

TP 02 - Construction de la structure du projet

Ce TP a pour but de vous faire créer la structure qui vous servira de base pour développer votre projet.

Vous allez créer un embryon d'application qui ne contient que la page de connexion et les pages de gestion des comptes.

Configuration d'Eclipse pour JSF

Pour rappel, voici quelques configurations qui peuvent être utiles lorsqu'on développe une application JSF.

Encodage UTF-8

- Menu Window > Preferences > Web
- > CSS Files > Encoding = ISO 10646/Unicode (UTF-8)
- > HTML Files :
 - Add this suffix (if not specified) = xhtml
 - Encoding = ISO 10646/Unicode (UTF-8)

Ajout de la vue Palette

- Ouvrez un fichier xhtml
- Menu Window > Show View > Other... > General > Palette
- Si la vue Palette a été mise dans le volet du bas, on peut la déplacer pour la mettre dans le volet de droite d'Eclipse, à côté de l'onglet "Outline"
- Dans la barre de menu de la vue Palette, clic sur l'icône "Show/Hide"
 - ☐ JBoss
 - ☒ JSF
 - ☒ Core
 - ☒ Facelets
 - ☒ HTML

Construction de l'application Projet

Création du projet global

- Menu " File > New > Maven Project"

- Bouton "Next >".
- Recherchez l'archétype : **wildfly-jakartaee-ear-archetype**
- Vers le bas, décochez la case : ☐ Show the last version of Archetype only
- Sélectionnez la version **26.1.0.Final**
car c'est celle qui correspond à la version de WildFly qui est installée.
Puis, bouton "Next >".
- GroupId : **fr.3il.jee**
ArtifactId : **projet**
Package : **projet**
Bouton "Finish"

Cette opération a dû créer un projet racine nommé **projet**, qui contient lui-même 3 projets imbriqués nommés :

- **projet-ear**
- **projet-ejb**
- **projet-web**

Il est possible que des erreurs soient signalées par Eclipse. Celles-ci vont disparaître après un certain temps.

Vous reconnaissez une organisation similaire à celle de l'application Exemple.

Création de projet-commun

Par rapport à l'architecture que nous utilisons habituellement, il manque le projet **projet-commun** qui contiendra les éléments partagés par les différentes couches.

- Sélectionnez le projet **projet**
- Menu "File > New > Maven Project"
- Cochez : ☒ **Create a simple project**
Décochez : ☐ **Use Default Workspace location**
Dans le champ "Location", il est probable que le chemin se termine par **\projet**
Modifiez-le pour qu'il se termine par **\projet\projet-commun**
Bouton "Next >"
- Dans la section "Parent Project", bouton "Browse"
Recherchez le projet **projet**.
Bouton "OK".
- Dans la section "Artifact" :
GroupId : **fr.3il.jee**
ArtifactId : **projet-commun**
Bouton "Finish".

Un sous-projet **projet-commun** a été créé au sein du projet **projet**.

Quelques modifications sont nécessaires pour qu'il soit conforme à nos attentes.

Fichier POM de projet-commun

- Dans le projet **projet-commun**, ouvrez le fichier **POM**.
- Les lignes marquées en jaunes, sont redondantes avec les informations contenues dans le fichier **POM** parent.
Supprimez ces lignes (il y a 2 lignes à supprimer).
- Enregistrez la modification et fermez le fichier.

Fichier POM parent

Ajout du module projet-commun

Le fichier **POM** parent se trouve dans le projet **projet**.

Attention ! dans le volet de gauche d'Eclipse, ce fichier apparaît tout à la fin, c'est-à-dire après les sous-projets.

- Ouvrez le fichier **POM** parent.
- Choisissez l'onglet "Overview".
- En bas et à gauche dans la vue centrale, repérez la section "Modules"
Dans le vocabulaire de Maven, les modules sont les différents sous-projets qui constituent l'application et dépendent du fichier **POM** parent.
- Bouton "Add..."
Cochez ☒ **projet-commun**
Puis bouton "OK".

Configuration du processeur d'annotations

Pour que MapStruct soit correctement géré par Maven et par Eclipse, il y a un petit travail de configuration supplémentaire à faire.

- Dans le fichier **POM** parent, choisissez l'onglet "pom.xml"
- Recherchez le plugin nommé **maven-compiler-plugin**.
(Probablement vers la ligne 133)
- Ce plugin est doté d'une section :

```
<configuration>  
  <!-- put your configurations here -->  
</configuration>
```

- Dans cette section, supprimez la ligne du milieu et mettez à sa place le code suivant :

```
<annotationProcessorPaths>
  <path>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct-processor</artifactId>
    <version>1.5.3.Final</version>
  </path>
</annotationProcessorPaths>
```

MapStruct est composé de 2 bibliothèques : la bibliothèque MapStruct principale (**mapstruct** tout court) et le processeur d'annotations (**mapstruct-processor**).

Ici, on a configuré uniquement le processeur. La bibliothèque principale sera ajoutée plus loin.

