



# Jpa relations fetchtype

Eager démo

# Créez un nouveau projet

- Créer une bdd en modifiant le script create-db.sql et recréez-y notre schéma
- Récupérez les deux classes executables CreateVoitureDemo.java , et CreateInterventionDemo.java
- Executez ces deux applications dans l'ordre , pour peupler le nouveau schéma
- Maintenant créons notre classe executable pour illustrer le chargement Eager et Lazy

# On obtient le résultat en bdd suivant

	id	dateheure	prix	titre	technicien	voiture_id	
▶	1	2020-10-10 08:00:00	80.50	Petite Révision	A. Didonk	1	
	2	2019-12-05 13:30:00	20.00	Changement des pneus avant	M. Imome	1	
	3	2017-06-21 06:45:00	80.50	Vidange	S. Ouraille	1	
	NULL	NULL	NULL	NULL	NULL	NULL	

	id	modele	immatriculati...	moteur_id	
▶	1	clio	444-ddd-44	1	
	NULL	NULL	NULL	NULL	

# Eager Fetching demo

```
public class EagerLazyDemo{

    public static void main(String[] args) {
        . . .

        // récupérer une session & ouvrir une transaction
        session = factory.getCurrentSession();
        session.beginTransaction();

        //récupérer une voiture
        Long id= 1L;
        Voiture v = session.get(Voiture.class , id);

        // afficher la voiture
        System.out.println("_____voiture : "+v);

        //afficher les Interventions associées
        System.out.println("_____interventions : "+v.getInterventions());

        //commit transaction
        session.getTransaction().commit();
        System.out.println("_____Terminé !");

        . . .
    }
}
```

# Dans Voiture , modifions le fetchType vers Interventions

```
@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;

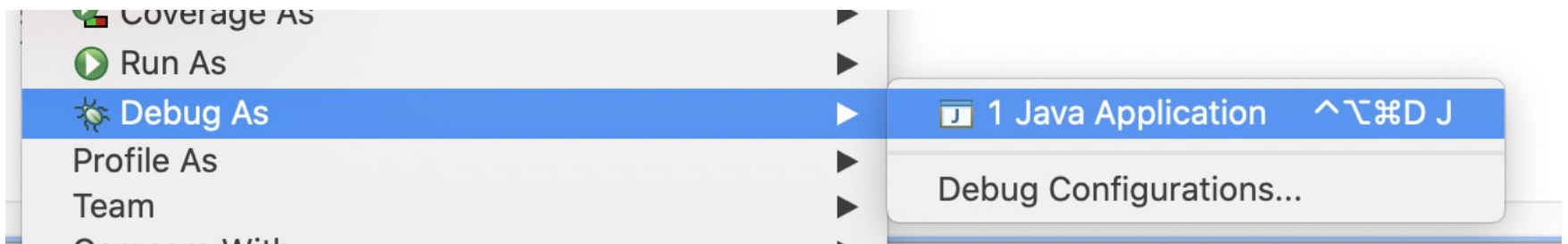
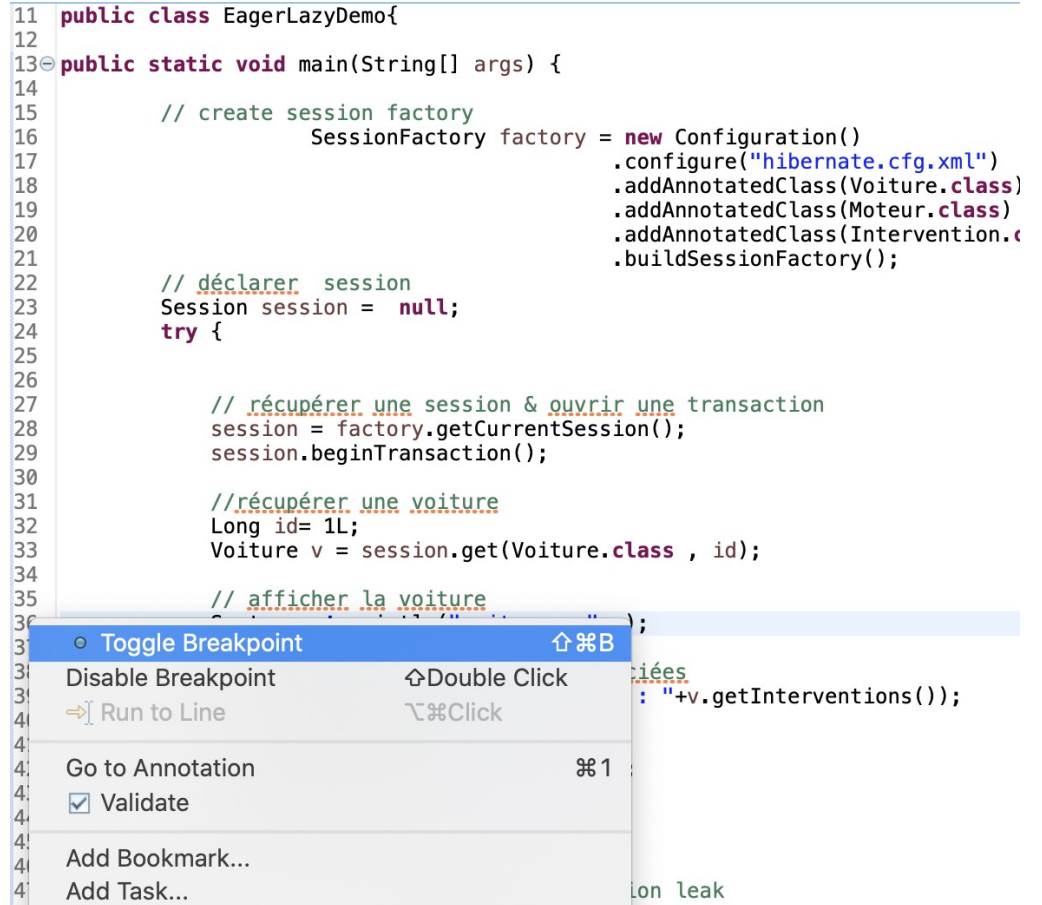
    @OneToMany(fetch=FetchType.EAGER, mappedBy="voiture" ,
               cascade= {CascadeType.PERSIST, CascadeType.MERGE,
                         CascadeType.DETACH, CascadeType.REFRESH})
    private List<Intervention> interventions;

    . . .
}
```

