**Spring简介**

**一、Spring是什么？**

在了解Spring之前，我们来了解在Java EE框架下企业级开发采用EJB框架的一些不足：

(1) EJB太笨重，而且Entity EJB不能脱离容器

(2) 企业级服务使用困难

(3) 开发的复杂度太高

(4) 侵入式方案，EJB要使用特定的接口

因此，Spring应运而生。

Spring是一个开源的用于简化采用Java语言开发企业级程序的一个分层的框架。

关于程序的分层结构：

1、Presentation layer（表示层）

(1) 表示逻辑（生成界面代码）

(2) 接收请求

(3) 处理业务层抛出的异常

(4) 负责规则验证（数据格式，数据非空等）

(5) 流程控制

2、Service layer（服务层/业务层）

(1) 封装业务逻辑处理，并且对外暴露接口

(2) 负责事务，安全等服务

3、Persistence layer（持久层）

(1) 封装数据访问的逻辑，暴露接口

(2) 提供方便的数据访问的方案（查询语言，API，映射机制等）

Domain layer（域层）

(1) 业务对象以及业务关系的表示

(2) 处理简单的业务逻辑

(3) 域层的对象可以穿越表示层，业务层，持久层

二、Spring的作用

为什么要使用Spring？

(1) 简化企业级开发

① 封装了大部分的企业级服务，提供了更好的访问这些服务的方式

② 提供了IOC，AOP功能的容器，方便编程

(2) 遵循Spring框架的应用程序，一定是设计良好的，针对接口编程，这样就简化了企业级程序的设计。

(3) Spring的组成

① Spring Core：核心容器，BeanFactory提供了组件生命周期的管理，组件的创建，装配，销毁等功能

SpringContext：ApplicationContext，扩展核心容器，提供事件处理、国际化等功能。它提供了一些企业级服务的功能，提供了JNDI，EJB，RMI的支持。

