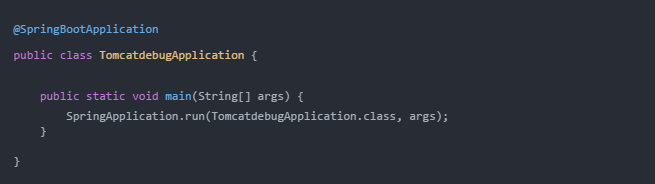
<https://mp.weixin.qq.com/s/arJ_HVTGEYEJCM1idboOmw>

## 1.前言

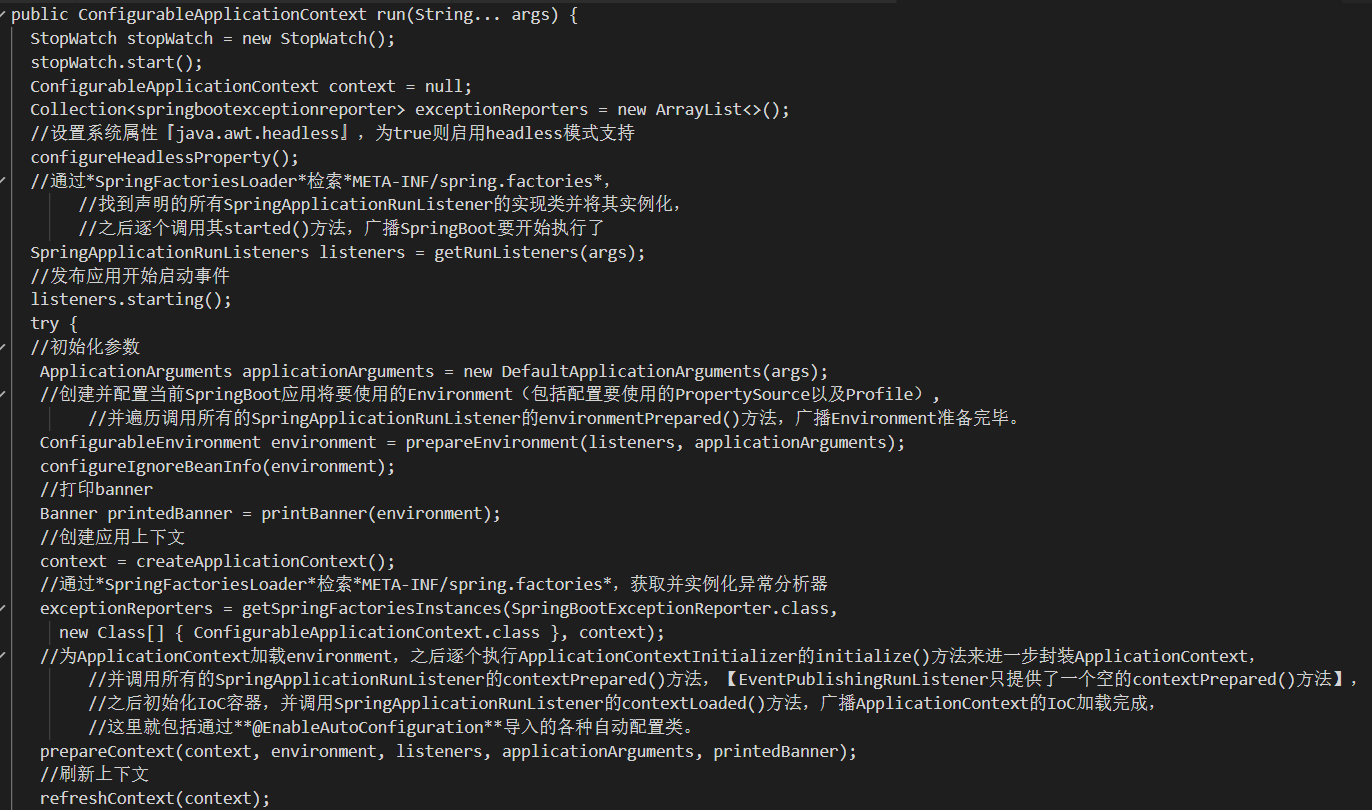
[我们知道 SpringBoot 给我们带来了一个全新的开发体验，我们可以直接把 web 程序达成 jar 包，直接启动，这就得益于 SpringBoot 内置了容器，可以直接启动，本文将以 Tomcat 为例，来看看 SpringBoot 是如何启动 Tomcat 的，同时也将展开学习下 Tomcat 的源码，了解 Tomcat 的设计。](https://mp.weixin.qq.com/s?__biz=MzI1NDY0MTkzNQ==&mid=2247488565&idx=1&sn=8b46ed47e6379ba9d212dc5b2b7b0b01&scene=21#wechat_redirect)

