

Tomcat 原理与实战



李良召 (John, 和风赛跑)

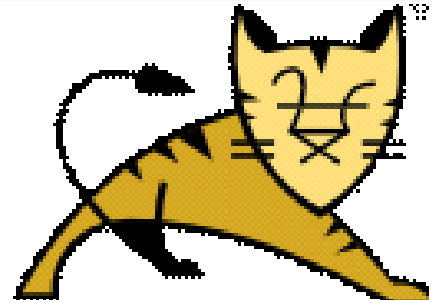
iamjohnli@gmail.com

<http://blackhouseapp.com>

提纲

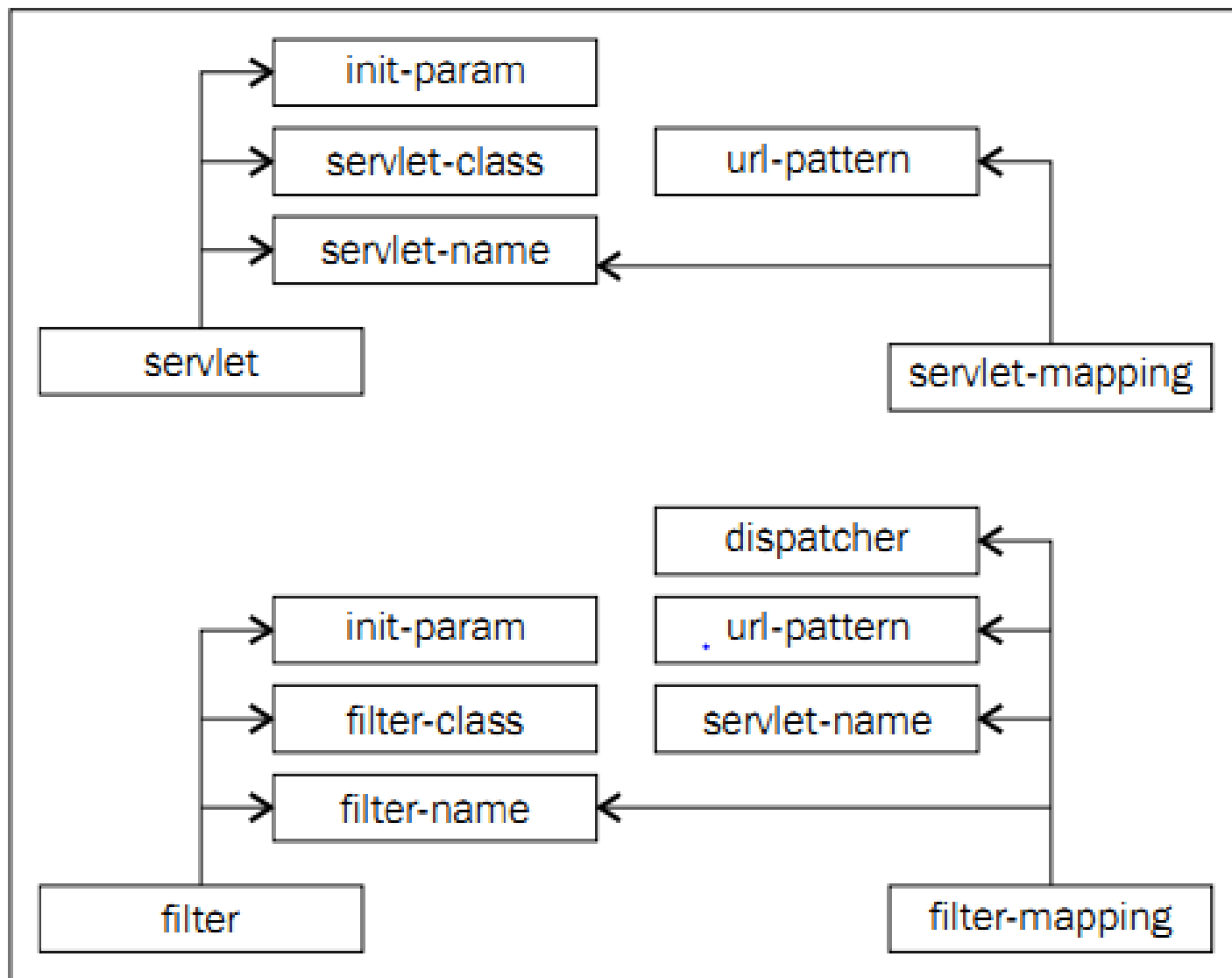
- **原理**
 - 概述 Tomcat
 - 架构设计
 - 连接器模型 (BIO 与 NIO)
 - 容器模型 (组合模式与管道模型)
- **实战**
 - 集群与多节点
 - (1) Session/Cookie 支撑有 / 无状态
 - (2) 部署, 日志, 故障
- **新特性**
- **调戏下老猫**

概述 Tomcat

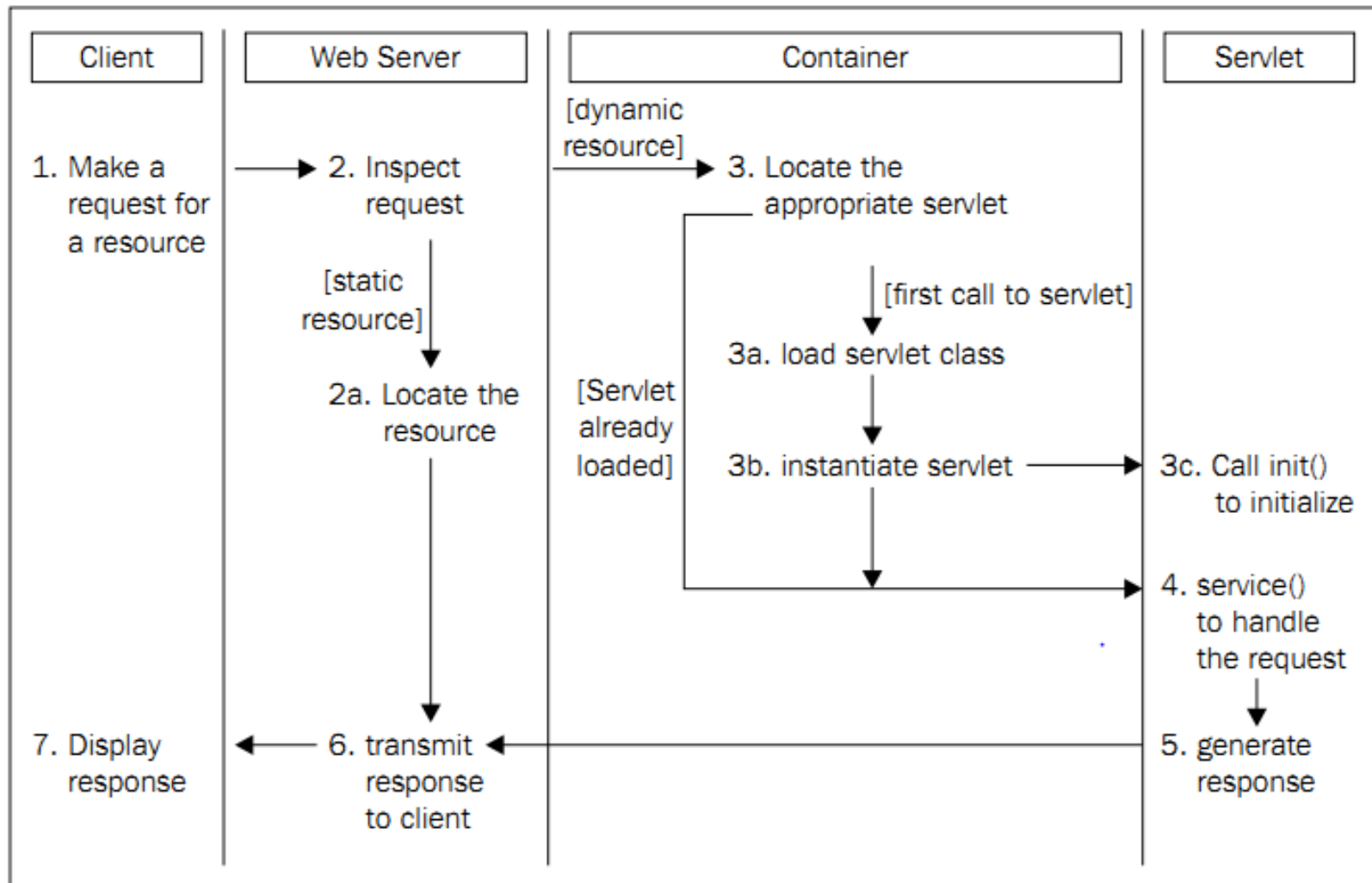


- Tomcat 到底是个什么东西
 - 抛弃那些标准，进行总结
 - 监听了某个端口，一般都是 8080, 浏览器能访问
 - 读取 webapp 目录 (解压 war), 加载类和资源
 - 读取浏览器的请求信息 (get/post, 访问路径等)
 - 调用我们编写的 listener , servlet 等，处理逻辑

回想写过的 Web.xml

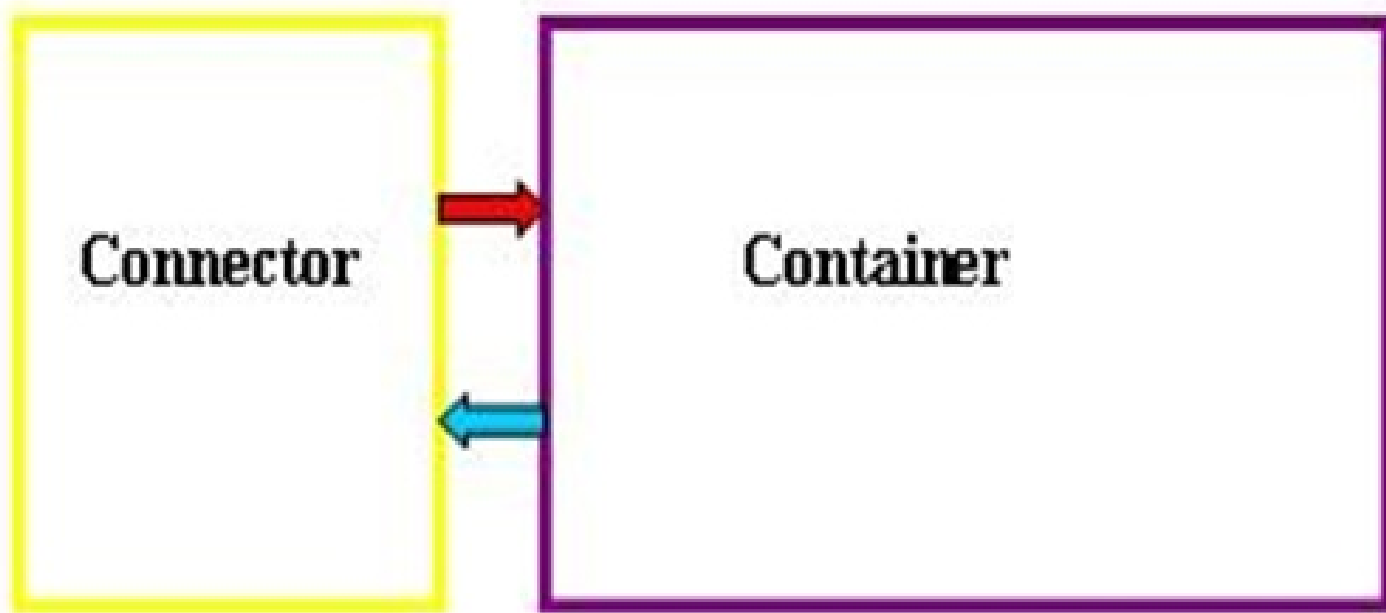


Servlet 容器



服务器模型组成

Server



抓取请求数据

Requested Resource

Request Method

HTTP Protocol version

POST /MyServlet?app=Locator HTTP/1.1

Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword,
application/x-shockwave-flash, */*

Referer: http://www.swengsol.com/MyServlet?app=SelectProduct

Accept-Language: en-us

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;

InfoPath.1; .NET CLR 2.0.50727; MS-RTC LM 8)

