

Compte Rendu

Tp JPA/ Servlet, Système d'information réparti

Réalisé par :

-Kaoutar Bennouna
-Mohammed Ayoub Boukadida

Encadré par Mr:

-Olivier Barais

M1 MIAGE

21 février 2016

Compte Rendu

Tp JPA/ Servlet, Système d'information réparti

Table des matières

I. Tp2 :	2
1. TRANSFORMER UNE CLASSE EN ENTITE :	2
2. Q3 : PEUPLEMENT DE LA BASE DE DONNEES	7
3. Q4 : CONNEXION A LA BASE DE DONNEES MYSQL:	10
4. HERITAGE :	11
II. Tp3 : SERVLET	16
1. QUESTION 1 : MODIFICATION DU POM.XML	17
2. QUESTION 2 :	18
3. QUESTION 3 : CREATION DE SERVLET :	18
4. CREATION DU FICHIER MYFORM.HTML :	19
5. Q5 : RETOUR SUR OPOWER :	20
6. QUESTION 6 : REST :	24

I. Tp2 :

L'objectif de ce projet est de réaliser une application réseau du type réseau social où les gens peuvent comparer leur consommation électrique.

Une personne a une ou plusieurs maisons et un ou plusieurs ElectronicDevice, la maison elle-même a un ou plusieurs chauffages.

1. Transformer une classe en entité :

Suivant le modèle de conception, nous aurons les classes suivantes : Personne, Maison, Devices, Chauffages. Ces classes sont présentes dans le package domain.

Pour répondre aux questions 1 et 2 nous avons utilisés :

- @Entity : écrit au-dessus du nom de la classe, elle indique que la classe est une entité
- @Id: écrit au-dessus du getter de l'id de la personne, cette notation désigne la clé primaire de l'entité
- @GeneratedValue: écrit juste au-dessous de @Id, cette annotation indique que la clé primaire sera générée automatiquement par le SGBD.
- @ManyToOne: écrit au-dessus du getter qui renvoie la clé étrangère du propriétaire. Dans la classe owner qui recherche à avoir la liste des pets
- @OneToMany: écrit au-dessus du getter qui renvoie la liste des informations de la table qui est associée à celle-ci avec une clé étrangère. Dans la classe pets juste au-dessus du getter du owner.

Pour utiliser ces annotations il faut importer les packages correspondants depuis javax.Persistence.

1.1. Class Person :

```
package domain;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;

public class Person {

    //la liste des attributs liés à une personne (id, nom, prenom, mail, sa ou
    //ses résidences et son ou ses electronicDevices
    private long id; //son id
    private String nom; //nom
    private String prenom; //prenom
    private String email; //email
    private List<Home> Residence; //liste des résidences
    private List<ElectronicDevices> devices; //liste des Devices
```

```
//le constructeur de la classe Personne
public Person(){

}
```

```
public Person(long id, String nom, String prenom, String email, List<Home>
maisons, List<ElectronicDevices> devices ){
    this.id=id; //id
    this.nom=nom; //nom
    this.prenom=prenom; //prenom
    this.email=email; //email
    this.Residence=maisons; //maisons
    this.devices=devices; //Devices
}
public Person( String nom, String prenom, String email, List<Home>
maisons){

    this.nom=nom; //nom
    this.prenom=prenom; //prenom
    this.email=email; //email
    this.Residence=maisons; //maisons

}
```

```
@Id
@GeneratedValue
public long getId() {
    return id;
}

public void setId(long id) {
    this.id = id;
}

//getter et setter du nom
public String getNom() {
    return nom;
}

public void setNom(String nom) {
    this.nom = nom;
}

//getter et setter du prénom
public String getPrenom() {
    return prenom;
}

public void setPrenom(String prenom) {
    this.prenom = prenom;
}

//getter et setter de l'email
public String getEmail() {
    return email;
}
```

```

    }

    public void setEmail(String email) {
        this.email = email;
    }

    //la liste des résidences de la personne
    @OneToMany(mappedBy = "owner", cascade = CascadeType.PERSIST)
    public List<Home> getResidence() {
        return Residence;
    }

    public void setResidence(List<Home> residence) {
        Residence = residence;
    }

    public void addMaisons(Home maison){
        Residence.add(maison);
    }

    public void removeMaisons(Home maison){
        Residence.remove(maison);
    }

    /*
    public void addElec(ElectronicDevices elec){
        devices.add(elec);
    }

    public void removeElec(ElectronicDevices elec){
        devices.remove(elec);
    }
    */

    @Override
    public String toString() {
        return "personne [id=" + id + ", name=" + nom + ", prenom=" +
prenom + ", email="+email+"]";
    }

}

