

MVC2 avec Struts

Version 1.0

Préparé par

F. BERNA

MVC2 avec Struts

INTRODUCTION.....	3
DECOMPOSITION DU MODELE-VUE-CONTROLEUR.....	3
LES TROIS COMPOSANTS MVC D'UNE APPLICATION WEB	4
LES AVANTAGES D'UNE TELLE DECOMPOSITION.....	5
TRAITEMENT D'UNE REQUETE PAR LE MODELE MVC	5
LE MODELE 1.....	3
LE MODELE 2.....	4
STRUTS.....	7
INTRODUCTION	7
ARCHITECTURE STRUTS MVC2.....	8
CONSTRUCTION DES COMPOSANTS DE MODELE	8
<i>Présentation</i>	8
<i>Portée des JavaBeans</i>	9
<i>Beans de Formulaire (ActionForm)</i>	9
<i>Les beans Action</i>	10
CONSTRUCTION DES COMPOSANTS DE VUE	11
<i>Présentation</i>	11
<i>Gestion des messages</i>	11
<i>Interaction Formulaires/FormBean</i>	12
<i>Construction de formulaires avec Struts</i>	12
<i>Types de champs supportés</i>	14
<i>Autres tags utiles</i>	14
<i>Validation automatique de formulaires</i>	14
CONSTRUIRE LE COMPOSANT DE CONTROLE	15
<i>Introduction</i>	15
<i>Fichier struts-config.xml</i>	15
<i>Déploiement de l'Application Web</i>	16
ASPECTS TECHNIQUES IMPORTANTS	19
GESTION DES VARIABLES	19
CONCLUSION.....	19

MVC2 avec Struts

Introduction

Struts est un Framework de développement d'application web, basé sur le pattern de conception Modèle-Vue-Contrôleur (MVC). Il est composé d'un contrôleur (un servlet), de la vue (des pages JSP), et de la logique métier (le modèle).

Struts fait partie du groupe ouvert « Jakarta Project » dont le but est de fournir des outils orientés Application Web, basés sur la technologie Java .

Décomposition du Modèle-Vue-Contrôleur

Le Modèle 1

Le premier modèle consiste en une collection de pages JSP.

La programmation d'un site de type Modèle 1 est facile. Il est notamment possible de mixer des routines en script. Ainsi, la logique métier (javaBeans), le traitement côté serveur (JSP et Java), et les scripts clients (HTML, JavaScript) sont mélangés dans une seule page exécutable. Alors le conteneur JSP fait tout le travail. Très facile à mettre en oeuvre, le modèle 1 ne permet pas la construction de grosses applications web, la logique étant mélangée dans toutes les pages JSP.

Les tâches complexes sont difficilement implantées dans ce modèle qui tend à produire du code difficile à étendre et à maintenir.

Les grosses pages JSP sont difficilement réutilisables.

Le code généré n'est pas modifiable. Une mise à jour met en branle l'ensemble du projet plutôt qu'un seul point précis.

Autant le modèle 1 est très appréciable par sa simplicité d'application, autant la gestion d'un grand nombre de pages JSP est laborieux. Dans le cas de l'implantation d'une grosse application web que l'on veut plus facile à maintenir, on utilisera plutôt une conception de type modèle 2 (basé sur le paradigme MVC).

