面试问题主要内容如下

**1.Spring，IOC，AOP，工作原理，以及配置文件如何配置。**

**2.数据库内容。**

**3.设计并实现一个功能。**

**4.SpringMVC工作原理，具体如何实现，配置文件怎么配置，哪个属性，及其作用是什么等。**

**5.问问题方式，通过某个功能讲述是通过什么技术实现的，这一点问的比较详细，具体某个工具类都会被提问到，如POI做导出，合并调哪的方法，字体调的方法等**

**6.需注意，简历所述技术几乎都会被问到，如简历中提及技术，还请深入了解。**

spring面试题

**一、spring工作原理：**

1.spring mvc请所有的请求都提交给DispatcherServlet,它会委托应用系统的其他模块负责负责对请求进行真正的处理工作。

2.DispatcherServlet查询一个或多个HandlerMapping,找到处理请求的Controller.

3.DispatcherServlet请请求提交到目标Controller

4.Controller进行业务逻辑处理后，会返回一个ModelAndView

5.Dispathcher查询一个或多个ViewResolver视图解析器,找到ModelAndView对象指定的视图对象

6.视图对象负责渲染返回给客户端。

**二、为什么要用spring:**

AOP 让开发人员可以创建非行为性的关注点，称为横切关注点，并将它们插入到应用程序代码中。使用 AOP 后，公共服务 （比 如日志、持久性、事务等）就可以分解成方面并应用到域对象上，同时不会增加域对象的对象模型的复杂性。

IOC 允许创建一个可以构造对象的应用环境，然后向这些对象传递它们的协作对象。正如单词 倒置 所表明的，IOC 就像反 过来的 JNDI。没有使用一堆抽象工厂、服务定位器、单元素（singleton）和直接构造（straight construction），每一个对象都是用 其协作对象构造的。因此是由容器管理协作对象（collaborator）。

Spring即使一个AOP框架，也是一IOC容器。 Spring 最好的地方是它有助于您替换对象。有了 Spring，只要用 JavaBean 属性和配置文件加入依赖性（协作对象）。然后可以很容易地在需要时替换具有类似接口的协作对象。

**三、请你谈谈SSH整合：**

SSH：

Struts（表示层）+Spring（业务层）+Hibernate（持久层）

Struts：

Struts是一个表示层框架，主要作用是界面展示，接收请求，分发请求。

在MVC框架中，Struts属于VC层次，负责界面表现，负责MVC关系的分发。（View：沿用JSP，HTTP，Form，Tag，Resourse ；Controller：ActionServlet，struts-config.xml，Action）

Hibernate：

Hibernate是一个持久层框架，它只负责与关系数据库的操作。

Spring：

Spring是一个业务层框架，是一个整合的框架，能够很好地黏合表示层与持久层。

**四、介绍一下Spring的事务管理：**

事务就是对一系列的数据库操作（比如插入多条数据）进行统一的提交或回滚操作，如果插入成功，那么一起成功，如果中间有一条出现异常，那么回滚之前的所有操作。

这样可以防止出现脏数据，防止数据库数据出现问题。

开发中为了避免这种情况一般都会进行事务管理。Spring中也有自己的事务管理机制，一般是使用TransactionMananger进行管理，可以通过Spring的注入来完成此功能。

spring提供了几个关于事务处理的类：

TransactionDefinition //事务属性定义

TranscationStatus //代表了当前的事务，可以提交，回滚。

PlatformTransactionManager这个是spring提供的用于管理事务的基础接口，其下有一个实现的抽象类AbstractPlatformTransactionManager,我们使用的事务管理类例如DataSourceTransactionManager等都是这个类的子类。

一般事务定义步骤：

TransactionDefinition td = new TransactionDefinition();

TransactionStatus ts = transactionManager.getTransaction(td);

try

{ //do sth

transactionManager.commit(ts);

