TUTORIAL

Struts

struts_antispam_nuxora.com

The Jakarta Project
http://jakarta.apache.org

Introduction

Struts est un framework proposant un cadre logiciel pour l'organisation des échanges avec l'utilisateur et la dynamique de l'application. Il promeut une structuration de la couche Servlet-JSP suivant le modèle MVC II. La définition de la dynamique de l'application est partiellement déclarative, via un fichier de configuration, partiellement implémentée, au sein des classes Action.

.I. Présentation et modèles équivalents

Struts est un projet Open Source développé par la communauté Jakarta d'Apache. Il a débuté en mai 2000 sous la direction de Craig R Mc Clanahan, qui participe également au développement de Tomcat. Aujourd'hui, Struts est géré par plusieurs committers. Sa mailing-list comporte un millier de personnes. C'est un projet très actif.

Expliquons tout d'abord ce qu'est un **framework** (lit. Structure) : architecture et ensemble d'outils de développement. Nous expliquerons plus en détail ce qu'est un framework dans le .II.

Struts fournit un **framework MVC** comprenant les composants suivants :

· Un *contrôleur* facilement configurable permettant d'associer des actions (méthode d'un objet Java) à des requêtes HTTP.

· Des *librairies de* tags spécifiques pour créer facilement une vue.

· Un *Digester*, permettant de parser un fichier XML et d'en récupérer seulement les informations voulues.

· Des utilitaires permettant de remplir automatiquement des champs et de créer des applications supportant plusieurs langages.

Le modèle

Le modèle d'une application Struts est complètement standard : suivant le paradigme MVC, Struts et le modèle sont indépendants. Le modèle peut très bien accéder directement à une base donnée relationnelle, XML ou utiliser des EJB. Il faudra juste veiller à permettre son initialisation.

La vue

La vue sous Struts est constituée de pages JSP. Afin de rendre la création de ces pages aisées par un designer et d'y éviter l'introduction de code Java, Struts propose l'utilisation de 4 librairies de tags ou taglib spécifiques : html, bean, logic et template.

Taglibs de Struts

Les taglibs de Struts sont très complètes et permettent de faire beaucoup de chose sans écrire de code Java. Des librairies répondant à des besoins plus spécifiques - création de composants, accès à une base de données, mises en page - sont en projet pour les prochaines versions de Struts.

Le contrôleur

Le contrôleur est la partie du framework qui fait le lien entre la vue et le modèle. C'est elle qui permet de gérer l'enchaînement de la navigation. Dans Struts, le point central du contrôleur est un servlet de la classe ActionServlet. Toutes les requêtes y aboutissent.

Du point de vue du développeur, la programmation du contrôleur passe par :

· L'écriture d'un fichier de configuration : struts-config.xml. Ce fichier décrit en particulier quelles sont les actions possibles et à quelles classes Actions il faut les associer (mapping).

· L'écriture des objets formulaires qui vont servir à la transmission des données de la vue au modèle. Ces objets étendent ActionForm. Ils contiennent toutes les propriétés des formulaires, ainsi qu'une méthode permettant de valider ou non les valeurs de ces propriétés. Lorsqu'un formulaire HTML est renvoyé, Struts instancie automatiquement l'ActionForm correspondant et initialise ses propriétés avant de les valider.

· L'écriture des Action à effectuer pour chaque requête.

Une fois que Struts a validé les données envoyées, il appelle la méthode perform () de l'Action correspondante. Dans cette méthode, le développeur peut récupérer les données rentrées dans le formulaire via l'ActionForm, et effectuer les opérations nécessaires sur le modèle. Il peut pour cela s'aider des utilitaires de Struts permettant de recopier des propriétés d'un bean à un autre. Struts fournissant le servlet cœur du contrôleur et des classes Action et ActionForm qu'il n'y a plus qu'à étendre, la création d'actions est très rapide.

Entités Struts

Struts s'appuie sur le modèle de conception des commandes : un discriminant, une chaîne de caractères en général, fait office de commande et indique le traitement à exécuter. Une table de correspondance permet de faire le lien entre commande et traitement à exécuter. Dans Struts, la commande est déterminée par une partie de l'URL transmise au servlet contrôleur, ActionServlet. Les traitements sont encapsulés dans les classes Actions. La correspondance (le "mapping") entre la commande et le l'Action à exécuter est définie par les classes ActionMapping. Le contenu des classes ActionsMapping est déclaré au sein du fichier de configuration Struts et consiste à la base en un couple morceau d'URL/nom de la classe Action. Struts intègre un mécanisme d'automatisation des relations entre Actions et JSP par le biais des ActionForms. Ces classes sont des JavaBeans renseignés automatiquement à partir des paramètres des requêtes transmises au servlet contrôleur et transmises telles quelles aux Actions.

