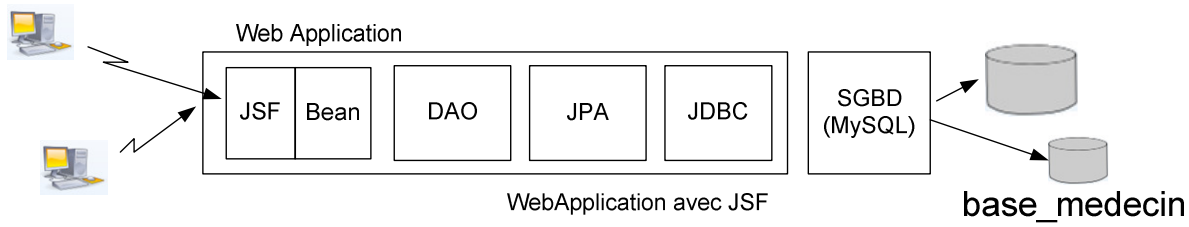
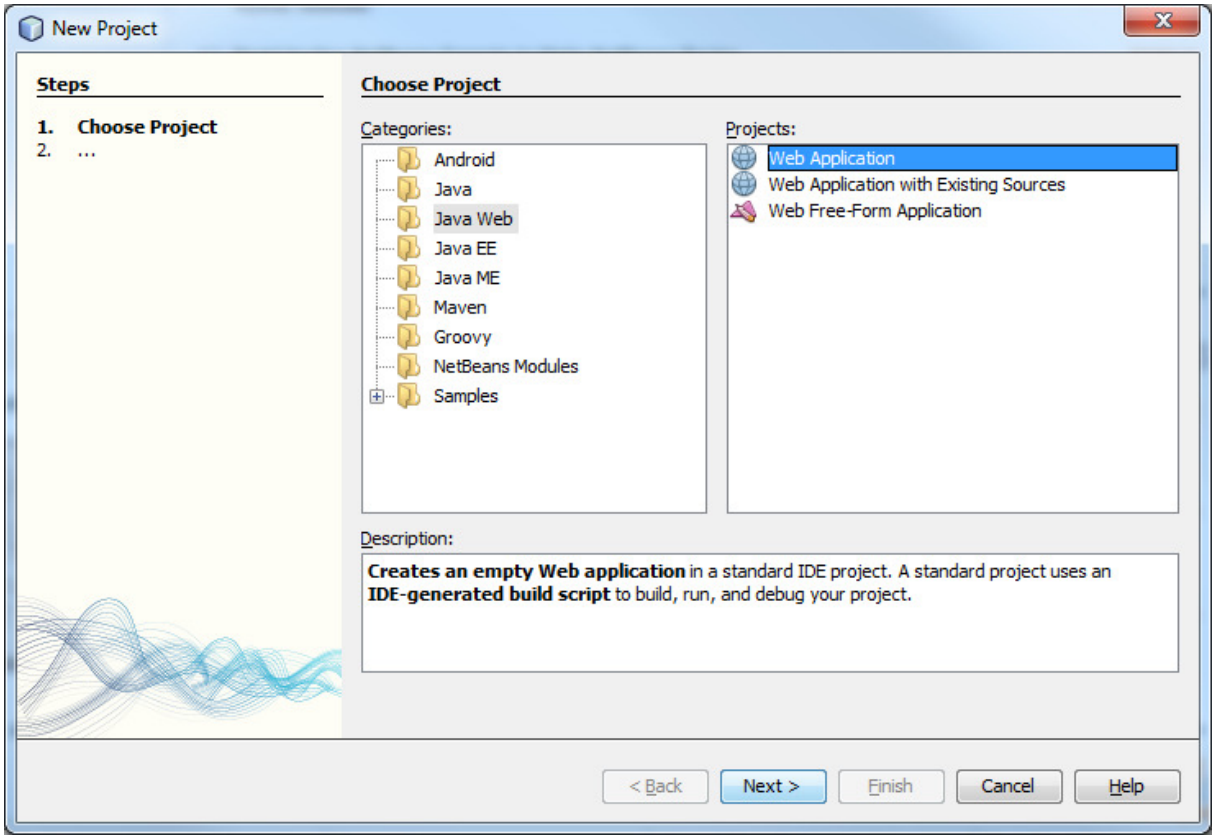
Création d'une Application Web / JSF

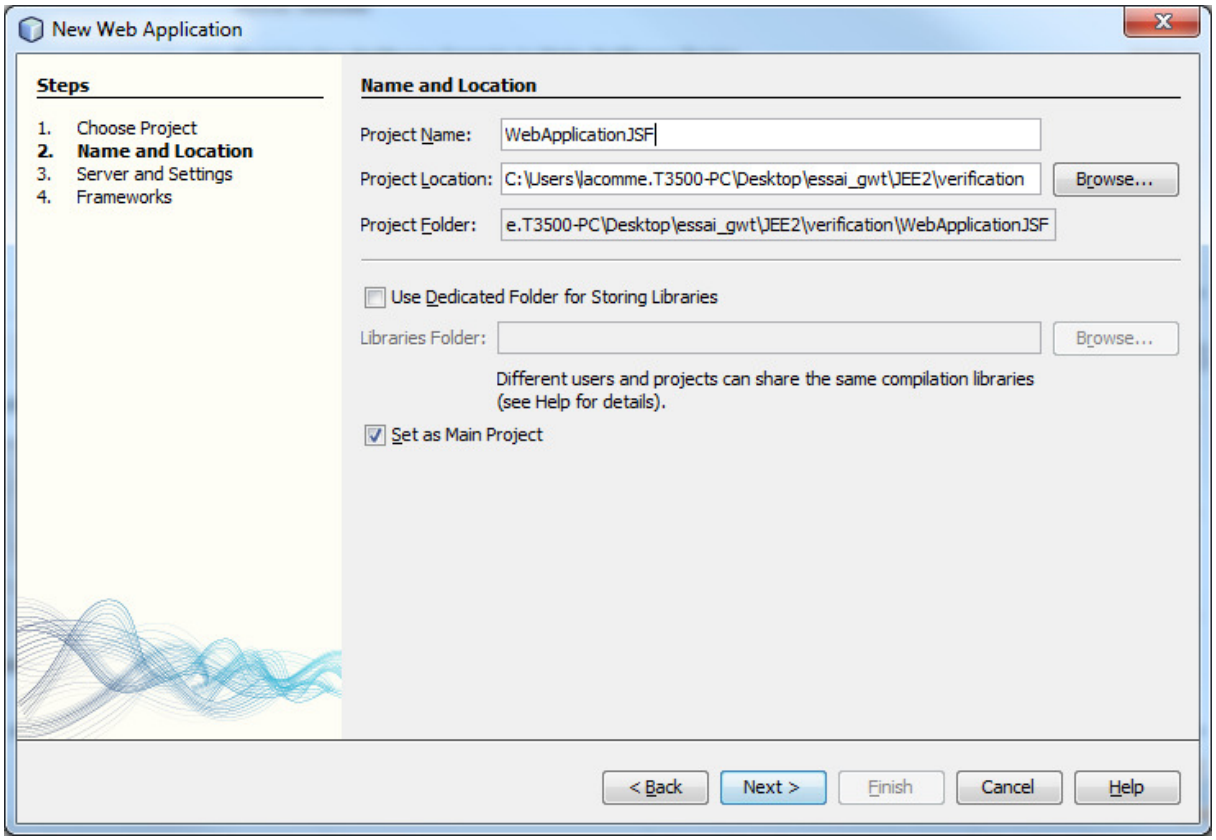
Rédacteurs : Alexandre Baillif, Philippe Lacomme et Raksmey Phan
Date : juillet 2010
Avertissement : ce document est une reprise d'une partie d'un document écrit par Serge Tahé. En particulier la base de données utilisée.

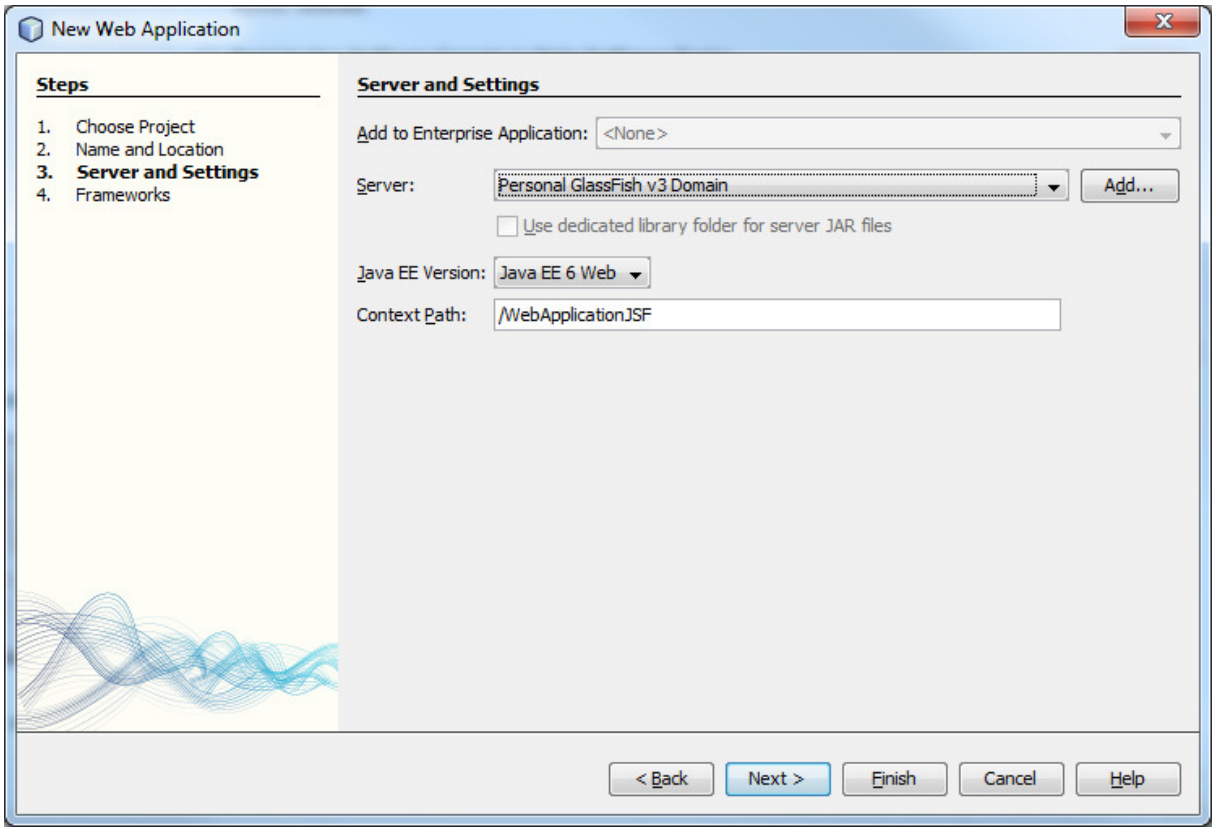


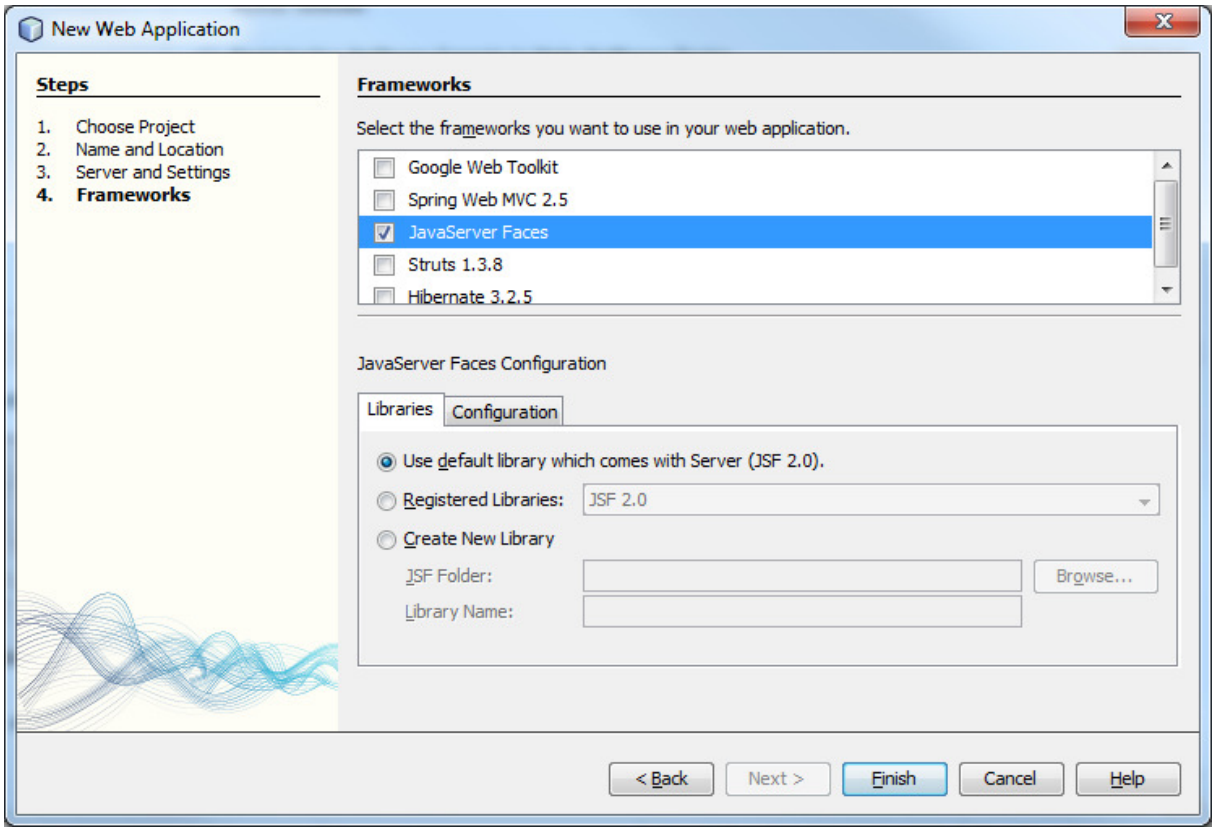
La particularité de cette solution est que l'on n'utilise pas le concept d'EJB. En effet, les beans et pages web vont directement venir se greffer sur la couche DAO. Cette solution peut paraître plus longue car on recrée tout depuis le début (à partir de JPA) mais en fait, elle est rapide à mettre en place.

