Web Tier : Java Server Pages JSP

Plan

La technologie Java Server Pages : JSP

Principales caractéristiques

- objectif : créer facilement du contenu web ayant à la fois des composants statiques et des composants dynamiques.
- capacités : identiques aux servlets, mais avec une approche plus naturelle pour le contenu statique.
- Un langage : documents texte qui décrivent comment traiter la requête et construire la réponse.
- Un langage pour accéder aux objets Java côté serveur.
- un mécanisme pour définir des extensions du langage JSP : permet de créer des pages sans connaissance du langage Java

Principe d'une page JSP

Généralités

- Une page JSP est un document texte qui contient 2 types de textes : du statique (HTML, SVG, WML, and XML) et des éléments JSP qui construisent du contenu dynamiquement.
- L'extension est jsp. La page peut être composée d'autres fichiers (d'autres jsp complètes ou des fragments de jsp)
- Pour les fragments l'extension recommandée est .jspf.

Contenu d'une page JSP

différents types de contenu

- données statiques : *e.g.* code HTML, XHTML, etc. . .
- des directives de compilation <%@ *directive* %> :
  -**include** : <%@ include file="file" %>
  -**page** : *e.g.* <%@ page import="java.util.*" %>
  -**taglib** : <%@ taglib uri="http ://java.sun.com/jsp/jstl/core" prefix="c" %>
- script en Java entouré par **<%** *code java* **%>** :
  - + **!** : déclaration d'attributs et de méthodes : <% ! int attribut = 0 ;%>
  - insertion de blocs de code java : <% *bloc java* %> (scriptlet)
  - + **=** : out.println sur une expression : <%= *appel Java* %>
- actions JSP : balises XML **<jsp : . . .>** appelant des fonctions serveur
  - <jsp :forward page="otherJSP.jsp" >
  - <jsp :param name="key" value="value" />
- balises personnalisés créées grâce à une *taglib*

Affichage d'un appel java simple

date.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>date page</title>
</head> <body>
Hello! The time is now <%= new java.util.Date() %>
</body> </html>
```

Affichage de la date

Hello! The time is now Mon Mar 21 19:48:13 CET 2022

Bloc java (scriptlet)

date2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html> <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>date page</title>
</head> <body>
<%
// This is a scriptlet. Notice that the "date" variable we declare
//here is available in the embedded expression later on.
System.out.println( "Evaluating date now" );
java.util.Date date = new java.util.Date();
%>
Hello! The time is now <%= date %>
</body> </html>
```

Une JSP contient des objets prédéfinis

- **out** : JSPWriter pour écrire dans la réponse HTTP.
- **page** : this
- **pageContext** : javax.servlet.jsp.PageContext
- **config** : javax.servlet.ServletConfig (paramètres servlet)
- **request**
- **response**
- **session**
- **application** : javax.servlet.ServletContext

Une JSP contient des objets prédéfinis

exemple d'utilisation de out

```
<html> <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>date page 3</title>
</head> <body>
<%
// This scriptlet declares and initializes "date"
System.out.println( "Evaluating date now" );
java.util.Date date = new java.util.Date();
%>
Hello! The time is now
<%
// This scriptlet generates HTML output
out.println( String.valueOf( date ));
%>
</body> </html>
```

Utilisation de la variable *request*

date4.jsp

```
<html> <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>date page 4</title>
</head> <body>
<% // This scriptlet declares and initializes "date"
System.out.println( "Evaluating date now" );
java.util.Date date = new java.util.Date(); %>
Hello! The time is now
<%
out.println( date );
out.println( "<br/>Your machine's address is " );
out.println( request.getRemoteHost());
%>
</body> </html>
```

Exemple d'utilisation de la variable *response*

Redirection de l'URL

```
<html> <head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>redirection</title>
<%
// envoie au client une redirection vers une nouvelle page
// (nouvelle requete cliente)
response.sendRedirect(anotherUrl );
%>
</body> </html>
```

Exemple de fragment jsp

```
<TABLE BORDER=2>
<%
     for ( int i = 0; i < n; i++ ) {
          %>
          <TR>
          <TD>Number</TD>
          <TD><%= i+1 %></TD>
          </TR>
          <%
     }
%>
</TABLE>
```

Exemple de fragment jsp

```
<%
    if ( hello ) {
        %>
        <P>Hello, world
        <%
    } else {
        %>
        <P>Goodbye, world
        <%
    }
%>
```

Exercice

- Créer une JSP qui affiche toutes les valeurs retournées par System.getProperties() avec "
" après chaque couple clé/valeur.

