Problématiques

- Les servlets sont des classes Java qui améliorent la fonctionnalité d'un serveur web en fournissant un contenu web dynamique.
- Elles sont couramment utilisées pour générer dynamiquement des pages HTML pour des sites web interactifs.

Problèmes de maintenance peuvent survenir :

- parce que la logique de présentation et la logique de génération de contenu dynamique sont mélangées;
- Chaque fois que la présentation est modifiée, le servlet doit être compilé et déployé à nouveau;
- Les développeurs doivent généralement écrire du code HTML dans le flux de sortie de la servlet, ce qui peut être fastidieux et peut coupler la servlet au format de sortie spécifique du contenu.

Solution Une des meilleures pratiques pour les servlets et les JSP consiste à séparer la logique de présentation et la logique de génération de contenu dynamique en utilisant le modèle de conception Modèle-Vue-Contrôleur (MVC) .

2

Java Server Page JSP

Karim AFDEL
kafdel@ymail.com

1

Servlet et JSP

- Exécutable avec tous les serveurs Web (Apache, IIS, …) auxquels on a ajouté un "**moteur**" de servlet/JSP (le plus connu :**Tomcat**)
- JSP compilées automatiquement en servlet par le moteur
- Servlet = du code Java contenant de l'HTML
- JSP = une page HTML contenant du code Java
- Concrètement avec les JSP :
  - les parties statiques de la page HTML sont écrites en HTML
  - les parties dynamiques de la page HTML sont écrites en Java

4

C'est quoi JSP?

- Code Java **embarqué** dans une page HTML entre les balises <% et %>
- Extension .jsp pour les pages JSP
- les fichiers .jsp sont stockés sur le serveur (comme des docs) ils sont désignés par une **URL** http://www.fsa.ac.ma/**exemple.jsp**
- le **chargement** de l'URL provoque l'**exécution** de la JSP **côté serveur**
- Programme Java s'exécutant **côté serveur Web**
- Servlet : prog. "autonome" stockés dans un fichier **.class** sur le serveur
- JSP : prog. **source** Java embarqué dans une page **.html**

3

3 parties d'une JSP

- Scriptlets **<% %>**
- Declarations **<%! %>**
- Expressions **<%= %>**

Scriptlets <% %>

Contient du code Java

Insérer dans jspservice() de la servlet, donc peut utiliser out, request, response, etc.

Exemple :

```
<% String[] langages = {"Java", "C++", "SmalItalk", "Simula 67"};
out.println("<h3>Principaux langages orientés objets : </h3>");
for (int i=0; i < langages.length; i++) { out.println("<p>" +
   langages[i] + "</p>"); } %>
```

6

Exemple 1

```
<HTML> <BODY>
<H1>Table des factorielles</H1>
<% int i,fact;
for ( i=1,fact=1 ; i<4 ; i++, fact*=i ) { out.print( i + "! =" + fact + "<BR>" ); }
%>
</BODY> </HTML>
```

Exemple 2: fichier date.jsp

```
<html><head><title>Obtenue par une JSP</title></head>
<body>
<h3>Bonjour de ma part </h3> <hr>
La date courante est : <%= new java.util.Date() %>
</body>
</html>
```

- Traité quand le client demande l'URL de la JSP :
  http://serveurWeb:<port>/.../date.jsp

5

Déclarations <%! %>

Sont des déclarations Java.

Permet de définir des méthodes ou des données membres

<%! field or method declaration %>

Exemples :

```
<%! int random4() {return
   (int)(Math.random() * 4);} %>
<%! int nombreFetiche = 2; %>
```

```
<%@page contentType="text/html"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type"
   content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%! int i = 38;%>
    <%= "Value of the variable is:" + i%>
  </body>
</html>
```

Expressions <%= %>

En fait expression Java qui renvoie un objet String ou un type primitif.

Un raccourci pour <% out.println(…); %>

<%= XXX %> ~ <% out.println(XXX); %>

!!!!!attention au **;**

est donc converti en out.println(…) dans la méthode _jspService(…) de la servlet.