1) Créer une web application

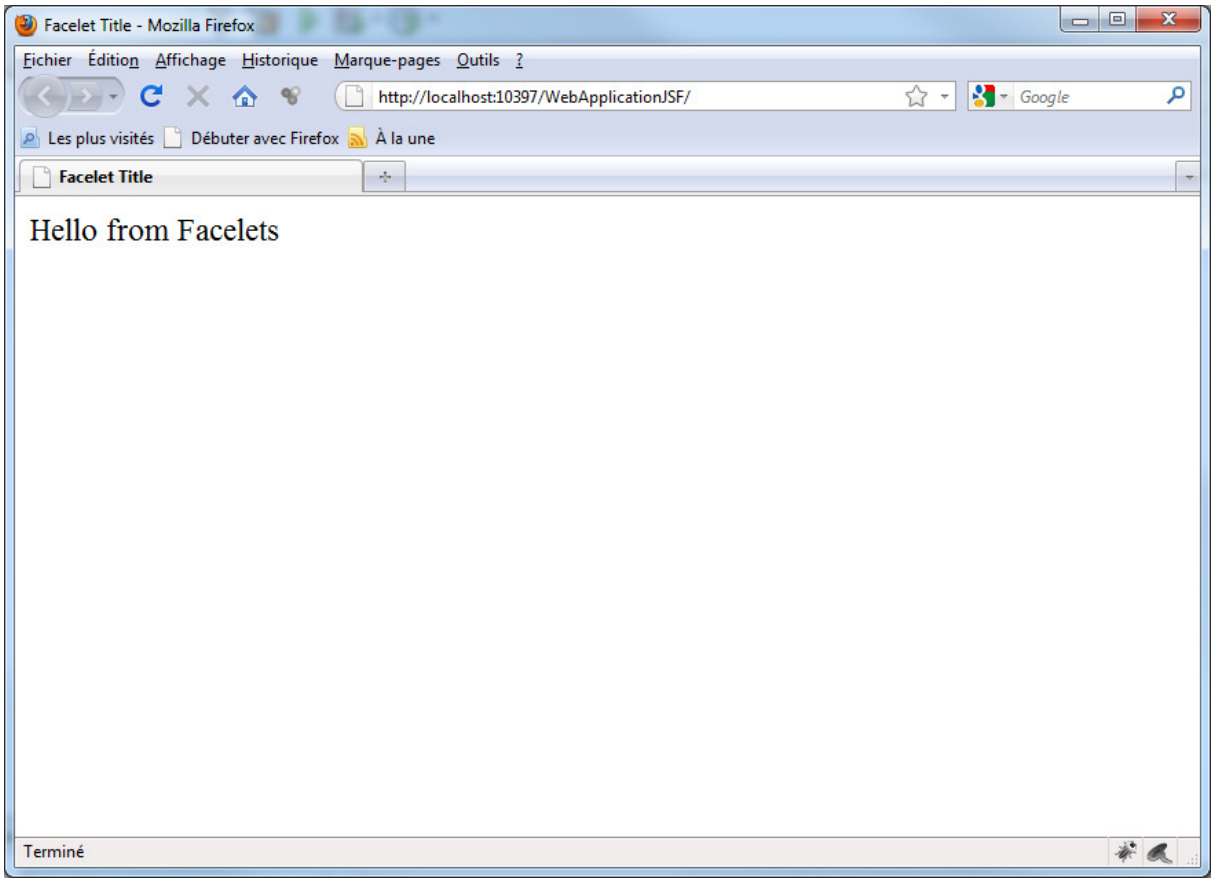





New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Server and Settings
Add to Enterprise Application: <None>
Server: Personal GlassFish v3 Domain
Add...
Use dedicated library folder for server JAR files
Java EE Version: Java EE 6 Web
Context Path: /WebApplicationJSF
< Back
Next >
Finish
Cancel
Help


New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Frameworks
Select the frameworks you want to use in your web application.
Google Web Toolkit
Spring Web MVC 2.5
JavaServer Faces
Struts 1.3.8
Hibernate 3.2.5
JavaServer Faces Configuration
Libraries
Configuration
Use default library which comes with Server (JSF 2.0).
Registered Libraries: JSF 2.0
Create New Library
JSF Folder:
Browse...
Library Name:
< Back
Next >
Finish
Cancel
Help

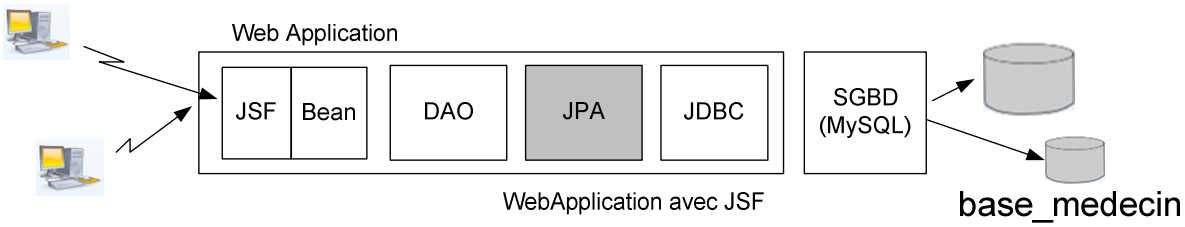
Le projet doit donner à l'exécution :

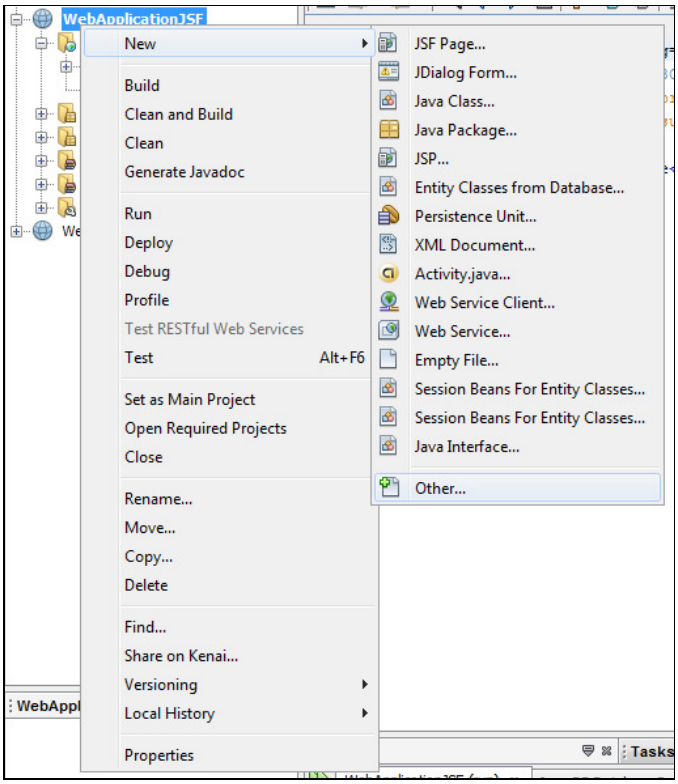


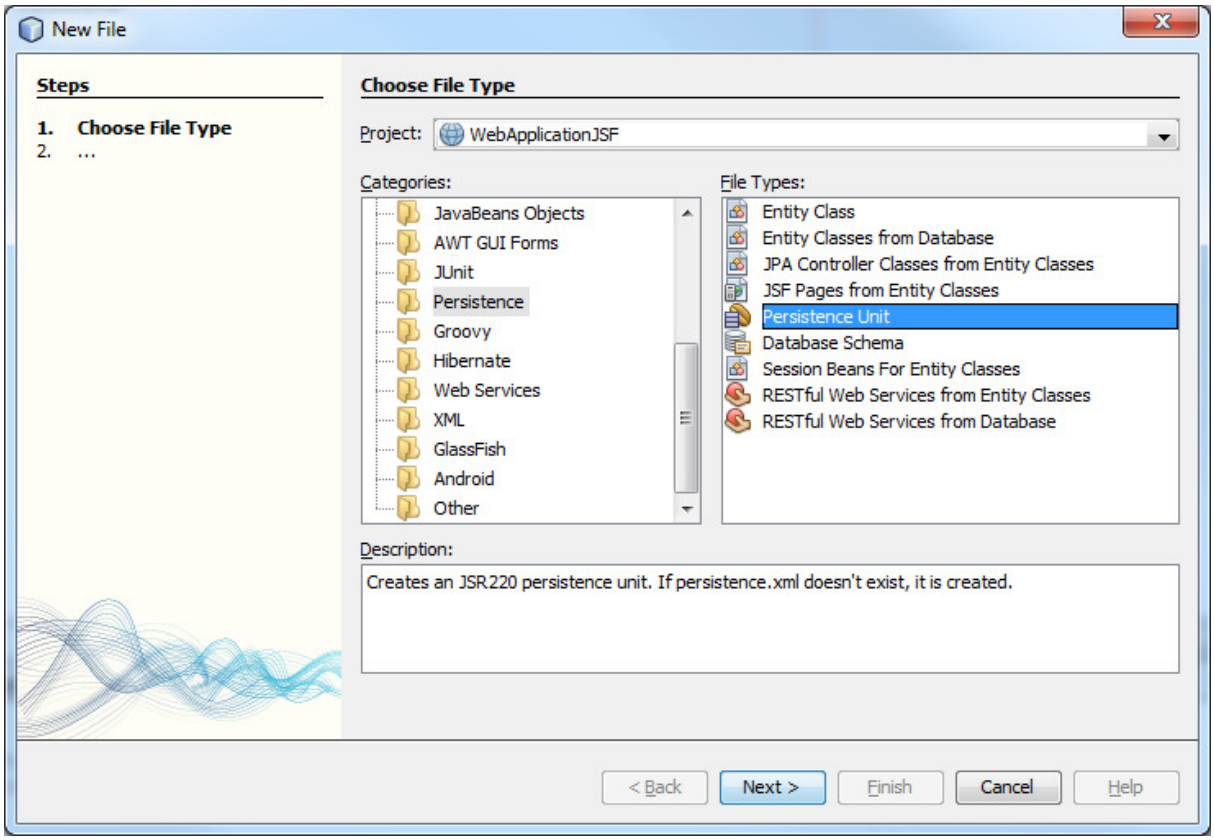
2) Créer la couche dao et jpa

Cette partie reprend certains points du cours 1.

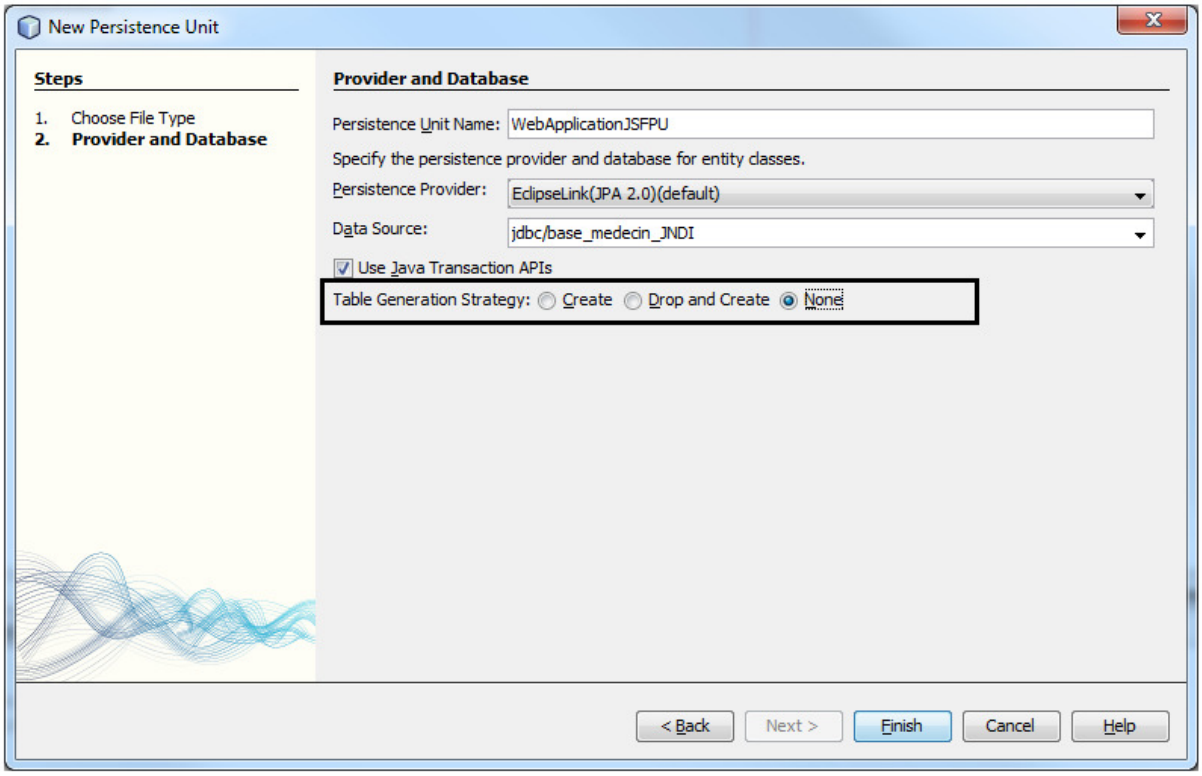
Partie 2.1. Dans un **premier temps** nous allons créer une unité de persistance (faire Clic Droit sur WebApplicationJSP et choisir New->Others).





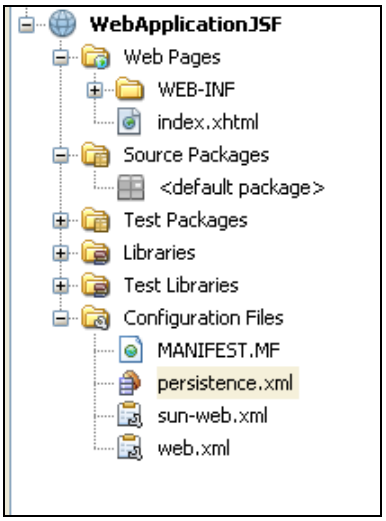


Choisir ensuite comme Data Source : base_medecin_JNDI
Et comme fournisseur de service de persistance EclipseLink.

