|  |  |  |  |
| --- | --- | --- | --- |
| **基本情况** | | | |
| **姓名** | 郑洋 | 性别 | 男 |
| **出生日期** | 1996-10-10 | 属相 | 猪 |
| **现住地** | 升龙又一城 |  |  |
| **大学情况** | | | |
| **学校** | 郑州工商学院 | 专业 | 计算机科学与技术 |
| **入学时间** | 2013-09-01 | 毕业时间 | 2017-07-01 |
| **专业课** | **C语言、数据结构、数据库原理、操作系统实训、Java程序设计、高级语言** | | |
| **学历** | **本科** | 学位 | **学士学位** |
| **学校地址** | 河南省郑州市郑东新区前程大道169号 | 校长 | **不知道** |
| **去学校的方式：** | **从郑州火车站乘坐郑州地铁一号线至市民中心站转570路公交车** | | |
| **上小学日期：** | **2001年9月 6岁** | | |
| **上初中日期：** | **2007年9月 12岁** | | |
| **上高中日期：** | **2010年9月 15岁** | | |
| **上大学日期：** | **2013年9月 18岁** | | |
| **毕业时间：** | **2017年 7月 22岁** | | |
| **入学时间：** | **2013年9月** | | |
| **毕业时间：** | **2017年7月** | | |
| **在校学生：** | **5000左右** | | |
| **有教师：** | **不太清楚** | | |

|  |  |
| --- | --- |
| 第一家公司 ： | 吉运软件 |
| **地址：** | 上海市 金山区 金山信息大楼 卫零路465号13楼 桥英馨苑-27号 |
| **入职时间：** | **2017年9月初** |
| **离职时间：** | **2018年10月** |
| **公司性质：** | **民营** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **武学信** |
| **薪资方面：** | **刚刚实习转正以后工资是6K左右，过完年大概5月份涨到7K左右** |
| **路线：** |  |

|  |  |
| --- | --- |
| **第二家公司情况：** | 北京明润华创科技有限责任公司(只说公司简称) |
| **公司地址**： | 北京市石景山区政达路北方中惠国际中心D座7层 |
| **公司规模：** | **60人左右，我们项目组6人，一个项目经理，1名前端，1个测试，1个运维，其他java开发** |
| **公司性质：** | **民营** |
| **公司行业：** | 计算机硬件、IT服务、计算机软件、互联网 |
| **去公司路线：** | 5号线到东单，转乘1号到八宝山，直行1公里到万达广场，后面就是我们公司 鲁谷小区  从火车站坐663路 |
| **入职时间：** | **2018年11月** |
| **离职时间：** | **2021年3月** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **席俊哥** |
| **税前：** | **14K** |
| **税后：** | **13K** |
| **缴税：** | **1K左右(具体原因不清楚当时公司就是这样扣的税)** |
| **期望薪资：** | **15K** |
| **薪资的构成:** | **岗位工资(5K)+绩效工资(9K)** |

|  |
| --- |
| 四年涨薪：  **2017年9月入职6k**  **2018年5月涨1k，7k 10月离职前7k**  **2018年11月跳槽涨3K总 10K左右**  **2019年12月工作一年涨了2K 12k**  **2020年7月工作一年涨了1K 13k**  **2011年2月离职时工资为14K** |

|  |  |
| --- | --- |
| **你 为什么从上家公司离职？** |  |
| **上上家公司的福利待遇也算不错，和同事相处的也蛮开心的，但是在技术提升方面不太符合自己发展的预期。就出来历练一下** |  |
| **你交社保了吗？为啥没交？** |  |
| **交了** |  |
| |  | | --- | | **三个词形容自己**： 交流表达能力强，团队协作能力强，责任心强 | | **你的优缺点是啥？** |   **优点:**  **善于倾听，团队协作能力强，责任心强。**  **缺点: 这几年做程序，因为要不断的对代码进行验证确认，所有感觉自己现在有点强迫症。** |  |
| **你知道五险一金都是啥吗？** |  |
| **养老保险、医疗保险、失业保险、工伤保险、生育保险，和住房公积金。** |  |
| **你能接受加班吗** |  |
| **能** |  |
| **你能接受出差吗** |  |
| **能** |  |
| 你期望薪资多少？你上家工资多少？税后拿到手的有多少？扣了多少钱的税？ |  |
| 期望薪资 16K 上家工资15K 税后14K左右 扣了1K左右的税 |  |
| 你的五年规划是啥？ |  |
| **近期的话我会先让自己的技术能力得到提高，当然，尽快的掌握公司的业务，投入到工作中，然后朝着项目经理发展** |  |
| 是统招吗 是是本科吗 是能在学信网查到吗 这个因为我当时上的是民办大学，所以在民教网查的到！ |  |
| 你们公司还有什么福利？多少薪： 13薪  **车补**  300左右  **项目奖金**  7k-10k，做完项目，收完尾款的时候。  **年终奖**  这个就是1个月的工资，  过年的时候和当月工资一块发  **电话补助**  300  **饭/餐补**  **15、**30/天  **电脑补**  300 |  |

