**<!-- 2014/8/4 -->**

**1.请写出实现webservice的几种方式,需要概述每种实现方式的特点**

**AXIS2**

**apache提供的重量级WebService框架，一个WebServices|SOAP|WSDL的引擎，webservice框架的集大成者，不但能制作和发布WebService，而且可以生成java和其他语言版的webservice代码，这是他的优势。同时，AXIS2也是相当复杂的，他所依赖的 JAR数量和大小是惊人的，打包部署发布都比较麻烦，不能很好地与现有应用整合为一体。但是如果需要开发非JAVA语言的客户端，AXIS2是不二的选择。**

**CXF**

**apache旗下另一个SOA框架。CXF来自于XFire项目，经过改造而形成的。**

**XFire**

**高性能的webservice框架。XFire的优点在于开发方便，于现有的web集成很好，可以融为一体，并且开发也很方便。但是对java之外的语言，没有提供相关的代码工具。**

**JAX\_WS**

**java语言对webservice服务的一种实现，用于开发和发布服务。**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1 如果应用程序需要多语言支持，首选AXIS2**

**2 如果应用程序使用spring集成开发，CXF是更好的选择，特别是对于嵌入式的webservice**

**3 如果应用程序没有新的特性需求，就仍用原来项目所用的框架，比如XFire等...**

**2.请写出Spring的七种事务传播行为**

**mandatory**

**支持当前事务，如果当前没有事务抛出异常**

**（我要找个有钱的男票，屌丝滚粗）**

**never**

**不支持当前事务，如果当前有事务抛出异常**

**（我不喜欢有钱的[太花心了?]）**

**not\_support**

**不支持当前事务，总是以非实物环境运行**

**（不喜欢有钱的，就想过没钱的日子）**

**required**

**支持当前事务，如果没有则创建一个新的（常用&&默认）**

**(喜欢有钱的，没钱我可以自己挣钱O(∩\_∩)O）**

**requires\_new**

**总是创建一个新事务，如果当前有事务则挂起当前事务（不是吊死他，是等新事物运行完了再运行原实物）**

**（不喜欢花你的钱，等花完我的再用你的）**

**supports**

**支持当前事务，如果当前没有则以非事务环境运行**

**（有钱过有钱的日子，没钱过没钱的日子）**

**nested**

**如果当前有事务则把新事物作为原实物的内嵌事务进行执行；否则以required方式运行**

**（我破产了你可以罩着我，你破产了我也玩完了...）**

**1: Springmvc和Struts有什么区别？**

    首先spring-mvc和struts1.x/2.x都是以servlet为模型衍生出的技术

**a 核心控制器的不同**

        对于所有的基于mvc模式的框架来说，核心控制器的主要用途在于拦截某一特定的请求，然后对这些请求进行统一的处理

        spring-mvc的核心控制器是servlet(DispatcherServlet);struts2的核心控制器是filter(StrutsPrepareAndExecuteFilter)

**b 请求处理器的不同**

        struts2是基于对象设计，每次发送的请求都会为其分配一个action实例，每个action都会被注入属性(表单项)

        spring-mvc是基于方法设计，只提供一个实例，每次请求执行实例中的不同方法。

**c 对annotation的支持**

        spring-mvc完全实现了零配置，也就是全注解方式管理；

        struts2则需要采用xml配置来进行管理(虽然也提供了零配置概念，但很少有人会用到)

**d 表单项封装**

        struts2提供了两种封装方式(属性驱动and模型驱动),但都是通过action的属性进行封装。

        spring-mvc则是通过方法参数进行封装(当然也会用到第三方javaBean或者直接使用@RequestParam注解)

**e 拦截器支持**

         struts2底层采用拦截器设计实现，几乎所有的服务都被封装在Interceptor中；虽然这种可插拔式的设计在使用上比较方便，但xml中的配置量过于庞大(这还是在interceptor之间可以继承的前提下)。

         spring-mvc的拦截器则是采用独立的aop支持，并且他的拦截器是方法级别的拦截，一个方法对应一个request上下文。

**f ajax支持**

        struts2为了实现对ajax支持内置了json插件，通过返回结果或者拦截器实现对json数据的封装

        spring-mvc则提供了@ResponseBody注解来实现对ajax支持

**g servlet-api支持**

        struts2是类级别的拦截，一个类对应一个request上下文；虽然方法之间也是相互独立的，但其所有的变量都是共享的，虽然不影响程序运行，但给我们的编码造成困扰。

        spring-mvc属于方法级别的拦截，一个方法对应一个request上下文；因此spring-mvc的方法之间基本上是相互独立的，独享request和response数据，方法之间不共享变量。

**h 最主要的....**

        spring-mvc和spring之间是无缝的(spring-mvc最初是作为spring的一个层提出)。在基于spring管理的项目上来说，管理性和安全性要比Struts2高

    [总结 : 两个框架的设计目的都是为了处理HTTP请求,成为优秀的MVC框架；但设计思想是截然不同的...随便拿出一点就是他们之间的区别...]

