Secteur Tertiaire Informatique
Filière étude - développement.

Développer une application n-tier

Positionner les composants de la couche présentation dans l'architecture x-tiers

Identifier les technologies de la couche présentation

Accueil

Apprentissage

Période en entreprise

Evaluation

SOMMAIRE

I ACRONYMES UTILISES

AJAX	Asynchronous Javascript and Xml
DOM	Document Object Model **(**Modèle Objet de Document)
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
IHM	Interface Homme Machine
JMS	Java Message Services
JSF	Java Server Faces
JSTL	Java Standard Tag Library
MVC	Modèle Vue Contrôleur
RCP	Rich Client Platform
RDA	Rich Desktop Applications
RIA	Rich Internet Applications
RMI	Remote Method Invocation
XML	eXtensible Markup Language
XUL	Extensible User Interface Language
WYSIWYG	What You See Is What You Get

II OBJECTIFS

L'objectif de ce document est de vous présenter :

- Les différents types d'applications clientes qui peuvent être implémentés au niveau de la couche présentation
- Les technologies concernées par chaque type d'application cliente
- La couche présentation d'une application WEB s'exécutant au sein d'un environnement JEE
- Une première approche des design patterns qui peuvent être mis en œuvre au niveau de la couche présentation
- Les principaux frameworks qui peuvent être utilisés pour faciliter l'implémentation de cette couche.

III TYPES DE CLIENT

Client léger, lourd, riche, … et pourquoi pas semi-lourd, semi-léger ou obèse. Il est difficile de mettre une définition claire sur ces termes. L'objectif de ce paragraphe est de définir plus précisément ces termes et de préciser les technologies qui se cachent derrière.

Trois types d'applications clientes sont majoritairement utilisés :

- Les clients légers
- Les clients lourds
- Les clients riches

III.1 CLIENT LEGER

Le terme « client léger » (thin client), né avec les technologies du Web, désigne une application accessible via une interface web (en HTML) consultable à l'aide d'un navigateur

web, où la totalité de la logique métier est traitée du côté du serveur. Pour ces raisons, le navigateur est parfois appelé client universel.
Le fait que l'essentiel des traitements soit réalisé du côté du serveur et que l'interface graphique soit envoyée au navigateur à chaque requête permet une grande souplesse de mise à jour. En contrepartie, l'application doit s'affranchir des différences d'interprétation du code HTML par les différents navigateurs (Internet Explorer, Firefox, Opera, …) et l'ergonomie de l'application possède un champ réduit.

Les technologies pour le transport entre le serveur et le client sont HTTP et HTTPS.

Les technologies pour la génération et le traitement de la présentation sont présentes à la fois sur le client mais aussi sur le serveur.

Coté client (navigateur), les technologies utilisées sont :

- HTML/XHTML : langage de description des pages Web.
- Feuilles de style (CSS) : langage de présentation des pages Web
- Javascript : langage de programmation de scripts qui permet de donner de l'interactivité aux pages web
- Applet : application Java qui s'exécute au sein d'un navigateur possédant sa propre JVM.

Coté serveur, les composants JEE concernés sont les servlets et les JSP. Il est également possible d'utiliser des librairies JSTL pour faciliter le développement des pages JSP.

III.2 CLIENT LOURD

Le terme « client lourd » (fat client ou heavy client), par opposition au client léger, désigne une application cliente graphique exécutée sur le système d'exploitation du poste de l'utilisateur. Un client lourd possède généralement des capacités de traitement évoluées et peut posséder une interface graphique sophistiquée. Néanmoins, ceci demande un effort de développement et tend à mêler la logique de présentation (l'interface graphique) avec la logique applicative (les traitements).

Par rapport au client léger, les principaux inconvénients de ce type de client sont le déploiement et l'accès au serveur aux travers des pare-feux (firewall).

