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..
- Comme pour les servlets, nous continuerons de travailler avec Tomcat.

Exemple de fichier JSP

test.jsp

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

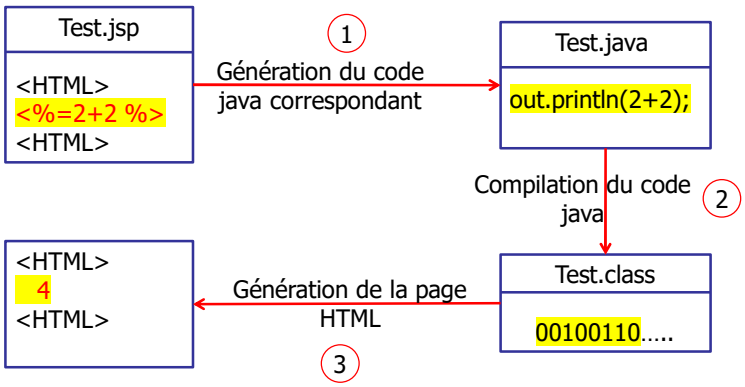
```
...
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    ...
    out.println("<!DOCTYPE html>
    <html>
        <head>
            <title> Exemple JSP</ title >
        </ head>
        <body>
            la somme de 2 et 2 est "+ 2+2+
        "</body>
    </html>");
    ...
}
```

Remarque : Une page JSP est transformée en Servlet avant d'être exécutée

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
- Et utilise les Objets Implicites suivants (sont déclarés dans la Servlet générée) :
  - **request** : requête courante
  - **response** : réponse courante
  - **session** : session courante
  - **out** : flot de sortie permet l'écriture sur la réponse
  - **application** : correspond au *ServletContext (pas de référence aux objets request, response, …)*
  - **pageContext** : est une alternative à l'objet application mais gardant en plus références aux autres objets implicites de la servlet
  - **exception** : disponible uniquement dans les pages erreurs donnant information sur les erreurs
- Ces objets permettent l'utilisation de toutes les méthodes que nous avons vu dans les servlets

Structure d'un fichier JSP

- Une page JSP se compose essentiellement de quatre types de tags:
  - Tag de directive : **<%@** texte **%>**
  - Tag de commentaire : **<%--** texte **--%>**
  - Tag de déclaration :**<%!** texte **%>**
  - Tag de Scriplet : **<%** texte **%>**
  - Tag d'expression : **<%=** texte **%>**

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:
  - **page** : définit les attributs spécifiques à une page
  - **include** : indique au compilateur d'inclure un autre fichier

Directives JSP : page

- La directive **page** définit les attributs spécifiques à une page.
- Exemple :

```
<%@ page import=" java.util.*" errorPage=" erreur.jsp"  buffer="5kb" session="false"  %>
```

- Remarque :
  - Vous n'avez pas besoin d'importer les classes suivantes, qui le sont déjà implicitement:
    - java.lang.*
    - jakarta.servlet.*
    - jakarta.servlet.http.*
    - jakarta.servlet.jsp.*

Directives JSP : 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.*,	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 : 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" %>
```


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 --%>

```
<html>
  <!- commentaire HTML -->
  <%-- commentaire JSP --%>
</html>
```

⇨

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

</html>
```

Éléments de scripts JSP : déclaration

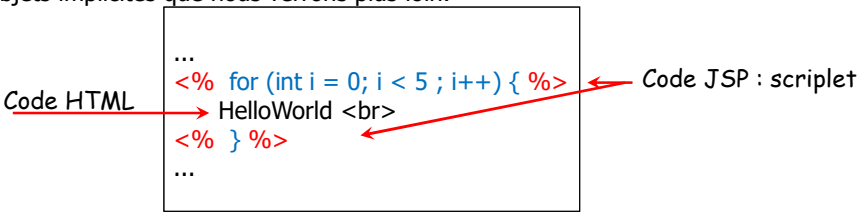
- Une déclaration permet d'insérer du code dans la classe de la Servlet
- Les déclarations sont placées entre les symboles <%! et %>
- Exemple:

```
<%!
   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

```
...
<% String[] noms={"Ali","Omar","Hassan"};
   for (int i = 0 ; i < noms.length ; i++) { %>
      Le <%= i %> ème nom est <%= noms[i] %>
<% } %>
...
```

scriplet

expression

afficher

Exercice 1

- Ecrire une JSP qui affiche un tableau de multiplication lu à partir d'un formulaire

Nombre : [ ] afficher ← « index.html »

Exemple pour 6 ←

6 x 0	=	0
6 x 1	=	6
6 x 2	=	12
6 x 3	=	18
6 x 4	=	24
6 x 5	=	30
6 x 6	=	36
6 x 7	=	42
6 x 8	=	48
6 x 9	=	54

« tabMultiplication.jsp »

Solution : index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>index</title>
  </head>
  <body>
    <FORM method="post" action="tabMultiplication.jsp">
        Nombre : <INPUT type="text" name="nombre">
        <INPUT type="submit" value="afficher">
    </FORM>
  </body>
</html>
```


