

没事写着玩。

昵称：王永东gg

园龄：2年1个月

粉丝：0

关注：0

+加关注

2016年11月						
<	一	二	三	四	五	>
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	7	8	9	10

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

- Composer(1)
- instr(1)
- jquery(1)
- laravel(1)
- like(1)
- LOCATE(1)
- mobile(1)
- mysql(1)
- POSITION(1)
- swipeleft(1)
- 更多

- 文章分类
- composer(4)

html+css(7)

JavaScript(16)

MYSQL(18)

PHP(61)

开发工具(3)

开发环境(20)

关于HTTP协议和它的无状态性质

第一部分 HTTP协议的内容和特点

HTTP（HyperTextTransferProtocol）是超文本传输协议的缩写，它用于传送WWW方式的数据，关于HTTP协议的详细内容请参考RFC2616。HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于1990年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在WWW中使用的是HTTP/1.0的第六版，HTTP/1.1的规范化工作正在进行之中，而且HTTP-NG(NextGenerationofHTTP)的建议已经提出。今天在实验室把相关的RFC文档仔细看了一下，对HTTP协议有了一个详细的认识，本文将做一个详细的介绍。

1 HTTP协议特点及其相关概念

HTTP协议的主要特点可概括如下：

- 1.支持客户/服务器模式。
- 2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。
- 3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。
- 4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
- 5.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。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。
- 1.连接(Connection)：一个传输层的实际环流，它是建立在两个相互通讯的应用程序之间。
- 2.消息(Message)：HTTP通讯的基本单位，包括一个结构化的八元组序列并通过连接传输。
- 3.请求(Request)：一个从客户端到服务器的请求信息包括应用于资源的方法、资源的标识符和协议的版本号
- 4.响应(Response)：一个从服务器返回的信息包括HTTP协议的版本号、请求的状态(例如“成功”或“没找到”)和文档的MIME类型。
- 5.资源(Resource)：由URI标识的网络数据对象或服务。
- 6.实体(Entity)：数据资源或来自服务资源的回映的一种特殊表示方法，它可能被包围在一个请求或响应信息中。一个实体包括实体头信息和实体的本身内容。
- 7.客户机(Client)：一个为发送请求目的而建立连接的应用程序。
- 8.用户代理(Useragent)：初始化一个请求的客户机。它们是浏览器、编辑器或其它用户工具。
- 9.服务器(Server)：一个接受连接并对请求返回信息的应用程序。
- 10.源服务器(Originserver)：是一个给定资源可以在其上驻留或被创建的服务器。
- 11.代理(Proxy)：一个中间程序，它可以充当一个服务器，也可以充当一个客户机，为其它客户机建立请求。请求是通过可能的翻译在内部或经过传递到其它的服务器中。一个代理在 发送请求信息之前，必须解释并且如果可能重写它。代理经常作为通过防火墙的客户机端的门户，代理还可以作为一个帮助应用来通过协议处理没有被用户代理完成 的请求。
- 12.网关(Gateway)：一个作为其它服务器中间媒介的服务器。与代理不同的是，网关接受请求就好像对被请求的资源来说它就是源服务器；发出请求的客户机并没有意识到它在同网关打交道。网关经常作为通过防火墙的服务器端的门户，网关还可以作为一个协议翻译器以便存取那些存储在非HTTP系统中的资源。
- 13.通道(Tunnel)：是作为两个连接中继的中介程序。一旦激活，通道便被认为不属于HTTP通讯，尽管通道可能是被一个HTTP请求初始化的。当被中继的连接两端关闭时，通道便消失。当一个门户(Portal)必须存在或中介(Intermediary)不能解释中继的通讯时通道被经常使用。
- 14.缓存(Cache)：反应信息的局域存储。

2 HTTP协议的运作方式

HTTP协议是基于请求 / 响应范式的。一个客户机与服务器建立连接后，发送一个请求给服务器，请求方式的格式为，统一资源标识符、协议版本号，后边是MIME信息包括请求修饰符、客户机信息和可能的内容。服务器接到请求后，给予相应的响应信息，其格式为一个状态行包括信息的协议版本号、一个成功或错误的代码，后边是MIME信息包括服务器信息、实体信息和可能的内容。在Internet上， HTTP通讯通常发生在TCP/IP连接之上。缺省端口是TCP80，但其它的端口也是可用的。但这并不预示着HTTP协议在Internet或其它网络的其它协议之上才能完成。HTTP只预示着一个可靠的传输。

3 HTTP协议的内部操作过程

首先，简单介绍基于HTTP协议的客户/服务器模式的信息交换过程，它分四个过程，建立连接、发送请求信息、发送响应信息、关闭连接。在WWW中，“客户”与“服务器”是一个相对的概念，只存在于一个特定的连接期间，即在某个连接中的客户在另一个连接中可能作为服务器。WWW服务器运行时，一直在TCP80端口(WWW的缺省端口)监听，等待连接的出现。