La communication avec la partie serveur peut se faire par différents moyens disponibles dans le cadre de la plateforme JEE : RMI, Web Services, voire JMS. Une telle application peut d'ailleurs elle-même être exécutée sous la forme d'une applet dans le contexte d'un navigateur supportant une JVM. Cela peut en simplifier le déploiement même si cela pose des problèmes de compatibilité de version de JVM ou encore de gestion de politique de sécurité.

La plateforme JSE propose 2 APIs pour le développement de clients lourds : SWING et AWT.

III.3 CLIENT RICHE

A ce jour, nous pouvons classifier les clients riches en deux grandes catégories :

- Les Rich Internet Applications (RIA)

- Les Rich Desktop Application (RDA)

Ces deux catégories désignent clairement deux types de solutions distinctes sur un point bien précis : la nécessité d'installer (ou pas) un environnement d'exécution quelconque sur le poste client, environnement qui se résume à un simple navigateur dans le cas des clients RIA.

Pour être plus pragmatique, on dira qu'un client riche doit allier à la fois les qualités de déploiement des applications web actuelles, avec la richesse de l'ergonomie des applications de type client lourd.

III.3.1 CLIENTS RICHES DE TYPE RIA (Rich Internet Applications)

Les clients RIA recourent à un navigateur comme socle d'exécution mais aussi pour accéder aux services sous-jacents. Ils répondent aux contraintes des applications grand public, qui doivent reposer exclusivement sur des standards.

III.3.1.1 XUL

Précurseur dans le domaine des clients de type RIA, XUL (Prononcé "zool") est un langage XML de description d'interface graphique. Il est nativement mieux pourvu en composants graphiques (les widgets XUL) de haut niveau qu'HTML, et utilise, un modèle de programmation événementiel pour gérer les actions de l'utilisateur (un clic sur un bouton par exemple). Un point très positif est surtout la qualité de l'intégration du langage JavaScript.

XUL fait partie du framework Mozilla, il est donc disponible sur les dérivés comme Firefox.

XUL a été initialement conçu pour fonctionner dans un navigateur web. L'évolution technologique aidant, des extensions le rapprochant du principe RDA sont en train d'apparaître, comme XulRunner qui permet de lancer des applications XUL sans avoir besoin d'un navigateur comme Firefox.

III.3.1.2 AJAX

AJAX est le nom donné à une technique de conception de pages HTML. Cette technique a été initiée par Microsoft il y a quelques années.
Le principe de base d'AJAX se résume en un mécanisme visant à éviter de mettre à jour la page HTML complète à chacune des interactions utilisateur.
Initialement, les technologies HTTP et HTML permettent de n'associer qu'une seule action au clic d'un utilisateur sur un bouton : l'envoi d'une requête HTTP vers le serveur web, requête à laquelle le serveur répond en régénérant une page HTML contenant le tableau mis à jour. Le navigateur, à réception de cette réponse, se contente d'effacer la page actuellement à l'écran pour la remplacer par la nouvelle. Cela sous-entend donc que la réponse du serveur web contient bien les nouvelles données du tableau, mais elle inclut aussi l'ensemble des informations de présentation de la page le contenant.

Les problèmes qu'induit ce fameux mode page, sont :

- Un « scintillement » de l'interface graphique HTML à chaque nouvel événement utilisateur
- La nécessité d'implémenter des mécanismes de mémorisation de façon à ce que les champs annexes au tableau soient restitués tels qu'ils étaient préalablement.

AJAX permet donc de compléter ce mécanisme : tout événement sur les composants graphiques du client pourra se voir associer un comportement particulier. Ce comportement pourra être l'envoi d'une requête vers le serveur mais surtout une régénération partielle d'une partie de la page HTML avec les informations qu'il recevra en retour (dépourvues de toute information de présentation).
AJAX remplace le comportement naturel du navigateur en implémentant lui-même l'action associée à l'événement sur l'IHM et est en charge d'envoyer une requête HTTP au serveur web. Contrairement à une page HTML standard, ce n'est pas le navigateur qui reçoit la réponse mais la couche AJAX qui va simplement modifier le code HTML de la page courante afin de mettre à jour uniquement la partie correspondant au tableau concerné.

