Université Sultan Moulay Slimane - Béni Mellal
École Nationale des Sciences Appliquées - Khouribga

JAVAEE

COMPTE RENDU

ATELIER 3

Réalisé par : *EL FATHI Zakaria*

Encadré par : *Pr. BAKKOURI Ibtissam*

SOMMAIRE

INTRODUCTION ---------- **3**

REALISATION : ---------- **3**

1. DEVELOPPER UN COMPOSANT EJB SESSION SOUS L'ENVIRONNEMENT ECLIPSE QUI PERMET D'ENREGISTRER LES INFORMATIONS D'UN ETUDIANT. ---------- 3
   - *Création d'un projet EJB sous Eclipse (TP3_EJB) :* ---------- *3*
   - *Ajout d'une classe Etudiant pour définir l'entité à stocker en base de données :* ---------- *3*
   - *Création d'une interface EtudiantLocal pour définir les méthodes à implémenter :* ---------- *5*
   - *Implémenter des méthodes pour gerer un étudiant en base de données :* ---------- *6*
   - *Source de données :* ---------- *6*
   - *Persistence.xml :* ---------- *7*
2. ÉCRIRE UNE PAGE HTML QUI CONTIENT UN FORMULAIRE DE SAISIE DES INFORMATIONS D'UN ETUDIANT. ---------- 8
   - *Création un projet Web sous Eclipse (TP3_JEE) :* ---------- *9*
   - *Formulaire :* ---------- *9*
   - *Sevlet :* ---------- *10*
3. LA PAGE JSP QUI RECUPERE LES INFORMATIONS SAISIES DANS LE FORMULAIRE ET QUI UTILISE LE BEAN SESSION. ---------- 11
4. DEPLOIEMENT DE TOUTE L'APPLICATION DANS LE SERVEUR D'APPLICATION JBOSS. ---------- 13
5. TEST DE L'APPLICATION SUR LE NAVIGATEUR WEB ---------- 13

CONCLUSION: ---------- **14**

INTRODUCTION

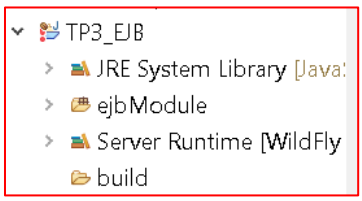
L'objectif de cet atelier était de mettre en pratique les concepts d'Enterprise JavaBeans (EJB) en développant une application de gestion d'étudiants avec Wildfly 15. Le projet a été réalisé sous Eclipse, en créant un projet EJB pour les modèles et les interfaces EJB, et un projet web pour l'interface utilisateur. La base de données a été configurée à l'aide d'un fichier persistence.xml et d'un data source. Dans ce contexte, nous avons développé des EJB de session pour gérer les opérations CRUD sur les étudiants et nous avons créé une interface utilisateur simple pour saisir les informations des étudiants.

REALISATION :

1. Développer un composant EJB session sous l'environnement Eclipse qui permet d'enregistrer les informations d'un étudiant.

Création d'un projet EJB sous Eclipse (TP3_EJB) :

On crée un projet EJB sous Eclipse, nommé "TP3_EJB" :



Ajout d'une classe Etudiant pour définir l'entité à stocker en base de données :

On ajoute un package "models" contenant la classe "Etudiant" annotée avec "@Entity" :

```
package models;
import java.io.Serializable;
import java.util.List;
import javax.persistence.*;

@Entity
public class Etudiant implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String lastName;
```

```
    private String email;
    private String cne;
    private String dNaiss;
    private String tele;

    @Transient
    @OneToMany(fetch = FetchType.LAZY)
    private List<Etudiant> etudiants;

    public Etudiant() {
        super();}

    public Etudiant(String firstName, String lastName, String email, String cne, String
dNaiss, String tele) {
        super();
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.cne = cne;
        this.dNaiss = dNaiss;
        this.tele = tele;}


    public Long getId() {
        return id;}

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

    public String getFirstName() {
        return firstName;}

    public void setFirstName(String firstName) {
        this.firstName = firstName;}

    public String getLastName() {
        return lastName;}

    public void setLastName(String lastName) {
        this.lastName = lastName;}

    public String getEmail() {
        return email;}

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

