

## TP 01 - Application Exemple

Afin de vous encourager à adopter de bonnes pratiques, une application complète vous est fournie à titre d'exemple. Vous pourrez ainsi vous en inspirer ou éventuellement y trouver des solutions techniques aux problèmes que vous rencontrerez.

### Création du workspace du projet

#### Dossier pour le workspace

- Pour travailler sur le projet, vous devez utiliser votre ordinateur personnel. Cela vous permettra de travailler chez vous ou dans n'importe quelle salle de l'école pendant les séances en autonomie.
- Créez un nouveau dossier destiné à être utilisé en tant que workspace pour votre projet. Par exemple : **C:\Projet-JEE**
- Démarrez Eclipse JEE et ouvrez le nouveau workspace que vous venez de créer.

#### Configuration d'Eclipse


##### Affichage de la vue "Maven Repositories"

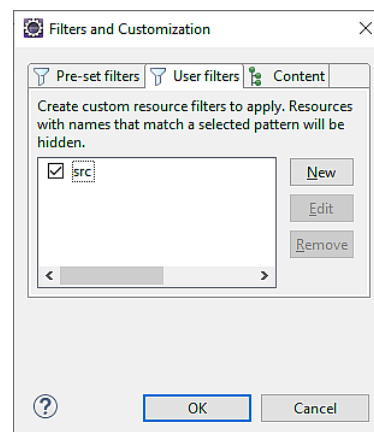
- Menu "Window > Show View > Other... > Maven > Maven Repositories".
- Allez dans la vue "Maven Repositories".
- Dépliez la branche "Local Repositories".
- Clic-droit sur **Local Repository (C:\TEMP\maven-repository)**, puis "Rebuild Index".  
S'il ne se passe rien - c'est-à-dire si l'élément **Local Repository** reste vide -, cela signifie qu'il est nécessaire de redémarrer Eclipse. Dans ce cas :
  - Menu "File > Restart".
  - Puis de nouveau, clic-droit sur **Local Repository** et "Rebuild Index".

##### Masquage du dossier src

Pour éviter les confusions, il vaut mieux ne plus faire apparaître le dossier **src** dans la vue "Project Explorer".

- Dans le volet de gauche d'Eclipse, ce doit être la vue "Project Explorer" qui est affichée.

- À droite du titre "Project Explorer", cliquez sur l'icône "View Menu"  .  
Puis, sélectionnez "Filters and Customization...".
- Choisissez l'onglet "User filters" et actionnez le bouton "New".
- Au centre, remplacez le texte : **\*.something**  
par : **src**
- Et cochez la case qui se trouve à côté.
- Validez pour revenir à l'espace de travail d'Eclipse.



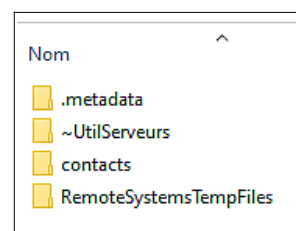
## Importation de l'application exemple

### Récupération du code-source

- Récupérez l'archive **tp-01-appli-exemple.zip** et enregistrez-le à un endroit où vous pourrez facilement y avoir accès.
- Décompressez le contenu de l'archive **tp-01-appli-exemple.zip** de façon à le placer directement dans le dossier qui vous sert de workspace Eclipse.

Si vous utilisez l'explorateur de fichiers de Windows pour observer le contenu du workspace, vous devez constater qu'il contient 4 dossiers :

**.metadata**  
**~UtilServeurs**  
**contacts**  
**RemoteSystemTempFiles**



- Dans Eclipse, vous allez importer les projets qui viennent d'être ajoutés au workspace
  - Menu "File > Import... > General > Existing Projects into Workspace", puis bouton "Next >".
  - Choisissez : ☒ **Select root directory**  
Actionnez le bouton "Browse..." situé à droite et, immédiatement, appuyez sur le bouton "Sélectionner un dossier".
  - Dans la section "Options", cochez la case ☒ **Search for nested projects**
  - Vérifiez que tous les projets sont sélectionnés, sauf le dernier qui est grisé.
  - Actionnez le bouton "Finish".

