## SpringMVC、Mybatis、Spring

#### \*请写出 spring 中常用的依赖注入方式。

常见的就是 setter 注入 和 构造方法 注入。

另外还有静态工厂的方法注入、实例工厂的方法注入。

#### 简述Spring中IOC容器常用的接口和具体的实现类。

1. BeanFactory SpringIOC容器的基本设置，是最底层的实现， 面向框架本身的.

2. ApplicationContext BeanFactory的子接口, 提供了更多高级的特定. 面向开发者的.

3. ConfigurableApplicationContext, ApplicationContext的子接口，扩展出了 close 和 refresh等 关闭 刷新容器的方法

4.ClassPathXmlApplicationContext：从classpath的XML配置文件中读取上下文，并生成上下文定义。应用程序上下文从程序环境变量中取得。

5、FileSystemXmlApplicationContext ：由文件系统中的XML配置文件读取上下文。

6、XmlWebApplicationContext：由Web应用的XML文件读取上下文。

#### 简述Spring中如何基于注解配置Bean和装配Bean,

(1). 首先要在Spring中配置开启注解扫描

|  |
| --- |
| <context:component-scan base-package=” ”></ context:component-scan> |

(2). 在具体的类上加上具体的注解.

(3). Spring 中通常使用@Autowired 或者是@Resource 等注解进行bean的装配.

#### 说出Spring 或者 Springmvc中常用的5个注解 ，并解释含义

[1]. @Component 基本注解，标识一个受Spring管理的组件

[2][. @Controller](mailto:.@Controller) 标识为一个表示层的组件

[3]. @Service 标识为一个业务层的组件

[4]. @Repository 标识为一个持久层的组件

[5]. @Autowired 自动装配

[6]. @Qualifier(“”) 具体指定要装配的组件的id值

[7]. @RequestMapping() 完成请求映射

[8]. @PathVariable 映射请求URL中占位符到请求处理方法的形参

只要说出5个注解并解释含义即可，如上答案只做参考

#### 请解释Spring Bean的生命周期？

1.默认情况下，IOC容器中bean的生命周期分为五个阶段:

1. 调用构造器 或者是通过工厂的方式创建Bean对象
2. 给bean对象的属性注入值
3. 调用初始化方法，进行初始化， 初始化方法是通过init-method来指定的.
4. 使用
5. IOC容器关闭时， 销毁Bean对象.

2.当加入了Bean的后置处理器后，IOC容器中bean的生命周期分为七个阶段:

① 调用构造器 或者是通过工厂的方式创建Bean对象

② 给bean对象的属性注入值

③ 执行Bean后置处理器中的 postProcessBeforeInitialization

1. 调用初始化方法，进行初始化， 初始化方法是通过init-method来指定的.
2. 执行Bean的后置处理器中 postProcessAfterInitialization
3. 使用
4. IOC容器关闭时， 销毁Bean对象

只需要回答出第一点即可。 第二点也回答可适当加分.

#### 简述SpringMvc里面拦截器是如何定义，如何配置，拦截器中三个重要的方法

(1).定义: 有两种方式

[1]. 实现HandlerInterceptor接口

[2]. 继承HandlerInterceptorAdapter

(2).配置:

|  |
| --- |
| <mvc:interceptors>  <!--默认是对所有请求都拦截 -->  <bean id="myFirstInterceptor" class="com.atguigu.interceptor.MyFirstInterceptor">  </bean>  <!-- 只针对部分请求拦截或者不拦截 -->  <mvc:interceptor>  <mvc:mapping path=" " /> <!—指定拦截-->  <mvc:exclude-mapping path=””/> <!—指定不拦截-->  <bean class=" com.atguigu.interceptor.MySecondInterceptor " /> </mvc:interceptor>  </mvc:interceptors> |

(3).拦截器中三个重要的方法

[1]. preHandle

[2]. postHandle

[3]. afterCompletion

#### 简单的谈一下SpringMVC的工作流程？

1、用户发送请求至前端控制器DispatcherServlet

2、DispatcherServlet收到请求调用HandlerMapping处理器映射器。

3、处理器映射器找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。

4、DispatcherServlet调用HandlerAdapter处理器适配器

5、HandlerAdapter经过适配调用具体的处理器(Controller，也叫后端控制器)。

6、Controller执行完成返回ModelAndView

7、HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet

8、DispatcherServlet将ModelAndView传给ViewReslover视图解析器

9、ViewReslover解析后返回具体View

10、DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图中）。

11、DispatcherServlet响应用户

#### Springmvc中如何解决POST请求中文乱码问题

Springmvc中通过CharacterEncodingFilter解决中文乱码问题.

