# 持久层框架

### 一.mvc思想

模型（Model）：持久层，负责实现应用程序的业务逻辑，封装有各种对数据的处理方法。它不关心它会如何被视图层显示或被控制器调用，它只接受数据并处理，然后返回一个结果

控制层（Controller）：负责控制应用程序的流程，它接收从视图层传过来的数据，然后选择业务层中的某个业务来处理，接收业务层返回的结果并选择视图层中的某个视图来显示结果

视图层（View）：负责应用程序对用户的显示，它从用户那里获取输入数据并通过控制层传给业务层处理，然后再通过控制层获取业务层返回的结果并显示给用户。

优点：

低耦合 重用性高

生命周期成本低：MVC模式使开发和维护用户接口的技术含量降低。

开发效率快：它使后台程序员集中精力于业务逻辑，界面程序员集中精力于表现形式上 减少开发时间

可维护性高：分离视图层和业务逻辑层使得WEB应用更易于维护和修改

### 二，orm

1、hibernate使用orm思想对数据库进行crud操作

2、在web阶段学习javabean更正确的叫法是：实体类

3、orm： object relational mapping 对象关系映射

\* O：面向对象领域的Object（JavaBean对象）

\* R：关系数据库领域的Relational（表的结构）

\* M：映射Mapping（XML的配置文件）

### 三，mybatis+plus

半ORM思想使用sql语句进行crud操作 数据库的 持久层框架

它支持自定义 SQL、存储过程以及高级映射。

MyBatis 免除了几乎所有的 JDBC 代码以及设置参数和获取结果集的工作。

MyBatis 可以通过简单的 XML 或注解来配置和映射原始类型、接口和 Java POJO（Plain Old Java Objects，普通老式 Java 对象）为数据库中的记录。

使用select delete insert update 标签做增删改查

（1）优点：

① 基于SQL语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL写在XML里，解除sql与程序代码的耦合，便于统一管理；提供XML标签，支持编写动态SQL语句，并可重用。

② 与JDBC相比，减少了50%以上的代码量，消除了JDBC大量冗余的代码，不需要手动开关连接；

③ 很好的与各种数据库兼容（因为MyBatis使用JDBC来连接数据库，所以只要JDBC支持的数据库MyBatis都支持）。

④ 能够与Spring很好的集成；

⑤ 提供映射标签，支持对象与数据库的ORM字段关系映射；提供对象关系映射标签，支持对象关系组件维护。

（2）缺点：

① SQL语句的编写工作量较大，尤其当字段多、关联表多时，对开发人员编写SQL语句的功底有一定要求。

② SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

（3）#{}和${}的区别是什么？

${}是字符串拼接，#{}是占位符赋值；

Mybatis在处理${}时，就是把${}直接替换成变量的值。而Mybatis在处理#{}时，会对sql语句进行预处理，将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；

使用#{}可以有效的防止SQL注入，提高系统安全性。

（4）,使用MyBatis的mapper接口调用时有哪些要求？

Mapper接口方法名和mapper.xml中定义的每个sql的id相同；

Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同；

Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同；

Mapper.xml文件中的namespace即是mapper接口的类路径。

（5）.Mybatis动态sql有什么用？执行原理？有哪些动态sql？

Mybatis动态sql可以在Xml映射文件内，以标签的形式编写动态sql，执行原理是根据表达式的值 完成逻辑判断 并动态拼接sql的功能。

Mybatis提供了9种动态sql标签：trim | where | set | foreach | if | choose | when | otherwise | bind。

1. 常用的方法

selectById() 根据id查询

selectList(); 查List<对象>

selectPage() 分页

selectOne()按条件查

insert()新增

updateById()根据id查询

deleteById()根据id删除

### 四，hibernate

hibernate是全自动跨数据库开发效率高于jdbc的持久成框架

使用的是hql语句最终会转化为sql语句.语法是 from 对象名

（1），hibernate运行原理

1.通过configuration加载了hibernate.cfg.xml文件。

2.加载后创建sessionFactory.(sessionFactory是线程安全的)

3.然后同sessionFactory获得session。（创建会话，相当于jdbc中的connection 是线程不安全的）

4.通过session操作数据库，最后通过transaction来进行事物的控制.

