

2013



Middlewares & WebServices

Dalila ZAHRAOUI
David BURIAN

2012-2013

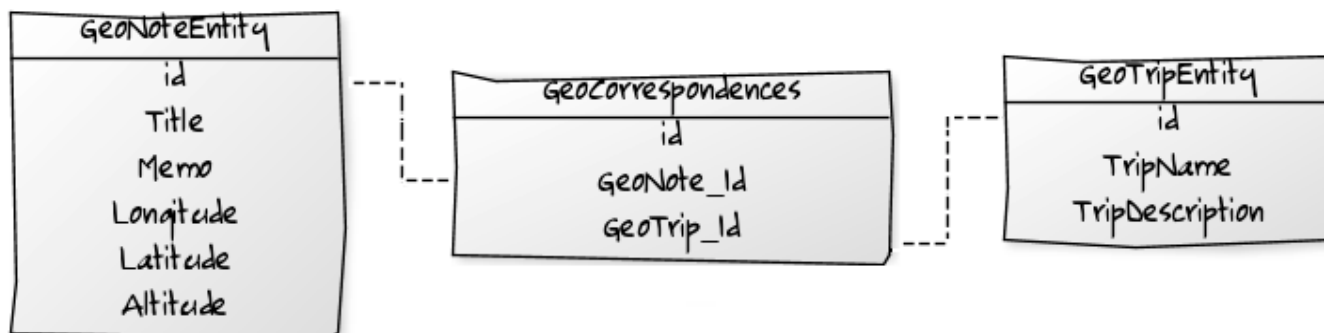


Sommaire

<u>1 - Modélisation de la base de données et liaison avec Netbeans.....</u>	<u>3</u>
<u>2 - Création des EJB.....</u>	<u>4</u>
<u>3 - Création des unités de persistance</u>	<u>5</u>
<u>4 - Création des entités JPA.....</u>	<u>5</u>
<u>5 - Création du web service.....</u>	<u>6</u>
<u>6 - Création de l'EntrepriseApplication</u>	<u>7</u>

1 - Modélisation de la base de données et liaison avec Netbeans

Le point d'entrée de notre application est la création de la base de données :



