JPA 系列课程

时间: 2017/11/01
作者:袁毅雄
邮箱: 896778954@qq.com

目录

课程目标:

Hibernate 级联映射关系,懒加载,立即加载,控制反转
Hibernate 映射配置: Annotation, XML
Hibernate 优化:DDL 优化,DML 优化,DQL 优化
JAP 规则
Hibernate JPA 对 JPA 的实现
Spring Data JPA 对 JPA 规则的支持

Day01:初始 Hibernate

课程目标

Hibernate 是什么

Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以使用对象编程思维来操作数据库。

Hibernate 不仅提供了从对象类到数据表之间的映射，还提供了数据查询和恢复机制，相对于使用 JDBC 和 SQL 的手工来操作数据，使用 Hibernate 可以大大减少操作数据库的编程工作量

Hibernate 发展史

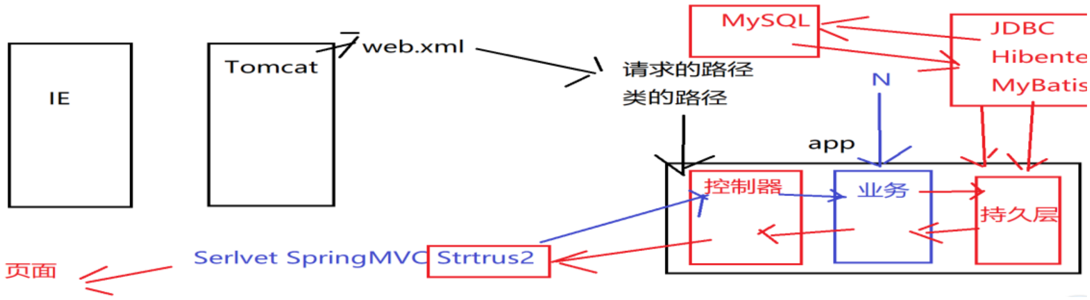
2001 年 11 月，澳大利亚墨尔本一位名为 Gavin King 的 27 岁的程序员发布了 Hibernate 第一个版本。

2003 年 9 月，Hibernate 开发团队进入 JBoss 公司，从这个时候开始 Hibernate 得到了突飞猛进的普及和发展。

2004 年，随着 Rod Johnson 的著作《Expert One-on-One J2EE Development without EJB》出版后，整个 Java 社区开始从实体 bean 向 Hibernate 转移。

2006 年，J2EE5.0 标准正式发布以后，持久化框架标准 Java Persistent API（简称 JPA）基本上是参考 Hibernate 实现的，而 Hibernate 在 3.2 版本开始，已经完全兼容 JPA 标准。

Hibernate 的定位

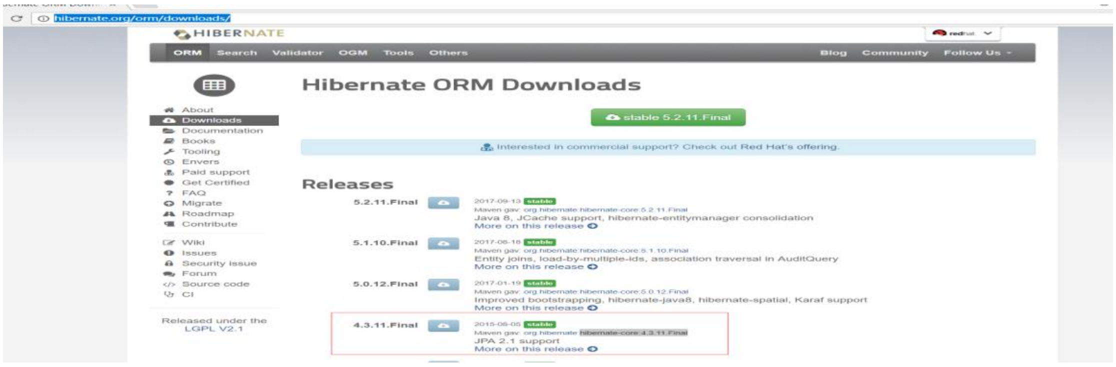


Hibernate 获取

版本:[hibernate-core:4.3.11.Final]

网址:http://hibernate.org/orm/downloads/

网站截图:



Hibernate ORM Downloads

stable 5.2.11.Final

Interested in commercial support? Check out Red Hat's offering.

Releases

5.2.11.Final 2017-09-13 stable
Maven gav: org.hibernate:hibernate-core:5.2.11.Final
Java 8, JCache support, hibernate-entitymanager consolidation
More on this release

5.1.10.Final 2017-08-18 stable
Maven gav: org.hibernate:hibernate-core:5.1.10.Final
Entity joins, load-by-multiple-ids, association traversal in AuditQuery
More on this release

5.0.12.Final 2017-01-19 stable
Maven gav: org.hibernate:hibernate-core:5.0.12.Final
Improved bootstrapping, hibernate-java8, hibernate-spatial, Karaf support
More on this release

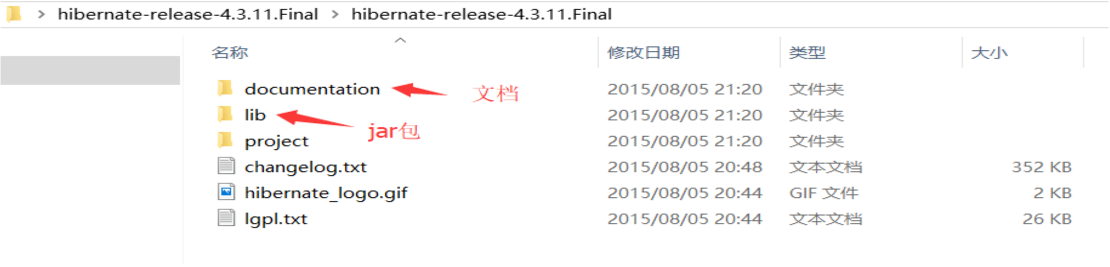
4.3.11.Final 2015-08-05 stable
Maven gav: org.hibernate:hibernate-core:4.3.11.Final
JPA 2.1 support
More on this release

Released under the LGPL V2.1

Hibernate 目录/文档简介/快速入门

Hibernate.cfg.xml

第一步:解压下载的 hibernate-core:4.3.11.Final

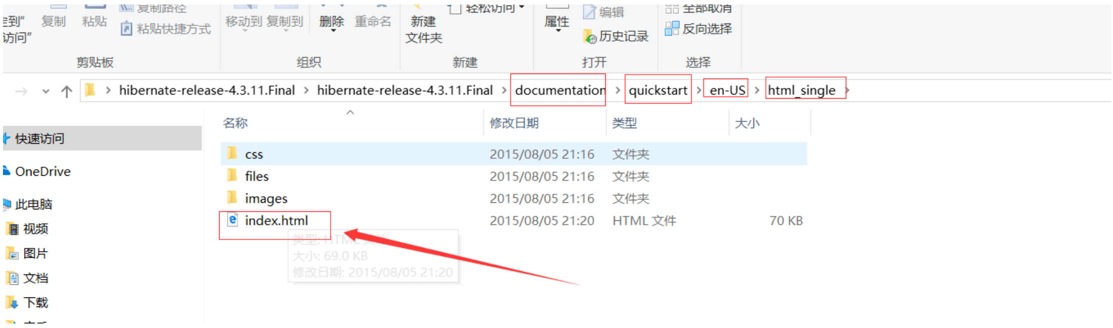


› hibernate-release-4.3.11.Final › hibernate-release-4.3.11.Final

名称	修改日期	类型	大小
documentation	2015/08/05 21:20	文件夹	
lib	2015/08/05 21:20	文件夹	
project	2015/08/05 21:20	文件夹	
changelog.txt	2015/08/05 20:48	文本文档	352 KB
hibernate_logo.gif	2015/08/05 20:44	GIF 文件	2 KB
lgpl.txt	2015/08/05 20:44	文本文档	26 KB

第二步:进入: html_single 目录

路径：\hibernate-release-4.3.11.Final\documentation\quickstart\en-US\html_single

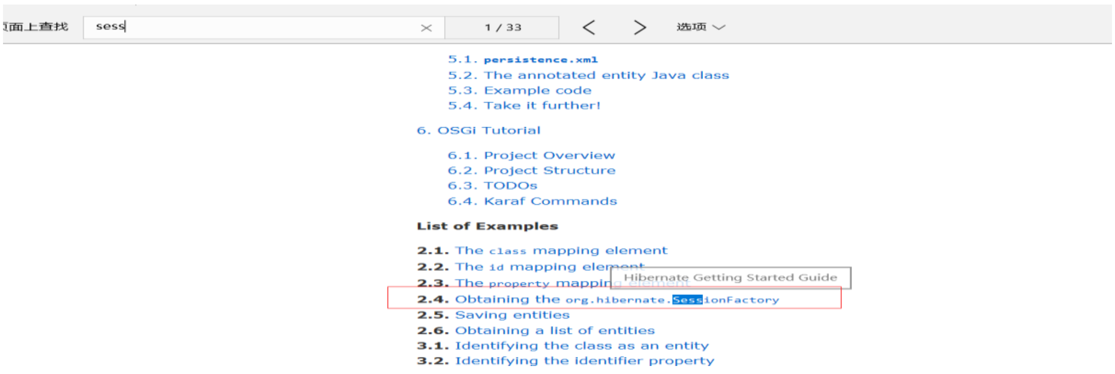


› hibernate-release-4.3.11.Final › hibernate-release-4.3.11.Final › documentation › quickstart › en-US › html_single ›

名称	修改日期	类型	大小
css	2015/08/05 21:16	文件夹	
files	2015/08/05 21:16	文件夹	
images	2015/08/05 21:16	文件夹	
index.html	2015/08/05 21:20	HTML 文件	70 KB

类型: HTML 文件
大小: 69.0 KB
修改日期: 2015/08/05 21:20

第三步:进入 index.html,全文检索 sessionFactory



5.1. persistence.xml
5.2. The annotated entity Java class
5.3. Example code
5.4. Take it further!

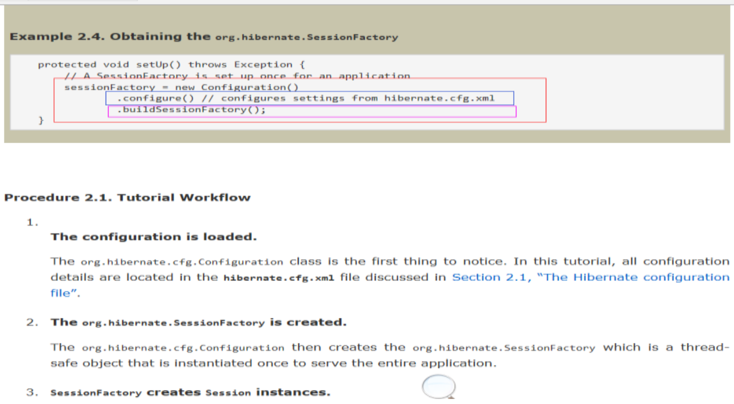
6. OSGi Tutorial

6.1. Project Overview
6.2. Project Structure
6.3. TODOs
6.4. Karaf Commands

List of Examples

2.1. The class mapping element
2.2. The id mapping element
2.3. The property mapping element
2.4. Obtaining the org.hibernate.SessionFactory
2.5. Saving entities
2.6. Obtaining a list of entities
3.1. Identifying the class as an entity
3.2. Identifying the identifier property

第四步:单机 Obtaining the org.hibernate.SessionFactory 观察如下开启界面框选中的代码



Example 2.4. Obtaining the org.hibernate.SessionFactory

```
protected void setUp() throws Exception {
    // A SessionFactory is set up once for an application
    sessionFactory = new Configuration()
            .configure() // configures settings from hibernate.cfg.xml
            .buildSessionFactory();
}
```

Procedure 2.1. Tutorial Workflow

1. **The configuration is loaded.**

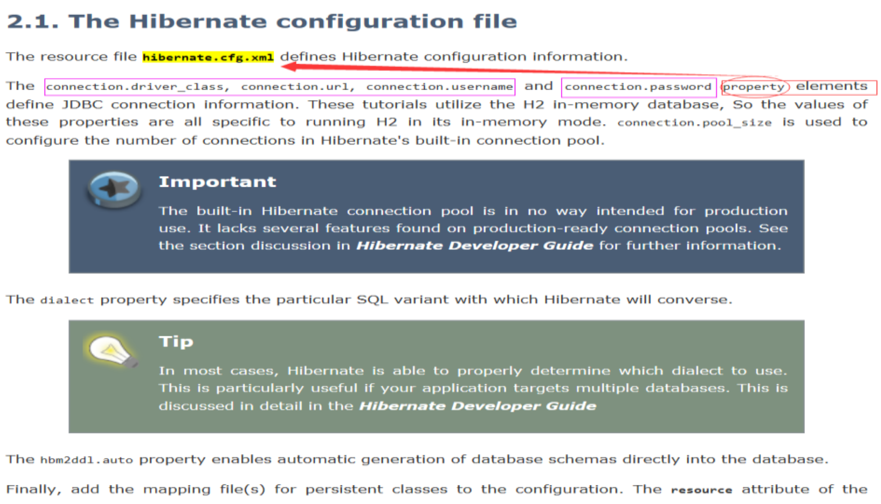
   The org.hibernate.cfg.Configuration class is the first thing to notice. In this tutorial, all configuration details are located in the **hibernate.cfg.xml** file discussed in Section 2.1, "The Hibernate configuration file".

2. **The org.hibernate.SessionFactory is created.**

   The org.hibernate.cfg.Configuration then creates the org.hibernate.SessionFactory which is a thread-safe object that is instantiated once to serve the entire application.

3. **SessionFactory creates Session instances.**

第五步:全文检索 hibernate.cfg.xml 观察如下开启界面框选中的说明



2.1. The Hibernate configuration file

The resource file **hibernate.cfg.xml** defines Hibernate configuration information.

The connection.driver_class, connection.url, connection.username and connection.password property elements define JDBC connection information. These tutorials utilize the H2 in-memory database, So the values of these properties are all specific to running H2 in its in-memory mode. connection.pool_size is used to configure the number of connections in Hibernate's built-in connection pool.

Important

The built-in Hibernate connection pool is in no way intended for production use. It lacks several features found on production-ready connection pools. See the section discussion in ***Hibernate Developer Guide*** for further information.

The dialect property specifies the particular SQL variant with which Hibernate will converse.

Tip

In most cases, Hibernate is able to properly determine which dialect to use. This is particularly useful if your application targets multiple databases. This is discussed in detail in the ***Hibernate Developer Guide***

The hbm2ddl.auto property enables automatic generation of database schemas directly into the database.

Finally, add the mapping file(s) for persistent classes to the configuration. The **resource** attribute of the

第六步:总结文档信息

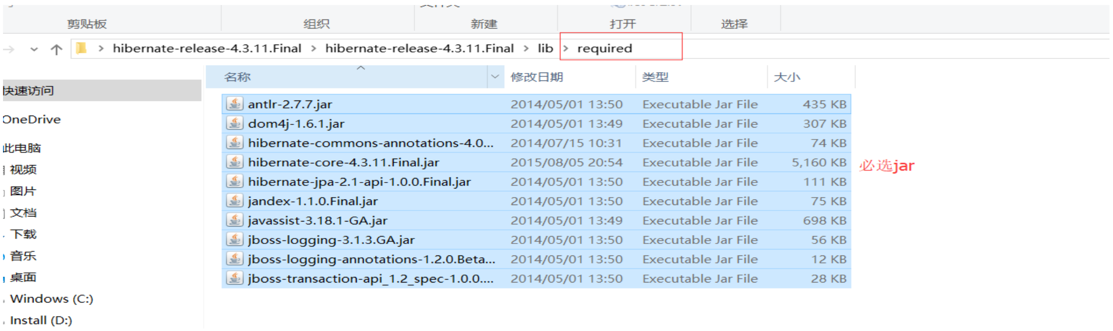
使用 hibernate 需要加载一个 hibernate.cfg.xml 文件才能构建出 sessionFactory

hibernate.cfg.xml 需要配置属性(property)

hibernate.cfg.xml 属性包含

属性	详细解释
connection.url	数据库连接字符串
connection.driver_class	驱动地址
connection.username	数据库连接账号
connection.password	数据库连接密码

Hibernate jars

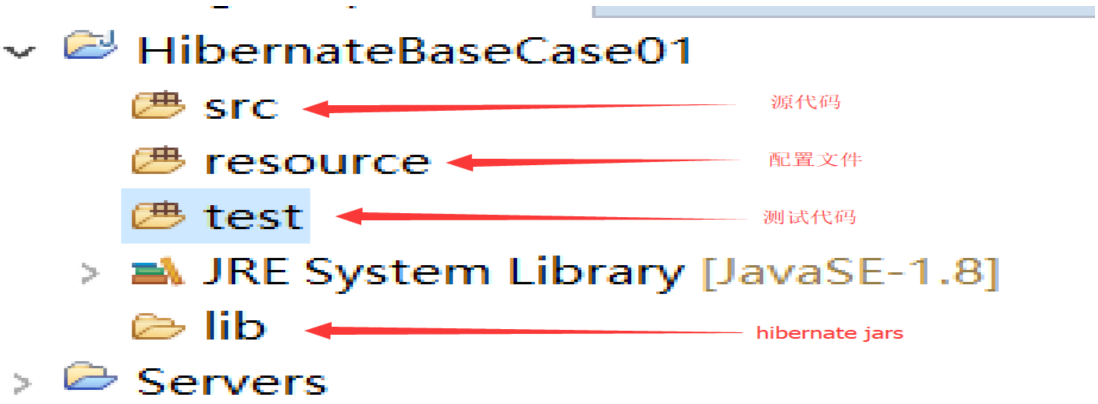


名称	修改日期	类型	大小
antlr-2.7.7.jar	2014/05/01 13:50	Executable Jar File	435 KB
dom4j-1.6.1.jar	2014/05/01 13:49	Executable Jar File	307 KB
hibernate-commons-annotations-4.0...	2014/07/15 10:31	Executable Jar File	74 KB
hibernate-core-4.3.11.Final.jar	2015/08/05 20:54	Executable Jar File	5,160 KB
hibernate-jpa-2.1-api-1.0.0.Final.jar	2014/05/01 13:50	Executable Jar File	111 KB
jandex-1.1.0.Final.jar	2014/05/01 13:50	Executable Jar File	75 KB
javassist-3.18.1-GA.jar	2014/05/01 13:49	Executable Jar File	698 KB
jboss-logging-3.1.3.GA.jar	2014/05/01 13:50	Executable Jar File	56 KB
jboss-logging-annotations-1.2.0.Beta...	2014/05/01 13:50	Executable Jar File	12 KB
jboss-transaction-api_1.2_spec-1.0.0....	2014/05/01 13:50	Executable Jar File	28 KB

必选jar

搭建 Hibernate 实现 DDL

项目骨架



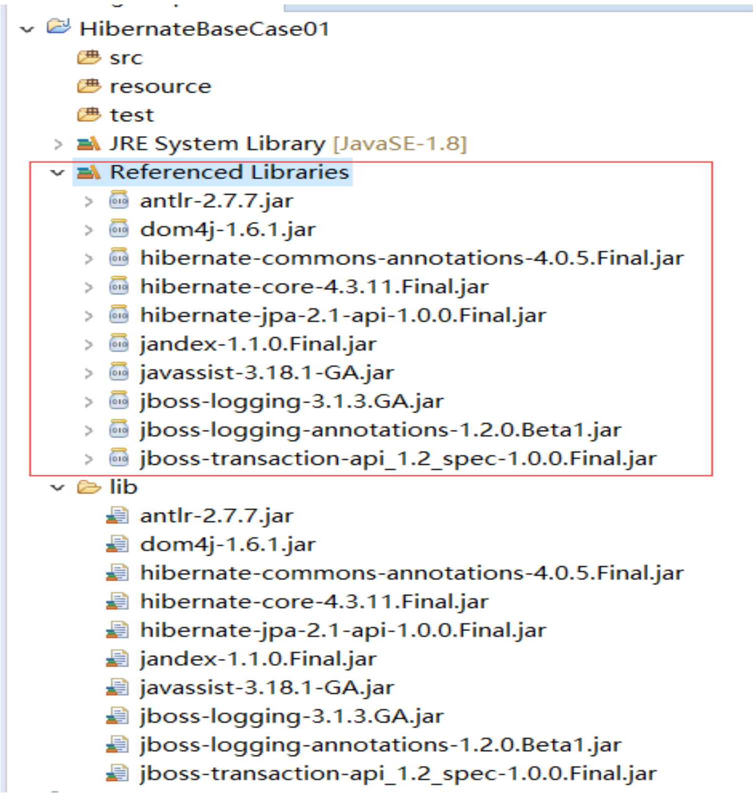
源代码

配置文件

测试代码

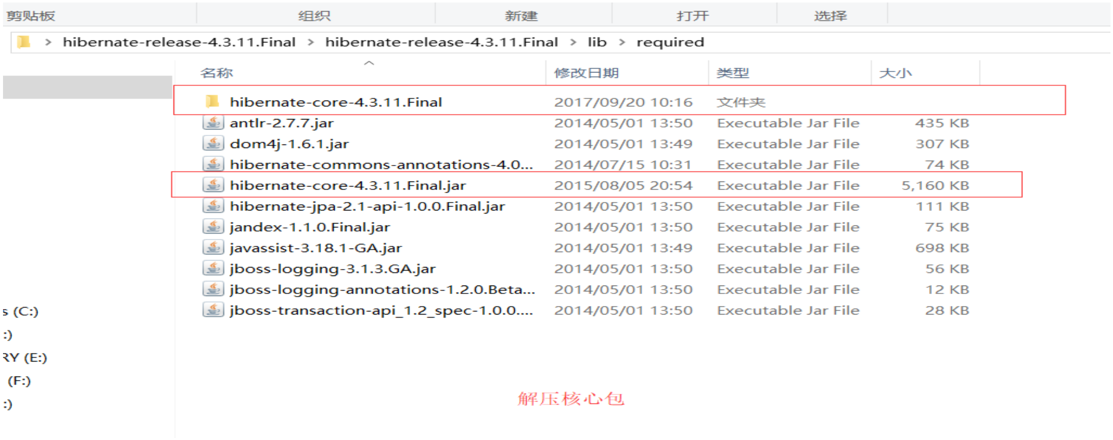
hibernate jars

导入 jars



```
HibernateBaseCase01
  src
  resource
  test
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
    antlr-2.7.7.jar
    dom4j-1.6.1.jar
    hibernate-commons-annotations-4.0.5.Final.jar
    hibernate-core-4.3.11.Final.jar
    hibernate-jpa-2.1-api-1.0.0.Final.jar
    jandex-1.1.0.Final.jar
    javassist-3.18.1-GA.jar
    jboss-logging-3.1.3.GA.jar
    jboss-logging-annotations-1.2.0.Beta1.jar
    jboss-transaction-api_1.2_spec-1.0.0.Final.jar
  lib
    antlr-2.7.7.jar
    dom4j-1.6.1.jar
    hibernate-commons-annotations-4.0.5.Final.jar
    hibernate-core-4.3.11.Final.jar
    hibernate-jpa-2.1-api-1.0.0.Final.jar
    jandex-1.1.0.Final.jar
    javassist-3.18.1-GA.jar
    jboss-logging-3.1.3.GA.jar
    jboss-logging-annotations-1.2.0.Beta1.jar
    jboss-transaction-api_1.2_spec-1.0.0.Final.jar
```

