**spring和springMVC的面试问题总结**

2017年08月30日 10:44:25

阅读数：47794

**1.Spring中AOP的应用场景、Aop原理、好处？**

答：AOP--Aspect Oriented Programming面向切面编程；用来封装横切关注点，具体可以在下面的场景中使用:

Authentication 权限、Caching 缓存、Context passing 内容传递、Error handling 错误处理Lazy loading懒加载、Debugging调试、logging, tracing, profiling and monitoring 记录跟踪优化　校准、Performance optimization　性能优化、Persistence 持久化、Resource pooling　资源池、Synchronization　同步、Transactions 事务

**原理：**AOP是面向切面编程，是通过动态代理的方式为程序添加统一功能，集中解决一些公共问题。

**优点：**1.各个步骤之间的良好隔离性耦合性大大降低   
    2.源代码无关性，再扩展功能的同时不对源码进行修改操作   
 3.非侵入式地为应用添加非业务逻辑功能

**2.Spring中IOC的作用与原理？对象创建的过程。**

答：IOC--Inversion of Control控制反转。当某个角色需要另外一个角色协助的时候，在传统的程序设计过程中，通常由调用者来创建被调用者的实例对象。但在spring中创建被调用者的工作不再由调用者来完成，因此称为控制反转。创建被调用者的工作由spring来完成，然后注入调用者 直接使用。

什么是依赖注入：

一个对象由另一个对象组成，对象自己不去创建其依赖的对象，而是在对象创建时从外部传入其依赖的对象，此过程叫依赖注入DI是IOC的一种，在spring中，该过程由spring容器来完成，

依赖注入好处：

降低业务对象替换的复杂性，方便将不同的对象组装起来

**3.介绍spring框架**

   它是一个一站式（full-stack全栈式）框架，提供了从表现层-springMVC到业务层-spring再到持久层-springdata的一套完整的解决方案。我们在项目中可以只使用spring一个框架，它就可以提供表现层的mvc框架，持久层的Dao框架。它的两大核心IoC和AOP更是为我们程序解耦和代码简洁易维护提供了支持。  
Spring是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架

4.Spring**常见创建对象的注解？**

答：@Component@Controller@ Service@ Repository

5.Sp**ring中用到的设计模式**

答：简单工厂、工厂方法、单例模式、适配器、包装器、代理、观察者、策略、模板方法

详细介绍:请参考本微博的：开发常用设计模式

6.S**pring的优点？**

答：1.降低了组件之间的耦合性 ，实现了软件各层之间的解耦   
2.可以使用容易提供的众多服务，如事务管理，消息服务等   
3.容器提供单例模式支持   
4.容器提供了AOP技术，利用它很容易实现如权限拦截，运行期监控等功能   
5.容器提供了众多的辅助类，能加快应用的开发   
6.spring对于主流的应用框架提供了集成支持，如hibernate，JPA，Struts等   
7.spring属于低侵入式设计，代码的污染极低   
8.独立于各种应用服务器   
9.spring的DI机制降低了业务对象替换的复杂，性有助于您替换对象   
10.Spring的高度开放性，并不强制应用完全依赖于Spring，开发者可以自由选择spring 的部分或全部   
11.Spring能有效地组织你的中间层对象

7.S**pring Bean的作用域之间有什么区别？**

Spring容器中的bean可以分为5个作用域。所有范围的名称都是自说明的，但是为了避免混淆，还是让我们来解释一下：

singleton：这种bean范围是默认的，这种范围确保不管接受到多少个请求，每个容器中只有一个bean的实例，单例的模式由bean factory自身来维护。

prototype：原形范围与单例范围相反，为每一个bean请求提供一个实例。

request：在请求bean范围内会每一个来自客户端的网络请求创建一个实例，在请求完成以后，bean会失效并被垃圾回收器回收。

Session：与请求范围类似，确保每个session中有一个bean的实例，在session过期后，bean会随之失效。

global-session：global-session和Portlet应用相关。当你的应用部署在Portlet容器中工作时，它包含很多portlet。如果你想要声明让所有的portlet共用全局的存储变量的话，那么这全局变量需要存储在global-session中。

全局作用域与Servlet中的session作用域效果相同。

8.S**pring管理事务有几种方式？**

答：有两种方式：

1、编程式事务，在代码中硬编码。(不推荐使用)

