**HTTP协议—会话技术**

目录

[二、 HTTP协议介绍 7](#_Toc495855089)

[1. HTTP协议概念 7](#_Toc495855090)

[1) 协议 7](#_Toc495855091)

[2) HTTP协议 8](#_Toc495855092)

[2. 特点 8](#_Toc495855093)

[1) 通常是基于b/s（browser/server）结构的访问 8](#_Toc495855094)

[2) 无连接请求 8](#_Toc495855095)

[3) 无状态 9](#_Toc495855096)

[3. 分类 9](#_Toc495855097)

[1) HTTP请求协议（request） 9](#_Toc495855098)

[2) HTTP响应协议（respond，response） 9](#_Toc495855099)

[4. 协议项格式 9](#_Toc495855100)

[5. 抓包工具介绍 10](#_Toc495855101)

[1) 监视工具fiddler 10](#_Toc495855102)

[2) firfox,google中如何抓取HTTP协议内容 11](#_Toc495855103)

[三、 HTTP请求协议 12](#_Toc495855104)

[1. 组成部分 12](#_Toc495855105)

[2. 请求行 13](#_Toc495855106)

[1) 格式 13](#_Toc495855107)

[3. 请求头 14](#_Toc495855108)

[1) 描述 14](#_Toc495855109)

[2) 常见的请求头协议项 15](#_Toc495855110)

[3) $\_SERVER数组详解 15](#_Toc495855111)

[4. 请求体 16](#_Toc495855112)

[1) POST方式请求数据举例—文件上传 17](#_Toc495855113)

[2) 从HTTP协议角度，分析GET方式与POST方式提交数据的差异（面试题） 17](#_Toc495855114)

[四、 HTTP响应协议 17](#_Toc495855115)

[1. 组成 18](#_Toc495855116)

[2. 响应行及常见状态码 19](#_Toc495855117)

[1) 200 20](#_Toc495855118)

[2) 302 20](#_Toc495855119)

[3) 403 22](#_Toc495855120)

[4) 404 22](#_Toc495855121)

[5) 500 23](#_Toc495855122)

[6) 状态码总结： 24](#_Toc495855123)

[3. 响应头 24](#_Toc495855124)

[1) 常见的协议项 25](#_Toc495855125)

[4. 响应体 25](#_Toc495855126)

[5. header函数 26](#_Toc495855127)

[1) 返回内容类型 26](#_Toc495855128)

[2) 跳转 26](#_Toc495855129)

[3) 刷新 27](#_Toc495855130)

[五、 会话技术介绍 27](#_Toc495855131)

[1. 为什么需要会话技术 27](#_Toc495855132)

[2. 什么是会话技术 27](#_Toc495855133)

[3. 会话技术分类 28](#_Toc495855134)

[六、 COOKIE技术初步 28](#_Toc495855135)

[1. 概念 28](#_Toc495855136)

[2. 设置cookie 28](#_Toc495855137)

[1) 设置语法 28](#_Toc495855138)

[2) 如何在浏览器查看COOKIE 28](#_Toc495855139)

[3. 读取COOKIE数据 29](#_Toc495855140)

[4. 修改cookie数据 31](#_Toc495855141)

[5. 删除cookie 32](#_Toc495855142)

[6. 从HTTP协议理解COOKIE数据的传输 32](#_Toc495855143)

[1) 第一次响应 32](#_Toc495855144)

[2) 第二次请求 33](#_Toc495855145)

[3) 其他次响应 33](#_Toc495855146)

[七、 COOKIE数据的属性 33](#_Toc495855147)

[1. 有效期（生命周期） 34](#_Toc495855148)

[1) 有效期设置语法 34](#_Toc495855149)

[2) 常见的COOKIE生命周期 35](#_Toc495855150)

[2. cookie的有效路径（数据的作用范围：目录） 36](#_Toc495855151)

[3. COOKIE的有效域名 38](#_Toc495855152)

[1) 默认有效域名 38](#_Toc495855153)

[2) 自定义有效域名，实现跨子域 38](#_Toc495855154)

[4. COOKIE总结 40](#_Toc495855155)

[1) 完整语法 40](#_Toc495855156)

[2) 默认值 40](#_Toc495855157)

[八、 session技术初步 40](#_Toc495855158)

[1. 什么是session 40](#_Toc495855159)

[2. 原理 40](#_Toc495855160)

[3. 基本操作 41](#_Toc495855161)

[1) 开启session机制 41](#_Toc495855162)

[2) session数据的读写操作 42](#_Toc495855163)

[3) 删除一个session数据 42](#_Toc495855164)

[4) 删除所有session信息 43](#_Toc495855165)

[5) 销毁session信息 43](#_Toc495855166)

[九、 SESSION 数据高级 44](#_Toc495855167)

[1. 从HTTP协议，为什么SESSION是基于COOKIE的？ 44](#_Toc495855168)

[2. SESSION配置选项 46](#_Toc495855169)

[1) session.name 46](#_Toc495855170)

[2) session.cookie\_lifetime 46](#_Toc495855171)

[3) session.cookie\_path 47](#_Toc495855172)

[4) session.cookie\_domain 47](#_Toc495855173)

[3. session与COOKIE之间的区别与联系 48](#_Toc495855174)

[1) 区别 48](#_Toc495855175)

[2) 联系 48](#_Toc495855176)

[十、 会话技术应用--登录、退出模块 48](#_Toc495855177)

[1. 数据表设计 49](#_Toc495855178)

[2. 登录表单设计 50](#_Toc495855179)

[3. 账号信息验证 50](#_Toc495855180)

[1) 接收账号信息 50](#_Toc495855181)

[2) 账号信息验证 50](#_Toc495855182)

[3) 登录成功，设置成功标志 51](#_Toc495855183)

[4. 其他页面的登录检查 51](#_Toc495855184)

[1) 添加登录入口 51](#_Toc495855185)

[5. 退出模块 52](#_Toc495855186)

[十一、 自学参考网站： 53](#_Toc495855187)

**回顾：**

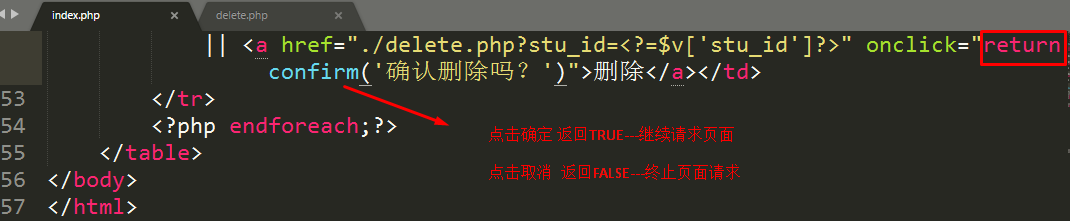
* 删除模块

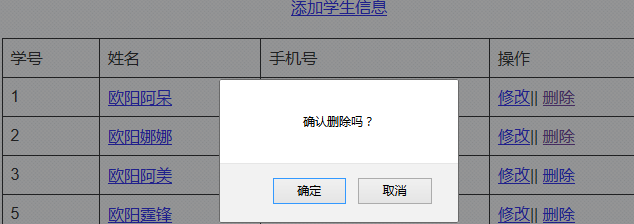
1. 在首页添加删除链接，传递学号参数：





1. 防止误操作

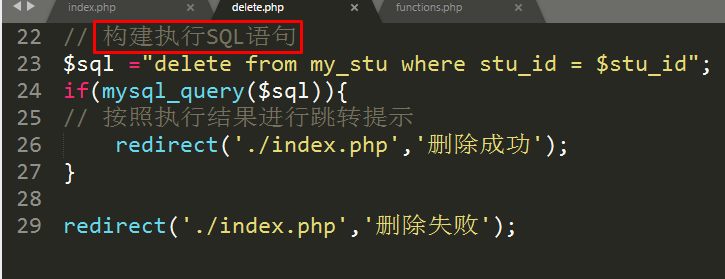




1. 判断接收学号



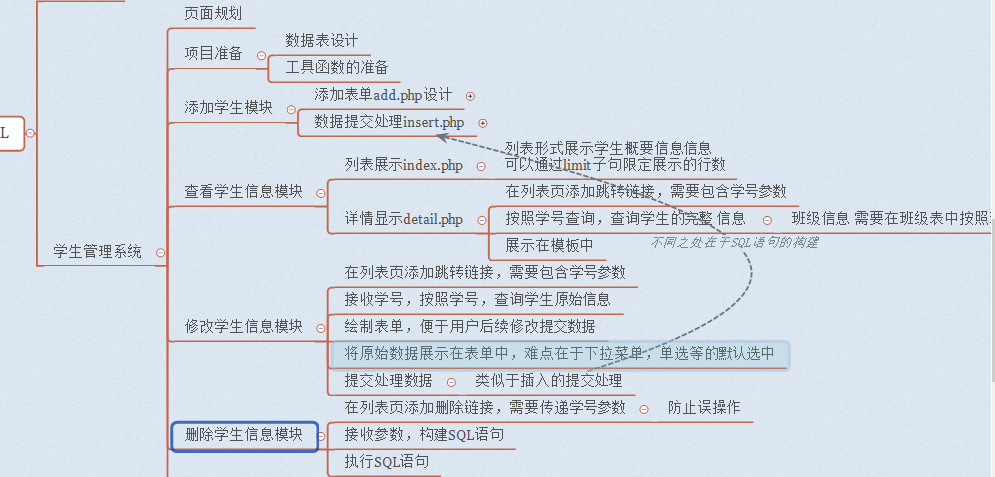
1. 按照学号进行删除



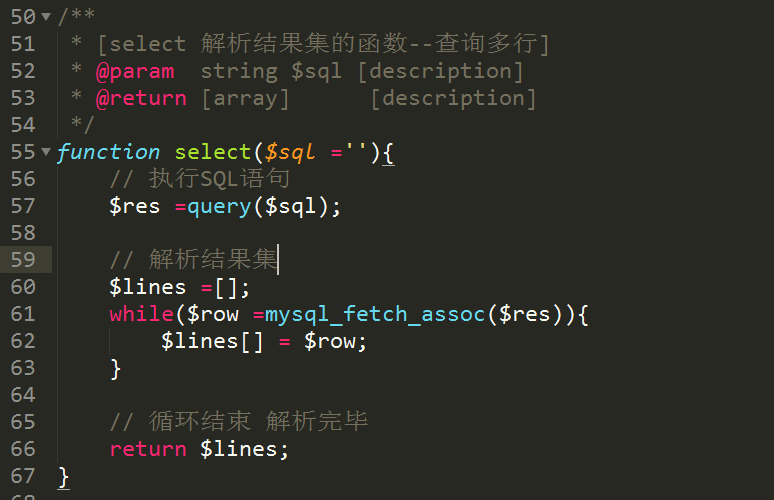
* 数据库操作步骤：



* 数据库操作函数封装介绍







**项目升级：**

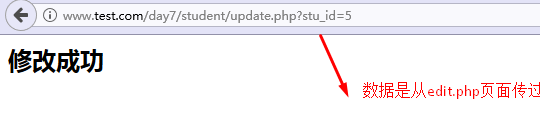


**思考：**

1. **什么是HTTP协议？**
2. **HTTP协议包括哪几种？每种协议的组成有哪些？**
3. **常见的状态码有哪些？**
4. **常见的响应头有哪些？**
5. **COOKIE数据的特点是什么？如何使用COOKIE数据？**
6. **SESSION数据的特点是什么？如何使用SESSION数据？**
7. **COOKIE数据与SESSION数据的联系是什么？各自的应用场景有哪些？**

