基于Java Socket API搭建简单的HTTP客户端和服务器端程序

1. 项目简介

本项目基于 **Java Socket API（BIO）** 实现一个简易 HTTP 服务器和客户端，支持 GET、POST、302 重定向、304 缓存验证等。

我们是第18组，所以使用的是8018端口。

2. 功能列表

- ☑ HTTP 服务器端（BIO）
- ☑ 支持GET和POST请求
- ☑ 服务器、客户端均支持状态码：200、301、302、304、404、405、500，客户端可根据不同状态码做出响应
- ☑ 实现长连接 keep-alive
- ☑ MIME 类型支持text/plain、json、text/html、jpg、Location
- ☑ 实现注册，登录功能,数据存在内存中
- ☑ HTTP客户端可以发送请求报文、呈现响应报文,实现了GUI

---

3. 状态码测试说明

1. 200：正常
2. 301：永久重定向，如/moved 是一个示例接口，勾选“自动跟随重定向”，会展示login界面（根目录），不勾选就会出现301
3. 302：GET:/login 选择不自动重定向，可以返回302
4. 304：使用缓存。示例：如果index.html内容不发生变化，就会不刷新index.html，出现304；如果index.html内容发生变化，就会刷新显示。
5. 401: 登录错误/POST:/login 401
6. 404：不合法请求，随便一个不正确的指令，示例：POST:register没有用户名和密码，会认为是未知的请求
7. 405：GET:/register 会报错405，因为register不允许GET，只支持POST请求
8. 500：输入“/bug”,弹出网络错误，因为在router里面写了这个命令，会故意抛出IOException，显示“Internal com Error”
9. 不开服务器，直接运行客户端：出现网络错误

4. 任务分配：

A: **郭佳荣**
B: **崔可喻**
C: **程心妍**
D: **熊雅琪**

角色	负责内容	产出文件

角色	负责内容	产出文件
A：服务器基础 & I/O框架	建立ServerSocket、接收连接、读取请求、实现长连接机制	com.server.HttpServer.java com.server.ConnectionHandler.java
B：HTTP协议解析 & 响应构造	解析请求行/头/体；构造响应行/头；处理MIME & 状态码（200/301/302/304/404/405/500）	com.http.HttpRequestParser.java com.http.HttpResponse.java com.http.MimeType.java
C：业务接口（注册/登录）与路由	HashMap存用户；实现/register /login POST处理；实现重定向；路由路径分发	com.http.Router.java com.model.UserService.java com.controller.UserController.java
D：客户端 + 演示	用Socket手写GET/POST请求，打印响应，支持302自动重定向；编写最终报告 + 演示流程	Client.HttpClient.java

5. 运行说明

1. 先运行`com.Main`的`main()`方法，启动服务器，监听8018端口
2. 再运行`Client.HttpClient`的`main()`方法，启动客户端GUI
3. 在客户端页面可以实现UI和命令的自行输入设置，并查看状态码响应情况
4. 同时支持在浏览器中进行注册和登录操作，在浏览器中输入“http://localhost:8080”进入网页
5. 服务器端控制台会打印请求和响应日志，客户端GUI会显示响应
6. 长连接的验证可以通过查看UI界面下方的Connection信息以及服务器打印的响应日志来验证。