Host: www.swengsol.com

Content-Length: 33

Connection: Keep-Alive

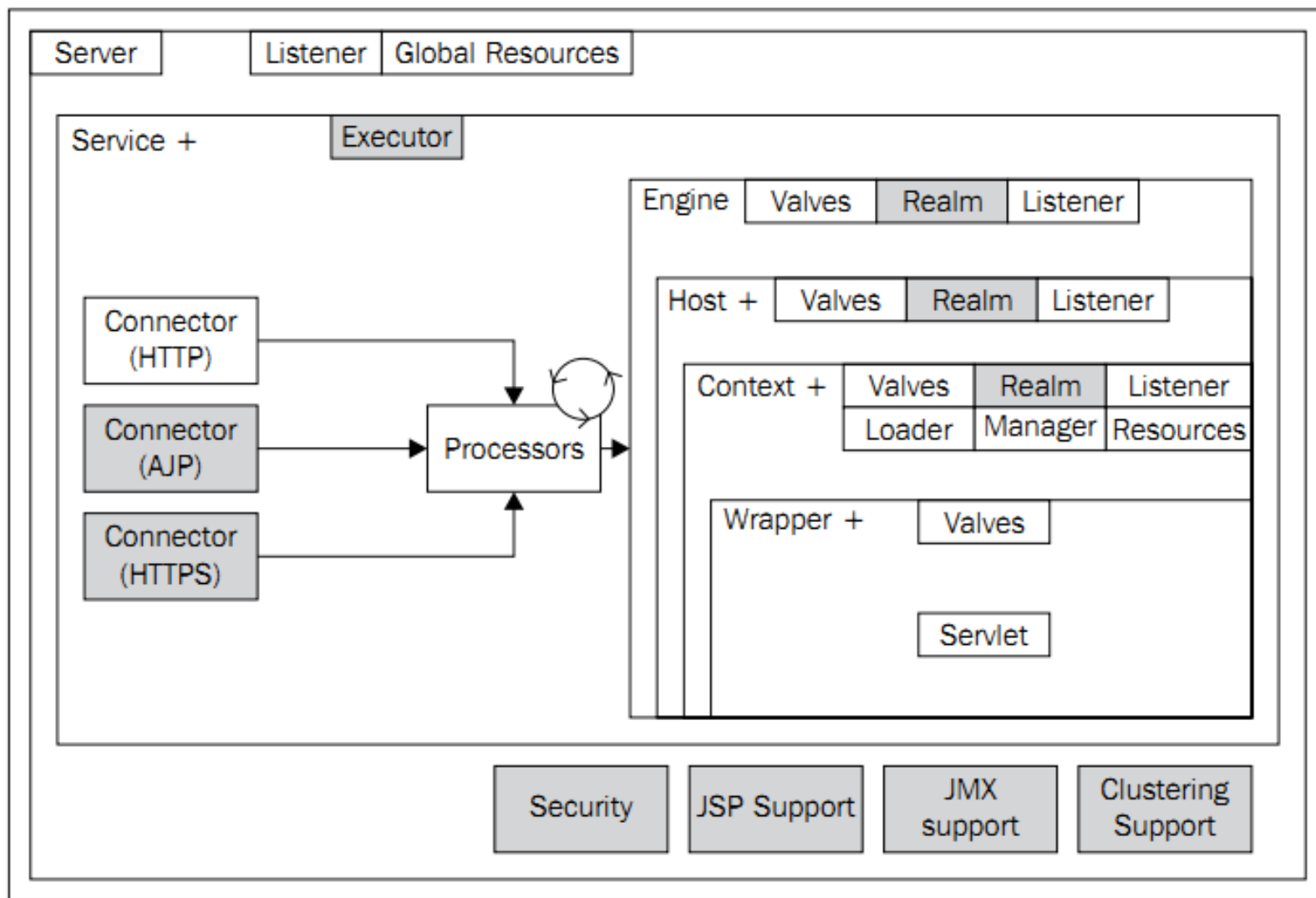
Pragma: no-cache

Cookie: JSESSIONID=2F8D401C1D1D192981DE4FE7B0D82601

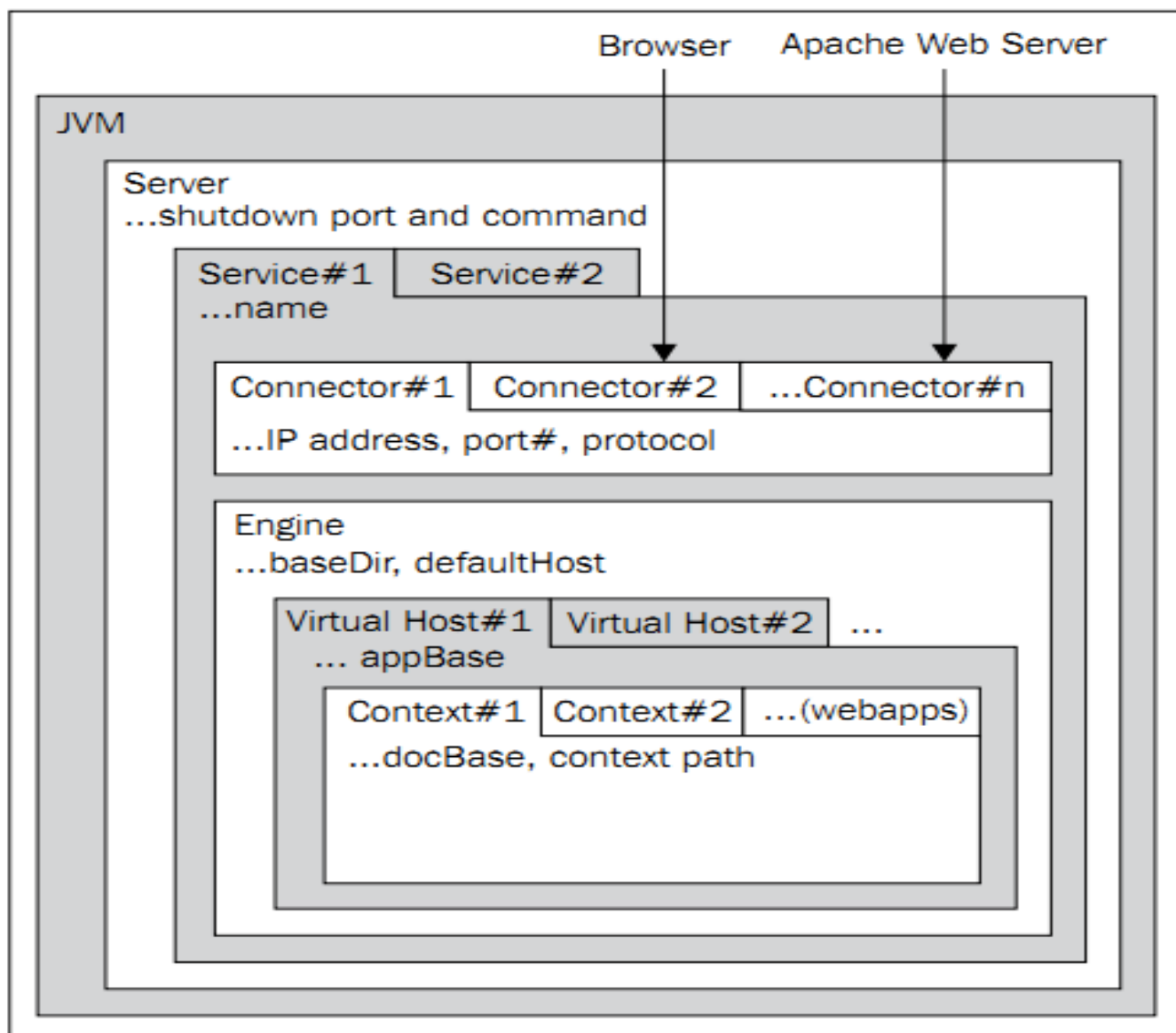
Post Data

firstName=Damodar&lastName=Chetty

Tomcat 容器

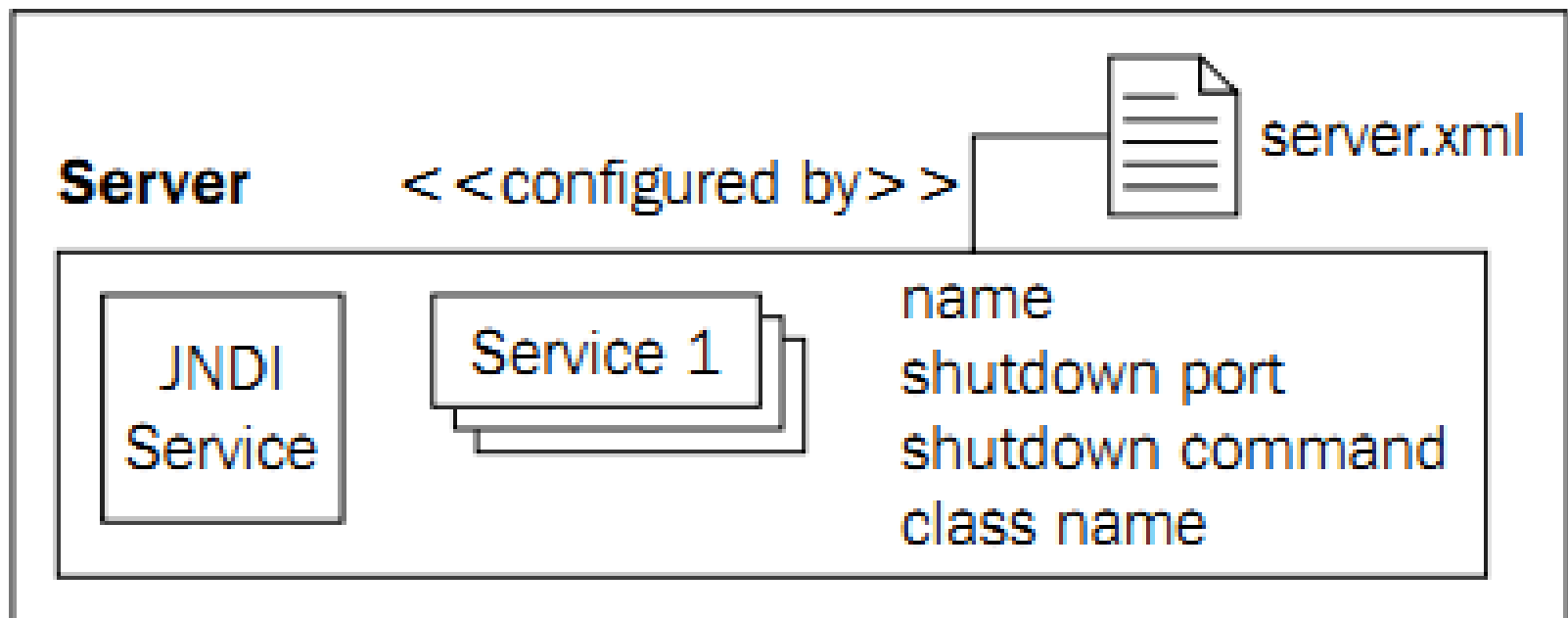


容器结构



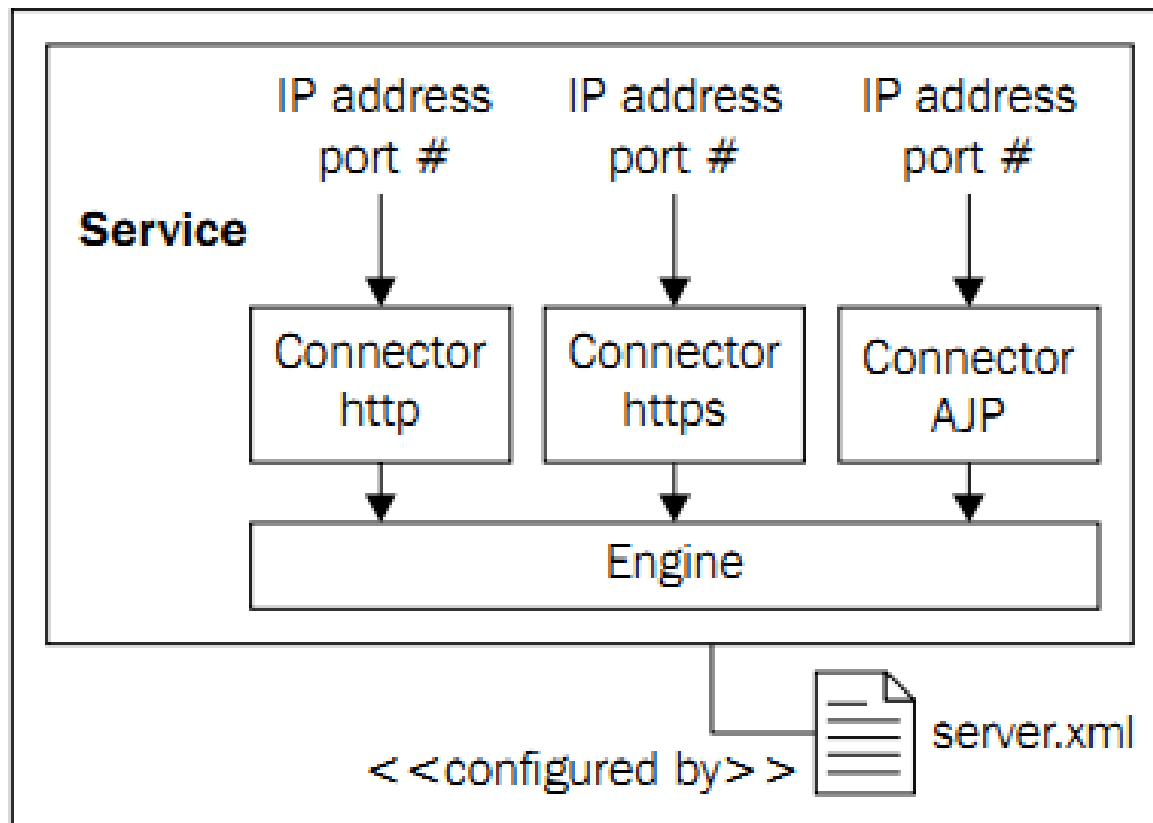
Tomcat Server

- 顶级元素
 - Tomcat 实例



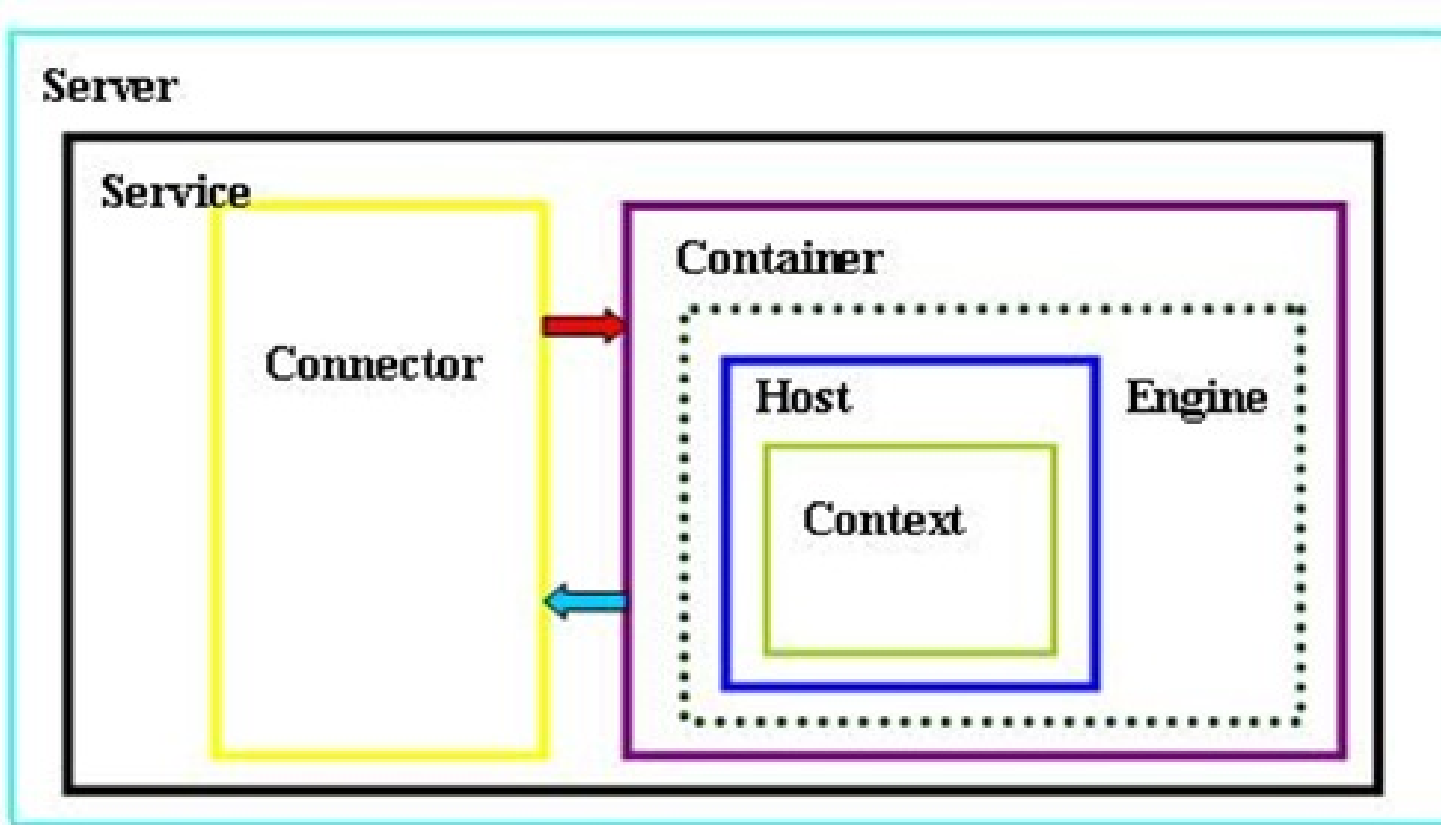