# HTTP协议介绍

一个问题: 如何确保浏览器及客户端之间的数据交互能够顺利进行？如在学生信息添加模块中，在表单中输入的信息，成功的其他页面接收，如何实现，遵循什么规范？



--HTTP协议

## HTTP协议概念

### 协议

协议，就是事先的一种约定、规则、规范、标准。（入学协议）

有：HTTP（超文本传输协议），ftp（文件传输协议，将网站程序传输到Apache服务器），SMTP（Simple Mail Transfer Protocol，简单邮件传输协议）、pop3（邮局协议版本3,收发邮件的协议），telnet（远程登录协议）。

### HTTP协议

HTTP协议：HyperText Transfer Protocol 超文本传输协议。Web服务器上存储有HTML文件及图像资源等，在请求资源时，客户端（浏览器端）与WEB服务器端之间的交互协议。

HTTP协议工作在TCP/IP协议之上（应用层），所有的web页面文件必须遵循的标准.

HTTP协议保证计算机可以正确快速的传输文本文档数据，以及指定数据的正确显示格式等。

## 特点

### 通常是基于b/s（browser/server）结构的访问

支持基于B/S，C/S模式。用户通过客户端（浏览器）发起请求，请求WEB服务器的指定端口（默认端口为80）。

### 无连接请求

**HTTP/1.0**

每次TPC/IP连接（拨通电话）只能处理一个请求（资源，web对象），浏览器向服务器发起一次请求后，只能得到服务器的一次响应。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。（短连接）采用这种方式可以节省传输时间。

给家里打电话要点生活费，邮寄点家乡小吃：

生活费

家乡小吃

**HTTP/1.1**

默认为长连接，持久连接。一次连接可以请求多个资源。

一个连接可被多个请求重复利用的保持连接机制被引入。这种连接持续化显著地减少了请求延迟，因为客户不用在首次请求后再次进行TCP交互确认创建连接。

无连接准确来说是HTTP1.0的标准特点。

生活费

家乡小吃

### 无状态

也叫作无记忆功能，多次请求之间没有对应关系。即同一个浏览器向同一个服务器发起多次请求的时候，服务器不能识别该浏览器。

如登录：已经登录，请求其他页面，还需要登录。

## 分类

### HTTP请求协议（request）

客户端(浏览器)向服务器索要数据时遵循的协议（按照什么格式发送数据，或者必须发送的内容）

### HTTP响应协议（respond，response）

服务器向浏览器返回数据时的协议

## 协议项格式

* 请求及响应客户端成。形式：

协议名：协议内容

* 每个协议独占一行。

location: ./index.php

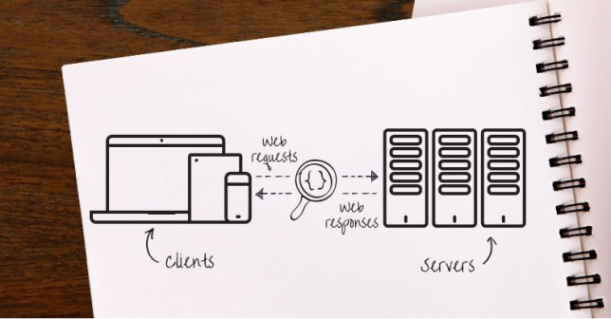
## 抓包工具介绍

### 监视工具fiddler

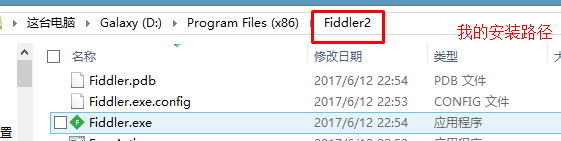
实现基于任何浏览器、系统、平台的网络调试代理工具，在浏览器与服务器之间对数据的传输进行监视。

下载地址：<http://www.telerik.com/fiddler>

**工作原理如下：**



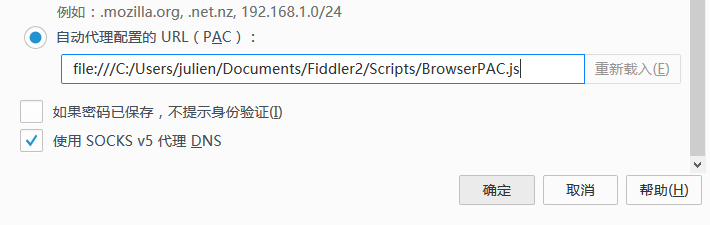
在浏览器与服务器之间建立用户代理，监视各个请求以及响应。



**火狐浏览器配置步骤：**

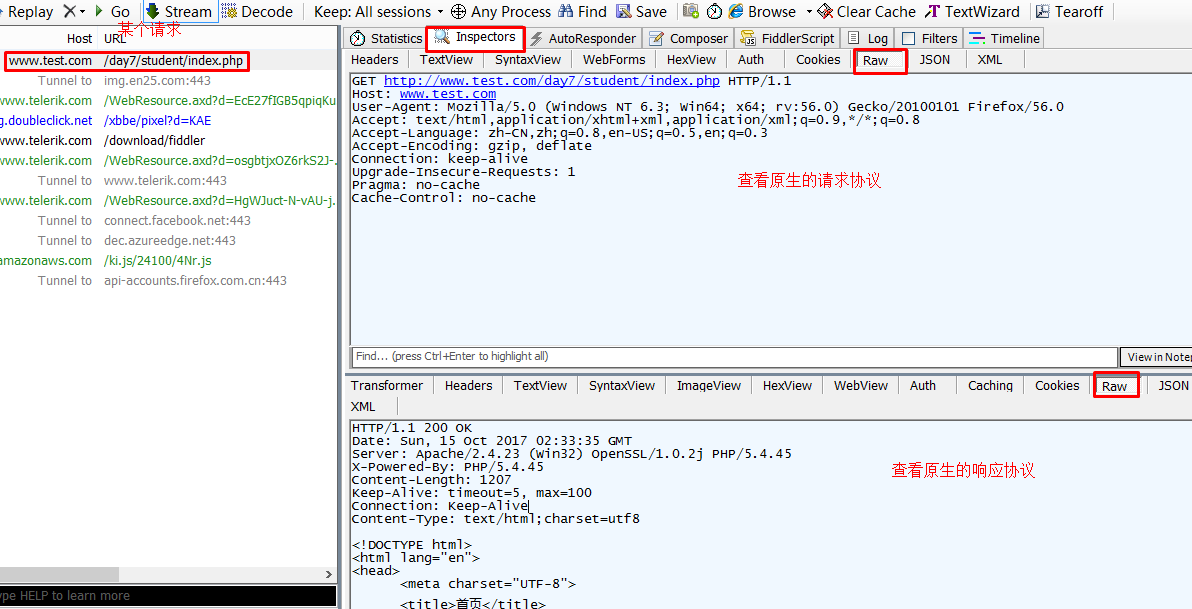
1. 打开Fiddler -> 菜单栏 Tools ->Telerik Fiddler Options ->Connections  ->点击蓝色字体的“Copy Browser Proxy Configuration URL” （即拷贝代理配置文件路径）

2. 打开FireFox ->菜单栏Tools ->Advanced ->选择Network ->点击Connection后的Settings按钮 ->弹出Connection Settings对话框 ->选择最后一个“Automatic proxy configuration URL”，并将拷贝的文件路径粘贴到空白处 ->点击OK，退出。



**Fiddler使用方法：**

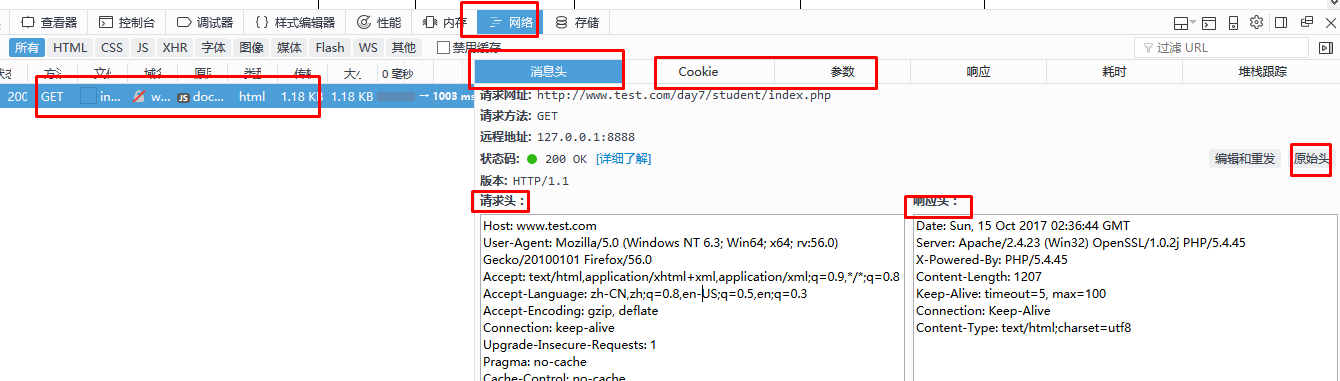
在运行界面, 点击某个请求 🡪 inspector(查看器) 🡪 raw(查看原始的HTTP协议内容)



