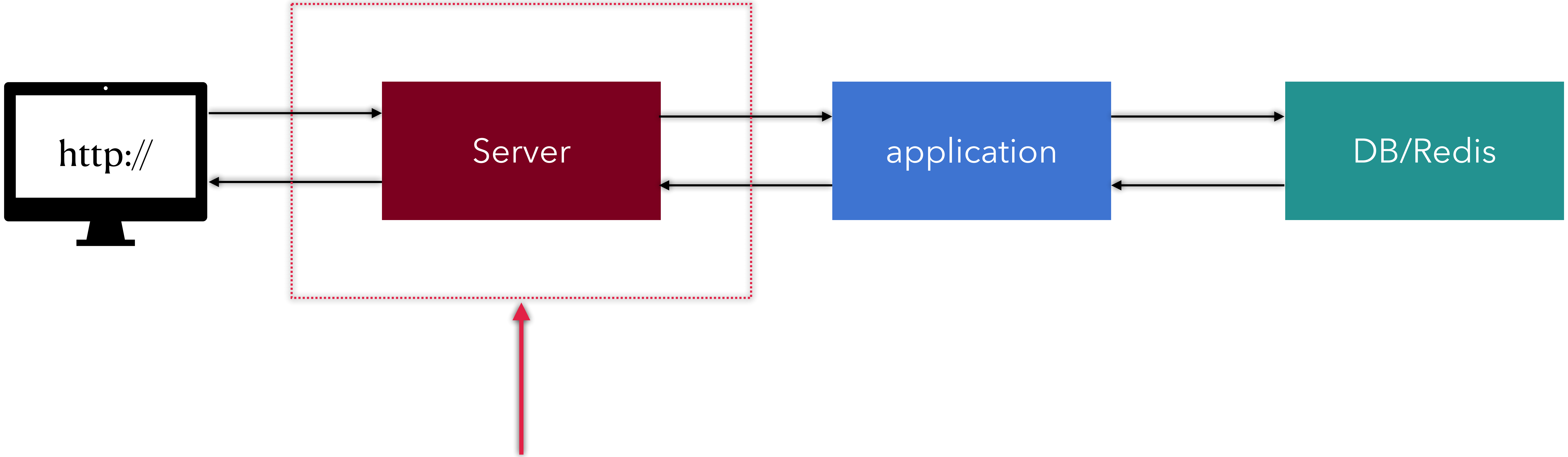
一次HTTP请求在Tomcat中的流转过程

主题



需求

1. 支持Servlet+JSP规范
2. 支持WebServer
3. 开源
4. 易扩展
5. 高性能
6. 自定义/被替换

Demo

```
/**
 * Demo版服务器处理逻辑:
 * 1. 启动服务器监听端口,如8080
 * 2. 开始监听客户端请求,如http请求
 * 3. 解析http请求参数
 * 4. 业务自己逻辑处理,如查询DB并返回数据
 * 5. 将返回数据按http协议格式返回
 */
private void process() throws Exception {
    // 1. 服务端启动8080端口，并一直监听；
    ServerSocket ss = new ServerSocket(8080);

    // 2. 监听到有客户端（比如浏览器）要请求http://localhost:8080/，那么建立连接，TCP三次握手；
    Socket socket = ss.accept();
    InputStream is = socket.getInputStream();
    OutputStream os = socket.getOutputStream();
    BufferedReader reader = new BufferedReader(new InputStreamReader(is));

    // 3. 建立连接后，读取此次连接客户端传来的内容（其实就是解析网络字节流并按HTTP协议去解析）；
    // GET /app/user HTTP/1.1
    String requestLine = reader.readLine();
    Logs.SERVER.info("requestLine is : {}", requestLine);
    if (requestLine == null || requestLine.length() < 1) {
        Logs.SERVER.error("could not read request");
        return;
    }
    String[] tokens = requestLine.split(" ");
    String method = tokens[0];
    String urlPath = tokens[1];

    // 4. 业务逻辑：组装给客户端的返回数据,如查DB
    String content = "服务端返回数据...";

    // 5. 找到资源后，再通过网络流将内容输出，当然，还是按照HTTP协议去输出，这样客户端（浏览器）就能正常渲染、显示网页内容；
    BufferedOutputStream bos = new BufferedOutputStream(os);
    int len = content.length();
    byte[] headerBytes = createHeaderBytes("HTTP/1.1 200 OK", len, "application/json;charset=utf-8");
    bos.write(headerBytes);
    byte[] buf = new byte[2000];
    bos.write(buf, 0, buf.length);
    bos.flush();
    socket.close();
}
```

Demo逻辑

Demo版服务器处理逻辑:

1. 启动服务器监听端口,如8080

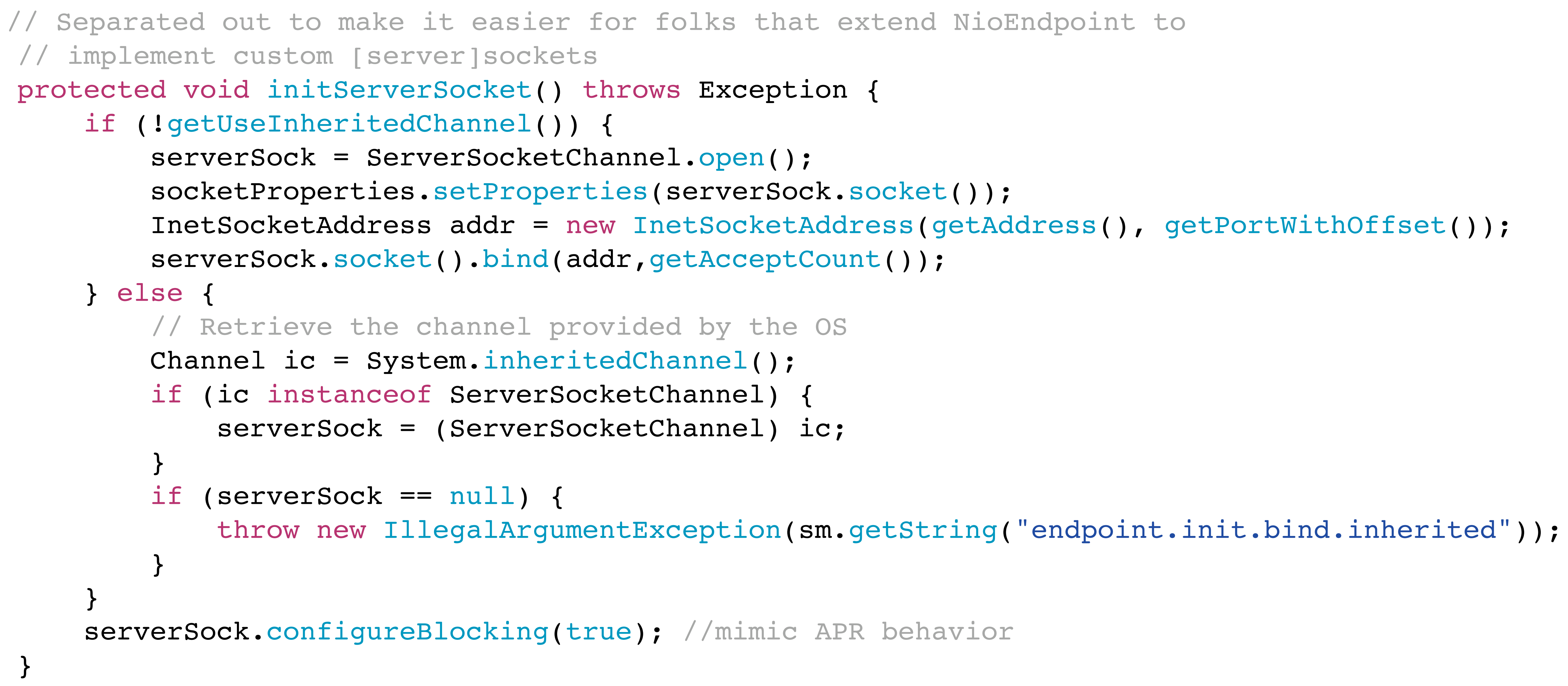
2. 开始监听客户端请求,如http请求

3. 解析http请求参数

4. 业务自己逻辑处理,如查询DB并返回数据

5. 将返回数据按http协议格式返回

1.启动端口



```java
// Separated out to make it easier for folks that extend NioEndpoint to
// implement custom [server]sockets
protected void initServerSocket() throws Exception {
    if (!getUseInheritedChannel()) {
        serverSock = ServerSocketChannel.open();
        socketProperties.setProperties(serverSock.socket());
        InetSocketAddress addr = new InetSocketAddress(getAddress(), getPortWithOffset());
        serverSock.socket().bind(addr,getAcceptCount());
    } else {
        // Retrieve the channel provided by the OS
        Channel ic = System.inheritedChannel();
        if (ic instanceof ServerSocketChannel) {
            serverSock = (ServerSocketChannel) ic;
        }
        if (serverSock == null) {
            throw new IllegalArgumentException(sm.getString("endpoint.init.bind.inherited"));
        }
    }
    serverSock.configureBlocking(true); //mimic APR behavior
}
```

2.监听请求

```java
protected SocketChannel serverSocketAccept() throws Exception {
    return serverSock.accept();
}
```

3.处理请求

```java
// ---------------------------------------------- Request processing methods

    /**
     * Process the given SocketWrapper with the given status. Used to trigger
     * processing as if the Poller (for those endpoints that have one)
     * selected the socket.
     *
     * @param socketWrapper The socket wrapper to process
     * @param event         The socket event to be processed
     * @param dispatch      Should the processing be performed on a new
     *                          container thread
     *
     * @return if processing was triggered successfully
     */
    public boolean processSocket(SocketWrapperBase<S> socketWrapper,
            SocketEvent event, boolean dispatch) {
        try {
            if (socketWrapper == null) {
                return false;
            }
            SocketProcessorBase<S> sc = null;
            if (processorCache != null) {
                sc = processorCache.pop();
            }
            if (sc == null) {
                sc = createSocketProcessor(socketWrapper, event);
            } else {
                sc.reset(socketWrapper, event);
            }
            Executor executor = getExecutor();
            if (dispatch && executor != null) {
                executor.execute(sc);
            } else {
                sc.run();
            }
        } catch (RejectedExecutionException ree) {
            getLog().warn(sm.getString("endpoint.executor.fail", socketWrapper) , ree);
            return false;
        } catch (Throwable t) {
            ExceptionUtils.handleThrowable(t);
            // This means we got an OOM or similar creating a thread, or that
            // the pool and its queue are full
            getLog().error(sm.getString("endpoint.process.fail"), t);
            return false;
        }
        return true;
    }
```

