Labo Spring 5

Hibernate

Objectif : Accéder à une base de données MySQL en utilisant Hibernate

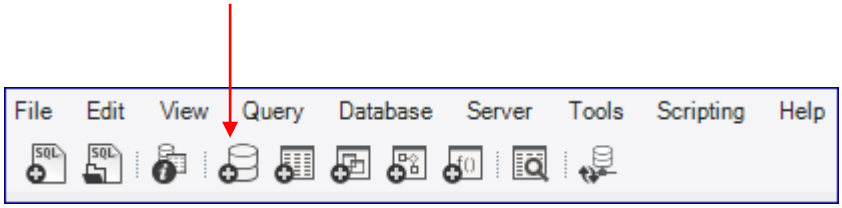
Dans la suite de l'exercice, le code magique va être recherché dans le base de données (accès en lecture) et l'enfant qui aura introduit ses coordonnées via le formulaire d'inscription sera enregistré dans la BD (accès en écriture).

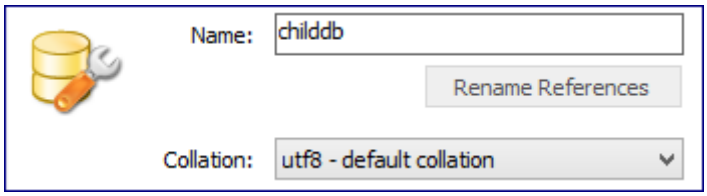
Etape 1 – Créer un schéma MySQL

Assurez-vous que le serveur MySQL tourne :

Task Manager ⇨ onglet Services ⇨ *MySQL56* : clic droit ⇨ Start

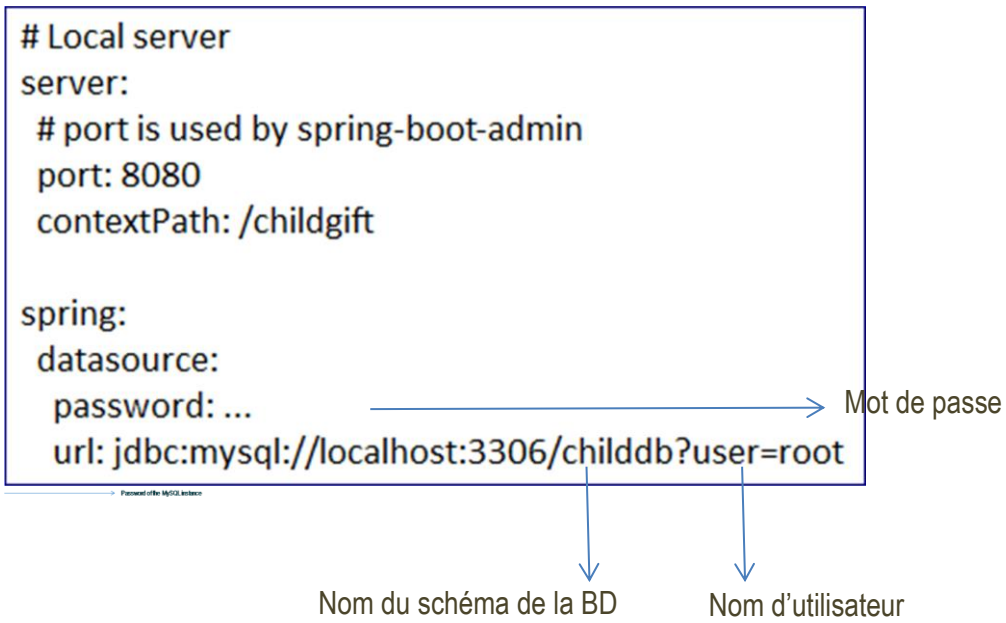
Créez un schéma MySQL appelé *childdb*.





Etape 2 – Configurer Spring Boot pour accéder à la base de données

Déclarez une *spring datasource* dans *application.yml*.



