

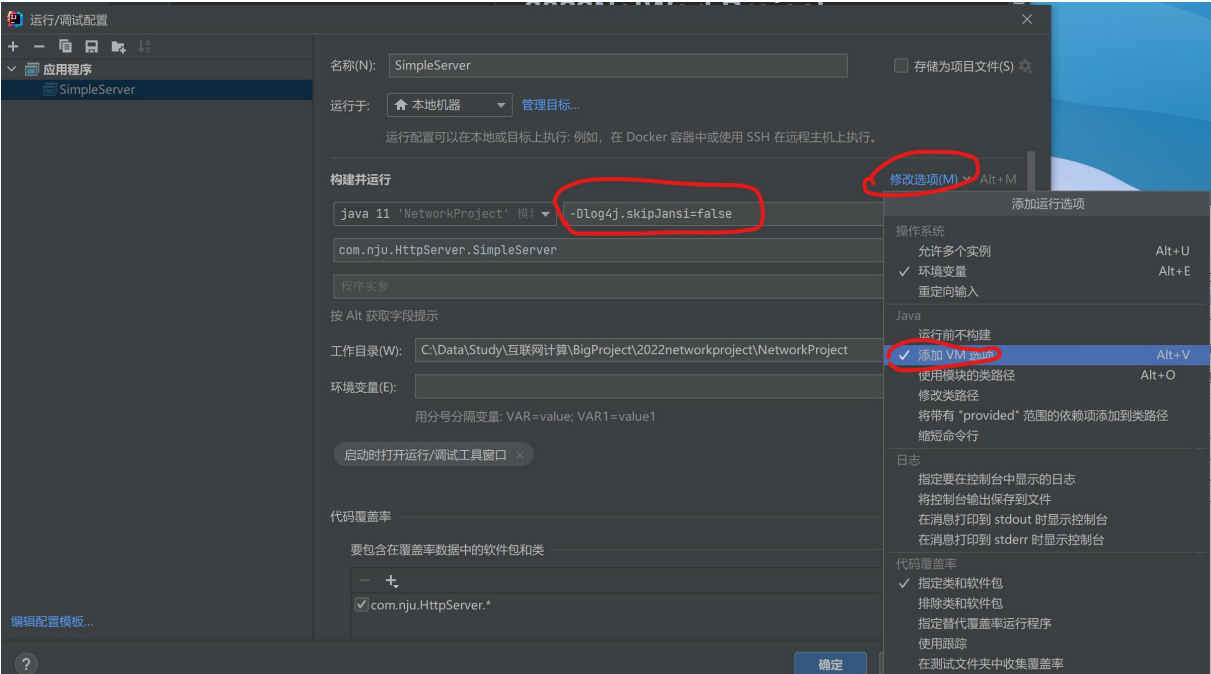
计网大作业说明文档（Team1）

组员	分工
江楠 201250033	Server
邓尤亮 201250035	Client
韩陈旭 201250037	Client
万沛沛 201250038（组长）	Server
华广松 201840309	Server

Before Start

本项目是基于 Maven 构建的，使用 IntelliJ IDEA 开发的 Java 项目。运行前，需通过 maven 安装相关依赖。请确保本项目存放于英文路径下。

本项目使用了 log4j2 记录日志，为保证控制台日志信息有颜色，请先设置 JVM 参数：`-Dlog4j.skipJansi=false`



服务端运行后，打开 <http://localhost:5000/>，经 301 重定向至 <http://localhost:5000/index.html>，出现一个带图片的注册登录页面，说明项目配置正常。

Http-Client

项目要求

实现基础的 HTTP 请求、响应功能，具体要求如下：

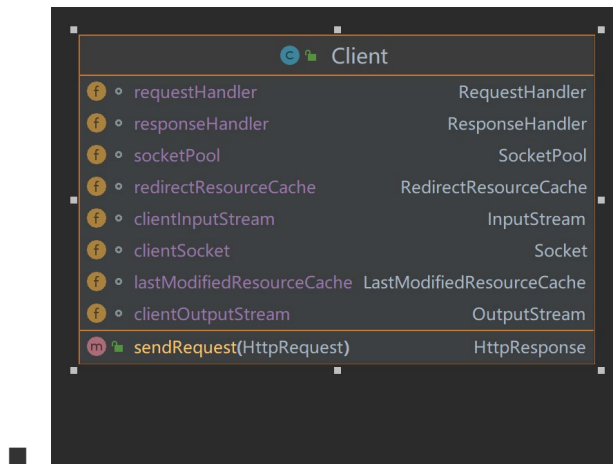
- HTTP 客户端可以发送请求报文、呈现响应报文（命令行形式）
- HTTP 客户端对 301、302、304 的状态码做相应的处理
- MIME 至少支持三种类型，包含一种非文本类型

项目简介

- 编写测试文件以发起客户端的 Http 请求
- Client 类使用 Socket 通信
- Componets 内包括 Http 报文的 Header, Body, 以及 Http 请求和 Http 回复报文
- Handler 包括对请求和回复报文的相关类，其中请求时会查找永久重定向与缓存资源，得到回复后处理各种状态码
- LocalCache 内包括对重定向和请求资源的缓存相关类