4.业务逻辑

```java
public class HelloWorldExample extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        ResourceBundle rb =
            ResourceBundle.getBundle("LocalStrings",request.getLocale());
        response.setContentType("text/html");
        response.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<!DOCTYPE html><html>");
        out.println("<head>");
        out.println("<meta charset=\"UTF-8\" />");

        String title = rb.getString("helloworld.title");

        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body bgcolor=\"white\">");

        out.println("<a href=\"../helloworld.html\">");
        out.println("<img src=\"../images/code.gif\" height=24 " +
                    "width=24 align=right border=0 alt=\"view code\"></a>");
        out.println("<a href=\"../index.html\">");
        out.println("<img src=\"../images/return.gif\" height=24 " +
                    "width=24 align=right border=0 alt=\"return\"></a>");
        out.println("<h1>" + title + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

5.返回结果

```
/**
 * Perform whatever actions are required to flush and close the output
 * stream or writer, in a single operation.
 *
 * @exception IOException if an input/output error occurs
 */
public void finishResponse() throws IOException {
    // Writing leftover bytes
    outputBuffer.close();
}
```

功能拆分

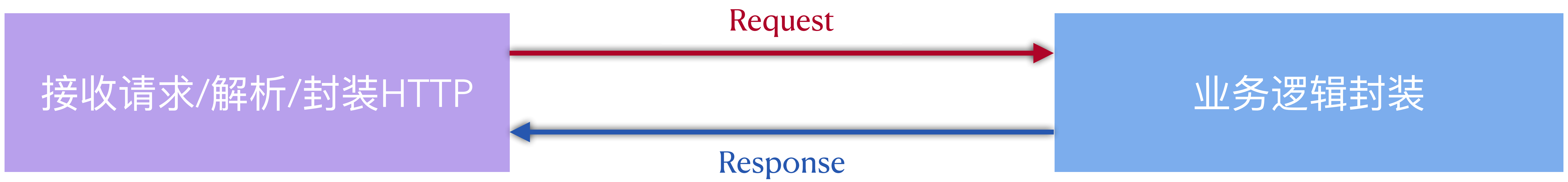
1. 启动服务器监听端口,如8080

2. 开始监听客户端请求,如http请求

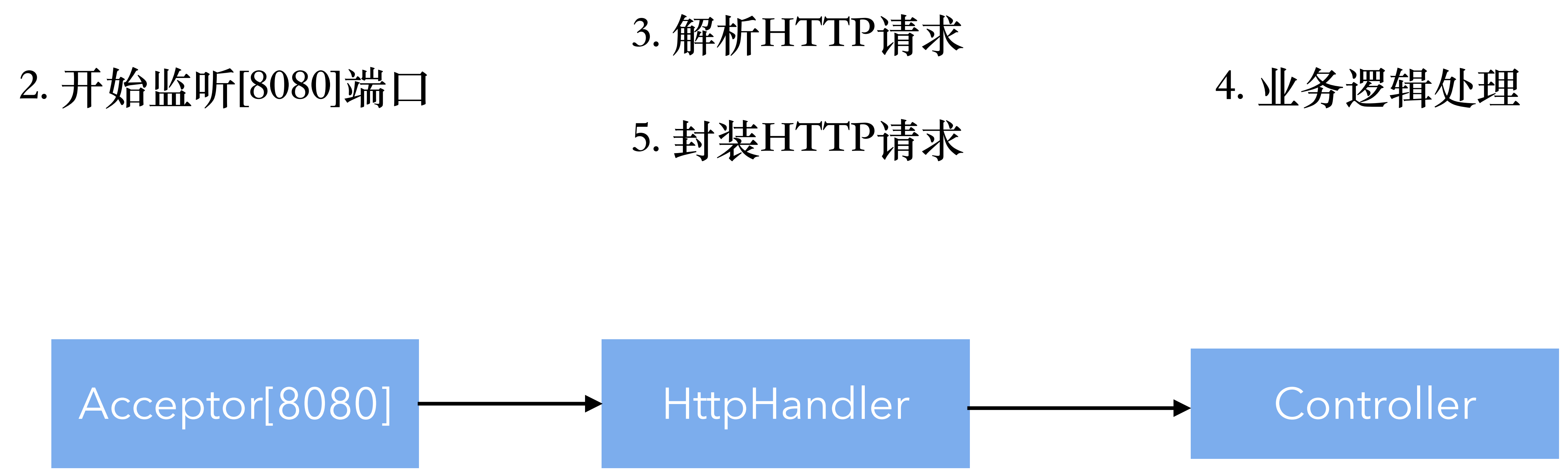
3. 解析http请求参数

5. 将返回数据按http协议格式返回

4. 业务自己逻辑处理,如查询DB并返回数据



功能再拆分



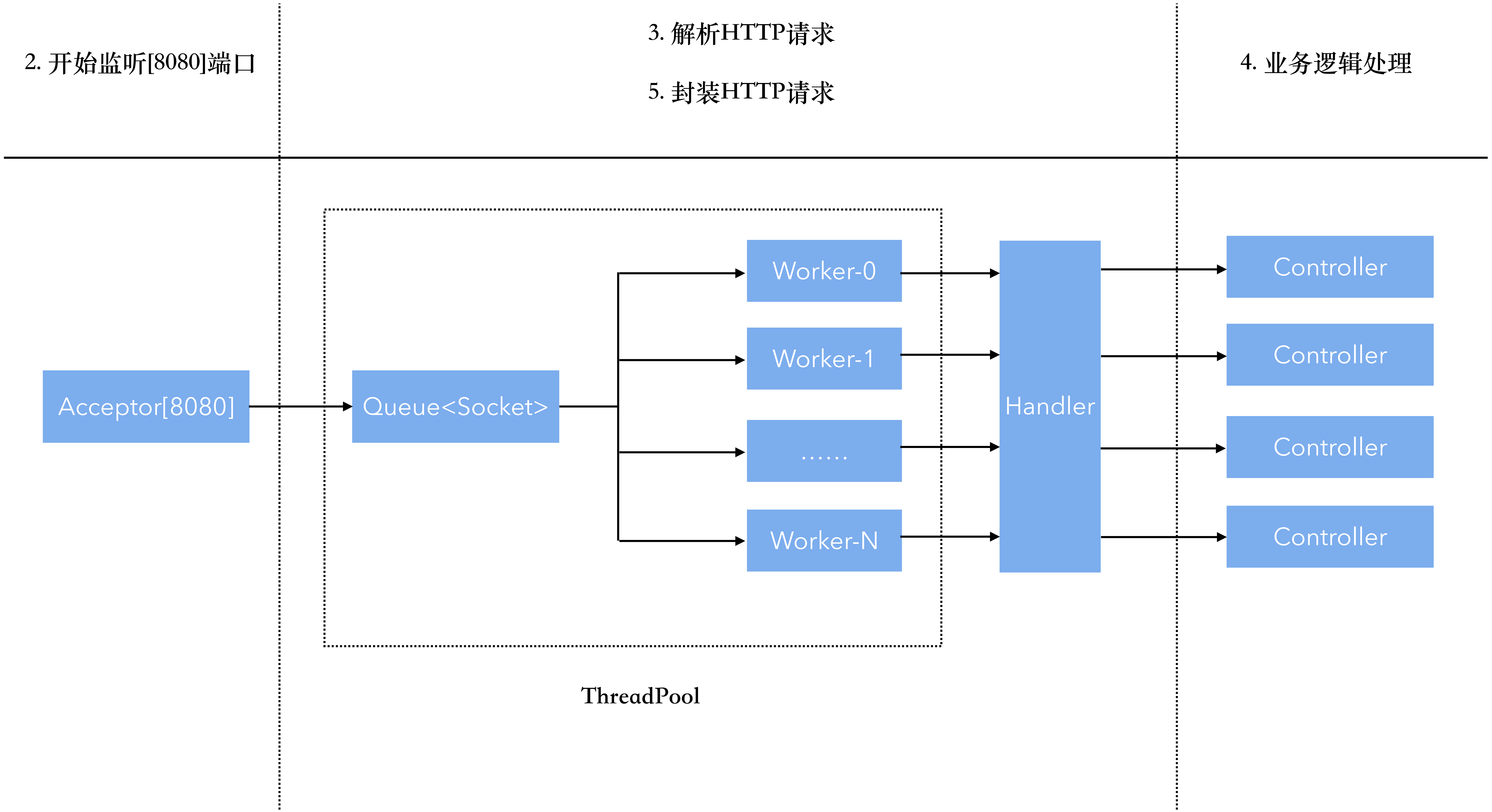
2. 开始监听[8080]端口

3. 解析HTTP请求

5. 封装HTTP请求

4. 业务逻辑处理

性能



2. 开始监听[8080]端口

3. 解析HTTP请求

5. 封装HTTP请求

4. 业务逻辑处理

Acceptor[8080]

Queue<Socket>

Worker-0

Worker-1

……

Worker-N

Handler

Controller

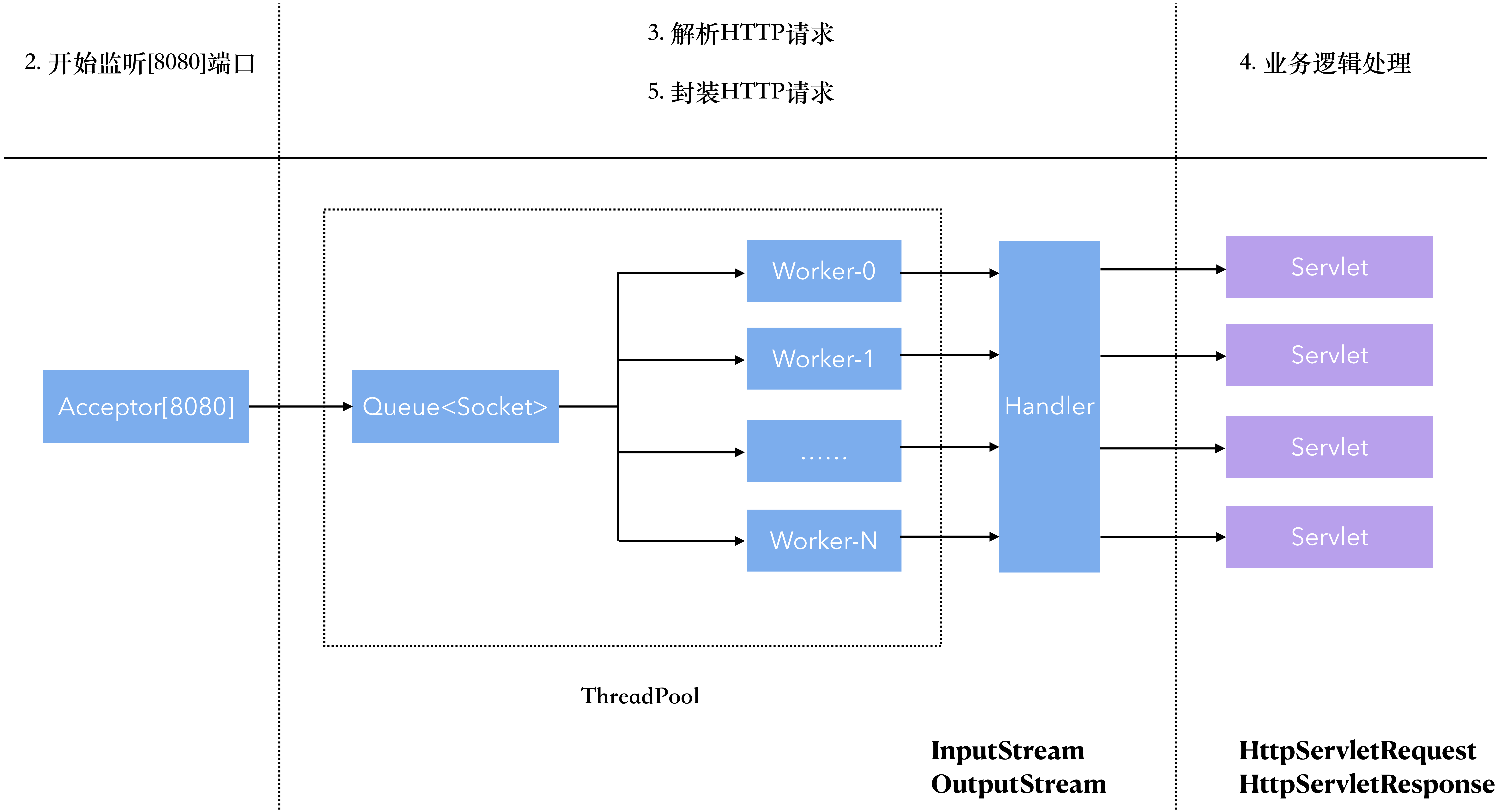
Controller

Controller

Controller

ThreadPool

Servlet



2. 开始监听[8080]端口

3. 解析HTTP请求

5. 封装HTTP请求

4. 业务逻辑处理

Acceptor[8080]

Queue<Socket>

Worker-0