Techniquement, AJAX n'est en fait que l'utilisation conjointe d'un certain nombre de technologies existantes :

- Présentation basée sur les standards HTML et CSS
- Javascript pour intercepter les événements et envoyer la requête au serveur web
- Echanges de données au format XML. Dans AJAX, le X représente XML, et fait référence au format des données que l'on peut envisager d'utiliser pour permettre au serveur web d'alimenter les requêtes en provenance de la couche AJAX.
- Récupération asynchrone de données via un objet XMLHttpRequest et Javascript
- Javascript pour modifier l'arbre DOM qui est associé à la page courante

L'exemple le plus typique d'utilisation d'AJAX est Google Suggest de la page d'accueil de Google, qui complète la saisie en allant chercher des mots clés sur le serveur.

III.3.1.3 JSF

Il est difficile de parler d'AJAX sans évoquer un certain nombre de problèmes qui risquent de découler de son utilisation. Parmi ces problèmes, on peut en retenir deux :

- La complexité des pages : un mélange HTML, DOM, JavaScript rend difficile le débogage ainsi que la maintenance évolutive et corrective.
- Une réutilisation des composants basés sur AJAX assez délicate et finalement assez limitée.

Dans le monde JAVA, il existe un framework nommé JSF qui comprend un certain nombre de composants ayant des comportements de type AJAX. C'est un framework extensible qui vous permet de capitaliser sur des composants correspondant à vos besoins spécifiques.

JSF est une spécification JEE 5. Elle dispose d'un certain nombre d'implémentations qui met à disposition de multiples composants déjà créés. Parmi ces implémentations nous pouvons citer **Sun JSF RI** (l'implémentation de référence de Sun), **Apache MyFaces** et **Apache Tomahawk** (Apache Tomahawk dispose par exemple d'un composant calendrier déjà implémenté). Contrairement à Apache Struts, JSF se met en place assez facilement et demande moins de temps de formation car les concepts qui forment la base de JSF sont simples.

Ces type de framework permet d'espérer pouvoir répondre favorablement aux points négatifs cités précédemment concernant AJAX, en rendant le processus de développement des pages HTML beaucoup moins technique et plus assisté par les outils de développement.

III.3.1.4 FLASH

Flash d'Adobe est connu comme l'outil permettant de développer des animations qui rendent les pages web plus agréables. Il fonctionne suivant le principe d'un plugin du navigateur web qui doit être installé sur le poste qui va l'exécuter. L'image que l'on se fait de Flash est maintenant obsolète ! Il est devenu un outil de développement d'interface homme machine pourvu de composants graphiques de haut niveau (onglet, menu, menu déroulant,…), et d'un look très séduisant. Il bénéficie de plus d'un mécanisme de déploiement hérité de ses versions précédentes : simplement téléchargé par un navigateur et exécuté par le plugin flash (installé sur 95% des postes connectés à Internet).
Plusieurs technologies permettent aujourd'hui de développer une application FLASH, les principales étant :

- FLEX (Adobe – Payant). Pourvu d'un environnement de développement wysiwyg.
- Laszlo (Open Source). Un outil de développement fourni par IBM sous forme d'un plugin d'Eclipse.

Les deux solutions proposent le même type de résultat : une application flash. Elles se différencient simplement par leurs propriétés intrinsèques à savoir leur coût de licence et la qualité de l'outillage qui leur est associé.

III.3.2 CLIENTS RICHES DE TYPE RDA (Rich Desktop Application)

Plusieurs technologies partent sur une solution technique bien plus radicale : la fin de l'utilisation du navigateur web comme conteneur d'application. Mais, si le navigateur web n'est plus utilisé pour télécharger et afficher la couche de présentation, qui s'en charge ?
La réponse à cette question dépend de la technologie choisie. En tout état de cause, il est nécessaire que ce nouveau conteneur soit installé sur le poste qui exécute l'application, rendant non accessible l'application depuis un poste banalisé. Attention à ne pas assimiler le conteneur à l'application ! Le conteneur fournit un socle d'installation et déploiement, de mise à jour et d'exécution aux futures applications.
Parmi les solutions techniques de type RDA qui tiennent le haut de l'affiche, l'une des principales est celle d'**Eclipse RCP** (Rich Client Platform), on trouve aussi une version desktop du produit FLEX.

