**使用注解来构造IoC容器**

**用注解来向Spring容器注册Bean。需要在applicationContext.xml中注册<context:component-scan base-package=”pagkage1[,pagkage2,…,pagkageN]”/>**。

如：在base-package指明一个包

1 <context:component-scan base-package="cn.gacl.java"/>

表明**cn.gacl.java**包及其子包中，如果某个类的头上带有特定的注解【@Component/@Repository/@Service/@Controller】，就会将这个对象作为Bean注册进Spring容器。也可以在<context:component-scan base-package=” ”/>中指定多个包，如：

1 <context:component-scan base-package="cn.gacl.dao.impl,cn.gacl.service.impl,cn.gacl.action"/>

多个包逗号隔开。

**1、@Component**

@Component  
是所有受Spring 管理组件的通用形式，@Component注解可以放在类的头上，@Component不推荐使用。

**2、@Controller**

@Controller对应表现层的Bean，也就是Action，例如：

1 @Controller

2 @Scope("prototype")

3 public class UserAction extends BaseAction<User>{

4 ……

5 }

**使用@Controller注解标识UserAction之后，就表示要把UserAction交给Spring容器管理，在Spring容器中会存在一个名字为"userAction"的action，这个名字是根据UserAction类名来取的。注意：如果**@Controller**不指定其**value【@Controller】，**则默认的bean名字为这个类的类名首字母小写，如果指定**value【@Controller(value="UserAction")】**或者**【@Controller("UserAction")】，**则使用value作为bean的名字**。

这里的UserAction还使用了@Scope注解，@Scope("prototype")表示将Action的范围声明为原型，**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。spring 默认scope 是单例模式(scope="singleton")，这样只会创建一个Action对象，每次访问都是同一Action对象，数据不安全，struts2 是要求每次次访问都对应不同的Action，scope="prototype" 可以保证当有请求的时候都创建一个Action对象**

**3、@ Service**

@Service对应的是业务层Bean，例如：

1 @Service("userService")

2 public class UserServiceImpl implements UserService {

3 ………

4 }

@Service("userService")注解是告诉Spring，当Spring要创建UserServiceImpl的的实例时，bean的名字必须叫做"userService"，这样当Action需要使用UserServiceImpl的的实例时,就可以由Spring创建好的"userService"，然后注入给Action：在Action只需要声明一个名字叫“userService”的变量来接收由Spring注入的"userService"即可，具体代码如下：

1 // 注入userService

2 @Resource(name = "userService")

3 private UserService userService;

注意：在Action声明的“userService”变量的类型必须是“UserServiceImpl”或者是其父类“UserService”，否则由于类型不一致而无法注入，由于Action中的声明的“userService”变量使用了@Resource注解去标注，并且指明了其name = "userService"，这就等于告诉Spring，说我Action要实例化一个“userService”，你Spring快点帮我实例化好，然后给我，当Spring看到userService变量上的@Resource的注解时，根据其指明的name属性可以知道，Action中需要用到一个UserServiceImpl的实例，此时Spring就会把自己创建好的名字叫做"userService"的UserServiceImpl的实例注入给Action中的“userService”变量，帮助Action完成userService的实例化，这样在Action中就不用通过“UserService userService = new UserServiceImpl();”这种最原始的方式去实例化userService了。如果没有Spring，那么当Action需要使用UserServiceImpl时，必须通过“UserService userService = new UserServiceImpl();”主动去创建实例对象，但使用了Spring之后，Action要使用UserServiceImpl时，就不用主动去创建UserServiceImpl的实例了，创建UserServiceImpl实例已经交给Spring来做了，Spring把创建好的UserServiceImpl实例给Action，Action拿到就可以直接用了。Action由原来的主动创建UserServiceImpl实例后就可以马上使用，变成了被动等待由Spring创建好UserServiceImpl实例之后再注入给Action，Action才能够使用。这说明Action对“UserServiceImpl”类的“控制权”已经被“反转”了，原来主动权在自己手上，自己要使用“UserServiceImpl”类的实例，自己主动去new一个出来马上就可以使用了，但现在自己不能主动去new“UserServiceImpl”类的实例，new“UserServiceImpl”类的实例的权力已经被Spring拿走了，只有Spring才能够new“UserServiceImpl”类的实例，而Action只能等Spring创建好“UserServiceImpl”类的实例后，再“恳求”Spring把创建好的“UserServiceImpl”类的实例给他，这样他才能够使用“UserServiceImpl”，这就是Spring核心思想“**控制反转**”，也叫“**依赖注入**”，“依赖注入”也很好理解，Action需要使用UserServiceImpl干活，那么就是对UserServiceImpl产生了依赖，Spring把Acion需要依赖的UserServiceImpl注入(也就是“给”)给Action，这就是所谓的“依赖注入”。对Action而言，Action依赖什么东西，就请求Spring注入给他，对Spring而言，Action需要什么，Spring就主动注入给他。

**4、@ Repository**

@Repository对应数据访问层Bean ，例如：

1 @Repository(value="userDao")

2 public class UserDaoImpl extends BaseDaoImpl<User> {

3 ………

4 }

@Repository(value="userDao")注解是告诉Spring，让Spring创建一个名字叫“userDao”的UserDaoImpl实例。

当Service需要使用Spring创建的名字叫“userDao”的UserDaoImpl实例时，就可以使用@Resource(name = "userDao")注解告诉Spring，Spring把创建好的userDao注入给Service即可。

1 // 注入userDao，从数据库中根据用户Id取出指定用户时需要用到

2 @Resource(name = "userDao")

3 private BaseDao<User> userDao;