Worker-1

……

Worker-N

Handler

Servlet

Servlet

Servlet

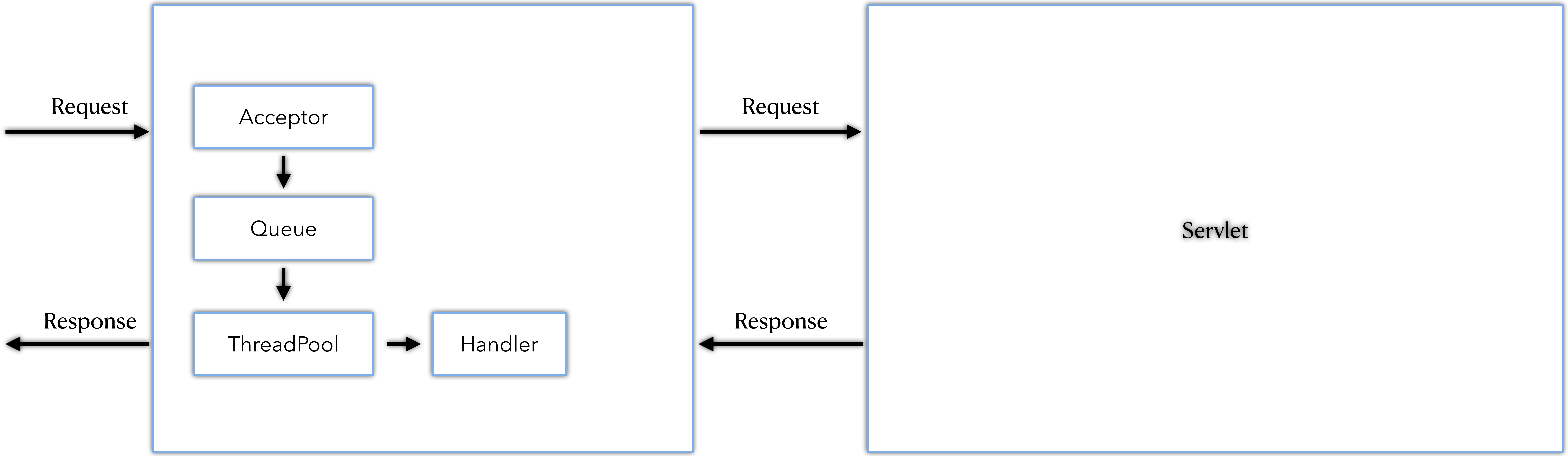
Servlet

ThreadPool

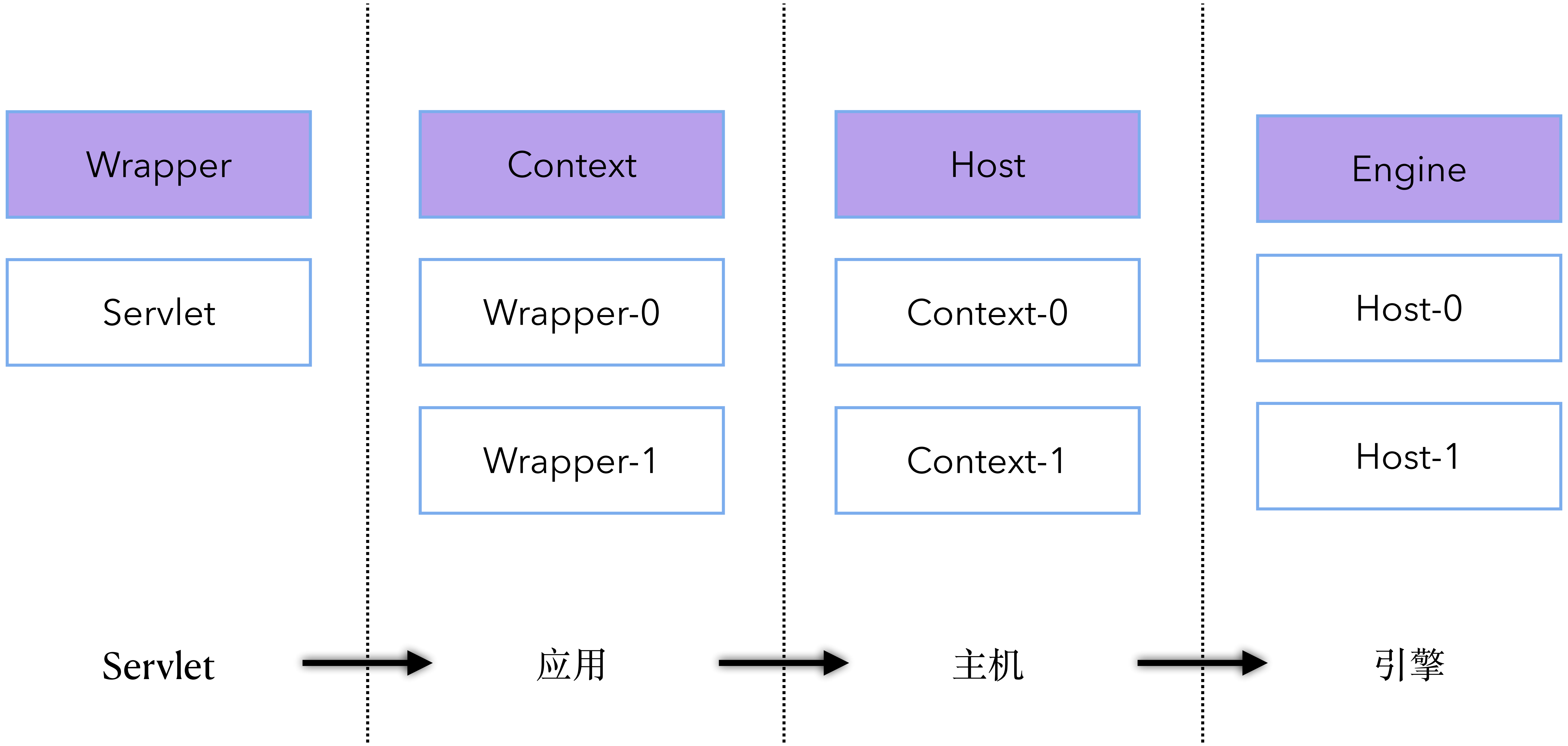
InputStream
OutputStream

HttpServletRequest
HttpServletResponse

流程简化


Request
Acceptor
Queue
Response
ThreadPool
Handler
Request
Response
Servlet

四层模型抽象



Wrapper

Servlet

Context

Wrapper-0

Wrapper-1

Host

Context-0

Context-1

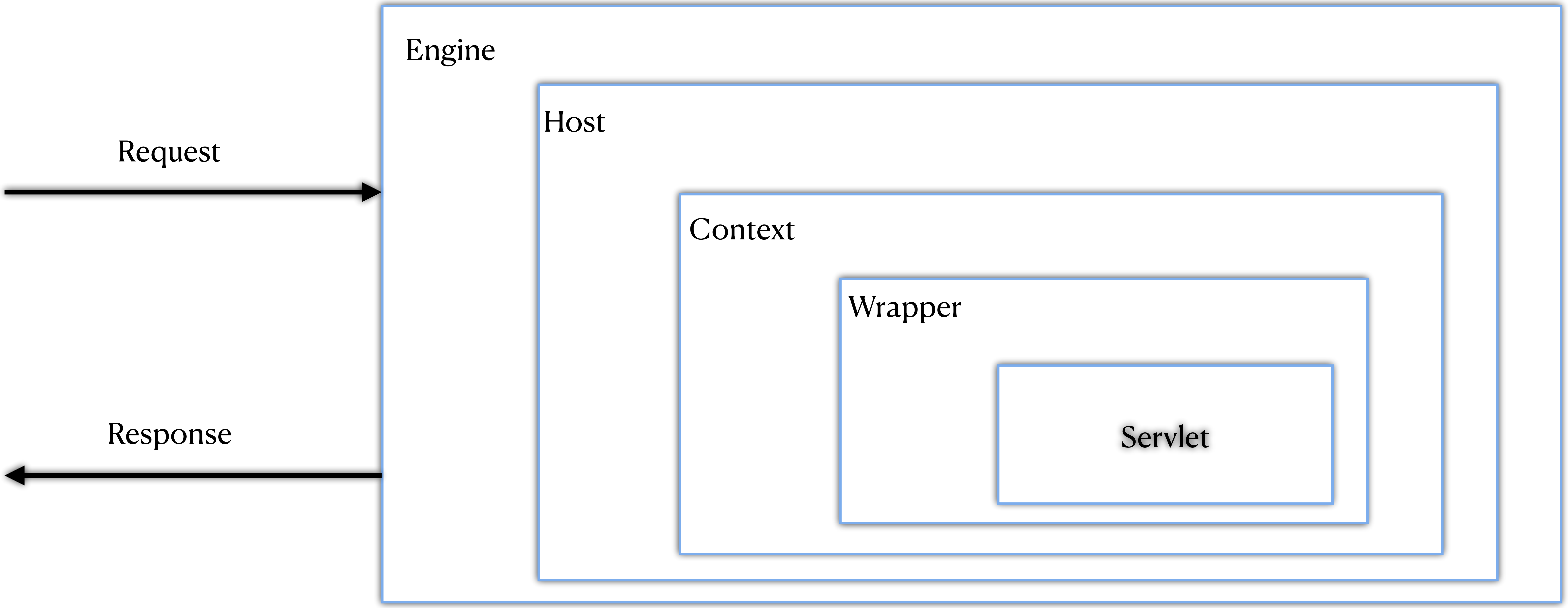
Engine

Host-0

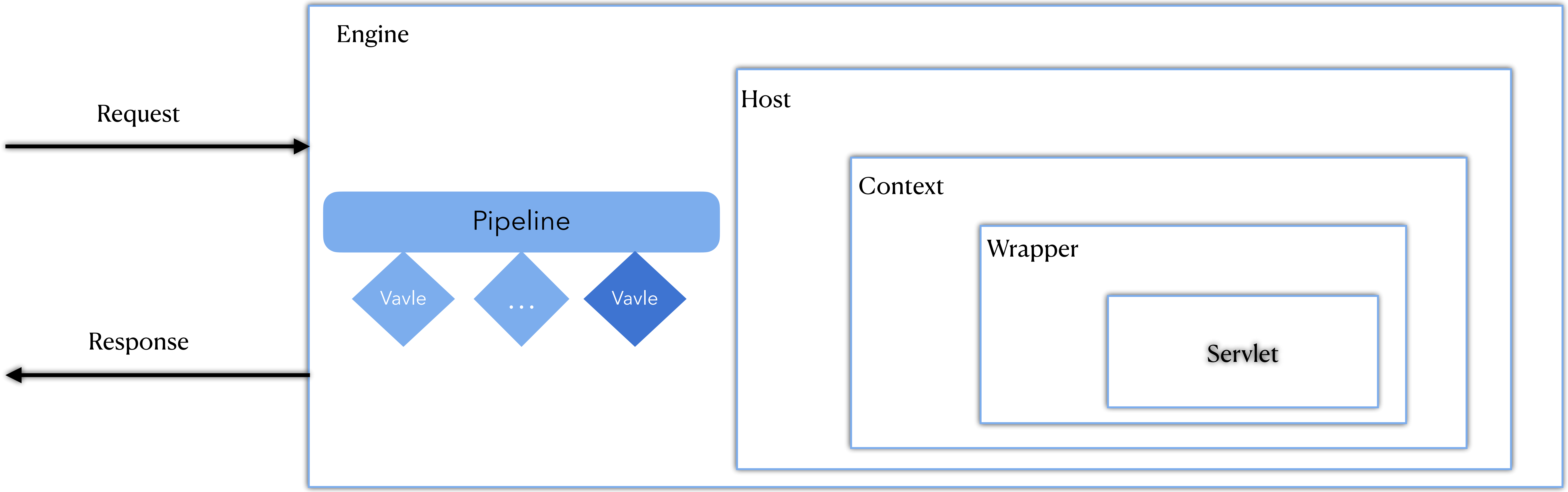
Host-1

Servlet → 应用 → 主机 → 引擎

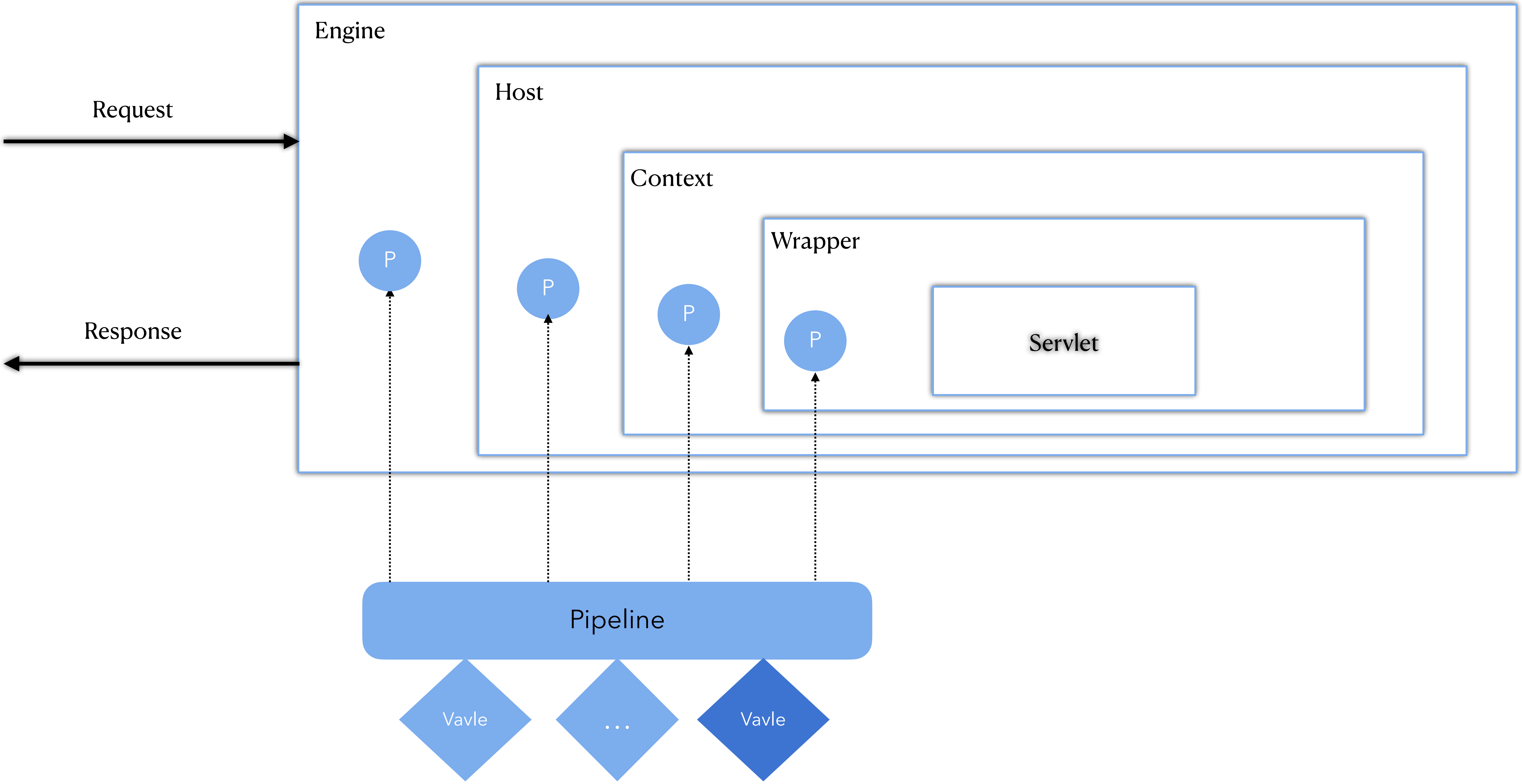
四层模型抽象



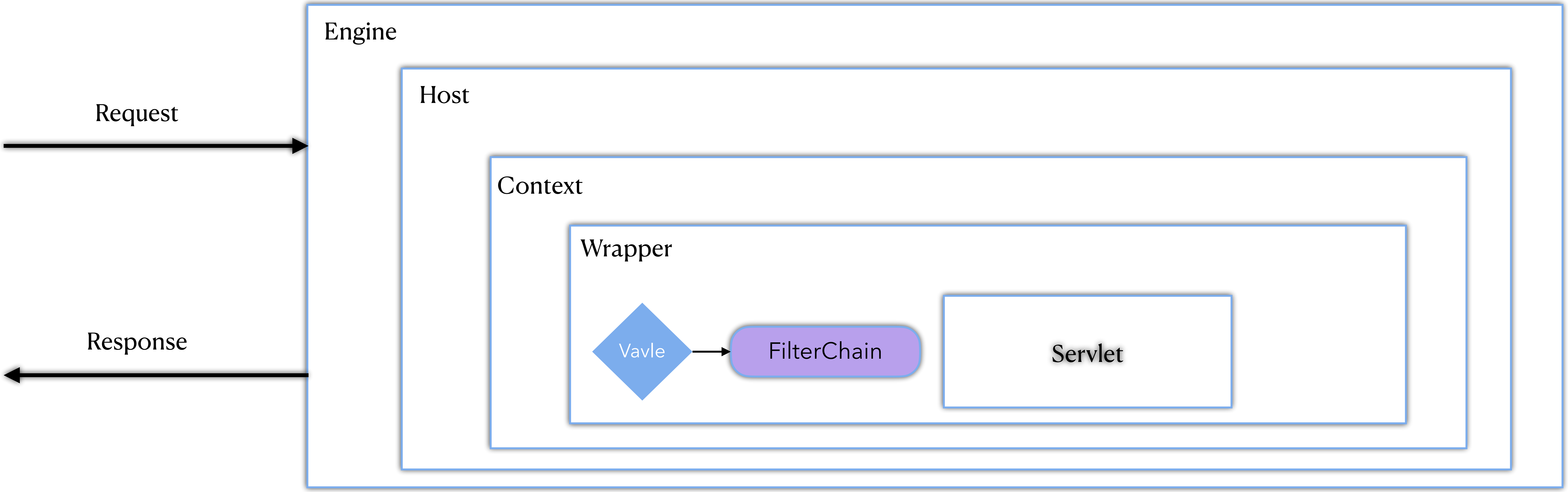
请求流转


Request
Response
Engine
Pipeline
