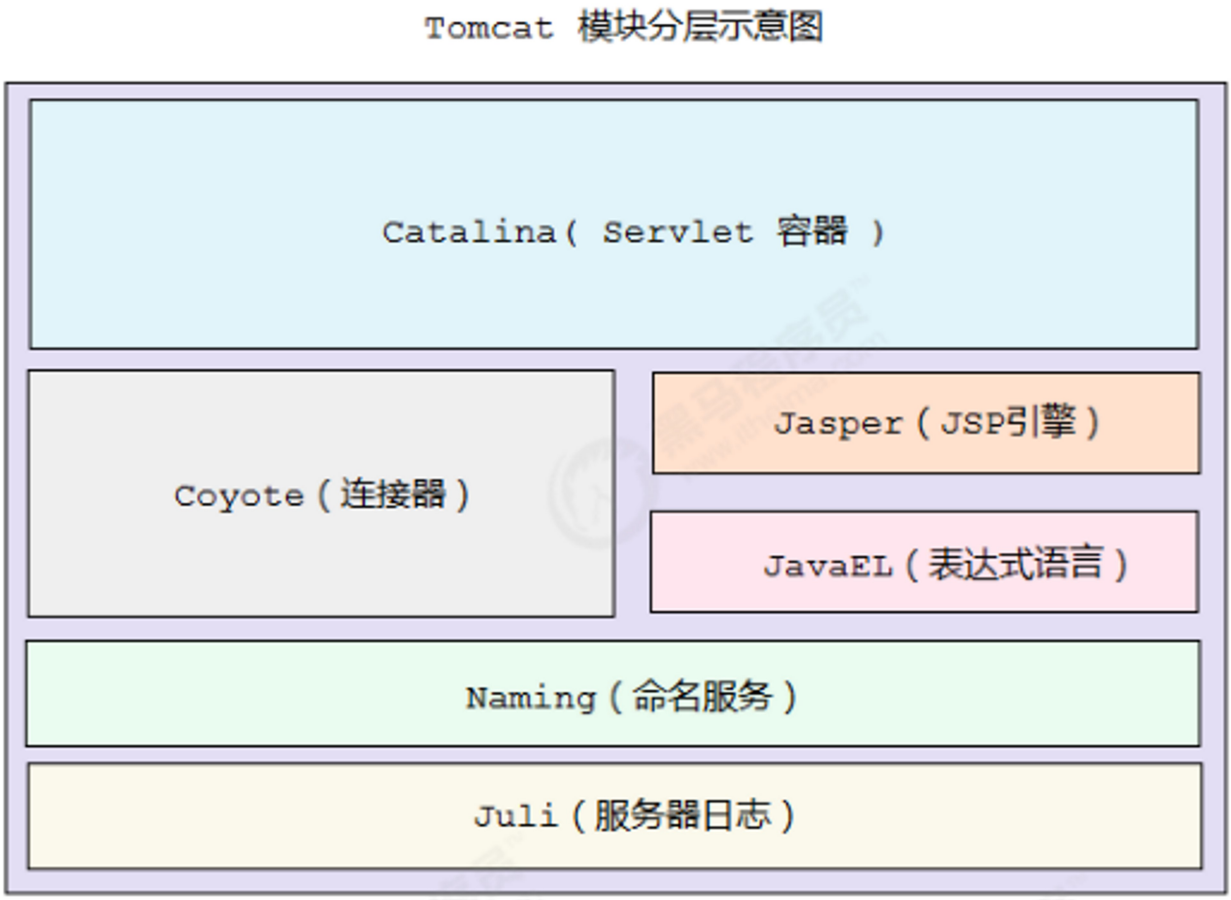
Tomcat整体架构

2020年6月4日 21:37

概述：

Tomcat 本质上就是一款Servlet容器，因此Catalina才是Tomcat的核心，其他模块都是为Catalina提供支撑的。

比如：通过Coyote模块提供链接通信，Jasper模块提供 JSP引擎，Naming提供NDI服务，Juli提供日志服务。



连接器-Coyote（用于实现Connector）

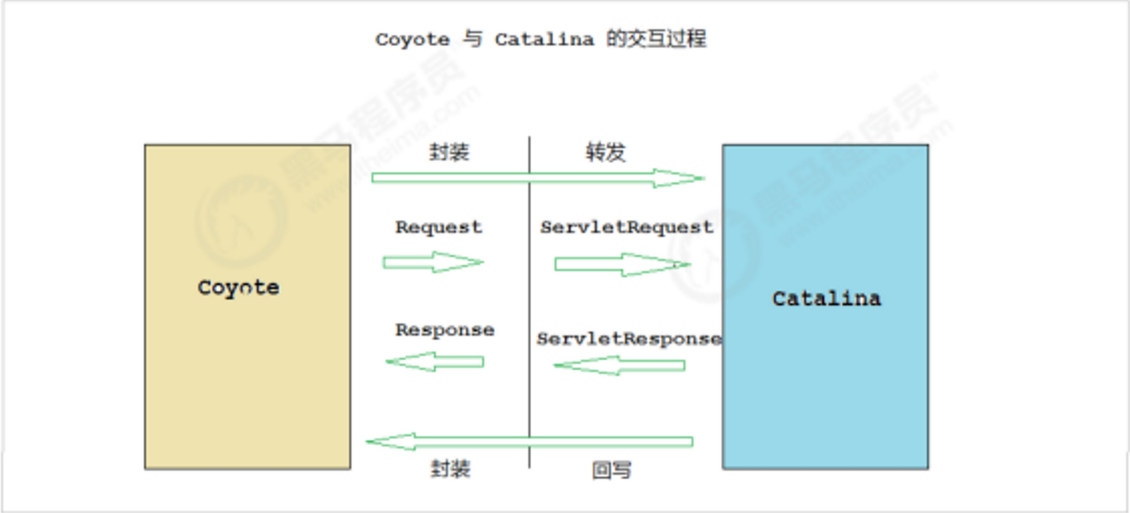
2020年6月4日 21:43

1. 概述：

   Coyote是Tomcat的连接器框架的名称，是Tomcat服务器提供的供客户端访问的外部接口。客户端通过Coyote与服务器建立连接、发送请求并接受响应。

   Coyote 封装了底层的网络通信（Socket 请求及响应处理），为Catalina容器提供了统一的接口，使Catalina容器与具体的请求协议及IO操作方式完全解耦。Coyote将Socket输入转换封装为Request对象，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。

