



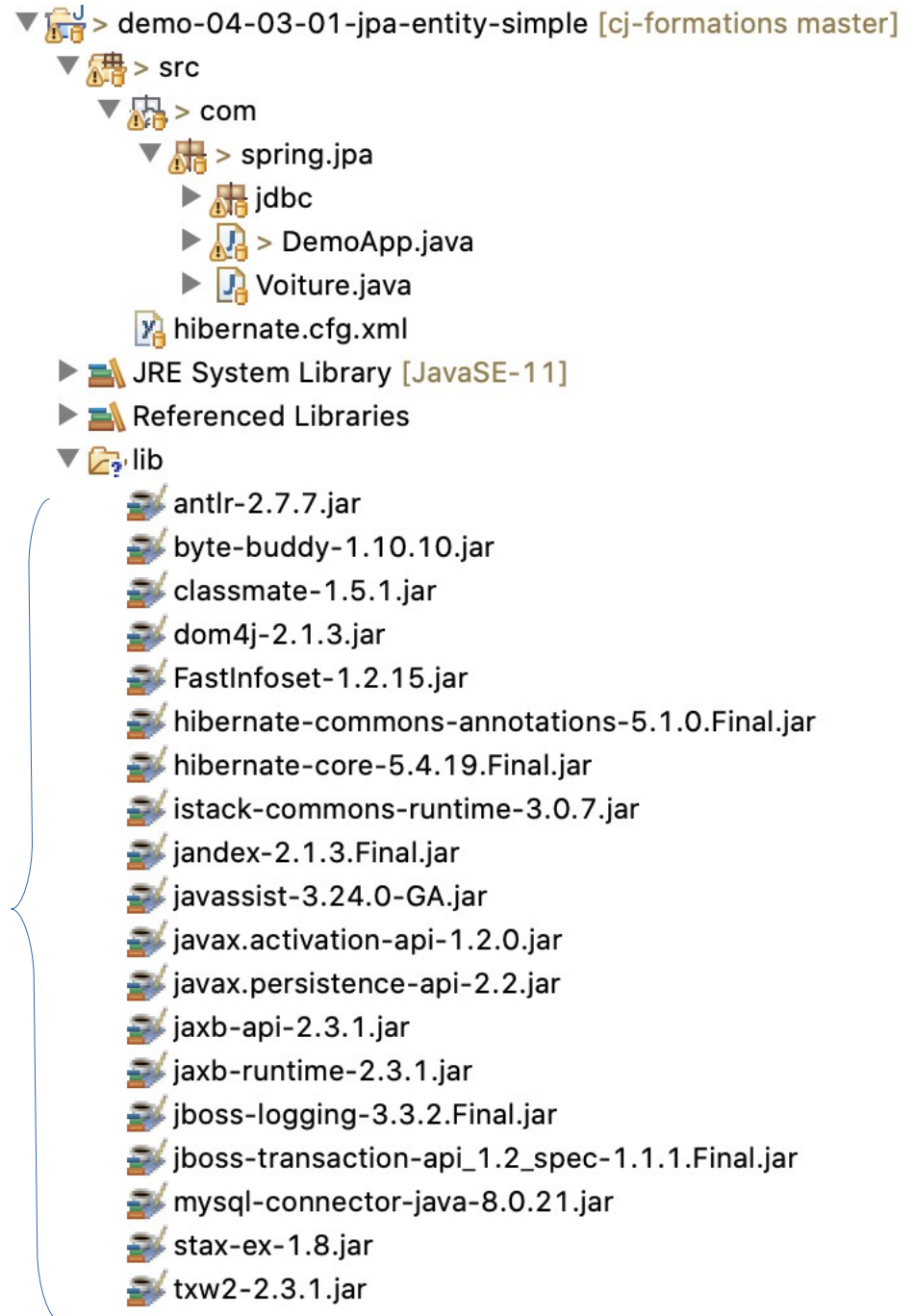
Hibernate - mapping

Apprendre par l'exemple

- Créons un projet java
- Plaçons les dépendances hibernate dans le classpath
- Ajoutons le fichier de configuration hibernate.cfg.xml
- Ajoutons le connecteur mysql au classpath
- Nous avons vu ces étapes précédemment dans l'introduction de cette section

