Les Java Server Pages

P.Mathieu

IUT de Lille
http://www.iut-a.univ-lille.fr
prenom.nom@univ-lille.fr

Les Java Server Pages (JSP)

Université de Lille

- JSP 1.0 proposés en 04/1999 par SUN
- JSP 2.0 proposés en 12/2003 par SUN
- Initialement, alternative aux ASP (Microsoft)
- Principe contraire aux servlets : on écrit du HTML avec du code dedans
- Ressemble sémantiquement et syntaxiquement à ASP ou PHP, ... en bien plus puissant !
- Ce sont avant tout des servlets ! !

Université de Lille

C'est du HTML qui contient du JAVA

```
<HTML>
<BODY>
<H1>Les 5 premiers entiers </H1>
<%
   for (int i=1; i<=5; i++) {
       out.println(i+"<br>");
   }
%>
</BODY>
</HTML>
```

Université de Lille

request pour l'objet `HttpServletRequest` de la Servlet.

response pour l'objet `HttpServletResponse` de la Servlet.

out pour l'objet `PrintWriter` de la Servlet.

in pour l'objet `BufferedReader` de la Servlet.

session pour l'objet `HttpSession`

application pour l'objet `ServletContext` du contexte

config pour l'objet `ServletConfig` de la servlet

page , l'instance de la servlet

exception pour l'exception qui a provoquée l'appel à une `errorPage`.

Les éléments de script

Les scriplets

Université de Lille

- Les scriptlets sont des bouts de code java exécutés au moment de l'affichage de la page.
- Tout ce que l'on peut écrire en Java peut aussi être mis dans une scriptlet
- Balises <% ...%>

```
<HTML>
<BODY>
<H1>Les 5 premiers entiers </H1>
<%
   for (int i=1; i<=5; i++) {
       out.println(i+"<br>");
   }
%>
</BODY>
</HTML>
```

Université de Lille

- la JSP expression est un code évalué et remplacé par son résultat
- Faire comme si l'expression était dans un `out.println()` ... il n'y a donc jamais de ';'
- Balises `<%= ...%>`

```
<HTML>
<BODY>
<H1>Les 5 premiers entiers </H1>
<% for (int i=1; i<=5; i++) { %>
       <%= i %> <br>
<% } %>
</BODY>
</HTML>
```

Université de Lille

L'affichage d'une variable

- par l'expression : `<%= nb %>`
- par la scriptlet : `<% out.print(nb); %>`

L'affichage du code HTML

- Hors d'une scriptlet : `<h1>Hello world</h1>`
- Dans une scriptlet : `out.println("<h1>Hello World</h1>");`

- les JSP déclarations contiennent des attributs, classes internes ou méthodes de servlet
- Elles peuvent être placées n'importe où dans la page
- Balises <%! ...%>

```
<HTML>
<BODY>
<%! int cpt=0; %>
   <H1>Revoilà notre compteur</H1>
   <% cpt++; %>
   Cette page a été accédée <%= cpt %> fois
</BODY>
</HTML>
```

