

Plan

- Problématique
- Bref Historique
- Solution JSF
 - Principes
 - Mise en Œuvre des principes
- JSF vs Struts
- Conclusion

2

Problématique

- Evolutions des besoins : Applications Web
- Problème : Comment concevoir une application Web de la même façon qu'une application traditionnelle
- Plusieurs solutions existent :
 - WebForms : Microsoft
 - Struts : Apache
 - JSF : Sun

3

Bref Historique

	Sun	Open-source	Microsoft
1996	Servlet	PHP, PERL, etc...	ASP
1997			
1998	JSP		
		MVC	
			ASP.NET, WebForms
2000		Struts	
2004	JSF		

- Divergences d'approche selon les éditeurs.
- Plus grande séparation des concepts.

4



Solution JSF

Framework de création
d'interfaces graphiques pour les
applications Web

Un concept en évolution

- Demande de spécification en Mai 2001
- Plusieurs versions de spécifications Sun
 - Final : 1.0 (début 2004)
 - Maintenance : 1.1 (fin 2004)
 - Aujourd'hui : 1.2 / 2.0
- Plusieurs implémentations
 - Référence (Sun/Oracle) compatible 1.2
 - MyFaces (Apache)

6

JSF et MVC

- Modèle :
 - Couplé à JSF par un Bean géré.
- Vue :
 - JSP + balises JSF
- Contrôleur :
 - Servlet (FacesServlet)
 - Règles définies dans un fichier xml

7

JSF

- MVC 2 « simple » ?

= struts ?

8

JSF va plus loin

- Possibilité d'étendre les différents modèles et de créer ses propres composants
- Configuration de la navigation entre les pages
- Support de l'internationalisation
- Support pour l'utilisation par des outils graphiques

9

JSF se compose:

- d'une spécification qui définit le mode de fonctionnement du framework
- d'une API : l'ensemble des classes de l'API est contenu dans les packages javax.faces
- d'une implémentation de référence
- de bibliothèques de tags personnalisés fournies par l'implémentation pour
 - utiliser les composants dans les JSP
 - gérer les événements
 - valider les données saisies


10

Principes

Département
d'Informatique

Principes : composants graphiques

- Fonctionnalité du composant définie dans la classe du composant
- Plusieurs rendus pour un même composant
- Possibilité de définir des rendus

Etiquette	Rendu
h:command_button	
h:command_link	hyperlink

12

Principe : gérer des événements

- Chaque composant génère des événements (Events)
- Le serveur possède des écouteurs (Listeners) qui traitent les événements

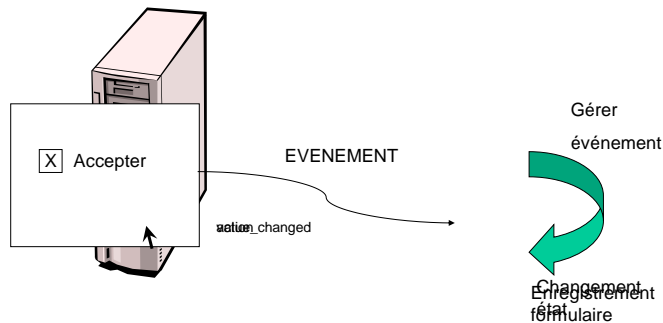
13

Principes : gestion de l'état des composants

- L'état peut changer chez le client entre 2 requêtes
- JSF prend en charge la gestion de cet état

14

Exemple d'événements



15

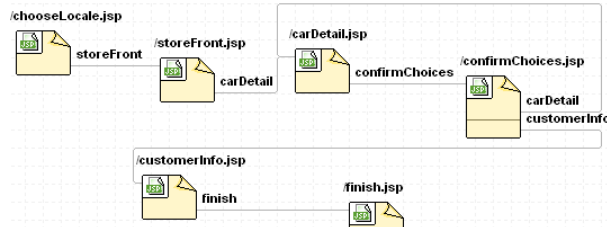
Principes : validation et conversion

- Validation :
 - Agit sur les composants graphiques (textfield)
 - Personnalisation des validateurs (bornes, types, etc.)
 - Lève des erreurs
- Conversion :
 - Formatage des données
 - Ex : 15000 → 15,000
 - Personnalisation possible