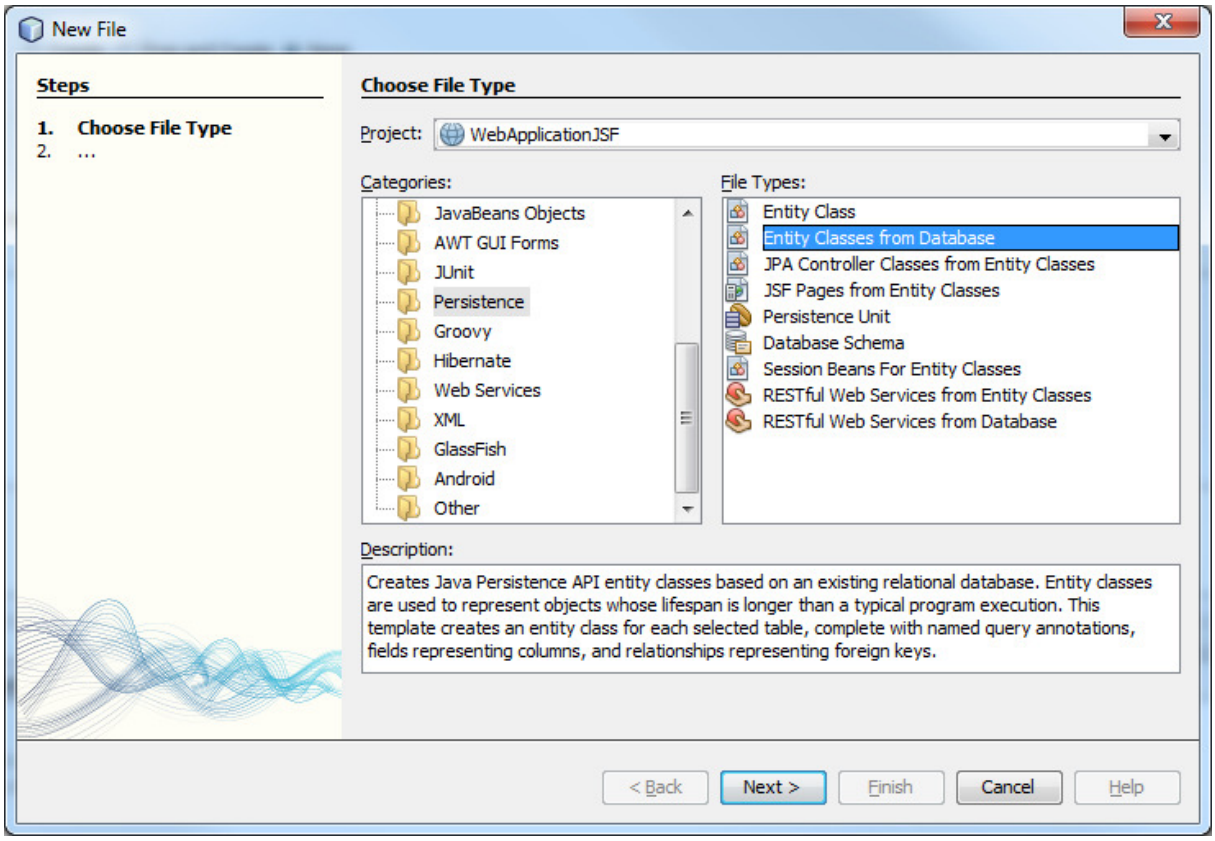


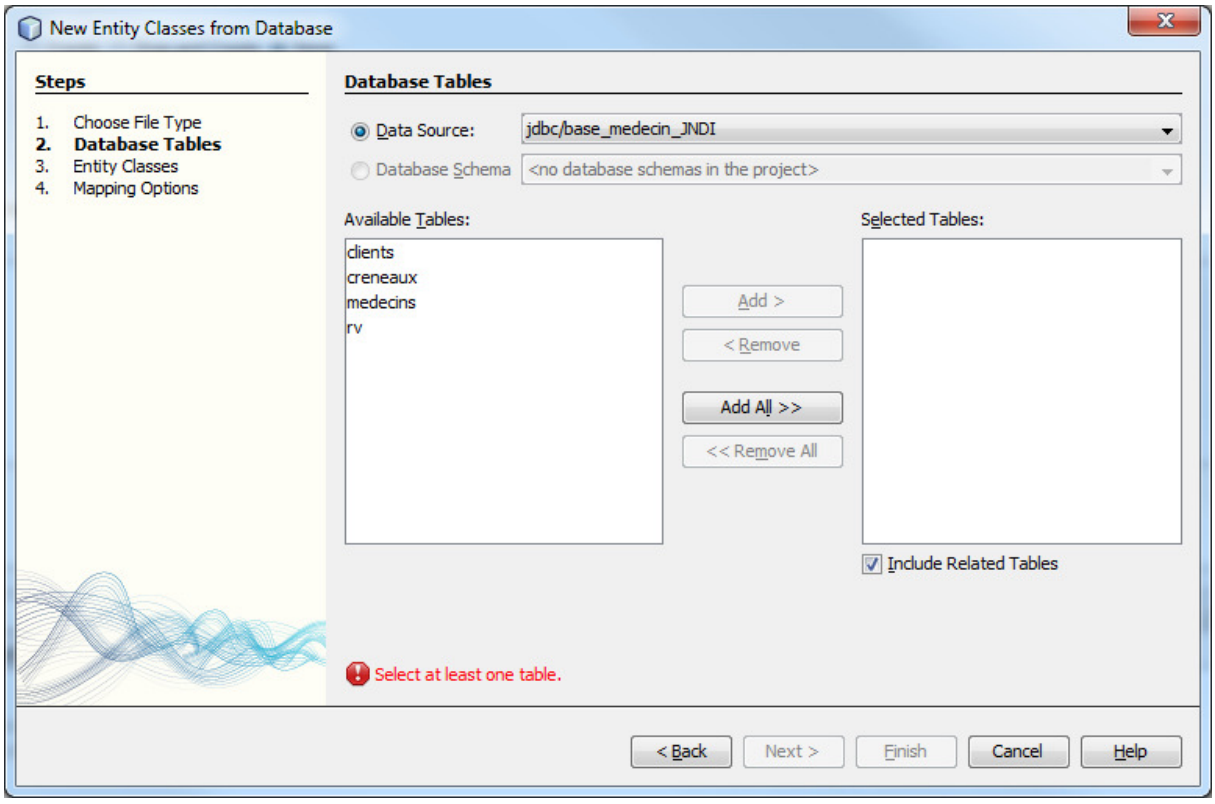
Attention à ne pas choisir une stratégie englobant la génération des tables. Elles existent déjà grâce au script SQL que nous avons utilisé au départ.

Dans la partie « Configuration Files », le fichier **persistence.xml** apparaît.

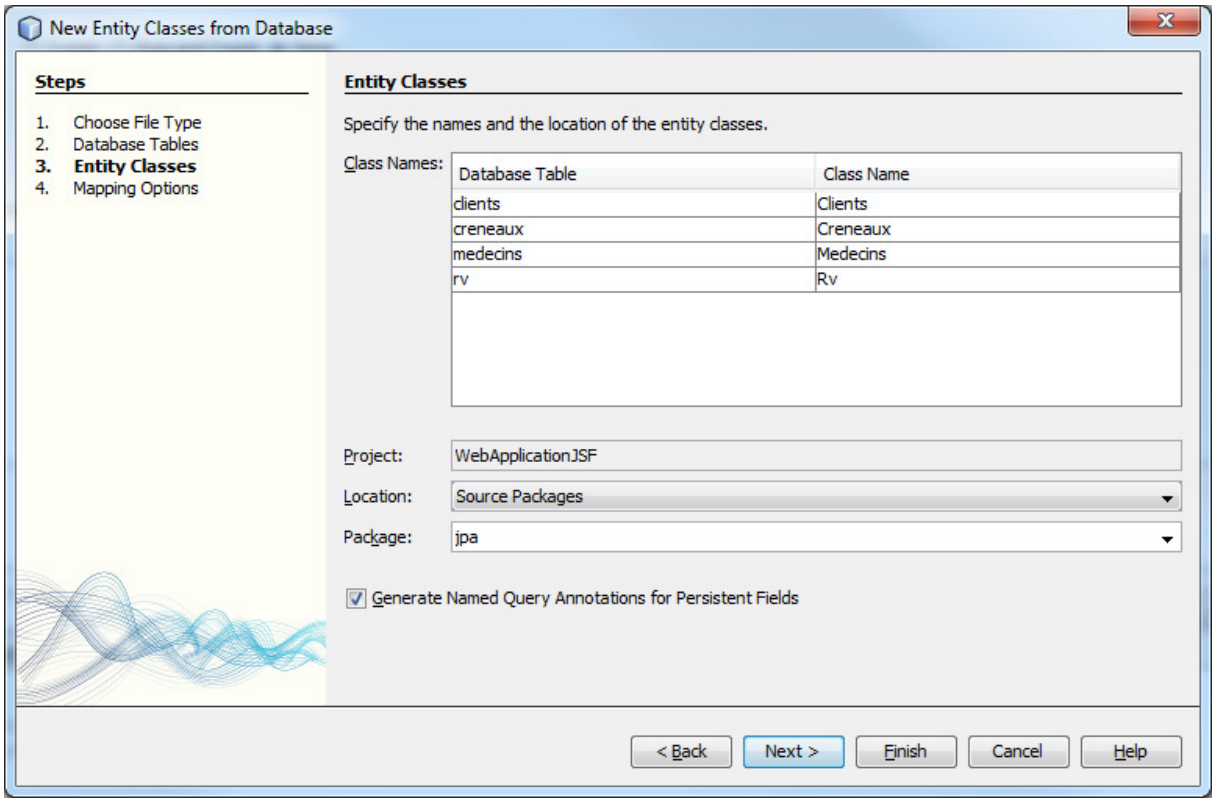


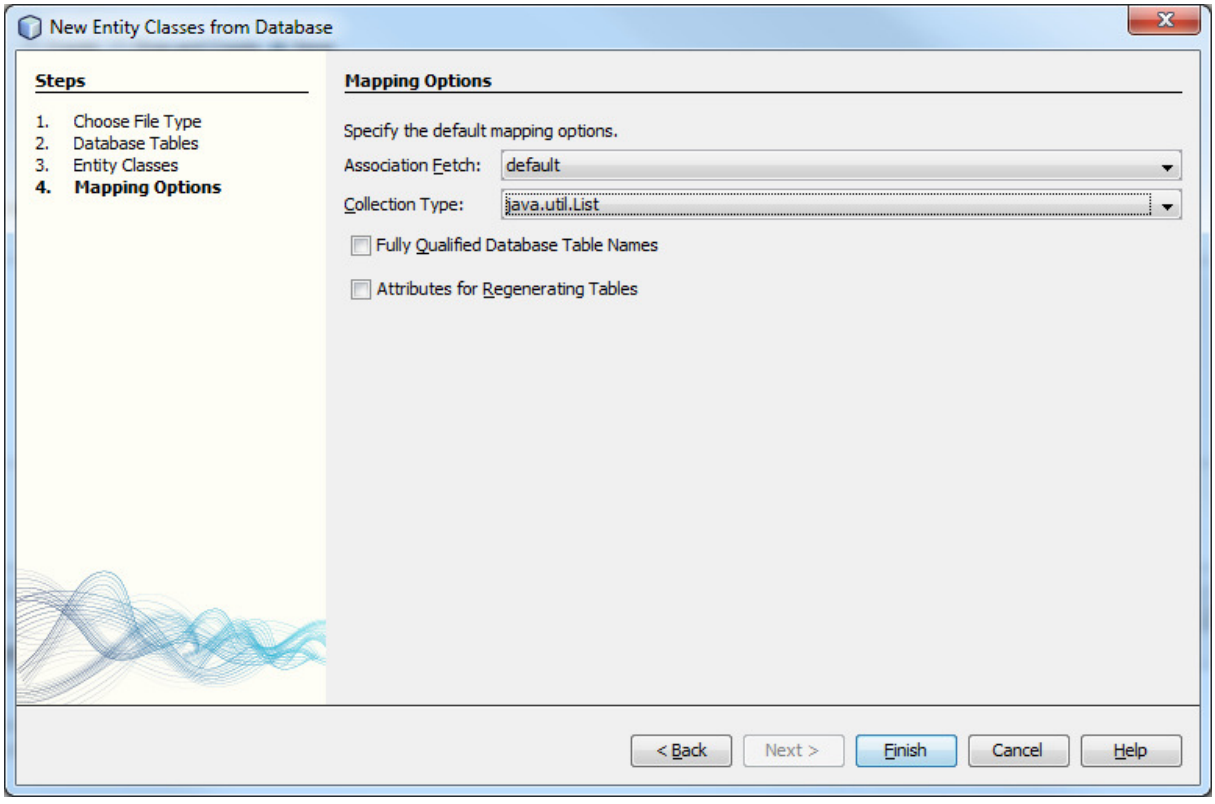
Partie 2.2. Dans un **deuxième temps**, nous allons créer des entités JPA. Comme précédemment faire Clic Droit / New / Other.


New File
Steps
1. Choose File Type
2. ...
Choose File Type
Project: WebApplicationJSF
Categories:
JavaBeans Objects
AWT GUI Forms
JUnit
Persistence
Groovy
Hibernate
Web Services
XML
GlassFish
Android
Other
File Types:
Entity Class
Entity Classes from Database
JPA Controller Classes from Entity Classes
JSF Pages from Entity Classes
Persistence Unit
Database Schema
Session Beans For Entity Classes
RESTful Web Services from Entity Classes
RESTful Web Services from Database
Description:
Creates Java Persistence API entity classes based on an existing relational database. Entity classes are used to represent objects whose lifespan is longer than a typical program execution. This template creates an entity class for each selected table, complete with named query annotations, fields representing columns, and relationships representing foreign keys.
< Back
Next >
Finish
Cancel
Help


New Entity Classes from Database
Steps
1. Choose File Type
2. Database Tables
3. Entity Classes
4. Mapping Options
Database Tables
Data Source: jdbc/base_medecin_JNDI
Database Schema <no database schemas in the project>
Available Tables:
clients
creneaux
medecins
rv
Add >
< Remove
Add All >>
<< Remove All
Selected Tables:
Include Related Tables
Select at least one table.
< Back
Next >
Finish
Cancel
Help

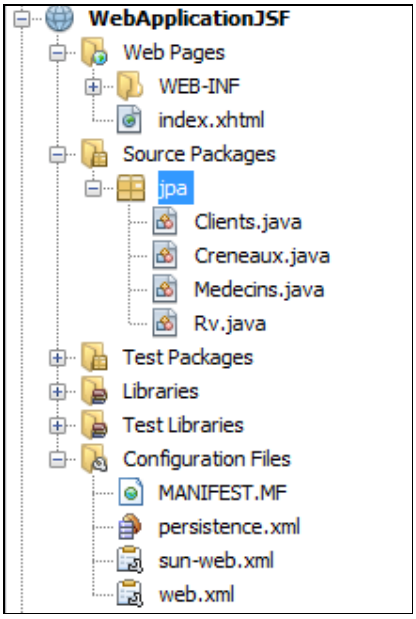
Etant donné qu'il s'agit de la génération d'entités JPA, on peut choisir JPA comme nom de package.