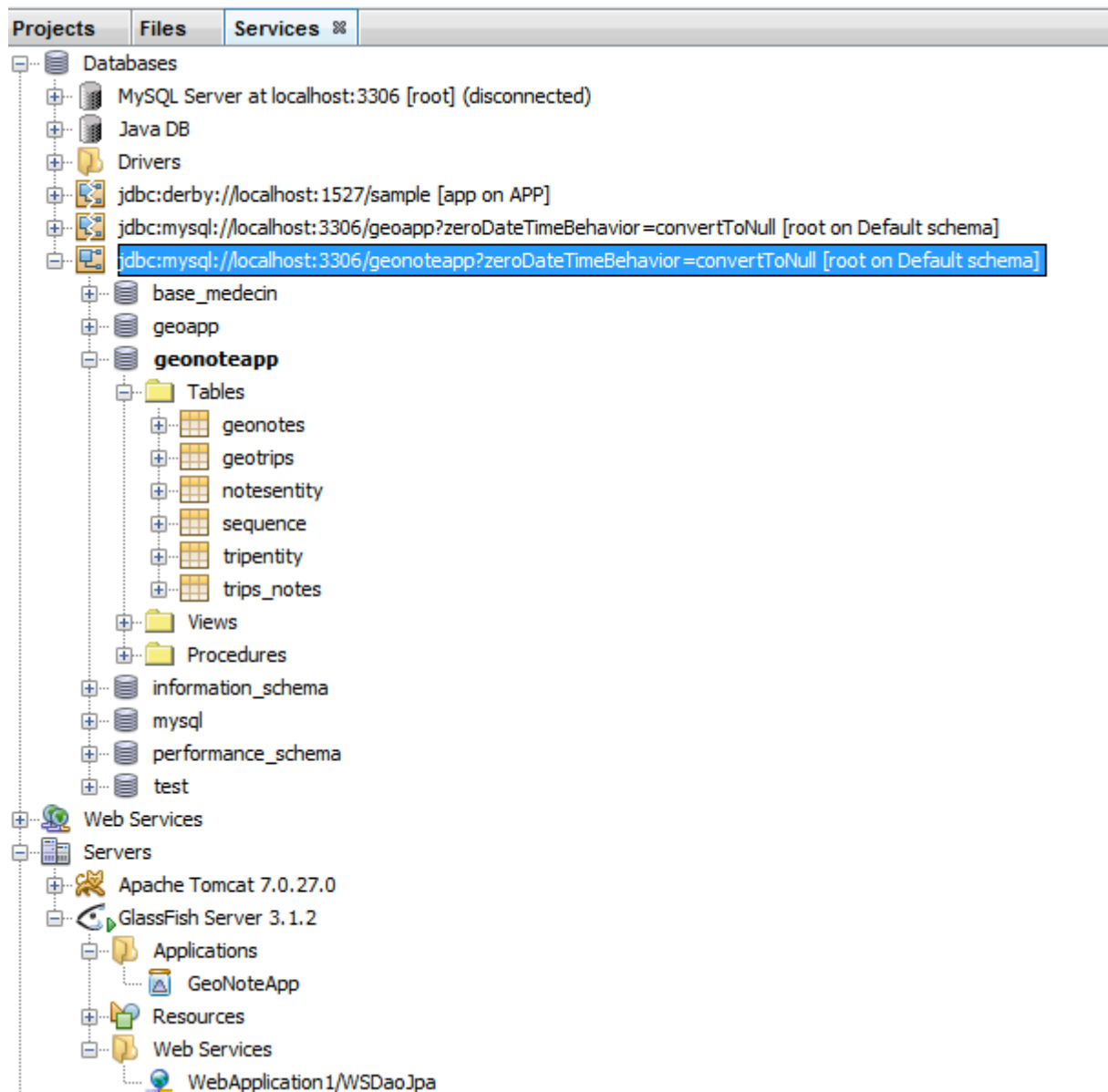
Les tables **GeoNoteEntity** et **GeoTripEntity** sont explicites de par les champs qu'elles contiennent.

L'une décrit les GeoNotes telles qu'elles sont censées être renseignées une fois le terminal mobile du client a saisi les coordonnées GPS de l'endroit, et que l'utilisateur en a défini un titre et un petit mémo descriptif.

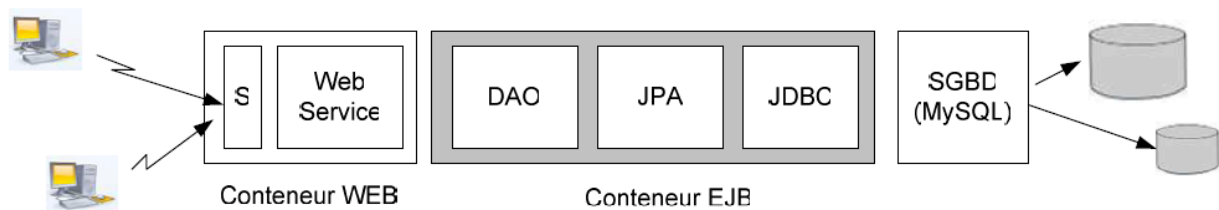
L'autre, GeoTripEntity décrit un parcours donné, en renseignant le nom et la description.

Sachant qu'une GeoNote peut appartenir à plusieurs parcours (GeoTrip) différents, et qu'a priori, un parcours GeoTrip n'a pas un nombre défini constant de GeoNotes attachées, nous créons une table intermédiaire nommée GeoCorrespondances qui permet, grâce aux ID uniques des deux autres tables, de les associer mutuellement.

Nous affichons succinctement le résultat de la connexion de NetBeans à notre serveur de données MySQL :



2 - Création des EJB



Nous avons créé un conteneur EJB sous le nom de **EJBModule1**, puis nous avons créé une ressource JDBC au serveur Glassfish. Elle porte le nom de **jdbc/geoapp_JNDI**. Le JDBC Connection pool porte lui le nom de **geoapp_pool**.