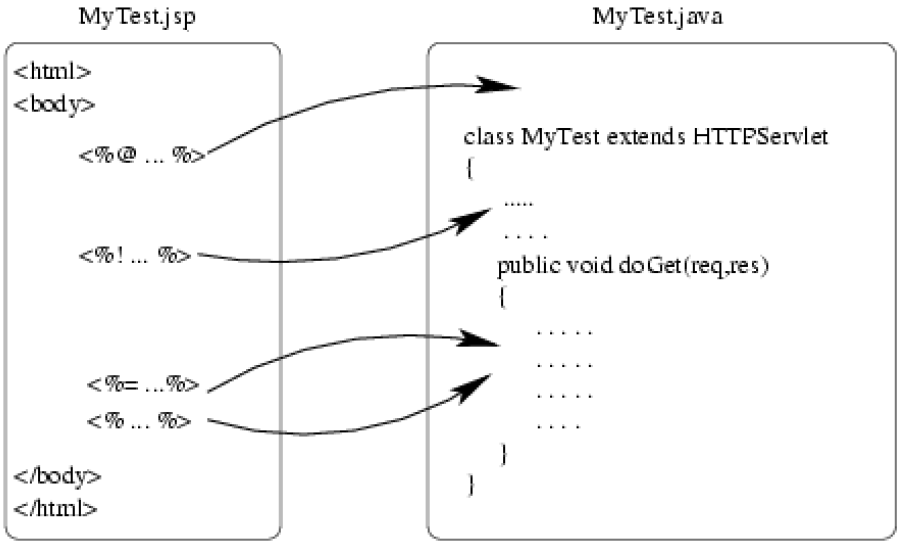
Université de Lille

- Les directives permettent d'indiquer les compléments nécessaires à la compilation
- Il en existe plusieurs, la plus courante sert à l'importation
- Balises `<%@page import ...%>`

```
<%@ page import="java.util.Date, jakarta.sql.*" %>
<HTML>
<BODY>
   <H1>Accès à la date par création d'un objet</H1>
   La date du jour est <%= new Date() %>.
</BODY>
</HTML>
```

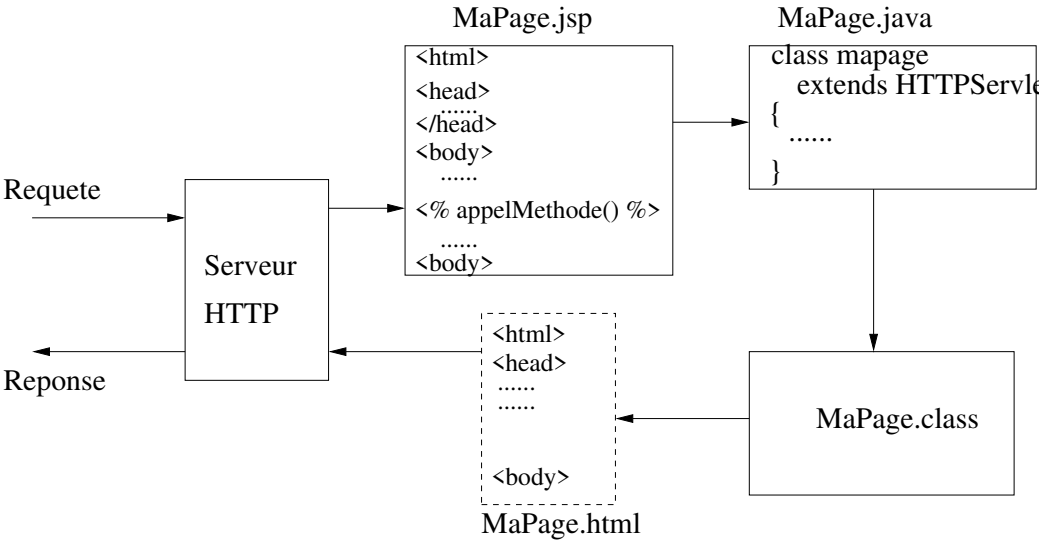
Les éléments de script

Traduction d'une JSP en Servlet



Les éléments de script

Exécution d'une JSP



Université de Lille

Les directives servent à envoyer des messages au conteneur JSP pour que la compilation aboutisse.

include : inclus un fichier à la compilation

page : définit les paramètres d'exécution de la JSP

taglib : permet d'utiliser des balises personnalisées (Jstl,displayTag...)

Université de Lille

- Permet d'insérer un fichier dans la JSP :
- `<%@ include file="toto.jspf" %>` insère les lignes du fichier `toto.jspf` en lieu et place de l'instruction au moment de la traduction en Servlet.
- Utile pour inclure des déclarations ou des objets.
  (Pour inclure le résultat d'une autre page nous verrons plus tard un tag `jsp:include` qui importe au moment où la page est servie.)
- Par convention les fichiers inclus portent l'extension `.jspf` et sont placés dans le répertoire `WEB-INF/jspf`

Utilisation avancée

La directive page

Université de Lille

Cette directive contient 11 attributs pour la compilation

language . default java. `<%@ page language="java" %>`

extends . default none. Permet préciser la superclasse éventuelle.

import . default none. `<%@ page import="java.util.*" %>`

session . default true. `<%@ page session="false" %>`

buffer . default 8ko. Taille pour le buffer de sortie.

autoFlush . default false. Vidage du buffer.

errorPage . default none. Indique l'URL de la page de gestion d'erreur

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

isErrorPage .default false. Indique si la page est une errorPage.

contentType . default text/html. Mime et encodage de la Servlet.

isThreadSafe . default true. JSP threadée ou pas.

