目录

[1.什么是Spring？ 2](#_Toc482089612)

[2.Spring有哪些优点？ 2](#_Toc482089613)

[3.Spring框架有哪些模块？ 2](#_Toc482089614)

[4.解释核心容器(应用上下文)模块 3](#_Toc482089615)

[5.BeanFactory – BeanFactory 实例 3](#_Toc482089616)

[6.XmlBeanFactory 3](#_Toc482089617)

[7.解释AOP模块 3](#_Toc482089618)

[8.解释抽象JDBC和DAO模块 4](#_Toc482089619)

[9.解释对象/关系映射集成模块 4](#_Toc482089620)

[10.解释web模块 4](#_Toc482089621)

[11.解释Spring MVC模块 4](#_Toc482089622)

[12.Spring的配置文件 4](#_Toc482089623)

[13.Spring IoC容器是什么？ 4](#_Toc482089624)

[14.IOC有什么优点？ 5](#_Toc482089625)

[15.应用上下文是如何实现的？ 5](#_Toc482089626)

[16.Bean Factory和ApplicationContext有什么区别？ 5](#_Toc482089627)

[17.Spring应用程序看起来像什么？ 5](#_Toc482089628)

[18.Spring中的依赖注入是什么？ 6](#_Toc482089629)

[19.有哪些不同类型的IOC(依赖注入)？ 6](#_Toc482089630)

[20.你推荐哪种依赖注入？构造器依赖注入还是Setter方法依赖注入？ 6](#_Toc482089631)

[21.什么是Spring Beans？ 6](#_Toc482089632)

[22.Spring Bean中定义了什么内容？ 7](#_Toc482089633)

[23.如何向Spring 容器提供配置元数据？ 7](#_Toc482089634)

[24.你如何定义bean的作用域？ 7](#_Toc482089635)

[25.说一下Spring中支持的bean作用域 7](#_Toc482089636)

[26.Spring框架中单例beans是线程安全的吗？ 8](#_Toc482089637)

[27.解释Spring框架中bean的生命周期 8](#_Toc482089638)

[28.哪些是最重要的bean生命周期方法？能重写它们吗？ 8](#_Toc482089639)

[29.什么是Spring的内部bean？ 9](#_Toc482089640)

[30.如何在Spring中注入Java集合类？ 9](#_Toc482089641)

[31.什么是bean wiring？ 9](#_Toc482089642)

[32.什么是bean自动装配？ 9](#_Toc482089643)

[33.解释自动装配的各种模式？ 9](#_Toc482089644)

[34.自动装配有哪些局限性？ 10](#_Toc482089645)

[35.你可以在Spring中注入null或空字符串吗？ 10](#_Toc482089646)

[36.什么是Spring基于Java的配置？给出一些注解的例子 10](#_Toc482089647)

[37.什么是基于注解的容器配置? 11](#_Toc482089648)

[38.如何开启注解装配？ 11](#_Toc482089649)

[39.@Required 注解 11](#_Toc482089650)

[40.@Autowired 注解 11](#_Toc482089651)

[41. @Qualifier 注解 11](#_Toc482089652)

[42.在Spring框架中如何更有效的使用JDBC？ 11](#_Toc482089653)

[43.JdbcTemplate 12](#_Toc482089654)

[44.Spring对DAO的支持 12](#_Toc482089655)

[45.使用Spring可以通过什么方式访问Hibernate？ 12](#_Toc482089656)

[46.Spring支持的ORM 12](#_Toc482089657)

[47.如何通过HibernateDaoSupport将Spring和Hibernate结合起来？ 12](#_Toc482089658)

[48.Spring支持的事务管理类型 13](#_Toc482089659)

[49.Spring框架的事务管理有哪些优点？ 13](#_Toc482089660)

[50.你更推荐那种类型的事务管理？ 13](#_Toc482089661)

[51.解释AOP 14](#_Toc482089662)

