어 데이터 bean이 엔티티 bean의 액세스 bean 클래스에서 확장할 경우, 데이터 bean은 refreshCopyHelper 메소드를 호출합니다. 해당 엔티티 bean의 모든 속성은 스마트 데이터 bean에 자

동으로 대량 자료 반입됩니다. 그러나 엔티티 bean에 연관된 오브젝트가 있는 경우, 해당 오브젝트의 속성은 검색되지 않습니다. 스마트 데이터 bean을 사용할 때의 기본적인 이점은 다음과 같습니다.

- 구현은 간단하며 데이터 bean 명령을 작성할 필요가 없습니다.
- 새 필드가 엔티티 bean에 추가되면 데이터 bean을 변경할 필요가 없습니다. 엔티티 bean이 수정되면 액세스 bean은 다시 생성되어야 합니다(VisualAge for Java의 도구를 사용). 액세스 bean이 다시 생성되자마자 자동으로 스마트 데이터 bean에 모든 새 속성을 사용할 수 있습니다.
- 엔티티 bean에는 종종 연관된 오브젝트를 나타내는 속성이 포함됩니다. 성능상의 이유로 인해 스마트 데이터 bean은 자동으로 이러한 속성을 검색하지 않습니다. 대신 다음 도표에 표시된 대로 필요할 때까지 이러한 속성 검색을 지연하는 것이 좋습니다.

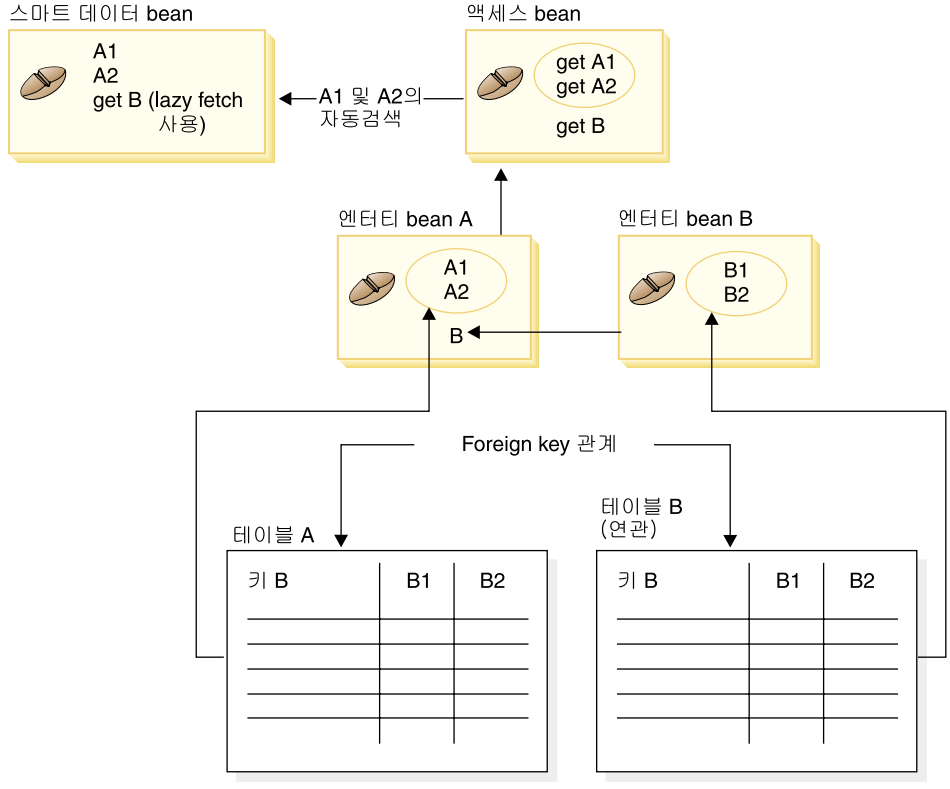


그림 11.

lazy fetch 검색 구현에 대한 자세한 내용은 48 페이지의 『Lazy fetch 데이터 검색』을 참조하십시오.

CommandDataBean 인터페이스: CommandDataBean 인터페이스를 구현하는 데이터 bean은 데이터 bean 명령에서 데이터를 검색합니다. 이 유형의 데이터 bean은 작은 오브젝트이며 데이터 bean 명령을 기초로 하여 해당 데이터에 대량 자료 반입합니다. 데이터 bean은 데이터 bean 명령의 인터페이스 이름을 리턴하는 getCommandInterfaceName() 메소드(com.ibm.commerce.CommandDataBean 인터페이스에서 정의한 대로)를 구현해야 합니다.

InputDataBean 인터페이스: InputDataBean 인터페이스를 구현하는 데이터 bean은 보기 명령에 의해 설정된 URL 매개변수 또는 속성으로부터 데이터를 검색합니다.

이 인터페이스에 정의된 속성은 추가 데이터를 가져오기 위해 1차 키 필드로 사용될 수 있습니다. JSP 템플리트가 호출되면 생성된 JSP Servlet 코드는 URL 매개변수와 일치하는 모든 속성에 대량 자료 반입한 후 데이터 bean 관리자에게 데이터 bean을 전달하여 데이터 bean을 활성화합니다. 그런 후 데이터 bean 관리자는 데이터 bean의 setRequestProperties() 메소드(com.ibm.commerce.InputDataBean 인터페이스에서 정의한 대로)를 호출하여 보기 명령에 의해 설정된 모든 속성을 전달합니다. WebSphere Studio가 Page Designer를 사용하여 페이지로 삽입되는 데이터 bean마다 다음 코드를 생성해야 합니다.

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

BeanInfo 클래스

데이터 bean은 java.lang.Object.BeanInfo 인터페이스를 구현하는 BeanInfo 클래스 없이 완성되지 않습니다. BeanInfo 클래스는 데이터 bean의 메소드 및 특성에 대한 명시적인 정보를 제공하는 데 사용됩니다. 웹 디자이너의 데이터 bean 구현 클래스에서 공용 런타임 메소드를 숨기거나 데이터 bean의 속성 각각에 대해 해당 표시 문자열을 설정하기 위해 해당 클래스가 사용될 수 있습니다.

BeanInfo 클래스에 대한 자세한 내용은 Sun Microsystems의 JavaBeans 스펙을 참조하십시오.

데이터 bean 활성화

데이터 bean은 com.ibm.commerce.beans.DataBeanManager 클래스에 있는 activate 또는 silentActivate 메소드를 사용하여 활성화될 수 있습니다. activate 메소드는 모든 속성을 사용할 수 있는 경우에만 활성화 이벤트가 완료되는 전체 활성화 메소드입니다. 하나의 속성만 사용할 수 없는 경우에도 전체 활성화 처리에 대해 예외가 발생합니다.

silentActivate 메소드에서는 개별 속성을 사용할 수 없을 때 예외가 발생하지 않습니다.

JSP 템플리트 내에서 제어기 명령 호출

JSP 템플리트를 사용하여 제어기 명령을 호출하는 것은 표시와 로직을 구분하는 것과 일관되지는 않지만 이렇게 해야 하는 상황이 있을 수도 있습니다. 그러한 경우 ControllerCommandInvokerDataBean을 사용할 수 있습니다.

이 데이터 bean을 사용하면 호출할 명령의 인터페이스 이름을 지정하거나, 호출할 명령의 이름을 직접 설정할 수 있습니다. 그 명령에 대해 요청 특성을 설정할 수도 있습니다.

이 데이터 bean이 데이터 bean 관리자에 의해 활성화될 경우, 제어기 명령이 실행되어 JSP 템플리트에 대해 응답 특성을 사용할 수 있습니다.

제어기 명령이 실행되면 보기를 실행할 수 있습니다.

Lazy fetch 데이터 검색

데이터 bean이 활성화되면 데이터 bean 명령 또는 데이터 bean의 populate() 메소드에 의해 대량 자료 반입될 수 있습니다. 검색된 속성은 데이터 bean의 해당 엔티티 bean에 있습니다. 또한 엔티티 bean에는 그 자체가 다수의 속성을 갖는 연관된 오브젝트가 있을 수도 있습니다.

활성화시, 연관된 모든 오브젝트의 속성이 자동으로 검색되면 성능 문제점이 발생할 수 있습니다. 성능은 연관된 오브젝트의 수가 증가함에 따라 저하될 수 있습니다.

다수의 비슷한 모델 소개, 나은 모델 소개 또는 부속 상품(연관된 오브젝트)이 들어 있는 상품 데이터 bean을 고려해 보십시오. 상품 데이터 bean이 활성화되자마자 연관된 모든 오브젝트에 대량 자료 반입할 수 있습니다. 그러나 이러한 방식으로 대량 자료 반입할 경우, 여러 번의 데이터베이스 조회가 필요할 수 있습니다. 페이지에서 모든 속성이 다 필요하지는 않은 경우, 여러 번의 데이터베이스 조회는 비능률적일 수 있습니다.

일반적으로 한 페이지에 모든 속성이 필요한 것은 아니므로 더 나은 설계 패턴은 아래에 표시된 대로 lazy fetch를 수행하는 것입니다.

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

JSP 속성 설정 - 개요

WebSphere Commerce 프로그램 모델은 MVC 설계 패턴을 한 차원 높인 것입니다. 이와 같이 URL 요청 결과 표시는 제어기 명령 및 작업 명령과는 분리됩니다. 이러한 명령은 장치와는 관계 없습니다. 이러한 명령은 클라이언트에 대한 정보없이 비즈니스 로직을 구현하고 클라이언트로 리턴될 데이터를 작성합니다. 반면 보기 명령은 장치에 따라 고유합니다.

제어기 및 작업 명령이 직접 보기를 구성하지는 않지만, 보기로 정보를 전달합니다. 정보가 보기로 전달되는 방법을 이해하는 것이 중요합니다. 다음 도표는 웹 제어기, 명령 레지스트리, 제어기 명령 및 보기 명령 간에 특성이 전달되는 방법을 보여줍니다.

CMREG

INTERFACENAME		PROPERTIES
com.ibm.xxx. NewCommand		parm1=1&parm2=2

CCPd: parm1=1&parm2=2

VIEWREG

INTERFACENAME		PROPERTIES
com.ibm.xxx.NewView		docName=NewView.jsp

VPd: docName=NewView.jsp

URL: http://hostname.com/NewCommand?storeID=1&....

CCPu: storeID=1&...

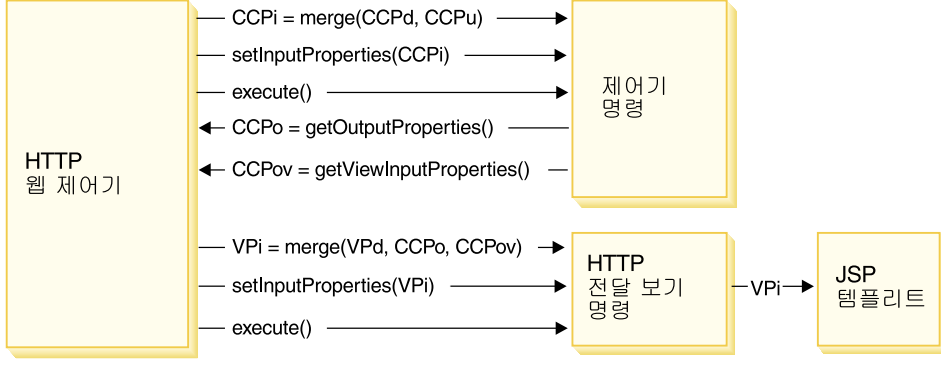


그림 12.

앞의 도표는 다음과 같은 상호 작용을 보여줍니다.

- 웹 제어기는 URL 매개변수의 입력 특성(CCPu)과 제어기 명령의 CMDREG 테이블에 있는 항목(CCPd)을 병합하여 CCPi를 작성합니다.
- 웹 제어기는 병합된 특성(CCPi)을 제어기 명령으로 전달하고 제어기 명령을 실행합니다.

- 제어기 명령은 CCPo로서 출력 특성을 설정합니다. 이러한 특성은 명령 자체에서 발생한 출력 특성입니다. 출력 특성 중 하나인 viewCommandName은 원하는 보기 명령 이름으로 설정됩니다. 이러한 특성은 get 메소드를 사용하여 웹 제어기에서 검색됩니다.
- 제어기 명령은 CCPov로서 다른 출력 특성 세트를 설정합니다. 기본적으로, 이러한 특성은 원래 병합된 입력 특성(CCPi)으로 설정되어 있습니다. 이러한 특성을 사용자 정의할 수 있습니다. 예를 들어 보기 명령으로 모든 입력 매개변수를 전달해야할 필요는 없을 수도 있습니다.
- 웹 제어기는 세 가지 특성 세트인 CCPo, CCPov 및 VPd(VIEWREG 테이블에 등록된 특성)를 보기 명령의 입력 특성(VPi)으로 병합합니다.
- 웹 제어기는 병합된 특성 VPi를 설정하고 보기 명령을 실행합니다.
- 보기 명령은 입력 특성의 JSP로 속성을 설정합니다.

새 명령 작성시, 특성 병합을 명시적으로 수행할 필요는 없습니다. 추상 명령 클래스에는 mergeProperties 메소드가 들어 있습니다. 이 메소드에 대한 자세한 내용은 WebSphere Commerce 온라인 도움말의 "참조" 절을 참조하십시오.

필수 특성 설정

제어기 명령은 보기 명령 유형마다 다음과 같은 특성을 설정해야 합니다. 특성이 명령에서 설정되지 않으면 VIEWREG 테이블에 정의되어야 합니다.

- ForwardView 명령을 사용 중이면 `docname = view_file_name`을 설정하십시오. 여기서, *view_file_name*은 표시 템플리트의 이름입니다(예: `docname = productDisplay.jsp`).
- DirectView 명령을 사용할 경우, 다음 중 하나를 수행하십시오.
  - `textDocument = xxx`를 설정하십시오. 여기서 *xxx*는 텍스트 양식의 문서를 포함하는 java.io.InputStream 오브젝트입니다.
  - `rawDocument = yyy`를 설정하십시오. 여기서 *yyy*는 2진 양식의 문서를 포함하는 java.io.InputStream 오브젝트입니다.

  DirectView 명령을 사용할 경우, `contentType = ttt`를 설정하는 것은 선택적입니다. 여기서 *ttt*는 문서 컨텐츠 유형입니다.

- RedirectView 명령을 사용할 경우, `url = uuu`를 설정하십시오. 여기서 *uuu*는 경로 재지정 URL입니다.

제 3 장 지속 오브젝트 모델

WebSphere Commerce는 다량의 지속 데이터를 처리합니다. 여기에는 현재 데이터베이스 스키마에 520개를 초과하는 테이블이 정의되어 있습니다. 이러한 대규모의 스키마에서도 특정 비즈니스 요구에 맞게 데이터베이스 스키마를 확장하거나 사용자 정의해야 할 수도 있습니다.

WebSphere Commerce는 EJB 구성요소 아키텍처를 기반으로 하는 엔티티 bean을 지속 오브젝트 계층으로 사용합니다. 이러한 엔티티 bean은 상거래 도메인에서 개념 및 오브젝트를 모델링하는 방식으로 WebSphere Commerce 데이터를 나타냅니다. 이러한 지속 계층은 확장 가능한 프레임워크를 제공합니다.

VisualAge for Java, Enterprise Edition에서는 이 프레임워크의 개발을 지원하는 정교한 EJB 도구 및 단위 테스트 환경을 제공합니다.

WebSphere Commerce 엔티티 bean의 구현

WebSphere Commerce 엔티티 bean - 개요

이전에 언급한 대로 WebSphere Commerce 아키텍처 내의 지속 계층은 EJB 구성요소 아키텍처에 따라 구현됩니다. EJB 아키텍처는 두 가지 유형의 엔터프라이즈 bean인 엔티티 bean과 세션 bean을 정의합니다. 엔티티 bean은 CMP(container-managed persistence) bean과 BMP(bean-managed persistenc) bean으로 세분화됩니다.

대부분의 WebSphere Commerce 엔티티 bean은 CMP 엔티티 bean입니다. 적은 수의 무상태 세션 bean은 특정 열에서 모든 행의 합계 산정과 같은 집약적인 데이터베이스 조작을 처리하는 데 사용됩니다. CMP 엔티티 bean을 사용할 때의 한 가지 이점은 개발자가 VisualAge for Java, Enterprise Edition에서 제공되는 EJB 도구를 이용할 수 있다는 점입니다. 개발자는 이러한 도구를 사용하여 Java 오브젝트 및 해당 데이터베이스 테이블 맵핑을 정의할 수 있습니다. 도구는 엔티티

bean의 필수 지속자를 자동으로 생성합니다. 지속자는 데이터베이스에 Java 필드를 지속시키고 데이터베이스의 데이터로 Java 필드에 대량 자료 반입하는 Java 오브젝트입니다.

VisualAge for Java에서는 현재 EJB 스펙을 확장한 EJB 상속 및 연관 두 가지를 제공합니다. EJB 상속을 사용하여 엔터프라이즈 bean은 동일한 그룹에 상주하는 또다른 엔터프라이즈 bean에서 특성, 메소드 및 메소드 레벨 제어 설명자 속성을 상속할 수 있습니다. 연관은 두 개의 CMP 엔티티 bean 사이에 존재하는 관계입니다.

일부 WebSphere Commerce 엔티티 bean은 EJB 상속 기능을 활용합니다. WebSphere Commerce 엔티티 bean은 VisualAge for Java에서 제공되는 연관 기능을 사용하지 않습니다. 고유의 엔티티 bean 개발시, VisualAge for Java의 연관 기능을 사용하지 않는 것이 바람직합니다. 이와 같이 권장하는 것은 오브젝트 모델에서 복잡도를 최소화하기 위해서 입니다. VisualAge for Java에서 제공되는 연관 기능을 사용하지 않고 엔터프라이즈 bean에 명시적 getter 메소드를 추가하여 엔터프라이즈 bean 사이의 오브젝트 관계를 설정할 수 있습니다.

WebSphere Commerce에서는 개인용 및 공용의 두 엔터프라이즈 bean 세트를 제공합니다. 개인용 엔터프라이즈 bean은 WebSphere Commerce 런타임 환경 및 도구에서 사용됩니다. 이러한 bean을 사용하거나 수정해서는 *안됩니다*.

이와는 반대로 공용 엔터프라이즈 bean은 상거래 응용프로그램에서 사용되며 사용되고 확장될 수 있습니다. 이러한 공용 엔터프라이즈 bean은 다음 EJB 그룹으로 구성됩니다.

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction
- WCSCatalog
- WCSCommon
- WCSContract
- WCSCoupon
- WCSFulfillment

- WCSInventory
- WCSMessageExtensions
- WCSOrder
- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

위에 나열된 EJB 그룹 중 일부에는 세션 bean이 들어 있습니다. 이후의 이주를 단순화하려면 세션 bean 클래스를 수정해서는 안됩니다. 필요한 경우, 새 EJB 그룹에서 새 세션 bean을 작성할 수 있습니다. 새 세션 bean 작성에 대한 자세한 내용은 85 페이지의 『새 세션 bean 작성』을 참조하십시오.

WebSphere Commerce 엔터프라이즈 bean의 전개 설명자

전개 설명자는 엔터프라이즈 bean의 런타임 설정이 들어 있는 직렬화된 특수 클래스입니다. WebSphere Commerce 엔터프라이즈 bean의 전개 설명자는 특별한 방법으로 설정되며 수정해서는 안됩니다.

새 엔터프라이즈 bean(엔티티 또는 세션 bean) 작성시, bean을 마우스 오른쪽 버튼으로 누르고 **특성**을 선택하여 EJB 개발 환경(VisualAge for Java) 내에서 전개 설명자를 설정하십시오. 새 엔터프라이즈 bean의 전개 설명자는 WebSphere Commerce 엔터프라이즈 bean의 경우와 동일한 규칙을 따라야 합니다. 특히 다음과 같이 속성을 설정해야 합니다.

속성	값
트랜잭션 속성	TX_REQUIRED
분리 레벨	TRANSACTION_READ_COMMITTED
실행 모드	SYSTEM_IDENTITY 또는 CLIENT_IDENTITY

속성	값
재진입	선택되었는지 확인하십시오.

엔터프라이즈 bean은 가끔 데이터베이스에서 정보를 읽기만 하고 데이터베이스를 갱신하지는 않는 메소드가 들어 있습니다. 이러한 메소드는 *읽기 전용* 메소드로 알려져 있습니다. 모든 읽기 전용 메소드는 명시적으로 다음과 같이 표시되어야 합니다(메소드를 마우스 오른쪽 버튼으로 누르기 및 **EJB 메소드 속성 > 읽기 전용 메소드** 선택). 읽기 전용 메소드가 이러한 방법으로 표시되지 않으면 EJB 컨테이너는 트랜잭션이 끝날 때 불필요하게 데이터베이스 갱신을 시도하며 이로 인해 읽기 전용 트랜잭션에서 트랜잭션 롤백 오류가 발생합니다. 이 경우 성능 문제점이 발생합니다.

분리 레벨

VisualAge for Java 안의 WebSphere Commerce 엔터프라이즈 bean에 사용되는 트랜잭션 분리 레벨은 `TRANSACTION_READ_COMMITTED`입니다. DB2®용 JDBC 드라이버 및 Oracle용 JDBC 드라이버의 분리 레벨 구현 간에는 차이점이 있음에 유의하십시오. WebSphere Application Server 환경으로 전개시 사용되는 트랜잭션 분리 레벨은 어떤 데이터베이스가 사용 중인가에 따라 다릅니다.

WebSphere Test Environment 외부에서 작동시 DB2 데이터베이스에 사용하는 분리 레벨은 `TRANSACTION_REPEATABLE_READ`입니다. WebSphere Test Environment 외부에서 작동시 Oracle 데이터베이스에 사용하는 분리 레벨은 `TRANSACTION_READ_COMMITTED`입니다.

DB2 데이터베이스로 전개하는 경우, 트랜잭션 분리 레벨을 수동으로 변경하지 않아도 되며 `modifyIsolationLevel` 명령을 발행할 때 전개 단계 중 변경됩니다.

다음 표는 DB2 및 Oracle 트랜잭션 분리 레벨을 해당 JDBC 트랜잭션 분리 레벨로 맵핑하는 것을 나타냅니다.

JDBC	DB2	Oracle
읽기 미확약	미확약 읽기	읽기 미확약
읽기 확약	포인터 안정성	(사용할 수 없음)
반복 가능한 읽기	읽기 안정성	읽기 확약
직렬화 가능	반복 가능한 읽기	직렬화 가능

JDBC	DB2	Oracle
없음	(사용할 수 없음)	(사용할 수 없음)

DB2에서의 포인터 안정성 트랜잭션 분리 레벨의 경우, 제공된 트랜잭션 중 갱신된 행만 배타적으로 잠깁니다. SQL 문에서 리턴된 결과 세트의 일부이지만 제공된 행의 어떠한 열도 갱신되지 않은 경우, 특정 행의 행 잠금은 포인터가 다른 행으로 이동하고 나서 해제됩니다. 일부 경우(예: 재고 갱신), 원하는 작동이 아닐 수도 있습니다. 그러므로 동시성에서의 약간의 손실을 감수하고 DB2의 읽기 안정성으로 트랜잭션 분리 레벨을 변경하면, 데이터 무결성을 크게 개선할 수 있습니다.

읽기 확약 트랜잭션 분리 레벨 또는 JDBC 반복 가능한 읽기가 사용 가능한 Oracle의 경우, 실제 구현은 DB2에서의 반복 가능한 읽기 트랜잭션 분리 레벨과 아주 유사한 작동을 수행하는 방식으로 수행됩니다.

WebSphere Commerce 오브젝트 모델로 확장

WebSphere Commerce 오브젝트 모델은 다음과 같은 방법으로 확장될 수 있습니다.

- WebSphere Commerce의 공용 엔터프라이즈 bean 확장
- 새 엔티티 bean 작성
- 새로운 무상태 세션 bean 작성

이러한 확장을 수행하는 방법에 대한 정보는 다음 절에 나와 있습니다.

오브젝트 모델 확장 방법론

응용프로그램 요구사항에 따라 기존의 WebSphere Commerce 오브젝트 모델을 확장해야 하는 경우도 있습니다. 이러한 요구사항의 예로는 응용프로그램에 속성을 추가하는 경우를 들 수 있습니다. 다음 중 한 가지 방법으로 이 작업을 수행할 수 있습니다.

기존 WebSphere Commerce 공개 엔티티 bean을 수정하지 않고

새 데이터베이스 테이블을 작성한 다음 해당 테이블의 새 엔티티 bean을 작성합니다. 필요에 따라 새 속성을 실행하는 데 필요한 필드와 메소드를 엔티티 bean에 추가하십시오. 새 엔티티 bean용 전개 코드 및 액세스 bean

을 생성하십시오. 응용프로그램에 새 속성이 필요한 경우, 응용프로그램은 액세스 bean 오브젝트의 인스턴스 생성 및, 해당 메소드를 사용하여 속성을 검색, 설정 및 실행합니다.

기존 WebSphere Commerce 공개 엔티티 bean을 수정하여
새 데이터베이스 테이블을 작성한 다음, 새 테이블과 현재 수정 중인 기존의 엔터프라이즈 bean에 해당하는 기존 테이블 간에 테이블 결합을 작성하십시오. 기존 WebSphere Commerce 공개 엔티티 bean에 새 필드를 작성한 다음, 2차 테이블 맵을 사용하여 새 테이블의 해당 열에 필드를 맵핑하십시오. 필요한 메소드를 추가하십시오. 기존 엔티티 bean용 전개 코드 및 액세스 bean을 다시 생성하십시오. 응용프로그램이 액세스 bean 오브젝트의 인스턴스를 생성하면 새 속성을 사용할 수 있습니다.

이러한 두 가지 접근 방법 간의 균형이 있습니다. 일반적으로 이 균형은 성능 및 코드 유지보수를 위한 노력과 관련이 있습니다.

확장 예: 고객의 주거 유형을 알아내야 하는 응용프로그램의 예를 고려해 보십시오. 고객 ID 및 주거 유형을 포함하는 USERRES 테이블을 작성합니다. 여기서 주거 유형(resType)은 주택, 빌라 또는 아파트입니다. 이러한 유형의 정보는 인적사항 정보이므로 기존의 Commerce Suite USERDEMO 테이블과 연관되어 있습니다. WebSphere Commerce 코드 저장소를 살펴보면 WCSUser EJB 그룹에 ″인적사항″ 엔터프라이즈 bean이 들어 있음을 알 수 있습니다. 이 bean에는 USERDEMO 테이블에 저장된 인적사항 정보에 사용되는 getter 및 setter가 들어 있습니다.

사용자 정의를 수행하기 위한 두 가지 옵션이 있습니다. USERRES 테이블과 상호 작용하는 새 엔티티 bean을 작성하거나 새 필드(해당 getter 및 setter 메소드 포함)를 인적사항 bean에 추가할 수 있습니다.

첫 번째 접근 방법(완전한 새 코드 작성)을 사용할 경우, Userres 엔티티 bean을 작성하고 해당 필드를 USERRES 테이블 열에 맵핑할 수 있습니다. 응용프로그램에 고객의 거주 유형이 필요할 경우, Userres 액세스 bean 오브젝트의 인스턴스를 생성하고 데이터를 검색해야 합니다. 응용프로그램에 동시에 기타 인적사항 정보가 필요할 경우, 인적사항 액세스 bean 오브젝트의 인스턴스 또한 작성하고 다른 모든 필수 속성을 검색해야 합니다. 고객의 완전한 인적사항 정보를 검색하는 모

든 응용프로그램의 로직 부분을 수정하여 원래의 액세스 bean 뿐만 아니라 액세스 bean의 인스턴스를 생성하도록 해야 합니다. 다음 도표는 오브젝트 모델을 확장하기 위한 접근 방법을 표시합니다.

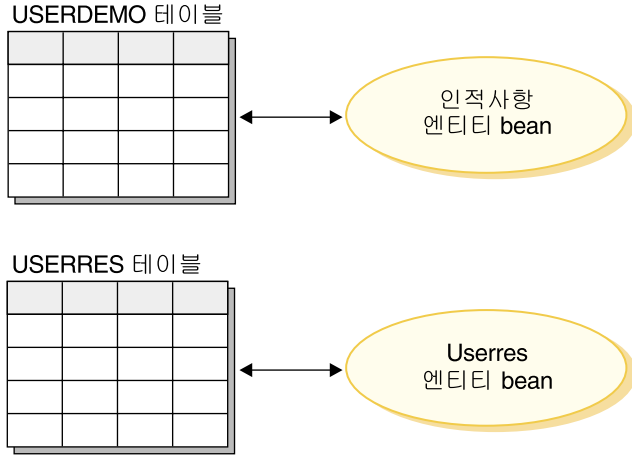


그림 13.

표시 템플리트 관점에서 보면 데이터 bean은 새 속성에 액세스할 수 있어야 해당 정보를 JSP 템플리트에서 사용할 수 있습니다. JSP 템플리트를 작성하는 웹 개발자들에게 공통된 보기를 제공하려면 기존의 원본 엔티티 bean용 액세스 bean을 확장하는 새 데이터 bean을 작성해야 합니다. 또한 데이터 bean은 위임을 사용하여 액세스 bean에서 속성에 대량 자료를 반입해야 합니다. 다음 도표는 이러한 데이터 bean 구현 시나리오를 보여줍니다.

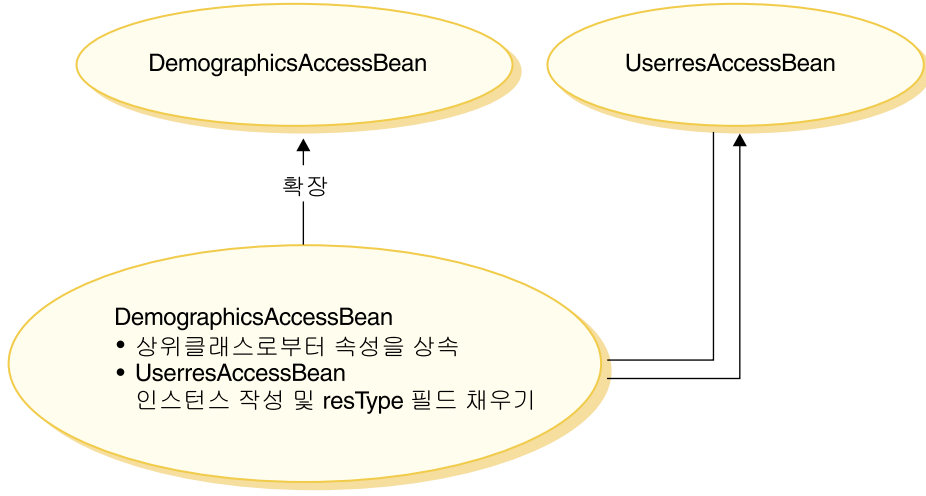


그림 14.

두 번째 접근 방법(기존 코드 수정)을 사용하여 인적사항 엔티티 bean에 새 필드를 추가하고 새 필드와 USERRES 테이블의 적절한 열 사이에 2차 테이블 맵핑을 작성하십시오. 응용프로그램에 고객의 거주 유형이 필요할 경우, 인적사항 액세스 bean 오브젝트의 인스턴스를 생성하고 거주 유형을 검색해야 합니다. 응용프로그램에 고객에 대한 기타 인적사항 정보가 필요할 경우 bean을 동일한 방식으로 호출하여 정보를 얻을 수 있습니다. 다음 도표는 엔터프라이즈 bean 수정을 위한 접근 방법을 보여줍니다.

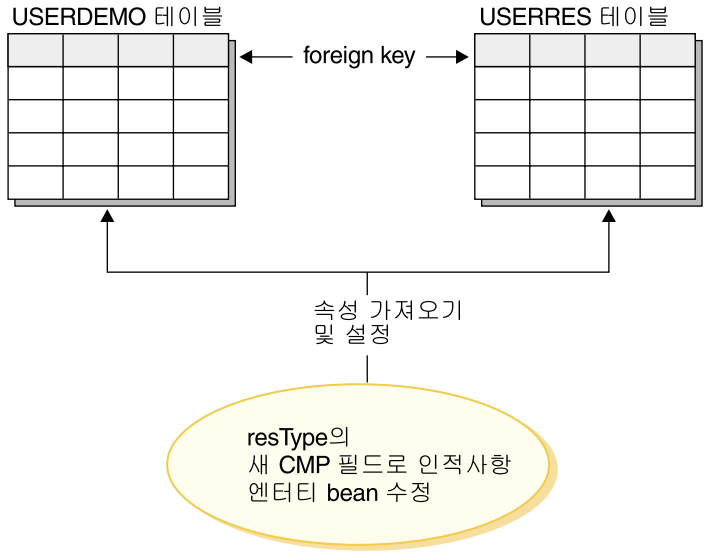


그림 15.

표시 템플리트 관점에서 보면 DemographicsAccessBean이 다시 작성되자마자 속성(resType)을 자동으로 데이터 bean에서 사용할 수 있게 됩니다.

오브젝트 모델을 확장할 때는 기존 WebSphere Commerce 데이터베이스 테이블에 새 열을 추가할 수 없음을 유의하십시오. 새 속성에 대한 새 테이블을 작성해야 합니다. 기존 테이블에 새 열을 추가하려고 할 경우, WebSphere Commerce의 향후 릴리스로 이주하면 새 속성이 유실됩니다.

성능 및 코드 유지보수 관련 고려사항: 두 번째 접근 방법의 경우 런타임 성능이 보다 우수합니다. 새 속성을 가져오고 설정할 경우 하나의 엔티티 bean의 인스턴스를 생성하고 한 번만 가져오면 필요한 모든 속성을 검색할 수 있기 때문입니다.

두 번째 접근 방법은 기존 WebSphere Commerce 코드를 수정하기 때문에 새 WebSphere Commerce 코드 저장소가 릴리스될 때 이주 문제가 발생합니다. 사용자 정의 코드를 새 코드로 이주해야 하지만 코드의 새 저장소를 반입할 경우 엔터프라이즈 bean과 새 테이블에 추가한 필드 사이의 정보 맵핑이 보존되지 않습니다. 따라서 WebSphere Commerce 코드 저장소의 새 릴리스로 이주할 때는 다음 단계를 수행해야 합니다.

1. 사용자 정의된 EJB 코드를 버전화하십시오.
2. WebSphere Commerce 코드의 새 버전을 반입하십시오.
3. VisualAge for Java의 도구를 사용하여 사용자 정의 코드 버전을 WebSphere Commerce 코드의 새 릴리스와 비교하십시오. 사용자 정의 코드를 다시 작업 영역으로 병합하십시오.
4. WebSphere Commerce 공개 엔터프라이즈 bean에 추가한 모든 속성을 데이터베이스의 해당 열에 수동으로 다시 맵핑하십시오.
5. 전개된 코드 및 4단계에서 수정한 엔터프라이즈 bean용 액세스 bean을 다시 생성하십시오.

이주가 더 간단하게 수행되도록 하려면 개발시 오브젝트 모델 확장을 완전 문서화해야 합니다.

오브젝트 모델을 확장할 때는 두 접근 방법을 혼합하여 사용하도록 선택할 수 있습니다. 성능 저하의 위험이 적은 시스템 영역의 경우 첫 번째 접근 방법을 사용할 수 있고, 성능이 문제가 되는 경우 두 번째 접근 방법을 사용할 수 있습니다. 이와 같이 시스템 성능을 양호한 수준으로 유지하면서도 향후 이주에 드는 노력을 최소화할 수 있습니다.

세션 bean의 권장 사용

WebSphere Commerce의 강점 중 하나는 CMP 엔티티 bean을 이용하는 기능에서 유래합니다. CMP 엔티티 bean은 VisualAge for Java 내에서 제공되는 도구화에 의해 생성될 수 있는 분배적, 지속적 및 트랜잭션적인 서버측 Java 구성요소입니다. 여러 경우에 CMP 엔티티 bean은 오브젝트 지속성을 위한 훌륭한 선택이며 최소한 기타 오브젝트 관계형 맵핑 옵션과 동등한 효율적 작업이 가능하거나 또는 그 보다 훨씬 효율적일 수 있습니다. 이러한 이유로 WebSphere Commerce는 CMP bean을 사용하여 핵심 commerce 오브젝트를 구현했습니다.

그러나 세션 bean JDBC 헬퍼를 사용하도록 권장되는 몇 가지 상황이 있습니다. 이러한 상황은 다음을 포함합니다.

- 조회로 많은 결과 세트를 리턴하는 경우. 이것은 *대량 결과 세트* 경우로 표시합니다.

- 조회시 여러 테이블에서 데이터를 검색하는 경우. 이것은 *통합 엔티티 경우*로 표시합니다.
- SQL 문이 데이터베이스 집약 조작을 수행하는 경우. 이것은 *임의 SQL* 경우로 표시합니다.

자세한 내용은 다음 절에서 제공됩니다.

세션 bean을 JDBC 랩퍼로 사용하여 데이터베이스에서 정보를 검색하는 경우, 자원 레벨 액세스 제어를 구현하기가 더 어려워집니다. 세션 bean을 이런 방식으로 사용하면, 세션 bean 개발자가 적절한 "`where`" 절을 "`select`" 문에 추가하여 권한없는 사용자가 자원에 액세스하지 못하도록 해야 합니다.

대량 결과 세트 경우: 조회로 많은 결과 세트를 리턴하는 경우이며 검색된 데이터는 주로 읽기 또는 표시를 위한 목적을 갖습니다. 이 경우, 무상태 세션 bean 사용에 좋으며 해당 세션 bean 내에서, 엔티티 bean의 finder 메소드와 동일한 기능을 수행하는 finder 메소드 작성에 좋습니다. 즉, 무상태 세션 bean의 finder 메소드는 다음을 수행해야 합니다.

- SQL select 문 수행
- 가져온 각 행에 대해 액세스 bean의 인스턴스 생성
- 검색된 각 열에 대해 액세스 bean에서 해당 속성 설정

액세스 bean이 리턴되면, 해당 명령은 액세스 bean이 세션 bean의 finder 메소드에 의해 리턴되었는지 또는 엔티티 bean의 finder 메소드에서 리턴되었는지 여부를 인식하지 못합니다. 결과적으로 세션 bean에서 finder 메소드를 사용하면 프로그래밍 모델에 대한 변경을 야기하지 않습니다. 호출 명령만이 세션 bean 또는 엔티티 bean의 finder 메소드를 호출 중인지 여부를 압니다. 프로그래밍 모델의 다른 모든 부분에 대해 명료합니다.

통합 엔티티 경우: 이 경우, 하나의 보기가 여러 오브젝트 부분을 결합하며 단일 표시 페이지는 여러 데이터베이스 테이블에서 제공되는 정보로 채워집니다. 예를 들어 "회원 정보"의 개념을 고려하십시오. 이것은 고객 정보 테이블의 정보(예: 고객 이름, 연령 및 고객 ID)와 주소 테이블의 정보(예: 주소는 거리 및 구/군/시로 구성)로 구성될 수 있습니다.

단순 SQL 문을 구성하여 SQL 결합을 수행함으로써 다양한 테이블에서 모든 정보를 검색하는 것이 가능합니다. 이것을 “deep fetch” 수행으로 나타낼 수 있습니다. 다음은 “회원 정보” 예에 대한 SQL select 문의 예이며, 여기서 CUSTOMER 테이블은 T1이며 ADDRESS 테이블은 T2입니다.

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

VisualAge for Java에서의 EJB 도구화는 deep fetch를 지원하지 않습니다. 대신 연관된 각 오브젝트에 대한 SQL select를 초래하는 lazy fetch를 수행합니다. 이것은 이러한 유형의 정보를 검색하기 위해 선호되는 메소드가 아닙니다.

deep fetch를 수행하려면, 세션 bean을 사용하는 것이 좋습니다. 해당 세션 bean에서 필수 정보를 검색하려면 finder 메소드를 작성하십시오. finder 메소드는 다음을 수행해야 합니다.

- deep fetch에 대해 SQL select 문을 수행하십시오
- 연관된 각 오브젝트에 대해서는 물론 기본 테이블에서 각 행에 대한 액세스 bean의 인스턴스를 생성하십시오.
- 가져온 각 열과 각 연관 오브젝트의 경우, 해당 속성을 액세스 bean에 설정하십시오.

액세스 bean은 예외를 발생시킨 getter 메소드를 캐시하지 않음에 유의하십시오. 이 경우, 다음 패턴을 사용하여 액세스 bean에 대한 단순 랩퍼 클래스를 작성해야 합니다.

```
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;
      }
}
```

CUSTOMER 및 ADDRESS 예에 이어 세션 bean finder 메소드는 CUSTOMER 테이블의 각 행에 대해 CustomerAccessBean의 인스턴스를 생성하며 ADDRESS 테이블의 각 해당 행에 대해 AddressAccessBean의 인스턴스를 생성합니다. 그런 다음, ADDRESS 테이블의 각 열에 대해 AddressAccessBean(거리 및 구/군/시)에 속성을 설정합니다. ADDRESS 테이블의 각 열마다 속성을 CustomerAccessBean(이름, 연령 및 주소)에 설정합니다. 이것은 다음 도표에 표시됩니다.

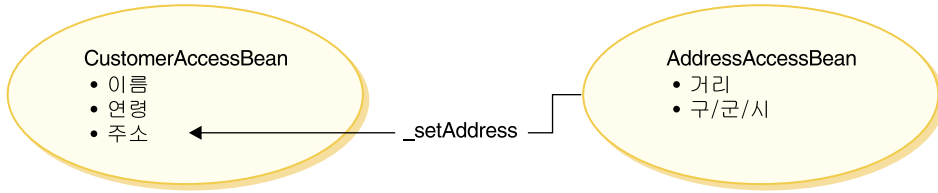


그림 16.

임의 SQL 경우: 이 경우, 데이터베이스 집약 조작을 수행하는 임의 SQL 문 세트가 있습니다. 예를 들어 테이블에서 모든 행을 합하는 조작을 데이터베이스 집약 조작이라고 합니다. 선택된 모든 행이 지속적 모델에서 엔티티 bean에 해당하는 것은 아닙니다.

임의 SQL 문의 작성 결과를 가져올 수 있는 경우는 고객이 매우 많은 데이터 세트에서 찾아보기하려고 할 때입니다. 예를 들어 고객이 온라인 공구 상점의 모든 쥐쇠나 온라인 의류 상점의 모든 드레스를 찾아보려는 경우가 있습니다. 이런 경우 큰 결과 세트가 작성되지만 대개 이 결과 세트에서 각 행 중 몇 개의 필드만이 필요합니다. 즉, 항목 이름, 그림 및 가격을 표시하는 정보 요약만이 처음에 고객에게 제공될 수 있습니다.

이 경우 세션 bean helper 메소드를 작성하십시오. 이 세션 bean helper 메소드는 읽기 또는 쓰기 조작을 수행합니다. 읽기 조작 수행시, 표시 용도로 사용되는 읽기 전용 값 오브젝트를 리턴합니다.

해당 데이터 모델링에서 임의 SQL 문의 경우의 수는 보통 최소화될 수 있습니다.

공용 엔티티 bean 확장

이 절에서는 WebSphere Commerce 공용 엔티티 bean의 설계 패턴에 대해 설명합니다. 이 설계 패턴을 사용하여 새 지속 필드, 새 비즈니스 메소드 또는 새 finder 메소드 추가와 같은 확장 기능을 수행할 수 있습니다.

다음 도표는 카탈로그 엔티티 bean의 구현 클래스를 보여줍니다.

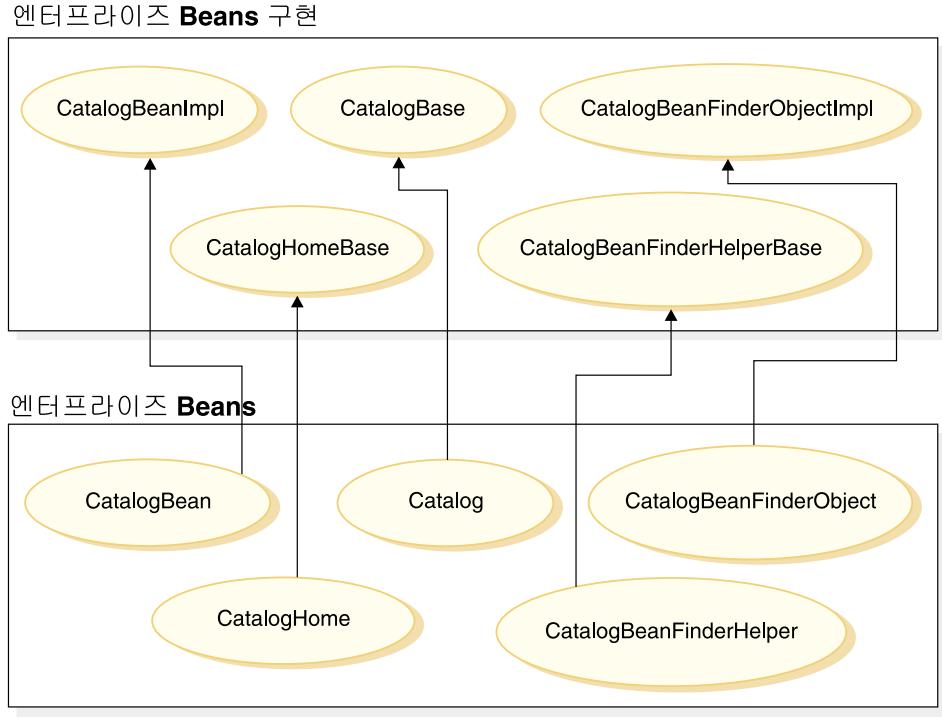


그림 17.

앞의 도표는 다른 엔티티 bean에도 적용됩니다. 이는 유사한 방식으로 구성되었으며 동일한 이름 지정 규칙을 따르기 때문입니다. 도표를 다른 엔티티 bean에 적용하려면 엔티티 bean 이름을 “Catalog”로 대체하십시오. 예를 들어 InterestItemBean 클래스는 InterestItemBeanImpl 클래스를 확장하며 InterestItem 인터페이스는 InterestItemBase 인터페이스를 확장합니다.

도표는 공용 엔터프라이즈 bean의 구현 클래스나 인터페이스가 Java 상속을 사용하여 두 부분으로 분리되었음을 보여줍니다. 최상위 클래스나 인터페이스에는 WebSphere Commerce 구현 코드가 들어 있습니다. 해당 최상위 클래스 및 인터페이스 모두 하위 클래스 및 인터페이스의 별도 패키지 및 프로젝트에 정의되어 있습니다.

*bean_name*BeanFinderHelperBase 및 *bean_name*BeanFinderObjectImpl 클래스를 제외하고 WebSphere Commerce 코드 저장소에는 이러한 최상위 클래스 및 인터페이스의 2진 코드가 들어 있습니다. finder마다 SQL 문이 정의되는 방법을 볼 수 있도록 *bean_name*BeanFinderHelperBase 및 *bean_name*BeanFinderObjectImpl 클래스의 소스 코드가 포함됩니다. 이러한 클래스는 수정할 수 없습니다.

CatalogBeanFinderHelperBase 및 CatalogBeanFinderObjectImpl에 대한 소스 코드를 조사하려면 com.ibm.commerce.catalog.objsrc 패키지를 여십시오. 기타 패키지는 유사한 이름 지정 규칙을 사용합니다.

하위 클래스 및 인터페이스는 수정될 수 있습니다.

공용 엔터프라이즈 bean에 finder 메소드를 추가할 경우, 메소드 관련 특정 이름 지정 규칙을 따라야 합니다. 메소드의 이름을 findX*a_description*과 같이 지정하십시오. 여기서, *a_description*은 선택항목에 대한 설명입니다. 이름의 예로는 findXByOwnerId 및 findXByOrderStatus가 있습니다. 이와 같이 이름 지정 규칙을 사용하면 WebSphere Commerce finder 메소드와 이름이 충돌(이름 중복)할 위험이 없습니다.

기존 WebSphere Commerce 공용 엔티티 bean을 수정하는 한가지 방법은 추가 필드를 추가하는 것입니다. 이 경우에는 새 필드를 추가한 후 bean에서 각 finder 메소드를 검사해야 합니다. finder 메소드의 where 절 부분이 데이터베이스 별명(예: T1. 또는 T2.)을 포함하는 경우, 별명을 제거해야 합니다.

새 CMP 엔터프라이즈 bean 작성

WebSphere Commerce 오브젝트 모델에 추가해야 할 새 속성이 있는 경우, 필수 속성 열이 있는 새 데이터베이스 테이블을 작성할 수 있습니다. 또한 엔터프라이즈 bean에서 이 속성을 포함해야 WebSphere Commerce 명령이 정보를 액세스할 수 있습니다.

새 속성을 WebSphere Commerce 오브젝트 모델로 통합하는 한 가지 방법은 새 CMP 엔터프라이즈 bean을 작성하는 것입니다. 이 bean에서 새 데이터베이스 테이블에 있는 속성에 해당하는 필드를 작성합니다.

새 CMP 엔터프라이즈 bean을 작성하려면 VisualAge for Java에서 다음 단계를 수행해야 합니다.

1. 작업 영역 소유자가 WCS 개발자로 설정되었는지 확인하십시오.
2. bean용으로 새 EJB 그룹을 작성하십시오.
3. 엔터프라이즈 bean 작성 SmartGuide 도구를 사용하여 새 CMP 엔터프라이즈 bean을 작성하십시오.
4. 해당 데이터베이스 테이블의 열마다 새 CMP 필드를 bean에 추가하십시오.
5. 필요한 경우, 작성되는 각 CMP 필드마다 getter 및 setter 메소드 쌍을 작성하십시오.
6. 필요한 경우, FinderHelper 필드를 FinderHelper 인터페이스에 정의하고 새 FinderHelper 메소드를 추가하십시오.
7. 필요할 경우, 새 ejbCreate 메소드를 작성하고 ejbCreate 메소드를 엔터프라이즈 bean의 홈 인터페이스로 프롬프트하십시오. 새 엔터프라이즈 bean이 해당 데이터베이스 테이블에 새 항목을 작성해야 하는 경우, 이 단계는 필수적입니다.
8. 엔터프라이즈 bean의 필드를 데이터베이스 테이블의 열로 맵핑하십시오.
9. 엔터프라이즈 bean에 대해 전개된 코드 및 액세스 bean을 생성하십시오.

이 각 단계에 대한 자세한 사항은 다음 절에 있습니다.

주: 새 엔터프라이즈 bean이 WebSphere Commerce 액세스 제어 시스템으로 보호되는 경우, 추가 정보는 101 페이지의 제 4 장『액세스 제어』를 참조하십시오. Bean을 작성한 후에 액세스 제어를 추가할 수 있습니다.

사용자 거주 유형 정보를 지정하는 USERRES라는 새 테이블이 있다고 가정하십시오. 이 테이블에는 세 개의 열이 들어 있습니다. USERID 열, 홈 유형을 지정하는 HOME 열 및 거주지의 침실 수를 지정하는 ROOMS 열이 있습니다.

새 EJB 그룹 작성: 엔티티 bean을 작성할 때 WebSphere Commerce EJB 그룹과는 별도의 EJB 그룹에서 작성해야 합니다. VisualAge for Java의 엔터프라이즈 bean으로 작업할 경우, 작업 영역의 EJB 탭으로 전환해야 합니다. 그런 다음 **EJB** 메뉴에서 **추가 > EJB 그룹**을 선택하여 EJB 그룹 추가 SmartGuide를 시작하십시오.

EJB 그룹 작성시 지정해야 하는 2개의 중요한 항목이 있습니다. EJB 코드가 저장되는 프로젝트와 EJB 그룹의 이름입니다.

EJB 프로젝트는 작업 영역의 프로젝트 탭에서 볼 수 있습니다. 새 EJB 그룹을 작성할 때 WebSphere Commerce 프로젝트와는 별도의 프로젝트를 지정해야 합니다. 예를 들어 자신의 EJB 그룹이 MyCustomEJB 프로젝트를 사용하도록 선택할 수 있습니다. VisualAge for Java가 이 프로젝트를 자동으로 작성할 수 있으므로 그룹을 작성하기 전에 이 프로젝트가 있어야 할 필요는 없습니다. 이 프로젝트는 EJB 코드용으로만 사용할 수 있습니다. 명령 또는 데이터 bean 코드에 사용해서는 안됩니다. 이러한 코드 유형 분리는 전개를 위해 필수적입니다. WebSphere Commerce 코드에서 사용자 정의 코드를 분리함으로써 향후 릴리스로 이주할 때의 영향을 최소화하게 됩니다

프로젝트 및 EJB 그룹 이름의 경우, 응용프로그램의 적절한 이름 지정 규칙을 따랐는지 확인하십시오.

새 CMP 엔터프라이즈 bean 작성: 새 CMP 엔터프라이즈 Bean을 작성하기 위해, 엔터프라이즈 bean 작성 SmartGuide 도구를 사용할 수 있습니다. 도구를 실행하려면 EJB 그룹을 마우스 오른쪽 버튼으로 눌러 이 그룹에 새 bean을 추가하고 **추가 > 엔터프라이즈 Bean**을 선택하십시오.

새 엔터프라이즈 bean을 작성하도록 선택하고 bean의 이름을 지정하십시오. 엔터프라이즈 bean의 WebSphere Commerce 이름 지정 규칙은 bean이 해당하는 테

이블과 동일한 이름을 사용하여 bean을 이름 지정합니다. 예를 들어 새 데이터베이스 테이블의 이름이 USERRES인 경우, 엔터프라이즈 bean의 이름은 UserRes이어야 합니다.

프로젝트 필드를 자동으로 프로젝트 이름으로 대량 자료 반입합니다. 코드가 저장되는 프로젝트 내의 패키지 이름을 지정해야 합니다. 패키지에 저장될 코드의 예로는 엔터프라이즈 bean용으로 작성한 액세스 bean 코드가 있습니다. 또한 패키지 이름을 지정할 때는 응용프로그램에 맞는 적절한 이름 지정 규칙을 따랐는지 확인하십시오. 패키지 이름의 예는 `com.mycompany.mycustombeans`입니다.

bean 클래스에서 VisualAge for Java는 `EntityContext`라는 개인용 필드를 작성합니다. WebSphere Commerce는 ECEntityBean에 자체 엔티티 컨텍스트 필드를 제공하며 새 엔티티 bean은 자체 클래스에 생성된 필드보다는 해당 필드를 사용해야 합니다. 이와 같이 새 엔티티 bean에서 생성된 `EntityContext`를 제거해야 합니다.

새 엔터프라이즈 bean에는 `serialVersionUID` 필드가 들어 있어야 합니다. 새 bean을 작성하기 위해 SmartGuide를 사용하는 경우, VisualAge for Java가 이 필드를 생성합니다. bean을 작성하기 위해 해당 도구를 사용하지 않는 경우, 이 필드를 추가해야 합니다.

최상위 클래스 필드에서 `com.ibm.commerce.base.objects.ECEntityBean` 클래스를 지정해야 합니다. 다음 코드 예는 최상위 클래스에서 제공되는 함수를 보여줍니다.

```
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 --
   }
}
```

또한 인수가 없는 ejbCreate() 및 ejbPostCreate() 메소드를 제거해야 합니다. 이 메소드를 엔티티 bean에 놓으면 런타임 오류가 발생할 수 있습니다.

데이터베이스 테이블의 열마다 Bean에 새 CMP 필드를 작성해야 합니다(bean에 CMP 필드 추가 옵션을 보려면 SmartGuide에서 다음을 눌러야 합니다). 필드마다 필드 이름과 필드 데이터 유형을 지정하십시오. 1차 키 또는 1차 키의 일부인 열의 경우, 키 필드 선택란을 사용하십시오. 다른 모든 열의 경우, 원격 인터페이스 선택란에 대한 승격 getter 및 setter 메소드를 사용하십시오.

모든 필드가 채워지면, 완료를 누르고 VisualAge for Java는 새 CMP 엔터프라이즈 bean을 작성합니다.

***Bean_Name*FinderHelper 인터페이스에서 새 FinderHelper 필드 작성:**
*Bean_Name*FinderHelper 인터페이스에는 findByPrimaryKey 메소드 이외의 모든 FinderHelper 메소드에 해당하는 SQL 검색 절이 들어 있습니다.

엔터프라이즈 bean은 FinderHelper 메소드와 함께 bean의 FinderHelper 인터페이스에 정의되어 있는 findXBy*ArgName*(여기서 *ArgName*은 인수의 이름) 메소드를 사용하여 SQL 조회를 작성합니다. 필드 이름이 WebSphere Commerce 필드 이름에서 항상 고유한지 확인하려면 사용자 필드 이름에 대한 “findXBy” 이름 지정 규칙을 사용하십시오.

Bean에 새 FinderHelper 필드를 작성하려면 *Bean_Name*_FinderHelper 인터페이스를 선택하고 소스 코드를 수정하여 새 select 문의 구성 방식을 설정해야 합니다. 예는 다음과 같습니다.

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

홈 인터페이스는 findXByHomeAndRooms라는 메소드가 필요합니다. 이것은 ?로 표시된 값을 대량 자료 반입하는 각각의 HOME 및 ROOMS에 대한 입력 매개변수를 가지고 있습니다. 이러한 방식으로 조회를 생성하는 것을 *매개변수 입력*이라 합니다.

입력 매개변수가 ″detached″ 및 ″3″인 경우, 생성된 SQL 문은 다음과 같습니다.

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

보안상의 이유로 엔티티 bean용 FinderHelper 메소드를 작성할 때는 매개변수 삽입을 사용해야 합니다. 이 방식을 권장하는 이유는 다른 사람이 조회를 변경하지 못하도록 방지할 수 있기 때문입니다. 또다른 접근 방법은 다음과 유사한 구성을 사용하는 것입니다.

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

여기서 *input_string*은 URL에서 전달된 문자열 값입니다. 이 방식은 바람직하지 않습니다. 권한이 없는 사용자가 “123 OR 1=1”과 같은 값을 입력하여 SQL 문을 변경할 수 있기 때문입니다. 사용자가 SQL 문을 변경할 수 있는 경우, 권한 없이 데이터에 액세스할 수 있습니다. 따라서 매개변수 삽입 접근 방법이 권장됩니다.

매개변수 삽입을 사용할 수 없기 때문에 입력 문자열을 사용하여 SQL 문을 작성해야 하는 경우, 입력 문자열의 매개변수를 확인하여 악의를 가지고 데이터에 액세스하기 위해 입력 매개변수를 사용하지 못하도록 할 수 있습니다.

_Bean_NameHome 인터페이스에 새 FinderHelper 메소드 작성:
_Bean_NameFinderHelper 인터페이스에 지정한 FinderHelper 필드마다 bean의 홈 인터페이스에 FinderHelper 메소드를 작성해야 합니다. FinderHelper 메소드의 이름은 ″WhereClause″이 삭제되는 것만 다르고 FinderHelper 필드 이름과 정확히 일치해야 합니다. 예를 들어 필드 이름이 findXByHomeAndRoomsWhereClause 이면, 해당하는 메소드 이름은 findXByHomeAndRooms입니다.

새 FinderHelper 메소드를 작성하려면 다음을 수행하십시오.

1. *Bean_Name***Home** 인터페이스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오.
   메소드 작성 SmartGuide가 열립니다.
2. **새 메소드 작성**을 선택한 후 **다음**을 누르십시오.
3. **메소드 이름** 필드에 FinderHelper 메소드의 이름을 입력하십시오. FinderHelper 메소드의 이름은 ″WhereClause″ 부분이 삭제되는 것만 다르고 FinderHelper 필드 이름과 정확히 일치해야 합니다. 예를 들어 findXByHomeAndRooms를 입력하십시오.
4. **리턴 유형** 필드에 다음 중 하나를 입력하십시오.

- FinderHelper 메소드가 1차 키를 사용하여 데이터베이스를 조회하고 메소드가 고유한 레코드를 리턴해야 하는 경우, 리턴 유형으로 EJB 오브젝트를 지정하십시오. 예를 들어 UserRes를 입력하십시오.
- FinderHelper 메소드가 고유 레코드 대신 결과 세트를 리턴할 경우, 리턴 유형을 java.util.Enumeration으로 지정하십시오.

5. **이 메소드의 매개변수는 ?** 옆에 있는 **추가** 버튼을 눌러 해당 매개변수를 추가하십시오. 예를 들어, String 유형의 argHome을 추가하여 거주지 유형을 보관하고 byte 유형의 argRooms를 추가하여 방 수를 보유할 수 있습니다.
6. 모든 매개변수를 추가하여 완료했으면 **다음**을 누르십시오.
7. **이 메소드는 어떤 예외를 발생시킵니까?** 옆에 있는 **추가** 버튼을 누르고 다음 예외를 추가하십시오.
   - java.rmi.RemoteException
   - javax.ejb.FinderException

   주: 이전 예외 목록에서는 메소드에서 발생시켜야 하는 최소 예외 세트를 표시합니다. 사용자 자신의 코드에 따라, 다른 예외를 지정할 필요가 있을 수 있습니다.
8. **완료**를 누르십시오.

새 ejbCreate 메소드 작성: 엔터프라이즈 bean이 작성되면 ejbCreate 메소드가 자동으로 생성됩니다. 이 메소드는 원격 인터페이스로 승격되므로 액세스 bean에서 사용 가능합니다. 기본 ejbCreate 메소드에는 1차 키 또는 1차 키의 일부인 매개변수만이 들어 있습니다. 즉, 인스턴스 생성 중 인스턴스가 생성된 값만 해당됩니다.

엔터프라이즈 bean에 1차 키의 일부가 아닌 필드와 널(Null)값이 허용되지 않는 필드가 들어 있는 경우, 새 ejbCreate 메소드를 작성하여 이 메소드에 해당 필드의 인스턴스를 생성해야 합니다. 이와 같이 하면 새 레코드가 작성될 때마다 널(Null)값이 허용되지 않는 모든 필드는 적합한 데이터로 채워집니다.

새 ejbCreate 메소드를 작성하려면 다음을 수행하십시오.

1. 유형 분할창에서 ***Bean_Name*Bean** 클래스를 펼치십시오. 예를 들어 **UserResBean**을 선택하십시오.

2. 소스 코드를 보려면 기존 ejbCreate 메소드를 누르십시오(엔터프라이즈 bean의 1차 키에 따라 ejbCreate(String), ejbCreate(String, int) 또는 다른 입력 매개변수를 취할 수 있습니다).
3. 소스 코드를 수정하여 각 CMP 필드가 메소드에 입력 매개변수로 포함되도록 하여 CMP 필드마다 해당 값을 가진 인스턴스가 생성되도록 해야 합니다. 사용자 ID가 1차 키인 UserRes 예에서 소스 코드는 초기에 다음과 같이 표시됩니다.

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

   그러나 방 갯수 및 홈 유형이 초기화되었는지 확인하려는 경우도 있습니다. 이 경우, 코드를 다음과 같이 변경합니다.

   ```
   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;
   }
   ```

   주: 시스템 생성 1차 키를 사용하려는 경우, 자세한 내용은 90 페이지의 『1차 키』를 참조하십시오.
4. 수정된 메소드를 저장하십시오. 코드를 저장하면 VisualAge for Java는 새 매개변수를 가져가는 새 ejbCreate 메소드를 작성합니다. 원래의 ejbCreate 메소드는 남아 있습니다.
5. 원래 ejbCreate 메소드(인수가 없는 메소드)를 삭제하십시오.
6. 새 ejbCreate 메소드를 마우스 오른쪽 버튼으로 누른 후 **추가 > EJB 홈 인터페이스**를 선택하십시오.
7. 해당 ejbPostCreate 메소드를 작성하십시오(이 메소드를 홈 인터페이스에 추가할 필요는 없습니다).

새 엔터프라이즈 bean에 데이터베이스 테이블 맵핑: 일단 새 엔터프라이즈 bean을 작성했으면 bean의 CMP 필드와 데이터베이스 테이블의 열 사이에 맵핑을 작성해야 합니다. 이러한 맵핑을 스키마라고 합니다. VisualAge for Java에서는 이 태스크를 단순화시키는 도구를 제공합니다.

스키마를 작성하려면 다음을 수행하십시오.

1. **EJB** 메뉴에서 **열기 > 데이터베이스 스키마**를 선택하십시오. 스키마 브라우저가 열립니다.
2. **스키마** 메뉴에서 **스키마 반입/반출 > 데이터베이스에서 스키마 반입**을 선택하십시오.
3. 새 스키마의 이름을 입력한 후 **확인**을 누르십시오.
4. 데이터베이스 연결 창에 다음 정보를 입력하십시오.

속성	DB2 값	Oracle 값
연결 유형	`COM.ibm.db2.jdbc.app.DB2Driver`	`Oracle.jdbc.driver.OracleDriver`
데이터 소스	`jdbc:db2:wc_database_name`	`jdbc:oracle:thin@`*`hostname`*`:`*`port`*`:`*`Oracle_SID`*
사용자 이름	*`wc_db_user_name`*	*`wc_db_user_name`*
암호	*`wc_db_password`*	*`wc_db_password`*

값은 다음과 같이 바뀝니다.

- DB2 *`wc_database_name`*은 WebSphere Commerce 데이터베이스의 이름입니다.
- Oracle *`hostname`*은 Oracle 서버 호스트 이름입니다.
- Oracle *`port`*는 Oracle 데이터베이스의 포트 번호입니다.
- Oracle *`Oracle_SID`*는 Oracle 인스턴스 ID입니다.
- *`wc_db_user_name`*은 데이터베이스의 사용자 이름입니다.
- *`wc_db_password`*는 데이터베이스 암호입니다.

5. **규정자** 목록에서 데이터베이스 규정자(데이터베이스 사용자 이름이 될 수 있음)를 선택하십시오.
6. **테이블 목록 빌드**를 누르십시오.

7. (생성된 목록에서) *your_new_table*을 선택하고 확인을 눌러 스키마를 생성하십시오.
8. 스키마가 생성된 후에 스키마 분할창에서 *your_new_table*을 누르십시오.
9. 테이블 분할창의 *your_new_table*을 강조표시한 다음 두 번 누르십시오.
   테이블 편집기 창이 열립니다.
10. **규정자** 필드에 있는 정보를 제거하십시오. 이것은 다른 시스템으로 이식 가능한 엔터프라이즈 bean을 작성합니다.
11. **스키마** 메뉴에서 **스키마 저장**을 선택하십시오. 해당 프로젝트, 패키지 및 클래스 이름을 입력하십시오.

그 다음 스키마 맵을 작성해야 합니다. 스키마 맵은 엔터프라이즈 bean에서 데이터베이스 열 및 필드 간의 맵핑입니다.

스키마 맵을 작성하려면 다음을 수행하십시오.

1. **EJB** 메뉴에서 **열기 > 스키마 맵**을 선택하십시오.
   데이터스토어 맵 창이 열립니다.
2. 맵의 이름을 입력하십시오. 새 맵 이름 지정에 대한 권장사항은 93 페이지의 『데이터베이스 스키마 오브젝트 이름 지정 고려사항』을 참조하십시오.
3. EJB 그룹 및 스키마를 선택하십시오. 새 스키마 이름 지정에 대한 권장사항은 93 페이지의 『데이터베이스 스키마 오브젝트 이름 지정 고려사항』을 참조하십시오.
4. 데이터스토어 맵 분할창에서 맵을 선택하십시오.
5. 지속 클래스 분할창에서 클래스를 선택하십시오.
6. **테이블 맵** 메뉴에서 **새 테이블 맵 > 상속하지 않고 테이블 맵 추가**를 선택하십시오.
7. **테이블** 드롭 다운 목록에서 테이블을 선택하고 **확인**을 누르십시오.
8. 테이블 맵 패널에서 테이블 맵을 강조표시한 다음 마우스 오른쪽 버튼으로 누르고 **특성 맵 편집**을 선택하십시오.
   특성 맵 편집기가 열립니다.

9. 각 클래스 속성(bean의 CMP 필드)마다 맵 유형과 맵핑할 데이터베이스 열을 지정해야 합니다. 예를 들어 ″Simple″의 맵 유형을 사용하여 사용자 ID 클래스 속성을 데이터베이스 테이블의 USERID 열로 맵합니다.
10. 모든 필드가 해당 데이터베이스 열에 맵되면 스키마 맵을 저장해야 합니다. **데이터베이스 맵** 메뉴에서 **데이터스토어 맵 저장**을 선택하십시오. 맵을 저장하기 위해 해당 프로젝트, 패키지 및 클래스 이름을 입력하십시오. **완료**를 누르십시오.

액세스 bean 작성 및 전개 코드 생성: 액세스 bean은 다른 구성요소가 엔터프라이즈 bean과 상호 작용하는 방식을 단순화하는 엔터프라이즈 bean에 대한 랩퍼 역할을 합니다. 새 엔터프라이즈 bean의 액세스 bean을 작성해야 합니다.

전개 코드를 생성할 때 VisualAge for Java의 도구는 bean을 분석하여 Sun Microsystems EJB 스펙에 지정된 규칙뿐 아니라 EJB 서버 고유의 규칙이 충족되었는지 확인합니다.

새 엔터프라이즈 bean의 액세스 bean을 작성하려면 다음을 수행하십시오.

1. 엔터프라이즈 bean 분할창에서 새 엔터프라이즈 bean을 마우스 오른쪽 버튼으로 누른 후 **추가 > 액세스 Bean**을 선택하십시오(bean을 보려면 새 bean이 들어 있는 EJB 그룹을 먼저 확장해야 하는 수도 있습니다).
   액세스 Bean 작성 SmartGuide가 열립니다.
2. **EJB 그룹**, **엔터프라이즈 bean** 및 **액세스 bean 이름** 필드에 해당 EJB 그룹, bean 이름 및 액세스 bean 이름을 지정하십시오.
3. **액세스 bean 유형의 엔티티 bean용 복사 헬퍼**를 선택한 후 **다음**을 누르십시오.
4. **인수없는 구조체의 홈 메소드 선택** 드롭 다운 목록에서 **findByPrimaryKey**를 선택하십시오.
5. **변환기** 드롭 다운 목록에서 초기 특성에 대한 **WCSStringConverter**를 선택하고 **다음**을 누르십시오.
6. 복사 헬퍼 창의 bean 특성 선택 및 사용자 정의에서 각 필드에 대한 **WCSStringConverter**를 선택하십시오.
7. **완료**를 누르십시오.

전개 코드를 생성하려면 다음을 수행하십시오.

1. 엔터프라이즈 bean 분할창에서 새 엔터프라이즈 bean을 마우스 오른쪽 버튼으로 누른 후 **전개 코드 생성**을 선택하십시오.

이 도구를 사용하여 생성한 전개 코드는 EJB 1.0 스펙을 따르고 VisualAge for Java에서 엔터프라이즈 bean을 실행하는 경우에만 사용됨에 유의하십시오. WebSphere Application Server V4.0에서 실행 중인 WebSphere Commerce 응용프로그램에 엔터프라이즈 bean을 전개한 다음 단계에서 EJB 1.1 스펙에 상응하는 전개 코드가 있는 JAR 파일을 생성해야 합니다. EJB 1.1 Export JAR 파일 작성에 대한 추가 정보는 214 페이지의 『EJB 전개 코드에 대한 정보』 및 396 페이지의 『전개 코드 생성』을 참조하십시오.

엔터프라이즈 bean을 테스트하기 위해 테스트 클라이언트 사용: VisualAge for Java는 엔터프라이즈 bean을 테스트하기 위해 사용할 수 있는 테스트 클라이언트를 제공합니다. 새 bean을 테스트하기 위해 테스트 클라이언트를 사용하려면 다음을 수행하십시오.

1. 테스트 중인 엔터프라이즈 bean이 들어 있는 EJB 서버를 시작하십시오.
2. 엔터프라이즈 bean을 마우스 오른쪽 버튼으로 누른 후 **테스트 클라이언트 실행**을 선택하십시오.
   EJB 테스트 클라이언트 및 EJB 찾기 창이 열립니다.
3. **JNDI 이름** 필드에 엔터프라이즈 bean의 JNDI를 입력한 후 **찾아보기**를 누르십시오.
4. 인수가 채워진 **findByPrimaryKey** 메소드를 마우스 오른쪽 버튼으로 누른 후 **호출**을 선택하십시오.

코딩 실례: 다음 엔터프라이즈 bean 코딩 실례를 주시하십시오.

- BLOB 또는 CLOB 데이터 유형 중 하나를 사용하지 마십시오.
- LONG 데이터 유형(LONG VARCHAR이라고도 함)의 CMP 필드는 엔터프라이즈 bean의 CMP 필드 목록에서 첫 번째 또는 마지막 구성원이 될 수 없습니다. 목록을 확인하려면 `EJSJDBCPersister._hydrate()` 메소드를 확인하고 목록의 첫 번째 또는 마지막 요소가 LONG VARCHAR 유형인지 확인하십시오. 첫 번째 필드가 이러한 유형인 경우, 다음을 수행하십시오.
  1. 첫 번째 필드를 설정 해제하십시오. 이 필드를 `fieldA`라고 합니다.

2. LONG VARCHAR 유형이 *아닌* 또다른 필드를 설정 해제하십시오. 이 필드를 `fieldB`라고 합니다.
3. `fieldA`를 재설정하십시오.
4. `fieldB`를 재설정하십시오.
5. 스키마 맵 브라우저를 열고 테이블 맵을 편집하여 이들 변경사항을 반영하십시오. 맵을 저장하십시오.
6. 전개 코드를 다시 생성합니다.

- 엔터프라이즈 bean 코드는 엔터프라이즈 bean 패키지 외부의 어떤 항목도 참조하지 않습니다. 예를 들어 엔터프라이즈 bean 코드에서 명령 또는 데이터 bean을 참조하지 말아야 합니다
- 엔터프라이즈 bean의 액세스 제어를 사용하도록 설정하려면 com.ibm.commerce.security.Protectable 인터페이스 또는 com.ibm.commerce.security.Groupable 인터페이스를 엔터프라이즈 bean의 원격 인터페이스에 추가하십시오. 이러한 인터페이스를 추가한 후 bean의 전개 코드 및 액세스 bean을 다시 생성하십시오. 또한 objsrc 패키지에서 액세스 헬퍼 클래스 오브젝트를 작성해야 합니다.

단순 데이터 bean 작성

데이터 bean은 엔터프라이즈 bean에서 정보를 검색하기 위해 JSP 템플리트에서 사용되는 `bean`입니다. 단순 데이터 `bean`은 해당 액세스 `bean`을 확장하며 SmartDataBean 인터페이스를 구현합니다. 데이터 bean에 대한 대부분의 코드는 VisualAge for Java에 의해 자동으로 생성됩니다.

단순 데이터 bean을 작성하려면 다음 단계를 수행해야 합니다.

1. 프로젝트 및 패키지를 작성하여 데이터 bean 코드를 저장하십시오.
2. 해당 액세스 bean을 확장하고 해당 데이터 bean 인터페이스를 구현하는 데이터 bean을 작성하십시오.
3. 데이터 bean의 set 메소드를 작성하십시오.
4. 데이터 bean의 get 메소드를 작성하십시오.

데이터 bean 코드용 프로젝트 및 패키지 작성: 프로젝트 및 패키지를 작성하면 데이터 bean 코드가 저장될 수 있는 장소가 작성됩니다.

새 프로젝트를 작성하려면 다음을 수행하십시오.

1. **프로젝트** 탭을 선택하십시오.
2. **선택** 메뉴에서 **추가 > 프로젝트**를 선택하십시오.
   프로젝트 추가 SmartGuide가 열립니다.
3. **이름 지정된 새 프로젝트 작성**이 선택되었는지 확인하고 새 프로젝트에 대한 이름을 입력하십시오(예: `My Data Beans`).
4. **완료**를 누르십시오.

새 패키지를 작성하려면 다음을 수행하십시오.

1. 사용자 데이터 bean용으로 작성한 프로젝트를 마우스 오른쪽 버튼으로 누르고 **추가 > 패키지**를 선택하십시오. 예를 들어 **내 데이터 bean**을 마우스 오른쪽 버튼으로 누르고 **추가 > 패키지**를 선택하십시오.
   패키지 추가 SmartGuide가 열립니다.
2. **이름 지정된 새 패키지 작성**이 선택되었는지 확인하고 데이터 bean 패키지에 적합한 이름을 입력하십시오. 예를 들어, `com.mycompany.mydatabeans`를 입력하십시오.
3. **완료**를 누르십시오.

데이터 bean 작성: 데이터 bean은 페이지에 동적 컨텐츠를 제공하기 위해 JSP 템플리트 내에서 사용되는 Java bean으로서, 일반적으로 액세스 bean을 확장하여 엔티티 bean을 간소화(간접적)합니다. 데이터 bean은 엔티티 bean 내에서 검색되거나 설정될 수 있는 특성을 캡슐화합니다.

데이터 bean을 작성하려면 다음을 수행하십시오.

1. 데이터 bean을 저장할 패키지를 마우스 오른쪽 버튼으로 누른 후 **추가 > 클래스**를 선택하십시오.
   클래스 작성 SmartGuide가 열립니다.
2. 프로젝트 및 패키지 이름 필드가 이미 채워져 있습니다.
3. **새 클래스 작성**이 선택되었는지 확인한 후 **다음**을 누르십시오.
4. **클래스 이름** 필드에 새 데이터 bean의 이름을 입력하십시오. 예를 들어 UserResAccessBean을 확장하는 데이터 bean을 작성하려면 `UserResDataBean`을 입력하십시오.

5. 최상위 클래스를 지정하려면 **찾아보기**를 누른 다음, 패턴 필드에 해당 액세스 bean의 이름을 입력하십시오. 예를 들어 UserResAccessBean를 입력한 후 **확인**을 누르십시오.
6. **다음**을 누르십시오.
7. 데이터 bean이 구현해야 하는 인터페이스를 지정하려면 **추가**를 누르십시오. 인터페이스 창에서 다음을 수행하십시오.
   a. **패턴** 필드에서 com.ibm.commerce.beans.SmartDataBean을 입력한 다음 **추가**를 누르십시오.
   b. **패턴** 필드에서 com.ibm.commerce.beans.InputDataBean을 입력한 다음 **추가**를 누르십시오.
   c. **종료**를 누르십시오.
8. **완료**를 누르십시오.

필수 필드를 데이터 bean에 추가: 이 절에서는 필수 필드를 새 데이터 bean에 추가하는 방법을 설명합니다.

iCommandContext 필드를 추가하려면, 다음을 수행하십시오.

1. 새 데이터 bean(UserResDataBean)을 마우스 오른쪽 버튼으로 누르고 추가 > 필드를 선택하십시오.
   필드 작성 SmartGuide가 열립니다.
2. **필드 이름** 필드에 iCommandContext를 입력하십시오.
3. **찾아보기**를 눌러 필드 유형을 추가하고 com.ibm.commerce.command.CommandContext를 입력하십시오. **확인**을 누르십시오.
4. 액세스 수정자의 경우 **보호됨**을 선택하십시오.
5. **완료**를 누르십시오.

iRequestProperties 필드를 추가하려면 다음을 수행하십시오.

1. 새 데이터 bean(UserResDataBean)을 마우스 오른쪽 버튼으로 누르고 추가 > 필드를 선택하십시오.
   필드 작성 SmartGuide가 열립니다.
2. **필드 이름** 필드에 iRequestProperties를 입력하십시오.

3. **찾아보기**를 눌러 필드 유형을 추가하고 com.ibm.commerce.datatype.TypedProperty를 입력하십시오. **확인**을 누르십시오.
4. 액세스 수정자의 경우 **보호됨**을 선택하십시오.
5. **완료**를 누르십시오.

데이터 bean의 set 메소드 수정: 데이터 bean을 작성한 후에 생성된 set 메소드의 일부에서 코드를 수정해야 합니다.

set 메소드를 갱신하려면, 다음을 수행하십시오.

1. 새 데이터 bean을 펼쳐 관련 필드 및 메소드를 보십시오.
2. **setCommandContext(CommandContext)** 메소드를 선택하여 관련 소스 코드를 보십시오.
   소스 분할창은 다음과 같이 소스 코드를 표시합니다.

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

3. 다음과 같이 메소드가 표시되도록 소스 코드를 수정하십시오.

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

   작업을 저장하십시오(Ctrl+S).

4. **setRequestProperties(TypedProperty)** 메소드를 선택하여 소스 코드를 보십시오.
   소스 분할창은 다음과 같이 소스 코드를 표시합니다.

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

5. 해당 액세스 bean의 1차 키를 채울 소스 코드를 수정하려고 할 수 있습니다. 이를 수행하는 권장 방식은 이 값을 간접적으로 설정하기 위해 데이터 bean 관리자를 사용하는 것입니다. 이 간접 메소드는 URL 특성으로부터 얻은 1차 키 값이 이전에 설정된 1차 키를 대체하지 않도록 하기 위해 설계되었습니다. setRequestProperties 메소드가 이 모델을 따르게 하려면, 다음 코드와 유사한

형태로 코딩하십시오. 다음 예에서 1차 키는 사용자 ID임에 유의하십시오. 이것은 상황에 따라 다를 수 있습니다(다음 코드는 사용자 응용프로그램에서 바로 컴파일되지 않을 수 있습니다).

```
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) {}
   }
```

액세스 bean에 대한 1차 키가 설정될 수 있는 두 가지 다른 방식이 있습니다. 예를 들어, JSP 템플리트에서 데이터 bean으로부터 외부적으로 수행될 수 있습니다. 이 경우 JSP 템플리트에서 데이터 bean을 활성화하기 전에 명시적으로 1차 키에 대한 데이터 bean의 set 메소드를 호출하십시오. 예를 들어 JSP는 다음과 유사한 코드를 포함할 수 있습니다(여기서, db는 데이터 bean 오브젝트입니다).

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

또한 1차 키는 직접 방식으로 설정될 수 있습니다. 즉, JSP 템플리트는 db.activate 메소드만을 포함하며 데이터 bean 관리자는 명시적으로 1차 키를 액세스 bean에 설정합니다. 예를 들어 데이터 bean의 setRequestProperties 메소드에 대한 코드는 다음과 유사하게 표시됩니다.

```
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) {}
      }
```

1차 키 설정을 위해 권장되는 프로시저는 5단계에 표시된 간접 메소드입니다.

데이터 bean의 get 메소드 수정: 데이터 bean을 작성한 후에, 생성된 get 메소드의 일부에서 코드를 수정해야 합니다.

get 메소드를 갱신하려면 다음을 수행하십시오.

1. 새 데이터 bean을 펼쳐 관련 필드 및 메소드를 보십시오.
2. **getCommandContext()** 메소드를 선택하여 관련 소스 코드를 보십시오.
   소스 분할창은 다음과 같이 소스 코드를 표시합니다.

