Tomcat架构解析

T H A N K Y O U F O R W A T C H I N G

主讲老师：King QQ: 2962938812

- 主要内容
  - 目录结构: `config` 中各种配置文件的用途和运用场景 `server.xml` 、 `web.xml` 等
  - 部署：应用的三种部署方式
  - Tomcat的整体架构：connector和container
  - 连接器：阻塞式I/O和非阻塞式I/O，线程池以及请求处理
  - 容器：容器以及责任链模式
  - 源码预览：下载、启动、调试源码

01 Tomcat的目录结构

配置文件

server.xml:核心配置文件

web.xml:Servlet标准的web.xml部署文件

lib

Tomcat依赖的jar包

logs目录

localhost-xxx.log

catalina-xxx.log

webapps目录

三种应用部署的方式

conf	配置文件
bin	执行命令
lib	加载的jar包
logs	日志目录
webapps	默认应用程序

Java自带的工具jconsole

1. server.xml修改连接池的大小

2. server.xml修改I/O模式(BIO与NIO)

3. server.xml去掉AJP的Connector

4. server.xml去掉access-log日志

03 Tomcat的三种部署模式

一、显式部署

1.添加context元素方式(server.xml)

```
<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
        <Context path="/Demo1" docBase="d:/Demo1" reloadable="true"></Context>
</Host>
```

2.创建xml的方式

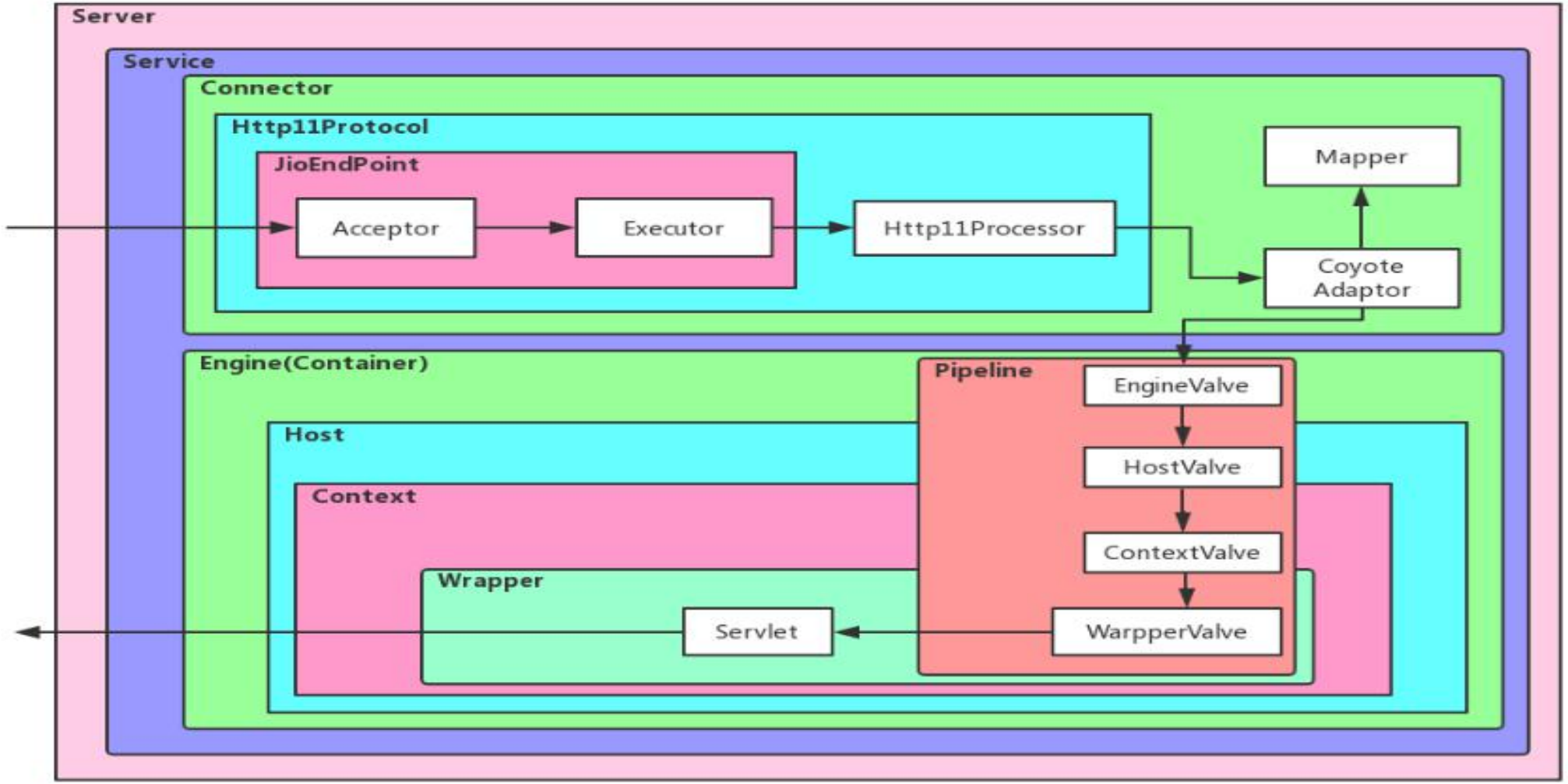
创建的文件conf/Catalina/localhost

```
<?xml version="1.0" encoding="UTF-8"?>
<Context docBase="d:/Demo1" reloadable="true"></Context>
```