Le framework Struts propose plusieurs taglibs, qui, combinées à des JavaBeans dédiés, les ActionForms, fournissent un support semi-automatisé à la manipulation de formulaires HTML (description, remplissage, validation côté client et serveur, remontée de message d'erreur, …). Struts fournit par ailleurs quelques services techniques, tels qu'un support à l'externalisation de chaînes de caractères (messages) et à l'internationalisation des ces chaînes, une gestion de sources de données (DataSources), une interface de journalisation, ainsi que quelques Actions spécialisées.

Taglibs

La manipulation des beans ActionForms au sein des JSP est grandement facilitée par l'emploi des taglibs Struts essentiellement dédiées à cet usage.

Struts fournit un peu plus d'une cinquantaine de tags personnalisés regroupés au sein de 4 librairies (taglibs) :

La librairie des tags bean pour la manipulation pure de beans

La librairie des tags HTML pour la manipulation des formulaires et éléments de formulaires HTML,

La librairie des tags logiques pour la mise en place de traitements conditionnels et/ou itératifs

La libraire des tags template qui propose un mécanisme de gestion de modèles de JSP (les templates).

Framework	Homepage
Action Servlet	http://dione.zcu.cz/~toman40/ActionServlet/
Barracuda	**http://barracuda.enhydra.org/**
Bento	http://www.bentodev.org/
Bishop	http://bishop.sourceforge.net/
Cocoon	**http://xml.apache.org/cocoon/**
Echo	http://www.nextapp.com/products/echo/
Expresso	http://www.jcorporate.com/
Hammock	***n'existe plus***
JADE	http://sourceforge.net/projects/salmon
Japple	http://www.japple.org/
JPublish	http://www.jpublish.org/

Framework	Homepage
JStateMachine	http://sourceforge.net/projects/jstatemachine/
Maverick	http://mav.sourceforge.net/
Melati	http://www.melati.org/
Niggle	http://niggle.sourceforge.net/
Open Symphony	**http://www.opensymphony.com/**
Struts	**http://jakarta.apache.org/struts/**
Tapestry	**http://tapestry.sourceforge.net/**
TeaServlet	http://opensource.go.com/TeaServlet/
Turbine	http://jakarta.apache.org/turbine/
Webwork	http://sourceforge.net/projects/webwork/
wingS	http://wings.mercatis.de/

Les frameworks MVC2 les plus avancés

Il existe actuellement un certain nombre de projet plus ou moins avancé visant à la création de framework MVC2. Ces projets tous basés sur les servlets progressent très vite. En voici une rapide présentation :

Barracuda

Ce framework qui se veut très complet est développé par la communauté Open Source Enhydra. Il est prévu qu'il comporte :

· Un modèle événementiel complet,

· Un modèle et une librairie de composants MVC pour l'interface utilisateur

· Un système de validation des formulaires et de mappings avec des objets Java.

· Un contrôleur MVC2

· Une librairie Javascript.

Hammock - *N'existe plus*

Ce framework permet de développer une application web de la même façon qu'une application Swing. Pour cela, Hammock utilise le même modèle : système d'événements identique, présence de composants IHM 'Panel', et de layout 'FlowLayout ' ou 'TableLayout' etc. Ce système fonctionne sans JSP. Signalons qu'il n'est pas Open Source mais développé par OOP.

Tapestry

Ce framework est axé sur les composants : la création d'une page se fait par le développement de composants qui vont définir cette page. Il propose des concepts intéressants comme la séparation de l'état de la page et de la page elle-même afin de disposer d'un pool de pages ou encore des possibilités d'internationalisation via des templates. Tapestry facilite la gestion des erreurs et de la charge.

Tapestry a d'abord été développé par Primix et est maintenant disponible en Open source.

Webwork - *N'existe plus*

Ce framework utilise le concept du Pull Hierarchical Model View Controller. Il est similaire techniquement au framework Struts dont la présentation suit, mais son API est plus réduite. Il pourrait être utilisé avec d'autres technologies que les servlets.

Nom	Version	url	Puissance	Complexité
Barracuda	1.1	http://barracuda.enhydra.org/	+++	+++
Hammock	1.0	n'existe plus	+++	+++
Tapestry	2.1	http://sourceforge.net/projects/tapestry	++	+++
Webwork	0.95	délaissé au profit de Tapestry	+	+
Struts	1.0.2	http://jakarta.apache.org/struts	++	++

Pourquoi choisir Struts comme framework MVC2

Struts fait parti du projet Apache. Or les projets Apache ont tendance à monopoliser toute l'attention des utilisateurs. Sachant que la qualité des projets Open Source est apportés par le nombre d'utilisateurs, on peut penser que Struts sera de meilleure qualité que les autres projets MVC2. Conclusion tirée il y a 18 mois. Depuis certains frameworks ont été abandonnés, alors que Struts continue son développement de manière certaines.

