TP N° 4 Création d'un projet Web Avec EJB JSF et Glassfish

I. Les objectifs :

- Compréhension des architectures EJB ainsi que le modèle MVC.
- Création d'une application Web.
- Création d'une base de données avec WAMP et la réalisation de la connexion avec JPA.
- Création d'interfaces Web assurant l'affichage des données relatifs aux employeurs.

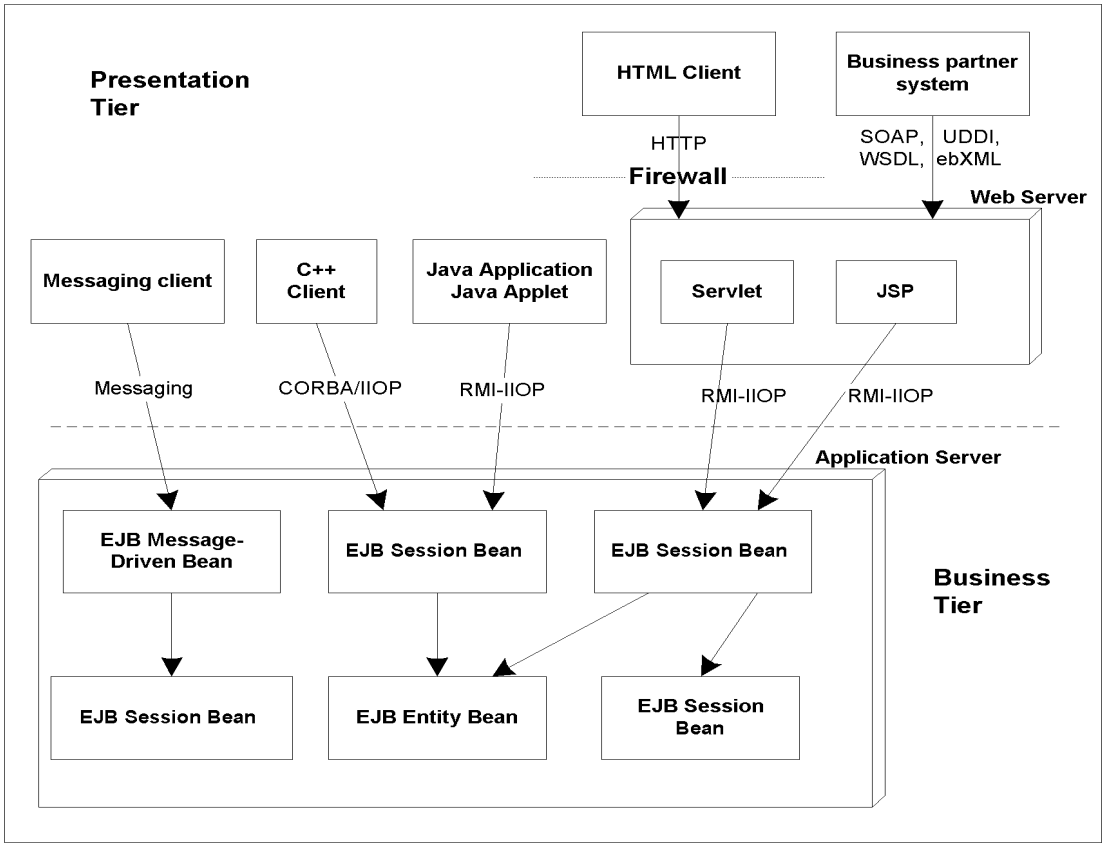
II. L'architecture de l'atelier

A rendre dans le compte rendu en évaluation du TP.

III. Rappel

Le standard EJB est une architecture de composants pour des composants serveur écrits en java. Il est adopté par l'industrie. "Train once, code anywhere" et facilement portable. EJB signifie deux choses : Une spécification et un ensemble d'interfaces et se divise en trois types :

- Session Beans :
  -Modélisent un traitement (business process)
  -Correspondent à des verbes, à des actions
  -Les actions impliquent des calculs, des accès à une base de données, consulter un service externe (appel téléphonique, etc.)
- Entity beans :
  -Modélisent des données
  -Correspondent à des noms
  -Ce sont des objets java qui cachent des données d'une base de données
  -Ce sont des objets persistants
  -Mapping base de donnée relationnelle/Objet facilité par EJB 2.0
- Message-Driven Beans
  -Introduits à partir de la norme EJB 2.0, nous sommes aujourd'hui en 3.0
  -Similaire aux Session bean : représentent des verbes ou des actions,
  -On les invoque en leur envoyant des messages.