二、隐式部署

将一个war文件或者整个应用程序复制到Tomcat的webapps

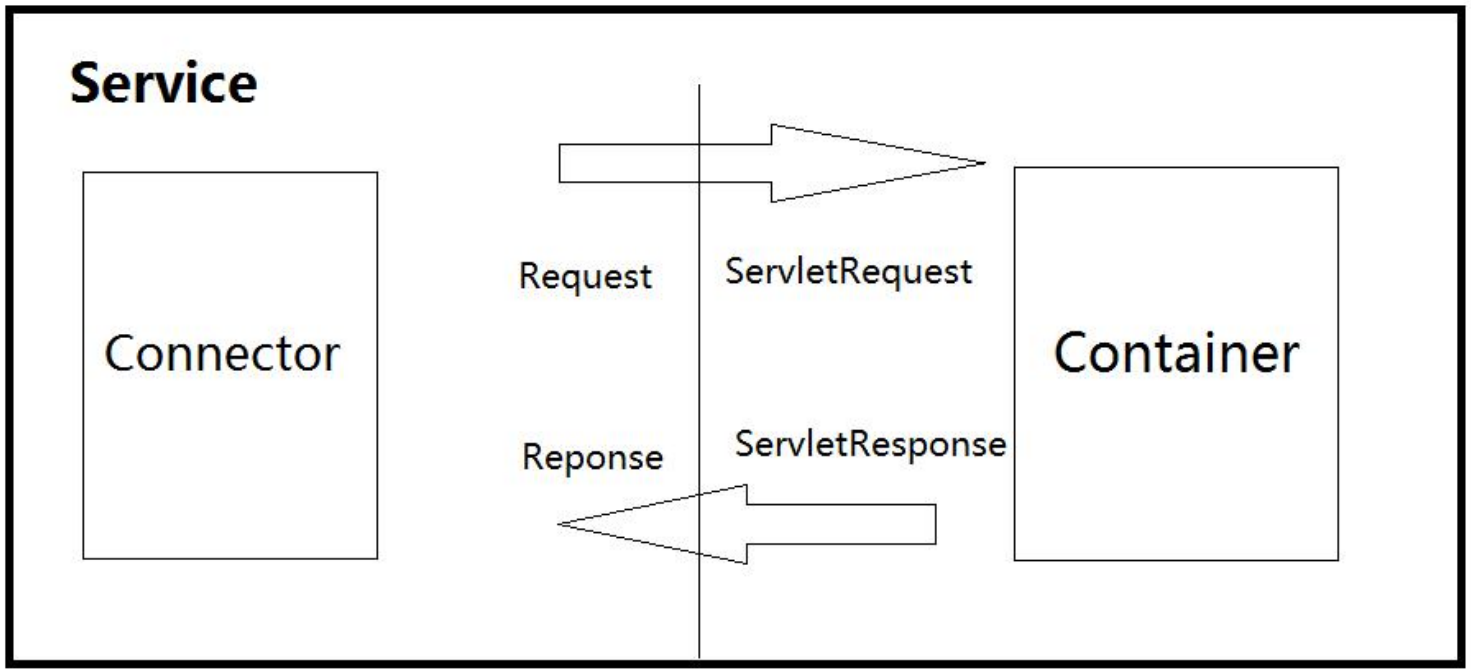
04 Tomcat总体架构

