

冬冬他哥哥

博客园 首页 新随笔 联系 订阅 管理

随笔 - 448 文章 - 0 评论 - 23 阅读 - 56万

公告

昵称： 谢呈勛
园龄： 7年7个月
粉丝： 78
关注： 28
+加关注

< 2023年4月 >

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

搜索

找找看

谷歌搜索

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

C#(6)

wpf(4)

NSIS(2)

image(2)

linux(2)

qt(2)

C#调用C++(2)

平台调用(2)

wxs(2)

win10(1)

更多

随笔分类

C#(67)

C#异步编程(22)

C#中的事件(6)

C#中的委托(4)

C++(101)

C++实现设计模式(1)

Http协议之详解

http协议之详解 (点我)

http协议之https (点我)

http协议之libcurl (点我)

一、http协议的特性

http协议是建立在TCP/IP协议之上应用层协议，默认端口为80,8080

http协议的的特点是 无状态，无连接

二、http协议的请求

利用抓包工具httpwatch可以获取报文

http协议的报文传输的是ASCII码，在TCP/IP协议之上，主要主要分为三部分

请求行、请求头、请求体

请求行

第一行，包含三个信息：请求方式，url，http协议版本

GET 请求

GET /books/?sex=man&name=Professional HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)

Gecko/20050225 Firefox/1.0.1

Connection: Keep-Alive

POST 请求

POST / HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)

Gecko/20050225 Firefox/1.0.1

Content-Type: application/x-www-form-urlencoded

Content-Length: 40

Connection: Keep-Alive

sex=man&name=Professional

区别：

1、url可见性：

get, 参数url可见；

COM技术(4)
C语言(1)
Docker技术(9)
Linux(16)
Nodejs(3)
NSIS(6)
QT(3)
react框架开发(3)
RPA(1)
更多

随笔档案

2023年3月(5)
2023年2月(1)
2022年12月(1)
2022年11月(5)
2022年10月(4)
2022年9月(11)
2022年4月(3)
2022年3月(2)
2022年2月(2)
2022年1月(5)
2021年12月(4)
2021年11月(2)
2021年10月(6)
2021年9月(14)
2021年8月(6)
更多

阅读排行榜

1. c++引用lib和dll的方法总结(21977)
2. LPCTSTR 用法(13780)
3. C#中的异步和同步(13545)
4. C# Task的暂停与终止(12582)
5. c++11の异步方法 及线程间通信(12312)

评论排行榜

1. .Net环境下调用ProtoBuf(5)
2. c++のurlmon实现下载文件并进度回调(2)
3. .Net的内存回收(1)
4. WPF 中ContextMenu 在mvvm模式中的绑定问题(1)
5. WPF的动态资源和静态资源(1)

推荐排行榜

1. wpfのuri (让你完全明白wpf的图片加载方式以及URI写法) (4)
2. WPF自定义控件 (一) の控件分类 (3)

post, url参数不可见

2、数据传输上：

get, 通过拼接url进行传递参数；

post, 通过body体传输参数

3、缓存性：

get请求是可以缓存的

post请求不可以缓存

4、后退页面的反应

get请求页面后退时, 不产生影响

post请求页面后退时, 会重新提交请求

5、传输数据的大小

get一般传输数据大小不超过2k-4k (根据浏览器不同, 限制不一样, 但相差不大)

post请求传输数据的大小根据php.ini 配置文件设定, 也可以无限大。

6、安全性

这个也是最不好分析的, 原则上post肯定要比get安全, 毕竟传输参数时url不可见, 但也挡不住部分人闲的没事在那抓包玩。安全性个人觉得是没多大区别的, 防君子不防小人就是这个道理。对传递的参数进行加密, 其实都一样。

本质区别：

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) 。

请求头

浏览器向服务器发送一些状态数据, 标识数据等等

一个信息一行, 包括信息名: 信息值 按行分隔



```
User-Agent: firefox//表示发送请求的浏览器（请求代理端）是firefox
Host: shop.100.com//表示请求的主机域名（基于域名的虚拟主机就是靠这个头判断的）
Cookie:name=itcast//浏览器携带的cookie数据。
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
Connection: Keep-Alive
```



注意，请求头信息，需要使用一个空行结束！

请求主体

请求代理端向服务器端, 发送的请求数据！

典型的就是POST形式发送的表单数据！

- 3. C# AOP的实现 (利用.Net自带的轻量级框架RealProxy) (2)
- 4. WPF可视对象变换 (RenderTransform) -----RotateTransform、TranslateTransform、ScaleTransform(2)
- 5. c++智能指针 (unique_ptr、shared_ptr、weak_ptr、auto_ptr) (2)

