무료 전자 책

배우기

hibernate

Free unaffiliated eBook created from
Stack Overflow contributors.

#hibernate

1:

`SessionFactory` `TransactionManager` , , . DAO `SessionFactory` autowire .

Hibernate " ?". `TransactionManager` `SesisonFactory` `SesisonFactory` . `@Transactional` . . . `SessionFactory` .

. 5 `SessionFactory` (`SELECT COUNT(*)`) 5 . .

4.2.0	http://hibernate.org/orm/documentation/4.2/	2013-03-01
4.3.0	http://hibernate.org/orm/documentation/4.3/	2013-12-01
5.0.0	http://hibernate.org/orm/documentation/5.0/	2015-09-01

Examples

XML Hibernate

classpath `database-servlet.xml` .

:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-
tx-3.2.xsd">

</beans>
```

`tx` `jdbc` `tx` . .

(`@Transactional`) . Spring Hibernate Spring . .

```
<tx:annotation-driven />
```

. Hibernate . . Hibernate , .

```
<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="" />
```

```
    <property name="url" value="" />
    <property name="username" value="" />
    <property name="password" value="" />
</bean>
```

bean `javax.sql.DataSource` . Spring . Apache Commons `org.apache.commons.dbcp.BasicDataSource` . .

- *driverClassName* : JDBC . JAR. . Oracle OracleDriver . MySQL MySQLDriver . . Google .
- *url* : URL. `jdbc\:oracle\:thin\:\path\to\your\database` `jdbc:mysql://path/to/your/database` . Google . `org.hibernate.HibernateException: Connection cannot be null when 'hibernate.dialect' not set` HibernateException `org.hibernate.HibernateException: Connection cannot be null when 'hibernate.dialect' not set` `org.hibernate.HibernateException: Connection cannot be null when 'hibernate.dialect' not set` URL 5 % / 5 % .
- *username* : .
- : .

`SessionFactory` . Hibernate , . .

```
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="packagesToScan" value="au.com.project />
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.use_sql_comments">true</prop>
            <prop key="hibernate.hbm2ddl.auto">validate</prop>
        </props>
    </property>
</bean>
```

- *dataSource* : . dataSource ID .
- *packagesToScan* : JPA . , POJO `@Entity` . .
- *annotatedClasses* (not shown) : Hibernate Hibernate . `packagesToScan` `annotatedClasses` . .

```
<property name="annotatedClasses">
    <list>
        <value>foo.bar.package.model.Person</value>
        <value>foo.bar.package.model.Thing</value>
    </list>
</property>
```

- *hibernateProperties* : . :
- *hibernate.hbm2ddl.auto* : . . validate SQL (in-memory) ( ) .
- *hibernate.show_sql* : . Hibernate SQL `stdout` . `log4j.logger.org.hibernate.type=TRACE`

`log4j.logger.org.hibernate.SQL=DEBUG` (log4j .).

- *hibernate.format_sql* : Boolean Hibernate SQL stdout .
- *hibernate.dialect* ( ) : Hibernate Dialect . Hibernate JDBC . 3 5 MySQL . Hibernate .

bean .

```
<bean class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"
    id="PersistenceExceptionTranslator" />

<bean id="transactionManager"
    class="org.springframework.orm.hibernate4.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
```

`PersistenceExceptionTranslator` `HibernateException` `SQLExceptions` Spring .

`TransactionManager` .

: DAO `SessionFactory` bean autowiring.

XML-less Hibernate

.

```
package com.reborne.SmartHibernateConnector.utils;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class LiveHibernateConnector implements IHibernateConnector {

    private String DB_DRIVER_NAME = "";
    private String DB_URL = "jdbc:h2:~/liveDB;MV_STORE=FALSE;MVCC=FALSE";
    private String DB_USERNAME = "sa";
    private String DB_PASSWORD = "";
    private String DIALECT = "org.hibernate.dialect.H2Dialect";
    private String HBM2DLL = "create";
    private String SHOW_SQL = "true";

    private static Configuration config;
    private static SessionFactory sessionFactory;
    private Session session;

    private boolean CLOSE_AFTER_TRANSACTION = false;

    public LiveHibernateConnector() {

        config = new Configuration();

        config.setProperty("hibernate.connector.driver_class",         DB_DRIVER_NAME);
        config.setProperty("hibernate.connection.url",                 DB_URL);
        config.setProperty("hibernate.connection.username",         DB_USERNAME);
        config.setProperty("hibernate.connection.password",         DB_PASSWORD);
```

```
        config.setProperty("hibernate.dialect",                    DIALECT);
        config.setProperty("hibernate.hbm2dll.auto",                HBM2DLL);
        config.setProperty("hibernate.show_sql",                   SHOW_SQL);

