
ATELIER 3

Mise en place d'une application distribuée JEE



Encadré Par : Mr.Lotfi El Aachak

Réaliser Par : Tlemzi Fatima

Table des matières

Objectif :	3
Outils :	3
Les étapes	4
Etape 1 :	4
Etape 2 :	5
Etape 3 :	6
Etape 4 :	8
Etape 5 :	9
Etape 6 :	10
Les Vues de notre Application web :	11

Objectif :

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

Outils :

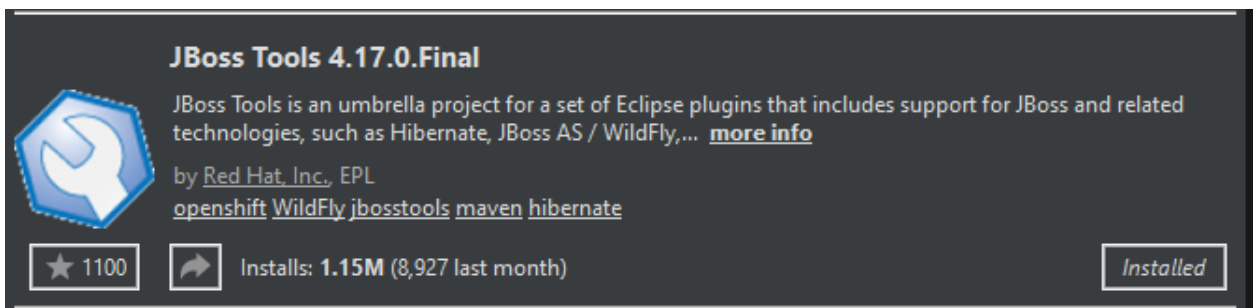
Eclipse, Maven, Tomcate, MySQL, JPA, EJB, WilddFly

Les étapes

Etape 1 :

Installez le plugin Jboss AS en tapant sur le marketplace d'eclipse le mot clé «JBoss Tools

(Oxygen) » Tools, puis installez WildFly via le Wizard.



```
<?xml version="1.0" encoding="UTF-8"?>
<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">
<persistence-unit name="Etudiant"
transaction-type="JTA">
<jta-data-source>java:/ mysql</jta-data-source>
<properties>
<property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
<property name="hibernate.hbm2ddl.auto" value="create" />
</properties>
</persistence-unit>
```

Etape 2 :

Créez une base de données sous le nom Getudiants avec une table étudiant sous le SGBD

MYSQL, la table doit contenir les champs suivant (id_etudiant, nom, prénom, cne, adresse, niveau).

getudiants etudiant
id_etudiant : int(11)
adresse : varchar(100)
cne : varchar(100)
niveau : varchar(100)
nom : varchar(100)
prenom : varchar(100)
id_prenom : int(11)
prénom : varchar(255)

Etape 3 :

Créez un projet EJB avec un EJB module supérieur à 3 puis créez le fichier persistence.xml dans le répertoire META-INF.

The screenshot shows the 'New EJB Project' wizard in Eclipse IDE. The window title is 'New EJB Project'. The main heading is 'EJB Project' with a subtitle 'Create an EJB Project and add it to a new or existing Enterprise Application.' and a folder icon. The wizard is divided into several sections:

- Project name:** A text field containing 'ejpEtudiant'.
- Project location:** A section with a checked checkbox 'Use default location'. Below it, a text field shows the location 'C:\Users\hajar\eclipse-workspace\ejpEtudiant' and a 'Browse...' button.
- Target runtime:** A dropdown menu showing 'WildFly 21.0 Runtime' and a 'New Runtime...' button.
- EJB module version:** A dropdown menu showing '3.2'.
- Configuration:** A dropdown menu showing 'Default Configuration for WildFly 21.0 Runtime' and a 'Modify...' button. Below the dropdown, a note states: 'A good starting point for working with WildFly 21.0 Runtime runtime. Additional facets can later be installed to add new functionality to the project.'
- EAR membership:** A section with an unchecked checkbox 'Add project to an EAR'. Below it, a text field shows 'EAR' and a 'New Project...' button.
- Working sets:** A section with an unchecked checkbox 'Add project to working sets' and a 'New...' button. Below it, a text field is empty and a 'Select...' button is present.

