Hibernate

主流 ORM 框架 Object Relation Mapping 对象关系映射，将面向对象映射成面向关系。

如何使用

1、导入相关依赖

2、创建 Hibernate 配置文件

3、创建实体类

4、创建实体类-关系映射文件

5、调用 Hibernate API 完成操作

具体操作

1、创建 Maven 工程，pom.xml

```
<dependencies>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.19</version>
    </dependency>

    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.4.10.Final</version>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.10</version>
    </dependency>
</dependencies>
```

2、hibernate.cfg.xml

核心配置：session-factory

SessionFactory：针对单个数据库映射经过编译的内存镜像文件，将数据库转换为一个 Java 可以识别的镜像文件。

构建 SessionFactory 非常耗费资源，所以通常一个工程只需要创建一个 SessionFactory。

```
<?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>
        <!-- 数据源配置 -->
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>
        <property
name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/test?
useUnicode=true&amp;characterEncoding=UTF-8</property>

        <!-- C3P0 -->
        <property name="hibernate.c3p0.acquire_increment">10</property>
        <property name="hibernate.c3p0.idle_test_period">10000</property>
        <property name="hibernate.c3p0.timeout">5000</property>
        <property name="hibernate.c3p0.max_size">30</property>
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_statements">10</property>

        <!-- 数据库方言 -->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

        <!-- 打印SQL -->
        <property name="show_sql">true</property>

        <!-- 格式化SQL -->
        <property name="format_sql">true</property>

        <!-- 是否自动生成数据库 -->
        <property name="hibernate.hbm2ddl.auto"></property>

    </session-factory>

</hibernate-configuration>
```

3、创建实体类

```java
package com.southwind.entity;

import lombok.Data;

import java.util.Set;

@Data
public class Customer {
    private Integer id;
    private String name;
    private Set<Orders> orders;
}
```

```java
package com.southwind.entity;

import lombok.Data;

@Data
public class Orders {
    private Integer id;
    private String name;
    private Customer customer;
}
```

4、创建实体关系映射文件

```java
package com.southwind.entity;

import lombok.Data;

@Data
public class People {
    private Integer id;
    private String name;
    private Double money;
}
```

Columns (3) | Keys (1) | Indices | Foreign Keys

```
id     int(11) -- part of primary key
name   varchar(11)
money  double
```

```xml
<?xml version="1.0"?>
```

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

    <class name="com.southwind.entity.People" table="people">

        <id name="id" type="java.lang.Integer">
            <column name="id"></column>
            <generator class="identity"></generator>
        </id>

        <property name="name" type="java.lang.String">
            <column name="name"></column>
        </property>

        <property name="money" type="java.lang.Double">
            <column name="money"></column>
        </property>

    </class>

</hibernate-mapping>
```

5、实体关系映射文件注册到 Hibernate 的配置文件中。

```
<!-- 注册实体关系映射文件 -->
<mapping resource="com/southwind/entity/People.hbm.xml"></mapping>
```

6、使用 Hibernate API 完成数据操作。

```
package com.southwind.test;

import com.southwind.entity.People;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Test {
    public static void main(String[] args) {
        //创建Configuration
        Configuration configuration = new
Configuration().configure("hibernate.xml");
        //获取SessionFactory
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        //获取Session
        Session session = sessionFactory.openSession();
        People people = new People();
```

