**面向对象的程序设计第二次实例分析报告**

**曾思远**

**2017K8009909012**

关于Httpcomponents-core阅读报告

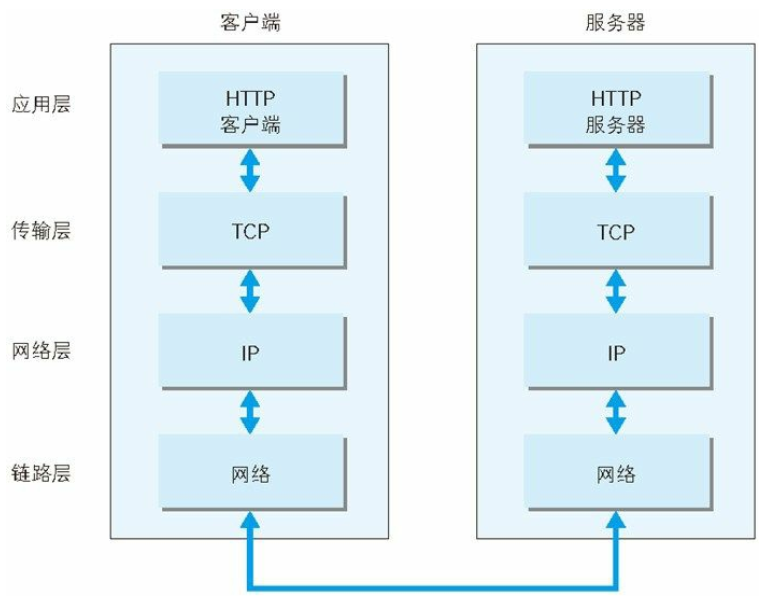
1. 浅谈HTTP协议
2. HTTP协议的使用场景

在我们的生活当中，每当我们在浏览器上输入网址并敲下回车键时，我们便运用到了HTTP协议将我们的请求进行包装并通过上层协议传送了出去。当我们的请求到达目标服务器时，目标服务器会根据我们的请求进行响应。同样的，请求的响应也会采用HTTP协议进行包装然后返回到我们的机器上，这样我们的浏览器就会显示服务器为我们传送的内容了。通过上面的过程可以看到，HTTP协议是建立在客户端和服务器通信之上的一种通信协议，这种协议作为规范约束着二者的通信行为。

1. HTTP协议究竟做了什么

HTTP通常被译为超文本传输协议，其实更加严谨的译名应当是“超文本传输协议”，但是由于前者已经约定俗成，故我们在之后的讨论当中采用前者。目前HTTP/1.1是目前主流的HTTP协议版本，于1997年1月公布，尽管之后的讨论多在Web的框架当中，但是HTTP协议已经被运用到了各个场景当中。

为了更好地理解Httpcomponents-core做了什么，我们可以看一下客户端和服务器之间的整个通信传输流：



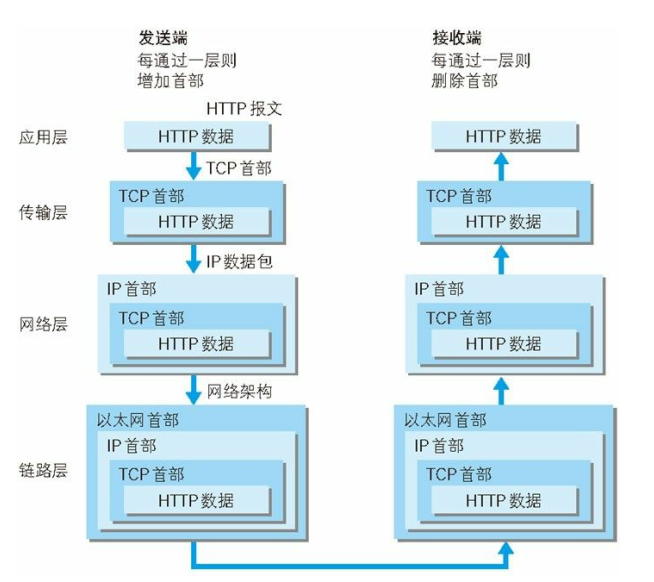
HTTP协议隶属于TCP/IP协议族，TCP/IP协议族时进行分层管理的，从上图可以看到，HTTP协议位于最底层的应用层，负责对请求或响应进行包装，决定了向用户提供应用服务时通信的活动，我们要探讨的Httpcomponent-core的目标便是将请求包装成符合上层接口的的请求包。其实应用层也不仅仅包括HTTP协议，它还包括了FTP（File Transfer Protocol，文件传输协议）和DNS（Domain Name System，域名系统）等一系列的协议。DNS域名系统可以将一台计算机的主机名和域名解析为IP地址。此外在这里简要介绍一下其他三个层的作用，也有助于我们更好的理解我们在最底层需要做什么事情：

传输层：对上层应用层，提供了处于网络连接中的两台计算机之间的数据传输，在这一层主要的通信协议是TCP协议，该协议提供了可靠的字节流服务。TCP协议主要完成了以下两个任务：首先是将应用层传过来的数据分割成以报文段为单位的数据包进行管理，便于传输；其次通过握手机制来确保数据能够传送到对方。

网络层：处理在网络上流动的数据包，数据包是网络传输的的数据单位。这一层规定了数据的传输路线，IP网络协议就位于该层。IP协议指明了节点被分配到的地址（IP地址），以及网卡所属的固定地址（网卡所属的固定地址）。

链路层：处理网络连接的硬件部分。硬件设备驱动、网卡、光纤等物理可见部分都属于该层。

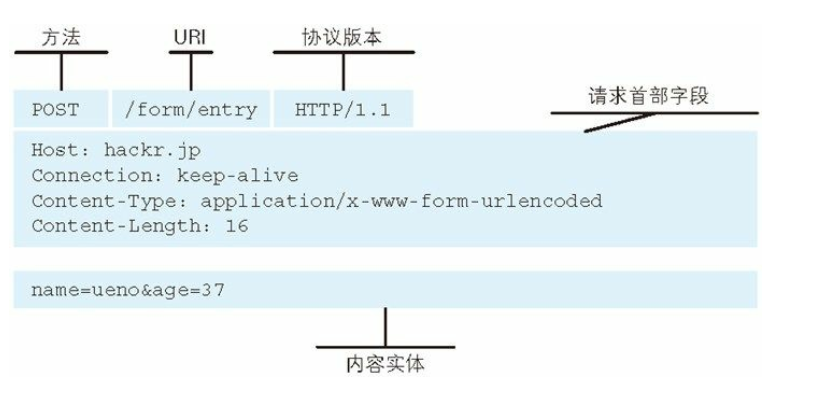
可以将整个的过程描述如下：



1. HTTP报文的构成

现在我们来到真正的HTTP协议的内部是如何实现的对请求和响应的包装的。HTTP协议规定，请求从客户端发出，最后服务器端响应该请求并返回，服务器端在没有接收到请求之前不会发送响应。

可以看一个具体的实例，来了解HTTP的报文构成，报文的不同组成也是在HttpComponents-core（以下简称Httpcore）中各个类的建立根据（之后会具体介绍）。通常Http报文分为请求报文和响应报文两种，请求报文通常是客户端发出来请求资源，响应报文通常是服务器回馈请求时发出。二者虽然结构类似但是还是有一定的区别，所以在Httpcore中将其分为了两个不同的类。下面我们具体看一下这两类报文的基本结构：



可以看到请求报文其实由请求行（第一行的方法、URI、协议版本）、请求首部字段和实体组成。请求行反映了请求访问服务器的类型（方法）、请求访问的资源对象（请求URI）以及HTTP的协议版本。不妨简要解读一下上述报文的内容：以Post方法请求访问主机域名为hacker.jp的.form/entry资源，其中协议版本为HTTP/1.1。需要说明的是，请求报文的各种首部字段大多是可选项，可长可短，需要根据具体的访问请求来进行构造。