Lorsque l'importation est terminée, vous devez voir apparaître les projets dans le volet de gauche d'Eclipse. Il se peut que des erreurs apparaissent temporairement, mais, à la fin, elles doivent toutes avoir disparu. Cela peut prendre un peu de temps.: surveillez le témoin d'exécution dans la barre d'état d'Eclipse (en bas et à droite).

Si, lorsque l'importation est terminée, il y a toujours des erreurs, utilisez le menu "Project > Clean" pour les faire disparaître.

Si nécessaire, vous pouvez aussi refaire la synchronisation de la configuration d'Eclipse avec celle de Maven :

**Alt+F5** ou clic-droit sur le projet contacts et "Maven > Update Project..."

## Observation de la structure de l'application

- Dépliez le projet **contacts**.  
Celui-ci contient 4 projets imbriqués nommés :
  - **contacts-commun**
  - **contacts-ear**
  - **contacts-ejb**
  - **contacts-web**

Cette disposition est différente de celle que vous avez utilisée au cours des TPs de la matière Architecture Java EE.

Dans les TPs, le projet **contacts** contenait uniquement le fichier **POM** parent. Il était placé à côté des autres projets correspondant aux différentes couches de l'application.

Ici, on a une structure qui est beaucoup plus conforme à Maven. Le projet **contacts** contient les sous-projets qui constituent l'application.

## Mise en place des serveurs

### Initialisation des serveurs

- Démarrez UtilServeurs.
- Sélectionnez PostgreSQL
  - Actionnez le bouton "Create" pour créer une base vide.
  - Actionnez le bouton "Start" pour démarrer PostgreSQL.
- Sélectionnez WildFly
  - Actionnez le bouton "Create" pour créer un dossier de base vide.
  - Ne démarrez pas WildFly.  
WildFly doit toujours être démarré à partir d'Eclipse et non avec UtilServeurs.
  - Actionnez le bouton "Info"
  - Repérez l'adresse du dossier de base de WildFly. Normalement, son chemin est du type : C:\TEMP\projet-jeelwildfly-26

## Mise en place de la base de données

### Ajout de la vue Ant

- Menu "Window > Show View > Other... > Ant > Ant".
- La vue "Ant" a probablement été ajoutée dans le volet inférieur d'Eclipse, ce qui n'est pas pratique.

- Par drag-and-drop, déplacer la vue "Ant" pour la placer dans le volet de droite, juste à côté de l'onglet "Outline".

## Exécution des scripts SQL

- À gauche, dans le projet **contacts**, déployez le dossier **scripts-sql**,
- Le dossier **scripts-sql** contient les fichiers suivants :
  - **0-user.sql** : script qui permet de créer l'utilisateur **contacts** qui sert pour établir la connexion à la base.
  - **1-tables.sql** : script de création des tables.
  - **2-procedures.sql** : script pour créer des procédures stockées et des triggers.
  - **3-data.sql** : script d'insertion de données dans les tables.
- Sélectionnez le fichier **execute-sql.xml** et, par drag-and-drop, venez le déposer dans la vue "Ant".
- Dans la vue "Ant", déployez **contacts-sql**;
- Double-cliquez sur la cible **0-user** pour l'exécuter.  
Cela crée le compte **contacts** qui servira à se connecter à la base.
- Puis, exécutez la cible **9-tout** pour créer les tables et y insérer des données.

