TP architecture MVC : corrigé

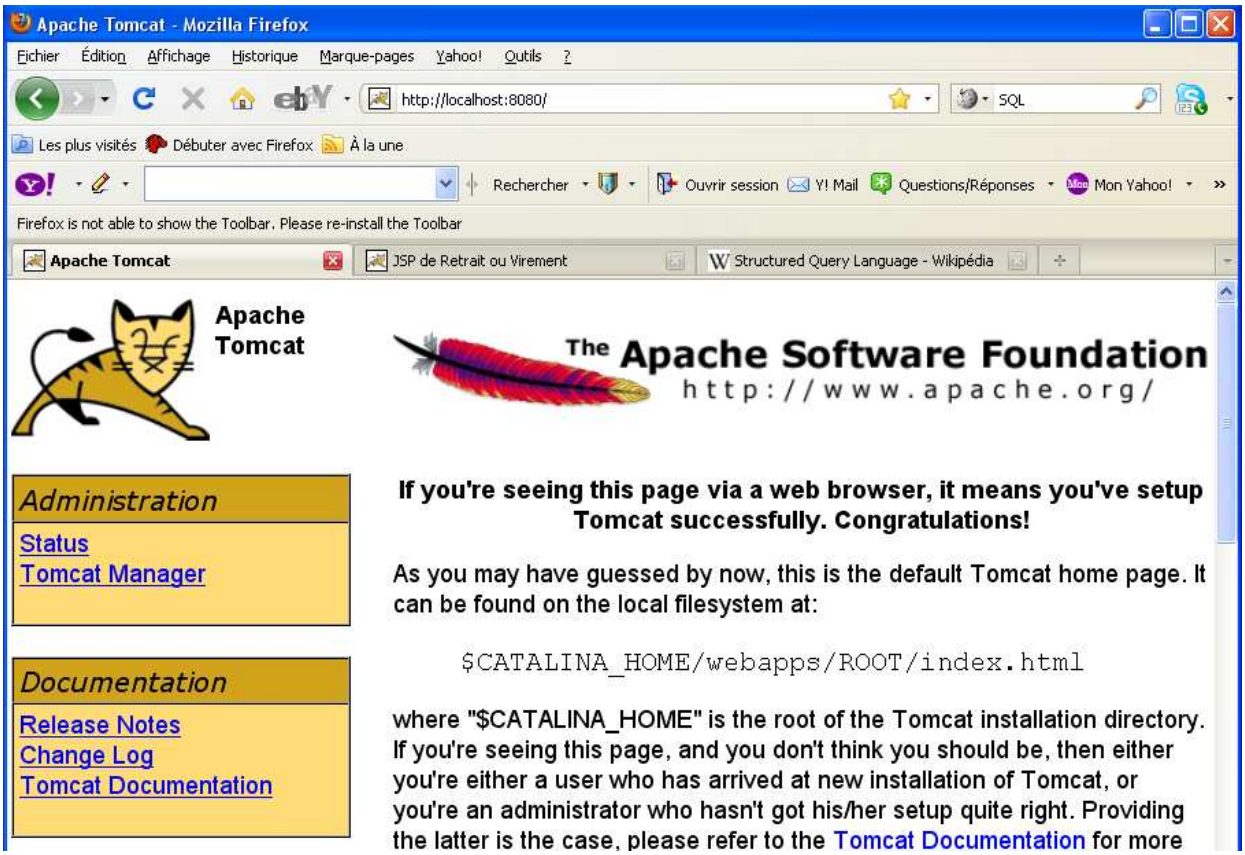
La principale difficulté de ce TP est de bien "ranger les choses". Il faut être cohérent quant à l'endroit où les divers fichiers sont mis dans l'archive construite, ainsi que certaines lignes du fichier `web.xml`. Bref il y a là tout autant un travail de développeur que de personne qui déploie.
Les parties ci dessous écrite en vert, gras sont les parties pour une correction.

