DÉVELOPPEMENT WEB

JAVA ENTERPRISE EDITION – JEE

Mohammed Achkari Begdouri

Université Abdelmalek Essaadi

Faculté Polydisciplinaire à Larache - Département Informatique

achkari.prof@gmail.com

Année universitaire 2020/2021

Chapitre 4: les JavaBeans

DÉVELOPPEMENT WEB – JEE

SMI – S6

Approche générale avec les JSP

- Dans les pages JSP, il est toujours très difficile de lire le mélange à la fois de code HTML et de code Java.

- L'approche la plus judicieuse est d'utiliser une écriture plus proche du HTML en utilisant la syntaxe du XML tout en faisant référence, malgré tout, à des classes Java.

  Ex: <jsp:useBean id="beanName" class="package.Class" />

- Les JavaBeans permettent de composer une structure particulière sur ces classes respectant un schémas standard afin qu'ils puissent être utilisés dans les pages JSPs sans code Java.

JavaBeans ?

- Les JavaBeans sont des classes Java (POJO) qui suivent certaines conventions :
  - Doivent avoir un constructeur vide (zero argument)
  - Ne doivent pas avoir d'attributs publics
  - La valeur des attributs doit être manipulée à travers des méthodes getXxx et setXxx
    - Si une classe possède une méthode getTitle qui retourne une String, on dit que le bean possède une propriété String nommée title
    - Les propriétés Boolean utilisent isXxx à la place de getXxx

Utilisation basique des Beans

- jsp:useBean
  - Cet élément permet de rendre un JavaBean accessible dans la page JSP.

  Utilisation :

  ```
  <jsp:useBean id="beanName"  class="package.Class" />
  ```

- jsp:setProperty
  - Cet élément modifie une propriété d'un bean (i.e., appel d'une méthode setXxx).

  Utilisation :

  ```
  <jsp:setProperty
     name="beanName"   property="propertyName"   value="propertyValue" />
  ```

- jsp:getProperty
  - Cet élément lit et retourne la valeur d'une propriété d'un bean.

  Utilisation :

  ```
  <jsp:getProperty name="beanName" property="propertyName" />
  ```

Approche générale avec des JSP et les tags jsp:useBean …

- La page JSP instancie un bean
  - <jsp:useBean id= " Personne" class="…"/>
- On passe des données de la requête au bean
  - <jsp:setProperty name="Personne" property="prenom" value="…"/>
- On affiche des valeurs issues des données de la requête
  - <jsp:getProperty name="Personne" property="prenom"/>

Création de Beans: jsp:useBean

- Format:
  - <jsp:useBean id="name" class="package.Class" />
- But:
  - Permettre l'instanciation de classes Java sans programmation Java explicite (syntaxe compatible XML)
- Remarques:
  - Interpretation simple :

    <jsp:useBean id="book1" class="servlets.Book" />

    peut être vu comme équivalent au scriptlet :

    <% servlets.Book book1 = new servlets.Book(); %>

Modification de propriétés de Bean : jsp:setProperty

- Format
  - –<jsp:setProperty name="name" property="property" value="value" />
- But
  - – Permettre de modifier les propriétes d'un bean properties (i.e., appel de méthodes setXxx) sans programmation Java explicite
- Remarques
  - – <jsp:setProperty name="book1" property="title" value=" JEE " />
    est équivalent au scriptlet :
  - – <% book1.setTitle("JEE"); %>

Accès aux propriétés de Bean : jsp:getProperty

- Format
  - <jsp:getProperty name="name" property="property" />
- But
  - Permettre d'accéder propriétés d'un bean (i.e., appel de méthodes getXxx) sans programmation Java explicite
- Remarques
  - <jsp:getProperty name="book1" property="title" />

  est équivalent à l'expression JSP :
  - <%= book1.getTitle() %>

Exemple d'utilisation: étape 1

```
package beans;
    public class StringBean {

        private String message = "No message specified";

        public String getMessage() {
        return(message);
       }

        public void setMessage(String message) {
              this.message = message;
        }
   }
```

- Les Beans doivent toujours être dans des packages !