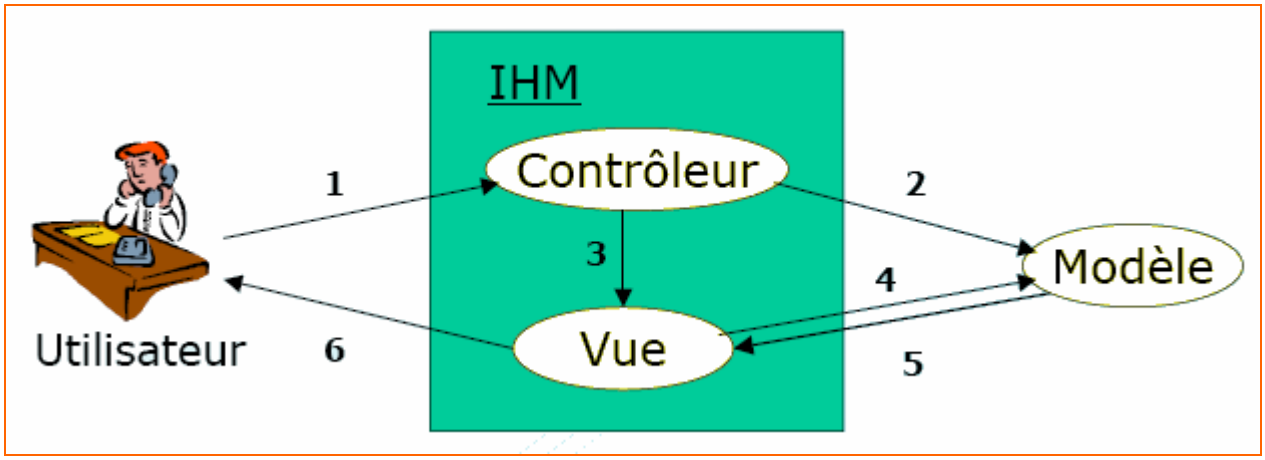
IV COUCHE PRESENTATION D'UNE APPLICATION WEB

IV.1.1 ARCHITECTURE MVC

L'architecture MVC est recommandée par SUN Microsystems comme architecture des applications WEB JEE.
Dans cette architecture, les responsabilités sont réparties de la façon suivante :

- Le modèle contient la logique et l'état de l'application.
- La vue représente l'interface utilisateur.
- Le contrôleur gère la synchronisation entre la vue et le modèle.
- Le contrôleur réagit aux actions de l'utilisateur en effectuant les actions nécessaires sur le modèle.
- Le contrôleur surveille les modifications du modèle et informe la vue des mises à jour nécessaires.