IV. L'environnement de développement

- L'IDE Netbeans
- WAMP Server
- Le serveur d'application Glassfish

V. Création d'un projet EJB

a. Création du projet

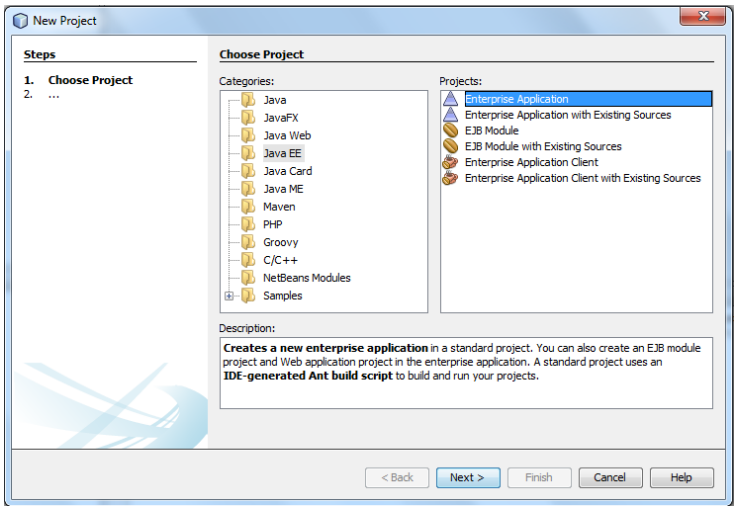
Vous allez essayer au cours de ce TP de créer une base de données avec WAMP et de faire une connexion avec l'IDE Netbeans.

La table que nous allons traiter est une table **« employeur »** qui contient les champs suivants : 'idemployeur : int, nom : varchar, prenom : varchar'. Avec idemployeur une clé primaire

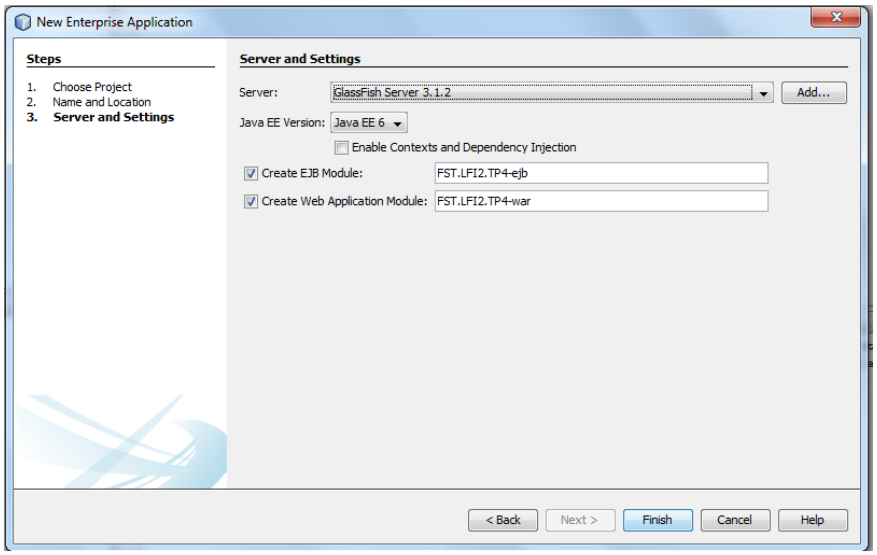
(Pensez à faire une table authentification qui vous permettra de traiter la partie authentification par la suite. Cette table contient les champs suivant « login : Sting, Password : int, Type : int »).

Vous allez utiliser Entreprise Java Beans (EJB) afin d'assurer la séparation entre les couches et de faire apparaitre le modèle MVC.

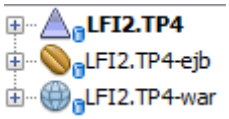
1. Commencez par créer une BD nommée 'test' contenant les tables 'employeur' et 'utilisateur' sous le WAMP Server et faites apparaitre cette base de données sous l'IDE Netbeans.
2. Créez un nouveau projet par le menu File > New Project-> java EE -> Entreprise Application. Vous nommerez ce projet LFI2.TP4 et vous finirez par cliquer sur Ok.