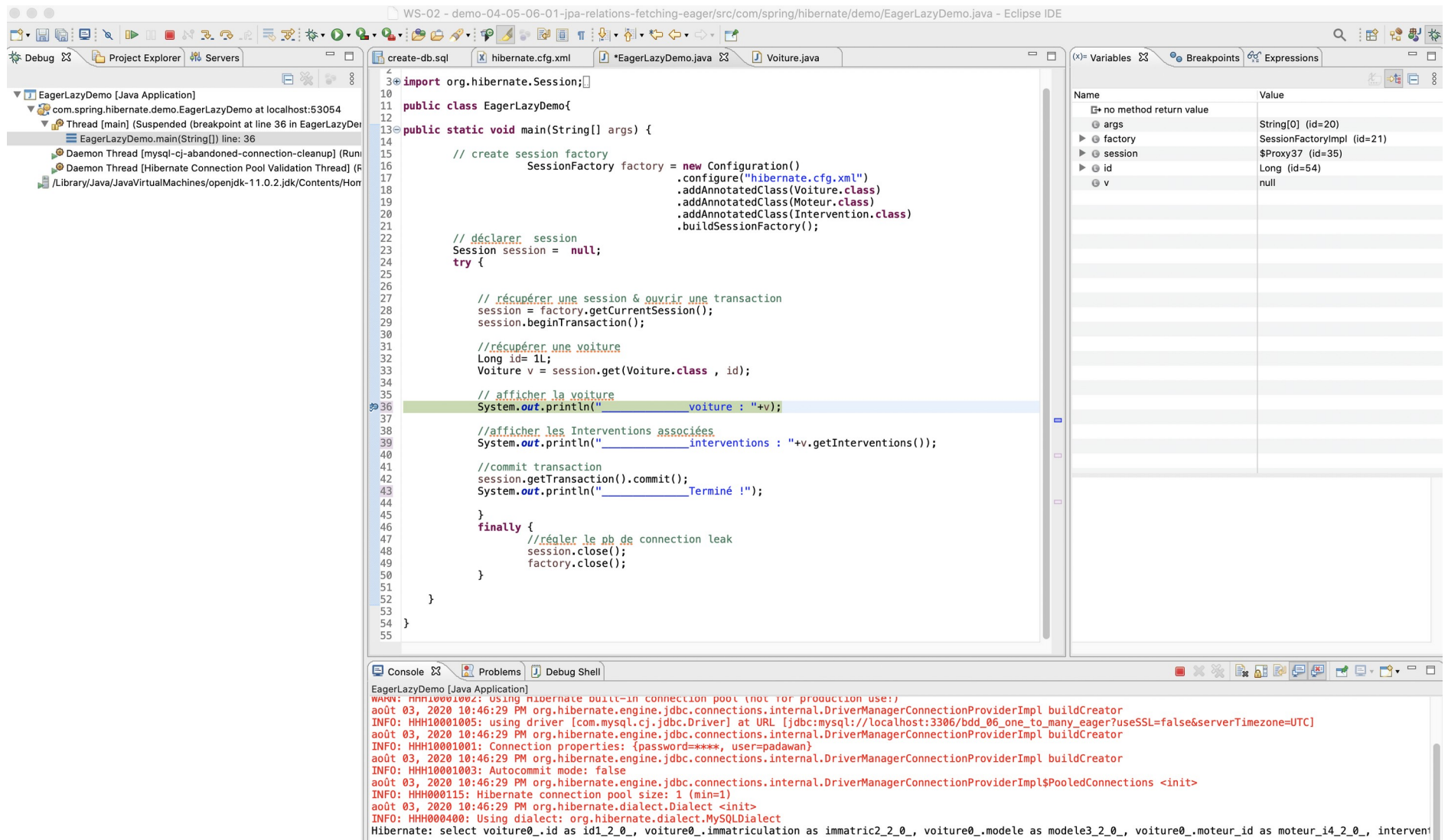
# Posons un point d'arrêt ligne 36 de EagerLazyDemo .java