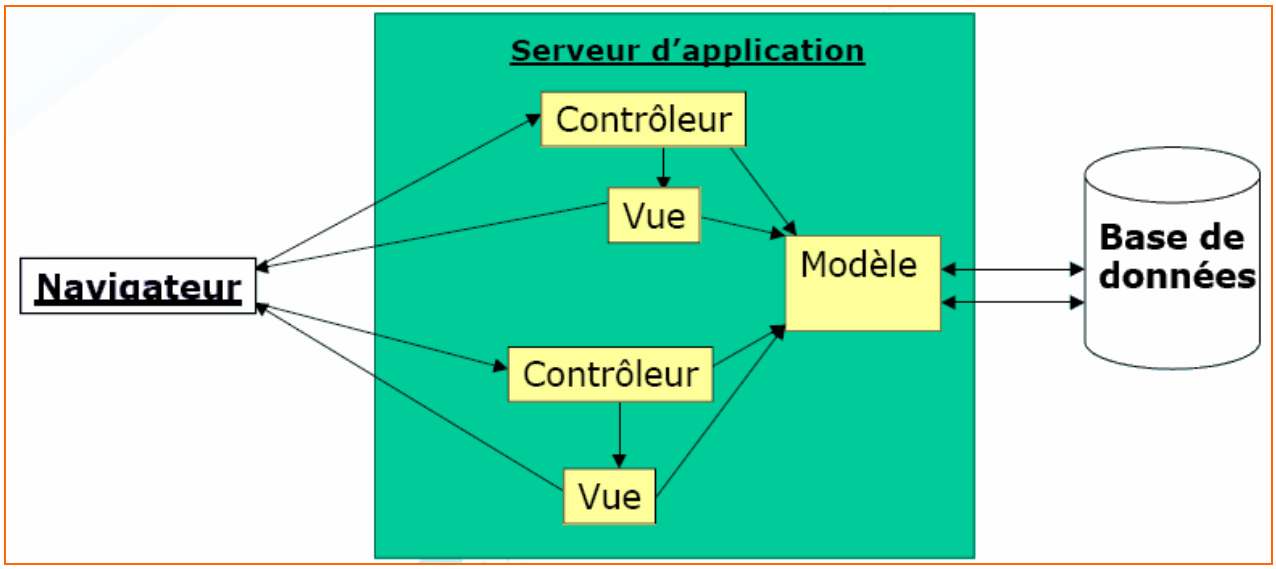
Le schéma ci-dessous présente le cheminement d'une requête lorsque l'utilisateur agit sur l'interface.



1. L'utilisateur manipule l'interface homme/machine. Un événement est envoyé. Cet événement est récupéré par le contrôleur.
2. Le contrôleur effectue l'action demandée par l'utilisateur en appelant les méthodes nécessaires sur le modèle.
3. Le contrôleur informe la vue d'un changement d'état du modèle.
4. La vue interroge le modèle afin de connaître son état.
5. Le modèle donne les renseignements nécessaires à la vue.
6. L'utilisateur voit le résultat de son action.

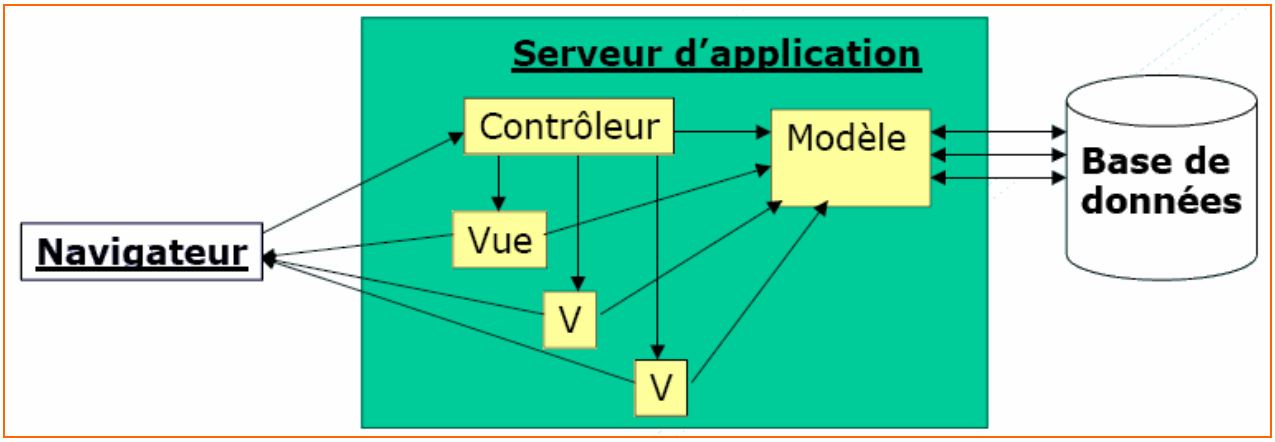
Une 1ère architecture MVC, appelée MVC1, a été définie. Elle implique l'utilisation de plusieurs contrôleurs pour répondre aux événements déclenchés par l'utilisateur.

