EISTI – TP de Java EE

Séance 1 : Prise en Main de l'environnement JEE

IMPORTANT ! Pour ce premier TP, nous n'utiliserons pas d'IDE. Nous verrons au prochain TP en quoi les IDE peuvent nous aider dans la réalisation d'un projet JEE.

1 : Création d'un projet JEE

1.1 : Apache Tomcat

Sur vos machines, un **serveur d'applications Tomcat** est installé. Ce serveur possède un **conteneur** JEE permettant la réalisation de projets JEE.

- Tout d'abord, vérifiez que Tomcat est correctement installé sur vos machines.

Correction :

Pour celà, lancez Tomcat (ou Tomcat Monitor qui permet un suivi visuel de l'état du serveur), puis ouvrez un navigateur Web à l'URL **http://localhost:8080/**

Note : Il se peut que Oracle tourne déjà sur le port 8080. Pour pallier au problème, modifier le port utilisé par Tomcat dans le fichier de configuration du serveur Tomcat (<Répertoire Tomcat>/conf/server.xml)

- Ouvrez le **Tomcat Manager** et regardez les différents projets d'exemples déjà installés sur le serveur.

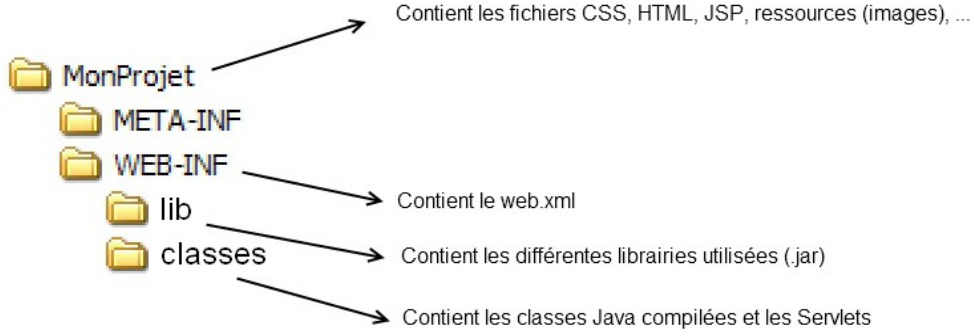
Note : Les logins / passwords des utilisateurs de Tomcat peuvent être trouvés/modifiés dans le fichier <Répertoire Tomcat>/conf/tomcat-users.xml

1.2 : Premier projet JEE

- Créez un nouveau projet JEE : *MonProjet* contenant un **debut.html** faisant un Hello World.

Correction :

Il faut créer un nouveau dossier *MonProjet* contenant le projet JEE. On y mettra les sources Java (servlets et classes) ainsi que les fichiers HTML/JSP/CSS dans leur phase de **développement.** Lorsque le projet est fonctionnel dans le répertoire de développement, on peut passer à la phase de **déploiement** sur le serveur. Pour cela, allez dans le répertoire de Tomcat puis dans le dossier **webapps** (contenant les projets JEE déployés sur le serveur)**.** Dans le conteneur web (Tomcat), la hiérarchie de dossiers suivante doit être respectée pour que Tomcat puisse charger le projet JEE :



Le strict minimum étant le dossier WEB-INF et le fichier web.xml (sauf depuis la v.6 où le web.xml n'est plus obligatoire). Ici on doit rajouter un fichier *debut.html* dans le répertoire racine du projet.

- Lancez votre Projet sur Tomcat et vérifiez qu'il fonctionne correctement.
- Configurez votre Projet pour que la page de démarrage par défaut soit votre *debut.html*

Correction :

Il faut configurer le fichier web.xml en ajoutant les lignes suivantes :

```
<web-app>
     <welcome-file-list>
          <welcome-file>debut.html</welcome-file>
     </welcome-file-list>
</web-app>
```

Ainsi, lorsque vous lancez votre site : http://localhost:8080/MonProjet/ sans spécifier de fichier, Tomcat lance le fichier *debut.html* par défaut.

