

Hibernate mapping

Primary keys

Primary keys

- Rappel :

Une clé primaire identifie chaque ligne d'une table de manière unique.

- La colonne d'une clé primaire ne peut contenir de valeur NULL
- Quand nous avons créé la table via un script SQL pour MySQL, l'id a été affublé du mot clé "AUTO_INCREMENT"

```
CREATE TABLE `voiture` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `modele_name` varchar(45) DEFAULT NULL,  
  `immatriculation` varchar(10) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
)
```



MySQL va gérer
l'incrémentation
automatiquement

Hibernate Identity – Primary Key

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

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



Les autres strategies de gestion des PK

Nom	Description
GenerationType.AUTO	La génération de la clé primaire est laissée à l'implémentation. C'est hibernate qui s'en charge et qui crée une séquence unique sur tout le schéma via la table hibernate_sequence.
GenerationType.IDENTITY	La génération de la clé primaire se fera à partir d'une Identité propre au SGBD. Il utilise un type de colonne spéciale à la base de données.
GenerationType.SEQUENCE	La valeur est fournie par une sequence maintenue dans le SGBD, auquel on ajoutera l'attribut generator.
GenerationType.TABLE	La génération de la clé primaire se fera en utilisant une table dédiée hibernate_sequence qui stocke les noms et les valeurs des séquences.

Tester une petite manipulation

- Commentez le @GeneratedValue de l'entity Voiture
- Dans DemoApp , donnez un id à la voitureTemp , par ex : 2L
- Sauvegardez, réexecutez => insertion successfull (vérifiez Workbench)
- Refaites une execution , avec un autre id (c'est vous qui devez le gérer) par ex= 28L
- Sauvegardez, réexecutez => insertion successfull (vérifiez Workbench)
- Essayez d'insérer à nouveau une voitureTemp avec le même id . Vous obtenez une belle erreur avec un message explicite, (rassurant)
- Maintenant décommentez @GeneratedValue dans Voiture.java et dans DemoApp.java créez une voitureTemp sans id, sauvegardez là .
- MySQL a généré un id , mais lequel ? A t-il repris à 2 , à 3 , ou à 29 ?

Réponse

Result Grid   Filter Rows: <input type="text" value="Search"/>			
	id	modele_name	immatriculation
▶	1	clio	AK-725-66
	2	twingo	AB-542-13
	28	golf	ZA-028-13
	29	golf	ZA-028-13
	NULL	NULL	NULL

Autre manipulation



- Créez plusieurs voitureTemp dans DemoApp (3 par exemple)
- Et sauvegardez les toutes les trois successivement comme ci-après et exécutez :

```
try {  
    Voiture voitureTemp1= new Voiture("picasso", "CDF-548-13");  
    Voiture voitureTemp2= new Voiture("countach", "ZE-022-13");  
    Voiture voitureTemp3= new Voiture("kangoo", "Zg-047-84");  
  
    // debut de transaction  
    session.beginTransaction();  
  
    // save la voiture  
    session.save(voitureTemp1);  
    session.save(voitureTemp2);  
    session.save(voitureTemp3);  
  
    // commit de la transaction  
    session.getTransaction().commit();  
  
} finally {  
    factory.close();  
}
```

On vérifie la gestion de l'id par le SGBD

```
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
juil. 29, 2020 7:40:07 PM org.hibernate.tuple.PojoInstantiator <init>
INFO: HHH000182: No default (no-argument) constructor for class: com.spring.hibernate.er
Hibernate: insert into voiture (immatriculation, modele_name) values (?, ?)
Hibernate: insert into voiture (immatriculation, modele_name) values (?, ?)
Hibernate: insert into voiture (immatriculation, modele_name) values (?, ?)
juil. 29, 2020 7:40:08 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerCo
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_hibernate]
```

Auto-increment
A bien géré notre id

Result Grid   Filter Rows: <input type="text" value="Search"/>			
	id	modele_name	immatriculation
▶	1	clio	AK-725-66
	2	twingo	AB-542-13
	28	golf	ZA-028-13
	29	golf	ZA-028-13
	30	picasso	CDF-548-13
	31	countach	ZE-022-13
	32	kangoo	Zg-047-84
	NULL	NULL	NULL

Pour refixer l'auto-increment on utilisera un script SQL : par exemple
`ALTER TABLE bdd_hibernate_demo.voiture AUTO_INCREMENT=200`

Pour reseter la table , auto-increment à 1

Revenir à 1 sur l'id signifie supprimer les lignes existantes (sinon il y aura incompatibilité des nouvelles lignes) :

Exécutez le script SQL est le suivant :


```
truncate bdd_hibernate_demo.voiture
```





The screenshot displays a database management interface. On the left, the 'Administration' tab is active, showing a tree view of 'SCHEMAS'. Under 'bdd_hibernate_demo', the 'Tables' folder is expanded, and 'voiture' is selected. The main area shows 'Query 3' with the SQL command: `SELECT * FROM bdd_hibernate_demo.voiture;`. Below the query, the 'Result Grid' is visible, showing a single row with three columns: 'id', 'modele_name', and 'immatriculation'. All three columns in the first row contain the value 'NULL'. A blue oval highlights the first row of the result grid.

Réexécutons la dernière manipulation (insérons 3 voitures dans la table Voiture)

On laisse le SGBD gérer l'Id :



Result Grid   Filter Rows: <input type="text" value="Search"/>			
	id	modele_name	immatriculation
▶	1	picasso	CDF-548-13
	2	countach	ZE-022-13
	3	kangoo	Zg-047-84
	NULL	NULL	NULL