**PHP核心编程第三天—HTTP协议及文件操作**

目录

[二、 HTTP协议介绍 5](#_Toc484880330)

[1. HTTP协议概念 5](#_Toc484880331)

[1) 协议 5](#_Toc484880332)

[2) HTTP协议 5](#_Toc484880333)

[2. 特点 5](#_Toc484880334)

[1) 通常是基于b/s（browser/server）结构的访问 6](#_Toc484880335)

[2) 无连接请求 6](#_Toc484880336)

[3) 无状态 6](#_Toc484880337)

[3. 分类 6](#_Toc484880338)

[1) HTTP请求协议（request） 6](#_Toc484880339)

[2) HTTP响应协议（respond） 6](#_Toc484880340)

[4. 协议项格式 6](#_Toc484880341)

[5. 抓包工具介绍 6](#_Toc484880342)

[1) 监视工具fiddler 6](#_Toc484880343)

[2) firfox,google中如何抓取HTTP协议内容 7](#_Toc484880344)

[三、 HTTP请求协议 8](#_Toc484880345)

[1. 组成部分 8](#_Toc484880346)

[2. 请求行 8](#_Toc484880347)

[1) 格式 8](#_Toc484880348)

[3. 请求头 9](#_Toc484880349)

[1) 描述 9](#_Toc484880350)

[2) 常见的请求头协议项 9](#_Toc484880351)

[3) $\_SERVER中的请求头信息 10](#_Toc484880352)

[4) 案例：多语言站点应用 11](#_Toc484880353)

[4. 请求体 11](#_Toc484880354)

[1) POST方式请求数据举例 12](#_Toc484880355)

[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） 12](#_Toc484880356)

[四、 HTTP响应协议 12](#_Toc484880357)

[1. 组成 12](#_Toc484880358)

[2. 响应行及常见状态码 13](#_Toc484880359)

[1) 200 13](#_Toc484880360)

[2) 302 13](#_Toc484880361)

[3) 403 13](#_Toc484880362)

[4) 404 14](#_Toc484880363)

[5) 500 14](#_Toc484880364)

[6) 状态码总结： 15](#_Toc484880365)

[3. 响应头 15](#_Toc484880366)

[1) 常见的协议项 15](#_Toc484880367)

[4. 响应体 16](#_Toc484880368)

[五、 常见HTTP响应设置 16](#_Toc484880369)

[1. header函数 16](#_Toc484880370)

[2. 跳转 16](#_Toc484880371)

[1) 协议项 17](#_Toc484880372)

[2) 状态码 17](#_Toc484880373)

[3. 刷新 17](#_Toc484880374)

[1) 跳转与刷新的区别 17](#_Toc484880375)

[4. 内容类型设置 17](#_Toc484880376)

[六、 PHP模拟HTTP请求 18](#_Toc484880377)

[1. CURL库介绍 18](#_Toc484880378)

[2. 操作步骤 18](#_Toc484880379)

[1) 初始化一个CURL资源 18](#_Toc484880380)

[2) 设置CURL执行的相关选项 18](#_Toc484880381)

[3) 执行curl会话并获取结果 18](#_Toc484880382)

[4) 关闭资源 18](#_Toc484880383)

[3. 案例 19](#_Toc484880384)

[1) 请求传智首页页面 19](#_Toc484880385)

[2) 案例：请求百度页面，将‘百度一下’修改为‘问飞哥一下’ 19](#_Toc484880386)

[七、 文件编程 20](#_Toc484880387)

[1. 介绍 20](#_Toc484880388)

[1) 文件编程的必要性 20](#_Toc484880389)

[2) 文件编程操作分类 20](#_Toc484880390)

[2. 目录操作 20](#_Toc484880391)

[1) 创建目录相关 20](#_Toc484880392)

[2) 打开、读取目录 21](#_Toc484880393)

[3) 目录操作应用：读取文件夹 22](#_Toc484880394)

[4) 封装读取目录的函数 23](#_Toc484880395)

[5) 删除目录 24](#_Toc484880396)

[6) 关闭目录 24](#_Toc484880397)

[3. 文件操作（PHP4） 24](#_Toc484880398)

[1) 打开文件相关函数 24](#_Toc484880399)

[2) 写入文件fwrite及不同打开方式对数据写入的影响 25](#_Toc484880400)

[3) 读取文件fread 27](#_Toc484880401)

[4) 关闭文件fclose 27](#_Toc484880402)