}

catch(Exception e){transactionManager.rollback(ts);}

spring提供的事务管理可以分为两类：编程式的和声明式的。编程式的，比较灵活，但是代码量大，存在重复的代码比较多；声明式的比编程式的更灵活。

编程式主要使用transactionTemplate。省略了部分的提交，回滚，一系列的事务对象定义，需注入事务管理对象.

void add(){

transactionTemplate.execute( new TransactionCallback(){

pulic Object doInTransaction(TransactionStatus ts)

{ //do sth}

}

}

声明式：

使用TransactionProxyFactoryBean:

PROPAGATION\_REQUIRED PROPAGATION\_REQUIRED PROPAGATION\_REQUIRED,readOnly

围绕Poxy的动态代理 能够自动的提交和回滚事务

org.springframework.transaction.interceptor.TransactionProxyFactoryBean

PROPAGATION\_REQUIRED–支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。

PROPAGATION\_SUPPORTS–支持当前事务，如果当前没有事务，就以非事务方式执行。

PROPAGATION\_MANDATORY–支持当前事务，如果当前没有事务，就抛出异常。

PROPAGATION\_REQUIRES\_NEW–新建事务，如果当前存在事务，把当前事务挂起。

PROPAGATION\_NOT\_SUPPORTED–以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

PROPAGATION\_NEVER–以非事务方式执行，如果当前存在事务，则抛出异常。

PROPAGATION\_NESTED–如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则进行与PROPAGATION\_REQUIRED类似的操作。

**五、Spring里面如何配置数据库驱动？**

使用”org.springframework.jdbc.datasource.DriverManagerDataSource”数据源来配置数据库驱动。示例如下：

<bean id=”dataSource”>

<property name=”driverClassName”>

<value>org.hsqldb.jdbcDriver</value>

</property>

<property name=”url”>

<value>jdbc:hsqldb:db/appfuse</value>

</property>

<property name=”username”><value>sa</value></property>

<property name=”password”><value></value></property>

</bean>

**六、Spring里面applicationContext.xml文件能不能改成其他文件名？**

ContextLoaderListener是一个ServletContextListener, 它在你的web应用启动的时候初始化。缺省情况下， 它会在WEB-INF/applicationContext.xml文件找Spring的配置。 你可以通过定义一个<context-param>元素名字为”contextConfigLocation”来改变Spring配置文件的位置。示例如下：

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener <context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/xyz.xml</param-value>

</context-param> </listener-class>

</listener>

**七、如何在web应用里面配置spring?**

在web.xml中加入如下同容,在启动web服务器时加载/WEB-INF/applicationContext.xml中的内容。

<servlet>

<servlet-name>context</servlet-name>

<servlet-class>

org.springframework.web.context.ContextLoaderServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

通过如下类得到ApplicationContext实例

WebApplicationContextUtils.getWebApplicationContext

**八、Spring里面如何定义hibernate mapping？**

添加hibernate mapping 文件到web/WEB-INF目录下的applicationContext.xml文件里面。示例如下：

<property name=”mappingResources”>

<list>

<value>org/appfuse/model/User.hbm.xml</value>

</list>

</property>

**九、解释一下Dependency injection(DI,依赖注入)和IOC(Inversion of control,控制反转)?**

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。

**十、spring中的BeanFactory与ApplicationContext的作用有哪些？**

1. BeanFactory负责读取bean配置文档，管理bean的加载，实例化，维护bean之间的依赖关系，负责bean的声明周期。

2. ApplicationContext除了提供上述BeanFactory所能提供的功能之外，还提供了更完整的框架功能：

a. 国际化支持

b. 资源访问：Resource rs = ctx. getResource(”classpath:config.properties”), “file:c:/config.properties”

c. 事件传递：通过实现ApplicationContextAware接口

3. 常用的获取ApplicationContext的方法：

FileSystemXmlApplicationContext：从文件系统或者url指定的xml配置文件创建，参数为配置文件名或文件名数组

ClassPathXmlApplicationContext：从classpath的xml配置文件创建，可以从jar包中读取配置文件

WebApplicationContextUtils：从web应用的根目录读取配置文件，需要先在web.xml中配置，可以配置监听器或者servlet来实现

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<servlet>

<servlet-name>context</servlet-name>

<servlet-class>org.springframework.web.context.ContextLoaderServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

