MVC / Les Tags JSP et JSTL

A.-E. Ben Salem

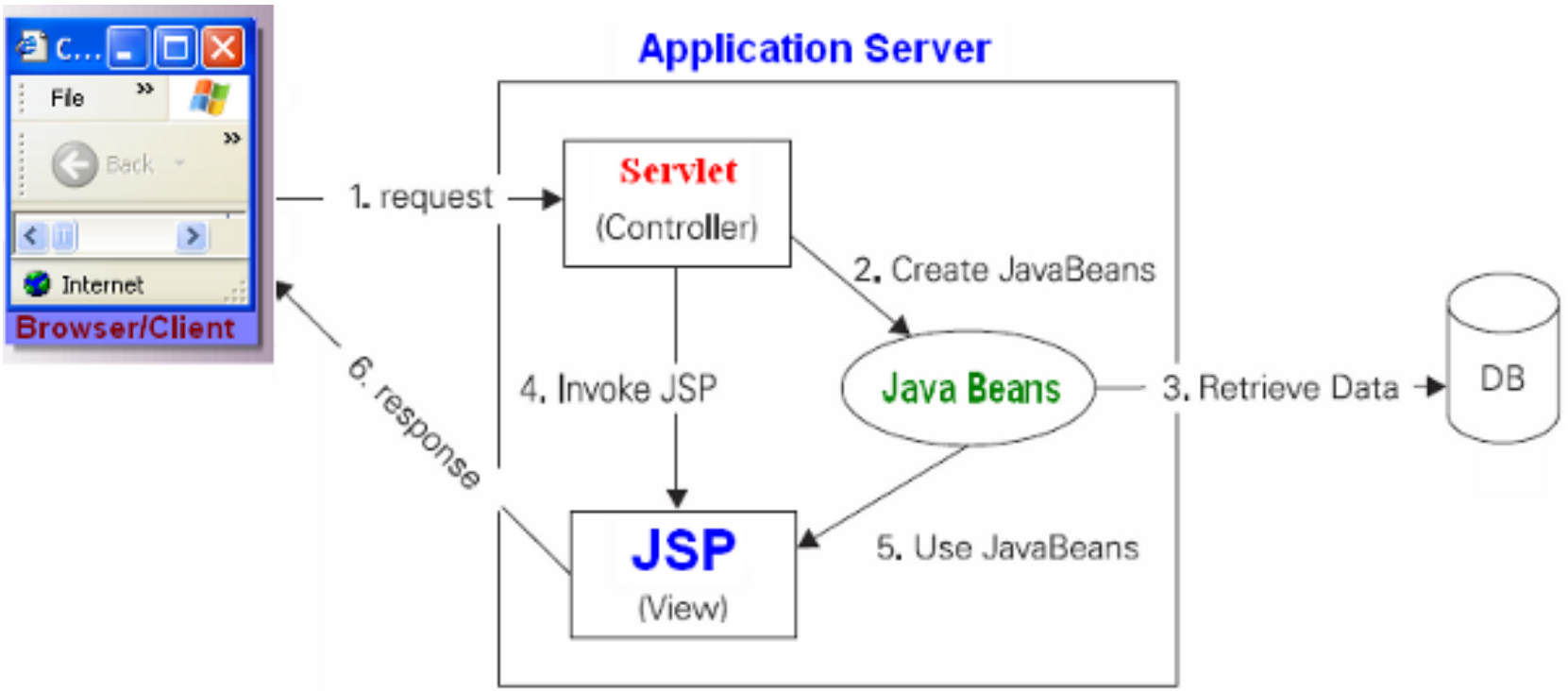
LRDE and LIP6

17 Octobre 2011

Plan

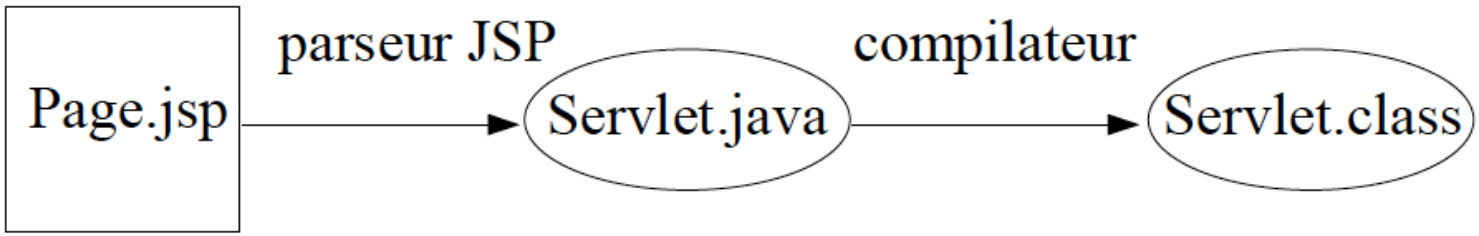
Architecture MVC

- Model–View–Controller (Modèle-vue-contrôleur)
- Séparation entre:
  - Le Controleur: Servlet qui aiguille les requête
  - La Vue: pages JSP pour l'affichage
  - Le Modèle: les classes ("Java Beans") qui traitent les données



Rappel JSP

- Une JSP mélange deux types de code:
  - code HTML (partie statique)
  - code Java (partie dynamique): exécuté coté serveur
- transformée dynamiquement en Servlet



- interaction avec des Java Beans.
- Une JSP est constituée :
  - de tags HTML
  - de scriptlets : code Java
  - de tags JSP : transformés en code Java

Scriplets

- Du code Java : <% code Java %>
- Des évaluations d'expression : <%= expression %>
- Des variables prédéfinies

```
<%@ page language="Java" %>
<html><head><title>First.jsp</title>
</head><body>
<h1>Nombres de 1 à 10</h1>
<% for(int i=1; i<=10; i++) { %>
        <%= i %> <br/>
<% } %>
</body>
</html>
```

Variables prédéfinies

HttpServletRequest request
HttpServletResponse response
HttpSession session
ServletContext application
PrintWriter out
Object page
ServletConfig config
javax.servlet.jsp.PageContext pageContext
Throwable exception

JSTL

Bibliohèques de balises (tags libraries)

La JSTL (JSP Standard Tag Library)

JSTL (JSP Standard Tag Library):

- Bibliothèques de tags (en plus des tags de base '<jsp: . . . >')
- Exemples de tags JSTL:
-
  - Tag d'itération <c:forEach>
  - Tag conditionelle <c:if>
  - Exécution de requête SQL <sql:query>,. . .