解压 xxx-core

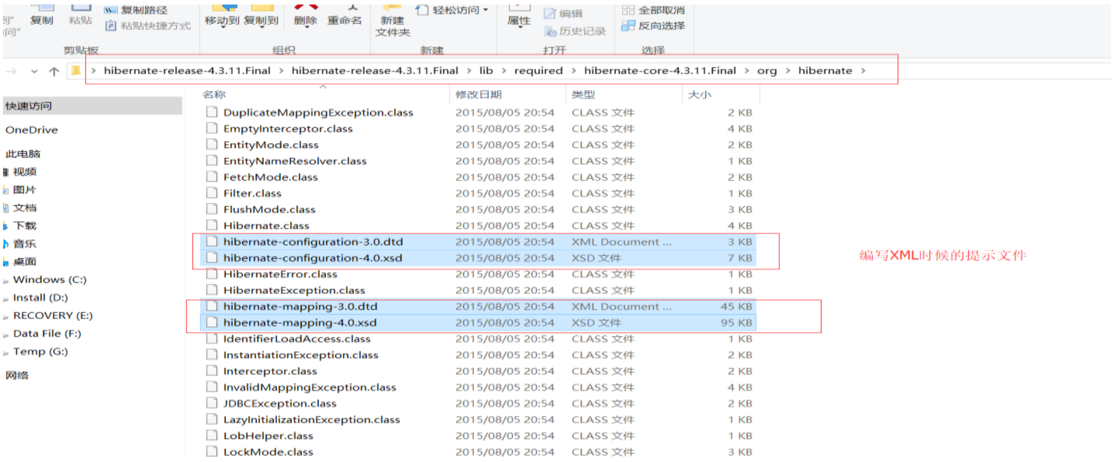


hibernate-release-4.3.11.Final > hibernate-release-4.3.11.Final > lib > required

名称	修改日期	类型	大小
hibernate-core-4.3.11.Final	2017/09/20 10:16	文件夹	
antlr-2.7.7.jar	2014/05/01 13:50	Executable Jar File	435 KB
dom4j-1.6.1.jar	2014/05/01 13:49	Executable Jar File	307 KB
hibernate-commons-annotations-4.0...	2014/07/15 10:31	Executable Jar File	74 KB
hibernate-core-4.3.11.Final.jar	2015/08/05 20:54	Executable Jar File	5,160 KB
hibernate-jpa-2.1-api-1.0.0.Final.jar	2014/05/01 13:50	Executable Jar File	111 KB
jandex-1.1.0.Final.jar	2014/05/01 13:50	Executable Jar File	75 KB
javassist-3.18.1-GA.jar	2014/05/01 13:49	Executable Jar File	698 KB
jboss-logging-3.1.3.GA.jar	2014/05/01 13:50	Executable Jar File	56 KB
jboss-logging-annotations-1.2.0.Beta...	2014/05/01 13:50	Executable Jar File	12 KB
jboss-transaction-api_1.2_spec-1.0.0....	2014/05/01 13:50	Executable Jar File	28 KB

解压核心包

找到 DTD 作用提供 XML 编写时候的提示



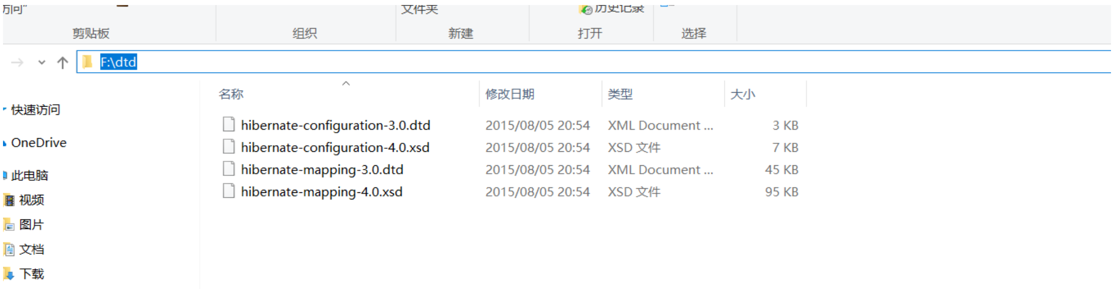
hibernate-release-4.3.11.Final > hibernate-release-4.3.11.Final > lib > required > hibernate-core-4.3.11.Final > org > hibernate >

名称	修改日期	类型	大小
DuplicateMappingException.class	2015/08/05 20:54	CLASS 文件	2 KB
EmptyInterceptor.class	2015/08/05 20:54	CLASS 文件	4 KB
EntityMode.class	2015/08/05 20:54	CLASS 文件	2 KB
EntityNameResolver.class	2015/08/05 20:54	CLASS 文件	1 KB
FetchMode.class	2015/08/05 20:54	CLASS 文件	2 KB
Filter.class	2015/08/05 20:54	CLASS 文件	1 KB
FlushMode.class	2015/08/05 20:54	CLASS 文件	3 KB
Hibernate.class	2015/08/05 20:54	CLASS 文件	4 KB
hibernate-configuration-3.0.dtd	2015/08/05 20:54	XML Document ...	3 KB
hibernate-configuration-4.0.xsd	2015/08/05 20:54	XSD 文件	7 KB
HibernateError.class	2015/08/05 20:54	CLASS 文件	1 KB
HibernateException.class	2015/08/05 20:54	CLASS 文件	1 KB
hibernate-mapping-3.0.dtd	2015/08/05 20:54	XML Document ...	45 KB
hibernate-mapping-4.0.xsd	2015/08/05 20:54	XSD 文件	95 KB
IdentifierLoadAccess.class	2015/08/05 20:54	CLASS 文件	1 KB
InstantiationException.class	2015/08/05 20:54	CLASS 文件	2 KB
Interceptor.class	2015/08/05 20:54	CLASS 文件	2 KB
InvalidMappingException.class	2015/08/05 20:54	CLASS 文件	4 KB
JDBCException.class	2015/08/05 20:54	CLASS 文件	2 KB
LazyInitializationException.class	2015/08/05 20:54	CLASS 文件	1 KB
LobHelper.class	2015/08/05 20:54	CLASS 文件	1 KB
LockMode.class	2015/08/05 20:54	CLASS 文件	3 KB

编写XML时候的提示文件

IED 配置 DTD

第一步:自定目录存放 DTD

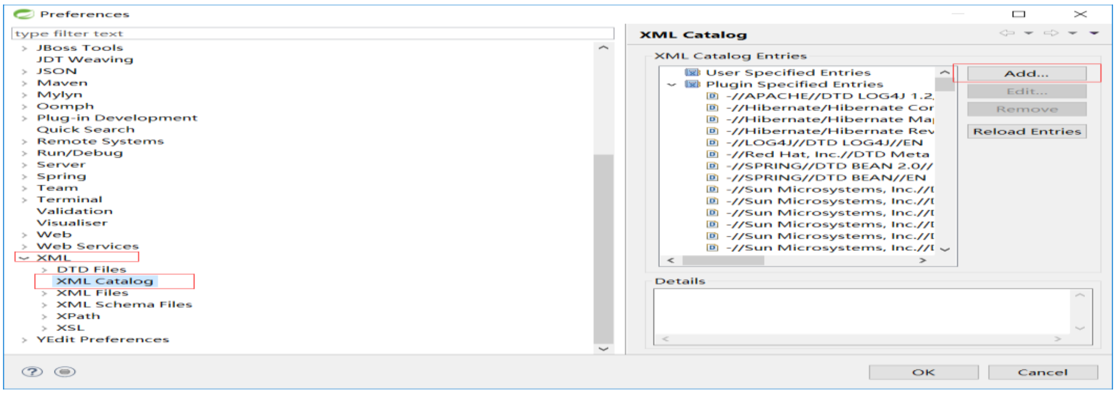


F:\dtd

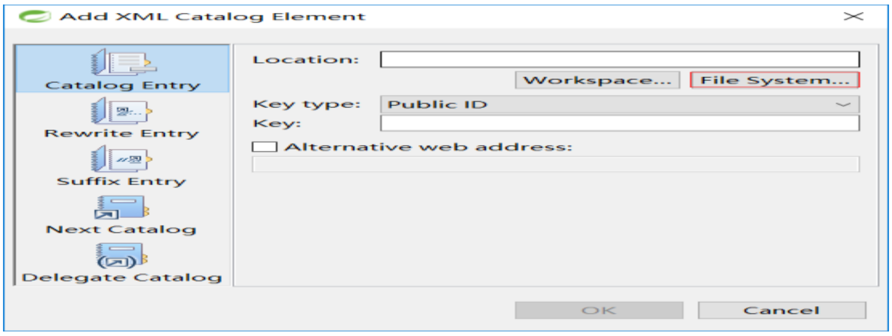
名称	修改日期	类型	大小
hibernate-configuration-3.0.dtd	2015/08/05 20:54	XML Document ...	3 KB
hibernate-configuration-4.0.xsd	2015/08/05 20:54	XSD 文件	7 KB
hibernate-mapping-3.0.dtd	2015/08/05 20:54	XML Document ...	45 KB
hibernate-mapping-4.0.xsd	2015/08/05 20:54	XSD 文件	95 KB

第二步:在 IDE 中配置 DTD

Window→preferences→XML→XML Cata Log



```
-//APACHE//DTD LOG4J 1.2,
-//Hibernate/Hibernate Cor
-//Hibernate/Hibernate Ma|
-//Hibernate/Hibernate Rev
-//LOG4J//DTD LOG4J//EN
-//Red Hat, Inc.//DTD Meta
-//SPRING//DTD BEAN 2.0//
-//SPRING//DTD BEAN//EN
-//Sun Microsystems, Inc.//I
-//Sun Microsystems, Inc.//I
-//Sun Microsystems, Inc.//I
-//Sun Microsystems, Inc.//I
-//Sun Microsystems, Inc.//I
-//Sun Microsystems, Inc.//I
```



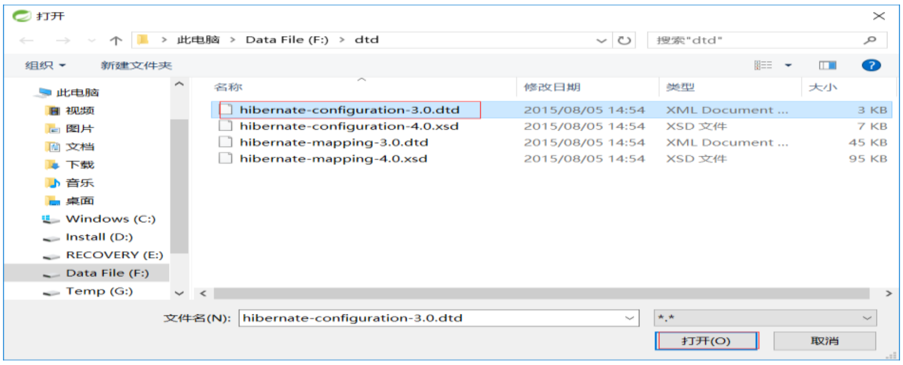
Add XML Catalog Element

Location:

Key type: Public ID

Key:

□ Alternative web address:



打开

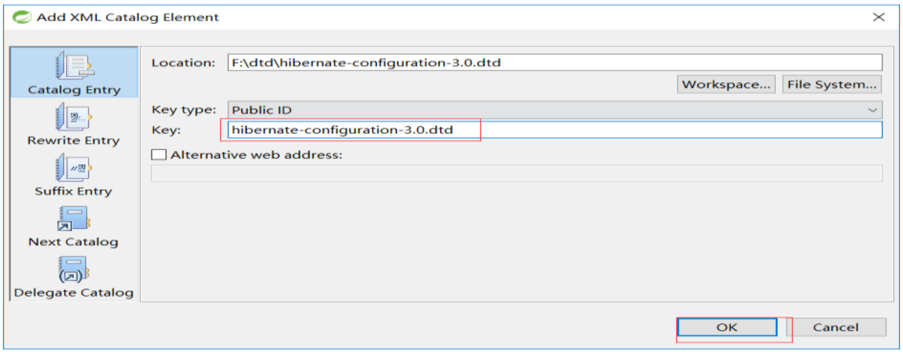
此电脑 > Data File (F:) > dtd

搜索"dtd"

名称	修改日期	类型	大小
hibernate-configuration-3.0.dtd	2015/08/05 14:54	XML Document ...	3 KB
hibernate-configuration-4.0.xsd	2015/08/05 14:54	XSD 文件	7 KB
hibernate-mapping-3.0.dtd	2015/08/05 14:54	XML Document ...	45 KB
hibernate-mapping-4.0.xsd	2015/08/05 14:54	XSD 文件	95 KB

文件名(N): hibernate-configuration-3.0.dtd

.



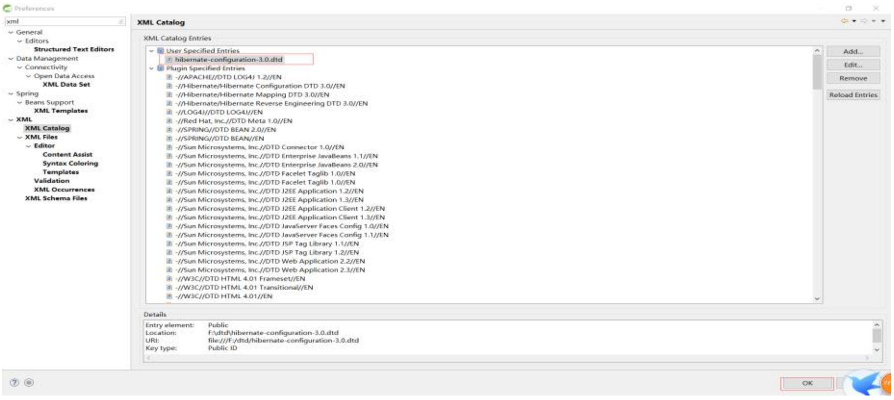
Add XML Catalog Element

Location: F:\dtd\hibernate-configuration-3.0.dtd

Key type: Public ID

Key: hibernate-configuration-3.0.dtd

□ Alternative web address:



Preferences

xml

XML Catalog

XML Catalog Entries

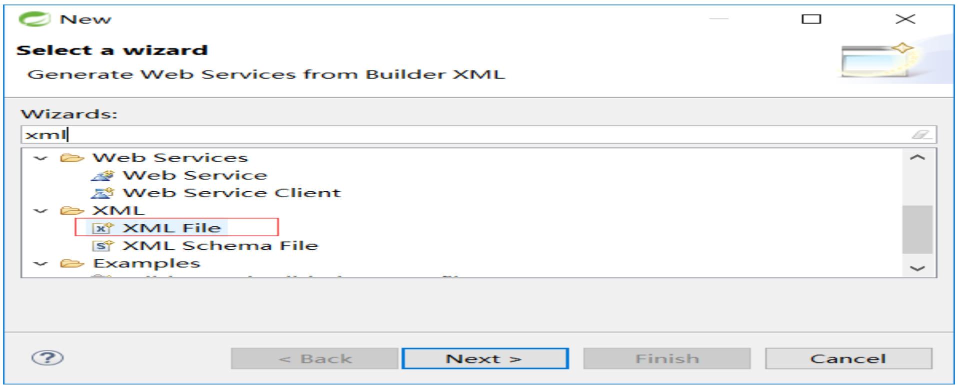
```
User Specified Entries
    hibernate-configuration-3.0.dtd
Plugin Specified Entries
    -//APACHE//DTD LOG4J 1.2//EN
    -//Hibernate/Hibernate Configuration DTD 3.0//EN
    -//Hibernate/Hibernate Mapping DTD 3.0//EN
    -//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN
    -//LOG4J//DTD LOG4J//EN
    -//Red Hat, Inc.//DTD Meta 1.0//EN
    -//SPRING//DTD BEAN 2.0//EN
    -//SPRING//DTD BEAN//EN
    -//Sun Microsystems, Inc.//DTD Connector 1.0//EN
    -//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN
    -//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN
    -//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN
    -//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN
    -//Sun Microsystems, Inc.//DTD J2EE Application 1.2//EN
    -//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN
    -//Sun Microsystems, Inc.//DTD J2EE Application Client 1.2//EN
    -//Sun Microsystems, Inc.//DTD J2EE Application Client 1.3//EN
    -//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN
    -//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN
    -//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN
    -//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN
    -//Sun Microsystems, Inc.//DTD Web Application 2.2//EN
    -//Sun Microsystems, Inc.//DTD Web Application 2.3//EN
    -//W3C//DTD HTML 4.01 Frameset//EN
    -//W3C//DTD HTML 4.01 Transitional//EN
    -//W3C//DTD HTML 4.01//EN
```

Details

```
Entry element:   Public
Location:        F:\dtd\hibernate-configuration-3.0.dtd
URI:             file:///F:/dtd/hibernate-configuration-3.0.dtd
Key type:        Public ID
```

配置 hibernate.cfg.xml

第一步构建 hibernate.cfg.xml

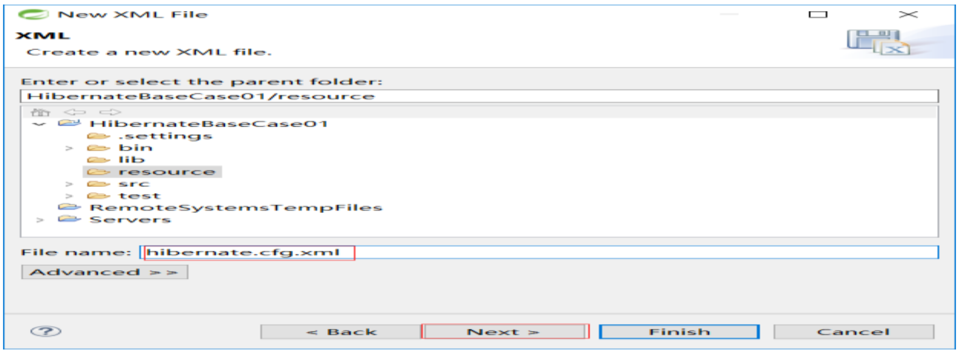


New

Select a wizard

Generate Web Services from Builder XML

Wizards: xml



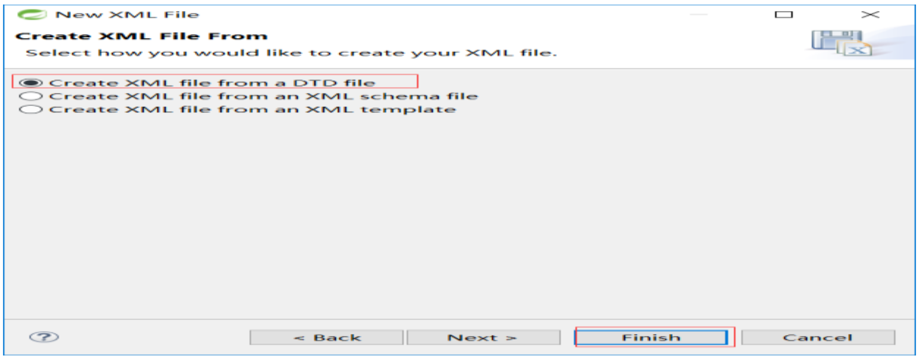
New XML File

XML

Create a new XML file.