### firfox,google中如何抓取HTTP协议内容

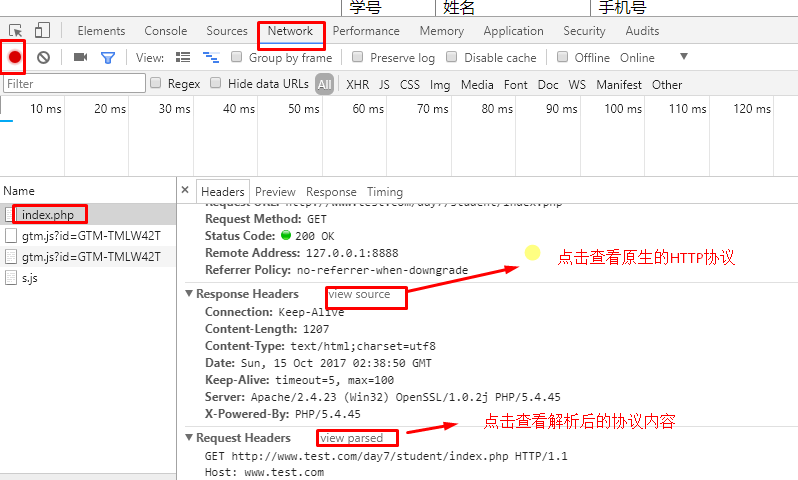
**火狐：**

CTRL + SHIFT + S 打开控制台 🡪网络 🡪 点击某个请求 🡪 查看消息头以及参数（POST上传）



**谷歌：**

F12🡪network🡪打开监视开关🡪点击某个请求

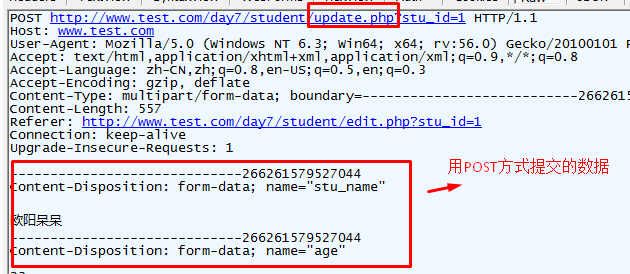


# HTTP请求协议

## 组成部分

用于规范客户端（浏览器）向服务器发送数据的格式：

* **请求行（request\_line）**：如请求方式，请求的页面，遵循的协议
* **请求头**（**request\_head**）：如通知服务器，浏览器支持的数据类型
* **空白行**：标识请求头结束，请求体开始。对两者进行分隔。
* **请求数据**（request\_content）：提交的核心数据：POST数据



## 请求行

独占一行，用来说明当前请求的最基本的信息

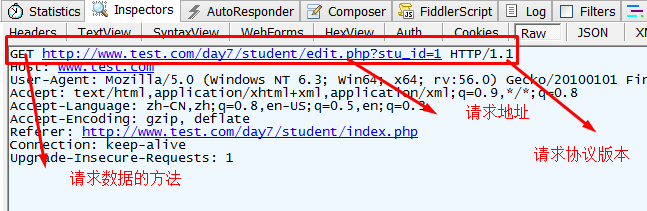
### 格式

请求方式GET|POST url HTTP/1.1

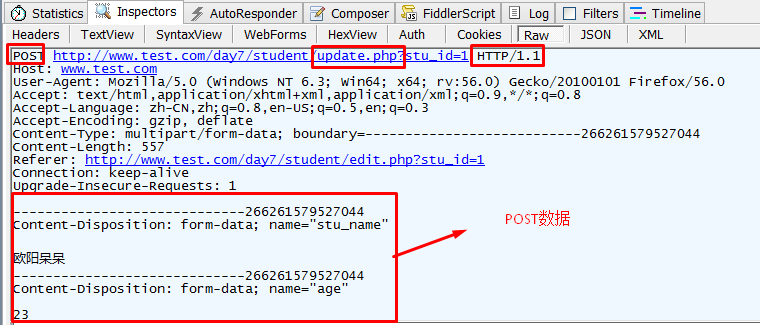
**不同的数据提交方式：**

提交方式包括GET与POST，不指定为POST，默认以GET方式提交数据。

**GET方式：**



**POST方式：**



**HTTP1.0与HTTP1.1**

* HTTP/1.0 短连接 在一次TCP/IP连接中，请求一次。
* HTTP/1.1 长连接。

如：Connection: keep-alive。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。

HTTP/1.1支持持久连接，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟。

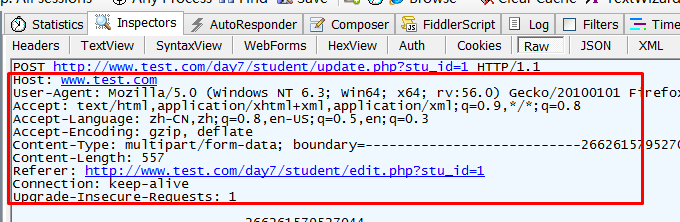


## 请求头

### 描述

请求中，所有当前要用到的协议项的集合，即浏览器在发送真正请求数据之前，事先告诉服务器的一些信息（一般是浏览器自身的信息），并且每个协议项都要独占一行。

### 常见的请求头协议项



* + **host**：当前url中所要请求的服务器的主机名（域名）。
  + accept：表示浏览器可以接收的数据类型(MIME类型)

如text/html,text/css,text/javascript,image/png,image/jpeg….

* + accept-language：可以接收的语言类型，zh-cn，en，fr等，
  + user-agent：用户代理UA，当前发起请求的浏览器的内核信息，作为识别客户端设备的重要信息。比如：

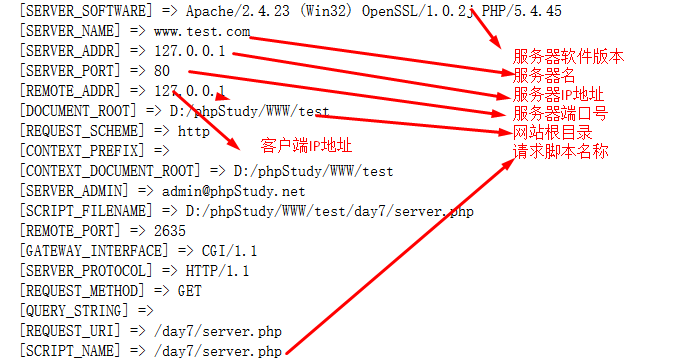
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0

winNT6.1—win7，winNT 6.3—win8，winNT 10.0—win10

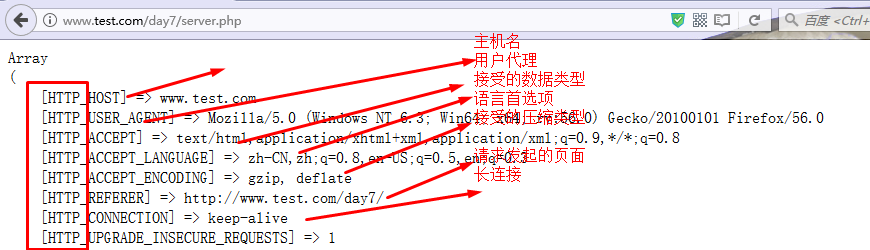
* + referer：此次请求来自的网址：可以判断请求是否是来自本网站,防止盗链（图片链接其他资源）。

### $\_SERVER数组详解

$\_SERVER为超全局、预定义变量，保存服务器配置信息、HTTP请求协议信息。



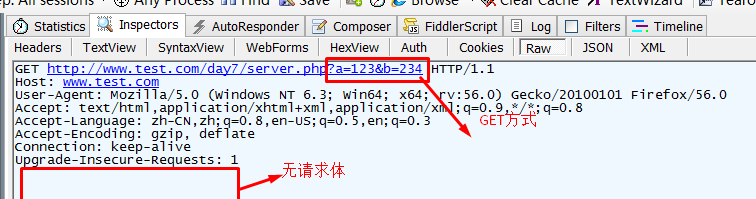
请求协议的信息（请求头）：



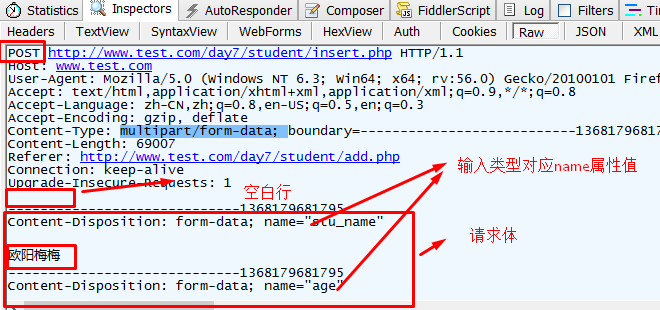
## 请求体

请求数据，主要是通过表单提交的数据。

消息体（entity-body），请求体。只有POST方式提交的数据才会在请求数据中出现。一般使用application/x-www-form-urlencoded方式对表单数据进行编码，文件上传时必须使用multipart/form-data方式。



### POST方式请求数据举例—文件上传

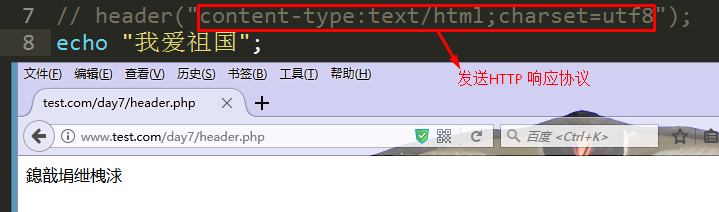




### 从HTTP协议角度，分析GET方式与POST方式提交数据的差异（面试题）

|  |  |  |
| --- | --- | --- |
| 请求方式 | GET | POST |
| 请求数据查看位置 | 地址栏（请求行） | 请求主体 |
| 安全性 | 低 | 高 |
| 数据上限 | 1kb | 8M |
| 是否在历史记录中看到参数 | Y | N |

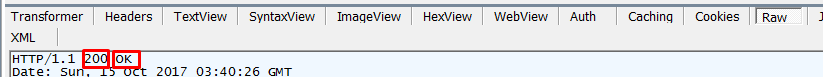
# HTTP响应协议



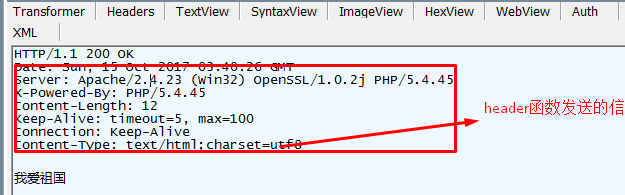
**解决乱码的方法**：通过header函数，响应给浏览器如何展示数据。

## 组成

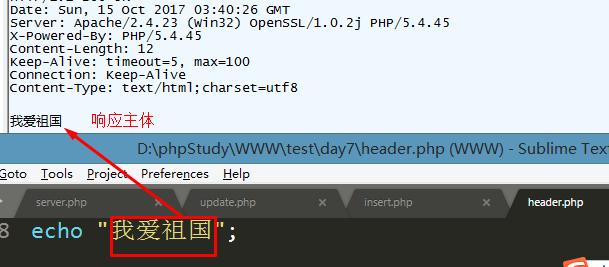
* **状态行**（响应行）：请求资源成功、失败、或者其他的状态。



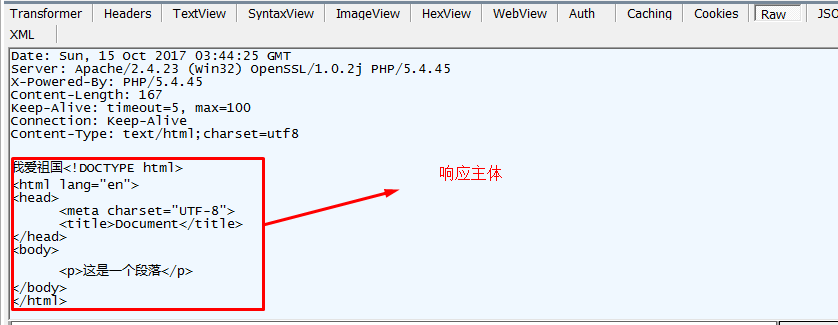
* **响应头**：通知浏览器一些信息，如返回数据的类型、使用什么字符集显示数据等



