Chapitre 3 : Java Server Pages (JSP)

Introduction

- JSP : Java Server Pages
- Java Server Pages est une technologie qui combine Java et des Tags HTML dans un même document pour produire un fichier JSP.
- But : faciliter la génération dynamique de contenu de sites Web.
- Similitudes : PHP, ASP, etc..

Exemple de fichier JSP

test.jsp

```
<html>
   <head>
                <title>Exemple JSP</title>
   </head>
   <body>
        la somme de 2 et 2 est <%=2+2%>
   </body>
</html>
```

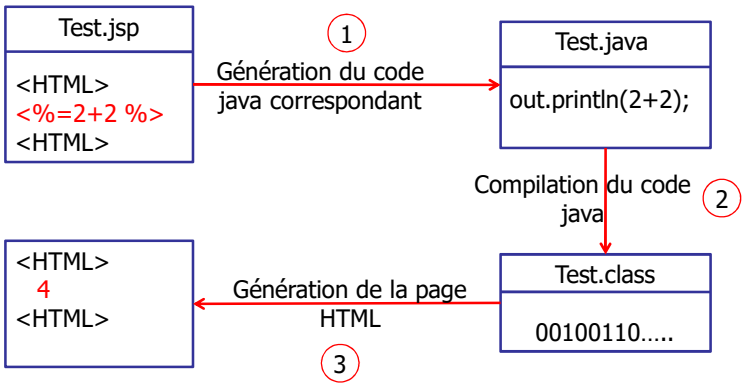

Serveur Web

- L'utilisation des JSP implique d'avoir un serveur HTTP (logiciel servant à diffuser les pages Web) disposant d'une extension capable de traiter les JSP.
- Exemple de serveurs HTTP gratuit supportant JSP:
  - **Tomcat** proposé par la fondation Apache.
  - **JSWDK** proposé par SUN
- Comme pour les servlets, nous travaillerons avec Tomcat.

Traitement des JSP

- L'interprétation d'une page contenant des instructions JSP se fait de la manière suivante :
  1. L' utilisateur demande, via son navigateur (client), un document possédant l'extension .jsp
  2. Le serveur HTTP lance une *servlet* (application Java serveur) qui construit le code Java à partir du code contenu dans la page HTML.
  3. Le programme résultant est compilé puis exécuté sur le serveur.
  4. Le résultat est réintroduit dans la page renvoyée au client.

Traitement des JSP



Structure d'un fichier JSP

- Similaire à la structure d'un fichier HTML
- Elle se compose essentiellement de quatre types de tags:
  - Tag de directive
  - Tag de commentaire
  - Tag de Scriplet
  - Tag d'expression

Directives JSP

- Les directives contrôlent comment le serveur WEB doit générer la Servlet
- Elles sont placées entre les symboles **<%@** et **%>**
- Syntaxe : <%@ directive { attribut="valeur"} %>
- Les directives les plus importantes sont:
  - **include** : indique au compilateur d'inclure un autre fichier
  - **page** : définit les attributs spécifiques à une page

Directives JSP : include

- Cette inclusion se fait au *moment de la conversion*
- Tout le contenu du fichier externe est inclus comme s'il était saisi directement dans la page JSP
- Pas de séparation de la portée des variables
- Exemple :
  - <%@ include file="unAutreFichier.jsp" %>

Exemple pratique

entete.html

```
<HTML>
<HEAD>
<TITLE>Page de démonstration</TITLE>
</HEAD>
<BODY>
  je suis dans l'entete de la page<br>
```

corps.jsp

```
<%! String mon_nom; %>
<% mon_nom = "Ali"; %>
```

piedpage.html

```
<br>Je suis dans le pied de page.
</BODY>
</HTML>
```

index.jsp

```
<%@ include file = "/entete.html" %>
<%@ include file = "/corps.jsp" %>
  Bonjour <%= mon_nom %>
<%@ include file = "/piedpage.html" %>
```