这两种方式都默认配置文件为web-inf/applicationContext.xml，也可使用context-param指定配置文件

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/myApplicationContext.xml</param-value>

</context-param>

**十一、如何在web环境中配置applicationContext.xml文件?**

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

或：

<servlet>

<servlet-name>context</servlet-name>

<servlet-class>

org.springframework.web.context.ContextLoaderServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

通过如下方法取出applicationContext实例:

ApplicationContext ac=WebApplicationContextUtils.getWebApplicationContext(this.getServletContext);

**十二、如何配置spring+struts?**

在struts-config.xml加入一个插件，通过它加载applicationContext.xml

? 在struts-config.xml修改action-mapping标记，具体action交给了DelegateActionProxy

? 通过DelegateActionProxy进入一spring的环境。

? 在spring的applicationContext.xml加入<bean name=”/login” class=”" singleton=”false”/>

**十三、spring+hibernate的配置文件中的主要类有那些?如何配置?**

dataSource

sessionFactory:hibernate.cfg.xml

transactionManager

userDao (extends HibernateDaoSupport)

sessionFactory

facade

proxy

sessionFactory

transactionManager

facade

在myeclipse中先加入spring环境再加入hibernate环境。

如果spring与hibernate结合在一起可以不需要hibernate.cfg.xml文件是否正确

**十四、如何在spring中实现国际化?**

在applicationContext.xml加载一个bean

<bean id=”messageSource” class=”org.springframework.context.support.ResourceBundleMessageSource”>

<property name=”basename”>

<value>message</value>

</property>

</bean>

? 在src目录下建多个properties文件

? 对于非英文的要用native2ascii -encoding gb2312 源 目转化文件相关内容

? 其命名格式是message\_语言\_国家。

? 页面中的中显示提示信息，键名取键值。

? 当给定国家，系统会自动加载对应的国家的properties信息。

? 通过applictionContext.getMessage(“键名”,”参数”,”区域”)取出相关的信息。

**十五、spring中的核心类有那些，各有什么作用?**

BeanFactory：产生一个新的实例，可以实现单例模式

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

**十六、什么是aop，aop的作用是什么?**

面向切面编程（AOP）提供另外一种角度来思考程序结构，通过这种方式弥补了面向对象编程（OOP）的不足

除了类（classes）以外，AOP提供了切面。切面对关注点进行模块化，例如横切多个类型和对象的事务管理

Spring的一个关键的组件就是AOP框架，可以自由选择是否使用AOP

提供声明式企业服务，特别是为了替代EJB声明式服务。最重要的服务是声明性事务管理，这个服务建立在Spring的抽象事物管理之上

允许用户实现自定义切面，用AOP来完善OOP的使用

可以把Spring AOP看作是对Spring的一种增强

**十七、使用Spring有什么好处？**

◆Spring能有效地组织你的中间层对象,无论你是否选择使用了EJB。如果你仅仅使用了Struts或其他的包含了J2EE特有APIs的framework，你会发现Spring关注了遗留下的问题，。

◆Spring能消除在许多工程上对Singleton的过多使用。根据我的经验，这是一个主要的问题，它减少了系统的可测试性和面向对象特性。

◆Spring能消除使用各种各样格式的属性定制文件的需要,在整个应用和工程中，可通过一种 一致的方法来进行配置。曾经感到迷惑，一个特定类要查找迷幻般的属性关键字或系统属性,为此不得不读Javadoc乃至源编码吗？有了Spring，你可 很简单地看到类的JavaBean属性。倒置控制的使用(在下面讨论)帮助完成这种简化。

