

UFR Sciences et Techniques du Havre



Compte rendu sur les entreprises Java Beans (EJB)

Relations dans les Beans entités (norme EJB 3.0)

Réalisé par: Imane ZEROUALI

Anass BOUATMANE



ENTERPRISE
Java Beans



2016/2017

Master 2 MATIS

Parcours SIRES

Sommaire :

1) La norme EJB 3.0.....	1
2) Types de relations dans les Beans Entités pour la norme EJB 3.0.....	3
3) Configuration pour le déploiement des projets avec le serveur JBoss 5.1.....	6
4) Exécution des projets du TP.....	14

1) La Norme EJB 3.0

La norme EJB 3.0 est une évolution importante dans le domaine des EJB, elle apporte notamment un nouveau modèle de gestion de la persistance des données inspiré d'Hibernate de Jboss ainsi qu'une simplification très importante des développements en intégrant l'approche de Programmation Orientée Aspect.

Spécification des interfaces locales ou distantes du Bean avec :

- "@Remote"
- "@Local".

Définiion de Bean session avec ou sans état avec

- "@Statefull"
- "@Stateless"

Exemple simple:

- L'interface

```
package hello;
public interface Hello {
    public String hello (String msg);
}
```

- Le Bean :

```
package hello;
import javax.ejb.*;
@Stateless
@Remote (Hello.class)
public class HelloBean implements Hello {
    public String hello (String msg){
        System.out.println ("Message reçu : "+msg);
        return "Hello "+msg;
    }
}
```

- Le descripteur de déploiement :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<ejb-jar xmlns="http://java.sun.com/xmlns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/ejb-jar\_3\_0.xsd
    version="3.0">
    <enterprise-beans>
    </enterprise-beans>
</ejb-jar>
```

- La définition du client :

```
import javax.naming.Context;
import javax.naming.InitialContext;
import hello.*;
import java.util.*;

public class HelloClient{
    public static void main(String[] args) throws Exception {
        Properties props = System.getProperties();
        props.put(Context.INITIAL_CONTEXT_FACTORY,
            "org.jnp.interfaces.NamingContextFactory");
        props.put(Context.URL_PKG_PREFIXES,
            "org.jboss.naming:org.jnp.interfaces");
        props.put(Context.PROVIDER_URL,
            "jnp://localhost:1099");

        Context ctx = new InitialContext(props);
        Hello hello = (Hello) ctx.lookup("HelloBean/remote");

        System.out.println(hello.hello("World"));
    }
}
```

Pour gérer le cycle de vie on utilise les méthodes **callback**, elles sont définies à l'aide d'annotation directement dans le Bean, ou dans une autre classe externe.

Annotations :

@CallbackListener(String classname)

@PostConstruct

@PreDestroy

@PostActivate

@PrePassivate

@EntityListener(String classname)

@PrePersist

@PostPersist

@PreRemove

@PostRemove

@PreUpdate

@PostLoad

2) Types de relations dans les Beans Entités pour la norme EJB 3.0

Les EJB utilisent un modèle de persistance léger:

Annotation : (*bean comme étant de type entité*)

@Entity (*import javax.persistence.Entity;*)

Gestion de la persistance :

Accès directe aux champs à rendre persistant :

@Entity(access=AccessType.FIELD)

Obligation d'utilisation d'accesseurs :

@Entity(access=AccessType.PROPERTY)

Il existe 4 types de relations entre entités:

- relation Un à Un : son annotation est **@OneToOne**.
- relation Un à Plusieurs : son annotation est **@OneToMany**.
- relation Plusieurs à Un : son annotation est **@ManyToOne**.
- relation Plusieurs à Plusieurs: son annotation est **@ManyToMany**.

Exemples :

1 à 1 : Cas unidirectionnelle

```
/**
 *Entité Directeur
 */

@Entity
public class Directeur implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(length=30)
    private String nom ;
    ....
}

/**
 * Entité Entreprise
 */

@Entity
public class Entreprise implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(length=30)
    private String nom ;

    @OneToOne
    private Directeur directeur ;
    ....
}
```

1 à 1: Cas bidirectionnelle

```
/**
 * Entité Directeur
 */

@Entity
public class Directeur implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(length=30)
    private String nom ;

    @OneToOne(mappedBy="directeur") // référence la relation dans la classe Entreprise
    private Entreprise entreprise ;

    ....
}
```

1 à plusieurs bidirectionnelle

```
/**
 * Entité Vendeur
 */

@Entity
public class Vendeur implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne
    private Magasin magasin ;

    ....
}

/**
 *Entité Magasin
 */
@Entity
public class Magasin implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @OneToMany(mappedBy="magasin")
    private Collection<Vendeur> vendeurs ;

    ....
}
```

Plusieurs à 1 unidirectionnelle

```
/**
 * Entité Vendeur
 */

@Entity
public class Vendeur implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne
    private Magasin magasin ;
    ....
}

/**
 * Entité Magasin
 */
@Entity
public class Magasin implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    ....
}
```

Plusieurs à plusieurs cas unidirectionnels:

