**BASE:**

      -use snake\_case: les tirets \_ separe les mots:

  ex: personne\_id

-pour les foreign key: atao du format: personne\_id fa tsy idpersonne ou autre

-pour les sequence, raha sequence relier a une table dia format: seq\_nomtable

-pour les view, le format cest v\_nomvue;

**METIER:**

1. Le naming se fait en CamelCase
2. Integration de la base:

           -créer un user garage avec le mot de passe garage, creer une base de données garage, metter garage en tant que propriétaire de la base de donnée garage

-puis connecter vous a garage et metter la base dans garage

1. Connection base postgres:

          -dans la classe framework.utilitaire.GConnection, il y a les attributs static qui definissent les parametres de la connection a la base;

          -les modifiers par rapport a votre base;

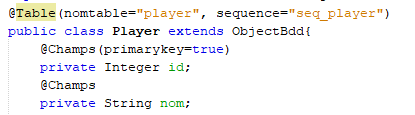
1. Gestion des méthodes utiltaires:

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

1. Explication d’une annotation:

Simplement, les annotations sont des marqueurs ou des commentaires ajoutées a un code source,il permet d’ajouter des informations en plus concernant des classes/methodes/attribut…, ces annotations servent a récupérer des informations durant l’excécution du code.

ex:



Ici, par exemple, sur la classe Player on a l’annotation @Table avec les parametre nomtable et sequence qui nous aidera a avoir les information que Player a une table dans la base et son nom c’est “player”, et que la sequence relier a la generation de son id est “seq\_player”

1. Model:

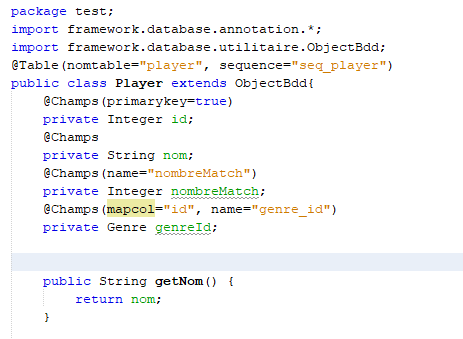
           -mettre tous les models dans packages model,

           -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:



Par exemple, ici on a une classe player,

-explication de @Table: utiliser @Table et nomtable pour dire le nom de la table dans la base de données, et sequence s’il y a une sequence relie a cette table, par defaut , si vous n’annotez pas la classe, alors le nom de la table sera le nom de la classe et si on ne reseigne pas la sequence relié a la table alors il n’y aura pas de sequence;

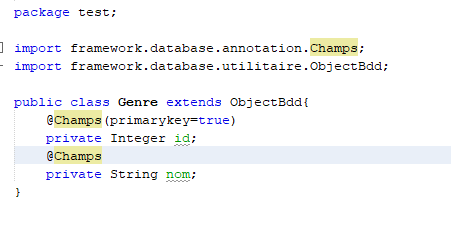
-explication de @Champs :

-l’annotation @Champs indique que l’attribut est une colonne de la table player, si on ne l’annote pas, donc ce n’est pas une colonne de la table player, et ca ne va pas le prendre en compte dans la generation de requete sql.

-les parametres primarykey=true, indique que l’attribut id est la cle primaire de cette table,

-le parametre name=”nombreMatch” veut dire que le nom de la colonne dans la base est nombreMatch sinon par defaut il va prendre le nom de l’attribut

-le parametre mapcol=”id”, veut dire que cette colonne est relie a la table de la classe Genre selon l’attribut id de la classe Genre, qui est ci dessous, le paramètre mapcol ne marche que pour les colonne de type ManyToOne, ce qui veut dire que player a un genreId qui est un foreign key par rapport a id dans la table Genre;



* puis renseigner votre classe en utilisant les annotations @Table sur la classe et @Champs sur les attributs,
* votre classe hérite donc des méthodes de ObjectBdd, les methodes disponibles gerent les connection a la base, ca veut dire que si la connection passé est null , la méthode se chargera de créer automatiquement une connection:
* 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 que votre classe qui ont extends 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:

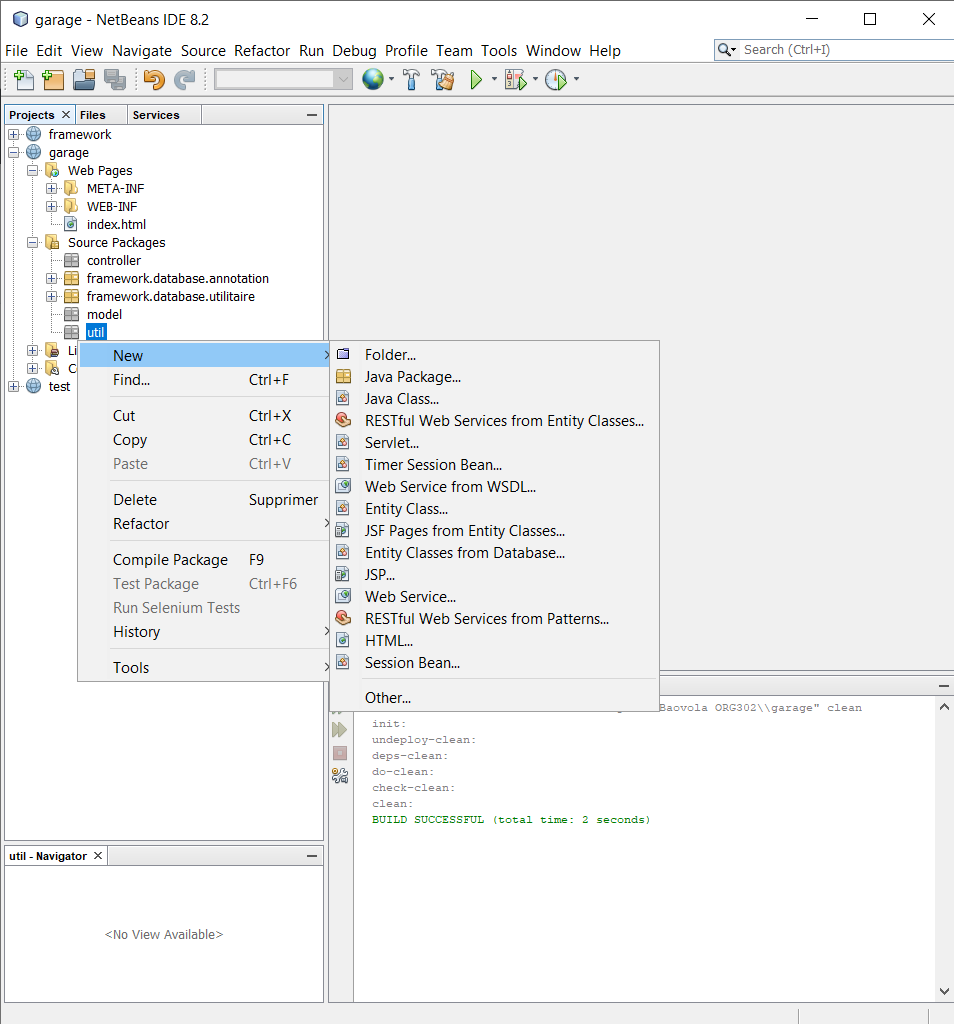
-les nom des attributs sont en snake\_case: dia mifanaraka @ classe relier @, par exemple, pour l’insertion d’employe: ny name pour les input, c’est date\_naissance ohatra, date\_embauche...

-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.

**Comment utiliser netbeans:**

-ouvrir votre projet sur netbeans

-et le projet aura cette structure:



-Les code java seront dans source packages:

-Les modeles : dans le package model

-Les servlet et controller dans controller

-Les classes utilitaires pour les methode utilitaire , dans le package util.

-Les pages jsp seront dans web pages

-Les assets pour les fichiers css des pages jsp sont dans Web Pages

-Pour creer un nouveau truc, click droit sur la où vous voulez créer votre fichier ou dossier, et choisir new et choisir ce que vous voulez créer.

-Pour runer le projet, clicker sur le triangle vert, ou sinon click droit sur le projet et runer

-Pour runer un fichier, click droit sur le fichier et run

-Pour generer automatiquement les setter et getter de vos attributs, click droit dans le code, et choisir insert code..., puis choisir getter and setter, on peut egalement generer d’autre trucs automatiquement