```
        people.setName("张三");
        people.setMoney(1000.0);
        session.save(people);
        session.beginTransaction().commit();
        session.close();
    }
}
```

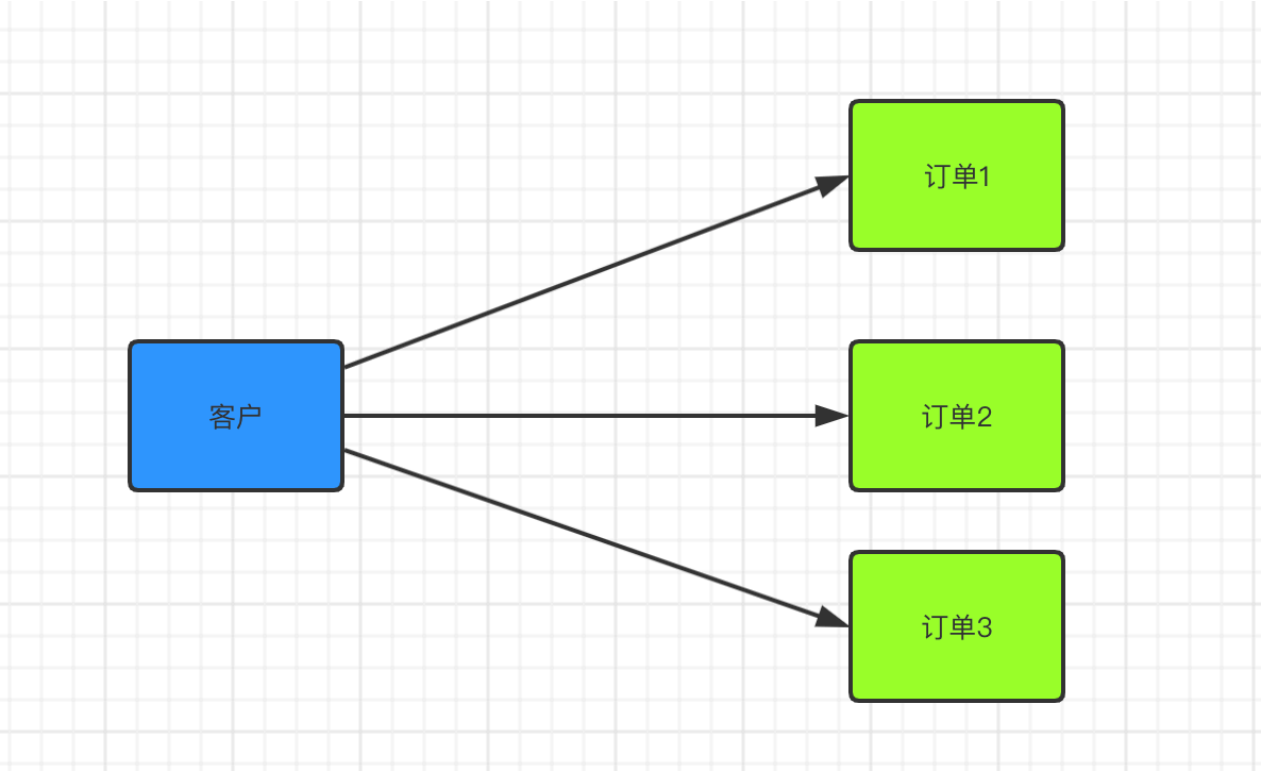
7、pom.xml 中需要配置 resource。

```
<build>

    <resources>
        <resource>
            <directory>src/main/java</directory>
            <includes>
                <include>**/*.xml</include>
            </includes>
        </resource>
    </resources>