```
/**
 * Entité Cycliste
 */

@Entity
public class Cycliste implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToMany
    private Collection<CleMolette> cleMonettes ;
    ....
}

/**
 * Entité CleMolette
 */
@Entity
public class CleMolette implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    ....
}
```

Plusieurs à plusieurs cas bidirectionnels:

```
/**
 * Entité Cycliste
 */

@Entity
public class Cycliste implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToMany
    private Collection<CleMolette> cleMolettes ;
    ...
}

/**
 * Entité CleMolette
 */
@Entity
public class CleMolette implements Serializable {

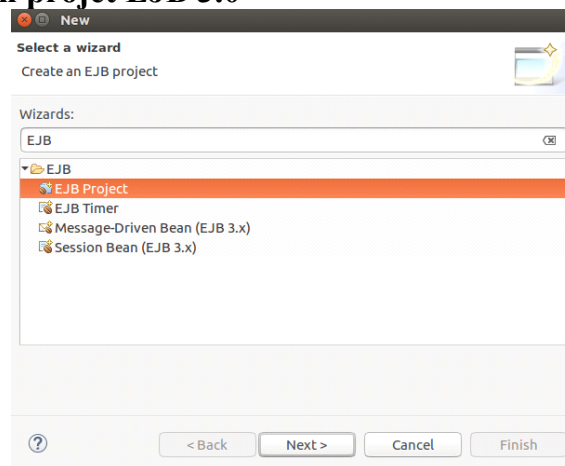
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToMany(mappedBy="cleMolettes")