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

3. 다음과 같이 메소드가 표시되도록 소스 코드를 수정하십시오.

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

   작업을 저장하십시오(Ctrl+S).

4. **getRequestProperties()** 메소드를 선택하여 소스 코드를 보십시오.
   소스 분할창은 다음과 같이 소스 코드를 표시합니다.

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

5. 다음과 같이 표시되도록 소스 코드를 수정하십시오.

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

   작업을 저장하십시오(Ctrl+S).

populate() 메소드 수정: 다음을 수정하여 populate 메소드를 수정해야 합니다.

1. 새 데이터 bean을 펼쳐 관련 필드 및 메소드를 보십시오.
2. **populate()** 메소드를 선택하여 관련 소스 코드를 보십시오.
   소스 분할창은 다음과 같이 소스 코드를 표시합니다.

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

3. 다음과 같이 메소드가 표시되도록 소스 코드를 수정하십시오.

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

   작업을 저장하십시오(Ctrl+S).

새 세션 bean 작성

새 세션 bean을 작성할 때 WebSphere Commerce EJB 그룹과는 별도의 EJB 그룹에서 세션 bean을 작성해야 합니다. 이러한 새 EJB 그룹을 WebSphere Commerce 프로젝트와는 별도의 새 프로젝트에 저장하십시오. 예를 들어 com.mycompany.mycustomcode 패키지 내에서 MyCustomBeans EJB 그룹을 작성할 수 있습니다. WebSphere Commerce 코드에서 사용자 정의 코드를 분리함으로써 향후 릴리스로 이주할 때의 영향을 최소화하게 됩니다

새 세션 bean은 com.ibm.commerce.base.helpers.BaseJDBCHelper 클래스를 확장해야 합니다. 최상위 클래스는 세션 bean이 다른 엔티티 bean과 동일한 트랜잭션에 참여하도록 commerce 서버에서 사용되는 데이터 소스 오브젝트로부터 JDBC 연결 오브젝트를 확보할 수 있도록 하는 메소드를 제공합니다. 다음은 최상위 클래스에서 제공되는 함수를 보여주는 코드 예입니다.

```
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 ---          //
       ////////////////////////////////////////////////////
```

```
    }
}
```

앞의 코드 예에서 executeQuery 메소드는 두 개의 입력 매개변수를 사용합니다. 첫 번째는 준비된 명령문이고 두 번째는 캐시 비우기 조작과 관련된 부울 플래그입니다. 조회를 실행하기 전에 현재 트랜잭션의 모든 항목 오브젝트를 캐시에서 비우기 위해 컨테이너가 필요한 경우, 해당 플래그를 true로 설정하십시오. 이는 일부 엔티티 오브젝트에 갱신을 수행하고 이 갱신 오브젝트를 검색하기 위해 조회가 필요한 경우 필요합니다. 플래그를 false로 설정하면 트랜잭션 끝까지 해당 엔티티 오브젝트 갱신이 데이터베이스에 작성되지 않습니다.

정말로 필요한 경우를 제외하고는, 이 비우기 조작 사용을 제한하고 일반적으로 플래그를 false로 설정해야 합니다. 비우기 조작은 자원 집약 조작입니다.

오브젝트 사용 주기

오브젝트 모델의 엔터프라이즈 bean에는 *독립* 오브젝트와 *종속* 오브젝트가 모두 들어 있습니다. 독립 오브젝트에는 오브젝트를 호출하는 비즈니스 로직의 작성 또는 제거 요청에 의해 직접 제어되는 자체 사용 주기가 있습니다. 종속 오브젝트에는 *소유자 오브젝트*라는 다른 오브젝트에 접속된 사용 주기가 있습니다(차례로 종속 오브젝트가 될 수도 있지만, 연관 계층을 따라가면 독립 오브젝트가 존재합니다). 소유자 오브젝트가 삭제되면 모든 종속 오브젝트 또한 삭제됩니다. 실제 삭제는 데이터베이스 내의 연속 삭제 스펙에 의해 제어됩니다.

예를 들어, 주소록 오브젝트 및 주문 목록 오브젝트를 리턴하는 사용자 오브젝트가 제공되는 경우, 사용자 오브젝트가 삭제되면 해당 주소록 오브젝트도 삭제되며(사용자가 주소록을 소유하므로) 주소록 내의 모든 주소 오브젝트도 삭제됩니다(주소록에 주소가 있으므로). 그러나 주문 소유자는 사용자 오브젝트가 아니라 상점 오브젝트이므로 주문 오브젝트가 삭제되지 않습니다.

종속 오브젝트 작성에는 특정 설계 패턴이 사용됩니다. 종속 오브젝트의 create 메소드는 해당 소유자 오브젝트에 대한 참조를 제공해야 하므로 종속 오브젝트를 작성하려면 소유자 오브젝트가 존재해야 합니다.

트랜잭션

Enterprise JavaBeans V1.0 아키텍처는 인스턴스 상태에 관해 세 가지 대체 확약 시간 옵션을 지정합니다. 이러한 옵션은 스펙 문서에서 옵션 A, B 및 C로 설명됩니다. 옵션에 대한 자세한 내용은 Sun Microystem Enterprise JavaBean V1.0 스펙 문서를 참조하십시오.

WebSphere Application Server에서 옵션 A 및 C를 구현하더라도, 옵션 A는 데이터베이스가 공유되지 않는 것으로 가정합니다.

옵션 C에서 엔터프라이즈 bean 컨테이너는 트랜잭션 간의 "준비" 인스턴스를 캐시하지 않습니다. 트랜잭션이 완료되자마자 인스턴스는 사용 가능한 인스턴스의 풀로 리턴됩니다. WebSphere Commerce에서는 데이터베이스가 복수의 WebSphere Commerce 응용프로그램에서 공유되므로 옵션 C를 사용합니다. 이 구현에서 컨테이너는 각 트랜잭션 시작 시 엔티티 bean의 지속 데이터를 로드하며, 엔티티 bean은 트랜잭션 지속 기간 동안에만 캐시됩니다. 컨테이너는 엔티티가 액세스되고 있는 트랜잭션마다 하나씩 엔티티 bean의 복수 인스턴스를 활성화합니다. 트랜잭션 동기화는 데이터베이스에서 수행됩니다.

각 엔터프라이즈 bean의 트랜잭션 속성은 `TX_REQUIRED`로 설정됩니다. 웹 제어기는 엔터프라이즈 bean에 액세스하는 명령을 실행하기 전에 트랜잭션을 시작하므로(해당 액세스 bean을 통해) 엔터프라이즈 bean의 비즈니스 메소드는 이 트랜잭션 컨텍스트 내에서 호출됩니다.

엔티티 bean의 기타 고려사항

Find for Update

행을 갱신하기 위해 여러 응용프로그램이 동일한 데이터베이스 행에 액세스하는 상황을 *동시 갱신*이라고 합니다. 동시 갱신이 허용되는 상황이 있고, 바람직하지 않은 상황이 있습니다.

데이터베이스 갱신이 덮어쓰기 방식이어서 새 값이 데이터베이스의 현재 값과 관련이 없는 경우, 동시 갱신이 허용됩니다. 동시 갱신이 허용되며 여러 응용프로그램이 동일한 데이터베이스 행을 갱신하려고 하는 경우, 마지막 시도가 데이터베이스를 갱신하는 시도입니다.

데이터베이스 갱신이 데이터베이스의 현재 값에 따라 다를 경우, 동시 갱신은 바람직하지 않습니다. 예를 들어 응용프로그램이 상품 재고를 갱신하는 경우, 한 번에 한 응용프로그램만 재고를 갱신할 수 있습니다.

동시 갱신이 허용되는지 여부에 영향을 주는 요소에는 데이터베이스 잠금 및 엔터프라이즈 bean 분리 레벨이 있습니다.

두 번째 응용프로그램이 행을 동시에 갱신하지 못하도록 하려면 행에 액세스하는 첫 번째 응용프로그램은 “find for update” 옵션을 사용하여 행을 가져와야 합니다. “find for update” 옵션을 사용한 경우 *쓰기 잠금*(배타적 잠금이라고도 함)이 적용됩니다. 이러한 쓰기 잠금이 행에 적용될 경우 “find for update”를 사용하여 행에 액세스하려 하는 모든 응용프로그램이 블록화됩니다.

응용프로그램이 동시 갱신을 허용할 경우, 해당 응용프로그램은 행을 잠그지 않고 데이터를 가져올 수만 있습니다.

UpdateInventory에서 주문에 포함된 모든 상품을 찾아 이에 따라 재고를 갱신해야 하는 OrderProcess 시나리오를 고려해 보십시오. 동일한 상품이 다수의 다른 주문에 포함될 수 있으므로 *find for update*가 사용되어야 하며 교착 상태 가능성을 줄이기 위해 트랜잭션 범위 내에서 가능한 한 빨리 사용되어야 합니다. 따라서 UpdateInventory 알고리즘은 다음 의사 코드로 표시될 수 있습니다.

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

주문에 많은 항목이 있을 수도 있는 장기 B2B 시나리오에서 find for update는 가능한 한 빨리 사용되어야 합니다. 로직은 다음과 같이 될 수 있습니다.

```
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
```

Flush remote 메소드

WebSphere Application Server는 트랜잭션 확약 시간까지 데이터베이스에 엔티티 bean의 변경사항을 기록하지 못하므로 데이터베이스는 엔티티 bean의 컨테이너에서 캐시된 데이터와의 동기화 상태를 일시적으로 벗어날 수 있습니다.

모든 트랜잭션(즉, 엔터프라이즈 bean 캐시에서 정보를 가져옴)의 확약된 모든 변경사항을 기록하고 데이터베이스를 갱신하는 flush remote 메소드가 com.ibm.commerce.base.helpers.BaseJDBCHelper 클래스에서 제공됩니다. 이 remote 메소드는 명령을 통해 호출할 수 있습니다. 오버헤드가 많이 발생하여 성능에 부정적 영향을 줄 수 있으므로, 반드시 필요할 때만 이 메소드를 사용하십시오.

다음과 같은 코드 부분이 있는 로그온 명령을 고려해 보십시오.

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

트랜잭션이 확약되기 전에 USER 테이블의 REGISTEREDSTAMP가 현재 시간소인으로 갱신되지 않았습니다. 트랜잭션 확약 시간에만 갱신이 발생합니다. *select from user where registeredstamp ...*와 같은 직접 JDBC 조회(동일한 트랜잭션에서)가 지정된 등록 시간소인으로 사용자를 리턴하도록 flush 메소드가 사용되어야 합니다.

엔터프라이즈 bean 보안

WebSphere Application Server를 사용하여 엔터프라이즈 bean의 보안을 설정하려는 경우, WebSphere Application Server 관리 콘솔을 사용하여 모든 새 엔터프라이즈 bean의 메소드를 WCSMethodGroup 보안 메소드 그룹에 지정해야 합니다. 엔터프라이즈 bean을 전개할 때는 이 단계를 수행하십시오. 또한 기존의 WebSphere Commerce 엔티티 bean을 수정할 경우, 영향을 받는 EJB 그룹에 있는 모든 엔티티 bean의 메소드를 WCSMethodGroup 보안 메소드 그룹에 지정해야 합니다. 사용자 정의 코드의 전개 처리에 대한 설명은 214 페이지의 『코드 전개』를 참조하십시오.

1차 키

1차 키는 테이블의 정의 부분인 고유 키입니다. 레코드를 서로 구별하기 위해 사용될 수 있습니다. 모든 레코드는 1차 키를 가져야 합니다. 테이블에서 새 레코드를 작성할 때 레코드에 대한 고유 1차 키를 생성해야 합니다.

WebSphere Commerce 프로그래밍 모델에서 지속 계층에는 데이터베이스와 상호작용하는 엔티티 bean이 들어 있습니다. 이와 같이 엔티티 bean의 인스턴스가 생성될 때는 데이터베이스 레코드를 작성할 수 있습니다. 따라서 엔티티 bean의 인스턴스 생성을 위한 ejbCreate 메소드에는 새 레코드의 1차 키를 생성하기 위한 로직이 들어 있어야 합니다.

응용프로그램에 데이터베이스 정보가 필요할 경우, 해당 응용프로그램은 bean의 해당 액세스 bean 인스턴스를 생성하고 여러 필드를 가져오거나 설정하여 간접적으로 엔티티 bean을 사용합니다. 데이터베이스의 특정 레코드에 대해(예를 들어, 특정 사용자 프로파일에 대해) 액세스 bean의 인스턴스가 생성되며, 그 액세스 bean은 1차 키를 사용하여 데이터베이스에서 올바른 정보를 선택합니다.

다음 절에서는 고유 1차 키를 작성하는 방법과 1차 키를 선택하는 방법에 대해 설명합니다.

1차 키 작성: `ejbCreate` 메소드를 사용하면 엔티티 bean의 새 인스턴스를 생성할 수 있습니다. 이 메소드는 자동으로 생성되지만 생성된 메소드는 정적 값으로 1차 키를 초기화하는 로직만을 포함합니다.

1차 키가 새 고유한 값임을 확인하는 것이 필요할 수 있습니다. 이 경우, ejbCreate 메소드는 다음 코드 부분과 유사합니다.

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

앞의 코드 부분에서 getNextKey 메소드는 1차 키용 고유 정수를 생성합니다. 메소드에 대한 *table_name* 입력 매개변수는 KEYS 값에 정의된 TABLENAME 값과 정확히 일치해야 합니다. 문자 및 대소문자가 정확히 일치하는지 확인하십시오.

ejbCreate 메소드에 선행 코드를 포함하는 것 이외에, KEYS 테이블에 항목을 작성해야 합니다. 다음은 KEYS 테이블에 항목을 작성하는 SQL 문의 예입니다.

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

선행 SQL 문에서 KEYS 테이블에서 다른 열에 대한 기본값이 승인됨에 유의하십시오. COUNTER의 값은 계수가 시작되어야 하는 값을 표시합니다. KEYS_ID에 대한 값은 양의 값이어야 합니다.

1차 키를 long 데이터 유형으로 정의한 경우(DB2의 경우 BIGINT 또는 Oracle의 경우 NUMBER) getNextKeyAsLong 메소드를 사용하십시오.

1차 키에 의해 선택: 액세스 bean 내에서, 1차 키를 사용하여 해당 데이터베이스 레코드를 선택해야 합니다. 다음 코드 부분은 이 선택을 수행하는 방법을 보여줍니다. 또한 다음에 설명할 추가 로직을 포함합니다.

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

앞의 코드 부분에서 첫 번째 행은 ″abUserProfile″이라는 새 UserProfileAccessBean의 인스턴스를 생성합니다. 두 번째 행은 액세스 bean에 1차 키를 설정합니다. VisualAge for Java는 setInitKey_*xxx*(여기서 *xxx*는 1차 키 필드 이름) 이름 지정 규칙을 사용하여 1차 키용 set 메소드의 이름을 지정합니다. 액세스 bean의 인스턴스를 생성할 때는 refreshCopyHelper 메소드를 사용하기 전에 setInitKey_xxx 메소드가 설정하는 모든 필드를 초기화해야 합니다. setInitKey_xxx 메소드가 호출되는 순서는 중요하지 않습니다.

모든 setInitKey_xxx 메소드가 호출되면 필요한 모든 필드가 초기화된 것이며 refreshCopyHelper 메소드를 사용하여 데이터베이스에서 정보를 검색할 수 있습니다.

액세스 bean의 로컬 캐시에서 값을 갱신할 경우, commitCopyHelper도 호출하여 데이터베이스를 갱신된 정보로 갱신해야 합니다. 예를 들어 refreshCopyHelper 메소드를 사용하여 데이터를 검색한 후 고객의 이름을 갱신(이름값을 설정하여)하는 경우, `abUserProfile.commitCopyHelper()`를 호출하여 새 정보로 데이터베이스를 갱신해야 합니다.

엔티티 bean 사용

엔터프라이즈 bean을 사용하는 프로그램은 JNDI(Java Naming and Directory Interface)와 엔터프라이즈 bean의 홈 및 원격 인터페이스를 처리해야 합니다. 프로그래밍 모델을 단순화하기 위해 각 엔터프라이즈 bean의 액세스 bean이 생성됩니다. 고유의 엔터프라이즈 bean 작성시, VisualAge for Java의 도구를 사용하여 이 액세스 bean을 생성하십시오.

WebSphere Commerce 명령은 엔티티 bean과 직접 상호 작용하지 않고 액세스 bean과 상호 작용합니다. 도표에 설명된 대로 액세스 bean을 사용하면 다음과 같은 이점이 있습니다.

- 프로그래밍 인터페이스가 보다 간편해집니다. 액세스 bean은 Java bean과 같이 활동하며 JNDI, 클라이언트의 홈 및 원격 인터페이스와 같은 모든 엔터프라이즈 bean의 고유 프로그래밍 인터페이스를 숨깁니다.
- 런타임 액세스 bean은 홈 오브젝트 찾아보기에 시간과 자원이 많이 소요되므로 엔터프라이즈 bean 홈 오브젝트를 캐시합니다.
- 액세스 bean은 명령이 엔터프라이즈 bean 속성을 가져와 설정할 때 엔터프라이즈 bean 호출 수를 줄이는 copyHelper 오브젝트를 구현합니다. 따라서 복수 엔터프라이즈 bean 속성을 읽거나 기록할 때 엔터프라이즈 bean을 한 번만 호출해도 됩니다.

다음 도표는 명령, 액세스 bean, 엔티티 bean과 데이터베이스 간의 상호 작용을 표시합니다.

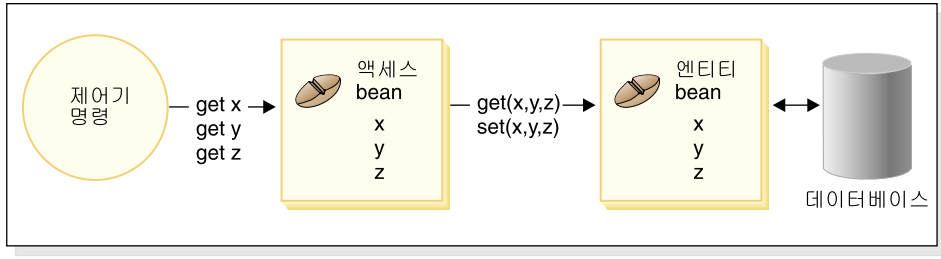


그림 18.

데이터베이스 고려사항

전자 상거래 응용프로그램을 사용자 정의할 때는 데이터베이스 테이블을 작성할 수 있습니다. 이 테이블을 작성할 때는 정해진 규칙을 사용하여 테이블이 WebSphere Commerce 테이블과 일관된 형태로 작성되도록 하는 것이 좋습니다.

데이터베이스 스키마 오브젝트 이름 지정 고려사항

다음 절에서는 데이터베이스 스키마 오브젝트의 이름을 지정하기 위한 지침을 제공합니다.

테이블 및 보기의 이름 지정 규칙

다음 목록은 새 테이블 및 보기의 이름을 지정하기 위한 지침을 제공합니다.

- 향후 릴리스에서의 WebSphere Commerce 테이블 및 보기와 이름 충돌(이름 중복)을 방지하기 위해 테이블 이름 또는 보기 이름의 첫 자를 X로 지정해야 합니다. 예를 들면 XMYTABLE과 같습니다.
- 테이블 이름 또는 보기 이름의 길이는 10자 미만이어야 합니다. 원하는 이름이 한도를 초과할 경우, 10자가 남을 때까지 이름의 끝에서 모음을 제거하여 길이를 줄이십시오.
- 테이블 또는 보기 이름에 “_”, “+”, “$”, “%”와 같은 특수 문자나 공백이 들어 있으면 안됩니다.
- 데이터베이스 예약어를 테이블 이름 또는 보기 이름으로 사용하지 마십시오.
- 보기 이름은 VW로 끝나야 합니다.

- 테이블 및 보기 이름은 단수 명사이어야 합니다.

열의 이름 지정 규칙

다음 목록은 테이블에서 열 이름을 지정하기 위한 지침을 제공합니다.

- 향후 릴리스에서의 WebSphere Commerce 테이블의 열과 이름 충돌(이름 중복)을 방지하기 위해 열 이름의 첫 자를 X로 지정해야 합니다. 예를 들면 XMYCOLUMN과 같습니다.
- 열 이름의 길이는 18자 미만이어야 합니다. 원하는 이름이 한도를 초과할 경우, 18자가 될 때까지 이름의 끝에서 모음을 제거하여 길이를 줄이십시오.
- 열 이름(foreign key가 아님)에 “_”, “+”, “$”, “%”와 같은 특수 문자나 공백이 들어 있으면 안됩니다.
- 데이터베이스 예약어를 열 이름으로 사용하지 마십시오.
- 결합된 단어는 활성화 음성 조합을 사용하여 열 이름으로 사용될 수 있습니다. 예를 들면 COMBINERESULT와 같습니다.
- 생성된 1차 키 열의 이름은 *table*_id이어야 합니다. 예를 들어 USERS 테이블의 1차 키는 USERS_ID입니다.
- 생성된 foreign key 열 이름을 변경해서는 안됩니다.
- 향후 사용자 정의를 위해 어떤 열을 예약할 경우, 그 이름은 field*x*이어야 합니다. 여기서, *x*는 1부터 시작하는 숫자입니다.

색인의 이름 지정 규칙

다음 목록은 테이블에서 색인 이름을 지정하기 위한 지침을 제공합니다.

- 색인 이름의 길이는 18자 미만이어야 합니다.
- 색인 이름에 공백이 들어 있으면 안됩니다.
- 색인 이름에 데이터베이스 예약어가 들어 있으면 안됩니다.
- 고유하지 않은 색인의 이름은 I_*tablex*로 지정해야 합니다. 여기서, *table*은 테이블의 이름이고, *x*는 1부터 시작하는 숫자입니다. 예를 들어, USERS 테이블의 고유하지 않은 색인은 I_USERS1입니다.
- 고유 색인의 이름은 UI_*tablex*로 지정해야 합니다. 여기서 *table*은 테이블의 이름이고, *x*는 1부터 시작하는 숫자입니다. 예를 들어 USERS 테이블의 고유 색인은 UI_USERS1입니다.

- 총 색인 크기는 254바이트 미만이어야 합니다.
- 색인 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

1차 키의 이름 지정 규칙

다음 목록은 테이블에서 1차 키 이름을 지정하기 위한 지침을 제공합니다.

- 1차 키 이름의 길이는 18자 미만이어야 합니다.
- 1차 키 이름에 공백이 들어 있으면 안됩니다.
- 1차 키 이름에 데이터베이스 예약어가 들어 있으면 안됩니다.
- 1차 키의 이름은 P_*table*로 지정해야 합니다. 여기서 *table*은 테이블의 이름입니다. 예를 들어 USERS 테이블의 1차 키는 P_USERS입니다.
- 1차 키 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

foreign key의 이름 지정 규칙

다음 목록은 테이블에서 foreign key 이름을 지정하기 위한 지침을 제공합니다.

- foreign key 이름의 길이는 18자 미만이어야 합니다.
- foreign key 이름에 공백이 들어 있으면 안됩니다.
- foreign key 이름에 데이터베이스 예약어가 들어 있으면 안됩니다.
- foreign 키의 이름은 F_*table*로 지정해야 합니다. 여기서, *table*은 테이블의 이름입니다. 예를 들어 USERS 테이블의 foreign key는 F_USERS1입니다.
- foreign key 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

데이터베이스 트리거의 이름 지정 규칙

다음 목록은 데이터베이스 트리거의 이름 지정을 위한 지침을 제공합니다.

- 데이터베이스 트리거 이름의 길이는 18자 미만이어야 합니다.
- 데이터베이스 트리거 이름에 공백이 들어 있으면 안됩니다.
- 데이터베이스 트리거 이름에 데이터베이스 예약어가 들어 있으면 안됩니다.
- 데이터베이스 트리거의 이름은 T_*table*로 지정해야 합니다. 여기서 *table*은 테이블의 이름입니다. 예를 들어 USERS 테이블의 foreign key는 T_USERS1입니다.
- 데이터베이스 트리거 이름은 전체 데이터베이스 스키마에서 고유해야 합니다.

데이터베이스 열 데이터 유형 고려사항

이 절에서는 새 테이블을 작성할 때 사용할 수 있는 열 데이터 유형을 소개합니다. 여러 데이터 유형 설명에는 DB2 용어가 사용됩니다. 98 페이지의 『데이터베이스 간의 데이터 유형 차이점』에서는 다른 데이터베이스를 사용하는 경우의 차이점에 대해 설명합니다.

BIGINT

9223372036854775807 - 92233720368547758070의 64비트로 이루어진 정수입니다. INTEGER는 이와 대조적으로 BIGINT의 절반 크기입니다.

INTEGER

2147483647 - 21474836470의 32비트로 이루어진 부호가 있는 정수입니다. 일반적으로 INTEGER가 BIGINT 대신 기본 유한 숫자 데이터 유형이어야 합니다. BIGINT를 사용하는 확고한 비즈니스 이유가 있는 경우를 제외하고는 성능상의 이유로 숫자 데이터 유형으로 INTEGER를 사용하는 것이 더 낫습니다. BIGINT 데이터 유형의 일반 용도는 시스템 생성 키입니다.

SMALLINT 또는 SHORT 데이터 유형은 사용하지 않는 것이 좋습니다. 이러한 데이터 유형은 비 오브젝트 Java 데이터 유형에 맵핑되고 이 오브젝트 데이터 유형이 아닌 것은 일부 엔터프라이즈 bean 오브젝트의 인스턴스 생성 시 문제를 일으키게 됩니다.

TIMESTAMP

이 값은 시간이 마이크로초의 소수 스펙을 포함한다는 점을 제외하고 날짜와 시간을 지정하는 7부분으로 구성된 값(연, 월, 일, 시간, 분, 초 및 마이크로초)입니다. 시간소인의 내부 표시는 10바이트의 문자열로서, 각각은 2자리의 10진수로 구성됩니다. 처음의 4바이트는 날짜를 나타내고 다음 3바이트는 시간을, 마지막 3바이트는 마이크로초를 나타냅니다.

CHAR

1-254자로 이루어진 INTEGER 길이의 고정 길이 문자열입니다. 길이 스펙을 생략할 경우, 길이가 1자로 간주됩니다. CHAR가 고정 길이 데이터베이스 열이기 때문에, 사용되지 않은 모든 후속 문자 공백은 빈 공백으로 바뀝니다. 성능상의 이유가 아니라면 CHAR는 유동적이지 않아서 나중에 길이를 변경할 수 없기 때문에 CHAR 데이터 유형을 사용하는 것

은 바람직하지 않습니다. 대략적으로 볼 때, 문자열의 열 길이가 64자보다 짧으며 정기적으로 검색되거나 갱신되는 경우, 보다 나은 성능을 위해 CHAR를 사용하십시오.

VARCHAR

1-254자로 이루어진 최대 길이 정수의 가변 길이 문자열 정수입니다. 그러나 열 데이터가 테이블과 함께 저장되는 CHAR과는 달리 VARCHAR은 내부적으로 데이터베이스 페이지 내의 참조 포인터로 표시됩니다. 따라서 작성 후 VARCHAR 열의 길이는 언제든지 변경될 수 있습니다.

LONG VARCHAR

동일한 데이터베이스 페이지 내에서 VARCHAR을 작성할 수 없을 경우 사용할 수 있는 가변 길이 문자열입니다. LONG VARCHAR은 여러 데이터베이스 페이지에 걸쳐 나타날 수 있다는 점을 제외하고 VARCHAR와 매우 유사합니다. 절대적으로 필요한 경우에만 LONG VARCHAR 데이터 유형을 사용하도록 하십시오. LONG VARCHAR 오브젝트에는 일반적으로 성능을 내기 위한 자원이 많이 소요됩니다.

CLOB

열이 LONG VARCHAR의 32KB 제한을 초과해야 할 경우 사용할 수 있는 또 다른 가변 길이 문자열입니다. CLOB 오브젝트의 길이는 데이터베이스 구성을 수정하지 않고 1GB에 도달할 수 있습니다. 다양한 시스템 간에 이동할 때는 CLOB로 저장된 텍스트 데이터가 적절히 변환됩니다.

BLOB

구성되지 않은 데이터를 데이터베이스에 저장하는 가변 길이 2진 문자열입니다. BLOB 오브젝트는 최대 4GB의 2진 데이터를 저장할 수 있습니다. 일반적으로, 꼭 필요한 경우가 아니라면 BLOB를 열 데이터 유형으로 사용하지 말아야 합니다. 성능면에서 BLOB 오브젝트는 모든 데이터베이스 중에서 가장 자원이 많이 소요되는 오브젝트 중 하나로 간주됩니다.

DECIMAL(20,5)

이 데이터 유형은 통화 단위와 같은 소수점 숫자용으로 사용되도록 특별히 정의되었습니다. 다른 부동 소수점 10진수의 경우, FLOAT를 사용할 수 있습니다.

데이터베이스 간의 데이터 유형 차이점

다음 표는 WebSphere Commerce 데이터베이스 스키마에서 사용하는 데이터 유형을 나열하고 다른 데이터베이스 구현을 위한 해당 데이터 유형을 표시합니다.

JDBC 오브젝트	Windows AIX Solaris Linux **DB2**	Windows AIX Solaris **Oracle**	400 **DB2**
해시 테이블	BLOB()	BLOB	BLOB()
시간소인	TIMESTAMP	DATE	TIMESTAMP
정수	INTEGER	INTEGER	INTEGER
BigDecimal	DECIMAL(,)	DECIMAL(,)	DECIMAL(,)
Long	BIGINT	NUMBER	BIGINT
배수	FLOAT	NUMBER	FLOAT
문자열	CHAR()	VARCHAR2()	GRAPHIC() CCSID 13488
바이트[]	비트 데이터용 CHAR()	RAW()	비트 데이터용 CHAR()
문자열	VARCHAR()	VARCHAR2()	VARGRAPHIC() CCSID 13488
문자열	LONG VARCHAR	VARCHAR2()(자세한 내용은 다음 표의 주를 참조하십시오.)	VARGRAPHIC(4000) ALLOCATE() CCSID 13488
바이트[]	비트 데이터용 LONG VARCHAR	LONG RAW	비트 데이터용 VARCHAR(8000) ALLOCATE()
문자열	CLOB()	CLOB()	DBCLOB() CCSID 13488

주:

일정하지 않은 성공률로 인해 Oracle JDBC 드라이버가 LONG 데이터 유형의 정보를 핸들할 때 가능하면 항상 LONG 데이터 유형을 사용하지 않도록 하는 것이 좋습니다. 이 상황에서 가장 일반적으로 보고되는 오류는 “스트림이 이미 닫혔습니다.”입니다.

이 데이터 유형을 사용해야 할 경우, LONG 유형을 사용하는 데이터베이스 테이블마다 하나의 열만 가질 수 있습니다. 또한 select 명령문을 작성할 때, select의 첫 번째 또는 마지막 요소로 LONG 열을 넣지 마십시오. 과부하 상태에서 조작에 대한 또다른 해결책은 이 특정 열을 엔티티 bean의 CMP 필드에 맵핑하지 않는 것입니다. 대신 세션 bean을 사용하여 이 열에 대한 검색 및 갱신을 수행하십시오.

제 4 장 액세스 제어

액세스 제어 이해

WebSphere Commerce 응용프로그램의 액세스 제어 모델에는 사용자, 조치 및 자원의 세 가지 기본 개념이 있습니다. 사용자는 시스템을 사용하는 사람입니다. 자원은 응용프로그램에 의해 유지 보수되는 엔티티입니다(예: 상품, 문화, 자원). 사람을 나타내는 사용자 프로파일도 자원입니다. 조치는 사용자가 자원에 대해 수행하는 활동입니다. 액세스 제어는 해당 사용자가 해당 자원에 대해 해당 조치를 수행할 수 있는지 여부를 결정하는 전자 상거래 응용프로그램 구성요소입니다.

WebSphere Commerce 응용프로그램에는 두 가지 주요 레벨의 액세스 제어가 있습니다. 첫 번째 액세스 제어 레벨은 WebSphere Application Server에 의해 수행됩니다. 따라서 WebSphere Commerce는 WebSphere Application Server를 사용하여 엔터프라이즈 bean 및 Servlet을 보호합니다. 액세스 제어의 두 번째 레벨은 WebSphere Commerce의 정교한 액세스 제어 시스템입니다.

WebSphere Commerce 액세스 제어 프레임워크는 액세스 제어 정책을 사용하여 해당 사용자가 해당 자원에 대해 해당 조치를 수행할 권한이 있는지 판별합니다. 이 액세스 제어 프레임워크는 정교한 액세스 제어를 제공합니다. WebSphere Application Server에서 제공하는 액세스 제어를 바꾸지 않고 결합하여 작업합니다.

WebSphere Application Server에서의 자원 보호 개요

WebSphere Application Server가 다음 WebSphere Commerce 자원을 액세스 제어하에서 보호합니다

- 엔티티 bean
  전자 상거래 응용프로그램에서 이러한 bean은 오브젝트를 모델링합니다. 이 bean은 원격 클라이언트에서 액세스할 수 있는 분배 오브젝트입니다.
- JSP 템플리트
  WebSphere Commerce는 표시 페이지에 JSP 템플리트를 사용합니다. 각 JSP

템플리트에는 엔티티 bean에서 데이터를 검색하는 하나 이상의 데이터 bean이 포함될 수 있습니다. 클라이언트는 URL 요청을 작성하여 JSP 페이지를 요청할 수 있습니다.

- 제어기 명령 및 보기 명령
  클라이언트는 URL 요청을 작성하여 제어기 명령 및 보기 명령을 요청할 수 있습니다. 또한 하나의 표시 페이지에는 VIEWREG 테이블에 등록된 대로 JSP 파일 이름이나 보기 이름을 사용한 또다른 표시 페이지로의 링크가 들어 있을 수 있습니다.

일반적으로 WebSphere Commerce Server는 다음 웹 경로를 사용하도록 구성됩니다.

- `/webapp/wcs/stores/servlet/*`
  요청 Servlet에 대한 요청에 사용됩니다.
- `/webapp/wcs/stores/*.jsp`
  JSP Servlet에 대한 요청에 사용됩니다.

다음 도표는 이전 웹 경로 구성의 경우 WebSphere Commerce 자원에 액세스하기 위해 요청이 따를 수 있는 라우트를 보여줍니다.

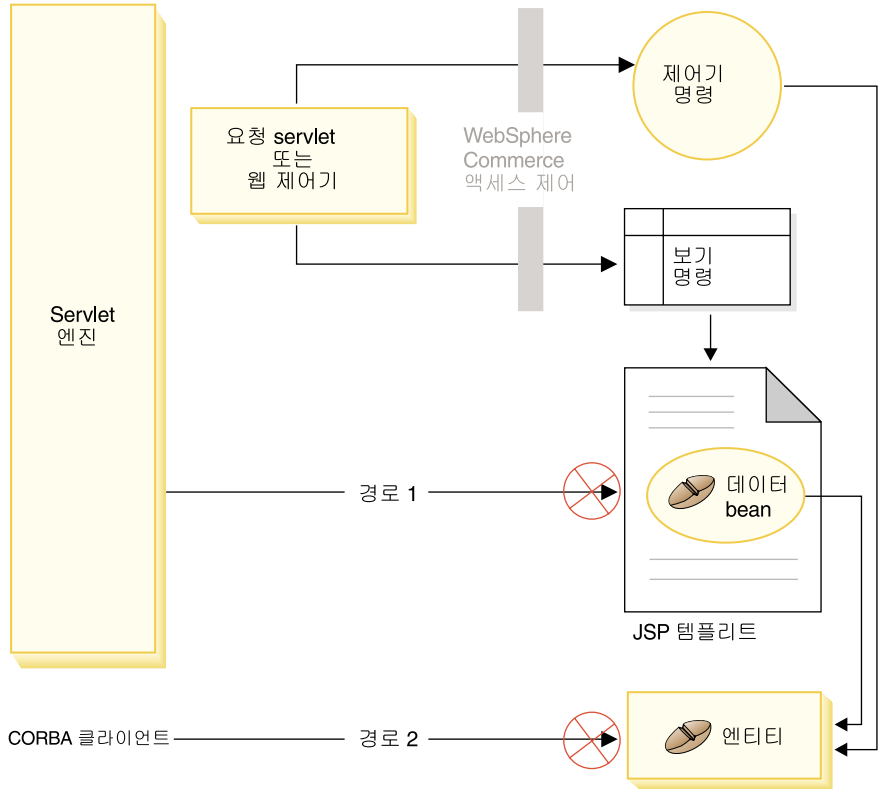


그림 19.

유효한 모든 요청은 요청 Servlet으로 지정되어야 하며 이들은 웹 제어기로 지정됩니다. 웹 제어기는 제어기 명령 및 보기에 대한 액세스 제어를 구현합니다. 그러나 위에 표시된 웹 경로는 권한없는 사용자가 JSP 템플리트(경로 1) 및 엔티티 bean(경로 2)에 직접 액세스할 수 있게 만듭니다. 이러한 고의적인 공격을 피하기 위해서는 런타임시 경로가 거부되어야 합니다.

JSP 템플리트 및 엔티티 bean에 대한 직접 액세스는 다음 접근 방법 중 하나를 사용하여 방지할 수 있습니다.

WebSphere Application Server 보안

WebSphere Application Server에서는 보안 기능을 제공합니다. 이러한 접근 방법을 사용하여 모든 엔터프라이즈 bean 메소드 및 JSP 템플리트는 시스템 ID만으로 호출되도록 구성됩니다. 이러한 WebSphere Commerce 자원에 액세스하려면 URL 요청은 시스템 신원을 웹 제어기로 전달하기

전에 현재 스레드로 설정하는 요청 Servlet으로 경로 지정되어야 합니다. 그런 후 웹 제어기에서는 해당 제어기 명령 또는 보기로 요청을 전달하기 전에 호출자에게 필수 권한이 있는지 확인합니다. JSP 템플리트 및 엔티티 bean에 직접 액세스하려는 모든 시도(즉, 웹 제어기를 사용하지 않고)는 WebSphere Application Server 보안 구성요소에서 거부됩니다.

WebSphere Application Server에 보안 WebSphere Commerce 자원 구성 방법에 대한 자세한 내용은 *WebSphere Commerce 설치 안내서*를 참조하십시오. WebSphere Application Server 내의 보안에 대한 자세한 내용은 WebSphere Application Server 문서에서 시스템 관리 주제 항목을 참조하십시오.

사용자 정의된 엔터프라이즈 beans에서 메소드에 대한 WebSphere Application Server 보안 구성 정보는 410 페이지의 『엔터프라이즈 응용프로그램으로 새 엔터프라이즈 bean 어셈블링』 및 416 페이지의 『엔터프라이즈 응용프로그램으로 수정 엔터프라이즈 bean 어셈블링』을 참조하십시오.

방화벽 보호

WebSphere Commerce 서버가 방화벽 안에서 실행되면 인터넷 클라이언트는 엔티티 bean에 직접 액세스할 수 없습니다. 이러한 접근 방법을 사용하면 페이지에 포함되는 데이터 bean에 의해 JSP 템플리트가 보호됩니다. 데이터 bean 관리자가 데이터 bean을 활성화합니다. 데이터 bean 관리자는 JSP 템플리트가 보기 명령에 의해 전달되었는지 검출합니다. 보기 명령에 의해 전달되지 않은 경우, 예외가 발생하며 JSP 템플리트의 요청이 거부됩니다.

WebSphere Commerce 액세스 제어 정책 소개

WebSphere Commerce 액세스 제어 모델은 액세스 제어 정책의 시행을 기반으로 합니다. 액세스 제어 정책을 통해 액세스 제어 규칙을 비즈니스 로직 코드로부터 구체화함으로써 액세스 제어 명령문을 코드로 하드 코딩하지 않아도 됩니다. 예를 들어, 다음과 유사한 코드를 포함시킬 필요가 없습니다.

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

액세스 제어 정책은 액세스 제어 정책 관리자에 의해 실시됩니다. 일반적으로 사용자가 보호된 자원에 액세스하려고 시도하면, 액세스 정책 관리자가 먼저 해당 보호된 자원에 적용 가능한 액세스 제어 정책을 판별한 후, 적용 가능한 액세스 제어 정책에 기초하여 사용자가 요청 자원에 액세스할 수 있는지 판별합니다.

액세스 제어 정책은 ACPOLICY 테이블에 저장된 4개의 튜플 정책입니다. 각 액세스 제어 정책은 다음 양식을 갖습니다.

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

사용자가 문제 자원과의 관계 또는 관계 그룹에 지정된 조건을 충족시키는 한 4 튜플 액세스 제어 정책의 요소는 특정 사용자 그룹에 속하는 사용자가 지정된 자원 그룹에 속하는 자원의 지정된 조치 그룹의 조치를 수행할 권한을 갖도록 지정합니다. 예를 들어 [AllUsers, UpdateDoc, doc, creator]는 사용자가 문서 작성자인 경우, 모든 사용자가 문서를 갱신할 수 있음을 지정합니다.

사용자 그룹은 MBRGRP 데이터베이스 테이블에 정의된 구성원 그룹의 특정 유형입니다. 사용자 그룹은 구성원 그룹 유형 -2와 연관되어야 합니다. -2는 액세스 그룹을 나타내며 MBRGRPTYPE 테이블에서 정의됩니다. 사용자 그룹과 구성원 그룹 유형 간의 연관은 MBRGRPUSG 테이블에 저장됩니다.

특정 사용자 그룹에 대한 사용자 멤버쉽은 명시적으로 또는 암시적으로 지정될 수 있습니다. MBRGRPMBR 테이블에서 사용자가 특정 구성원 그룹에 속하도록 지정할 경우, 명시적 스펙이 발생합니다. 사용자가 MBRGRPCOND 테이블에 지정된 조건(예: 상품 관리자 역할을 수행하는 모든 사용자)을 충족시킬 경우, 암시적 스펙이 발생합니다. 또한 복합 조건(예: 상품 관리자 역할을 수행하는 사용자와 적어도 6개월 이상 역할을 수행한 사용자) 및 명시적 제외도 있습니다.

사용자 그룹에 사용자를 포함시키는 대부분의 조건은 특정 역할을 수행하는 사용자에 기초합니다. 예를 들어, 상품 관리자 역할을 수행하는 모든 사용자가 카탈로그 관리 작업을 수행하도록 하는 액세스 제어 정책이 있을 수 있습니다. 이 경우 MBRROLE 테이블에서 상품 관리자 역할을 지정받은 모든 사용자는 사용자 그룹에서 암시적으로 제외됩니다.

구성원 그룹 서브시스템에 대한 자세한 정보는 WebSphere Commerce 온라인 도움말을 참조하십시오.

ActionGroup 요소는 ACACTGRP 테이블에서 나온 것입니다. 조치 그룹은 명시적으로 지정된 조치 그룹을 나타냅니다. 조치 목록은 ACACTION 테이블에 저장되며 각 조치와 조치 그룹과의 관계는 ACACTACTGP 테이블에 저장됩니다. 조치 그룹의 예로는 ″OrderWriteCommands″ 조치 그룹이 있습니다. 이 조치 그룹은 주문 갱신을 위해 사용되는 다음과 같은 조치를 포함합니다.

- 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

자원 그룹은 특정 유형의 자원을 그룹화하는 메커니즘입니다. 자원 그룹의 한 자원의 멤버쉽은 다음 두 가지 방법 중 한 가지로 지정할 수 있습니다.

- ACRESGRP 테이블의 조건 열 사용
- ACRESGPRES 테이블 사용

대부분의 경우, 자원과 자원 그룹을 연관시키는 데 ACRESGPRES 테이블을 사용하는 것으로 충분합니다. 이 방법을 사용하면 해당 Java 클래스 이름을 사용하여 ACRESCGRY 테이블에 자원을 정의합니다. 그리고 난 후, 이 자원은 ACRESGPRES 연관 테이블을 사용하여 적절한 자원 그룹(ACRESGRP 테이블)과 연관됩니다. Java 클래스 이름만으로 자원 그룹의 구성원을 정의하기에 충분하지 않은 경우(예를 들어, 자원 속성에 따라 이 클래스의 오브젝트를 더 제한해야 하는 경우)에는 ACRESGRP 테이블의 조건 열을 사용하여 자원 그룹을 완전하게 정의할 수 있습니다. 속성에 따라 이 자원 그룹화를 수행하려면 자원이 그룹화 가능 인터페이스도 구현해야 함에 유의하십시오.

다음 도표는 자원 그룹화 스펙 예를 보여줍니다. 이 예에서 자원 그룹 10023은 ACRESGPRES 테이블에서 연관된 모든 자원을 포함합니다. 자원 그룹 10070은 ACRESGRP 테이블에서 조건 필드 열을 사용하여 정의됩니다. 이 자원 그룹에는 또한 상태가 "Z"(공유 요청 목록 지정)인 주문 원격 인터페이스의 인스턴스가 들어 있습니다.

주: ACRESGRP 테이블의 조건 열에 대한 XML 정보의 자세한 내용은 *WebSphere Commerce 액세스 제어 안내서*를 참조하십시오.

ACRESGRP

AcResGrp_Id	GrpName	조건
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>

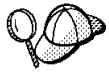
ACRESGRPES

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

그림 20.



ACACTGRP, ACRESGRP 및 ACRELGRP 테이블의 MEMBER_ID 열에는 -2001(루트 조직)값이 있어야 합니다.

액세스 제어 정책은 네 번째 요소로서 Relationship 또는 RelationshipGroup 요소를 선택적으로 포함할 수 있습니다.

액세스 제어 정책이 Relationship 요소를 사용하는 경우, 이 요소는 ACRELATION 테이블에 있습니다. 반면에 RelationshipGroup 요소를 포함하고 있는 경우, 이 요소는 ACRELGRP 테이블에 있습니다. 아무 요소도 포함하지 않아도 되지만 한 가지 요소를 포함하게 되면 다른 요소를 포함할 수 없음에 유의하십시오. ACRELGRP 테이블의 RelationshipGroup 스펙은 ACRELATION 테이블의 관계 정보에 우선합니다.

ACRELATION 테이블은 사용자와 자원 간에 존재하는 관계 유형을 지정합니다. 관계 유형의 몇 가지 예에는 작성자, 제출자 및 소유자가 포함됩니다. 관계 요소 사용의 한 예는 해당 요소를 사용하여 주문 작성자가 언제나 주문을 갱신할 수 있는지 확인하는 것입니다.

ACRELGRP 테이블은 특정 자원과 연관될 수 있는 관계 그룹 유형을 지정합니다. 관계 그룹은 한 개 이상의 관계 체인의 그룹화입니다. 관계 체인은 일련의 또 하나의 관계입니다. 관계 그룹의 한 예는 사용자가 자원의 작성자가 되어야 하고 또한 자원에서 참조하는 구매 조직 엔티티에 속하도록 지정하는 것입니다.

관계 그룹(또는 관계) 스펙은 액세스 제어 정책의 선택 부분입니다. 사용자 고유 명령을 작성하고 이 명령이 특정 역할로 제한되는 경우에 주로 사용됩니다. 이런 경우, 사용자와 자원 간의 관계를 시행하려 할 수 있습니다. 일반적으로 특정 역할로 명령이 제한되면 Relationship 요소를 사용하지 않고 액세스 제어 정책의 UserGroup 요소를 통해 달성됩니다.

액세스 제어 정책과 관련된 다른 중요한 개념은 액세스 제어 정책 *소유자*의 개념입니다. 액세스 제어 정책 소유자는 액세스 제어 정책을 소유하는 조직 엔티티입니다. 액세스 제어 정책은 액세스 제어 정책 소유자가 소유한 자원에만 적용될 수 있으므로 액세스 제어 정책의 소유자를 아는 것은 중요합니다.

각 해당 자원의 경우, 권한을 부여하는 정책이 있거나 모든 정책이 확인되고 아무것도 권한을 부여하지 않을 때까지 액세스 제어 정책 관리자는 소유 조직 엔티티 또는 구성원 계층의 상위 조직 엔티티가 소유하는 액세스 제어 정책을 적용합니다.

구성원 계층을 표시하는 다음과 같은 도표를 고려하십시오.

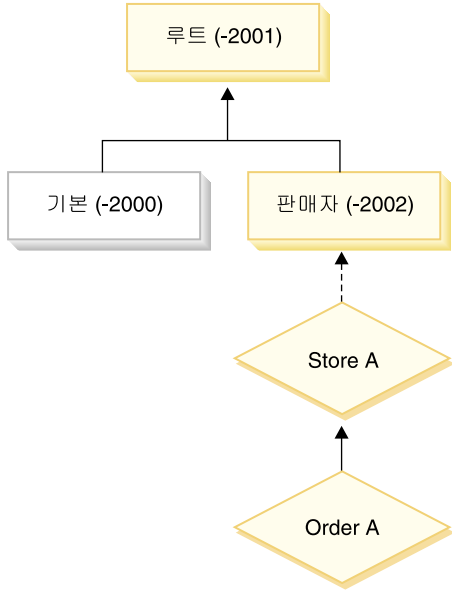


그림 21.

"OrderA" 자원의 경우, 판매자 또는 루트 조직이 소유하는 어떤 액세스 제어 정책도 적용할 수 있습니다. 액세스 제어 정책 관리자가 사용자 권한을 부여하는 이러한 조직 중 한 조직이 소유하는 한 가지 정책을 발견하게 되면(액세스 제어 정책의 네 가지 요소를 기초) 액세스 제어 정책 검색을 즉시 중지합니다. 그러나 보호된 자원에 조치를 수행하기 위한 사용자 권한을 부여하는 해당 조직이 소유하는 액세스 제어 정책을 발견하지 못하는 경우, 액세스가 거부됩니다.

관계 그룹

관계 그룹을 사용하면 복수 관계를 지정할 수 있습니다. 관계는 문제 자원과 사용자 간의 직접 관계이거나 또는 사용자와 자원을 간접적으로 관련시키는 관계 체인일 수 있습니다.

주: 관계 그룹과 관련된 다음 절에서는 WebSphere Commerce Professional Edition에서 사용 가능한 유일한 조직이 RootOrganization, DefaultOrganization 및 SellerOrganization임을 인식하는 것이 중요합니다. 다른 조직에 대한 예는 WebSphere Commerce Business Edition에만 적용됩니다.

관계와 관계 그룹 비교: 액세스 제어 정책은 액세스 중인 자원에 대해 사용자가 특정 관계를 수행해야함을 지정하거나 관계 그룹에 지정한 조건을 사용자가 수행해야 함을 지정할 수 있습니다.

대부분의 경우, 관계 지정은 응용프로그램의 액세스 제어 요구사항을 충족시켜야 합니다. 그러나 사용자와 자원 사이에 직접적이지 않지만 실제로는 사용자 자원 사이의 일련의 관계 시리즈인 관계를 지정해야 하는 정책의 경우, 관계 그룹을 사용해야 합니다.

예를 들어 사용자가 해당 조직에 대해 특정 역할을 수행하거나 사용자가 구매 조직의 구성원이어야 하는 관계의 구매 조직과 사용자 간의 연관을 지정해야 하는 경우, 관계 그룹 및 관계 체인을 사용해야 합니다.

사용자와 문제 자원 간의 직접 연관만을 시행해야 하는 경우, 단순 관계를 사용할 수 있습니다. 예를 들어 사용자가 자원 작성자여야 함을 시행하는 경우를 들 수 있습니다.

복수의 단순 관계를 결합하는 경우, 사용자는 작성자 또는 제출자여야 하며 그렇게 되면 이는 관계 체인이 되고 사용자는 관계 그룹을 사용해야 합니다. 이러한 단순 관계의 결합은 WebSphere Commerce Professional Edition 또는 WebSphere Commerce Business Edition 사용시 발생할 수 있습니다.

관계 그룹에 대한 일반 정보: 관계 체인은 일련의 또 하나의 관계입니다. 관계 체인의 길이는 체인이 포함하고 있는 관계의 수에 의해 판별됩니다. 이는 관계 체인의 XML 표시에 있는 <parametername="aName" value="aValue" /> 항목 수를 검사하여 판별할 수 있습니다.

자원의 fulfills() 메소드를 사용하여 마지막 <parameter name="Relationship" value="aValue" /> 요소만을 처리해야 합니다. 나머지는 액세스 제어 정책 관리자가 내부적으로 처리합니다.

관계 체인의 길이가 2이면, 첫 번째 <parameter name="aName" value="aValue" /> 요소는 사용자와 조직 엔티티 사이에 있습니다. 마지막 <parameter name="aName" value="aValue" /> 요소는 조직 엔티티와 자원 사이에 있습니다.

관계 그룹을 정의해야 하는 경우, XML 파일에 관계 그룹 정보를 정의해야 합니다. defaultAccessControlPolicies.xml 파일을 수정하거나 사용자 고유 XML 파일을 작성할 수 있습니다. 이러한 XML 기반 정보 작성에 대한 추가 정보는 *WebSphere Commerce 액세스 제어 안내서*를 참조하십시오.

다음 절에서는 관계 그룹의 다른 유형 예를 보여줍니다.

단순 관계 체인으로 구성된 관계 그룹: Business 액세스 제어 정책의 일부로서, 사용자가 자원의 BuyingOrganizationalEntity인 조직 엔티티에 속하도록 해야할 수 있습니다. 이를 위해서는 길이가 2인 한 개의 관계 체인으로 구성되는 관계 그룹을 작성해야 합니다. 관계 체인은 두 개의 개별 관계로 구성되기 때문에 길이가 ″2″입니다. 첫 번째 관계는 사용자의 해당 상위 조직 엔티티 사이입니다. 사용자는 해당 관계의 ″하위″입니다. 두 번째 관계의 경우, 액세스 제어 정책 관리자는 상위 조직 엔티티가 자원과의 BuyingOrganizationalEntity 관계를 이행하는지 여부를 확인합니다. 즉, 자원의 구매 조직 엔티티인 경우 ″true″를 리턴합니다.

다음 XML 부분은 defaultAccessControlPolicies.xml 파일의 일부이며 이러한 관계 그룹 유형 정의 방법을 보여줍니다.

```
<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 다른 예는 사용자가 문제 자원의 구매 조직 엔티티인 조직 엔티티의 회계 담당 역할을 하도록 해야 하는 경우입니다. 여기서도 다시 길이가 2인 한 개의 관계 체인으로 구성되는 관계 그룹을 사용합니다. 체인의 첫 번째 부분은 사용자가 회계 담당 역할을 하는 모든 조직 엔티티를 발견하게 됩니다. 그리고 난 후, 이 조직 엔티티 세트의 경우, 최소 한 개의 해당 엔티티가 자원과의

BuyingOrganizationalEntity 관계를 이행하는지 여부를 액세스 제어 정책 관리자가 확인합니다. 즉, 그들 중 하나가 자원의 구매 조직 엔티티인 경우 ″true″를 리턴합니다.

다음 XML 부분은 defaultAccessControlPolicies.xml 파일의 일부이며 이러한 관계 그룹 유형 정의 방법을 보여줍니다.

```
<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>
```

복수 관계 체인으로 구성된 관계 그룹: 복수의 관계 체인을 포함하도록 관계 그룹을 구성할 수 있습니다. 이런 경우, 사용자가 모든 관계 체인을 충족시켜야 하는지(즉 *AND* 시나리오인지) 또는 사용자가 최소 한 가지의 관계 체인을 충족시켜야 하는지(즉 *OR* 시나리오인지) 지정해야 합니다.

Business 이 관계 유형의 예를 보여주기 위해 사용자가 자원의 작성자여야 하고 또한 자원에서 지정한 BuyingOrganizationalEntity에 사용자가 반드시 속하도록 다음 XML 부분을 사용합니다. 사용자가 자원 작성자여야 함을 지정하는 첫 번째 체인의 길이는 1입니다. 사용자가 자원에서 지정한 BuyingOrganizationalEntity에 속해야 함을 지정하는 두 번째 체인의 길이는 2입니다.

```
<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>
```

시나리오 대신 사용자가 두 가지 관계 체인 중 한 가지를 만족시켜야 하는 경우, <andListCondition> 태그를 <orListCondition> 태그로 변경해야 합니다.

Professional Business WebSphere Commerce Professional Edition(WebSphere Commerce Business Edition에서 뿐 아니라)에서 사용할 수 있는 관계 그룹 예를 보여주려면 사용자가 자원 제출자 또는 작성자가 되도록 하는 데 사용하는 관계 그룹을 고려해야 합니다. 이는 다음 XML 부분에 나와 있습니다.

```
<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>
```

액세스 제어 유형

액세스 제어 유형에는 명령 레벨 액세스 제어와 자원 레벨 액세스 제어의 두 가지가 있으며, 둘 다 정책을 기반으로 합니다.

명령 레벨("역할 기반"라고도 함) 액세스 제어는 광범위한 정책 유형을 사용합니다. 특정 역할의 모든 사용자는 특정 유형의 명령을 실행할 수 있음을 지정할 수 있습니다. 예를 들어 회계 담당 역할의 사용자가 AccountRepresentativesCmdResourceGroup 자원 그룹의 모든 명령을 실행할 수 있도록 지정할 수 있습니다. 또는 다음 도표에서 설명한 대로 또다른 정책의 예를

들면 모든 상점 운영자가 StoreAdminCmdResourceGrp로 지정된 모든 자원에 대해 ExecuteCommandAction Group에 지정된 모든 조치를 수행할 수 있도록 지정하는 것입니다.

주: MBRGRPCOND 테이블의 조건 열에 대한 XML 정보는 관리 콘솔을 사용하여 액세스 그룹을 설정할 때 생성됩니다. 관리 콘솔을 사용한 액세스 그룹 설정에 대한 정보는 WebSphere Commerce 온라인 도움말을 참조하십시오.

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	조건
-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

그림 22.

명령 레벨 액세스 제어 정책의 경우 제어기 명령의 조치 그룹은 항상 ExecuteCommandActionGroup입니다. 보기의 경우 자원 그룹은 항상 ViewCommandResourceGroup입니다.

모든 제어기 명령은 명령 레벨 액세스 제어에 의해 보호되어야 합니다. 또한 바로 호출할 수 있거나 다른 명령의 간접 지시에 의해 실행할 수 있는(반대로 해당 보기로 전달하여 실행) 보기도 명령 레벨 액세스 제어에 의해 보호되어야 합니다.

명령 레벨 액세스 제어는 명령이 작용하는 자원을 고려하지 않습니다. 단지 사용자가 특정 명령을 실행할 수 있는지 여부를 판별할 뿐입니다. 사용자가 명령을 실행할 수 있는 경우, 후속 자원 레벨 액세스 제어 정책을 적용하여 사용자가 해당 자원에 액세스할 수 있는지 여부를 판별할 수 있습니다.

언제 상점 운영자가 관리 태스크 수행을 시도하는지 고려해 보십시오. 첫 번째 액세스 제어 확인 레벨을 통해 이 사용자가 특정 상점 관리 명령을 실행할 수 있는지 여부를 판별할 수 있습니다. 사용자가 이 작업을 수행할 수 있다고 판별되면 (상점 운영자가 storeAdminCmds 그룹에서 명령을 실행할 수 있기 때문에) 자원 레벨 액세스 제어 정책이 호출될 수 있습니다. 이 정책은 상점 운영자만이 사용자가 상점 운영자로 있는 조직이 소유하는 상점의 관리 태스크를 수행할 수 있도록 지정할 수 있습니다.

요약하면, 명령 레벨 액세스 제어에서 “resource”는 명령 자체이며 “action”은 단지 명령을 실행하는 것입니다(즉, 명령 오브젝트의 인스턴스를 생성하는 것입니다). 액세스 제어 확인은 사용자가 명령을 실행하도록 허용되었는지 여부를 판별합니다. 대조적으로 자원 레벨 액세스 제어에서 “resource”는 명령 또는 bean 액세스 및 “action”이 명령 자체인 보호 가능한 자원입니다.

액세스 제어 상호 작용

이 절에서는 WebSphere Commerce 액세스 제어 정책 프레임워크에서 액세스 제어가 작동하는 방법을 나타내는 상호 작용 도표를 보여줍니다.

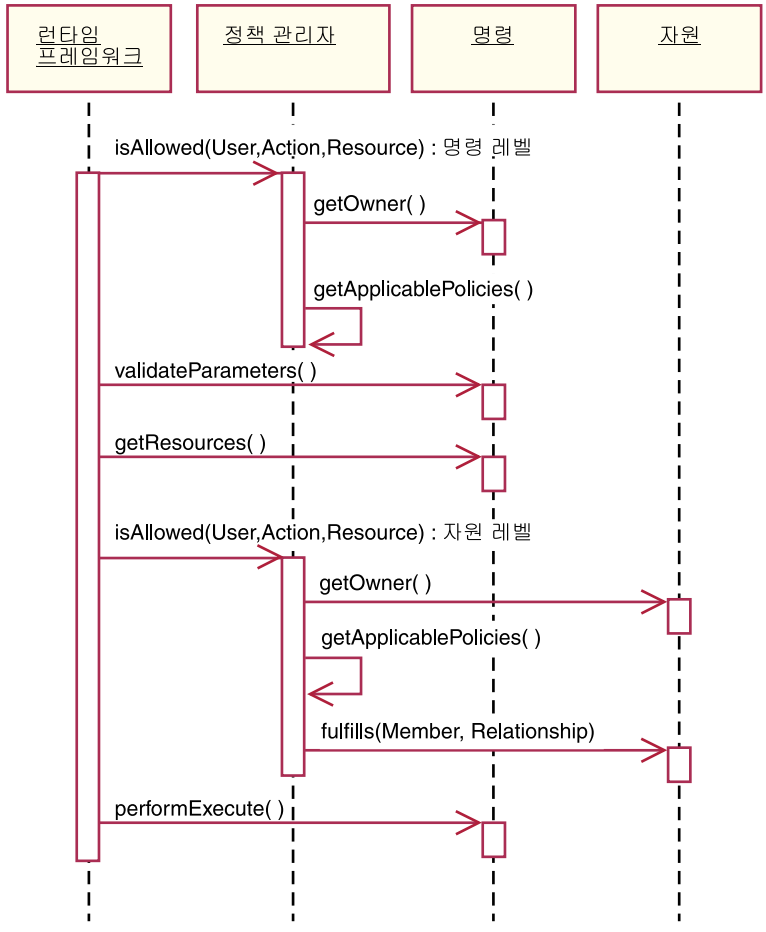


그림 23.

앞의 도표는 액세스 제어 *정책 관리자*가 수행하는 조치를 표시합니다. 액세스 제어 정책 관리자는 현재 사용자가 지정 자원에 대한 지정 조치를 실행할 수 있는지 여부를 판별하는 액세스 제어 구성요소입니다. 정책 관리자는 자원 소유자 및 해당 상위 조직이 소유하는 정책을 검색하여 이를 판별합니다. 최소한 하나의 정책이 액세스를 부여하는 경우, 권한이 부여됩니다.

다음 목록은 앞의 상호 작용 도표에 있는 조치에 대해 설명합니다. 도표의 맨 위에서 맨 아래로 순서화되어 있습니다.

1. isAllowed()
   런타임 구성요소는 사용자에게 제어기 명령 또는 보기에 대해 명령 레벨 액세스 권한이 있는지 여부를 판별합니다.
2. getOwner()
   액세스 제어 정책 관리자는 명령 레벨 자원의 소유자를 판별합니다. 기본 구현은 명령 컨텍스트에 있는 상점 소유자(storeId)의 구성원 식별자(memberId)를 리턴합니다. 명령 컨텍스트에 상점 식별자가 없는 경우, 루트 조직(-2001)이 리턴됩니다.
3. getApplicablePolicies()
   액세스 제어 정책 관리자는 지정 사용자, 조치 및 자원에 기초하여 적용 가능한 정책을 찾아 처리합니다.
4. validateParameters()
   초기 매개변수 점검 및 분석
5. getResources()
   자원과 조치 쌍의 벡터인 액세스 벡터를 리턴합니다.

   아무 것도 리턴되지 않으면, 자원 레벨 액세스 제어 확인은 수행되지 않습니다. 보호되어야 하는 자원이 있는 경우 액세스 벡터(자원과 조치 쌍으로 구성)가 리턴되어야 합니다.

   각 *자원*은 보호 가능한 오브젝트(com.ibm.commerce.security.Protectable 인터페이스를 구현하는 오브젝트)의 인스턴스입니다. 여러 경우에 자원은 액세스 bean입니다.

   액세스 bean은 com.ibm.commerce.security.Protectable 인터페이스를 구현할 수 없지만 122 페이지의 『엔터프라이즈 bean에서 액세스 제어 구현』에 포함된 정보에 따라 해당 엔터프라이즈 bean이 보호되는 한 액세스 제어 확인이 여전히 발생할 수 있습니다.

   *조치*는 자원에서 수행되는 조작을 나타내는 문자열입니다. 대부분의 경우, 조치는 명령의 인터페이스 이름입니다.
6. isAllowed()
   런타임 구성요소는 사용자에게 getResources()로 지정된 모든 자원-조치 쌍에 대한 자원 레벨 액세스 권한이 있는지 여부를 판별합니다.

7. `getOwner()`
   자원은 소유자의 `memberId`를 리턴합니다. 이것은 어떤 정책이 적용되는지 결정합니다. 자원 소유자 및 해당 상위 조직이 소유한 정책만 적용됩니다.
8. `getApplicablePolicies()`
   액세스 제어 정책 관리자는 적용 가능한 정책을 검색한 후 적용합니다. 자원에 액세스할 사용자 권한을 부여하는 정책이 자원-조치 쌍마다 적어도 하나가 있으면 액세스가 부여되고, 그렇지 않으면 액세스가 거부됩니다.
9. `fulfills()`
   적용 가능한 정책이 관계 그룹을 지정하면, 구성원이 자원에 대한 한 가지 또는 여러 지정 관계를 충족시키는지 여부를 확인하기 위해 자원에 확인 작업이 수행됩니다.
10. `performExecute()`
    명령의 비즈니스 로직

Protectable 인터페이스

WebSphere Commerce 액세스 제어 정책에 의해 보호되는 자원을 가지는 키 요소는 자원이 `com.ibm.commerce.security.Protectable` 인터페이스를 구현해야 하는 것입니다. 이 인터페이스는 엔터프라이즈 bean 및 데이터 bean과 함께 주로 사용되지만, 보호를 필요로 하는 해당 특정 bean만이 인터페이스를 구현해야 합니다.

Protectable 인터페이스를 사용하는 데 있어 자원은 두 가지 키 메소드 `getOwner()` 및 `fulfills(Long member, String relationship)`을 제공해야 합니다.

조직이나 조직 엔티티가 액세스 제어 정책을 소유합니다. `getOwner` 메소드는 보호 가능한 자원 소유자의 memberId를 리턴합니다. 또한 액세스 제어 정책 관리자가 자원 소유자를 판별한 후에는 구성원 계층에 있는 소유자의 각 상위 소유자 memberId를 가져옵니다. 소유자의 상위 소유자에게 속하는 모든 액세스 제어 정책뿐 아니라 원래 getOwner 요청의 소유자에게 속하는 모든 액세스 제어 정책이 적용됩니다.

구성원 계층에서 소유자보다 높은 레벨의 상위 소유자에게 적용되는 액세스 제어 정책뿐 아니라 지정된 소유자에게 적용되는 액세스 제어 정책이 적용됩니다.

해당 구성원이 자원과의 필수 관계를 충족시키는 경우, fulfills 메소드는 참만을 리턴합니다. 일반적으로 구성원은 단일 사용자이지만 조직일 수도 있습니다. 액세스 제어 정책의 관계 그룹을 사용하고 있는 경우, 조직이 됩니다.

Groupable 인터페이스

액세스 제어 정책의 응용프로그램은 자원 그룹에 특정합니다. 클래스 이름, 주문 상태 또는 storeId 값과 같은 속성에 기초하여 자원 그룹화가 수행됩니다.

액세스 제어 정책을 적용하기 위해 클래스 이름이 아닌 속성으로 자원을 그룹화하는 경우, 해당 자원은 com.ibm.commerce.grouping.Groupable 인터페이스를 구현해야 합니다.

다음 코드 부분은 Groupable 인터페이스를 나타냅니다.

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

예를 들어, 보류 상태(status = P(pending))인 주문에만 적용되는 정책을 구현하기 위해서는 주문 엔티티 bean의 원격 인터페이스가 Groupable 인터페이스를 구현하고 attributeName의 값을 "status"로 설정합니다.

Groupable 인터페이스는 자주 사용되지 않습니다.

액세스 제어에 관한 추가 정보 찾기

WebSphere Commerce 액세스 제어 모델에 대한 추가 정보는 *WebSphere Commerce 액세스 제어 안내서*를 참조하십시오. 이 안내서는 액세스 제어에 대한 자세한 개요를 제공하고 관리 콘솔을 사용한 정책, 조치 그룹 및 자원 그룹 작성 및 수정 방법에 대해 설명합니다.

액세스 제어 구현

이 절에서는 사용자 정의 코드에 액세스 제어를 구현하는 방법에 대해 설명합니다.

보호 가능한 자원 식별

일반적으로 엔터프라이즈 bean 및 데이터 bean은 보호하려는 자원입니다. 그러나 모든 엔터프라이즈 bean 및 데이터 bean을 보호해야 하는 것은 아닙니다. 기존 WebSphere Commerce 응용프로그램 내에서 보호가 필요한 자원은 이미 Protectable 인터페이스를 구현했습니다. 새 엔터프라이즈 bean 및 데이터 bean을 작성할 때 무엇을 보호할 것인가 하는 질문이 나옵니다. 보호할 자원의 결정은 응용프로그램에 의해 좌우됩니다.

명령이 `getResources` 메소드에서 엔터프라이즈 bean을 리턴하는 경우, 액세스 제어 정책 관리자가 엔터프라이즈 bean에서 getOwner 메소드를 호출하므로 엔터프라이즈 bean을 보호해야 합니다. 관계가 해당 자원 레벨 액세스 제어 정책에 지정된 경우, `fulfills` 메소드도 호출됩니다.

모든 자체 엔터프라이즈 bean 및 데이터 bean에 대한 protectable 인터페이스를 구현하려면(자원을 보호하려면) 응용프로그램에 여러 정책이 필요할 수 있습니다. 정책 수가 증가하면 성능은 저하되고 정책 관리는 더욱 어려워집니다.

1차 자원과 종속 자원은 이론적으로 구별됩니다. *기본 자원*은 자체 자원 상에 존재할 수 있습니다. *종속 자원*은 관련 기본 자원이 존재할 때만 존재합니다. 예를 들어, 기본 자원이 없는 WebSphere Commerce 응용프로그램 코드에서 Order 엔티티 bean은 보호 가능한 자원이지만 OrderItem 엔티티 bean은 그렇지 않습니다. 그 이유는 OrderItem의 존재 여부는 Order에 달려 있기 때문입니다. Order는 기본 자원이고 OrderItem은 종속 자원입니다. 사용자가 Order에 액세스해야할 경우, 주문 항목에도 액세스해야 합니다.

이와 마찬가지로 사용자 엔티티 bean은 보호 가능한 자원이지만, 주소 엔티티 bean은 그렇지 않습니다. 이 경우 주소는 사용자에 의해 좌우되므로 사용자에 대한 액세스를 갖는 모든 것은 주소에 대한 액세스도 가져야 합니다.

1차 자원은 보호되어야 하지만, 종속 자원은 가끔 보호를 요구하지 않습니다. 사용자가 기본 자원에 액세스할 수 있는 경우, 기본적으로 종속 자원에도 액세스할 수 있어야 합니다.

엔터프라이즈 bean에서 액세스 제어 구현

액세스 제어 정책에 의한 보호를 요구하는 새 엔터프라이즈 bean을 작성하는 경우, 다음을 수행해야 합니다.

1. 새 엔터프라이즈 bean을 작성하여 com.ibm.commerce.base.objects.ECEntityBean을 확장하는지 확인하십시오.
2. bean의 원격 인터페이스가 com.ibm.commerce.security.Protectable 인터페이스를 확장하는지 확인하십시오.
3. Java 클래스 이름이 아닌 속성으로 그룹화되는 경우, bean의 원격 인터페이스는 com.ibm.commerce.grouping.Groupable 인터페이스도 확장해야 합니다.
4. 엔터프라이즈 bean 클래스에는 다음 메소드의 기본 구현이 들어 있습니다.
   - getOwner
   - fulfills
   - getGroupingAttributeValue

   필요한 모든 메소드를 대체하십시오. 최소한 getOwner 메소드를 대체해야 합니다.
   이러한 메소드의 기본 구현은 다음 코드 발췌 부분에 나와 있습니다.

```
*********************************************************************
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;
}
*********************************************************************
```

   다음은 OrderBean에 기초한 이러한 메소드들의 견본 구현입니다.

```
*********************************************************************
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. 엔터프라이즈 bean의 액세스 bean 및 생성된 코드를 작성(또는 재작성)하십시오.

데이터 bean에서 액세스 제어 구현

데이터 bean을 보호하면 해당 데이터 bean은 액세스 제어 정책에 의해 직접적으로 또는 간접적으로 보호될 수 있습니다. 데이터 bean이 직접 보호되면, 특정 데이터 bean에 적용하는 액세스 제어 정책이 존재합니다. 데이터 bean이 간접적으로 보호되면, 액세스 제어 정책이 존재하는 다른 데이터 bean으로 보호를 위임합니다.

액세스 제어 정책에 의해 직접적으로 보호되는 새 데이터 bean을 작성하는 경우, 데이터 bean은 다음을 수행해야 합니다.

1. com.ibm.commerce.security.Protectable 인터페이스를 구현하십시오. 이와 같이, bean은 getOwner() 및 fulfills(Long member, String relationship) 메소드 구현을 제공합니다. 이는 bean의 원격 인터페이스에서도 구현되어야 합니다.

   데이터 bean이 Protectable 인터페이스를 구현하면, 데이터 bean 관리자가 isAllowed 메소드를 호출하여 현재 액세스 제어 정책에 따라 사용자가 적절한 액세스 제어 특권을 갖고 있는지 판별합니다. 다음 코드 부분에서는 isAllowed 메소드에 대해 설명합니다.

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

2. bean과 상호 작용하는 자원이 자원의 Java 클래스 이름 이외의 속성에 의해 그룹화되는 경우, bean은 com.ibm.commerce.grouping.Groupable 인터페이스를 구현해야 합니다.

3. com.ibm.commerce.security.Delegator 인터페이스를 구현하십시오. 다음 코드 부분에서는 이 인터페이스에 대해 설명합니다.

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

   주: 직접 보호되도록 하려면, getDelegate 메소드는 데이터 bean 자체를 리턴해야 합니다(즉, 데이터 bean은 액세스 제어를 위해 스스로 위임합니다).

데이터 bean을 직접 보호하는 것과 간접적으로 보호하는 것 간의 차이는 1차 및 종속 자원 간의 차이와 유사합니다. 데이터 bean 오브젝트가 자체적으로 존재할 수 있는 경우, 간접적으로 보호되어야 합니다. 데이터 bean의 존재가 다른 데이터 bean의 존재에 의해 좌우되는 경우, 보호를 위해 다른 데이터 bean으로 위임시켜야 합니다.

직접적으로 보호되는 데이터 bean의 예로는 Order 데이터 bean이 있습니다. 간접적으로 보호되는 데이터 bean의 예로는 OrderItem 데이터 bean이 있습니다.

액세스 제어 정책에 의해 간접적으로 보호되는 새 데이터 bean을 작성하는 경우, 데이터 bean은 다음을 수행해야 합니다.

1. com.ibm.commerce.security.Delegator 인터페이스를 구현하십시오. 다음 코드 부분에서는 이 인터페이스에 대해 설명합니다.

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

주: getDelegate에서 리턴하는 데이터 bean은 Protectable 인터페이스를 구현해야 합니다.

데이터 bean이 Delegator 인터페이스를 구현하지 않을 경우, 액세스 제어 정책의 보호 없이 대량 자료 반입됩니다.

제어기 명령에서 액세스 제어 구현

새 제어기 명령을 작성할 때 새 명령에 대한 구현 클래스는 com.ibm.commerce.commands.ControllerCommandImpl 클래스를 확장하고 인터페이스는 com.ibm.commerce.command.ControllerCommand 인터페이스를 확장해야 합니다.

제어기 명령에 대한 명령 레벨 정책에 대해, 명령의 인터페이스 이름은 자원으로 지정됩니다. 자원을 보호하려면, 자원을 Protectable 인터페이스로 구현해야 합니다. WebSphere Commerce 프로그래밍 모델에 따라, 이것은 com.ibm.commerce.command.ControllerCommand 인터페이스에서 명령의 인터페이스를 확장하고 com.ibm.commerce.commands.ControllerCommandImpl에서 명령의 구현을 확장하여 수행됩니다. ControllerCommand 인터페이스는 com.ibm.commerce.command.AccCommand 인터페이스를 확장하고 다시 Protectable을 확장합니다. AccCommand 인터페이스는 명령 레벨 액세스 제어에 의해 보호되도록 하기 위해 명령이 구현해야 하는 최소 인터페이스입니다.

명령이 보호해야 하는 자원에 액세스할 경우, 그 자원을 보유하기 위해 AccessVector 유형의 개인용 인스턴스 변수를 작성하십시오. 그런 다음 getResources 메소드를 대체하십시오. 이 메소드의 기본 구현은 널(null)값을 리턴하므로 자원을 확인하지 않기 때문입니다.

새 getResources 메소드에서 명령 작동의 기초가 될 수 있는 자원-조치 쌍의 배열을 리턴해야 합니다. 조치를 명시적으로 지정하지 않을 경우, 조치의 기본값은 실행 중인 명령의 인터페이스 이름입니다.

또한 메소드가 자원을 인스턴스화해야 하는지, 또는 자원에 대한 참조를 보유하는 기존 인스턴스 변수를 사용할 수 있는 지에 대해 메소드가 결정하는 것이 바람직합니다. 자원 오브젝트가 이미 존재하는지 확인하면 시스템 성능 향상에 도움이 될 수 있습니다. 그리고 난 후, 필요한 경우 새 제어기 명령의 performExecute 메소드에서 동일한 getResources 메소드를 사용할 수 있습니다.

다음은 getResources 메소드 예입니다.

```
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;
}
```

예를 들어 OrderItemUpdate 명령이 있습니다. 이 명령의 getResources 메소드는 주문 및 사용자 보호 가능 오브젝트를 리턴합니다. 조치가 지정되어 있지 않으므로, OrderItemUpdate 명령에 대한 인터페이스가 기본값입니다.

getResources 메소드는 복수 자원을 리턴할 수 있습니다. 이 경우, 조치가 수행되려면 지정된 모든 자원에 사용자 액세스를 제공하는 정책이 있어야 합니다. 사용자가 세 가지 중 두 가지 자원에 액세스한 경우, 조치는 수행될 수 없습니다(세 가지 중 세 가지에 대한 액세스가 필요합니다).

추가 매개변수 확인 또는 제어기 명령의 매개변수 분석을 수행해야 할 경우, validateParameters() 메소드를 사용하면 됩니다. 이것은 선택적입니다.

추가 자원 레벨 확인

제어기 명령의 getResources 메소드를 호출할 때, 보호해야 할 모든 자원을 판별할 수 있는 것은 아닙니다.

필요한 경우, 태스크 명령은 명령이 작용할 수 있는 자원 목록을 리턴하도록 getResources 메소드를 구현할 수 있습니다.

자원 레벨 확인을 호출하는 다른 방식은 checkIsAllowed(Object resource, String action) 메소드를 사용하여 액세스 제어 정책 관리자에 대한 직접 호출을 작성하는 것입니다. 이 메소드는 com.ibm.commerce.command.AbstractECTargetableCommand 클래스에서 확장하는 모든 클래스에 사용 가능합니다. 예를 들어, 다음 클래스는 AbstractECTargetableCommand 클래스에서 확장됩니다.

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

또한 checkIsAllowed 메소드는 com.ibm.commerce.command.AbstractECCommand 클래스를 확장하는 클래스에 사용 가능합니다. 예를 들어, 다음 클래스는 AbstractECCommand 클래스에서 확장합니다.

- com.ibm.commerce.command.TaskCommandImpl

다음은 checkIsAllowed 메소드의 서명을 보여줍니다.

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

이 메소드는 현재 사용자가 지정된 자원에서 지정된 조치를 수행할 수 없는 경우에 ECApplicationException을 발생시킵니다. 액세스가 부여되면, 메소드는 간단히 리턴됩니다.

"create" 명령을 위한 액세스 제어

getResources 메소드는 명령의 performExecute 메소드보다 먼저 호출되므로, 아직 작성되지 않은 자원의 액세스 제어에 대해서는 다른 접근 방법을 사용해야 합니다. 예를 들어 WidgetAddCmd가 있는 경우, getResources 메소드는 작성되기 시작하려는 자원을 리턴할 수 없습니다. 이 경우 getResources 메소드는 자원 작성자를 리턴해야 합니다. 예를 들어 명령은 명령 팩토리에서 작성하고 주문은 상점에서 작성하며 사용자는 조직에서 작성합니다.

명령 레벨 액세스 제어를 위한 기본 구현

명령 레벨 액세스 제어의 경우, getOwner() 메소드의 기본 구현은 storeId가 지정된 경우에 상점 소유자의 memberId를 리턴하는 것입니다. storeId를 지정하지 않으면 루트 조직의 memberId가 리턴됩니다(memberId = -2001).

getResources() 메소드의 기본 구현은 null을 리턴합니다.

validateParameters()의 기본 구현은 없습니다.

보기에서 액세스 제어 정책 구현

보기의 자원 레벨 액세스 제어는 데이터 bean 관리자가 수행합니다. 데이터 bean 관리자는 다음 경우에 호출됩니다.

1. JSP 템플리트에 <useBean> 태그가 있고 데이터 bean이 속성 목록에 없을 경우
2. JSP 템플리트가 다음 활성화 메소드를 포함 할 경우

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

주: 보호될(직접 또는 간접) 데이터 bean은 위임자 인터페이스를 구현해야 합니다. 직접 보호될 데이터 bean은 스스로 위임하게 되며 따라서 Protectable 인터페이스도 구현해야 합니다. 간접적으로 보호되는 데이터 bean은 Protectable 인터페이스를 구현하는 데이터 bean으로 위임해야 합니다.

권장하지는 않지만 액세스 제어 확인의 생략이 다음 경우에 발생합니다.

