JEE : exemple d’application

Description de l’application
Le bean entity
Connexion à la base avec Glassfish
Le bean session
Le déploiement
Un client java utilisant JNDI
Un autre client java déployé par Java Web Start

Description de l’application

Gestion (très primitive!) d’une bibliothèque :

- on gère des livres (auteur et titre)
- on veut pouvoir :
  - ajouter des livres
  - lister les titres
  - récupérer un livre

Le bean entity

```
package biblio;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.NamedQuery;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```
@Entity
@NamedQuery(name="allbooks",query="select b.title from Book AS b")
public class Book implements Serializable {
    private static final long serialVersionUID = 1L;
    private Long id;
    private String author;
    private String title;

    public Book() {}

    public Book(String author, String title) {
      this.author = author; this.title = title;
    }
```

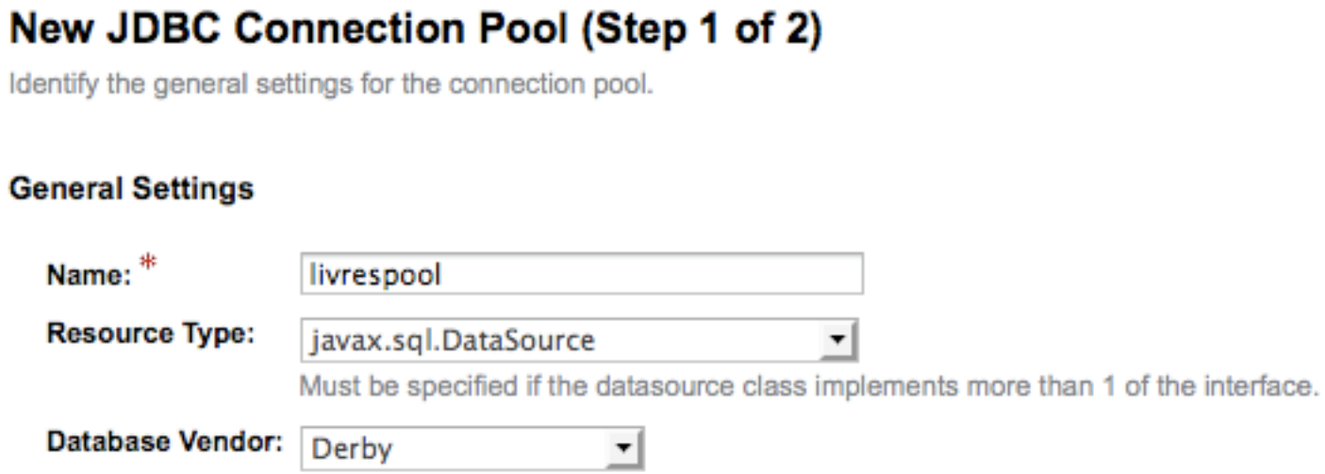
Le bean entity

```
public String getAuthor() { return author; }
public void setAuthor(String author) { this.author = author;}

public String getTitle() { return title; }
public void setTitle(String title) { this.title = title; }

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }

@Override
public int hashCode() { int hash=0; hash+=(id!=null ? id.hashCode():0);
    return hash;}

@Override
public boolean equals(Object object) {
  if (!(object instanceof Book))return false; Book other=(Book) object;
  if((this.id==null && other.id!=null)||(this.id!=null &&
      !this.id.equals(other.id))) return false;
  return true;}
}
```

Connexion à la base avec Glassfish

La procédure est similaire sur Jboss

- Tout d'abord il faut créer un nouveau pool de connexion : (ressources/jdbc/ connection Pools), Derby est la base de données embarquée avec Glassfish. Elle n'est pas démarrée par défaut. On peut la démarrer par la commande : `asadmin start-database`



Connexion à la base avec Glassfish

Additional Properties (6)

Add Property | Delete Properties

	Name	Value
☐	User	APP
☐	DatabaseName	biblio
☐	ConnectionAttributes	;create=true
☐	Password	toto
☐	ServerName	localhost
☐	PortNumber	1527

Connexion à la base avec Glassfish

- Création d'une source de données JDBC : (ressources/jdbc/jdbc ressources)

Resources (3)

New... | Delete | Enable | Disable

	JNDI Name	Enabled	Connection Pool	Description
☐	jdbc/__TimerPool	true	__TimerPool	
☐	jdbc/__default	true	DerbyPool	
☐	jdbc/__CallFlowPool	true	__CallFlowPool	

New JDBC Resource

OK

Specify a unique JNDI name that identifies the JDBC resource you want to create. Name must contain only alphanumeric, underscore, dash, or dot characters.