```
    public String getCne() {
        return cne;}

    public void setCne(String cne) {
        this.cne = cne;}

    public String getdNaiss() {
        return dNaiss;}

    public void setdNaiss(String dNaiss) {
        this.dNaiss = dNaiss;}

    public String getTele() {
        return tele;}

    public void setTele(String tele) {
        this.tele = tele;}

    public List<Etudiant> getEtudiants() {
        return etudiants;   }

    public void setEtudiants(List<Etudiant> etudiants) {
        this.etudiants = etudiants;}
}
```

Création d'une interface EtudiantLocal pour définir les méthodes à implémenter :

Ensuite, on crée un package "dao" contenant une interface "etudiantLocal" annotée avec "@Local" et deux méthodes "submit" et "getOneById" :

```
package dao;

import javax.ejb.Local;
import models.Etudiant;

@Local
public interface etudiantLocal {

    public Etudiant submit(Etudiant etudiant);
    public Etudiant getOneById(Long id);
}
```

Implémenter des méthodes pour gerer un étudiant en base de données :

On crée également une classe "EtudiantImplementation" annotée avec "@Stateless" et "@LocalBean" qui implémente l'interface "etudiantLocal". Cette classe utilise l'EntityManager pour persister et récupérer les objets "Etudiant" en base de données :

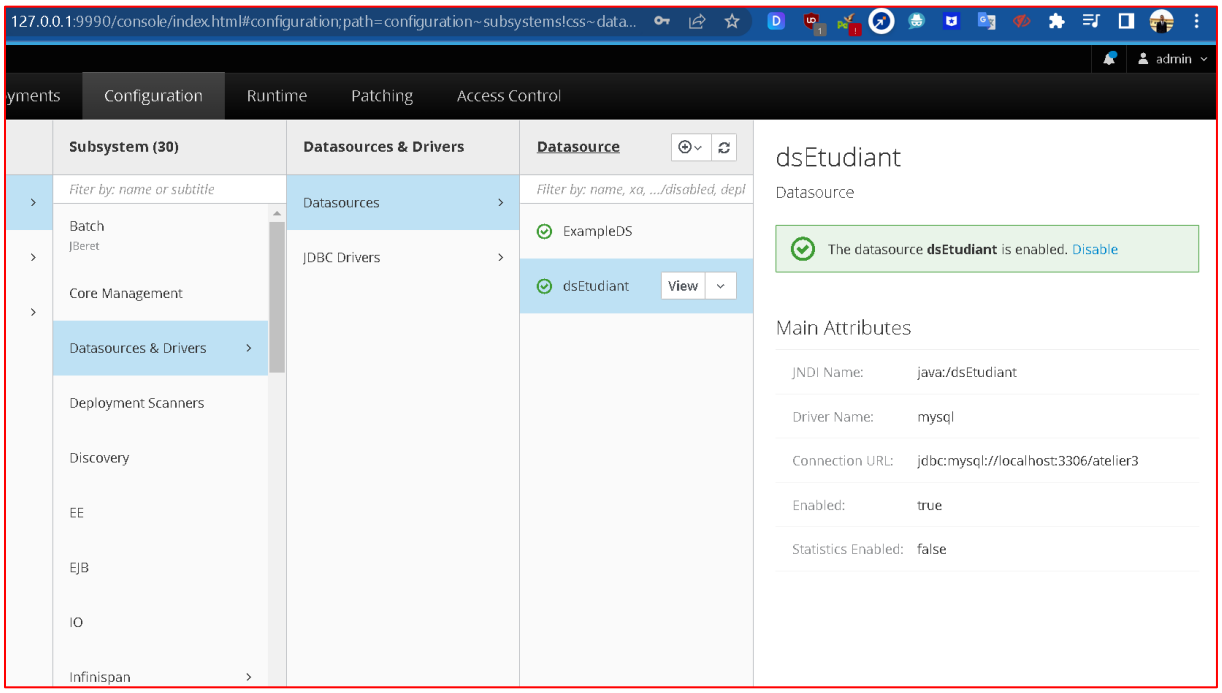
```
package dao;

import javax.ejb.LocalBean;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import models.Etudiant;


@Stateless
@LocalBean
public class EtudiantImplementation implements etudiantLocal {
    public EtudiantImplementation() {
        super();}
    @PersistenceContext(unitName = "TP3_EJB")
    EntityManager EM;

    @Override
    public Etudiant submit(Etudiant etudiant) {
        // TODO Auto-generated method stub
        EM.persist(etudiant);
        return etudiant;}