Déclaration de méthodes/variables

Pour pouvoir ajouter des méthodes/variables, il faut que la scriptlet soit encadrée par <%! et %>

Exemple

```
<%@ page import="java.util.*" %>
<HTML> <BODY>
<%!
     Date theDate = new Date();
     Date getDate()     {
          System.out.println( "In getDate() method" );
          return theDate;
     }
%>
Hello!  The time is now <%= getDate() %>
</BODY> </HTML>
```

Attention

- Dans cet exemple la date ne sera pas recalculée !

Cycle de vie d'une JSP

Principe

- Une JSP traite les requêtes de la même manière qu'une servlet : Les JSP reposent entièrement sur la technologie issue des servlets.
- Si une requête correspond à une JSP, le conteneur web vérifie d'abord si la servlet de la JSP est plus vieille que la jsp.
- Si c'est le cas, le conteneur web traduit la JSP en servlet et la compile.
- Avantage : durant le développement, le processus de construction (compilation) est réalisé automatiquement.

Traduction de la JSP

(1) les données statiques sont directement intégrées dans la réponse et (2) les éléments JSP sont traités comme suit :

Traduction et compilation

1. Des directives (serveur) sont utilisées pour controler comment le conteneur web traduit et exécute la JSP.
2. Les éléments de scripts sont insérés dans la servlet de la JSP.
3. les expressions sont passées en paramètres à un évaluateur JSP ($).
4. les éléments *jsp :[set|get]Property* : appels de méthodes sur des composants JavaBeans.
5. les éléments *jsp :[include|forward]* : appels sur l'API des servlets.
6. les éléments *jsp :plugin* : convertis en tags pour le navigateur (pour insérer une applet par exemple).
7. Les tags personnalisés sont convertis en appels sur le *tag handler* qui implémente ces tags.

Cycle de vie d'une JSP

Dans le serveur d'application Java EE

- Le fichier source d'une servlet créée à partir d'une JSP appelée *pageName* se trouve dans un fichier appelé : *pageName_jsp.java* (la localisation dépend du serveur Java EE utilisé)
- Par exemple, sous Eclipse avec Tomcat, dans : .metadata/.plugins/org.eclipse.wst.server.core/tmp0/work /Catalina/localhost/nom_projet/...

En cas d'erreur

- Les phases de traduction et de compilation peuvent toutes les deux produirent des erreurs qui ne sont observées que lors du premier chargement de la JSP.
- En cas d'erreur, le serveur renvoie une *JasperException* et un message qui inclue le nom de la JSP et la ligne d'erreur.

Cycle de vie d'une JSP

Une fois la JSP traduite et compilée :

- le cycle de vie suit le même principe que pour une servlet normale :
- Si la servlet n'existe pas encore, le conteneur :
  1. charge la classe de la servlet
  2. l'instancie
  3. initialise l'instance de la servlet avec la méthode *jspInit*
- Enfin, le conteneur invoque la méthode _jspService : passage des objets *request* et *response*
- Si le conteneur a besoin d'enlever la servlet, il invoque la méthode *jspDestroy*

Cycle de vie d'une JSP : Exécution

Il est possible de controler l'exécution d'une page JSP en utilisant des *directives*

Buffering

```
<%@ page buffer="none|xxxkb" %>
```

Cette directive permet de spécifier la taille du buffer avant envoi (pour finir la construction de la page par exemple (petit buffer = envoi rapide vers le client)

Cycle de vie d'une JSP : en cas d'erreur

Traitement des erreurs

```
<%@ page errorPage="file-name" %>
```

Cette directive, placée dans l'en-tête de la page, permet de spécifier une page à afficher en cas d'erreur.

Cycle de vie d'une JSP : jsp pour les erreurs

Exemple de page pour récupérer les erreurs : *errorpage.jsp*

```
<%@ page isErrorPage="true" %> <%-- genere un objet de
   type javax.servlet.jsp.ErrorData --%>
<html> <head> <title>Exceptional Event Occurred!</title>
 </head>
<%-- Exception Handler --%>
<font color="red">
<%= exception.toString() %><br>
<%
 out.println("<!--");
 StringWriter sw = new StringWriter();
 PrintWriter pw = new PrintWriter(sw);
 exception.printStackTrace(pw);
 out.print(sw);
 sw.close();
 pw.close();
 out.println("-->");
%>
 </body> </html>
```

Cycle de vie d'une JSP : Exécution

Exemple d'autres directives

```
<%-- inclusion de fichier --%>
<%@ include file="hello.jsp" %>

<%-- importation de packages --%>
<%@ page import="java.util.*" %>

<%-- pour que le navigateur puisse interpreter le resultat --%>
<%@ page contentType="text/vnd.wap.wml"%>

<%-- encodage de la reponse --%>
<%@ page contentType="text/html; charset=UTF-8" %>

<%-- encodage de la page elle-meme --%>
<%@ page pageEncoding="UTF-8" %>
```