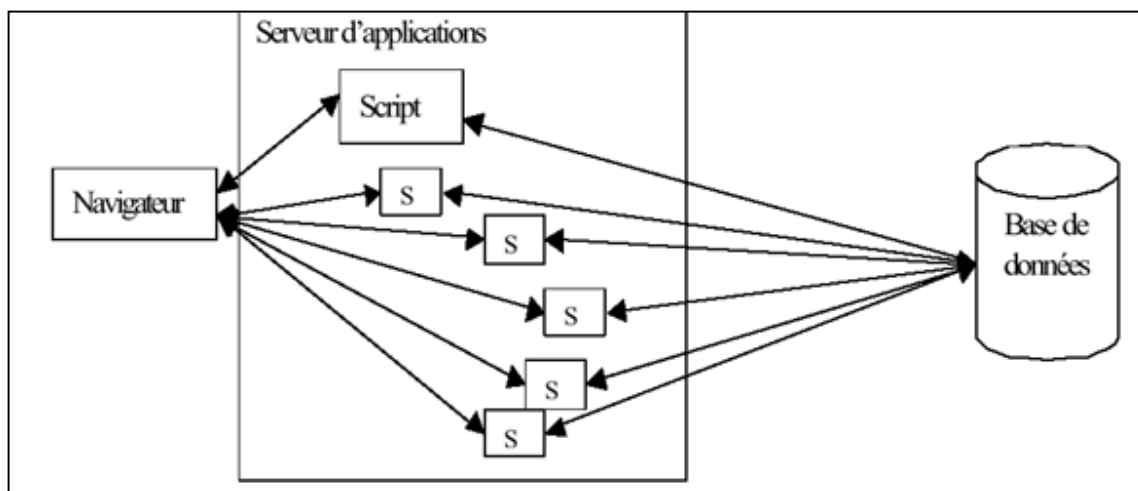


Figure 1. Architecture de type Modèle 1. Les composants sont indépendants

MVC2 avec Struts

Le Modèle 2

La principale caractéristique du Modèle 2 est d'être composé d'un seul servlet de contrôle à l'inverse du modèle 1 qui associe un servlet à chaque vue.

Le modèle 2 est un framework de développement basé sur le pattern de conception **modèle-vue-contrôleur MVC**.

Le "modèle" contient la logique métier et l'état du système.

La "vue" est générée par les pages JSP.

Le "contrôleur", implanté dans un servlet, apporte une gestion centralisée du traitement.

Le pattern MVC rend le code plus propre et sépare la logique métier, le traitement côté serveur et l'affichage dans des composants distincts. Chaque composant est ainsi réutilisable et remplaçable. Par contre, un framework complexe est nécessaire pour assembler ces différents composants.

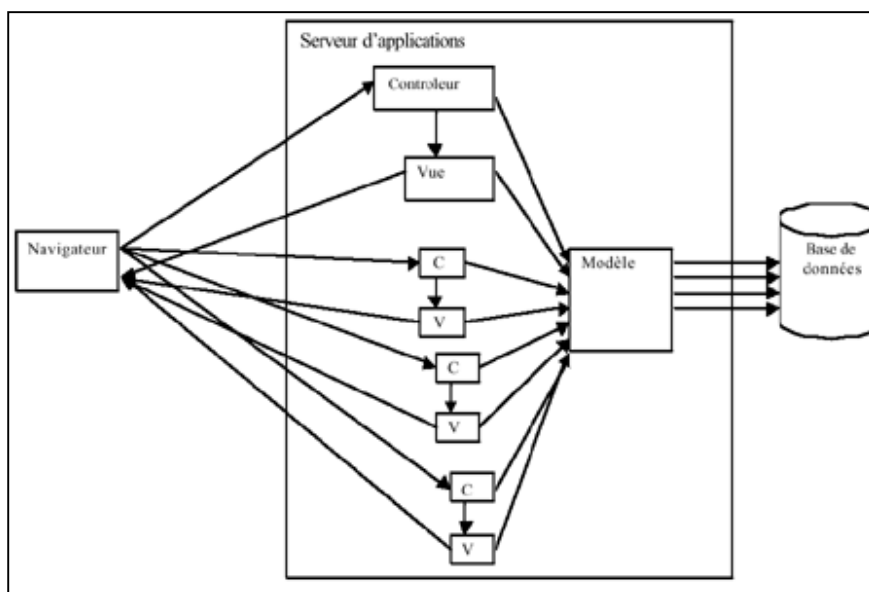


Figure 2. Architecture de type MVC

Les trois composants MVC d'une application web

L'architecture MVC comprend trois classes d'objets :

- **Le modèle** : ce document contient les données et la logique métier
- **La vue** : il s'agit du composant présentation, ou IHM. Plusieurs vues peuvent fournir différentes présentations d'un modèle unique
- **Le contrôleur** : Il s'agit du composant qui répond à la saisie utilisateur. Le contrôleur traduit les événements de l'IHM en modifications du modèle, puis définit la manière dont l'IHM réagit à ces événements

Ces trois classes sont des divisions logiques de fonctionnalités utilisées pour séparer la logique métier, la logique de présentation et la conception du site web.

Les avantages d'une telle décomposition

Cette architecture rend les trois parties de l'application : la logique de présentation, la logique métier et la structure du site web parfaitement indépendantes. Ainsi, la programmation de chacune peut être distribuée à des équipes différentes.

La maintenance de l'application est plus souple. En effet, le partitionnement de l'application permet de modifier la présentation, les pages web, sans toucher à la structure du site et à la logique métier. De même, les composants implémentant la logique métier peuvent être remplacés et la structure du site modifiée.