At the bottom of the wizard, there is a navigation bar with a help icon, '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel' buttons.

Création du fichier persistence.xml dans le répertoire META-INF:

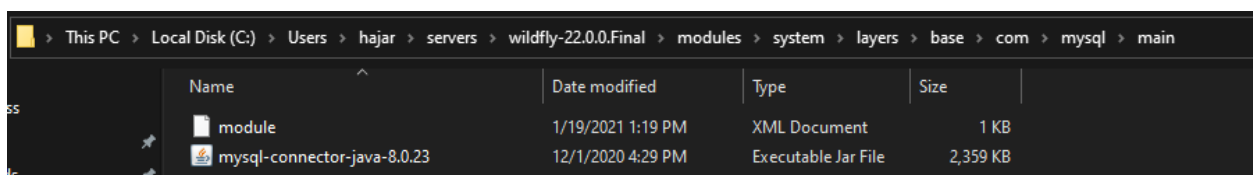
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"

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">
<persistence-unit name="Etudiant"
transaction-type="JTA">
<jta-data-source>java:/MySqlDS</jta-data-source>
<class>ma.fstt.entity.Etudiant</class>
<properties>
<property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5Dialect" />
<property name="hibernate.hbm2ddl.auto" value="update" />
</properties>
</persistence-unit>
</persistence>
```

Configuration du data source au niveau des fichiers de configuration WildFly :

-L'ajout d'un "\wildfly-22.0.0.Final\modules\system\layers\base\com\mysql\main" qui contient un fichier module.xml et et le fichier jar mysql connector :



Name	Date modified	Type	Size
module	1/19/2021 1:19 PM	XML Document	1 KB
mysql-connector-java-8.0.23	12/1/2020 4:29 PM	Executable Jar File	2,359 KB

L'ajout du driver et datasource au niveau du fichier "\wildfly-22.0.0.Final\standalone\configurationstandalone.ini" :

```

<datasource jndi-name="java:/MySqlDS" pool-name="MySqlDS">
  <connection-url>jdbc:mysql://localhost:3306/getudiants</connection-url>
  <driver-class>com.mysql.cj.jdbc.Driver</driver-class>
  <driver>mysql</driver>
  <security>
    <user-name>root</user-name>
  </security>
  <validation>
    <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQL"
    <background-validation>true</background-validation>
    <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLException"
  </validation>
</datasource>
</drivers>

```

Etape 4 :

Création de la classe persistante Etudiant en utilisant les annotations JPA :

```

1 package ma.fstt.entity;
2
3 import java.io.Serializable;
4
5
6 @Entity
7 @Table(name = "etudiant")
8 public class Etudiant implements Serializable {
9
10     private static final long serialVersionUID = 1L;
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     @Column(name = "id_etudiant")
15     private Integer id_etudiant;
16     @Column(name = "nom", length = 100)
17     private String nom;
18     @Column(name = "prenom", length = 100)
19     private String prenom;
20     @Column(name = "cne", length = 100)
21     private String cne;
22     @Column(name = "adresse", length = 100)
23     private String adresse;
24     @Column(name = "niveau", length = 100)
25     private String niveau;
26     public Integer getId_etudiant() {
27         return id_etudiant;
28     }
29     public void setId_etudiant(Integer id_etudiant) {
30         this.id_etudiant = id_etudiant;
31     }
32     public String getNom() {
33         return nom;
34     }
35     public void setNom(String nom) {
36         this.nom = nom;
37     }
38     public String getPrenom() {

```


Etape 5 :

Créez une EJB de type SessionBean avec une interface Remote cette SessionBean doit contenir tous les méthodes CRUD de l'étudiant en Utilisant :

```
1 package ma.fstt.beans;
2
3 import java.util.List;
4
5 @Stateless
6 public class Gestionetud implements TraitementLocale,TraitementRemote{
7     @PersistenceContext
8     private EntityManager em;
9
10    @Override
11    public void edit(Etudiant et) {
12        // TODO Auto-generated method stub
13        em.merge(et);
14    }
15
16    @Override
17    public void delete(Etudiant et) {
18        // TODO Auto-generated method stub
19        Etudiant e=em.merge(et);
20        em.remove(e);
21    }
22
23    public boolean addEtudiant(Etudiant et) {
24        em.persist(et);
25        return true;
26    }
27
28    public List listAll()
29    {
30        Query queryObj = em.createQuery("SELECT e FROM Etudiant e");
31        List<Etudiant> eList = queryObj.getResultList();
32        if (eList != null && eList.size() > 0) {
33            return eList;
34        } else {
35            return null;
36        }
37    }
38 }
```

L'interface Remote :

