Technologie Web

Les JavaServer Pages (JSP)

Alexandre Pauchet

INSA Rouen - Département ASI

BO.B.RC.18, pauchet@insa-rouen.fr

Plan

Introduction (1/5)
Description d'une JSP

JSP = JavaServer Page
Les JSP sont comparables aux scripts PHP
Langage de script côté serveur

Une page JSP contient :

- Du contenu statique (texte simple, HTML, XML, . . .)
- Du code JSP qui produit dynamiquement du contenu

Introduction (2/5)
Premier exemple

hello.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
<!DOCTYPE html>
<html>
  <head>
    <title>Ma première page JSP</title>
  </head>
  <body>
    <% String prenom=request.getParameter("prenom"); %>
    <h1>Bonjour <%=(prenom!=null && prenom.length()!=0)?prenom:"
        bel(le) inconnu(e)"%></h1>
    <% if (prenom!=null && prenom.equals("le monde")) { %>
        <h2>Bien joué !!!!</h2>
    <% } %>
    <form action="hello.jsp" method="post">
      <label>Prénom : </label><input type="text" name="prenom"
          size="30">
      <input type="submit" value="envoyer">
    </form>
  </body>
</html>
```

Introduction (3/5)
Premier exemple

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!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>Première page JSP</display-name>
  <description>
     Première page JSP
  </description>
</web-app>
```

Introduction (4/5)
Traduction, compilation, instanciation, ...

Une JSP est transformée en un HttpJspBase

Fonctionnement :

1. Requête sur une JSP
2. JBoss compare les dates de la JSP et de son HttpJspBase
   Si le HttpJspBase de la JSP est plus ancien que la page :
   - Traduction de la JSP en un HttpJspBase (dérivation)
   - Compilation du HttpJspBase (*cf.* répertoire `$JBOSS_HOME/standalone/tmp`)
3. Si le HttpJspBase de la page n'a pas encore été chargé
   - Création du HttpJspBase
   - Instanciation
   - Initialisation du HttpJspBase avec appel de la méthode `jspInit()`
4. Invocation de la méthode `_jspService()`
5. Appel de `jspDestroy()` lors du déchargement du HttpJspBase

Introduction (5/5)
Quelques éléments JSP

- <%@ ... %> : directives (validité : page)
- <%! ... %> : déclarations
- <% ... %> : scriptlet (*i.e.* script : tests, itération, . . .)
- <%= ... %> : récupération d'une valeur d'expression (*i.e.* évaluation)
- <jsp:include ... /> : inclusion à l'exécution
- <jsp:forward ... /> : délégation à un autre composant
- <jsp:useBean ... /> : utilisation d'un javabean

Fonctionnement (1/12)
Directives de page

Ces directives s'appliquent à la page

Syntaxe

```
<%@ page directive %>
```

Exemples

```
<\%@ page contentType="text/plain; charset=UTF-8" \%>
<\%@ page import="java.io.*, java.util.*" \%>
<\%@ page isThreadSafe="true" \%>
  // 1 seule requête à la fois entre le conteneur
  // et la JSP
```

Fonctionnement (2/12)
Paramètres d'exécution d'une page JSP

L'exécution d'une JSP est paramétrable via la directive page

- <%@ page buffer="none|xxxkb"%>
  Spécifie la taille du buffer où est renvoyée la réponse.
  Un petit buffer allège en charge le serveur d'application et envoie la réponse plus rapidement.
- <%@ page errorPage="file_name"%>
  Spécifie la page (JSP ou non) vers laquelle le serveur d'application renvoie lorsqu'une exception non gérée est lancée par la JSP.
  La directive <%@ page isErrorPage="true|false" %> permet de spécifier si la page de renvoi est une page d'erreur et lui autorise ainsi la transmission de l'exception pour un éventuel traitement.