**2: spring中用过的注解有哪些？分别有什么作用？@Autowired和@Resource有什么区别?**

**@Component @Service @Controller @Repository**

        类级别的注解;被标注的类会成为ioc容器实例化的对象;相当于在ioc容器中手动配置bean

        @Controller也常用在spring-mvc中,标注请求处理器

**@Autowired @Resource**

        标注属性或者setter方法

        实现bean对象之间的被动注入

**Autowired默认按类型装载;是spring内置的;如果想使用名字装载，可结合@Qualifier注解进行使用**

          @Autowired

          @Qualifier("empDao")

           private EmpDaoIf empDao;

**Resource默认按名字装载;是javaEE提供的**

          @Resource(name="empDao")

           private EmpDaoIf empDao;

**@RequestMapping(path)**

        标注请求处理器中的方法，使用被标注的方法处理path指向的请求

        当然也可以使用它标注Controller类，表示这个类中的方法处理url为前缀的请求

**@ResponseBody**

        标注请求处理器中用于处理ajax请求的方法，将被标注方法的返回值封装成json返回给客户端的js

**@RequestParam(paramName)**

        标注请求处理器方法中的参数，使用该参数封装'paramName'指向的参数

**@SessionAttributes({key1,key2...})**

        标注请求处理器，将该处理器方法中添加到ModelMap中的数据key转存到session范围中

        注意:数组中指向的key值不能和方法的参数名重名！

**@Aspect**

**@Pointcut**

**@Before**

**@After**

**@AfterReturning**

**@AfterThrowing**

**@Around**

**@Transactional**

**3 如何设置ajax同步请求**

    A 传统的ajax请求发送方式中

**XMLHttpRequest.open(1,2,false)**      以同步的方式打开到服务器的连接

    B jquery提供的ajax组件中

**$.ajax({async:false})**      将async参数设置为false

        \*\* **同步请求会锁住浏览器**，用户的后续操作需要等待服务器的响应完毕之后才能继续

**4 spring mvc有没有与之类似的框架**

    struts1 struts2 webwork jsf ...

**5 简单介绍一下你对spring mvc的理解**

    spring-mvc是spring框架中提供的一个用于处理客户端HTTP请求的技术。

    在spring-mvc中提供了自己的一套用于和请求处理相关的bean对象：比如请求接收器DispatcherServlet ； 请求解析器HandlerMapping和MultipartResolver ； 请求处理器Controller ； 视图解析器ViewResolver和视图对象View ；以及拦截器HandlerInterceptor。

    spring-mvc的运行和使用需要依附于spring的ioc容器。由ioc容器的工厂特性将spring-mvc的各组件组合运行。

**流程** : 客户端的请求首先被注册在web容器中的DispatcherServlet统一拦截，并转发给ioc容器中的请求解析器（这里会根据请求的类型[字符or字节]自动选择匹配的解析器），接下来在ioc容器环境下，会根据解析之后的字符串[/前缀]查找用于处理请求的处理器Controller，通过@RequestMapping(path)在Controller中查找与之映射的处理方法。请求处理完毕之后，会根据处理请求方法的返回值[String or ModelAndView]以ViewResolver为基础返回一个View视图对象。

    [流程详细版见课件...]

**6 spring的两大核心是什么 / 简述spring的IOC和AOP**

    spring框架提供的两大核心是 : **IOC和AOP**

    ioc和aop是spring框架提供的用于实现对象管理的两种技术。

    IOC:

        全称 : inversion of control?

