

Les Enterprise JavaBeans

Création d'un EJB 2.1 avec Eclipse et JBoss

Claude Duvallet

Université du Havre
UFR Sciences et Techniques
25 rue Philippe Lebon - BP 540
76058 LE HAVRE CEDEX
Claude.Duvallet@gmail.com
<http://litis.univ-lehavre.fr/~duvallet/>

Création d'un EJB 2.1 avec Eclipse et JBoss

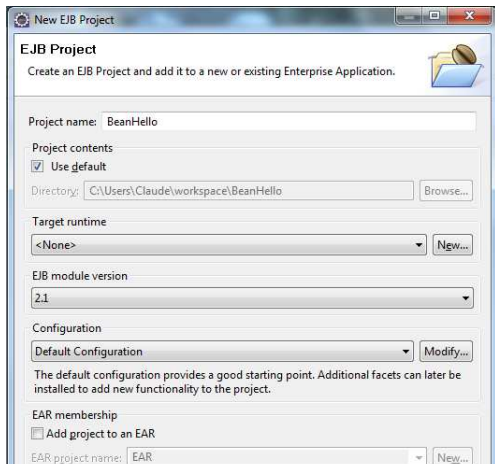
- 1 Création et déploiement d'un Bean
- 2 Création d'un client

Préambule

- ▶ Il faut travailler avec un nouveau "workspace" au niveau d'Eclipse.
- ▶ Il faut supprimer les traces de vos anciens projets EJB qui seraient présents dans le répertoire "deploy" de JBoss.
- ▶ JBoss doit être installé dans un répertoire où l'utilisateur d'Eclipse pourra avoir les droits en écriture.
- ▶ JBoss ne doit pas avoir été démarré.
- ▶ Vous pouvez maintenant démarrer Eclipse : il vous faut bien sûr une version "JEE".

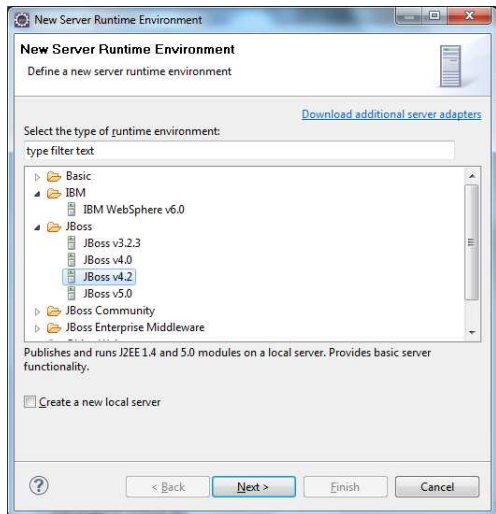
Création d'un nouveau projet EJB

- ▶ Il faut d'abord créer un nouveau projet EJB.
- ▶ Il faut lui donner un nom puis sélection un environnement d'exécution.
- ▶ Il faut sélectionner la version du type de Bean : 2.1.



Sélection de l'environnement d'exécution (1/2)

- ▶ Il faut renseigner le serveur d'application que l'on va utiliser et sa version : JBoss 4.2.



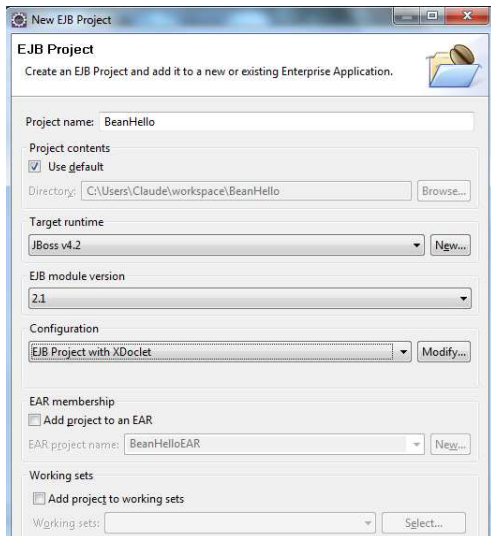
Sélection de l'environnement d'exécution (2/2)

- ▶ Il faut sélectionner le JRE et sa version.
- ▶ Il faut renseigner le répertoire où est installé le serveur JBoss.