**公司内其他部门：**

**技术部（开发部/研发部），行政部，销售部，财务部**

**---------------------------------------------------------------------**

**其他问题：**

**笔试问题：我工作这么多长的时间了，理论上的知识忘得差不多了，你和你们经理联系一下直接面试吧，不行的话那么我就去直接去下一家！**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

坐稳再开口！！ 主动性！！ 开门见山！！大白话！！三秒钟原则！！学会反问！！

**Hr人事，leader领导，tean团队，pm项目/产品经理，cto首席技术执行官，coo首席运营官，CEO董事长、**

您好！我是来面试咱们公司的java 开发的！我是17年毕业的，到今年呢已经是有3年多的工作经验了，在这3年里我大大小小做的有几个项目。在这四年里面我也掌握了不少技术，比如说你像一些主流的开发框架，像什么Spring、SpringMVC啊、MyBatis、Hibernate、MyBatis-Plus啊、SpringBoot啊、还有SpringCloud都接触过。这是主流的开发框架，然后你像一些常用的关系型数据库比如说MySQL、Oracle这些我也用的比较多，像MySQL的主从复制还有读写分离啊也做过。然后你像一些分布式缓存比如说Redis我也用的不少，像Redis集群我也搭建过。还有的话你像消息队列中间件比如说RabbitMQ之类的我也用过,微信支付或者是支付宝支付这块我之前也做过。然后**像**接口测试的话，你像Postman对接口进行功能测试，我也用过。还有就是像SQL优化啊、JVM优化啊、Tomcat优化之类的我也了解过。这就是我这几年来的一个工作经历。

主动说：

我最近做的一个项目是使用 **Spring Boot+springCloud 来做的微服务项目的开发**

首先先来说一下，这个springboot项目具体是怎么做的吧！

第一你是需要搭建springboot项目的

**1 关于redis整理**

五大数据类型分别是什么

List类型

List 说白了就是链表（redis 使用双端链表实现的 List），是有序的，value可以重复，可以通过下标取出对应的value值，左右两边都能进行插入和删除数据。

因为list 是先进先出 所以可以用来做消息队列

Set类型

和java的set集合差不多 都是无序没有小标 不能排序

Zset类型

和java的zset集合差不多 有序有下表 可以排序

hash数据类型 是一个 key file value 结构

**给面试官介绍**

redis就是一个基于缓存的非关系型数据库，他的优点的话就是丰富的数据类型，hash，list，set，zset，String，它是支持事务的，他还有aof和rdb两种持久化方案，aof的话就是把咱们的操作都记录再日志中，每次把redis启动起来之后它就会把这个日志表中的所有命令都进行执行，而rdb的话就是镜像，再我们修改或者新增一定数据的时候它会自动进行保存这时候的镜像，我们启动redis的时候，它就会回到最近的一个镜像，它相对与aof就效率要高一些，但是容易丢失一些数据，所以一般存储一些比较重要点的数据的话还是建议用aof持久化 ，redis还是它查询是基于内存的，所以是非常的快 停顿，我们在项目中的话，一般是用它来存点热点数据，把那些访问量比较高的给存进去，用它来做一下缓存不直接访问到数据库，之后再就是用它来解决过分布式锁的问题，我当时是做一个定时发短信的任务，就是用户生日的时候发一封短信，单个服务的时候就好好的，就是集群的时候，它到了这个时间段就一下发了好多封短信，每个集群的服务都调用了这个定时任务，这时候我们就借助了redis，通过哪个getAndSet方法取和存都是一步操作，然后就判断没值的时候就往下运行，有值的时候就直接返回，正当发送的时候，执行完getAndset报错了，找到错之后去运行发现，现在判断根本就过不了了，redis中数据一直存在了，这就是另一个小问题，就相当于你a进程拿到了这个锁，但是他死了，锁也就跟着没了，下面的进程永远都进不去了，这样的话就费了，所以我就给key设置了一个失效时间，这样就算他报错了，照样不会把锁也带走。这时候就如果还不说话的话