* **空白行**：分隔响应头及响应主体。
* **响应主体**（数据）:请求的HTML，PHP输出语句返回的的数据。





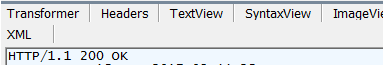


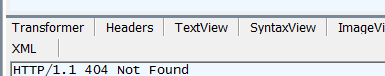
## 响应行及常见状态码

格式：

协议版本 状态码 状态描述

通知浏览器，请求资源的返回结果状态。



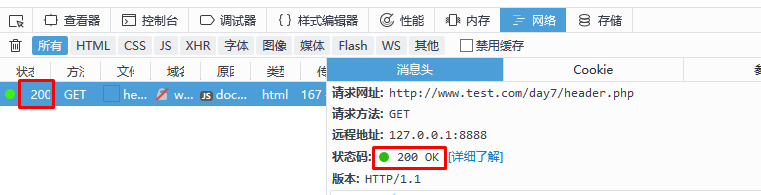


* 1XX：请求尚未成功。
* 2XX：请求正常
* 3XX：重定向
* 4XX：请求失败
* 5XX：服务器错误

### 200

200 ok

请求成功。



### 302

302 redirect |Found 重定向

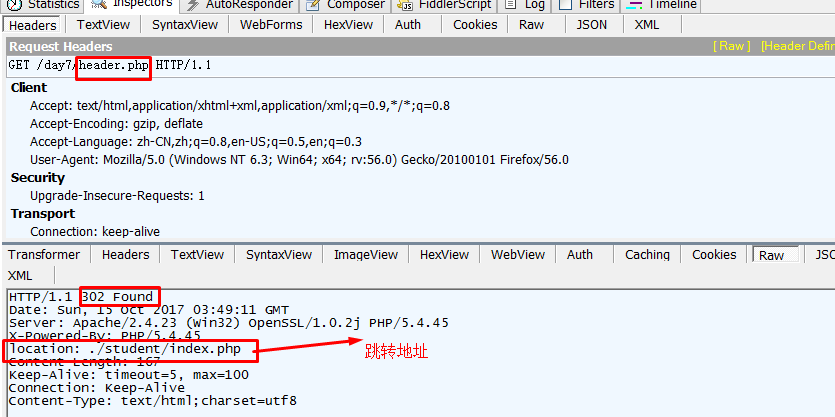
重定向至其他页面：在响应头中通知浏览器如何跳转、刷新。

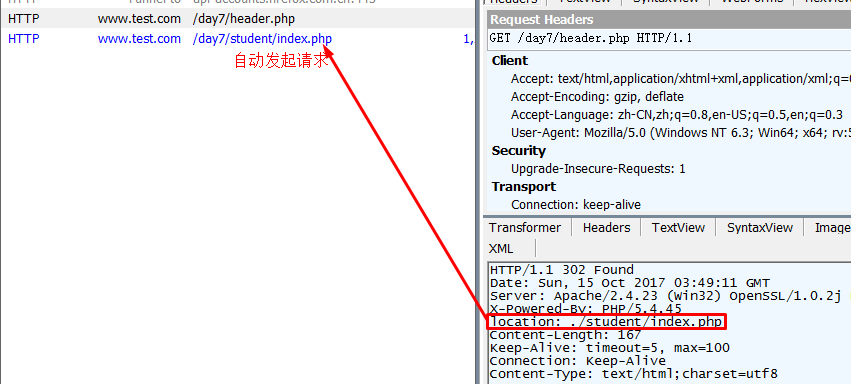
header(“location:http://www.itcast.cn”);

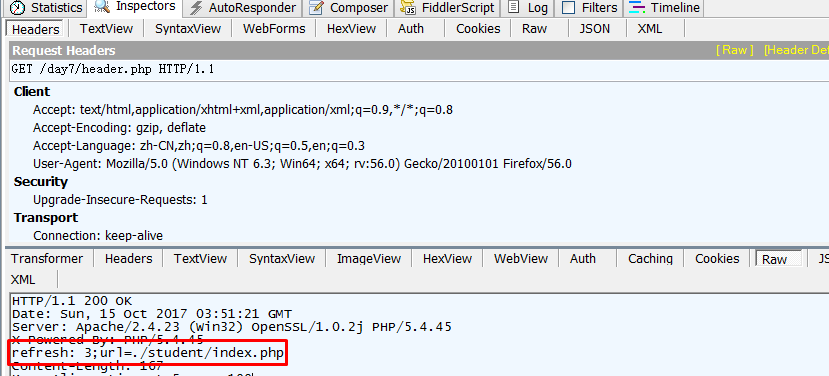
//发送响应头信息location:http://www.itcast.cn

header(“refresh:3;url=http://www.itcast.cn”);

//发送响应头信息refresh:3;url=http://www.itcast.cn



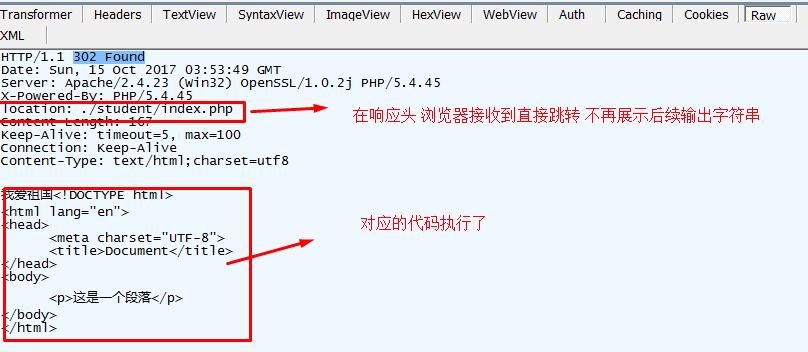




**一个问题**：

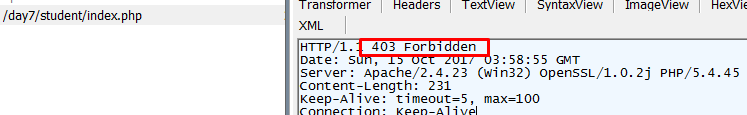
使用location协议项时，打开请求的页面A直接跳转至新的页面B，那么A页面中location后续的代码是否执行呢？

执行，但是输出语句的内容无法展示。



### 403

403 forbidden 禁止访问



如何模拟：在虚拟主机配置文件的Directory配置段中，添加Deny from all，此时无访问权限。

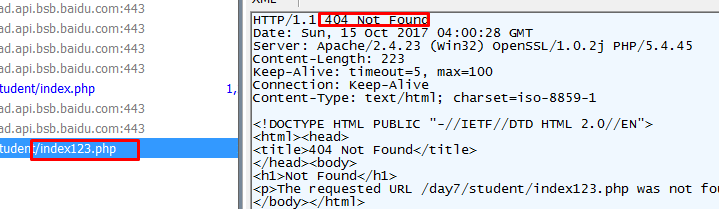


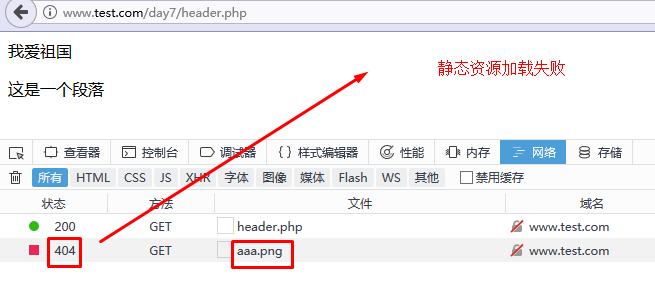
### 404

404 Not Found未找到页面

404页面可以设置为访问其他页面，或者错误页面。

设计前端页面时，注意检查资源是否加载成功，加载失败时返回404，需要检查路径的正确性。





可以在**未找打页面**时提示其他信息：

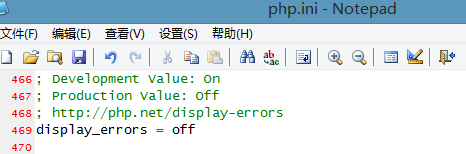


### 500

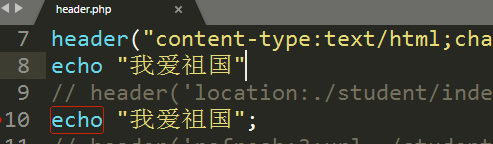
500 internal server error 服务器内部错误

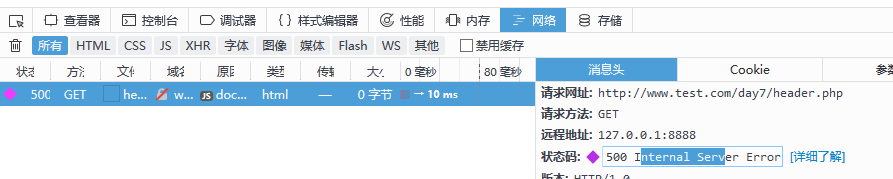
**模拟方法：**

* 在php.ini中设置：display\_errors = off



* 写一行错误代码：





### 状态码总结：

|  |  |  |
| --- | --- | --- |
| 状态码 | 状态描述 | 意义 |
| 200 | ok | 请求成功 |
| 302 | found | 重定向 |
| 403 | forbidden | 禁止访问 |
| 404 | not found | 页面不存在 |
| 500 | internal server error | 服务器内部错误 |