        /*
         * Config connection pools
         */

        config.setProperty("connection.provider_class",
"org.hibernate.connection.C3P0ConnectionProvider");
        config.setProperty("hibernate.c3p0.min_size", "5");
        config.setProperty("hibernate.c3p0.max_size", "20");
        config.setProperty("hibernate.c3p0.timeout", "300");
        config.setProperty("hibernate.c3p0.max_statements", "50");
        config.setProperty("hibernate.c3p0.idle_test_period", "3000");


        /**
         * Resource mapping
         */

//        config.addAnnotatedClass(User.class);
//        config.addAnnotatedClass(User.class);
//        config.addAnnotatedClass(User.class);

        sessionFactory = config.buildSessionFactory();
    }


    public HibWrapper openSession() throws HibernateException {
        return new HibWrapper(getOrCreateSession(), CLOSE_AFTER_TRANSACTION);
    }


    public Session getOrCreateSession() throws HibernateException {
        if (session == null) {
            session = sessionFactory.openSession();
        }
        return session;
    }

    public void reconnect() throws HibernateException {
        this.sessionFactory = config.buildSessionFactory();
    }


}
```

Hibernate . (Hibernate 5.2 release)

XML

XML hibernate.cfg.xml, POJO, EntityName.hbm.xml . MySQL .

hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
```

```
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>
   <property name="hibernate.dialect">
      org.hibernate.dialect.MySQLDialect
   </property>
   <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
   </property>

   <property name="hibernate.connection.url">
      jdbc:mysql://localhost/DBSchemaName
   </property>
   <property name="hibernate.connection.username">
      testUserName
   </property>
   <property name="hibernate.connection.password">
      testPassword
   </property>

   <!-- List of XML mapping files -->
   <mapping resource="HibernatePractice/Employee.hbm.xml"/>

</session-factory>
</hibernate-configuration>
```

DBSchemaName, testUserName testPassword . .

Employee.java

```
package HibernatePractice;

public class Employee {
    private int id;
    private String firstName;
    private String middleName;
    private String lastName;

    public Employee(){

    }
    public int getId(){
        return id;
    }
    public void setId(int id){
        this.id = id;
    }
    public String getFirstName(){
        return firstName;
    }
    public void setFirstName(String firstName){
        this.firstName = firstName;
    }
    public String getMiddleName(){
        return middleName;
    }
    public void setMiddleName(String middleName){
        this.middleName = middleName;
```

```
    }
    public String getLastName(){
        return lastName;
    }
    public void setLastName(String lastName){
        this.lastName = lastName;
    }
}
```

Employee.hbm.xml

```
<hibernate-mapping>
   <class name="HibernatePractice.Employee" table="employee">
      <meta attribute="class-description">
         This class contains employee information.
      </meta>
      <id name="id" type="int" column="empolyee_id">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="middleName" column="middle_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
   </class>
</hibernate-mapping>
```

, packageName.className .

.

: https://riptutorial.com/ko/hibernate/topic/907/---

2: Criterias and Projections

Examples

City "title" TravelReview .

```
 Criteria criteria =
    session.createCriteria(TravelReview.class);
  List review =
    criteria.add(Restrictions.eq("title", "Mumbai")).list();
  System.out.println("Using equals: " + review);
```

.

```
List reviews = session.createCriteria(TravelReview.class)
   .add(Restrictions.eq("author", "John Jones"))
   .add(Restrictions.between("date",fromDate,toDate))
   .add(Restrictions.ne("title","New York")).list();
```

Projections . , title .

```
 // Selecting all title columns
  List review = session.createCriteria(TravelReview.class)
        .setProjection(Projections.property("title"))
    .list();
  // Getting row count
  review = session.createCriteria(TravelReview.class)
    .setProjection(Projections.rowCount())
    .list();
  // Fetching number of titles
  review = session.createCriteria(TravelReview.class)
    .setProjection(Projections.count("title"))
    .list();
```

@Filter WHERE . .

```
@Entity
@Table(name = "Student")
public class Student
{
    /*...*/

    @OneToMany
    @Filter(name = "active", condition = "EXISTS(SELECT * FROM Study s WHERE state = true and
s.id = study_id)")
    Set<StudentStudy> studies;

    /* getters and setters methods */
}
```

```
@Entity
@Table(name = "Study")
```

```
@FilterDef(name = "active")
@Filter(name = "active", condition="state = true")
public class Study
{
    /*...*/

    @OneToMany
    Set<StudentStudy> students;

    @Field
    boolean state;

    /* getters and setters methods */
}
```

StudentStudy Entity

```
@Entity
@Table(name = "StudentStudy")
@Filter(name = "active", condition = "EXISTS(SELECT * FROM Study s WHERE state = true and s.id
= study_id)")
public class StudentStudy
{
    /*...*/

    @ManytoOne
    Student student;

    @ManytoOne
    Study study;

    /* getters and setters methods */
}
```

"" ,

- `state = true` `state = true`

- `state = true` `state = true`

StudentStudy entiy -Every `state = true`

Pls study_id SQL StudentStudy .

