**中软国际考题-笔试**

1. 单选题
2. 下列关于Spring特性中IoC描述错误的是（A）。

A. IoC就是指程序之间的关系由程序代码直接操控。

B. 所谓“控制反转”是指控制权由应用代码转到外部容器，即控制权的转移。

C. IoC将控制创建的职责搬进了框架中，从应用代码脱离开来。

D. 使用Spring的IoC容器时只需指出组件需要的对象，在运行时Spring的IoC容器会根据XML配置数据提供给它

2、下面关于在Spring中配置Bean的init-method的说法正确的是（C）

A. init-method是在最前面执行的

B. init-method在构造方法后，依赖注入前执行

C. init-method在依赖注入之后执行

D. init-method在依赖注入之后，构造函数之前执行

3、下面关于Spring管理Bean的说法正确的是（B ）

A. Bean在Spring容器中有两种管理方式，工厂模式和单例模式

B. Bean在 Spring容器中有两种管理方式，原型模式和单例模式

C. Spring默认的利用工厂模式管理Bean

D. Spring默认利用原型模式管理Bean

4、下面关于设置注入优点说法正确的是（C ）

A. 构造期即创建一个完整、合法的对象

B. 需要写繁琐的setter方法的

C. 对于复杂的依赖关系，设置注入更简洁，直观

D. 以上说法都不对

5、下面是哪个是对IOC的正确描述。（A）

A. 对象的创建、维护、销毁等生命周期的控制由程序控制改为由IOC容器控制，需要某个对象时就直接通过名字去IOC容器中获取。

B.用new实例化对象。

C.程序主动获取对象。

6. 下面关于spring描述错误的是：（C ）  
A Spring支持可插入的事务管理器，使事务划分更轻松，同时无需处理底层的问题。  
B Spring事务管理的通用抽象层还包括JTA策略和一个JDBC DataSource。  
C 与JTA或EJB CMT一样，Spring的事务支持依赖于Java EE环境。  
D Spring事务语义通过AOP应用于 POJO，通过XML进行配置。

7、下列（    ）不是Spring AOP中的通知类型( C )

A．前置通知  B．后置通知  C．代理通知  D．异常通知

8、Scopes a single bean definition to any number of object instances. 描述的是哪一种scope （B）

A．singleton   B．prototype   C．request  D．session

9. Spring中实现AOP需要以下哪个包(C)

A.Web

B.Transcations

C.Aspects

D.Messaging

9、下列关于Spring特性中IoC描述错误的是(A)

A.IoC就是指程序之间的关系由程序代码直接操控

B.所谓”控制反转”是指控制权由应用代码转到外部容器,即控制权的转移

C.IoC将控制创建的职责搬进了框架中;并把它从应用代码脱离开来

D.当使用Spring的IoC容器时只需要指出组件需要的对象,在运行时Spring的容器会根据XML配置数据提供给它

10、使用Quartz框架创建调度器，正确的是（C ）。

A．Scheduler sched = new Scheduler();

B．Scheduler sched = SchedulerFactory.getScheduler();

C．SchedulerFactory sfc = new StdSchedulerFactory();

Scheduler sched = sfc.getScheduler();

D．Scheduler sched = StdSchedulerFactory.getScheduler();

11、spring的核心是（B）

A.bean B.ioc C.coi D.动态代理

12、下面对Spring中Bean的注入说法正确的是（B）

A、Bean根据注入方式不同可以分为构造方法注入和自动装载。

B、使用构造方法注入构造对象的同时，完成依赖关系的建立

C、在关系的对象很多时，使用构造方法注入更适合

D、构造方法通过constructor-index属性来指定，在该标签下要指定索引的位置

13. 下面关于事务管理说法错误的是（D）

A、Spring事务管理方式包括编程方式、声明式

B、编程式采用的方式来控制事物的边界

C、声明式采用配置xml方式，将事务管理与实际业务代码解耦

D、实现编程事物，可以实现PlatformTransaction接口或是使用事务模板TransactionTemplate

14关于ApplicationContext接口，下列说法错误的是（C）

A、它是beanfactory的子接口

B、ApplicationContext是基于beanfactory而建立的，ApplicationContext继承了beanFactory

的功能

C、fileSystemXmlApplicationContext是其子接口（应说“子实现”？）

D、ApplicationContext提供了对资源文件进行访问的支持

15、以下哪一项不是Spring中Bean的自动装配的选择方式（ ）

A． By Type

B． By Name

C． No

D． Constructor方式

E． Autodetect方式

1. 多选
2. Spring是一个功能丰富而易用的集成框架，它的核心是（AB）

A 依赖注入（DI）

B 面向方面编程（AOP）

C 分层架构

D 丰富的标签库

2、下面关于IOC的理解，正确的是（A B）【选两项】

A. 控制反转

B. 对象被动的接受依赖类

C. 对象主动的去找依赖类

D. 一定要用接口

3、可以实现IoC (Inversion of Control) 模式的方法都有哪些？（A、B、C、D）

A：依赖注入(Dependency Injection)

