HTTPCORE源码分析

2016K8009929009

秦 宏

2018

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

[写在前面 1](#_Toc531700274)

[一、HTTP初步了解 3](#_Toc531700275)

[（一）概念 3](#_Toc531700276)

[（二）目的 3](#_Toc531700277)

[（三）详细内容 3](#_Toc531700278)

[（四）八种方法 3](#_Toc531700279)

[1.GET 3](#_Toc531700280)

[2.HEAD 3](#_Toc531700281)

[3.POST 3](#_Toc531700282)

[4.PUT 4](#_Toc531700283)

[5.DELETE 4](#_Toc531700284)

[6.TRACE 4](#_Toc531700285)

[7.OPTIONS 4](#_Toc531700286)

[8.CONNECT 4](#_Toc531700287)

[（五）安全方法 4](#_Toc531700288)

[（六）关于HTTPcore 4](#_Toc531700289)

[二、源代码文件结构解析 4](#_Toc531700290)

[（一） httpcore 5](#_Toc531700291)

[1. annotation 5](#_Toc531700292)

[2. concurrent 5](#_Toc531700293)

[3. config 5](#_Toc531700294)

[4. entity 5](#_Toc531700295)

[5. impl 5](#_Toc531700296)

[6. io 5](#_Toc531700297)

[7. message 5](#_Toc531700298)

[8. params 5](#_Toc531700299)

[9. pool 5](#_Toc531700300)

[10. protocol 5](#_Toc531700301)

[11. ssl 5](#_Toc531700302)

[12. util 5](#_Toc531700303)

[（二） httpcore-ab 5](#_Toc531700304)

[（三） httpcore-nio 6](#_Toc531700305)

[（四） httpcore-osgi 6](#_Toc531700306)

[三、代码具体功能解析 6](#_Toc531700307)

# 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

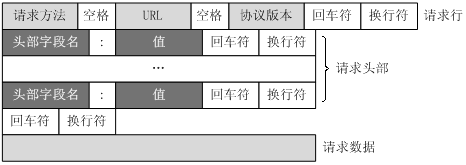
设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。通过HTTP或者HTTPS协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。

## （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。

### 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。

### 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)

### 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。

## 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。

## httpcore-ab

该文件下存储的是用于测试httpcore执行的benchmark。

## 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。

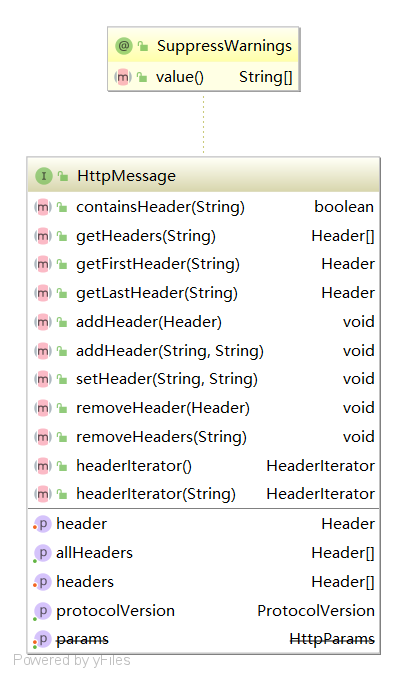
## httpcore-osgi

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

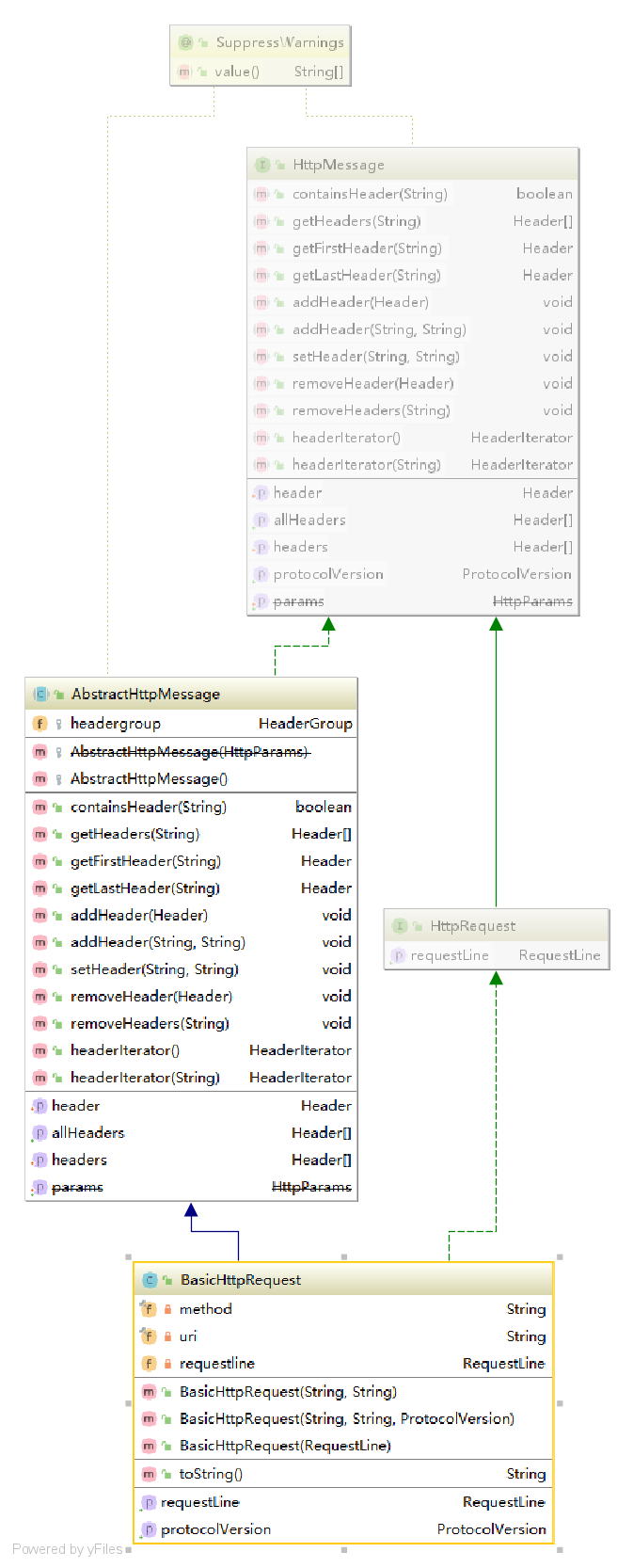
/

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。