② Spring AOP：提供切面支持

③ Spring DAO：提供事务支持，JDBC，DAO支持

④ Spring ORM：对流行的O/R Mapping封装或支持

⑤ Spring Web：提供Web应用上下文，对Web开发提供功能上的支持，如请求，表单，异常等。

⑥ Spring Web MVC：全功能MVC框架，作用等同于Struts。

chapter02 Spring的IoC

一、IoC的概念

IoC，Inversion of Control，控制反转。

对象的协作关系由对象自己负责。

依赖注入：对象的协作关系有容器来建立。

二、IoC的类型

(1) 基于特定接口（侵入式方案）

(2) 基于set方法

(3) 基于构造器

三、Spring容器

Spring容器负责生成、组装、销毁组件，并负责事件处理、国际化等功能。

(1) BeanFactory<interface>

① 核心容器，负责组件生成和装配

② 实现主要包括Xml BeanFactory

(2) ApplicationContext

(3) WebApplicationContext

(4) ……

四、IoC的使用

Resource：interface，用来包装资源

xmlBeanFactory：BeanFactory的一个实现，使用Resource对象来查找配置文件

BeanFactory.gerBean（“BeanId”）：取得以参数命名，或者Id等于参数值的一个Bean实例。

BeanFactory（容器）在默认情况下，会采用单例方式返回对象。容器只到调用getBean方法时，才会实例化某个对象。

(1) Spring可以采用XML或者.properties文件作配置

(2) 配置文件（XML）

根元素<beans>可以有多个<bean>子元素，每个<bean>代表一个需要装配的对象。

1、setter注入

(1) 注入简单属性（String和8中基本类型）

<beans>

<bean id=”BeanId” class=”classpath” autowire=” ” dependency-check=” ”>

<property name=”parameterName”>

<value>parameterValue</value>

</property>

</bean>

</beans>

对于基本类型，Spring容器会自动作类型转换，以便赋值。

(2) 注入对象

<bean>

<ref local=”BeanId”>

</bean>

A、让Spring容器在当前配置文件中找到相应的Bean，并调用set方法，注入该Bean。

B、将一个Bean的定义嵌套在另一个Bean中（可读性差），被嵌套的Bean不能采用getBean（）返回

C、采用<ref bean=” ”>搜索多个配置文件来注入

(3) 注入集合类型

① Set

Set中存放字符串，对象，集合，不允许重复

② List

List中可以放入字符串，对象，List

③ Map

Map 有<entry>子元素来存取key，value，key只能为String

④ Properties

Properties有<props>子元素

2、consctructor注入

如果Bean属性不多，并且属性值必须要注入才能使用，则应该采用constructor注入，其他情况就要set方法注入。

装配关系检查（Dependency checking）

simple：检查简单类型

ojects：检查对象类型

all：检查所有

<bean dependency-check=“all”></bean>

自动装配（Autowring Properties）

装配方式：byName，byType，autodetect，constructor

autowire=“byName”：按照名称匹配

按照Bean的Id与属性的名称进行匹配

自动装配与手动装配可以结合使用，手动装配会覆盖自动装配。

autowire=“byType”：按照类型匹配

要注入的属性类型与配置文件中的Bean类型匹配的个数超过一个，会出错。

autowire=“antodetect”：

先按照construct，后按照byType。

autowire=“constructor”：

先去匹配构造器中参数类型，后与配置文件中的Bean类型匹配。

3、比较两种注入方式

关于自动匹配：

优点：快速开发

缺点：依赖关系不清楚，易出错，不易维护。

自动匹配的应用场合：

(1) 构建系统原型

(2) 与依赖关系检查（Dependency-check）结合使用

(3) 自动装配与手动装配结合

4、特殊的IoC

① 后处理Bean

接口：org.springframework.beans.factory.config.BeanPostProcessor

Spring已经实现该接口的BeanPostProcessor(不用再注册)

ApplicationContextAwareProcessor：

把应用上下文传递给所用实现了 ApplicationContextAware接口的Bean

ApplicationContextAware接口使用举例，可参照事件监听机制

DefaultAdvisorAutoProxyCreator自动对Bean应用切面

② Bean工厂后处理(只能在应用上下文中使用)

接口：org.springframework.beans.factory.config.BeanFactoryPostProcessor

Spring内部接口实现：

org.springframework.beans.factory.config.PropertyPlaceholderConfigurer

属性编辑

org.springframework.beans.factory.config.CustomEditorConfigurer

CustomerInfo("fname-lname-address")

