Modélisation d'architectures d'application Web avec UML

Jim Conallen

Livre blanc de Rational Software

TP 157, 6/99

Une version de ce document a été diffusée en octobre 1999 (volume 42, numéro 10), issue de Communications de l'ACM.

Rational®
the software development company

Table des matières

Résumé

Les applications Web deviennent de plus en plus complexes et indispensables au fonctionnement de l'entreprise. Afin de mieux gérer cette complexité, elles doivent être modélisées. UML (Unified Modeling Language) est le langage standard utilisé pour la modélisation des systèmes hautement informatisés. Lors d'une tentative de modélisation des applications Web à l'aide du langage UML, il devient flagrant que certains de ces composants ne sont pas adaptés aux éléments de modélisation UML standard. Afin d'adopter une notation de modélisation pour tout le système (composants Web et composants intermédiaires traditionnels), le langage UML doit être étendu. Ce document présente une extension du langage UML, mettant en oeuvre son mécanisme d'extension formel. L'extension est conçue de telle sorte que des composants spécifiques au Web puissent être intégrés au reste du modèle de système et que le niveau adéquat d'abstraction et de détails nécessaires aux concepteurs, aux implémenteurs et aux architectes d'applications Web soit affiché.

Présentation

Il y a quelques années, le nouveau terme "Application Web" a été ajouté au vocabulaire informatique. Il semblerait que toute personne impliquée dans des systèmes logiciels métier envisage de construire des applications Web, engageant également de nombreux efforts informatiques non liés au métier. Pour de nombreuses personnes ayant depuis longtemps adopté cette architecture, le terme Application Web, tout comme les systèmes eux-mêmes, a évolué des petits programmes complémentaires de site Web aux robustes applications à plusieurs niveaux. Il n'est pas rare pour une application Web de prendre en charge des dizaines de milliers d'utilisateurs simultanés, répartis dans le monde entier. La conception d'architectures d'applications Web est un travail sérieux.

Si l'on interroge différentes personnes, le terme Application Web a des significations légèrement différentes. Certains pensent qu'une application Web est quelque chose qui utilise le langage Java, d'autres considèrent que les applications Web sont quelque chose qui utilise un serveur Web. Le consensus général se trouve quelque part entre les deux. Dans ce document, nous allons définir librement une application Web comme étant un système Web (serveur Web, réseau, HTTP, navigateur) dans lequel une entrée utilisateur (navigation et saisie de données) affecte l'état métier. Cette définition tente d'établir qu'une application Web est un système logiciel présentant un état métier et que son système frontal est en grande partie fourni via un système Web.

L'architecture générale d'une application Web est celle d'un système serveur client, avec quelques distinctions cependant. L'un des principaux avantages d'une application Web est son déploiement. Le déploiement d'une application Web est généralement une question de configuration des composants côté serveur sur un réseau. Aucun logiciel, ni configuration spécifique n'est requis de la part du client. Une autre différence importante réside dans la nature des communications client-serveur. Le principal protocole de communication d'une application Web est HTTP. Il s'agit d'un protocole sans connexion qui a été conçu pour la robustesse et la tolérance aux pannes, et non pour un débit de communications maximal. Les communications entre un client et un serveur dans une application Web tournent généralement autour de la navigation dans les pages Web, et non de communications directes entre objets côté serveur et côté client. A un certain niveau d'abstraction, tous les messages d'une application Web peuvent être décrits comme la demande et la réception d'entités de page Web. De manière générale, l'architecture d'une application Web n'est pas très différente de celle d'un site Web dynamique.

Les différences entre une application Web et un site Web, même s'il est dynamique, impliquent son utilisation. Des applications Web mettent en oeuvre la logique applicative et son utilisation modifie l'état métier (tel qu'il est capturé par le système). Ceci est important car l'effort de modélisation est ainsi mis en évidence. Des applications Web exécutent la logique applicative et en contrepartie, les modèles les plus importants du système mettent l'accent sur la logique applicative et l'état métier et non sur les détails de présentation. La présentation est importante (sinon, le système ne serait bon pour personne). Cependant, vous devez vous efforcer à séparer nettement les problèmes métier de ceux de présentation. Si les problèmes de présentation sont importants, ou même compliqués, ils doivent eux aussi être modélisés, mais pas nécessairement comme partie intégrante du modèle de logique applicative. En outre, les ressources qui travaillent sur la présentation tendent à être plus artistiques que concernées par l'implémentation des règles métier.

RMM (Relationship Management Methodology) est une méthodologie/notation associée au développement des systèmes Web. Il s'agit d'une méthodologie pour la conception, la construction et la maintenance de systèmes Web intranet et Internet. Son principal objectif est de réduire les coûts de maintenance des sites Web dynamiques, gérés par base de données. Il préconise une représentation visuelle du système pour faciliter les discussions relatives à la conception. Il s'agit d'un processus itératif qui inclut la décomposition des éléments visuels contenus dans les pages Web et leur association aux entités de base de données. RMM est une approche “facilitant” la création et la maintenance de sites Web dynamiques.

Cependant, cette méthodologie est insuffisante pour la construction d'applications Web. Ces dernières, étant axées sur la logique applicative, incluent un certain nombre de mécanismes technologiques destinés à l'implémentation de la logique

applicative qui n'est pas traitée de manière adéquate par la notation RMM. Des technologies telles que le scriptage côté client, les applets et les contrôles ActiveX contribuent largement à l'exécution des règles métier du système. De plus, les applications Web peuvent être utilisées comme mécanisme de livraison pour un objet système réparti. Les applets et les contrôles ActiveX peuvent contenir des composants qui interagissent de manière asynchrone avec les composants côté serveur via RMI ou DCOM, indépendamment du serveur Web. Des applications sophistiquées utilisent également plusieurs instances et cadres de navigateur sur le client, établissant et gérant ainsi leurs propres mécanismes de communication.

