2018

HTTPCORE 源码分析

2016K8009929009

秦 宏

写在前面

面向对象的源码分析文档，内容不定时更新，主要目标依然是为了完成课堂任务，同时有助于大型工程的源码阅读分析。原有的内容也可能会修改，以最新文档内容为准。

我的 github：https://github.com/qinhongUcaser

目录

一、HTTP 初步了解

（一）概念

超文本传输协议（英语：HyperText Transfer Protocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议[1]。HTTP 是万维网的数据通信的基础。

（二）目的

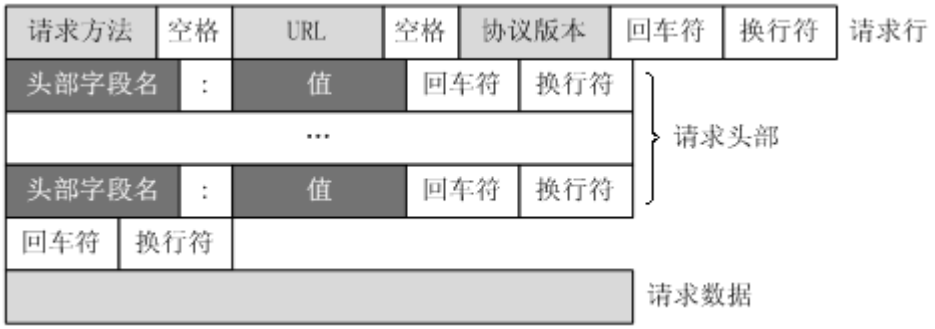
设计 HTTP 最初的目的是为了提供一种发布和接收 HTML 页面的方法。通过 HTTP 或者 HTTPS 协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。

（三）详细内容

HTTP 是一个客户端终端（用户）和服务器端（网站）请求和应答的标准（TCP）。通过使用网页浏览器、网络爬虫或者其它的工具，客户端发起一个 HTTP 请求到服务器上指定端口（默认端口为 80）。我们称这个客户端为用户代理程序（user agent）。应答的服务器上存储着一些资源，比如 HTML 文件和图像。我们称这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在多个“中间层”，比如代理服务器、网关或者隧道（tunnel）。

（四）消息(报文)格式

客户端发送一个 HTTP 请求到服务器的请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据四个部分组成，下图给出了请求报文的一般格式。



（五）八种方法

1.GET

向指定的资源发出“显示”请求。使用 GET 方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中，例如在 Web Application 中。

2.HEAD

与 GET 方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部分。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”。

3.POST

向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中。这个请求可能会创建新的资源或修改现有资源，或二者皆有。

4.PUT

向指定资源位置上传其最新内容。

5.DELETE

6.TRACE

7.OPTIONS

8.CONNECT

（六）安全方法

对于 GET 和 HEAD 方法而言，除了进行获取资源信息外，这些请求不应当再有其他意义。也就是说，这些方法应当被认为是“安全的”。 客户端可能会使用其他“非安全”方法，例如 POST，PUT 及 DELETE，应该以特殊的方式（通常是按钮而不是超链接）告知客户可能的后果（例如一个按钮控制的资金交易），或请求的操作可能是不安全的（例如某个文件将被上传或删除）。

但是，不能想当然地认为服务器在处理某个 GET 请求时不会产生任何副作用。事实上，很多动态资源会把这作为其特性。这里重要的区别在于用户并没有请求这一副作用，因此不应由用户为这些副作用承担责任。

（七）关于 HTTPcore（摘自 HTTPcore 官网）

HttpCore 是一组底层 HTTP 传输组件，可用于以最小的占用空间构建自定义客户端和服务器端 HTTP 服务。HttpCore 支持两种 I / O 模型：基于经典 Java I / O 的阻塞 I / O 模型

和基于 Java NIO 的非阻塞，事件驱动的 I / O 模型。

1. HTTPcore 的范围

构建客户端/代理/服务端的一致 API；构建同步和异步 HTTP 服务的一致 API；基于阻塞（经典）和非阻塞（NIO）I/O 模型的一组低级组件。

2. HTTP core 的目标

实现大多数的基本 HTTP 传输方面；平衡 API 的性能和清晰性和可表达性；低内存消耗(可预测)；独立的库(除 JRE 外，没有外部依赖)

