**中软国际考题-笔试**

1. 单选题
2. 在Hibernate关系映射配置中，inverse属性的值为true含义是（B ）。

A. 定义在<one-to-many>节点上，声明要负责关联的维护

B. 声明在<set>节点上，声明要对方负责关联的维护

C. 定义在<one-to-many>节点上，声明对方要负责关联的维护

D. 声明在<set>节点上，声明要负责关联的维护

2、关于Hibernate，下列说法错误的是（ B  ）。

A．Hibernate是一个对象-关系映射工具

B．在MVC模式中，Hibernate属于控制层

C．Hibernate在JDBC的方式上进行封装，以简化JDBC方式繁琐的编码工作

D．在Hibernate中，可以通过xml格式的配置文件保存对象-关系的映射信息

3、hibernate的缓存机制是哪一个（B）

A.对象缓存机制 B. hibernate一级缓存 C.引用缓存机制 D.顺序缓存机制

4、对于以下程序，Customer对象在第几行变为游离状态？（D）

Customer customer=new Customer(); //line1

customer.setName(\”Tom\”); //line2

Session session1=sessionFactory.openSession(); //line3

Transaction tx1 = session1.beginTransaction();

session1.save(customer); //line4

tx1.commit(); //line5

session1.close(); //line6

1. line1 B. line4 C. line5 D. line6

5、下面关于Hibernate的说法，错误的是（C）。

A.Hibernate是一个“对象-关系映射”的实现

B.Hibernate是一种数据持久化技术

C.Hibernate是JDBC的替代技术

D.使用Hibernate可以简化持久化层的编码

6. 在Hibernate关系映射配置中，inverse属性的含义是（B）。

A.定义在<one-to-many>节点上，声明要负责关联的维护

B.声明在<set>节点上，声明要对方负责关联的维护

C.定义在<one-to-many>节点上，声明对方要负责关联的维护

D.声明在<set>节点上，声明要负责关联的维护

7. 在使用了Hibernate的系统中，要想在删除某个客户数据的同时删除该客户对应的所有订单数据，下面方法可行的是（B）。

A.配置客户和订单关联的cascade属性为save-update

B.配置客户和订单关联的cascade属性为all

C.设置多对一关联的inverse属性为true

D.设置多对一关联的inverse属性为false

8.执行下面的hql查询，关于返回值的说法，正确的是（B）。

select u.username,u.userid from com.x.entity.User u where u is not null

A.语法错误，不能执行 B.返回一个类型为List的实例

C.返回一个数组 D.当返回结果只有一条时，返回一个数组

9. Hibernate的生命周期的三种状态。不包括下列哪一项？(B)

A．Transient b。Contacted c 。Persistent d。Detached

10、关于Hibernate，下列说法错误的是（ B   ）。

A. Hibernate是一个对象-关系映射工具

B. 在MVC模式中，Hibernate属于控制层

C. Hibernate在JDBC的方式上进行封装，以简化JDBC方式繁琐的编码工作

D. 在Hibernate中，可以通过xml格式的配置文件保存对象-关系的映射信息

11、 在Hibernate中，下列哪个选项不属于Session的方法（ C ）。

A. load

B. save

C. open

D. delete

12、 Hibernate是一个优秀的Java持久化层解决方案，那么数据的持久化指的是（ B ）。

A. 数据在程序退出后就不复存在

B. 数据在程序退出后以文件等形式保存在存储设备中

C. 数据的内容是不变的，处于静态中

D. 数据的内容是变化的

13、下面关于Hibernate中load和get方法说法正确的是（D ）

A）这两个方法是一样的，没有任何的区别

B）这两个方法不一样，load先找缓存，再找数据库

C）这两个方法不一样，get先找缓存，再找数据库

D）以上说法都不对

14、在Hibernate中，以下（ B ）不属于session的方法

A、 close（） B. open() C. update() D. delete()

15下面关于Hibernate中load和get方法说法正确的是（D ）

A）这两个方法是一样的，没有任何的区别

B）这两个方法不一样，load先找缓存，再找数据库

C）这两个方法不一样，get先找缓存，再找数据库

D）以上说法都不对

16、使用Hibernate技术实现数据持久化时，下面（ D）内容不在 Hibernate配置文件中配置(选择一项)

A) 数据库连接信息

B) 数据库类型（dialect）

C) show\_sql参数

D) 数据库表和实体的映射信息

17、关于Hibernate中关系的说话正确的是（C ）

A）一对多必须用Set来映射

B）多对一必须用Set来映射

C）一对多可以用Set来映射,也可以用List、Map来映射

D）多对一必须用Set来映射,也可以用List、Map来映射

18.对于以下程序，Customer对象在第几行变为游离状态？（F）

Line-1. Customer customer=new Customer(); //line1

Line-2. customer.setName(\"Tom\"); //line2

Line-3. Session session1=sessionFactory.openSession(); //line3

Line-4. Transaction tx1 = session1.beginTransaction(); //line4

Line-5. session1.save(customer); //line4

Line-6. tx1.commit(); //line5

Line-7.session1.close(); //line6

A) line1 B)line2 C)line3 D)line4 E)line5 F)line6

19、从SessionFactory中得到Session的方法是（B ）

A）getSession

B）openSession

C）currentSession

D）以上都不是

20、下面创建Criteria对象的语句中正确的是（D ）。

a. Criteria c = query.createCriteria();

b. Criteria c = query.addCriteria();

c. Criteria c = session.createCriteria();

d. Criteria c = session.createCriteria(User.class);

21、下列选项属于Hibernate核心技术（B）

A.IOC B.ORM C.DI D.AOP

1. 多选
2. 在Hibernate 中，持久化对象具有以下（A、B、C ）特征

A. 位于一个Session 实例的缓存中

B. 持久化对象和数据库中的相关记录对应。

C. Session 在清理缓存时，会根据持久化对象的属性变化，来同步更新数据库。

D. Session 在调用delete()方法时，会将一个对象置于持久化状态。

2、在Hibernate中，下列说法正确的有（ABC）。

A．Hibernate是一个开放源代码的对象关系映射框架

B．Hibernate对JDBC进行了轻量级的对象封装

C．Hibernate可以大大减少操作数据库的工作量

D．Hibernate提供了数据查询机制，但没有提供恢复机制

3、在Hibernate 的配置文件中，以下选项对session-factory配置说法正确的是（ABC）。

A. 属性connection.url用于指定数据库接的地址

B. 属性connection.driver\_class用于指定数据库驱动程序类

C. 属性show\_sql参数为true，则程序运行时在控制台输出执行的SQL语句

D. 在配置session-facotry 对象时，属性dialect不是必须配置的

4、Hibernate的查询方式有几种？（A、B，C，D）

A. 对象导航查询(objectcomposition)

B. HQL查询

C. Criteria 查询

D. SQLQuery本地SQL查询

5、下面不属于持久化的是（A D）

1. 把程序数据存储到List集合
2. 把程序数据存储到数据库中
3. 把程序数据存储为xml文件；

D．把程序数据存储为Properties类

6、Hibernate配置文件中，不包含下面的（A,B）。

A.“对象-关系映射”信息 B.实体间关联的配置

C.show\_sql等参数的配置 D.数据库连接信息

7、在Hibernate中，下列说法正确的有（ ABC   ）。【选三项】

A. Hibernate是一个开放源代码的对象关系映射框架

B. Hibernate对JDBC进行了轻量级的对象封装

C. Hibernate可以大大减少操作数据库的工作量

D. Hibernate提供了数据查询机制，但没有提供恢复机制

8、下面关于Hibernate说法正确的是（AD ）（选择两项）

A） Hibernate是ORM的一种实现方式

B） Hibernate不要JDBC的支持

C） 属于控制层

D） 属于数据持久层

9、下面关于ORM的说法错误的是（ AC )（选择两项）

A）对象关系映射，让现实中的对象同数据库的表产生映射（类与表产生映射）

B）对象关系映射，让类同表产生关系

C）对象关系映射，让类同记录产生关系 （类的实例与记录（表中的一行数据）产生关系）

D）对象关系映射，让类中的属性同表中的列产生关系

10、下面关于Hibernate中Session的说法正确的是（AD ）（选择两项）

A）Session是轻量级的，可以随意的创建和销毁

B）Session是重量级的，不能随意的创建和销毁

C）Session是线程安全的

D） Session不是线程安全的

11、在Hibernate中修改对象的说话错误的是（AD ）

A）只能利用update方法来做修改

B）可以利用saveOrUpdate方法来做修改

C）可以利用HQL语句来做修改

D）不能利用HQL语句来修改

12、下面关于Hibernate中Transaction的使用说法正确的是（ BD ）(选择两项)