Directives JSP : page

- La directive **page** définit les attributs spécifiques à une page.
- La liste des attributs possibles pour la directive page est comme suit :

Attribut	exemple valeurs	Description
language	java	Indique le langage utilisé. Java par défaut
extends	Package.class	Hérite de l'interface du package choisi.
session	false	Si initialisé à false vous ne pouvez pas utiliser les sessions. True par défaut.
import	Java.util.*, *.class , java.*	Importe les classes dont vous avez besoin.
buffer	5 kb	Taille en kb de la mémoire tampon qui contient le flux de données à imprimer sur la JSP. 8 kb par défaut.
autoflush	true	Si à false il ne vide pas automatiquement le buffer une fois rempli. Si vous avez mis le buffer à none vous ne pouvez mettre autoFlush à false. Défaut à true.

Directives JSP : page

Attribut	Exemple de valeur	Description
isThreadSafe	false	Si à false le serveur d'applications ne permet qu'à un client à la fois d'accéder à la JSP. Défaut à true.
info	Ma première JSP	Information qui apparaît dans le document jsp compilé et utilisé par le serveur.
errorPage	Erreurpage.html Erreur.jsp	Adresse de la page d'erreur sur laquelle est renvoyé le visiteur en cas d'erreur (Exception) de la JSP.
contentType	text/html	Le type MIME et le jeu de caractères à utiliser dans cette JSP. Par défaut text/html; charset=ISO-8859-1
isErrorPage	true	Si true la JSP peut afficher l'erreur renvoyée par l'exception. True par défaut.
pageEnconding	ISO-8859-1	ISO-8859-1 par défaut.

Directives JSP : page

- Remarque :
  - Vous n'avez pas besoin d'importer les classes suivantes, qui le sont déjà implicitement:
    - java.lang.*
    - javax.servlet.*
    - javax.servlet.http.*
    - javax.servlet.jsp.*
- Exemple de directives :
  - <%@ page import=" java.util.*" errorPage=" erreur.jsp" buffer="5kb" session="false" %>

Éléments de scripts JSP : commentaire

- Cet élément de script est utilisé pour faire un commentaire dans le code JSP
- Le texte dans un commentaire JSP ne sera pas envoyé au client ni compilé dans la Servlet
- Les commentaires sont placés entre les symboles <%-- et --%>

```
Example.jsp:
<html>
  <!- commentaire HTML -->
  <%-- commentaire JSP --%>
</html>
```

```
<html>
  <!-- commentaire HTML -->

</html>
```

Éléments de scripts JSP : déclaration

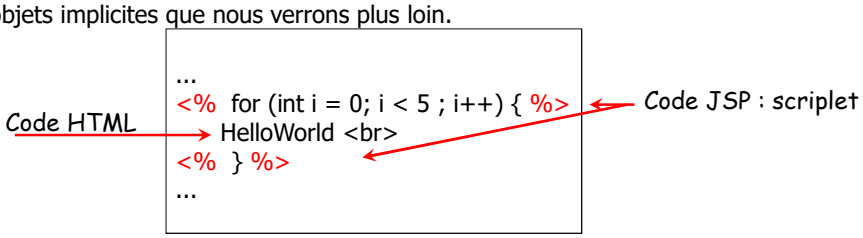
- Une déclaration permet d'insérer du code dans la classe de la Servlet
- Les déclarations sont placés entre les symboles <%! et %>
- Exemple:

```
<%!
   private int count = 0;
   private int incrementCount() { return count++;}
%>
```

- Elle peut être utilisée pour :
  - Déclarer un attribut de classe
  - Spécifier et implémenter des méthodes
  - Les attributs et les méthodes déclarées dans la page JSP sont utilisables dans toute la page JSP