1. JSP 템플리트가 데이터 bean을 사용하지 않고 액세스 bean을 직접 호출할 경우
2. JSP 템플리트가 데이터 bean의 populate() 메소드를 직접 호출할 경우

제어기 명령의 결과가 보기로 전달될 경우(ForwardViewCommand 사용), 해당 보기에 대해 명령 레벨 액세스 제어가 수행되지 않습니다. 그리고 제어기 명령이 대량 자료가 반입된 데이터 bean(보기에서 사용)을 응답 특성의 속성 목록에 넣은 다음 보기로 전달하는 경우, JSP 템플리트는 데이터 bean 관리자를 통하지 않고 데이터에 액세스할 수 있습니다. 이것은 <useBean> 태그가 JSP 템플리트에서 사용되도록 요구합니다. 사용자가 제어기 명령을 통해 이미 액세스를 부여한 자원(데이터 bean)에 대한 중복된 자원 레벨 액세스 제어 확인을 생략할 수 있으므로 이런 방식은 JSP 템플리트를 더 효과적으로 만들 수 있습니다.

제 5 장 오류 처리 및 메시지

명령 오류 처리

WebSphere Commerce에서는 사용자 정의 코드에서 쉽게 사용할 수 있도록 잘 정의된 명령 오류 처리 프레임워크를 사용합니다. 프레임워크는 다국어 지원 상점을 지원하는 방식으로 오류를 처리하도록 설계되어 있습니다. 다음 절에서는 명령에서 발생할 수 있는 예외 유형, 예외 처리 방법, 메시지 텍스트 저장 및 사용 방법, 예외 기록 방법 및 사용자 고유 명령에서 제공된 프레임워크 사용 방법에 대해 설명합니다.

예외 유형

명령에서는 다음 예외 중 하나가 발생할 수 있습니다.

ECApplicationException

오류가 사용자와 관련된 경우 예외가 발생합니다. 예를 들어, 사용자가 유효하지 않은 매개변수를 입력하면 ECApplicationException이 발생합니다. 이러한 예외가 발생하면 웹 제어기는 재시도 가능한 명령으로 지정된 경우에도 명령을 재시도하지 않습니다.

ECSystemException

이 예외는 런타임 예외 또는 WebSphere Commerce 구성 오류가 검출된 경우에 발생합니다. 이러한 예외 유형의 예로는 널(Null)값 포인터 예외 및 트랜잭션 롤백 예외가 있습니다. 이러한 유형의 예외가 발생하면, 웹 제어기는 명령이 재시도 가능하여 데이터베이스 교착 상태 또는 데이터베이스 롤백으로 인해 예외가 발생한 경우 명령을 재시도합니다.

위의 나열된 예외 둘 모두는 com.ibm.commerce.exception 패키지에 있는 ECException 클래스를 확장한 클래스입니다.

이러한 예외 중 하나가 발생하려면 다음 정보가 지정되어야 합니다.

- 오류 보기 이름
  웹 제어기는 VIEWREG 테이블에서 이 이름을 찾아봅니다.

- ECMessage 오브젝트
  이 값은 특성 파일 내에 포함된 메시지 텍스트에 해당됩니다.
- 오류 매개변수
  이러한 이름 값 쌍은 오류 메시지로 정보를 대체하는 데 사용됩니다. 예를 들어, 메시지는 예외가 발생한 메소드의 이름을 보유하는 매개변수를 포함할 수 있습니다. 예외가 발생할 때 이 매개변수가 설정된 후, 오류 메시지가 기록될 때 로그 파일에 실제 메소드 이름이 수록됩니다.
- 오류 데이터
  오류 데이터 bean을 통해 JSP 템플리트에서 사용할 수 있는 선택적 속성입니다.

예외 처리는 로그 작성 시스템과 밀접하게 통합됩니다. 예외가 발생하면 자동으로 기록됩니다.

오류 메시지 특성 파일

오류 메시지의 유지보수를 단순화하고 다국어 지원 상점을 지원하기 위해, 오류 메시지 텍스트는 특성 파일에 저장됩니다. WebSphere Commerce 메시지 텍스트는 ecServerMessages_*XX_XX*.properties 파일에 저장됩니다. 여기서, *_XX_XX*는 로케일 지시자(예: _en_US)입니다.

명령 컨텍스트는 클라이언트에서 사용되는 언어를 나타내는 식별자를 리턴합니다. 메시지가 필요하면 웹 제어기는 언어 식별자에 따라 사용할 특성 파일을 판별합니다.

ecServerMessages*XX_XX*.properties 파일에는 두 가지 유형의 메시지인 사용자 메시지 및 시스템 메시지가 정의되어 있습니다. 사용자 메시지는 해당 브라우저에서 고객에게 표시됩니다. 시스템 메시지 및 사용자 메시지 둘 다 메시지 로그에 자동으로 캡쳐됩니다.

오류가 발생하면 필수 매개변수 중 하나는 메시지 오브젝트입니다.
ECSystemExceptions의 경우, 메시지 오브젝트는 두 개의 키(시스템 메시지에 대한 키와 사용자 메시지에 대한 키)를 포함합니다. ECApplicationExceptions의 경우, 메시지 오브젝트에는 사용자 메시지에 대한 키가 들어 있습니다(시스템 메시지는 사용되지 않습니다).

모든 시스템 메시지는 사전정의되어 있습니다. 자체 시스템 메시지를 작성할 수도 있습니다. 따라서 사용자 정의 코드가 ECSystemException을 발생시키면 사전 정의된 메시지 중 하나에 메시지 키를 지정해야 합니다. 사용자 정의 사용자 메시지를 작성할 수 있습니다. 새 사용자 메시지는 별도의 특성 파일에 저장되어야 합니다.

예외상황 처리 플로우

다음 도표는 예외 발견시 정보의 플로우를 보여줍니다. 각 단계에 대한 설명은 다음과 같습니다.

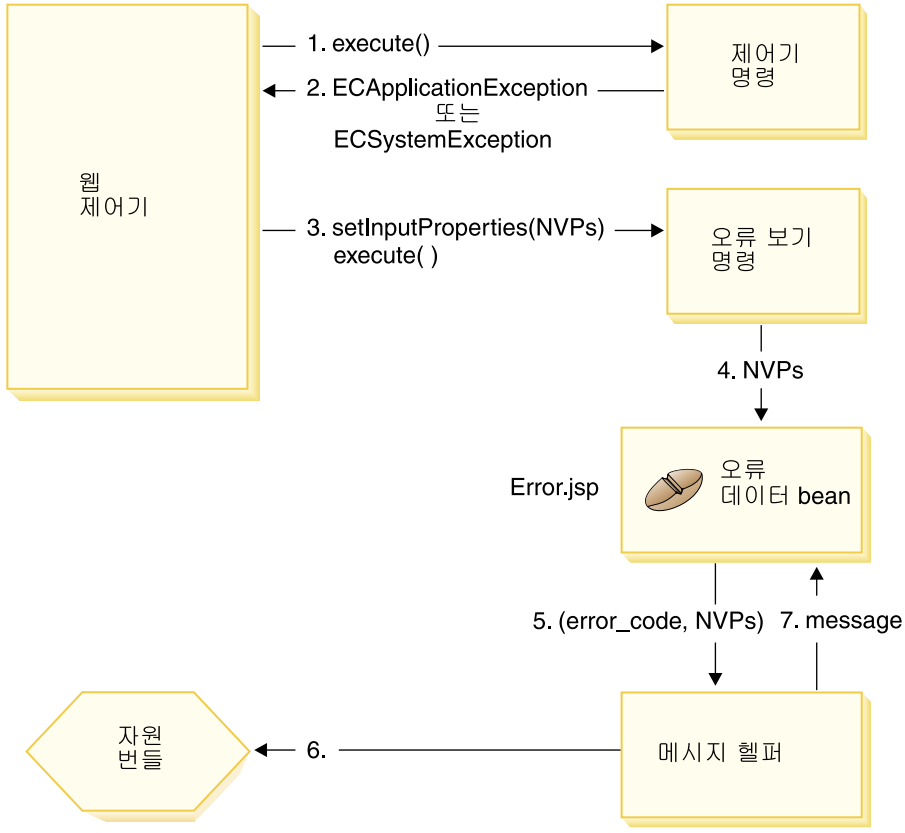


그림 24.

1. 웹 제어기가 제어기 명령을 호출합니다.

2. 명령은 웹 제어기에서 발견한 예외를 발생시킵니다. ECApplicationException 또는 ECSystemException일 수 있습니다. 예외 오브젝트에는 다음과 같은 정보가 있습니다.
   - 오류 보기 이름
   - ECMessage 오브젝트
   - 오류 매개변수
   - (선택적) 오류 데이터
3. 웹 제어기는 VIEWREG 테이블에서 오류 보기 이름을 판별하여 지정된 오류 보기 명령을 호출합니다. 명령 호출시, 웹 제어기는 ECException 오브젝트에서 특성 세트를 작성하고 보기 명령의 setInputProperties 메소드를 사용하여 보기 명령으로 설정합니다.
4. 보기 명령은 오류 JSP 템플리트(예: Error.jsp)를 호출하고 이름-값 쌍은 JSP 템플리트로 전달됩니다.
5. ErrorDataBean은 메시지 헬퍼 오브젝트로 오류 매개변수를 전달합니다.
6. 메시지 helper 오브젝트는 해당 특성 파일에서 필수 메시지(메시지 오브젝트 및 오류 매개변수 사용)를 가져옵니다.
7. 오류 데이터 bean은 JSP 템플리트로 메시지를 리턴합니다.

사용자 정의 코드에서 예외 상황 처리

새 명령 작성시, 적절한 예외 처리를 포함하는 것이 중요합니다. 예외 발견시, 필수 정보를 지정하여 WebSphere Commerce에서 제공되는 오류 처리 및 메시지 프레임워크를 이용할 수 있습니다

예외상황 처리 로직 작성 단계는 다음과 같습니다.

1. 특수 처리가 필요한 명령에서 예외를 발견합니다.
2. 발견된 예외 유형에 따라 ECApplicationException 또는 ECSystemException을 작성합니다.
3. ECApplicationException에서 새 메시지를 사용하면 새 특성 파일에 메시지를 정의합니다.

예외 발견 및 작성

처음 두 단계를 설명하기 위해 다음 코드 부분은 명령 내에서 시스템 예외 발견의 예를 보여줍니다.

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

앞의 _ERR_FINDER_EXCEPTION ECMessage 오브젝트는 다음과 같이 정의되어 있습니다.

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

_ERR_FINDER_EXCEPTION 메시지 텍스트는 다음과 같이 ecServerMessages_*xx_XX*.properties 파일 내에 정의되어 있습니다(여기서, *_xx_XX*는 *_ko_KR*과 같은 로케일 지시자입니다).

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

시스템 예외 발견시, 사용할 수 있는 사전 정의된 메시지 세트가 있습니다. 이 메시지 세트는 다음 테이블에 설명되어 있습니다.

메시지 오브젝트	설명
_ERR_FINDER_EXCEPTION	오류가 EJB finder 메소드 호출에서 리턴될 때 발생합니다.
_ERR_REMOTE_EXCEPTION	오류가 EJB 원격 메소드 호출에서 리턴될 때 발생합니다.
_ERR_CREATE_EXCEPTION	EJB 인스턴스 작성 중에 오류가 발생할 때 발생합니다.
_ERR_NAMING_EXCEPTION	오류가 이름 서버에서 리턴될 때 발생합니다.
_ERR_GENERIC	예기치 않은 시스템 오류가 발생할 때 발생합니다. (예: 널(Null)값 포인터 예외)

응용프로그램 예외 발견시, 해당 ecServerMessages_xx_xx.properties 파일에 지정된 기존의 메시지를 사용하거나 새 특성 파일에 저장된 새 메시지를 작성할 수 있습니다. 앞서 설명한 대로 ecServerMessages_xx_XX.properties 파일을 수정해서는 *안됩니다*.

다음 코드 부분에서는 명령 내에서 응용프로그램 예외 발견의 예를 보여줍니다.

```
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);
}
```

앞의 _ERR_CUSTOMER_INVALID ECMessage 오브젝트는 다음과 같이 정의되어 있습니다.

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

새 사용자 메시지 작성시, 다음과 같이 USER 유형으로 메시지를 지정해야 합니다.

```
ECMessageType.USER
```

_ERR_CUSTOMER_INVALID 메시지 텍스트가 ecCustomerMessages.properties 파일에 들어 있습니다. 이 파일은 클래스 경로에 있는 디렉토리에 상주해야 합니다. 텍스트는 다음과 같이 정의되어 있습니다.

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

메시지 작성

명령에서 새 메시지를 사용하는 ECApplicationException이 발생하면 이러한 새 메시지를 작성해야 합니다. 새 메시지 작성 단계는 다음과 같습니다.

1. 메시지 키를 포함하는 클래스 작성
2. ECMessage 오브젝트를 포함하는 클래스 작성
3. 자원 번들 작성

4. 메시지 단위 테스트

각 단계에 대한 정보는 다음 절을 참조하십시오.

메시지 키를 포함하는 클래스 작성

새 사용자 메시지 작성의 첫 번째 단계는 새 메시지 키를 포함하는 클래스를 작성하는 것입니다. 메시지 키는 자원 번들에서 해당 메시지 텍스트를 찾기 위해 로그 작성 서비스에서 사용하는 고유 지시자입니다. 이러한 새 클래스는 사용자 자신의 패키지 내에서 작성되어야 하며, WebSphere Commerce 프로젝트와는 별도의 프로젝트에 저장되어야 합니다

_ERR_CUSTOMER 및 _ERR_CUSTOMER_INVALID_ID 메시지 키를 포함하는 MyMessageKeys라는 새 클래스를 작성하고 com.mycompany.messages 패키지에 이 클래스를 넣는 MyNewMessages라는 예를 고려해 보십시오. 이 경우, 클래스 정의는 다음과 같이 표시됩니다.

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

메시지 키의 문자열 랩퍼를 제공하면 컴파일러는 해당 유효성을 확인할 수 있습니다.

ECMessage 오브젝트를 포함하는 클래스 작성

메시지 키용 클래스를 작성한 동일한 패키지 내에 ECMessage 오브젝트를 포함하는 또 다른 클래스를 작성하십시오. ECMessage 클래스는 메시지 오브젝트 구조를 정의합니다. 이 오브젝트는 로케일에 따라 달라지는 텍스트 메시지를 검색하고 존속시키는 데 사용됩니다.

메시지 오브젝트에는 심각도, 유형, 키, 자원 번들 및 연관된 자원 번들과 같은 속성이 있습니다. 이 클래스용 생성자 메소드에는 여러 가지가 있습니다. 자세한 내용은 WebSphere Commerce 온라인 도움말의 “참조” 절을 참조하십시오.

MyNewMessages 예를 따라 다음과 같이 com.mycompany.messages 패키지 내에 MyMessages라는 새 클래스를 작성하십시오.

```
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);

    }
```

위의 코드 부분에서 import 문은 ECMessage 오브젝트를 작성하기 위해 반드시 필요합니다. MyMessage._ERR_CUSTOMER 오브젝트는 심각도 ERROR의 사용자 메시지입니다. MyMessageKeys._ERR_CUSTOMER는 *ecCustomerMessages* 특성 파일에 들어 있는 메시지 텍스트를 찾기 위해 WebSphere Commerce 로그 작성 서비스에서 사용됩니다.

자원 번들 작성

해당 메시지 텍스트의 메시지 키가 저장된 새 자원 번들을 작성해야 합니다. 이 자원 번들은 Java 오브젝트 또는 특성 파일로서 구현됩니다. 특성 파일은 번역과 유지보수가 쉬우므로 이 파일을 사용하는 것이 바람직합니다. 특성 파일은 WebSphere Commerce 메시지용으로 사용됩니다

MyNewMessages 예를 계속 진행하려면 ecCustomerMessages.properties라는 이름으로 텍스트 파일을 작성하십시오. 단일 상점 Servlet에서 메시지가 사용될 경우, 이 파일을 다음 디렉토리에 놓으십시오.

drive:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_*instanceName*.ear\
wcstores.war\WEB-INF\classes

단일 도구 Servlet에서 메시지가 사용될 경우, 이 파일을 다음 디렉토리에 놓으십시오.

drive:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_*instanceName*.ear\
wctools.war\WEB-INF\classes

엔터프라이즈 응용프로그램의 Servlet에서 전체적으로 메시지가 사용될 경우, 파일을 다음 디렉토리에 놓으십시오.

```
drive:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_instanceName.ear\
   properties
```

특성 파일에 메시지 키 쌍 및 해당 메시지 텍스트가 들어 있으므로 ecCustomerMessages.properties 파일에는 다음 행이 들어 있습니다.

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

메시지 단위 테스트

메시지 키 ECMessage 오브젝트를 포함하는 클래스 및 자원 번들이 작성되면 새 메시지를 단위 테스트해야 합니다.

앞의 절에 설명된 새 메시지를 테스트하려면 다음을 수행하십시오.

1. 테스트용으로 새 클래스를 작성하십시오. 이 경우, MyTestingClass를 작성하십시오.
2. 클래스에 다음 import 문을 추가하십시오.

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

3. 클래스에 main() 메소드를 추가하십시오.
4. 다음 코드 부분을 살펴보십시오. 요구사항에 따라 코드 부분을 수정하고(예: 디렉토리 경로가 사용자 시스템의 올바른 구성 파일을 가리키는지 확인) main() 메소드에 삽입하십시오.

   ```
   // 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");
   ```

   코드의 첫 3행은 WebSphere Commerce 로그 작성 서비스를 초기화합니다. 마지막 코드 행은 MyMessage._ERR_CUSTOMER 메시지를 출력하도록 ECMessageLog에 지시합니다. MyTestingClass 및 main은 로그 추적 포맷의 일부입니다. Hello 문자열은 ecCustomerMessages.properties 파일에 정의된 메시지의 {0} 플레이스홀더에 위치합니다.

5. 클래스를 실행하십시오. 로그 파일에서 메시지 추적은 다음과 비슷하게 표시됩니다.

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

이와 비슷한 테스트를 실행하여 두 번째 메시지를 볼 수 있습니다.

실행 플로우 추적

WebSphere Commerce에는 WebSphere Commerce Server에서 실행 중인 구성요소 실행 플로우를 추적하는 데 사용되는 ECTrace 클래스가 들어 있습니다. ECTrace 클래스는 com.ibm.commerce.ras 패키지의 일부입니다.

새 비즈니스 로직 작성시, 코드 내에 추적을 삽입하여 디버깅용 메소드를 추적할 수 있습니다. 추적의 정보는 추적 로그에 캡쳐됩니다. 추적의 시작점 및 종료점을 지정할 수 있습니다. 또한 이 두 지점 사이에서 특정 데이터를 추적하도록 지정할 수도 있습니다.

추적을 사용하려면 추적을 실행할 구성요소에 추적이 작동되어야 합니다. 특정 구성요소에 대한 추적 기능을 사용하려면 관리 콘솔 또는 구성 관리자를 사용하십시오.

사용자 정의 코드 추적시, EXTERN 구성요소를 *사용해야* 합니다. 구성 관리자에서는 이를 *외부*라고 합니다.

코드 내에서 추적의 시작점을 설정하려면 다음 구문을 사용하십시오.

```
ECTrace.entry (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName, myMethodName);
```

여기서 *myClassName*은 추적된 메소드가 들어 있는 클래스의 문자열 표시입니다. 이 문자열이 파일 구문 분석을 추적하는 데 사용될 수 있으므로, 완전한 클래스 이름을 포함해야 합니다. 추적되고 있는 메소드가 static이면 myClassName의 선언 예는 다음과 같습니다.

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

추적되고 있는 메소드가 static이 아니면 myClassName의 선언 예는 다음과 같습니다.

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

메소드 내에서 데이터를 추적할 추적점을 설정하려면 다음 구문을 사용하십시오.

```
ECTrace.trace (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName, myText);
```

여기서 *myText*는 추적 로그에 표시될 텍스트입니다.

코드 내에서 추적의 종료점을 설정하려면 다음 구문을 사용하십시오.

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName);
```

추적되고 있는 메소드에서 리턴된 오브젝트를 추적해야 할 경우, 다음과 같이 종료점을 설정하십시오.

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName, returnedObject);
```

여기서 returnedObject는 메소드에서 리턴된 Java 오브젝트를 나타냅니다.

MyNewControllerCmd라는 새 제어기 명령의 performExecute 메소드를 추적해야 하는 예를 고려해 보십시오. 다음 코드 부분에서는 performExecute 메소드 내에 ECTrace 메소드를 사용하는 방법을 보여줍니다.

```
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");

}
```

앞의 performExecute 메소드가 호출되면 추적 로그 파일에는 다음과 같은 정보가 기록됩니다.

```
==============
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
```

추적은 주요 함수에만 사용되는 것이 바람직합니다. 추적은 상점 개발자가 사용하게 되므로 여러 언어로 작동되지 않습니다. 시스템 메시지는 운영용으로 사용되며 사용자 메시지는 고객에게 표시되므로 복수 언어로 사용 가능하게되는 메시지와는 대조적입니다.

JSP 템플리트 오류 처리

JSP 템플리트의 오류 처리는 여러 가지 방법으로 수행될 수 있습니다.

- 페이지 내에서의 오류 처리
  좀더 복잡한 오류 처리 및 복구가 필요한 JSP 파일의 경우, 데이터 bean에서 직접 오류를 처리하도록 파일을 작성할 수 있습니다. JSP 파일은 데이터 bean에서 발생한 예외를 발견하거나 데이터 bean이 활성화된 방법에 따라 각 데이터 bean 내의 오류 코드 세트를 확인할 수 있습니다. 그런 후 JSP 파일은 수신된 오류에 따라 해당 복구 조치를 취할 수 있습니다. JSP 파일이 다음 오류 처리 범위 조합을 사용할 수 있다는 점에 유의하십시오.
- 페이지 레벨에서의 오류 JSP
  JSP 파일은 JSP 오류 태그를 통해 자체 내에서 발생한 예외에서 고유의 기본 오류 JSP 템플리트를 지정할 수도 있습니다. 이렇게 함으로써, JSP 프로그램에서는 고유의 오류 처리를 지정할 수 있습니다. JSP 오류 태그를 지정하지 않는 JSP 파일은 응용프로그램 레벨 JSP 오류 템플리트에서 오류를 처리하게 됩니다. 페이지 레벨 오류 JSP에서 현재 트랜잭션을 롤백하려면 JSP helper 클래스(com.ibm.server.JSPHelper)를 호출해야 합니다.
- 응용프로그램 레벨의 오류 JSP
  WebSphere의 응용프로그램은 해당 Servlet 또는 JSP 파일에서 예외 발생 시 기본 오류 JSP 템플리트를 지정할 수 있습니다. 응용프로그램 레벨 오류 JSP 템플리트는 상가 레벨 또는 상점 레벨(단일 상점 모델용) 오류 핸들러로 사용할 수 있습니다. 응용프로그램 레벨 오류 JSP 템플리트에서 현재 트랜잭션을 롤백하려면 Servlet helper 클래스를 호출해야 합니다. 이는 웹 제어기가 트랜잭션을 롤백하기 위해 실행 경로에 없기 때문입니다. 가능할 때마다 앞에 나온 두 유형의 JSP 오류 처리를 사용해야 합니다. 필요한 경우에만 응용프로그램 레벨 오류 처리 전략을 사용하십시오.

제 6 장 명령 구현

이 장에는 제어기, 태스크 및 데이터 bean 명령을 작성하는 방법에 대한 정보가 있습니다. 또한 기존 제어기, 태스크 및 데이터 bean 명령 확장 방법에 대해서도 설명합니다.

주: Business 이 장에서는 비즈니스 정책 명령에 대해 설명하지 않습니다. 비즈니스 정책 명령에 대한 자세한 내용은 169 페이지의 제 7 장『거래 계약 및 비즈니스 정책(Business Edition)』을 참조하십시오.

새 명령 - 소개

WebSphere Commerce 프로그래밍 모델에서는 네 가지 유형의 명령(제어기, 태스크, 보기 및 데이터 bean 명령)을 정의합니다. e-commerce 응용프로그램을 위한 새 비즈니스 로직을 작성할 경우, 새 제어기, 태스크 및 데이터 bean 명령을 작성해야 할 수도 있습니다. 새 보기 명령을 작성할 필요는 없습니다. 보기 명령에 대한 추가 정보는 다음에 나옵니다.

새 명령은 해당 인터페이스(기존 인터페이스 확장)를 구현해야 합니다. 명령 작성을 단순화하기 위해 WebSphere Commerce에는 각 명령 유형마다 추상 구현 클래스가 있습니다. 새 명령은 이러한 클래스를 확장해야 합니다.

다음 표에는 새 명령이 확장해야 하는 구현 클래스와 새 명령이 구현해야 하는 인터페이스에 대한 정보가 나와 있습니다.

명령 유형	예제 명령 이름	확장할 대상	예제 인터페이스 구현
제어기 명령	MyControllerCmdImpl	com.ibm.commerce.command.ControllerCommandImpl	MyControllerCmd
태스크 명령	MyTaskCmdImpl	com.ibm.commerce.command.TaskCommandImpl	MyTaskCmd

명령 유형	예제 명령 이름	확장할 대상	예제 인터페이스 구현
데이터 bean 명령	MyDataBeanCmdImpl	com.ibm.commerce. command. DataBeanCommandImpl	MyDataBean

주: 구현 클래스의 이름에 있는 공간은 표시용일 뿐입니다.

다음 도표에서는 새 제어기 명령의 인터페이스 및 구현 클래스 사이의 관계를 기존의 추상 구현 클래스 및 인터페이스와 함께 보여줍니다. 추상 클래스 및 인터페이스는 모두 com.ibm.commerce.command 패키지에 있습니다.

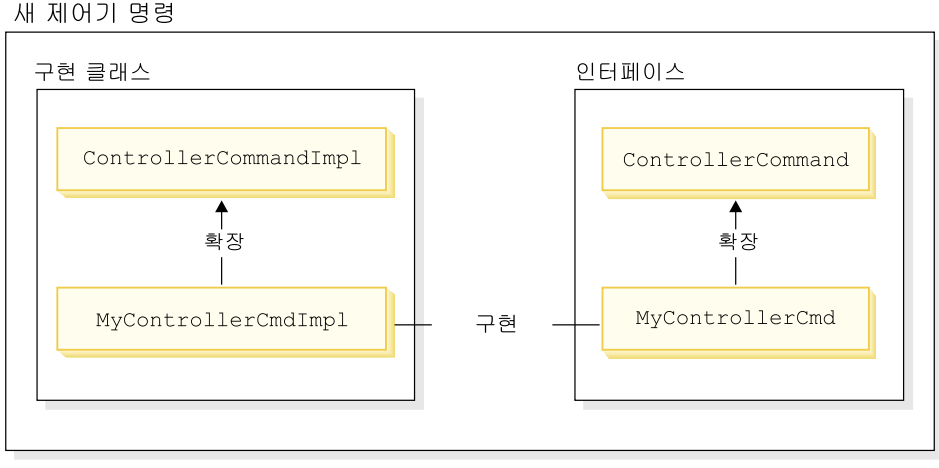


그림 25.

다음 도표에서는 새 태스크 명령의 인터페이스 및 구현 클래스 사이의 관계를 기존의 추상 구현 클래스 및 인터페이스와 함께 보여줍니다. 추상 클래스 및 인터페이스는 모두 com.ibm.commerce.command 패키지에 있습니다.

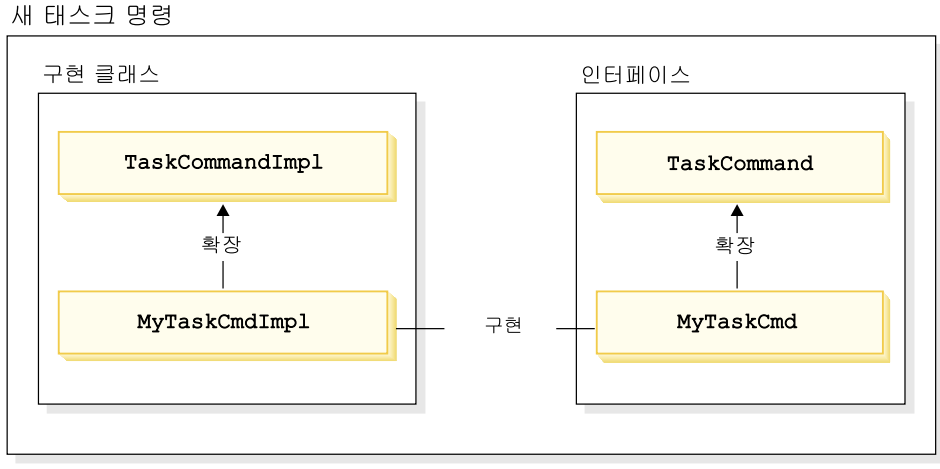


그림 26.

다음 도표에서는 새 데이터 bean 명령의 인터페이스 및 구현 클래스 사이의 관계를 기존의 추상 구현 클래스 및 인터페이스와 함께 보여줍니다. 추상 클래스 및 인터페이스는 모두 com.ibm.commerce.command 패키지에 있습니다.

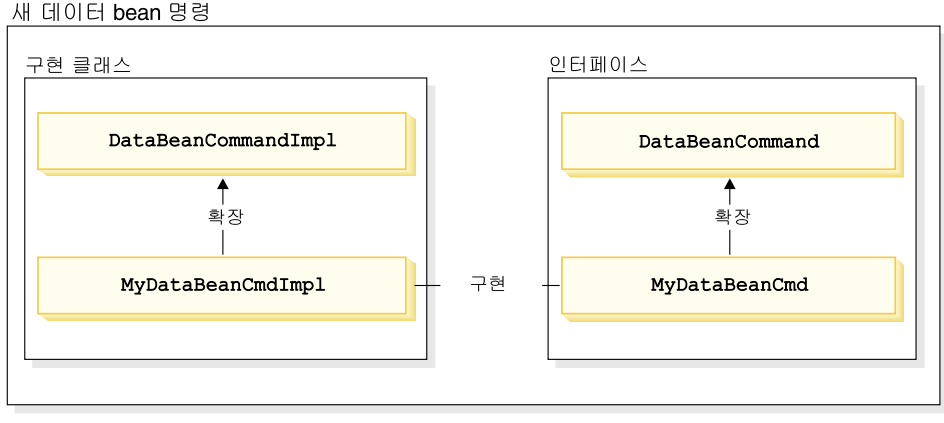


그림 27.

보기 명령에는 응답을 포맷하고 클라이언트에 응답을 보내는 두 가지 기본 기능이 있습니다. 다양한 프로토콜을 사용하여 클라이언트에 다시 응답을 보내는 다수의 일반 보기 명령이 제공됩니다. 포맷팅 기능은 일반적으로 JSP 템플리트를 호출하여 보기 명령에서 처리됩니다. 예를 들어 RedirectViewCommand 보기 명령은 URL

로 클라이언트를 지정하여 응답을 가져옵니다(그런 후 응답은 지정된 JSP 템플리트에 의해 포맷됩니다). ForwardViewCommand 보기 명령은 포맷팅을 위해 JSP 템플리트로 요청을 전달하고 페이지가 클라이언트에 표시됩니다.

이 보기 명령 모델을 사용하여 새 JSP 템플리트를 작성함으로써 새 *보기*(클라이언트에 응답)를 작성할 수 있습니다. 그러나 JSP 템플리트는 기존의 보기 명령 중 하나로 호출됩니다.

사용자 정의 코드 패키지

사용자 정의 코드 작성시, 특정 코드 구성 구조를 따라야 합니다. 일반적으로 사용자 정의 코드는 WebSphere Commerce에 포함된 것과는 별도의 패키지 및 프로젝트에서 유지보수됩니다.

새 명령 작성시, 비즈니스 요구사항을 위해 적절한 이름의 패키지에 해당 명령을 배치해야 합니다. 즉, 명령이 특정 상점에 적용되면 상점에 고유한 패키지로 이 명령을 패키지하십시오. 명령이 둘 이상의 상점에 적용되면 이에 따라 명령을 패키지하십시오. 예를 들어 다음과 같은 패키지가 있을 수 있습니다.

- `com.bigbusiness.storeA.commands`
- `com.bigbusiness.storeB.commands`
- `com.bigbusiness.commands`

앞의 패키지 구조에서는 상점 레벨의 다양한 비즈니스 로직을 허용합니다. 또한 이러한 패키지는 WebSphere Commerce 프로젝트와는 별도의 프로젝트에 저장되어야 합니다. 예를 들어 앞의 패키지는 `BigBusinessCustomCode`라는 프로젝트에 전개될 수 있습니다.

새 데이터 bean 작성시, 새 데이터 bean은 명령 로직과는 별도의 패키지에 보관되어야 합니다. 이 패키지는 명령 패키지를 저장하는 프로젝트 내에 보관될 수 있습니다. 앞 예 `BigBusinessCustomCode` 프로젝트 내에 `com.bigbusiness.databeans` 패키지를 전개하게 됩니다.

새 엔티티 bean 작성시, 새 엔티티 bean은 고유 프로젝트에 저장되어야 합니다. 따라서 `com.bigbusiness.objects` 패키지를 포함하는 `BigBusinessCustomEntityBeans` 프로젝트가 있을 수 있습니다.

이러한 패키지 전략은 코드 전개용으로 필수입니다.

명령 컨텍스트

명령 컨텍스트는 웹 제어기에 대한 핸들입니다. 명령은 명령 컨텍스트를 사용하여 웹 제어기에서 정보를 얻을 수 있습니다. 사용 가능한 정보의 예로는 사용자 ID, 사용자 오브젝트, 언어 식별자 및 상점 식별자가 있습니다.

명령 작성시, 명령 최상위 클래스의 getCommandContext() 메소드를 호출하여 명령 컨텍스트에 액세스합니다. 명령 컨텍스트는 웹 제어기에서 명령을 호출할 때 제어기 명령으로 설정됩니다. 제어기 명령은 작성시 호출되는 모든 태스크 또는 제어기 명령으로 명령 컨텍스트를 전달해야 합니다. 명령은 명령 컨텍스트에서 다음과 같은 주요 정보를 가져올 수 있습니다.

getUserId() 및 getUser()

현재 사용자 ID 또는 사용자 오브젝트를 가져옵니다. 현재 세션의 사용자 ID는 세션 컨텍스트에 저장됩니다. 세션 컨텍스트는 WebSphere Commerce 쿠키를 사용하거나 또는 WebSphere Application Server 지속 세션 오브젝트를 사용하는 두 가지 방법 중 하나로 지속될 수 있습니다. 명령 컨텍스트는 명령으로부터 세션 관리의 복잡도를 숨깁니다.

getStoreId(), getStore() 및 getStore(storeId)

현재 요청과 연관된 상점을 가져옵니다. 웹 제어기는 URL에서 상점 ID를 리턴합니다. 상점 ID가 URL에 지정되어 있지 않으면 이전 요청에서 저장된 세션 오브젝트에서 검색될 수 있습니다. WebSphere Commerce 런타임 환경은 자주 액세스되는 오브젝트 세트를 유지보수합니다. 예를 들어 상점 오브젝트 세트를 유지보수합니다. 명령은 항상 웹 제어기의 오브젝트 캐시를 이용하기 위해 명령 컨텍스트에서 상점 오브젝트를 가져와야 합니다. getStore() 메소드를 호출하여 현재 상점을 가져오거나 명령 컨텍스트에서 getStore(storeId) 메소드를 호출하여 특정 상점 오브젝트를 가져올 수 있습니다.

getLanguageId()

현재 요청에 사용되어야 하는 언어 ID를 리턴합니다. 웹 제어기는 글로벌화 프레임워크를 구현합니다. 이 프레임워크 배후 개념은 사용자가 선호하

며 상점에서 지원하는 언어를 결정하는 것입니다. URL에 언어 ID가 들어 있는 경우, 웹 제어기는 이 언어가 상점에서 지원되는지 여부를 결정합니다. 그런 경우 getLanguageId() 메소드가 get을 리턴하는 언어 ID입니다. 언어 ID가 URL에 포함되지 않는 경우, 웹 제어기는 의사결정 트리를 통해 이동하여 현재 세션 오브젝트에 또는 사용자의 등록된 환경설정에 언어 ID(상점에서 지원)가 있는지 여부를 결정하거나 결국 상점에 대한 기본 언어 ID를 리턴합니다.

getCurrency()

현재 요청에 대해 사용되는 선택한 통화를 리턴합니다. 통화는 글로벌화 프레임워크의 부분이므로, 이 메소드 배후의 로직은 getLanguageId() 메소드와 유사합니다.

getCurrentTradingAgreements() 및 getTradingAgreement (tradingAgreementId)

현재 세션에 대해 사용되는 거래 계약 세트를 리턴합니다. 이 세트는 사용자에게 자격이 부여되는 모든 거래 계약이거나 ContractSetInSession 명령에 의해 정의된 서브세트일 수 있습니다. 명령은 항상 웹 제어기의 오브젝트 캐시를 이용하기 위해 명령 컨텍스트에서 거래 계약 오브젝트를 가져와야 합니다. getCurrentTradingAgreements() 메소드를 호출하여 현재 거래 계약을 가져오거나 명령 컨텍스트에서 getTradingAgreement (tradingAgreementId) 메소드를 호출하여 특정 거래 계약을 가져올 수 있습니다.

명령 컨텍스트는 읽기 전용 오브젝트로 사용되어야 합니다. 해당 setter 메소드를 호출해서는 안됩니다. setter 메소드는 WebSphere Commerce 런타임 환경에서 사용하기 위해 예약되어 있으며 향후 릴리스에서는 사용되지 않을 수 있습니다.

명령 컨텍스트 API에 대한 자세한 내용은 WebSphere Commerce 온라인 도움말의 “참조” 절을 참조하십시오.

새 제어기 명령

위에서 언급한 대로, 새 제어기 명령은 추상 제어기 명령 클래스(com.ibm.commerce.command.ControllerCommandImpl)에서 확장해야 합니다. 새 제어기 명령 작성시, 추상 클래스에서 다음 메소드를 대체해야 합니다.

- isGeneric()
- isRetriable()
- setRequestProperties(com.ibm.commerce.datatype.TypedProperty reqParms)
- validateParameters()
- getResources()
- performExecute()

앞의 각 메소드에 대한 추가 정보는 다음 절을 참조하십시오.

isGeneric 메소드

표준 WebSphere Commerce 구현에서는 여러 유형의 사용자가 있습니다. 여기에는 일반, 게스트 및 등록 사용자가 포함됩니다. 등록 사용자 그룹 내에 고객 및 운영자가 있습니다.

일반 사용자에게는 전체 시스템에 걸쳐 사용되는 공통 사용자 ID가 있습니다. 이 공통 사용자 ID는 시스템 자원 사용을 최소화하는 방식으로 사이트에서 일반 찾아보기를 지원하는 데 사용됩니다. 웹 제어기는 일반 사용자가 호출하는 명령에 대해 사용자 오브젝트를 검색하지 않아도 되므로 일반 찾아보기에는 이 일반 사용자 ID를 사용하는 것이 보다 효율적입니다.

isGeneric 메소드는 부울값을 리턴하고 이 값은 일반 사용자가 명령을 호출할 수 있는지 여부를 지정합니다. 제어기 명령 최상위 클래스의 isGeneric 메소드는 이 값을 `false`(호출자가 등록된 사용자나 게스트 사용자여야 한다는 것을 의미)로 설정합니다. 일반 사용자가 새 제어기 명령을 호출할 수 있으면 `true`를 리턴하도록 이 메소드를 대체하십시오.

새 명령이 사용자와 연관된 자원을 가져오거나 작성하지 않는 경우, `true`를 리턴하도록 이 메소드를 대체해야 합니다. 일반 사용자가 호출할 수 있는 명령의 예로는 ProductDisplay 명령이 있습니다. 모든 사용자가 상품을 볼 수 있게 하는 것

이 좋습니다. 사용자가 게스트 또는 등록된 사용자여야 하는 명령의 예로는 (isGeneric은 false를 리턴) OrderItemAdd 명령이 있습니다.

isGeneric이 true 값을 리턴하면 웹 제어기는 현재 세션의 새 사용자 오브젝트를 작성하지 않습니다. 이와 같이 웹 제어기는 사용자 오브젝트를 검색할 필요가 없으므로 일반 사용자가 호출할 수 있는 명령이 보다 빠르게 실행됩니다.

이 메소드를 사용하여 일반 사용자가 명령을 호출할 수 있는 구문은 다음과 같습니다.

```
public boolean isGeneric()
{
     return true;
}
```

isRetriable 메소드

isRetriable 메소드는 부울값을 리턴하고 이 값은 트랜잭션 롤백 예외에서 명령을 재시도할 수 있는지 여부를 지정합니다. 새 제어기 명령 최상위 클래스의 isRetriable 메소드는 false 값을 리턴합니다. 명령이 트랜잭션 롤백 예외에서 재시도될 수 있는 경우, true 값을 리턴하도록 이 메소드를 대체해야 합니다.

트랜잭션 예외의 경우 재시도 되어서는 안되는 명령의 예로는 OrderProcess 명령이 있습니다. 이 명령은 제3자 지불 권한 처리를 호출합니다. 해당 권한은 취소될 수 없으므로 재시도될 수 없습니다. 재시도될 수 있는 명령의 예로는 ProductDisplay 명령이 있습니다.

트랜잭션 롤백 예외의 경우에 명령을 재시도할 수 있도록 하는 구문은 다음과 같습니다.

```
public boolean isRetriable()
{
     return true;
}
```

setRequestProperties 메소드

setRequestProperties 메소드는 제어기 명령에 모든 입력 특성을 전달하기 위해 웹 제어기에서 호출됩니다. 제어기 명령은 입력 특성을 구문 분석하고 이 메소드 내

에 명시적으로 각각의 개별 특성을 설정해야 합니다. 제어기 명령에 의한 이러한 명시적 특성 설정은 유형 안전 특성의 개념을 한 차원 높입니다.

이 메소드 사용 구문은 다음과 같습니다.

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty reqParms)
{

     // parse the input properties and explicitly set each parameter

}
```

validateParameters 메소드

validateParameters 메소드는 초기 매개변수 점검과 매개변수의 필요한 해결책을 수행하기 위해 사용됩니다. 예를 들어, `orderId=*`를 해결하기 위해 사용될 수 있습니다. 이 메소드는 getResources 및 performExecute 메소드 모두 이전에 호출됩니다. 이 순서에 대한 자세한 내용은 116 페이지의 『액세스 제어 상호 작용』을 참조하십시오.

getResources 메소드

이 메소드는 자원 레벨 액세스 제어를 구현하기 위해 사용됩니다. 이 메소드는 명령 작동의 기초가 될 수 있는 자원-작업 쌍의 벡터를 리턴합니다. 아무 것도 리턴되지 않으면, 자원 레벨 액세스 제어는 수행되지 않습니다. 액세스 제어에 대한 자세한 내용은 101 페이지의 제 4 장 『액세스 제어』를 참조하십시오.

performExecute 메소드

performExecute 메소드에는 명령의 비즈니스 로직이 들어 있습니다. 새 비즈니스 로직을 실행하기 전에 명령 최상위 클래스의 performExecute 메소드를 호출해야 합니다. 또한 마지막에는 보기 이름을 리턴해야 합니다.

다음은 새 제어기 명령의 performExecute 메소드 구문 예를 보여줍니다. 이 경우, 응답은 경로 재지정 보기 명령을 사용하거나 전달 보기 명령 또는 직접 보기 명령을 사용할 수 있습니다.

```
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);
}
```

performExecute 메소드 내의 경로 재지정 URL을 지정하고 VIEWREG 테이블에 항목이 존재하면, 코드에 지정된 값은 VIEWREG 테이블의 값보다 우선순위가 높습니다. 코드 내에서 JSP 템플리트 스펙의 경우에도 동일한 우선순위가 적용됩니다.

장기 실행 제어기 명령

제어기 명령을 실행하는 데 시간이 오래 걸리는 경우, 명령을 두 명령으로 분리할 수 있습니다. URL 요청 결과로 실행되는 첫 번째 명령은 간단히 스케줄러에 두 번째 명령을 추가하여 백그라운드 작업으로 실행되도록 합니다. 이러한 내용은 다음 도표에서 보여줍니다.

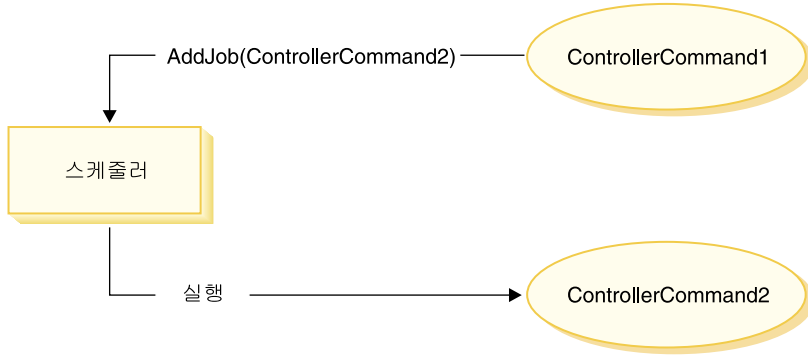


그림 28.

앞의 도표에 나타난 플로우는 다음과 같습니다.

1. ControllerCommand1이 URL 요청 결과로 실행됩니다.
2. ControllerCommand1은 스케줄러에 작업을 추가합니다. 작업은 ControllerCommand2입니다. ControllerCommand1은 스케줄러에 작업을 추가한 후 바로 보기를 리턴합니다.
3. 스케줄러는 백그라운드 작업으로 ControllerCommand2를 실행합니다.

이 시나리오에서 클라이언트는 일반적으로 ControllerCommand2에서 결과를 폴합니다. ControllerCommand2는 데이터베이스에 작업 상태를 기록합니다.

보기 명령에 대한 입력 특성 포맷팅

제어기 명령이 완료되면, 실행되어야 하는 보기의 이름을 리턴합니다. 이 보기는 가져온 여러 입력 특성이 전달되도록 요구할 수 있습니다. 다음 목록에서 설명한 것처럼, 이 입력 매개변수에 대해 세 개의 소스가 있을 수 있습니다.

- CMDREG 테이블의 PROPERTIES 열에 저장되는 기본 특성
- VIEWREG 테이블의 PROPERTIES 열 기본 특성
- URL의 입력 특성

이 특성이 병합되고 JSP 템플리트 속성에 설정되는 방식에 대한 추가 정보는 49 페이지의 『JSP 속성 설정 - 개요』를 참조하십시오. 이 절에서는 보기 명령에 대해 입력 특성이 포맷될 수 있는 방식을 설명합니다.

보기 명령을 경로 재지정하려면 두 가지 주제가 검토됩니다.

- URL 경로 재지정을 지원하기 위해 조회 문자열 고르기
- URL 경로 재지정의 길이 제한 처리

전달 보기 명령의 경우, 입력 매개변수의 열거와 HttpServletRequestObject에서 입력 변수를 속성으로 설정하는 주제가 검토됩니다.

HttpRedirectView의 조회 문자열로 입력 매개변수 고르기

보기 명령을 경로 재지정하기 위해 전달된 모든 입력 매개변수는 URL 경로 재지정을 위한 조회 문자열로 정돈됩니다. 예를 들어, 경로 재지정 보기 명령에 대한 입력에는 다음 특성이 들어 있다고 가정하십시오.

```
URL = "MyView?p1=v1&p2=v2";
ip1 = "iv1";   // input to orginal controller command
ip2 = "iv2" ;  // input to original controller command
op1 = "ov1";
op2 = "ov2";
```

이전 입력 매개변수에 근거하여 최종 URL은 다음과 같습니다.

```
MyView?p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2
```

SSL을 사용하기 위한 명령인 경우, 매개변수가 암호화되고 최종 URL을 다음과 같이 표시됩니다.

```
MyView?krypto=encrypted_value_of"p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2"
```

제한된 길이의 URL 경로 재지정 핸들링

기본값으로, 제어기 명령에 대한 모든 입력 매개변수는 경로 재지정 보기 명령으로 전달됩니다. URL 경로 재지정에서 문자수 제한이 있는 경우, 문제점이 발생할 수 있습니다. 길이가 제한될 수 있는 경우는 클라이언트가 Internet Explorer 브라우저를 사용 중일 때입니다. 이 브라우저의 경우, URL은 2083바이트를 초과할 수 없습니다. URL이 이 한계를 초과하는 경우, URL이 잘립니다. 암호화 문자열의 길이는 암호화되지 않은 문자열에 2 - 3배 입니다. 따라서 많은 수의 입력 매개변수가 있거나 암호화를 사용 중인 경우, 문제점이 발생할 수 있습니다.

제한된 길이의 URL 경로 재지정 핸들링을 위한 두 가지 접근 방법이 있습니다.

1. 경로 재지정 보기 명령에 전달되는 데 필요한 매개변수 세트만을 리턴하려면 제어기 명령에서 getViewInputProperties 메소드를 대체하십시오.
2. 매개변수가 입력 매개변수 문자열에서 제거될 수 있음을 나타내려면 URL 매개변수에서 지정된 특수 문자를 사용하십시오.

각 선행 접근 방법을 설명하려면 제어기 명령에 대해 다음과 같은 입력 매개변수 세트를 고려하십시오.

```
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";
```

getViewInputProperties 메소드를 대체하려는 경우, 다음과 같은 매개변수만이 보기 명령으로 전달되도록 새 메소드를 작성할 수 있습니다.

```
ir2="ipr2";
is="isv";
```

두 번째 접근 방법 사용시, 특정 매개변수를 사용하여 보기 명령을 호출하여 특정 입력 매개변수를 제거해야 함을 나타낼 수 있습니다. 예를 들어, URL 매개변수로 다음을 지정하여 동일한 결과를 얻을 수 있습니다.

```
URL="MyView?ip*=&iq*=&ir1="
```

이 URL 매개변수는 다음의 WebSphere Commerce 런타임 프레임워크를 지시합니다.

- `ip*=` 스펙은 이름이 `ip`로 시작하는 모든 매개변수를 제거해야 함을 의미합니다.
- `iq*=` 스펙은 이름이 `iq`로 시작하는 모든 매개변수를 제거해야 함을 의미합니다.
- `ir1=` 스펙은 `ir1` 매개변수를 제거해야 함을 의미합니다.

HttpForwardView에 대한 HttpServletRequest 오브젝트에서 속성 설정

기본 HttpForwardViewCommandImpl은 명령으로 전달되는 모든 매개변수를 열거하며 HttpServletRequest 오브젝트에서 속성으로 설정합니다.

예를 들어, 전달 보기 명령으로 전달되는 requestProperties 오브젝트에는 다음 매개변수가 들어 있다고 가정하십시오.

```
p1="pv1";
p2="pv2";
p3=pv3;   // pv3 is an object
```

다음 속성은 request.setAttribute() 메소드를 사용하여 JSP 템플리트로 전달됩니다.

```
request.setAttribute("p1", "pv1");
request.setAttribute("p2", "pv2");
request.setAttribute("p1", pv1);
request.setAttribute("RequestProperties", requestProperties);
request.setAttribute("CommandContext", commandContext);
```

여기서 requestProperties은 명령으로 전달되는 TypedProperty 오브젝트이며, commandContext는 명령으로 전달되는 명령 컨텍스트 오브젝트입니다. p1, p2 및 p3은 requestProperties 오브젝트에 정의된 매개변수입니다.

제어기 명령에 대한 데이터베이스 확약 및 롤백

제어기 명령 실행을 통해, 데이터가 자주 작성되거나 갱신됩니다. 이 경우, 트랜잭션이 끝날 때 데이터베이스가 정보로 갱신되어야 합니다. 트랜잭션은 웹 제어기로 관리됩니다.

웹 제어기는 제어기 명령을 호출하기 전에 트랜잭션의 처음을 표시합니다. 제어기 명령 실행이 완료되면 제어기 명령은 웹 제어기로 보기 이름을 리턴합니다. 웹 제어기는 트랜잭션의 종료를 표시하는 데 관여합니다. 트랜잭션이 종료하는 실제 지점(보기를 호출하기 전이나 후)은 사용되는 보기 유형에 의해 좌우됩니다.

보기 명령에는 다음 세 가지 유형이 있습니다.

- 전달 보기 명령

- 경로 재지정 보기 명령
- 직접 보기 명령

웹 제어기는 VIEWREG 테이블에서 보기 이름을 찾아보기하여 보기에 사용되는 보기 명령을 결정합니다.

VIEWREG 테이블의 항목이 ForwardViewCommand의 사용을 지정하는 경우, 웹 제어기는 제어기 명령의 결과를 해당 ForwardViewCommand 구현 클래스(역시 VIEWREG에 지정)에 대한 제어기 명령 결과를 전달합니다. 보기 명령은 현재 트랜잭션의 컨텍스트 내에서 실행합니다. 이 경우, 데이터베이스 확약 또는 롤백은 보기 명령이 완료될 때까지 발생하지 않습니다

VIEWREG 테이블의 항목이 RedirectViewCommand의 사용을 지정하는 경우, 웹 제어기는 제어기 명령의 결과를 해당 RedirectViewCommand 구현 클래스에 대한 제어기 명령 결과를 전달합니다. 그러면 보기 명령은 현재 트랜잭션 범위 외부에서 동작하며, 경로 재지정 보기 명령이 호출되기 전에 데이터베이스 확약 또는 롤백이 발생합니다.

VIEWREG 테이블의 항목이 DirectViewCommand의 사용을 지정하는 경우, 웹 제어기는 제어기 명령의 결과를 해당 DirectViewCommand 구현 클래스에 대한 제어기 명령 결과를 전달합니다. 보기 명령은 현재 트랜잭션의 컨텍스트 내에서 실행합니다. 이 경우, 데이터베이스 확약 또는 롤백은 보기 명령이 완료될 때까지 발생하지 않습니다. (ForwardViewCommand와 DirectViewCommand는 유사합니다. ForwardViewCommand는 결과를 JSP 템플리트로 전달합니다. 반면 DirectViewCommand는 입력 스트림으로 결과를 받아서 출력 스트림으로 전달합니다. 데이터를 바이트로 취급하는 getRawDocument 메소드나 데이터를 텍스트로 취급하는 getTextDocument를 사용합니다.)

보기 명령이 제어기 명령과 동일한 트랜잭션 범위에서 실행될 경우, 보기 명령에 오류가 발생하면 전체 트랜잭션이 롤백됩니다. 비즈니스 로직에 따라 이것은 바람직한 결과일 수도 있고 아닐 수도 있습니다.

제어기 명령 트랜잭션 범위의 예

제어기 명령의 트랜잭션 범위의 차이점을 알아보려면 사용된 보기 명령 유형에 따라 다음 예를 고려해 보십시오.

사례 1: 제어기 명령 트랜잭션 범위 내에서 보기 실행

YourControllerCmdA라는 새 제어기 명령을 작성한 경우, 명령의 performExecute 메소드는 다음을 포함합니다.

```
.
.
// Create a new TypedProperty object for output.
TypedProperty rspProp = new TypedProperty();

//////////////////////
// Business logic   //
//////////////////////

// Return the view
rspProp.put(ECConstants.EC_VIEWTASKNAME, "YourView");
SetResponseProperties(rspProp);
```

앞의 코드 부분에서 제어기 명령은 “YourView”를 보기로 리턴합니다. YourView는 VIEWREG 테이블에 등록됩니다. 다음은 YourView를 등록하기 위한 삽입 명령문의 예입니다.

```
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');
```

여기서 XX는 상점 식별자입니다. 보기는 com.ibm.commerce.command.HttpForwardViewCommandImpl 구현 클래스를 사용하므로 웹 제어기는 그래픽 전달 보기 명령을 사용합니다.

앞의 명령 등록에 따라 웹 제어기는 제어기 명령 트랜잭션 범위 내에서 YourView.jsp 파일을 실행합니다. YourView.jsp에서 오류가 발생한 경우, 트랜잭션이 실패하며 데이터베이스 롤백이 발생합니다. 결과적으로 전체 제어기 명령이 실패합니다.

사례 2: 제어기 명령 트랜잭션 범위 외부에서 보기 실행

오류가 보기에서 발생할 수 있는 경우일지라도 정보를 데이터베이스에 확약하는 것을 선호한다고 가정하십시오. 보기가 제어기 명령 트랜잭션 범위 외부에서 실행되도록 하려면 보기를 경로 재지정으로 실행해야 합니다.

보기를 경로 재지정으로 실행하기 위해 제어기 명령의 performExecute 메소드는 다음과 같은 방식으로 보기를 리턴합니다.

```
.
.
// 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");
```

다음 예제 SQL 문은 경로 재지정 전략을 지원합니다.

```
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');
```

여기서, XX는 상점 식별자입니다.

명령은 EC_GENERIC_REDIRECTVIEW 값을 응답 특성 매개변수로 전달하므로, 웹 제어기는 일반 경로 재지정 보기 명령을 사용합니다. 일반 경로 재지정 보기는 다음 정보와 함께 VIEWREG 테이블에 등록됩니다.

- ViewName = RedirectView
- DeviceFmt_Id = -1
- InterfaceName = com.ibm.commerce.command.RedirectViewCommand
- ClassName = com.ibm.commerce.command.HttpRedirectViewCommandImpl

웹 제어기는 일반 경로 재지정 보기 명령을 호출하고, 이 명령은 경로 재지정 URL을 입력 특성으로 취합니다. 응답은 경로 재지정 URL로 경로 재지정됩니다. 경로 재지정이 발생하고 나면 YourView2가 호출됩니다. 이는 일반 전달 보기로 구현됩니다.

새 작업 명령

새 태스크 명령은 추상 태스크 명령 클래스(com.ibm.commerce.command.TaskCommandImpl)를 확장해야 하며 com.ibm.commerce.TaskCommand 인터페이스를 확장하는 인터페이스를 구현해야 합니다. 145 페이지의 도표에 표시된 대로 새 태스크 명령은 다음과 같이 정의되어야 합니다.

```
public class MyTaskCmdImpl extends com.ibm.commerce.command.TaskCommandImpl
     implements MyTaskCmd {

}
```

작업 명령의 모든 입력 및 출력 특성은 MyTaskCmd와 같은 명령 인터페이스에 정의되어야 합니다. 호출자는 작업 명령 구현 클래스가 아니라 작업 명령 인터페이스로 프로그램합니다. 그러면 호출자는 호출할 구현 클래스에 대해 걱정하지 않고 복수 태스크 명령을 구현할 수 있습니다(각 상점마다 하나).

인터페이스에 정의된 모든 메소드는 구현 클래스에서 구현되어야 합니다. 명령 컨텍스트를 호출자(제어기 명령)가 설정하므로, 작업 명령은 명령 컨텍스트를 설정할 필요가 없습니다. 그러나 작업 명령은 명령 컨텍스트를 사용하여 웹 제어기에서 정보를 확보할 수 있습니다.

태스크 명령 인터페이스에 정의된 메소드를 구현할 뿐만 아니라, com.ibm.commerce.command.TaskCommandImpl 클래스에서 performExecute 메소드를 대체해야 합니다.

performExecute 메소드에는 작업 명령이 수행하는 특정 작업 단위의 비즈니스 로직이 포함되어 있습니다. 비즈니스 로직을 수행하기 전에 작업 명령 최상위 클래스의 performExecute 메소드를 호출해야 합니다. 다음 코드 부분에서는 태스크 명령의 performExecute 메소드 예를 보여줍니다.

```
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.
}
```

런타임 프레임워크는 제어기 명령의 getResources 메소드를 호출하여 명령이 액세스할 보호 가능한 자원을 결정할 수 있습니다. 태스크 명령이 제어기 명령 범위 및 제어기 명령의 getResources 메소드가 리턴한 자원에 액세스하려는 중 실행되는 경우일 수 있습니다. 이 경우, 태스크 명령 자체가 getResources 메소드를 구현하여 보호 가능한 자원에 대해 액세스 제어가 제공되는지 확인할 수 있습니다.

기본값으로, getResources는 태스크 명령에 대해 Null을 리턴하며 자원 레벨 액세스 제어 확인은 수행되지 않습니다. 그러므로 태스크 명령이 보호 가능한 자원을 액세스하는 경우 이를 대체해야 합니다.

기존 명령 사용자 정의

이 절에서는 기존의 제어기, 태스크 및 데이터 bean 명령을 사용자 정의할 수 있는 다양한 방법에 대해 설명합니다.

기존 제어기 명령 사용자 정의

제어기 명령은 비즈니스 처리의 비즈니스 로직을 캡슐화합니다. 비즈니스 처리 내의 각각의 작업 단위는 태스크 명령으로 수행될 수 있습니다. 이와 같이 제어기 명령을 사용자 정의할 수 있는 여러 방법이 있으며, 이 중 일부는 태스크 명령의 사용자 정의를 수반합니다.

제어기 명령을 사용자 정의할 때, 다음을 수행할 수 있습니다.

- 기존 제어기 명령에 추가적인 처리 및 로직을 추가합니다. 기존 비즈니스 로직 이전, 기존 로직 다음 또는 이전과 이후 둘 다에 추가할 수 있습니다.
- 하나 이상의 태스크 명령을 바꾸어 비즈니스 처리에서 특정 단계의 수행 방법이 수정됩니다.

- 제어기 명령으로 호출되는 보기를 바꿉니다.

다음 절에서는 앞의 사항을 수정하는 방법에 대한 정보를 제공합니다.

제어기 명령에 새 비즈니스 로직 추가

ExistingControllerCmd라는 기존의 WebSphere Commerce 제어기 명령이 있다고 가정하십시오. WebSphere Commerce 이름 지정 규칙에 따라, 이 제어기 명령은 ExistingControllerCmd라는 인터페이스 클래스와 ExistingControllerCmdImpl이라는 구현 클래스를 갖습니다. 이제 비즈니스 요구사항이 발생하고 기존 요청 명령에 새 비즈니스 로직을 추가해야 한다고 가정하십시오. 로직의 한 부분은 기존 요청 명령 로직 이전에 실행되어야 하며 다른 부분은 기존 명령 로직 다음에 실행되어야 합니다.

새 비즈니스 로직을 추가하는 첫 번째 단계는 원래의 구현 클래스를 확장하는 새 구현 클래스를 작성하는 것입니다. 이 예에서 ExistingControllerCmdImpl 클래스를 확장하는 새 ModifiedControllerCmdImpl 클래스를 작성합니다. 새 구현 클래스는 원래의 인터페이스(ExistingControllerCmd)를 구현해야 합니다.

새 구현 클래스에서 새 performExecute 메소드를 작성하여 기존 요청 명령의 performExecute를 대체해야 합니다. 새 performExecute 메소드 내에서 새 비즈니스 로직을 삽입할 수 있는 두 가지 방식이 있습니다. 제어기 명령에 직접 코드를 포함하거나, 새 고객 태스크 명령을 작성하여 새 비즈니스 로직을 수행할 수 있습니다. 새 태스크 명령을 작성하는 경우 제어기 명령 내에서 새 태스크 명령 오브젝트의 인스턴스를 생성해야 합니다.

다음 일부 코드는 제어기 명령에 직접 로직을 넣어 기존 제어기 명령의 시작과 끝 부분에 새 비즈니스 로직 추가 방법을 보여줍니다.

```
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.
            */
          }
    }
```

다음 일부 코드는 제어기 명령 내에서 새 태스크 명령의 인스턴스를 생성하여 기존 제어기 명령 시작 부분에 새 비즈니스 로직을 추가하는 방법을 보여줍니다. 또한 새 태스크 명령 인터페이스 및 구현 클래스를 작성하고 명령 레지스트리에 태스크 명령을 등록하게 됩니다.

```
// 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();
         }
   }
```

제어기 명령에 새 비즈니스 로직을 넣을 것인지 로직을 수행할 태스크 명령을 작성할 것인지 여부에 관계없이 WebSphere Commerce 명령 레지스트리의 CMDREG 테이블을 갱신하여 새 제어기 명령 구현 클래스를 기존 제어기 명령 인터페이스와 연관시켜야 합니다. 다음 SQL 문은 갱신 예를 보여줍니다.

```
update CMDREG
set CLASSNAME='ModifiedControllerCmdImpl'
where INTERFACENAME='ExistingControllerCmd'
```

제어기 명령으로 호출되는 태스크 명령 바꾸기

제어기 명령은 각 태스크를 수행하는 여러 태스크 명령을 호출합니다. 이러한 태스크는 함께 제어기 명령으로 표시되는 비즈니스 처리를 구성합니다. 제어기 명령의 시작 또는 끝 부분에 새 비즈니스 로직을 추가하기보다 처리의 특정 단계가 수

행되는 방식을 변경해야 할 수 있습니다. 이 경우 대체하려는 태스크 명령의 인스턴스를 원하는 방식으로 태스크를 수행하는 새 태스크 명령의 인스턴스로 바꾸어야 합니다.

WebSphere Commerce 프로그래밍 모델의 설계 때문에 태스크 명령을 바꿀 새 제어기 명령 구현 클래스를 작성할 필요가 없습니다. 제어기 명령은 명령 팩토리의 createCommand 메소드를 호출하여 태스크 명령의 인스턴스를 생성합니다. 명령 팩토리는 태스크 명령의 인터페이스 이름을 사용한 다음 명령 레지스트리에 따라 올바른 구현 클래스를 판별합니다. 이와 같이 인스턴스가 생성된 태스크 명령을 바꾸려면, 새 태스크 명령 구현 클래스를 작성한 다음 명령 레지스트리를 갱신하여 원래의 태스크 명령 인터페이스 이름이 새 태스크 명령 구현 클래스와 연관되도록 해야 합니다. 자세한 내용은 165 페이지의 『기존 태스크 명령 사용자 정의』를 참조하십시오

제어기 명령으로 호출되는 보기 바꾸기

제어기 명령으로 호출되는 보기를 바꾸기 위해, 제어기 명령에 대한 새 구현 클래스를 작성합니다. 예를 들어, ExistingControllerCmdImpl을 확장하고 ExistingControllerCmd 인터페이스를 구현하는 새 ModifiedControllerCmdImpl을 작성하십시오.

ModifiedControllerCmdImpl 클래스 내에서 performExecute 메소드를 대체하십시오. 새 performExecute 메소드에서 super.performExecute를 호출하여 모든 명령 처리가 발생하는지 확인하십시오. 명령 로직이 실행된 후, 응답 특성을 사용하여 호출되는 보기를 대체할 수 있습니다. 다음 일부 코드는 보기가 경로 재지정으로 실행될 때 보기를 대체하는 방법을 표시합니다.

```
// 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);

         }
   }
```

다음 일부 코드는 보기가 전달 보기로 실행될 때 보기를 대체하는 방법을 표시합니다.

```
// 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);

         }
    }
```

어떠한 보기가 기존 제어기 명령에서 사용되는지를 판별하려면 WebSphere Commerce 온라인 도움말의 참조 절을 참조하십시오.

기존 태스크 명령 사용자 정의

기존의 WebSphere Commerce 태스크 명령을 수정하기 위한 두 가지 표준 방법이 있습니다. 이러한 수정 메소드로 다음을 수행할 수 있습니다.

- 기존 태스크 명령에 추가 처리 및 로직을 추가합니다. 기존 비즈니스 로직 이전, 기존 로직 다음 또는 이전과 이후 둘 다에 추가할 수 있습니다.
- 사용자 자신의 비즈니스 로직으로 기존 비즈니스 로직을 완전히 바꿉니다.

위의 수정 작업을 수행하기 위해 실제로 새 태스크 명령 구현 클래스를 작성합니다. 자세한 내용은 다음 절에서 제공됩니다.

태스크 명령에 새 비즈니스 로직 추가

ExistingTaskCmd라는 기존의 WebSphere Commerce 태스크 명령이 있다고 가정하십시오. WebSphere Commerce 이름 지정 규칙에 따라, 이 제어기 명령은 ExistingTaskCmd라는 인터페이스 클래스와 ExistingTaskCmdImpl이라는 구현 클래스를 갖습니다. 이제 비즈니스 요구사항이 발생하고 기존 요청 명령에 새 비즈니스 로직을 추가해야 한다고 가정하십시오. 로직의 한 부분은 기존 요청 명령 로직 이전에 실행되어야 하며 다른 부분은 기존 명령 로직 다음에 실행되어야 합니다.

새 비즈니스 로직을 추가하는 첫 번째 단계는 원래의 구현 클래스를 확장하는 새 구현 클래스를 작성하는 것입니다. 이 예에서 ExistingTaskCmdImpl 클래스를 확장하는 새 ModifiedTaskCmdImpl 클래스를 작성합니다. 새 구현 클래스는 원래의 인터페이스(ExistingTaskCmd)를 구현해야 합니다.

새 명령 내에서 기존 performExecute 메소드를 대체하고 super.performExecute 메소드를 호출하기 전과 후에 새 로직을 포함합니다.

다음 가상 코드는 기존 태스크 명령에 새 비즈니스 로직을 추가하는 방법을 보여줍니다.

```
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.
        */
}
```

CMDREG 테이블을 갱신하여 새 구현 클래스를 기존 인터페이스와 연관시켜야 합니다. 다음 SQL 문은 갱신 예를 보여줍니다.

```
update CMDREG
set CLASSNAME='ModifiedTaskCmdImpl'
where INTERFACENAME='ExistingTaskCmd'
```

기존 태스크 명령의 비즈니스 로직 바꾸기

기존 태스크 명령의 비즈니스 로직을 바꾸려면, 태스크 명령에 대한 새 구현 클래스를 작성해야 합니다. 이 새로운 구현 클래스는 기존 태스크 명령에서 확장해야 하지만 기존 인터페이스는 구현하지 않아야 합니다. 또한 새 구현 클래스에서도 상위 클래스의 performExecute 메소드를 호출하지 마십시오.

바꾸고 있는 정확한 명령에서 확장하는 것이 직관에 반대되어 보일 수는 있지만, 이 접근 방법을 사용하는 이유는 WebSphere Commerce의 나중 버전을 지원하기 위해서입니다. 이 접근 방법은 WebSphere Commerce의 나중 버전에서 명령 인터페이스에 대해 코드가 변경되지 않도록 보호합니다.

예에서 처럼, OrderNotifyCmdImpl 태스크 명령의 비즈니스 로직을 바꾸려고 했다고 가정합시다. 이 경우, CustomizedOrderNotifyCmdImpl이라고 하는 새 태스크 명령을 작성합니다. 이 명령은 OrderNotifyCmdImpl을 확장합니다. 새 CustomizedOrderNotifyCmdImpl에서 새 비즈니스 로직을 작성하지만 최상위 클래스에서 performExecute super 메소드를 호출하지는 않습니다. WebSphere Commerce의 나중 버전에서 인터페이스에 newMethod라고 하는 새 메소드를 도입할 경우, OrderNotifyCmdImpl 명령의 해당 버전에 newMethod 메소드의 기본 구현이 포함됩니다. 이 때, 새 명령은 OrderNotifyCmdImpl에서 확장되므로 컴파일러는 OrderNotifyCmdImpl 명령에서 이 새 메소드의 기본 구현을 찾아서 사용자의 새 명령은 인터페이스 변경으로부터 보호됩니다.

새 구현 클래스가 기존 클래스와 동일한 외부 행위를 제공하는지 확인하려면 WebSphere Commerce 온라인 도움말의 참조 절을 참조하십시오.

데이터 bean 사용자 정의

데이터 bean은 일반적으로 액세스 bean을 확장합니다. VisualAge for Java에서 생성할 수 있는 액세스 bean은 엔티티 bean의 정보에 액세스하는 간단한 방법을 제공합니다. 엔티티 bean이 수정되면(예: 새 필드, 새 비즈니스 메소드 또는 새 finder 추가) 액세스 bean이 다시 생성되자마자 액세스 bean에 갱신이 반영됩니다. 데이터 bean이 액세스 bean을 확장하므로, 자동으로 새 속성을 상속합니다. 이러한 관계의 결과 별도의 코드 없이도 데이터 bean이 엔티티 bean의 새 속성을 사용하도록 할 수 있습니다.

엔티티 bean에서 가져오지 않은 데이터 bean에 새 속성을 추가해야 할 경우, Java 상속을 사용하여 기존의 데이터 bean을 확장할 수 있습니다. 예를 들어 OrderDataBean에 새 필드를 추가하려면 다음과 같이 MyOrderDataBean을 정의하십시오.

```
public class MyOrderDataBean extends OrderDataBean
{
     public String myNewField () {
     //  implement the new field here
     }

}
```

새 데이터 bean에는 BeanInfo 클래스도 있어야 합니다. 이 클래스 선언의 견본은 다음과 같습니다.

```
public class MyOrderDataBeanInfo extends java.beans.SimpleBeanInfo
{

}
```

VisualAge for Java에서는 이 BeanInfo 클래스를 생성할 수 있는 도구를 제공합니다.

제 7 장 거래 계약 및 비즈니스 정책(Business Edition)

이 장은 WebSphere Commerce Business Edition에만 적용됩니다.

소개

B2B(Business-to-Business) commerce의 키 요소 중 하나는 관계 관리입니다. 거래 계약은 구매자 및 판매자 조직 간의 비즈니스 관계를 관리하기 위해 사용됩니다. WebSphere Commerce Business Edition에서 사용하는 거래 계약 모델은 장기 구매 계약 및 RFQ(Request For Quote)와 같은 다양한 거래 계약 유형을 지원합니다.

거래 계약의 기본 요소는 규정 세트입니다. 각 규정은 거래 중 사용되는 특정 비즈니스 규칙을 정의합니다. WebSphere Commerce Business Edition을 사용하면, RFQ 온라인 처리를 사용하거나 오프라인으로 규정을 협상한 후 WebSphere Commerce 액셀러레이터의 비즈니스 관계 관리 인터페이스를 사용하여 캡처할 수 있습니다.

규정을 모델화하는 여러 가지 방법이 있습니다.

- 가격 목록 및 반품 정책과 같이 사전정의된 비즈니스 정책 중 하나를 선택하는 규정. 사용자가 작성한 비즈니스 정책을 선택할 수도 있습니다. 하나의 규정 오브젝트가 복수 비즈니스 정책 오브젝트를 참조할 수도 있습니다.
- 표준 가격 책정에 대한 조정과 같이 비즈니스 정책에 대한 특정 조정을 적용하는 규정.
- 비즈니스 처리를 제어하는 매개변수 세트를 정의하는 규정. 예를 들어, 특정 장기 구매 계약으로 특정 서비스 센터를 사용하도록 지정할 수 있습니다.

장기 구매 계약은 규정 세트로 구성됩니다. 이것은 다음 도표에 표시됩니다.

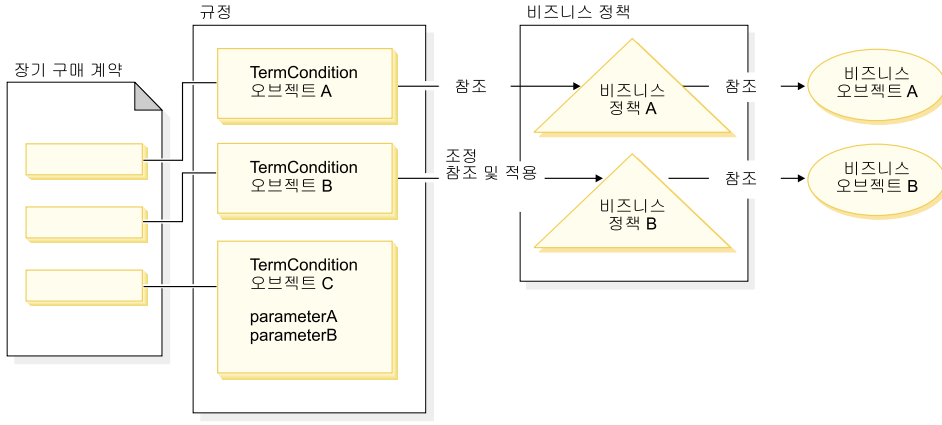


그림 29.

이전 도표에서 다음에 유의하십시오.

- “조정”이라는 용어는 비즈니스에 대한 수정을 말합니다. 예를 들어, 이것은 10% 할인이 표준 가격에 적용하기 위해 비즈니스 정책의 결과에 할인을 적용하기도 합니다. 또한 매개변수 세트를 사용하여 비즈니스 정책에 영향을 주기 위해 사용될 수도 있습니다.
- 예를 들어, TermCondition 오브젝트 A는 운송 규정 오브젝트를 표시할 수 있습니다. 이 경우 비즈니스 정책 A는 운송 모드 비즈니스 정책을 표시하고 비즈니스 오브젝트 A는 운송 회사 XYZ의 운송 모드 “A3”을 표시할 수 있습니다.
- 다른 예로 TermCondition 오브젝트 B는 비즈니스 정책 B에 의해 정의된 가격의 50% 할인을 적용하는 가격 규정 오브젝트를 표시할 수 있습니다. 이 경우 비즈니스 정책 B는 가격 정책이고 비즈니스 오브젝트 B는 마스터 카테고리에 대한 거래 포지션을 정의하는 거래 포지션 컨테이너입니다.

이 장에서는 새 비즈니스 정책 및 새 규정을 작성하는 방법에 대한 지시사항을 프로그래머에게 제공합니다.

ToolTech 견본 상점은 비즈니스 플로우에서의 운송 규정 오브젝트와 가격 규정 오브젝트를 보여줍니다. 이러한 예를 지원하는 장기 구매 계약 데이터에 대한 정보는 172 페이지의 『ToolTech 견본 장기 구매 계약 데이터』를 참조하십시오.

비즈니스 정책 오브젝트 및 명령

비즈니스 정책 오브젝트에는 다음 정보가 포함됩니다.

- 정책 ID
  이것은 비즈니스 정책 오브젝트의 1차 키입니다.
- 정책 유형
  비즈니스 정책 유형을 정의합니다. 정책 유형의 예로는 `Price` 및 `ProductSet`가 있습니다.
- 정책 이름
  각 비즈니스 정책은 고유한 이름을 가져야 합니다.
- 상점 엔티티
  비즈니스 정책이 전개된 상점 또는 상점 그룹
- 특성
  비즈니스 정책 명령에 전달될 수 있는 기본 특성 세트. 비즈니스 정책 오브젝트와 연관된 명령은 BusinessPolicyCmd 테이블에 저장됩니다.
- 유효 기간
  비즈니스 정책 오브젝트의 유효 기간
- 비즈니스 정책 명령
  비즈니스 정책을 구현하는 0개 이상의 비즈니스 정책 명령 비즈니스 정책 명령은 보통 태스크 명령이나 제어기 명령이 될 수 있는 비즈니스 처리에 의해 호출됩니다. 예를 들어 `getContractPrice()` 명령은 가격 규정을 가져옵니다. 이 가격 규정은 특정 가격 책정 정책 명령을 참조하며 이 가격 책정 정책 명령은 가격을 계산하는 데 사용됩니다.

복수 비즈니스 정책 명령이 단일 비즈니스 정책 오브젝트와 연관될 수 있습니다. 각 비즈니스 정책 명령은 비즈니스 정책 유형 오브젝트에서 정의한 동일한 인터페이스를 구현해야 합니다. 새 비즈니스 정책 명령의 구조는 다음 도표로 설명됩니다.

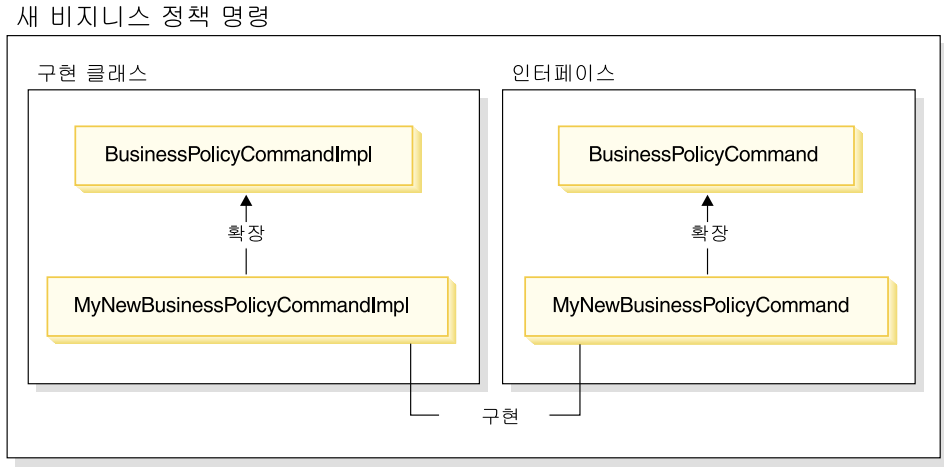


그림 30.

앞의 도표와 같이 새 비즈니스 정책 명령을 작성하려면 WebSphere Commerce BusinessPolicyCmdImpl 구현 클래스를 확장하는 새 구현 클래스를 작성해야 합니다. BusinessPolicyCmd 인터페이스를 확장하는 새 인터페이스도 작성하십시오.

ToolTech 견본 장기 구매 계약 데이터

이 절에서는 ToolTech 견본 상점에서 사용되는 장기 구매 계약 데이터 중 일부에 대한 소개를 제공합니다.

다음 절에 있는 견본 데이터는 데이터베이스 테이블로 구성됩니다. 관련된 행 및 열만 표시됩니다. 견본을 설치할 때 고유한 식별자(예: CONTRACT_ID)가 여기에 표시된 것과 다른 값을 가질 수 있다는 점도 유의하십시오.

CONTRACT 테이블 견본 데이터

다음 표는 ToolTech CONTRACT 데이터베이스 테이블에서 관련된 견본 데이터를 보여줍니다. 표시 목적으로 데이터베이스 열 헤딩이 첫 번째 열에, 견본 데이터의 행이 두 번째 열에 표시됩니다.

열 이름	견본 데이터
CONTRACT_ID	10007
MAJORVERSION	1

열 이름	견본 데이터
MINORVERSION	0
NAME	ToolTechContractNumber 4567
MEMBER_ID	-2001
ORIGIN	0
STATE	3
USAGE	1
MARKFORDELETE	0

TERMCOND 테이블 견본 데이터

다음 표는 ToolTech TERMCOND 데이터베이스 테이블에서 관련된 견본 데이터를 보여줍니다. 표시 목적으로 데이터베이스 열 헤딩이 첫 번째 열에, 견본 데이터의 행이 두 번째 및 세 번째 열에 표시됩니다.

열 이름	견본 데이터 행 1	견본 데이터 행 2
TERMCOND_ID	10025	10030
TCSUBTYPE_ID	PriceTCPriceListWithSelectiveAdjustment	ShippingTCShippingMode
TRADING_ID	10007	10007
STRINGFILED1	ProductSet2	
INTEGERFIELD2	10002	
INTEGERFIELD3	1	
BIGINTFIELD1	10051	
FLOATFIELD1	-50.0	
SEQUENCE	1	6

POLICYTC 테이블 견본 데이터

다음 표는 ToolTech POLICYTC 데이터베이스 테이블에서 관련된 견본 데이터를 보여줍니다. 이 표는 정책과 규정 오브젝트 사이의 관계를 설정합니다.

	열 이름	
	POLICY_ID	**TERMCOND_ID**
견본 데이터 행 1	10053	10025

	열 이름	
	POLICY_ID	**TERMCOND_ID**
견본 데이터 행 2	10056	10030

POLICY 테이블 견본 데이터

다음 표는 ToolTech POLICY 데이터베이스 테이블에서 관련된 견본 데이터를 보여줍니다.

열 이름	견본 데이터 행 1	견본 데이터 행 2
POLICY_ID	10053	10056
POLICYNAME	MasterCatalogPriceList	A3
POLICYTYPE_ID	Price	ShippingMode
STOREENT_ID	10051	10051
PROPERTIES	n a m e = T o o l T e c h & member_id=-2001	shippingMode=A3
STARTTIME	널(NULL)값	널(NULL)값
ENDTIME	널(NULL)값	널(NULL)값

TRADEPOSCN 테이블 견본 데이터

다음 표는 ToolTech TRADEPOSCN 데이터베이스 테이블에서 관련된 견본 데이터를 보여줍니다.

	열 이름			
	READEPOSCN_ID	**MEMBER_ID**	**NAME**	**TYPE**
견본 데이터 행	10051	-2001	ToolTech	S

SHIPMODE 테이블 견본 데이터

다음 표는 ToolTech SHIPMODE 데이터베이스 테이블에서 관련된 견본 데이터를 보여줍니다.

	열 이름			
	SHIPMODE_ID	**STOREENTITY_ID**	**CODE**	**CARRIER**
견본 데이터 행	10053	10051	A3	XYZ 운송 회사

기존 장기 구매 계약 모델 확장

장기 구매 계약은 하나 이상의 규정 오브젝트(각각은 정책을 참조함)로 구성될 수 있습니다. 후속 절에서는 새 비즈니스 정책을 작성하고 이를 비즈니스 플로우에 통합하는 데 필요한 단계를 설명합니다.

다음은 태스크를 수행하기 위한 상위 레벨 단계들입니다.

1. 새 비즈니스 정책을 작성합니다. 다음 태스크는 새 비즈니스 정책 명령 작성에 관련됩니다.
   a. 새 비즈니스 정책 유형 작성(필요한 경우)
      몇 가지의 비즈니스 정책 유형이 제공되지만, 표준 유형이 비즈니스 요구사항을 충족시키지 못할 경우, 새 비즈니스 정책 유형을 작성하십시오.
   b. 새 비즈니스 정책 명령 작성
   c. 새 비즈니스 정책 및 비즈니스 정책 명령을 등록하십시오.
2. 규정 오브젝트를 새 비즈니스 정책에 관련짓기
   이는 기존 규정 오브젝트를 새 비즈니스 정책에 관련짓거나 새 규정 오브젝트를 작성하여 수행할 수 있습니다. 새 규정 오브젝트를 작성할 경우, 다음 단계를 수행해야 합니다.
   a. 데이터베이스에서 새 규정 등록
   b. 장기 구매 계약(DTD)에서 새 규정 등록
   c. 규정에 대한 새 CMP 엔터프라이즈 bean 작성
   d. 새 규정을 반영하도록 WebSphere Commerce 액셀러레이터 갱신
3. 비즈니스 플로우 중 새 비즈니스 정책 호출

새 비즈니스 정책 작성

새 비즈니스 정책 작성에는 일반적으로 데이터베이스에서의 고유 비즈니스 정책 등록과 새 비즈니스 정책 명령 작성이 포함됩니다.

새 비즈니스 정책 명령 작성에는 다음과 같은 고급 단계가 수반됩니다.

1. 새 비즈니스 정책 유형 작성(필요한 경우)
2. 새 비즈니스 정책 명령 작성
3. 데이터베이스에 새 비즈니스 정책 및 비즈니스 정책 명령 등록

각각의 앞 단계에 대해 후속 절에서 더 자세히 설명합니다.

새 비즈니스 정책 유형 작성

이 절에서는 새 비즈니스 정책 유형을 작성하는 방법에 대해 설명합니다. 비즈니스 정책 유형은 정책이 적용되는 트랜잭션 영역을 나타냅니다. 비즈니스 정책 유형의 예는 다음과 같습니다.

- Price
- ProductSet
- ShippingMode
- ShippingCharge
- Payment
- ReturnCharge
- ReturnApproval
- ReturnPayment
- InvoiceFormat

기존 비즈니스 정책 유형이 비즈니스 요구사항을 충족시키지 못할 경우, 새 비즈니스 정책 유형을 작성해야 합니다. 새 비즈니스 정책 유형 작성은 비즈니스 정책 유형의 정의 및 등록으로 구성됩니다.

새 정책 유형을 정의하고 등록할 때, 다음과 같은 데이터베이스 테이블을 갱신해야 합니다.

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

POLICYTYPE 테이블은 작성 중인 비즈니스 정책의 유형을 지정합니다. 이 테이블은 1차 키인 한 개 열 POLICYTYPE_ID가 있습니다. 예를 들어 Price가 있습니다. 새 비즈니스 정책 유형을 작성하는 경우, 고유한 POLICYTYPE_ID를 지정해야 합니다.

PLCYTYCMIF 테이블은 명령 인터페이스 관계 스펙 테이블에 대한 비즈니스 정책 유형입니다. 즉, 비즈니스 정책 유형마다 비즈니스 정책 오브젝트에 대한 Java 명령 인터페이스를 지정합니다. 비즈니스 정책을 구현하는 0개 이상의 비즈니스 정책 명령이 있을 수 있지만, 각 비즈니스 정책 명령은 여기에 지정된 인터페이스를 구현해야 합니다.

PLCYTYPDSC 테이블은 비즈니스 정책 유형의 설명을 지정합니다. 이것은 언어 식별자 설명과 비즈니스 정책 유형의 설명을 포함합니다.

새 비즈니스 정책 유형을 작성하려면, 새 비즈니스 정책 유형에 대한 각 테이블에서 항목을 작성하십시오. 다음 SQL 문에서는 예를 제공합니다.

