非阻塞HTTP服务

信息与通信工程学院　胡 铮

huzheng@bupt.edu.cn

主要内容

- HTTP协议介绍
- 非阻塞的HTTP服务示例

思考

- 想浏览一个网站的时候，只要在浏览器的地址栏里输入网站的地址就可以了，例如输入"www.microsoft.com"，但是在浏览器的地址栏里面出现的却是"http://www.microsoft.com"，为什么会多出一个"http"吗？

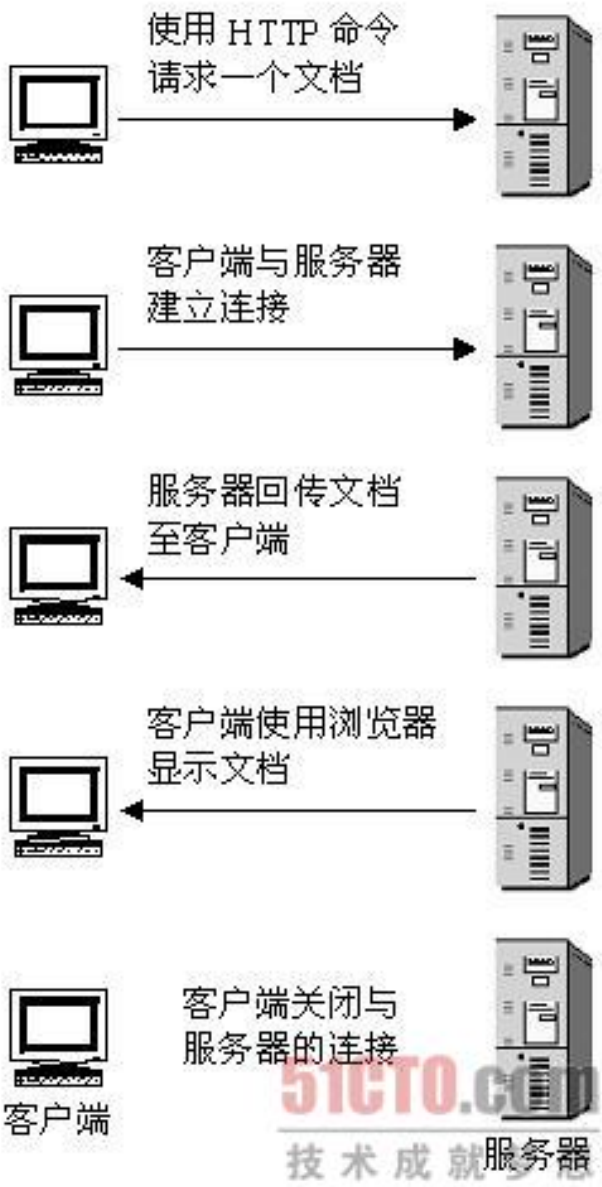
HTTP

- HTTP（Hypertext Transfer Protocol）协议，超文本传输协议。RFC2068、2616—HTTP/1.1
- HTTP-NG (Next Generation of HTTP)的建议已经提出。
- 网络应用层的协议，建立在TCP/IP协议基础上。使用可靠的TCP连接，默认端口80
- 主要服务于WWW服务，但不限于WWW，允许用户在统一的界面下，采用不同的协议访问不同的服务，如FTP、Archive、SMTP、NNTP、名字服务器和分布式对象管理。

HTTP协议的主要特点

- 支持客户/服务器模式。
- 简单快速：HTTP协议简单，HTTP服务器的程序规模小，因而通信速度很快。
- 灵活：允许传输任意类型的数据对象。
- 无连接：无连接的含义是限制每次连接只处理一个请求。采用这种方式可以节省传输时间。(http1.1里，提出了持久连接（persistentconnection）的概念，也就是说同一条 HTTP连接，可以依次处理多个请求。)
- 无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。

HTTP服务工作原理



- HTTP客户端使用HTTP命令向一个HTTP服务器发出HTTP请求；
- 如果该服务器在特定端口处接收到HTTP请求，就发送一个应答，并在客户端和服务器之间建立连接；
- HTTP服务器查找客户端所需的文档，如果HTTP服务器查找到所请求的文档，就会将所请求的文档传送给HTTP客户端；
- HTTP客户端接收到文档后，就将它显示出来；
- 当客户端浏览完成后，断开与服务器的连接。(HTTP/1.1的持久连接不是这样)