必须实现org.springframework.context.ApplicationContextAware接口

Spring的AOP

一、AOP(Aspect-oriented programming,面向切面编程)：

什么是AOP？

定义：将程序中的交叉业务逻辑提取出来，称之为切面。将这些切面动态织入到目标对象，然后生成一个代理对象的过程。

二、AOP核心概念

1、Aspect（切面）

切面，是对交叉业务逻辑的统称。

2、Joinpoint（连接点）

连接点，指切面可以织入到目标对象的位置（方法，属性等）。

3、Advice（通知）

通知，指切面的具体实现。

4、Pointcut（切入点）

切入点，指通知应用到哪些类的哪些方法或属性之上的规则。

5、Introduction（引入）

引入，指动态地给一个对象增加方法或属性的一种特殊的通知。

6、Weaving（织入）

织入，指将通知插入到目标对象。

7、Target（目标对象）

目标对象，指需要织入切面的对象。

8、Proxy（代理对象）

代理对象，指切面织入目标对象之后形成的对象。

三、Spring AOP原理

采用动态代理模式。

Spring AOP采用动态代理的过程：

(1) 将切面使用动态代理的方式动态织入到目标对象（被代理类），形成一个代理对象；

(2) 目标对象如果没有实现代理接口，那么Spring会采用CGLib来生成代理对象，该代理对象是目标对象的子类；

(3) 目标对象如果是final类，并且也没实现代理接口，就不能运用AOP。

四、Spring的通知

1、Spring的通知类型

(1) MethodBeforeAdvice

类全名：org.springframework.aop.MethodBeforeAdvice

在方法调用之前，做处理

不能够改变返回值

不能够改变目标方法的流程，也不能中断流程的处理过程（除非抛出异常）

(2) AfterReturningAdvice

类全名：org.springframework.aop.AfterReturningAdvice

在方法调用之后，做处理。

不能够改变返回值

不能够改变目标方法的流程，也不能中断流程的处理过程（除非抛出异常）

(3) MethodInterceptor

类全名：org.aopalliance.intercept.MethodInterceptor

在方法调用之前以及之后，做处理。

可以改变返回值，也可以改变流程。

(4) ThrowsAdvice

类全名：org.springframework.aop.ThrowsAdvice

在方法抛出异常后，做处理。

当该通知处理完异常后，会简单地将异常再次抛出给目标调用方法。

2、配置过程：

（1）配置目标对象

（2）配置通知

（3）利用ProxyFactoryBean将通知织入到目标对象，形成一个动态代理对象

（4）客户端使用动态代理来访问目标对象的方法。

在默认情况下，通知会应用到所有的方法之上。

Pointcut：

根据方法和类决定在什么地方织入通知

Advisor：

将Pointcut与Advice结合到一起。

自定义切入点：

步骤：

1）实现org.springframework.aop.ClassFilter

2）实现org.springframework.aop.MethodMatcher

3）实现org.springframework.aop.Pointcut

4）实现org.springframework.aop.PointcutAdvisor

注意：

在此可定义

private Advice advice;

private Pointcut pointcut;

在配置文件中，将自定义的切入点与通知绑订到一起

5）利用ProxyFactoryBean将advisor织入到目标对象

ProxyFactoryBean的作用：

依照配置信息，将切面应用到目标对象，生成动态代理对象。

(1) Spring只支持方法连接点，不支持属性连接点。

（原因是Spring AOP采用的是动态代理模式，而动态代理本身就是在方法调用前加入代码。）

(2) Spring只在运行时，将切面织入到目标对象。

五、切入点（Pointcut）

1、Pointcut<interface>

切入点是指通知/切面可以应用到哪些类，哪些方法之上。

Pointcut API

Pointcut：org.springframework.aop.Pointcut

对某些类某些方法应用切面。

Classfilter：org.springframework.aop.ClassFilter

用来过滤类（哪些类可以应用切面）

MethodMather：org.springframework.aop.MethodMatcher

用来过滤方法（类中哪些方法应用切面）

Advisor：org.springframework.aop.PointcutAdvisor

将Pointcut与Advice结合到一起

2、预定义切入点

① 静态切入点：

a、NameMatchMethodPointAdviosr

org.springframework.aop.support.NameMatchMethodPointcutAdvisor

根据方法名称的特点进行匹配

核心XML：mappedName→advice(ref)

② 动态切入点：

org.springframework.aop.support.ControlFlowPointcut

③ 切入点的交叉与合并：

Pointcuts.union

④ Introduction

一种特殊类型的Advice，为类动态增加方法和属性。

编程步骤：

1)实现org.springframework.aop.IntroductionInterceptor或继承org.springframework.aop.support.DelegatingIntroductionInterceptor