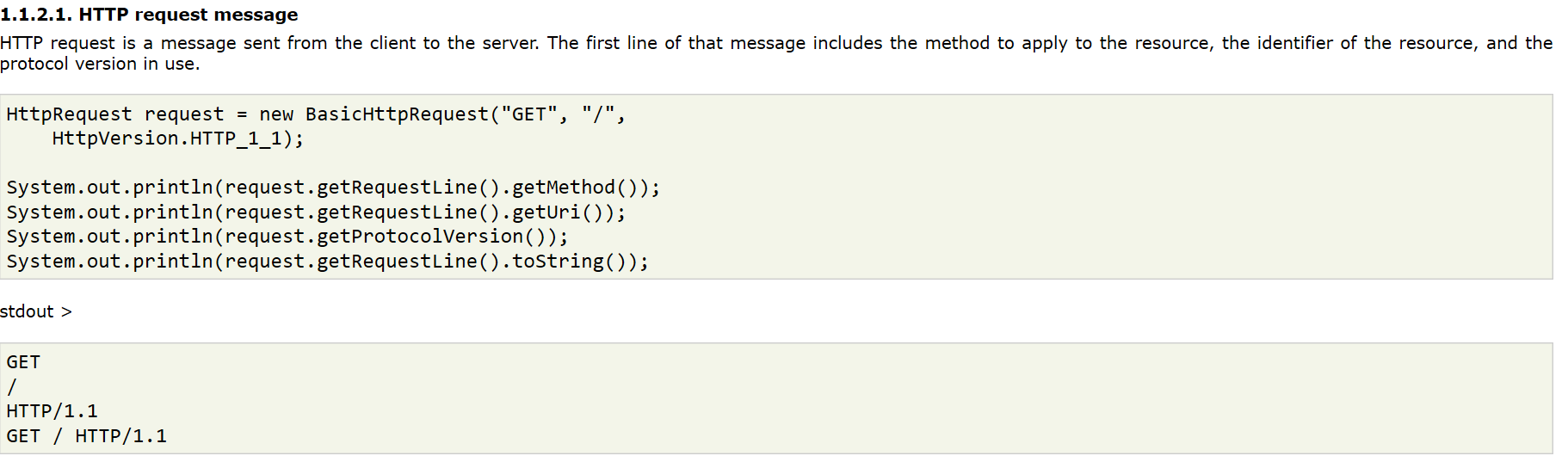
响应报文和请求报文的层次结构类似，但是在每个层次中的具体内容却并不相同。第一行为响应报文的状态行，状态行包含了协议版本、状态码以及状态码的原因短语。第二层次依旧是各种响应首部字段，可以体现当前的响应时间、响应主体的长度和响应的内容格式等等。同样的，响应的首部字段也是可选的，代表了响应的基本信息。

1. Httpcore的基本类及组成

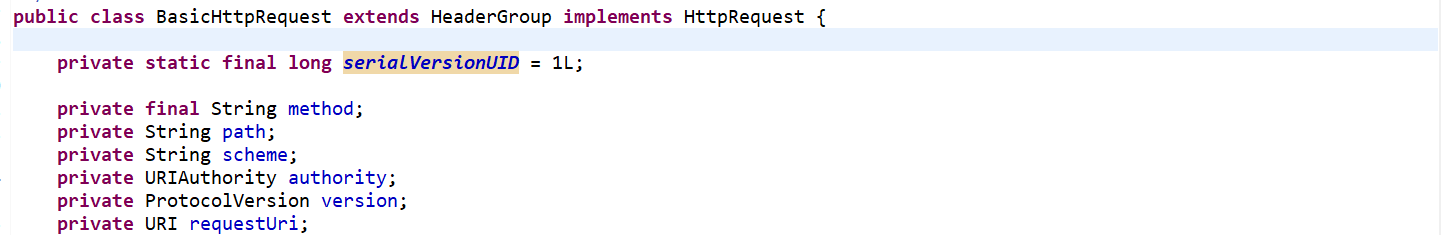
Httpcore按照HTTP协议定义了HTTP报文的对象模型（也就是类）以及提供了对报文元素序列化（格式化）以及反序列化（语法解析）的扩展支持。

1. Http请求报文

我们不妨来看一下官方文档中给出的例子：



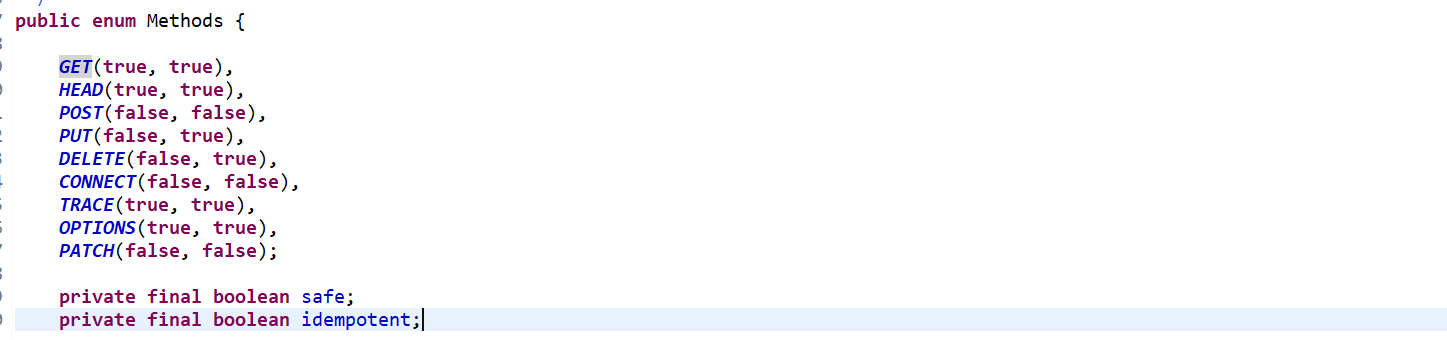
可以看到，当我们需要发出请求报文的时候，需要创建一个BasicHttpRequest类的对象作为报文的主体。其实在上面的介绍中可以看到，请求报文的首部字段是可选的，我们在创建该对象的时候可能只知道请求报文的请求行内容，也有可能已经知道主机域名等其他信息，这就说明BasicHttpRequest类的构造方法并不只能有一种，需要多种不同参数的构造方法，当然BasicHttpRequest类在实现的过程中就已经返回体现了这一点。话不多说，我们先一起来看看BasicHttpRequest类的基本结构是什么样的：



BasicHttpRequest类中包含了7个基本的成员变量。首先是序列版本的网络用户证明，采用final关键字保证这个成员变量是不能被修改的，一定程度上体现了安全性。剩下的部分就是我们熟知的Http请求报文中的一些内容。

1. method请求方法

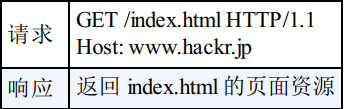
首先是method成员变量，由于被final修饰，代表了请求的方法必须在创建对象时就被赋初值并且不能再被改变。method会告知该服务器的请求意图是什么，由于method被定义为一个简单的String基本类，则内部的具体实现不必多谈，在此我们介绍一些常见的方法和其具体作用，其实在Httpcore当中已经通过枚举类的方式对Http方法进行了定义：



可以看到Methods类其实定义9种方法，每种方法又对应着两个属性，分别就是底部的两个布尔变量safe和idempotent。**安全方法（Safe Methods）**是指用户不管进行多少次操作，资源的状态都不会改变。比如说，GET和HEAD方法仅能获取资源，而不会执行动作，这些方法属于“安全”的方法。而POST，PUT，DELETE方法改变了资源的状态，可能会执行不安全的动作，使用者应该意识到这一点。**幂等（Idempotent Methods）**是数学的一个用语，对于单个输入或者无输入的运算方法，如果每次都是同样的结果，则称其是幂等的。方法GET，HEAD，PUT，DELETE都有这种性质。POST方法不是幂等的，比如常见的POST重复加载问题：当我们多次发出同样的POST请求后，其结果是创建出了若干的资源。一些常用的请求方法此简要介绍一下，有助于我们对不同的method的执行方式有更深入的理解：

