Spring知识总结

**1.Spring框架是一个开源的容器性质的轻量级框架。主要有三大特点：容器、IOC（控制反转）、AOP（面向切面编程）。**

**2.Spring框架主要有三大优点：**

**(1) 容器。Spring框架是一个容器，能够管理项目中的所有对象。**

**(2) IOC（控制反转）。Spring将创建对象的方式反转了，从程序员自己创建反转给了程序。**

**(3) AOP（面向切面）。面向切面编程，简而言之，就是将纵向重复的代码横向抽取出来。Spring框架应用了面向切面的思想，主要体现在为容器中管理的对象生成动态代理对象。**

**3.IOC：控制反转，指得是将对象的创建权反转给Spring。作用是实现了程序的解耦合。**

**DI：依赖注入，需要有IOC环境，在Spring创建Bean对象时，动态的将依赖对象注入到Bean对象中去。依赖注入最大的好处就是解耦合。**

**4.BeanFactory接口是Spring框架的顶层接口，是最原始的接口，通过new （BeanFactory的实现类）来启动Spring容器时，并不会创建Spring容器里面的对象，只有在每次通过getBean()获得对象时才会创建。**

**ApplicationContext接口是用来替代BeanFactory接口的，通过new （ApplicationContext接口的实现类）ClassPathXmlApplicationContext来启动Spring容器时，就会创建容器中配置的所有对象。**

1. **Spring中的输入注入方式包括set方法注入、构造函数注入、p名称空间注入、spel注入，除此之外，还包括复杂方式注入，如数组、List、Map、Properties等属性的注入。**
2. **bean有5种作用域，分别是singleton（单例，默认）、prototype（默认）、request、session、globalSession。**
3. **Spring框架最早出现的只有@Component()注解，但如果所有的对象都使用同一个注解，很难区分对象究竟属于哪一层架构。基于此，Spring又推出了@Service()、@Controller()、@Respository()三种注解，用于区分对象属于哪一层架构。**
4. **引用类型注入可以通过三种注解方式来完成，分别为：@Autowired、@Autowired和@Qualifier()二者结合、@Resource()。建议使用@Resource()，但是一般用@Autowired。**
5. **aop名词学习**

**Joinpoint（连接点）：目标对象中，所有可以增强的方法**

**Pointcut（切入点）：目标对象，已经增强的方法。**

**Advice（通知/增强）：增强的代码**

**Target（目标对象）：被代理对象**

**Weaving（织入）：将通知应用到切入点的过程**

**Proxy（代理）：将通知织入目标对象之后，形成代理对象**

1. **AOP，面向切面编程，是对OOP(面向对象编程)的补充和完善。简单来讲就是将纵向重复的代码，横向抽取出来。**
2. **在Spring的核心配置文件applicationContext.xml中配置事务，主要配置三大方面：事务管理器、事务通知和定义事务性切面.**

**实例：<beans xmlns="http://www.springframework.org/schema/beans"**

**xmlns:context="http://www.springframework.org/schema/context" xmlns:p="http://www.springframework.org/schema/p"**

**xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"**

**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-4.2.xsd**

**http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd**

**http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.2.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.2.xsd**

**http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.2.xsd">**

**<!-- 事务管理器 -->**

**<bean id="transactionManager"**

**class="org.springframework.jdbc.datasource.DataSourceTransactionManager">**

**<!-- 数据源 -->**

**<property name="dataSource" ref="dataSource" />**

**</bean>**

**<!-- 事务通知 -->**

**<tx:adviceid="txAdvice" transaction-manager="transactionManager">**

**<tx:attributes>**

**<!-- 传播行为 -->**

**<tx:method name="save\*" propagation="REQUIRED" />**

**<tx:method name="insert\*" propagation="REQUIRED" />**

**<tx:method name="add\*" propagation="REQUIRED" />**

**<tx:method name="create\*" propagation="REQUIRED" />**

**<tx:method name="delete\*" propagation="REQUIRED" />**

**<tx:method name="update\*" propagation="REQUIRED" />**

**<tx:method name="find\*" propagation="SUPPORTS" read-only="true" />**

**<tx:method name="select\*" propagation="SUPPORTS" read-only="true" />**

**<tx:method name="get\*" propagation="SUPPORTS" read-only="true" />**

**</tx:attributes>**

**</tx:advice>**

**<!-- 定义事务性切面 -->**

**<aop:config>**

**<aop:advisor advice-ref="txAdvice"**

**pointcut="execution(\* cn.e3mall.service..\*.\*(..))" />**

**</aop:config>**

**</beans>**

SpringMVC知识总结：

**1.spring mvc是靠spring 启动的。通过springjar包的org.springframework.web.servlet.DispatcherServlet这个servlet类具体启动的。  
<servlet-name>springmvc</servlet-name>  
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>**