        直译为控制反转，其核心思想为依赖注入

        ioc是一种用于描述对象创建 及 对象依赖形成的一种技术。

        首先，ioc底层采用工厂模式实现，所以其首要目的-也就是工厂的目的-创建对象；但是ioc创建对象的方式和普通工厂不同，普通工厂是在工厂类中new对象；而ioc则是将需要其创建的对象配置在ioc容器中，使用工厂类解析ioc容器，然后以反射的方式创建对象。ioc的直译'控制反转'正是描述了ioc环境下对象创建的特点：将对象的创建权利从内部代码转移到外部容器，这个转移的过程即为反转。

        其次，ioc的核心思想则是他的被动注入。我们使用ioc的方式创建对象的目的就是为了采用其提供的被动注入思想，以此形成对象之间的依赖关系。ioc为了实现被动注入提供了setter、构造器和方法注入。

    AOP:

        全称 : aspect oriented programming 面向切面编程思想

        底层采用代理模式实现的一种技术

        首先，AOP是OOP的一种延续思想，用于解决OOP开发过程中对象封装、继承、多态所造成的对象之间高度耦合性而提出的一种新的编程思想。

        AOP通过分离组成软件的核心业务逻辑和系统级服务进行内聚性开发。其关注点在于系统级服务部分。

        AOP的实现策略是将系统级服务封装成为一种AOP组件，采用XML配置及引用的方式将这个AOP组件所提供的系统级服务添加给需要该服务的核心业务逻辑。从而实现核心逻辑与通用服务之间的解耦。

        比如，spring中提供的声明式事务功能就是以AOP为基础而提出的一种技术。

        struts2内置的Interceptor 和 spring-mvc内置的HandlerInterceptor都属于AOP的实现技术。

**---------------   spring-mvc的核心是什么 -------------------**

        其实就是spring的核心+mvc的核心

        mvc的核心在于DispatcherServlet[核心控制器] ： 拦截某一特定的请求，然后对这些请求进行统一的处理

        当然里面也涉及到和请求处理相关的 : 请求的解析(HandlerMapping|MultipartResolver) 请求处理(Contorller) 视图解析(ViewResolver) 和 视图对象(View)

**7 webservice里面有什么方法**

    webservice技术有很多种 。 **axis xfire cxf**...

    客户端访问服务器

    cxf提供的是JaxWsProxyFactoryBean

        该工厂提供了 setServiceClass() setAddress() create() 等方法

    xfire提供的是Client

        该对象提供了 invoke("方法名" , 参数列表)

    axis提供的是RPCServiceClient

        getOptions().setTo(需要调用的webservice地址)

        invokeBlocking(调用的方法，参数列表，参数类型)

**8: spring中的事务用过没？**

    ------------------------------------------------------------------------------------------

**8.1 (那你谈谈什么是事务，你的项目中哪里用到事务了)**

    ------------------------------------------------------------------------------------------

    在一次请求过程中，对多次数据库的连接操作进行统一的提交或回滚操作，以此保证数据执行的一致性 。 或者是对多个sql语句进行统一的提交或回滚操作。

    比如涉及到批量添加、删除的操作，或者银行转账这种一次请求涉及多次数据库连接修改的操作。

    ------------------------------------------------------------------------------------------

**8.2 SPRING有哪些事务？**

    ------------------------------------------------------------------------------------------

    spring提供了编程式事务支持 **TransactionTemplate**

    也提供了声明式事务支持 **底层采用AOP实现的事务通知bean**

    也有注解形式的 **@Transactional**

    \*\*\* 还是问spring内置了哪些事务管理器？？？

    Spring的事务管理器有5个，都实现了PlatformTransactionManager接口，如下所示：

    DataSourceTransactionManager        JDBC事务管理器

    HibernateTransactionManager            Hibernate事务管理器

    JdoTransactionManager                JDO事务管理器

    JtaTransactionManager                JTA事务管理器

    PersistenceBrokerTransactionManager    Apache的OJB事务管理器

    ------------------------------------------------------------------------------------------

**8.3 SPRING中怎么使用事务？**

    ------------------------------------------------------------------------------------------

spring即提供了对全局事务的支持, 也提供了对局部事务的支持

即提供了对编程式事务的支持,也提供了对声明式事务的支持

spring中对事务的管理提供了一个接口 PlatformTransactionManager 事务管理器

通过该接口描述spring环境中的事务如何创建，以及应该具备何种特性(提交、回滚)

同时，根据当前工程选择的持久层技术，为其分配不同的事务管理器实现

· 全局事务支持 JTA

· 局部事务支持 [jdbc hibernate]

