**REpresentational State Transfer**

**RESTful的约束：**

**1. 使用客户/服务器模型。客户和服务器之间通过一个统一的接口来互相通讯。**

**2. 层次化的系统。在一个REST系统中，客户端并不会固定地与一个服务器打交道。**

**3. 无状态。在一个REST系统中，服务端并不会保存有关客户的任何状态。也就是说，客户端自身负责用户状态的维持，并在每次发送请求时都需要提供足够的信息。**

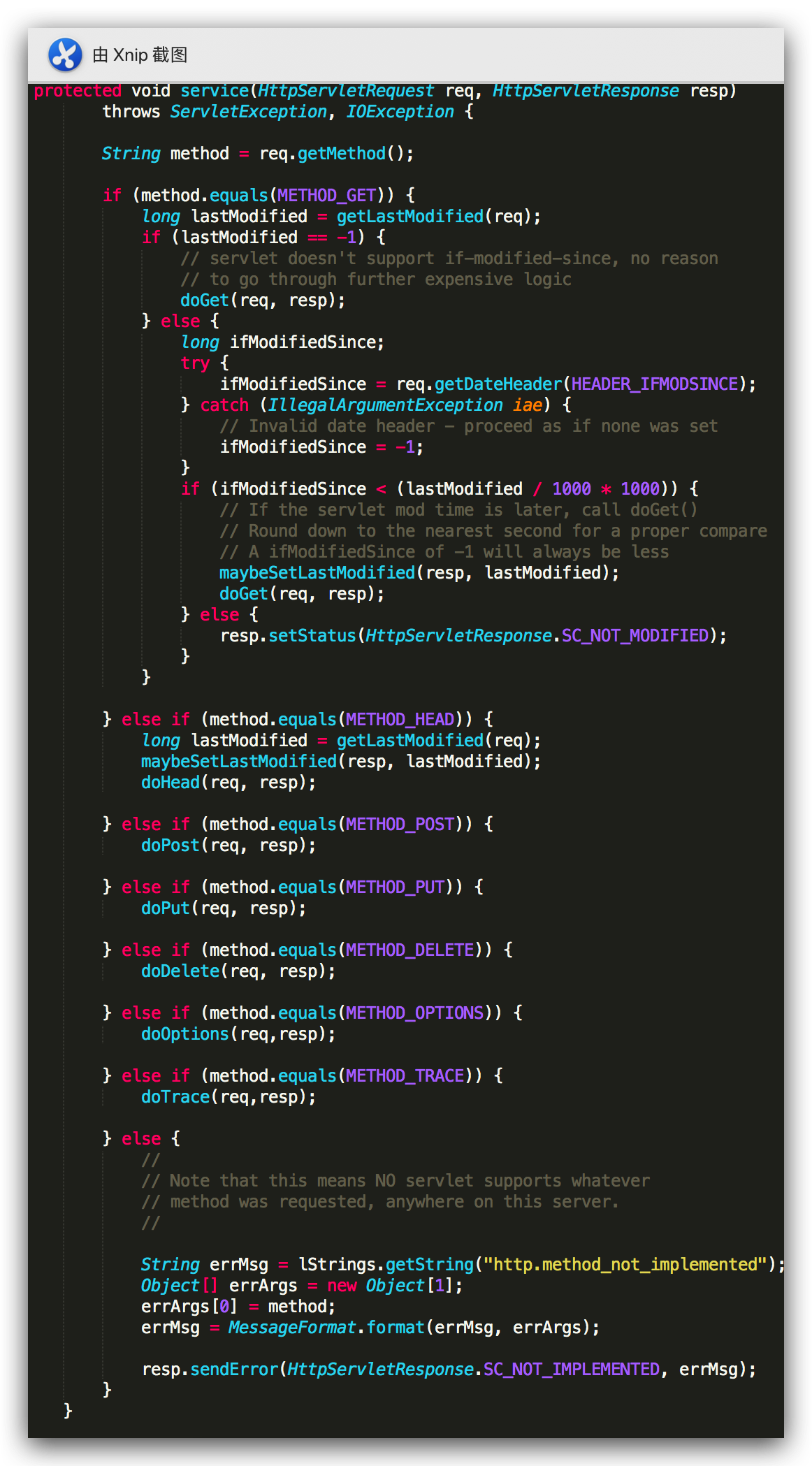
**4. 可缓存。REST系统需要能够恰当地缓存请求，以尽量减少服务端和客户端之间的信息传输，以提高性能。**

**5. 统一的接口。一个REST系统需要使用一个统一的接口来完成子系统之间以及服务与用户之间的交互。这使得REST系统中的各个子系统可以独自完成演化。**

**Restful web service是一种常见的REST的应用，是遵守REST风格的web服务，是一种ROA(The Resource-Oriented Architecture)(面向资源的架构)。**

**不是RESTful：  
 http://127.0.0.1/user/query/1 GET  根据用户id查询用户数据  
 http://127.0.0.1/user/save POST 新增用户  
 http://127.0.0.1/user/update POST 修改用户信息  
 http://127.0.0.1/user/delete GET/POST 删除用户信息  
RESTful风格：  
 http://127.0.0.1/user/1 GET  根据用户id查询用户数据  
 http://127.0.0.1/user  POST 新增用户  
 http://127.0.0.1/user  PUT 修改用户信息  
 http://127.0.0.1/user  DELETE 删除用户信息**