GET：获取资源

GET方法用来请求访问已被URI识别的资源，指定的资源经服务端解析以后返回响应内容。



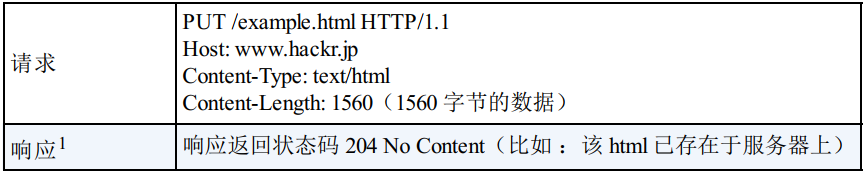
POST：传输实体主体

POST方法用来传输实体的主体，其主要目的并不是获取相应的主体内容。



PUT：传输文件

PUT方法用来传输文件，要求在请求报文中包含文件内容，然后保存到请求URI指定的位置。



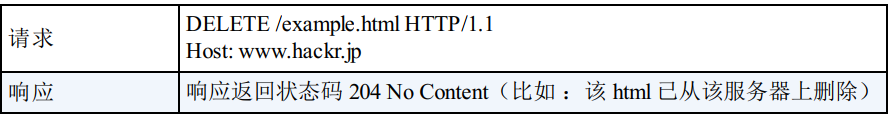
HEAD：获得报文首部

HEAD方法主要用于确认URI的有效性及资源更新的日期时间等。



DELETE：删除文件

DELETE方法按请求URI删除指定的资源。



OPTIONS：询问支持的方法

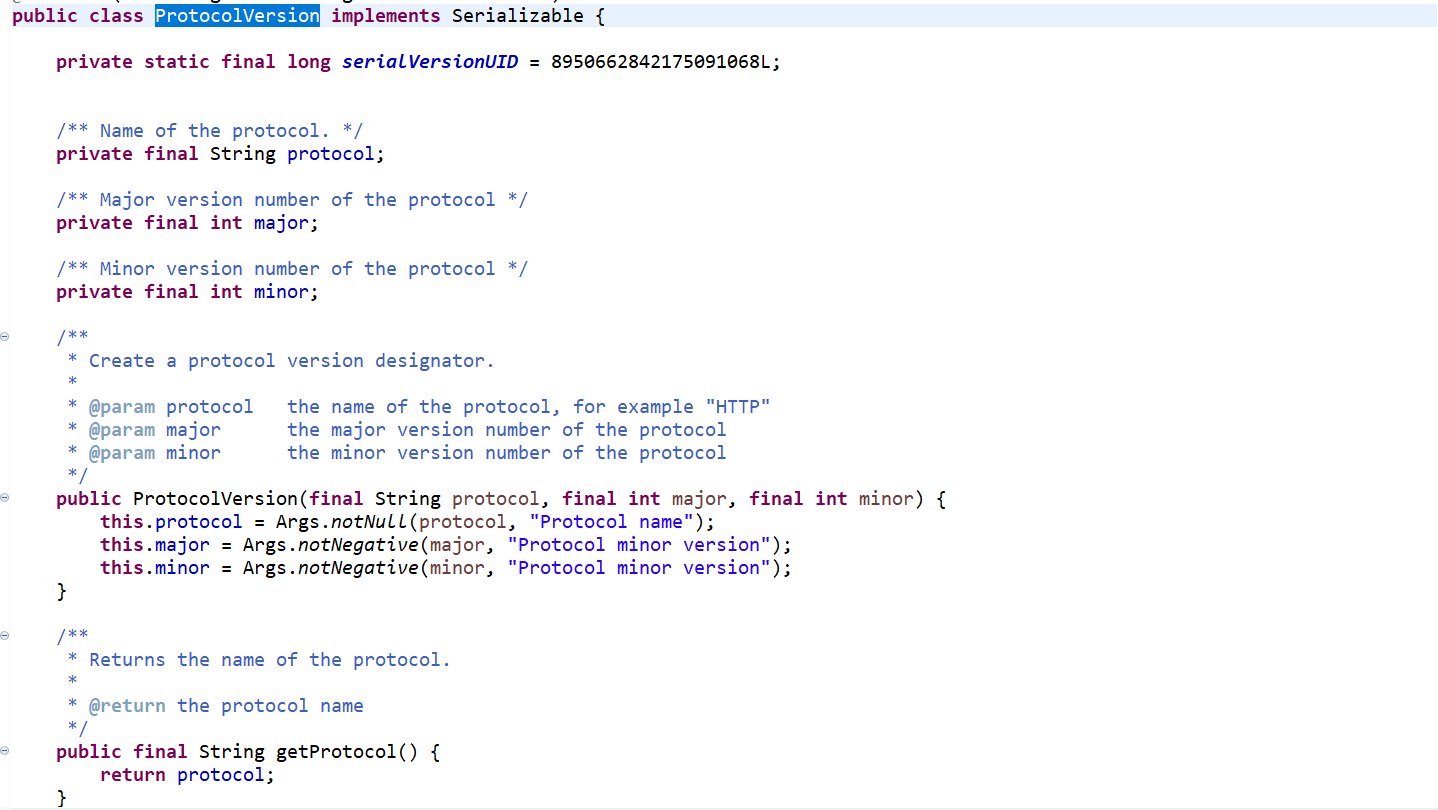
OPTIONS方法用来查询针对请求URI指定的资源支持方法。



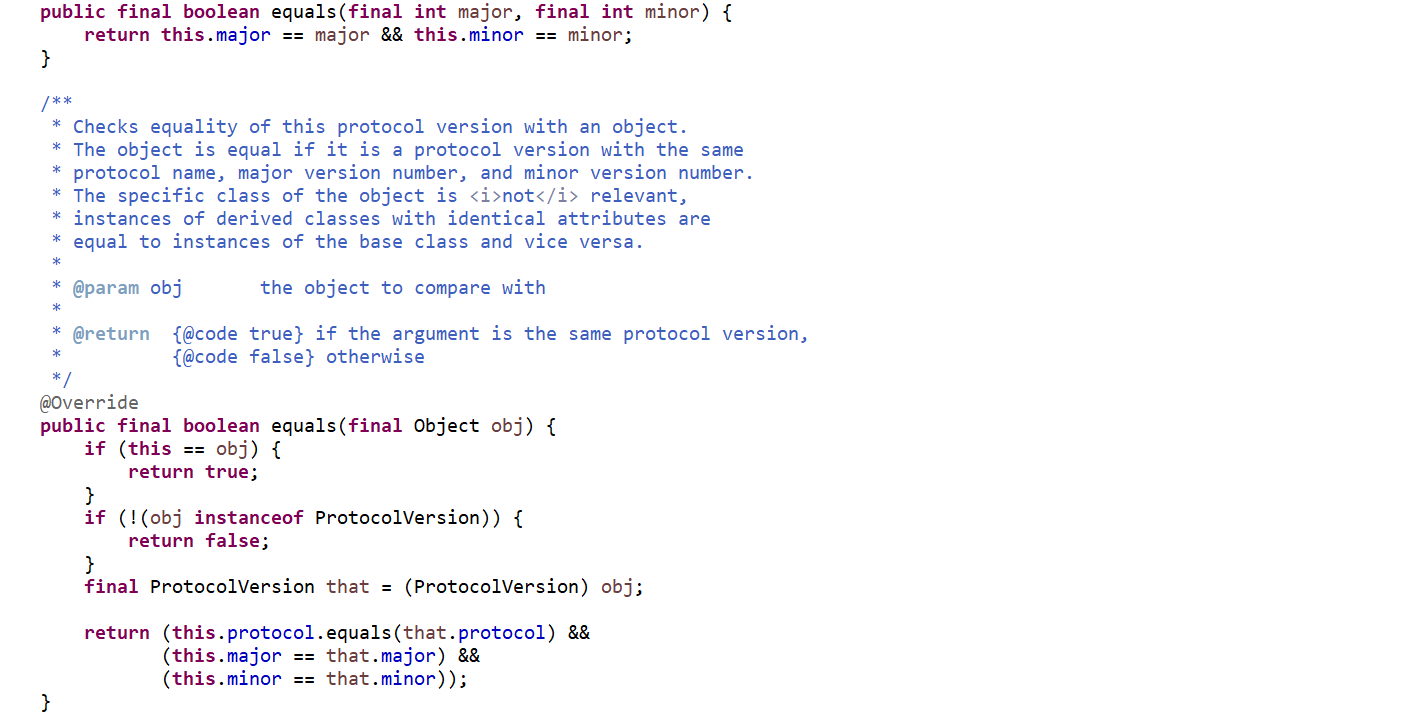
剩余的TRACE方法和CONNECT方法并不常用，在此不做过多介绍。但是可以看到，每一种方法的背后都有两个属性，从功能上可以看到其安全性和幂等性如枚举类中定义的那样。当然Methods类也有自己的成员方法来设置安全性和幂等性，但是暂时没有找到用处，在此不再赘述。

