目录

[1) 今日课程概述04:03(了解) 2](#_Toc523412284)

[2) 回顾jdbc操作完成保存用户 09:40 (理解) 2](#_Toc523412285)

[3) orm思想的引入07:18(理解) 2](#_Toc523412286)

[4)-orm思想的概述 03:38 (理解) 3](#_Toc523412287)

[5) hibernate介绍和jpa引入 05:42 (了解) 3](#_Toc523412288)

[6) jpa介绍 07:03 (理解) 3](#_Toc523412289)

[07)jpa的优势 03:33 (理解) 4](#_Toc523412290)

[08)案例说明 03:34(掌握) 4](#_Toc523412291)

[09-入门案例：创建maven工程导入坐标 05:21(掌握) 5](#_Toc523412292)

[10-入门案例：创建jpa的核心配置文件 07:49(掌握) 5](#_Toc523412293)

[11-入门案例：配置jpa的核心配置文件 08:35(掌握) 6](#_Toc523412294)

[12-入门案例：编写客户的实体类 03:43(掌握) 6](#_Toc523412295)

[13-入门案例：使用jpa注解配置映射关系 09:13(掌握) 6](#_Toc523412296)

[14-入门案例：jpa的操作步骤说明05:05(掌握) 7](#_Toc523412297)

[15-入门案例：完成客户保存(掌握) 7](#_Toc523412298)

[16-入门案例：验证客户保存和自动生成数据库表的说明05:46(理解) 8](#_Toc523412299)

[17-主键生成策略的说明06:26 (理解) 8](#_Toc523412300)

[18-主键生成策略的演示 04:36(掌握) 8](#_Toc523412301)

[19-api对象讲解 12:03(掌握) 8](#_Toc523412302)

[20-抽取jpaUtils工具类 06:47(掌握) 10](#_Toc523412303)

[21-验证工具类 02:54(掌握) 10](#_Toc523412304)

[22-find方法：根据id查询客户 08:07(掌握) 10](#_Toc523412305)

[23-延迟加载与立即加载 12:26(掌握) 10](#_Toc523412306)

[24-根据id删除客户 03:35(掌握) 10](#_Toc523412307)

[25-客户的更新操作 03:58(掌握) 11](#_Toc523412308)

[26-jpql的介绍04:11(理解) 11](#_Toc523412309)

[27-jpql查询：查询全部 08:04(掌握) 11](#_Toc523412310)

[28-jpql查询：倒序03:20(掌握) 11](#_Toc523412311)

[29-jpql查询：统计查询 05:47(掌握) 12](#_Toc523412312)

[30-jpql查询：分页查询04:57(掌握) 12](#_Toc523412313)

[31-jpql查询：条件查询 04:52(掌握) 12](#_Toc523412314)

## 1) 今日课程概述04:03(了解)

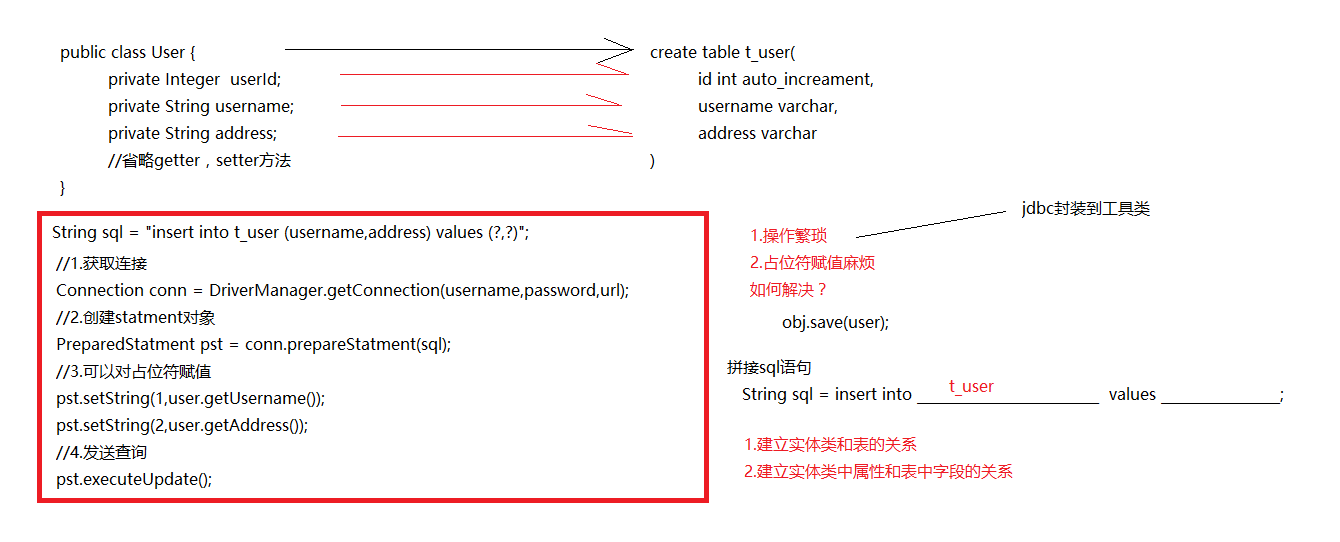
|  |
| --- |
| ##SpringData-JPA 课程整体概述     day01orm思想和hibernate 以及jpa概述的基本操作     day02 SpringdataJPA 的基本操作及运行原理     day03 多表查询及复杂查询  ## day01 概述  1)ORM 思想     2)hibername 框架介绍     3)JPA      4)JPA 基本操作         i)搭建环境         ii) 单表的增删改查 |