[5) 重命名 28](#_Toc484880403)

[6) 删除文件(物理文件) 28](#_Toc484880404)

[4. PHP5文件操作函数 28](#_Toc484880405)

[1) file\_get\_contents($path/$url) 28](#_Toc484880406)

[2) 案例：抓取百度首页，将百度一下替换为问飞哥一下 29](#_Toc484880407)

[3) file\_put\_contents(目标名称，内容) 29](#_Toc484880408)

[4) 案例：抓取百度首页，将百度一下替换为谷歌一下，并存储为baidu.html 29](#_Toc484880409)

[八、 遍历文件夹（读取所有的子目录及文件）（面试） 30](#_Toc484880410)

[1. 实现的逻辑 30](#_Toc484880411)

[2. 关键代码 30](#_Toc484880412)

[1) 打开目录，循环读取里面的文件及子目录 30](#_Toc484880413)

[2) 判断是否是目录，是目录，递归调用封装好的函数 30](#_Toc484880414)

[3) 处理不同层级之间的缩进问题 30](#_Toc484880415)

[4) 处理中文乱码的问题 31](#_Toc484880416)

[5) 最终效果： 32](#_Toc484880417)

[九、 (了解)文件下载 33](#_Toc484880418)

[1. 实现方法介绍 33](#_Toc484880419)

[2. 步骤介绍 33](#_Toc484880420)

[1) 通知浏览器返回的数据类型 33](#_Toc484880421)

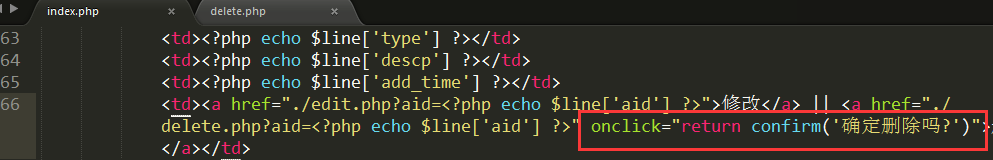
[2) 指导浏览器如何保存文件(附件方式，弹出下载窗口) 33](#_Toc484880422)

[3) 将文件传输给浏览器(传输下载的内容)： 34](#_Toc484880423)

**回顾：**

* 删除模块

1. 防止误操作：



1. 数据处理：

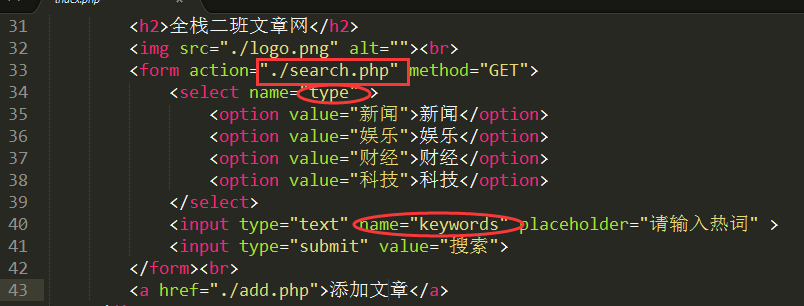


* 站内搜索

本质使用where …like

1. 需要一个表单：收集类型及关键字





1. 在search.php页面完成数据处理：将文章信息以列表的形式展示



**思考：**

* 从HTTP协议角度，分析GET方式、POST方式提交数据的区别？
* 说出常见的状态码及其意义
* 常见的响应设置有哪些？
* 定义一个函数，实现多层文件夹的遍历读取

# 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？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）

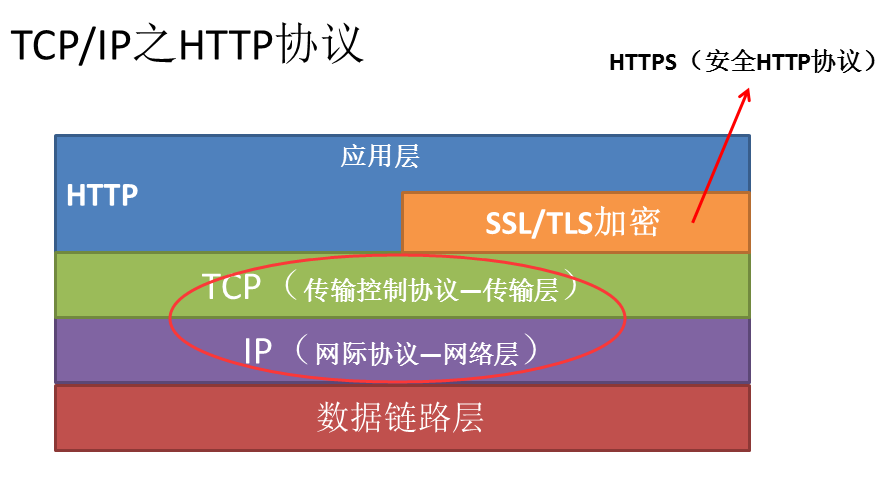
有：HTTP（超文本传输协议），ftp（文件传输协议，将网站程序传输到Apache服务器），SMTP、pop3（收发邮件的协议），telnet（远程登录协议）。