· 编程式事务 :模板模式的实现类 TransactionTemplate

     将事务的管理过程封装在模板类

· 声明式事务 : 底层采用AOP技术实现,将事务的管理过程

     封装在一个事务通知bean中 ,通过配置切入点的方式描述将

     事务通知bean提供的服务引用给哪些对象/方法

    1)**编程式事务** 在DAO中通过TransactionTemplate模板类封装事务的管理过程

    2)**声明式事务**：底层采用AOP思想实现，将事务的管理过程（包括事务对象的创建，提交和回滚操作）封装在一个事务通知bean中，然后通过配置AOP切入点的方式来描述需要将事务通知bean所提供的事务管理服务引用给哪些核心对象/方法。

    配置方式依赖于AOP的配置

    <tx:advice id="" transaction-manager="指向一个事务管理器">

        <tx:attributes> 为不同的方式设置不同的事务属性

            <tx:method name="表达式匹配方法" 四个属性...rollback-for="监听到什么异常回滚事务"/>

            ....

        </tx:attributes>

    </tx:advice>

    <aop:config>

        <aop:pointcut id="" expression="表达式指向需要引用事务的方法列表"/>

        <aop:advisor advice-ref="" pointcut-ref=""/>

    </aop:config>

    声明式事务的特性 ： 不需要在核心代码中编写事务的管理过程，以此将核心代码与事务管理服务动态分离。

    声明式事务回滚的方式 ： 运行时异常

    ------------------------------------------------------------------------------------------

**8.4 在spring中如何使用事务？在不使用编程式事务和声明式事务的前提下**

注解 @Transactional

    ------------------------------------------------------------------------------------------

**9.spring web.xml怎么配置**

    <listener>

        <listener-class> ... **ContextLoaderListener**</listener-class>

    </listener>

    <context-param>

        <param-name>**contextConfigLocation**</param-value>

        <param-value>classpath:applicationContext.xml</param-value> ? ??

    </context-param>

**10.spring的创始人是谁?**

    Rod Johson

**11.webservice怎么配置发布接口**

     选择一种webservice技术发布 (cxf xfire ...)

**cxf发布方式[基于spring环境]**:

**a 使用@WebService标注需要发布的接口和实现类**

**b 在ioc容器中配置发布的过程** :

<cxf:server serviceClass="指向发布的接口" address="为发布的webservice配置一个唯一地址">

<cxf:serviceBean>

<bean class="指向发布的webservice接口实现类"/>

</cxf:serviceBean>

</cxf:server>

**c 配置用于接受webservice请求的接收器(web.xml)**

<servlet>

<servlet-name>cxf</servlet-name>

<servlet-class>...CXFServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>cxf</servlet-name>

<url-pattern>/ws/\*</url-pattern>

</servlet-mapping>

 '/ws/\*' 表示告诉其他工程如何向本工程发送webservice请求，调用本工程提供的webservice接口。

**12.spring依赖注入实现有哪些方式**

**setter**

     <property name="setter方法后缀" ref="需要注入的对象"/>

**构造器**

     <constructor-arg index="第几个参数" ref="需要注入的对象"/>

**方法注入**

     <lookup-method name="方法名" bean="需要注入的对象"/>

除了这三种方式，也可以使用注解的方式实现 :

**@Autowired @Resource**

**13.任务调度做过哪些方面？ 在spring中如何配置任务调度?**

**JDK内置的Timer和TimerTask实现**

1） 创建一个继承自TimerTask的对象作为任务对象；覆盖其提供的run()，在该方法中实现任务内容

2） 提供一个ServletContextListener用于监听tomcat的启动过程，当tomcat启动之后，在该监听器的初始化方法中，创建一个计时器对象

3） 通过计时器Timer对象加载任务对象的执行（同时设置执行的时间点或者执行的周期）

弊端 : 只能对周期固定的任务进行调度！

**spring集成的quartz框架实现**

1） 提供一个普通的javaBean作为任务对象

2） 将任务对象配置成ioc容器中的bean对象，交给ioc容器管理

3） 描述任务细节(MethodInvokingJobDetailFactoryBean);配置任务对象以及其提供的任务方法(setTargetObject() setTargetMethod())

4） 配置触发器（CronTriggerFactoryBean）;为任务设置触发的时间或周期(setJobDetail() setCronExpression())

