山东大学 计算机科学与技术 学院

现代软件开发技术 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：2020001300 | 姓名： | | 班级： |
| 实验题目： Spring Boot Data JPA（2） | | | |
| 实验学时：2学时 | | 实验日期： 2023-11-14 | |
| 实验目的：  1.能使用Spring Data JPA 进行数据库关联操作（双类/双表）  2.掌握实体类中一对一、一对多关联的声明 | | | |
| 硬件环境： | | | |
| 软件环境：  操作系统：windows 10  开发平台与工具：IDEA开发平台、jdk-11.0.1、apache-tomcat-9.0.37 | | | |
| 实验内容与问题回答：  1.练习使用Spring Boot JPA实现数据库一对一关联查询（人员person和身份证idcard）  用STS4打开课本项目ch6\_2。  （1）分析持久化实体类  打开src/main/java下的com.ch.ch6\_2.entity包中的Person.java和IdCard.java实体类，分析其中的代码，  Q:在两个类中，注解@Table 创建的表名分别是什么？  在两个类中，注解@Table创建的表名分别是：person\_table和idcard\_table  Q:在两个类中，注解@OneToOne中的属性分别有哪些，意义是什么？  在第一个类Person类中@OneToOne 注解用于建立 Person 实体类与 IdCard 实体类之间的一对一关系。以下是 @OneToOne 注解中使用的属性及其意义：  **optional = true：**  意义： 表示这个一对一关系是可选的，即 Person 实体类的 idCard 属性可以为 null。如果不设置 optional，默认情况下一对一关系是必须存在的，即每个 Person 必须关联一个 IdCard。  **fetch = FetchType.LAZY：**  意义： 定义了加载策略。FetchType.LAZY 表示延迟加载，即在访问 idCard 属性时才会实际加载 IdCard 对象。相反，FetchType.EAGER 表示立即加载，即在加载 Person 对象时就同时加载关联的 IdCard 对象。  Q:注解@JoinColumn 在哪个类中？其中"id\_Card\_id"的含义？  注解@JoinColumn在Person类中，  在该类中，"id\_Card\_id" 是通过 @JoinColumn 注解指定的外键列名，用于指明 Person 对应表的外键列与关联的 IdCard 对应表的主键列进行关联。@JoinColumn(name = "id\_Card\_id")：name = "id\_Card\_id"：指定了在数据库表中创建的外键列的名称为 "id\_Card\_id"。这表示在 Person 表中，会有一个名为 "id\_Card\_id" 的外键列，用于关联到 IdCard 表的主键列。  Q:在两个类中，哪个是维护类？哪个是被维护类？关联的外键定义在哪个类？相互关联的成员属性分别是哪个？  上述两个类中，Person类是维护类，IdCard类是被维护类，关联的外键定义在了Person类中；  在Person类中互相关联的成员属性是idCard属性，在IdCard类中关联的成员属性是person属性。  （2）数据访问层  打开src/main/java下的com.ch.ch6\_2.repository包中的PersonRepository.java和IdCardRepository.java，分析其代码，  Q:在IdCardRepository.java中，哪个方法实现了关联查询？该方法名中Person\_id代表的含义？在IdCardRepository.java的代码中，以下方法实现了关联查public IdCard findByPerson\_id(Integer id);方法名中的 findByPerson\_id 表示通过关联属性 person 的 id 属性来进行查询。具体含义如下findByPerson\_id：findBy 表示是一个查询方法。Person 是 IdCard 类中关联的属性。  \_id 表示关联属性 person 的 id 属性。  Q:在PersonRepository.java中，哪个方法实现了关联查询？该方法名中IdCard\_id代表的含义？  public Person findByIdCard\_id(Integer id);  IdCard\_id代表卡id  （3）业务服务层  打开src/main/java下的com.ch.ch6\_2.service包中的PersonAndIdCardService.java和PersonAndIdCardServiceImpl.java，分析其代码,  Q:PersonAndIdCardService.java中，为什么该接口不用像数据访问层中的接口那样 extends JpaRepository ？  JpaRepository 接口提供了一些基本的数据操作方法，例如根据ID查询、保存实体等，它是Spring Data JPA提供的用于简化数据访问层的接口。而在服务层，可能会定义更高层次的业务逻辑，需要多个数据操作的组合，因此不必直接继承 JpaRepository。  Q:PersonAndIdCardServiceImpl.java中，saveAll()方法分别创建了实体类的几个实例？  