Notre projet Hibernate

Les dépendances de
hibernate
+ connecteur MySQL



Hibernate.cfg.xml

src/hibernate-cfg.xml

```
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <!-- JDBC Database connection settings -->
        <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/voiture?
useSSL=false&serverTimezone=UTC</property>
        <property name="connection.username">padawan</property>
        <property name="connection.password">padawan</property>

        <!-- JDBC connection pool settings ... using built-in test pool -->
        <property name="connection.pool_size">1</property>

        <!-- Select our SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

        <!-- Echo the SQL to stdout -->
        <property name="show_sql">true</property>

        <!-- Set the current session context -->
        <property name="current_session_context_class">thread</property>

    </session-factory>

</hibernate-configuration>
```

Hibernate inititiation

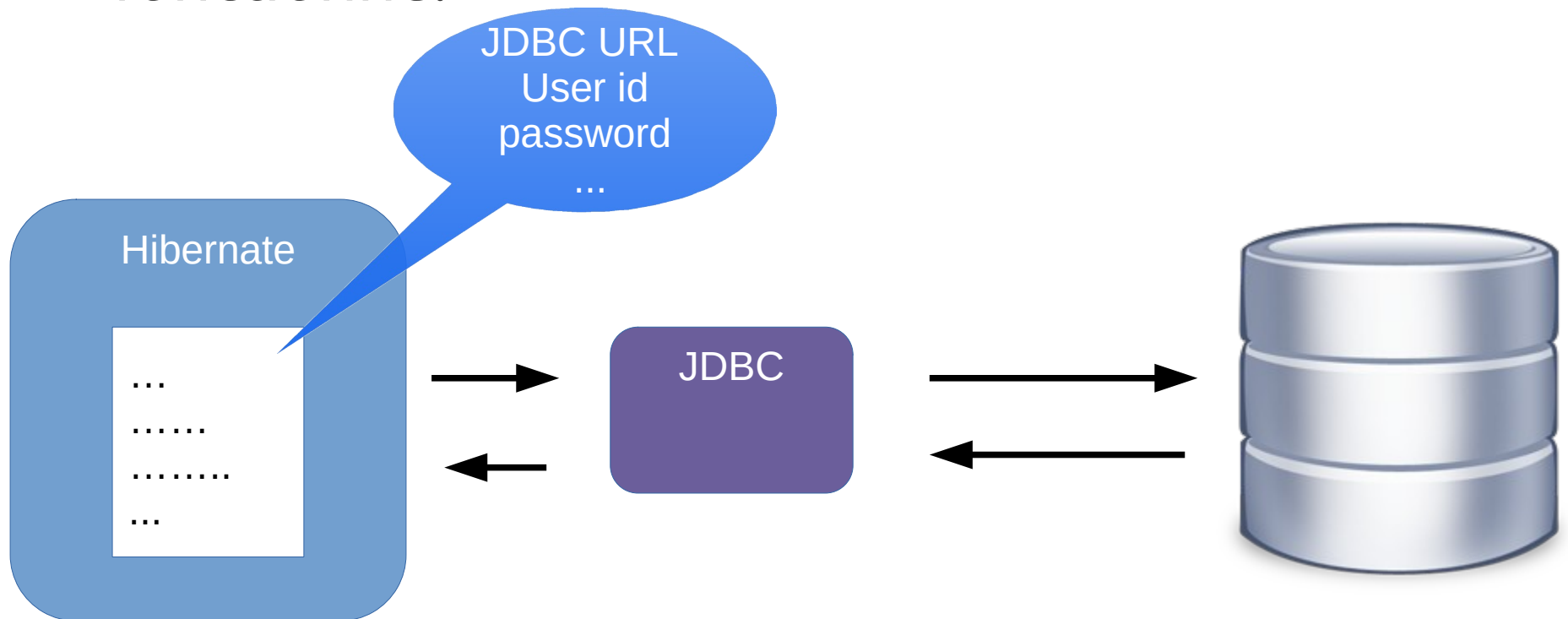
- On dispose d'un projet initial pour commencer à utiliser hibernate
- On doit maintenant réaliser le mapping entre une classe java et une Table de MySQL
- Ce mapping va s'effectuer avec des annotations (il est possible de le réaliser en xml mais cela devient rare)
- Nous allons créer et utiliser l'objet session d'hibernate pour écrire et lire les données en base.