1. version（HTTP协议版本）

version成员变量 ProtocolVersion类来定义的：

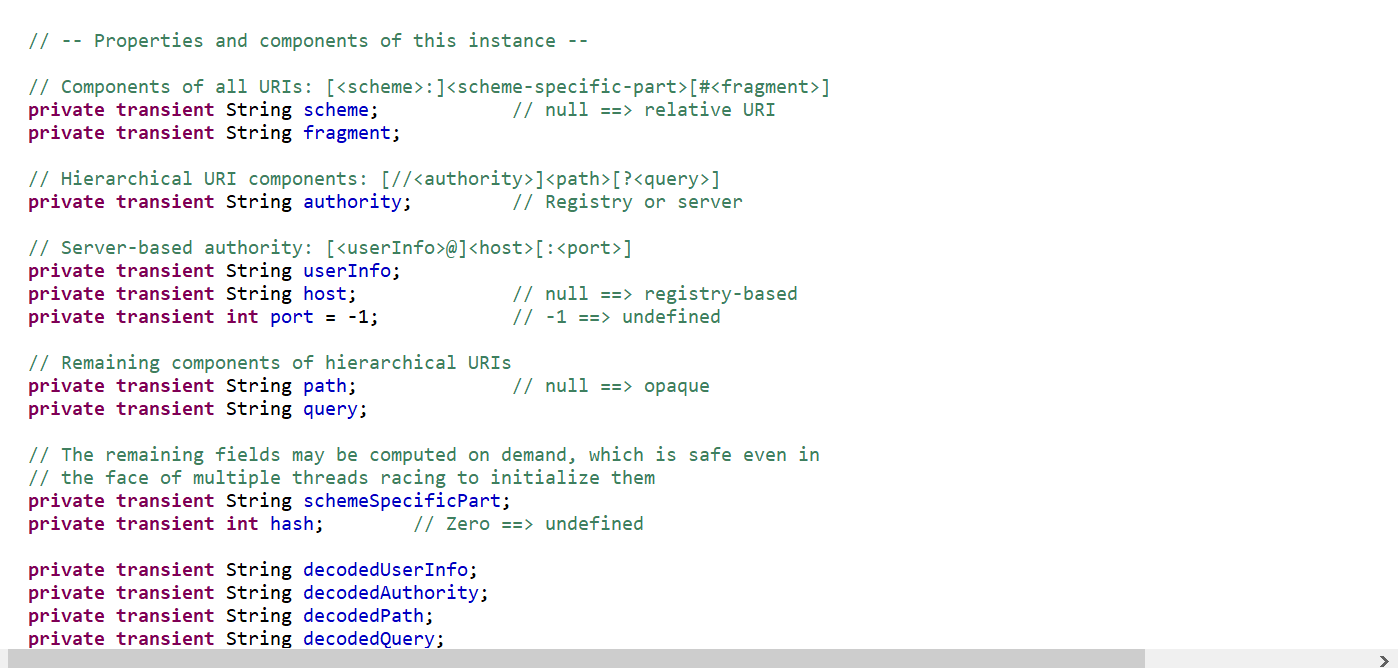


该类继承了Serializable接口，主要分为三个成员变量，一个是协议名称protocol（比如HTTP），其次还有最大版本号和最低版本号两个成员变量。该类不仅提供了修改和获取的成员方法，还提供了比对的功能：



1. URI（统一资源标识符）

URI域是统一资源标识符，用于定位请求资源的具体位置。URI类有如下定义：



可以看到URI类中对绝对URI的格式中不同的部分运用了不同的成员变量进行了分解，为了理解不同的部分的含义，我们可以对绝对URI进行一下了解：



登录信息（认证）：指定用户名和密码作为从服务器端获取资源时必要的登录信息（身份认证）。（可选）

服务器地址：使用绝对URI必须指定带访问的服务器地址。地址可以是类似hacker.jp这种DNS可解析的名称，或是192.168.1.1这类IPV4地址名，也可以是[0:0:0:0:0:0:0:1]这样的IPV6地址名。