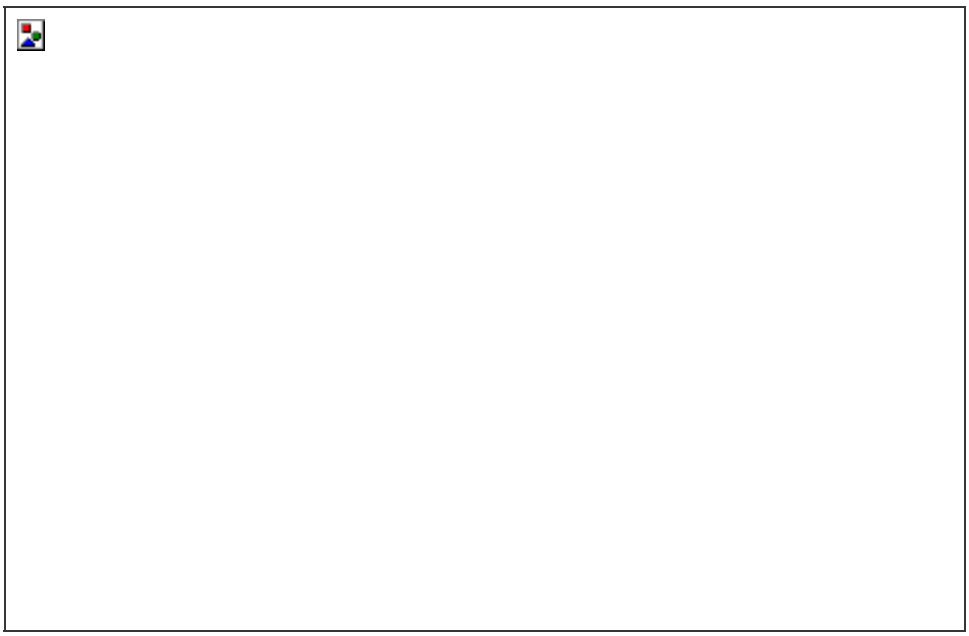
.II. Description plus technique

Struts est un Framework MVC pour développer des applications web, développé en Open Source dans le cadre du projet Jakarta de l'ASF (Apache Software Foundation).

Les ***framewoks*** sont des structures logicielles qui définissent des cadres dans lesquels viennent s'insérer les objets et concepts spécifiques à une application. En pratique, un framework (traduction littérale: squelette d'application) est un ensemble de classes et de mécanismes associés à une architecture logicielle qui fournissent un ou plusieurs services techniques ou métiers aux applications qui s'appuient dessus. Alors qu'un framework métier fournit des services à forte plus value fonctionnelle (gestion de clients, d'abonnements, de news, ...) un framework technique apporte les concepts, entités et mécanismes qui permettent, dans le cadre d'une architecture logicielle retenue, de s'abstraire d'un certain nombre de problématiques conceptuelles et techniques récurrentes. J2EE est un Framework Java pour les développements web. Les Frameworks ont l'avantage de structurer, simplifier, segmenter les développement et donc les accélérer. Struts prend en compte les problématiques de performance, sécurité, multi-langue...

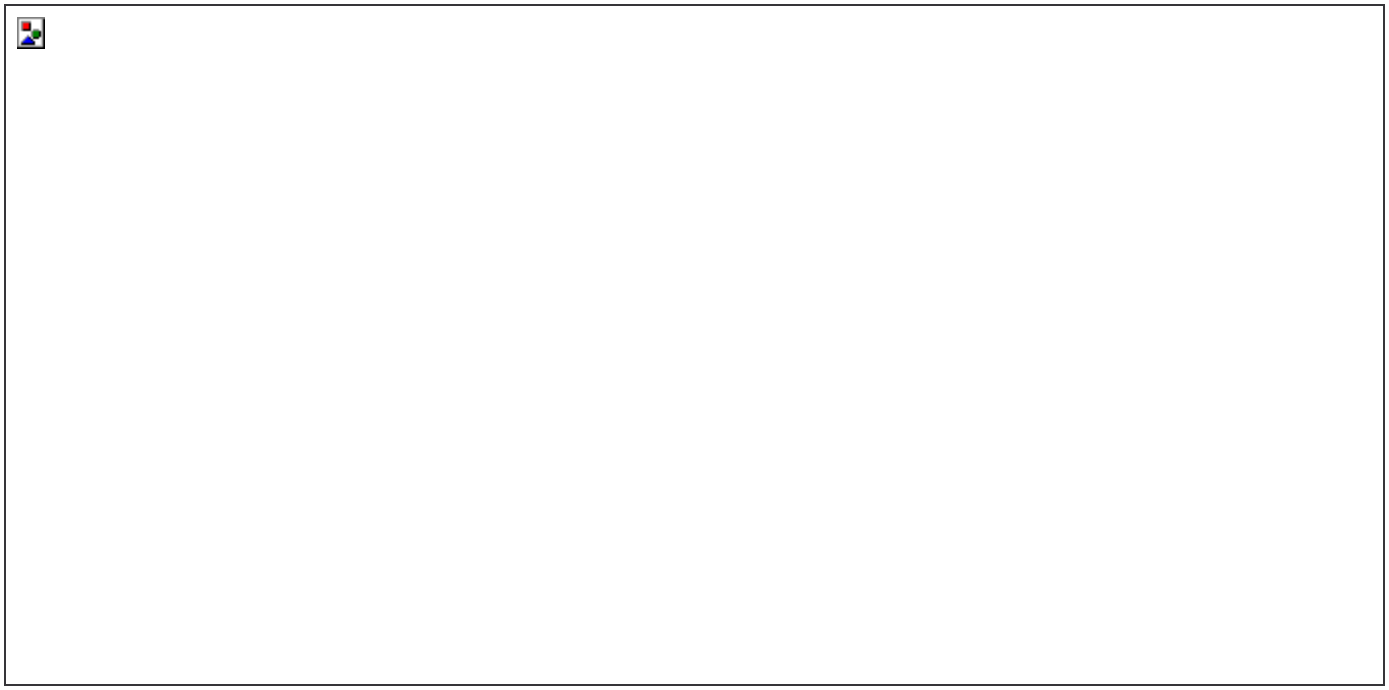
Le ***MVC*** : acronyme synonyme de séparation des composants de type "Model" (logique métier et accès aux données), "View" (présentation des informations) et "Controller" (la gestion des interactions des utilisateurs).