Exemple:

La somme est: **<%= (195 + 9 + 273) %>**

Je vous réponds à l'adresse :

<%=request.getParameter("email") %>

Directive <%= ... %>

La directive <%= *expr* %> génère l'affichage d'une valeur de l'expression expr

<%= expr %> raccourci pour **<% out.print(*expr*); %>**

Exemple :

```
<HTML> <BODY>
<% int aleat = (int) (Math.random() * 5); %>
<H1> <%= aleat %> </H1>
</BODY> </HTML>
```

8

Scriptlets <% %>

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <title>JSP Page</title>
  </head>
  <body>
    <% out.print("Hello World");%>
  </body>
</html>
```

Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <body>
    <form action="Welcome.jsp">
      <pre>
      UserName : <input type="text" name="uname">
       Email  : <input type="text" name="Email">
            <input type="submit" value="Login"><br/>
    </pre>
    </form>
  </body>
</html>
```

Welcome.jsp

```
<%@page contentType="text/html"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <body>
    <pre>
    <% String name =
    request.getParameter("uname");
      String Email =
    request.getParameter("Email");
out.println("Welcome " + name+" Email : "
    +Email);
    %>
    </pre>
  </body>
</html>
```

7

Méthodes et variables d'instance

Des **méthodes** et des **variables** d'instance peuvent être associées à une JSP entre les directives <%! et %>

Exemple:

```
<HTML> <BODY>
<H1>Compteur</H1>
<%! int cpt = 0; // Variable d'instance- initialisée à l'instanciation de la JSP
// - persiste entre 2 invocations tant que la JSP ne change pas
// Méthode d'instance - attachée à l'objet correspondant à la JSP
int getCpt() {return cpt++;}
%>
<H1> <%= getCpt() %> </H1>
```

Remarque

Variables d'instance

Attention !!

<%! int cpt = 0; %> ≠ **<%** int cpt = 0; %>

Variable d'instance de la JSP (**persiste**)

Variable locale à la JSP (**réinitialisée à chaque invocation de la JSP**)

10

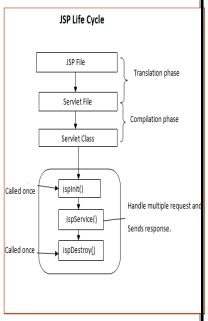
Mécanismes mis en œuvre

- plusieurs zones <% ... %> peuvent cohabiter dans une même JSP
- lors du premier chargement d'une JSP (ou après modification), le **moteur** rassemble **tous** les fragments <% ... %> de la JSP dans une classe
  - la **compile**
  - l'**instancie**

    JSP = objet Java présent dans le moteur
- puis, ou lors des chargements suivants, le **moteur** exécute le code dans un ***thread***

  → délai d'attente lors de la 1ère invocation dû à la compilation

  → en cas d'erreur de syntaxe dans le code Java de la JSP message récupéré dans le navigateur



9

Les objets implicites

Objets prédéclarés utilisables dans le code Java des JSPs

le flux de sortie pour générer le code HTML	out
la requête qui a provoqué le chargement de la JSP	request
la réponse à la requête de chargement de la JSP	response
l'instance de servlet associée à la JSP courante (= this)	page
l'exception générée en cas d'erreur sur une page	exception
suivi de session pour un même client	session
espace de données partagé entre toutes les JSP	application

12

D'autres Directives

La directive **<%@ page ... %>**

Donne des informations sur la JSP (non obligatoire, valeurs par défaut)

Exemple :

<%@ page import="..."%>

<%@ page import="java.io.*"%>

les "import" nécessaires au code Java de la JSP

<%@ page errorPage="..."%> (ex. **<%@ page errorPage="err.jsp"%>)**

fournit l'URL de la JSP à charger en cas d'erreur

<%@ page contentType="..."%> (ex. <%@ page contentType="text/html"%>)

le type MIME du contenu retourné par la JSP

<%@ page isThreadSafe="..." %> true ou false

true la JSP peut être exécutée par +sieurs clients à la fois (valeur par défaut)

<%@ page isErrorPage="..." %> true ou false

true la JSP est une page invoquée en cas d'erreur

11

Récupération des données d'un formulaire

Méthode String getParameter(String) de l'objet prédéfini request

- →retourne le texte saisi
- →ou null si le nom de paramètre n'existe pas

Exemple : Formulaire HTML

```
<HTML> <BODY>
<FORM ACTION=http://www.fsa.ac.ma/form.jsp METHOD=POST>
Nom <INPUT NAME="nom"> <P>
Prénom <INPUT NAME="prenom"> <P>
<INPUT TYPE=SUBMIT VALUE="Envoi">
<INPUT TYPE=RESET VALUE="Remise à zéro">
</FORM>
</BODY> </HTML>
```