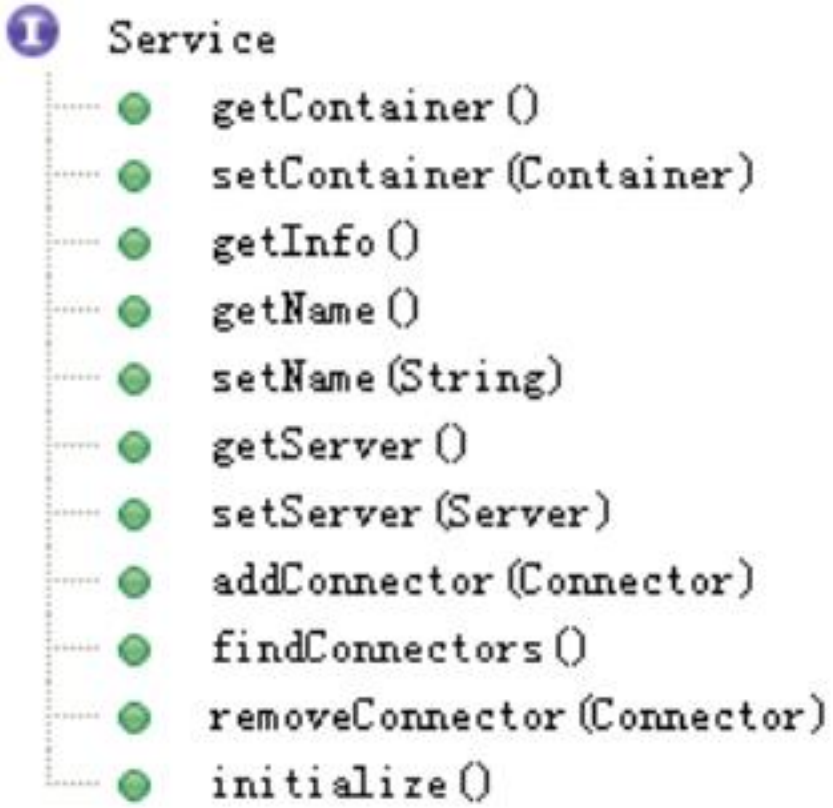


05 两大核心组件

Connector组件：连接器，主要负责Tomcat与客户端的通讯

Container组件：Servlet容器