B：Service Locator

C：Java 注解

D：Schema XML方式

4. Spring中ApplicationContext能够用到的Bean的作用域都有哪些?(ABC)

A：Singleton B：Prototype

C：Session   D：Response

5、我们说，Spring是个容器，我们初始化Spring容器后获得这个容器，就能获得里面所配置JavaBean，那么如何初始化Spring容器？ ( C、 D 　)

A. 初始化FactoryBean接口实例

B. 以Bean配置文件初始化FactoryBean接口实例

C. new 出ApplicationContext接口实例

D. 以Bean配置文件初始化ApplicationContext接口实例

6、Spring容器初始化后，我们可以从中获得配置在其内的Bean，前提是我们能获取Spring容器的引用,那么Spring容器引用可以是以下哪个接口？ ( A B C )

A. ApplicationContext B. BeanFactory

C. WebApplicationContext D. ProxyFactory

7、下面关于IoC的理解，正确的是（A B）

A.控制反转

B.对象被动的接受依赖类

C.对象主动的去找依赖类

D.一定要用接口

8、使用JdbcTemplate API进行查询操作数据库，下列说法正确的是（A、C ）。

A．查询表记录总数能用queryForInt()方法实现

B．查询表记录总数不能使用queryForObject()方法实现

C．queryForList()方法返回一个List，其中List里的每个对象一定是个Map

D．queryForList()方法返回一个List，其中List里的每个对象一定是个实体类对象

9、下面哪些项是spring的中的核心技术？（A、B）

A. ioc B. aop C. 动态代理

D. 注入技术; E. 以上都是

10、在关于依赖注入，描述正确的是（BC）

A、依赖注入可以增强系统各组件的依赖关系

B、所谓依赖注入，就是明确地定义组件接口，独立开发各个组件

，然后根据组件件依赖关系组装运行的设计模式

C、依赖注入能够降低系统各组件的依赖关系，提倡面向接口编程

D、Spring的依赖注入和控制反转是完全不同的两个概念

11、 Spring中Bean的作用域都有哪些?（ ）

A． Singleton

B． Prototype

C． Session

D． Response

1. 填空题
2. Spring框架是一个轻量级容器，以\_\_\_ IoC \_\_\_\_\_模式作为核心,从而可以实现应用组件的\_\_\_ 松散耦合 \_\_\_\_结构，让应用程序组件可以容易进行测试。
3. Spring的通知类型有Before通知、After通知、Throws通知和Around通知
4. Spring 的框架是一个轻量级容器，以IOC模式作为核心，从而可以实现应用程序组件的松散耦合结构。
5. Spring中实现IoC模式的方法都有：注入依赖、java注解、java注解、serviceLocator
6. Spring中定义的有具体意义的组件注解有：\_@Controller\_、\_@Service\_、\_@Repository\_。
7. IoC的三种注入方式\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_(接口注入、setter方式注入、构造器注入)
8. 使用JdbcTemplate API操作数据库，\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_ 方法能够插入一条数据。(execute、save)
9. Spring注入有是：构造注入、\_\_\_\_\_\_\_\_\_\_ 、接口注入()
10. IOC翻译过来是什么\_\_\_\_\_\_\_ (控制反转)
11. Spring框架是一个 容器,以模式作为核心, 从而可以实现应用程序组件的\_\_\_\_\_\_\_\_\_\_\_\_\_结构,让应用程序组件可以容易进行测试、扩展。（轻量级、IOC、松散耦合）
12. spring ioc三种注入方式： 、 、 。（接口注入、set注入、构造方法注入）
13. 判断题
14. 在Spring的核心机制中，当一个Java实例需要其他JAVA实例的时候，必须程序显示获取（F）（系统可以自动提供需要的实例）
15. 依赖注入基本思想是：明确地定义组件接口，独立开发各个组件，然后根据组件的依赖关系组装运行（T）
16. 通过依赖注入，JavaEE应用中的各种组件不需要以硬编码的方式进行耦合（T）
17. 使用构造注入法时，在构造函数中决定依赖关系的注入顺序。（ √ ）
18. 所谓依赖注入，就是明确地定义组件接口，独立开发各个组件。（ T ）
19. 依赖注入能够降低系统各组件的依赖关系，提倡面向接口编程。（T）
20. IoC就是指程序之间的关系由程序代码直接操控（ F ）
21. Spring框架提供了一种IoC自动注入功能，需要我们以new Object()方式创建对象(F)
