**一.Tomcat是什么？**

Tomcat是一个Web应用服务器，同时也是一个Servlet/JSP容器。**Tomcat作为Servlet容器，负责处理客户端请求，把请求传送给Servlet，并将Servlet的响应返回给客户端。**

**二.Tomcat的体系结构**

Tomcat是一个基于组件的服务器，它的构成组件都是可配置的。其各个组件都在Tomcat安装目录下的../conf/server.xml文件中配置。

server.xml文件源代码如下：

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>

<!--顶层类元素，可以包含多个Service-->

<Server port="8005" shutdown="SHUTDOWN">

<Listener className="org.apache.catalina.startup.VersionLoggerListener" />

<Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />

<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />

<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />

<Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

<GlobalNamingResources>

<Resource name="UserDatabase" auth="Container"

type="org.apache.catalina.UserDatabase"

description="User database that can be updated and saved"

factory="org.apache.catalina.users.MemoryUserDatabaseFactory"

pathname="conf/tomcat-users.xml" />

</GlobalNamingResources>

<!--顶层类元素，可包含一个Engine（container），多个connector-->

<Service name="Catalina">

<!--连接器类元素，代表通信接口-->

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

<!--容器类元素，为特定的service组件处理客户请求-->

<Engine name="Catalina" defaultHost="localhost">

<Realm className="org.apache.catalina.realm.UserDatabaseRealm"

resourceName="UserDatabase"/>

</Realm>

<!--容器类元素，为特定的虚拟主机组件处理客户请求-->

<Host name="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="true">

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="localhost\_access\_log" suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>

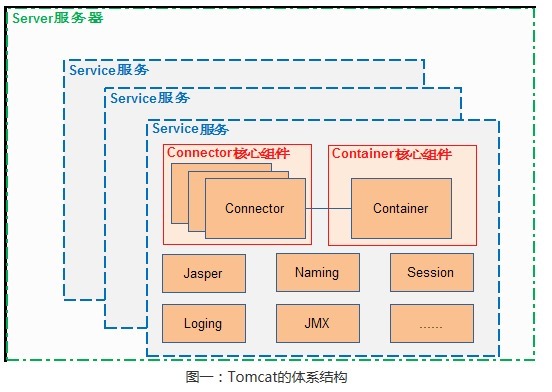
</Engine>

</Service>

</Server>

[复制代码](javascript:void(0);)

由上面的源码中可得出Tomcat的体系结构如下：

[](http://images2017.cnblogs.com/blog/1131840/201712/1131840-20171215142449996-360033997.jpg)

由上图可以看出Tomcat的心脏是两个核心组件**：Connector和Container**。其中一个Container可以选择多个Connector。下面就对这两个组件进行详细讲解。

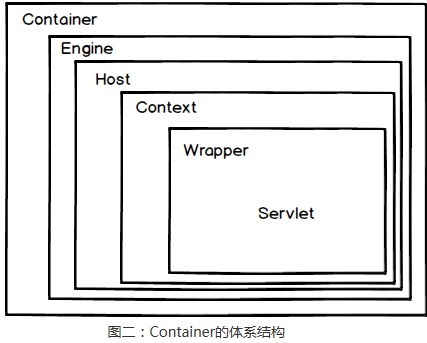
**三.Tomcat的两个核心组件：Connector和Container**

**1.Connector组件**

一个Connector组件将在某个指定的端口上侦听客户请求，接收浏览器发过来的tcp连接请求，创建一个Request和一个Response对象分别用于和其你去端交换数据，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Engine，从Engine中获得响应并返回给客户端。 Tomcat有两个经典的Connector，一个直接侦听来自浏览器的HTTP请求，另外一个侦听来自其他的WebServer的请求。Cotote HTTP/1.1 Connector在端口8080处侦听来自客户浏览器的HTTP请求，Coyote JK2 Connector在端口8009处侦听其他WebServer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。

**2.Container组件**

Container组件的体系结构如下：

[](http://images2017.cnblogs.com/blog/1131840/201712/1131840-20171215142450871-601807957.jpg)

**Container**

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则定义多个Wrapper，如果有多个Wrapper就要定义一个更高的Container，如Context。 Context定义在父容器 Host 中，其中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**

Engine 容器比较简单，它只定义了一些基本的关联关系 Host 容器

**Host**

Host 是 Engine 的字容器，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**

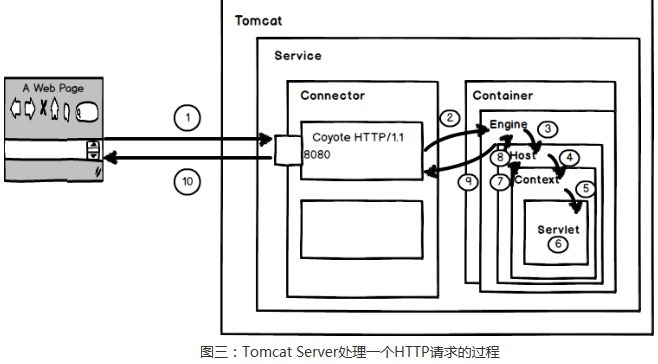
Context 代表 Servlet 的 Context，它具备了 Servlet 运行的基本环境，理论上只要有 Context 就能运行 Servlet 了。简单的 Tomcat 可以没有 Engine 和 Host。Context 最重要的功能就是管理它里面的 Servlet 实例，Servlet 实例在 Context 中是以 Wrapper 出现的，还有一点就是 Context 如何才能找到正确的 Servlet 来执行它呢？ Tomcat5 以前是通过一个 Mapper 类来管理的，Tomcat5 以后这个功能被移到了 request 中，在前面的时序图中就可以发现获取子容器都是通过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

**Wrapper**

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。

**说明：除了上述组件外，Tomcat中还有其他重要的组件，如安全组件security、logger日志组件、session、mbeans、naming等其他组件。这些组件共同为Connector和Container提供必要的服务。**

**四.Tomcat 处理一个HTTP请求的过程**

Tomcat Server处理一个HTTP请求的过程如下图所示： [](http://images2017.cnblogs.com/blog/1131840/201712/1131840-20171215142451699-1297214931.jpg)

1.用户在浏览器中输入网址localhost:8080/test/index.jsp，请求被发送到本机端口8080，被在那里监听的Coyote HTTP/1.1 Connector获得；

2.Connector把该请求交给它所在的Service的Engine（Container）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；

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对象返回给客户Browser。

**只要客户端一访问服务器，就开始创建servlet对象，通过调用init（）方法对其初始化，只创建一次，以后客户端访问就直接从内存里拿这个servlet对象，在这个期间会执行service（httpservletrequest req ,httpservletresponse res）方法，然后就可以doget和dopost传参了。最后服务器通过 res这个对象向客户端回送数据（一般先得到OutputStream os=res.getOutputStream()对象，在os.write(...)写数据，回送给客户端）。关于servlet对象的销毁，会执行一个destroy（）方法，一般是在关闭web应用或关闭服务器的时候会调用这个destroy（）方法。**