5 )  配置调度类(SchedulerFactoryBean）;在调度类中配置所有的触发任务（setTriggers()）

**14：redis缓存工具 在什么地方用 怎么用 数据格式**

redis缓存系统时搭建在服务层和持久层之间；程序加载时，将持久层连接数据库查询出的数据（使用频率比较高的数据）存储在缓存系统中。当浏览器的页面需要用到这种数据时，不再去数据库中查询，而是通过服务层连接redis服务器，从redis服务器中检索数据。

java中提供了jedis框架用于实现对redis缓存系统的支持。

当然如果工程中添加了spring环境，也可以使用spring-data组件中提供的spring-data-redis，在spring支持下去操作缓存系统。

使用过程：**配置redis连接池 -> 配置jedis连接工厂 -> 配置用于封装连接过程的模板类对象（RedisTemplate） -> 调用jedis方法操作数据**

可以在redis缓存系统中存储普通的字符串、基本数据类型、Map、List等集合、以及自定义类型对象。

**15:  使用springmvc是不是每次请求都创建一个controller对象**

由于springmvc的请求处理是依存于IOC容器的，用于处理请求的controller也属于ioc容器中的bean

而ioc容器中bean对象的默认状态为单例状态【scope="singleton"】,所以**默认情况下，spring-mvc只维护一个controller实例来处理多个客户端提交的请求。**

当然，也可以将其设置为非单例状态，以保证每次请求都可以创建一个新的controller对象进行处理。只需要将controller对象的scope属性设置为prototype。或者使用@Scope注解

**17: 用过的前端技术 easyui里面提供了什么布局 有哪些组件**

     easyui bootstrap jquery extjs miniUI

     easyui布局 : layout布局[东西南北中]

     panel window messager datagrid treegrid

     form validatebox combobox .....

     pagination

**19: HTTP和webservice请求的区别**

     首先这是两个完全不相关的概念

     HTTP是一种网络协议（超文本数据传输协议）

     webService是一种为了实现应用程序之间数据通信的技术，底层采用SOAP协议实现

     而SOAP协议是HTTP POST的一个专用版本，遵循一种特殊的 XML消息格式 [ContentType - text/xml] 。

     跨域方面 ： HTTP不能实现跨域；只能访问本服务中的内容；而webservice设计的目的就是为了实现跨域的消息传输，可以调用其他应用程序的服务。

     效率方面 ： HTTP在数据处理方面效率更高 ； 但webservice能处理更加复杂的数据类型

     两者之间是有联系的，Webservice实现消息的传送就是将消息依附在http上进行传输的

**20： spring如何整合hibernate**

     ORM提供的模板模式的实现类 : HibernateTemplate 封装传统hibernate链接数据库操作数据的流程

     通过在DAO中对模板类的使用，实现对传统开发流程的代替。

      要在DAO中使用模板类对象，需要先获取一个模板类实例，而获取实例的方式是采用了IOC的被动注入

      因此需要在ioc容器中配置一个HibernateTemplate-Bean对象

      而HibernateTemplate的使用需要一个SessionFactory实例的支持。

      因此还需要在IOC容器中配置一个SessionFactory实例 - 配置方式：LocalSessionFactoryBean

      通过其提供的三个setter方法 ： setDataSource setMappingResouces setHibernateProperties 实现SessionFactory的配置过程！

**21： spring如何整合struts**

     使用spring的ioc容器管理struts中用于处理请求的Action

     将Action配置成ioc容器中的bean

     1）客户端提交的被struts拦截到的请求，通过struts.xml中的<action>元素来配置其映射关系，name属性指向请求名，class属性指向IOC容器中某个bean的ID值，表示使用该bean指向的处理器处理当前请求

     2）将action配置在ioc容器中之后，该action默认会在tomcat启动时初始化，并且初始化的单例状态为：单例（只维护一个Action实例处理多用户提交的请求）。区别于单独使用struts时客户端每次发送请求创建一个新的action处理。

     3）整合过程使用了struts2提供的插件 struts-spring-plugin.jar 该插件中注册的一个ObjectFactory ： **StrutsSpringObjectFactory** . 该工厂负责根据class属性的值，访问ioc容器，在ioc容器中查找匹配的bean作为请求处理器。

**22：spring如何整合mybatis**

使用spring的IOC容器管理mybatis的接口实例

     1）在ioc容器中配置mybatis核心对象SqlSessionFactory，使用工厂bean:SqlSessionFactoryBean,通过其提供的setDataSource()和setMapperResources()为该对象注入数据源和映射文件

