

Les servlets sous eclipse

Ou « Comment faire le labo 6 »

Introduction

Ce petit (tout petit) document a été fait en vitesse, à la demande de quelques étudiants qui galerait avec les servlets. Je ne garantis cependant rien du tout de ce document, juste que c'est la marche à suivre que j'ai utilisée chez moi et que ça a marché. Pour les réclamations demandez le service après vente ... ;)

Pré-requis

Pour pouvoir utiliser correctement ce document, il faut avoir :

- Eclipse 3.1 avec le plugin Tomcat
- Tomcat 5.x fonctionnel (vérifiez la version de java utilisée par celui-ci en allant dans **configurer tomcat / Java**. Pour le tester aller sur <http://localhost:8080/> une page devrait s'afficher.

Création du projet TomCat

Il faut avant tout créer un projet TomCat.

Sélectionnez le dossier lors de la création. C'est par ce dossier que vous allez accéder au projet. Lorsque l'on parlera de `http://localhost:8080/[Nom_de_votre_projet]/` Ca sera celui la qu'il faudra mettre.

Création du package et de la classe

Cliquez droit sur votre projet et sélectionnez **new / package**. Donnez lui un nom. Il faut remplacer [pack] par le nom que vous venez de lui donner.

Cliquez droit sur votre package et créer une classe. Donnez lui un nom. Il faut remplacer [classe] par ce nom.

Ne pas oublier :

- Dans superclass mettre : `javax.servlet.http.HttpServlet`
- Cocher Inherit Abstract Method
- Decocher public void main ...

Attention, le package et la classe doivent être dans WEB-INF/src.

Création du fichier Web.xml

Dans le dossier WEB-INF créer un fichier web.xml (**clique droit / new / File -> Name : web.xml**)

Modifier les infos ci-dessous et les mettre dans ce fichier :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app SYSTEM "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>Authentification</servlet-name>
    <servlet-class>[pack].[classe]</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Authentification</servlet-name>
    <url-pattern>/[chemin]</url-pattern>
  </servlet-mapping>
</web-app>
```

/[chemin] est le chemin d'accès votre classe.

Sauvegardez tout les fichiers et relancez Tomcat (je sais pas si c'est nécessaire mais j'ai déjà eu des erreurs quand je ne le faisait pas)

Test de fonctionnalité

Allez sur **http://localhost:8080/[Nom_de_votre_projet]/[chemin]**

Vous devriez avoir une page affichant :

```
type Rapport d'état
message La méthode HTTP GET n'est pas supportée par cette URL
description La méthode HTTP spécifiée n'est pas autorisée pour la ressource demandée
(La méthode HTTP GET n'est pas supportée par cette URL).
```

Si pas c'est qu'une erreur s'est glissée dans votre manipulation. Une erreur courante et qui fait peur est la version utilisée. Pour vérifier, fait un clique droit sur votre projet.

Properties / java compiler / use default compliance setting et mettez une valeur correspondante à ce qui a été utilisé par tomcat (voir pré requis).

Création de la classe

Vu que ça marche, on peut créer la classe. Retournons dans le fichier de classe.

Pour commencer on va mettre une méthode init. Celle-ci est appelée lors du premier chargement de la page. Dans mon cas elle contient :

```
private String defaultNom = null;
private String defaultAge = null;
public void init()
{
    ServletConfig config = getServletConfig();
    defaultNom = config.getInitParameter("defaultNom");
    if(defaultNom==null) defaultNom="NNNNNNNNNNNNNNNNNN";
    defaultAge = config.getInitParameter("defaultAge");
    if(defaultAge==null) defaultAge="AAA";
}
```

Ensuite, il faut réécrire les méthodes doGet et doPost.

doGet est appelée lors de l'appel à une page par la méthode get.

doPost est appelée lors de l'appel à une page par la méthode post.

doGet :

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

//    on récupère les paramètres du formulaire
String nom = request.getParameter("nom");
if (nom == null) {
    nom = "";
}
String pass = request.getParameter("pass");
if (pass == null) {
    pass = "";
}

//    on affiche le formulaire

    PrintWriter out = response.getWriter();

    out.println("<html><head><title>Page
d'authentification</title></head><body>");
```

```

if (nom != "" && pass != "")
{
    out.println("<b>Verification du login</b><br>" + nom + pass + "<br>");
    if (nom == "Funcky" && pass == "Max")
    {
        out.println("login ok");
    }
}

out.println("<form method='post' action='[chemin]'" +
    "Login : <input type = 'text' name = 'nom'" +
    "Pass : <input type = 'password' name = 'pass'" +
    "<input type='submit' value='Log'" +
    "</form>");

out.println("</body></html>");
}

```

N'oubliez pas de remplacer [chemin].

La méthode doPost doit appeler doGet :

```

public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException,
ServletException {
//    on passe la main au GET
doGet(request, response);
}

```

Relancez Tomcat et placez vous sur l'adresse :

[http://localhost:8080/\[Nom_de_votre_projet\]/\[chemin\]](http://localhost:8080/[Nom_de_votre_projet]/[chemin])

Normalement votre script devrait être fonctionnel.

Conclusion

Un servlet simple n'est pas compliqué à mettre en place, il faut juste suivre des étapes bien précises.

On peut aussi utiliser des formulaire en JSP pour traiter les données mais pour cel je vous renvoie au PDF *progwebjavaavecclipseettomcat.pdf* à la page 23.

PS : le pdf est dispo sur <http://www.funcky.be/java/> dans la rubrique tutos.

Bonne M\$^*ù pour votre exam ! En espérant avoir pu vous aider ...