Criterias and Projections : https://riptutorial.com/ko/hibernate/topic/3939/criterias-and-projections

3: HQL

HQL Hibernate Query Language, SQL SQL . / . .

hql SQL .

Examples

```
hql = "From EntityName";
```

```
hql = "Select id, name From Employee";
```

Where

```
hql = "From Employee where id = 22";
```

```
hql = "From Author a, Book b Where a.id = book.author";
```

HQL : https://riptutorial.com/ko/hibernate/topic/9388/hql

4: SQL /

, Hibernate . . . Hibernate .

Examples

.

```
# log the sql statement
org.hibernate.SQL=DEBUG
# log the parameters
org.hibernate.type=TRACE
```

.

Log4j :

```
log4j.logger.org.hibernate.SQL=DEBUG
log4j.logger.org.hibernate.type=TRACE
```

Spring `application.properties` :

```
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type=TRACE
```

`logback.xml` :

```
<logger name="org.hibernate.SQL" level="DEBUG"/>
<logger name="org.hibernate.type" level="TRACE"/>
```

SQL .

```
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
    <property name="hibernateProperties">
        <props>
            <!-- show the sql without the parameters -->
            <prop key="hibernate.show_sql">true</prop>
            <!-- format the sql nice -->
            <prop key="hibernate.format_sql">true</prop>
            <!-- show the hql as comment -->
            <prop key="use_sql_comments">true</prop>
        </props>
    </property>
</bean>
```

SQL /

Hibernate SQL . SQL / .

IDE .

```
org.apache.log4j.Logger.getLogger("org.hibernate.SQL")
    .setLevel(org.apache.log4j.Level.DEBUG)
```

.

```
org.apache.log4j.Logger.getLogger("org.hibernate.SQL")
    .setLevel(org.apache.log4j.Level.OFF)
```

SQL / : https://riptutorial.com/ko/hibernate/topic/3548/sql-------

5:

Examples

.

Hibernate pom.xml dependencies .

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.1.Final</version>
</dependency>
```

1.

:

Eager Loading . ( ).

Lazy Loading . .

2.

Lazy XML .

```
lazy="true"
```

. User .

```
public class User implements Serializable {

    private Long userId;
    private String userName;
    private String firstName;
    private String lastName;
    private Set<OrderDetail> orderDetail = new HashSet<>();

    //setters and getters
    //equals and hashcode
    }
```

OrderDetail . **OrderDetail** .

```
public class OrderDetail implements Serializable {

    private Long orderId;
    private Date orderDate;
    private String orderDesc;
    private User user;
```

```
    //setters and getters
    //equals and hashcode
}
```

`UserLazy.hbm.xml` `UserLazy.hbm.xml` .

```
<set name="orderDetail" table="USER_ORDER" inverse="true" lazy="true" fetch="select">
    <key>
        <column name="USER_ID" not-null="true" />
    </key>
   <one-to-many class="com.baeldung.hibernate.fetching.model.OrderDetail" />
</set>
```

. . `lazy = "false"` . User.hbm.xml eager loading .

```
<set name="orderDetail" table="USER_ORDER" inverse="true" lazy="false" fetch="select">
    <key>
        <column name="USER_ID" not-null="true" />
    </key>
   <one-to-many class="com.baeldung.hibernate.fetching.model.OrderDetail" />
</set>
```

, **Session** of *SessionFactory* . *Eager* . , . / . . POJO .

```
@Entity
public class User {
    private int userId;
    private String username;
    @OneToMany
    private Set<Page> likedPage;

    // getters and setters here
}

@Entity
public class Page{
    private int pageId;
    private String pageURL;

    // getters and setters here
}

public class LazzyTest{
    public static void main(String...s){
        SessionFactory sessionFactory = new SessionFactory();
        Session session = sessionFactory.openSession();
        Transaction transaction = session.beginTransaction();

        User user = session.get(User.class, 1);
        transaction.commit();
        session.close();

        // here comes the lazy fetch issue
        user.getLikedPage();
    }
}
```

lazyinitializeException . oneToMany (DB) . *favoritePage* DB .

.

1. - .
2. `Hibernate.initialize(user.getLikedPage())` - hibernate .

: https://riptutorial.com/ko/hibernate/topic/7249/----

6: SQL

Examples

Hibernate `Session` , `session` :

```
List<Object[]> result = session.createNativeQuery("SELECT * FROM some_table").list();
for (Object[] row : result) {
    for (Object col : row) {
        System.out.print(col);
    }
}
```

`some_table` `result` .

