Modellazione delle architetture dell'applicazione Web con UML

Jim Conallen

White Paper del Software Rational

TP 157, 6/99

Una versione di questa documentazione appare nell'edizione di ottobre 1999 (volume 42, numero 10) delle comunicazioni dell'ACM.

Rational®
the software development company

Indice

Abstract 1

Panoramica 1

Modellazione 2

Architettura dell'applicazione Web 3

Modellazione di pagine Web... ...4

Forme... .7

Cornici 8

Conclusioni... ..9

Abstract

Le applicazioni Web stanno diventando sempre più complesse e cruciali per la mission. Per consentire la gestione di tale complessità, è necessario che vengano modellate. UML (Unified Modeling Language) è il linguaggio standard per la modellazione dei sistemi con molti software. Quando si cerca di modellare le applicazioni Web con UML, diventa evidente che alcuni componenti non corrispondono agli elementi della modellazione standard UML. Per attenersi ad una sola notazione di modellazione relativa all'intero sistema (componenti Web e tradizionali componenti di livello intermedio), è necessario estendere UML. Questo documento mostra un'estensione di UML, utilizzando il meccanismo di estensione formale. L'estensione viene progettata in modo che i componenti specifici del Web possano essere integrati con il resto del modello del sistema e mostrare un livello di astrazione e di dettaglio adeguato a progettisti, implementatori e architetti di applicazioni Web.

Panoramica

Negli ultimi anni, un nuovo termine è entrato a far parte del vocabolario IT: applicazione Web. Sembra che tutti quelli coinvolti con sistemi software di business per la creazione di applicazioni Web, siano anche interessati agli sforzi del software non correlato al business. Per i primi che hanno adottato questa architettura, il termine applicazione Web, come gli stessi sistemi, si è evoluto da piccoli siti Web di successo in grandi applicazioni ad "n" livelli. Per un'applicazione Web, è normale soddisfare migliaia di utenti simultanei, sparsi in tutto il mondo. La strutturazione delle applicazioni Web è un'attività seria.

Quando viene messo alla prova, il termine applicazione Web ha significati diversi per persone differenti. Alcuni ritengono che utilizzi Java; altri credono che utilizzi un server Web, tutti sono concordi che si tratta di qualcosa che sta nel mezzo. Per gli scopi del presente documento, un'applicazione Web è un sistema Web (server, rete, HTTP e browser Web) in cui l'input dell'utente (input di navigazione e di dati) influisce sullo stato del business. Tale definizione vuole dimostrare che un'applicazione Web è un sistema software con stato del business e che il relativo "front end" viene trasmesso principalmente tramite un sistema Web.

L'architettura di un'applicazione Web è quella di un sistema di server client, con poche differenze. Uno dei vantaggi più significativi di un'applicazione Web è la distribuzione. Distribuire un'applicazione Web di solito è problema relativo all'impostazione dei componenti lato server in una rete. Non sono richiesti software o configurazioni speciali da parte del client. Un'altra differenza significativa è la natura delle comunicazioni tra client e server. Il protocollo di comunicazione principale di un'applicazione Web è HTTP, senza connessione, progettato per la solidità e la tolleranza di errori anziché per la massima velocità di elaborazione di comunicazione. In un'applicazione Web, la comunicazione tra client e server ruota intorno alla navigazione di pagine Web e non a comunicazioni dirette tra oggetti lato server e lato client. A livello uno di astrazione, tutta la messaggistica in un'applicazione Web può essere descritta come la richiesta e la ricezione di entità di pagine Web. Parlando in generale, l'architettura di un'applicazione Web non si differenzia molto da quella di un sito Web dinamico.

Le differenze tra i due, fatta eccezione per il dinamismo, riguardano il relativo utilizzo. Le applicazioni Web implementano la logica di business e l'uso modifica lo stato del business (come se venisse catturato dal sistema). Tutto questo è importante perché definisce il punto focale dello sforzo di modellazione. Le applicazioni Web eseguono la logica di business e in questo modo i modelli fondamentali del sistema si focalizzano sulla logica e lo stato del business, non sui dettagli di presentazione. La presentazione è importante (altrimenti il sistema non potrebbe fare nulla di buono), ad ogni modo si deve tentare una separazione netta tra le problematiche di business e di presentazione. Se le problematiche di presentazioni risultano rilevanti, o complicate, allora devono essere modellate, ma non necessariamente come parte integrante del modello della logica di business. Inoltre, le risorse che lavorano sulla presentazione tendono ad essere più artistiche e meno attente all'implementazione delle regole di business.

Una metodologia/notazione associata con lo sviluppo dei sistemi Web è RMM (Relationship Management Methodology), che riguarda la progettazione, costruzione e manutenzione dei sistemi intranet e Internet. Lo scopo principale è ridurre i costi di manutenzione di siti Web dinamici, basati su database. RMM sostiene una rappresentazione visiva del sistema per facilitare dibattiti di progettazione, un processo iterativo che include la scomposizione degli elementi visivi in pagine Web e la relativa associazione con entità di database ed è un approccio “dalla A alla Z” per la creazione e manutenzione di siti Web dinamici.