◆Spring能通过接口而不是类促进好的编程习惯，减少编程代价到几乎为零。

◆Spring被设计为让使用它创建的应用尽可能少的依赖于他的APIs。在Spring应用中的大多数业务对象没有依赖于Spring。

◆使用Spring构建的应用程序易于单元测试。

◆Spring能使EJB的使用成为一个实现选择,而不是应用架构的必然选择。你能选择用POJOs或local EJBs来实现业务接口，却不会影响调用代码。

◆Spring帮助你解决许多问题而无需使用EJB。Spring能提供一种EJB的替换物，它们适于许多web应用。例如,Spring能使用AOP提供声明性事务而不通过使用EJB容器，如果你仅仅需要与单个的数据库打交道，甚至不需要JTA实现。

■Spring为数据存取提供了一致的框架,不论是使用JDBC或O/R mapping产品（如Hibernate）。

Spring确实使你能通过最简单可行的解决办法解决你的问题。这些特性是有很大价值的。

总结起来，Spring有如下优点：

◆低侵入式设计，代码污染极低

◆ 独立于各种应用服务器，可以真正实现Write Once,Run Anywhere的承诺

◆Spring的DI机制降低了业务对象替换的复杂性

◆Spring并不完全依赖于Spring，开发者可自由选用Spring框架的部分或全部

**十八、什么是Spring, 它有什么特点？**

Spring是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架。

◆轻量——从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并 且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：典型地，Spring应用中的对象不依赖于Spring的特定类。

◆控制反转——Spring通过一种称作控制反转（IoC）的技术促进了松耦 合。当应用了IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为IoC与JNDI相反——不 是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

◆面向切面——Spring提供了面向切面编程的丰富支持，允许通过分离应用的 业务逻辑与系统级服务（例如审计（auditing）和事务（）管理）进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们 并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。

◆容器——Spring包含并管理应用对象的配置和生命周期，在这个意义上它是 一种容器，你可以配置你的每个bean如何被创建——基于一个可配置原型（prototype），你的bean可以创建一个单独的实例或者每次需要时都生 成一个新的实例——以及它们是如何相互关联的。然而，Spring不应该被混同于传统的重量级的EJB容器，它们经常是庞大与笨重的，难以使用。

◆框架——Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，应用对象被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久化框架集成等等），将应用逻辑的开发留给了你。

**十九、请介绍一下Spring框架中Bean的生命周期**

一、Bean的定义

Spring通常通过配置文件定义Bean。如：

<?xml version=”1.0″ encoding=”UTF-8″?>

<beans xmlns=”http://www.springframework.org/schema/beans”

xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”

xsi:schemaLocation=”http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd”>

<bean id=”HelloWorld” class=”com.pqf.beans.HelloWorld”>

<property name=”msg”>

<value>HelloWorld</value>

</property>

</bean>

</beans>

这个配置文件就定义了一个标识为 HelloWorld 的Bean。在一个配置文档中可以定义多个Bean。

二、Bean的初始化

有两种方式初始化Bean。

1、在配置文档中通过指定init-method 属性来完成

在Bean的类中实现一个初始化Bean属性的方法，如init()，如：

public class HelloWorld{

public String msg=null;

public Date date=null;

public void init() {

msg=”HelloWorld”;

date=new Date();

}

……

}

然后，在配置文件中设置init-mothod属性：

<bean id=”HelloWorld” class=”com.pqf.beans.HelloWorld”init-mothod=”init” >

</bean>

2、实现 org.springframwork.beans.factory.InitializingBean接口

Bean实现InitializingBean接口，并且增加 afterPropertiesSet() 方法：

public class HelloWorld implement InitializingBean {

public String msg=null;

public Date date=null;

public void afterPropertiesSet() {

msg=”向全世界问好！”;

date=new Date();

}

……

}

那么，当这个Bean的所有属性被Spring的BeanFactory设置完后，会自动调用afterPropertiesSet()方法对Bean进行初始化，于是，配置文件就不用指定 init-method属性了。

三、Bean的调用

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());

四、Bean的销毁

1、使用配置文件中的 destory-method 属性

与初始化属性 init-methods类似，在Bean的类中实现一个撤销Bean的方法，然后在配置文件中通过 destory-method指定，那么当bean销毁时，Spring将自动调用指定的销毁方法。