Note v.6: Il n'existe pas d'annotation remplaçant ceci, le fichier web.xml est obligatoire si l'on veut définir une url par défaut.

1.3 : Ma première Servlet

Pour l'instant, nous avons réussi à créer une page Web statique et à la lancer sur notre serveur Tomcat. C'est bien mais ce n'est pas du Web dynamique. Pour faire celà, nous allons créer notre première **Servlet** et allons essayer de la faire fonctionner sur notre site Web. Une Servlet étant une classe Java, il faudra écrire son code dans un fichier .java puis le compiler.

- Créez une Servlet affichant : Bonjour le monde.

Correction :

Il faut commencer par créer le fichier java : **MaServlet.java**

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class MaServlet extends HttpServlet
{
      protected void doGet(HttpServletRequest request, HttpServletResponse
                                 response) throws ServletException, IOException
      {
            PrintWriter out = response.getWriter();
            out.println("<html><head><title>Ma Premiere
      Servlet</title></head><body><p>Bonjour le monde</p></body></html>");
            out.close();
      }
}
```

On récupère le flot de sortie : **PrintWriter**, et on écrit dedans le fichier HTML qui sera généré par le serveur. Puis on compile le fichier, pour celà il faut mettre dans le classpath l'url du .jar gérant les servlet (pour les import javax.servlet.*). Cette librairie est présente dans le JDK orienté JEE, mais également dans Tomcat dans le répertoire lib (afin qu'il puisse charger les fichier .class de vos projets). Ainsi, on peut compiler en spécifiant l'url du jar :

```
javac -classpath ../../lib/servlet-api.jar MaServlet.java
```

On doit mettre le fichier **.class** dans le répertoire **WEB-INF/classes** afin que Tomcat puisse les trouver.

Ensuite on doit configurer le fichier web.xml qui se charge de faire le mapping entre une URL et notre Servlet.

```
<servlet>
  <description></description>
  <display-name>Une Servlet trop bien</display-name>
  <servlet-name>Servlet1</servlet-name>
  <servlet-class>MaServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Servlet1</servlet-name>
  <url-pattern>/Hello</url-pattern>
</servlet-mapping>
```

Ainsi, on peut ensuite accéder à notre servlet par l'URL **http://localhost:8080/MonProjet/Hello**
Attention : Lorsque le fichier web.xml est modifié, il faut recharger le Projet par Tomcat

Correction v.6

La nouvelle version permet de s'affranchir du fichier web.xml dans des cas très simples comme celui-ci (il reste obligatoire si l'on veut définir davantage de propriétés avancées). Ici, on peut définir le mapping de la Servlet par une annotation (`@WebServlet`) directement dans le code de la servlet. Cette annotation contient l'**url-pattern** correspondant à l'url qui permettra d'appeler la servlet sur le serveur.

Correction :

Il suffit donc de créer le fichier java : **MaServlet.java**

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.WebServlet;

@WebServlet("/Hello")
public class MaServlet extends HttpServlet
{
	protected void doGet(HttpServletRequest request, HttpServletResponse
				response) throws ServletException, IOException
	{
		PrintWriter out = response.getWriter();
		out.println("<html><head><title>Ma Premiere
	Servlet</title></head><body><p>Bonjour le monde</p></body></html>");
		out.close();
	}
}
```

2 : Module WAR

La plupart des applications JEE sont livrés dans un module WAR (équivalent du JAR pour un projet JEE). Ce WAR contient l'ensemble de votre projet et peut être déployé directement sur un serveur d'applications JEE (comme Tomcat).

- Créez le WAR de votre projet.

Correction :

Il suffit de se placer à la racine du dossier du projet et taper la commande :

jar cvf MonProjet.war *

- Déployez-le sur Tomcat

Correction :

Il suffit d'aller dans l'onglet Tomcat Manager sur **http://localhost:8080/** , en bas de la page on peut déployer un WAR sur le serveur Tomcat. Il apparaît ensuite dans la liste des projets.