**尚硅谷——WEB+SSM题库**

#### 请写出 spring 中常用的依赖注入方式。

通过 setter 方法注入

通过构造方法注入

#### 简述Spring中如何基于注解配置Bean和装配Bean,

(1). 首先要在Spring中配置开启基于注解的组件扫描

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

(2). 在具体的类上加上具体的注解.@Controller、@Service、@Repository、@Component

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

#### 三、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 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

@CookieValue , @RequestParm , @ResponseBody , @RequestBody , @RestController

@Transactional @Before , @AfterReturning , @After , @AfterThrowing , @PointCut…

只要说出5个注解并解释含义即可，如上答案只做参考

#### 四、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. Destroy-method

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.

另外，bean默认是单例的。在IOC容器启动时，单例的这些实例会被创建。

我们可以使用scope=prototype来设置对象是多例的。多例的实例不会随着容器的创建而实例化。仅仅是在需要的时候才会创建。

#### 五、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. 需要注意的是这个过滤器一定要放在所有过滤器之前。这样保证request.setCharacterEncoding()先执行。

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>

#### 六、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。在某些情况下，我们只能使用${},例如: select \* from ${tablename} , selet…from t\_book order by ${columnName}

#### 七、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。

#### 八、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arg0 arg1 arg2 .. 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.

#### 九、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

#### 十、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.

#### 十一、 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。通过多态，父类类型指向子类对象，通过父类类型调用目标方法，实际上调用的是代理对象中的代理方法。从而达到增强原始目标方法功能的目的。

动态代理的方式

基于接口实现动态代理： JDK动态代理

基于字节码编辑技术实现动态代理： Cglib、Javassist动态代理

CGLIB和JDK代理的比较：

1. 性能上的比较：CGLIB创建代理对象的性能较差，调用代理方法的性能较高。JDK正好相反。因此，两者的适用场合不同：CGLIB适合于反复调用代理方法的场景。JDK适用于频繁创建代理对象的场景。只不过，目前JDK版本的发展，创建和调用代理方法的性能都得到了大幅度提升。因此，目前框架中也会优先考虑JDK代理（早期默认就是采取CGLIB代理）
2. 语法上的比较：CGLIB代理要求目标类不能被final修饰，也就意味着可被继承。因此CGLIB产生的代理对象是目标对象的子类。而JDK要求目标类必须要有父接口，因为JDK产生的代理对象是接口的实现类，是目标类的兄弟，这两者之间是没有继承关系的。
3. 在spring中AOP的底层就是使用的动态代理技术。换句话说，spring的AOP是对两种代理技术的封装，这样的话，程序员没有必要去接触原生的代理API代码。在spring的AOP中，目标对象称之为Target、目标对象中有目标方法。增强称之为Advice。增强又分为前置、后置返回、后置抛异常、后置和环绕。将增强操作追加到目标方法上的这个动作我们称之为织入（weave）。那么在哪些类的哪些方法上织入增强呢？我们称之为切入点（pointcut）。在一个方法的某一个连接点（joinpoint）上织入，这样的连接点会形成一个面，我们称之为切面（Aspect）。
4. 在springAOP中，底层使用哪一种代理技术，我们也有一个标签属性或者注解属性：proxy-target- class用来决策使用何种代理技术。

#### 十二、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，如果能找到并唯一，则装配成功。
2. 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. 当根据类型自动装配失败。根据名字又没有找到符合的。此时我们可以通过@qualifier来寻找指定名称的bean实例进行装配。
2. 如果@autowire注解中的required=false，表示能装就装。一般不设置，必须要装配。

#### 十三、简述Springmvc中ContextLoaderListener的作用以及实现原理

1. 语法：ContextLoaderListener实现了ServletContextListener接口。那么这个接口有两个方法，分别用于监听上下文对象的初始化和销毁的过程。那么此处我们关心的是初始化的过程。在tomcat启动时，servlet上下文对象被创建，那么ServletContextListener能够监听到上下文对象初始化的过程，此时contextInitialized方法被执行。那么我们会在这个方法中创建IOC容器（bean实例的创建，bean之间依赖关系的注入）

