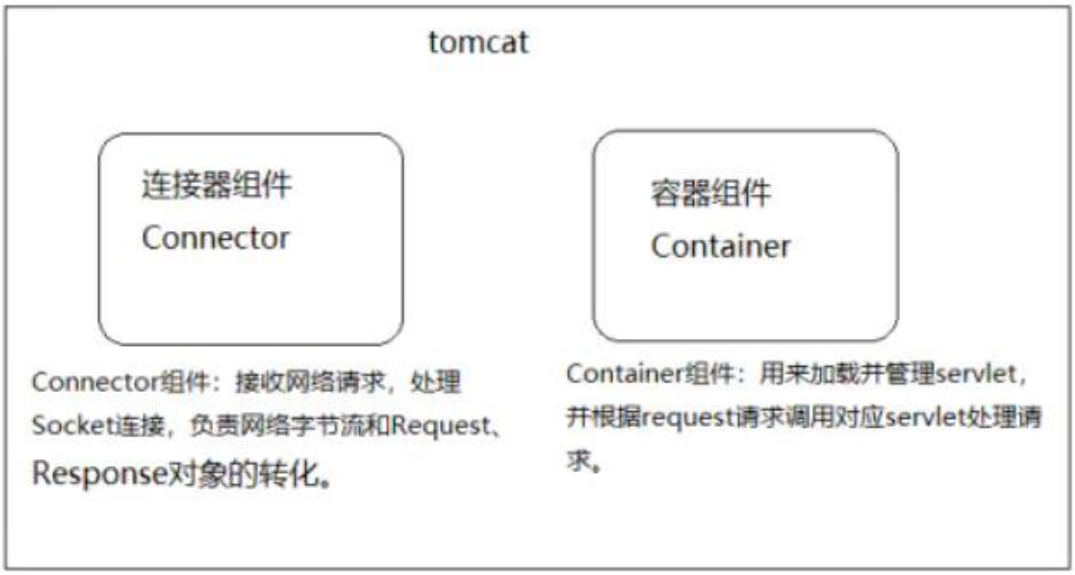
Tomcat 系统总体架构：



Tomcat 设计了两个核心组件连接器（Connector）和容器（Container）来完成 Tomcat 的两大核心 功能。

连接器：负责对外交流： 处理 Socket 连接，负责网络字节流与 Request 和 Response 对象的转化；

容器：负责内部处理：加载和管理 Servlet，以及具体处理 Request 请求；

Tomcat 连接器组件 Coyote：

Coyote 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

求并接受响应 。

（1）Coyote 封装了底层的网络通信（Socket 请求及响应处理）

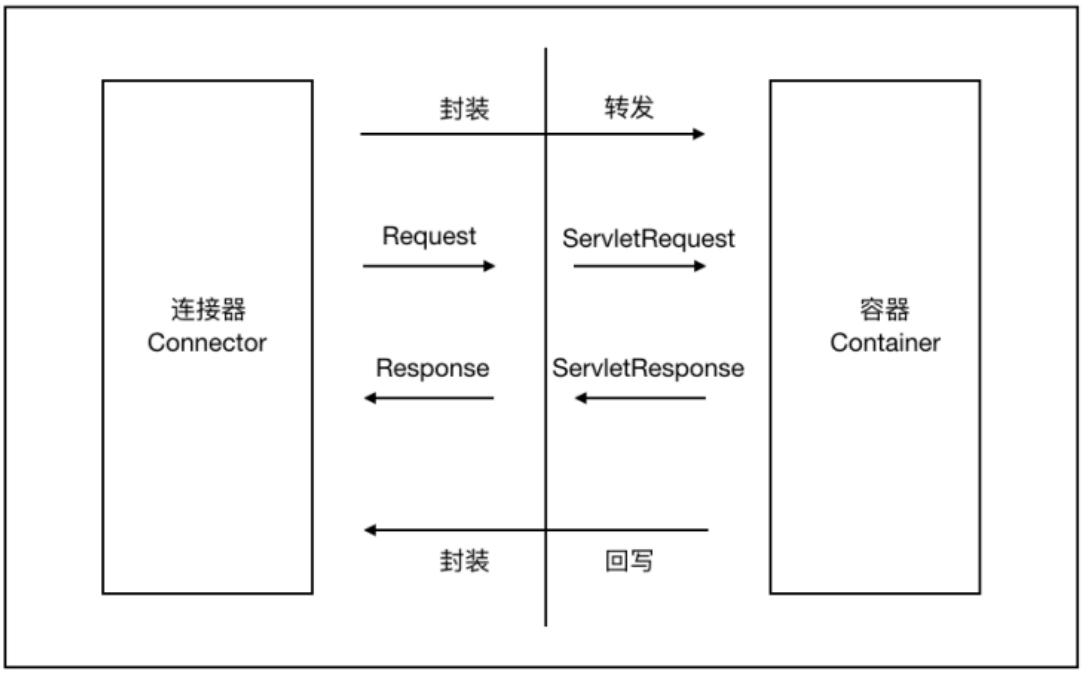
（2）Coyote 使 Catalina 容器（容器组件）与具体的请求协议及 IO 操作方式完全解耦

