ING2
Semestre 1

Développement d'applications Web

Éric Pipard

Travail dirigé n° 2
Synopsis de développement d'une webapp

Objectifs : connaître le synopsis de développement d'une application Web

Mots-clés : webapp, servlets, Eclipse

Table des matières

1 – Création manuelle d'un projet webapp....3
2 – Création d'un projet webapp avec Eclipse....3
2.1 - Création d'un serveur Tomcat sous Eclipse....3
2.2 - Création d'un premier projet Web....3
2.3 - Création d'un fichier de bienvenue....6
2.4 - Création d'un fichier html....9
3 - Création d'une première servlet....10

1 – Installation de Tomcat et Eclipse

Avant de commencer ce TD vous devez installer le serveur d'applications Tomcat ou un autre serveur Java EE et un IDE comme Eclipse ou un autre (ex NetBeans)

2 – Création d'un projet webapp avec Eclipse

2.1 - Création d'un serveur Tomcat sous Eclipse

Aller dans la fenêtre des *Preferences* d'Eclipse, à laquelle on peut accéder dans le menu *Windows*.

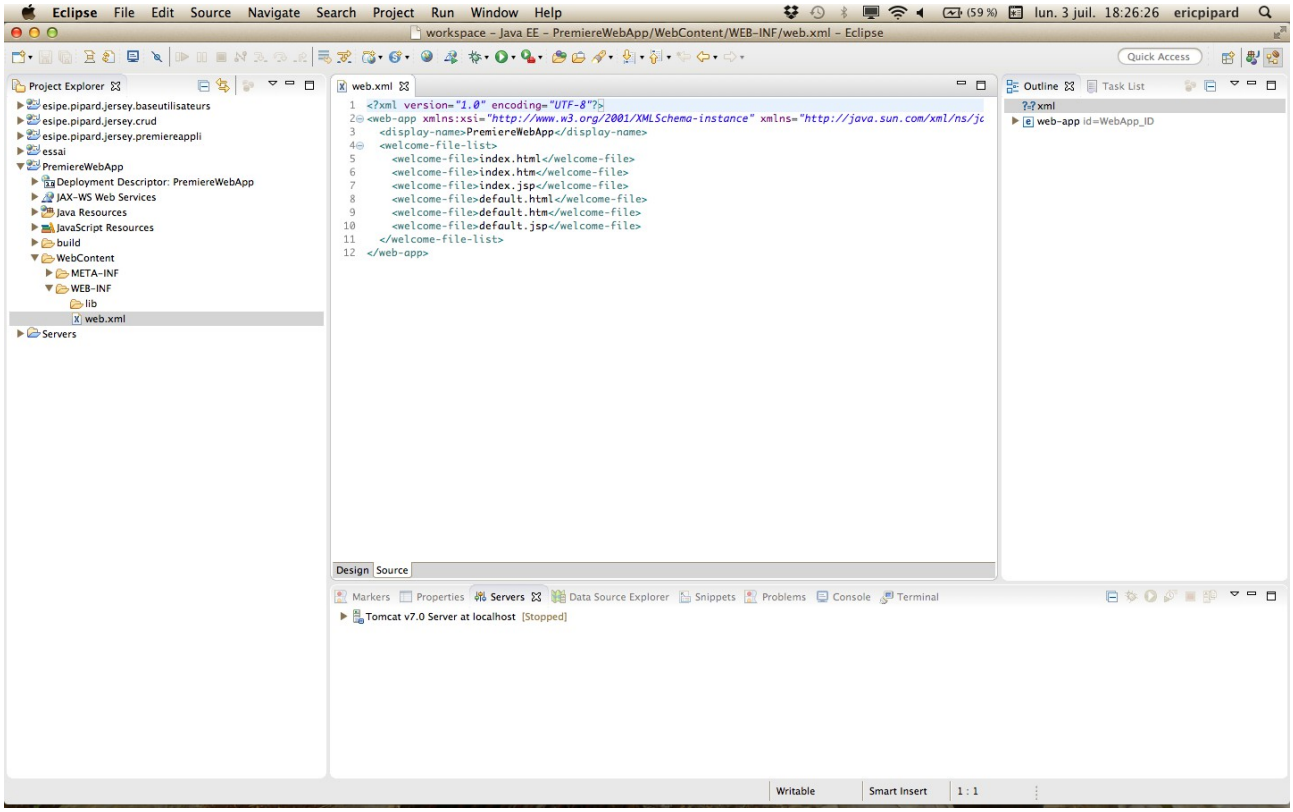
Choisir l'option *Server Runtime Environments*.

Cliquer sur le bouton *Add...* ouvre la liste des *plugins* qu'Eclipse possède déjà.
On sélectionne celui qui correspond à la version installée de Tomcat.