Ajoutez deux dépendances *dans pom.xml*.

```
<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
</dependency>
```

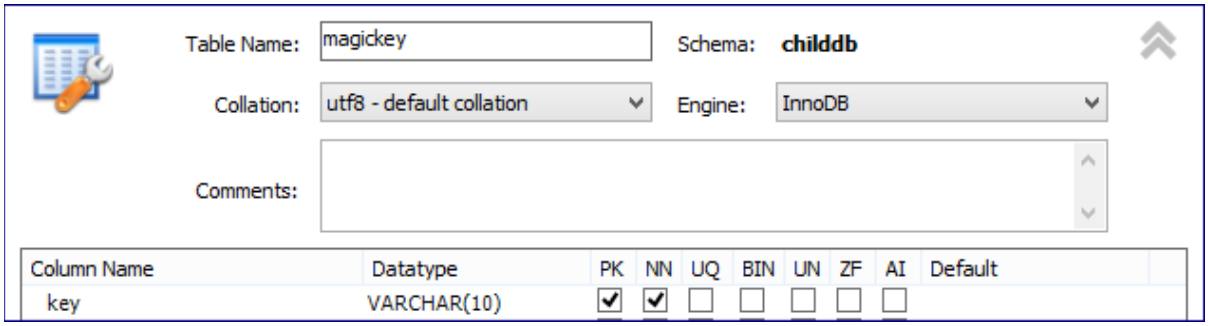
```
<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Etape 3 – Créer les tables

Créez la table *magickey*. N'y prévoyez qu'une seule colonne appelée *key* contenant les clés d'accès.

Cette colonne est identifiante.

N.B. Collation : utf8 – default collation

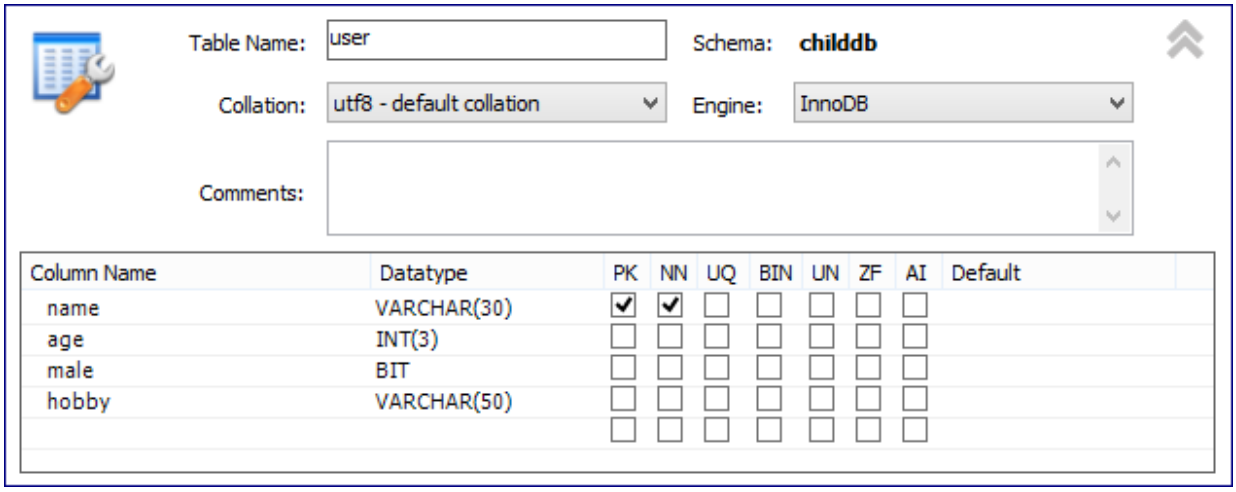


Insérez au moins 5 lignes dans cette table.

Créez la table *user*. Prévoyez les colonnes suivantes :

- *name* (identifiant)
- *age*
- *male* (ex : type BIT pour booléen – valeurs *true* ou *false*)
- *hobby*

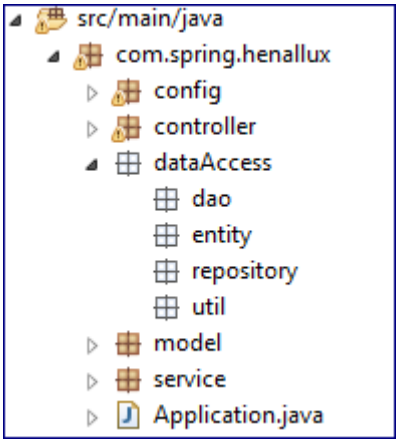
N.B. Collation : utf8 – default collation



Etape 4 – Créer les packages de gestion de la persistance des données

Créez le répertoire *dataAccess* dans le package *com.spring.henallux*.

Créez-y 4 nouveaux packages : les packages *entity*, *repository*, *util* et *dao*.



Etape 5 – Créer la classe *MagicKeyEntity*

Créez la classe *MagicKeyEntity* dans le package *entity*.

Cette classe doit être le miroir de la table *magickey* : utilisez les annotations *@Entity* et *@Table* pour lier cette classe à la table *magickey*.

Définissez-y une variable d'instance correspondant à la colonne identifiante *key* de la table (utilisez pour ce faire les annotations *@Id* et *@Column*).

N'oubliez pas les gettor/settor et au moins un constructeur sans argument.

Etape 6 – Créer l'interface *MagicKeyRepository*

Créez l'interface *MagicKeyRepository* dans le package *repository*. Cette interface offrira les méthodes CRUD de base pour lire et écrire dans la table *magickey*.

Cette interface hérite de l'interface générique *JpaRepository*. Soyez attentif à bien définir les paramètres de l'interface que vous créez : donnez le nom de l'entity class correspondante et préciser le bon type d'identifiant.

Utilisez les annotations *@Repository* et *@Transactional*.

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import com.spring.henallux.dataAccess.entity.MagicKeyEntity;

@Repository
@Transactional
public interface MagicKeyRepository extends JpaRepository<MagicKeyEntity, String>{
}
```