**这个dispatcherservlet这个类，有个contextConfigLocation参数，指定springmvc的配置文件xml。  
2.理清一下思路：每个tomcat下的程序war包都有一个web.xml。web.xml可以启动servlet。  
web.xml只能配置servlet相关信息，如servletmapping等等，不能配置bean的注入，扫描包路径，注解驱动等，这**

**3.二、配置解析**

**1.Dispatcherservlet**

**DispatcherServlet是前置控制器，配置在web.xml文件中的。拦截匹配的请求，Servlet拦截匹配规则要自已定义，把拦截下来的请求，依据相应的规则分发到目标Controller来处理，是配置spring MVC的第一步。  
4. <!-- scan the package and the sub package -->  
<context:component-scan base-package="test.SpringMVC"/>  
  
<!-- don't handle the static resource -->  
<mvc:default-servlet-handler />  
  
<!-- if you use annotation you must configure following setting -->  
<mvc:annotation-driven />  
  
<!-- configure the InternalResourceViewResolver -->  
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"  
id="internalResourceViewResolver">  
<!-- 前缀 -->  
<property name="prefix" value="/WEB-INF/jsp/" />  
<!-- 后缀 -->  
<property name="suffix" value=".jsp" />  
</bean>  
</beans>**

**5.spring mvc配置文件里的配置标签，必背：  
<context:component-scan base-package="test.SpringMVC"/>  
<mvc:annotation-driven />  
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"  
  
<!-- scan the package and the sub package -->  
<context:component-scan base-package="test.SpringMVC"/>  
  
<!-- don't handle the static resource -->  
<mvc:default-servlet-handler />  
  
<!-- if you use annotation you must configure following setting -->  
<mvc:annotation-driven />  
  
<!-- configure the InternalResourceViewResolver -->  
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"  
id="internalResourceViewResolver">  
<!-- 前缀 -->  
<property name="prefix" value="/WEB-INF/jsp/" />  
<!-- 后缀 -->  
<property name="suffix" value=".jsp" />  
</bean>  
</beans>**

**6.spring mvc的那些controller标签，是注入到spring的ioc容器中的。不是spring mvc自己的ioc，spring mvc没有自己的ioc容器，都是spring的**

**7.@RequestBody**

**该注解用于读取Request请求的body部分数据，使用系统默认配置的HttpMessageConverter进行解析，然后把相应的数据绑定到要返回的对象上 ,再把HttpMessageConverter返回的对象数据绑定到 controller中方法的参数上**

**@ResponseBody**

**该注解用于将Controller的方法返回的对象，通过适当的HttpMessageConverter转换为指定格式后，写入到Response对象的body数据区**

**8.requestParam正确用法@RequestParam(value = "invitCode", required = false) string invitCode  
requestParam修饰的东西全在小括号里面（）。**

**9.四、自动匹配参数  
//match automatically  
@RequestMapping("/person")  
public String toPerson(String name,double age){  
System.out.println(name+" "+age);  
return "hello";  
}  
　五、自动装箱**

**1.编写一个Person实体类  
package test.SpringMVC.model;  
  
public class Person {  
public String getName() {  
return name;  
}  
public void setName(String name) {  
this.name = name;  
}  
public int getAge() {  
return age;  
}  
public void setAge(int age) {  
this.age = age;  
}  
private String name;  
private int age;  
  
}**

**2.在Controller里编写方法  
  
//boxing automatically  
@RequestMapping("/person1")  
public String toPerson(Person p){  
System.out.println(p.getName()+" "+p.getAge());  
return "hello";  
}**

**10.在SpringMVC中，bean中定义了Date，double等类型，如果没有做任何处理的话，日期以及double都无法绑定。**

**解决的办法就是使用spring mvc提供的@InitBinder标签**

**11.比较简单的可以直接应用springMVC的注解@initbinder和spring自带的WebDataBinder类和操作**

**[java] view plain copy  
在CODE上查看代码片派生到我的代码片**

**@InitBinder  
public void initBinder(WebDataBinder binder) {  
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  
dateFormat.setLenient(false);  
binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat, true));  
}**

**还要在springMVC配置文件中加上**

**[html] view plain copy  
在CODE上查看代码片派生到我的代码片**

**<!-- 解析器注册 -->  
<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">  
<property name="messageConverters">  
<list>  
<ref bean="stringHttpMessageConverter"/>  
</list>  
</property>  
</bean>  
<!-- String类型解析器，允许直接返回String类型的消息 -->  
<bean id="stringHttpMessageConverter" class="org.springframework.http.converter.StringHttpMessageConverter"/>**