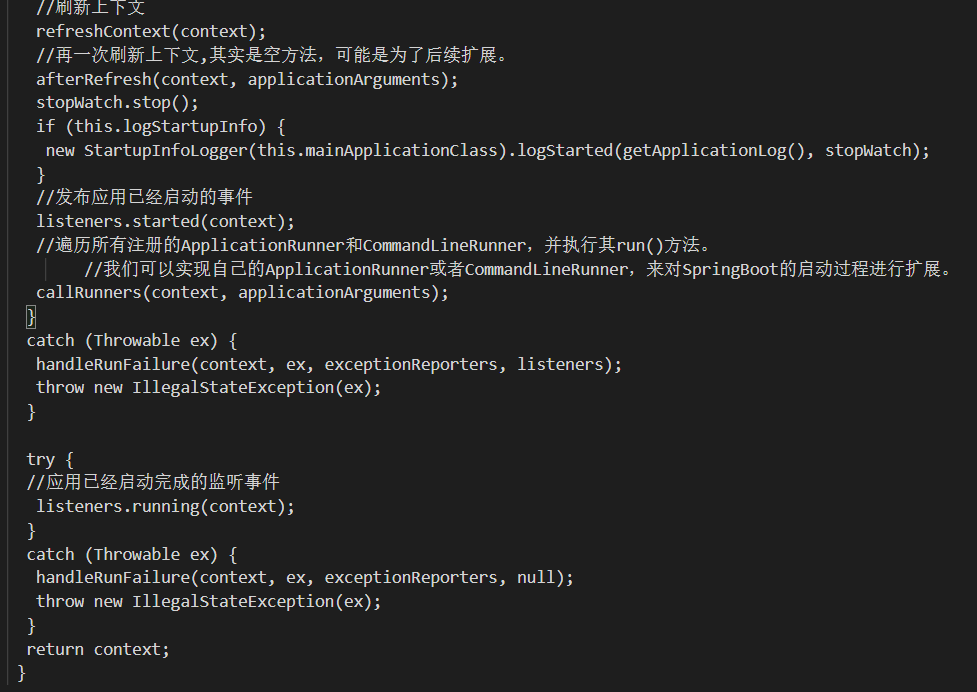
## 2.[从 Main 方法说起](https://mp.weixin.qq.com/s?__biz=MzI1NDY0MTkzNQ==&mid=2247488565&idx=1&sn=8b46ed47e6379ba9d212dc5b2b7b0b01&scene=21#wechat_redirect)

[用过SpringBoot的人都知道，首先要写一个main方法来启动：](https://mp.weixin.qq.com/s?__biz=MzI1NDY0MTkzNQ==&mid=2247488565&idx=1&sn=8b46ed47e6379ba9d212dc5b2b7b0b01&scene=21#wechat_redirect)



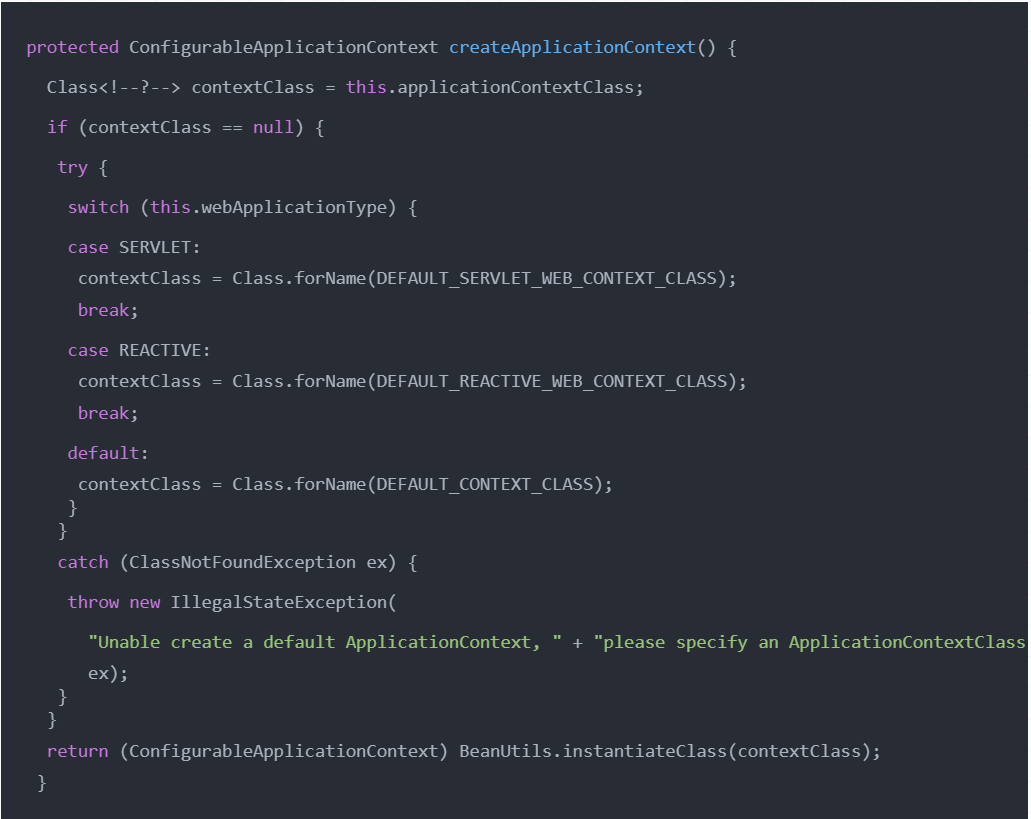
[我们直接点击 run 方法的源码，跟踪下来，发下最终的 run 方法是调用 ConfigurableApplicationContext 方法，源码如下：](https://mp.weixin.qq.com/s?__biz=MzI1NDY0MTkzNQ==&mid=2247488565&idx=1&sn=8b46ed47e6379ba9d212dc5b2b7b0b01&scene=21#wechat_redirect)