2、实现 org.springframwork.bean.factory.DisposebleBean接口

如果实现了DisposebleBean接口，那么Spring将自动调用bean中的Destory方法进行销毁，所以，Bean中必须提供Destory方法。

**二十、AOP里面重要的几个名词概念解释：**

切面（Aspect）： 一个关注点的模块化，这个关注点可能会横切多个对象。事务管理是J2EE应用中一个关于横切关注点的很好的例子。 在Spring AOP中，切面可以使用通用类（基于模式的风格） 或者在普通类中以 @Aspect 注解（@AspectJ风格）来实现。

连接点（Joinpoint）： 在程序执行过程中某个特定的点，比如某方法调用的时候或者处理异常的时候。 在Spring AOP中，一个连接点 总是 代表一个方法的执行。 通过声明一个org.aspectj.lang.JoinPoint类型的参数可以使通知（Advice）的主体部分获得连接点信息。

通知（Advice）： 在切面的某个特定的连接点（Joinpoint）上执行的动作。通知有各种类型，其中包括“around”、“before”和“after”等通知。 通知的类型将在后面部分进行讨论。许多AOP框架，包括Spring，都是以拦截器做通知模型， 并维护一个以连接点为中心的拦截器链。

切入点（Pointcut）： 匹配连接点（Joinpoint）的断言。通知和一个切入点表达式关联，并在满足这个切入点的连接点上运行（例如，当执行某个特定名称的方法时）。 切入点表达式如何和连接点匹配是AOP的核心：Spring缺省使用AspectJ切入点语法。

引入（Introduction）： （也被称为内部类型声明（inter-type declaration））。声明额外的方法或者某个类型的字段。 Spring允许引入新的接口（以及一个对应的实现）到任何被代理的对象。 例如，你可以使用一个引入来使bean实现 IsModified 接口，以便简化缓存机制。

目标对象（Target Object）： 被一个或者多个切面（aspect）所通知（advise）的对象。也有人把它叫做 被通知（advised） 对象。 既然Spring AOP是通过运行时代理实现的，这个对象永远是一个 被代理（proxied） 对象。

AOP代理（AOP Proxy）： AOP框架创建的对象，用来实现切面契约（aspect contract）（包括通知方法执行等功能）。 在Spring中，AOP代理可以是JDK动态代理或者CGLIB代理。 注意：Spring 2.0最新引入的基于模式（schema-based）风格和@AspectJ注解风格的切面声明，对于使用这些风格的用户来说，代理的创建是透明的。

织入（Weaving）： 把切面（aspect）连接到其它的应用程序类型或者对象上，并创建一个被通知（advised）的对象。 这些可以在编译时（例如使用AspectJ编译器），类加载时和运行时完成。 Spring和其他纯Java AOP框架一样，在运行时完成织入。

通知的类型：

前置通知（Before advice）： 在某连接点（join point）之前执行的通知，但这个通知不能阻止连接点前的执行（除非它抛出一个异常）。

返回后通知（After returning advice）： 在某连接点（join point）正常完成后执行的通知：例如，一个方法没有抛出任何异常，正常返回。

抛出异常后通知（After throwing advice）： 在方法抛出异常退出时执行的通知。

后通知（After (finally) advice）： 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）。

环绕通知（Around Advice）： 包围一个连接点（join point）的通知，如方法调用。这是最强大的一种通知类型。 环绕通知可以在方法调用前后完成自定义的行为。它也会选择是否继续执行连接点或直接返回它们自己的返回值或抛出异常来结束执行。

环绕通知是最常用的一种通知类型。大部分基于拦截的AOP框架，例如Nanning和JBoss4，都只提供环绕通知。

切入点（pointcut）和连接点（join point）匹配的概念是AOP的关键，这使得AOP不同于其它仅仅提供拦截功能的旧技术。 切入点使得定位通知（advice）可独立于OO层次。 例如，一个提供声明式事务管理的around通知可以被应用到一组横跨多个对象中的方法上（例如服务层的所有业务操作）。