|  |
| --- |
| CCINCA |
| TP Java J2EE |
| DAO EJB JPA |

|  |
| --- |
| AMSELEM Jonathan  08/01/2020 |

Table des matières

[Exercice 5 2](#_Toc29454332)

[L’architecture 2](#_Toc29454333)

[Le patron de conception DAO 2](#_Toc29454334)

[Le Patern singleton 2](#_Toc29454335)

[Dao Factory 3](#_Toc29454336)

[Daos 3](#_Toc29454337)

[Exercice 6 3](#_Toc29454338)

[Gestion du fichier properties 3](#_Toc29454339)

[Mot de passe encrypté 4](#_Toc29454340)

[Pool de connexion 4](#_Toc29454341)

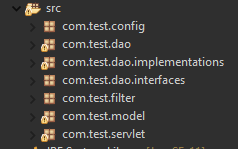
[Exercice 7 5](#_Toc29454342)

# Exercice 5

Dans cet exercice nous allons mettre en place le patron de conception dao dans notre application. Créer une connexion unique dans une variable ayant pour portée l’application. Ajouter un pool de connexion en récupérant les information depuis un fichier « config.properties ». Pour cela nous devrons :

## L’architecture

Nous allons commencer par ajouter des packages. Voici la nouvelle architecture de notre projet

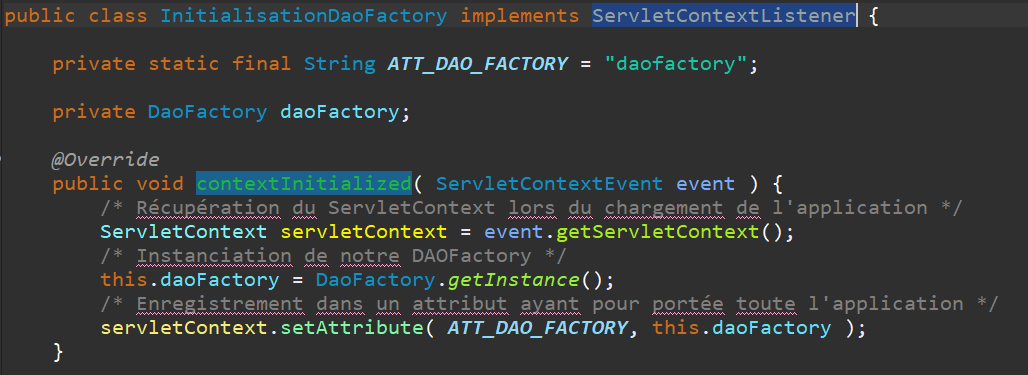


## Le patron de conception DAO

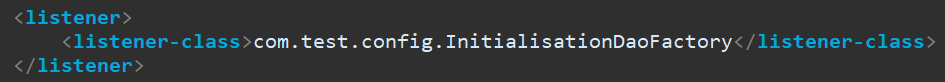
### Le Patern singleton

Créez une connexion unique dans une variable ayant pour portée l’application

1. Dans le package com.test.config, créez une classe InitialisationDaoFactory qui implémente l’interface ServletContextListener et implémenté la méthode ConextInitialized



1. Ajoutez la classe InitialisationDaoFactory en tant que listener dans le web.xml afin de spécifier au container de charger la classe au démarrage de l’application



### Dao Factory

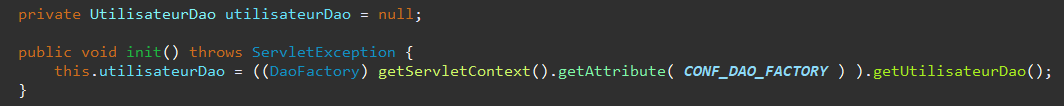
La dao Factory donne accès aux DAO. Elle contient l’instance de connexion que les dao utiliseront pour effectuer le CRUD.

1. Dans le package com.test.dao créez une class DaoFactory et implémentez la méthode getInstance qui renvoie une DaoFactory en faisant appel au contructeur DaoFactory(Connection con). Pour crééer la connexion vous déplacerez le code de la création de la connexion de votre servlet Connexion dans la methode getInstance.
2. Ajoutez les méthodes qui getUtilisateurDao (DaoFactory f) et getVehiculeDao(DaoFactory f)

### Daos

Les Daos (Data Access Object) sont les classes qui vont assurer le lien entre la base et le modèle objet de l’application. Vous y trouverez des méthodes telles que trouver, supprimer ect.