A）Transaction是可有可无的

B）Transaction在做查询的时候是可有可无的

C）Transaction在做修改的时候是可有可无的

D）Transaction在做修改的时候是必须的

13、关于Hibernate中show\_sql配置说法正确的是（BD ）(选择两项)

A）默认值为true

B）默认值为false

C）为true时表示在后台打印出SQL语句

D）为false表示在后台打印出SQL语句

14、在Hibernate映射文件的ID的配置中,主键生成器native的说法正确的是（ ）

A）包括desc、asc和identity

B）包括desc、asc和sequence

C）包括increment、sequence、hilo和identity

D）包括sequence、hilo和identity

15、下列关于延迟加载说法正确的是（ AC ）（选择两项）

A）lazy="true"表示采用延迟加载

B）lazy="false"表示采用延迟加载

C）Lazy默认情况下位true

D）Lazy默认情况下位false

16、下面哪些项属于Hibernate支持的关系映射？（A、B、C、D）

A. 一对一

B. 一对多

C. 多对一

D. 多对多

1. 填空题
2. Hibernate管理的实体对象的三种状态是：\_\_\_\_ 瞬时（临时）\_\_\_、\_\_\_持久\_\_\_\_、\_\_\_\_\_脱管（游离）\_\_\_
3. Hibernate中cascade属性的值有\_\_\_\_\_ none\_\_\_\_、\_\_\_\_ save-update\_\_\_\_\_、\_\_\_\_\_ delete\_\_\_\_、\_\_\_\_ all\_\_\_\_\_
4. 在Hibernate应用中Java对象的状态有\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_。(临时状态、持久化状态、游离状态)
5. Session 是Hibernate持久化操作的基础，提供了众多持久化方法，如save、update、delete等。通过这些方法，透明地完成对象的增加、删除、修改、查找等操作。
6. Hibernate使用 save() 方法保存持久化对象时，该方法返回该持久化对象的\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(标识属性值(即对应记录的主键值)；)

1. Customer类中有一个Set类型的orders属性，用来存放Order订单对象，在Customer.hbm.xml文件中，用 元素映射orders属性？（<set>）
2. 在Hibernate中，元素有一个cascade属性，如果希望Hibernate级联保存集合中的对象，casecade属性应该设值为 ？（save-update）
3. 在三层结构中，数据访问层承担的责任是\_\_\_\_\_\_\_\_\_\_ 。(数据的增删改查操作)
4. 在Hibernate中，用于比较字符串的like操作符“like '部%'”表示：所有以“部”开头的字符串
5. 在持久化层，对象分为哪些状态：\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_。(临时、持久化、游离)
6. 在Hibernate映射文件的ID的配置中,主键生成器native相当于包含了\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_四种策略（increment、sequence、hilo和identity）
7. 在使用了Hibernate的系统中，要想在删除某个客户数据的同时删除该客户对应的所有订单数据，配置客户和订单关联的cascade属性为。\_\_\_\_\_\_\_\_\_\_、(all)
8. Hibernate中对象的三种状态分别是：\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_。(瞬时 、持久、脱管)
9. 判断题
10. Hibernate拥有分布式、安全检查、集群、负载均衡的支撑（T）
11. Hibernate具有可扩展性，API开放，但是当本身功能不够用的时候，不可以自己编码进行扩展（F）（可以继续编码进行扩展）
12. HQL具备更强大的功能，实体的更新与删除即其主要特征之一（ T ）
13. delete与update子句为Hibernate3引入的新特性（ T ）
14. Hibernate是一个“对象-关系映射”的实现 （ T ）
15. Hibernate是一种数据持久化技术（ T ）
16. Hibernate是JDBC的替代技术 （ F ）
17. 使用Hibernate可以简化持久化层的编码（ T ）
18. Hibernate 中，Session对象的创建代价很高，它是线程安全对象，为所有的应用程序线程所共享。（ F ）
19. getCurrentSession是不需要我们手动关闭的，因为工厂会自己管理，而openSession需要我们手动关闭。。（ T ）
20. HQL：from 后面跟的是表名 where 后 用表中字段做条件（F）
21. SessionFactory 接口：负责保存、更新、删除、加载和查询对象，是线程不安全的，避免多个线程共享同一个session，是轻量级、一级缓存。（ F ）
22. HQL通常用于查询数据，SQL可以增删改查数据。（ T ）
23. Hibernate中关系一对多必须用Set来映射。（ T）
24. HQL查询可以执行T-SQL语句，执行速度更快。（ X ）
25. 关于Hibernate中show\_sql默认值为true。（ F ）
26. 一对多可以用Set来映射,也可以用List、Map来映射(T)
27. 在多对一得配置中，关于fetch其值可以是join或select
28. 在多对一得配置中，关于fetch其值可以是lazy或inverse
29. HQL不是SQL，不区分大小写（F）
30. Hibernate可以看做是JDBC 的框架。（ T ）

五、简答题

# 1、Hibernate是如何延迟加载?

答案：当Hibernate在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，就实现了延迟加载，他节省了服务器的内存开销，从而提高了服务器的性能。

2、Hibernate的检索方式（5分）

答案：①导航对象图检索 ②OID检索 ③HQL检索 ④QBC检索 ⑤本地SQL检索

3、 session的清理和清空有什么区别？

答案：session清理缓存是指按照缓存中对象的状态的变化来同步更新数据库；清空是session关闭；

4、hibernate中Session的find()方法以及Query接口的区别

答案：Session类的find()方法以及Query接口都支持HQL检索方式。这两者的区别在于，前者只是执行一些简单HQL查询语句的便捷方法，它不具有动态绑定参数的功能，而且在Hibernate3.2版本中，已经淘汰了find()方法；而Query接口才是真正的HQL查询接口，它提供了以上列出的各种查询功能。

5、hibernate中session的清理和清空有什么区别？

答案：

（1）Session可以按照缓存中对象的变化来执行数据库相关的SQL语句，来同步数据库，这个过程就叫做Session清理缓存。

（2）清空是session清除部分或全部对象或者session关闭；当调用session.evict(customer); 或者session.clear(); 或者session.close()方法时，Session的缓存被清空。

6、请简述hibernate中Session的特点有哪些？

答：

（1）不是线程安全的，因此在设计软件架构时，应该避免多个线程共享同一个Session实例。

（2）Session实例是轻量级的，所谓轻量级是指它的创建和销毁不需要消耗太多的资源。这意味着在程序中可以经常创建或销毁Session对象，例如为每个客户请求分配单独的Session实例，或者为每个工作单元分配单独的Session实例。