</build>
```

Hibernate 级联操作

1、一对多关系

客户和订单：每个客户可以购买多个产品，生成多个订单，但是一个订单只能属于一个客户，所以客户是一，订单是多。



数据库中一的一方是主表，多的一方时候从表，通过主外键关系来维护。

面向对象中

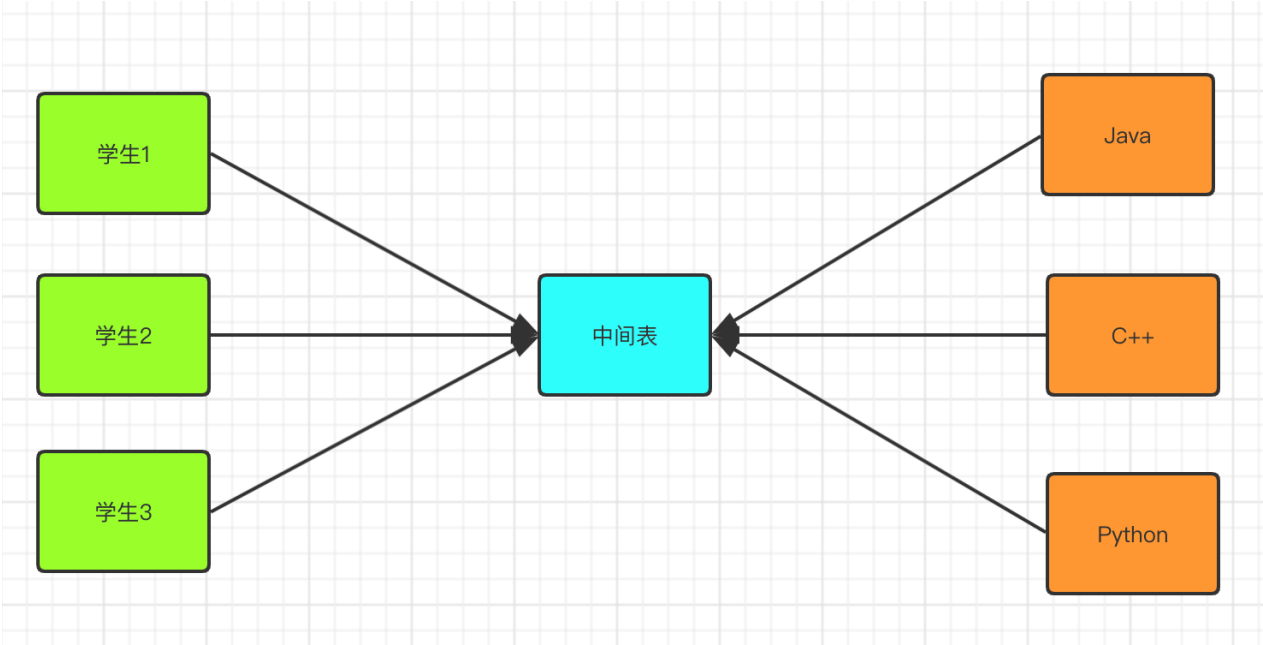
```
package com.southwind.entity;

import lombok.Data;

import java.util.Set;

@Data
public class Customer {
    private Integer id;
    private String name;
    private Set<Orders> orders;
}
```

```
package com.southwind.entity;

import lombok.Data;

@Data
public class Orders {
    private Integer id;
    private String name;
    private Customer customer;
}
```

2、多对多关系

学生选课：一门课程可以被多个学生选择，一个学生可以选择多门课程，学生是多，课程也是多。

数据库中是通过两个一对多关系来维护的，学生和课程都是主表，额外增加一张中间表作为从表，两张主表和中间表都是一对多关系。



面向对象中

```
package com.southwind.entity;

import lombok.Data;

import java.util.Set;

@Data
public class Account {
    private Integer id;
    private String name;
    private Set<Course> courses;
}
```

```
package com.southwind.entity;

import lombok.Data;

import java.util.Set;

@Data
public class Course {
    private Integer id;
    private String name;
    private Set<Account> accounts;
}
```

Java 和数据库对于这两种关系的体现完全是两种不同的方式，Hibernate 框架的作用就是将这两种方式进行转换和映射。

Hibernate 实现一对多

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

    <class name="com.southwind.entity.Customer" table="customer">

        <id name="id" type="java.lang.Integer">
            <column name="id"></column>
            <generator class="identity"></generator>
        </id>

        <property name="name" type="java.lang.String">
            <column name="name"></column>
        </property>

        <set name="orders" table="orders">
            <key column="cid"></key>
            <one-to-many class="com.southwind.entity.Orders"></one-to-many>
        </set>

    </class>

</hibernate-mapping>
```

- set 标签来配置实体类中的集合属性 orsers
- name 实体类属性名
- table 表名
- key 外键
- one-to-many 与集合泛型的实体类对应

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

    <class name="com.southwind.entity.Orders" table="orders">

        <id name="id" type="java.lang.Integer">
            <column name="id"></column>
            <generator class="identity"></generator>
        </id>

        <property name="name" type="java.lang.String">
```

```
            <column name="name"></column>
        </property>

        <many-to-one name="customer" class="com.southwind.entity.Customer"
column="cid"></many-to-one>