Le modèle MVC s'appuie essentiellement sur la séparation en 2 couches verticales regroupant d'un coté les objets métiers (Modèle) et de l'autre les objets IHM, ces derniers étant eux mêmes regroupés en objets chargés de l'acquisition d'informations, en provenance de l'utilisateur (contrôleur) et en objets chargés de la restitution d'informations vers l'utilisateur (Vue).



Le modèle MCV II hérite des propriétés du modèle MVC. Son organisation cependant capitalise sur la longue expérience acquise avec MVC et s'adapte au contexte des applications Internet et de la plate forme J2EE.

Dans le modèle MVC II, le servlet est unique, en classe et en instance. Il garantit l'unicité du point d'entrée de l'application. Il prend en charge une partie du contrôle de l'application. Les contrôleurs MVC se retrouvent alors partiellement déportés dans l'entité dynamique de l'application qui assure le contrôle de la dynamique de l'application et qui gère les relations entre les objets métier et la présentation. Les contrôleurs deviennent essentiellement des contrôleurs du dialogue entre l'utilisateur et les objets métiers.



.1. Le client envoie une requête à l'application. La requête est prise en charge par le servlet d'entrée.

.2. Le servlet d'entrée analyse la requête et réoriente celle-ci vers un contrôleur adapté.

.3. Le contrôleur sélectionné par le servlet d'entrée est responsable de l'exécution des traitements nécessaires à la satisfaction de la requête. Il sollicite les objets métiers lorsque nécessaire.

.4. Les objets métiers fournissent des données au contrôleur.

.5. Le contrôleur encapsule les données métiers dans des javaBeans, sélectionne la JSP qui sera en charge de la construction de la réponse et lui transmet les JavaBean contenant les données métiers.

.6. La JSP construit la réponse en faisant appel aux javaBeans qui lui ont été transmis et l'envoie au navigateur.

.7. Lorsque nécessaire, pour le traitement d'erreurs essentiellement, le servlet d'entrée traite la requête directement, sélectionne la JSP de sortie et lui transmet par JavaBean les informations dont elle a besoin.

La taglib bean

Les tags de cette librairie peuvent être vue comme des améliorations des tags JSP <jsp :useBean> <jsp :getProperty> et <jsp :setProperty>. En effet, ils permettent de rendre accessible des objets dans les contextes standards (page, session, application) via des variables de scriptings. Il est ensuite possible de récupérer, de modifier ou d'afficher les valeurs des propriétés de ces objets.

Par exemple si la session contient un objet nommé 'item' disposant d'une méthode getPrize() et que l'on veut afficher son prix, il suffira d'écrire :

```
<bean:write name='item' property='prize'/>
```

Struts supporte également les nested et indexed properties. Par exemple si la session contient un objet nommé 'items' et a une méthode getItem(int item) qui renvoie un item. il est possible d'écrire :

```
<bean:write name='items'
property='item[2].prize'/>
```

La taglib bean propose également un tag permettant d'afficher un message en fonction de la langue de l'utilisateur. Dans l'optique de créer une application supportant plusieurs langues, les pages JSP ne doivent pas contenir de texte mais faire appel à ce tag. Dans Struts, les messages destinés à l'utilisateur sont associés à une clé et définis dans des fichiers spéciaux. Chaque fichier contient les messages dans une langue donnée. Le tag

```
<bean :message key='titre.index'/>
```

affichera le texte sous la clé titre.index dans la langue principale définie par le browser si disponible ou sinon dans la langue par défaut.

