



# Hibernate Relations

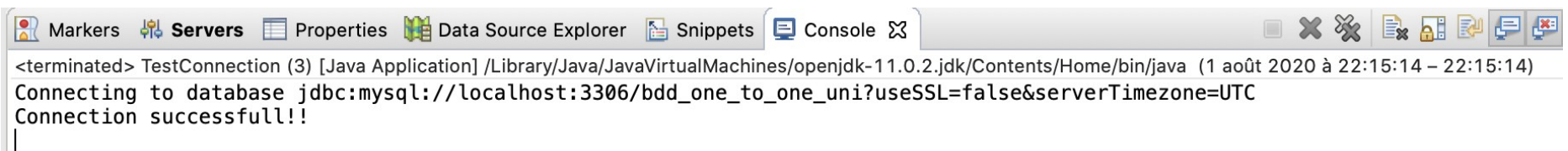

One to One Unidir - demo

# Préparez un nouveau projet

- Exécutez le script SQL `create.sql` si ce n'est pas déjà fait
- Spécifiez la configuration d'hibernate  
notamment le schéma `bdd_one_to_one_uni` dans  
l'adresse de connection ; login , password  
et les autres paramètres sont identiques
- Codez les classes `Voiture` et `Moteur` comme vu  
précédemment
- N'oubliez pas les constructeurs que nous utilisons et  
celui sans paramètre.
- N'oubliez pas les `getters()` & `setters()`

# On commence par exécuter une classe de test pour la connection, pour être sûr qu'on a bien configuré les choses à ce niveau

```
public class TestConnection {  
    public static void main(String[] args) {  
        String jdbcUrl="jdbc:mysql://localhost:3306/bdd_one_to_one_uni?useSSL=false&serverTimezone=UTC";  
        String user = "padawan";  
        String password = "padawan";  
        try {  
  
            System.out.println("Connecting to database " + jdbcUrl);  
            Connection connection = DriverManager.getConnection(jdbcUrl, user, password);  
            System.out.println("Connection successfull!!");  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
  
    }  
}
```



```
<terminated> TestConnection (3) [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java (1 août 2020 à 22:15:14 - 22:15:14)  
Connecting to database jdbc:mysql://localhost:3306/bdd_one_to_one_uni?useSSL=false&serverTimezone=UTC  
Connection successfull!!  
|
```

# Coder les "entities"

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

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

    @Column
    private String modele;

    @Column
    private String immatriculation;

    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name="moteur_id")
    private Moteur moteur;

    //getters() & setters() ...
    //Constructors

    public Voiture() {
        super();
    }

    public Voiture(String modele, String immatriculation) {
        super();
        this.modele = modele;
        this.immatriculation = immatriculation;
    }

    //toString . . .
}
```

```
@Entity
public class Moteur {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column
    private int id;

    @Column
    private int puissance;

    @Column
    private String carburant;

    @Column
    private int cylindree;

    //getters() & setters() ...
    //Constructors
    public Moteur() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Moteur(int puissance, String
carburant, int cylindree) {
        super();
        this.puissance = puissance;
        this.carburant = carburant;
        this.cylindree = cylindree;
    }

