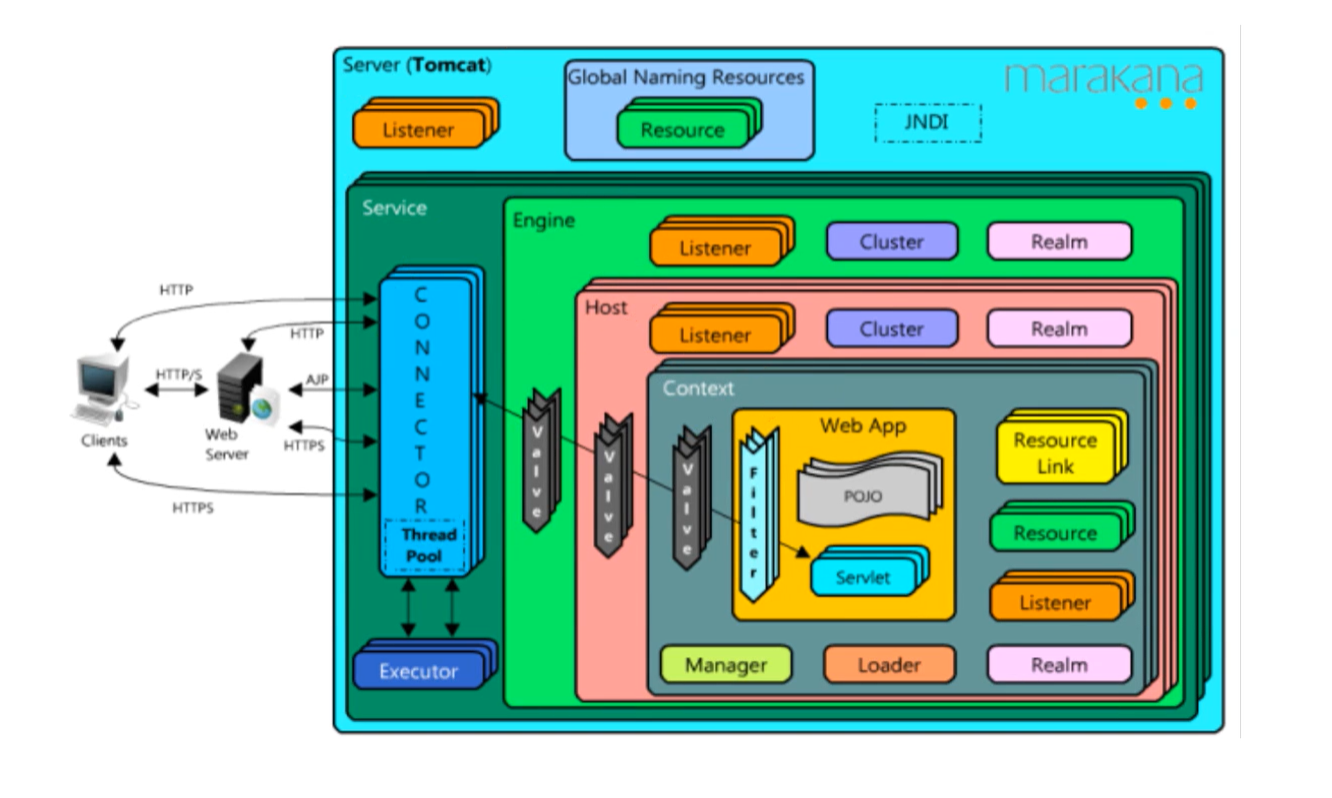
# 架构设计



Tomcat 的本质其实就是一个 WEB 服务器 + 一个 Servlet 容器，那么它必然需要处理网络的连接与 Servlet 的管理，因此，Tomcat 设计了两个核心组件来实现这两个功能，分别是连接器(connect)和容器(container)，连接器用来处理外部网络连接，容器用来处理内部 Servlet。

一个 Tomcat 代表一个 Server 服务器，一个 Server 服务器可以包含多个 Service 服务，Tomcat 默认的 Service 服务是 Catalina，而一个 Service 服务可以包含多个连接器，因为 Tomcat 支持多种网络协议，包括 HTTP/1.1、HTTP/2、AJP 等等，一个 Service 服务还会包括一个容器，容器外部会有一层 Engine 引擎所包裹，负责与处理连接器的请求与响应，连接器与容器之间通过 ServletRequest 和 ServletResponse 对象进行交流。

## Server.xml文件



# 开发时war 和 war exploded

## war：先打成war包，再发布，正常的发布就是用的这种

工程target目录下会生成war包, 并且解压后的war包会被部署到tomcat的webapp 目录下。

注意：如果application-context 设置为 /， tomcat目录下不会有解压后的war文件，只有指定文件目录，才会生成文件夹，这个文件夹作为项目的根目录

