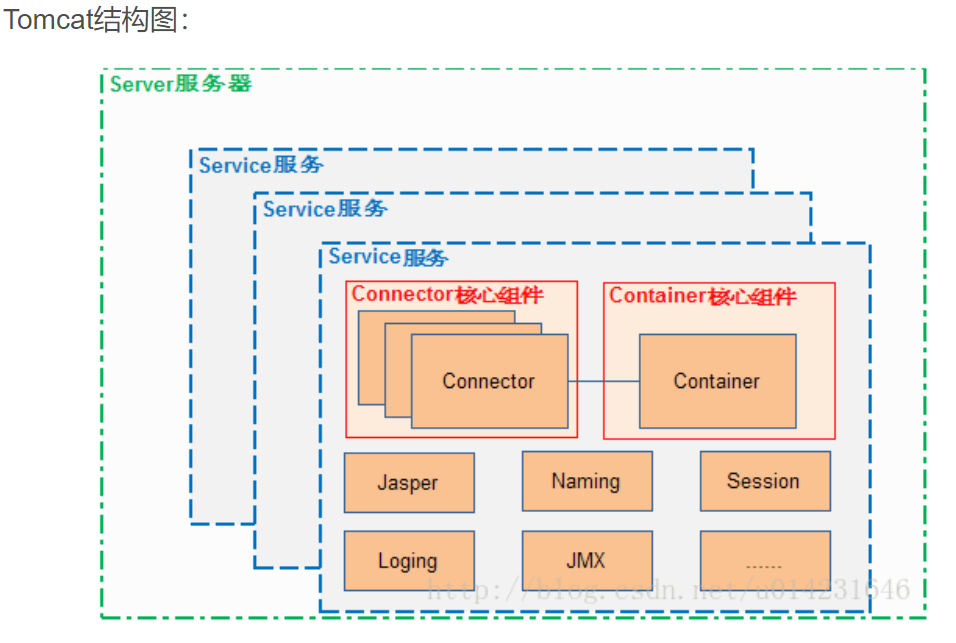
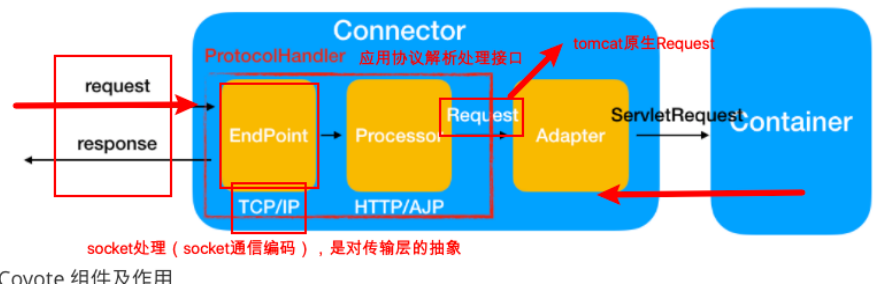
## Tomcat的结构图



Tomcat主要组件：服务器Server，服务Service，连接器Connector、容器Container。连接器Connector和容器Container是Tomcat的核心。

1. 整个Tomcat就是一个Catalina实例，Tomcat启动的时候会初始化这个实例，Catalina实例通过加载server.xml完成其他实例的创建，创建并管理一个Server。
2. Server创建并管理多个服务Service。
3. 每个服务Service可以有多个Connector和一个Container。