RMM non raggiunge lo scopo quando si creano applicazioni Web, che, essendo incentrate sulla logica di business, includono una serie di meccanismi tecnologici per implementare tale logica, non adeguatamente coperti dalla notazione RMM. Tali tecnologie come scrittura lato client, applet e controlli ActiveX, spesso contribuiscono in modo significativo all'esecuzione delle regole di business del sistema. Inoltre, le applicazioni Web possono essere utilizzate come un meccanismo di distribuzione per un sistema di oggetto distribuito. Gli applet e i controlli ActiveX possono contenere componenti che interagiscono in modo asincrono con componenti lato server via RMI o DCOM, indipendenti dal server Web. Applicazioni sofisticate si avvalgono di istanze multiple di browser e di cornici sul client, che creano e mantengono i meccanismi di comunicazione.

Poiché tutti questi meccanismi contribuiscono alla logica di business del sistema, devono essere modellati in questo modo e, siccome rappresentano soltanto una parte di tale logica, devono essere integrati con il resto dei modelli del sistema. In molte situazioni, lo sforzo della logica di business viene eseguito dietro il server Web in uno dei livelli lato server. La scelta del linguaggio e della notazione di modellazione viene decisa di solito in base alle esigenze di questo lato dell'applicazione. Poiché OMG accetta UML come un linguaggio ufficiale di modellazione dell'oggetto, molti altri sistemi devono essere espressi con notazione UML. Secondo un parere diffuso, UML è il linguaggio scelto per sistemi con molti software di modellazione. Nella modellazione di applicazioni Web, il problema maggiore potrebbe diventare il modo in cui esprimere la logica di business nei componenti specifici del Web insieme al resto dell'applicazione, ad esempio.
La soluzione è nell'abilità di delineare l'esecuzione della logica di business del sistema in questi elementi specifici del Web e nelle tecnologie con UML.

Questo documento rappresenta un'introduzione alle problematiche e alle possibili soluzioni per la modellazione di applicazioni Web. Si concentra sui componenti strutturalmente significativi, in particolare sulle applicazioni Web e sul modo in cui modellarle con UML. Risulta scontato che il lettore possiede familiarità con UML, gli elementi principali object-oriented e lo sviluppo delle applicazioni Web. Il lavoro descritto in questo documento si basa su alcuni presupposti:

- Le applicazioni Web sono sistemi con molti software che stanno diventando più complessi e si stanno inserendo in più ruoli cruciali per la mission.
- Un modo per gestire le complessità all'interno dei sistemi software è ricavarle e modellarle.
- Di solito, un sistema software possiede modelli multipli e ognuno rappresenta un punto di vista, livello di astrazione e dettaglio diversi.
- Il livello di astrazione e dettaglio adatto dipende dagli artefatti e dalle attività nel processo di sviluppo.
- Il linguaggio di modellazione standard per sistemi con molti software è UML (Unified Modeling Language).

I concetti e le idee espressi nel presente documento sono trattati in modo più completo nel libro: “Building Web Applications with UML”, in attesa di pubblicazione in Object Technology Series di Addison Wesley Longman.

Modellazione

I modelli consentono di capire il sistema semplificando alcuni dettagli. La scelta di cosa modellare ha un effetto enorme sulla comprensione del problema e la relativa soluzione. Le applicazioni Web, come altri sistemi con molti software, di solito vengono rappresentate con un insieme di modelli: modelli di caso d'uso, di implementazione, di distribuzione e così via. Un modello aggiuntivo, utilizzato dai sistemi Web, è la mappa del sito, una sintesi delle pagine Web e dei tragitti di navigazione in tutto il sistema.

La maggior parte delle tecniche di modellazione praticate oggi si adattano bene allo sviluppo di vari modelli di un'applicazione Web e non sono necessari ulteriori dibattiti. In ogni modo, un modello molto importante, ADM (Analysis/Design Model), presenta alcune difficoltà quando si cerca di includere le pagine Web e il codice eseguibile associato, insieme ad altri elementi.

Una volta deciso come modellare qualcosa e determinato il livello giusto di astrazione e dettaglio, risulta decisivo fornire qualcosa di utile agli utenti del modello. Generalmente, la cosa migliore è modellare gli artefatti del sistema, quelle entità di vita reale che saranno costruite e manipolate per creare il prodotto finale. La modellazione degli elementi interni del server Web o dei dettagli del browser Web non sarà di aiuto a progettisti e architetti di un'applicazione Web. Una volta sul client, la modellazione delle pagine, dei link reciproci, di tutti i contenuti dinamici che partecipano alla creazione delle pagine stesse e il contenuto dinamico delle pagine risultano di primaria importanza. Sono questi gli artefatti che vengono creati dai progettisti e implementati dagli implementatori. È necessario modellare pagine, collegamenti ipertestuali e contenuto dinamico sul client e sul server.

Il passo successivo è l'associazione di tali artefatti per la modellazione degli elementi. Ad esempio, i collegamenti ipertestuali vengono associati per unire gli elementi all'interno del modello. Un collegamento ipertestuale rappresenta un path di navigazione da una pagina all'altra. Volendo ampliare questa idea, le pagine possono associarsi a classi nella vista logica del modello. Se una pagina Web era una classe all'interno del modello, allora gli script della pagina dovranno associarsi alle operazioni della classe stessa. Le variabili page-scoped negli script devono associarsi agli attributi di classe. Un problema emerge quando una pagina Web contiene un insieme di script che si eseguono sul server (che prepara il contenuto dinamico della pagina) e un altro insieme di script completamente differente che si eseguono soltanto sul client (cioè JavaScript). In questo scenario, guardando una classe di pagina Web all'interno del modello, è facile notare confusione tra quali operazioni, relazioni e attributi sono attivi sul server (mentre la pagina è in fase di preparazione) e quali sul client, nel momento in cui l'utente interagisce con la pagina. Inoltre, una pagina Web, trasmessa in una relativa applicazione, è modellata perfettamente come componente del sistema. La semplice associazione di una pagina Web ad una classe UML non aiuta a comprendere meglio il sistema.