---

6. 注意事项

- 服务器端和客户端需要分别运行在不同的进程中
- 确保8080端口未被占用
- 如果需要测试不同的状态码，可以通过修改请求路径或参数来触发相应的处理逻辑

---

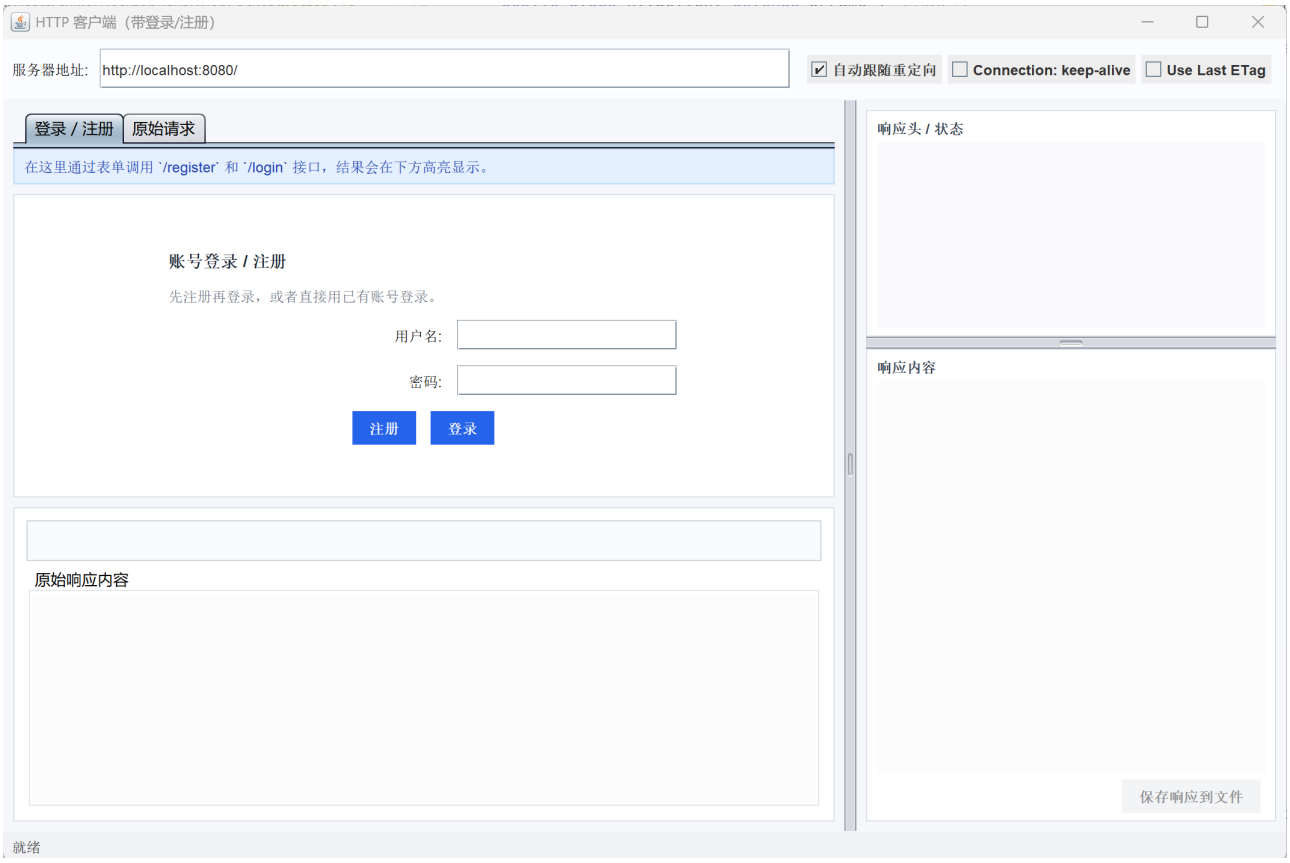
7. 项目结构

```
src
├── Client
│   └── HttpClient.java                  # 简单的 HTTP 客户端，用于测试 GET/POST、重
定向等功能
│
└── com
    ├── controller
    │   └── UserController.java          # 处理 /register 和 /login 的业务逻辑
    │
    ├── http
    │   ├── HttpRequestParser.java       # 解析原始 HTTP 报文 → 得到 HttpRequest 对象
    │   ├── HttpResponse.java            # 构造 HTTP 响应报文（状态行、头部、响应体）
    │   ├── Router.java                  # 根据路径选择交给哪个 Controller 或静态文件处
理
    │   └── MimeType.java                # 根据文件扩展名返回 MIME 类型
```