    private Collection<Cycliste> cyclistes ;
    ...
}
```

3) Configuration pour le déploiement des projets avec le serveur Jboss 5.1

Etapes :

- **Création d'un projet EJB 3.0**



New EJB Project

EJB Project
Create an EJB Project and add it to a new or existing Enterprise Application.

Project name:

Project location:
☒ Use default location
Location:

Target runtime:

EJB module version:

Configuration:

Hint: Get started quickly by selecting one of the pre-defined project configurations.

EAR membership:
☐ Add project to an EAR
EAR project name:

Working sets:
☐ Add project to working sets
Working sets:

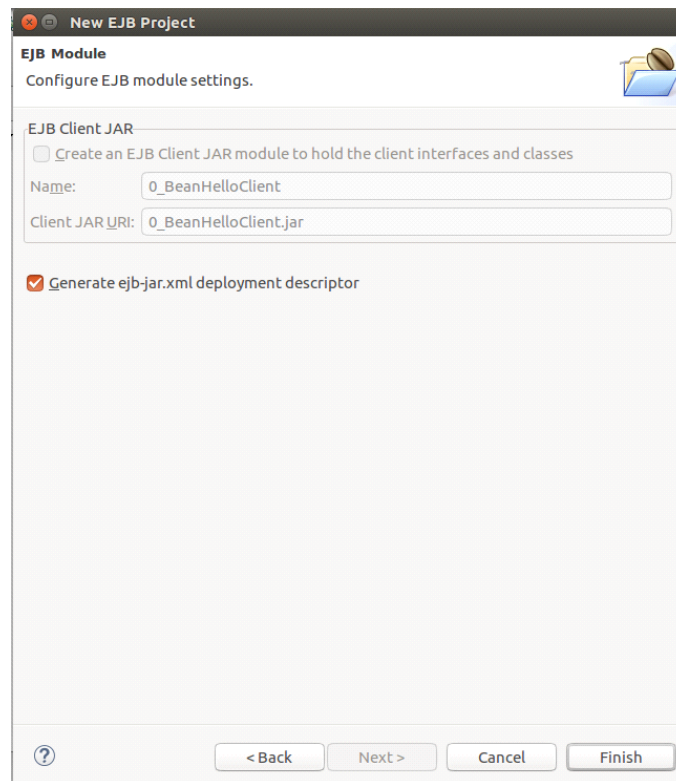
New EJB Project

Java