I creatori di UML hanno capito che ci saranno sempre situazioni in cui UML, fuori dalla casella, non sarà sufficiente a catturare la semantica di un dominio o architettura particolari. Per raggiungere tale obiettivo, un meccanismo di estensione formale è stato definito per consentire ai professionisti di ampliare la semantica di UML. Tale meccanismo permette di definire stereotipi, valori con tag e vincoli applicabili a elementi del modello.

Uno stereotipo è un ornamento che consente di determinare un significato semantico nuovo per un elemento della modellazione. I valori con tag sono coppie di valore fondamentali, che possono essere associati ad un elemento della modellazione, il quale consente di "applicare un tag" a tutti i valori su un elemento di modellazione. I vincoli sono regole che definiscono la creazione di un modello e possono essere espressi come testi a formato libero o con il più formale OCL (Object Constraint Language).

Il lavoro descritto in questo documento introduce un'estensione a UML per applicazioni Web, la quale va completamente oltre l'ambito del documento; ad ogni modo, la maggior parte dei concetti e delle spiegazioni vengono illustrati qui.
Nella fase finale della modellazione, è necessaria una distinzione netta tra la logica di business e di presentazione. Per quanto riguarda le tipiche applicazioni di business, soltanto la logica di business appartiene a ADM, mentre i dettagli di presentazione, come i pulsanti animati, l'aiuto fly-over e altri miglioramenti UI di solito non gli appartengono. Se un modello separato UI viene costruito per l'applicazione, è nel posto giusto. È necessario che ADM rimanga focalizzato sulla risoluzione dei problemi di business e di spazio. In periodi come questo, pieno di artisti Web, l'aspetto e l'atmosfera di una pagina Web vengono progettati e implementati perfettamente da uno specialista (artista tecnico e grafico) e non dagli sviluppatori tradizionali.

Architettura dell'applicazione Web

L'architettura di base di un'applicazione Web include alcuni browser, una rete e un server Web. I browser richiedono "pagine Web" dal server. Ogni pagina è un'unione di contenuto e istruzioni di formattazione, espressa con HTML. Alcune pagine comprendono degli script lato client, interpretati dal browser, che definiscono ulteriori comportamenti dinamici, relativi alla visualizzazione, e che spesso interagiscono con il browser, il contenuto della pagina e i controlli addizionali (applet, controlli ActiveX e plugin) compresi nella pagina. L'utente guarda e interagisce con il contenuto della pagina e qualche volta inserisce informazioni negli elementi del campo all'interno della pagina e le inoltra al server per l'elaborazione. L'utente può anche interagire con il sistema navigando in pagine diverse all'interno del sistema stesso tramite collegamenti ipertestuali. In entrambi i casi, l'utente sta fornendo un input al sistema che ne può alterare lo "stato di business".

Secondo la prospettiva del client, la pagina Web è sempre un documento HTML formattato. Ad ogni modo, sul server una "pagina Web" può mostrarsi in modi differenti. Nelle applicazioni Web precedenti, le relative pagine dinamiche era costruite con CGI (Common Gateway Interface), che definisce un'interfaccia per script e moduli compilati da utilizzare per accedere alle informazioni inoltrate insieme alla richiesta di pagina. In un sistema basato su CGI, una directory speciale viene configurata sul server Web per poter eseguire script in risposta alla richiesta di pagina. Quando viene richiesto uno script CGI, il server, anziché restituire i contenuti del file (come farebbe per i file HTML formattati), elabora o esegue il file con l'interprete appropriato (solitamente shell di PERL) e riconsegna l'output al client che lo aveva richiesto. Il risultato finale di questo processo è un flusso HTML formattato, restituito al client che ne aveva fatto richiesta. La logica di business viene eseguita nel sistema mentre viene elaborato il file; durante questo tempo, possiede gli elementi potenziali per interagire con le risorse lato server, quali database e componenti di livello intermedio.

I server Web odierni hanno apportato miglioramenti alla progettazione di base. Ora sono molto più sicuri e includono funzioni quali gestione dello stato del client sul server, transazione dell'integrazione di elaborazione, amministrazione remota e assemblaggio di risorse, per elencarne alcune. L'ultima generazione di server Web sta risolvendo queste problematiche, importanti per architetti di applicazioni cruciali per la mission, scalabili e robuste.

Guardando al ruolo degli script CGI, i server Web odierni possono essere divisi in tre grandi categorie: pagine di script, pagine compilate e un insieme dei due. Nella prima, ogni pagina Web che può essere richiesta da un browser del client viene rappresentata sul sistema di file del server Web come un file creato con script, che di solito è un'unione di HTML e altri linguaggi di script. Quando una pagina viene richiesta, il server Web ne delega l'elaborazione ad un motore che la riconosce; il risultato è la restituzione di un flusso HTML formattato al client che lo aveva richiesto. Esempi possono essere Active Server Page, Java Server Page e Cold Fusion di Microsoft.

