# maven pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.louie.learning.springboot</groupId>  
 <artifactId>springboot-jpa</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>springboot-jpa</name>  
 <description>Demo project for Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.5.6.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
  
</project>

# 实体Student

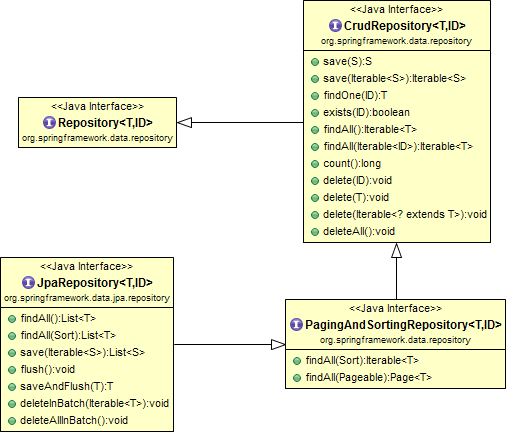
@Entity  
@Table(name="t\_stu")  
public class Student {  
 @Id()  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private int id;  
 private String name;  
 private String address;  
 private int age;  
  
 public Student() {  
 }  
  
 public Student(String name, String address, int age) {  
 this.name = name;  
 this.address = address;  
 this.age = 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 String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
}

# application.properties

#开启包的自动扫描  
entitymanager.packagesToScan= com.louie.learning.springboot.model  
# 数据库连接  
spring.datasource.url=jdbc:h2:file:D:/WorkSpaceIDEA/springboot-learning-demo/db/demo  
# 用户名  
spring.datasource.username=admin  
# 密码  
spring.datasource.password=admin  
# 数据库驱动  
spring.datasource.driver-class-name=org.h2.Driver  
# 自动更新表  
spring.jpa.properties.hibernate.hbm2ddl.auto=update  
# 使用MYSQL5作为数据库访问方言  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect  
#显示sql语句  
spring.jpa.properties.hibernate.show\_sql=true  
  
#h2 web consloe配置  
#进行该配置后，h2 web consloe就可以在远程访问了。否则只能在本机访问。  
spring.h2.console.settings.web-allow-others=true  
#进行该配置，你就可以通过YOUR\_URL/h2-console访问h2 web consloe。YOUR\_URL是你程序的访问URl。  
spring.h2.console.path=/h2-console  
#进行该配置，程序开启时就会启动h2 web consloe。当然这是默认的，如果你不想在启动程序时启动h2 web consloe，那么就设置为false。  
spring.h2.console.enabled=true

# Dao接口

首先看一下Spring Data JPA中的几个接口



最高层的Repository<T,ID>是一个空接口，我们定义的数据访问类只要实现这个接口，这个数据访问类就可以被spring data所管理，就此可以使用spring为我们提供操作方法(在原来的spring data中我们需要配置很多和Spring Data Repository相关的设置，但是现在有了spring boot，全部都已经自动配置好了)。这个接口要实现有两个泛型参数，第一个T表示实体类，第二个表示主键的类型，我们写一个数据库访问接口。

public interface StudentDao extends Repository<Student, Integer> {  
 @Query("select s from Student s where s.id=?1")  
 Student loadById(int id);  
  
 //根据地址和年龄进行查询  
 List<Student> findByAddressAndAge(String address, int age);  
  
 //根据id获取对象，即可返回对象，也可以返回列表  
 Student readById(int id);  
  
 //根据id获取列表，这里如果确定只有一个对象，也可以返回对象  
 List<Student> getById(int id);  
  
 //根据id获取一个对象，同样也可以返回列表  
 Student findById(int id);  
}

这个接口实现了Repository接口，我们定义了两个方法，这两个方法代表Repository使用的一种基本方法，第一个方法增加了一个Query的annotation，通过这个声明，Spring Data JPA就知道该使用什么HQL去查询数据，?1表示用方法中的第一个参数。第二个函数我们并没有定义任何的Annotation，但是它也可以查询得出来，在Spring Data JPA中提供了一种衍生查询，只要函数的声明有findBy,getBy,readBy,他就会去读取,findByAddressAnAge表示根据address和age进行查询，方法的第一个参数就是address，第二个参数就是age，readByXX,getByXX都是一样的道理，这些方法的返回值可以是一个列表，也可以是一个对象，spring Data JPA会自动根据返回类型来进行处理。我们不用写实现类，Spring Data JPA会自动帮助我们实现查询。很多时候在项目中会用到这些简单的查询，但是不得不写个方法来实现，但是现在使用了Spring Data JPA之后，这个操作被完全简化了。

# 测试类

@RunWith(SpringRunner.class)  
@SpringBootTest  
public class SpringbootJpaApplicationTests {  
  
 @Autowired  
 private StudentDao studentDao;   
  
 @Test  
 public void testStudent() {  
 Assert.*assertEquals*("foo", studentDao.findById(1).getName());  
 Assert.*assertEquals*("foo", studentDao.readById(1).getName());  
 Assert.*assertEquals*(1, studentDao.getById(1).size());  
 Assert.*assertEquals*("foo", studentDao.loadById(1).getName());  
 Assert.*assertEquals*(2, studentDao.findByAddressAndAge("zt", 22).size());  
 }

}