[52.切面(Aspect) 14](#_Toc482089663)

[53.在Spring AOP中concern和 cross-cutting concern的区别是什么？ 14](#_Toc482089664)

[54.连接点(Join point) 14](#_Toc482089665)

[55.通知(Advice) 14](#_Toc482089666)

[56.切入点(Pointcut) 15](#_Toc482089667)

[57.什么是引入？ 15](#_Toc482089668)

[58.什么是目标对象？ 15](#_Toc482089669)

[59.什么是代理？ 15](#_Toc482089670)

[60.有几种不同类型的自动代理？ 15](#_Toc482089671)

[61.什么是织入？什么是织入应用的不同点？ 15](#_Toc482089672)

[62.解释基于XML Schema方式的切面实现 16](#_Toc482089673)

[63.解释基于注解方式(基于@AspectJ)的切面实现 16](#_Toc482089674)

[64.什么是Spring的MVC框架？ 16](#_Toc482089675)

[65.DispatcherServlet 16](#_Toc482089676)

[66.WebApplicationContext 16](#_Toc482089677)

[67.什么是Spring MVC框架的控制器？ 16](#_Toc482089678)

[68.@Controller annotation 16](#_Toc482089679)

[69.@RequestMapping annotation 17](#_Toc482089680)

## **1.什么是Spring？**

Spring是一个开源的Java EE开发框架。Spring框架的核心功能可以应用在任何Java应用程序中，但对Java EE平台上的Web应用程序有更好的扩展性。Spring框架的目标是使得Java EE应用程序的开发更加简捷，通过使用POJO为基础的编程模型促进良好的编程风格。

## **2.Spring有哪些优点？**

* **轻量级：**Spring在大小和透明性方面绝对属于轻量级的，基础版本的Spring框架大约只有2MB。
* **控制反转(IOC)：**Spring使用[控制反转技术](http://www.javacodegeeks.com/2011/08/what-is-dependency-inversion-is-it-ioc.html)实现了松耦合。依赖被注入到对象，而不是创建或寻找依赖对象。
* **面向切面编程(AOP)：** [Spring支持面向切面编程](http://www.javacodegeeks.com/2011/01/aspect-oriented-programming-spring-aop.html)，同时把应用的业务逻辑与系统的服务分离开来。
* **容器：**Spring包含并管理应用程序对象的配置及生命周期。
* **MVC框架：**Spring的web框架是一个设计优良的[web MVC框架](http://www.javacodegeeks.com/2011/02/spring-mvc-development-tutorial.html)，很好的取代了一些web框架。
* **事务管理：**Spring对下至本地业务上至全局业务(JAT)提供了统一的事务管理接口。
* **异常处理：**Spring提供一个方便的API将特定技术的异常(由JDBC, Hibernate, 或JDO抛出)转化为一致的、Unchecked异常。

## **3.Spring框架有哪些模块？**

Spring框架的基本模块如下所示：

* Core module
* Bean module
* Context module
* Expression Language module
* JDBC module
* ORM module
* OXM module
* Java Messaging Service(JMS) module
* Transaction module
* Web module
* Web-Servlet module
* Web-Struts module
* Web-Portlet module

## **4.解释核心容器(应用上下文)模块**

这是Spring的基本模块，它提供了Spring框架的基本功能。BeanFactory 是所有Spring应用的核心。Spring框架是建立在这个模块之上的，这也使得Spring成为一个容器。

## **5.BeanFactory – BeanFactory 实例**

BeanFactory是工厂模式的一种实现，它使用控制反转将应用的配置和依赖与实际的应用代码分离开来。

最常用的BeanFactory实现是XmlBeanFactory类。

## **6.XmlBeanFactory**

最常用的就是org.springframework.beans.factory.xml.XmlBeanFactory，它根据XML文件中定义的内容加载beans。该容器从XML文件中读取配置元数据，并用它来创建一个完备的系统或应用。