3. En cliquant sur « Next », vous allez sélectionner Glassfish Server comme serveur d'application sur lequel l'application va être déployée et Java EE 6 Comme version de Java EE.



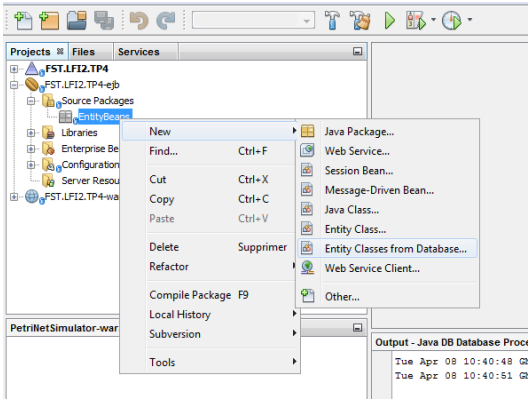
4. IDE Netbeans va créer trois projets nommés LFI2.TP4 (Enterprise Application Project), LFI2.TP4-ejb (EJB Project), LFI2.TP4-war (Web Project).



b. Création des « Entity classes » à partir de la base de données

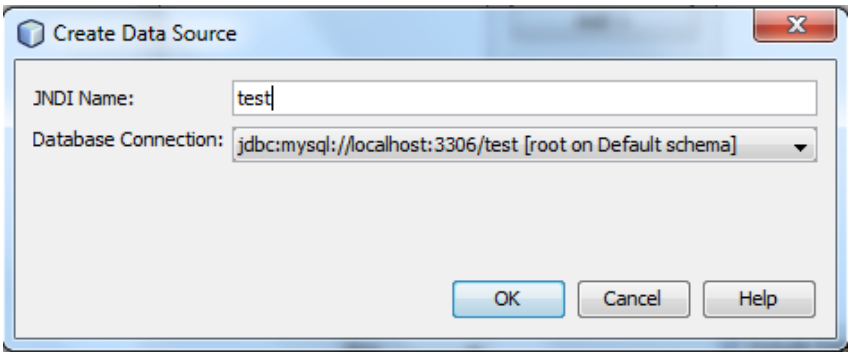
Avant toute chose, nous devons d'abord créer les Entity classes et vu que les Session classes sont responsables de la manipulation des données, ils seront créés dans le projet EJB.

1. Créez un nouveau package nommé « entities »
2. Dans la fenêtre Projects, cliquez-droit sur le package crée et sélectionnez "New> Entity Classes from Database".

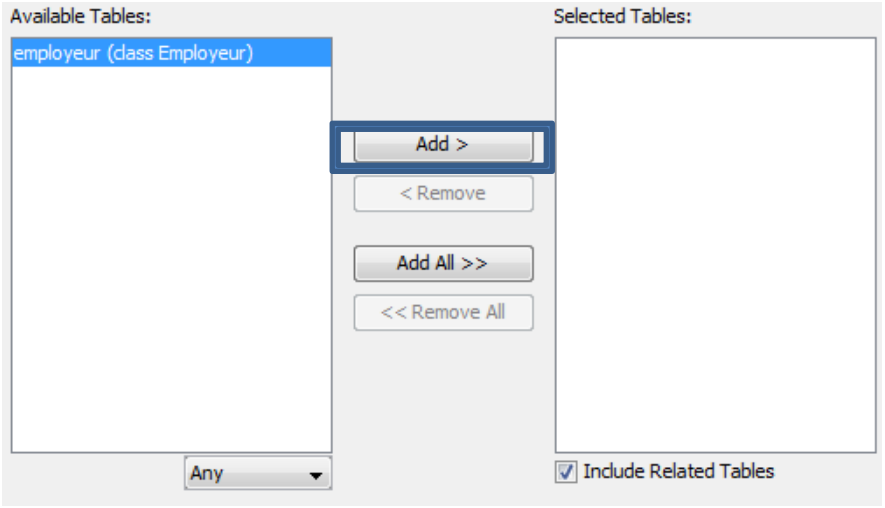


3. Ajoutez une nouvelle source de données

4. Choisissez la connexion que vous venez de créer et ajouter un nom à votre JNDI



5. Ajoutez toutes les tables disponibles comme suit :



6. La classe que vous allez créer sera nommée « Employeur » et le code de cette dernière sera généré automatiquement.

Cette classe est un entity EJB permettant de modéliser une donnée persistance pour être utilisée par le session bean.