2)使用org.springframework.aop.support.DefaultIntroductionAdvisor

六、自动代理

Spring在生成代理对象的时候，默认情况下，会使用被代理对象的接口来生成代理对象。

如果被代理对象没有实现接口，此时，Spring会使用CGLIB生成代理对象，此时该代理对象是被代理对象的子类。

a、BeanNameAutoProxyCreator

org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator

根据类的名称来为符合相应名称的类生成相应代理对象。

beanNames(list),interceptorNames

b、DefaultAdvisorAutoProxyCreator

org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator

自动将Advisor与匹配的Bean进行绑定

只能与Advisor配合使用

chapter04 Spring对持久层的支持

Spring对持久层的支持：① JDBC，② O/R Mapping（Hibernate，TopLink等）

一、Spring对持久层支持采用的策略：

1、Spring对持久层“不发明重复的轮子”，即没有重新实现新的持久层方案，对现有持久层方案做封装，更利于使用。

2、采用DAO模式

3、提供了大量的模板类来简化编程（HibernateDaoSupport，JdbcTemplate等）

4、重新设计了一套完善的异常体系结构

① 类型丰富，细化异常类型

② 全都是运行时异常（RuntimeException）

二、Spring对JDBC的支持

1、配置数据源

方式一：采用Spring内置的数据源，Spring内置实现 方式二：采用开源数据库产品如DBCP

方式三： 直接使用容器提供的数据源（如Tomcat，Weblogic，Sun Application Server）

2、配置JdbcTemplate模板类（封装了绝大多数数据库操作）

3、配置DAO

4、配置Service

三、Spring对Hibernate的支持

Step1： 配置数据源

Step2： 配置sessionfactory

Step3： 配置DAO

注意：以上配置是要求dao 继承HibernateDaoSupport

chapter05 Spring对事务的支持

一、AOP事务的含义：

事务当作一个切面，动态地织入到目标对象，形成一个代理对象。

二、Spring的事务机制

Spring支持声明式事务。

Spring使用事务服务代理和事务管理器（如HibernateTransactionManager）来支持事务服务。

Spring对事务的边界多了一种嵌套事务（PROPAGATION\_NESTED）。

PROPAGATION\_NESTED：

如果客户端启动了事务T1，那么Bean启动一个嵌套在T1中的子事务T2；

如果客户端没有启动事务，那么Bean会启动一个新的事务，类似于REQUIRED\_NEW

三、Spring中事务的使用

1、Spring中使用Hibernate事务

Step1：配置数据源

Step2：配置sessionfactory (同上)

Step3：配置事务管理器

<bean id="myTransactionManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory">

<ref bean="mySessionFactory" />

</property>

</bean>

Step4：创建事务服务代理

<bean id="saleService" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">

<property name="proxyInterfaces">

<value>lab7.SaleService</value>

</property>

<property name="transactionManager">

<ref bean="myTransactionManager" />

</property>

<property name="target">

<ref bean="saleServiceTarget" />

</property>

<property name="transactionAttributes">

<props>

<prop key="\*">PROPAGATION\_REQUIRED</prop>

</props>

</property>

</bean>

事务属性描述格式：

传播行为，隔离级别，只读事务(read-only)，回滚规则

在默认情况下，Spring的容器对于非受查异常（服务模块中抛出的非受查异常），会回滚事务。对于受查异常，会提交事务。

如果即使发生了某种受查异常，也要回滚事务，可以用 “- 异常类型“来声明。同样，对于非受查异常，如果不要求回滚事务，可以用"+异常类型"来声明。

如何简化事务配置？

使用继承（抽象的Service类）、自动代理。

四、Spring事务与EJB事务

1、EJB事务：

EJB的CMT管理事务方式，只能设置事务边界（传播行为），对于隔离性是不能设置的，并且EJB不支持嵌套事务。

2、Spring事务：

对于Spring来说， Spring的声明式事务可以设置事务边界（传播行为），设置隔离级别，设置只读事务，回滚规则（+：对于任何异常都提交，-：对于任何异常都回滚）

<property name=”transactionAttributes”>

<props>

<prop key=”\*”>+异常类型1，-异常类型2</prop>

</property>

PS：Spring对嵌套事务的支持依赖与数据库底层对嵌套式事务的支持。

chapter06 SSH整合

一、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。

① Web分层架构中业务层为什么都选择Spring？

Service层需要处理业务逻辑和交叉业务逻辑，处理事务，日志，安全等，而这些与Spring的IoC，AOP等不谋而合。

② Web分层架构中，对于各层技术的采用应该遵循一个怎样的标准？

1、选择发展成熟的技术：

A、经过了项目实践证实可行性良好

B、文档完善

C、技术一直处于可持续发展更新

2、Web应用中对于技术的选择有赖于开发人员的技术掌握情况

二、Spring与Struts整合

前提：

必须在Web应用启动时，创建Spring的ApplicationContext实例方式：

1、采用ContextLoaderListener来创建ApplicationContext：

<context-param>

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

<param-value>

/WEB-INF/spring-config/applicationContext.xml

</param-value>

</context-param>

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

2、采用ContextLoaderPlugIn来创建ApplicationContext

<plug-in className="org.springframework.web.struts.ContextLoaderPlugIn">

<set-property property="contextConfigLocation" value="/WEB-INF/config/sale.xml" />