Traitement d'une requête par le modèle MVC

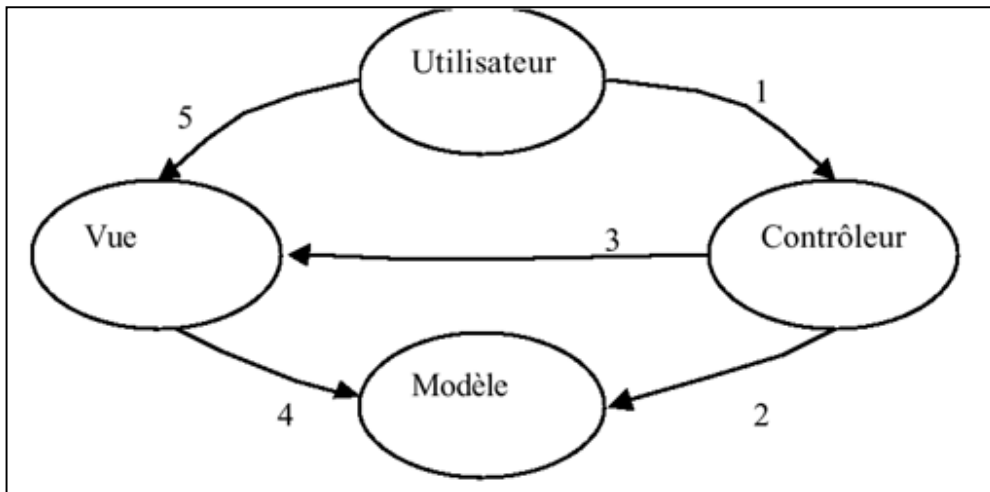


Figure 3. Cheminement d'une requête

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. L'utilisateur voit le résultat de son action

MVC2 avec Struts

MVC2

La construction d'un système MVC2 est complexe. Toutes les pages et les composants doivent être inscrits dans le framework, ce qui produit un surcoût de travail.

Chaque page HTML ne peut plus être conçue indépendamment des autres. Elle fait partie d'un processus plus large, contenant :

- **le modèle** : deux types de JavaBeans pour la logique métier (Action) et pour l'état de l'application (FormBean)
- **la vue** : des pages JSP qui fournissent au client la réponse à ses requêtes
- **le contrôleur** : une servlet qui gère les transactions. Il exécute le code nécessaire à une requête, et en coordonne la réponse

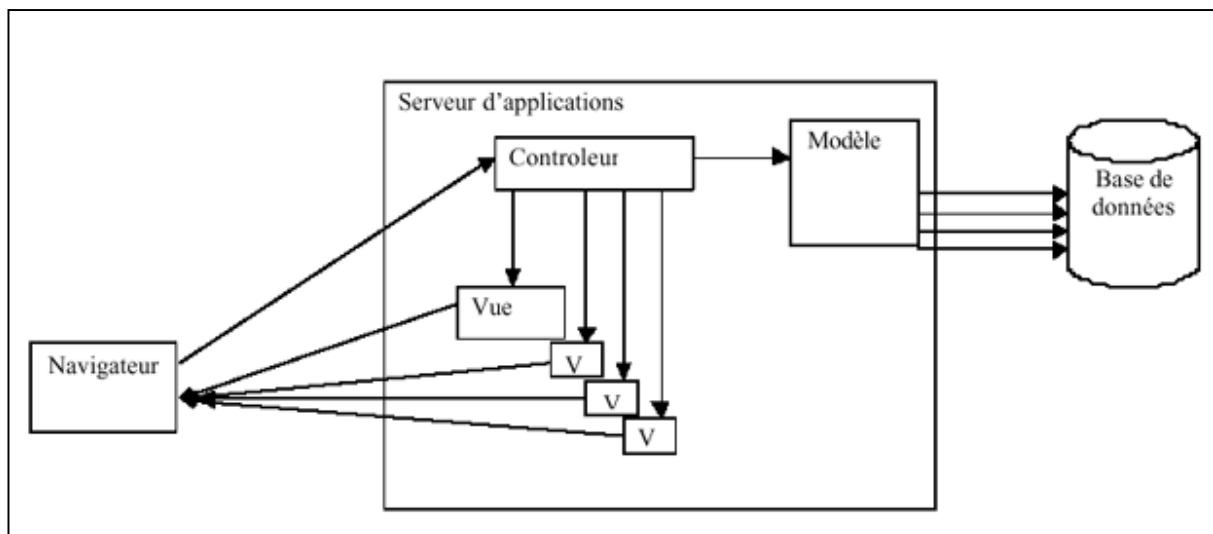


Figure 4. Architecture de type MVC2. Le navigateur agit sur un seul composant, le contrôleur