Configure project for building a Java application.

Source folders on build path:

ejbModule

Default output folder:



New EJB Project

EJB Module
Configure EJB module settings.

EJB Client JAR

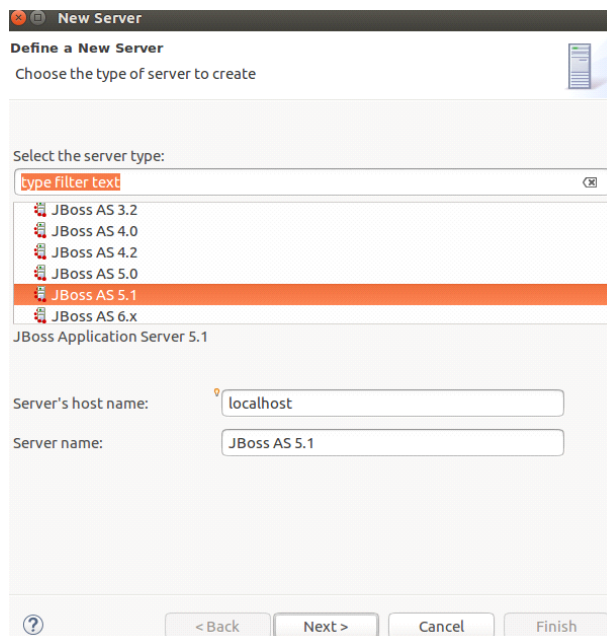
☐ Create an EJB Client JAR module to hold the client interfaces and classes

Name:

Client JAR URI:

☒ Generate ejb-jar.xml deployment descriptor

- **Création d'un serveur Jboss.**



New Server

Define a New Server
Choose the type of server to create

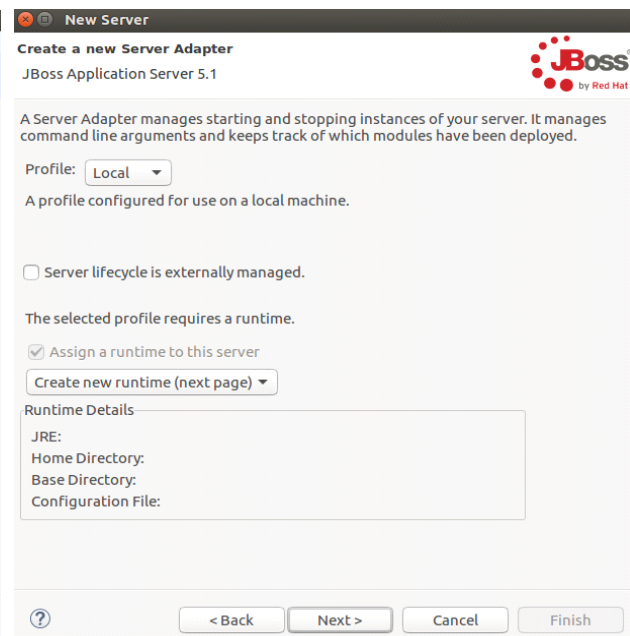
Select the server type:

- JBoss AS 3.2
- JBoss AS 4.0
- JBoss AS 4.2
- JBoss AS 5.0
- JBoss AS 5.1**
- JBoss AS 6.x

JBoss Application Server 5.1

Server's host name:

Server name:



New Server

Create a new Server Adapter
JBoss Application Server 5.1

A Server Adapter manages starting and stopping instances of your server. It manages command line arguments and keeps track of which modules have been deployed.

Profile:

A profile configured for use on a local machine.

☐ Server lifecycle is externally managed.

The selected profile requires a runtime.

☒ Assign a runtime to this server

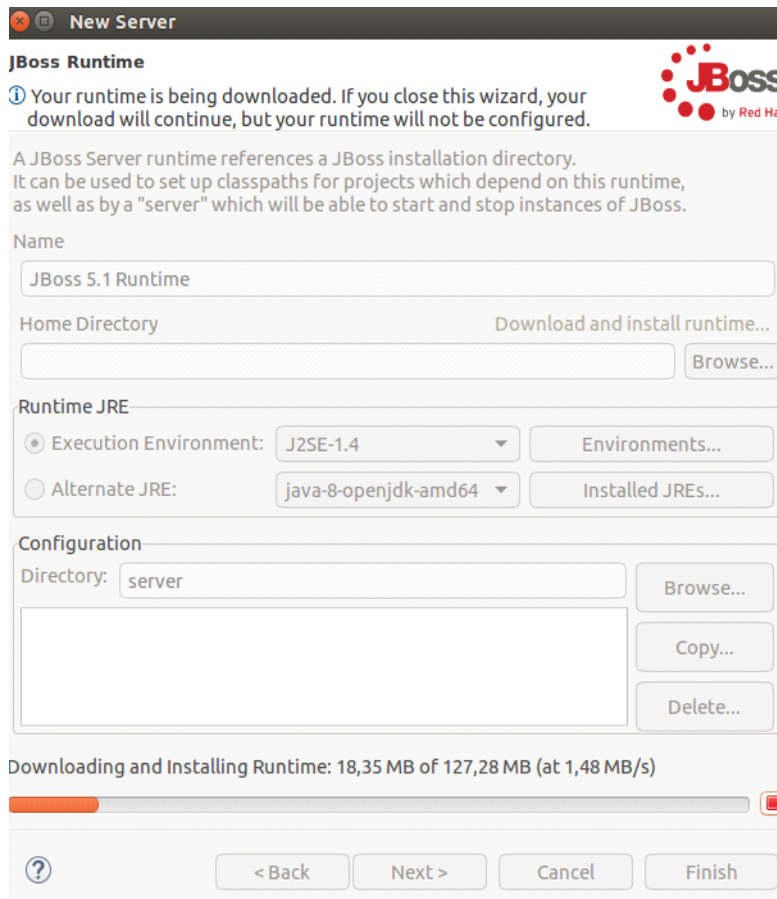
Runtime Details

JRE:

Home Directory:

Base Directory:

Configuration File:



New Server

JBoss Runtime

① Your runtime is being downloaded. If you close this wizard, your download will continue, but your runtime will not be configured.

