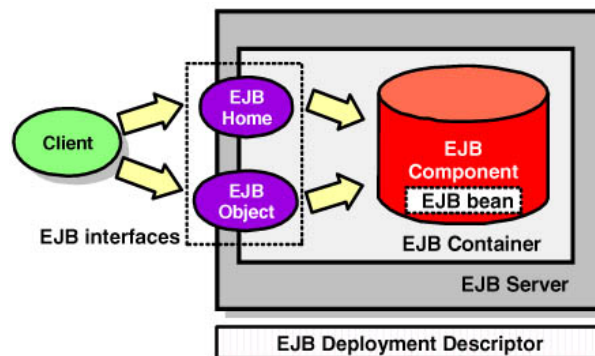


Université Abdelmalek Essaadi

Faculté des Sciences et Techniques de Tanger

Département Génie Informatique

Mise en place d'une application distribuée JEE



Encadré par :
Prof. Lotfi ELAACHAK

Elaboré par :
ELHANSALI Mouaad

Table des matières

1	Objectif du Projet	2
2	Création de la base de données	2
3	Développement de l'application EJB	2
3.1	Classe persistante	2
3.2	SessionBean et Interface Remote	3
4	Consommation des EJB dans une application web	4
4.1	Servlet pour ajouter un étudiant	4
5	Conclusion	6

1 Objectif du Projet

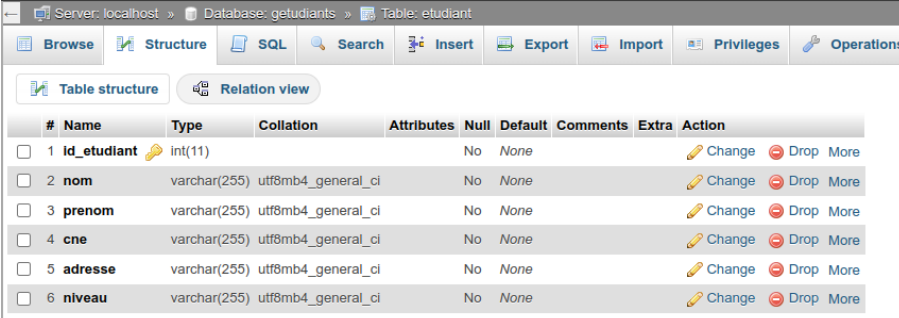
L'objectif principal de cet atelier est de pratiquer la mise en place d'une application distribuée en utilisant une variété de technologies centrées sur **EJB3**.

2 Création de la base de données

Création de la base de données **Getudiant** avec MySQL. Voici le script SQL utilisé :

Listing 1 – Script SQL pour la base de données Getudiant

```
CREATE DATABASE Getudiants;
USE Getudiants;
CREATE TABLE etudiant (
    id_etudiant INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50),
    prenom VARCHAR(50),
    cne VARCHAR(20),
    adresse TEXT,
    niveau VARCHAR(20)
);
```



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_etudiant	int(11)			No	None			Change Drop More
2	nom	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
3	prenom	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
4	cne	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	adresse	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
6	niveau	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More

FIGURE 1 – Création de la base de données dans MySQL

3 Développement de l'application EJB

3.1 Classe persistante

La classe Etudiant représente la table etudiant dans la base de données. Voici son implémentation :

Listing 2 – Classe persistante Etudiant

```
import jakarta.persistence.*;
```

```

@Entity
@Table(name = "etudiant")
public class Etudiant {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idEtudiant;

    private String nom;
    private String prenom;
    private String cne;
    private String adresse;
    private String niveau;

    // Getters et setters
}

```

3.2 SessionBean et Interface Remote

Le **SessionBean** EtudiantServiceImpl implémente les opérations CRUD pour la gestion des étudiants.

Listing 3 – Interface Remote EtudiantService

```

import jakarta.ejb.Remote;
import java.util.List;

@Remote
public interface EtudiantService {
    void ajouterEtudiant(Etudiant etudiant);
    Etudiant trouverEtudiant(int id);
    List<Etudiant> listerEtudiants();
    void modifierEtudiant(Etudiant etudiant);
    void supprimerEtudiant(int id);
}

```

Listing 4 – Implémentation SessionBean EtudiantServiceImpl

```

import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import java.util.List;

```

```

@Stateless
public class EtudiantServiceImpl implements EtudiantService {

    @PersistenceContext(unitName = "Etudiant")
    private EntityManager entityManager;

    @Override
    public void ajouterEtudiant(Etudiant etudiant) {
        entityManager.persist(etudiant);
    }

    @Override
    public Etudiant trouverEtudiant(int id) {
        return entityManager.find(Etudiant.class, id);
    }

    @Override
    public List<Etudiant> listerEtudiants() {
        return entityManager.createQuery("SELECT e FROM Etudiant e"
            , Etudiant.class).getResultList();
    }

    @Override
    public void modifierEtudiant(Etudiant etudiant) {
        entityManager.merge(etudiant);
    }

    @Override
    public void supprimerEtudiant(int id) {
        Etudiant etudiant = trouverEtudiant(id);
        if (etudiant != null) {
            entityManager.remove(etudiant);
        }
    }
}

```

4 Consommation des EJB dans une application web

4.1 Servlet pour ajouter un étudiant

Listing 5 – Servlet AjouterEtudiantServlet

```
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.naming.InitialContext;
import jakarta.naming.NamingException;
import java.io.IOException;

@WebServlet("/ajouter")
public class AjouterEtudiantServlet extends HttpServlet {

    private EtudiantService etudiantService;

    @Override
    public void init() throws ServletException {
        try {
            etudiantService = (EtudiantService) new InitialContext
                ().lookup("java:global/atelier3/EtudiantServiceImpl!
                    ma.fstt.ejb.EtudiantService");
        } catch (NamingException e) {
            throw new ServletException(e);
        }
    }

    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException,
        IOException {
        Etudiant etudiant = new Etudiant();
        etudiant.setNom(req.getParameter("nom"));
        etudiant.setPrenom(req.getParameter("prenom"));
        etudiant.setCne(req.getParameter("cne"));
        etudiant.setAdresse(req.getParameter("adresse"));
        etudiant.setNiveau(req.getParameter("niveau"));

        etudiantService.ajouterEtudiant(etudiant);

        resp.getWriter().println("Etudiant ajout avec succ s !")
        ;
    }
}
```

```
}  
}
```

5 Conclusion

Ce projet nous a permis de suivre une approche méthodique pour construire une application distribuée complète basée sur **JEE**. En respectant les étapes définies, nous avons appris à :

- Modéliser les données avec un diagramme de classe clair.
- Implémenter une couche persistance fiable avec JPA.
- Créer une interface utilisateur dynamique et intuitive.
- Relier les composants grâce à des Managed Beans et des Servlets.