（3）在Session 中，每个数据库操作都是在一个事务(transaction)中进行的，这样就可以隔离开不同的操作（甚至包括只读操作）。

7、叙述hibernate中Session的缓存的作用

答：

（1）第一减少访问数据库的频率。因为应用程序从缓存中读取数据比直接从数据库中读取数据速度要快的多。所以Session的缓存可以起到提高数据访问性能的作用。

（2）第二保证缓存中的相关对象与数据库中的相关记录同步，位于缓存中的对象称为持久化对象。

（3）第三当缓存中的持久化对象之间存在关联时，Session 保证不出现对象图的死锁。

8、请简述Hibernate与jdbc的联系

答：Hibernate是jdbc的轻量级封装，包括jdbc的与数据库的连接（用hibernate的配置文件实现，本质是封装了jdbc），和查询，删除等代码，都用面向对象的思想用代码联系起来，Hibernate通过hbm 配置文件（或注解）把持久化类的字段和数据库的字段关联起来比如数据库的id。

HQL语句也是面向对象的，它的查询语句不是查询数据库而是查询类的。

9、请简述Hibernate是如何实现分页的？如果不使用Hibernate提供的方法实现分页功能，则采用什么方式分页？

答：

（1）hibernate自带的分页机制:获得Session对象后从Session中获得Query对象。

用Query.setFirstResult()设置要显示的第一行数据， Query.setMaxResults()设置要显示的最后一行数据。

（2）不使用hibernate自带的分页,可采用sql语句分页

10、Hibernate是如何延迟加载(懒加载)?

答案：通过设置属性lazy进行设置是否需要懒加载

当Hibernate在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，就实现了延迟加载，他节省了服务器的内存开销，从而提高了服务器的性能。

11、hibernate都支持哪些缓存策略

答案：有4种：

放入二级缓存的对象，只读(Read-only);

非严格的读写(Nonstrict read/write)

读写； 放入二级缓存的对象可以读、写(Read/write)；

基于事务的策略(Transactional)



12、简述Session的find()方法以及Query接口的区别？

答：Session类的find()方法以及Query接口都支持HQL检索方式。这两者的区别在于，前者只是执行一些简单HQL查询语句的便捷方法，它不具有动态绑定参数的功能，而且在Hibernate3.x版本中，已经淘汰了find()方法；而Query接口才是真正的HQL查询接口，它提供了以上列出的各种查询功能。

13、Hibernate工作原理及为什么要用Hibernate?

答案：

1.读取并解析配置文件 2.读取并解析映射信息，创建SessionFactory 3. 打开Sesssion 4.创建事务Transation 5.持久化操作 6.提交事务7.关闭Session 8.关闭SesstionFactory

为什么要用：

1. 对JDBC访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。

2. Hibernate是一个基于JDBC的主流持久化框架，是一个优秀的ORM实现。他很大程度的简化DAO层的编码工作

3. hibernate使用Java反射机制，而不是字节码增强程序来实现透明性。

4. hibernate的性能非常好，因为它是个轻量级框架。映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。

14、Hibernate如何延迟加载(懒加载)，目的是什么?

答案：通过设置属性lazy进行设置是否需要懒加载。当Hibernate在查询数据的时候，数据并没有存在与内存中，当程序真正对数据的操作时，对象才存在与内存中，就实现了延迟加载，他节省了服务器的内存开销，从而提高了服务器的性能。

15、 什么是 O/RM？

对象关系映射(英语:(Object Relational Mapping，简称ORM，或O/RM，或O/R mapping)，是一种程序技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换 。从效果上说，它其实是创建了一个可在编程语言里使用的--"虚拟对象数据库"。

面向对象是从软件工程基本原则(如耦合、聚合、封装)的基础上发展起来的，而关系数据库则是从数学理论发展而来的，两套理论存在显著的区别。为了解决这个不匹配的现象，对象关系映射技术应运而生。

对象关系映射(Object-Relational Mapping)提供了概念性的、易于理解的模型化数据的方法。ORM方法论基于三个核心原则: 简单:以最基本的形式建模数据。 传达性:数据库结构被任何人都能理解的语言文档化。 精确性:基于数据模型创建正确标准化的结构。 典型地，建模者通过收集来自那些熟悉应用程序但不熟练的数据建模者的人的信息开发信息模型。建模者必须能够用非技术企业专家可以理解的术语在概念层次上与数据结构进行通讯。建模者也必须能以简单的单元分析信息，对样本数据进行处理。ORM专门被设计为改进这种联系。

简单的说:ORM相当于中继数据。具体到产品上，例如ADO.NET Entity Framework。DLINQ中实体类的属性[Table]就算是一种中继数据。

16、在 O/RM 中，JAVA里的类名，属性，对象分别转化成数据库中的什么？

类对应表

熟悉对应列

对象对应一条记录

17、hibernate.cfg.xml 中<property name=’”show\_sql”>true</property>的用处。

是否显示所生成 SQL 语句

18、ORACLE 能否使用 identity 这种主键生成策略？

不行。identity:由底层数据库生成标识符。identity是由数据库自己生成的，但这个主键必须设置为自增长，前提条件是低层数据库支持自动增长字段类型 ,一般来说象SQL，MYSQL这类的数据库都是选择identity

19、hibernate中get（）和load()区别?（5分）

答：

1）、get如果没有找到会返回null， load如果没有找到会抛出异常。

2）、get会先查一级缓存，再查二级缓存，然后查数据库；load会先查一级缓存，如果没有找到，就创建代理对象，等需要的时候去查询二级缓存和数据库。

20、 hibernate中N+1问题是什么?（5分）

答：

Hibernate中常会用到set，bag等集合表示1对多的关系，在获取实体的时候就能根据关系将关联的对象或者对象集取出。解决方法一个是延迟加载， 即lazy=true；一个是预先抓取， 即fetch=join。

21、 hibernate中inverse的好处?（5分）

答：

在关联关系中用inverse在控制由哪一端来控制关联关系。举customer和order的例子来说。他们是一对多的关系，如果只单向关联，且由customer控制关联关系，则如果我想添加一个order，则先取customer， 然后getOrders得到所有的order集合，然后往集合里面多加入一个order，然后save(customer)， 这样开销太大。 如果改双向关联且由order主控关系，则如果想为customer增加一个order， 则new一个order，然后给order设置customer，然后save(order)即可。

22、 如何优化hibernate框架?（5分）

答：1）使用双向一对多关联，不使用单向一对多

2）灵活使用单向一对多关联

3）不用一对一，用多对一取代

4）配置对象缓存，不使用集合缓存

5）一对多集合使用Bag,多对多集合使用Set

6）继承类使用显式多态

7) ac表字段要少，表关联不要怕多，有二级缓存撑腰

六、程序实现题

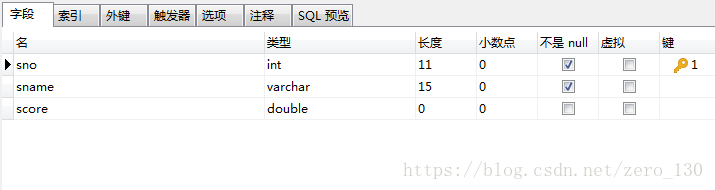
# 1、使用HQL语句实现一个简单登录例子(代码直接嵌入到JSP页面即可)；登录用户表loginusers，字段为id,userName，password；一个login.jsp登录页面；一个validate.jsp登录验证页面；一个success.jsp登录成功页面；一个error.jsp登录失败页面