     2）在ioc容器中配置用于扫描接口的工厂bean，通过扫描接口获取接口的实例bean：使用工厂bean:MapperScannerConfigurer，通过其提供的setBasePackage()和setSqlSessionFactoryBeanName()来设置需要扫描的包结构和SqlSessionFactory

**23.Spring怎么控制页面跳转的，spring处理完请求怎么把返回数据到页面，页面中怎么把spring返回的数据取出来**

     通过处理请求方法的**返回值**进行控制

     返回值提供了两种形式 : **String  ModelAndView**

**String** 提供了两种方式 : redirect forward

**ModelAndView**提供了两种方式 : 注入模型 | 注入视图

          对于注入模型的方式，需要在ioc容器中注册视图解析器对模型进行解析，以此匹配视图对象

     将数据返回到页面 两种方式：

     对于普通请求，使用servlet-api对象作用域，ModelMap以及@SessionAttributes，也可以使用ModelAndView提供的addObject(key,value)  -  对应的jsp页面中使用el表达式读取request|session范围中的数据

     对于ajax请求，采用@ResponseBody，将需要返回的数据作为处理请求方法的返回值 - 对应的jsp页面中使用ajax的回调接口接收数据

**24 解释Spring支持的几种bean的作用域**

singleton (默认)

     prototype

     request

     session

     global session

 注意 ： global session作用域类似于标准的HTTP Session作用域，不过它仅仅在基于portlet的web应用中才有意义。Portlet规范定义了全局Session的概念，它被所有构成某个portlet web应用的各种不同的portlet所共享。在global session作用域中定义的bean被限定于全局portlet Session的生命周期范围内

**25 Spring框架中的单例Bean是线程安全的吗?**

**是**

**spring通过采用ThreadLocal解决线程安全问题**

Spring框架里的bean，获取实例的时候都是默认的单例模式。

          一般情况下，只有无状态的Bean才可以在多线程环境下共享，在Spring中，绝大部分Bean都可以声明为singleton作用域。就是因为Spring对一些Bean中非线程安全状态采用ThreadLocal进行处理，让它们也成为线程安全的状态，因为有状态的Bean就可以在多线程中共享

 ThreadLocal会为每一个线程提供一个独立的变量副本，从而隔离了多个线程对数据的访问冲突。因为每一个线程都拥有自己的变量副本，从而也就没有必要对该变量进行同步了。ThreadLocal采用的这种“以空间换时间”的方式，为每一个线程都提供了一份变量，因此可以同时访问而互不影响。

**26 为什么说spring是轻量级的**

 spring的提出是为了解决传统JavaEE开发的复杂性而提出，其中最主要的在于 ： 使用普通的javaBean代替传统javaEE规范中的EJB技术。

     spring的轻量级主要体现在相比于重量级的EJB而言：

      1） 从大小和开销方面，spring相比EJB更低

      2） 对服务器没有任何依赖性；spring作为javaEE框架，既可以在web环境下使用，同样也可以脱离web容器单独使用。

          3） 对代码的低侵入性；spring管理对象的方式是采用普通的javaBean，不需要让其管理的对象继承或实现springAPI对象或接口。

**27、redis存储的数据类型是什么？**

redis是作为一个key-value型的数据存储仓库，因此存储的数据类型是键值对类型

     其中 key 使用字符串表示

     value 可以是普通的字符数据（字符串），也可以是基本数据类型；当然也可以存储 List、Set、Map类型 ， 自定义类型对象通过实现序列化也可以存储在redis中。

**28、redis怎样使用的？怎样存储数据的？主要存储的什么类型？（不是说以键值对的形式）**

首先在服务器中启动redis服务 : redis-server.exe redis.conf

     连接redis可以采用spring提供的spring-data-redis组件，该组件封装了jedis访问redis服务器的过程 ：

     首先在ioc容器中配置redis连接池(JedisPoolConfig)、连接工厂(JedisConnectionFactory)和模板类对象(RedisTemplate)

     然后将RedisTemplate注入给业务对象，通过使用模板类实现对redis服务器的连接和操作。

     存储数据 ：

     redisTemplate.set(key , value)  get(key)

     rt.opsForList() -> ListOperations -> rightPush() leftPush() range()

     rt.opsForSet() -> SetOperations -> add() members()

     rt.opsForHash() -> HashOperations -> put(key,value-key,value) entries(key)

                                             get(key,value-key)

**29.Maven怎么用的?**

本地安装apache-maven

     配置maven环境变量 ： m2\_home classpath path

     修改配置文件，指向本地资源库 apache-maven/conf/settings.xml

     命令控制台运行mvn命令，安装maven基础jar

     eclipse中配置maven环境，指向本地安装的maven工具

     创建maven工程,设置groupid（基础包结构），artifactId（工程名）

     需要的jar包在pom.xml的dependencys元素下声明

     ...