然后redis的话还有三大问题，就是缓存击穿，缓存穿透，缓存雪崩，缓存击穿的话就是一个热点数据的key到了失效时间，突然就来了一大堆的请求，直接击穿了我们的redis，访问到了数据库，这样瞬间数据库压力就大了，所以为了防止这个的问题出现，我们可以写一个定时任务，当时间快到的时候给它进行续命，还可以直接就设置成永不过期，缓存穿透的话就是有人恶意访问的情况下，就一直访问你缓存和数据库都不存在的一个key，这样的话我们加这个缓存就没有任何作用了，解决方案的话就是把这个key存到redis中并赋个空值，他再进行查的话就直接返回一个空，就不会访问到数据库，之后再就是缓存雪崩，缓存雪崩的话就是某一时间大量的key失效，所有的请求都压到了数据库，也会导致数据库崩溃，解决方案的话就是尽量不要设置同一时间吧。我介绍完了

**2 多线程导出**

线程的实现方式的话就是实现一个Runnable接口，还有继承Thread类重写它的run方法，在里面写具体实现，在主线程中创建线程调用一个start方法就创建好了，线程的五个状态的话就是，你创建Thread的时候就是新建状态，然后调用start方法的时候就就绪状态，当cpu把时间片分配给你的时候，这个线程就是运行状态，其他线程的话就在就绪状态中排队，当你执行完就到了死亡状态，如果你没有执行完，cpu把时间片分给了别人，这时候你就会回到就绪状态，等待着cpu选到你，还有个阻塞状态的话，就是当你加锁的情况下，a线程拿到了锁，但是cou分配的时间到了，你没执行完，就会把资源锁住，这时候b线程被cpu选到了，它去执行一看被锁了，这时候b线程就会进入到阻塞状态，直到等到a线程执行完，把锁给释放了b线程才会从阻塞状态出来

我当时的话是用它做了一个poi的导出，当时一开始是正常的通过反射加自定义注解通过这个HSSFWorkbook做一个导出，但是发现效率太慢，数据过多，之后我就把每创建一个Excel就启动一个线程，然后每次查询只查一个sheet页的数据，这样确实快了很多，但是还是不完美，之后我又加了线程池，把线程数设置最大线程数，这样线程也不需要频繁的创建和销毁了，就快完成的时候了，一下载发现导出来的是个空的，之后一想，多线程是异步的，线程还没完成，主线程是不等待的，直接执行了，然后我就百度搜索解决方案，最后找到了CountDownLatch这个方法，把线程的总数给它 ，在主线程最后下载的地方调用一个await方法，这个方法就是让你等待，当CountDownLatch数量到0的时候才会执行， 然后再线程内调用countDown方法，这个就是每次执行了这个方法就会线程减一当，减到0就证明所有线程走完了，然后主线程就会执行，加了这个之后就把这个问题解决了

**3 定时任务 （发邮件 发短信 会员续费）**

我当时是做一个定时发短信的任务，就是用户生日的时候发一封短信，

我们当时是分布式框架spring+springboot+springCloud定时器我们用的是springboot的@Scheduled 定时器我们只需要创建一个类加上三个注解@Component 交给ioc进行管@Configuration声明这是一个文件@EnableScheduling //启用调度器然后规定一个定时时间定时扫描就行了当时我们是把所有用户的信息查询出来，然后进行便利看对比今天的时间是哪个用户的生日，然后就进行起一个线程去发邮件，然后再继续便利，单个服务的时候就好好的，就是集群的时候，它到了这个时间段就一下发了好多封短信，每个集群的服务都调用了这个方法，这时候我们就借助了redis，通过哪个getAndSet方法取和存都是一步操作，然后就判断没值的时候就往下运行，有值的时候就直接返回，正当发送的时候，执行完getAndset报错了，找到错之后去运行发现，现在判断根本就过不了了，redis中数据一直存在了，这就是另一个小问题，就相当于你a进程拿到了这个锁，但是他死了，锁也就跟着没了，下面的进程永远都进不去了，这样的话永远也发不了短信了，然后我就百度了一下知道了redis中能给key设置一个失效时间，这样就算他报错了，一定时间，锁也会自动消失