## **7.解释AOP模块**

AOP模块用来开发Spring应用程序中具有切面性质的部分。该模块的大部分服务由AOP Aliance提供，这就保证了Spring框架和其他AOP框架之间的互操作性。另外，该模块将元数据编程引入到了Spring。

## **8.解释抽象JDBC和DAO模块**

通过使用[抽象JDBC和DAO模块](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/)保证了与数据库连接代码的整洁与简单，同时避免了由于未能关闭数据库资源引起的问题。它在多种数据库服务器的错误信息之上提供了一个很重要的异常层。它还利用Spring的AOP模块为Spring应用程序中的对象提供事务管理服务。

## **9.解释对象/关系映射集成模块**

Spring通过提供ORM模块在JDBC的基础上支持对象关系映射工具。这样的支持使得Spring可以集成主流的ORM框架，包括Hibernate, JDO, 及iBATIS SQL Maps。Spring的事务管理可以同时支持以上某种框架和JDBC。

## **10.解释web模块**

[Spring的web模块](http://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-hello-world-example/)建立在应用上下文(application context)模块之上，提供了一个适合基于web应用程序的上下文环境。该模块还支持了几个面向web的任务，如透明的处理多文件上传请求及将请求参数同业务对象绑定起来。

## **11.解释Spring MVC模块**

Spring提供MVC框架构建web应用程序。Spring可以很轻松的同其他MVC框架结合，但[Spring的MVC](http://www.javacodegeeks.com/2012/09/spring-adding-spring-mvc-part-1.html)是个更好的选择，因为它通过控制反转将控制逻辑和业务对象完全分离开来。

## **12.Spring的配置文件**

Spring的配置文件是一个XML文件，文件包含了类信息并描述了这些类是如何配置和互相调用的。

## **13.Spring IoC容器是什么？**

Spring IOC负责创建对象、管理对象(通过依赖注入)、整合对象、配置对象以及管理这些对象的生命周期。

## **14.IOC有什么优点？**

IOC或依赖注入减少了应用程序的代码量。它使得应用程序的测试很简单，因为在单元测试中不再需要单例或JNDI查找机制。简单的实现以及较少的干扰机制使得松耦合得以实现。IOC容器支持勤性单例及延迟加载服务。

## **15.应用上下文是如何实现的？**

**FileSystemXmlApplicationContext** 容器加载XML文件中beans的定义。XML Bean配置文件的完整路径必须传递给构造器。

**FileSystemXmlApplicationContext** 容器也加载XML文件中beans的定义。注意，你需要正确的设置CLASSPATH，因为该容器会在CLASSPATH中查看bean的XML配置文件。

**WebXmlApplicationContext：**该容器加载xml文件，这些文件定义了web应用中所有的beans。

## **16.Bean Factory和ApplicationContext有什么区别？**

ApplicationContext提供了一种解决文档信息的方法，一种加载文件资源的方式(如图片)，他们可以向监听他们的beans发送消息。另外，容器或者容器中beans的操作，这些必须以bean工厂的编程方式处理的操作可以在应用上下文中以声明的方式处理。应用上下文实现了MessageSource，该接口用于获取本地消息，实际的实现是可选的。

## **17.Spring应用程序看起来像什么？**

* 一个定义功能的接口
* 实现包括属性，setter和getter方法，功能等
* Spring AOP
* Spring的XML配置文件
* 使用该功能的客户端编程

**依赖注入**

## **18.Spring中的依赖注入是什么？**

[依赖注入](http://www.javacodegeeks.com/2014/02/dependency-injection-options-for-java.html)作为控制反转(IOC)的一个层面，可以有多种解释方式。在这个概念中，你不用创建对象而只需要描述如何创建它们。你不必通过代码直接的将组件和服务连接在一起，而是通过配置文件说明哪些组件需要什么服务。之后IOC容器负责衔接。