</plug-in>

或者：通过listener装载spring应用上下文

方式一：通过Spring的ActionSupport类

ActionSupport类：

知道ApplicationContext的获得方式。

步骤：

1、Action直接继承ActionSupport

2、使用ApplicationContext ctx = getWebApplicationContext();取得Spring上下文

3、取得相应Bean

注意：有可能需要替换commons-attributes-compiler.jar包。

优点：简单

缺点： 耦合高

违反IOC

无法使用多方法的Action

方式二：通过Spring的DelegatingActionProxy类

步骤：

1、Action中，使用IOC获得服务

2、配置struts-config.xml

<action path="/somepath" type="org.springframework.web.struts.DelegatingActionProxy"/>

3、在Spring配置文件中

<bean name="/somepath" class="SomeAction">

<property name="service"><ref bean=""/>

</bean>

注意，要用bean name命名。

/somepath:Action的path

优点：

不使用Spring api编写 Action利用了IOC装配。

可以利用容器的scope="prototype"来保证每一个请求有一个单独的Action来处理，避免struts中Action的线程安全问题。

缺点：

struts配置文件中，所有path都映射到同一个代理类

方式三：通过Spring的DelegatingRequestProcessor类

步骤：

1、Action中，使用IOC获得服务

2、配置struts-config.xml

<controller processorClass="org.springframework.web.struts.DelegatingRequestProcessor" />

3、在Spring配置文件中

<bean name="/somepath" class="SomeAction">

<property name="service"><ref bean=""/>

</bean>

小结：

Spring与Struts整合方式只有两种：

（1）由Spring容器来管理Action（方式二，方式三）

（2）Action处于容器之外（方式一）

注意：

中文问题： 设置过滤器，设置页面编码，数据库编码

三、关于Spring与EJB

1、Spring与EJB3.0的对比

Spring与EJB3.0之间的关系是竞争关系。

① Spring是一个开源的框架，而EJB3.0是一个标准（标准意味着将得到广泛的支持以及良好的兼容性），并且，采用EJB3.0，项目的后期维护得到了保证。

② Spring是一个轻量级框架，EJB3.0是一个重量级框架（完整的容器，包含所有的服务）。

③ Spring的IoC，AOP集成了大量的开源框架，可扩展性良好。EJB3.0的可扩展性则完全依赖于新的容器。

④ Spring对事务支持不如EJB3.0成熟，Spring对集群的兼容也不够。

⑤ Spring与EJB3.0都是一个企业级开发框架，都支持声明式事务。

2、Spring的优势与劣势

Spring的优势：

① 简化了企业级开发（对企业级服务进行了进一步的封装）

② 采用Spring框架的程序意味着良好的分层结构设计，并保证是面向接口编程的

③ 用IoC，AOP容器，模块是可配置的，松耦合的，方便了后期维护

Spring的劣势：

① 配置复杂，不方便维护

② 容器大量使用反射等机制装配对象，影响性能，对于高并发的大型应用无能为力。

**Spring Web MVC**

Spring Web MVC是一种基于Java的实现了Web MVC设计模式的请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将web层进行职责解耦，基于请求驱动指的就是使用请求-响应模型。Spring Web MVC前端控制器是DispatcherServlet；应用控制器其实拆为处理器映射器(Handler Mapping)进行处理器管理和视图解析器(View Resolver)进行视图管理；页面控制器/动作/处理器为Controller接口（仅包含ModelAndView handleRequest(request, response) 方法）的实现（也可以是任何的POJO类）；支持本地化（Locale）解析、主题（Theme）解析及文件上传等；提供了非常灵活的数据验证、格式化和数据绑定机制；提供了强大的约定大于配置（惯例优先原则）的契约式编程支持。