22. Spring是一个轻型的容器,是J2EE规范的轻量级实现,是企业应用的”一站式”解决方案,其核心就是bean工厂,spring是非侵入式的(T)
23. Spring的依赖注入和控制反转是完全不同的两个概念(F)
24. Spring框架最基本的部分是Spring Core模块，提供IOC和依赖注入(T)
25. Spring是一个轻量级框架，大大简化了java企业级开发(T)
26. 依赖注入需要大量接口,增加了编码复杂度(F)
27. IoC就是指程序之间的关系由程序代码直接操控（ F ）
28. 所谓“控制反转”是指控制权由应用代码转到外部容器，即控制权的转移（ T ）
29. IoC将控制创建的职责搬进了框架中，从应用代码脱离开来（ T ）
30. 使用Spring的IoC容器时只需指出组件需要的对象，在运行时Spring的IoC容器会根据XML配置数据提供给它（ T ）
31. Job是一个类，可以直接被继承创建工作类（ F ）
32. 如果实现指定时间点的计划任务，可以选择SimpleTrigger（ F ）
33. 一个Trigger可以和多个Calendar关联，以便排除某些时间点（ T ）
34. JobDataMap可以通过调度上下文获取（ T ）
35. Spring主要好处是不用写代码，代码少，不易于维护。（ F ）
36. IOC容易是服务器吗（ F）
37. Spring中,我们可以选择DBCP,C3P0来配置数据源。（ T ）
38. Spring中配置的Bean,要么只能都用Autowire自动装配,要么都不用Autowire自动装配。（ F ）
39. 下面配置的Bean,Spring会在启动时,IOC容器默认会将该bean提前进行实例化。（ F ）

<bean id="lazy" class="com.foo.ExpensiveToCreateBean" lazy-init="true">

<!-- various properties here...?-->

</bean>

五、简答题

# 1、简单说一下对Spring的IoC理解？

答案：IoC与DI是密不可分的

IoC，控制反转：将对象交给容器管理，你只需要在spring配置文件中配置相应的bean，以及设置相关的属性，让spring容器生成类的实例对象以及管理对象。在spring容器启动的时候，spring会把你在配置文件中配置的bean都初始化以及装配好，然后在你需要调用的时候，就把它已经初始化好的那些bean分配给你需要调用这些bean的类。就是将对象的控制权反转给spring容器管理

DI机制（Dependency Injection，依赖注入）：可以说是IoC的其中一个内容，在容器实例化对象的时候主动的将被调用者（或者说它的依赖对象）注入给调用对象。比如对象A需要操作数据库，以前我们总是要在A中自己编写代码来获得一个Connection对象，有了 spring我们就只需要告诉spring，A中需要一个Connection，至于这个Connection怎么构造，何时构造，A不需要知道。在系统运行时，spring会在适当的时候制造一个Connection，然后像打针一样，注射到A当中，这样就完成了对各个对象之间关系的控制。

2、IOC的优点是什么？

答案：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。

### 3、使用Spring框架的好处是什么？。(分别写出2条即可)（5分）

答案：

轻量：Spring 是轻量的。

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

### 4、解释一下你理解的IOC?

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。

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。

5、Spring Bean的作用域之间的区别?

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。

session：与请求范围类似，确保每个session中有一个bean的实例。

global-session：与标准的HTTP Session作用域，不过它仅仅在基于portlet的web应用中才有意义

5、什么是IoC的核心功能?

答案:IoC:Inversion of Control 控制反转指的是对象的创建权反转(交给)给IoC容器(Spring).控制了外部资源获取,IoC容器帮对象找相应的依赖对象并注入，而不是由对象主动去找

6、Spring中依赖注入与传统编程之间的差别是什么?

答案: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。但在Spring里，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;创建被调用者实例的工作通常由Spring 容器来完成，然后注入调用者，因此也称为依赖注入。依赖注入的主要作用是起到解耦合的作用。

7、IoC最大的好处是什么?