**Servlet.java / HttpServlet.java**

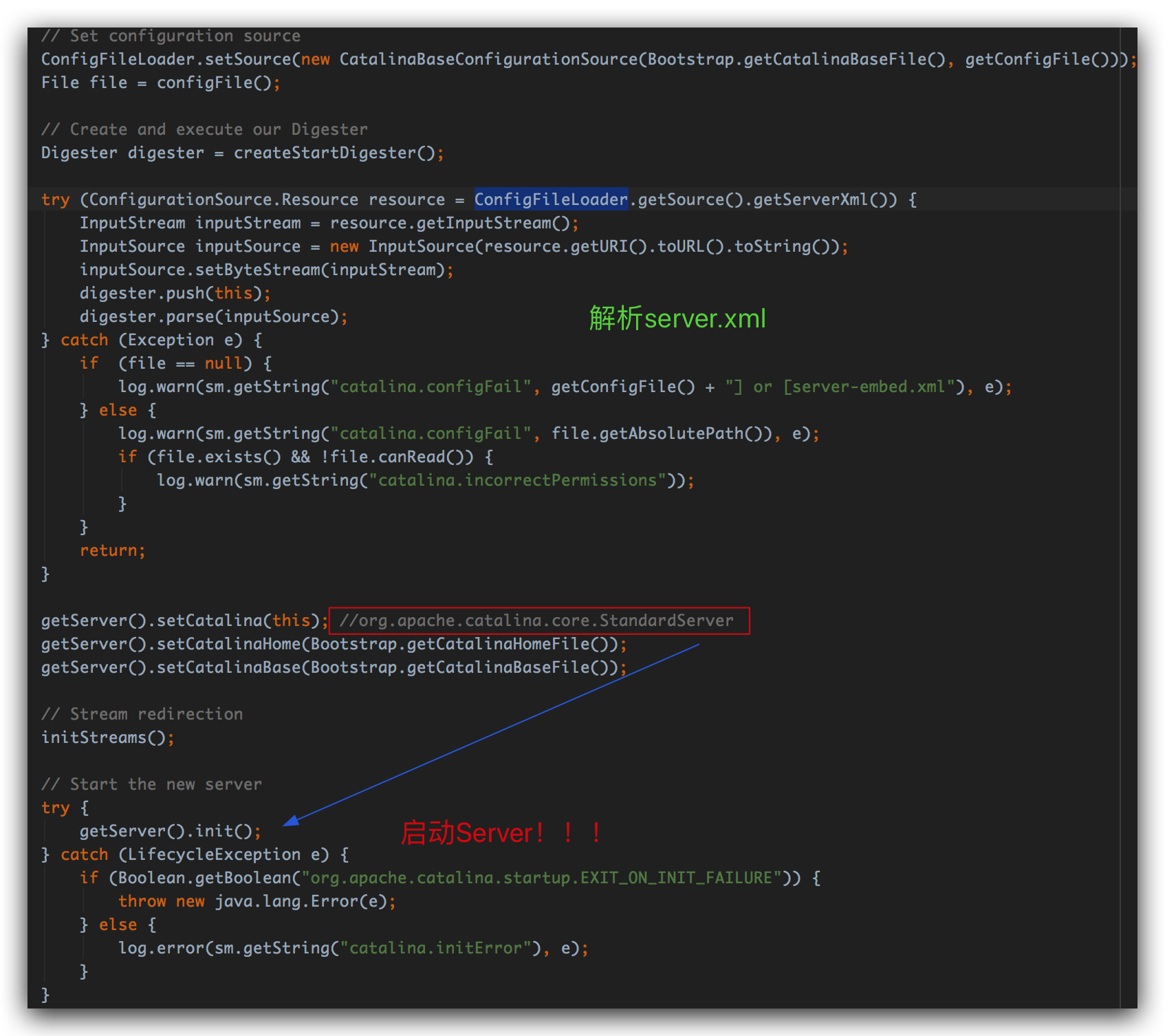
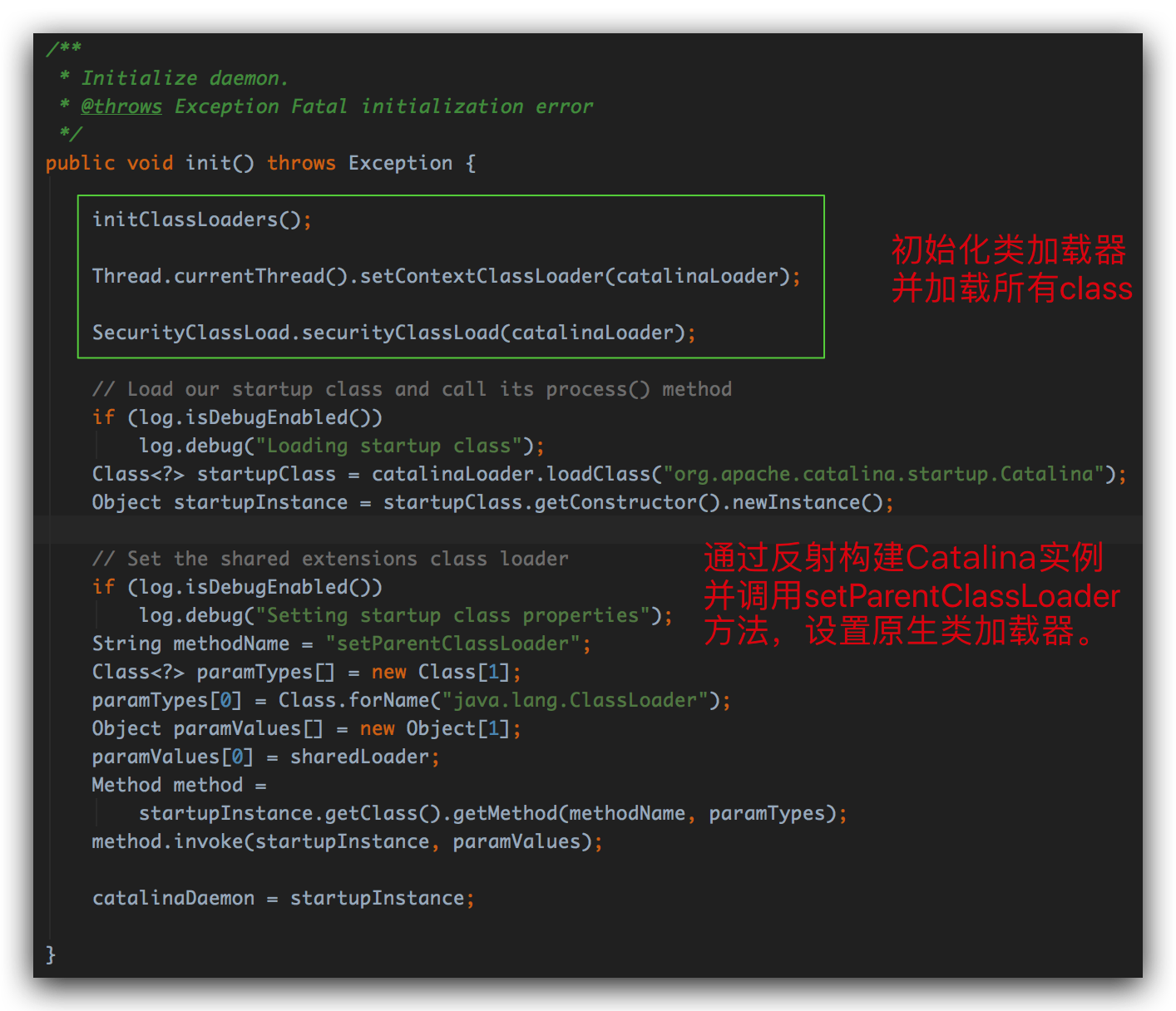
****

