13. 什么是Spring IOC 容器？

Spring IOC 负责创建对象，管理对象（通过依赖注入（DI），装配对象，配置对象，并且管理这些对象的整个生命周期。

14. IOC的优点是什么？

IOC 或 依赖注入把应用的代码量降到最低。它使应用容易测试，单元测试不再需要单例和JNDI查找机制。最小的代价和最小的侵入性使松散耦合得以实现。IOC容器支持加载服务时的饿汉式初始化和懒加载。

15. ApplicationContext通常的实现是什么？

- **FileSystemXmlApplicationContext**：此容器从一个XML文件中加载beans的定义，XML Bean 配置文件的全路径名必须提供给它的构造函数。
- **ClassPathXmlApplicationContext**：此容器也从一个XML文件中加载beans的定义，这里，你需要正确设置classpath因为这个容器将在classpath里找bean配置。
- **WebXmlApplicationContext**：此容器加载一个XML文件，此文件定义了一个WEB应用的所有bean。

16. Bean 工厂和 Application contexts 有什么区别？

Application contexts提供一种方法处理文本消息，一个通常的做法是加载文件资源（比如镜像），它们可以向注册为监听器的bean发布事件。另外，在容器或容器内的对象上执行的那些不得不由bean工厂以

程序化方式处理的操作，可以在Application contexts中以声明的方式处理。Application contexts实现了MessageSource接口，该接口的实现以可插拔的方式提供获取本地化消息的方法。

17. 一个Spring的应用看起来象什么？

- 一个定义了一些功能的接口
- 这实现包括属性，它的Setter ， getter 方法和函数等
- Spring AOP
- Spring 的XML 配置文件
- 使用以上功能的客户端程序

依赖注入

18. 什么是Spring的依赖注入？

依赖注入，是IOC的一个方面，是个通常的概念，它有多种解释。这概念是说你不用创建对象，而只需要描述它如何被创建。你不在代码里直接组装你的组件和服务，但是要在配置文件里描述哪些组件需要哪些服务，之后一个容器（IOC容器）负责把他们组装起来。

19. 有哪些不同类型的IOC（依赖注入）方式？

- 构造器依赖注入 构造器依赖注入通过容器触发一个类的构造器来实现的，该类有一系列参数，每个参数代表一个对其他类的依赖。
- **Setter**方法注入：Setter方法注入是容器通过调用无参构造器或无参static工厂 方法实例化bean之后，调用该bean的setter方法，即实现了基于setter的依赖注入。

20. 哪种依赖注入方式你建议使用，构造器注入，还是 Setter方法注入？

你两种依赖方式都可以使用，构造器注入和Setter方法注入。最好的解决方案是用构造器参数实现强制依赖，setter方法实现可选依赖。

Spring Beans

21.什么是Spring beans？

Spring beans 是那些形成Spring应用的主干的java对象。它们被Spring IOC容器初始化，装配，和管理。这些beans通过容器中配置的元数据创建。比如，以XML文件中<bean/> 的形式定义。

Spring 框架定义的beans都是单件beans。在bean tag中有个属性"singleton"，如果它被赋为TRUE，bean 就是单件，否则就是一个 prototype bean。默认是TRUE，所以所有在Spring框架中的beans 缺省都是单件。点击这里一图Spring Bean的生命周期。

22. 一个 Spring Bean 定义 包含什么？

一个Spring Bean 的定义包含容器必知的所有配置元数据，包括如何创建一个bean，它的生命周期详情及它的依赖。

23. 如何给Spring 容器提供配置元数据？

这里有三种重要的方法给Spring 容器提供配置元数据。

XML配置文件。

基于注解的配置。

基于java的配置。

24. 你怎样定义类的作用域？

当定义一个<bean> 在Spring里，我们还能给这个bean声明一个作用域。它可以通过bean 定义中的scope属性来定义。如，当Spring要在需要的时候每次生产一个新的bean实例，bean的scope属性被指

定为prototype。另一方面，一个bean每次使用的时候必须返回同一个实例，这个bean的scope 属性 必须设为 singleton。

25. 解释Spring支持的几种bean的作用域

Spring框架支持以下五种bean的作用域：

- **singleton :** bean在每个Spring ioc 容器中只有一个实例。
- **prototype**：一个bean的定义可以有多个实例。
- **request**：每次http请求都会创建一个bean ，该作用域仅在基于web的Spring ApplicationContext情形下有效。
- **session**：在一个HTTP Session中，一个bean定义对应一个实例。该作用域仅在基于web的Spring ApplicationContext情形下有效。
- **global-session**：在一个全局的HTTP Session中，一个bean定义对应一个实例。该作用域仅在基于web的Spring ApplicationContext情形下有效。