1. Dans le package com.test.dao.interfaces ajoutez les interfaces UtilisateurDao et VehiculeDao
2. Dans le package com.test.dao.implementation ajoutez les classes UtilisateurDaoImpl et VehiculeDaoImpl qui implémentent les interfaces UtilisateurDao et VehiculeDao.
3. Dans vos servlets chargés vos dao dans la méthode init () qui s’execute à la création de la servlet en faisant appel à la variable de porté application « daoFactory »



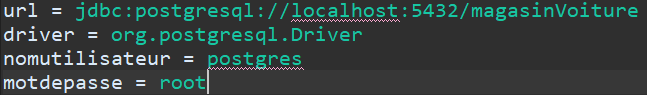
# Exercice 6

Continuons à étoffer notre application afin de la rendre plus sûre et adapté au milieu industriel. Pour cela ajoutez :

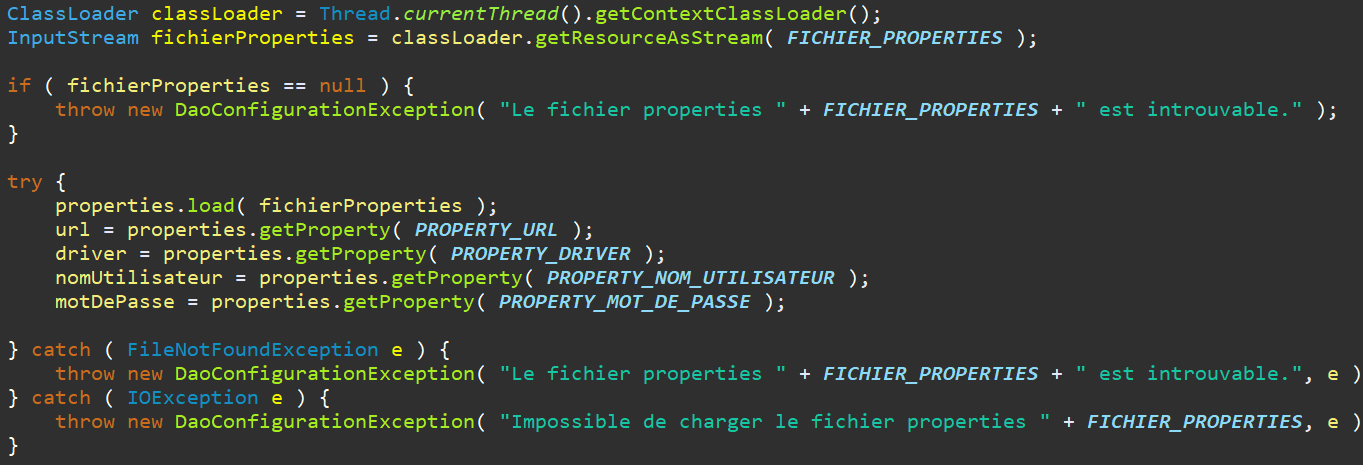
1. L’utilisation d’un mot de passe encrypté
2. Un formulaire d'ajout d'utilisateur
3. Un fichier config.properties pour stocker les données de connexion
4. Un pool de connexion

## Gestion du fichier properties

1. Dans le package com.test.config ajoutez un fichier config.properties qui contiendra nos données de connexion.



1. Chargez les paramètres de connexion depuis votre servlet en utilisant la gestion de Properties.



## Mot de passe encrypté

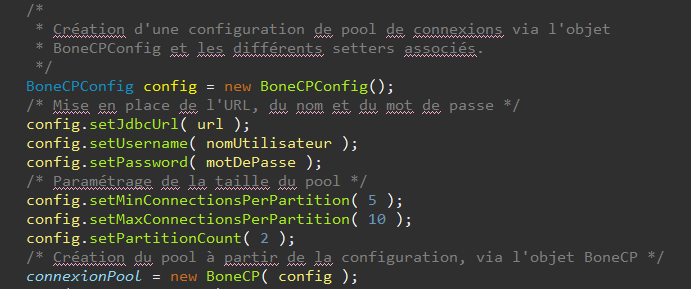
Il est important que les mots de passes soient encryptés afin que si une personne malveillante parvient à entrer dans votre base de données celle-ci ne puisse pas les lire.

Dans votre Servlet de création utilisateur nous allons encrypter notre mot de passe avant d’insérer le nouvel utilisateur en base de données. Pour cela nous allons :

1. Ajouter la librairie jasypt.jar dans le dossier WEB-INF/
2. Ajouter la librairie dans le ClassPath.
3. Créer une classe Utilitaire dans laquelle nous allons créer :
   1. Une méthode ConfigurablePasswordEncryptor getEncryptor() pour récupérer l’encryptor
   2. Une methode String encrypt() pour encrypter le mot de passe
   3. Nous utiliserons l’algorithme SHA-256.
4. Dans la servlet d’ajout Utilisateur nous encryptons le mot de passe.
5. Dans la servlet de connexion nous comparons le mot de passe du formulaire avec l’emprunte du mot de passe encrypté stocké en base de donnée : Utilitaire.getEncryptor().checkPassword(user.getPassword(), u.getPassword())

## Pool de connexion

1. Ajoutez les librairies boneCP.jar, guava.jar et slf4J.jar dans le dossier WEB-INF/
2. Ajoutez les librairies dans le ClassPath.
3. Dans la classe DaoFactory créez une configuration BoneCP au lieu d’instancier une connexion JDBC.