Solution : tabMultiplication.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
  <head> <title>Table de Multiplication</title></head>
  <body>
    <% String nombreF=request.getParameter("nombre");
      int nombre=Integer.parseInt(nombreF);//Exception non prise en charge
    %>
    <table border="1">
    <% for (int i = 0; i < 10; i++) {%>
        <tr>
            <td width="50" align="center"><%=nombre%> x <%=i%></td>
            <td width="50" align="center"> = </td>
            <td width="50" align="center"> <%=nombre*i%> </td>
        </tr>
    <%}%>
    </table>
  </body>
</html>
```

Exercice 2

- Construire un formulaire HTML comprenant les informations suivantes :
  - Nom (zone de texte)
  - Prénom (zone de texte)
  - Sexe (boutons radio M ou F)
  - Fonction (options)
    - Enseignant
    - Étudiant (choix initial)
    - Ingénieur
    - Retraité
    - Autre
  - Loisirs (cases à cocher)
    - Lecture
    - Sport
    - Voyage
  - Commentaire (textarea)
- Ecrire une page JSP qui récupère et affiche:
  - Method d'envoi du client
  - Adresse IP du client
  - Les données saisis par l'utilisateur

Solution 1/2

Index.html

```
<!DOCTYPE html>
<HTML>
 <HEAD> <title> formulaires et servlets </title></HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<BODY>
    <FORM method="post" action="affichage.jsp">
     Enregistrement d'un utilisateur : <br>
      Nom : <INPUT type="text" name="nom"> <br>
      Prénom :<INPUT type="text" name="prenom"><br>
      Sexe : <br> <INPUT type="radio" name="sexe" value="M" checked>Homme<br>
            <INPUT type="radio" name="sexe" value="F">Femme<br>
      Fonction :<SELECT name="fonction">
                        <OPTION VALUE="enseignant">Enseignant</OPTION>
                        <OPTION VALUE="etudiant">Etudiant</OPTION>
                        <OPTION VALUE="ingenieur" selected>Ingénieur</OPTION>
                        <OPTION VALUE="retraite">Retraité</OPTION>
                        <OPTION VALUE="autre">Autre</OPTION>
              </SELECT> <br>
      Loisirs : <br><INPUT type="checkbox" NAME="loisirs" value="lecture" CHECKED>Lecture <br>
             <INPUT type="checkbox" NAME="loisirs" value="sport">Sport <br>
             <INPUT type="checkbox" NAME="loisirs" value="voyage">Voyage<br>
      Commentaire :<br><TEXTAREA rows="3" name="commentaire">Votre Commentaire</TEXTAREA><br>
    <INPUT type="submit" value="Envoyer">
    </FORM>
 </BODY>
</HTML>
```

Solution 2/2

affichage.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
   <head> <title>Contenu Formulaire</title></head>
   <body>
    <ul> <li>Method d'envoi du client : <%=request.getMethod()%></li>
          <li>Adresse IP du client : <%=request.getRemoteAddr()%></li>
          <li>Nom :<%=request.getParameter("nom")%></li>
          <li>Prénom : <%=request.getParameter("prenom")%></li>
          <li>Sexe : <%=request.getParameter("sexe")%></li>
          <li>Fonction : <%=request.getParameter("fonction")%></li>
          <li>Commentaire : <%=request.getParameter("commentaire")%></li>
          <li>Loisirs : <%     String[] valeursDeLoisirs = request.getParameterValues("loisirs");
                    for (int i = 0; i < valeursDeLoisirs.length; i++) {  out.println(valeursDeLoisirs[i]);  }  %></li>
    </ul>
   </body>
</html>
```


Cycle de vie d'une JSP

- Le cycle de vie d'une Java Server Page est identique à une Servlet.
- Cependant ses méthodes commence avec « jsp » :
  - 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)
- **Remarque** :
  - Il est possible de redéfinir les méthodes *jspInit()* et *jspDestroy() dans les blocs de* déclaration JSP

Exercice 3

- En s'appuyant sur la méthode jspInit() créer une page JSP qui compte le nombre de ses visites
- Instructions :
  - Définir un attribut compteur que vous devez initialiser par 0 dans la méthode jspInit() et incrémenter dans la page JSP.