### HTTP协议

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

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，以及指定显示的格式等。



## 特点

### 通常是基于b/s（browser/server）结构的访问

但也支持c/s（QQ，迅雷）访问。通过客户端发起请求，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

**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。

### 无状态

服务器是否认识客户端？

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。

登录：已经登录，请求其他页面，还需要登录。

## 分类

数据的一去一回。

### 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）

服务器向浏览器返回数据时的协议

## 协议项格式

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

告诉浏览器。

* 请求及响应客户端成。形式：

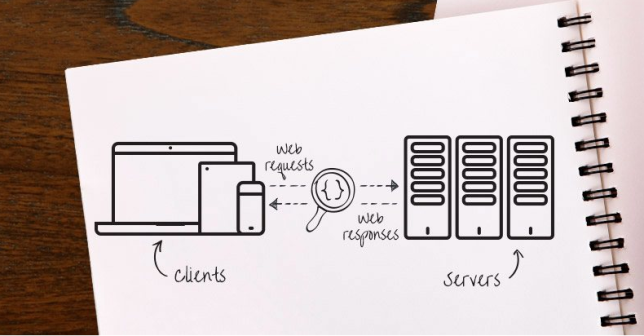
协议名：协议内容

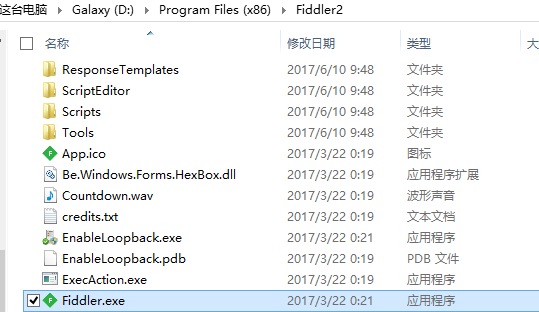
* 每个协议独占一行。

## 抓包工具介绍

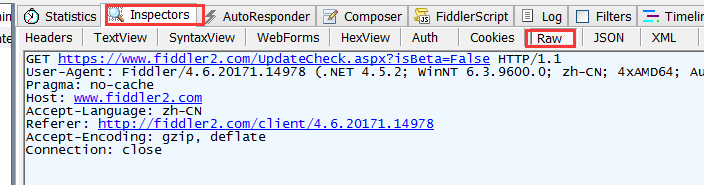
### 监视工具fiddler

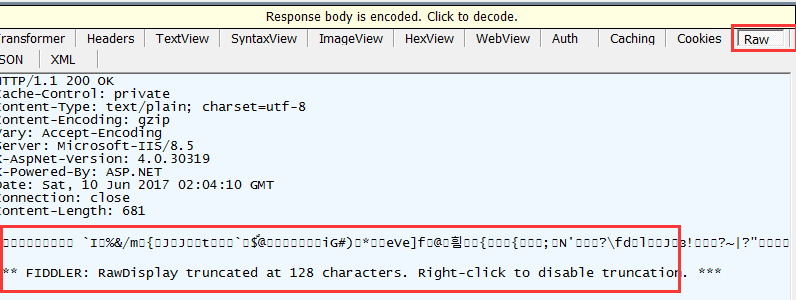
实现基于任何浏览器、系统、平台的网络调试代理工具。<http://www.telerik.com/fiddler>





