Java Server Pages

Fonctionnement + Expression Language + Taglib JSTL

JSP en pratique

- Une JSP est un fichier texte
  - Il contient du code html (appelé template) et des balises spécifiques permettant de coder en java les scriptlets.
- Transformée par le serveur en servlet
  - Tout ce qu'une servlet peut faire, une jsp peut le faire
  - Il est utile d'aller voir le fichier java correspondant à la servlet générée afin de bien comprendre et situer ce que les balises de tags jsp produisent.
    - Répertoire work pour tomcat

Transition : JSP => Servlet

```
<%@page contentType="text/html" 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>JSP Page</title>
    </head>
    <body>
        <h1>Hello JSP World!</h1>
    </body>
</html>
```

Transition : JSP => Servlet

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

  private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();

  private static java.util.List _jspx_dependants;

  private javax.el.ExpressionFactory _el_expressionfactory;
  private org.apache.AnnotationProcessor _jsp_annotationprocessor;

  public Object getDependants() {
    return _jspx_dependants;
  }

  public void _jspInit() {
    _el_expressionfactory = _jspxFactory.getJspApplicationContext(getServletConfig().getServletContext()).getExpressionFactory();
    _jsp_annotationprocessor = (org.apache.AnnotationProcessor) getServletConfig().getServletContext().getAttribute(org.apache.AnnotationProcessor.class.getName());
  }

  public void _jspDestroy() {
  }
```

```
public void _jspService(HttpServletRequest request, HttpServletResponse response)       throws java.io.IOException, ServletException {

    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;

    try {         response.setContentType("text/html;charset=UTF-8");
                          pageContext = _jspxFactory.getPageContext(this, request, response,null, true, 8192, true);
                          _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();      _jspx_out = out;
            out.write("\n");
      out.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"\n");
      out.write("   \"http://www.w3.org/TR/html4/loose.dtd\">\n");
      out.write("\n");
      out.write("<html>\n");
      out.write("    <head>\n");
      out.write("        <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n");
      out.write("        <title>JSP Page</title>\n");
      out.write("    </head>\n");
      out.write("    <body>\n");
      out.write("        <h1>Hello JSP World!</h1>\n");
      out.write("    </body>\n");
      out.write("</html>\n");
    } catch (Throwable t) {
      if (!(t instanceof SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          try { out.clearBuffer(); } catch (java.io.IOException e) {}
        if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

Cycle de vie

- Identique aux servlet
  - jspInit()
  - _jspService()
  - jspDestroy()

  - Ces méthodes sont utilisées comme celles des servlets et sont redéfinissables
    - On pourrait imaginer modifier init() tout court mais le server l'utilise pour la génération de servlet de manière plus complexe, il est donc mis a disposition ces fonctions pour éviter des « collisions »

Eléments de script

- Quatres types d'éléments de script :

  - **Les directives :** indiquent à la pages les informations globales (par exemple les instructions d'importations)
  - **Les déclarations :** destinées à la déclaration de méthodes et de variables à l'échelle d'une page
  - **Les scriplets :** code Java intégré dans la page
  - **Les expressions :** sous forme de chaîne, en vue de leur insertion dans la sortie de la page

JSP : les directives de page

- **Page** : informations relatives à la page
- **Include** : fichiers à inclure littéralement
- **Taglib** : URI d'une bibliothèque de balises utilisée dans la page

<%@ page

[language="java"] [extends="package.class"][import="{package.class|package.*}, ..."]
[session="true|false"]
[buffer="none|8kb|sizekb"] [autoflush="true|false"]
[contentType="mimeType [charset=characterSet]" |
"text/html;charset=ISO-8859-17"]
[iserrorPage="true|false"]

%>

JSP : les directives de page

- **Définir les "import"** nécessaires au code Java de la JSP
  - <%@ page import="java.io.*"%>
- **Définir le type MIME** du contenu retourné par la JSP
  - <%@ page contentType="text/html"%>
- **Fournir l'URL de la JSP à charger en cas d'erreur**
  - <%@ page errorPage="err.jsp"%>
- **Définir si la JSP est une page invoquée en cas d'erreur**
  - <%@ page isErrorPage="true" %>
- **Déclarer si la JSP peut être exécutée par plusieurs clients à la fois (note : experience à faire avec servlets)**
  - <%@ page isThreadSafe="false" %>

JSP : les directives de page

```jsp
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page errorPage="ErrorDiv.jsp" %>
<html>
    <head>
        <title>Exemple @ errorPage</title>
    </head>
    <body>

    <h3>Division de 2 nombres au hasard compris entre 0 et 5</h3>
    <%int denom = (int)(Math.random() * 5 );
        int numer = (int)(Math.random() * 5 );%>
    Le résultat de la division de
    <%=numer%> par <%=denom%> est : <%=numer/denom%>
    </body>
</html>
```

Exp_errorPage.jsp

```jsp
<%@ page isErrorPage="true" %>
<html>
    <head>
    </head>
    <body>
 <h2>Erreur : Division par zéro </h2>
    </body>
</html>
```