```
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.');
```

새 비즈니스 정책 유형을 작성하기 위한 최종 단계로, 하나 이상의 새 비즈니스 정책 유형 인터페이스를 코드화할 수 있습니다. 이 인터페이스는 해당되는 비즈니스 정책 유형의 범주에 속하는 비즈니스 정책 명령에 의해 구현됩니다. 예를 들어, ToolTech 견본 상점에서 가격은 비즈니스 정책 유형으로 정의됩니다. 이에 따라, 모든 가격 관련 비즈니스 정책 명령에 의해 구현되는 com.ibm.commerce.price.commands.ResolvePriceListsCmd 명령 및 com.ibm.commerce.price.commands.RetrievePricesCmd 인터페이스가 있습니다.

새 비즈니스 정책 유형에 대해 조작을 수행하는 비즈니스 정책 명령이 없으면, 새 인터페이스를 작성하지 않아도 됩니다. 이는 극히 드문 경우이며, 대부분 새 비즈니스 정책 유형을 작성할 때 새 비즈니스 정책 유형 인터페이스도 작성해야 합니다.

비즈니스 정책 유형 인터페이스를 작성할 때, 새 인터페이스는 com.ibm.commerce.command.BusinessPolicyCommand 인터페이스를 확장해야 합니다.

새 비즈니스 정책 명령 작성

새 비즈니스 정책 명령을 작성하려면, 명령이 관련되는 비즈니스 정책의 인터페이스를 구현하는 새 명령을 작성해야 합니다. 새 명령은 또한 com.ibm.commerce.command.BusinessPolicyCommandImpl 구현 클래스를 확장해야 합니다. 이것은 새 제어기 또는 태스크 명령을 작성하는 것과 매우 유사합니다.

비즈니스 정책 명령으로 입력 특성을 전달하는 방법에는 두 가지가 있습니다. 첫 번째 방법은 POLICY 테이블의 PROPERTIES 열에 기본 입력 특성을 지정하는 것입니다. 이 테이블에 대한 추가 정보는 다음 절을 참조하십시오.

두 번째 방법은 각 입력 특성에 대해 명령에 새 필드를 작성하는 것입니다. 필드마다 getter 및 setter 메소드의 새 쌍을 작성하십시오.

비즈니스 정책 명령에서 requestProperties 설정

requestProperties가 비즈니스 정책 명령 오브젝트에서 설정되는 두 가지 방식이 있습니다. 첫 번째 방식은 POLICY 테이블의 PROPERTIES 열을 사용하여 기본 특성을 설정합니다. 이것은 setRequestProperties 메소드로 수행됩니다. 특성을 설정하는 두 번째 방식은 비즈니스 정책 명령을 호출하는 명령(제어기 또는 태스크)이 명시적으로 기타 필수 특성을 설정하게 하는 것입니다.

새 비즈니스 정책 명령 작성시, requestProperties 오브젝트에 포함된 각 매개변수를 명시적으로 설정하는 로직을 포함하기 위해 setRequestProperties 메소드를 대체해야 합니다.

MyNewBusinessPolicyCmd의 인터페이스 이름 및 MyNewBusinessPolicyCmdImpl의 구현 클래스 이름을 갖는 새 비즈니스 정책 명령의 예를 고려하십시오.

이 새 비즈니스 정책 명령에 대한 POLICY 테이블의 항목이 다음과 같은 값을 PROPERTIES 열에 포함한다고 가정하십시오.

- defaultProperty1=apple
- defaultProperty2=orange
- defaultProperty3=banana

이 새 비즈니스 정책 명령의 인터페이스는 다음과 같이 정의됩니다.

```
public interface MyNewBusinessPolicyCmd extends
   com.ibm.commerce.command.BusinessPolicyCmd {
      java.lang.String defaultCommandClassName =
         'com.mycompany.mycommands.MyNewBusinessPolicyCmdImpl';
      public void setProperty1();
      public void setProperty2();
}
```

이 새 비즈니스 정책 명령의 구현 클래스는 다음과 같이 정의됩니다.

```
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"));
         }
}
```

새 비즈니스 정책을 호출하는 명령은 다음과 유사한 방식으로 정의될 수 있습니다.

```
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();
}
```

새 비즈니스 정책 및 비즈니스 정책 명령 등록

새 비즈니스 정책 명령을 작성한 후에는 비즈니스 정책 및 비즈니스 정책 명령을 모두 데이터베이스에 등록해야 합니다.

비즈니스 정책은 POLICY 테이블에 등록됩니다. 이 테이블에는 다음과 같은 열이 포함됩니다.

- POLICY_ID
  1차 키. 정책 식별자입니다.
- POLICYNAME
  고유한 정책 이름
- POLICYTYPE_ID
  정책 유형 식별자. POLICYTYPE 테이블에 대한 foreign key입니다.
- STOREENT_ID
  정책이 적용되는 상점 또는 상점 그룹
- PROPERTIES
  비즈니스 정책 명령으로 설정될 수 있는 기본 특성. 이름 값 쌍으로 지정됩니다 (예; `parm1=val1&parm2=val2`).
- STARTDATE
  정책의 시작 날짜(시간소인으로 지정). 널(Null)값인 경우, 시작 날짜는 즉시입니다.
- ENDDATE
  정책의 종료 날짜(시간소인으로 지정). 널(Null)값인 경우, 종료 날짜가 없습니다.

POLICY 테이블에 새 정책이 등록되고 나면 정책과 비즈니스 정책을 구현하는 비즈니스 정책 명령 간의 관계를 등록해야 합니다. POLICYCMD 테이블을 이와 같이 사용할 수 있습니다. POLICYCMD 테이블에는 다음과 같은 열이 포함됩니다.

- POLICY_ID
  POLICY 테이블에 참조하는 foreign key
- BUSINESSCMDCLASS
  정책을 구현하는 비즈니스 정책 명령
- PROPERTIES
  비즈니스 정책 명령으로 설정될 수 있는 기본 특성. 이름 값 쌍으로 지정됩니다 (예; `parm1=val1&parm2=val2`).

규정 오브젝트를 새 비즈니스 정책에 관련짓기

WebSphere Commerce 장기 구매 계약 및 정책 프레임워크에서 규정(*조건*이라고도 함)은 구매자와 판매자 간의 계약을 설명하는 방법을 제공합니다. 규정은 장기 구매 계약 및 RFQ와 같은 다양한 거래 계약 유형에서 사용할 수 있습니다. 규정 오브젝트는 보통 선택 조정으로 비즈니스 정책을 나타냅니다. 예를 들어, 가격 규정 오브젝트는 가격 책정 정책 오브젝트 중 하나를 선택하여 작성됩니다. 가격 규정에서 계정 관리자는 다음과 같은 상점 표준 가격을 조정할 수 있습니다.

- 표준 가격 목록상의 할인율
- 제품 지정 세트의 할인율

각 조정은 규정으로 지정됩니다.

새 비즈니스 정책을 작성할 때, 정책이 장기 구매 계약에 사용될 경우 이 비즈니스 정책을 참조하는 최소한 하나의 규정 오브젝트가 있어야 합니다. 기존 규정 오브젝트를 새 비즈니스 정책에 관련짓거나(`B2BTrading.dtd` 파일에서 이는 기존 규정 오브젝트와 새 비즈니스 정책 사이의 관계를 캡처하여 수행됨) 새 비즈니스 정책에 관련되는 새 규정 오브젝트를 작성할 수 있습니다.

새 규정 작성

WebSphere Commerce 아키텍처 내에서 새 규정 오브젝트는 다음 단계를 수행하여 작성됩니다.

1. 새 규정을 포함하도록 데이터베이스 스키마 갱신
2. 새 규정을 반영하도록 `B2BTrading.dtd` 파일 갱신
3. 규정에 대한 새 엔터프라이즈 bean 작성
4. 새 규정을 반영하도록 WebSphere Commerce 액셀러레이터를 갱신하거나, 장기 구매 계약 로드 명령을 사용하여 새 계약을 사용하는 새 장기 구매 계약 작성

다음 절에서 `MyTC`의 예는 새 규정 오브젝트입니다.

데이터베이스에서 새 규정 등록

새 규정 오브젝트 작성시, 이 오브젝트를 포함하도록 갱신 스키마를 갱신해야 합니다. 갱신되어야 하는 데이터베이스 테이블은 TCTYPE 및 TCSUBTYPE입니다.

다음 SQL 문은 스키마를 갱신하는 방법의 예를 표시합니다.

```
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');
```

장기 구매 계약 문서 유형 정의에서 새 규정 등록

B2BTrading.dtd는 비즈니스 정책 내에서 사용될 수 있는 다양한 규정을 지정하는 문서 유형 정의(DTD) 파일입니다. 장기 구매 계약에서 새 규정을 사용 가능하도록 하려면, 새 규정을 포함하도록 이 파일을 갱신해야 합니다.

새 규정을 작성했으면, 새 규정을 TermCondition 정의에 추가하고 규정을 설명하는 새 요소를 작성해야 합니다.

B2BTrading.dtd 파일을 갱신하려면 다음을 수행하십시오.

1. 다음 디렉토리로 탐색하십시오.
   - Windows *drive*:\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. B2BTrading.dtd 파일을 여십시오.
3. 새 규정으로 TermCondition 정의를 갱신하십시오. 예를 들어, 다음 TermCondition 정의에서 갱신이 굵은체로 표시됩니다.

   ```
   <!ELEMENT TermCondition (TermConditionDescription?,Participant*,
   CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
   PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
   PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
   OrderTC|MyTC))>
   ```

   행 바꾸기는 표시 전용임에 유의하십시오.
4. 이제 새 요소를 B2BTrading.dtd 파일에 추가하십시오. 예를 들어, 다음을 비즈니스 정책을 참조하고 두 개의 필수 속성을 가지고 있는 MyTC 요소를 추가하기 위한 갱신을 보여줍니다.

```
<!ELEMENT MyTC (MySubTC)>
<!ELEMENT MySubTC (PolicyReference)>
<!ATTLIST MySubTC
   attr1 CDATA #REQUIRED
   attr2 CDATA #REQUIRED
>
```

5. 파일을 저장하십시오.

규정에 대한 새 CMP 엔터프라이즈 bean 작성

규정 오브젝트에 대한 새 CMP 엔터프라이즈 bean을 작성해야 합니다. 규정 부속 유형에 대해 bean이 작성됩니다.

일반적으로 새 엔터프라이즈 Bean을 작성할 때 WebSphere Commerce 엔티티 bean을 포함하는 EJB 그룹 중 하나에 포함하기 보다는 사용자 고유의 EJB 그룹에 Bean을 놓습니다. 그러나 이러한 경우, 규정에 대한 새로운 모든 엔티티 bean이 WebSphere Commerce TermCondition bean으로부터 계승되어야 하므로, 새 규정을 WCS 장기 구매 계약 EJB 그룹에 놓아야 합니다.

규정 오브젝트에 대한 새 CMP 엔터프라이즈 bean을 작성하려면, VisualAge for Java에서 다음을 수행하십시오.

1. 마법사에서 다음을 수행하여 새 엔터프라이즈 bean을 작성하십시오.
   a. VisualAge for Java 워크벤치에서 EJB 탭을 선택하십시오.
   b. **WCS 장기 구매 계약** EJB 그룹을 강조표시한 후 마우스 오른쪽 버튼을 클릭하고 **추가 > 계승되는 엔터프라이즈 Bean**을 선택하십시오.
      계승되는 엔터프라이즈 Bean 작성 SmartGuide가 열립니다.
   c. SmartGuide에서 bean에 대한 해당 정보를 입력하십시오. 예를 들어, 다음 표는 예 값을 보여줍니다.

속성	값
bean 이름	MySubTC
계승원	TermCondition
패키지	com.ibm.commerce.contract.objects
Bean 클래스	MySubTCBean
원격 인터페이스	MySubTC
홈 인터페이스	MySubTCHome

d. CMP 필드를 bean에 추가하기 위해 **추가**를 누르고 필요에 따라 bean에 대해 새 필드를 작성하십시오. 이 예의 경우, 다음 정보를 사용하여 두 가지의 새 CMP 필드가 작성됩니다.

상태	값
필드 이름	attr1
필드 유형	String
getter 및 setter 메소드로 액세스	사용 가능
getter 및 setter 메소드를 원격 인터페이스로 승격	사용 가능

상태	값
필드 이름	attr2
필드 유형	Integer
getter 및 setter 메소드로 액세스	사용 가능
getter 및 setter 메소드를 원격 인터페이스로 승격	사용 가능

e. **완료**를 누르십시오.

2. 다음 단계는 TERMCOND 테이블에서 새 bean으로부터 열로 필드를 맵핑하는 것입니다. 이 맵핑 정보를 작성하려면 다음을 수행하십시오.
   a. **EJB** 메뉴에서 **열기 > 스키마 맵**을 선택하십시오.
      맵 브라우저가 열립니다.
   b. 맵 브라우저의 데이터스토어 맵 패널에서 **WCS 장기 구매 계약**을 두 번 누르기하십시오.
   c. 지속 클래스 패널에서 **TermCondition**을 두 번 누르기한 후 **MySubTC**를 선택하십시오.
   d. 테이블 맵 메뉴에서 **새 테이블 맵 > 단일 계승 테이블 맵 추가**를 선택하십시오.
      단일 계승 테이블 맵 편집기가 열립니다.
   e. **식별자 값** 필드에 TCSUBTYPE_I 값을 입력하십시오. 예를 들어, 이 경우 'MySubTC'(따옴표 포함)를 입력하고 **확인**을 누르십시오.

f. 지속 클래스 패널에서 **MySubTC**가 계속 선택되어 있는지 확인하십시오. 테이블 맵 패널에서 강조표시하고 TERMCOND 테이블을 마우스 오른쪽 버튼으로 누르기하십시오. 특성 맵 편집을 선택하십시오.
특성 맵 편집기가 열립니다.

g. 특성 맵 편집기에서 속성을 다음과 같이 설정하십시오.

클래스 속성	맵 유형	테이블 열
attr1	Simple	STRINGFIELD2
attr2	Simple	INTEGERFIELD1

확인을 누르십시오.

h. 데이터스토어 맵 메뉴에서 데이터스토어 맵 저장을 선택하십시오. 맵 저장 시 다음 정보를 입력하십시오.

속성	값
프로젝트	IBM WCS Enterprise Beans
패키지	WCSContract EJB Reserved
클래스 이름	WCSContractMap

완료를 누르고 맵 브라우저를 닫으십시오.

3. 새 엔터프라이즈 bean(즉 MySubTCBean)에서 다음과 같이 새 `ejbCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement)` 메소드를 작성하십시오.

```
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. 다음과 같이 새 `ejbPostCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement)` 메소드를 작성하십시오.

```
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. 다음과 같이 MySubTCBean에서 parseXMLElement(org.w3c.dom.Element argElement) 메소드를 대체하십시오.

```
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. 다음과 같이 MySubTCBean에서 createNewVersion(Long argNewTradingId) 메소드를 대체하십시오.

```
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. 다음과 같이 MySubTCBean에서 getXMLString() 메소드를 대체하십시오.

```
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. 다음과 같이 MySubTCBean에서 markForDelete() 메소드를 대체하십시오.

```
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. ejbCreate 메소드가 홈 인터페이스에 추가되었고 다른 모든 수정된 메소드가 원격 인터페이스에 추가되었는지 확인하십시오.

10. 다음 단계는 다음을 수행하여 MySubTC 엔티티 bean에 대해 액세스 bean을 작성하는 것입니다.
    a. **MySubTC** 엔티티 bean을 마우스 오른쪽 버튼으로 누르고 **추가 > 액세스 Bean**을 선택하십시오.
    액세스 Bean 작성 SmartGuide가 열립니다.
    b. 다음 정보가 입력되었는지 확인하십시오.

표 1.

속성	값
EJB 그룹	WCSContract
엔터프라이즈 bean	MySubTC
액세스 bean 이름	MySubTCAccessBean
액세스 bean 유형	Copy Helper for an Entity Bean

    다음을 누르십시오.
    c. **인수없는 구조체의 홈 메소드 선택** 드롭 다운 목록에서 **findByPrimaryKey(TermConditionKey)**를 선택하십시오.
    d. **initKey_referenceNumber**의 경우(초기 특성 열에서), **변환기**를 com.ibm.commerce.base.objects.WCStringConverter로 설정한 후 **다음**을 누르십시오.
    e. 추가한 모든 새 필드에 대해 **CopyHelper**가 선택되어 있는지 확인하고 각각의 변환기 값을 com.ibm.commerce.base.objects.WCStringConverter로 설정하십시오. **완료**를 누르십시오.
    코드 생성이 완료된 후, **프로젝트** 탭으로 전환하십시오. **IBM WCS 엔터프라이즈 Bean** 프로젝트를 펼친 후 **com.ibm.commerce.contract.objects** 패키지를 펼쳐서 새 노드를 볼 수 있습니다.
11. EJB 탭으로 다시 가서, **MySubTC** 엔터프라이즈 bean을 마우스 오른쪽 버튼으로 누르고 **전개된 코드 생성**을 선택하십시오.
12. 상위 bean(TermCondition bean)과 모든 동위 bean (이름에 “TC”를 포함하는 WCS 장기 구매 계약 EJB 그룹에 있는 다른 모든 bean)에 대해서도 전개된 코드를 다시 생성해야 합니다. 새 필드를 추가했거나 기존 TermCondition bean의 원격 인터페이스를 수정한 경우, 자체와 해당되는 모든 하위 bean에

대해 액세스 bean을 다시 생성해야 합니다.
전개 코드를 재생성하려면 다음을 수행하십시오.

a. TermCondition bean과 이름에 "TC"를 포함하는 다른 모든 bean을 강조표시하십시오(예를 들어, DisplayCustomizationTC, FulfillmentTC, InvoiceTC는 동위 bean의 일부입니다).

b. 이러한 모든 bean을 강조표시하고, 마우스 오른쪽 버튼을 누른 후 **전개된 코드 생성**을 선택하십시오.

13. 다음 단계는 `ValidateContractCmd` 태스크 명령에 있는 메소드를 대체하는 것입니다. 이 명령에는 새 규정 오브젝트를 지원하기 위해 대체하려고 할 수 있는 세 가지 메소드가 있습니다. 다음과 같습니다.

- `validateTCType()`
  이 메소드는 어떤 유형의 규정이 장기 구매 계약에 있을 수 있는지 확인합니다. 예를 들어 InvoiceTC는 계정에 속하므로 장기 구매 계약에서 표시할 수 없습니다.
- `validateTCOccurrence()`
  이 메소드는 규정의 발생을 확인합니다. 예를 들어, 이 메소드의 기본 구현에서 장기 구매 계약은 최소한 하나의 PriceTC를 가져야 합니다.
- `otherValidateCheck()`
  이 메소드의 기본 구현은 비어 있습니다. 처음 두 가지 메소드에 속하지 않는 추가적인 유효성 검증을 추가할 수 있습니다.

14. 규정이 전개되어야 하는 경우, 새 전개 명령을 작성하고 이 명령을 데이터베이스에 등록해야 합니다. 필요한 경우, 다음을 수행하십시오.

a. 이 예에서 새 전개 명령 인터페이스를 `MySubTCDeployedCmd`라고 하며 구현 클래스를 `MySubTCDeployedCmdImpl`라고 합니다. 그런 다음 명령은 `packagename` 패키지로 패키지화됩니다. 이 명령을 등록하려면, 다음 SQL 명령을 실행하십시오.

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, CLASSNAME, TARGET)
values (0, 'packagename.MySubTCDeployCmd',
'packagename.MySubTCDeployCmdImpl', 'Local');
```

b. packagename 패키지에서 새 MySubTCDeployedCmd 인터페이스를 작성하십시오. 이 인터페이스는 com.ibm.commerce.contract.commands.DeployTCCmd 명령 인터페이스를 확장해야 합니다. 다음은 새 명령 인터페이스를 설명합니다.

```
public interface MySubTCDeployCmd extends
   com.ibm.commerce.contract.commands.DeployTCCmd
   {
       // customized code
   }
```

보호된 매개변수 abTC와 getTargetStoreId()라는 메소드가 DeployTCCmd에 있습니다. abTC 값은 MySubTCAccessBean이며 getTargetStoreId() 메소드는 장기 구매 계약이 전개 중인 상점의 식별자를 리턴합니다.

c. 동일한 패키지에서 MySubTCDeployCmdImpl 구현 클래스를 작성하십시오. 이 구현 클래스는 com.ibm.commerce.contract.commands.DeployTCCmdImpl을 확장해야 합니다. 다음은 새 명령 구현 클래스를 설명합니다.

```
public class MySubTCDeployCmdImpl
   extends com.ibm.commerce.contract.commands.DeployTCCmdImpl
   implements MySubTCDeployCmd
   {
     // customer code
   }
```

새 규정을 사용하도록 WebSphere Commerce 액셀러레이터 갱신

새 규정을 작성하고 나면 WebSphere Commerce 액셀러레이터를 수정하여 새 규정을 포함하는 새 장기 구매 계약을 작성하는데 사용할 수 있습니다. 이러한 목적으로 WebSphere Commerce 액셀러레이터를 갱신하는 데에는 다음 단계가 포함됩니다.

1. 새 규정에 대해 새 JavaScript 파일을 작성. 이 절에 있는 예의 목적에 따라, 이 파일을 Extensions.js라고 합니다.
2. 사용자가 새 규정에 대해 필요한 정보를 입력할 수 있는 HTML 섹션을 포함하는 새 JSP 템플리트 작성. 이 절에 있는 예의 목적에 따라, 이 파일을 ContractMyTC.jsp라고 합니다.

3. 새 규정에 대해 새 데이터 bean 작성. 이 절에 있는 예의 목적에 따라, 이 파일을 MyTCDataBean이라고 합니다.
4. VIEWREG 테이블에 새 보기 등록.
5. 새 자원을 포함하도록 ContractRB_*locale*.properties 파일 갱신.
6. 새 페이지를 포함하도록 ContractNotebook.xml 파일 편집.

각 단계는 다음 절에 자세히 설명되어 있습니다.

새 JavaScript 파일 작성: 규정을 사용하기 위해 WebSphere Commerce 액셀러레이터를 갱신하는 첫 번째 단계는 그 규정에 대한 새 JavaScript 파일을 작성하는 것입니다. 참조용으로 다음 견본 파일을 참조할 수 있습니다.

- Windows *drive*:\WebSphere\CommerceServer\samples\contract\Extensions.js
- Windows *drive*:\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

이 견본 파일을 사용하려면, 이를 다음 디렉토리로 복사하십시오.

- Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wctools.war\javascript\tools\contract

- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/javascript/tools/contract
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/javascript/tools/contract
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/javascript/tools/contract
- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/javascript/tools/contract

이 새 파일에서는 새 규정에 대한 데이터를 저장할 JavaScript 오브젝트를 작성해야 합니다. 이는 다음 코드 발췌 부분에 나와 있습니다.

```
function ContractMyTCModel() {

   this.tcReferenceNumber = "";
   this.policyReferenceNumber = "";

   this.attr1 = "";
   this.attr2 = "";

   this.policyList = new Array();
   this.selectedPolicyIndex = "0";
}
```

새 규정을 제출하기 위해 새 JavaScript 오브젝트도 작성해야 합니다. 이는 B2BTrading.dtd 파일에 대해 수행한 확장과 일관되는 방식으로 수행해야 합니다. 이는 다음 코드 발췌 부분에 나와 있습니다.

```
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;
}
```

새 JSP 템플리트 작성: 다음 단계는 사용자가 새 규정에 대해 필요한 정보를 입력할 수 있는 HTML 섹션을 포함하는 새 JSP 템플리트를 작성하는 것입니다. 참조용으로 다음 견본 파일을 참조할 수 있습니다.

- Windows *drive*:\WebSphere\CommerceServer\samples\contract\ContractMyTC.jsp
- Windows *drive*:\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

이 견본 파일을 사용하려면, 이를 다음 디렉토리로 복사하십시오.

- Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wctools.war\tools\contract
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract
- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wctools.war/tools/contract

다음 코드 발췌 부분은 MyTC에 사용할 수 있는 JSP 템플리트의 HTML 섹션 예입니다.

```
<!--
////////////////////////////////////////
// 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>
```

새 데이터 bean 작성: 이 단계에서는 MySubTC 액세스 bean으로부터 필요한 데이터를 로드하는 새 데이터 bean을 작성합니다. 관련된 코드 섹션이 다음 코드 발췌 부분에 표시됩니다.

```
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;
            }

}
```

VIEWREG 테이블에 새 보기 등록: VIEWREG 테이블에 새로 작성한 보기를 등록해야 합니다. 다음은 새 보기를 등록하기 위한 SQL 문의 예입니다.

```
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)
```

ContractRB_locale.properties **파일 갱신:** 다음 특성 파일을 새 규정에 고유한 정보로 갱신해야 합니다.

- Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\properties\com\ibm\commerce\tools\contract\properties\ ContractRB_*locale*.properties
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ ContractRB_*locale*.properties

- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*locale*.properties

다음은 파일에 추가하는 정보의 예입니다.

```
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
```

***ContractNotebook.xml* 파일 편집:** 새 규정을 WebSphere Commerce 액셀러레이터에 포함시키기 위한 마지막 단계는 새 페이지를 포함하기 위해 다음 파일을 갱신하는 것입니다.

- Windows *drive*:\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/*instanceName*/xml/tools/contract/ContractNotebook.xml

다음은 이 예에 새 페이지를 포함시키기 위해 사용되는 코드 발췌 부분의 예입니다.

```
<panel name="MyTCHeading"
    url="ContractMyTCPanelView"
    parameters="contractId,accountId"
    helpKey="MC.contract.MyTCPanel.Help" />
```

새 규정을 사용하여 새 장기 구매 계약 반입

새 규정을 사용하기 위해 WebSphere Commerce 도구를 갱신하는 방법의 대안으로, 장기 구매 계약 반입 명령(이 명령에 대한 정보는 WebSphere Commerce 온라인 도움말 참조)을 사용하여 새 규정을 포함하는 새 장기 구매 계약을 반입할 수 있습니다. 반입 후, Contract.xml 파일의 관련 섹션은 다음과 같습니다.

```
<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>
```

새 비즈니스 정책 호출

새 비즈니스 정책을 작성하고 이 비즈니스 정책이 최소한 하나의 규정 오브젝트와 연관된 경우, 새 비즈니스 정책 명령을 호출하도록 응용프로그램 로직을 갱신해야 합니다.

비즈니스 정책 명령은 제어기 및 태스크 명령 내에서 호출됩니다.

명령 팩토리는 비즈니스 정책 명령을 호출하는 데 사용됩니다. 비즈니스 정책 명령을 호출하기 위해 사용될 수 있는 두 개의 create 메소드가 있습니다. 첫 번째 메소드는 단 하나의 비즈니스 정책 명령이 비즈니스 정책과 연관되어 있을 경우 비즈니스 정책 명령을 호출하는 데 사용됩니다. 이것은 다음 코드로 표시됩니다.

```
CommandFactory createBusinessPolicyCommand(Long policyId);
```

두 번째 메소드는 둘 이상의 비즈니스 정책 명령이 비즈니스 정책과 연관되어 있을 경우 비즈니스 정책 명령을 호출하는 데 사용됩니다. 이것은 다음 코드로 표시됩니다.

```
CommandFactory createBusinessPolicyCommand(Long policyId, String cmdIfName);
```

앞의 예에서, cmdIfName은 작성되는 비즈니스 정책 명령의 인터페이스 이름을 지정하기 위해 사용됩니다.

명령 팩토리는 POLICYCMD 테이블의 정책 오브젝트를 찾아보고 이 정책을 구현하는 명령을 결정합니다. 또한 테이블로부터 모든 기본 특성을 가져와서 비즈니스 정책 명령에 requestProperties로 설정합니다.

다음 코드는 반환 정책을 호출하는 예제입니다.

```
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()
```

장기 구매 계약 작성

장기 구매 계약의 비즈니스 처리로의 확장을 완전히 통합하기 위한 다음 단계는 새 비즈니스 정책을 참조하는 규정을 포함하는 장기 구매 계약을 작성하는 것입니다. 장기 구매 계약은 WebSphere Commerce 액셀러레이터를 사용하거나 장기 구매 계약 URL 명령(ContractImportApprovedVersion 및 ContractImportDraftVersion) 중 하나를 사용하여 작성할 수 있습니다. 장기 구매 계약 작성에 대한 추가 정보는 WebSphere Commerce 온라인 도움말을 참조하십시오.

장기 구매 계약 사용자 정의 시나리오

이 절에서는 다음의 장기 구매 계약 사용자 정의 시나리오에 관련되는 단계들의 개요를 제공합니다.

- 리베이트 사용

리베이트 시나리오

이 시나리오 예에서, 플랫 비율 리베이트가 작성됩니다. ToolTech 견본 상점에는 리베이트 시나리오와 일치하는 규정이나 정책 유형이 포함되어 있지 않으므로 이를 작성해야 합니다. 또한 새 비즈니스 정책과, 리베이트 코드를 저장할 데이터베이스 테이블도 작성해야 합니다.

이러한 리베이트 구현 시나리오에는 다음과 같은 상위 레벨 단계가 포함됩니다.

1. XREBATECODE 데이터베이스 테이블과, 이 테이블의 정보에 액세스하기 위해 사용되는 해당되는 XRebateCodeBean 엔티티 bean 작성.
2. 다음의 부속 태스크를 수행하여 새 5DollarRebate 비즈니스 정책 작성.
   a. 해당되는 새 비즈니스 정책 유형을 작성합니다. 이는 새 비즈니스 정책 명령이 구현할 인터페이스(RebatePolicyCmd)를 정의합니다.
   b. 새 CalculateRebateCmdImpl 비즈니스 정책 명령을 작성합니다.
   c. 데이터베이스에서 새 비즈니스 정책 명령과 비즈니스 정책 유형을 등록합니다.
3. 다음의 부속 태스크를 수행하여 리베이트에 대한 새 규정(RebateTC) 작성.
   a. 데이터베이스에 RebateTC 용어 및 조건 등록
   b. 새 RebateTC를 반영하도록 `B2BTrading.dtd` 파일 갱신
   c. RebateTC용으로 새 엔터프라이즈 작성
   d. 새 RebateTC를 반영하도록 WebSphere Commerce 액셀러레이터 갱신
4. RebateTC를 작성하는 새 장기 구매 계약 작성.
5. 쇼핑 플로우로 새 비즈니스 정책 통합.

각 단계는 다음 절에 자세히 설명되어 있습니다.

단계 1: 새 테이블 및 엔터프라이즈 bean 작성

기존의 데이터베이스 스키마에는 리베이트 금액 및 코드 스펙이 없으므로, 새 테이블을 작성해야 합니다. 일반적으로 새 테이블이 작성될 때, 이 테이블에 포함되는 정보에 액세스할 때 사용하는 새 엔티티도 작성됩니다.

이 예의 목적에 따라, 다음과 같은 XREBATECODE 데이터베이스 테이블이 작성된다고 가정합시다.

표 2. XREBATECODE 데이터베이스 테이블

	열 이름		
	REBATECODE_ID	AMOUNT	CURRENCY
견본 데이터	201	5	CAD
	202	10	CAD

또한 새 CMP 엔티티 bean(XRebateCodeBean)도 작성됩니다. 이 bean 작성에 대한 자세한 정보는 68 페이지의 『새 CMP 엔터프라이즈 bean 작성』을 참조하십시오.

2단계: "5DollarRebate" 비즈니스 정책 작성

이러한 새 비즈니스 정책을 작성하려면 다음 단계를 수행해야 합니다.

1. 새 비즈니스 정책 유형 인터페이스를 작성하십시오. 이는 `CalculateRebateCmdImpl`이 구현할 `RebatePolicyCmd` 인터페이스입니다.
2. 새 `CalculateRebateCmdImpl` 비즈니스 정책 명령을 작성하십시오.
3. 데이터베이스에 새 비즈니스 정책 및 비즈니스 정책 명령을 등록하십시오.

"리베이트" 비즈니스 정책 유형 작성: 리베이트에 해당하는 기존 비즈니스 정책 유형이 없으므로, 새 비즈니스 정책 유형을 작성해야 합니다. 새 비즈니스 정책 유형 작성에는 데이터베이스에서의 정책 유형 정의 및 등록이 포함됩니다. 다음 테이블은 갱신되어야 합니다.

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

이 시나리오의 경우, 새 REBATE 정책 유형을 작성하려면 다음 SQL 문이 사용됩니다.

```
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.');
```

결과로, 다음 표는 정책 유형과, 이 유형이 관련되는 비즈니스 정책 명령 사이의 관계를 표시하는 PLCYTYCMIF 테이블의 관련된열을 보여줍니다.

표 3. PLCYTYCMIF 테이블에 대해 수행된 갱신사항

	열 이름	
	POLICYTYPE_ID	BUSINESSCMDIF
견본 데이터	리베이트	com.mycompany.mybusinesspolicycommands.RebatePolicyCmd

새 RebatePolicyCmd 인터페이스도 코드화해야 합니다. 이 인터페이스는 com.ibm.commerce.command.BusinessPolicyCommand 인터페이스를 확장해야 합니다. 이전 표에 제시된 대로, 이 인터페이스를 사용자 고유 패키지에 묶으십시오.

CalculateRebateCmdImpl 비즈니스 정책 명령 작성: 새 비즈니스 정책 명령을 작성하려면, com.ibm.commerce.command.BusinessPolicyCommandImpl 구현 클래스를 확장하는 CalculateRebateCmdImpl라는 새 명령을 작성해야 합니다. 이 명령은 이전 단계에서 작성된 RebatePolicyCmd 인터페이스를 구현해야 합니다.

이 예에서, 인터페이스 이름과 명령 이름은 식별자입니다. 이 이름은 비즈니스 정책의 리베이트 유형을 구현하는 많은 비즈니스 정책 명령이 있을 수 있음을 표시하기 위해 선택된 것입니다. 각 구현(즉, 각 비즈니스 정책 명령)은 고유한 방식으로 리베이트를 구현합니다.

명령의 로직은 고객이 상품을 픽업하는 방법에 대한 특정 구현에 따라 다릅니다. 또한 이 CalculateRebateCmdImpl은 사용자 응용프로그램에서 별도의 제어기나 태스크 명령으로 호출해야 합니다.

새 비즈니스 정책 및 새 비즈니스 정책 명령 등록: 새 비즈니스 정책은 데이터베이스에 등록해야 합니다. 또한 새 비즈니스 정책과 새 비즈니스 정책 명령 사이의 관계도 등록해야 합니다.

이 정보를 등록하기 위해 com.ibm.commerce.contract.commands.PolicyAddCmd 명령을 사용할 수 있습니다. 다음은 이 시나리오에 대한 PolicyAdd 명령 사용법의 예를 표시합니다.

```
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
```

URL 예약 문자는 입력 등록 정보에 대해 해당되는 ASCII 코드로 바꿔야 합니다. 이에 따라, 일반적인 =(등호)는 “%3D”로, &(앰퍼샌드)는 “%26”으로, 공백 문자는 “%20”으로 바뀝니다.이전 예에서 사용된 날짜 형식은 yyyy-mm-dd hh:mm:ss이며, ASCII 코드로 URL 예약 문자를 바꿉니다.

다음 표는 갱신 수행한 영향을 받는 데이터베이스 테이블의 관련된열을 보여줍니다.

표 4. POLICY 테이블에 대해 수행된 갱신사항

	열 이름				
	POLICY_ID	POLICY NAME	POLICYTYPE_ID	STOREENT_ID	PROPERTIES
견본 데이터	301	5Dollar 리베이트	리베이트	-1	rebatecode_id= 201

시작 날짜와 종료 날짜 값이 null로 설정되어 있다고 가정한 점에도 유의하십시오.

표 5. POLICYCMD 테이블에 대해 수행된 갱신사항

	열 이름		
	POLICY _ID	BUSINESS CMDCLASS	PROPERTIES
견본 데이터	301	com.mycompany.mybusinesspolicycommands.CalculateRebateCmdImpl	널(NULL)값

결과로, 이제는 CalculateRebateCmd 비즈니스 정책 명령에 관련되는 “5DollarRebate”라고 하는 새 비즈니스 정책을 갖게 됩니다.

3단계: “RebateTC” 규정 작성

“RebateTC” 규정 작성에는 다음 단계를 수행해야 합니다.

1. 데이터베이스에 RebateTC 용어 및 조건 등록
2. 새 RebateTC를 반영하도록 B2BTrading.dtd 파일 갱신

3. RebateTC용으로 새 엔터프라이즈 작성
4. 새 RebateTC를 반영하도록 WebSphere Commerce 액셀러레이터 갱신

데이터베이스에서 "RebateTC" 규정 등록: 새 규정 오브젝트 작성시, 이 오브젝트를 포함하도록 갱신 스키마를 갱신해야 합니다. 갱신되어야 하는 데이터베이스 테이블은 TCTYPE 및 TCSUBTYPE입니다.

다음 SQL 문은 데이터베이스에 RebateTC를 등록하는 방법의 예를 표시합니다.

```
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);
```

다음 표는 TCTYPE 및 TCSUBTYPE 테이블에서 관련된 열을 추출한 것을 보여줍니다.

표 6. TCTYPE 테이블에 대해 수행된 갱신사항

	열 이름
	TCTYPE_ID
견본 데이터	RebateTC

표 7. TCSUBTYPE 테이블에 대해 수행된 갱신사항

	열 이름			
	TCSUBTYPE _ID	TCTYPE _ID	ACCESSBEAN NAME	D E P L O Y COMMAND
견본 데이터	RebateTC	RebateTC	com.ibm.commerce. contract. objects. RebateTCAccessBean	널(NULL)값

새 RebateTC를 반영하도록 B2BTrading.dtd 파일 갱신: 장기 구매 계약에서 새 규정을 사용 가능하도록 하려면, 새 규정을 포함하도록 `B2BTrading.dtd` 파일을 갱신해야 합니다. 이 파일을 갱신할 때, 새 규정을 TermCondition 정의에 추가하고 이 규정을 설명하는 새 요소를 작성해야 합니다.

굵게 표시된 텍스트는 새 RebateTC를 TermCondition 정의에 추가하는 방법에 대한 예를 보여줍니다.

```
<!ELEMENT TermCondition (TermConditionDescription?,Participant*,
CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
OrderTC|RebateTC))>
```

행 바꾸기는 표시 전용임에 유의하십시오.

그런 다음 이 규정을 설명하는 새 스탠자를 파일에 추가해야 합니다. 다음은 RebateTC의 예를 표시합니다.

```
<!ELEMENT RebateTC (PolicyReference?)>
```

RebateTC용 새 엔터프라이즈 bean 작성: 새 RebateTC에 대한 새 엔터프라이즈 bean을 작성해야 합니다. 새 bean은 WebSphere Commerce TermCondition bean에서 계승되어야 합니다.

규정에 대한 새 엔터프라이즈 bean의 이름은 일반적으로 부속 유형 다음에 지정됩니다. 이러한 경우, 규정 부속 유형은 규정 유형과 같으므로, bean의 이름은 규정 유형과 같게 됩니다.

다음 표는 작성해야 하는 새 bean에 대한 일반 정보 중 일부를 보여줍니다. 대체할 메소드를 포함하여, Bean에 대한 자세한 내용은 184 페이지의 『규정에 대한 새 CMP 엔터프라이즈 bean 작성』 부분을 참조하십시오.

표 8.

속성	값
bean 이름	RebateTC
계승원	TermCondition
패키지	com.ibm.commerce.contract.objects
Bean 클래스	RebateTCBean
원격 인터페이스	RebateTC
홈 인터페이스	RebateTCHome

새 RebateTC를 반영하도록 WebSphere Commerce 액셀러레이터 갱신: 새 규정을 작성하고 나면 WebSphere Commerce 액셀러레이터를 수정하여 새 규정을 포함하는 새 장기 구매 계약을 작성하는데 사용할 수 있습니다. 이 도구를

업데이트하는 방법에 대한 정보는 191 페이지의 『새 규정을 사용하도록 WebSphere Commerce 액셀러레이터 갱신』을 참조하십시오.

단계 4: 새 장기 구매 계약 작성

"RebateTC" 규정을 포함하고 "5DollarRebate" 비즈니스 정책을 참조하는 새 장기 구매 계약을 작성해야 합니다. WebSphere Commerce 액셀러레이터 또는 XML을 사용하여 새 장기 구매 계약을 작성할 수 있습니다. 새 장기 구매 계약을 작성하기 위한 이러한 방법은 각각 WebSphere Commerce 온라인 도움말에 설명되어 있습니다.

다음 표는 장기 구매 계약이 작성된 후 TERMCOND 및 POLICYTC 데이터베이스 테이블의 관련된 열에 대한 갱신사항을 보여줍니다.

표 9. TERMCOND 테이블에 대해 수행된 갱신사항

	열 이름		
	TRADING _ID	TERMCOND _ID	TCSUBTYPE _ID
견본 데이터	25	901	RebateTC

표 10. POLICYTC 테이블에 대해 수행된 갱신사항

	열 이름	
	POLICY _ID	TERMCOND_ID
견본 데이터	301	901

5단계: 쇼핑 플로우에 비즈니스 정책 통합

이 시나리오에서는 고객이 로그온하여 리베이트를 청구할 수 있는 새 페이지가 상점에 추가된다고 가정합니다. 고객이 리베이트를 청구하기 위해 누르면, 새 `RebatePolicyCmd` 인터페이스를 호출하는 명령이 호출되어야 합니다. 예를 들어, `RebatePolicyCmd`를 호출하는 새 ClaimRebateCmd 제어기 명령이 있을 수 있습니다. 그러면 해당되는 비즈니스 정책이 발견되고 (이 경우) "5DollarRebate" 비즈니스 정책이 적용됩니다.

제 3 부 개발 환경

제 8 장 개발 도구 및 전개

이 장에서는 WebSphere Commerce 응용프로그램을 사용자 정의하는 데 사용되는 기본 개발 도구를 소개합니다. VisualAge for Java에서 WebSphere Commerce Server에 이르기까지 사용자 정의 코드 전개 처리에 대해 설명합니다. 또한 JSP 템플리트 개발시, 사용자 정의 코드를 이용할 수 있도록 Commerce Studio로 코드를 전개하는 방법도 설명합니다.

개발 환경

WebSphere Commerce Business Edition에서 사용할 사용자 정의 코드 작성을 위한 권장 개발 패키지는 WebSphere Commerce Studio, Business Developer Edition 제품입니다. WebSphere Commerce Professional Edition에서 사용할 사용자 정의 코드 작성을 위한 권장 개발 패키지는 WebSphere Commerce Studio Professional Developer Edition 제품입니다. 이 두 가지 패키지는 모두 사용자 정의 코드 작성 및 웹 개발 태스크 수행에 필요한 모든 도구를 포함합니다.

WebSphere Commerce Studio Business Developer Edition 및 WebSphere Commerce Studio Professional Developer Edition은 VisualAge for Java의 WebSphere Test Environment 구성요소에서 사용하는 견본 WebSphere Commerce 상점 포함 옵션을 제공합니다. 이렇게 하면 개발자가 WebSphere Commerce 도구를 사용하여 상점을 작성한 다음 상점 자원을 개발 환경으로 이동할 필요가 없어지기 때문에 개발 환경 구성이 간소화됩니다.

개발 환경에 대한 다른 키 구성요소는 WebSphere Commerce 코드의 저장소입니다. 이 저장소는 상품 설치가 완료된 후에 VisualAge for Java 작업 영역으로 반입되어야 합니다. 이 저장소를 반입한 후에 개발자는 WebSphere Test Environment와 관련된 일부 구성 단계를 수행해야 합니다.

모든 설치 및 구성 태스크가 완료된 후에 개발자는 사용자 정의된 WebSphere Commerce 코드가 작성되고 테스트될 수 있는 독립형 개발 시스템을 갖습니다. 개발자는 개발 시스템에 WebSphere Commerce를 설치하지 않아도 됩니다.

WebSphere Commerce Studio

WebSphere Commerce Studio Business Developer Edition 및 WebSphere Commerce Studio Professional Developer Edition에는 모두 VisualAge for Java, Enterprise Edition, 버전 4.0이 포함됩니다. 이 VisualAge for Java 개정판에는 상점 입구 자원 개발을 돕는 고급 JSP 템플리트를 개발 및 디버깅하기 위한 강력한 도구 기능이 있습니다. 또한 WebSphere Studio 와 통합되어 이 JSP 템플리트에 컨텐츠를 신속하게 추가할 수 있도록 함으로써, 프로그래머 및 웹 개발자의 생산성을 높힐 수 있습니다. 백오피스 응용프로그램 코드 개발을 위해 Enterprise JavaBeans 기술을 지원하고 CICS® TS, MQSeries 및 SAP R/3 등과 같은 다른 시스템으로의 통합을 지원하는 연결 기능이 있습니다. 또한 VisualAge for Java, Enterprise Edition의 통합 WebSphere Test Environment를 사용하여 개발자는 VisualAge for Java를 종료하지 않고 WebSphere Commerce 기능을 실행할 수 있습니다. 즉, WebSphere Commerce Server로 코드를 전개하지 않고도 사용자 정의 WebSphere Commerce 코드를 테스트할 수 있음을 나타냅니다.

주: VisualAge for Java, Enterprise Edition, 버전 4.0의 WebSphere Test Environment 구성요소는 WebSphere Application Server V3.5.4에서 응용프로그램을 실행합니다. WebSphere Commerce Business Edition 및 WebSphere Commerce Professional Edition은 두 가지 모두 WebSphere Application Server V4.0을 사용합니다. 이러한 버전 차이에 의해 WebSphere Application Server V4.0에서 실행하는 WebSphere Commerce 인스턴스에 새 엔터프라이즈 bean 또는 수정 엔터프라이즈 bean을 전개하려면 WebSphere Application Server V4.0과 함께 제공되는 EJBDeploy 도구를 사용하여 VisualAge for Java 외부에서 전개 코드를 생성해야 합니다. 실제로 이 전개 코드는 WebSphere Application Server V4.0을 설치한 시스템에서 생성해야 합니다. 자세한 정보는 214 페이지의 『EJB 전개 코드에 대한 정보』를 참조하십시오.

227 페이지의 제 4 부 『학습』에서 설명한 학습을 완료하려면 WebSphere Commerce Studio Business Developer Edition 또는 WebSphere Commerce Studio Professional Developer Edition을 설치해야 합니다. Commerce Studio 설치 및 VisualAge for Java 구성에 대한 추가 정보는 *WebSphere Commerce*

Studio Business Developer Edition 설치 안내서 또는 *WebSphere Commerce Studio Professional Developer Edition 설치 안내서*를 참조하십시오.

VisualAge for Java의 특징 및 함수

이 *프로그래머 안내서*는 VisualAge for Java를 사용하는 방법을 알려주기 위한 것이 아닙니다. 해당 상품에 대해, WebSphere Commerce용 전자 상거래 응용프로그램을 작성하기 위한 프로그래밍 태스크를 수행하는 하나의 방법으로서 VisualAge for Java에서 태스크를 수행하는 방법을 일반적으로 보여주고 있습니다. 만약 VisualAge for Java의 새로운 사용자이면 이 책에 포함된 학습을 수행하여 VisualAge for Java에서 태스크를 수행하는 방법을 먼저 배우게 됩니다. 그러나 이 도구를 전문가처럼 사용하려면 VisualAge for Java 문서, 학습 및 과정들을 참조해야 합니다.

예를 들어, (선택적) VisualAge for Java에서 디버거 사용 학습 주제는 디버거를 사용하여 코드 실행 중 태스크 명령의 변수값을 볼 수 있는 방법에 대한 빠른 소개를 제공합니다. VisualAge for Java의 디버거 구성요소는 강력한 디버깅 도구입니다. 특징 및 사용 방법에 대한 자세한 내용은 VisualAge for Java 문서를 참조하십시오.

WebSphere Commerce 코드 저장소

WebSphere Commerce 응용프로그램의 사용자 정의 코드를 작성하려면, VisualAge for Java 작업 영역으로 WebSphere Commerce 코드 저장소를 반입해야 합니다. 저장소는 WebSphere Commerce Business Edition 디스크 2 CD 및 WebSphere Commerce Professional Edition 디스크 2 CD에서 사용 가능합니다. 현재의 저장소(공개시) 이름은 `WC_54.dat`입니다.

코드 전개

전자 상거래 응용프로그램 사용자 정의시, 다음 중 하나를 수행할 수 있습니다.

- 새 명령, 데이터 bean 또는 엔티티 bean 작성
- 기존의 WebSphere Commerce 엔티티 bean 확장
- 기존의 명령 또는 데이터 bean 로직 수정

VisualAge for Java 내에서 코드를 개발할 때는 WebSphere Test Environment 내에서 코드를 테스트할 수 있습니다. 어떤 시점이 되면 개발 환경 외부의 WebSphere Commerce Server로 코드를 전개해야 합니다.

다음 절에서 *대상 WebSphere Commerce Server*는 사용자 정의 코드를 전개할 WebSphere Commerce Server를 나타냅니다. 일부 테스트 시나리오에 VisualAge for Java와 동일한 시스템에서 실행되는 WebSphere Commerce Server를 전개할 수 있습니다. 다른 상황에서 대상 WebSphere Commerce Server는 다른 시스템에 있으며 다른 플랫폼에서 실행되고 있을 수도 있습니다.

다음 절에서는 여러 가지 유형의 사용자 정의 코드를 전개하기 위한 고급 단계에 대해 설명합니다. 개발 처리와 관련된 단계를 이해하려면 해당 절을 참조하고, 단계별 지시사항은 383 페이지의 부록 B『전개 정보』를 참조하십시오.

사용자 정의한 코드의 전개를 설명하는 다음 절의 경우 뿐만 아니라 전개 환경에 새 액세스 제어 정책을 작성한 경우에도 동일한 액세스 제어 정책을 대상 WebSphere Commerce Server에 작성해야 합니다. 이 프로시저 예에 대해서는 학습 307 페이지의『새 자원에 대한 액세스 제어 정책 로드』를 참조하십시오.

EJB 전개 코드에 대한 정보

VisualAge for Java V4.0의 WebSphere Test Environment 구성요소가 WebSphere Application Server V3.5.4를 사용한다는 사실을 잘 알아야 합니다. 이 WebSphere Application Server 버전에서 엔터프라이즈 bean은 EJB V1.0 스펙 레벨에서 지원합니다. 따라서 WebSphere Test Environment에서 엔터프라이즈 bean을 실행하기 위해서는 VisualAge for Java 도구를 사용하여 EJB V1.0 스펙을 따르는 엔터프라이즈 bean의 전개 코드를 생성해야 합니다.

이와 대조적으로 WebSphere Commerce Business Edition 및 WebSphere Commerce Professional Edition은 WebSphere Application Server V4.0을 사용합니다. WebSphere Application Server V4.0은 EJB V1.1 스펙을 지원합니다. 따라서 WebSphere Test Environment에서 실행하는 엔터프라이즈 bean의 전개 코드는 WebSphere Application Server V4.0에서 실행하는 엔터프라이즈 bean의 전개 코드와 다릅니다.

개발 및 전개에 대한 영향은 다음과 같습니다.

- WebSphere Test Environment에서 엔터프라이즈 bean을 실행하려면 VisualAge for Java 도구를 사용하여 WebSphere Test Environment의 WebSphere Application Server V3.5.4 요구사항에 알맞는 조건 전개 코드를 생성하십시오. 엔터프라이즈 bean을 마우스 오른쪽 버튼을 누르고 **전개된 코드 생성**을 선택하여 이 코드를 생성합니다. 이로 인해 JAR 파일이 작성되지는 않음에 유의하십시오.
- WebSphere Application Server V4.0으로 엔터프라이즈 bean을 전개하려면 다음을 수행해야 합니다.
  1. VisualAge for Java 도구를 사용하여 엔터프라이즈 bean을 해당 문서에서 *EJB 1.1 Export JAR 파일*이라 언급되는 파일로 반출하십시오. JAR 파일은 코드를 EJBDeploy 도구에서 사용되는 포맷으로 패키지합니다. 이 파일은 전개될 엔터프라이즈 bean을 포함하는 EJB 그룹을 마우스 오른쪽 버튼으로 누르고 **반출 > EJB 1.1 JAR**를 선택하여 작성됩니다. 이 JAR 파일 작성시, 코드를 전개할 시스템의 적절한 데이터베이스 유형을 선택해야 함에 유의하십시오.
  2. WebSphere Application Server V4.0을 실행하는 대상 서버로 EJB 1.1 Export JAR 파일을 전송하십시오.
  3. 대상 서버에서, 전개 코드의 새 JAR 파일을 작성하기 위한 입력으로 EJB 1.1 Export JAR 파일이 있는 EJBDeploy 도구를 실행하십시오.

엔터프라이즈 bean 전개에 관련된 다른 단계가 있습니다. 추가 정보는 216 페이지의 『새 엔티티 bean 전개』 및 220 페이지의 『수정 WebSphere Commerce 공용 엔티티 bean 전개』를 참조하십시오. EJBDeploy 도구 사용에 대한 추가 정보는 396 페이지의 『전개 코드 생성』을 참조하십시오.

새 명령 및 데이터 bean 전개

새 명령 및 데이터 bean 작성시, 응용프로그램에 맞게 이름이 지정된 패키지에 이들을 전개해야 합니다. 예를 들어 새 명령을 보관하는 com.mycompany.mycommands라는 이름으로 새 패키지를 작성할 수 있습니다. 이 패키지는 WebSphere Commerce 프로젝트(IBM WC Commerce Server 및 IBM WC Enterprise Bean)와 다른 프로젝트에 저장해야 합니다. 예를 들어 My Project라는 새 프로젝트를 작성할 수 있습니다. 해당 전개 기능이 예상한 대로 구현되도록 하려면 신중한 코드 구성이 필요합니다.

고유의 프로젝트에 모든 새 명령 및 데이터 bean이 저장되면 전개는 다음 고급 단계로 구성됩니다.

1. 개발 시스템을 사용하여 프로젝트용 JAR 파일을 작성하십시오. VisualAge for Java의 프로젝트 페이지에서 JAR 파일에 프로젝트를 반출할 도구를 사용하십시오. 예를 들어, CustomCommands.jar와 같이 코드에 적합한 JAR 파일 이름을 지정하십시오. 또한 VisualAge for Java 외부에서 도구를 사용하여 JAR 파일을 다시 jar 파일로 생성하여 필요한 이름 지정 정보 모두가 WebSphere Application Server로의 전개를 위해 포함되었는지 확인해야 합니다. 자세한 내용은 383 페이지의 『사용자 정의 명령 및 데이터 bean용 JAR 파일』을 참조하십시오.
2. JAR 파일, JSP 템플리트, 기타 상점 자원을 대상 WebSphere Commerce Server의 적절한 디렉토리에 복사하십시오. 자세한 내용은 391 페이지의 『대상 WebSphere Commerce Server에 자원 저장』을 참조하십시오.
3. 대상 WebSphere Commerce Server의 명령 레지스트리에 새 명령을 등록하십시오. 자세한 내용은 31 페이지의 『명령 등록 프레임워크』를 참조하십시오.
4. WebSphere Application Server 관리 콘솔을 사용하여 WebSphere Commerce 엔터프라이즈 응용프로그램을 중지하고 다시 시작하십시오. 이 응용프로그램 시작 및 중지에 대한 추가 정보는 적절한 *WebSphere Commerce 설치 안내서*를 참조하십시오.

새 엔티티 bean 전개

새 엔티티 bean 작성시, WebSphere Commerce 엔티티 bean을 포함하는 EJB 그룹과 다른 EJB 그룹에 작성해야 합니다. 또한 사용자 고유 프로젝트를 사용해야

하며 이 프로젝트에 명령 또는 데이터 bean 코드가 포함되어서는 안됩니다. 예를 들어, MyEntityBeans라는 새 EJB 그룹과 MyEntityBeansProject라는 이름으로 프로젝트를 작성할 수 있습니다. 프로젝트에 새 엔티티 beans의 코드가 아닌 코드가 들어 있으면 전개되지 않을 수 있습니다.

WebSphere Test Environment 내에서 엔티티 bean이 기능하는 방식에 만족하면 엔티티 bean을 전개해야 합니다. 다음 정보는 전개 단계에 대한 개요입니다.

1. 개발 시스템을 사용하여 새 EJB 그룹에 새 EJB 1.1 Export JAR 파일을 작성하십시오. VisualAge for Java의 EJB 페이지에서 새 EJB 그룹을 마우스 오른쪽 버튼으로 누른 후 EJB 1.1 JAR 파일로 코드를 반출하도록 선택하십시오. 예를 들어, MyEntityBeans_DT.jar와 같이 코드에 적합한 파일 이름을 지정하십시오. 자세한 내용은 385 페이지의 『새 엔티티 bean용 JAR 파일 작성』을 참조하십시오.
2. 새 EJB 프로젝트용 새 구현 JAR 파일을 작성하십시오. 이러한 JAR 파일을 작성하려면 프로젝트 페이지를 선택하고 새 EJB 프로젝트를 마우스 오른쪽 버튼으로 누른 후 JAR 파일로 코드 반출을 선택하십시오. 예를 들어, MyEntityBeansImpl.jar와 같이 코드에 맞는 파일 이름을 지정하십시오. 또한 VisualAge for Java 외부에서 도구를 사용하여 구현 JAR 파일을 다시 jar 파일로 생성하여 필요한 이름 지정 정보 모두가 WebSphere Application Server로의 전개를 위해 포함되었는지 확인해야 합니다. 자세한 내용은 385 페이지의 『새 엔티티 bean용 JAR 파일 작성』을 참조하십시오.
3. JAR 파일을 대상 WebSphere Commerce Server의 해당 디렉토리로 복사하십시오. 자세한 내용은 391 페이지의 『대상 WebSphere Commerce Server에 자원 저장』을 참조하십시오.
4. 대상 WebSphere Commerce Server에서, WebSphere Application Server가 제공하는 EJB 전개 도구를 사용하여 새 엔터프라이즈 bean의 전개 코드를 생성하십시오. 이 도구는 1단계에서 작성한 JAR 파일을 입력으로 사용하여 EJB 그룹의 모든 엔터프라이즈 bean의 전개 코드를 포함하고 있는 해당 JAR 파일을 작성합니다. 자세한 내용은 396 페이지의 『전개 코드 생성』을 참조하십시오.
5. 대상 WebSphere Commerce Server에서, 전개 코드의 JAR 파일에 포함되어 있는 엔터프라이즈 bean의 트랜잭션 분리 레벨을 수정하십시오. WebSphere

Commerce가 제공하는 modifyIsolationLevel 명령행 유틸리티를 사용하여 트랜잭션 분리 레벨을 적절한 데이터베이스 유형 레벨로 설정하십시오. 자세한 내용은 400 페이지의 『엔티티 bean의 트랜잭션 분리 레벨 수정』을 참조하십시오.

6. 대상 WebSphere Commerce Server에서 사용자가 작성한 새 자원의 액세스 제어 정책을 로드하십시오. WebSphere Commerce acpload 및 acpnlsload 명령을 사용하여 정책 정보를 로드하십시오. 새 자원의 액세스 제어 정책 로드 예는 제 9 장 『학습: 새 비즈니스 로직 작성』 학습의 307 페이지의 『새 자원에 대한 액세스 제어 정책 로드』 절을 참조하십시오.
7. 대상 WebSphere Commerce Server에서, 현재 WebSphere Commerce 엔터프라이즈 응용프로그램을 WebSphere Application Server에서 반출하십시오. 반출을 완료하면 전체 응용프로그램을 포함하는 .ear 파일이 작성됩니다. 현재 응용프로그램을 반출하려면 WebSphere Application Server 관리 콘솔을 열고 WebSphere Commerce 엔터프라이즈 응용프로그램을 선택한 다음 반출 옵션을 선택하십시오. 완료되면 WC_Enterprise_App_*instanceName*.ear 파일이 작성됩니다. 자세한 내용은 401 페이지의 『현재 WebSphere Commerce 엔터프라이즈 응용프로그램 반출』을 참조하십시오.
8. 대상 WebSphere Commerce Server에서 WebSphere Application Server에서 실행 중인 현재 WebSphere Commerce 엔터프라이즈 응용프로그램에 포함되는 엔터프라이즈 bean의 구성 정보를 반출하십시오. 이 정보는 WebSphere Application Server가 제공하는 XMLConfig 명령행 유틸리티의 -export 옵션을 사용하여 XML 파일에 반출됩니다. 생성된 XML 파일(OutputFile.xml 파일에 대해서는 여기를 참조)은 엔터프라이즈 응용프로그램에 포함되어 있는 각 엔터프라이즈 bean에 대한 구성 정보 스탠자를 포함합니다. 그리고 난 후 전개 중인 각 새 엔터프라이즈 bean에 대해 이 파일에 새 스탠자를 추가해야 합니다. 자세한 내용은 403 페이지의 『엔터프라이즈 bean의 구성 정보 반출』을 참조하십시오.
9. WebSphere Application Server가 제공하는 응용프로그램 어셈블리 도구를 사용하여 엔터프라이즈 응용프로그램에 새 엔터프라이즈 bean을 추가하십시오. 이 도구를 사용하여 현재 응용프로그램의 WC_Enterprise_App_*instanceName*.ear 를 연 후 응용프로그램으로 새 엔터프라이즈 bean을 반입할 수 있습니다. 또한 종속 JAR 파일을 포함하도록 새 엔터프라이즈 bean의 클래스 경로를 설정

하고 파일로서의 구현 JAR 파일을 응용프로그램에 추가합니다. 마지막으로, 이 도구를 사용하여 엔터프라이즈 bean에 포함되는 메소드의 WebSphere Application Server 보안을 구성합니다. 이 단계를 완료한 후 응용프로그램을 저장하면 새 `.ear` 파일이 엔터프라이즈 응용프로그램에 작성됩니다. 자세한 내용은 410 페이지의 『엔터프라이즈 응용프로그램으로 새 엔터프라이즈 bean 어셈블링』을 참조하십시오.

10. WebSphere Application Server에 새 엔터프라이즈 응용프로그램을 반입하십시오. 이 단계는 다음 세 가지 서브태스크로 구성됩니다.
    a. WebSphere Application Server 관리 콘솔을 사용하여 원래 WebSphere Commerce 엔터프라이즈 응용프로그램 중지 및 제거. 자세한 내용은 421 페이지의 『엔터프라이즈 응용프로그램 중지 및 제거』를 참조하십시오.
    b. `XMLConfig` 명령행 유틸리티의 `-import` 옵션을 사용하여 WebSphere Application Server로 새 엔터프라이즈 응용프로그램 반입. 자세한 내용은 422 페이지의 『엔터프라이즈 응용프로그램 반입』을 참조하십시오.
    c. WebSphere Application Server 관리 콘솔을 사용하여 새 엔터프라이즈 응용프로그램을 표시하도록 보기를 최신 정보로 고친 후 새 응용프로그램 시작. 자세한 내용은 424 페이지의 『엔터프라이즈 응용프로그램 시작』을 참조하십시오.

기존 명령 및 데이터 bean으로 확장 전개

기존 명령을 확장하는 방법은 필요한 수정 유형에 따라 다릅니다. 확장 방법은 161 페이지의 『기존 명령 사용자 정의』에 설명되어 있습니다. 일반적으로 기존 로직을 수정할 때는 사용자 정의가 필요한 클래스에서 상속되는 새 클래스를 작성하게 됩니다. 필요한 만큼 최상위 클래스의 메소드를 대체하여 로직을 바꾸거나 수정하십시오.

데이터 bean 사용자 정의시, 기존의 데이터 bean을 확장하는 새 클래스를 작성하기도 합니다. 새 클래스 내에서 필요한 만큼 수정하십시오.

새 클래스를 작성할 경우, 고유의 패키지 중 하나에 이 클래스들이 저장되어 있어야 하며 이 패키지는 고유의 프로젝트 중 하나에 저장됩니다.

서브클래스 상속에 의해 확장이 효과적으로 처리되므로 명령 및 데이터 bean에 대해 확장 전개는 새 명령 및 데이터 bean의 전개와 동일합니다. 자세한 내용은 216 페이지의 『새 명령 및 데이터 bean 전개』를 참조하십시오.

수정 WebSphere Commerce 공용 엔티티 bean 전개

WebSphere Commerce 공용 엔터프라이즈 bean을 수정할 때는 WebSphere Commerce 코드를 수정해야 합니다. 따라서 사용자 정의 엔티티 bean의 전개 방법은 새 엔티티 bean에 사용되는 것과 약간 다릅니다. 다음은 전개 단계의 개요입니다.

1. 수정 엔티티 bean을 포함하는 WebSphere Commerce EJB 그룹에 대한 EJB 1.1 Export JAR 파일을 작성하십시오. VisualAge for Java 워크벤치의 EJB 탭을 선택하여, 해당 EJB 그룹을 선택한 후 해당 그룹을 EJB 1.1 Export JAR 파일로 선택하여 반출하십시오. JAR 파일에 이름을 지정할 때, `Cust_`*`EJBGroupName`*`-ejb_DT.jar` 이름 지정 규칙을 사용할 수 있습니다. 여기서 *`EJBGroupName`*은 수정된 EJB 그룹의 이름이며 _DT 접미부가 JAR 파일 이름 끝에 추가됩니다. 예를 들어 WCSUser EJB 그룹에 대한 JAR 파일 이름 지정할 경우, 이 파일 이름을 `Cust_WCSUser-ejb_DT.jar`로 지정할 수 있습니다. _DT 접미부는 메모로서 단순히 사용하는 것이기 때문에 이후에 JAR 파일을 EJBDeploy 도구로 전달하여 EJB 그룹에 포함되어 있는 bean의 전개 도구를 생성해야 한다는 점에 유의하십시오. 자세한 내용은 388 페이지의 『사용자 정의 WebSphere Commerce 엔티티 bean용 JAR 파일 작성』을 참조하십시오.
2. 모든 WebSphere Commerce EJB 그룹에 대한 클라이언트 코드를 포함하는 새 클라이언트 JAR 파일을 작성하십시오. 선택된 모든 WebSphere Commerce EJB 그룹으로(모든 이름은 `WCS`로 시작) 클라이언트 JAR 파일에 반출하십시오. 자세한 내용은 388 페이지의 『사용자 정의 WebSphere Commerce 엔티티 bean용 JAR 파일 작성』을 참조하십시오.
3. JAR 파일을 대상 WebSphere Commerce Server의 해당 디렉토리로 복사하십시오. 자세한 내용은 391 페이지의 『대상 WebSphere Commerce Server에 자원 저장』을 참조하십시오.
4. 대상 WebSphere Commerce Server에서 WebSphere Application Server가 제공하는 EJBDeploy 도구를 사용하여 사용자가 전개 중인 EJB 그룹에 포함

되어 있는 엔터프라이즈 bean의 전개 코드를 생성하십시오. 이 도구는 1단계에서 작성한 JAR 파일을 입력으로 사용하여 EJB 그룹의 모든 엔터프라이즈 bean의 전개 코드를 포함하고 있는 해당 JAR 파일을 작성합니다. 자세한 내용은 396 페이지의 『전개 코드 생성』을 참조하십시오.

5. 대상 WebSphere Commerce Server에서 전개 코드의 JAR 파일에 포함되어 있는 엔터프라이즈 bean의 트랜잭션 분리 레벨을 수정하십시오. WebSphere Commerce가 제공하는 `modifyIsolationLevel` 명령행 유틸리티를 사용하여 트랜잭션 분리 레벨을 적절한 데이터베이스 유형 레벨로 설정하십시오. 자세한 내용은 400 페이지의 『엔티티 bean의 트랜잭션 분리 레벨 수정』을 참조하십시오.
6. 대상 WebSphere Commerce Server에서 현재 WebSphere Commerce 엔터프라이즈 응용프로그램을 WebSphere Application Server에서 반출하십시오. 반출을 완료하면 전체 응용프로그램을 포함하는 .ear 파일이 작성됩니다. 현재 응용프로그램을 반출하려면 WebSphere Application Server 관리 콘솔을 열고 WebSphere Commerce 엔터프라이즈 응용프로그램을 선택한 다음 반출 옵션을 선택하십시오. 완료되면 `WC_Enterprise_App_`*`instanceName`*`.ear` 파일이 작성됩니다. 자세한 내용은 401 페이지의 『현재 WebSphere Commerce 엔터프라이즈 응용프로그램 반출』을 참조하십시오.
7. 대상 WebSphere Commerce Server에서 WebSphere Application Server에서 실행 중인 현재 WebSphere Commerce 엔터프라이즈 응용프로그램에 포함되는 엔터프라이즈 bean의 구성 정보를 반출하십시오. 이 정보는 WebSphere Application Server가 제공하는 `XMLConfig` 명령행 유틸리티의 `-export` 옵션을 사용하여 XML 파일에 반출됩니다. 생성된 XML 파일(`OutputFile.xml` 파일에 대해서는 여기를 참조)은 엔터프라이즈 응용프로그램에 포함되어 있는 각 엔터프라이즈 bean에 대한 구성 정보 스탠자를 포함합니다. 자세한 내용은 403 페이지의 『엔터프라이즈 bean의 구성 정보 반출』을 참조하십시오. 이 파일 내에서 새 `Cust_`*`EJBGroupName`*`-ejb.jar` 파일을 수정하도록 지시한 엔터프라이즈 bean에 대해 설명하는 스탠자를 수정해야 합니다.
8. 대상 WebSphere Commerce Server에서 응용프로그램 어셈블리 도구를 사용하여 엔터프라이즈 응용프로그램으로 수정된 EJB 그룹을 통합하십시오. 이 도구를 사용하여 현재 응용프로그램에 대한 `.ear` 파일을 열 수 있습니다. 파일이 열리면 다음 태스크를 수행하십시오.

a. 수정한 EJB 그룹의 원래 버전에 대한 클래스 경로 정보를 기록하십시오. 예를 들어 WCSUser EJB 그룹의 사용자 bean을 수정한 경우, WCSUser 그룹의 클래스 경로 정보를 텍스트 파일로 복사하십시오.

b. 수정한 EJB 그룹의 원래 버전을 삭제하십시오(예: WCSUser 그룹 삭제).

c. 수정한 EJB 그룹의 새 버전을 반입하십시오.

d. 방금 반입한 EJB 그룹에 원래 클래스 경로 정보를 추가하십시오.

e. 수정 EJB 그룹의 엔터프라이즈 bean에 포함되어 있는 메소드에 WebSphere Application Server 보안을 구성하십시오.

f. 응용프로그램을 새 `.ear`로 저장하십시오.

자세한 내용은 416 페이지의 『엔터프라이즈 응용프로그램으로 수정 엔터프라이즈 bean 어셈블링』을 참조하십시오.

9. WebSphere Application Server에 새 엔터프라이즈 응용프로그램을 반입하십시오. 이 단계는 다음 세 가지 서브태스크로 구성됩니다.

   a. WebSphere Application Server 관리 콘솔을 사용하여 원래 WebSphere Commerce 엔터프라이즈 응용프로그램 중지 및 제거. 자세한 내용은 421 페이지의 『엔터프라이즈 응용프로그램 중지 및 제거』를 참조하십시오.

   b. `XMLConfig` 명령행 유틸리티의 `-import` 옵션을 사용하여 WebSphere Application Server로 새 엔터프라이즈 응용프로그램 반입. 자세한 내용은 422 페이지의 『엔터프라이즈 응용프로그램 반입』을 참조하십시오.

   c. WebSphere Application Server 관리 콘솔을 사용하여 새 엔터프라이즈 응용프로그램을 표시하도록 보기를 최신 정보로 고친 후 새 응용프로그램 시작. 자세한 내용은 424 페이지의 『엔터프라이즈 응용프로그램 시작』을 참조하십시오.

Commerce Studio에 사용할 새 데이터 bean 전개

JSP 템플리트 개발에 Commerce Studio를 사용할 경우, Commerce Studio로 새 데이터 bean을 전개해야 합니다. 특히 데이터 bean의 JAR 파일을 작성하십시오.

데이터 bean 패키지를 마우스 오른쪽 버튼으로 누른 후 이를 JAR 파일로 반출하도록 선택하여 JAR 파일을 작성하십시오. 옵션에서 다음을 포함하도록 선택하십시오.

- 클래스
- 자원
- **bean**

또한 **참조된 유형 및 자원 선택**을 선택하여 데이터 bean에 필요한 명령 및 자원도 포함시키십시오.

JAR 파일을 반출하면 이 JAR 파일을 포함하도록 Commerce Studio의 클래스 경로를 갱신해야 합니다.

기존의 WebSphere Commerce 데이터 bean을 수정해야 할 경우, 사용자 정의가 필요한 데이터 bean을 확장하는 새 데이터 bean을 작성해야 합니다. 따라서 실제로 새 데이터 bean을 작성하게 됩니다. 전개는 이 절에서 설명된 것과 같습니다.

Commerce Studio에서 사용할 사용자 정의 공용 엔티티 bean 전개

공용 엔티티 bean을 수정하고 JSP 템플리트 개발에 Commerce Studio를 사용하면, 모든 공용 엔티티 bean용 새 클라이언트 JAR 파일을 작성하고 Commerce Studio의 클래스 경로를 수정하여 원래 JAR 파일 이름 앞에 새 JAR 파일 이름을 넣어야 합니다. Commerce Studio에서는 데이터 bean이 Page Designer 도구에 사용될 때 클라이언트 JAR 파일을 사용합니다.

이 JAR 파일을 작성하려면 VisualAge for Java의 도구를 사용하십시오. VisualAge for Java의 EJB 페이지에서 모든 WebSphere Commerce EJB 그룹(수정한 것만이 아니라)을 선택하고 클라이언트 JAR 파일에 이들을 반출하도록 선택하십시오. 반출이 완료된 후에는 Commerce Studio의 클래스 경로 시작 부분에 새 JAR 파일을 지정해야 합니다.

로그 파일

상품 설치 단계 전반에 걸쳐 코드 개발 및 코드 전개 로그 파일이 생성됩니다. 이 절에서는 이 단계에서 전반적으로 접할 수 있는 일부 로그 파일들을 나열합니다.

Commerce Studio 로그 파일

Commerce Studio는 설치 처리 동안 로그 파일을 빌드합니다. 이 로그에 액세스하려면 다음 파일을 여십시오.

```
C:\Winnt\WCStudioInstall.log
```

WebSphere Test Environment용 Commerce Studio 구성에 대한 로그

백엔드 개발 태스크를 수행할 것을 선택할 경우, WebSphere Test Environment에 대한 여러 구성 단계가 WebSphere Commerce Studio 설치 동안 수행됩니다. 예를 들어 VisualAge for Java의 WebSphere Test Environment 구성요소 내에 실행되는 견본 상점을 포함하도록 선택하면 대량 로드 처리 및 데이터베이스 작성에 관련되는 로그 파일이 작성됩니다. 이러한 로그에 액세스하려면 다음 디렉토리로 이동하십시오.

drive:\WebSphere\CommerceServerDev\instances*instance_name*\logs

WebSphere Test Environment 내에서 실행 중인 WebSphere Commerce 구성요소의 로그 파일

상점을 실행하거나 WebSphere Test Environment 내에서 개별 구성요소를 테스트할 경우, 추적 파일과 메시지 파일이 생성될 수 있습니다. 이러한 파일의 기본 위치는 다음 디렉토리입니다.

drive:\WebSphere\CommerceServerDev\instances*instance_name*\logs

실행 중인 modifyIsolationLevel 명령의 로그

엔터프라이즈 bean 전개시, modifyIsolationLevel 명령을 사용하여 JAR 파일의 각 엔터프라이즈 bean에 대한 트래잭션 분리 레벨을 수정하십시오. 생성되는 로그 파일은 명령을 실행할 때 지정하는 로그 파일에 저장됩니다. 자세한 내용은 400 페이지의 『엔티티 bean의 트랜잭션 분리 레벨 수정』을 참조하십시오.

VisualAge for Java 내에서 로그 작성

WebSphere Test Environment 내에서 코드를 실행할 경우, 콘솔 창은 활성 로그처럼 실행됩니다. 또한 EJB 서버에 대한 추적 레벨을 수정하여 콘솔 창에서 볼 수 있는 정보의 양을 늘릴 수 있습니다. 이러한 정보는 EJB 서버의 특성 내에서 추적 레벨로 지정됩니다.