关键类与包的描述

- Client 类
 - 主要功能
 - 模拟客户端发送 http 请求，处理对应的 http 响应报文
 - 类图

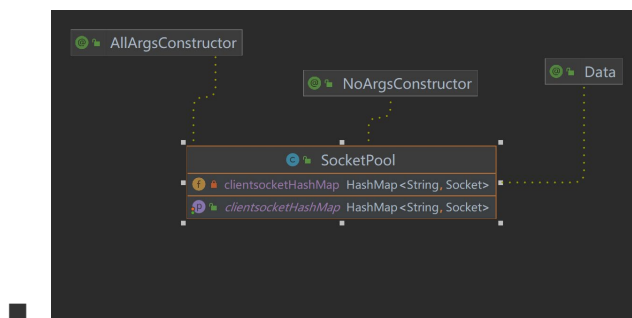


- SocketPool

- 主要功能

- 模拟 socket 连接池，主要用于存储长连接时没有关闭的连接，用于复用 socket 连接

- 类图

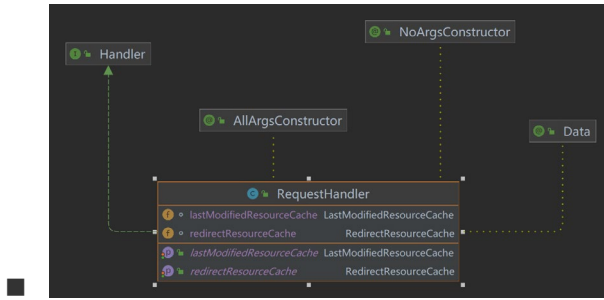


- RequestHandler

- 主要功能

- 用于重构请求报文

- 类图

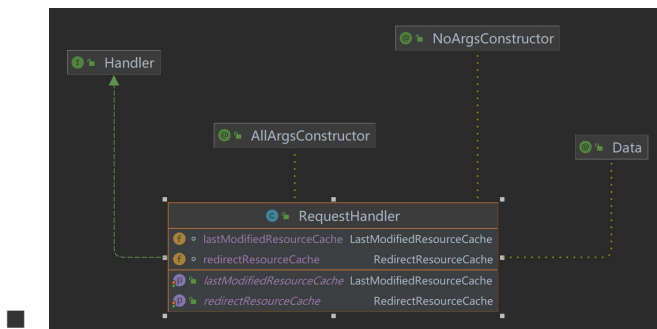


● ResponseHandler

○ 主要功能

- 用于对 http 响应报文进行处理，重点关注对 301、302、304 等响应码的处理

○ 类图

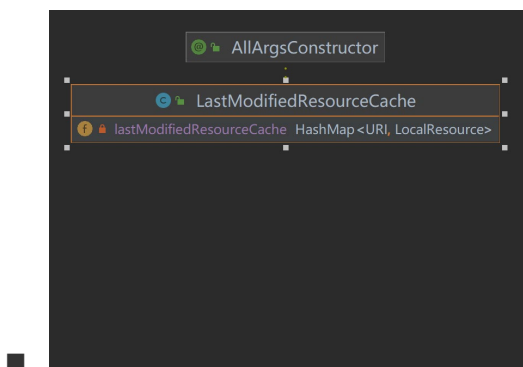


● LastModifiedResourceCache

○ 主要功能

- 主要辅助请求报文在请求头里面添加 If-Modified-Since

○ 类图

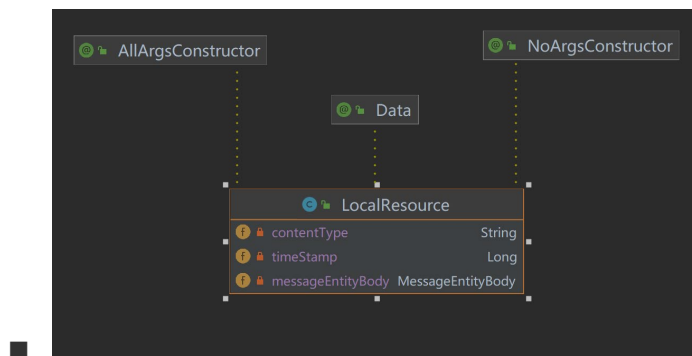


- LocalResource

- 主要功能

- 表示存储在本地缓存中的资源

- 类图

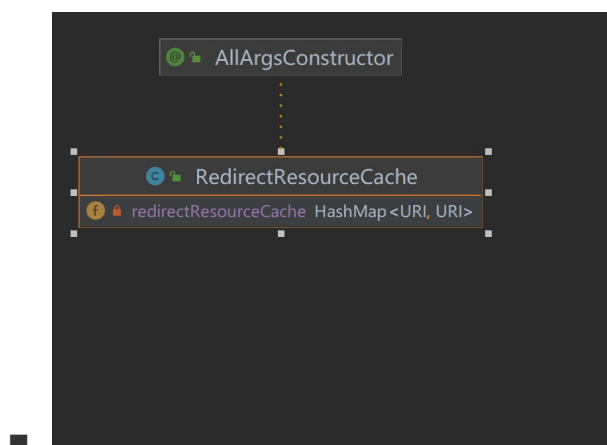


- RedirectResourceCache

- 主要功能

- 存放重定向的资源

- 类图

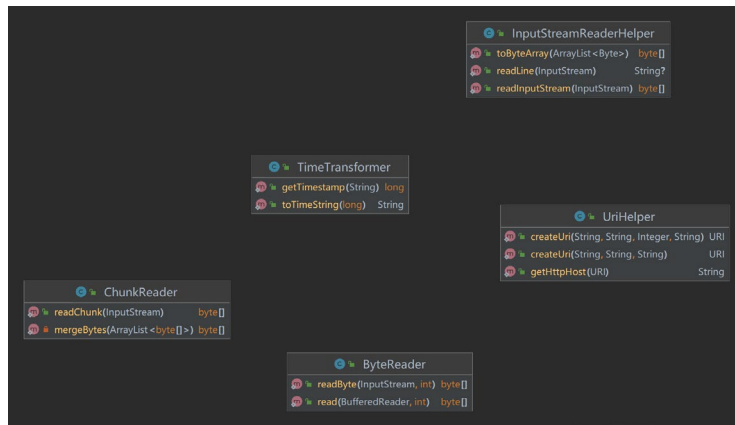


- Utils 包

- 主要功能

- 开发时的工具包，辅助进行文本的读取

○ 包图



运行测试

普通请求 200

首先编写测试文件

```
@Test
public void testMain() throws IOException {
    RequestLine requestLine = new RequestLine(Method.GET, requestURL: "/");
    MessageHeader messageHeader = new MessageHeader();
    messageHeader.putField(HeaderFields.Host, fieldValue: "www.baidu.com");
    messageHeader.putField(HeaderFields.Accept, fieldValue: "*/*");
    messageHeader.putField(HeaderFields.Accept_Encoding, fieldValue: "gzip, deflate, br");
    messageHeader.putField(HeaderFields.Connection, fieldValue: "keep-alive");
    MessageEntityBody messageBody = new MessageEntityBody();
    HttpRequest httpRequest = new HttpRequest(requestLine, messageHeader, messageBody);
    HttpResponse httpResponse = client.sendRequest(httpRequest);
    httpResponse.saveBody(path: path + "baidu.html");
}
```

由于未设定 UA，网站未启用 Cache，故此测试用例结果将保持状态码为 200

```
2022-06-13T23:15:41,839 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:56] Create connection
2022-06-13T23:15:43,543 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:65] Get return status code: 200
2022-06-13T23:15:43,543 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:66] Response header:
HTTP/1.1 200 OK
Bdpagetype: 1
Bdqid: 0xe94ee265006ad03
Cache-Control: private
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html;charset=utf-8
Date: Mon, 13 Jun 2022 15:15:42 GMT
Expires: Mon, 13 Jun 2022 15:15:42 GMT
P3p: CP=" OTI DSP COR IVA OUR IND COM "
Server: BWS/1.1
Set-Cookie: H_PS_PSSID=36552_36597_36454_31254_36511_36452_36420_36165_36570_36074_36520_36336_26350_36469_22160; path=/; domain=.baidu.com
Traceid: 1655133342270399489016811623382440652035
X-Frame-Options: sameorigin
X-UA-Compatible: IE=Edge,chrome=1
Transfer-Encoding: chunked
```

301 Moved Permanently

请求服务器名为 `"/movedIndex.html"` 文件时，服务器将返回状态码 301 并告知新的路径

```
2022-06-13T23:15:43,553 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:85] Get status code 301
2022-06-13T23:15:43,553 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:86] Response Line:
HTTP/1.1 301 301 Moved Permanently

2022-06-13T23:15:43,554 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:87] Response Header:
Keep-Alive: timeout=10
Content-Length: 173
Content-Type: text/html
Location: /index.html
```

此时服务器将保存此次重定向结果，下一次请求 `"/movedIndex.html"` 时直接更改路径

302 Found

请求服务器名为 `"/movedIndex2.html"` 文件时，服务器将返回状态码 302 并告知临时的新路径。此时客户端将重构此次请求，但不会缓存重定向结果。

```
2022-06-13T23:15:43,590 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:107] Get status code 302
2022-06-13T23:15:43,590 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:108] ResponseLine: HTTP/1.1 302 302 Found

2022-06-13T23:15:43,591 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:111] Get temporary redirect path: /index.html
2022-06-13T23:15:43,591 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:115] Send new http request
2022-06-13T23:15:43,591 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-06-13T23:15:43,591 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /index.html HTTP/1.1
Host: localhost:5000
Connection: keep-alive
content-length: 0
```