## 2) 回顾jdbc操作完成保存用户 09:40 (理解)

|  |
| --- |
| 传统jdbc操作存在的问题:  2.1) 操作繁琐  2.2) 占位符赋值麻烦 |

## 3) orm思想的引入07:18(理解)

|  |
| --- |
| 3.1) 操作繁琐?  封装工具类  3.2) 占位符赋值麻烦?  思路 :我们能不能通过面向对象的方法进行封装呢? 达到封装好之后 save(Object obj 的效果)  思路延伸: 我们能不能自动 拼接sql呢? 只要完成了自动拼接sql 我们就能实现上述的想法 |
| 通过分析:我们发现,只要能 1)建立实体类之间的对应关系  2)建立实体类属性和表字段之间的关系  这样我们就能够拼接出来sql 达到封装的目的 |
|  |



|  |
| --- |
| 上述 建立了实体类对象和数据库表之间的映射关系,我们就叫做  对象-关系-映射 (Object Relational Mapping) 简称ORM |
|  |
|  |

## 4)-orm思想的概述 03:38 (理解)

|  |
| --- |
| 主要目的：操作实体类就相当于操作数据库表save(User)  建立两个映射关系：  实体类和表的映射关系  实体类中属性和表中字段的映射关系  不再重点关注：sql语句    实现了ORM思想的框架：mybatis，hibernate |
|  |
| “Orm思想”: 是一个思想,一个想法  “ORM 框架” : 实现了上述思想的框架  Mybatis 半自动  Hibernate 全自动 |

## 5) hibernate介绍和jpa引入 05:42 (了解)

|  |
| --- |
| Hibernate 介绍(了解)  1) 他是一个orm 框架  2) hibernate可以自动生成SQL语句 |
| JPA  Java Persistence API  说明: JPA 是一套接口和抽象类 不负责具体操作 |

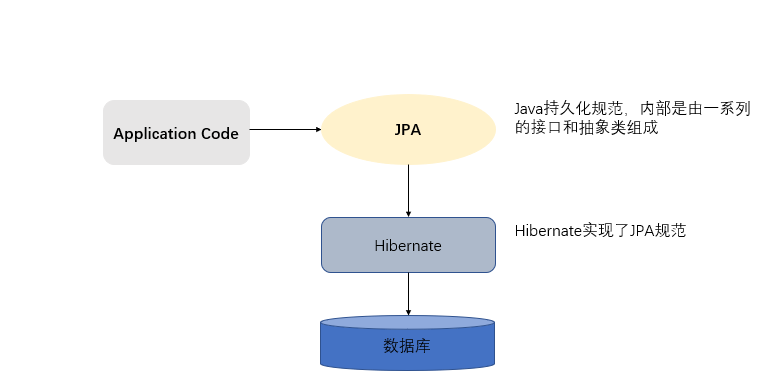
## 6) jpa介绍 07:03 (理解)

|  |
| --- |
| 回顾jdbc:  JAVA 制定了一套JDBC 的接口,给我们带来了一些好处:比如我们讲数据库由mysql 替换为Oracle的时候,我们的代码时不需要改变的,更改相关配置 ,极大的方便了我的操作 |
| JPA接口也一样,后续我们更换框架也不需要变更代码,只需要更换底层配置即可  我们用Java 定义的一套接口开发.当我们框架使用变更的时候就不用变更底层代码了 |
| 1. JPA 是一套规范,它不干活,干活的是实现了JPA这一套接口的框架 2. 我们掌握了JPA 的使用就相当于掌握了所有实现JPA的框架的使用 |
|  |