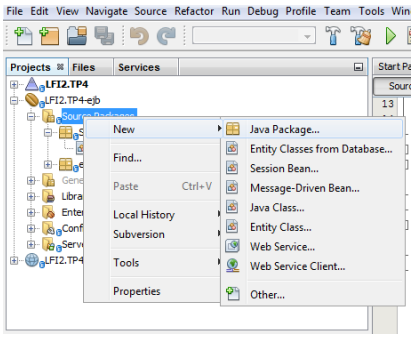
Le fichier « persistence.xml » est aussi créé automatiquement afin de contenir les informations liées à la source de données.

```
<?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/XML
  <persistence-unit name="LFI2.TP4-ejbPU" transaction-type="JTA">
    <jta-data-source>Test</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties/>
  </persistence-unit>
</persistence>
```

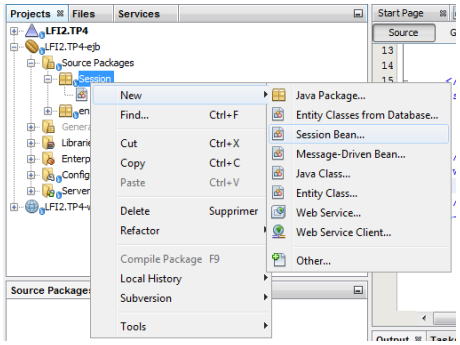
c. Création des « session classes »

Les sessions beans sont présents sous le projet EJB. Pour pouvoir les manipuler vous allez suivre les étapes suivantes :

1. Créez le nouveau package que vous nommerez « session »



2. Créer le « session bean » que vous nommerez « EmployerManager »



Cette classe contiendra le code suivant :

```
package Session;

import entities.Employeur;
import java.util.List;
import javax.ejb.Stateless;
import javax.ejb.LocalBean;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

/**
 *
 * @author Dell
 */
@Stateless
@LocalBean
public class EmployerManager {
    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")

    @PersistenceContext(unitName = "LFI2.TP4-ejbPU")
    private EntityManager em;


    public List<Employeur> getallemployers() {
        Query query = em.createNamedQuery("Employeur.findAll");
        return query.getResultList();
    }
```

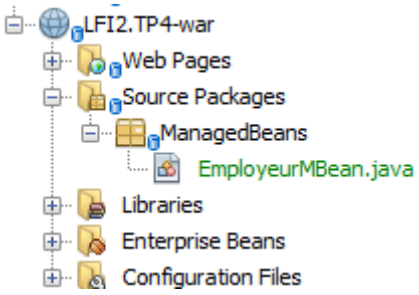
```
    public void persist(Object object) {
        em.persist(object);
    }


}
```

Remarque **:** Pour créer @PersistenceContext, cliquez droit sur la classe-> insert code-> use Entity Manager.