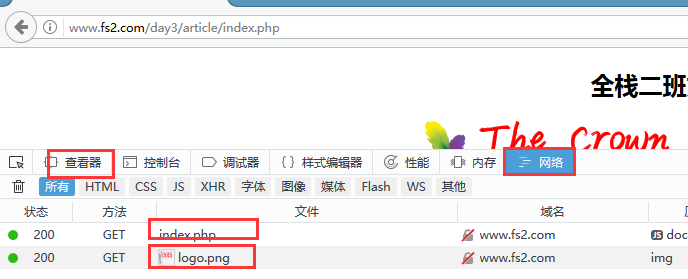
使用方法：点击请求—inspector---raw



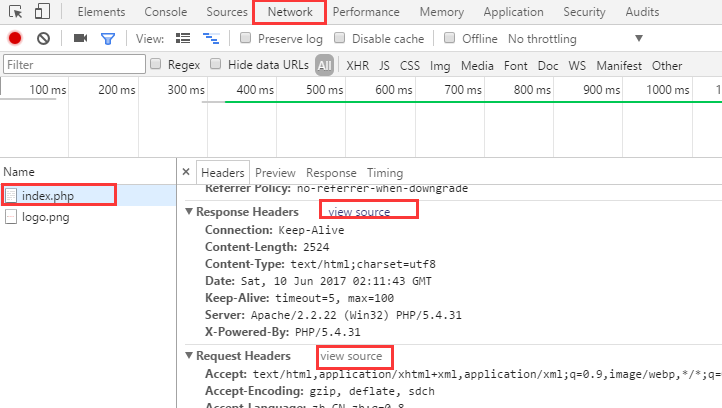


### firfox,google中如何抓取HTTP协议内容

火狐：



谷歌：F12



# HTTP请求协议

## 组成部分

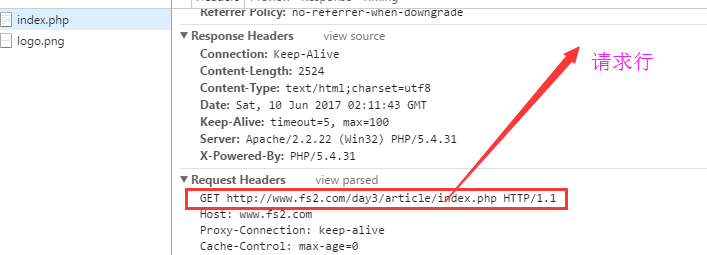
用于规范客户端（浏览器）向服务器发送数据的格式：

* 请求行（request\_line）：如请求的页面
* 请求头（request\_head）：如支持的数据类型
* 空白行：分割作用
* 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 url HTTP/1.1

HTTP1.1表示协议版本号为HTTP1.1

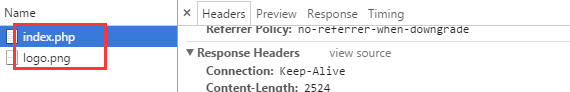
**HTTP1.0与HTTP1.1**

* HTTP/1.0 短连接

HTTP 1.0规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个TCP连接，服务器完成请求处理后立即断开TCP连接，服务器不跟踪每个客户也不记录过去的请求。

* HTTP/1.1 长连接。keep-Alive：”timeout=5,max=100”

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。（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）。

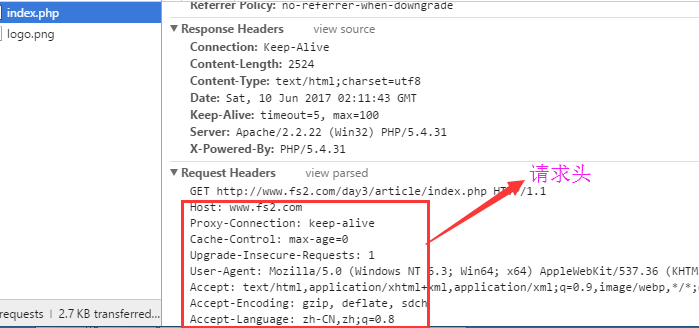


## 请求头

### 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，并且每个协议项都要独占一行。

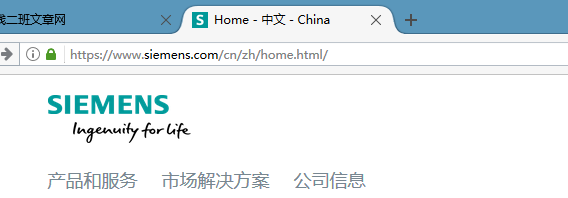
### 常见的请求头协议项

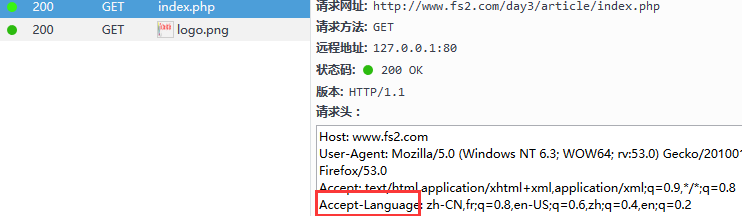


* 常见的请求头协议项名称：
  + host：当前url中所要请求的服务器的主机名（域名）。
  + accept：表示浏览器可以接收的数据类型(MIME类型)