304 Not Modified

若请求服务器的同一个资源两次，第一次请求完毕后该资源和 **Last-Modified** 值将被保存至缓存中；第二次请求时发现本地缓存，加上 **If Modified Since** 字段后请求服务器。若服务器资源未改变，则返回 304 代码以告知客户端直接使用缓存资源。

```
2022-06-13T23:15:43,657 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:80] URI: http://localhost:5000/style.css
2022-06-13T23:15:43,659 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-06-13T23:15:43,659 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:88] Find local resource
2022-06-13T23:15:43,659 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:89] Add If-Modified-Since
2022-06-13T23:15:43,659 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /style.css HTTP/1.1
Host: localhost:5000
Connection: keep-alive
content-length: 0
If-Modified-Since: Mon, 13 Jun 2022 01:17:08 GMT

2022-06-13T23:15:43,659 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:44] Reuse connection
2022-06-13T23:15:43,662 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:122] Get status code 304
2022-06-13T23:15:43,663 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:123] ResponseLine: HTTP/1.1 304 304 Not Modified
```

内容编码 gzip

支持内容编码 `gzip`，可参考第一个测试

MIME

在 Test302 测试文件中已覆盖超过三种 MIME 支持的测试

text/html

```
2022-07-01T18:12:33,010 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:94] Add redirect resource cache,
  URI: http://localhost:5000/index.html
2022-07-01T18:12:33,010 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:101] Send new http request
2022-07-01T18:12:33,010 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-07-01T18:12:33,010 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /index.html HTTP/1.1
Host: localhost:5000
Connection: keep-alive
content-length: 0

2022-07-01T18:12:33,010 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:44] Reuse connection
2022-07-01T18:12:33,026 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:65] Get return status code: 200
2022-07-01T18:12:33,026 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:66] Response header:
HTTP/1.1 200 OK
Keep-Alive: timeout=10
Last-Modified: Sun, 26 Jun 2022 02:06:06 GMT
Content-Length: 757
Content-Type: text/html
```

text/css

```
2022-07-01T17:02:55,587 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-07-01T17:02:55,587 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /style.css HTTP/1.1
Host: localhost:5000
Connection: keep-alive
content-length: 0

2022-07-01T17:02:55,587 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:56] Create connection
2022-07-01T17:02:55,603 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:65] Get return status code: 200
2022-07-01T17:02:55,603 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:66] Response header:
HTTP/1.1 200 OK
Keep-Alive: timeout=10
Last-Modified: Sun, 26 Jun 2022 02:06:06 GMT
Content-Length: 43
Content-Type: text/css
```


image/png

```
2022-07-01T17:02:55,550 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:111] Get temporary redirect
path: /pic.png
2022-07-01T17:02:55,550 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:115] Send new http request
2022-07-01T17:02:55,550 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-07-01T17:02:55,550 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /pic.png HTTP/1.1
Host: localhost:5000
Connection: keep-alive
content-length: 0

2022-07-01T17:02:55,550 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:44] Reuse connection
2022-07-01T17:02:55,550 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:65] Get return status code: 200
2022-07-01T17:02:55,550 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:66] Response header:
HTTP/1.1 200 OK
Keep-Alive: timeout=10
Last-Modified: Sun, 26 Jun 2022 02:06:06 GMT
Content-Length: 134360
Content-Type: image/png
```

image/jpeg

```
2022-07-01T17:02:55,565 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:111] Get temporary redirect
path: /pic.jpg
2022-07-01T17:02:55,565 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:115] Send new http request
2022-07-01T17:02:55,565 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-07-01T17:02:55,565 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /pic.jpg HTTP/1.1
Host: localhost:5000
Connection: keep-alive
content-length: 0

2022-07-01T17:02:55,565 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:44] Reuse connection
2022-07-01T17:02:55,581 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:65] Get return status code: 200
2022-07-01T17:02:55,587 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:66] Response header:
HTTP/1.1 200 OK
Keep-Alive: timeout=10
Last-Modified: Sun, 26 Jun 2022 02:06:06 GMT
Content-Length: 2154105
Content-Type: image/jpeg
```

长连接

测试的逻辑是前三次请求不使用长连接，后两次请求使用长连接，socket 连接只被创建 4 次，最后一次的连接复用第四次连接创建的 socket。

测试代码如下：

```

@Test
public void testKeepAlive() {
    sendRequest( path: "/favicon.ico", enableAlive: false, savePath: path + "favicon.ico");
    sendRequest( path: "/favicon.ico", enableAlive: false, savePath: path + "favicon.ico");
    sendRequest( path: "/favicon.ico", enableAlive: false, savePath: path + "favicon.ico");
    sendRequest( path: "/favicon.ico", enableAlive: true, savePath: path + "favicon.ico");
    sendRequest( path: "/favicon.ico", enableAlive: true, savePath: path + "favicon.ico");
}