Executer en mode debug

```
11 public class EagerLazyDemo{
12
13 public static void main(String[] args) {
14
15     // create session factory
16     SessionFactory factory = new Configuration()
17         .configure("hibernate.cfg.xml")
18         .addAnnotatedClass(Voiture.class)
19         .addAnnotatedClass(Moteur.class)
20         .addAnnotatedClass(Intervention.class)
21         .buildSessionFactory();
22
23     // déclarer session
24     Session session = null;
25     try {
26
27         // récupérer une session & ouvrir une transaction
28         session = factory.getCurrentSession();
29         session.beginTransaction();
30
31         // récupérer une voiture
32         Long id= 1L;
33         Voiture v = session.get(Voiture.class , id);
34
35         // afficher la voiture
36         System.out.println(v);
37     } catch (Exception e) {
38         e.printStackTrace();
39     }
40 }
```



# Passez en perspective debug



The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer:** Shows the project structure for "EagerLazyDemo [Java Application]". The main class is "com.spring.hibernate.demo.EagerLazyDemo".
- Source Editor:** Displays the code for "EagerLazyDemo.java". The code is in French and uses Hibernate for database operations. It includes comments like "créer une session", "récupérer une voiture", and "afficher les interventions associées".
- Variables View:** Shows the current state of the program. The variables are: "args" (String[]), "factory" (SessionFactoryImpl), "session" (\$Proxy37), "id" (Long), and "v" (null).
- Console:** Shows the output of the application. It includes logs from Hibernate and the application's own output, such as "Terminé !".

```
import org.hibernate.Session;

public class EagerLazyDemo {

    public static void main(String[] args) {

        // créer une session
        Session factory = new Configuration()
            .configure("hibernate.cfg.xml")
            .addAnnotatedClass(Voiture.class)
            .addAnnotatedClass(Moteur.class)
            .addAnnotatedClass(Intervention.class)
            .buildSessionFactory();

        // déclarer session
        Session session = null;
        try {

            // récupérer une session & ouvrir une transaction
            session = factory.getCurrentSession();
            session.beginTransaction();

            // récupérer une voiture
            Long id= 1L;
            Voiture v = session.get(Voiture.class, id);

            // afficher la voiture
            System.out.println("voiture : "+v);

            // afficher les interventions associées
            System.out.println("interventions : "+v.getInterventions());

            // commit transaction
            session.getTransaction().commit();
            System.out.println("Terminé !");

        } finally {
            // régler le pb de connection leak
            session.close();
            factory.close();
        }
    }
}
```

Console Output:

```
EagerLazyDemo [Java Application]
WARN: hbm10001002: using hibernate built-in connection pool (not for production use!)
aout 03, 2020 10:46:29 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/bdd_06_one_to_many_eager?useSSL=false&serverTimezone=UTC]
aout 03, 2020 10:46:29 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {password=****, user=padawan}
aout 03, 2020 10:46:29 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
aout 03, 2020 10:46:29 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
aout 03, 2020 10:46:29 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select voiture0_.id as id1_2_0_, voiture0_.immatriculation as immatric2_2_0_, voiture0_.modele as modele3_2_0_, voiture0_.moteur_id as moteur_id_4_2_0_, interven
```