Comme tous ces mécanismes participent à la logique applicative du système, ils doivent être modélisés en tant que tels. En outre, du fait qu'ils ne représentent qu'une partie de la logique applicative, ils doivent être intégrés au reste des modèles du système. Dans de nombreux cas, la masse de la logique applicative s'exécute derrière le serveur Web dans l'un des niveaux côté serveur. Le choix de la notation et du langage de modélisation est généralement décidé par les besoins de ce côté de l'application. Grâce à l'acceptation du langage UML par le groupe OMG comme langage officiel de modélisation d'objets, un nombre croissant de systèmes est exprimé avec la notation UML. Pour de nombreuses personnes, UML est le langage de choix pour la modélisation des systèmes hautement informatisés. La principale question de modélisation des applications Web est alors la suivante : “Comment exprimer la logique applicative exécutée dans mes composants spécifiques au Web avec le reste de mon application ?” La réponse réside dans notre capacité à exprimer l'exécution de la logique applicative du système dans les technologies et les éléments spécifiques au Web à l'aide du langage UML.

Ce document est une introduction aux problèmes et solutions possibles pour la modélisation d'applications Web. Il met l'accent sur les composants architecturalement importants, spécifiques aux applications Web et sur leur modélisation avec UML. Il est présumé que le lecteur possède des connaissances sur le langage UML, les entités orientées objet et le développement d'applications Web. Le travail décrit dans ce document est basé sur des hypothèses relativement inoffensives.

- Les applications Web sont des systèmes hautement informatisés qui deviennent de plus en plus complexes et qui s'investissent dans des rôles de plus en plus indispensables aux activités.
- Un moyen de gérer la complexité des systèmes informatiques consiste à les réduire et à les modéliser.
- Un système informatique dispose généralement de plusieurs modèles, représentant chacun un point de vue différent, un niveau d'abstraction et de détails.
- Le niveau adéquat d'abstraction et de détails dépend des artefacts et des activités du processus de développement.
- Le langage de modélisation standard des systèmes hautement informatisés est le langage UML (Unified Modeling Language).

Un traitement plus complet des concepts et des idées abordés dans ce document est en cours de développement pour le futur manuel : “Building Web Applications with UML” devant être publié dans la série Object Technology Series par Addison Wesley Longman cette année.

Modélisation

Les modèles permettent de mieux comprendre le système en simplifiant certains détails. Le choix de ce qui est modélisé a un énorme effet sur la compréhension du problème et de la forme de la solution. Les applications Web, tout comme d'autres systèmes hautement informatisés, sont généralement représentées par un ensemble de modèles : modèles de cas d'utilisation, d'implémentation, de déploiement, de sécurité, etc. Un modèle supplémentaire utilisé exclusivement par des systèmes Web est le plan du site, une abstraction de pages Web et de chemins de navigation à travers le système.

La plupart des techniques de modélisation mises en pratique aujourd'hui sont parfaitement adaptées au développement de divers modèles d'application Web et ne nécessitent pas d'autre explication. Un modèle très important cependant, le modèle d'analyse/conception (ADM, Analysis/Design Model) présente quelques difficultés lorsqu'il est tenté d'ajouter des pages Web, ainsi que le code exécutable qui leur est associé, aux autres éléments du modèle.

Lors de la prise de décision du mode de modélisation, l'identification du niveau d'abstraction et de détails est primordiale pour aboutir à un résultat avantageux pour les utilisateurs du modèle. En règle générale, il est préférable de modéliser les artefacts du système : les entités de la vie réelle qui seront construites et manipulées pour générer le produit final. La modélisation de la structure interne du serveur Web, ou des détails du navigateur Web, n'aide pas les concepteurs et les architectes d'une application Web. La modélisation des pages, de leurs liens, de tout le contenu dynamique impliqué dans la création des pages et le contenu dynamique des pages une fois sur le client est important, très important. Ce sont ces artefacts que les concepteurs conçoivent et que les implémenteurs implémentent. Les pages, les hyperliens et le contenu dynamique sur le client et le serveur représentent ce qui doit être modélisé.

L'étape suivante est celle du mappage de ces artefacts sur les éléments de modélisation. Par exemple, les hyperliens mappent naturellement vers des éléments d'association du modèle. Un hyperlien représente un chemin de navigation d'une page à une autre. En étendant cette idée, les pages peuvent mapper vers des classes de la vue logique du modèle. Si une page Web est une classe du modèle, les scripts de la page mappent alors naturellement vers les opérations de la classe. Toutes les variables dont la portée est la page dans les scripts mappent vers des attributs de classe. Un problème se pose si vous considérez qu'une page Web peut contenir un ensemble de scripts qui s'exécute sur le serveur (préparant le contenu dynamique de la page) et un ensemble totalement différent de scripts qui s'exécute uniquement sur le client (JavaScript). Dans ce scénario, si nous examinons une classe de page Web dans le modèle, il y a confusion entre les opérations, les attributs et même les relations actifs sur le serveur (pendant la préparation de la page) et ceux qui sont actifs lorsque l'utilisateur interagit avec la page sur le client. En outre, une page Web telle qu'elle est fournie dans une application Web est véritablement mieux modélisée en tant que composant du système. Le simple mappage d'une page Web sur une classe UML ne permet pas de mieux comprendre le système.

