**Cours 26 janvier 2018**

**. HIBERNATE .**

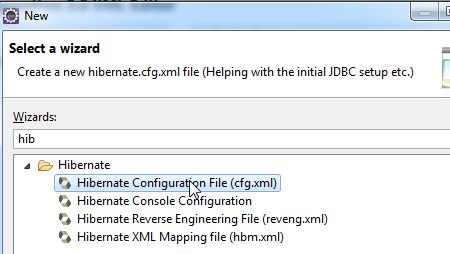
Professeur : Thibaud de Mijolla

Un projet pour les interfaces, un projet pour IDataUsers, un pour IDataProduit, etc

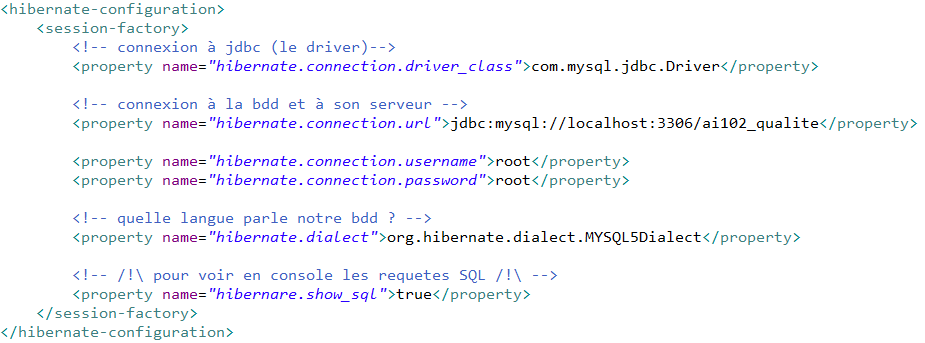
Hibernate implémente JPA.

TP :

Création du hibernate.cfg

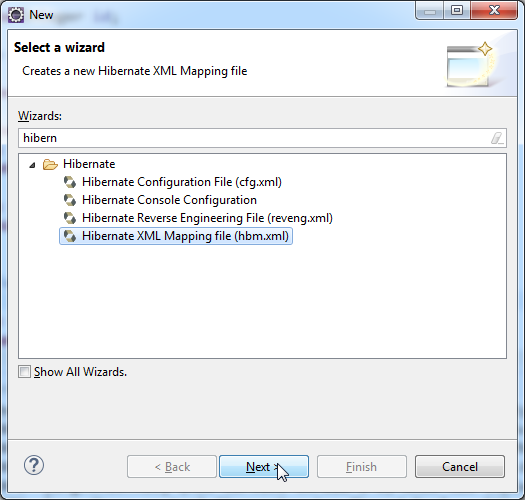


Dans ce fichier de configs, on remplit :



On aura aussi besoin d’un hibernate.util

On crée le mapping



Manque le cours de l’après midi !!

**Cours 27 mars 2018**

**. HIBERNATE .**

Professeur : S. Péricard

Hibernate est un ORM : Object Relation Mapping

Parce qu’on est en object.

La configuration se retrouve par défaut dans le fichier hibernate.cfg.xml ;  
Il contient

* le mapping des entités (les entités qu’il doit lire au démarrage pour pouvoir générer les requêtes) ;
* la datasource pour pouvoir se connecter à la base (datasource qui possède toujours le username, le password, l’url au format jndi) ;

Hibernate ouvre une session (avec la SessionFactory) ;  
La session permet de sauvegarder un objet save(Obj) pour l’enregistrer dans sa session ; il gère l’unicité avec sa clé primaire, elle permet l’identification de l’objet.  
Si la base n’en a pas, hibernate en crée pour les entités objet.  
Hibernate peut aussi update, delete, il peut rechercher (get par clé primaire ou load)  
Get retourne null s’il ne trouve pas, alors que load lève une exception 🡪 permet le choix pour le développeur.