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

* + accept-language：可以接收的语言类型，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，比如：

识别客户端设备的重要信息。

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

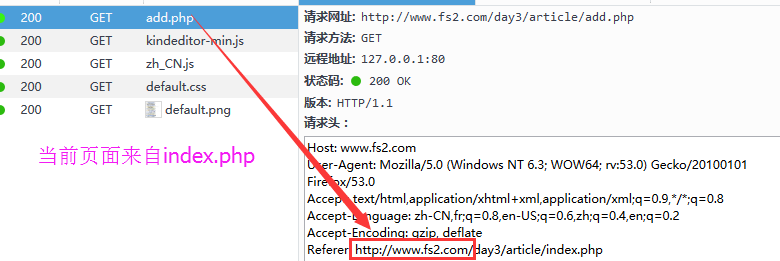
winNT6.1—win7

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）。

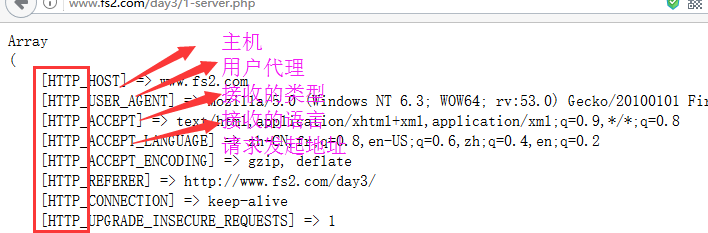




### $\_SERVER中的请求头信息

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

可以获取HTTP的请求头协议项信息。协议项前面添加了”HTTP\_”。



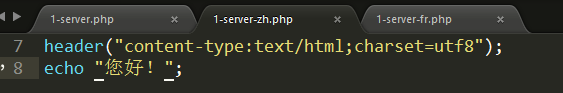
### 案例：多语言站点应用

判断首选语言是中文，展示中文页面。否则展示其他页面。



代码：





## 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。安全性高。

### POST方式请求数据举例

包括一般表单项及文件域。



### 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）

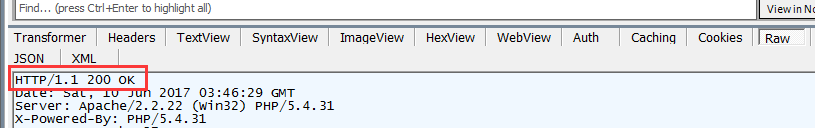
|  |  |  |
| --- | --- | --- |
| **请求方式** | **GET** | **POST** |
| 请求数据位置 | 地址栏，请求行中 | 请求体中（一般表单域及文件域） |
| 安全性 | 低 | 高 |
| 数据量 | 小（2kb） | 大(8M) |

# HTTP响应协议

## 组成

* 状态行（响应行）：成功或者失败的状态。
* 响应头：如避免乱码content-type:text/html;charset=utf8;
* 空白行
* 响应主体（数据）:请求的HTML，PHP echo 的数据。