```

测试结果如下：

```

2022-07-01T18:49:30,770 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:56] Create connection
2022-07-01T18:49:30,770 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:65] Get return status code: 200
2022-07-01T18:49:30,770 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:66] Response header:
HTTP/1.1 200 OK
Last-Modified: Sun, 26 Jun 2022 02:06:06 GMT
Content-Length: 4286
Content-Type: image/x-icon

2022-07-01T18:49:30,770 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:79] Save local resource
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:80] URI: http://localhost:5000/favicon.ico
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:88] Find local resource
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:89] Add If-Modified-Since
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /favicon.ico HTTP/1.1
Host: localhost:5000
Content-Length: 0
If-Modified-Since: Sun, 26 Jun 2022 02:06:06 GMT

2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:56] Create connection
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:122] Get status code 304
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:123] ResponseLine: HTTP/1.1 304 304 Not Modified

2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:88] Find local resource
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:89] Add If-Modified-Since
2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /favicon.ico HTTP/1.1
Host: localhost:5000
Content-Length: 0
If-Modified-Since: Sun, 26 Jun 2022 02:06:06 GMT

2022-07-01T18:49:30,780 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:56] Create connection
2022-07-01T18:49:30,790 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:122] Get status code 304
2022-07-01T18:49:30,790 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:123] ResponseLine: HTTP/1.1 304 304 Not Modified

2022-07-01T18:49:30,790 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-07-01T18:49:30,790 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:88] Find local resource
2022-07-01T18:49:30,790 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:89] Add If-Modified-Since
2022-07-01T18:49:30,790 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /favicon.ico HTTP/1.1
Host: localhost:5000
Connection: keep-alive

```

```
2022-07-01T18:49:30,790 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:56] Create connection
2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:122] Get status code 304
2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:123] ResponseLine: HTTP/1.1 304
304 Not Modified

2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:47] Add content-length
2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:88] Find local resource
2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:89] Add If-Modified-Since
2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.RequestHandler:54] Request message:
GET /favicon.ico HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Content-Length: 0
content-length: 0
If-Modified-Since: Sun, 26 Jun 2022 02:06:06 GMT

2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Clien.SocketPool:44] Reuse connection
2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:122] Get status code 304
2022-07-01T18:49:30,801 DEBUG [LOGID:] [main] [com.nju.HttpC.Handl.ResponseHandler:123] ResponseLine: HTTP/1.1 304
304 Not Modified
```

从测试结果可以看到有四次"Create connection",有一次"Reuse connection"

Http-Server

项目要求

实现基础的 HTTP 请求、响应功能，具体要求如下：

- HTTP 服务器端支持 GET 和 POST 请求
- HTTP 服务器端支持 200、301、302、304、404、405、500 的状态码
- HTTP 服务器端实现长连接
- MIME 至少支持三种类型，包含一种非文本类型

项目简介

SimpleServer 类负责启动主线程 ServerHandler 以监听请求。监听到连接请求，回调 AcceptHandler，AcceptHandler 回调 RequestHandler 处理 Http 信息。

Http 包负责封装请求和响应。

Controller 包负责匹配路径并分发指令（如 GET 静态资源等），其中 RequestMapper

是仿 Springboot 控制器风格的 uri 匹配器，内含的 Router 包是 uri 匹配器和路由的具体实现。

Services 包负责处理静态资源和业务逻辑，生成 http 响应。

Common 包存放状态码等枚举。StaticResouces 文件夹下放了一个演示网页。

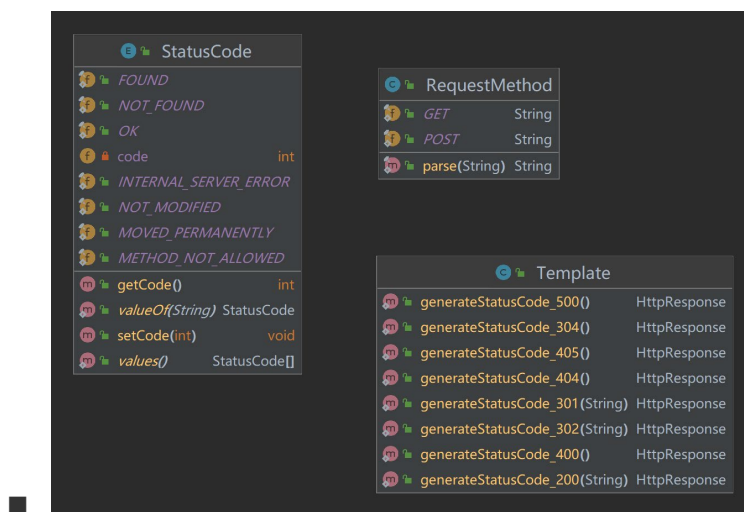
关键类与包的描述

- Common 包

- 主要功能

- 定义状态码、请求方法、响应模板

- 包图

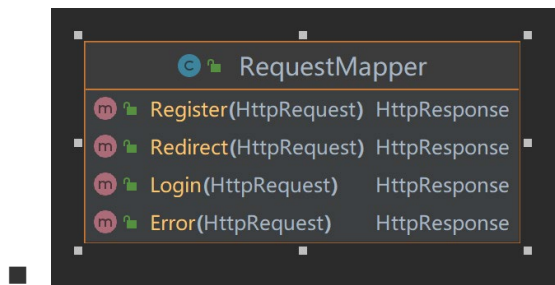


- RequestMapper

- 主要功能

- 类 Springboot 控制器风格的注解匹配器，将对应的请求精准匹配到相应的方法

- 类图



○ 匹配和路由的实现在 Router 包内

```

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface RequestMapping {

    5 个用法  🧑 HC_Plantern
    String uri(); // 注解需要声明路由的uri

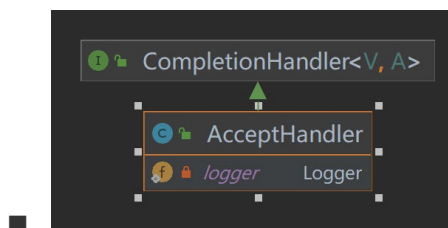
    5 个用法  🧑 HC_Plantern
    String method(); // 注解需要声明路由的请求方法
}
  