Nella seconda categoria, pagine compilate, il server Web carica ed esegue un componente binario, il quale, come con le pagine create con script, ha accesso a tutte le informazioni che arrivano insieme alla richiesta di pagina (valori di campi forma e parametri). Il codice compilato utilizza i dettagli richiesti e di solito accede alle risorse lato server per produrre il flusso HTML ritornato al client. Sebbene non sia una regola, le pagine compilate tendono a includere una funzionalità più ampia delle pagine create con script. Una funzionalità diversa si può ottenere passando parametri alla richiesta di pagina compilata. Qualsiasi componente compilato può includere in effetti tutte le funzionalità delle pagine create con script di una intera directory. Le tecnologie che rappresentano questo tipo di architettura sono ISAPI di Microsoft e NSAPI di Netscape.

La terza e ultima categoria rappresenta pagine create con script che, una volta richieste, vengono compilate; questa versione compilata viene poi utilizzata da tutte le richieste successive da lì in avanti. Soltanto quando i contenuti della pagina originale si modificano, avviene una nuova compilazione. Questa categoria è una sorta di compromesso tra la flessibilità delle pagine create con script e l'efficienza delle pagine compilate.

Modellazione di pagine Web

In UML, le pagine Web, sia create con script che compilate, si associano direttamente ai componenti. Un componente è una parte "fisica" e sostituibile del sistema. La vista Implementazione (vista Componente) del modello descrive i componenti del sistema e le relazioni che intercorrono. In un'applicazione Web, questa vista definisce tutte le pagine Web del sistema e le reciproche relazioni, cioè i collegamenti ipertestuali. A livello uno, un diagramma di componente di un sistema Web è simile ad una mappa del sito.
Poiché i componenti rappresentano solo l'involucro fisico delle interfacce, non sono adatti alla modellazione delle collaborazioni all'interno delle pagine. È necessario che questo livello di astrazione, estremamente importante per i progettisti e implementatori, diventi parte del modello. Ogni pagina Web è una classe UML all'interno della vista Progettazione (vista Logica) del modello e le relazioni con altre pagine (associazioni) rappresentano collegamenti ipertestuali. Ad ogni modo, questa astrazione crolla quando ciascuna pagina Web data può rappresentare potenzialmente un insieme di funzioni e collaborazioni che esiste soltanto sul server e un insieme completamente differente che può esistere solo sul client. Un esempio può essere la pagina Web del server creato con script che adopera pagine dinamiche HTML (scripting lato client) come parte del relativo output. La reazione automatica a questo problema può essere stereotipare ogni attributo o operazione nella classe per indicare se era efficace sul lato server o lato client. A questo punto, il modello, progettato originariamente per rendere le cose più semplici, sta diventando complesso.

Un approccio migliore al problema è considerare l'elemento principale della "separazione delle considerazioni". Logicamente, una pagina Web si comporta in modo differente sul server e sul client. Mentre è in esecuzione sul server, ha accesso (cioè si relaziona con) alle risorse lato server (componenti di livello intermedio, database, sistema di file, ecc.). Sul client, questa stessa pagina, o l'output HTML emesso di questa pagina, ha un comportamento e un insieme di relazioni diversi. Una pagina creata con script infatti si relaziona con il browser tramite DOM (Document Object Model) e con gli applet Java, i controlli ActiveX o i plugin specificati dalla pagina. Per un progettista serio, possono esistere relazioni aggiuntive con ulteriori pagine "attive" sul client, che appaiono in altre cornici HTML o istanze browser.

Separando le considerazioni, è possibile modellare l'aspetto lato server di una pagina Web con una classe e l'aspetto lato client con un'altra. È possibile distinguere i due aspetti utilizzando il meccanismo di estensione di UML per definire gli stereotipi e le icone per ogni pagina ½server╗ e ½client╗. In UML, gli stereotipi consentono di determinare una semantica nuova per un elemento di modellazione. Le classi di stereotipi possono essere restituite in un diagramma UML o con un'icona cliente o semplicemente decorate con il nome dello stereotipo tra virgolette (½╗). Le icone sono utili per i diagrammi di panoramica, dove è meglio utilizzare semplici tag quando vengono mostrati gli attributi e le operazioni di classe.

Per le pagine Web, gli stereotipi indicano che la classe è un'astrazione del comportamento logico di una pagina Web o sul client o sul server. Le due astrazioni sono collegate l'una all'altra tramite una relazione direzionale. Questa associazione è stereotipata come ½build╗, poiché una pagina server crea una pagina client (Figura 1). Tutte le pagine Web dinamiche (cioè pagine il cui contenuto è determinato nel momento del runtime) sono costruite con una pagina server. Tutte le pagine client sono prodotte al massimo da una sola pagina server; ad ogni modo, quest'ultima è in grado di creare pagine client multiple.
Una normale relazione tra pagine Web è costituita dal collegamento ipertestuale, che, in un'applicazione Web, rappresenta un path di navigazione in tutto il sistema. Questa relazione si esprime nel modello con un'associazione stereotipata di ½ link╗, la quale viene originata sempre da una pagina client e indica una pagina client o server.

I collegamenti ipertestuali vengono implementati nel sistema come una richiesta per una pagina Web, mentre le pagine Web vengono modellate nella vista Implementazione. Un associazione di link ad una pagina client è quasi interamente equivalente ad un'associazione di link ad una pagina server, che ne crea una client. Tutto ciò perché un link in effetti non è una richiesta né per una pagina Web né per delle astrazioni di classe.
Un componente della pagina Web che esegue entrambe le astrazioni della pagina corrisponde ad un link a qualsiasi classe, realizzato dal componente della pagina.