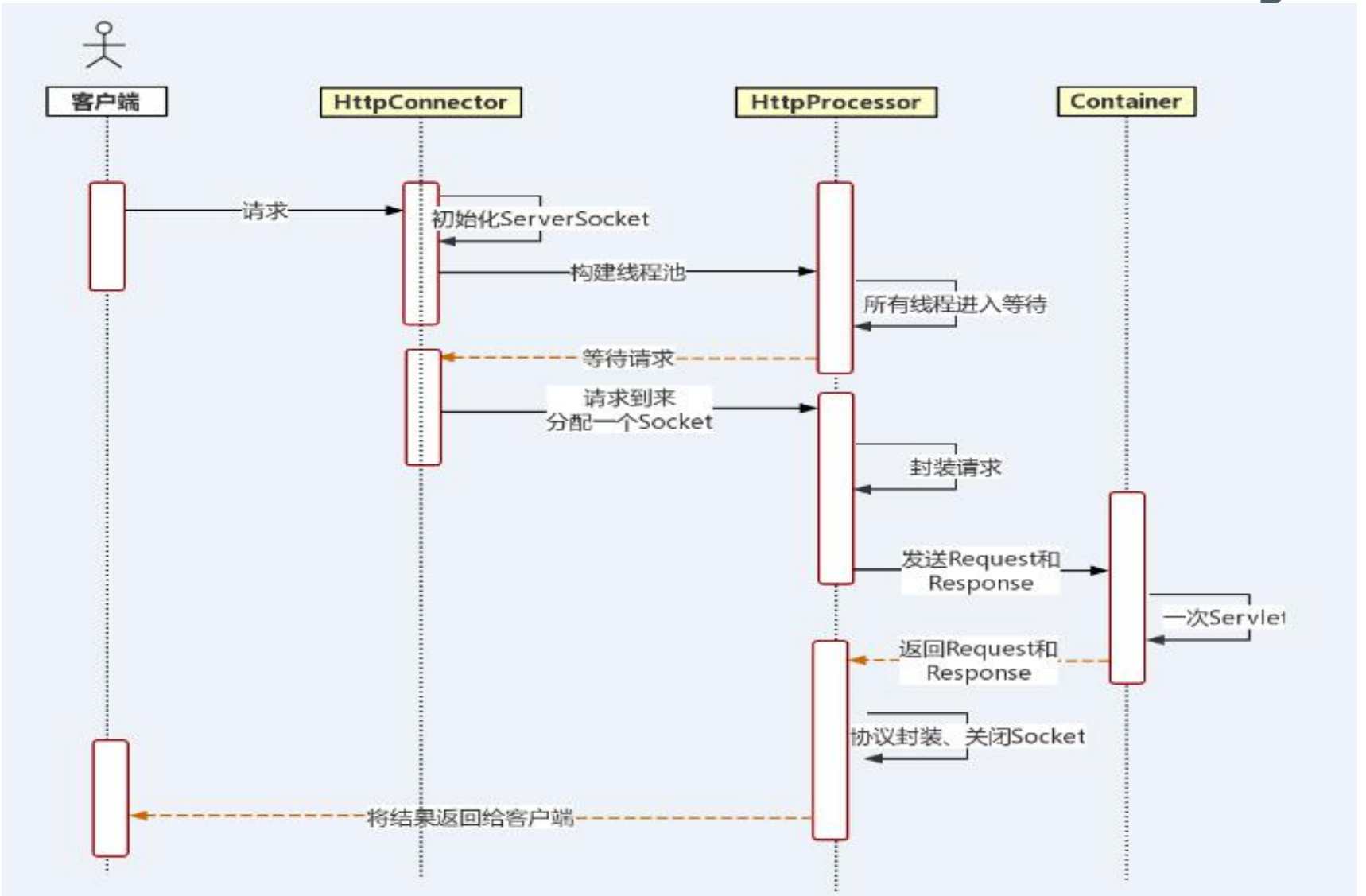
06 一个请求的处理流程

1.监听服务器端口，读取来自客户端的请求。

2.讲请求数据按照指定协议进行解析。

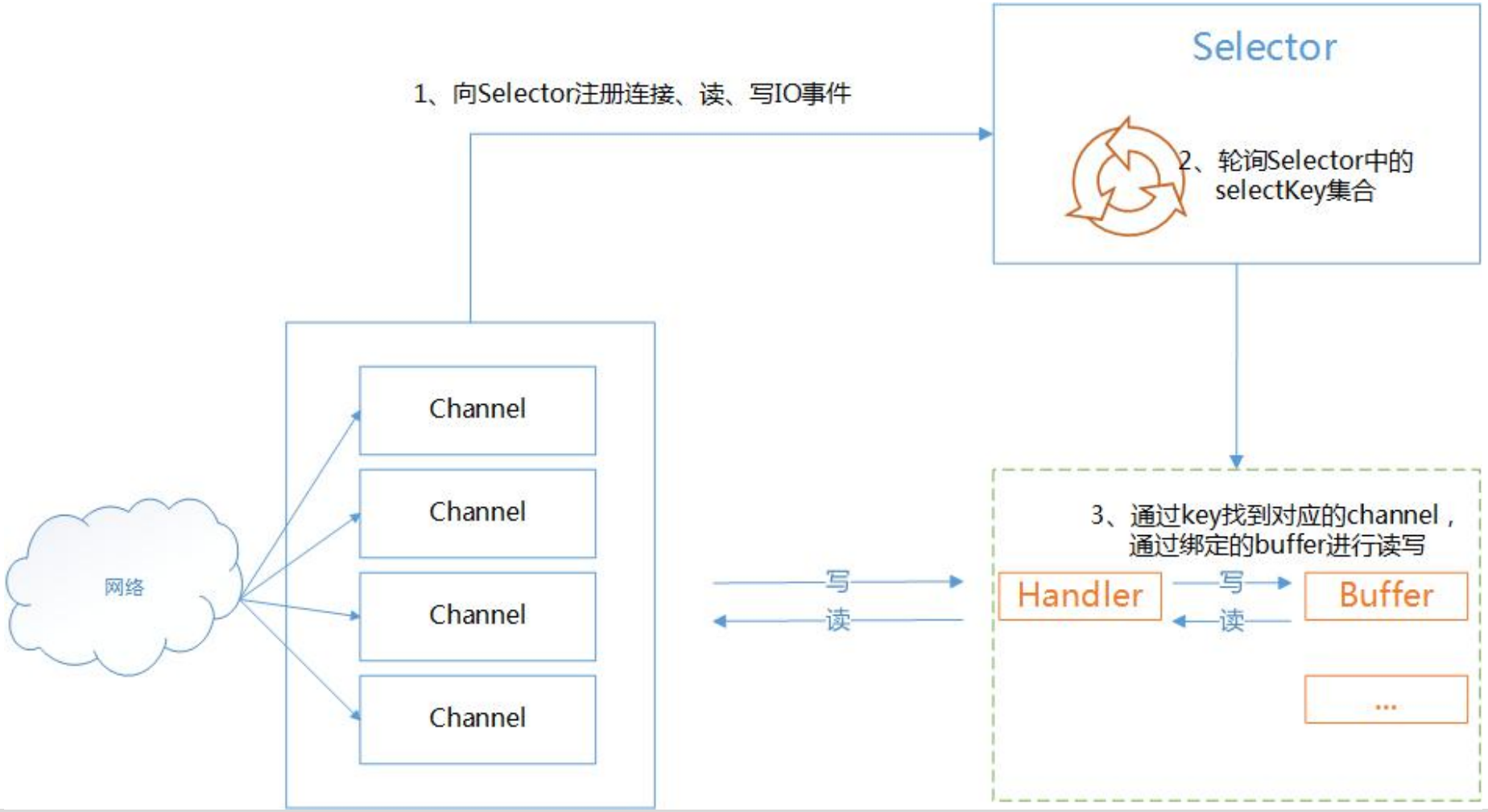
3.根据请求地址匹配正确的容器进行处理。

4.将响应返回客户端。



07 连接器的非阻塞模式(NIO)

通道(Channel)、缓冲区(Buffer)、选择器(Selector)