0°) Préliminaire

a) Créer un dossier dans lequel vous travaillerez pour tout ce TP : ce sera votre répertoire de travail. Recopier tout l'arborescence de tomcat (qui se trouve peut être sous `C:\Program Files\ApacheTomcat\6.0.18`) dans un répertoire éditable par exemple sous `C:\www\tomcat`.
b) Créer le fichier script `lanceTomcat.bat`, qui va permettre de lancer le serveur tomcat.
fichier `lanceTomcat.bat`

```
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_11
set CATALINA_HOME=C:\www\tomcat
%CATALINA_HOME%\bin\startup.bat
```

L'exemple ci dessus a supposé que le JDK (pas le JRE) Java est rangé sous `C:\Program Files\Java\jdk1.6.0_11`, et tomcat est installé sous `C:\www\tomcat`. (au besoin il a été recopié sous ce répertoire). Ajuster les deux variables `JAVA_HOME` et `CATALINA_HOME` correctement en fonction des configuration de vos machines.
Sauvegarder ce script dans votre répertoire de travail.
c) Ouvrer une fenêtre dos dans votre répertoire de travail et lancer `lanceTomcat.bat`.
d) Ouvrer un navigateur et connecter vous à l'URL `http://localhost:8080`. Vous devriez avoir une fenêtre comme :



e) Créer le fichier script `arreteTomcat.bat`, qui va permettre de lancer le serveur tomcat.
fichier `arreteTomcat.bat`

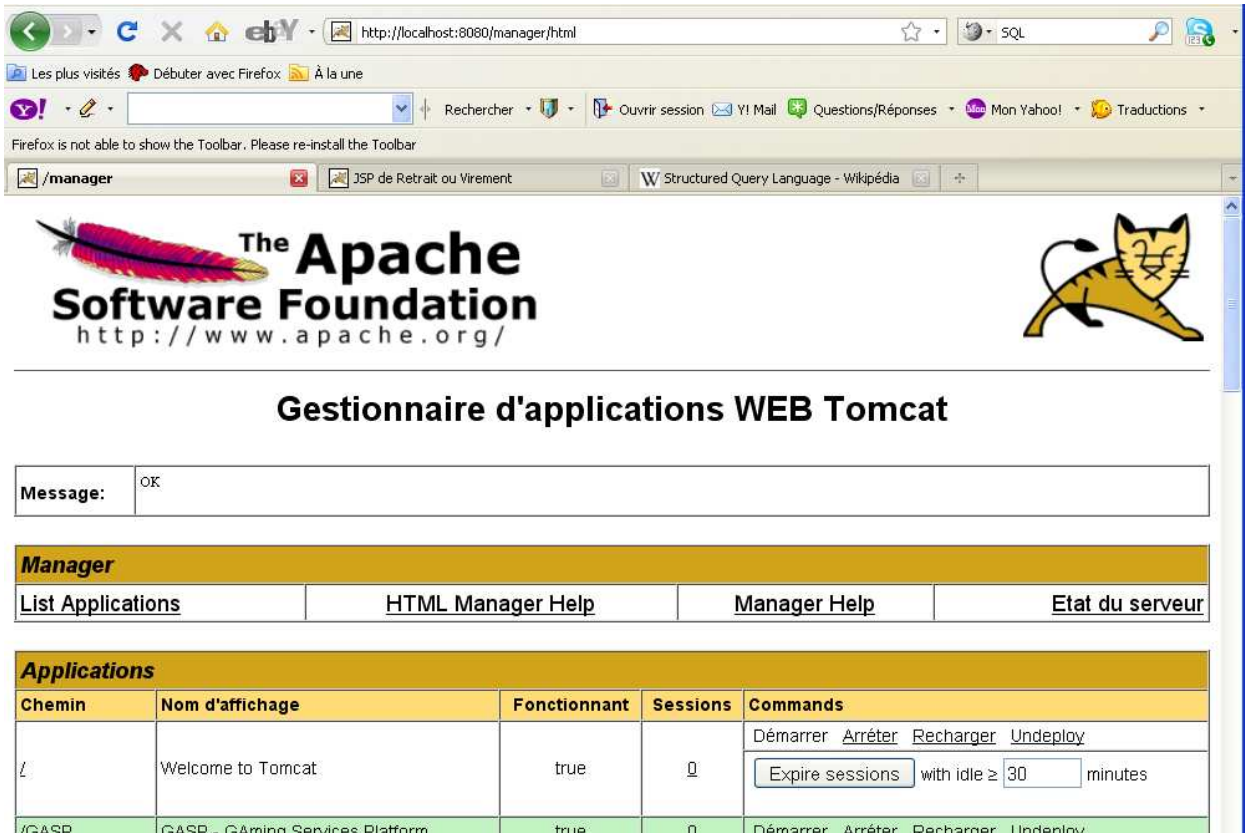
```
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_11
set CATALINA_HOME=C:\www\tomcat
%CATALINA_HOME%\bin\shutdown.bat
```