```

1.2. Class Home

```

package domain;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;

```

```

import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.xml.bind.annotation.XmlTransient;

import org.codehaus.jackson.annotate.JsonIgnore;

@Entity
public class Home {
    // les attributs de la classe Home : maison
    private long id;
    private int taille;
    private int NbPiece;

    private List<SmartDevices> heaters;
    private List<SmartDevices> electronicDevices;

    //le proprietaire de la maison
    @JsonIgnore
    @XmlTransient
    private Person owner;
    //une maison a une liste chauffages
    //private List<Heaters> chauffages = new ArrayList<Heaters>();

    public Home(){
        super();
        this.heaters=new ArrayList<SmartDevices>();
        this.electronicDevices=new ArrayList<SmartDevices>();
    }

    public Home(int taille, int NbPiece, Person person, List<SmartDevices>
heaters,
        List<SmartDevices> electronicDevices){
        super();
        this.taille=taille;
        this.NbPiece=NbPiece;
        this.owner=person;
        this.heaters=heaters;
        this.electronicDevices=electronicDevices;
    }
    public Home(int taille, int NbPiece){
        super();
        this.taille=taille;
        this.NbPiece=NbPiece;}

    @Id
    @GeneratedValue
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public int getTaille() {

```

```

        return taille;
    }

    public void setTaille(int taille) {
        this.taille = taille;
    }

    public int getNbPiece() {
        return NbPiece;
    }

    public void setNbPiece(int nbPiece) {
        NbPiece = nbPiece;
    }

    @ManyToOne
    public Person getOwner() {
        return owner;
    }

    public void setOwner(Person owner) {
        owner = owner;
    }

    @OneToMany(mappedBy="home")
    public List<SmartDevices> getHeaters() {
        return heaters;
    }

    public void setHeaters(List<SmartDevices> heaters) {
        this.heaters = heaters;
    }

    @OneToMany(mappedBy="home")
    public List<SmartDevices> getElectronicDevices() {
        return electronicDevices;
    }

    public void setElectronicDevices(List<SmartDevices> electronicDevices) {
        this.electronicDevices = electronicDevices;
    }

    public void addDevice(SmartDevices device){
        //Test s'il s'agit d'un chauffage
        if (device instanceof Heaters){
            heaters.add(device);
        }else if (device instanceof ElectronicDevices){
            electronicDevices.add(device);
        }
    }

    /*
     * avant l'heritage
     * public List<Heaters> getChaufagges() {
     *
     *     return chaufagges;
     * }

```

```

        public void setChaufagges(List<Heaters> chaufagges) {
            this.chaufagges = chaufagges;
        }

        public void addChauffrage(Heaters h){
            chaufagges.add(h);
        }

        public void removeChauffrage(Heaters h){
            chaufagges.remove(h);
        }
        */
    }
}

```

2. Q3 : Peuplement de la base de données

```

package jpa;

import java.awt.List;
import java.util.ArrayList;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.persistence.TypedQuery;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;

import domain.ElectronicDevices;
import domain.Heaters;
import domain.Home;
import domain.Person;
import domain.SmartDevices;
public class JpaTest {

    /**
     * @param args
     */
    public JpaTest(EntityManager entityManager){

    }

    public static void main(String[] args) {
        /*Initialisation de l'entityManager*/
        EntityManagerFactory factory = Persistence