在web.xml中加入：

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

#### MyBatis中 #{}和${}的区别是什么？

#{}是预编译处理，${}是字符串替换。

Mybatis在处理#{}时，会将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；

Mybatis在处理${}时，就是把${}替换成变量的值。

使用#{}可以有效的防止SQL注入，提高系统安全性。

#### Mybatis 结果集 的 映射方式有几种，并分别解释每种映射方式如何使用。

自动映射 ，通过resultType来指定要映射的类型即可。

自定义映射 通过resultMap来完成具体的映射规则，指定将结果集中的哪个列映射到对象的哪个属性。

#### 简述MyBatis的单个参数、多个参数如何传递及如何取值。

MyBatis传递单个参数，如果是普通类型(String+8个基本)的，取值时在#{}中可以任意指定，如果是对象类型的，则在#{}中使用对象的属性名来取值

MyBatis传递多个参数，默认情况下，MyBatis会对多个参数进行封装Map. 取值时

在#{}可以使用0 1 2 .. 或者是param1 param2..

MyBatis传递多个参数，建议使用命名参数，在Mapper接口的方法的形参前面使用

@Param() 来指定封装Map时用的key. 取值时在#{}中使用@Param指定的key.

#### MyBatis如何获取自动生成的(主)键值?

在<insert>标签中使用 useGeneratedKeys 和 keyProperty 两个属性来获取自动生成的主键值。

示例:

<insert id=”insertname” usegeneratedkeys=”true” keyproperty=”id”>

insert into names (name) values (#{name})

</insert>

#### 简述Mybatis的动态SQL，列出常用的6个标签及作用

动态SQL是MyBatis的强大特性之一 基于功能强大的OGNL表达式。

动态SQL主要是来解决查询条件不确定的情况，在程序运行期间，根据提交的条件动态的完成查询

常用的标签:

<if> : 进行条件的判断

<where>：在<if>判断后的SQL语句前面添加WHERE关键字，并处理SQL语句开始位置的AND 或者OR的问题

<trim>：可以在SQL语句前后进行添加指定字符 或者去掉指定字符.

<set>: 主要用于修改操作时出现的逗号问题

<choose> <when> <otherwise>：类似于java中的switch语句.在所有的条件中选择其一

<foreach>：迭代操作

#### Mybatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复？

不同的Xml映射文件，如果配置了namespace，那么id可以重复；如果没有配置namespace，那么id不能重复；

#### Mybatis 如何完成MySQL的批量操作,举例说明

MyBatis完成MySQL的批量操作主要是通过<foreache>标签来拼装相应的SQL语句.

例如:

<insert id="insertBatch" >

insert into tbl\_employee(last\_name,email,gender,d\_id) values

<foreach collection="emps" item="curr\_emp" separator=",">

(#{curr\_emp.lastName},#{curr\_emp.email},#{curr\_emp.gender},#{curr\_emp.dept.id})

</foreach>

</insert>

#### 简述Spring中如何给bean对象注入集合类型的属性。

Spring使用 <list> <set> <map> 等标签给对应类型的集合注入值

#### \*简述Spring 中bean的作用域

1、singleton 表示在spring容器中的单例，通过spring容器获得该bean时总是返回唯一的实例

2、prototype表示每次获得bean都会生成一个新的对象

3、request表示在一次http请求内有效（只适用于web应用）

4、session表示在一个用户会话内有效（只适用于web应用）

5、global session表示在全局会话内有效（只适用于web应用）

#### 简述Spring中自动装配常用的两种装配模式

byName: 根据bean对象的属性名 进行装配

byType： 根据bean对象的属性的类型进行装配,需要注意匹配到多个兼容类型的bean对象时，会抛出异常.