    @Override
    public Etudiant getOneById(Long id) {
        // TODO Auto-generated method stub
        Etudiant e = EM.find(Etudiant.class, id);
        return e ;}
}
```

Source de données :

Dans cette étape, j'ai configuré une source de données JNDI nommée "dsEtudiant" pour permettre à notre application d'accéder à la base de données. J'ai créé cette source de données en utilisant la technologie JDBC. Pour ce faire, j'ai d'abord ouvert la console d'administration WildFly, puis j'ai navigué vers la section "Configuration", où j'ai choisi "Subsystems" -> "Datasources" -> "Non-XA". J'ai ensuite cliqué sur "Add" pour créer une nouvelle source de données et j'ai rempli les informations nécessaires, telles que le nom de la source de données, le nom de la base de données, le nom d'utilisateur et le mot de passe.

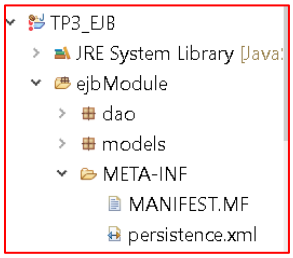
Après avoir créé la source de données, j'ai testé sa connexion pour m'assurer que l'application pouvait accéder à la base de données. Pour ce faire, j'ai sélectionné la source de données "dsEtudiant" et j'ai cliqué sur le bouton "Test Connection". Si la connexion était établie avec succès, j'ai reçu un message indiquant que la connexion était réussie. Sinon, j'ai vérifié les informations de connexion pour corriger les erreurs éventuelles.

La configuration d'une source de données JNDI est essentielle pour permettre à notre application d'interagir avec une base de données. En créant une source de données, nous pouvons fournir les informations de connexion à notre application de manière centralisée, ce qui facilite la maintenance et la gestion de la base de données.



Persistence.xml :

On crée également un fichier "persistence.xml" dans le répertoire "META-INF" :



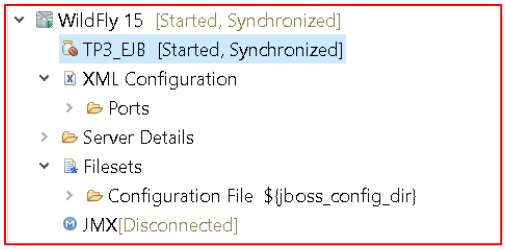
Persistence.xml :

On précise également dans ce fichier le nom de notre data-source

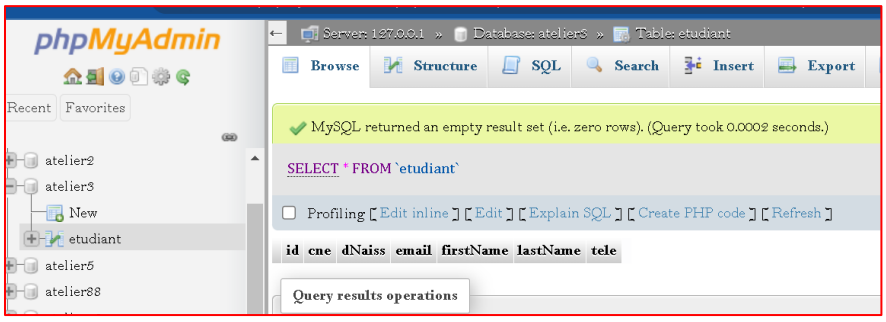
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
```

```
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">

    <!-- définition d'une unité de persistence -->
    <persistence-unit
        name="TP3_EJB"
        transaction-type="JTA">
        <jta-data-source>java:/dsEtudiant</jta-data-source>
        <!-- spécification de la classe entité -->
        <class>models.Etudiant</class>
        <!-- configuration des propriétés -->
        <properties>
            <property
                name="javax.persistence.schema-generation.database.action"
                value="drop-and-create"  />
        </properties>
    </persistence-unit>
</persistence>
```

Apres ces étapes notre application EJB devrait être capable de créer la table etudiant :



La table s'est également créée dans la base de données :



2. Écrire une page HTML qui contient un formulaire de saisie des informations d'un étudiant.

La deuxième étape consiste à créer une page qui permettra de saisir les informations de l'étudiant. Pour cela, j'ai créé un projet Web sous Eclipse en utilisant le nom "TP3_JEE"

Création un projet Web sous Eclipse (TP3_JEE) :

Ensuite, j'ai ajouté une page appelée "part1-form.jsp" qui contiendra les champs nécessaires pour saisir les informations de l'étudiant, comme son nom, son prénom, son âge et son adresse.

