

Exemple d'application JEE

CARA M2 Miage FA-FC

Jean-François Roos - bâtiment M3 extension - bureau 218 - Jean-Francois.Roos@univ-lille1.fr

10 janvier 2016

Sommaire

- 1 Description de l'application
- 2 Le bean entity
- 3 Le bean session
- 4 Compilation
- 5 Connexion à la base avec Glassfish
- 6 Configuration de la persistance
- 7 Le déploiement
- 8 Un client utilisant JNDI

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

Sommaire

- 1 Description de l'application
- 2 Le bean entity**
- 3 Le bean session
- 4 Compilation
- 5 Connexion à la base avec Glassfish
- 6 Configuration de la persistance
- 7 Le déploiement
- 8 Un client utilisant JNDI

```
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;
    }
}
```

```

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) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    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;
}

@Override
public String toString() { return "biblio.Book[id=" + id + "]; }
}

```

Sommaire

- 1 Description de l'application
- 2 Le bean entity
- 3 Le bean session**
- 4 Compilation
- 5 Connexion à la base avec Glassfish
- 6 Configuration de la persistance
- 7 Le déploiement
- 8 Un client utilisant JNDI

L'interface distante

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);
}
```


L'implémentation

```
package biblio;

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

@Stateless
public class BiblioBeanBean implements BiblioBeanRemote {

    @PersistenceContext(unitName = "Biblio-ejbPU")
    EntityManager persistence;

    public List<String> liste() {
        Query q = persistence.createNamedQuery("allbooks");
        try {
            return (List<String>)q.getResultList();
        } catch (ClassCastException e) {
            return null;
        }
    }
}
```

L'implémentation suite

```
public void ajouter(String author, String title) {
    Book b = new Book();
    b.setTitle(title);
    b.setAuthor(author);
    persistance.persist(b);
}

public Book getLivre(String title) {
    String texteRequete = "SELECT b FROM Book AS b WHERE b.title="+
        title;
    Query requete = persistance.createQuery(texteRequete);
    Book resultat = null;
    try {
        resultat = (Book)requete.getSingleResult();
    } catch (NonUniqueResultException e) {
    } catch (EntityNotFoundException ee) {
    }
    return resultat;
}
}
```

Sommaire

- 1 Description de l'application
- 2 Le bean entity
- 3 Le bean session
- 4 Compilation**
- 5 Connexion à la base avec Glassfish
- 6 Configuration de la persistance
- 7 Le déploiement
- 8 Un client utilisant JNDI

La compilation se fait simplement en utilisant la librairie fournie par Glassfish.

