**尚硅谷——SSM阶段考题（二）及答案**

#### 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。

#### 二、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

#### 三、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.

#### 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

[2]. postHandle

[3]. afterCompletion

#### 五、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

#### 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。

#### 七、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.

#### 八、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>

#### 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

#### 十一、简述Spring 中bean的作用域

总共有四种作用域:

1. Singleton 单例的
2. Prototype 原型的
3. Request
4. Session

#### 十二、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

#### 十三、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

#### 十四、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

#### 十五、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。

#### 十六、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

#### 十七、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

#### 十八、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.

#### 十九、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

#### 二十、 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.

#### 二十一、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。

#### 二十二、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。