Hibernate permet aussi de créer des requêtes : SQL natif, ou queries qui seront en HQL (Hibernate Query Language)

On n’a donc plus besoin de connaitre la base pour créer des requêtes !

Une fois Hibernate configurée, il faut paramétrer nos entités.  
@nnotations d’Hibernate qui sont celles de JPA

Java Persistence API est aussi un framework ORM qui ne contient que les services (sous forme d’interface) et les @nnotations (aussi sous forme d’interfaces).

Sun ne fournit donc que l’API.  
Dans les deux grandes implémentations de JPA on retrouve : Eclipse & RedHat.

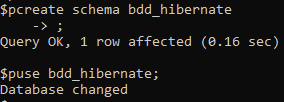
Hibernate est une implémentation de JPA.

Le fichier de paramétrage de Hibernate est hbm.xml

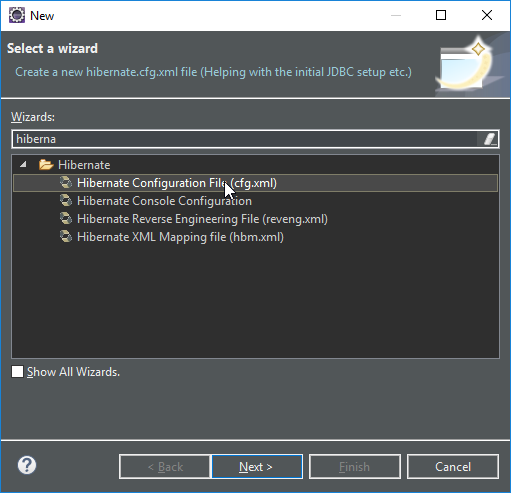
Hibernate.hbm2ddl.auto :

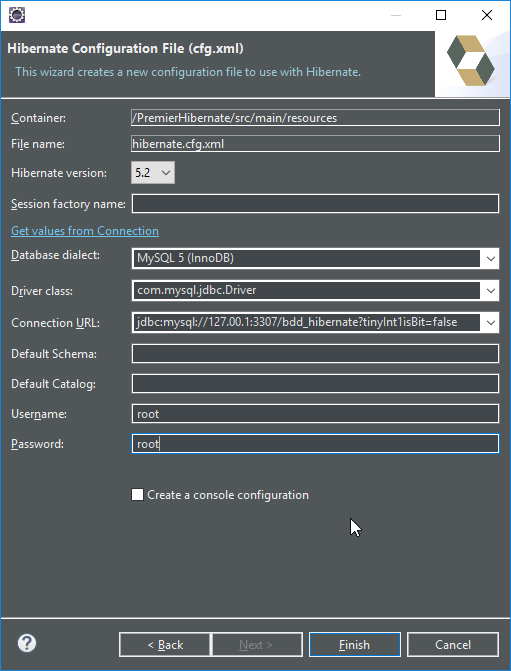
* create 🡪 import.sql
* validate
* update

Dans la génération de la base, le seul truc que nos ORM ne peuvent faire c’est le schema : ce n’est pas à nous de le faire, il est donné.

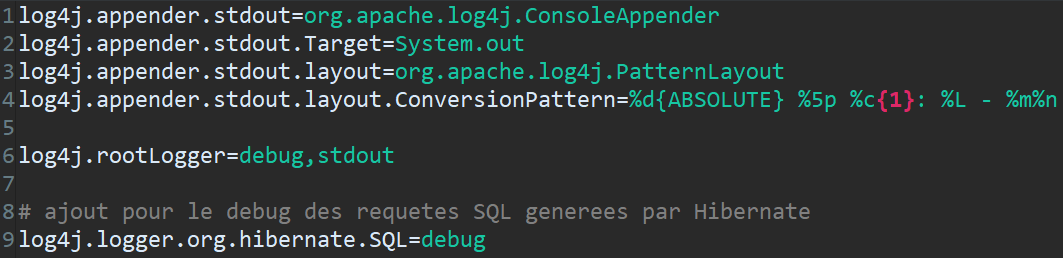
On crée la base :  