- Enregistrez les modifications et fermez le fichier **POM**.
- Eclipse signale des erreurs, car vous venez de faire une modification importante du fichier **POM** qui nécessite la mise à jour des projets.
⇒ **Alt+F5**, sélectionnez tous les projets, puis "OK".

Toutes les erreurs doivent avoir disparu et de nouvelles branches **target/generated** ont dû apparaître dans certains projets.

Ces branches sont destinées à contenir le code généré par MapStruct.

Récupération du code-source

Vous allez à présent implanter le code-source dans les projets **projet-commun**, **projet-ejb** et **projet-web**.

Projet projet-commun

- Ouvrez le fichier **tp-02-projet-commun.zip**. Celui-ci contient un dossier **src**.
- Faites le nécessaire pour coller une copie de ce dossier **src**, directement dans le dossier **projet-commun**.
Si cette copie a eu lieu à l'extérieur d'Eclipse, pensez à faire "Refresh" sur le projet **projet-commun** quand vous revenez dans Eclipse.

La branche **src/main/java** doit contenir 3 packages.

Eclipse ne doit signaler aucune erreur.

Projet projet-ejb

- Ouvrez l'archive **tp-02-projet-ejb.zip** et faites le nécessaire pour coller une copie du dossier **src** qu'elle contient, à la racine du projet **projet-ejb**.

Après la copie, la branche **src/main/java** contient 5 packages dont certains sont en erreur, ce qui est normal.

Les erreurs proviennent du fait que certaines classes du projet **projet-ejb** ont besoin des classes contenues dans le projet **projet-commun**, ainsi que de la bibliothèque MapStruct.

- Ouvrez le fichier **POM** du projet **projet-ejb**.
Attention à ne pas vous tromper de fichier **POM** ! Il est facile de les confondre.
Pour être sûr de ne pas faire d'erreur, il suffit de choisir l'onglet "Overview". pour vérifier qu'on est bien dans le bon fichier **POM**.
- Choisissez l'onglet "Dependencies".
- Dans la liste "Dependencies" (à gauche), vous allez ajouter les éléments suivants.
 - **projet-commun**
 - **mapstruct**
- Pour pouvoir facilement changer le n° de version de ces dépendances, vous allez leur donner le statut "managed" et stocker leur définition de référence dans le fichier **POM** parent.
 - Actionnez le bouton "Manage...".
 - À gauche, sélectionnez les 2 dépendances **projet-commun** et **mapstruct**.
 - À droite, sélectionnez l'item qui représente le projet parent (**projet tout court**).
 - Actionnez le bouton "OK".
- Une fois que c'est fait, vérifiez que l'indication **managed** apparaît bien à côté de chacune des deux dépendances.

Lorsque les modifications ont été enregistrées, les erreurs doivent avoir disparu dans la branche **src/main/java** du projet **projet-ejb**.

Projet projet-web

- Ouvrez l'archive **tp-02-projet-web.zip** et faites le nécessaire pour coller une copie du dossier **src** qu'elle contient, à la racine du projet **projet-web**.
Après la copie, la branche **src/main/java** contient 7 packages dont certains sont en erreur,.
- Ouvrez le fichier **POM** du projet **projet-web**.
- Choisissez l'onglet "Dependencies".
Observez que toutes les dépendances ont une portée qui est soit **provided**, soit **test**.
- Supprimez la dépendance suivante :
 - **projet-ejb**
Explication : La dépendance au projet **projet-ejb** n'est pas utile car, dans notre architecture, nous utilisons le projet **projet-commun** pour stocker les composants communs aux différentes couches.
- Ajoutez les dépendances suivantes. Elles doivent être de portée **provided**.
 - **projet-commun**
 - **mapstruct**
 - **validation-api**
- Vérifiez que l'indication [**provided**] apparaît bien à côté de chacune des dépendances que vous venez d'ajouter.
- Faites le nécessaire pour donner le statut "Managed" à la bibliothèque **validation-api**.
- Lorsque les modifications du fichier **POM** sont enregistrées. Toutes les erreurs doivent disparaître de la branche **src/main/java**.

Fichier POM parent

- Ouvrez le fichier **POM** parent et choisissez l'onglet "Dependencies".
- Dans la liste "Dependency Management", sélectionnez **projet-ejb**, puis actionnez le bouton "Properties".
- Observez le contenu des champs **GroupId** et **Version**. Ils contiennent les valeurs :
 `${project.groupId}`
 `${project.version}`
- Ensuite, revenez à la liste "Dependency Management", sélectionnez **projet-commun** et actionnez le bouton "Properties".
- Faites le nécessaire pour que les champs **GroupId** et **Version** contiennent les mêmes valeurs que pour **projet-ejb**.
Ainsi, il sera très facile de faire évoluer l'application et notamment de changer de n° de version de l'application.