A JBoss Server runtime references a JBoss installation directory. It can be used to set up classpaths for projects which depend on this runtime, as well as by a "server" which will be able to start and stop instances of JBoss.

Name: JBoss 5.1 Runtime

Home Directory: Download and install runtime...

Runtime JRE

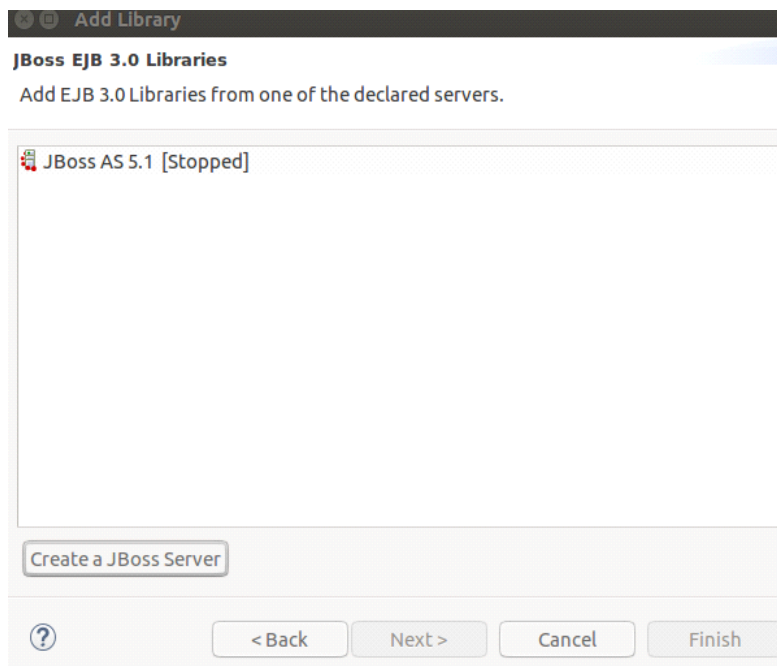
☒ Execution Environment: J2SE-1.4

☐ Alternate JRE: java-8-openjdk-amd64

Configuration

Directory: server

Downloading and Installing Runtime: 18,35 MB of 127,28 MB (at 1,48 MB/s)



Add Library

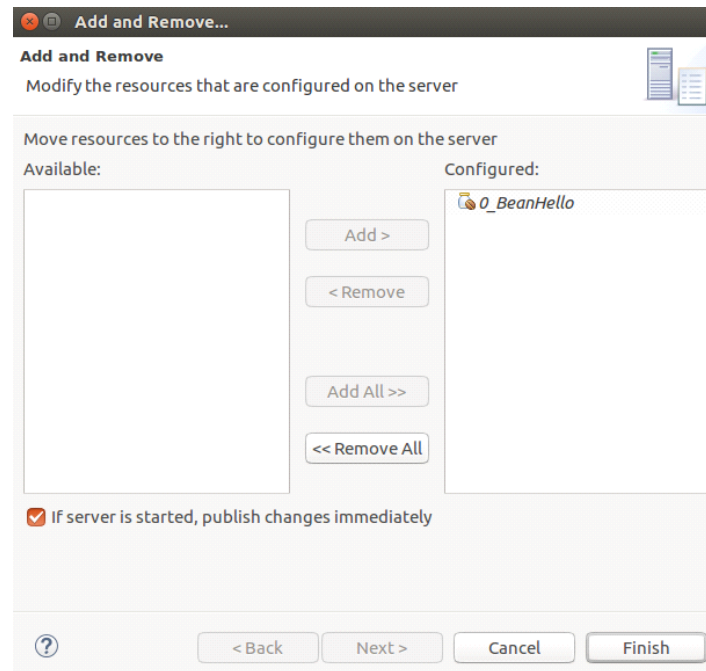
JBoss EJB 3.0 Libraries

Add EJB 3.0 Libraries from one of the declared servers.

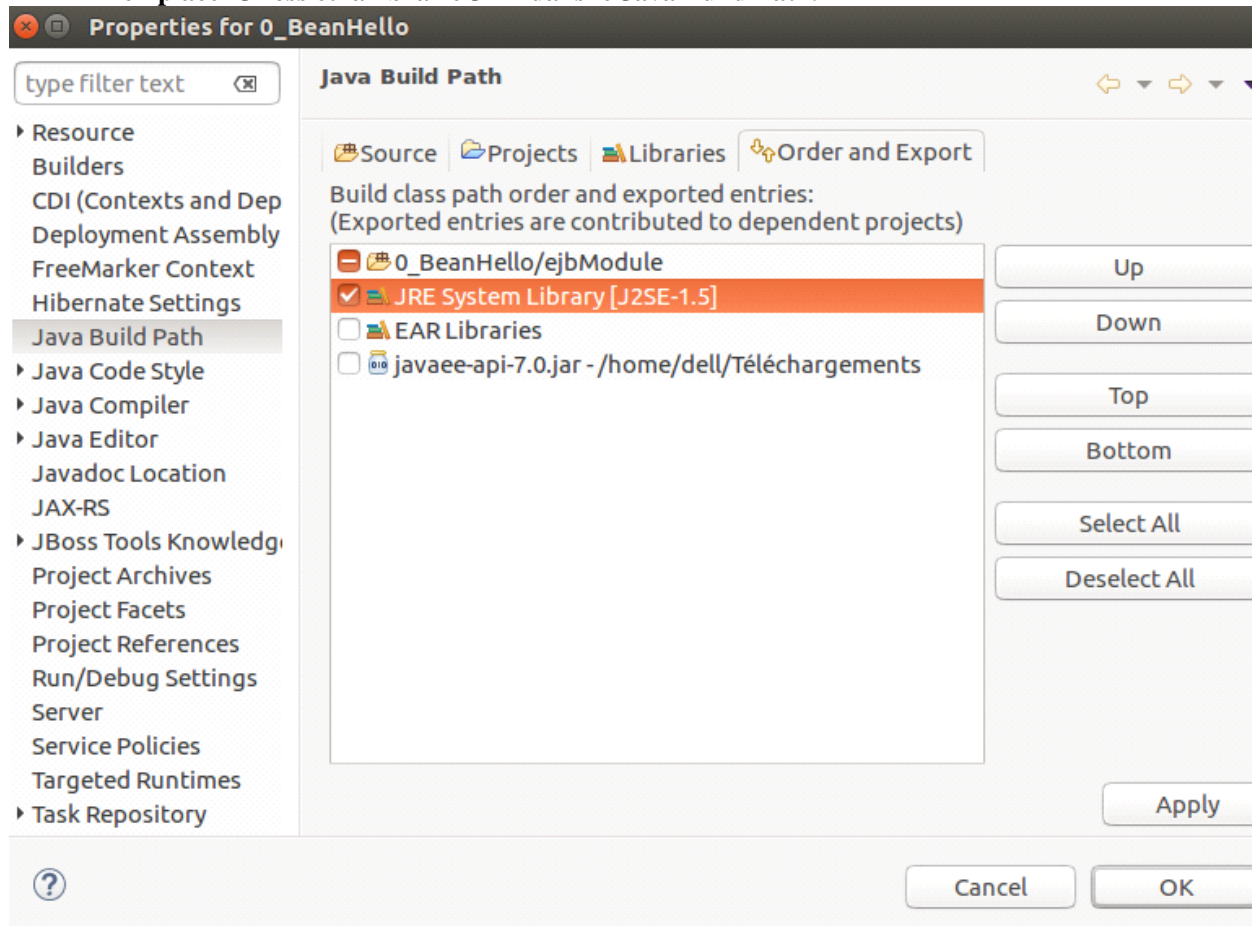
JBoss AS 5.1 [Stopped]

- Ajout du Bean à déployer