Solution

```
<%@page contentType="text/plain" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
  <head>
    <title>Compteur de visites</title>
  </head>
  <body>
    <%!
      int compteur;
      public void jspInit() { compteur = 0; }
    %>
    La valeur du compteur est <%= compteur++ %>
  </body>
</html>
```

Exercice 4

- En s'appuyant sur les méthode jspInit() et jspDestroy(), écrire une page JSP qui compte et affiche le nombre de visites depuis la date de son démarrage.
- Une fois arrêtée, elle doit écrire le nombre de visites entre la date de son démarrage et la date de son arrêt dans un fichier journal grâce à la méthode log() de l'objet context.
- Instructions :
  - Utiliser un attribut compteur et et un autre début de type Date que vous devez initialiser dans la méthode jspInit()
  - Utilisez l'instruction « ServletContext context = getServletConfig().getServletContext() » pour récupérer le contexte.

Solution

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

Java Beans

- Les Java Beans sont des classes Java normales respectant un ensemble de directives
  - A un constructeur public sans argument
  - Des getters et setters sont appelés propriétés pour lire et modifier ses attributs
- Les Java Beans Permettent de coder la logique métier de l'application WEB en provenance de formulaires ou bases de données
- 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;
    // constructeur par défaut
     public Client(){}
    //méthodes d'accès et de modification
     public String getNom () { return nom; }
     public void setNom (String nom) { this.nom= nom;}
     public String getAdresse () { return adresse; }
     public void setAdresse (String adresse) { this.adresse= adresse;   }
}
```

Java Beans et JSP

- Syntaxe de déclaration d'un Java Bean :
- <jsp:useBean **id**="nomObjet" **class**="Package.nomClasse" **scope**="portée" />
  - id : nom de l'instance pour identification
  - class : Nom de la classe du bean
  - 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="nomObjet" property="nomPropriété" />
  - En utilisant l'élément de scripts JSP : expression
    - <%= nomObjet.getNomPropriéte() %>

Java Beans et JSP : écriture propriétés

- Modification de la valeur d'une propriété en utilisant *<jsp:setProperty>*
- Syntaxe:
  - <jsp:setProperty name="nomObjet" property="nomPropriété" value="valeur" />
  - <jsp:setProperty name="nomObjet" property="nomPropriété" param="nomParametre" />
- **Remarque** :
  - La valeur affectée dans le deuxième exemple correspond à un paramètre qui vient avec la requête (provenant d'un formulaire par exemple)

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

Remarque : les java beans sont placé dans le dossier Java comme les Servlets

Java Beans et JSP : lecture et écriture propriétés

- Exemple d'utilisation du bean précédent:

```
<jsp:useBean id="cl" class=" toto.Client"/>
<jsp:setProperty name="cl" property= "nom" value="Ali"/>
<jsp:setProperty name="cl" property="adresse" value="31, Bd des FAR, Casablanca"/>
<html>
    <head>
        <title>Page pour lecture d'information</title>
    </head>
    <body>
        Nom du Client : <jsp:getProperty name="cl" property="nom"/> <br>
        Adresse du Client : <jsp:getProperty name="cl" property="adresse"/>
    </body>
</html>
```

✓ **Remarque**: nous pouvons aussi utiliser `<%= cl.getNom() %>` et `<%= cl.getAdresse() %>`

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

Exercice 5

- Ecrire un Bean Etudiant composé des attributs :
  - cne
  - nom
  - prenom
- Ecrire une page « index.html » qui est un formulaire permettant de récupérer les valeurs saisies correspondant aux cne, nom, prenom
- Ecrire une page JSP « affichage.jsp » qui crée un bean « etudiant » et l'initialise à partir du formulaire et qui affiche ensuite les propriétés du bean.

Solution

- 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>
```

Solution

- 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;}
}
```

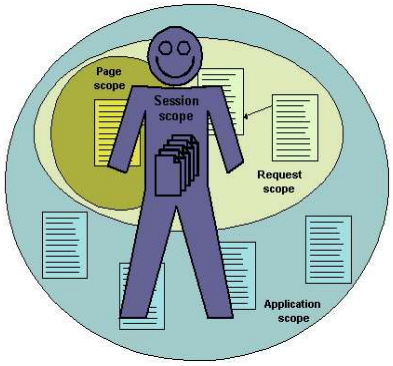

Solution

- 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 Boite{
    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.Boite"/>
<jsp:useBean id="b2" scope="session" class="toto.Boite ">
<jsp:useBean id="b3" scope="application" class="toto.Boite"/>
<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>
      Avant<br>
      b1 = <%= b1.getContenu() %><br>
      b2 = <%= b2.getContenu() %><br>
      b3 = <%= b3.getContenu() %><br>
      <form method="GET" action="lecture.jsp">
          <input type="submit" name="Submit">
      </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.Boite "/>
<jsp:useBean id="b2" scope="session" class="toto.Boite ">
<jsp:useBean id="b3" scope="application" class="toto.Boite "/>
<html>
<head><title> Récupération du bean </title></head>
<body>
      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("msg","Bonjour tout le monde"); %>
    ```

  - Page2.jsp :
    - ```
      <%=application.getAttribute("msg") %>
      ```


Partage du contrôle avec Forward

- Forward sans passage de paramètres à la requête partagée :