En ajustant les variables comme indiqué en b).

f) lancer le script `arreteTomcat.bat`. Tomcat doit être arréter, Verifier le en essayer de vous connecter dans votre navigateur à l'URL `http://localhost:8080`.
g) relancer tomcat et, après avoir obtenu, la page d'URL `http://localhost:8080`, cliquer sur le lien Tomcat Manager (en haut à gauche). Une fenêtre doit apparaître demandant Utilisateur et Mot de passe. Taper le couple (admin, admin). Si vous ne parvenez pas à vous connecter éditer le fichier `tomcat-users.xml`. Il devra contenir :

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager"/>
  <user username="admin" password="admin" roles="manager"/>
</tomcat-users>
```

A priori seules les lignes grasses sont à ajouter. Arrêter tomcat et refaites l'étape g). Après authentification du gestionnaire de tomcat dans votre navigateur, vous devriez obtenir un écran comme :



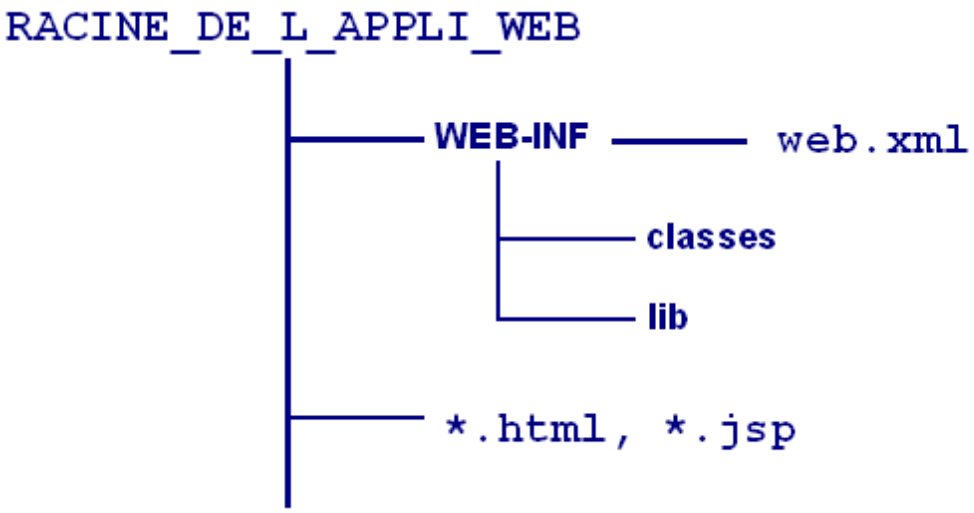
Dans toute cette partie, il n'y a qu'à suivre les instructions, pour arriver à l'écran ci dessus.

Construction d'une application Web : page d'accueil `index.html`

1°) Préparer une arborescence par exemple dans le répertoire `SiteWebTP` qui est une copie de la future application web lorsqu'elle sera déployée. A partir de la racine de cette arborescence placer correctement :

- le répertoire `WEB-INF`
- le répertoire `classes` où seront placées les servlets compilés
- le fichier `web.xml`

On rappelle que cette arborescence doit être :



Construire la page HTML `index.html`, la mettre dans cette arborescence.

Il faut "vraiment" suivre cette arborescence et mettre le fichier `index.html` directement sous la racine de l'application, si on veut bien poursuivre l'exercice.

2°) Ecrire un fichier `web.xml` de la forme :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
<display-name>TP servlets / JSP</display-name>
<description>un TP illustrant les servlets et les JSP</description>

<welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
      <welcome-file>index.html</welcome-file>
</welcome-file-list>

</web-app>
```

3°) Construire une archive `smb111.war` à l'aide de la commande `jar`. en mettant la page `index.html` dans cette archive. On pourra utiliser le script suivant :
fichier `construitWar.bat`

```
cd SiteWebTP
jar cvf smb111.war .
copy smb111.war ..
del smb111.war
cd ..
```

Ce fichier sera mis dans le répertoire père du répertoire `SiteWebTP`.