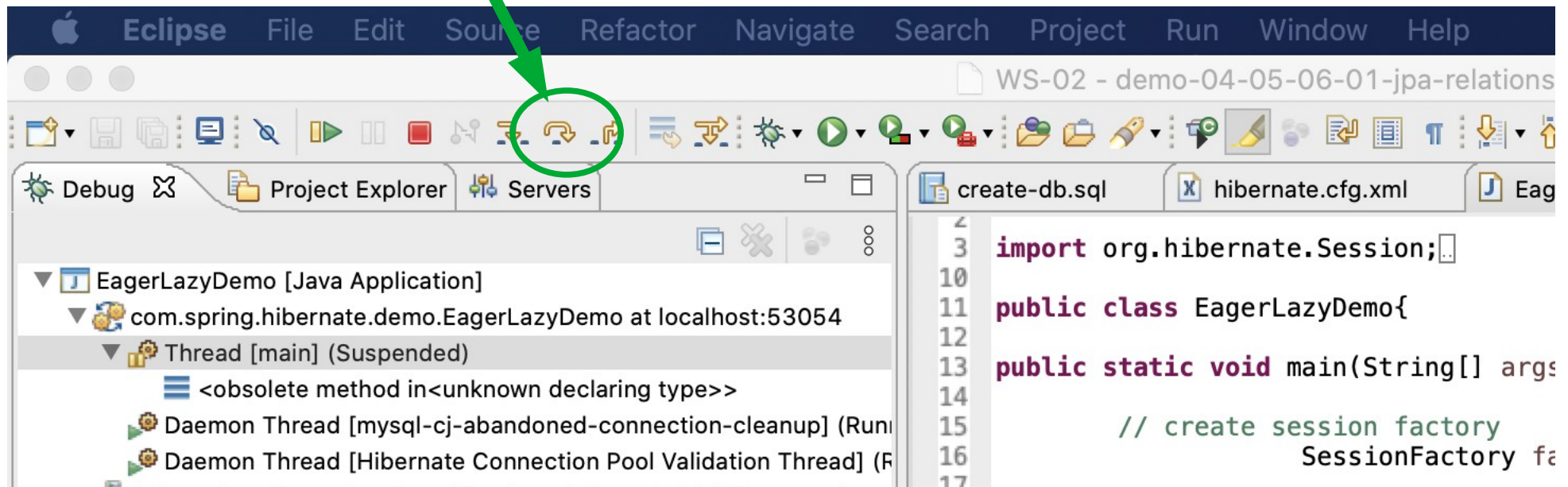
# Quand Eclipse s'arrête sur le point d'arrêt

```
10
11 public class EagerLazyDemo{
12
13     public static void main(String[] args) {
14
15         // create session factory
16         SessionFactory factory = new Configuration()
17             .configure("hibernate.cfg.xml")
18             .addAnnotatedClass(Voiture.class)
19             .addAnnotatedClass(Moteur.class)
20             .addAnnotatedClass(Intervention.class)
21             .buildSessionFactory();
22
23         // déclarer session
24         Session session = null;
25         try {
26
27             // récupérer une session & ouvrir une transaction
28             session = factory.getCurrentSession();
29             session.beginTransaction();
30
31             //récupérer une voiture
32             Long id= 1L;
33             Voiture v = session.get(Voiture.class , id);
34
35             // afficher la voiture
36             System.out.println("_____voiture : "+v);
37
38             //afficher les Interventions associées
39             System.out.println("_____interventions : "+v.getInterventions());
40
41             //commit transaction
42             session.getTransaction().commit();
43             System.out.println("_____Terminé !");
44         }
45     }
46 }
```

Après la ligne 33  
Toutes les  
données sont  
présentes grâce à  
notre fetchType  
eager  
Voiture et  
Interventions sont  
déjà récupérées



# Avancer en mode debug d'un step (step over)



```

34
35 // afficher la voiture
36 System.out.println("_____voiture : "+v);
37
38 //afficher les Interventions associées
39 System.out.println("_____interventions : "+v.getInterventions());
40
41 //commit transaction
42 session.getTransaction().commit();
43 System.out.println("_____Terminé !");
44
45 }
46 finally {
47     //régler le pb de connection leak
48     session.close();
49     factory.close();
50 }
51
52 }
53
54 }
55

```

## EagerLazyDemo [Java Application]

```
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/hibernate?serverTimezone=UTC]
août 03, 2020 11:06:23 PM org.hibernate.engine.jdbc.connections.internal.DriverManager
INFO: HHH10001001: Connection properties: {password=****, user=padawan}
août 03, 2020 11:06:23 PM org.hibernate.engine.jdbc.connections.internal.DriverManager
INFO: HHH10001003: Autocommit mode: false
août 03, 2020 11:06:23 PM org.hibernate.engine.jdbc.connections.internal.DriverManager
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 03, 2020 11:06:23 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select voiture0_.id as id1_2_0_, voiture0_.immatriculation as immatric2_2_0_, voiture0_.marque as marque3_2_0_ from voiture0_ where id=1
voiture : Voiture [id=1, modele=clio, immat=444-ddd-44]
```