**Tomcat**

**组件(server.xml)**

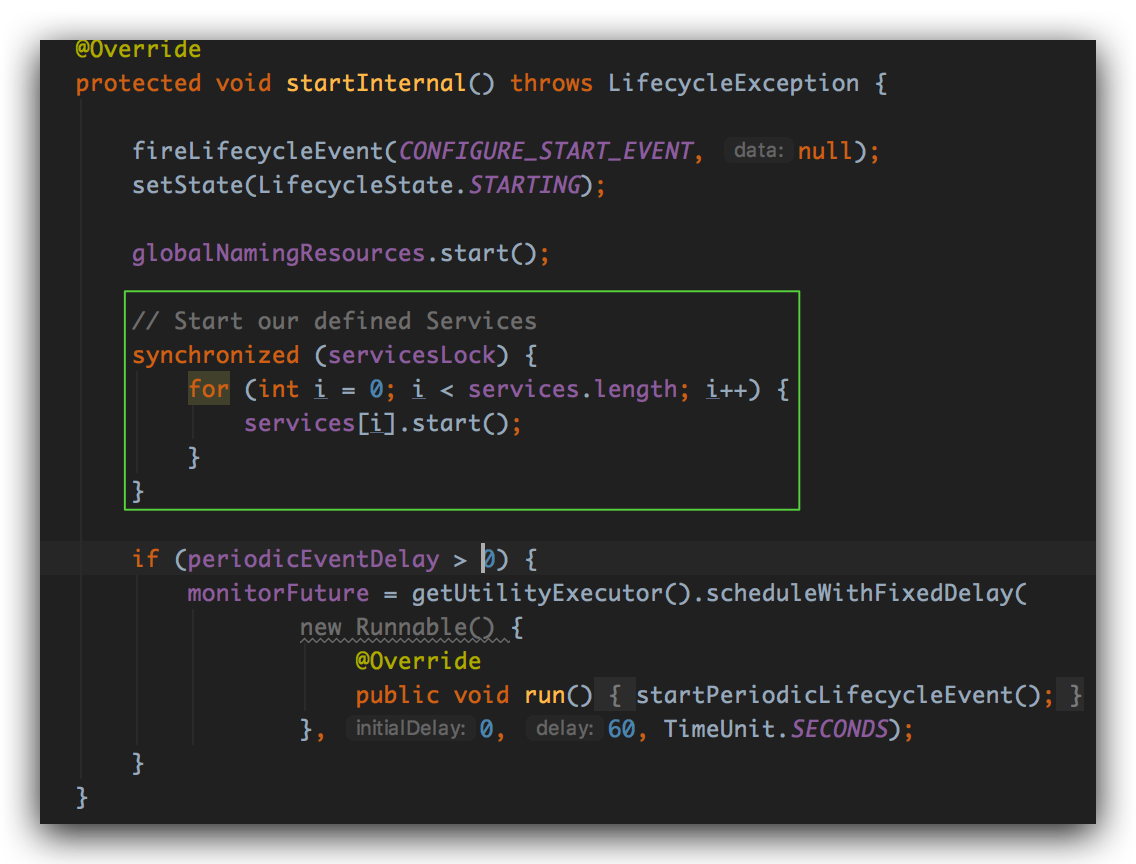
|  |  |
| --- | --- |
| Lifecycle | 组件的生命周期：init、start、stop、destroy |
| Server | Service的容器 |
| Service | Engine和Connector的容器 |
| Connector | 解析某种网络协议的连接器，解析后转发给相应的Engine处理。 |
| Engine | 接收不同Connector的请求，处理后将结果返回给Connector。 |
| Host | 虚拟主机，/webapps |
| Context | 业务应用上下文，/webapps/ROOT |
| Wrapper | 封装servlet【约定：Servlet容器将Servlet类载入内存，并产生Servlet实例和调用service方法。**在一个进程中，每个Servlet类通常只有一个实例**。】（web.xml） |