Enter or select the parent folder: HibernateBaseCase01/resource

File name: hibernate.cfg.xml

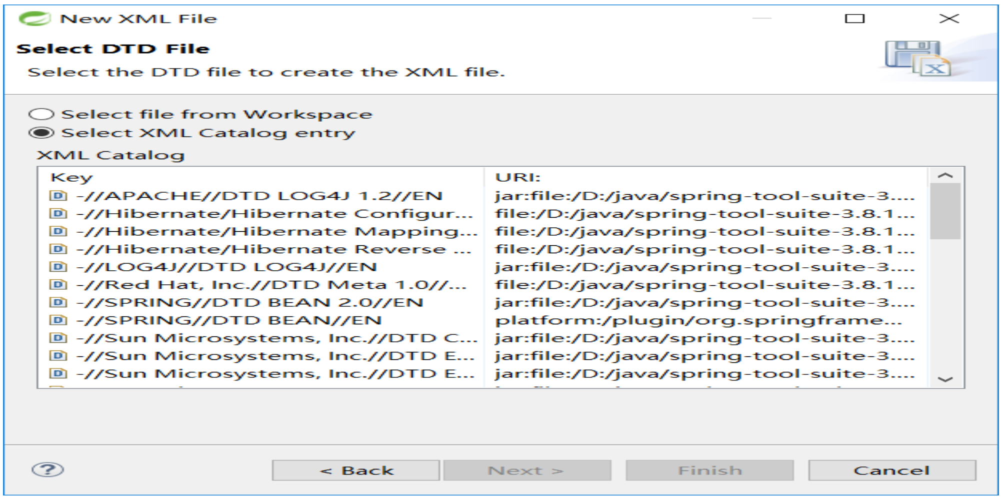


New XML File

Create XML File From

Select how you would like to create your XML file.

● Create XML file from a DTD file
○ Create XML file from an XML schema file
○ Create XML file from an XML template



New XML File

Select DTD File

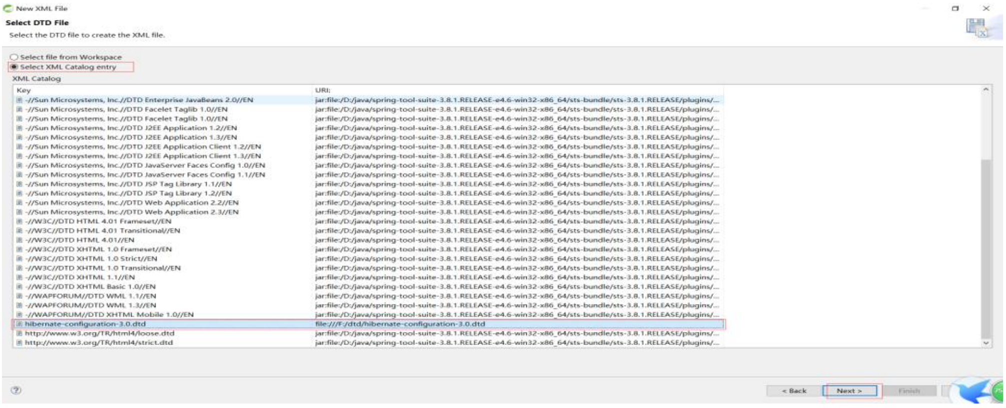
Select the DTD file to create the XML file.

○ Select file from Workspace

● Select XML Catalog entry

XML Catalog

Key	URI:
-//APACHE//DTD LOG4J 1.2//EN	jar:file:/D:/java/spring-tool-suite-3....
-//Hibernate/Hibernate Configur...	file:/D:/java/spring-tool-suite-3.8.1...
-//Hibernate/Hibernate Mapping...	file:/D:/java/spring-tool-suite-3.8.1...
-//Hibernate/Hibernate Reverse ...	file:/D:/java/spring-tool-suite-3.8.1...
-//LOG4J//DTD LOG4J//EN	jar:file:/D:/java/spring-tool-suite-3....
-//Red Hat, Inc.//DTD Meta 1.0//...	file:/D:/java/spring-tool-suite-3.8.1...
-//SPRING//DTD BEAN 2.0//EN	jar:file:/D:/java/spring-tool-suite-3....
-//SPRING//DTD BEAN//EN	platform:/plugin/org.springframe...
-//Sun Microsystems, Inc.//DTD C...	jar:file:/D:/java/spring-tool-suite-3....
-//Sun Microsystems, Inc.//DTD E...	jar:file:/D:/java/spring-tool-suite-3....
-//Sun Microsystems, Inc.//DTD E...	jar:file:/D:/java/spring-tool-suite-3....



New XML File

Select DTD File

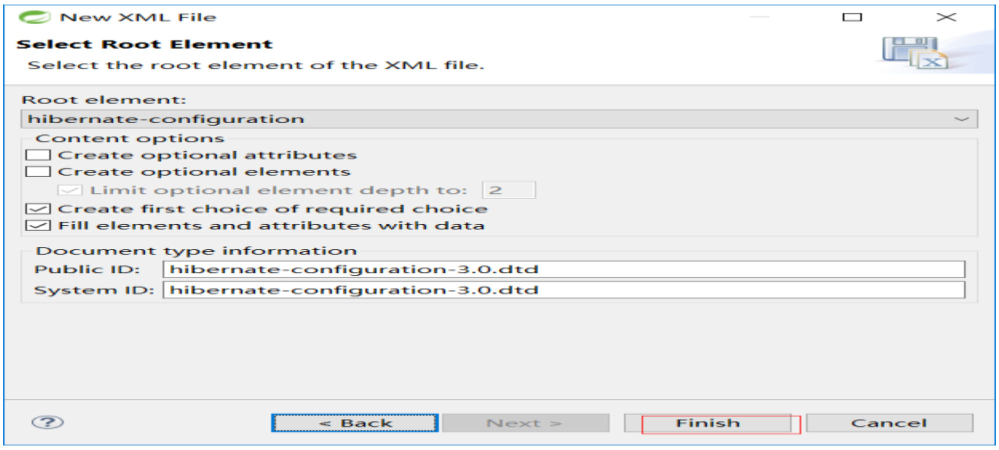
Select the DTD file to create the XML file.

○ Select file from Workspace

● Select XML Catalog entry

XML Catalog

Key	URI:
-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD J2EE Application 1.2//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD J2EE Application Client 1.2//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD J2EE Application Client 1.3//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD HTML 4.01 Frameset//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD HTML 4.01 Transitional//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD HTML 4.01//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD XHTML 1.0 Frameset//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD XHTML 1.0 Strict//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD XHTML 1.0 Transitional//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD XHTML 1.1//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//W3C//DTD XHTML Basic 1.0//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//WAPFORUM//DTD WML 1.1//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//WAPFORUM//DTD WML 1.3//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
-//WAPFORUM//DTD XHTML Mobile 1.0//EN	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
hibernate-configuration-3.0.dtd	file:///F:/dtd/hibernate-configuration-3.0.dtd
http://www.w3.org/TR/html4/loose.dtd	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...
http://www.w3.org/TR/html4/strict.dtd	jar:file:/D:/java/spring-tool-suite-3.8.1.RELEASE-e4.6-win32-x86_64/sts-bundle/sts-3.8.1.RELEASE/plugins/...



New XML File

Select Root Element

Select the root element of the XML file.

Root element: hibernate-configuration

Content options

□ Create optional attributes

□ Create optional elements

☑ Limit optional element depth to: 2

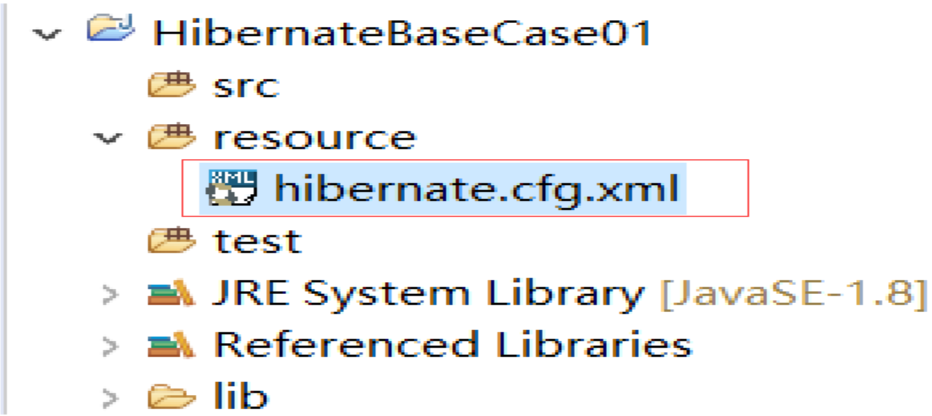
☑ Create first choice of required choice

☑ Fill elements and attributes with data

Document type information

Public ID: hibernate-configuration-3.0.dtd

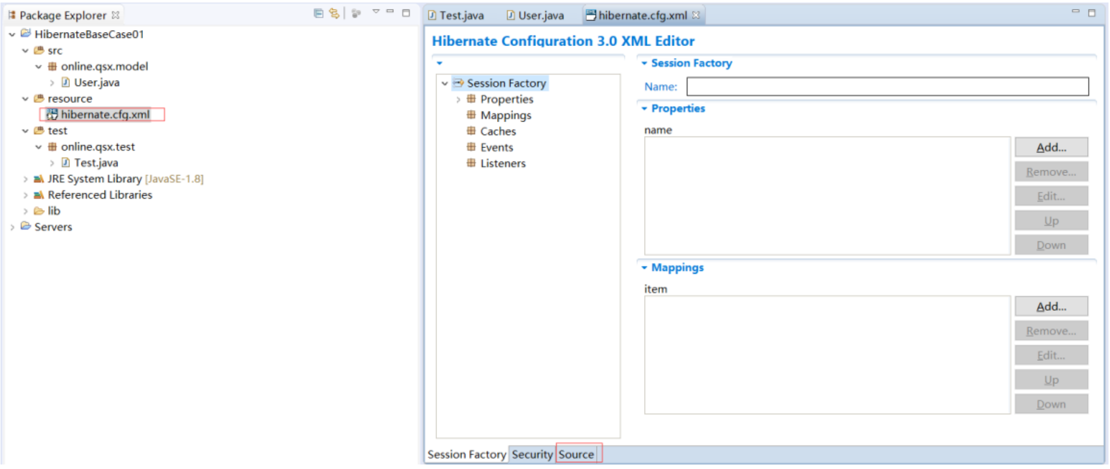
System ID: hibernate-configuration-3.0.dtd



HibernateBaseCase01
src
resource
hibernate.cfg.xml
test
JRE System Library [JavaSE-1.8]
Referenced Libraries
lib

第二步编辑 **hibernate.cfg.xml[数据库连接信息]**

进入编辑页面



Hibernate.cfg.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
       "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
     "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory/>
</hibernate-configuration>
```

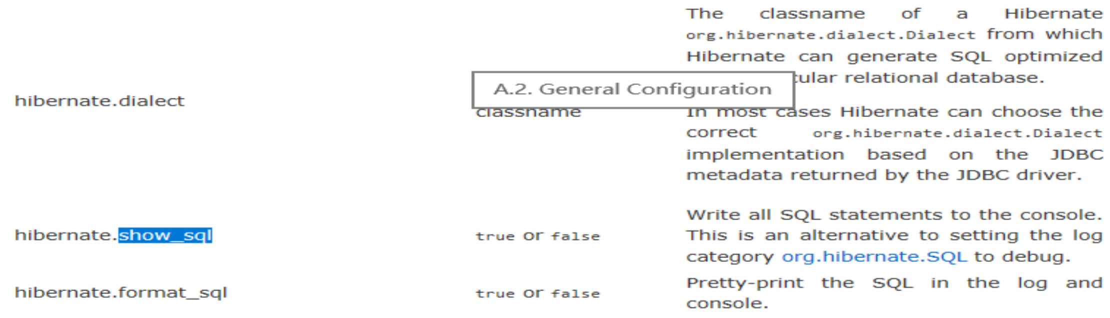
配置 hibernate.cfg.xml 数据库连接信息

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
       "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
     "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
    <session-factory>
```

```
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property
name="connection.url">jdbc:mysql://localhost:3306/myschool</property
>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

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

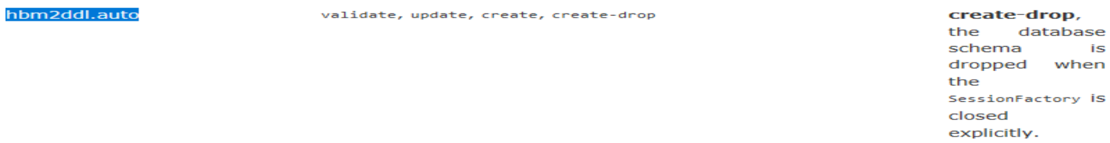
第三步编辑 hibernate.cfg.xml[hibernate 参数控制]

hibernate 参数官网描述截图



A.2. General Configuration

hibernate.dialect	classname	The classname of a Hibernate org.hibernate.dialect.Dialect from which Hibernate can generate SQL optimized ...ular relational database. In most cases Hibernate can choose the correct org.hibernate.dialect.Dialect implementation based on the JDBC metadata returned by the JDBC driver.
hibernate.show_sql	true or false	Write all SQL statements to the console. This is an alternative to setting the log category org.hibernate.SQL to debug.
hibernate.format_sql	true or false	Pretty-print the SQL in the log and console.



hbm2ddl.auto	validate, update, create, create-drop	create-drop, the database schema is dropped when the SessionFactory is closed explicitly.

hibernate.cfg.xml 参数配置

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property
name="connection.url">jdbc:mysql://localhost:3306/myschool</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>
        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property><!-- sql正对的
是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示sql -->
        <property name="hibernate.format_sql">true</property><!-- sql格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->

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

第四步编辑 hibernate.cfg.xml[hibernate 表的映射信息]

Hibernate.cfg.xml 配置

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property
name="connection.url">jdbc:mysql://localhost:3306/myschool</property
>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</proper
ty><!-- sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示
sql -->
        <property name="hibernate.format_sql">true</property><!-- sql
格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][注解]-->
        <mapping class="online.qsx.model.User"/>

    </session-factory>
```

```
</hibernate-configuration>
```

注解实习实体映射配置

```
package online.qsx.model;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "tb_user")
public class User {
	@Id //  主键
	@GeneratedValue(strategy = GenerationType.IDENTITY) //  自增
	@Column(name = "user_id")
	private Long id;

	@Column(name = "user_name", unique = true)
	private String name;

	@Column(name = "user_sex")
	private Short sex;

	@Column(name = "user_age")
	private Short age;

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

	// TemporalType.DATE yyyy-MM-dd
	// TemporalType.TIME yyyy-MM-dd hh:mm:ss
	@Temporal(TemporalType.DATE)
	@Column(name = "user_birthday")
	private Date birthdayDate;

	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 Short getSex() {
        return sex;
    }

    public void setSex(Short sex) {
        this.sex = sex;
    }

    public Short getAge() {
        return age;
    }

    public void setAge(Short age) {
        this.age = age;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public Date getBirthdayDate() {
        return birthdayDate;
    }

    public void setBirthdayDate(Date birthdayDate) {
        this.birthdayDate = birthdayDate;
    }
}
```

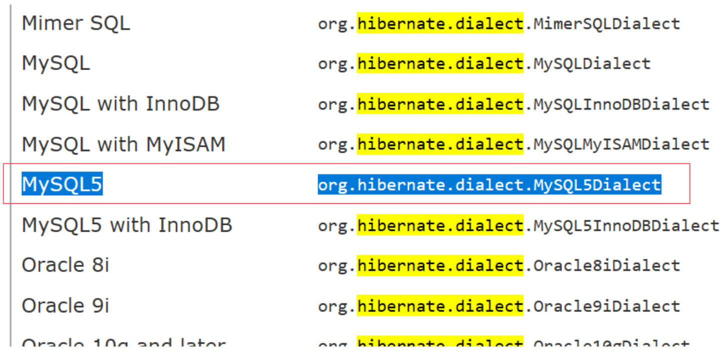
第五步测试

```
package online.qsx.test;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

public class Test {
    public static void main(String[] args) {
        //加载配置文件
        Configuration configuration=new Configuration();
        configuration.configure("hibernate.cfg.xml");

        //注册标准服务
        StandardServiceRegistryBuilder ssrb=new StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr=ssrb.applySettings(configuration.getProperties()).build();

        //通过标准服务加载配置文件后获得会话工厂
        SessionFactory sf=configuration.buildSessionFactory(ssr);//二级缓存

        //开启一个会话
        Session session=sf.openSession();//一级缓存

        //操作
        System.out.println("连接开启成功");

        //关闭
        session.close();
        sf.close();

        System.out.println("连接关闭成功");

    }
}
```

搭建 Hibernate 实现 DDL 扩展

MySQL 表类型/解决 MySQL 表字符集的问题

扩展 hibernate API



Mimer SQL	org.hibernate.dialect.MimerSQLDialect
MySQL	org.hibernate.dialect.MySQLDialect
MySQL with InnoDB	org.hibernate.dialect.MySQLInnoDBDialect
MySQL with MyISAM	org.hibernate.dialect.MySQLMyISAMDialect
MySQL5	org.hibernate.dialect.MySQL5Dialect
MySQL5 with InnoDB	org.hibernate.dialect.MySQL5InnoDBDialect
Oracle 8i	org.hibernate.dialect.Oracle8iDialect
Oracle 9i	org.hibernate.dialect.Oracle9iDialect
Oracle 10g and later	org.hibernate.dialect.Oracle10gDialect

实现扩展

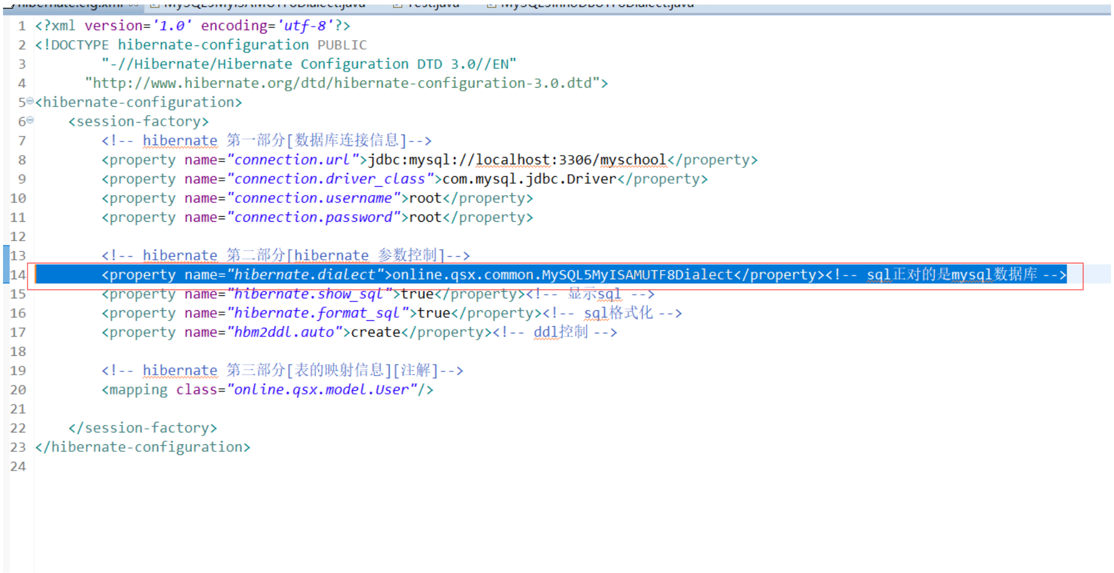
InnoDB utf-8

```
package online.qsx.common;

import org.hibernate.dialect.MySQL5Dialect;

public class MySQL5InnoDBUTF8Dialect extends MySQL5Dialect {

    @Override
    public String getTableTypeString() {
        return "ENGINE=InnoDB CHARSET=utf8";
    }

}
```

MyISAM utf-8

```
package online.qsx.common;

import org.hibernate.dialect.MySQL5Dialect;

public class MySQL5MyISAMUTF8Dialect extends MySQL5Dialect {

    @Override
    public String getTableTypeString() {
        return "ENGINE=MyISAM CHARSET=utf8";
    }

}
```

hibernate.cfg.xml



```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property name="connection.url">jdbc:mysql://localhost:3306/myschool</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <property name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</property><!-- sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示sql -->
        <property name="hibernate.format_sql">true</property><!-- sql格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][注解]-->
        <mapping class="online.qsx.model.User"/>

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

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property name="connection.url">jdbc:mysql://localhost:3306/myschool</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <property name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</property><!-- sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示
```

```
sql -->
        <property name="hibernate.format_sql">true</property><!-- sql
格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][注解]-->
        <mapping class="online.qsx.model.User"/>

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

User.java