4°) Déployer cette archive dans le serveur web tomcat (à l'aide de l'outil Tomcat Manager accessible depuis l'URL `http://localhost:8080/`).

5°) Accéder à cette page HTML à l'aide d'un navigateur web après avoir lancé le serveur web tomcat à l'aide d'une URL de la forme `http://localhost:8080/smb111/`

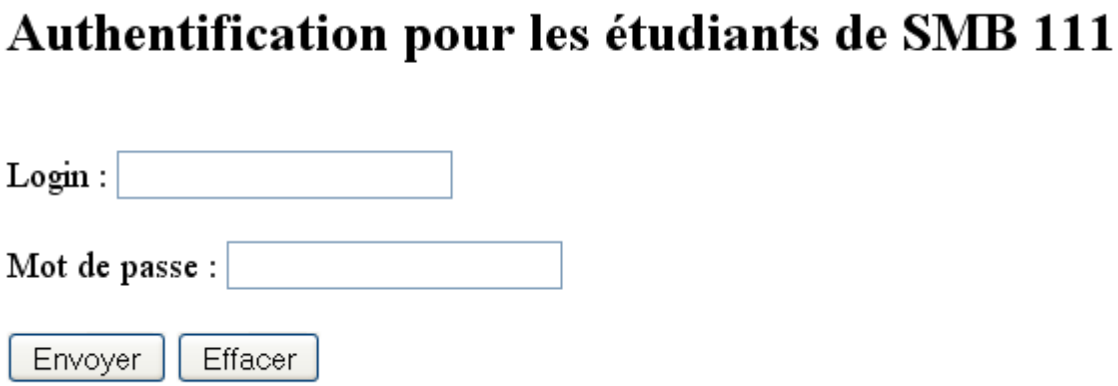
Si tout a été respecté, la page `index.html` est alors visible dans le navigateur.

Constructions de pages JSP

6°) Transformer la page `index.html` ci dessus en page `index.jsp`. Que doit on faire ? Accéder à cette nouvelle page à l'aide de l'URL
`http://localhost:8080/smb111/index.jsp`
puis à l'aide de l'URL
`http://localhost:8080/smb111/`

Il n'y a essentiellement rien à faire dans cette partie sinon remplacer l'extension `.html` par `.jsp`. Comme quoi remplacer un site web statique par un site web dynamique, c'est parfois simple !
Si tout a été respecté, la page `index.html` est alors visible dans le navigateur.

7°) Ecrire une page JSP qui retourne au client web la page suivante :



On construira un formulaire dont le code pourra être :
```
<form action="AuthentificationServlet" method="post">
  <br/>Login : <input type="text" name="login"><br/>
  <br/>Mot de passe : <input type="password" name="motDePasse"><br/>
  <br/><input type="submit" value="Envoyer" />
  <input type="reset" value="Effacer" />
</form>
```

Dans un tel formulaire lorsque l'utilisateur clique sur le bouton Envoyer, une connexion est faite sur la servlet dont l'URL se termine par `AuthentificationServlet`.

Le code de cette page JSP est :
```
<html>
<head><title>Authentification pour SMB111</title></head>
<body>
  <h2>Authentification pour les étudiants de SMB 111</h2>
    <form action="AuthentificationServlet" method="post">
      <br/>Login : <input type="text" name="login"><br/>
      <br/>Mot de passe : <input type="password" name="motDePasse"><br/>
      <br/><input type="submit" value="Envoyer" />
           <input type="reset" value="Effacer" />
    </form>
</body>
</html>
```

C'est plutôt une page HTML : c'est donc une page JSP !

8°) Faire afficher cette page avec une URL de la forme :
`http://localhost:8080/smb111/authentification.jsp`
Il faudra pour cela, en plus d'écrire cette page JSP, construire l'archive `smb111.war` et déployer cette archive.

