http://www.sigl.epita.net

Struts

Romain Couturier
Aurélia Fermé
Frédéric Lung Tung
Matthieu Nicolas

17/06/2002

- Rappels
  - Servlet
  - JSP
  - JavaBean
  - MVC (1 & 2)
- Présentation
- Vue
- Contrôleur
- Modèle
- Exemple d'application

Servlet – Rappel(1/8)

- Programme java permettant d'étendre les fonctionnalités d'une page Web
- Application côté serveur
- Utilisé pour générer du contenu dynamique
- Chargée dynamiquement
- Avantages
  - Indépendance issue de la plateforme java
  - Modèle de sécurité issu du serveur Web
  - Support dans la plupart des serveurs Web
  - Exploite toute l'API Java (+ protocoles)

JSP – Rappel(2/8)

- Séparent la présentation du contenu
- Les fichiers JSP contiennent:
  - un code pour le traitement du serveur (JSP et Java)
  - un code pour le traitement du client (HTML et Javascript)
- Avantages
  - génère une page vers le client
  - est portable (Write Once, Run EveryWhere)
  - met en avant l'approche par composants
  - permet la mise en œuvre facile des sites dynamiques
- Inconvénients
  - difficile à manier en raison du mélange de code
  - codes difficiles à réutiliser
  - documentation difficile
- Equivalents : ASP, PHP

JavaBean – Rappel(3/8)

- Environnement d'exécution distribué intégrant :
  - Moniteur transactionnel
  - Support de persistance
- Permettent :
  - le développement des solutions métier
  - Lier des entités autonomes sur une plate–forme
- L'architecture EJB réutilise des technologies Java :
  - RMI-IIOP : invocation de méthodes distantes
  - JNDI : accès à un service de nommage
  - JDBC : connexion à des bases de données
  - JTS : gestion des transactions (spec. basées sur CORBA OTS)
  - JMS : communications asynchrones
  - JSP/servlet : clients web

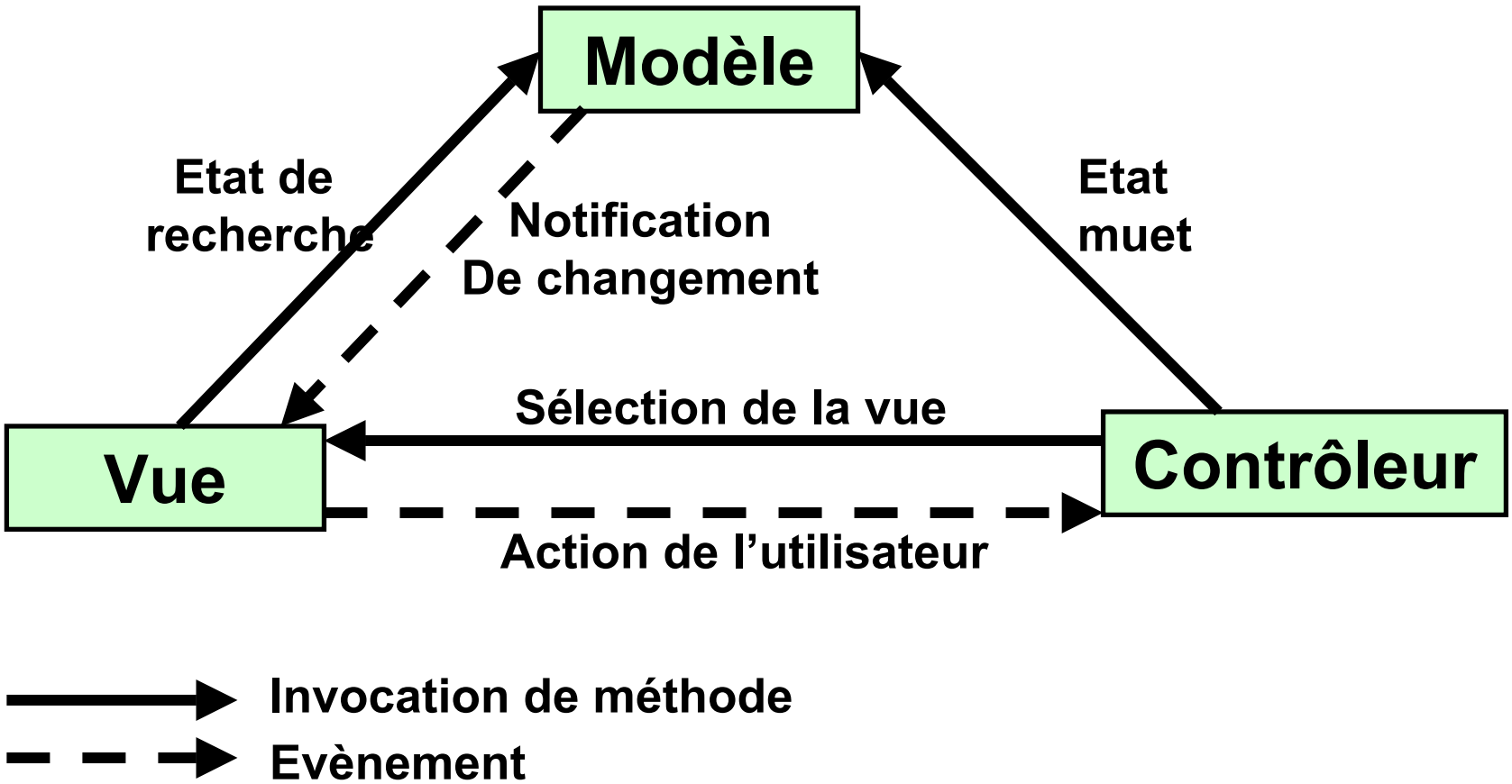
MVC – Rappel(4/8)