当IOC容器准备好之后，ContextLoaderListener会将IOC容器保存在application作用域，从而方便后面的获取。

1. 从分工上：SSM整合中，IOC容器有两个，一个是spring的IOC，一个是springmvc的IOC。前者是父容器，后者是子容器。子容器只维护springmvc相关的组件：控制器、异常映射器、视图解析器、国际化解析器。父容器维护除了springmvc中这些组件之外的所有的bean实例。包含数据访问层组件、业务逻辑层组件、消息（MQ）中间件等
2. ContextLoaderListener只负责加载springIOC容器。而springmvc的IOC容器是由中央控制器中去处理的。我们知道在javaee规范中，listener是先执行的，servlet默认是访问时才会去执行的。因此中央控制器DispatcherServlet中配置了<load-on-startup>从而提前它的创建过程。但是再怎么提前，也在listener之后。因此，先创建spring的IOC容器，然后才会去创建springmvc的IOC容器。在DispatcherServlet中，首先寻找根容器（其实就是spring的IOC容器），如果能够找到，则通过setParent进行设置他们之间的父子关系。

#### 十四、简述Mybatis提供的两级缓存，以及缓存的查找顺序

1. MyBatis中分为一级缓存（SqlSession级别）和二级缓存（SqlSessionFactory级别）。
2. 一级缓存默认是开启的。
3. 一级缓存在何种情况下失效：两次查询条件不同；两次查询之间有增删改操作
4. 二级缓存默认是不开启的，我们通过<cache/>开启二级缓存。开启的是mybatis自带的二级缓存。实际开发时，也会使用第三方缓存中间件，例如Ehcache。
5. 查询出的对象何时存入二级缓存呢？必须是sqlSession关闭之后，那么查询出的对象才会存入二级缓存。（第一个sqlSession关闭之后，第二个sqlSession去查询就不会发出SQL语句了）
6. 缓存的查找顺序：先查找二级缓存，如果没有命中，查找一级缓存，如果还没有命中，则查询数据库。

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

两种解决方法：

1. 配置扫描的包路径不同：

Springmvc扫描:com.atguigu.book.controller .

spring扫描: com.atguigu.book.service,com.atguigu.book.mapper,….可以同时添加多个扫描的包路径

1. 配合使用exclude-filter和include-filter

#### 十六、简述Spring声明式事务中@Transactional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。

SUPPORTS

MANDATORY

NEVER

NOT\_SUPPORT

NESTED

另外，隔离级别也是高频次面试题。

事务的四大特性：ACID。原子性，一致性，隔离性，永久性

数据库的三大并发问题：脏读、不可重复读、幻读

脏读：读取到别人将来要回滚的数据。

不可重复读：两次读取的都是别人确认过的（提交的）数据，但是两次读取的数据不一致。

幻读：记录的条数一会儿多一会儿少

为了解决数据库的三大并发问题，数据库提供了四大隔离级别：读未提交、读已提交、可重复读、串行化。

读未提交：可以读取到别人还没提交的数据。那么可能会出现脏读

读已提交：读取的一定是别人提交的。那么可以避免脏读，但是会出现不可重复读

可重复读：可以避免脏读和不可重复读。但是无法避免幻读。Mysql默认是这个级别，通过特定的机制也可以避免幻读。

串行化：可以避免脏读、不可重复读、幻读问题。但是效率极其低下，因为已经不存在并发了。

#### 十七、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Model或ModelMap 作为方法的形参. 将模型数据添加到Map或者是Model中.

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

1)为什么需要这个过滤器：正常的HTTP请求，只有get和post两种。但是我们还需要PUT和DELETE请求。这样不同的请求方式代表不同的操作。

2)这个过滤器如何使用：

* 表单一定是post请求
* 发送的请求参数中一定有一个name是”\_method”（可以通过配置修改，但一般不修改）
* \_method可以指定：put , delete . 大小写不敏感

3) 原理：这个过滤器的内部有一个内部类，继承了request的真实实现类，然后覆盖了它中的method属性以及重写了getMethod方法。

#### 十九、简述Springmvc中如何返回JSON数 据

1. 在工程中加入fast-json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。如果我们采取的前后端分离的方式，也就意味着所有的控制器方法都是给前端返回json数据，那么我们可以将控制器使用@RestController进行修饰。