Veuillez à bien choisir « **java.util.List** » pour Collection Type (pour avoir plus de lisibilité).

Le projet se présente maintenant comme suit :

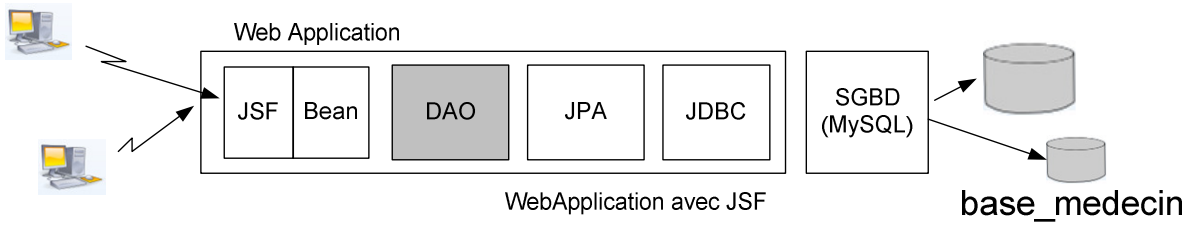


Dans la classe **Rv** nous ajoutons un nouveau constructeur permettant de créer un rendez vous en donnant une date, un client et un jour.

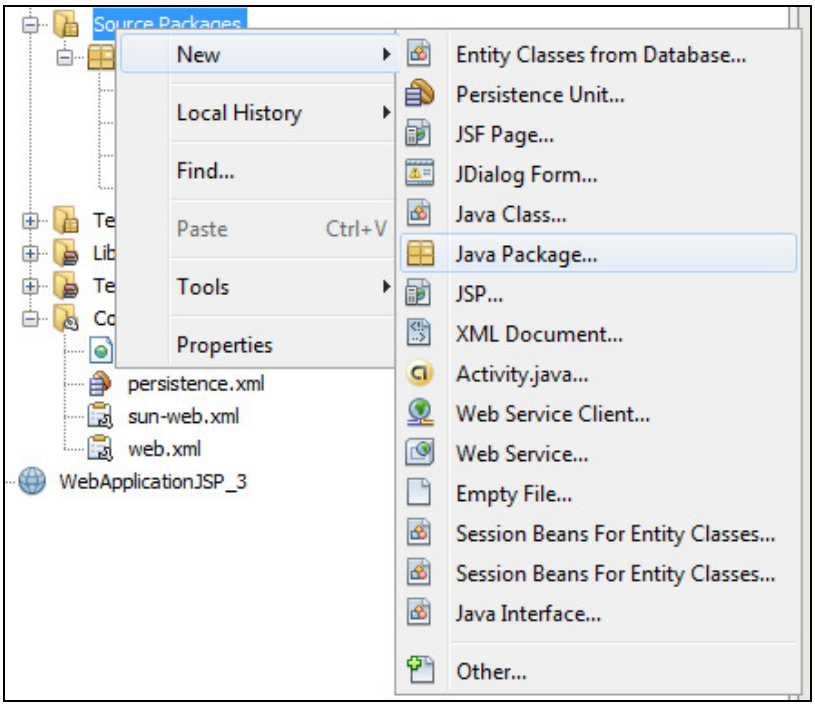
```
    public Rv(Date jour, Clients client, Creneaux creneau) {
        this.jour = jour;
        this.idClient = client;
        this.idCreneau = creneau;
    }
```

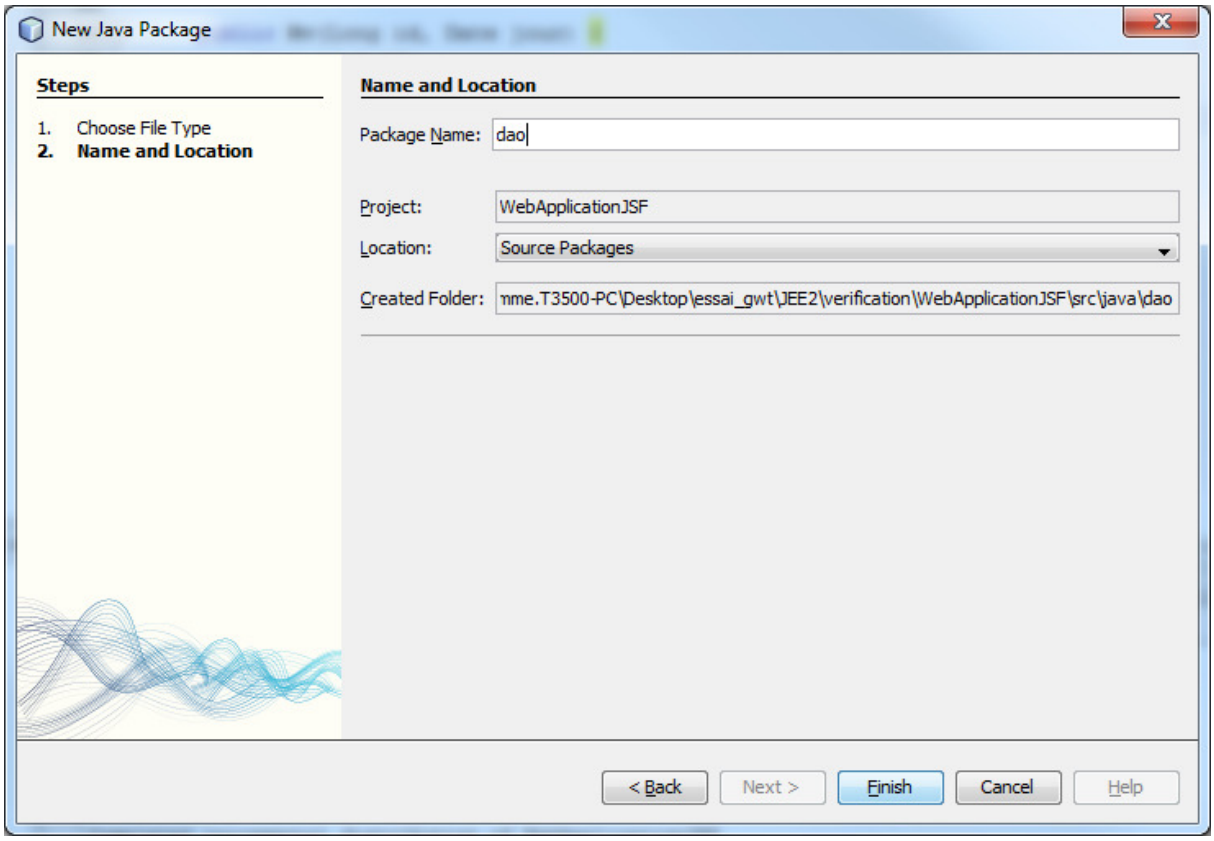
Partie 2.3. Création de la couche d'accès aux données

Enfin nous allons créer un package nommée **dao**, qui représente l'interface de l'EJB pour l'accès aux données.

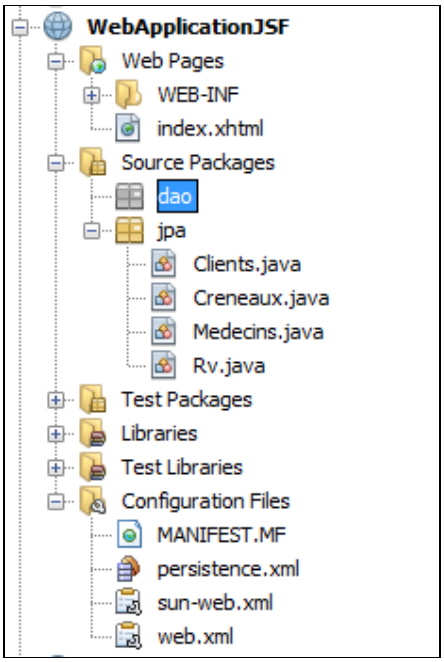


Créer un nouveau package nommé par exemple **dao**.



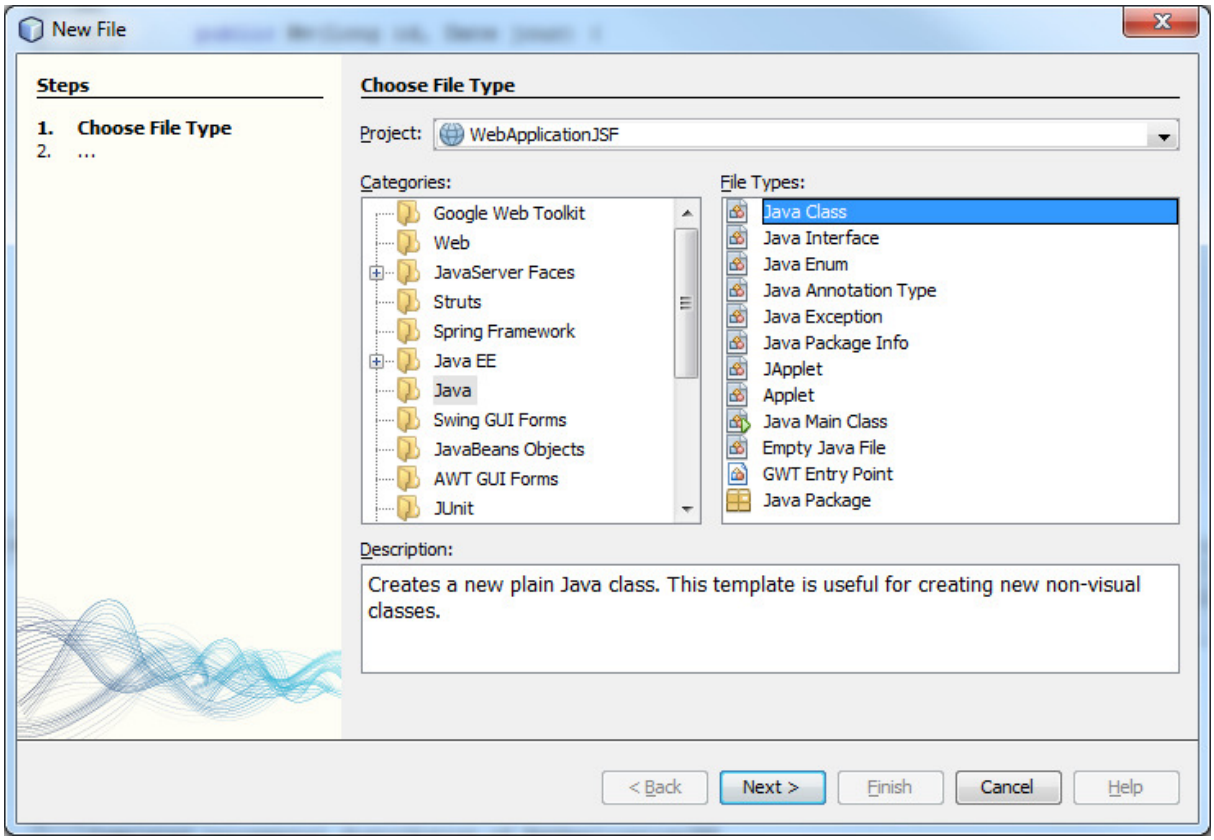


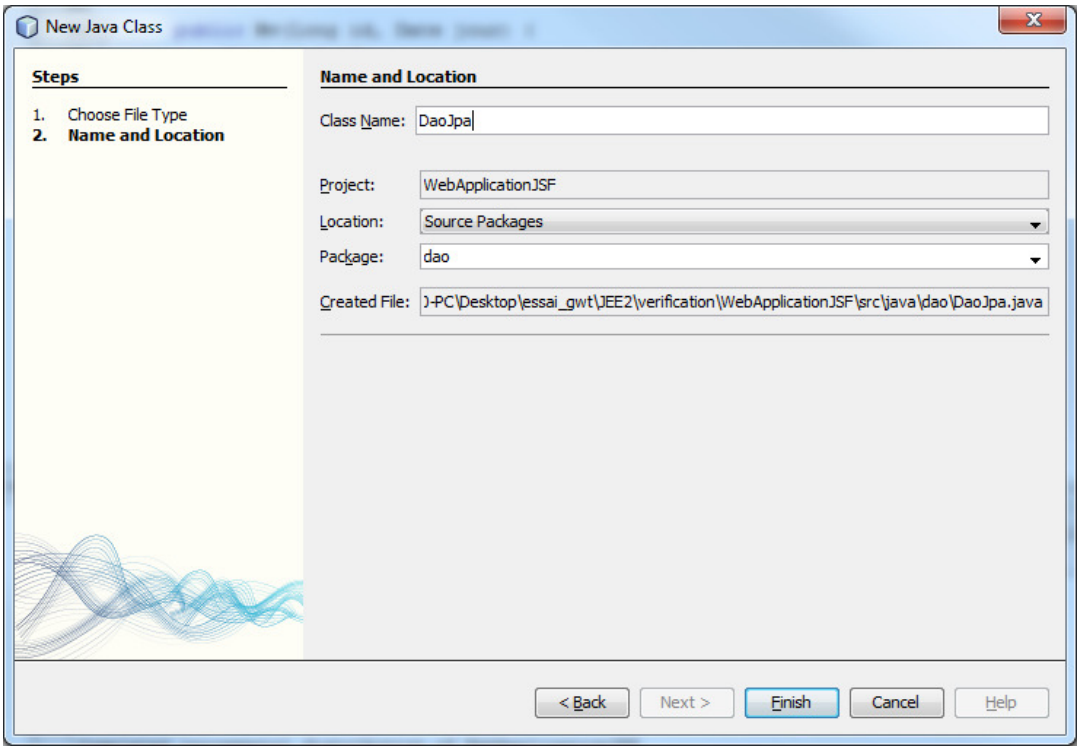
Le projet se présente alors comme suit :