```



```

        .createEntityManagerFactory("mysql");
EntityManager manager = factory.createEntityManager();

EntityTransaction tx = manager.getTransaction();
/*Ajout des données pour remplir la base de données*/
tx.begin();

try {
//Ajouter des maison
    Home H1 = new Home();
    H1.setTaille(183);
    H1.setNbPiece(5);

    Home H2 = new Home();
    H2.setTaille(53);
    H2.setNbPiece(2);

    //Création des personnes
    Person P1= new Person();
    P1.setNom("Dupont");

    Person P2= new Person();
    P2.setNom("Legentil");

    //Ajouter des chauffages
    Heaters C1= new Heaters();
    C1.setConso(200);
    //attribuer le chauffage C1 à la maison H1
    C1.setHome(H1);
    manager.persist(C1);

    //ajout d'un autre chauffage : heater
    Heaters C2= new Heaters();
    C2.setConso(180);
    C2.setHome(H2);
    manager.persist(C2);

    //ajout d'un autre chauffage : heater
    Heaters C3= new Heaters();
    C3.setConso(250);
    C3.setHome(H2);
    manager.persist(C3);

    //Compléter les données des maisons
    //Compléter maison H1
    H1.setOwner(P1);
    H1.getChaufagges().add(C1);
    manager.persist(H1);

    //Compléter maison H2
    H2.setOwner(P2);
    H2.getChaufagges().add(C2);
    H2.getChaufagges().add(C3);
    manager.persist(H2);

    //Ajouter des Electronics Devices

```

```

        ElectronicDevices ED1= new ElectronicDevices();
        ED1.setConso(34);
        ED1.setOwner(P1);
        manager.persist(ED1);

        //Ajouter un deuxieme Device
        ElectronicDevices ED2= new ElectronicDevices();
        ED2.setConso(340);
        ED2.setOwner(P2);
        manager.persist(ED2);

        //Compléter les personnes pour pouvoir les ajouter

        //Personne 1
        //Créer les listes
        ArrayList<Home> homes= new ArrayList<Home>();
        ArrayList<ElectronicDevices> ElcDevices= new
ArrayList<ElectronicDevices>();

        //Remplir les listes
        homes.add(H1);
        ElcDevices.add(ED1);

        //Attribuer les listes à la personne
        P1.setDevices(ElcDevices);
        P1.setResidence(homes);
        manager.persist(P1);

        //Personne 2
        //Créer les listes
        ArrayList<Home> homes2= new ArrayList<Home>();
        ArrayList<ElectronicDevices> ElcDevices2= new
ArrayList<ElectronicDevices>();

        //Remplir les listes
        homes.add(H2);
        ElcDevices.add(ED2);

        //Attribuer les listes à la personne
        P2.setDevices(ElcDevices2);
        P2.setResidence(homes2);
        manager.persist(P2);

    } catch (Exception e) {
        e.printStackTrace();
    }
    tx.commit();

    manager.close();
    factory.close();
}

}

```

3. Q4 : Connexion à la base de données MySQL:

-Ajouter les paramètres de connexion de notre base MySQL dans le fichier Persistence.xml.

```
<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://localhost/testjpa"/>
    <property name="hibernate.connection.username" value="root"/>
    <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>
```

-Modifier EntityManagerFactory définit dans la classe EntityManagerHelper pour se connecter à la base de données MySQL. On indique en paramètre le nom de la connexion saisie dans le fichier persistence.xml :

```
public static void main(String[] args) {
    EntityManagerFactory factory = Persistence
        .createEntityManagerFactory("mysql");
    EntityManager manager = factory.createEntityManager();









    EntityTransaction tx = manager.getTransaction();
    tx.begin();

    try {
        Person p1= new Person();
        manager.persist(p1);
        ElectronicDevices Elc= new ElectronicDevices();
        Elc.setConso(200);
        Elc.setOwner(p1);
        manager.persist(Elc);







        Heaters h1= new Heaters();
        h1.setConso(200);
        Heaters h2= new Heaters();
        h2.setConso(180);
        Heaters h3= new Heaters();
        h3.setConso(250);
    }
}
```