Nous pouvons retrouver l'ensemble de ces informations au niveau du fichier **glassfish-resources.xml** comme suit :

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE resources PUBLIC "-//GlassFish.org/DTD GlassFish Application Server 3.1 Resource Definitions//EN" "http://glassfish.
3 <resources>
4 <jdbc-resource enabled="true" jndi-name="jdbc/geoapp_JNDI" object-type="user" pool-name="geoapp_pool">
5   <description/>
6 </jdbc-resource>
7 <jdbc-connection-pool allow-non-component-callers="false" associate-with-thread="false" connection-creation-retry-attempts="0
8   <property name="URL" value="jdbc:mysql://localhost:3306/geoapp?zeroDateTimeBehavior=convertToNull"/>
9   <property name="User" value="root"/>
10  <property name="Password" value=""/>
11 </jdbc-connection-pool>
12 </resources>
13

```

3 - Création des unités de persistance

Cette couche permettra de configurer la couche JPA. En spécifiant la *DataSource* **jdbc/geoapp_JNDI**, et une non génération des tables par NetBeans (car notre base de données a été préalablement pensée et créé via phpMyAdmin), nous obtenons le fichier **persistence.xml** comme suit :

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.0" 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_2_0.xsd">
3   <persistence-unit name="EJBModule1PU" transaction-type="JTA">
4     <jta-data-source>jdbc/geoapp_JNDI</jta-data-source>
5     <exclude-unlisted-classes>false</exclude-unlisted-classes>
6     <properties/>
7   </persistence-unit>
8 </persistence>
9

```

4 - Création des entités JPA

La création des entités JPA se fait suite à connexion aux tables que nous avons créées dans la base de données **geoapp** dans le serveur de données **MySQL**. On génère ainsi une entité par table de notre base de données.

Pour info, et pour plus de facilité, nous avons utilisé **java.util.List** comme *CollectionType*.

Nous générons ainsi les classes suivantes : **Geonotesentity** & **Geotripentity** dans le package **jpa**.

La couche d'accès aux entités JPA via les SessionBeans :

Cette phase nous permet de créer les classes **DaoJpa** & **Idao** et **IdaoLocal** dans le package **dao**.

C'est à ce niveau-là que nous générons les méthodes qui nous permettent de manipuler les données, notamment les opérations du **CRUD** (création, lecture, modification et suppression).

A titre d'exemple, voici la définition de la méthode nous permettant d'obtenir

```
// liste des GeoNotes

public List<Geonotesentity> getAllGeonotesentity() {

    return em.createQuery("SELECT g FROM Geonotesentity g").getResultList();

}
```

En déployant ce conteneur EJB, nous pourrions obtenir le fichier .JAR qui nous servira à créer le Webservice.

5 - Création du web service

Dans cette partie, nous créons une application Web, nommée par défaut WebApplication1 s'appuyant sur un serveur **GlassFish** et le framework **JavaServer Faces**.

Nous ajoutons ensuite le fichier **EJBModule1.JAR** aux librairies de notre **WebApplication1**.

Nous créons ensuite un webservice que nous nommons **geonote** auquel nous l'interfaçons sur **DaoJpa**. Nous obtenons un fichier que nous nommons **WSDaoJpa**.

En voici un aperçu :

```
19 @WebService(serviceName = "WSDaoJpa")
20 public class WSDaoJpa {
21     @EJB
22     private IdaoLocal dao; // Add business logic below. (Right-click in editor and choose
23     // "Insert Code > Add Web Service Operation")
24
25     @WebMethod
26     public List<Geonotesentity> getAllGeonotesentity() {
27         return dao.getAllGeonotesentity();
28     }
29
30     @WebMethod
31     public List<Geotripentity> getAllGeotripentity() {
32         return dao.getAllGeotripentity();
33     }
34
35 }
36
```

6 - Création de l'EntrepriseApplication

Une fois les phases de tests du Webservice validées, nous pouvons créer l'EntrepriseApplication finale.