16

Principes : navigation des pages

- Définitions de l'enchaînement des pages par un ensemble de règles de navigation



17

Principes : internationalisation

- Possibilité de définir des locales
- Correspondance
 - clé → valeur internationalisée

```

useridLabel=UserId
passwordLabel=Password
loginSubmitLabel=Login
  
```

```

useridLabel=Nom de l'Utilisateur
passwordLabel=Mot de Passe
loginSubmitLabel=Connectez
  
```

18

Principes : balises JSP

- Rappels JSP:
 - Page html qui peut contenir du code Java
 - A l'exécution cette page est convertie en servlet
 - Contient des balises JSP
 - JSTL (bibliothèque de balises standard pour java)
- JSP propose une bibliothèque de balises
- Utilisation du langage EL :
 - version JSP : \${...} évaluation immédiate
 - version JSF : #{...} évaluation tardive

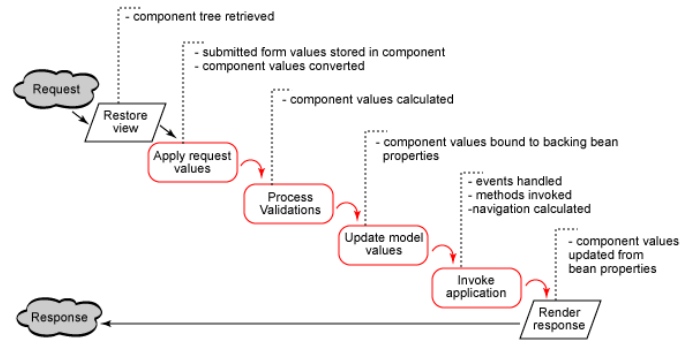
19

Cycle de vie d'une requête

- Création de l'arbre de composants
- Extraction des données des différents composants de la page
- Conversion et validation des données
- Extraction des données validées et mise à jour du modèle de données (javabean)
- Traitements des événements liés à la page
- Génération du rendu de la réponse

20

Cycle de vie d'une requête



21

Configuration d'une application

- La structure est celle d'une application j2ee

```

/
/WEB-INF
/WEB-INF/web.xml
/WEB-INF/lib
/WEB-INF/classes
  
```

22

Les bibliothèques nécessaires

```

jsf-api.jar
jsf-ri.jar
jstl.jar
standard.jar
common-beanutils.jar
commons-digester.jar
commons-collections.jar
commons-logging.jar
  
```

23

Configuration : web.xml

```

<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Test JSF</display-name>
  <description>Application de tests avec JSF</description>
  <context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>client</param-value>
  </context-param>

  <!-- Servlet faisant office de controleur -->
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup> 1 </load-on-startup>
  </servlet>

  <!-- Le mapping de la servlet -->
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.faces</url-pattern>
  </servlet-mapping>
</web-app>
  
```

24

Configuration: faces-config.xml

- Ce fichier est au même niveau que web.xml
- Il peut aussi être découpé en plusieurs fichiers différents
 - Dans ce cas:

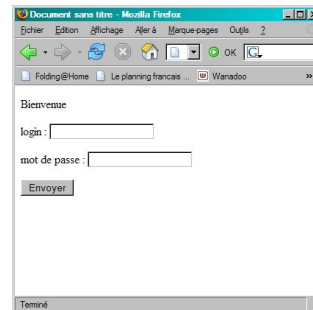
```
...
<context-param>
  <param-name>javax.faces.application.CONFIG_FILES</param-name>
  <param-value>
    /WEB-INF/ma-faces-config.xml, /WEB-INF/navigation-faces.xml, /WEB-INF/beans-faces.xml
  </param-value>
</context-param>
...
```