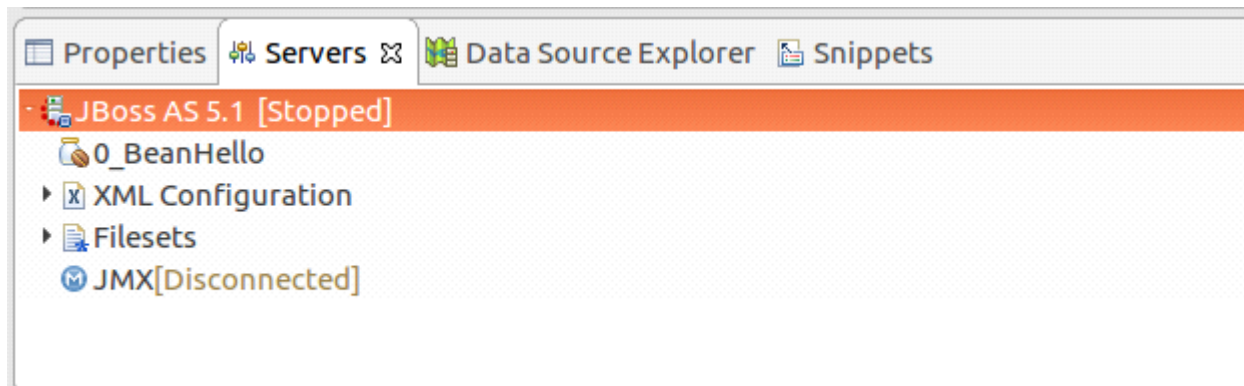
-



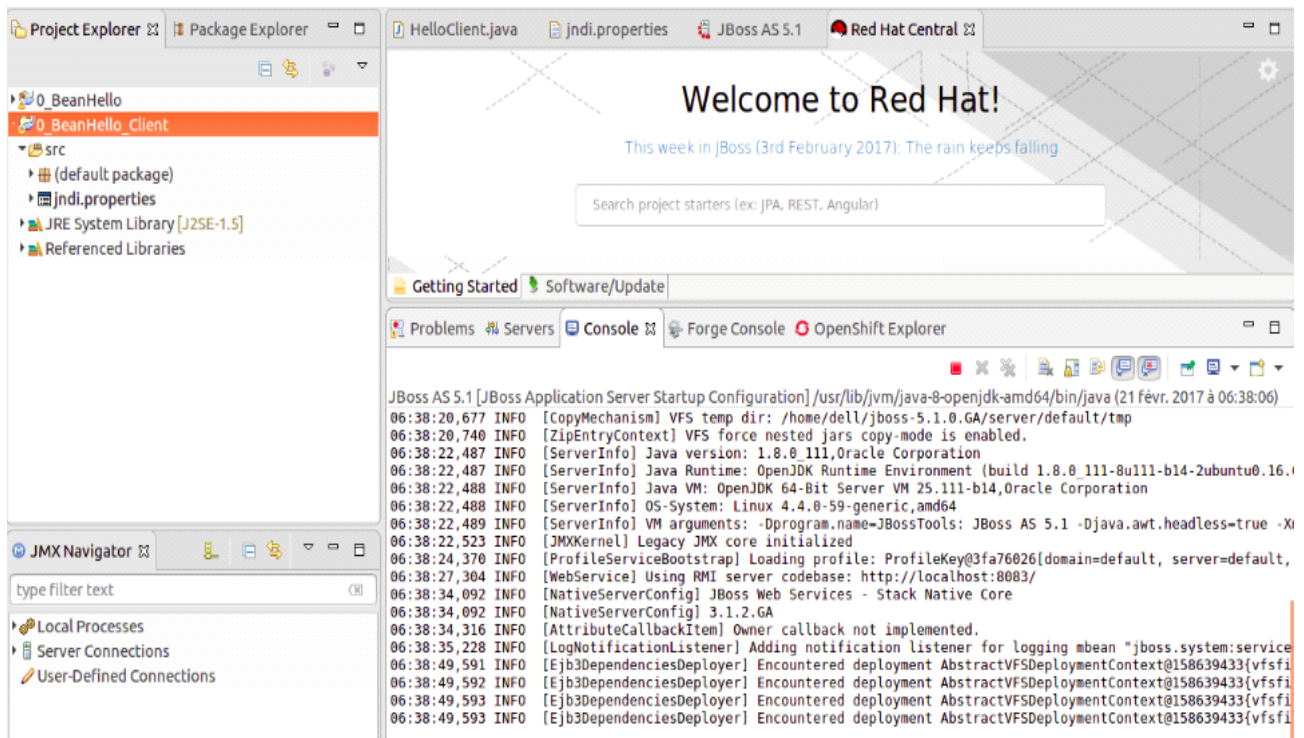
- Remplacer JBoss et la librairie JRE dans le Java Build Path.



- Passer le compilateur en 1.8 dans Java Compiler
- Modifier la version de java en 1.8 dans Projet Facets.
- Changer le Targeted Runtimes pour notre serveur.



- Démarré le serveur



- Création d'un projet Java pour le client.

New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location
Location:

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'java-8-openjdk-amd64')

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files

Working sets

☐ Add project to working sets

Working sets:

New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location
