Introduction à JSP (Java Server Pages 1.x)

Code: java-jsp

Originaux

url: http://tecfa.unige.ch/guides/tie/html/java-JSP/java-jsp.html

url: http://tecfa.unige.ch/guides/tie/pdf/files/java-jsp.pdf

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 0.1 (modifié le 12/6/00 par VS)

Prérequis

- Java de base

  Module technique précédent: java-intro
- Avoir une idée du standard "CGI"

  Module technique précédent: cgi-intro

- Pages actives avec PHP (ou équivalent), conseillé mais pas obligatoire.

  Module technique précédent: php-html

Autres modules

Module technique suppl.: java-jhtml (anciens Java Server Pages 0.9, déconseillé)

Module technique suivant: java-servl

Documentation

url: http://java.sun.com/products/jsp/syntax.html (Sun JSP Syntax cards)

- imprimez une carte pour accompagner ces slides

url: Doc locale: http://tecfa.unige.ch/guides/jsp/pointers.html

url: http://java.sun.com/products/jsp/

Objectifs

- Faire des simples pages actives avec JSP
- (à faire: utilisation de "Beans")

Remerciements:

- Yan Bodain, CRIM Formation (correction d'une erreur sur request.getParameter)

1. Table des matières détaillée

2. Introduction à JSP (Java Server Pages)

Principe de base:

- Permet d'écrire des pages hybrides HTML / Java (comme PHP)
- XML tags et scripts en Java:
- Compile et exécute la source comme "servlet"
- La traduction se fait automatiquement pour tous les fichiers *.jsp
  - le résultat (servlet source et classe) est placée dans un répertoire du système
  - Après chaque update d'un fichier *.jsp la classe servlet est recompilée

Utilité:

- Création de pages "design"
- Analyse de formulaires
- possibilité d'interfaçage avec des "vrais" servlets
- accès à toute classe Java mise à disposition (notamment des "beans")

2.1 Ecrire des pages JSP simples

- pas besoin d'importer les classes (c'est fait par le serveur!)
- Le code Java est délimité dans une page HTML par des tags spéciaux
  - <% ..... >
  - voir exemple 2-2 "JSP simple (2)" [6]

Exemple 2-1: JSP simple (1)

url: Programme: http://tecfa.unige.ch/guides/jsp/ex/demo1.jsp
url: Source: http://tecfa.unige.ch/guides/jsp/ex/demo1.jsp.text

```
<BODY>
  <H1>JSP Test</H1>
   <ul>
    <%
    for (int i = 0; i < 5; i++) out.println ("<li>" + i);
    %>
   </ul>
</BODY>
```

- `out` est un objet "gratuitement" mis à votre disposition pour les "sorties"
- Curieux: voir -6. "JSP inside" [19] pour le Java que cette page génère

2.2 Faire une soupe HTML/Java plus mélangée

Exemple 2-2: JSP simple (2)

url: Programme: http://tecfa.unige.ch/guides/jsp/ex/demo2.jsp
url: Source: http://tecfa.unige.ch/guides/jsp/ex/demo2.jsp.text

Dans les sections Java, il faut quoter les " qui doivent apparaître dans HTML (\")

```
int emphasize = 5;
for (int i = 0; i < size; i++) {
  out.println ("<li>");
  if (i == emphasize) out.println ("<b>");
  out.println(" \"item" + i + ".jsp\" - item number " + i);
  if (i == emphasize) out.println ("</b>");
}
```

On peut mélanger variables Java et strings HTML

- utiliser le tag <%= .... > (expressions Java)

```
<% int font_size = 20; String color = "red"; %>
......
<font color="<%= color %>" size="<%= font_size %>"> Youppie ! </font>
......
```

2.3 Gestion des erreurs

url: http://tecfa.unige.ch/guides/jsp/ex/error.jsp.text (Source page gestion)

url: http://tecfa.unige.ch/guides/jsp/ex/err-demo.jsp (Page avec Erreur)

url: http://tecfa.unige.ch/guides/jsp/ex/err-demo.jsp.text (Source)

- On peut arranger un peu l'affichage des erreurs (exceptions) run-time
- Rien à faire pour la compilation :(

Page error.jsp

```
<%@ page isErrorPage="true"  %>
The name of the exception was:<%= exception.toString() %>
The message of the exception was: <%= exception.getMessage() %>
The stack trace was:<br>
    <%
      java.io.PrintWriter outstream = new java.io.PrintWriter(out);
     exception.printStackTrace(outstream); %>
```

- Attention à la directive "isErrorPage="true" !