Utilisation avancée

Gestion de compteurs

```
<!--   cpt.jsp   -->
<html>
<head>
<title>Compteurs et Objets</title>
        <%@ page contentType="text/html;
↪   charset=UTF-8"%>
        <%@ page import="java.util.*" %>
        <%@ page errorPage="erreur.jsp" %>
</head>
<body>
   <%!  // déclaration de l'objet compteur
      public class Cpt
      {  private int val=0;
         public String getVal()
         {   return "" + val; }
         public void incr() {val++;}
      }
   %>
```

```
<% // initialisation du compteur global
      Cpt global=(Cpt)application.getAttribute("global");
      if (global==null)
      {  global=new Cpt();
         application.setAttribute("global",global);
      }
      global.incr();

      // initialisation du compteur local
      Cpt local=(Cpt)session.getAttribute("local");
      if (local==null)
      {  local=new Cpt();
         session.setAttribute("local",local);
      }
      local.incr();
   %>

   <h1>
     Vous avez accédé <%= local.getVal() %> fois à cette
     page sur les <%= global.getVal() %> accès effectués.
   </h1>
</body>
</html>
```

Utilisation avancée

La page d'erreurs

Université de Lille

```
<!-- erreur.jsp -->
<%@ page isErrorPage="true" %>
<html><head>
        <title>page d'affichage d'erreur</title>
      </head>
      <body>
         <center>
             <h3><%= exception.toString() %></h3>
         </center>
      </body>
</html>
```

Utilisation avancée

L'appel d'objets via une JSP

Université de Lille

- Comme pour les Servlets, n'importe quel objet peut être appelé par une Servlet
- Dans le cas d'une JSP, l'objet doit impérativement être placé dans un package

```
package metier;
public class Personne
{
    public Personne(){...}
    ....
}
```

```
<%@ page contentType="text/html;
↪  charset=UTF-8"%>
<%@ page import="metier.Personne" %>
<html>
<body>
<h1> ... </h1>
<%
      ...
      p = new Personne();
      ...
%>
<%=p%>
```

Les EL expressions

Objectif

Université de Lille

- L'objectif d'une JSP est avant tout d'afficher des informations.
- les informations sont rangées un peu partout !
  (dans la page, la session, la requete etc ...)
- Les tags EL évitent d'avoir à passer d'un mode HTML à un mode java pour les récupérer.

- Introduites par SUN depuis JSP 2.0 (version Java EE 1.4)
- En standard dans JSP

Les EL expressions

Avantages

Université de Lille

- Eloigner encore plus le code Java des JSP
- Améliorer la lisibilité du code
- Eviter les cast
- Unifier l'accès aux Map et aux objets
- Permettre les accès en cascade (Maps de Maps ou d'objets)
- Gerér les exceptions classiques sur les accès

Les EL expressions

Principe

Université de Lille

- Une EL expressions est de la forme `${expression}`
- Elle est remplacée à l'exécution par le résultat de son évaluation.

```
${387}

${(2*3)>8}

${(2*3)>5?"grand":"petit"}
```

Les EL expressions

Accès aux paramètres HTTP

Université de Lille

La map `param` permet d'accéder directements aux paramètres

```
http://localhost:8080/vide/test.jsp?nom=paul
```

test.jsp

```
${param.nom}
```

(plus souple que `<%=request.getParameter("nom") %>`-

Les EL expressions

Les opérateurs classiques et les expressions peuvent se mélanger

Université de Lille

opérateurs classiques : `+ - * / %`, inline if (?), `&& || !`

```
http://localhost:8080/vide/test.jsp?age=12
```

test.jsp

```
${2*param.age}


${param.age>18?"majeur":"mineur"}
```

Le cast est automatique !

Les EL expressions

Accès aux données en session

Université de Lille

La map `sessionScope` permet d'accéder directements aux paramètres

```
session.setAttribute("login","paul");
```

test.jsp

```
${sessionScope.login}
```

(plus souple que `<%=request.getSession(true).getAttribute("login")%>` ou que `<%=session.getAttribute("login") %>`)

Les EL expressions

Plusieurs maps implicites sont fournies !

Université de Lille

pageScope : Map d'accès aux différents attributs de 'page'.

requestScope : Map d'accès aux différents attributs de 'request'.

sessionScope : Map d'accès aux différents attributs de 'session'.

applicationScope : Map d'accès aux attributs de 'application'.

param : Map d'accès aux paramètres de la requête

header : Map d'accès au Header HTTP

cookie : Map d'accès aux différents Cookies.

initParam : Map d'accès aux init-params du web.xml.