Les créateurs du langage UML ont pris conscience du fait qu'il y aura toujours des cas dans lesquels le langage UML, à lui seul, est insuffisant pour capturer la sémantique adéquate d'une architecture ou d'un domaine particulier. Pour corriger ce problème, un mécanisme d'extension formel a été défini afin de permettre aux praticiens d'étendre la sémantique du langage UML. Le mécanisme permet de définir des *stéréotypes*, des *valeurs marquées* et des *contraintes* qui peuvent être appliqués pour modéliser des éléments.

Un *stéréotype* est un commentaire qui permet de définir une nouvelle signification de sémantique pour un élément de modélisation. Les *valeurs marquées* sont des paires de valeur de clé qui peuvent être associées à un élément de modélisation permettant de “marquer” une valeur sur un élément de modélisation. Les *contraintes* sont des règles qui définissent la syntaxe d'un modèle. Elles peuvent être exprimées sous forme de texte libre ou en langage OCL plus formel.

Les tâches abordées dans ce document introduisent une extension au langage UML pour les applications Web. Cette extension, dans sa totalité, dépasse la portée de ce document. Néanmoins, la plupart des concepts et des explications y sont traités.

Une dernière remarque sur la modélisation : une très nette distinction doit être faite entre la logique applicative et la logique de présentation. Dans le cas d'une application métier standard, seule la logique applicative doit faire partie de l'ADM. Les détails de présentation, tels que les boutons animés, l'aide contextuelle et autres extensions de l'interface utilisateur n'appartiennent généralement pas à l'ADM. Si un modèle d'interface utilisateur distinct est généré pour l'application, c'est donc l'endroit pour de tels éléments. L'ADM doit resté focalisé sur l'expression du problème métier et de l'espace de solution. A l'époque des artistes Web, la présentation d'une page Web est mieux conçue et implémentée par un spécialiste (graphiste technique) que par un développeur traditionnel.

Architecture d'application Web

L'architecture de base d'une application Web inclut des navigateurs, un réseau et un serveur Web. Les navigateurs demandent des “pages Web” au serveur. Chaque page est un mélange de contenu et d'instructions de formatage, exprimés en langage HTML. Certaines pages incluent des scripts côté client qui sont interprétés par le navigateur. Ces scripts définissent un comportement dynamique supplémentaire pour la page affichée et interagissent souvent avec le navigateur, le contenu de la page et des contrôles supplémentaires (applets, contrôles ActiveX et plug-ins) contenus dans la page. L'utilisateur visualise et interagit avec le contenu de la page. Parfois, l'utilisateur saisit des informations dans des champs de la page et les envoie au serveur pour traitement. L'utilisateur peut également interagir avec le système en naviguant dans différentes pages du système, grâce aux hyperliens. Dans l'un ou l'autre cas, l'utilisateur fournit des entrées au système qui sont susceptibles de modifier l'“état métier” du système.

Du point de vue du client, la page Web est toujours un document au format HTML. Sur le serveur, cependant, une “page Web” peut se manifester de différentes façons. Dans les anciennes applications Web, des pages Web dynamiques étaient générées à l'aide de l'interface CGI (Common Gateway Interface). CGI définit une interface pour les scripts et les modules compilés permettant d'obtenir l'accès aux informations transmises avec une demande de page. Dans un système CGI, un répertoire spécial est généralement configuré sur le serveur Web afin de pouvoir exécuter les scripts en réponse aux demandes de page. Lorsqu'un script CGI est demandé, le serveur, au lieu de renvoyer uniquement le contenu du fichier (comme ce serait le cas pour tout fichier au format HTML), traite ou exécute le fichier à l'aide de l'interpréteur adapté (généralement un interpréteur de commandes PERL) et renvoie la sortie au client demandeur. Le résultat final de ce traitement est un flux au format HTML qui est renvoyé au client demandeur. La logique applicative est exécutée dans le système lors du traitement du fichier. Pendant ce temps, il peut interagir avec les ressources côté serveur, comme des bases de données ou des composants intermédiaires.

Les serveurs Web d'aujourd'hui se sont améliorés par rapport à cette conception de base. Aujourd'hui, ils sont bien plus soucieux de la sécurité et présentent des fonctionnalités comme la gestion de l'état du client sur le serveur, l'intégration du traitement des transactions, l'administration à distance et la mise en pool des ressources pour n'en nommer que quelques-unes. Collectivement, la génération la plus récente de serveurs Web répond aux questions importantes, posées aux architectes d'applications robustes, évolutives et indispensables à l'activité de l'entreprise.

Si l'on considère le rôle des scripts CGI, les serveurs Web d'aujourd'hui peuvent être divisés en trois catégories principales : pages scriptées, pages compilées et un hybride des deux. Dans la première catégorie, chaque page Web pouvant être demandée par le navigateur d'un client est représentée dans le système de fichiers du serveur Web sous forme de fichier script. Ce fichier est généralement un mélange de langage HTML et d'un autre langage de script. Lorsque la page est demandée, le serveur Web délègue le traitement de cette page à un moteur qui la reconnaît, aboutissant au renvoi d'un flux en format HTML au client demandeur. Les pages Active Server de Microsoft, les pages JSP (Java Server Pages) et les pages Cold Fusion en sont des exemples.

