2017.9.11

1.根据《深入实践springboot》建立了一个最基本的springboot框架的小程序，了解了springboot项目的搭建、运行和发布，springboot的主程序是一个入口，他的内容基本固定，用于启动springboot的程序，一些注解的介绍

### 2.springboot对数据库的访问站在了很多前人的基础上更进一步， 利用Spring Data JPA提供的接口，对数据库的访问大大简化，这里的很多知识来自于hibernate，所以先学习一下hibernate的一些知识，JPA是一个规范，他有很多实现，一般使用JpaRepository这个接口，可以实现很多查询功能，具体可以网上寻找相应的解释

### 3.hibernate是将面向对象编程的java和面向关系的数据库结合起来的东西，大大简化了操作，只需要在面向对象的角度就可以完成数据库的操作，而且hibernate提供了大量的资源库和配置，可以让开发者免于使用sql语句

### 4.hibernate的hbmddl可以自动建表，利用他的一些属性值的变化可以对表直接操作

5.使用hibernate的Annotation可以简化对实例的配置，直接在注解上操作。

2017.9.12

1.hibernate的一些基本配置

@Table（name=“表名”）当数据库表名与实体名不同时，用以指定实体对应的表；

@Column（name=”字段名”）当字段名与属性名不同时，用以指定字段的名称;

@Transient用以将某个属性排除出数据库存储，中文为透明意思；

@Temporal（TemporalType.DATE）用以指定日期精度

@Enumerated(EnumType.STRING)

2.注解可以写在字段上面也可以写在方法上，一般我们写在方法上。

3.Junit不报异常的解决办法，1）对@beforeclass的方法捕获异常2）再写一个main方法，里面只写@beforeclass

4.ID生成策略，uuid是一种几乎不会产生一致的id。

使用注解生成id的方法@GeneratedValue（strategy=generatype.AUTO）

5.联合主键：先建立一个联合主键类，里面的属性作为 类的主键

然后在主要类的前面（@Entity后面）注解@IdClass（“主键类名.class”）并且在属性上面注解@Id

2017.9.13

今日告知任务：打印任务（在收发文记录里，机要专员（jyzy角色）），参考合同部分的操作，在资产管理部里的编号回填可以看到打印按钮。

1.session的get和load方法：get（类文件名，主键名）可以取出数据库中的一条数据，而load方法可以生成一个代理对象，并没有立刻取出数据，有时出现的lazyinitialization的异常是由于这个产生的

2.update方法：更新detached的对象，更新完后会变成persistent的状态，更新的对象需要由主键，否则无法更新，update会全部更新记录的所有字段，

若需要指定更新部分字段

1）对字段加上@Column（updatable=false）可以将update语句中去除掉该字段的更新（不好的方法）

2）在相应的类的xml文件里可以加上dynamic-update=true，可以让更新自动处理，改了就更新，不改不更新

3一对一关联（很少出现，出现也使用外键关联）：

1）单向外键关联：比如husband和wife，在husband中有wife属性，在wife属性的get方法上注解@OneToOne，该注解的一些可以设置的条件@JoinColumn（name=“。。外键名称。。”）可以设置在husband中设置外键的名称

2）双向外键关联：在husband和wife里面各自加上对面的属性，并且都是用@ OneToOne注解，这是两张表会产生两个外键，这是冗余的信息，所以需要额外设置，可以在wife的类里加上（此处怀疑是否在huaband里加也会有同样的效果，未做测试）@ OneToOne（mappedBy=”wife”）可以告诉谁做了映射（Husband类里的wife属性上做了映射），此时，wife表里面就没有外键了。一般双向关联必设mappedBy

3）单向主键关联：同单向外键关联，只是将@ JoinColumn换成了PrimaryKey JoinColumn

4双向主键关联：同双向外键关联，只是将@ JoinColumn（mappedBy=”wife”）换成了PrimaryKey JoinColumn，这里不需要设置mappedBy=。。。因为主键关联，表只有一个主键

5）联合主键关联：单向外键关联，同非联合主键形式，需要修改@JoinColumns（{@JoinColumn（name=。。。，referencedClolumnName=。。。。），

@JoinColumn(name=...，referencedClolumnName=。。。。)}）

6.）组件映射，对于两张一对一的表，可以将他们在数据库里合成一张表，可以在husband类里的wife属性前加入注解@Embedded，此时应当让husband和wife类的属性名都不一样，或者@Column修改表里的字段名

4.多对一关联

1）多对一单向关联：在多方加外键，就类似于一个组有很多人，那么关联的方法是在人的属性上加一个组名，而不是在组里面加人名（意思是一个组会被分成多条记录，每条记录的任命不一样，组名相同）此时使用注解@ManyToOne会自动在One的表格里添加Many\_Id的字段