IoC 最大的好处是降低了对象的耦合性，实现了应用的松散耦合,使应用容易测试。因为把对象生成放在了XML里定义，所以当我们需要换一个实现了类将会变成很简单(一般这样的对象都是现实于某种接口的>，只要修改XML就可以了。

8、请写出bean的生命周期的方法?

(1) 通过设置bean的init-method属性指定初始化的方法，他的限制是方法无法接受任何参数，方法可以为static。(2)实现InitializingBean接口的afterPrioertesset (方法。(3)销毁对象可以通过disposableBean的destroy的实现。

9、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，它的生命周期详情级它的依赖

10、Bean工厂创建bean的三种方式?

1) 直接构造,需要一个default constructor和相应的setters/getters方法

2) 采用静态工厂方法,如

<bean id=”exampleBean”class=”examples.ExampleBean2”

factory-method=”createInstance”/>这里要求examples.ExampleBean2类有个静态方法 createInstance

（3）非静态工厂方法

<bean id="myFactoryBean" class="..."> </bean>

<bean id="exampleBean" factory-bean="myFactoryBean" factory-method="createInstance"/>

在这里,必须没有class标签,factory-bean是BeanFactory,factory-method是它的非静态方法，myFactoryBean可以通过容器来管理和配置。

11、简述Spring对于邮件发送、纯文本邮件、带附件邮件所提供的支持。

答：

1)采用JobDetailBean类，它扩展了JobDetai类，采用可声明方式配置任务数据。缺省情况下，采用<bean>标签的id作为任务名，使用缺省组作为组名，通过jobDataAsMap作为配置任务数据。此外，Spring提供了一个MethodInvokingJobDetailFactoryBean，通过这个FactoryBean可以将Spring容器中Bean的方法包装成Quartz任务，这样开发者就不必为Job创建对应的类。

2)建立触发器。可以选择SimpleTriggerBean或CronTriggerBean类。SimpleTriggerBean缺省情况下把可重复执行次数设为无限。

3)创建SchedulerFactory。缺省情况下，SchedulerFactoryBean创建一个StdSchedulerFactory的实例，后者创建Scheduler的实现。可以通过设置schedulerFactoryClass属性来覆盖这个行为，需要继承SchedulerFactory接口来实现你自己的版本。Spring的SchedulerFactoryBean让我们可以脱离Quartz自身的配置体系，而以更具Spring风格的方式定义Scheduler。此外，还可以享受Scheduler生命周期和Spring 容器生命周期绑定的好处。

12. 简述在Spring中使用Quartz的步骤。

答：

1.建立Web工程，加入Spring和MyBatis的有关Jar

2.建立开发目录结构，如创建实体类Employee.java

3.创建数据访问接口

4.创建数据访问接口的实现类

5.配置SQL映射语句文件

6.配置MyBatis应用配置文件，配置MyBatis应用配置文件MyBatis-Configuration.xml。

7.配置Spring应用配置文件

13．简述Spring对于邮件发送、纯文本邮件、带附件邮件所提供的支持。

答：

Spring提供了非常方便的Mail抽象层。它通过一个MailSender接口封装了邮件发送Bean，而SimpleMailMessage封装了纯文本的简单邮件，他是一个纯粹的POJO。

在JavaMail中发送带有附件的MimeMessage是非常麻烦的，需要构建Multipart和多个MimeBodyPart对象，而Spring提供了一个助手类MimeMessageHelper，能非常方便的操作MimeMessage。

14、简述IOC是什么？及能做什么？

答案：

IOC—Inversion of Control，即“控制反转”，不是什么技术，而是一种设计思想。在Java开发中，IOC意味着将设计好的对象交给容器控制，而不是传统的在你的对象内部直接控制

IOC不是一种技术，只是一种思想，一个重要的面向对象编程的法则，它能指导我们如何设计出松耦合、更优良的程序。传统应用程序都是由我们在类内部主动创建依赖对象，从而导致类与类之间高耦合，难于测试；有了IOC容器后，把创建和查找依赖对象的控制权交给了容器，由容器进行注入组合对象，所以对象与对象之间是松散耦合，这样也方便测试，利于功能复用，更重要的是使得程序的整个体系结构变得非常灵活。

15、简述IOC和DI的关系

IOC—Inversion of Control，即“控制反转”

DI—Dependency Injection，即“依赖注入”

它们是同一个概念的不同角度描述，由于控制反转概念比较含糊（可能只是理解为容器控制对象这一个层面，很难让人想到谁来维护对象关系），所以2004年大师级人物Martin Fowler又给出了一个新的名字：“依赖注入”，相对IOC 而言，“依赖注入”明确描述了“被注入对象依赖IOC容器配置依赖对象”。