**4 aop日志记录**

首先，日志记录我们是通过aop+自定义注解来做的

为什么用aop呢，就是因为aop是不改变原代码的情况下增加新的功能，如果你不用的话你需要再每个方法都进行记录一下日志，这样是非常繁琐的，所以我用了aop思想，

然后我们创建一个类，在类上添加两个注解，分别是@Component @Aspect，这两个注解的作用分别是：

@Component：将本类交给ioc管理

@Aspect：声明这个是切面类。

在该类中新建方法，方法上添加注解，声明aop的通知，aop的通知共有五个，@Before前置通知，@After后置通知，@AfterThrowing异常通知， @Around环绕通知，@AfterReturning最终通知，设置参数，表明横切那个层，在这里我使用@Before前置通知，横切controller层。

在方法上添加参数JoinPoint，使用它，来获取方法的签名，再根据方法签名来查找到所有的方法。这时候我们通过getDeclaredAnnotations 获取方法上面的注解，判断方法上是否有注解如果有注解的话就进行下面的操作，把数据保存到我们日志里面

反射的常用方法

①、得到 Class 的三种方式

//1、通过类对象调用 getClass() 方法来获取

　　Person p1 = new Person();

　　Class c1 = p1.getClass();

//2、直接通过 类名.class 的方式得到,该方法最为安全可靠，程序性能更高

　　Class c2 = Person.class;

//3、通过 Class 对象的 forName() 静态方法来获取，用的最多，

// 但可能抛出 ClassNotFoundException 异常

　　Class c3 = Class.forName("com.ys.reflex.Person");

其中有很多方法，比如说

getModifiers();获取类上的修饰符

　　getName()：获得类的完整名字。

getPackage(): 获得该类所在的包路径

getDeclaredAnnotations(); 获取该类上所有注解

　　getFields()：获得类的public类型的属性。

　　getDeclaredFields()：获得类的所有属性。包括private 声明的和继承类

　　getMethods()：获得类的public类型的方法。

　　getDeclaredMethods()：获得类的所有方法。包括private 声明的和继承类

　　getConstructors()：获得类的public类型的构造方法。

**5基于token的登陆**

用户点击登录 把用户名和密码传过去 服务端接收账号密码进行判断账号是否存在，

存在的话再判断密码是否正确，如果正就使用JWT的工具类来进行数据加密 得出一个token值

把token值返回给客户端，客户端把他存到sessionstor里面，然后使用路由守卫获取

sessionstorre里面的值判断值是否为空，为空就直接跳转到登录页面，不为空直接下一步

，服务端的拦截器先获取传过来的token值判断token值是否为空，不为空在进行解密

判断解密出来是否有数据，有数据就证明用户登录过了，就放开请求，

又因为我们认证需要请求中带有token值，所有每个请求都带有token值特别繁琐，

所以我们使用请求拦截器，所有的请求，请求之前都给他加上token值，就解决了每个请求都加token值的繁琐。

1. **SSM框架搭建的步骤**
2. 导入Spring、SpringMVC、MyBatis的依赖
3. 导入Spring、SpringMVC、MyBatis的配置文件
4. 在Spring的配置文件中配置如下内容：