## 响应头

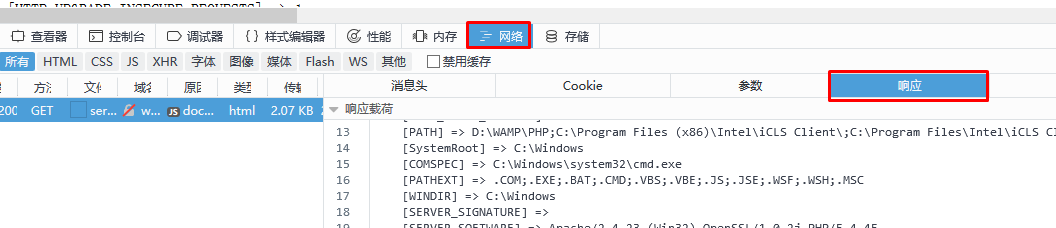
格式：

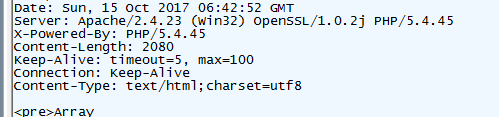
协议项 ：协议值

如location:./index.php

content-type:text/html;charset=utf8

### 常见的协议项





* + Date:被请求页面的最后更新时间（格林尼治时间，非本地时间）
  + Server:服务器版本信息
  + Content-Length:内容长度，单位为字节。
  + Content-Type: 通知浏览器返回数据的内容类型，浏览器进而选择正确的解析方式进行解析。

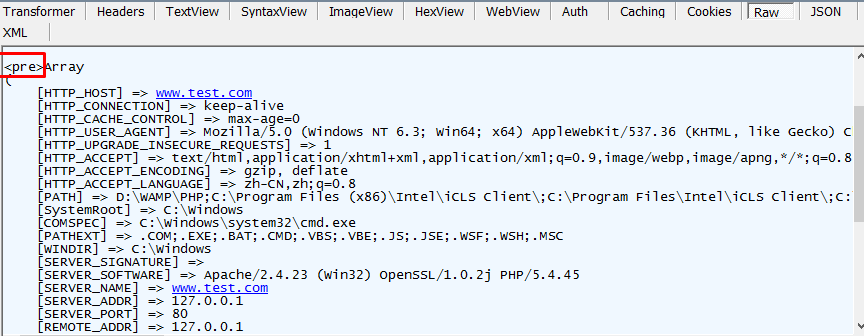
content-type:text/html；charset=utf8 告诉浏览器返回数据的类型是HTML格式，需要浏览器调用HTML解析器解析。使用UTF8字符集展示数据。

* + - 如果内容类型是浏览器不能解析的，浏览器会直接提示下载。



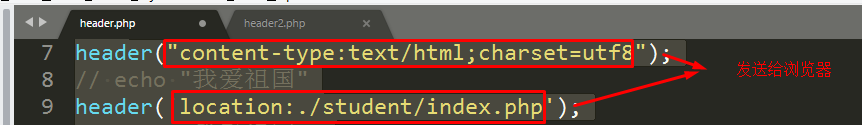
## 响应体

响应数据，为服务器返回的具体数据。即静态代码或者服务器脚本的输出（echo,print,print\_r,var\_dump）数据。



## header函数

发送原生的HTTP响应头，通知浏览器的一些动作信息。如跳转至指定页面，或者刷新或者使用什么类型解析返回的数据。



### 返回内容类型

header(“content-type:text/html;charset=utf8”);

将内容类型content-type:text/html;charset=utf8响应头发给浏览器，告知浏览器返回数据的类型以及展示数据使用的正确字符集。

图片类型的返回:

header(“content-type:image/png”);//返回的时图像的二进制信息，解析时必须按照图像格式解析，而不是HTML

header(“content-type:image/jpeg”);

header(“content-type:image/gif”);

### 跳转

通知浏览器，从一个页面跳转其他页面。

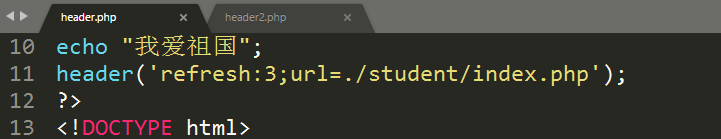
header(“location:请求的页面”);

在响应头中设置跳转的页面，浏览器接收到响应头立即跳至指定页面，响应体数据无法显示。

### 刷新

返回refresh响应头的内容给浏览器。浏览器等待一定秒数之后，自动跳转到其他页面。在等待期间，显示返回的数据。

header(“refresh:3;url=./请求地址”);



可以用来提示成功、失败、错误信息。

# 会话技术介绍

## 为什么需要会话技术

HTTP协议是无状态、无记忆的。多次请求之间，无任何的联系，即同一用户请求同一网站的不同页面，服务器无法识别是否是同一用户发起的请求。

因此，无法把请求的状态保存下去，用户无法进行连续的业务逻辑。

如：登录，在A页面登录，请求B页面，提示登录。

## 什么是会话技术

会话：浏览器与服务器之间的数据交流。

在同一台浏览器对服务器的多次请求中，将数据持久化存储的技术，以实现连续的业务逻辑

## 会话技术分类

按照数据持久化的位置进行区分，

数据持久化存储在服务器，为session技术。

数据持久化存储在浏览器，为cookie技术。

# COOKIE技术初步

## 概念

学生进每次教学楼，如何证明是传智播客学生： 每次开证明 or 带着胸牌（携带自身身份信息）

允许服务器脚本（PHP脚本）在浏览器端存储数据，并以此跟踪及识别用户的技术。

数据设置后，浏览器再次请求服务器指定页面时，将相关的数据(身份信息)发送给服务器脚本，供脚本使用。

## 设置cookie

cookie本质为简单数据(string 类型)，基本操作逻辑为增删改查。

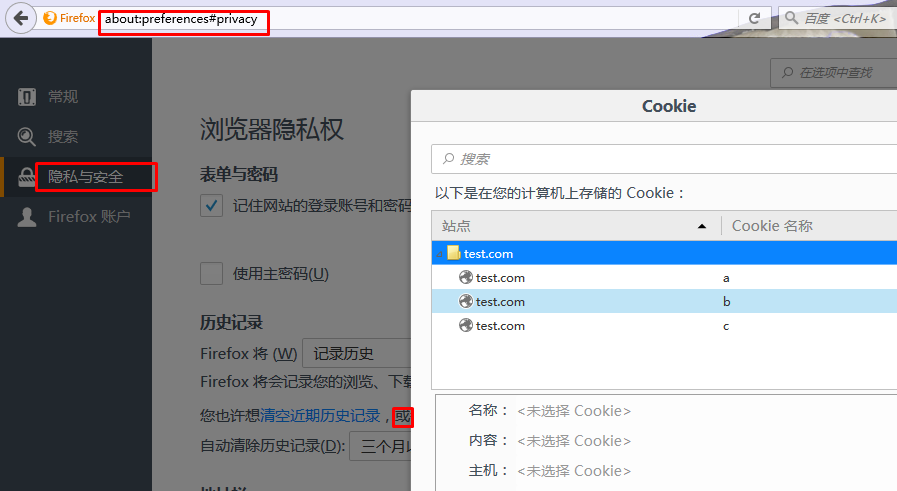
### 设置语法

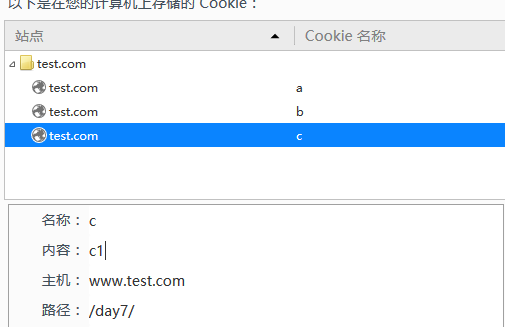
setcookie(数据名，数据)；

执行以上代码时，PHP脚本通过HTTP协议，通知浏览器在其COOKIE数据区存储数据。数据的类型为字符串，无法存储其他类型的数据。

### 如何在浏览器查看COOKIE

**火狐浏览器：**选项---隐私---历史记录---移除单个COOKIE。



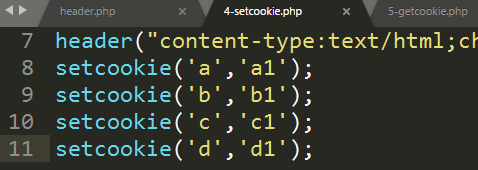


**谷歌浏览器：**设置—隐私设置—内容设置—cookie—所有cookie和网站数据

## 读取COOKIE数据

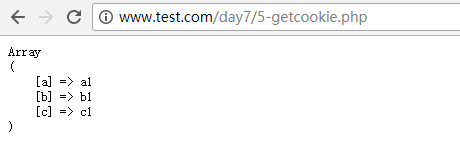
* 设置好的COOKIE数据，下次请求同一网站的页面时，浏览器将对应数据传输给PHP脚本（COOKIE数据对应不同的用户隐私数据，如账户编号，浏览文章编号等）。PHP脚本将数据存储在预定义变量$\_COOKIE中。可以通过$\_COOKIE进行读取，实现其他业务逻辑（如推荐信息等）。

**设置COOKIE：**



**读取COOKIE：**

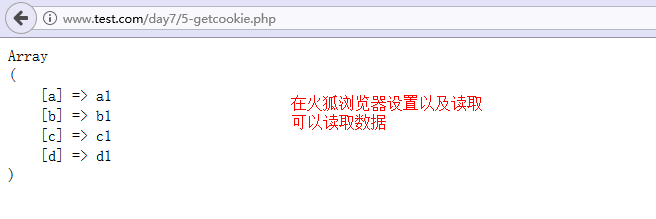




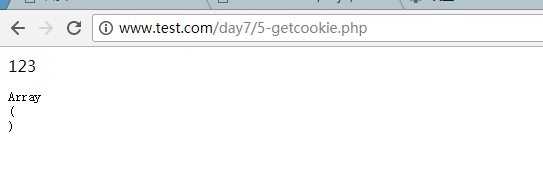
* 同一网站的不同页面之间可以进行COOKIE数据的共享：对于同一个浏览器，在网站的A页面设置的COOKIE数据，可以在网站的B脚本中读取到。使用同一台浏览器，请求相同网站时，浏览器携带跟该网站对应的全部COOKIE数据，每个页面都可以使用，实现数据的共享。



* COOKIE数据是基于浏览器的：在A浏览器请求页面设置的COOKIE数据，存储在A浏览器的COOKIE数据区，在B浏览器请求同样的页面，无法访问，因为在B浏览器的COOKIE数据区无对应的数据。



**在谷歌浏览器读取：无法读取到火狐浏览器设置的COOKIE数据**



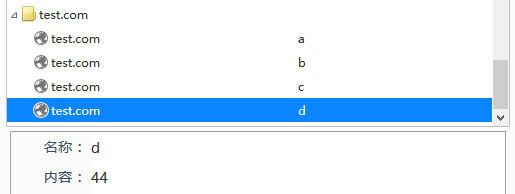
## 修改cookie数据

setcookie语法，将已经存在的COOKIE变量设置为其他值。

setcookie(“变量名”，“新值”);