```
package online.qsx.model;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "tb_user")
public class User {
	@Id //  主键
	@GeneratedValue(strategy = GenerationType.IDENTITY) //  自增
	@Column(name = "user_id")
	private Long id;

	@Column(name = "user_name", unique = true)
	private String name;

	@Column(name = "user_sex")
	private Short sex;

	@Column(name = "user_age")
	private Short age;
```

```
@Column(name = "user_phone")
private String phone;

// TemporalType.DATE yyyy-MM-dd
// TemporalType.TIME yyyy-MM-dd hh:mm:ss
@Temporal(TemporalType.DATE)
@Column(name = "user_birthday")
private Date birthdayDate;

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 Short getSex() {
    return sex;
}

public void setSex(Short sex) {
    this.sex = sex;
}

public Short getAge() {
    return age;
}

public void setAge(Short age) {
    this.age = age;
}

public String getPhone() {
    return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public Date getBirthdayDate() {
        return birthdayDate;
    }

    public void setBirthdayDate(Date birthdayDate) {
        this.birthdayDate = birthdayDate;
    }
}
```

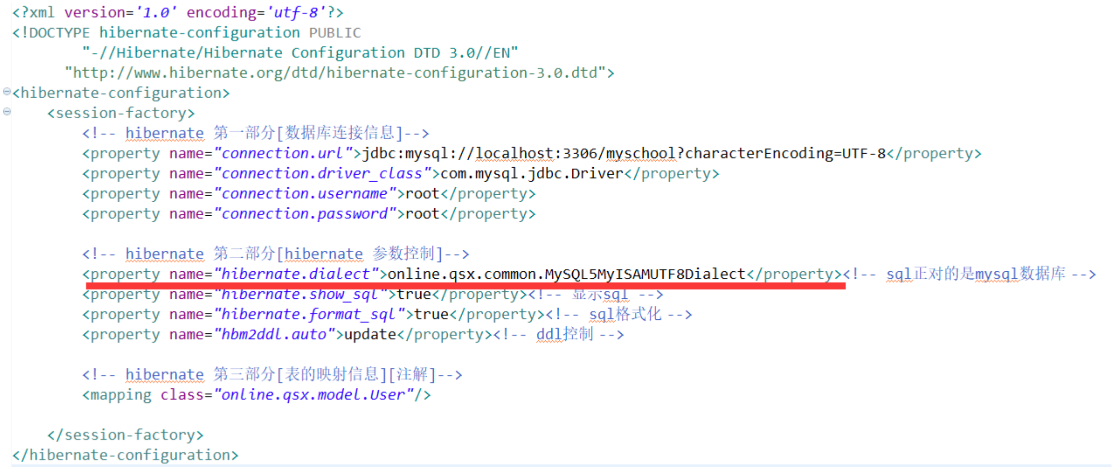
Test.java

```
package online.qsx.test;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

public class Test {
    public static void main(String[] args) {
        //加载配置文件
        Configuration configuration=new Configuration();
        configuration.configure("hibernate.cfg.xml");

        //注册标准服务
        StandardServiceRegistryBuilder ssrb=new StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr=ssrb.applySettings(configuration.getProperties()).build();

        //通过标准服务加载配置文件后获得会话工厂
        SessionFactory sf=configuration.buildSessionFactory(ssr);//二级缓存

        //开启一个会话
        Session session=sf.openSession();//一级缓存

        //操作
```

```
            System.out.println("连接开启成功");

            //关闭
            session.close();
            sf.close();

            System.out.println("连接关闭成功");

     }
}
```

使用 Hibernate 完成 CRUD

Hibernate 添加时中文乱码处理



```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property name="connection.url">jdbc:mysql://localhost:3306/myschool?characterEncoding=UTF-8</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <property name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</property><!-- sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示sql -->
        <property name="hibernate.format_sql">true</property><!-- sql格式化 -->
        <property name="hbm2ddl.auto">update</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][注解]-->
        <mapping class="online.qsx.model.User"/>

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

优化代码抽取方法

```
package online.qsx.test;

import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.User;
```

```
public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     * 开启hibernate连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new
StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr =
ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        //开启事物
        transaction=session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭hibernate连接
     */
    public void destroy() {
        //提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }
}
```

save

```
    /**
     * 添加 save
     */
    public void save() {
        init();
        User user =new User("1111", Short.valueOf("1"),
Short.valueOf("1"), "111111", new Date());
        session.save(user);
        destroy();
    }
```

```
Hibernate:
    insert
    into
        tb_user
        (user_age, user_birthday, user_name, user_phone, user_sex)
    values
        (?, ?, ?, ?, ?)
```

update

```
    /**
     * 添加 update
     */
    public void update() {
        init();
        User user =new User(3L,"66666", Short.valueOf("1"),
Short.valueOf("1"), "111111", new Date());
        session.update(user);
        destroy();
    }
```

```
Hibernate:
    update
        tb_user
    set
        user_age=?,
        user_birthday=?,
        user_name=?,
        user_phone=?,
        user_sex=?
    where
        user_id=?
```

saveOrUpdate

```
    /**
     * 添加 save
     */
    public void saveOrUpdate_save() {
        init();
        User user =new User("2222", Short.valueOf("1"),
Short.valueOf("1"), "111111", new Date());
        session.saveOrUpdate(user);
        destroy();
    }
```

```
Hibernate:
    insert
    into
        tb_user
        (user_age, user_birthday, user_name, user_phone, user_sex)
    values
        (?, ?, ?, ?, ?)
```

```
    /**
     * 添加 update
     */
    public void saveOrUpdate_update() {
        init();
        User user =new User(3L,"444", Short.valueOf("1"),
Short.valueOf("1"), "111111", new Date());
        session.saveOrUpdate(user);
        destroy();
    }
```

```
Hibernate:
    update
        tb_user
    set
        user_age=?,
        user_birthday=?,
        user_name=?,
        user_phone=?,
        user_sex=?
    where
```

```
        user_id=?
```

merge

```
    /**
     * 合并
     * save---->insert
     */
    public void merge_save(){
        init();
        User user =new User("123123", Short.valueOf("1"), 
Short.valueOf("1"), "111111", new Date());
        session.merge(user);
        destroy();
    }
```

```
Hibernate: 
    insert 
    into
        tb_user
        (user_age, user_birthday, user_name, user_phone, user_sex) 
    values
        (?, ?, ?, ?, ?)
```

```
    /**
     * 合并
     * update--->select update
     */
    public void merge_update(){
        init();
        User user =new User(1L,"123123", Short.valueOf("1"), 
Short.valueOf("1"), "111111", new Date());
        session.merge(user);
        destroy();
    }
```

```
Hibernate: 
    select
        user0_.user_id as user_id1_0_0_,
        user0_.user_age as user_age2_0_0_,
        user0_.user_birthday as user_bir3_0_0_,
        user0_.user_name as user_nam4_0_0_,
        user0_.user_phone as user_pho5_0_0_,
        user0_.user_sex as user_sex6_0_0_ 
    from
```

```
        tb_user user0_
    where
        user0_.user_id=?
Hibernate: 
    insert 
    into
        tb_user
        (user_age, user_birthday, user_name, user_phone, user_sex) 
    values
        (?, ?, ?, ?, ?)
```

delete

```
    /**
     * 删除
     */
    public void delete(){
        init();
        session.delete(new User(1L));
        destroy();
    }
```

```
Hibernate: 
    delete 
    from
        tb_user 
    where
        user_id=?
```

get

```
    /**
     * 查询 get 立即加载  查询的时候立刻加载数据
     */
    public void get(){
        init();
        User user=(User) session.get(User.class, 1L);
        System.out.println(user.toString());
        destroy();
    }
```

```
Hibernate: 
	select
		user0_.user_id as user_id1_0_0_,
		user0_.user_age as user_age2_0_0_,
		user0_.user_birthday as user_bir3_0_0_,
		user0_.user_name as user_nam4_0_0_,
		user0_.user_phone as user_pho5_0_0_,
		user0_.user_sex as user_sex6_0_0_
	from
		tb_user user0_
	where
		user0_.user_id=?
User [id=1, name=张三, sex=1, age=12, phone=15384562145, birthdayDate=2017-09-20]
```

load

```
	/**
	 * 查询 load 懒加载  使用数据的时候才去加载数据
	 */
	public void load(){
		init();
		User user=(User) session.load(User.class, 1L);
		System.out.println(user.toString());
		destroy();
	}
```

```
Hibernate:
	select
		user0_.user_id as user_id1_0_0_,
		user0_.user_age as user_age2_0_0_,
		user0_.user_birthday as user_bir3_0_0_,
		user0_.user_name as user_nam4_0_0_,
		user0_.user_phone as user_pho5_0_0_,
		user0_.user_sex as user_sex6_0_0_
	from
		tb_user user0_
	where
		user0_.user_id=?
User [id=1, name=张三, sex=1, age=12, phone=15384562145, birthdayDate=2017-09-20]
```

课程作业:

平台推送

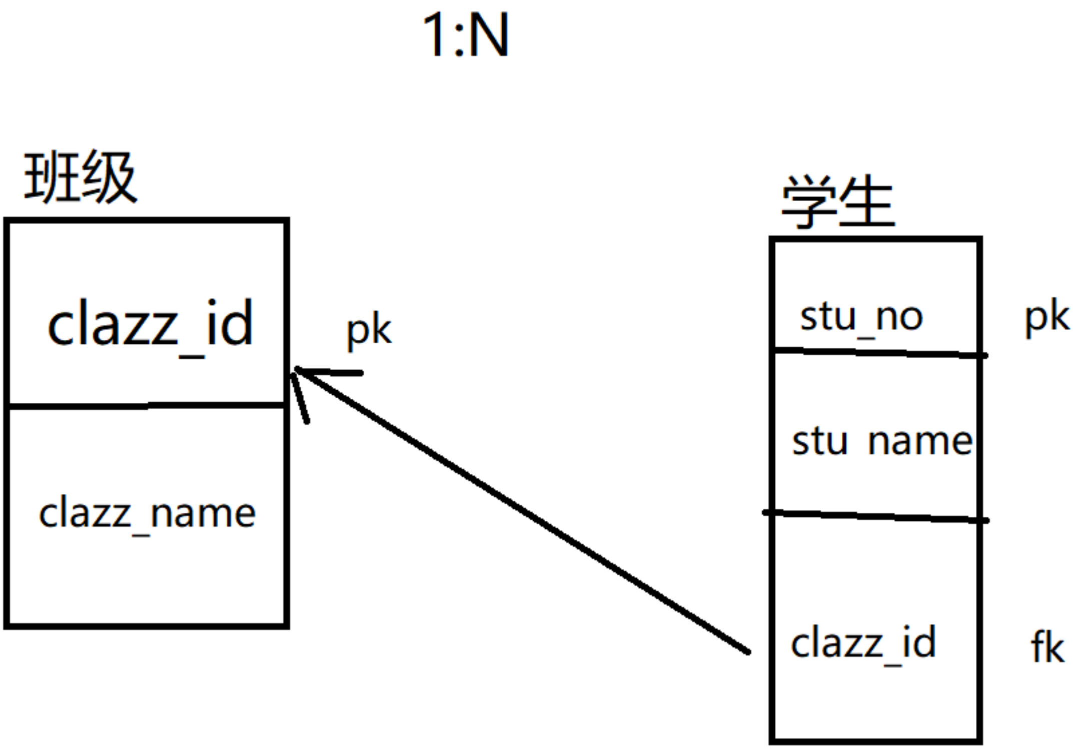
利用 **Hibernate** 实现商品数据的增删改查

Day02:初始 Hibernate 关系映射

课程目标

单项一对多关系映射

图解



代码演示

Student.java

```
package online.qsx.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "tb_student")
public class Student {

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

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

    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 Student(Long id, String name) {
        super();
        this.id = id;
        this.name = name;
    }

    public Student(String name) {
        super();
        this.name = name;
    }

    public Student(Long id) {
        super();
        this.id = id;
    }

    public Student() {
        super();
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + "]";
    }

}
```

Clazz.java

```
package online.qsx.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;

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

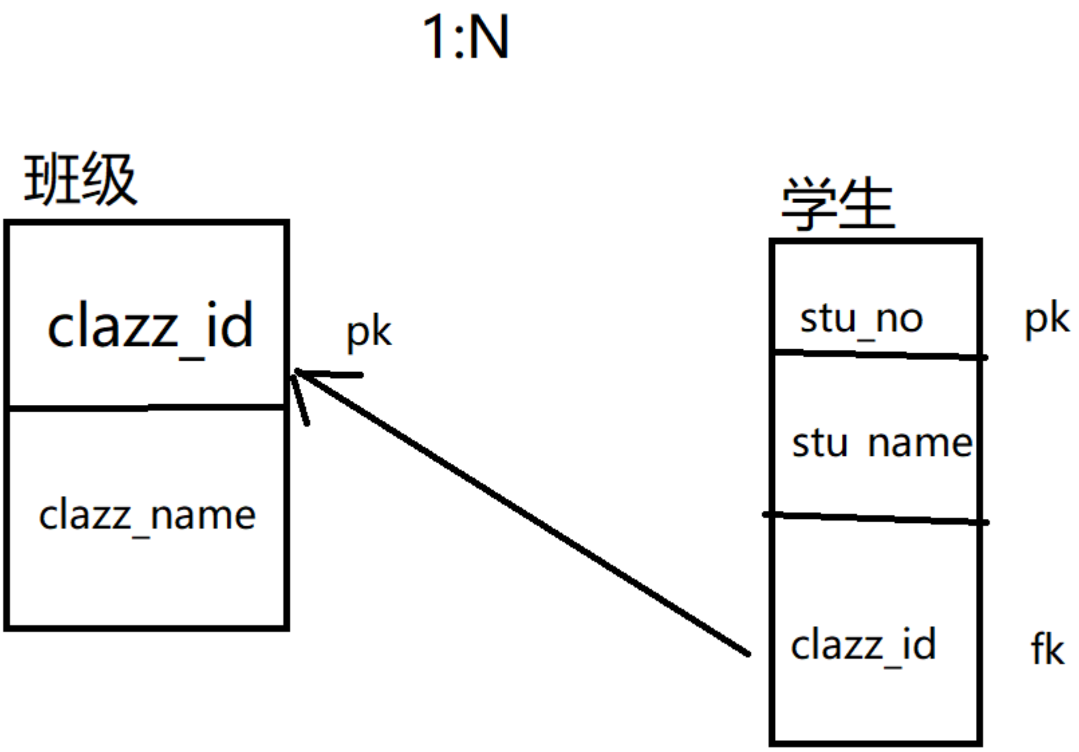
```
public class Clazz {

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

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

    // 特殊属性
    // 1 : N
    @OneToMany
    @JoinColumn(name = "clazz_id")
    private Set<Student> students = new HashSet<Student>();

    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 Set<Student> getStudents() {
        return students;
    }

    public void setStudents(Set<Student> students) {
        this.students = students;
    }

    public Clazz(Long id, String name) {
        super();
        this.id = id;
        this.name = name;
    }

    public Clazz(String name) {
        super();
        this.name = name;
    }

    public Clazz(Long id) {
        super();
        this.id = id;
    }

    public Clazz() {
        super();
    }

    @Override
    public String toString() {
        return "Clazz [id=" + id + ", name=" + name + "]";
    }

    public String toStringAndStudents() {
        return "Clazz [id=" + id + ", name=" + name + ",students=" + students + "]";
    }

}
```

Test.java

```
package online.qsx.test;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Clazz;
import online.qsx.model.Student;

public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     * 开启hibernate连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new
StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr =
ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭hibernate连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    /**
     * 级联添加
     */
    public void save(){
        init();
        System.out.println("构建表结构");
        destroy();
    }
    public static void main(String[] args) {
        Test test = new Test();
        test.save();
    }
}
```

单项多对一关系映射

图解



代码演示

Student.java

```
package online.qsx.model;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "tb_student")
// N
public class Student {

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

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

    // 特殊属性
    @ManyToOne(cascade = { CascadeType.ALL }, fetch = FetchType.EAGER)
    @JoinColumn(name = "clazz_id")
    private Clazz clazz;

    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 Clazz getClazz() {
        return clazz;
    }

    public void setClazz(Clazz clazz) {
        this.clazz = clazz;
    }

    public Student(Long id, String name) {
        super();
        this.id = id;
        this.name = name;
    }

    public Student(String name) {
        super();
        this.name = name;
    }

    public Student(Long id) {
        super();
        this.id = id;
    }

    public Student() {
        super();
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + "]";
    }

}
```

Clazz.java

```
package online.qsx.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
```

```
@Entity
@Table(name = "tb_clazz")
// 1
public class Clazz {

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

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

    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 Clazz(Long id, String name) {
        super();
        this.id = id;
        this.name = name;
    }

    public Clazz(String name) {
        super();
        this.name = name;
    }

    public Clazz(Long id) {
        super();
        this.id = id;
    }

    public Clazz() {
        super();
    }

    @Override
    public String toString() {
        return "Clazz [id=" + id + ", name=" + name + "]";
    }
}
```

Test.java

```
package online.qsx.test;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Clazz;
import online.qsx.model.Student;

public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     * 开启 hibernate 连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr = ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭 hibernate 连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    /**
     * 级联添加
     */
    public void save() {
        init();
        Student stu=new Student("张三");
        stu.setClazz(new Clazz("大四"));
        session.save(stu);
        destroy();
    }



    public static void main(String[] args) {
        Test test = new Test();
        test.save();
    }
}
```

懒加载,立即加载

Test.java and Clazz.java

懒加载 fetch = FetchType.LAZY

```
    // 特殊属性
    // 1 : N
    // fetch 控制 查询
    @OneToMany(fetch = FetchType.LAZY)
    @JoinColumn(name = "clazz_id")
    private Set<Student> students = new HashSet<Student>();
```

```
    /**
     * 懒加载  fetch = FetchType.LAZY
     */
    public void getClazz(){
        init();
        Clazz clazz=(Clazz)session.get(Clazz.class, 4L);
        System.out.println(clazz.toStringAndStudents());
        destroy();
    }
```

```
Hibernate: 
    select
        clazz0_.clazz_id as clazz_id1_0_0_,
        clazz0_.clazz_name as clazz_na2_0_0_ 
    from
        tb_clazz clazz0_ 
    where
        clazz0_.clazz_id=?
Hibernate: 
    select
        students0_.clazz_id as clazz_id3_0_0_,
        students0_.stu_no as stu_no1_1_0_,
        students0_.stu_no as stu_no1_1_1_,
        students0_.stu_name as stu_name2_1_1_ 
    from
        tb_student students0_ 
    where
        students0_.clazz_id=?
```

立即加载 fetch = FetchType.EAGER

```
    // 特殊属性
    // 1 : N
    // fetch 控制 查询
    @OneToMany(fetch = FetchType.EAGER)
    @JoinColumn(name = "clazz_id")
    private Set<Student> students = new HashSet<Student>();
```

```
    /**
     * 立即加载 fetch = FetchType.EAGER
     */
    public void getClazz(){
        init();
        Clazz clazz=(Clazz)session.get(Clazz.class, 4L);
        System.out.println(clazz.toStringAndStudents());
        destroy();
    }
```

```
Hibernate: 
    select
        clazz0_.clazz_id as clazz_id1_0_0_,
        clazz0_.clazz_name as clazz_na2_0_0_,
        students1_.clazz_id as clazz_id3_0_1_,
        students1_.stu_no as stu_no1_1_1_,
        students1_.stu_no as stu_no1_1_2_,
        students1_.stu_name as stu_name2_1_2_ 
    from
        tb_clazz clazz0_ 
    left outer join
        tb_student students1_ 
            on clazz0_.clazz_id=students1_.clazz_id 
    where
        clazz0_.clazz_id=?
```

一级缓存

数据来源走 session 缓存

```
/**
     * 查询的数据会存放到session中
     * 每次查询如果条件不变，一般先查session缓存,
     * 存在就不会发送SQL直接出数据,
     * 不存在才会去发送SQL查询结果，并把结果放置到session缓存
     */
    public void getClazz() {
        init();
        Clazz clazz1 = (Clazz) session.get(Clazz.class, 5L);
        System.out.println(clazz1.toStringAndStudents());

        Clazz clazz2 = (Clazz) session.get(Clazz.class, 5L);
        System.out.println(clazz2.toStringAndStudents());
        destroy();
    }
```

```
Hibernate: 
    select
        clazz0_.clazz_id as clazz_id1_0_0_,
        clazz0_.clazz_name as clazz_na2_0_0_,
        students1_.clazz_id as clazz_id3_0_1_,
        students1_.stu_no as stu_no1_1_1_,
        students1_.stu_no as stu_no1_1_2_,
        students1_.stu_name as stu_name2_1_2_ 
    from
        tb_clazz clazz0_ 
    left outer join
        tb_student students1_ 
            on clazz0_.clazz_id=students1_.clazz_id 
    where
        clazz0_.clazz_id=?
Clazz [id=5, name=大一,students=[Student [id=17, name=李四], Student [id=19, name=张三], Student [id=18, name=王五], Student [id=20, name=马六]]]
Clazz [id=5, name=大一,students=[Student [id=17, name=李四], Student [id=19, name=张三], Student [id=18, name=王五], Student [id=20, name=马六]]]
```

数据来源不走 session 缓存