(1)配置数据源

(2)配置MyBatis的会话工厂SqlSessionFactoryBean，并注入数据源

(3)配置MyBatis自动扫描DAO层Mapper接口和配置MyBatis映射文件的路径

(4)通过component-scan注解开启对service层的扫描

(5)配置事务的传播特性、隔离级别、回滚策略，让那些以add、delete、update开头的方法开启事务的提交和回滚，其它方法走的是只读事务。

(6)配置事务管理切面类，横切service层。

4. 在SpringMVC配置文件中配置如下内容：

(1)使用component-scan标签开启对controller层的扫描

(2)通过annotation-driven标签开启注解驱动

(3)配置视图解析器

(4)如果需要进行文件上传操作，还需要配置文件上传解析器，并指定文件上传的大小 限制

5. 在MyBatis的配置文件中配置如下内容：

(1)配置在控制台打印SQL语句

(2)配置开启二级缓存

(3)配置下划线自动转小驼峰

6. 在web.xml中配置如下内容

(1)配置Spring的监听，并加载Spring的配置文件

(2)配置SpringMVC的核心控制器DispatcherServlet并加载SpringMVC的配置文件

(3)为了防止中文乱码，一般还会配置一个编码过滤器并设置字符编码集为UTF-8

7. 其实到此为止，咱们的SSM框架就算搭建完毕了，接下来就可以进行项目的开发了。

新建控制层并在类上加上@Controller注解，然后在控制层类的方法上添加@RequestMapping注解，接下来在控制层类中通过@Autowired或@Resource注解注入Service层，在Service层通过@Autowired或@Resource注解注入DAO层，这里需要注意的是，DAO层只有接口没有实现类，是通过MyBatis映射文件中的namespace指明实现的DAO层Mapper接口，并通过MyBatis映射文件中的SQL的ID指明调用DAO层Mapper接口方法时要执行的SQL语句。

1. **谈谈你对Spring的理解？**

Spring 的核心就是 IOC 和AOP。

IOC（Inverse Of Control）控制反转，就是将原本在程序中手动创建对象的控制权，交由Spring框架来管理。 Spring使用IOC容器来实现IOC， IOC容器实际上就是个Map（key，value）,key就是Bean的ID，value就是对应的Bean对象。通过将对象之间的相互依赖关系交给IOC容器来管理，然后IOC容器来完成对象的注入。这样可以很大程度上简化应用的开发，把应用从复杂的依赖关系中解放出来。IOC容器就像是一个工厂一样，当我们需要创建一个对象的时候，只需要配置好配置文件/注解即可，完全不用考虑对象是如何被创建出来的。 然后在需要的地方引用就行了，这大大增加了项目的可维护性且降低了开发难度。

AOP基于代理设计模式，代理分为动态代理和静态代理，在项目中默认使用的是基于JDK的动态代理，它需要接口的支持，这就是在写 service 业务逻辑层的时候通常先写接口，再写实现类的原因，如果没有接口只有类，这个时候可以使用 CGLib 这个动态代理来实现。AOP说白了就是面向切面编程，就是把一些非核心业务逻辑提取出来形成一个切面，从而让咱们程序员在编写代码时候，只用关注核心业务逻辑的处理，至于这些非核心业务逻辑统一在切面里面处理。 像日志记录，性能统计，事务处理，安全控制，这些都可以通过AOP的方式进行统一的处理。切面说的简单点就是在指定类的指定方法的前后执行特定的横切业务逻辑。切面由切点和通知构成，通知又包括方位和横切业务逻辑，切点就是为了定位指定类的指定方法，方位包括前置通知，后置通知，环绕通知，抛出异常通知，返回后通知这些。横切业务逻辑说白了就是一些公共的非核心业务代码。AOP的切面在项目中用的还是比较多的，咱们都知道事务管理就是通过AOP切面的方式来实现的。切面的好处就是把分散在代码中的重复代码提取出来进行统一的维护和控制。我在项目中就使用aop完成日志的统一处理。原来是把日志记录的相关代码分散到各个控制层的相关方法中，但这就会导致程序员在开发时候不能将精力集中到业务逻辑的处理上，还得考虑记录日志，工作效率就大打折扣，我在项目中负责过日志管理模块，通过...[可以将基于AOP的日志统一处理融进去。

1. **Spring的优点？**
2. 方便解耦，简化开发：通过Spring提供的IOC容器，我们可以将对象之间的依赖关系交由Spring进行控制，避免硬编码所造成的过度程序耦合。
3. AOP面向切面编程：通过Spring提供的AOP功能可以让我们对系统中一些横切业务逻辑如安全、事务、日志等进行集中式管理，从而让这些功能得到更好的复用。
4. 声明式事务管理：Spring的声明式事务管理让我们无须通过编程的方式管理事务，只需要在配置文件中或者通过注解进行相关的规则声明，就可以将事务规则应用到业务逻辑中，提高开发效率和质量。
5. 方便集成各种优秀框：Spring不仅不排斥各种优秀的开源框架，相反，Spring可以降低各种框架的使用难度，Spring提供了对各种优秀框架（如Struts,Hibernate）等的直接支持。
6. **Spring的缺点？**