**Connector**

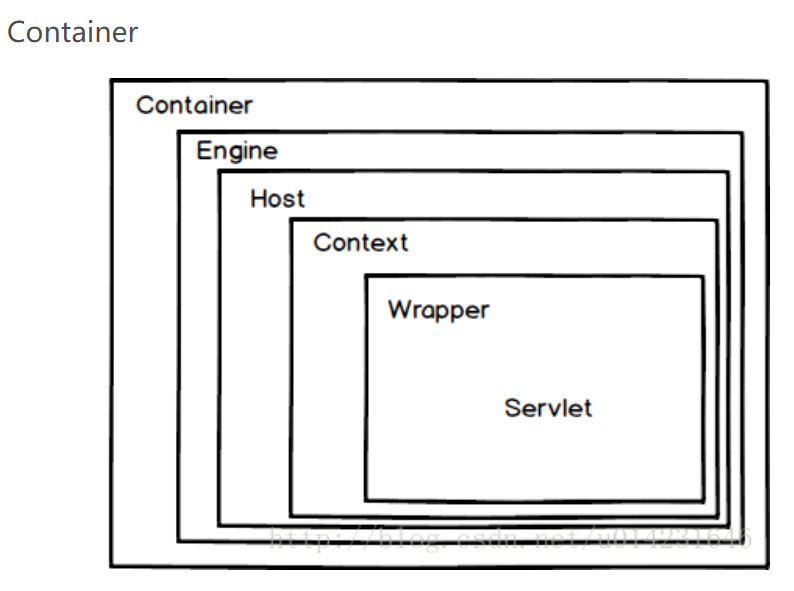


一个Connecter将在某个指定的端口上侦听客户请求，接收浏览器的发过来的 tcp 连接请求，创建一个 Request 和 Response 对象分别用于和请求端交换数据，然后会产生一个线程来处理这个请求并把产生的 Request 和 Response 对象传给处理Engine(Container中的一部分)，从Engine出获得响应并返回客户。

Tomcat中有两个经典的Connector，一个直接侦听来自Browser的HTTP请求，另外一个来自其他的WebServer请求。HTTP/1.1 Connector在端口8080处侦听来自客户Browser的HTTP请求，AJP/1.3 Connector在端口8009处侦听其他Web Server（其他的HTTP服务器）的Servlet/JSP请求。

Connector 最重要的功能就是接收连接请求然后分配线程让 Container 来处理这个请求，所以这必然是多线程的，多线程的处理是 Connector 设计的核心。

**Container组件**



Container是容器的父接口，该容器的设计用的是典型的责任链的设计模式，它由四个自容器组件构成，分别是Engine、Host、Context、Wrapper。这四个组件是负责关系，存在包含关系。通常一个Servlet class对应一个Wrapper，如果有多个Servlet定义多个Wrapper，如果有多个Wrapper就要定义一个更高的Container，如Context。   
Context 还可以定义在父容器 Host 中，Host 不是必须的，但是要运行 war 程序，就必须要 Host，因为 war 中必有 web.xml 文件，这个文件的解析就需要 Host 了，如果要有多个 Host 就要定义一个 top 容器 Engine 了。而 Engine 没有父容器了，一个 Engine 代表一个完整的 Servlet 引擎。

* Engine 容器

包含多个Host  
 表示整个Catalina的Servlet引擎，用来管理多个虚拟站点，一个Service最多只能有一个Engine，但 是一个引擎可包含多个Host。

* Host 容器

包含多个Context（应用）  
 Host 是 Engine 的子容器，一个 Host 在 Engine 中代表一个虚拟主机，这个虚拟主机的作用就是运行多个应用，它负责安装和展开这些应用，并且标识这个应用以便能够区分它们。它的子容器通常是 Context，它除了关联子容器外，还有就是保存一个主机应该有的信息。

* Context 容器

包含多个Wrapper  
 Context 代表 Servlet 的 Context，它具备了 Servlet 运行的基本环境，理论上只要有 Context 就能运行 Servlet 了。简单的 Tomcat 可以没有 Engine 和 Host。Context 最重要的功能就是管理它里面的 Servlet 实例，Servlet 实例在 Context 中是以 Wrapper 出现的，还有一点就是 Context 如何才能找到正确的 Servlet 来执行它呢？ Tomcat5 以前是通过一个 Mapper 类来管理的，Tomcat5 以后这个功能被移到了 request 中，在前面的时序图中就可以发现获取子容器都是通过 request 来分配的。