ErrorDiv.jsp

http://localhost:8084/AppliWeb_JSP/Exp_errorPage.jsp

Division de 2 nombres au hasard compris entre 0 et 5

Le résultat de la division de 3 par 1 est : 3

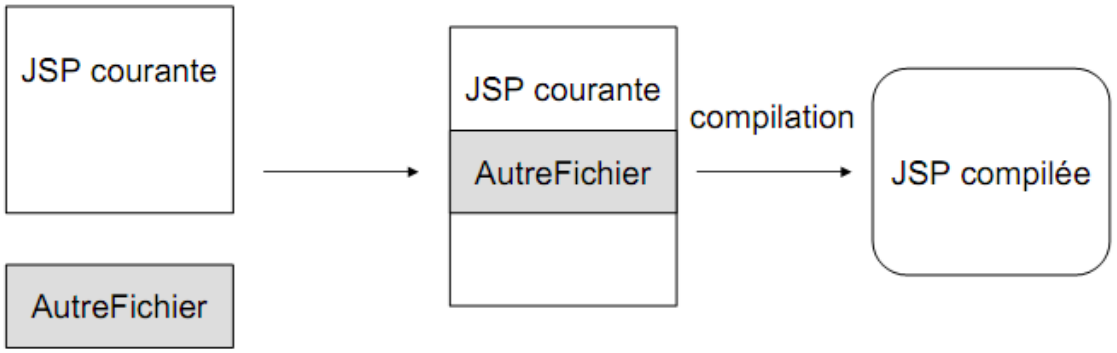
http://localhost:8084/AppliWeb_JSP/Exp_errorPage.jsp

Erreur : Division par zéro

JSP : les directives d'inclusion

- **<%@ include ……%>**
  - – Permettent d'inclure le contenu d'un autre fichier dans la page JSP courante
  - – Inclusion effectuée avant la compilation de la jsp

```
<%@ include file="AutreFichier"%>
```



JSP : les directives d'inclusion

```jsp
<html>
    <head>
        <title>Exp @ include</title>
    </head>
    <body>
    <%
    String le_nom = "Dupont";
    String le_prenom="toto";
    String l_adresse="Nice";
    %>
    <%@ include file="ficheInfo.jsp" %>
    </body>
</html>
```

Exp_include.jsp

```jsp
<h3> Informations sur la personne </h3>
Nom : <%=le_nom%><br>
Prénom : <%=le_prenom%><br>
Adresse : <%=l_adresse%><br>
```

ficheInfo.jsp

JSP : les balises personnalisées

- **<%@ taglib ……%>**
  - – Permettent d'indiquer une bibliothèque de balises

    adresse et préfixe, pouvant être utilisées dans la page

<%@ taglib prefix="pref" uri="taglib.tld" %>

JSP : les déclarations

- **<%! ……%>**
  - – Permettent de déclarer des méthodes et des variables d'instance connus dans toute la page JSP

```
<%!
private int mon_entier;
private int somme(int a, int b) {return (a+b);}
%>
```

Les scriplets

<%.....%>

– Permettent d'insérer des blocs de code java (*qui seront placés dans _jspService(…)*)

```
<% int som=0;
    for (int i=1; i< 15; i++)
    {som=somme(som,i);}
%>
```

Les scriplets

Donnent accès à une liste d'objets implicites à partir de l'environnement de la servlet :

- request : requete du client (classe dérivée de HttpServletRequest)
- response : réponse de la page JSP (classe dérivée de HttpServletRsponse)
- session : session HTTP correspondant à la requête
- out : objet représentant le flot de sortie
- Etc.

Les expressions

<%=……%>

- Permettent d'évaluer une expression et renvoyer sa valeur (string)
- Correspond à *out.println(…);*

```
Nous sommes le : <%=new java.util.Date()%>
```

Les commentaires

Les commentaires:

<%--……--%>

– Permettent d'insérer des commentaires (*qui ont l'avantage de ne pas être visibles pour l'utilisateur*)

```
<%-- ceci est un commentaire --%>
```

JSP : éléments de script-objets implicites

- **request** : requête courante (HttpServletRequest)
- **response** : réponse courante (HttpServletResponse)
- **out** : flot de sortie permet l'écriture sur la réponse
- **session** : session courante (HttpSession)
- **application** : espace de données partagé entre toutes les JSP (ServletContext)
- **page** : l'instance de servlet associée à la JSP courante (this)

JSP : les éléments d'action

- permettent de faire des traitements au moment où la page est demandée par le client
  - utiliser des Java Beans
  - inclure dynamiquement un fichier
  - rediriger vers une autre page

- Constitués de balises pouvant être intégrées dans une page jsp (syntaxe XML)

<jsp: ……/>

JSP : les éléments d'action

– Actions jsp standards:

- jsp:include et jsp:param
- jsp:forward
- jsp:useBean
- jsp:setProperty et jsp:getProperty

