Applications Web

Cours 4: Les JSP

Khaled Khelif

PLAN

- JSP : présentation générale
- JSTL : une taglib simplifiant l'écriture des JPS

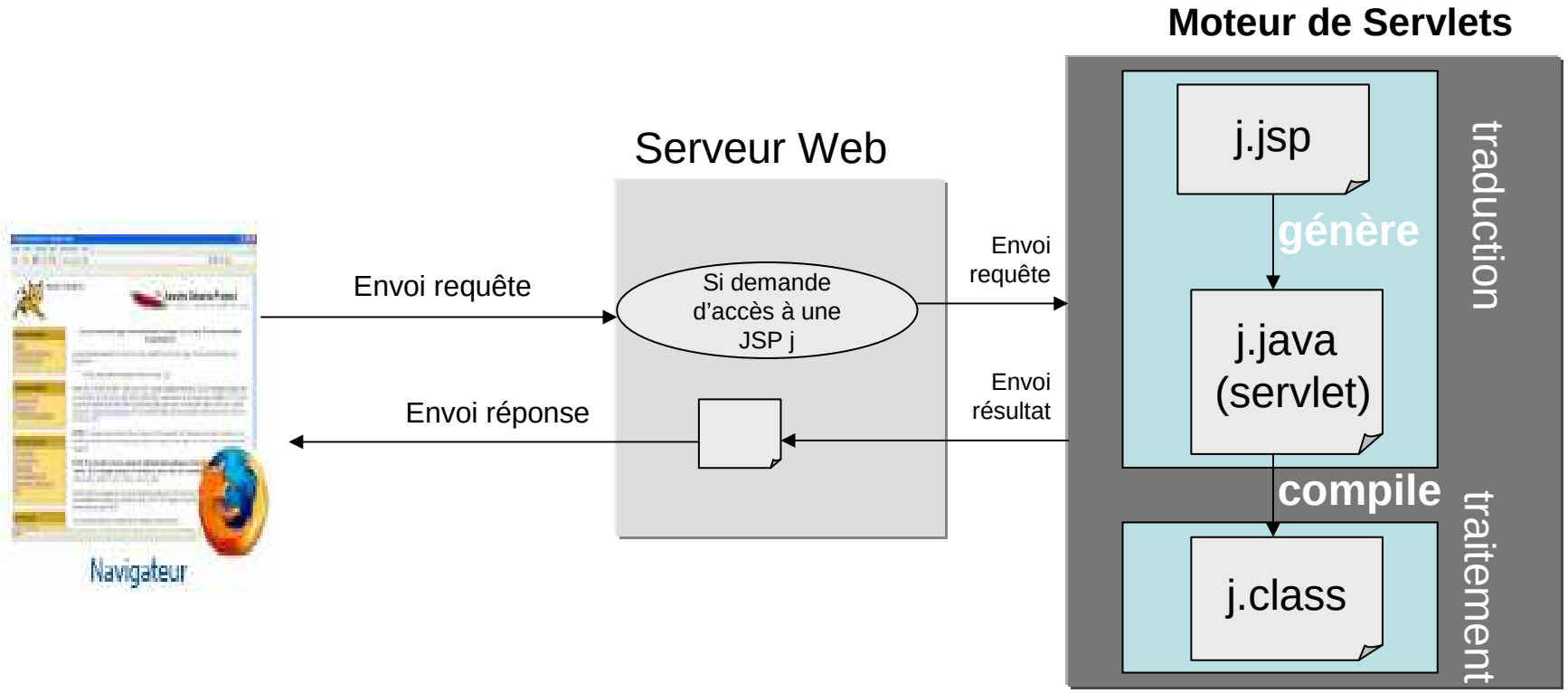
JSP : présentation

- Servlet:
  - classe java qui étend javax.servlet.httpServlet
  - Accent mis sur le code java
- JSP:
  - Java Server Pages
  - Code Java embarqué dans une page HTML entre les balises <% et %>
  - Séparation entre traitement de la requête et génération du flux html

JSP : présentation

- JSP:
  - désignés par une URL **http://localhost:8080/AppliWeb_JSP/index.jsp**
  - fragments de code Java exécutés sur le moteur de Servlets
  - pages JSP sont converties en Servlet par le moteur de Servlets lors du premier appel à la JSP