Dans la deuxième catégorie, celle des pages compilées, le serveur Web charge et exécute un composant binaire. Ce composant, comme pour les pages scriptées, accède à toutes les informations fournies avec la demande de page (valeurs des champs de formulaire et des paramètres). Le code compilé utilise les détails de la demande et accède généralement aux ressources côté serveur pour générer le flux HTML renvoyé au client. Bien que ce ne soit pas une règle, les pages compilées tendent à couvrir une fonctionnalité plus large que les pages scriptées. En transmettant des paramètres à la demande de page compilée, une fonctionnalité différente peut être obtenue. Chaque composant compilé peut en fait inclure la fonctionnalité complète des pages scriptées d'un répertoire entier. Les technologies ISAPI de Microsoft et NSAPI de Netscape représentent ce type d'architecture.

La troisième catégorie représente des pages scriptées qui, une fois demandées, sont compilées. Cette version compilée est alors utilisée par toutes les demandes ultérieures. La page subit une autre compilation, uniquement si le contenu de la page d'origine change. Cette catégorie est un compromis entre la flexibilité des pages scriptées et l'efficacité des pages compilées.

Modélisation de pages Web

Les pages Web, scriptées ou compilées, mappent une à une vers des composants en langage UML. Un composant est une partie “physique” et remplaçable du système. La vue d'implémentation (Vue composant) du modèle décrit les composants du système et leurs relations. Dans une application Web, cette vue décrit toutes les pages Web du système et les relations qu'elles ont entre elles (hyperliens). A un certain niveau, un diagramme de composants est similaire à un plan de site.

Comme les composants ne représentent que le regroupement physique d'interfaces, ils sont plus adaptés à la modélisation des collaborations au sein des pages. Ce niveau d'abstraction, extrêmement important pour le concepteur et l'implémenteur, doit cependant faire partie du modèle. Pour commencer, nous pouvons dire que chaque page Web est une classe UML dans la vue de conception du modèle (Vue logique) et que ses relations avec les autres pages (associations) représentent des hyperliens. Cependant, cette abstraction perd sa signification si vous considérez que toute page Web donnée peut représenter un ensemble de fonctions et de collaborations qui existent uniquement sur le serveur et qu'un ensemble totalement différent peut exister uniquement sur le client. Un exemple d'une telle page peut être une page Web scriptée du serveur qui utilise un langage DHTML (scriptage côté client) comme élément de sortie. La réaction en chaîne à ce problème peut être le stéréotypage de chaque attribut ou opération de la classe pour indiquer s'il était ou non valide du côté serveur ou client. A ce point, notre modèle, destiné à l'origine à simplifier les choses, devient relativement complexe.

Une meilleure approche du problème consiste à envisager le principe de la “séparation des couches”. Logiquement, le comportement d'une page Web sur le serveur est complètement différent de celui sur le client. Lors de son exécution sur le serveur, elle a accès (c'est-à-dire, qu'elle est en relation avec) aux ressources côté serveur (composants intermédiaires, bases de données, système de fichiers, etc.). Cette même page, ou la sortie HTML résultant de cette page, présente un comportement et un ensemble de relations totalement différents sur le client. Sur le client, une page scriptée est en relation avec le navigateur via le modèle DOM (Document Object Model) et avec des applets Java, des contrôles ActiveX ou des plug-ins spécifiés par la page. Pour le concepteur expérimenté, il peut y avoir des relations supplémentaires avec d'autres pages “actives” sur le client qui s'affichent dans un autre cadre HTML ou une autre instance de navigateur.

En séparant les couches, il est possible de modéliser l'aspect côté serveur d'une page Web avec une classe et l'aspect côté client avec une autre. Nous distinguons les deux grâce au mécanisme d'extension du langage UML afin de définir des stéréotypes et des icônes destinés à chaque «server page» et «client page». En langage UML, les stéréotypes permettent de définir une nouvelle sémantique pour un élément de modélisation. Les classes stéréotypées peuvent être rendues dans un diagramme UML avec une icône personnalisée ou simplement avec le nom du stéréotype apposé entre guillemets («»). Les icônes sont utiles aux diagrammes de présentation, alors que l'utilisation de simples balises est préférable si des attributs et des opérations de classe sont exposés.

Pour les pages Web, les stéréotypes indiquent que la classe est une abstraction du comportement logique d'une page Web sur le client ou le serveur. Les deux abstractions sont liées l'une à l'autre par une relation directionnelle placée entre les deux. Cette association est stéréotypée sous le nom de «build» (construction), du fait que l'on peut dire qu'une page de serveur génère une page client (voir figure 1). Chaque page Web dynamique (autrement dit, les pages dont le contenu est déterminé au moment de l'exécution) est constituée à partir d'une page du serveur. Chaque page du client est au plus générée par une page de serveur, cependant une seule page de serveur peut générer plusieurs pages de client.

La relation commune entre les pages Web est l'hyperlien. Un hyperlien dans une application Web représente un chemin de navigation à travers le système. Cette relation s'exprime dans le modèle par une association stéréotypée «link» (lien). Cette association tire toujours son origine d'une page client et pointe vers une page du client ou du serveur.

Les hyperliens sont implémentés dans le système comme demande de page Web et les pages Web sont modélisées sous forme de composants dans la vue d'implémentation. Une association de lien à une page de client est quasiment équivalente à celle du lien à la page de serveur qui génère la page du client. Ceci est dû au fait qu'un lien est en fait une demande de page, et non des abstractions de classe. Comme un composant de page Web réalise les deux abstractions de page, un lien à l'une des classes réalisé par le composant de page est équivalent.