Exemple d'utilisation: étape 2

```
<jsp:useBean id="testBean" class="beans.StringBean" />


<jsp:setProperty name="testBean" property="message" value="autre message" />
<jsp:getProperty name="testBean" property="message" />

<% testBean.setMessage("Mon message favoris est bonjour"); %>
<%= testBean.getMessage()%>
```

jsp:setProperty: association propriété/paramètre

```
<jsp:useBean id = "utilisateur" class = "Personne" />
<jsp:setProperty name= "utilisateur" property = "prénom"
                                    value = '<%= request.getParameter("prénom") %>' />
```

<jsp:useBean id = "utilisateur" class = "Personne" />

<jsp:setProperty name = "utilisateur" property = "prénom" param = "prénom" />

<jsp:useBean id = "utilisateur" class = "Personne" />

<jsp:setProperty name = "utilisateur" property = "prénom" />

jsp:setProperty : associer toutes les propriétés aux paramètres

- Utilisation de "*" pour la valeur de l'attribut property de jsp:setProperty

  – On indique que la valeur doit provenir des paramètres de requête qui correspondent aux noms des propriétés

  – Particulièrement pratique pour réaliser des « Beans formulaires »

```
<jsp:useBean id= "monBean" class="beans.Personne" />
<jsp:setProperty name="monBean" property="*" />
```

Différents scopes pour jsp:useBean

- request
  - <jsp:useBean id="..." type="..." scope="request" />
- session
  - <jsp:useBean id="..." type="..." scope="session" />
- application
  - <jsp:useBean id="..." type="..." scope="application" />
- page
  - <jsp:useBean id="..." type="..." scope="page" />
  - Ce scope n'est pas utilisé dans MVC

Exemple : Partage de données sur requête

- Servlet

```
ValueObject value = new ValueObject (request.getParameter("ObjectID"));
request.setAttribute("key", value);
request.getRequestDispatcher("SomePage.jsp").forward(request, response);
```

- JSP 1.2

```
<jsp:useBean id="key" type="somePackage.ValueObject"  scope="request" />
<jsp:getProperty name="key" property="someProperty" />
<jsp:forward page=" SomePage.jsp"  />
```

- JSP 2.0

```
${key.someProperty}
```

Partage de données sur session

- Servlet

```
ValueObject value = new ValueObject(...);
HttpSession session = request.getSession();
session.setAttribute("key", value);
request.getRequestDispatcher("SomePage.jsp").forward(request, response);
```

- JSP 1.2

```
<jsp:useBean id="key" type="somePackage.ValueObject"  scope="session" />
<jsp:getProperty name="key" property="someProperty" />
<jsp:forward page=" SomePage.jsp"  />
```

- JSP 2.0

```
${key.someProperty}
```

Utilisation du JSTL (JSP Standard Tag Lib)

- La JSTL est une bibliothèque regroupant des balises implémentant des fonctionnalités communes aux applications web :
  - Mise en place de boucles
  - Tests conditionnels
  - Le formatage des données …

- Le but principal d'utilisation du JSTL, c'est d 'éviter d'écrire du code JAVA dans les pages JSPs :

  - Améliorer la lisibilité du code, et donc sa maintenabilité
    - Les pages JSPs ne contiennent que des balises compatibles XML
  - Réduire de la quantité du code à écrire
  - Respecter au mieux le découpage en couche recommandé par le modèle MVC

Configuration du JSTL

- Pour pouvoir utiliser les balises JSTL, il faut ajouter la bibliothèque suivante au classPath du projet :
  - http://download.java.net/maven/1/jstl/jars/jstl-1.2.jar

- Après il faut ajouter la directive taglib dans la page JSP :
  - <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

- Maintenant, les balises JSTL sont prêtes à être utiliser dans la page JSP:

```
<body>
    <c:out value="test" />
</body>
```

Exemple JSTL : Les boucles

- Une boucle avec une scriptlet Java

```
<%
        List<Integer> list = (ArrayList<Integer>)request.getAttribute("tirage");
        for(int i = 0; i < list.size();i++){
                out.println(list.get(i));
        }
 %>
```

- La même boucle avec des tags JSTL :

```
<c:forEach var="item" items="${tirage}" >
        <c:out value="${item}" />
</c:forEach>
```

Exemple JSTL : Les Conditions

- Une condition simple

```
<c:if test="${ 12 > 7 }" var="maVariable" scope="session">
        Ce test est vrai.
</c:if>
```

- Des conditions multiples

```
<c:choose>
        <c:when test="${expression}">Action ou texte.</c:when>
        ...
        <c:otherwise>Autre action ou texte.</c:otherwise>
</c:choose>
```

Il est temps de voir les Frameworks JEE …

- En Maitrisant les bases du J2SE et le paradigme du développement web avec JAVA (Servlet, JSP, Java Beans, …) :
  - Il est temps maintenant de commencer à voir le fonctionnement des différentes Frameworks JEE

- Il existe plusieurs Frameworks JEE, mais parmi les plus répandus on trouve :
  - JSF (Java Server Faces) : Pour gérer tous ce qui est vues, et Beans (Managed Beans)
  - Hibernate : Pour gérer la persistance des données et le mapping Objet-Relationnel
  - Spring : Pour gérer les sessions Hibernate, les transactions, l'injection des dépendances (CDI : Context and Dependency Injection), l'inversion de contrôle (IOC : Inversion of Control) sécurité, Web services, synchronisation avec JSF, …
    - Spring est un Framework à tout faire !!