**ENSET-M 2021-2022**

**COMPTE RENDU**

**TP : JPA Hibernate Spring Data - Many To Many**

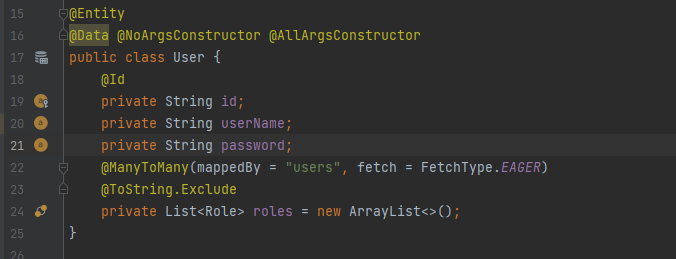
Par : **Assimi DIALLO** Encadrant : **M. YOUSSFI**

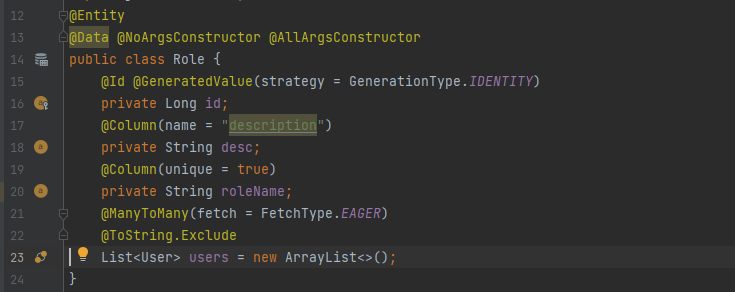
**JPA Hibernate Spring Data - Many To Many**

Dans ce TP nous allons voir comment se font les relations Many To Many avec JPA Hibernate et Spring Data. La particularité des relations Many To Many c’est la création de la table d’association résultante des 2 autres.

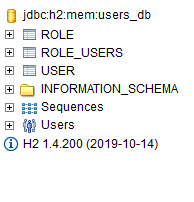
Nous allons créer une table **Users** et une table **Role.** Entreces 2 tables on a une relation Many To Many car un User peut avoir plusieurs rôles et un Role peut correspondre à plusieurs users.

Création des entités : On crée 2 classes User et Role. Chacune des 2 classes prend comme attribut une liste de l’autre classe. Juste au dessus de chaque liste on a l’annotation **@ManyToMany** et c’est ce qui rend effective la relation.



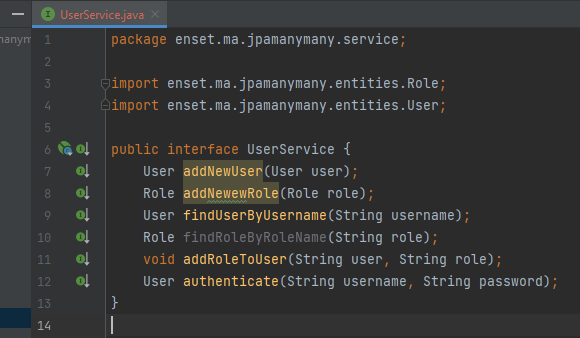


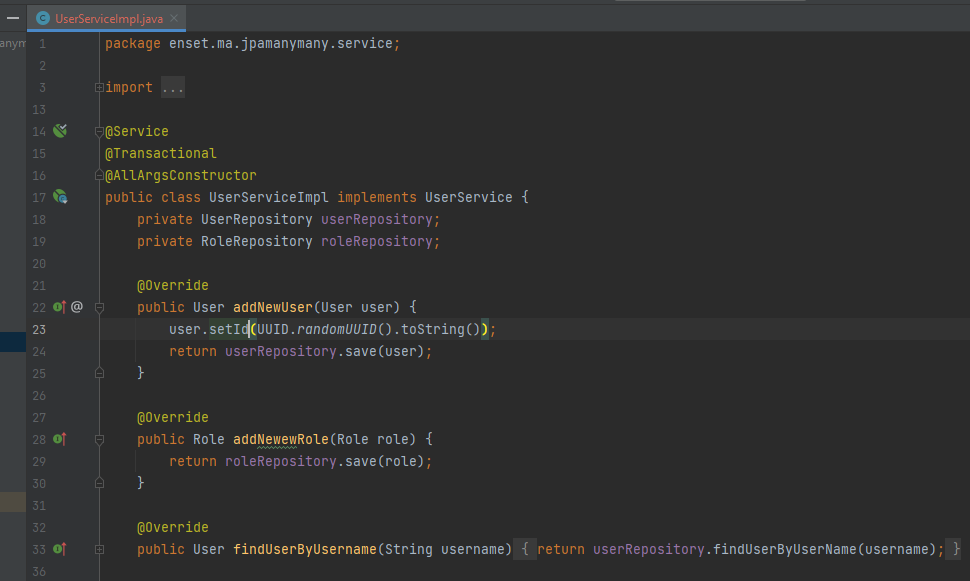
Quand on lance l’application on obtient dans notre base de données **H2** :