Tomcat Server/Service

- 处理请求逻辑组件集合



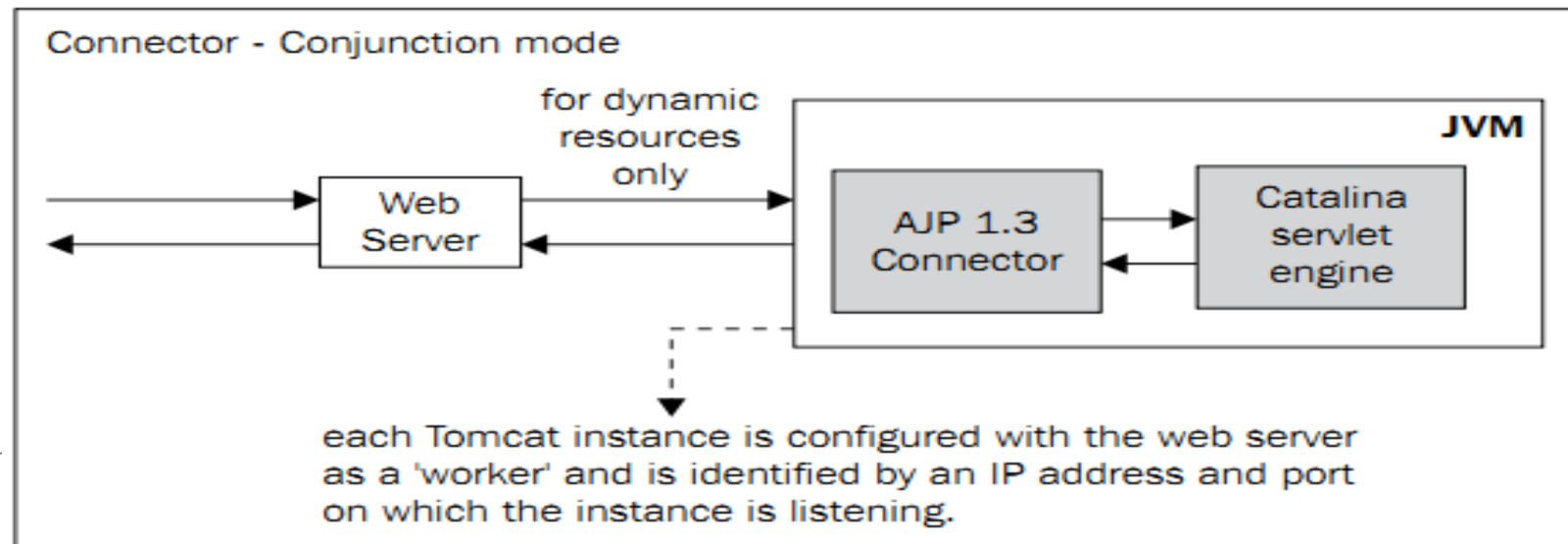
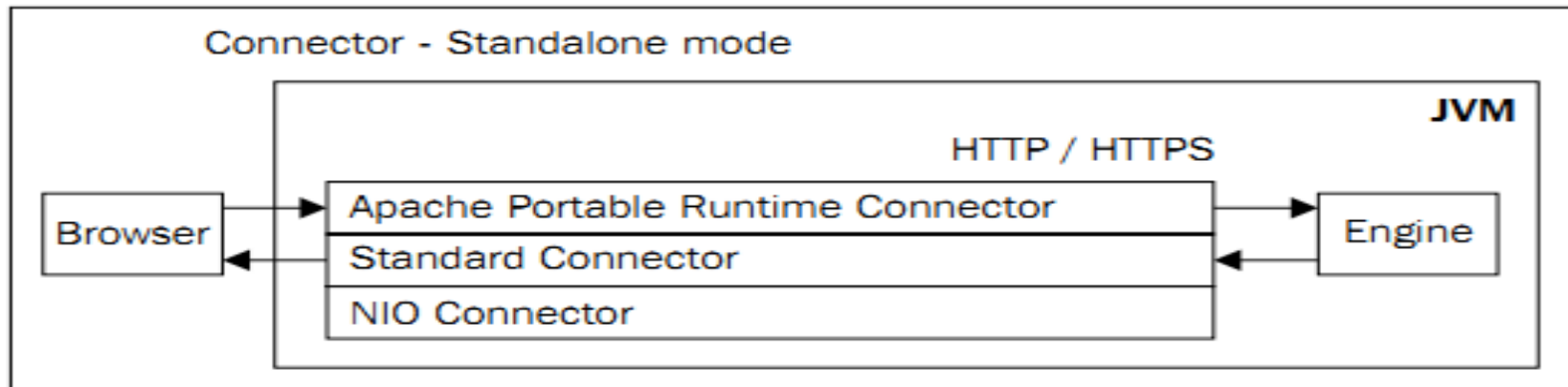
抽象出 Service

- Connector 和一个 Container 有机的组合在一起构成 Service 供 Servlet 服务



Tomcat Server/Service /Connector

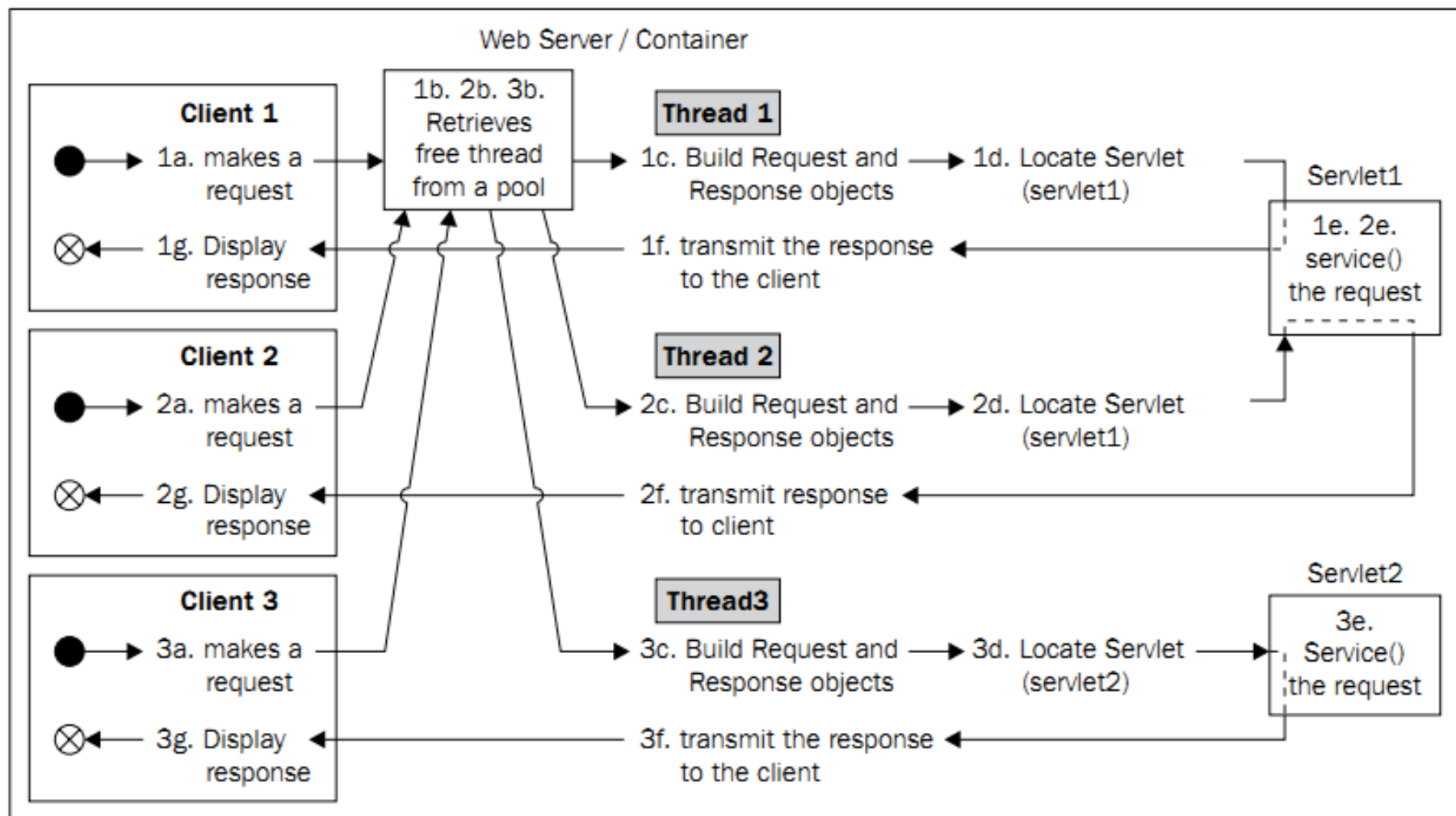
- 连接器



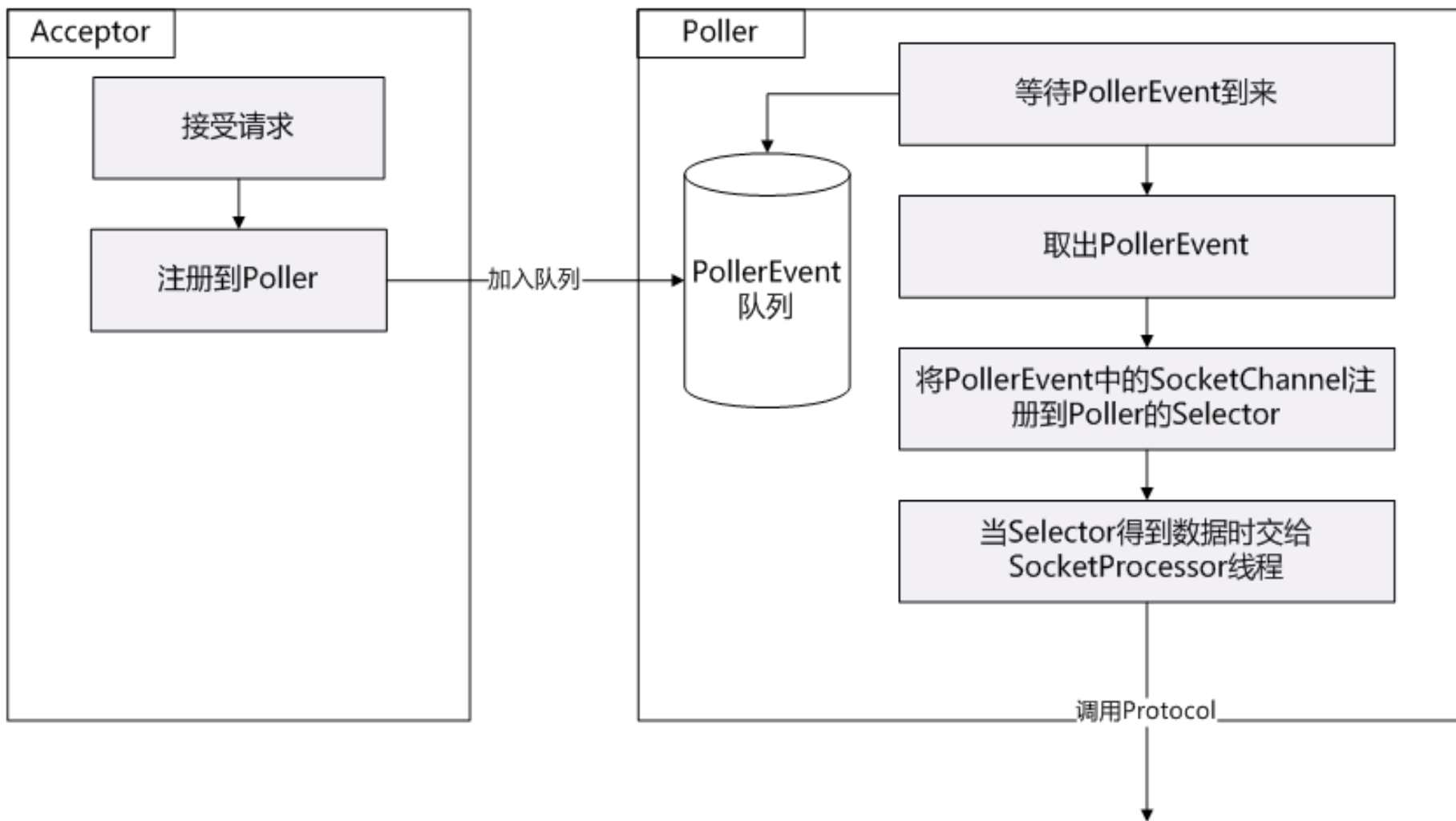
Connector

- 网络连接
 - 监听端口并处理连接
 - 总就分配一个线程针对每个请求进行解析
 - 为每个请求访问分配一个线程处理
 - » 线程优化为线程池调用
 - NIO 进行监听事件