## **19.有哪些不同类型的IOC(依赖注入)？**

* **构造器依赖注入：**构造器依赖注入在容器触发构造器的时候完成，该构造器有一系列的参数，每个参数代表注入的对象。
* **Setter方法依赖注入：**首先容器会触发一个无参构造函数或无参静态工厂方法实例化对象，之后容器调用bean中的setter方法完成Setter方法依赖注入。

## **20.你推荐哪种依赖注入？构造器依赖注入还是Setter方法依赖注入？**

你可以同时使用两种方式的依赖注入，最好的选择是使用构造器参数实现强制依赖注入，使用setter方法实现可选的依赖关系。

**Spring Beans**

## **21.什么是Spring Beans？**

[Spring Beans](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-bean-reference-example/)是构成Spring应用核心的Java对象。这些对象由Spring IOC容器实例化、组装、管理。这些对象通过容器中配置的元数据创建，例如，使用XML文件中定义的创建。

在Spring中创建的beans都是单例的beans。在bean标签中有一个属性为”singleton”,如果设为true，该bean是单例的，如果设为false，该bean是原型bean。Singleton属性默认设置为true。因此，spring框架中所有的bean都默认为单例bean。

## **22.Spring Bean中定义了什么内容？**

Spring Bean中定义了所有的配置元数据，这些配置信息告知容器如何创建它，它的生命周期是什么以及它的依赖关系。

## **23.如何向Spring 容器提供配置元数据？**

有三种方式向Spring 容器提供元数据:

* XML配置文件
* 基于注解配置
* [基于Java的配置](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-java-config-example/)

## **24.你如何定义bean的作用域？**

在Spring中创建一个bean的时候，我们可以声明它的作用域。只需要在bean定义的时候通过’scope’属性定义即可。例如，当Spring需要产生每次一个新的bean实例时，应该声明bean的scope属性为prototype。如果每次你希望Spring返回一个实例，应该声明bean的scope属性为singleton。

## **25.说一下Spring中支持的bean作用域**

Spring框架支持如下五种不同的作用域：

* **singleton**：在Spring IOC容器中仅存在一个Bean实例，Bean以单实例的方式存在。
* **prototype**：一个bean可以定义多个实例。
* **request**：每次HTTP请求都会创建一个新的Bean。该作用域仅适用于WebApplicationContext环境。
* **session**：一个HTTP Session定义一个Bean。该作用域仅适用于WebApplicationContext环境.
* **globalSession**：同一个全局HTTP Session定义一个Bean。该作用域同样仅适用于WebApplicationContext环境.

bean默认的scope属性是’**singleton**‘。

## **26.Spring框架中单例beans是线程安全的吗？**

不是，Spring框架中的单例beans不是线程安全的。

## **27.解释Spring框架中bean的生命周期**

* Spring容器读取XML文件中bean的定义并实例化bean。
* Spring根据bean的定义设置属性值。
* 如果该Bean实现了BeanNameAware接口，Spring将bean的id传递给setBeanName()方法。
* 如果该Bean实现了BeanFactoryAware接口，Spring将beanfactory传递给setBeanFactory()方法。
* 如果任何bean BeanPostProcessors 和该bean相关，Spring调用postProcessBeforeInitialization()方法。
* 如果该Bean实现了InitializingBean接口，调用Bean中的afterPropertiesSet方法。如果bean有初始化函数声明，调用相应的初始化方法。
* 如果任何bean BeanPostProcessors 和该bean相关，调用postProcessAfterInitialization()方法。
* 如果该bean实现了DisposableBean，调用destroy()方法。

## **28.哪些是最重要的bean生命周期方法？能重写它们吗？**

有两个重要的bean生命周期方法。第一个是setup方法，该方法在容器加载bean的时候被调用。第二个是teardown方法，该方法在bean从容器中移除的时候调用。

bean标签有两个重要的属性(init-method 和 destroy-method)，你可以通过这两个属性定义自己的初始化方法和析构方法。Spring也有相应的注解：@PostConstruct 和 @PreDestroy。