Ici les interventions apparaissent dans les logs car elles sont chargées en eager

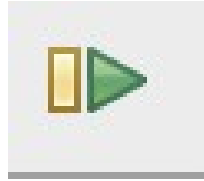


## Toutes les infos sont présentes

## EagerLazyDemo [Java Application]

```
INFO: HHH10001003: Autocommit mode: false
août 03, 2020 11:06:23 PM org.hibernate.engine.jdbc.connections
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 03, 2020 11:06:23 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select voiture0_.id as id1_2_0_, voiture0_.immatricu
_____voiture : Voiture [id=1, modele=clio, immat=444-d
_____interventions : [ Intervention [id=1, dateheure=20
, Intervention [id=2, dateheure=2019-12-05T14:30, titre=Changer
, Intervention [id=3, dateheure=2017-06-21T08:45, titre=Vidang
```

**Terminer l'exécution  
debug**



**Revenir en perspective  
Java**



# Modifions quelques éléments de configuration

- Passons le techType en LAZY dans voiture sur la relation vers les Interventions
- Les interventions vont être chargées uniquement si on le demande désormais

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

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

    @OneToMany(fetch=FetchType.LAZY, mappedBy="voiture",
               cascade= {CascadeType.PERSIST, CascadeType.MERGE,
                         CascadeType.DETACH, CascadeType.REFRESH})
    private List<Intervention> interventions;

    . . .
}
```

# EagerLazyDemo.java (no change)

```
public class EagerLazyDemo{

    public static void main(String[] args) {
        . . .

        // récupérer une session & ouvrir une transaction
        session = factory.getCurrentSession();
        session.beginTransaction();

        //récupérer une voiture
        Long id= 1L;
        Voiture v = session.get(Voiture.class , id);

        // afficher la voiture
        System.out.println("_____voiture : "+v);

        //afficher les Interventions associées
        System.out.println("_____interventions : "+v.getInterventions());

        //commit transaction
        session.getTransaction().commit();
        System.out.println("_____Terminé !");

        . . .
    }
}
```

Get Interventions  
(lazy fetch)

Executer en mode debug

- 100

# les interventions ne sont pas chargées

```
Voiture v = session.get(Voiture.class, id);

// afficher la voiture
System.out.println("voiture : "+v);

//afficher les Interventions associées
System.out.println("interventions : "+v.getInterventions());

//commit transaction
session.getTransaction().commit();
System.out.println("Terminé !");

}
finally {
    //régler le pb de connection leak
    session.close();
    factory.close();
}

}

}
```

Console

```
EagerLazyDemo [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java
août 03, 2020 11:33:00 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProvider
INFO: HHH10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/ma_base?useSSL=false&serverTimezone=UTC]
août 03, 2020 11:33:00 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProvider
INFO: HHH10001001: Connection properties: {password=****, user=padawan}
août 03, 2020 11:33:00 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProvider
INFO: HHH10001003: Autocommit mode: false
août 03, 2020 11:33:00 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProvider
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 03, 2020 11:33:01 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select voiture0_.id as id1_2_0_, voiture0_.immatriculation as immatric2_2_0_, voiture0_.modele as modele3_2_0_ from voiture where voiture0_.id=?
```

Ici les interventions ne sont pas chargées en Lazy (chargement à la demande)

- Avancer d'un step  → La voiture est affichée en console (sans contenir les Interventions)
- Avancer encore d'un step 



# Partir de la ligne 39

Les informations deviennent complètes car on appelle explicitement la liste des interventions liées à l'objet voiture qu'on manipule

Appel explicite  
Le fetchtype Lazy  
va accéder aux  
données dans  
ce second temps