```
Object pollAnswered = getCurrentSession().createSQLQuery(
        "select * from TJ_ANSWERED_ASW where pol_id = "+pollId+" and prf_log =
'"+logid+"'").uniqueResult();
```

.

.

org.hibernate.NonUniqueResultException

.

.

SQL : https://riptutorial.com/ko/hibernate/topic/6978/-sql-

7:

Examples

ImplicitNamingStrategy

ImplicitNamingStrategy Hibernate , , , Entity .

, Hibernate .

```
FKe6hidh4u0qh8y1ijy59s2ee6m
```

.

```
FK_asset_tenant
```

ImplicitNamingStrategy .

ImplicitNamingStrategyJpaCompliantImpl ImplicitNamingStrategy .

```
import org.hibernate.boot.model.naming.Identifier;
import org.hibernate.boot.model.naming.ImplicitForeignKeyNameSource;
import org.hibernate.boot.model.naming.ImplicitNamingStrategyJpaCompliantImpl;

public class CustomNamingStrategy extends ImplicitNamingStrategyJpaCompliantImpl {

    @Override
    public Identifier determineForeignKeyName(ImplicitForeignKeyNameSource source) {
        return toIdentifier("FK_" + source.getTableName().getCanonicalName() + "_" +
source.getReferencedTableName().getCanonicalName(), source.getBuildingContext());
    }

}
```

ImplicitNamingStrategy ImplicitNamingStrategy Hibernate , persistence.xml hibernate.cfg.xml hibernate.implicit_naming_strategy .

```
<property name="hibernate.implicit_naming_strategy"
                  value="com.example.foo.bar.CustomNamingStrategy"/>
```

hibernate.properties :

```
hibernate.implicit_naming_strategy=com.example.foo.bar.CustomNamingStrategy
```

name CustomNamingStrategy CustomNamingStrategy .

@Table . , Hibernate @Table . .

, .

```
ApplicationEventLog
```

.

```
application_event_log
```

snake case db . `PhysicalNamingStrategyStandardImpl` :

```
import org.hibernate.boot.model.naming.Identifier;
import org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl;
import org.hibernate.engine.jdbc.env.spi.JdbcEnvironment;

public class PhysicalNamingStrategyImpl extends PhysicalNamingStrategyStandardImpl {

    private static final long serialVersionUID = 1L;
    public static final PhysicalNamingStrategyImpl INSTANCE = new
PhysicalNamingStrategyImpl();

    @Override
    public Identifier toPhysicalTableName(Identifier name, JdbcEnvironment context) {
        return new Identifier(addUnderscores(name.getText()), name.isQuoted());
    }

    @Override
    public Identifier toPhysicalColumnName(Identifier name, JdbcEnvironment context) {
        return new Identifier(addUnderscores(name.getText()), name.isQuoted());
    }

    protected static String addUnderscores(String name) {
        final StringBuilder buf = new StringBuilder(name);
        for (int i = 1; i < buf.length() - 1; i++) {
            if (Character.isLowerCase(buf.charAt(i - 1)) &&
                    Character.isUpperCase(buf.charAt(i)) &&
                    Character.isLowerCase(buf.charAt(i + 1))) {
                buf.insert(i++, '_');
            }
        }
        return buf.toString().toLowerCase(Locale.ROOT);
    }
}
```

db `toPhysicalTableName` `toPhysicalColumnName` .

`hibernate.physical_naming_strategy` `PhysicalNamingStrategyImpl` .

```
hibernate.physical_naming_strategy=com.example.foo.bar.PhysicalNamingStrategyImpl
```

`@Table` `@Column` .

```
@Entity
public class ApplicationEventLog {
    private Date startTimestamp;
    private String logUser;
    private Integer eventSuccess;

    @Column(name="finish_dtl")
```

```
    private String finishDetails;
}
```

db .

```
CREATE TABLE application_event_log (
  ...
  start_timestamp timestamp,
  log_user varchar(255),
  event_success int(11),
  finish_dtl varchar(2000),
  ...
)
```

db . `@Column(name="finish_dtl")`

: https://riptutorial.com/ko/hibernate/topic/3051/--

8:

Examples

. .

Country.java

```
package com.entity;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "countries")
public class Country {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

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

    @Column(name = "national_language")
    private String nationalLanguage;

    @OneToOne(mappedBy = "country")
    private Capital capital;

    //Constructor

    //getters and setters

   }
```

Capital.java

```
package com.entity;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "capitals")
public class Capital {
```

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

    private String name;

    private long population;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "country_id")
    private Country country;

    //Constructor

    //getters and setters

}
```

HibernateDemo.java

```
package com.entity;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateDemo {

public static void main(String ar[]) {
    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();
    Country india = new Country();
    Capital delhi = new Capital();
    delhi.setName("Delhi");
    delhi.setPopulation(357828394);
    india.setName("India");
    india.setNationalLanguage("Hindi");
    delhi.setCountry(india);
    session.save(delhi);
    session.close();
  }

}
```

: https://riptutorial.com/ko/hibernate/topic/6478/-

9:

Examples

EAGER

Hibernate , `EAGER` `LAZY` .