Les valeurs marquées sont utilisées pour définir les paramètres transmis avec une demande de lien. La valeur marquée de l'association «link» “Paramètres” est une liste de noms de paramètres (et de valeurs facultatives) qui sont attendus et utilisés par la page de serveur qui traite la demande. A la figure 2, la page SearchResults contient un nombre variable d'hyperliens (0..*) avec la page de serveur GetProduct dans laquelle chaque lien a une valeur différente pour le paramètre productId. La page GetProduct génère la page ProductDetail du produit spécifié par le paramètre productId.

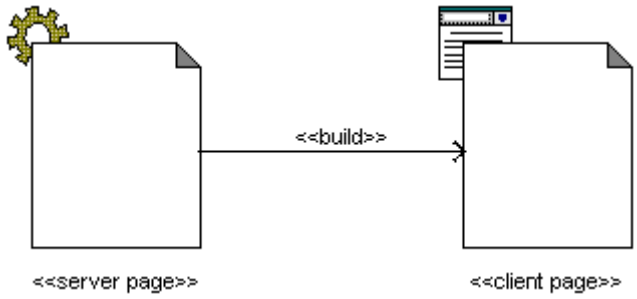

Figure 1. Les pages de serveur génèrent les pages de client

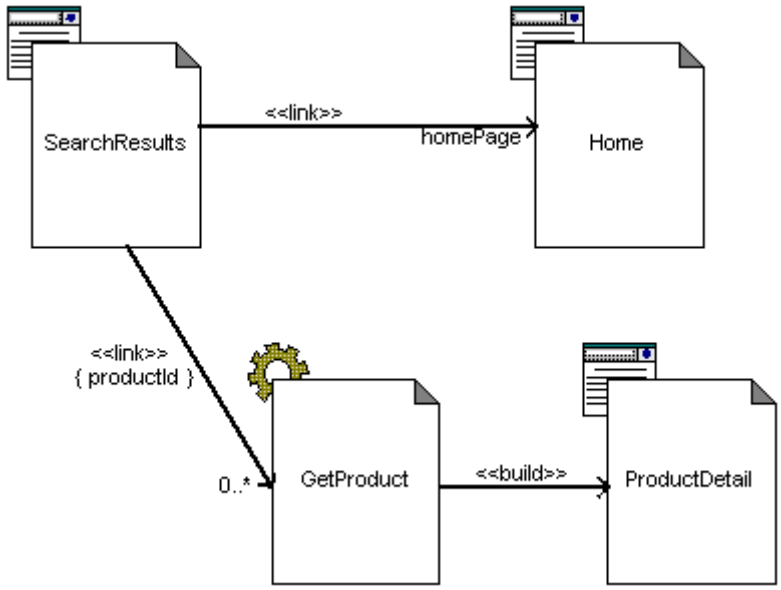

Figure 2. Utilisation des paramètres d'hyperlien

L'utilisation de ces stéréotypes facilite la modélisation des scripts et des relations d'une page. Les opérations de la classe «server page»deviennent des fonctions dans les scripts côté serveur de la page et les attributs deviennent des variables de portée page (globalement accessibles par les fonctions de la page). Les opérations et les attributs de la classe «client page» deviennent de même des fonctions et des variables visibles sur le client. Le principal avantage de la séparation des aspects côté serveur et côté client d'une page dans des classes différentes réside dans les relations entre les pages et d'autres classes du système. Les pages de client sont modélisées avec des relations aux ressources côté client : modèle DOM, applets Java, contrôles ActiveX et plug-ins (voir figure 3). Les pages de serveur sont modélisées à l'aide de relations avec les ressources côté serveur : composants intermédiaires, composants d'accès aux bases de données, système d'exploitation serveur, etc. illustrés à la figure 4.

