 浅谈Action+Service +Dao 功能

1. Action/Service/DAO简介：

Action是管理业务（Service）调度和管理跳转的。

Service是管理具体的功能的。

Action只负责管理，而Service负责实施。

DAO只完成增删改查，虽然可以1-n，n-n，1-1关联，模糊、动态、子查询都可以。但是无论多么复杂的查询，dao只是封装增删改查。至于增删查改如何去实现一个功能，dao是不管的。

总结这三者，通过例子来解释：

Action像是服务员，顾客点什么菜，菜上给几号桌，都是ta的职责；

Service是厨师，action送来的菜单上的菜全是ta做的；

Dao是厨房的小工，和原材料打交道的事情全是ta管。

相互关系是，小工(dao)的工作是要满足厨师(service)的要求，厨师要满足服务员(action)转达的客户（页面用户）的要求，服务员自然就是为客户服务喽。

现在最基本的分层方式，结合了SSH架构。Model层就是对应的数据库表的实体类。Dao层是使用了Hibernate连接数据库、操作数据库（增删改查）。Service层：引用对应的Dao数据库操作。Action层：引用对应的Service层，在这里结合Struts的配置文件，跳转到指定的页面，当然也能接受页面传递的请求数据，也可以做些计算处理。

以上的Hibernate, Struts，都需要注入到Spring的配置文件中，Spring把这些联系起来，成为一个整体。

2. 三大框架Struts/Hibernate/Spring

简单地说：

Struts——控制用的；

Hibernate——操作数据库的；

Spring——解耦用的。

详细地说：

Struts在SSH框架中起控制的作用，其核心是Controller，即ActionServlet，而ActionServlet的核心就是Struts-config.xml，主要控制逻辑关系的处理。

Hibernate是数据持久化层，是一种新的对象、关系的映射工具，提供了从Java类到数据表的映射，也提供了数据查询和恢复等机制，大大减少数据访问的复杂度。把对数据库的直接操作，转换为对持久对象的操作。

Spring是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架。面向接口的编程，由容器控制程序之间的依赖关系，而非传统实现中，由程序代码直接操控。这就是所谓“控制反转”的概念所在：（依赖）控制权由应用代码中转到了外部容器，控制权的转移，是所谓反转。依赖注入，即组件之间的依赖关系由容器在运行期决定，形象地说，即由容器动态地将某种依赖关系注入到组件之中，起到的主要作用是解耦。

Struts、Spring、Hibernate在各层的作用：

（1）Struts负责Web层：ActionFormBean接收网页中表单提交的数据，然后通过Action进行处理，再Forward到对应的网页。在Struts-config.xml中定义<action-mapping>，ActionServlet会加载。

（2） Spring负责业务层管理，即Service（或Manager）。

Service为action提供统计的调用接口，封装持久层的DAO；

可以写一些自己的业务方法；

统一的Javabean管理方法；

声明式事务管理；

集成Hibernate。

（3）Hibernate，负责持久化层，完成对数据库的crud操作。提供OR/Mapping。它由一组.hbm.xml文件和POJO，是跟数据库中的表相对应的。然后定义DAO，这些是跟数据库打交道的类，它们会使用PO。

3. 框架业务逻辑分析：

在Struts + Spring + Hibernate的系统中，

对象的调用流程是：JSP—Action—Service—DAO—Hibernate。

数据的流向是：ActionFormBean接受用户的数据，Action将数据从ActionFormBean中取出，封装成VO或PO，再调用业务层的Bean类，完成各种业务处理后再Forward。而业务层Bean收到这个PO对象之后，会调用DAO接口方法，进行持久化操作。

SSH框架的优点：

Hibernate的最大好处就是根据数据库的表，反向生成实体类，并且还有关系在里面，还有就是它对数据的操作也很方便；

Spring，省去了在类里面new对象的过程，把这个调用与被调用的关系直接展示到了配置文件里，做任何操作都变得简单了。

简单流程举例说明：

程序框架搭建好，并且把各种jar包导入后，就开始进行业务逻辑分析——

假设一个最基本的注册功能：页面有两个文本框，一个用户名(username)和一个密码(password)。以QQ注册网页说明，这里以昵称和密码为代表进行举例。

首先是action层：它是负责在页面和程序之间传输数据的，还有作用是做页面跳转。页面由用户填写表单数据，点击提交按钮，页面的表单数据由Hibernate自动封装到该页面表单所对应的ActionFrom（ActionFrom跟实体类不是一个东西，ActionFrom是页面有什么值，类里就写什么属性，是用来封装表单数据用的；而实体类是完全按照数据库的字段生成的，实体类可以当做ActionFrom用，但ActionFrom绝对不可以当做实体类用），这样表单数据就以ActionFrom对象的形式在Action的点击“提交按钮”执行的那个方法里存在了。这个时候需要做的就是把表单数据存入数据库中。此时，Action的功能告一段落，接着是把数据传入BIZ层。