`EAGER` JPA .

`Person` `Address` :

```
@Entity
public class Person {

  @OneToMany(mappedBy="address", fetch=FetchType.EAGER)
  private List<Address> addresses;

}
```

`Person` `Address` `Person` .

```
@ManyToMany(mappedBy="address", fetch=FetchType.EAGER)
```

:

```
@ManyToMany(mappedBy="address", fetch=FetchType.LAZY)
```

`@OneToOne` `@ManyToOne` . EAGER. .

```
@ManyToOne(fetch=FetchType.LAZY)
```

:

```
@OneToOne(fetch=FetchType.LAZY)
```

Hibernate . `JOINED` JOIN .

Hibernate .

`JOINED` `Bicycle` `MountainBike` :

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Bicycle {

}
```

:

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public class MountainBike extends Bicycle {

}
```

`MountainBike` JPQL `Bicycle` SQL .

```
select mb.*, b.* from MountainBike mb JOIN Bicycle b ON b.id = mb.id WHERE ...
```

`Bicycle` (: `Transport`) JOIN .

, `EAGER` . `MountainBike` .

.

`MountainBike` `bicycle` .

```
@Entity
public class MountainBike {

    @OneToOne(fetchType = FetchType.LAZY)
    private Bicycle bicycle;

}
```

`Bicycle` :

```
@Entity
public class Bicycle {

}
```

`MountainBike` .

: https://riptutorial.com/ko/hibernate/topic/2326/-

10: Hibernate

`@OneToOne`	.
`@OneToMany`	.
`@ManyToOne`	.
`@Entity`	.
`@Table`	.
`@JoinColumn`	foregin .
`@JoinTable`	.

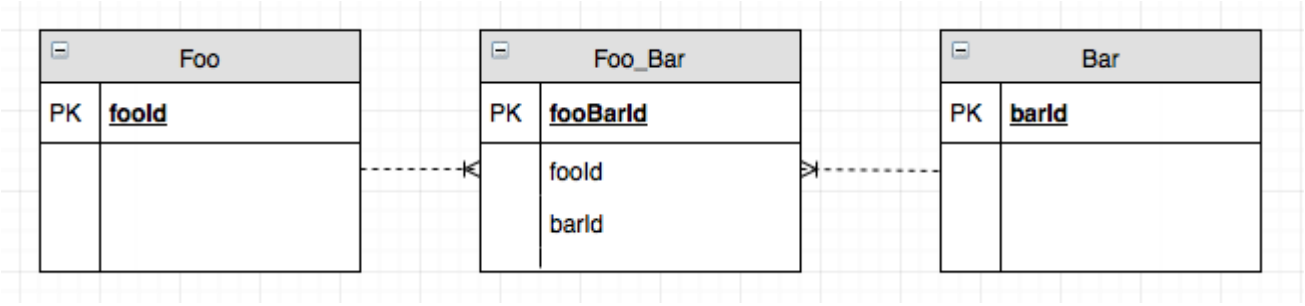
Examples

```
@Entity
@Table(name="FOO")
public class Foo {
    private UUID fooId;

    @OneToMany(mappedBy = "bar")
    private List<FooBar> bars;
}

@Entity
@Table(name="BAR")
public class Bar {
    private UUID barId;

    @OneToMany(mappedBy = "foo")
    private List<FooBar> foos;
}

@Entity
@Table(name="FOO_BAR")
public class FooBar {
    private UUID fooBarId;

    @ManyToOne
    @JoinColumn(name = "fooId")
    private Foo foo;

    @ManyToOne
    @JoinColumn(name = "barId")
    private Bar bar;

    //You can store other objects/fields on this table here.
}
```

Foo Bar .

Foo FOO . Bar BAR . Foo Bar FOO_BAR . FooBar .

.