    </class>

</hibernate-mapping>
```

- many-to-one 配置实体类对应的对象属性
- name 属性名
- class 属性对应的类
- column 外键

需要在 Hibernate 配置文件中进行注册

```
<!-- 注册实体关系映射文件 -->
<mapping resource="com/southwind/entity/Customer.hbm.xml"></mapping>
<mapping resource="com/southwind/entity/Orders.hbm.xml"></mapping>
```

一对多

Hibernate API

```
package com.southwind.test;

import com.southwind.entity.Customer;
import com.southwind.entity.Orders;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Test2 {
    public static void main(String[] args) {
        //创建 Configuration
        Configuration configuration = new
Configuration().configure("hibernate.xml");
        //获取 SessionFactory
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        //获取 Session
        Session session = sessionFactory.openSession();
        //创建 Customer
        Customer customer = new Customer();
        customer.setName("张三");

        //创建 Orders
        Orders orders = new Orders();
        orders.setName("订单1");
```

```
        //建立关联关系
        orders.setCustomer(customer);

        //保存
        session.save(customer);
        session.save(orders);

        //提交事务
        session.beginTransaction().commit();

        //关闭session
        session.close();
    }
}
```

多对多

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

    <class name="com.southwind.entity.Account" table="t_account">

        <id name="id" type="java.lang.Integer">
            <column name="id"></column>
            <generator class="identity"></generator>
        </id>

        <property name="name" type="java.lang.String">
            <column name="name"></column>
        </property>

        <set name="courses" table="account_course">
            <key column="aid"></key>
            <many-to-many class="com.southwind.entity.Course" column="cid">
</many-to-many>
        </set>

    </class>

</hibernate-mapping>
```

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
```

```
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

    <class name="com.southwind.entity.Course" table="t_course">

        <id name="id" type="java.lang.Integer">
            <column name="id"></column>
            <generator class="identity"></generator>
        </id>

        <property name="name" type="java.lang.String">
            <column name="name"></column>
        </property>

        <set name="accounts" table="account_course">
            <key column="cid"></key>
            <many-to-many class="com.southwind.entity.Account" column="aid">
</many-to-many>
        </set>

    </class>

</hibernate-mapping>
```

name 实体类对应的集合属性名

table 中间表名

key 外键

many-to-many 与集合泛型的实体类对应

column 属性与中间表的外键字段名对应

注册 Hibernate 配置文件中

```
<mapping resource="com/southwind/entity/Account.hbm.xml"></mapping>
<mapping resource="com/southwind/entity/Course.hbm.xml"></mapping>
```

Hibernate API

```
package com.southwind.test;

import com.southwind.entity.Account;
import com.southwind.entity.Course;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

import java.util.HashSet;
```

```java
import java.util.Set;

public class Test3 {
    public static void main(String[] args) {
        //创建 Configuration
        Configuration configuration = new
Configuration().configure("hibernate.xml");
        //获取 SessionFactory
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        //获取 Session
        Session session = sessionFactory.openSession();

        Course course = new Course();
        course.setName("Java");

        Account account = new Account();
        account.setName("张三");

        Set<Course> courses = new HashSet<>();
        courses.add(course);

        account.setCourses(courses);

        session.save(course);
        session.save(account);

        session.beginTransaction().commit();

        session.close();
    }
}
```

Hibernate 延迟加载

延迟加载、惰性加载、懒加载

使用延迟加载可以提高程序的运行效率，Java 程序与数据库交互的频次越低，程序运行的效率就越高，所以我们应该尽量减少 Java 程序与数据库的交互次数，Hibernate 延迟加载就很好的做到了这一点。

客户和订单，当我们查询客户对象时，因为有级联设置，所以会将对应的订单信息一并查询出来，这样就需要发送两条 SQL 语句，分别查询客户信息和订单信息。

延迟加载的思路是：当我们查询客户的时候，如果没有访问订单数据，只发送一条 SQL 语句查询客户信息，如果需要访问订单数据，则发送两条 SQLL。

延迟加载可以看作是一种优化机制，根据具体的需求，自动选择要执行的 SQL 语句数量。

一对多

1、查询 Customer，对 orders 进行延迟加载设置，在 customer.hbm.xml 进行设置，延迟加载默认开启。