MODELE MVC 1



Le modèle MVC1 n'est cependant pas encore idéal. Il oblige à écrire une multitude de servlets, une pour chaque point d'entrée dans l'application.
Pour palier cet inconvénient, le modèle MVC2 a été introduit, dans lequel il existe une seule servlet qui fait office de contrôleur. L'architecture est présentée sur le schéma ci-dessous.

MODELE MVC2



L'utilisation d'un seul contrôleur paramétrable permet :

- Une maintenance facilitée
- Une factorisation de traitements comme l'authentification, la sécurité, la gestion des erreurs, …

Dans une architecture JEE, les composants sont utilisés de la façon suivante :

- Les contrôleurs sont des servlets
- Les vues sont des pages JSP
- Le modèle est composé d'EJB ou de Java Bean

IV.1.2 PATTERNS DE LA COUCHE PRESENTATION DANS LES APPLICATIONS WEB

La liste ci-dessous présente les différents patterns qui peuvent être mis en place au niveau de la couche PRESENTATION.

- **Contrôleur de pages (Page Controller)** : Contrôleur de pages est une variante d'implémentation de la partie contrôleur du modèle MVC. Le contrôleur de pages impose un contrôleur par page. Il constitue le point de départ approprié pour le développement de la plupart des applications. Il est recommandé de ne passer au Front Controller que lorsque le besoin s'en fait sentir.
- **Contrôleur frontal (Front Controller)** : Le contrôleur frontal résout le problème de décentralisation qui existe avec le Contrôleur de pages en canalisant toutes les requêtes via un seul contrôleur. Il est donc le seul et unique point d'entrée de l'application.
- **Filtre d'interception (Intercepting Filter)** : ce pattern permet de filtrer les requêtes et les réponses HTTP après et avant traitement. Il décrit une alternative pour implémenter une fonctionnalité récurrente dans une application Web. Le Filtre d'interception fonctionne en transmettant chaque requête dans une chaîne de filtres configurable avant de donner la main au contrôleur. Les filtres traitent généralement les fonctions de bas niveau telles que le décodage, l'octroi d'autorisation, l'authentification et la gestion des sessions alors que le modèle Front Controller gère les fonctionnalités au niveau de l'application. Autre aspect des filtres : ils sont généralement sans état. Lorsqu'un utilisateur obtient une autorisation, par exemple, le serveur Web doit authentifier la session. Si l'utilisateur est authentifié, le traitement se poursuit. Dans le cas contraire, l'utilisateur est redirigé ailleurs. L'un des avantages du

Filtre d'interception est que dans la majorité des implémentations, il est inutile de modifier les pages elles-mêmes pour ajouter une fonctionnalité complémentaire.

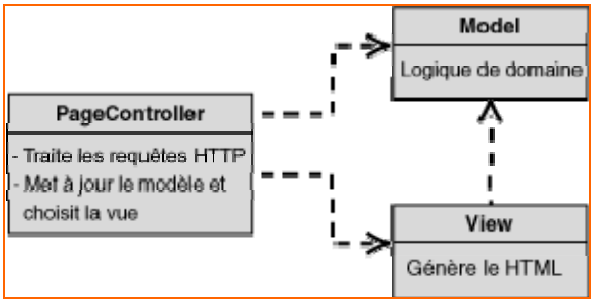
- **View Helper** : Ce pattern permet d'encapsuler les données manipulées par les JSP
- **Composite View** : Ce pattern permet de décomposer l'affichage en fragment de pages
- **Service to Worker et Dispatcher View** : Ces pattern permettent le traitement et la génération de réponses.

Une présentation de ces différents patterns est accessible sur le site de **Core J2EE Patterns** à l'URL suivante :

http://www.corej2eepatterns.com/Patterns2ndEd/index.htm

IV.1.3 EXEMPLE : LE CONTRÔLEUR DE PAGES

Le schéma ci-dessous présente la structure du contrôleur de pages.