服务器端口号：指定服务器连接的端口号。（可选项）

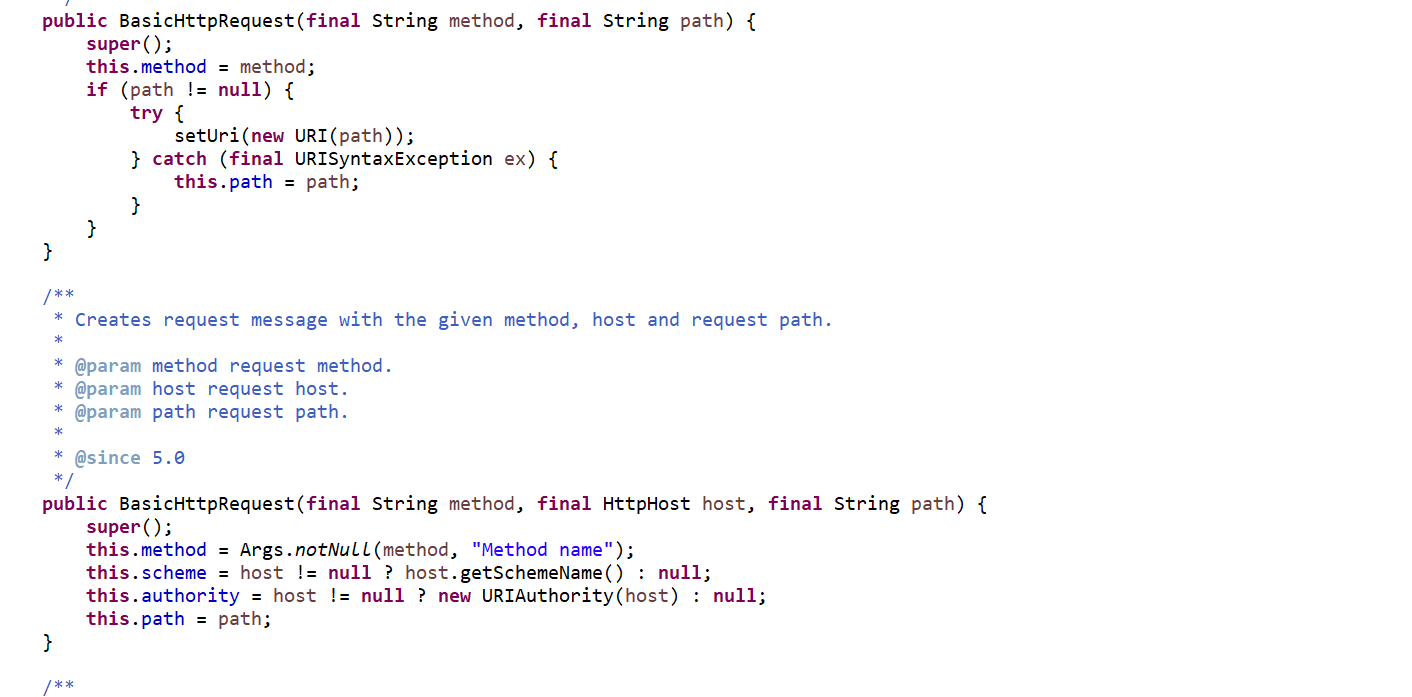
带层次的文件路径：指定服务器上的文件路径定位特指的资源。

查询字符串：针对已指定的文件路径内的资源，可以使用查询字符串传入任意参数。（可选项）

片段标识符：通常可标记触以获取资源中的子资源（文档内的某个位置）。（可选项）

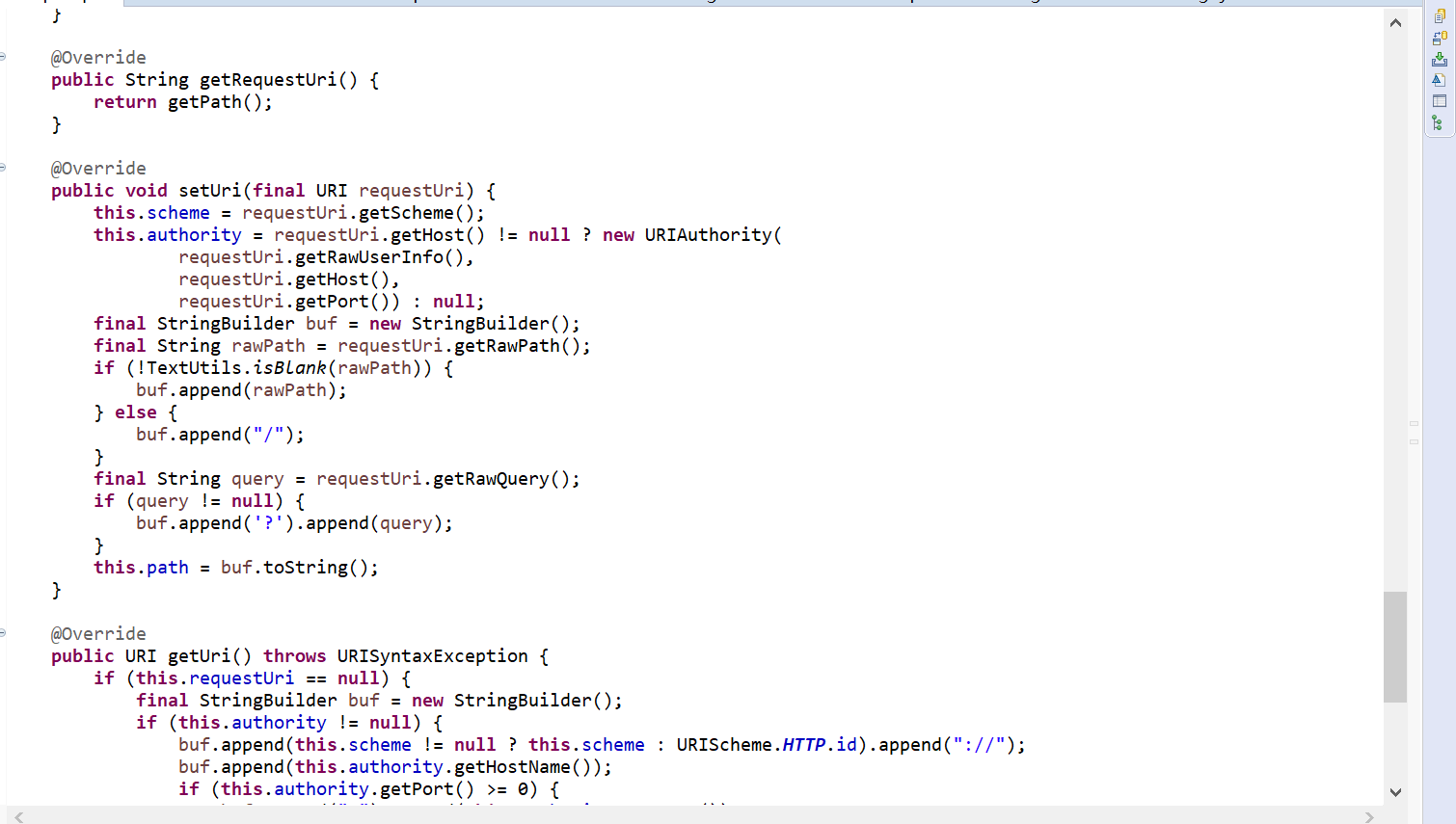
在URI类当中，同样也提供了对以上各个成员变量的赋值和获取的成员方法，由于计算机网络的具体内容笔者实在不是非常了解，所以不敢具体分析每一个函数的作用。

在BasicHttpRequest提供了查询这以上成员变量的方法以及不同的构造方法：



这些都是不同的构造函数来对请求报文对象进行构建，请求报文类和之后讲到的响应报文类都继承了HeaderGroup类，因为二者都拥有各种各样的首部字段来对报文进行描述。





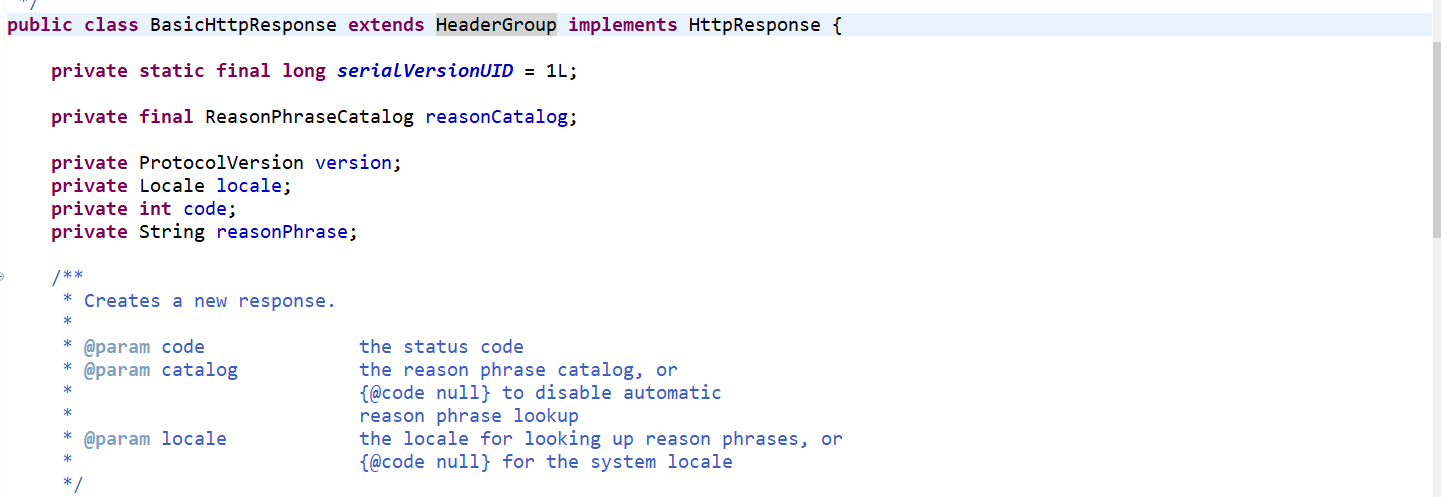
以上是对请求报文成员变量赋值和查询的成员方法。

1. Http响应报文

同样的，我们也来看一下官方文档中给出的示例：



可以看到，在官方文档给出的示例中，响应报文是以BasicHttpResponse类来实现的。

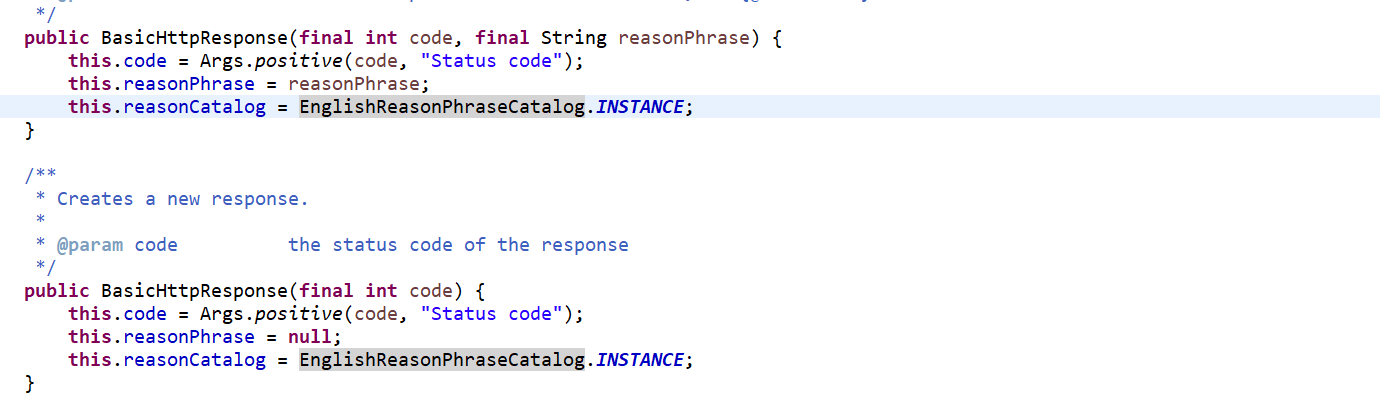


可以看到BasicResponse类定义了成员变量，分别对应的是响应报文中的状态行的几个参数。其中version成员变量已经在上面提到，代表协议的版本。但是其中更主要的是code成员变量以及reasonPhrase成员变量，二者分别对应的是响应报文的状态码和状态码的原因短语。很有趣的我们看到有一个locale成员变量，可以打开Locale类来看一下：