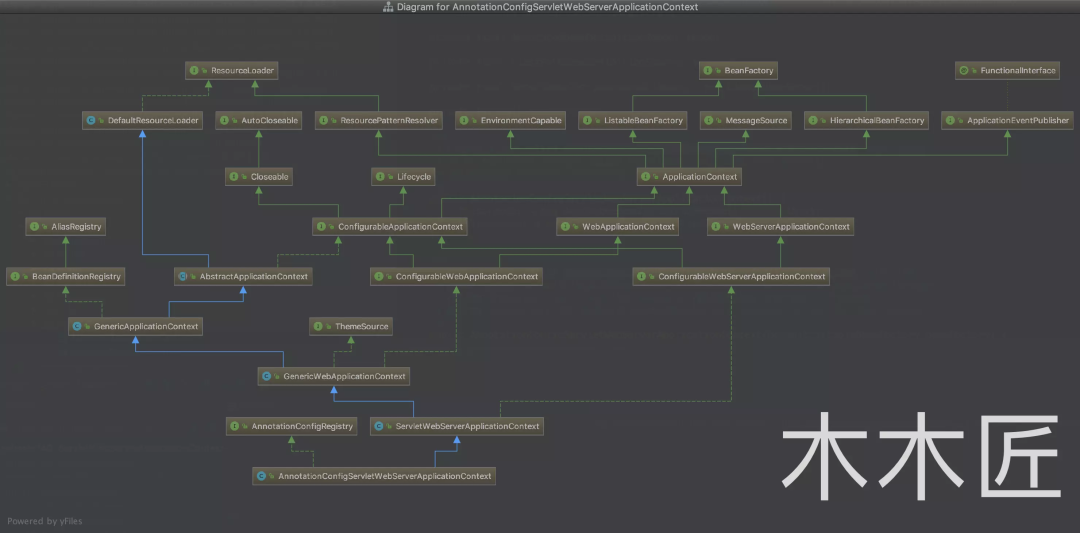


其实这个方法我们可以简单的总结下步骤为 > 1. 配置属性 > 2. 获取监听器，发布应用开始启动事件 > 3. 初始化输入参数 > 4. 配置环境，输出banner > 5. 创建上下文 > 6. 预处理上下文 > 7. 刷新上下文 > 8. 再刷新上下文 > 9. 发布应用已经启动事件 > 10. 发布应用启动完成事件

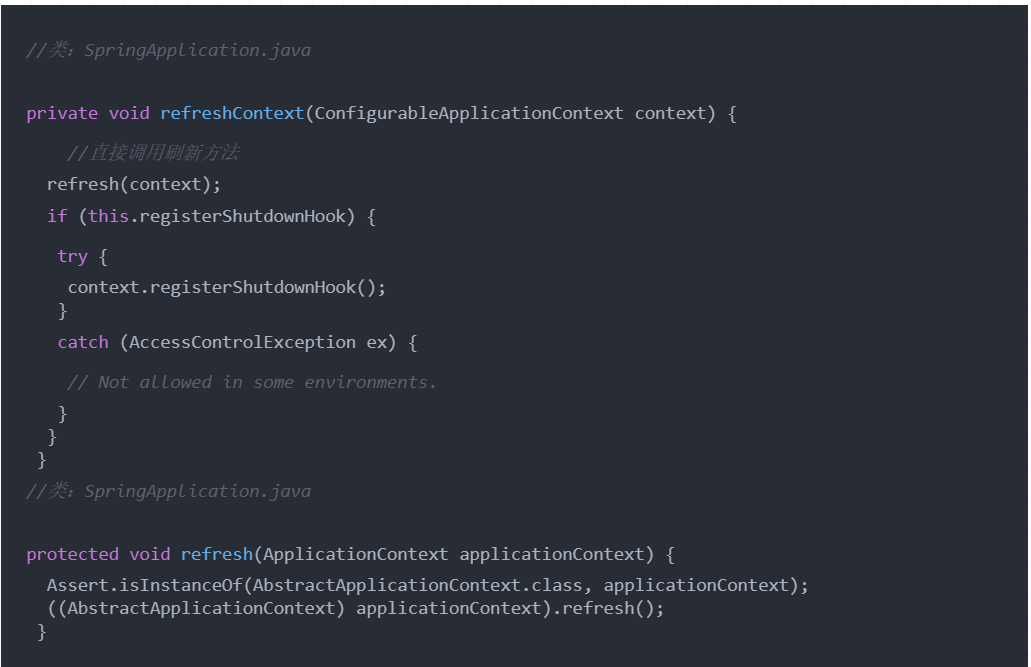
其实上面这段代码，如果只要分析 tomcat 内容的话，只需要关注两个内容即可，上下文是如何创建的，上下文是如何刷新的，分别对应的方法就是 createApplicationContext() 和 refreshContext(context)，接下来我们来看看这两个方法做了什么。