Une fois que la page HTML est créée, j'ai ajouté une action nommée "part1Servlet" pour gérer la soumission du formulaire. Cette action sera exécutée lorsque l'utilisateur cliquera sur le bouton de soumission du formulaire. Elle permettra de récupérer les données saisies par l'utilisateur et de les envoyer à la base de données en utilisant la source de données JNDI nommée "dsEtudiant".

Formulaire :

```
<html lang="en">
<head>
<title>formulaire Etudiant</title>

<meta charset="utf-8">
<meta name="viewport"
    content="width=device-width, initial-scale=1, shrink-to-fit=no">
<link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="login-box">
        <h2>Inscription:</h2>
        <form method="post" action="part1Servlet">
            <div class="input-group">
                <div class="user-box">
                    <input type="text" class="form-control" id="firstName"
                        name="firstName" placeholder="votre prenom">
                </div>
                <div class="user-box">
                    <input type="text" class="form-control" id="lastName"
                        name="lastName" placeholder="votre prenom">
                </div>
                <div class="user-box">
                    <input type="email" class="form-control" id="email" name="email"
                        placeholder="votre email">
                </div>
                <div class="user-box">
                    <input type="text" class="form-control" id="cne" name="cne"
                        placeholder="votre CNE">
                </div>
```

```
                <label class="user-box">Date de naissance:</label>
                <div class="user-box">
                    <input type="date" id="dNaiss" name="dNaiss">
                </div>
                <div class="user-box">
                    <input type="Tel" class="form-control" id="tele" name="tele"
                        placeholder="votre numero de telephone">
                </div>
                <input type="submit" value="Envoyer!">
            </div>
        </form>
    </div>
</body>
</html>
```

Les donnees sont envoyees a une servlet, la classe "part1Servlet" est une servlet Java qui gère la soumission du formulaire HTML créé à l'étape précédente. Elle est annotée avec "@WebServlet" pour permettre au serveur d'application de la repérer et de la gérer lorsqu'une requête est envoyée vers l'URL correspondante. La méthode "doPost" récupère les données du formulaire à l'aide de la classe HttpServletRequest, crée une instance de la classe "Etudiant" en utilisant ces données, et envoie cette instance au bean EJB "etudiantLocal" en appelant sa méthode "submit". Ensuite, elle stocke l'instance de l'étudiant dans l'objet de requête et redirige la requête vers la page "part1-response.jsp". La méthode "doGet" est également définie pour renvoyer la page "part1-form.jsp" lorsqu'une requête GET est envoyée à l'URL correspondante.

Sevlet :

```
import models.Etudiant;
import dao.etudiantLocal;
import javax.servlet.annotation.WebServlet;

@WebServlet("/part1Servlet")
public class part1Servlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    @EJB
    etudiantLocal EL;

    public void init() throws ServletException {
        // TODO Auto-generated method stub
        super.init();}

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Retrieve the first name and last name and email from myForm.jsp
        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");
        String email = request.getParameter("email");
```

```
        String cne = request.getParameter("cne");
        String tele = request.getParameter("tele");
        String dNaiss = request.getParameter("dNaiss");

        // Create a new instance of the Etudiant and populate its properties
        Etudiant etudiant = new Etudiant();

        etudiant.setFirstName(firstName);
        etudiant.setLastName(lastName);
        etudiant.setEmail(email);
        etudiant.setTele(tele);
        etudiant.setdNaiss(dNaiss);
        etudiant.setCne(cne);

        EL.submit(etudiant);

        // Store the Etudiant in the request object
        request.setAttribute("Etudiant", etudiant);

        // Forward the request to the part1-response.jsp page
        request.getRequestDispatcher("part1-response.jsp").forward(request, response);
    }

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

        request.getRequestDispatcher("part1-form.jsp").forward(request, response);
    }
}
```

