P3

2018年4月24日　　15:07

Ref:
https://blog.csdn.net/qq_20480611/article/details/50982688
Rio:
https://www.cnblogs.com/cknightx/p/7518085.html
GET和POST：
https://sunshinevvv.coding.me/blog/2017/02/09/HttpGETvsPOST/

GET把参数包含在URL中，POST通过request body传递参数
GET产生一个TCP数据包；POST产生两个TCP数据包
对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。
1. GET与POST都有自己的语义，不能随便混用。
2. 如果网络环境好的话，发一次包的时间和发两次包的时间差别基本可以无视。如果网络环境差的话，两次包的TCP在验证数据包完整性上，有非常大的优点。
3. 并不是所有浏览器都会在POST中发送两次包，Firefox就只发送一次。

- GET在浏览器回退时是无害的，而POST会再次提交请求。
- GET产生的URL地址可以被Bookmark，而POST不可以。
- GET请求会被浏览器主动cache，而POST不会，除非手动设置。
- GET请求只能进行url编码，而POST支持多种编码方式。
- GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。
- GET请求在URL中传送的参数是有长度限制的，而POST么有。
- 对参数的数据类型，GET只接受ASCII字符，而POST没有限制。
- GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。
- GET参数通过URL传递，POST放在Request body中。

原 **CSAPP Tiny web 服务器源码分析及搭建运行**

标签： csapp　服务器　源码

2016年03月25日 17:21:50　　2411人阅读　评论(1)　收藏　举报

分类：　CSAPP（2）　linux（83）

目录(?)　[+]

1. Web基础

web客户端和服务器之间的交互使用的是一个基于文本的应用级协议HTTP(超文本传输协议)。一个web客户端(即浏览器)打开一个到服务器的因特网连接，并且请求某些内容。服务器响应所请求的内容，然后关闭连接。浏览器读取这些内容，并把它显示在屏幕上。

对于web客户端和服务器而言，内容是与一个MIME类型相关的字节序列。常见的MIME类型：

MIME类型	描述
text/html	HTML页面
text/plain	无格式文本
image/gif	GIF格式编码的二进制图像

text/plain	无格式文本
image/gif	GIF格式编码的二进制图像
image/jpeg	JPEG格式编码的二进制图像

web服务器以两种不同的方式向客服端提供内容：
1. 静态内容：取一个磁盘文件，并将它的内容返回给客户端
2. 动态内容：执行一个可执行文件，并将它的输出返回给客户端

统一资源定位符：URL

> http://www.google.com:80/index.html

表示因特网主机 www.google.com 上一个称为 index.html 的HTML文件，它是由一个监听端口80的Web服务器所管理的。 HTTP默认端口号为80

可执行文件的URL可以在文件名后包括程序参数， “？”字符分隔文件名和参数，而且每个参数都用“&”字符分隔开，如：

> http://www.ics.cs.cmu.edu:8000/cgi-bin/adder?123&456

表示一个 /cgi-bin/adder 的可执行文件，带两个参数字符串为 123 和 456

确定一个URL指向的是静态内容还是动态内容没有标准的规则，常见的方法就是把所有的可执行文件都放在 cgi-bin 目录中

2. HTTP

HTTP标准要求每个文本行都由一对回车和换行符来结束

```
1  unix> telnet www.aol.com 80          Client: open connection to server
2  Trying 205.188.146.23...             Telnet prints 3 lines to the terminal
3  Connected to aol.com.
4  Escape character is '^]'.
5  GET / HTTP/1.1                       Client: request line
6  Host: www.aol.com                    Client: required HTTP/1.1 header
7                                       Client: empty line terminates headers
8  HTTP/1.0 200 OK                      Server: response line
9  MIME-Version: 1.0                    Server: followed by five response headers
10 Date: Mon, 8 Jan 2010 4:59:42 GMT
11 Server:  Apache-Coyote/1.1
12 Content-Type: text/html              Server: expect HTML in the response body
13 Content-Length: 42092                Server: expect 42,092 bytes in the response body
14                                      Server: empty line terminates response headers
15 <html>                               Server: first HTML line in response body
16 ...                                  Server: 766 lines of HTML not shown
17 </html>                              Server: last HTML line in response body
18 Connection closed by foreign host.   Server: closes connection
19 unix>                                Client: closes connection and terminates
```