```
1 package ma.fstt.beans;
2 import java.util.List;
7
8 @Remote
9 public interface TraitementRemote {
10 public void edit(Etudiant et);
11 public void delete(Etudiant et);
12 public boolean addEtudiant(Etudiant et);
13 public List listAll();
14
15
```

L'interface Local :

```
1 package ma.fstt.beans;
2
3 import java.util.List;
8
9 @Local
10 public interface TraitementLocale {
11     public void edit(Etudiant et);
12     public void delete(Etudiant et);
13     public boolean addEtudiant(Etudiant et);
14     public List listAll();
15
16
```

Etape 6 :

Créez un projet web dynamique avec un web module 4 et JSF 2.x qui va consommer les méthodes du projet EJB « CRUD Etudiant ».

Tout d'abord on exporte le projet EJB sous forme d'un fichier jar et on l'ajoute à notre dossier lib :



Création de la classe Bean pour Etudiant : qui contient les getters et les setters avec les fonctions du CRUD

```

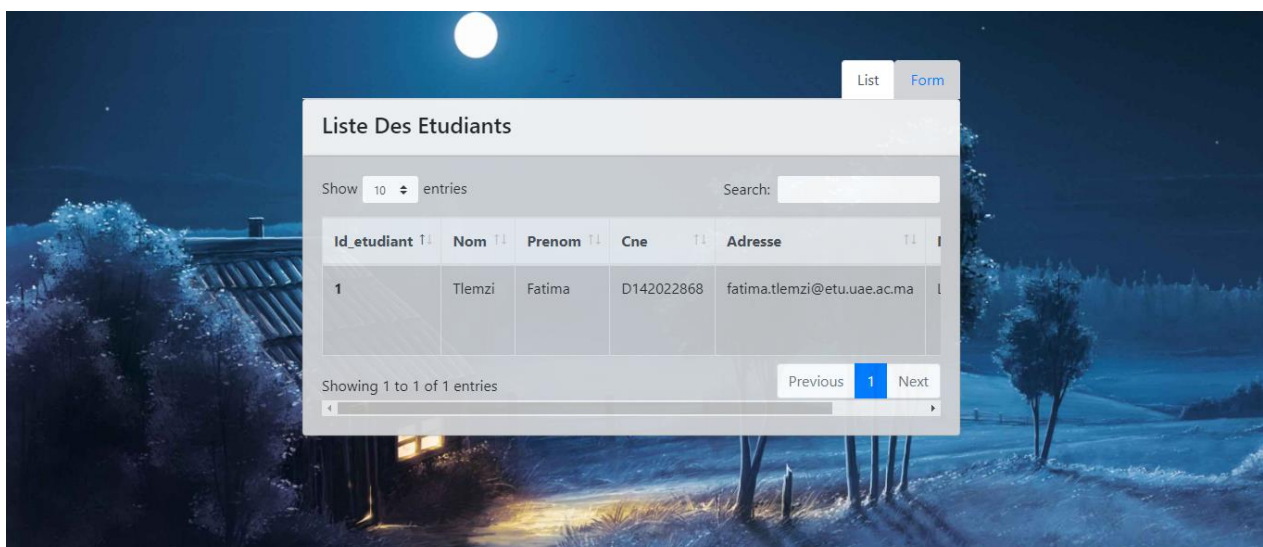
import java.util.ArrayList;
@ManagedBean(name = "etudiantbean")
public class EtudiantBean {
    @EJB
    (mappedName="java:global/Ejbproject/Gestionetud!ma.fstt.beans.TraitementRemote")
    TraitementRemote remote;
    private int id_prenom;
    private String nom;
    private String prénom;
    private String cne;
    private String adresse;
    private String niveau;
    public static int id_param=0;
    @PostConstruct
    public void init() {
        if(id_param!=0) {
            Etudiant et=remote.getbyid(id_param);
            this.id_prenom=id_param;
            this.prénom=et.getPrénom();
            this.nom=et.getNom();
            this.adresse=et.getAdresse();
            this.cne=et.getCne();
            this.niveau=et.getNiveau();
        }
    }
}