```

● AcceptHandler

○ 主要功能

- 连接成功的处理类，使服务端能够继续监听客户端连接请求，实现异步非阻塞

○ 类图

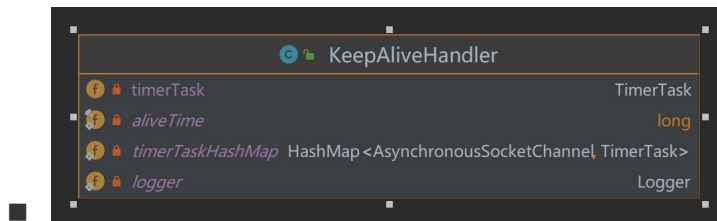


● KeepAliveHandler

○ 主要功能

- 用于处理长连接

○ 类图



- 实现：使用 TimerTask 实现 channel 的定时关闭，使用 HashMap 确保每个 channel 的 TimerTask 是唯一的、最新的。

```

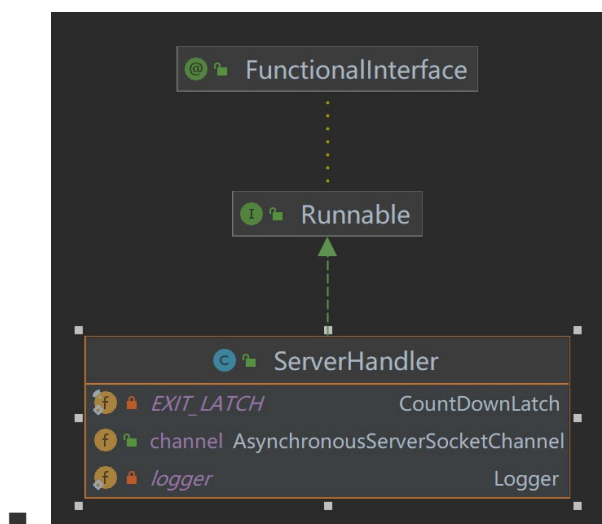
private TimerTask timerTask = null; // 定时器
2 个用法
private static final long aliveTime = 10L; // 响应给客户端的 keep-alive 的时间，单位：秒
4 个用法
private static HashMap<AsynchronousSocketChannel, TimerTask> timerTaskHashMap = new HashMap<>();
  
```

- ServerHandler

- 主要功能

- 用于开启服务端线程

- 类图

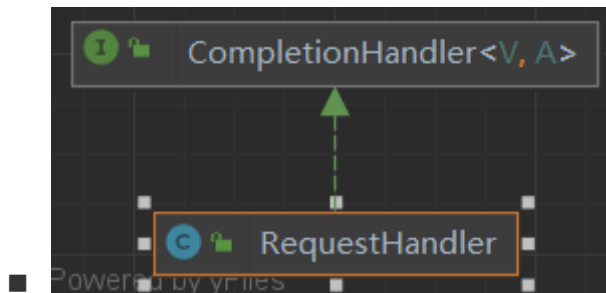


- RequestHandler

- 主要功能

- 实现 channel 的写，和字节流向 Http 报文的解码

- 类图

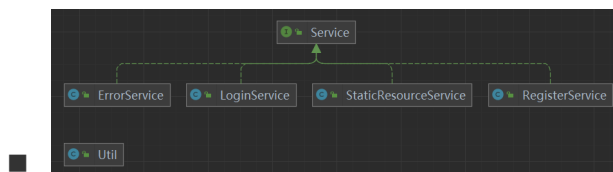


- Service 包

- 主要功能

- 静态资源处理和业务逻辑处理，都实现 `HttpResponse handle(HttpRequest request) throws Exception` 接口

- 类图



- 关键数据结构

- 静态资源处理时，使用 `HashMap` 进行 MIME, 301, 302, 304 的映射

```

文件类型 -> MIME类型

7 个用法
private static HashMap<String, String> MIME = new HashMap<>();

永久移动的资源 对应状态码301 301 Moved Permanently 永久移动。是指请求的资源已被永久的移动到新的
URL，返回信息会包括新的URL，浏览器还会自动定向到新的URL。今后任何新的请求都应该使用新的URL来代
替

5 个用法
public static HashMap<String, String> MovedPermanentlyResource = new HashMap<>();

暂时移动的资源 对应状态码302 302 Found 临时移动。与301类似。但是资源只是临时被移动。客户端应该继
续使用原有的URI

6 个用法
public static HashMap<String, String> MovedTemporarilyResource = new HashMap<>();

private static HashMap<String, String> ModifiedTime = new HashMap<>();
  
