**STRUCTURE DU PROJET :**

1. Le nom des classes en CamelCase
2. Connection base postgres:

          -dans la classe framework.utilitaire.GConnection, modifer les valeurs des champs static selon votre base de donnees

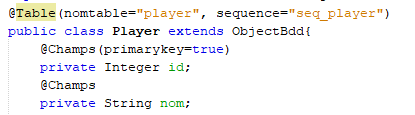
1. Gestion des méthodes utiltaires:

-mettre toutes les classes utilitaire dans le package util, par exemple pour le calcule de date .

1. Model

Lorsque le nom de la classe et la table dans la base sont differents, mettre la valeur du nomTable le nom de la table dans la base et sequence sa sequence

ex:



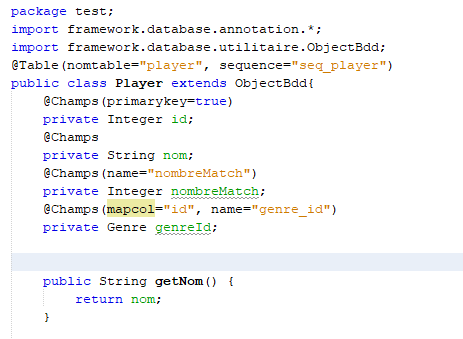
           -mettre tous les models dans packages model,

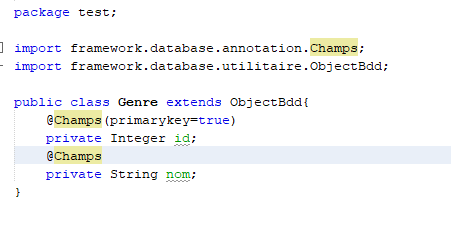
           -les models qui sont relié à une table de la base doivent extends ObjectBdd (ObjectBdd se trouve dans framework.utilitaire)

          -pour utiliser ObjectBdd et les méthode géneraliser:

* créer une classe qui extends ObjectBdd
* importer framework.utilitaire.\* et framework.annotation.\*;
* creer les attributs et les setters et getters, les attributs avec les annotations @Champs doivent tous être des objets donc on utilise pas des types primitifs sur eux.

ex:





* les methodes statics de ObjectBdd sont:

|  |  |  |
| --- | --- | --- |
| method | commentaire | exemple |
| static Object getColByExec(String sql, Object o): | ca rend un objet du type de l’objet envoyer | par exemple Player.getColByExec(“SELECT 4+4 FROM table”,1), rend 8 |
| static int sequence(String name, Connection) | rend la valeur de la sequence name | ObjectBdd.sequence(«masequence», con) |
| static int sequence\_sql(String sql,Connection) | rend la sequence selon la requete sql |  |

* les méthodes des classes qui ont herites ObjectBdd sont:

|  |  |  |
| --- | --- | --- |
| method | commentaire | exemple |
| void create(Connection ): | insere l'objet dans la base, si la connection est null, la methode se chargera de gerer ca et creera sa propre connection |  |
| void createAll(Connection , ArrayList<Object>liste) | insere la liste des objet données dans la base, ca gere la connection |  |
| void update(Connection,String where, String set) | si par exemple on a player.update(con, “id=2”,”nom=’camille’)=> et ca va update player set nom=’camille’ where id=2. |  |
| void update(Connection) | ca va juste update tout l’objet |  |
| void delete(Connection) | ca va delete l’objet dans la base |  |
| ArrayList<T> findAll(Connection con) | rend toute la liste dans la table, le T veut dire qu’il va rendre un objet du meme type qui va l’appeler, par exemple si je fais player.findAll(null),ca va rendre une ArrayList<Player>, donc pas besoin de caster. |  |
| ArrayList<T>findWhere(Connection con, String where) | rend la liste selon une condition where | par exemple: player.findWhere(con, “nom like ‘camille%’; |
| ArrayList<T>findBySql(Connection con, String where) | rend une liste selon ta requete | player.findBySql(null, “SELECT \* from v\_player where 2=2”) |
| T findById(Connection , Object id): | rend un objet selon son id | par exemple player.findById(2) |
| T findOne(Connection) | rend la premiere ligne d’un findAll |  |
| T findOneWhere(Connection,string where) | rend un objet selon une condition where | player.findOneWhere(null, “nom is null”) |
| T findOneBySql(Connection, String sql) | meme chose que findBySql mais rend qu’un seul objet |  |
| ArrayList<T>findAllByLimit(Connection,Integer debut, Integer fin) | rend les données de la table de la ligne debut jusqu’ a la ligne fin, si debut est null alors ce sont les données dont les ligne sont inferieurs ou egale a fin, si fin est null alors ce sont les données dont les lignes sont supérieurs ou égale a debut |  |
| ArrayList<T>findWhereByLimit(Connection con, String where,Integer debut, Integer fin) | comme findWhere(con, where) mais de la ligne du debut jusqu’a la ligne fin et meme gestion de ligne que findAllByLimit |  |

1. Gestion des controller/Servlet:

- mettre les classe Servlet dans le package controller

-nommage des Servlet:

-mettre Controller a chaque fin de nom, par exemple: EmployeController

-Par exemple pour les operations crud sur un Employe: EmployeCreateController, EmployerUpdateController, EmployerViewController pour voir un employe, EmployerViewsController pour voir une liste d’employe.

**FRONT**:

-utiliser le template

-pour les formulaire:

-les nom des attributs sont en snake\_case: dia mifanaraka @ classe relier @, par exemple, pour l’insertion d’employe: ny name pour les input, c’est date\_naissance ohatra, date\_embauche...

-pour les nom des fichiers: ils sont en snake \_case: dia manaraka create\_employe.jsp, views\_employe raha liste des employes, view\_employe raha employe iray, update\_employe.jsp raha formulaire de update, sns, et on fait comme ca pour tous les noms des fichiers raha crud.