- Origine : SmallTalk-80
- Le design pattern MVC permet de séparer la logique métier (Modèle) de l'interface utilisateur (Vue) et des flux (Contrôleur)
- Avantages :
  - Modularité
  - Plusieurs vues possibles pour un même Modèle
  - Réutilisabilité

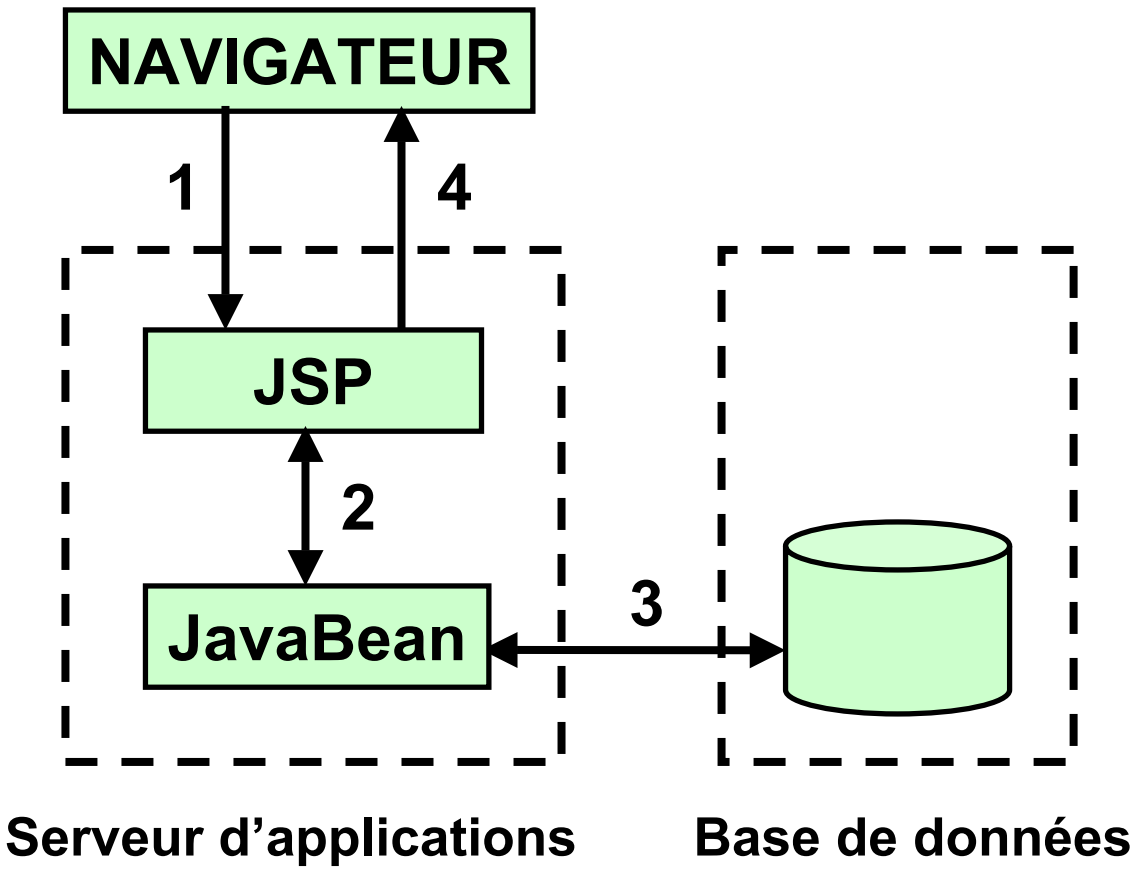
MVC – Rappel(5/8)