Fonctionnement (3/12)
Redirection d'erreur vers une page statique

page-HTML.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page errorPage="erreur.html" %>
<!DOCTYPE html>
<html>
  <head><title>Ma première page jsp</title></head>
  <body><h1>1/0 = <%=1/0 %></h1></body>
</html>
```

erreur.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=
        UTF-8" />
    <title>Page d'erreur</title>
  </head>
  <body><h1>Il doit y avoir une erreur dans la JSP...</h1></body>
</html>
```

Fonctionnement (4/12)
Redirection d'erreur vers une JSP

page-JSP.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page errorPage="erreur.jsp" %>
<!DOCTYPE html>
<html>
  <head><title>Ma première page jsp</title></head>
  <body><h1>1/0 = <%=1/0 %></h1></body>
</html>
```

erreur.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page isErrorPage="true" %>
<!DOCTYPE html>
<html>
  <head><title>Page d'erreur</title></head>
  <body>
    <h1>Il doit y avoir une erreur dans la JSP...</h1>
    <p>Dans une JSP, on peut traiter l'exception : "<%= exception.getMessage() %>"</p>
  </body>
</html>
```

Fonctionnement (5/12)
Déclarations

Les déclarations permettent de définir des classes, des variables, ...
Elles seront utilisables dans toute la JSP

Syntaxe

```
<%!
  déclarations
%>
```

Ces déclarations sont placées dans la classe au moment de la traduction de la JSP en HttpJspBase
⇒ **les variables sont donc des attributs de la HttpJspBase**

Fonctionnement (6/12)
Initialisation/Finalisation d'une JSP

Paramétrage de l'initialisation et de la finalisation

- Initialisation lors du chargement par le serveur d'application : `public void jspInit()`
- Finalisation lors du déchargement par le serveur d'application : `public void jspDestroy()`

Rédéfinition dans les déclarations JSP

```
<%! public void jspInit() {
      ...
    }
    public void jspDestroy() {
      ...
    }
%>
```

Utilité : ouverture persistante d'une connexion à une BD, ...

Fonctionnement (7/12)
Scriptlets (scripts)

Utilisation des objets, variables, structures de contrôle, ...

Syntaxe

```
<%
  script
%>
```

Les scriptlets seront placés dans la méthode `_jspService()` qui exécute la requête (`GET`, `POST`, ...) effectuée sur la page JSP

Fonctionnement (8/12)
Expressions

Insertion de valeurs dans la page résultat

Syntaxe

```
<%= expression %>
```

L'expression doit retourner une valeur

Fonctionnement (9/12)
Exemple

compteur.jsp

```
<%@ page isThreadSafe="false" %>
<%@ page contentType="text/plain; charset=UTF-8" %>
<%@ page import="java.io.*" %>

<%!
int compteur=0;
String fichier="/tmp/compteur.txt";
public void jspInit() {
  try {
    FileReader file = new FileReader(fichier);
    BufferedReader reader = new BufferedReader(file);
    compteur = Integer.parseInt(reader.readLine());
    reader.close();
    }
    catch (FileNotFoundException exception) {}
    catch (IOException exception)           {}
    catch (NumberFormatException exception) {}
}

...
```

Fonctionnement (10/12)
Exemple

compteur.jsp (suite)

```
...
public void jspDestroy() {
  try {
    FileWriter file = new FileWriter(fichier);
    PrintWriter writer = new PrintWriter(file);
    writer.println(compteur);
    writer.close();
  }
  catch (IOException exception) {
    log("ErreurServlet : enregistrement de l'etat impossible pour
         la JSP compteur2.jsp");
    log(""+exception);
  }
}
%>

<% compteur+=1; %>

Le compteur : <%=compteur%>
```

Fonctionnement (11/12)
Objets disponibles

Objets directement disponibles dans une JSP

- **request** (`httpServlet`) : correspondant à la requête
- **response** (`httpServlet`) : correspondant à la réponse
- **session** : permet de gérer une session
- **out** : le flot de sortie de la réponse
- **application** : Servlet application ; contient, entre autres, la méthode `log()` pour écrire dans le fichier log
- **pageContext** : gestion du contexte et des attributs de la JSP
- ...