25

Mise en Œuvre des principes

Etude de cas

- Une page de login
 - Un seul utilisateur valide
 - 3 pages :
 - Authentification
 - Accueil (si réussie)
 - Erreur (sinon)



27

1. Le bean du support

- Dans un package jsfLogin

LoginServer
-userid : String
-password : String
+getUserId() : String
+setUserId(userid : String)
+setPassword(password : String)
+getPassWord() : String
+loginAction()

28

2. Déclaration du bean dans face-config.xml

```
<managed-bean>
  <managed-bean-name>LoginServer</managed-bean-name>
  <managed-bean-class>jsflogin.LoginServer</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

29

3. Création des pages

■ Import des librairies

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
```

■ Ressources (internationalisation)

```
<f:loadBundle basename="jsflogin.Resources" var="jsfloginBundle"/>
```

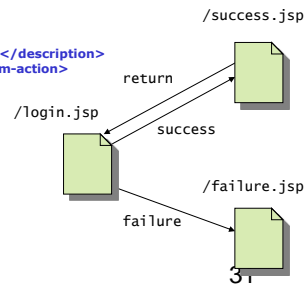
■ Utilisation des balises

```
<h:form id="loginForm">
  <h:input_text id="userid" required="true" value="#{LoginServer.userid}" >
    <f:validate_length minimum="4" maximum="7" />
  </h:input_text>
  <h:command_button id="submit" action="#{LoginServer.loginAction}"
    value="#{jsfloginBundle.loginSubmitLabel}" />
</h:form>
```

30

4. Navigation des pages face-config.xml

```
<navigation-rule>
  <from-view-id>/login.jsp</from-view-id>
  <navigation-case>
    <description>Handle case for successful login</description>
    <from-action>#{LoginServer.loginAction}</from-action>
    <from-outcome>success</from-outcome>
    <to-view-id>/success.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <description>Handle case for unsuccessful login</description>
    <from-action>#{LoginServer.loginAction}</from-action>
    <from-outcome>failure</from-outcome>
    <to-view-id>/failure.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <from-view-id>/success.jsp</from-view-id>
  <navigation-case>
    <from-outcome>return</from-outcome>
    <to-view-id>/login.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```



31

Bilan

- JSF vs Struts
- Attentes respectées ?

JSF vs Struts (1/3)

- Rappels Struts :
 - Framework d'application Web
 - Licence Apache
 - Mêmes principes (MVC, fichier de navigation)
 - Pas de spécifications formalisées

33

JSF vs Struts (2/3)

- Struts est plus mature
- Meilleur facteur de confiance pour JSF (Ex: support IBM pour WebSphere)
- Struts contraint le modèle, JSF est plus flexible

34

JSF vs Struts (3/3)

- La vue est plus générique dans JSF grâce à sa conception des composants
- JSF est plus extensible
 - Balises personnalisés
 - Composants personnalisés
- Conclusion :
 - JSF a su tirer expérience de Struts

35

Attentes respectées (1/2)

- JSF est-il une bonne solution ?
 - Standardisé
 - Règles de navigation
 - Mise en œuvre des composants graphiques
 - Approche RAD

36

Attentes respectées (2/2)

- MVC
 - Modèle indépendant
 - Séparation du comportement et du rendu
 - Approche page du contrôleur : spécifique aux applications Web

37

Conclusion

Le futur de ce Framework

- Intégration de JSF 1.2 à la spécification J2EE 5.0
- Participation de l'industrie à la spécification
- Pas de conversion Struts → JSF
- Beaucoup de projets/compétences en Struts
- Adoption de JSF ou Struts 2 pour les nouveaux projets

- A FAIRE : tuto JSF sur vogella.de
<http://www.vogella.de/articles/JavaServerFaces/article.html>

39