容器的责任链模式

1.请求被Connector组件接收，创建Request和Response对象。

2.Connector将Request和Response交给Container,先通过Engine的pipeline组件流经内部的每个Valve。

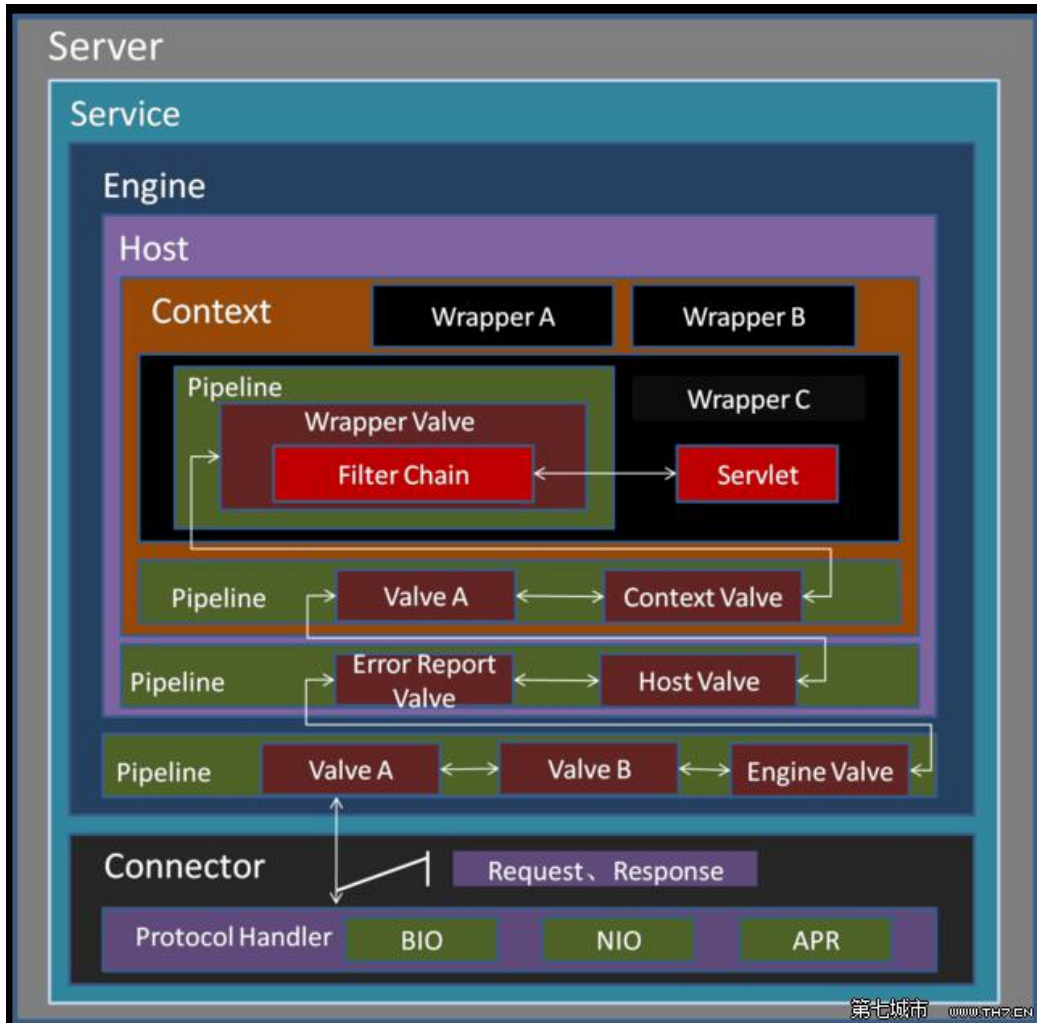
3.请求流转到Host的pipeline组件中，并且经过内部Valve的过滤。

4.请求流转到Context的pipeline组件中，并且经过内部的Valve的过滤。

5.请求流转到Wrapper的pipeline组件中，并且经过内部的Valve的过滤。

6.Wrapper内部的WrapperValve创建FilterChain实例，调用指定的Servlet实例处理请求。

7.返回



09 Tomcat源码预览

下载和部署源码

启动入口类Bootstrap

启动大致流程：Bootstrap -> catalina -> server -> service

模板模式与生命周期管理（Lifecycle）

完 毕！

享学课堂：King老师