Struts est un framework de type modèle 2. Il permet de simplifier la construction d'un système MVC qui a l'avantage de fournir du code propre et réutilisable. **Le temps passé à appréhender cette conception est rattrapé dans les phases de maintenance et de mise à jour.**

Struts

Introduction

Struts est un framework en open source, utilisé pour développer des applications web basées sur les technologies Servlet et JSP, et sur une architecture de type MVC2.

Il permet notamment de séparer la partie Modèle (programmation, traitement des informations) de la partie Présentation (affichage)

Il est composé de :

- Un contrôleur : un servlet
- La vue : des pages JSP, avec des bibliothèques de tag struts pour assister le développeur
- La logique métier : le modèle.

Pour expliquer le fonctionnement de Struts, nous utiliserons un exemple très simple :

- Un utilisateur saisit son nom et son prénom dans un formulaire d'une page JSP.
- S'il n'a pas saisi les deux informations, un message d'erreur est affiché et il est invité à recommencer sa saisie
- Si la saisie est correcte, la classe user est instanciée, des traitements (calcul des initiales) sont effectués.
- Une page JSP affiche les informations saisies ainsi que les initiales de la personne.

Pour ce faire, nous utiliserons la classe user :

```
public class User{
    protected String nom;
    protected String prenom;
    protected String initiales;

    public void setNom(String nom) {this.nom = nom;}
    public String getNom() {return nom;}

    public void setPrenom(String prenom) {this.prenom = prenom ;}
    public String getPrenom () {return prenom;}

    public void setInitiales(String initiales)
        {this.initiales = initiales;}
    public String getInitiales () {return initiales;}
}
```

MVC2 avec Struts

Architecture Struts MVC2



Fonctionnement global de Struts :

1. Le client effectue une requête (via un formulaire par exemple)
2. Le contrôleur, configuré par le fichier struts-config.xml affecte le FormAction avec les données saisies dans le formulaire
3. Le contrôle donne la main à Action
4. Action effectue des traitements sur le FormAction ou sur autre source de données
5. Action rend la main au contrôleur en lui spécifiant la réussite ou non de ses traitements et la page de présentation à appeler
6. Le contrôleur appelle la page de présentation JSP
7. La page JSP consulte les données du FormAction et les formate
8. La page JSP envoie une réponse au client.

Construction des composants de modèle

Présentation

Le composant de modèle se concentre sur la création de classes JavaBeans qui supportent toutes les fonctions requises pour effectuer les traitements (comme la gestion d'une Bases de données, ou l'identification des utilisateurs).

On distinguera sous Struts deux type de JavaBeans :

- Les Beans **FormAction** : Beans de formulaires, destinés à récupérer et à effectuer des traitements de validité sur les données postées par un formulaire.
- Les Beans **Action** : Beans d'action, destinés aussi à effectuer des traitements. Ils informent le contrôleur quelle page JSP appeler en fonction du résultat des traitements.

MVC2 avec Struts

Portée des JavaBeans

Dans une application web, les JavaBeans peuvent avoir différentes portées (définissant la durée de vie et la visibilité des Beans). Dans chaque portée, on trouvera une collection d'attributs.

La spécification JSP définit la portée en utilisant les termes suivants :

- **page** : Beans qui sont visibles par une seule page JSP, pour la durée de vie de la requête
- **request** : Beans qui sont visibles par une seule page JSP, ainsi que par toutes les pages ou servlet inclus dans cette page
- **session** : Beans visibles par toutes les pages JSP et servlets d'une session utilisateur
- **application** : Beans visibles par toutes les pages JSP et servlets d'une application web

Il est important de rappeler que les Servlets et pages JSP d'une application web partagent les mêmes collections d'attributs pour une portée donnée. Par exemple, un Bean stocké comme attribut de requête dans un servlet comme suit :

```
MyCart mycart = new MyCart(...) ;  
Request.setAttribute(« cart », mycart) ;
```

Est immédiatement visible par une page JSP appelée par ce servlet, en utilisant le tag action :

```
<jsp:useBean id='cart' scope='request' class='com.mycompany.MyApp.MyCart' />
```

Beans de Formulaire (ActionForm)

Le Framework Struts gère des beans ActionForm (qui héritent de la classe ActionForm) pour chaque formulaire de l'application. Pour être fonctionnel, chaque Bean devra être défini dans le fichier de configuration struts-config.xml. (voir section 'Composant de contrôle').