```
    /**
     * 查询的数据会存放到session中
     * 每次查询如果条件不变，一般先查session缓存，
     * 存在就不会发送SQL直接出数据，
     * 不存在才会去发送SQL查询结果，并把结果放置到session缓存
     */
    public void getClazz() {
        init();
        Clazz clazz1 = (Clazz) session.get(Clazz.class, 5L);
        System.out.println(clazz1.toStringAndStudents());

        Clazz clazz2 = (Clazz) session.get(Clazz.class, 6L);
        System.out.println(clazz2.toStringAndStudents());
        destroy();
    }
```

```
Hibernate: 
    select
        clazz0_.clazz_id as clazz_id1_0_0_,
        clazz0_.clazz_name as clazz_na2_0_0_,
        students1_.clazz_id as clazz_id3_0_1_,
        students1_.stu_no as stu_no1_1_1_,
        students1_.stu_no as stu_no1_1_2_,
        students1_.stu_name as stu_name2_1_2_ 
    from
        tb_clazz clazz0_ 
    left outer join
        tb_student students1_ 
            on clazz0_.clazz_id=students1_.clazz_id 
    where
        clazz0_.clazz_id=?
Clazz [id=5, name=大一,students=[Student [id=17, name=李四], Student [id=19, name=张三], Student [id=18, name=王五], Student [id=20, name=马六]]]
Hibernate: 
    select
        clazz0_.clazz_id as clazz_id1_0_0_,
        clazz0_.clazz_name as clazz_na2_0_0_,
        students1_.clazz_id as clazz_id3_0_1_,
        students1_.stu_no as stu_no1_1_1_,
        students1_.stu_no as stu_no1_1_2_,
        students1_.stu_name as stu_name2_1_2_ 
    from
        tb_clazz clazz0_ 
    left outer join
        tb_student students1_ 
            on clazz0_.clazz_id=students1_.clazz_id 
    where
        clazz0_.clazz_id=?
Clazz [id=6, name=大二,students=[Student [id=22, name=张三], Student [id=23, name=马六], Student [id=21, name=李四], Student [id=24, name=王五]]]
```

级联关系

Student.java

```
package online.qsx.model;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "tb_student")
public class Student {

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

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

    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 Student(Long id, String name) {
        super();
        this.id = id;
        this.name = name;
    }

    public Student(String name) {
        super();
```

```
		this.name = name;
	}

	public Student(Long id) {
		super();
		this.id = id;
	}

	public Student() {
		super();
	}

	@Override
	public String toString() {
		return "Student [id=" + id + ", name=" + name + "]";
	}

}
```

Clazz.java

```
package online.qsx.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "tb_clazz")
public class Clazz {

	@Id
```

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

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

//  特殊属性
// 1 : N
// cascade  控制  添加,删除,修改
@OneToMany(cascade = { CascadeType.ALL })
@JoinColumn(name = "clazz_id")
private Set<Student> students = new HashSet<Student>();

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 Set<Student> getStudents() {
        return students;
}

public void setStudents(Set<Student> students) {
        this.students = students;
}

public Clazz(Long id, String name) {
        super();
        this.id = id;
        this.name = name;
}
```

```
    public Clazz(String name) {
        super();
        this.name = name;
    }

    public Clazz(Long id) {
        super();
        this.id = id;
    }

    public Clazz() {
        super();
    }

    @Override
    public String toString() {
        return "Clazz [id=" + id + ", name=" + name + "]";
    }

    public String toStringAndStudents() {
        return "Clazz [id=" + id + ", name=" + name + ",students=" + students + "]";
    }

}
```

Test.java

```
package online.qsx.test;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Clazz;
import online.qsx.model.Student;

public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     * 开启hibernate连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new
StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr =
ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭hibernate连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    /**
     * 级联添加
     */
    public void save() {
        init();
        Clazz clazz = new Clazz("大二");
        clazz.getStudents().add(new Student("张三"));
        clazz.getStudents().add(new Student("李四"));
        clazz.getStudents().add(new Student("王五"));
        clazz.getStudents().add(new Student("马六"));
        session.save(clazz);
        destroy();
    }

    public static void main(String[] args) {
        Test test = new Test();
        test.save();
    }
}
```

课程作业

平台推送

利用 Hibernate 实现一对多关联映射,使用 Annotation 配置

利用 Hibernate 实现多对一关联映射,使用 Annotation 配置

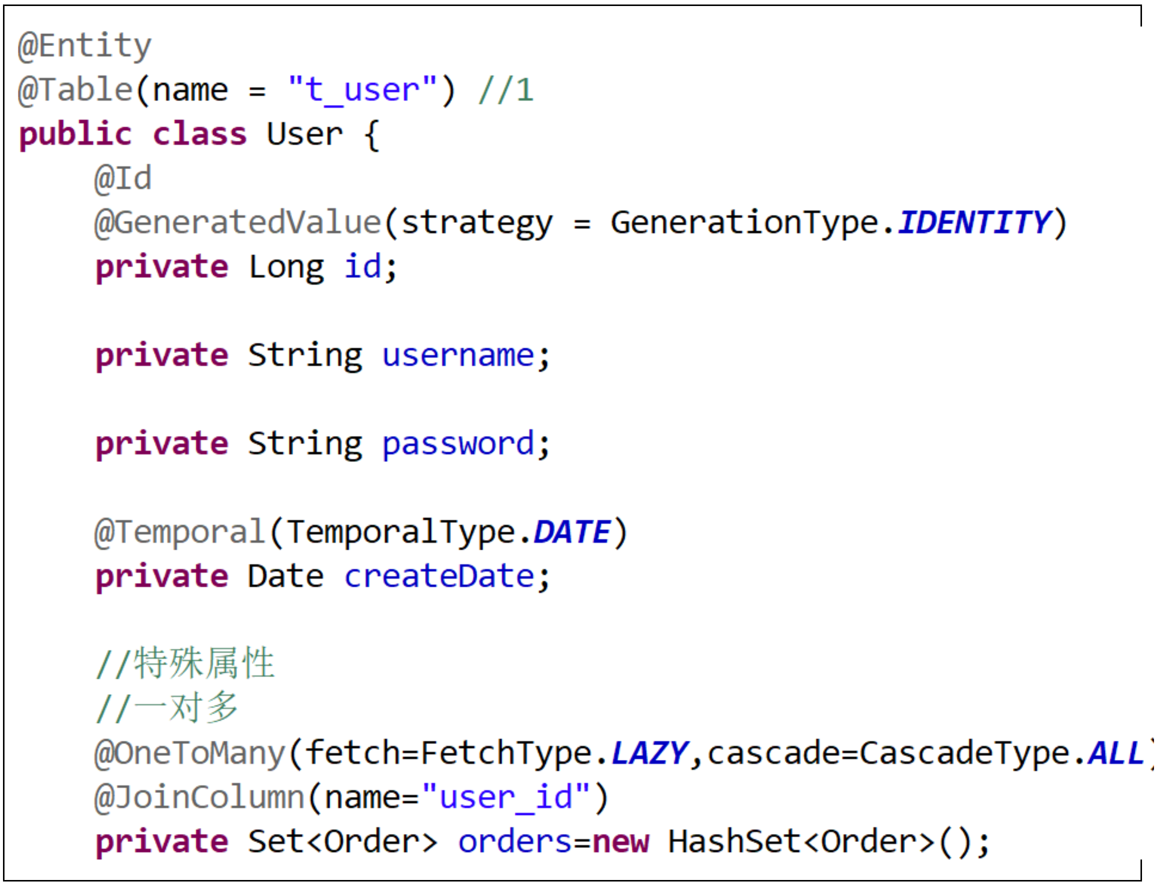
Day03:深入 Hibernate 关系映射

课程目标

双向一对多/多对一关系映射

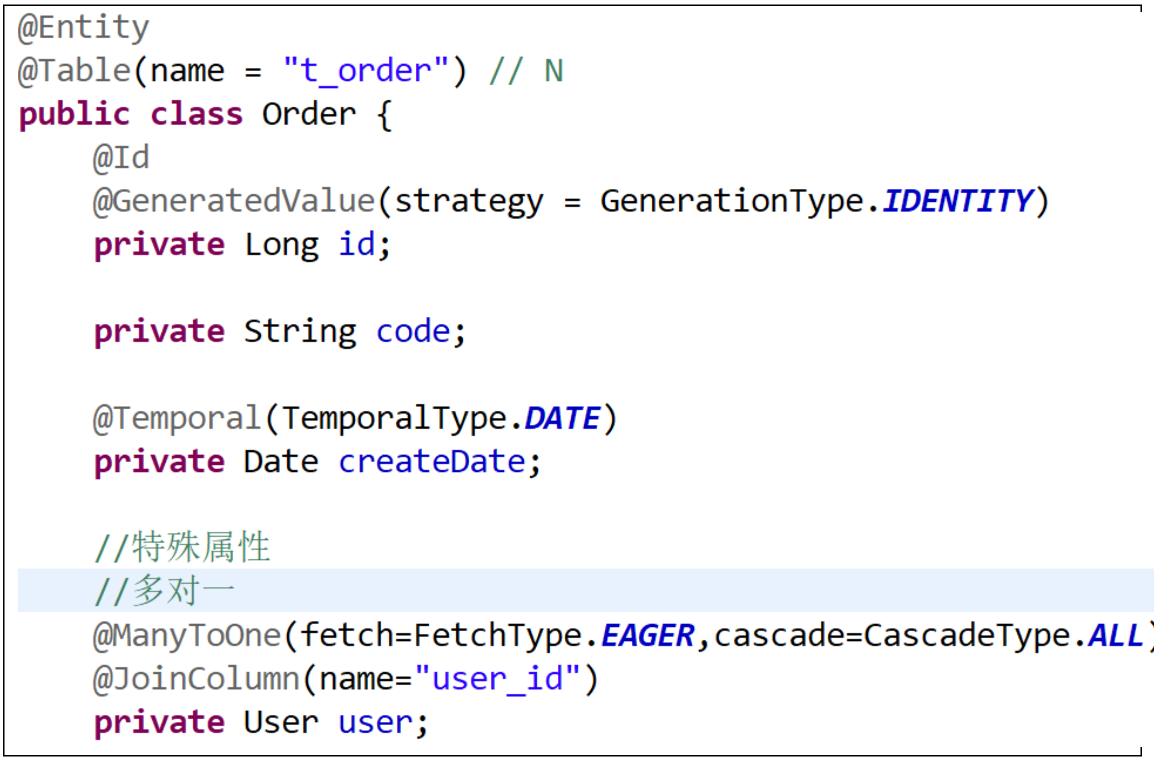
代码演示

User.java



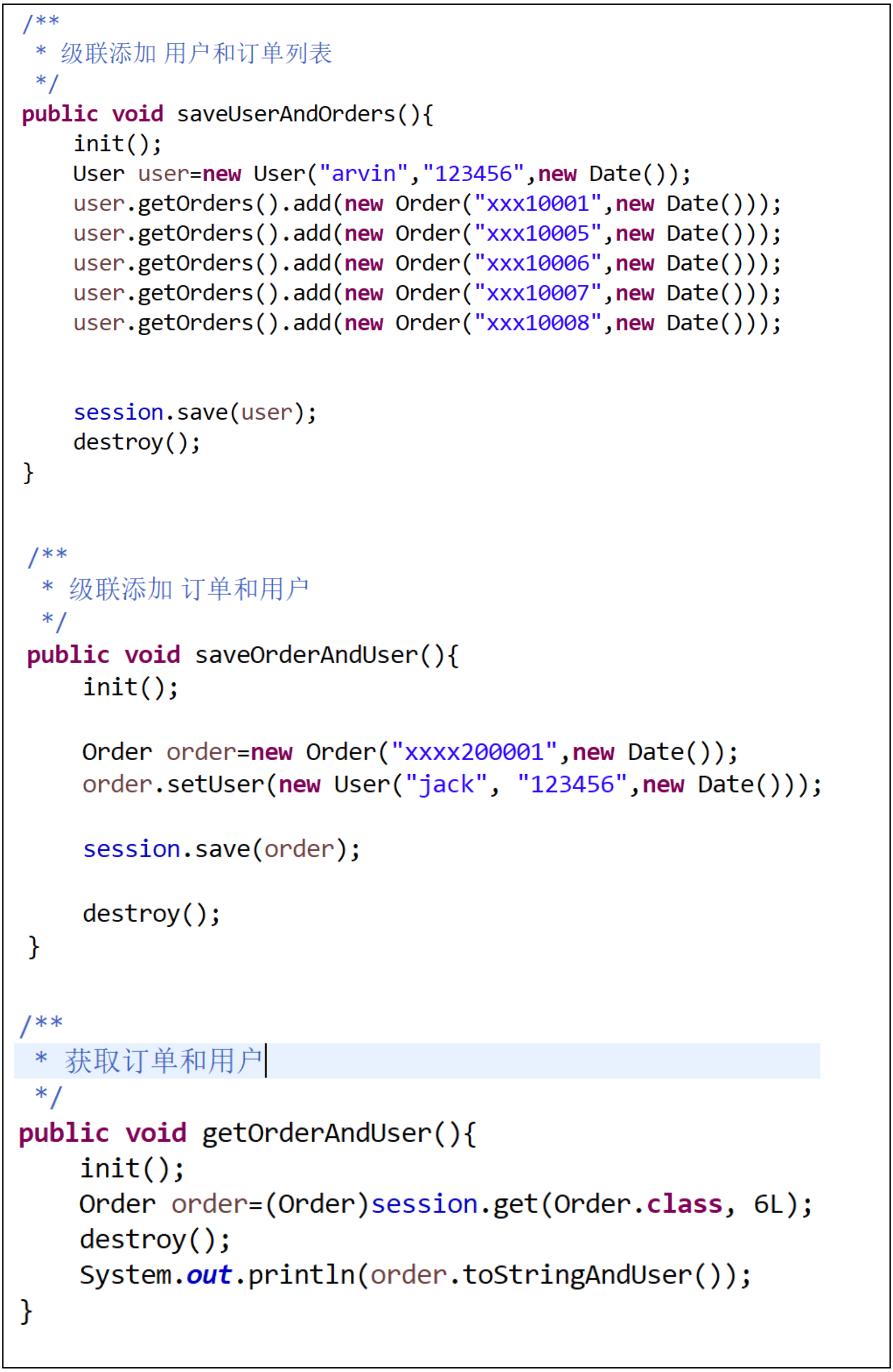
```java
@Entity
@Table(name = "t_user") //1
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;

    private String password;

    @Temporal(TemporalType.DATE)
    private Date createDate;

    //特殊属性
    //一对多
    @OneToMany(fetch=FetchType.LAZY,cascade=CascadeType.ALL)
    @JoinColumn(name="user_id")
    private Set<Order> orders=new HashSet<Order>();
```

Order.java

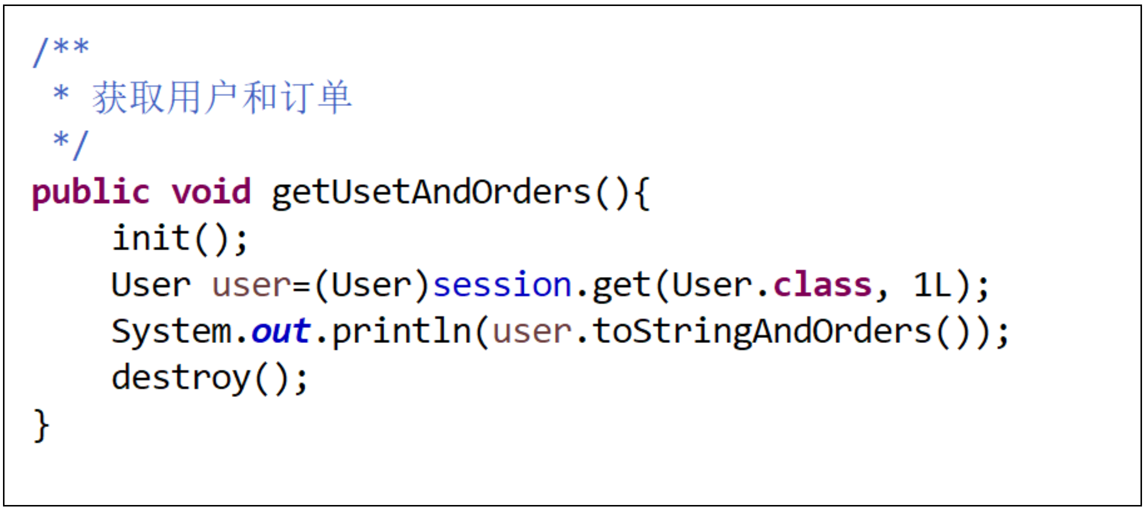


```java
@Entity
@Table(name = "t_order") // N
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String code;

    @Temporal(TemporalType.DATE)
    private Date createDate;

    //特殊属性
    //多对一
    @ManyToOne(fetch=FetchType.EAGER,cascade=CascadeType.ALL)
    @JoinColumn(name="user_id")
    private User user;
```

Test.java

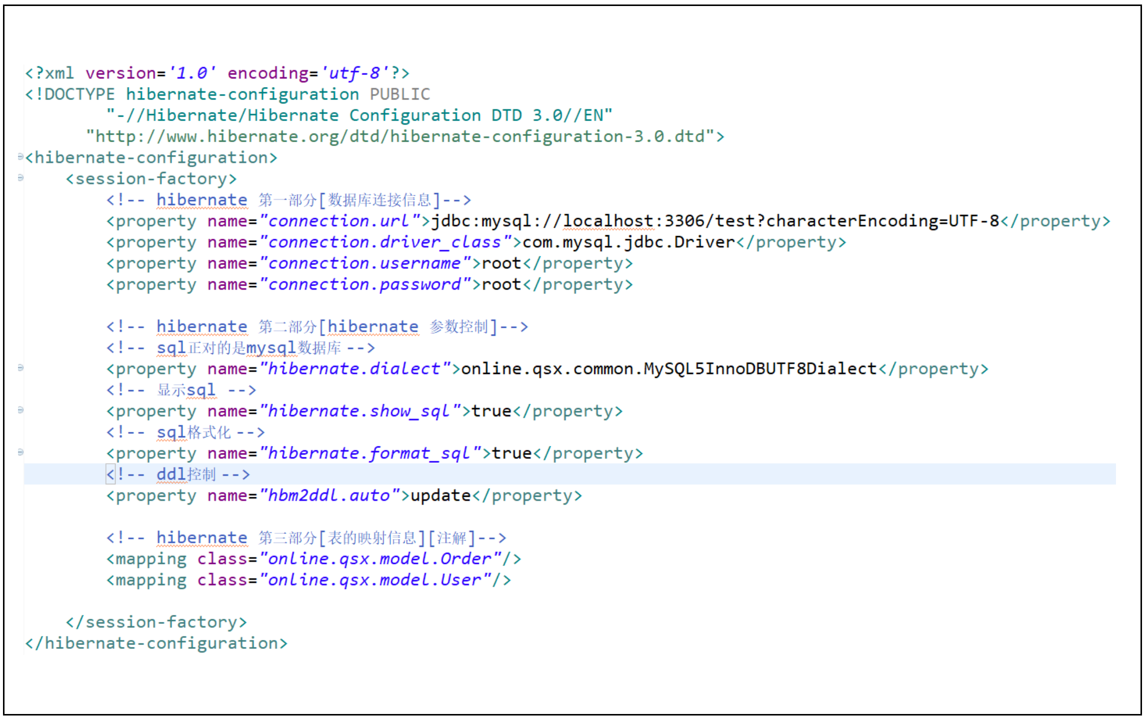


```java
/**
 * 级联添加 用户和订单列表
 */
public void saveUserAndOrders(){
    init();
    User user=new User("arvin","123456",new Date());
    user.getOrders().add(new Order("xxx10001",new Date()));
    user.getOrders().add(new Order("xxx10005",new Date()));
    user.getOrders().add(new Order("xxx10006",new Date()));
    user.getOrders().add(new Order("xxx10007",new Date()));
    user.getOrders().add(new Order("xxx10008",new Date()));


    session.save(user);
    destroy();
}


/**
 * 级联添加 订单和用户
 */
public void saveOrderAndUser(){
    init();

    Order order=new Order("xxxx200001",new Date());
    order.setUser(new User("jack", "123456",new Date()));

    session.save(order);

    destroy();
}


/**
 * 获取订单和用户
 */
public void getOrderAndUser(){
    init();
    Order order=(Order)session.get(Order.class, 6L);
    destroy();
    System.out.println(order.toStringAndUser());
}
```



```
/**
 * 获取用户和订单
 */
public void getUsetAndOrders(){
    init();
    User user=(User)session.get(User.class, 1L);
    System.out.println(user.toStringAndOrders());
    destroy();
}
```