JNDI Name: * jdbc/livresDS

Pool Name: * livrespool

Use the JDBC Connection Pools page to create new pools

Description: biblio data source

Status: ☑ **Enabled**

Le fichier persistence.xml

- Il s'agit de définir l'unité de persistance et donc de configurer vers quelle source de données les entity beans sont mappés.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://
java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="Biblio-ejbPU">
    <provider>oracle.toplink.essentials.PersistenceProvider</provider>
    <jta-data-source>jdbc/livresDS</jta-data-source>
    <properties>
      <property name="toplink.ddl-generation" value="create-tables"/>
    </properties>
  </persistence-unit>
</persistence>
```

Le bean session

- On n'accède pas directement aux entity beans mais par l'intermédiaire d'un session bean. Ici, il ne sera accédé qu'à distance donc seule l'interface `Remote` est définie.

```
package biblio;

import javax.ejb.Remote;
import java.util.List;

@Remote
public interface BiblioBeanRemote {

    public List<String> liste();
    public void ajouter(String author, String title);
    public Book getLivre(String title);

}
```

Le bean session

- Aucun état n'est à conserver dans les traitements demandés par les clients, nous utiliserons donc un session bean stateless.

```
package biblio;

import javax.persistence.*;
import javax.ejb.Stateless;
import java.util.List;
import javax.persistence.Query;

@Stateless
public class BiblioBeanBean implements BiblioBeanRemote {

  // correspond à l'unité de persistance définie dans
  // le fichier persistence.xml
  @PersistenceContext(unitName = "Biblio-ejbPU")
  EntityManager persistance;
```

Le bean session

```
public List<String> liste() {
  // ici nous utilisons la requête nommée définie par
  // annotation dans l'entity bean
  Query q = persistance.createNamedQuery("allbooks");
  return q.getResultList();
}

public void ajouter(String author, String title) {
  // simplement en utilisant l'entitymanager
  // on rend persistant le livre créé
  Book b = new Book();
  b.setTitle(title);
  b.setAuthor(author);
  persistance.persist(b);
}
```

Le bean session

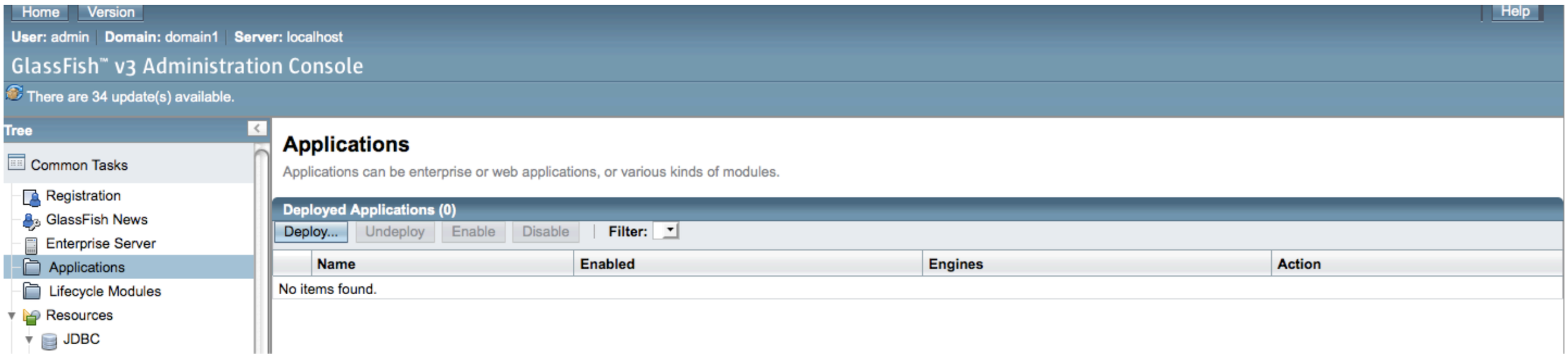
```
public Book getLivre(String title) {
    // ici création d'une requête qui recherche un livre
    // par titre, utilisation de getSingleResult au lieu
    // de getResultList, une seule réponse est attendue
    // sinon exceptions
    String texteRequete="SELECT b FROM Book AS b WHERE
                         b.title=:btitle";
    Query requete = persistance.createQuery(texteRequete);
    Book resultat = null;
    try {
      resultat = (Book)requete.getSingleResult();
    } catch (NonUniqueResultException e) {
    } catch (EntityNotFoundException ee) {
    }
    return resultat;
  }
}
```

Le déploiement

- Le déploiement se fait sous forme d'un fichier .jar :