在上述代码中，saveAll() 方法分别创建了以下实体类的实例：  IdCard 类的实例：  IdCard ic1  IdCard ic2  IdCard ic3  Person 类的实例：  Person p1  Person p2  Person p3  Q: 为何要先建IdCard 的对象，再建Person的对象？  在saveAll()方法中，先创建 IdCard 的对象，然后再创建 Person 的对象，是因为在这个数据模型中，Person 和 IdCard 之间存在一对一的关系，而且 Person 类中包含了一个对 IdCard 的引用。  在关系型数据库中，通常有两种建模方式来表示一对一关系：  1.共享主键（Shared Primary Key）： Person 和 IdCard 共享同一个主键，即 Person 的主键同时也是 IdCard 的主键。  2.外键关联（Foreign Key Association）： Person 持有一个指向 IdCard 的外键。  在这个例子中，采用的是外键关联的方式，即 Person 类中有一个属性 idCard，它是对 IdCard 对象的引用。为了建立关联关系，需要先创建 IdCard 对象，然后将这个对象赋给 Person 的 idCard 属性。这样，在保存 Person 对象时，JPA 会自动处理关联关系，并在数据库中保存相关的外键关联。  如果采用共享主键的方式，那么在保存 Person 对象时，IdCard 对象的主键也会自动被设置为相同的值。这两种方式各有优劣，选择哪种方式通常取决于具体的业务需求和数据模型设计。在这里，通过先创建 IdCard 对象再创建 Person 对象，更符合外键关联的场景。  Q：两个实体类间的关联是用什么语句建立的？其中有对id\_Card\_id的赋值语句吗？  两个实体类之间的关联是通过 @OneToOne 注解和 @JoinColumn 注解来建立的。  Q:关联（级联）查询是通过什么方法实现的？  关联（级联）查询是通过 JPA（Java Persistence API）中的关联属性和查询方法实现的。在 JPA 中，通过配置实体类之间的关联关系，可以指定在执行某个实体查询时是否要同时加载关联的实体数据，从而实现关联查询。  Q:为何服务层提供的方法比数据访问层的多？在实现时，调用注入的数据访问层对象却都能实现?  服务层提供的方法比数据访问层的方法多的原因有几个方面：  业务逻辑的封装： 服务层的方法通常封装了底层数据访问层的多个方法，完成了一个高层次的业务功能。一个服务层方法可能需要调用多个数据访问层方法来满足业务需求。  事务管理： 服务层通常负责事务的管理，确保业务操作的一致性和完整性。一个服务层方法可能包含多个数据访问层的方法，而这些方法需要在同一个事务中执行。  业务层的多样性： 不同的业务功能需要不同的业务逻辑，因此服务层提供的方法数量会根据业务需求的多样性而增多。  在实现时，服务层调用注入的数据访问层对象是为了实现具体的业务功能。通过依赖注入，服务层可以使用数据访问层提供的持久化操作，将底层的数据库交互从业务逻辑中解耦，提高代码的可维护性和灵活性。这种分层的结构有助于代码的组织、维护和测试。  （4）控制器类  打开src/main/java下的com.ch.ch6\_2.controller包中的TestOneToOneController.java，分析其中的代码。  特别注意几个注解的作用。  Q: 注解@Autowired ，注入的是？  在TestOneToOneController.java文件的代码中，注解 @Autowired 用于依赖注入（Dependency Injection），而被注入的是 PersonAndIdCardService 类型的对象。这意味着通过 Spring 的依赖注入机制，容器会在启动时实例化 PersonAndIdCardService 类，并将其注入到 TestOneToOneController 控制器中，以便在控制器中调用 PersonAndIdCardService 提供的服务方法。  Q:@RequestMapping 对应的查询方法中的形参是如何传入的？  public IdCard findByPerson\_id(Integer id) {  在上面这个方法中@RequestMapping("/findByPerson\_id") 注解表示当请求的URI是 "/findByPerson\_id" 时，会调用下面的方法处理该请求。方法的参数 Integer id 是通过URL中的路径变量传入的。在这里，Spring MVC 会从请求的URI中解析出路径变量，并将其转换为方法参数。  例如，如果请求的URI是 "/findByPerson\_id/123"，那么Spring MVC会将 "123" 解析为整数，并传递给 findByPerson\_id 方法的 id 参数。  需要注意的是，路径变量的名称（这里是 "123"）需要与方法参数的名称一致。Spring MVC 使用这种方式来匹配路径变量和方法参数。  （5）运行  运行src/main/java下com.ch.ch6\_2包中Ch62Application.java(run as Java Application)，  （a）在浏览器地址栏输入：http://localhost:8080/ch6\_2/save/，保存数据（创建表）。  用外部可视化工具(如：Navicat for MySQL）打开MySQL数据库中创建的表：person\_table和idcard\_table，    Q：其中的属性都有哪些？记录有几条？外键的值是否都已填入？为何表中没有实体类中声明的另一个类的引用属性？      （b）在浏览器地址栏输入：http://localhost:8080/ch6\_2/findByIdCard\_id?id=1，关联查询身份证id为1的人员信息。    （c）在浏览器地址栏输入：http://localhost:8080/ch6\_2/findByPerson\_id?id=1，关联查询人员id为1的身份证信息。    Q:用工具（如Navicat）连接打开数据库，查看库中与person、idcard有关的表有几个？关联属性在哪个表中？  为何被维护类中声明的关联属性没有出现在表中？  与person、idcard有关的表有2个分别是person\_table和idcard\_table这两个表；  关联属性id\_card\_id在person\_table这个表中；      为什么被维护类中声明的关联属性没有出现在表中呢？  因为在 JPA（Java Persistence API）中，关系维护端（被维护端）和关系被维护端（维护端）在数据库中的表结构设计上有所不同。关系被维护端的属性通常不会在表中生成相应的列，而是通过外键关联的方式体现在关系维护端的表中。  2.练习使用Spring Boot JPA实现两个类之间一对多和多对一数据库关联查询(作者Author和文章Article）  用STS4打开课本项目ch6\_2。  （1）分析持久化实体类  打开src/main/java下的com.ch.ch6\_2.entity包中的Author.java和Article.java实体类，分析其中的代码，  Q:注解@Table 创建的表名是什么？  在Author.java文件中创建的表名是author\_table;  在Article.java文件中创建的表名是article\_table；  Q:注解@OneToMany和@ManyToOne注解中的属性与之前的@OneToOne有什么不同？谁是多方？谁是一方？  @OneToMany：表示一对多关系，即一个实体可以关联多个其他实体，但每个其他实体只能关联一个当前实体。  @ManyToOne：表示多对一关系，即多个实体可以关联到同一个实体。  @OneToOne：表示一对一关系，即一个实体只能关联一个其他实体，反之亦然。  @OneToOne：可以使用 mappedBy 属性，也可以不使用，具体取决于关系的设计。  Q:注解 @JoinColumn 表达的含义？  在Article.java文件中该注释表达的含义是设置在article表中的关联字。  Q:两个实体类中哪个是维护类，哪个是被维护类？关联的外键定义在哪个类？相互关联的成员属性是哪个？  在这两个实体类中，Author类是维护类，Article类是被维护类；  关联的外键定义在Article类中，通过@JoinColumn注解定义了”id\_author\_id”作为外键。  相互关联的成员属性：  Author 类中的关联属性是 articleList。\*\*这是一个列表，表示一个作者可能拥有多篇文章。  Article 类中的关联属性是 author。\*\*这是一个作者对象，表示一篇文章属于一个作者。  （2）数据访问层  打开src/main/java下的com.ch.ch6\_2.repository包中的AuthorRepository.java和ArticleRepository.java，分析其代码，  Q:其中哪个方法实现了关联查询？  **在AuthorRepository.java中**：    这方法使用了 @Query 注解，通过 JPQL（Java Persistence Query Language）语句实现了关联查询。 JPQL语句中使用 inner join a.articleList t 部分表示与 Author 类的 articleList 属性进行内连接，然后通过 where t.title like %?1% 部分过滤出符合条件的结果。这样，该方法实现了根据文章标题包含的内容查询作者的关联查询。  **在ArticleRepository.java中：**    这方法使用了 Spring Data JPA 的命名查询规则，根据方法名自动生成查询语句。方法名中的 findByAuthor\_id 表示根据 Article 类中的 author 属性的 id 进行查询。因此，该方法实现了根据作者id查询文章信息的关联查询。  （3）业务服务层  打开src/main/java下的com.ch.ch6\_2.service包中AuthorAndArticleService.java和AuthorAndArticleServiceImpl.java，分析代码。  Q:其中saveAll()方法分别创建了实体类的几个实例？先存的谁？why？两个实体类间的关联式怎样建立的？其中有对外键赋值指令吗？  该方法saveAll()创建了两个 Author 实例：  Author a1，姓名为 "陈恒1"  Author a2，姓名为 "陈恒2"  创建了四个 Article 实例：  Article at1，标题为 "JPA的一对多111"，内容为 "其实一对多映射关系很常见111"，关联到 Author a1  Article at2，标题为 "JPA的一对多222"，内容为 "其实一对多映射关系很常见222"，关联到 Author a1  Article at3，标题为 "JPA的一对多333"，内容为 "其实一对多映射关系很常见333"，关联到 Author a2  Article at4，标题为 "JPA的一对多444"，内容为 "其实一对多映射关系很常见444"，关联到 Author a2  其中，先存储了 Author 实例，然后存储了 Article 实例。