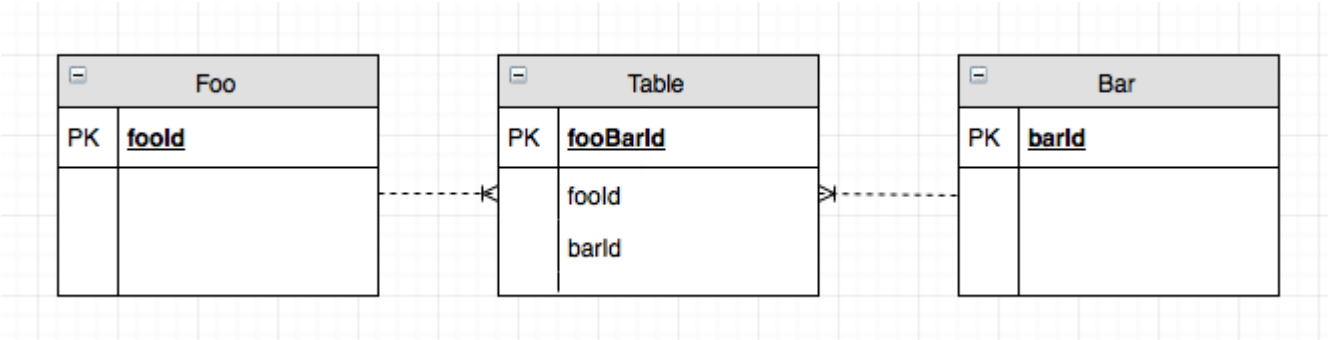


Hibernate

```
@Entity
@Table(name="FOO")
public class Foo {
    private UUID fooId;

    @OneToMany
    @JoinTable(name="FOO_BAR",
        joinColumns = @JoinColumn(name="fooId"),
        inverseJoinColumns = @JoinColumn(name="barId"))
    private List<Bar> bars;
}

@Entity
@Table(name="BAR")
public class Bar {
    private UUID barId;

    @OneToMany
    @JoinTable(name="FOO_BAR",
        joinColumns = @JoinColumn(name="barId"),
        inverseJoinColumns = @JoinColumn(name="fooId"))
    private List<Foo> foos;
}
```

Hibernate Foo Bar .

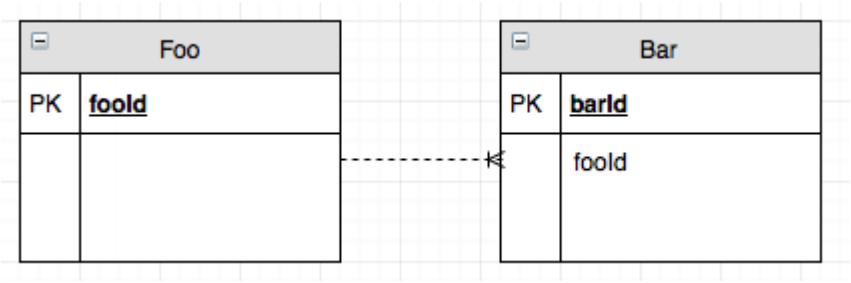
Foo FOO . Bar BAR . Foo Bar FOO_BAR . FooBar .



```
@Entity
@Table(name="FOO")
public class Foo {
    private UUID fooId;

    @OneToMany(mappedBy = "bar")
    private List<Bar> bars;
}

@Entity
@Table(name="BAR")
public class Bar {
    private UUID barId;

    @ManyToOne
    @JoinColumn(name = "fooId")
    private Foo foo;
}
```

Foo Bar .

Foo FOO . Bar BAR . fooId C BAR .



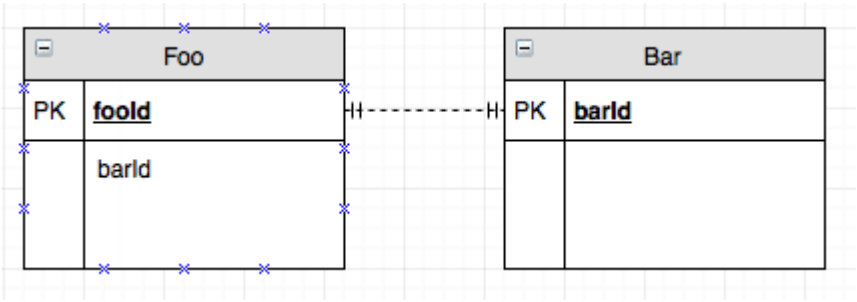
Foo.class

```
@Entity
@Table(name="FOO")
public class Foo {
    private UUID fooId;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "barId")
    private Bar bar;
}

@Entity
@Table(name="BAR")
public class Bar {
    private UUID barId;

    @OneToOne(mappedBy = "bar")
    private Foo foo;
}
```

Foo Bar .

Foo FOO . Bar BAR . barId FOO .

mappedBy .