URL

- HTTP URL (URL是一种特殊类型的URI，包含了用于查找某个资源的足够的信息)的格式如下：
  http://host[“:”port][abs_path]
  http表示要通过HTTP协议来定位网络资源；
  host表示合法的Internet主机域名或者IP地址；
  port指定一个端口号，为空则使用缺省端口 80；
  abs_path指定请求资源的URI；如果URL中没有给出abs_path，那么当它作为请求URI时，必须以“/”的形式给出，通常这个工作 浏览器自动帮我们完成。
- eg:
  1、输入：www.bupt.edu.cn
  浏览器自动转换成：http://www.bupt.edu.cn/
  2、http://192.168.0.116:8080/index.jsp

HTTP消息

- HTTP 请求 Request
- HTTP 响应 Response

HTTP请求格式

- HTTP请求由3部分构成
  - 请求方式、URI、HTTP协议的版本
  - 请求报头（Request Header）
  - 请求正文（Request Content）

```
POST /hello.htm HTTP/1.1
Accept: image/gif, image/jpeg, */*
Refer: http://localhost/login.htm
Accept-Language: en, zh-cn; q=0.5
Content-type: application/x-www-forum-urlencoded
Accept-Encoding: gzip, deflate
……
```

HTTP请求模型

发送HTTP请求

1. 请求行：请求行由三个标记组成：请求报头、请求URI和HTTP版本，它们用空格分隔。

例如：GET /index.html HTTP/1.1

HTTP规范定义了8种可能的请求报头方法：

GET	请求获取请求URI所标识的资源
POST	在请求URI所标识的资源后附加新的数据
HEAD	请求获取由请求URI所标识的资源的响应消息报头
PUT	请求服务器存储一个资源，并用请求URI作为其标识
DELETE	请求服务器删除请求URI所标识的资源
TRACE	请求服务器回送收到的请求信息，主要用于测试或诊断
CONNECT	保留将来使用
OPTIONS	请求查询服务器的性能，或者查询与资源相关的选项和需求

HTTP请求报头

- **请求报头又称为元信息**，即信息的信息，利用元信息可以实现有条件的请求或应答。
- 由关键字/值对组成，每行一对，关键字和值用冒号（：）分隔。
- 请求报头通知服务器有关于客户端的功能和标识。

请求报头

- **Accept**：浏览器可接受的MIME类型。
- **Accept Charset**：浏览器可接受的字符集。
- **Accept Encoding**：浏览器能够进行解码的数据编码方式，比如gzip。
- **Accept Language**：浏览器所希望的语言种类，当服务器能够提供一种以上的语言版本时要用到.
- **Authorization**：授权信息，通常出现在对服务器发送的WWW - Authenticate头的应答中。
- **Connection**：表示是否需要持久连接，如Keep-Alive
- **Content Length**：表示请求消息正文的长度。
- **Cookie**：这是最重要的请求头信息之一。
- **From**：请求发送者的email地址，由一些特殊的Web客户程序使用，浏览器不会用到它。
- **Host**：初始URL中的主机和端口。

请求正文

- 请求头与请求正文之间必须以空行分割。
- 请求正文，使用POST传送数据，最常使用的是Content-Type和Content-Length请求头信息。

HTTP请求信息示例

```
POST /hello.htm HTTP/1.1
Accept: image/gif, image/jpeg, */*
Refer: http://localhost/login.htm
Accept-Language: en, zh-cn; q=0.5
Content-type: application/x-www-forum-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Connection: Keep-Alive

Username=jnpstudent&password=jnpstudent
```

HTTP响应格式

- HTTP响应由3部分构成
  - HTTP协议的版本、状态代码、描述
  - 响应报头（Response Header）
  - 响应正文（Response Content）

```
HTTP/1.1 200 OK
Date: Sat, 31 Dec 2005 23:59:59 GMT
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 122

<html>
<head>
<title>Wrox Homepage</title>
</head>
<body>
<!-- body goes here -->
</body>
</html>
```

实例1

- 浏览器发出请求
  GET /index.html HTTP/1.1

- 服务器返回响应:
  HTTP /1.1 200 OK
  Date: Apr 11 2006 15:32:08 GMT
  Server: Apache/2.0.46(win32)
  Content-Length: 119
  Content-Type: text/html

  <HTML>
  <HEAD>
  <LINK REL="stylesheet" HREF="index.css">
  </HEAD>
  <BODY>
  <IMG SRC="image/logo.png">
  </BODY>
  </HTML>