执行过程，相当于修改变量值。

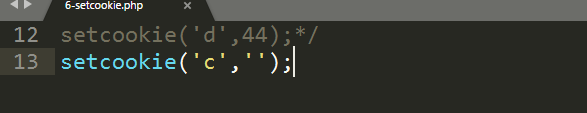


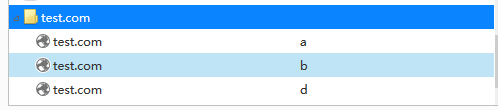


## 删除cookie

没有显式的删除函数。可以通过设置COOKIE数据为空字符串来实现。COOKIE数据为空后，浏览器自动消除该数据空间（减少空间占用，减少带宽占用）。

setcookie(“变量名”，“”)；



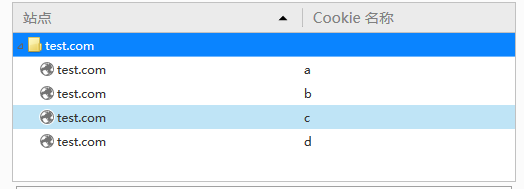


## 从HTTP协议理解COOKIE数据的传输

### 第一次响应

第一次响应，PHP脚本在响应头中，通过setcookie协议项，告诉浏览器如何设置数据。

设置好的COOKIE数据：

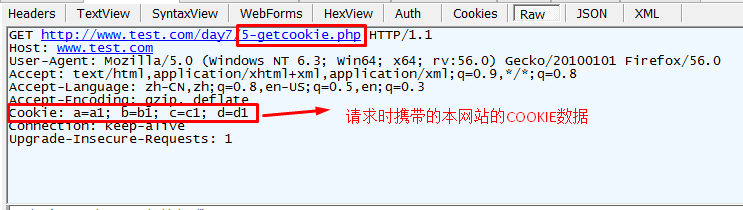




### 第二次请求

浏览器请求该网站的页面时，携带该网站对应的全部COOKIE数据。服务器脚本可以获取用户的信息 。

服务器脚本将数据自动存在$\_COOKIE预定义变量中。（下标就是COOKIE数据名）





### 其他次响应

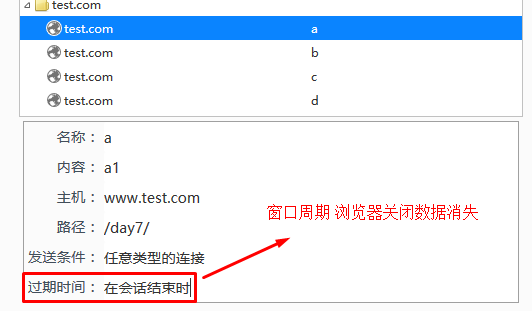
如果修改COOKIE数据，在响应头中会有setcookie协议项。

# COOKIE数据的属性



## 有效期（生命周期）

COOKIE数据的默认的生命周期是会话周期，即在浏览器窗口关闭之前，一直存在。 关闭浏览器：相当于结束默认的会话，COOKIE数据自动消失。再次打开浏览器，相同页面读取数据，无法读到。



好处：可以清除过期的COOKIE数据，减少空间占用，减少HTTP传输的数据量

### 有效期设置语法

设置COOKIE的第三个参数，为设置其有效期。

setcookie(“名称”,”值”，“有效期”);

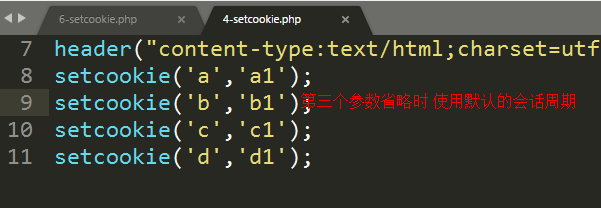
即COOKIE数据的有效时间，使用时间戳（秒数）来表示，即多少秒后对应的COOKIE数据消失。

COOKIE数据过期后，会被浏览器自动清除掉。HTTP请求中，仅携带在有效期内的COOKIE数据，可以减少携带的数据量。

### 常见的COOKIE生命周期

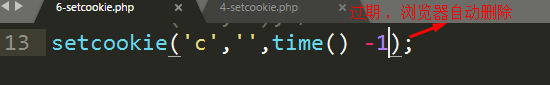
* 默认会话周期：窗口周期

窗口关闭，会话结束，COOKIE数据自动被清除。



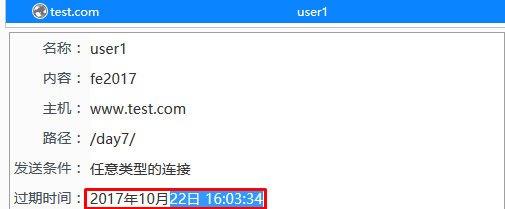
* 删除单个COOKIE数据的另外一种方法：

setcookie(名，’’,time() - 1);



* 一周有效期：

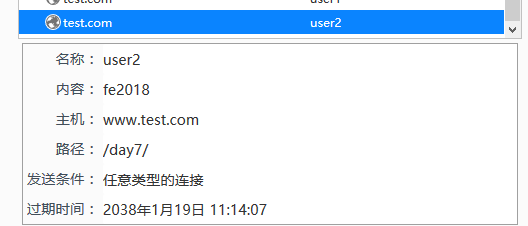




* 永久有效：

使用常量PHP\_INT\_MAX作为时间戳使用。

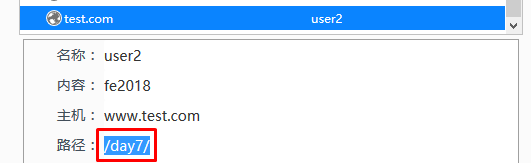




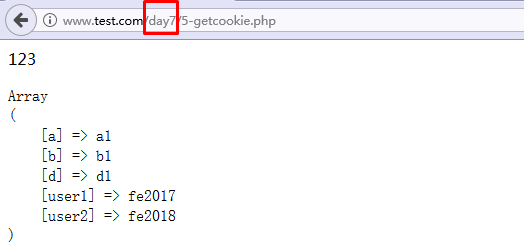
## cookie的有效路径（数据的作用范围：目录）

* 默认在当前目录（脚本所在的目录）及子目录，脚本可以直接访问对应COOKIE信息。父目录无法访问子目录的COOKIE信息。

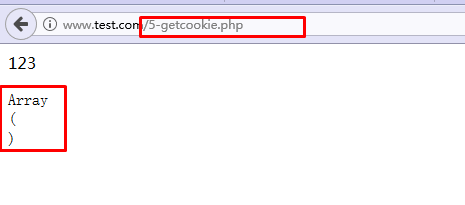
**设置有效路径：**



**在day7下读取：**



**在day7上一级目录读取：无法读到**

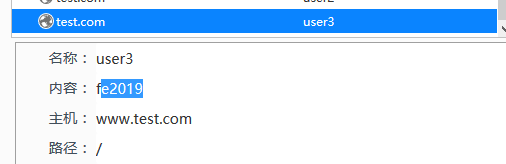


* 通常设置为/，表示网站根目录、整站有效

setcookie(名称，值，有效期，有效路径);

设置整站有效：





**其他目录中读取：**



## COOKIE的有效域名

域名：itcast.cn

主机(子域名或者二级域名)：[www.itcast.cn](http://www.itcast.cn) oa.itcast.cn tb.itcast.cn mail.itcast.cn

### 默认有效域名

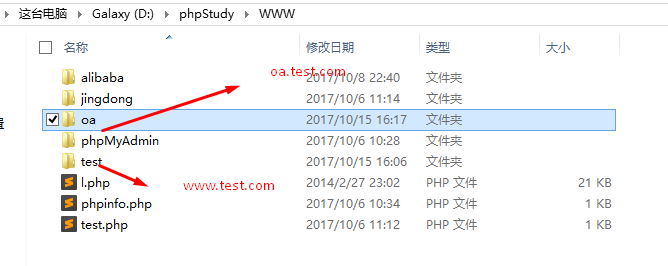
COOKIE数据默认都是当前域名有效，确保A网站设置的COOKIE数据，不会发送到B网站。

### 自定义有效域名，实现跨子域

setcookie(名，值，有效期，有效路径，’.website.com’);

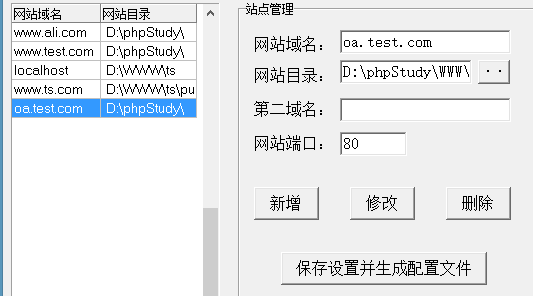
以[www.test.com](http://www.jd.com) 与oa. test.com 为例。

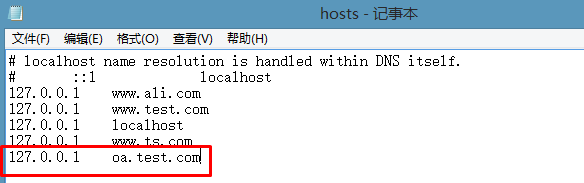
1. 创建两个文件夹，每个文件夹对应不同虚拟主机的网站根目录



1. 配置虚拟主机：都是test.com的子域名

将两个目录分别解析为不同子域名。

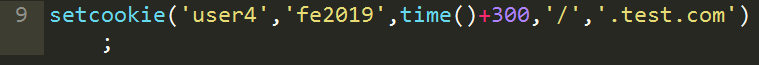




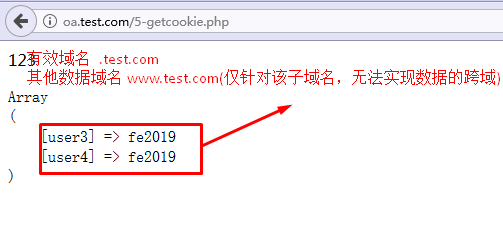
1. 设置参数：设置为一级域名有效，‘.域名’