|  |
| --- |
| login.jsp：  <%@ page language="java" import="java.util.\*" pageEncoding="utf-8"%>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>Hibernate login</title>  </head>  <body>  <h3>User Login</h3>  <form action="validate.jsp" method="post">  <label>Account:&nbsp;&nbsp;&nbsp;</label>  <input type="text" name="userName"/><br>  <label>Password:</label>  <input type="password" name="password"/><br><br>  <input type="reset" value="reset"/>  <input type="submit" value="submit"/>  </form>  </body>  </html>  validate.jsp：  <%@ page language="java" import="java.util.\*,com.bean.Loginusers" pageEncoding="utf-8"%>  <%@ page import="org.hibernate.\*,com.tools.HibernateSessionFactory" %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>hibernate validate.jsp</title>  </head>    <body>  <%  request.setCharacterEncoding("utf-8");  String account=request.getParameter("userName");  String password=request.getParameter("password");  boolean flag=false;  Loginusers user=null; //创建登录用户  user=(Loginusers)session.getAttribute("user");    if(user==null){ //判断是否是第一次登录  String hql="from Loginusers u where u.userName=? and u.password=?";  Session hsession=HibernateSessionFactory.getSession();  Query query=hsession.createQuery(hql);  query.setParameter(0,account);  query.setParameter(1,password);  List list=query.list();  if(list!=null){  Iterator it=list.iterator();  while(it.hasNext()){  user=(Loginusers)it.next();  session.setAttribute("user",user);  flag=true;  }  }  }else{  flag=true;  }  if(flag)  response.sendRedirect("success.jsp");  else  response.sendRedirect("error.jsp");  %>  </body> </html>  success.jsp：  <%@ page language="java" import="java.util.\*" import="com.bean.Loginusers" pageEncoding="utf-8"%>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>My JSP 'success.jsp' starting page</title>  </head>  <body>  <%  Loginusers user=(Loginusers)session.getAttribute("user");  String uName=user.getUserName();  %>  <h2 style="text-align:center;margin:200xp auto">Welcome <%=uName %>,Login Success</h2>  </body>  </html>  error.jsp：  <%@ page language="java" import="java.util.\*" pageEncoding="utf-8"%>    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <title>My JSP 'error.jsp' starting page</title>  </head>  <body>  <h2 style="text-align:center;margin:100px auto">Login Error</h2>  </body> |

2、增删改是基本功能,请写出增加一种水果信息(id,名称/价格/产地/数量/图片)到数据库中. 要求如果失败要回滚事务。（10分）

|  |
| --- |
| Fruit data = new Fruit();  data.setIds("i001");  data.setName("鸭梨");  data.setPrice(1.0);  data.setSource("山东");  data.setNumbers(1);  data.setImage("无");  try{  session.beginTransaction();  session.save(data);  session.getTransaction().commit();  }  catch(Exception es){  session.getTransaction().rollback();  }  session.close(); |

3、按照图片内容使用Hibernate往下表添加一条记录，并立刻查出添加内容进行输入。



（10分）

|  |
| --- |
| package com.test;    import java.util.List;    import org.hibernate.Query;  import org.hibernate.Session;  import org.hibernate.Transaction;    import com.bean.Student;  import com.tools.HibernateSessionFactory;    public class Test {  public static void main(String[] args){    Session session=HibernateSessionFactory.getSession();  Transaction ts=session.beginTransaction();  Student student=new Student();  student.setSno(1005);  student.setSname("gan");  student.setScore(100d);  try{  session.save(student);  ts.commit();  Query query=session.createQuery("from Student where id=1005");  List list=query.list();  Student stu=(Student)list.get(0);  System.out.println(stu);  }catch(Exception e){  e.printStackTrace();  ts.rollback();  }finally{  HibernateSessionFactory.closeSession();  }  }  } |

4、Course和Student多对多关联映射的hbm配置

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  <hibernate-mapping package="com.chinasofti.manytomany.doubl">  <class name="Course" table="t\_course">  <id name="id" column="tid">  <generator class="native"></generator>  </id>  <property name="name" column="tname"></property>  <set name="students" table="t\_choose">  <key column="courseid"></key>  <many-to-many class="Student" column="stuid"></many-to-many>  </set>  </class>  </hibernate-mapping>  <?xml version="1.0"?>  <!DOCTYPE hibernate-mapping PUBLIC  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  <hibernate-mapping package="com.chinasofti.manytomany.doubl">  <class name="Student" table="t\_student">  <id name="id" column="tid">  <generator class="native"></generator>  </id>  <property name="name" column="tname"></property>  <set name="courses" table="t\_choose">  <key column="stuid"></key>  <many-to-many class="Course" column="courseid"></many-to-many>  </set>  </class>  </hibernate-mapping> |

5、使用Maven,junit和Hibernate框架以及mysql实现学生表的student(字段有:id,name,age)的增删改查

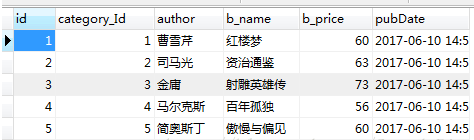
|  |
| --- |
| pom.xml:  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <scope>test</scope>  </dependency>  <!-- 添加Hibernate依赖 -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>4.1.1.Final</version>  </dependency>  <!-- 添加Log4J依赖 -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.16</version>  </dependency>    <!-- mysql数据库的驱动包 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.6</version>  </dependency>  </dependencies> |
| hibernate.cfg.xml:  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  <!-- Generated by MyEclipse Hibernate Tools. -->  <hibernate-configuration>  <session-factory>  <property name=*"connection.username"*>root</property>  <property name=*"connection.url"*>  jdbc:mysql://localhost:3306/gg  </property>  <property name=*"dialect"*>  org.hibernate.dialect.MySQLDialect  </property>  <property name=*"connection.password"*>123456</property>  <property name=*"connection.driver\_class"*>  com.mysql.jdbc.Driver  </property>  <property name=*"show\_sql"*>true</property>  <property name=*"format\_sql"*>true</property>  <property name=*"hbm2ddl.auto"*>update</property>  <mapping class=*"com.pojo.Student"*/>  </session-factory>  </hibernate-configuration> |
| Student实体类:  **package** com.pojo;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.Id;  @Entity  **public** **class** Student {  @Id  @GeneratedValue  **private** **int** id;  @Column  **private** String name;  @Column  **private** **int** age;  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** **int** getAge() {  **return** age;  }  **public** **void** setAge(**int** age) {  **this**.age = age;  }  **public** Student() {  // **TODO** Auto-generated constructor stub  }  **public** Student(**int** id, String name, **int** age) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.age = age;  }  **public** Student(String name, **int** age) {  **super**();  **this**.name = name;  **this**.age = age;  }  @Override  **public** String toString() {  **return** "Student [id=" + id + ", name=" + name + ", age=" + age + "]";  }  } |
| 单元测试类实现增删改查:  **package** com.test;  **import** java.util.List;  **import** org.hibernate.Session;  **import** org.hibernate.SessionFactory;  **import** org.hibernate.Transaction;  **import** org.hibernate.cfg.Configuration;  **import** org.junit.Test;  **import** com.pojo.Student;  **public** **class** StudentTest {  SessionFactory factory = **new** Configuration().configure().~~buildSessionFactory~~();  @Test  **public** **void** testAdd() {  Session session = factory.openSession();  Transaction transaction = session.beginTransaction();  Student student=**new** Student("张三", 20);  session.save(student);  transaction.commit();  session.close();  }  @Test  **public** **void** testUpdate() {  Session session = factory.openSession();  Transaction transaction = session.beginTransaction();  Student student=**new** Student(1,"李四", 18);  session.update(student);  transaction.commit();  session.close();  }  @Test  **public** **void** testDel() {  Session session = factory.openSession();  Transaction transaction = session.beginTransaction();  Student student=(Student) session.get(Student.**class**, 1);  session.delete(student);  transaction.commit();  session.close();  }  @Test  **public** **void** testGetById() **throws** Exception {  Session session = factory.openSession();  Student student = (Student) session.get(Student.**class**, 1);  System.***out***.println(student);  session.close();  }  @Test  **public** **void** testGetList() **throws** Exception {  Session session = factory.openSession();  List<Student> list = session.createQuery("from Student").list();  **for** (Student student : list) {  System.***out***.println(student);  }  session.close();  }  } |