```


- 注册、登录功能，使用静态 HashMap 保存用户名密码

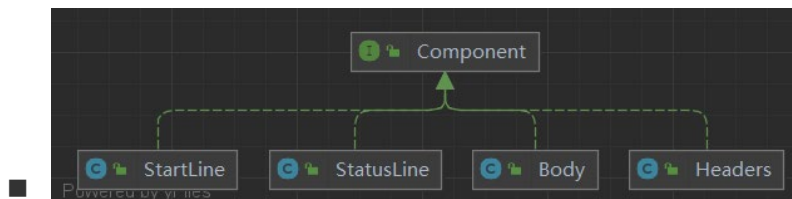
```
public static HashMap<String, String> db = new HashMap<>(); //静态HashMap保存用户名和密码
```

- Http 包

- 主要功能

- 封装 Http 请求和响应，并实现和报文文本的相互转换

- 其中 Components 包的类图



功能点

HTTP 服务器端支持 GET 请求，200 状态码

浏览器 GET 请求静态资源，服务端呈现请求报文：

```
2022-07-01T17:13:32,637 INFO [LOGID:] [Thread-19] [com.nju.HttpS.Handl.AcceptHandler:32] 建立连接:address/0:0:0:0:0:0:1:53788, hashCode: 2078989146
2022-07-01T17:13:32,637 INFO [LOGID:] [Thread-20] [com.nju.HttpS.Handl.AcceptHandler:32] 建立连接:address/0:0:0:0:0:0:1:53789, hashCode: 140384846
2022-07-01T17:13:32,643 INFO [LOGID:] [Thread-17] [com.nju.HttpS.Handl.RequestHandler:52] 服务器收到请求，完整打印:
GET /index.html HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
sec-ch-ua: " Not;A Brand";v="99", "Microsoft Edge";v="103", "Chromium";v="103"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.53 Safari/537.36 Edg/103.0.1264.37
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
```

服务器响应 200 状态码

名称	× 标头 预览 响应 发起程序 计时
index.html	<div> <div>常规</div> <div> <p>请求 URL: http://localhost:5000/index.html</p> <p>请求方法: GET</p> <p>状态代码: 200 OK</p> <p>远程地址: [::1]:5000</p> <p>引用站点策略: strict-origin-when-cross-origin</p> </div> </div> <div> <div>响应头 查看源</div> <div> <p>Content-Length: 731</p> <p>Content-Type: text/html</p> <p>Keep-Alive: timeout=10</p> <p>Last-Modified: Sat, 18 Jun 2022 02:15:15 GMT</p> </div> </div>
style.css	
pic.jpg	
main.js	
favicon.ico	

HTTP 服务器端支持 POST 请求

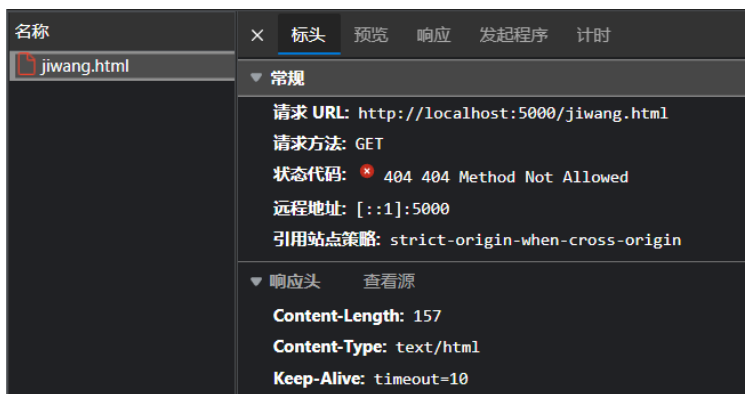
```
2022-07-01T17:17:04,718 INFO [LOGID:] [Thread-17] [com.nju.HttpS.Handl.RequestHandler:52] 服务器收到请求，完整打印：
POST /register HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Content-Length: 25
Cache-Control: max-age=0
sec-ch-ua: " Not;A Brand";v="99", "Microsoft Edge";v="103", "Chromium";v="103"
```

名称	× 标头 负载 预览 响应 发起程序 计时
register	<div> <div>常规</div> <div> <p>请求 URL: http://localhost:5000/register</p> <p>请求方法: POST</p> <p>状态代码: 200 OK</p> <p>远程地址: [::1]:5000</p> <p>引用站点策略: strict-origin-when-cross-origin</p> </div> </div> <div> <div>响应头 查看源</div> <div> <p>Content-Length: 183</p> <p>Content-Type: text/html</p> <p>Keep-Alive: timeout=10</p> </div> </div>