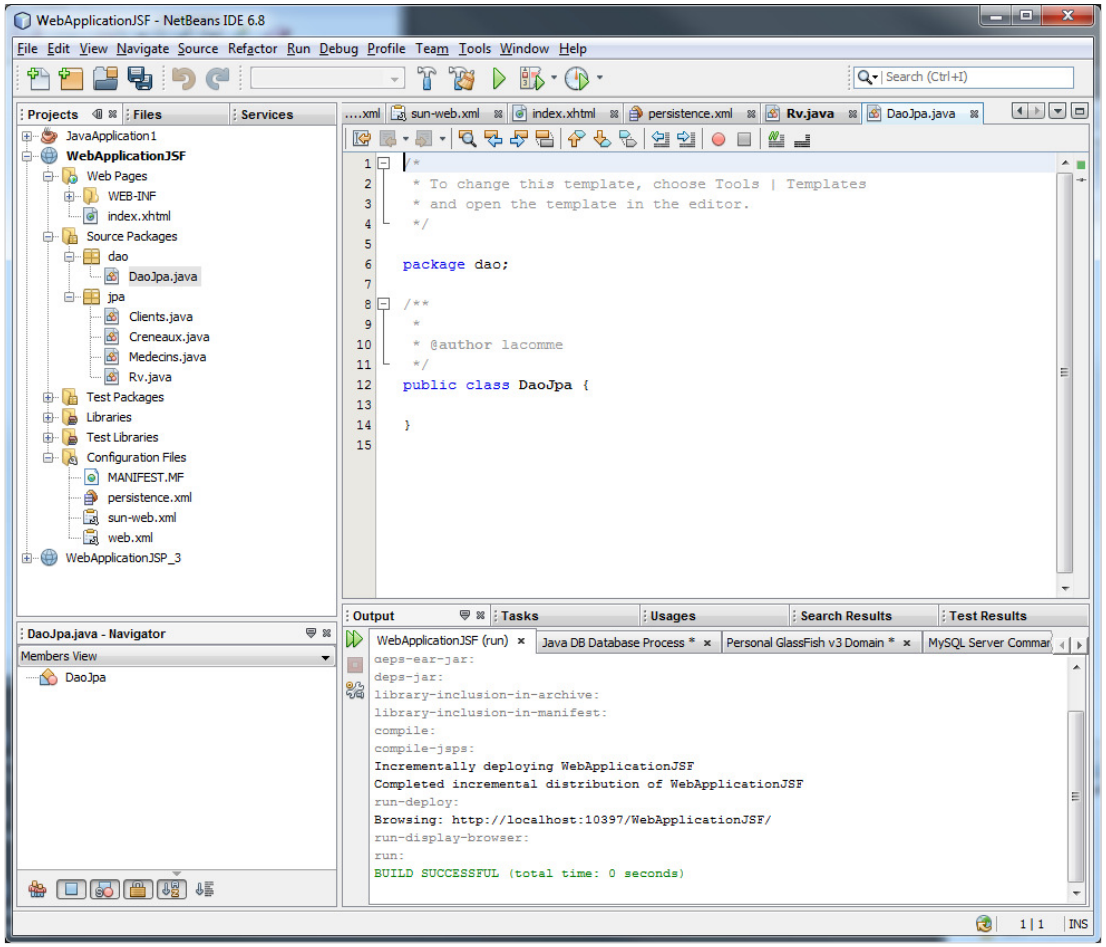
Ajoutons une classe **DaoJpa** qui contiendra l'intelligence métier.

Faire **New / Java / Java Class**.





Ceci donne une classe initialement vide.



Il faut rajouter le code suivant dans le fichier DaoJpa.java.

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package dao;

import java.text.SimpleDateFormat;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import jpa.Clients;
import jpa.Creneaux;
import jpa.Medecins;
import jpa.Rv;

public class DaoJpa {
 private EntityManager em;

    private EntityManagerFactory emf;

    private EntityTransaction tx ;

    public void init()
    {
        emf = Persistence.createEntityManagerFactory("WebApplicationJSFPU");

        em = emf.createEntityManager();

        tx = em.getTransaction();

        tx.begin();
    }

    public void close()
    {

        em.close();

        emf.close();
    }

    // liste des clients
    public List<Clients> getAllClients() {
        try {
            return em.createQuery("select c from Clients c").getResultList();
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }

    // liste des mÃ©decins
    public List<Medecins> getAllMedecins() {
        try {
            return em.createQuery("select m from Medecins m").getResultList();
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }

    // liste des crÃ©neaux horaires d'un mÃ©decin donnÃ©
    // medecin : le mÃ©decin
    public List<Creneaux> getAllCreneaux(Medecins medecin) {
        try {
            return em.createQuery("select c from Creneaux c join c.medecin m where
m.id=:idMedecin").setParameter("idMedecin", medecin.getId()).getResultList();
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }
```

```
    // liste des Rv d'un mÃ©decin donnÃ©, un jour donnÃ©
    // medecin : le mÃ©decin
    // jour : le jour
    public List<Rv> getRvMedecinJour(Medecins medecin, String jour) {
        try {
            return em.createQuery("select rv from Rv rv join rv.creneau c join c.medecin m where
m.id=:idMedecin and rv.jour=:jour").setParameter("idMedecin", medecin.getId()).setParameter("jour",
new SimpleDateFormat("yyyy:MM:dd").parse(jour)).getResultList();
        }catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }

    // ajout d'un Rv
    public Rv ajouterRv(String jour, Creneaux creneau, Clients client) {
        try {
            Rv rv = new Rv(new SimpleDateFormat("yyyy:MM:dd").parse(jour), client, creneau);
            em.persist(rv);
            return rv;
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }

    // suppression d'un Rv
    // rv : le Rv supprimÃ©
    public void supprimerRv(Rv rv) {
        try {
            em.remove(em.merge(rv));
        }catch (Exception e) {
         e.printStackTrace();

        }
    }

    // recuperer un client donnÃ©
    public Clients getClientById(Long id) {
        try {
            return (Clients) em.find(Clients.class, id);
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }

    // recuperer un mÃ©decin donnÃ©
    public Medecins getMedecinById(Long id) {
        try {
            return (Medecins) em.find(Medecins.class, id);
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }

    // recuperer un Rv donnÃ©
    public Rv getRvById(Long id) {
        try {
            return (Rv) em.find(Rv.class, id);
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }

    // recuperer un crÃ©neau donnÃ©
    public Creneaux getCreneauById(Long id) {
        try {
            return (Creneaux) em.find(Creneaux.class, id);
        } catch (Exception e) {
         e.printStackTrace();
         return null;
        }
    }
}
```

Remarques

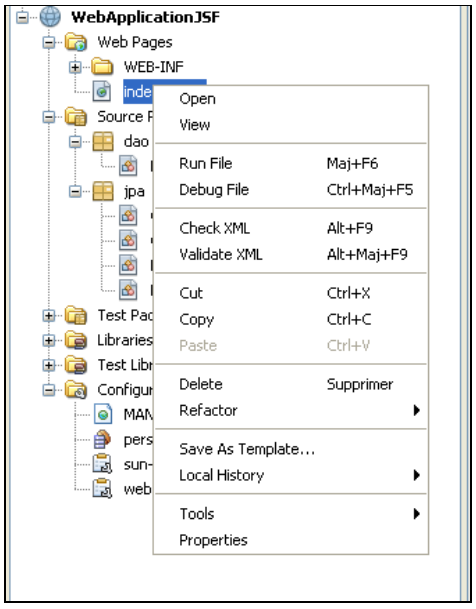
Notez que nous définissons une méthode **Init()** et une méthode **close()**. Ces deux méthodes vont nous être indispensables par la suite.

```
public void init()
{
    emf = Persistence.createEntityManagerFactory("WebApplicationJSFPU");

    em = emf.createEntityManager();

    tx = em.getTransaction();

    tx.begin();
}
```

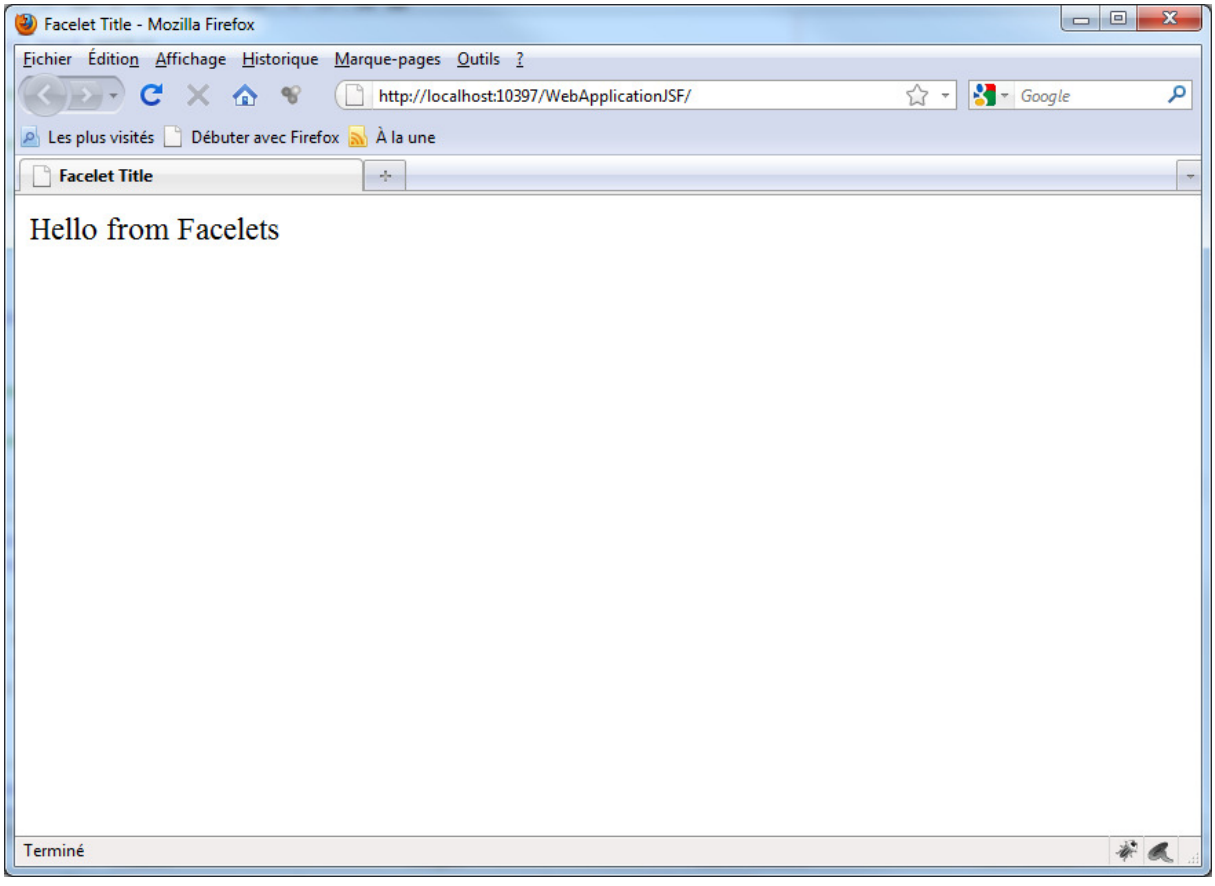
```
public void close()
{

    em.close();