I valori con tag vengono utilizzati per definire i parametri passati insieme ad una richiesta di link. Il valore con tag dell'associazione di ½link╗ "Parametri" è un elenco di nomi di parametri (e di valori opzionali) previsti e utilizzati dalla pagina server che elabora la richiesta. Nella Figura 2, la pagina SearchResults contiene un numero variabile di collegamenti ipertestuali (0..*) alla pagina server GetProduct, in cui ogni link ha un valore differente per il parametro Id del prodotto. La pagina GetProduct crea la pagina ProductDetail del prodotto specificato dal parametro Id del prodotto.

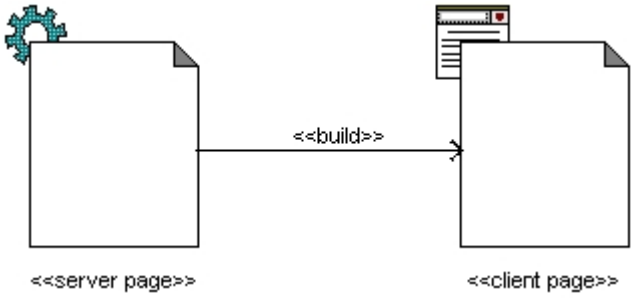

Figura 1. Pagine server creano pagine client

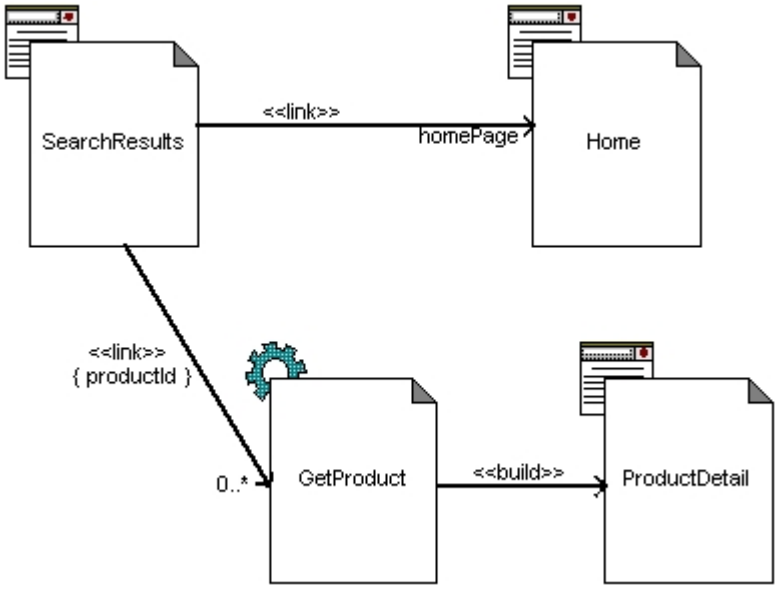

Figura 2. Utilizzo dei parametri dei collegamenti ipertestuali

L'utilizzo di questi stereotipi rende più semplice modellare gli script e le relazioni di una pagina. Le operazioni della classe di pagina ½server╗ sono diventate funzioni negli script lato server della pagina, mentre i relativi attributi sono diventati variabili page-scoped (accessibili dalle funzioni della pagina). Allo stesso modo, le operazioni e gli attributi di classe della pagina ½client╗ sono diventati funzioni e variabili visibili sul client. Il vantaggio maggiore della divisione in classi differenti degli aspetti lato client e server di una pagina è nelle relazioni tra pagine e altre classi del sistema. Le pagine client sono modellate con relazioni alle risorse lato server: DOM, applet Java, controlli ActiveX e plugin (Figura 3). Le pagine server sono modellate con relazioni alle risorse lato server, componenti di livello intermedio, componenti di accesso a database, sistemi operativi di server e così via, illustrati nella Figura 4.