Plus d'informations sur les types de contenu possibles

- The Internet Assigned Numbers Authority (IANA)

Les tags JSP

```
<prefix:tagName />
<prefix:tagName param="blabla"/>
<prefix:tagName> body </prefix:tagName>
```

Principe

- similaire au principe des tags html
- Ils peuvent être de 2 types différents :
  1. prédéfinis en standard : ***jsp :fonction***
  2. Chargés depuis une librairie externe (e.g. la JSTL)

Exemple

1. `<jsp:include page="hello.jsp"/>`
2. `<mytag:helloWorld />`

La JSP Standard Tag Library (JSTL)

Objectifs et caractéristiques

- Faciliter la création de jsp pour les développeurs web
- Permettre le développement des pages JSP en utilisant des balises XML : conception de pages dynamiques sans connaissance du langage Java.
- Sun a définit les spécifications mais l'implémentation est libre.

JavaServer Pages Standard Tag Library (JSTL)

- ▸ Tag Library Documentation
- ▸ *Tag Library API implementation*, fondation Apache

Les librairies JSP Tag

Un tag JSP

- balise XML qui fait la correspondance avec une classe Java
- ces tags sont remplacés lors de la compilation par l'utilisation de la classe

Les librairies JSP Tag

Exemple

```
<tagLibName:tagName key="value">
<tagLibName:tagName>
```

Détails

- *tagLibName* : préfixe déterminant la taglib à utiliser
- *tagName* : correspond à la classe Java à utiliser
- *key="value"* (optionnel) : des couples attribut/valeur
- *body* (optionnel) : utilisation

Un exemple

Affichage de la date avec Locale

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>

<jsp:useBean id="date" class="java.util.Date" />

<html>
<head> <title><fmt:message key="Welcome" /></title> </head>
<body>
  <h2>
    <fmt:message key="Hello" /> <fmt:message key="and" />
    <fmt:message key="Welcome" />
  </h2>

  <fmt:formatDate value="${date}" type="both" dateStyle="full"
    timeStyle="short" />
  <br /> Locale:
  <c:out value="${pageContext.request.locale.language}" />
  <c:out value="${pageContext.request.locale.country}" />
</body> </html>
```

Hello World version JSP + tag

Hello World

Hello, my name is Duke !! What's yours?

Submit Reset

Hello, bob!!!!

Hello World version JSP + tag

hello.jsp

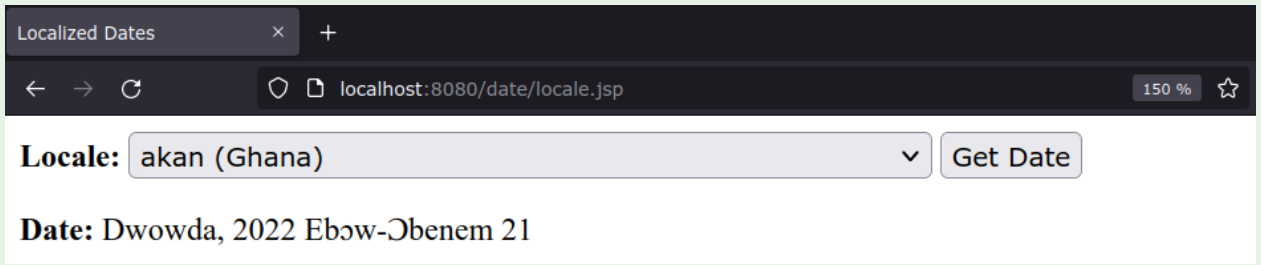
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html> <head><title>Hello</title></head> <body bgcolor="white">
<img src="duke.waving.gif"> <h2>Hello, my name is Duke. What's yours?
</h2><form method="get"> <input type="text" name="username" size="25">
<input type="submit" value="Submit"> <input type="reset" value="Reset"
</form>

<c:if test="${fn:length(param.username) > 0}" >
  <%@include file="response.jsp" %>
</c:if>
</body> </html>
```

response.jsp

```
<h2><font color="black">Hello, ${param.username}!</font></h2>
```

Création d'une librairie de tag

Affichage de l'heure en fonction de la localisation de l'utilisateur



Sources

- ▸ MyDate.java ▸ MyLocales.java
- ▸ functions.tld (à mettre dans le répertoire WEB-INF)
- ▸ locale.jsp

Bibliographie

Références

- Ce cours reprend largement le tutoriel Java EE proposé par Sun :
- ▸ Java EE tutoriel

Bibliographie

Références

- Ce cours reprend largement le tutoriel Java EE proposé par Sun :
- ▸ Jakarta EE tutoriel