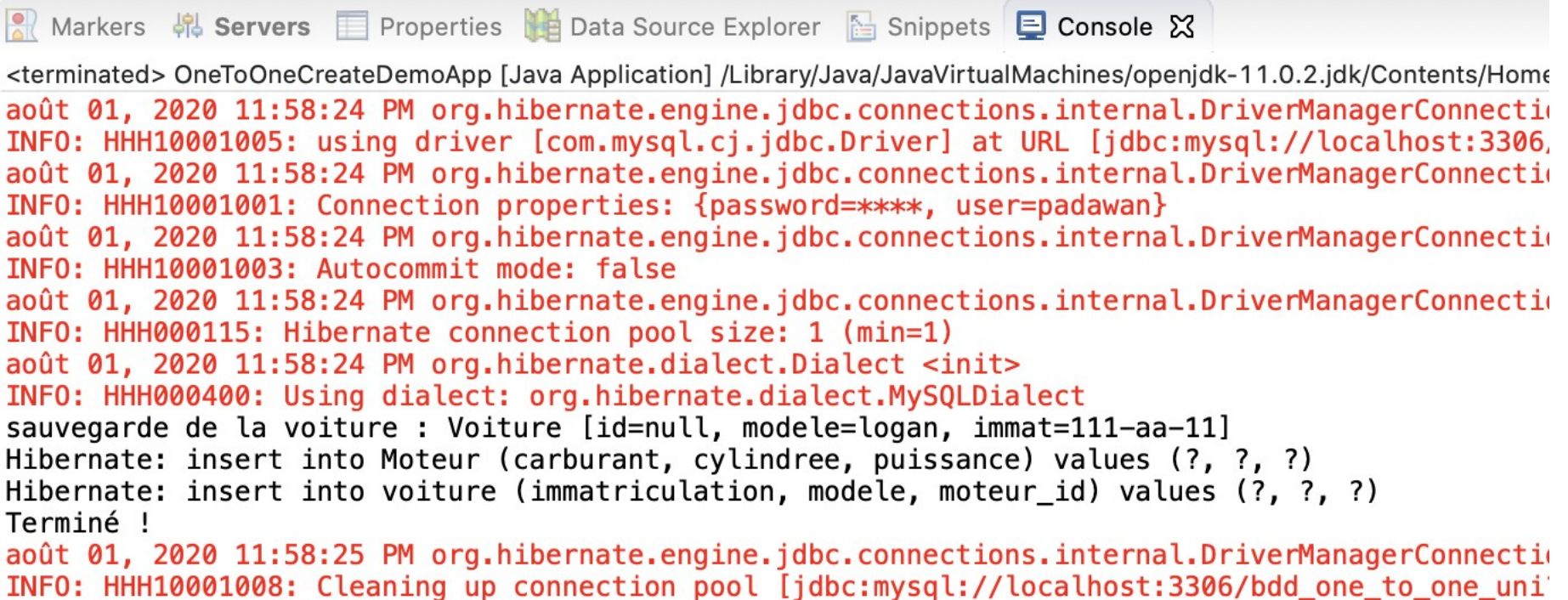
    //toString . . .
}
```

# main

```
public class OneToOneCreateDemoApp {  
  
    public static void main(String[] args) {  
  
        // create session factory  
        SessionFactory factory = new Configuration()  
            .configure("hibernate.cfg.xml")  
            .addAnnotatedClass(Voiture.class)  
            .addAnnotatedClass(Moteur.class)  
            .buildSessionFactory();  
  
        // déclarer session  
        Session session = null;  
  
        try {  
            // créer un objet voiture et un moteur  
            Voiture v= new Voiture ("logan","111-aa-11");  
            Moteur m = new Moteur(90, "essence", 1500);  
  
            //associer les deux objets  
            v.setMoteur(m);  
  
            // récupérer une session & ouvrir une transaction  
            session = factory.getCurrentSession();  
            session.beginTransaction();  
            System.out.println("sauvegarde de la voiture : " +v);  
            session.save(v);  
  
            //commit transaction  
            session.getTransaction().commit();  
            System.out.println("Terminé !");  
        } finally {  
            factory.close();  
        }  
    }  
}
```

On ne  
sauvegarde que  
la voiture.  
Le Moteur sera  
sauvegardé aussi  
en cascade en  
arrière plan

# On peut exécuter le code



