**Spring+SpringMVC配置加载顺序**

DispatcherServlet是SpringMVC的核心。在这里请求会第一次接触到框架，它要负责将请求路由到其他的组件之中。

它的配置有XML方式和JavaConfig方式

**JavaConfig的基本原理：**

在Servlet3.0环境中(12/2009发布 Tomcat7以上版本)，容器会在classpath中查找实现了javax.servlet.ServletContainerInitializer(SC)接口的类，并用它来配置Servlet容器。

Spring提供了这个接口的实现，名为SpringServletContainerInitializer(SSCI)，这个类反过来又会查找实现WebApplicationInitializer的类并将配置的任务交给它们来完成。Spring 3.2引入了一个便利的WebApplicationInitializer基础实现，对注解方式的实现来说，也就是AbstractAnnotationConfigDispatcherServletInitializer(AACDSI)。

AACDSI名字很长，一般我们使用它的子类WebMvcConfigurerAdapter，主要有如下方法：

**1.String[] getServletMappingg()：**

将一个或多个路径映射到Dispatcher-Servlet上，SpringMVC一般都是映射到**/**。一个斜杠表示默认的servlet映射，也就是说，没有人处理的servlet会移交到此处。如果设置成/\*，则所有请求都会被拦截，.jsp也不例外，不是我们要的结果。

**2.void configureDefualtServletHandling(DefaultServletHandlerConfigurer config)：**

如果通过1的方法，我们把所有请求都发给了DS(包括jsp/html/...)，DispatcherServlet找不到处理对应Servlet的控制器，对html/jsp的查找，

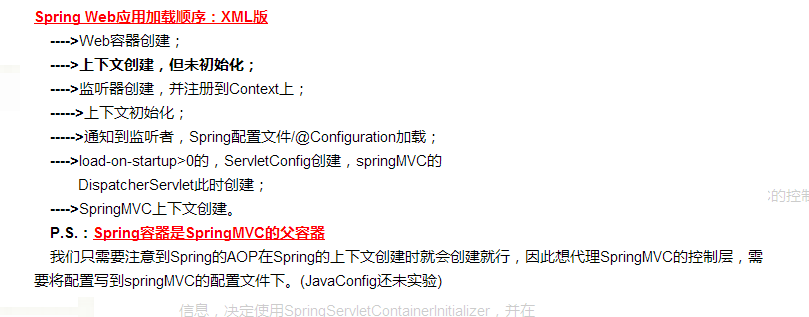
显然这应该交给Tomcat处理。故一般直接调用config.enable()即可。

**3.Class<?>[] getRootConfigClasses()：**返回Spring的配置类们，需有@Configuration标注。

**4.Class<?>[] getServletConfigClasses()：**返回SpringMVC的配置类们，需有@Configuration标注。

**5.VeiwResolver veiwResolver()：**配置视图解析器，一般用InternalResourceViewResolver，默认使用BeanNameResolver。

**创建过程解析：**





----->其他操作，一一处理完毕后，Servlet容器会对Context进行初始化，此时会通知监听器。

SpringInAction的示例代码：