（2），十．hibernate的五大核心接口

1.Configuration(类):

加载配置文件hibernate.cfg.xml文件中的配置信息，从而得到：

1）.hibernate的底层信息：

数据库连接，jdbc驱动，方言，用户名，密码。

2）.hibernate的映射文件(\*.hbm.xml)

2.sessionFactory（接口）

通过configuration创建sessionFactory。

可以用来获得session。

sessionFactory是线程安全的，

里边加载的数据信息有数据库的配置信息和映射关系.

3.Session(接口)

线程不安全。

相当于jbdc的connection.

用来操作数据库.

4.Transaction（接口）

进行事物的控制实现事物的commit（提交)和rollback（回滚）

5.Query(接口)

（3），hibernate主键生成策略

1》increment(银科瑞门特) :先选出当前表的最大ID，在此基础上加1，hibernate会自行处理，适用于int short,long类型的主键

2》sequence:oracle数据库会自行处理，会增加一个序列配置

<generator class="sequence">

<param name="sequence">sequence\_id</param>

</generator>

3》native（通用多个数据库，数据库的本地生成策略）

4》uuid：生成一个32位的，不重复的字符串，可以达到夸数据库

（4）．hibernate中session中的常用方法

save():保存

saveOrUpdate()保存或者修改

delete()删除

update()修改

get()根据唯一标示获取对象 立即加载

load()根据唯一标示获得对象 ，延迟加载

### 五，Spring Data JPA 框架

Spring Data JPA 是⼀套规范，内部是由接⼝和抽象类组成的， 是应⽤于Dao层的⼀个框架，简化数据库开发的，最明显的⼀个特点，Spring Data Jpa 开发Dao的时候，很多场景我们

连sql语句都不需要开发。由Spring出品。

Spring Data JPA 是 Spring 基于JPA 规范的基础上封装的⼀套 JPA 应⽤框架，可使开发者⽤极简的代码即可实现对数据库的访问和操作。它提供了包括增删改查等在内的常⽤功能！学习并使⽤Spring Data JPA 可以极⼤提⾼开发效率。

说明：：：：Spring Data JPA 极⼤简化了数据访问层代码。

如何简化呢？使⽤了Spring Data JPA，我们Dao层中只需要写接⼝，不需要写实现类，就⾃动具有

了增删改查、分⻚查询等⽅法。

使⽤Spring Data JPA 很多场景下不需要我们⾃⼰写sql语句

（1），Spring Data JPA 常用注解

1.@Entity：标识实体类是JPA实体，告诉JPA在程序运行时生成实体类对应

2.@Table：设置实体类在数据库所对应的表名

3.@Id：标识类里所在变量为主键

4.@GeneratedValue：设置主键生成策略，此方式依赖于具体的数据库

5.@Column：表示属性所对应字段名进行个性化设置

6.@Transient：表示属性并非数据库表字段的映射,ORM框架将忽略该属性

7.@Temporal：（很重要）

当我们使用到java.util包中的时间日期类型，则需要此注释来说明转化成java.util包中的类型。

注入数据库的类型有三种：

TemporalType.DATE（2008-08-08）

TemporalType.TIME（20:00:00）

TemporalType.TIMESTAMP（2008-08-08 20:00:00.000000001）

8.@Enumerated：（很重要）

使用此注解映射枚举字段，以String类型存入数据库

注入数据库的类型有两种：EnumType.ORDINAL（Interger）、EnumType.STRING（String）

9.@Embedded、@Embeddable：

当一个实体类要在多个不同的实体类中进行使用，而其不需要生成数据库表

@Embeddable：注解在类上，表示此类是可以被其他类嵌套

@Embedded：注解在属性上，表示嵌套被@Embeddable注解的同类型类

10.@ElementCollection：集合映射

11.@CreatedDate、@CreatedBy、@LastModifiedDate、@LastModifiedBy：（很重要）

表示字段为创建时间字段（insert自动设置）、创建用户字段（insert自动设置）、最后修改时间字段（update自定设置）、最后修改用户字段（update自定设置）

