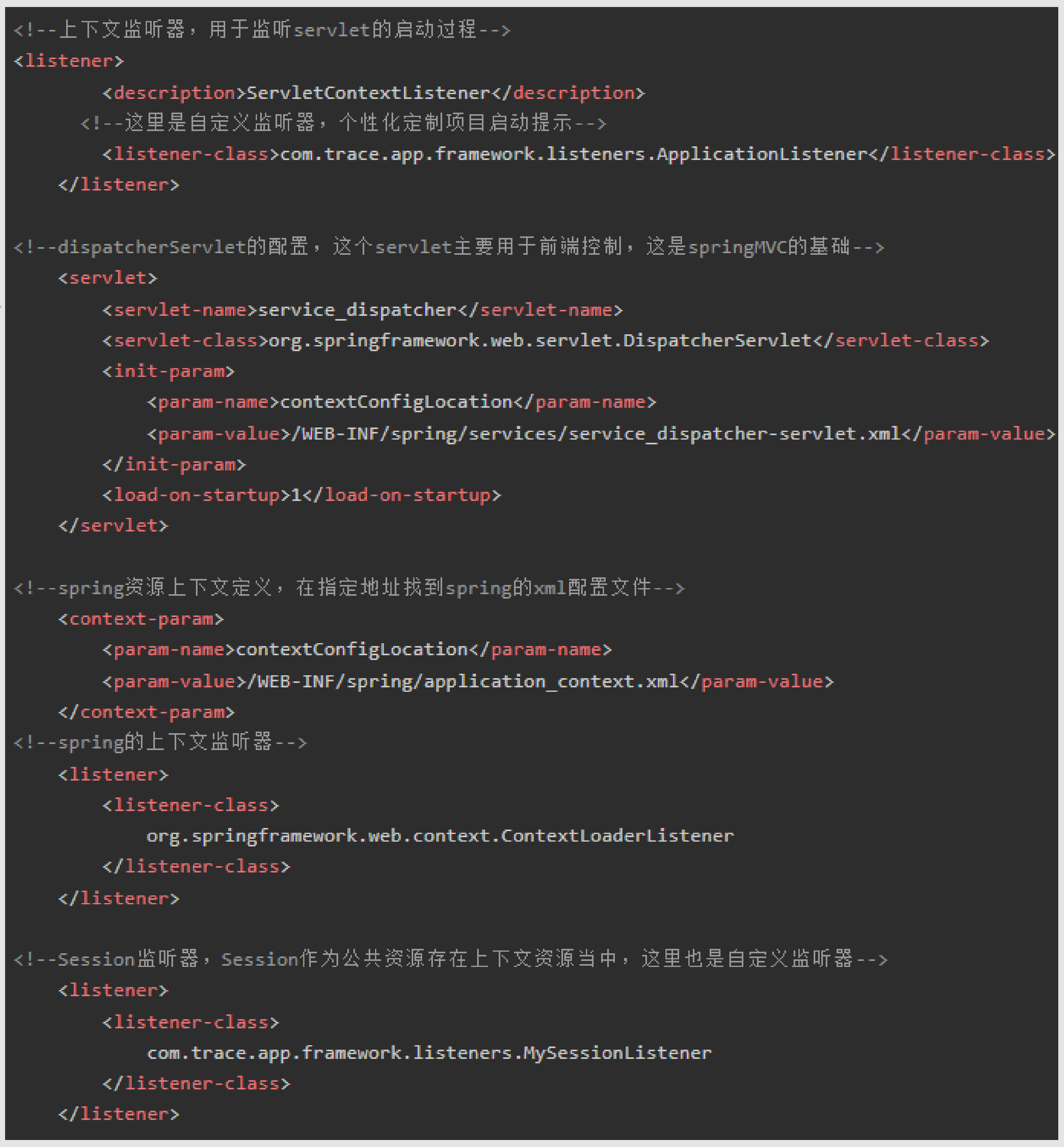
Spring的启动流程

spring的启动是建筑在servlet容器之上的

初始位置就是**web.xml**

web.xml中
配置了servlet的上下文（context）和监听器（Listener）



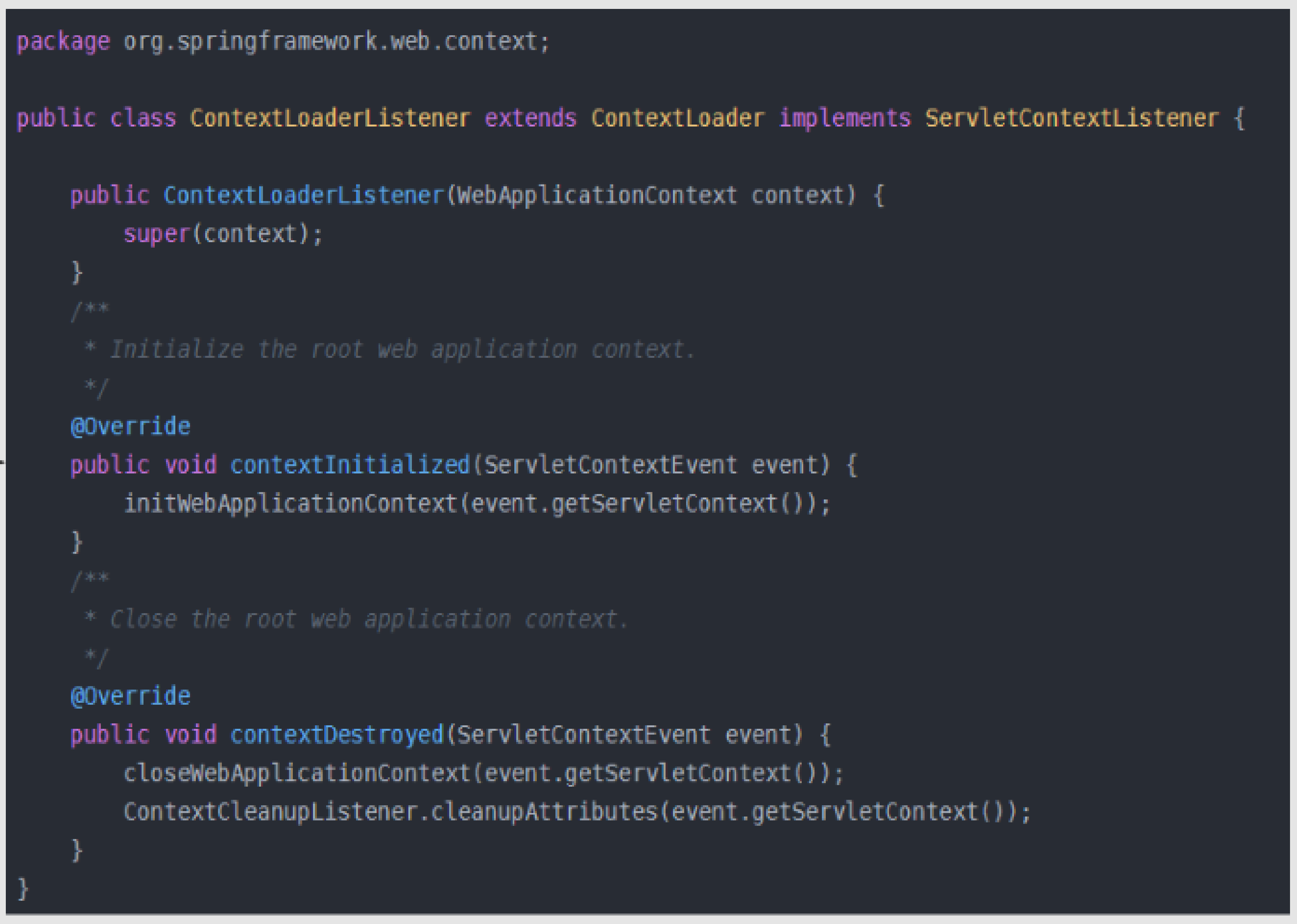
```
<!--上下文监听器，用于监听servlet的启动过程-->
<listener>
        <description>ServletContextListener</description>
      <!--这里是自定义监听器，个性化定制项目启动提示-->
        <listener-class>com.trace.app.framework.listeners.ApplicationListener</listener-class>
    </listener>

<!--dispatcherServlet的配置，这个servlet主要用于前端控制，这是springMVC的基础-->
    <servlet>
        <servlet-name>service_dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/services/service_dispatcher-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

<!--spring资源上下文定义，在指定地址找到spring的xml配置文件-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/application_context.xml</param-value>
    </context-param>
<!--spring的上下文监听器-->
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

<!--Session监听器，Session作为公共资源存在上下文资源当中，这里也是自定义监听器-->
    <listener>
        <listener-class>
            com.trace.app.framework.listeners.MySessionListener
        </listener-class>
    </listener>
```