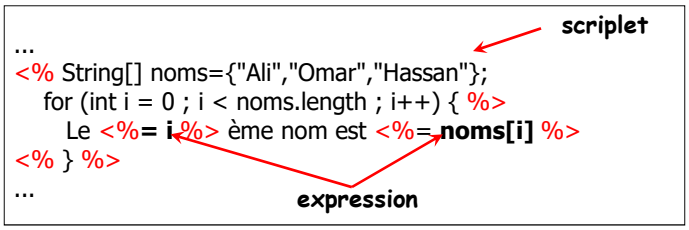
Éléments de scripts JSP : scriplet

- C'est un bloc de code Java qui est placé dans *_jspService(…)* de la Servlet générée (équivalent à *service(...)*)
- Les scriplets sont placées entre les symboles **<%** et **%>**
- Tout code java a accès :
  - aux attributs et méthodes définis par le tag déclaration **<%!** … **%>**
  - aux objets implicites que nous verrons plus loin.



Éléments de scripts JSP : expression

- Sert à évaluer une expression et à renvoyer sa valeur
- Les expressions sont placées entre les symboles <%= et %>
- Retourne une valeur String de l'expression
- Correspond à une scriplet de la forme *<% out.println(…); %>*
- Se transforme en *out.println("…");* dans la méthode *_jspService(…)* après génération



Éléments de scripts JSP : scriplet et objets implicites

- Les objets implicites sont les objets présents dans la méthode *service(…)* qui ont été employés dans la partie Servlet
- Ils sont identifiés par des noms de variables uniques :
  - **request** : requête courante
  - **response** : réponse courante
  - **session** : session courante
  - **out** : flot de sortie permet l'écriture sur la réponse
  - **application** : contient des méthodes log() permettant d'écrire des messages dans le journal du contenu (*ServletContext*)
  - **pageContext** : utilisé pour partager directement des variables entre des pages JSP et supportant les beans et les balises
  - **exception** : disponible uniquement dans les pages erreurs donnant information sur les erreurs

Éléments de scripts JSP : scriplet et objets implicites

- Exemple : JSP qui récupère des informations du client

```
<%@ page language="java" contentType="text/html" %>
<html>
<head>
<title>Informations client</title>
</head>
<body bgcolor="white">
  Protocol : <%= request.getProtocol() %><br>
  Scheme : <%= request.getScheme() %><br>
  ServerName : <%= request.getServerName() %><br>
  ServerPort : <% out.println(request.getServerPort()); %><br>
  RemoteAddr : <% out.println(request.getRemoteAddr()); %><br>
  RemoteHost : <% out.println(request.getRemoteHost()); %><br>
  Method : <%= request.getMethod() %><br>
</body>
```


Cycle de vie d'une JSP

- Le cycle de vie d'une Java Server Page est identique à une Servlet :
  - La méthode *jspInit()* est appelée après le chargement de la page
  - La méthode _jspService() est appelée à chaque requête
  - La méthode *jspDestroy()* est appelé lors du déchargement (fermeture d'une base de données)
- Possibilité de redéfinir dans le code JSP les méthodes *jspInit()* et *jspDestroy()* en utilisant un élément de scripts déclaration
- Exemple :

```
<html> <head><title>Bonjour tout </title></head><body>
<%! int compteur; %>
<%! public void jspInit() {
      compteur = 0;} %>
   La valeur du compteur est <%= compteur++ %>
</body></html>
```

Cycle de vie d'une JSP

- **Exemple** : un compteur avec une initialisation et une destruction

```
<%@ page language="java" contentType="text/html" %>
<%@ page import="java.util.Date" %>
<%!
int global_counter = 0;
Date start_date;
public void jspInit() {
start_date = new Date();
}
public void jspDestroy() {
ServletContext context = getServletConfig().getServletContext();
context.log("test.jsp a été visitée " + global_counter + "fois entre le
" + start_date + " et le " + (new Date()));
}
%>
<html>
<head><title>Page avec un compteur</title></head>
<body bgcolor="white">
Cette page a été visitée : <%= ++global_counter %> fois depuis le <%=
start_date %>.
</body></html>
```