Cette application finale est celle qui rassemblera en un projet unique la partie EJB et la partie Webservice.

Après déploiement final, nous remarquons que grâce au test du Webservice, nous pouvons accéder aux méthodes exposées, et ainsi aux données de notre base de données (en passant par les différents Middleware que nous avons créés).

WSDaoJpa Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.util.List geonote.WSDaoJpa.getAllGeonotesentity()

()

public abstract java.util.List geonote.WSDaoJpa.getAllGeotripentity()

()

Voici le résultat final de ce montage :

Les screenshots des tables dans phpMyAdmin nous permettent de faire le lien entre les données affichées suite à la requête SOAP et le contenu initial de nos tables :

	ID	TRIPDESCRIPTION	TRIPNAME
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	1	Visiting momuments in Saint Etienne, Loire, France	History in Saint-Etienne
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	2	Having some good night time in Saint Etienne, Loir...	Thirst Trip in Saint Etienne
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	3	Mustsee expos in Lyon, Rhone, France	Fete des Lumieres in Lyon
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	4	Short circuit for 2 days tourism in Paris,Paris, F...	Tourism in Paris













SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<S:Header/>
<S:Body>
  <ns2:getAllGeotripentity xmlns:ns2="http://geonote/">
</S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getAllGeotripentityResponse xmlns:ns2="http://geonote/">
      <return>
        <id>1</id>
        <tripdescription>Visiting momuments in Saint Etienne, Loire,
France</tripdescription>
        <tripname>History in Saint-Etienne</tripname>
      </return>
      <return>
        <id>2</id>
        <tripdescription>Having some good night time in Saint Etienne, Loire,
France</tripdescription>
        <tripname>Thirst Trip in Saint Etienne</tripname>
      </return>
      <return>
        <id>3</id>
        <tripdescription>Mustsee expos in Lyon, Rhone, France</tripdescription>
        <tripname>Fete des Lumieres in Lyon</tripname>
      </return>
      <return>
        <id>4</id>
        <tripdescription>Short circuit for 2 days tourism in Paris,Paris,
France</tripdescription>
        <tripname>Tourism in Paris</tripname>
      </return>
    </ns2:getAllGeotripentityResponse>
  </S:Body>
</S:Envelope>
```

← T →	ID	ALTITUDE	LATITUDE	LONGITUDE	MEMO	TITLE
  	1	NULL	4.384811	45.447984	Nice Place :)	Carnot, Saint-Etienne
  	2	NULL	4.387279	45.435819	Best Shopping Ever!	Mango, Saint-Etienne
  	3	NULL	4.386334	45.442443	Sweet Church	Jean Jaures, Saint-Etienne
  	4	NULL	4.39039	45.436557	Have a break, have there a drink	Smocking Dog, Saint-Etienne

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:getAllGeonotesentity xmlns:ns2="http://geonote/">
  </S:Body>
</S:Envelope>
```


SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getAllGeonotesentityResponse xmlns:ns2="http://geonote/">
      <return>
        <id>1</id>
        <latitude>4.384811</latitude>
        <longitude>45.447984</longitude>
        <memo>Nice Place :)</memo>
        <title>Carnot, Saint-Etienne</title>
      </return>
      <return>
        <id>2</id>
        <latitude>4.387279</latitude>
        <longitude>45.435819</longitude>
        <memo>Best Shopping Ever!</memo>
        <title>Mango, Saint-Etienne</title>
      </return>
      <return>
        <id>3</id>
        <latitude>4.386334</latitude>
        <longitude>45.442443</longitude>
        <memo>Sweet Church</memo>
        <title>Jean Jaures, Saint-Etienne</title>
      </return>
      <return>
        <id>4</id>
        <latitude>4.39039</latitude>
        <longitude>45.436557</longitude>
        <memo>Have a break, have there a drink</memo>
        <title>Smocking Dog, Saint-Etienne</title>
      </return>
    </ns2:getAllGeonotesentityResponse>
  </S:Body>
</S:Envelope>
```