1.用户向服务器发送请求，请求被Spring 前端控制Servelt DispatcherServlet捕获；  
2.DispatcherServlet对请求URL进行解析，得到请求资源标识符（URI）。然后根据该URI，调用HandlerMapping获得该Handler配置的所有相关的对象（包括Handler对象以及Handler对象对应的拦截器），最后以HandlerExecutionChain对象的形式返回；  
3.DispatcherServlet 根据获得的Handler，选择一个合适的HandlerAdapter。（附注：如果成功获得HandlerAdapter后，此时将开始执行拦截器的preHandler(...)方法）  
4.提取Request中的模型数据，填充Handler入参，开始执行Handler（Controller)。 在填充Handler的入参过程中，根据你的配置，Spring将帮你做一些额外的工作：  
    HttpMessageConveter： 将请求消息（如Json、xml等数据）转换成一个对象，将对象转换为指定的响应信息  
    数据转换：对请求消息进行数据转换。如String转换成Integer、Double等  
    数据根式化：对请求消息进行数据格式化。 如将字符串转换成格式化数字或格式化日期等  
    数据验证： 验证数据的有效性（长度、格式等），验证结果存储到BindingResult或Error中  
5.Handler执行完成后，向DispatcherServlet 返回一个ModelAndView对象；  
6.根据返回的ModelAndView，选择一个适合的ViewResolver（必须是已经注册到Spring容器中的ViewResolver)返回给DispatcherServlet ；  
7.ViewResolver 结合Model和View，来渲染视图  
8.将渲染结果返回给客户端

Spring MVC 九大组件

SpringMVC中的Servlet一共有三个层次，分别是HttpServletBean、FrameworkServlet和 DispatcherServlet。  
HttpServletBean直接继承自java的HttpServlet，其作用是将Servlet中配置的参数设置到相应的属性；  
FrameworkServlet初始化了WebApplicationContext，DispatcherServlet初始化了自身的9个组件。本篇博客的内容就是对9个组件做个概览。我们需要先了解Handler的概念，也就是处理器。它直接应对着MVC中的C也就是Controller层，它的具体表现形式有很多，可以是类，也可以是方法。在Controller层中@RequestMapping标注的所有方法都可以看成是一个Handler，只要可以实际处理请求就可以是Handler。  
【1. HandlerMapping】  
是用来查找Handler的。在SpringMVC中会有很多请求，每个请求都需要一个Handler处理，具体接收到一个请求之后使用哪个Handler进行处理呢？这就是HandlerMapping需要做的事。  
【2. HandlerAdapter】  
从名字上看，它就是一个适配器。因为SpringMVC中的Handler可以是任意的形式，只要能处理请求就ok，但是Servlet需要的处理方法的结构却是固定的，都是以request和response为参数的方法。如何让固定的Servlet处理方法调用灵活的Handler来进行处理呢？这就是HandlerAdapter要做的事情。  
小结：Handler是用来干活的工具；HandlerMapping用于根据需要干的活找到相应的工具；HandlerAdapter是使用工具干活的人。  
【3. HandlerExceptionResolver】  
其它组件都是用来干活的。在干活的过程中难免会出现问题，出问题后怎么办呢？这就需要有一个专门的角色对异常情况进行处理，在SpringMVC中就是HandlerExceptionResolver。具体来说，此组件的作用是根据异常设置ModelAndView，之后再交给render方法进行渲染。  
【4. ViewResolver】  
ViewResolver用来将String类型的视图名和Locale解析为View类型的视图。View是用来渲染页面的，也就是将程序返回的参数填入模板里，生成html（也可能是其它类型）文件。这里就有两个关键问题：使用哪个模板？用什么技术（规则）填入参数？这其实是ViewResolver主要要做的工作，ViewResolver需要找到渲染所用的模板和所用的技术（也就是视图的类型）进行渲染，具体的渲染过程则交由不同的视图自己完成。  
【5. RequestToViewNameTranslator】  
 ViewName是根据ViewName查找View，但有的Handler处理完后并没有设置View也没有设置ViewName，这时就需要从request获取ViewName了，如何从request中获取ViewName就是RequestToViewNameTranslator要做的事情了。RequestToViewNameTranslator在Spring MVC容器里只可以配置一个，所以所有request到ViewName的转换规则都要在一个Translator里面全部实现。  
【6. LocaleResolver】  
解析视图需要两个参数：一是视图名，另一个是Locale。视图名是处理器返回的，Locale是从哪里来的？这就是LocaleResolver要做的事情。LocaleResolver用于从request解析出Locale，Locale就是zh-cn之类，表示一个区域，有了这个就可以对不同区域的用户显示不同的结果。SpringMVC主要有两个地方用到了Locale：一是ViewResolver视图解析的时候；二是用到国际化资源或者主题的时候。  
【7. ThemeResolver】  
用于解析主题。SpringMVC中一个主题对应一个properties文件，里面存放着跟当前主题相关的所有资源、如图片、css样式等。SpringMVC的主题也支持国际化，同一个主题不同区域也可以显示不同的风格。SpringMVC中跟主题相关的类有 ThemeResolver、ThemeSource和Theme。主题是通过一系列资源来具体体现的，要得到一个主题的资源，首先要得到资源的名称，这是ThemeResolver的工作。然后通过主题名称找到对应的主题（可以理解为一个配置）文件，这是ThemeSource的工作。最后从主题中获取资源就可以了。  
【8. MultipartResolver】  
用于处理上传请求。处理方法是将普通的request包装成MultipartHttpServletRequest，后者可以直接调用getFile方法获取File，如果上传多个文件，还可以调用getFileMap得到FileName->File结构的Map。此组件中一共有三个方法，作用分别是判断是不是上传请求，将request包装成MultipartHttpServletRequest、处理完后清理上传过程中产生的临时资源。  
【9. FlashMapManager】  
用来管理FlashMap的，FlashMap主要用在redirect中传递参数。