Lors de la validation d'un formulaire, le contrôleur Struts crée le ActionForm associé dans la session utilisateur (pour que ses données soient accessibles par les pages JSP), affecte toutes les variables avec les données du formulaire, et passe le bean en paramètre à la méthode Perform() du Bean Action associé.

Principes à respecter pour coder les beans ActionForm :

- Ce sont des classes qui ne requièrent pas de méthodes spécifiques. Ils ne possèdent que des méthodes getter et setter.
- La méthode validate() permet de contrôler la validité des données reçues du formulaire (ex. : age d'une personne) et d'informer le contrôleur si les données saisies sont valides ou non. Cette méthode n'est pas obligatoire, la validation peut être effectuée dans le bean Action associé.
- Chaque propriété de la classe doit avoir ses propres accesseurs de la forme getMapropriété() et setMapropriété(Mapropriété). Les accesseurs permettent aux pages JSP de consulter le contenu des Beans, grâce aux taglibs.

Le Bean ActionForm est donc un Bean qui reprend toutes les données saisies dans un formulaire, effectue un contrôle sur ces données, et sera utilisé par le bean Action associé pour des traitements.

Retour à l'exemple.

Voici la classe UserForm, qui récupère les informations saisies dans le formulaire, et qui vérifie que le nom et le prénom aient bien été saisis. Dans le cas contraire, elle retourne un message d'erreur.

MVC2 avec Struts

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.upload.FormFile;

public class UserForm extends ActionForm {
    protected String nom;
    protected String prenom;

    public void setNom(String nom) {this.nom = nom;}
    public String getNom() {return nom;}

    public void setPrenom(String prenom) {this.prenom = prenom ;}
    public String getPrenom () {return prenom;}

    public ActionErrors validate(ActionMapping mapping,
                                HttpServletRequest request) {

        ActionErrors errors = new ActionErrors();
        if ((nom == null) || (nom.length() < 1))
            errors.add("nom", new ActionError("error.nom.vide"));
        if ((prenom == null) || (prenom.length() < 1))
            errors.add("prenom", new
                ActionError("error.prenom.vide"));

        return errors;
    }
}
```

Note : la classe UserForm doit être déclarée dans le fichier WEB-INF/struts-config.xml pour être exécutable (voir section 'construire le composant de contrôle') ;

Les beans Action

Lors de la réception d'une requête utilisateur, le contrôleur appelle le bean Action associé. Celui-ci va effectuer tous les traitements demandés (Recherche d'informations dans une Base de données, validation d'informations postées). Puis, en fonction du résultat, le bean Action rend la main au contrôleur (par un ActionForward) en lui spécifiant la page JSP de présentation à appeler.

Ces traitements se font dans la méthode perform() du bean Action.

Retour à l'exemple.

Voici l'Action qui récupère le nom et le prénom du Bean FormAction, Calcule les initiales, et rend la main au contrôleur.

MVC2 avec Struts

```
public final class inscriptionAction extends DicoBaseOngletAction {
    public ActionForward perform(ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
        throws IOException, ServletException{
        //Initialisation
        User user = null ;

        //Récupération du nom saisi dans un formulaire associé à la classe
        UserForm (voir section 'contrôleur'
        String usernom = ((UserForm) form).getNom();
        String userprenom = ((UserForm) form).getPrenom() ;

        //Traitements
        user.setNom(usernom) ;
        user.setPrenom(userprenom) ;
        user.setInitiales(usernom.substring(0,1) + userprenom.substring(0,1)) ;

        // on sauve les informations dans la session, pour être
        accessibles par les pages JSP
        HttpSession session = request.getSession();
        session.setAttribute("user", user);