1.建立连接 连接的建立是通过申请套接字(Socket)实现的。客户打开一个套接字并把它约束在一个端口上，如果成功，就相当于建立了一个虚拟文件。以后就可以在该虚拟文件上写数据并通过网络向外传送。

2.发送请求打开一个连接后，客户机把请求消息送到服务器的停留端口上，完成提出请求动作。HTTP/1.0 请求消息的格式为：

请求消息=请求行(通用信息|请求头[实体头])CRLF[实体内容]
请求 行=方法 请求URL HTTP版本号 CRLF
方 法=GET|HEAD|POST|扩展方法
U R L=协议名称+宿主名+目录与文件名
请求行中的方法描述指定资源中应该执行的动作，常用的方法有GET、HEAD和POST。不同的请求对象对应GET的结果是不同的，对应关系如下：
对象 GET的结果

文件	文件的内容
程序	该程序的执行结果
数据库查询	查询结果

HEAD——要求服务器查找某对象的元信息，而不是对象本身。
POST——从客户机向服务器传送数据，在要求服务器和CGI做进一步处理时会用到**POST**方法。
POST主要用于发送HTML文本中**FORM**的内容，让CGI程序处理。
一个请求的例子为：**GET**http://networking.zju.edu.cn/zju/index.htm**HTTP/1.0**
头信息又称为元信息，即信息的信息，利用元信息可以实现有条件的请求或应答。
请求头——告诉服务器怎样解释本次请求，主要包括用户可以接受的数据类型、压缩方法和语言等。
实体头——实体信息类型、长度、压缩方法、最后一次修改时间、数据有效期等。
实体——请求或应答对象本身。

3.发送响应
服务器在处理完客户的请求之后，要向客户机发送响应消息。
HTTP/1.0的响应消息格式如下：
响应消息=状态行(通用信息头|响应头|实体头) CRLF （实体内容）
状态行=**HTTP**版本号 状态码 原因叙述
响应头的信息包括：服务程序名，通知客户请求的**URL**需要认证，请求的资源何时能使用。

4.关闭连接
客户和服务器双方都可以通过关闭套接字来结束TCP/IP对话
通常**HTTP**消息包括客户机向服务器的请求消息和服务器向客户机的响应消息。这两种类型的消息由一个起始行，一个或者多个头域，一个只是头域结束的空行和可选的消息体组成。**HTTP**的头域包括通用头，请求头，响应头和实体头四个部分。每个头域由一个域名，冒号（:）和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符，头域可以被扩展为多行，在每行开始处，使用至少一个空格或制表符。

第二部分 它的无状态性质的缺陷及其解决方法

既然**HTTP**协议的目的在于支持超文本的传输，更加广义一些就是支持资源的传输，那么在客户端浏览器向**HTTP**服务器发送请求，继而**HTTP**服务器将相应的资源发回给客户端这样一个过程中，无论对于客户端还是服务器，都没有必要记录这个过程，因为每一次请求和响应都是相对独立的，就好像你在自动售货机前投下硬币购买商品一样，谁都不会也不需要记住这样一个交易过程。一般而言，一个**URL**对应着唯一的超文本，而**HTTP**服务器也绝对公平公正，不管你是**Michael**，还是**Jordon**，它都会根据接收到的**URL**请求返回相同的超文本。正是因为这样的唯一性，使得记录用户的行为状态变得毫无意义，所以，**HTTP**协议被设计为无状态的连接协议符合它本身的需求。

然而，随着时间的推移，人们发现静态的**HTML**着实无聊而乏味，增加动态生成的内容才会令**Web**应用程序变得更加有用。于是乎，**HTML**的语法在不断膨胀，其中最重要的是增加了表单（**Form**）；客户端也增加了诸如脚本处理、**DOM**处理等功能；对于服务器，则相应的出现了**CGI**（**Common Gateway Interface**）以处理包含表单提交在内的动态请求。在这种客户端与服务器进行动态交互的**Web**应用程序出现之后，**HTTP**无状态的特性严重阻碍了这些应用程序的实现，毕竟交互是需要承前启后的，简单的购物车程序也要知道用户到底在之前选择了什么商品。于是，两种用于保持**HTTP**连接状态的技术就应运而生，一个是**Cookie**，而另一个则是**Session**。