  ```
  <jsp:forward page="afficher.jsp" />
  ```

- Forward avec passage de paramètres à la requête partagée (ne permet que des chaines de caractères seulement):

  ```
  <jsp:forward page="afficher.jsp" >
    <jsp:param name="nom" value="Alaoui" />
  </jsp:forward>
  ```

- La récupération dans la deuxième JSP est faite avec :
  - request.getParameter("nom")
  ```

Partage du contrôle avec Forward

- Forward avec passage d'attributs à la requête partagée (permet des objets):

```
<% RequestDispatcher dispatch = request.getRequestDispatcher("/afficher.jsp");
   request.setAttribute("et",new Etudiant(1,"Alaoui", "Ali"));
   dispatch.forward(request,response); %>
```

- La récupération dans la deuxième JSP est faite avec :
  - request.getAttribute("et")

Partage du contrôle avec Include

- Include sans passage de paramètres à la requête partagée :

```
<jsp:include page="afficher.jsp" />
```

- Include avec passage de paramètres à la requête partagée (chaines de caractères seulement):

```
<jsp:include page="afficher.jsp" >
  <jsp:param name="nom" value="Alaoui" />
</jsp:include>
```

- La récupération dans la deuxième JSP est faite avec :
  - request.getParameter("nom")

Partage du contrôle avec Include

- Include avec passage d'attributs à la requête partagée (permet des objets):

```
<% RequestDispatcher dispatch = request.getRequestDispatcher("/afficher.jsp");
request.setAttribute("et",new Etudiant(1,"Alaoui", "Ali"));
dispatch.include(request,response); %>
```

- La récupération dans la deuxième JSP est faite avec :
  - request.getAttribute("et")

Exercice 6

- Créer un formulaire (index.html) contenant un login et mot de passe et un bouton envoyer
- Le formulaire mène vers une Servlet (VerificationServlet) contenant 3 paramètres d'initialisations dans le fichier « web.xml » qui sont : nom, login et password.
- La servlet récupère le login et le mot de passe et le compare avec le login et le mot de passe qu'elle stocke.
- S'ils correspondent alors, elle met le nom dans la requête et le transfère à la page « affichage.jsp » et qui l'affiche avec un message de bien venu
- S'il y a erreur alors elle affiche un message d'erreur

Solution : Fichier « index.html »

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <form method="GET" action="VerificationServlet">
     Login    : <input type="text" name="login"/><br/>
     Password :<input type="password" name="password"/>
              <input type="submit" value="Log in"/>
</form>
</body>
</html>
```


Solution : Servlet « VerificationServlet.java »

```
package tp;
//les imports cachés
public class VerificationServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  private String nom;
  private String login;
  private String password;
  @Override
  public void init() throws ServletException {
     nom = this.getInitParameter("nom"); login = this.getInitParameter("login"); password = this.getInitParameter("password");
  }
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
     String log = request.getParameter("login")  , pass = request.getParameter("password");
     if (login.equals(log) && password.equals(pass))    request.setAttribute("nom", nom);
     RequestDispatcher dispat = request.getRequestDispatcher("affichage.jsp");
     dispat.forward(request, response);
  }
  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    doGet(request, response);}
}
```

Solution : Fichier « web.xml »

```
...
<servlet>
        <servlet-name>Verification</servlet-name>
        <servlet-class>tp.VerificationServlet</servlet-class>
        <init-param>
                <param-name>nom</param-name>
                <param-value>Ali Alaoui</param-value>
        </init-param>
        <init-param>
                <param-name>login</param-name>
                <param-value>exo5</param-value>
        </init-param>
        <init-param>
                <param-name>password</param-name>
                <param-value>exo5</param-value>
        </init-param>
</servlet>
...
```


Solution

- Fichier « affichage.jsp »

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
 <head>
    <title>affichage</title>
 </head>
 <body>
  <% String nom = (String) request.getAttribute("nom");
      if (nom != null) { %>
      <h2> Bievenu <%=nom%></h2>
      <% } else { %>
      <h2>Erreur login ou mot de passe</h2>
      <% } %>
 </body>
</html>
```