- MVC divise le système en 3 parties :
  - Modèle
    - Représente les données et les règles d'accès et d'actualisation de ces données
  - Vue
    - Présente le contenu du modèle, comment les données sont présentées à l'utilisateur
  - Contrôleur
    - Fait le lien entre la Vue et le Modèle. Contrôle les flux de l'application. Traduit les interactions avec la "Vue" des actions qui doivent être réalisées

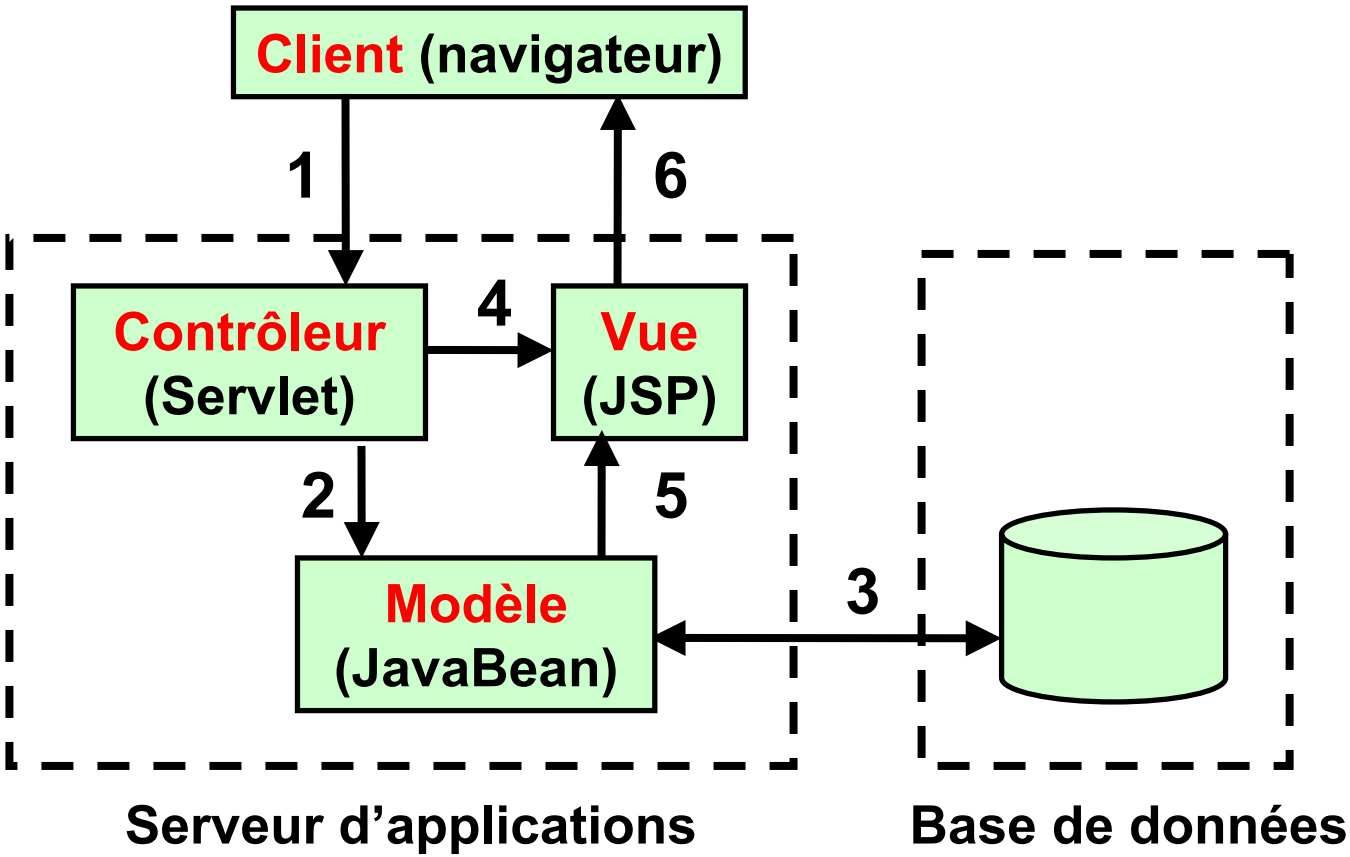
MVC – Rappel(6/8)



MVC 1 – Rappel(7/8)



MVC 2 – Rappel(8/8)