3.1. Résultat :

La base de données est ainsi créée et peuplée :

localhost » testipa																							
Structure		SQL		Rechercher		Requête		Exporter		Importer		Opérations		Privilèges		Procédures stoc							
Table		Action										Lignes		Type		Interclassement		Taille		Perte			
<input type="checkbox"/>	electronicdevices		Afficher		Structure		Rechercher		Insérer		Vider		Supprimer	~1		InnoDB		latin1_swedish_ci		32 Kio		-	
<input type="checkbox"/>	heaters		Afficher		Structure		Rechercher		Insérer		Vider		Supprimer	~3		InnoDB		latin1_swedish_ci		32 Kio		-	
<input type="checkbox"/>	home		Afficher		Structure		Rechercher		Insérer		Vider		Supprimer	~1		InnoDB		latin1_swedish_ci		32 Kio		-	
<input type="checkbox"/>	person		Afficher		Structure		Rechercher		Insérer		Vider		Supprimer	~1		InnoDB		latin1_swedish_ci		16 Kio		-	
4 tables		Somme										6		InnoDB		latin1_swedish_ci		112 Kio		0 0			

➔ Remplissage :

+ Options					id	email	nom	prenom		
<input type="checkbox"/>		Modifier		Copier		Effacer	1	NULL	Dupont	NULL
<input type="checkbox"/>		Modifier		Copier		Effacer	2	NULL	Legentil	NULL

4. Héritage :

4.1.SmartDevices :

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.ManyToOne;
import javax.xml.bind.annotation.XmlTransient;

import org.codehaus.jackson.annotate.JsonIgnore;

//Question 5 la classe mère des classes Heaters et ElectronicDevices

@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE) //Définir l'héritage
public abstract class SmartDevices {
    //attributs
    private long id;
    private int conso;

    private Home home;

    //constructeur
    public SmartDevices(int conso, Home home) {
        super();
        this.conso=conso;
        this.home = home;
    }

    public SmartDevices(){
    }
}
```

```

//Création des getters et setters
@Id
@GeneratedValue
public long getId() {
    return id;
}

public void setId(long id) {
    this.id = id;
}

public int getConso() {
    return conso;
}

public void setConso(int conso) {
    this.conso = conso;
}

@ManyToOne
//@JsonIgnore
//@XmlTransient
public Home getHome() {
    return home;
}

public void setHome(Home home) {
    this.home = home;
}

}

```

4.2. Heaters :

```
package domain;
```

```

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

```

```
@Entity
```

```
@DiscriminatorValue("Heaters")
```

```
public class Heaters extends SmartDevices {
```

```
    public Heaters(){
```

```
        super();
```

```
    }
```

```
    //constructeur apres heritages
```

```
    public Heaters(int conso, Home home){
```

```
        super(conso, home);
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Electronic device : [id=" + super.getId() + ", conso=" + super.getConso() +  
    "];
```

```
    }
```

```
    /* Question 1 à 5
```

```
    @Id
```

```
    @GeneratedValue
```

```
    public long getId() {
```

```
        return id;
```

```
    }
```

```
    public void setId(long id) {
```

```
        this.id = id;
```

```
    }
```

```

        public int getConso() {
            return conso;
        }

        public void setConso(int conso) {
            this.conso = conso;
        }

        @ManyToOne(fetch=FetchType.LAZY)
        @JoinColumn(name="HOME_ID")
        public Home getHome() {
            return home;
        }

        public void setHome(Home home) {
            this.home = home;
        }
    }
    */
}

```

4.3. Classe ElectronicDevices :

```

package domain;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

```

@Entity

@DiscriminatorValue("Electronic Device")

```
public class ElectronicDevices extends SmartDevices{

    /*
        * Attributs de la classe Electronics devices
    */
    private long id;
    private int conso;
    private Person owner;

    /*
    public ElectronicDevices(){
        super();
    }

    public ElectronicDevices( int conso, Home home){
        super(conso, home);
    }

    /*Question 1 à 5
    @Id
    @GeneratedValue
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public int getConso() {
        return conso;
    }
}
```



```

        public void setConso(int conso) {

            this.conso = conso;

        }

        @ManyToOne

        public Person getOwner() {

            return owner;

        }

        public void setOwner(Person owner) {

            this.owner = owner;

