Plan

Partie 1: Rappel

- Chapitre 1 : Plateforme JEE - Introduction
- Chapitre 2 : Servlets
- Chapitre 3 : JSP

Partie 2:

- Chapitre 4 : Java Bean
- Chapitre 5 : Le modèle MVC
- Chapitre 6 : Frameworks MVC: struts
- Chapitre 7 : Persistance en Java : EJB et JPA

JavaBean

☞ **Les JavaBean**

- Les JavaBeans sont des classes classiques qui proposent des méthodes « accesseurs » pour accéder à leurs attributs.
- La construction des beans doit respecter des règles bien précises.
- Un bean est attaché à une page en se comportant comme une variable que l'on peut utiliser en tout endroit de cette page.

JavaBean

☞ **...et en Java**

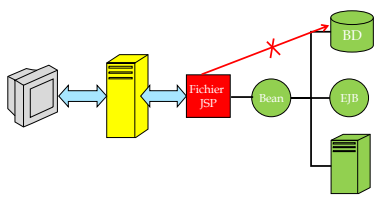
- Un beans
  - ✓ Doit posséder un constructeur vide;
  - ✓ Doit exposer des propriétés, sous forme de paires getters / setters.
  - ✓ Doit déclaré les attributs comme privés
- Les JavaBeans sont déployés sous WEB-INF/classes

JavaBean

☞ **Concrètement, pourquoi les beans ??**

- Le JavaBean sert à la capture d'informations et la garder durant une session.
- Permet de passer l'information entre pages
- Utilisée pour effectuer la validation des données (logique de présentation)

JavaBean



JavaBean

☞ **Exemple**

```
package produits;

public class Produit {
  private String marque;
  private int prix ;

  public String getMarque() {
    return marque;
  }
  public void setMarque(String marque) {
    this.marque = marque;
  }
  public int getPrix() {
    return prix;
  }
  public void setPrix(int prix) {
    this.prix = prix;
  }
}
```

JavaBean

☞ **Exemple**

- <jsp:useBean id="nomId" class="Classe" />
  Permet de référencer une Java bean.
- <jsp:setProperty name="nomId" property="nomP" value="Val" />
  Permet d'appeler la propriété set d'une Java bean
- <jsp:getProperty name=" nomId " property="nomP" />
  Permet d'appeler la propriété get d'une Java bean

```
<HTML>
  <HEAD><TITLE>Test du Bean </TITLE></HEAD>
  <BODY>
    <jsp:useBean id="MonBean" scope="session"
          class="produits.Produit"
     type="produits.Produit"/>

   Définition de la marque et du prix :<P>

    <jsp:setProperty name="MonBean"
          property="marque" value="Dell" />
<BR>
    <jsp:setProperty name="MonBean"
          property="prix" value="2900" /> <BR>

      Récupération de la marque et du prix
     du produit :<P>
     La marque est:<jsp:getProperty
name="MonBean"
                    property="marque" /> <BR>
     Le prix est:<jsp:getProperty
name="MonBean"
                    property="prix" /> <BR>
   </BODY>
<HTML>
```


113

JavaBean

☞ **Interaction entre formulaire et action (ou pages JSP)**

- On peut remplir les beans à partir des formulaires soit manuellement soit automatiquement.
- Manuellement :

```
<HTML><TITLE>JavaBeans et  JSP</TITLE>
<BODY>
<P><
H1>Somme de deux nombres </H1>
<jsp:useBean id="test1" class="produits.produit" />
<jsp:setProperty name="test1"   property="marque"    value="<%= request.getParameter("c1") %>" />
<jsp:setProperty name="test1"   property="prix"   value="<%= request.getParameter("c2") %>" />
</BODY>
```

```
<form action="JSP/calform.jsp" method=GET>
    Marque: <input type=text name=c1 >
    Prix: <input type=text name=c2 >
    <input type=submit>
</form>
</BODY>
```