多线程处理

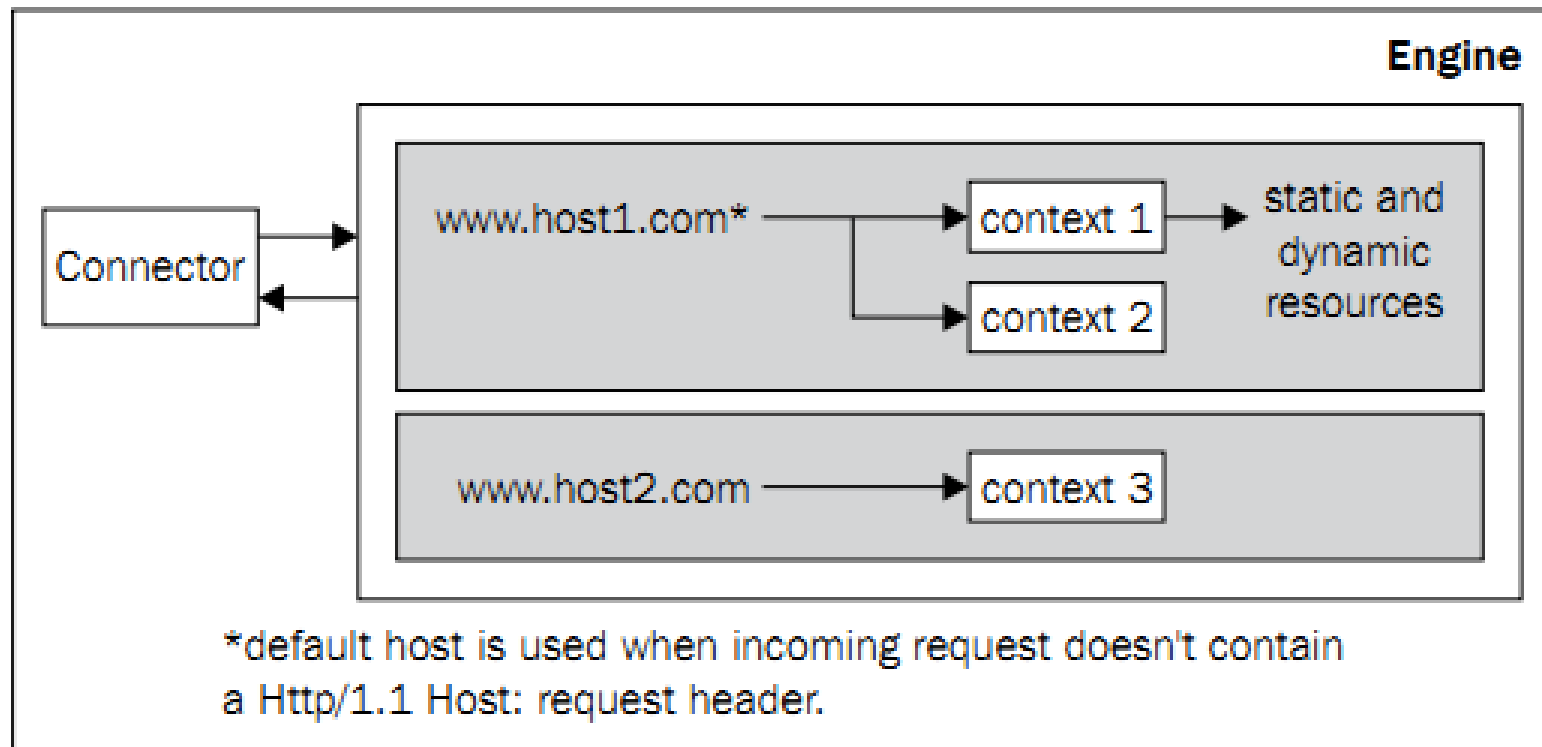


NIO 处理过程

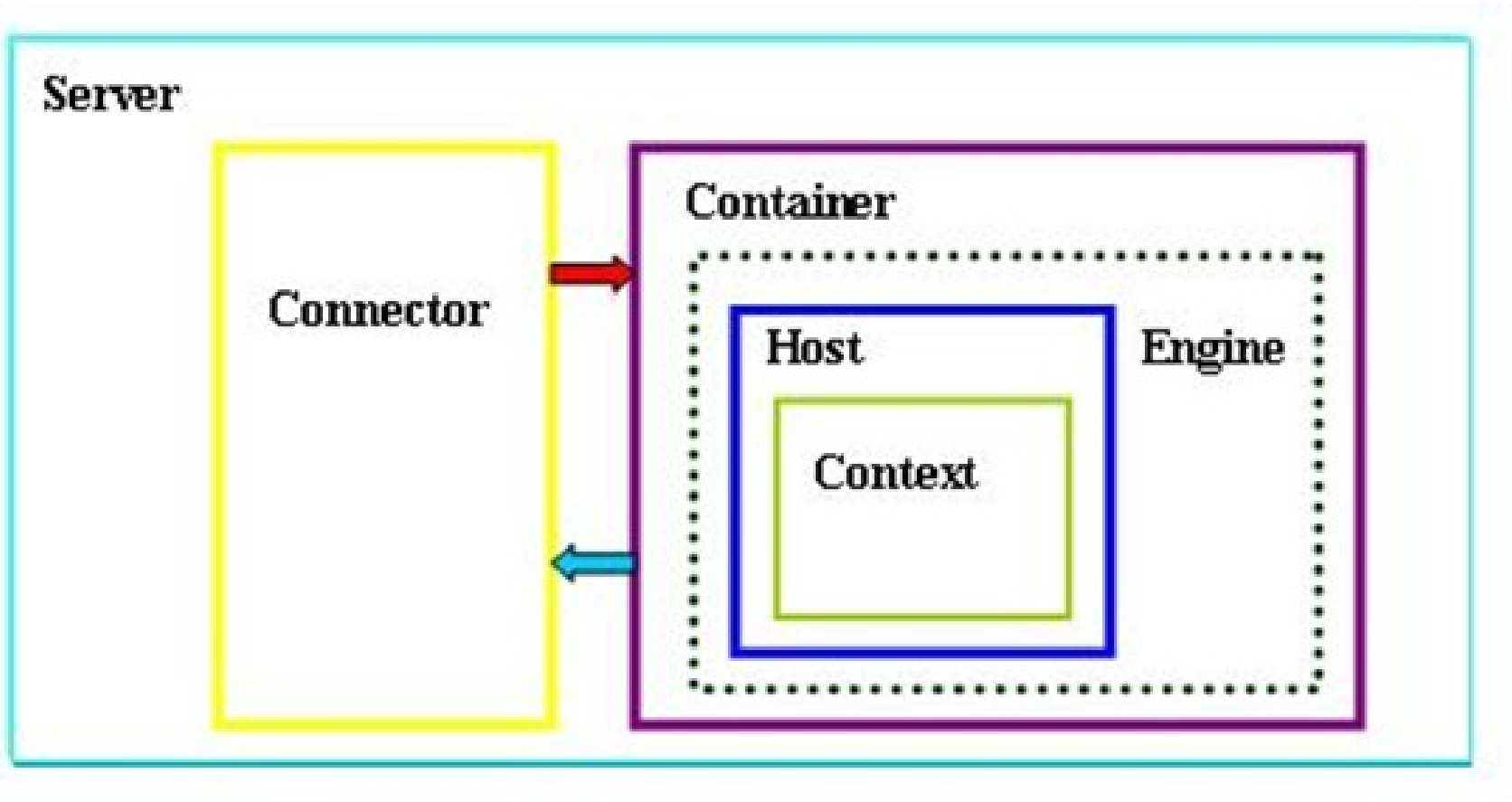


Tomcat Server/Service /Engine

- Servlet 引擎

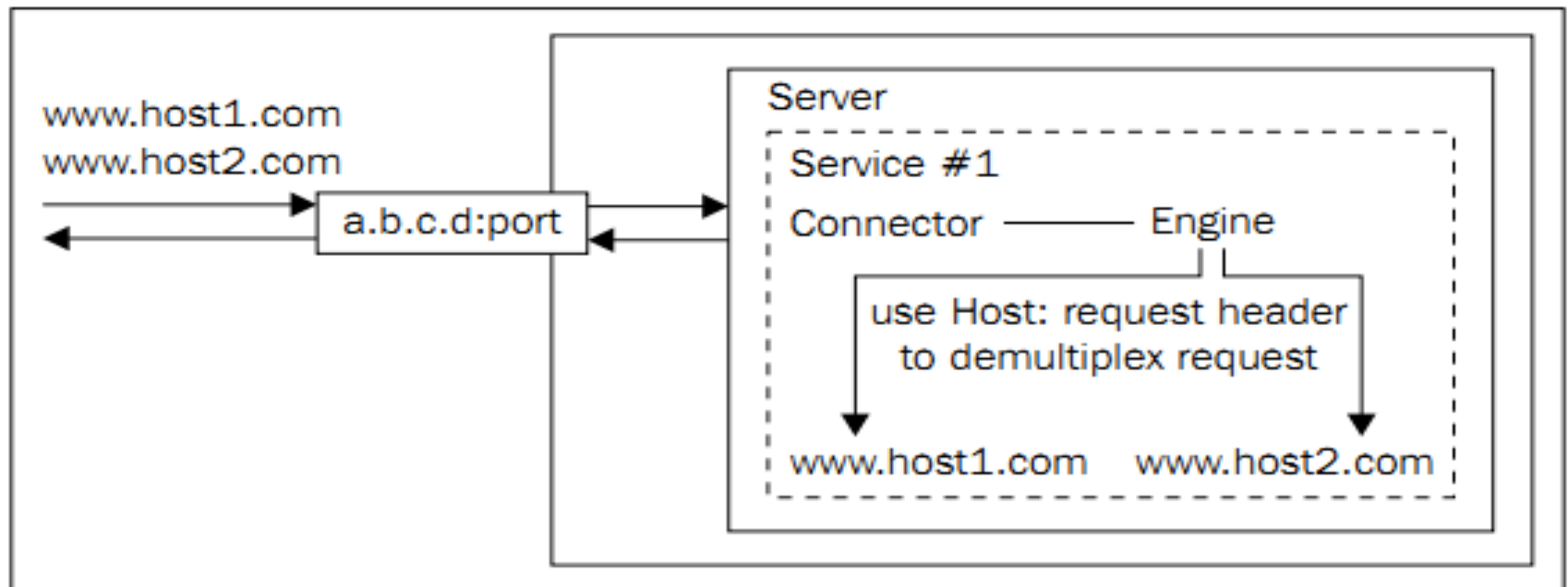
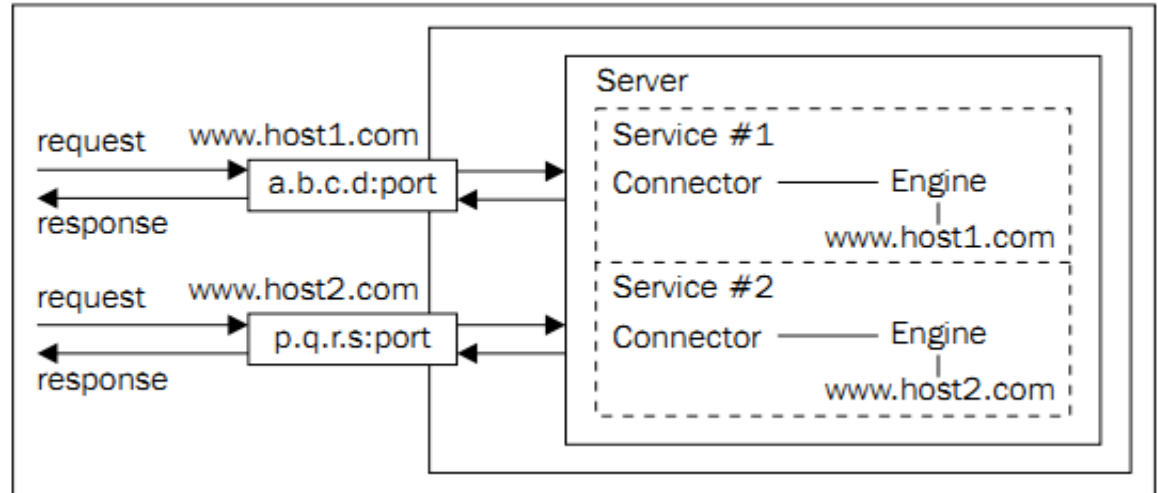


Engine 包含 HOST



Tomcat Server/Service /Engine/Host

- 主机
 - 基于域名
 - 基于 IP 地址



请求数据中的 HOST

http://www.localswengsol.com:8080/webapp1/

GET /webapp1/ HTTP/1.1

Host: www.localswengsol.com:8080

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0;

Accept: text/html,application/xhtml+xml,application/

Accept-Language: en-us,en;q=0.5

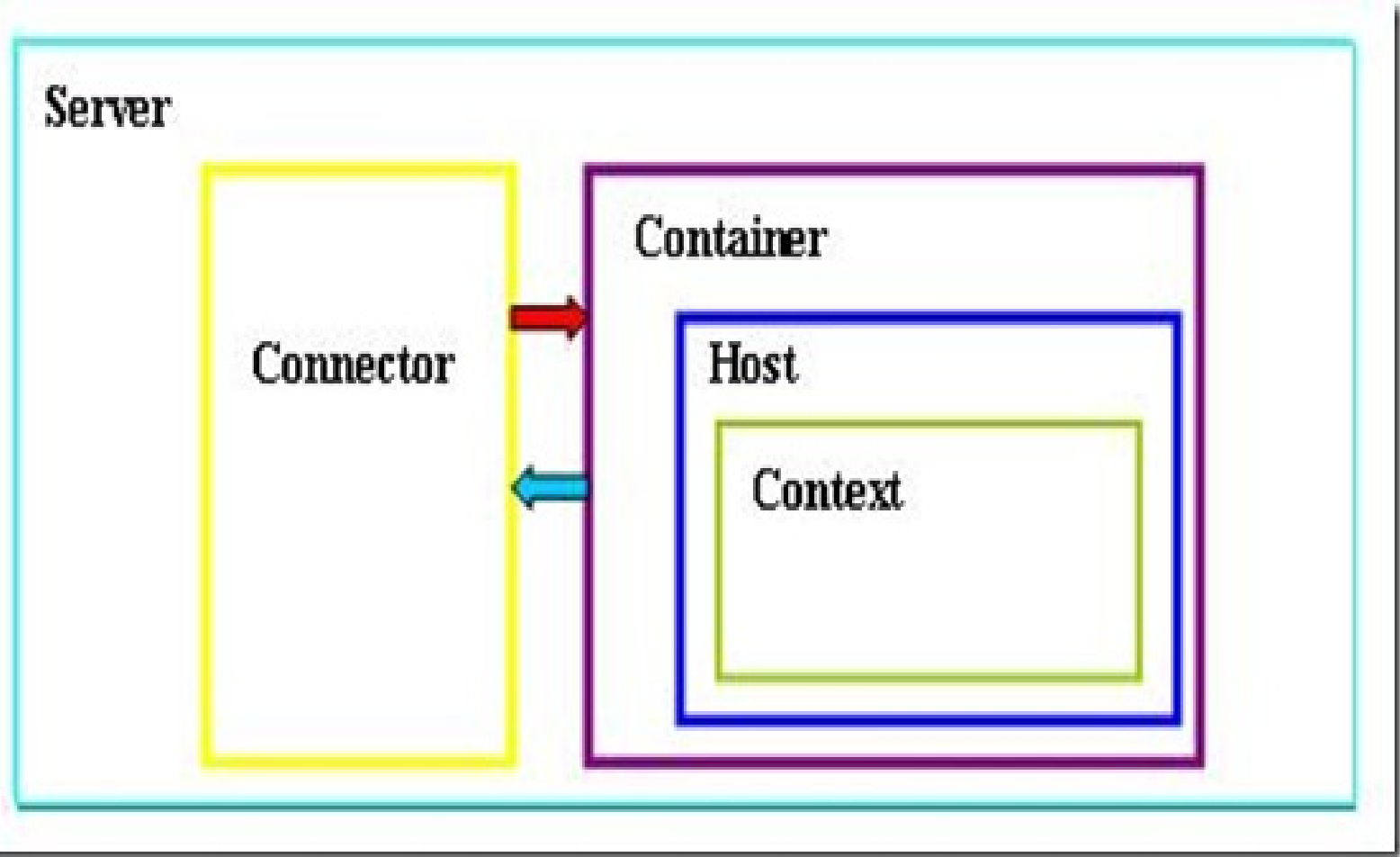
Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

抽象出 Host+Context

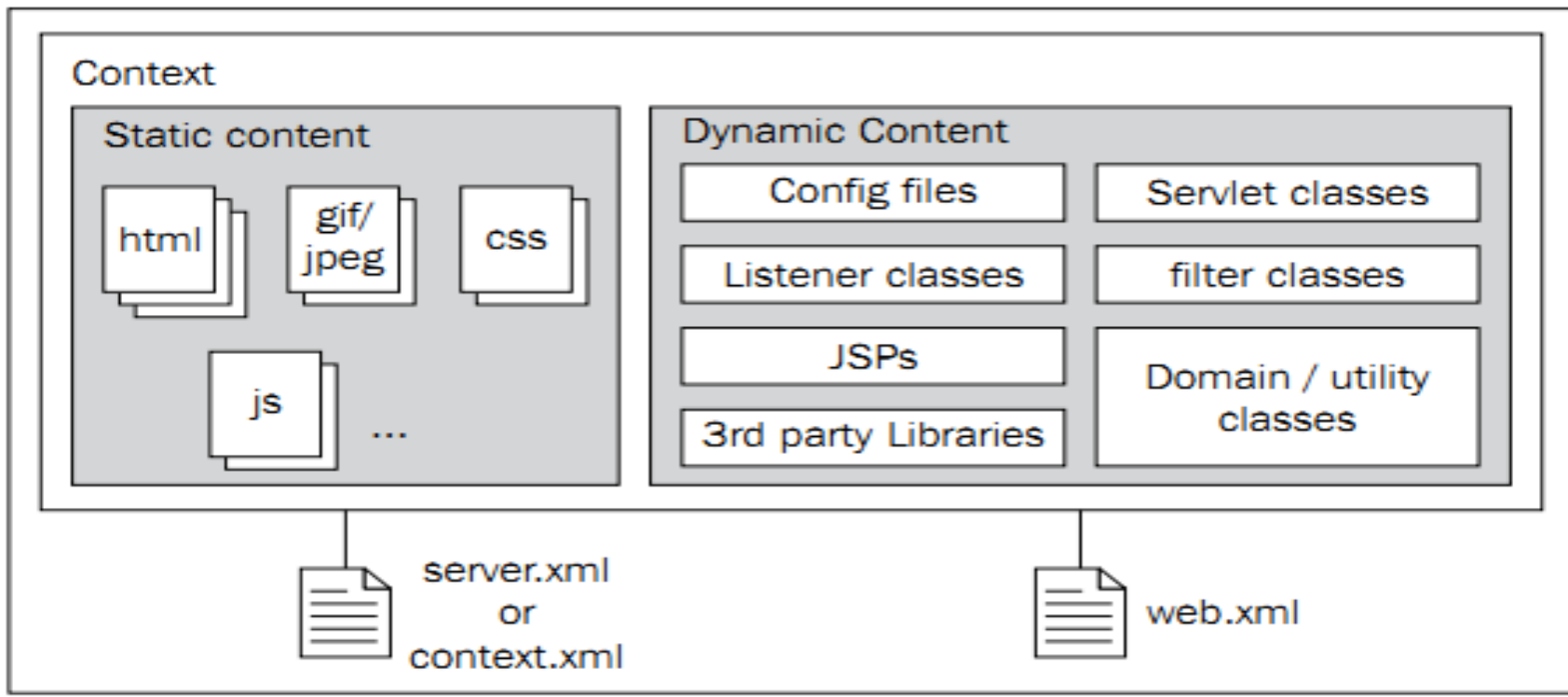


Host 中定位 Web 程序

- 浏览器发出请求
- Connector 接受请求，将请求交由 Container 处理，
- Container 查找请求对应的 Host 并将请求传递给它，
- Host 拿到请求后查找相应的应用上下文环境，
- 准备 servlet 环境并调用 service 方法。