  - Format <fmt:formatDate>,. . .
- But: ne manipuler qu'un langage de balises dans les pages JSP
- Téléchargement à l'URL (JSTL 1.1): http://tomcat.apache.org/taglibs/standard/
- La documentation de la JSTL 1.1: http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/

Les 4 bibliothèques de tags de la JSTL

La JSTL est composée de 4 bibliothèques de tags :

Bibliothèque	Prèfixe	Exemples de tags	URI
core	<c:. . . >	<c:forEach>, <c:if>, <c:out>, <c:choose>, . . .	http://java.sun.com/jsp/jstl/core
SQL	<sql:. . . >	<sql:query>, <sql:update>, . . .	http://java.sun.com/jsp/jstl/sql
Format	<fmt:. . . >	<fmt:formatNumber>, <fmt:parseNumber>, . . .	http://java.sun.com/jsp/jstl/fmt
XML	<x:. . . >	<x:parse>, <x:transform>, . . .	http://java.sun.com/jsp/jstl/xml

Bibliothèque JSTL: core

Bibliothèque JSTL core:

- Bibliothèque des tags fournissant les fonctions de base
- Pour utiliser JSTL core, il faut la déclarer dans la JSP:
  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
- Les principales balises de la bibliothèque core associée au préfixe 'c' sont : c:out, c:catch, c:if, c:forEach :

```
<c:out value="expression" />

est équivalent à :

<%= expression %>
```

Bibliothèque JSTL core: Tag <c:if>

Tag <c:if test="...">:

- c'est le si(condition) en JSTL
- permet de réaliser un test conditionnel
- l'attribut "test" permet de décrire la condition
- Exemple:

```
<c:if test="${empty person.name}"> Inconnu </p>

est équivalent à :

<% if(person.getName() == null) { %>
<p>Inconnu </p>
<% } %>
```

Bibliothèque JSTL core: Tag <c:forEach>

Tag <c:forEach var="..." items="...">:

- permet de parcourir les différents éléments d'une collection
- l'attribut "items"= la collection à parcourir (scope: page, request, session,...)
- l'attribut "var"= nom de la variable qui contient l'élément en cours

```
<c:forEach var="person" items="${listPerson}">
<tr>
<td>${person.name}</td>
<td>${person.age}</td>
</tr>
</c:forEach>
```

est équivalent à :

```
<% for (Iterator it = listPerson.iterator(); it.hasNext();) {
out.println("<tr>");
Person person = (Person) it.next();
out.println("<td>" + person.getName() + "</td>");
out.println("<td>" + person.getAge() + "</td>");
out.println("</tr>");
} %>
```

Bibliothèque JSTL core: Tag <c:catch>

Tag <c:catch var="...">:

- permet de capturer des exceptions
- l'attribut "var"= nom de la variable qui va contenir l'exception

```
<c:catch var="erreur" />
actions à surveiller
</c:catch>
<c:if test="${not empty erreur}">
<c:out value="${erreur.message}"/>
</c:if>
```

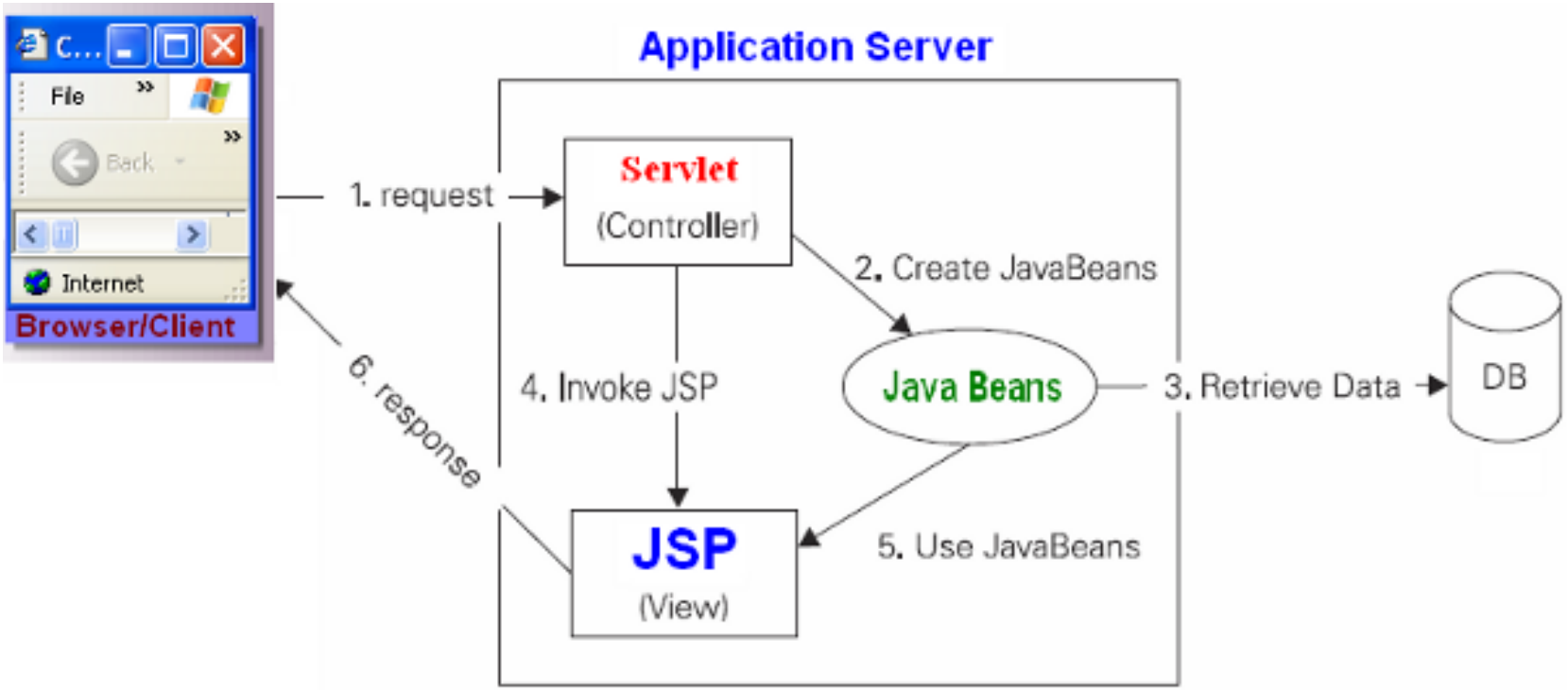
est équivalent à :

```
<% try {
actions á surveiller
} catch (Throwable erreur) {
out.println(erreur.getMessage());
}%>
```