```
<terminated> OneToOneCreateDemoApp [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home
août 01, 2020 11:58:24 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnection
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/
août 01, 2020 11:58:24 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnection
INFO: HHH10001001: Connection properties: {password=****, user=padawan}
août 01, 2020 11:58:24 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnection
INFO: HHH10001003: Autocommit mode: false
août 01, 2020 11:58:24 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnection
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 01, 2020 11:58:24 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
sauvegarde de la voiture : Voiture [id=null, modele=logan, immat=111-aa-11]
Hibernate: insert into Moteur (carburant, cylindree, puissance) values (?, ?, ?)
Hibernate: insert into voiture (immatriculation, modele, moteur_id) values (?, ?, ?)
Terminé !
août 01, 2020 11:58:25 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnection
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_one_to_one_uni
```

# Coté bdd on vérifie les insertions

Table voiture

	id	modele	immatriculation	moteur_id	
▶	1	logan	111-aa-11	1	
	NULL	NULL	NULL	NULL	

Table moteur

	id	puissance	carburant	cylindree	^
▶	1	90	essence	1500	
	NULL	NULL	NULL	NULL	

OK

Refaisons la manipulation et créons d'autres voitures ..

```
...  
Voiture v= new Voiture ("Quashqaï", "222-bb-22");  
Moteur m = new Moteur(90, "diesel", 2200);  
  
//associer les deux objets  
v.setMoteur(m);  
  
// récupérer une session & ouvrir une transaction  
session = factory.getCurrentSession();  
session.beginTransaction();  
session.save(v);  
...
```

Executer

	id	modele	immatriculation	moteur_id	
▶	1	logan	111-aa-11	1	
	2	Quashqaï	222-bb-22	2	
	NULL	NULL	NULL	NULL	

	id	puissance	carburant	cylindree	
▶	1	90	essence	1500	
	2	90	diesel	2200	
	NULL	NULL	NULL	NULL	

OK

# Peut-on doubler la mise ?

Essayons de supprimer une voiture ...

Créons une classe exécutable DeleteDemo.java

Repartons avec un cadre : hibernate + transaction prêt à accueillir nos opérations avec la bdd.

```
public class OneToOneDeleteDemoApp {  
    public static void main(String[] args) {  
        . . .  
        try {  
            // récupérer une session & ouvrir une transaction  
            session = factory.getCurrentSession();  
            session.beginTransaction();  
  
            //commit transaction  
            session.getTransaction().commit();  
            . . .  
        }  
    }  
}
```



# Coder la suppression

```
try {  
  
    // récupérer une session & ouvrir une transaction  
    session = factory.getCurrentSession();  
    session.beginTransaction();  
  
    // lire une voiture avec son id  
    Long id=1L;  
    Voiture v = session.get(Voiture.class, id);  
  
    System.out.println("Voiture trouvée : " + v);  
  
    // supprimer la voiture  
    if(v!=null) {  
        System.out.println("Suppression de la voiture : "+v);  
        session.delete(v);  
    }  
  
    //commit transaction  
    session.getTransaction().commit();  
}
```

Retournera null  
si non trouvé

Supprimera aussi  
le moteur, car on  
Cascade l'opération  
Delete

```
public class Voiture {  
    . . .  
    @OneToOne(cascade=CascadeType.ALL)  
    @JoinColumn(name="moteur_id")  
    private Moteur moteur;  
}
```

```

août 02, 2020 12:36:46 AM org.hibernate.engine.jdbc.connections.internal.DriverMar
INFO: HHH10001003: Autocommit mode: false
août 02, 2020 12:36:46 AM org.hibernate.engine.jdbc.connections.internal.DriverMar
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 02, 2020 12:36:46 AM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select voiture0_.id as id1_1_0_, voiture0_.immatriculation as immatric2
Voiture trouvée : Voiture [id=1, modele=logan, immat=111-aa-11]
Suppression de la voiture : Voiture [id=1, modele=logan, immat=111-aa-11]
Hibernate: delete from voiture where id=?
Hibernate: delete from Moteur where id=?
Terminé !
août 02, 2020 12:36:47 AM org.hibernate.engine.jdbc.connections.internal.DriverMar
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_or

```



	id	modele	immatriculation	moteur_id	
▶	2	Quashqaï	222-bb-22	2	
	NULL	NULL	NULL	NULL	

	id	puissance	carburant	cylindree	
	2	90	diesel	2200	
	NULL	NULL	NULL	NULL	

OK