#### 简述动态代理的原理， 常用的动态代理的实现方式

动态代理的原理: **使用一个代理将对象包装起来**，然后用该代理对象取代原始对象。任何对原始对象的调用都要通过代理。

代理对象决定是否以及何时将方法调 用转到原始对象上

动态代理的方式

基于接口实现动态代理： JDK动态代理

基于继承实现动态代理： Cglib、Javassist动态代理

#### 请解释@Autowired注解的工作机制及required属性的作用

1. 首先会使用byType的方式进行自动装配，如果能唯一匹配，则装配成功，

如果匹配到多个兼容类型的bean, 还会尝试使用byName的方式进行唯一确定.

如果能唯一确定，则装配成功，如果不能唯一确定，则装配失败，抛出异常.

1. 默认情况下， 使用@Autowired标注的属性必须被装配，如果装配不了，也会抛出异常.

可以使用required=false来设置不是必须要被装配.

#### 请解释简述Springmvc中ContextLoaderListener的作用以及实现原理

1. 作用:ContextLoaderListener的作用是通过监听的方式在WEB应用服务器启动时将Spring的容器对象进行初始化.
2. 原理: ContextLoaderListener 实现了ServletContextListener接口，用于监听

ServletContext的创建，当监听到ServletContext创建时，在对应contextInitialized

方法中，将Spring的容器对象进行创建，并将创建好的容器对象设置到ServletContext域对象中，

目的是让各个组件可以通过ServletContext共享到Spring的容器对象

#### 简述Mybatis提供的两级缓存，以及缓存的查找顺序

MyBatis的缓存分为一级缓存和 二级缓存。

一级缓存是SqlSession级别的缓存，默认开启。

二级缓存是NameSpace级别(Mapper)的缓存，多个SqlSession可以共享，使用时需要进行配置开启。

缓存的查找顺序: 二级缓存 一级缓存 数据库

#### 简述Spring与Springmvc整合时，如何解决bean被创建两次的问题

Bean被创建两次的问题是在组建扫描的配置中指定Springmvc只负责扫描WEB相关的组件，Spring扫描除了Springmvc之外的组件。

#### 简述Spring与Mybatis整合时，主要整合的两个地方:

1. SqlSession创建的问题，通过SqlSessionFactoryBean来配置用于创建SqlSession的信息。例如: Mybatis的核心配置文件、Mapper映射文件、数据源等
2. Mapper接口创建的问题， 使用MapperScannerConfigurer批量为MyBatis的Mapper接口生成代理实现类并将具体的对象交给Spring容器管理

#### 简述Spring声明式事务中@Transaction中常用的两种事务传播行为

通过propagation来执行事务的传播行为

REQUIRED: 使用调用者的事务，如果调用者没有事务，则启动新的事务运行

REQUIRES\_NEW: 将调用者的事务挂起，开启新的事务运行。

#### 简述@RequestMapping注解的作用 可标注的位置 常用的属性

1. 该注解的作用是用来完成请求 与 请求处理方法的映射
2. 该注解可以标注在类上或者是方法上
3. 常用的属性:

value: 默认属性， 用于指定映射的请求URL

method: 指定映射的请求方式

params: 指定映射的请求参数

headers: 指定映射的请求头信息

#### 简述Springmvc中处理模型数据的两种方式

1. 使用ModelAndView 作为方法的返回值，将 模型数据 和 视图信息封装到ModelAndView中
2. 使用Map 或者是Model 作为方法的形参. 将模型数据添加到Map或者是Model中.

#### 简述REST中的四种请求方式及对应的操作

GET 查询操作

POST 添加操作

DELETE 删除操作

PUT 修改操作

#### 简述 视图 和 视图解析的关系及作用

1. 视图是由视图解析器解析得到的。
2. 视图解析器的作用是根据ModelAndView中的信息解析得到具体的视图对象
3. 视图的作用是完成模型数据的渲染工作，最终完成转发或者是重定向的操作

#### 说出三个常用的视图类

InternalResourceView

JstlView

RedirectView

#### 简述REST中 HiddenHttpMethodFilter过滤器的作用

该过滤器主要负责转换客户端请求的方式，当浏览器的请求方式为POST,并且在请求中能通过 \_method获取到请求参数值,

该过滤器就会进行请求方式的转换.