注解  
组件类注解  
@Component ：标准一个普通的spring Bean类。   
@Repository：标注一个DAO组件类。   
@Service：标注一个业务逻辑组件类。   
@Controller：标注一个控制器组件类。

装配bean时常用的注解  
@Autowired：属于Spring 的org.springframework.beans.factory.annotation包下,可用于为类的属性、构造器、方法进行注值   
@Resource：不属于spring的注解，而是来自于JSR-250位于java.annotation包下，使用该annotation为目标bean指定协作者Bean。   
@PostConstruct 和 @PreDestroy 方法 实现初始化和销毁bean之前进行的操作

spring MVC模块注解  
@Controller ：表明该类会作为与前端作交互的控制层组件。  
@RequestMapping ： 这个注解用于将url映射到整个处理类或者特定的处理请求的方法。  
@RequestParam ：将请求的参数绑定到方法中的参数上。  
@PathVariable ： 该注解用于方法修饰方法参数，会将修饰的方法参数变为可供使用的uri变量（可用于动态绑定）。  
@RequestBody ： @RequestBody是指方法参数应该被绑定到HTTP请求Body上。  
@ResponseBody ： @ResponseBody与@RequestBody类似，它的作用是将返回类型直接输入到HTTP response body中。   
@RestController ：控制器实现了REST的API，功能类型@Controller+@ResponseBody   
@ModelAttribute ：@ModelAttribute可以作用在方法或方法参数上，当它作用在方法上时，标明该方法的目的是添加一个或多个模型属性（model attributes）。  
@SessionAttributes ：即将值放到session作用域中，写在class上面。  
@RequestHeader ：可以把Request请求header部分的值绑定到方法的参数上。  
@CookieValue ：可以把Request header中关于cookie的值绑定到方法的参数上。

Spring事务模块注解  
@Transactional

1、什么是Mybatis？

 （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，不需要花费精力去处理加载驱动、创建连接、创建statement等繁杂的过程。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。

（2）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，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

（3）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并将结果映射为java对象并返回。（从执行sql到返回result的过程）。

2、Mybaits的优点

（1）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里，解除sql与程序代码的耦合，便于统一管理；提供XML标签，支持编写动态SQL语句，并可重用。

（2）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；

（3）很好的与各种数据库兼容（因为MyBatis使用JDBC来连接数据库，所以只要JDBC支持的数据库MyBatis都支持）。

（4）能够与Spring很好的集成；

（5）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、MyBatis框架的缺点

（1）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。

（2）SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

4、MyBatis框架适用场合

（1）MyBatis专注于SQL本身，是一个足够灵活的DAO层解决方案。

（2）对性能的要求很高，或者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与Hibernate有哪些不同

