UNIVERSITÉ DE RENNES 1

Compte Rendu

CR_SIR_TP2-3

Gahimbare /Ibn Mrabet

2016

ISTIC

JPA :

Le but est de créer et remplir une base de donnée à partir des connexions persistantes JPA .

Entités :

4 entités on était créé pour ce faire :

Class Person :

@Entity

…

```
	@Id
	@GeneratedValue(strategy=GenerationType.AUTO)
	public long getId() {
		return id;
	}

	public String getName() {
		return name;
	}
	public String getMail() {
		return mail;
	}
	@OneToMany( cascade = CascadeType.PERSIST)
	public List<Residence> getResidence() {
		return residence;
	}
   @OneToMany(mappedBy ="proprio" , cascade = CascadeType.PERSIST)
	public List<ElectronicDevice> getDevices() {
		return devices;
	}
```

Class Residence

@Entity

…

```
	@Id

	@GeneratedValue(strategy=GenerationType.AUTO)
	public long getId() {
		return id;
	}
	public int getNoChambre() {
		return noChambre;
	}
	public int getNoSurface() {
		return noSurface;
	}
	@ManyToOne
	public Person getProprio() {
		return proprio ;
	}
	@OneToMany(mappedBy="residence" , cascade=CascadeType.PERSIST)
	public List<Chauffage> getChauffages() {
		return chauffages;
```

```
}
```

Abstract Class IntelligentDevice :

Entité de type classe abstraite méthode d'héritage : ***SINGLE_TABLE*** , colonne descriminante : name= 'Device_type'.

@Entity

…

```
@Id

@GeneratedValue(strategy=GenerationType.AUTO)
public long getId() {
	return id;
}
public String getMarque() {
	return marque;
}
public int getConso() {
	return conso;
}
```

Class ElectronicDevice :

Classe héritant de IntelligentDevice avec un *@DiscriminatorValue= 'ElectronicDevice'* .

@Entity

…

```
public String getType() {

	return type;
}
@ManyToOne
public Person getProprio() {
	return proprio;
}
```

Class Chauffage :

Classe héritant de IntelligentDevice avec un *@DiscriminatorValue= 'chauffage'* .

```
@ManyToOne
public Residence getResidence() {
	return residence;
}
```

Class TestJpa

```
public JpaTest() {
		this.factory = Persistence.createEntityManagerFactory("mysql2");
		this.manageR = factory.createEntityManager();
		this.tx = manageR.getTransaction();
		this.tx.begin();
	}
```

Résultat :

Création de 3 table 

Exemple :

```
Person p1 = new Person();

                    p1.setName("dupond");
                    p1.setMail("dupond@gmail.com");

                    Residence residence1 = new Residence();
                    residence1.setNoChambre(2);
                    residence1.setNoSurface(300);
                    residence1.setProprio(p1);

                    Chauffage ch1 = new Chauffage();
                    ch1.setConso(300);
                    ch1.setMarque("A/C");
                    ch1.setResidence(residence1);

                    ElectronicDevice ed1 = new ElectronicDevice();
                    ed1.setConso(200);
                    ed1.setMarque("Galaxy");
                    ed1.setProprio(p1);
                    ed1.setType("SmartPhone");

                    manageR.persist(p1);
                    manageR.persist(residence1);
                    manageR.persist(ch1);
                    manageR.persist(ed1);
```

Table Person :

id	mail	name
1	dupond@gmail.com	dupond

Table IntelligentDevice :

Device_typ	id	conso	marque	type	fix	proprio	residence
chauffage	1	300	A/C	*NULL*	0	*NULL*	1
ElectronicDevice	2	200	Galaxy	SmartPhone	*NULL*	1	*NULL*

Table Residence :

id	noChambre	noSurface	proprio_id
1	2	300	1

Récupérer de la base de donnée :

Public List<Person> listPerson() : renvoie la liste des personne à partir de la base de données.

```
nombre de personne : 1
Person [id=1, name=dupond, mail=dupond@gmail.com,
```

Public List<Residence> listResidence() : renvoie la liste des résidence à partir de la base de données.

```
Residence [id=1, noChambre=2, noSurface=300, proprio=Person [id=1, name=dupond, mail=dupond@gmail.com,
```

Connexion à une base SQL :