```
Terminal x
westmalle-fil-roos--/Documents/Enseignement/cara/TP16/AppliBiblio/composants/src ll
total 12
drwxr-xr-x 3 roos roos 4096  5 janv. 10:14 ./
drwxr-xr-x 4 roos roos 4096 10 janv. 14:26 ../
drwxr-xr-x 2 roos roos 4096 10 janv. 14:47 biblio/
westmalle-fil-roos--/Documents/Enseignement/cara/TP16/AppliBiblio/composants/src javac -classpath ./home/roos/glassfish4/glassfish/lib/javaee.jar
-d ../build/ biblio/*.java
Note: biblio/BiblioBeanBean.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
westmalle-fil-roos--/Documents/Enseignement/cara/TP16/AppliBiblio/composants/src
```

Sommaire

- 1 Description de l'application
- 2 Le bean entity
- 3 Le bean session
- 4 Compilation
- 5 Connexion à la base avec Glassfish**
- 6 Configuration de la persistance
- 7 Le déploiement
- 8 Un client utilisant JNDI

Création d'un pool de connexions

New JDBC Connection Pool (Step 1 of 2) – Mozilla Firefox

localhost:4848/common/index.jsf

Les plus visités Getting Started Le Monde.fr - Actual... Liberation - A la une ... Freenews.fr

Home About Help

User: admin Domain: domain1 Server: localhost

GlassFish™ Server Open Source Edition

Tree

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JDBC Resources
 - JDBC Connection Pools
 - DerbyPool
 - __TimerPool
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
 - Configurations
 - default-config

New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

Next Cancel

* Indicates required field

General Settings

Pool Name: * LivresPool

Resource Type: javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.

Database Driver Vendor: Derby
Select or enter a database driver vendor

Introspect: ☐ Enabled
If enabled, data source or driver implementation class names will enable introspection.

Création d'un pool de connexions : positionnement de propriétés

New JDBC Connection Pool (Step 2 of 2) - Mozilla Firefox

localhost:4848/common/index.jsf

ConnectionAttributes glassfish

Les plus visités Getting Started Le Monde.fr - Actual... Liberation - A la une ... Freenews.fr

Home About... Help

User: admin Role: domain1 Server: localhost

GlassFish™ Server Open Source Edition

Total # of available updates : 48

Tree

- Common Tasks
 - Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JDBC Resources
 - JDBC Connection Pools
 - DerbyPool
 - TimerPool
 - WebSphere
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
 - Configurations
 - default-config
 - server-config
 - Update Tool

Description:

Pool Settings

Initial and Minimum Pool Size: 8 Connections
Minimum and initial number of connections maintained in the pool

Maximum Pool Size: 32 Connections
Maximum number of connections that can be created to satisfy client requests

Pool Resize Quantity: 2 Connections
Number of connections to be removed when pool idle timeout expires

Idle Timeout: 300 Seconds
Maximum time that connection can remain idle in the pool

Max Wait Time: 60000 Milliseconds
Amount of time caller waits before connection timeout is sent

Transaction

Non Transactional Connections: ☐ Enabled
Returns non-transactional connections

Transaction Isolation:
If unspecified, use default level for JDBC Driver

Isolation Level: ☒ Guaranteed
All connections use same isolation level, requires Transaction Isolation

Additional Properties (6)

Select Add Property Delete Properties

Select	Name	Value	Description
<input type="checkbox"/>	ConnectionAttributes	create=true	
<input type="checkbox"/>	User	APP	
<input type="checkbox"/>	DatabaseName	bblio	
<input type="checkbox"/>	ServerName	localhost	
<input type="checkbox"/>	PortNumber	1527	
<input type="checkbox"/>	Password	toto	

Previous Finish Cancel

Création d'une ressource jdbc

New JDBC Resource – Mozilla Firefox

localhost:4848/common/index.jsf

Les plus visités Getting Started Le Monde.fr - Actual... Liberation - A la une ... Freenews.fr

Home About...

User: admin Role: domain1 Server: localhost

GlassFish™ Server Open Source Edition

Total # of available updates : 48

Tree

- Common Tasks
 - Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JDBC Resources
 - jdbc/___TimerPool
 - jdbc/___default
 - jdbc/livresDS
 - JDBC Connection Pools
 - DerbyPool
 - ___TimerPool

New JDBC Resource

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

JNDI Name: * jdbc/livresDS

Pool Name: livresspool
Use the [JDBC Connection Pools](#) page to create new pools

Description:

Status: ☒ Enabled

Additional Properties (0)

[Add Property](#) [Delete Properties](#)

Select	Name	Value	Description
No items found.			

Sommaire

- 1 Description de l'application
- 2 Le bean entity
- 3 Le bean session
- 4 Compilation
- 5 Connexion à la base avec Glassfish
- 6 Configuration de la persistance**
- 7 Le déploiement
- 8 Un client utilisant JNDI

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>org.eclipse.persistence.jpa.
      PersistenceProvider</provider>
    <jta-data-source>jdbc/livresDS</jta-data-source>
    <properties>
<property name="eclipselink.ddl-generation"
      value="drop-and-create-tables"/>
    </properties>
  </persistence-unit>
</persistence>
```

Sommaire

- 1 Description de l'application
- 2 Le bean entity
- 3 Le bean session
- 4 Compilation
- 5 Connexion à la base avec Glassfish
- 6 Configuration de la persistance
- 7 Le déploiement**
- 8 Un client utilisant JNDI