Rappel

- L'objet session est lui même crée à partir d'un objet SessionFactory
- L'objet Session factory va recevoir la configuration de la connection à la bdd et la liste des classes qui seront mappées, donc gérées par l'orm hibernate.
- Session et SessionFactory sont fournies par hibernate. Elles sont définies dans les dépendances que nous avons ajouté au classpath.

Hibernate et jdbc

- En coulisse hibernate utilise jdbc. Rappelez-vous que pendant la création du SessionFactory on lui fournit les informations pour que le driver jdbc fonctionne.

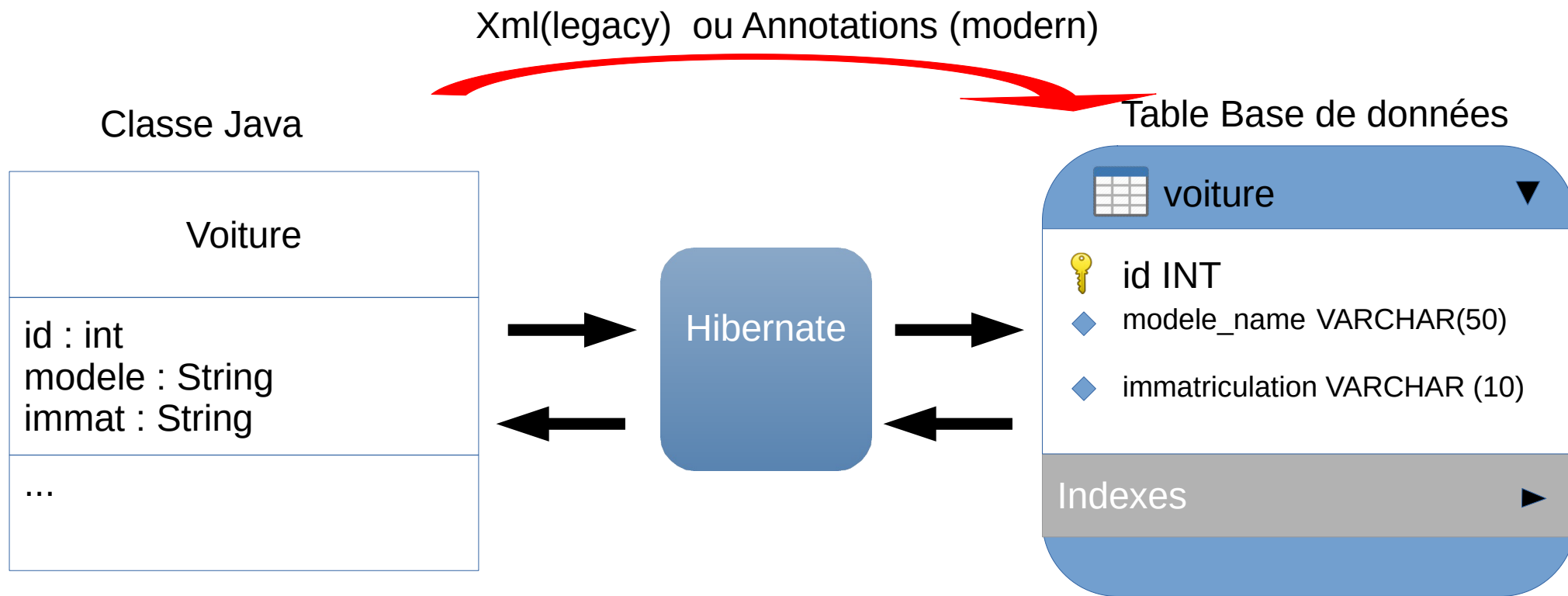


Annoter une classe java

- Hibernate introduit la notion de classe d'entité ("entity")
- Définition : une entité c'est une classe java qui est mappée vers une table.
- Il s'agit d'une classe java avec des champs , des getters et des setters dans laquelle on insère des annotations pour la faire correspondre à une table présente dans la base

ORM = objet relationnal mapping (EN)