# Exercice 7

Nous allons maintenant utiliser le framework Hibernate pour utiliser les EJBs 3.

Pour cela installez le container d’EJB glassfish

Creez votre fichier persistence.xml dans META-INF

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**persistence** version=*"2.1"* xmlns=*"http://xmlns.jcp.org/xml/ns/persistence"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd"*>

<**persistence-unit** name=*"bdd\_PU"* transaction-type=*"JTA"*>

<**jta-data-source**>jdbc/bonecp\_resource</**jta-data-source**>

<**class**>com.test.model.Utilisateur</**class**>

<**class**>com.test.model.Vehicule</**class**>

<**class**>com.test.model.Voiture</**class**>

<**class**>com.test.model.Motocyclette</**class**>

</**persistence-unit**>

</**persistence**>

Creez votre fichier glassfish-web.xml dans WEB-INF

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<!**DOCTYPE** glassfish-web-app PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Servlet 3.0//EN" "http://glassfish.org/dtds/glassfish-web-app\_3\_0-1.dtd">

<**glassfish-web-app**>

<**context-root**>/TestServletJPA</**context-root**>

<**class-loader** delegate=*"true"*/>

<**jsp-config**>

<**property** name=*"keepgenerated"* value=*"true"*>

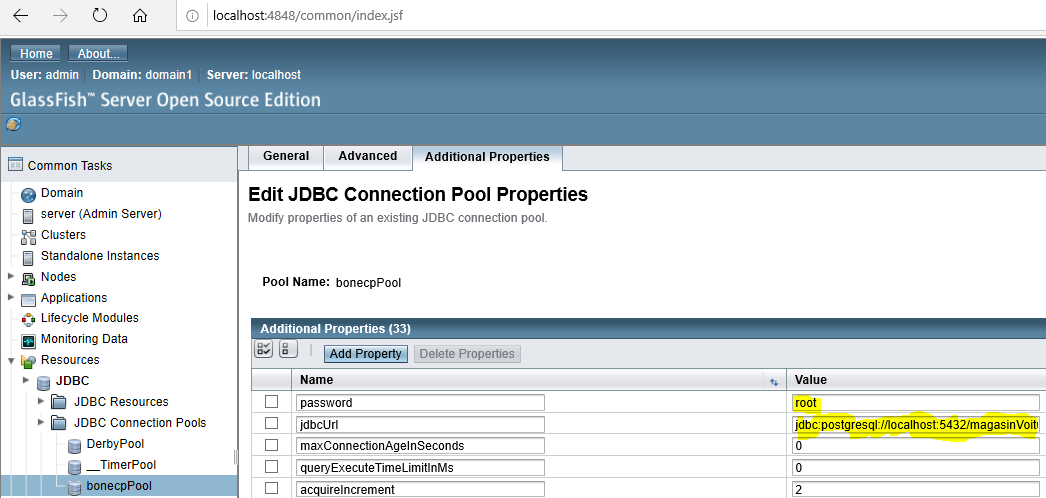
<**description**>Conserve une copie du code des servlets auto-générées.</**description**>

</**property**>

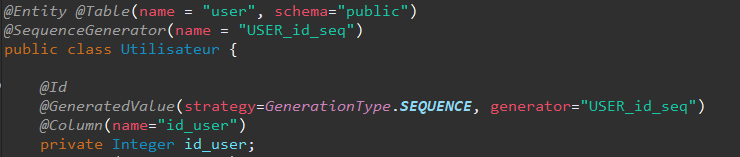
</**jsp-config**>

</**glassfish-web-app**>

Dans votre page d’administration glassfish <http://localhost:4848/common/index.jsf> mettez les détails de votre unité de persistence



Annoter les classes du modèle :



Ajouter les routes directement dans vos servlets



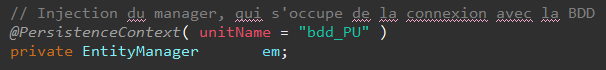
Et ajoutez y vos daos par injections grâce aux annotations



Dans les Dao ajouter l’annotation stateless pour en faire des managedBean.



Insérer l’entity Manager



Et mettez à jour le corps des méthodes des dao