## 07)jpa的优势 03:33 (理解)

1. **简单方便**
2. 标准

掌握下图中之间的关系



## 08)案例说明 03:34(掌握)

|  |
| --- |
| 今天的案例整体是客户表的增删改查 |

## 09-入门案例：创建maven工程导入坐标 05:21(掌握)

|  |
| --- |
| 创建项目的步骤   1. 导入jar (Manven 到如配置文件即可) 2. 编写JPA 的核心配置文件 3. 编写客户的实体类 4. 配置映射关系 5. 保存客户信息到数据库中 |

## 10-入门案例：创建jpa的核心配置文件 07:49(掌握)

|  |
| --- |
| 1. Resources 目录下创建 META-INF/persistence.xml 2. 导入约束 见(下图) 3. 需要配置persistence-unit   Name : 单元名称  **transaction-type="RESOURCE\_LOCAL" 本地事务管理**  **transaction-type="JTA" 分布式事务管理 (了解)** |
|  |

## 11-入门案例：配置jpa的核心配置文件 08:35(掌握)



## 12-入门案例：编写客户的实体类 03:43(掌握)

|  |
| --- |
|  |

## 13-入门案例：使用jpa注解配置映射关系 09:13(掌握)

建立表与实体类之间的关系

@Entity : 声明实体类

@Table(name = **"cst\_customer"**) :配置表的名称

建立表字段与实体类属性之间的关系

@

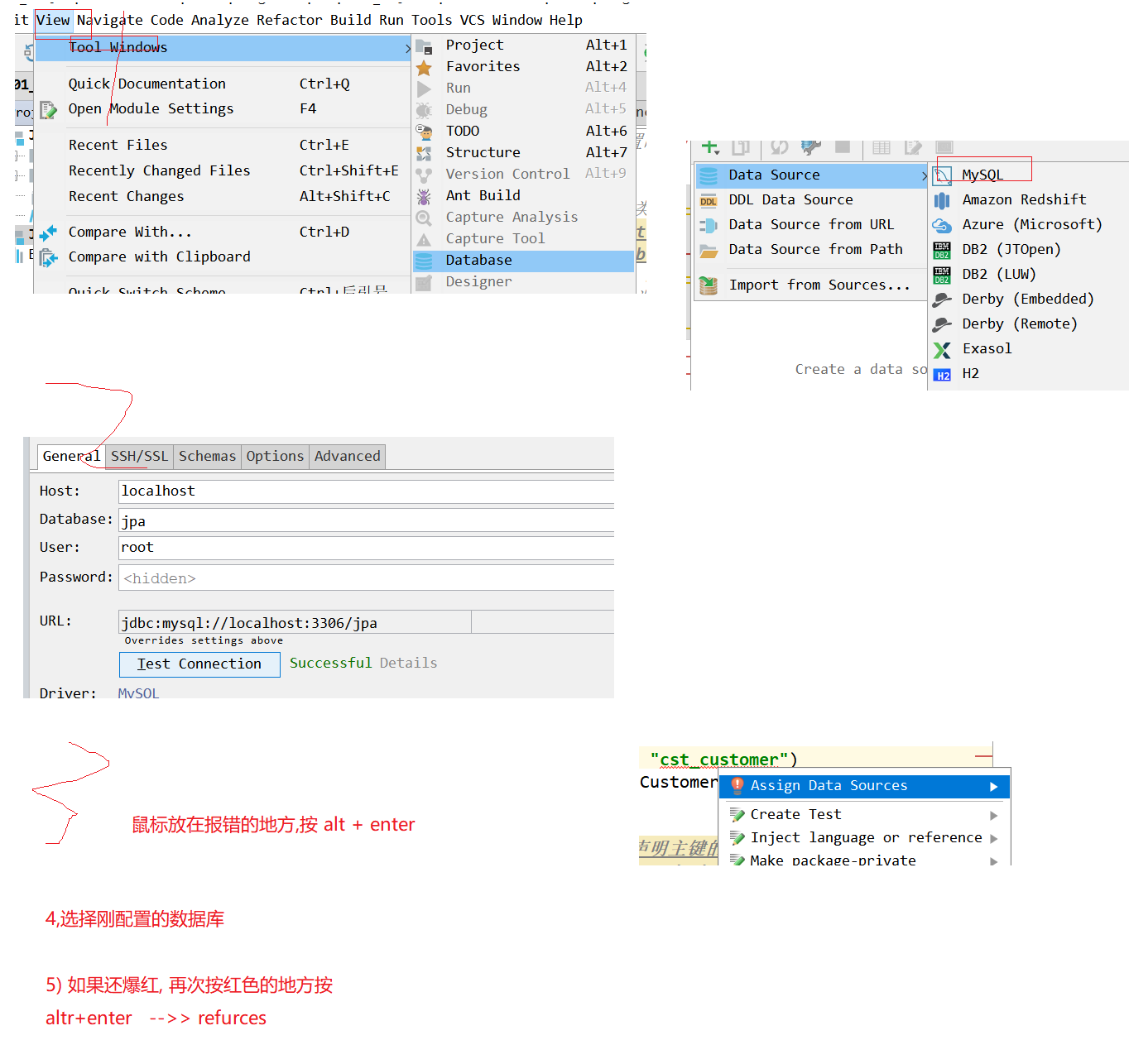
主键上的

@ID : 表示主键

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

|  |
| --- |
| GenerationType.***IDENTITY*** 主键由数据库自动生成（主要是自动增长型） |
|  |

===============红色处理方案



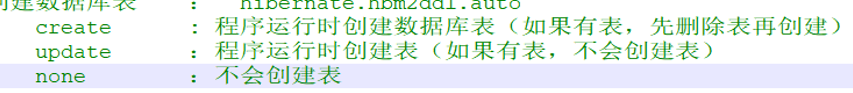
## 14-入门案例：jpa的操作步骤说明05:05(掌握)

|  |