Récupération des données d'un formulaire: form.jsp

```
<HTML> <BODY>
<H1>Exemple de résultat</H1>
Bonjour
<%= request.getParameter("prenom") %>
<%= request.getParameter("nom") %>
</BODY> </HTML>
```

13

Gestion des erreurs

- Erreur de syntaxe
  - dans les directives JSP (ex. : oubli d'une directive %>)
  - dans le code Java
- Erreur d'exécution du code Java (ex. : NullPointerException)

→ dans tous les cas, erreur récupérée dans le **navigateur** client

2 possibilités

- conserver la **page par défaut** construite par le moteur
- en concevoir une adaptée aux besoins particuliers de l'application
- en utilisant des directives **<%@ page errorPage="..." %> et**
- **<%@ page isErrorPage="..." %>**

Exemple de gestion d'erreur

```
<HTML> <BODY>
<H1>Pourvu ... !!</H1>
<% int hasard = (int) ( Math.random() * 5 );
%>
<H1> <%= 12 / hasard %> </H1></BODY> </HTML>
```

Résultat Si **hasard = 0** page d'erreur par défaut

14

Exemple :

```
<HTML> <BODY>
<H1>Pourvu ... !!</H1>
<%@ page errorPage="err.jsp" %>
<% int hasard = ... %>
<H1> <%= 12 / hasard %> </H1>
</BODY> </HTML>
```

err.jsp

```
<HTML> <BODY>
<%@ page isErrorPage="true" %>
<h1> Le 0 est sorti !! </h1>
erreur:
<%= exception.getMessage() %>
</BODY> </HTML>
```

Si hasard = 0 → page d'erreur **err.jsp** = 0

Récupération de l'erreur via l'objet prédéfini **exception**

15

Inclusion de JSP

- Agrégations des résultats fournis par **plusieurs JSP**

  → Meilleure modularité

  → Meilleure réutilisation

Directives <jsp:include> et </jsp:include>

```
<HTML> <BODY>
<H1>JSP principale</H1>
<jsp:include page="inc.jsp" >
</jsp:include>
</BODY> </HTML>
```

inc.jsp

```
<B>JSP incluse</B>
<P> <%= (int) (Math.random()*5) %>
</P>
```

!!! **Pas de <HTML> <BODY>**

16

Délégation de JSP

- Une JSP peut déléguer le traitement d'une requête à une autre JSP
  → prise en compte **complète** de la requête par la JSP déléguée
- **Directives <jsp:forward> et </jsp:forward>**

<HTML> <BODY>
<H1>JSP principale</H1>
<jsp:forward page="forw.jsp**" >**
</jsp:forward>
Ignoré !!
</BODY> </HTML>

18

Résultat

<HTML> <BODY>
<H1>JSP principale</H1>
<B>JSP incluse</B>
<P> <%= (int) (Math.random()*5) %>
</P> </BODY> </HTML>

Remarque

1. directives <jsp:include> et </jsp:include> inclusion statique
2. directive <%@ page include file="…"%> inclusion dynamique

L'élément <jsp:include> vous permet d'inclure soit un fichier statique ou dynamique dans un fichier JSP. Les résultats d'inclure des fichiers statiques et dynamiques sont très différentes. Si le fichier est statique, son contenu est inclus dans le fichier appelant JSP. Si le fichier est dynamique, elle agit sur la demande et le renvoie du résultat qui est inclus dans la page JSP. Lorsque l'action de "include" est terminée, le conteneur JSP continue le traitement du reste du fichier JSP.

17

Délégation et inclusion de JSP

Transmission de paramètres aux **inclus** et aux **délégués par** Utilisation de couples (name, value)

Directive <jsp:param name="..." value="..." />

Exemple :

```
<HTML> <BODY>
<H1>JSP principale</H1>
<jsp:forward page="forw.jsp">
<jsp:param name="nom" value="Taha" />
<jsp:param name="prenom" value="Mohamed" />
</jsp:forward>
</BODY> </HTML>
```

Récupération des paramètres ≡ à la récupération des paramètres transmis via un formulaire

```
<HTML> <BODY>
<H1>JSP déléguée/incluse</H1>
Nom : <%= request.getParameter("nom") %>
Prénom : <%= request.getParameter("prenom") %>
</BODY> </HTML>
```

20

Fichier forw.jsp

```
<HTML> <BODY>
<H1>JSP déléguée</H1>
<P>
<%= (int) (Math.random()*5) %>
</P>
</BODY> </HTML>
```