配置<context-param>是初始化上下文
配置<listener>来加载配置文件

ContextLoaderListener是实现了ServletContextListener接口的监听器

ContextLoaderListener继承了ContextLoader

启动项目后，将会触发contextInitialized()方法初始化上下文



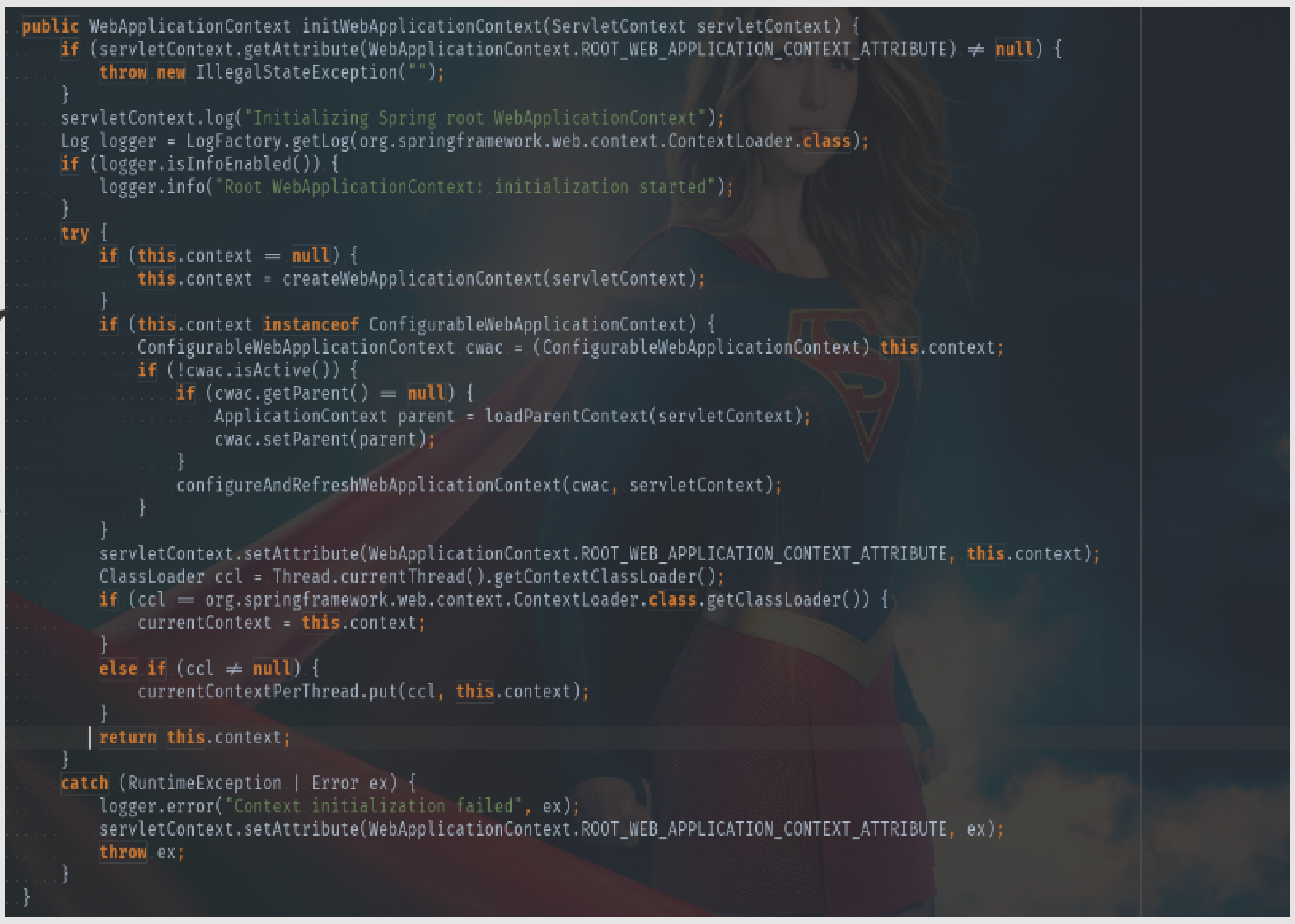
```
package org.springframework.web.context;

public class ContextLoaderListener extends ContextLoader implements ServletContextListener {

    public ContextLoaderListener(WebApplicationContext context) {
        super(context);
    }
    /**
     * Initialize the root web application context.
     */
    @Override
    public void contextInitialized(ServletContextEvent event) {
        initWebApplicationContext(event.getServletContext());
    }
    /**
     * Close the root web application context.
     */
    @Override
    public void contextDestroyed(ServletContextEvent event) {
        closeWebApplicationContext(event.getServletContext());
        ContextCleanupListener.cleanupAttributes(event.getServletContext());
    }
}
```