        }

    }

}

```

4.4. Résultat:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> electronicdevices		~2	InnoDB	latin1_swedish_ci	32 Kio	-
<input type="checkbox"/> heaters		~3	InnoDB	latin1_swedish_ci	32 Kio	-
<input type="checkbox"/> home		~2	InnoDB	latin1_swedish_ci	32 Kio	-
<input type="checkbox"/> person		~2	InnoDB	latin1_swedish_ci	16 Kio	-
<input type="checkbox"/> smartdevices		~0	InnoDB	latin1_swedish_ci	32 Kio	-
5 tables	Somme	9	InnoDB	latin1_swedish_ci	144 Kio	0 o

Ajout de la nouvelle table dans la base de données.

II. Tp3 : Servlet

Ce Tp est la suite du Tp 2

1. Question 1 : Modification du pom.xml

```
<packaging>war</packaging>
```

-Ajouter une dépendance :

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.0.1</version>
    <scope>provided</scope>
</dependency>
```

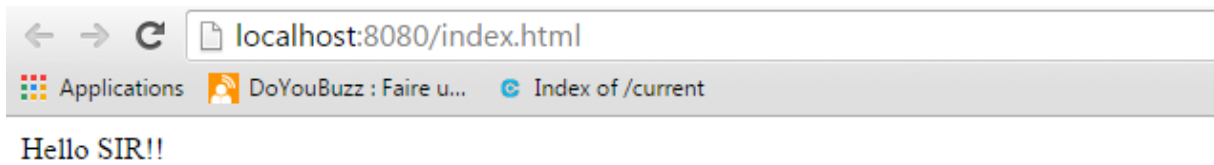
-Ajouter un plugin :

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/</path>
    </configuration>
</plugin>
```

➔ Résultat :

```
TpJpa [Maven Build] C:\Program Files\Java\jdk1.8.0_72\bin\javaw.exe (18 févr. 2016 01:42:50)
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.0.3/plexus-io-2.0.3.jar (57 KB at 875.6 KB
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.15/plexus-utils-3.0.15.jar
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.15/plexus-utils-3.0.15.jar (234 KB at
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/1.0/maven-filtering-1.0.jar
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/1.0/maven-filtering-1.0.jar (42 KB
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.2.1/maven-reporting-api-
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.2.1/maven-reporting-api-2
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-sink-api/1.1/doxia-sink-api-1.1.jar
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-sink-api/1.1/doxia-sink-api-1.1.jar (13 KB at
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1/doxia-logging-api-1.1.jar
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-logging-api/1.1/doxia-logging-api-1.1.jar (12
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/slf4j/jcl-over-slf4j/1.7.5/jcl-over-slf4j-1.7.5.jar
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/slf4j/jcl-over-slf4j/1.7.5/jcl-over-slf4j-1.7.5.jar (17 KB at 304.3 KB
[INFO] Démarrage du war sur http://localhost:8080/
[INFO] Création de la configuration du serveur Tomcat sur C:\Users\kaou\git\TpJpa\target\tomcat
[INFO] create webapp with contextPath:
févr. 18, 2016 1:43:06 AM org.apache.coyote.AbstractProtocol init
INFOS: Initializing ProtocolHandler ["http-bio-8080"]
févr. 18, 2016 1:43:06 AM org.apache.catalina.core.StandardService startInternal
INFOS: Starting service Tomcat
févr. 18, 2016 1:43:06 AM org.apache.catalina.core.StandardEngine startInternal
INFOS: Starting Servlet Engine: Apache Tomcat/7.0.47
févr. 18, 2016 1:43:12 AM org.apache.coyote.AbstractProtocol start
INFOS: Starting ProtocolHandler ["http-bio-8080"]
```

2. Question 2 :



On crée le répertoire src/main/webapp et on ajoute le fichier index.html.

On affiche correctement notre page à l'adresse <http://localhost:8080/index.html>.

3. Question 3 : Création de Servlet :

Créer une classe qui étend HttpServlet.