La taglib html

Cette librairie contient les tags nécessaires à la création d'interface utilisateur dynamique, et notamment des formulaires.

Les tags permettent de remplir automatiquement les champs à afficher et de retourner les valeurs en vues de leur gestion par le contrôleur. Ceci se fait en spécifiant dans le tag le nom de la propriété qui doit être affichée / modifiée. Cette propriété correspond à une propriété Javabeans d'un objet Formulaire qui aura été spécialement écrit par le développeur pour servir d'interface entre la vue et le contrôleur.

Les autres tags de la librairie permettent :

· D'afficher les messages d'erreur relatifs à l'application ou au remplissage d'un champ donné.

· De gérer plus facilement les liens et les sessions.

Voici par exemple comment créer un formulaire permettant de se loguer :

```
<html:form action= '/logon' focus='username'>

Username : <html:text property='username'/>
<br>

Password: <html:password property='password'/>

<html:submit property="submit"
value="Submit"/>

</html:form>
```

L'utilisation de ce formulaire est plus détaillée dans la section exemple.

La taglib logic

Cette librairie définit tout un ensemble de tags permettant d'inclure ou de ne pas inclure le corps des tags en fonction de critères logiques portant sur la valeur des propriétés d'un objet. Il est également possible d'effectuer des itérations.

Par exemple, pour afficher un message en fonction de l'âge de l'utilisateur :

```
<logic:greaterThan name='utilisateur'
property='age' value='17'>

Vous êtes majeurs !

</logic:greaterThan>
```

Où pour afficher tous les prénoms des enfants d'une personne :

```
<logic:iterate name='user' property='enfants'
id='enfant'>

<bean:write name='enfant'/><br> </logic
:iterate>
```

La taglib template

Cette dernière librairie est destinée à faciliter la création de pages suivant un modèle. Pour un exemple courant, imaginons la barre de navigation en haut de la page, une zone centrale et un copyright en bas. Les templates permettent de définir la barre de navigation et le copyright, puis ensuite de les inclure aisément dans toutes les pages.

.III. Installation

.III.a. Easy Way to start

Essayer Struts

Essayer Struts est très facile: la distribution binaire de Struts est fournie avec des fichiers .war parmi lesquels une application exemple. Il suffit avec Tomcat de copier ses fichiers dans le répertoire webapps après avoir installé un parseur XML **JAXP** (l'implémentation de référence de Sun [download] ou Xerces par exemple) et de redémarrer le serveur pour faire fonctionner Struts.

Les personnes désirant recompiler Struts ou utiliser d'autres serveurs trouveront les informations nécessaires sur le site de Struts.

.III.b. Expert and Full installation

.III.b.1. Pré-requis logiciel

Pour installer et utiliser Struts quelques opérations préliminaires sont nécessaires. Il faut télécharger et installer différents packages:

- **Java Development Kit** - Télécharger et installer une version de Java2 [java.sun.com/j2se].
- **Servlet Container** - Télécharger et installer un serveur pour les JSP, par exemple le serveur [Tomcat] (version 3.1 ou supérieure, version 3.2 ou plus recommandée).
- **Ant Build System** - Si vous compilez une version de Struts , il faudra utiliser , après l'avoir télécharger une version de [Ant]. Ce package est aussi recommandé pour le développement de leur application web basées sur Struts.
  - Si c'est la version 1.3 de Ant que vous utilisez, vous devrez télécharger le fichier "optional.jar" qui contient l'implémentation de la commande `<style>` de Ant.
  - Vérifier que "Ant" et le script "ant.bat" puissent s'exécuter en ajoutant le répertoire $ANT_HOME/bin à la variable d'environnement PATH.
- **JDBC 2.0 Optional Package Classes** - Struts supporte une implémentation optionnelle de `javax.sql.DataSource`, qui requiert une API pour être compiler. Pour le download [jdbc]
- **XML Parser** - Struts nécessite la présence d'un parseur XML, compatible avec le parseur XML pour l'API JAVA (JAXP 1.0 ou plus). Télécharger et installer JAXP [ici], requis pour la compilation de Struts à partir des sources de Struts. Pour les applications webs liées à Struts vous pouvez utiliser JAXP ou tout autre parseur équivalent : [Xerces]

- **Xalan XSLT Processor** - Si vous compilez une version de Struts , vous devez télécharger et installer la version 1.2 de [Xalan] XSLT processor (qui inclut le parseur XML Xerces) ou alors utiliser la version de Xalan incluse dans la version 1.1 de JAXP. Il est utilisé pour convertir la documentation interne de Struts basée sur du XML en HTML.