实例2

- 浏览器发出请求
  GET image/logo.png HTTP/1.1

- 服务器返回响应:
  HTTP /1.1 200 OK
  Date: Apr 11 2006 15:32:08 GMT
  Server: Apache/2.0.46(win32)
  Connection: Keep-alive, close
  Content-Length: 1280
  Content-Type: text/plane

  {Binary image data follows}

状态代码、描述

- HTTP响应码由三位十进制数字组成，它们出现在由HTTP服务器发送的响应的第一行。
- 响应码分五种类型，由它们的第一位数字表示：
  - 1xx：信息，请求收到，继续处理
  - 2xx：成功，行为被成功地接受、理解和采纳
  - 3xx：重定向，为了完成请求，必须进一步执行的动作
  - 4xx：客户端错误，请求包含语法错误或者请求无法实现
  - 5xx：服务器错误，服务器不能实现一种明显无效的请求

常见的状态代码和描述

- 200 OK：是最普遍的，也就是表示协议一切正常，凡是2开头的代码表示的都是成功进行中。
- 304 Not Modified：该资源在上次请求之后没有任何修改。这通常用于浏览器的缓存机制。
- 404 Not Found：这也是最普遍的吧，其实大多数错误就是所要求的资源无法得到，通常表示文件不存在。
- 403 Forbidden：表示服务器无法满足现在的请求，有可能是现在连接数太多等原因。
- 401 Unauthorized：未认证的请求，通常浏览器接受到这个状态值，就会弹出一个对话框，要求你输入密码。
- 500 InternalServerError：服务器内部错误，一般的原因是因为所执行的程序有错误，无法返回正确应答。
- 206 PartialContent：部分的内容，这个状态码表示下面传递的是部分的内容，也是断点续传的标准返回码。

响应报头

- 响应报头也和请求报头一样包含许多有用的信息，如服务器类型，正文类型和正文长度等

```
Server: nio/1.1                                  //服务器类型
Content-type: text/html; charset = GBK           //正文类型
Content-length: 102                              //正文长度
```

归纳一下

- HTTP消息都是由客户端到服务器的请求和服务器到客户端的响应组成。
- 请求消息和响应消息都是由开始行（对于请求消息，开始行就是请求行，对于响应消息，开始行就是状态行），消息报头（可选），空行（只有CRLF的行），消息正文（可选）组成。

归纳一下（请求和响应报头）

- 都是报头，包括
  - **普通报头**

    在普通报头中，有少数报头域用于所有的请求和响应消息，但并不用于被传输的实体，只用于传输的消息。
  - **请求报头**

    请求报头允许客户端向服务器端传递请求的附加信息以及客户端自身的信息
  - **响应报头**

    响应报头允许服务器传递不能放在状态行中的附加响应信息，以及关于服务器的信息和对Request-URI所标识的资源进行下一步访问的信息。
  - **实体报头**

    请求和响应消息都可以传送一个实体。一个实体由实体报头域和实体正文组成，但并不是说实体报头域和实体正文要在一起发送，可以只发送实体报头域。实体报头定义了关于实体正文（eg：有无实体正文）和请求所标识的资源的元信息。

响应正文

- 服务器返回的具体的文档，最常见的是HTML网页
- HTTP响应头与响应正文之间也必须用空行分隔

```
<html>
<head>
  <title>helloapp</title>
</head>
<body >
  <h1>hello</h1>
</body>
</html>
```

小试验：测试HTTP请求

- 在浏览器输入一个URL或者点击网页链接，浏览器就会生成一个HTTP请求，建立与远程HTTP服务器的连接，然后把HTTP请求发送到远程HTTP服务器
- SimpleHttpServer.java 创建了一个非常简单的HTTP服务器，接收客户HTTP请求，并把它打印到控制台。
- 启动SimpleHttpServer，使用浏览器输入URL