```
{gembloux-roos-~/Biblio/Biblio-ejb} jar tvf Biblio-ejb.jar
     0 Sun Feb 08 21:00:02 CET 2009 META-INF/
    95 Sun Feb 08 21:00:00 CET 2009 META-INF/MANIFEST.MF
   594 Sun Feb 08 20:42:54 CET 2009 META-INF/persistence.xml
     0 Sun Feb 08 20:43:00 CET 2009 biblio/
  1907 Sun Feb 08 21:00:02 CET 2009 biblio/BiblioBeanBean.class
   394 Sun Feb 08 21:00:02 CET 2009 biblio/BiblioBeanRemote.class
  2152 Sun Feb 08 21:00:02 CET 2009 biblio/Book.class
{gembloux-roos-~/NetBeansProjects/Biblio/Biblio-ejb}
```

Le déploiement avec Glassfish



Le déploiement avec Glassfish

Deploy Applications or Modules

OK Can

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

* Indicates required

Location:

- (•) **Packaged File to Be Uploaded to the Server**
  /Users/roos/Documents/Enseignement/cara/TP/jee/exemple/Bib [Parcourir...]
- ( ) **Local Packaged File or Directory That Is Accessible from the Enterprise Server**
  [Browse Files...] [Browse Folders...]

Type: * EJB Jar

Application Name: * Biblio-ejb

Status: [x] **Enabled**

Run Verifier: [ ] **Enabled**

Force Redeploy: [ ]
Force redeployment if this application is already deployed.

Libraries:
A comma-separated list of library JAR files. Specify the library JAR files by their relative or absolute paths. Specify relative paths relative to *instance-root*`/lib/applibs`. The libraries are made available to t
application in the order specified.

Description:

Un client Java utilisant JNDI

```
package biblio;
import javax.naming.Context;

public class Main {

  public static void main(String[] args) throws Exception {
    Context ctx = new javax.naming.InitialContext();
    BiblioBeanRemote bb = (BiblioBeanRemote)
                ctx.lookup(BiblioBeanRemote.class.getName());
    bb.ajouter("Balzac", "Eugenie Grandet");
    bb.ajouter("Hugo", "Les Miserables");
    java.util.List<String> lt = bb.liste();
    for(String t:lt) System.out.println(t);
  }

}
```

Un client Java utilisant JNDI

- Nous utilisons JNDI pour retrouver le session Bean, il nous faut définir un fichier `jndi.properties` qui permet de spécifier où rechercher :

```
java.naming.factory.initial=
      com.sun.enterprise.naming.SerialInitContextFactory

java.naming.factory.url.pkgs=com.sun.enterprise.naming

java.naming.factory.state=
   com.sun.corba.ee.impl.presentation.rmi.JNDIStateFactoryImpl

org.omg.CORBA.ORBInitialHost=localhost
org.omg.CORBA.ORBInitialPort=3700
```

Un client Java utilisant JNDI

- Il faut que les codes de `Book.class` et `BiblioBeanRemote.class` soient présents, préciser les bonnes librairies dans le `CLASSPATH` et que le fichier `jndi.properties` soit à la racine :

```
{gembloux-roos-~/jee/Biblio-app-client//jar} ls -R
biblio/          jndi.properties

./biblio:
BiblioBeanRemote.class  Book.class              Main.class
{gembloux-roos-~/jee/Biblio-app-client/build/jar}
```

Un client Java utilisant JNDI

```
{gembloux-roos-~/Documents/.../jee/exemple/Biblio-app-client/build/jar} java -
classpath .:/Users/roos/glassfishv3/glassfish/lib/appserv-deployment-client.jar:/
Users/roos/glassfishv3/glassfish/lib/appserv-ext.jar:/Users/roos/glassfishv3/
glassfish/lib/appserv-rt.jar:/Users/roos/glassfishv3/glassfish/lib/javaee.jar
biblio/Main
1 févr. 2010 12:16:11
com.sun.enterprise.transaction.JavaEETransactionManagerSimplified initDelegates
INFO: Using com.sun.enterprise.transaction.jts.JavaEETransactionManagerJTSDelegate
as the delegate
Eugenie Grandet
Les Miserables
{gembloux-roos-~/Documents/.../jee/exemple/Biblio-app-client/build/jar}
```

Un autre client Java déployé avec Java Web Start

- Une autre façon pour déployer un client est d'utiliser AAC, un container pour client et Java Web Start pour effectivement le déployer sur le poste client. Pour cela, nous allons construire une archive "enterprise application" avec le suffixe ear.