| --- |
| //Jpa的操作步骤  //1.加载配置文件创建工厂（实体管理器工厂）对象  //2.通过实体管理器工厂获取实体管理器  //3.获取事务对象，开启事务  //4.完成增删改查操作  //5.提交事务（回滚事务）  //6.释放资源 |
|  |

## 15-入门案例：完成客户保存(掌握)

|  |
| --- |
|  |

## 16-入门案例：验证客户保存和自动生成数据库表的说明05:46(理解)



## 17-主键生成策略的说明06:26 (理解)

\* @GeneratedValue:配置主键的生成策略

\* strategy

\* GenerationType.IDENTITY ：自增，mysql

\* \* 底层数据库必须支持自动增长（底层数据库支持的自动增长方式，对id自增）

\* GenerationType.SEQUENCE : 序列，oracle

\* \* 底层数据库必须支持序列

\* GenerationType.TABLE : jpa提供的一种机制，通过一张数据库表的形式帮助我们完成主键自增

\* GenerationType.AUTO ： 由程序自动的帮助我们选择主键生成策略

Oracle

1. @Id
2. @GeneratedValue(strategy=GenerationType.SEQUENCE,generator="mseq")
3. @SequenceGenerator(name="mseq",sequenceName="metadata\_seq",allocationSize=1)
4. @Column(name = "ENTID", unique = **true**, nullable = **false**, precision = 22, scale = 0)
5. **public** **long** getEntid() {
6. **return** **this**.entid;
7. }

sequenceName 是自己在oracle中创建的序列。allocationSize要指定为1，否则它会按照默认50个数字增长。



## 18-主键生成策略的演示 04:36(掌握)

备注:

MySql 数据库AUTO 使用的是table;