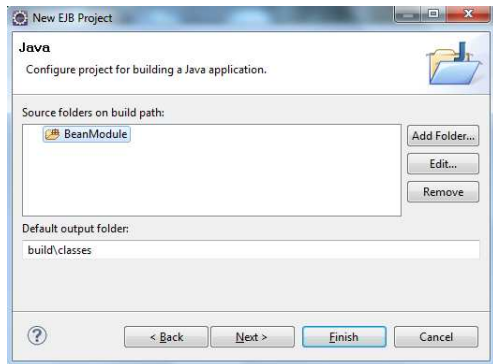
Configuration du projet

- ▶ Choisir "EJB Project avec XDoclet".
- ▶ Passer à l'étape suivante.



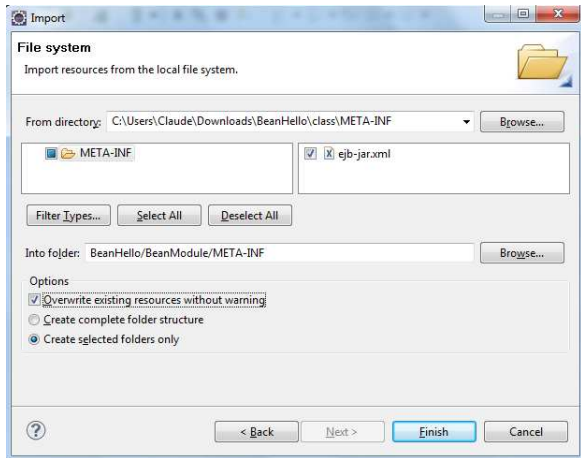
Définition du nom du module EJB

- ▶ Il faut cliquer sur "Edit" pour modifier le nom du module à votre convenance.



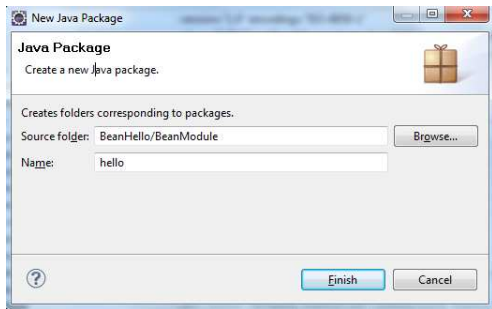
Importation du descripteur de déploiement

- ▶ Bien qu'un descripteur de déploiement (fichier `ejb-jar.xml`) ait été créée, il faut le remplacer.
- ▶ Pour notre exemple, nous importerons celui du bean Hello.



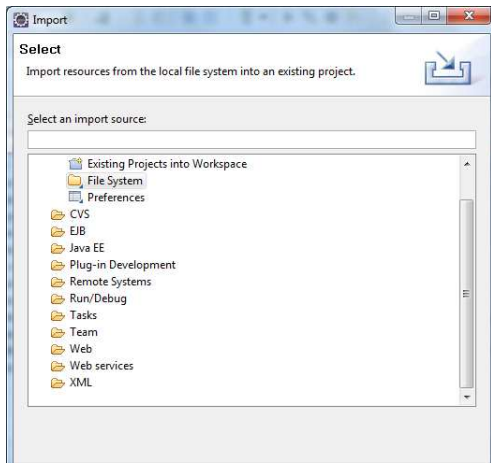
Création d'un package `hello`

- ▶ En vue de l'importation des fichiers sources, il faut créer un nouveau package `hello`.



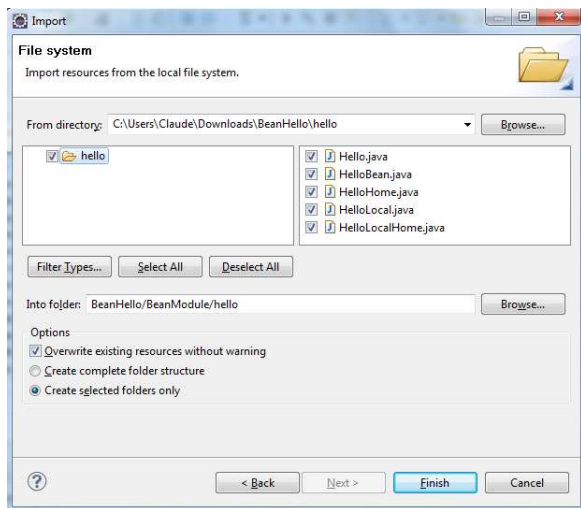
Importation du code source de l'exemple (1/2)

- ▶ Il faut importer au sein du package `hello` tous les fichiers Java de l'exemple.
- ▶ Il faut le faire par le biais de l'option "Système de fichier" ("File System").