contextInitialized()方法调用了父类ContextLoader的
initWebApplicationContext(event.getServletContext())方法

这是对ApplicationContext的初始化方法，
就正是进入了springIOC的初始化。

看看initWebApplicationContext做了什么工作？

1：创建WebApplicationContext
2：加载对应的spring配置文件中的Bean
3：将WebApplicationContext放入ServletContext（Java Web的全局变量）



```
public WebApplicationContext initWebApplicationContext(ServletContext servletContext) {
    if (servletContext.getAttribute(WebApplicationContext.ROOT_WEB_APPLICATION_CONTEXT_ATTRIBUTE) ≠ null) {
        throw new IllegalStateException("");
    }
    servletContext.log("Initializing Spring root WebApplicationContext");
    Log logger = LogFactory.getLog(org.springframework.web.context.ContextLoader.class);
    if (logger.isInfoEnabled()) {
        logger.info("Root WebApplicationContext: initialization started");
    }
    try {
        if (this.context == null) {
            this.context = createWebApplicationContext(servletContext);
        }
        if (this.context instanceof ConfigurableWebApplicationContext) {
            ConfigurableWebApplicationContext cwac = (ConfigurableWebApplicationContext) this.context;
            if (!cwac.isActive()) {
                if (cwac.getParent() == null) {
                    ApplicationContext parent = loadParentContext(servletContext);
                    cwac.setParent(parent);
                }
                configureAndRefreshWebApplicationContext(cwac, servletContext);
            }
        }
        servletContext.setAttribute(WebApplicationContext.ROOT_WEB_APPLICATION_CONTEXT_ATTRIBUTE, this.context);
        ClassLoader ccl = Thread.currentThread().getContextClassLoader();
        if (ccl == org.springframework.web.context.ContextLoader.class.getClassLoader()) {
            currentContext = this.context;
        }
        else if (ccl ≠ null) {
            currentContextPerThread.put(ccl, this.context);
        }
        return this.context;
    }
    catch (RuntimeException | Error ex) {
        logger.error("Context initialization failed", ex);
        servletContext.setAttribute(WebApplicationContext.ROOT_WEB_APPLICATION_CONTEXT_ATTRIBUTE, ex);
        throw ex;
    }
}
```

createWebApplicationContext(servletContext)方法
即是完成创建WebApplicationContext工作

configureAndRefreshWebApplicationContext就是用来加载spring配置文件中的Bean实例，封装ApplicationContext数据并且初始化所有相关Bean对象

configureAndRefreshWebApplicationContext会从web.xml中读取名为contextConfigLocation的配置
也就是spring xml数据源设置，然后放到ApplicationContext中

用传说中的refresh方法执行所有Java对象的创建

最后完成ApplicationContext创建之后就是将其放入ServletContext中，注意它存储的key值常量

servletContext.setAttribute(WebApplicationContext.ROOT_WEB_APPLICATION_CONTEXT_ATTRIBUTE, this.context);