JSP : fonctionnement



JSP vs Servlet : Exemple

```
public class welcome extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h3>Appli Web</h3>");
        out.println("Intitulé du cours : JSP<br>");
        out.println("Date du cours : " + new java.util.Date().toString());
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
```

Servlet welcome.java

JSP vs Servlet : Exemple

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h3>Appli Web</h3>
        Intitulé du cours : JSP<br>
        Date du cours : <%= new java.util.Date().toString()%>
    </body>
</html>
```

Page JSP welcome.jsp

JSP vs Servlet : Exemple

```
public final class welcome_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {
  public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    try {
        ....
      response.setContentType("text/html;charset=UTF-8");
      pageContext = _jspxFactory.getPageContext(this, request, response,null, true, 8192, true);
      out = pageContext.getOut();
      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("<html>\n");
      out.write("    <head>\n");
      out.write("        <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n");
      out.write("    </head>\n");
      out.write("    <body>\n");
      out.write("        <h3>Appli Web</h3>\n");
      out.write("        Intitulé du cours : JSP<br>\n");
      out.write("        Date du cours : ");
      out.print( new java.util.Date().toString());
      out.write(" \n");
      out.write("    </body>\n");
      out.write("</html>\n");
    } catch (Throwable t) {
```

Servlet générée
welcome_jsp.java

JSP : cycle de vie

- Identique au cycle de vie d'une Servlet:
  - Appel de la méthode jspInit() après le chargement de la page
  - Appel de la méthode _jspService() à chaque requête
  - Appel de la méthode jspDestroy() lors du déchargement

Rq: Il est possible de redéfinir dans la JSP les méthodes jspInit() et jspDestroy()

JSP : exemple (modification de jspInit())

```
<%@ page import="java.util.Date" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
    <%!
    int mon_compteur;
    Date ma_date;
    public void jspInit() {
        ma_date = new Date();
        mon_compteur=0;
    }
    %>
    <h3>Test : Modification de la méthode jspInit() </h3>
    Date de la première visite: <%= ma_date %><br>
    Nombre de visites : <%= ++mon_compteur %>
    </body>
</html>
```

JSP : éléments du code

- Page JSP:
  - Html: structure statique de la page
  - Code JSP: éléments dynamiques de la page

- 4 types d'éléments

JSP: éléments de script

- 4 types d'éléments de script:
  - Les directives : indiquent à la pages les informations globales (par exemple les instruction 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 chaine, en vue de leur insertion dans la sortie de la page

JSP : les directives

<%@……%>

Les directives de jsp 1.2:

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

```
<%@ 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 +sieurs clients à la fois

  <%@ 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

JSP : les directives d'inclusion

Les directives d'inclusion:

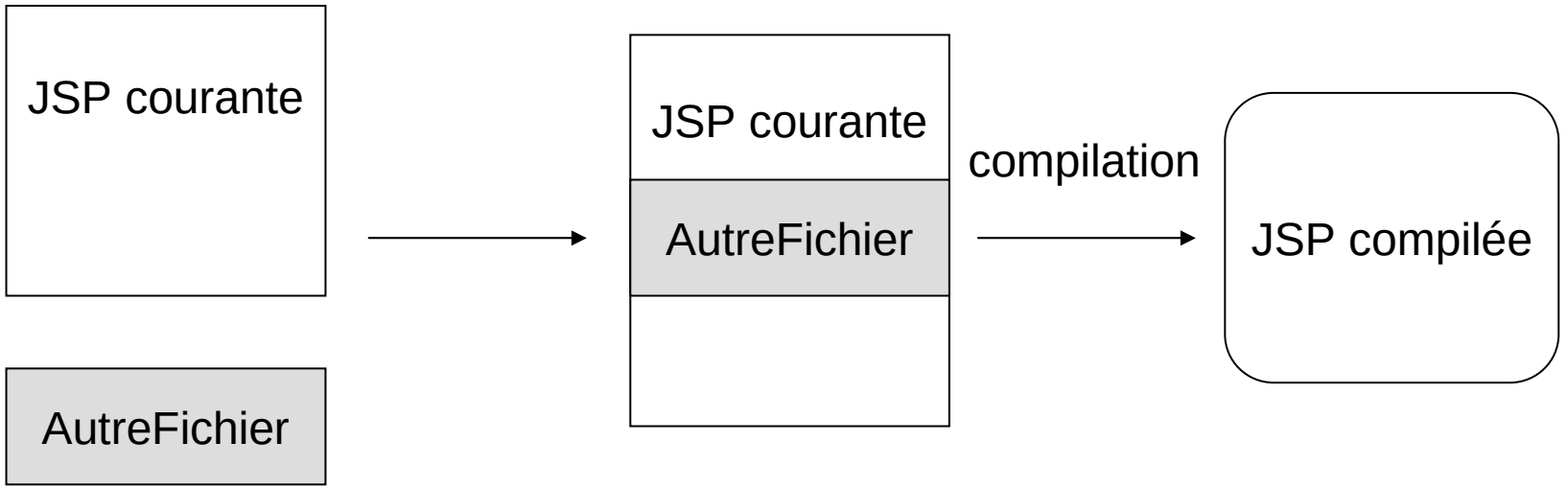
<%@ include ……%>

- Permettent d'inclure le contenu d'un autre fichier dans la page JSP courante

- Inclusion effectuée avant la compilation de la jsp

JSP : les directive d’inclusion

<%@ 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

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

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);}
%>
```

JSP : les scriplets

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);}
%>
```

JSP : 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.

JSP : les expressions

Les expressions:

<%=……%>

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

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

JSP : 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

JSP : useBean-Java Bean

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" classe="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)

Conclusion

- Les JavaServer Pages conjugue la puissance des servlets et la convivialité de HTML
- Une méthode performante pour la création d'applications Web coté serveur
- Un domaine qui évolue rapidement : voir JSP 2.0 http://java.sun.com/products/jsp/syntax/2.0/syntaxref20.html
- Les JavaBeans permettent de décomposer la présentation et la logique

Exercice

Compléter cette page de DEUX manières différentes pour
pouvoir afficher le message suivant : «Vous avez choisi la couleur <couleur> »

```
<html>
<body>
<form method=post action="selection.jsp">
Choisissez une couleur<p>
Rouge<input type=radio name=couleur value="rouge">
Bleu<input type=radio name=couleur value="bleu">
Vert<input type=radio name=couleur value="vert">
<p>
<input type=submit>
</form>

<! -- Ajout du code-->
</body>
</html>
```

JSTL

JSTL (Java Server Pages Standard Tag Library) : propose une librairie standard pour la plupart des fonctionnalités de base d'une application J2EE

- Extension pour le support de bibliothèque de “tags”
- Solution standardisée permettant d'utiliser uniquement des balises
- Possibilité de faire évoluer le comportement d'une balise sans d devoir réécrire le code qui l'utilise

JSTL

C’est un ensemble de balises personnalisées qui propose des fonctionnalités souvent rencontrées dans les page JSP :

- Tag de structure (itération, conditionnement)
- Internationalisation et formatage
- Exécution de requête SQL
- Utilisation de document XML

JSTL

- Des structures complexes: boucles, tests conditionnels...
- Des variables utilisateurs
- Accès à certaines variables particulières telles que request ou response
- Utilisation de l'Expression Language (EL) pour manipuler les données
  - en général entre accolades et précédées d'un dollar:

    ${param.exemple}

JSTL

Fonctionnement

⇨ Une bibliothèque (.jar) contient les fonctions associées

⇨ Un document XML (.tld) décrit les relations entre tags et fonctions

JSTL : la bibliothèque « core »

➢ Cette bibliothèque « core » comporte les actions de base pour la gestion des variables de scope d'une application web :

- Affichage de variable, création, modification et suppression de variables de scope et de gestion des exceptions
- Actions conditionnelles et boucles
- Manipulation d'URL et redirection

➢ Utilisation de la bibliothèque JSTL:core dans une appli WEB :

- Copier jstl.jar et standard.jar dans le répertoire WEB-INF/lib
- Copier le fichiers « c.tld » dans un sous répertoire de WEB-INF
- Modifier le fichier « web.xml » de manière à enrichir l'URI

```
<taglib>
    <taglib-uri>cjstl</taglib-uri>
    <taglib-location>/WEB-INF/tld/c.tld</taglib-location>
</taglib>
```

JSTL : la bibliothèque « core »