## 19-api对象讲解 12:03(掌握)

|  |
| --- |
| 加载配置文件创建实体管理器工厂  Persisitence：静态方法（根据持久化单元名称创建实体管理器工厂）  createEntityMnagerFactory（持久化单元名称）  作用：创建实体管理器工厂    2.根据实体管理器工厂，创建实体管理器  EntityManagerFactory ：获取EntityManager对象  方法：createEntityManager  \* 内部维护的很多的内容  内部维护了数据库信息，  维护了缓存信息  维护了所有的实体管理器对象  再创建EntityManagerFactory的过程中会根据配置创建数据库表  \* EntityManagerFactory的创建过程比较浪费资源  特点：线程安全的对象  多个线程访问同一个EntityManagerFactory不会有线程安全问题  \* 如何解决EntityManagerFactory的创建过程浪费资源（耗时）的问题？  思路：创建一个公共的EntityManagerFactory的对象  \* 静态代码块的形式创建EntityManagerFactory    3.创建事务对象，开启事务  EntityManager对象：实体类管理器  getTransaction : 创建事务对象  presist ： 保存  merge ： 更新  remove ： 删除  find/getRefrence ： 根据id查询    Transaction 对象 ： 事务  begin：开启事务  commit：提交事务  rollback：回滚  4.增删改查操作  5.提交事务  6.释放资源 |

## 20-抽取jpaUtils工具类 06:47(掌握)

|  |
| --- |
|  |

## 21-验证工具类 02:54(掌握)

## 22-find方法：根据id查询客户 08:07(掌握)

|  |
| --- |
| 说明: 配置数据库表策略如果是Create 则会有问题  因为会重新生成数据库表, 查询就会没有数据,返回为空,解决方案为修改配置文件 |
| Customer customer = entityManager.find(Customer.class, 1l); |

## 23-延迟加载与立即加载 12:26(掌握)

Find(立即加载): 查询时请求数据库并返回对象本身

GetReference(..) 延迟加载

反回代理对象

,当我们调用对象的方法时(使用对象时)才会请求数据库

## 24-根据id删除客户 03:35(掌握)

先查询后删除:

底层是Hibernate 进行删除的,Hibernate 游离状态不能直接删除,所以需要先查询后删除

|  |
| --- |
| //ii 调用remove方法完成删除操作  entityManager.remove(customer); |

也可以自己创建一个对象,给主键赋值

## 25-客户的更新操作 03:58(掌握)

注意: 这里是根据主键字段全部更新其他字段

|  |
| --- |
| entityManager.merge(customer); |

// find/getReference

//find : 立即加载

Customer : new

getReference: 延迟加载

返回的是代理对象

Remove()

需要先查询后删除

Merge()

需要先查询后更新

## 26-jpql的介绍04:11(理解)

JPQL全称Java Persistence Query Language

JPQL JAVA提供的一套方便开发人员开发复杂查询的查询方法

## 27-jpql查询：查询全部 08:04(掌握)

1. 没有select \*
2. 从from 开始
3. 直接查询对象, 可以时对象类名称,也可以时包含包名的全限定类名 ,如果项目下有同名的对象,则需要使用全限定类名(包名.类名)

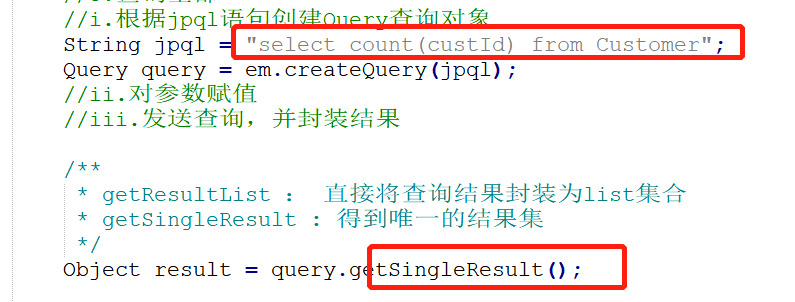
|  |
| --- |
|  |

## 28-jpql查询：倒序03:20(掌握)

1. 排序: order by 属性名称(不是字段名称)

|  |
| --- |
|  |

## 29-jpql查询：统计查询 05:47(掌握)



## 30-jpql查询：分页查询04:57(掌握)

Mysql: 分页查询 limit 0,2 “0,” 可以省略

query.setFirstResult(0);  
query.setMaxResults(2);

/\*\*

\* getResultList ： 直接将查询结果封装为list集合

\* getSingleResult : 得到唯一的结果集

\*/

List list = query.getResultList();

## 31-jpql查询：条件查询 04:52(掌握)