.III.b.2. Installer une distribution binaire de Struts

Tout d'abord, télécharger une distribution binaire de Struts [ici] vérifier d'avoir installé les logiciels précédents décrits ci dessus.

Décompressez dans un répertoire approprié la distribution de Struts. Le résultat obtenu se décompose en:

- **lib/struts.jar** - Le fichier JAR qui contient toutes les classes JAVA incluses dans struts. Copier le dans le répertoire `WEB-INF/lib` de votre application web. *Attention* - Si vous utilisez plusieurs applications basées sur struts avec le même serveur au répertoire ou vous placerez le fichier `struts.jar` au risque d'avoir une erreur : ClassNotFoundException.
- **lib/struts*.tld** - Fichiers qui devront être copiés dans le répertoire `WEB-INF`. Ils constituent une description de la librairie de tag de struts.
- **webapps/struts-blank.war** - Elle contient une archive d'une application web basique totalement vide qui servira de départ au développement d'une application struts.
- **webapps/struts-documentation.war** - Archive d'une application web basée sur struts, elle constitue aussi à la fois toute la documentation de struts.

- **webapps/struts-example.war** - Exemple d'applications web qui utilisent un large pourcentage des possibilités de struts. il peut être installé avec un serveur tomcat a partir de la version 3.1, et si il ne contient pas de parseur XML, il faudra en rajouter un dans le répertoire `WEB-INF/lib` de l'application web.

- **webapps/struts-exercise-taglib.war** - Application web contenant des pages mettant en valeur les différents tags de struts.
- **webapps/struts-template.war** - Exemple de mise en application des template tags de struts.
- **webapps/struts-upload.war** - Application utilisant le framework struts pour réaliser un upload des fichiers.

Pour utiliser Struts dans vos applications, suivez les indications suivantes:

- Copier le fichier `lib/struts.jar` a partir de la distribution de struts dans le répertoire `WEB-INF/lib` de votre application web.
- Copier tous les fichiers `lib/struts*.tld` de la distribution de struts dans le répertoire `WEB-INF` de votre application web.
- Modifier le fichier `WEB-INF/web.xml` de votre application web pour inclure la définition de l'élément `<servlet>` , et `<servlet-mapping>` pour établir le mapping des URL correspondante a cette servlet. Utiliser le fichier `WEB-INF/web.xml` de struts pour vous aidez dans la syntaxe.
- Modifier le fichier `WEB-INF/web.xml` de votre application en y introduisant

les déclarations des librairies de tags suivantes:

```
<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>
```

- Créer un fichier `WEB-INF/struts-config.xml` qui définit les actions de mapping et les autres caractéristiques de votre application. Vous pouvez utiliser votre fichier `struts-config.xml` provenant de la distribution de struts pour vous aider au niveau de la syntaxe.
- Au début de chaque JSP qui utilisera les tags de struts, il faut quelques lignes de déclarations, par exemple :

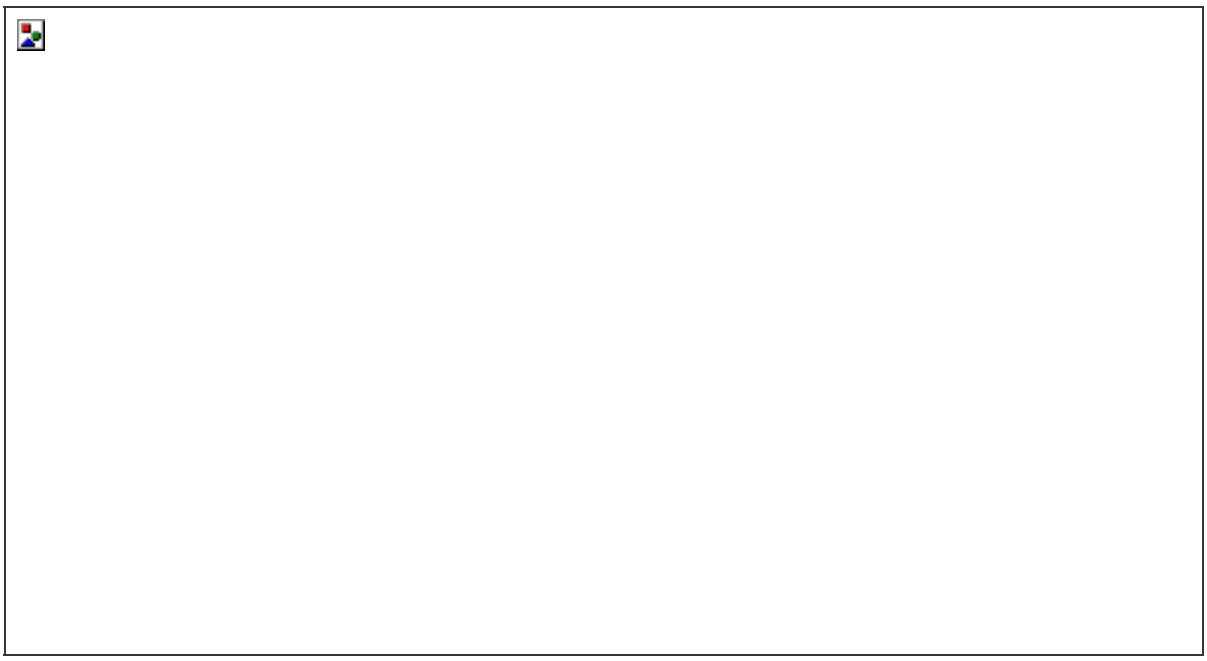
```
<@ taglib url="/WEB-INF/struts-bean.tld"
prefix="struts-bean" %>
<@ taglib url="/WEB-INF/struts-html.tld"
prefix="struts-html" %>
<@ taglib url="/WEB-INF/struts-logic.tld"
prefix="struts-logic" %>
<@ taglib url="/WEB-INF/struts-
template.tld" prefix="struts-template" %>
```