很明显，这个成员变量是作为响应的语言类型来进行相应的。因为不同的客户端可能对语言的响应格式要求是不一样的，所以同一份资源，需要对不同的语言格式进行解析加以反馈。

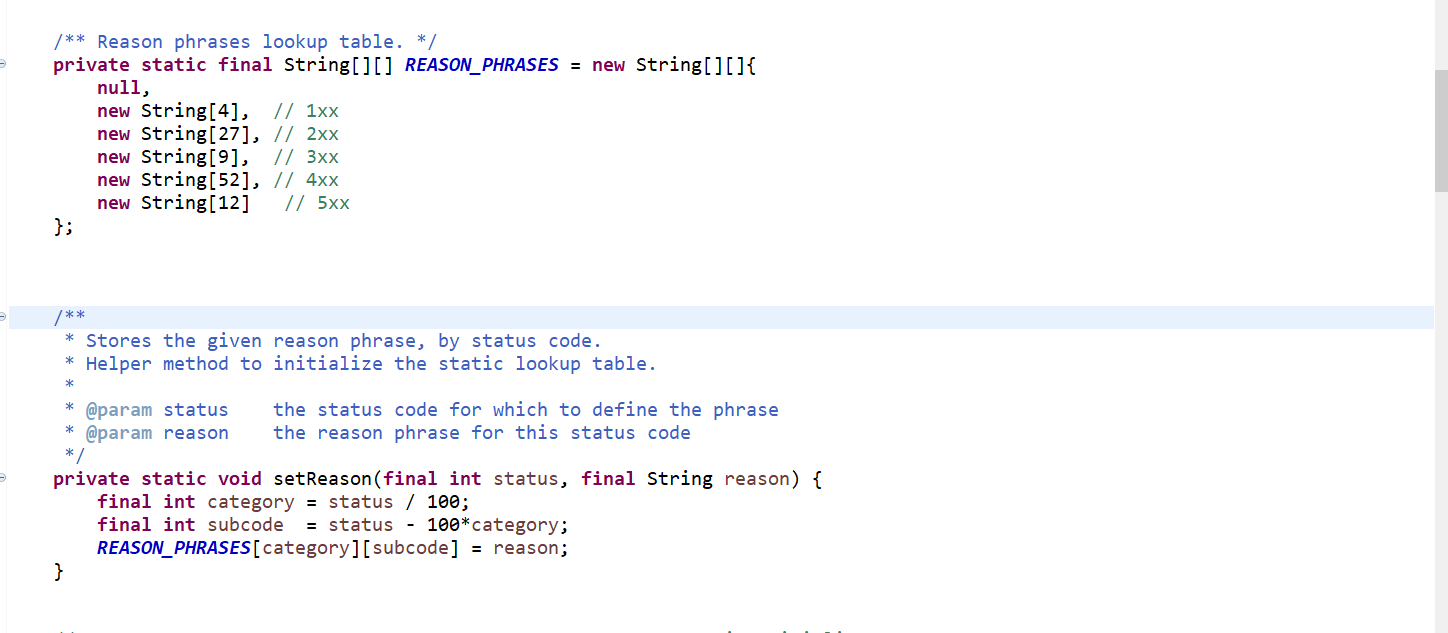
同样的，BasicHttoResponse类也有不同的构造方法：



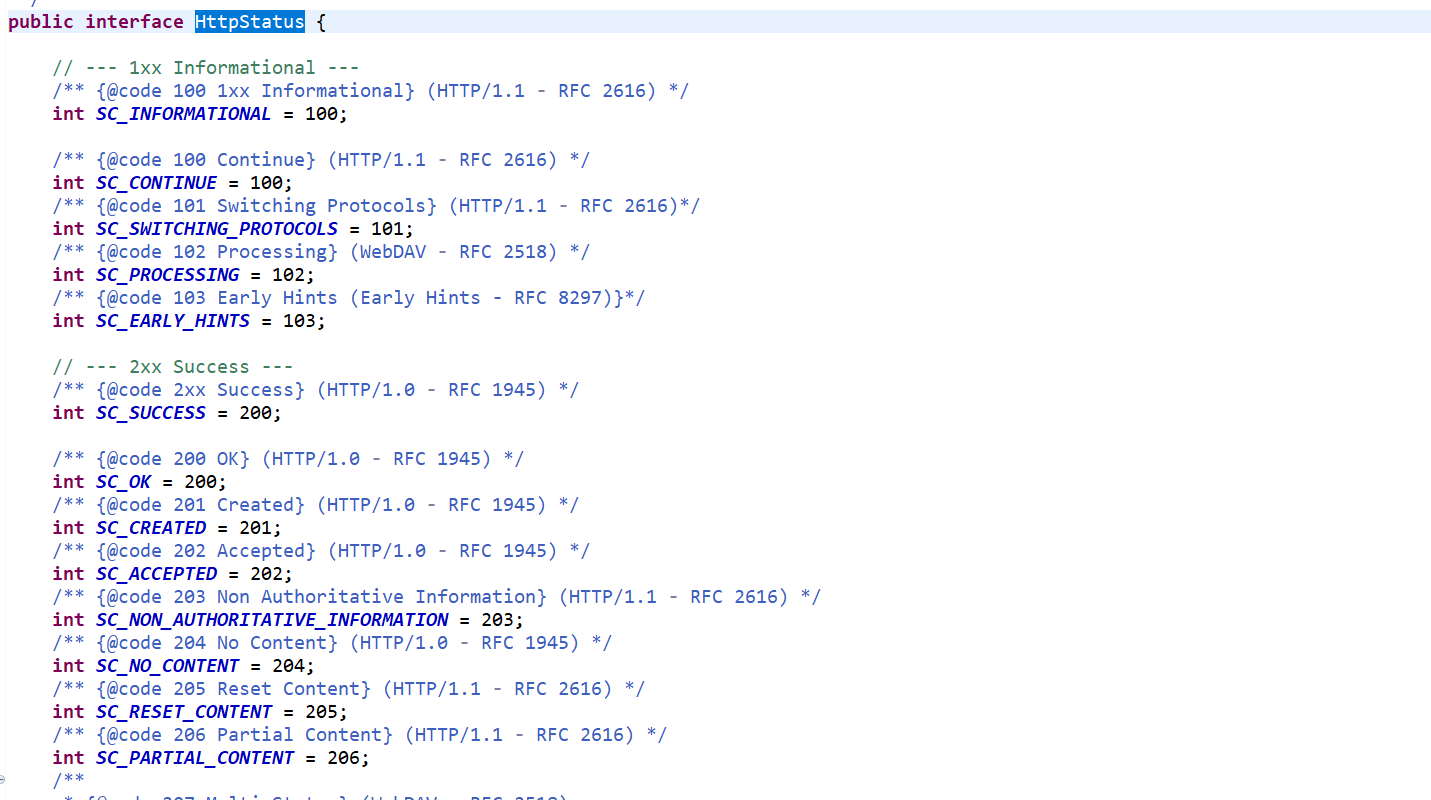
引起我注意的是是一个比较特殊的ReasonPhraseCatalog类的作用：