Nous avons pu nous connecter sur une base données SQL distante et en local :

```
<!--  CONNEXION DISTANTE -->
 <persistence-unit name="mysql">
        <properties>
            <!--
            <property name="hibernate.ejb.cfgfile" value="/hibernate.cfg.xml"/>
            <property name="hibernate.hbm2ddl.auto" value="create"/>
            -->
            <property name="hibernate.hbm2ddl.auto" value="create"/>
            <property name="hibernate.archive.autodetection" value="class, hbm"/>
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.connection.driver_class" value="com.mysql.jdbc.Driver"/>
            <property name="hibernate.connection.password" value="*******"/>
            <property name="hibernate.connection.url" value="jdbc:mysql://anteros.istic.univ-rennes1.fr/base_12010154"/>
            <property name="hibernate.connection.username" value="user_12010154"/>
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>
            <property name="hibernate.c3p0.min_size" value="5"/>
            <property name="hibernate.c3p0.max_size" value="20"/>
            <property name="hibernate.c3p0.timeout" value="300"/>
            <property name="hibernate.c3p0.max_statements" value="50"/>
            <property name="hibernate.c3p0.idle_test_period" value="3000"/>
        </properties>
    </persistence-unit>
    <!-- LOCAL -->
    <persistence-unit name="mysql2">
        <properties>

         <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/jpa" />
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="" />

            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.hbm2ddl.auto" value="create" />

         </properties>
```

Mis en évidence du problème N+1 Select :

Les temps de réponse sont bien meilleur avec un Joinftch car une seul requêtre est effectué.

Par contre avec N+1 select car plusieurs fois est exécuté la requête.

Servlet :

Nous avons le plug-in de tomcat7

```
<plugins>
            <!-- * Start of user code for plugins -->
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
```

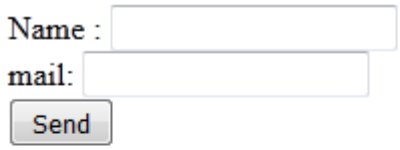
```
                <configuration>
                    <path>/</path>
                </configuration>
            </plugin>

        </plugins>
```

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building testjpa 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> tomcat7-maven-plugin:2.2:run (default-cli) > process-classes @ testjpa >>>
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ testjpa ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ testjpa ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] <<< tomcat7-maven-plugin:2.2:run (default-cli) < process-classes @ testjpa <<<
[INFO]
[INFO] --- tomcat7-maven-plugin:2.2:run (default-cli) @ testjpa ---
[INFO] Démarrage du war sur http://localhost:8080/
[INFO] Utilisation de la configuration existante du serveur Tomcat sur G:\nouveau dossier\testjpa\target\tomcat
[INFO] create webapp with contextPath:
févr. 21, 2016 5:56:42 PM org.apache.coyote.AbstractProtocol init
INFOS: Initializing ProtocolHandler ["http-bio-8080"]
févr. 21, 2016 5:56:42 PM org.apache.catalina.core.StandardService startInternal
INFOS: Starting service Tomcat
févr. 21, 2016 5:56:42 PM org.apache.catalina.core.StandardEngine startInternal
INFOS: Starting Servlet Engine: Apache Tomcat/7.0.47
févr. 21, 2016 5:56:50 PM com.sun.jersey.api.core.PackagesResourceConfig init
INFOS: Scanning for root resource and provider classes in the packages:
  fr.istic.sir.rest
févr. 21, 2016 5:56:50 PM com.sun.jersey.api.core.ScanningResourceConfig logClasses
INFOS: Root resource classes found:
  class fr.istic.sir.rest.SampleWebService
févr. 21, 2016 5:56:50 PM com.sun.jersey.api.core.ScanningResourceConfig init
```

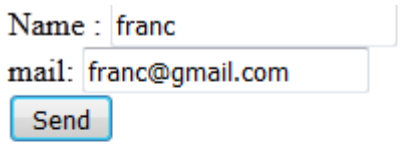
Première page :

http://localhost:8080/index.html

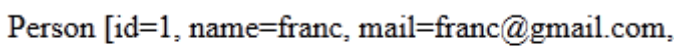
Page sous forme de formulaire pour ajouter une nouvelle personne :



On peut ajouter un utilisateur sur notre base de données en appuyant sur Send.



Comme reponse sur la page HTML la récupération de la liste des personnes de la base de données :




```
@WebServlet(name="userInfo" ,
urlPatterns={"/UserInfo"})
public class UserInfo extends HttpServlet{

	public void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {
		response.setContentType("text/html");

		PrintWriter out = response.getWriter();
		String nom = request.getParameter("name");
		String mail =  request.getParameter("mail");

		JpaTest jpa = new JpaTest();
		jpa.createPerson(nom, mail);
		List<Person> p = jpa.listPerson();

		for(Person result : p ){
			out.print(result.toString());
		}


	}
```