테스트 지불 방법

WebSphere Test Environment 내에서 실행되는 InFashion 견본 상점은 기본적으로 테스트 지불 방법을 사용합니다. 이 테스트 지불 방법은 원격 Payment Manager를 호출하지 않고 WebSphere Test Environment 내에서 쇼핑 플로우를 완료할 수 있도록 하기 위해 포함된 것입니다. 이 지불 방법을 사용하여 제출된 주문의 상태는 'M'(지불이 시작되어 처리를 기다리고 있는 상태)입니다.

이 테스트 지불 방법은 구매를 완료하게만 하며 이 지불 방법과 함께 제출된 주문을 더 이상 처리할 수 있게 하지는 않습니다. 이처럼 테스트 지불 방법은 WebSphere Test Environment 내에서만 사용되어야 합니다.

지불 관련 명령에 대한 구현 클래스는 비즈니스 정책 명령의 유형입니다. 이와 같은 구현 클래스를 선택하는 것은 비즈니스 정책에서 다루어집니다. WebSphere Test Environment를 사용하여 실행하는 InFashion 견본 상점에는 기본값으로 다음과 같은 지불 관련 명령 구현을 사용하는 비즈니스 정책(TestPaymentMethod 정책)이 포함되어 있습니다.

- 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

이러한 명령은 반드시 테스트 및 주문 시작 처리에 대해서만 제공해야 합니다. 앞에 제공된 일련의 명령은 보다 자세히 설명하기 위해 제공된 것이며 DoDepositTestCmdImpl은 호출할 경우 예외가 발생하는 명령입니다. 주문에서는 주문 관리 함수를 사용하지 마십시오. 이러한 기타 함수들을 테스트할 수 있는 기능이 필요하면 WebSphere Test Environment에서 원격 Payment Manager를 사용하도록 구성하면 됩니다.

대상 WebSphere Commerce Server로 사용자 코드를 전개할 때, 사용자 개발에서 대상 시스템으로 테스트 지불 방법에 관련된 비즈니스 정책을 복사하지 말아야 합니다. 또한 대상 WebSphere Commerce Server에서 테스트 지불 방법에 관련된 명령을 등록해서는 안됩니다.

테스트 지불 방법을 사용하지 않게 하는 방법에 대해서는 *WebSphere Commerce Business Edition 설치 안내서* 또는 *WebSphere Commerce Professional Edition 설치 안내서*의 "VisualAge for Java 구성" 장을 참조하십시오.

원격 Payment Manager 사용

사용자의 개발 작업에 제출 후의 주문에 대한 작업(예: 주문 관리 처리에서의 단계 수정판)이 포함되어 있으면 WebSphere Test Environment에서 실행되는 상점이 원격 Payment Manager를 사용하도록 수정해야 합니다. 자세한 구성 단계는 *WebSphere Commerce Studio, Business Developer Edition 설치 안내서*를 참조하십시오. 이 책에서는 데이터베이스에서 테스트 지불 방법으로 제출된 주문을 제거하는 방법에 대한 정보도 제공합니다.

제 4 부 학습

여기서는 e-commerce 응용프로그램 사용자 정의에 익숙해지도록 도움을 주는 학습 내용을 제공합니다. 다음 학습 내용이 제공됩니다.

- 새 비즈니스 로직 작성
  이 학습은 새 비즈니스 로직 작성을 위한 개발 처리를 보여줍니다. 다음과 같은 서브 태스크가 포함됩니다.
  - 견본 프로젝트 준비
  - 새 명령 작성
  - 새 데이터 bean을 작성하고 요청으로부터 특성 가져오기
  - 엔티티 bean 사용
  - 새 엔터프라이즈 bean 작성
- 기본 비즈니스 로직 갱신
  여기서는 기존 WebSphere Commerce 비즈니스 로직을 갱신하기 위한 개발 처리를 보여줍니다. 다음과 같은 서브 태스크가 포함됩니다.
  - 기존 WebSphere Commerce 제어기 명령 확장
  - 기존 WebSphere Commerce 공용 엔티티 bean 수정
  - 기존 WebSphere Commerce 태스크 명령을 확장하는 새 태스크 명령 작성

학습은 VisualAge for Java에 입력해야 하는 코드 섹션을 포함합니다. 이 문서의 PDF 버전은 다음 웹 사이트에서 구할 수 있습니다:

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
```

프로그래머 안내서의 PDF 버전에서 이 코드를 잘라내어 붙여넣을 수 있습니다.

제 9 장 학습: 새 비즈니스 로직 작성

환경 학습

이 책에 수록된 학습을 수행하려면 WebSphere Commerce Business Edition V5.4 또는 WebSphere Commerce Professional Edition V5.4가 설치되어 있어야 합니다. 또한 제품을 설치할 때 다음 옵션을 설치할 것을 선택해야 합니다.

- **WebSphere Studio를 사용하여 상점 입구 자원 개발**
- **VisualAge for Java를 사용하여 상점 백엔드 로직 개발**
- **데이터베이스 작성**
- **견본 상점 포함**

전체 설치 및 구성 정보에 대해서는 *WebSphere Commerce Studio, Business Developer Edition 설치 안내서* 또는 *WebSphere Commerce Studio Professional Developer Edition 설치 안내서*의 지시사항을 따르십시오.

학습을 시작하기 전에 WebSphere Test Environment 내에서 견본 상점을 실행하고 상점에서 구매를 완료할 수 있어야 합니다.

학습 코드 전개 단계

이 책에 수록된 학습에는 대상 WebSphere Commerce Server에 사용자 정의 코드를 전개하는 방법을 설명하는 선택적 단계가 포함됩니다. 대상 WebSphere Commerce Server는 개발 환경과 다른 시스템의 Windows NT 또는 Windows 2000에서 실행 중인 것으로 가정합니다. WebSphere Commerce Studio Business Developer Edition을 개발 환경으로 사용하는 경우 WebSphere Commerce Business Edition으로 전개해야 함에 유의하십시오. WebSphere Commerce Studio Professional Developer Edition을 개발 환경으로 사용하는 경우 WebSphere Commerce Professional Edition으로 전개해야 합니다.

또한 대상 WebSphere Commerce Server에서 InFashion 견본 상점에 근거한 상점을 공개해야 합니다. 해당 상점 내에서 구매를 완료할 수 있어야 합니다. 지불 처리를 위해 원격 또는 로컬 Payment Manager 중 하나를 사용할 수 있습니다.

개발 환경과 동일한 시스템에 있거나 다른 운영체제에서 실행되는 대상 WebSphere Commerce Server에 전개하는 경우, 적절한 전개 정보는 211 페이지의 제 8 장 『개발 도구 및 전개』 및 383 페이지의 부록 B 『전개 정보』를 참조하십시오.

견본 프로젝트 준비

이 절에서는 "새 비즈니스 로직 작성" 학습의 시작점이 있는 WC_SAMPLE_54.dat 저장소를 반입합니다.

환경을 준비하려면 다음을 수행하십시오.

1. WebSphere Commerce 코드의 WC_54.dat 저장소를 사용하고 있는지 확인하십시오. 이는 WebSphere Commerce Business Edition V5.4 디스크 2 CD 또는 WebSphere Commerce Professional Edition V5.4 디스크 2 CD에서 사용 가능합니다.
2. 다음을 수행하여 견본 프로젝트를 작업 영역에 반입하십시오.
   a. 다음 CD를 개발 시스템의 CD 드라이브에 삽입하십시오.
      - Business WebSphere Commerce Business Edition, V5.4 디스크 2
      - Professional WebSphere Commerce Professional Edition, V5.4 디스크 2
   b. VisualAge for Java를 여십시오.
   c. **파일** 메뉴에서 **반입**을 선택하십시오.
      반입 SmartGuide가 열립니다.
   d. **저장소** 반입을 선택한 후 **다음**을 누르십시오.
   e. 다른 저장소 창의 반입에서 다음을 수행하십시오.
      1) **로컬 저장소**를 선택하십시오.
      2) **저장소 이름** 필드에 다음을 입력하십시오.
         CD_drive:\repository\samples\programguide\WC_SAMPLE_54.

dat

여기서, *CD_drive*는 CD 드라이브입니다.

3) **프로젝트**를 선택하고 **자세히 보기**를 누르십시오. **_WCSSamples** 프로젝트를 선택하십시오. **WC 견본 5.4** 버전을 선택하고 **확인**을 누르십시오.

4) **작업 영역에 가장 최근의 프로젝트 개정판 추가**가 선택되어 있어야 합니다.

5) **완료**를 눌러 반입을 시작하십시오.
프로젝트를 반입하는 데 몇 분 걸립니다.

3. 다음을 수행하여 작업 영역 소유자를 WCS 개발자로 설정했는지 확인하십시오.

   a. **작업 영역** 메뉴에서 **작업 영역 소유자 변경**을 선택하십시오.

   b. **WCS 개발자**를 선택한 후 **확인**을 누르십시오.

4. 연습용 JSP 템플리트를 해당 디렉토리로 복사하여 WebSphere Test Environment에서 사용할 수 있도록 하십시오. 이들 파일을 복사하려면 다음을 수행하십시오.

   a. 다음 디렉토리에 Sample.jsp 및 Sample_All.jsp 파일을 위치 지정하십시오.

      CD_drive:\repository\samples\programguide\

      *CD_drive*는 다음 디렉토리에 대한 CD 드라이브입니다.

   b. 이 두 파일을 다음 디렉토리로 복사하십시오.

      vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web

      여기서 *vaj_drive*는 VisualAge for Java을 설치한 드라이브입니다.

5. 액세스 제어 정책 파일을 해당 디렉토리에 복사하십시오. 이들 파일은 학습 과정에 작성하는 새 자원에 대한 새 액세스 제어 정책을 로드하는데 사용됩니다. 이들 파일을 복사하려면 다음을 수행하십시오.

   a. 다음 디렉토리로 전환하십시오.

      CD_drive:\repository\samples\programguide\

b. 해당 디렉토리에서 다음 파일을 위치 지정하십시오.

- SampleCmdACPolicy.xml
  이 XML 파일은 사용자가 새 제어기 명령을 작성할 때 사용되는 액세스 제어 정책을 포함합니다.
- SampleACPolicy.xml
  이 XML 파일은 새 엔터프라이즈 bean을 작성할 때 사용되는 액세스 제어 정책입니다.
- SampleACPolicy_*locale*.xml
  여기서 *locale*은 언어 식별자입니다. 이 XML 파일에는 액세스 제어 정책 설명이 들어 있습니다.

c. 앞의 파일을 다음 디렉토리에 복사하십시오.
drive:\WebSphere\CommerceServerDev\xml\policies\xml
여기서 *drive*는 WebSphere Commerce Studio를 설치한 드라이브입니다.

6. 다음을 수행하여 학습을 시작할 준비가 되었는지 확인하기 위해 사용자 환경을 테스트하십시오.

a. 379 페이지에 설명된 대로 PNS(Persistent Name Server)를 시작하십시오.

b. 380 페이지에 표시된 대로 EJB 서버를 시작하십시오.

c. 380 페이지에 설명된 대로 Servlet 엔진을 시작하십시오.

d. 브라우저를 열고 다음 URL을 입력하십시오.

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
   storeId=store_ID&catalogId=catalog_Id&langId=-1
```

여기서 *store_ID*는 견본 상점에 대한 식별자이고(10001은 예제 값) *catalog_Id*는 견본 상점의 카탈로그에 대한 식별자입니다(10001은 예제 값).
견본 상점의 홈페이지가 표시되면 제품을 선택하고 구입하십시오. 쇼핑 플로우에서 “주문 확인” 페이지에 도달할 수 있어야 합니다.

상점에 대한 상점 ID 값을 검증하기 위해 STOREENT 테이블을 참조할 수 있습니다.

e. 모든 브라우저를 종료하고 WebSphere Test Environment를 중지하십시오.

이제 학습을 시작할 준비가 되었습니다.

명령 작성

제어기 명령 작성

이 절에서는 제어기 명령을 작성하는 방법을 보여줍니다. 이 연습이 끝나면 새 명령 *MyNewControllerCmd*가 생성됩니다. 모든 상점에서 이 명령을 사용하게 되며 각 상점은 명령을 동일한 방식으로 구현합니다.

새 제어기 명령 작성시, 다음과 같은 세 가지 기본 단계가 있습니다.

1. 명령 레지스트리 프레임워크에 명령 등록
2. 명령의 인터페이스 작성
3. 명령의 구현 클래스 작성

명령에 대한 자세한 내용은 25 페이지의 『명령 설계 패턴』을 참조하십시오.

학습을 시작하기 전에

230 페이지의 『견본 프로젝트 준비』 단계를 완료해야 합니다.

새 명령을 작성하려면 다음을 수행하십시오.

1. 제어기 명령을 작성하는 첫 번째 단계는 명령의 이름을 설정하는 것입니다. 이 예에서 명령의 이름은 *MyNewControllerCmd*입니다.
2. 명령 레지스트리 프레임워크에 명령을 등록해야 합니다. 새 제어기 명령의 등록 처리를 수행하려면 URLREG 테이블의 항목을 작성해야 합니다.

   ▶ DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 명령을 등록하십시오.

   a. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.
   b. **도구** 메뉴에서 **도구 설정**을 선택하십시오.
   c. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(**;**)인지 확인하십시오.

d. 스크립트 탭을 선택하고, 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

```
connect to your_database_name;
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)
```

여기서 *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.

모든 판매자가 이 명령을 사용합니다(STOREENT_ID에 0 값으로 표시).

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 명령을 등록하십시오.

a. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).

b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.

c. **암호** 필드에 Oracle 암호를 입력하십시오.

d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.

e. SQL Plus 창에서 URLREG 테이블에 필수 항목을 작성하려면 다음을 입력하십시오.

```
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);
```

Enter를 눌러 SQL 문을 실행하십시오.

f. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL 문을 실행하십시오.

주: 이 학습을 단순화하기 위해 새 명령에는 하나의 구현 클래스만 있습니다. 즉, 모든 상점에 동일한 구현 클래스가 사용됩니다. 이 구현 클래스는 인터페이스 코드에 직접 지정됩니다. 이와 같이 CMDREG 테이블에서 인터

페이스와 구현 클래스 사이의 맵핑을 등록할 필요는 없습니다. 학습 환경 외부에서는 URLREG 테이블 뿐만 아니라 CMDREG 테이블의 제어기 명령도 등록해야 합니다.

3. 제어기 명령은 보기를 리턴해야 합니다. 작성할 새 제어기 명령은 SampleViewTask 보기를 리턴합니다. SampleViewTask 보기를 VIEWREG 테이블에 등록해야 합니다.

   DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 보기를 등록하십시오.

   a. 스크립트 창에 다음을 입력하여 VIEWREG 테이블에 항목을 작성하십시오(우선 창에서 이전 SQL 문을 지워야 합니다).

      ```
      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)
      ```

      실행 아이콘을 누르십시오.

   Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 사용자 보기를 데이터베이스에 등록하십시오.

   a. SQL Plus 창에서 다음 SQL 문을 입력하십시오.

      ```
      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);
      ```

      Enter를 눌러 SQL 문을 실행하십시오.

   b. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

      ```
      commit;
      ```

      Enter를 눌러 SQL 문을 실행하십시오.

4. VisualAge for Java 워크벤치 창에서 **_WCSSamples** 프로젝트를 펼치십시오.

5. **com.ibm.commerce.sample.commands** 패키지를 펼치고 **MyNewControllerCmd** 인터페이스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오.
   필드 작성 SmartGuide가 열립니다.
6. 다음을 수행하여 인터페이스의 기본 구현 클래스를 지정하는 필드를 작성하십시오.
   a. **필드 이름** 필드에 defaultCommandClassName을 입력하십시오.
   b. **필드 유형** 드롭 다운 목록에서 문자열을 선택하십시오.
   c. 초기값 필드에 "com.ibm.commerce.sample.commands.MyNewControllerCmdImpl"을 입력하십시오(큰따옴표를 입력해야 합니다).
   d. **완료**를 누르십시오.
      새 필드용 코드가 생성됩니다.

이 학습에서 최상위 클래스에 있는 필드 또는 메소드를 대체할 때, 새 필드 또는 메소드가 상속된 필드를 숨기거나 메소드가 표시될 수 있다는 경고가 표시됩니다. 이러한 경우 **예**를 눌러 계속하십시오.

7. **MyNewControllerCmdImpl** 클래스를 펼친 후 해당 **performExecute** 메소드를 선택하여 해당 소스 코드를 보십시오.
8. performExecute 메소드의 소스 코드에서 Section 1과 Section 5의 설명 표시를 제거하십시오. Section 1에서는 메소드에 다음 코드가 삽입됩니다.

```
// Create a new TypedProperties for output.
TypedProperty rspProp = new TypedProperty();
```

   Section 5에서는 메소드에 다음 코드가 삽입됩니다.

```
// see how controller command call a JSP

rspProp.put(ECConstants.EC_VIEWTASKNAME, "SampleViewTask");
setResponseProperties( rspProp );
```

   작업을 저장하십시오(**Ctrl + S**). 앞의 코드 부분은 제어기 명령에서 리턴된 보기 이름을 설정합니다.
9. 이 새 제어기 명령에 대해 명령 레벨 액세스 제어가 지정되어야 합니다. 이런 경우, 명령 레벨 액세스 제어 정책은 모든 사용자가 명령을 실행할 수 있도록

지정합니다. 이 정책은 SampleCmdACPolicy.xml 파일에 지정됩니다. 새 액세스 제어 정책을 로드하려면 다음을 수행하십시오.

a. 명령 프롬프트에서 다음 디렉토리로 전환하십시오.
   drive:\WebSphere\CommerceServerDev\bin

b. 다음 양식으로 acpload 명령을 발행해야 합니다.

   ```
   acpload db_name db_user db_password inputXMLFile
   ```

   여기서,

   - *db_name*은 데이터베이스의 이름입니다.
   - *db_user*는 데이터베이스 사용자 이름입니다.
   - *db_password*는 데이터베이스 암호입니다.
   - *inputXMLFile*은 정책을 포함하는 XML 파일의 이름입니다.

   예를 들어 다음과 같은 명령을 발행할 수 있습니다.

   ```
   acpload VAJ_Demo user password SampleCmdACPolicy.xml
   ```

10. 다음을 수행하여 Servlet 엔진에 대한 클래스 경로에 새 _WCSSamples 프로젝트를 추가하십시오.

    a. **작업 영역** 메뉴에서 **도구 > WebSphere Test Environment**를 선택하십시오.
       WebSphere Test Environment 제어 센터가 열립니다.

    b. **Servlet 엔진**을 누른 다음 **클래스 경로 편집**을 누르십시오.
       Servlet 엔진 클래스 경로 창에서 **모두 선택**을 누른 다음 **확인**을 누르십시오.

11. 다음을 수행하여 새 명령을 테스트하십시오.

    a. 379 페이지에 설명된 대로 PNS 시작하십시오.

    b. 380 페이지에 표시된 대로 EJB 서버를 시작하십시오.

    c. 380 페이지에 설명된 대로 Servlet 엔진을 시작하십시오.

    d. 브라우저를 열어 다음 URL을 입력하십시오.

       ```
       http://localhost:8080/webapp/wcs/stores/servlet/
          MyNewControllerCmd
       ```

몇 분 후, 브라우저가 아래와 같이 “견본 JSP”를 나타내는 페이지를 표시합니다.

그림 31.

12. 현재 상태의 코드 버전을 작성하십시오. 그러면 코드를 어떤 시점의 상태로도 복원할 수 있습니다. 버전을 작성하려면 다음을 수행하십시오.
    a. **com.ibm.commerce.sample.commands** 및 **com.ibm.commerce.sample.databeans** 패키지를 선택하십시오(`Ctrl` 키를 누른 상태에서 여러 패키지를 강조표시하여 선택하십시오). **관리 > 개방판 작성**을 마우스 오른쪽 버튼으로 누른 후 선택하십시오.
    b. **_WCSSamples** 프로젝트를 마우스 오른쪽 버튼으로 누른 후 **관리 > 개방판 작성**을 선택하십시오.
    c. **_WCSSamples** 프로젝트를 다시 마우스 오른쪽 버튼으로 누른 후 **관리 > 버전**을 선택하십시오.
    선택된 항목 버전화 창이 열립니다.
    d. **한 개의 이름** 라디오 버튼을 선택한 후 `mySample 1.2 Completed`를 입력하고 **확인**을 누르십시오.

이제 통합된 테스트 환경에서 새 명령을 추가하고 테스트하였습니다(간략하게).

MyNewControllerCmd 수정

이전 절에서는 MyNewControllerCmd를 작성했습니다. 이 연습에서는 사용자 정의된 고유 제어기 명령을 작성하는 방법을 더 명확히 이해하기 위해 새 명령의 컨텐츠를 보다 자세히 살펴보겠습니다.

작성한 코드의 구조를 살펴보는 작업부터 시작합니다. MyNewControllerCmd 인터페이스는 ControllerCommand 인터페이스를 확장합니다. 또한 기본값으로 사용할 구현 클래스를 정의합니다. 이 클래스는 명령이 CMDREG 테이블에 등록되어 있지 않거나 구현 클래스가 해당 테이블에 지정되어 있지 않을 때 사용됩니다.

또한 MyNewControllerCmdImpl 클래스를 작성하였습니다. 구현 클래스는 구현하려는 비즈니스 로직을 포함할(또는 태스크 명령을 호출하여 각 비즈니스 태스크를 수행할) 클래스입니다. 구현 클래스에는 다음 메소드가 들어 있습니다.

MyNewControllerCmdImpl의 메소드	**설명**
MyNewControllerCmdImpl()	생성자 메소드
validateParameters()	명령 입력 매개변수를 서버측에서 유효성 확인하는 데 사용됩니다.
isGeneric()	일반 사용자가 명령을 호출하는지 여부를 결정합니다.
isRetriable()	데이터베이스가 롤백된 후 명령을 재시도할 것인지 여부를 결정합니다.
performExecute()	명령에 대한 비즈니스 로직을 포함합니다.

다음 절에서는 제어기 명령을 갱신하는 방법에 대해 더 자세히 설명합니다.

JSP 템플리트로 변수 전달

이 절에서는 JSP 템플리트로 변수를 전달하도록 MyNewControllerCmd를 수정합니다. 변수를 표시하려면 DataBeanSampleBean이라는 새 데이터 bean을 사용해야 합니다. JSP 템플리트에서 변수를 표시하려면 다음을 수행하십시오.

1. VisualAge for Java 워크벤치 창에서 **_WCSSamples** 프로젝트를 펼치십시오.
2. **com.ibm.commerce.sample.commands** 패키지와 **MyNewControllerCmdImpl** 클래스를 펼친 후 해당 **performExecute** 메소드를 선택하십시오.
3. performExecute 메소드의 소스 코드에서 Section 2의 설명 표시를 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

```
// 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!");
```

앞의 코드 부분에서는 JSP 템플리트로 전달된 특성에 들어갈 두 개의 매개변수를 작성합니다. 작업을 저장하십시오(**Ctrl + S**).

4. DataBeanSampleBean 데이터 bean은 변수를 표시하기 위해 JSP 템플리트에서 사용됩니다. bean이 이미 작성되어 있으나 다음과 같이 소스 코드를 수정해야 합니다.
   a. **com.ibm.commerce.sample.databean** 패키지를 펼치십시오.
   b. **DataBeanSampleBean** 클래스를 펼친 후 **setRequestProperties** 메소드를 선택하여 해당 소스 코드를 보십시오.
   c. setRequestProperties 메소드의 소스 코드에서 Section 1의 설명 표시를 제거하십시오. 메소드에 다음 코드가 삽입됩니다.

      ```
      // copy input TypedProperties to local

      requestProperties = aParam;
      ```

      작업을 저장하십시오. 위의 코드 부분에서는 aParam(최상위 클래스에 정의)에서 값을 로컬로 복사합니다. 이 코드는 요청 오브젝트에서 특성을 가져오는 데 사용됩니다.
5. 다음을 수행하여 새 데이터 bean을 사용하고 변수를 표시하도록 Sample.jsp 파일을 갱신하십시오.
   a. 텍스트 편집기를 사용하여 Sample.jsp 및 Sample_All.jsp 파일을 여십시오. 이 파일은 다음 디렉토리에 있습니다.
      vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host \default_app\web
   b. <!-- SECTION 1 -->과 <!-- END OF SECTION 1 --> 표시기 사이의 Section 1을 Sample_All.jsp 코드에서 Sample.jsp로 복사하십시오. JSP 템플리트에 다음 코드가 삽입됩니다.

```
<!-- SECTION 1 -->
<%
Databeanamplebean testbean = new Databeanamplebean ();
com.ibm.commerce.bean.DatabeanManager.activate (testbean, request);
%>
<!-- END OF SECTION 1 -->
```

이 코드 섹션은 데이터 bean의 인스턴스를 생성합니다.

c. <!-- SECTION 2 -->와 <!-- END OF SECTION 2 --> 표시기 사이의 Section 2를 Sample_all.jsp에서 Sample.jsp로 복사하십시오. 그러면 JSP 템플리트에 다음 코드가 삽입됩니다.

```
<!-- 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 -->
```

이 파일을 저장하십시오.

이 코드 섹션에서는 testBean 데이터 bean 오브젝트의 getRequestProperties 메소드를 사용합니다. 특성을 통해 순환하며 브라우저에 이들을 표시합니다.

6. 다음과 같이 수정사항을 테스트하여 변수가 JSP 템플리트에 표시되어 있는지 검증하십시오.

   a. WebSphere Test Environment가 실행 중인지 확인하십시오.

   b. 브라우저에 다음 URL을 입력하십시오.

      ```
      http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd
      ```

제어기 명령의 특성을 보여주는 Sample JSP 파일이 표시됩니다. 출력은 다음과 같이 표시됩니다.

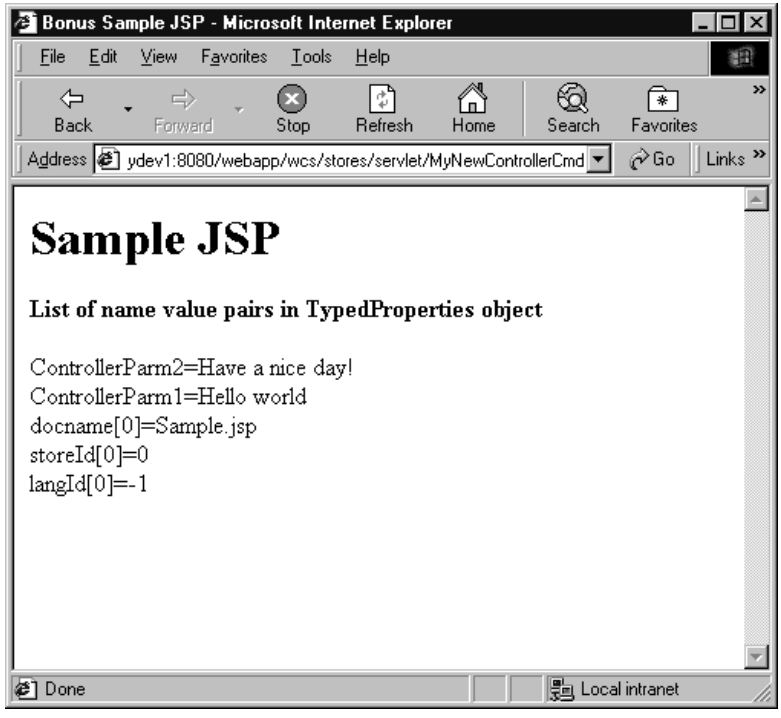


그림 32.

7. 현재 상태의 코드 버전을 작성하십시오. 버전 이름을 `mySample 1.3` 완료로 지정하십시오. 코드 버전화에 대한 자세한 내용은 238 페이지의 12단계를 참조하십시오.

validateParameters 메소드 수정

현재 명령에 있는 validateParameters 메소드는 실제로는 명령 스텁(stub)일 뿐입니다. 이 메소드는 다음 코드로 구성됩니다.

```
public void validateParameters() throws ECException {
     }
```

이 절에서는 명령에 사용자 정의 매개변수 확인을 추가한 후 JSP 템플리트에 해당 매개변수를 전달하게 됩니다.

validateParameters 메소드를 수정할 때는 매개변수용으로 사용되는 클래스에 새 필드를 추가합니다.

새 필드 작성: 기존 인터페이스 또는 클래스에 새 필드를 추가할 때는 VisualAge for Java의 필드 작성 SmartGuide를 사용할 수 있습니다. 이 절에서는 필드를 작성하기 위한 일반적인 단계에 대해 설명합니다. 인터페이스 및 클래스에 새 필드를 추가해야 할 경우, 다음 학습 절에 이 정보를 사용하십시오.

새 필드를 작성하려면 다음을 수행하십시오.

1. 새 필드를 추가 중인 인터페이스 또는 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오.
   필드 작성 SmartGuide가 열립니다.
2. **필드 이름** 필드에 새 필드의 이름을 입력하십시오.
3. 필드 유형을 지정하려면 다음 중 하나를 수행하십시오.
   - **필드 유형** 드롭 다운 목록에서 필드 유형을 선택하십시오.
   - 목록에 필수 필드 유형이 지정되어 있지 않은 경우, **찾아보기**를 누르십시오. **패턴** 필드에 필드 유형의 이름(또는 부분 이름)을 입력한 후 **확인**을 누르십시오.

   주: 학습에서 문자열의 필드 유형이 지정될 때마다 이것은 `java.lang` 패키지에서 나온 것입니다.
4. **초기값** 필드에 필드의 초기값을 입력하십시오. 초기값이 문자열인 경우, 값은 큰따옴표(“ ”)로 묶으십시오.
5. **액세스 수정자** 값에 대해 필요한 경우 해당 라디오 버튼(public, protected, none 또는 private)을 선택하십시오.
6. 필드에 대해 getter 및 setter 메소드를 생성하려는 경우, **getter 및 setter 메소드로 액세스** 선택란을 선택하십시오. 이 선택란을 선택할 경우, 다음과 같이 getter 및 setter 메소드용 특성도 지정해야 합니다.
   - getter 메소드의 경우 public, protected, private 또는 none 라디오 버튼 중 하나를 선택하십시오.
   - setter 메소드의 경우 public, protected, private 또는 none 라디오 버튼 중 하나를 선택하십시오.

7. **완료**를 누르십시오.

validateParameters 메소드를 수정하려면 다음을 수행하십시오.

1. MyNewControllerCommandImpl 클래스에 두 개의 필드를 작성해야 합니다. 첫 번째 필드는 입력 문자열에 사용되고 두 번째 필드는 입력 정수에 사용됩니다. 이러한 필드를 작성하려면 다음을 수행하십시오.
   a. **com.ibm.commerce.sample.commands** 패키지를 펼치십시오.
   b. **MyNewControllerCmdImpl** 클래스를 선택하십시오.
   c. 다음 값을 사용하여 클래스에 필드를 추가하십시오. 새 필드를 작성하는 방법에 대한 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.

속성 이름	값
필드 이름	inputString
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

   d. 다음 값을 사용하여 입력 정수용 기타 필드를 작성하십시오.

속성 이름	값
필드 이름	inputInteger
필드 유형	Integer **주:** **찾아보기**를 누르고 Integer를 입력하십시오. int를 선택하지 마십시오.
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

2. MyNewControllerCmdImpl 클래스에서 **performExecute** 메소드를 선택하십시오.
3. performExecute 메소드의 소스 코드에서 Section 3의 설명 표시를 제거하여 메소드로 다음 코드를 삽입하십시오.

```
// see how controller command pass in input variables to JSP
     rspProp.put("ControllerInput1", getInputString());
     rspProp.put("ControllerInput2", getInputInteger().toString());
```

이 코드는 제어기 명령에서 데이터 bean으로 변수를 전달합니다. 작업을 저장하십시오.

4. MyNewControllerCmdImpl 클래스에서 **validateParameters** 메소드를 선택하십시오.
5. validateParameters 소스 코드에서 Section 1의 설명 표시를 제거하여 메소드에 다음 코드를 삽입하십시오.

```
// 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));
```

작업을 저장하십시오.

앞의 코드 부분에서는 두 입력 매개변수를 확인합니다. try 블록은 첫 번째 매개변수가 있는지 여부를 판별하며, 없는 경우 예외가 발생합니다. 두 번째 매개변수는 선택적이므로, 이 코드는 매개변수가 누락되거나 잘못된 유형인 경우 매개변수의 값을 0으로 설정합니다.

6. 다음을 수행하여 명령을 테스트하십시오.
   a. PNS, EJB 서버 및 Servlet 엔진이 실행 중인지 확인하십시오.
   b. 브라우저에서 다음 URL을 입력하십시오.
      - 케이스 1: 매개변수 누락
        다음을 입력하십시오.

        ```
        http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd
        ```

        명령에 아무 매개변수도 전달되지 않았으므로, 매개변수가 누락되었다는 것을 나타내기 위해 일반 응용프로그램 오류가 표시됩니다. 결과가 다음 화면에 표시됩니다.

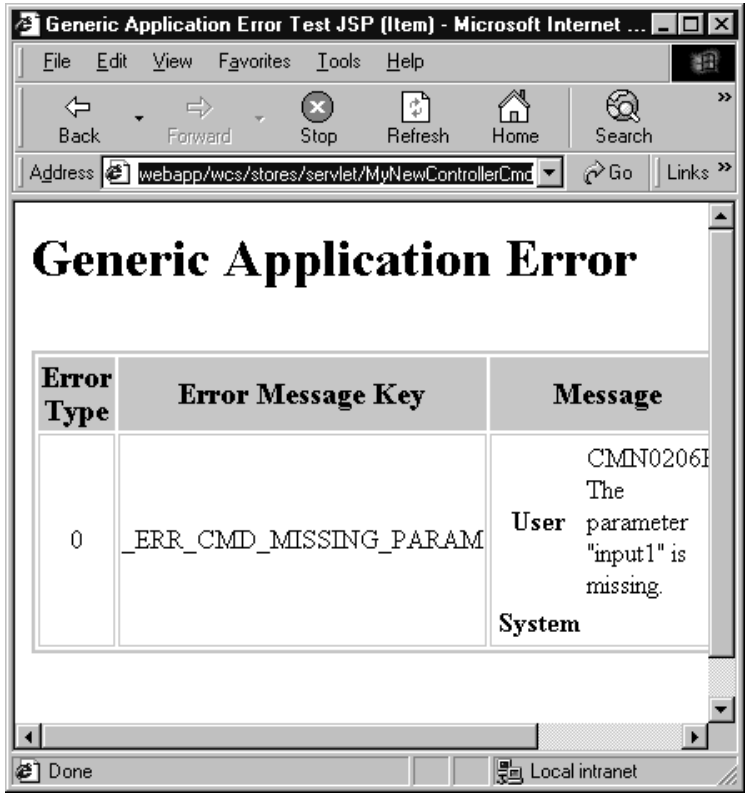


그림 33.

주: "페이지를 찾을 수 없습니다."라는 오류가 표시되면 Servlet 엔진이 중지된 것입니다. 자세한 내용은 WebSphere Test Environment 제어 센터를 확인하십시오. 일반 응용프로그램 오류 페이지 대신 견본 JSP 페이지가 표시되면, Servlet 엔진을 중지하고 다시 시작하거나 브라우저에서 페이지를 다시 로드해야 합니다.

- 케이스 2: 첫 번째 매개변수 올바름, 두 번째 매개변수 누락
  다음을 입력하십시오.

  ```
  http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=abc
  ```

  이 명령의 결과는 두 번째 매개변수가 생략되었음에도 불구하고 Sample JSP 페이지가 표시되는 것입니다. 다음 화면은 이러한 결과를 보여줍니다. 화면 다음에는 오류가 리턴되지 않은 이유에 대한 설명이 뒤따릅니다.

그림 34.

getInteger 메소드가 사용된 방식 때문에 오류가 리턴되지 않았습니다. 특히 코드의 setInputInteger(prop.getInteger("input2", 0)) 행은 input2에 기본값 0을 설정합니다. 매개변수가 누락되었거나 유형이 잘못된 경우, 이 기본값이 사용됩니다. 이 매개변수에 대한 유형 검사가 실행되도록 하려면 코드를 setInputInteger(prop.getInteger("input2"))로 변경하고 URL을 다시 입력하십시오(브라우저를 최신정보로 고치십시오). 일반 응용프로그램 오류 페이지가 표시됩니다.

주: 코드에서 setInputInteger(prop.getInteger("input2")) 수정을 테스트할 경우 다음 테스트 케이스로 계속하기 전에 setInputInteger(prop.getInteger("input2", 0))으로 다시 전환하십시오.

- 케이스 3: 첫 번째 매개변수 올바름, 두 번째 매개변수 올바르지 않음
  다음을 입력하십시오.

  ```
  http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
       input1=abc&input2=abc
  ```

  이 명령의 결과는 Sample JSP 페이지가 표시되고 두 번째 매개변수 값(정수)이 기본값 0으로 설정되는 것입니다. 앞의 테스트 케이스에서 설명된 대로 `getInteger` 메소드를 사용한 결과입니다. 결과가 다음 화면에 표시됩니다.

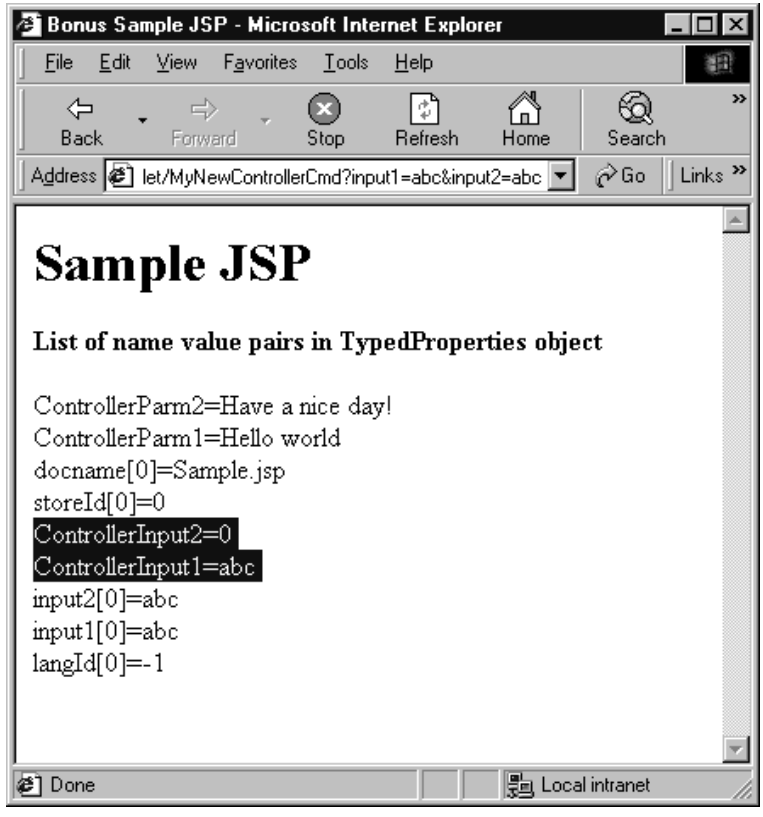


그림 35.

- 케이스 4: 두 개의 올바른 매개변수
  다음을 입력하십시오.

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

이 명령의 결과는 Sample JSP 페이지가 표시되고 두 입력값이 모두 입력된 대로 표시되는 것입니다. 결과가 다음 화면에 표시됩니다.

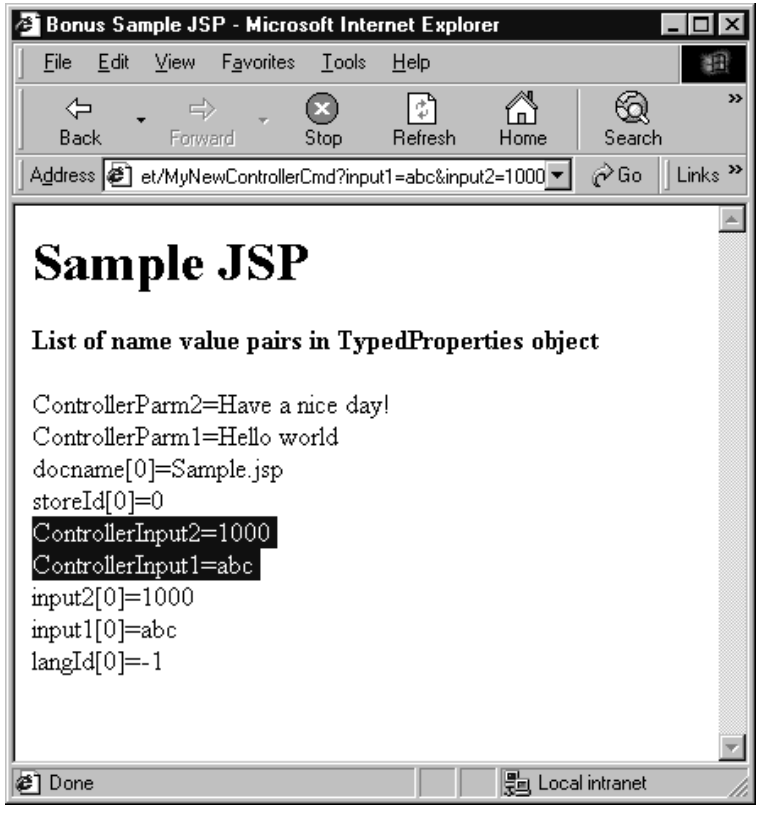


그림 36.

7. 현재 상태의 코드 버전을 작성하십시오. 버전의 이름을 mySample 1.4로 지정하십시오. 코드 버전화에 대한 자세한 내용은 238 페이지의 12단계를 참조하십시오.

태스크 명령 작성

일반적으로, 제어기 명령은 비즈니스 처리나 복잡한 기능을 표시합니다. 예를 들어 주문 처리와 관련된 모든 비즈니스 로직은 OrderProcessCmd 제어기 명령에서 캡슐화됩니다. 비즈니스 처리는 종종 보다 작은 특정 태스크로 분리될 수 있습니다.

예를 들어 OrderProcessCmd 제어기 명령 내에는 개별 작업 단위를 수행하기 위해 호출되는 몇 가지 태스크 명령이 있습니다. OrderProcessCmd 제어기 명령에서 호출되는 이러한 태스크 명령 중 하나는 CalculateOrderTaxTotalCmd입니다.

MyNewControllerCmdImpl은 현재 어떤 태스크 명령도 호출하지 않습니다. 이 연습에는 두 섹션이 있습니다. 첫 번째 섹션에서는 태스크 명령을 작성하게 됩니다. 두 번째 섹션에서는 제어기 명령의 performExecute 메소드가 태스크 명령을 호출하도록 수정합니다.

다음 코드 부분은 모든 설명을 제거한 상태로 MyNewControllerCmdImpl 클래스의 현재 performExecute 메소드를 보여줍니다.

```
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);
}
```

태스크 명령 코드 작성: 이 절에서는 새 태스크 명령을 작성하는 방법을 보여줍니다. 태스크 명령을 작성하려면 인터페이스 및 구현 클래스를 작성해야 합니다. 태스크 명령을 작성할 때 해당 인터페이스는 `com.ibm.commerce.commands.TaskCommand`를 확장해야 합니다. 구현 클래스는 `com.ibm.commerce.command.TaskCommandImpl`을 확장해야 합니다.

이 연습이 끝나면 새 명령인 *MyNewTaskCmd*가 생성됩니다. 모든 상점에서 이 명령을 사용하게 되며 각 상점은 명령을 동일한 방식으로 구현합니다.

여기서는 새 태스크 명령의 인터페이스에 필드와 메소드를 추가합니다.
MyNewControllerCmdImpl 클래스용 새 필드를 이미 작성했습니다. 인터페이스용 필드를 스스로 작성해 보십시오. 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.

메소드 작성: 이 절에서는 기존 클래스와 인터페이스에 메소드를 추가하기 위한 일반적인 단계에 대해 설명합니다. 다음 설명을 읽어보고, 메소드를 작성해야 할 경우 다시 참조하십시오.

새 메소드를 작성하려면 다음을 수행하십시오.

1. 메소드를 추가 중인 인터페이스 또는 클래스를 마우스 오른쪽 버튼으로 누른 후, **추가 > 메소드**를 선택하십시오.
   메소드 작성 SmartGuide가 열립니다.
2. **새 메소드 작성**이 선택되었는지 확인한 후 **다음**을 누르십시오.
3. **메소드 이름** 필드에 새 메소드의 이름을 입력하십시오.
4. 다음 중 하나를 수행하여 메소드의 리턴 유형을 지정하십시오.
   - **리턴 유형** 드롭 다운 목록에서 해당 리턴 유형을 선택하십시오. 예를 들어 `String`을 선택하십시오.
   - 목록에 리턴 유형이 지정되어 있지 않은 경우, **찾아보기**를 누르십시오. **패턴** 필드에 리턴 유형을 입력하고 **확인**을 누르십시오.

   주: 학습에서 `String`이 리턴 유형으로 지정될 때마다 이는 `java.lang` 패키지에서 나옵니다.
5. 메소드에 매개변수가 있는 경우, **추가**를 누르십시오. 매개변수 창에 매개변수 이름 및 기타 필수 정보를 지정한 후 **추가**를 누르십시오. 모든 매개변수가 추가되고 나면 **닫기**를 누르십시오.
6. **다음**을 누르십시오.
   속성 창이 열립니다.
7. 메소드에서 예외가 발생할 경우, 속성 창에서 **추가**를 누르십시오. **패턴** 필드에 예외의 이름을 입력한 후 **추가**를 누르십시오. 모든 예외가 추가되고 나면 **닫기**를 누르십시오.
8. **완료**를 누르십시오.
   메소드용 코드가 생성됩니다.

MyNewTaskCmd 명령을 작성하려면 다음을 수행하십시오.

1. **com.ibm.commerce.sample.commands** 패키지를 펼치십시오.

2. **MyNewTaskCmd** 인터페이스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오.
3. 필드 작성 SmartGuide를 사용하여 인터페이스가 사용할 기본 구현 클래스를 지정하는 필드를 작성하십시오. 다음 테이블의 값을 사용하십시오. 필드 작성에 대한 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.

속성 이름	값
필드 이름	defaultCommandClassName
필드 유형	String
초기값	"com.ibm.commerce.sample.commands.MyNewTaskCmdImpl"

필드용 코드가 생성되면 다음과 같이 표시됩니다.

```
java.lang.String defaultCommandClassName =
    "com.ibm.commerce.sample.commands.MyNewTaskCmdImpl";
```

전체 사이트에 동일한 구현 클래스가 사용되며 기본 특성이 명령에 전달되지 않으므로 코드에 기본 구현을 바로 지정할 수 있습니다. 복수 구현이 있거나 기본 특성(CMDREG 테이블에 저장)을 가진 명령이 있는 경우, CMDREG 테이블에 명령을 등록하여 인터페이스 및 구현 클래스 간의 맵핑을 작성해야 합니다.

4. 다음을 수행하여 MyNewTaskCmd 인터페이스에 새 메소드를 추가하십시오.
   a. **MyNewTaskCmd** 인터페이스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오. 메소드 작성 SmartGuide를 사용하여 다음 단계에 지정된 값으로 새 필드를 작성하십시오. 메소드 작성에 대한 자세한 내용은 252 페이지의 『메소드 작성』을 참조하십시오.
   b. 다음 값을 사용하여 고객의 보너스 점수 잔고를 검색하는 새 메소드를 작성하십시오.

속성 이름	값
메소드 이름	getOldBonusPoint
리턴 유형	String
매개변수	없음
예외	없음

주: 앞에서 작성된 상속 추상 메소드가 MyNewTaskCmdImpl 구현 클래스에 구현되지 않음을 나타내는 오류가 표시될 수 있습니다. 다음 단계에서는 이러한 문제를 해결합니다.

c. 다음 값을 사용하여 태스크 명령에서 사용자 ID 출력을 검색하는 새 메소드를 작성하십시오.

속성 이름	값
메소드 이름	getTask_output_userId
리턴 유형	String
매개변수	없음
예외	없음

d. 다음 값을 사용하여 태스크 명령에서 출력값을 검색하는 새 메소드를 작성하십시오.

속성 이름	값
메소드 이름	getTask_output1
리턴 유형	String
매개변수	없음
예외	없음

e. 다음 값을 사용하여 태스크 명령에 첫 번째 입력값을 설정하는 새 메소드를 작성하십시오.

속성 이름	값
메소드 이름	setTask_input1
리턴 유형	void
매개변수 이름	newTask_input1
참조 유형	String
예외	없음

f. 다음 값을 사용하여 태스크 명령에 두 번째 입력값을 설정하는 새 메소드를 작성하십시오.

속성 이름	값
메소드 이름	setTask_input2

속성 이름	값
리턴 유형	void
매개변수 이름	newTask_input2
Primitive 타입	int
예외	없음

5. 다음을 수행하여 MyNewTaskCmdImpl 클래스에 새 필드를 추가하십시오.
   a. **MyNewTaskCmdImpl** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오. 필드 작성 SmartGuide를 사용하여 다음 단계에 지정된 값으로 새 필드를 작성하십시오. 새 필드 작성에 대한 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.
   b. 다음 값을 사용하여 구현 클래스에 새 필드를 작성하십시오.

속성 이름	값
필드 이름	task_input1
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

   c. 다음 값을 사용하여 구현 클래스에 새 필드를 작성하십시오.

속성 이름	값
필드 이름	task_input2
필드 유형	int
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

d. 다음 값을 사용하여 구현 클래스에 새 필드를 작성하십시오.

속성 이름	값
필드 이름	task_output_userId
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

e. 다음 값을 사용하여 구현 클래스에 새 필드를 작성하십시오.

속성 이름	값
필드 이름	oldBonusPoint
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

f. 다음 값을 사용하여 구현 클래스에 새 필드를 작성하십시오.

속성 이름	값
필드 이름	task_output1
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

6. **MyNewTaskCmdImpl** 클래스의 **performExecute** 메소드를 선택하여 해당 소스 코드를 보십시오.
7. 소스 코드에서, Section 1의 주석 표시를 제거하여 메소드에 다음 코드를 삽입하십시오.

```
// modify the task_input1 and see it in the NVP list

    setTask_output1( "Hello ! " + getTask_input1() );
```

작업을 저장하십시오

앞의 코드 섹션에서는 명령의 출력으로 사용 가능한 새 속성을 작성합니다.

태스크 명령 호출: 일단 태스크 명령을 작성하면 제어기 명령으로부터 명령을 호출해야 합니다. 다음 단계에서는 이러한 방식으로 제어기 명령을 수정하는 방법에 대해 설명합니다.

1. 워크벤치에서 **MyNewControllerCmdImpl** 클래스의 **performExecute** 메소드를 선택하십시오.
2. 소스 분할창에서 태스크 명령을 호출할 Section 4의 설명 표시를 제거하여 메소드에 다음 코드가 삽입됩니다.

```
// 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;
}
```

작업을 저장하십시오.

앞의 코드 부분에서는 명령 팩토리를 사용하여 새 태스크 명령을 작성합니다. 또한 명령 컨텍스트 설정, 태스크 명령의 실행 호출 및 태스크 명령의 출력 매개변수를 검색합니다.

3. 다음과 같이 제어기 명령의 URL을 입력하여 명령을 테스트하십시오.

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

Sample JSP는 작업 출력 값을 포함하여 요청 오브젝트의 이름 값 쌍 목록을 표시합니다. 결과는 다음과 같습니다.

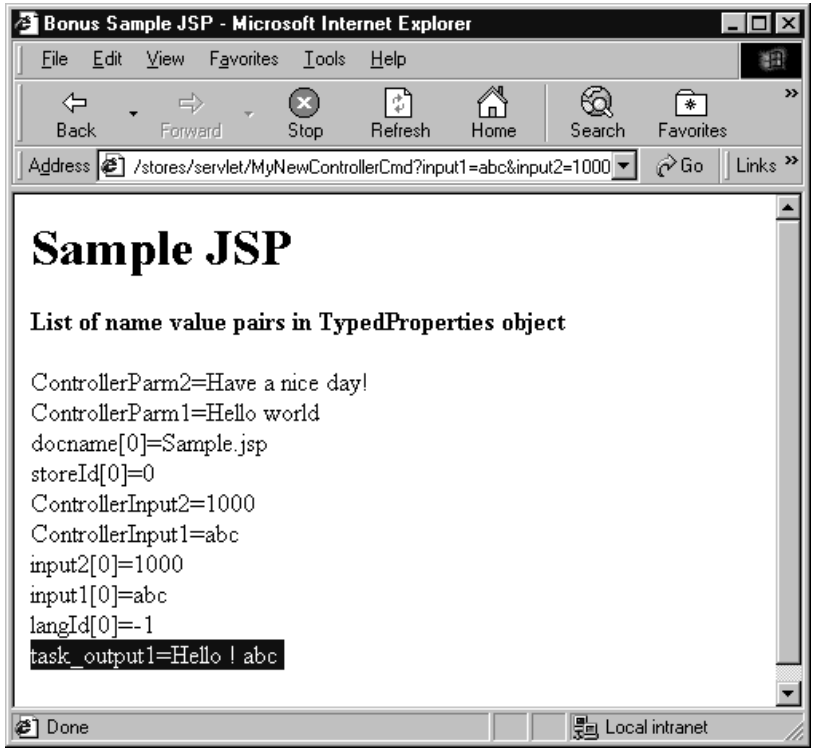


그림 37.

4. 현재 상태의 코드 버전을 작성하십시오. 버전의 이름을 mySample 1.5로 지정하십시오. 코드 버전화에 대한 자세한 내용은 238 페이지의 12단계를 참조하십시오.

사용자 ID 유효성 확인

다음 단계에서는 사용자가 입력한 값이 등록된 사용자 값인지 여부를 확인하기 위해 UserRegistryAccessBean을 사용하는 태스크 명령을 수정해야 합니다. 태스크 명령 뿐만 아니라, DataBeanSampleBean도 수정해야 합니다.

사용자 ID 유효성 확인을 위한 MyNewTaskCmdImpl 수정:

UserRegistryAccessBean을 사용하여 사용자 ID 유효성을 확인하도록 MyNewTaskCmdImpl 클래스에서 performExecute 메소드를 수정해야 합니다. performExecute 메소드에 이러한 새 기능을 추가하려면 다음을 수행하십시오.

1. 워크벤치에서 **MyNewTaskCmdImpl** 클래스의 **performExecute** 메소드를 선택하십시오.
2. 소스 분할창에서 performExecute 메소드의 Section 2의 설명 표시를 제거하여 메소드에 다음 코드가 삽입됩니다.

```
// 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);
```

작업을 저장하십시오.

DataBeanSampleBean 수정: 사전 정의된 입력 매개변수를 사용하려면 DataBeanSampleBean을 수정해야 합니다. 이 bean을 수정하려면 다음을 수행하십시오.

1. 워크벤치에서 **com.ibm.commerce.sample.databeans** 패키지를 펼치십시오.
2. 다음을 수행하여 데이터 bean에 새 필드를 추가하십시오.
   a. **DataBeanSampleBean** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오. 필드 작성 SmartGuide를 사용하여 다음 단계에 지정된 값으로 새 필드를 작성하십시오. 필드 작성에 대한 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.

b. 다음 값을 사용하여 데이터 bean에 새 필드를 추가하십시오.

속성 이름	값
필드 이름	input1
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

c. 다음 값을 사용하여 데이터 bean에 새 필드를 추가하십시오.

속성 이름	값
필드 이름	input2
필드 유형	int
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

d. 다음 값을 사용하여 데이터 bean에 새 필드를 추가하십시오.

속성 이름	값
필드 이름	task_output_userId
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

3. DataBeanSampleBean 클래스의 **populate** 메소드를 선택하여 해당 소스 코드를 보십시오.
4. 소스 코드에서 Section 1A 및 Section 1B 설명 표시를 제거하여 메소드에 다음 코드가 삽입됩니다.

```
//// 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 ////
```

작업을 저장하십시오.

앞의 코드 부분에서는 getRequestProperties 메소드를 사용하여 제어기 명령에서 데이터 필드 값을 가져옵니다.

사용자 ID 유효성 확인을 위한 Sample.jsp 수정: 사용자 ID 유효성 확인을 수행하려면 현재 JSP 템플리트를 수정해야 합니다. Sample.jsp 파일을 수정하려면 다음을 수행하십시오.

1. 텍스트 편집기에서 Sample.jsp 및 Sample_All.jsp 파일을 여십시오.
2. <!-- SECTION 3 -->과 <!-- END OF SECTION 3 --> 표시기 사이의 Section 3을 Sample_all.jsp에서 Sample.jsp로 복사하십시오. 다음 코드가 JSP 템플리트로 삽입됩니다.

```
<!-- 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 -->
```

Sample.jsp 파일을 저장하십시오.

3. 브라우저를 열어 다음 URL을 입력하십시오.

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

다음 화면에 표시된 대로 브라우저는 input1의 값이 올바른 사용자 ID가 아니라는 것을 나타내는 Sample JSP 파일을 보여줍니다.

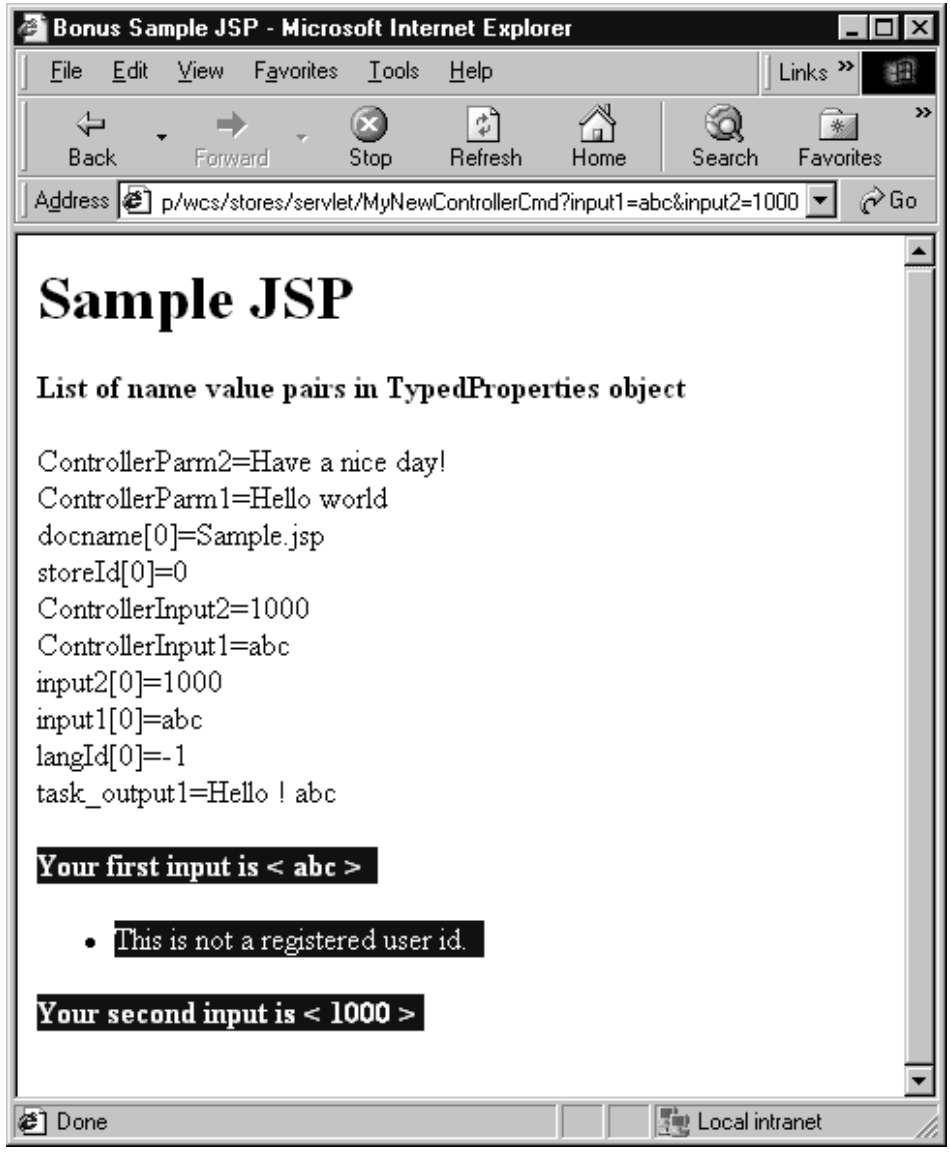


그림 38.

4. input1의 값이 올바른 사용자 ID일 때의 결과를 표시하려면 다음 URL을 입력하십시오.

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
    input1=wcsadmin&input2=1000
```

다음 화면에 표시됩니다.

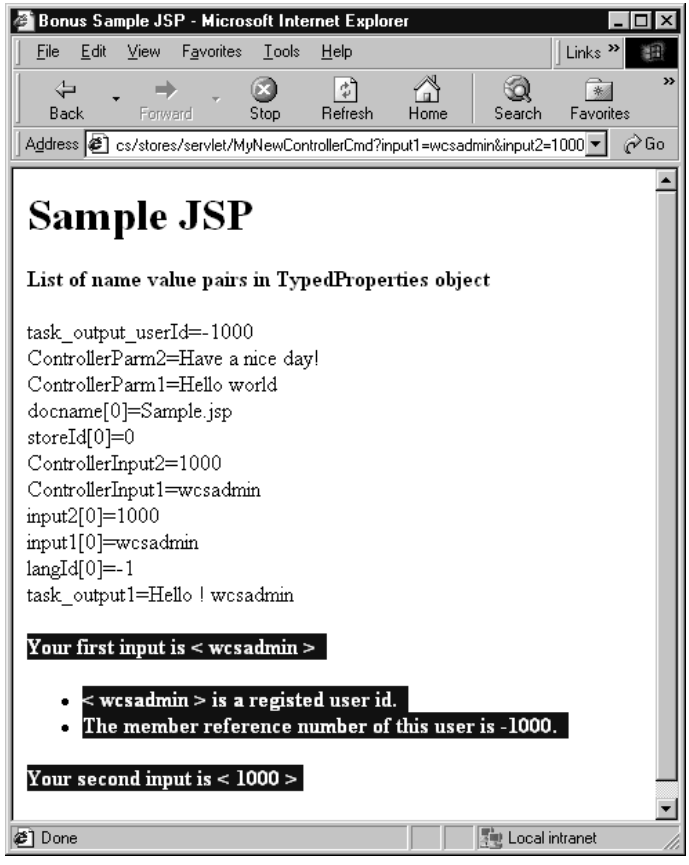


그림 39.

5. 현재 상태의 코드 버전을 작성하십시오. 버전의 이름을 mySample 1.6으로 지정하십시오. 코드 버전화에 대한 자세한 내용은 238 페이지의 12단계를 참조하십시오.

새 엔티티 bean 작성

이 절에서는 VisualAge for Java를 사용하여 새 엔티티 bean을 작성하는 방법에 대해 설명합니다. 이 예제 시나리오에는 상거래 응용프로그램의 각 사용자마다 보너스 점수 기록이 점차 올라가는 비즈니스 요구사항이 있습니다. WebSphere Commerce 데이터베이스 스키마에는 정보가 없으므로, 이 정보를 보유할 새 데이

터베이스 테이블을 작성해야 합니다. WebSphere Commerce 프로그래밍 모델에 따라 일단 데이터베이스 테이블을 작성하면 데이터에 액세스하기 위한 엔티티 bean(EJB)을 작성해야 합니다

이 예에서는 VisualAge for Java SmartGuide를 사용하여 이 엔티티 bean을 작성하게 됩니다.

새 데이터베이스 테이블 작성

엔티티 bean 작성 준비 단계에서 우선 데이터베이스 테이블을 작성해야 합니다. 작성할 테이블의 이름은 Bonus입니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. 스크립트 창에서 다음을 입력하십시오.

```
connect to your_database_name;
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 (users_id) on delete cascade)
```

여기서 *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.

Bonus 테이블이 작성됩니다.

주: 누군가 이전에 이 데이터베이스를 사용하여 예를 실행한 경우, Bonus 테이블을 작성하기 전에 다음 명령을 실행해야 합니다.

```
drop table Bonus
```

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 테이블을 작성하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.

3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서, 다음 SQL 문을 입력하십시오.

```
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);
```

   Enter를 눌러 SQL 문을 실행하십시오. BONUS 테이블이 이제 작성됩니다.

   주: 누군가 이전에 이 데이터베이스를 사용하여 예를 실행한 경우, Bonus 테이블을 작성하기 전에 다음 명령을 실행해야 합니다.

```
drop table Bonus;
```

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL 문을 실행하십시오.

BonusBean 엔티티 bean 작성

일단 데이터베이스가 작성되면 새 엔티티 bean 작성을 시작할 준비가 된 것입니다. 다음 단계에서는 VisualAge for Java가 사용됩니다. 새 엔티티 bean을 작성하려면 다음을 수행하십시오.

1. EJB 그룹을 작성하십시오. EJB 그룹은 엔터프라이즈 bean을 구성할 수 있도록 해주는 논리적 그룹입니다. 해당 그룹에 있는 모든 엔터프라이즈 bean마다 반복될 EJB 그룹에 대한 글로벌 조작을 수행할 수 있습니다. 예를 들어 한 EJB 그룹이 EJB JAR 파일에 반출되도록 선택한 경우, 해당 그룹에 있는 모든 엔터프라이즈 bean이 반출됩니다.
   이 경우 Bonus 테이블 사용자 정의와 관련된 모든 엔터프라이즈 bean을 구성할 EJB 그룹을 작성합니다.
   EJB 그룹을 작성하려면 다음을 수행하십시오.
   a. 워크벤치에서 **EJB** 탭을 누르십시오.
   b. **EJB** 메뉴에서 **추가 > EJB 그룹**을 선택하십시오.
      EJB 그룹 추가 SmartGuide가 열립니다.
   c. **프로젝트** 필드에 _WCSSamplesEntityBeansProject를 입력하십시오.

주: 전개를 위해 엔티티 bean 코드를 자체 프로젝트 내에 저장해야 합니다.

d. **새 EJB 그룹 이름 작성** 필드에 WCSSamplesEntityBeans를 입력한 후 **완료**를 누르십시오

2. 새 엔티티 bean을 작성하십시오.

   새 엔티티 bean을 작성하려면 다음을 수행하십시오.

   a. 엔터프라이즈 bean 분할창에서 **WCSSamplesEntityBeans** EJB 그룹을 마우스 오른쪽 버튼으로 누른 후 **추가 > 엔터프라이즈 Bean**을 선택하십시오.

      엔터프라이즈 Bean 작성 SmartGuide가 열립니다.

   b. 다음 정보를 입력하십시오.

속성	값
bean 이름	Bonus **주:** 이름 지정 규칙은 액세스할 수 있는 테이블과 같은 이름으로 엔티티 bean을 호출하는 것입니다.
bean 유형	Entity bean with container-managed persistence (CMP) fields
새 bean 클래스 작성	enable
프로젝트	_WCSamplesEntityBeansProject
패키지	com.ibm.commerce.sample.objects
클래스 이름	Bonusbean
최상위 클래스	com.ibm.commerce.base.objects.ECEntityBean

      다음을 누르십시오.

   c. **bean에 CMP 필드 추가** 텍스트 상자 옆의 **추가** 버튼을 눌러 BONUS 테이블에 MEMBERID 열의 필드를 추가하십시오.

      CMP 필드 작성 SmartGuide가 열립니다.

   d. 다음 정보를 입력하십시오.

속성	값
필드 이름	memberId
필드 유형	Long **주:** *long*이 아닌 *Long* 데이터 유형을 사용해야 합니다.

속성	값
키 필드	사용 가능

완료를 누르십시오.

e. **추가**를 다시 눌러 BONUS 테이블에 BONUSPOINT 열의 필드를 추가하십시오.

f. 다음 정보를 가진 다른 필드를 작성하십시오.

속성	값
필드 이름	bonusPoint
필드 유형	정수 **주:** *int*가 아닌 *Integer* 데이터 유형을 사용해야 합니다.
getter 및 **setter** 메소드로 액세스	사용 가능
getter 및 **setter** 메소드를 원격 인터페이스로 승격	사용 가능

그런 다음 CMP 필드 작성 창에서 **완료**를 누르십시오.

g. 다음을 수행하여 액세스 제어를 사용하여 엔터프라이즈 bean을 보호하십시오.

1) **어떤 인터페이스가 원격 인터페이스를 확장해야 합니까?** 옆의 **추가**를 누르십시오.

2) `com.ibm.commerce.security.Protectable`을 **패턴** 필드에 입력하고 **추가**를 누르십시오. **종료**를 눌러 창을 닫으십시오.

h. **완료**를 다시 누르십시오.

Bonus 엔티티가 엔터프라이즈 bean으로 작성됩니다.

3. 다음을 수행하여 엔티티 bean의 분리 레벨을 설정하십시오.

a. **Bonus** bean을 마우스 오른쪽 버튼으로 누른 후 **특성**을 선택하십시오.

b. **분리 레벨** 드롭 다운 목록에서 **TRANSACTION_READ_COMMITTED**를 선택한 후 **확인**을 누르십시오.

4. 새 엔터프라이즈 bean 작성시, VisualAge for Java는 해당 getEntityContext() 및 setEntityContext(EntityContext) 메소드는 물론 EntityContext 필드를 bean에 생성합니다. WebSphere Commerce 프로그래밍 모델에 따라 새 bean은 com.ibm.commerce.base.objects.ECEntityBean 클래스를 확장하며

ECEntityBean은 필드 및 메소드의 자체 구현을 제공합니다. EntityContext, getEntityContext 및 setEntityContext를 대체하지 말아야 하므로 이제는 생성된 필드 및 메소드를 사용자 bean에서 삭제해야 합니다.
생성된 EntityContext 필드와 관련 getter 및 setter 메소드를 삭제하려면 다음을 수행하십시오.

a. 유형 분할창에서 **BonusBean** 클래스를 선택하십시오.
구성원 분할창은 해당 클래스에 대한 필드 및 메소드를 표시합니다.

주: 유형 분할창이 보이지 않는 경우, 특성 분할창에서 C/I 아이콘(클래스/인터페이스)을 누르십시오. 유형 분할창이 열립니다.

b. 구성원 분할창에서 다음을 수행하십시오.

1) **entityContext** 필드를 마우스 오른쪽 버튼으로 누르고 **삭제**를 선택하십시오.

2) **getEntityContext()** 메소드를 마우스 오른쪽 버튼으로 누르고 **삭제**를 선택하십시오.

3) **setEntityContext(EntityContext)** 메소드를 마우스 오른쪽 버튼으로 누르고 **삭제**를 선택하십시오.

c. 작업을 저장하십시오(Ctrl+S).

5. 다음을 수행하여 새 getMemberId 메소드를 엔터프라이즈 bean에 추가하십시오.

a. **BonusBean** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오.
메소드 작성 SmartGuide가 열립니다.

b. 다음 테이블에 지정된 값을 사용하여 새 메소드를 작성하십시오. 새 메소드 작성에 대한 자세한 내용은 252 페이지의 『메소드 작성』을 참조하십시오.

표 11.

속성 이름	값
메소드 이름	getMemberId
리턴 유형	Long
매개변수	없음

표 11. (계속)

속성 이름	값
예외	없음

c. 새 메소드가 생성되면, 소스 코드를 보십시오.

d. 기본값으로, 메소드에는 다음 코드가 들어 있습니다.

```
return null;
```

이 코드를 다음과 같이 변경하십시오

```
return memberId;
```

e. **getMemberId** 메소드를 마우스 오른쪽 버튼으로 누르고 **추가 > EJB 원격 인터페이스**를 선택하여 원격 인터페이스에 새 메소드를 추가하십시오.

6. BonusbeanFinderHelper에 FinderHelper 필드를 추가하십시오.
이 인터페이스에는 다음 단계에서 작성되는 FinderHelper 메소드에 해당하는 검색 절이 있습니다. FinderHelper 필드를 추가하려면 다음을 수행하십시오.

a. 유형 분할창에서 **BonusBeanFinderHelper** 인터페이스를 누르십시오.

b. 소스 분할창의 코드를 다음과 같이 수정하십시오.

```
public interface BonusbeanFinderHelper {
public static final String
                                    findByMemberIdWhereClause = " (MEMBERID = ?) ";
}
```

주: "WhereClause" 구문은 매우 중요합니다. FinderHelper 메소드에 사용된 메소드 이름과 일치해야 합니다. 이 경우, `findByMemberIdWhereClause`의 "findByMemberId"는 다음 단계에서 작성할 메소드의 이름과 정확히 일치합니다(`findByMemberId`).

7. BonusHome 인터페이스에 FinderHelper 메소드를 추가하십시오.
FinderHelper 메소드를 추가하려면 다음을 수행하십시오.

a. 유형 분할창에서 **BonusHome** 인터페이스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오.
메소드 작성 SmartGuide가 열립니다.

b. **새 메소드 작성**을 선택한 후 **다음**을 누르십시오.

c. **메소드 이름** 필드에 `findByMemberId`를 입력하십시오.

d. **ReturnType** 필드에 Bonus를 입력하십시오.

e. **이 메소드의 매개변수는?** 옆에 있는 **추가**를 누르십시오.
매개변수 창이 열립니다.

f. **이름** 필드에 argMemberId를 입력하십시오.

g. **참조 유형**을 선택하고 Long을 입력하십시오. **추가**를 누른 다음 **종료**를 누르십시오.

h. **다음**을 누르십시오.

i. **발생할 예외는?** 필드 옆의 **추가**를 누르고 **패턴** 필드에 RemoteException을 입력한 후 **추가**를 누르십시오. 이렇게 하면 속성 창에 java.rmi.RemoteException 예외가 추가됩니다(속성 창은 예외 창 뒤에 위치할 수 있습니다).

j. 예외 창의 **패턴** 필드에 FinderException을 입력하고 **추가**를 누른 후 **닫기**를 누르십시오. javax.ejb.FinderException 예외가 속성 창에 나열됩니다.

k. **완료**를 누르십시오.

8. 새 ejbCreate 메소드를 EJB에 추가하십시오. 이 메소드는 홈 인터페이스로 프롬프트 됩니다. 생성되는 액세스 bean은 이를 사용할 수 있습니다. 이 메소드를 작성하려면 다음을 수행하십시오.

a. 유형 분할창에서 **BonusBean** 클래스를 선택하십시오.

b. 구성원 분할창에서 **ejbCreate(Long)**를 누르십시오.

c. 코드를 다음과 같이 수정하십시오.

```
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. 코드를 저장하면 VisualAge for Java에서는 구성원 분할창에 **ejbCreate(Long, Integer)**라는 새 메소드를 작성합니다.

e. 원래의 **ejbCreate(Long)** 메소드를 마우스 오른쪽 버튼으로 누른 후 **삭제**를 선택하십시오.

9. 홈 인터페이스에 새 메소드를 추가하십시오. 액세스 bean 클래스에서 이 메소드를 사용할 수 있습니다.
   a. Bonusbean 클래스의 **ejbCreate(Long, Integer)** 메소드를 마우스 오른쪽 버튼으로 누른 후 **추가 > EJB 홈 인터페이스**를 선택하십시오.
10. 다음을 수행하여 getOwner 메소드를 갱신하십시오.
   a. 유형 분할창에서 **BonusBean** 클래스를 선택하십시오.
   b. 구성원 분할창에서 **getOwner()** 메소드를 누르십시오.

      주: 상속된 메소드 보기를 선택하면 두 가지의 getOwner() 메소드를 보게 됩니다. 하나는 ECEntityBean 클래스로부터 상속됩니다. 이것은 이 단계에서 선택해야 하는 메소드가 아닙니다. 반드시 BonusBean 클래스에 특정한 getOwner 메소드를 선택하십시오.
   c. getOwner 메소드에 대한 소스 코드는 다음과 같이 표시됩니다.

```
public Long getOwner()
   throws Exception, java.rmi.RemoteException {
return null;
   }
```

      메소드가 리턴하는 값을 변경해야 합니다. 변경해야 하는 코드의 부분은 다음에서 굵은체로 표시됩니다.

```
public Long getOwner()
   throws Exception, java.rmi.RemoteException {
      return getMemberId();
   }
```

      작업을 저장하십시오.
   d. 구성원 분할창에서 **fulfills(Long, String)** 메소드를 누르십시오.

      주: 상속된 메소드 보기를 선택한 경우, 두 가지 fulfills(Long, String) 메소드가 표시됩니다. 하나는 ECEntityBean 클래스로부터 상속됩니다. 이것은 이 단계에서 선택해야 하는 메소드가 아닙니다. BonusBean 클래스에 고유한 fulfills(Long, String) 메소드를 선택하십시오.
   e. fulfills(Long, String) 메소드의 소스 코드는 다음과 같이 표시됩니다.

```
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException {
      return false;
  }
```

사용자가 이행해야 하는 관계를 지정해야 합니다. 이를 수행하려면, 다음에서 굵은체로 표시된 코드 부분을 변경해야 합니다.

```
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException {
      if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }      return false;
  }
```

작업을 저장하십시오.

11. BONUS 데이터베이스 테이블을 Bonusbean에 맵하십시오. 데이터베이스 스키마를 Bonusbean 엔티티에 맵핑할 때의 첫 번째 단계는 VisualAge for Java의 도구를 사용하여 데이터베이스 스키마를 작성하는 것입니다. 다음을 수행하여 스키마를 작성하십시오.
    a. 작업 영역 창의 **EJB** 메뉴에서 **열기 > 데이터베이스 스키마**를 선택하십시오.
    b. **스키마** 메뉴에서 **스키마 반입/반출 > 데이터베이스에서 스키마 반입**을 선택하십시오.
       필요한 정보 창이 열립니다.
    c. **스키마 이름** 필드에 WCSSamples를 입력한 후 **확인**을 누르십시오.
       데이터베이스 연결 정보 창이 열립니다.
    d. 다음 정보를 입력하십시오.

속성	DB2 **DB2 값**	Oracle **Oracle 값**
연결 유형	COM.ibm.db2.jdbc.app.DB2Driver	Oracle.jdbc.driver.OracleDriver
데이터 소스	jdbc:db2:*wcs_db_name*	jdbc:oracle:thin:@*hostname*:*port*:*SID*
사용자 이름	*cs_db_user_name*	*wcs_db_user_name*
암호	*wcs_db_password*	*wcs_db_password*

값은 다음과 같이 바뀝니다.

- DB2 *wcs_database_name*은 WebSphere Commerce 데이터베이스의 이름입니다
- Oracle *hostname*은 Oracle 호스트 이름입니다
- Oracle *port*는 Oracle 데이터베이스의 포트 번호입니다(예: 1521).
- *wcs_db_user_name*은 데이터베이스의 사용자 이름입니다.
- *wcs_db_password*는 데이터베이스 암호입니다.

확인을 누르십시오.
테이블 선택 창이 열립니다.

e. **규정자** 목록에서 데이터베이스 규정자(데이터베이스 사용자 이름 또는 시스템 이름이 될 수 있음)를 선택한 후 **테이블 목록 빌드**를 누르십시오. 사용할 수 있는 데이터베이스 테이블 목록이 로드됩니다.

f. 테이블 패널에서 **Bonus**를 선택하고 **확인**을 누르십시오. 잠시만 기다리십시오.

g. 스키마 브라우저에서 새로 추가된 테이블을 눌러 테이블의 두 열이 모두 표시되는지 살펴보십시오.

h. **Bonus** 테이블을 마우스 오른쪽 버튼으로 누른 후 **테이블 편집**을 선택하십시오.
테이블 편집기가 열립니다.

i. **규정자** 필드에 있는 항목을 제거하십시오. 규정자 정보를 제거하여 코드를 다른 데이터베이스를 사용하고 있는 다른 시스템으로 전개할 수 있습니다.

j. Oracle 다음과 같이 열 데이터 유형을 수정하십시오.

   1) **MEMBERID** 열을 선택하고 **편집**을 누르십시오. 드롭 다운 목록 유형에서 **BIGINT**를 선택하고 **확인**을 누르십시오.

   2) **BONUSPOINT** 열을 선택하고 **편집**을 누르십시오. 드롭 다운 목록 유형에서 **INTEGER**를 선택하고 **확인**을 누르십시오.

k. 테이블 편집기를 종료하려면 **확인**을 누르십시오.

l. **스키마** 메뉴에서 **스키마 저장**을 선택하십시오
스키마 저장 창이 열립니다.

m. 다음 정보를 입력하십시오.

속성	값
프로젝트	_WCSamplesEntityBeansProject
패키지	com.ibm.commerce.sample.objects
클래스 이름	WCSSamplesSchema

완료를 누르고 스키마 브라우저를 닫으십시오.

12. 일단 스키마가 작성되면 스키마 맵을 작성할 수 있습니다. BONUS 테이블과 Bonusbean 엔티티 간에 이 맵을 작성하려면 다음을 수행하십시오.
    a. **EJB** 메뉴에서 **열기 > 스키마 맵**을 선택하십시오. 맵 브라우저가 열립니다.
    b. 맵 브라우저의 **데이터스토어 맵** 메뉴에서 **새 EJB 그룹 맵**을 선택하십시오.
    새 데이터스토어 맵 창이 열립니다.
    c. 다음 정보를 입력하십시오.

속성	값
이름	WCS Samples
EJB 그룹	WCSSamplesEntityBeans
스키마	WCSSamples

확인을 누르십시오.

    d. 데이터스토어 맵 패널에서 **WCS 견본**을 누르십시오.
    e. 지속 클래스 패널에서 **Bonus**를 누르십시오.
    f. **테이블 맵** 메뉴에서 **새 테이블 맵 > 상속하지 않고 테이블 맵 추가**를 선택하십시오.
    g. **테이블** 드롭 다운 목록에서 **Bonus**를 선택한 후 **확인**을 누르십시오.
    h. 테이블 맵 패널에서 **Bonus**를 선택하고 마우스 오른쪽 버튼을 누른 후 **특성 맵 편집**을 선택하십시오.
    특성 맵 편집기가 열립니다.

i. 다음과 같이 속성을 설정하십시오.

클래스 속성	맵 유형	테이블 열
memberId	Simple	MEMBERID
bonusPoint	Simple	BONUSPOINT

확인을 누르십시오.

j. **데이터스토어 맵** 메뉴에서 **데이터스토어 맵 저장**을 누르십시오.
데이터스토어 맵 저장이 열립니다.

k. 다음 정보를 입력하십시오.

속성	값
프로젝트	_WCSamplesEntityBeansProject
패키지	com.ibm.commerce.sample.objects
클래스 이름	WCSSamplesMap

완료를 누르고 맵 브라우저를 닫으십시오.

