Wildfly + EJB

Dans cet exemple nous réaliserons une application Java effectuant une mise en majuscules d'un mot saisi.

Wildfly est requis configuré (add-user) avec comme fichier de paramétrage "standalone-full.xml".

Etape 1 - EJB

Créez un nouveau projet EJB (attention : ne pas cliquer sur Finish mais suivre les étapes pour configurer Wildfly).

Dans ejbModule :

- créer package src
- créer package interfaces

Dans interfaces : on crée une interface **test_interface** (IE).

```
package interfaces;

import javax.ejb.Remote;

@Remote
public interface test_interface {
    String processText(String text);
}
```

Dans src : on crée une classe **test** (IE) qui implémentera l'interface test_interface. Elle contiendra les fonctions permettant de récupérer les données de la BDD.

```
package src;

import javax.ejb.Stateless;

import interfaces.test_interface;

@Stateless
public class test implements test_interface {
    public String processText(String text) {
        return text.toUpperCase();
    }
}
```

Etape 2 - Page web

Créez un nouveau projet HTML "Dynamic Web Project" dans lequel on affichera la page d'accueil de l'application Java EE.

Dans le dossier "WebContent" de votre projet, créez un fichier HTML et nommez-le "index" pour plus de facilité par la suite.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>PA - le dieu</title>
</head>
<body>
    <form action="test_servlet" method="post">

        <input type="text" name="test_var" placeholder="bite">
        <button type="submit">Mise en majuscules</button>

    </form>

</body>
</html>
```

Ajouter le jar "jboss-client".

Etape 3 - Servlet

Créer un nouveau projet Servlet qui effectuera le lien entre l'opération réalisée dans le HTML et fait l'opération définie dans le Backend c'est à dire dans l'EJB.

Le code peut faire peur mais il est généré automatiquement, il faut juste le modifier quelque peu pour adapter.

```
package src;

import java.io.IOException;
import java.io.Writer;

import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import interfaces.test_interface;
```

```java
/**
 * Servlet implementation class test_servlet
 */
@WebServlet("/test_servlet")
public class test_servlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @EJB
    private test_interface test_inter;


    /**
     * @see HttpServlet#HttpServlet()
     */
    public test_servlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // TODO Auto-generated method stub

    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // TODO Auto-generated method stub
        String value = String.valueOf(request.getParameter("test_var"));
        Writer value_writer = response.getWriter();
        value_writer.write("<html>\r\n" +
                "<head>\r\n" +
                "<meta charset=\"ISO-8859-1\">\r\n" +
                "<title>PA - le dieu</title>\r\n" +
                "</head>\r\n" +
                "<body>");
        value_writer.write("Resultat = " + test_inter.processText(value));
        value_writer.write("</body></html>");
        value_writer.flush();
        value_writer.close();

    }

}
```

Etape 4 - EAR

Créer un nouveau projet "Enterprise Application Project" (nommé ici EAR).

Une fois le projet créé, il faut aller (en bas d'Eclipse par défaut) dans l'onglet *Servers*, effectuer un clic droit sur *Wildfly 14*, choisir "Add and remove" et ajouter votre **EAR** dans la colonne *Configured*.

Une fois cette étape réalisée, effectuez un clic droit sur **EAR** dans le défilement de *Wildfly 14* et "Full Publish".

Plus qu'à "Start" le serveur Wildfly et accéder à la page : http://127.0.0.1:9990.

Effectuez une petite visite puis accédez à la partie "Deployments".

Votre EAR est présent, cliquez dessus et suivez le lien correspondant à la case à droite de "Context Roots:".

Vous êtes maintenant sur votre page HTML qui vous permettra d'effectuer votre MAJUSCULISATION !

Etape 5 - BDD

Modifiez le fichier standalone-full dans Wildfly -> standalone -> configuration

```
<datasources>
                <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-
name="ExampleDS" enabled="true" use-java-context="true">
                    <!-- <connection-
url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE</connection-url> -->
                    <connection-url>jdbc:h2:tcp://localhost/~/test</connection-url>
                    <driver>h2</driver>
<!--                     <security>
                        <user-name>sa</user-name>
                        <password>sa</password>
                    </security> -->
                </datasource>
                <drivers>
                    <driver name="h2" module="com.h2database.h2">
                        <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-
datasource-class>
                    </driver>
                </drivers>
            </datasources>
```

Lancez le serveur via l'exécutable Java "start h2-1.4.193". Choisissez "Generic H2 Server" et lancez le serveur.

Créez un nouveau "JPA Project"

Modifier persistence.xml.

Signé : Vahflouz.