Indiquer où se trouve le répertoire d'installation de Tomcat. Une fois le répertoire validé, l'installation de Tomcat que l'on a spécifiée apparaît dans la liste des *Server Runtime Environments*.

2.2 - Création d'un premier projet Web

La création d'une application Web débute par la création d'un projet Eclipse en sélectionnant *Dynamic Web* Project. Donner *PremierWebApp* comme nom à l'application.
Cliquer sur *Next*. Puis sur *Next* et cochez *Generate web xml deployment descriptor*.
Vous devriez obtenir ceci :

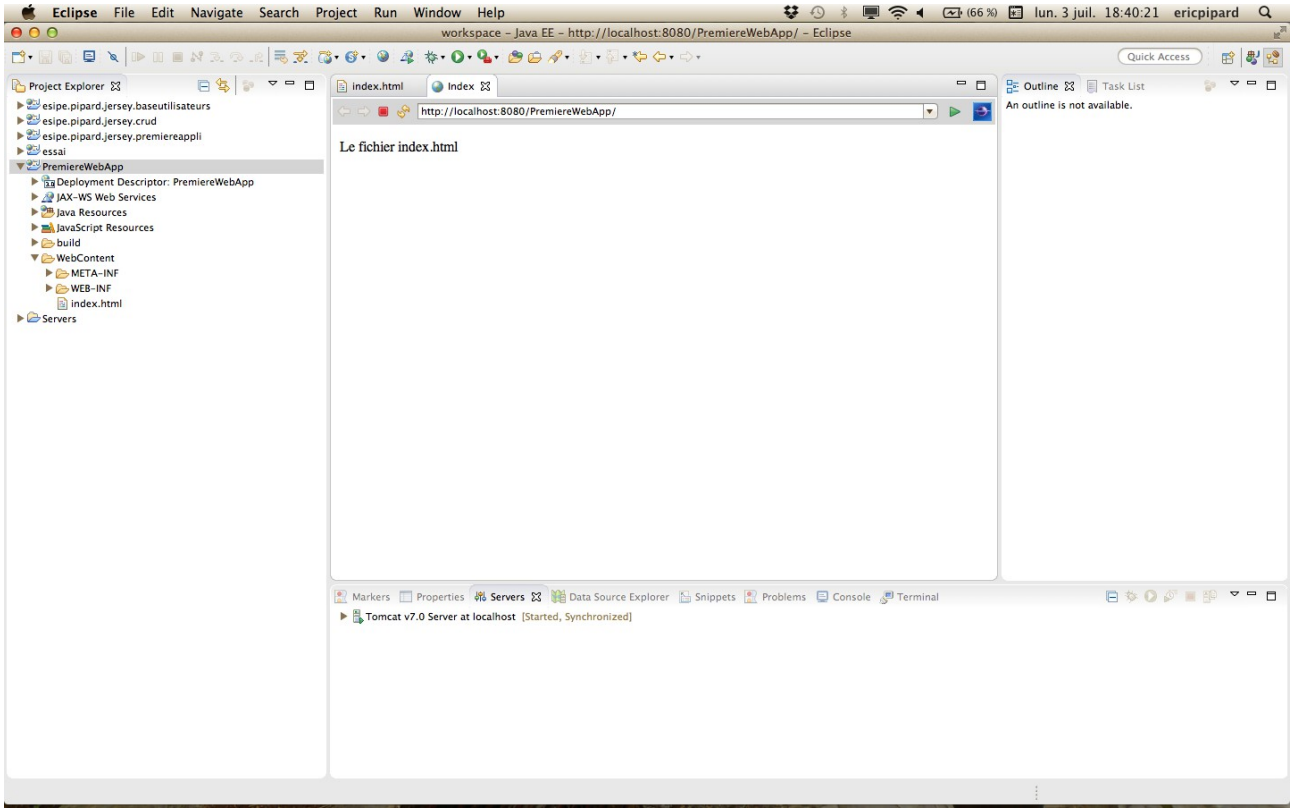


2.3 - Création d'un fichier de bienvenue

La balise <welcome-file-list> est utilisée pour spécifier les fichiers qui doivent être invoqués par le serveur, si vous ne spécifiez pas de nom de fichier pendant le chargement du projet sur le navigateur.
Ainsi http://localhost:8080/PremiereWebApp recherchera la balise <welcome-file-list> dans le fichier web.xml du projet.
Le serveur recherchera un des fichiers précisés dans <welcome-file-list> dans l'ordre de leur écriture. Si la balise <welcome-file-list> n'est pas définie dans web.xml ou que les fichiers de bienvenue définis dans les balises <welcome-file> n'existent pas, le serveur recherchera les fichiers suivants dans cet ordre :

1. index.html
2. index.htm
3. index.jsp

Créer un fichier *index.html*.
Exécuter l'application.
Vous devriez obtenir ceci :



Si vous créez un répertoire *html* et y placez le fichier *index.html* alors il faudra écrire http://localhost:8080/PremiereWebApp/html.

