Application HMABWE

Guide de programmation

Table des matières

Structure de l’application

L’application est structurée suivant le modèle de programmation MVC (Modèle-Vue-Contrôleur).
Cette structure rend modulable le développement de l’application, et il est ainsi aisé de travailler sur une partie seulement de l’application sans impacter trop de changements dans les autres parties. En pratique les différentes parties sont structurées dans des packages différents:

- fr.unice.hmabwe.**controleur**
- fr.unice.hmabwe.**modele**
- fr.unice.hmabwe.**vue**

Partie Modèle

La partie modèle est constituée des différents objets représentant les entités métier de l’application :

- L’étudiant (classe Etudiant)
- L’enseignant (classe Enseignant)
- Le cours (classe Cours)
- La filière (classe Filiere)
- L’inscription (classe Inscription)
- Le coefficient (classe Coefficient)

On les trouve dans le package fr.unice.hmabwe.modele.

Remarque : on a la classe Etudiant et Enseignant qui héritent de la classe Personne, et on justifie cela par la volonté de factoriser le plus possible le code et éviter ainsi la redondance de colonnes dans la base de données.

Partie Vue

La partie Vue est composée de tous les panneaux graphiques et écouteurs qui constituent l’interface graphique et qui participent à l’expérience utilisateur. Notamment c’est cette partie qui va accueillir le Main de l’application.

On les trouve dans le package fr.unice.hmabwe.vue.

Remarque : Tout les constructeurs des fenêtres prennent en paramètre une instance du DAO qu'il va utiliser. Il faut aussi remarquer que les écouteurs sont déclarés en classes internes à l'intérieur des classes qui les appellent.

Partie Contrôleur

La partie Contrôleur est constituée des sous-parties suivantes :

- Entrée/sortie
- Interaction avec la base de données

Entrée/sortie

La sous partie Entrée/sortie gère d'une part l'import et l'export mais aussi le publipostage des notes aux élèves.
Cette sous-partie est structurée de la manière suivante :

- fr.unice.hmabwe.controleur.es.mail
- fr.unice.hmabwe.controleur.es.tableur

Ces packages utilisent respectivement les librairies suivantes :

- mail.jar pour l'envoi de mails
- jxl.jar qui est utilisée pour l'import et l'export de fichiers excel