## Mise en route de WildFly

### Configuration d'Eclipse

- Dans le volet du bas d'Eclipse, choisissez l'onglet "Servers", puis clic-droit et "New > Server".
- Dans le panneau "Define New Server" :
  - Server type : **JBoss Community > WildFly 24+**
  - Bouton "Next >"
- Dans le panneau "Create a new Server Adapter" :
  - Rien à modifier
  - Bouton "Next >"
- Dans le panneau "JBoss Runtime" :
  - Home Directory = **C:\dev23\wildfly-26**
  - Runtime JRE : ☉ **Alternate JRE**
  - Server base directory : indiquez l'emplacement du dossier de base de WildFly (du type **C:\TEMP\projet-jeelwildfly-26**).
  - Bouton "Finish".


Votre serveur WildFly doit apparaître dans la vue "Servers".

### Démarrage et ouverture de console d'administration web



- Démarrez le serveur WildFly :
  - Clic-droit sur le serveur, puis "Start"

- Attendre que le serveur ait fini de démarrer.
- Pour ouvrir la console d'administration web de WildFly dans un navigateur :
  - Revenir à l'onglet "Servers".
  - WildFly doit être marqué **[Started, Synchronized]** pour indiquer qu'il est démarré et à jour.
  - Clic-droit sur WildFly, puis "Show In > Web Management Console".
  - Données de connexion :
    - User name : **admin**
    - Password : **adminadmin**

## Ajout du pilote JDBC

- Allez dans la section Deployments et cliquez sur l'icône "Add" .  
Puis, sélectionnez "Upload Deployment".
- Faites le nécessaire pour ajouter le fichier **postgresql-xx.x.x.jar** qui se trouve dans le dossier **C:\dev23\jdbc-drivers\postgresql**. Puis, appuyez sur le bouton "Next >".
- Dans le champ "Runtime Name", indiquez **postgresql** (tout court).  
Pour le champ "Enabled", choisissez **ON**.  
Puis appuyez sur les boutons "Finish" et "Close".

## Configuration de la DataSource

- Toujours dans la Console d'Administration Web de WildFly, utilisez le menu "Configuration > Subsystems > Datasources & Drivers > Datasources" et exécutez l'action  "Add Datasource".
- Sélectionnez  **PostgreSQL** et appuyez sur "Next >".
- Dans le panneau "Attributes", indiquez les valeurs suivantes :
  - Name : **Contacts**
  - JNDI Name : **java:/ds/contacts**Puis appuyez sur "Next >".
- Dans le panneau "JDBC Driver", pour le champ "Driver Name", sélectionnez le pilote que vous avez ajouté précédemment.  
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 : **contacts**
  - Password : **contacts**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".

Une nouvelle DataSource nommée **Contacts** doit apparaître dans la liste. En cliquant sur la flèche à côté du bouton "View", vous pouvez exécuter l'action "Test Connection". Cela vous sera peut-être utile par la suite si vous rencontrez des problèmes.

- 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".
- Pour déployer l'application sur le serveur, faites un clic-droit sur WildFly, puis "Add and Remove...".
- Sélectionnez **contacts-ear**, puis actionnez le bouton "Add >", et 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 **contacts-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".
- Puis, bouton "OK".
- Ici, il faut parfois faire preuve d'un peu de patience, avant de voir l'application s'afficher dans le navigateur web.
- L'application web utilise les mêmes identifiants que l'application JavaFX Contacts puisqu'il s'agit de la même base de données :
  - geek / geek
  - chef / chef
  - job / job

L'application doit être opérationnelle.

Il s'agit d'une application JSF qui utilise les EJBs et JPA.

### Mais où sont les pages XHTML ?

En dépliant, le projet **contacts-web**, il est facile d'y trouver les classes Java., car elles se trouvent dans la branche **Java Resources**.

Mais, on peut se poser la question : où sont les pages **XHTML** et comment fait-on pour les modifier ou en ajouter de nouvelles ?

- Dans le projet **contacts-web**, dépliez la branche **Deployed Resources**.
- Dans cette branche, vous trouvez le dossier **webapp** qui contient toutes les pages **XHTML**, les fichiers **CSS**, les images, les fichiers de configuration de JSF, etc.