Vavle
. . .
Vavle
Host
Context
Wrapper
Servlet

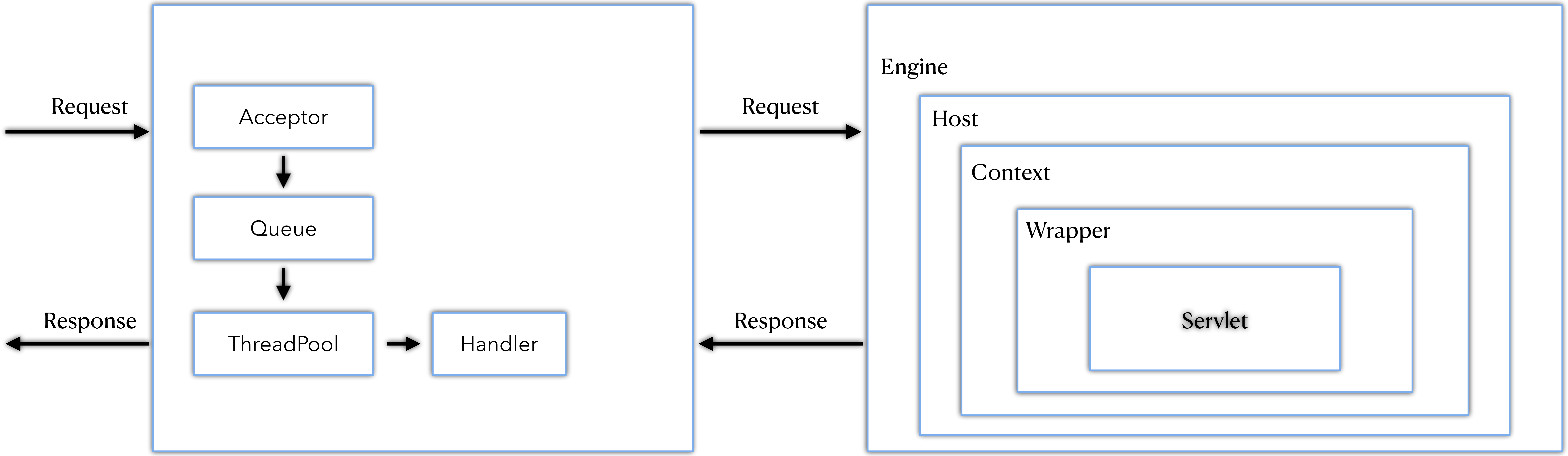
请求流转


Engine
Host
Context
Wrapper
Servlet
Request
Response
P
P
P
P
Pipeline
Vavle
...
Vavle

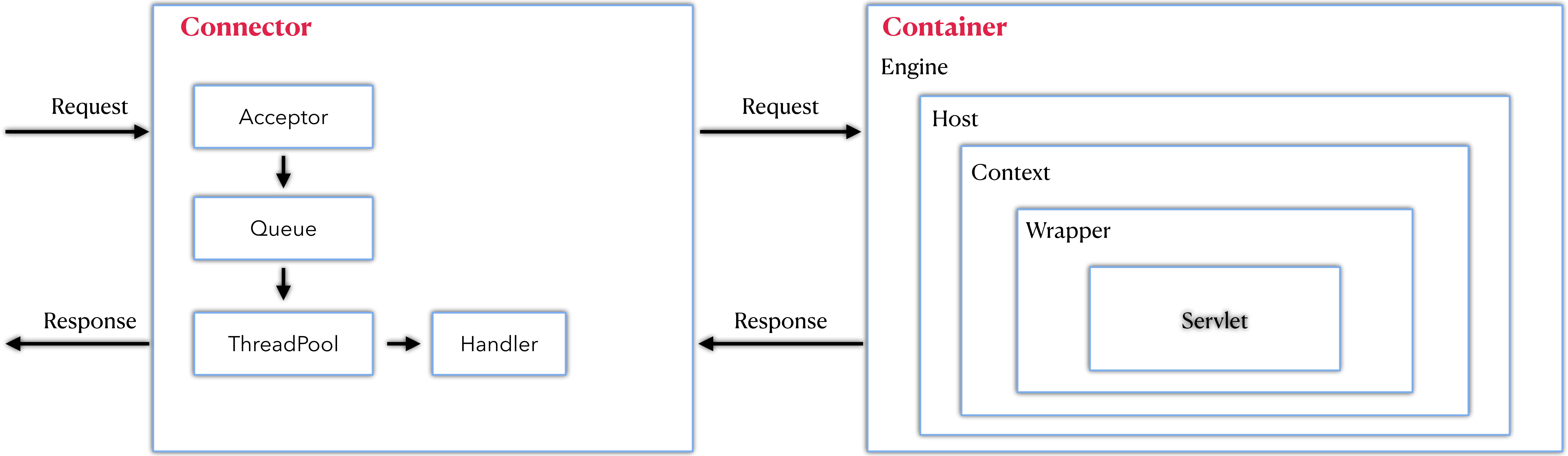
FilterChain



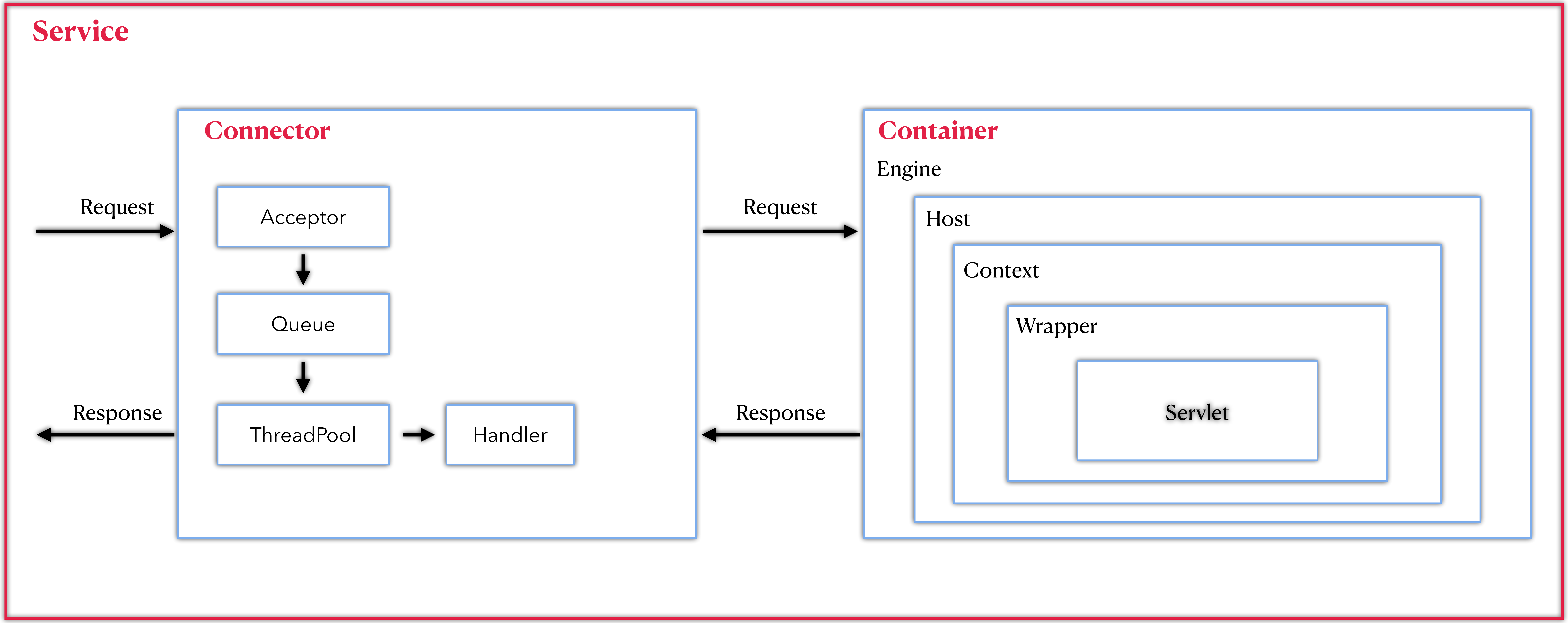
整合一下


Request
Acceptor
Queue
Response
ThreadPool
Handler
Request
Response
Engine
Host
Context
Wrapper
Servlet

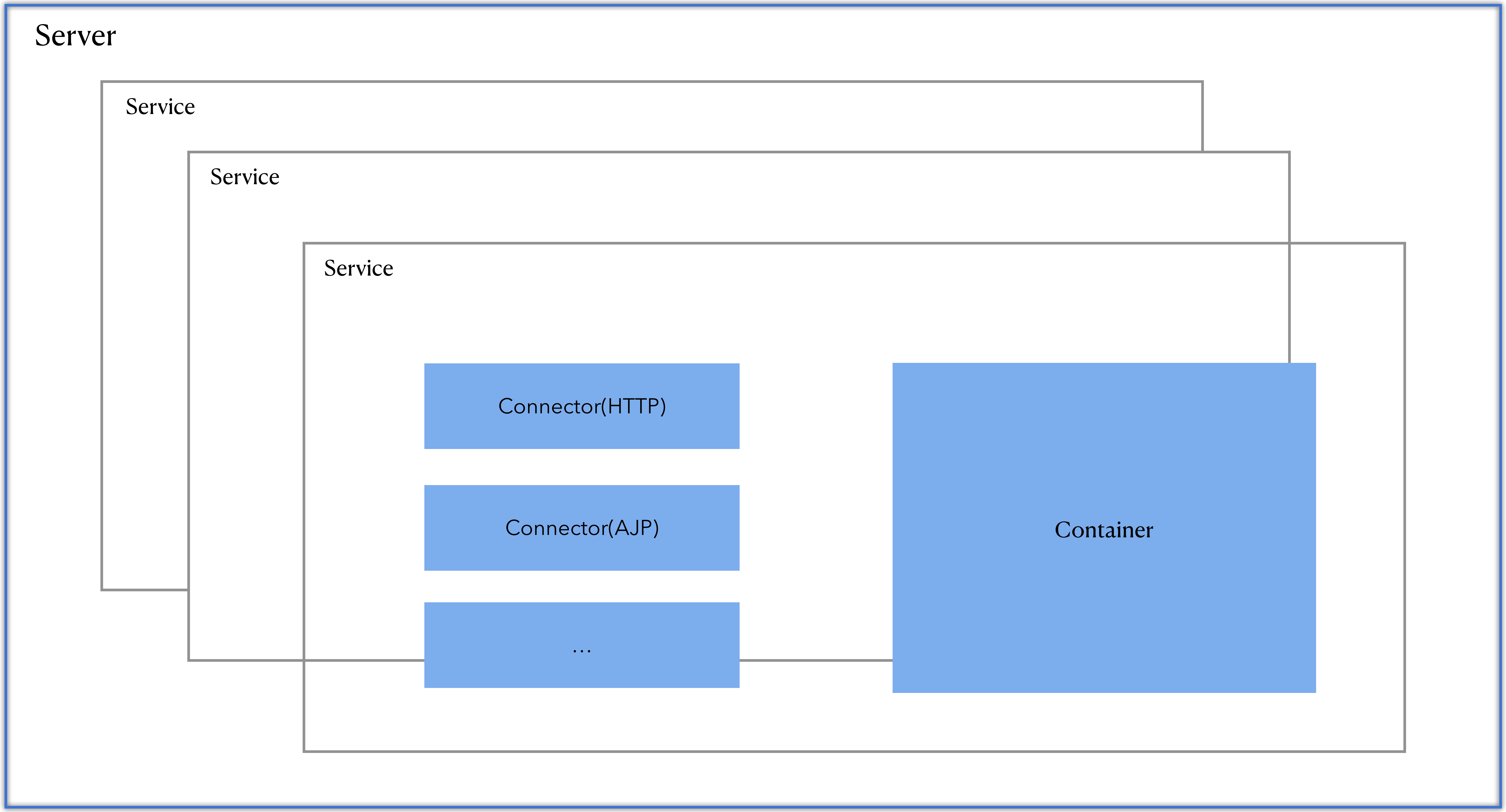
起个名字-Connector&Container



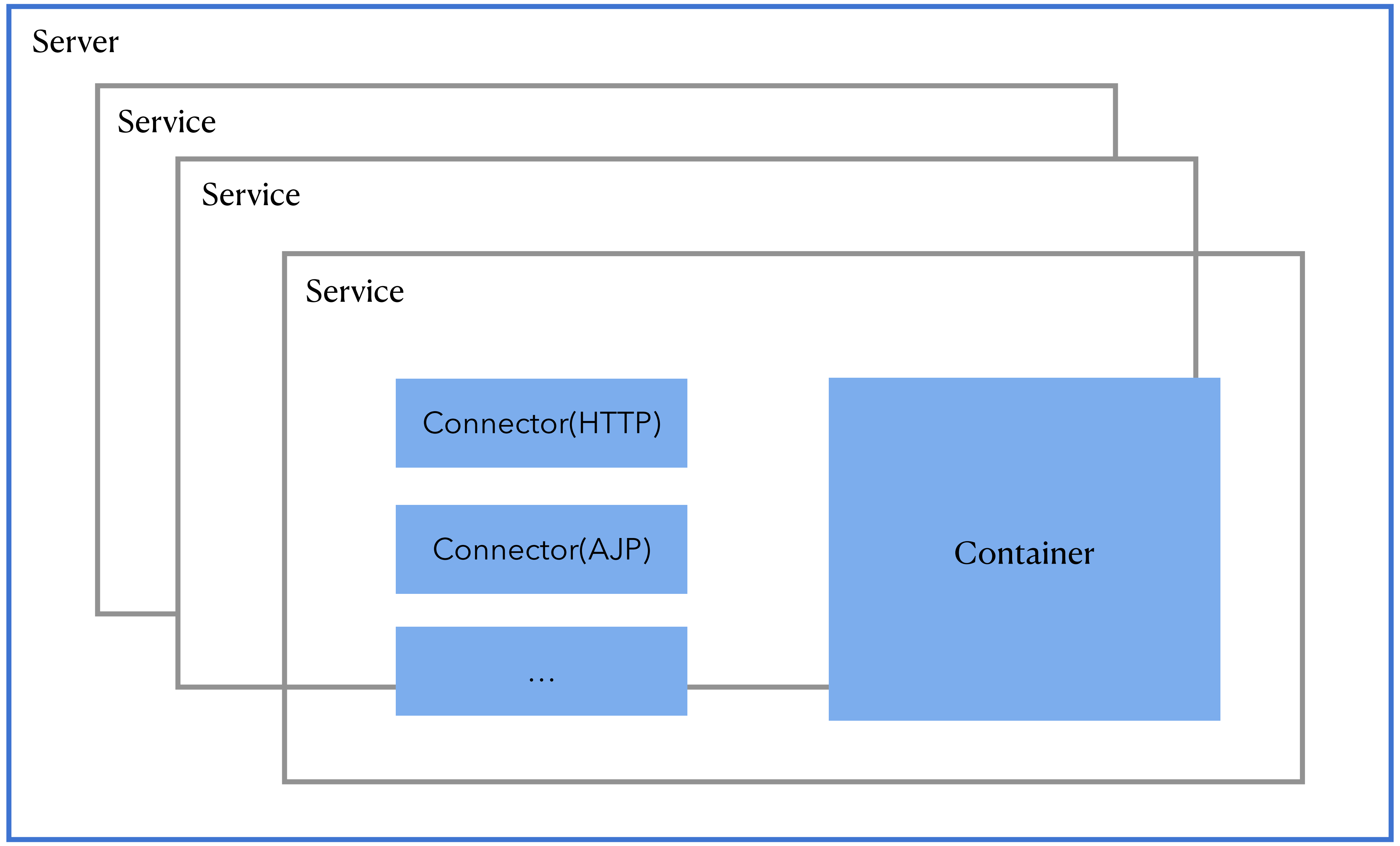
再起名字-Service



再起名字-Server


Server
Service
Service
Service
Connector(HTTP)
Connector(AJP)
…
Container

配置文件

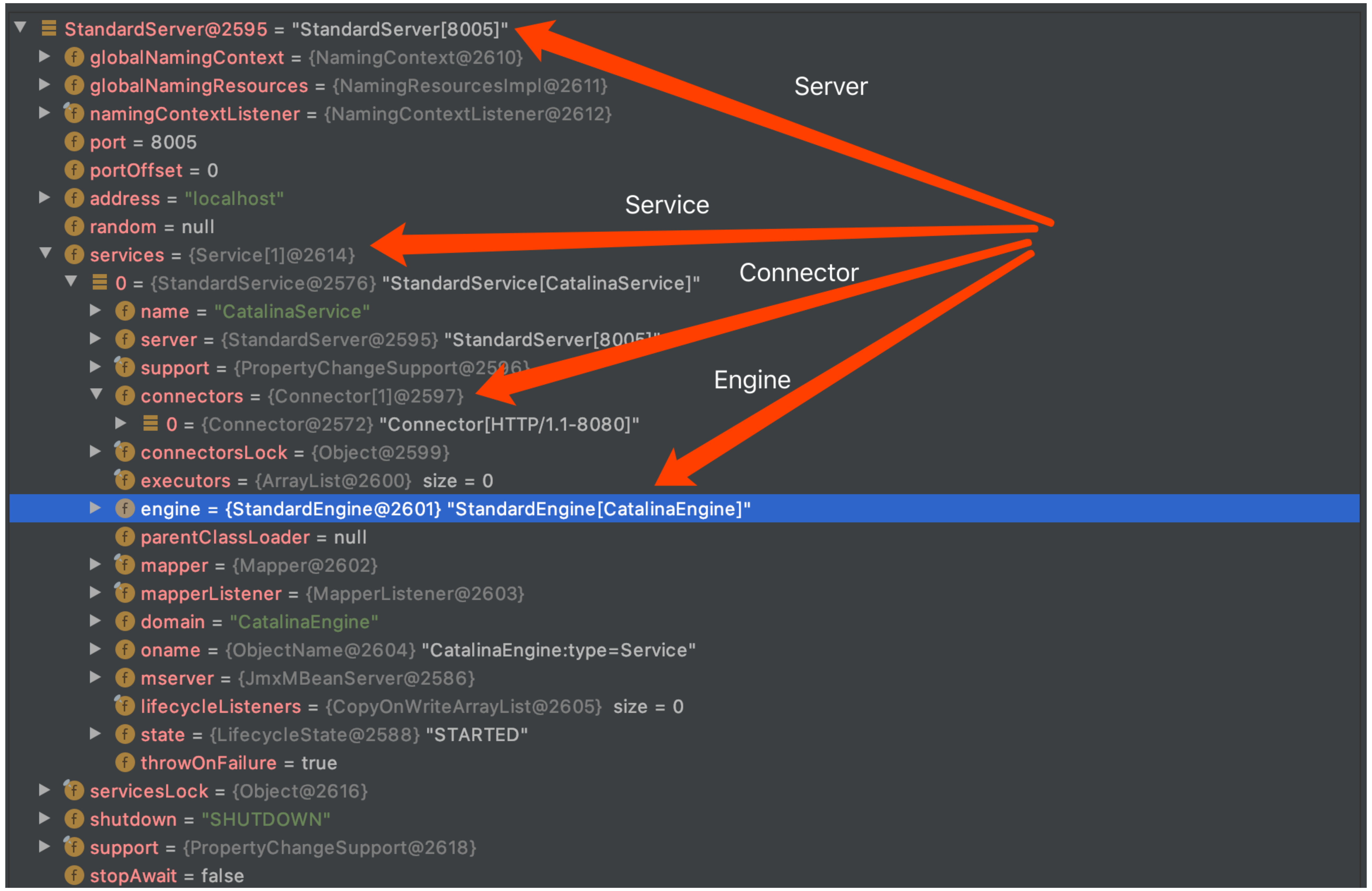


Tomcat组件结构