6、Hibernate实现曾删改查及分页

（10分）

|  |
| --- |
| package com.entity;  public class Student {  private Integer sid;  private String sname;  private String password;  private String sex;  private Integer cid;    public Student() {  }  public Student(Integer sid, String sname, String password, String sex,  Integer cid) {  this.sid = sid;  this.sname = sname;  this.password = password;  this.sex = sex;  this.cid = cid;  }  public Student(String sname, String password, String sex,  Integer cid) {  this.sname = sname;  this.password = password;  this.sex = sex;  this.cid = cid;  }  public Integer getSid() {  return sid;  }  public void setSid(Integer sid) {  this.sid = sid;  }  public String getSname() {  return sname;  }  public void setSname(String sname) {  this.sname = sname;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getSex() {  return sex;  }  public void setSex(String sex) {  this.sex = sex;  }  public Integer getCid() {  return cid;  }  public void setCid(Integer cid) {  this.cid = cid;  }  @Override  public String toString() {  return "Student [cid=" + cid + ", password=" + password + ", sex="  + sex + ", sid=" + sid + ", sname=" + sname + "]";  }      }  [复制代码](javascript:void(0);)  5.配置映射文件  [复制代码](javascript:void(0);)  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" >  <hibernate-mapping>  <!--1.实体类的字段和数据库的字段一一映射  name:实体类的全名  table:对应的数据库表名,默认值为类名  -->  <class name="com.entity.Student" table="STUDENT">    <!--1.1主键配置  id:对应表中的主键  name:实体类中的属性名  type:主键属性的数据类型  column:属性名对应表的字段名,默认值为属性名-->  <id name="sid" type="java.lang.Integer" column="SID">    <!--generator:为持久化实例产生一个唯一标识-->  <generator class="sequence">    <!--数据库的序列名-->  <param name="sequence">SEQ\_STUDENT</param>  </generator>  </id>    <!--1.2普通字段配置-->  <property name="sname" type="java.lang.String" column="SNAME"/>  <property name="password" type="java.lang.String" column="PASSWORD"/>  <property name="sex" type="java.lang.String" column="SEX"/>  <property name="cid" type="java.lang.Integer" column="CID"/>  </class>  </hibernate-mapping>  [复制代码](javascript:void(0);)  6.配置主配置文件  [复制代码](javascript:void(0);)  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd" >  <hibernate-configuration>  <!--通常一个session-factory节点代表一个数据库-->  <session-factory>  <!--1.配置数据库的驱动-->  <property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>    <!--2.配置数据库的url地址-->  <property name="connection.url">jdbc:oracle:thin:@127.0.0.1:1521:orcl</property>    <!--3,配置数据库的username-->  <property name="connection.username">holly</property>    <!--4.配置数据库的密码-->  <property name="connection.password">sys</property>    <!--5.配置数据库的方言-->  <property name="dialect">org.hibernate.dialect.Oracle10gDialect</property>    <!--6.显示sql语句-->  <property name="show\_sql">true</property>    <!--7.sql语句格式化-->  <property name="format\_sql">true</property>    <!--8.加载映射文件-->  <mapping resource="com/entity/Student.hbm.xml"/>  </session-factory>  </hibernate-configuration>  [复制代码](javascript:void(0);)  7.创建一个工具类  [复制代码](javascript:void(0);)  package com.util;  import org.hibernate.HibernateException;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.cfg.Configuration;    public class Hibernate\_Util {  private static ThreadLocal<Session> threadLocal=new ThreadLocal<Session>();  private static Configuration configuration=null;  private static SessionFactory sessionFactory=null;  static{    try {  configuration=new Configuration().configure("/hibernate.cfg.xml");  sessionFactory=configuration.buildSessionFactory();  } catch (HibernateException e) {  System.out.println("解析xml和创建Session工厂error");  e.printStackTrace();  }    }    public static Session getSession(){  Session session=threadLocal.get();  if(session==null){  session=sessionFactory.openSession();  threadLocal.set(session);  }  return session;  }  public static void closeSession(){  Session session=threadLocal.get();  if(session!=null){  threadLocal.set(null);  session.close();  }  }    8.创建分页实体类  [复制代码](javascript:void(0);)  package com.page;  public class Page {  private Integer pagesize; //页面大小  private Integer pageno; //当前页  private Integer startrow; //起始行  private Integer totalpage; //总页数  private Integer totalcount; //总条数      public Page() {  }  public Page(Integer pageSize, Integer pageNo, Integer totalCount) {  this.pagesize = pageSize; //页面大小  this.pageno = pageNo; //当前页  this.totalcount = totalCount; //计算总条数  this.setStartrow(pageNo, pageSize); //计算起始行  this.setTotalpage(totalCount, pageSize); //计算总页数  }  public Integer getPagesize() {  return pagesize;  }  public void setPagesize(Integer pageSize) {  this.pagesize = pageSize;  }  public Integer getPageno() {  return pageno;  }  public void setPageno(Integer pageNo) {  this.pageno = pageNo;  }  public Integer getStartrow() {  return startrow;  }    public void setStartrow(Integer pageNo,Integer pageSize) {  this.startrow = (pageNo-1)\*pageSize;  }  public Integer getTotalpage() {  return totalpage;  }    public void setTotalpage(Integer totalCount,Integer pageSize) {  this.totalpage = totalCount%pageSize==0?totalCount/pageSize:totalCount/pageSize+1;  }  public Integer getTotalcount() {  return totalcount;  }  public void setTotalCount(Integer totalCount) {  this.totalcount = totalCount;  }  @Override  public String toString() {  return "[{\"pageno\":" + pageno + ",\"pagesize\":" + pagesize  + ",\"startrow\":" + startrow + ",\"totalcount\":" + totalcount  + ",\"totalpage\":" + totalpage + "}]";  }        }  [复制代码](javascript:void(0);)  9.创建Dao类  [复制代码](javascript:void(0);)  package com.dao;  import java.util.List;  import com.entity.Student;  import com.page.Page;  public interface StudentDao {    Student findByNameAndPwd(String sname,String password);    List<Student> findAll();    int save(Student stu);    Student findById(Integer sid);    int update(Student stu);    int delete(Student stu);  /\*\*  \* 7.分页查询  \*/  List<Student> findPage(Page page);  /\*\*  \* 8.查询总条数  \*/  int getTotalCount();        }  [复制代码](javascript:void(0);)  10.实现Dao接口  [复制代码](javascript:void(0);)  package com.dao.impl;  import java.util.List;  import org.hibernate.HibernateException;  import org.hibernate.Query;  import org.hibernate.Session;  import com.dao.StudentDao;  import com.entity.Student;  import com.page.Page;  import com.util.Hibernate\_Util;  public class StudentDaoImpl implements StudentDao{  /\*\*  \* 1.查询所有:立即加载  \*/  public List<Student> findAll() {  //1.获取session对象  Session session=Hibernate\_Util.getSession();  //2.定义hql语句  String hql="from Student";  //3.Session创建Query对象  Query query=session.createQuery(hql);  //4.query对象执行操作  List<Student> it=query.list();  //5.释放资源  Hibernate\_Util.closeSession();  return it;  }  /\*\*  \* 2.根据用户名和密码查询  \*/  public Student findByNameAndPwd(String sname, String password) {  //1.获取session对象  Session session=Hibernate\_Util.getSession();  //2.定义hql语句(命名参数)  String hql="from Student where sname=:sname and password=:password";  //3.Session创建Query对象  Query query=session.createQuery(hql);  //4.query对象给占位符赋值  /\*query.setString("sname", sname);  query.setString("password", password);\*/  query.setParameter("sname", sname);  query.setParameter("password", password);    // query.setProperties(user);  //5.