缺省的Spring bean 的作用域是Singleton。

26. Spring框架中的单例bean是线程安全的吗？

不，Spring框架中的单例bean不是线程安全的。

27. 解释Spring框架中bean的生命周期

- Spring容器 从XML 文件中读取bean的定义，并实例化bean。
- Spring根据bean的定义填充所有的属性。
- 如果bean实现了BeanNameAware 接口，Spring 传递bean 的ID 到 setBeanName方法。
- 如果Bean 实现了 BeanFactoryAware 接口， Spring传递beanfactory 给setBeanFactory 方法。
- 如果有任何与bean相关联的BeanPostProcessors，Spring会在postProcesserBeforeInitialization()方法内调用它们。
- 如果bean实现IntializingBean了，调用它的afterPropertySet方法，如果bean声明了初始化方法，调用此初始化方法。
- 如果有BeanPostProcessors 和bean 关联，这些bean的postProcessAfterInitialization() 方法将被调用。
- 如果bean实现了 DisposableBean，它将调用destroy()方法。

点击这里一图Spring Bean的生命周期。

28. 哪些是重要的bean生命周期方法？ 你能重载它们吗？

有两个重要的bean 生命周期方法，第一个是setup ， 它是在容器加载bean的时候被调用。第二个方法是 teardown 它是在容器卸载类的时候被调用。

The bean 标签有两个重要的属性（init-method和destroy-method）。用它们你可以自己定制初始化和注销方法。它们也有相应的注解（@PostConstruct和@PreDestroy）。

29. 什么是Spring的内部bean？

当一个bean仅被用作另一个bean的属性时，它能被声明为一个内部bean，为了定义inner bean，在Spring 的 基于XML的 配置元数据中，可以在 <property/>或 <constructor-arg/> 元素内使用<bean/> 元素，内部bean通常是匿名的，它们的Scope一般是prototype。

30. 在 Spring中如何注入一个java集合？

Spring提供以下几种集合的配置元素：

- <list>类型用于注入一列值，允许有相同的值。
- <set> 类型用于注入一组值，不允许有相同的值。
- <map> 类型用于注入一组键值对，键和值都可以为任意类型。
- <props>类型用于注入一组键值对，键和值都只能为String类型。

31. 什么是bean装配？

装配，或bean 装配是指在Spring 容器中把bean组装到一起，前提是容器需要知道bean的依赖关系，如何通过依赖注入来把它们装配到一起。

32. 什么是bean的自动装配？

Spring 容器能够自动装配相互合作的bean，这意味着容器不需要<constructor-arg>和<property>配置，能通过Bean工厂自动处理bean之间的协作。

33. 解释不同方式的自动装配

有五种自动装配的方式，可以用来指导Spring容器用自动装配方式来进行依赖注入

- **no**：默认的方式是不进行自动装配，通过显式设置ref 属性来进行装配。
- **byName**：通过参数名 自动装配，Spring容器在配置文件中发现bean的autowire属性被设置成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**：通过参数类型自动装配，Spring容器在配置文件中发现bean的autowire属性被设置成byType，之后容器试图匹配、装配和该bean的属性具有相同类型的bean。如果有多个bean符合条件，则抛出错误。
- **constructor**：这个方式类似于byType， 但是要提供给构造器参数，如果没有确定的带参数的构造器参数类型，将会抛出异常。
- **autodetect**：首先尝试使用constructor来自动装配，如果无法工作，则使用byType方式。

34.自动装配有哪些局限性？

自动装配的局限性是：

- 重写：你仍需用 <constructor-arg>和 <property> 配置来定义依赖，意味着总要重写自动装配。
- 基本数据类型 :你不能自动装配简单的属性，如基本数据类型，String字符串，和类。
- 模糊特性：自动装配不如显式装配精确，如果有可能，建议使用显式装配。

35. 你可以在Spring中注入一个null 和一个空字符串吗？

可以。

Spring注解

36. 什么是基于Java的Spring注解配置? 给一些注解的例子

基于Java的配置，允许你在少量的Java注解的帮助下，进行你的大部分Spring配置而非通过XML文件。

以@Configuration 注解为例，它用来标记类可以当做一个bean的定义，被Spring IOC容器使用。另一个例子是@Bean注解，它表示此方法将要返回一个对象，作为一个bean注册进Spring应用上下文。点击这里学习JAVA几大元注解。

37. 什么是基于注解的容器配置？

相对于XML文件，注解型的配置依赖于通过字节码元数据装配组件，而非尖括号的声明。

开发者通过在相应的类，方法或属性上使用注解的方式，直接组件类中进行配置，而不是使用xml表述bean的装配关系。

38. 怎样开启注解装配？

注解装配在默认情况下是不开启的，为了使用注解装配，我们必须在Spring配置文件中配置<context:annotation-config/>元素。

39. @Required 注解

这个注解表明bean的属性必须在配置的时候设置，通过一个bean定义的显式的属性值或通过自动装配，若@Required注解的bean属性未被设置，容器将抛出BeanInitializationException。