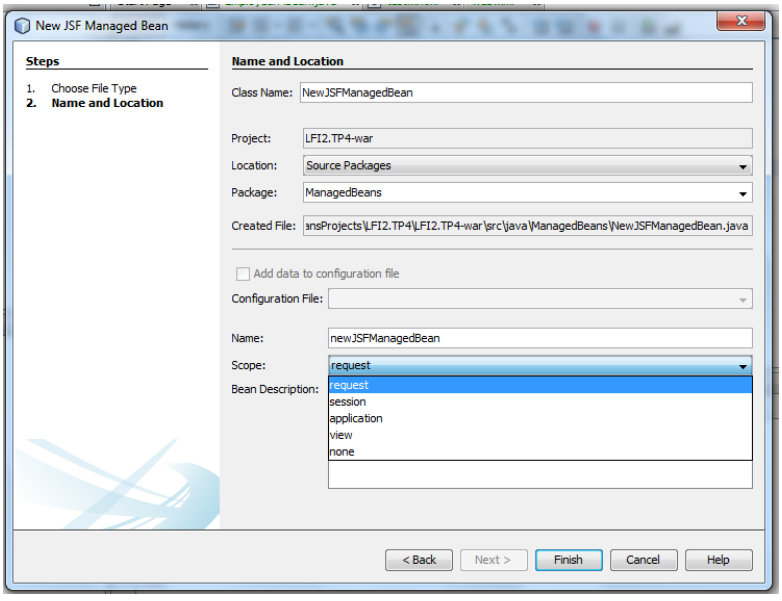
d. Création des Managed Beans

1. Créez sous le projet LFI2.TP4-war un package nommé ManagedBeans

2. Cliquez droit sur le package ManagedBeans-> JSF Managed beans -> EmployeurMBean.



3. Pensez à modifier le scope en le mettant 'session' :

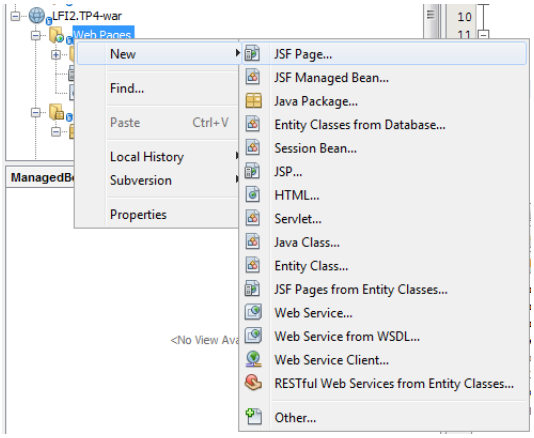


Le code que contiendra cette classe est le suivant :

```
package ManagedBeans;

import Session.EmployerManager;
import entities.Employeur;
```

```
import java.io.Serializable;
import java.util.List;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

/**
 *
 * @author Dell
 */
@ManagedBean
@SessionScoped
public class EmployeurMBean implements Serializable {

    @EJB
    private EmployerManager employerManager;
    private Employeur employeur;

    /**
     * Creates a new instance of EmployeurMBean
     */
    public EmployeurMBean() {
    }

    public List<Employeur> getEmployeurs() {
        return employerManager.getallemployers();
    }

    public Employeur getDetails() {
        return employeur;
    }
}
```

e. Création des pages Web



1. Nommez cette page ‘test’, elle aura comme extension .xhtml et contiendra le code xml suivant :

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    <f:view>
      <h:form>
        <h1><h:outputText value="List"/></h1>
        <h:dataTable value="#{employeurMBean.employeurs}" var="item">
          <h:column>
            <f:facet name="header">
              <h:outputText value="Idemployeur"/>
            </f:facet>
            <h:outputText value="#{item.idemployeur}"/>
          </h:column>
          <h:column>
            <f:facet name="header">
              <h:outputText value="Nom"/>
            </f:facet>
            <h:outputText value="#{item.nom}"/>
          </h:column>
          <h:column>
            <f:facet name="header">
              <h:outputText value="Prenom"/>
            </f:facet>
            <h:outputText value="#{item.prenom}"/>
          </h:column>
        </h:dataTable>
      </h:form>
    </f:view>

  </h:body>
</html>
```

VI. Test de l'application web

Avant de tester l'application, allez vers Web Pages -> WEB-INF-> web.xml et faites les modifications suivantes pour que la première page qui s'ouvre lors de l'exécution soit test.xhtml :

```
<welcome-file-list>
    <welcome-file>faces/test.xhtml</welcome-file>
  </welcome-file-list>
```

Cliquez droit sur le projet 'Entreprise Application'-> run. Ceci déclenchera automatiquement le déploiement du projet sous Glassfish.

La page web (JSF) s'exécutera au niveau du browser pour afficher la liste des employés

List

Idemployeur	Nom	Prenom
1	TAKTAK	Hajer
2	ZAIBI	Dorra
3	SMITH	Jhon
4	BELL	Jim

VII. C'est à vous

1. Quels sont les éléments encapsulés dans la classe « Employeur » ?
2. Ou est-ce que le modèle MVC apparait dans ce que vous venez d'implémenter ?
3. Comment est assurée la communication entre ces classes ?
4. Expliquez d'avantage les fichiers ayant comme extension .xhtml.
5. Illustrez par un schéma la communication entre les classes.