执行操作(uniqueResult()查询的单行数据)  Student stu=(Student) query.uniqueResult();  //6.释放资源  Hibernate\_Util.closeSession();  return stu;    }    /\*\*  \* 根据id主键查询单条  \*/  public Student findById(Integer sid) {  //1.获取session对象  Session session=Hibernate\_Util.getSession();  //2.利用session根据主键查询  Student stu=(Student) session.get(Student.class, sid);    Hibernate\_Util.closeSession();  // session.clear();  return stu;  }  /\*\*  \* 删除对象  \*/  public int delete(Student stu) {  int num=1;  //1.获取session对象  Session session=Hibernate\_Util.getSession();    try {  //session删除操作  session.delete(stu);  session.beginTransaction().commit();  } catch (HibernateException e) {  num=0;  // TODO Auto-generated catch block  e.printStackTrace();  }finally{  //5.释放资源  Hibernate\_Util.closeSession();  }    return num;  }  /\*\*  \* 添加  \*/  public int save(Student stu) {  int num=1;  //1.获取session对象  Session session=Hibernate\_Util.getSession();  //执行操作  try {  session.save(stu);  session.beginTransaction().commit();  } catch (HibernateException e) {  num=0;  e.printStackTrace();  }finally{  //5.释放资源  Hibernate\_Util.closeSession();  }  return num;  }  /\*\*  \* 修改  \*/  public int update(Student stu) {  int num=1;  //1.获取session对象  Session session=Hibernate\_Util.getSession();  try {  //执行操作  //该对象的主键id必须存在  session.update(stu);  session.beginTransaction().commit();  } catch (HibernateException e) {  num=0;  e.printStackTrace();  }finally{  //5.释放资源  Hibernate\_Util.closeSession();  }  return num;  }  /\*\*  \* 7.分页查询  \*/  public List<Student> findPage(Page page) {  //1.获取session  Session session=Hibernate\_Util.getSession();    //2.定义查询最大记录数的hql  String hql="from Student";    //3.定义查询最大记录数的Query对象  Query querypage=session.createQuery(hql);    //4.查询最大记录数的数据  querypage.setMaxResults(page.getPagesize());    //5.确定查询起点  querypage.setFirstResult(page.getStartrow());    //6.分页查询  List<Student> list=querypage.list();    //7.关闭session  Hibernate\_Util.closeSession();    return list;  }  /\*\*  \* 8.查询总条数  \*/  public int getTotalCount() {  //1.获取session  Session session=Hibernate\_Util.getSession();    //2.定义查询总条数hql语句  String hqlcount="select count(\*) from Student";    //3.利用Session创建Query对象  Query querycount=session.createQuery(hqlcount);    //4.获取总条数(返回单行数据uniqueResult())  Integer totalCount=Integer.parseInt(querycount.uniqueResult().toString());  //5.释放资源  Hibernate\_Util.closeSession();  return totalCount;  }    }  [复制代码](javascript:void(0);)  11.创建业务层的接口  [复制代码](javascript:void(0);)  package com.service;  import java.util.List;  import com.entity.Student;  import com.page.Page;  /\*\*  \* 业务接口  \* @author pc  \*  \*/  public interface StudentService {  /\*\*  \* 1.登录的业务  \* @param sname  \* @param password  \* @return  \*/  Student login(String sname,String password);  /\*\*  \* 2.注册  \* @param stu  \* @return  \*/  int register(Student stu);    /\*\*  \* 3.查询所有  \*/  List<Student> findAll();  /\*\*  \* 4.根据主键id查询  \* @param sid  \* @return  \*/  Student findById(Integer sid);  /\*\*  \* 5.修改  \*/  int update(Student stu);  /\*\*  \* 6.删除  \*/  int delete(Student stu);    /\*\*  \* 7.分页查询  \*/  List<Student> findPage(Page page);  /\*\*  \* 8.查询总条数  \*/  int getTotalCount();    }  [复制代码](javascript:void(0);)  12.实现业务层接口  [复制代码](javascript:void(0);)  package com.service.impl;  import java.util.List;  import com.dao.StudentDao;  import com.dao.impl.StudentDaoImpl;  import com.entity.Student;  import com.page.Page;  import com.service.StudentService;  public class StudentServiceImpl implements StudentService {  /\*\*  \* 使用多态引用数据访问层对象  \*/  private StudentDao dao = new StudentDaoImpl();  /\*\*  \* 登录  \*/  public Student login(String sname, String password) {  return dao.findByNameAndPwd(sname, password);  }  public int delete(Student stu) {  // TODO Auto-generated method stub  return dao.delete(stu);  }  public List<Student> findAll() {  // TODO Auto-generated method stub  return dao.findAll();  }  public Student findById(Integer sid) {  // TODO Auto-generated method stub  return dao.findById(sid);  }  public int register(Student stu) {  // TODO Auto-generated method stub  return dao.save(stu);  }  public int update(Student stu) {  // TODO Auto-generated method stub  return dao.update(stu);  }  public List<Student> findPage(Page page) {  // TODO Auto-generated method stub  return dao.findPage(page);  }  public int getTotalCount() {  // TODO Auto-generated method stub  return dao.getTotalCount();  }  }  [复制代码](javascript:void(0);)  13.创建Servlet类  Servlet类里的增删改查我是分开写的,所以就不一一粘贴代码,这里只粘贴部分  [复制代码](javascript:void(0);)  package com.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.entity.Student;  import com.service.StudentService;  import com.service.impl.StudentServiceImpl;  public class RegisterServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doPost(request, response);  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  response.setContentType("text/html;charset=UTF-8");  // 2.接受参数  String sname = request.getParameter("sname");  String password = request.getParameter("password");  String sex = request.getParameter("sex");  Integer cid = Integer.valueOf(request.getParameter("cid"));  Student stu = new Student(sname, password, sex, cid);  // 3.业务处理  StudentService service = new StudentServiceImpl();  int num = service.register(stu);  if (num > 0) {  System.out.println("register success!");  response.sendRedirect("FindAllServlet");  } else {  System.out.println("register fail!");  }  }  }  [复制代码](javascript:void(0);)  [复制代码](javascript:void(0);)  package com.servlet;  import java.io.IOException;  import java.io.PrintWriter;  import java.util.List;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.codehaus.jackson.map.ObjectMapper;  import com.entity.Student;  import com.page.Page;  import com.service.StudentService;  import com.service.impl.StudentServiceImpl;  /\*\*  \* 分页查询  \*  \* @author pc  \*  \*/  public class FindPageServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doPost(request, response);  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  response.setContentType("text/html;charset=UTF-8");  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  /\*\*  \* ☆  \* 在使用ajax往前台通过输出流对象的print方法发送json时  \* 该行获取输出流对象的代码必须放在  \* response.setContentType("text/html;charset=UTF-8");  \* 之后,否则ajax回调时,页面拿到的中文数据就永远都是乱码,  \* 原因是:如果将改行代码写在处理客户端乱码之前,表示编码格式已经确定,  \* 所以,编码格式的处理应该放在获取PrintWriter对象之前  \*  \*/  PrintWriter out=response.getWriter();    // 2.接受参数  String no=request.getParameter("pageNo");    // 3.业务处理  StudentService service = new StudentServiceImpl();    int pageSize=3; //页面大小  int pageNo=1; //默认的pageNo为1  if(no!=null && no!