Page jsp normale:

```
<%@ page errorPage="error.jsp" %>
```

- Cette directive envoye l'exception vers la page error.jsp (URL relatif !!)

3. Utilisation d'autres classes JAVA

- Vous pouvez importer toutes les classes Java avec la "page directive":

```
<%@ page
import="java.util.Date"
%>
```

Exemple 3-1: Dates avec JSP

url: Programme: http://tecfa.unige.ch/guides/jsp/ex/weekday.jsp
url: Source: http://tecfa.unige.ch/guides/jsp/ex/weekday.jsp.text

```
<%@ page
import="java.util.Date"
%>

<html><head><title>Today</title></head><body>
<h1>Today</h1>
<%
// Maps day number to a name
String days [] = { "Dimanche", "Lundi", "Mardi", "Mercredi",
                   "Jeudi", "Vendredi", "Samedi" };
// Get today's date
Date today = new Date ();
int weekday = today.getDay ();
out.println ("<p>On est " + days [weekday] + " aujourd'hui!");
%>
</body></html>
```

4. Traitement de formulaires

4.1 Principe de base du traitement de requêtes

- Les pages JSP peuvent utiliser les fonctionalités de la classe javax.servlet (puisqu'elles sont traduites en un servlet).
- On a gratuitement à disposition certains objets sous forme de variables, par ex.

1. `request` de la classe javax.servlet.http.HttpServletRequest
2. `out` de la classe javax.servlet.ServletOutputStream
3. `response` de la classe javax.servlet.http.HttpServletResponse

- Pour récupérer une variable "formulaire" utiliser:

```
request.getParameter("paramètre")
```

  - **`paramètre`** est le nom du paramètre qui vient des GET, POST, DELETE et PUT.
  - **`request`** possède d'autres méthodes utiles, par exemple getHeader.
    On peut lire un cookie avec request.getHeader("cookie")

Voir la servlet doc, par exemple:

url: http://tecfa2.unige.ch/guides/java/jsdk2/doc/

Exemple 4-1: Simple GET Demo.

url: Programme: http://tecfa.unige.ch/guides/jsp/ex/request1.jsp
url: Avec paramètre: http://tecfa.unige.ch/guides/jsp/ex/request1.jsp?MESSAGE=Hello
url: Source: http://tecfa.unige.ch/guides/jsp/ex/request1.jsp.text

```
<html>
<body>
<%
String message;
     if ((message = request.getParameter("MESSAGE")) == null) {
      out.print("No message query argument supplied.");
      out.print("Please use an URL like request1.jsp?MESSAGE=Hello");
     } else
     out.print("Message is " + message);
%>
</body> </html>
```

4.2 Traitement simple de formulaires

Exemple 4-2: Simple calcul (Formulaire + JSP)

url: http://tecfa.unige.ch/guides/jsp/ex/simple-calcul/formulaire.html
url: source jsp: /simple-calcul/calcul.jsp.text

Formulaire HTML:

```
<form action="calcul.jsp" method=post>
Quelles sont vos connaissances de HTML ?
<input type="radio" name="choice" value="1" checked>faibles
<input type="radio" name="choice" value="2">moyennes
<input type="radio" name="choice" value="3">bonnes
<br>
Indiquez votre expertise en programmation:
<input type="radio" name="choice2" value="1" checked>absente
<input type="radio" name="choice2" value="2">moyenne
<input type="radio" name="choice2" value="3">bonne
<P>
<input type="submit" value="Voir le résultat!">
</form>
```

JAVA:

```
<%
// Parameters come as strings
String choice = request.getParameter("choice");
String choice2 = request.getParameter("choice2");
String nom = request.getParameter("nom");

out.println(nom + ", votre input était: question a=" + choice + " ,question b="
+ choice2);

// Integer.parseInt() translates a string to an Integer
int score = Integer.parseInt(choice) + Integer.parseInt(choice2);

out.println("<h3>Votre score est de " + score + "</h3>");

if (score < 3) {
  out.print ("<p>Vous êtes un débutant</p>");
} else if (score < 5) {
  out.print ("<p>Vous avez un niveau moyen</p>");
} else {
  out.print ("<p>Vous êtes un expert !</p>");
}

%>
```

Explications

- la méthode **`request.getParameter`** retourne la valeur d'une variable
  - sous forme de string, il faut donc déclarer **`choice`** comme String
- la méthode **`Integer.parseInt("string")`** traduit un string en entier
  - nécessaire pour faire une addition dans l'exemple ci-dessus.