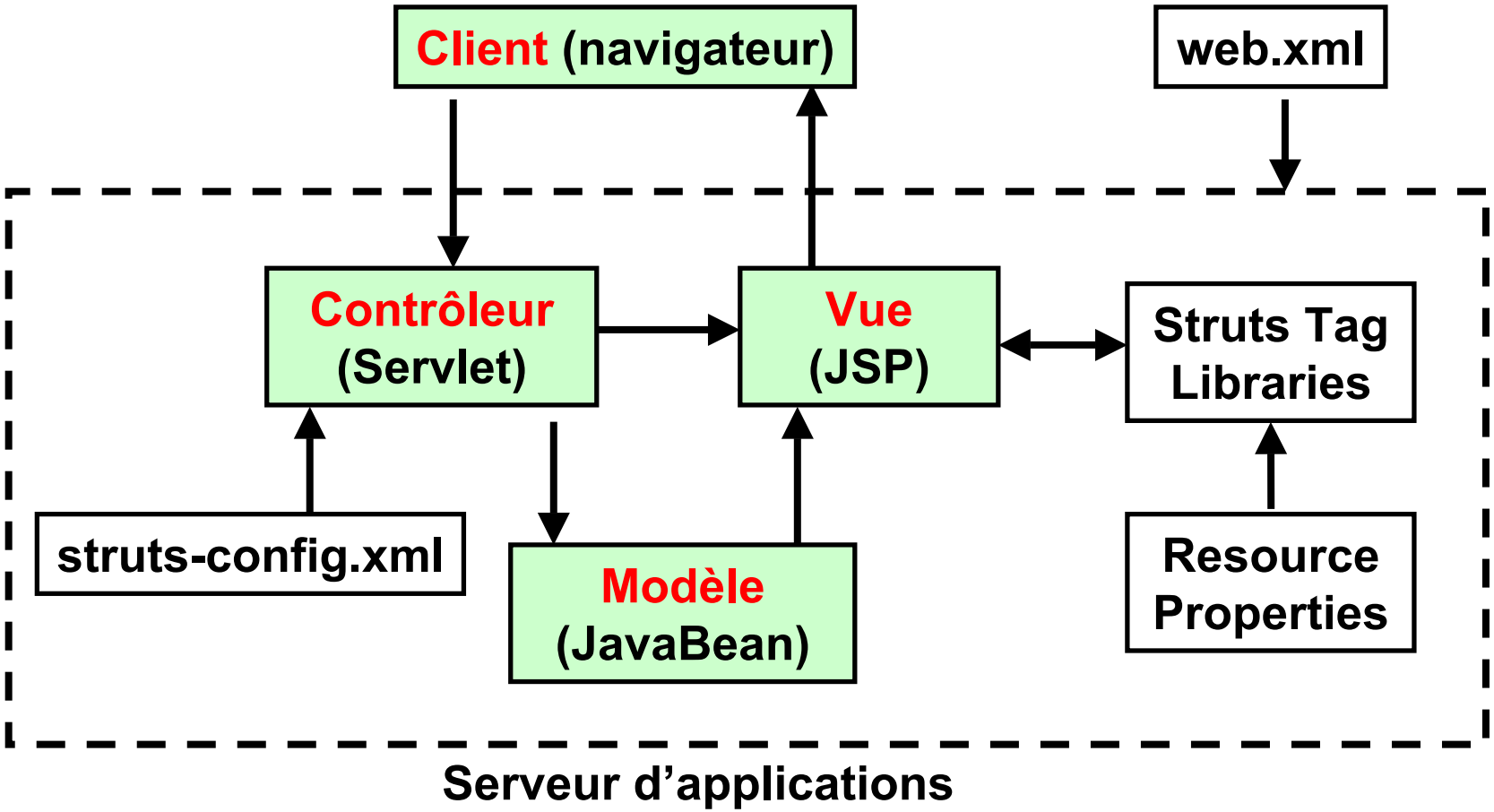
Présentation (1/3)

- Projet Struts initié en mai 2000 par Craig R. McClanahan
- Sortie de Struts version 1.0 en juin 2001
- Implémentation open source du framework MVC par le groupe Jakarta (jakarta.apache.org)
- Développement très actif
- Version stable 1.0.2

Présentation (2/3)

- Les composants MVC avec Struts:
  - La vue : JSP ou page HTML qui renvoie la réponse HTTP
  - Le modèle : des objets Java (JavaBean)
  - Le contrôleur : Servlet qui dispatche toutes les requêtes. Utilise le fichier struts-config.xml pour son initilisation

Présentation (3/3)



Vue (1/3)

- Séparation des rôles : maquettage IHM à l'aide de pages JSP
- Utilisation des taglibs pour les éléments du formulaire
- Donne accès aux pages Web du côté serveur
- Représente les informations du modèle

Vue (2/3)

- La librairie de tags struts :
  - Struts-bean fait de la manipulation pure des beans
    - <bean:write name="personne" property="prenom" /> affiche le prénom contenu dans un objet à l'aide de la méthode getPrenom()
  - Struts-html manipule des formulaires et des éléments de formulaires html pour la création d'IHM dynamique

```
<html:form action="/logon">
User name : <html:text property="username" size="16"/>
<br>
Password : <html:password property="password" size="16"/>
<html:submit property="submit" value="Submit"/>
```

Vue (3/3)

- La librairie de tags struts :
  - Struts-logic met en place des traitements conditionnels et/ou itératifs :

```
<logic:greaterThan name="utilisateur" property="age" value="17">
Vous êtes majeur
</logic:greaterThan>
```