在[www.test.com设置COOKIE](http://www.test.com设置COOKIE) 在oa.test.com 读取：

设置时，需要注意有效域名必须设置为 .website.com 格式。







## COOKIE总结

### 完整语法

setcookie(“名称”,”值”，“有效期”，“有效路径”，“有效域名”)；

### 默认值

**有效期**：默认会话周期。浏览器不关闭，数据一直存在。浏览器关闭数据消失。

time() -1： 数据已经过期，清除COOKIE数据。

**有效路径**：当前目录及子目录有效。设置整站有效:/（根目录）

**有效域名**：当前域名（设置COOKIE数据对应的子域名才可以读取，其他子域名默认无法读取）。实现跨子域：.itcast.cn.

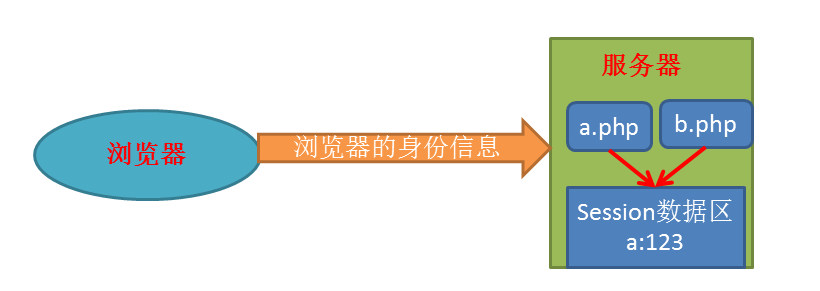
# session技术初步

思考：如何实现网站登录功能？验证完账号信息后，登录标志保存在哪里？

## 什么是session

为了实现业务逻辑的连续性，将部分信息持久化存储在服务器端，可以保证数据的安全性及各个脚本之间的共享。

## 原理



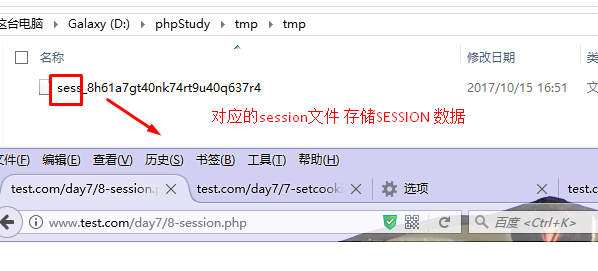
* 将客户端的状态信息（区分不同用户）存储在服务器端的SESSION数据区。
* 可以在网站的不同页面之间传递数据（页面数据共享）
* 可以识别不同的浏览器（基于浏览器）。

## 基本操作

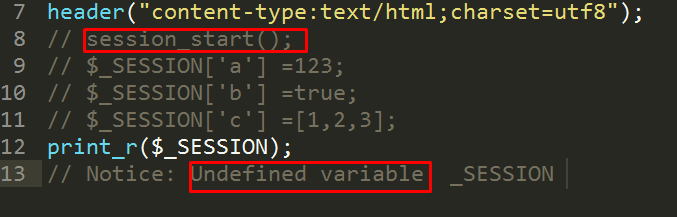
### 开启session机制

session\_start();

作用：开启session会话或者重用已经创建的会话。



没有开启SESSION会话机制时，$\_SESSION数组不会被定义出来，提示使用未定义变量“Notice: Undefined variable: \_SESSION in …”

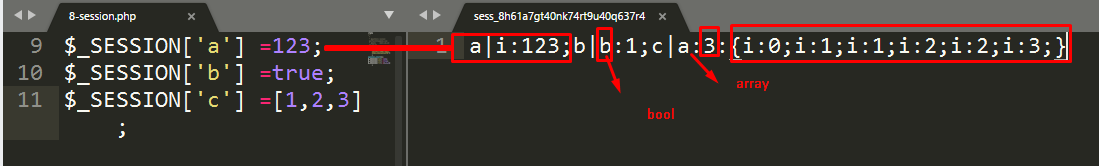


使用PHPstudy时，默认的SESSION文件路径为：/phpStudy/tmp/tmp

session文件的保存路径可以在php.ini中session.save\_path：修改

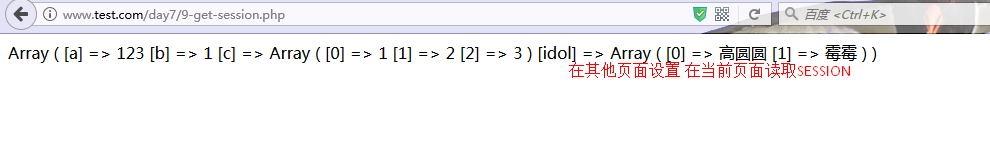
### session数据的读写操作

* SESSION数据的增删改查操作都是通过数组$\_SESSION进行的。元素的下标不能使用数字。可以存放多个数据（多个键值对）。



* 存储的数据类型几乎可以为PHP的全部数据类型。
* 数据存储在session文件中，实现同一网站多个页面共享session数据。

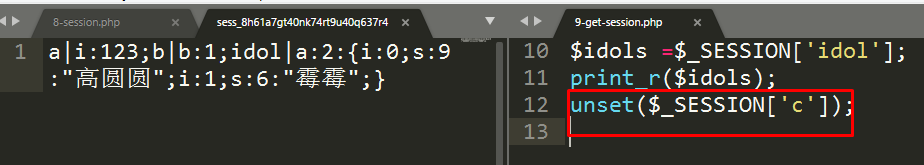




### 删除一个session数据

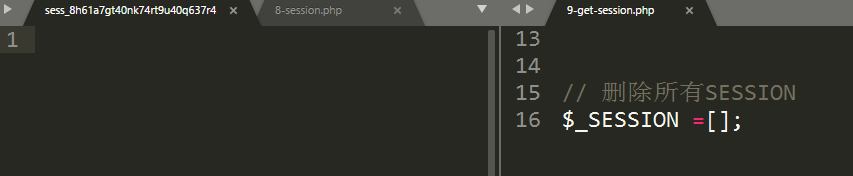
unset($\_SESSION[名称]);

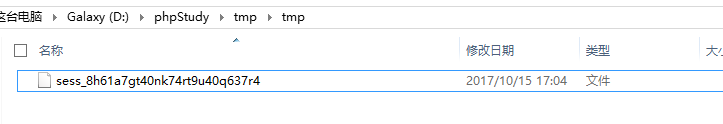
可以使用unset语法删除一个session数据（数组的一个元素）。删除的是session文件中的指定数据，不删除session文件。



### 删除所有session信息

$\_SESSION设置为空数组。本质为使用空数组覆盖session文件中对应的数据。session文件不被删除。





### 销毁session信息

session\_destroy();

直接删除session会话文件，PHP脚本将无法读取session数据。

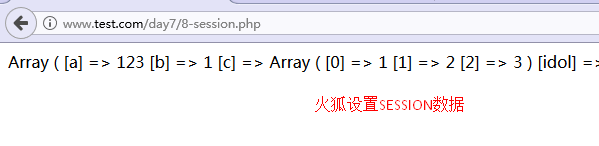


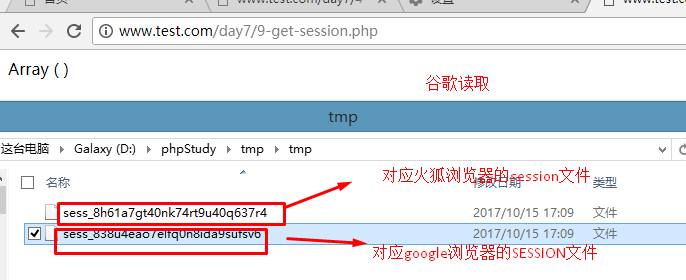
# SESSION 数据高级

**一个问题：**

在通过A浏览器请求网站的脚本，设置对应的SESSION数据，在A浏览器请求时，可以读取；通过B浏览器请求同样网站时，无法获取，为什么？

SESSION数据是基于COOKIE的。当浏览器不同时，COOKIE数据不同，对应的SESSION文件也不同，无法读取到SESSION数据。





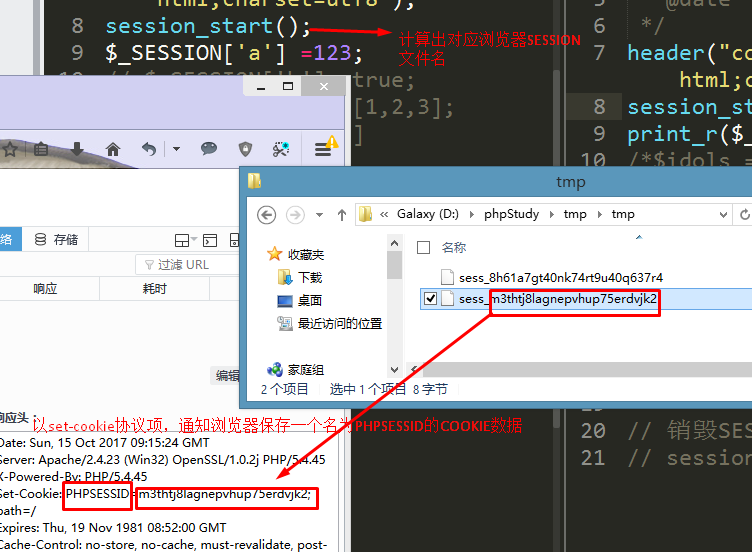
## 从HTTP协议，为什么SESSION是基于COOKIE的？

浏览器请求a.php时：

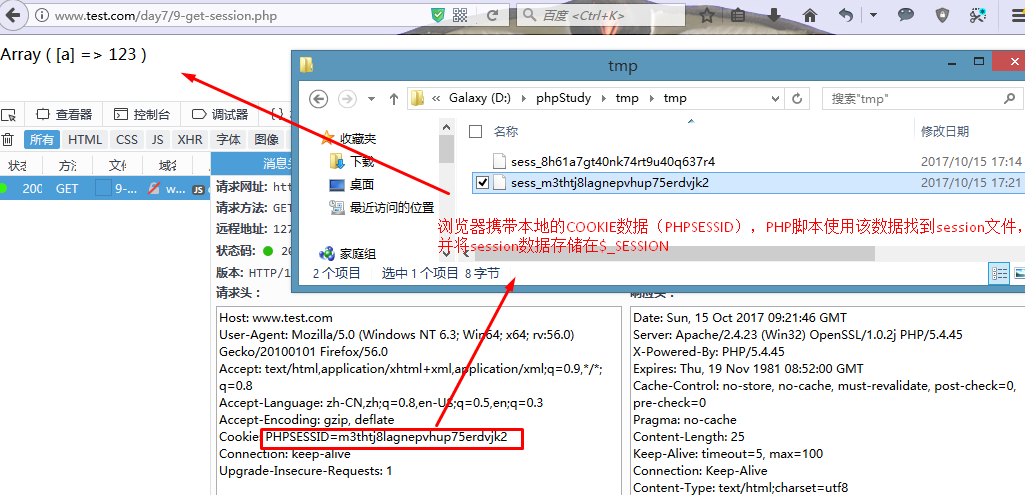
* PHP脚本中，基于不同浏览器、使用特定的算法，生成浏览器对应的唯一身份信息。并判断该浏览器对应的SESSION文件是否存在。不存在则创建以session开头，后缀以与浏览器对应的字符串身份信息。