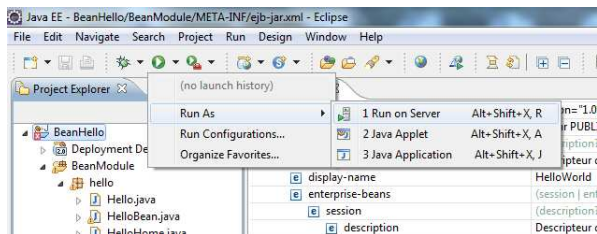
Importation du code source de l'exemple (2/2)

- ▶ Il faut sélectionner le répertoire où se trouve les fichiers sources puis les sélectionner.



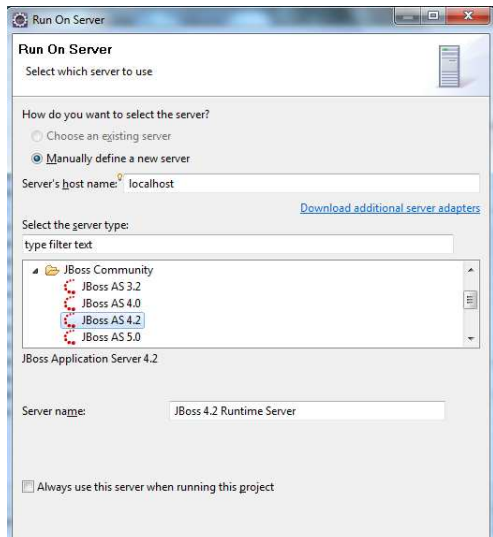
Déploiement du Bean

- ▶ Vous pouvez maintenant déployer le bean.
- ▶ Dans le menu "Run As", sélectionner "Run on Server".



Configuration du serveur JBoss (1/2)

- ▶ Sélectionner la version de JBoss à utiliser : 4.2.
- ▶ Cliquer sur "suivant".



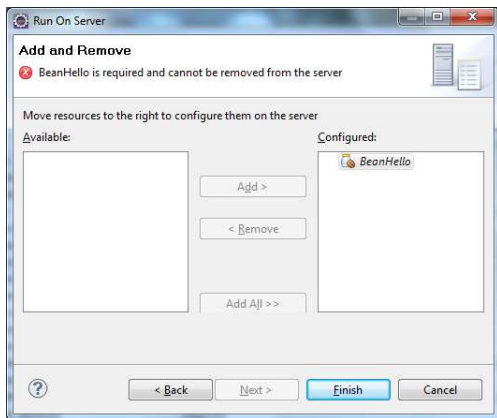
Configuration du serveur JBoss (2/2)

- ▶ Indiquer le répertoire où est installer JBoss.
- ▶ Sélectionner le JRE à utiliser.



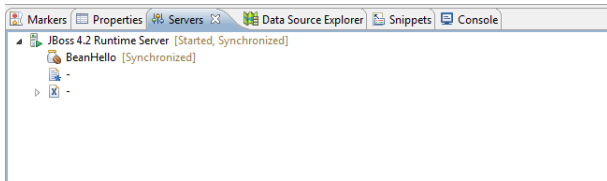
Ajout du Bean à déployer

- ▶ Si cela est nécessaire il faut ajouter le Bean à déployer.
- ▶ Normalement, il n'y a rien à faire.

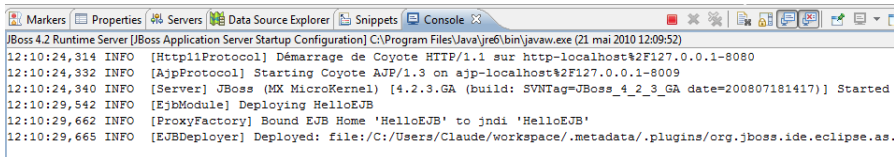


Vérification

- ▶ Vous pouvez vérifier si le bean a été déployé.