```
<?xml version="1.0" encoding="UTF-8"?>
<Server port="8005" shutdown="SHUTDOWN">
  <Service name="Catalina">

    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" URIEncoding="UTF-8"/>

    <Engine name="Catalina" defaultHost="localhost">

      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">

        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
               prefix="localhost_access_log" suffix=".txt"
               pattern="%h %l %u %t &quot;%r&quot; %s %b" />
        <Context />
      </Host>
    </Engine>
  </Service>
</Server>
```

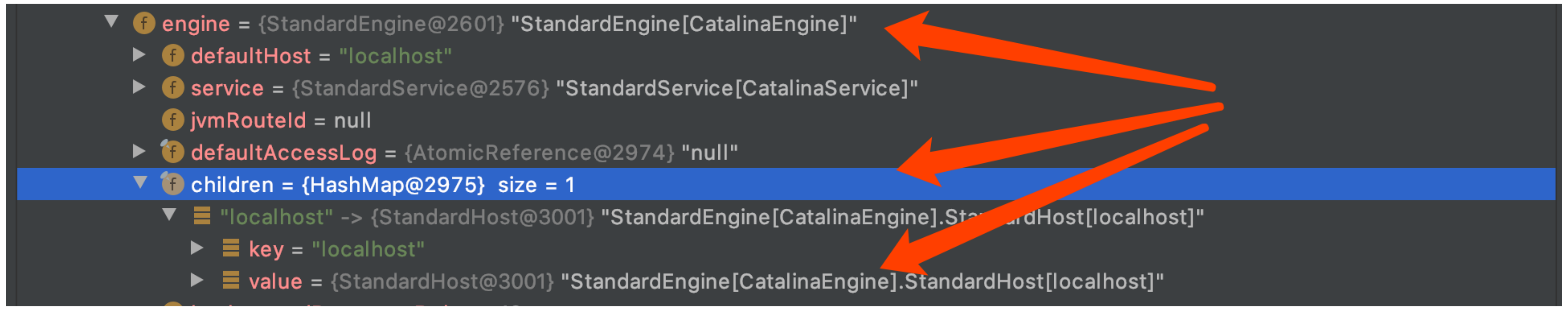
Tomcat配置文件

Server实例



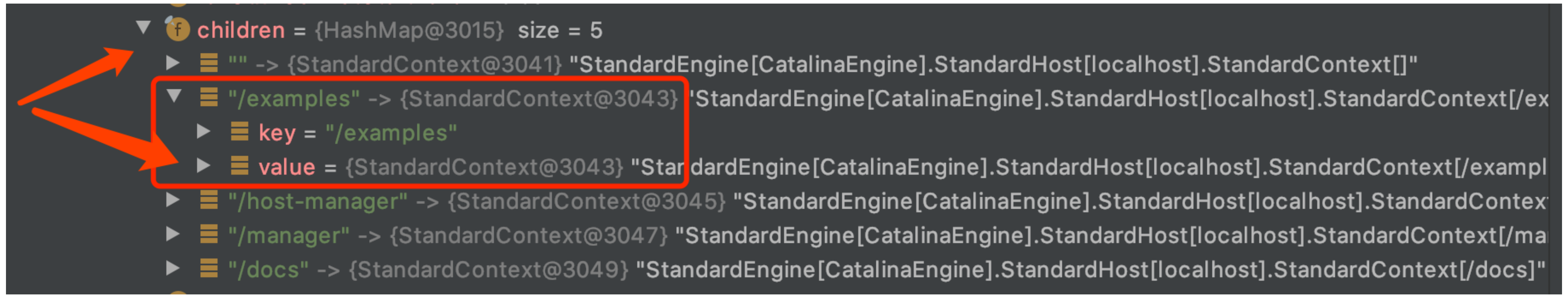
```
▼ StandardServer@2595 = "StandardServer[8005]"
  ▶ globalNamingContext = {NamingContext@2610}
  ▶ globalNamingResources = {NamingResourcesImpl@2611}
  ▶ namingContextListener = {NamingContextListener@2612}
    port = 8005
    portOffset = 0
  ▶ address = "localhost"
    random = null
  ▼ services = {Service[1]@2614}
    ▼ 0 = {StandardService@2576} "StandardService[CatalinaService]"
      ▶ name = "CatalinaService"
      ▶ server = {StandardServer@2595} "StandardServer[8005]"
      ▶ support = {PropertyChangeSupport@2596}
      ▼ connectors = {Connector[1]@2597}
        ▶ 0 = {Connector@2572} "Connector[HTTP/1.1-8080]"
      ▶ connectorsLock = {Object@2599}
        executors = {ArrayList@2600}  size = 0
      ▶ engine = {StandardEngine@2601} "StandardEngine[CatalinaEngine]"
        parentClassLoader = null
      ▶ mapper = {Mapper@2602}
      ▶ mapperListener = {MapperListener@2603}
      ▶ domain = "CatalinaEngine"
      ▶ oname = {ObjectName@2604} "CatalinaEngine:type=Service"
      ▶ mserver = {JmxMBeanServer@2586}
        lifecycleListeners = {CopyOnWriteArrayList@2605}  size = 0
      ▶ state = {LifecycleState@2588} "STARTED"
        throwOnFailure = true
  ▶ servicesLock = {Object@2616}
  ▶ shutdown = "SHUTDOWN"
  ▶ support = {PropertyChangeSupport@2618}
    stopAwait = false
```

Container实例->Engine



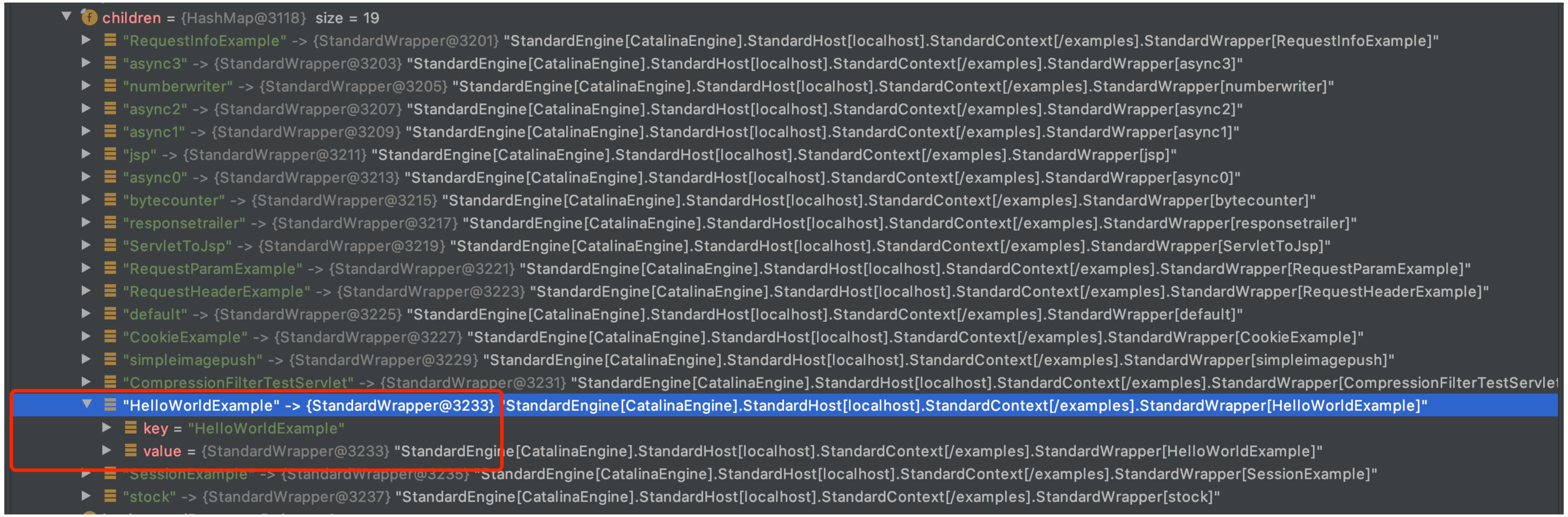
```
▼ engine = {StandardEngine@2601} "StandardEngine[CatalinaEngine]"
  ▶ defaultHost = "localhost"
  ▶ service = {StandardService@2576} "StandardService[CatalinaService]"
    jvmRouteId = null
  ▶ defaultAccessLog = {AtomicReference@2974} "null"
  ▼ children = {HashMap@2975}  size = 1
    ▼ "localhost" -> {StandardHost@3001} "StandardEngine[CatalinaEngine].StandardHost[localhost]"
      ▶ key = "localhost"
      ▶ value = {StandardHost@3001} "StandardEngine[CatalinaEngine].StandardHost[localhost]"
```

Container实例->Context



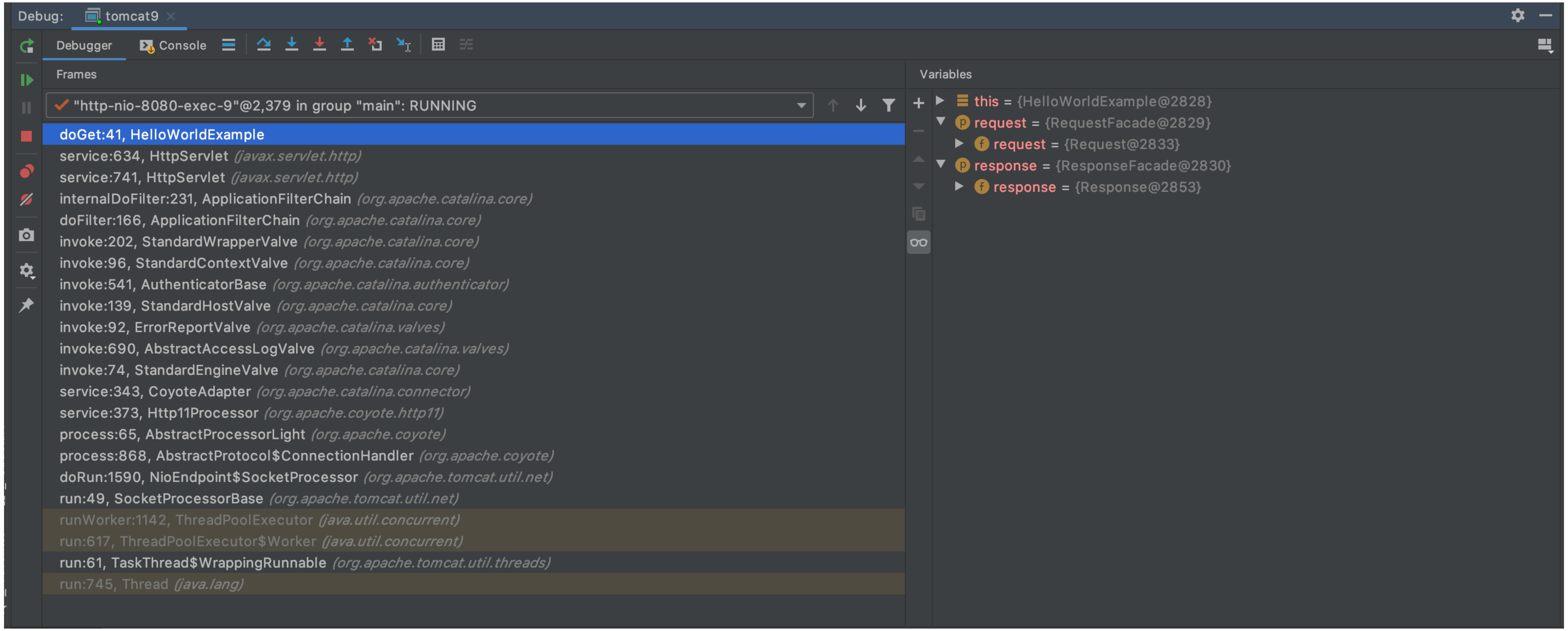
```
▼ children = {HashMap@3015}  size = 5
  ▶ "" -> {StandardContext@3041} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[]"
  ▼ "/examples" -> {StandardContext@3043} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/ex
    ▶ key = "/examples"
    ▶ value = {StandardContext@3043} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/exampl
  ▶ "/host-manager" -> {StandardContext@3045} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContex
  ▶ "/manager" -> {StandardContext@3047} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/ma
  ▶ "/docs" -> {StandardContext@3049} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/docs]"
```

Container实例->Wrapper