40. @Autowired 注解

@Autowired 注解提供了更细粒度的控制，包括在何处以及如何完成自动装配。它的用法和@Required一样，修饰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却只有一个需要自动装配时，将@Qualifier 注解和@Autowire 注解结合使用以消除这种混淆，指定需要装配的确切的bean。点击这里学习更多常用注解。

Spring数据访问

42.在Spring框架中如何更有效地使用JDBC？

使用SpringJDBC 框架，资源管理和错误处理的代价都会被减轻。所以开发者只需写statements 和queries从数据存取数据，JDBC也可以在Spring框架提供的模板类的帮助下更有效地被使用，这个模板叫JdbcTemplate（例子见这里here）

43. JdbcTemplate

JdbcTemplate 类提供了很多便利的方法解决诸如把数据库数据转变成基本数据类型或对象，执行写好的或可调用的数据库操作语句，提供自定义的数据错误处理。

44. Spring对DAO的支持

Spring对数据访问对象（DAO）的支持旨在简化它和数据访问技术如JDBC，Hibernate or JDO 结合使用。这使我们可以方便切换持久层。编码时也不用担心会捕获每种技术特有的异常。

45. 使用Spring通过什么方式访问Hibernate？

在Spring中有两种方式访问Hibernate：

- 控制反转 Hibernate Template和 Callback
- 继承 HibernateDAOSupport提供一个AOP 拦截器

46. Spring支持的ORM

Spring支持以下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？

用Spring的 SessionFactory 调用 LocalSessionFactory。集成过程分三步：

- 配置the Hibernate SessionFactory
- 继承HibernateDaoSupport实现一个DAO
- 在AOP支持的事务中装配

48. Spring支持的事务管理类型

Spring支持两种类型的事务管理：

- 编程式事务管理 这意味你通过编程的方式管理事务，给你带来极大的灵活性，但是难维护。
- 声明式事务管理 这意味着你可以将业务代码和事务管理分离，你只需用注解和XML配置来管理事务。

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，提供一个不变的编程模式。
- 它为编程式事务管理提供了一套简单的API而不是一些复杂的事务API如
- 它支持声明式事务管理。
- 它和Spring各种数据访问抽象层很好得集成。

50. 你更倾向用那种事务管理类型？

大多数Spring框架的用户选择声明式事务管理，因为它对应用代码的影响最小，因此更符合一个无侵入的轻量级容器的思想。声明式事务管理要优于编程式事务管理，虽然比编程式事务管理（这种方式允许你通过代码控制事务）少了一点灵活性。

Spring面向切面编程（AOP

51. 解释AOP

面向切面的编程，或AOP， 是一种编程技术，允许程序模块化横向切割关注点，或横切典型的责任划分，如日志和事务管理。

52. Aspect 切面

AOP核心就是切面，它将多个类的通用行为封装成可重用的模块，该模块含有一组API提供横切功能。比如，一个日志模块可以被称作日志的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。

52. 在Spring AOP 中，关注点和横切关注的区别是什么？

关注点是应用中一个模块的行为，一个关注点可能会被定义成一个我们想实现的一个功能。

横切关注点是一个关注点，此关注点是整个应用都会使用的功能，并影响整个应用，比如日志，安全和数据传输，几乎应用的每个模块都需要的功能。因此这些都属于横切关注点。

54. 连接点

连接点代表一个应用程序的某个位置，在这个位置我们可以插入一个AOP切面，它实际上是个应用程序执行Spring AOP的位置。

55. 通知

通知是个在方法执行前或执行后要做的动作，实际上是程序执行时要通过SpringAOP框架触发的代码段。

Spring切面可以应用五种类型的通知：

- **before**：前置通知，在一个方法执行前被调用
- **after**：在方法执行之后调用的通知，无论方法执行是否成功
- **after-returning**：仅当方法成功完成后执行的通知
- **after-throwing**：在方法抛出异常退出时执行的通知
- **around**：在方法执行之前和之后调用的通知

56. 切点

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？

引入允许我们在已存在的类中增加新的方法和属性。

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。也指被通知（advised）对象。

59. 什么是代理？

代理是通知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

DefaultAdvisorAutoProxyCreator

Metadata autoproxying

61. 什么是织入。什么是织入应用的不同点？

织入是将切面和到其他应用类型或对象连接或创建一个被通知对象的过程。

织入可以在编译时，加载时，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，切面由常规类以及基于XML的配置实现。

63. 解释基于注解的切面实现

在这种情况下(基于@AspectJ的实现)，涉及到的切面声明的风格与带有java5标注的普通java类一致。