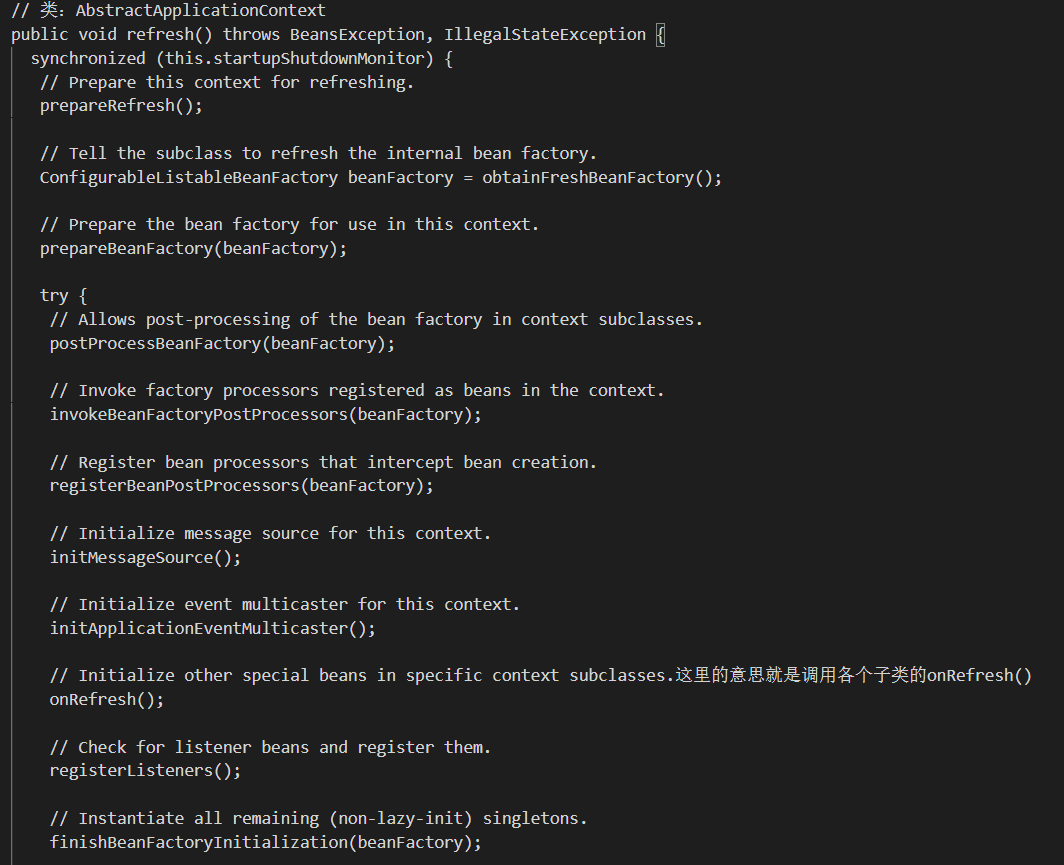
这里就是根据我们的 webApplicationType 来判断创建哪种类型的 Servlet,代码中分别对应着 Web 类型(SERVLET),响应式 Web 类型（REACTIVE),非 Web 类型（default),我们建立的是 Web 类型，所以肯定实例化 DEFAULT\_SERVLET\_WEB\_CONTEXT\_CLASS 指定的类，也就是AnnotationConfigServletWebServerApplicationContext 类，我们来用图来说明下这个类的关系:

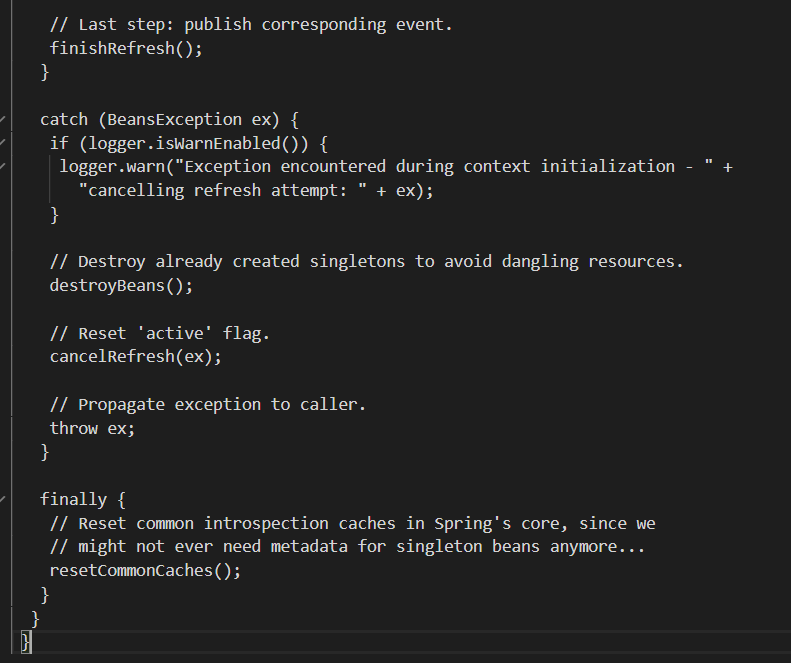


通过这个类图我们可以知道，这个类继承的是ServletWebServerApplicationContext,这就是我们真正的主角，而这个类最终是继承了AbstractApplicationContext，了解完创建上下文的情况后，我们再来看看刷新上下文，相关代码如下：



这里还是直接传递调用本类的 refresh(context) 方法，最后是强转成父类 AbstractApplicationContext 调用其 refresh() 方法,该代码如下：

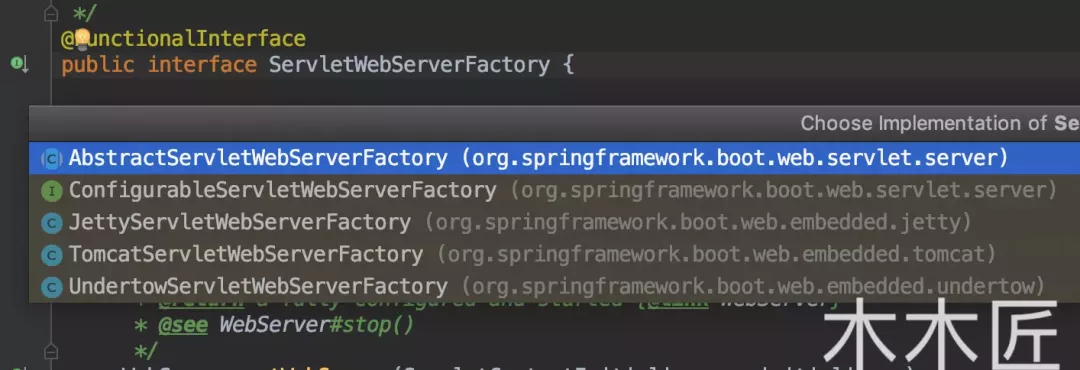




这里我们看到 onRefresh() 方法是调用其子类的实现，根据我们上文的分析，我们这里的子类是 ServletWebServerApplicationContext。



到这里，其实庐山真面目已经出来了，createWebServer() 就是启动 web 服务，但是还没有真正启动 Tomcat，既然 webServer 是通过 ServletWebServerFactory 来获取的，我们就来看看这个工厂的真面目。



## 3.走进 Tomcat 内部

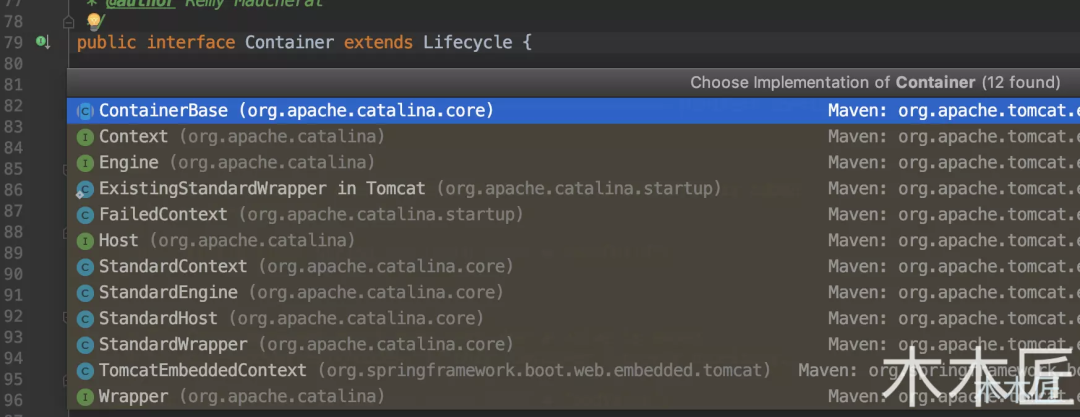
根据上图我们发现，工厂类是一个接口，各个具体服务的实现是由各个子类来实现的，所以我们就去看看 TomcatServletWebServerFactory.getWebServer() 的实现。