3. HTTPcore 不是什么

不是 HttpClient 的替代品；不是 Servlet API 的替代品。

本次分析选取的版本为 HTTPcore 中 4.4.10 版本的源代码，具体包含了 HTTP 方法中的 GET 等通信方法。还未具体确定要选择的功能分析，代码结构仍然在学习阶段。

二、源代码文件结构解析

文件目录下一共有 4 个主要的文件夹，分别是 httpcore、httpcore-ab、httpcore-nio、httpcore-osgi。

（一）httpcore

该文件下存储了 4.4 版本下的 httpcore 的主要的核心代码。具体分析其中 main 文件夹下的文件内容。

Annotation 是注释文件夹；concurrent 是并发（使用未来模式）的文件夹；config 是配置文件夹；entity 是 http 消息的实体类文件夹；impl 是实现类文件夹；io 是输入输出类文件夹；message 是消息类文件夹；params 是参数文件夹，但是多被注释掉，不考虑使用；pool 的内容还不清楚；protocol 为标准文件夹；ssl 为 HTTPS 以安全为目标建立的 HTTP 通道文件；util 包含集合框架、遗留的 collection 类、事件模型、日期和时间设施、国际化和各种实用工具类（字符串标记生成器、随机数生成器和位数组、日期 Date 类、堆栈 Stack 类、向量 Vector 类等）。集合类、时间处理模式、日期时间工具等各类常用工具包(摘自百度)。