2.4 - Création d'un fichier html

Créer un fichier html nommé *essaiHTML.html* puis exécuter l'application.

3 - Création d'une première servlet

Une servlet étend HttpServlet et redéfinie les méthodes doGet et doPost.

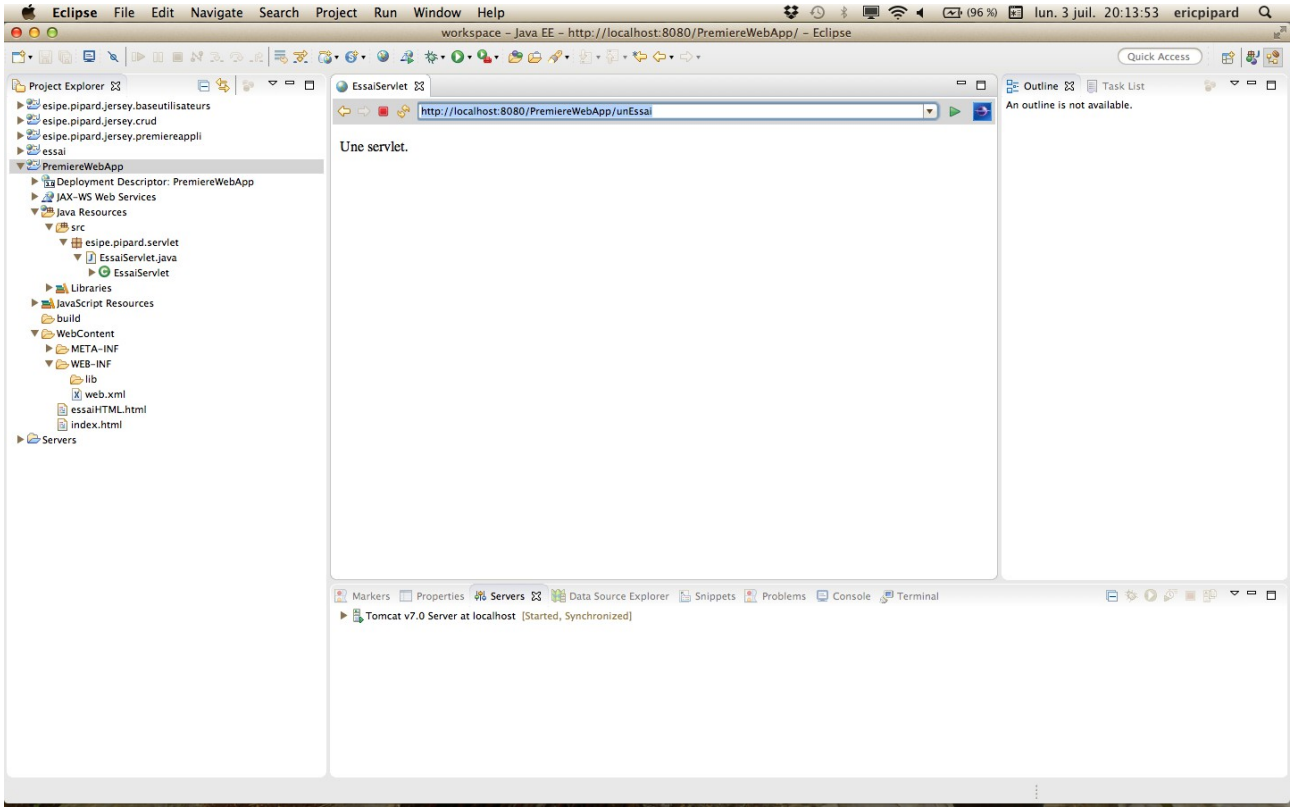
Créer un fichier EssaiServlet.java sous le répertoire src.
Écrire une méthode doGet vide.
Écrire le fichier web.xml.
Exécuter l'application.

Afin d'avoir un affichage, écrire un contenu pour la servlet comme par exemple :

```
public void doGet( HttpServletRequest request, HttpServletResponse response )
throws ServletException, IOException {
        response.setContentType("text/html");
        response.setCharacterEncoding( "UTF-8" );
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<meta charset=\"utf-8\" />");
        out.println("<title>EssaiServlet</title>");
        out.println("</head>");
        out.println("<body>");
```

```
            out.println("<p>Une servlet.</p>");
            out.println("</body>");
            out.println("</html>");
        }
```

Et le résultat :



N.B. : la servlet n'est pas censée s'occuper de l'affichage, c'est la vue qui doit s'en charger. Il ne faudra plus procéder de cette façon.