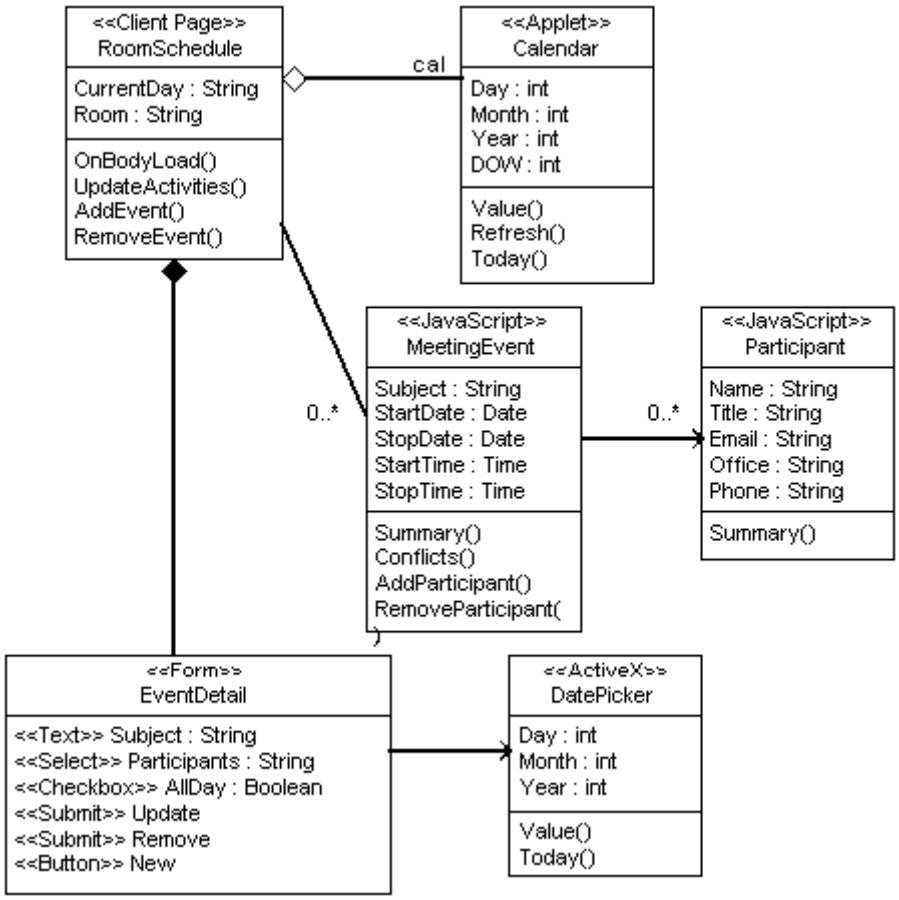

Figure 3. Collaborations client

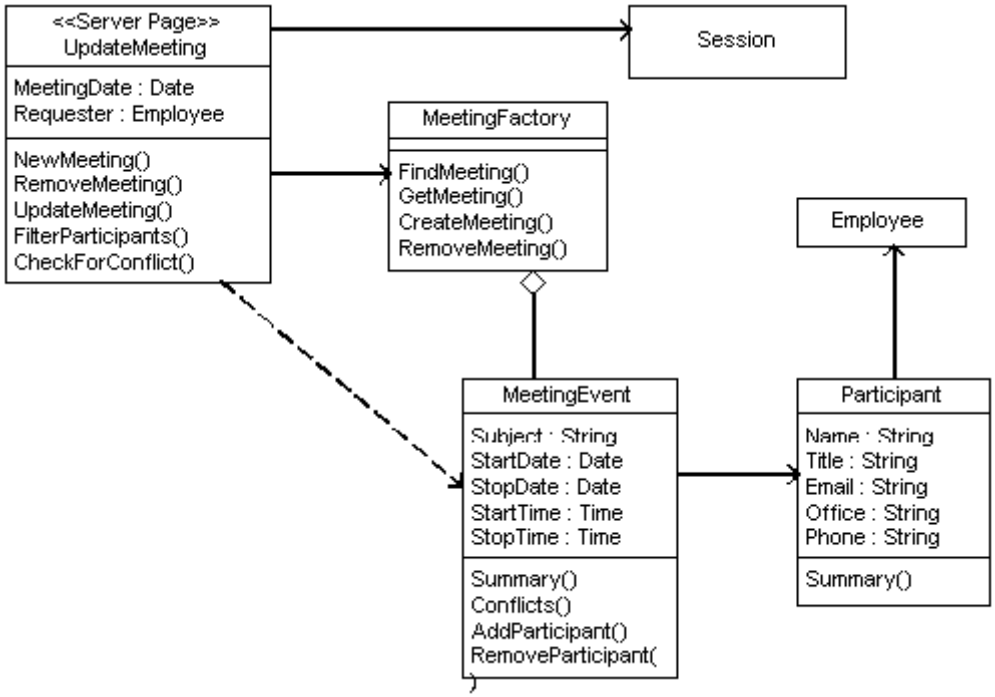

Figure 4. Collaborations serveur

L'un des plus gros avantages présentés par l'utilisation de stéréotypes de classe pour modéliser les comportements logiques des pages Web est le fait que leurs collaborations avec les composants côté serveur peuvent être exprimées sensiblement de la même manière que toute autre collaboration côté serveur. La classe «server page» est simplement une autre classe qui participe à la logique applicative du système. A un niveau plus conceptuel, les pages de serveur jouent généralement le rôle de contrôleurs, orchestrant l'activité d'objet métier nécessaire à l'accomplissement des objectifs métier initialisés par la demande de page du navigateur.

Du côté client, les collaborations peuvent devenir un peu compliquées. Ceci est en partie dû à la variété de technologies qui peut être employée. Une page client, à son stade le plus simple, est un document HTML contenant des informations de contenu et de présentation. Les navigateurs affichent les pages HTML à l'aide des instructions de formatage contenues dans la page, parfois à l'aide de feuilles de style distinctes. Dans le modèle logique, cette relation peut être exprimée avec une dépendance d'une page client en classe stéréotypée «Style Sheet». Cependant, les feuilles de style abordent principalement les questions de présentation et sont souvent exclues de l'ADM.

Formulaires

Le formulaire constitue le principal mécanisme de saisie des données. Les formulaires sont définis dans un document HTML à l'aide des balises <form>. Chaque formulaire spécifie la page à laquelle il se soumet. Un formulaire contient un certain nombre d'éléments en entrée, tous exprimés sous forme de balises HTML. Les balises les plus courantes sont les balises <input>, <select> et <textarea>. La balise d'entrée est quelque peu surchargée puisqu'il peut s'agir d'une zone de texte, d'une case à cocher, d'un bouton d'option, d'un bouton de fonction, d'une image, d'une zone masquée, ainsi que quelques autres types moins courants. La modélisation des formulaires signifie un autre stéréotype de classe : «Form». Une classe «Form» n'a pas d'opération, du fait que les opérations qui peuvent être définies dans une balise <form> appartiennent en fait à la page client. Les éléments d'entrée d'un formulaire sont tous des attributs stéréotypés de la classe «Form». Elle peut avoir des relations avec des applets ou des contrôles ActiveX qui agissent comme commandes d'entrée. Chaque formulaire a également une relation avec une page de serveur, celle qui traite la soumission du formulaire. Cette relation est stéréotypée «submit». Comme les formulaires sont totalement inclus dans un document HTML, ils sont exprimés dans un diagramme UML avec une forte forme d'agrégation. La figure 5 illustre une simple page de panier électronique qui définit un formulaire et montre la relation de soumission à la page du serveur de traitement.