```
33      Voiture v = session.get(Voiture.class, id);
34
35      // afficher la voiture
36      System.out.println("_____voiture : "+v);
37
38      //afficher les Interventions associées
39      System.out.println("_____interventions : "+v.getInterventions());
40
41      //commit transaction
42      session.getTransaction().commit();
43      System.out.println("_____Terminé !");
44
45  }
46  finally {
47      //régler le pb de connection leak
48      session.close();
49      factory.close();
50  }
51
52  }
53
54  }
55  }
```

Console

EagerLazyDemo [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java (3 août 2020 à 23:32:58)

août 03, 2020 11:33:00 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl\$Po  
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)

août 03, 2020 11:33:01 PM org.hibernate.dialect.Dialect <init>  
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect

Hibernate: select voiture0\_.id as id1\_2\_0\_, voiture0\_.immatriculation as immatric2\_2\_0\_, voiture0\_.modele as mo  
\_\_\_\_\_voiture : Voiture [id=1, modele=clio, immat=444-ddd-44]

Hibernate: select interventi0\_.voiture\_id as voiture\_6\_0\_0\_, interventi0\_.id as id1\_0\_0\_, interventi0\_.id as id  
\_\_\_\_\_interventions : [ Intervention [id=1, dateheure=2020-10-10T10:00, titre=Petite Révision, prix=80.  
, Intervention [id=2, dateheure=2019-12-05T14:30, titre=Changement des pneus avant, prix=20.0, technicien=M. I  
, Intervention [id=3, dateheure=2017-06-21T08:45, titre=Vidange, prix=80.5, technicien=S. Ouraille]

On va mettre en évidence la limite du FetchType Lazy en fermant la session avant l'appel explicite aux interventions

```
public class EagerLazyDemo{

    public static void main(String[] args) {
        . . .

        // récupérer une session & ouvrir une transaction
        session = factory.getCurrentSession();
        session.beginTransaction();

        //récupérer une voiture
        Long id= 1L;
        Voiture v = session.get(Voiture.class , id);

        // afficher la voiture
        System.out.println("_____voiture : "+v);

        //afficher les Interventions associées
        System.out.println("_____interventions : "+v.getInterventions());

        //commit transaction
        session.getTransaction().commit();

        System.out.println("_____Terminé !");
        . . .
    }
}
```

Lazy data



# Fermons la session

```
public class EagerLazyDemo{

    public static void main(String[] args) {
        . . .

        // récupérer une session & ouvrir une transaction
        session = factory.getCurrentSession();
        session.beginTransaction();

        //récupérer une voiture
        Long id= 1L;
        Voiture v = session.get(Voiture.class , id);

        // afficher la voiture
        System.out.println("_____voiture : "+v);

        //commit transaction
        session.getTransaction().commit();

        // fermer la session
        session.close();

        //afficher les Interventions associées
        System.out.println("_____interventions : "+v.getInterventions());

        System.out.println("_____Terminé !");
        . . .
    }
}
```

Lazy Loading doit échouer

Et lever une exception

## Exception in thread "main"

### org.hibernate.LazyInitializationException

Console Problems Debug Shell

```
<terminated> EagerLazyDemo [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java (4 août 2020 à 00:01:39 – 00:01:41)
août 04, 2020 12:01:41 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState sto
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_06_one_to_many_eager?useSSL=false&serverTi
Exception in thread "main" org.hibernate.LazyInitializationException: failed to lazily initialize a collection of role: co
    at org.hibernate.collection.internal.AbstractPersistentCollection.throwLazyInitializationException(AbstractPersist
    at org.hibernate.collection.internal.AbstractPersistentCollection.withTemporarySessionIfNeeded(AbstractPersistentC
    at org.hibernate.collection.internal.AbstractPersistentCollection.initialize(AbstractPersistentCollection.java:585)
    at org.hibernate.collection.internal.AbstractPersistentCollection.read(AbstractPersistentCollection.java:149)
    at org.hibernate.collection.internal.PersistentBag.toString(PersistentBag.java:621)
    at java.base/java.lang.String.valueOf(String.java:2951)
    at java.base/java.lang.StringBuilder.append(StringBuilder.java:168)
    at com.spring.hibernate.demo.EagerLazyDemo.main(EagerLazyDemo.java:47)
```

La ligne précise de l'erreur dans mon code et le nom du fichier

Nous avons fermé la session avant d'accéder aux données Lazy

La trace d'erreur nous indique qu'il n'a pas pu initialiser une collection, indique laquelle, et la raison

com.spring.hibernate.entity.Voiture.interventions, could not initialize proxy – no Session

FetchType.LAZY

# Solution : accéder aux data en Lazy Loading

- Solution 1 : appeler le getter pendant que la session est ouverte

Nous pouvons réordonner notre EagerLazyDemo.java

```
public static void main(String[] args) {  
    . . .  
    //récupérer une voiture  
    . . .  
  