**这样就可以直接将上传的日期时间字符串绑定为日期类型的数据了**

Mybats知识总结

**1.MyBatis是一个半ORM（对象关系映射）框架，它内部封装了JDBC，开发时只需要关注SQL语句本身，不需要花费精力去加载驱动、创建连接、创建Statement等繁杂过程。程序员直接编写原生态sql，可以严格控制sql执行性能，灵活度高。Mybatis可以使用XML或注解来配置和映射原生信息，将POJO映射成数据库中的记录，避免几乎所有JDBC代码和手动设置参数以及获得结果集。通过xml文件或注解的方式将要执行的各种statement配置起来，并通过java对象和statement中sql的动态参数进行映射生成最终执行的sql语句，最后由Mybatis框架执行sql并将返回结果映射为java对象并返回。（从执行sql到返回result的过程）。**

**2.基于SQL语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL写在XML文件中，接触SQL与程序代码的耦合，便于统一管理；提供XML便签，支持编写动态SQL语句，并可以重用。与JDBC相比，减少了50%以上的代码量，消除了JDBC大量冗余的代码，不需要手动开关连接。很好的与各种数据库兼容（因为MyBatis使用JDBC来连接数据，所以只要JDBC支持数据库MyBatis都支持）。能够与Spring框架很好的集成。提供映射标签，支持对象与数据库的ORM字段关系映射；提供对象关系映射标签，支持对象关系组件维护。**

**3.SQL语句的编写工作量较大，尤其当字段多，关联表多时，对开发人员编写SQL语句的功底有一定要求。**

**SQL语句依赖数据库，导致数据库移植性差，不能随意更换数据库。**

**4.MyBatis专注于SQL本身，是一个足够灵活的DAO层解决方案。**

**对性能的要求很高，或者对需求变化较多的项目，如互联网项目，MyBatis将是不错的选择。**

**5.Mybatis不完全是一个ORM框架，因为Mybatis需要程序员自己编写SQL语句。Mybatis直接编写原生态SQL，可以严格控制SQL执行性能，灵活度高，非常适合对关系数据模型要求不高的软件开发，因为这类软件需求变化频繁，一旦需求变化要求迅速输出成果。但是灵活的前提是Mybatis无法做到数据库无关性，如果需要实现支持多种数据库的软件，则需要自定义多套SQL映射文件，工作量大。**

**Hibernate对象/关系映射能力强，数据无关性好，对于关系模型要求高的软件，如果用Hibernate开发可以节省很多代码，提高效率。**

**6.MyBatis使用RowBounds对象进行分页，它是针对ResultSet结果集执行的内存分页而非物理分页。可以在SQL内直接书写带有物理分页的参数来完成分页功能。也可以使用分页插件来完成物理分页。**

**分页插件的基本原理是使用Mybatis提供的插件接口，实现自定义插件，在插件的拦截方法内拦截执行的SQL，然后重新SQL，根据dialect方言，添加对应的物理分页语句和物理分页参数。**

**7.第一种是使用标签：逐一定义数据库列明和对象属性名之间的映射关系。第二种是使用SQL列的别名功能：将列的别名书写为对象属性名。有了列名和属性名的映射关系后，Mybatis通过反射创建对象，同时使用反射给对象的属性逐一赋值并返回，那些找不到映射关系的属性是无法赋值的。**

**8.Mybatis动态SQL可以在xml映射文件内，以标签的形式编写动态sql，执行原理是根据表达式的值完成逻辑判断并动态拼接SQL的功能。**

**Mybatis提供了9种动态SQL标签：trim、where、set、foreach、if、choose、when、otherwise、bind**

**9.还有很多其他的标签，<resultMap>、<parameterMap>、<sql>、<include>、<selectKey>，加上动态sql的9个标签，trim|where|set|foreach|if|choose|when|otherwise|bind等，其中<sql>为sql片段标签，通过<include>标签引入sql片段，<selectKey>为不支持自增的主键生成策略标签。**

**10.不同的XML映射文件，如果配置了namespace，那么id可以重复；如果没有配置namespace，那么id不能重复；**

**原因是namespace+id是作为Map的key使用的，如果没有namespace，就剩下id，那么，id重复会导致数据互相覆盖。有了namespace，自然id就可以重复，namespace不同，那么namespace+id自然也就会不同。**

**11.Hibernate属于全自动的ORM映射工具，使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。而Mybatis在查询关联对象或关联集合对象时，需要手动编写SQL来完成，所以称之为半自动ORM映射工具。**

**12.有联合查询和嵌套查询，联合查询是几个表联合查询，只查询一次，通过在resultMap里配置association节点配置一对一的类就可以完成。**