!!! Avec **<HTML> <BODY>**

Résultat

<HTML> <BODY>
<H1>JSP déléguée</H1>
<P>
<%= (int) (Math.random()*5) %>
</P>
</BODY> </HTML>

19

l'attribut scope

Il indique la portée du bean

Description	valeur
Le bean est valide pour cette requête. Il est utilisable dans les pages de redirection de la requête (<jsp:forward>). Il est détruit à la fin de la requête	**request**
Similaire à request, mais le bean n'est pas transmis aux pages de redirection <jsp:forward>. C'est la portée par défaut	**page**
Le bean est valide pour la session courante. S'il n'existe pas encore dans la session courante, il est créé et placé dans la session du client. Il est réutilisé jusqu'à ce que la session soit invalidée	**session**
Le bean est valide pour l'application courante. Il est créé une fois et partagé par tous les clients des JSP.	**application**

22

JSP et Java beans

But : avoir le moins de code Java possible dans une page JSP (HTML)

Sous-traiter le code à un Java bean

balise XML : <jsp:useBean>

Syntaxe générale :

<jsp:useBean id="*nomInstanceJavaBean*"

class="*nomClasseDuBean*"

scope="request|session|application|page">

</jsp:useBean>

- Le bean est alors utilisable par *nomInstanceJavaBean*
- balise sans corps donc utilisation de <jsp:useBean … />

21

Etape2 : Utilisation du bean dans une JSP

- Utilisation à l'aide de son nom
- Récupération des propriétés :
  - Par appel de méthode getXXX() :
  - Par la balise <jsp:getProperty …>

```
<p> on repère le bean par le nom nomBean <br>
<jsp:useBean id="nomBean" class="SimpleBean"
   scope="session">
</jsp:useBean>
<p> On accède a une propriété avec une expression:
<br> compteur = <%= nomBean.getCompter() %>
<hr>
On incrémente le compteur <% nomBean.increment(); %>
<p>On peut accéder à la propriété par une balise :<br>
<jsp:getProperty name="nomBean" property="compter" />
```

24

Exemple:Etape 1: Bean

```
public class SimpleBean implements java.io.Serializable
{
private int compter;
public SimpleBean() {
compter = 0;
}
public void setCompter(int theValue) {
compter = theValue;
}
public int getCompter() {
return compter;
}
public void increment() {
compter++;
}
}
```

23

Exemple

```
//File: index.jsp
<%@ page import="fsa.ac.ma.MessageBean" %>
<HTML> <HEAD>  <TITLE>Using a JavaBean</TITLE>
 </HEAD>
<BODY>
  <h1>Using a JavaBean</h1>
  <% MessageBean m = new MessageBean(); %>
  The message is: <%= m.msg() %>  </BODY>
</HTML>
////////////////////////////////////////////////
//File: MessageBean.java
package beans;
public class MessageBean
{  public MessageBean()  {  }
   public String msg()   { return "Hello from JSP!
";  }}
```

26

Positionner les propriétés du bean dans une JSP

- Par appel de méthode setXXX(…) :
- Par la balise <jsp:setProperty …>

```
<p> on repere le bean par le nom nomBean<br>
<jsp:useBean id="nomBean" class="SimpleBean"
  scope="session">
</jsp:useBean>
<p> On positionne une propriéte avec une expresion:
<br> compteur = <%= nomBean.setCompter(6) %>
<p>ou par une balise :<br>
<jsp:setProperty
  name="nomBean"property="compter" value="6" />
```

25

Un développement **efficient** est un développement **efficace**, c'est-à-dire qui atteint les objectifs du projet mais en plus **à moindre coût**, avec **moins de ressources** humaines et en **moins de temps** !

28

Bean Counter

```
package fsa.ac.ma;
import java.io.Serializable;
public class Counter implements Serializable{
 private  int count = 0;
  public Counter() {  }
  public int getCount() { count++; return count;  }
  public void setCount(int count) { this.count = count;  }}
```

```
//File: BeanCounter.jsp
<HTML> <HEAD> </HEAD>
<BODY>
<%@ page language="java" %>
<jsp:useBean id="counter" scope="session" class=" fsa.ac.ma.Counter" />
<jsp:setProperty name="counter" property="count"  value="6"  />
<%  out.println("Count from scriptlet code : " + counter.getCount() + "<BR>");%>
Count from jsp:getProperty :
  <jsp:getProperty name="counter" property="count" /><BR>
</BODY>
</HTML>
```