    - 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>
```

  - Struts-template propose un mécanisme de gestion de modèles de JSP
    - Barre de navigation, zone de copyright…

Contrôleur (1/3)

- Dirige l'utilisateur sur la vue approprié
- Le mapping est effectué avec ActionServlet

http://myhost/authorize.do

↓

Le serveur est configuré pour passer les extensions ***.do** extensions à
org.apache.struts.action.ActionServlet
via un fichier de configuration web.xml

↓

Les objets ActionServlet inspectent les URI et essayent de les "matcher" avec un ActionMapping situé dans le fichier `struts-config.xml`.

↓

L'instance appropriée de la classe Action est trouvée et la méthode perform() est appelée

`L'objet` *`Action`* traite la requête et retourne le contrôle à la vue

Contrôleur (2/3)

- Servlet ActionServlet
  - Reçoit toutes les requêtes au travers d'un mapping avec le fichier "web.xml"
  - Lit le fichier struts-config.xml à l'initialisation et développe ActionMappings
  - Avec la base de l'URI de chaque requête, dispatche le traitement pour la classe Action adéquate

Contrôleur (3/3)

- Classe Action
  - Aide à contrôler les flux de l'application
  - En accord avec la requête, traite la logique implémentée dans le Modèle
  - Actualise les JavaBeans qui seront utilisés pour développer la prochaine page
  - Dispatche le contrôle pour le composant Vue adéquate
  - Bon endroit pour valider la session de l'utilisateur

Modèle (1/2)

- Code indépendant des framework
- JavaBeans enregistre l'état du système et implémente les actions qui peuvent être réutilisées sur cet état (logique de négociation)
  - System State Beans
  - Business Logic Beans
  - ActionForm Beans

Modèle (2/2)

- ActionForm Beans
  - Enregistre les données d'une formule automatiquement
  - Définissent une propriété et les méthodes getXxx() et setXxx() pour chaque élément du formulaire
  - Possède un mécanisme de validation
  - Sont passés à la classe Action


Exemple
d'application
avec Struts

Structure du projet

- joinMVC.jsp : page appelée
- Struts-config.xml : contient le mapping des actions
- joinAction.java : classe de traitement appelée
- joinForm.java : classe relative au formulaire
- welcome.jsp : page résultante

Exemple JSP simple

```
<%@ page language="java" %>
<%@ page import="business.util.Validation" %>
<%@ page import="business.db.MailingList" %>
<%
String error = "";
String email = request.getParameter("email");
// do we have an email address
if( email!=null ) {
     // validate input...
     if( business.util.Validation.isValidEmail(email) ) {
          // store input...
          try {
               business.db.MailingList.AddEmail(email);
          } catch (Exception e) {
               error = "Error adding email address to
    system. " + e;
          }
          if( error.length()==0 ) {
%>
     // redirect to welcome page...
     <jsp:forward page="welcome.html"/>
<% }
          } else {
          // set error message and redisplay page
          error = email + " is not a valid email address,
    please try again.";
     }
} else {
     email = "";
} %>
```

```
<html>
<head>
<title>Join Mailing List</title>
</head>
<body>
<font color=red><%=error%></font><br>
<h3>Enter your email to join the group</h3>
<form action="join.jsp" name="joinForm">
     <input name="email" id="email"
    value=<%=email%>></input>
     <input type=submit value="submit">
</form>
</body>
</html>
```

Struts par l'exemple

- Tags Struts

```
<form:form action="join.do" focus="email" >
     <form:text property="email" size="30" maxlength="30"/>
     <form:submit property="submit" value="Submit"/>
</form:form>
```

- HTML résultant envoyé au navigateur

