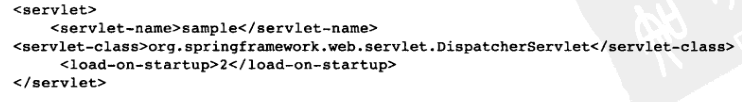
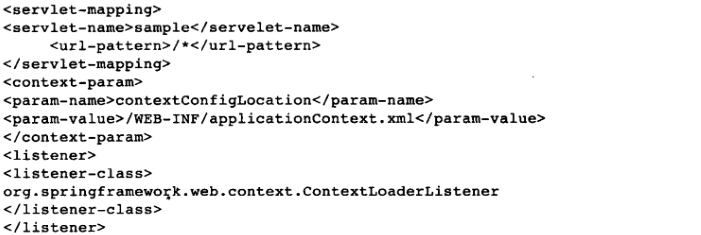
**1.web环境中的spring mvc**

Web.xml文件中的配置



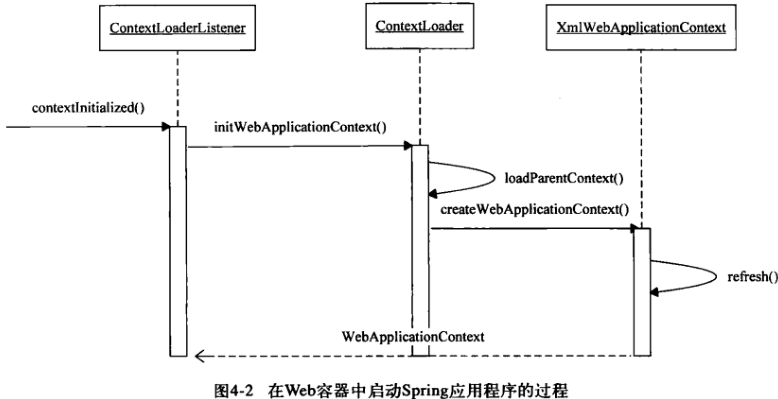


**首先**定义了一个servlet对象，它是Spring mvc的dispatcherServlet，起着分发请求的作用。**同时**通过servlet-mapping为dispatcherServlet定义了对应的URL映射，指定了需要处理的HTTP请求。**其次**，Context-param参数的配置用来指定Spring IoC容器读取Bean定义的xml文件，这里是/WEB-INF/applicationContext.xml。**最后**作为Spring mvc 的启动类，ContextLoaderListener被定义为一个监听器，与WEB服务器的生命周期相关联，由该监听器完成IoC容器在web环境中的启动工作。

DispatcherServlet和ContextLoaderListener提供了web容器与Spring交互的接口，通过servletContext实现，servletContext提供了一个宿主环境，Spring mvc建立一个IoC容器体系，这个体系通过ContextLoaderListener的初始化来建立的，在建立IoC容器体系后，把DispatcherServlet作为Spring mvc处理web请求的转发器建立起来，从而完成响应HTTP请求的准备。

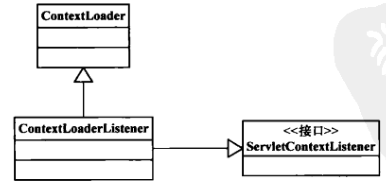
**2.上下文在web容器中的启动**

**2.1 IoC容器启动的基本过程**



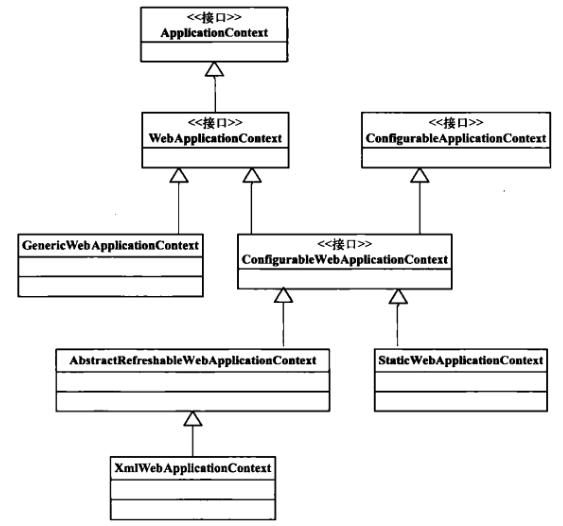
ContextLoaderListener启动的上下文为根上下文，还有一个与web mvc相关的上下文用来保存控制器dispatcherServlet需要的mvc对象，作为根上下文的子上下文，构成一个层次化的上下文体系。在web.xml中，配置了ContextLoaderListener，这个contextLoaderListener是Spring提供的类，是为了在web容器中建立IoC容器提供服务的，它实现了ServletContextListener接口；提供了与servlet生命周期结合的回调，比如contextInitialized和contextDestroyed方法。建立webApplicationContext的过程，是在contextInitialized的接口实现中完成的。

IoC容器的载入的过程由ContextLoaderListener交给ContextLoader来完成，而ContextLoader本身就是ContextLoaderListener的基类；

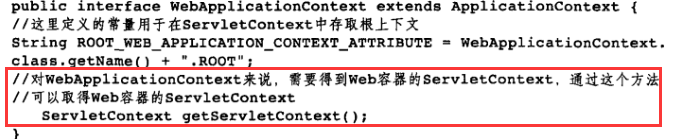


ContextLoader中，完成了两个IoC容器建立的基本过程，一个是在web容器中建立起双亲IoC容器，另一个是生成相应的WebApplicationContext并将其初始化。