Configuration des serveurs

Mise en place de la base de données

Importation des scripts SQL

- Ouvrez le fichier **tp-02-scripts-sql.zip**. Celui-ci contient 2 dossiers **scripts-sql** et **test-git**.
- Faites le nécessaire pour coller une copie de ces dossiers, directement dans le dossier **projet**.

Exécution des scripts SQL

PostgreSQL doit être démarré.


- À gauche, dans le dossier **scripts-sql**, sélectionnez le fichier **execute-sql.xml** et, par drag-and-drop, venez le déposer dans la vue "Ant".
- Dans la vue "Ant", déployez **projet-sql**;
- Double-cliquez sur la cible **0-user** pour l'exécuter.
Cela crée le compte **projet** qui servira à se connecter à la base.
- Puis, exécutez la cible **9-tout** pour créer les tables et y insérer des données.

Création de la connexion dans Eclipse

- En bas, dans la vue "Data Source Explorer", clic-droit sur "Database Connections", puis "New...".
- Appliquez les paramètres suivants :

Connection Profile Type	PostgreSQL
Name	Projet

Appuyez sur le bouton "Next"

- Dans la page "Specify a Driver and Connection Details, si la liste déroulante "Drivers :" est vide, cliquez sur l'icône  "New Driver Definition" qui se trouve à sa droite. Faites alors les opérations suivantes :

- Au centre, sélectionnez la ligne "PostgreSQL JDBC Driver".
- En haut, sélectionnez l'onglet "JAR List".
 - Cliquez sur **postgresql-x.x-xxx.jdbc2.jar**, puis bouton "Edit JAR/Zip..."
 - Naviguez pour aller dans le dossier **C:\dev23\jdbc-drivers\postgresql**
 - Sélectionnez le fichier **postgresql-x.x.x.jar** qu'il contient.
 - Puis, bouton "Ouvrir" et bouton "OK"

- De retour dans la page "Specify a Driver and Connection Details", indiquez les paramètres suivants :

Driver	PostgreSQL JDBC Driver
Database	projet
URL	Remplacer database par postgres (postgres est le nom de la base de données).
User name	projet
Password	projet
Save password	<input checked="" type="checkbox"/> (coché)

- Appuyez sur le bouton "Test Connection"
- Puis, sur le bouton "Finish"

Association du projet projet-ejb

- Dans le volet de gauche, clic-droit sur le projet **projet-ejb**, puis, "Properties > JPA".
- Renseignez les champs suivants :
 - *Platform* : **Generic 2.2**
 - *JPA Implementation* : **Disable Library Configuration**
 - *Connection* : La connexion que vous venez de créer.
- Validez pour revenir à l'espace de travail d'Eclipse.


Ainsi, lorsque vous travaillerez sur les entités JPA dans le projet **projet-ejb**, les assistants d'Eclipse pourront se connecter à la base pour vous faire bénéficier d'une aide supplémentaire.

Configuration de WildFly

Configuration de la DataSource

WildFly doit être démarré.

- Ouvrez la console d'administration web de WildFly dans un navigateur.

- Menu "Configuration > Subsystems > Datasources & Drivers > Datasources" .
Puis, exécutez l'action  "Add Datasource".
- Sélectionnez "PostgreSQL" et appuyez sur "Next >".
- Dans le panneau "Attributes", indiquez les valeurs suivantes :
 - Name : **Projet**
 - JNDI Name : **java:/ds/projet**Puis appuyez sur "Next >".
- Dans le panneau "JDBC Driver", pour le champ "Driver Name", sélectionnez le pilote pour PostgreSQL.
Puis appuyez sur "Next >".
- Dans le panneau "Connection", indiquez les valeurs suivantes :
 - Connection URL : **jdbc:postgresql://localhost:5432/postgres** ← *attention modif à faire*
 - User Name : **projet**
 - Password : **projet**Le champ "Security Domain" doit rester vide.
- Passez au panneau suivant en appuyant sur "Next >" et testez la connexion.
- Terminez en appuyant sur les boutons "Finish" et "Close".
- En haut et à droite, cliquez sur "Reload required" afin que WildFly actualise sa configuration.

Exécution de l'application web

Déploiement sur le serveur Wildfly

- Dans Eclipse, allez dans la vue "Servers".
- Clic-droit sur WildFly, puis "Add and Remove...".
- Sélectionnez **projet-ear**, puis actionnez le bouton "Add >", puis le bouton "Finish".
Normalement, il ne devrait pas y avoir d'erreurs.

Exécution de l'application

- Dans le volet de gauche d'Eclipse, faites un clic-droit sur le projet **projet-web**, puis "Run As > Run on Server...".
- Dépliez la branche **localhost** et sélectionnez WildFly.
- En bas, cochez la case ☒ **Always use this server when running this project**
- Actionnez le bouton "Finish".
- Vous pouvez vous connecter avec les comptes habituels :
 - geek / geek**
 - chef / chef**
 - job / job**

L'application doit être opérationnelle.

Seules les pages qui permettent de gérer les comptes sont disponibles.

Vous pouvez, à présent, démarrer le développement de votre projet.