Problèmes avec cette page

- Lorsque l'on appelle directement la page *.jsp elle saute

  url: http://tecfa.unige.ch/guides/jsp/ex/simple-calcul/calcul.jsp
  - car elle essaye d'additionner des variables qui n'existent pas
  - voir 4.4 "Vérification de l'input et HTML conditionnel" [15]
- Pour les tricheurs, un emploi "GET" de la page:
  http://tecfa.unige.ch/guides/jsp/ex/simple-calcul/calcul.jsp?choice=10&choice2=15

4.3 Debugging de paramètres

- Alternative: utiliser "GET" comme methode dans le formulaire HTML

  `<form action="calcul.jsp" method="get">`

Exemple 4-3: Lister paramètres et valeurs reçus

url: http://tecfa.unige.ch/guides/jsp/ex/simple-calcul/formulaire2.html

url: source jsp: /simple-calcul/calcul2.jsp.text

```
<%@ page import="java.util.*" %>
<%
Enumeration pNameList; // contient une liste des params transmises
String pName;          // contient un de ces noms
String pVals [];       // contient l'array des valeurs pour chaque param
String val;            // 1 valeur
out.println("<p>Method used was: " + request.getMethod());
out.println("<p>Liste de paramètres et valeurs (pas forcément dans l'ordre):
<ol>");
for (pNameList = request.getParameterNames();pNameList.hasMoreElements();){
    pName = (String) pNameList.nextElement();
    out.println("<li>name=" + pName + ": ");
    pVals = request.getParameterValues(pName);
    if (pVals != null) {
     for (int i=0; i<pVals.length; i++)out.print ("val=" + pVals[i] + " ");
    }   }
out.println("</ol>");
%>
```

4.4 Vérification de l'input et HTML conditionnel

Voir si un paramètre est vide:

1. Champs texte
   - on regarde s'ils sont vides (string de taille zéro)
   - un input type "text" (HTML) est donc toujours passé au serveur !!

```
// input
String nomString = req.getParameter("nom");
.....
if  ( nomString.equals("") ) { ... geuler ...}
```

   - Voici une autre méthode sans utiliser emptyString (JDK < 1.2 ?)

```
if ( nomString.compareTo("") == 0)
```

2. Widgets de type "radio" etc.
   - on regarde si l'objet est présent ou absent (le formulaire ne transmet rien)
   - si "choix" n'a pas été transmis, getParameter retourne `null` !

```
String Choix = req.getParameter("choix");
....
if (Choix == null) { .... geuler .... }
```

HTML conditionnel

- On peut afficher conditionellement du HTML
- Attention: si problèmes, utilise out.print(" ...") à la place

Exemple 4-4: Un simple test avec JSP (vérification et calcul des résultats)

url: http://tecfa.unige.ch/guides/jsp/ex/simple-calcul/form-test.html

url: source jsp: /simple-calcul/calcul-test.jsp.text

```
if (request.getMethod().equals("GET")) {
    out.println("sorry you can't"); out.print("</body></html>");
    return;
}
// Parameters come as strings
choice = request.getParameter("choice");
choice2 = request.getParameter("choice2");
nom = request.getParameter("nom");

if (nom.equals("")) {
    %>
<p>Sorry on ne répond pas aux anons, <a href="form-test.html">refaire SVP</a>
</body></html>
     <%
     return;
}

if ((choice == null) || (choice2 == null)) {
    %>
<p>Utilisez la touche BACK pour compléter .... vous avez oublié qc
</body></html>
<%
     return;}
```

Exemple 4-5: Simple calcul en une seule page JSP

url: http://tecfa.unige.ch/guides/jsp/ex/simple-calcul/formulaire2.jsp
url: http://tecfa.unige.ch/guides/jsp/ex/simple-calcul/formulaire2.jsp.text

- L' exemple 4-2 "Simple calcul (Formulaire + JSP)" [11] en une page...

- En règle générale il semble plus simple d'utiliser 2 pages: une pour le formulaire et une autre pour le traitement.

```
if ((process != null) || (process2 !=null)) {
   ... montrer les résultats
if (process2 != null) {
   ... afficher un merci
} else {
    ... dire que l'on peut essayer de nouveau
}
if (process2 == null) {
%>
   ... afficher le formulaire
<input type="submit" name="process" value="Voir le résultat!">
<input type="submit" name="process2" value="Voir le résultat et finir!">
<% } %>
```

- Note: Pour filter des "get" directs sur la page, on peut aussi tester la méthode, voir 4.3 "Debugging de paramètres" [14]