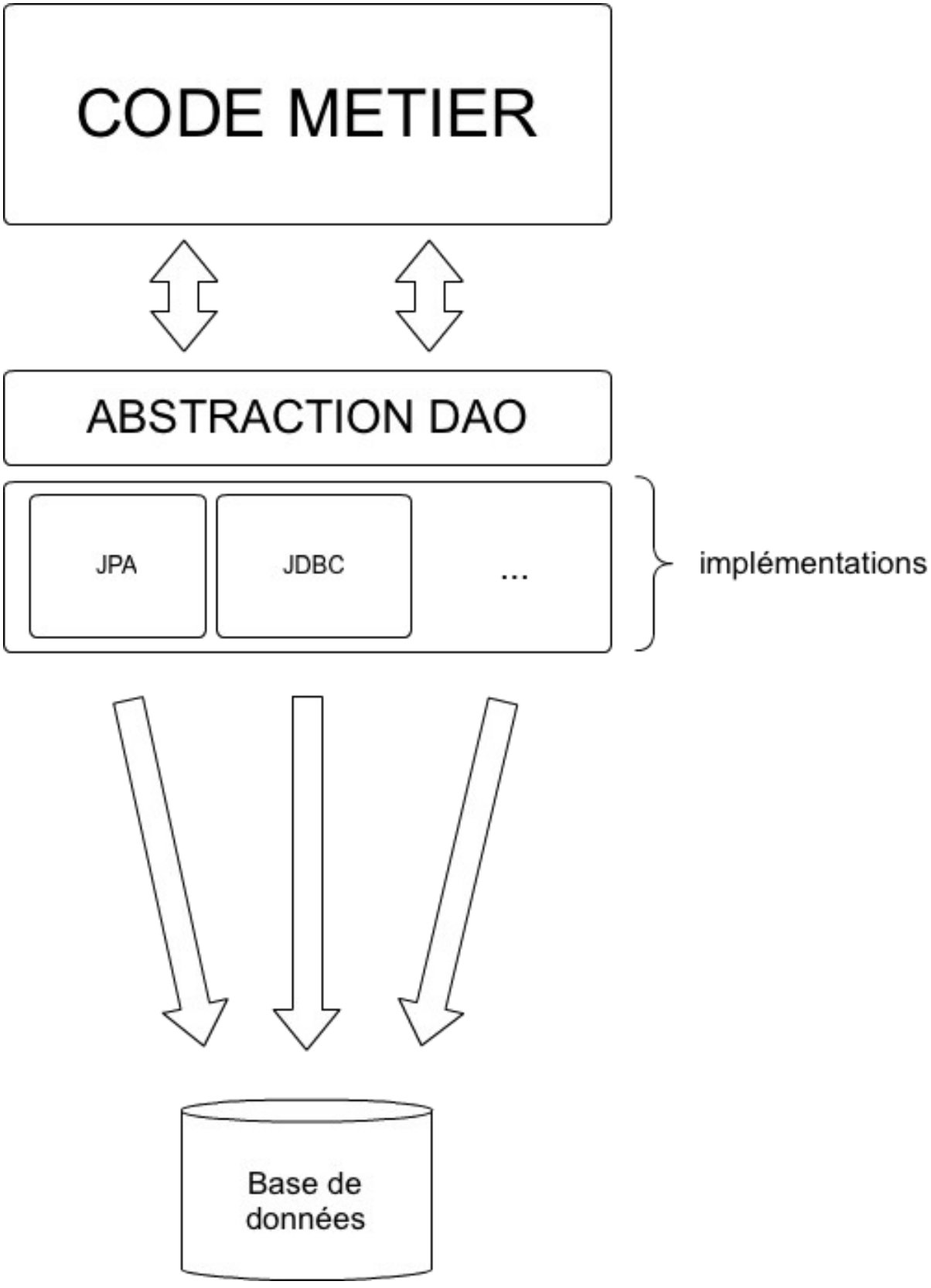
Interaction avec la base de données

On accède à la base de données par les Classes DAO correspondants aux objets métier que l'on manipule. Si on veut par exemple gérer la persistance d'un étudiant on utilisera le DaoEtudiant. De même :

- DaoCours gère la persistance d'un Cours.
- DaoEnseignant gère la persistance d'un Enseignant.
- DaoInscription gère la persistance d'une Inscription
- DaoFiliere gère la persistance d'une Filiere
- DaoCoefficient gère la persistance d'un Coefficient

Chaque DAO possède d'office 4 méthodes dites CRUD (Create, Retrieve, Update et Delete) qui permettent une interaction basique avec la base de données.

Les DAO forment la couche qui permet au code métier d’interagir avec la base de données, notamment pour faire des requêtes plus évoluées.



Les DAO sont utilisés ici pour masquer le type de persistance utilisé, le but étant de donner la possibilité au programmeur d'implémenter d'autres types de persistance, en impactant le moins possible le reste du code de l'application.

C'est le package fr.unice.hmabwe.bd qui possède toutes les classes regroupant les DAO. Lui même est découpé en sous packages selon la structure suivante :

- fr.unice.hmabwe.controleur.bd
- fr.unice.hmabwe.controleur.bd.config
- fr.unice.hmabwe.controleur.bd.dao
- fr.unice.hmabwe.controleur.bd.dao.jpa

Nous allons présenter chaque sous package maintenant.

fr.unice.hmabwe.controleur.bd

Ce package contient l'interface de la Connexion, que devront implémenter les différents types de persistance. En l'occurrence ici il n'y a que l'implémentation pour le type de persistance JPA.