1

```
@Entity
@Table(name="FOO")
public class Foo {
    private UUID fooId;

    @OneToMany
    @JoinTable(name="FOO_BAR",
        joinColumns = @JoinColumn(name="fooId"),
        inverseJoinColumns = @JoinColumn(name="barId", unique=true))
    private List<Bar> bars;
}

@Entity
@Table(name="BAR")
public class Bar {
    private UUID barId;

    //No Mapping specified here.
}

@Entity
@Table(name="FOO_BAR")
public class FooBar {
    private UUID fooBarId;

    @ManyToOne
    @JoinColumn(name = "fooId")
    private Foo foo;

    @ManyToOne
    @JoinColumn(name = "barId", unique = true)
    private Bar bar;

    //You can store other objects/fields on this table here.
}
```

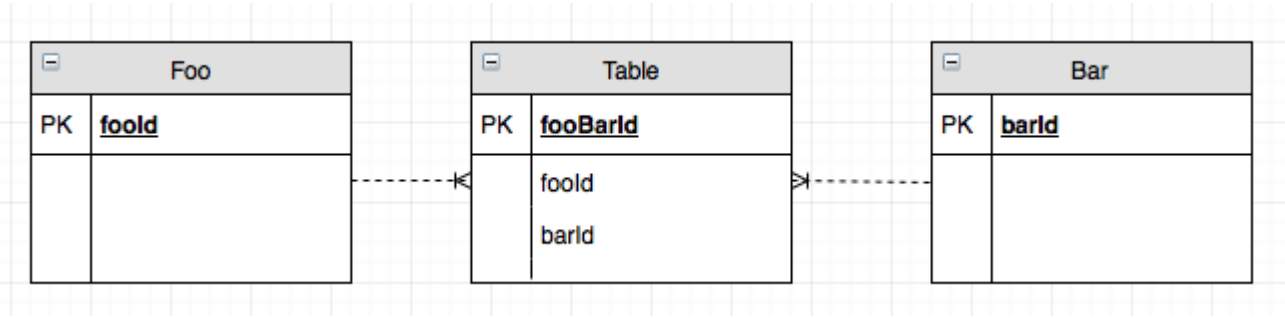
Foo Bar .

ManyToMany unique OneToMany .

Foo FOO . Bar BAR . Foo Bar FOO_BAR . FooBar .

Foo Bar . Bar Foo .

Role User Spring Security . Role .

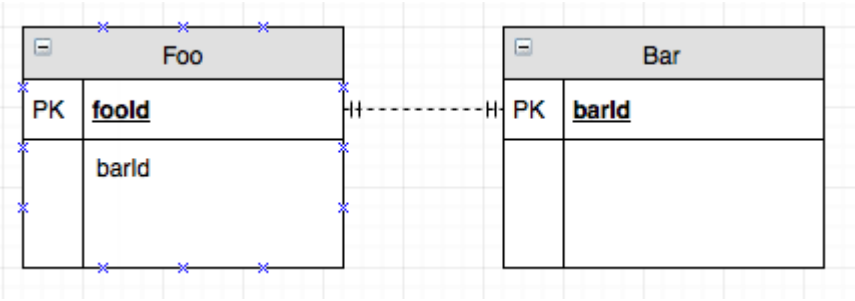



```
@Entity
@Table(name="FOO")
public class Foo {
    private UUID fooId;

    @OneToOne
    private Bar bar;
}

@Entity
@Table(name="BAR")
public class Bar {
    private UUID barId;
    //No corresponding mapping to Foo.class
}
```

Foo Bar .

Foo FOO . Bar BAR .

Foo Bar . Bar Foo .



Hibernate : https://riptutorial.com/ko/hibernate/topic/5742/--hibernate--

11:

Examples

OneToMany

OneToMany 2 . .

CountryEntity .

```
@Entity
@Table(name = "Country")
public class CountryEntity implements Serializable
{
   private static final long serialVersionUID = 1L;

   @Id
   @Column(name = "COUNTRY_ID", unique = true, nullable = false)
   @GeneratedValue(strategy = GenerationType.SEQUENCE)
   private Integer           countryId;

   @Column(name = "COUNTRY_NAME", unique = true, nullable = false, length = 100)
   private String            countryName;

   @OneToMany(mappedBy="country", fetch=FetchType.LAZY)
   private Set<CityEntity> cities = new HashSet<>();

   //Getters and Setters are not shown
}
```

.

```
@Entity
@Table(name = "City")
public class CityEntity implements Serializable
{
   private static final long serialVersionUID = 1L;

   @Id
   @Column(name = "CITY_ID", unique = true, nullable = false)
   @GeneratedValue(strategy = GenerationType.SEQUENCE)
   private Integer           cityId;

   @Column(name = "CITY_NAME", unique = false, nullable = false, length = 100)
   private String            cityName;

   @ManyToOne(optional=false, fetch=FetchType.EAGER)
   @JoinColumn(name="COUNTRY_ID", nullable=false)
   private CountryEntity country;

   //Getters and Setters are not shown
}
```

XML (one to many)

XML . .

