**Spring**

* [**Page History**](https://github.com/springside/springside4/wiki/Spring/_history)

**1.用Profile 统一环境配置**

Spring 3.1的功能，以后就不用为了区分Test, Dev, Production环境，搞几个只有细微区别的application.xml, application-test.xml及引用它们的web.xml了。

1. 首先，将applicationContext.xml中的namespace从3.0升级到3.1.xsd， 然后就可以在文件末尾加入不同环境的定义，比如不同的dataSource

<beans profile="test">

<jdbc:embedded-database id="dataSource">

<jdbc:script location="classpath:com/bank/config/sql/schema.sql"/>

</jdbc:embedded-database>

</beans>

<beans profile="production">

<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/datasource"/>

</beans>

2.在web.xml里，你需要定义使用的profile，最聪明的做法是定义成context-param，注意这里定义的是**default值**，在非生产环境，可以用系统变量"spring.profiles.active"进行覆盖。

<context-param>

<param-name>spring.profiles.default</param-name>

<param-value>production</param-value>

</context-param>

3.在其他地方进行覆盖

3.1 在development和functional test启动Jetty前设置系统变量

System**.**setProperty**(**"spring.profiles.active"**,** "development"**);**

server**.**start**()**

3.2 在用到ApplicationContext的单元测试用例中，用 @ActiveProfiles定义

@ContextConfiguration**(**locations **=** **{** "/applicationContext.xml" **})**

@ActiveProfiles**(**"test"**)**

**public** **class** **AccountDaoTest** **extends** SpringTxTestCase **{**

**}**

在springside里有演示了production,development，test,functional三个环境， 大家可以根据实际情况组合自己的环境管理。另外可以与Spring的properties文件加载时可顺位覆盖的特性(放一些不在版本管理中的xx.local.properties文件)，更好的支持本地开发环境，Jenkins上的functional test等其他环境。

**2.配置方式**

Spring3.0的精力都在Java Config和Spring EL上，但暂时没看到很适合使用的场景。

配置还是建议老规矩，Dao，Manager, Controller, WebService这些类似的，配置要求不高的Bean就用@Component标注并使用@Autowired注入依赖.其他独立性比较强的就用XML配置并在XML里显式的说明注入关系。

**3.注入配置？**

**用@Autowired还是@Resource?**

@Autowired是Spring自己定义的，按属性的Class注入， @Resource是JSR规范，按属性的名称注入，个人更喜欢用前者，因为@Autowired的耦合度更低，而且还有一个required=true/false的有用属性。

**在setter方法上注释还是直接在属性上注释?**

有些情况还是需要提供setter方法，比如要在XML里显式的配置注入，比如要在单元测试用例里，向待测试Bean注入Mock对象。除此之外，都可以省略掉setter方法。 所以在SpringSide中，Web层的POJO一般没有setter对象，Business层的就会有。

**4.AOP陷阱**

* 一般用基于CGI-Lib的AOP<tx:annotation-driven proxy-target-class="true" />， 那如果用到<aop:aspectj-autoproxy/>也应该加上proxy-target-class="true"，另外shiro的方法级AOP用到的DefaultAdvisorAutoProxyCreator也同样需要设置target-class属性为true。
* Aspectj只能用1.6.9版本，不能升级到最新版....