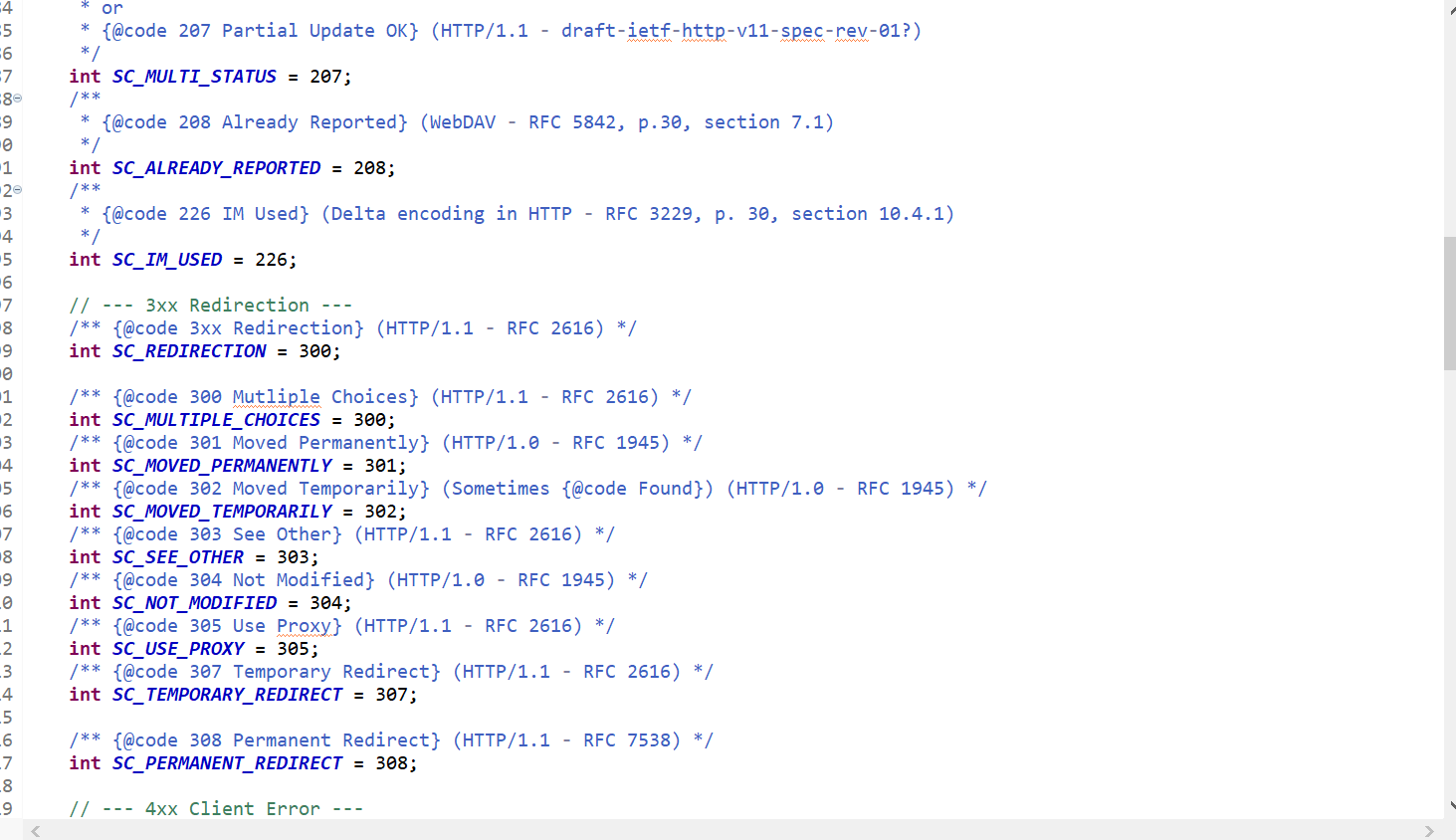


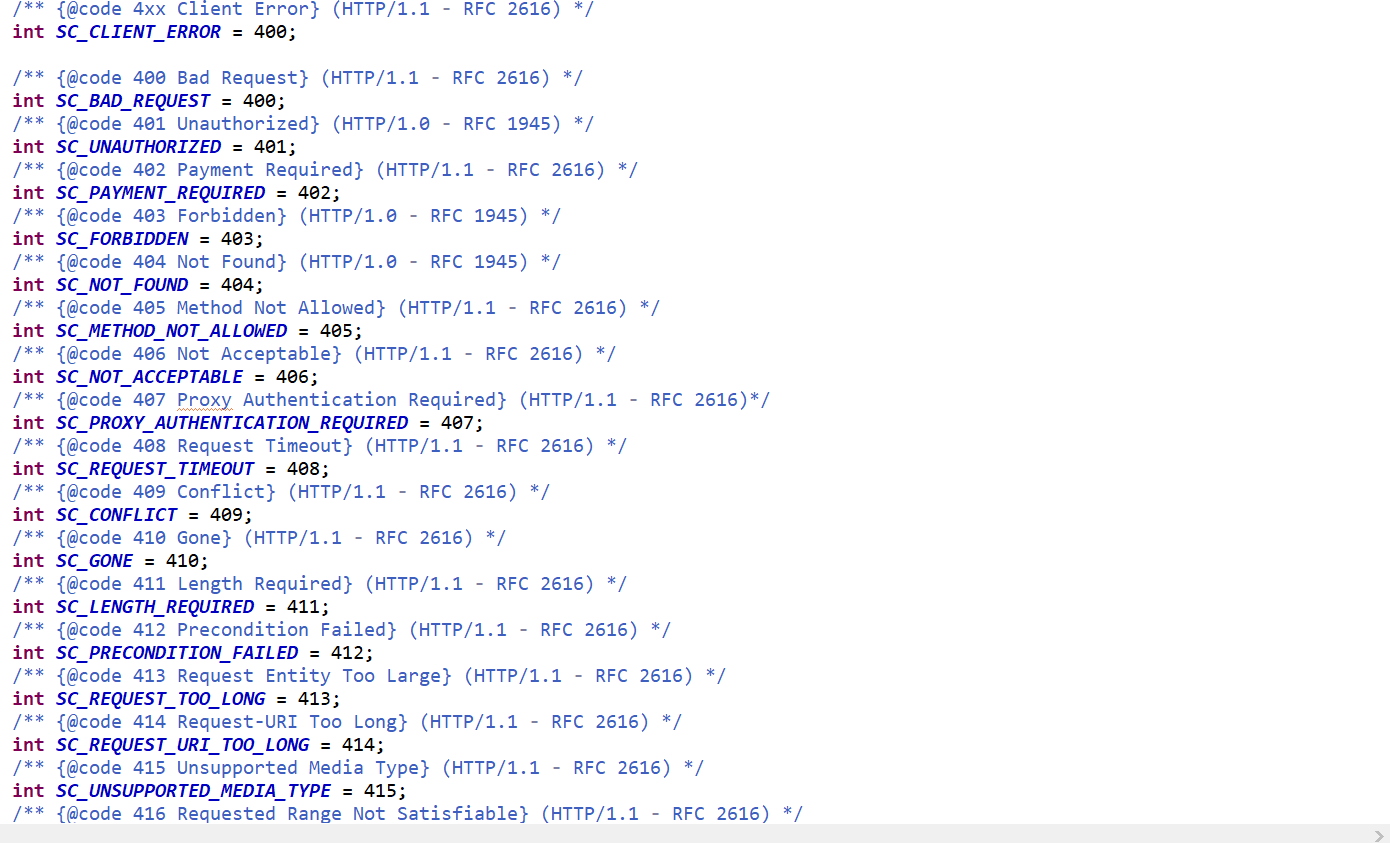
原来在这个类中，定义了一个静态代码块，他将不同的状态码（在HttpStatus接口中定义）对应的原因短语进行设置：



可以看到，在这个类中首先设置了一个REASON\_PHRASE字符串二位数组，数组的长度根据状态码首位来决定。setReason方法会利用状态码计算类别，将不同状态码对应的原因短语赋值到相应地字符串数组当中。HttpStatus接口定义如下：