Fonctionnement (12/12)
Exemple

log.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html>
<% request.setCharacterEncoding("UTF-8");%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Ecriture dans le fichier log</title>
  </head>
  <body>
    <%
      String phrase = "";
      if(request.getParameter("phrase")!=null && request.getParameter("phrase").length
          ()!=0) {
        phrase = request.getParameter("phrase");
        application.log(phrase);
      }
    %>
    <form action="log.jsp" method="post" enctype="application/x-www-form-urlencoded">
      <label>Phrase a écrire dans le fichier log : </label><input type="text" name="
          phrase" size="30">
      <% out.println("<input type=\"submit\" value=\"envoyer\">"); %>
    </form>
  </body>
</html>
```

Inclusion/Délégation (1/6)

Partage du contrôle

- Appel d'une ressource à partir d'une autre ressource (Servlet ou JSP) afin de mandater à cette dernière l'affichage d'informations
- Deux types d'invocations sont possibles :
  - la délégation
  - l'inclusion
- Il y a transmission des objets requête (`ServletRequest`) et réponse (`ServletResponse`)
- Il est possible d'ajouter de nouveaux attributs à la requête

Inclusion/Délégation (2/6)

Inclusion

Inclusion d'une page statique ou dynamique dans une JSP

Inclusion à la traduction en HttpJspBase (= inclusion de texte)

```
<\%@ include file="FICHIER" \%>
```

Inclusion à l'exécution (= inclusion du résultat)

```
<jsp:include page="FICHIER" />
```

ou

```
<jsp:include page="FICHIER">
  <jsp:param name="PARAMETRE" value="VALEUR"/>
  \ldots
</jsp:include>
```

Inclusion/Délégation (3/6)

Exemple

inclusion.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html>
<html>
  <head><title>JSP avec inclusion</title></head>
  <body><%@ include file="titre.txt" %>
    <p><jsp:include page="texte.jsp" /></p>
    <p><jsp:include page="parametre.jsp">
          <jsp:param name="nom" value="Bob" />
       </jsp:include>
    </p>
  </body>
</html>
```

Inclusion/Délégation (4/6)

Exemple

titre.txt

```
<h1>Titre de la JSP, inclus a partir d'un fichier texte</h1>
```

texte.jsp

```
<%@ page contentType="text/plain; charset=UTF-8" %>
<% out.println("Texte genere par la JSP texte.jsp"); %>
```

parametre.jsp

```
<%@ page contentType="text/plain; charset=UTF-8" %>
<%
  String nom = request.getParameter("nom");
  out.println("Salut " + nom);
%>
```

Inclusion/Délégation (5/6)

Délégation

Délégation à un autre composant web

Syntaxe

```
<jsp:forward page="fichier" />
```

ou

```
<jsp:forward page="fichier" >
  <jsp:param name="nom" value="valeur"/>
  ...
</jsp:forward>
```

Inclusion/Délégation (6/6)

Exemple

delegation.jsp

```
<html>
  <head><title>JSP avec délégation</title></head>
  <body>
    <jsp:forward page="parametre.jsp">
      <jsp:param name="nom" value="Bob" />
    </jsp:forward>
    <p>Ceci ne peut pas être affiché vu qu'on a délégué l'affichage !</p>
  </body>
</html>
```

parametre.jsp

```
<html>
  <head><title>Affichage delegue</title></head>
  <body>
    <%
      String nom = request.getParameter("nom");
      out.println("<p>Salut " + nom + " !</p>");
    %>
  </body>
</html>
```

JSP&JavaBeans (1/8)
Définition

Définition (JavaBean) :

Classe java respectant certaines conventions (API) faisant d'elle un composant java réutilisable et facilement intégrable dans des applications.

Les JSP offrent des tags pour interagir avec un JavaBean

JSP&JavaBeans (2/8)
Contraintes de conception

Pour être un JavaBean, une classe doit :

- Être `Serializable`,
- Posséder un constructeur sans argument,
- Avoir des accesseurs sur ses attributs privés respectant des conventions de nommage ; ex. pour un attribut `String name` :
  ⇒ `public void setName(String)`
  ⇒ `public String getName()`
- Ne pas être déclarée `final`
- Contenir les méthodes d'interception d'événements nécessaires.