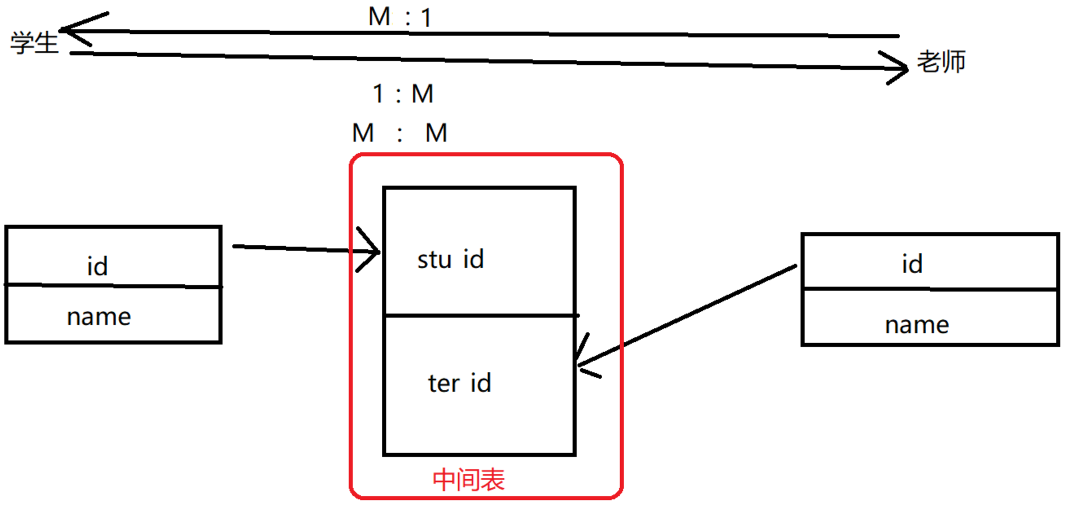
Hibernate.cfg.xml



```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property name="connection.url">jdbc:mysql://localhost:3306/test?characterEncoding=UTF-8</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <!-- sql正对的是mysql数据库 -->
        <property name="hibernate.dialect">online.qsx.common.MySQL5InnoDBUTF8Dialect</property>
        <!-- 显示sql -->
        <property name="hibernate.show_sql">true</property>
        <!-- sql格式化 -->
        <property name="hibernate.format_sql">true</property>
        <!-- ddl控制 -->
        <property name="hbm2ddl.auto">update</property>

        <!-- hibernate 第三部分[表的映射信息][注解]-->
        <mapping class="online.qsx.model.Order"/>
        <mapping class="online.qsx.model.User"/>

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

多对多关系映射

图解



代码演示

Student.java

```
package online.qsx.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "tb_student")
public class Student {
```

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

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

// 特殊属性
@ManyToMany(cascade=CascadeType.ALL,fetch=FetchType.LAZY)
@JoinTable(
        name = "tb_temp", // 中间表的表名
        joinColumns = {
                @JoinColumn(name = "stu_id")   //关联列
        },
        inverseJoinColumns = {
                @JoinColumn(name = "tea_id")   //其他列
        }
)
private Set<Teacher> teachers = new HashSet<Teacher>();

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 Set<Teacher> getTeachers() {
    return teachers;
}

public void setTeachers(Set<Teacher> teachers) {
    this.teachers = teachers;
}
```

```java

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + "]";
    }

    public String toStringAndTeachers() {
        return "Student [id=" + id + ", name=" + name + ",teachers="+teachers+"]";
    }

    public Student(String name) {
        super();
        this.name = name;
    }

    public Student(Long id) {
        super();
        this.id = id;
    }

    public Student() {
        super();
    }

}
```

Teacher.java

```java
package online.qsx.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
```

```
@Entity
@Table(name = "tb_teacher")
public class Teacher {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "tea_id")
    private Long id;

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

    // 特殊属性
    @ManyToMany(cascade=CascadeType.ALL,fetch=FetchType.EAGER)
    @JoinTable(
            name = "tb_temp", //  中间表的表名
            joinColumns = {
                    @JoinColumn(name = "tea_id")   //关联列
            },
            inverseJoinColumns = {
                    @JoinColumn(name = "stu_id")   //其他列
            }
    )
    private Set<Student> students = new HashSet<Student>();

    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 Set<Student> getStudents() {
        return students;
    }
    public void setStudents(Set<Student> students) {
        this.students = students;
    }

    @Override
    public String toString() {
        return "Teacher [id=" + id + ", name=" + name + "]";
    }



    public String toStringAndStudents() {
        return "Teacher [id=" + id + ", name=" + name + ",students="+students+"]";
    }



    public Teacher(String name) {
        super();
        this.name = name;
    }

    public Teacher(Long id) {
        super();
        this.id = id;
    }

    public Teacher() {
        super();
    }

}
```

Test.java

```
package online.qsx.test;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Student;
import online.qsx.model.Teacher;

public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     * 开启 hibernate 连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr = ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭 hibernate 连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    /**
     * 构建表结构
     */
    public void createTables(){
        init();
        System.out.println("构建表结构");
        destroy();
    }

    /**
     * 添加一个学生同时添加多个老师
     */
    public void saveStudentAndTeachers(){
        init();

        Student student=new Student("张三");
        student.getTeachers().add(new Teacher("aaaaa-张三"));
        student.getTeachers().add(new Teacher("bbbbb-张三"));

        session.save(student);
        destroy();
    }

    /**
     * 添加一个学生同时添加多个老师
     */
    public void saveTeacherAndStudents(){
        init();
        Teacher teacher=new Teacher("李四");
        teacher.getStudents().add(new Student("ccccc-李四"));
        teacher.getStudents().add(new Student("ddddd-李四"));
        session.save(teacher);
        destroy();
    }

    /**
     * 查询一个老师及对应的所有学生
     */
    public void getTeacherAndStudents(){
        init();
        Teacher teacher=(Teacher)session.get(Teacher.class, 1L);
        System.out.println(teacher.toStringAndStudents());
        destroy();
    }

    /**
     * 查询一个学生及对应的所有老师
     */
    public void getStudentAndTeachers(){
```

```
		init();
		Student student=(Student)session.get(Student.class, 1L);
		System.out.println(student.toStringAndTeachers());
		destroy();
	}

	public static void main(String[] args) {
		Test test = new Test();
		test.getStudentAndTeachers();
	}
}
```

Hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property name="connection.url">jdbc:mysql://localhost:3306/test?characterEncoding=UTF-8</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <!-- sql正对的是mysql数据库 -->
        <property name="hibernate.dialect">online.qsx.common.MySQL5InnoDBUTF8Dialect</property>
        <!-- 显示sql -->
        <property name="hibernate.show_sql">true</property>
        <!-- sql格式化 -->
        <property name="hibernate.format_sql">true</property>
        <!-- ddl控制 -->
        <property name="hbm2ddl.auto">update</property>

        <!-- hibernate 第三部分[表的映射信息][注解]-->
        <mapping class="online.qsx.model.Student"/>
        <mapping class="online.qsx.model.Teacher"/>
    </session-factory>
</hibernate-configuration>
```

MySQL5InnoDBUTF8Dialect.java

```
package online.qsx.common;

import org.hibernate.dialect.MySQL5Dialect;

public class MySQL5InnoDBUTF8Dialect extends MySQL5Dialect {

    @Override
    public String getTableTypeString() {
        return "ENGINE=InnoDB CHARSET=utf8";
    }

}
```

一对一关系映射[扩展][自学]

课程作业

平台推送

利用 Hibernate 实现多对多关联映射,使用 Annotation 配置

利用 Hibernate 实现一对一关联映射,使用 Annotation 配置

Day04:深入 Hibernate HQL 查询,DQL 优化

课程目标

List

```
    /**
```

```
 * 获取用户和订单
 */
public void getUsersAndOrders() {
    init();

    // HQL--->SQL
    Query query = session.createQuery("from User");
    List<User> list = query.list();
    for (User temp : list) {
        System.out.println(temp.toStringAndOrders());
    }
    destroy();
}
```

Iterate

```
/**
 * 获取用户和订单
 */
public void getUsersAndOrders() {
    init();

    // HQL--->SQL
    Query query = session.createQuery("from User");

    // 一条一条的查询，缓存中有就读取缓存
    Iterator<User> iterator = query.iterate();
    while (iterator.hasNext()) {
        User user = iterator.next();
        System.out.println(user.toStringAndOrders());
    }

    destroy();
}
```

一级缓存

初始 HQL 语法

基础 HQL

```
Query query  = session.createQuery("from User");
```

条件 HQL

占位符

```
Query query = session.createQuery("from User where id>=? and username
like ? ");
query.setLong(0, 5);
query.setString(1,"%5%");
```

别名

```
Query query = session.createQuery("from User where id>= :id and username
like :username ");
query.setLong("id", 5);
query.setString("username","%5%");
```

深入 HQL 分页

```
Query query = session.createQuery("from User where id>=? ");
query.setLong(0, 5);

// 分页
query.setMaxResults(3); // 每页显示的数据条数
query.setFirstResult((3 - 1) * 3); // 越过的查询数据条数
```

深入 HQL 实现子查询

```
Query query=session.createQuery(
"
    from Order o where o.user.id=(
       select u.id from User u where u.username='arvin'
    )
 ");
List<Order> list=query.list();
```

深入 HQL 链接查询

inner join

```
    public void inner_join(){
        init();
        Query query=session.createQuery("from User u inner join 
u.orders");
        List<Object[]> list=query.list();
        
        for (Object[] objects : list) {
            for (Object object : objects) {
                if(object instanceof User){
                    User user=(User) object;
                    System.out.println(user.toString());
                }else if(object instanceof Order){
                    Order order=(Order) object;
                    System.out.println(order.toString());
                }
            }
        }
        destroy();
    }
```

left join

```
public void left_join(){
        init();
        Query query=session.createQuery("from User u left join 
u.orders");
        List<Object[]> list=query.list();
        for (Object[] objects : list) {
            for (Object object : objects) {
                if(object instanceof User){
                    User user=(User) object;
                    System.out.println(user.toString());
                }else if(object instanceof Order){
                    Order order=(Order) object;
                    System.out.println(order.toString());
                }
            }
        }
        destroy();
    }
```

right join

```
public void right_join(){
        init();
        Query query=session.createQuery("from User u right join 
u.orders");
        List<Object[]> list=query.list();

        for (Object[] objects : list) {
            for (Object object : objects) {
                if(object instanceof User){
                    User user=(User) object;
                    System.out.println(user.toString());
                }else if(object instanceof Order){
                    Order order=(Order) object;
                    System.out.println(order.toString());
                }
            }
        }
        destroy();
    }
```

深入 HQL 迫切链接查询

inner join fetch

```
public void inner_join_fetch(){
        init();
        Query query=session.createQuery("from User u inner join fetch 
u.orders");
        List<User> list=query.list();
        for (User user : list) {
            System.out.println(user.toStringAndOrders());
        }
        destroy();
    }
```

left join fetch

```
public void left_join_fetch(){
        init();
        Query query=session.createQuery("from User u left join fetch 
u.orders");
        List<User> list=query.list();
        for (User user : list) {
            System.out.println(user.toStringAndOrders());
        }
        destroy();
    }
```

right join fetch

```
public void right_join_fetch(){
        init();
        Query query=session.createQuery("from User u right join fetch 
u.orders");
        List<User> list=query.list();
        for (User user : list) {
            System.out.println(user.toStringAndOrders());
        }
        destroy();
```

```
    }
```

命名 HQL

User.java

```
@Entity
@Table(name = "t_user") //1
@NamedQueries(value={
        @NamedQuery(name="findListById",query="from User where id>=?"),
        @NamedQuery(name="findListByIdAndUserName",query="from User
where id>= :id and username like :name")
})
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;

    private String password;

    @Temporal(TemporalType.DATE)
    private Date createDate;
}
```

Test.java

```
    /**
     * 获取用户和订单
     */
    public void getUsersAndOrders() {
        init();

        Query query =session.getNamedQuery("findListByIdAndUserName");
        query.setLong("id", 5);
        query.setString("name", "%5%");

        List<User> list = query.list();
        for (User temp : list) {
            System.out.println(temp.toStringAndOrders());
        }
        
        destroy();
    }
```

课程作业

平台推送

利用 Hibernate HQL 技术实现商品信息查询

利用 Hibernate HQL 技术实现带条件的商品信息查询

利用 Hibernate HQL 技术实现分页商品信息查询

利用 Hibernate HQL 实现子查询

Day05:深入 Hibernate Criteria 查询,DDL 优化,DML 优化

课程目标

Criteria 条件查询

```
public void find() {
        init();
        Criteria criteria = session.createCriteria(User.class);
        //添加条件
        criteria.add(Restrictions.gt("age", Short.valueOf("20")));
        criteria.add(Restrictions.lt("age", Short.valueOf("40")));
        criteria.add(Restrictions.like("name", "%3%"));
        
        List<User> list = criteria.list();
        for (User user : list) {
            System.out.println(user);
        }
        destroy();
    }
```

Criteria 分页查询

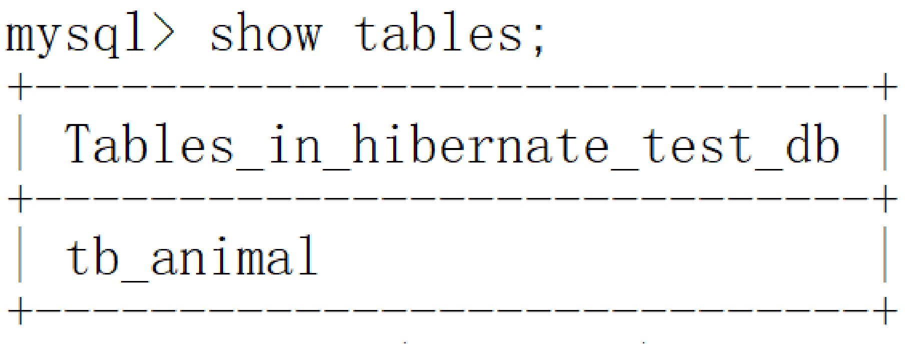
```
public void find() {
        init();
        Criteria criteria = session.createCriteria(User.class);
        //添加条件
        criteria.add(Restrictions.gt("age", Short.valueOf("20")));
        criteria.add(Restrictions.lt("age", Short.valueOf("40")));
        criteria.add(Restrictions.like("name", "%3%"));

        //分页
        criteria.setMaxResults(3);
        criteria.setFirstResult((1-1)*3);

        List<User> list = criteria.list();
        for (User user : list) {
            System.out.println(user);
        }
        destroy();
    }
```

表结构生成策略

InheritanceType.SINGLE_TABLE

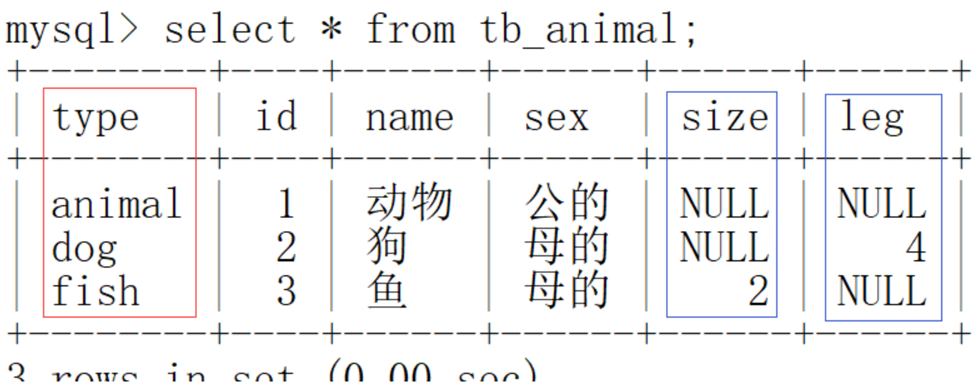
解释：一个表,包含所有信息,依靠关键字区分，父子表[可使用自增]

表结构



```
mysql> show tables;
+-----------------------------+
| Tables_in_hibernate_test_db |
+-----------------------------+
| tb_animal                   |
+-----------------------------+
```

数据结构



```
mysql> select * from tb_animal;
+--------+----+------+------+------+------+
| type   | id | name | sex  | size | leg  |
+--------+----+------+------+------+------+
| animal |  1 | 动物 | 公的 | NULL | NULL |
| dog    |  2 | 狗   | 母的 | NULL |    4 |
| fish   |  3 | 鱼   | 母的 |    2 | NULL |
+--------+----+------+------+------+------+
3 rows in set (0.00 sec)
```

代码

Animal.java

```
@Entity
@Table(name = "tb_animal")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE) //一个表
@DiscriminatorColumn(name = "type", discriminatorType =
DiscriminatorType.STRING) //区分各表的关键字，列
@DiscriminatorValue(value = "animal")//当前表的关键字
public class Animal {

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

Dog.java

```
@Entity
@DiscriminatorValue(value = "dog")//当前表的关键字
public class Dog extends Animal {
    private Long leg;
}
```

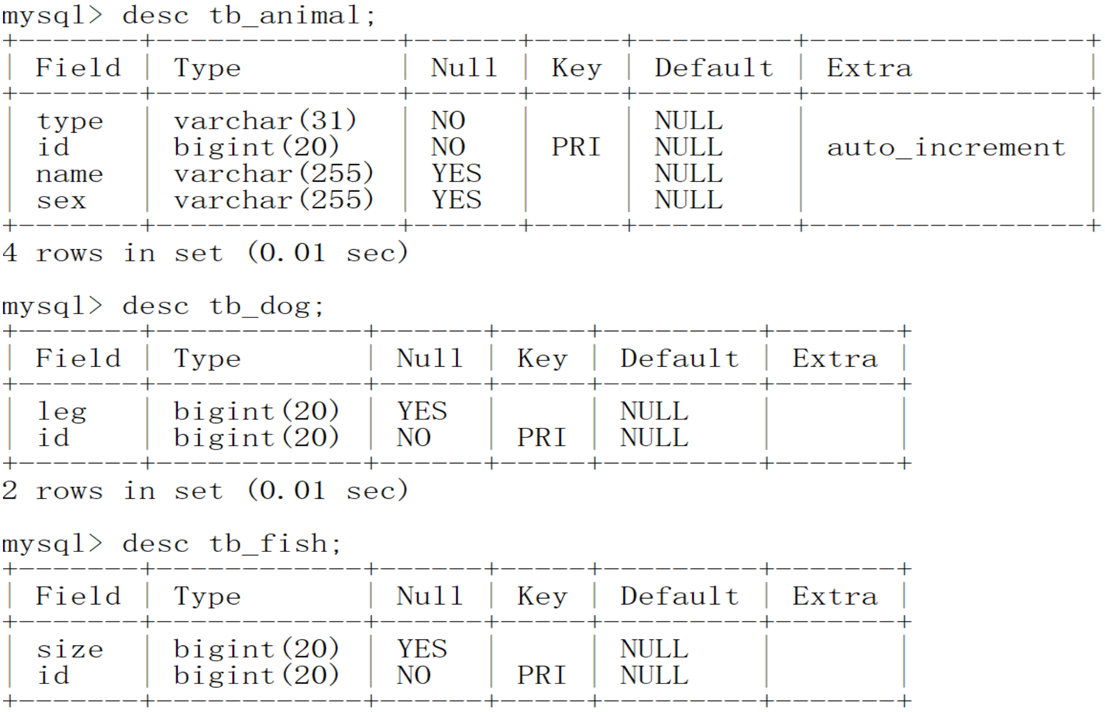
Fish.java

```
@Entity
@DiscriminatorValue(value = "fish")//当前表的关键字
public class Fish extends Animal {
    private Long size;
}
```

InheritanceType.JOINED

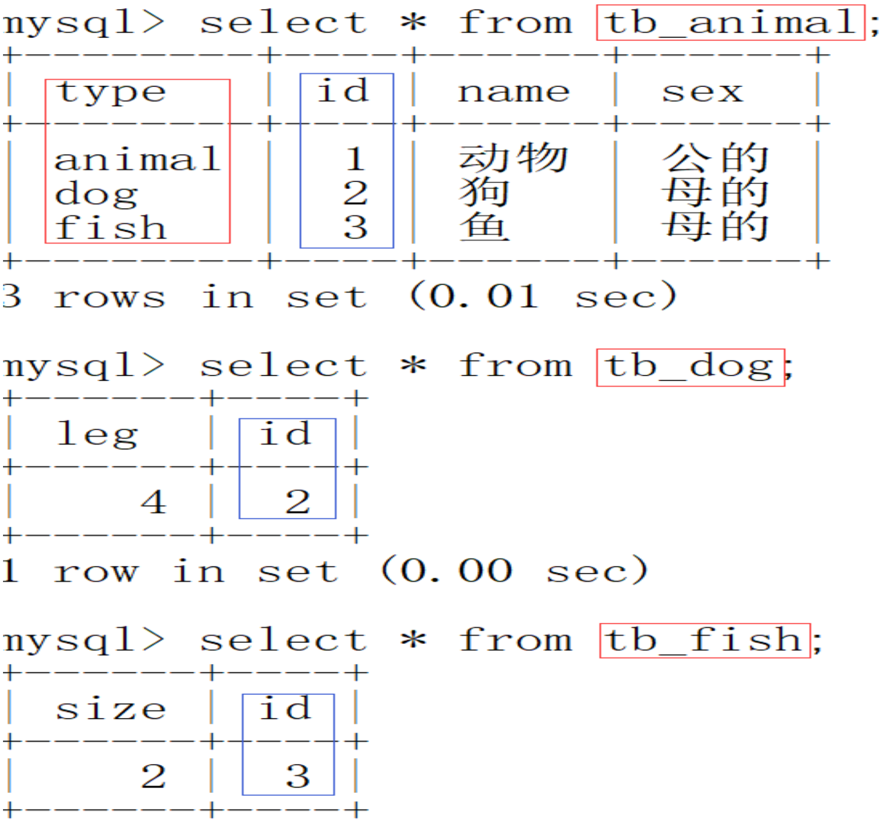
解释：父类,之类均有表结构,子表存放特有属性,有一个外键和父表关联[可使用自增]