**30.Redis缓存工具里面有什么方法**

redisTemplate提供的方法 :

     rt.execute(){ RedisConnection.set(key , value)  get(key)}

     rt.opsForList() -> ListOperations -> rightPush() leftPush() range()

     rt.opsForSet() -> SetOperations -> add() members()

     rt.opsForHash() -> HashOperations -> put(key,value-key,value) entries(key)

                                             get(key,value-key)

**1.Aop用什么框架实现的？**

    aspectJ

**2.aop是哪种模式实现的？**

        代理模式（jdk动态代理和cglib代理两种）

**3 AOP是spring特有的吗？**

        不是 。 AOP是一种编程思想，类似于OOP，可以在多种编程语言中进行使用，不仅局限在java语言中。

        spring框架仅仅是引入AOP思想来实现对象的管理（怎么管理对象...配置引用代替OOP中的对象继承和方法调用）

**4 如何验证身份信息**

    spring security

**5: JSONObject是哪个包下的**

     net.sf.json.JSONObject

**6:请写一个校验数据格式为 10-500 的正则表达式**

      ([1-9]\d)|([1-4]\d{2})|(500)

**7.MVC框架？常用的MVC框架有哪些？**

MVC是模型(model)－视图(view)－控制器(controller)的缩写。 他是一个框架模式，它强制性的使应用程序的输入、处理和[输出](http://baike.baidu.com/view/700402.htm" \t "D:/%E6%A1%86%E6%9E%B6%E4%B8%AD%E7%A7%91%E8%BD%AF%E9%80%9A/%E5%91%A8%E5%90%89%E6%98%93/%E9%9D%A2%E8%AF%95%E9%A2%98/_blank)分开。使用MVC应用程序被分成三个核心部件：模型、[视图](http://baike.baidu.com/view/71981.htm" \t "D:/%E6%A1%86%E6%9E%B6%E4%B8%AD%E7%A7%91%E8%BD%AF%E9%80%9A/%E5%91%A8%E5%90%89%E6%98%93/%E9%9D%A2%E8%AF%95%E9%A2%98/_blank)、控制器。它们各自处理自己的任务。最典型的MVC就是JSP + [servlet](http://baike.baidu.com/view/25169.htm" \t "D:/%E6%A1%86%E6%9E%B6%E4%B8%AD%E7%A7%91%E8%BD%AF%E9%80%9A/%E5%91%A8%E5%90%89%E6%98%93/%E9%9D%A2%E8%AF%95%E9%A2%98/_blank) + [javabean](http://baike.baidu.com/view/28155.htm" \t "D:/%E6%A1%86%E6%9E%B6%E4%B8%AD%E7%A7%91%E8%BD%AF%E9%80%9A/%E5%91%A8%E5%90%89%E6%98%93/%E9%9D%A2%E8%AF%95%E9%A2%98/_blank)的模式。

       常用的MVC框架：

        struts1 struts2 webwork spring-mvc ...

**8.spring mvc如果想返回一个list 在控制层应该怎么写？用什么注解 返回类型是什么**

@RequestMapping(path)

              @ResponseBody

     public List xxx(){ return list};

**9.如果类和方面上面都有@RequestMapping("")  url请求应该怎么写？**

@RequestMapping("/abc")

     public class XxController{

          @RequestMapping("/cba")

          public String hello(){

               // 请求写法  abc/cba.do

               return "xxx";