    emf.close();
}
```

3) Ajout de page JSF

3.1. Remarques sur le fichier **index.xtml**
On peut le visualiser simplement par un clic droit (**View** ou **Run File**).



Ceci doit donner la visualisation suivante :

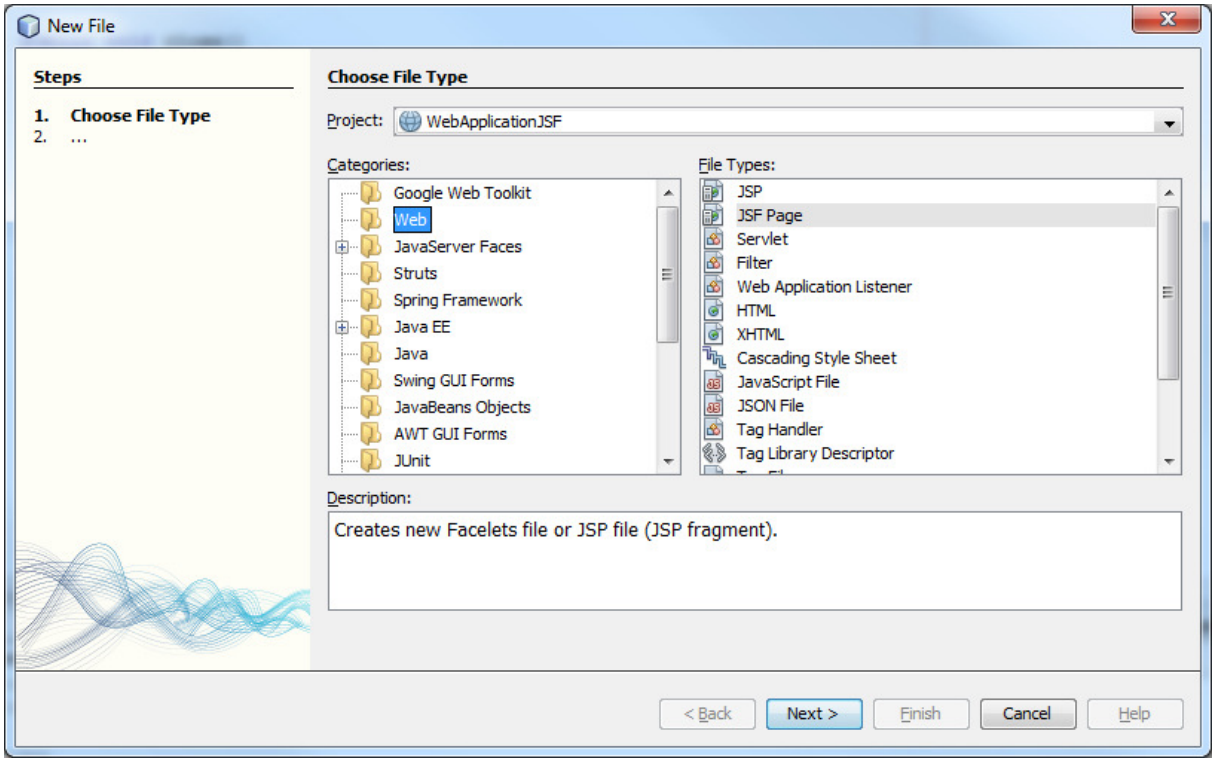


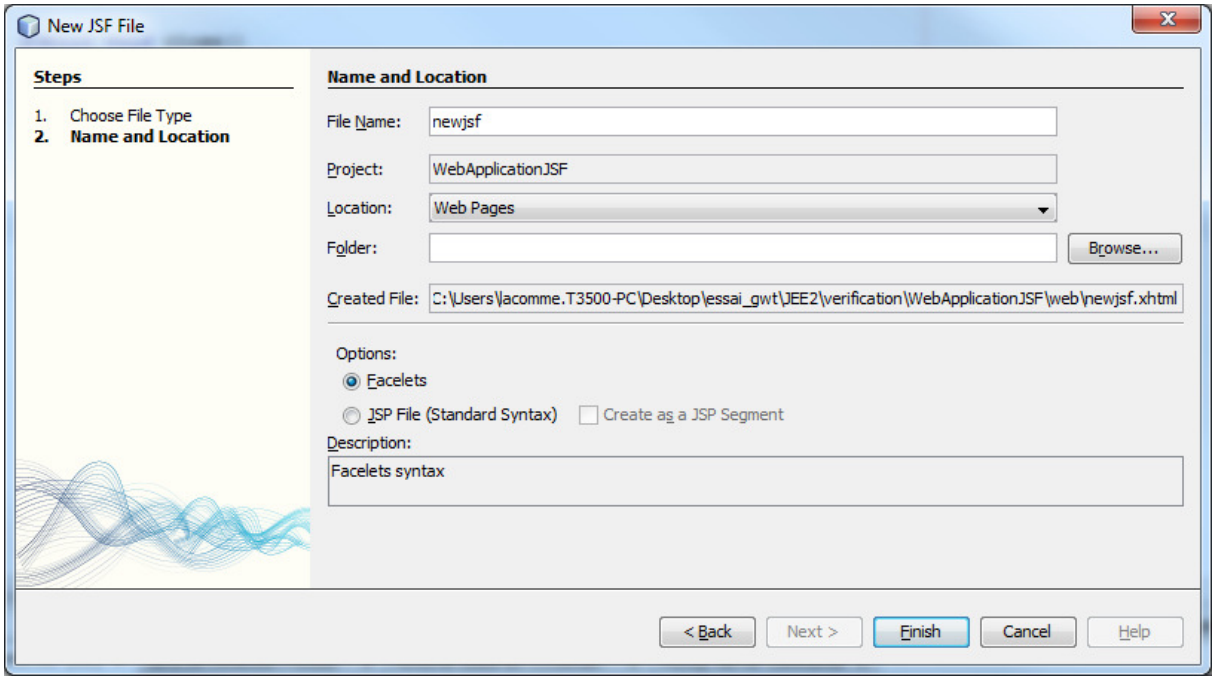
Lorsque que vous faites un clic droit sur le projet et cliquer sur « Run » vous devriez arriver au même résultat étant donné que la page « index.xhtml » est la page d'accueil du projet.

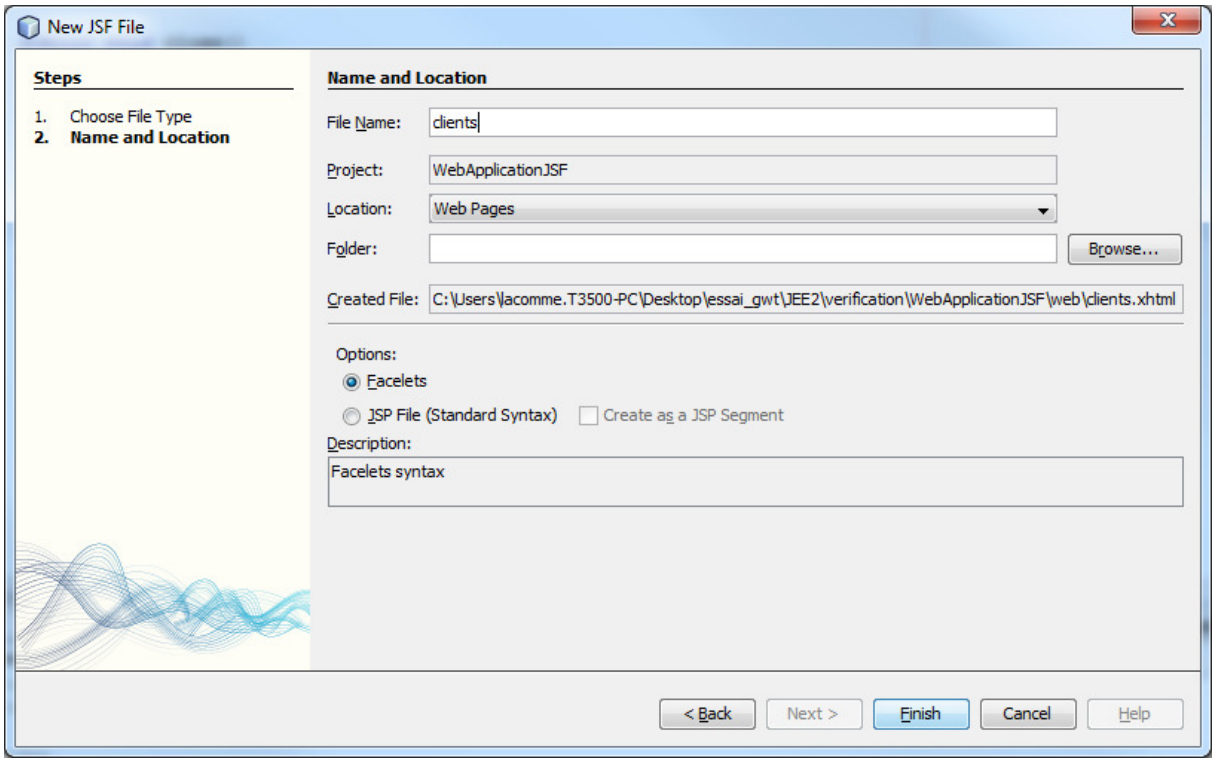
NB : Remarquez l'extension « .xhtml » qui ici sera spécifique aux pages JSF.

3.2. Ajout d'une page **JSF**

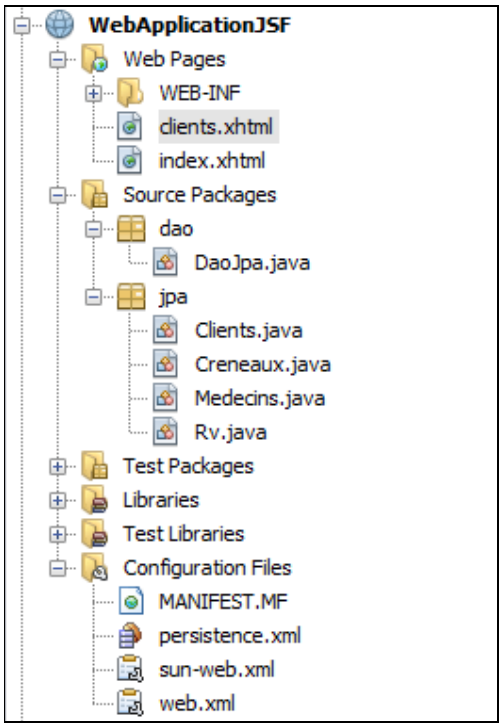
Faire un clic droit et choisir JSF Page.







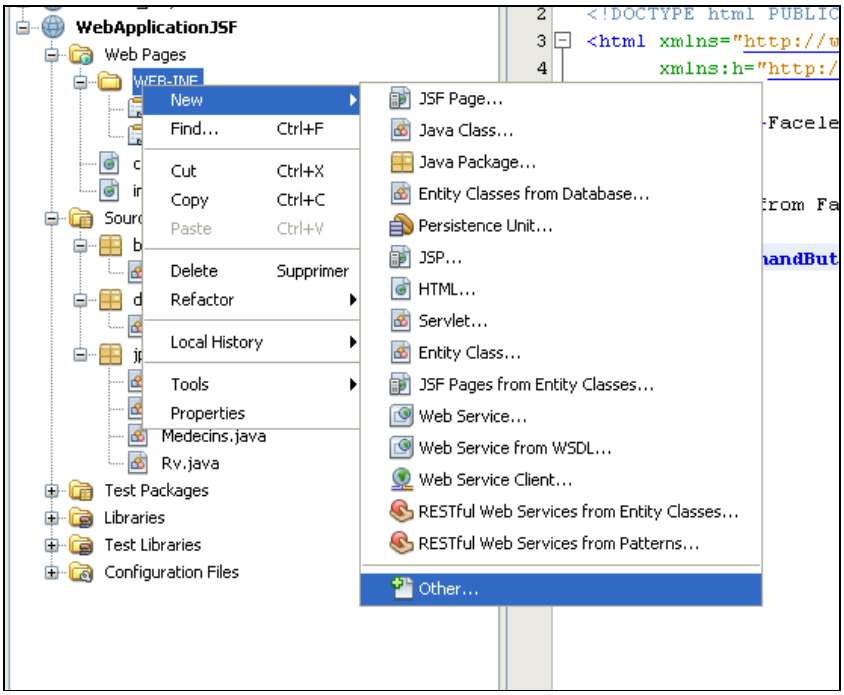
Le projet doit ressembler à ce qui suit :

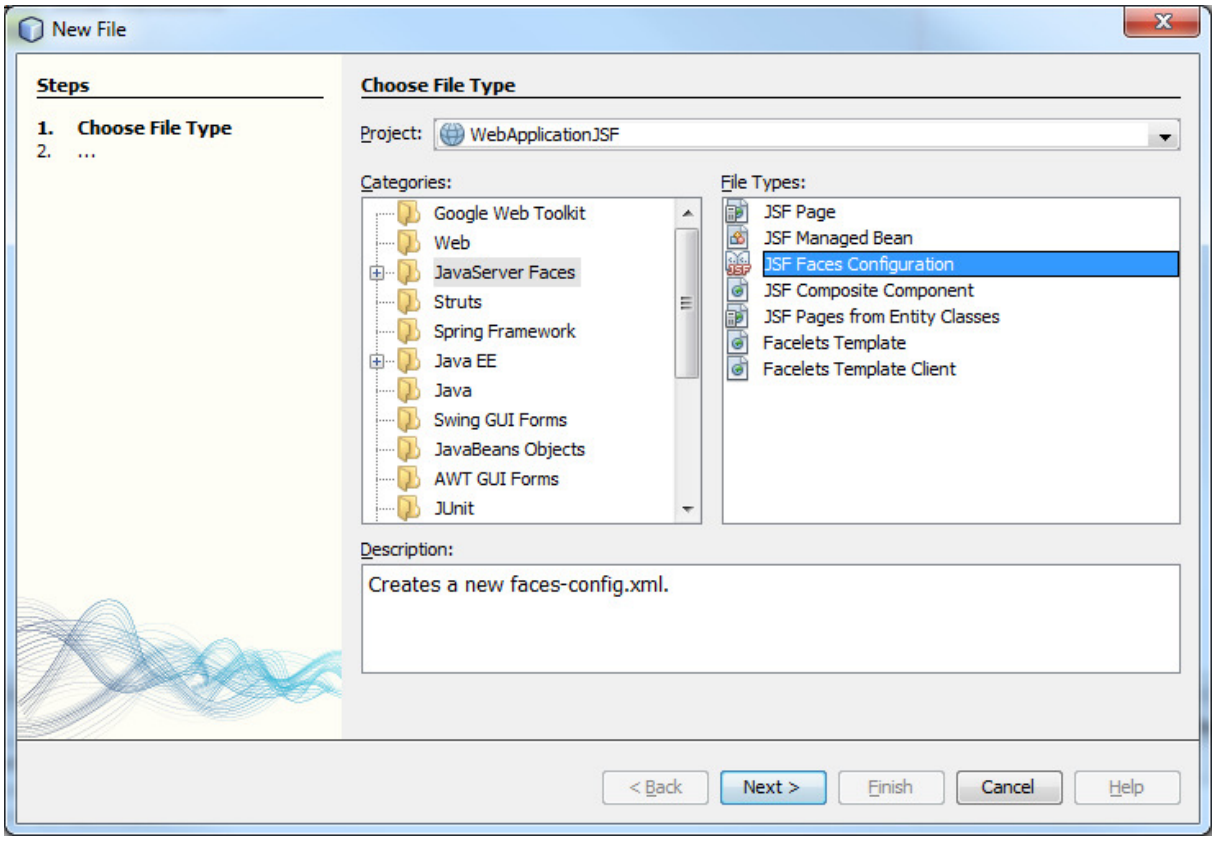


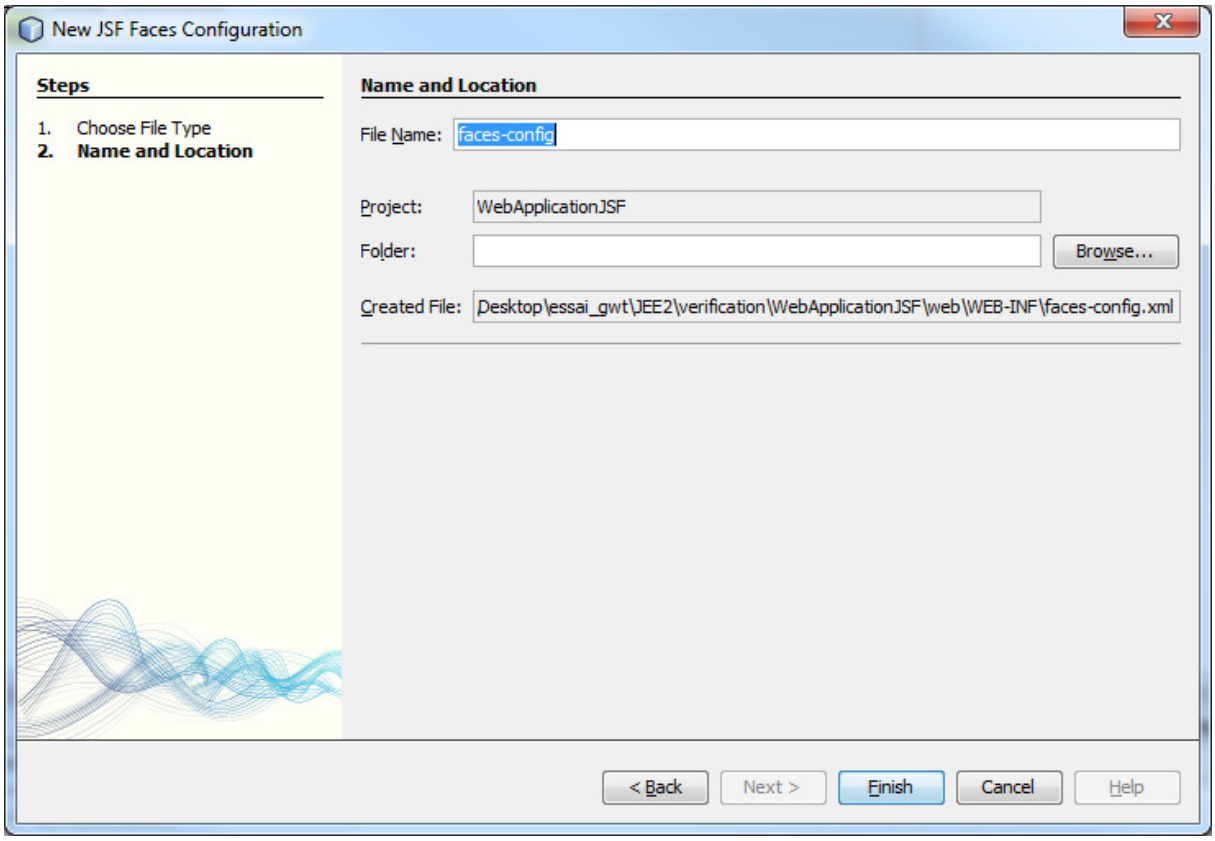
Cette page contiendra le résultat de notre requête à partir de la page index.xhtml pour avoir la liste de clients.

Pour ce faire, nous allons rajouter un fichier de configuration : faces-config.xml. Ce fichier contient les règles de navigation d'une page vers une autre et la définition des beans.

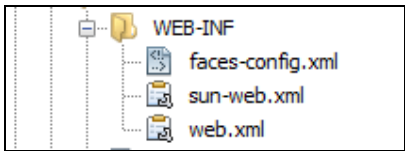
Sur le dossier Web-INF, clic droit, ajouter un nouveau fichier :







Le fichier est ajouté à WEB-INF.



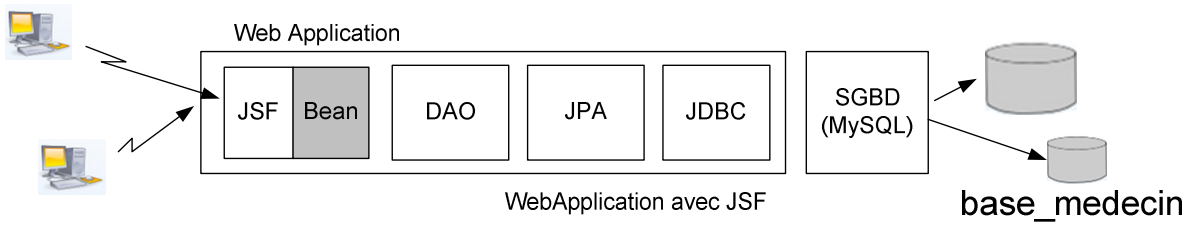
Nous allons maintenant définir les règles de navigation. Modifier le fichier comme suit :

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- ========== FULL CONFIGURATION FILE ================================== -->

<faces-config version="2.0"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
facesconfig_2_0.xsd">

    <navigation-rule>
        <from-view-id>/index.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>clients</from-outcome>
            <to-view-id>/clients.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>

</faces-config>
```