```
{gembloux-roos-~/Documents/Enseignement/cara/TP/jee/build} ll
total 32
-rw-r--r--  1 roos  staff  5975  9 fév 00:02 Biblio-app-client.jar
-rw-r--r--  1 roos  staff  5993  9 fév 00:05 Biblio-ejb.jar
{gembloux-roos-~/Documents/.../build} jar tvf Biblio-app-client.jar
     0 Mon Feb 09 00:01:06 CET 2009 META-INF/
    60 Mon Feb 09 00:01:06 CET 2009 META-INF/MANIFEST.MF
     0 Sun Feb 08 21:05:40 CET 2009 J2EE.dpf
   353 Sun Feb 08 19:54:06 CET 2009 META-INF/application-client.xml
     0 Sun Feb 08 21:00:00 CET 2009 META-INF/lib/
     0 Mon Feb 09 00:00:00 CET 2009 biblio/
   394 Sun Feb 08 21:22:04 CET 2009 biblio/BiblioBeanRemote.class
  2152 Sun Feb 08 21:24:10 CET 2009 biblio/Book.class
  1305 Sun Feb 08 23:58:08 CET 2009 biblio/Main2.class
{gembloux-roos-~/Documents/Enseignement/cara/TP/jee/build}
```

Un autre client Java déployé avec Java Web Start

- Nous remarquons que la classe cliente a changé. L'intérêt du container client est de ne plus avoir à utiliser JNDI mais d'utiliser l'injection de dépendances grâce aux annotations :

```
package biblio;
import javax.ejb.EJB;

public class Main2 {

  @EJB
  private static BiblioBeanRemote bb;

  public static void main(String[] args) throws Exception {
       bb.ajouter("Balzac", "Eugenie Grandet");
       bb.ajouter("Hugo", "Les Miserables");
       java.util.List<String> lt = bb.liste();
       for(String t:lt) System.out.println(t);
   }