Context（我们的 web 程序）

- Context
 - conf/server.xml
 - Context 片段
 - conf/<EngineName>/<HostName>
 - <ContextPath>.xml

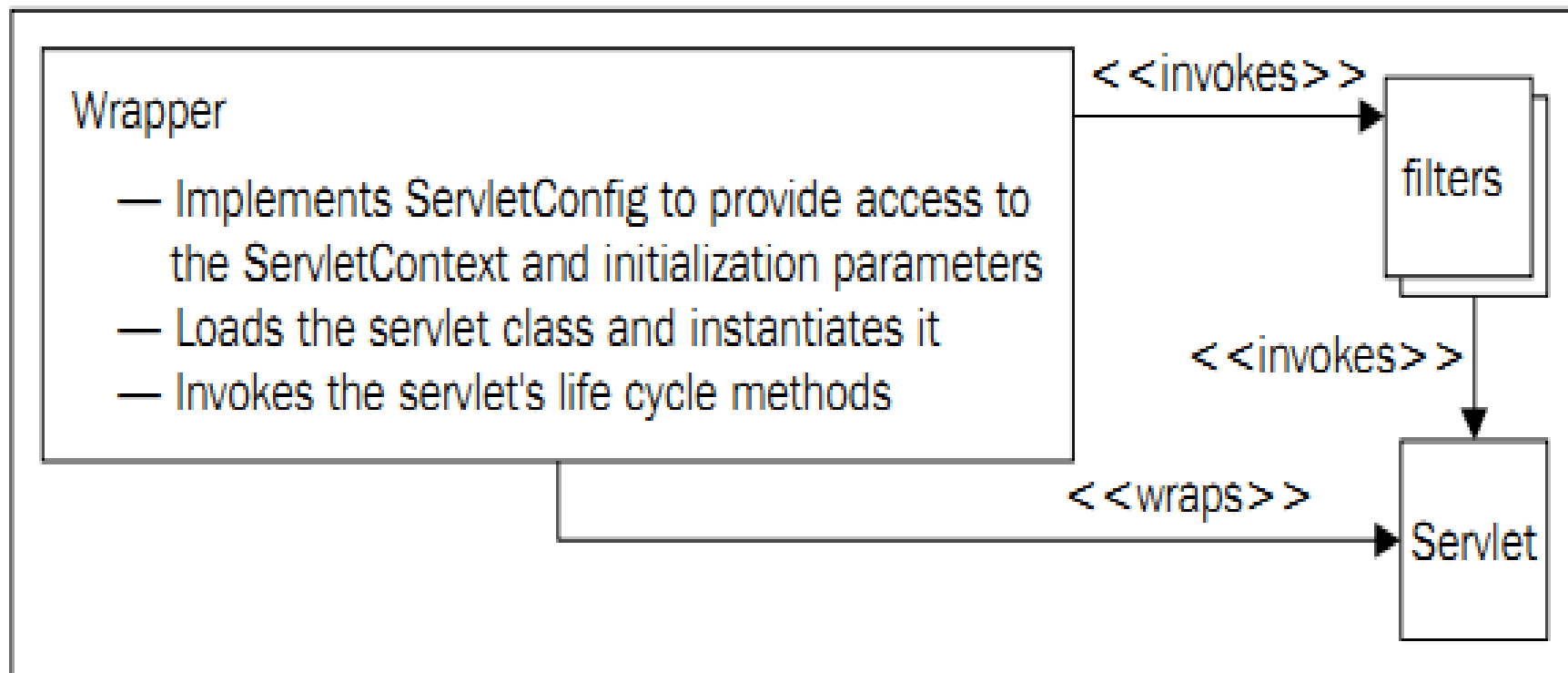


Context 配置

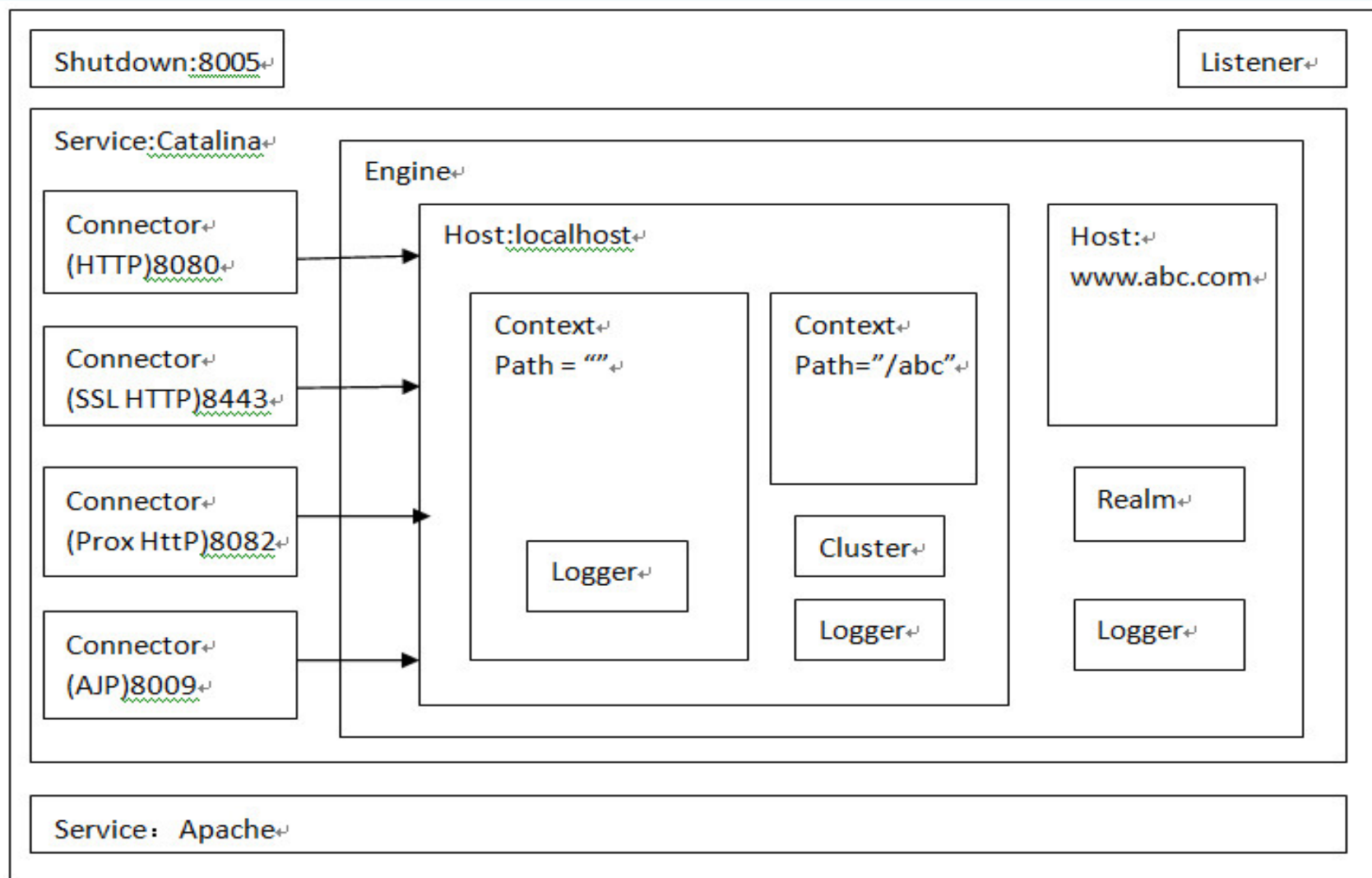
- `<Context crossContext=" false " docBase="E:/tomcatApp/CMS" path="/" reloadable="false" >`
`</Context>`
- **crossContext** 是否允许应用程序交互调用 `ServletContext.getContext()`
- **reloadable** 是否监测和加载 `/WEB-INF/classes/` 和 `/WEB-INF/lib` 中的类文件的更新。

Wrapper(包装 Servlet)