        //On redonne la main au contrôleur qui appelle la page JSP
        associée à la clé "success"
        return(mapping.findForward("success"));
    }
}
```

Note : la classe inscriptionAction doit être déclarée dans le fichier WEB-INF/struts-config.xml pour être exécutable (voir section 'construire le composant de contrôle') ;

Construction des Composants de Vue

Présentation

Ce chapitre porte sur la construction des composants de vue d'une application Struts. Les composants de vue sont basés sur des pages JSP, avec des tags spécifiques à Struts. Les pages JSP utilisent des bibliothèques de tags pour accéder aux données des Beans (taglib bean), pour insérer des pages JSP (taglib template), pour créer des formulaires dont les données sont accessibles par les FormBean (taglib HTML), ou pour effectuer des traitements logiques comme les boucles, les test (taglib logic).

Gestion des messages

Struts permet d'externaliser tous les messages utilisés dans les pages JSP, comme dans les Beans (messages d'erreur par exemple). Ces messages sont stockés dans le fichier MonApplication.properties, et sont indexés sous la forme clé=valeur

MVC2 avec Struts

Retour à l'exemple.

Notre fichier `MonApplication.properties` possède les entrées :

```
error.nom.vide=<LI>Vous devez saisir votre nom
error.prenom.vide=<LI>Vous devez saisir votre prénom
prompt.nom=Nom
prompt.prenom=Prénom
button.submit=Envoyer
button.reset=Effacer
```

Les messages sont accessibles dans les pages JSP par le tag :

```
<bean:message key="clé"/>
```

Notes :

- le fichier `MonApplication.properties` doit être déclaré dans le fichier `WEB-INF/web.xml` (voir section 'construire le composant de contrôle')
- l'installation de ce fichier sur serveur WebSphere doit respecter certaines contraintes (voir document « Ajouter Struts à une application Web sous WebSphere 3.5 »)

Interaction Formulaires/FormBean

Dans un système de développement traditionnel, des contrôles sont effectués sur les données saisies dans les formulaires, directement sur le client (JavaScripts de contrôle). Ainsi, le client n'a pas à ressaisir toutes les autres informations.

Dans le cas où les contrôles sont effectués sur le serveur (le formulaire est posté), il faudra réafficher toutes les informations. Ceci peut s'avérer laborieux. En effet, pour notre exemple, il faudrait entrer la ligne :

```
<input type="text" name="prenom" value="<%= userBean.getUserNom() %>"/>
```

Struts fournit des outils simples d'utilisation pour construire des formulaires qui respectent ces contraintes. Ainsi le code précédent deviendra sous Struts :

```
<html:text property="nom"/>
```

Il n'est pas nécessaire de préciser le Bean associé. Il est retrouvé automatiquement (celui utilisé pour poster le formulaire).

Construction de formulaires avec Struts

Voici un exemple de formulaire qui illustre comment Struts fonctionne avec les formulaires.

Les fonctionnalités Struts :

- Appel des bibliothèques de Tag en début de page
- Utilisation des messages du fichier `MonApplication.properties`. Pour afficher les intitulés « Nom » utilise la clé `prompt.nom` et pour « Prénom » la clé `prompt.prenom`. De même pour les boutons valider et recommencer.
- Le tag `errors` affiche les messages d'erreur en cas de saisie erronée
- Le tag `form` crée un formulaire couplé avec le FormBean inscription. Ce bean est d'abord utilisé pour initialiser la valeur de chacun des champs, puis pour recueillir les informations postées par le formulaire.
- Le tag `text` crée un champ `<input>` de type « text ».

MVC2 avec Struts

- Les tag `submit` et `reset` créent des boutons d'envoi et de remise à zéro.

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<body bgcolor="white">

<!-- la balise suivante récupère les messages d'erreurs générées
par les FormAction et Action -->
<html:errors/>
<html:form action="/inscription" focus="username">
<table border="0" width="100%">
  <tr>
    <th align="right">
      <html:message key="prompt.nom"/>
    </th>
    <td align="left">
      <html:text property="nom" size="16"/>
    </td>
  </tr>
  <tr>
    <th align="right">
      <html:message key="prompt.prenom"/>
    </th>
    <td align="left">
      <html:text property="prenom" size="16"/>
    </td>
  </tr>
  <tr>
    <td align="right">
      <html:submit>
        <bean:message key="button.submit"/>
      </html:submit>
    </td>
    <td align="right">
      <html:reset>
        <bean:message key="button.reset"/>
      </html:reset>
    </td>
  </tr>
</table>
</html:form>
</body>
</html:html>
```

Note : le fichier `formulaire.jsp` doit être déclaré dans le fichier `WEB-INF/struts-config.xml` pour être exécutable (voir section 'construire le composant de contrôle').