表结构



```
mysql> desc tb_animal;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| type  | varchar(31)  | NO   |     | NULL    |                |
| id    | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | YES  |     | NULL    |                |
| sex   | varchar(255) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)

mysql> desc tb_dog;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| leg   | bigint(20) | YES  |     | NULL    |       |
| id    | bigint(20) | NO   | PRI | NULL    |       |
+-------+------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> desc tb_fish;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| size  | bigint(20) | YES  |     | NULL    |       |
| id    | bigint(20) | NO   | PRI | NULL    |       |
+-------+------------+------+-----+---------+-------+
```

数据结构



```
nysql> select * from tb_animal;
+--------+----+------+------+
| type   | id | name | sex  |
+--------+----+------+------+
| animal |  1 | 动物 | 公的 |
| dog    |  2 | 狗   | 母的 |
| fish   |  3 | 鱼   | 母的 |
+--------+----+------+------+
3 rows in set (0.01 sec)

nysql> select * from tb_dog;
+------+----+
| leg  | id |
+------+----+
|    4 |  2 |
+------+----+
1 row in set (0.00 sec)

nysql> select * from tb_fish;
+------+----+
| size | id |
+------+----+
|    2 |  3 |
+------+----+
```

代码

Animal.java

```
@Entity
@Table(name = "tb_animal")
@Inheritance(strategy = InheritanceType.JOINED) //多个表,子表存放特有属性,有一个外键和父表关联
@DiscriminatorColumn(name = "type", discriminatorType = DiscriminatorType.STRING) //区分各表的关键字，列
@DiscriminatorValue(value = "animal")//当前表的关键字
public class Animal {

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

Dog.java

```
@Entity
@Table(name="tb_dog")
@DiscriminatorValue(value = "dog")//当前表的关键字
public class Dog extends Animal {
    private Long leg;
}
```

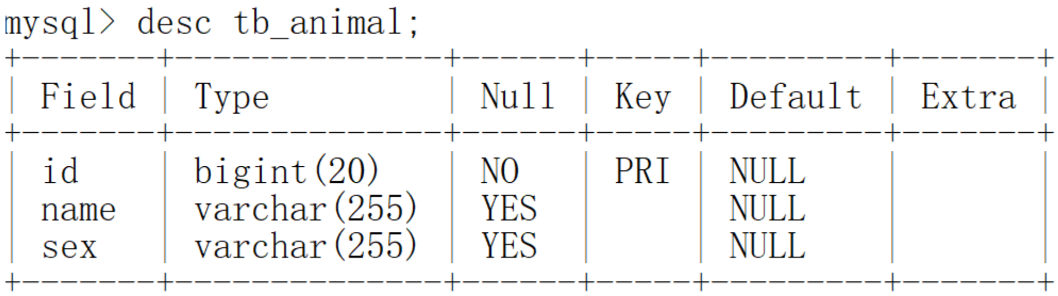
Fish.java

```
@Entity
@Table(name="tb_fish")
@DiscriminatorValue(value = "fish")//当前表的关键字
public class Fish extends Animal {
    private Long size;
}
```

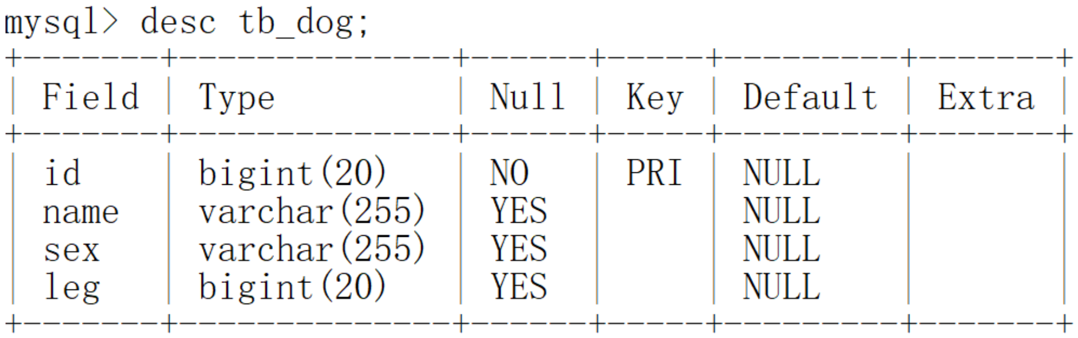
InheritanceType.TABLE_PER_CLASS

解释：父类,子类均有表结构，子表是单独的一个完成表属性被继承到子表中去[不可用自增,且多个表之间 Id 不可重复]

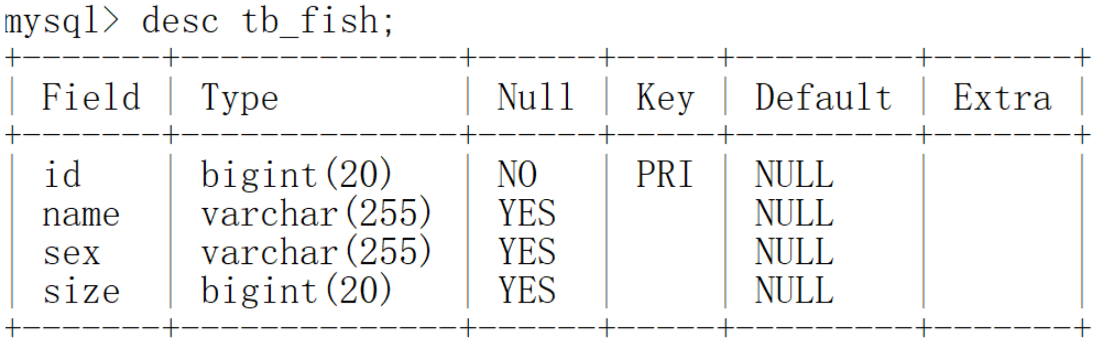
表结构



```
mysql> desc tb_animal;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id     | bigint(20)   | NO   | PRI | NULL    |       |
| name   | varchar(255) | YES  |     | NULL    |       |
| sex    | varchar(255) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
```

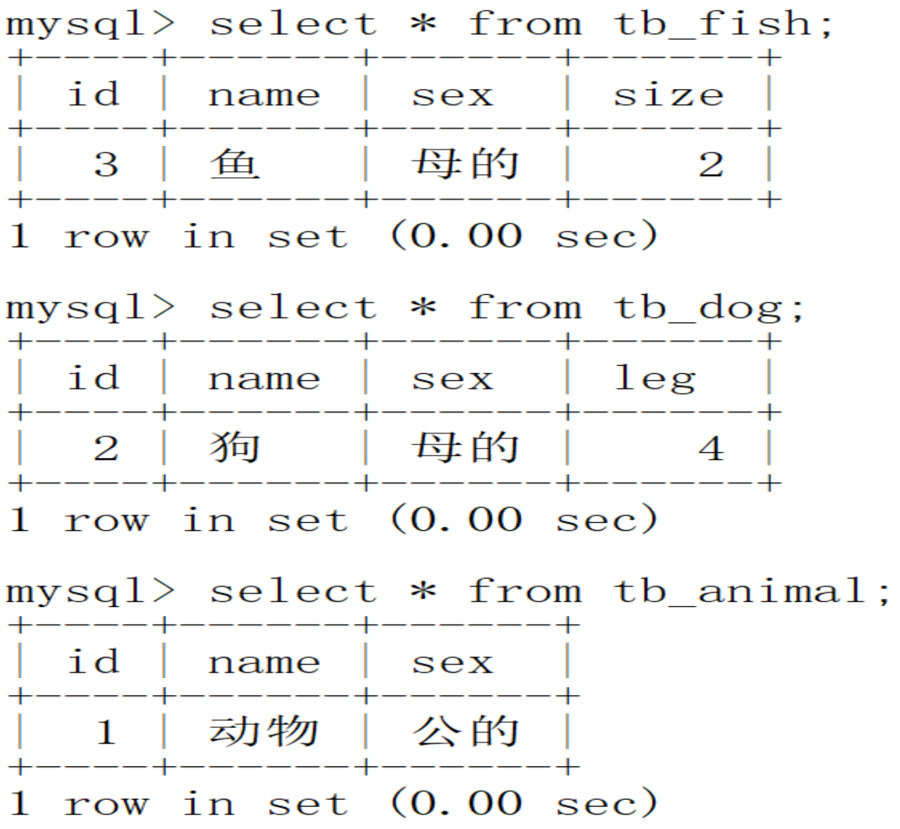


```
mysql> desc tb_dog;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id     | bigint(20)   | NO   | PRI | NULL    |       |
| name   | varchar(255) | YES  |     | NULL    |       |
| sex    | varchar(255) | YES  |     | NULL    |       |
| leg    | bigint(20)   | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
```



```
mysql> desc tb_fish;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id     | bigint(20)   | NO   | PRI | NULL    |       |
| name   | varchar(255) | YES  |     | NULL    |       |
| sex    | varchar(255) | YES  |     | NULL    |       |
| size   | bigint(20)   | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
```

数据结构



```
mysql> select * from tb_fish;
+----+--------+--------+--------+
| id | name   | sex    | size   |
+----+--------+--------+--------+
|  3 | 鱼     | 母的   |      2 |
+----+--------+--------+--------+
1 row in set (0.00 sec)

mysql> select * from tb_dog;
+----+--------+--------+--------+
| id | name   | sex    | leg    |
+----+--------+--------+--------+
|  2 | 狗     | 母的   |      4 |
+----+--------+--------+--------+
1 row in set (0.00 sec)

mysql> select * from tb_animal;
+----+--------+--------+
| id | name   | sex    |
+----+--------+--------+
|  1 | 动物   | 公的   |
+----+--------+--------+
1 row in set (0.00 sec)
```

代码

Animal.java

```java
@Entity
@Table(name = "tb_animal")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS) // 指定继承映射的策略
public class Animal {
    @Id
    private Long id;
    private String name;
    private String sex;
}
```

Dog.java

```java
@Entity
@Table(name = "tb_dog")
public class Dog extends Animal {
    private Long leg;
}
```

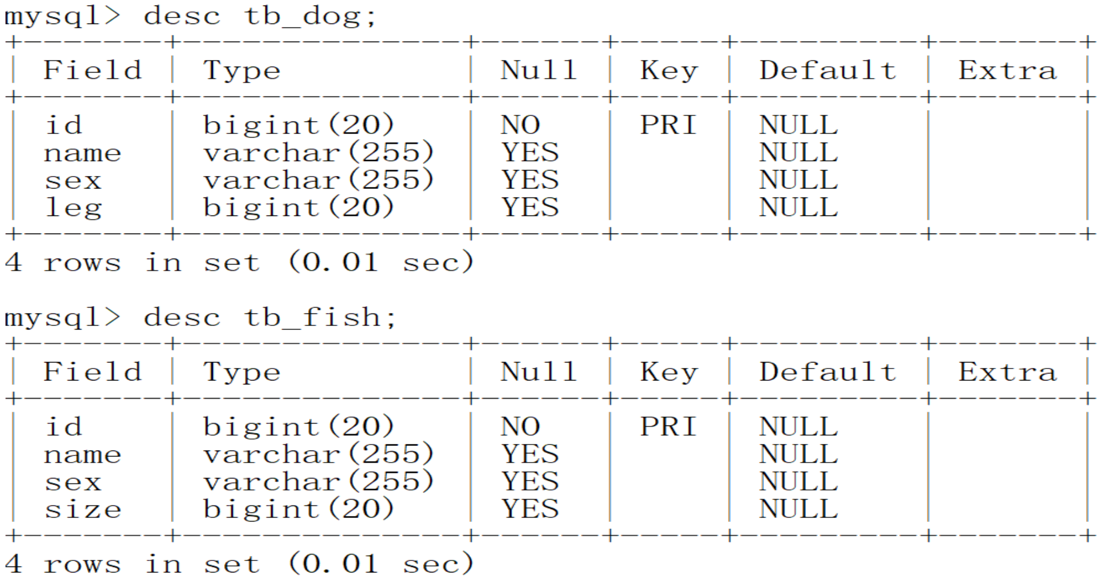
Fish.java

```
@Entity
@Table(name = "tb_fish")
public class Fish extends Animal {
    private Long size;
}
```

MappedSuperclass

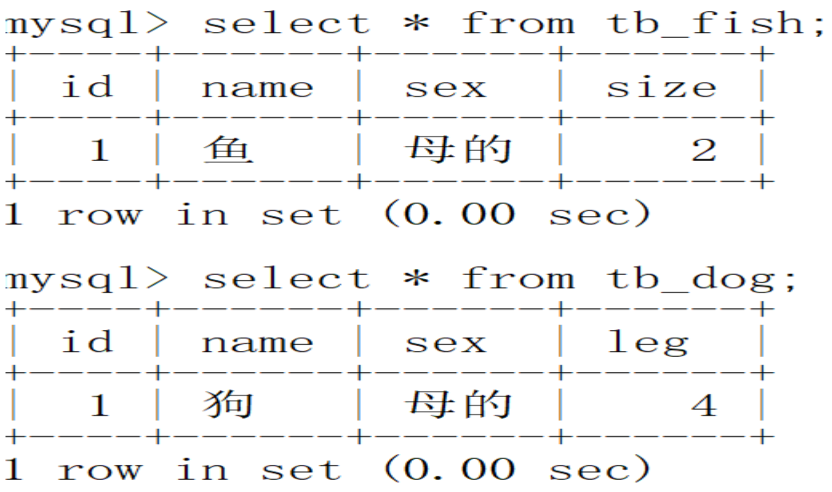
解释：父类直接抽象掉,不会构建出表结构,仅存在完整的子表[可使用自增]

表结构



```
mysql> desc tb_dog;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | bigint(20)   | NO   | PRI | NULL    |       |
| name  | varchar(255) | YES  |     | NULL    |       |
| sex   | varchar(255) | YES  |     | NULL    |       |
| leg   | bigint(20)   | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> desc tb_fish;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| id    | bigint(20)   | NO   | PRI | NULL    |       |
| name  | varchar(255) | YES  |     | NULL    |       |
| sex   | varchar(255) | YES  |     | NULL    |       |
| size  | bigint(20)   | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

数据结构



```
mysql> select * from tb_fish;
+----+--------+--------+--------+
| id | name   | sex    | size   |
+----+--------+--------+--------+
|  1 | 鱼     | 母的   |      2 |
+----+--------+--------+--------+
1 row in set (0.00 sec)

mysql> select * from tb_dog;
+----+--------+--------+--------+
| id | name   | sex    | leg    |
+----+--------+--------+--------+
|  1 | 狗     | 母的   |      4 |
+----+--------+--------+--------+
1 row in set (0.00 sec)
```

代码

Anmal.java

```
@MappedSuperclass // 该实体不会被构建成表
public abstract class Animal {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String sex;
}
```

Dog.java

```
@Entity
@Table(name = "tb_dog")
public class Dog extends Animal {
    private Long leg;
}
```

Fish.java

```
@Entity
@Table(name = "tb_dog")
public class Dog extends Animal {
    private Long leg;
}
```

hibernate 实现动态 SQL

```
@Entity
@Table(name = "tb_commodity")
@SelectBeforeUpdate(value=true) // 取值为false DynamicUpdate将无效果
@DynamicUpdate // 动态SQl修改,必须先查询,在修改，修改语句才能动态化
@DynamicInsert // 动态SQl添加
public class Commodity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY) // 主键自增
    private Long id;

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

    @Temporal(TemporalType.DATE)
    private Date createDate;
}
```

课程作业

非平台推送：

Day04 HQL 相关练习全部使用 Criteria 实现

完成上课案例,实现四种表结构的生成

Day06:回顾 Hibernate 历史进程之 XML 解决方案

课程目标

使用 XML 实现单表映射

User.java

```
package online.qsx.model;

import java.util.Date;


public class User {

    private Long id;

    private String name;

    private Short sex;
```

```java
    private Short age;

    private String phone;

    private Date birthdayDate;

    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 Short getSex() {
        return sex;
    }

    public void setSex(Short sex) {
        this.sex = sex;
    }

    public Short getAge() {
        return age;
    }

    public void setAge(Short age) {
        this.age = age;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public Date getBirthdayDate() {
        return birthdayDate;
    }

    public void setBirthdayDate(Date birthdayDate) {
        this.birthdayDate = birthdayDate;
    }

    public User(String name, Short sex, Short age, String phone, Date
birthdayDate) {
        super();
        this.name = name;
        this.sex = sex;
        this.age = age;
        this.phone = phone;
        this.birthdayDate = birthdayDate;
    }

    public User(Long id, String name, Short sex, Short age, String phone,
Date birthdayDate) {
        super();
        this.id = id;
        this.name = name;
        this.sex = sex;
        this.age = age;
        this.phone = phone;
        this.birthdayDate = birthdayDate;
    }

    public User(Long id) {
        super();
        this.id = id;
    }

    public User() {
        super();
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + ", sex=" + sex + ",
age=" + age + ", phone=" + phone
              + ", birthdayDate=" + birthdayDate + "]";
    }

}
```

User.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.User" table="tb_user">
        <id name="id" column="user_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="user_name" unique="true" length="20" />
        </property>
        <property name="sex" type="java.lang.Short">
            <column name="user_sex" />
        </property>
        <property name="age" type="java.lang.Short">
            <column name="user_age" />
        </property>
        <property name="phone" type="java.lang.String">
            <column name="user_phone" />
        </property>
        <property name="birthdayDate" type="java.util.Date">
            <column name="user_birthdayDate" />
        </property>
    </class>
</hibernate-mapping>
```

Hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
```

```
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息]-->
        <property
name="connection.url">jdbc:mysql://localhost:3306/test?characterEnco
ding=UTF-8</property>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制]-->
        <property
name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</
property><!-- sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示
sql -->
        <property name="hibernate.format_sql">true</property><!-- sql
格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][XML]-->
        <mapping resource="online/qsx/model/User.hbm.xml"/>

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

使用 XML 实现单向一对多

User.java

```
package online.qsx.model;

import java.util.Date;
import java.util.HashSet;
import java.util.Set;

public class User {
```

```
private Long id;

private String name;

private Short sex;

private Short age;

private String phone;

private Date birthdayDate;

//一对多
private Set<Order> orders = new HashSet<Order>();

public Set<Order> getOrders() {
	return orders;
}

public void setOrders(Set<Order> orders) {
	this.orders = orders;
}

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 Short getSex() {
	return sex;
}
```

```
    public void setSex(Short sex) {
        this.sex = sex;
    }

    public Short getAge() {
        return age;
    }

    public void setAge(Short age) {
        this.age = age;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public Date getBirthdayDate() {
        return birthdayDate;
    }

    public void setBirthdayDate(Date birthdayDate) {
        this.birthdayDate = birthdayDate;
    }

    public User(String name, Short sex, Short age, String phone, Date birthdayDate) {
        super();
        this.name = name;
        this.sex = sex;
        this.age = age;
        this.phone = phone;
        this.birthdayDate = birthdayDate;
    }

    public User(Long id, String name, Short sex, Short age, String phone, Date birthdayDate) {
        super();
        this.id = id;
        this.name = name;
        this.sex = sex;
        this.age = age;
        this.phone = phone;
```

```
		this.birthdayDate = birthdayDate;
	}

	public User(Long id) {
		super();
		this.id = id;
	}

	public User() {
		super();
	}

	@Override
	public String toString() {
		return "User [id=" + id + ", name=" + name + ", sex=" + sex + ", age=" + age + ", phone=" + phone
				+ ", birthdayDate=" + birthdayDate + "]";
	}

}
```

User.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.User" table="tb_user">
        <id name="id" column="user_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="user_name" unique="true" length="20" />
        </property>
        <property name="sex" type="java.lang.Short">
            <column name="user_sex" />
        </property>
        <property name="age" type="java.lang.Short">
            <column name="user_age" />
        </property>
        <property name="phone" type="java.lang.String">
```

```
            <column name="user_phone" />
        </property>
        <property name="birthdayDate" type="java.util.Date">
            <column name="user_birthdayDate" />
        </property>

        <!-- 一对多 -->
        <set name="orders" cascade="all" lazy="true" >
            <key column="user_id" />
            <one-to-many class="online.qsx.model.Order"/>
        </set>
    </class>
</hibernate-mapping>
```

Order.java

```
package online.qsx.model;

import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

public class Order {

    private Long id;

    private String code;

    private Date createDate;

    public Long getId() {
        return id;
    }

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

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public Date getCreateDate() {
        return createDate;
    }

    public void setCreateDate(Date createDate) {
        this.createDate = createDate;
    }

    public Order(Long id, String code, Date createDate) {
        super();
        this.id = id;
        this.code = code;
        this.createDate = createDate;
    }

    public Order(String code, Date createDate) {
        super();
        this.code = code;
        this.createDate = createDate;
    }

    public Order(Long id) {
        super();
        this.id = id;
    }

    public Order() {
        super();
    }
    @Override
    public String toString() {
        return "Order [id=" + id + ", code=" + code + ", createDate=" +
createDate + "]";
    }

}
```