```
package servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name="mytest",
urlPatterns={"/myurl"})
public class MyServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

        PrintWriter p = new PrintWriter(resp.getOutputStream());
        p.print("Hello world SIR");
        p.flush();

    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        super.doPost(req, resp);
    }
}
```

}

4. Création du fichier myform.html :

Après avoir créer le fichier myform.html et la classe UserInfo, on va sur l'Url suivante :

localhost:8080/myform.html

Applications DoYouBuzz : Faire u... Index of /current

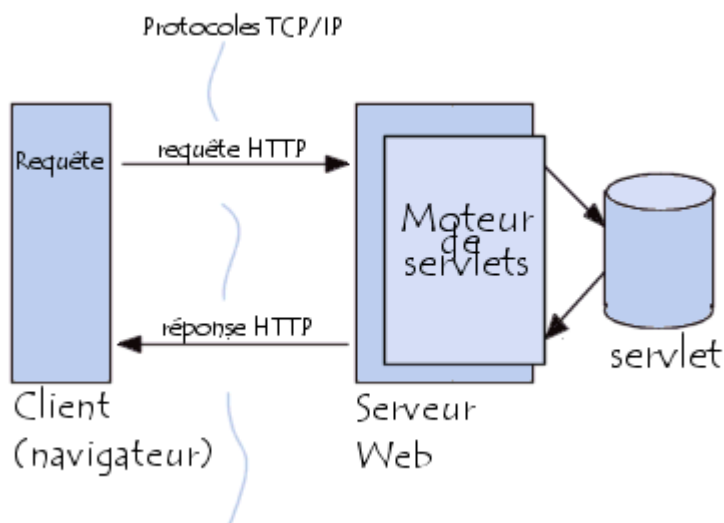
Name :

Firstname :

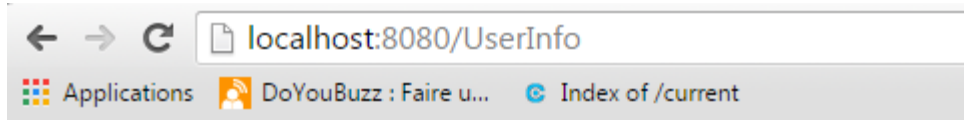
Age :

Introduction aux servlets :

Les servlets sont au serveur web ce que les applets sont au navigateur pour le client. Elles sont des applications java fonctionnant du côté serveur.



Résultat :



Recapitulatif des informations

- Nom: Bennouna
- Prenom: kaoutar
- Age: 21

5. Q5 : Retour sur OPOWER :

5.1. OpowerServlet :

```
package servlet;
```

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import javax.persistence.EntityTransaction;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import domain.Heaters;
```

```
import domain.Home;
```

```
import domain.Person;
```

```

import jpa.EntityManagerHelper;

@WebServlet(name="opower",
urlPatterns={"/Opower"})
public class Opower extends HttpServlet {
    public void doPost(HttpServletRequest request,
                        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        out.println("<HTML>\n<BODY>\n" +
                    "<H1>Affichage des informations</H1>\n" +
                    "<UL>\n" +
                    "<LI>Nom: "
                    + request.getParameter("nom") + "\n" +
                    "<LI>Prenom: "
                    + request.getParameter("prenom") + "\n" +
                    "<LI>Email: "
                    + request.getParameter("email") + "\n" +
                    "<LI>Maison: "
                    + request.getParameter("maison") + "\n" +
                    "</UL>\n" +
                    "</BODY></HTML>");

        //test
        EntityTransaction tx = EntityManagerHelper.getEntityManager().getTransaction();
        tx.begin();
        try {

```

```

        Home home0 = new Home(3,100);

        Heaters heater6=new Heaters(200, home0);

        List<Home> homes = new ArrayList<Home>();

        homes.add(home0);

        Person personne = new Person(request.getParameter("nom"),
request.getParameter("prenom"), request.getParameter("email"), homes);

        EntityManagerHelper.getEntityManager().persist(home0);

        EntityManagerHelper.getEntityManager().persist(heater6);

        EntityManagerHelper.getEntityManager().persist(personne);
    } catch (Exception e) {
        e.printStackTrace();
    }

    tx.commit();

    out.println("Enregistrement effectué</BODY></HTML>");
}

public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

    String query = "select p from Person as p";

    List result2 =
EntityManagerHelper.getEntityManager().createQuery(query).getResultList();

    out.println("<HTML>\n<BODY>\n" +

        "<H1>Recapitulatif des informations</H1>\n" +

        "<UL>\n");

    for (Object enregistrement : result2) {

        out.println("<LI> enregistrement : " + enregistrement+"\n");

    }

    out.println("</UL>\n" +

        "</BODY></HTML>");

}