**Bootstrap** [每个类职责分明，委派不属于自己的职责]





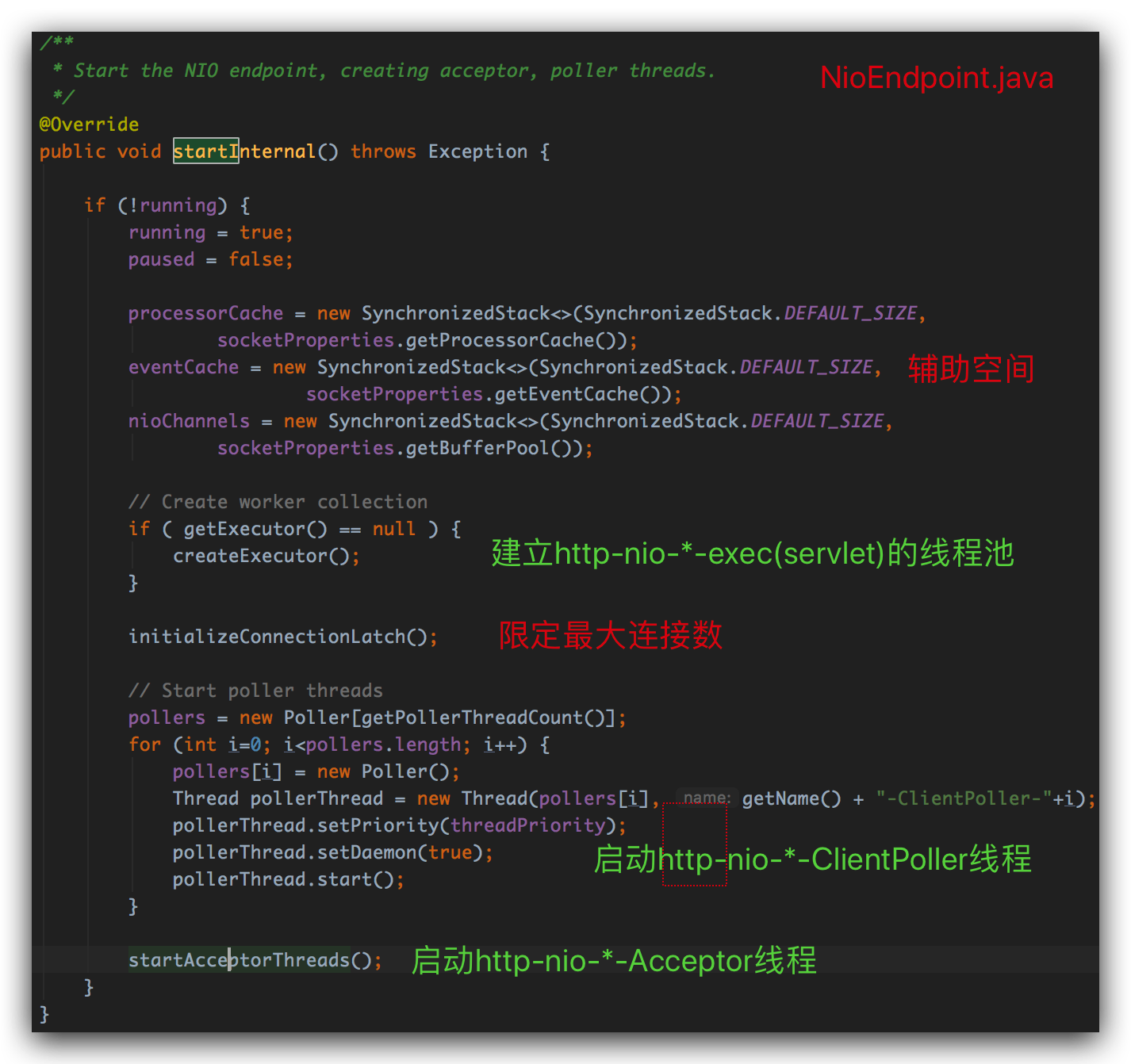
**StandardServer**

****

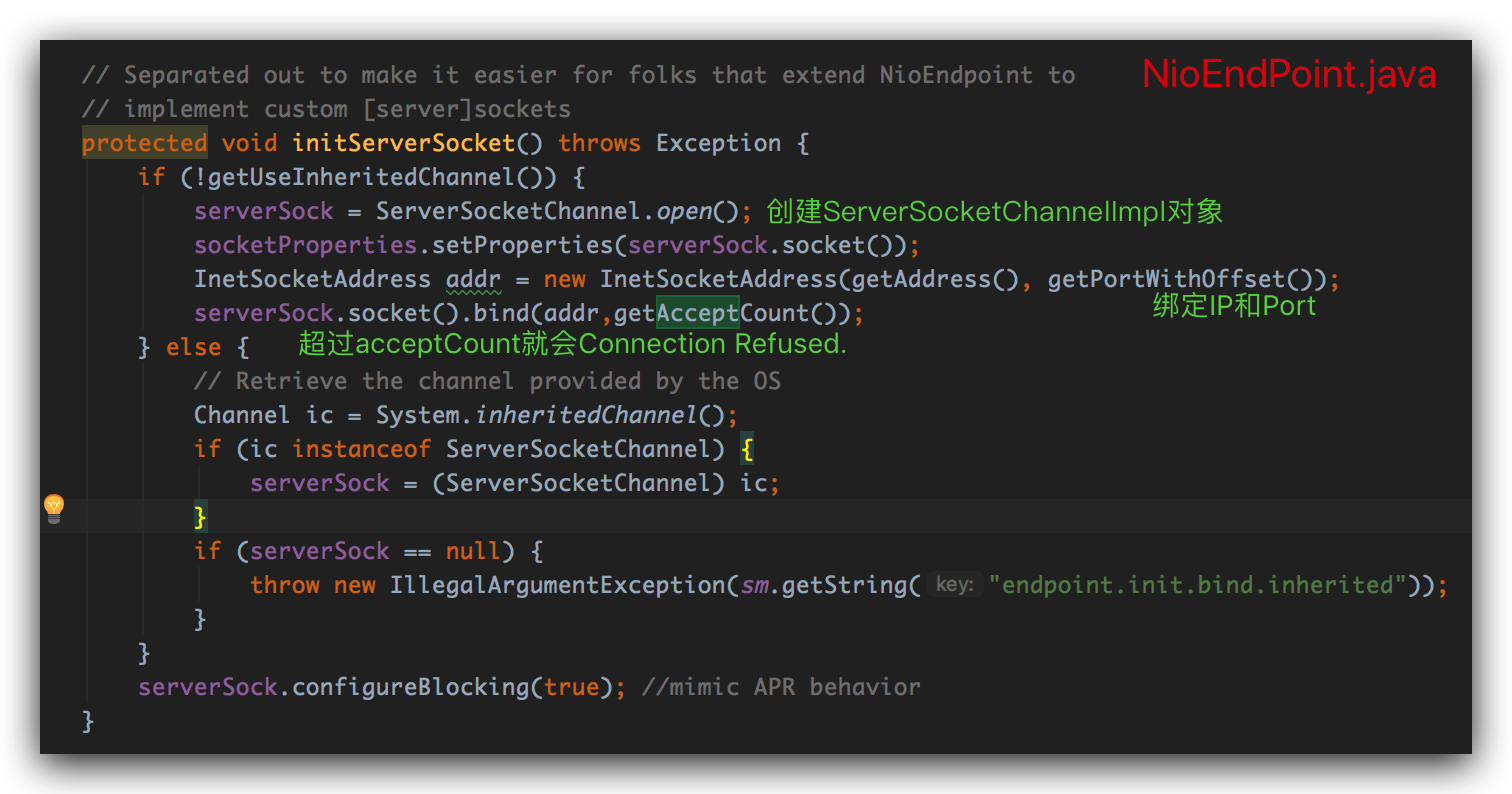
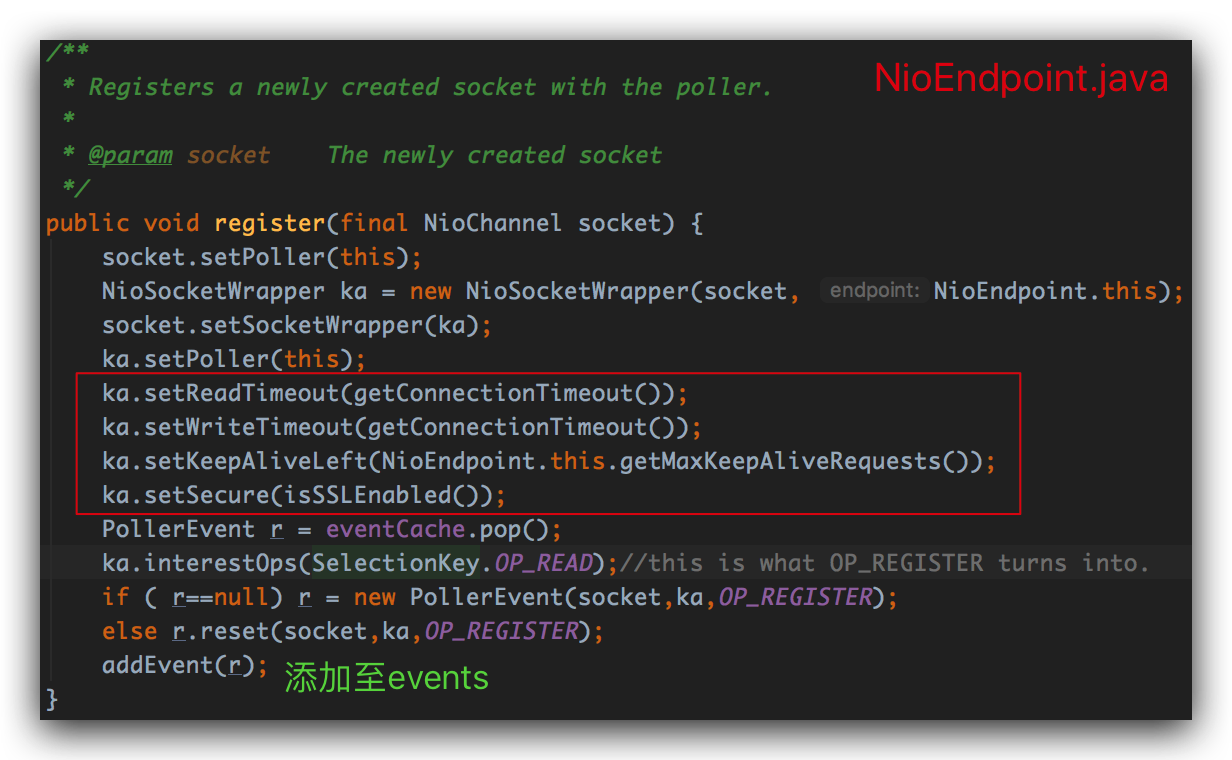
**StandardService**