另外有多个独立的 Java 文件表示部分具体操作行为，具体操作将在后面分析。

（二）httpcore-ab

该文件下存储的是用于测试 httpcore 执行的 benchmark。

（三）httpcore-nio

该文件下存储了 Java NIO 的详细代码，用于实现非阻塞驱动模型。

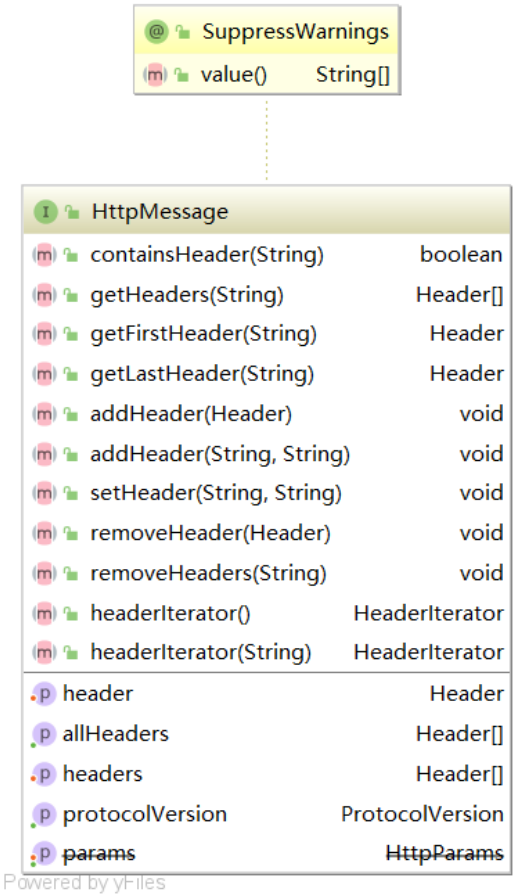
（四）httpcore-osgi

这里详细代码用途不清楚，看内容是存储了 OSGI 框架注释，用来区别一个叫做 IT 的变量。

三、代码具体功能解析

（一）以 request 和 response message 为例分析源码中的数据结构。

两种模式都连接了 HttpMessage 接口。可以看到定义的方法。



1.request message

如下图的一次 HTTP 请求消息(报文)，输入包含了使用消息的方法、标识符和协议版本。

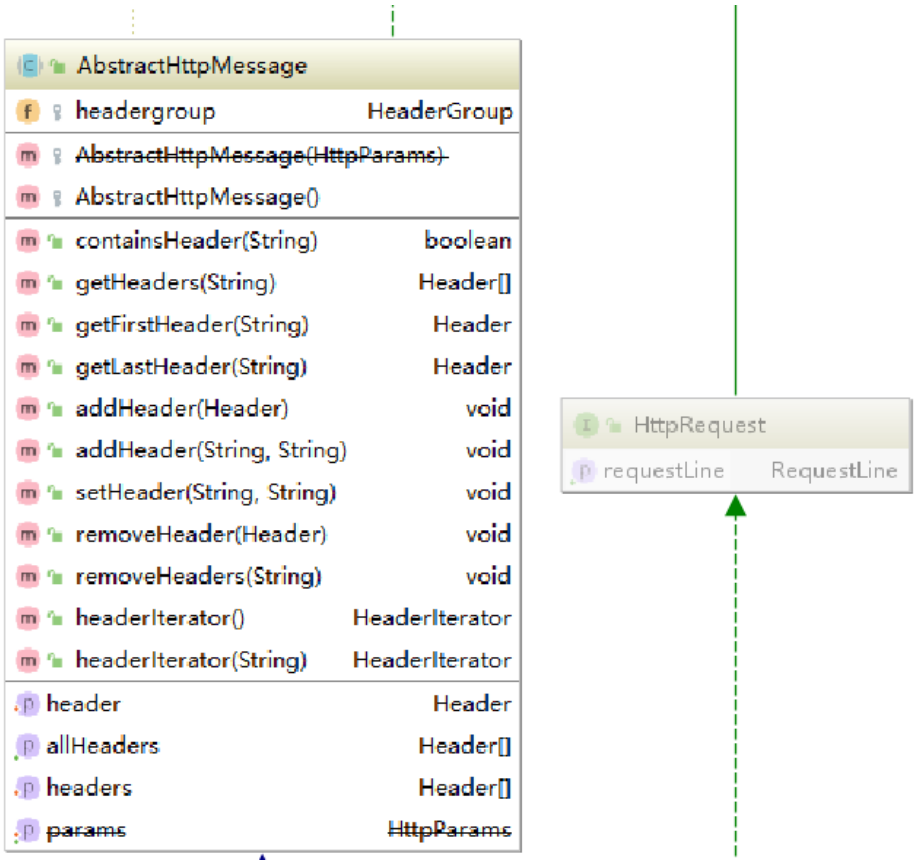
```
HttpRequest request = new BasicHttpRequest("GET", "/",
    HttpVersion.HTTP_1_1);

System.out.println(request.getRequestLine().getMethod());
System.out.println(request.getRequestLine().getUri());
System.out.println(request.getProtocolVersion());
System.out.println(request.getRequestLine().toString());
```

可以看到输出流的结果

```
GET
/
HTTP / 1.1
GET / HTTP / 1.1
```

方法和标识符的类型均为串类型，协议版本类型类包含了版本 UID、协议名和协议版本号。

BasicRequest 中类 BasicRequest 继承了抽象类 AbstractHttpMessage，两者又同时与接口 HTTPMessage 相连，HTTPMessage 中包含了如下的一些方法。下图是消息请求过程中可以用到的方法：



根据命名可以推断出方法的目的以及作用。

2. response message

下图表示 HTTP 响应是服务器在接收并解释请求消息后发送回客户端的消息。该消息的第一行包括协议版本，后跟数字状态代码及其相关的文本短语。

```
HttpResponse response = new BasicHttpResponse(HttpVersion.HTTP_1_1,
    HttpStatus.SC_OK, "OK");

System.out.println(response.getProtocolVersion());
System.out.println(response.getStatusLine().getStatusCode());
System.out.println(response.getStatusLine().getReasonPhrase());
System.out.println(response.getStatusLine().toString());
```

输出流如下

```
HTTP/1.1
200
OK
HTTP/1.1 200 OK
```

总的类输入内容包含了状态行、协议版本、数字状态代码、文本短语、

不同于请求消息时输入的参数是满的，这里只输入了 ProtocolVersion,code,reasonPhrase.

BasicHttpResponse 的结构依然是继承了 AbstractHttpMessage,连接了 HttpResponse 接口（继承了 HttpMessage）。这里消息(报文)的 header 内可以包含多个，用于表示消息的具体属性，包括请求内容的长度和类型等。