JSP et Actions

- Les actions 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
- Les actions sont ajoutées à la page JSP à l'aide d'une syntaxe d'éléments XML (définis par des balises personnalisées). Exemple :
  - <ma:balise ... />
  - <ma:balise ... > ...</ma:balise>

Java Beans

- Permet de coder la logique métier de l'application WEB
- L'état d'un Bean est décrit par des attributs appelés propriétés
- La spécification des Java Beans définit les Beans comme des classes qui supportent
  - Introspection : permet l'analyse d'un Bean (nombre de propriétés)
  - Événements : métaphore de communication
  - Persistance : pour sauvegarder l'état d'un Bean
  - ...

Java Beans

- Les Java Beans sont des classes Java normales respectant un ensemble de directives
  - A un constructeur public sans argument
  - Les propriétés d'un Bean sont accessibles au travers de méthodes *getXXX* (lecture) et *setXXX* (écriture) portant le nom de la propriété
- Lecture et écriture des propriétés
  - *type getNomDeLaPropriété()* : pas de paramètre et son type est celui de la propriété
  - *void setNomDeLaPropriété(type)* : un seul argument du type de la propriété et son type de retour est void
- En option, un Java Beans implémente l'interface *java.io.Serializable* permettant la sauvegarde de l'état du Bean

Exemple : classe Client

```
public class Client {
    // attributs
     private String nom;
     private String adresse;
    //méthodes d'accès et de modification
     public String getNom () { return nom;              }
     public void setNom (String nm) { nom=nm;           }
     public String getAdresse () { return adresse;      }
     public void setAdresse (String adr) { adresse=adr; }
}
```


Java Beans et JSP

- Pour déclarer et allouer un Java Beans dans une page JSP il faut employer l'action <jsp:useBean>
- Exemple :
- <jsp:useBean **id**="nomObjet" **class**="Package.nomClasse" **scope**=" ***page*** ou ***request*** ou ***session*** ou ***application*** " />
  - id : nom de l'instance pour identification
  - class : Nom de la classe du bean

Java Beans et JSP

- scope : portée de la validité de l'objet Bean :
  - *page* : Bean valide pour la requête sans transmission
  - *request* : Bean valide pour la requête et peut être transmise (forward)
  - *session* : Bean ayant la durée de vie de la session
  - *application* : Bean créée pour l'application WEB courante

Java Beans et JSP : lecture propriétés

- Pour lire une propriété du Bean deux éléments sont utilisés
  - La référence du Bean définie par l'attribut id
  - Le nom de la propriété
- Deux manières existent pour interroger la valeur d'une propriété et la convertir en String
  - En utilisant un tag action *<jsp:getProperty>*
    - *Syntaxe:* <jsp:getProperty name="référence Bean" property="nom propriété" />
  - En utilisant l'élément de scripts JSP : expression
    - <%= nom_instance.getNomPropriete() %>

Java Beans et JSP : écriture propriétés

- Modification de la valeur d'une propriété en utilisant *<jsp:setProperty>*
- Syntaxe:
  - <jsp:setProperty name="référence Bean" property="nom propriété" value="valeur" />
  - <jsp:setProperty name="référence Bean" property="nom propriété" param="nomParametre" />

Java Beans et JSP : lecture et écriture propriétés

- Exemple : Soit le java bean suivant :

```
package toto;
public class Client {
      private String nom;
      private String adresse;
      public String getNom () { return nom;                }
       public void setNom (String nm) { nom=nm;            }
       public String getAdresse () { return adresse;       }
       public void setAdresse (String adr) { adresse=adr;  }
}
```

Attention :
les classes des beans doivent être mise dans le répertoire WEB-INF/classes