MVC2 avec Struts

Types de champs supportés

Struts définit des tags HTML pour tous les champs suivants :

- checkbox
- input de type password
- champs de type hidden
- boutons radio
- boutons reset
- boutons submit
- listes select
- options
- champs input de type text
- textareas

Dans tous les cas, chacun de ces champs doit être inclus dans un tag `form`. Ceci pour connaître le bean associé.

Autres tags utiles

Struts définit aussi une liste de tags autres que ceux réservés aux formulaires :

- `<logic :iterate>` pour répéter une boucle pour tous les éléments d'une collection (ex. : Vecteur, Hashtable, tableaux)
- `<html :link>` pour générer une ancre ou un lien hypertexte (dont l'URL peut être directement modifiée à partir des informations contenues dans un Bean)
- `<html :img>` pour insérer une image (dont l'URL peut être directement modifiée à partir des informations contenues dans un Bean)
- `<bean :parameter>` pour récupérer les paramètres envoyés à cette page.

Validation automatique de formulaires

En plus des interactions Formulaire/FormBean, Struts offre une méthode simple de validation des informations saisies : la méthode `validate()` dans le FormBean.

Cette méthode est appelée après que le formulaire ait été posté, et avant que la méthode `perform()` du FormBean soit appelée.

On y effectue les divers contrôles de saisie (données obligatoires...). En cas de saisie erronée, un message d'erreur doit être ajouté dans le conteneur `ActionErrors`.

Une fois la méthode `validate()` exécutée, si le conteneur de messages d'erreur est vide, la méthode `perform()` est appelée. Dans le cas contraire, le formulaire d'appel est réaffiché, et le tag `<html :errors>` s'occupe d'afficher les différents messages d'erreur.

Comme dit dans le chapitre « FormBeans », cette méthode est optionnelle et doit être configurée dans le fichier `struts-config.xml` pour être exécutée.

Construire le composant de contrôle

Introduction

Struts inclut un servlet de contrôle qui associe chaque requête à une Action, un FormAction, ou une redirection.

Pour fonctionner, ce servlet doit être déclaré dans le fichier web.xml, et configuré à l'aide du fichier struts-config.xml.

Fichier struts-config.xml

Le fichier struts-config.xml permet de configurer les requêtes, les actions, les formulaires, et les renvois vers les pages JSP.

A l'intérieur de la balise `<struts-config>`, on trouve essentiellement trois éléments :

- `form-beans` : contient la définition des FormAction. Il est nécessaire d'utiliser un élément `form-bean` par ActionForm, en précisant l'URL de la requête (`name`) et la classe utilisée pour cet ActionForm (`type`)
- `global-forwards` : contient les déclarations de renvois d'URL vers des pages JSP ou des Action.
- `action-mapping` : contient la définition des Actions.
 - Pour chaque Action, il faut définir :
 - le chemin de la requête associée (`path`)
 - la classe à utiliser (`type`)
 - si nécessaire, le nom du FormAction à utiliser (`name`)
 - la portée de la classe (`scope`)
 - si oui ou non la méthode `validate()` du FormBean doit être appelée (`validate = true` ou `false`)

MVC2 avec Struts

Exemple d'un fichier struts-config.xml

```
<struts-config>
  <!-- beans de formulaires -->
  <form-beans>
    <form-bean name="userForm"
               type="dicoepc.admin.userForm"
    />
  </form-beans>

  <!-- redirections globales -->
  <global-forwards type="org.apache.struts.action.ActionForward" />
  <forward name="index" path="/index.html" redirect="false" />
</global-forwards>

  <!--mapping des actions -->
  <action-mappings>
    <!-- Accueil -->
    <action
      path="/inscription"
      type="dicoepc.admin.inscriptionAction"
      scope="request"
      name="userForm"
      validate="true"
      unknown="true">
      <forward name="success" path="/jsp/index.jsp" redirect="false" />
      <forward name="failure" path="/jsp/erreurs.jsp" redirect="false"/>
    </action>
  </action-mappings>
</struts-config>
```

Déploiement de l'Application Web

La dernière étape est la configuration du descripteur de déploiement de l'application Struts, pour intégrer tous les composants nécessaires.

Sous Tomcat, ce descripteur est le fichier WEB-INF/web.xml.

