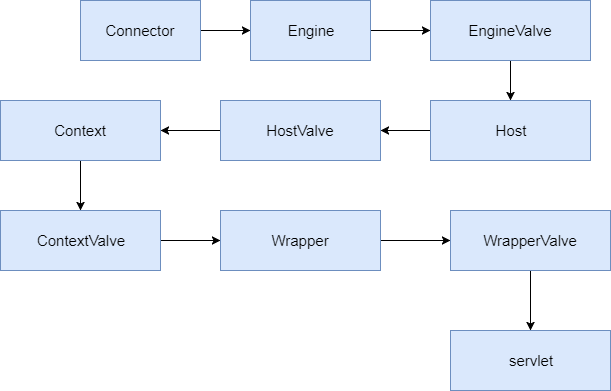


Tomcat 的心脏是两个组件：Connector 和 Container，Connector 主要负责接收请求，Container 主要对请求进行处理，对应Jetty的Handler。Service 只是在 Connector 和 Container 外面多包一层，把它们组装在一起，向外面提供服务。Server对Service进行管理，会提供一个让其它程序能够访问到这个 Service的接口。

Connector 组件是 Tomcat 中两个核心组件之一，它的主要任务是负责接收浏览器的发过来的连接请求，创建一个Request和 Response对象，然后将请求传给Container。

Container 是包含四个子容器，分别是Engine、Host、Context、Wrapper。Engine 容器比较简单，它只定义了一些基本的关联关系。Host保存了主机相关信息。Context 用来管理里面的 Servlet 实例。Wrapper 直接与Servlet打交道，它负责Servlet的装载、初始化、执行以及资源回收。

另外，还有一个Valve实现来具体业务逻辑，每个容器都有一个valve，主要实现了对其子对象的Valve的调用。



Jetty 的架构从前面的分析可知，它的所有组件都是基于 Handler 来实现，当然它也支持 JMX。但是主要的功能扩展都可以用 Handler 来实现。可以说 Jetty 是面向 Handler 的架构，就像 Spring 是面向 Bean 的架构，iBATIS 是面向 statement 一样，而 Tomcat 是以多级容器构建起来的，它们的架构设计必然都有一个“元神”，所有以这个“元神“构建的其它组件都是肉身。

从设计模板角度来看 Handler 的设计实际上就是一个责任链模式，接口类 HandlerCollection 可以帮助开发者构建一个链，而另一个接口类 ScopeHandler 可以帮助你控制这个链的访问顺序。另外一个用到的设计模板就是观察者模式，用这个设计模式控制了整个 Jetty 的生命周期，只要继承了 LifeCycle 接口，你的对象就可以交给 Jetty 来统一管理了。所以扩展 Jetty 非常简单，也很容易让人理解，整体架构上的简单也带来了无比的好处，Jetty 可以很容易被扩展和裁剪。

相比之下，Tomcat 要臃肿很多，Tomcat 的整体设计上很复杂，前面说了 Tomcat 的核心是它的容器的设计，从 Server 到 Service 再到 engine 等 container 容器。作为一个应用服务器这样设计无口厚非，容器的分层设计也是为了更好的扩展，这是这种扩展的方式是将应用服务器的内部结构暴露给外部使用者，使得如果想扩展 Tomcat，开发人员必须要首先了解 Tomcat 的整体设计结构，然后才能知道如何按照它的规范来做扩展。这样无形就增加了对 Tomcat 的学习成本。不仅仅是容器，实际上 Tomcat 也有基于责任链的设计方式，像串联 Pipeline 的 Vavle 设计也是与 Jetty 的 Handler 类似的方式。要自己实现一个 Vavle 与写一个 Handler 的难度不相上下。表面上看，Tomcat 的功能要比 Jetty 强大，因为 Tomcat 已经帮你做了很多工作了，而 Jetty 只告诉，你能怎么做，如何做，有你去实现。

单纯比较 Tomcat 与 Jetty 的性能意义不是很大，只能说在某种使用场景下，它表现的各有差异。因为它们面向的使用场景不尽相同。从架构上来看 Tomcat 在处理少数非常繁忙的连接上更有优势，也就是说连接的生命周期如果短的话，Tomcat 的总体性能更高。

而 Jetty 刚好相反，Jetty 可以同时处理大量连接而且可以长时间保持这些连接。例如像一些 web 聊天应用非常适合用 Jetty 做服务器，像淘宝的 web 旺旺就是用 Jetty 作为 Servlet 引擎。

另外由于 Jetty 的架构非常简单，作为服务器它可以按需加载组件，这样不需要的组件可以去掉，这样无形可以减少服务器本身的内存开销，处理一次请求也是可以减少产生的临时对象，这样性能也会提高。另外 Jetty 默认使用的是 NIO 技术在处理 I/O 请求上更占优势，Tomcat 默认使用的是 BIO，在处理静态资源时，Tomcat 的性能不如 Jetty。

作为一个标准的 Servlet 引擎，它们都支持标准的 Servlet 规范，还有 Java EE 的规范也都支持，由于 Tomcat 的使用的更加广泛，它对这些支持的更加全面一些，有很多特性 Tomcat 都直接集成进来了。但是 Jetty 的应变更加快速，这一方面是因为 Jetty 的开发社区更加活跃，另一方面也是因为 Jetty 的修改更加简单，它只要把相应的组件替换就好了，而 Tomcat 的整体结构上要复杂很多，修改功能比较缓慢。所以 Tomcat 对最新的 Servlet 规范的支持总是要比人们预期的要晚。