- Lorsque vous compilez soyez sur d'inclure le fichier `struts.jar` dans la variable d'environnement CLASSPATH.

.IV. Exemple

Exemple d'application Struts

Struts est fourni avec une application exemple très simple. Celle-ci montre comment effectuer des taches courantes comme afficher un formulaire, récupérer les données, les réafficher, afficher des listes dynamiques etc.



Fonctionnalités

L'application exemple permet à un utilisateur de s'enregistrer et de maintenir une liste des différents comptes mails qu'il a sur Internet. Les actions qu'il est possible d'effectuer avec cette application sont :

· Se loguer (associée par le contrôleur à la classe Logon Action)

· Se déloguer (classe LogoffAction)

· Enregistrer ses coordonnées (classe SaveRegistrationAction)

· Enregistrer les propriétés d'un compte (classe SaveSubscriptionAction)

· Afficher ses coordonnées (classe EditRegistrationAction)

· Afficher les propriétés d'un compte (classe EditSubscriptionAction)

L'application est composée de cinq pages :

· la page d'accueil, **index.jsp**,

· la page de login, **logon.jsp**,

· le menu principal, **mainMenu.jsp**,

· la page permettant d'afficher les propriétés de l'utilisateur, **registration.jsp**,

· la page permettant d'afficher la configuration d'un compte de l'utilisateur, **subscription.jsp**.

Grâce à l'emploi de la taglib logic, les pages servant à l'affichage des propriétés de l'utilisateur et de ses comptes servent aussi pour la création, la modification et la suppression. Les pages contenant des formulaires (logon.jsp, registration.jsp, subscription.jsp) sont chacune associées à un objet Form (on trouve respectivement les classes LogonForm, RegistrationForm et SubscriptionForm)

Ensemble des composants formant un traitement

Examinons en détail le processus permettant de se loguer :

La page logon.jsp contient les tags suivants :

```
<html:form action= '/logon' focus='username'> …

<html:text property='username'/> … <html:password
property='password'/> … <html:submit
property="submit" value="Submit"/> …

</html:form> …
```

ce qui provoque l'affichage d'un formulaire avec deux champs login/password et un bouton submit.

Voici ce qui se passe lorsqu'on clique sur submit :

1. La requête est envoyée à l'adresse logon.do à laquelle va répondre le contrôleur de Struts.

2. Le contrôleur utilise le fichier struts-config.xml pour déterminer quel est le formulaire qui va permette la validation des données (dans notre exemple il s'agit de LogonForm) et quelle est l'action associée à cette requête (dans notre exemple il s'agit de LogonAction).

Extraits du fichier struts-config.xml :

```
…

<!-- Logon form bean -->

<form-bean name="logonForm"
type="org.apache.struts.example.LogonForm"/> …

<!-- Process a user logon -->

<action path="/logon"
type="org.apache.struts.example.LogonAction"
name="logonForm"

scope="request" input="/logon.jsp">

</action> …
```

1. Le servlet va donc instancier un LogonForm et l'initialiser avec les valeurs des attributs fournis dans la requête (password et username). Pour se faire LogonForm propose les méthodes setUsername et setPassword :

```
public class LogonForm extends ActionForm { …
public void setPassword(String password) { …}
public void setUsername(String username) {…} public
String getUsername() {…} public String
getPassword() {…} … }
```

2. Le servlet va ensuite appeler la méthode validate() de LogonForm. Celle-ci renvoie une erreur si au moins un des deux champs est vide.

```
public class LogonForm extends ActionForm { …
public ActionErrors validate( ActionMapping
mapping, HttpServletRequest request) { ActionErrors
errors = new ActionErrors();

if ((username == null) || (username.length() < 1))
errors.add("username", new

ActionError("error.username.required"));

if ((password == null) || (password.length() < 1))
errors.add("password", new ActionError(
"error.password.required")); return errors; } }
```

3. Si il n'y a pas d'erreur, le contrôleur va appeler la méthode perform() de LogonAction. Sinon il va renvoyer à la page de login.