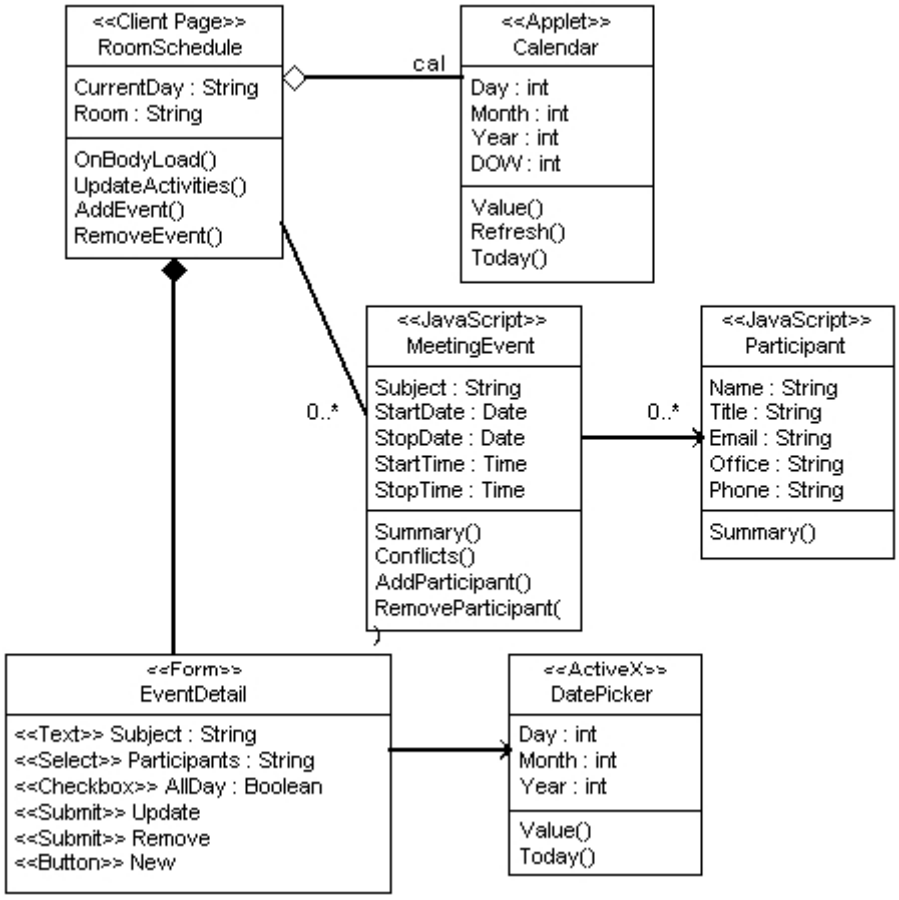

Figura 3. Collaborazioni client

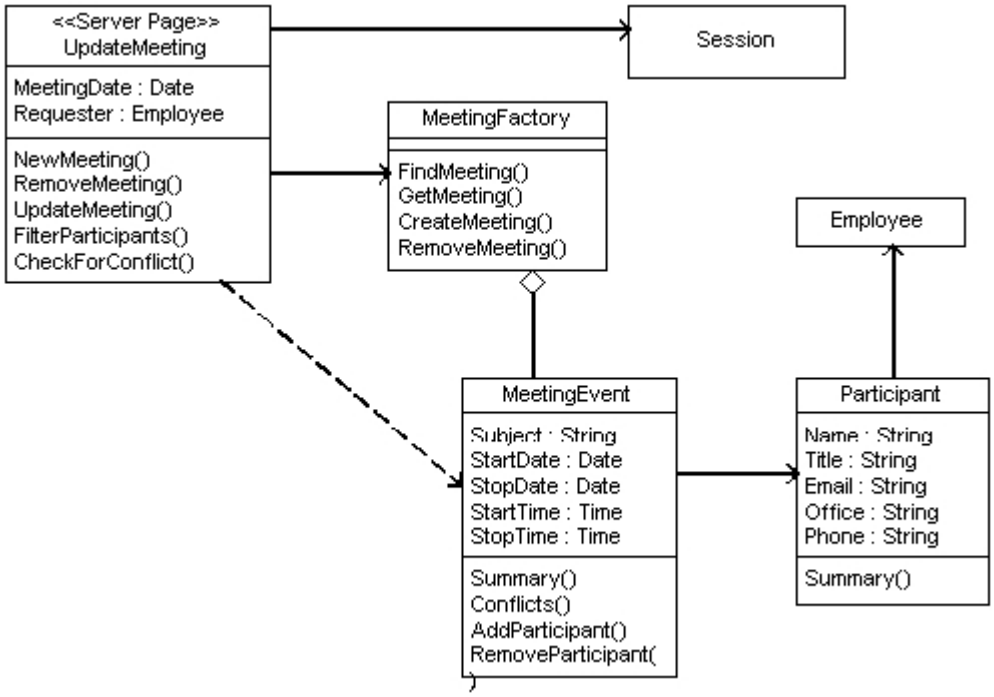

Figura 4. Collaborazioni server

Uno dei maggiori vantaggi dell'utilizzo di stereotipi di classe per modellare il comportamento logico delle pagine Web sta nel fatto che le collaborazioni con i componenti lato server possono essere espresse allo stesso modo di quelle lato client. La pagina ½server╗ è semplicemente un'altra classe che partecipa alla logica di business del sistema. Ad un livello più concettuale, le pagine server assumono il ruolo di controllori, che organizzano l'attività dell'oggetto di business necessaria per raggiungere l'obiettivo iniziato dalla richiesta della pagina browser.

Dal lato client, le collaborazioni si complicano un po'. Ciò è dovuto in parte alla varietà di tecnologie che possono essere impiegate. Una pagina client è un documento HTML che racchiude informazioni sia sul contenuto che sulla presentazione. I browser restituiscono pagine HTML utilizzando le istruzioni di formattazione all'interno della pagina, qualche volta con fogli di stile separati. Nel modello logico, questa relazione può essere espressa con una dipendenza dalla pagina client ad una classe stereotipata ½Foglio di stile╗. I fogli di stile sono principalmente un elemento di presentazione e sono spesso lasciati fuori da ADM.

Forme

Il meccanismo più importante per l'immissione di dati per le pagine Web è la Forma. Le forme vengono definite in un documento HTML con tag <form> e ogni forma specifica la pagina alla quale sta per essere inoltrata. Una forma contiene un numero di elementi di input, tutti espressi come tag HTML. I più comuni sono: <input>, <select> e <textarea>. Il tag input è un po' sovraccaricato poiché può essere un campo di testo, un checkbox, un pulsante radio, un pulsante di spinta, un'immagine, un campo nascosto e altri tipi meno comuni.