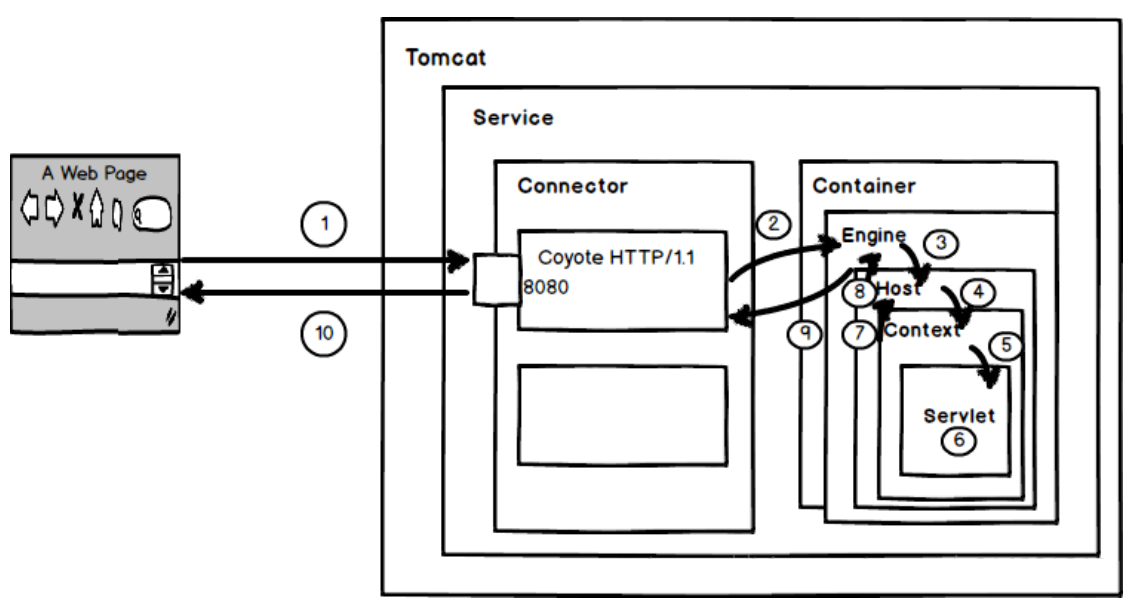
* Wrapper 容器

也就是Servlet  
 Wrapper 代表一个 Servlet，它负责管理一个 Servlet，包括的 Servlet 的装载、初始化、执行以及资源回收。Wrapper 是最底层的容器，它没有子容器了，所以调用它的 addChild 将会报错。Wrapper 的实现类是 StandardWrapper，StandardWrapper 还实现了拥有一个 Servlet 初始化信息的 ServletConfig，由此看出 StandardWrapper 将直接和 Servlet 的各种信息打交道。

**其他组件**

Tomcat 还有其它重要的组件，如安全组件 security、logger 日志组件、session、mbeans、naming 等其它组件。这些组件共同为 Connector 和 Container 提供必要的服务。

## Tomcat处理请求的过程



Tomcat Server处理一个HTTP请求的过程

1、用户点击网页内容，请求被浏览器Browser发送到对应的服务器端口8080，被在那里监听的Coyote HTTP/1.1 Connector获得。

2、Connector将通过TCP/IP协议传输过来的socket数据封装成Request对象，并适配成HttpRequest对象，把该请求信息交给它所在的Service的Engine来处理，并等待Engine的回应。

3、Engine获得请求localhost/test/index.jsp，匹配所有的虚拟主机Host。

4、Engine匹配到名为localhost的Host（即使匹配不到也把请求交给该Host处理，因为该Host被定义为该Engine的默认主机），名为localhost的Host获得请求/test/index.jsp，匹配它所拥有的所有的Context。Host匹配到路径为/test的Context（如果匹配不到就把该请求交给路径名为“ ”的Context去处理）。

5、path=“/test”的Context获得请求/index.jsp，在它的mapping table中寻找出对应的Servlet。Context匹配到URL PATTERN为\*.jsp的Servlet,对应于JspServlet类。

6、构造HttpServletRequest对象和HttpServletResponse对象，作为参数调用JspServlet的doGet（）或doPost（）.执行业务逻辑、数据存储等程序。

7、Context把执行完之后的HttpServletResponse对象返回给Host。

8、Host把HttpServletResponse对象返回给Engine。

9、Engine把HttpServletResponse对象返回Connector。

10、Connector把HttpServletResponse对象适配成ServletReponse，并转化换成TCP/IP协议的socket响应给客户端Browser。