Les EL expressions

Principe

Université de Lille

L'expression peut être composée d'opérateurs qui s'appuient sur :

- un type de base java (int,long,double, ... String)
- Une Map
- un objet implicite prédéfini
- un attribut d'un Bean
- une fonction définie par l'utilisateur

Les EL expressions

Plusieurs types d'écriture possibles

Université de Lille

```
${map.cle}

${map["cle"]}

${map['cle']}
```

La notation crochets [] fonctionne aussi avec les List et les Array.

Université de Lille

Uniformatisation des accès aux Maps et aux objets
Utilisation d'une notation pointée, éventuellement en cascade

```
${map.cle}
ou
${objet.attribut}
```

```
${sessionScope.personne.nom}
```

Les EL expressions

La reflexivité

Université de Lille

```
<%!
public class Personne
{
  private String nom="paul";
  public String getNom(){return nom;}
}
%>

<% session.setAttribute("p",new Personne()); %>
```

```
${sessionScope.p.nom}
```

Pour les objets les accesseurs sont construits par réflexivité
Quand on accéde à la propriété `nom`, c'est bien la méthode `getNom()` qui est appelée.

Les EL expressions

Attention !

Université de Lille

Si la variable `nom` n'est pas définie ...

```
${param.nom} -> ""

<%= request.getParameter("nom") %> -> null
```

Les EL expressions

Scan automatique des 4 Scopes

Université de Lille

Ordre : `page`, `request`, `session`, `application`

les `*Scope` sont donc facultatifs (bien que conseillés)

```
<%
Map notes=new HashMap();
notes.put("paul","5");
notes.put("pierre","15");
notes.put("jean","25");
notes.put("jacques","35");
session.setAttribute("notes",notes);
%>
```

`${sessionScope.notes.paul}` s'écrit plus simplement `${notes.paul}`

Les EL expressions

Accès aux tableaux et listes

Université de Lille

```
<%
  String[] animaux = {"chien", "chat", "souris"};
%>

${animaux[2]} <br />
${animaux['2']} <br />
${animaux["2"]} <br />
```

Le même code fonctionne avec une `List`

Université de Lille

Ensemble d'actions XML, raccourcis d'écriture

jsp :useBean Associe un bean à une JSP

jsp :setProperty Affecte une propriété d'un bean

jsp :getProperty Lit une propriété d'un bean

jsp :include Appelle la page au moment de l'exécution

```
<jsp:include page='fichierImporté' />
```

jsp :forward Renvoie la requête à une autre page

```
<jsp:forward page='urlRedirection' />
```

jsp :param permet de passer des paramètres à include ou forward

jsp :plugin Permet de gérer des balises spécifiques au browser

Université de Lille

- Gestion automatique des objets "métier"
- Accès en notation XML, sans avoir à connaitre Java
- Banalise les accès aux objets des différents contextes (`page`, `request`, `session`, `application`)
- Raccourci d'écriture : la balise `useBean` s'occupe de rechercher l'objet concerné, de l'instancier s'il n'existe pas et de le ranger au bon endroit pour sa persistance.

Université de Lille

- Constructeur vide, méthodes "`public`", accesseurs.
- Placé obligatoirement dans un package
- Exemple :

```
package test;
public class Personne
{private String nom;
    public Personne(){}
    public String getNom(){return nom;}
    public void setNom(String s){nom=s;}
}
```

Université de Lille

```
<jsp:useBean id="t" class="test.Personne" scope="session"/>
<%= t %>
```

- `id` : nom de l'instance du Bean dans la Servlet.
- `class` : nom de la classe définissant le Bean.
- `scope` : précise la durée de vie du Bean (`page` (par défaut), `request`, `session`, `application`)
- `beanName` : nom de fichier pour les beans sérialisés.
- `type` : type à associer au fichier. permet de "caster" dans le type spécifié

Les JSP Beans

Exemple d'utilisation

Université de Lille

```java
// Définition du Bean
package test;

public class Personne
{private String nom;
    public Personne(){}
    public String getNom(){return nom;}
    public void setNom(String s){nom=s;}
}
```

```
<!-- test.jsp -->
<html>
<body>

<jsp:useBean id="t" class="test.Personne"
                    scope="application"/>

<% t.setNom("Dupont"); %>

Valeur : <%= t.getNom() %>

</body>
</html>
```