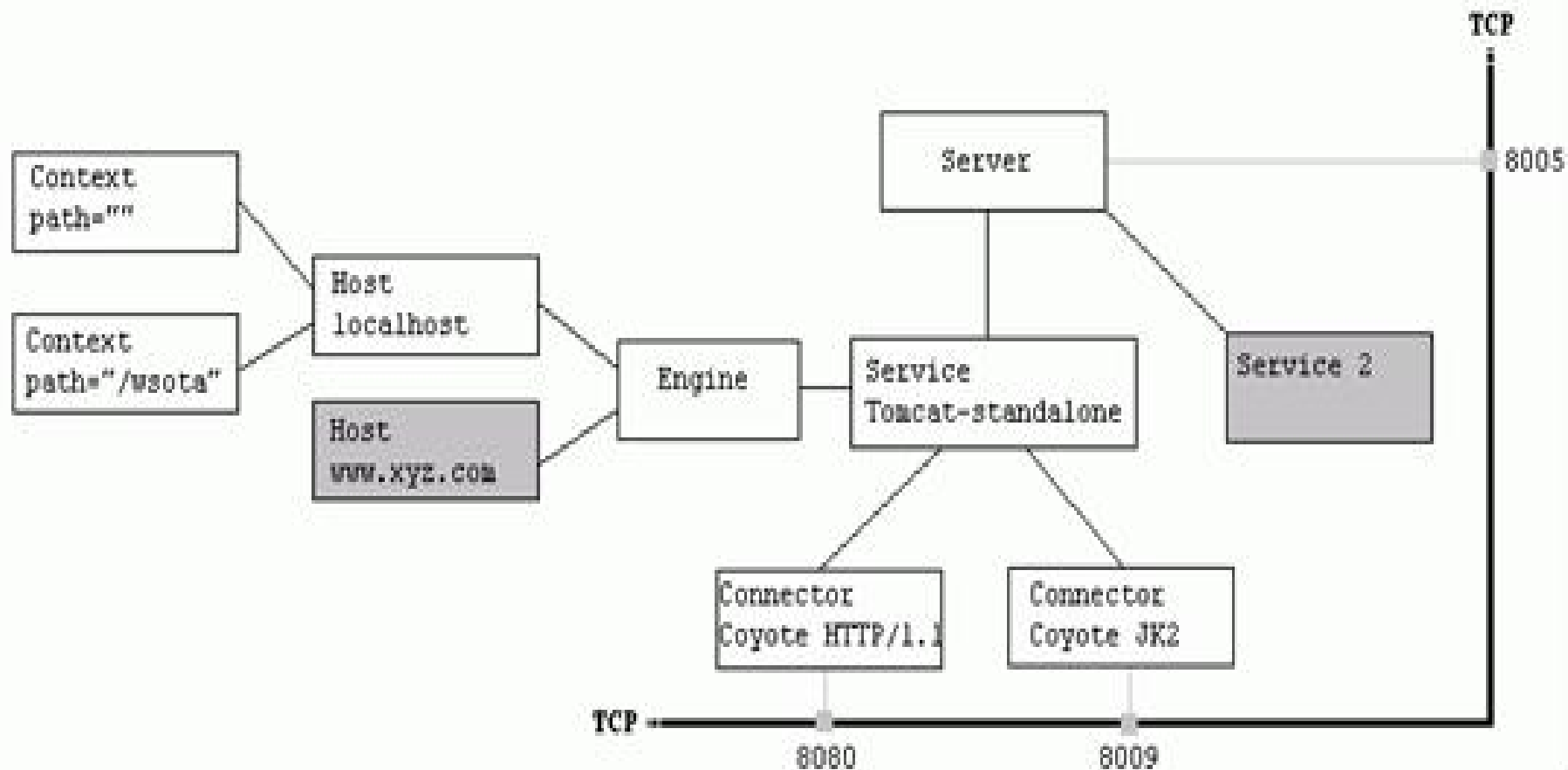
- 包装 javax.servlet.Servlet 实例



容器（含组件）结构

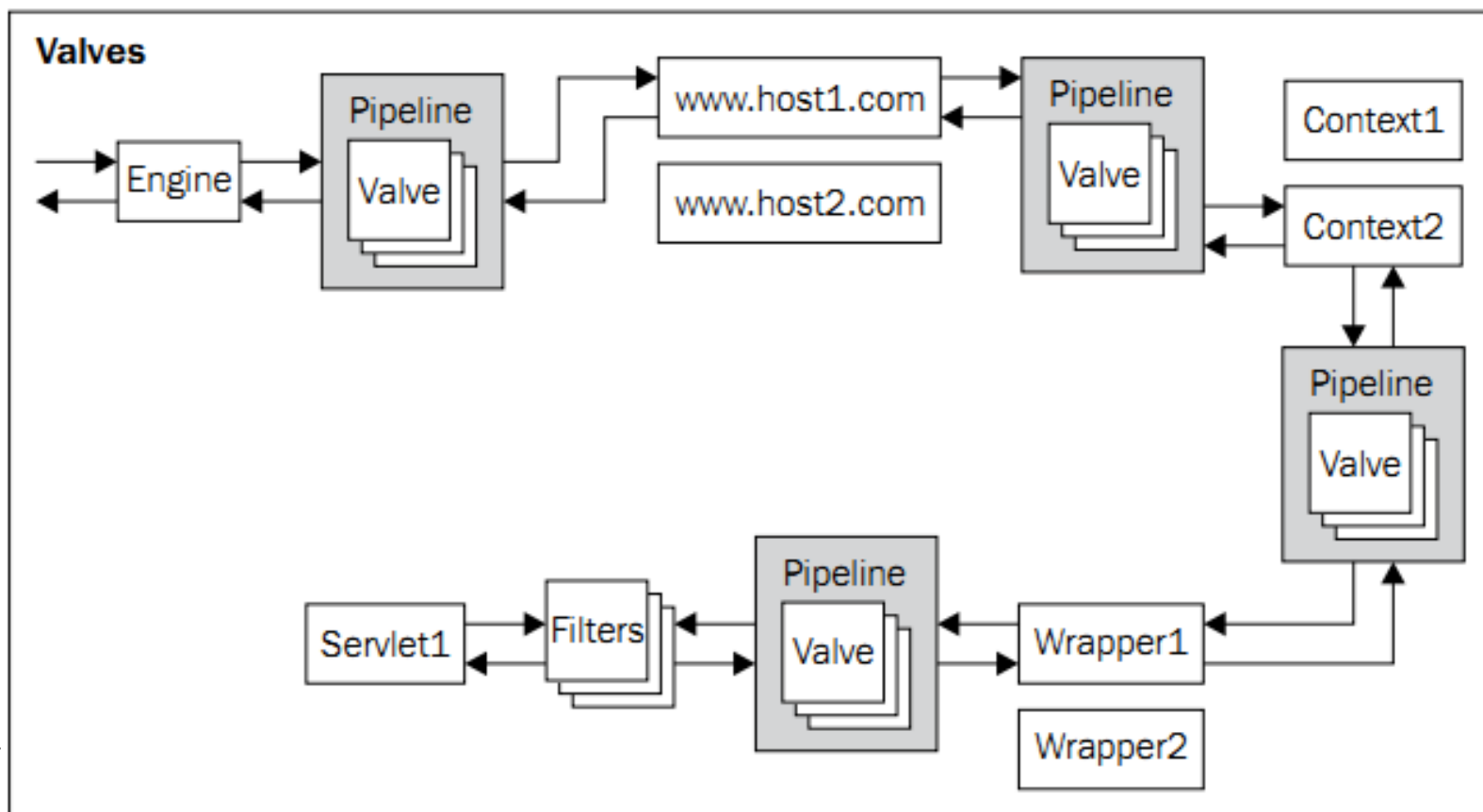


请求处理容器结构

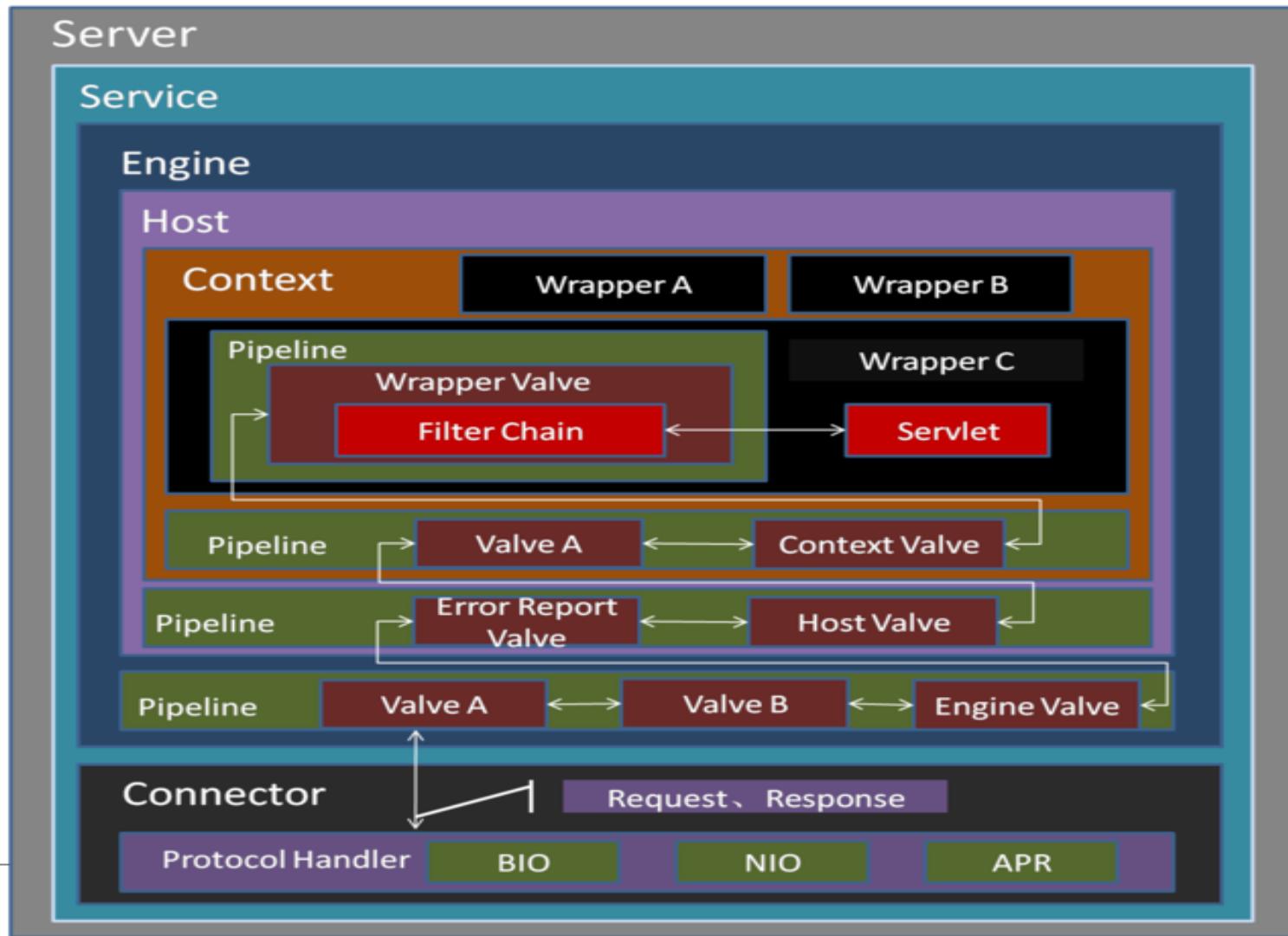


Valve

- 每一个容器的末端都有一个固定的阀
- 在末尾阀之前可以插入任意数量的自定义阀
- 末尾的作用是把管道带入下一个容器



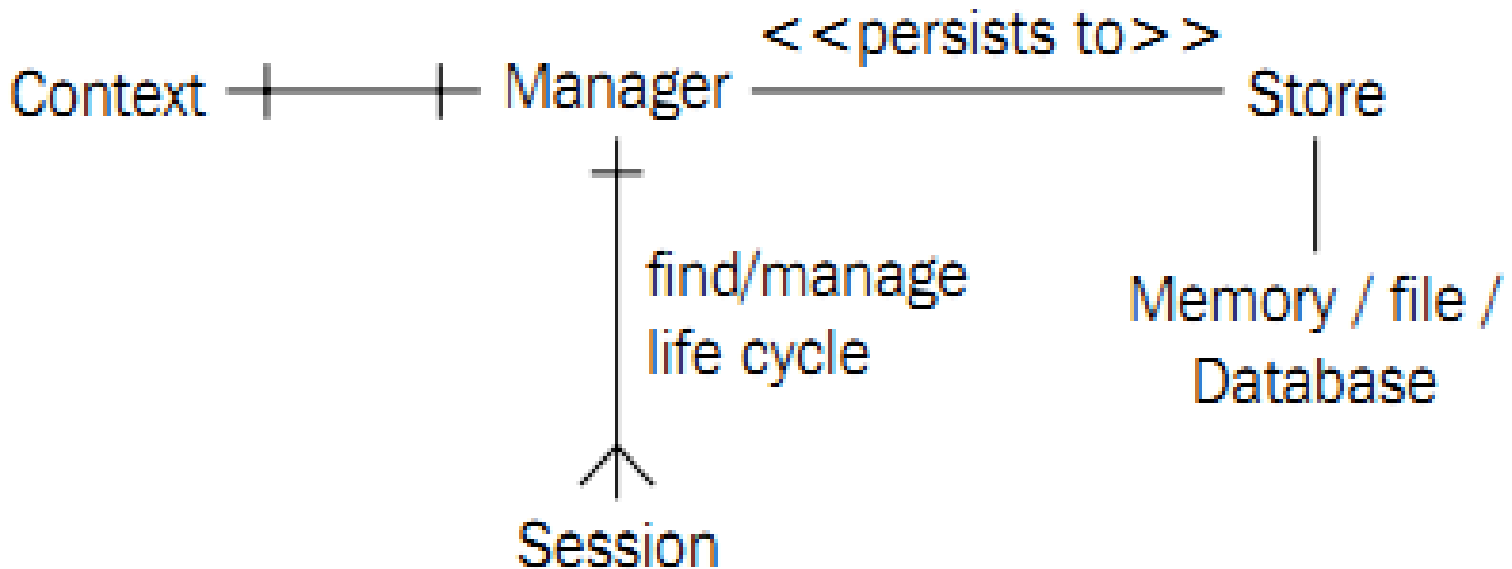
请求处理 Valve 执行流程



Manager 管理 Session

- 管理 Session
 - 创建、查找、存储、销毁

Session Manager



Session

- Cookie

- Cookie 中存放 jsessionid , 标识会话 , 变有状态
- URL 重写

- Session

- 后台线程扫描检测是否过期

```
Host: www.swengsol.com
```

```
Content-Length: 33
```

```
Connection: Keep-Alive
```

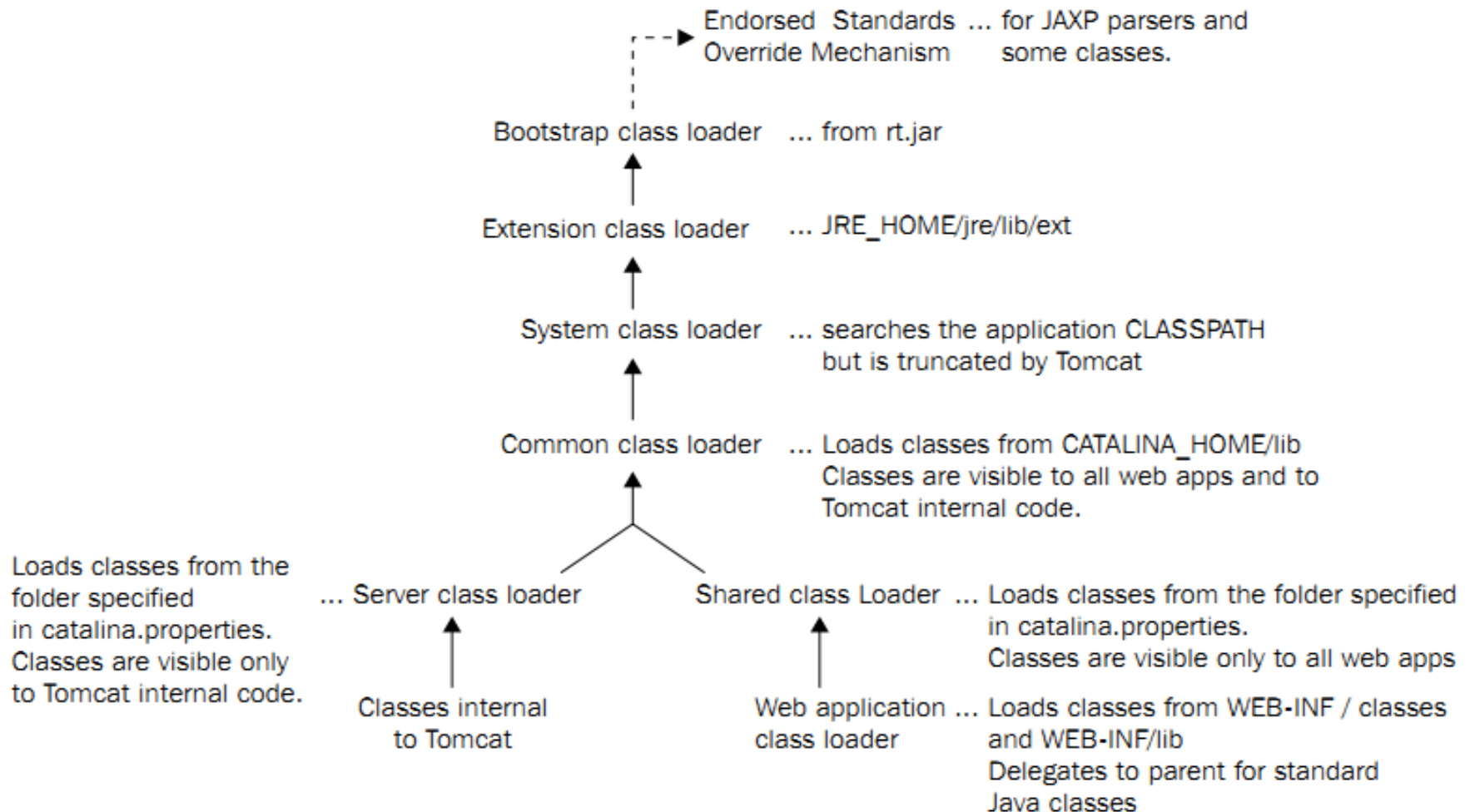
```
Pragma: no-cache
```

```
Cookie: JSESSIONID=2F8D401C1D1D192981DE4FE7B0D82601
```

```
— firstName=Damodar&lastName=Chetty
```

Loader

• 加载类、资源



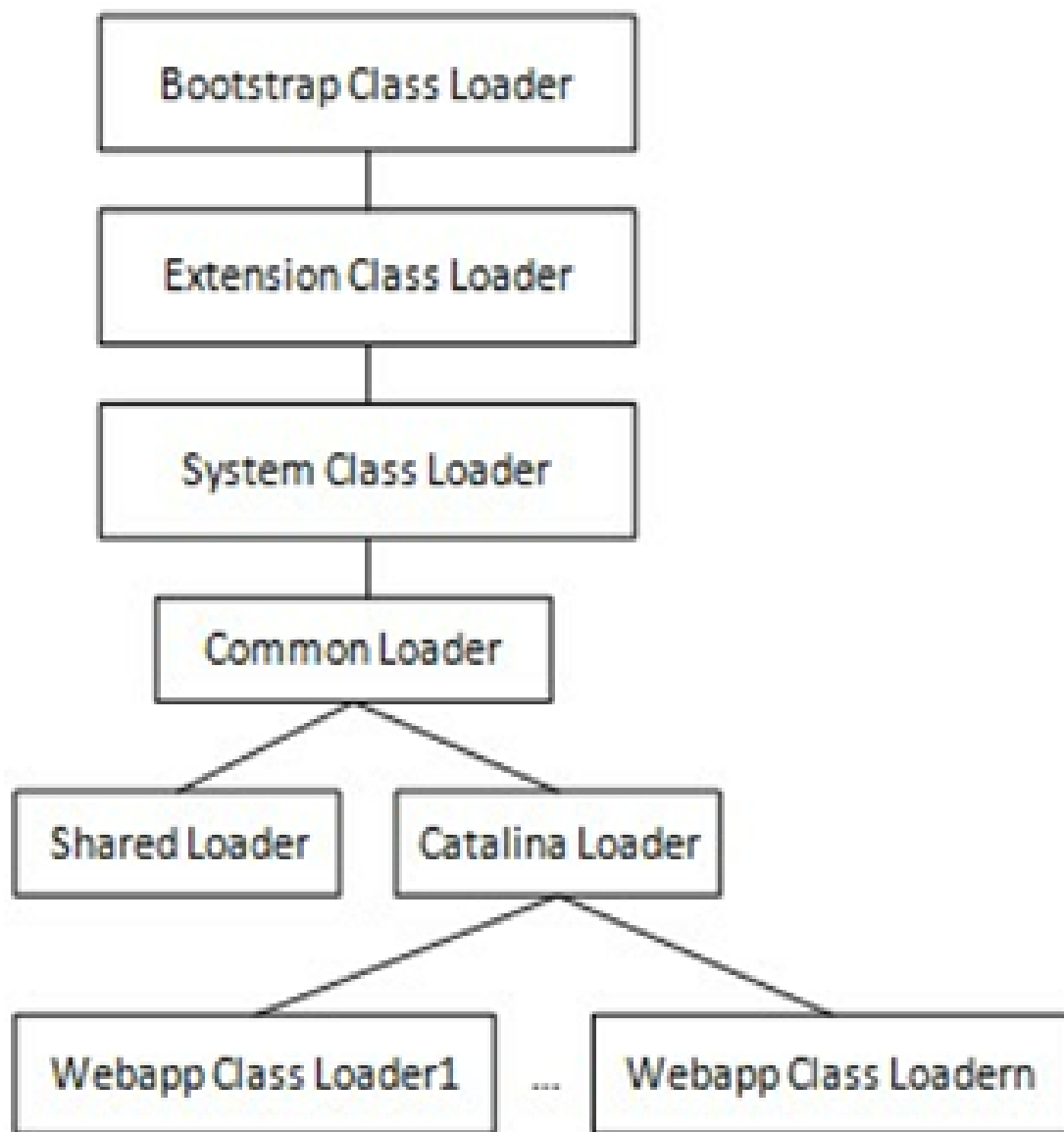
目录结构

- **bin** 启动和关闭 Tomcat 的脚本文件
- ~~common~~ 面向 Tomcat 和所有 web 应用
- ~~shared~~ 面向所有 web 应用非 Tomcat
- ~~server~~ 存放 tomcat 应用和 jar, 非面向 web 应用
- **work** jsp 生成的 servlet 文件
- **temp** Tomcat 运行时候存放临时文件
- **logs** 存放 Tomcat 的日志文件
- **conf** 各种配置文件