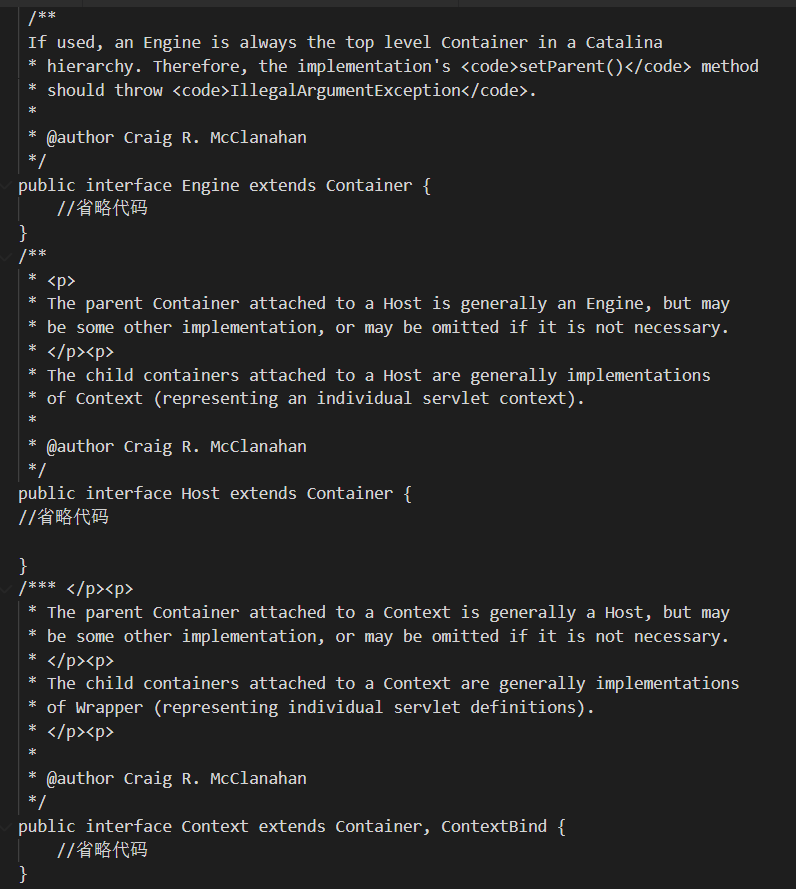
根据上面的代码，我们发现其主要做了两件事情，第一件事就是把 Connnctor(我们称之为连接器)对象添加到 Tomcat 中，第二件事就是 configureEngine,这连接器我们勉强能理解（不理解后面会述说），那这个 Engine 是什么呢？我们查看 tomcat.getEngine() 的源码：

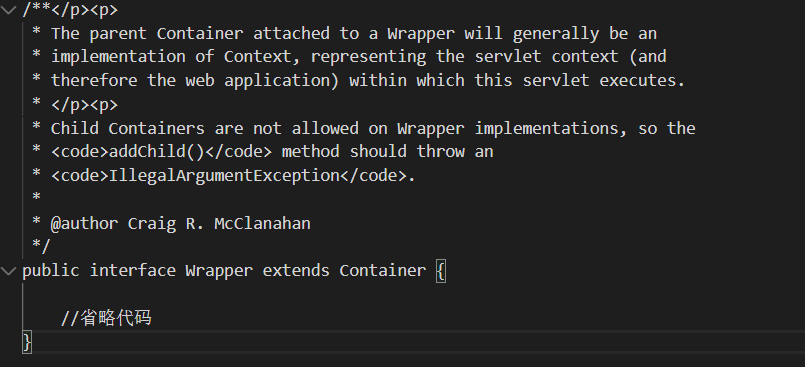


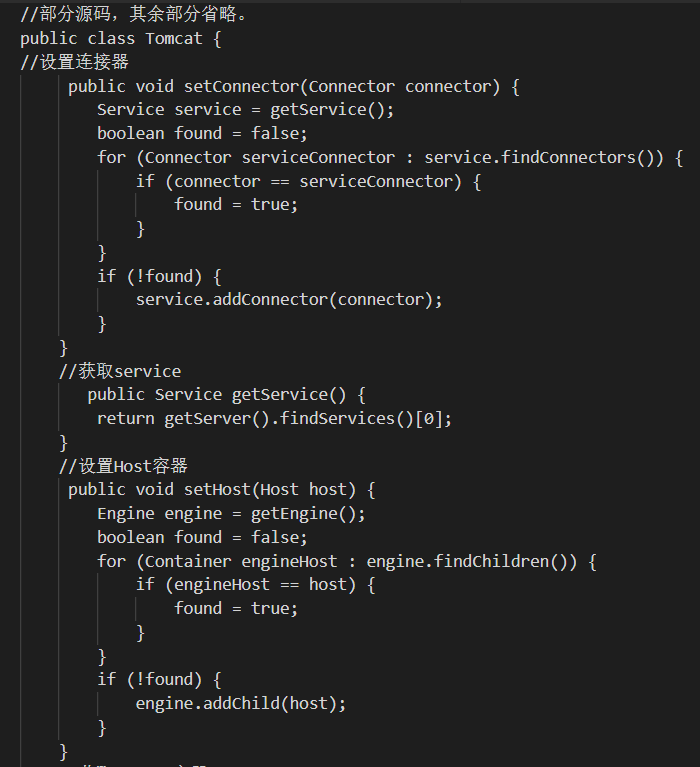
根据上面的源码，我们发现，原来这个Engine是容器，我们继续跟踪源码，找到Container接口