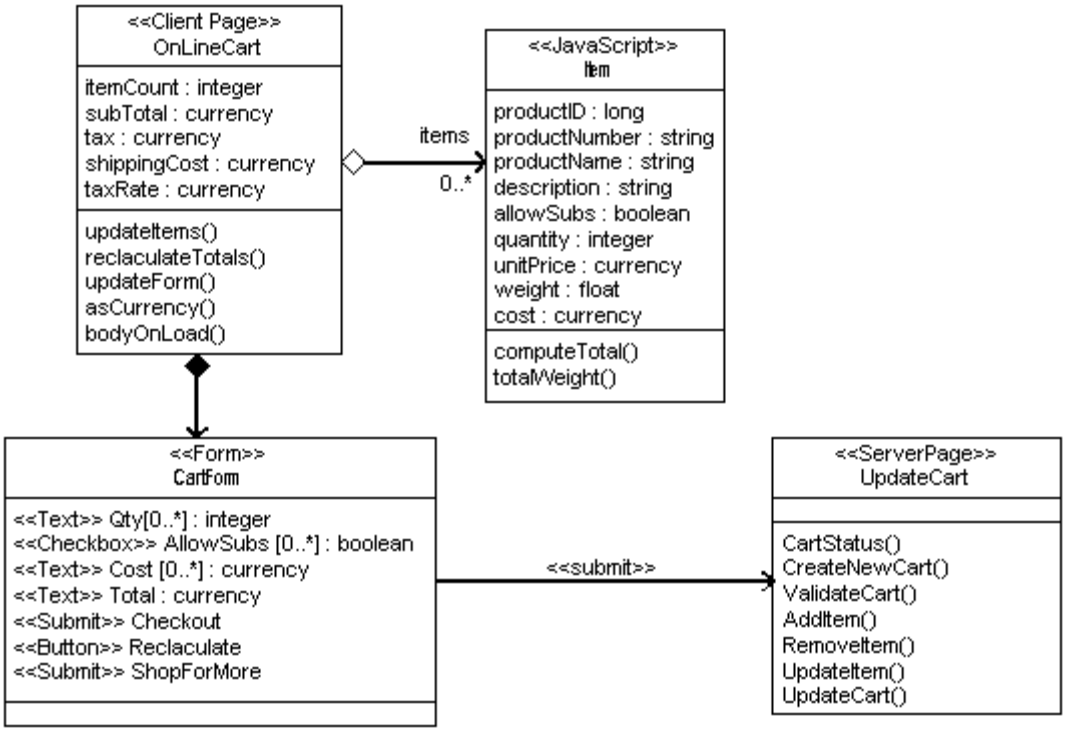

Figure 5. Formulaires soumis aux pages de serveur

A la figure 5, la classe stéréotypée «JavaScript» est un objet représentant les articles contenus dans le panier électronique. Une syntaxe en tableau est utilisée dans la description des propriétés du formulaire pour les zones dont le nombre d'instances est variable. Dans le cas de ce panier électronique, cela signifie qu'il peut contenir aucun ou de nombreux articles, avec pour chacun d'eux un élément <input> Quantité, Prix, Total...

Comme toute l'activité de la page client est exécutée en JavaScript qui est un langage moins typé, les types de donnés spécifiés pour ces attributs sont utilisés uniquement dans un but de clarification, destinée à l'implémenteur. Lorsqu'il est implémenté en JavaScript ou sous forme de balises d'entrée HTML, le type est ignoré. Ceci s'applique également aux paramètres de fonction, qui bien qu'ils n'apparaissent pas totalement dans cette illustration, font partie du modèle.

Cadres

L'utilisation de cadres HTML dans un site ou une application Web est un sujet de discussion depuis son introduction. Les cadres permettent à plusieurs pages d'être actives et visibles par l'utilisateur à tout moment. Le jeu de fonctionnalités le plus récent pour les navigateurs les plus courants aujourd'hui permet également à plusieurs instances de navigateur d'être actives sur la machine utilisateur. Grâce aux composants et aux scripts HTML, ces pages peuvent interagir entre elles. Le potentiel d'interactions complexes sur le client est important et le besoin de modélisation encore plus grand.

L'utilisation de cadres ou de plusieurs instances de navigateur dans une application, relève de la décision de l'architecte logiciel. Si c'est le cas, alors pour les mêmes raisons que celles évoquées précédemment, le modèle de comportement côté client doit être représenté dans l'ADM. Pour modéliser l'utilisation de cadre, nous avons défini deux classes supplémentaires, «frameset» et «target», et un stéréotype d'association «targeted link». Une classe d'ensemble de cadres représente un objet conteneur et mappe directement sur une balise HTML <frameset>. Elle contient des pages client et des cibles. Une classe cible est un cadre ou une instance de navigateur désigné, référencé par d'autres pages client. Une association de lien ciblé est un hyperlien vers une autre page, mais qui est rendue dans une cible spécifique. Dans l'exemple illustré à la figure 6, une vue d'aperçu courant est présentée dans un navigateur utilisant deux cadres. Un cadre est désigné par une cible (Contenu), tandis que l'autre contient simplement une page client. Ce cadre de page client correspond à la table des matières du manuel. Les hyperliens contenus dans cette page sont ciblés, de façon à s'afficher dans le cadre Contenu. Il en résulte une table des matières statique sur le côté gauche de la page et le contenu des manuels, chapitre par chapitre, sur le côté droit.