On remarque que les 2 tables ont été créées. Mais on remarque aussi la table d’association : ROLE\_USERS.

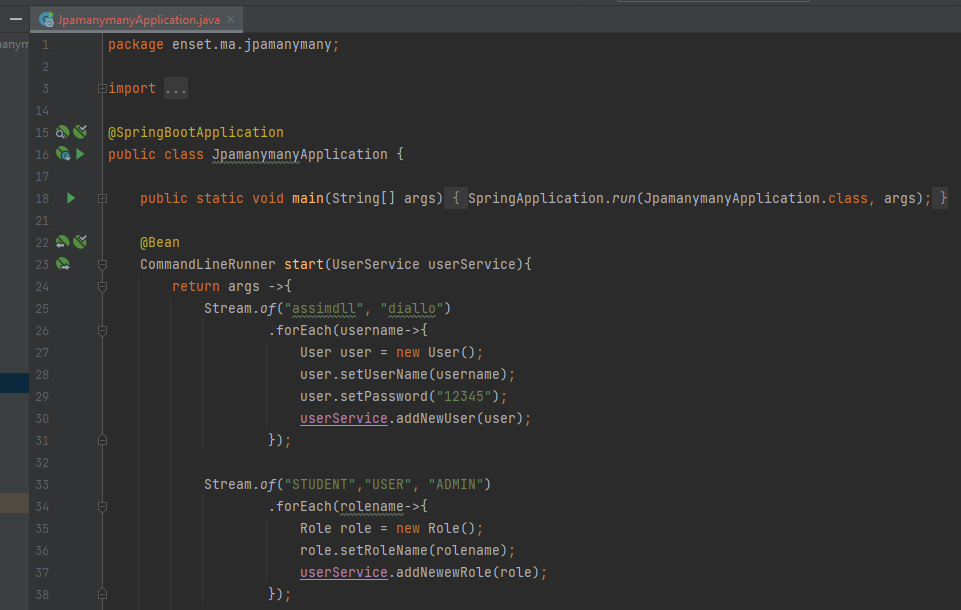
On crée une interface **UserService** dans un package **service.** L’implémentation **UserServiceImpl** de l’interface **UserService** va nous permettre d’interagir avec la base de données. On y utilise **userRepository** et **roleRepository** des interfaces **UserRepository** et **RoleRepository**, qui héritent de **JpaRepository**, et qui sont dans le package **repositories** :







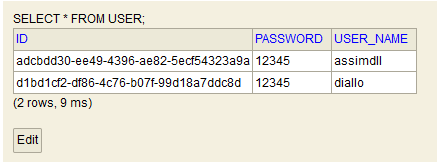
On peut donc faire les différentes opérations de remplissage de la base de données depuis **JpamanymanyApplication :**