Création de l'archive

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

```
Terminal
westmalle-ftp-roos--/Documents/Enseignement/cara/TP16/AppliBiblio/composants/build ls -R
.:
./ ../ biblio/ META-INF/

./biblio:
./ ../ BibloBeanBean.class BibloBeanRemote.class Book.class

./META-INF:
./ ../ persistence.xml
westmalle-ftp-roos--/Documents/Enseignement/cara/TP16/AppliBiblio/composants/build jar cvf BiblioEjb.jar biblio/ META-INF/
manifeste ajouté
ajout : biblio/(entrée = 0) (sortie = 0) (stockage : 0 %)
ajout : biblio/Book.class(entrée = 1850) (sortie = 940) (compression : 49 %)
ajout : biblio/BibloBeanRemote.class(entrée = 394) (sortie = 248) (compression : 37 %)
ajout : biblio/BibloBeanBean.class(entrée = 1779) (sortie = 969) (compression : 45 %)
entrée META-INF/ ignorée
ajout : META-INF/persistence.xml(entrée = 618) (sortie = 304) (compression : 50 %)
westmalle-ftp-roos--/Documents/Enseignement/cara/TP16/AppliBiblio/composants/build
```

Déploiement dans Glassfish

Deploy Applications or Modules - Mozilla Firefox

localhost:4848/common/index.jsf

elle est fournie

Les plus visités Getting Started Le Monde.fr - Actual... Liberation - A la une ... Freenews.fr

Home About...

User: admin Role: domain1 Server: localhost

GlassFish™ Server Open Source Edition

Total # of available updates : 48

Tree

- Common Tasks
- Domain
 - server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JDBC Resources
 - jdbc/___TimerPool
 - jdbc/___default
 - jdbc/liresDS
 - JDBC Connection Pools
 - DerbyPool
 - ___TimerPool
 - lirespool
 - JMS Resources
 - JNDI

Deploy Applications or Modules

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

Location: ☐ Packaged File to Be Uploaded to the Server

BiblioEjb.jar

☐ Local Packaged File or Directory That Is Accessible from GlassFish Server

Type:

Application Name:

Status: ☒ Enabled

Allows users to access the application.

Run Verifier: ☐ Enabled

Verifies the syntax and semantics of the deployment descriptor. Verifier packages must be installed.

Compatibility: ☐

Supports the backward compatibility of JAR visibility in v2 instead of the stricter Java EE 6 requirements implemented in v3.

Force Redeploy: ☐

Forces redeployment even if this application has already been deployed or already exists.

Keep State: ☐

Retains web sessions, SFSB instances, and persistently created EJB timers between redeployments.

Deployment Order:

A number that determines the loading order of the application at server startup. Lower numbers are loaded first. The default is 100.

Libraries:

Sommaire

- 1 Description de l'application
- 2 Le bean entity
- 3 Le bean session
- 4 Compilation
- 5 Connexion à la base avec Glassfish
- 6 Configuration de la persistance
- 7 Le déploiement
- 8 Un client utilisant JNDI**

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

public class Client {

    /**
     * @param args the command line arguments
     */
    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);
    }
}
```

Compilation du client

```
Terminal x
westmalle-fil-roos-~/Documents/Enseignement/cara/TP16/AppliBiblio/clientjndi/src ls -R
.:
./ ../ biblio/

./biblio:
./ ../ BiblioBeanRemote.java Book.java Client.java
westmalle-fil-roos-~/Documents/Enseignement/cara/TP16/AppliBiblio/clientjndi/src javac -classpath ./home/roos/glassfish4/glassfish/lib/javaee.jar
-d ../build/ biblio/*.java
westmalle-fil-roos-~/Documents/Enseignement/cara/TP16/AppliBiblio/clientjndi/src
```


Le fichier jndi.properties

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
```

Lancement du client

```
Terminal
westmalle-fil-roos ~/Documents/Enseignement/cara/TP16/AppliBiblio/clientjndi/build ls -R
.:
./ ../ biblio/ jndi.properties

./biblio:
./ ../ BiblioBeanRemote.class Book.class Client.class
westmalle-fil-roos ~/Documents/Enseignement/cara/TP16/AppliBiblio/clientjndi/build java -classpath ./home/roos/glassfish4/glassfish/lib/javaee.jar
:/home/roos/glassfish4/glassfish/lib/jndi-properties.jar:/home/roos/glassfish4/glassfish/lib/gf-client.jar biblio.Client
Eugenie Grandet
Les Miserables
Eugenie Grandet
Les Miserables
westmalle-fil-roos ~/Documents/Enseignement/cara/TP16/AppliBiblio/clientjndi/build
```