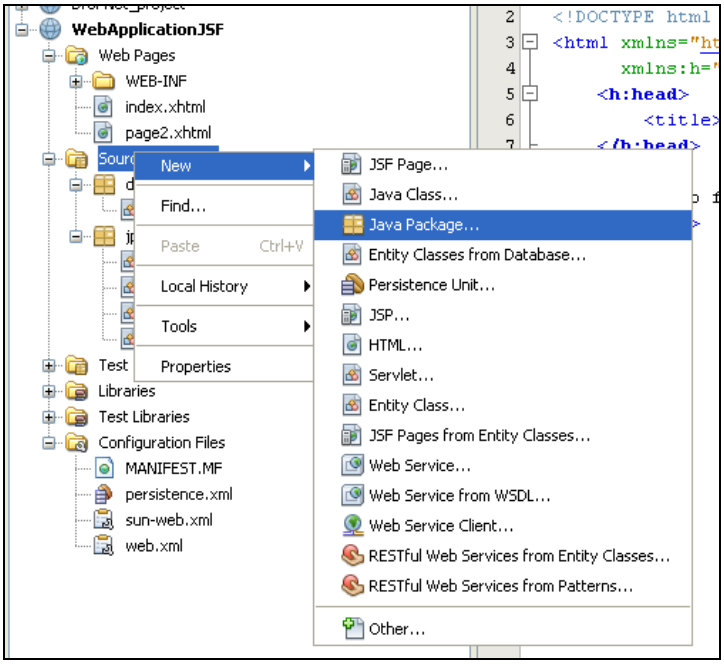
Cela signifie quand la page « index.xhtml » va émettre une chaine de caractère « clients » le serveur va lancer la page clients.xhtml.

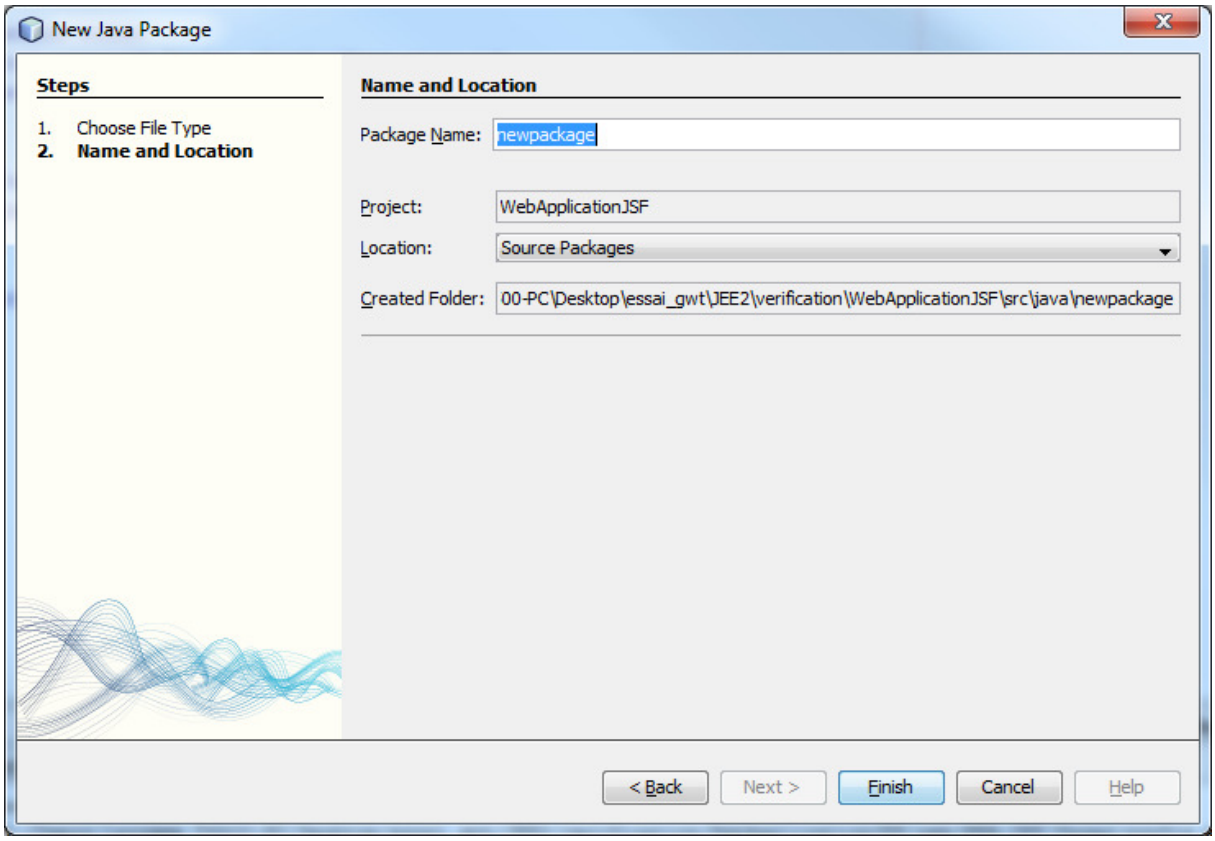
3.3. Création d'un package bean et d'une classe IndexClient

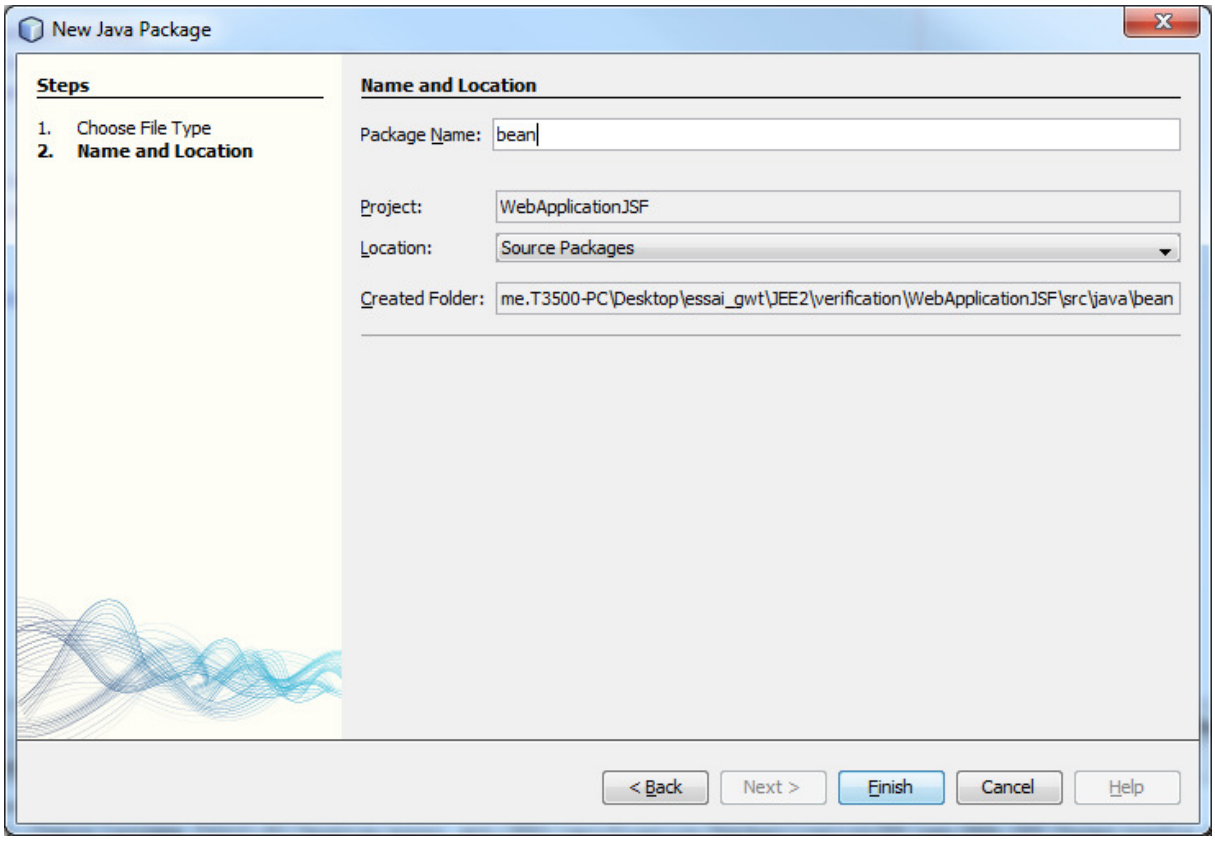
Nous allons créer un package nommé **Bean** dans lequel nous allons définir une classe nommée **IndexClient** qui va servir d'intermédiaire vers JPA.



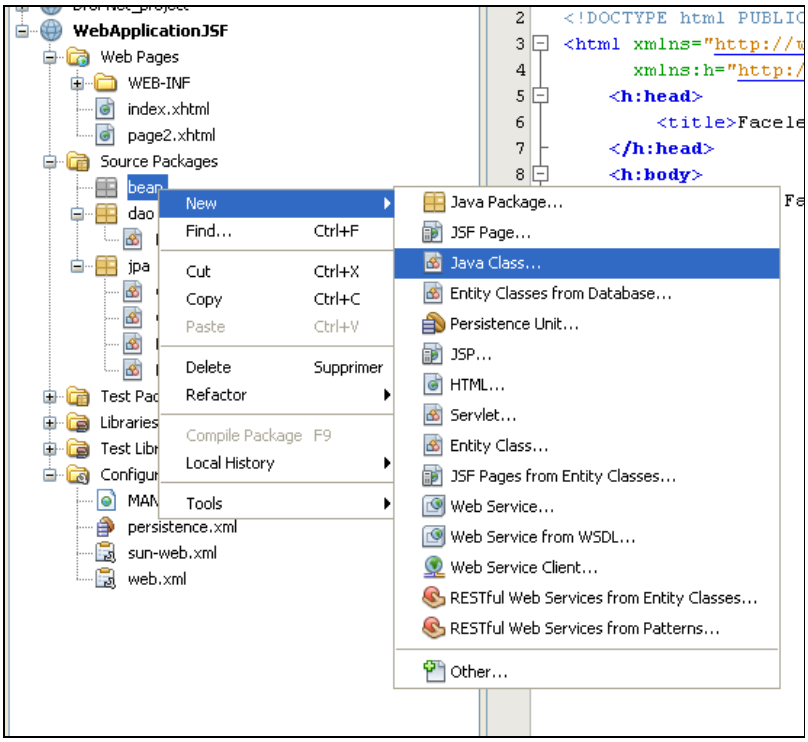
Faire un clic droit sur **Source Package** et choisir **Java Package**.