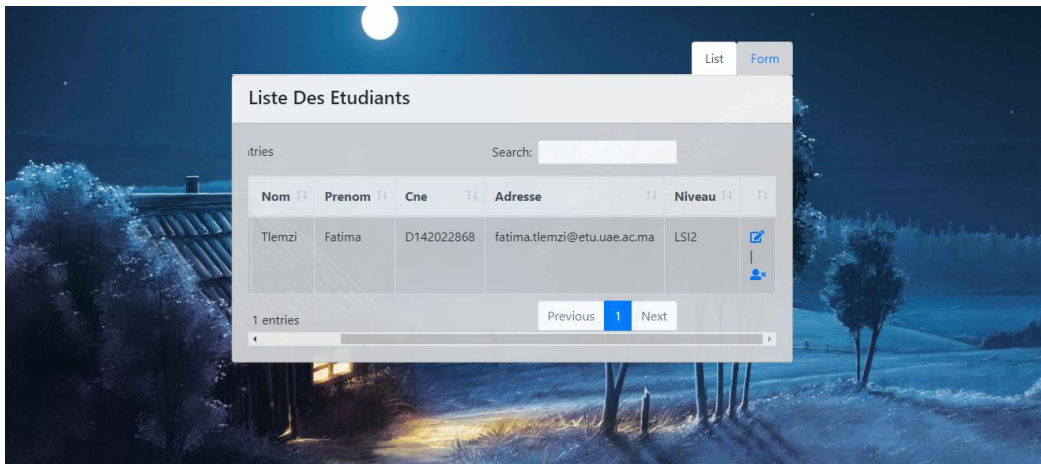
```

Les Vues de notre Application web :

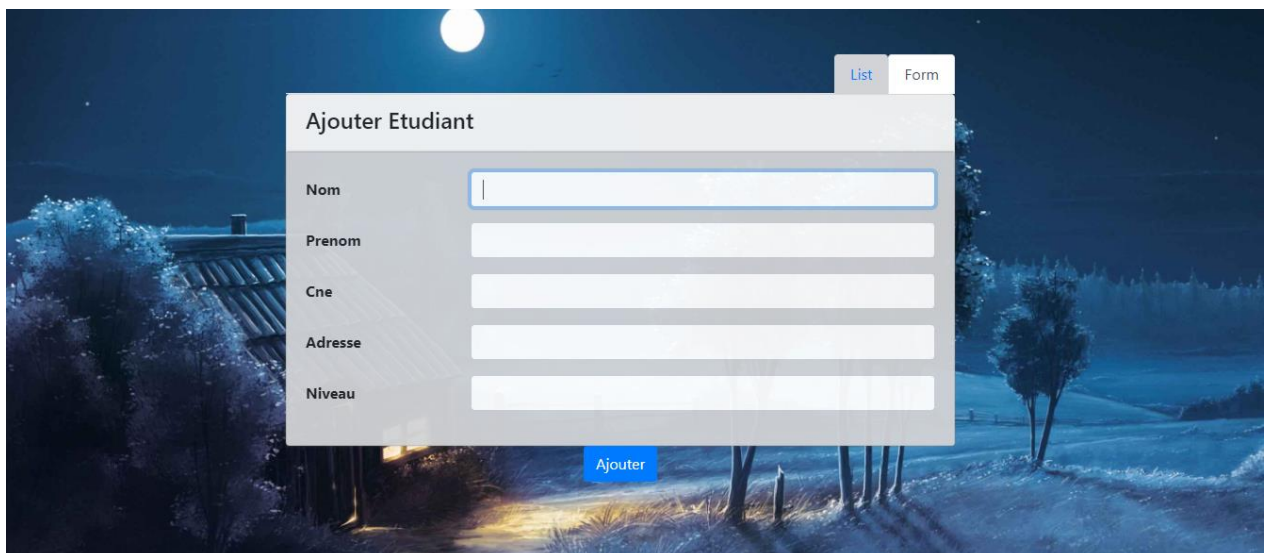
Création des pages *.xhtml en utilisant JSF:

page list.xhtml :





page ajouter.xhtml :



Lien guithub :

<https://github.com/TLEMZI-fatima/Application-distribue-JEE.git>