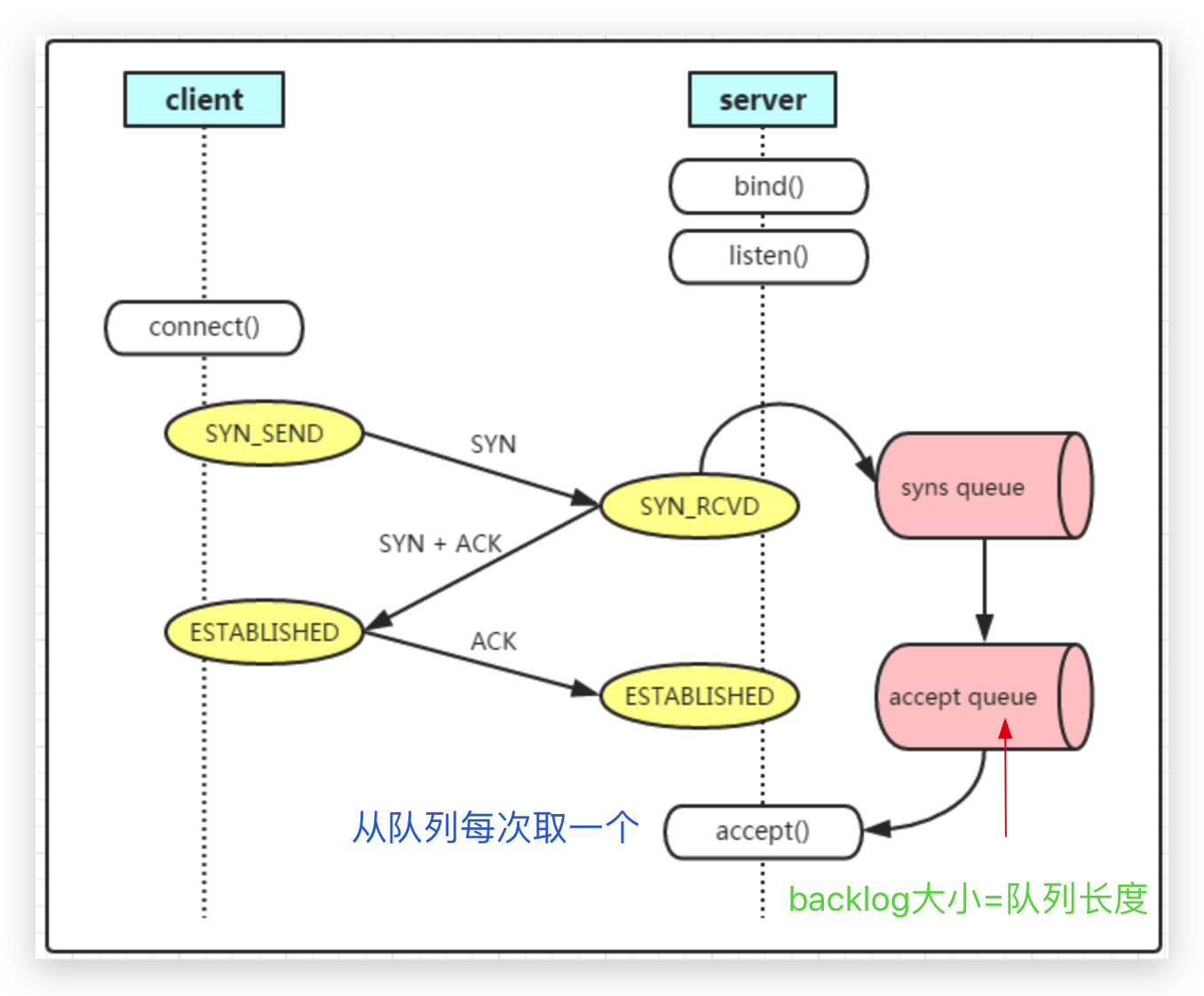
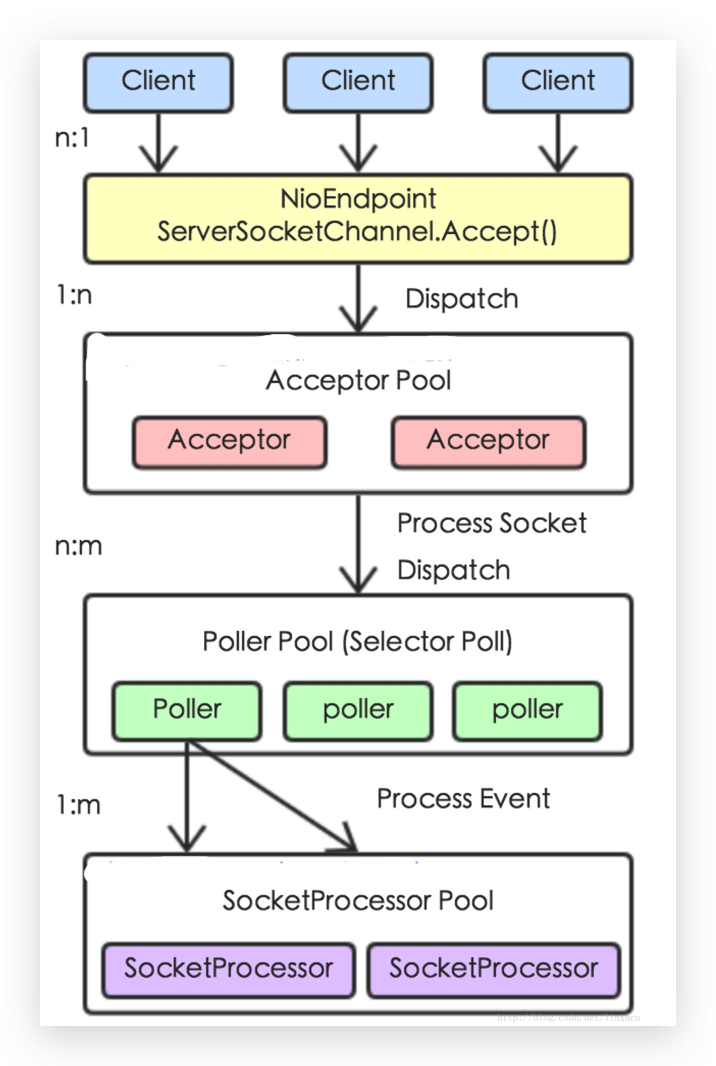
****