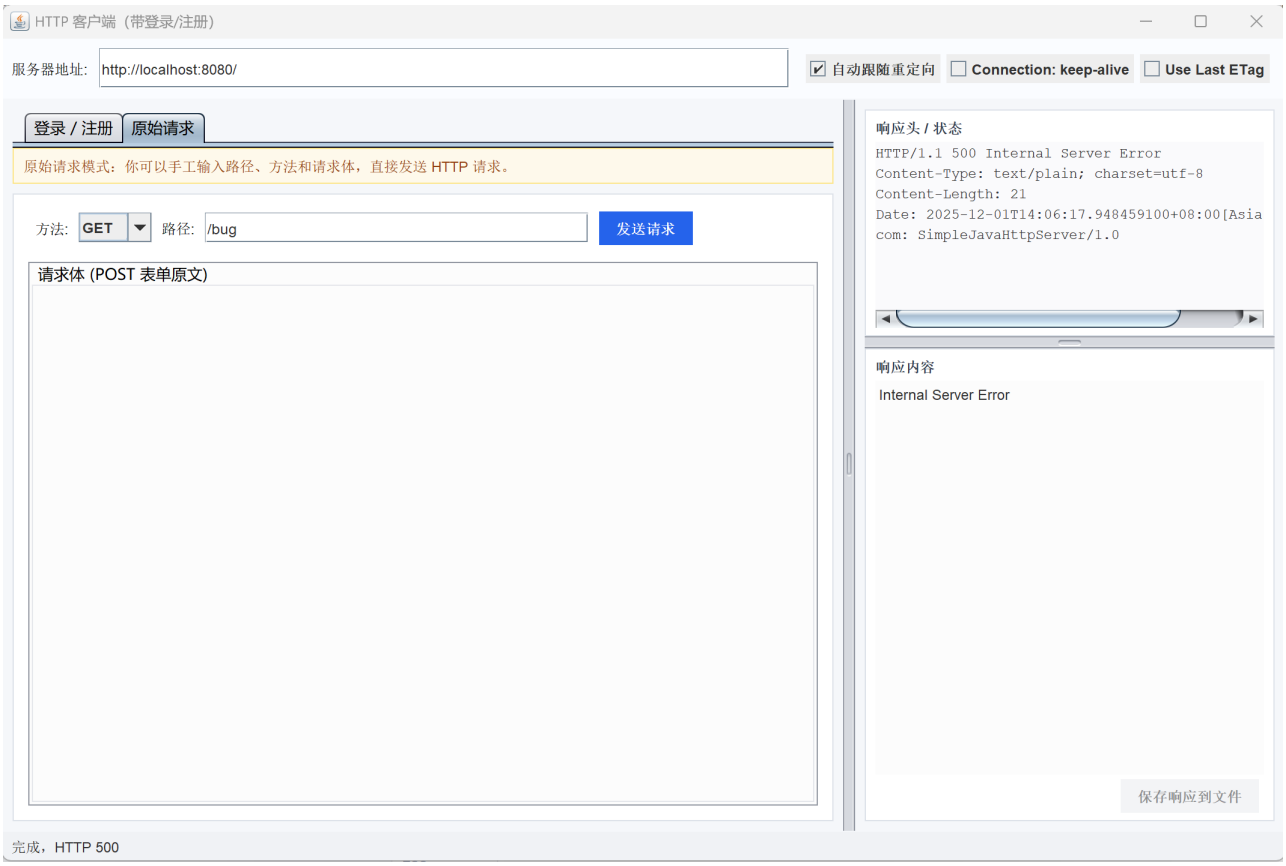
```
  (text/html,image/png等)
    │
    ├── model
    │   └── UserService.java            # 处理用户数据（注册/登录），数据存储在内存中
    │
    └── server
    │   ├── ConnectionHandler.java      # 一个线程处理一个 Socket 连接（BIO），实现
  Keep-Alive
    │   ├── HttpServer.java             # 启动服务器、监听端口、等待客户端连接
    │   └── StaticFileHandler.java      # 处理静态文件请求（如 HTML/CSS/JS）
    │
    └── Main.java                           # 程序入口，启动 HttpServer
```