最新评论

- 1. Re:C# Task的暂停与终止
if
(_cancellationToken.IsCancellationRequested) { return; } 如果后面的逻辑需要执行10分钟猜完毕，岂不是要等待10分钟猜能正在取消？ 这个cancelIT...
--wgscd
- 2. Re:WPF的动态资源和静态资源
666 这份介绍的太详细了
--彭二狗的牵引绳
- 3. Re:WPF 中ContextMenu 在mvvm模式中的绑定问题
您好！请问你下这个上下文动态绑定的有没有源码，可以发一份吗？
564178640@qq.com
--天空之上的鱼儿
- 4. Re:WPF自定义控件 (四) の自定义控件
能把源码发一下吗？ 多谢
--zhhu
- 5. Re:.Net的内存回收
string是引用类型，所有引用类型的默认值都是null。
--羽烈

get请求，没有请求主体部分！ get数据是在请求行中的url上进行传递的！

三、http协议的响应

响应包括：响应行、响应头、响应体



```
HTTP/1.1 200 OK
Date: Tue, 19 Nov 2013 03:08:55 GMT
Server: Apache/2.2.22 (Win32) PHP/5.3.13
X- Powered -By: PHP/5.3.13
Content-Length: 16
Content-Type: text/html
```



响应行

响应行包括：协议版本、状态码、状态消息

典型的：

- 1xx:消息
 - 2xx:成功
 - 3xx:请求被重定向
 - 4xx:浏览器端错误
 - 5xx:服务器端错误
- 典型：
- 500 服务器内部错误
 - 404 请求的页面没有找到
 - 403 没有权限
 - 200 请求成功

响应头

Content-Type: text/html 内容类型，告知浏览器接下来发送的响应主体数据是什么格式！

Content-Length: 响应主体数据的长度！

Date: 响应的时间。GMT时间！

响应主体

主要的响应数据，在浏览器的主体区域显示的数据都是相应主体！

注意，每行，包括相应行和响应头，都需要一个 \r\n结尾

四、持久连接

HTTP 协议采用“请求-应答”模式，当使用普通模式，即非 Keep-Alive 模式时，每个请求/应答客户和服务器都要新建一个连接，完成之后立即断开连接（HTTP 协议为无连接的协议）；当使用 Keep-Alive 模式（又称持久连接、连接重用）时，Keep-Alive 功能使客户端到服务器端的连接持续有效，当出现对服务器的后继请求时，Keep-Alive 功能避免了建立或者重新建立连接。

在 HTTP 1.0 版本中，并没有官方的标准来规定 Keep-Alive 如何工作，因此实际上它是被附加到 HTTP 1.0协议上，如果客户端浏览器支持 Keep-Alive，那么就在HTTP请求头中添加一个字段 Connection: Keep-Alive，当服务器收到附带有 Connection: Keep-Alive 的请求时，它也会在响应头中添加一个同样的字段来使用 Keep-Alive。这样一来，客户端和服务端之间的HTTP连接就会

被保持，不会断开（超过 Keep-Alive 规定的时间，意外断电等情况除外），当客户端发送另外一个请求时，就使用这条已经建立的连接。

在 HTTP 1.1 版本中，默认情况下所有连接都被保持，如果加入 "Connection: close" 才关闭。目前大部分浏览器都使用 HTTP 1.1 协议，也就是说默认都会发起 Keep-Alive 的连接请求了，所以是否能完成一个完整的 Keep-Alive 连接就看服务器设置情况。

由于 HTTP 1.0 没有官方的 Keep-Alive 规范，并且也已经基本被淘汰，以下讨论均是针对 HTTP 1.1 标准中的 Keep-Alive 展开的。

注意：

- HTTP Keep-Alive 简单说就是保持当前的TCP连接，避免了重新建立连接。
- HTTP 长连接不可能一直保持，例如 `Keep-Alive: timeout=5, max=100`，表示这个 TCP 通道可以保持5秒，max=100，表示这个长连接最多接收100次请求就断开。
- HTTP 是一个无状态协议，这意味着每个请求都是独立的，Keep-Alive 没能改变这个结果。另外，Keep-Alive 也不能保证客户端和服务端之间的连接一定是活跃的，在 HTTP1.1 版本中也如此。唯一能保证的就是当连接被关闭时你能得到一个通知，所以不应该让程序依赖于 Keep-Alive 的保持连接特性，否则会有意想不到的后果。
- 使用长连接之后，客户端、服务端怎么知道本次传输结束呢？两部分：1. 判断传输数据是否达到了Content-Length 指示的大小；2. 动态生成的文件没有 Content-Length，它是分块传输（chunked），这时候就要根据 chunked 编码来判断，chunked 编码的数据在最后有一个空 chunked 块，表明本次传输数据结束。