## **29.什么是Spring的内部bean？**

当一个bean被用作另一个bean的属性时，这个bean可以被声明为内部bean。在基于XML的配置元数据中，可以通过把元素定义在 或元素内部实现定义内部bean。内部bean总是匿名的并且它们的scope总是prototype。

## **30.如何在Spring中注入Java集合类？**

Spring提供如下几种类型的[集合配置元素](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-collections-list-set-map-and-properties-example/)：

* list元素用来注入一系列的值，允许有相同的值。
* set元素用来注入一些列的值，不允许有相同的值。
* map用来注入一组”键-值”对，键、值可以是任何类型的。
* props也可以用来注入一组”键-值”对，这里的键、值都字符串类型。

## **31.什么是bean wiring？**

Wiring，或者说bean Wiring是指beans在Spring容器中结合在一起的情况。当装配bean的时候，Spring容器需要知道需要哪些beans以及如何使用依赖注入将它们结合起来。

## **32.什么是bean自动装配？**

Spring容器可以[自动配置](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-autowire-example/)相互协作beans之间的关联关系。这意味着Spring可以自动配置一个bean和其他协作bean之间的关系，通过检查BeanFactory 的内容里没有使用和< property>元素。

## **33.解释自动装配的各种模式？**

自动装配提供五种不同的模式供Spring容器用来自动装配beans之间的依赖注入:

* **no：**默认的方式是不进行自动装配，通过手工设置ref 属性来进行装配bean。
* **byName：**通过参数名自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byName。之后容器试图匹配、装配和该bean的属性具有相同名字的bean。
* **byType：**通过参数的数据类型自动自动装配，Spring容器查找beans的属性，这些beans在XML配置文件中被设置为byType。之后容器试图匹配和装配和该bean的属性类型一样的bean。如果有多个bean符合条件，则抛出错误。
* **constructor：**这个同byType类似，不过是应用于构造函数的参数。如果在BeanFactory中不是恰好有一个bean与构造函数参数相同类型，则抛出一个严重的错误。
* **autodetect：**如果有默认的构造方法，通过 construct的方式自动装配，否则使用 byType的方式自动装配。

## **34.自动装配有哪些局限性？**

自动装配有如下局限性：

* **重写：**你仍然需要使用 和< property>设置指明依赖，这意味着总要重写自动装配。
* **原生数据类型:**你不能自动装配简单的属性，如原生类型、字符串和类。
* **模糊特性：**自动装配总是没有自定义装配精确，因此，如果可能尽量使用自定义装配。

## **35.你可以在Spring中注入null或空字符串吗？**

完全可以。

**Spring注解**

## **36.什么是Spring基于Java的配置？给出一些注解的例子**

[基于Java的配置](http://www.javacodegeeks.com/2013/04/spring-java-configuration.html)允许你使用Java的注解进行Spring的大部分配置而非通过传统的XML文件配置。

以注解@Configuration为例，它用来标记类，说明作为beans的定义，可以被Spring IOC容器使用。另一个例子是@Bean注解，它表示该方法定义的Bean要被注册进Spring应用上下文中。

## **37.什么是基于注解的容器配置?**

另外一种替代XML配置的方式为基于注解的配置，这种方式通过字节元数据装配组件而非使用尖括号声明。开发人员将直接在类中进行配置，通过注解标记相关的类、方法或字段声明，而不再使用XML描述bean之间的连线关系。

## **38.如何开启注解装配？**

注解装配默认情况下在Spring容器中是不开启的。如果想要开启基于注解的装配只需在Spring配置文件中配置元素即可。

## **39.@Required 注解**

@Required表明bean的属性必须在配置时设置，可以在bean的定义中明确指定也可通过自动装配设置。如果bean的属性未设置，则抛出BeanInitializationException异常。