**Connector核心逻辑**

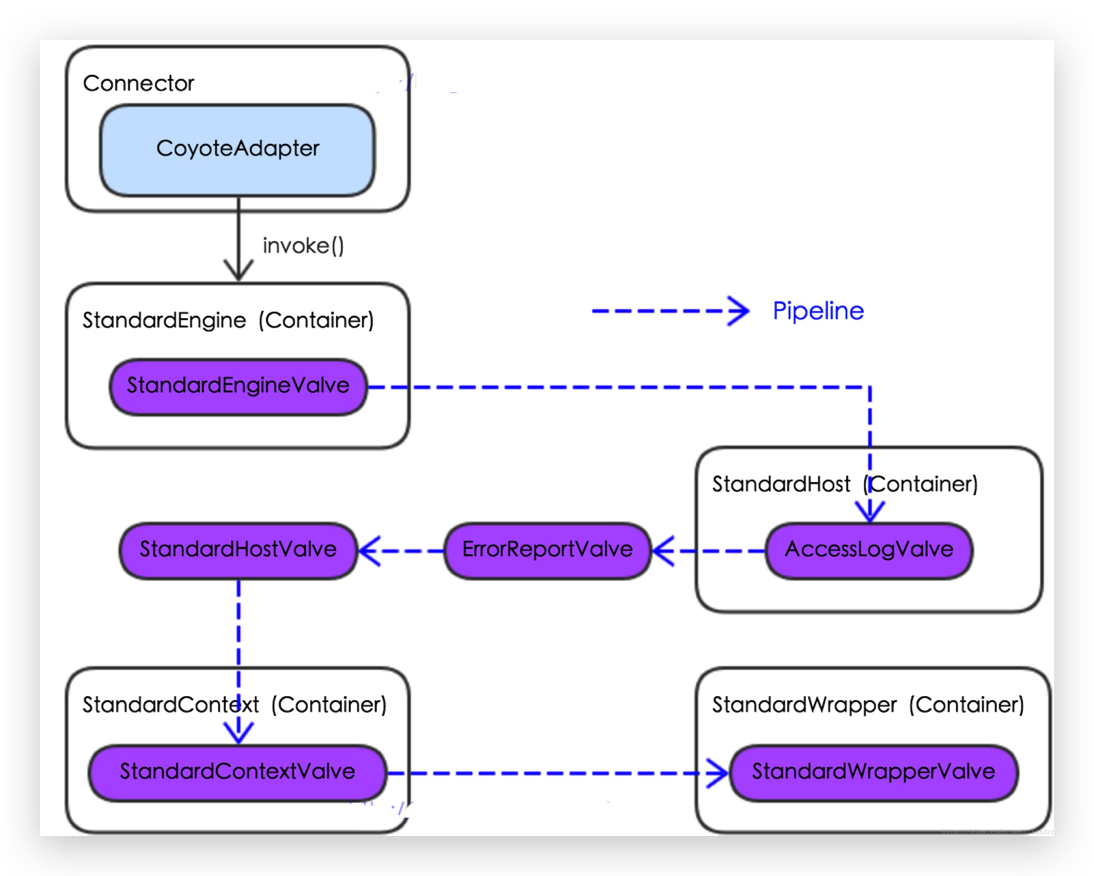
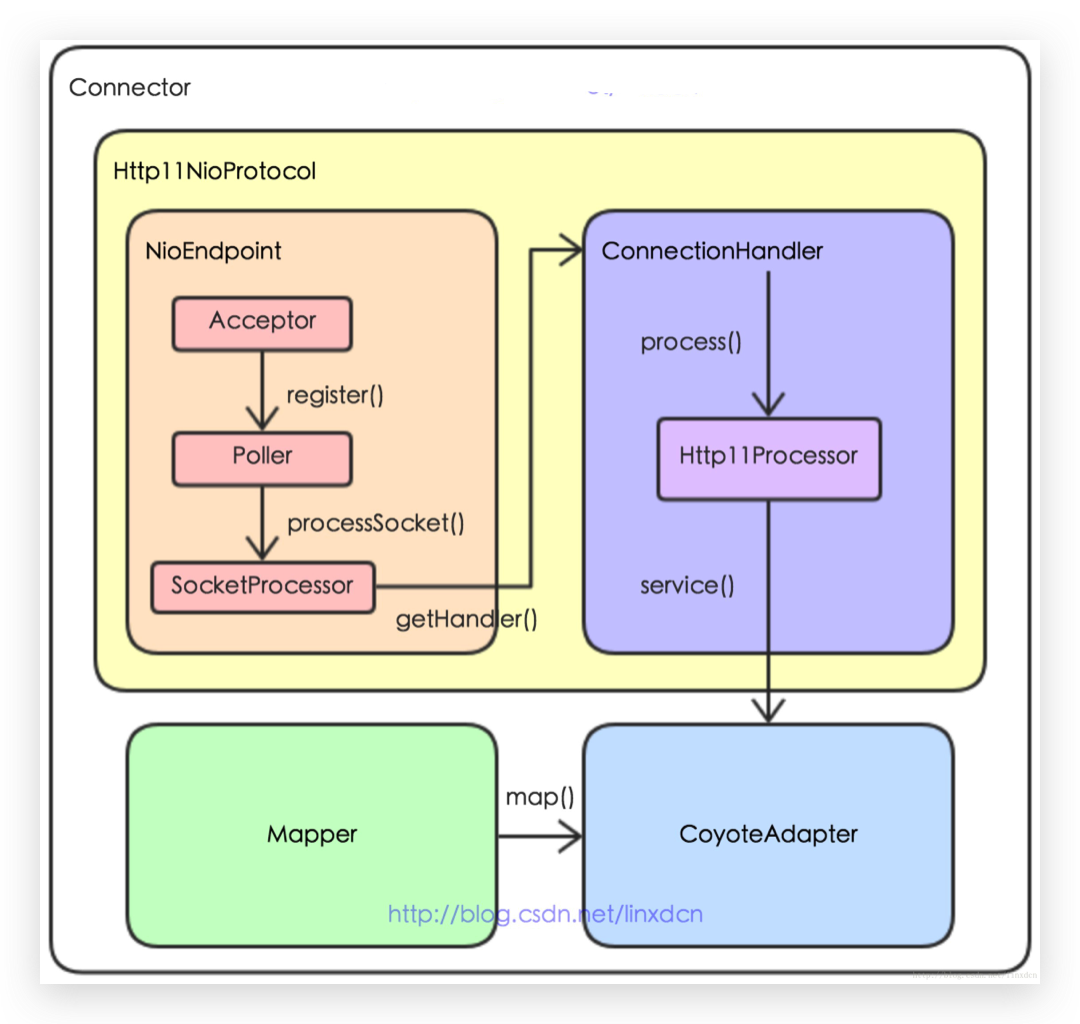
****