五、分块传输

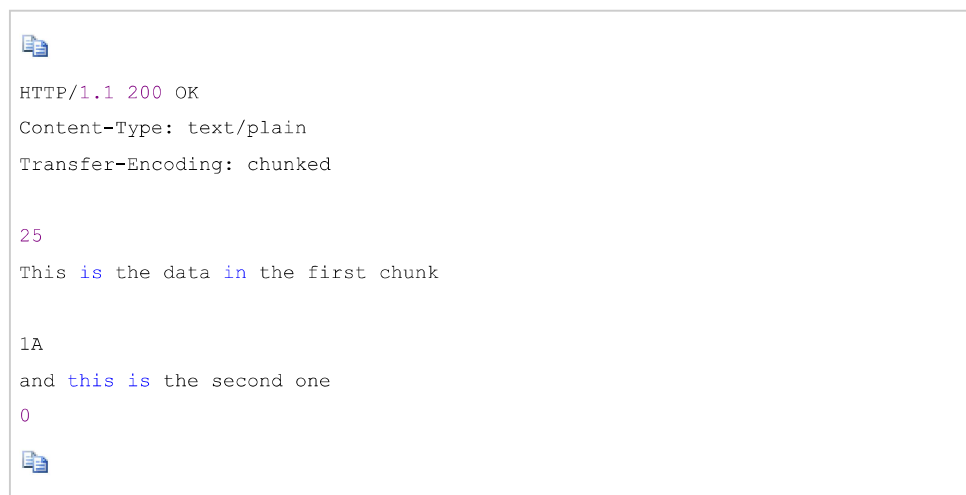
Transfer-Encoding

Transfer-Encoding 是一个用来标示 HTTP 报文传输格式的头部值。尽管这个取值理论上可以有多种，但是当前的 HTTP 规范里实际上只定义了一种传输取值——chunked。

如果一个HTTP消息（请求消息或应答消息）的Transfer-Encoding消息头的值为chunked，那么，消息体由数量未定的块组成，并以最后一个大小为0的块为结束。

每一个非空的块都以该块包含数据的字节数（字节数以十六进制表示）开始，跟随一个CRLF（回车及换行），然后是数据本身，最后块CRLF结束。在一些实现中，块大小和CRLF之间填充有白空格（0x20）。

最后一块是单行，由块大小（0），一些可选的填充白空格，以及CRLF。最后一块不再包含任何数据，但是可以发送可选的尾部，包括消息头字段。消息最后以CRLF结尾。



```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked

25
This is the data in the first chunk

1A
and this is the second one

0
```

六、HTTP Pipelining（HTTP 管线化）

默认情况下 HTTP 协议中每个传输层连接只能承载一个 HTTP 请求和响应，浏览器会在收到上一个请求的响应之后，再发送下一个请求。在使用持久连接的情况下，某个连接上消息的传递类似于

请求1 -> 响应1 -> 请求2 -> 响应2 -> 请求3 -> 响应3 。

HTTP Pipelining（管线化）是将多个 HTTP 请求整批提交的技术，在传送过程中不需等待服务端的回应。使用 HTTP Pipelining 技术之后，某个连接上的消息变成了类似这样

请求1 -> 请求2 -> 请求3 -> 响应1 -> 响应2 -> 响应3 。

注意下面几点：

- 管线化机制通过持久连接（persistent connection）完成，仅 HTTP/1.1 支持此技术（HTTP/1.0不支持）
- 只有 GET 和 HEAD 请求可以进行管线化，而 POST 则有所限制
- 初次创建连接时不应启动管线机制，因为对方（服务器）不一定支持 HTTP/1.1 版本的协议
- 管线化不会影响响应到来的顺序，如上面的例子所示，响应返回的顺序并未改变
- HTTP /1.1 要求服务器端支持管线化，但并不要求服务器端也对响应进行管线化处理，只是要求对于管线化的请求不失败即可
- 由于上面提到的服务器端问题，开启管线化很可能并不会带来大幅度的性能提升，而且很多服务器端和代理程序对管线化的支持并不好，因此现代浏览器如 Chrome 和 Firefox 默认并未开启管线化支持