一般在REST中，都是将POST请求转换为对应的DELETE 或者是PUT

#### 简述Springmvc中如何返回JSON数据

1. 在工程最终加入jackson的jar包
2. 在请求处理方法中 将返回值改为具体返回的数据的类型， 例如 数据的集合类型 List<Employee>等
3. 在请求处理方法上使用@ResponseBody注解

#### 简述如何在myBatis中的增删改操作获取到对数据库的影响条数

直接在Mapper接口的方法中声明返回值即可.

#### Springmvc中的控制器的注解用哪个，可以是否用别的注解代替

使用@Controller注解来标注控制器。 不能使用别的注解代替

#### 如何在Springmvc中获取客户端提交的请求参数

直接在请求处理方法中声明对应的形参，也可以是用@RequestParam注解来具体指定将那些请求参数映射到方法中对应的形参。

#### 简述Springmvc中InternalResourceViewResolver解析器的工作机制

使用prefix + 方法的返回值 + suffix 生成一个物理视图路径。

#### Springmvc中如何完成重定向

在请求处理方法的返回值前面加 redirect: 前缀, 最终会解析得到RedirectView，RedirectView会完成重定向的操作.

#### 简述Spring中切面中常用的几种通知，并简单解释

前置通知 在目标方法执行之前执行

后置通知 在目标方法执行之后执行，不管目标方法有没有抛出异常

返回通知 在目标方法成功返回之后执行， 可以获取到目标方法的返回值.

异常通知 在目标方法抛出异常后执行

环绕通知 环绕着目标方法执行，

#### 解释MyBatis中 @Param注解的作用

通过该注解来指定Mybatis底层在处理参数时封装Map使用的key，方便在SQL映射文件中取参数.

#### 简述Mybatis中使用Mapper接口开发，如何完成Mapper接口与SQL映射文件、方法与SQL语句的绑定

Mapper接口与SQL映射文件绑定: SQL映射文件中的namespace的值指定成Mapper接口的全类名

接口中方法与SQL语句的绑定: SQL语句的id 指定成接口中的方法名.

#### SpringMVC 的工作原理

（1）用户向服务器发送请求，请求被springMVC 前端控制器 DispatchServlet 捕获；

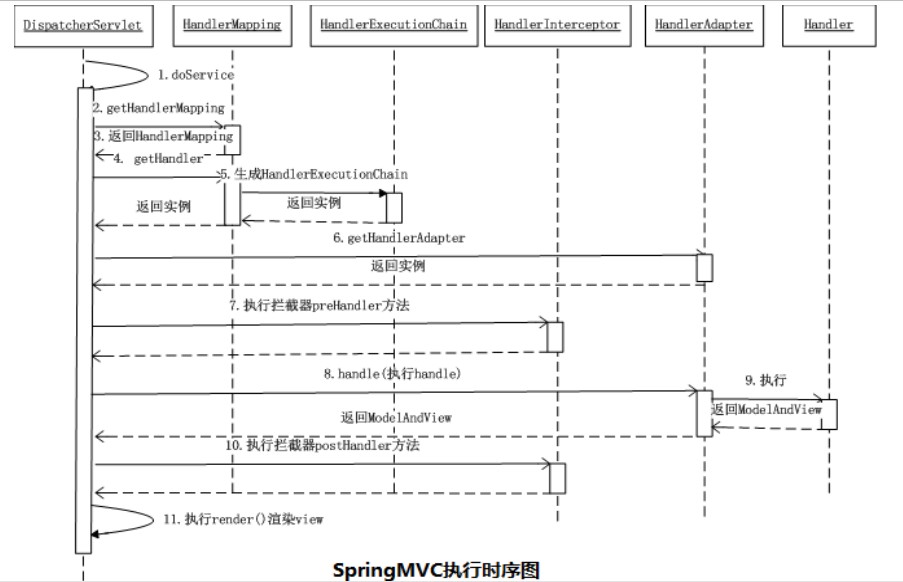
（2）DispatcherServle 对请求 URL 进行解析，得到请求资源标识符（URL），然后根据该 URL 调用 HandlerMapping将请求映射到处理器 HandlerExcutionChain；