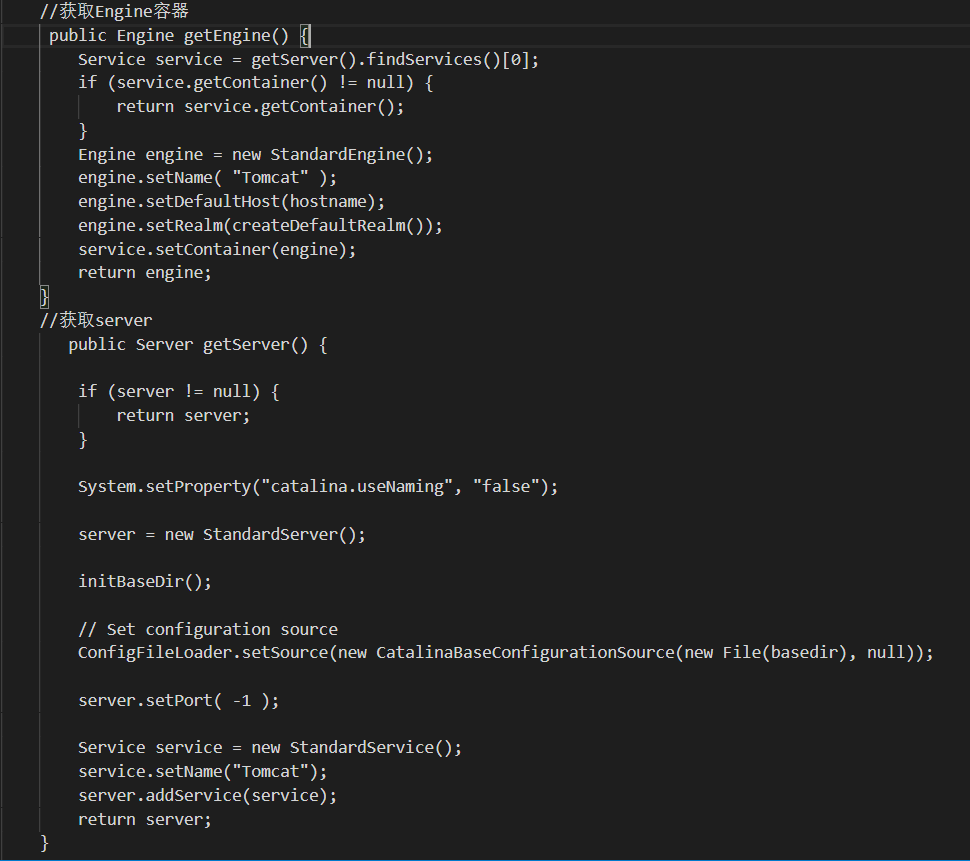


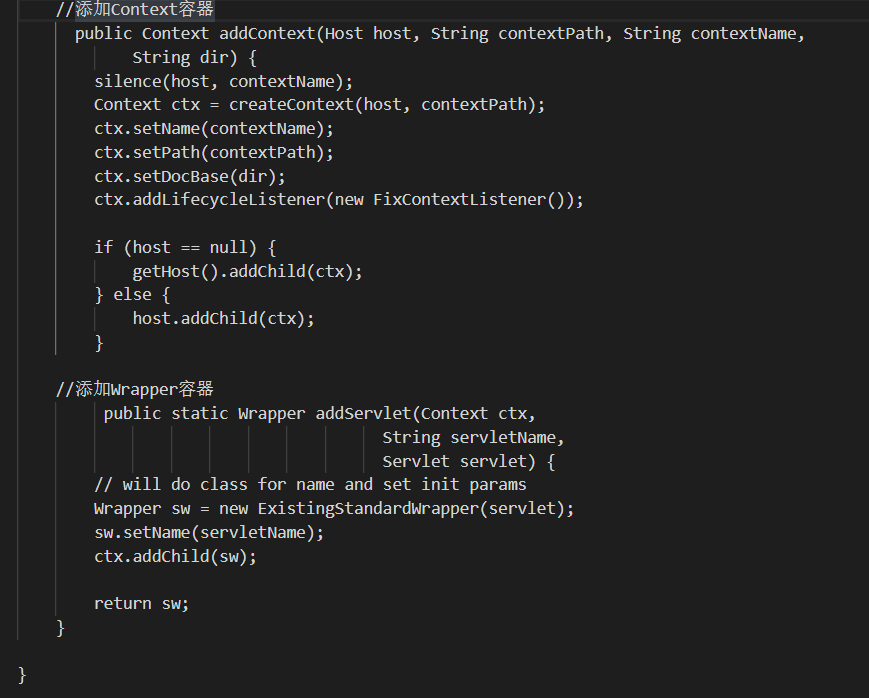
上图中，我们看到了 4 个子接口，分别是 Engine,Host,Context,Wrapper。我们从继承关系上可以知道他们都是容器，那么他们到底有啥区别呢？我看看他们的注释是怎么说的。