```

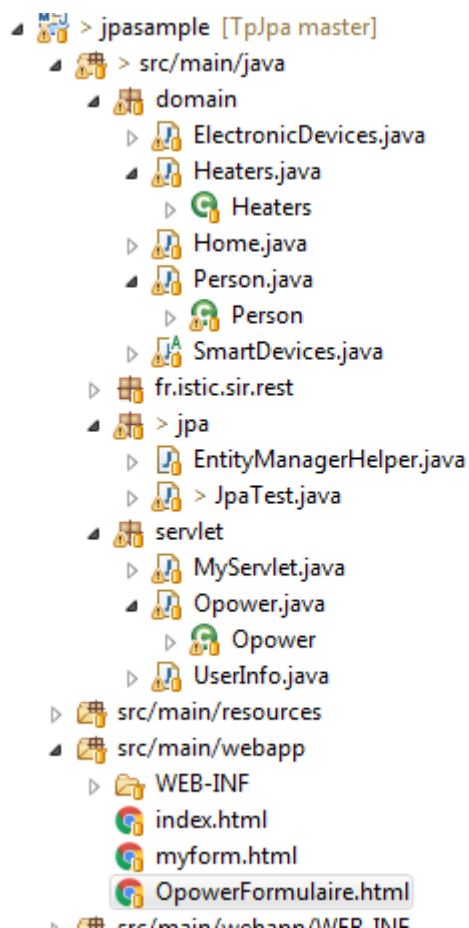
```
}
```

5.2. Opower Formulaire :

opowerFormulaire.html

```
<html>
<body>
<FORM Method="POST" Action="/Opower">
Nom :      <INPUT type="text" size=20 name=nom><BR>
Prenom :   <INPUT type="text" size=20 name=prenom><BR>
Email :    <INPUT type="text" size=50 name=email><BR>
Maison :   <INPUT type="text" size=50 name=maison><BR>
          <INPUT type="submit" value=Envoyer>
</FORM>
</body>
</html>
```

Nous avons utilisé EntityJpaHelper qui est sur le package jpa :



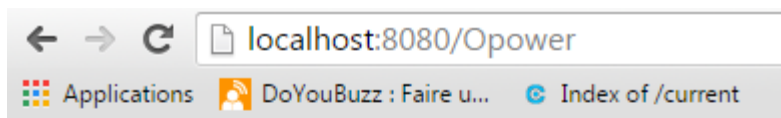
5.3. Résultat :

Une fois les données rentrées sur le formulaire, elles sont enregistrées sur la base de données et affichées sur /opower

Sur mon formulaire OpowerFormulaire.html je saisis les données

Nom :
Prenom :
Email :
Maison :

Une fois les informations envoyées :












Affichage des informations

- Nom: NomTest
- Prenom: PrenomTest
- Email: Test.email
- Maison: Appartement

Enregistrement effectué

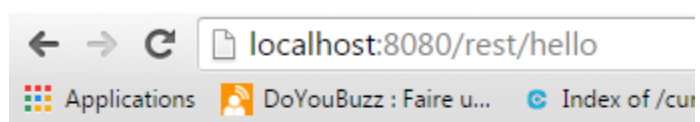
Et la base de données est :

<input type="checkbox"/>		Modifier		Copier		Effacer	4	mina.gmail	Lagentille	Mina
<input type="checkbox"/>		Modifier		Copier		Effacer	5	toto.email	toto	preomtoto
<input type="checkbox"/>		Modifier		Copier		Effacer	6	Test.email	NomTest	PrenomTest

La table home est remplie aussi.

6. Question 6 : Rest :

Relancez tomcat à l'aide de maven. Et rendez-vous ici <http://localhost:8080/rest/hello>



Hello, how are you?

Résultat après les instructions:

```
{ "heaters": [ { "avgCons": "500.0", "id": "0" }, { "avgCons": "600.0", "id": "0" } ], "id": "0", "name": "toto", "nbRoom": "0", "size": "0" }
```