（3）DispatchServlet 根据获得 Handler 选择一个合适的HandlerAdapter 适配器处理；

（4）Handler 对数据处理完成以后将返回一个 ModelAndView（）对象给 DisPatchServlet;

（5）Handler 返回的 ModelAndView() 只是一个逻辑视图并不是一个正式的视图， DispatcherSevlet 通过ViewResolver 试图解析器将逻辑视图转化为真正的视图View;

（6）DispatcherServle 通过 model 解析出 ModelAndView()中的参数进行解析最终展现出完整的 view 并返回给客户端;



#### 谈谈你对Spring 的理解

Spring 是一个开源框架，为简化企业级应用开发而生。Spring 可以是使简单的JavaBean 实现以前只有EJB 才能实现的功能。Spring 是一个 IOC 和 AOP 容器框架。

Spring 容器的主要核心是：

控制反转（IOC），传统的 java 开发模式中，当需要一个对象时，我们会自己使用 new 或者 getInstance 等直接或者间接调用构造方法创建一个对象。而在 spring 开发模式中，spring 容器使用了工厂模式为我们创建了所需要的对象，不需要我们自己创建了，直接调用spring 提供的对象就可以了，这是控制反转的思想。

依赖注入（DI），spring 使用 javaBean 对象的 set 方法或者带参数的构造方法为我们在创建所需对象时将其属性自动设置所需要的值的过程，就是依赖注入的思想。

面向切面编程（AOP），在面向对象编程（oop）思想中，我们将事物纵向抽成一个个的对象。而在面向切面编程中，我们将一个个的对象某些类似的方面横向抽成一个切面，对这个切面进行一些如权限控制、事物管理，记录日志等公用操作处理的过程就是面向切面编程的思想。AOP 底层是动态代理，如果是接口采用 JDK 动态代理，如果是类采用CGLIB 方式实现动态代理。

#### Spring中常用的设计模式

1. 代理模式——spring 中两种代理方式，若目标对象实现了若干接口，spring 使用jdk 的java.lang.reflect.Proxy类代理。若目标兑现没有实现任何接口，spring 使用 CGLIB 库生成目标类的子类。
2. 单例模式——在 spring 的配置文件中设置 bean 默认为单例模式。
3. 模板方式模式——用来解决代码重复的问题。

比如：RestTemplate、JmsTemplate、JpaTemplate

1. 工厂模式——在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用同一个接口来指向新创建的对象。Spring 中使用 beanFactory 来创建对象的实例。

#### 请描述一下Spring的事务管理

1、声明式事务管理的定义：用在 Spring 配置文件中声明式的处理事务来代替代码式的处理事务。这样的好处是，事务管理不侵入开发的组件，具体来说，业务逻辑对象就不会意识到正在事务管理之中，事实上也应该如此，因为事务管理是属于系统层面的服务，而不是业务逻辑的一部分，如果想要改变事务管理策划的话，也只需要在定义文件中重新配置即可，这样维护起来极其方便。

基于 TransactionInterceptor 的声明式事务管理：两个次要的属性： transactionManager，用来指定一个事务治理器， 并将具体事务相关的操作请托给它； 其他一个是 Properties 类型的transactionAttributes 属性，该属性的每一个键值对中，键指定的是方法名，方法名可以行使通配符， 而值就是表现呼应方法的所运用的事务属性。

1. 基于 @Transactional 的声明式事务管理：Spring 2.x 还引入了基于 Annotation 的体式格式，具体次要触及@Transactional 标注。@Transactional 可以浸染于接口、接口方法、类和类方法上。算作用于类上时，该类的一切public 方法将都具有该类型的事务属性。
2. 编程式事物管理的定义：在代码中显式挪用 beginTransaction()、commit()、rollback()等事务治理相关的方法， 这就是编程式事务管理。Spring 对事物的编程式管理有基于底层 API 的编程式管理和基于 TransactionTemplate 的编程式事务管理两种方式。