我说一下我的感受吧，现在Spring越来越大，支持的东西越来越多，这就导致大多数开发人员对Spring的体系及结构都一知半解，用到某个东西的时候在网上乱七八糟的文章来回粘来粘去，有些时候甚至搞得人都分不清楚到底哪种使用方式是对的，因为越来越臃肿的体系，导致Spring的学习成本倍增。

1. **Spring 中的 bean 的作用域有哪些?**

singleton : 单例，每个bean都只会创建一个实例，Spring 中的 bean 默认都是单例的。

prototype : 原型，每次从IOC容器中获取都会都会产生一个新的bean实例。

request : 每次HTTP请求都会产生一个新的bean实例，这个实例仅在当前请求内有效。

session : 每个HTTP会话都只会创建一个 bean实例，这个实例只在当前会话内有效。

global-session： 所有的HTTP会话都只会创建一个bean实例。

1. **Spring中的Bean为什么默认是单例的？有什么缺点吗？**

一句话就是为了提高性能，具体点：

1. 由于不会每次都新创建新对象，所以就减少了新生成实例的消耗。因为spring会通过反射或者cglib来生成bean实例这都是耗性能的操作，其次给对象分配内存也会涉及复杂算法。
2. 减少JVM垃圾回收，由于不会给每个请求都新生成bean实例，所以自然回收的对象少了。

第三，可以快速获取到bean，因为单例的获取bean操作除了第一次生成之外其余的都是从缓存里获取的所以很快。

缺点就是在并发环境下可能会出现线程安全问题。

1. **Spring的Bean加载的过程？**
2. 通过反射实例化Bean;
3. 对bean的依赖属性进行填充
4. 如果实现了BeanNameAware接口,调用setBeanName设置Bean对象的ID或者Name;
5. 如果实现BeanFactoryAware接口,调用setBeanFactory 设置BeanFactory;
6. 如果实现ApplicationContextAware,调用setApplicationContext设置ApplicationContext
7. 调用BeanPostProcessor的预先初始化方法;
8. 调用InitializingBean的afterPropertiesSet()方法;
9. 调用定制init-method方法；
10. 调用BeanPostProcessor的后初始化方法;
11. **SpringMVC的运行原理**

整个处理过程从一个HTTP请求开始：

1. 客户端发送请求到 DispatcherServlet。
2. DispatcherServlet根据请求信息调用HandlerMapping，解析请求对应的Handler。
3. 解析到对应的Handler（也就是我们平常说的Controller控制器）后，开始由HandlerAdapter适配器处理。
4. HandlerAdapter会根据Handler来调用真正的处理器开处理请求，并处理相应的业务逻辑。
5. 处理器处理完业务后，会返回一个ModelAndView对象，Model是返回的数据对象，View是个逻辑上的View。
6. ViewResolver会根据逻辑View查找实际的View。
7. DispaterServlet把返回的Model传给View进行视图渲染。
8. 最后DispatcherServlet将渲染好的视图响应给客户端
9. **SpringMVC的常用注解有哪些？**

@RequestMapping：用来设置方法的请求地址。它呢可以用在类上也可以用在方法上，用在类上的话，则访问所有的请求方法时都需要加上该地址作为父路径。

@GetMapping：也是用来设置方法的请求地址的，但是只支持get请求。一般在进行查询操作的时候会用到这个注解。

@PostMapping：也是用来设置方法的请求地址的，但是只支持post请求。一般在进行新增操作或者传递的参数比较多的情况下会用到这个注解。

@DeleteMapping：也是用来设置方法的请求地址的，但是只支持delete请求。一般在进行删除操作的时候会用到这个注解。

@PutMapping：也是用来设置方法的请求地址的，但是只支持delete请求。一般在进行修改操作的时候会用到这个注解。

@ResponseBody：用于将服务器端返回java对象转为JSON数据。

@RequestBody: 用于将客户端传递过来的JSON数据转换为java对象。

@RestController: 它是一个组合注解，组合了@Controller和@ResponseBody这两个注解。如果说某个Controller所有的方法都需要返回JSON格式的数据，直接在Controller类上加上这个注解，这样就不用我们在每个方法上再单独添加@ResponseBody注解了。