****

****

****

**请求-响应**

****

**StandardEngine**

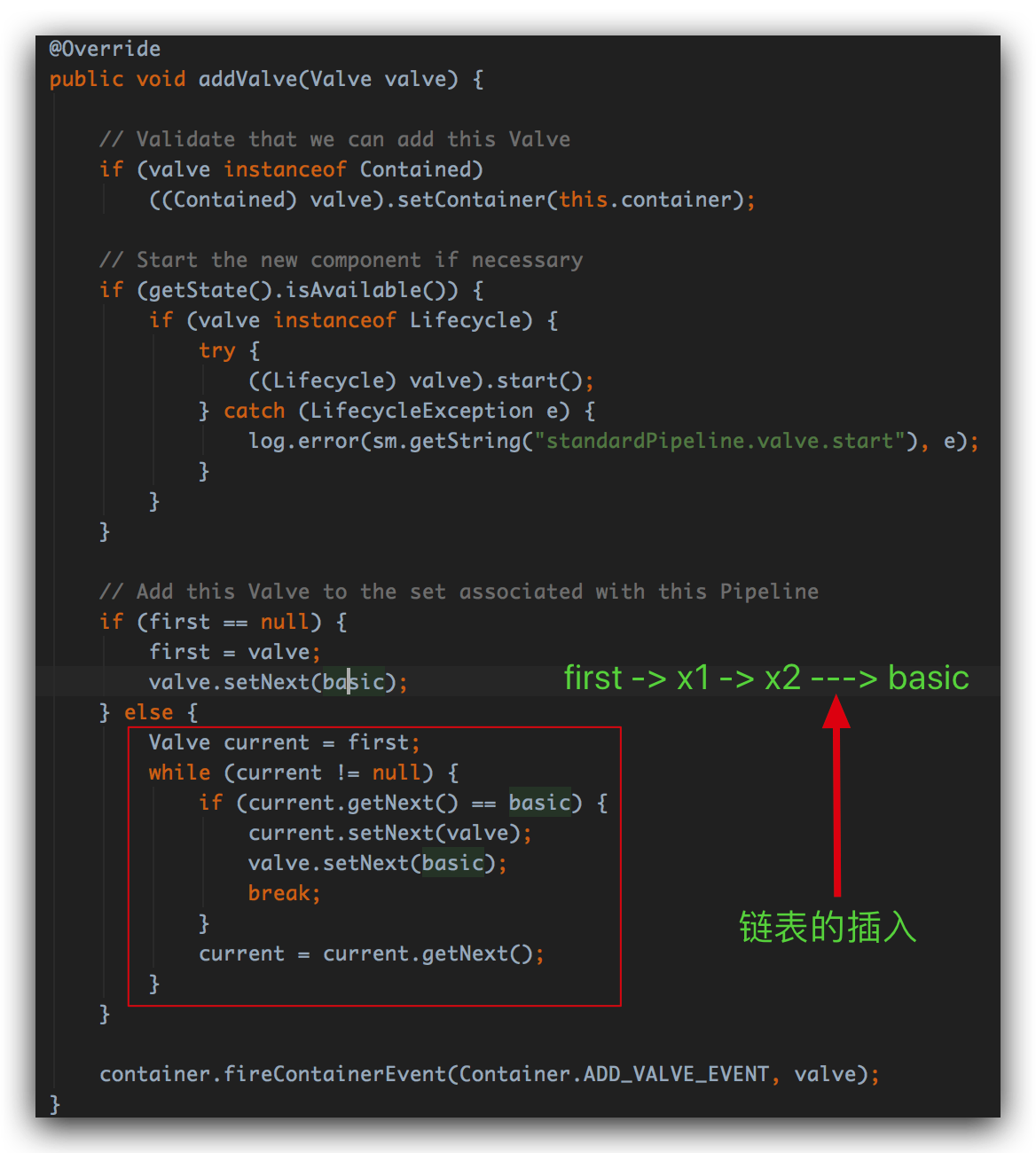
---------------------------------------------------------------------

**LifeCycle:** init() -> start() -> stop() -> destroy()



**ContainerBase** [Chain of Responsibility]

**StandardPipeline:** Valve的链表



**Valve:** 封装invoke()的链表节点

public void setNext(Valve valve);

public void invoke(Request request, Response response) //业务流程

throws IOException, ServletException;

public Valve getNext();