Cookie是通过客户端保持状态的解决方案。从定义上来说，**Cookie**就是由服务器发给客户端的特殊信息，而这些信息以文本文件的方式存放在客户端，然后客户端每次向服务器发送请求的时候都会带上这些特殊的信息。让我们说得更具体一些：当用户使用浏览器访问一个支持**Cookie**的网站的时候，用户会提供包括用户名在内的个人信息并且提交至服务器；接着，服务器在向客户端回传相应的超文本的同时也会发回这些个人信息，当然这些信息并不是存放在**HTTP**响应体（**Response Body**）中的，而是存放于**HTTP**响应头（**Response Header**）；当客户端浏览器接收到来自服务器的响应之后，浏览器会将这些信息存放在一个统一的位置，对于**Windows**操作系统而言，我们可以从：[系统盘]\Documents and Settings\[用户名]\Cookies目录中找到存储的**Cookie**；自此，客户端再向服务器发送请求的时候，都会把相应的**Cookie**再次发回至服务器。而这次，**Cookie**信息则存放在**HTTP**请求头（**Request Header**）了。


有了**Cookie**这样的技术实现，服务器在接收到来自客户端浏览器的请求之后，就能够通过分析存放于请求头的**Cookie**得到客户端特有的信息，从而动态生成与该客户端相对应的内容。通常，我们可以从很多网站的登录界面中看到“请记住我”这样的选项，如果你勾选了它之后再登录，那么在下次访问该网站的时候就不需要进行重复而繁琐的登录动作了，而这个功能就是通过**Cookie**实现的。

与**Cookie**相对的一个解决方案是**Session**，它是通过服务器来保持状态的。由于**Session**这个词汇包含的语义很多，因此需要在这里明确一下**Session**的含义。首先，我们通常都会把**Session**翻译成会话，因此我们可以把客户端浏览器与服务器之间一系列交互的动作称为一个**Session**。从这个语义出发，我们会提到**Session**持续的时间，会提到在**Session**过程中进行了什么操作等等；其次，**Session**指的是服务器端为客户端所开辟的存储空间，在其中保存的信息就是用于保持状态。从这个语义出发，我们则会提到往**Session**中存放什么内容，如何根据键值从**Session**中获取匹配的内容等。

要使用**Session**，第一步当然是创建**Session**了。那么**Session**在何时创建呢？当然还是在服务器端程序运行的过程中创建的，不同语言实现的应用程序有不同创建**Session**的方法，而在Java中是通过调用HttpServletRequest的getSession方法（使用true作为参数）创建的。在创建了**Session**的同时，服务器会为该**Session**生成唯一的**Session id**，而这个**Session id**在随后的请求中会被用来重新获得已经创建的**Session**；在**Session**被创建之后，就可以调用**Session**相关的方法往**Session**中增加内容了，而这些内容只会保存在服务器中，发到客户端的只有**Session id**；当客户端再次发送请求的时候，会将这个**Session id**带上，服务器接受到请求之后就会依据**Session id**找到相应的**Session**，从而再次使用之。正式这样一个过程，用户的状态也就得以保持了。有关**Session**的内容还比较多，在以后的Post中，我还将继续讲述。

另外，还有一种**Application**对象，它和**Cookie**及**Session**的区别主要如下图所示：

它的功能可以实现对所有用户的操作，用下面这个例子可以很容易的理解：



```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="utf-8" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Application和Session的例子</title>

</head>

<body>

<%

int applicationCount=1;

int sessionCount=1;

//如果Application["ApplicationCount"]为空,即没有设置该名字的session

if(Application["ApplicationCount"]==null)

{

Application["ApplicationCount"]=1;

}

else //否则取出以前的ApplicationCount ,并在上面加1

{

applicationCount=(int)Application["ApplicationCount"]+1;

Application["ApplicationCount"]=applicationCount;

}

//如果session["sessionCount"]为空,即没有设置该名字的session

if(Session["SessionCount"]==null)

{

Session["SessionCount"]=1;

}

else

{

sessionCount=(int)Session["SessionCount"]+1;

Session["SessionCount"]=sessionCount;

}

Response.Write("当前页面Application记录访问到了"+applicationCount+"次<br/>");

Response.Write("当前页面Session记录访问到了"+sessionCount+"次<br/>");

%>

</body>

</html>

这是一个Dreamweaver中的.aspx的文件,运行可知结果。
```

分类: [PHP](#)[好文要顶](#)[关注我](#)[收藏该文](#)



王永东gg

关注 - 0

粉丝 - 0

+加关注

00

posted @ 2015-11-10 10:51 王永东gg 阅读(18) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 最新IT新闻：
- 转型旅游平台，Airbnb打情怀牌能否取胜？
 - 乐视汽车：浙江工厂年底前开工，天津工厂仍需商洽
 - Linux Kernel 4.9分支的第6个候选版本发布
 - 雷军：红米廉价不等于Low
 - "中国科研诚信问题严重程度史无前例"非危言耸听
- » 更多新闻...
- 最新知识库文章：
- 循序渐进地代码重构
 - 技术的正宗与野路子
 - 陈皓：什么是工程师文化？
 - 没那么难，谈CSS的设计模式
 - 程序猿媳妇儿注意事项
- » 更多知识库文章...