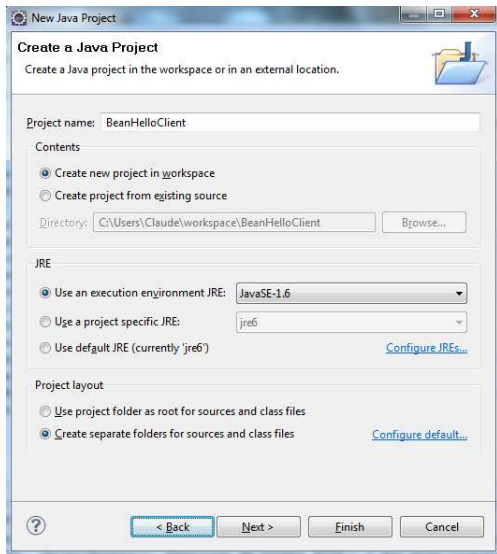


- ▶ La console vous indique si le serveur JBoss est correctement démarré.



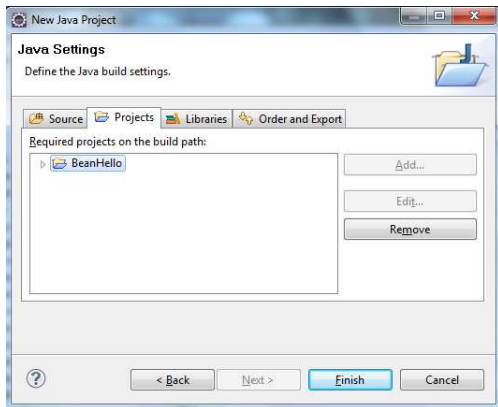
Création du client

- ▶ Il vous faut créer un projet Java pour le client.



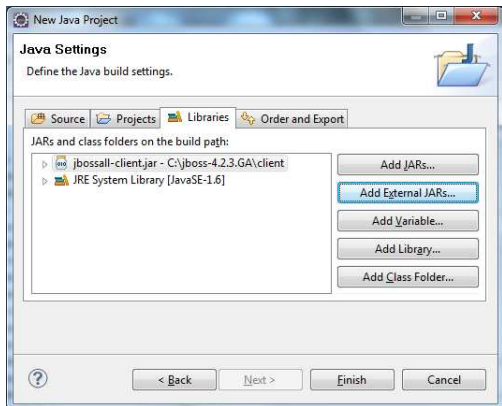
Ajout du Bean

- ▶ Il vous faut ajouter le "BeanHello" dans la zone projet.



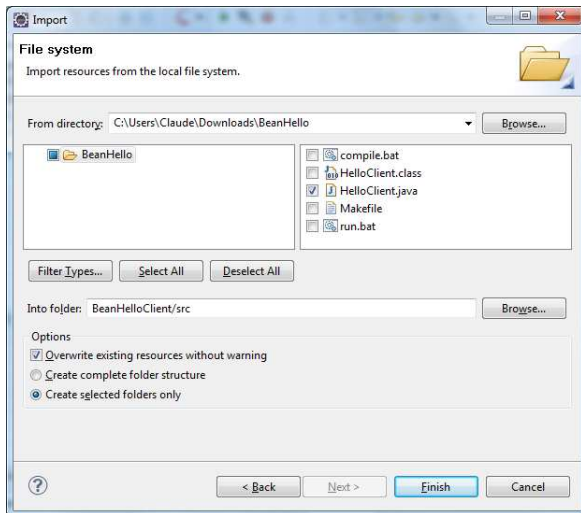
Ajout de la librairie client

- ▶ Il vous faut ajouter le jar "jbossall-client.jar" dans la zone librairie.



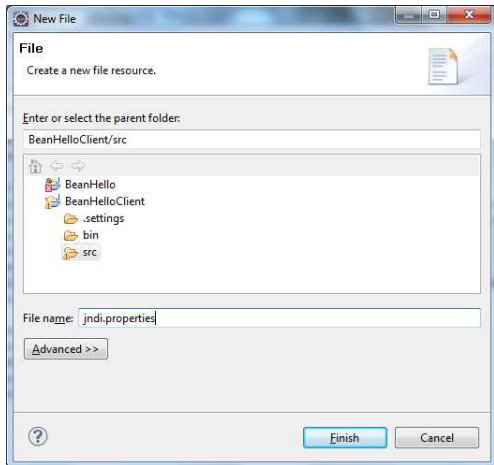
Importation du fichier client

- ▶ Reprenez le client de l'exemple pour l'importer dans le projet.