Pour rappel, vous devez définir l'interface mais pas donner d'implémentation. C'est Hibernate qui s'en charge !

Etape 7 – Créer la classe *MagicKeyDAO*

Créez la classe *MagicKeyDAO* dans le package *dao*.

Utilisez les annotations *@Service* et *@Transactional* sur la classe.

Cette classe doit proposer toutes les méthodes que vous voulez exposer aux couches supérieures.

Ces méthodes ne peuvent donc ni recevoir en argument ni retourner des objets de type Entity classes, car liées au type de persistance des données choisi, en l'occurrence ici une base de données relationnelle. Les type des arguments ainsi que les types de retour de ces méthodes doivent être des classes du package model.

Dans notre exemple, prévoyez la méthode *getMagicKeys()* qui doit retourner la liste des clés contenues dans la table *magickey*.

Cette méthode devra appeler la méthode *findAll()* disponible dans le repository.

Récupérez une référence vers le repository correspondant par injection de dépendance (*@Autowired*).

Voici un début de code, à vous de le compléter.

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.spring.henallux.dataAccess.entity.MagicKeyEntity;
import com.spring.henallux.dataAccess.repository.MagicKeyRepository;

@Service
@Transactional
public class MagicKeyDAO {

    @Autowired
    private MagicKeyRepository magicKeyRepository;


    public ArrayList<String> getMagicKeys()
    {
         List <MagicKeyEntity> magicKeyEntities = magicKeyRepository.findAll();
```

Etape 8 – Adapter la classe *WelcomeController*

Adaptez la méthode *POST* de *WelcomeController* : faites-en sorte que le code magique introduit par l'utilisateur soit maintenant comparé aux codes (clés) de la base de données retournés par la méthode *getMagicKeys()* de la classe *MagicKeyDAO*.

Récupérez une référence vers une instance de *MagicKeyDAO* par injection de dépendance (utilisez l'annotation *@Autowired*).

Pour rappel, la méthode *contains* existe dans la classe *ArrayList*.

Etape 9 – Enregistrer l'utilisateur dans la base de données

Basez-vous sur les étapes précédentes pour enregistrer l'utilisateur dans la base de données dès qu'il a rempli le formulaire d'inscription.

Pour ce faire, créez la classe *UserEntity*, l'interface *UserRepository* et la classe *UserDAO* dans les bons packages.

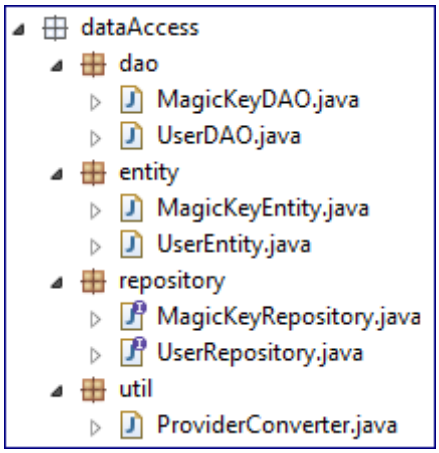
Comme vous devrez transformer un objet de type *User* en un objet de type *UserEntity* et inversement, prévoyez la classe *ProviderConverter* dans le package *util* qui proposera les méthodes qui s'en chargeront, à savoir :

public UserEntity userModelToUserEntity(User user)

public User userEntityToUserModel(UserEntity userEntity)

A vous de les implémenter !

Prévoyez dans *UserDAO* une méthode qui permet d'enregistrer un objet de type *User* dans la base de données : *public User save(User user)*



Adaptez la méthode *POST* de *InscriptionController* : appelez la méthode *save* de *UserDAO* pour enregistrer dans la base de données l'utilisateur correspondant au formulaire.