ORM = objet table cartographe schématiser représenter (Fr)



Etape 1 : mapper la classe vers une table

Etape 2 : mapper les champs vers les colonnes de cette table

Etape 1 : "mapping" d'une classe vers une table.

```
@Entity
@Table(name="voiture")
public class Voiture {
    ...
    //getters & setters ...
}
```

@Table
est optionnel ici

Classe Java

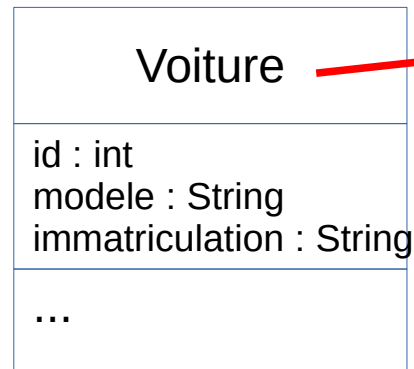
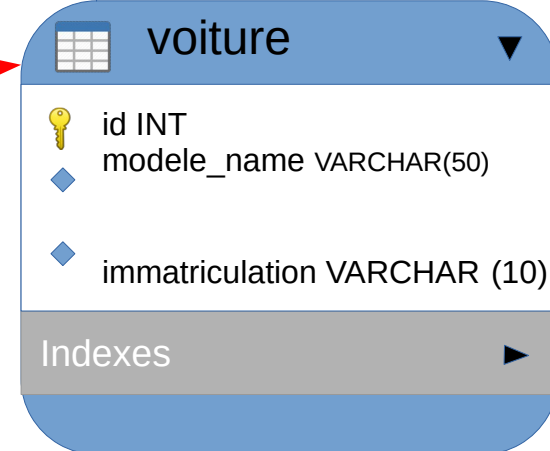


Table Base de données



Etape 2 : "mapping" des champs vers des colonnes

Si le nom du champ et de la table , de la colonne sont identiques

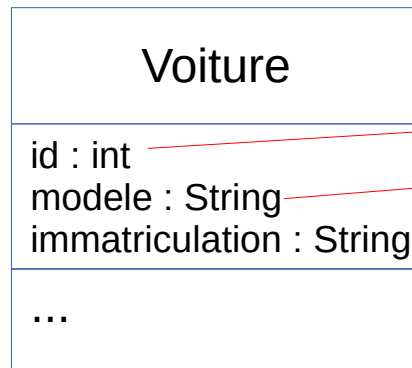
```
@Entity
@Table(name="voiture")
public class Voiture {

@Id
@Column(name="id")
private Long id ;

@Column(name="modele_name")
private String modele;

...
}
```

Classe Java

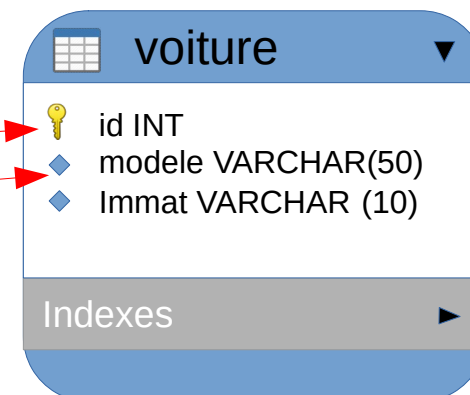


```
@Entity
public class Voiture {

@Id
@Column
private Long id ;

@Column
private String modele;

...
}
```