3. La page JSP qui récupère les informations saisies dans le formulaire et qui utilise le bean session.

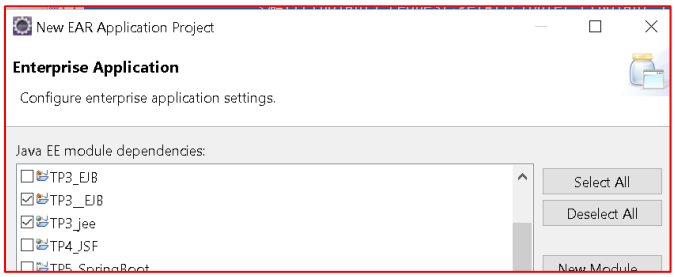
```
<!DOCTYPE html>
<%@ page import="models.*"%>

<html>
<head>
<link rel="stylesheet" href="css/styleAffichage.css">
<title>my Response</title>
</head>
<body>
    <div class="login-box">
```

```
        <div class="input-group">
            <div class="user-box">
                <h1>
                    <strong>PRENOM:</strong>
                    <%=((Etudiant) request.getAttribute("Etudiant")).getFirstName()%>
                </h1>
            </div>
            <div class="user-box">
                <h1>
                    <strong>NOM:</strong>
                    <%=((Etudiant) request.getAttribute("Etudiant")).getLastName()%></h1>
            </div>
            <div class="user-box">
                <h1>
                    <strong>EMAIL:</strong>
                    <%=((Etudiant) request.getAttribute("Etudiant")).getEmail()%></h1>
            </div>
            <div class="user-box">
                <h1>
                    <strong>CNE:</strong>
                    <%=((Etudiant) request.getAttribute("Etudiant")).getCne()%></h1>
            </div>
            <div class="user-box">
                <h1>
                    <strong>DATE DE NAISSANCE:</strong>
                    <%=((Etudiant) request.getAttribute("Etudiant")).getdNaiss()%>
                </h1>
            </div>
            <div class="user-box">
                <h1>
                    <strong>TELEPHONE:</strong>
                    <%=((Etudiant) request.getAttribute("Etudiant")).getTele()%></h1>
            </div>
            <div>
                <a href="part1-form.jsp"> <input type="submit" value="revenir">
                </a>
            </div>
        </div>
    </div>
</body>
</html>
```

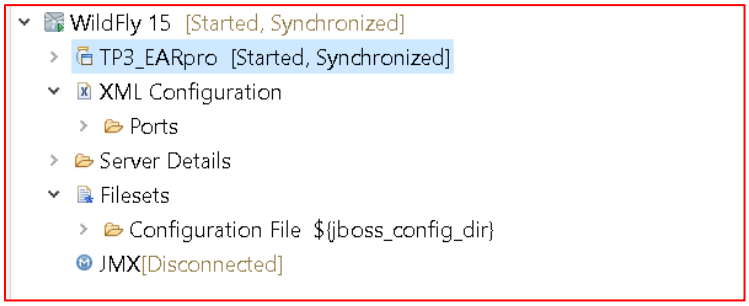
4. Déploiement de toute l'application dans le serveur d'application JBoss.

Pour déployer toute l'application dans le serveur d'application JBoss, nous devons d'abord configurer le serveur pour qu'il puisse déployer les projets TP3_EJB et TP3_JEE. Une fois que cela est fait, nous pouvons procéder au déploiement des projets sur le serveur.

Pour déployer les projets, nous devons créer un fichier EAR (Enterprise Archive) qui contiendra les projets TP3_EJB et TP3_JEE. Ensuite, nous pouvons déployer ce fichier EAR sur le serveur JBoss.

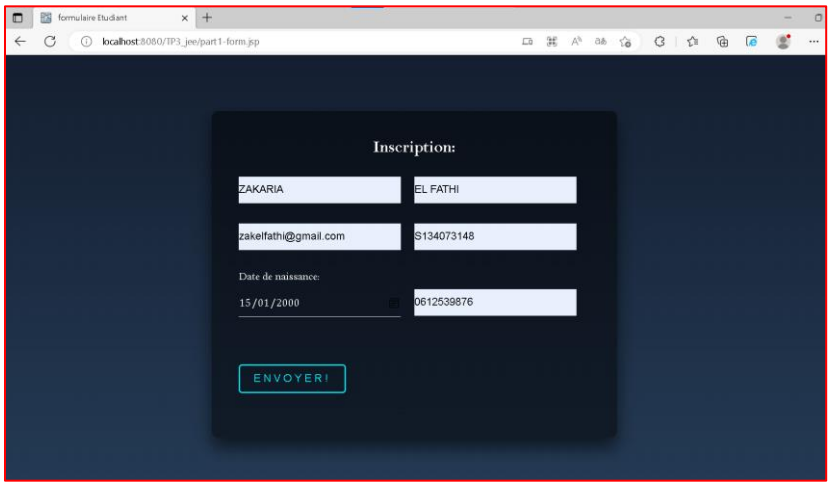


Maintenant on fait ajouter ce projet EAR dans le serveur comme suit :