#### 谈谈 AOP

AOP，Aspect Oriented Programming 面向切面编程。可以使开发人员在不修改原来的代码的基础上，增加系统的新功能。

AOP  主要应用于日志记录，性能统计，安全控制，事务处理（项目中使用的）等方面。

通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。

#### 谈谈 IOC 和 DI

IOC是 Inverse Of Control 的简写，意思是控制反转。是降低对象之间的耦合关系的设计思想。

通过IOC，开发人员不需要关心对象的创建过程，交给Spring容器完成。具体的过程是，程序读取Spring 配置文件，获取需要创建的 bean 对象，通过反射机制创建对象的实例，并通过DI注入对象的依赖关系。

DI是 Dependency Injection依赖注入，说的是创建对象实例时，同时为这个对象注入它所依赖的属性。相当于把每个bean与bean之间的关系交给容器管理。而这个容器就是spring。例如我们通常在 Service 层 注入 它所依赖的 Dao 层的实例；在 Controller层注入 Service层的实例。

#### Spring的底层实现机制是什么？

使用Dom4j（解析XML）+ Java反射机制

Dom4j其实就是解析XML。使用反射机制实例化bean。

#### Spring配置文件applicationContext.xml可以修改名称吗？

可以修改，在web.xml文件中配置，如下：

  <context-param>

    <param-name>contextConfigLocation</param-name>

    <param-value>classpath:bean.xml</param-value>

  </context-param>

#### Spring配置文件中都有哪些配置

数据源dataSource，sessionFactoryBean，service层和dao层的注解扫描，事物，其它中间件对象的初始化例如jedis

#### Spring容器的bean是单例的吗

默认是单例的。但是如果配置了 scope = “prototype” ，则是多实例的。

scope 的 默认 值 是 "singleton"，表示单实例。

#### Spring MVC 的 Controller 是单例的吗？是线程安全的吗？

是单例的。是线程安全的。

原因：Spring MVC 虽然是单例的， 但是 Spring MVC 是方法级别的，数据传输是通过方法的参数传递的，因此在类中没有共享属性，不存在线程不安全的问题。

#### Spring 和 springMVC的区别？

Spring是一个 Java EE 框架，支持IOC、DI 和 AOP。Spring 框架有很多模块组成，每个模块可以独立运行。

Spring MVC 是 Spring的一个模块，实现MVC设计思想。

#### SpringMVC中都有哪些配置

视图解析器、放行静态资源、扫描controller、文件上传下载、json类型转换器、自定义拦截器、全局异常处理

#### Spring，SpringMVC ，mybatis  分别是解决什么问题的？

SpringMVC 是MVC框架，主要是将系统分为视图、模型、控制器三个层次，负责接收请求，调用业务层接口，给视图层返回需要的数据。降低系统的耦合性。

mybatis 是持久层框架，解决如何存取数据库中的数据的问题，提高数据操作的效率。

Spring 实现依赖注入的功能，例如将dao对象注入 service 层，同时 spring 面向切面的特性可以实现程序中事物管理的功能，日志记录的功能等。

#### MyBatis的三层架构

1、基础支撑层：负责读取mybatis配置文件，链接数据库，做事物管理

2、数据处理层：解析sql、执行sql、映射结果集

3、API接口层：提供操作数据的api

#### 56、写出MyBatis配置自定义映射使用的标签，并解释含义

1. 自定义resultMap，实现高级结果集映射
2. id ：用于完成主键值的映射
3. result ：用于完成普通列的映射
4. association ：一个复杂的类型关联;许多结果将包成这种类型
5. collection ： 复杂类型的集

#### 57、写出MyBatis一级缓存失效的几种情况

1. 不同的SqlSession对应不同的一级缓存
2. 同一个SqlSession但是查询条件不同
3. 同一个SqlSession两次查询期间执行了任何一次增删改操作
4. 同一个SqlSession两次查询期间手动清空了缓存