2、声明式事务，在配置文件中配置（推荐使用）

声明式事务又分为两种：

a、基于XML的声明式事务  
 <tx:advice id=“tmd” transaction-manager=“transactionManager”>  
 用trsactionManager定义一个通知  
 <aop:config>  
 <aop:poincut id=“pc”>  
 <aop:advisor advisor-ref=“tmd” poincut-ref=“pc”>   
 将通知与切点关联

b、基于注解的声明式事务  
 <tx:annotation-dirver transaction-manager=“transactionManager”>  
 用trsactionManager定义一个通知，并用过注解的方式将通知应用到切点  
 @Transaction

9.s**pring中自动装配的方式有哪些？**

答：1、 No：即不启用自动装配。

2、 byName：通过属性的名字的方式查找JavaBean依赖的对象并为其注入。比如说类Computer有个属性printer，指定其autowire属性为byName后，Spring IoC容器会在配置文件中查找id属性为printer的bean，然后使用Seter方法为其注入。

3、 byType：通过属性的类型查找JavaBean依赖的对象并为其注入。比如类Computer有个属性printer，类型为Printer，那么，指定其autowire属性为byType后，Spring IoC容器会查找Class属性为Printer的bean，使用Seter方法为其注入。

4、 constructor：通byType一样，也是通过类型查找依赖对象。与byType的区别在于它不是使用Seter方法注入，而是使用构造子注入。

5、 autodetect：在byType和constructor之间自动的选择注入方式。

6、 default：由上级标签<beans>的default-autowire属性确定。

10.s**pring中的核心类有那些，各有什么作用？**

答：BeanFactory：产生一个新的实例，可以实现单例模式，懒加载

BeanWrapper：提供统一的get及set方法

ApplicationContext:提供框架的实现，包括BeanFactory的所有功能，急加载

11.B**ean的调用方式有哪些？**

答：有三种方式可以得到Bean并进行调用：   
1、使用BeanWrapper   
HelloWorld hw=new HelloWorld();   
BeanWrapper bw=new BeanWrapperImpl(hw);   
bw.setPropertyvalue(”msg”,”HelloWorld”);   
system.out.println(bw.getPropertyCalue(”msg”));   
2、使用BeanFactory   
InputStream is=new FileInputStream(”config.xml”);   
XmlBeanFactory factory=new XmlBeanFactory(is);   
HelloWorld hw=(HelloWorld) factory.getBean(”HelloWorld”);   
system.out.println(hw.getMsg());   
3、使用ApplicationConttext   
ApplicationContext actx=new FleSystemXmlApplicationContext(”config.xml”);   
HelloWorld hw=(HelloWorld) actx.getBean(”HelloWorld”);   
System.out.println(hw.getMsg());

12.什**么是IOC，什么又是DI，他们有什么区别？**

答：依赖注入DI是一个程序设计模式和架构模型， 一些时候也称作控制反转，尽管在技术上来讲，依赖注入是一个IOC的特殊实现，依赖注入是指一个对象应用另外一个对象来提供一个特殊的能力，例如：把一个 数据库连接已参数的形式传到一个对象的结构方法里面而不是在那个对象内部自行创建一个连接。控制反转和依赖注入的基本思想就是把类的依赖从类内部转化到外 部以减少依赖

应用控制反转，对象在被创建的时候，由一个调控系统内所有对象的外界实体，将其所依赖的对象的引用，传递给它。也可以说，依赖被注入到对象中。所 以，控制反转是，关于一个对象如何获取他所依赖的对象的引用，这个责任的反转。

13.s**pring有两种代理方式：**

答: 若目标对象实现了若干接口，spring使用JDK的java.lang.reflect.Proxy类代理。

优点：因为有接口，所以使系统更加松耦合

缺点：为每一个目标类创建接口

若目标对象没有实现任何接口，spring使用CGLIB库生成目标对象的子类。

优点：因为代理类与目标类是继承关系，所以不需要有接口的存在。

缺点：因为没有使用接口，所以系统的耦合性没有使用JDK的动态代理好。

14.s**pringMVC的流程？**

答：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.处理器映射器根据请求url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通过HandlerAdapter处理器适配器调用处理器

5.执行处理器(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对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

15.S**pringmvc的优点**

答：1.它是基于组件技术的.全部的应用对象,无论控制器和视图,还是业务对象之类的都是 java组件.并且和Spring提供的其他基础结构紧密集成.

2.不依赖于Servlet API(目标虽是如此,但是在实现的时候确实是依赖于Servlet的)

3. 可以任意使用各种视图技术,而不仅仅局限于JSP

4 . 支持各种请求资源的映射策略

5 .它应是易于扩展的