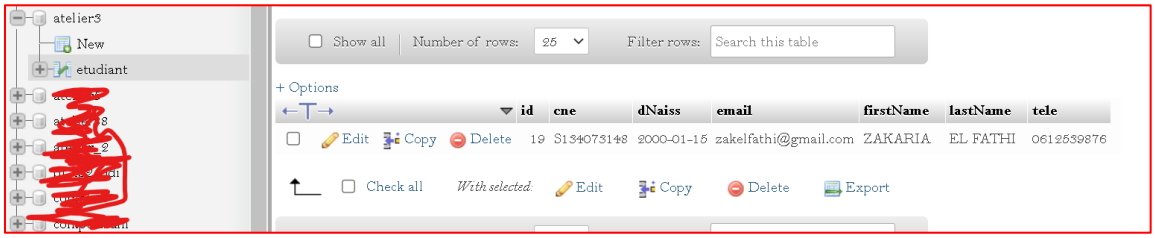


5. Test de l'application sur le navigateur Web

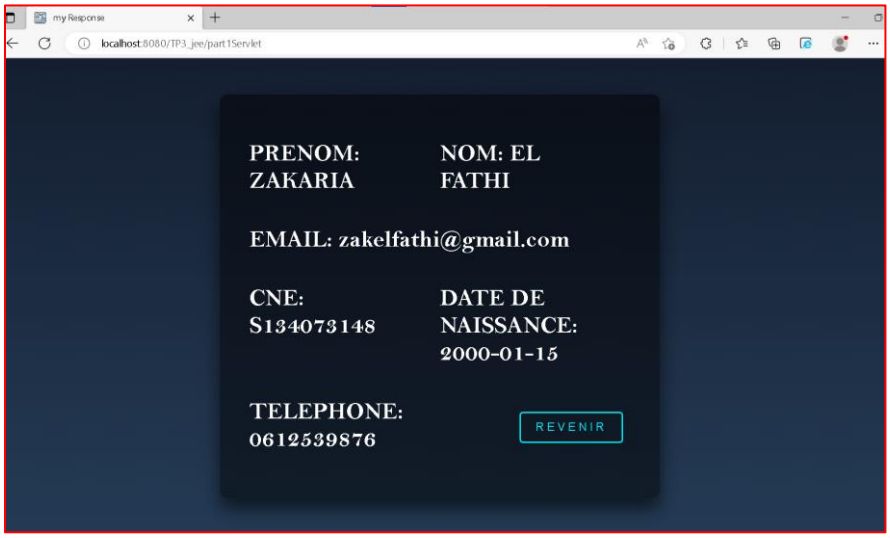
Pour tester cette application sur le navigateur, il faut tout d'abord s'assurer que le serveur JBoss est en cours d'exécution. Ensuite, il suffit d'ouvrir un navigateur Web et d'entrer l'URL suivante : "http://localhost:8080/TP3_jee/part1-form.jsp ". Cela affichera la page de formulaire où l'utilisateur peut saisir les informations d'un étudiant.



Après avoir soumis le formulaire, l'application enregistre les informations dans la base de données



et affiche une page de confirmation (part1-response.jsp) avec les informations de l'étudiant enregistré.



CONCLUSION:

En conclusion, nous avons présenté une proposition pour la création d'une application de gestion d'inscription des étudiants. En utilisant le diagramme de classe, nous avons détaillé les différentes étapes nécessaires pour la mise en place du projet, y compris la création des entités Etudiant et Reinscription avec les annotations appropriées, la configuration de Hibernate, et la création d'une base de données sous MySql. Nous avons également présenté une série de fonctionnalités telles que l'affichage de la liste des étudiants, l'affichage des réinscriptions pour un étudiant spécifique, la suppression et la modification d'étudiants, ainsi que l'affichage de la liste des étudiants en fonction de leur niveau. Cette application de gestion de l'inscription des étudiants peut être étendue en ajoutant de nouvelles fonctionnalités et en adaptant le modèle de données pour répondre à d'autres besoins spécifiques.