    // afficher la voiture  
    System.out.println("_____voiture : "+v);  
  
    //accès en lazy loading aux Interventions associées  
    System.out.println("_____interventions : "+v.getInterventions());  
  
    //commit transaction  
    session.getTransaction().commit();  
  
    // fermer la session  
    session.close();  
  
    //afficher les Interventions associées  
    System.out.println("_____interventions : "+v.getInterventions());  
    . . .  
}
```

Lazy data  
Récupérée à la  
demande

Maintenant la  
session  
est fermée

Ce code va fonctionner

# 1ère solution

```
août 04, 2020 2:23:33 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$P
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 04, 2020 2:23:33 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select voiture0_.id as id1_2_0_, voiture0_.immatriculation as immatric2_2_0_, voiture0_.modele as
voiture : Voiture [id=1, modele=clio, immat=444-ddd-44]
Hibernate: select interventi0_.voiture_id as voiture_6_0_0_, interventi0_.id as id1_0_0_, interventi0_.id as
interventions : [ Intervention [id=1, dateheure=2020-10-10T10:00, titre=Petite Révision, prix=8
, Intervention [id=2, dateheure=2019-12-05T14:30, titre=Changement des pneus avant, prix=20.0, technicien=M.
, Intervention [id=3, dateheure=2017-06-21T08:45, titre=Vidange, prix=80.5, technicien=S. Ouraille]
]
interventions : [ Intervention [id=1, dateheure=2020-10-10T10:00, titre=Petite Révision, prix=8
, Intervention [id=2, dateheure=2019-12-05T14:30, titre=Changement des pneus avant, prix=20.0, technicien=M.
, Intervention [id=3, dateheure=2017-06-21T08:45, titre=Vidange, prix=80.5, technicien=S. Ouraille]
]
Terminé !
août 04, 2020 2:23:34 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$P
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_06_one_to_many_eager?useSSL=f
```

Cette partie est celle qui est exécutée pendant que la session est ouverte

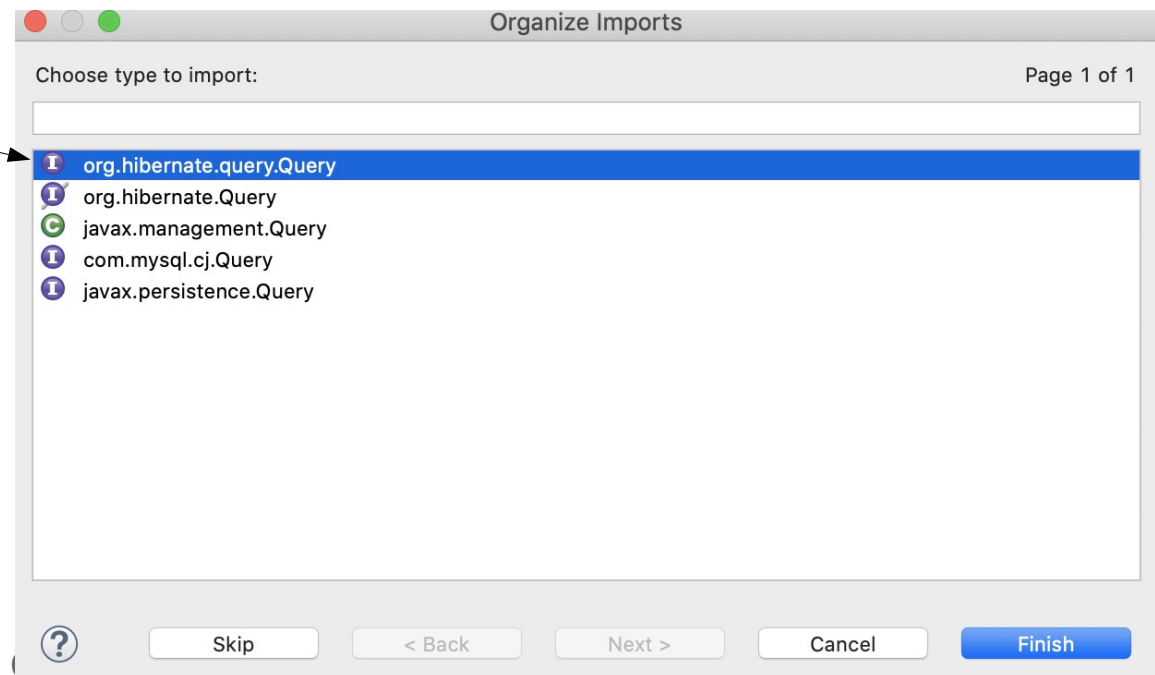
# 2nde Solution : Requête HQL

- Copiez coller EagerLazyDemo.java
- renommez-la en FetchJoinDemo.java
- Faites un peu de nettoyage

```
public class FetchJoinDemo{  
    ...  
    //récupérer une voiture  
    Long id= 1L;  
  
    // afficher la voiture  
    System.out.println("_____voiture : "+v);  
  
    //commit transaction  
    session.getTransaction().commit();  
  