}
```

Un autre client Java déployé avec Java Web Start

```
{gembloux-roos-~/Documents/.../TP/jee/build} jar tvf Biblio-ejb.jar
     0 Sun Feb 08 19:54:08 CET 2009 META-INF/
    95 Sun Feb 08 19:54:06 CET 2009 META-INF/MANIFEST.MF
     0 Sun Feb 08 19:54:06 CET 2009 biblio/
   617 Sun Feb 08 19:54:06 CET 2009 META-INF/persistence.xml
  1907 Sun Feb 08 19:54:06 CET 2009 biblio/BiblioBeanBean.class
   394 Sun Feb 08 19:54:06 CET 2009 biblio/BiblioBeanRemote.class
  2152 Sun Feb 08 19:54:06 CET 2009 biblio/Book.class
{gembloux-roos-~/Documents/Enseignement/cara/TP/jee/build}
```

Un autre client Java déployé avec Java Web Start

- Déploiement :

Deploy Applications or Modules

OK Cancel

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

* Indicates required field

Location: **Packaged File to Be Uploaded to the Server**

/Users/roos/Documents/Enseignement/cara/TP/jee/build/Biblio. Parcourir...

Local Packaged File or Directory That Is Accessible from the Enterprise Server

Browse Files... Browse Folders...

Type: * Enterprise Application

Application Name: * Biblio

Virtual Servers: server

Associates an Internet domain name with a physical server

Status: **Enabled**

Precompile JSPs: **Enabled**

Precompile JSPs, deploy only resulting class files

Run Verifier: **Enabled**

Force Redeploy:

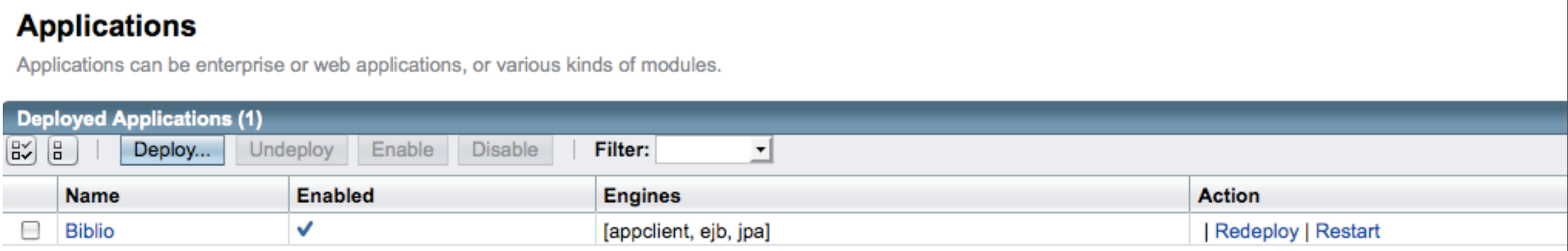
Force redeployment if this application is already deployed.

Java Web Start: **Enabled**

Libraries:

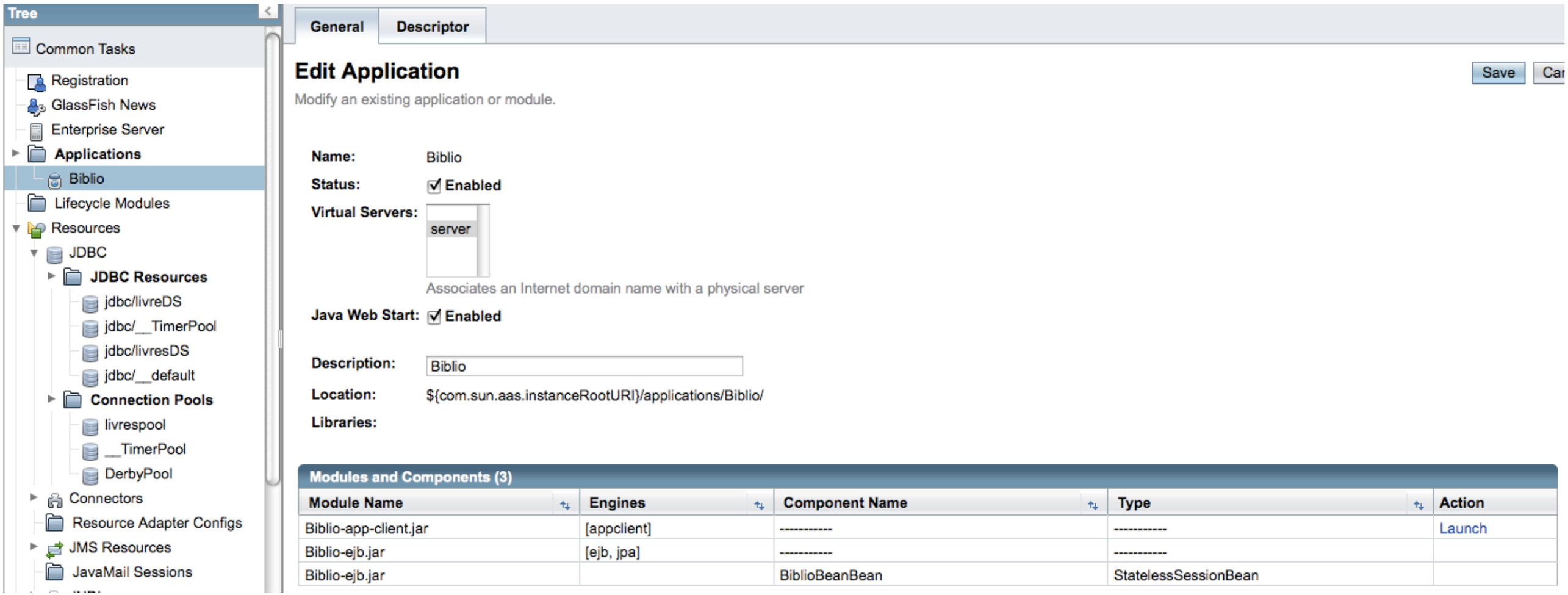
A comma-separated list of library JAR files. Specify the library JAR files by their relative or absolute paths. Specify relative paths relative to *instance-root*`/lib/applibs`. The libraries are made available to

Un autre client Java déployé avec Java Web Start



- Il faut maintenant déployer l'application cliente avec son conteneur AAC toujours avec la console d'administration de Glassfish en utilisant le lien launch sur le module client.

Un autre client Java déployé avec Java Web Start



Un autre client Java déployé avec Java Web Start

Application Client Launch Page

Launch Ba

If the server or listener is not running, links may not work. In such case, check the status of the server instance.

Application: Biblio

Module: Biblio-app-client.jar

Links: http://localhost:8080/Biblio/Biblio-app-client

Arguments:

Arguments to append to the URL for launching the application; for example, `arg=first&arg=second`

Un autre client Java déployé avec Java Web Start