13. 일단 Bonusbean 엔티티가 작성되고 스키마가 맵핑되면 엔티티 bean의 액세스 bean을 작성해야 합니다. 이 액세스 bean은 응용프로그램이 Bonus 엔티티 bean에 들어 있는 정보에 더 간단하게 액세스하도록 합니다. VisualAge for Java의 도구는 이미 작성한 엔티티에 기초하여 액세스 bean을 생성하는 데 사용됩니다(특히 액세스 bean은 원격 인터페이스로 승격된 메소드만 사용합니다). Bonus 엔티티 bean의 액세스 bean을 작성하려면 다음을 수행하십시오.

a. 워크벤치에서 EJB 탭을 선택하고 **Bonus** 엔터프라이즈 bean을 마우스 오른쪽 버튼으로 누른 후 **추가 > 액세스 bean**을 선택하십시오.
액세스 bean 작성 SmartGuide 창이 열립니다(다소 시간이 걸립니다).

b. 다음 정보가 입력되었는지 확인하십시오.

속성	값
EJB 그룹	WCSSamplesEntityBeans
엔터프라이즈 **bean**	Bonus
액세스 **bean** 이름	BonusAccessbean
액세스 **bean** 유형	Copy Helper for an Entity Bean

다음을 누르십시오.

c. **인수없는 생성자의 홈 메소드 선택** 드롭 다운 목록에서 **findByMemberId(Long)**를 선택하십시오.

d. **init_argMemberId**의 경우(초기 특성 열에서), **변환기**를 com.ibm.commerce.base.objects.WCSStringConverter로 설정하고 **다음**을 누르십시오.

e. bonusPoint의 경우, **CopyHelper**를 선택했는지 확인하고 **변환기** 값을 com.ibm.commerce.base.objects.WCSStringConverter로 설정한 후 **완료**를 누르십시오.

주: 구성원 ID 필드는 수정하지 않아도 됩니다.

f. "코드 생성 완료" 메시지가 표시되면 **확인**을 누르십시오.

프로젝트 탭으로 전환하고 **_WCSSamplesEntityBeansProject** 프로젝트를 펼친 후, **com.ibm.commerce.sample.objects**를 펼쳐서 새로 생성된 코드를 볼 수 있습니다. 패키지 내에는 BonusAccessbean이라는 새 클래스가 표시됩니다.

14. 다음 단계는 전개 코드 생성에 관한 것입니다.
코드 생성 유틸리티는 bean을 분석하여 Sun Microsystems의 EJB 스펙에 맞는지 확인하고 EJB 서버의 고유 규칙을 따랐는지 확인합니다. 또한 선택된 엔터프라이즈 bean마다 코드 생성 도구는 CMP bean용 JDBC 지속 및 finder 클래스뿐 아니라 홈 및 EJBObject(원격) 구현, 홈 및 원격 인터페이스용 구현 클래스를 생성합니다. 또한 홈 및 원격 인터페이스용 스텁(stub)뿐 아니라, IIOP를 통한 RMI 액세스에 필요한 Java ORB, 스텁(stub), 타이 클래스를 생성합니다.
CMP 엔터프라이즈 bean이 들어 있는 EJB 그룹을 선택한 경우 또는 각 CMP 엔터프라이즈 bean을 선택한 경우, 다음 항목도 생성됩니다.
    - 지속 클래스로 생성된 create table 문자열
    - 테이블과 맵되는 Persister 구현

    전개 코드를 생성하려면 다음을 수행하십시오.

    a. 엔터프라이즈 bean 패널에서 EJB 탭을 선택하고 **Bonus** 엔터프라이즈 bean을 마우스 오른쪽 버튼으로 누른 후 **전개 코드 생성**을 선택하십시오. 코드 생성에는 몇 분이 걸립니다.

15. Bonus 엔터프라이즈 bean을 테스트하기 전에 모든 WebSphere Commerce EJB 그룹과 새 WCSSamplesEntityBeans EJB 그룹이 들어 있는 새 EJB 서버를 작성해야 합니다. 트랜잭션 범위를 유지하기 위해 이러한 그룹은 동일한 서버에서 실행되어야 합니다. 일단 새 서버를 작성하면 해당 서버를 시작해야 합니다.

    새 EJB 서버를 작성하고 시작하려면 다음을 수행하십시오.

    a. 모든 EJB 그룹이 접혀 있는지 확인하십시오.

    b. 엔터프라이즈 bean 분할창에서 모든 WebSphere Commerce EJB 그룹과 새 EJB 그룹을 선택하십시오(즉, WCS로 시작하는 모든 EJB 그룹 선택).

    c. 이러한 EJB 그룹을 선택하고 마우스 오른쪽 버튼을 누른 후 **추가 > 서버 구성**을 선택하십시오.

    EJB 서버 구성 창이 열립니다(다소 시간이 걸립니다).

    d. 새로 작성된 EJB 서버(예: **EJB 서버(server2)**)를 마우스 오른쪽 버튼으로 누른 후 **특성**을 선택하여 다음을 입력하십시오.

속성	DB2 **DB2 값**	Oracle **Oracle 값**
데이터 소스	WebSphere Commerce DB2 DataSource *instance_name*	WebSphere Commerce Oracle DataSource *instance_name*
연결 유형	<DataSource>	<DataSource>
사용자 이름	*cs_db_user_name*	*cs_db_user_name*
암호	*wcs_db_password*	*wcs_db_password*
트랜잭션 시간 종료	1200	1200
트랜잭션 비활성 시간 종료	600000	600000

    확인을 누르십시오.

    주: 데이터 소스의 값은 *instance_name*.xml 파일에 지정된 데이터 소스 값과 일치해야 합니다.

개발 시스템의 하드웨어에 따라(예: 프로세서 속도) **트랜잭션 시간 종료** 및 **트랜잭션 비활동** EJB 서버 특성에 대한 값을 증가해야 할 수 있습니다.

e. WebSphere Test Environment를 실행 중이면 379 페이지의 부록 A 『WebSphere Test Environment 시작 및 중지』에 설명된 대로 PNS 및 기타 EJB 서버도 함께 중지하십시오.

f. 379 페이지의 『PNS 시작 및 중지』에 설명된 대로 PNS를 시작하십시오.

g. 380 페이지의 『EJB 서버 시작 및 중지』에 설명된 대로 새 EJB 서버를 시작하십시오.

16. 일단 EJB 서버가 시작되면 테스트 클라이언트를 시작할 수 있습니다. 테스트 클라이언트를 사용하여 데이터베이스에 새 레코드를 작성합니다. 테스트 클라이언트를 시작하여 레코드를 작성하려면 다음을 수행하십시오.

a. EJB 탭을 선택하고 **Bonus** 엔터프라이즈 bean을 마우스 오른쪽 버튼으로 누른 후 **테스트 클라이언트 실행**을 선택하십시오.

b. EJB 찾아보기 창에서 **찾아보기**를 누르십시오.
Bonus 창이 열립니다.

c. **create(Long, Integer)**를 누르십시오. 정보 분할창에서 다음을 채우십시오.

속성	값
Long	-1000
Integer	100

EJB 테스트 클라이언트 창에서 호출 아이콘을 누르십시오.

주: 첫 번째 속성은 등록된 사용자의 memberId와 일치해야 합니다. USERS 테이블을 살펴보면 memberId를 판별할 수 있습니다.

17. 데이터베이스를 직접 조회하여 데이터베이스 레코드가 제대로 작성되었는지 확인하십시오.

DB2 DB2 데이터베이스를 사용 중인 경우, 다음을 수행하십시오.

a. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.

b. **대화식** 탭을 선택하십시오.

c. `connect to` *`wcs_database_name`*을 입력한 후 실행 아이콘을 누르십시오.

d. SELECT * FROM Bonus를 입력한 후 실행 아이콘을 누르십시오.
다음 값이 리턴되어야 합니다.

열	값
MEMBERID	-1000
BONUSPOINT	100

위의 레코드는 EJB 테스트 클라이언트에 의해 작성됩니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하십시오.

a. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle - OraHome81 > Application Development > SQL Plus**).
b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
c. **암호** 필드에 Oracle 암호를 입력하십시오.
d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
e. SQL Plus 창에서 다음을 입력하십시오.

```
select * from BONUS;
```

그리고 Enter를 눌러 SQL 문을 실행하십시오.
다음 값이 리턴되어야 합니다.

열	값
MEMBERID	-1000
BONUSPOINT	100

18. 다음과 같이 테스트 클라이언트를 사용하여 Bonus 엔터프라이즈 bean이 데이터베이스 레코드에 정상적으로 액세스하는지 검증하십시오.
a. Bonus 창에서 **홈** 탭을 선택하십시오.
b. 메소드 패널에서 **FindByMemberId(Long)**를 누르십시오.
c. **Long** 필드에 -1000을 입력하고 호출 아이콘을 누르십시오.
d. **원격** 탭을 선택한 상태에서 **메소드**를 펼치고 **getBonusPoint**를 누른 후 호출 아이콘을 누르십시오.
정보 패널에서는 정수 결과 100을 보여줍니다.

e. Bonus 및 EJB 테스트 클라이언트 창을 종료하십시오.

보너스 엔티티 bean과 MyNewControllerCmd 통합

이전 절에서는 VisualAge for Java 내에서 생성된 테스트 클라이언트를 사용하여 Bonus 엔티티 bean을 테스트했습니다. 이 절에서는 MyNewControllerCmd 로직과 Bonus 엔티티 bean을 통합합니다. Java 코드가 갱신되면 Sample.jsp 템플리트는 구매자의 보너스 점수 잔고를 갱신할 수 있는 인터페이스를 작성하도록 갱신됩니다

Bonus 엔티티 bean 통합에는 다음과 같은 고급 단계가 수반됩니다.

1. MyNewTaskCmdImpl 클래스의 performExecute 메소드를 수정하여 새 보너스 점수를 계산하고 BONUS 테이블에 점수를 저장하십시오.
2. getResources 메소드를 MyNewControllerCmdImpl 클래스에 추가하여 명령이 사용하는 자원 목록을 리턴하십시오. 이 메소드는 액세스 제어 목적으로 포함됩니다.
3. 새 자원에 대한 새 액세스 제어 정책을 작성하십시오.
4. DataBeanSampleBean을 수정하여 Bonus 엔티티 bean의 액세스 bean에서 확장하도록 하십시오. 액세스 bean에서 데이터 bean을 확장하면 데이터 bean은 액세스 bean의 모든 속성을 상속하게 됩니다.
5. DataBeanSampleBean에서 메소드를 수정하십시오.
6. WebSphere Test Environment에서 Servlet 엔진의 클래스 경로를 수정하여 새 _WCSSamplesEntityBeansProject를 포함시키십시오.
7. Sample.jsp 템플리트를 수정하여 사용자가 보너스 점수를 입력하고 결과를 표시할 수 있게 하십시오.

보너스 점수 계산을 위한 MyNewTaskCmdImpl 수정: MyNewTaskCmdImpl은 Bonus 엔티티 bean과 MyNewControllerCmd의 통합 점수로 사용됩니다 (MyNewControllerCmd가 MyNewTaskCmd를 호출하므로).

보너스 점수 계산을 수행하도록 MyNewTaskCmdImpl을 수정하려면 다음을 수행하십시오.

1. VisualAge for Java 워크벤치 창에서 **_WCSSamples** 프로젝트를 펼치십시오.

2. **com.ibm.commerce.sample.commands** 패키지를 펼친 후 **MyNewTaskCmdImpl** 클래스를 선택하여 해당 소스 코드를 보십시오.
3. 다음 import 문의 설명 표시를 삭제하십시오.

   ```
   import com.ibm.commerce.sample.objects.*;
   ```

   작업을 저장하십시오(**Ctrl + S**).
4. **MyNewTaskCmdImpl** 클래스의 **performExecute** 메소드를 선택하십시오.
5. performExecute 메소드의 소스 코드에서 Section 3의 설명 표시를 제거하여 메소드에 다음 코드가 삽입됩니다.

```
// 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");
}
```

작업을 저장하십시오.

getResources 메소드를 MyNewControllerCmdImpl 클래스에 추가: 이 절에서, 새 getResources 메소드를 MyNewControllerCmdImpl 클래스에 추가합니다. 이 메소드는 처리 중 명령이 사용하는 자원 목록을 리턴합니다. 이 메소드는 자원 레벨 액세스 제어를 위해 필요합니다.

getResources 메소드를 추가하려면, 다음을 수행하십시오.

1. **프로젝트** 탭을 선택한 상태에서 **_WCSSamples** 프로젝트를 펼치십시오.
2. **com.ibm.commerce.sample.commands** 패키지를 펼치십시오.
3. **MyNewControllerCmdImpl** 클래스를 선택하여 해당 소스 코드를 보십시오.
4. 소스 코드에서 액세스 제어 절의 설명 표시를 제거하십시오. 이 절은 다음과 같은 코드 단편에 표시된 것처럼 표시됩니다.

```
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);

    }
```

작업을 저장하십시오(Ctrl+S).

앞의 코드 부분을 저장하면 VisualAge for Java에서는 이 특정 보기로부터 필드 및 액세서를 분리합니다. 지금 이들 필드와 액세서는 MyNewControllerCmdImpl 클래스 아래에 표시되며 메소드는 M으로 표시되어 있다는 점에 유의하십시오.

주: 학습의 단순성을 위해 자원 오브젝트가 getResources 메소드에 작성되었습니다. 실제 응용프로그램에서는 validateParameters 메소드에 자원 오브젝트를 작성하고 그것을 인스턴스 변수로 저장하는 방식을 선호합니다. 이 같은 이유로 getResources 및 performExecute 메소드에 의해 오브젝트가 재사용될 수 있습니다.

새 자원에 대한 액세스 제어 정책 설정: 견본 액세스 제어 정책이 제공됩니다. 이 정책은 다음과 같은 액세스 제어 오브젝트를 작성합니다.

조치 작성되는 조치는 com.ibm.commerce.sample.commands.MyNewControllerCmd 입니다

조치 그룹

작성되는 조치 그룹은 MyNewControllerCmdActionGroup입니다. 이 조치 그룹에는 하나의 조치만이 들어 있습니다. com.ibm.commerce.sample.commands.MyNewControllerCmd

자원 카테고리

작성되는 자원 카테고리는 com.ibm.commerce.sample.objects.BonusResourceCategory입니다. 이 자원 카테고리는 Bonus 엔티티 bean용입니다.

자원 그룹

작성된 자원 그룹은 BonusResourceGroup입니다. 이 자원 그룹에는 선행 자원 카테고리만이 들어 있습니다.

정책 작성되는 정책은 AllUsersUpdateBonusResourceGroup입니다. 이 정책은 사용자가 보너스 오브젝트의 "소유자"인 경우에만 Bonus bean에서 MyNewControllerCmd 조치를 수행할 수 있게 합니다. 예를 들어 사용자가 wcsadmin 사용자로 로그온하는 경우, 사용자는 wcsadmin에 대한 보너스 포인트만을 수정할 수 있습니다.

AllUsersUpdateBonusResourceGroup 정책 설정은 다음 단계와 관련됩니다.

1. acpload 명령을 사용하여 SampleACPolicy.xml 파일 로드
2. acpnlsload 명령을 사용하여 SampleACPolicy_***locale***.xml 설명 로드
3. 정책 레지스트리를 최신 정보로 고침. 이 단계는 액세스 제어 정책이 로드될 때 Servlet 엔진이 실행 중인 경우에만 필요합니다.

AllUsersUpdateBonusResourceGroup 정책을 설정하려면 다음을 수행하십시오.

1. 명령 프롬프트에서
   drive:\WebSphere\CommerceServerDev\bin
   다음 디렉토리로 전환하십시오.
2. SampleACPolicy.xml 파일을 로드하려면 다음 양식으로 acpload 명령을 발행해야 합니다.

   ```
   acpload db_name db_user db_password inputXMLFile
   ```

   여기서

   - *db_name*은 데이터베이스의 이름입니다.
   - *db_user*는 데이터베이스 사용자 이름입니다.
   - *db_password*는 데이터베이스 암호입니다.
   - *inputXMLFile*은 정책을 포함하는 XML 파일의 이름입니다.

   예를 들어 다음과 같은 명령을 발행할 수 있습니다.

   ```
   acpload VAJ_Demo user password SampleACPolicy.xml
   ```
3. 정책을 로드하려면 다음 양식으로 acpnlsload 명령을 발행해야 합니다.

   ```
   acpnlsload db_name db_user db_password inputXMLFile
   ```

   예를 들어 다음과 같은 명령을 발행할 수 있습니다.

   ```
   acpnlsload VAJ_Demo user password SampleACPolicy_en_US.xml
   ```

4. WebSphere Test Environment용 Servlet 엔진이 현재 실행 중인 경우, 정책 레지스트리를 최신 정보로 고치려면 이 엔진을 중지한 다음 다시 시작하십시오.

보너스 점수 관련 DataBeanSampleBean 수정: 이 절에서는 다음을 수행하여 BonusAccessBean을 확장하도록 DataBeanSampleBean을 수정합니다.

1. 워크벤치에서 **com.ibm.commerce.sample.databeans** 패키지를 펼치십시오.
2. 다음을 수행하여 데이터 bean에 새 필드를 추가하십시오.
   a. **DataBeanSampleBean** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오. 필드 작성 SmartGuide를 사용하여 다음 단계에 설명된 대로 새 필드를 작성하십시오. 필드 작성에 대한 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.
   b. 다음 값을 사용하여 데이터 bean에 새 필드를 추가하십시오.

속성 이름	값
필드 이름	task_output_oldBonusPoint
필드 유형	String
초기값	공백으로 두십시오.
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

   완료를 누르십시오.
3. **DataBeanSampleBean** 클래스를 눌러 해당 소스 코드를 보십시오. 소스 코드에서 다음 import 문의 설명 표시를 제거하십시오.

   ```
   import com.ibm.commerce.sample.objects.*;
   ```

   작업을 저장하십시오.
4. DataBeanSampleBean 클래스의 소스 코드 내에서 Section 1의 설명 표시를 지운 후 Section 2를 설명으로 만드십시오. BonusAccessBean에서 확장할 bean이 변경되고 코드가 다음과 같이 표시됩니다.

```
/// 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 {
*/

//
//////////////////////////////////////////////////////////////
```

작업을 저장하십시오.

5. **setTask_output_userId(String)** 메소드를 선택하여 해당 소스 코드를 보십시오. 다음 코드 행을 찾으십시오.

```
public void setTask_output_userId(java.lang.String newTask_output_userId) {
       task_output_userId = newTask_output_userId;
```

앞의 행 다음에 다음 코드를 입력하여 새 BonusAccessBean의 인스턴스를 생성하십시오.

```
//////////////////////////////////////
// Section A : instantiate BonusAccessBean

if (task_output_userId != null)
     this.setInit_argMemberId(newTask_output_userId);

//////////////////////////////////////
```

작업을 저장하십시오.

6. **populate()** 메소드를 선택하여 해당 소스 코드를 보십시오. BonusAccessBean의 인스턴스를 생성하기 위해 Section 2의 설명 표시를 제거하여 메소드에 다음 코드가 삽입됩니다.

```
setTask_output_oldBonusPoint(getRequestProperties().getString(
      "task_output_oldBonusPoint"));
```

클래스 경로 수정: WebSphere Test Environment에서 수정된 명령을 테스트하기 전에 새 Bonus 엔티티 bean용으로 작성된 새 프로젝트를 포함하도록 클래스 경로를 수정해야 합니다. 이 클래스 경로를 수정하려면 다음을 수행하십시오.

1. VisualAge for Java의 **작업 영역** 메뉴에서 **도구 > WebSphere Test Environment**를 선택하십시오.
   WebSphere Test Environment 제어 센터가 열립니다.
2. **Servlet 엔진**을 누르십시오.
3. Servlet 엔진이 실행 중이면 **Servlet 엔진 중지**를 누른 다음 **클래스 경로 편집**을 누르십시오.
4. **모두 선택**을 누른 다음 **확인**을 누르십시오.

보너스 점수 관련 Sample.jsp 템플리트 수정: 표시 템플리트를 수정하려면 다음을 수행하십시오.

1. 텍스트 편집기에서 Sample.jsp 및 Sample_All.jsp 파일을 여십시오.
2. <!-- SECTION 4 -->와 <!-- END OF SECTION 4 --> 표시기 사이의 Section 4를 Sample_all.jsp에서 Sample.jsp로 복사하십시오. 다음 코드가 JSP 템플리트로 삽입됩니다.

```
<!-- 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 -->
```

Sample.jsp 파일을 저장하십시오.

3. 새 Bonus bean은 액세스 제어 하에서 보호되고 사용자는 소유한 bean에 대해 MyNewControllerCmd 조치만 수행할 수 있으므로 사용자는 로그인해야 합니다. 이 같은 이유로 사용자는 견본 상점에서 로그인 기능을 사용하여 해당 사용자에게 로그인을 허용합니다. 이렇게 하려면 Sample.jsp 파일을 상점의 디렉토리 구조에 복사해야 합니다. 일단 사용자가 상점에 로그인하면 웹 제어기가 상점 디렉토리에서 Sample.jsp 파일을 검색합니다. Sample.jsp 파일을 *vaj_drive*:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web 디렉토리에서 *vaj_drive*:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web*store_directory* 디렉토리로 복사하십시오.

주: 견본 상점의 경우 *store_directory*의 값은 InFashion입니다.

4. WebSphere Test Environment가 실행되고 있는지 확인하십시오(379 페이지의 부록 A『WebSphere Test Environment 시작 및 중지』 참조).
5. wcsadmin 사용자로 로그인하려면 다음을 수행하십시오.
   - 브라우저에 다음 URL을 입력하십시오.

     ```
     http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
          storeId=store_Id&catalogId=catalog_Id&langId=-1
     ```

   - **등록** 링크를 누르십시오.
     등록 또는 로그인 페이지가 표시됩니다.
   - **전자 우편 주소** 필드에 wcsadmin을 입력하십시오.
   - **암호** 필드에 wcsadmin 사용자의 암호를 입력한 다음 **로그인**을 누르십시오.

     주: 원래 암호는 wcsadmin이지만, contractPublish 명령이 설치 과정의 부분으로 실행될 때 변경되었습니다. 이 단계는 다음 문서에서 설명합니다.

     – Business *WebSphere Commerce Studio Business Developer Edition 설치 안내서*

     – Professional *WebSphere Commerce Studio Professional Developer Edition 설치 안내서*
6. 로그인이 완료되면 같은 브라우저에 다음 URL을 입력하십시오.

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=wcsadmin&input2=1000
   ```

   사용자가 보너스 포인트 대차 대조를 갱신할 수 있는 양식과 이전의 모든 출력 매개변수가 들어 있는 페이지가 함께 표시됩니다. 표시된 페이지는 다음 화면과 유사합니다.

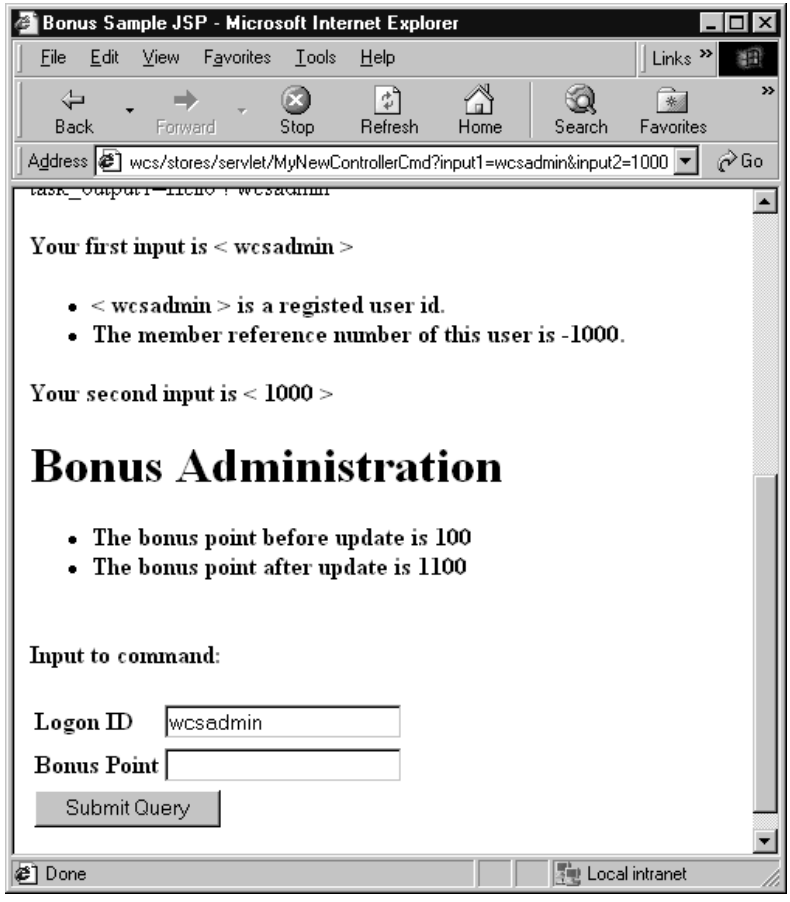


그림 40.

7. 현재 상태의 코드 버전을 작성하십시오. 버전의 이름을 `mySample 1.7`로 지정하십시오. 코드를 버전화할 때는 프로젝트를 둘 다 선택하도록 하십시오. 코드 버전화에 대한 자세한 내용은 238 페이지의 12단계를 참조하십시오.

(선택적) VisualAge for Java에서 디버거 사용

이 절에서는 사용자 코드에 중단점을 추가하고 VisualAge for Java의 디버거 구성요소를 실행하는 방법을 보여줍니다. 이 절은 디버거를 소개하기 위해 포함되었습니다. 이 강력한 기능에 대한 정보는 VisualAge for Java에 대한 온라인 도움말을 참조하십시오.

이 절은 학습에 필요한 새 코드에 대한 소개가 없으므로 선택적입니다. 대신 중단점을 작성하고, 그 중단점에서 일부 변수의 값을 검증한 다음 중단점을 제거합니다.

코드에 중단점 추가

코드에 중단점을 추가하려면 다음을 수행하십시오.

1. 다음을 수행하여 작업 영역에서 중단점 사용이 가능한지 확인하십시오.
   a. **창** 메뉴에서 **디버그**를 선택하십시오. **중단점 글로벌 사용**이 선택되어 있는지 확인하십시오.
2. **프로젝트** 탭을 선택하십시오.
3. **_WCSamples** 프로젝트를 펼치십시오.
4. **com.ibm.commerce.sample.commands** 패키지를 펼치십시오.
5. **MyNewTaskCmdImpl** 클래스를 펼치십시오.
6. 해당되는 소스 코드를 보려면 **performExecute** 메소드를 선택하십시오.
7. 소스 분할창에서 다음 코드 행의 맨 앞에 커서를 놓으십시오(누르십시오).

   ```
   setTask_output1( "Hello ! " + getTask_input1() );
   ```

   커서를 이 위치에 그대로 두십시오.
8. **편집** 메뉴에서 **중단점**을 선택하십시오.
9. 구성: 중단점 #1 창에서 **선택된 스레드 내**를 선택하고 **확인**을 누르십시오.
10. WebSphere Test Environment가 실행되고 있는지 확인하십시오(379 페이지의 부록 A『WebSphere Test Environment 시작 및 중지』 참조).
11. wcsadmin 사용자로 로그인하려면 다음을 수행하십시오.
    - 브라우저에 다음 URL을 입력하십시오.

      ```
      http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
          storeId=store_Id&catalogId=catalog_Id&langId=-1
      ```

    - **등록** 링크를 누르십시오.
      등록 또는 로그인 페이지가 표시됩니다.
    - **전자 우편 주소** 필드에 wcsadmin을 입력하십시오.

- **암호** 필드에 wcsadmin user에 대한 암호를 입력하고 **로그인**을 누르십시오.

12. 로그인 프로세스가 완료되면 다음 URL을 입력하십시오.

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
    input1=wcsadmin&input2=1000
```

13. 코드에서 중단점에 도달하면 디버거 창이 열립니다.

변수의 값 검증

이 절에서는 명령 실행 중 다양한 지점에서의 task_input1 및 task_output1 변수값을 검증하는 방법을 보여줍니다.

MyNewTaskCmdImpl의 performExecute 메소드에 있는 중단점으로 인해 디버거가 열리면 해당되는 창으로 전환한 후 다음을 수행하십시오.

1. 변수 분할창에서 **this**를 펼치십시오.
2. **task_input1**을 누르십시오.
   값 분할창에서 실행 중 이 지점에 해당 변수의 값이 표시됩니다. wcsadmin이 표시되어야 합니다.

task_output1 변수의 값이 예상대로 설정되었는지 검증하려면 performExecute 메소드를 계속 실행하는 디버거가 해당되는 변수가 설정되어 있는 코드 내의 지점에 도달하도록 해야 합니다. 이는 다음과 같이 수행할 수 있습니다.

1. 프로시저 단위 실행 기능을 사용하여 코드에서 단계별로 이동하십시오. 이는 다음 방식 중 하나로 수행할 수 있습니다.
   - **선택** 메뉴에서 **프로시저 단위 실행**을 선택하십시오.
   - F6을 누르십시오.
   - 프로시저 단위 실행 아이콘을 누르십시오.

   이 예의 목적에 따라 F6을 4번 누르십시오. 그러면 task_output1 변수를 설정하는 코드가 실행됩니다.
2. 변수 분할창에서 task_output1을 누르십시오.
   값 분할창에서 실행 동안 이 지점에 해당 변수의 값이 표시됩니다. Hello ! wcsadmin이 표시되어야 합니다.
3. 다시 시작 아이콘을 누르면 명령 실행을 완료할 수 있습니다.

중단점 제거

코드에서 중단점을 제거하려면 다음을 수행하십시오.

1. 다시 워크벤치 창으로 전환하십시오.
2. MyNewTaskCmdImpl 클래스의 performExecute 메소드에 대한 소스 코드를 볼 수 있는지 확인하십시오.
3. 소스 분할창에서 다음 코드 행을 탐색하여 이동하십시오.

   ```
   setTask_output1( "Hello ! " + getTask_input1() );
   ```

   분할창의 왼쪽 여백에 중단점을 표시하기 위한 파란색 점이 있습니다.
4. 파란색 점을 두 번 눌러서 중단점을 제거하십시오.

이제 중단점은 코드에서 제거됩니다.

WebSphere Test Environment 내에서 견본 상점과 MyNewControllerCmd 통합

이 절에서는 MyNewControllerCmd를 호출하는 견본 상점의 홈페이지에 대한 링크를 추가합니다. 통합 단계를 수행하려면 다음을 수행하십시오.

1. 텍스트 편집기를 사용하여 sidebar.jsp 파일을 여십시오. 이 파일은 다음 디렉토리에 있습니다.

   vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*\include

   여기서, *vaj_drive*는 VisualAge for Java를 설치한 드라이브이며 *store_directory*는 견본 상점의 디렉토리 이름입니다.
2. </table> 태그 앞에 다음 코드를 삽입하여 테이블에 MyNewControllerCmd에 대한 링크가 있는 또다른 행을 추가하십시오.

   ```
   <tr>
   <td>
   <a href = "MyNewControllerCmd?input1=wcsadmin&input2=1000">
        MyNewControllerCmd</a>
   </td>
   </tr>
   ```

   작업을 저장하십시오.

3. WebSphere Test Environment가 실행되고 있는지 확인하십시오.
4. 브라우저에 다음 URL을 입력하여 통합을 테스트하십시오.

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=store_Id&catalogId=catalog_Id&langId=-1
   ```

   등록 링크를 누르십시오.
   등록 또는 로그인 페이지가 표시됩니다.
5. **전자 우편 주소** 필드에 wcsadmin을 입력하십시오.
6. **암호** 필드에 wcsadmin user에 대한 암호를 입력하고 **로그인**을 누르십시오.
7. 사용자가 로그온되면 옆의 탐색 분할창에서 **MyNewControllerCmd** 링크를 누르십시오. 견본 JSP가 표시됩니다.

   주: 옆의 탐색 분할창에 새 링크가 표시되지 않으면 sidebar.jsp가 캐시될 수 있습니다. 캐시에서 이를 제거한 후 페이지를 다시 로드하십시오. 컴파일된 JSP 파일 삭제에 대한 자세한 내용은 427 페이지의 부록 C『VisualAge for Java 추가정보』를 참조하십시오. 컴파일된 JSP 파일을 삭제한 후에는 Servlet 엔진을 다시 시작해야 할 수 있습니다.

(선택적) 원격 WebSphere Commerce Server에 새 비즈니스 로직 전개

이 절에서는 원격 WebSphere Commerce Server에서 실행 중인 상점으로 새 비즈니스 로직을 전개하는 방법에 대해 설명합니다. 이 전개 단계를 시작하려면 원격 WebSphere Commerce Server에 상점(InFashion 견본 상점에 근거)을 작성해 놓아야 합니다.

개발 처리에는 대상 WebSphere Commerce Server에서 수행되는 단계 뿐만 아니라 개발 시스템에서 수행되는 단계도 포함됩니다.

새 명령 로직용 JAR 파일 작성

새 명령 및 데이터 bean 로직이 들어 있는 JAR 파일을 작성해야 합니다. JAR 파일을 작성하려면 다음을 수행하십시오.

1. 379 페이지의 부록 A『WebSphere Test Environment 시작 및 중지』에서 설명한 대로 WebSphere Test Environment를 중지하십시오.
2. 프로젝트 탭을 선택한 상태에서 **_WCSSamples** 프로젝트를 선택하십시오
3. 프로젝트를 강조표시하고 마우스 오른쪽 버튼을 누른 후 **반출**을 선택하십시오. 반출 SmartGuide가 열립니다.
4. **Jar 파일**을 선택한 후 **다음**을 누르십시오
5. Jar 파일 필드에 다음을 입력하십시오.
   drive:\WebSphere\CommerceServerDev\mytemp\wcssamples_1.jar
   여기서 *drive*는 Commerce Studio를 설치한 드라이브입니다.
6. 다음과 같이 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
bean	선택함
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
7. **완료**를 누르십시오.
8. 프롬프트가 표시되면 새 디렉토리 작성을 확인하십시오.

작성된 JAR 파일이 완료된 패키지 이름 정보를 포함하고 있지 않기 때문에 JAR 파일을 다시 패키지하는 데 다른 패키지 유틸리티(VisualAge for Java 제외)를 사용해야 합니다. 파일을 다시 패키지하려면 다음을 수행하십시오.

1. 명령창에서 다음 디렉토리로 이동하십시오.
   drive:\WebSphere\CommerceServerDev\mytemp
2. mkdir temp1을 입력하십시오.
3. cd temp1을 입력하십시오.
4. 경로를 set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio\bin;으로 설정하십시오.
   여기서, *drive*는 WebSphere Studio를 설치한 드라이브입니다.

5. `jar xvf ../wcssamples_1.jar`을 입력하십시오.
6. `jar cvf ../wcssamples.jar *`를 입력하십시오(이름에 _1이 제거되었음을 유의하십시오).

새 EJB 그룹에 대한 JAR 파일 작성

새 EJB 그룹에 대한 JAR 파일을 작성해야 합니다. 이 파일을 작성하려면 다음을 수행하십시오.

1. EJB 탭을 선택하고 **WCSSamplesEntityBeans** EJB 그룹을 마우스 오른쪽 버튼으로 누르기한 후, **반출 > EJB 1.1 JAR**를 선택하십시오.
   반출 EJB 1.1 JAR 파일 SmartGuide가 열립니다.
2. **JAR 파일** 필드에서 *drive*`:\WebSphere\CommerceServerDev\mytemp\sampleEntityBeans_DT.jar`를 입력하십시오
3. 다음과 같이 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
대상 데이터베이스	DB2 DB2 데이터베이스로 전개 중인 경우 NT용 **DB2 V7.1**을 선택하십시오. Oracle Oracle 데이터베이스를 전개 중인 경우 **Oracle, V8**을 선택하십시오.
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
4. **완료**를 누르십시오.

JAR 파일이 작성됩니다.

JAR 파일의 이름은 "_DT" 접미부가 있습니다. 이는 WebSphere Commerce 응용프로그램에 JAR 파일을 전개하기 전, WebSphere Application Server가 제공하는 EJB 전개 도구로 JAR 파일이 실행되어야 되어야 함을 상기시킵니다.

새 엔터프라이즈 bean용 구현 JAR 파일 작성

Bonus 엔터프라이즈 bean용 구현 코드가 들어 있는 JAR 파일을 작성해야 합니다. 이 파일을 작성하려면 다음을 수행하십시오.

1. 프로젝트 탭을 선택하고 **_WCSSamplesEntityBeansProject**를 마우스 오른쪽 버튼으로 누른 후 **반출**을 선택하십시오.
   반출 SmartGuide가 열립니다.
2. **Jar 파일**을 선택한 후 **다음**을 누르십시오
3. **JAR 파일** 필드에서
   drive:\WebSphere\CommerceServerDev\mytemp\sampleImpl_1.jar를 입력하십시오.
4. 다음과 같이 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
bean	선택함
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
5. **완료**를 누르십시오.

JAR 파일이 작성됩니다. VisualAge for Java를 종료하십시오.

작성된 JAR 파일이 완료된 패키지 이름 정보를 포함하고 있지 않기 때문에 JAR 파일을 다시 패키지하는 데 다른 패키지 유틸리티(VisualAge for Java 제외)를 사용해야 합니다. 파일을 다시 패키지하려면 다음을 수행하십시오.

1. 명령창에서 다음 디렉토리로 이동하십시오.
   drive:\WebSphere\CommerceServerDev\mytemp
2. mkdir temp2를 입력하십시오.
3. cd temp2를 입력하십시오.

4. 경로를 set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio\bin;으로 설정하십시오.
   여기서 *drive*는 WebSphere Studio가 설치된 드라이브입니다.
5. jar xvf ../wcssamples_1.jar을 입력하십시오.
6. jar cvf ../sampleImpl.jar * 를 입력하십시오(이름에 _1이 제거되었음을 유의하십시오).

대상 WebSphere Commerce Server로 JSP 파일 복사

Sample.jsp 및 갱신된 sidebar.jsp 파일을 코드 전개 중인 상점의 올바른 디렉토리에 복사해야 합니다.

갱신된 JSP 템플리트를 대상 WebSphere Commerce Server로 복사하기 전에 해당 시스템에서 원래 JSP 템플리트의 백업 사본을 만들 수 있습니다. 즉, 기존 파일의 이름을 sidebar.jsp.bak로 바꾸십시오.

이들 파일을 복사하려면 다음을 수행하십시오.

1. 개발 시스템에서 다음 디렉토리를 탐색하십시오.
   vaj_drive:\VAJava\ Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web*store_directory*
   여기서 *vaj_drive*는 VisualAge for Java를 설치한 드라이브이고,
   *store_directory*는 상점 디렉토리의 입니다.
   Sample.jsp를 복사하십시오.
2. 대상 WebSphere Commerce Server의 다음 디렉토리에 Sample.jsp를 붙여넣으십시오.

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instance_name.ear\
      wcstores.war\store_directory
   ```

   여기서, *drive*는 WebSphere Commerce가 설치된 드라이브이고
   *store_directory*는 상점용 디렉토리 이름이며, *instance_name*은 WebSphere Commerce 인스턴스의 이름입니다.
3. 개발 시스템에서 다음 디렉토리로 이동하십시오.
   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test

Environment
\hosts\default_host\default_app\web*store_directory*\include
sidebar.jsp를 복사하십시오.

4. 대상 WebSphere Commerce Server의 다음 디렉토리에 sidebar.jsp를 붙여넣으십시오.

   drive:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_*instance_name*.ear\
   wcstores.war*store_directory*\include

대상 WebSphere Commerce Server로 JSP 파일 복사

JAR 파일을 개발 시스템에서 대상 WebSphere Commerce Server의 적절한 디렉토리로 복사해야 합니다.

새 EJB 그룹을 포함하는 JAR 파일에서 수행되어야 하는 추가적인 두 개의 처리 단계가 있습니다. 먼저 파일에 대한 WebSphere Application Server에서 EJB 전개 도구를 실행해야 합니다. 그런 다음 파일에 대해 modifyIsolationLevel 명령을 실행해야 합니다. 결과적으로 대상 WebSphere Commerce Server에 JAR 파일을 복사하는 단계에서 특정 JAR 파일은 추가 단계를 대기하는 임시 디렉토리의 상점을 얻게 됩니다.

이들 파일을 복사하려면 다음을 수행하십시오.

1. 개발 시스템에서 다음
   drive:\WebSphere\CommerceServerDev\mytemp 디렉토리로 이동하고 다음 파일을 위치 지정하십시오.

   - wcssamples.jar
   - sampleImpl.jar
   - sampleEntityBeans_DT.jar

   여기서 *drive*는 WebSphere Commerce Studio, Business Developer Edition을 설치한 드라이브입니다.

   이전의 개별 파일은 대상 WebSphere Commerce Server에서 특정 디렉토리로 복사됩니다. 개별 파일이 수정된 위치에 저장될 수 있도록 다음을 주의깊게 읽으십시오.

2. 대상 WebSphere Commerce Server의 다음 디렉토리에 wcssamples.jar 파일을 복사하십시오.

   drive:\WebSphere\AppServer\InstalledApps\
   WC_Enterprise_App_*instance_name*.ear\wcstores.war\WEB-INF\lib

   여기서 *drive*는 WebSphere Commerce Business Edition을 설치한 드라이브이고, *instance_name*은 인스턴스 이름(예: demo)입니다.

3. JAR 파일을 놓을 임시 라이브러리 디렉토리를 작성하십시오. 즉, 다음 디렉토리를 작성하십시오.

   drive:\WebSphere\CommerceServer\temp\lib

4. 대상 WebSphere Commerce Server의 다음 디렉토리에 sampleImpl.jar 파일을 복사하십시오.

   drive:\WebSphere\CommerceServer\temp\lib

5. 대상 WebSphere Commerce Server의 다음 디렉토리에 sampleEntityBeans_DT.jar 파일을 복사하십시오.

   drive:\WebSphere\CommerceServer\temp

EJB 전개 도구 실행

테스트 서버 또는 프로덕션 서버 둘 중 하나에서 엔터프라이즈 bean을 실행하려면, 엔터프라이즈 bean용 전개 코드를 생성해야 합니다. WebSphere Application Server가 제공하는 Enterprise JavaBean용 전개 도구(EJB 전개 도구로도 불림)는 엔터프라이즈 bean 전개 코드를 생성하는 데 사용할 수 있는 명령행 인터페이스를 제공합니다.

이 도구를 실행하려면 다음을 수행하십시오.

1. 명령 프롬프트에서 다음 디렉토리로 이동하십시오.

   drive:\WebSphere\CommerceServer\temp

2. 다음 명령을 입력해서 시스템 경로에 도구를 임시로 추가하십시오.

   PATH=*drive*:\WebSphere\AppServer\deploytool;%PATH%

3. 다음과 같이 ejbdeploy 명령을 입력하십시오.

   ejbdeploy *EJBGroupJARFile WorkingDir OutputJARFile* -nowarn -keep -35 -cp
   ClassPathOfDepJARFiles

여기서

- *EJBGroupJARFile*은 사용자가 전개 코드를 생성하려는 엔터프라이즈 bean JAR 파일의 완전한 이름입니다. 예를 들어 *drive*:\WebSphere\CommerceServer\temp\sampleEntityBeans_DT.jar입니다.
- *WorkingDir*은 코드 생성에 필요한 임시 파일이 저장된 디렉토리의 이름입니다.
- *OutputJARFile*은 출력된 JAR 파일의 완전한 이름입니다. 이 경우에는 *drive*:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar를 입력하십시오.
- -nowarn은 경고 및 정보 메시지를 억제하기 위한 선택적 매개변수입니다.
- -keep은 ejbdeploy 명령을 실행한 후, 작업 디렉토리를 유지하기 위한 선택적 매개변수입니다.
- -35는 WebSphere Application Server 버전 3.5와 함께 제공되는 EJB 전개 도구에 사용되는 CMP 엔티티 bean용 하향식 맵핑 규칙을 사용하는 필수 매개변수입니다.
- -cp*ClassPathOfDepJARFiles*는 종속 JAR 파일의 클래스 경로입니다. 이 경우 다음을 입력하십시오.

  ```
  "drive:\WebSphere\CommerceServer\temp\lib\sampleImpl.jar;
  drive:\WebSphere\AppServer\installedApps\
     WC_Enterprise_App_instanceName.ear\lib\wcsejbimpl.jar"
  ```

  주: 클래스 경로 값은 큰 따옴표(“”)로 묶어야 합니다.

Bonus bean에 대한 트랜잭션 분리 레벨 수정

이 단계에서 Bonus bean의 트랜잭션 분리 레벨을 수정하려면 modifyIsolationLevel 명령을 사용하십시오. 또한 이 도구는 bean의 분리 레벨을 특정 유형의 데이터베이스에 대한 필수 레벨로 설정합니다.

modifyIsolationLevel 명령을 실행하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server에서 명령창을 여십시오.

2. 다음 디렉토리로 이동하십시오.

   drive:\WebSphere\CommerceServer\bin

3. 다음과 같은 일반 구문의 modifyIsolationLevel 명령을 발행해야 합니다.

   ```
   modifyIsolationLevel -jarFile jar_file_name.jar
      -logFile log_file_name -dbType db_type
   ```

   여기서

   - *jar_file_name.jar*은 사용자에 맞게 정의된 코드를 포함하는 JAR 파일의 이름입니다.
   - *log_file_name*은 정보를 기록해야 하는 완전한 파일 이름입니다.
   - *db_type*은 사용 중인 데이터베이스의 유형입니다. DB2 또는 ORACLE을 입력하십시오.

   다음은 모든 값이 지정된 modifyIsolationLevel 명령의 예입니다.

   ```
   modifyIsolationLevel -jarFile
      D:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar
      -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
      -dbType DB2
   ```

예외가 명령창에 표시되지 않은 경우 명령이 정상적으로 실행되었습니다. 완료 후, 전개된 JAR 파일에서 시간소인이 변경되었음에 유의하십시오.

주: 매개변수 이름은 대소문자가 구분됩니다. 즉, jar**F**ile은 jar**f**ile과 다릅니다. 매개변수 이름을 정확하게 입력했는지 확인하십시오.

대상 데이터베이스 갱신

새 비즈니스 로직을 WebSphere Test Environment가 아닌 다른 데이터베이스를 사용하는 대상 WebSphere Commerce Server에 전개하므로, 명령 레지스트리의 변경사항을 반영하여 대상 데이터베이스를 갱신하고 BONUS 테이블을 작성해야 합니다.

▶ DB2 DB2 데이터베이스를 사용하는 경우 다음을 수행하여 대상 데이터베이스를 갱신하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.
2. **도구** 메뉴에서 **도구 설정**을 선택하십시오.

3. **명령문 종료 문자 사용** 선택란을 선택한 후 지정된 문자가 세미콜론(;)인지 확인하십시오.
4. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 URLREG 테이블에 필수 항목을 작성하십시오.

```
connect to your_database_name;
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)
```

   여기서 *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.

   모든 판매자가 이 명령을 사용합니다(STOREENT_ID에 0 값으로 표시).
5. 스크립트 창에 다음을 입력하여 VIEWREG 테이블에 항목을 작성하십시오.

```
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)
```

   실행 아이콘을 누르십시오.
6. 스크립트 창에서 다음을 입력하여 BONUS 테이블을 작성하십시오.

```
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 (users_id) on delete cascade)
```

   실행 아이콘을 누르십시오.

앞의 단계에서는 MyNewControllerCmd 및 SampleViewTask를 명령 레지스트리에 등록하고 BONUS 테이블을 작성합니다.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 대상 데이터베이스를 갱신하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.

3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 정보를 입력하여 URLREG 테이블에서 필수 항목을 작성하십시오.

```
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);
```

   Enter를 눌러 SQL 문을 실행하십시오.
6. SQL Plus 창에 다음을 입력하여 VIEWREG 테이블에 항목을 작성하십시오.

```
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);
```

   Enter를 눌러 SQL 문을 실행하십시오.
7. SQL Plus 창에서 다음을 입력하여 BONUS 테이블을 작성하십시오.

```
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);
```

   Enter를 눌러 SQL 문을 실행하십시오.
8. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL 문을 실행하십시오.

새 자원에 대한 액세스 제어 정책 로드

학습에서 보호 가능한 자원인 새 엔터프라이즈 bean(Bonus bean)을 작성했습니다. 그에 따라 이 자원에 관련되는 액세스 제어 정책이 있습니다. 또한 모든 사용자가 실행할 수 있는 새 제어기 명령을 작성했습니다. 개발 시스템에 대해 작업하

는 동안, 그 시스템에 액세스 제어 정책 정보를 로드하였습니다. 이제는 대상 WebSphere Commerce Server에 같은 액세스 제어 정책 정보를 로드해야 합니다.

액세스 제어 정책을 설정하려면 다음을 수행하십시오.

1. 다음 CD를 CD 드라이브에 삽입하십시오.
   - Business WebSphere Commerce Business Edition, V5.4 디스크 2 CD
   - Professional WebSphere Commerce Professional Edition, V5.4 디스크 2 CD
2. 다음 디렉토리로 전환하십시오.

   CD_drive:\repository\samples\programguide\
3. 해당 디렉토리에서 다음 파일을 위치 지정하십시오.
   - SampleCmdACPolicy.xml
     이 XML 파일은 새 제어기 명령에 의해 사용되는 액세스 제어 정책을 포함합니다.
   - SampleACPolicy.xml
     이 XML 파일은 새 엔터프라이즈 bean을 작성할 때 사용되는 액세스 제어 정책을 포함합니다.
   - SampleACPolicy_*locale*.xml
     여기서 *locale*은 언어 식별자입니다. 이 XML 파일에는 액세스 제어 정책 설명이 들어 있습니다.
4. 이전의 3개 파일을 다음 *drive*:\WebSphere\CommerceServer\xml\policies\xml 디렉토리에 복사하십시오.
5. SampleCmdACPolicy.xml 파일을 로드하려면 다음 디렉토리로 전환하는 명령 프롬프트를 사용하십시오.

   drive:\WebSphere\CommerceServer\bin

   다음 양식을 갖는 acpload 명령을 발행해야 합니다.

   acpload *db_name db_user db_password inputXMLFile*

   여기서
   - *db_name*은 데이터베이스의 이름입니다.

- *db_user*는 데이터베이스 사용자 이름입니다.
- *db_password*는 데이터베이스 암호입니다.
- *inputXMLFile*은 정책을 포함하는 XML 파일의 이름입니다.

예를 들어 다음과 같은 명령을 발행할 수 있습니다.

```
acpload mall user password SampleCmdACPolicy.xml
```

6. SampleACPolicy.xml 파일을 입력 파일로 지정하는 acpload 명령을 발행하여 SampleACPolicy.xml 파일을 로드하십시오. 예를 들어 다음과 같은 명령을 발행할 수 있습니다.

   ```
   acpload mall user password SampleACPolicy.xml
   ```

7. 정책을 로드하려면 다음 양식으로 acpnlsload 명령을 발행해야 합니다.

   ```
   acpnlsload db_name db_user db_password inputXMLFile
   ```

   예를 들어 다음과 같은 명령을 발행할 수 있습니다.

   ```
   acpnlsload mall user password SampleACPolicy_en_US.xml
   ```

일반적으로 정책이 적용되도록 하려면 레지스트리를 최신 정보로 고쳐야 합니다. 이러한 경우, 엔터프라이즈 bean에 대한 전개 단계의 일부로 WebSphere Application Server에서 WebSphere Commerce Server 응용프로그램을 중지한 다음 다시 시작할 것이므로 이 단계를 수행할 필요가 없습니다. 그러한 경우가 아니면, WebSphere Commerce에서 관리 콘솔을 사용하여 레지스트리를 갱신할 수 있습니다. 관리 콘솔에 대한 추가 정보는 WebSphere Commerce 온라인 도움말을 참조하십시오.

400 iSeries 시스템에서 실행 중인 WebSphere Commerce 인스턴스를 전개하려면 액세스 제어 정책 정보를 로드하는 명령을 사용합니다. acpload 명령 대신에 LODWCSAC 명령, acpnlsload 명령 대신에 LODWCSACD 명령을 사용하십시오.

LODWCSAC 명령에 대한 구문은 다음과 같습니다.

```
LODWCSAC DATABASE(dbName) SCHEMA(schemaName)
PASSWD(instancePassword) INSTROOT('instanceRoot')
INFILE('inputFile')
```

여기서

- *dbName*은 WRKRDBDIRE 명령에 정의된 관계형 데이터베이스의 이름입니다.
- *schemaName*은 인스턴스용 데이터베이스 스키마의 이름입니다(이것은 인스턴스 이름과 동일합니다).
- *instancePassword*는 인스턴스 암호입니다.
- *instanceRoot*는 인스턴스 루트입니다. 인스턴스 루트 예는 다음과 같습니다.

  /QIBM/UserData/WebCommerce/instances/*instanceName*

- *inputFile*은 액세스 정책이 있는 입력 XML 파일의 완전한 이름입니다.

LODWCSACD 명령에 대한 구문은 다음과 같습니다.

```
LODWCSACD DATABASE(dbName) SCHEMA(schemaName)
PASSWD(instancePassword) INSTROOT('instanceRoot')
INFILE('inputFile')
```

액세스 제어 정책에 대한 XML 파일은 다음 디렉토리에 저장할 수 있습니다.

/QIBM/UserData/WebCommerce/instances/*instanceName*

또한 액세스 제어 정책에 대한 XML 파일 내에서 액세스 제어 DTD에 대해 전체 경로를 사용해야 합니다. 액세스 제어 정책에 대한 DTD는 /QIBM/ProdData/WebCommerce/xml/policies/dtd 디렉토리에 저장됩니다.

예를 들어, 학습용 액세스 제어 정책을 iSeries 시스템에서 실행 중인 Websphere Commerce 인스턴스에 전개할 경우, 학습용 액세스 제어 정책을 위해 XML 파일의 DTD 스펙을 수정해야 합니다.

```
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">
```

다음으로 변경하십시오.

```
<!DOCTYPE Policies SYSTEM "/QIBM/ProdData/WebCommerce/
xml/policies/dtd/accesscontrolpolicies.dtd">
```

WebSphere Commerce 액세스 제어 모델에 대한 추가 정보는 *WebSphere Commerce 액세스 제어 안내서*를 참조하십시오.

WebSphere Application Server에서 현재 진행 중인 엔터프라이즈 응용프로그램 반출

이 단계에서는 현재 엔터프라이즈 응용프로그램을 WebSphere Application Server에서 반출하여 응용프로그램 어셈블리 도구에서 열 수 있도록 합니다.

현재 진행 중인 엔터프라이즈 응용프로그램을 반출하려면 다음을 수행하십시오.

1. 현재 진행 중인 엔터프라이즈 응용프로그램을 반출할 디렉토리를 작성하십시오. 전개된 사용자 정의 코드가 수행하는 방식에 사용자가 만족을 확신할 때까지는 루틴 시스템 유지보수 도중 파일을 삭제할 위험이 있으므로 이를 "temp" 디렉토리라 부르지 않는다는 점에 유의하십시오. 명령 프롬프트에 이 디렉토리를 작성하려면 다음을 수행하십시오.
   a. 다음 디렉토리로 탐색하십시오.

      drive:\WebSphere\CommerceServer\

   b. 다음 명령을 입력하십시오.

      ```
      mkdir working
      ```

      이는 *drive*:\WebSphere\CommerceServer\working 디렉토리를 작성합니다.
2. WebSphere Application Server 관리 콘솔을 여십시오.
3. **WebSphere 관리 도메인**을 펼치십시오.
4. **엔터프라이즈 응용프로그램**을 펼치십시오.
5. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **데모** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후, **응용프로그램 반출**을 선택하십시오.
6. **반출 디렉토리** 필드에서 *drive*:\WebSphere\CommerceServer\working을 입력하십시오. 이것은 WC_Enterprise_App_*instanceName*.ear 파일의 모든 자원을 포함하는 전체 응용프로그램을 반출합니다(여기서 *instanceName*은 WebSphere Commerce 인스턴스의 이름입니다).
7. **확인**을 누르십시오. 응용프로그램을 반출하는 데는 몇 분이 걸립니다.

엔터프라이즈 응용프로그램용 XML 구성 정보 반출

엔터프라이즈 응용프로그램용 XML 구성 정보 또한 반출해야 합니다. 이 정보를 반출하려면 WebSphere Application Server가 제공하는 명령행 유틸리티 XMLConfig를 사용하십시오.

이러한 구성 정보를 반출하려면 다음을 수행하십시오.

1. 다음 디렉토리에서 was.export.app.xml 파일을 복사하십시오.

   drive:\WebSphere\CommerceServer\xml\config

   다음 디렉토리로 복사하십시오.

   drive:\WebSphere\CommerceServer\working

2. 텍스트 편집기의 was.export.app.xml 파일을 여십시오. 이 파일은 다음과 같이 $Enterprise_Application_Name$의 모든 발생을 대체합니다.

   WebSphere Commerce Enterprise Application - *instanceName*

   여기서, *instanceName*은 WebSphere Commerce 인스턴스의 이름입니다(예: demo). 파일을 저장하십시오.

   주: 삽입하는 값은 WebSphere 고급 관리 콘솔에 표시되는 인스턴스 정보와 일치해야 합니다.

3. 명령 프롬프트에서 다음 디렉토리로 이동하십시오.

   drive:\WebSphere\CommerceServer\working

4. 다음 명령을 입력해서 부분 반출을 수행하려면 XMLConfig 도구를 호출하십시오.

   ```
   xmlConfig -export OutputFile.xml -partial was.export.app.xml
      -adminNodeName wasHostName
   ```

   여기서 *wasHostName*은 현재 엔터프라이즈 응용프로그램을 포함하는 WebSphere Application Server의 노드 이름입니다. 추가적으로 OutputFile.xml은 이 명령 실행의 결과로 작성된 파일 이름이며 was.export.app.xml은 2단계에서 수정한 파일입니다.

현재 진행 중인 엔터프라이즈 응용프로그램의 엔터프라이즈 bean에 관한 정보를 반출한 다음, 새 스탠자에 보너스 bean을 설명하는 XML 파일을 추가해야만 합니다.

보너스 bean을 설명하는 새 스탠자를 추가하려면 다음을 수행하십시오.

1. 다음 디렉토리로 탐색하십시오.

   drive:\WebSphere\CommerceServer\working

2. 텍스트 편집기에서 OutputFile.xml 파일을 여십시오.
3. <ear-file-name> 태그의 위치를 지정하고 다음으로 값을 바꾸십시오.

   drive:\WebSphere\CommerceServer\working\
   WC_Enterprise_App_*instanceName*.ear

4. 다음 견본에 표시된 대로 Bonus bean에 대한 새 스탠자에 추가해야 합니다.

```
<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>user</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>
```

   여기서

   - *user*는 데이터베이스 사용자 이름입니다.
   - *password*는 데이터베이스 사용자의 암호입니다.

   주:

   a. 이전 예에서의 행 바꾸기는 단지 보기 쉽도록 하기 위한 것입니다.

   b. **$hostName$** 값이 현재 관리 노드 서버 이름과 일치하는지 확인하십시오. 또한 이 행에 캐리지 리턴 문자가 없는지 확인하십시오.

c. <application-server-full-name> 스펙은 두 줄 이상 간격을 넓힐 수 없습니다

d. Oracle 데이터베이스를 사용하는 경우, 데이터 소스 정보를 수정해야 합니다. 앞의 코드 부분에서 다음 행을

```
<jndi-name>jdbc/WebSphere Commerce DB2 DataSource demo</jndi-name>
```

다음 행으로 변경하십시오.

```
<jndi-name>jdbc/WebSphere Commerce Oracle DataSource demo</jndi-name>
```

e. 자신의 응용프로그램을 전개할 경우(예: 학습 이외의 경우), XML 파일에 지정된 엔터프라이즈 beans용 JNDI 이름이 WebSphere Commerce 이름은 아니지만 VisualAge for Java에서 사용되는 JNDI 이름과 일치하는지 확인해야 합니다.

5. OutputFile.xml 파일을 저장하십시오.

엔터프라이즈 응용프로그램에 새 EJB 그룹 어셈블링

이 단계에서 응용프로그램 어셈블러 도구의 엔터프라이즈 응용프로그램을 여십시오. 도구를 열어 엔터프라이즈 응용프로그램에 새 Bonus bean을 추가하려면 다음을 수행하십시오.

1. 새 Bonus bean을 반입하십시오. 새 EJB 그룹에 대한 JAR 파일은 엔터프라이즈 응용프로그램의 EJB 모듈 섹션 내에 저장됩니다.
2. Bonus bean용 클래스 경로가 구현 JAR 파일을 포함하도록 설정하십시오.
3. 구현 JAR 파일을 응용프로그램에 추가하십시오. 이 JAR 파일은 엔터프라이즈 응용프로그램의 파일 섹션 내에 저장됩니다.
4. Bonus bean에 포함된 메소드용 보안 WebSphere Application Server를 설정하십시오.

엔터프라이즈 응용프로그램에 새 EJB 그룹을 어셈블링하려면 다음을 수행하십시오.

1. 다음을 수행하여 현재 진행 중인 엔터프라이즈 응용프로그램을 백업하십시오.
   a. 명령 프롬프트에서 다음 디렉토리로 이동하십시오.

      drive:\WebSphere\CommerceServer\working

b. 다음 명령을 입력하십시오.

```
copy WC_Enterprise_App_instanceName.ear
   WC_Enterprise_App_instanceName.ear.bak
```

2. WebSphere Application Server 관리 콘솔을 여십시오.
3. **파일** 메뉴에서 **도구 > 응용프로그램 어셈블리 도구**를 선택하십시오.
4. 환영 창이 열리면 **취소**를 선택하여 해당 창을 종료하십시오.
5. 다음을 수행해서 작업을 하려면 엔터프라이즈 응용프로그램을 여십시오.
   a. **파일** 메뉴에서 **열기**를 선택하십시오.
   b. **파일 이름** 필드에 다음을 입력한 후

      ```
      drive:\WebSphere\CommerceServer\working\
         WC_Enterprise_App_instanceName.ear
      ```

      열기를 누르십시오. 다음 단계를 계속하기 전에 응용프로그램이 열릴 때까지 잠시 대기하십시오. 이 작업에는 몇 분이 걸립니다.
6. **EJB Modules**을 마우스 오른쪽 버튼으로 누른 후 **반입**을 선택하십시오.
7. **파일 이름** 필드에 다음을 입력하십시오.

   ```
   drive:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar
   ```

   열기를 누르십시오. 확인값 창에서 **확인**을 누르십시오.
8. sampleEntityBeans.jar 파일이 반입되면 **WCSSamplesEntityBeans** EJB 그룹으로 화면이동하여 이 그룹을 선택하십시오.
   이 그룹에 대한 정보는 오른쪽 분할창에 표시됩니다.
9. 새 엔터프라이즈 bean용 클래스 경로 필드에서 종속 JAR 파일을 입력하십시오. 이 경우 다음을 입력하십시오.

   ```
   lib/sampleImpl.jar lib\wcsejbimpl.jar
   ```
10. **적용**을 누르십시오.
11. sampleImpl.jar 파일을 응용프로그램에 추가하려면 다음을 수행하십시오.
    a. 노드 엔터프라이즈 응용프로그램용 **파일** 노드를 마우스 오른쪽 버튼으로 누른 후 **파일 추가**를 선택하십시오. 엔터프라이즈 응용프로그램용 **파일** 노드는 계층 구조 트리의 맨 아래 부근에 위치합니다. 엔터프라이즈 응용프

로그램 내의 구성요소에 대한 다른 파일 노드가 있지만, 전체 응용프로그램에 대한 파일 노드를 선택해야 함에 유의하십시오.

b. 파일 추가 창에서 **찾아보기**를 누르십시오.

c. *drive*:\WebSphere\CommerceServer\temp로 이동하십시오.

d. 강조표시된 디렉토리에서 **선택**을 누르십시오.

e. 파일 추가 창으로 리턴하십시오.
drive:\WebSphere\CommerceServer\temp 디렉토리의 컨텐츠가 표시된다는 사실에 유의하십시오. lib 디렉토리를 강조표시하십시오.
lib의 구성요소는 오른쪽 분할창에 표시됩니다.

f. 오른쪽 분할창에서 sampleImpl.jar 파일을 선택한 후, **추가**를 누르십시오. 그러면 파일이 선택된 파일 분할창에 표시됩니다.

g. **확인**을 누르십시오.

12. Bonus bean용 보안을 구성하려면 다음을 수행하십시오.

a. EJB 모듈 노드를 펼쳐 **WCSSamplesEntityBeans** 노드를 위치 지정하고 펼치십시오.

b. **엔티티 Bean**을 펼치십시오.

c. **Bonus**를 펼치십시오.

d. **메소드 확장자**를 누르고 오른쪽 분할창에서 다음을 수행하십시오.

1) **고급** 탭을 선택하십시오.

2) **보안 동일성**이 선택되었는지 확인하십시오.

3) 개별 메소드에 대해서 **EJB 서버의 동일성 사용**이 선택되었는지 확인하십시오.

4) **적용**을 누르십시오(수정한 경우).

e. 왼쪽 탐색 분할창에서 WCSamplesEntityBeans EJB 그룹 아래의 **보안 역할**을 마우스 오른쪽 버튼으로 눌러 **새로 만들기**를 선택한 후 다음을 수행하십시오.

1) **이름** 필드에서 WCSecurityRole를 입력한 후 **적용**을 누르십시오. 이 역할이 이미 존재하는 경우, 이 단계를 수행하지 않아도 됨을 유의하십시오.

f. 왼쪽 탐색 분할창에서, WCSamplesEntityBeans EJB 그룹 아래의 **메소드 허용**을 마우스 오른쪽 버튼으로 눌러 **새로 만들기**를 선택한 후 다음을 수행하십시오.

1) **메소드 허용 이름** 필드에 WCMethodPermission을 입력하십시오.
2) **메소드** 선택 영역에서 **추가**를 누르십시오.
   메소드 추가 창이 열립니다.
3) **sampleEntityBeans.jar**와 **Bonus**를 차례로 펼친 후 메소드의 각 홈 및 **원격** 목록을 펼치십시오.
4) Shift 키를 누른 상태에서 모든 홈 메소드를 선택한 후 **확인**을 누르십시오.
5) 메소드 선택 처리를 반복하여 원격 메소드도 추가하십시오(원격 메소드가 존재할 경우).
6) 역할 선택 영역에서 **추가**를 누르십시오. WCSecurityRole을 선택한 후 **확인**을 누르십시오.
7) 각 갱신마다 **적용**을 누르십시오.

13. **파일** 메뉴에서 **저장**을 선택하십시오.
14. 응용프로그램 어셈블러 도구를 종료하십시오.

이 단계를 완료하면 사용자의 새로운 비즈니스 로직 뿐만 아니라 이전의 모든 로직을 포함하는 새 엔터프라이즈 응용프로그램을 작성합니다. 새로 수정된 WC_Enterprise_App_*instanceName*.ear 파일에 이 모든 것이 포함됩니다.

WebSphere Application Server로 새 엔터프라이즈 응용프로그램 반입

다음은 WebSphere Application Server에 새 엔터프라이즈 응용프로그램을 반입에 관련된 상위 레벨 단계입니다.

1. WebSphere Application Server에서 현재 실행 중인 엔터프라이즈 응용프로그램을 중지 및 제거. 이들 단계는 WebSphere Application Server 관리자의 콘솔에서 수행됩니다.
2. XMLConfig 명령행 유틸리티을 사용하여 새 응용프로그램 반입

3. WebSphere Application Server 관리자의 콘솔을 최신 정보로 고치고 새 엔터프라이즈 응용프로그램 시작

각 단계는 다음 절에 자세히 설명되어 있습니다.

현재 진행 중인 엔터프라이즈 응용프로그램의 중지 및 제거

WebSphere Application Server에서 사용자가 현재 진행 중인 엔터프라이즈 응용프로그램을 중지 및 제거하려면, 다음을 수행하십시오.

1. WebSphere Application Server 관리 콘솔을 여십시오.
2. **WebSphere 관리 도메인**을 펼치십시오.
3. **노드**를 펼치십시오.
4. *nodeName*을 펼치십시오(여기서 *nodeName*은 사용자 노드의 이름입니다).
5. **응용프로그램 서버**를 펼치십시오.
6. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server - *instanceName***을 마우스 오른쪽 버튼으로 누른 후 **중지**를 선택하십시오.
7. **엔터프라이즈 응용프로그램**을 펼치십시오.
8. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어, **WebSphere Commerce Enterprise Application - 데모** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 **중지**를 선택하십시오.
9. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Enterprise Application - 데모** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 **제거**를 선택하십시오.
10. 응용프로그램의 반출 여부 표시가 프롬프트될 때, **아니오**를 선택하십시오.

XMLConfig를 사용하여 새 엔터프라이즈 응용프로그램 반입

XMLConfig 명령행 유틸리티를 사용하여 새 엔터프라이즈 응용프로그램을 반입하려면 다음을 수행하십시오.

1. 다음 디렉토리로 탐색하십시오.

   ```
   drive:\WebSphere\CommerceServer\working
   ```

2. 명령 프롬프트에서 WebSphere Application Server에 엔터프라이즈 응용프로그램을 반입하려면 다음 명령을 입력하십시오.

```
xmlConfig -import OutputFile.xml -adminNodeName was_hostname
```

여기서 *was_hostname*은 현재 진행중인 응용프로그램을 포함하는 WebSphere Application Server 노드의 이름입니다.

주: 400 iSeries에서 실행 중인 WebSphere Commerce 인스턴스를 전개한 경우, 응용프로그램 반입 후 디렉토리 권한을 수정하기 위한 추가 단계를 수행해야 합니다. 이러한 사용권한의 수정 방법에 대한 자세한 내용은 422 페이지의 『엔터프라이즈 응용프로그램 반입』을 참조하십시오.

새 엔터프라이즈 응용프로그램 시작

XMLConfig 명령행 유틸리티를 사용하여 새 엔터프라이즈 응용프로그램을 반입한 후, WebSphere Application Server 관리자 콘솔을 사용하여 최신 정보로 고쳐 새 응용프로그램을 시작할 수 있습니다.

콘솔을 최신 정보로 고치고 새 응용프로그램을 시작하려면, 다음을 수행하십시오.

1. WebSphere Application Server 관리 콘솔을 여십시오.
2. **WebSphere 관리 도메인**을 펼치십시오.
3. **노드**를 강조표시하십시오.
4. **선택된 서브트리를 최신 정보로 고침** 아이콘을 누르십시오.
5. 다음을 수행하여 WebSphere Commerce 응용프로그램을 시작하십시오.
   - **응용프로그램 서버**를 펼치십시오.
   - WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server - *instanceName***을 마우스 오른쪽 버튼으로 누른 후 **시작**을 선택하십시오.

MyNewControllerCmd 테스트

다음 단계에서는 WebSphere Application Server 환경에서 실행 중인 상점의 새 로직을 테스트하게 됩니다. MyNewControllerCmd를 테스트하려면, 다음을 수행하십시오.

1. 브라우저에 다음 URL을 입력하여 통합을 테스트하십시오.

```
http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=store_Id&catalogId=catalog_Id&langId=-1
```

여기서 *store_Id*는 상점 식별자이고 *catalog_Id*는 상점 카탈로그 식별자입니다.

2. **등록** 링크를 누르십시오.
   등록 또는 로그인 페이지가 표시됩니다.
3. **전자 우편 주소** 필드에 wcsadmin을 입력하십시오.
4. **암호** 필드에 이 사이트에 사용되는 wcsadmin ID에 대한 암호를 입력하고 **로그인**을 누르십시오.
5. 사용자가 로그온되면, 옆의 탐색 분할창에서 **MyNewControllerCmd** 링크를 누르십시오. 견본 JSP가 표시됩니다.

갱신된 sidebar.jsp 파일이 표시되지 않으면 다음을 수행하여 캐시를 지우십시오.

1. 캐시된 파일을 다음 디렉토리에서 삭제하십시오.
   drive:\WebSphere\CommerceServer\instances*instanceName*\cache
2. 관련 있는 캐시된 파일을 다음 디렉토리에서 삭제하십시오.

```
drive:\WebSphere\AppServer\temp\hostName\
WebSphere_Commerce_Server_instanceName\
WebSphere_Commerce_Enterprise_Application_-_instanceName\
   wcstores.war\storeName

drive:\WebSphere\AppServer\temp\hostName\
WebSphere_Commerce_Server_instanceName\
WebSphere_Commerce_Enterprise_Application_-_instanceName\
wcstores.war
```

3. 브라우저의 캐시를 지우십시오.

JSP 페이지 컴파일에 시간이 너무 오래 걸리면 페이지가 표시되지 않을 수 있습니다. 이 경우, 페이지를 다시 로드하십시오.

제 10 장 기존 비즈니스 로직 수정 및 확장

다음 학습에서는 기존 WebSphere Commerce 비즈니스 로직을 확장하거나 수정하는 방법을 보여줍니다.

기존 제어기 명령 확장

이 절에서는 기존 OrderProcess 제어기 명령을 확장하여 구매에 대해 누적된 총 보너스 점수가 주문 확인 페이지에 표시됩니다.

주: 이 학습의 목표는 기존 제어기 명령 수정 처리를 보여주는 것입니다. 이 학습은 쇼핑 플로우에서 주문 처리를 수정하기 위한 가장 좋은 방법을 보여주도록 설계된 것은 아닙니다. 사실 WebSphere Commerce는 쇼핑 플로우에서 주문 처리 단계를 수정하기 위해 사용할 수 있는 `ExtOrderProcess` 태스크 명령을 제공합니다.

학습을 시작하기 전에

229 페이지의 제 9 장『학습: 새 비즈니스 로직 작성』단계를 완료해야 합니다.

다음 목록에서는 OrderProcess 명령 확장과 관련된 단계를 요약합니다.

1. 사용자 정의된 코드가 저장된 새 패키지 작성. 모든 사용자 정의 코드(명령 및 데이터 bean용)는 WebSphere Commerce 코드와는 별도의 프로젝트 및 패키지에 저장되어야 함을 기억하십시오.
2. 기존 `OrderProcessCmdImpl` 명령을 확장하는 새 `OrderProcessCmdBonusImpl` 클래스 작성
3. `OrderProcessCmdBonusImpl` 클래스에 필드와 메소드 추가
4. `OrderProcessCmdBonusImpl` 클래스를 사용하기 위해 명령 레지스트리 수정

5. 새 비즈니스 로직을 표시하도록 confirmation.jsp 템플리트 수정
6. WebSphere Test Environment 내에서 새 비즈니스 로직 테스트
7. (선택적) 원격 WebSphere Commerce Server의 상점에 새 비즈니스 로직 전개

*OrderProcessCmdBonusImpl*에 대한 새 패키지 작성

OrderProcessCmdBonusImpl 명령이 저장된 새 패키지를 작성하려면 다음을 수행하십시오.

1. VisualAge for Java 워크벤치 창에서 프로젝트 탭을 선택했는지 확인하십시오.
2. **_WCSSamples** 프로젝트를 마우스 오른쪽 버튼으로 누른 후 **추가 > 패키지**를 선택하십시오.
   추가 패키지 SmartGuide가 열립니다.
3. **새 패키지 이름 작성** 라디오 버튼이 선택되어 있는지 확인하고 com.ibm.commerce.sample.order를 입력하십시오.
4. **완료**를 누르십시오.

OrderProcessCmdBonusImpl 클래스 작성

새 OrderProcessCmdBonusImpl 클래스를 작성하려면 다음을 수행하십시오.

1. com.ibm.commerce.sample.order 패키지를 마우스 오른쪽 버튼으로 누른 후 **추가 > 클래스**를 선택하십시오.
   클래스 작성 SmartGuide가 열립니다.
2. **새 클래스 작성** 라디오 버튼을 선택했는지 확인하십시오.
3. **클래스 이름** 필드에 OrderProcessCmdBonusImpl을 입력하십시오.
4. 최상위 클래스를 지정하려면 **찾아보기**를 누르고 **패턴** 필드에 com.ibm.commerce.order.commands.OrderProcessCmdImpl을 입력한 후 **확인**을 누르십시오.
5. **다음**을 누르십시오.
6. 반입되어야 하는 패키지를 지정하려면 **패키지 추가**를 누르십시오. **패턴** 필드에 다음 패키지를 입력하십시오.

- com.ibm.commerce.datatype을 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.exception을 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.order.commands를 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.order.objects를 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.ras를 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.server를 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.sample.objects를 입력한 후 **추가**를 누르십시오.
- javax.ejb를 입력한 후 **추가**를 누르십시오.
- java.io를 입력한 후 **추가**를 누르십시오.
- java.math를 입력하고 **추가**를 누른 후 **종료**를 누르십시오.

7. 클래스가 구현해야 하는 인터페이스를 지정하려면 **추가**를 누르십시오. 패턴 필드에 다음 인터페이스를 입력하십시오.
   - OrderProcessCmd를 입력하고 **추가**를 누른 후 **종료**를 누르십시오.
8. **완료**를 누르십시오.

필드 및 메소드를 *OrderProcessCmdBonusImpl*에 추가

두 필드와 performExecute 메소드를 클래스에 추가해야 합니다.

OrderProcessCmdBonusImpl 클래스에 theOrder 필드를 추가하려면 다음을 수행하십시오.

1. OrderProcessCmdBonusImpl 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오.
   필드 작성 SmartGuide가 열립니다.
2. 다음 속성을 사용하여 클래스에 필드를 추가하십시오. 새 필드를 작성하는 방법에 대한 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.

속성 이름	값
필드 이름	theOrder
필드 유형	OrderAccessBean
초기값	공백으로 두십시오.
액세스 수정자	private

속성 이름	값
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

완료를 누르십시오.

bonusPercentAmount 필드를 OrderProcessCmdBonusImpl 클래스에 추가하려면 다음을 수행하십시오.

1. OrderProcessCmdBonusImpl 클래스를 마우스 오른쪽 버튼으로 누르고 **추가 > 필드**를 다시 선택하십시오.
2. 다음 속성을 사용하여 클래스에 필드를 추가하십시오. 새 필드를 작성하는 방법에 대한 자세한 내용은 243 페이지의 『새 필드 작성』을 참조하십시오.

속성 이름	값
필드 이름	bonusPercentAmount
필드 유형	double
초기값	1000
액세스 수정자	private
기타 수정자	모두 선택되지 않은 상태로 두십시오.
getter 및 setter 메소드로 액세스	checked
Getter	public
Setter	public

완료를 누르십시오.

performExecute 메소드를 OrderProcessCmdBonusImpl 클래스에 추가하려면 다음을 수행하십시오.

1. **OrderProcessCmdBonusImpl** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오.
   메소드 작성 SmartGuide가 열립니다.
2. **새 메소드 작성** 라디오 버튼이 선택되어 있는지 확인한 후 **다음**을 누르십시오.
3. **메소드 이름** 필드에 performExecute를 입력하십시오.
4. **리턴 유형** 드롭 다운 목록에서 Void를 선택하십시오. **다음**을 누르십시오.

5. 메소드에서 발생할 수 있는 예외를 선택하기 위해 **추가**를 누르십시오. **패턴** 필드에 ECException을 입력하고 **추가**를 누른 후 **종료**를 누르십시오.
6. **완료**를 누르십시오.
   메소드가 생성되며 메소드에 대한 소스 코드가 표시됩니다.
7. 소스 코드를 수정해야 합니다. performExecute 메소드의 소스 코드에서 다음 행을 찾으십시오.

```
public void performExecute() throws
   com.ibm.commerce.exception.ECException {
```

앞의 행 다음에 다음 코드를 입력하시오.

프로그래머 안내서의 PDF 버전에서 해당 코드 부분을 잘라내어 붙여넣기 할 수 있습니다. VisualAge for Java에 있는 스크랩북 창에 코드를 우선 복사하고(추가 정보는 VisualAge for Java 온라인 도움말 참조) 코드를 검사하여 잘라내서 붙여넣기 조작 중 없어지거나 수정된 문자는 없는 지 확인하는 것이 좋습니다. 코드를 확인한 후, 코드를 대상 위치로 복사하십시오. 텍스트를 다른 편집기로 복사하면 일부 문자가 수정될 수 있음을 주의하십시오.

```
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. 작업을 저장하십시오.

*OrderProcessCmdBonusImpl*을 사용하기 위해 명령 레지스트리 수정

이 예에서는 주문 처리가 필요할 때마다 주문 처리용 새 구현 클래스를 사용하려고 합니다. 이렇게 하려면 명령 레지스트리를 갱신하여 원래의 OrderProcess 인터페이스를 새 OrderProcessCmdBonusImpl 구현 클래스와 연관시켜야 합니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 명령 레지스트리를 갱신하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.
2. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 CMDREG 테이블에 필수 항목을 작성하십시오.

```
connect to your_database_name;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0;
```

   여기서 *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.
   모든 판매자가 이 명령을 사용합니다(STOREENT_ID에 0 값으로 표시).

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 명령 레지스트리를 갱신하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.

4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 정보를 입력하여 URLREG 테이블에서 필수 항목을 작성하십시오.

   ```
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0;
   ```

   Enter를 눌러 SQL 문을 실행하십시오.
   모든 판매자가 이 명령을 사용합니다(STOREENT_ID에 0 값으로 표시).
6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

   ```
   commit;
   ```

   Enter를 눌러 SQL 문을 실행하십시오.

confirmation.jsp 템플리트 수정

주문 처리 비즈니스 처리에 추가한 새 비즈니스 로직을 표시하려면 confirmation.jsp 템플리트를 수정해야 합니다. 표시 템플리트를 수정하려면 다음을 수행하십시오.

1. 다음 디렉토리로 이동하십시오.

   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*
2. confirmation.jsp의 사본을 작성하고 이를 confirmation.jsp.bak로 이름 지정하십시오.
3. 텍스트 편집기에서 confirmation.jsp를 여십시오.
4. 기존 import 문 뒤에 다음을 추가하십시오.

   ```
   <%@ page import="com.ibm.commerce.datatype.*"   %>
   ```
5. JSP 템플리트의 다음 행 바로 뒤에

   ```
   String orderRn = jhelper.getParameter("orderId");
   ```

   다음을 추가하십시오.

   ```
   String bonus = ((TypedProperty)request.getAttribute(
      ECConstants.EC_REQUESTPROPERTIES)).getString("bonus");
   ```
6. JSP 템플리트에서 다음 절을 찾으십시오.

```
<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>
```

그리고 나서 다음을 추가하십시오.