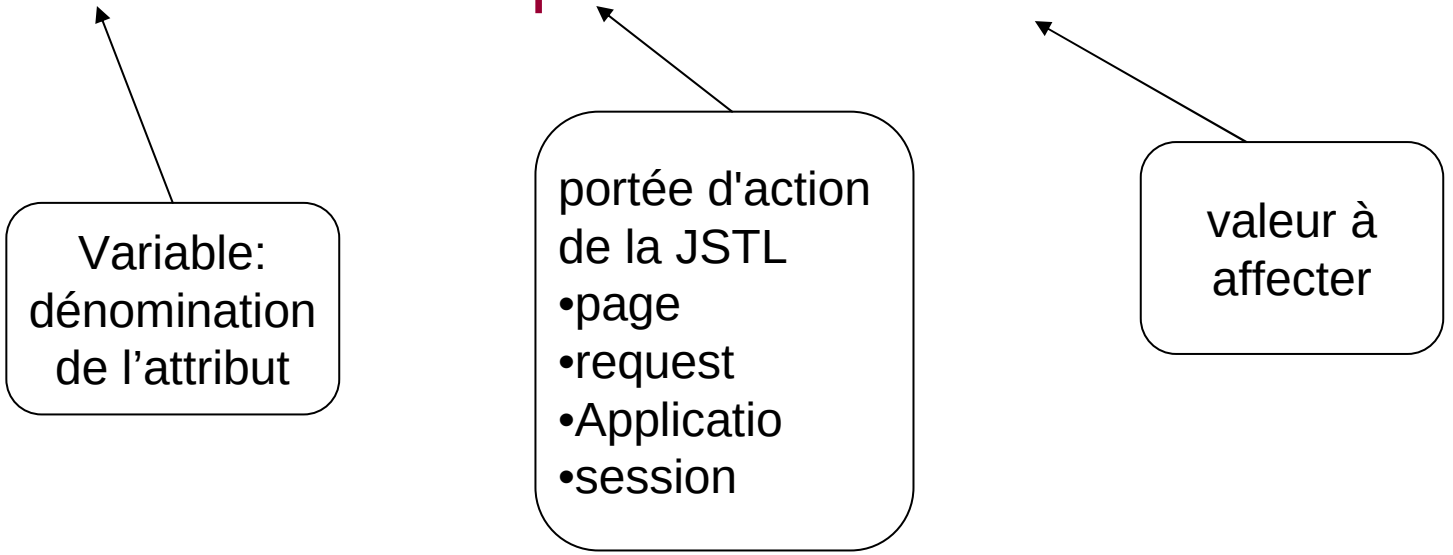
- Déclarer dans la page JSP la balise :

```
<%@ taglib uri="cjstl" prefix= "c" %>
```

JSTL

Exemple de balise + attributs (core):

```
<c:set var="varNom" scope="s"  value ="value" />
```



JSTL

Balises usuelles de core:

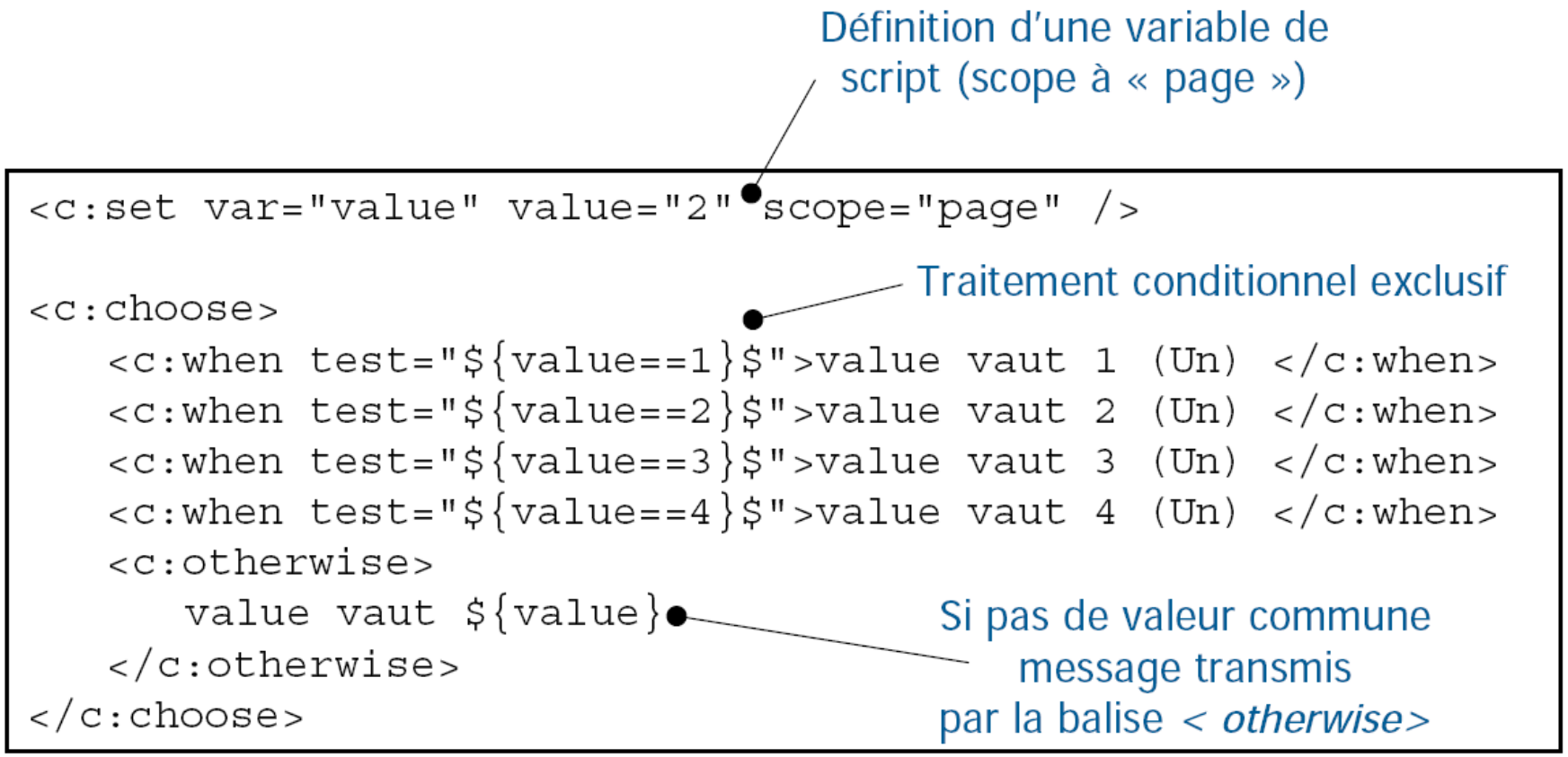
- <c:out value="${valeur}"/> :
  afficher une valeur dans la page HTML
- <c:set var="nom" scope="session" value="${10+2}"/>
  fixer la valeur d'une variable
- <c:if test="${...}">... </c:if>
  exécuter du code que si la condition est vérifiée.
- <c:forEach items="${Values}" var="i"> ... </c:forEach>
  boucler sur le contenu d'une structure (ici Values) en affectant chaque valeur à la variable i.
- <c:remove var="nom" scope="session" >
  effacer une variable.

Exemple JSP vs JSTL

```
<html>
<head>
<title>Compter en JSP</title>
</head>
<body>
<%
for(int i=1;i<=10;i++)
    {%>
    <%=i%><br/>
    <% } %>
</body>
</html>
```

```
<%@ taglib uri="http://java.sun.com/jstl/core"
prefix="c" %>
<html>
<head>
<title>Compter en JSTL</title>
</head>
<body>
<c:forEach var="i" begin="1" end="10" step="1">
    <c:out value="${i}" />
    <br />
</c:forEach>
</body>
</html>
```

JSTL :exemple




```
<c:set var="value" value="2" scope="page" />

<c:choose>
   <c:when test="${value==1}$">value vaut 1 (Un) </c:when>
   <c:when test="${value==2}$">value vaut 2 (Un) </c:when>
   <c:when test="${value==3}$">value vaut 3 (Un) </c:when>
   <c:when test="${value==4}$">value vaut 4 (Un) </c:when>
   <c:otherwise>
      value vaut ${value}
   </c:otherwise>
</c:choose>
```