Tags JSP pour gérer les Beans

- Tag <jsp:useBean>:

<jsp:useBean id="nomBean" scope="portée" class="Classe"/>



Tags JSP pour gérer les Beans

- Tag <jsp:useBean>:

<jsp:useBean id="nomBean" scope="portée" class="Classe"/>

- Exemple:

```
<jsp:useBean id="person" class="Person" scope="session/>

est équivalent à :

<% Person person = (Person) session.getAttribute("person");
if (person == null) {
person = new Person();
session.setAttribute("person", person); } %>
```

Tag <jsp:setProperty>

<jsp:setProperty name="nomBean" property="nomAttribut"/>

- Le tag <jsp:setProperty> s'utilise en complément de <jsp:useBean>
- permet de mettre à jour la valeur d'un attribut d'un Bean à partir d'un paramètre de la requête.
- il utilise le setter de l'attribut dans le Bean (méthode setXXX(...) où XXX est le nom de l'attribut avec la première lettre en majuscule)
- Exemple:

```
<jsp:useBean id="person" class="Person" scope="session/>
<jsp:setProperty name="person" property="name" />

est équivalent à :

<jsp:useBean id="person" class="Person" scope="session/>
<% if (request.getParameter("name") != null)
person.setName(request.getParameter("name")); %>
```

Tag <jsp:getProperty>

<jsp:getProperty name="nomBean" property="nomAttribut"/>

- Le tag <jsp:getProperty> permet d'afficher un attribut d'un Bean,
- il utilise le getter de l'attribut (méthode getXxx() où xxx est le nom de l'attribut avec la première lettre en majuscule)
- Exemple:

```
<jsp:useBean id="person" class="Person" scope="session/>
<jsp:getProperty name="person" property="name" />
```

est équivalent à :

```
<jsp:useBean id="person" class="Person" scope="session/>
<%=person.getName() %>
```

l'attribut scope du tag <jsp:useBean>

<jsp:useBean id="nomBean" class="Class" **scope**="page"/>

- L'attribut scope permet de définir la portée durant laquelle le bean est défini et utilisable
- La valeur de cet attribut détermine la manière dont le tag localise ou instancie le bean
- Les valeurs possibles du scope sont :
  - **page** : Le bean est utilisable dans toute la page JSP (valeur par défaut).
  - **request** : le bean est accessible durant la durée de vie de la requête. La méthode getAttribute() de l'objet request permet d'obtenir une référence sur le bean.
  - **session** : Le bean est utilisable tout au long de la session utilisateur. La JSP qui créé le bean doit avoir l'attribut session = « true » dans sa directive page.
  - **application** : le bean est utilisable par toutes les JSP de la même application. Le bean n'est instancié que lors du rechargement de l'application.

Tags d'action: <jsp:forward> et <jsp:include>

- **<jsp:forward page="page.jsp">**
  - permet de rediriger la requête vers une autre URL (ressource HTML, JSP ou Servlet)
  - Cette URL est relative à la JSP courante
  - Si l'URL commence par un "/", elle est absolue
  - Dès que le moteur de JSP rencontre ce tag, il ignore le reste de la JSP courante
  - il est possible de passer des paramètres vers la ressource appelée grâce au tag <jsp:param . . . />
- **<jsp:include page="page.jsp">**
  - permet d'inclure le contenu généré par une autre JSP ou Servlet
  - inclusion dynamique au moment où la JSP est exécutée.
  - passage de paramètres: <jsp:param . . . />

Tags de directives <%@ ... %>

- Les directives permettent de préciser des informations globales sur la page JSP
- Syntaxe : <%@ directive attribut="valeur" . . . %>
- 3 directives possibles:
  - **page** : informations relatives à la page (langage et import): **<%@ page import="java.util.*" %>**
  - **include** : inclure des fichiers statiques dans la JSP avant la génération de la Servlet,
    <%@ include file="chemin relatif du fichier" %>
  - **taglib** : permet de définir des tags personnalisés