JSP&JavaBeans (3/8)

Utilisation dans les JSP

- `<jsp:useBean id="nom"class="classbean"scope="portée"/>`

  avec `portée = page (par défaut)/ request / session / application`
- `<jsp:setProperty name="nom" property="propriété" value="valeur"/>`

  avec `valeur = String / <%= ... %>}`

  identique à : `<% nom.setPropriété(valeur)%>`
- `<jsp:setProperty name="nom" property="propriété" param="parametre_requete"/>`

  si `parametre_requete` est absent, `param=property` ;

  si `parametre_requete=*`, règle toutes les propriétés ayant des noms identiques aux paramètres de requête
- Définition des attributs à la déclaration :

```
<jsp:useBean id="nom" class="classbean" scope="portée" >
  <jsp:setProperty name="nom" property="propriété" .../>
  ...
</jsp:useBean>
```

- `<jsp:getProperty name="nom" property="propriété"/>` identique à :

  `<%=nom.getPropriété()%>`

JSP&JavaBeans (4/8)
Exemple

Personne.java

```
package asi;
import java.io.*;

public class Personne implements Serializable {
  private String nom;
  private String prenom;

  public Personne() { this.nom = "Leponge"; this.prenom = "Bob";
      }

  public String getNom() { return this.nom; }

  public void setNom(String nom) { this.nom=nom; }

  public String getPrenom() { return this.prenom; }

  public void setPrenom(String prenom) { this.prenom=prenom; }
}
```

JSP&JavaBeans (5/8)
Exemple

javabean.jsp

```
<%@ page contentType="text/html" %>
<%@ page language="java" import="asi.*" %>

<jsp:useBean id="bob" class="asi.Personne" scope="page" />

<!DOCTYPE html>
<html>
  <head>
    <title>Un Hello Bob</title>
  </head>
  <body>
    <p>Salut <%= bob.getPrenom() %> <jsp:getProperty name="bob"
        property="nom"/> !</p>
  </body>
</html>
```

JSP&JavaBeans (6/8)

JSP, javabeans et formulaires

Mapping requête entrante / javabean

- Le parser JSP permet le mapping entre la requête entrante (ensemble des paramètres) et un javabean (propriétés)
  <jsp:setProperty name="aliasBean" property="*"/>
- Utilisation de l'**introspection** (exposition par une classe de ses propriétés par méthodes d'accession et de modification)
- La correspondance peut être transgressée par un mapping manuel
  <jsp:setProperty name="aliasBean" param="nomAttribut" property="beanProp"/>"

JSP&JavaBeans (7/8)

JSP, javabeans et formulaires

Personne.java

```
package asi;
import java.io.*;

public class Personne implements Serializable {
  private String nom;
  public Personne() { this.nom = ""; }
  public String getNom() { return this.nom; }
  public void setNom(String nom) { this.nom=nom; }
}
```

JSP&JavaBeans (8/8)

JSP, javabeans et formulaires

hello.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page language="java" import="asi.*" %>

<!DOCTYPE html>
<html>
  <head><title>Un autre Hello World !</title></head>
  <body>
    <form action="hello.jsp" method="post">
      <label>Nom : </label><input type="text" name="nom" size="30
          ">
      <input type="submit" value="envoyer">
    </form>

    <jsp:useBean id="personne" class="asi.Personne" scope="page">
      <jsp:setProperty name="personne" property="*"/>
    </jsp:useBean>
    <p>Salut <%= personne.getNom() %> !</p>
  </body>
</html>
```

JSP&JavaBeans (1/6)
Exemple : Livre d'or en JSP

UnMessage.java

```
package libMessage;

import java.io.*;
import java.text.*;
import java.util.*;

public class UnMessage implements Serializable {
  private static SimpleDateFormat formatter = new SimpleDateFormat("E d MMM yyyy, H:m:s
       ", Locale.FRANCE);
  private Date date = new Date();
  private String email;
  private String texte;

  public UnMessage() { date.setTime(System.currentTimeMillis()); }

  public String getDate() { return formatter.format(date); }

  public void setEmail(String email) { this.email=email; }

  public String getEmail() { return email; }

  public void setTexte(String texte) { this.texte=texte; }

  public String getTexte() { return texte; }
}
```

JSP&JavaBeans (2/6)
Exemple : Livre d'or en JSP

LivreOr.java