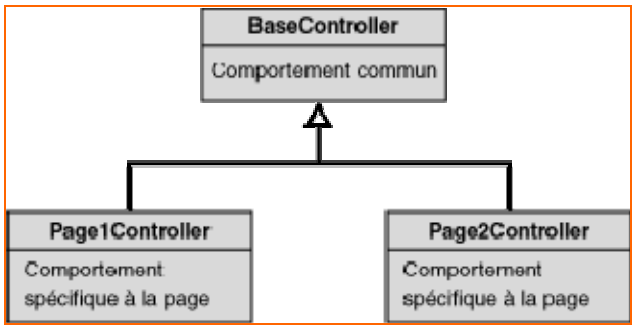
Le contrôleur de page est en charge de :

- Recevoir une demande de page
- Extraire toutes les données pertinentes
- Appeler les mises à jour du modèle
- Transférer la requête à la vue.

La vue dépend donc du modèle pour l'extraction des données à afficher. Le fait de définir un contrôleur de page distinct isole le modèle des spécificités de la requête Web, comme la gestion de la session, ou l'utilisation de chaînes de requête ou de champs masqués de formulaire pour transmettre les paramètres à la page. En simplifiant, nous pouvons dire que vous créez un contrôleur pour chaque lien de l'application Web. Les contrôleurs restent ainsi simples car vous n'avez à vous occuper que d'une action à la fois.

La création d'un contrôleur distinct pour chaque page Web (ou action) peut conduire à une duplication importante du code. Pour cette raison, vous devez créer un contrôleur de base (BaseController) qui intègre les fonctions communes, telles que la validation des paramètres. Chaque contrôleur individuel de page peut hériter de cette fonctionnalité courante issue du contrôleur de base. Outre l'héritage de cette classe commune de base, vous pouvez également définir un ensemble de classes d'assistants que les contrôleurs peuvent appeler pour effectuer des fonctions courantes.

Le schéma ci-dessous présente l'utilisation d'un contrôleur de base pour éliminer la duplication de code.



V FRAMEWORKS DE LA COUCHE PRESENTATION

Avec les technologies Servlets et JSP, la création d'interfaces graphiques ergonomiques peut être complexe. Pour résoudre ce problème, il existe des frameworks qui permettent de simplifier certains processus comme la création de composants tels que des tableaux ou formulaires ou encore la transmission des données à la couche métier.

Parmi les différents frameworks, nous pouvons citer :

- **Apache Struts** : Apache Struts est un framework open source utilisé pour faciliter le développement des applications web JEE. Son but premier est de permettre la mise en place d'une architecture MVC plus aisément. Il utilise pour cela l'API des servlets en les étendant et en donnant accès à des objets qui améliorent l'approche de ces dernières. Cela débouche sur une meilleure subdivision et structuration du code d'une application web. Cette structuration permet ainsi une meilleure maintenabilité et modularité pour des développements futurs.
- **JSF** : JSF est un framework pour les applications web respectant l'architecture JEE. Le but premier de JSF est de procurer un environnement de développement permettant de construire une interface de type web sans devoir toucher aux codes HTML et JavaScript. Ceci est réalisé par la mise en place d'un mapping entre l'HTML et les objets concernés (Managed Bean).
- **Apache Tapestry** : Apache Tapestry est un framework JEE qui permet la création d'applications Web dynamiques à la manière d'Apache Struts ou JSF. Il se démarque de ses concurrents par une approche totalement différente. Pour Apache Tapestry, la création de pages HTML se fait en assemblant des composants. Il se repose donc sur un concept simple et relativement proche du développement d'interface graphique en Java avec Swing.

Oeuvre collective de l'AFPA

Etablissement référent
AFPA Toulouse Palays

Equipe de conception
Pascal DANGU

Date de mise à jour 2009
afpa © Date de dépôt légal mois année

afpa / Direction de l'Ingénierie13 place du Générale de Gaulle / 93108 Montreuil Cedex
association nationale pour la formation professionnelle des adultes
Ministère des Affaires sociales du Travail et de la Solidarité