用法：

1、@EntityListeners(AuditingEntityListener.class)：申明实体类并加注解`

2 、实现AuditorAware类

3、springboot 启动类加上注解@EnableJpaAuditing

4、在实体类中属性中加上面四种注解

后续会写个这个的demo

12.@MappedSuperclass：（很重要）

实现将实体类的多个属性分别封装到不同的非实体类中

注解的类将不是完整的实体类，不会映射到数据库表，但其属性将映射到子类的数据库字段

注解的类不能再标注@Entity或@Table注解，也无需实现序列化接口

注解的类继承另一个实体类 或 标注@MappedSuperclass类，他可使用@AttributeOverride 或 @AttributeOverrides注解重定义其父类属性映射到数据库表中字段。

（2），dao层接口

⼀个符合SpringDataJpa要求的Dao层接⼝是需要继承JpaRepository和JpaSpecificationExecutor

JpaRepository<操作的实体类类型,主键类型>

封装了基本的CRUD操作

JpaSpecificationExecutor<操作的实体类类型>

封装了复杂的查询（分⻚、排序等）

1. jpa常用方法

1.persist()：将临时状态的对象保存进数据库。【插入一条新记录】

//由于涉及数据库增删改，执行该语句前需启用事务

2.entityManager.persist(modelObject);

2.merge()：将对象存入数据库，不同于persist()，merger()对于操作的对象，如果对象存在于数据库则对对象进行修改，

如果对象在数据库中不存在，则将该对象作为一条新记录插入数据库。

entityManager.merge(modelObject);

3.find()与getReference()：查找对象。不同点：

当对象不存在时，find()返回null, getReference()会抛出javax.persistence.EntityNotFoundException异常

4.remove()：将对象从数据库中删除。

entityManager.remove(entityManager.getReference(ModelObject.class,key));

5.refresh(Object obj)：重新从数据库中读取数据。

6.contains(Object obj)：判断对象是否在数据库中存在，返回true \ false。

7.flush()：立即写入数据库。

执行persist()、merger()时，数据并不是立即写入数据库中，

而是由JPA缓存起来，在执行flush()时写入。

在事务提交的时候，JPA会自动执行flush()一次性保存所有数据。

如果需要立即保存，可手动执行flush()。

setFlushModel()：修改EntityManager的Flush模式。

EntityManager的Flush模式默认为FlushModel.AUTO，这种模式下,在执行查询

（指使用JPQL语句查询时，不包括find()和getReference()查询）或事务提交时自动执行flush()。通过setFlushModel( FlushModel f )设置为FlushModel.COMMIT，该模式下只有在事务提交时才会执行flush()写入数据库。

### 六，JDBC

(1) 什么是JDBC，在什么时候会用到它？

JDBC的全称是Java DataBase Connection，也就是Java数据库连接，我们可以用它来操作关系型数据库。JDBC接口及相关类在java.sql包和javax.sql包里。我们可以用它来连接数据库，执行SQL查询，存储过程，并处理返回的结果。

JDBC接口让Java程序和JDBC驱动实现了松耦合，使得切换不同的数据库变得更加简单。

(2) JDBC的DriverManager是用来做什么的？

JDBC的DriverManager是一个工厂类，我们通过它来创建数据库连接。当JDBC的Driver类被加载进来时，它会自己注册到DriverManager类里面，你可以看下JDBC Driver类的源码来了解一下。

然后我们会把数据库配置信息传成DriverManager.getConnection()方法，DriverManager会使用注册到它里面的驱动来获取数据库连接，并返回给调用的程序。

（3）JDBC的ResultSet是什么？

在查询数据库后会返回一个ResultSet，它就像是查询结果集的一张数据表。

ResultSet对象维护了一个游标，指向当前的数据行。开始的时候这个游标指向的是第一行。如果调用了ResultSet的next()方法游标会下移一行，如果没有更多的数据了，next()方法会返回false。可以在for循环中用它来遍历数据集。

默认的ResultSet是不能更新的，游标也只能往下移。也就是说你只能从第一行到最后一行遍历一遍。不过也可以创建可以回滚或者可更新的ResultSet，像下面这样。