```
package libMessage;

import java.util.*;
import java.io.*;

public class LivreOr implements Serializable {

  private ArrayList messages;
  private String nomFichier;

  public LivreOr() { }

  public String getFichier() {
    return this.nomFichier;
  }

  public void setFichier(String fichier) throws FileNotFoundException,IOException,
      ClassNotFoundException {
    this.nomFichier = fichier;
    FileInputStream fis;
    ObjectInputStream ois = null;
    try {
      fis = new FileInputStream(nomFichier);
      ois = new ObjectInputStream(fis);
      messages = (ArrayList) ois.readObject();
    }


...
```

JSP&JavaBeans (3/6)
Exemple : Livre d'or en JSP

LivreOr.java (suite)

```
...
    catch (Exception e) {
      messages = new ArrayList();
    }
  }

  public ArrayList getMessages() {
    return messages;
  }

  public void setMessages(ArrayList messages) {
    this.messages =  messages;
  }

  public void addMessage(UnMessage msg) {
    messages.add(msg);
  }

  public void enregistrer() throws FileNotFoundException, IOException{
    FileOutputStream fos = new FileOutputStream(nomFichier);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(messages);
  }
}
```

JSP&JavaBeans (4/6)
Exemple : Livre d'or en JSP

livredor.jsp

```
<%@ page contentType="text/html" %>
<%@ page language="java" import="libMessage.*" %>

<!DOCTYPE html>
<html>
  <head>
    <title>Livre d'or</title>
  </head>
  <body>
    <form action='livredor.jsp' method='POST'>
      <label>email :</label>
      <input type='text' name='email' size='50'/><br/>
      <textarea name='texte' rows='10' cols='80'>Saisissez votre message ici</textarea
          ><br/>
      <input type='submit' name='submit' value='Envoyer'/>
    </form>

<jsp:useBean id="msg" class="libMessage.UnMessage" scope="page" >
  <jsp:setProperty name="msg" property="*"/>
</jsp:useBean>

<jsp:useBean id="livreor" class="libMessage.LivreOr" scope="page"/>
<jsp:setProperty name="livreor" property="fichier" value="/tmp/messages.txt"/>


...
```

JSP&JavaBeans (5/6)
Exemple : Livre d'or en JSP

livredor.jsp (suite)

```
...
<%
  if(msg!=null && msg.getEmail()!=null && !msg.getEmail().equals("")) {
    livreor.addMessage(msg);
    livreor.enregistrer();
  }

  if(livreor!=null && livreor.getMessages()!=null && !livreor.getMessages().isEmpty())
      {
    for(Object lemsg : livreor.getMessages()) {
%>
        <table border="1pt">
        <tr>
          <td><%= ((UnMessage)lemsg).getEmail() %></td>
          <td><%= ((UnMessage)lemsg).getDate() %></td>
        </tr>
        <tr>
          <td colspan="2">
            <pre>
<%= ((UnMessage)lemsg).getTexte() %>
            </pre>
          </td>
        </tr>
        </table>
<% }} %>

  </body>
</html>
```

JSP&JavaBeans (6/6)

Exemple : Livre d'or en JSP

Déploiement

```
LivredorJSP
  |_ WEB-INF
  |    |_ web.xml
  |    |_ classes
  |         |_ libMessage
  |              |_ LivreOr.class
  |              |_ UnMessage.class
  |_ livredor.jsp
```

Standard Tag Library

Les JSP fournissent une bibliothèque de tags standards répondant à des besoins de base

Quelques domaines :

- Core : tags permettant des fonctionnalités de base
- XML : tags permettant de manipuler des données XML
- I18n : tags permettant de traiter l'internationalisation
- Database : tags permettant d'effectuer des requêtes SQL (travail normalement dédié aux JavaBeans)

Il est possible d'étendre le langage en créant d'autre tags