Java Beans et JSP : lecture et écriture propriétés

- Exemple d'utilisation du bean précédent:

```
<jsp:useBean id="cl" class=" toto.Client"/>
<%  cl.setNom("Ali");
     cl.setAdresse("31, Bd des FAR, Casablanca");
%>
<html>
     <head>
        <title>Page pour lecture d'information</title>
     </head>
     <body bgcolor="white">
        Nom du Client : <%= cl.getNom() %><br>
         Adresse du Client : <jsp:getProperty name="cl" property="adresse"/>
     </body>
</html>
```


Java Beans et JSP : lecture et écriture propriétés

- Modification de l'ensemble des propriétés :
  - Exemple : <jsp:setProperty name="référence" property="*" />
  - Condition : Les noms des paramètres de requête doivent être identiques aux noms des propriétés

Java Beans et JSP : lecture et écriture propriétés

- Exemple : Soit le fichier « index.html » suivant

```
<!DOCTYPE html>
<html>
    <head>
        <title>Formulaire</title>
    </head>
    <Form method="GET" action="afficher.jsp">
        CNE:<input type="text" name="cne"/><br>
        Nom:<input type="text" name="nom"/><br>
        Prénom:<input type="text" name="prenom"/><br>
        <input type="submit" value="afficher">
    </Form>
</html>
```


Java Beans et JSP : lecture et écriture propriétés

- Bean Etudiant

```
package toto;
public class Etudiant {
    private String cne;
    private String nom;
    private String prenom;
    public Etudiant() { }
    public String getCne() { return cne;}
    public void setCne(String cne) { this.cne = cne;}
    public String getNom() { return nom;}
    public void setNom(String nom) { this.nom = nom;}
    public String getPrenom() {return prenom;}
    public void setPrenom(String prenom) { this.prenom = prenom;}
}
```

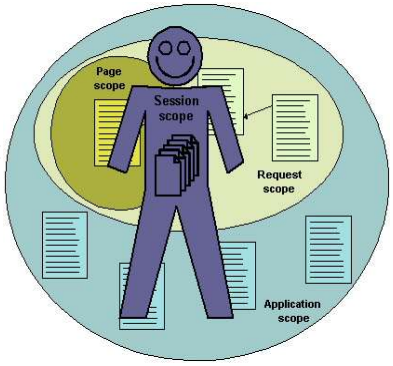
Java Beans et JSP : lecture et écriture propriétés

- Fichier « afficher.jsp »

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<jsp:useBean id="etudiant" class="toto.Etudiant"/>
<jsp:setProperty name="etudiant" property="*" />
<!DOCTYPE html>
<html>
    <head>
        <title>afficher.jsp</title>
    </head>
    <body>
        CNE étudiant : <jsp:getProperty name="etudiant" property="cne"/><br>
        Nom étudiant : <jsp:getProperty name="etudiant" property="nom"/><br>
        Prénom étudiant : <jsp:getProperty name="etudiant" property="prenom"/><br>
    </body>
</html>
```


Java Beans et JSP : scope

- Toute variable dans une page JSP a une portée
- Il y a 4 types de portées :
  - Portée **Page** : la variable n'est reconnue qu'au sein de la page
  - Portée **Request** : la variable est reconnue là ou la requête est partagée
  - Portée **Session** : la variable est reconnue tant que la session de l'utilisateur est reconnue
  - Portée **Application** : la variable est reconnue dans toute l'application quelque soit la page, quelque soit la requête, quelque soit l'utilisateur.



Java Beans et JSP : scope

- **Exemple** : affectation et récupération des valeurs d'un Java Bean
- Soit le java bean suivant :

```
Package toto;

public class TestBean{

    private String contenu;

    public String getContenu () {  return contenu;  }

    public void setContenu (String c) {  contenu=c; }

}
```


Java Beans et JSP : scope

- Utilisation du bean avec différentes portées dans une première JSP.