La modellazione delle forme significa un altro stereotipo di classe: ½Forma╗. Una ½Forma╗ non ha operazioni, poiché quelle che possono essere definite in un tag <form> appartengono alla pagina client. Gli elementi di un input di forma sono tutti attributi stereotipati della classe ½Forma╗. Una ½Forma╗ può relazionarsi con gli applet o i controlli ActiveX che agiscono come controlli di input. Ogni forma si relaziona anche con una pagina server (la pagina che elabora l'inoltro della forma). Questa relazione è un ½inoltro╗ stereotipato. Poiché le forme sono contenute completamente in un documento HTML, vengono espresse in un diagramma UML con una solida forma di aggregazione. La Figura 5 mostra una semplice pagina di carrello che definisce una forma e illustra la relazione di inoltro alla pagina server di elaborazione.

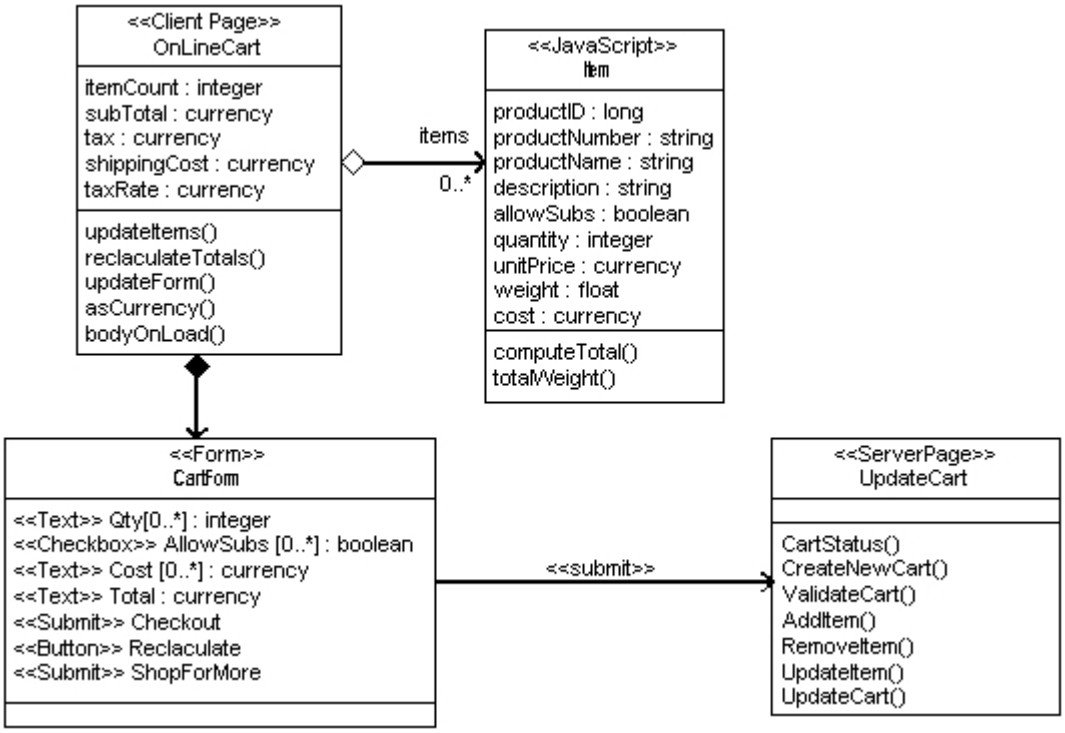

Figura 5. Inoltro di forme a pagine server

Nella Figura 5, la classe stereotipata ½JavaScript╗ è un oggetto che rappresenta gli elementi nel carrello. La sintassi di matrice viene utilizzata nella descrizione delle proprietà della forma per quei campi che hanno un numero di istanze variabile. In questo caso specifico, il carrello può avere valore zero in corrispondenza di molte voci, ognuna con un elemento di <input> di Qty, AllowSubs, Cost e Total.

Poiché tutte le attività nella pagina client vengono eseguite con JavaScript, che è un linguaggio meno tipico, i tipi di dati specificati per ciascuno di questi attributi sono utilizzati soltanto affinché l'implementatore abbia un chiarimento. Il tipo viene ignorato quando tag di input vengono implementati in JavaScript o come HTML. Questo riguarda anche i parametri della funzione, che, sebbene non siano visualizzati correttamente in questa figura, sono parte del modello.

Cornici

L'utilizzo di cornici HTML in un sito o applicazione Web è stato oggetto di dibattiti fin dalla sua introduzione. Le cornici rendono pagine multiple attive e visibili all'utente in un determinato momento. Anche l'ultimo insieme di funzioni, relativo ai browser odierni più comuni, rende attive istanze di browser multipli sulla macchina dell'utente. Utilizzando script e componenti di pagine HTML dinamiche, queste ultime possono interagire tra loro. Risultano significativi gli elementi potenziali per interazioni complesse sul client, il client stesso e la necessità di modellarlo.