4. La méthode perform() contient le code métier associé à une opération d'authentification.

```
public class LogonAction extends Action { ... public ActionForward
perform( ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response) throws
... { ... // récupération des paramètres String username =
((LogonForm) form).getUsername(); String password =
((LogonForm) form).getPassword();

// recuperation de la base de données Hashtable database =
(Hashtable) servlet.getServletContext().getAttribute(..) ; ...

// création de l'utilisateur // (utilisation du modèle)

User user = (User) database.get(username); if (user!=null &&
(!user.getPassword().equals(password))) user=null; if (user==null)
errors.add(ActionErrors.GLOBAL_ERROR, new
ActionError("error.password.mismatch"));

// si on a des erreurs, retour à la page de login if (!errors.empty()) {
saveErrors(request, errors); return (new
ActionForward(mapping.getInput())); }

// ok, sauvegarde de l'utilisateur

HttpSession session = request.getSession();
session.setAttribute(Constants.USER_KEY, user); ...

// renvoie à la page suivante return
(mapping.findForward("success")); }
```

Le code source complet de l'application est fourni avec Struts.

Extension du framework

Struts a été conçu de façon à pouvoir être aisément modifié par les développeurs pour répondre à leurs besoins particuliers. Cela se voit dans l'application exemple, qui redéfinit la classe ActionMapping pour introduire la notion de 'success' et de 'failure' pour une action. Des tags spécifiques sont également introduits pour vérifier si un utilisateur est logué ou non et pour écrire plus facilement certains liens.

.V. Conclusion

Struts est un framework qui permet de développer des applications web en utilisant le modèle MVC2. Les applications conçues ainsi sont plus facilement maintenables et évolutives, de part la séparation des fonctionnalités du modèle, du contrôle, et de la vue. L'utilisation du framework permet au développeur de se concentrer sur le modèle, Struts fournissant le cadre nécessaire au contrôle et à l'interface de l'application, permettant un développement rapide de ces parties. Struts préfigure une nouvelle méthode de développement, succession de la programmation par servlets. Néanmoins, il convient d'ajouter que l'écriture d'une application utilisant Struts ne doit pas se décider à la légère : un certain temps est nécessaire pour prendre en main le concept et les API.

Le déploiement d'un servlet unique dont le comportement est configuré par fichier donne une grande flexibilité à la gestion de la navigation de l'application. Ce fonctionnement limite, par ailleurs, considérablement les adhérences entre les applications développées et les serveurs d'application sur lesquels elles sont déployées. Cet avantage est cependant estompé par la généralisation du support, par les différents serveurs d'application, des formats de déploiement standardisés (format WAR notamment).

La centralisation des accès à l'application permet un contrôle fin et personnalisé des accès aux traitements (Actions) et offre une grande marge de manœuvre pour la gestion des profils ou des rôles utilisateurs. L'abstraction des traitements et des ressources par le modèle des commandes, en occultant les informations relatives à la structuration de l'application, apporte une plus grande sécurité de cette dernière. Struts fait partie des projets Open Source Jakarta de l'ASF (Apache Software Foundation). A ce titre les sources du framework sont publiques. **Le framework Struts en conséquence aisément adaptable et extensible**. Son appartenance au projet Jakarta d'Apache garantit par ailleurs sa pérennité et la capacité importante de développement, dans un but opérationnel et non de recherche et développement, que la communauté Java lui consent.

Enfin, Struts suscitant un grand intérêt de la part de la communauté Java, de nombreux articles et exemples de mise en œuvre ainsi que diverses extensions sont aujourd'hui publiées sur divers site Internet, que ce soit ou non sous l'égide du projet officiel.

.VI. Liens

Tutorial en version html, pour obtenir tous les liens disponibles à

http://www.nuxora.com/

Bibliographie :

http://arkzoyd.free.fr/struts902/struts-01.html

http://www.application-servers.com/

http://www.jspinsider.com/tutorials/jsp/struts/struts.view

http://www.waferproject.org/index.html

http://www.improve-institute.com/formation_jakarta_struts.html

http://struts.application-servers.com/

http://www-106.ibm.com/developerworks/library/j-struts/?dwzone=java

http://www.waferproject.org/index.html

Le projet Struts http://jakarta.apache.org/struts

Le serveur Tomcat http://jakarta.apache.org/tomcat

Le parseur XML de Sun http://java.sun.com/xml