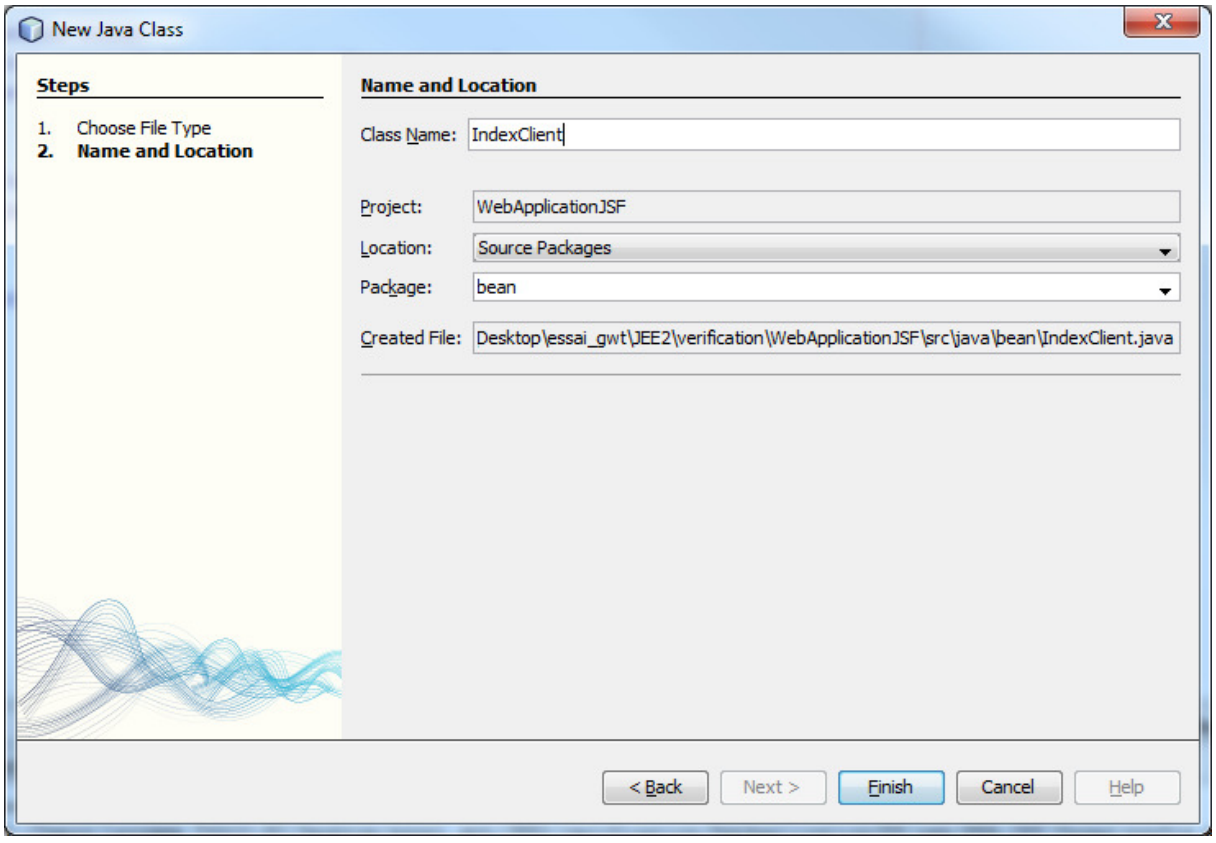



New Java Package
Steps
1. Choose File Type
2. Name and Location
Name and Location
Package Name: newpackage
Project: WebApplicationJSF
Location: Source Packages
Created Folder: 00-PC\Desktop\essai_gwt\JEE2\verification\WebApplicationJSF\src\java\newpackage
< Back
Next >
Finish
Cancel
Help


New Java Package
Steps
1. Choose File Type
2. Name and Location
Name and Location
Package Name: bean
Project: WebApplicationJSF
Location: Source Packages
Created Folder: me.T3500-PC\Desktop\essai_gwt\JEE2\verification\WebApplicationJSF\src\java\bean
< Back
Next >
Finish
Cancel
Help

Dans **Source Package**, faire un clic droit sur **bean** et choisir Java Class





Ouvrir le fichier IndexClient.java et insérer le code suivant :

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package bean;

import java.util.ArrayList;
import java.util.List;
import dao.DaoJpa;
import javax.faces.model.SelectItem;
import jpa.Clients;

public class IndexClient {

    private DaoJpa dao;
    private List<Clients> listClients;


    public void setListClients(List<Clients> clients) {
        this.listClients = clients;
    }

    public List<Clients> getListClients() {
        return listClients;
    }

    public String retrieveClientsfromDAO() {


        dao = new DaoJpa();

        dao.init(); //On initialise la couche de persistence

        System.out.println("--- affichage liste des clients ---");

        listClients = dao.getAllClients();

        dao.close();


        return "clients";
    }
}
```

Remarque

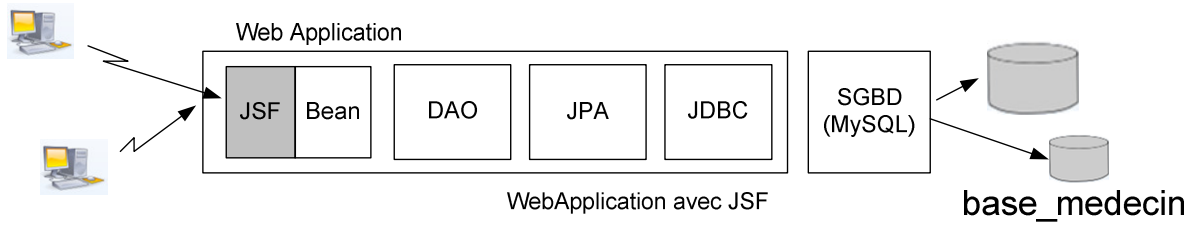
Notez que nous définissons une méthode **retrieveClientsfromDAO** () qui renvoie la liste des clients en utilisant les méthodes **init()** et bien **close()** précédemment définies.

Modifions le fichier faces-config.xml pour y définir notre bean.

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- ========== FULL CONFIGURATION FILE ================================== -->

<faces-config version="2.0"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
facesconfig_2_0.xsd">
    <navigation-rule>
        <from-view-id>/index.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>clients</from-outcome>
            <to-view-id>/clients.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>

    <managed-bean>
        <description>ClientBean</description>
        <managed-bean-name>IndexClient</managed-bean-name>
        <managed-bean-class>bean.IndexClient</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>


</faces-config>
```

3.4. Modification des pages index.xhtml et client.xhtml

Nous allons réaliser maintenant la partie JSF.



Modifiez la page **index.xhtml** comme suit pour y rajouter un bouton d'appel de la page clients.xhtml :

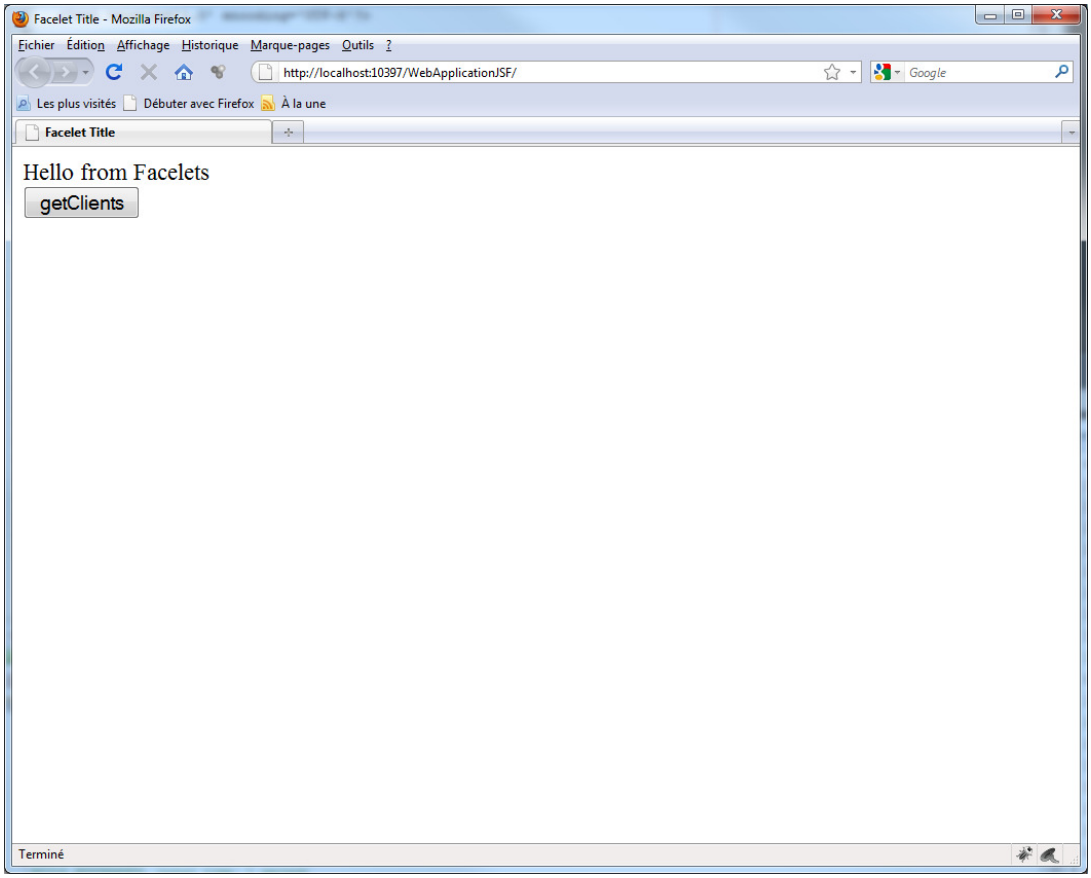
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>

    <h:body>
        Hello from Facelets ! This is the index page !

        <h:form>
            <br/>
            <h:commandButton action="#{IndexClient.retrieveClientsfromDao}" value="getClients"/>
        </h:form>

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

En réalité, ce bouton va appeler la méthode du bean **retrieveClientsfromDao()** qui va activer la règle de navigation qui entrainer le chargement de la nouvelle page **clients.xhtml** dont le nom a été donnée dans la partie **</navigation-rule>.**

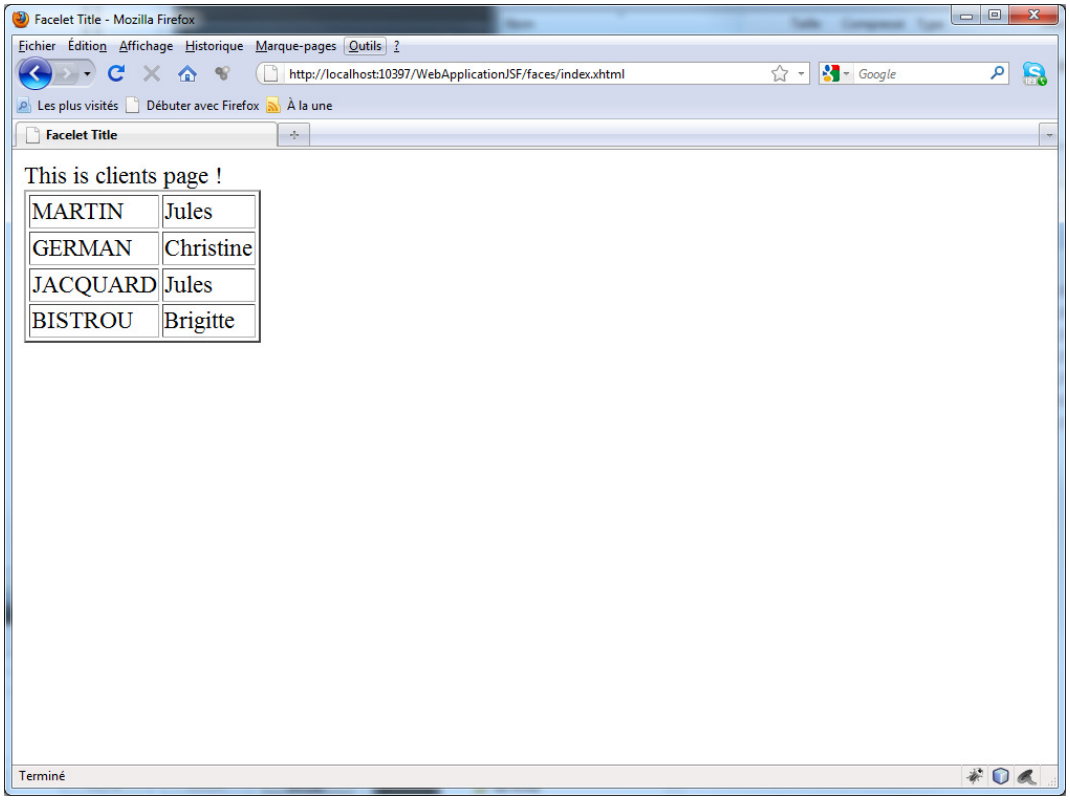
Modifier ensuite la page client.xhtml comme suit :

```
<?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">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        This is clients page !

        <h:dataTable var="c" value="#{IndexClient.listClients}" border="2">
            <h:column rowHeader="Nom">
                <h:outputText value="#{c.nom}"/>
            </h:column>
            <h:column rowHeader="Prénom">
                <h:outputText value="#{c.prenom}"/>
            </h:column>
        </h:dataTable>


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

Ce qui donne comme résultat d'exécution :



Après appui sur le bouton « getClients », on a la page clients.xhtml :


Facelet Title - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils ?
http://localhost:10397/WebApplicationJSF/faces/index.xhtml
Google
Les plus visités Débuter avec Firefox À la une
Facelet Title
This is clients page !
MARTIN Jules
GERMAN Christine
JACQUARD Jules
BISTROU Brigitte
Terminé

---------- fin ----------