BIZE层（业务逻辑层）：负责的是对数据的处理。如果没有数据处理任务的话，此层只做单纯的数据传递作用，而后又到了DAO层。

DAO层（数据库操作层）：负责对数据向数据库增删改查的操作。

详解spring （面向切面编程，依赖注入（控制反转））

面向切面编程 是对业务处理过程中的切面进行提取 （面对的是过程中的某个步骤或阶段 以获得逻辑过程中各部分之间降低耦合度的隔离效果

Spring，无需直接创建类的实例，而是通过Spring容器获取类的实例。(类似工厂模式) Spring容器根据配置文件信息，负责创建对象的实例，并为对象实例设置属性值.---这种由Spring容器为对象设置属性的方式被称为控制反转(Inversion of Control, IOC).

Spring 的核心机制：依赖注入 Spring通过配置文件为对象属性赋值，称之为注入。 如果A对象调用了B对象的方法，我们可以称A对象依赖于B对象。 所以Spring容器为Bean注入属性，称之为依赖注入。 而且，Spring还能为Bean注入其他Bean的引用。(为某个java对象实例提供其他需要的java实例).

控制反转与依赖注入，含义相同: 当某个java实例需要另外一个java实例，在传统的程序设计过程中，通常由某个调用者来创建被调用者的实例。 而在依赖注入的模式下，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是由Spring容器来完成，然后注入调用者，因此也称为依赖注入。

案例：人需要一把斧子. 1.调用者直接使用new关键字创建被调用的java实例，高耦合，效率低下。 2.面向接口编程，调用者只需找到符合某种标准（接口）的实例，即可使用。此时，调用的代码面向接口编程，可以让调用者和被调用者解耦，但是调用者依然需要主动定位工厂，与工厂耦合。 3. 调用者无需理会被调用者的实现，也无须定位工厂，这是最好的解耦方式。

Spring 强调面向接口编程. 采用setter方法为目标bean注入所以来的Bean，这种方式称为设值注入。

还有一种方式是通过构造函数来设置依赖关系，称为构造注入.

一般以设值注入为主，构造注入为辅， 在依赖关系无须变化的注入，尽量采用构造注入。

Spring 的关键就是配置文件 xml。 7.5.9 组合属性只有最后一个属性才是调用setter方法，前面各个属性实际上对应于调用getter方法. 所以前面的属性都不能是null. eg: 对应: a.getFoo().getBar().getX().setY("xx");

组件和组件之间的耦合，采用依赖注入管理，但是普通的javaBean属性值，应直接在代码中设置。

8.1.2 Bean 后处理器的用处 （其实就是对定义好的Bean做自定义操作） BeanNameAutoProxyCreator:根据Bean实例的name属性，创建Bean实例的代理。 DefaultAdvisorAutoProxyCreator: 根据提供的Advisor，对容器所有的Bean实例创建代理。 用于根据容器配置的拦截器，创建代理Bean， 代理Bean就是对目标Bean进行增强，在目标Bean的基础上进行修改得到新的bean。 \*如果需要对容器中某一批Bean进行通用的增强处理，则可以考虑使用Bean后处理器。

8.1.3 容器 后处理器 容器后处理器负责处理容器本身 必须实现BeanFactoryPostProcessor接口，(必须实现postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory)). 若在代码中使用ApplicationContext作为Spring容器，那么该Spring容器会自动搜索容器中所有实现了BeanPostProcessor接口的类，并将它注册成容器后处理器;(不用自己在xml里配置).

PropertyPlaceholderConfigurer 后处理器，负责读取Properties属性文件里的属性值，并将这些属性值设置成Spring配置文件的元素据 (比如数据库.properties)，可以将数据库连接信息从Spring配置文件中分离，Spring容器可以从properties文件读取需要的信息。

8.2. Spring零配置. 摒弃配置，使用Annotation来标注Bean类: @Component: 普通的Spring Bean类 @Controller: 控制器组件类 @Service: 标注一个业务逻辑组件 @Repository: 标记一个DAO组件类. 但是仍然需要一个配置文件xml. 通过语句: <context:component-scan base-package = "org.xx.package"/> 扫描制定包下所有的bean.