```
<jsp:useBean id="b1" scope="page" class="toto.TestBean"/>
<jsp:useBean id="b2" scope="session" class="toto.TestBean">
<jsp:useBean id="b3" scope="application" class="toto.TestBean"/>
<jsp:setProperty name="b1" property= "contenu" value="page"/>
<jsp:setProperty name="b2" property= "contenu" value="session"/>
<jsp:setProperty name="b3" property= "contenu" value="application"/>
<html>
<head><title> Utilisation du bean </title></head>
<body bgcolor="white">
      Avant<br>
      b1 = <%= b1.getContenu() %><br>
      b2 = <%= b2.getContenu() %><br>
      b3 = <%= b3.getContenu() %><br>
      <form method=GET action="lecture.jsp">
      <p align="center"><input type="submit" name="Submit"></p>
      </form>
</body>
</html>
```

Java Beans et JSP : scope

- Récupération du bean avec différentes portées dans une deuxième JSP.

```
<jsp:useBean id="b1" scope="page" class="toto.TestBean"/>
<jsp:useBean id="b2" scope="session" class="toto.TestBean"/>
<jsp:useBean id="b3" scope="application" class="toto.TestBean"/>
<html><head><title> Récupération du bean </title></head>
    <body bgcolor="white">
     Après<br>
     b1 = <jsp:getProperty name="b1" property="contenu"/><br>
     b2 = <jsp:getProperty name="b2" property="contenu"/><br>
     b3 = <jsp:getProperty name="b3" property="contenu"/><br>
    </body>
</html>
```


Java Beans et JSP : scope

- Les résultats de ces deux jsp est comme suit :
  - Avant
    - b1 = page
    - b2 = session
    - b3 = application
  - Après
    - b1 =null
    - b2 = session
    - b3 = application

Collaboration de JSP

- Rappel sur la collaboration (voir partie Servlet)
  - partage d'information : un état ou une ressource
  - partage du contrôle : une requête
- Processus identique à la collaboration de Servlet pour le partage d'information et de contrôle
- Partage d'information :
  - Utilisation du contexte pour transmettre des attributs
  - Méthode *getContext(…)*, *setAttribute(…)* et *getAttribute(…)*
- Partage du contrôle : Utilisation des tags action JSP *include* et *forward*

Partage d'information

- Le partage se fait grâce à l'objet implicite application qui est de type *ServletContext*
- Exemple : transmettre un simple attribut à tout un contexte
  - Page1.jsp :

    Enregistrement dans le contexte d'un attribut

    <% application.setAttribute("attribut1","Bonjour tout le monde"); %>
  - Page2.jsp :
    - <% out.println(application.getAttribute("attribut1")); %>

Partage du contrôle

- forward:
  - Exemple1 : renvoi sans passage de paramètres
    ```
    <jsp:forward page="page.html" />
    ```
  - Exemple2 : renvoi avec passage de paramètres
    ```
    <jsp:forward page="page.html" >
      <jsp:param name="defaultparam" value="nouvelle" />
    </jsp:forward>
    ```
  - Remarque :
    - ne pas modifier l'objet response
    - Ne pas modifier l'objet request après le renvoi
- Include :
  - Exemple1 : inclusion sans passage de paramètres
    ```
    <jsp:include page="page.html" />
    ```
  - Exemple2 : inclusion avec passage de paramètres
    ```
    <jsp:include page="page.html" >
      <jsp:param name="defaultparam" value="nouvelle" />
    </jsp:include>
    ```


Partage du contrôle

- Remarques :
  - Le partage du contrôle et plus précisément l'inclusion et le renvoi par des balises actions ne permettent que le transfert d'attributs de types chaînes de caractères.
  - Nécessité d'utiliser *RequestDispatcher* et les objets implicites request et response pour transférer des attributs objets
  - Exemple pour une inclusion (même chose pour un renvoi)

```
<% RequestDispatcher dispatch = request.getRequestDispatcher("/fichier.jsp");%>

<% request.setAttribute("attribut1","bonjour"); %>

<% dispatch.include(request,response); %>
```