六、程序实现题

1、假设现在已经有了一个Person类，里面有两个属性name,age并且有set/get方法，现在需要实现Spring的IOC，写出spring配置文件（比如配置文件命名为Bean.xml）中bean的配置，对Person类中的name以及age赋予初始值“s”和”20”，以及IOC获取实例的测试代码。

|  |
| --- |
| 配置文件：  **<bean id="person" class="XX.XX.XX.Person">**  **<property name="name" value="s"></property>**  **<property name="age" value="20"></property>**  **</bean>**  调用测试：  ApplicationContext context = new ClassPathXmlApplicationContext("Bean.xml");  Person p = (Person)context.getBean("person");  System.out.println(p); |

2、使用IoC给Boss类注入爱好属性(使用List集合)和日程安排(使用Map集合)

|  |
| --- |
| Boss.java  public class Boss {  private List favorites = new ArrayList();  private Map<String, String> jobTime = new HashMap<String, String>();  // 省略get/set……  }  applicationContext.xml  <bean id="bossBean" class="com.pojo.Boss">  <property name="favorites">  <list>  <value>看报</value><value>赛车</value>  </list>  </property>  <property name="jobTime">  <map>  <entry>  <key><value>AM</value></key>  <value>会见客户</value>  </entry>  ……  </map>  </property>  </bean> |

3、定义一个Dog类，属性包括名字、年龄，请通过IOC方式创建该类对象，并在控制台打印属性值。

（10分）

|  |
| --- |
| **//Dog类**  **public class** Dog {  **private** String **name**;  **private int age**;  // gettter/setter  }  // **applicationContext.xml**  *<bean id="dog" class="com.chinasofti.model.Dog">  <constructor-arg index="0" value="京巴"></constructor-arg>  <constructor-arg index="1" value="8"></constructor-arg>  <constructor-arg index="2" value="白色"></constructor-arg> </bean>*  //测试类  @Test **public void** testContext(){  ApplicationContext ac = **new** ClassPathXmlApplicationContext(**"applicationContext.xml"**);  *//Dog dog = new Dog();* Dog dog = (Dog) ac.getBean(**"dog"**);  System.***out***.println(dog.getName()+**"||"**+dog.getAge()+**"||"**+dog.getColor()); } |

4、使用代码片段说明如何在Spring中配置声明式事务处理。

（10分）

|  |
| --- |
| 定义事务管理  <bean id="myTransactionManager"  class="org.springframework.orm.hibernate3.HibernateTransactionManager">  <property name =”sessionFactory” ref=”sessionFactory”>  </bean>  定义事务增强，并定指定事务管理器  <tx:advice id="txAdvice"  transaction-manager="myTransactionManager">  <tx:attributes>  <tx:method name="save\*" propagation="REQUIRED"/>  …  </tx:attributes>  </tx:advice>  定义切面  <aop:config>  定义切入点  <aop:pointcut id="bookshopServiceMethods"  expression="execution(\* com.\*.\*(..))" />  织入增强  <aop:advisor advice-ref="txAdvice"  pointcut-ref="bookshopServiceMethods" />  </aop:config> |

5、<xml>

<bean id=”demo” type=”Person”/>

</xml>

Person中包括二个属性，name和age,由程序读取xml并实例化一个Person

6、public Object getInstance(String ClassName){

private Object result=null;

try {

result=Class.forName(ClassName).newInstance();

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

}

return result;

}

1 TestIOC testIOC=new TestIOC();

2 User user=(User)testIOC.getInstance("com.smile.UserImpl");

3 user.sayHey();

7、定义学生类，包含学号、姓名、性别、爱好，四个属性，其中爱好属性为List集合，在spring中通过IOC方式定义一个学生对象，并赋值为：学号：1001，姓名：张三，性别：男，爱好：打球、旅游

|  |
| --- |
| 学生类：  public class Student {  private String id;  private String name;  private String gender;  private List<String> hobby;  …get/set方法省略  }  Spring配置文件：（文件头部分可省略）  <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context <http://www.springframework.org/schema/context/spring-context.xsd>">  <bean id="s1" class="com.xx.Student">  <property name="id" value="1001" />  <property name="name" value="张三" />  <property name="gender" value="男" />  <property name="hobby">  <list>  <value>打球</value>  <value>旅游 </value>  </list>  </property>  <bean> |