Listing du package :

- fr.unice.hmabwe.controleur.bd.Connexion
- fr.unice.hmabwe.controleur.bd.JpaConnexion

Remarque : l 'utilisateur n'instanciera jamais une Connexion mais la récupérera de la FabriqueDao associée au type de persistance. (voir plus bas)

fr.unice.hmabwe.controleur.bd.config

Ce package contient l'interface ConfigConnexion qui servira à occulter la configuration de la Connexion spécifique au type de persistance utilisé, cette classe jette néanmoins des ConnexionException qui encapsulent les exceptions internes au type de persistance utilisé, et permet de s'en abstraire. Enfin il y a l'implémentation JPA de la ConfigConnexion.

Listing du package :

- fr.unice.hmabwe.controleur.bd.config.ConfigConnexion
- fr.unice.hmabwe.controleur.bd.config.ConnexionException
- fr.unice.hmabwe.controleur.bd.config.JpaConfigConnexion

Remarque : pour que l'utilisateur puisse configurer une Connexion, il devra récupérer une nouvelle ConfigConnexion en appellant la méthode de classe newConfigConnexion() de ConfigConnexion. Ensuite il pourra modifier cette nouvelle configuration avec la méthode setProprietes() (voir la javadoc pour la signature complète de la méthode), et enfin rendre persistant cette configuration avec la méthode d'instance sauvegarder().

Exemple :

```
ConfigConnexion configuration = ConfigConnexion.newConfiguration();
configuration.setProprietes("euterpe.unice.fr",
                            "numero_port",
                            "nom_utilisateur",
                            "mot_de_passe",
                            "nom_base_de_donnees");
configuration.sauvegarder();
```

fr.unice.hmabwe.controleur.bd.dao

Ce package contient toutes les interfaces nécessaires à l'abstraction du type de persistance, et représentent les *objets* que va manipuler l'utilisateur.
Ce sont ces classes que devra implémenter le programmeur, pour avoir à modifier le moins possible le reste du code de l'application.

Pour faciliter la tâche du programmeur et ainsi éviter la redondance de code évidente, chaque DAO spécifique à un objet métier implémente déjà par défaut les méthodes CRUD basiques grâce à l'interface DaoGenerique<T, ID implements Serializable>. Ainsi si le programmeur souhaite implémenter un autre type de persistance, il n'aura qu'à coder sa classe abstraite TypePersistanceDaoGenerique<T, ID implements Serializable>, ainsi , il ne lui restera qu'à implémenter que les méthodes spécialisées à des DAO en particulier dans ses classes d'implémentation, qui devront alors hériter de cette classe générique paramétrée. Le paramétrage de DaoGenerique permet de factoriser le code des méthodes CRUD et contraindre le type de la clé primaire de l'objet métier qui serra alors enregistré dans la base de données.

Remarque : on n'instancie jamais un DAO métier. Dans un premier temps on renseigne quel type de persistance on utilise dans l'application à la classe DaoFabrique (pattern Fabrique abstraite), ensuite on demande à la classe DaoFabrique une nouvelle fabrique de Dao. (voir exemple plus bas)

fr.unice.hmabwe.controleur.bd.dao.jpa

Ce package est l'implémentation des DAO pour le type de persistance JPA. Elles respectent les exigences des interfaces à implémenter et fournissent le comportement attendu par l'utilisateur, le comportement interne étant masqué par les interfaces.

Remarque : Il est préférable de garder l'implémentation des DAO d'un type de persistance donné dans son propre sous package, pour ainsi garantir et faciliter la pérennité du code.

Exemple d'utilisation :

```
DaoFabrique df = DaoFabrique.getDaoFabrique();
Connexion conn = df.getConnexion();
DaoEnseignant daoEnseignant = df.getDaoEnseignant();
DaoCours daoCours = df.getDaoCours();
Enseignant ens = new Enseignant("Grin",
                                "Richard",
                                "grin@unice.fr");
Cours cours1 = new Cours("POO", ens);

try {
      conn.beginTransaction();
      daoEnseignant.create(ens);
      daoCours.create(cours1);
      conn.commitTransaction();
}
catch(DaoException e) {
      try {
            conn.rollbackTransaction();
      }
      catch(DaoException ee) {
            ee.printStackTrace();
      }
```