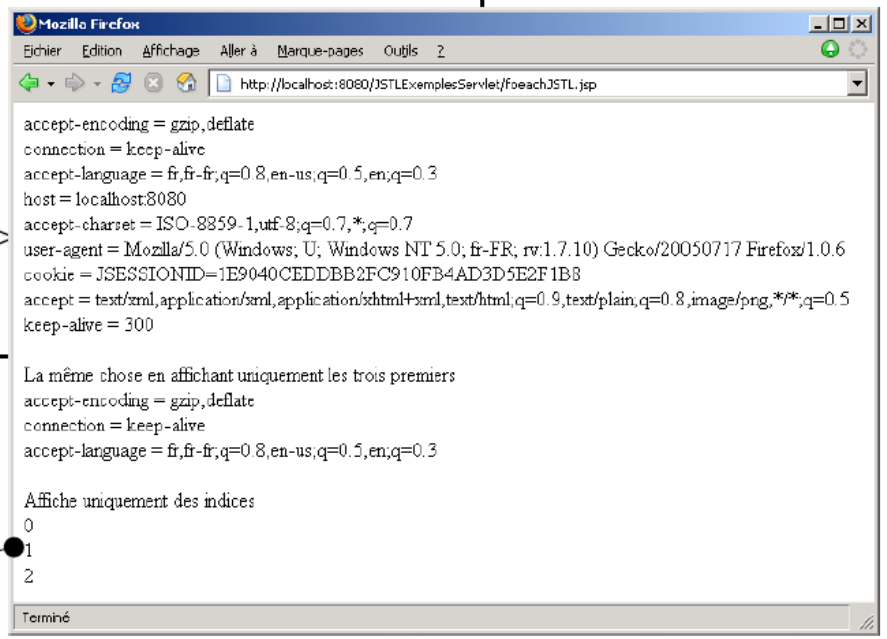
JSTL :exemple

```
<%@ taglib uri="montag" prefix="c" %>

<c:forEach var="entry" items="${header}" >
    ${entry.key} = ${entry.value}<br>
</c:forEach>

<br>
La même chose en affichant uniquement les trois premiers<br>
<c:forEach var="entry" items="${header}" begin="0" end="2" >
    ${entry.key} = ${entry.value}<br>
</c:forEach>

<br>
Affiche uniquement des indices<br>
<c:forEach var="entry" begin="0" end="2" >
    ${entry}<br>
</c:forEach>
```

Affiche tous les éléments

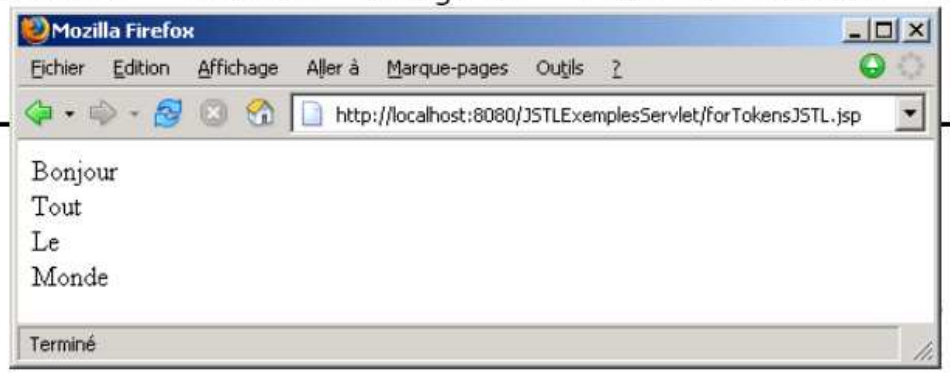
Affiche uniquement les trois premiers éléments

Affiche les nombres de 0 à 2



JSTL :exemple

```
<%@ taglib uri="montag" prefix="c" %>

<c:forTokens var="content" items="Bonjour Tout Le Monde" delims=" ">
    ${content}<br/>
</c:forTokens>
```



JSTL

Utilisation des JavaBean

- Accès direct' aux attributs sans passer par les accesseurs / modificateurs (<jsp:getProperty...>...)

⇨ instanceDuBean.NomDeAttribut

- exemple :

  Prénom : <c:out value="${lapersonne.le_prenom}"/>

Exemple d’utilisation des JavaBean

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<jsp:useBean id= "personne" class= "mesbeans.Personne" />
<jsp:useBean id= " billet" class= "mesbeans.Billet" />
<html>
      <c:if test="${personne.age > 25}">
          Prix du billet: <c:out value="${billet.prixAdulte} "/>
      </c:if>
</html>
```

Tutorial JSTL

http://adiguba.developpez.com/tutoriels/j2ee/jsp/jstl/