JSP : include/param

jsp:include et jsp:param

- jsp:include : identique à la directive <%@ include … sauf que l'inclusion est faite au moment de la requête
  - Donc après compilation…
- jsp:param : permet de passer des informations à la ressource à inclure

JSP : include/param

jsp:include et jsp:param

```
<jsp:include page="ficheInfo.jsp" flush="true">
    <jsp:param name="le_nom" value="Dupont"/>
    <jsp:param name="le_prenom" value="toto"/>
    <jsp:param name="l_adresse" value="Nice"/>
</jsp:include>
```

JSP : forward

jsp:forward

- Permet de passer le contrôle de la requête à une autre ressource
- jsp:param permet ici aussi de passer des informations à la ressource de redirection

```
<jsp:forward page="Redirect.jsp">
    <jsp:param name="monParam" value="maValeur"/>
</jsp:forward>
```

JSP : useBean

jsp:useBean

- Permet de séparer la partie traitement de la partie présentation
- Permet d'instancier un composant JavaBean (classe java) qui pourra être appelé dans la page JSP

```
<jsp:useBean id="testBean" class="exemplesjsp.MonBean"/>
```

JSP : useBean-Java Bean

Java Bean

- Permet de coder la logique métier de l'application WEB
- L'état d'un Bean est décrit par des attributs appelés propriétés

JSP : useBean-Java Bean

Java Bean

– classe Java respectant un ensemble de directives

- Un constructeur public sans argument
- Des propriétés « prop » accessibles au travers de getteurs et setteurs: getProp (lecture) et setProp (écriture) portant le nom de la propriété

JSP : useBean-Java Bean

Java Bean

type getNomDeLaPropriété()

⇨ pas de paramètre et son type est celui de la propriété

void setNomDeLaPropriété(type)

⇨ un seul argument du type de lapropriété et son type de retour est void

Java Bean: Exemple

- Utilise le constructeur par défaut ne possédant aucun paramètre

```
import java.util.Date;
public class BeanPersonne {

    private String leNom,lePrenom;
    private Date dateNaiss;

    public String getLeNom()
    {return leNom;}
     public void setLeNom(String leNom)
    {this.leNom=leNom;}

     public String getLePrenom()
    {return lePrenom;}
     public void setLePrenom(String lePrenom)
    {this.lePrenom=lePrenom;}

     public Date getDateNaiss()
    {return dateNaiss;}
     public void setDateNaiss(Date dateNaiss)
    {this.dateNaiss=dateNaiss;}
}
```

JSP : useBean

jsp:useBean

```
<jsp:useBean id="personne" class="mesBeans.BeanPersonne" scope="session"/>
```

Nom de l'instance

package.class du bean

Champ d'existence de l'objet Bean:

- request
- page
- session
- application

JSP : get/setproperty

jsp:setProperty et jsp:getProperty

– Permet de récupérer ou de modifier les valeurs d'une instance de JavaBean

- **Récupération :**

```
<jsp:getProperty name= "testBean" property= "maProp" />
```

Équivalent à: `<%=testBean.getMaProp()%>`

JSP : get/setproperty

jsp:setProperty et jsp:getProperty

- **Modification :**

  - Attribuer automatiquement aux attributs les valeurs récupérés de la requête

```
<jsp:setProperty name= "testBean" property= "*" />
```

  - Attribuer directement une valeur de paramètre à un attribut

```
<jsp:setProperty name= "testBean" property= "maProp" param="monParam"/>
```

  - Attribuer directement une valeur à un attribut

```
<jsp:setProperty name= "testBean" property= "maProp" value="3"/>
```

JSP : get/setproperty

```jsp
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>


<jsp:useBean id="personne" class="beans.BeanPersonne" scope="session"/>
<jsp:setProperty name= "personne" property= "leNom" value="Martin"/>
<jsp:setProperty name= "personne" property= "lePrenom" value="toto"/>
<jsp:setProperty name= "personne" property= "dateNaiss" value="<%=new java.util.Date()%>"/>
<html>
    <head>
        <title>Fiche Personne</title>
    </head>
    <body>
    nom: <jsp:getProperty name= "personne" property= "leNom"/><br>
    prénom: <jsp:getProperty name= "personne" property= "lePrenom"/> <br>
    date de naissance: <jsp:getProperty name= "personne" property= "dateNaiss"/>
    </body>
</html>
```

nom: Martin
prénom: toto
date de naissance: Mon Mar 31 10:21:43 CEST 2008

Combinaison Servlet et JSP

Transmission de requête vers une jsp ou inclusion de la jsp dans la servlet : Utilisation de RequestDispatcher

```
RequestDispatcher MyJspDispat =
    getServletContext().getRequestDispatcher("/folder/page.jsp");
```

- MyJspDispat.include(req,res) : la ressource est insérée dans la servlet active.
- MyJspDispat.forward(req,res) : le flux est complètement redirigé vers la nouvelle ressource (l'appelant ne peut plus effectuer de sortie au client, cette tâche est transmise à la nouvelle ressource uniquement)