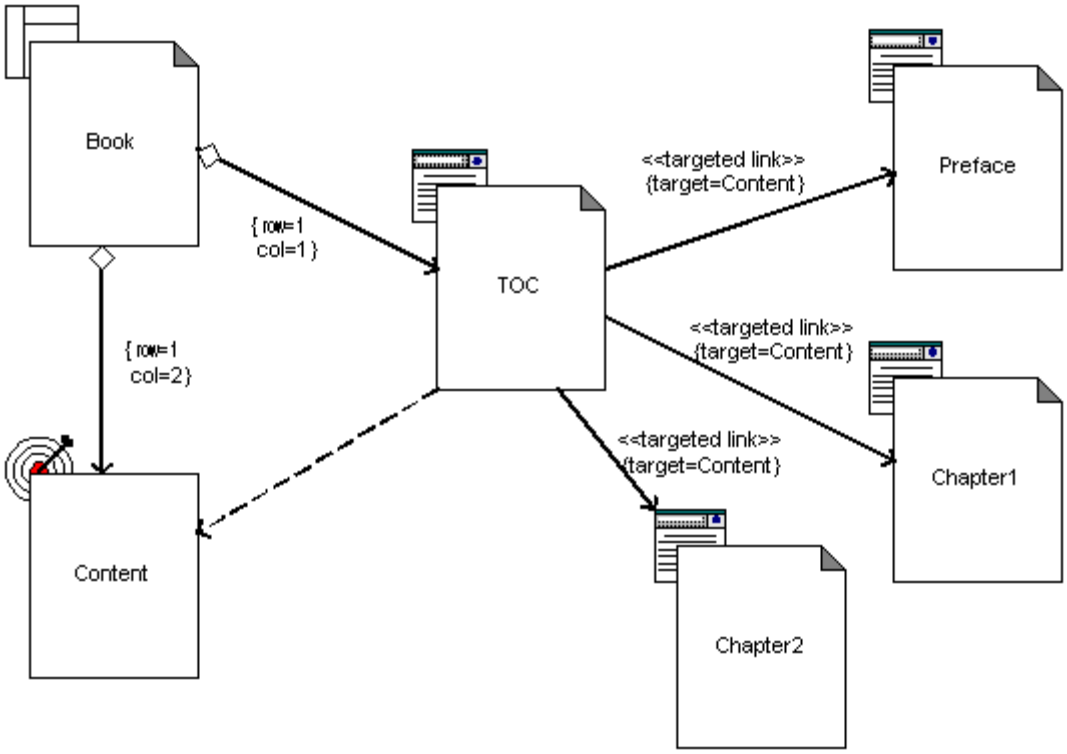

Figure 6. Exemple de cadres

Un grand nombre des caractéristiques réelles de présentation est capturé par des valeurs marquées dans l'ensemble de cadres et les associations. Deux valeurs marquées dans la relation d'agrégation entre un ensemble de cadres et une cible, ou une page client, spécifient la ligne et la colonne de l'ensemble de cadres auquel appartient la cible ou page. La valeur marquée “Cible” dans l'association de lien marqué identifie la cible dans laquelle la page doit être affichée.

Lorsqu'une cible n'est pas agrégée avec un ensemble de cadres, cela signifie qu'une instance de navigateur distincte est utilisée pour rendre les pages. Il est important de ne pas oublier que cette notation exprime une seule instance d'une machine client. Plusieurs cibles indépendantes sont toutes présumées s'exécuter sur la même machine. Le diagramme exprime le comportement côté client d'une instance client. Toute autre configuration de déploiement doit être largement documentée dans le modèle pour une meilleure compréhension.

Conclusion

Les idées et les concepts abordés dans ce document constituent une introduction aux problèmes et solutions de modélisation d'éléments spécifiques à une application Web en langage UML. Le but de ce travail est de présenter un moyen cohérent et complet d'intégration de la modélisation des éléments spécifiques au Web avec le reste de l'application, de sorte que les niveaux de détails et d'abstraction soient appropriés pour les concepteurs, implémenteurs et architectes d'applications Web. Une première version d'une extension formelle au langage UML pour les applications Web est presque terminée. Cette extension fournira un moyen commun aux architectes et aux concepteurs pour exprimer la totalité de la conception de leurs applications Web en langage UML.

Les informations les plus récentes relatives à cette extension figurent sur Internet aux sections Rational Rose et UML du site Web de Rational Software.

Rational
the software development company

Sièges :

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

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

Appel gratuit : (800) 728-1212
Adresse électronique : info@rational.com
Site Web : www.rational.com
Sites internationaux : www.rational.com/worldwide

Rational, le logo Rational et Rational Unified Process sont des marques de Rational Software Corporation aux Etats-Unis et/ou dans certains autres pays. Microsoft, Microsoft Windows, Microsoft Visual Studio, Microsoft Word, Microsoft Project, Visual C++ et Visual Basic sont des marques de commerce ou des marques déposées de Microsoft Corporation. Tous les autres noms ne sont utilisés qu'à des fins d'identification et sont des marques de commerce ou des marques déposées de leurs sociétés respectives. TOUS DROITS RESERVES. Rédigé aux Etats-Unis.

© Copyright 2002 Rational Software Corporation.
Document susceptible d'être modifié sans préavis.