Création d'un fichier `jndi.properties` (1/2)

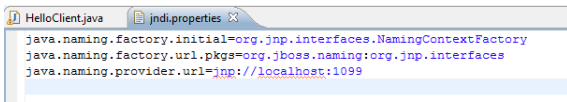
- ▶ Il vous faut créer un nouveau fichier à la racine de votre projet : `jndi.properties`.



Création d'un fichier `jndi.properties` (2/2)

- ▶ Dans le fichier `jndi.properties`, il faut ajouter les trois lignes suivantes :

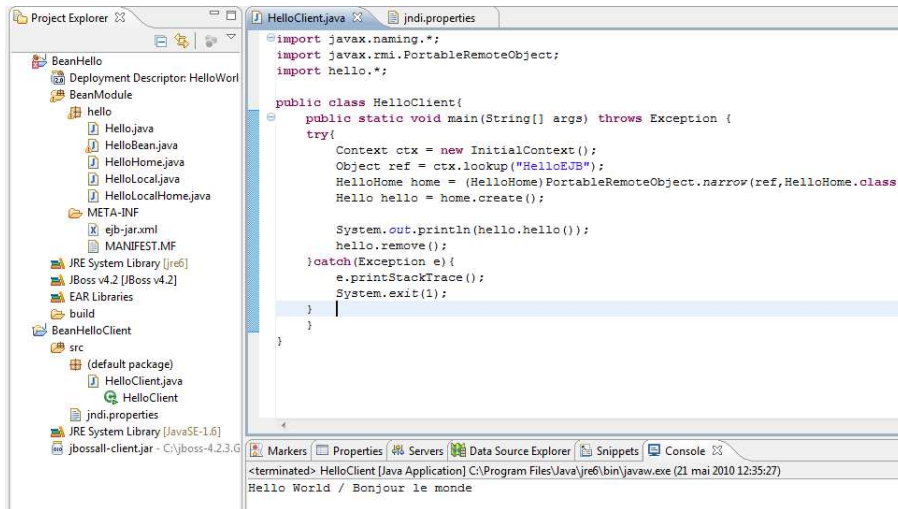
```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
java.naming.provider.url=jnp://localhost:1099
```

A screenshot of an IDE window showing the file `jndi.properties`. The window title bar includes `HelloClient.java` and `jndi.properties`. The code content is as follows:

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
java.naming.provider.url=jnp://localhost:1099
```

Modification du fichier source

- ▶ Vous pouvez enlever les lignes qui renseignent les paramètres d'initialisation du JNDI.



The screenshot shows an IDE with two main windows. On the left is the Project Explorer, and on the right is the source code editor for HelloClient.java.

Project Explorer: Shows a project named 'BeanHelloClient' with a source folder 'src'. Inside 'src', there is a package '(default package)' containing 'HelloClient.java' and 'HelloClient'. There is also a 'jndi.properties' file. Other project elements include 'BeanHello', 'Deployment Descriptor: HelloWorl', 'BeanModule', 'hello' package with 'Hello.java', 'HelloBean.java', 'HelloHome.java', 'HelloLocal.java', and 'HelloLocalHome.java', 'META-INF' folder with 'ejb-jar.xml' and 'MANIFEST.MF', 'JRE System Library [jre6]', 'JBoss v4.2 [JBoss v4.2]', 'EAR Libraries', 'build' folder, and 'BeanHelloClient' folder with 'src' sub-folder.

Source Code Editor (HelloClient.java): Shows the following code:

```
import javax.naming.*;
import javax.rmi.PortableRemoteObject;
import hello.*;

public class HelloClient{
    public static void main(String[] args) throws Exception {
        try{
            Context ctx = new InitialContext();
            Object ref = ctx.lookup("HelloEJB");
            HelloHome home = (HelloHome)PortableRemoteObject.narrow(ref,HelloHome.class);
            Hello hello = home.create();

            System.out.println(hello.hello());
            hello.remove();
        }catch(Exception e){
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```