## **40.@Autowired 注解**

@Autowired 注解提供更加精细的控制，包括自动装配在何处完成以及如何完成。它可以像@Required一样自动装配setter方法、构造器、属性或者具有任意名称和/或多个参数的PN方法。

## **41. @Qualifier 注解**

当有多个相同类型的bean而只有其中的一个需要自动装配时，将@Qualifier 注解和@Autowire 注解结合使用消除这种混淆，指明需要装配的bean。

**Spring数据访问**

## **42.在Spring框架中如何更有效的使用JDBC？**

使用Spring JDBC框架，资源管理以及错误处理的代价都会减轻。开发人员只需通过statements和queries语句从数据库中存取数据。Spring框架中通过使用模板类能更有效的使用JDBC，也就是所谓的JdbcTemplate([例子](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/))。

## **43.JdbcTemplate**

JdbcTemplate类提供了许多方法，为我们与数据库的交互提供了便利。例如，它可以将数据库的数据转化为原生类型或对象，执行写好的或可调用的数据库操作语句，提供自定义的数据库错误处理功能。

## **44.Spring对DAO的支持**

[Spring对数据访问对象(DAO)](http://www.javacodegeeks.com/2012/09/spring-dao-and-service-layer.html)的支持旨在使它可以与数据访问技术(如 JDBC, Hibernate 及JDO)方便的结合起来工作。这使得我们可以很容易在的不同的持久层技术间切换，编码时也无需担心会抛出特定技术的异常。

## **45.使用Spring可以通过什么方式访问Hibernate？**

使用Spring有两种方式访问Hibernate：

* 使用Hibernate Template的反转控制以及回调方法
* 继承HibernateDAOSupport，并申请一个AOP拦截器节点

## **46.Spring支持的ORM**

Spring支持一下ORM：

* Hibernate
* iBatis
* JPA (Java -Persistence API)
* TopLink
* JDO (Java Data Objects)
* OJB

## **47.如何通过HibernateDaoSupport将Spring和Hibernate结合起来？**

使用Spring的SessionFactory 调用LocalSessionFactory。结合过程分为以下三步：

* 配置Hibernate SessionFactory
* 继承HibernateDaoSupport实现一个DAO
* 使用AOP装载事务支持

## **48.Spring支持的事务管理类型**

Spring支持如下两种方式的事务管理：

* **编程式事务管理：**这意味着你可以通过编程的方式管理事务，这种方式带来了很大的灵活性，但很难维护。
* **声明式事务管理：**这种方式意味着你可以将[事务管理和业务代码分离](http://www.javacodegeeks.com/2011/09/spring-declarative-transactions-example.html)。你只需要通过注解或者XML配置管理事务。

## **49.Spring框架的事务管理有哪些优点？**

* 它为不同的事务API(如JTA, JDBC, Hibernate, JPA, 和JDO)提供了统一的编程模型。
* 它为编程式事务管理提供了一个简单的API而非一系列复杂的事务API(如JTA).
* 它支持声明式事务管理。
* 它可以和Spring 的多种数据访问技术很好的融合。

## **50.你更推荐那种类型的事务管理？**

许多Spring框架的用户选择声明式事务管理，因为这种方式和应用程序的关联较少，因此更加符合轻量级容器的概念。声明式事务管理要优于编程式事务管理，尽管在灵活性方面它弱于编程式事务管理(这种方式允许你通过代码控制业务)。

**Spring面向切面编程(AOP)**

## **51.解释AOP**

[面向切面编程](http://www.javacodegeeks.com/2014/02/applying-aspect-oriented-programming.html),或AOP允许程序员模块化横向业务逻辑，或定义核心部分的功能，例如日志管理和事务管理。

## **52.切面(Aspect)**

AOP的核心就是切面，它将多个类的通用行为封装为可重用的模块。该模块含有一组API提供 cross-cutting功能。例如,日志模块称为日志的AOP切面。根据需求的不同，一个应用程序可以有若干切面。在Spring AOP中，切面通过带有@Aspect注解的类实现。