`useBean` s'occupe de tout ! il fait les déclarations, teste si l'objet existe, le crée le cas échéant, fait les cast nécessaires ... etc

Université de Lille

- Lire une propriété d'un Bean
  ```
  <jsp:getProperty name="t" property="nom" />
  ```
- Ecrire une propriété d'un Bean
  ```
  <jsp:setProperty name="t" property="nom" value="Dupont" />
  ```
- Initaliser auto tous les attributs du Bean avec les params HTTP
  ```
  <jsp:setProperty name="t" property="*" />
  ```

Attention : Les méthodes `getProperty` et `setProperty` ne fonctionnent qu'avec des String

Les JSP Beans

Yet Another Compteur

Université de Lille

```java
// Compteur.java
package test;

public class Compteur
{
    private int val=0;
    public String getVal()
    { return "" + val; }
    public void incr() {val++;}
}
```

```jsp
<!--   compteur.jsp   -->
<html>
<head>
   <title>Les compteurs à l'aide de Beans</title>
</head>
<body>

<%@ page language="java" errorPage="erreur.jsp" %>
<jsp:useBean id="local"  class="test.Compteur"
                         scope="session" />
<jsp:useBean id="global" class="test.Compteur"
                         scope="application" />


<% local.incr(); global.incr(); %>

Vous avez accédé <%= local.getVal() %> fois à cette
page sur les <%= global.getVal() %> accès effectués.

</body>
</html>
```

Quand la balise BEAN est utilisée, c'est la balise "class" qui permet de retrouver la classe, tandis qu'avec une instanciation d'objet "à la main" il faut mettre un import.

```
package tools;
public class MonObjet
{
    public String toString(){return "all is ok with my object";}
}
```

```
<%@ page contentType="text/html; charset=UTF-8"  import="tools.*" %>
<html>
<body>
<h1>Test de mon objet sans balise BEAN</h1>

<% MonObjet m1 = new MonObjet(); %>
<%=m1%>

<h1>Test de mon objet avec balise BEAN</h1>

<jsp:useBean id="m2" scope="page" class="tools.MonObjet" />
<%=m2%>

</body>
</html>
```

Université de Lille

- Les formulaires WEB sont très souvent utilisés pour interagir avec l'utilisateur et collecter des données
- JSP fournit grâce aux Beans un dispositif de gestion très pratique

```
<HTML>
<BODY>
<FORM METHOD=POST ACTION="Traitement.jsp">
Votre nom ? <INPUT TYPE=TEXT NAME=username SIZE=20><BR>
Votre e-mail ? <INPUT TYPE=TEXT NAME=email SIZE=20><BR>
Votre age ? <INPUT TYPE=TEXT NAME=age SIZE=4>
<P><INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

Université de Lille

- Pour traiter ce formulaire on crée simplement un Bean dont les attributs correspondent exactement aux champs du formulaire
- Setters pour chacun des champs

```
package tools;
public class Personne
{
    String username;
    String email;
    int age;

    public void setUsername( String value ){ username = value;}
    public void setEmail( String value ) { email = value;}
    public void setAge(int value){ age = value;}

    public String getUsername() { return username; }
    public String getEmail() { return email; }
    public int getAge() { return age; }
}
```

Traitement.jsp


```
<jsp:useBean id="p" class="tools.Personne" scope="session"/>
<jsp:setProperty name="p" property="*"/>
<HTML>
<BODY>
<A HREF="NextPage.jsp">Continuer</A>
</BODY>
</HTML>
```

- Tout ce que l'on a à faire c'est utiliser `property="*"`
- Le tag `useBean` va automatiquement rechercher une instance de Personne dans la session
- S'il en trouve une, il la met à jour
- S'il n'en trouve pas, il en crée une, l'initialise et la range dans la session
- Le tag `setProperty` va collecter automatiquement toutes les données en entrée et les palcer dans le Bean !

Bean et Formulaires

Les autres pages

Université de Lille

```
<jsp:useBean id="p" class="tools.Personne" scope="session"/>
<HTML>
<BODY>
Vous êtes<BR>
Nom: <%= p.getUsername() %><BR>
Email: <%= p.getEmail() %><BR>
Age: <%= p.getAge() %><BR>
</BODY>
</HTML>
```