图 11-23　一个服务静态内容的 HTTP 事务

(1)HTTP请求

一个HTTP请求：一个请求行(request line) 后面跟随0个或多个请求报头(request header), 再跟随一个空的文本行来终止报头

请求行：`<method> <uri> <version>`

来终止报头

请求行： `<method> <uri> <version>`
HTTP支持许多方法，包括 GET，POST，PUT，DELETE，OPTIONS，HEAD，TRACE。
URI是相应URL的后缀，包括文件名和可选参数
version 字段表示该请求所遵循的HTTP版本

请求报头： `<header name> : <header data>` 为服务器提供了额外的信息，例如浏览器的版本类型
HTTP 1.1中 一个IP地址的服务器可以是 多宿主主机，例如 www.host1.com www.host2.com 可以存在于同一服务器上。
HTTP 1.1 中必须有 host 请求报头，如 host:www.google.com:80 如果没有这个host请求报头，每个主机名都只有唯一IP，IP地址很快将用尽。

(2)HTTP响应

一个HTTP响应：一个响应行(response line) 后面跟随0个或多个响应报头(response header)，再跟随一个空的文本行来终止报头，最后跟随一个响应主体(response body)

响应行： `<version> <status code> <status message>`
status code 是一个三位的正整数

状态代码	状态消息	描述
200	成功	处理请求无误
301	永久移动	内容移动到位置头中指明的主机上
400	错误请求	服务器不能理解请求
403	禁止	服务器无权访问所请求的文件
404	未发现	服务器不能找到所请求的文件
501	未实现	服务器不支持请求的方法
505	HTTP版本不支持	服务器不支持请求的版本

两个最重要的响应报头：
Content-Type 告诉客户端响应主体中内容的MIME类型
Content-Length 指示响应主体的字节大小
响应主体中包含着被请求的内容。

3.服务动态内容

(1) 客户端如何将程序参数传递给服务器

GET请求的参数在URI中传递， “? ”字符分隔了文件名和参数，每个参数都用一个”&”分隔开，参数中不允许有空格，必须用字符串“%20”来表示
HTTP POST请求的参数是在请求主体中而不是 URI中传递的

(2)服务器如何将参数传递给子进程

```
GET /cgi-bin/adder?123&456 HTTP/1.1
```

它调用 fork 来创建一个子进程，并调用 execve 在子进程的上下文中执行 /cgi-bin/adder 程序

在调用 execve 之前，子进程将CGI环境变量 QUERY_STRING 设置为”123&456”， adder 程序在运行时可以用unix gete

它调用 fork 来创建一个子进程，并调用 execve 在子进程的上下文中执行 /cgi-bin/adder 程序

在调用 execve 之前，子进程将CGI环境变量 QUERY_STRING 设置为"123&456"， adder 程序在运行时可以用unix getenv 函数来引用它

(3)服务器如何将其他信息传递给子进程

环境变量	描述
QUERY_STRING	程序参数
SERVER_PORT	父进程侦听的端口
REQUEST_METHOD	GET 或 POST
REMOTE_HOST	客户端的域名
REMOTE_ADDR	客户端的点分十进制IP地址
CONTENT_TYPE	只对POST而言，请求体的MIME类型
CONTENT_LENGTH	只对POST而言，请求体的字节大小

(4) 子进程将它的输出发送到哪里

一个CGI程序将它的动态内容发送到标准输出，在子进程加载并运行CGI程序之前，它使用UNIX dup2 函数将它标准输出重定向到和客户端相关连的已连接描述符
因此，任何CGI程序写到标准输出的东西都会直接到达客户端