这是因为在这个关系模型中，Author 是一的一端，而 Article 是多的一端。通常情况下，我们先保存一的一端，然后再保存多的一端。  Q:其中的级联查询是通过什么方法实现的？  在该项目中，级联查询是通过AuthorRepository接口中的自定义查询方法实现的，具体是findByArticleList\_titleContaining方法和findAuthorByArticleListtitleContaining方法。  **findByArticleList\_titleContaining方法：**  该方法使用Spring Data JPA的方法命名规则，通过作者的文章标题包含的内容来进行关联查询。  相当于JPQL语句：select a from Author a inner join a.articleList t where t.title like %?1%  **findAuthorByArticleListtitleContaining方法：**  该方法使用@Query注解，通过JPQL语句来进行关联查询。相当于JPQL语句：select a from Author a inner join a.articleList t where t.title like %?1%  （4）控制器类  打开src/main/java下的com.ch.ch6\_2.controller包中的TestOneToManyController.java，分析其中的代码，  Q:注解@Autowired 注入的是？  在上述代码中，@Autowired 注解注入的是 AuthorAndArticleService 接口的实现类，即 AuthorAndArticleServiceImpl。  Q:哪几个请求映射对应的是级联查询？  在这个Controler类中，以下请求映射对应的是级联查询：  findByArticleList\_titleContaining  findAuthorByArticleListtitleContaining  （5）运行  重新运行src/main/java下com.ch.ch6\_2包中Ch62Application.java(run as Java Application)，  （a）在浏览器地址栏输入：http://localhost:8080/ch6\_2/saveOneToMany/，保存数据（创建表）。    用外部可视化工具(如：Navicat for MySQL）打开MySQL数据库中创建的表：author\_table和article\_table，  Q：其中的属性都有哪些？记录有几条？外键的值是否都已填入？  Author\_table表中有下面两个属性，记录有两条；    在Article表中属性有以下四个，记录有四条，外键的值都已经填入。    （b）输入：http://localhost:8080/ch6\_2/findArticleByAuthor\_id?id=2，关联查询作者id为2（id的值需要依据实际数据库中的值）的文章列表。    （c）输入：http://localhost:8080/ch6\_2/findArticleByAuthor\_aname?aname=陈恒1，关联查询作者名为陈恒1的文章列表。    （d）输入http://localhost:8080/ch6\_2/findByArticleList\_titleContaining?title=对多1，关联查询文章标题含“对多1”的作者。    Q:用工具（如Navicat）连接打开数据库，查看库中与author、article有关的表有几个？关联属性在哪个表中？  数据库中和author和article有关的表有2个，分别是author\_table和article\_table这两个表，关联属性id\_author\_id在article\_table中。 | | | |
| 思考题解答：  （1）如果用SQL描述 person和身份证idcard 两个表一对一的关联，命令应该是怎样的？  创建idcard表:  CREATE TABLE idcard (  id INT PRIMARY KEY,  code VARCHAR(20),  );  创建person表  CREATE TABLE person (  id INT PRIMARY KEY,  pname VARCHAR(50),  psex VARCHAR(10),  page INT,  id\_card\_id INT UNIQUE, );  （2）如果用SQL描述 Author和文章Article 两个表一对多的关联，命令应该是怎样的？  创建Author表:  CREATE TABLE author (  id INT PRIMARY KEY,  aname VARCHAR(50),  );  创建Article表  CREATE TABLE article (  id INT PRIMARY KEY,  title VARCHAR(100),  content TEXT,  author\_id INT,  FOREIGN KEY (author\_id)  );  审视以上所写的两个SQL命令，从命令上能区分出谁是一对一、谁是一对多的关联吗？  从上面的两个SQL命令中，我们可以通过外键的设置来区分出一对一和一对多的关联。  一对一关联（例如，Person和IdCard）：在这种情况下，Person表的主键（例如，id列）通常是 IdCard 表的外键。这意味着每个 Person 对象都关联一个唯一的 IdCard 对象，而每个 IdCard 对象也仅关联一个唯一的 Person 对象。一对一关联通常使用相同的主键值。 | | | |