=""){  pageNo=Integer.valueOf(no);  }  //获取总条数  int totalCount=service.getTotalCount();  //封装分页所需字段  Page page=new Page(pageSize, pageNo, totalCount);  //分页查询  List<Student> list = service.findPage(page);    ObjectMapper om=new ObjectMapper();  String str=om.writeValueAsString(list);  String pagestr=om.writeValueAsString(page);  str="{\"page\":"+pagestr+",\"list\":"+str+"}";  System.out.println(str);    out.print(str);  /\*if (list != null) {  System.out.println("find page success");  request.setAttribute("list", list);  request.setAttribute("page", page);  request.getRequestDispatcher("index.jsp").forward(request, response);  } else {  System.out.println("find page fail!");  }\*/  }  }  [复制代码](javascript:void(0);)  [复制代码](javascript:void(0);)  package com.servlet;  import java.io.IOException;  import java.io.PrintWriter;  import java.util.List;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.entity.Student;  import com.page.Page;  import com.service.StudentService;  import com.service.impl.StudentServiceImpl;  public class FindAllServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doPost(request, response);  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  response.setContentType("text/html;charset=UTF-8");    /\*\*  \* ☆  \* 在使用ajax往前台通过输出流对象的print方法发送json时  \* 该行获取输出流对象的代码必须放在  \* response.setContentType("text/html;charset=UTF-8");  \* 之后,否则ajax回调时,页面拿到的中文数据就永远都是乱码,  \* 原因是:如果将改行代码写在处理客户端乱码之前,表示编码格式已经确定,  \* 所以,编码格式的处理应该放在获取PrintWriter对象之前  \*  \*/  // 2.接受参数  String no=request.getParameter("pageNo");    // 3.业务处理  StudentService service = new StudentServiceImpl();    int pageSize=3; //页面大小  int pageNo=1; //默认的pageNo为1  if(no!=null && no!=""){  pageNo=Integer.valueOf(no);  }  //获取总条数  int totalCount=service.getTotalCount();  Page page=new Page(pageSize, pageNo, totalCount);    List<Student> list = service.findPage(page);  if(list!=null){  System.out.println("find All page success");  request.setAttribute("list", list);  request.setAttribute("page", page);  request.getRequestDispatcher("index.jsp").forward(request, response);  }else{  System.out.println("find All page fail!");  }  }    }  [复制代码](javascript:void(0);)    代码  [复制代码](javascript:void(0);)  package com.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.entity.Student;  import com.service.StudentService;  import com.service.impl.StudentServiceImpl;  public class LoginServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doPost(request, response);  }    public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  response.setContentType("text/html;charset=UTF-8");    //2.接受参数  String sname=request.getParameter("sname");  String password=request.getParameter("password");  //3.业务处理  StudentService service=new StudentServiceImpl();  Student stu=service.login(sname, password);  if(stu!=null){  System.out.println("login success!");  //4.页面跳转  request.getSession().setAttribute("sname", sname);  response.sendRedirect("FindAllServlet");  }else{  System.out.println("login fail!");    }  }  }  [复制代码](javascript:void(0);)  [复制代码](javascript:void(0);)  package com.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.entity.Student;  import com.service.StudentService;  import com.service.impl.StudentServiceImpl;  import com.util.Hibernate\_Util;  public class FindByIdSerlvet extends HttpServlet {  private static final long serialVersionUID = 1L;  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doPost(request, response);  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  response.setContentType("text/html;charset=UTF-8");  // 2.接受参数  Integer sid=Integer.valueOf(request.getParameter("sid"));    // 3.业务处理  StudentService service = new StudentServiceImpl();    Student stu=service.findById(sid);  System.out.println("stu:"+stu);  if(stu!=null){  System.out.println("根据id查询到对象");  request.setAttribute("stu", stu);  request.getRequestDispatcher("update.jsp").forward(request, response);  }else{  System.out.println("没有根据id查询到对象");  }  }  }  [复制代码](javascript:void(0);)  [复制代码](javascript:void(0);)  package com.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.entity.Student;  import com.service.StudentService;  import com.service.impl.StudentServiceImpl;  public class UpdateServlet extends HttpServlet {  private static final long serialVersionUID = 1L;  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doPost(request, response);  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  response.setContentType("text/html;charset=UTF-8");  // 2.接受参数  Integer sid=Integer.valueOf(request.getParameter("sid"));  String sname = request.getParameter("sname");  String password = request.getParameter("password");  String sex = request.getParameter("sex");  Integer cid=Integer.valueOf(request.getParameter("cid"));  Student stu=new Student(sid,sname, password, sex, cid);    // 3.业务处理  StudentService service = new StudentServiceImpl();  int num=service.update(stu);  if(num>0){  System.out.println("update success!");  response.sendRedirect("FindAllServlet");  }else{  System.out.println("update fail!");  }  }  }  [复制代码](javascript:void(0);)  [复制代码](javascript:void(0);)  package com.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.entity.Student;  import com.service.StudentService;  import com.service.impl.StudentServiceImpl;  public class DeleteSerlvet extends HttpServlet {  private static final long serialVersionUID = 1L;  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doPost(request, response);  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  response.setContentType("text/html;charset=UTF-8");  // 2.接受参数  Integer sid=Integer.valueOf(request.getParameter("sid"));  // 3.业务处理  StudentService service = new StudentServiceImpl();  Student stu=service.findById(sid);  if (stu != null) {  System.out.println("根据id查询成功");  int num=service.delete(stu);  if(num>0){  System.out.println("delete success!!");  // 4.页面跳转  response.sendRedirect("FindAllServlet");    }else{  System.out.println("delete success!!");    }  } else {  System.out.println("根据id查询失败!");  }  }  }  [复制代码](javascript:void(0);)  14.配置web.xml  [复制代码](javascript:void(0);)  <?xml version="1.0" encoding="UTF-8"?>  <web-app version="2.5"  xmlns="http://java.sun.com/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">    <servlet>  <servlet-name>LoginServlet</servlet-name>  <servlet-class>com.servlet.LoginServlet</servlet-class>  </servlet>  <servlet>  <servlet-name>FindAllServlet</servlet-name>  <servlet-class>com.servlet.FindAllServlet</servlet-class>  </servlet>  <servlet>  <servlet-name>FindByIdSerlvet</servlet-name>  <servlet-class>com.servlet.FindByIdSerlvet</servlet-class>  </servlet>  <servlet>  <servlet-name>UpdateServlet</servlet-name>  <servlet-class>com.servlet.UpdateServlet</servlet-class>  </servlet>  <servlet>  <servlet-name>DeleteSerlvet</servlet-name>  <servlet-class>com.servlet.DeleteSerlvet</servlet-class>  </servlet>  <servlet>  <servlet-name>RegisterServlet</servlet-name>  <servlet-class>com.servlet.RegisterServlet</servlet-class>  </servlet>  <servlet>  <servlet-name>FindPageServlet</servlet-name>  <servlet-class>com.servlet.FindPageServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>LoginServlet</servlet-name>  <url-pattern>/LoginServlet</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>FindAllServlet</servlet-name>  <url-pattern>/FindAllServlet</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>FindByIdSerlvet</servlet-name>  <url-pattern>/FindByIdSerlvet</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>UpdateServlet</servlet-name>  <url-pattern>/UpdateServlet</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>DeleteSerlvet</servlet-name>  <url-pattern>/DeleteSerlvet</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>RegisterServlet</servlet-name>  <url-pattern>/RegisterServlet</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>FindPageServlet</servlet-name>  <url-pattern>/FindPageServlet</url-pattern>  </servlet-mapping>  <welcome-file-list>  <welcome-file>login.jsp</welcome-file>  </welcome-file-list>  </web-app>  [复制代码](javascript:void(0);)  15.