L'architetto software decide se impiegare cornici o istanze di browser multipli in una applicazione e se lo fa, per la stessa ragione, è necessario che il modello di questo comportamento lato client venga rappresentato in ADM. Riguardo l'utilizzo della cornice di modello, vengono definiti altri due stereotipi di classe, ½frameset╗ e ½target╗, e uno stereotipo di associazione ½link con tag╗. Una classe di cornici rappresenta un oggetto di contenitore, si associa direttamente al tag <frameset> HTML e contiene pagine client e target. Una classe di target è una cornice denominata o un'istanza di browser cui fanno riferimento altre pagine client. Un'associazione di link di destinazione non è un collegamento ipertestuale ad un'altra pagina, ma un'associazione che viene restituita in un target specifico. Nell'esempio della Figura 6, una vista di struttura comune viene presentata in un browser che utilizza due cornici. Una cornice viene denominata con un target (Contenuto), laddove l'altra cornice contiene semplicemente una pagina client. La cornice di questa pagina rappresenta il TOC (Table of Contents) del libro, mentre i relativi collegamenti ipertestuali possiedono target, così da poter essere restituiti nella cornice Contenuto. L'effetto prodotto è che l'indice si trova nella pagina di sinistra, mentre gli argomenti del libro, capitolo per capitolo, si trovano nella pagina di destra.

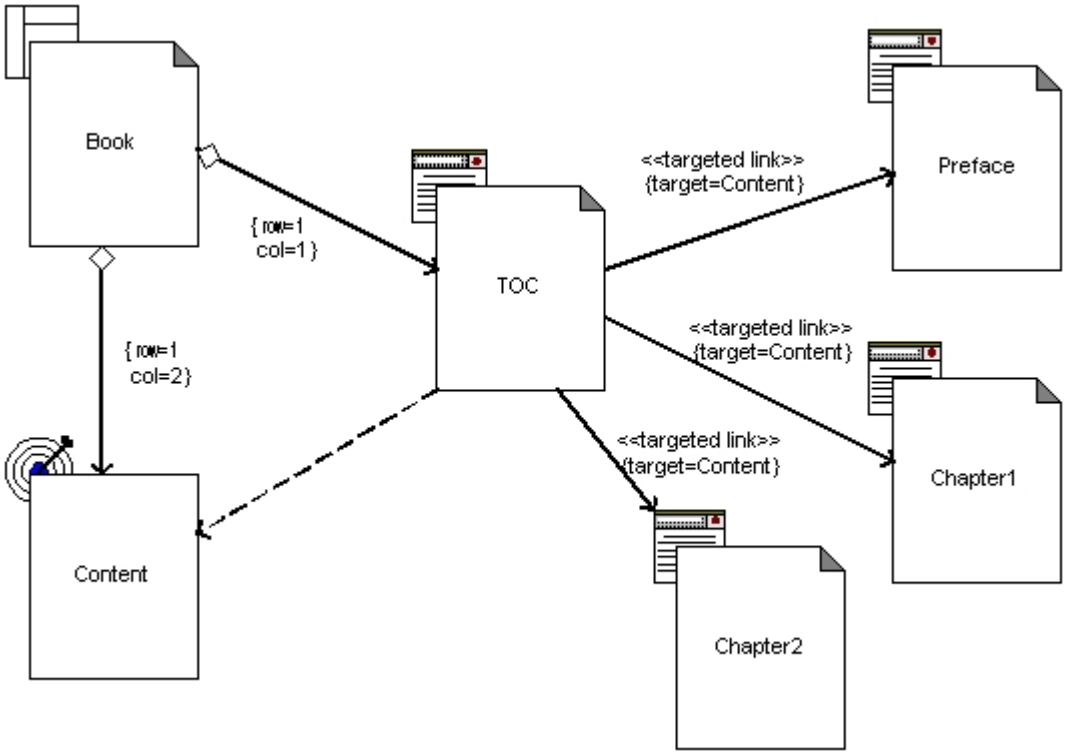

Figura 6. Esempi di cornici

La maggior parte delle specifiche della presentazione effettiva viene catturata dai valori con tag all'interno del frameset e delle associazioni. Due valori con tag sulla relazione di aggregazione tra un frameset e un target, o una pagina client, specificano la riga e la colonna del frameset cui appartiene il target, o la pagina. Il valore con tag “Target” sull'associazione di link di destinazione identifica il ½target╗ in cui deve essere restituita la pagina.

Quando un target non è aggregato ad un frameset, viene utilizzata un'istanza di browser separato per restituire la pagina. È importante ricordare che questa notazione esprime una singola istanza di una macchina client. Target multipli e indipendenti sono considerati tutti eseguibili sulla stessa macchina e il diagramma rappresenta il comportamento lato client dell'istanza client uno. Per una migliore comprensione, è necessario che ogni altra configurazione di distribuzione venga documentata nel modello.

Conclusioni

Le idee e i concetti illustrati nel presente documento sono un'introduzione a problematiche e soluzioni per la modellazione di elementi specifici di applicazioni Web con UML. L'obiettivo di questo lavoro è presentare un metodo completo e coerente per integrare la modellazione di elementi specifici del Web con il resto dell'applicazione, in modo che i livelli di dettaglio e astrazione risultino appropriati per progettisti, implementatori e architetti di applicazioni Web. Sta per essere terminata una prima versione dell'estensione formale ad UML per applicazioni Web, che fornirà ad architetti e progettisti un metodo comune per rappresentare la completezza della progettazione delle applicazioni Web con UML.

Le informazioni più recenti su questa estensione si trovano in Internet, consultando le sezioni dedicate a Rational Rose e UML del sito Web del Software di Rational.

Rational®
the software development company

Sedi principali:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Numero verde: (800) 728-1212
E-mail: info@rational.com
Sito Web: www.rational.com
Sito internazionale: www.rational.com/worldwide