2）一对多单向关联：在一方建立set<多方>去存储数据，，使用@OneToMany注解会生成中间表，被认为是多对多，此时需要在下面注解@JoinColumn（name=“。。。”）

！！！无论是一对多还是多对一，这两种注解在数据库里生成的表是一样的，都只会在多方加外键，所以一对多的在一方（Group）下面给set<User>加的注解@JoinColumn（name=“groupId”）中的字段名是groupId（相对于User和group来说）

2017.9.14

3）一对多双向关联（等于多对一双向关联）：就是在Group和User里面分别设置@OneToMany和@ManyToOne，此时不需要给@OneToMany设置@JoinColumn，否则会产生冗余，但是需要在@OneToMany设置（mappedBy=group）来确立映射主体，这里的group是User里的group属性

4）多对多单向关联：比如学生和老师，单向关联是指老师知道要教多少学生，所以老师对象里面有一个set<Student>，而学生不需要知道老师是谁，没有老师的相关信息，此时应该在set,<Student>上注解@ManyToMany那么会生成一张关系表Teacher\_Student,里面会有两个字段Teacher\_id和students\_id,这里的students是Teacher的一个属性名，若要自定义这个表的一些名字，可以用

@JoinTable（name=””，

joinColumns{@JoinColumn（name=“teacher\_id“）},

inverseJoinColumns={@JoinColumn（name=“student\_id“）}

）

这里有JoinColumns{@JoinColumn（name=“teacher\_id“）}指定了关系表里参考了该对象id（代码所在的对象）的外键名称，而inverse则是说明了teacher对应的另一张表的一些情况

5）多对多双向关联：学生知道老师，老师也知道学生（一般情况下双向是很少见的）

这时候就需要在多对多单向的基础上在另一个类中也建立对应的set属性，这里是在Student类里面建立set<Teacher>，并且注解上@ManyToMany（mappedBy=“students“）这里的student是指Teacher类里的Set<Student> students属性

5.增删改查（CRUD）

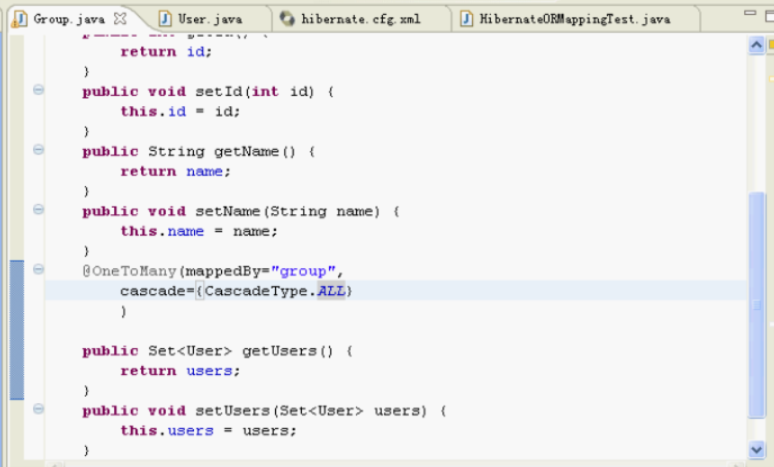
1）如果两张表是关联的，这时候需要设置cascade=(CascadeType.All)这里CascadeType是一个枚举类型，User和Group是ManyToOne的关系，这时候在User里面的@ManyToOne后面设置cascade=(CascadeType.All)，可以在只保存User的时候，把user关联导向的group也会在Group表里保存进去，这就是cascade的作用

如果是@ManyToOne

2）在一对多双向关联中，想在存group时，顺带存user，由于group\_id是由user里的group属性来mappedBy的，所以需要先设置user里的setGroup，否则user 的grpup\_id为null.

这里需要加上这两条，因为group\_id是由user里的group属性来mappedBy的，所以不这么设User表里的group\_id会变成空值





3）双向关联规律：双向关系需要设定双向关联

需要设置mappedBy

4）一对多双向关联关系中，如果取出user（多的一方），会默认取出对应的group

但是取group（一的一方），不会顺带取出多的一方，

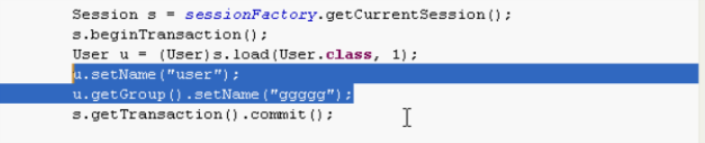
5）注cascade不会影响读取，而是fetch来管，cascade只管增删改，而fetch管查