```
▼ children = {HashMap@3118}  size = 19
  ▶ "RequestInfoExample" -> {StandardWrapper@3201} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[RequestInfoExample]"
  ▶ "async3" -> {StandardWrapper@3203} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[async3]"
  ▶ "numberwriter" -> {StandardWrapper@3205} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[numberwriter]"
  ▶ "async2" -> {StandardWrapper@3207} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[async2]"
  ▶ "async1" -> {StandardWrapper@3209} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[async1]"
  ▶ "jsp" -> {StandardWrapper@3211} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[jsp]"
  ▶ "async0" -> {StandardWrapper@3213} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[async0]"
  ▶ "bytecounter" -> {StandardWrapper@3215} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[bytecounter]"
  ▶ "responsetrailer" -> {StandardWrapper@3217} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[responsetrailer]"
  ▶ "ServletToJsp" -> {StandardWrapper@3219} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[ServletToJsp]"
  ▶ "RequestParamExample" -> {StandardWrapper@3221} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[RequestParamExample]"
  ▶ "RequestHeaderExample" -> {StandardWrapper@3223} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[RequestHeaderExample]"
  ▶ "default" -> {StandardWrapper@3225} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[default]"
  ▶ "CookieExample" -> {StandardWrapper@3227} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[CookieExample]"
  ▶ "simpleimagepush" -> {StandardWrapper@3229} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[simpleimagepush]"
  ▶ "CompressionFilterTestServlet" -> {StandardWrapper@3231} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[CompressionFilterTestServle
  ▼ "HelloWorldExample" -> {StandardWrapper@3233} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[HelloWorldExample]"
    ▶ key = "HelloWorldExample"
    ▶ value = {StandardWrapper@3233} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[HelloWorldExample]"
  ▶ "SessionExample" -> {StandardWrapper@3235} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[SessionExample]"
  ▶ "stock" -> {StandardWrapper@3237} "StandardEngine[CatalinaEngine].StandardHost[localhost].StandardContext[/examples].StandardWrapper[stock]"
```

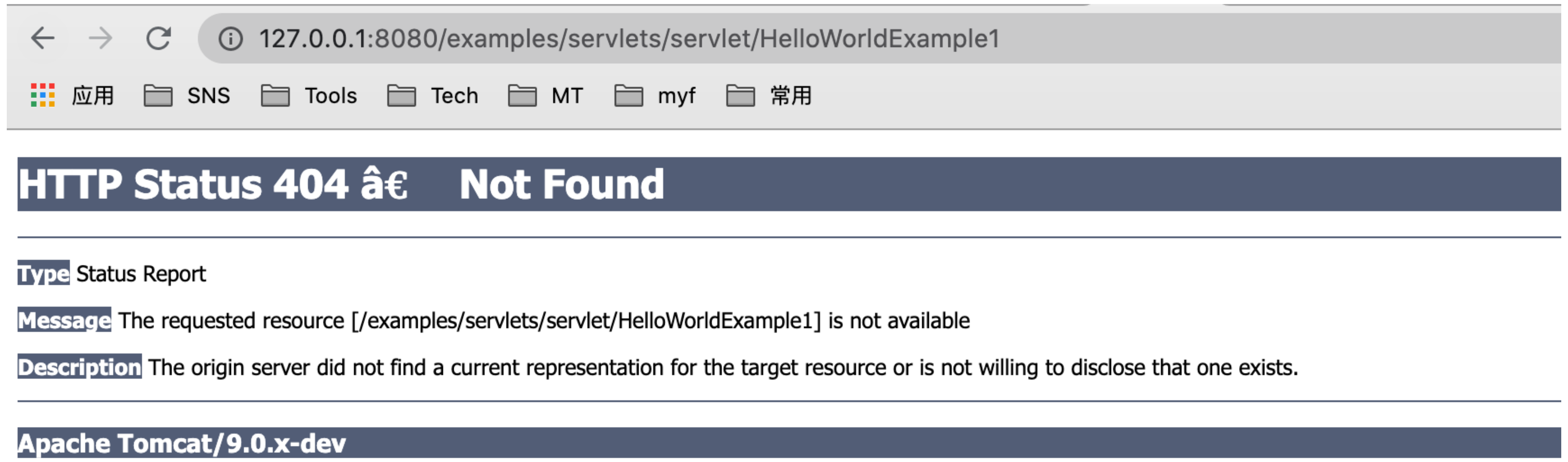
HelloWorldExample



```
"http-nio-8080-exec-9"@2,379 in group "main": RUNNING
doGet:41, HelloWorldExample
service:634, HttpServlet (javax.servlet.http)
service:741, HttpServlet (javax.servlet.http)
internalDoFilter:231, ApplicationFilterChain (org.apache.catalina.core)
doFilter:166, ApplicationFilterChain (org.apache.catalina.core)
invoke:202, StandardWrapperValve (org.apache.catalina.core)
invoke:96, StandardContextValve (org.apache.catalina.core)
invoke:541, AuthenticatorBase (org.apache.catalina.authenticator)
invoke:139, StandardHostValve (org.apache.catalina.core)
invoke:92, ErrorReportValve (org.apache.catalina.valves)
invoke:690, AbstractAccessLogValve (org.apache.catalina.valves)
invoke:74, StandardEngineValve (org.apache.catalina.core)
service:343, CoyoteAdapter (org.apache.catalina.connector)
service:373, Http11Processor (org.apache.coyote.http11)
process:65, AbstractProcessorLight (org.apache.coyote)
process:868, AbstractProtocol$ConnectionHandler (org.apache.coyote)
doRun:1590, NioEndpoint$SocketProcessor (org.apache.tomcat.util.net)
run:49, SocketProcessorBase (org.apache.tomcat.util.net)
runWorker:1142, ThreadPoolExecutor (java.util.concurrent)
run:617, ThreadPoolExecutor$Worker (java.util.concurrent)
run:61, TaskThread$WrappingRunnable (org.apache.tomcat.util.threads)
run:745, Thread (java.lang)
```

```
▶ this = {HelloWorldExample@2828}
▼ request = {RequestFacade@2829}
    ▶ request = {Request@2833}
▼ response = {ResponseFacade@2830}
    ▶ response = {Response@2853}
```

ErrorReportVavle



HTTP Status 404 â€   Not Found

Type Status Report

Message The requested resource [/examples/servlets/servlet/HelloWorldExample1] is not available

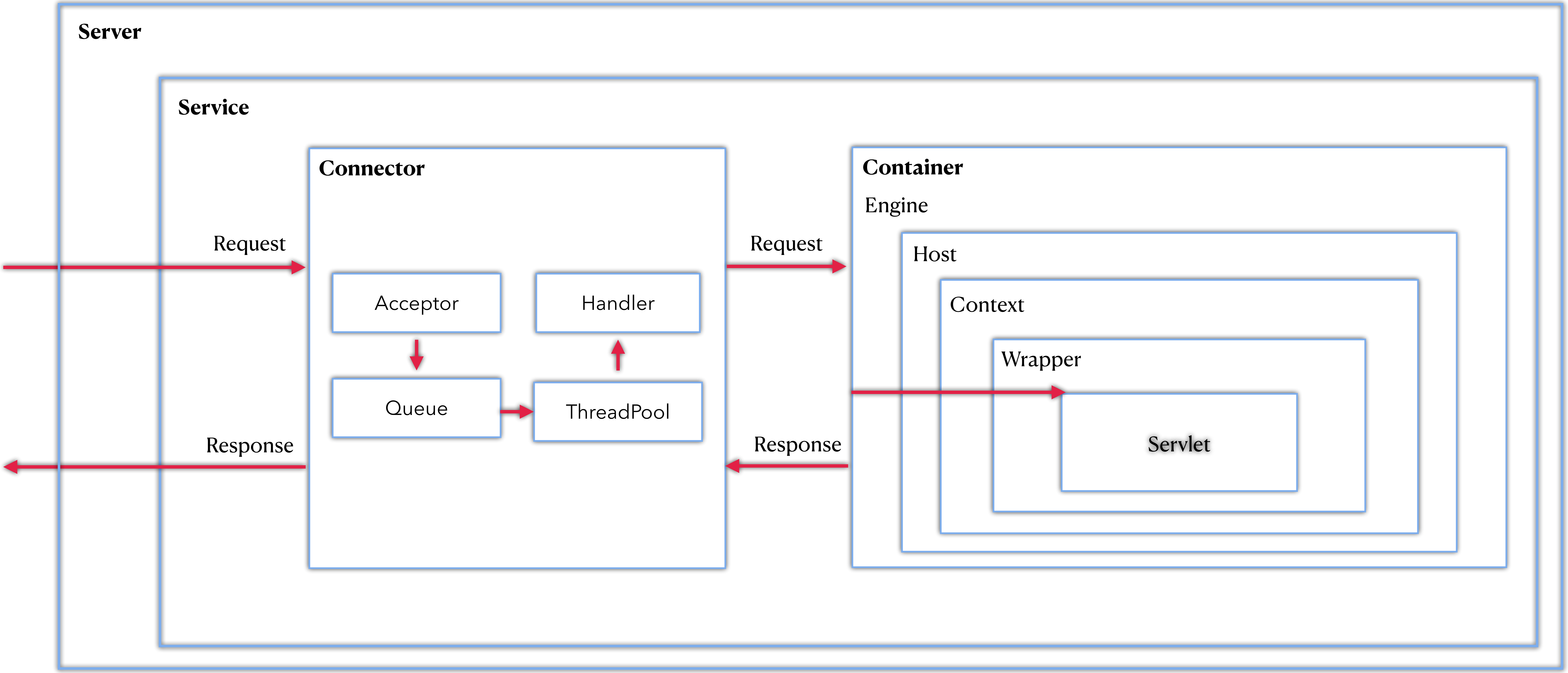
Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/9.0.x-dev

ErrorReportVavle

```java
StringBuilder sb = new StringBuilder();
sb.append("<!doctype html><html lang=\"");
sb.append(smClient.getLocale().getLanguage()).append("\">");
sb.append("<head>");
sb.append("<title>");
sb.append(smClient.getString("errorReportValve.statusHeader",
        String.valueOf(statusCode), reason));
sb.append("</title>");
sb.append("<style type=\"text/css\">");
sb.append(TomcatCSS.TOMCAT_CSS);
sb.append("</style>");
sb.append("</head><body>");
sb.append("<h1>");
sb.append(smClient.getString("errorReportValve.statusHeader",
        String.valueOf(statusCode), reason)).append("</h1>");
if (isShowReport()) {
    sb.append("<hr class=\"line\" />");
    sb.append("<p><b>");
    sb.append(smClient.getString("errorReportValve.type"));
    sb.append("</b> ");
    if (throwable != null) {
        sb.append(smClient.getString("errorReportValve.exceptionReport"));
    } else {
        sb.append(smClient.getString("errorReportValve.statusReport"));
    }
    sb.append("</p>");
    if (!message.isEmpty()) {
        sb.append("<p><b>");
        sb.append(smClient.getString("errorReportValve.message"));
        sb.append("</b> ");
        sb.append(message).append("</p>");
    }
```

回顾一遍


Server
Service
Connector
Request
Acceptor
Handler
Queue
ThreadPool
Response
Request
Response
Container
Engine
Host
Context
Wrapper
Servlet

优点

1. 扩展
2. 可替换

参考

1. How Tomcat Works
2. apache-tomcat-9.0.36-src

后续

1. Tomcat启动&关闭过程-组件装配
2. Tomcat的IO实现-几种实现
3. Tomcat中的设计模式
4. Tomcat各组件的生命周期设计
5. Tomcat如何隔离不同的应用
6. 一次HTTP请求在Jetty中的处理流程
7. 其它