

Tomcat由Connector（连接器）和Container（容器）两大核心组件构成。

连接器，连接器组件是Coyote，负责对外交流：处理Socket连接，负责网络字节流和Request和Respond对象转化；

容器，又四种组件组成，分别是Engine、Host、Context和Wrapper，也是整个tomcat的核心，负责内部处理：加载和管理Servlet，以及具体的Servlet请求；

组件介绍：

**Catalina**

Catalina是Tomcat的servlet容器，负责解析Tomcat的配置⽂件（server.xml） , 以此来创建服务器Server组件并进⾏管理。

**Server**

Server是最顶级的组件，它代表Tomcat的运行实例，在一个JVM中只会包含一个Server。负责组装并启动Servlet引擎。Server通过实现Lifecycle接口，提供了一种优雅的启动和关闭整个系统的方式。在Server的整个生命周期中，不同阶段会有不同的事情要完成。为了方便扩展，它引入了监听器方式，所以它也包含了Listener组件。另外，为了方便在Tomcat中集成JNDI，引入了GlobalNamingResources组件。

**Service**

Service在设计上Server组件可以包含多个Service组件，每个Service组件都包含了若干用于接收客户端消息的Connector组件和处理请求的Engine组件。不同的Connector组件使用不同的通信协议，如HTTP协议和AJP协议，当然还可以有其他的协议A和协议B。若干Connector组件和一个客户端请求处理组件Engine组成的集合即为Service。此外，Service组件还包含了若干Executor组件，每个Executor都是一个线程池，它可以为Service内所有组件提供线程池执行任务。它将若干个Connector组件绑定到一个 Container。

**Connector（Coyote）**

Coyote（Coyote是Connector框架的名字） 是Tomcat 中连接器的组件名称 , 是对外的接口。客户端通过Coyote与服务器建立连接、发送请求并接收响应 。

1.Coyote 封装了底层的网络通信(Socket 请求及响应处理)

2.Coyote 使Catalina 容器(容器组件)与具体的请求协议及IO操作方式完全解耦

3.Coyote 将Socket 输入转换封装为 Request 对象，进一步封装后交由Catalina 容器进行处理，处理请求完成后, Catalina 通过Coyote 提供的Response 对象将结果写入输出流

4.Coyote 负责的是具体协议(应用层)和IO(传输层)相关内容

**EndPoint**

EndPoint 是 Coyote 通信端点，即通信监听的接口，是具体Socket接收和发送处理器，是对传输层的抽象，因此EndPoint用来实现TCP/IP协议的。

**ProtocolHandler**

Coyote 协议接口， 通过Endpoint 和 Processor ， 实现针对具体协议的处 理能力。Tomcat 按照协议和I/O 提供了6个实现类 : AjpNioProtocol ， AjpAprProtocol， AjpNio2Protocol ， Http11NioProtocol ， Http11Nio2Protocol ，Http11AprProtocol

**Processor**

Processor 是Coyote 协议处理接口 ，如果说EndPoint是用来实现TCP/IP协 议的，那么Processor用来实现HTTP协议，Processor接收来自EndPoint的 Socket，读取字节流解析成Tomcat Request和Response对象，并通过 Adapter将其提交到容器处理，Processor是对应用层协议的抽象

**Adapter**

由于协议不同，客户端发过来的请求信息也不尽相同，Tomcat定义了自己的 Request类来封装这些请求信息。ProtocolHandler接口负责解析请求并生成 Tomcat Request类。但是这个Request对象不是标准的ServletRequest，不 能用Tomcat Request作为参数来调用容器。Tomcat设计者的解决方案是引 入CoyoteAdapter，这是适配器模式的经典运用，连接器调用 CoyoteAdapter的Sevice方法，传入的是Tomcat Request对象， CoyoteAdapter负责将Tomcat Request转成ServletRequest，再调用容器

**Container**

Container组件下有几种具体的组件，分别是Engine、Host、Context和Wrapper。这4种组件(容器)是父子关系。Tomcat通过一种分层的架构，使得Servlet容器具有很好的灵活性。

Engine

Engine表示整个Catalina的Servlet引擎，用来管理多个虚拟站点，每个Service组件只能包含一个Engine容器组件， 但Engine组件可以包含若干Host容器组件。

**Host**

代表一个虚拟主机，或者说一个站点，可以给Tomcat配置多个虚拟主机地址，而一个虚拟主机下可包含多个Context。

**Context**

表示一个Web应用程序， 一个Web应用可包含多个Wrapper。

**Wrapper**

表示一个Servlet，Wrapper 作为容器中的最底层，不能包含子容器

**请求流转的具体过程（主流程）**：

1．当Tomcat 启动后，Connector组件的接收器（Acceptor）将会监听是否有客户端套接字连接并接收Socket。

2.一旦监听到客户端连接，则将连接交由线程池Executor处理，开始执行请求响应任务。

3.Http11NioProcessor组件负责从客户端连接中读取消息报文，然后开始解析HTTP的请求行、请求头部、请求体。将解析后的报文封装成Request对象，方便后面处理时通过Request对象获取HTTP协议的相关值。

4.Mapper组件根据HTTP协议请求行的URL属性值和请求头部的Host属性值匹配由哪个Host容器、哪个Context容器、哪个Wrapper容器处理请求，这个过程其实就是根据请求从Tomcat中找到对应的Servlet。然后将路由的结果封装到Request对象中，方便后面处理时通过Request对象选择容器。

5.CoyoteAdaptor组件负责将Connector组件和Engine容器连接起来，把前面处理过程中生成的请求对Request和响应对象Response传递到Engine容器，调用它的管道。

6.Engine容器的管道开始处理请求，管道里包含若干阀门（Valve），每个阀门负责某些处理逻辑。可以根据自己的需要往这个管道中添加多个阀门，它会负责匹配并调用Host容器的管道。

7.Host容器的管道开始处理请求，它同样也包含若干阀门（Valve），它继续往下调用匹配Context容器的管道。

8.Context容器的管道开始处理请求，它负责调用Wrapper容器的管道。

9．Wrapper容器的管道开始处理请求，它会执行该Wrapper容器对应的Servlet对象的处理方法，对请求进行逻辑处理，并将结果输出到客户端。