114

JavaBean

☞ **Interaction entre formulaire et action (ou pages JSP)**

- Automatiquement (ou aussi):

```
<HTML><TITLE>JavaBeans et  JSP</TITLE>
<BODY>
<P><
H1>Somme de deux nombres </H1>
<jsp:useBean id="test1" class="produits.produit" />
<jsp:setProperty name="test1"   property="marque"    param= " marque " />
<jsp:setProperty name="test1"   property="prix"   param= " prix " />
</BODY>
```

```
<form action="JSP/calform.jsp" method=GET>
    Marque: <input type=text name=marque >
    Prix: <input type=text name=prix >
    <input type=submit>
</form>
</BODY>
```


115

JavaBean

☞ **Interaction entre formulaire et action (ou pages JSP)**

- automatiquement:

```
<HTML><TITLE>JavaBeans et  JSP</TITLE>
<BODY>
<P><
H1>Somme de deux nombres </H1>
<jsp:useBean id="test1" class="produits.produit" />
<jsp:setProperty name="test1"   property="*"    />
</BODY>
```

```
<form action="JSP/calform.jsp" method=GET>
    Marque: <input type=text name=marque >
    Prix: <input type=text name=prix >
    <input type=submit>
</form>
</BODY>
```

Si property="*" dans jsp:setProperty , alors les propriétés invoquées correspondent aux champs de mêmes noms.

116

Plan

117

Le modèle MVC

☞ **Introduction**

- Objectif :
  - organiser une application interactive en séparant :
    - ✓ les données (Modèle)
    - ✓ la représentation des données (Vues)
    - ✓ le comportement de l 'application (Contrôle)

118

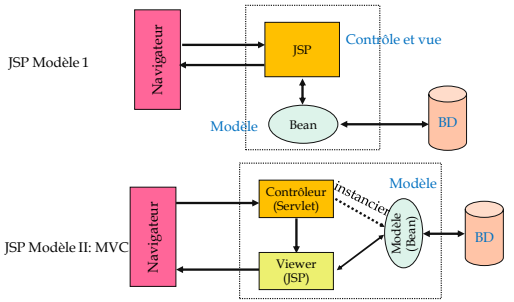
Le modèle MVC

Introduction

- Le modèle (M) représente la structure des données dans l 'application
- La vue (V) présente les données sous une certaine forme à l 'utilisateur.
- Le contrôleur (C) traduit les interactions utilisateur par des appels de méthodes sur le modèle et sélectionne la vue appropriée.

Le modèle MVC

MVC pour les applications web



Le modèle MVC

Le modèle avec les beans

- Chaque java bean représente une classe du modèle.
- Cette classe et caractérisé par :
  - ✓ des attributs (propriétés et attributs d'associations avec les autres classes) qui sont souvent privés,
  - ✓ les accesseurs (get..), mutateurs (set..)
  - ✓ des méthodes métiers qui assurent les différents traitements du métier et la persistance de ces objets (souvent dans une base de données relationnelle).

Le modèle MVC

La vue avec JSP

- Les vues sont implémentées par les pages JSP. Une page JSP se charge de :
  - ✓ Récupérer les données des résultats stockés, préalablement par le contrôleur, dans le un bean.
  - ✓ Afficher ces résultats dans les parties dynamiques de la page HTML
  - ✓ Afficher les autres parties statiques, de la vue, qui vont permettre à l'utilisateur d'envoyer d'autres requêtes vers le contrôleur via des formulaires ou des liens hypertextes.

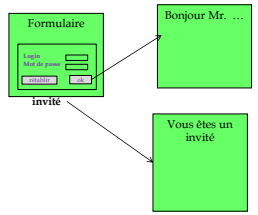
Le modèle MVC

Le contrôle avec les servlets

- Permet de rediriger des requêtes
- Les servlets peuvent jouer le rôle du contrôleur. Ce dernier se charge de :
  - Recevoir les différentes requêtes http des utilisateurs et leurs données
  - Stocker ces données dans un objet intermédiaire associé à la requête et appelé bean
  - Faire appel au modèle, qui se charge du traitement, en lui transmettant les données de la requête.
  - Récupérer les résultats éventuels retournés par le modèle.
  - Stocker ces résultats dans un bean et la stocker dans la session courante.
  - Faire une redirection vers une page JSP qui va se charger de l'affichage de la des résultats relatif à la réponse http.