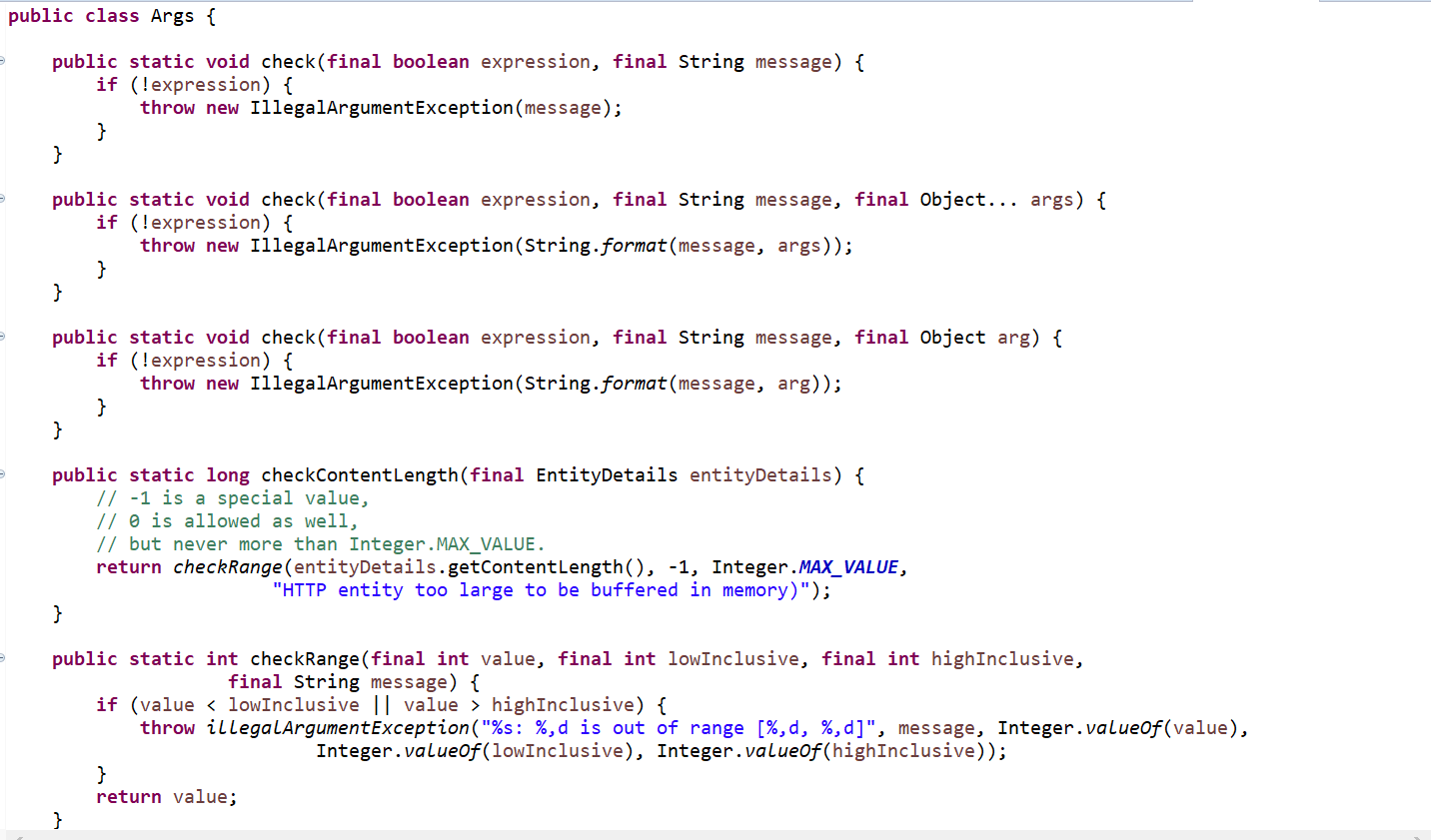




里面有我们常见的404 NOT FOUND的状态码，其实首位相同的状态码是一类，代表着相同的处理结果：

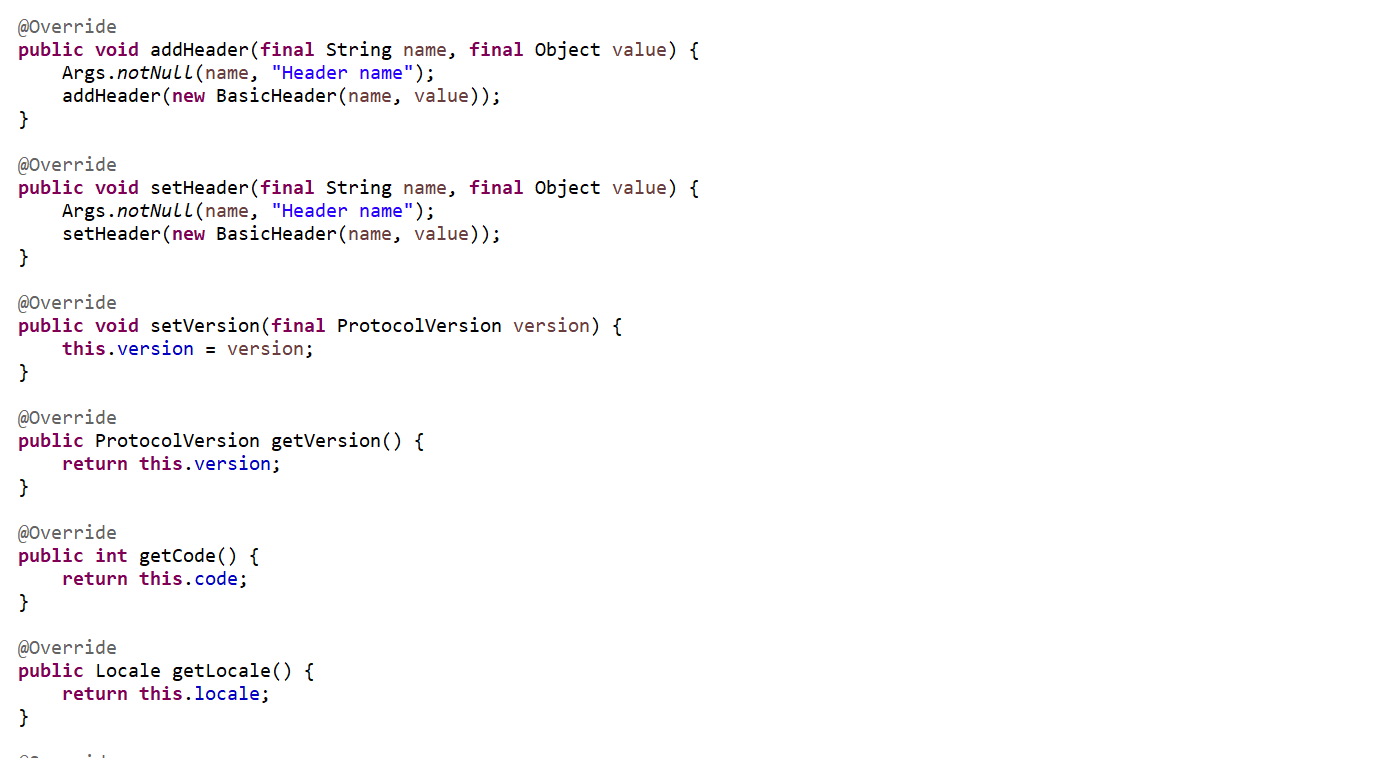


同样引起注意的是在Httpcore代码中随处可见的调用Args类中的静态方法，该类是为了对信息进行正确性核实的类：

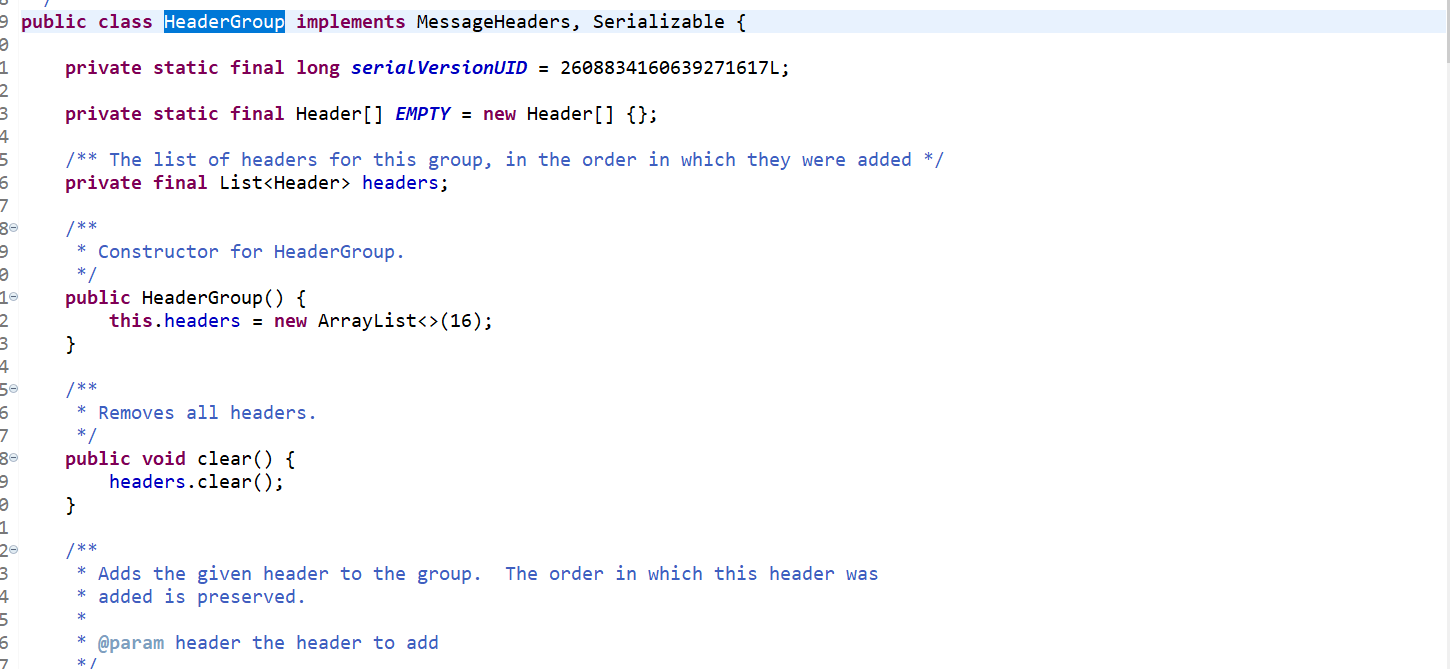


每当要对报文中的成员变量进行赋值，都需要不同的检测来防止错误。如果发生错误，则会在调用这个类中的静态函数时抛出例外。

BasicHttpResponse类同样有对响应报文信息的查询以及设置的成员方法：



但是无论是BasicHttpRequest和BasicHttpRespnse都是HeaderGroup类的子类，HeaderGroup类中定义了一个List成员变量来存放报文的首部，供外部添加：



最终的类图：