5. Session tracking

Exemple 5-1: Session tracking I

url: http://tecfa.unige.ch/guides/jsp/ex/session/session-track.jsp
url: http://tecfa.unige.ch/guides/jsp/ex/session/session-track.jsp.text

```
<%@ page session="true" %>

<%
// Get the session data value
Integer ival = (Integer) session.getValue ("sessiontest.counter");
if (ival == null) ival = new Integer (1);
else ival = new Integer (ival.intValue() + 1);
session.putValue ("sessiontest.counter", ival);
%>
    You have hit this page <%= ival %> times.<br>
<%
    out.println("Your Session ID is " + session.getId() +  "<br>");
%>
<p>Click here to see the <a href="session-track.jsp.text">source</a>
</body></html>
```

Exemple 5-2: Session tracking II

url: http://tecfa.unige.ch/guides/jsp/ex/session/session-track2.jsp
url: http://tecfa.unige.ch/guides/jsp/ex/session/session-track2.jsp.text

```
...PAS FINI tout cela .... à suivre !
```

6. JSP inside

- Section à option (à ce stade)

  Il n'est pas nécessaire de comprende le code java produit pour utiliser les pages JSP. Par contre il est utile de savoir lire une telle page approximativement pour comprendre des messages d'erreur

Mécanisme

- Le server Java met les fichiers *.java et *.class générés
  dans le répertoire `tecfa:/usr/local/jakarta/tomcat/work/localhost_8080/`
- Pour chaque page JSP, un "servlet" est créé, par exemple:
  - `_0002fguides_0002fjsp_0002fex_0002fdemo_00031_0002ejspdemo1_jsp_0.java`
  - il s'agit d'une sous-classe de HttpServlet
- Une seule méthode "service" est crée
  - elle incorpore les instructions java dans la page JSP
  - elle "copie" plus ou moins "tel quel" les commandes html (voir méthode writeBytes)
- Les classes *.webserver.pagecompile assurent la mise à jour de la classe dès que le code source *.jsp change.

Sécurité

- Les servlets générés par JSP sont "sandboxed":
  - pas d'accès aux fichiers et autres ressources locales
  - (Sauf si je me trompe) il faut soit écrire des vrais servlets pour cela.

Exemple 6-1: Java généré pour l'exemple 2-1 "JSP simple (1)" [5]

```
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public class jsp__guides__jsp__ex__demo1_2ejsp extends org.gjt.jsp.HttpJspPageImpl{

   public final long _gnujspGetTimestamp() {
      return 948897919123L;
   }
   private static final String[] _gnujspDeps = new String[] { "/guides/jsp/ex/demo1.jsp" };
   public final String[] _gnujspGetDeps() {
      return _gnujspDeps;
   }
   public final long _gnujspGetCompilerVersion() {
      return 1999101701L;
   }
   public void _jspService (HttpServletRequest  request,
                            HttpServletResponse response)
       throws ServletException, IOException
   {
       response.setContentType ("text/html");
       JspFactory  factory     = JspFactory.getDefaultFactory ();
       PageContext pageContext = factory.getPageContext (this,request,response,
```

```
                                null,true, 8192, true);
      HttpSession session   = pageContext.getSession ();
      ServletContext application = pageContext.getServletConfig().getServletContext();
      JspWriter   out       = pageContext.getOut ();
      ServletConfig config  = pageContext.getServletConfig();
      Object      page      = this;

      try {

// line:/guides/jsp/ex/demo1.jsp:1
out.print ("<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 3.2 Final//EN'>\n<HTML>
\n  <HEAD>\n    <TITLE>JSP Test ( 1-Dec-1998)</TITLE>\n    <!-- Created by: D
.K.S.,  1-Dec-1998 -->\n    <!-- Changed by: D.K.S.,  1-Dec-1998 -->\n\n\n  <
/HEAD>\n  <BODY>\n    <H1>JSP Test</H1>\n\n\n<ul>\n      ");
// line:/guides/jsp/ex/demo1.jsp:16

     for (int i = 0; i < 5; i++) out.println ("<li>" + i);

// line:/guides/jsp/ex/demo1.jsp:18
out.print ("\n</ul>\n\n\n    <ADDRESS>\n      <A NAME='Signature'\n HREF=
'http://tecfa.unige.ch/tecfa-people/schneider.html'>D.K.S.</A>\n    </ADDRE
SS>\n  </BODY>\n</HTML>\n");
       } catch (Exception e) {
            out.clearBuffer ();
            pageContext.handlePageException (e);
       } finally {
            out.close ();
            factory.releasePageContext (pageContext);
       }

   }

}
```