MVC2 avec Struts

Configuration du Servlet d'Action

La déclaration du servlet de contrôle Struts se fait comme l'exemple suivant :

```
<!-- Configuration du servlet de contrôle Struts -->
<!-- Nom du servlet et classe associée -->
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>

  <!-- fichier contenant les messages utilisés dans les pages JSP et les beans -->
  <init-param>
    <param-name>application</param-name>
    <param-value>dicoepc.ApplicationResources</param-value>
  </init-param>

  <load-on-startup>2</load-on-startup>
</servlet>
```

Configuration du mapping du Servlet d'Action

Le servlet d'action Struts doit être configuré pour ne s'appliquer que sur certaines URLs. Ainsi, on peut décider que le contrôleur sera appelé pour toutes les URLs d'extension .do :

```
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Configuration des bibliothèques de Tag de Struts

Ensuite, il faut définir les bibliothèques de tag de Struts. Il en existe actuellement quatre fournies avec Struts :

- La bibliothèque struts-bean contient les tags utilisés pour accéder aux Beans et leur propriétés, aussi bien que pour créer des Beans.
- La bibliothèque struts-html contient les tags utilisés pour créer des formulaires Struts
- La bibliothèque struts-logic contient les tags utilisés pour gérer les sorties texte, gérer les flots de l'application, les tests, les boucles itératives.
- La bibliothèque struts-template contient les tags utilisés pour importer une page JSP à l'intérieur d'une autre

MVC2 avec Struts

```
<taglib>
  <taglib-uri>
    /WEB-INF/struts-bean.tld
  </taglib-uri>
  <taglib-location>
    /WEB-INF/struts-bean.tld
  </taglib-location>
</taglib>
<taglib>
  <taglib-uri>
    /WEB-INF/struts-html.tld
  </taglib-uri>
  <taglib-location>
    /WEB-INF/struts-html.tld
  </taglib-location>
</taglib>
<taglib>
  <taglib-uri>
    /WEB-INF/struts-logic.tld
  </taglib-uri>
  <taglib-location>
    /WEB-INF/struts-logic.tld
  </taglib-location>
</taglib>
<taglib>
  <taglib-uri>
    /WEB-INF/struts-template.tld
  </taglib-uri>
  <taglib-location>
    /WEB-INF/struts-template.tld
  </taglib-location>
</taglib>
```

Ajouter des composants Struts à votre application

Une fois les deux fichiers créés, il faut intégrer tous les composants Struts à l'application.

- Télécharger le fichier d'installation de Struts sur le site :
<http://jakarta.apache.org/struts>
- Copier les fichiers commons-*.jar dans le répertoire WEB-INF/lib de l'application (fichiers communs à tous les programmes du projet Jakarta (contient notamment un parseurs XML, des applicatif Java)
- Copier struts.jar dans le répertoire WEB-INF/lib de l'application (contient les classes nécessaires au fonctionnement de Struts, comme les classes Action, FormAction et autres)
- Copier les fichiers struts-*.tld dans le répertoire WEB-INF (contient les librairies de tag pour les pages JSP propres à Struts)
- Installer les classes de l'application dans le répertoire WEB-INF/classes

Aspects techniques Importants

Gestion des variables

Pour une application Struts donnée, chaque classe n'est instanciée qu'une seule fois. Tous les utilisateurs utilisent la même instance.

Toutes les classes Action de l'application ne doivent utiliser que des variables locales.

Struts s'occupe d'empiler ces variables pour éviter tout conflit. En revanche, toute variable d'instance est partagée entre tous les utilisateurs.

Echange de données

Toutes les données calculée dans les Beans, ou saisies dans les formulaires transitent par la session utilisateur. Ainsi, tout résultat calculé dans une Action devra être monté en session pour être accessible par les pages JSP. Inversement, toute donnée saisie dans un formulaire est accessible par les Beans dans la session.

Conclusion

Struts est un Framework qui permet de développer des applications web en utilisant le modèle MVC2. Il apporte de nombreux avantages : séparation des fonctions de modèle, de contrôle et de vue. Il permet ainsi de bien distinguer la partie programmation métier de la partie affichage. Struts fournit les outils nécessaires pour que la conception de la vue ne fasse pas intervenir de compétences en programmation, et pour que la partie modèle ne fasse pas intervenir de compétences en infographie et design.

La prise en main de Struts est difficile : il faut se familiariser avec le concept MVC2 et apprendre à utiliser les API et les tags Struts ; l'installation du Struts peut aussi s'avérer laborieuse (notamment sur le serveur WebSphere 3.5). Cependant, les gains de temps sont importants par la suite : facilité de maintenance (que ce soit du côté modèle ou du côté vue), portions de code facilement réutilisables.

On réservera donc l'utilisation de Struts pour les grosses applications gérant un grand nombre de pages.