**Loader**

public ClassLoader getClassLoader(); //类加载器

public Context getContext(); //上下文

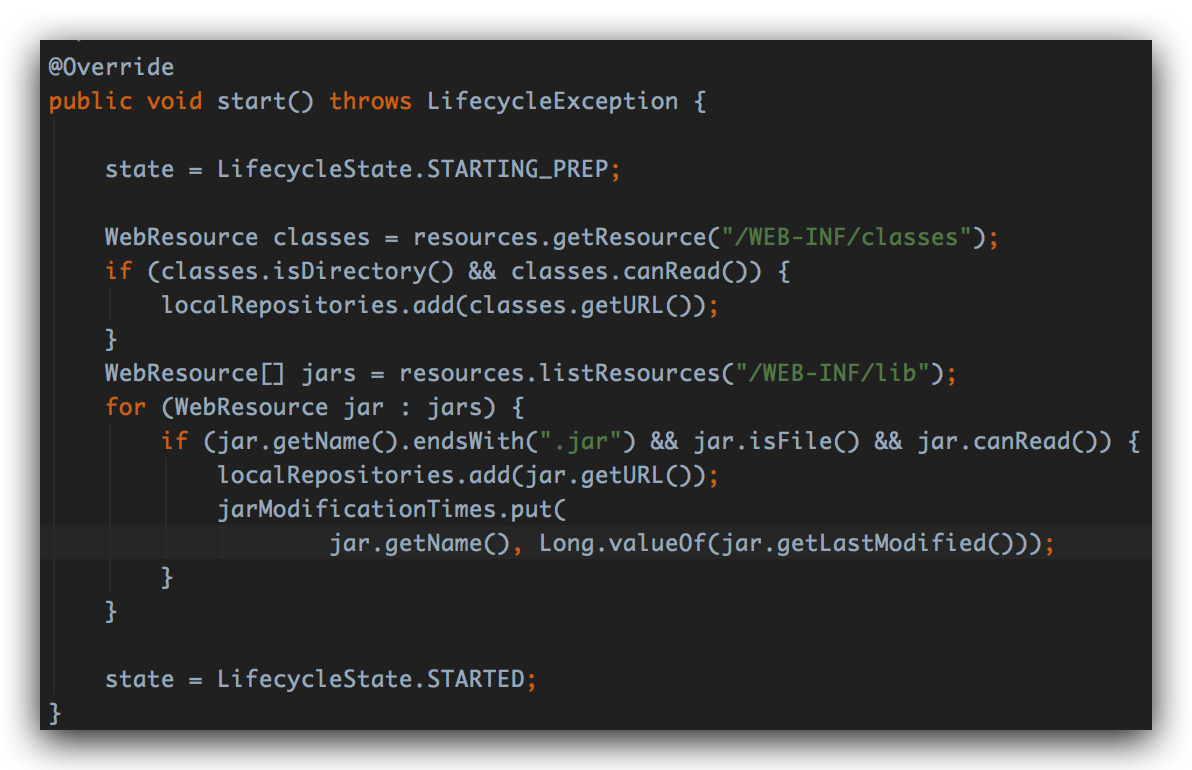
public boolean getDelegate(); //是否委托父加载器

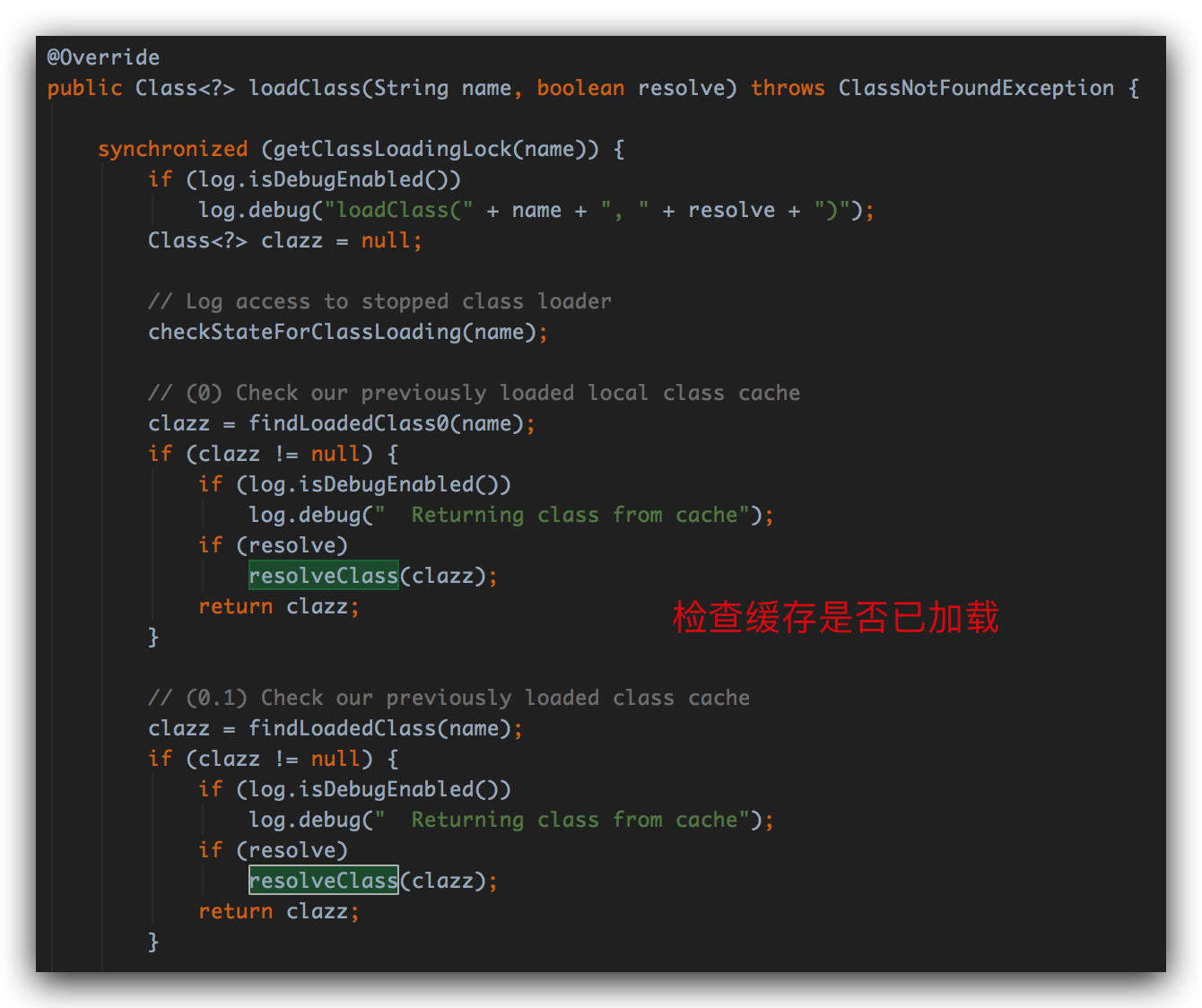
public boolean getReloadable(); //是否自动重载class

public boolean modified(); //检测目录变更的方法

**URLClassLoader** is used to load classes and resources from a search path of URLs referring to both JAR files and directories.

**WebappClassLoaderBase** extends URLClassLoader





**Jetty**