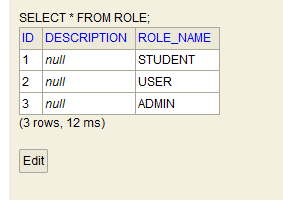


Quand on exécute **JpamanymanyApplication,** on obtient**:**

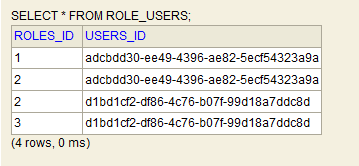
**Table USER :**



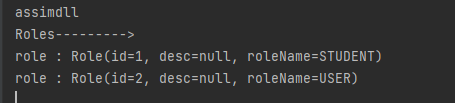
**Table ROLE :**



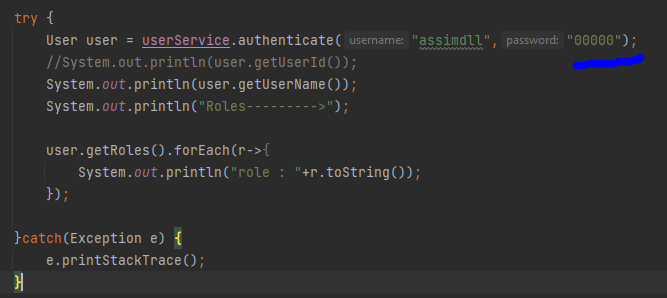
**Table ROLE\_USERS**



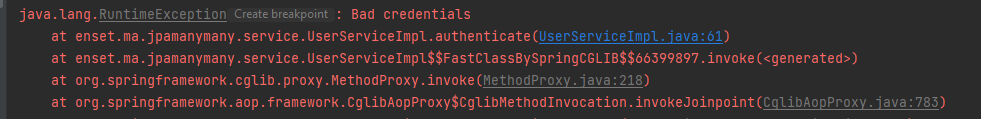
On peut vérifier aussi que la fonction authenticate definie tout en bas. On nous affiche l’utilisateur et ses rôles quand **le mot de passe** et le **username** sont corrects sinon on affiche **Bad Credentials**.



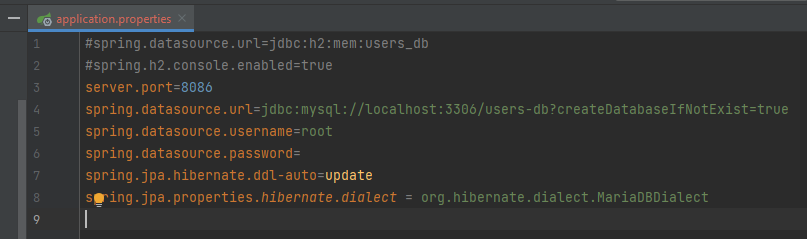
Maintenant modifions le mot de passe passé en paramètre et mettons un mot e passe erroné :

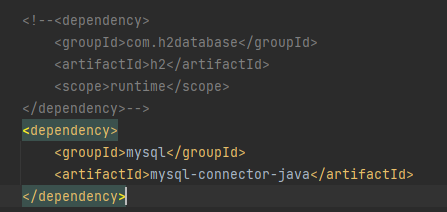


L’exécution nous donne cette fois ci :

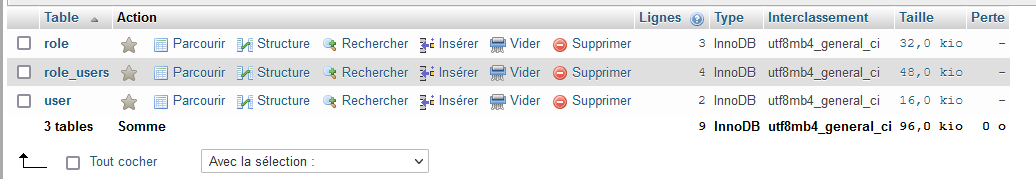


On peut maintenant déplacer notre base de données de H2 vers MySQL :





Quand on exécute :