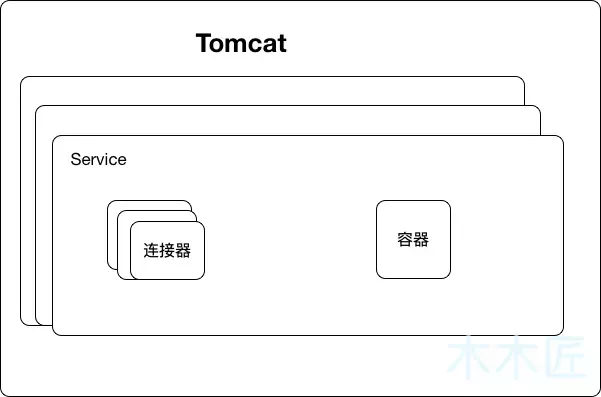


上面的注释翻译过来就是，Engine 是最高级别的容器，其子容器是 Host,Host 的子容器是 Context,Wrapper 是 Context 的子容器，所以这 4 个容器的关系就是父子关系，也就是 Engine>Host>Context>Wrapper。我们再看看 Tomcat 类的源码:

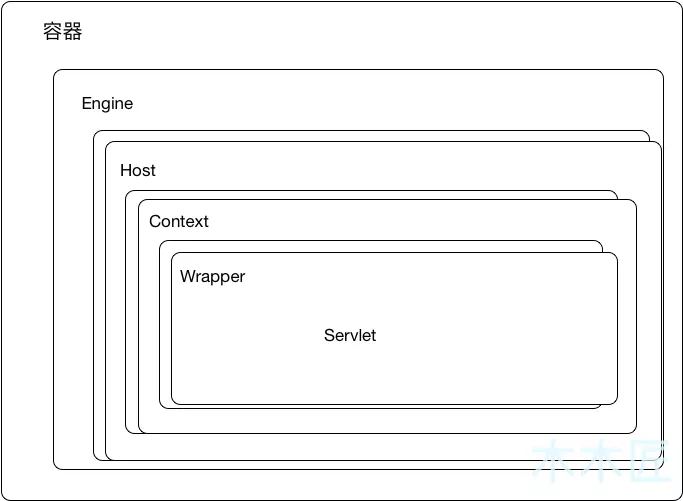
阅读 Tomcat 的 getServer() 我们可以知道，Tomcat 的最顶层是 Server,Server 就是 Tomcat 的实例，一个 Tomcat 一个 Server;通过 getEngine() 我们可以了解到 Server 下面是 Service，而且是多个，一个 Service 代表我们部署的一个应用，而且我们还可以知道，Engine 容器，一个 service 只有一个；根据父子关系，我们看 setHost() 源码可以知道，host 容器有多个；同理，我们发现 addContext() 源码下，Context 也是多个；addServlet() 表明 Wrapper 容器也是多个，而且这段代码也暗示了，其实 Wrapper 和 Servlet 是一层意思。

另外我们根据 setConnector 源码可以知道，连接器 (Connector) 是设置在 service 下的，而且是可以设置多个连接器 (Connector)。

根据上面分析，我们可以小结下：Tomcat 主要包含了 2 个核心组件，连接器(Connector)和容器(Container),用图表示如下：



一个 Tomcat 是一个 Server,一个 Server 下有多个 service，也就是我们部署的多个应用，一个应用下有多个连接器(Connector)和一个容器（Container）,容器下有多个子容器，关系用图表示如下：



Engine 下有多个 Host 子容器，Host 下有多个 Context 子容器，Context 下有多个 Wrapper 子容器。

## 4.总结

SpringBoot 的启动是通过 new SpringApplication() 实例来启动的，启动过程主要做如下几件事情：> 1. 配置属性 > 2. 获取监听器，发布应用开始启动事件 > 3. 初始化输入参数 > 4. 配置环境，输出banner > 5. 创建上下文 > 6. 预处理上下文 > 7. 刷新上下文 > 8. 再刷新上下文 > 9. 发布应用已经启动事件 > 10. 发布应用启动完成事件

而启动 Tomcat 就是在第7步中“刷新上下文”；Tomcat 的启动主要是初始化 2 个核心组件，连接器(Connector)和容器（Container），一个 Tomcat 实例就是一个 Server，一个 Server 包含多个 Service，也就是多个应用程序，每个 Service 包含多个连接器（Connetor）和一个容器（Container),而容器下又有多个子容器，按照父子关系分别为：Engine,Host,Context,Wrapper，其中除了 Engine 外，其余的容器都是可以有多个。