（1）Mybatis和hibernate不同，它不完全是一个ORM框架，因为MyBatis需要程序员自己编写Sql语句。

（2）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，一但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。

（3）Hibernate对象/关系映射能力强，数据库无关性好，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、#{}和${}的区别是什么

  #{}是预编译处理，${}是字符串替换。

  ${}是Properties文件中的变量占位符，它可以用于标签属性值和sql内部，属于静态文本替换，比如${driver}会被静态替换为com.mysql.jdbc.Driver。

   #{}是sql的参数占位符，Mybatis会将sql中的#{}替换为?号，在sql执行前会使用PreparedStatement的参数设置方法，按序给sql的?号占位符设置参数值，比如ps.setInt(0, parameterValue)，#{item.name}的取值方式为使用反射从参数对象中获取item对象的name属性值，相当于param.getItem().getName()。

7、Mybatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复

  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；

  原因就是namespace+id是作为Map<String, MapperStatement>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，namespace+id自然也就不同。

8、为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里Hibernate属于全自动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。

1. MyBatis实现一对一有几种方式?具体怎么操作的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里面配置association节点配置一对一的类就可以完成；

  嵌套查询是先查一个表，根据这个表里面的结果的 外键id，再去另外一个表里面查询数据,也是通过association配置，但另外一个表的查询通过select属性配置。

10、MyBatis实现一对多有几种方式,怎么操作的

   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里面的collection节点配置一对多的类就可以完成；嵌套查询是先查一个表,根据这个表里面的结果的外键id,再去另外一个表里面查询数据,也是通过配置collection,但另外一个表的查询通过select节点配置。

11、MyBatis实现一对多有几种方式,怎么操作的

  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里面的collection节点配置一对多的类就可以完成；嵌套查询是先查一个表,根据这个表里面的 结果的外键id,再去另外一个表里面查询数据,也是通过配置collection,但另外一个表的查询通过select节点配置。

12、Mybatis动态sql有什么用？执行原理？有哪些动态sql？

      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。

13、Xml映射文件中，除了常见的select|insert|updae|delete标签之外，还有哪些标签？

   <resultMap>、<parameterMap>、<sql>、<include>、<selectKey>，加上动态sql的9个标签，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。

14、Mybatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复？

       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；

      原因就是namespace+id是作为Map<String, MapperStatement>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，namespace+id自然也就不同。

15、通常一个Xml映射文件，都会写一个Dao接口与之对应，请问这个Dao接口的工作原理是什么？Dao接口里的方法，参数不同时，方法能重载吗？

      Dao接口即Mapper接口。接口的全限名，就是映射文件中的namespace的值；接口的方法名，就是映射文件中Mapper的Statement的id值；接口方法内的参数，就是传递给sql的参数。

      Mapper接口是没有实现类的，当调用接口方法时，接口全限名+方法名拼接字符串作为key值，可唯一定位一个MapperStatement。在Mybatis中，每一个<select>、<insert>、<update>、<delete>标签，都会被解析为一个MapperStatement对象。

16、MyBatis与Hibernate有哪些不同？

.      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

      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，数据库无关性好，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。

17、Mybatis的一级、二级缓存

（1）一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当 Session flush 或 close 之后，该 Session 中的所有 Cache 就将清空，默认打开一级缓存。

（2）二级缓存与一级缓存其机制相同，默认也是采用 PerpetualCache，HashMap 存储，不同在于其存储作用域为 Mapper(Namespace)，并且可自定义存储源，如 Ehcache。默认不打开二级缓存，要开启二级缓存，使用二级缓存属性类需要实现Serializable序列化接口(可用来保存对象的状态),可在它的映射文件中配置<cache/> ；

（3）对于缓存数据更新机制，当某一个作用域(一级缓存 Session/二级缓存Namespaces)的进行了C/U/D 操作后，默认该作用域下所有 select 中的缓存将被 clear。

18、使用MyBatis的mapper接口调用时有哪些要求？

（1）Mapper接口方法名和mapper.xml中定义的每个sql的id相同；

（1）Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同；

（1）Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同；

（1） Mapper.xml文件中的namespace即是mapper接口的类路径。