Création dans src/main/resources

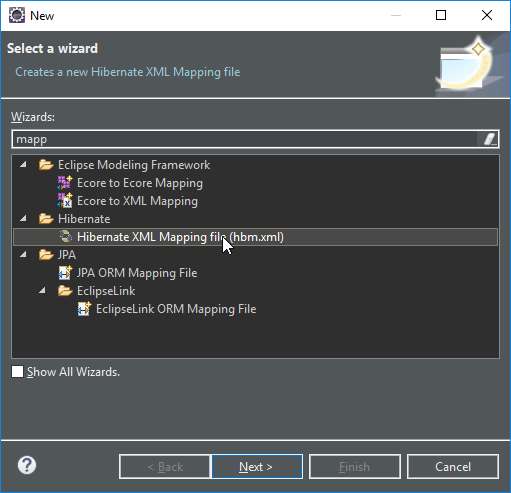




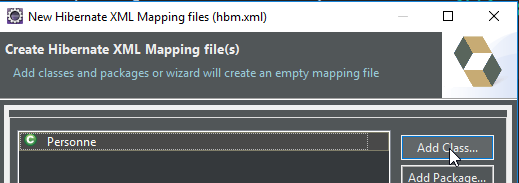
Aout d’une ligne pour Hibernate dans log4j prop :

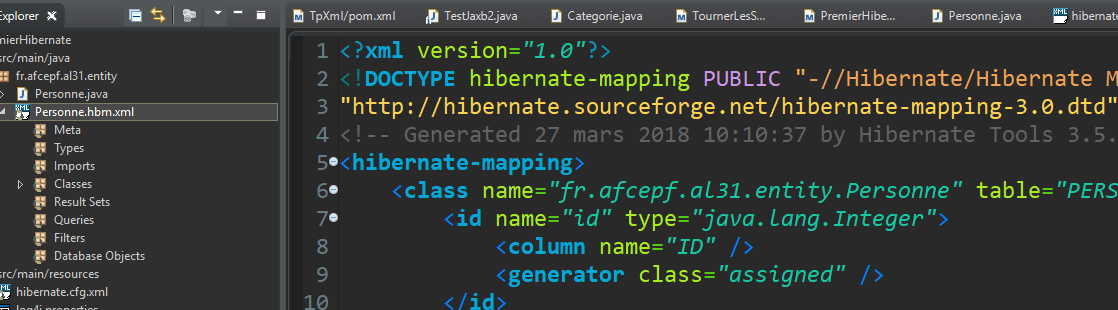


Utilisation du mapping CTRL N puis :

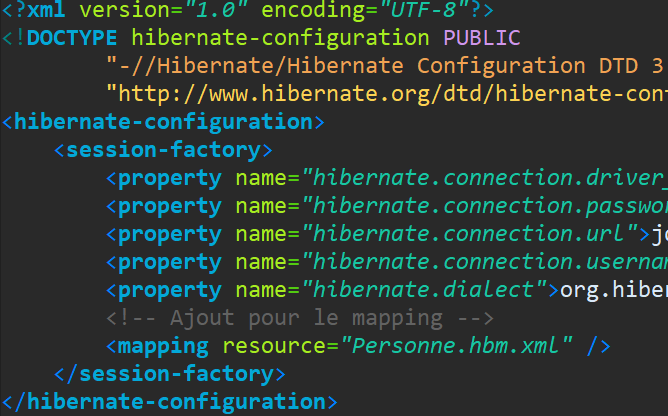


Recherche de la classe adéquate puis finish :