8. 运行截图

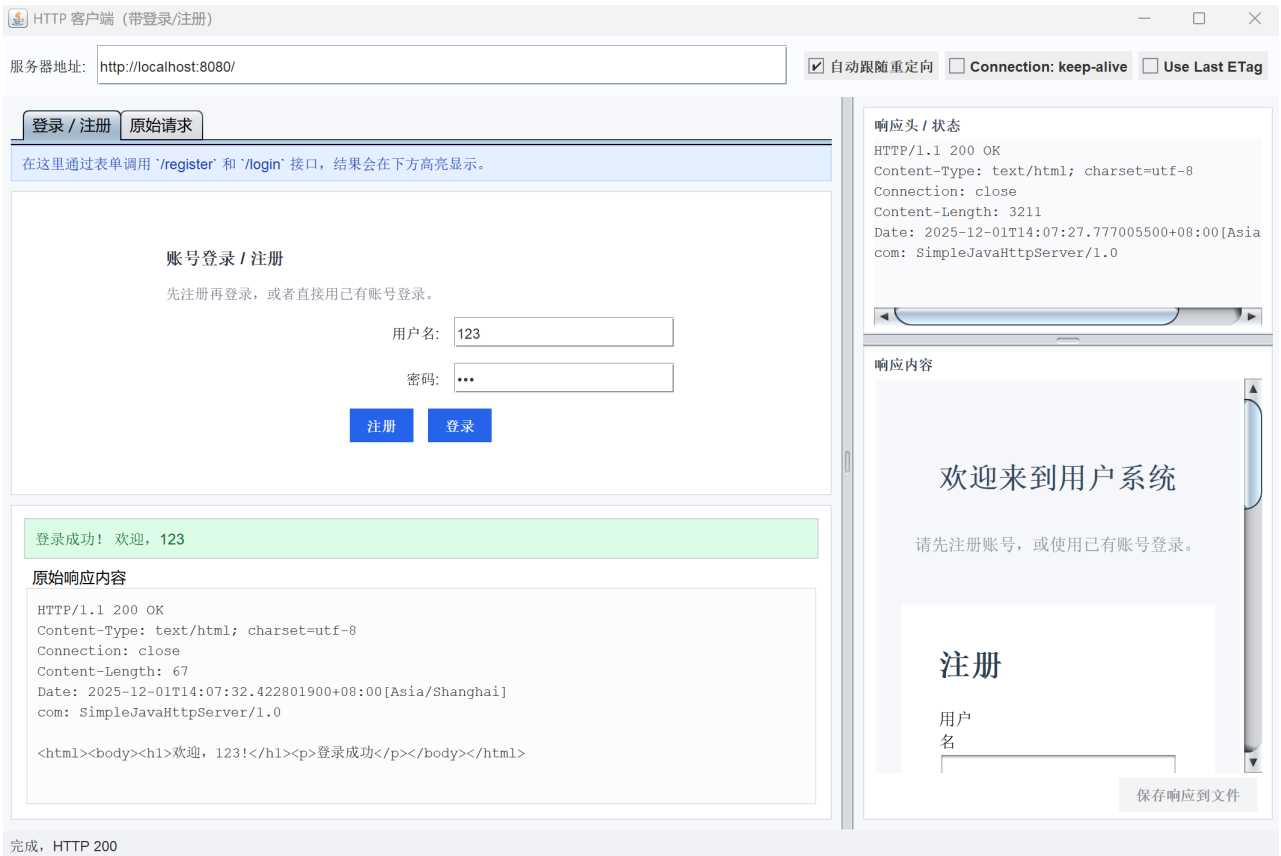
UI交互注册登录页面：



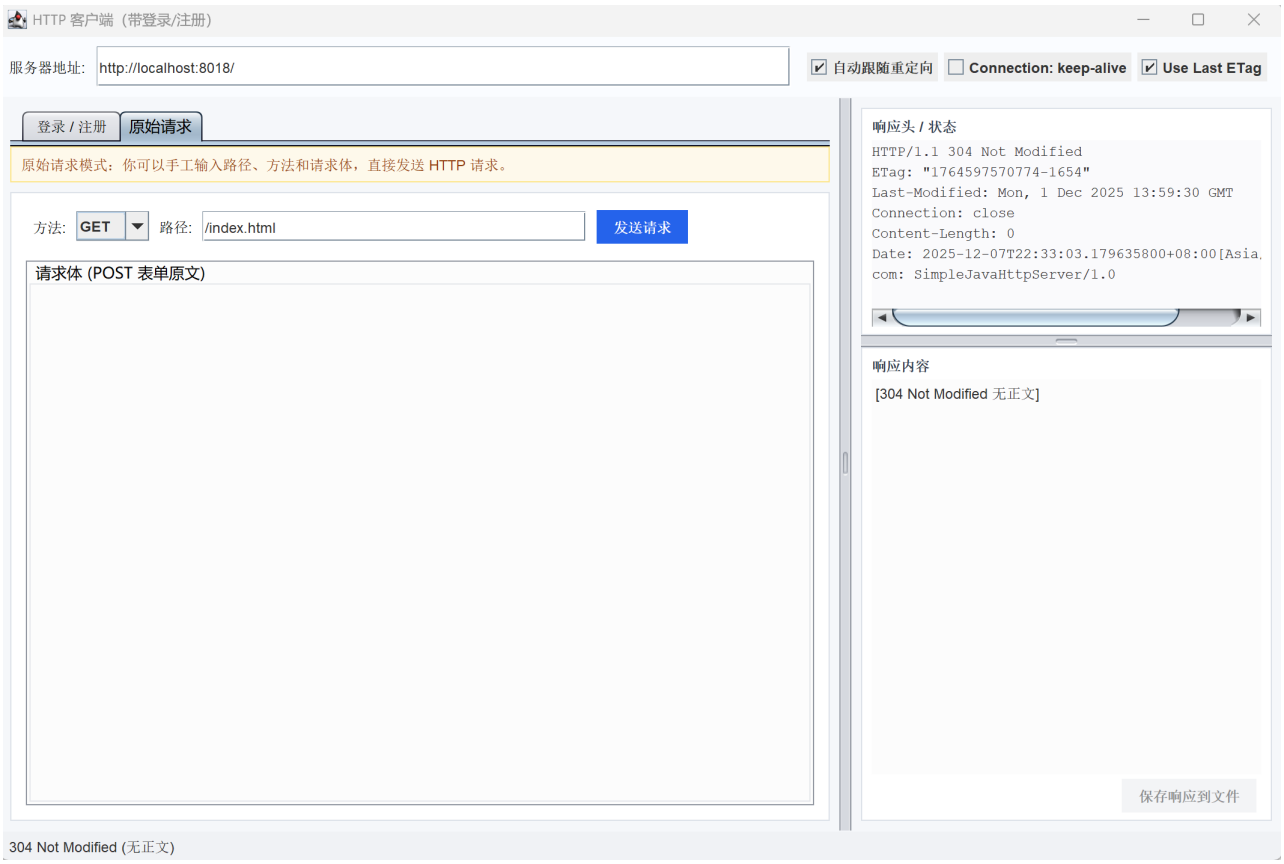
UI路径与参数输入页面：


HTTP 客户端（带登录/注册）
服务器地址: http://localhost:8080/
自动跟随重定向
Connection: keep-alive
Use Last ETag
登录 / 注册
原始请求
原始请求模式：你可以手工输入路径、方法和请求体，直接发送 HTTP 请求。
方法: GET
路径: /bug
发送请求
请求体 (POST 表单原文)
响应头 / 状态
HTTP/1.1 500 Internal Server Error
Content-Type: text/plain; charset=utf-8
Content-Length: 21
Date: 2025-12-01T14:06:17.948459100+08:00[Asia
com: SimpleJavaHttpServer/1.0
响应内容
Internal Server Error
保存响应到文件
完成，HTTP 500