## war exploded： 模式是直接把文件夹、jsp页面 、classes等等移到Tomcat 部署文件夹里面，进行加载部署。因此这种方式支持热部署，一般在开发的时候也是用这种方式。

# 部署web项目方式

## 打包成war放在webapps目录下(war解压后也是文件夹)

## 拷贝项目的target里面的项目文件夹复制进webapps

# Catalina

1. 设置了一些系统属性
2. 通过 digester 解析 server.xml 配置文件
3. 调用Server#init 初始化

# Acceptor(Runnable)

主要监听网络连接并且进行任务分发的后台线程

负责接收网络请求，建立连接，连接建立之后，将这个socket连接交给Poller

# Poller(Runnable)

在启动NioEndpoint的时候，会启动pollers。

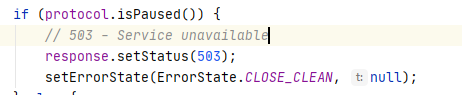
每个Poller都有一个自己的Selector对象，

在Poller的构造方法中通过Selector.open方法生成

主要的工作是将通道与选择器关联，然后

# Http11Processor是最后的http处理类

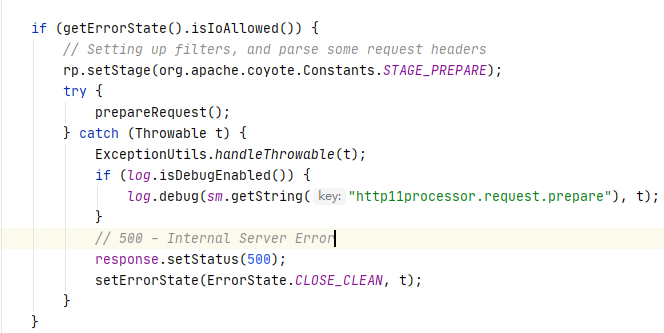
## 服务无效(503)



## 找不到服务(400)



## 服务器内部错误(500)



# CoyoteAdapter调用容器

调用容器()

# 一个网络请求过程

主线程

1. 在tomcat程序启动时，就会启动Acceptor类和Poller类的后台线程

一个后台线程

1. Acceptor类不断循环接收socket，调用Endpoint# setSocketOptions
2. 使用Wrapper封装socket，调用Poller#register，放到一个队列中

一个后台线程

1. 遍历队列，将socket与selector关联
2. 调用Endpoint#processSocket，给线程池添加任务
3. 线程池会调用SocketProcessor父类实现的run线程方法，最后调用自身的doRun
4. 调用ConnectionHandler#process，
5. 调用AbstractProcessorLight#process
6. 调用Http11Processer#service，解析协议

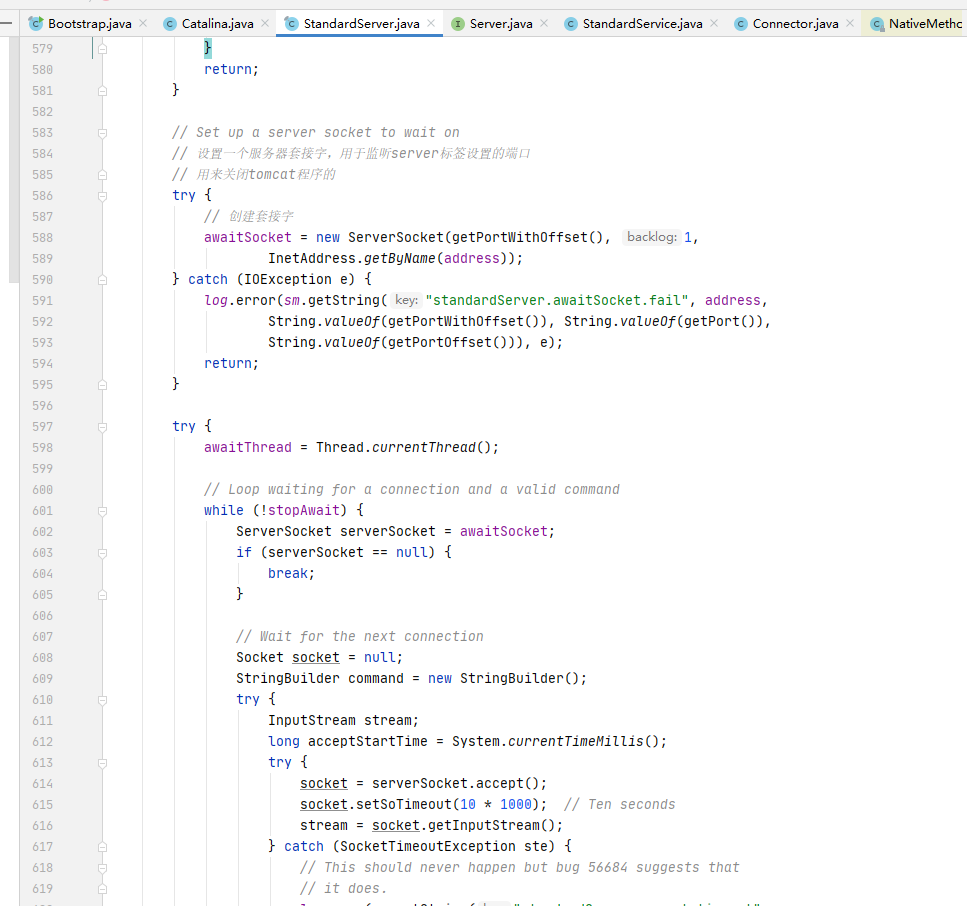
10、调用CoyoteAdapter#service，封装request和response，调用容器链

# Server标签的port和shutdown参数设置

## 配置文件图片



## StanderServer的socket



这个socket就是用来监听server标签设置的端口，当我们用给这个端口发送shutdown设置的命令就可以关闭tomcat进程

# 访问不到servlet的原因

## doXX方法不能再去调用父类的doXX方法(去掉super.doXX（）)