Location:

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'java-8-openjdk-amd64')

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files

Working sets

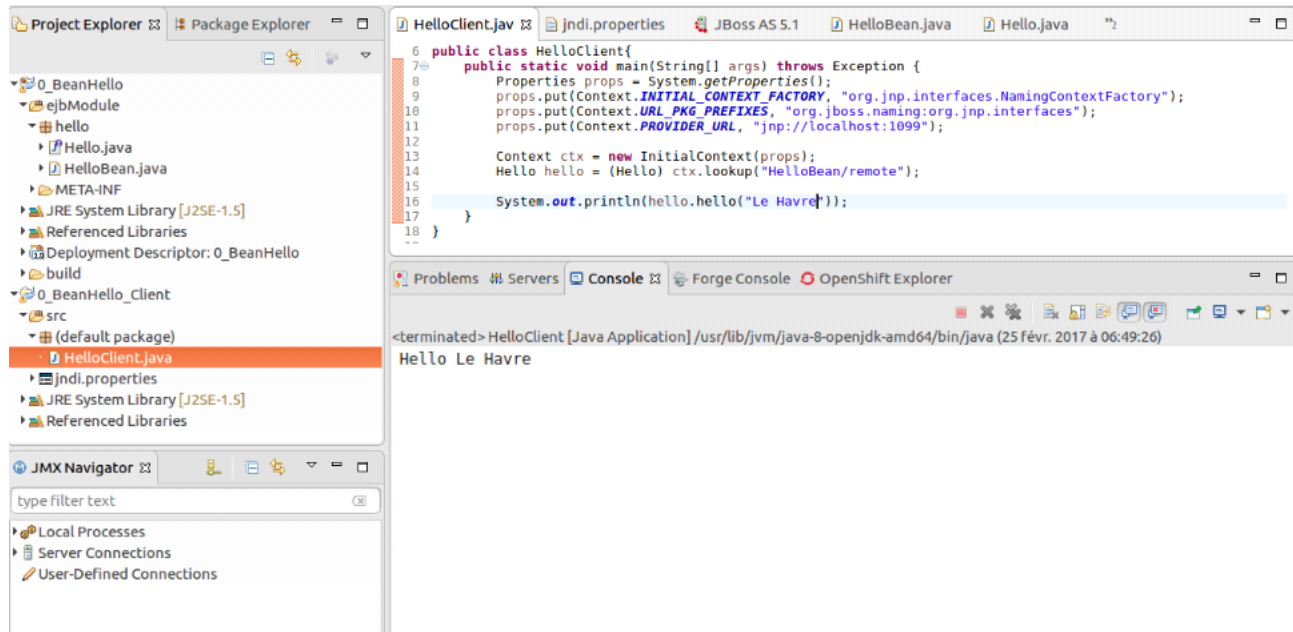
☐ Add project to working sets

Working sets:

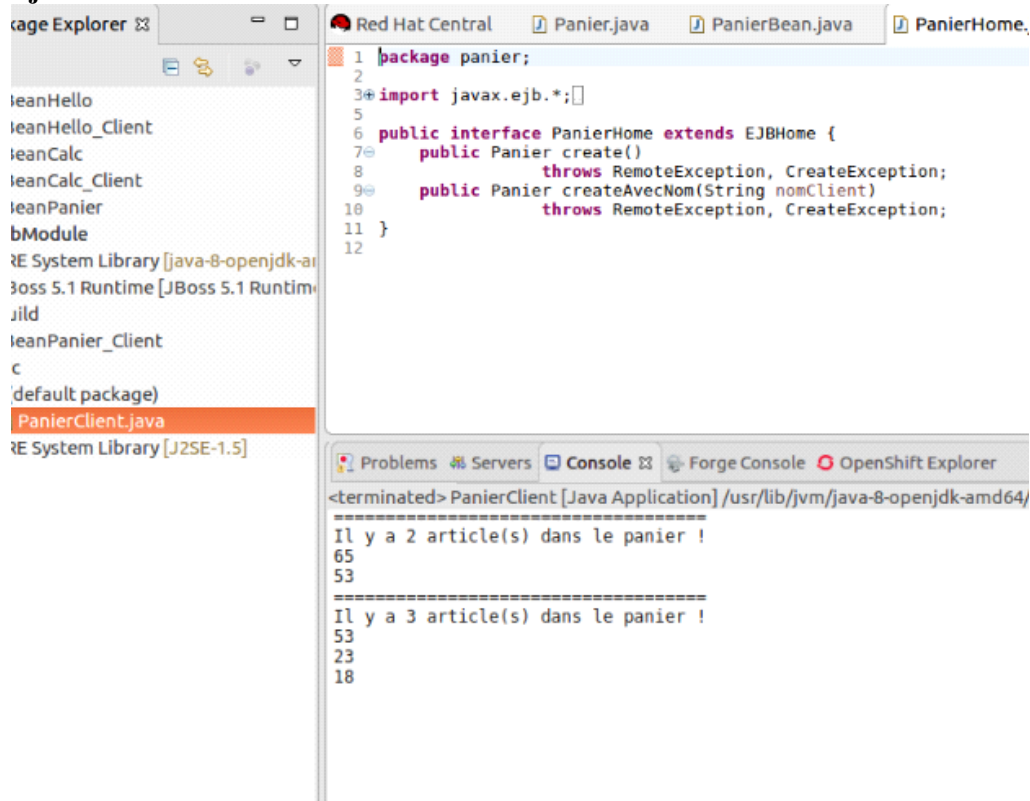
4) Exécution des projets du TP.

Déploiement et exécution des exemples EJB 3.0 du TP avec Eclipse et JBoss 5.1

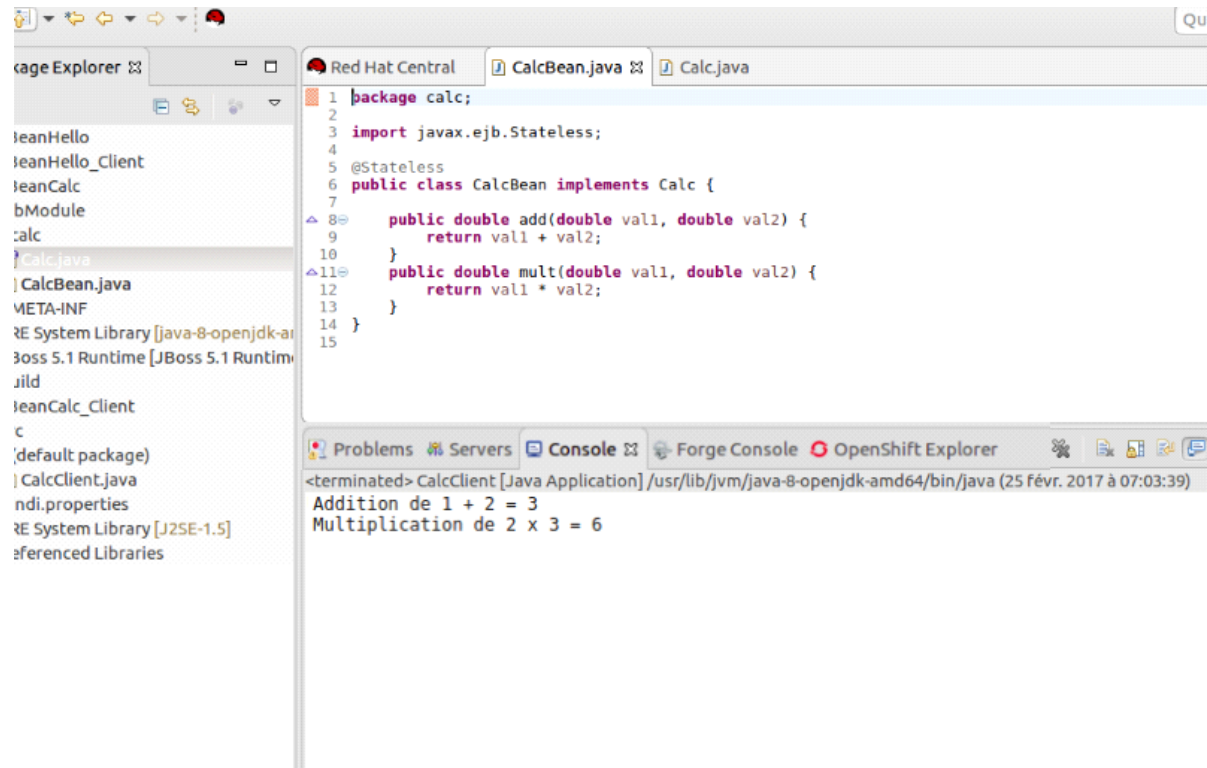
• Projet BeanHello



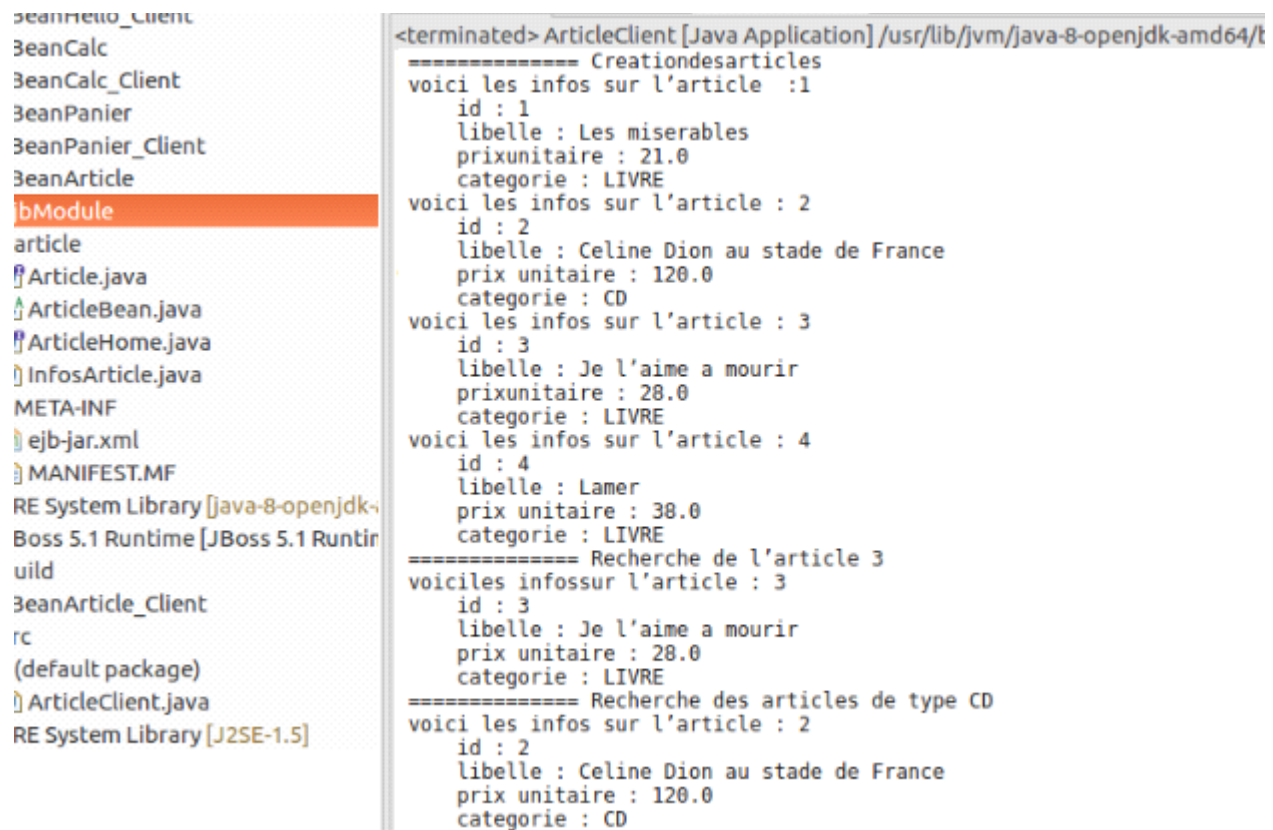
• Projet BeanPanier



- **Projet BeanCalc**



- **Projet BeanArticle**



Bibliographie :

M. Claude DUVALLET, Cours et TP : <http://litis.univ-lehavre.fr/~duvallet/enseignements/enseignements-JEE-fr.php>