:

```
public class Author {
    private int id;
    private String firstName;
    private String lastName;

    public Author(){

    }
    public int getId(){
        return id;
    }
    public void setId(int id){
        this.id = id;
    }
    public String getFirstName(){
        return firstName;
    }
    public void setFirstName(String firstName){
        this.firstName = firstName;
    }
    public String getLastName(){
        return lastName;
    }
    public void setLastName(String lastName){
        this.lastName = lastName;
    }
}
```

:

```
public class Book {
    private int id;
    private String isbn;
    private String title;
    private Author author;
    private String publisher;

    public Book() {
        super();
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getIsbn() {
        return isbn;
    }
    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }
```

```
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public Author getAuthor() {
        return author;
    }
    public void setAuthor(Author author) {
        this.author = author;
    }
    public String getPublisher() {
        return publisher;
    }
    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }
}
```

Author.hbm.xml :

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >

<hibernate-mapping>
   <class name="Author" table="author">
      <meta attribute="class-description">
         This class contains the author's information.
      </meta>
      <id name="id" type="int" column="author_id">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
   </class>
</hibernate-mapping>
```

Book.hbm.xml :

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >

<hibernate-mapping>
    <class name="Book" table="book_title">
      <meta attribute="class-description">
         This class contains the book information.
      </meta>
        <id name="id" type="int" column="book_id">
            <generator class="native"/>
        </id>
        <property name="isbn" column="isbn" type="string"/>
        <property name="title" column="title" type="string"/>
         <many-to-one name="author" class="Author" cascade="all">
             <column name="author"></column>
```

```
        </many-to-one>
        <property name="publisher" column="publisher" type="string"/>
    </class>
</hibernate-mapping>
```

Book Author XML <many-to-one> . cascade / .

: https://riptutorial.com/ko/hibernate/topic/6165/---

12: JPA

Examples

Hibernate JPA

Hibernate JPA . , Hibernate .

Hibernate JPA . JPA . Hibernate .

JPA : https://riptutorial.com/ko/hibernate/topic/6313/----jpa

13:

JPA (Java Persistence API) . JPA HQL (Hibernate Query Language) JPQL (Java Persistence Query Language) . JPA . SQL .

Examples

FetchType.LAZY . .

Employer , . **fetch = FetchType.EAGER** fetch = FetchType.LAZY . LAZY .

```
@Entity
@Table(name = "employer")
    public class Employer
    {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long id;

        @Column(name = "name")
        private String Name;

        @OneToMany(mappedBy = "employer", fetch = FetchType.LAZY,
               cascade = { CascadeType.ALL }, orphanRemoval = true)
        private List<Employee> employees;

        public Long getId() {
            return id;
        }

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

        public String getName() {
            return name;
        }

        public void setName(String name) {
            this.name = name;
        }

        public List<Employee> getEmployees() {
            return employees;
        }

        public void setEmployees(List<Employee> employees) {
            this.employees = employees;
        }
    }
```

LAZY LazyInitializationException . HQL / JPQL JOIN FETCH LazyInitializationException .

```
SELECT Employer employer FROM Employer
```

```
        LEFT JOIN FETCH employer.name
        LEFT JOIN FETCH employer.employee employee
        LEFT JOIN FETCH employee.name
        LEFT JOIN FETCH employer.address
```

: https://riptutorial.com/ko/hibernate/topic/9475/-----

14:

Examples

WildFly

WildFly Hibernate 2 `persistence.xml` .

```
<property name="hibernate.cache.use_second_level_cache" value="true"/>
```

.

```
<property name="hibernate.cache.use_query_cache" value="true"/>
```

WildFly Infinisan Hibernate Second-Level Cache . `hibernate.cache.provider_class` .

: https://riptutorial.com/ko/hibernate/topic/3462/

S. No		Contributors
1		Community, JamesENL, Michael Piefel, Naresh Kumar, Reborn, user7491506
2	Criterias and Projections	Saifer, Sameer Srivastava
3	HQL	Daniel Käfer, user7491506
4	SQL /	Daniel Käfer, Dherik, JamesENL, Michael Piefel
5		BELLIL, Pramod, Pritam Banerjee, vicky
6	SQL	Daniel Käfer, Nathaniel Ford, Sandeep Kamath
7		Mitch Talmadge, Naresh Kumar, veljkost
8		Dherik, omkar sirra
9		Dherik, Michael Piefel
10	Hibernate	Aleksei Loginov, JamesENL
11		StanislavL, user7491506
12	JPA	Michael Piefel
13		rObOtAndChalie
14		Mitch Talmadge