Order.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.Order" table="tb_order">
        <id name="id" column="order_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="code" type="java.lang.String">
            <column name="order_code" />
        </property>
        <property name="createDate" type="java.util.Date">
            <column name="order_createDate" />
        </property>
    </class>
</hibernate-mapping>
```

hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息] -->
        <property
```

```
name="connection.url">jdbc:mysql://localhost:3306/test?characterEnco
ding=UTF-8</property>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制] -->
        <property
name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</
property><!--
            sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示
sql -->
        <property name="hibernate.format_sql">true</property><!-- sql
格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][XML] -->
        <mapping resource="online/qsx/model/User.hbm.xml" />
        <mapping resource="online/qsx/model/Order.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

Test.java

```
package online.qsx.test;

import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Order;
import online.qsx.model.User;

public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     * 开启 hibernate 连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr = ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭 hibernate 连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    public void createTable() {
        init();
        System.out.println("构建表结构");
        destroy();
    }

    public void save() {
        init();
        User user=new User("arvin", Short.valueOf("1"), Short.valueOf("1"), "123486789", new Date());
        user.getOrders().add(new Order("arvin1001", new Date()));
        user.getOrders().add(new Order("arvin1002", new Date()));
        user.getOrders().add(new Order("arvin1003", new Date()));
        user.getOrders().add(new Order("arvin1004", new Date()));
        user.getOrders().add(new Order("arvin1005", new Date()));

        session.save(user);

        destroy();
    }



    public void getUser(){
        init();
        session.get(User.class, 2L);
        destroy();
    }
    public static void main(String[] args) {
        Test test = new Test();
        test.save();
    }
}
```

使用 XML 实现单向多对一

User.java

```
package online.qsx.model;

import java.util.Date;
import java.util.HashSet;
import java.util.Set;

public class User {

    private Long id;

    private String name;

private Short sex;

private Short age;

private String phone;

private Date birthdayDate;

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 Short getSex() {
    return sex;
}

public void setSex(Short sex) {
    this.sex = sex;
}

public Short getAge() {
    return age;
}

public void setAge(Short age) {
    this.age = age;
}

public String getPhone() {
    return phone;
}
```

```
    public void setPhone(String phone) {
        this.phone = phone;
    }

    public Date getBirthdayDate() {
        return birthdayDate;
    }

    public void setBirthdayDate(Date birthdayDate) {
        this.birthdayDate = birthdayDate;
    }

    public User(String name, Short sex, Short age, String phone, Date birthdayDate) {
        super();
        this.name = name;
        this.sex = sex;
        this.age = age;
        this.phone = phone;
        this.birthdayDate = birthdayDate;
    }

    public User(Long id, String name, Short sex, Short age, String phone, Date birthdayDate) {
        super();
        this.id = id;
        this.name = name;
        this.sex = sex;
        this.age = age;
        this.phone = phone;
        this.birthdayDate = birthdayDate;
    }

    public User(Long id) {
        super();
        this.id = id;
    }

    public User() {
        super();
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", name=" + name + ", sex=" + sex + ", age=" + age + ", phone="
+ phone
```

```
                        + ", birthdayDate=" + birthdayDate + "]";
    }

}
```

User.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.User" table="tb_user">
        <id name="id" column="user_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="user_name" unique="true" length="20" />
        </property>
        <property name="sex" type="java.lang.Short">
            <column name="user_sex" />
        </property>
        <property name="age" type="java.lang.Short">
            <column name="user_age" />
        </property>
        <property name="phone" type="java.lang.String">
            <column name="user_phone" />
        </property>
        <property name="birthdayDate" type="java.util.Date">
            <column name="user_birthdayDate" />
        </property>

    </class>
</hibernate-mapping>
```

Order.java

```
package online.qsx.model;
```

```java
import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

public class Order {

    private Long id;

    private String code;

    private Date createDate;

    //多对一
    private User user;

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Long getId() {
        return id;
    }

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

    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }

    public Date getCreateDate() {
        return createDate;
    }

    public void setCreateDate(Date createDate) {
        this.createDate = createDate;
    }

    public Order(Long id, String code, Date createDate) {
        super();
        this.id = id;
        this.code = code;
        this.createDate = createDate;
    }

    public Order(String code, Date createDate) {
        super();
        this.code = code;
        this.createDate = createDate;
    }

    public Order(Long id) {
        super();
        this.id = id;
    }

    public Order() {
        super();
    }

    @Override
    public String toString() {
        return "Order [id=" + id + ", code=" + code + ", createDate=" +
createDate + "]";
    }

}
```

Order.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.Order" table="tb_order">
        <id name="id" column="order_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="code" type="java.lang.String">
            <column name="order_code" />
        </property>
        <property name="createDate" type="java.util.Date">
            <column name="order_createDate" />
        </property>

        <!-- 多对一 -->
        <many-to-one cascade="all" lazy="false"  name="user"
class="online.qsx.model.User" column="user_id"/>
    </class>
</hibernate-mapping>
```

hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
       "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
     "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息] -->
        <property
name="connection.url">jdbc:mysql://localhost:3306/test?characterEnco
ding=UTF-8</property>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制] -->
        <property
name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</
property><!--
            sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示
sql -->
        <property name="hibernate.format_sql">true</property><!-- sql
格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][XML] -->
        <mapping resource="online/qsx/model/User.hbm.xml" />
        <mapping resource="online/qsx/model/Order.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

Test.java

```
package online.qsx.test;

import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Order;
import online.qsx.model.User;

public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     *  开启 hibernate 连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr = ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭 hibernate 连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    public void createTable() {
        init();
        System.out.println("构建表结构");
        destroy();
    }

    public void save() {
        init();

        Order order=new Order("jack10001", new Date());
        order.setUser(new  User("jack", Short.valueOf("1"), Short.valueOf("1"), "123486789",
new Date()));
        session.save(order);

        destroy();
    }



    public static void main(String[] args) {
        Test test = new Test();
        test.save();
    }
}
```

使用 XML 实现双向一对多/多对一

User.java

```
package online.qsx.model;

import java.util.Date;
import java.util.HashSet;
import java.util.Set;

public class User {

    private Long id;

    private String name;

    private Short sex;

    private Short age;

    private String phone;

    private Date birthdayDate;

    //一对多
    private Set<Order> orders = new HashSet<Order>();

    public Set<Order> getOrders() {
        return orders;
    }

    public void setOrders(Set<Order> orders) {
        this.orders = orders;
    }

    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 Short getSex() {
        return sex;
    }

    public void setSex(Short sex) {
        this.sex = sex;
    }

    public Short getAge() {
        return age;
    }

    public void setAge(Short age) {
        this.age = age;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public Date getBirthdayDate() {
```

```
		return birthdayDate;
	}

	public void setBirthdayDate(Date birthdayDate) {
		this.birthdayDate = birthdayDate;
	}

	public User(String name, Short sex, Short age, String phone, Date birthdayDate) {
		super();
		this.name = name;
		this.sex = sex;
		this.age = age;
		this.phone = phone;
		this.birthdayDate = birthdayDate;
	}

	public User(Long id, String name, Short sex, Short age, String phone, Date birthdayDate) {
		super();
		this.id = id;
		this.name = name;
		this.sex = sex;
		this.age = age;
		this.phone = phone;
		this.birthdayDate = birthdayDate;
	}

	public User(Long id) {
		super();
		this.id = id;
	}

	public User() {
		super();
	}

	@Override
	public String toString() {
		return "User [id=" + id + ", name=" + name + ", sex=" + sex + ", age=" + age + ", phone="
+ phone
				+ ", birthdayDate=" + birthdayDate + "]";
	}

}
```

User.hbm.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.User" table="tb_user">
        <id name="id" column="user_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="name" type="java.lang.String">
            <column name="user_name" unique="true" length="20" />
        </property>
        <property name="sex" type="java.lang.Short">
            <column name="user_sex" />
        </property>
        <property name="age" type="java.lang.Short">
            <column name="user_age" />
        </property>
        <property name="phone" type="java.lang.String">
            <column name="user_phone" />
        </property>
        <property name="birthdayDate" type="java.util.Date">
            <column name="user_birthdayDate" />
        </property>

        <!-- 一对多 -->
        <set name="orders" cascade="all" lazy="true" inverse="true" >
            <key column="user_id" />
            <one-to-many class="online.qsx.model.Order"/>
        </set>
    </class>
</hibernate-mapping>
```

Order.java

```java
package online.qsx.model;

import java.util.Date;
```

```
public class Order {

    private Long id;

    private String code;

    private Date createDate;

    //多对一
    private User user;

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Long getId() {
        return id;
    }

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

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public Date getCreateDate() {
        return createDate;
    }

    public void setCreateDate(Date createDate) {
        this.createDate = createDate;
    }
    public Order(Long id, String code, Date createDate) {
        super();
        this.id = id;
        this.code = code;
        this.createDate = createDate;
    }

    public Order(String code, Date createDate) {
        super();
        this.code = code;
        this.createDate = createDate;
    }

    public Order(Long id) {
        super();
        this.id = id;
    }

    public Order() {
        super();
    }

    @Override
    public String toString() {
        return "Order [id=" + id + ", code=" + code + ", createDate=" +
createDate + "]";
    }

}
```

Order.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.Order" table="tb_order">
        <id name="id" column="order_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="code" type="java.lang.String">
            <column name="order_code" />
        </property>
        <property name="createDate" type="java.util.Date">
            <column name="order_createDate" />
        </property>

        <!-- 多对一 -->
        <many-to-one cascade="all" lazy="false"  name="user"
class="online.qsx.model.User" column="user_id"/>
    </class>
</hibernate-mapping>
```

hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息] -->
        <property
name="connection.url">jdbc:mysql://localhost:3306/test?characterEnco
ding=UTF-8</property>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制] -->
        <property
name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</
property><!--
            sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示
sql -->
        <property name="hibernate.format_sql">true</property><!-- sql
格式化 -->
        <property name="hbm2ddl.auto">create</property><!-- ddl控制 -->
        <!-- hibernate 第三部分[表的映射信息][XML] -->
        <mapping resource="online/qsx/model/User.hbm.xml" />
        <mapping resource="online/qsx/model/Order.hbm.xml" />
    </session-factory>
</hibernate-configuration>
```

Test.java

```
package online.qsx.test;

import java.util.Date;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Order;
import online.qsx.model.User;

public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     *  开启 hibernate 连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr = ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭 hibernate 连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    public void createTable() {
        init();
        System.out.println("构建表结构");
        destroy();
    }

    public void save() {
        init();

        Order order=new Order("jack10001", new Date());
        order.setUser(new   User("jack", Short.valueOf("1"), Short.valueOf("1"), "123486789",
new Date()));
        session.save(order);


//      User  user=new   User("arvin", Short.valueOf("1"), Short.valueOf("1"), "123486789",
new Date());
//      user.getOrders().add(new Order("arvin1001", new Date()));
//      user.getOrders().add(new Order("arvin1002", new Date()));
//      user.getOrders().add(new Order("arvin1003", new Date()));
//      user.getOrders().add(new Order("arvin1004", new Date()));
//      user.getOrders().add(new Order("arvin1005", new Date()));
//
//      session.save(user);

        destroy();
```

```
	}



	public void getUser(){
		init();
		session.get(User.class, 2L);
		destroy();
	}
	public static void main(String[] args) {
		Test test = new Test();
		test.save();
	}
}
```

使用 XML 实现双向多对多

Order.java

```
package online.qsx.model;

import java.util.Date;
import java.util.HashSet;
import java.util.Set;

/**
 * 订单
 */
public class Order {

	// 普通属性
	private Long id;

	private String code;

	private Date createTime;

	// 特殊属性
	// 一个订单 对应 多个商品
	private Set<Commdity> commdities = new HashSet<Commdity>();

	public Long getId() {
        return id;
    }

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

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public Date getCreateTime() {
        return createTime;
    }

    public void setCreateTime(Date createTime) {
        this.createTime = createTime;
    }

    public Set<Commdity> getCommdities() {
        return commdities;
    }

    public void setCommdities(Set<Commdity> commdities) {
        this.commdities = commdities;
    }

    public Order(Long id, String code, Date createTime) {
        super();
        this.id = id;
        this.code = code;
        this.createTime = createTime;
    }

    public Order(String code, Date createTime) {
        super();
        this.code = code;
        this.createTime = createTime;
    }
    public Order(Long id) {
        super();
        this.id = id;
    }

    public Order() {
        super();
    }

    @Override
    public String toString() {
        return "Order [id=" + id + ", code=" + code + ", createTime=" + createTime + "]";
    }

    public String toStringAndCommditys() {
        return "Order [id=" + id + ", code=" + code + ", createTime=" + createTime +
",commdity="+commdities+"]";
    }
}
```

Order.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.Order" table="tb_order">
        <id name="id" column="order_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="code" column="order_code"
type="java.lang.String"/>
        <property name="createTime" column="order_createTime"
type="java.util.Date"/>

        <!-- 多对多 -->
        <set name="commdities" table="tb_commdity_order"
inverse="false" cascade="save-update" lazy="true">
            <key column="order_id"/>
            <many-to-many class="online.qsx.model.Commdity"
column="commdity_id"/>
```

```
        </set>
    </class>
</hibernate-mapping>
```

Commdity.java

```
package online.qsx.model;

import java.util.HashSet;
import java.util.Set;

/**
 * 商品
 */
public class Commdity {
	// 普通属性
	private Long id;
	private String name;
	private Double money;

	// 特殊属性
	//一个商品对应多个订单
	private Set<Order> orders = new HashSet<Order>();

	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 Double getMoney() {
        return money;
    }

    public void setMoney(Double money) {
        this.money = money;
    }

    public Set<Order> getOrders() {
        return orders;
    }

    public void setOrders(Set<Order> orders) {
        this.orders = orders;
    }

    public Commdity(Long id, String name, Double money) {
        super();
        this.id = id;
        this.name = name;
        this.money = money;
    }

    public Commdity(String name, Double money) {
        super();
        this.name = name;
        this.money = money;
    }

    public Commdity(Long id) {
        super();
        this.id = id;
    }

    public Commdity() {
        super();
    }

    @Override
    public String toString() {
        return "Commdity [id=" + id + ", name=" + name + ", money=" + money + "]";
    }

    public String toStringAndOrders() {
        return "Commdity [id=" + id + ", name=" + name + ", money=" + money + ", orders=" +
orders + "]";
    }

}
```

Commdity.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="online.qsx.model.Commdity" table="tb_commdity">
        <id name="id" column="commdity_id" type="java.lang.Long">
            <generator class="identity" />
        </id>
        <property name="name" column="commdity_name"
type="java.lang.String"/>
        <property name="money" column="commdity_money"
type="java.lang.Double"/>

        <!-- 多对多 -->
        <set name="orders" table="tb_commdity_order">
            <key column="commdity_id"/>
            <many-to-many class="online.qsx.model.Order"
column="order_id"/>
        </set>
    </class>
</hibernate-mapping>
```

hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- hibernate 第一部分[数据库连接信息] -->
        <property
name="connection.url">jdbc:mysql://localhost:3306/test?characterEnco
ding=UTF-8</property>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

        <!-- hibernate 第二部分[hibernate 参数控制] -->
        <property
name="hibernate.dialect">online.qsx.common.MySQL5MyISAMUTF8Dialect</
property><!--
            sql正对的是mysql数据库 -->
        <property name="hibernate.show_sql">true</property><!-- 显示
sql -->
        <property name="hibernate.format_sql">true</property><!-- sql
格式化 -->
        <property name="hbm2ddl.auto">update</property><!-- ddl控制 -->

        <!-- hibernate 第三部分[表的映射信息][XML] -->
        <mapping resource="online/qsx/model/Commdity.hbm.xml"/>
        <mapping resource="online/qsx/model/Order.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Test.java

```
package online.qsx.test;

import java.util.Date;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import online.qsx.model.Commdity;
import online.qsx.model.Order;
```

```
public class Test {
    SessionFactory sf = null;
    Session session = null;
    Transaction transaction = null;

    /**
     * 开启hibernate连接
     */
    public void init() {
        // 加载配置文件
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        // 注册标准服务
        StandardServiceRegistryBuilder ssrb = new
StandardServiceRegistryBuilder();
        StandardServiceRegistry ssr =
ssrb.applySettings(configuration.getProperties()).build();
        // 通过标准服务加载配置文件后获得会话工厂
        sf = configuration.buildSessionFactory(ssr);// 二级缓存
        // 开启一个会话
        session = sf.openSession();// 一级缓存
        // 开启事物
        transaction = session.beginTransaction();
        // 操作
        System.out.println("连接开启成功");
    }

    /**
     * 关闭hibernate连接
     */
    public void destroy() {
        // 提交事物
        transaction.commit();
        // 关闭
        session.close();
        sf.close();
        System.out.println("连接关闭成功");
    }

    public void createTable() {
        init();
        System.out.println("构建表结构");
        destroy();
    }

    public void save(){
        init();
        Order order=new Order("jack1001",new Date());
        order.getCommdities().add(new Commdity("aaaa",5000.0));
        order.getCommdities().add(new Commdity("aa2a",5000.0));
        order.getCommdities().add(new Commdity("aa4a",5000.0));

        session.save(order);
        destroy();
    }

    public void find(){
        init();
        Query query=session.createQuery("from Order where id=1");
        Order order=(Order)query.uniqueResult();

        System.out.println(order.toStringAndCommditys());

        destroy();


    }

    public static void main(String[] args) {
        Test test = new Test();
        test.find();
    }
}
```

使用 XML 实现一对一[平台学学习]

级联关系:cascade

```
<!-- 一对多 -->
<set name="orders" cascade="all" >
    <key column="user_id" />
    <one-to-many class="online.qsx.model.Order"/>
</set>
```

```
<!-- 多对一 -->
<many-to-one
    cascade="all"
    name="user"
    class="online.qsx.model.User"
    column="user_id"
/>
```

懒加载,立即加载:lazy

```
<!-- 一对多 -->
<set name="orders"  lazy="true" >
          <key column="user_id" />
  <one-to-many class="online.qsx.model.Order"/>
</set>

<!-- 多对一 -->
<many-to-one
    lazy="false"
    name="user"
    class="online.qsx.model.User"
    column="user_id"
/>
```

控制反转: inverse

```
<!-- 一对多 -->
<set name="orders" inverse="true" >
    <key column="user_id" />
    <one-to-many class="online.qsx.model.Order"/>
</set>

<!-- 多对一 -->
<many-to-one
    cascade="all"
    name="user"
    class="online.qsx.model.User"
    column="user_id"
/>
```

课程作业

平台推送

利用 Hibernate 实现一对多关联映射,使用 XML 配置

利用 Hibernate 实现多对一关联映射,使用 XML 配置

利用 Hibernate 实现多对多关联映射,使用 XML 配置

利用 Hibernate 实现一对一关联映射,使用 XML 配置

Day07-Day10:Hibernate 实现系统权限管理模块

课程目标

权限表结构, ER 图

核心结构字段

资源字段

角色字段

用户字段

表结构截图

资源表

角色表

用户表

资源 and 角色关系表

角色 and 用户关系表

课程作业

非平台推送

使用 Hibernate 实现系统权限管理模块,(java 控制台项目)

完成功能：

1. 游客

1.1 注册账户

1.2 登陆系统

2.普通用户

2.1**[all]**输入账号密码登陆系统

2.2**[all]**登陆系统后能修改自己信息功能(改账号,改密码,改备注)

2.3**[all]**登陆系统后**能够浏览**自己的**角色信息列表**

2.4**[all]**登陆系统后能够输入**自己拥有的角色名称**浏览到对应的资源信息列表

3.系统管理员

3.1**[all]**输入账号密码登陆系统

3.2**[all]**登陆系统后能修改自己信息功能(改账号,改密码,改备注)

3.3**[all]**登陆系统后**能够浏览**自己的**角色信息列表**

3.4**[all]**登陆系统后能够输入**自己拥有的角色名称**浏览到对应的资源信息列表

3.5**[system]**浏览所有用户对应的角色及对应的资源信息

3.6**[system]**修改

3.6.1 资源信息功能(改名称,改状态,改备注)

3.6.2 角色信息功能(改名称,改状态,改备注)

3.6.3 用户信息功能(改状态)

3.7**[system]**浏览

3.7.1 用户列表

3.7.2 角色列表

3.7.3 资源列表

3.8**[system]**删除

3.8.1 用户信息

3.8.2 角色信息

3.8.3 资源信息

要求：

1.使用 Hibernate JPA,XMl,各实现一套

Day11:初始 JPA 规范

Day12:初始 Spring Data JPA + Hibernate 实现

Day13: Repository 组件优化数据访问

Day14: CrudRepository 组件优化 CRUD

Day15: PagingAndSortingRepository 组件优化分页,排序

Day16: JpaRepository 组件对缓存扩展

Day17:JpaSpecificationExecutor 组件对动态条件的扩展

Day18: Repository 组件优化数据访问自定义 SQL