```
</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. 작업을 저장하십시오.

WebSphere Test Environment 내에서 *OrderProcessCmdBonusImpl* 테스트

WebSphere Test Environment를 사용하여 새 비즈니스 로직을 테스트할 수 있습니다. OrderProcessCmdBonusImpl 명령을 테스트하려면, 다음을 수행하십시오.

1. 379 페이지의 부록 A『WebSphere Test Environment 시작 및 중지』에서 설명한 대로 WebSphere Test Environment를 시작하십시오.
2. 브라우저를 열고 상점에 대한 URL을 입력하십시오. 예를 들어 다음 URL을 입력하십시오.

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=10001&catalogId=10001&langId=-1
   ```

3. 상품을 선택하고 구매하십시오.
4. 상품을 구매한 후, 주문 확인에서는 주문으로 획득한 보너스 포인트 수를 표시합니다.

(선택적) 원격 WebSphere Commerce Server에 사용자 정의된 비즈니스 로직 전개

WebSphere Test Environment에서 비즈니스 로직 테스트를 완료했으며 코드에 만족한다면 원격 WebSphere Commerce Server의 상점에 코드를 전개할 수 있습니다. 이 학습에서 사용자 정의는 다음을 포함합니다.

- 새 OrderProcessCmdBonusImpl 클래스

- 갱신된 confirmation.jsp 템플리트 파일
- 갱신된 명령 레지스트리

코드 전개는 다음 단계를 포함합니다.

1. VisualAge for Java 도구를 사용하여 명령 로직용 JAR 파일 작성
2. JAR 파일 및 JSP 템플리트를 대상 WebSphere Commerce Server의 적절한 디렉토리에 복사
3. 대상 WebSphere Commerce Server의 명령 레지스트리 갱신

테스트 지불 방법에 대한 주

기본값으로 WebSphere Test Environment 내에서 실행하는 견본 상점은 테스트 지불 방법을 사용합니다. 이 테스트 지불 방법이 사용되면 Payment Manager로의 호출을 요구하지 않고 WebSphere Test Environment 내에서 쇼핑 플로우를 완료할 수 있습니다. 이 테스트 지불 방법은 구매를 완료하게만 하며, 이 지불 방법과 함께 제출된 주문을 더 이상 처리할 수 있게 하지는 않습니다. 이와 같이, 테스트 지불 방법은 WebSphere Test Environment 내에서만 사용되어야 합니다.

사용자 정의 코드를 전개 중인 상점에서 구입을 완료할 수 있는지 확인하십시오. 지불 처리는 로컬 또는 원격 Payment Manager를 사용하여 수행할 수 있습니다.

테스트 지불 방법에 대한 추가 정보는 225 페이지의 『테스트 지불 방법』을 참조하십시오.

명령 로직을 위한 JAR 파일 작성

명령 로직을 대상 WebSphere Commerce Server로 전개되게 하려면 명령 로직을 JAR 파일로 패키지해야 합니다. `OrderProcessCmdBonusImpl`이 _WCSSamples project 프로젝트에 저장되므로 해당 프로젝트에 대해 JAR 파일을 작성합니다.

JAR 파일을 작성하려면 다음을 수행하십시오.

1. 379 페이지의 부록 A 『WebSphere Test Environment 시작 및 중지』에서 설명한 대로 WebSphere Test Environment를 중지하십시오.
2. **_WCSSamples** 프로젝트를 마우스 오른쪽 버튼으로 누른 후 **반출**을 선택하십시오.
   반출 SmartGuide가 열립니다.

3. **Jar 파일**을 선택한 후 **다음**을 누르십시오
4. Jar 파일 필드에
   drive:\WebSphere\CommerceServerDev\mytemp_b\wcssamplesb_1.jar
   를 입력하십시오. 여기서 *drive*는 Commerce Studio를 설치한 드라이브입니다.
5. 다음과 같이 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
bean	선택함
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
6. **완료**를 누르십시오.

작성된 JAR 파일이 완료된 패키지 이름 정보를 포함하고 있지 않기 때문에, JAR 파일을 다시 패키지하는 데 다른 패키지 유틸리티(VisualAge for Java 제외)를 사용해야 합니다. 파일을 다시 패키지하려면 다음을 수행하십시오.

1. 명령창에서
   drive:\WebSphere\CommerceServerDev\ mytemp_b 디렉토리로 이동하십시오
2. mkdir temp1을 입력하십시오.
3. cd temp1을 입력하십시오.
4. 경로를 set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio\bin;으로 설정하십시오.
   여기서, *drive*는 WebSphere Studio가 설치된 드라이브입니다.
5. jar xvf ../wcssamplesb_1.jar를 입력하십시오.
6. jar cvf ../wcssamplesb.jar *를 입력하십시오

대상 WebSphere Commerce Server에 자원 저장

명령 로직에 대한 JAR 파일과 수정된 confirmation.jsp 템플리트는 대상 WebSphere Commerce Server에서 해당 디렉토리에 위치되어야 합니다.

원격 WebSphere Commerce Server의 해당 디렉토리에 JAR 파일을 저장하려면, 다음을 수행하십시오.

1. 개발 시스템에서 명령창을 열고
   drive:\WebSphere\CommerceServerDev\mytemp_b
   디렉토리로 이동하여 wcssamplesb.jar 파일 위치를 지정하십시오.
2. 이 파일을 대상 WebSphere Commerce Server의 다음 디렉토리에 복사하십시오.

   drive:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib

대상 WebSphere Commerce Server에서 해당 디렉토리에 confirmation.jsp 템플리트를 저장하려면 다음을 수행하십시오.

1. 개발 시스템에서 다음 디렉토리를 탐색하십시오.
   vaj_drive:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_directory*
2. confirmation.jsp 템플리트를 대상 WebSphere Commerce Server의 다음 디렉토리에 복사하십시오.

   drive:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_*instanceName*.ear\wcstores.war*storeDir*

명령 레지스트리 갱신

OrderProcessCmdBonusImpl 명령을 WebSphere Test Environment와 다른 데이터베이스를 사용하는 대상 WebSphere Commerce Server에 전개할 경우, 명령 레지스트리의 변경사항을 반영하여 대상 데이터베이스를 갱신해야 합니다.

DB2 DB2 데이터베이스를 사용하는 경우, 다음을 수행하여 대상 WebSphere Commerce Server의 데이터베이스를 갱신하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 여십시오.

2. 스크립트 탭을 선택하고 스크립트 창에 다음 정보를 입력하여 CMDREG 테이블에 필수 항목을 작성하십시오.

```
connect to your_target_database_name;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0
```

여기서, *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.

Oracle Oracle 데이터베이스를 사용하는 경우, 다음을 수행하여 명령 레지스트리를 갱신하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에 다음 정보를 입력하여 CMDREG 테이블에 필수 항목을 작성하십시오.

```
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0;
```

Enter를 눌러 SQL 문을 실행하십시오.
모든 판매자가 이 명령을 사용합니다(STOREENT_ID에 0 값으로 표시).

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL 문을 실행하십시오.

WebSphere Application Server에서 엔터프라이즈 응용프로그램 다시 시작

파일 자원을 해당 디렉토리에 위치 지정하고 명령 레지스트리를 갱신하여 엔터프라이즈 응용프로그램에 명령 로직을 추가한 후, 효력 발생을 위해 엔터프라이즈 응용프로그램을 중지하고 다시 시작해야 합니다.

엔터프라이즈 응용프로그램을 중지하고 다시 시작하려면 다음을 수행하십시오.

1. WebSphere Application Server 관리 콘솔을 여십시오.
2. **WebSphere 관리 도메인**을 펼치십시오.
3. **노드**를 펼치십시오.
4. ***nodeName***을 펼치십시오(여기서 ***nodeName***은 사용자 노드의 이름입니다).
5. **응용프로그램 서버**를 펼치십시오.
6. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server - 데모** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후, **중지**를 선택하십시오.
7. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server - 데모** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후, **시작**을 선택하십시오.

WebSphere Application Server에서 실행 중인 InFashion의 새 로직 테스트

이제 WebSphere Application Server에서 실행 중인 InFashion 상점에서 새 비즈니스 로직을 검증할 수 있습니다.

이러한 최종 검증을 수행하려면 다음을 수행하십시오.

1. WebSphere Commerce Server 인스턴스가 시작된 후 브라우저를 열고 상점 홈페이지 URL을 입력하십시오. 예를 들어 다음 URL을 입력하십시오.

   ```
   http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?
       storeId=store_Id&catalogId=catalog_Id&langId=-1
   ```

   여기서 *store_Id*는 상점 식별자이고 *catalog_Id*는 상점 카탈로그 식별자입니다.
2. 상품을 선택하고 구매하십시오.
3. 상품을 구매한 후, 주문 확인에서는 주문으로 획득한 보너스 포인트 수를 표시합니다.

기존 엔티티 bean 수정 및 기존 태스크 명령 확장

주: 이 학습의 목표는 기존 엔티티 bean을 수정하고 기존 태스크 명령을 확장하는 데 사용되는 처리를 보여주는 것입니다. 상품 가격 책정을 수정하기 위한 가장 좋은 방법을 보여주도록 설계된 것은 아닙니다. 가격 할인에 대한 정보는 *WebSphere Commerce 계산 프레임워크 안내서*를 참조하십시오.

제 9 장『학습: 새 비즈니스 로직 작성』에서 완전한 새 비즈니스 로직 세트를 작성했습니다. 여기에는 새 제어기 명령, 새 JSP 템플리트, 새 데이터베이스 테이블, 테이블 액세스용 새 엔터프라이즈 bean, 데이터 액세스 bean은 물론 대응하는 액세스 bean 작성을 포함합니다. 모든 로직은 로직이 서로 작동하여 새 제어기 명령이 실행한 JSP 템플리트를 사용하여 사용자의 보너스 점수 잔고가 갱신되는 단순화된 보너스 점수 응용프로그램을 작성합니다.

BONUS 테이블이 제 9 장『학습: 새 비즈니스 로직 작성』에서 사용된 방식은 다음 도표로 나타납니다.

"새 비즈니스 로직 작성" 학습에 있는 BONUS 테이블 사용

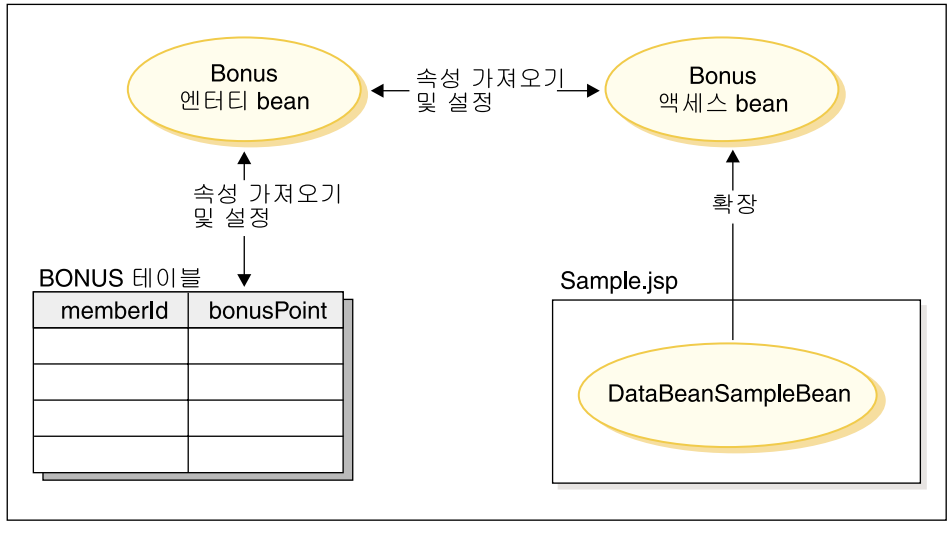


그림 41.

다음 학습에서는 BONUS 테이블이 다른 방식으로 사용됩니다. 특히 사용자 엔티티 bean은 BONUS 테이블의 BONUSPOINT 열이 실제로는 USERS 테이블의

열인 응용프로그램에 나타난 것과 같은 방식으로 사용자 정의됩니다. USERS 테이블에서 새 레코드가 작성되면 BONUS 테이블에도 자동으로 상응하는 레코드가 삽입됩니다.

이 *테이블 결합*을 작성하려면 사용자 엔티티 bean에 새 CMP 필드가 추가되어야 합니다. 이 CMP 필드는 VisualAge for Java 맵 브라우저에 있는 2차 테이블 맵 기능을 사용하여 BONUS 테이블의 BONUSPOINT 열로 맵합니다.

수정된 사용자 엔티티 bean을 쇼핑 플로우로 통합하기 위해 상품의 새 가격을 작성합니다. 이러한 새 가격은 구매자의 현재 보너스 점수 잔고를 고려합니다. GetProductContractUnitPriceCmd 태스크 명령을 확장하여 새 가격을 작성합니다. 이 명령을 확장할 경우에는 GetProductContractUnitPriceCmd 인터페이스를 확장하는 새 인터페이스를 작성하십시오. 새 인터페이스는 속성(보너스 가격용)을 추가합니다. 또한 GetContractUnitPriceCmdImpl 구현 클래스를 확장하는 새 구현 클래스를 작성합니다. 이러한 새 구현 클래스를 "GetNewContractUnitPriceCmdImpl"이라고 합니다. 새 구현 클래스는 해당 최상위 클래스의 performExecute 메소드를 호출한 후 비즈니스 로직을 추가하여 새 보너스 가격을 결정합니다.

다음 도표는 다음 학습에서 BONUS 테이블이 사용되는 방법을 표시합니다.

"사용자 엔터티 bean 사용자 정의"에 있는 BONUS 테이블 사용

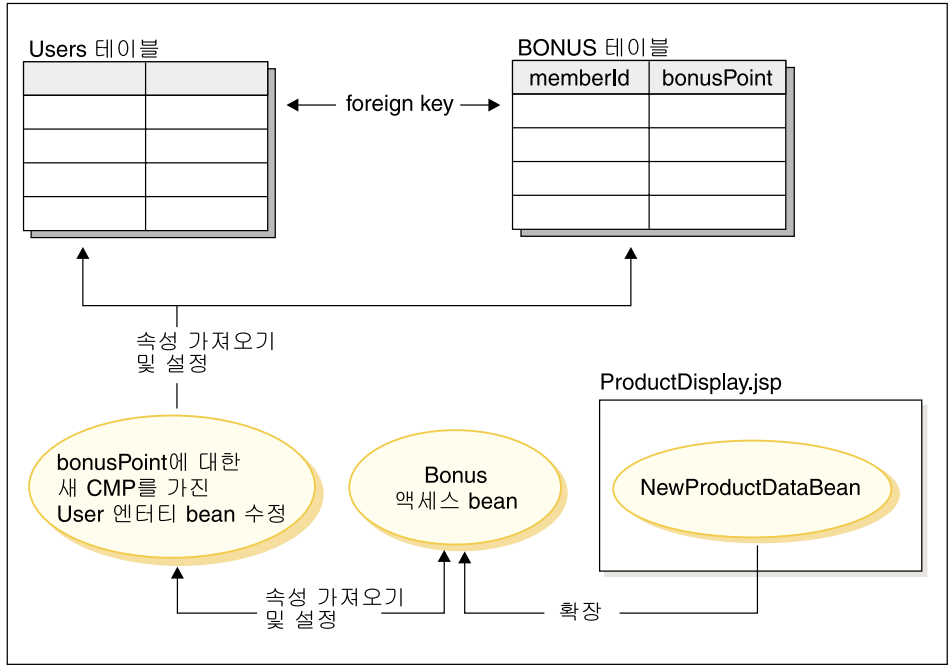


그림 42.

이 학습에서 다룰 내용은 다음과 같습니다.

1. 사용자 엔티티 bean에 새 CMP 필드 추가
2. BONUS 테이블 작성 및 대량 자료 반입
3. 새 BONUS 테이블을 포함시키기 위해 WCSUser 데이터베이스 스키마 및 테이블 맵핑 갱신
4. BONUS와 USERS 테이블 간의 foreign key 관계 작성
5. BONUS 테이블 맵 작성
6. 사용자 엔티티 bean용 전개 코드 및 액세스 bean 생성
7. 갱신 엔티티 bean의 예비 테스트를 위해 테스트 클라이언트 사용
8. 새 태스크 명령 인터페이스 및 구현 클래스 작성. 새 태스크 명령은 `GetBaseUnitProductPriceCmd`를 확장합니다. 이 명령의 가장 중요한 새 기능은 구현 클래스의 performExecute() 메소드에 있는 로직입니다. 이 메소드는 상품의 새 *보너스 가격*을 계산합니다. 이 보너스 가격은 구매자의 보너스

점수 잔고에 따라 할인되어 작성되는데 최대 할인액은 계산 가격의 20%입니다. 다음 표는 사용 중인 가격 할인 공식의 예를 표시합니다.

계산 가격	보너스 점수 잔고	최대 할인액 (계산 가격의 20%)	새 보너스 가격
$1000	100	$200	$900 보너스 점수가 최대 할인액 미만이므로 원가는 보너스 점수만큼 할인됩니다.
$1000	300	$200	$800 보너스 점수 잔고가 최대 할인액을 초과하므로 원가는 최대 할인액 $200만큼 할인됩니다.

9. ProductDataBean을 확장하는 NewProductDataBean을 작성하십시오. 이 bean에 새 메소드를 추가하여 새 보너스 가격이 상품 표시 페이지에 쉽게 사용될 수 있도록 하십시오.
10. InFashion 상점에 대한 상품 표시 JSP 템플리트를 갱신하여 보너스 가격을 표시하십시오.
11. WebSphere Test Environment 내에서 실행 중인 InFashion 상점의 비즈니스 로직을 테스트하십시오.
12. (선택적) 원격 WebSphere Commerce Server로 갱신된 비즈니스 로직을 전개하고 WebSphere Test Environment 외부에서 이를 테스트하십시오.

학습을 시작하기 전에

229 페이지의 제 9 장『학습: 새 비즈니스 로직 작성』을 완료하지 못했으면 이 학습을 시작하기 전에 230 페이지의『견본 프로젝트 준비』에서 설명된 단계를 수행해야 합니다.

사용자 엔티티 bean에 새 bonusPoint 필드 추가

이 절에서는 VisualAge for Java의 EJB 도구를 사용하여 새 CMP 필드를 엔티티 bean에 추가합니다. 새 필드는 *bonusPoint*라고하며, 결국 BONUS 테이블의 BONUSPOINT 열로 맵됩니다.

사용자 엔티티 bean에 새 CMP 필드를 추가하려면 다음을 수행하십시오.

1. 실행 중이면 Servlet 엔진, EJB 서버 및 PNS를 379 페이지의 부록 A 『WebSphere Test Environment 시작 및 중지』에서 설명한 대로 중지시키십시오.
2. 워크벤치에서 **EJB** 탭을 누르십시오.
3. **WCSUser** EJB 그룹을 펼치십시오.
4. **사용자** bean을 마우스 오른쪽 버튼으로 누른 후 **추가 > CMP 필드**를 선택하십시오.
   CMP 필드 작성 SmartGuide가 엽니다.
5. 다음 특성을 가진 새 CMP 필드를 작성하십시오.

특성	값
필드 이름	bonusPoint
필드 유형	int
초기값	0
getter 및 setter 메소드로 액세스	enable
getter 및 setter 메소드를 원격 인터페이스로 승격	enable
Getter	public
Setter	public

   완료를 누르십시오.

bonusPoint 필드가 특성 분할창에 표시됩니다. 경고가 나타날 수 있지만 엔티티 bean의 코드를 다시 생성할 때 수정됩니다.

BONUS 테이블 작성 및 대량 자료 반입

학습에서는 사용자의 보너스 점수를 기록하는 데이터베이스 테이블이 사용됩니다.

229 페이지의 제 9 장『학습: 새 비즈니스 로직 작성』을 완료했으면 이미 이 테이블에 익숙한 것입니다. 이 경우, USERS 테이블에서 각 사용자에 대한 행을 추가하려면 테이블을 갱신해야 합니다. 229 페이지의 제 9 장『학습: 새 비즈니스 로직 작성』을 완료하지 않았으면 테이블을 작성하고 대량 자료 반입해야 합니다. 이러한 각 시나리오에 대한 지시사항이 제공됩니다.

DB2 DB2 데이터베이스를 사용하고 있으며 BONUS 테이블을 *갱신*해야 하는 경우, 다음을 수행하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. **명령** 필드에 다음을 입력하십시오.

   ```
   connect to your_database_name
   ```

   여기서 *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.
3. **명령** 필드에 다음을 입력한 후, 실행 아이콘을 누르십시오.

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
   ```

   이제 BONUS 테이블이 갱신되었습니다.

DB2 DB2 데이터베이스를 사용하고 있으며 BONUS 테이블을 *작성하고 대량 자료 반입*하는 경우, 다음을 수행하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. **명령** 필드에 다음을 입력하십시오.

   ```
   connect to your_database_name
   ```

   여기서, *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.
3. **명령** 필드에 다음을 입력한 후, 실행 아이콘을 누르십시오.

```
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 (users_id) on delete cascade)
```

이제 BONUS 테이블이 작성되었습니다.

4. 대량의 테이블 자료를 반입하려면 **명령** 필드에 다음을 입력한 후, 실행 아이콘을 누르십시오.

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```

5. DB2 명령 센터를 종료하십시오.

Oracle Oracle 데이터베이스를 사용하고 있으며 BONUS 테이블을 *갱신*해야 하는 경우, 다음을 수행하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 정보를 입력하여 BONUS 테이블을 갱신하십시오.

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
   ```

   Enter를 눌러 SQL 문을 실행하십시오.
   이제 BONUS 테이블이 갱신되었습니다.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

   ```
   commit;
   ```

   Enter를 눌러 SQL 문을 실행하십시오.

Oracle Oracle 데이터베이스를 사용하고 있으며 BONUS 테이블을 *작성하고 대량 자료 반입*할 경우, 다음을 수행하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).

2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 정보를 입력하여 BONUS 테이블을 작성하십시오.

```
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);
```

   Enter를 눌러 SQL 문을 실행하십시오.
   이제 BONUS 테이블이 작성되었습니다.
6. 다음 정보를 입력하여 BONUS 테이블에 대량 자료 반입하십시오.

```
insert into BONUS (select USERS_ID, 0 from USERS);
```

7. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

   Enter를 눌러 SQL 문을 실행하십시오.

스키마 및 테이블 맵핑 갱신

다음 절에서는 새 BONUS 테이블이 있는 WCSUser 스키마를 갱신하고 새 테이블용 foreign key 관계를 작성하며 사용자 엔티티 bean과 BONUS 테이블 열의 필드 간의 테이블 맵을 작성합니다.

BONUS 테이블 스키마 작성

테이블 스키마를 작성하려면 다음을 수행하십시오.

1. **사용자** 엔티티 bean을 마우스 오른쪽 버튼으로 누른 후 **열기 > 데이터베이스 스키마**를 선택하십시오.
   스키마 브라우저 창이 열립니다.
2. **스키마** 목록에서 **WCS 사용자**를 선택하십시오.
3. **테이블** 메뉴에서 **새 테이블**을 선택하십시오.
   테이블 편집기가 열립니다.
4. **이름** 필드에 BONUS를 입력하십시오.
5. **규정자** 및 **물리적 이름** 필드는 공백으로 남겨 두십시오.

6. 데이블 열 섹션에서 **새로 만들기**를 누르십시오. 열 편집기가 열립니다. 다음과 같이 열을 작성하십시오.

속성	값
이름	memberId
물리적 이름	이 필드는 공백으로 남겨 두십시오.
유형	BIGINT
유형 정보	VapConverter
널(Null)값 허용	선택하지 않음

확인을 누르십시오.

7. 테이블의 열 목록에서 **memberId**를 선택한 후 1차 키로 사용하기 위해 >> 를 누르십시오.
8. 다음과 같은 다른 열(**새로 만들기**를 다시 누름)을 작성하십시오.

속성	값
이름	bonusPoint
물리적 이름	이 필드는 공백으로 남겨 두십시오.
유형	INTEGER
유형 정보	VapConverter
널(Null)값 허용	선택 안함

확인을 누르십시오.

9. 테이블 편집기에서 **확인**을 누르십시오.
   몇 분 후에 스키마 브라우저 창의 **테이블** 목록에 **BONUS**가 나열됩니다.
10. 확인을 위해 **테이블** 목록에 있는 **BONUS**를 눌러 두 개의 열이 **열** 목록에 표시되었는지 확인하십시오.

Foreign key 관계 작성

이 절에서는 BONUS와 USERS 테이블 간에 foreign key 관계를 설정합니다.

Foreign key 관계를 작성하려면 다음을 수행하십시오.

1. 스키마 브라우저 창에서 **WCS 사용자** 스키마를 누르십시오.
   이 스키마에 대한 테이블과 foreign key의 관계가 표시됩니다.

2. **Foreign Keys** 메뉴에서 **새 foreign key 관계**를 누르십시오.
   Foreign Key 관계 편집기가 열립니다.
3. **이름** 필드에 F_User_Bonus를 입력하십시오.
4. **데이터베이스에 제한자 있음** 선택란을 선택했는지 확인하십시오.
5. **1차 키 테이블** 드롭 다운 목록에서 USERS를 선택하십시오.
6. **Foreign key 테이블** 드롭 다운 목록에서 BONUS를 선택하십시오.
7. **Foreign Key** 헤딩 아래의 공백 필드를 눌러 드롭 다운 목록을 표시하십시오.
   이 목록에서 **memberId**를 선택한 후 **확인**을 누르십시오.
   Foreign key 관계 목록에 F_User_Bonus가 표시됩니다.
8. **스키마** 메뉴에서 **스키마 저장**을 선택한 후 **완료**를 누르십시오.
9. 스키마 브라우저를 종료하십시오.

BONUS 테이블 맵 작성

이 절에서는 BONUS 테이블의 BONUSPOINT 열과 사용자 엔티티 bean의 bonusPoint 필드 사이에 맵핑을 작성합니다.

BONUS 테이블 맵을 작성하려면 다음을 수행하십시오.

1. EJB 탭을 선택한 워크벤치가 열려 있는지 확인하십시오.
2. **EJB** 메뉴에서 **열기 > 스키마 맵**을 선택하십시오.
   맵 브라우저가 열립니다.
3. 데이터스토어 맵 목록에서 **WCS 사용자**를 선택하십시오.
4. **지속 클래스** 목록에서 다음을 수행하십시오.
   a. **구성원**을 두 번 누르십시오.
   b. **사용자**를 선택하십시오(**사용자**는 4a단계에서 구성원을 확장한 후 **구성원** 아래에만 표시됩니다).
5. **테이블 맵**에서 **새 테이블 맵 > 2차 테이블 맵 추가**를 선택하십시오.
   2차 테이블 맵 창이 열립니다.
6. **테이블** 드롭 다운 목록에서 **BONUS**를 선택하십시오.
7. Foreign key 관계 드롭 다운 목록에서 **F_User_Bonus**를 선택한 후 **확인**을 누르십시오.
   잠시 후에 BONUS(2차) 맵이 테이블 맵 목록에 표시됩니다.

8. **BONUS(2차)** 테이블 맵을 강조표시한 후 마우스 오른쪽 버튼을 눌러 **특성 맵 편집**을 선택하십시오.
   특성 맵 편집기가 열립니다.
9. bonusPoint 클래스 속성을 표시하는 행이 나올 때까지 화면이동하십시오. **bonusPoint**를 선택하십시오.
10. **맵 유형** 열에 있는 상응하는 항목을 누르고 드롭 다운 목록에서 **단순**을 선택하십시오.
11. **테이블 열** 열에 있는 상응하는 항목을 누르고 드롭 다운 목록에서 **bonusPoint**를 선택하십시오.
12. 다른 필드는 모두 변경하지 않은 상태로 두고 **확인**을 누르십시오.
    새 특성 맵이 생성되고 몇 분 후

    ```
    (a) bonusPoint (bonusPoint)
    ```

    가 맵 브라우저의 **특성 맵** 열에 표시됩니다.
13. **WCS 사용자** 데이터스토어 맵을 마우스 오른쪽 버튼으로 누르고 **데이터스토어 맵 저장**을 선택한 후 **완료**를 누르십시오.
14. 맵 브라우저를 닫으십시오.

여기서 사용자 bean에는 일부 추상 클래스가 구현되지 않았음을 나타내는 오류가 발생합니다. 이 오류는 전개 코드가 생성될 때 수정됩니다.

전개 코드 및 액세스 bean 생성

사용자 엔티티 bean에 대한 코드를 수정하였으므로 액세스 bean 및 전개 코드를 다시 생성해야 합니다. VisualAge for Java에 있는 도구는 코드 생성 단계를 단순화합니다.

이 단계를 수행하려면 다음을 수행하십시오.

1. EJB 탭을 선택한 워크벤치가 열려 있는지 확인하십시오.
2. **WCSUser** EJB 그룹을 펼치십시오.
3. **사용자** bean을 마우스 오른쪽 버튼으로 누른 후 **전개 코드 생성**을 선택하십시오.

패키지 개정판을 작성할 것인지를 묻는 메시지 창이 표시되면 **예**를 누르십시오.

코드 생성에는 몇 분이 걸립니다.

4. 전개 코드가 생성되면 **사용자** bean을 마우스 오른쪽 버튼으로 누른 후 **추가 > 액세스 Bean**을 선택하십시오. 액세스 bean 작성 SmartGuide가 열립니다.
5. 액세스 bean 특성 선택 페이지에 있는 기본값을 승인하고 **다음**을 선택하십시오.
6. 제로 인수 생성자 정의 페이지에 있는 기본값을 승인하고 **다음**을 선택하십시오.
7. 복사 헬퍼용 bean 특성 선택 및 사용자 정의 페이지에서 엔터프라이즈 bean 열의 **bonusPoint**까지 아래로 화면이동하십시오. bean에 대한 **복사 헬퍼**를 확인하고 변환기를 `com.ibm.commerce.base.objects.WCSStringConverter`로 설정하십시오.
8. **완료**를 누르십시오.
9. "코드 생성 완료"라는 메시지가 표시되면 **확인**을 누르십시오.

테스트 클라이언트를 사용하여 수정사항 테스트

테스트 클라이언트를 사용하여 새로이 사용자 정의된 사용자 엔티티 bean을 테스트할 수 있습니다. 테스트 클라이언트를 시작하여 엔티티 bean을 테스트하려면 다음을 수행하십시오.

1. 379 페이지의 『PNS 시작 및 중지』에 설명된 대로 PNS를 시작하십시오.
2. 380 페이지의 『EJB 서버 시작 및 중지』에 설명된 대로 그 위에 WCSUser EJB 그룹이 있는 EJB 서버를 시작하십시오.
3. 엔터프라이즈 bean 분할창에서 **WCSUser** EJB 그룹을 펼치십시오. **사용자** 엔티티 bean을 마우스 오른쪽 버튼으로 누른 후 **테스트 클라이언트 실행**을 선택하십시오.
4. EJB 찾아보기 창에서 **찾아보기**를 누르십시오.
5. 메소드 목록에서 **findByPrimaryKey(MemberKey)**를 선택하십시오.
6. 정보 분할창에서 **<null>**을 누르고 인수 상자에 `-1000`을 입력한 후 EJB 테스트 클라이언트 창에서 호출 아이콘을 누르십시오.

여기에 입력되는 값은 USERS 테이블의 USERS_ID 열값과 반드시 일치해야함을 주의하십시오.

이 메소드의 실행이 완료되면 -1000의 ID를 가진 사용자용 정보가 메소드 분할창의 트리 보기에 표시됩니다. 이 보기 내에는 *메소드*라는 노드가 있습니다.

7. **메소드** 노드를 펼치십시오.
8. **getBonusPoint()**를 누른 후 호출 아이콘을 누르십시오.
   이 메소드는 고객의 보너스 점수 잔고를 검색합니다. 현재 보너스 점수 잔고가 리턴됩니다.
9. **setBonusPoint(int)**를 누르고 정보 분할창에 100을 입력한 후 호출 아이콘을 누르십시오.
10. **getBonusPoint()**를 누른 후 호출 아이콘을 누르십시오.
    100 값을 리턴합니다. 이는 사용자 엔터프라이즈 bean이 데이터베이스를 성공적으로 갱신했음을 표시합니다.
11. 사용자 및 EJB 테스트 클라이언트 창을 종료하십시오.

GetNewProductContractUnitPriceCmd 인터페이스 작성

이 사용자 정의 연습의 일부로서, 고객의 보너스 점수 잔고를 근거로 새 비즈니스 로직을 작성하여 할인 가격을 계산하십시오. 새 태스크 명령이 이 계산을 수행합니다. 이 절에서는 태스크 명령에 대한 인터페이스를 작성합니다.

새 태스크 명령에 대한 인터페이스를 작성하려면 다음을 수행하십시오.

1. 프로젝트 탭이 선택된 워크벤치에서 **_WCSSamples** 프로젝트를 펼치십시오.
2. **com.ibm.commerce.sample.commands** 패키지를 마우스 오른쪽 버튼으로 누른 후 **추가 > 인터페이스**를 선택하십시오.
3. **새 인스턴스 작성**이 선택되어 있는지 확인한 후 **인터페이스 이름** 필드에 `GetNewProductContractUnitPriceCmd`를 입력하십시오.
4. **추가**를 눌러 확장되어야 할 인터페이스를 선택하고 **패턴** 필드에 `com.ibm.commerce.price.commands.GetProductContractUnitPriceCmd`를 입력한 후 **추가**를 누르고 **종료**를 누르십시오.

5. **다음**을 누르십시오.
6. 적합한 import 문을 추가하려면 **패키지 추가**를 누른 후 다음을 수행하십시오.
   a. **패턴** 필드에 com.ibm.commerce.price.utils를 입력한 후 **추가**를 누르십시오.
   b. **패턴** 필드에 com.ibm.commerce.exception을 입력한 후 **추가**를 누르십시오.
   c. **종료**를 누르십시오.
7. **인터페이스 공개**가 선택되었는지 확인하십시오.
8. **완료**를 누르십시오.
   새 인스턴스에 대한 소스 코드가 소스 분할창에 표시됩니다.
9. 다음을 수행하여 getBonusPrice() 메소드에 대한 메소드 서명을 작성하십시오.
   a. **GetNewProductContractUnitPriceCmd** 인스턴스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오.
      메소드 작성 SmartGuide가 열립니다.
   b. **새 메소드 작성**이 선택되었는지 확인한 후 **다음**을 누르십시오.
   c. 메소드 이름 필드에 getBonusPrice를 입력하십시오.
   d. 메소드 리턴 유형을 선택하기 위해 **찾아보기**를 누르십시오. **패턴** 필드에 MonetaryAmount를 입력하고 **확인**을 누른 후 **다음**을 누르십시오.
   e. 메소드에서 발생할 수 있는 예외를 선택하기 위해 **추가**를 누르십시오. **패턴** 필드에 ECSystemException을 입력하고 **추가**를 누른 후 **종료**를 누르십시오.
   f. **완료**를 누르십시오.
10. 명령에 대한 기본 구현 클래스를 지정하는 인터페이스에 필드를 추가하려면 다음을 수행하십시오.
    a. **GetNewProductContractUnitPriceCmd** 인터페이스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오.
       필드 작성 SmartGuide가 열립니다.
    b. **필드 이름** 필드에 defaultCommandClassName을 입력하십시오.
    c. **필드 유형** 드롭 다운 목록에서 **문자열**을 선택하십시오.

d. **초기값** 필드에 다음을 입력하십시오.

```
"com.ibm.commerce.sample.commands.
   GetNewContractUnitPriceCmdImpl"
```

완료를 누르십시오.

주:

1) 이 값을 입력할 경우에는 큰따옴표를 입력해야 합니다.

2) 이 새 필드를 작성하여 최상위 클래스의 필드가 숨겨질 것이라고 표시하는 경고 메시지의 경우, **예**를 눌러 계속하십시오.

11. 명령의 이름을 지정하는 인터페이스에 필드를 추가하려면 다음을 수행하십시오.

    a. **GetNewProductContractUnitPriceCmd** 인터페이스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오.
    필드 작성 SmartGuide가 열립니다.

    b. **필드 이름** 필드에 NAME을 입력하십시오.

    c. **필드 유형** 드롭 다운 목록에서 **문자열**을 선택하십시오.

    d. **초기값** 필드에 다음을 입력하십시오.

```
"com.ibm.commerce.sample.commands.
   GetNewProductContractUnitPriceCmd"
```

완료를 누르십시오.

GetNewContractUnitPriceCmdImpl 구현 클래스 작성

새 태스크 명령에 대한 구현 클래스를 작성해야 합니다. 이 구현 클래스는 새로 작성된 인터페이스를 구현하고 태스크 명령에 대한 비즈니스 로직을 포함합니다.

GetNewContractUnitPriceCmdImpl 구현 클래스를 작성하려면 다음을 수행하십시오.

1. 프로젝트 탭이 선택된 워크벤치에서 **_WCSSamples** 프로젝트를 펼치십시오.
2. **com.ibm.commerce.sample.commands** 패키지를 마우스 오른쪽 버튼으로 누른 후 **추가 > 클래스**를 선택하십시오.
   클래스 SmartGuide 작성이 열립니다.

3. **새 클래스 작성**을 선택하고 다음과 같이 클래스를 작성하십시오.
   a. **클래스 이름** 필드에 GetNewContractUnitPriceCmdImpl을 입력하십시오.
   b. 최상위 클래스를 지정하려면 **찾아보기**를 누르고 **패턴** 필드에 com.ibm.commerce.price.commands.GetContractUnitPriceCmdImpl을 입력한 후 **확인**을 누르십시오.
   c. **다음**을 누르십시오.
   d. 반입되어야 하는 패키지를 지정하려면 **패키지 추가**를 누르십시오. **패턴** 필드에 다음 패키지를 입력하십시오.
      - com.ibm.commerce.command를 입력한 후 **추가**를 누르십시오.
      - com.ibm.commerce.exception을 입력한 후 **추가**를 누르십시오.
      - com.ibm.commerce.price.commands를 입력한 후 **추가**를 누르십시오.
      - com.ibm.commerce.price.utils을 입력한 후 **추가**를 누르십시오.
      - com.ibm.commerce.ras를 입력한 후 **추가**를 누르십시오.
      - com.ibm.commerce.server를 입력한 후 **추가**를 누르십시오.
      - java.math를 입력하고 **추가**를 누른 후 **종료**를 누르십시오.
   e. 클래스가 구현해야 하는 인터페이스를 지정하려면 **추가**를 누르십시오. 패턴 필드에 다음 인터페이스를 입력하십시오.
      - GetContractSpecialPriceCmd를 입력한 후 **추가**를 누르십시오.
      - GetContractUnitPriceCmd를 입력한 후 **추가**를 누르십시오.
      - GetProductContractUnitPriceCmd를 입력한 후 **추가**를 누르십시오.
      - GetNewProductContractUnitPriceCmd를 입력하고 **추가**를 누른 후 **종료**를 누르십시오.
   f. **완료**를 누르십시오.
4. **GetNewContractUnitPriceCmdImpl** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 필드**를 선택하십시오. 필드 작성 SmartGuide가 열립니다. 다음과 같은 정보를 입력하십시오.
   a. **필드 이름** 필드에 bonusPrice를 입력하십시오.
   b. 필드 유형을 선택하기 위해 **찾아보기**를 누르십시오. 패턴 필드에 MonetaryAmount를 입력한 후 **확인**을 누르십시오.

c. **완료**를 누르십시오.

5. 새 performExecute() 메소드를 클래스에 추가하십시오. 메소드는 다음과 같은 기능을 수행합니다.
   - 최상위 클래스(GetContractUnitPriceCmdImpl)의 performExecute() 메소드 호출
   - thisClass 및 methodName 값을 설정하여 예외상황 처리 메커니즘 사용
   - StoreAccessBean 인스턴스 생성
   - UserAccessBean 인스턴스 생성
   - 원본 상품 가격 얻기
   - 적용 가능한 최대 할인 계산
   - 새 보너스 가격 계산(가격 유형은 *double*임)
   - 상점 통화 유형 얻기
   - 보너스 가격 반올림 및 정확한 가격에 이 금액을 저장

   이 방법을 추가하려면 **GetNewContractUnitPriceCmdImpl** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오. 메소드 작성 SmartGuide가 열립니다. 다음과 같은 정보를 입력하십시오.

   a. **새 메소드 작성**이 선택되었는지 확인한 후 **다음**을 누르십시오.

   b. **메소드 이름** 필드에 performExecute를 입력하십시오.

   c. **리턴 유형** 드롭 다운 목록에서 **Void**를 선택한 후 **다음**을 선택하십시오.

   d. 메소드에서 발생할 수 있는 예외를 선택하기 위해 **추가**를 누르십시오. **패턴** 필드에 ECException을 입력하고 **추가**를 누른 후 **종료**를 누르십시오.

   e. **완료**를 누르십시오.

   f. 소스 코드의 다음 행 다음에

```
public void performExecute () throws com.ibm.commerce.exception.ECException {
```

다음 코드를 추가하십시오.

프로그래머 안내서의 PDF 버전에서 해당 코드 부분을 잘라내어 붙여넣기 할 수 있습니다. VisualAge for Java(자세한 내용은 VisualAge for Java 온라인 도움말 참조)에 있는 스크랩북 창에 코드를 우선 복사하고 코드를 검사하여 잘라내기 및 붙여넣기 조작 중 없어진 문자는 없는 지 확인하는 것이 좋습니다. 코드를 확인한 후, 코드를 대상 위치로 복사하십시오. 텍스트를 다른 편집기로 복사하면 일부 문자가 수정될 수 있음을 주의하십시오.

```
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);
```

작업을 저장하십시오.

6. **GetNewContractUnitPriceCmdImpl** 클래스에서 **getBonusPrice()** 메소드를 선택하여 해당 소스 코드를 보십시오. 소스 코드에서

   ```
   return null;
   ```

   을 다음으로 변경하십시오.

   ```
   return bonusPrice;
   ```

   작업을 저장하십시오.

NewProductDataBean 데이터 bean 작성

`getCalculatedBonusPrice()` 메소드가 기존 WebSphere Commerce ProductDataBean에 추가되어야 합니다. ProductDataBean용 코드를 실제로 수정해서는 안되므로, ProductDataBean을 확장하는 새 데이터 bean을 작성한 후 새 데이터 bean에 메소드를 추가해야 합니다.

새 데이터 bean을 작성하려면 다음을 수행하십시오.

1. 프로젝트 탭이 선택된 워크벤치에서 **_WCSSamples** 프로젝트를 펼치십시오.
2. **com.ibm.commerce.sample.databeans** 패키지를 마우스 오른쪽 버튼으로 누른 후 **추가 > 클래스**를 선택하십시오. 클래스 작성 SmartGuide가 열립니다. 다음과 같은 정보를 입력하십시오.
   a. **클래스 이름** 필드에 `NewProductDataBean`을 입력하십시오.
   b. 최상위 클래스를 지정하기 위해 **찾아보기**를 누른 후 **패턴** 필드에 `com.ibm.commerce.catalog.beans.ProductDataBean`을 입력하십시오. **확인**을 누른 후 **다음**을 누르십시오.
   c. **패키지 추가**를 눌러 클래스에 적절한 import 문을 추가한 후 다음을 수행하십시오.
      - `com.ibm.commerce.beans`을 입력한 후 **추가**를 누르십시오.
      - `com.ibm.commerce.catalog.objects`를 입력한 후 **추가**를 누르십시오.
      - `com.ibm.commerce.command`를 입력한 후 **추가**를 누르십시오.

- com.ibm.commerce.datatype을 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.exception을 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.price.beans을 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.ras를 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.sample.commands를 입력한 후 **추가**를 누르십시오.
- com.ibm.commerce.server를 입력한 후 **추가**를 누르십시오.
- java.util을 입력하고 **추가**를 누른 후 **종료**를 누르십시오.

d. **완료**를 누르십시오.

3. **NewProductDataBean** 클래스를 마우스 오른쪽 버튼으로 누른 후 **추가 > 메소드**를 선택하십시오.
   메소드 추가 SmartGuide가 열립니다.
4. **새 메소드 작성**이 선택되었는지 확인한 후 **다음**을 누르십시오.
5. **메소드 이름** 필드에 getCalculatedBonusPrice를 입력하십시오.
6. 리턴 유형을 선택하기 위해 **찾아보기**를 누르십시오. **패턴** 필드에 PriceDataBean을 입력하고 **확인**을 누른 후 **다음**을 누르십시오.
7. 메소드에서 발생할 수 있는 예외를 선택하기 위해 **추가**를 누르십시오. **패턴** 필드에 ECSystemException을 입력하고 **추가**를 누른 후 **종료**를 누르십시오.
8. **완료**를 누르십시오.
9. 소스 코드에서 return null;을 다음으로 바꾸십시오.

```
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;
```

작업을 저장하십시오.

상품 표시 템플리트에 새 보너스 가격 추가

다음 단계에서는 보너스 가격을 상품 표시 템플리트에 추가하여 구매자가 사용자 정의된 가격을 볼 수 있도록 합니다. 표시 템플리트를 갱신하면 새 할인 가격이 표시됩니다.

상점 견본은 상품 표시용으로 ProductDisplay.jsp 템플리트를 사용합니다. 따라서 새 가격을 표시하려면 이 템플리트를 정보로 갱신해야 합니다.

표시 템플리트를 갱신하려면 다음을 수행하십시오.

1. 다음 디렉토리로 이동하십시오.
   vaj_drive:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*store_name*
2. ProductDisplay.jsp 파일의 사본을 작성하고 이름을 ProductDisplay.jsp.bak로 지정하십시오.
3. 텍스트 편집기에서 ProductDisplay.jsp를 여십시오.
4. <%@ page import="com.ibm.commerce.common.beans.*" %> 다음에 다음 import 문을 추가하십시오.

   ```
   <%@ page import="com.ibm.commerce.sample.commands.*" %>
   <%@ page import="com.ibm.commerce.sample.databeans.*" %>
   ```

5. 모든 ProductDataBean을 NewProductDataBean으로 바꾸십시오.
6. 다음 행 위치를 지정하십시오.

   ```
   <font class="price"><%=product.getCalculatedContractPrice()%></font>
      <br><br>
   ```

   이 행 다음에 다음 행을 삽입하여 상품에 대한 보너스 가격을 검색하고 표시하십시오.

   ```
   <font class="price"><%=product.getCalculatedBonusPrice()%> Bonus Price </font>
   <br><br>
   ```

7. 파일을 저장하십시오.

주: *WebSphere Commerce 프로그래머 안내서*의 PDF 버전에서 절을 잘라내어 표시 템플리트에 붙여넣을 경우, 처리 중 문자가 수정되지 않도록 하십시오.

엔터프라이즈 bean 확장자 테스트

이 절에서는 InFashion 견본 상점에서 상품 보기를 통해 엔터프라이즈 bean에 만든 확장자를 테스트할 수 있습니다. 새 보너스 가격이 표시됩니다.

견본을 단순화할 경우, 모든 구매자(등록된 구매자 또는 게스트 구매자)가 보너스 가격을 볼 수 있음을 유의하십시오. 보너스 점수가 없는 구매자의 경우에 보너스 가격은 정상 가격과 동일합니다.

엔터프라이즈 bean 확장자를 테스트하고 표시된 보너스 가격을 보려면 다음을 수행하십시오.

1. 다음을 수행하여 `_WCSSamples` 프로젝트가 Servlet 엔진에 대한 경로에 포함되어 있는지 검증하십시오.
   a. VisualAge for Java의 **작업 영역** 메뉴에서 **도구 > WebSphere Test Environment**를 선택하십시오.
      WebSphere Test Environment 제어 센터가 열립니다.
   b. **Servlet 엔진**을 누르십시오
   c. Servlet 엔진이 실행 중이면 **Servlet 엔진 중지**를 누른 후 **클래스 경로 편집**을 누르십시오.
   d. **_WCSSamples**가 아직 선택되지 않았으면 지금 선택하고 **확인**을 누르십시오.
2. 379 페이지의 부록 A『WebSphere Test Environment 시작 및 중지』에 설명된 대로 WebSphere Test Environment를 시작하십시오. PNS와 EJB 서버가 실행 중인 경우에는 Servlet 엔진을 시작하기만 하면 됩니다.
3. 브라우저를 열고 다음 URL을 입력하십시오.

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
       storeId=store_Id&catalogId=catalog_Id&langId=-1
   ```

4. 서비스 헤딩 아래 있는 **등록** 링크를 누른 후 처음 방문 고객 헤딩 아래 있는 **등록**을 누르십시오.

wctester@wc의 전자 우편 주소와 wctester1의 암호를 사용하여 처음 방문 고객을 등록하십시오. 다른 필드도 테스트 값으로 채우고 **제출**을 누르십시오. 브라우저는 열린 상태로 두십시오.

5. DB2 DB2 명령 센터를 열고 다음을 수행하십시오.
   a. **대화식** 탭을 누르십시오.
   b. **명령** 필드에서 다음을 수행하십시오.
      1) 다음을 입력하십시오.

         ```
         connect to your_database_name
         ```

         여기서 *your_database_name*은 WebSphere Commerce 데이터베이스 이름입니다. 실행 아이콘을 누르십시오.
      2) `select users_id from userreg where logonid = 'wctester@wc'`를 입력하고 실행 아이콘을 누르십시오.
   c. 조회 결과 탭은 4단계에서 등록한 고객에 대한 항목을 표시합니다. 다음에 고객의 USERS_ID 값을 기록하십시오. _____
   d. 새로 등록한 고객의 보너스 점수 잔고를 갱신하십시오. 대화식 탭을 누른 후 **명령** 필드에 다음을 입력하십시오.

      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id
      ```

      여기서, *users_id*는 5c단계의 값입니다. 실행 아이콘을 누르십시오.

6. Oracle 다음을 수행하여 테스트 사용자의 보너스 포인트 대차 대조를 갱신하십시오.
   a. Oracle SQL Plus 명령창을 여십시오(**시작 > 프로그램 > Oracle > Application Development > SQL Plus**).
   b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
   c. **암호** 필드에 Oracle 암호를 입력하십시오.
   d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
   e. `select users_id from userreg where logonid = 'wctester@wc';`를 입력하십시오.

f. 4단계에서 등록한 고객에 대한 항목이 표시됩니다. 다음에 고객의 USERS_ID 값을 기록하십시오. _____

g. 다음을 입력하여 새로 등록한 고객의 보너스 점수 잔고를 갱신하십시오.

```
update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id;
```

여기서, *users_id*는 6f단계의 값입니다.

h. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL 문을 실행하십시오.

7. 브라우저에서 **남성의류** 링크를 눌러 상점의 남성의류 섹션을 보십시오.
8. 기획 상품에 대한 링크를 눌러 상품 페이지를 보십시오. 페이지에는 정상 가격과 고객의 보너스 점수 잔고에 근거한 할인 가격이 표시됩니다.

주: 보너스 가격이 아닌 스택 추적이 표시되면, 캐시를 사용 안함으로 설정해야 할 경우도 있습니다. 다음과 같이 *instance_name*.xml 파일에서 CacheDaemon 구성요소 값을 false로 설정하여 이 작업을 수행할 수 있습니다.

```
<component compClassName="com.ibm.commerce.cache.daemon.CacheDaemonComponent"
                enable="false"
                name="CacheDaemon" />
```

instance_name.xml 파일에서 값을 변경하고 나면 WebSphere Test Environment에서 Servlet 엔진을 중지하고 다시 시작해야 합니다.

(선택적) 원격 WebSphere Commerce Server에 사용자 정의된 비즈니스 로직 전개

이 절에서는 수정된 엔티티 bean과 새 태스크 명령을 WebSphere Test Environment 외부에서 실행 중인 상점으로 전개하는 방법에 대해 설명합니다.

전개에는 WebSphere Commerce 공용 엔터프라이즈 bean과 명령 및 데이터 bean 로직용 JAR 파일 작성, JAR 파일을 대상 서버에 있는 해당 디렉토리에 전개, WebSphere Commerce 인스턴스 중지, 클래스 경로 수정, XMLConfig 유틸리티를 사용하여 엔터프라이즈 bean 전개와 인스턴스 다시 시작이 있습니다.

새 가격 명령용 JAR 파일 작성

_WCSSamples 프로젝트에 대한 JAR 파일을 작성하여 새 태스크 명령이 전개되도록 해야 합니다. 이 JAR 파일을 작성하려면 개발 시스템에서 다음을 수행하십시오.

1. 379 페이지의 부록 A『WebSphere Test Environment 시작 및 중지』에서 설명한 대로 WebSphere Test Environment를 중지하십시오.
2. 프로젝트 탭을 선택한 상태에서 **_WCSSamples** 프로젝트를 선택하십시오
3. 프로젝트가 강조표시된 상태에서 마우스 오른쪽 버튼을 누르고 **반출**을 선택하십시오.
   반출 SmartGuide가 열립니다.
4. **Jar 파일**을 선택한 후 **다음**을 누르십시오
5. **Jar 파일** 필드에 다음을 입력하십시오.
   drive:\WebSphere\CommerceServerDev\mytemp_c\wcssamplesc_1.jar
   여기서 *drive*는 WebSphere Commerce를 설치한 드라이브입니다.
6. 다음과 같이 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
bean	선택함
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
7. **완료**를 누르십시오.

작성된 JAR 파일이 완료된 패키지 이름 정보를 포함하고 있지 않기 때문에, JAR 파일을 다시 패키지하는 데 다른 패키지 유틸리티(VisualAge for Java 제외)를 사용해야 합니다. 파일을 다시 패키지하려면 다음을 수행하십시오.

1. 명령창에서 다음 디렉토리로 이동하십시오.
   drive:\WebSphere\CommerceServerDev\mytemp_c
2. mkdir temp3을 입력하십시오.
3. cd temp3을 입력하십시오.

4. 경로를 set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio\bin;으로 설정하십시오.
   여기서, *drive*는 WebSphere Studio가 설치된 드라이브입니다.
5. jar xvf ../wcssamplesc_1.jar를 입력하십시오.
6. jar cvf ../wcssamplesc.jar * 를 입력하십시오(이름에서 _1이 제거되었음을 유의하십시오).

WCSUser EJB 그룹에 대한 JAR 파일 작성

수정된 엔터프라이즈 bean이 있는 EJB 그룹용으로 EJB 1.1 Export JAR 파일을 작성해야 합니다. 이와 같이 JAR 파일을 작성할 경우 다음 그룹을 선택하십시오.

- WCSUser

WCSUser EJB 그룹에 대한 JAR 파일을 작성하려면 다음을 수행하십시오.

1. EJB 탭을 선택하여 **WCSUser** EJB 그룹을 강조표시하십시오.
2. **WCSUser** EJB 그룹을 마우스 오른쪽 버튼으로 누른 후 **반출 > EJB 1.1 JAR**를 선택하십시오.
   EJB 1.1 JAR File SmartGuid로 반출이 열립니다.
3. **JAR 파일** 필드에 *drive*:\WebSphere\CommerceServerDev\mytemp_c\CustomizedWCSUserDeployed_DT.jar를 입력하십시오.
4. 다음과 같이 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
대상 데이터베이스	DB2 DB2 데이터베이스로 전개 중인 경우 NT용 **DB2 V7.1**을 선택하십시오. Oracle Oracle 데이터베이스로 전개 중인 경우 **Oracle, V8**을 선택하십시오.
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
5. **완료**를 누르십시오.

JAR 파일이 작성됩니다.

JAR 파일의 이름은 "_DT" 접미부가 있습니다. 이는 WebSphere Commerce 응용프로그램에 JAR 파일을 전개하기 전, WebSphere Application Server가 제공하는 EJB 전개 도구로 JAR 파일이 실행되어야 되어야 함을 상기시킵니다.

wcsejsclient.jar 파일 작성

클라이언트 JAR 파일을 작성하려면 다음을 수행하십시오.

1. EJB 탭을 선택하여 WCS로 시작되는 이름의 모든 WebSphere Commerce EJB 그룹을 강조표시하십시오. 모든 그룹을 강조표시하고 마우스 오른쪽 버튼을 누른 후, **반출 > Client JAR**를 선택하십시오.
   반출 SmartGuide가 열립니다.
2. **JAR 파일** 필드에
   drive:\WebSphere\CommerceServerDev\mytemp_c\wcsejsclient.jar를 입력하십시오.
3. 다음과 같이 속성을 선택하십시오.

속성	값
bean	선택함
클래스	선택함
java	선택하지 않음
자원	선택함
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
4. **완료**를 누르십시오.

JAR 파일이 작성됩니다.

대상 상점 디렉토리로 갱신된 JSP 템플리트 복사

355 페이지의 『상품 표시 템플리트에 새 보너스 가격 추가』에서 새로 작성된 가격을 반영하기 위해 사용자가 표시 템플리트를 갱신했습니다. 이 단계에서 갱신된 JSP 템플리트를 WebSphere Test Environment 디렉토리 구조에서 상점에서 사용되는 디렉토리(WebSphere Test Environment 외부에서 실행 중일 경우)로 복사합니다.

1. 개발 시스템에서 *vaj_drive*:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web*store_directory* 디렉토리로 이동하십시오. 여기서, *vaj_drive*는 VisualAge for Java가 설치된 드라이브이며 *store_directory*는 견본 상점의 디렉토리 이름입니다.

   ProductDisplay.jsp 파일을 복사하십시오.

2. ProductDisplay.jsp 파일을 다음의 디렉토리에 붙여넣으십시오.

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instance_name.ear\
      wcstores.war\store_directory
   ```

   여기서, *drive*는 WebSphere Commerce가 설치된 드라이브이고 *store_directory*는 상점용 디렉토리 이름이며 *instance_name*은 WebSphere Commerce 인스턴스의 이름입니다.

대상 WebSphere Commerce Server로 JSP 파일 복사

JAR 파일을 개발 시스템에서 대상 WebSphere Commerce Server의 적절한 디렉토리로 복사해야 합니다. 이들 파일을 복사하려면 다음을 수행하십시오.

1. 개발 시스템에서, *drive*:\WebSphere\CommerceServerDev\mytemp_c 디렉토리로 이동하고 다음 파일을 위치 지정하십시오.

   - wcssamplesc.jar
   - CustomizedWCSUserDeployed_DT.jar
   - wcsejsclient.jar

   여기서 *drive*는 WebSphere Commerce Studio, Business Developer Edition을 설치한 드라이브입니다.

   이전의 개별 파일은 대상 WebSphere Commerce Server의 특정 디렉토리에 복사되어야 합니다. 개별 파일이 수정된 위치에 저장될 수 있도록 다음 단계를 주의깊게 읽으십시오.

2. 대상 WebSphere Commerce Server의 다음 디렉토리에 wcssamplesc.jar 파일을 복사하십시오.

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instance_name.ear\
      wcstores.war\WEB-INF\lib
   ```

여기서 *drive*는 WebSphere Commerce Business Edition을 설치한 드라이브이고, *instance_name*은 사용자 인스턴스의 이름입니다(예: demo).

3. 대상 WebSphere Commerce Server의 다음 디렉토리에 wcsejsclient.jar 파일을 복사하십시오.

   drive:\WebSphere\CommerceServer\temp\lib

4. 대상 WebSphere Commerce Server의 디렉토리에 CustomizedWCSUserDeployed_DT.jar 파일을 복사하십시오.

   drive:\WebSphere\CommerceServer\temp

EJB 전개 도구 실행

새 EJB 그룹을 포함하는 JAR 파일에 반하는 EJB 전개 도구를 실행해야 합니다. 이 도구는 WebSphere Application Server와 함께 포함됩니다.

이 도구를 실행하려면 다음을 수행하십시오.

1. 명령 프롬프트에서 다음 디렉토리로 이동하십시오.

   drive:\WebSphere\CommerceServer\temp

2. 다음 명령을 입력해서 시스템 경로에 도구를 임시로 추가하십시오.

   PATH=*drive*:\WebSphere\AppServer\deploytool;%PATH%

3. 다음 ejbdeploy 명령을 입력하십시오.

   ejbdeploy *EJBGroupJARFile WorkingDir OutputJARFile* -nowarn -keep -35 -cp *ClassPathOfDepJARFiles*

   여기서

   - *EJBGroupJARFile*은 EJB 그룹에 대한 JAR 파일의 이름입니다. 이 경우, 이는 CustomizedWCSUserDeployed_DT.jar입니다.
   - *WorkingDir*은 작업 디렉토리입니다.
   - *OutputJARFile*은 출력된 JAR 파일의 이름입니다. 이 경우에는 CustomizedWCSUserDeployed.jar를 입력하십시오.
   - -nowarn은 경고 및 정보 메시지를 억제하기 위한 선택적 매개변수입니다.
   - -keep은 ejbdeploy 명령을 실행한 후에 작업 디렉토리를 유지하기 위한 선택적 매개변수입니다.

- -35는 WebSphere Application Server 버전 3.5와 함께 제공되는 EJB 전개 도구에 사용되는 CMP 엔티티 bean용 하향식 맵핑 규칙을 사용하는 필수 매개변수입니다.
- -cp *ClassPathOfDepJARFiles*는 종속 JAR 파일의 클래스 경로입니다. 기존의 WebSphere Commerce 엔터프라이즈 bean을 수정한 경우, 종속 JAR 파일의 클래스 경로에 wcsejsclient.jar, wcsejbimpl.jar 및 xml4j.jar 파일이 있어야 합니다. 다음과 같이 입력하십시오.

  ```
  "drive:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
  drive:\WebSphere\AppServer\InstalledApps\
  WC_Enterprise_App_instanceName.ear\lib\wcsejbimpl.jar;
  drive:\WebSphere\AppServer\InstalledApps\
  WC_Enterprise_App_instanceName.ear\lib\xml4j.jar;"
  ```

엔티티 bean의 트랜잭션 분리 레벨 수정

이 단계에서 modifyIsolationLevel 명령을 사용하여 엔티티 bean의 트랜잭션 분리 레벨을 특정 데이터베이스 유형의 필수 레벨로 수정하십시오.

modifyIsolationLevel 명령을 실행하려면, 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server의 다음 디렉토리로 이동하는 명령 프롬프트를 사용하십시오.

   ```
   drive:\WebSphere\CommerceServer\bin
   ```

2. 다음과 같은 일반 구문의 modifyIsolationLevel 명령을 발행해야 합니다.

   ```
   modifyIsolationLevel -jarFile jar_file_name.jar
   -logFile log_file_name -dbType db_type
   ```

   여기서

   - *jar_file_name.jar*은 사용자에 맞게 정의된 코드를 포함하는 JAR 파일의 이름입니다.
   - *log_file_name*은 정보를 기록해야 하는 완전한 파일 이름입니다.
   - *db_type*은 사용 중인 데이터베이스의 유형입니다. DB2 또는 ORACLE을 입력하십시오.

   다음은 모든 값이 지정된 modifyIsolationLevel 명령의 예입니다.

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

주: 매개변수 이름은 대소문자가 구분됩니다. 즉, **jarFile**은 **jarfile**과 다릅니다. 매개변수 이름을 정확하게 입력했는지 확인하십시오.

예외가 명령창에 표시되지 않은 경우 명령이 정상적으로 실행된 것입니다. 완료 후에, 전개된 JAR 파일에서 시간소인이 변경되었음에 유의하십시오.

대상 데이터베이스 갱신

WebSphere Test Environment에서 사용한 데이터베이스와 다른 데이터베이스를 사용하는 대상 WebSphere Commerce Server에 전개할 경우 다음과 같이 대상 데이터베이스를 갱신해야 합니다.

- 229 페이지의 제 9 장 『학습: 새 비즈니스 로직 작성』 작업을 완료한 경우, USERS 테이블에서 각 사용자에 대한 행을 추가하려면 테이블을 갱신해야 합니다.
- 229 페이지의 제 9 장 『학습: 새 비즈니스 로직 작성』 작업을 완료하지 않았으면 테이블을 작성하고 대량 자료 반입해야 합니다.

229 페이지의 제 9 장 『학습: 새 비즈니스 로직 작성』 작업을 완료하지 않았으면 테이블을 작성하고 대량 자료 반입해야 합니다. 이러한 각 시나리오에 대한 지시사항이 제공됩니다.

DB2 DB2 데이터베이스를 사용하고 있으며 BONUS 테이블을 *갱신*해야 하는 경우, 다음을 수행하십시오.

1. DB2 명령 센터(**시작 > 프로그램 > IBM DB2 > 명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. **스크립트** 필드에 다음을 입력하십시오.

   ```
   connect to your_database_name
   ```

   여기서 *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.
3. **스크립트** 필드에 다음을 입력한 후 실행 아이콘을 누르십시오.

```
INSERT INTO BONUS
(SELECT USERS_ID, 0
FROM USERS
WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
```

이제 BONUS 테이블이 갱신되었습니다.

DB2 DB2 데이터베이스를 사용하고 있으며 BONUS 테이블을 *작성하고 대량 자료 반입*하는 경우, 다음을 수행하십시오.

1. DB2 명령 센터(**시작** > **프로그램** > **IBM DB2** > **명령 센터**)를 열고 **스크립트** 탭을 누르십시오.
2. **스크립트** 필드에 다음을 입력하십시오.

   ```
   connect to your_database_name
   ```

   여기서 *your_database_name*은 데이터베이스의 이름입니다. 실행 아이콘을 누르십시오.
3. **스크립트** 필드에 다음을 입력한 후 실행 아이콘을 누르십시오.

   ```
   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 (users_id) on delete cascade)
   ```

   이제 BONUS 테이블이 작성되었습니다.
4. 테이블에 대량 자료를 반입하려면 **스크립트** 필드에 다음을 입력한 후, 실행 아이콘을 누르십시오.

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```
5. DB2 명령 센터를 종료하십시오.

Oracle Oracle 데이터베이스를 사용하고 있으며 BONUS 테이블을 *갱신*해야 하는 경우, 다음을 수행하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.

5. SQL Plus 창에서 다음 정보를 입력하여 BONUS 테이블을 갱신하십시오.

```
INSERT INTO BONUS
(SELECT USERS_ID, 0
FROM USERS
WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
```

Enter를 눌러 SQL 문을 실행하십시오.
이제 BONUS 테이블이 갱신되었습니다.

6. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL 문을 실행하십시오.

Oracle Oracle 데이터베이스를 사용하고 있으며 BONUS 테이블을 *작성하고 대량 자료 반입* 하는 경우, 다음을 수행하십시오.

1. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
2. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
3. **암호** 필드에 Oracle 암호를 입력하십시오.
4. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
5. SQL Plus 창에서 다음 정보를 입력하여 BONUS 테이블을 작성하십시오.

```
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);
```

Enter를 눌러 SQL 문을 실행하십시오.
이제 BONUS 테이블이 작성되었습니다.

6. 다음 정보를 입력하여 BONUS 테이블에 대량 자료 반입하십시오.

```
insert into BONUS (select USERS_ID, 0 from USERS);
```

7. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL 문을 실행하십시오.

WebSphere Application Server에서 현재 진행 중인 엔터프라이즈 응용프로그램 반출

이 단계에서는 현재 엔터프라이즈 응용프로그램을 WebSphere Application Server에서 반출하여 응용프로그램 어셈블리 도구에서 다음에 열 수 있도록 합니다.

다음을 수행하여 현재 엔터프라이즈 응용프로그램을 WebSphere Application Server에서 반출하십시오.

1. WebSphere Application Server 관리 콘솔을 여십시오.
2. **WebSphere 관리 도메인**을 펼치십시오.
3. **엔터프라이즈 응용프로그램**을 펼치십시오.
4. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **데모** 응용프로그램을 펼쳐 **반출 응용프로그램**을 선택하십시오.
5. **반출 디렉토리** 필드에서 *drive*:\WebSphere\CommerceServer\working을 입력하십시오. 이것은 WC_Enterprise_App_*instanceName*.ear 파일의 모든 자원을 포함하는 전체 응용프로그램을 반출합니다(여기서, *instanceName*은 WebSphere Commerce 인스턴스의 이름입니다).

   주: 이미 기존의 WC_Enterprise_App_*instanceName*.ear 파일이 있다면, 이전 파일의 이름을 바꾸거나 겹쳐쓸 수 있습니다.

엔터프라이즈 응용프로그램용 XML 구성 정보 반출

엔터프라이즈 응용프로그램용 XML 또한 반출해야 합니다. 이 정보를 반출하려면 WebSphere Application Server가 제공하는 명령행 유틸리티 XMLConfig를 사용합니다.

구성 정보를 반출하려면 다음을 수행하십시오.

1. 명령 프롬프트에서 다음 디렉토리로 이동하십시오.

   drive:\WebSphere\CommerceServer\working

2. 다음 명령을 입력해서 부분 반출을 수행하려면 XMLConfig 도구를 호출하십시오.

```
xmlConfig -export OutputFileB.xml -partial was.export.app.xml
   -adminNodeName wasHostName
```

여기서, *wasHostName*은 현재 엔터프라이즈 응용프로그램을 포함하는 WebSphere Application Server의 노드 이름입니다. 또한 OutputFile.xml은 이 명령 실행의 결과로 작성되는 파일의 이름입니다.

현재 진행 중인 엔터프라이즈 응용프로그램의 엔터프라이즈 beans을 반출 한 다음, OutputFileB.xml 파일을 갱신해야 합니다. 이는 수정한 사용자 bean에 대한 코드가 있는 JAR 파일을 지칭합니다.

OutputFileB.xml 파일을 갱신하려면, 다음을 수행하십시오.

1. 다음 디렉토리로 이동하십시오.

   drive:\WebSphere\CommerceServer\working

2. 텍스트 편집기에서 OutputFileB.xml 파일을 여십시오.
3. <ear-file-name> 태그의 위치를 지정하고 다음으로 값을 바꾸십시오.

   drive:\WebSphere\CommerceServer\working\
   WC_Enterprise_App_*instanceName*.ear

4. WCSUser EJB 그룹에 대한 <ejb-module> 스탠자를 찾아서 <jar-file> 태그에 포함된 값을 CustomizedWCSUserDeployed.jar로 변경하십시오.
5. OutputFileB.xml 파일을 저장하십시오.

엔터프라이즈 응용프로그램에 수정된 EJB 그룹 어셈블링

이 단계에서 응용프로그램 어셈블러 도구의 엔터프라이즈 응용프로그램을 여십시오. 도구를 열어 엔터프라이즈 응용프로그램에 수정된 사용자 bean을 포함하려면 다음을 수행하십시오.

1. WCSUser EJB 그룹의 기존 버전에 대한 클래스 경로의 사본을 작성하십시오.
2. WCSUser EJB 그룹의 기존 버전을 제거하십시오.
3. WCSUser EJB 그룹의 새 버전을 반입하십시오. 새 EJB 그룹에 대한 JAR 파일은 엔터프라이즈 응용프로그램의 EJB 모듈 섹션 내에 저장됩니다.
4. WCSUser EJB 그룹에 대한 클래스 경로를 설정하십시오.

엔터프라이즈 응용프로그램에 새 EJB 그룹을 어셈블링하려면 다음을 수행하십시오.

1. 다음을 수행하여 현재 진행 중인 엔터프라이즈 응용프로그램을 백업하십시오.

a. 명령 프롬프트에서 다음 디렉토리로 이동하십시오.

```
drive:\WebSphere\CommerceServer\working
```

b. 다음 명령을 입력하십시오.

```
copy WC_Enterprise_App_instanceName.ear
   WC_Enterprise_App_instanceName.ear.bak2
```

2. WebSphere Application Server 관리 콘솔을 여십시오.
3. **도구** 메뉴에서 **응용프로그램 어셈블리 도구**를 선택하십시오(환영 창이 열리면 **취소**를 선택하여 콘솔에 액세스하십시오).
4. 다음을 수행해서 작업을 하려면 엔터프라이즈 응용프로그램을 여십시오.
   a. **파일** 메뉴에서 **열기**를 선택하십시오.
   b. **파일 이름** 필드에 다음을 입력하십시오.

      ```
      drive:\WebSphere\CommerceServer\working\
         WC_Enterprise_App_instanceName.ear
      ```

      열기를 누르십시오.응용프로그램을 열어 다음 단계를 계속하기 전에, 잠시 대기하십시오. 이 작업에는 몇 분이 걸립니다.
5. **EJB 모듈**을 누르십시오. 오른쪽 분할창이 엔터프라이즈 응용프로그램 내의 EJB 모듈을 표시합니다.
6. **WCSUser** EJB 모듈을 누르십시오.
7. 기존의 WCSUser EJB 모듈에 대한 클래스 경로 정보를 보려면 일반 탭을 누르십시오. 텍스트 파일로 기존의 클래스 경로 정보를 복사하십시오(예: WCSUser_path.txt).
8. **WCSUser** EJB 모듈을 마우스 오른쪽 버튼으로 누른 후 **삭제**를 선택하십시오.
9. **EJB Modules**을 마우스 오른쪽 버튼으로 누른 후 **반입**을 선택하십시오.
10. **파일 이름** 필드에 다음을 입력하십시오.

    ```
    drive:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
    ```

    그리고 **열기**를 누르십시오. 확인값 창에서 **확인**을 누르십시오.

11. CustomizedWCSUserDeployed.jar이 반입되면 **WCSUser** EJB 그룹으로 화면이동하고, 해당 그룹을 선택하십시오.
    이 그룹에 관한 정보는 오른쪽 분할창에 표시됩니다.
12. WCSUser EJB 그룹의 이전 버전에 대한 클래스 경로 정보를 포함하는 텍스트 파일을 여십시오. 클래스 경로를 선택해서 복사하십시오.
13. 새 WCSUser EJB 그룹에 대한 클래스 경로 필드에서 이 클래스 경로 정보를 붙여넣으십시오.
14. **적용**을 누르십시오.
15. **파일** 메뉴에서 **종료**를 선택하십시오.
16. 파일이 닫힐 때까지 기다린 후 **파일** 메뉴에서 **열기**를 선택하고 *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear 파일을 다시 여십시오.
17. 사용자 bean용 보안을 구성하려면 다음을 수행하십시오.
    a. 펼쳐진 EJB 모듈 노드로 **WCSUser** 노드를 위치 지정하고 펼치십시오.
    b. **엔티티 Bean**을 펼치십시오.
    c. **사용자**를 펼치십시오.
    d. **메소드 확장자**를 누르고 오른쪽 분할창에서 다음을 수행하십시오.
        1) **고급** 탭을 선택하십시오.
        2) **보안 동일성**이 선택되었는지 확인하십시오.
        3) 개별 메소드에 대해서, **EJB 서버의 동일성 사용**이 선택되었는지 확인하십시오.
        4) **적용**을 누르십시오(수정한 경우).
    e. 왼쪽 탐색 분할창에서 WCSUser EJB 그룹 아래의 **보안 역할**을 마우스 오른쪽 버튼으로 눌러 **새로 만들기**를 선택한 후 다음을 수행하십시오.
        1) **이름** 필드에서 WCSecurityRole를 입력한 후, **적용**을 누르십시오. 이 역할이 이미 존재하는 경우, 이 단계를 수행하지 않아도 됨에 유의하십시오.
    f. 왼쪽 탐색 분할창에서 WCSUser EJB 그룹 아래의 **메소드 권한**을 마우스 오른쪽 버튼으로 누르고 **새로 만들기**를 선택한 후 다음을 수행하십시오.

1) **메소드 허용 이름** 필드에 WCMethodPermission을 입력하십시오.
2) **메소드** 선택 영역에서 **추가**를 누르십시오.
메소드 추가 창이 열립니다.
3) **CustomizedWCSUserDeployed.jar**를 펼치고 모든 엔터프라이즈 bean을 선택하고(Shift 키를 누른 상태에서 선택) **확인**을 누르십시오. 모든 엔터프라이즈 bean은 엔터프라이즈 bean 열에 표시되고 **모든 메소드**는 유형 열에 표시됩니다.
4) 역할 선택 영역에서 **추가**를 누르십시오. WCSecurityRole을 선택한 후 **확인**을 누르십시오.
5) **적용**을 누른 후 **확인**을 누르십시오.

18. **파일** 메뉴에서 **저장**을 선택하십시오.
19. 응용프로그램 어셈블러 도구를 종료하십시오.

이 단계를 완료하면 새로운 비즈니스 로직 뿐만 아니라 이전의 모든 로직을 포함하는 새 엔터프라이즈 응용프로그램이 작성됩니다. 새로 수정된 WC_Enterprise_App_*instanceName*.ear 파일에 이 모든 것이 포함됩니다.

WebSphere Application Server로 새 엔터프라이즈 응용프로그램 반입

다음은 WebSphere Application Server에 새 엔터프라이즈 응용프로그램을 반입하는 것과 관련된 상위 레벨 단계입니다.

1. WebSphere Application Server에서 현재 실행 중인 엔터프라이즈 응용프로그램을 중지 및 제거. 이러한 단계는 WebSphere Application Server 관리 콘솔에서 수행됩니다.
2. XMLConfig 명령행 유틸리티를 사용하여 새 응용프로그램을 반입합니다.
3. WebSphere Application Server 관리자의 콘솔을 최신 정보로 고쳐 새 엔터프라이즈 응용프로그램을 시작합니다.

각 단계는 다음 절에 자세히 설명되어 있습니다.

현재 진행 중인 엔터프라이즈 응용프로그램의 중지 및 제거: WebSphere Application Server에서 사용자가 현재 진행 중인 엔터프라이즈 응용프로그램을 중지 및 제거하려면 다음을 수행하십시오.

1. WebSphere Application Server 관리 콘솔을 여십시오.

2. **WebSphere 관리 도메인**을 펼치십시오.
3. **노드**를 펼치십시오.
4. *nodeName*을 펼치십시오(여기서, *nodeName*은 사용자 노드의 이름입니다).
5. **응용프로그램 서버**를 펼치십시오.
6. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server - *instanceName***을 마우스 오른쪽 버튼으로 누른 후 **중지**를 선택하십시오.
7. **엔터프라이즈 응용프로그램**을 펼치십시오.
8. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Enterprise Server - 데모** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 **중지**를 선택하십시오.
9. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Enterprise Server - 데모** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 **제거**를 선택하십시오.
10. 응용프로그램의 반출 표시가 프롬프트될 때, **아니오**를 선택하십시오.

XMLConfig를 사용하여 새 엔터프라이즈 응용프로그램 반입: XMLConfig 명령행 유틸리티를 사용하여 새 엔터프라이즈 응용프로그램을 반입하려면 다음을 수행하십시오.

1. 다음 디렉토리로 탐색하십시오.

   ```
   drive:\WebSphere\CommerceServer\working
   ```

2. 명령 프롬프트에서 WebSphere Application Server에 엔터프라이즈 응용프로그램을 반입하려면 다음 명령을 입력하십시오.

   ```
   xmlConfig -import OutputFileB.xml -adminNodeName was_hostname
   ```

   여기서, *was_hostname*은 현재 진행중인 응용프로그램을 포함하는 WebSphere Application Server 노드의 이름입니다.

주: 400 iSeries에서 실행 중인 WebSphere Commerce 인스턴스에 전개했을 경우, 응용프로그램 반입 후 디렉토리 권한을 수정하기 위한 추가 단계를 수행해야 합니다. 이러한 사용권한의 수정 방법에 대한 자세한 내용은 422 페이지의 『엔터프라이즈 응용프로그램 반입』을 참조하십시오.

새 엔터프라이즈 응용프로그램 시작: XMLConfig 명령행 유틸리티를 사용하여 새 엔터프라이즈 응용프로그램을 반입한 후, WebSphere Application Server 관리자 콘솔을 사용하여 최신 정보로 고쳐 새 응용프로그램을 시작할 수 있습니다.

콘솔을 최신 정보로 고치고 새 응용프로그램을 시작하려면, 다음을 수행하십시오.

1. WebSphere Application Server 관리 콘솔을 여십시오.
2. **WebSphere 관리 도메인**을 펼치십시오.
3. **노드**를 강조표시하십시오.
4. **선택된 서브트리를 최신 정보로 고침** 아이콘을 누르십시오.
5. 다음을 수행하여 WebSphere Commerce 응용프로그램을 시작하십시오.
   - **응용프로그램 서버**를 펼치십시오.
   - WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server** - *instanceName*을 마우스 오른쪽 버튼으로 누른 후 **시작**을 선택하십시오.

대상 상점의 새 코드 테스트

WebSphere Application Server에서 실행 중인 상점의 수정된 엔티티 bean 및 새 태스크 명령을 테스트하려면 다음을 수행하십시오.

1. 브라우저를 열어 다음 URL을 입력하십시오.

   ```
   http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=store_Id&catalogId=catalog_Id&langId=-1
   ```

   여기서 *store_Id*는 상점 식별자이고 *catalog_Id*는 상점 카탈로그 식별자입니다.

   주: 오류 500이 발생할 경우, WebSphere Application Server를 다시 시작하여 엔터프라이즈 응용프로그램을 최신 정보로 고쳐야 할 수도 있습니다.

2. 서비스 헤딩 아래 있는 **등록** 링크를 누른 후 처음 방문 고객 헤딩 아래 있는 **등록**을 누르십시오.
   `wctester@wc`의 전자 우편 주소와 `wctester1`의 암호를 사용하여 처음 방문 고객을 등록하십시오. 다른 필드도 테스트 값으로 채우고 **제출**을 누르십시오. 브라우저는 열린 상태로 두십시오.

3. DB2 DB2 명령 센터를 열고 다음을 수행하십시오.
   a. **대화식** 탭을 누르십시오.
   b. **명령** 필드에서 다음을 수행하십시오.
      1) 다음을 입력하십시오.

         ```
         connect to your_database_name
         ```

         여기서 *your_database_name*은 WebSphere Commerce 데이터베이스 이름입니다. 실행 아이콘을 누르십시오.
      2) `select users_id from USERREG where LOGONID = 'wctester@wc'`를 입력하고 실행 아이콘을 누르십시오.
   c. 조회 결과 탭은 2단계에서 등록한 고객에 대한 항목을 표시합니다. 다음에 고객의 USERS_ID 값을 기록하십시오 _____
   d. 새로 등록한 고객의 보너스 점수 잔고를 갱신하십시오. 대화식 탭을 누른 후 **명령** 필드에 다음을 입력하십시오.

      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id
      ```

      여기서 *users_id*는 3c단계의 값입니다. 실행 아이콘을 누르십시오.