La classe Voiture

```
package com.spring.jpa;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="voiture")
public class Voiture {

    @Id
    @GeneratedValue(strategy=
        GenerationType.IDENTITY)
    @Column(name="id")
    private Long id ;

    @Column(name="modele_name")
    private String modele;

    private String immatriculation;
```

```
//getters & setters
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getModele() {
    return modele;
}

public void setModele(String modele) {
    this.modele = modele;
}

public String getImmat() {
    return immat;
}

public void setImmat(String immat) {
    this.immat = immat;
}

@Override
    public String toString() {
        return "Voiture [id=" + id + ", modele="
            + modele + ", immat=" + immat + "];"
    }
}
```

NB : on utilise des annotation JPA

- JPA est une spécification standard
- Hibernate une implémentation de cette spécification JPA
- Hibernate implémente toutes les annotations JPA
- L'équipe d'hibernate recommande l'usage des annotations JPA en tant que bonne pratique.

Persister un objet java avec hibernate

Class	Description
SessionFactory	<p>Lire le fichier de configuration d'hibernate</p> <p>Créer des objets session</p> <p>Il s'agit d'un objet dont la création mobilise des ressources</p> <p>On n'en créer qu'un seul dans toute l'application</p>
Session	<p>Contient (wrap) une connexion jdbc</p> <p>L'objet utile pour écrire et lire les objets</p> <p>C'est un objet dont la durée de vie est courte.</p> <p>Il est généré à partir d'un objet sessionFactory</p>

Créer un objet session

On peut lui donner un autre nom si on veut

```
public class DemoApp {  
  
    public static void main(String[] args) {  
  
        // create session factory  
        SessionFactory factory = new Configuration()  
            .configure("hibernate.cfg.xml")  
            .addAnnotatedClass(Voiture.class)  
            .buildSessionFactory();  
  
        // create session  
        Session session = factory.getCurrentSession();  
  
        try {  
            //utilisons l'objet session pour accéder à la bdd  
        } finally {  
            factory.close();  
        }  
    }  
}
```

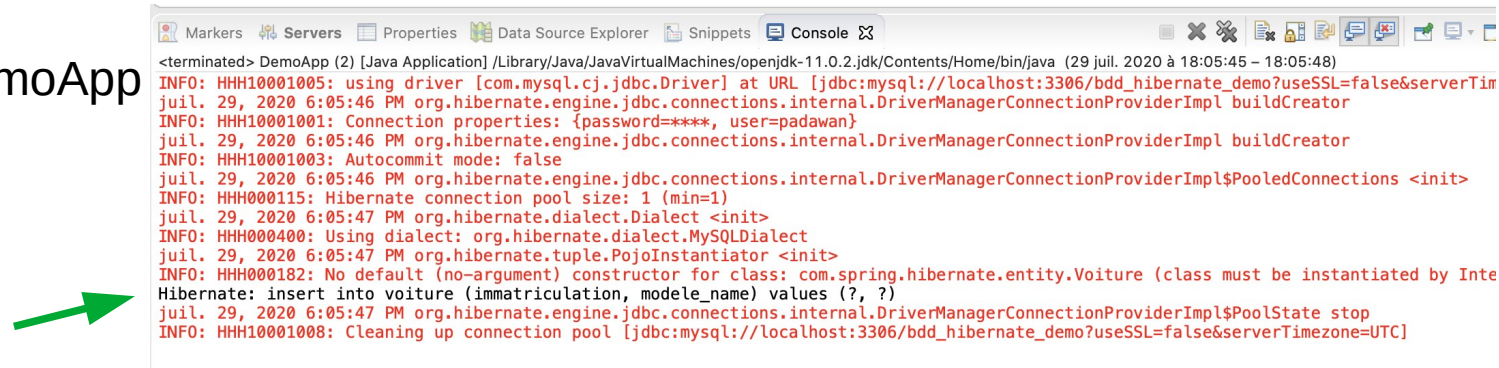
Par défaut , si on ne fournit pas cette ligne, hibernate cherche ce fichier par défaut avec ce nom ...