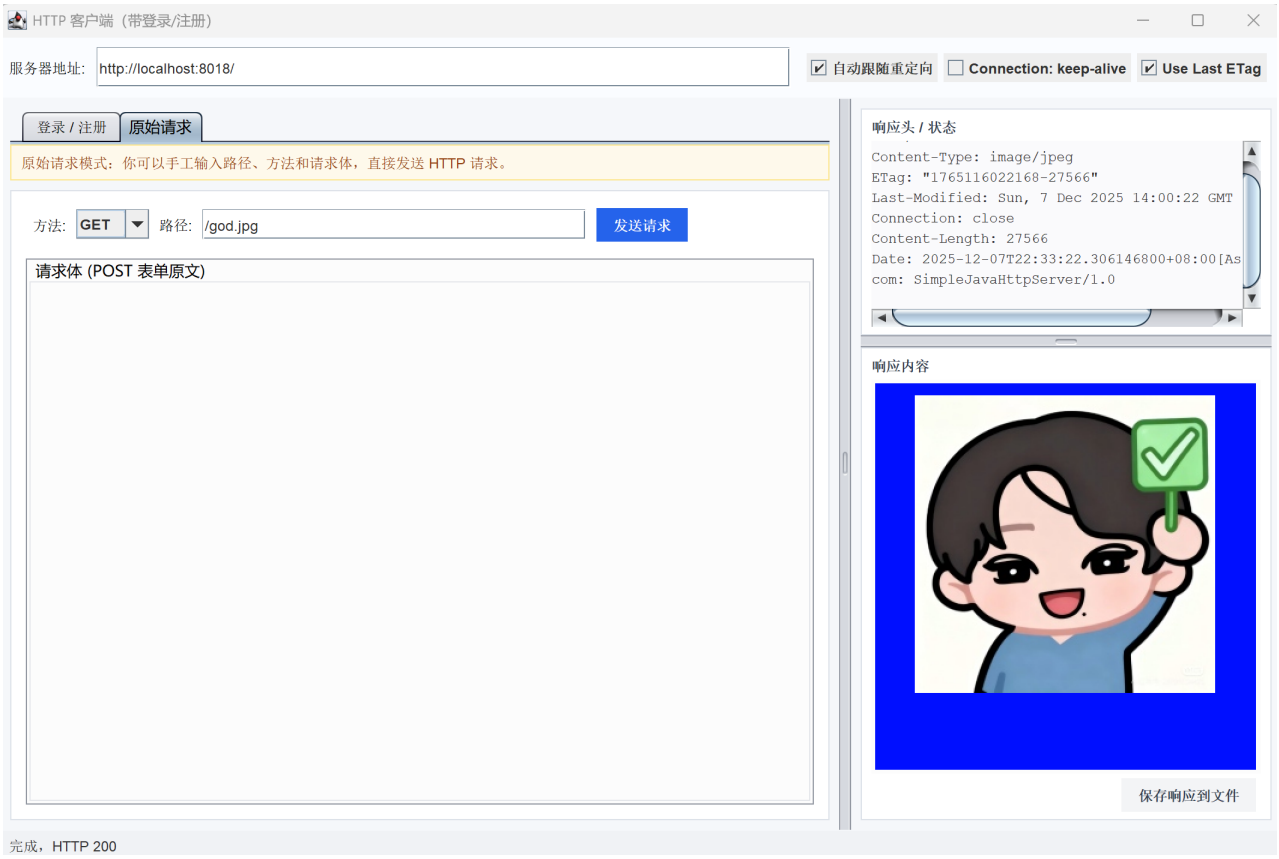
示例：登录成功页面


HTTP 客户端（带登录/注册）
服务器地址: http://localhost:8080/
自动跟随重定向
Connection: keep-alive
Use Last ETag
登录 / 注册
原始请求
在这里通过表单调用 `/register` 和 `/login` 接口，结果会在下方高亮显示。
账号登录 / 注册
先注册再登录，或者直接用已有账号登录。
用户名: 123
密码: •••
注册
登录
登录成功！ 欢迎，123
原始响应内容
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 67
Date: 2025-12-01T14:07:32.422801900+08:00[Asia/Shanghai]
com: SimpleJavaHttpServer/1.0
<html><body><h1>欢迎，123!</h1> 登录成功 </body></html>
响应头 / 状态
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 3211
Date: 2025-12-01T14:07:27.777005500+08:00[Asia
com: SimpleJavaHttpServer/1.0
响应内容
欢迎来到用户系统
请先注册账号，或使用已有账号登录。
注册
用户名
保存响应到文件
完成，HTTP 200

html文件处理与显示页面：



jpg图片文件处理与显示页面：



运行环境

- java: JDK21、JDK11