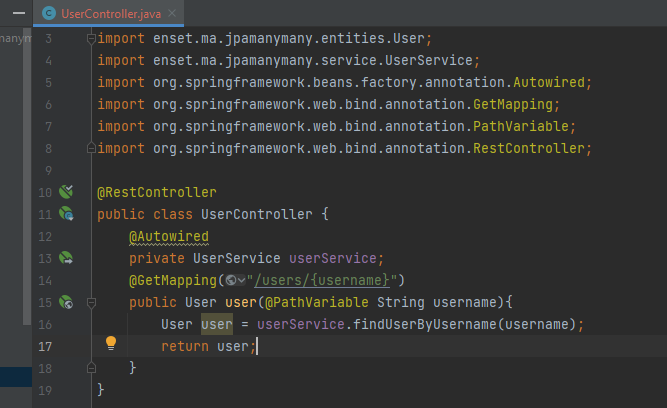




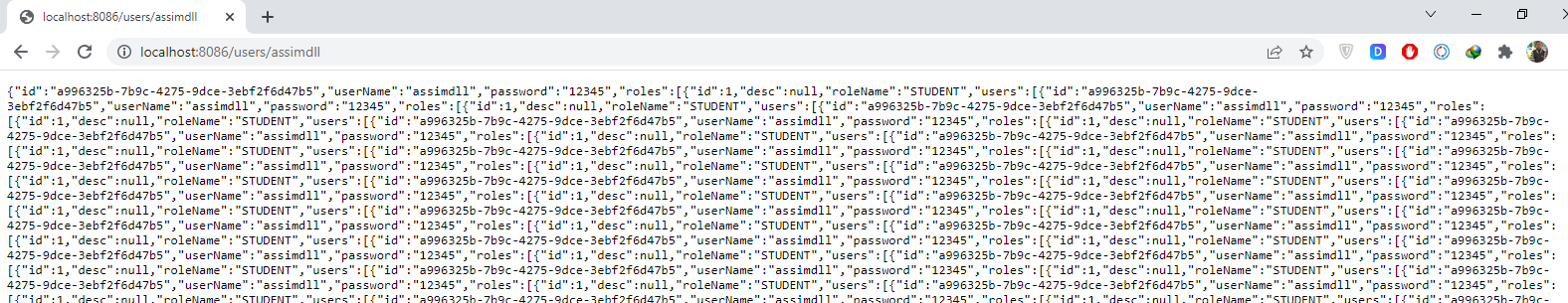




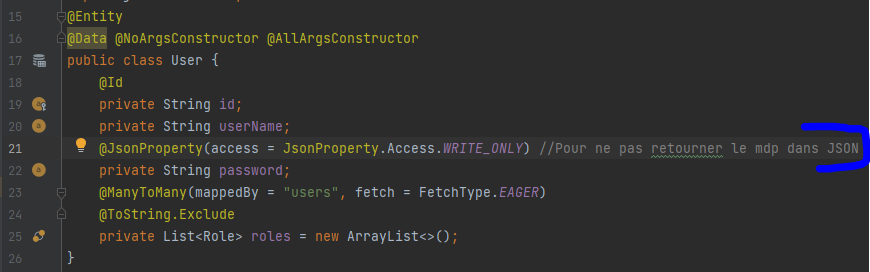
La prochaine étape sera de mettre en place un contrôleur pour retourner des données sous format JSON en fonction d’un paramètre passé dans l’url. Ce contrôleur sera défini dans un package **web** :

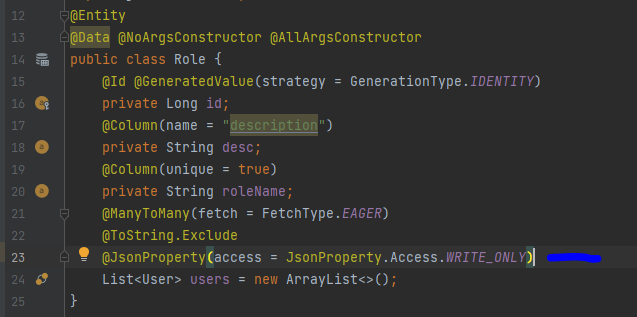


On obtient sur l’adresse http://localhost:8086/users/assimdll :

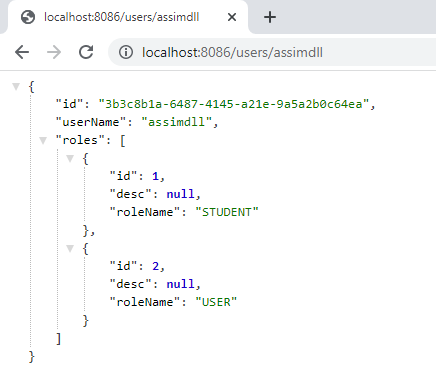


Il s’agit d’une boucle infinie car on retourne un utilisateur qui vient avec un rôle qui retourne l’utilisateur qui retourne le rôle et ainsi de suite. Pour y remedier on doit ajouter une propriété qui va empêcher le rôle d’afficher ou de retourner l’utilisateur ou les utilisateurs correspondant. On va profiter avec la même propriété empêcher l’affichage du mot de passe de l’utilisateur. On aura dans les classes User et Role :





Quand on exécute maintenant on obtient :



En allant sur l’adresse : http://localhost:8086/users/diallo



C’est ainsi que se termine ce TP sur JPA, Hibernate et Spring Data pour les relations Many To Many.