Statement stmt= con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

当生成ResultSet的Statement对象要关闭或者重新执行或是获取下一个ResultSet的时候，ResultSet对象也会自动关闭。

可以通过ResultSet的getter方法，传入列名或者从1开始的序号来获取列数据。

（4）如何回滚事务？

通过Connection对象的rollback方法可以回滚事务。它会回滚这次事务中的所有修改操作，并释放当前连接所持有的数据库锁。

# 控制层框架

### 一，springmvc

（1）springMVC的运行原理

1. 客户端请求提交到DispatcherServlet(diss趴扯Servlet)

2. 由DispatcherServlet控制器查询一个或多个HandlerMapping(憨的了卖屏)，找到处理请求的handler(憨的了)。

3. DispatcherServlet将请求转发给到handler。

4. 通过HandlerAdapter(憨的了额搭铺特)执行handler调用业务逻辑处理后，返回ModelAndView。

5. DispatcherServlet查询一个或多个ViewResolver()视图解析器，将model渲染到views。

6.将views结果显示到客户端

（2）接参传参

接参：request 基本数据类型 对象（注意：时间和文件）

传参：转发传值 json传值 artreebute

（3） 页面跳转

页面跳转的方式有 转发forward 重定向redirect

Redirect和forward的区别：

转发(forward)和重定向(redirect)的区别：

转发地址栏不发生变化，数据共享，一次请求，执行效率高；

重定向地址栏发生变化，数据不共享，多次请求，执行效率低。

1. 常用的注解有：@controller @RequestMapping @ResponseBody

### 二，springboot

微服务 进行接口开发的

Spring Boot是由 Pivotal 团队提供的全新的框架，spring boot 整合了所有的框架让spring应用程序尽快跑起来并且尽可能减少你的配置文件

（1）优点：

1）为所有 Spring 开发提供一个更快更广泛的入门体验

2）零配置。无冗余代码生成和XML 强制配置，遵循“约定大于配置”（我们可以直接用它里面默认的配置，不需要去修改）

3）集成了大量常用的第三方库的配置， Spring Boot 应用为这些第三方库提供了几乎可以零配置的开箱即用的能力

4）提供一系列大型项目常用的非功能性特征，如嵌入式服务器web、安全性、度量、运行状况检查、外部化配置等

5）Spring Boot 不是Spring 的替代者，Spring 框架是通过 IOC 机制来管理 Bean 的。Spring Boot 依赖 Spring 框架来管理对象的依赖。Spring Boot 并不是Spring 的精简版本，而是为使用 Spring 做好各种产品级准备

（2）常用注解：

@restcontroller @getmapping @PathVariable @MapperScan

@springbootapplication（他是由三个注解组成的分别是@springbootConfiguration读取配置文件，配置文件的路径是当前根目录@EnableAutoConfiguration开启自动配置，扫描当前的所有依赖的jar包，发现新的依赖出现将会根据依赖完成各种自动配置@ComponentScan属于spring 框架(@Component,@Service,@Controller,@Repository,@Entity)，扫描范围默认情况下是启动类所在的同名包及其子孙包）

### 三，Servlet

（1）Servlet工作过程

1、Web Client 向Servlet容器（Tomcat）发出Http请求

                   2、Servlet容器接收Web Client的请求

                    3、Servlet容器创建一个HttpRequest对象，将Web Client请求的信息封装到这个对象中

                    4、Servlet容器创建一个HttpResponse对象

                    5、Servlet容器调用HttpServlet对象的service方法，把HttpRequest对象与HttpResponse 对象作为参数传给 HttpServlet对象

                   6、HttpServlet调用HttpRequest对象的有关方法，获取Http请求信息

                   7、HttpServlet调用HttpResponse对象的有关方法，生成响应数据

                   8、Servlet容器把HttpServlet的响应结果传给Web Client

（2）Servlet生命周期

在Servlet框架中所有的Servlet类都必须实现Servlet这个接口。其中定义了三个方法：

                            1、init方法：负责初始化Servlet对象。

                            2、service方法：用于响应客户端的请求

                            3、destroy：销毁Servlet对象，释放占用的资源。

                       Servlet生命周期四个阶段：

                             ●  加载阶段:加载并实例化(创建Servlet实例)