* 响应时，PHP脚本会将当前SESSION会话的会话名称（文件名一部分），以COOKIE数据的形式返回给浏览器（在setcookie协议项中，COOKIE数据名为PHPSESSID）。



* 浏览器再次请求本网站的任意页面时，自动携带名为PHPSESSID的COOKIE数据，PHP按照会话名称，查找对应session文件，将数据读取到$\_SESSION数组中。

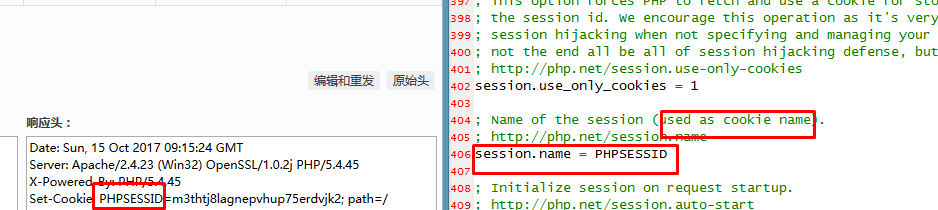


## SESSION配置选项

### session.name

设置SESSION依赖的COOKIE中，COOKIE数据的名称。

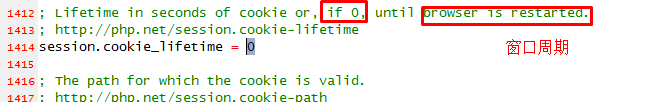
PHP脚本将当前会话的名称（session文件的名称）返回给COOKIE，默认保存在对应的COOKIE PHPSESSID中。对应HTTP协议set-cookie:PHPSESSID =…

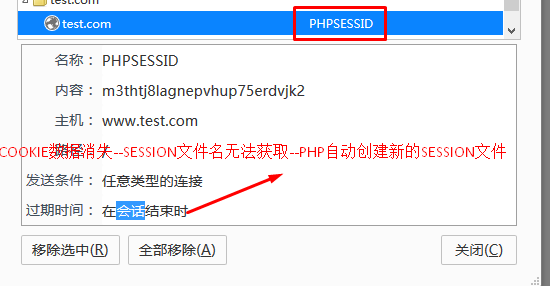


### session.cookie\_lifetime

按照cookie信息（浏览器的身份信息），读取服务器端的相应的session文件的信息。

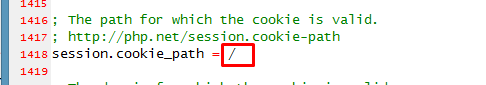
session.cookie\_lifetime=0,意味着COOKIE数据的生命周期为会话周期，关闭浏览器，COOKIE数据消失，无法使用浏览器的身份信息查到session信息。但是session暂时还是存在的。





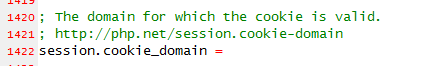
### session.cookie\_path

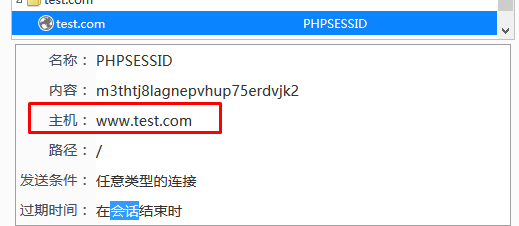
session所使用的COOKIE的有效路径。/整站有效。



### session.cookie\_domain

session所使用的COOKIE的有效域名。默认为：当前站点有效





## session与COOKIE之间的区别与联系

### 区别

* **从存储位置**：

COOKIE存储在浏览器，SESSION存储的服务器端。

* **数据安全性：**

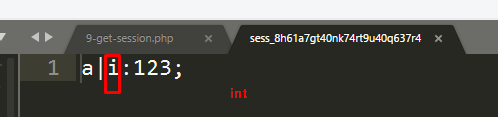
session存储在服务器端，不易获取，更加安全。（COOKIE存储安全性要求不高的信息）

* **通过HTTP协议传输的数据量：**

每次请求，COOKIE传输量较大。传递该网站相关的全部的COOKIE数据。

* **支持的数据类型：**

COOKIE支持字符串，SESSION几乎所有的数据类型。



### 联系

* 都是用来解决HTTP协议的无状态、无记忆的不足之处，以实现业务的连续性，如登录及购物车模块的实现。
* 二者都是会话技术的范畴 。
* SESSION需要依赖COOKIE（浏览器的身份信息，存储浏览器对应会话的名称）进行数据的传递。禁用COOKIE后，session无法正常使用）。

# 会话技术应用--登录、退出模块

**登录模块的核心逻辑**：

将用户提交的账号信息与数据库中存储的账号信息进行比对，两者完全一致，则登录成功。

为了防止用户访问其他页面时要求重新登录，可以将成功标志存放在SESSION中，其他页面直接判断SESSION中是否有对应数据，即可获取用户登录状态。

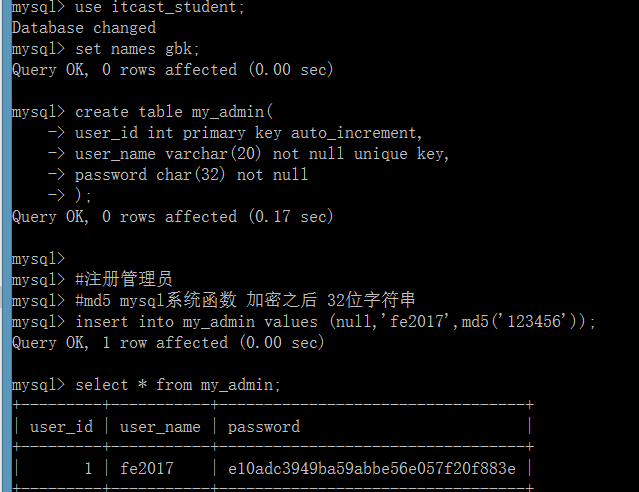
## 数据表设计

设计一个管理员表，保存管理员账号信息。管理员进行登录时，需要通过账号信息验证。

|  |  |  |
| --- | --- | --- |
| 用户编号 | int | primary key auto\_increment |
| 用户名 | varchar(20) | unique key |
| 密码 | char(32) | not null |

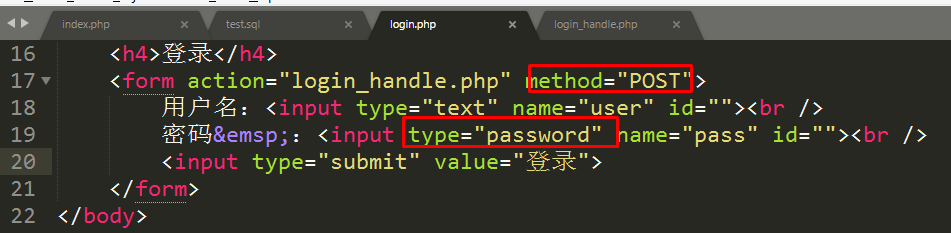
注意事项：

账号密码不能明文存储，需要使用算法加密。



## 登录表单设计

注意数据提交的方式。



## 账号信息验证

### 接收账号信息



### 账号信息验证

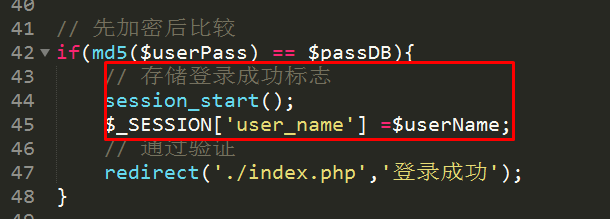
按照用户提交的用户名，查询数据库中对应的用户密码。密码不存在或者比对不成功，则登录失败。否则，登陆成功。

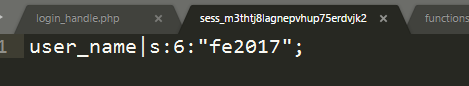


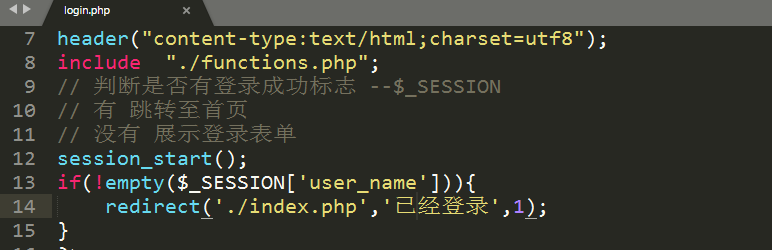


### 登录成功，设置成功标志

使用session技术来实现。







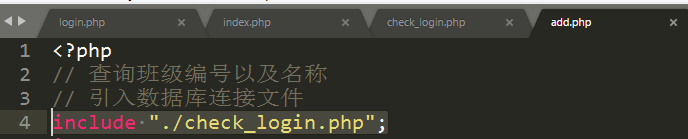
## 其他页面的登录检查

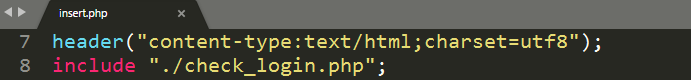
### 添加登录入口

在首页添加登录入口，点击跳转至登录页。登录成功，展示退出按钮；没有登录，展示登录按钮。

其他的页面，需要先检查是否登录。

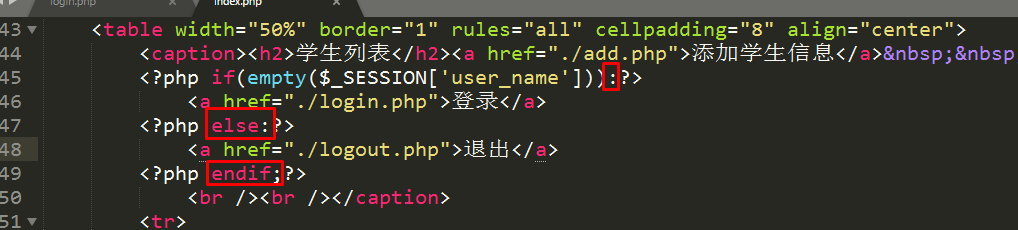






有待完善。

if分支的替换语句。



## 退出模块

删除登录标志即可。



# 自学参考网站：

[www.csdn.net](http://www.csdn.net)

[www.segmentfault.com](http://www.segmentfault.com)

stackoverflow.com