Tomcat5.X 目录结构

名称

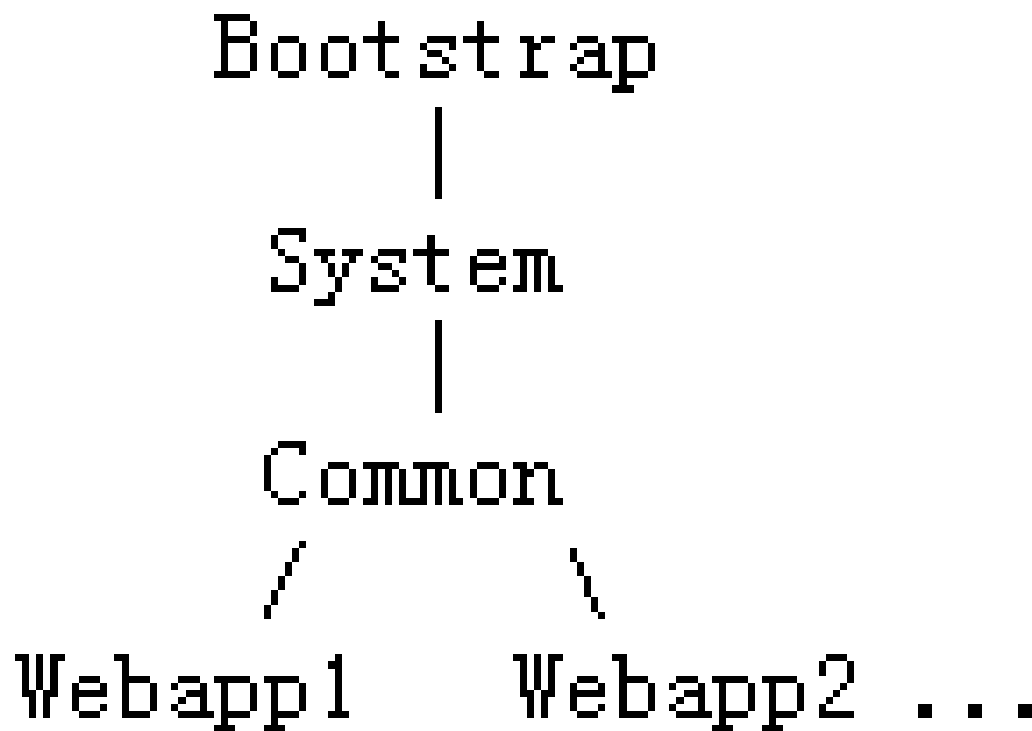
- bin
- common
- conf
- logs
- server
- shared
- temp
- webapps
- work
- LICENSE
- NOTICE
- RELEASE-NOTES
- RUNNING.txt



Tomcat 目录结构

名称

- bin
- conf
- lib
- logs
- temp
- webapps
- work
- LICENSE
- NOTICE
- RELEASE-NOTES
- RUNNING.txt

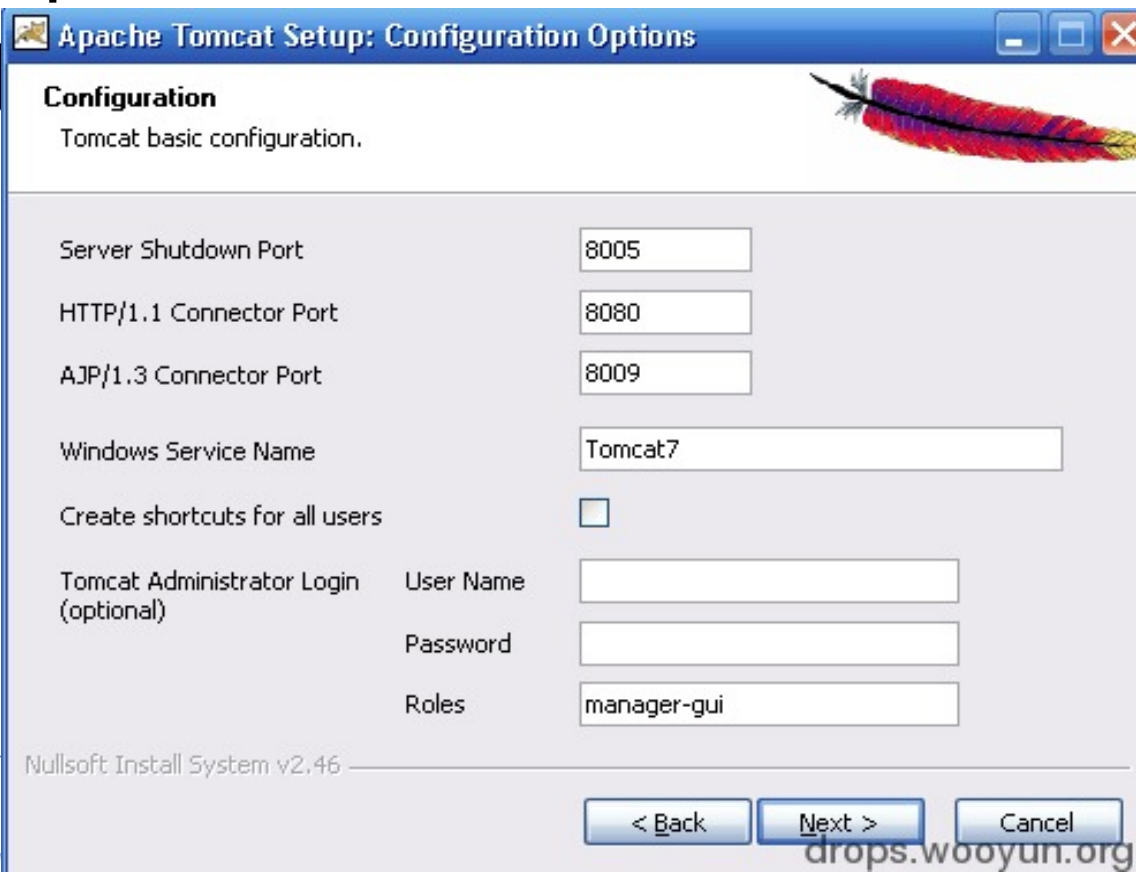


启动顺序

1. 设置 Tomcat 相关的环境变量
 - (CATALINA_BASE, CATALINA_HOME)
2. 初始化加载器
3. 初始化 Tomcat 实例
4. 为 Tomcat 实例解析配置文件 (server.xml)
 - 对里面的每个容器和组件进行实例化
5. 启动顶级容器 (Server) 实例，调 start 方法
6. 安装关闭回调钩子

启动关闭端口

- 关闭，默认端口
 - 接受关闭字符串命令
- 启动，默认端口 8080
 - 接受 http 请求
- SSL :8443
- AJP:8009



Apache Tomcat Setup: Configuration Options

Configuration
Tomcat basic configuration.

Server Shutdown Port: 8005

HTTP/1.1 Connector Port: 8080

AJP/1.3 Connector Port: 8009

Windows Service Name: Tomcat7

Create shortcuts for all users: ☐

Tomcat Administrator Login (optional)

User Name:

Password:

Roles: manager-gui

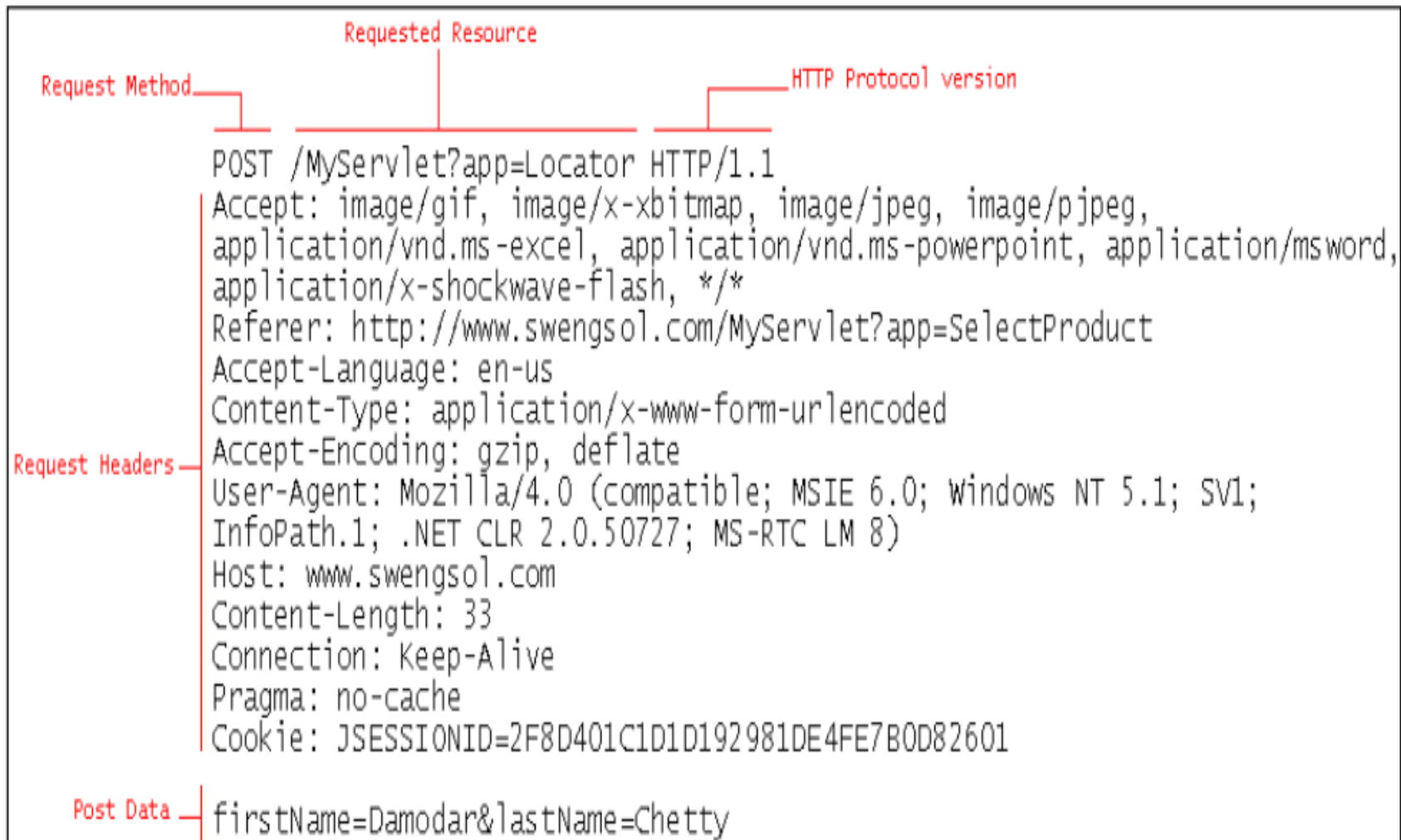
Nullsoft Install System v2.46

< Back Next > Cancel

Tomcat 设计模式总结

- 1, 容器模型
 - 使用**组合模式**抽象父子嵌套模型
- 2, 逻辑处理
 - **责任链模式** (使用管道阀门)
 - 设置一个固定的阀门放在每个容器的管道中的末端
 - 用户自定义的阀门可以随意放在非末端的位置
- 3, 生命周期
 - 观察者模式
- 4, 门面模式
 - 封装 Request 和 Response

看图说流程



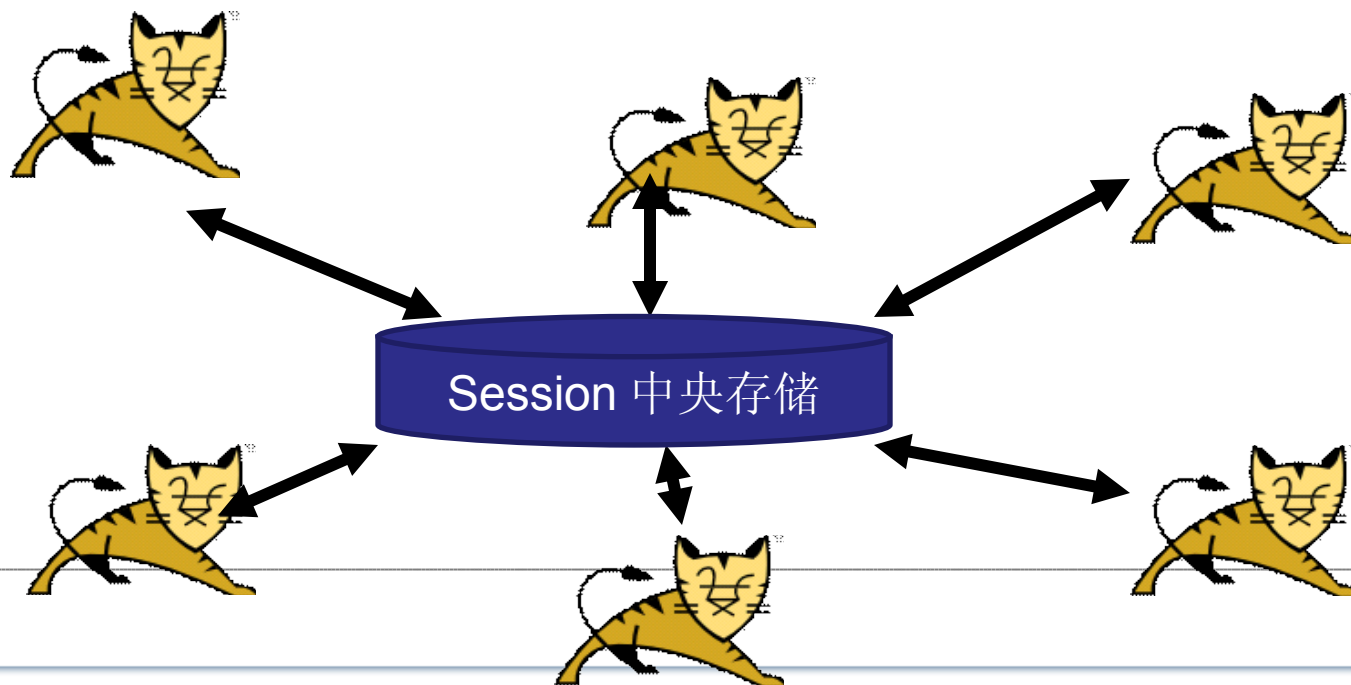
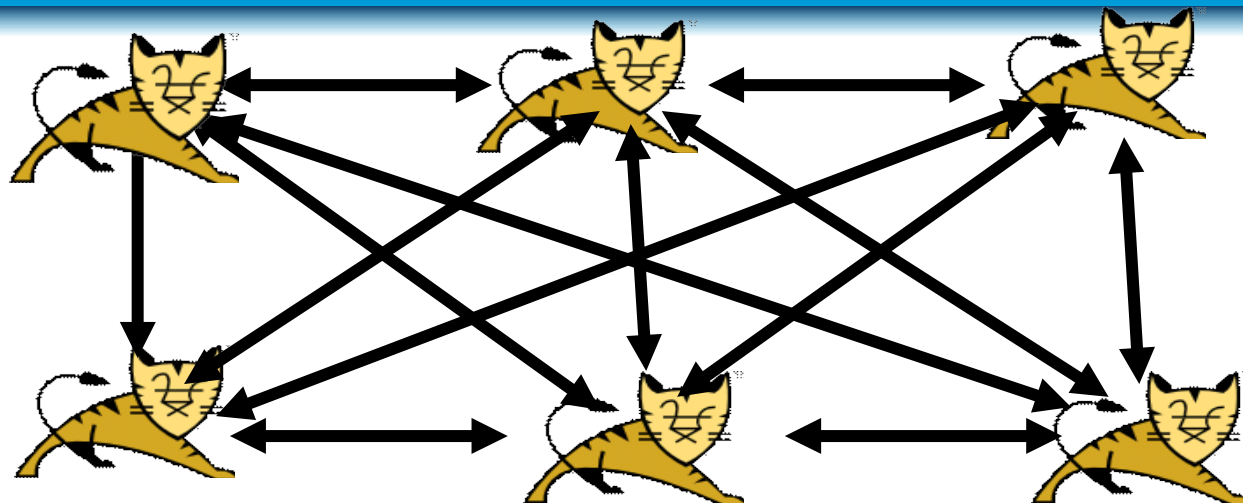
应用