首页  [复制代码](javascript:void(0);)  <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  <%  String path = request.getContextPath();  String basePath = request.getScheme() + "://"  + request.getServerName() + ":" + request.getServerPort()  + path + "/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="<%=basePath%>">  <title>My JSP 'index.jsp' starting page</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  </head>  <body>  <CENTER>  <h3>学生信息表</h3>  <table border="2">  <tr>  <td>  序号  </td>  <td>  姓名  </td>  <td>  性别  </td>  <td align="center">  操作  </td>  </tr>    <c:forEach var="i" items="${list}" varStatus="f">  <tr <c:if test="${f.index%2==0}">style="background-color:orange;"</c:if>>  <td>  ${i.sid }  </td>  <td>  ${i.sname }  </td>  <td>  ${i.sex }  </td>  <td>  <a href="register.jsp">添加</a>  &nbsp;&nbsp;  <a href="FindByIdSerlvet?sid=${i.sid}">修改</a>  &nbsp;&nbsp;  <a href="DeleteSerlvet?sid=${i.sid}">删除</a>    </td>  </tr>  </c:forEach>    </table>  <div>      第${page.pageno}/${page.totalpage}页 &nbsp;&nbsp;  <a href="FindAllServlet?pageNo=1">首页</a>  <c:choose>  <c:when test="${page.pageno gt 1}">  <a href="FindAllServlet?pageNo=${page.pageno-1 }">上一页</a>  </c:when>  <c:otherwise>  <a href="javascript:alert('已经是第一页了,没有上一页!');">上一页</a>  </c:otherwise>  </c:choose>  <c:choose>  <c:when test="${page.pageno lt page.totalpage}">  <a href="FindAllServlet?pageNo=${page.pageno+1 }">下一页</a>  </c:when>  <c:otherwise>  <a href="javascript:alert('已经是最有一页了,没有下一页!');">下一页</a>  </c:otherwise>  </c:choose>    <a href="FindAllServlet?pageNo=${page.totalpage}">末页</a>  &nbsp;&nbsp;  共${page.totalcount}条    </div>    </CENTER>    </body>  </html>  [复制代码](javascript:void(0);)  16.登录页面  [复制代码](javascript:void(0);)  <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%  String path = request.getContextPath();  String basePath = request.getScheme() + "://"  + request.getServerName() + ":" + request.getServerPort()  + path + "/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="<%=basePath%>">  <title>My JSP 'index.jsp' starting page</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  </head>  <body>  <center>  <fieldset style="width: 300px;">  <legend>  登录  </legend>  <form action="LoginServlet" method="post">  <table>  <tr>  <td>  用户名:  </td>  <td>  <input type="text" name="sname" />  </td>  </tr>  <tr>  <td>  密码:  </td>  <td>  <input type="password" name="password" />  </td>  </tr>  <tr>  <td>  <input type="submit" value="提交" />  </td>  <td>  <input type="reset" value="重置" />  </td>  </tr>  </table>  </form>  </fieldset>  </center>  </body>  </html>  [复制代码](javascript:void(0);)  17.注册页面  [复制代码](javascript:void(0);)  <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%  String path = request.getContextPath();  String basePath = request.getScheme() + "://"  + request.getServerName() + ":" + request.getServerPort()  + path + "/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="<%=basePath%>">  <title>My JSP 'index.jsp' starting page</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  </head>  <body>  <center>  <fieldset style="width: 500px;">  <legend>  注册  </legend>  <form action="RegisterServlet" method="post">  <table>  <tr>  <td>  用户名:  </td>  <td>  <input type="text" name="sname"/>  </td>  </tr>  <tr>  <td>  密码:  </td>  <td>  <input type="password" name="password"/>  </td>  </tr>  <tr>  <td>  性别:  </td>  <td>  <input type="radio" name="sex" value="男" checked="checked"/>男  <input type="radio" name="sex" value="女"/>女  </td>  </tr>  <tr>  <td>  班级编号:  </td>  <td>  <select name="cid">  <option value="1">TB07</option>  <option value="2">TB13</option>  <option value="3">TB24</option>  <option value="4">TB31</option>  </select>  </td>  </tr>  <tr>  <td>  <input type="submit" value="注册" />  </td>  <td>  <input type="reset" value="重置" />  </td>  </tr>  </table>  </form>  </fieldset>  </center>  </body>  </html>  [复制代码](javascript:void(0);)  18.修改  [复制代码](javascript:void(0);)  <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%  String path = request.getContextPath();  String basePath = request.getScheme() + "://"  + request.getServerName() + ":" + request.getServerPort()  + path + "/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="<%=basePath%>">  <title>My JSP 'index.jsp' starting page</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  </head>  <body>  <center>  <fieldset style="width: 500px;">  <legend>  <br>修改  </legend>  <form action="UpdateServlet?sid=${stu.sid}" method="post">  <table>  <tr>  <td>  用户名:  </td>  <td>  <input type="text" name="sname" value="${stu.sname}"/>  </td>  </tr>  <tr>  <td>  密码:  </td>  <td>  <input type="password" name="password" value="${stu.password}"/>  </td>  </tr>  <tr>  <td>  性别:  </td>  <td>  <c:choose>  <c:when test="${stu.sex eq '男'}">  <input type="radio" name="sex" value="男" checked="checked"/>男  <input type="radio" name="sex" value="女"/>女  </c:when>  <c:otherwise>  <input type="radio" name="sex" value="男"/>男  <input type="radio" name="sex" value="女" checked="checked"/>女  </c:otherwise>  </c:choose>  </td>  </tr>  <tr>  <td>  班级编号:  </td>  <td>  <select name="cid">  <option value="1">TB07</option>  <option value="2">TB13</option>  <option value="3">TB24</option>  <option value="4">TB31</option>  </select>  </td>  </tr>    <tr>  <td>  <input type="submit" value="修改" />  </td>  <td>  <input type="reset" value="重置" />  </td>  </tr>  </table>  </form>  </fieldset>  </center>  </body>  </html> |

7、已有两张表，根据要求进行查询。



查询所有图书的书名、价格和类别。

|  |
| --- |
| Session session = null;  try {  session = SessionUtil.getSession();  List<Object[]> list = session.createQuery("select BName,BPrice from TBook").list();  for(Object[] book : list){  System.out.println("书名："+book[0]+"\t\t价格："+book[1]);  }  } catch (Exception e) {  e.printStackTrace();  }finally{  SessionUtil.close(session);  } |

8、在 ORACLE 中，要设定一个持久对象的主键生成策略是 sequence,并且设 sequence 的名字为 user\_seq时，映射文件该怎么写？

**<generator class="sequence">**

**<param name="sequence">** user\_seq**</param>**

**</generator>**

**<param name="sequence">** user\_seq**</param>** **指定sequence的名称**

2在Hibernate中，给定sql和 hql分别获取数据TblFwxx 集合。已知代码

1)hql写法：

Session session=this.getSession();

String hql=”from TblFwxx fw where fw.title like ?”;

1. sql写法:

Session session=this.getSession();

String sql=”select \* from TblFwxx fw where fw.title like ?”;