   Coyote作为独立的模块，只负责具体协议和IO的相关操作，与Servlet规范实现没有直接关系，因此即便是Request和Response对象也并未实现Servlet规范对应的接口，而是在Catalina中将他们进一步封装为ServletRequest和ServletResponse。



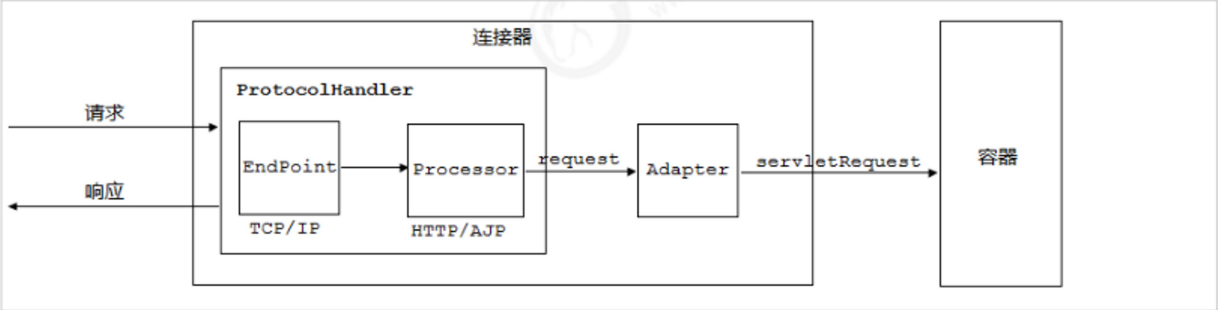
2. 组件：

   ProtocolHandler：Coyote协议接口，通过Endpoint和Processor，实现针对具体协议的处理能力。

   EndPoint：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：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“存放”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，也就意味着，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，再调用容器的Service方法。



3. IO模型与协议：

Tomcat支持的IO模型（自8.5起，移除对BIO支持）：

IO模型	描述
NIO	非阻塞I/O，采用Java NIO类库实现。
NIO2	异步I/O，采用JDK 7最新的NIO2类库实现。
APR	采用Apache可移植运行库实现，是C/C++编写的本地库。如果选择该方案，需要单独安装APR库。