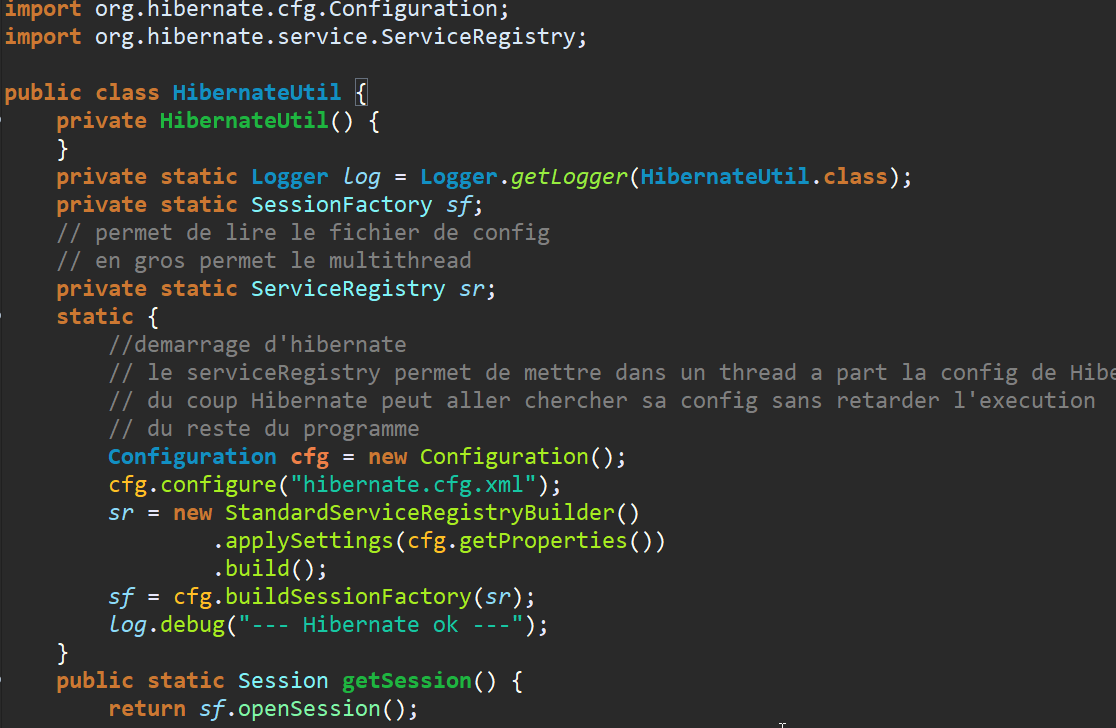


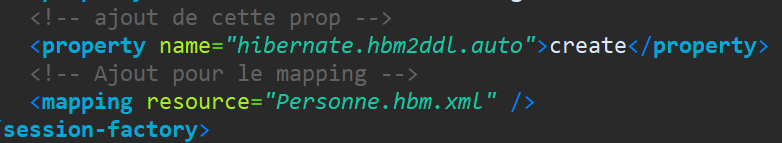
On a donc ceci :  


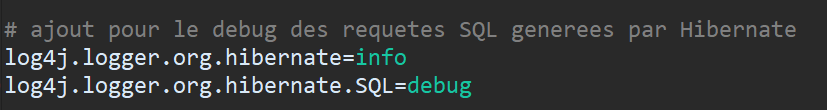
On déclare ensuite (dans hbm.cfg.xml) via un mapping notre entité, pour qu’il voit la ressource au démarrage

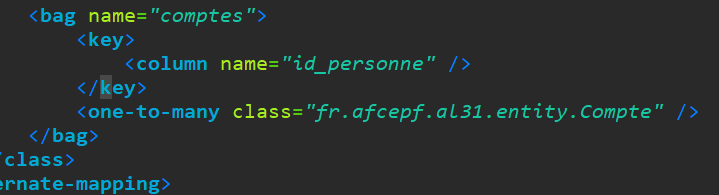


Création d’un classe utilitaire :



Ajout dans les configs de hibernate de hbm2bddl.auto :  


Petite modif du fichier de prop log4j, pour lisibilité :  


Ajout d’un BAG dans Personne.hbm.xml:  


Ajout au configs hibernate de la création des nouvelles tables :