七、http协议的安全性

跨站请求伪造（CSRF篡改本地信息）

跨站脚本攻击（XSS，就是在html总嵌入js脚本）

HTTP头部攻击

OS命令攻击

SQL注入攻击

目录攻击

Dos攻击

Dos攻击主要有两种

- 1)是集中利用访问请求，或者攻击者刻意制造访问请求，造成资源过载，资源耗尽，服务停止
- 2)通过攻击安全漏洞使服务停止

八、会话跟踪

1. 什么是会话？

客户端打开与服务器的连接发出请求到服务器响应客户端请求的全过程称之为会话。

2. 什么是会话跟踪？

会话跟踪指的是对同一个用户对服务器的连续的请求和接受响应的监视。

3. 为什么需要会话跟踪？

浏览器与服务器之间的通信是通过HTTP协议进行通信的，而HTTP协议是“无状态”的协议，它不能保存客户的信息，即一次响应完成之后连接就断开了，下一次的请求需要重新连接，这样就需要判断是否是同一个用户，所以才有会话跟踪技术来实现这种要求。

1. 会话跟踪常用的方法:

1. URL 重写

URL(统一资源定位符)是Web上特定页面的地址，URL重写的技术就是在URL结尾添加一个附加数据以标识该会话,把会话ID通过URL的信息传递过去，以便在服务器端进

行识别不同的用户。

2. 隐藏表单域

将会话ID添加到HTML表单元素中提交到服务器，此表单元素并不在客户端显示

3. Cookie

Cookie 是Web 服务器发送给客户端的一小段信息，客户端请求时可以读取该信息发送到服务器端，进而进行用户的识别。对于客户端的每次请求，服务器都会将 Cookie 发送到客户端,在客户端可以进行保存,以便下次使用。

客户端可以采用两种方式保存这个 Cookie 对象，一种方式是保存在客户端内存中，称为临时 Cookie，浏览器关闭后这个 Cookie 对象将消失。另外一种方式是保存在客户机的磁盘上，称为永久 Cookie。以后客户端只要访问该网站，就会将这个 Cookie 再次发送到服务器上，前提是这个 Cookie 在有效期内，这样就实现了对客户的跟踪。

Cookie 是可以被客户端禁用的。

4. Session:

每一个用户都有一个不同的 session，各个用户之间是不能共享的，是每个用户所独享的，在 session 中可以存放信息。

在服务器端会创建一个 session 对象，产生一个 sessionID 来标识这个 session 对象，然后将这个 sessionID 放入到 Cookie 中发送到客户端，下一次访问时，sessionID 会发送到服务器，在服务器端进行识别不同的用户。

Session 的实现依赖于 Cookie，如果 Cookie 被禁用，那么 session 也将失效。

九、cookie的传递

HTTP 头中

Cookie: Cookie变量名=cookie值;expire=到期时间;path=作用路径;domain=作用域

[http协议之详解（点我）](#)

[http协议之https（点我）](#)

[http协议之libcurl（点我）](#)

分类: [网络](#)

好文要顶

关注我

收藏该文

谢呈勛

粉丝 - 78 关注 - 28

00

[+加关注](#)

« [上一篇：.net core关注点](#)

» [下一篇：Http协议之Https](#)

posted @ 2020-07-06 15:36 谢呈勛 阅读(2314) 评论(0) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

发表评论

编辑 预览

B

支持 Markdown



自动补全

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

【推荐】博客园人才出海服务第一站，联合日本好融社推出日本IT人才移民直通车
【推荐】中国云计算领导者：阿里云轻量应用服务器2核2G，新用户低至108元/年



编辑推荐:

- 从内核源码看 slab 内存池的创建初始化流程
- [C#表达式树] 最完善的表达式树 Expression.Dynamic 的玩法
- 有意思的气泡 Loading 效果
- Three.js 进阶之旅：全景漫游-高阶版在线看房
- 解 Bug 之路 - 应用 999 线升高

阅读排行:

- 园子的商业化努力-困境求助：开设捐助通道
- 程序员的哲学
- 【从零开始】Docker Desktop：听说你小子要玩我
- AI时代下普通小程序员的想法
- [MAUI]模仿微信“按住-说话”的交互实现

历史上的今天:

- 2016-07-06 C#の单例模式
- 2016-07-06 如何把drawing图像转换成wpf控件的source
- 2016-07-06 byte,bitmap,image互转

Copyright © 2023 谢呈勳

Powered by .NET 7.0 on Kubernetes