```
<form name="joinForm" method="POST" action="join.do;jsessionid=ndj71hjo01">
     <input type="text" name="email" maxlength="30" size="30" value="">
     <input type="submit" name="submit" value="Submit">
</form>
<script language="JavaScript">
     <!-- document.joinForm.email.focus() // -->
</script>
```

Fichier JSP revisité

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts.tld" prefix="struts" %>
<%@ taglib uri="/WEB-INF/struts-form.tld" prefix="form" %>
<html>
<head>
<title><struts:message key="join.title"/></title>
</head>
<body bgcolor="white">
<form:errors/>
<h3>Enter your email to join the group</h3>
<form:form action="join.do" focus="email" >
     <form:text property="email" size="30" maxlength="30"/>
     <form:submit property="submit" value="Submit"/>
</form:form>
</body>
</html>
```

Contrôleur – struts-config.xml

- Définition et configuration
  - de l'origine des données
  - des forms-beans
  - des termes globaux
  - des mapping d'action

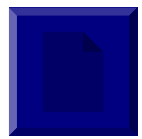
- ActionMapping
  - Evènement respectif de chaque classe d'action
  - Path : URI
  - Type : nom de la classe qui met ces actions en œuvre
  - Name : FormBean qui utilise ces Action

JoinAction

- Résulte du dispatch des actions
- Surcharge la méthode perform()
- Retourne une classe ActionForward qui indique au contrôleur où aller ensuite

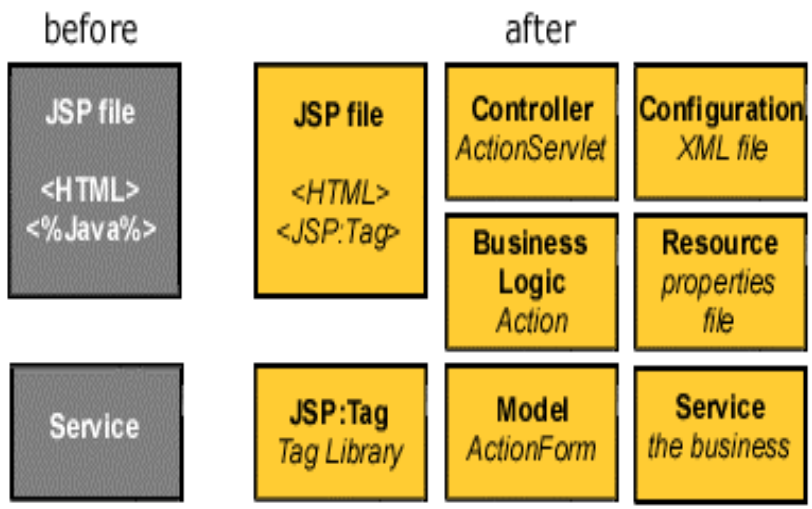
JoinForm

- Contient toutes les données du formulaire
- Surcharge la méthode validate() pour la validation du formulaire
- Contient des setters et des getters



Pour résumé

- joinForm est utilisé pour contenir les informations du formulaire
- Si la validation est activée, joinForm essaiera de se valider lui-même
- web.mailinglist.JoinAction est la classe utilisée pour traiter les requêtes pour ce mapping
- Si tout fonctionne, welcome.jsp est appelé
- S'il y a une erreur dans la logique businee, le flux retourne à joinMVC.jsp qui est la page original qui fait la requête




Conclusion

Les forces

- Intégration avec J2EE
- Open Source
- Support des taglib
- Cohabite avec les applications existantes
- Gestion des erreurs (via ActionError)
- Maintenance à long terme modulaire

Les problèmes

- Connaissances approfondies pour l'apprentissage et la maintenance (peu intéressant pour les petits projet)
- Quelques problèmes d'intégration avec les environnements non-Tomcat
- Framework encore jeune

Dans le futur

- Java Server Faces
  - Standardisation du modèle MVC
  - Création d'un framework GUI standard
  - Promouvoir le modèle de JavaBeans pour « dispatché » les évènements côté client vers le code côté serveur

Ressource

- Struts homepage
  - http://jakarta.apache.org/struts/
- Sun's Servlet Specification
  - http://java.sun.com/products/servlet/download.html#specs
- Sun's JSP Specification
  - http://java.sun.com/products/jsp/download.html


Questions