- 集群与多节点
 - (1) Session/Cookie 支撑有 / 无状态
 - (2) 部署, 日志, 故障

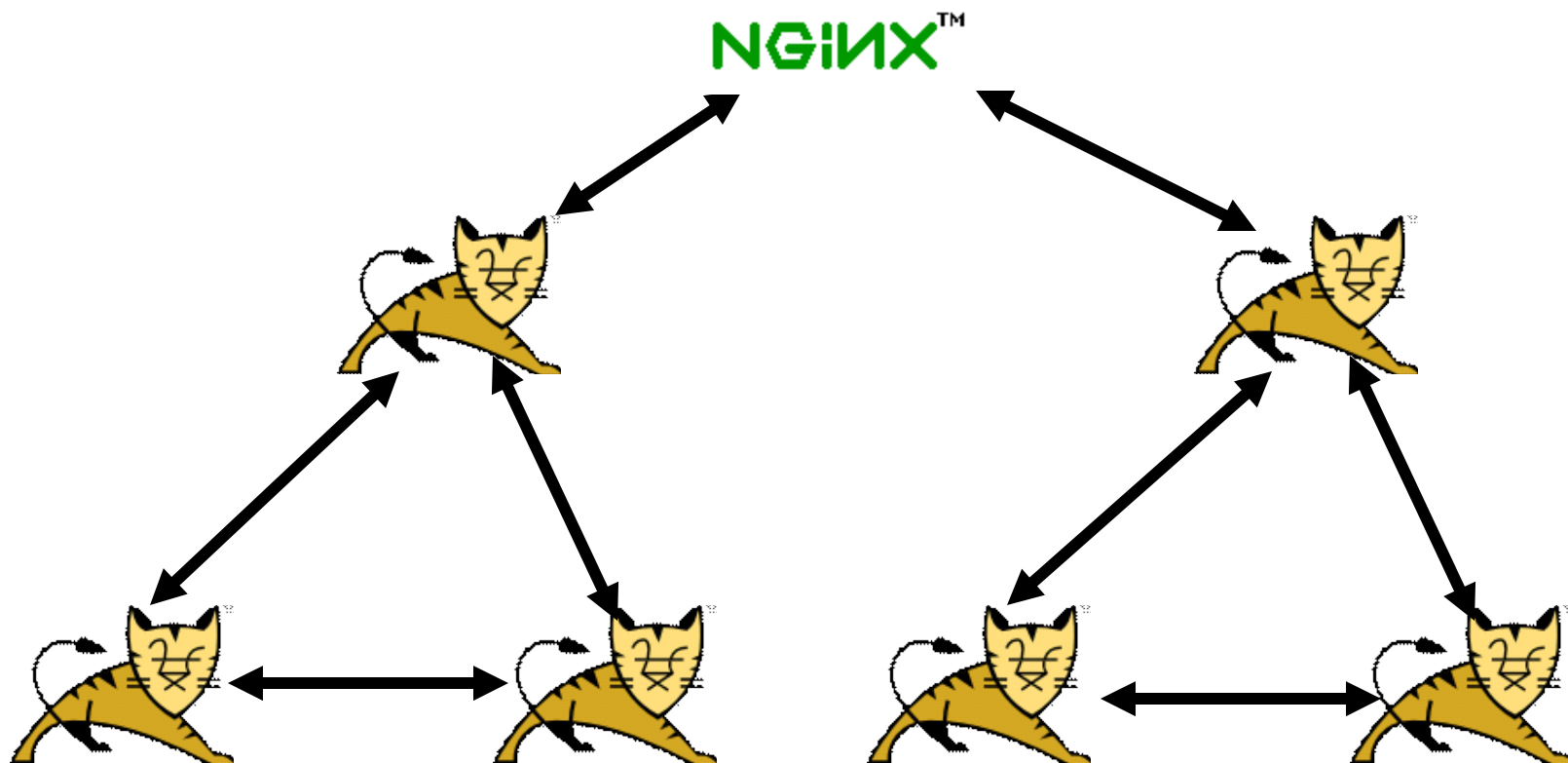
集群

- 使用 Tribe 通信框架进行通信
 - 心跳检测
 - Session 复制
- 缺点
 - 每个 server.xml 都要进行配置，扩展性差
 - 出现 session 复制风暴

Tomcat 集群结构



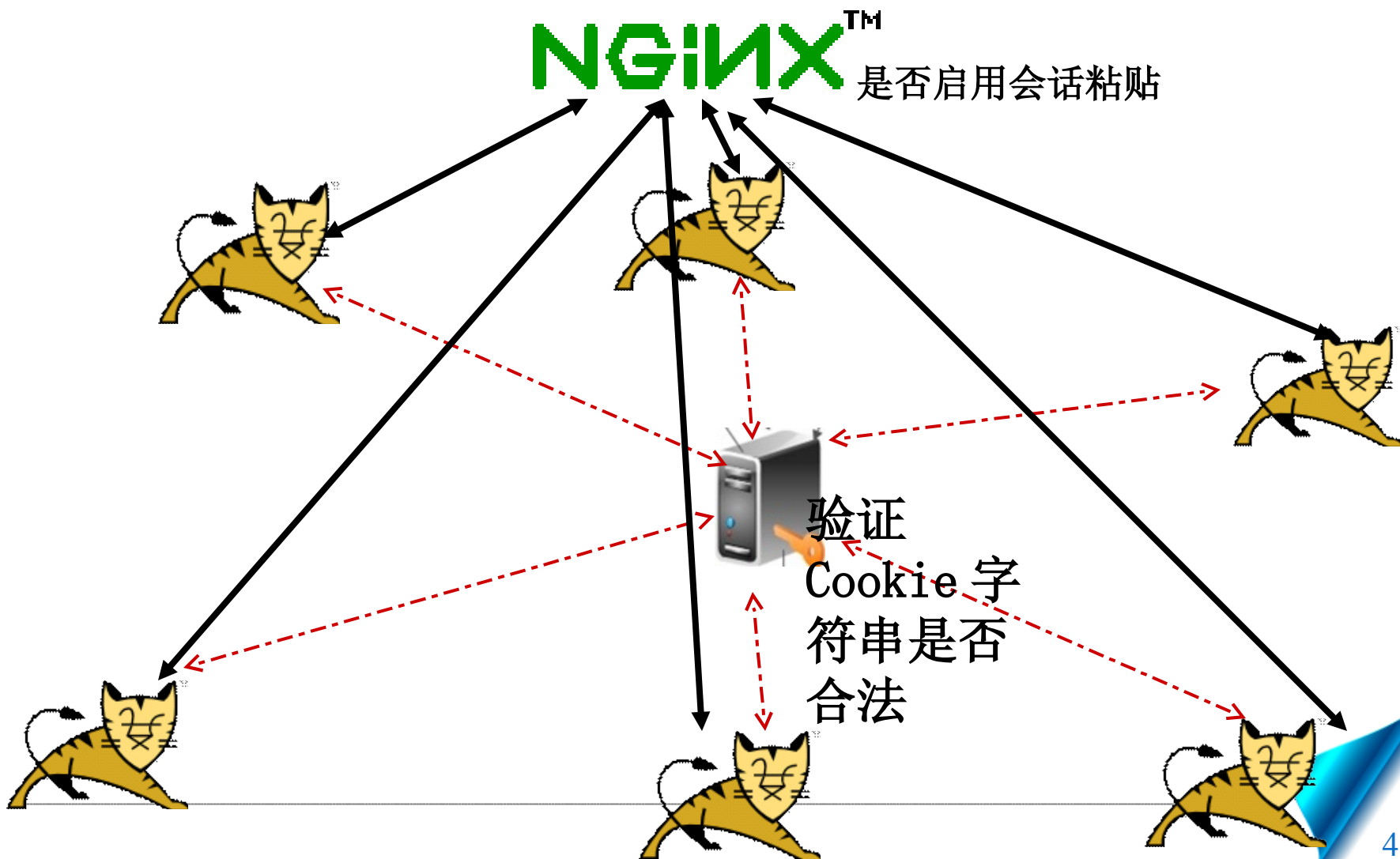
分组集群



多节点无状态应用（禁 session）

- 禁止 Session
 - jsp 页面中加上 `<%@ page session="false"%>`
 - session 过期时间为零
 - `<session-config>`
`<session-timeout>0</session-timeout>`
`</session-config>`

无状态多节点部署图



多节点无状态应用（用 cookie）

- 使用 Cookie
 - 使用 SSL
 - 使用某算法生成一个字符串
 - 例如：组合 IP, 当前时间戳，随即字符串，用户名
 - 该字符串为 Key 存到中央存储, Value 为 user 对象
 - 该字符串存储到用户浏览器的 Cookie 内
 - 请求访问时从 Cookie 获取字符串进行验证合法性

部署

- 保证一个 Tomcat 一个 web 应用
 - 防止交叉感染
 - 单个物理机器上可以启动多个 tomcat 实例
- 配置 Context.xml 中 web 应用的根目录
 - 关闭 reload 属性
 - 不放 webapp 下 ,tomcat 目录和 web 应用目录分开
- 使用 web server(Nginx) 处理静态资源
 - 静态资源可以单独部署到二级域名指向的服务器
 - 静态资源路径可以用 nginx upstream 方式处理

重新部署顺序

- 先备份当前 web 应用目录
 - 比如：
 - `mv /home/appName /home/appName_20140420`
- 把新 web 应用打包，比如 tar 格式
- 上传 tar 包到预发环节，预发环节解包
 - 循环每个 tomcat 节点
 - 删除其在 nginx 的配置信息
 - 使用命令：`nginx -s reload`，这样避免 nginx 保证可持续性（不报 502 错误）
 - 关闭 tomcat 并放新 web 应用，启动 tomcat
 - 配置其在 nginx 中节点信息

日志处理

- 分布式节点上收集日志
 - 使用工具去每个节点收集
 - Scribe : Facebook 开源的分布式日志搜集系统
 - 自定义过滤器写到某服务器目录
 - 路径配置成固定服务器的某目录
- 分析日志
 - 404
 - 502
 - Access Log
 - 用户行为
 - 在做某个业务之前访问过哪些页面 每个页面停留过多少时间

故障

- 提高吞吐量
 - 配置 nio(相关连接参数) 和 thread pool
 - Server.xml:
 - Executor name= “tomcatThreadPool ” 可以被多个连接器共享
- 数据库连接池泄露
- Windows 平台下
 - 访问慢
 - 调整启动参数 , 启动脚本中加入 JAVA_OPTS 参数
 - 使用 web server, 安装 APR, bin 放 dll 文件 , *-windows-x86.zip
 - 经常宕机
 - 快速解决 : 重启
- 内存溢出
 - Eclipse Memory Analyze
 - jmap -heap PID
 - 调整启动参数 , 在内存溢出时候打印出 log
 - XX:+HeapDumpOnOutOfMemoryError
 - XX:HeapDumpPath= ./java_pid<pid>

新特性

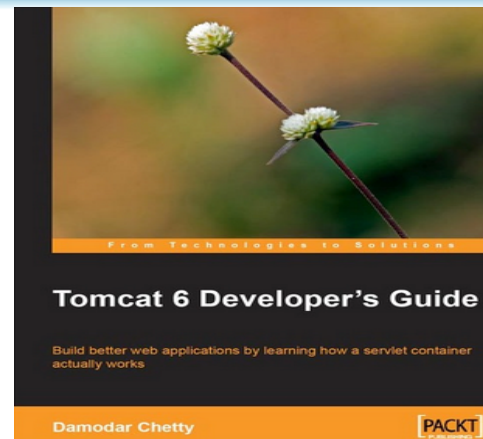
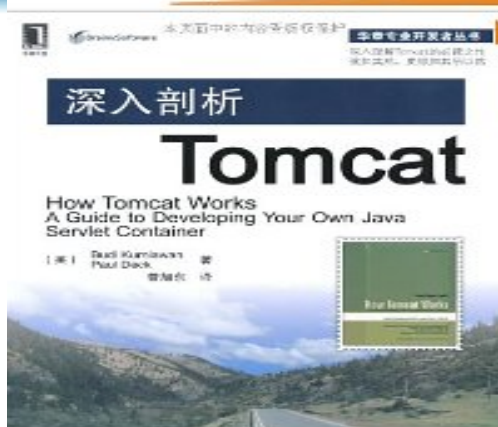
- Tomcat JDBC Connection Pool
 - 拦截器
 - 慢查询统计
- Servlet 3.0
 - 注释
 - 异步
- WebSocket
 - 演示画板程序

低调调戏下老猫

- 1 , 多个 Service
- 2 , 多个连接器
- 3 , 偷窥内部
 - 远程监控需在 tomcat 启动加 jmx 参数
- 4 , Kill !



参考资料



- 深入理解各 JEE 服务器 Web 层集群原理
- How Tomcat Works(深入剖析 Tomcat)
- Tomcat 6 Developer's Guide – Build better web applications by learning how a servletcontainer actually works (Packt, 2009)

■ 谢谢大家！



李良召 (John, 和风赛跑)

iamjohnli@gmail.com

<http://blackhouseapp.com>

2014-04-20 济南明月大酒店