Network Project MyHttpDemo

2014 Dec 8, by whimsycwd

Overview

课程PJ, 模仿HTTP协议, 并Java Socket编程简单实现协议. 课程设计协议MyHTTP是一个简化版的HTTP协议, 是作为HTTP协议阅读后的一次总结和练手. 文档对Demo运用到的操作进行着重描述, 同时简单描述HTTP其他重要的操作. 本次项目做的Demo, 实现了一个支持HTTP 请求的服务端, 服务端保存了一份学生名单, 因为兼容http请求, 因此可接受来自浏览器的请求. 利用浏览器来作为客户端. 完成对学生学号,姓名的插入和插片的插入以及通过学号来查看学生信息.

Part I: 协议介绍

Abstract

HTTP是一个应用层的协议, 是一个用来构建分布协作, 处理富媒体的系统. 它是一个通用的无状态的协议.

What is MyHTTP

HTTP(Hypertext Transfer Protocol.) 是一个用来传输文件和其他数据的协议, 我们这里称其为Resource(资源). 浏览器通过HTTP服务器来把请求(Request)发送到HTTP服务器. 服务器进行处理, 然后将应答(Response) 返回用户. HTTP监听端口80, MyHTTP使用端口 10080.

MyHTTP 的一般结构

和HTTP一样, MyHTTP采用Client-Server模型. 客户端把请求信息发送到HTTP服务器, 服务器返回应答信息. MyHTTP在应答结束后,关闭连接. 因此说这是一个无状态的协议.

请求和应答的一般结构

- an initial line,
- zero or more header lines,
- a blank line (i.e. a CRLF by itself), and
- an optional message body (e.g. a file, or query data, or query output).

Request Line

请求行包括三个部分 1. 方法名, 2. 请求资源地址, 和请求的HTTP版本, (祝, 因为实现的服务器要兼容浏览器请求, 因此, 浏览器版本沿用HTTP请求, HTTP/1.1)

```
GET /path/to/file/index.html HTTP/1.1
```

HTTP的请求包括

1. GET
2. HEAD

3. POST
4. PUT
5. DELETE
6. TRACE
7. CONNECT

这边我只关注和实现了GET & POST. GET 用来请求资源. POST 用来新增资源. 提交请求到服务端.这里我实现的Demo里, 运用GET来进行对已插入信息的浏览. 运用POST来完成插入新条目, 上传图片的操作.

Response Line

应答行又叫状态行,

`HTTP/1.1 200 OK` or `HTTP/1.1 404 Not Found`

包括三部分分别是

1. 执行HTTP的HTTP版本
2. 状态码
3. 可读提示

状态包含:

- 1xx indicates an informational message only
- 2xx indicates success of some kind
- 3xx redirects the client to another URL
- 4xx indicates an error on the client's part
- 5xx indicates an error on the server's part

本项目里只用到了200 及 404 分别表示请求成功和没找到资源.

Header line

Header lines 提供了请求或者响应主题的信息. 本项目中用到了

1. Content-Type来区分,请求的资源是文本还是图片
2. Content-Length来, 对请求进行切割分段

详细一览

Request部分

Header	解释	示例
Accept	指定客户端能够接收的内容类型	Accept: text/plain, text/html|
Accept-Charset	浏览器可以接受的字符编码集。	Accept-Charset: iso-8859-5
Accept-Encoding	指定浏览器可以支持的web服务器返回内容压缩编码类型。	Accept-Encoding: compress, gzip
Accept-Language	浏览器可接受的语言	Accept-Language: en,zh
Cookie	HTTP请求发送时，会把保存在该请求域名下的所有cookie值一起发送给web服务器。	Cookie: $Version=1; Skin=new;
Content-Length	请求的内容长度	Content-Length: 348
Content-Type	请求的与实体对应的MIME信息	Content-Type: application/x-www-form-urlencoded
Date	请求发送的日期和时间	Date: Tue, 15 Nov 2010 08:12:31 GMT
From	发出请求的用户的	Email From: user@email.com
Host	指定请求的服务器的域名和端口号	Host: www.zcmhi.com
Referer	先前网页的地址，当前请求网页紧随其后,即来路	Referer: http://www.zcmhi.com/archives/71.html
User-Agent	User-Agent的内容包含发出请求的用户信息	User-Agent: Mozilla/5.0 (Linux; X11)

Response部分

Header	解释	示例
Content-Encoding	web服务器支持的返回内容压缩编码类型。	Content-Encoding: gzip
Content-Language	响应体的语言	Content-Language: en,zh
Content-Length	响应体的长度	Content-Length: 348
Content-Type	返回内容的MIME类型	Content-Type: text/html; charset=utf-8
Date	原始服务器消息发出的时间	Date: Tue, 15 Nov 2010 08:12:31 GMT

Message Body

一个HTTP信息包含一个主题, 主体的土体类型, 该如何解析通过header `Content-Type` & `Content-Length`来描述比如:

- `Content-Type = text/html or image/gif`. 那么客户端或者服务端就知道该如何解释拿到的数据
- `Content-Length` 用来标记body的字节树, 在Socket编程时, 会发现如果没有这个字节数, 那么会导致, 服务端不知道数据读完了没有,从而也就不知道合适开始处理数据并进行回复. 导致阻塞.

Part II MyHTTP Demo

总览

包含四个文件

1. Server.java, 服务器, 监听10080端口, 当有新的请求来的时候,启动新的线程来处理该请求
2. ServerThread, 具体的处理线程. 对请求进行分类,区分GET or POST区分图片请求
3. ServerCenter, 具体的业务逻辑, 把通过Socket传来的数据进行保存, 图片及姓名信息. 包括一个对返回HTML的渲染.
4. 封装简单的工具, 包括时间转化函数, byte转字符串, 读入byte数组及, 在处理表单提交是的, 处理富媒体的时候, 分段读取的工具类

遇到的困难

1. 图片处理是, 不得不对, 字节流进行处理, 很容易错误.
2. 在读入文件时, Java InputStream.read接口接口在buffer返回的数字是读到的内容数, 所以不是说打开一个很大的buffer就可以读到.

```
byte [] bytes = new byte[contentLenth];

is.read(bytes); 不一定能够读到contentLength个字节.
```

3. 在服务端接受客户传来的数据时, 如何判定终止比较trick. 尝试了通过特殊的`bye`字符串来进行通讯, 但是这样实现出来不能够兼容浏览器请求, 最后想到了可以通过Content-Length header的信息来结束通讯,关闭socket

如何使用

1. git clone https://github.com/whimsycwd/MyHTTPDemo.git
2. 运行Server.java
3. 打开浏览器127.0.0.1:10080/index.html可查看创建页面, 填写姓名, 学号, 上传图片
4. 创建成功后, 比如创建了11300240057学号, 127.0.0.1:10080/queryById/1003545371

可得到查看页面.

参考文献

1. http://jmarshall.com/easy/http/
2. http://tools.ietf.org/html/rfc2616
3. https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html
4. http://bbs.csdn.net/topics/390290625?page=1#post-393011354