#### 二十、 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.

* 重定向：return “redirect:/index.html” -> 重定向到/index.html
* 转发： return “forward:/index.html” -> 转发到/index.html，注意，没有视图渲染
* 返回逻辑视图名称return “index” -> 转发到index对应的物理视图，注意，有视图渲染

#### 二十一、SpringMVC 的工作原理

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

（2）DispatcherServlet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；在请求处理之前，会进行数据绑定，在请求处理之后会进行视图处理。

（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物理视图， DispatcherSevlet 通过ViewResolver 视图解析器将逻辑视图转化为物理视图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;

#### 二十二、 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(静态工厂方法模式、工厂方法模式、抽象工厂模式)——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。

#### 二十三、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是BeanFactory的子接口, 提供了更多高级的特性.面向开发者

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。

#### 二十四、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：拦截器往往会出现多个，我们称之为拦截器链。如果preHandle方法返回true，则表示继续往下传递，如果返回false，则表示不传递，那么会调用afterCompletion。调用的顺序和配置的顺序一致。

[2]. postHandle：调用的顺序和配置的顺序相反 for(int i =interceptors.size()-1 ; i--)

[3]. afterCompletion : 调用的顺序和配置的顺序相反

#### 二十五、MyBatis如何获取自动生成的(主)键值?

在<insert>标签中使用

* useGeneratedKeys=true
* keyProperty=bid

示例:

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

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

</insert>

#### 二十六、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><array><props> 等标签给对应类型的集合注入值

#### 二十七、简述Spring 中bean的作用域

总共有四种作用域:

1. Singleton 单例的 . 在IOC容器创建的时候，bean实例就会被创建
2. Prototype 原型的 。 在主动获取时才会被创建。
3. Request
4. Session

#### 二十八、简述Spring与Mybatis整合时，主要整合的两个地方:

1. 对SqlSession的管理（或者是SqlSessionFactory的管理），通过SqlSessionFactoryBean来配置用于创建SqlSessionFactory的信息。

例如: Mybatis的核心配置文件、Mapper映射文件、数据源等

<bean id="sqlSessionFactory" class="……SqlSessionFactoryBean”>

1. 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

<mybatis-spring:scan base-package="com.atguigu.ssm.mapper"/>

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

1. 该注解是用来完成一个url和具体的某一个控制器的某一个方法对应的过程。

(MappingRegistry)

1. 该注解可以标注在类上或者是方法上
2. 常用的属性:

value: 默认属性， 用于指定映射URL；method: 指定请求方式;params: 指定请求参数;headers: 指定映射的请求头信息

#### 三十、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

REST风格是面向资源的，而不是面向功能的 <http://localhost/book/1>

传统的url是面向功能的，<http://localhost/book/getBook?id=1>

REST操作满足幂等性。幂等性表示一次操作和重复多次操作对系统的影响是相同的，则表示符合幂等性。

#### 三十一、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.

#### 三十二、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

#### 三十三、 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

1. SQL映射文件中的namespace的值指定成Mapper接口的全类名
2. SQL语句的id 指定成接口中的方法名.

#### 三十四、请描述一下Spring的事务管理

1. 声明式事务管理的定义：采用配置文件或者注解的方式来进行事务管理。从而代替编程式事务管理。
2. 这样的好处是，事务管理的代码不侵入到业务逻辑代码中
3. 事务管理的接口是TransactionManager。

集成Mybatis进行事务管理，我们使用的是它的子类：DataSourceTransactionManager。

另外，还有HibernateTransactionmanager，还有JPATransactionManager，还有分布式事务管理器-JTATransactionManager。

1. TransactionManager中常用的四个方法：提交、回滚、挂起、回复
2. 事务管理可以配置XML，也可以使用注解。我们使用@Transactional来修饰业务类或者业务方法。表示具备默认的事务管理策略（Required、可重复读）
3. 另外，事务除了隔离级别和传播机制之外，还可以设置一些常规属性：只读、超时、回滚和不回滚事务

#### 三十五、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。