  http://localhost:8001/login.htm
- 修改login.htm文件，将

  <form name="loginForm" method="**post**" action="hello.htm">

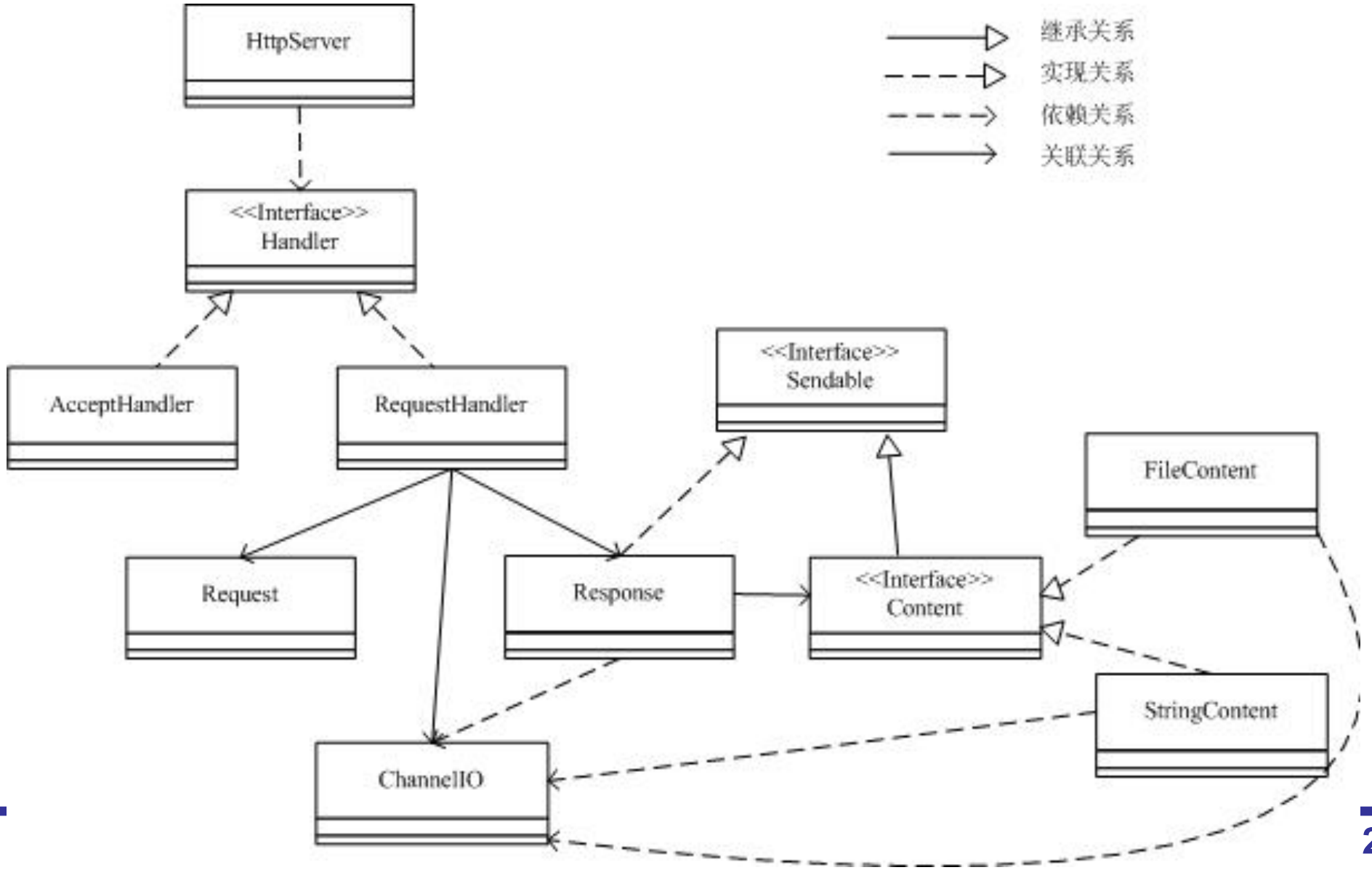
  改为

  <form name="loginForm" method="**get**" action="hello.htm">

  观察打印出来的请求

- 报头域不分大小写。
- 更深一步了解HTTP协议，可以查看RFC2616，在http://www.letf.org/rfc上找到该文件。

非阻塞HTTP服务器示例—对象模型



对象模型介绍（1）

- HttpServer类：
  - 服务器主程序，由它启动服务器
  - 仅启用单个主线程、非阻塞模式接收客户连接、收发数据
  - 依赖Handler接口实现其服务的具体功能
- AcceptHandler类
  - 实现了Handler接口，负责接收客户连接
- RequestHandler
  - 实现了Handler接口，负责接受客户的HTTP请求，对其解析，然后生成响应的HTTP响应，再把它发送给客户

对象模型介绍（2）

- Request类
  - 表示HTTP请求及相应的处理
- Response类
  - 表示HTTP响应及相应处理
- Content接口
  - 集成了Sendable接口，定义了可发送的HTTP响应正文的处理方法
- FileContent和StringContent
  - 实现了Content接口，表示HTTP响应正文的两种具体形式及相关处理方法
- ChannelIO类
  - 对SocketChannel进行了包装，增加了自动增长缓冲区容量的功能

The End