@PathVariable：用来获取路径参数

@RequestParam：用来设置请求参数和请求方法里的参数的映射，一般用于请求参数名和请求方法中参数名不一致的情况。

1. **@Resource和@Autowired的区别？**

@Autowired是Spring提供的注解，@Resource是JDK提供的注解。

@Autowired只能按类型注入，@Resource默认按照名称注入，也可以通过类型注入。

@Resource默认按照名称进行注入，如果注入失败的话会转为按类型进行注入，如果按类型也注入失败会抛出异常。

1. **MyBatis是什么?**

MyBatis是一个持久化框架，是一个半自动ORM 框架，它的本质是对JDBC的封装。MyBatis不仅支持自定义SQL语句、存储过程，而且使用MyBatis不用我们手动设置SQL语句中参数，也不需要我们手动把查询出来的结果集映射为java对象。在使用传统JDBC时，我们往往需要写很多JDBC代码，比如说加载驱动、创建连接、创建statement。然后还需要自己写SQL语句以及自己手动给SQL语句中参数赋值，最后自己对结果集进行封装处理，而MyBatis帮我们简化了以上功能，只需要一些配置文件（xml）或是注解的方式即可完成对数据库的查询以及结果的映射封装。相比于原生JDBC而言，使用MyBatis让我们只需要关注SQL语句本身，而不需要花费精力去处理加载驱动、创建连接、创建statement等这些和SQL语句无关的繁琐过程。

1. **MyBatis的优点有哪些？**

第一实现了SQL语句和代码的分离，解除了SQL语句与代码的耦合。第二使用MyBatis可以自动将查询结果集转换为java对象。第三使用MyBatis无需我们手动设置SQL语句中的参数。第四MyBatis提供了很多xml标签方便我们编写动态SQL语句（比如说where,if,foreach,sql,include等标签）。

1. **MyBatis的缺点有哪些？**

第一就是SQL语句的编写工作量较大，尤其是字段多、关联表多时更是如此，对开发人员编写SQL语句的功底有一定要求。第二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，特别是在SQL语句中使用到一些函数的时候移植性更差。第三由于映射文件里SQL语句的id必须唯一，导致DAO层Mapper接口中的方法不支持方法重载。第四不支持级联更新、级联删除。

1. **MyBatis和Hibernate的区别(至少3点)?**

MyBatis容易掌握，只要会写SQL语句就行了。而Hibernate门槛较高，要精通门槛更高，而且怎么设计O/R映射，在性能和对象模型之间如何权衡取得平衡，以及怎样用好Hibernate方面需要你的经验和能力都很强才行。

MyBatis可以进行更为精细的SQL优化，而Hibernate要进行SQL优化会比较困难一些。

Hibernate无需写SQL就可以完成单表简单增删改查，而MyBatis单表简单增删改查还需要自己写SQL语句。

Hibernate数据库移植性很好，不同的数据库需要写不同SQL。

Hibernate有更好的二级缓存机制，可以使用第三方缓存。MyBatis本身提供的缓存机制不佳。

1. **MyBatis和Hibernate的区别(至少3点)?**

MyBatis容易掌握，只要会写SQL语句就行了。而Hibernate门槛较高，要精通门槛更高，而且怎么设计O/R映射，在性能和对象模型之间如何权衡取得平衡，以及怎样用好Hibernate方面需要你的经验和能力都很强才行。

MyBatis可以进行更为精细的SQL优化，而Hibernate要进行SQL优化会比较困难一些。

Hibernate无需写SQL就可以完成单表简单增删改查，而MyBatis单表简单增删改查还需要自己写SQL语句。

Hibernate数据库移植性很好，不同的数据库需要写不同SQL。

Hibernate有更好的二级缓存机制，可以使用第三方缓存。MyBatis本身提供的缓存机制不佳。

1. **MyBatis#{}占位符和${}拼接符的区别？**

#传入的参数在SQL中显示为字符串，#方式能够很大程度防止sql注入；$传入的参数在SqL中直接显示为传入的值，$方式无法防止Sql注入。

Select ${cloumn1},${cloumun2} from ${table\_name}

1. **Mybatis中映射文件和Mapper接口有什么对应关系吗？**

映射文件中的mapper标签的namespace属性值必须要和对应的Mapper接口路径保持一致

映射文件中SQL语句的ID要和Mapper接口中的方法名保持一致

映射文件中SQL语句的resultType要和Mapper接口中的方法的返回值类型保持一致

映射文件中SQL语句的parameterType要和Mapper接口中方法的参数类型保持一致