          }

     }

**10.有没有用过注解？注解怎么用？注解都是干什么用的?**

**用过**

**注解创建 ： @Interface**

**注解分为类级别注解和属性、方法级别注解；可以标注在类体，属性或者方法体上**

**注解的使用是为了简化XML文件的配置。**

**22: 在spring中怎么从一个页面跳到另一个页面？**

**spring-mvc中的页面跳转**

**根据处理请求方法的返回值为客户端返回一个视图 ：**

**String类型返回值 : "redirect:xx.jsp" "forward:xx.jsp"**

**ModelAndView类型返回值 :**

**为该对象提供一个模型字符串 new ModelAndView("success") 根据提供的字符串使用mvc的视图解析器(ViewResolver)匹配一个视图返回**

**直接为该对象提供一个视图对象  new ModelAndView(new InternalResourceView("success.jsp")|RedirectView("success.jsp")**

**20: springmvc的依赖注入方式？**

**spring依赖注入的方式（setter 构造器 方法注入）**

**19.Spring mvc controller中返回类型的注解？页面跳转用注解吗？**

**@ResponseBody**

**15.spring mvc 多文件上传和struts2的上传区别**

**区别在于服务器端的封装方式**

**mvc采用 MultipartFile[] 封装文件列表**

**struts2采用 File[] 封装文件列表**

**17.spring 特性是什么？**

**轻量级**

**控制反转**

**面向切面**

**组件/框架集成**

**（具体的自己详细描述...）**

**14.aop的原理**

**底层采用代理模式实现（jdk动态代理和cglib代理组合）**

**将非核心服务封装在AOP组件中，采用XML配置和引用的方式将AOP组件提供的服务引用给需要的核心业务对象或方法。**

**典型的AOP实现技术：Interceptor**

**31.Springmvc三层是那三层**

**模型 - javaBean**

**视图 - JSP**

**控制器 - Controller**

**23.spring如何实现前台和后台的传值？每个层都有什么作用？**

**spring中前后台传值需要用到它的MVC层**

**每个层什么作用 == 流程(DispatcherServlet HandlerMapping Controller ModelAndView ViewResolver View)的具体使用方式**

**32.事务中用到的声明标签,注解有哪些**

**@Transactional(**

**isolation**

**propagation**

**readOnly**

**timeout**

**rollback-for**

**)**

**21: 切入点在方法中怎么调用？**

**@Pointcut("")**

**public void pc(){}**

**@Before("pc()")**

**public void before(){}**

**34: spring的aop是怎么影响代码的?**

**将业务代码中的核心部分和非核心部分动态分离**

**将非核心部分封装在AOP组件中**

**采用xml配置、引用方式将AOP组件中封装的非核心部分添加给核心部分**

**从而实现两部分逻辑的解耦**

**16.spring struts hibernate 都担当了什么角色？**

     Struts作为系统的整体基础架构，负责MVC的分离，在Struts框架的模型部分，控制业务跳转，利用Hibernate框架对持久层提供支持，Spring做管理，管理struts和hibernate。

     具体的实现方式：用面向对象的分析方法根据需求提出一些模型，将这些模型实现为基本的Java对象，然后编写基本的DAO接口，并给出Hibernate的DAO实现，采用Hibernate架构实现的DAO类来实现Java类与数据库之间的转换和访问，最后由Spring做管理，管理struts和hibernate。

**12.spring mvc 中的c在项目中是怎么实现的？**

**c - Controller 控制器**

**分为三部分 :**

**中央控制器 : DispatcherServlet 注册在web容器中,负责请求的拦截**

**请求解析器 : HandlerMapping 或 MultipartResolver 注册在IOC容器中，负责请求的解析工作**

**请求处理器 : 实现Controller接口 或 使用@Controller标注 ，负责请求的处理**

**13.项目中怎么开启事务**

**首先 - 项目里面如果使用了spring，可以使用spring的AOP技术对事务进行隔离（抽离）。基于AOP的声明式事务可以帮我们从负责的事务操作中得到解脱。被引用事务通知的方法在开始执行之前，容器会自动为我们开始事务；在方法执行完毕并且没有任何问题时，事务自动提交；如果需要回滚事务，可以通过抛异常方式（运行时异常）来进行回滚**

**如果项目里面没有使用spring，每种持久层框架都提供了不同的管理方式。例如JDBC默认属于自动管理（Connection.setAutoCommit(true/false)）;Hibernate可以通过会话Session获取事务对象（Session.beginTransaction()）进行手动提交或回滚。**

**18.做过webservices相关的吗？写过什么方面的接口？在项目中哪块用到了？**

**需要调用第三方程序提供的服务时**

**比如调用支付宝接口、天气信息接口、票务信息接口、QQ授权登录等等。**