# jpa-study  
学过了数据库自动生成代码的mp，来使用一下代码生成数据库的jpa  
  
\* 什么是jpa呢？  
  
JPA顾名思义就是Java Persistence API的意思，  
是JDK 5.0注解或XML描述对象－关系表的映射关系，  
并将运行期的实体对象持久化到数据库中。  
  
\* jpa具有什么优势？  
  
2.1标准化  
JPA 是 JCP 组织发布的 Java EE 标准之一，  
因此任何声称符合 JPA 标准的框架都遵循同样的架构，  
提供相同的访问API，  
这保证了基于JPA开发的企业应用能够经过少量的修改就能够在不同的JPA框架下运行。  
  
  
2.2容器级特性的支持  
JPA框架中支持大数据集、事务、并发等容器级事务，  
这使得 JPA 超越了简单持久化框架的局限，  
在企业应用发挥更大的作用。  
  
  
2.3简单方便  
JPA的主要目标之一就是提供更加简单的编程模型：  
在JPA框架下创建实体和创建Java 类一样简单，  
没有任何的约束和限制，只需要使用 javax.persistence.Entity进行注释，  
JPA的框架和接口也都非常简单，没有太多特别的规则和设计模式的要求，开发者可以很容易的掌握。JPA基于非侵入式原则设计，因此可以很容易的和其它框架或者容器集成。  
  
2.4查询能力  
JPA的查询语言是面向对象而非面向数据库的，它以面向对象的自然语法构造查询语句，可以看成是Hibernate HQL的等价物。JPA定义了独特的JPQL（Java Persistence Query Language），JPQL是EJB QL的一种扩展，它是针对实体的一种查询语言，操作对象是实体，而不是关系数据库的表，而且能够支持批量更新和修改、JOIN、GROUP BY、HAVING 等通常只有 SQL 才能够提供的高级查询特性，甚至还能够支持子查询。  
  
2.5高级特性  
JPA 中能够支持面向对象的高级特性，如类之间的继承、多态和类之间的复杂关系，这样的支持能够让开发者最大限度的使用面向对象的模型设计企业应用，而不需要自行处理这些特性在关系数据库的持久化。  
  
注解 解释  
@Entity 声明类为实体或表。  
  
@Table 声明表名。  
  
@Basic 指定非约束明确的各个字段。  
  
@Embedded 指定类或它的值是一个可嵌入的类的实例的实体的属性。  
  
@Id 指定的类的属性，用于识别（一个表中的主键）。  
  
@GeneratedValue 主键生成策略，即自层，  
  
@Transient 指定的属性，它是不持久的，即：该值永远不会存储在数据库中。  
  
@Column 指定持久属性栏属性。  
  
@SequenceGenerator 指定在@GeneratedValue注解中指定的属性的值。它创建了一个序列。  
  
@TableGenerator 指定在@GeneratedValue批注指定属性的值发生器。它创造了的值生成的表。  
  
@AccessType 这种类型的注释用于设置访问类型。如果设置@AccessType（FIELD），则可以直接访问变量并且不需要getter和setter，但必须为public。如果设置@AccessType（PROPERTY），通过getter和setter方法访问Entity的变量。  
  
@JoinColumn 指定一个实体组织或实体的集合。这是用在多对一和一对多关联。  
  
@UniqueConstraint 指定的字段和用于主要或辅助表的唯一约束。  
  
@ColumnResult 参考使用select子句的SQL查询中的列名。  
  
@ManyToMany 定义了连接表之间的多对多一对多的关系。  
  
@ManyToOne 定义了连接表之间的多对一的关系。  
  
@OneToMany 定义了连接表之间存在一个一对多的关系。  
  
@OneToOne 定义了连接表之间有一个一对一的关系。  
  
@NamedQueries 指定命名查询的列表。  
  
@NamedQuery 指定使用静态名称的查询  
   
 jpa.hibernate.ddl-auto是hibernate的配置属性，其主要作用是：  
  
 自动创建、更新、验证数据库表结构。该参数的几种配置如下：  
   
 ·create：每次加载hibernate时都会删除上一次的生成的表，然后根据你的model类再重新来生成新表，哪怕两次没有任何改变也要这样执行，这就是导致数据库表数据丢失的一个重要原因。  
 ·create-drop：每次加载hibernate时根据model类生成表，但是sessionFactory一关闭,表就自动删除。  
 ·update：最常用的属性，第一次加载hibernate时根据model类会自动建立起表的结构（前提是先建立好数据库），以后加载hibernate时根据model类自动更新表结构，即使表结构改变了但表中的行仍然存在不会删除以前的行。要注意的是当部署到服务器后，表结构是不会被马上建立起来的，是要等应用第一次运行起来后才会。  
 ·validate：每次加载hibernate时，验证创建数据库表结构，只会和数据库中的表进行比较，不会创建新表，但是会插入新值。

使用jpa：

1. 引入依赖：

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>

1. 写配置类

spring:  
 datasource:  
 username: root  
 password: 123456  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 url: jdbc:mysql://localhost:3306/jpa\_mp?useSSL=false&useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Shanghai  
 type: com.alibaba.druid.pool.DruidDataSource  
  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true

3、写entity类，或者叫pojo类：

@Entity  
@AllArgsConstructor  
@NoArgsConstructor  
@Data  
public class Person implements Serializable {  
 @Id //主键  
 @GeneratedValue //指定如何标识属性可以被初始化，例如自动、手动、或从序列表中获得的值。  
 @TableId(type = IdType.AUTO)  
 private Long id;  
 @Column(name = "name", nullable = true, length = 20)  
 @TableField("name")  
 private String name;  
 @Column(name = "age", nullable = true, length = 20)  
 @TableField("age")  
 private int age;  
  
 @Column(name = "gender", nullable = false, length = 20 ,columnDefinition = "COMMENT '这是性别'")  
 @TableField("gender")  
 private int gender=0;  
  
 @Column(name = "classroom", nullable = false ,columnDefinition = "varchar(200) COMMENT '班级'")  
 @TableField("classroom")  
 private String classroom="1班";  
}

4、新建一个文件夹repository在里面写入PersonRepository然后继承JpaRepository

public interface PersonRepository extends JpaRepository<Person, Long> {  
}

1. controller调用即可：

@RestController  
@RequestMapping("/person")  
public class PersonController {  
 @Autowired  
 PersonRepository personRepository;  
 @Autowired  
 PersonService personService;  
 @GetMapping("/addPerson/{name}/{age}")  
 public String addPerson(@PathVariable(value = "name") String name,  
 @PathVariable(value = "age") int age){  
 Person person = new Person();  
 person.setId(null);  
 person.setName(name);  
 person.setAge(age);  
 personRepository.save(person);  
 return "添加成功";  
 }