6）fetch由lazy和eager两种属性，在@ManyToone默认为eager，而@OneTomany默认时lazy

2017.9.15

7）lazy情况下，与你所读取的对象相关联的另一张表不会被读取，但是在eager情况下，则会读取。

8）一对多双向关联时，两边不要都设成eager，默认是两边不一样，而一般都用默认的，从多那边读取时顺便也把一读出来

9）更新的操作时，可以先load这个记录，在使用set方法修改其中的属性，然后commit可以仅从更新

或者使用update方法进行更新，update要对detached状态下的对象进行操作，这里面的对象取出要用get方法

注意：Cascade的几个值分别对应几个方法，当你调用这些方法是，如果设定了相应的值，就可以级联的处理相应的表格对象，不过一般直接设ALL就行了，all可以让你的所有操作都获得级联效果，其中merge操作等于save加update，refresh操作是A里面需要读取B改过的数据。

10）删除操作：delete，由于级联，如果所有数据都级联（All），会导致把所有数据都删除，所以删除时需要先把关联关系删除，



或者使用hql语言删除

6．集合映射

如果对象里面存在一个集合是可排序的，比如List，那时可以加上注解@OrderBy

如果对象实体里面由Map类型，可以由注解设置map的key的值

2017.09．19

继承映射三种方式（以student、teacher继承person类）

Singletable：放在一张表里存储

需要在person类前面加上注释

@Inheritance(strategy=InheritanceType.SINGLE\_TABLE)

@DiscriminatorColumn(name="discriminator",discriminatorType=DiscriminatorType.STRING)

@DiscriminatorValue("person")

然后分别在teacher和student类上面加上注释

@DiscriminatorValue("teacher")

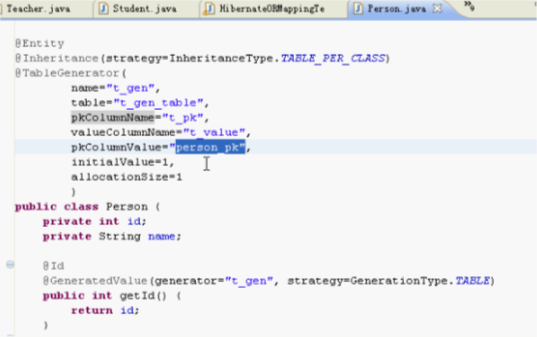
和

@DiscriminatorValue("student")

就可以实现单表映射

Tablepercalss：每个类一张表

此时需要两张表的id不能重复，所以id生成策略需要变成表生成方式



这种存储方式需要在person类前加上注释

@Inheritance(strategy=InheritanceType. TABLE\_PER\_CLASS)

这种存储方式不需要student和teacher里面加上id和name等共有属性

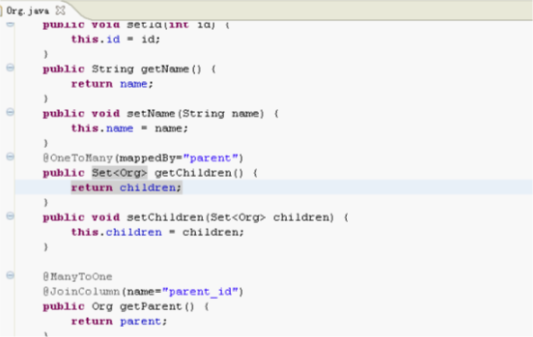
Jpined方式去存储：

这种存储方式需要在person类前加上注释

@Inheritance(strategy=InheritanceType.JOINED)

这种存储方式，teacher和student类里面就不想要id，name等共有属性，只留下独有的属性就可以了。

树状映射



事物隔离机制：

事务的四大特性（ACID）:

原子性

一致性

独立性

持久性

悲观锁与乐观锁：

悲观锁是用session.load(Animal.class,1,锁的级别)对这次访问加锁，等同于

Select \*\*\* for update;

乐观锁是认为的在表里面加个Version，访问前读取，访问后会修改，如果访问前读取的和更新时不一样，则这次操作不执行。

20170921

Hibernate的一级缓存二级缓存

一级缓存是指session级别的缓存，这级缓存只能在一个session里共用，不能在两个不同的session里使用。

配置二级缓存：

需要加入相应的cache的xml文件并导入相应的cache的jar文件，并在需要使用二级缓存的类名前加入注解，

@Cache（

usage=CacheConcurrencyStrategy.READ\_WRITE

region = “自定义的缓存策略，在cache的xml文件里配置”

）

Load和iterate默认使用二级缓存

List会默认往二级缓存里存加数据，但是查询时候不使用

如果使用query用二级缓存，则需要查询缓存

查询缓存：

这是依赖于二级缓存的一级，所以要在配置完二级缓存的基础上再设置一个缓存配置，在访问时加上session.creatQuery(“from 表名“).setCacheable(true).list();