（3）Coyote 将 Socket 输入转换封装为 Request 对象，进一步封装后交由

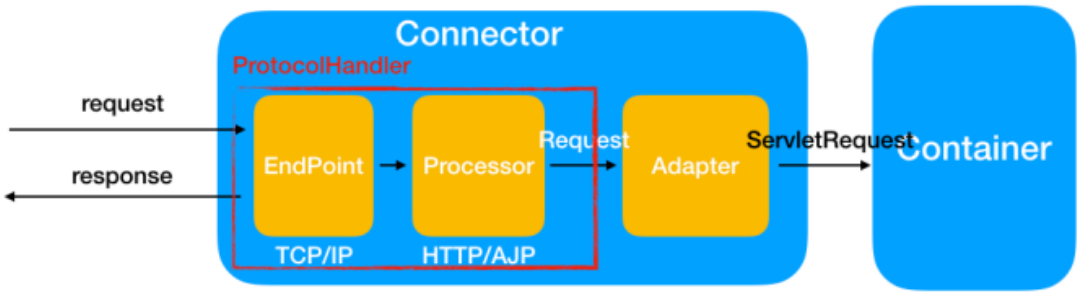
Catalina 容器进行处理，处

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



Coyote 的内部组件及流程



其中各个组件：

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

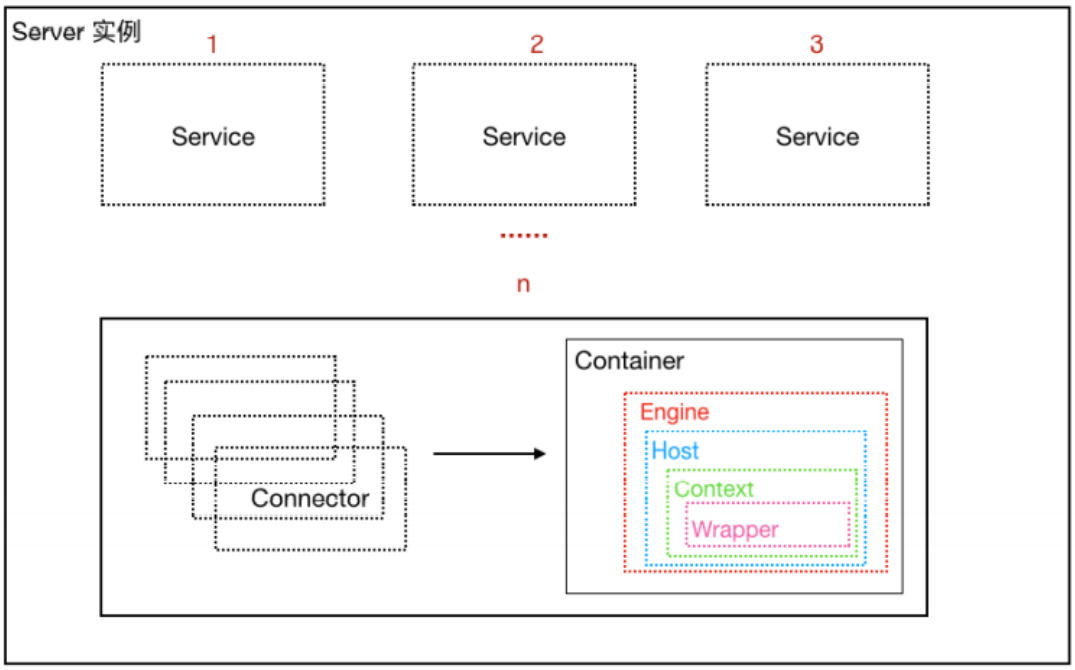
Processor：是 Coyote 协议处理接口 ，如果说 EndPoint 是用来实现 TCP/IP 协议的，那么 Processor用来实现 HTTP 协议，Processor 接收来自EndPoint 的 Socket，

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

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

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

Tomcat Servlet 容器 Catalina:



其实，可以认为整个 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，Server 创建并管理多个服务， 每个服务又可以有多个 Connector 和一个 Container。

Container 组件的具体结构

Engine：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：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