Là aussi, la principale difficulté est de bien construire l'archive et de bien la déployer.

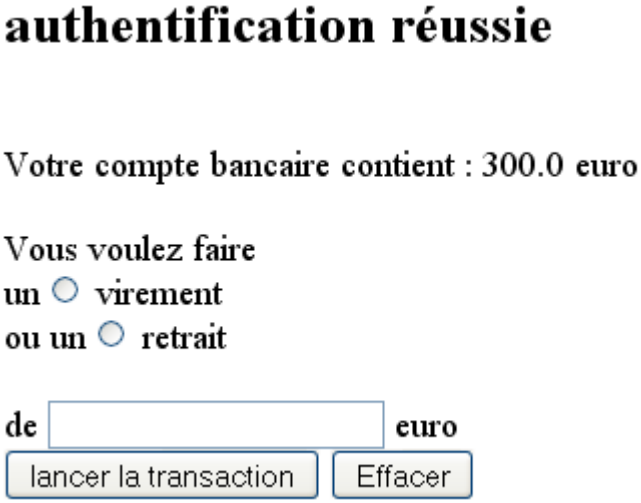
Construction de servlets

9°) Construire la servlet accessible par l'URL
`http://localhost:8080/smb111/AuthentificationServlet`
Il faudra pour cela :
- insérer dans le fichier `web.xml` des lignes comme :

```
<servlet>
     <servlet-name>unNomQuelconque</servlet-name>
     <servlet-class>AuthentificationServlet</servlet-class>
</servlet>

<servlet-mapping>
     <servlet-name>unNomQuelconque</servlet-name>
     <url-pattern>/AuthentificationServlet</url-pattern>
</servlet-mapping>
```

- écrire une servlet (eh oui). Cette servlet doit fabriquer et retourner une page comme



si le couple (nom de login, mot de passe) est correct On pourra prendre le couple (smb111, smb111pw). Si ce couple n'est pas correct on doit avoir une page comme :

Pour compiler et déployer la servlet, on pourra utiliser, quitte à l'adapter, le script :
fichier `compileEtDeploieAuthentificationServlet.bat`

```
set TOMCAT_HOME=C:\www\tomcat
set OLD_CLASSPATH=%CLASSPATH%
set CLASSPATH=%TOMCAT_HOME%\lib\servlet-api.jar;%CLASSPATH%
cd src
javac -d ..\SiteWebTP\WEB-INF\classes AuthentificationServlet.java
cd ..
set CLASSPATH=%OLD_CLASSPATH%
```

Ce fichier sera mis dans le répertoire père du répertoire `SiteWebTP` et le code de la servlet (`AuthentificationServlet.java`) sera dans le répertoire `src`.

Le code de la servlet est :

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AuthentificationServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    // Etape 1.  Spécifier le type MIME du contenu de la réponse
    response.setContentType("text/html");
   // Etape 2.  Récupère le PrintWriter
   // pour envoyer des données au client
   PrintWriter out = response.getWriter();
   // Step 3.  Envoyer l'information au client
   out.println("<html>");
   out.println("<head><title>servlet d'authentification</title></head>");
   out.println("<body>");
   String login = request.getParameter("login");
   String motDePasse = request.getParameter("motDePasse");
   if (login.equals("smb111") && motDePasse.equals("smb111pw")) {
      ServletContext context = getServletContext();
      RequestDispatcher dispatcher =
        context.getRequestDispatcher("/virOuRetrait.jsp");
        dispatcher.forward(request, response);
      } else {
        out.println("<h1>échec à l'authentification</h1>");
      }
        out.println("</body></html>");
   }

   public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
      doGet(request, response);
   }
}
```

Architecture MVC

10°) Modifier la servlet `AuthentificationServlet` de sorte qu'elle appelle des pages JSP qui affichent les réponses ci dessus (authentification réussie et échec à l'authentification).

Il suffit d'insérer dans la servlet, le code vu en cours :

```
  ServletContext context = getServletContext();  // héritée de GenericServlet
```

```
RequestDispatcher dispatcher =
    context.getRequestDispatcher("/maPageMiseEnForme.jsp");
 dispatcher.forward(request, response);
```

en indiquant les bonnes pages JSP vers lesquelles on doit être redirigé.