27

Couche Controller

```
/ / / / servlet controleur
package fsa.ac.ma;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import fsa.ac.ma.dto.User;
public class LoginServlet extends javax.servlet.http.HttpServlet
    implements javax.servlet.Servlet {
static final long serialVersionUID = 1L;
    public LoginServlet() {super();}
    protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException
    {String userId,password;
    userId= request.getParameter("userId");
    password= request.getParameter("password");
    LoginService loginService=new LoginService();
    boolean result=loginService.authenticate(userId, password);
if(result){
        User  user=loginService.getUserDetails(userId);
/ / request.getSession().setAttribute("user", user);
/ / utilisation de session scope pour trametttre l'objet user
        / / response.sendRedirect("success.jsp");
        request.setAttribute("user", user);
RequestDispatcher dispatcher=
        request.getRequestDispatcher("success.jsp");
        dispatcher.forward(request, response);
        return; }
        else {response.sendRedirect("login.jsp");
return;}}}
```

Exemple App MVC

Partie Vue (Input)

Login.jsp

```
<%@ page language="java" contentType="text/html; charset=windows-1256"
    pageEncoding="windows-1256"%>
<!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=windows-1256">
<title>Login Page< /title>
< /head>
<body>
<form action="login" method="post">
<br> User ID  : <input type="text" name="userId" / >
<br> Password : <input type="password" name="password" / >
<br> <input type="submit" / >
< /form>
< /body>
< /html>
```


Fichier déploiement Web.Xml

Web.xml

```
<?xml version="1.0"  encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http: / /java.sun.com/xml/ns/ j2ee "
        xmlns:xsi= "http: / /www.w3.org/ 2001/ XMLSchema-instance "
        xsi: schemaLocation= "http: / /java.sun.com/ xml/ns/ j2ee
        http: / /java.sun.com/ xml/ns/ j2ee/ web-app_ 2_ 4.xsd ">
    <display-name>
    LoginApps< / display-name>
    <servlet>
    <description>
    < / description>
    <display-name>LoginServlet< / display-name>
    <servlet-name>LoginServlet< / servlet-name>
<servlet-class>fsa.ac.ma.LoginServlet< / servlet-class>
    < / servlet>
    <servlet-mapping>
            <servlet-name>LoginServlet< / servlet-name>
            <url-pattern> /login< / url-pattern>
    < / servlet-mapping>
```

```
<welcome-file-list>
    <welcome-file>index.html< / welcome-file>
    <welcome-file>index.htm< / welcome-file>
    <welcome-file>index.jsp< / welcome-file>
    <welcome-file>default.html< / welcome-file>
    <welcome-file>default.htm< / welcome-file>
    <welcome-file>login.jsp< / welcome-file>
< / welcome-file-list>
< / web-app>
```


Exemple App MVC

```
/ / Couche Model ( couche métier)
package fsa.ac.ma;
import java.util.*;
import fsa.ac.ma.dto.User;
public class LoginService {
HashMap<String, String> users=new HashMap<String, String>();
    LoginService()
    {users.put("AFDEL", "AFDEL Karim");
    users.put("TAHA", "TAHA Mohamd");
    users.put("MOUSSA", "MOUSSA SADR");          }
    public boolean authenticate (String userId, String password)
    { if(password == null | | password.trim()== "")    {return
    false;      }
    return true;                }
    public String getUserName(String userId)
    {return users.get(userId); }
    public User getUserDetails(String userId)
    {User user=new User();
    user.setUserName(users.get(userId));
    user.setUserId(userId);
    return user;          }}
```

```
package fsa.ac.ma.dto;

public class User {
    private String userName;
    private String userId;
    public String getUserName() {
            return userName;
    }
    public void setUserName(String userName)
    {
            this.userName = userName;
    }
    public String getUserId() {
            return userId;
    }
    public void setUserId(String userId) {
            this.userId = userId;
    }

}
```

Partie Vue : Ouput

Success.jsp

```
<%@ page language="java" contentType="text/html; charset=windows-1256"
    pageEncoding="windows-1256" import="fsa.ac.ma.dto.User"%>
<!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=windows-1256">
<title>Success</title>
</head>
<body>
<h3> Login Successful </h3>
<%User user=(User)request.getAttribute("user");%>
Hello <%=user.getUserName() %>
</body>
</html>
```

33