4. Oracle 다음을 수행하여 테스트 사용자의 보너스 포인트 대차 대조를 갱신하십시오.
   a. Oracle SQL Plus 명령창을 여십시오(**시작** > **프로그램** > **Oracle** > **Application Development** > **SQL Plus**).
   b. **사용자 이름** 필드에 Oracle 사용자 이름을 입력하십시오.
   c. **암호** 필드에 Oracle 암호를 입력하십시오.
   d. **호스트 문자열** 필드에 연결 문자열을 입력하십시오.
   e. 사용자의 ID를 판별하려면 `select users_id from USERREG where LOGONID = 'wctester@wc';`를 입력하십시오.
   f. 2단계에서 등록한 고객에 대한 항목이 표시됩니다. 다음에 고객의 USERS_ID 값을 기록하십시오. _____
   g. 다음을 입력하여 새로 등록한 고객의 보너스 점수 잔고를 갱신하십시오.

      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id;
      ```

여기서 *users_id*는 4f단계의 값입니다

h. 다음을 입력하여 데이터베이스 변경을 확약하십시오.

```
commit;
```

Enter를 눌러 SQL 문을 실행하십시오.

5. 브라우저에서 **남성의류** 링크를 눌러 상점의 남성의류 섹션을 보십시오.
6. 기획 상품에 대한 링크를 눌러 상품 페이지를 보십시오. 페이지에는 정상 가격과 고객의 보너스 점수 잔고에 근거한 할인 가격이 표시됩니다.

제 5 부 부록

부록 A. WebSphere Test Environment 시작 및 중지

이 절에서는 VisualAge for Java 내에서 WebSphere Test Environment를 시작하는 단계에 대해 설명합니다. 일반적으로 이 환경을 시작하려면 다음 단계를 수행해야 합니다.

1. PNS를 시작하십시오.
2. EJB 서버를 시작하십시오.
3. Servlet 엔진을 시작하십시오.

이들 각각의 단계는 다음 절에 설명되어 있습니다.

테스트 환경을 중지하려면 단계의 순서를 반대로 수행하십시오.

주: WebSphere Test Environment의 모든 기능을 사용하려면 WebSphere Test Environment를 시작하기 전에 WebSphere Application Server가 실행되고 있지 않아야 합니다. WebSphere Application Server 중지에 대한 자세한 내용은 *WebSphere Commerce 설치 안내서*를 참조하십시오.

PNS 시작 및 중지

PNS는 엔터프라이즈 bean의 클라이언트에서 찾아보기 요청을 수신합니다. 모든 엔터프라이즈 bean은 PNS로 등록됩니다.

PNS를 시작하거나 중지하려면 다음을 수행하십시오.

1. VisualAge for Java의 **작업 영역** 메뉴에서 **도구 > WebSphere Test Environment**를 선택하십시오.
   WebSphere Test Environment 제어 센터가 열립니다.
2. **PNS**를 누른 후 다음 중 하나를 누르십시오.
   - **이름 서버 시작**
     콘솔 창에서 "서버가 business를 위해 시작되었습니다." 메시지를 기다리십시오. 이 서버를 시작하는 데에는 몇 분이 걸릴 수도 있습니다.

- 이름 서버 중지

EJB 서버 시작 및 중지

EJB 서버는 엔터프라이즈 bean을 사용하는 서버 응용프로그램 실행을 지원하기 위해 런타임 환경을 제공하는 고급 처리 또는 응용프로그램입니다.

EJB 서버를 시작하거나 중지하려면 다음을 수행하십시오.

1. EJB 서버 구성 창을 여십시오(EJB 탭을 누른 후 **EJB** 메뉴에서 **열기 > 서버 구성**을 선택하십시오).
2. 시작하거나 중지할 EJB 서버(예: **EJB Server {Server 1}**)를 마우스 오른쪽 버튼으로 누른 후 다음 중 하나를 수행하십시오.
   - **서버 시작** 선택
     콘솔 창에서 "서버는 business용으로 열립니다." 메시지를 기다리십시오(이 서버의 상태를 보기 위해 콘솔에서 **EJB 서버**를 선택해야 할 경우도 있습니다). 이 서버를 시작하는 데는 컴퓨터에 따라 10분에서 15분이 걸릴 수 있습니다.
   - **서버 중지** 선택

콘솔은 IDE의 Java 프로그램용 표준 입력 및 출력 장치입니다. 특정 프로그램의 출력을 보려면 먼저 모든 프로그램 분할창에서 프로그램을 선택하십시오. 그러면 해당 출력이 출력 분할창에 표시됩니다.

Servlet 엔진 시작 및 중지

Servlet 엔진은 웹 서버와 WebSphere Application Server 기능을 통합하여 테스트용 환경을 작성합니다.

Servlet 엔진을 시작하거나 중지하려면 다음을 수행하십시오.

1. VisualAge for Java의 **작업 영역** 메뉴에서 **도구 > WebSphere Test Environment**를 선택하십시오.
   WebSphere Test Environment 제어 센터가 열립니다.
2. **Servlet 엔진**을 누른 후 다음 중 하나를 누르십시오.

- **Servlet 엔진 시작**
  Servlet 엔진이 시작되면 콘솔에는 ***Servlet 엔진이 시작되었습니다.*** 메시지가 표시됩니다.
- **Servlet 엔진 중지**

부록 B. 전개 정보

VisualAge for Java에서 사용자 정의 코드를 작성하고 WebSphere Test Environment에서 테스트한 후, WebSphere Test Environment 외부에서 실행 중인 WebSphere Commerce 인스턴스로 전개해야 합니다. WebSphere Commerce 인스턴스는 개발 시스템에서 로컬로 실행되거나, 다른 시스템에서 실행될 수 있습니다(동일한 또는 다른 운영체제 사용).

이 부록에서는 WebSphere Test Environment 외부에서 실행 중인 WebSphere Commerce 인스턴스로 사용자 정의 코드를 전개하는 데 필요한 단계에 대해 설명합니다. 214 페이지의 『코드 전개』를 참조하여 전개 처리의 상위 레벨 단계를 이해한 다음, 이 부록에서 자세한 내용을 참조해야 합니다.

IFS에 맵핑(iSeries)

400 이 절은 대상 WebSphere Commerce Server가 iSeries 플랫폼에서 실행 중일 경우에만 적용됩니다.

대상 WebSphere Commerce Server가 iSeries 플랫폼에서 실행 중일 경우, 사용자 개발 시스템의 로컬 드라이브를 iSeries 서버 기반의 IFS(Integrated File System)에 맵핑해야 합니다. 이 문서의 다음 절에서 *iSeries_drive*는 IFS에 맵핑한 해당 로컬 드라이브를 간주합니다. 추가적으로, *drive*는 사용자 개발 시스템(IFS체 맵핑되지 못함)의 로컬 드라이브입니다.

사용자 정의 명령 및 데이터 bean용 JAR 파일

사용자 정의 명령 및 데이터 bean 코드는 WebSphere Commerce 코드와 별도의 프로젝트에 저장해야 합니다. WebSphere Test Environment 내에서 테스트를 마쳤으면, 사용자 정의 명령 및 데이터 bean 코드를 포함하는 프로젝트용 JAR 파일을 작성한 다음 JAR 파일을 대상 WebSphere Commerce Server의 해당 디렉토리에 전개해야 합니다.

사용자 정의 명령 및 데이터 bean용 JAR 파일을 작성하려면 다음을 수행하십시오.

1. VisualAge for Java의 워크벤치에서 **프로젝트** 탭을 선택하십시오.
2. 사용자 정의 명령 및 데이터 bean 코드를 포함하는 프로젝트를 마우스 오른쪽 버튼으로 누른 후 **반출**을 선택하십시오.
   반출 SmartGuide가 열립니다.
3. **Jar 파일**을 선택한 후 **다음**을 누르십시오.
4. **Jar 파일** 필드에 다음을 입력하십시오.
   drive:\WebSphere\CommerceServerDev*jar_file_name*_1.jar
   여기서 *drive*:\WebSphere\CommerceServerDev\는 JAR 파일에 충분한 여유 공간을 갖고 있는 임시 디렉토리이고 *jar_file_name*_1.jar는 _1을 첨부한 JAR 파일 이름입니다.
5. 다음과 같이 JAR 파일의 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
bean	선택함
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
6. **완료**를 누르십시오.

작성된 구현 JAR 파일이 완료된 패키지 이름 정보를 포함하고 있지 않기 때문에, JAR 파일을 다시 패키지하는 데 다른 패키지 유틸리티(VisualAge for Java 이외)를 사용해야 합니다. 파일을 다시 패키지하려면 다음을 수행하십시오.

1. 명령창의 앞 단계에서 *jar_file_name*_1.jar를 저장한 디렉토리로 이동하십시오.
2. mkdir temp1을 입력하십시오.
3. cd temp1을 입력하십시오.

4. 경로를 set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio\bin;으로 설정하십시오.
   여기서 *drive*는 WebSphere Studio가 설치된 드라이브입니다.
5. jar xvf ../*jar_file_name*_1.jar를 입력하십시오.
6. jar cvf ../*jar_file_name*.jar * 를 입력하십시오(이름에서 _1을 제거해야 함에 유의하십시오).
7. *drive*:\WebSphere\CommerceDev*temp_directory* 디렉토리로 전환하십시오.
8. 로컬 WebSphere Commerce 인스턴스로 전개 중인 경우, *jar_file_name*.jar를 다음 디렉토리에 복사하십시오.

   ```
   drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instanceName.ear\wcstores.war\WEB-INF\lib
   ```

   디렉토리. 그렇지 않은 경우, 대상 WebSphere Commerce Server로 모든 파일 자원을 전송해야 할 때까지 JAR 파일을 임시 디렉토리에 남겨 두십시오.

새 엔티티 bean용 JAR 파일 작성

엔티티 bean을 작성할 때는 모든 WebSphere Commerce 코드와는 별도의 프로젝트에 코드를 저장해야 합니다. 또한 사용자 정의 명령 및 데이터 bean 코드와는 별개여야 합니다. 새 엔티티 bean은 WebSphere Commerce 공용 엔티티 bean을 포함하는 EJB 그룹과 다른 EJB 그룹에서 작성해야 합니다.

새 엔티티 bean을 전개할 때는 두 개의 JAR 파일을 작성해야 합니다. 첫 번째 JAR 파일은 EJB 1.1 Export JAR이고 EJB 탭을 선택했을 때 사용 가능한 도구를 사용하여 작성합니다. 두 번째 JAR 파일은 엔티티 bean용 구현 코드를 포함하며, 파일은 엔티티 bean 코드를 포함하는 프로젝트를 통해 작성됩니다. 두 번째 JAR 파일은 VisualAge for Java 워크벤치에서 프로젝트 탭을 선택했을 때 사용 가능한 도구를 사용하여 작성합니다.

EJB 1.1 Export JAR 파일

새 엔티티 beans용 EJB 1.1 Export JAR 파일을 작성하려면 다음과 같이 하십시오.

1. VisualAge for Java의 워크벤치에서 **EJB** 탭을 선택하십시오.
2. 사용자 정의 엔티티 bean 코드를 포함하는 EJB 그룹을 마우스 오른쪽 버튼으로 누른 후 > **EJB 1.1 JAR 반출**을 선택하십시오.
   EJB 1.1 JAR 파일 반출 SmartGuide가 열립니다.
3. **Jar 파일** 필드에 *drive*:\WebSphere\CommerceServerDev\temp_directory*jarFileName*_DT.jar를 입력하십시오.
   여기서, *drive*:\WebSphere\CommerceServerDev\temp_directory는 JAR 파일에 충분한 여유 공간을 갖고 있는 임시 디렉토리이고 *jarFileName*_DT.jar는 _DT 접미부를 추가한 JAR 파일 이름입니다.

JAR 파일의 이름은 "_DT" 접미부가 있습니다. 이는 WebSphere Commerce 응용프로그램에 JAR 파일을 전개하기 전에 WebSphere Application Server가 제공하는 EJB 전개 도구로 JAR 파일이 실행되어야 함을 상기시킵니다.

4. 다음과 같이 JAR 파일의 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
대상 데이터베이스	DB2 DB2 데이터베이스로 전개하려면 다음 중 하나를 선택하십시오. • Windows AIX Solaris Linux **NT용 DB2, V7.1** • AS/400용 400 **DB2, V4** Oracle Oracle 데이터베이스로 전개하려면, **Oracle, V8**을 선택하십시오.
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
5. **완료**를 누르십시오.

JAR 파일이 작성됩니다.

구현 JAR 파일 작성

새 엔티티 bean용 구현 JAR 파일을 작성하려면 다음을 수행하십시오.

1. VisualAge for Java의 워크벤치에서 **프로젝트** 탭을 선택하십시오.
2. 사용자 정의 엔티티 bean 코드를 포함하는 프로젝트를 마우스 오른쪽 버튼으로 누른 후 **반출**을 선택하십시오.
   반출 SmartGuide가 열립니다.
3. **Jar 파일**을 선택한 후 **다음**을 누르십시오.
4. **Jar 파일** 필드에 *drive*:\WebSphere\CommerceServerDev*temp_directory**jarFileName*_1.jar를 입력하십시오.
   여기서 *drive*:\WebSphere\CommerceServerDev*temp_directory*는 JAR 파일에 대해 충분한 여유 공간을 갖고 있는 임시 디렉토리이고, *jarFileName*_1.jar는 _1을 첨부한 JAR 파일 이름입니다.
5. 다음과 같이 JAR 파일의 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
bean	선택함
.class 파일에 디버그 속성 포함	선택함

   다른 속성에 대해서는 기본값을 사용하십시오.
6. **완료**를 누르십시오.

작성된 구현 JAR 파일이 완료된 패키지 이름 정보를 포함하고 있지 않기 때문에, JAR 파일을 다시 패키지하는 데 다른 패키지 유틸리티(VisualAge for Java 이외)를 사용해야 합니다. 파일을 다시 패키지하려면 다음을 수행하십시오.

1. 명령창의 앞 단계에서 *jarFileName*_1.jar를 저장한 디렉토리로 이동하십시오.
2. mkdir temp1을 입력하십시오.
3. cd temp1을 입력하십시오.

4. 경로를 set PATH=%PATH%;*drive*:\WebSphere\WebSphereStudio\bin;으로 설정하십시오.
   여기서 *drive*는 WebSphere Studio가 설치된 드라이브입니다.
5. jar xvf ../*jarFileName*_1.jar를 입력하십시오.
6. jar cvf ../*jarFileName*.jar * 를 입력하십시오(이름에서 _1을 제거해야 함에 유의하십시오).
7. *drive*:\WebSphere\CommerceServerDev*temp_directory* 디렉토리로 전환하십시오.
8. 로컬 WebSphere Commerce 인스턴스로 전개하는 경우, *jarFileName*_1.jar를 다음 디렉토리로 복사하십시오.
   drive:\WebSphere\CommerceServer\temp\lib
   그렇지 않은 경우, 모든 파일 자원을 대상 WebSphere Commerce Server로 전송해야 할 때까지 JAR 파일을 임시 디렉토리에 남겨 두십시오.

사용자 정의 WebSphere Commerce 엔티티 bean용 JAR 파일 작성

공용 WebSphere Commerce 엔티티 bean을 확장할 수 있습니다. 이 bean은 다음 EJB 그룹에 있습니다.

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction
- WCSCatalog
- WCSCommon
- WCSContract
- WCSCoupon
- WCSFulfillment
- WCSInventory
- WCSMessageExtensions
- WCSOrder

- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

공용 WebSphere Commerce 엔티티 bean을 확장하는 경우, 수정 WebSphere Commerce 공용 엔티티 bean을 포함하는 EJB 그룹의 EJB 1.1 Export JAR 파일 및 모든 WebSphere Commerce 엔티티 bean의 클라이언트 JAR 파일을 작성해야 합니다.

EJB 1.1 Export JAR 파일 작성

수정 엔티티 bean이 있는 EJB 그룹용 EJB 1.1 Export JAR 파일을 작성하려면, 다음을 수행하십시오.

1. VisualAge for Java의 워크벤치에서 **EJB** 탭을 선택하십시오.
2. 사용자 정의 엔티티 bean 코드를 포함하는 EJB 그룹을 마우스 오른쪽 버튼으로 누른 후 > **EJB 1.1 JAR 반출**을 선택하십시오.
   EJB 1.1 JAR 파일 반출 SmartGuide가 열립니다.
3. **Jar 파일** 필드에 다음을 입력하십시오.
   drive:\WebSphere\CommerceServerDev*temp_directory**EJBGroupJARFile*_DT.jar
   여기서, *drive*:\WebSphere\CommerceServerDev*temp_directory*는 JAR 파일에 대한 충분한 여유 공간을 갖고 있는 임시 디렉토리이고 *EJBGroupName*_DT.jar는 _DT 접미부를 추가한 JAR 파일 이름입니다.

JAR 파일의 이름은 "_DT" 접미부가 있습니다. 이는 WebSphere Commerce 응용프로그램에 JAR 파일을 전개하기 전에 WebSphere Application Server가 제공하는 EJB 전개 도구로 JAR 파일이 실행되어야 함을 상기시킵니다.

4. 다음과 같이 JAR 파일의 속성을 선택하십시오.

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
대상 데이터베이스	DB2 DB2 데이터베이스로 전개하려면 다음 중 하나를 선택하십시오. • Windows AIX Solaris Linux **NT용 DB2, V7.1** • AS/400용 400 **DB2, V4** Oracle Oracle 데이터베이스로 전개하려면 **Oracle, V8**을 선택하십시오.
.class 파일에 디버그 속성 포함	선택함

다른 속성에 대해서는 기본값을 사용하십시오.

5. **완료**를 누르십시오.

클라이언트 JAR 파일 작성

클라이언트 JAR 파일을 작성하려면 다음을 수행하십시오.

1. EJB 탭을 선택하여 WCS로 시작되는 이름의 모든 WebSphere Commerce EJB 그룹을 강조표시하십시오. 모든 그룹을 강조표시하고 마우스 오른쪽 버튼을 누른 후, **반출 > Client JAR**를 선택하십시오.
   반출 SmartGuide가 열립니다.
2. **JAR 파일** 필드에 다음을 입력하십시오.
   drive:\WebSphere\CommerceServerDev*temp_directory*\wcsejsclient.jar
3. 다음과 같이 속성을 선택하십시오.

속성	값
bean	선택함

속성	값
클래스	선택함
java	선택하지 않음
자원	선택함
.class 파일에 디버그 속성 포함	선택함

다른 속성에 대해서는 기본값을 사용하십시오.

4. **완료**를 누르십시오.

JAR 파일이 작성됩니다.

대상 WebSphere Commerce Server에 자원 저장

사용자 정의 코드와 연관된 자원을 대상 WebSphere Commerce Server에 복사해야 합니다. 이러한 자원으로는 사용자 정의 명령, 데이터 bean, 엔티티 bean용 JAR 파일이 있습니다. 또한 사용자 정의를 지원하는 JSP 템플리트 및 그래픽도 있어야 합니다.

AIX Solaris Linux *WebSphere Commerce 설치 안내서*의 *"사후 설치 스크립트 실행"* 절에 있는 단계를 수행할 때 작성한 사용자를 사용하여 대상 WebSphere Commerce Server에 대해 모든 전개 단계를 수행해야 합니다. 기본값으로, `wasuser`입니다. 또한 파일 자원(예: JAR 파일) 및 이러한 자원이 위치하는 디렉토리가 해당 사용자에 대해 권한이 부여된 파일을 읽고, 쓰며 실행할 수 있는지 확인하십시오.

400 `/QIBM/UserData/WebCommerce/instances/`*`instanceName`* 및 `/QIBM/UserData/WebCommerce/instances/`*`instanceName`*`/working` 디렉토리에 대한 권한이 사용자 `QEJB`가 포함되어 있는지 확인하십시오. 데이터 권한을 `*RWX`로 설정하여, 이 사용자를 두 디렉토리 모두에 추가하십시오.

다음 표는 Windows NT 또는 Windows 2000을 실행 중인 대상 WebSphere Commerce Server에 자원을 저장하는 표준 디렉토리를 요약합니다.

표 12.

자원 유형	대상 **WebSphere Commerce Server**의 디렉토리 위치
명령 및 데이터 bean 로직용 JAR 파일	*drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib
VisualAge for Java를 사용하여 작성한 EJB 1.1 Export JAR	*drive*:\WebSphere\CommerceServer\temp **주:** 이 파일은 WebSphere Application Server V4.0에서 사용되는 전개된 코드를 생성하기 위해 EJBDeploy 도구에 입력을 전달합니다.
EJB 클라이언트 코드의 JAR 파일	*drive*:\WebSphere\CommerceServer\temp\lib **주:** 이 파일은 EJBDeploy 도구의 클래스 경로에서만 사용됩니다.
EJB 구현 코드의 JAR 파일	*drive*:\WebSphere\CommerceServer\temp\lib **주:** 이 파일은 파일 자원으로서 엔터프라이즈 응용프로그램에 추가됩니다.
JSP 템플리트	*drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*storeDir*
이미지	*drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*storeDir*\images

대상 WebSphere Commerce Server에 자원을 저장하려면 다음을 수행하십시오.

1. 명령 및 데이터 bean 코드용 JAR 파일 위치를 지정하십시오. 이 파일은 개발 시스템의 다음 디렉토리 중 하나에 있어야 합니다.
   - Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib
   - Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*
2. 대상 WebSphere Commerce Server의 다음 디렉토리 중 하나에 명령 및 데이터 bean 코드용 JAR 파일을 복사하십시오.
   - Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war\WEB-INF\lib
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/WEB-INF/lib

- Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_instanceName.ear/wcstores.war/WEB-INF/lib`
- Linux `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_instanceName.ear/wcstores.war/WEB-INF/lib`
- 400 `/QIBM/UserData/WebASAdv4/WAS_AdminInstanceName/installedApps/WC_Enterprise_App_instanceName.ear/wcstores.war/WEB-INF/lib`

3. 개발 시스템의 다음 디렉토리에 있는 수정 WebSphere Commerce 엔티티 bean 뿐만 아니라 새 EJB 그룹용 EJB 1.1 Export JAR 파일의 위치를 지정하십시오.

   Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*

4. 대상 WebSphere Commerce Server의 다음 디렉토리 중 하나에 EJB 1.1 Export JAR 파일을 복사하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp
   - Solaris /opt/WebSphere/CommerceServer/temp
   - Linux /opt/WebSphere/CommerceServer/temp
   - 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/temp
5. 새 인터프라이즈 bean을 작성한 경우, 개발 시스템의 다음 디렉토리에 있는 이 bean의 구현 JAR 파일 위치를 지정하십시오.

   Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*

6. 새 엔터프라이즈 bean용 구현 JAR 파일을 대상 WebSphere Commerce Server의 다음 디렉토리로 복사하십시오.
   - Windows *drive*:\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/*instanceName*/temp/lib

7. 기존 WebSphere Commerce 엔티티 bean을 수정한 경우, 개발 시스템의 다음 디렉토리에 있는 클라이언트 JAR 파일의 위치를 지정하십시오.

   Windows *drive*:\WebSphere\CommerceServerDev*temp_directory*

8. 대상 WebSphere Commerce Server의 다음 디렉토리 중 하나에 클라이언트 JAR 파일을 복사하십시오.
   - Windows *drive*:\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/*instanceName*/temp/lib
9. 대상 WebSphere Commerce Server의 다음 디렉토리에 JSP 템플리트를 복사하십시오.
   - Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*store_directory*
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*

- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*
- 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*

여기서, *store_directory*는 상점 디렉토리입니다.

10. 대상 WebSphere Commerce Server의 다음 디렉토리에 이미지를 복사하십시오.
    - Windows *drive*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*instanceName*.ear\wcstores.war*store_directory*\images
    - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*/images
    - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*/images
    - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instanceName*.ear/ wcstores.war/*store_directory*/images
    - 400 /QIBM/UserData/WebASAdv4/*WAS_AdminInstanceName*/installedApps/WC_Enterprise_App_*instanceName*.ear/wcstores.war/*store_directory*/images

    여기서, *store_directory*는 상점 디렉토리입니다.

대상 데이터베이스 갱신

대상 WebSphere Commerce Server가 개발 시스템이 아닌 다른 데이터베이스를 사용할 경우, 대상 WebSphere Commerce Server에서 사용되는 데이터베이스에서 개발 데이터베이스에 수행한 모든 갱신사항을 수행해야 합니다. 여기에는 새 명령이나 수정된 명령의 등록, 작성된 추가 테이블 그리고 작성된 새 자원에 대한 액세스 제어 정책 작성에 대한 갱신사항이 포함됩니다.

(다양한 플랫폼 및 디렉토리 권한 요구사항에 대한 명령 구문을 포함) 액세스 제어 정책 로드에 대한 정보는 *WebSphere Commerce 액세스 제어 안내서*를 참조하십시오.

400 실행 중인 SQL 문에 대한 유틸리티를 가지고 있어야 합니다. 이를 위한 한 가지 방법은 Client Access Express V5R1(전체 설치)을 사용하는 것입니다. 이 유틸리티를 열려면 다음을 수행하십시오.

1. **작업 탐색기**를 여십시오.
2. 작업 탐색기가 열리면, 특정 시스템에 사인온해야 합니다. 대상 iSeries 시스템을 선택하고 WebSphere Commerce 인스턴스 사용자 프로파일 및 암호를 사용하십시오. 작성된 새로운 모든 테이블을 WebSphere Commerce 인스턴스 사용자 프로파일이 소유하는지 확인하십시오.
3. 왼쪽에 있는 패널에서 사용하는 시스템을 누른 후 **DATABASE**를 마우스 오른쪽 버튼으로 누르고 드롭 다운 목록에서 **SQL 스크립트 실행**을 선택하십시오.

   SQL 스크립트 실행 창이 열립니다. 이 창을 사용하면 SQL 문에서 잘라내서 붙여넣거나 SQL 스크립트를 열 수 있습니다. 연결/JDBC 설정 옵션을 사용하여 기본 스키마를 설정할 수 있습니다.

전개 코드 생성

이 절에서는 EJB 전개 도구를 사용하여 EJB 1.1 Export JAR 파일에 포함된 엔터프라이즈 bean용 전개 코드를 생성하는 방법에 대해 설명합니다. 이 도구는 WebSphere Application Server와 함께 제공됩니다.

전개 코드를 생성하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server의 명령 프롬프트에서 다음 디렉토리로 이동하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp
   - Solaris /opt/WebSphere/CommerceServer/temp
   - Linux /opt/WebSphere/CommerceServer/temp
2. 400 대상 WebSphere Commerce Server의 명령 프롬프트에서 다음 명령을 입력하십시오.

   ```
   STRQSH
   cd /QIBM/UserData/WebCommerce/instances/instanceName/temp
   ```
3. 다음 명령을 입력해서 시스템 경로에 도구를 임시로 추가하십시오.
   - Windows PATH=*drive*:\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=*ClassPathOfDepJARFiles*

     여기서, *ClassPathOfDepJARFiles*는 종속 JAR 파일의 클래스 경로입니다. 기존의 WebSphere Commerce 엔터프라이즈 bean을 수정할 경우, 종속 JAR 파일의 클래스 경로에 wcsejsclient.jar, wcsejbimpl.jar 및 xml4j.jar 파일이 있어야 합니다. 다음은 기존 WebSphere Commerce 엔터프라이즈 bean이 수정된 경우에 대한 클래스 경로의 예입니다.

     ```
     CP=/QIBM/UserData/WebCommerce/instances/instanceName/temp/lib/
     wcsejsclient.jar:
     /QIBM/UserData/WebASAdv4/WAS_AdminInstanceName/installedApps/
     WC_Enterprise_App_instanceName.ear/lib/wcsejbimpl.jar:
     /QIBM/UserData/WebASAdv4/WAS_AdminInstanceName/installedApps/
     WC_Enterprise_App_instanceName.ear/lib/xml4j.jar
     ```

주: 앞에 있는 클래스 경로 예에서 행 바꾸기는 단지 보기 쉽도록 하기 위한 것입니다. 클래스 경로 정보를 한 행에 입력해야 합니다.

4. Solaris Linux 명령행 인터페이스에서 문자 제한을 초과하지 않으려면, 스크립트를 사용하여 ejbdeploy 명령을 호출하면 됩니다. 스크립트를 작성하고 명령을 호출하려면 다음을 수행하십시오.

   a. 다음 디렉토리로 탐색하십시오.

      ```
      /opt/WebSphere/AppServer/bin
      ```

   b. 새 파일을 작성하여 스크립트에 적절한 이름을 지정하십시오. 예를 들어 파일 이름으로 myejbd.sh를 지정하십시오.

   c. 다음 정보를 myejbd.sh 파일에 포함시키십시오.

      ```
      #!/bin/sh
      ./ejbdeploy.sh EJBGroupJARFile WorkingDir OutputJARFile
      -nowarn -keep -35 -cp ClassPathOfDepJARFiles
      ```

      여기서 변수는 5단계에서와 같은 정의를 갖습니다(그 단계를 수행하지 않아도 됩니다). 다음은 변수의 값이 포함된 스크립트 파일에 포함할 사항의 예입니다.

      ```
      #!/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"
      ```

      주: ./ejbdeploy.sh 명령 다음의 모든 행 바꾸기는 단지 보기 쉽도록 하기 위한 것입니다. 파일에서는 ./ejbdeploy.sh 명령 다음에 행 바꾸기가 있어서는 안됩니다.

      스크립트를 저장하고 실행 가능하도록 만드십시오.

   d. 다음을 입력하여 스크립트를 호출하십시오.

      ```
      ./myejbd.sh
      ```

5. Windows AIX 400 다음과 같이 ejbdeploy 명령을 입력하십시오.

   Windows 400

```
ejbdeploy EJBGroupJARFile_DT WorkingDir OutputJARFile
-nowarn -keep -35 -cp ClassPathOfDepJARFiles
```

AIX

```
/usr/WebSphere/AppServer/bin/ejbdeploy.sh EJBGroupJARFile_DT
WorkingDir OutputJARFile -nowarn -keep -35
-cp ClassPathOfDepJARFiles
```

여기서

- *EJBGroupJARFile*_DT는 EJB 그룹용 JAR 파일 이름입니다. 예를 들어 WCSUser EJB 그룹에서 bean을 수정한 경우, Windows 기반 플랫폼에서의 사용 예는 다음과 같습니다.

  ```
  drive:\WebSphere\CommerceServer\temp\CustomizedWCSUser_DT.jar
  ```

  400 iSeries 플랫폼에 대한 사용 예는 다음과 같습니다.

  ```
  /QIBM/UserData/WebCommerce/instances/instanceName/temp/
  CustomizedWCSUser_DT.jar
  ```

- *WorkingDir*은 코드 생성에 필요한 임시 파일이 저장되는 디렉토리의 이름입니다.
- *OutputJARFile*은 출력된 JAR 파일의 완전한 이름입니다. 예를 들어 CustomizedWCSUserDeployed.jar를 입력하십시오.
- -nowarn은 경고 및 정보 메시지를 억제하기 위한 선택적 매개변수입니다.
- -keep은 ejbdeploy 명령을 실행한 후에 작업 디렉토리를 유지하기 위한 선택적 매개변수입니다.
- -35는 WebSphere Application Server 버전 3.5와 함께 제공되는 EJB 전개 도구에 사용되는 CMP 엔티티 bean용 하향식 맵핑 규칙을 사용하는 필수 매개변수입니다.
- -cp *ClassPathOfDepJARFiles*는 종속 JAR 파일의 클래스 경로입니다. 기존의 WebSphere Commerce 엔터프라이즈 bean을 수정한 경우, 종속 JAR 파일의 클래스 경로에 wcsejsclient.jar, wcsejbimpl.jar 및 xml4j.jar 파일이 있어야 합니다. 다음은 기존 WebSphere Commerce 엔터프라이즈 bean이 수정된 경우에 대한 클래스 경로의 예입니다.

```
"drive:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
drive:\WebSphere\AppServer\installedApps\
WC_Enterprise_App_instanceName.ear \lib\wcsejbimpl.jar;
drive:\WebSphere\AppServer\installedApps\
WC_Enterprise_App_instanceName.ear\lib\xml4j.jar;"
```

주: 400 클래스 경로 값으로 -cp $CP를 입력하십시오. 여기서 CP는 적절한 클래스 경로 항목으로 이전에 정의되었습니다. 또한 따옴표는 필요하지 않습니다.

엔티티 bean의 트랜잭션 분리 레벨 수정

이 절에서는 modifyIsolationLevel 명령행 유틸리티를 사용하여 엔티티 bean의 트랜잭션 분리 레벨을 적절한 데이터베이스 유형 레벨로 설정하는 방법에 대해 설명합니다.

modifyIsolationLevel 명령을 실행하려면, 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server의 다음 디렉토리로 이동하는 명령 프롬프트를 사용하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\bin
   - AIX /usr/WebSphere/CommerceServer/bin
   - Solaris /opt/WebSphere/CommerceServer/bin
   - Linux /opt/WebSphere/CommerceServer/bin
2. 400 다음 명령을 입력하십시오.

   ```
   STRQSH
   cd /QIBM/ProdData/WebCommerce/bin
   ```
3. 다음과 같은 일반 구문이 들어 있는 modifyIsolationLevel 명령을 발행해야 합니다.

   Windows AIX 400

   ```
   modifyIsolationLevel -jarFile jar_file_name.jar
      -logFile log_file_name -dbType db_type
   ```

   Solaris Linux

```
./modifyIsolationLevel.sh -jarFile jar_file_name.jar
   -logFile log_file_name -dbType db_type
```

여기서

- *jar_file_name.jar*는 사용자에 맞게 정의된 코드를 포함하는 JAR 파일의 이름입니다.
- *log_file_name*은 정보를 기록해야 하는 완전한 파일 이름입니다.
- *db_type*은 사용 중인 데이터베이스의 유형입니다. DB2 또는 ORACLE을 입력하십시오.

다음은 Windows 플랫폼에서 사용할 수 있도록 모든 값이 지정된 `modifyIsolationLevel` 명령의 예입니다.

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

400 다음은 iSeries에서 사용할 수 있도록 모든 값이 지정된 `modifyIsolationLevel` 명령의 예입니다.

```
modifyIsolationLevel -jarFile
   /QIBM/UserData/WebCommerce/instances/instanceName/temp/
   CustomizedWCSUserDeployed.jar -logFile /QIBM/UserData/WebCommerce/
   instances/instanceName/logs/output.log -dbType DB2
```

이전 예에서 행 바꾸기는 단지 보기 쉽도록 하기 위한 것입니다.

주: 매개변수 이름은 대소문자가 구분됩니다. 즉, `jarFile`은 `jarfile`과 다릅니다. 매개변수 이름을 정확하게 입력했는지 확인하십시오.

예외가 명령창에 표시되지 않은 경우 명령이 정상적으로 실행된 것입니다. 완료 후, 전개된 JAR 파일에서 시간소인이 변경되었음에 유의하십시오.

현재 WebSphere Commerce 엔터프라이즈 응용프로그램 반출

이 절에서는 WebSphere Application Server 관리 콘솔을 사용하여 현재 WebSphere Commerce 엔터프라이즈 응용프로그램을 반출하는 방법에 대해 설명합니다.

WebSphere Application Server에서 현재 엔터프라이즈 응용프로그램을 반출하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server에 다음 디렉토리가 있는지 확인하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working
   - 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working

   아직 이 디렉토리가 없는 경우, 지금 작성하십시오.

   주: AIX Solaris Linux *WebSphere Commerce 설치 안내서*의 *"사후 설치 스크립트 실행"* 절에서 단계를 수행할 때 작성된 /working 디렉토리에 대한 권한이 사용자로 설정되어 있는지 확인하십시오.

   400 /QIBM/UserData/WebCommerce/instances/*instanceName* 및 /QIBM/UserData/WebCommerce/instances/*instanceName*/working 디렉토리에 대한 권한에 사용자 QEJB가 포함되어 있는지 확인하십시오. 데이터 권한을 *RWX로 설정하여 이 사용자를 두 디렉토리 모두에 추가하십시오.
2. WebSphere Application Server 관리 콘솔을 여십시오.
3. **WebSphere 관리 도메인**을 펼치십시오.
4. **엔터프라이즈 응용프로그램**을 펼치십시오.
5. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **데모** 응용프로그램을 펼쳐 **반출 응용프로그램**을 선택하십시오.
6. **반출 디렉토리** 필드에 다음을 입력하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working

- Solaris /opt/WebSphere/CommerceServer/working
- Linux /opt/WebSphere/CommerceServer/working
- 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working

이는 모든 자원을 포함한 전체 응용프로그램을 WC_Enterprise_App_*instanceName*.ear 파일로 반출합니다(여기서 *instanceName*은 WebSphere Commerce 인스턴스 이름입니다).

엔터프라이즈 bean의 구성 정보 반출

이 절에서는 XMLConfig 명령행 유틸리티의 -export 옵션을 사용하여 기존 엔터프라이즈 응용프로그램에 포함되어 있는 엔터프라이즈 bean의 구성 정보를 반출하는 방법에 대해 설명합니다.

기존 응용프로그램의 bean 정보를 반출한 후, 응용프로그램에 추가하는 새 bean에 대한 정보를 수동으로 추가해야 합니다.

이 구성 정보를 반출하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server의 다음 디렉토리에서
   - Windows *drive*:\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

   was.export.app.xml 파일을 대상 WebSphere Commerce Server의 다음 디렉토리로 복사하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working

- Linux /opt/WebSphere/CommerceServer/working
- 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working
  여기서,
  - *instanceName*은 WebSphere Commerce 인스턴스의 이름입니다.

2. Windows AIX Solaris Linux 텍스트 편집기에서 was.export.app.xml 파일을 여십시오. 이 파일은 다음과 함께 $Enterprise_Application_Name$의 모든 발생을 대체합니다.

   WebSphere Commerce Enterprise Application - *instanceName*

   여기서 *instanceName*은 WebSphere Commerce 인스턴스 이름(예: demo)입니다. 이 파일을 저장하십시오.

   주: 삽입하는 값은 WebSphere Application Server 관리 콘솔에 표시되는 인스턴스 정보와 일치해야 합니다.

3. 400 텍스트 편집기의 was.export.app.xml 파일을 여십시오. 이 파일은 다음과 함께 $Enterprise_Application_Name$의 모든 발생을 대체합니다.

   instanceName - WebSphere Commerce Enterprise Application

   여기서, *instanceName*은 WebSphere Commerce 인스턴스 이름(예: demo)입니다. 이 파일을 저장하십시오.

   주: 삽입하는 값은 WebSphere Application Server 관리 콘솔에 표시되는 인스턴스 정보와 일치해야 합니다.

4. Windows AIX Solaris Linux 명령 프롬프트에서 다음 디렉토리를 탐색하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working

5. 400 명령 프롬프트에서 다음을 입력하십시오.

```
STRQSH
PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
cd /QIBM/UserData/WebCommerce/instances/instanceName/working
```

6. Windows AIX Solaris Linux 부분적으로 반출하려면 아래 명령을 입력하여 XMLConfig 도구를 호출하십시오.

Windows

```
xmlConfig -export OutputFile.xml -partial was.export.app.xml
   -adminNodeName wasHostName
```

AIX

```
/usr/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
   -partial was.export.app.xml -adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
```

Solaris Linux

```
/opt/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
   -partial was.export.app.xml -adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
```

여기서

- *wasHostName*은 현재 엔터프라이즈 응용프로그램을 포함하는 WebSphere Application Server의 노드 이름입니다.
- `OutputFile.xml`은 이 명령 실행의 결과로 작성된 파일 이름이며 `was.export.app.xml`은 2단계에서 수정한 파일입니다.
- *wasAdminPort*는 WebSphere Application Server 관리자 포트입니다.

7. 400 다음 명령을 입력해서 부분 반출을 수행하려면 XMLConfig 도구를 호출하십시오.

```
xmlConfig -export OutputFile.xml -partial was.export.app.xml
-adminNodeName wasHostName -nameServiceHost wasHostName
-nameServicePort wasAdminPort -instance wasInstanceName
```

여기서,

- *wasHostName*은 현재 엔터프라이즈 응용프로그램을 포함하는 WebSphere Application Server의 노드 이름입니다.

  주: *wasHostName*의 값은 대소문자가 구분되며 TCP/IP 구성에 있는 값과 일치해야 합니다(명령행 프로세서를 사용해서 CFGTCP, 옵션 12에 액세스하여 호스트 이름을 검증하십시오).

- `OutputFile.xml`은 이 명령 실행의 결과로 작성된 파일 이름이며, `was.export.app.xml`은 3단계에서 수정한 파일입니다.
- *wasAdminPort*는 WebSphere Application Server 관리자 포트입니다.
- *wasInstanceName*은 WebSphere Application Server 인스턴스 이름입니다.

현재 엔터프라이즈 응용프로그램에 포함되어 있는 각 bean의 구성 정보를 반출한 후, 응용프로그램에 추가할 각 엔터프라이즈 bean을 설명하는 새 스탠자에 추가해야 합니다. 예를 들어 “Bonus”라는 새 항목을 갖고 있는 경우, 이 보너스 bean을 설명하는 스탠자를 추가해야 합니다. 또한 구성 파일의 변수를 바꾸어 엔터프라이즈 응용프로그램에 정확한 .ear 파일 이름을 지정해야 합니다.

`OutputFile.xml` 파일을 갱신하려면 다음을 수행하십시오.

1. 대상 WebSphere Commerce Server의 다음 디렉토리를 탐색하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - Linux /opt/WebSphere/CommerceServer/working
   - 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working
2. 텍스트 편집기에서 `OutputFile.xml` 파일을 여십시오.
3. `<ear-file-name>` 태그의 위치를 지정하고 다음으로 값을 바꾸십시오.
   - Windows *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear

- AIX /usr/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
- Solaris /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
- Linux /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
- 400 /QIBM/UserData/WebCommerce/instances/*instanceName*/working/WC_Enterprise_App_*instanceName*.ear

4. 또한 엔터프라이즈 응용프로그램에 추가하는 각 새 bean의 새 스탠자에 추가해야 합니다. 다음 예는 “Bonus”라는 새 bean을 추가하는 방법을 보여줍니다.

Windows AIX Solaris Linux

```
<ejb-module name="yourEJBGroup">
   <jar-file>yourDeployedJarFile.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         WebSphere Commerce Server - instanceName/
      </application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/WebSphere Commerce DB2 DataSource instanceName
   </jndi-name>
         <default-user>user</default-user>
         <default-password>password</default-password>
      </data-source>
      <enterprise-bean-binding name="BeanBindingName">
         <jndi-name>instanceNameJNDINameOfBean</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

400

```
<ejb-module name="yourEJBGroup">
   <jar-file>yourDeployedJarFile.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         instanceName - WebSphere Commerce Server/
      </application-server-full-name>
```

```
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/instanceName WebSphere Commerce DB2 DataSource
   </jndi-name>
         <default-user>user</default-user>
         <default-password>password</default-password>
      </data-source>
      <enterprise-bean-binding name="BeanBindingName">
         <jndi-name>instanceNameJNDINameOfBean</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

여기서

- *yourEJBGroup*은 엔터프라이즈 응용프로그램에 추가 중인 bean을 포함하는 EJB 그룹의 이름입니다.
- *yourDeployedJarFile*은 EJB 그룹의 전개 코드를 포함하는 JAR 파일 이름입니다.
- *instanceName*은 코드를 전개 중인 WebSphere Commerce 인스턴스 이름입니다.
- *user*는 데이터베이스 사용자 이름입니다.
- *password*는 데이터베이스 사용자의 암호입니다.
- *BeanBindingName*은 엔터프라이즈 bean의 바인딩 이름입니다. 예를 들어, Bonus라는 bean의 경우, Bonus_Binding입니다.
- *instanceNameJNDINameOfBean*은 WebSphere Commerce 인스턴스를 앞에 첨부한 엔터프라이즈 bean의 JNDI 이름입니다. 이 JNDI 이름은 엔터프라이즈 bean 이름과 정확히 일치해야 합니다. 예제 값은 democom/ibm/commerce/sample/objects/Bonus입니다. 이 예에서 "demo"는 인스턴스 이름이고 "com/ibm/commerce/sample/objects/Bonus"는 엔터프라이즈 bean의 JNDI 이름입니다. 이 값은 한 행에 입력해야 합니다.

  VisualAge for Java 또는 응용프로그램 어셈블리 도구를 사용하여 JNDI 이름을 검증할 수 있습니다.

  VisualAge for Java를 사용하여 JNDI 이름을 검증하려면 다음을 수행하십시오.

a. bean을 마우스 오른쪽 버튼으로 누른 후 **특성**을 선택하십시오.
   JNDI 이름이 표시됩니다.

JNDI 이름은 응용프로그램 어셈블리 도구를 사용하여 검증할 수 있지만, 이미 새 엔터프라이즈 bean을 엔터프라이즈 응용프로그램으로 어셈블해야 합니다. 응용프로그램 어셈블리 도구는 WebSphere Application Server 관리 콘솔에 있는 도구 메뉴에서 액세스됩니다.

응용프로그램 어셈블리 도구를 사용하여 JNDI 이름을 검증하려면 다음을 수행하십시오.

a. bean을 포함하는 .ear 파일을 여십시오.
b. bean을 포함하는 EJB 모듈을 펼치십시오.
c. bean을 선택하십시오.
d. **바인딩** 탭을 누르십시오.
   JNDI 이름이 표시됩니다.

이 메소드 중 하나를 사용하면 JNDI 이름이 표시되지만, XML 파일에 정보 추가시 계속 WebSphere Commerce 인스턴스 이름을 앞에 첨부해야 함에 유의하십시오.

주:

a. 이전 스탠자에서의 행 바꾸기는 단지 보기 쉽도록 하기 위한 것입니다.
b. **$hostName$** 값이 현재 관리 노드 서버 이름과 일치하는지 확인하십시오. 또한, 이 행에 캐리지 리턴 문자가 없는지 확인하십시오.
c. <application-server-full-name> 스펙은 두 줄 이상 간격을 넓힐 수 없습니다
d. Oracle 데이터베이스를 사용 중인 경우, 데이터 소스 정보를 수정해야 합니다. 앞의 코드 부분에서 다음 행을

```
<jndi-name>jdbc/WebSphere Commerce DB2 DataSource instanceName
   </jndi-name>
```

다음과 같이 변경하십시오.

```
<jndi-name>jdbc/WebSphere Commerce Oracle DataSource instanceName
   </jndi-name>
```

5. 수정된 WebSphere Commerce 엔티티 bean을 전개할 경우, 수정된 bean에 전개된 코드를 포함하는 JAR 파일의 이름을 반영하도록 bean에 대한 스탠자를 갱신해야 합니다.
6. `OutputFile.xml` 파일을 저장하십시오.

엔터프라이즈 응용프로그램으로 새 엔터프라이즈 bean 어셈블링

이 절에서는 응용프로그램 어셈블리 도구를 사용하여 기존 엔터프라이즈 응용프로그램으로 새 엔터프라이즈 bean을 어셈블하는 방법에 대해 설명합니다.

400 응용프로그램 어셈블리 도구는 플랫폼에서 실행되므로, 완전한 경로 이름을 입력하도록 프롬프트할 때 iSeries IFS에 맵핑한 드라이브를 언급해야 합니다. 이는 *iSeries_drive*로 언급됩니다.

AIX Solaris Linux 신규 또는 수정된 `.ear` 파일을 저장할 때 메모리 부족을 피하려면 `assembly.sh` 파일에서 메모리 힙 *크기* 값을 증가시키는 것이 좋습니다.

이 파일은 다음 디렉토리에 있습니다.

- AIX `/usr/WebSphere/AppServer/bin`
- Solaris `/opt/WebSphere/AppServer/bin`
- Linux `/opt/WebSphere/AppServer/bin`

메모리 힙 *크기*를 증가하려면 다음 행을 수정하십시오.

```
$JAVA_HOME/jre/bin/java
```

그러면 다음과 같이 나타납니다.

```
$JAVA_HOME/jre/bin/java -mx512M
```

이 단계에서 현재 WebSphere Commerce 엔터프라이즈 응용프로그램 반출에서 작성한 엔터프라이즈 응용프로그램의 `.ear` 파일을 응용프로그램 어셈블러 도구에서 여십시오. 해당 도구에서 파일이 열리면, 다음 태스크를 수행하여 엔터프라이즈 응용프로그램에 새 엔티티 bean을 추가하십시오.

1. 새 항목 bean을 포함하고 있는 EJB 그룹을 반입하십시오. 새 EJB 그룹의 JAR 파일은 엔터프라이즈 응용프로그램의 EJB 모듈 섹션에 저장됩니다.
2. 새 항목 bean의 클래스 경로를 구현 JAR 파일을 포함하도록 설정하십시오.
3. 구현 JAR 파일을 응용프로그램에 추가하십시오. 이 JAR 파일은 엔터프라이즈 응용프로그램의 파일 섹션에 저장됩니다.
4. 새 항목 bean에 포함된 메소드에 WebSphere Application Server 보안을 설정하십시오.

엔터프라이즈 응용프로그램에 새 EJB 그룹을 어셈블링하려면 다음을 수행하십시오.

1. Windows AIX Solaris Linux 대상 WebSphere Commerce Server에서 다음을 수행하여 현재 엔터프라이즈 응용프로그램을 백업하십시오.
   a. 명령 프롬프트에서 다음 디렉토리를 선택하십시오.
      - Windows *drive*:\WebSphere\CommerceServer\working
      - AIX /usr/WebSphere/CommerceServer/working
      - Solaris /opt/WebSphere/CommerceServer/working
      - Linux /opt/WebSphere/CommerceServer/working

   b. 기존 WC_Enterprise_App_*instanceName*.ear 파일 사본을 작성하고 이름을 WC_Enterprise_App_*instanceName*.ear.bak으로 지정하십시오.
2. 400 다음을 수행하여 현재 엔터프라이즈 응용프로그램을 백업하십시오.
   a. 명령 프롬프트에서 다음을 입력하십시오.

      ```
      STRQSH
      cd /QIBM/UserData/WebCommerce/instances/instanceName/working
      cp WC_Enterprise_App_instanceName.ear
      WC_Enterprise_App_instanceName.ear.bak
      ```

   b. 로컬 클라이언트 시스템과 WebSphere Application Server를 실행 중인 iSeries 시스템 사이의 데이터 전송으로 인한 불필요한 장시간 대기를 피하려면 로컬 클라이언트 시스템에 다음 디렉토리를 작성한 후 WC_Enterprise_App_*instanceName*.ear 파일을 *drive*:\WebSphere\CommerceServer\working 디렉토리에 복사하십시

오.

여기서 *drive*는 로컬 드라이브입니다.

주: *drive*:\WebSphere\CommerceServer\working 디렉토리가 로컬 시스템에 존재하지 않을 경우, 지금 작성하십시오.

3. WebSphere Application Server 관리 콘솔을 여십시오.
4. **도구** 메뉴에서 **응용프로그램 어셈블리 도구**를 선택하십시오.
5. 환영 창이 열리면, **취소**를 선택하여 창을 종료하십시오.
6. 다음을 수행해서 작업을 하려면 엔터프라이즈 응용프로그램을 여십시오.
   a. **파일** 메뉴에서 **열기**를 선택하십시오.
   b. **파일 이름** 필드에 다음을 입력하십시오.
      - Windows *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear
      - AIX /usr/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Solaris /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Linux /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - 400 *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear

      열기를 누르십시오. 다음 단계를 계속하기 전에 응용프로그램이 열릴 때까지 잠시 대기하십시오. 이 작업에는 몇 분이 걸립니다.
7. **EJB Modules**을 마우스 오른쪽 버튼으로 누른 후, **반입**을 선택하십시오.
8. **파일 이름** 필드에 다음을 입력하십시오.
   - Windows *drive*:\WebSphere\CommerceServer\temp*yourDeployedJarFile*.jar

- AIX `/usr/WebSphere/CommerceServer/temp/`*`yourDeployedJarFile`*`.jar`
- Solaris `/opt/WebSphere/CommerceServer/temp/`*`yourDeployedJarFile`*`.jar`
- Linux `/opt/WebSphere/CommerceServer/temp/`*`yourDeployedJarFile`*`.jar`
- 400 *`iSeries_drive`*`:\QIBM\UserData\WebCommerce\instances\`*`instanceName`*`\temp\`*`yourDeployedJarFile`*`.jar`

여기서

- *yourDeployedJarFile*은 EJB 그룹의 전개 코드를 포함하고 있는 JAR 파일 이름입니다.
- *iSeries_drive*는 iSeries IFS에 맵핑되는 로컬 드라이브입니다.

열기를 누른 후, 값 확인 창에서 **확인**을 누르십시오.

9. *yourDeployedJarFile*.jar 파일이 반입되면, *yourEJBGroup* EJB 그룹(여기서 *yourEJBGroup*은 EJB 그룹 이름)으로 화면이동하여 이 그룹을 선택하십시오.
   해당 그룹에 대한 정보는 오른쪽 분할창에 표시됩니다.
10. 새 엔터프라이즈 bean용 클래스 경로 필드에서 종속 JAR 파일을 입력하십시오. 예를 들어 아래에 표시된 것처럼 해당 구현 JAR 파일 및 WebSphere Commerce 엔티티 bean의 구현 JAR 파일을 입력할 수 있습니다.

    `lib/`*`yourImplJarFile`*`.jar lib/wcsejbimpl.jar`
11. **적용**을 누르십시오.
12. 다음을 수행하여 EJB 그룹용 구현 JAR 파일을 응용프로그램에 추가하십시오.
    a. 엔터프라이즈 응용프로그램의 **파일** 노드를 마우스 오른쪽 버튼으로 누른 후 **파일 추가**를 선택하십시오(엔터프라이즈 응용프로그램의 **파일** 노드는 계층 구조 맨 아래 근처에 있습니다). 엔터프라이즈 응용프로그램 내의 구성요소에 대한 다른 파일 노드가 있지만, 전체 응용프로그램에 대한 파일 노드를 선택해야 함에 유의하십시오.

b. 파일 추가 창에서 **찾아보기**를 누르십시오.

c. 다음 디렉토리로 탐색하십시오.

- Windows *drive*:\WebSphere\CommerceServer\temp
- AIX /usr/WebSphere/CommerceServer/temp
- Solaris /opt/WebSphere/CommerceServer/temp
- Linux /opt/WebSphere/CommerceServer/temp
- 400 *iSeries_drive*:\QIBM\UserData\WebCommerce \instances\ *instanceName*\temp

d. 강조표시된 디렉토리에서 **선택**을 누르십시오.

e. 파일 추가 창으로 리턴하십시오. 임시 디렉토리의 컨텐츠가 표시됩니다. lib 디렉토리를 강조표시하십시오.
lib의 구성요소는 오른쪽 분할창에 표시됩니다.

f. 오른쪽 분할창에서 *yourImplJarFile* 파일을 선택한 후 **추가**를 누르십시오. 그러면 파일이 선택된 파일 분할창에 표시됩니다.

g. **확인**을 누르십시오.

13. 다음을 수행하여 엔티티 bean에 보안을 구성하십시오.

a. EJB 모듈 노드를 펼쳐 *YourEJBGroup* 노드 위치를 지정하고 펼치십시오.

b. **엔티티 Bean**을 펼치십시오.

c. *yourEntityBean*을 펼치십시오. 여기서 *yourEntityBean*은 엔티티 bean 이름입니다.

d. **메소드 확장자**를 누르고 오른쪽 분할창에서 다음을 수행하십시오.

1) **고급** 탭을 선택하십시오.

2) **보안 동일성**이 선택되었는지 확인하십시오.

3) 개별 메소드에 대해서 **EJB 서버의 동일성 사용**이 선택되었는지 확인하십시오.

4) **적용**을 누르십시오(수정한 경우).

e. 왼쪽 탐색 분할창에서 *YourEJBGroup* 아래의 **보안 역할**에서 마우스 오른쪽 버튼을 눌러 **새로 만들기**를 선택한 후 다음을 수행하십시오.

   1) **이름** 필드에서 WCSecurityRole을 입력한 후, **적용**을 누르십시오. 이 역할이 이미 존재하는 경우, 이 단계를 수행하지 않아도 됨에 유의하십시오.

f. 왼쪽 탐색 분할창에서 *YourEJBGroup* EJB 그룹 아래의 **메소드 허용**을 마우스 오른쪽 버튼으로 눌러 **새로 만들기**를 선택한 후, 다음을 수행하십시오.

   1) **메소드 허용 이름** 필드에 WCMethodPermission을 입력하십시오.
   2) **메소드** 선택 영역에서 **추가**를 누르십시오.
      메소드 추가 창이 열립니다.
   3) *yourDeployedJarFile*.**jar**와 **Bonus**를 차례대로 펼친 후 메소드의 각 **홈** 및 **원격** 목록을 펼치십시오.
   4) Shift 키를 누른 상태에서 모든 홈 메소드를 선택한 후 **확인**을 누르십시오.
   5) 메소드 선택 처리를 반복하여 원격 메소드도 추가하십시오(원격 메소드가 존재할 경우).
   6) 역할 선택 영역에서 **추가**를 누르고 WCSecurityRole을 선택한 후 **확인**을 누르십시오.
   7) **적용**을 누르십시오.

14. **파일** 메뉴에서 **저장**을 선택하십시오.
15. 응용프로그램 어셈블리 도구를 종료하십시오.
16. 400 새로 수정된 WC_Enterprise_App_*instanceName*.ear 파일을 로컬 시스템에서 WebSphere Application Server를 실행 중인 iSeries 시스템으로 복사하십시오. 즉, 다음 디렉토리에서 파일을 복사하십시오.

    drive:\WebSphere\CommerceServer\working

    다음 디렉토리로 복사하십시오.

    iSeries_drive:\QIBM\UserData\WebCommerce\instances*instanceName*\working

    여기서 *iSeries_drive*는 iSeries IFS에 맵핑된 드라이브 이름입니다.

이 단계를 완료하면, 새로운 비즈니스 로직 뿐만 아니라 이전의 모든 로직을 포함하는 새 엔터프라이즈 응용프로그램이 작성됩니다. 새로 수정된 WC_Enterprise_App_*instanceName*.ear 파일에 이 모든 것이 포함됩니다.

엔터프라이즈 응용프로그램으로 수정 엔터프라이즈 bean 어셈블링

이 절에서는 응용프로그램 어셈블리 도구를 사용하여 엔터프라이즈 응용프로그램으로 수정 WebSphere Commerce 엔터프라이즈를 어셈블하는 방법에 대해 설명합니다.

이 단계에서 응용프로그램 어셈블러 도구의 엔터프라이즈 응용프로그램을 여십시오. 해당 도구에서 열리면 다음을 수행하여 수정 WebSphere Commerce 엔터프라이즈 bean을 엔터프라이즈 응용프로그램에 포함할 수 있습니다.

1. 수정한 기존 EJB 그룹 버전의 클래스 경로 사본을 작성하십시오.
2. 수정한 EJB 그룹의 기존 버전을 제거하십시오.
3. 수정한 EJB 그룹의 새 버전을 반입하십시오. 새 EJB 그룹에 대한 JAR 파일은 엔터프라이즈 응용프로그램의 EJB 모듈 섹션 내에 저장됩니다.
4. 수정 EJB 그룹의 클래스 경로를 설정하십시오.
5. 수정 엔티티 bean에 포함된 메소드에 WebSphere Application Server 보안을 설정하십시오.

AIX Solaris Linux 신규 또는 수정된 .ear 파일을 저장할 때 메모리 부족을 피하려면 assembly.sh 파일에서 메모리 힙 크기 값을 증가시키는 것이 좋습니다.

이 파일은 다음 디렉토리에 있습니다.

- AIX /usr/WebSphere/AppServer/bin
- Solaris /opt/WebSphere/AppServer/bin
- Linux /opt/WebSphere/AppServer/bin

메모리 힙 크기를 증가하려면 다음 행을 수정하십시오.

```
$JAVA_HOME/jre/bin/java
```

그러면 다음과 같이 나타납니다.

```
$JAVA_HOME/jre/bin/java -mx512M
```

수정 EJB 그룹을 엔터프라이즈 응용프로그램으로 어셈블하려면 다음을 수행하십시오.

1. Windows AIX Solaris Linux 대상 WebSphere Commerce Server에서 다음을 수행하여 현재 엔터프라이즈 응용프로그램을 백업하십시오.
   a. 명령 프롬프트에서 다음 디렉토리를 선택하십시오.
      - Windows *drive*:\WebSphere\CommerceServer\working
      - AIX /usr/WebSphere/CommerceServer/working
      - Solaris /opt/WebSphere/CommerceServer/working
      - Linux /opt/WebSphere/CommerceServer/working
   b. 기존 WC_Enterprise_App_*instanceName*.ear 파일 사본을 작성하고 이름을 WC_Enterprise_App_*instanceName*.ear.bak으로 지정하십시오.
2. 400 다음을 수행하여 현재 엔터프라이즈 응용프로그램을 백업하십시오.
   a. 명령 프롬프트에서 다음을 입력하십시오.

```
STRQSH
cd /QIBM/UserData/WebCommerce/instances/instanceName/working
cp WC_Enterprise_App_instanceName.ear
WC_Enterprise_App_instanceName.ear.bak
```

b. 로컬 클라이언트 시스템과 WebSphere Application Server를 실행 중인 iSeries 시스템 사이의 데이터 전송으로 인한 불필요한 장시간 대기를 피하려면 로컬 클라이언트 시스템에 다음 디렉토리를 작성한 후 WC_Enterprise_App_*instanceName*.ear 파일을 *drive*:\WebSphere\CommerceServer\working 디렉토리에 복사하십시오.
여기서 *drive*는 로컬 드라이브입니다.

주: *drive*:\WebSphere\CommerceServer\working 디렉토리가 로컬 시스템에 존재하지 않을 경우, 지금 작성하십시오.

3. WebSphere Application Server 관리 콘솔을 여십시오.
4. **도구** 메뉴에서 **응용프로그램 어셈블리 도구**를 선택하십시오.
5. 다음을 수행해서 작업을 하려면 엔터프라이즈 응용프로그램을 여십시오.
   a. **파일** 메뉴에서 **열기**를 선택하십시오.
   b. **파일 이름** 필드에 다음을 입력하십시오.
      - Windows *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear
      - AIX /usr/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Solaris /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - Linux /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*instanceName*.ear
      - 400 *drive*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*instanceName*.ear

      열기를 누르십시오. 다음 단계를 계속하기 전에 응용프로그램이 열릴 때까지 잠시 대기하십시오. 이 작업에는 몇 분이 걸립니다.

6. **EJB 모듈**을 누르십시오. 오른쪽 분할창이 엔터프라이즈 응용프로그램 내의 EJB 모듈을 표시합니다.
7. 수정한 EJB 그룹용 EJB 모듈을 누르십시오. 예를 들어, WCSUser EJB 그룹의 bean을 수정한 경우, **WCSUser**를 누르십시오.
8. 일반 탭을 눌러 클래스 경로 정보를 보십시오. 이 기존 클래스 경로 정보를 텍스트 파일(예: `WCSUser_path.txt`)로 복사하십시오.
9. EJB 모듈을 마우스 오른쪽 버튼으로 누른 후 **삭제**를 선택하십시오.
10. **EJB Modules**을 마우스 오른쪽 버튼으로 누른 후, **반입**을 선택하십시오.
11. **파일 이름** 필드에 다음을 입력하십시오.
    - Windows *drive*`:\WebSphere\CommerceServer\temp\Cust_`*EJBGroupName*`-ejb.jar`
    - AIX `/usr/WebSphere/CommerceServer/temp/Cust_`*EJBGroupName*`-ejb.jar`
    - Solaris `/opt/WebSphere/CommerceServer/temp/Cust_`*EJBGroupName*`-ejb.jar`
    - Linux `/opt/WebSphere/CommerceServer/temp/Cust_`*EJBGroupName*`-ejb.jar`
    - 400 *iSeries_drive*`:\QIBM\UserData\WebCommerce\instances\`*instanceName*`\temp\Cust_`*EJBGroupName*`-ejb.jar`

    그런 다음, **열기**를 누르십시오. 확인값 창에서 **확인**을 누르십시오.
12. *EJBGroupJARFile*`.jar` 파일이 반입되면, 수정 EJB 그룹으로 화면이동하여 이 그룹을 선택하십시오.
    이 그룹에 대한 정보는 오른쪽 분할창에 표시됩니다.
13. 이전 EJB 그룹 버전에 대한 클래스 경로 정보를 포함하고 있는 텍스트 파일을 여십시오. 클래스 경로를 선택해서 복사하십시오.
14. 수정된 EJB 그룹의 **클래스 경로** 필드에서 이 클래스 경로 정보로 붙여넣으십시오.
15. **적용**을 누르십시오.
16. **파일** 메뉴에서 **종료**를 선택하십시오.

17. 파일이 닫힐 때까지 기다린 후 **파일** 메뉴에서 **열기**를 선택하고 WC_Enterprise_App_*instanceName*.ear 파일을 다시 여십시오.
18. 다음을 수행하여 수정 bean의 보안을 구성하십시오.
    a. EJB 모듈 노드를 펼쳐 수정 EJB 그룹 노드의 위치를 지정하고 펼치십시오.
    b. **엔티티 Bean**을 펼치십시오.
    c. 수정 EJB 그룹을 펼치십시오.
    d. **메소드 확장자**를 누르고 오른쪽 분할창에서 다음을 수행하십시오.
        1) **고급** 탭을 선택하십시오.
        2) **보안 동일성**이 선택되었는지 확인하십시오.
        3) 개별 메소드에 대해서 **EJB 서버의 동일성 사용**이 선택되었는지 확인하십시오.
        4) **적용**을 누르십시오(수정한 경우).
    e. 왼쪽 탐색 분할창에서 수정된 EJB 그룹 아래의 **보안 역할**을 마우스 오른쪽 버튼으로 눌러 **새로 만들기**를 선택한 후 다음을 수행하십시오.
        1) **이름** 필드에서 WCSecurityRole을 입력한 후, **적용**을 누르십시오. 이 역할이 이미 존재하는 경우, 이 단계를 수행하지 않아도 됨에 유의하십시오.
    f. 왼쪽 탐색 분할창에서 수정된 EJB 그룹 아래의 **메소드 허용**을 마우스 오른쪽 버튼으로 눌러 **새로 만들기**를 선택한 후 다음을 수행하십시오.
        1) **메소드 허용 이름** 필드에 WCMethodPermission을 입력하십시오.
        2) **메소드** 선택 영역에서 **추가**를 누르십시오.
           메소드 추가 창이 열립니다.
        3) *modifiedEJBGroup*을 펼치고 모든 엔터프라이즈 bean을 선택하십시오(Shift 키를 누른 상태에서 선택). **확인**을 누르십시오. 모든 엔터프라이즈 bean은 언터프라이즈 bean 열에 표시되고, 모든 메소드는 유형 열에 표시됩니다.
        4) 역할 선택 영역에서 **추가**를 누르고 WCSecurityRole을 선택한 후 **확인**을 누르십시오.
        5) **적용**을 누르십시오.

19. **파일** 메뉴에서 **저장**을 선택하십시오.
20. 응용프로그램 어셈블리 도구를 종료하십시오.
21. 400 새로 수정된 WC_Enterprise_App_*instanceName*.ear 파일을 로컬 시스템에서 WebSphere Application Server를 실행 중인 iSeries 시스템으로 복사하십시오. 즉, 다음 디렉토리에서 파일을 복사하십시오.

    drive:\WebSphere\CommerceServer\working

    다음 디렉토리로 복사하십시오.

    iSeries_drive:\QIBM\UserData\WebCommerce\instances*instanceName*\working

    여기서 *iSeries_drive*는 iSeries IFS에 맵핑된 드라이브 이름입니다.

이 단계를 완료하면, 새로운 비즈니스 로직 뿐만 아니라 이전의 모든 로직을 포함하는 새 엔터프라이즈 응용프로그램이 작성됩니다. 새로 수정된 WC_Enterprise_App_*instanceName*.ear 파일에 이 모든 것이 포함됩니다.

엔터프라이즈 응용프로그램 중지 및 제거

이 절에서는 WebSphere Application Server 관리 콘솔을 사용하여 현재 실행 중인 엔터프라이즈 응용프로그램을 중지한 후 제거하는 방법에 대해 설명합니다.

엔터프라이즈 응용프로그램을 중지한 후 제거하려면 다음을 수행하십시오.

1. WebSphere Application Server 관리 콘솔을 여십시오.
2. **WebSphere 관리 도메인**을 펼치십시오.
3. **노드**를 펼치십시오.
4. ***nodeName***을 펼치십시오(여기서, ***nodeName***은 사용자 노드의 이름입니다).
5. **응용프로그램 서버**를 펼치십시오.
6. Windows AIX Solaris Linux WebSphere Commerce Application Server를 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server - *nodeName***을 마우스 오른쪽 버튼으로 누른 후 **중지**를 선택하십시오.

7. 400 WebSphere Commerce Application Server를 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 ***instanceName*** **- WebSphere Commerce Server**를 마우스 오른쪽 버튼으로 누르고 **중지**를 선택하십시오.
8. **엔터프라이즈 응용프로그램**을 펼치십시오.
9. Windows AIX Solaris Linux WebSphere Commerce Application Server를 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Enterprise Application - *instanceName* 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 중지를 선택하십시오.**
10. 400 WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 ***instanceName*** **- WebSphere Commerce Enterprise Application** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 **중지**를 선택하십시오.
11. WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Enterprise Application -*instanceName***(또는 iSeries의 경우 ***instanceName*** **- WebSphere Commerce Enterprise Application**) 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 **제거**를 선택하십시오.
12. 응용프로그램의 반출 여부 표시가 프롬프트될 때, **아니오**를 선택하십시오.
13. 응용프로그램의 제거 여부를 표시하도록 요청하는 프롬프트가 표시될 때, **예**를 선택하십시오.

엔터프라이즈 응용프로그램 반입

이 절에서는 XMLConfig 명령행 유틸리티를 사용하여 엔터프라이즈 응용프로그램을 반입하는 방법에 대해 설명합니다.

새 엔터프라이즈 응용프로그램을 반입하려면 다음을 수행하십시오.

1. Windows AIX Solaris Linux 다음 명령으로 XMLConfig 도구를 호출하여 엔터프라이즈 응용프로그램을 WebSphere Application Server에 반입하십시오.

   Windows

```
xmlConfig -import OutputFile.xml -adminNodeName wasHostName
```

AIX

```
/usr/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
   -adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
```

Solaris Linux

```
/opt/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
   -adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
```

여기서

- *wasHostName*은 현재 엔터프라이즈 응용프로그램을 포함하는 WebSphere Application Server의 노드 이름입니다.
- *OutputFile.xml*은 모든 엔터프라이즈 bean을 설명하는 XML 파일입니다.
- *wasAdminPort*는 WebSphere Application Server 관리자 포트입니다.

2. 400 다음 명령을 입력하여 엔터프라이즈 응용프로그램을 WebSphere Application Server에 반입하려면 XMLConfig 도구를 호출하십시오.

```
STRQSH
PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
xmlConfig -import
/QIBM/UserData/WebCommerce/instances/instanceName/working/OutputFile.xml
-adminNodeName wasHostName
 -nameServiceHost wasHostName -nameServicePort wasAdminPort
-instance wasInstanceName
```

여기서

- *OutputFile.xml*은 모든 엔터프라이즈 bean을 설명하는 XML 파일의 완전한 이름입니다.
- *wasHostName*은 현재 엔터프라이즈 응용프로그램을 포함하는 WebSphere Application Server의 노드 이름입니다.

  주: 400 *wasHostName*의 값은 대소문자가 구분되며 TCP/IP 구성에 있는 값과 일치해야 합니다(명령행 프로세서를 사용해서 CFGTCP, 옵션 12에 액세스하여 호스트 이름을 검증하십시오).

- *wasAdminPort*는 WebSphere Application Server 관리자 포트입니다.
- *wasInstanceName*은 WebSphere Application Server 인스턴스 이름입니다.

400 XMLConfig -import 명령을 실행하는 중, "/QIBM/UserData/WebAsAdv4/*wasInstanceName*/ installedApps/ WC_Enterprise_App_*instanceName*.ear에서 ear 파일을 펼칠 수 없습니다" 라는 오류 메시지를 받을 수 있습니다. 이 메시지를 받으면, 이전 디렉토리를 제거하거나 이름을 바꾼 후 다시 명령을 실행하십시오.

3. 400 엔터프라이즈 응용프로그램을 반입한 후, 스크립트를 실행하여 디렉토리 권한을 수정해야 합니다. 이 스크립트를 실행하려면 다음을 수행하십시오.
   - 명령 프롬프트에서 다음을 입력하십시오.

     ```
     STRSQH
     cd /QIBM/ProdData/WebCommerce/bin
     changeAuthority wasAdminInstanceName instanceName
     ```

     여기서
     - *wasAdminInstanceName*은 WebSphere Application Server 관리자 인스턴스의 이름입니다.
     - *instanceName*은 WebSphere Commerce 인스턴스의 이름입니다.

엔터프라이즈 응용프로그램 시작

이 장에서는 WebSphere Application Server 관리 콘솔을 사용하여 보기를 최신 정보로 고친 후 엔터프라이즈 응용프로그램을 시작하는 방법에 대해 설명합니다.

1. WebSphere Application Server 관리 콘솔을 여십시오.
2. **WebSphere 관리 도메인**, **노드** 및 *nodeName*을 차례로 펼치십시오.
3. *nodeName* 노드를 강조표시하십시오.
4. **선택된 서브트리를 최신 정보로 고침** 아이콘을 누르십시오.
5. 다음을 수행하여 WebSphere Commerce 응용프로그램을 시작하십시오.
   - **응용프로그램 서버**를 펼치십시오.

- Windows AIX Solaris Linux WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 **WebSphere Commerce Server** - *instanceName*을 마우스 오른쪽 버튼으로 누른 후 **시작**을 선택하십시오.
- 400 WebSphere Commerce 응용프로그램을 마우스 오른쪽 버튼으로 누르십시오. 예를 들어 *instanceName* - **WebSphere Commerce Server** 응용프로그램을 마우스 오른쪽 버튼으로 누른 후 **시작**을 선택하십시오.

부록 C. VisualAge for Java 추가정보

이 절에서는 개발 환경 내의 문제점 해결, 성능 향상 및 단순화와 관련된 몇 가지 추가 정보를 설명합니다.

WebSphere Test Environment에서 Servlet 엔진의 특성 변경

WebSphere Test Environment의 Servlet 엔진 특성은 default.servlet_engine 특성 파일에서 제어됩니다. 이 파일 내에서 웹 서버의 문서 루트를 수정할 수 있으며 WebSphere Test Environment에서 사용되는 포트를 변경할 수 있습니다.

WebSphere Test Environment에서 기능이 중단된 상태로 재부트 방법을 사용할 수 없는 경우에 포트를 변경하려고 할 수 있습니다. 포트를 변경하려면 다음을 수행하십시오.

1. 텍스트 편집기에서 *VAJ_install_path*\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine 파일을 여십시오.
2. <transport> 스탠자에서 다음 행의 8080 값을 사용 가능한 포트로 변경하십시오.

   ```
   <arg name="port" value="8080"/>
   ```

3. 다음 행의 8080 값을 위에 지정된 동일한 포트로 변경하십시오.

   ```
   <hostname-binding hostname="localhost:8080" servlethost="default_host"/>
      <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
   ```

4. 파일을 저장하십시오.
5. WebSphere Test Environment 제어 센터를 열고 Servlet 엔진을 시작하십시오.

WebSphere Test Environment의 기본 문서 루트는 *VAJ_install_path*\IDE\ProjectResources\IBM WebSphere Test Environment\hosts\default_host \default_app\web입니다. 이 디렉토리를 다른 디렉토리로 이동하려면 다음을 수행하십시오.

1. WebSphere Test Environment 제어 센터를 열고 Servlet을 중지하십시오.
2. 텍스트 편집기에서 *VAJ_install_path*\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine 파일을 여십시오.
3. <websphere-webgroup name="default_app"> 스탠자에서 <document-root>$approot$/web</document-root>를 다음으로 변경하십시오.

   ```
   <document-root>your_document_root</document-root>
   ```

   여기서 *your_document_root*는 원하는 문서 루트입니다.
4. 다음 행의 8080 값을 위에 지정된 동일한 포트로 변경하십시오.

   ```
   <hostname-binding hostname="localhost:8080" servlethost="default_host"/>
      <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
   ```
5. 파일을 저장하십시오.
6. WebSphere Test Environment 제어 센터를 열고 Servlet 엔진을 시작하십시오.

PNS 문제점 해결

PNS에서 문제점을 발견한 경우, PNS 데이터베이스를 제거한 후 재작성해야 할 수도 있습니다. 이 데이터베이스 작성에 대한 자세한 내용은 *Commerce Studio 설치 안내서*를 참조하십시오

또한 메시지가 포트가 현재 사용 중임을 알리는 경우, WebSphere Application Server를 실행해서는 안됩니다.

컴파일된 JSP 파일 삭제

VisualAge for Java에서는 성능상의 이유로 컴파일된 JSP 파일이 저장된 프로젝트 폴더를 유지보수합니다. 컴파일된 JSP 파일을 삭제해야 할 경우가 있을 수 있습니다. 예를 들어, JSP 파일에서 데이터 bean을 제거하면 다음 번에 JSP 파일을 호출할 때 오류가 발생할 수 있습니다. 이 경우, 컴파일된 JSP 파일을 삭제할 수 있습니다.

컴파일된 JSP 파일을 삭제하려면 다음을 수행하십시오.

1. VisualAge for Java에서 프로젝트 탭을 선택하십시오.
2. **JSP 페이지 컴파일 생성 코드** 프로젝트로 화면이동하십시오.
3. 전체 프로젝트를 선택하거나(다수의 컴파일된 JSP 파일을 삭제해야 할 경우) 프로젝트를 펼쳐 다시 컴파일해야 하는 파일만 삭제하십시오.

주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 사용권까지 부여하는 것은 아닙니다. 사용권에 대한 의문사항은 다음으로 문의하십시오.

137-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트(DBCS) 정보와 관련된 사용권 문의는 한국 IBM 고객만족센터에 문의하거나 다음의 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음의 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다.

IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없

이 이 책을 현상태대로 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 암시적인 보증의 면책사항을 허용하지 않으므로 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통고없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 갖게 됩니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(1) 독립적으로 작성된 프로그램 및 기타 프로그램(이 프로그램 포함) 간의 정보 교환 (2) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 사용권자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건에 따라(예를 들면, 사용권 지불 포함) 사용할 수 있습니다.

이 정보에 기술된 사용권 프로그램 및 사용 가능한 모든 사용권 자료는 IBM이 IBM 기본 계약, IBM 프로그램 사용권 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

이 책에 포함된 성능 데이터는 제한된 환경에서 산출된 것입니다. 그러므로 다른 운영 환경에서의 결과와 상당히 다를 수도 있습니다. 일부 측정은 개발 단계의 시스템에서 이루어진 것이므로 그 측정치가 일반적으로 사용 가능한 시스템에서도 동

일하다고 보장할 수 없습니다. 또한 일부 측정치는 보외법을 통해 이루어졌으므로 실제 결과는 다를 수도 있습니다. 이 책의 사용자는 자신의 고유 환경에 적합한 데이터를 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 배상 청구에 대해서는 확신할 수 없습니다. 타사 제품의 기능에 대한 질문은 해당 제품의 제공업체로 해야 합니다.

IBM의 장래 방향이나 의도에 대한 모든 진술은 통지 없이 변경되거나 취소될 수 있으며 목적과 목표만을 나타낼 뿐입니다.

표시된 모든 IBM 가격은 IBM이 제안하는 소매가이며, 최신 가격으로 통지 없이 변경될 수 있습니다. 소매업체 가격들은 다양할 수 있습니다.

이 정보는 계획 목적으로만 사용됩니다. 여기의 정보는 설명된 제품이 사용 가능해지기 전에 변경될 수 있습니다.

이 책에서는 일상적인 비즈니스 운영에 사용되는 데이터 및 보고서 예가 수록되어 있습니다. 가능한 한 완벽하게 표시하기 위해 예제에는 개인, 회사, 브랜드, 상품의 이름이 포함됩니다. 그러한 이름들은 모두 가공의 이름으로서, 실제 회사에서 사용하는 이름 및 주소와 유사하다 하더라고 전적으로 우연입니다.

저작권

이 책에서는 소스 언어로 된 견본 응용프로그램이 들어 있어, 여러 운영체제에서의 프로그래밍 기술에 대해 설명합니다. 견본 프로그램이 작성된 운영체제에 해당하는 응용프로그램 프로그래밍 인터페이스를 따르는 응용프로그램을 개발, 사용, 마케팅, 배포하려는 목적으로 해당 견본 프로그램을 무료로 복사, 수정, 배포할 수 있습니다. 이 책의 예들은 모든 조건에서 완전하게 테스트된 것은 아닙니다. 따라서 IBM은 해당 프로그램의 신뢰성, 서비스 가능성 또는 기능에 대해 보증하거나 암시할 수 없습니다. IBM의 응용프로그램 프로그래밍 인터페이스를 따르는 응용프로그램을 개발, 사용, 마케팅, 배포하려는 목적으로 해당 견본 프로그램을 무료로 복사, 수정, 배포할 수 있습니다.

이러한 견본 프로그램의 각 사본이나 일부 또는 여기에서 파생된 저작물에는 다음과 같이 저작권 주의사항이 포함되어야 합니다.

이 정보를 소프트카피로 보는 경우, 사진과 컬러 그림은 나타나지 않을 수 있습니다.

상표 및 서비스표

다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표 또는 등록상표입니다.

400
AIX
AS/400
CICS
DB2
IBM
iSeries
MQSeries
Net.Commerce
Net.Data
VisualAge
WebSphere

Windows 및 Windows NT는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표 또는 등록상표입니다.

Oracle은 미국 또는 기타 국가에서 사용되는 Oracle Corporation의 등록상표입니다.

Solaris, Java 및 모든 Java 기반 상표와 로고는 미국 또는 기타 국가에서 사용되는 Sun Microsystems의 상표 또는 등록상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

색인

[카]

[타]

[파]

C

E

F

J

M

S

U

V

IBM 한글 지원에 관한 설문

FAX : (02) 3787-0123

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다.
독자 여러분의 좋은 의견을 기다립니다.

책 제목: IBM WebSphere Commerce
프로그래머 안내서
버전 5.4

책 번호: GA30-1614-00

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	o 중대형 서버 o UNIX 서버 o PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까? 그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

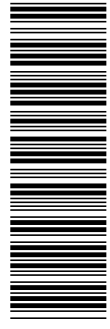
6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다. 어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

☞ IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.

부품 번호: CT024KO

(1P) P/N: CT024KO



GA30-1614-00

Spine information:

IBM WebSphere Commerce

프로그래머 안내서

버전 5.4