**Spring面试题答案：**

**1. 使用Spring框架的好处是什么？**

* **轻量：**Spring 是轻量的，基本的版本大约2MB。
* **控制反转：**Spring通过控制反转实现了松散耦合，对象们给出它们的依赖，而不是创建或查找依赖的对象们。
* **面向切面的编程(AOP)：**Spring支持面向切面的编程，并且把应用业务逻辑和系统服务分开。
* **容器：**Spring 包含并管理应用中对象的生命周期和配置。
* **MVC框架**：Spring的WEB框架是个精心设计的框架，是Web框架的一个很好的替代品。
* **事务管理：**Spring 提供一个持续的事务管理接口，可以扩展到上至本地事务下至全局事务（JTA）。
* **异常处理：**Spring 提供方便的API把具体技术相关的异常（比如由JDBC，Hibernate or JDO抛出的）转化为一致的unchecked 异常。

**2、Spring有几种配置方式？**

将Spring配置到应用开发中有以下三种方式：

基于XML的配置  
基于注解的配置  
基于Java的配置

**3、如何用基于Java配置的方式配置Spring？**

            Spring对Java配置的支持是由**@Configuration注解和@Bean**注解来实现的。由@Bean注解的方法将会实例化、配置和初始化一个新对象，这个对象将由Spring的IoC容器来管理。**@Bean声明所起到的作用与<bean/> 元素类似**。被**@Configuration所注解的类则表示这个类的主要目的是作为bean定义的资源。**被@Configuration声明的类可以通过在同一个类的内部调用@bean方法来设置嵌入bean的依赖关系。

最简单的@Configuration 声明类请参考下面的代码：

**[java]** [view plain](http://blog.csdn.net/u011225629/article/details/48010643) [copy](http://blog.csdn.net/u011225629/article/details/48010643" \o "copy)

1. @Configuration
2. **public** **class** AppConfig
3. {
4. @Bean
5. **public** MyService myService() {
6. **return** **new** MyServiceImpl();
7. }
8. }

对于上面的@Beans配置文件相同的XML配置文件如下：

**[xml]** [view plain](http://blog.csdn.net/u011225629/article/details/48010643) [copy](http://blog.csdn.net/u011225629/article/details/48010643)

1. **<beans>**
2. **<bean** id="myService" class="com.howtodoinjava.services.MyServiceImpl"**/>**
3. **</beans>**

上述配置方式的实例化方式如下：利用**AnnotationConfigApplicationContext** 类进行实例化

**[java]** [view plain](http://blog.csdn.net/u011225629/article/details/48010643) [copy](http://blog.csdn.net/u011225629/article/details/48010643)

1. **public** **static** **void** main(String[] args) {
2. ApplicationContext ctx = **new** AnnotationConfigApplicationContext(AppConfig.**class**);
3. MyService myService = ctx.getBean(MyService.**class**);
4. myService.doStuff();
5. }

要使用组件组建扫描，仅需用@Configuration进行注解即可：

**[java]** [view plain](http://blog.csdn.net/u011225629/article/details/48010643) [copy](http://blog.csdn.net/u011225629/article/details/48010643)

1. @Configuration
2. @ComponentScan(basePackages = "com.howtodoinjava")
3. **public** **class** AppConfig  {
4. ...
5. }

在上面的例子中，com.acme包首先会被扫到，然后再容器内查找被@Component 声明的类，找到后将这些类按照Sring bean定义进行注册。

如果你要在你的web应用开发中选用上述的配置的方式的话，需要用AnnotationConfigWebApplicationContext 类来读取配置文件，可以用来配置Spring的Servlet监听器ContrextLoaderListener或者Spring MVC的DispatcherServlet。

**[xml]** [view plain](http://blog.csdn.net/u011225629/article/details/48010643) [copy](http://blog.csdn.net/u011225629/article/details/48010643)

1. **<web-app>**
2. <!-- Configure ContextLoaderListener to use AnnotationConfigWebApplicationContext
3. instead of the default XmlWebApplicationContext --**>**
4. **<context-param>**
5. **<param-name>**contextClass**</param-name>**
6. **<param-value>**
7. org.springframework.web.context.support.AnnotationConfigWebApplicationContext
8. **</param-value>**
9. **</context-param>**
11. <!-- Configuration locations must consist of one or more comma- or space-delimited
12. fully-qualified @Configuration classes. Fully-qualified packages may also be
13. specified for component-scanning --**>**
14. **<context-param>**
15. **<param-name>**contextConfigLocation**</param-name>**
16. **<param-value>**com.howtodoinjava.AppConfig**</param-value>**
17. **</context-param>**
19. <!-- Bootstrap the root application context as usual using ContextLoaderListener -->
20. **<listener>**
21. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
22. **</listener>**
24. <!-- Declare a Spring MVC DispatcherServlet as usual -->
25. **<servlet>**
26. **<servlet-name>**dispatcher**</servlet-name>**
27. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
28. <!-- Configure DispatcherServlet to use AnnotationConfigWebApplicationContext
29. instead of the default XmlWebApplicationContext --**>**
30. **<init-param>**
31. **<param-name>**contextClass**</param-name>**
32. **<param-value>**
33. org.springframework.web.context.support.AnnotationConfigWebApplicationContext
34. **</param-value>**
35. **</init-param>**
36. <!-- Again, config locations must consist of one or more comma- or space-delimited
37. and fully-qualified @Configuration classes --**>**
38. **<init-param>**
39. **<param-name>**contextConfigLocation**</param-name>**
40. **<param-value>**com.howtodoinjava.web.MvcConfig**</param-value>**
41. **</init-param>**
42. **</servlet>**
44. <!-- map all requests for /app/\* to the dispatcher servlet -->
45. **<servlet-mapping>**
46. **<servlet-name>**dispatcher**</servlet-name>**
47. **<url-pattern>**/app/\***</url-pattern>**
48. **</servlet-mapping>**
49. **</web-app>**