**2.2web容器红的上下文设计**

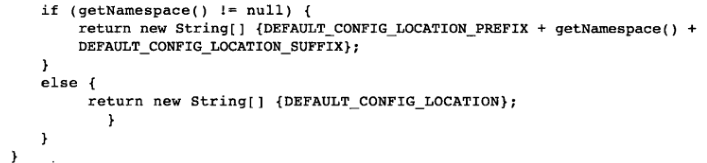


WebApplicationContext定义了getServletContext方法，这个方法可以得到当前web容器的servlet上下文环境。



启动过程中，spring会使用一个默认的WebApplicationContext实现作为IoC容器，为XmlWebApplicationContext；在ApplicationContext功能的基础上，增加了对web环境和xml配置定义的处理。





基本上下文已经通过类的继承获得，这里需要处理，如何获取BeanDefinition，默认从WEB-INF/applicationContext.xml中获得BeanDefinition，获得之后，就是通过BeanDefinitionReader来载入BeanDefinition的，最终完成整个Context的初始化过程。

**2.3 ContextLoader的设计与实现**

ContextLoaderListener通过使用ContextLoader来完成实际的WebApplicationContext，即IoC容器的初始化工作；首先从servlet时间中得到ServletContext，然后可以读取配置在web.xml中的各个相关的属性值，接着ContextLoader会实例化WebApplicationContext，并完成其载入和初始化过程。这个被初始化的第一个上下文作为根上下文，根上下文被载入后，被绑定到web应用程序的servletContext上。任何需要访问根上下文的代码都可以从WebApplicationContextUtils类的静态方法中得到



ContextLoaderListener中，实现的是ServletContextListener接口，这个接口里的函数会结合web容器的生命周期被调用。因为ServletContextListener是ServletContext的监听者，如果ServletContext发生变化，会触发出相应的事件，而监听器一直在对这个事件进行监听，如果接收到监听的时间，就会做出预先设计好的响应动作。根据ServletContext变化而触发的监听器响应：服务器启动时，ServletContext被创建时；服务器关闭时；ServletContext将要被销毁的时候等；对应方法：contextInitialized，contextDestroyed等。



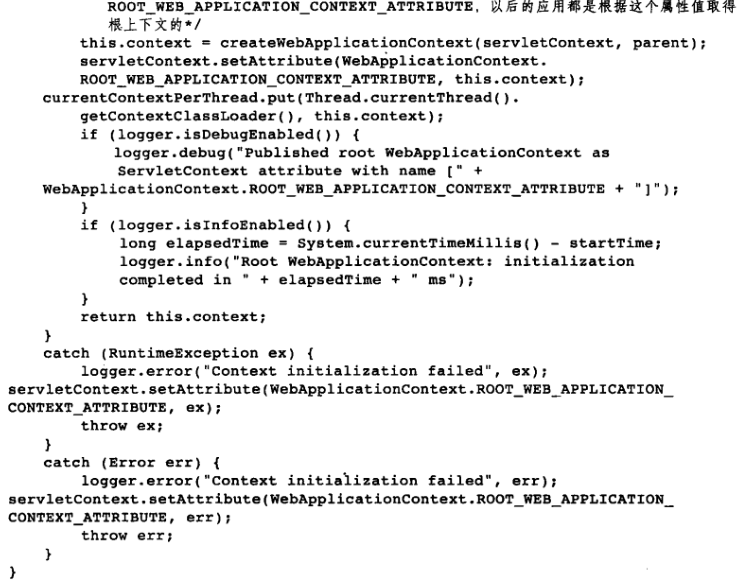
-----------------------------------------------------------------------------

WebApplicationContext作为web容器中唯一的实例而存在的，如果在初始化过程中，WebApplicationContext已经被创建了，这里会抛出异常提示创建失败。WebApplicationContext创建成功以后，会被存到web容器中的servletContext中去。存储路径是由Spring预先设置好的，ROOT\_WEB\_APPLICATION\_CONTEXT\_ATTRIBUTE的属性中定义的这个路径，默认如下

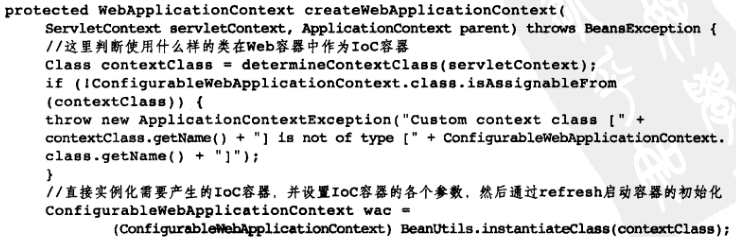


-------------------------------------------------------------------------------





------------------------------------------------------------------------





上面表示对WebApplicationContext的创建，设置双亲上下文、对servletContext的引用等；完成基本的设置后，就调用refresh方法，重启IoC容器。

使用什么样的类作为上下文是在determineContextClass方法中进行确定的，这个操作通过CONTEXT\_CLAS\_PARAM参数的设置来完成，如果没有指定特定的IoC容器，将使用默认的IoC容器，即XmlWebApplicationContext对象作为web环境中使用的IoC容器。