    // fermer la session  
    session.close();  
    System.out.println("\n_____Session est fermée \n");  
}
```

# Solution 2 : hibernate HQL query

```
try {  
  
    // récupérer une session & ouvrir une transaction  
    session = factory.getCurrentSession();  
    session.beginTransaction();  
  
    //récupérer une voiture  
    Long id= 1L;  
    Query <Voiture> q = session.createQuery("select v from Voiture v", Voiture.class);
```



# Paramétrer la requête HQL

```
try {
```

```
    // récupérer une session & ouvrir une transaction  
    session = factory.getCurrentSession();  
    session.beginTransaction();
```

```
    //récupérer une voiture
```

```
    Long id= 1L;
```

```
    Query <Voiture> query = session.createQuery(" select v from Voiture v "  
                                                +"JOIN FETCH v.interventions "  
                                                +"where v.id=:unId",  
                                                Voiture.class);
```

```
    // set parameter on query
```

```
    query.setParameter("unId", id);
```

# Executer la query HQL

```
try {  
  
    // récupérer une session & ouvrir une transaction  
    session = factory.getCurrentSession();  
    session.beginTransaction();  
  
    //récupérer une voiture  
    Long id= 1L;  
    Query <Voiture> query = session.createQuery(" select v from Voiture v "  
                                                +"JOIN FETCH v.interventions "  
                                                +"where v.id=:unId",  
                                                Voiture.class);  
  
    // set parameter on query  
    query.setParameter("unId", id);  
  
    // executer la requête et récupérer une voiture  
    Voiture v = query.getSingleResult();
```

Charge la voiture et toutes les interventions en un seul accès



# Afficher le résultat en console

```
try {

    // récupérer une session & ouvrir une transaction
    session = factory.getCurrentSession();
    session.beginTransaction();

    //récupérer une voiture
    Long id= 1L;
    Query <Voiture> query = session.createQuery(" select v from Voiture v "
                                                +"JOIN FETCH v.interventions "
                                                +"where v.id=:unId",
                                                Voiture.class);

    // set parameter on query
    query.setParameter("unId", id);

    // executer la requête et récupérer une voiture
    Voiture v = query.getSingleResult();

    // afficher la voiture
    System.out.println("_____voiture : "+v);

    //commit transaction
    session.getTransaction().commit();

    // fermer la session
    session.close();
    System.out.println("\n_____Session est  fermée \n");

    //afficher les Interventions associées
    System.out.println("_____interventions : "+v.getInterventions());
    System.out.println("_____Terminé !");

}
```

# Affichage - Monitoring

```
INFO: HHH000115: Hibernate connection pool size: 1 (min=1)
août 04, 2020 4:12:26 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
Hibernate: select voiture0_.id as id1_2_0_, interventil_.id as id1_0_1_, voiture0_.immatriculation as immatric2_2_0_, voiture0_.modele as m
Hibernate: select moteur0_.id as id1_1_0_, moteur0_.carburant as carburan2_1_0_, moteur0_.cylindree as cylindre3_1_0_, moteur0_.puissance
Hibernate: select voiture0_.id as id1_2_1_, voiture0_.immatriculation as immatric2_2_1_, voiture0_.modele as modele3_2_1_, voiture0_.moteu
voiture : Voiture [id=1, modele=clio, immat=444-ddd-44]

Session est fermée

interventions : [ Intervention [id=1, dateheure=2020-10-10T10:00, titre=Petite Révision, prix=80.5, technicien=A. Didonk]
, Intervention [id=2, dateheure=2019-12-05T14:30, titre=Changement des pneus avant, prix=20.0, technicien=M. Imome]
, Intervention [id=3, dateheure=2017-06-21T08:45, titre=Vidange, prix=80.5, technicien=S. Ouraille]
]

terminé !
août 04, 2020 4:12:27 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/bdd_06_one_to_many_eager?useSSL=false&serverTimezone=UTC]
```

La session est fermée ici.  
On n'a pas fait le get pendant la session.  
C'est la requête HQL qui a fait la jointure.  
Et on a les interventions dans l'objet voiture  
dans notre application.

# On contourne le problème

- Dans mes expériences , on va plutôt gérer ce problème de données manquantes avec une nouvelle session et une requête sur les interventions avec un filtre sur la voiture liée...

```
// get interventions for a given voiture
Query<Intervention> query = session.createQuery("select i from Intervention i "
                                                + "where i.voiture.id=:theVoitureId",
                                                Intervention.class);

query.setParameter("theVoitureId", id);
List<Intervention> interventions = query.getResultList();
System.out.println("interventions: " + interventions);
voiture.setInterventions(interventions);
```