                             ●  初始化阶段:调用init()方法

                             ●  响应客户请求阶段：调用service()方法,doGet、doPost

                             ●  终止阶段：调用destroy()方法

1. HTTP的请求方式包括DELETE、GET、OPTIONS、POST、PUT和TRACE,在HttpServlet类中分别提供了相应的服务方法,它们是doDelete()、doGet()、doOptions()、doPost()、doPut()和doTrace()。不过目前很多服务器只支持get和post请求，所以通常自定义的Servlet中只需要重写doPost和doGet方法即可。

# 前端框架

vue是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。

### （1）属性

{{info}} 取值

el:"#",绑定元素

data:{} 绑定数据

methods:{}绑定元素中包含的函数

Watch 监听属性

Computed 计算函数

### （2）VUE的常用指令

v-model指令 创建双向数据绑定

v-if指令，v-else指令 控制元素 有或没有

v-show指令，v-else指令 控制数据显示或不显示

v-for指令 遍历数组元素

v-bind指令 绑定标签的属性

v-on指令 用于给监听事件

### （3）生命周期 钩子函数

beforeCreate（创建前），created（创建后），

beforeMount（载入前），mounted（载入后），

beforeUpdate（更新前），updated（更新后），

beforeDestroy（销毁前），destroyed（销毁后）

### （4）什么是路由：

在我们浏览网页的过程中，进入一个网页，大多时候就会发起一次http请求，而前端中的路由，就是实现在页面中跳转而不会再次发送请求，有时我们再网页中看到 地址中带有#符号，这就代表了当前使用了路由

### （5）组件：

组件是可复用的 Vue 实例, 如果网页中的某一个部分需要在多个场景中使用，那么我们可以将其抽出为一个组件进行复用。组件大大提高了代码的复用率用来做分布式开发的

需要自定义标签 $emit子调父 $props父传子

### （6）什么是跨域

当一个请求url的协议、域名、端口三者之间任意一个与当前页面url不同即为跨域

### （7）为什么会出现跨域问题

出于浏览器的同源策略限制。同源策略（Sameoriginpolicy）是一种约定，它是浏览器最核心也最基本的安全功能，如果缺少了同源策略，则浏览器的正常功能可能都会受到影响。可以说Web是构建在同源策略基础之上的，浏览器只是针对同源策略的一种实现。同源策略会阻止一个域的javascript脚本和另外一个域的内容进行交互。所谓同源（即指在同一个域）就是两个页面具有相同的协议（protocol），主机（host）和端口号（port）

### （8）跨域解决方法：

JSONP 是服务器与客户端跨源通信的常用方法。最大特点就是简单适用，兼容性好（兼容低版本IE），缺点是只支持get请求，不支持post请求

Urus：

@CrossOrigin

以便启用CORS（默认情况下，@CrossOrigin允许在@RequestMapping注解中指定的所有源和HTTP方法

### （9）什么是同源（域）：

相同的协议

相同的ip

相同的端口

# 安全框架

### （1）什么是shiro

Shiro是apache旗下一个开源框架，它将软件系统的安全认证相关的功能抽取出来，实现用户身份认证，权限授权、加密、会话管理等功能，组成了一个通用的安全认证框架。

Shiro可以完成，认证，授权，加密，会话管理，Web集成 ，缓存等方面

### （2）为什么要学shiro

既然shiro将安全认证相关的功能抽取出来组成一个框架，使用shiro就可以非常快速的完成认证、授权等功能的开发，降低系统成本。

shiro使用广泛，shiro可以运行在web应用，非web应用，集群分布式应用中越来越多的用户开始使用shiro。

### （3）Shiro的认证

① 创建Security Manager：Security Manager是用来提供安全服务的，所以在做shiro认证的时候要先创建此对象

② 主体Subject提交请求给Security Manager

③ Security Manager调用Authenticator组件做认证

④Authenticator通过Realm来从数据源中获取认证数据

### （4）Shiro的授权（shiro授权过程与认证过程基本一致）

① 创建Security Manager

② 主体subject授权

③ 主体授权是交给Security Manager授权

④ Security Manager调用授权器Authorizer授权

⑤ 通过Realm在数据库或者缓存中来获取授权的数据（角色数据和权限数据）