```
<set name="orders" table="orders" lazy="true">
    <key column="cid"></key>
    <one-to-many class="com.southwind.entity.Orders"></one-to-many>
</set>
```

2、查询 Customer

```
package com.southwind.test;

import com.southwind.entity.Customer;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Test4 {
    public static void main(String[] args) {
        //创建 Configuration
        Configuration configuration = new
Configuration().configure("hibernate.xml");
        //获取 SessionFactory
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        //获取 Session
        Session session = sessionFactory.openSession();

        Customer customer = session.get(Customer.class,15);
        System.out.println(customer);
        session.close();
    }
}
```

```
INFO: HHH000490: Using JtaPlatform implementation: [o
Hibernate:
    select
        customer0_.id as id1_1_0_,
        customer0_.name as name2_1_0_
    from
        customer customer0_
    where
        customer0_.id=?
Customer{id=15, name='张三'}

Process finished with exit code 0
```

```
package com.southwind.test;
```

```java
import com.southwind.entity.Customer;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Test4 {
    public static void main(String[] args) {
        //创建 Configuration
        Configuration configuration = new
Configuration().configure("hibernate.xml");
        //获取 SessionFactory
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        //获取 Session
        Session session = sessionFactory.openSession();

        Customer customer = session.get(Customer.class,15);
        System.out.println(customer.getOrders());
        session.close();
    }
}
```

```
Hibernate:
    select
        customer0_.id as id1_1_0_,
        customer0_.name as name2_1_0_
    from
        customer customer0_
    where
        customer0_.id=?
Hibernate:
    select
        orders0_.cid as cid3_2_0_,
        orders0_.id as id1_2_0_,
        orders0_.id as id1_2_1_,
        orders0_.name as name2_2_1_,
        orders0_.cid as cid3_2_1_
    from
        orders orders0_
    where
        orders0_.cid=?
[Orders{id=26, name='订单1'}]
```

```java
package com.southwind.test;

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

```java
public class Test4 {
    public static void main(String[] args) {
        //创建 Configuration
        Configuration configuration = new
Configuration().configure("hibernate.xml");
        //获取 SessionFactory
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        //获取 Session
        Session session = sessionFactory.openSession();

        Customer customer = session.get(Customer.class,15);
        System.out.println(customer);
        session.close();
    }
}
```

```
Hibernate:
    select
        customer0_.id as id1_1_0_,
        customer0_.name as name2_1_0_
    from
        customer customer0_
    where
        customer0_.id=?
Hibernate:
    select
        orders0_.cid as cid3_2_0_,
        orders0_.id as id1_2_0_,
        orders0_.id as id1_2_1_,
        orders0_.name as name2_2_1_,
        orders0_.cid as cid3_2_1_
    from
        orders orders0_
    where
        orders0_.cid=?
Customer{id=15, name='张三'}
```

lazy 除了可以设置 true 和 false 之外，还可以设置 extra，extra 是比 true 更加懒惰的一种加载方式，或者说是更加智能的一种加载方式，通过例子看区别：

查询 Customer 对象，打印该对象对应的 orders 集合的长度

```java
package com.southwind.test;

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

```java
public class Test4 {
    public static void main(String[] args) {
        //创建 Configuration
        Configuration configuration = new
Configuration().configure("hibernate.xml");
        //获取 SessionFactory
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        //获取 Session
        Session session = sessionFactory.openSession();

        Customer customer = session.get(Customer.class,15);
        System.out.println(customer.getOrders().size());
        session.close();
    }
}
```

```
Hibernate:
    select
        customer0_.id as id1_1_0_,
        customer0_.name as name2_1_0_
    from
        customer customer0_
    where
        customer0_.id=?
Hibernate:
    select
        count(id)
    from
        orders
    where
        cid =?
1
```