Le modèle MVC

Communication entre Servlet et JSP (1ere Solution)



Le modèle MVC

Communication entre Servlet et JSP (1ere Solution)

```
public class Test extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
…
}
public void doPost(HttpServletRequest request, HttpServletResponse response)  throws IOException, ServletException {
doGet(request, response);
}
// affichage formulaire vide
void doInit(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
…
}
void doInvite (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
…
}
// affichage formulaire pré-rempli
void doRetourFormulaire(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
…
}
void doValidationFormulaire(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException{
…
}
```


125

Le modèle MVC

Communication entre Servlet et JSP (1ere Solution)

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {

// on récupère l'action à exécuter
String action=request.getParameter("action");
if(action==null){
action="init";}
if(action.equals("init")){
doInit(request,response);
return;}

if(action.equals("invite")){
doInvite(request,response);
return;
}
if(action.equals("validationFormulaire")){
// validation du formulaire de saisie
doValidationFormulaire(request,response);
return;
}
doInit(request,response);}
```


126

Le modèle MVC

Communication entre Servlet et JSP (1ere Solution)

```
// affichage formulaire vide
void doInit(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException{
getServletContext().getRequestDispatcher("/formulaire.jsp").forward(request, response);
return;}


void doInvite(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException{
// on affiche le formulaire
getServletContext().getRequestDispatcher(« /invite.jsp").forward(request, response);
Return;}
```


127

Le modèle MVC

Communication entre Servlet et JSP (1ere Solution)

```
void doValidationFormulaire(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException{
String nom = request.getParameter("txtNom");
String age = request.getParameter("txtAge");
// qu'on mémorise dans la session
HttpSession session = request.getSession(true);
session.setAttribute("nom", nom);
session.setAttribute("age", age);
// vérification des paramètres
ArrayList<String> erreursAppel = new ArrayList<String>();
// le nom doit être non vide
nom = nom.trim();

getServletContext().getRequestDispatcher("/reponse.jsp").forward(request , response);
return;
}
```


128

Le modèle MVC

Communication entre Servlet et JSP (1ere Solution)

invité

```
<form action="test1" method="POST">
login <input type="text" name="login"> <br>
<input type="hidden" name="action"
value="validerformulaire"> <br>
mot de passe <input type="password" name="pwd">
<br>

<input type="submit" name="ok" value="ok"> <br>
</form>

<a href="test1?action=invite">se connecter en tant
qu'invité </a>
```


129

JavaBean

Interaction entre formulaire et action (ou pages JSP)

1. Création d'un bean
2. Création de liste d'objet dans la servlet
3. Ajout du bean à la liste
4. Ajouter la liste à la request

Bean Produit dans le package Model

```
ArrayList<Produit> liste = new ArrayList<Produit>();
Produit p = new Produit();
p.setDes("a");
p.setId(1);
liste.add(p);
Produit p1 = new Produit();
p1.setDes("a1");
p1.setId(2);
liste.add(p1);
request.setAttribute("liste", liste);
request.getRequestDispatcher("/ChercherJSTL.jsp").forward(request,response);
```


130

JavaBean

☞ Coté JSP: sans les JSTL

- Récupérer l'objet
- Parcours en utilisant la boucle foreach

```
<%@page import="java.util.ArrayList,model.*" %>
```


```
<%
ArrayList<Produit> list = (ArrayList<Produit>)
request.getAttribute("liste");
for(Produit a : list){
out.println(a.getDes());
out.println(a.getId());
}
%>
```


JavaBean

☞ Coté JSP: avec les JSTL

- Récupérer l'objet
- Parcours en utilisant la boucle foreach

```
<%@page import="java.util.ArrayList,model.*" %>
```


```
<c:forEach var="prod" items= "${requestScope.liste}">
<c:out value="${prod.id}" />
<c:out value="${prod.des}" />
</c:forEach>
```