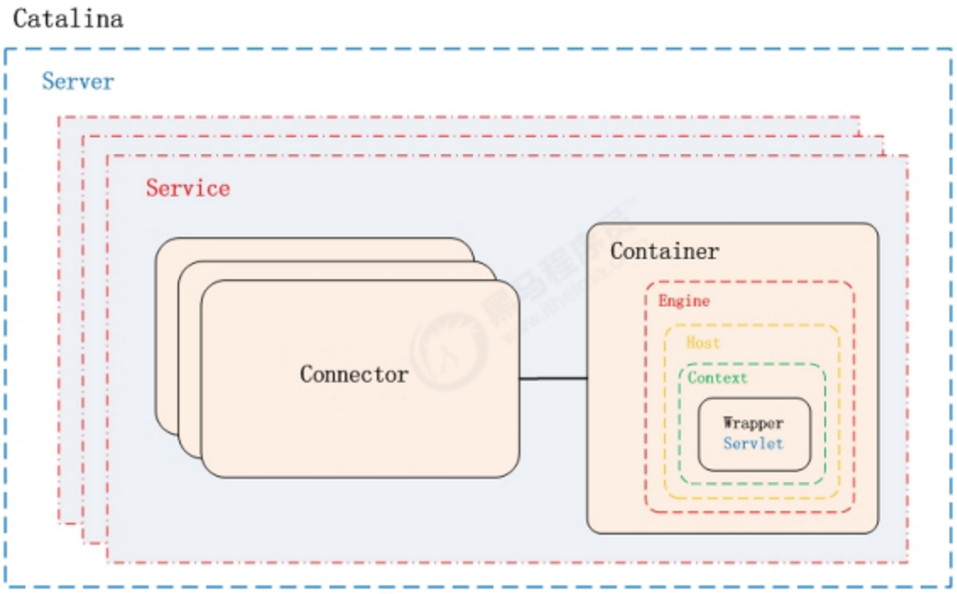
Tomcat支持的应用层协议：

应用层协议	描述
HTTP/1.1	这是大部分Web应用采用的访问协议。
AJP	用于和Web服务器集成（如Apache），以实现对静态资源的优化以及集群部署，当前支持AJP/1.3。
HTTP/2	HTTP 2.0大幅度的提升了Web性能。下一代HTTP协议， 自8.5以及9.0版本之后支持。

容器-Catalina

2020年6月5日 21:57

1. 主要组件：



Catalina负责管理Server，而Server表示着整个服务器。
Server下面有多个服务Service，每个服务都包含着多个连接器组件Connector（Coyote实现）和一个容器组件Container。
在Tomcat启动的时候，会初始化一个Catalina的实例。

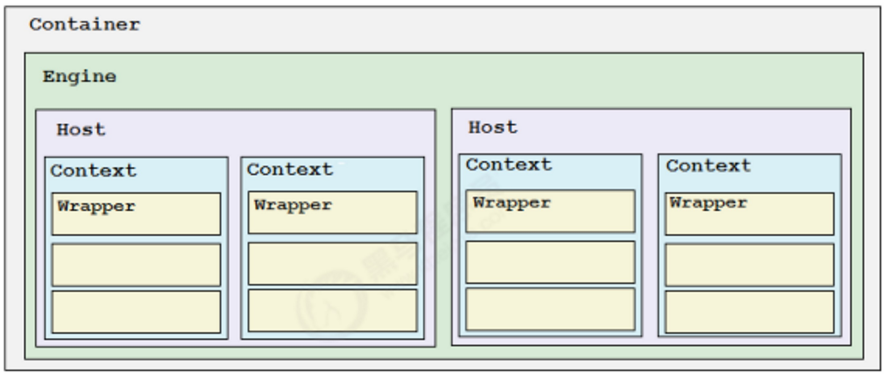
小总结：

一个Tomcat中只有一个Server，
一个Server可以包含多个Service，
一个Service只有一个Container，但可以有多个Connectors（因为一个服务可以有多个连接，如同时提供http和https连接，也可以提供相同协议不同端口的连接），
一个Container只有一个Engine，
一个Engine可以包含多个Host，
一个Host可包含多个Context（指的就是一个应用程序），
一个Context可包含多个Wrapper（指的就是Servlet）；

2. 组件职责：

组件	职责
Catalina	负责解析Tomcat的配置文件，以此来创建服务器Server组件，并根据命令来对其进行管理
Server	服务器表示整个Catalina Servlet容器以及其它组件，负责组装并启动Servlet引擎,Tomcat连接器。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
Service	服务是Server内部的组件，一个Server包含多个Service。它将若干个Connector组件绑定到一个Container（Engine）上
Connector	连接器，处理与客户端的通信，它负责接收客户请求，然后转给相关的容器处理，最后向客户返回响应结果
Container	容器，负责处理用户的servlet请求，并返回对象给web用户的模块

3. Container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；

4. server.xml:

```
<Server>
  <Service>
    <Connector/>
    <Connector/>
    <Engine>
      <Host>
        <Context></Context>
      </Host>
    </Engine>
  </Service>
</Server>
```

Tomcat就是用组合模式来管理这些容器的。具体实现方法是，所有容器组件都实现了Container接口，因此组合模式可以使得用户对单容器对象和组合容器对象的使用具有一致性。这里单容器对象指的是最底层的Wrapper，组合容器对象指的是上面的Context、Host或者Engine。