## **53.在Spring AOP中concern和 cross-cutting concern的区别是什么？**

Concern(核心逻辑)：表示在应用程序中一个模块的行为。Concern可以定义为我们想要实现的功能。

Cross-cutting concern(横向的通用逻辑)：指的是整个应用程序都会用到的功能，它影响整个应用程序。例如，日志管理（Logging）、安全管理（Security）以及数据交互是应用程序的每个模块都要涉及到的，因此这些都属于Cross-cutting concern。

## **54.连接点(Join point)**

连接点代表应用程序中插入AOP切面的地点。它实际上是Spring AOP框架在应用程序中执行动作的地点。

## **55.通知(Advice)**

通知表示在方法执行前后需要执行的动作。实际上它是Spring AOP框架在程序执行过程中触发的一些代码。

Spring切面可以执行一下五种类型的通知:

* before(前置通知)：在一个方法之前执行的通知。
* after(最终通知)：当某连接点退出的时候执行的通知（不论是正常返回还是异常退出）。
* after-returning(后置通知)：在某连接点正常完成后执行的通知。
* after-throwing(异常通知)：在方法抛出异常退出时执行的通知。
* around(环绕通知)：在方法调用前后触发的通知。

## **56.切入点(Pointcut)**

切入点是一个或一组连接点，通知将在这些位置执行。可以通过表达式或匹配的方式指明切入点。

## **57.什么是引入？**

引入允许我们在已有的类上添加新的方法或属性。

## **58.什么是目标对象？**

被一个或者多个切面所通知的对象。它通常是一个代理对象。也被称做被通知（advised）对象。

## **59.什么是代理？**

代理是将通知应用到目标对象后创建的对象。从客户端的角度看，代理对象和目标对象是一样的。

## **60.有几种不同类型的自动代理？**

* BeanNameAutoProxyCreator：bean名称自动代理创建器
* DefaultAdvisorAutoProxyCreator：默认通知者自动代理创建器
* Metadata autoproxying：元数据自动代理

## **61.什么是织入？什么是织入应用的不同点？**

织入是将切面和其他应用类型或对象连接起来创建一个通知对象的过程。织入可以在编译、加载或运行时完成。

## **62.解释基于XML Schema方式的切面实现**

在这种情况下，切面由使用XML文件配置的类实现。

## **63.解释基于注解方式(基于@AspectJ)的切面实现**

在这种情况下(基于@AspectJ的实现)，指的是切面的对应的类使用Java 5注解的声明方式。

**Spring的MVC框架**

## **64.什么是Spring的MVC框架？**

Spring提供了一个[功能齐全的MVC框架用于构建Web应用程序](http://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-hello-world-example/)。Spring框架可以很容易的和其他的MVC框架融合(如Struts)，该框架使用控制反转(IOC)将控制器逻辑和业务对象分离开来。它也允许以声明的方式绑定请求参数到业务对象上。

## **65.DispatcherServlet**

Spring的MVC框架围绕DispatcherServlet来设计的，它用来处理所有的HTTP请求和响应。

## **66.WebApplicationContext**

WebApplicationContext继承了ApplicationContext，并添加了一些web应用程序需要的功能。和普通的ApplicationContext 不同，WebApplicationContext可以用来处理主题样式，它也知道如何找到相应的servlet。

## **67.什么是Spring MVC框架的控制器？**

控制器提供对应用程序行为的访问，通常通过服务接口实现。控制器解析用户的输入，并将其转换为一个由视图呈现给用户的模型。Spring 通过一种极其抽象的方式实现控制器，它允许用户创建多种类型的控制器。

## **68.@Controller annotation**

@Controller注解表示该类扮演控制器的角色。Spring不需要继承任何控制器基类或应用Servlet API。

## **69.@RequestMapping annotation**

@RequestMapping注解用于将URL映射到任何一个类或者一个特定的处理方法上。