Insérer un objet Voiture en bdd

```
try {  
    Voiture voitureTemp= new Voiture("clio", "AK-725-66");  
  
    // debut de transaction  
    session.beginTransaction();  
  
    // save la voiture  
    session.save(voitureTemp);  
  
    // commit de la transaction  
    session.getTransaction().commit();  
  
} finally {  
    factory.close();  
}
```


Run the code – testons cet insertion avec hibernate

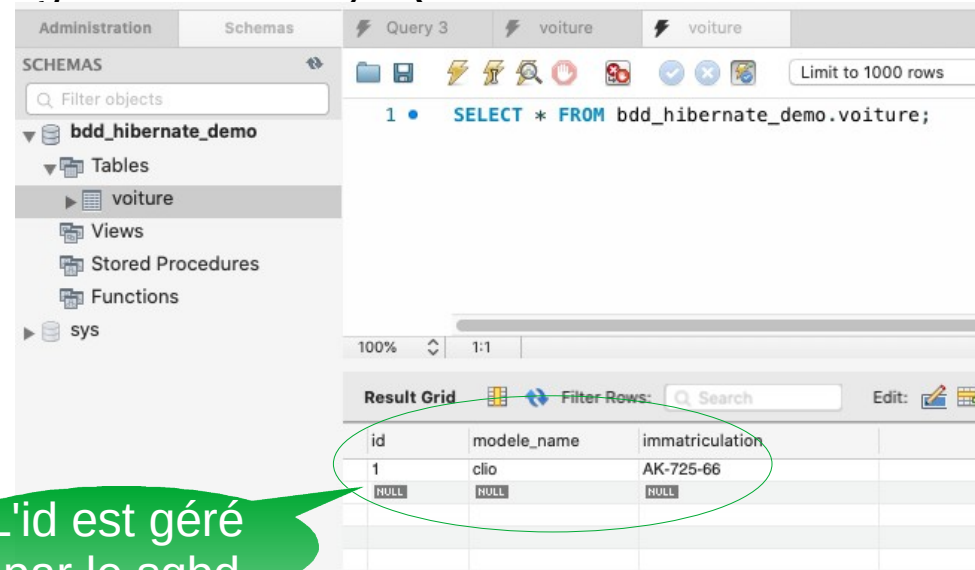
Exécutez la classe DemoApp



```
<terminated> DemoApp (2) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java (29 juil. 2020 à 18:05:45 – 18:05:48)
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/bdd_hibernate_demo?useSSL=false&serverTim
juil. 29, 2020 6:05:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {password=****, user=padawan}
juil. 29, 2020 6:05:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
juil. 29, 2020 6:05:46 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
juil. 29, 2020 6:05:47 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
juil. 29, 2020 6:05:47 PM org.hibernate.tuple.PojoInstantiator <init>
INFO: HHH000182: No default (no-argument) constructor for class: com.spring.hibernate.entity.Voiture (class must be instantiated by Inte
Hibernate: insert into voiture (immatriculation, modele_name) values (?, ?)
juil. 29, 2020 6:05:47 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_hibernate_demo?useSSL=false&serverTimezone=UTC]
```

On peut voir la requête SQL qui est envoyée par hibernate, quand l'insertion est réalisée.

Mais aussi on peut regarder avec MySQL Worbench le contenu de la table Voiture



id	modele_name	immatriculation
1	clió	AK-725-66
NULL	NULL	NULL

L'id est géré par le sgbd