HTTP 服务器端支持 404 状态码

```
2022-07-01T17:20:59,257 INFO [LOGID:] [Thread-17] [com.nju.HttpS.Handl.RequestHandler:52] 服务器收到请求，完整打印：
GET /jiwang.html HTTP/1.1
Host: localhost:5000
Connection: keep-alive
sec-ch-ua: " Not;A Brand";v="99", "Microsoft Edge";v="103", "Chromium";v="103"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.53 Safari/537.36 Edg/103.0.1264.37
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6

2022-07-01T17:20:59,258 ERROR [LOGID:] [Thread-17] [com.nju.HttpS.Servi.StaticResourceService:86] /jiwang.html is not found, 404
```

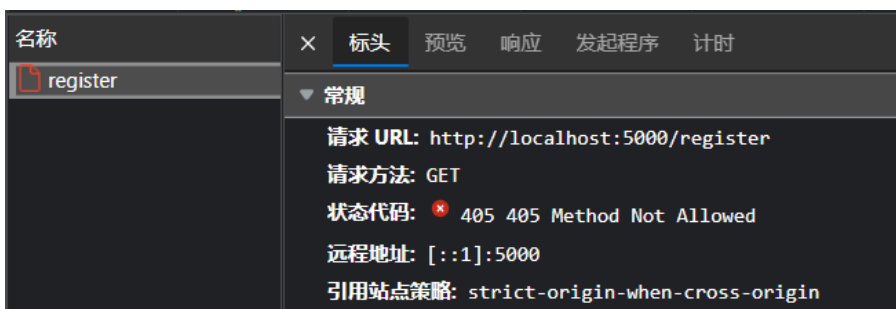


HTTP 服务器端支持 405 状态码

浏览器地址栏访问 `http://localhost:5000/register`，以 GET 方法请求注册为 POST 的接口

```
@RequestMapping(uri = "/register", method = RequestMethod.POST)
public HttpResponse register(HttpRequest request) throws Exception {
    return new RegisterService().handle(request);
}
```

响应 405



HTTP 服务器端支持 500 状态码

浏览器地址栏访问 `http://localhost:5000/error`，服务端内方法抛出异常

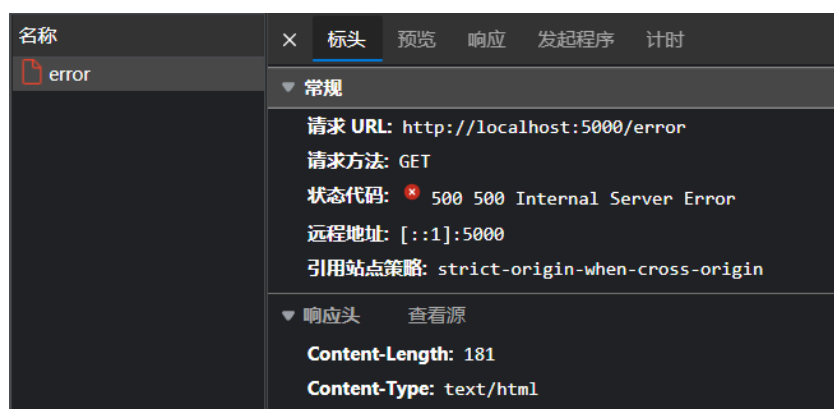
```
@RequestMapping(uri = "/error", method = RequestMethod.GET)
public HttpResponse Error(HttpRequest request) throws Exception {
    return new ErrorService().handle(request);
}
```

```

public class ErrorService implements Service {
    4 个用法 HC_Plantern
    public HttpResponse handle(HttpRequest request) throws Exception {
        // do something bad
        // 返回500页面
        throw new Exception("自定义异常，用于测试异常抛出");
    }
}

```

响应 500



HTTP 服务器端支持 301，302，304 状态码


已在客户端部分给出详细测试说明。

HTTP 服务器端支持长连接

已在客户端部分给出详细测试说明。

HTTP 服务器端支持至少三种 MIME 类型

hello



注册

Username:

Password:

Register

登录

Username:

Password:

Login

hi

确定

元素 控制台 源代码 网络 x 13 设置 帮助 ... X

网络 x 13 设置 帮助 ... X

全部 Fetch/XHR JS CSS img 媒体 字体 文档 WS Wasm 清单 其他 已阻止 Cookie

已阻止请求 第三方请求

50 ms 100 ms 150 ms

名称	状态	类型	发起...	大小	时间	时间线
index.html	200	do...	其他	860 B	3 ms	
style.css	200	styl...	index...	168 B	5 ms	
pic.jpg	200	tex...	index...	2.2 MB	37 ms	
main.js	200	scri...	index...	218 B	3 ms	
favicon.ico	200	x-ic...	其他	4.4 kB	3 ms	

5 次请求 已传输 2.2 MB 2.2 MB 多资源 完成: 159 ms DOMContentLoaded: 71 ms 加载: 153 ms

```
static {  
  MIME.put(".html", "Content-Type: text/html");  
  MIME.put(".png", "Content-Type: image/png");  
  MIME.put(".jpg", "Content-Type: image/jpeg");  
  MIME.put(".js", "Content-Type: text/javascript");  
  MIME.put(".css", "Content-Type: text/css");  
  MIME.put(".ico", "Content-Type: image/x-icon");  
}
```