## 响应行及常见状态码



格式：

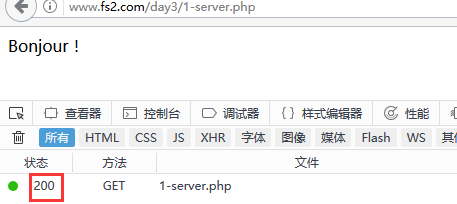
协议版本 状态码 状态描述

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。



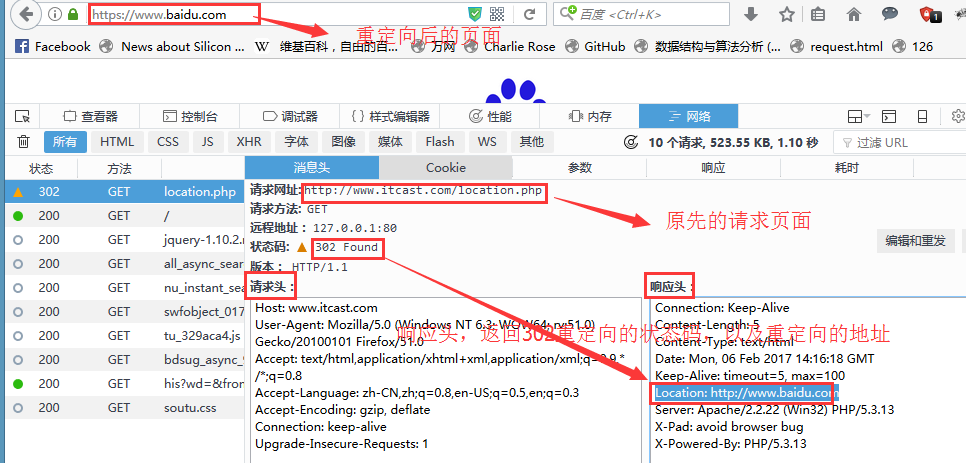
### 302

302 redirect重定向

重定向至其他页面：通知浏览器如何跳转、刷新。

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

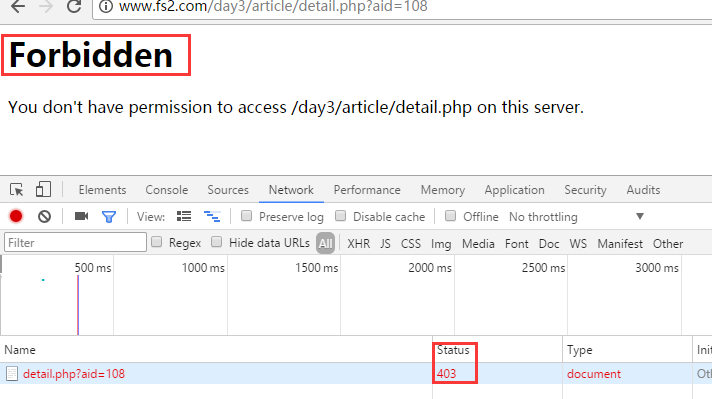
header(“refresh:3;url=http://www.itcast.cn”);



### 403

403 forbidden 禁止访问

如何模拟：删除虚拟主机的配置段中的directory.



### 404

404 Not Found未找到页面



404页面可以设置：

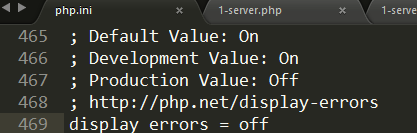


### 500

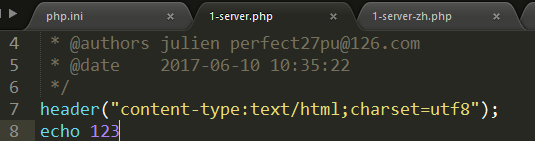
500 internal server error 服务器内部错误

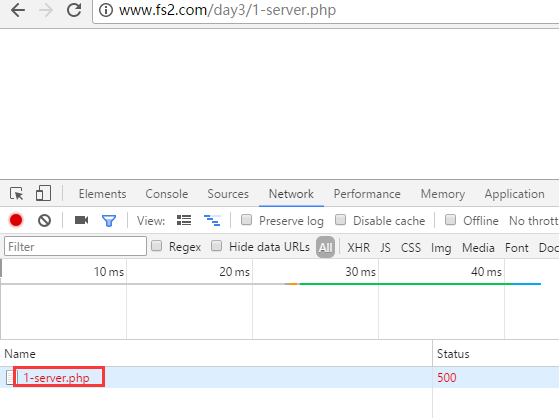
**模拟方法：**

* 在php.ini中设置：display\_errors = off



* 写一行错误代码：





### 状态码总结：

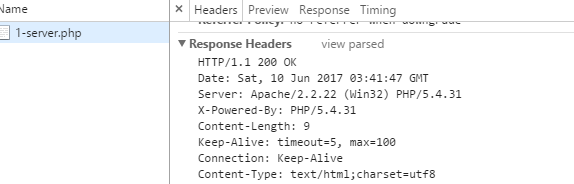
|  |  |  |
| --- | --- | --- |
| 状态码 | 状态描述 | 意义 |
| 200 | ok | 请求成功 |
| 302 | redirect | 重定向 |
| 403 | forbidden | 禁止访问 |
| 404 | not found | 为找到页面 |
| 500 | internal server error | 服务器内部错误 |

## 响应头

乱码：通过响应头解决，content-type

格式：协议项 ：协议值

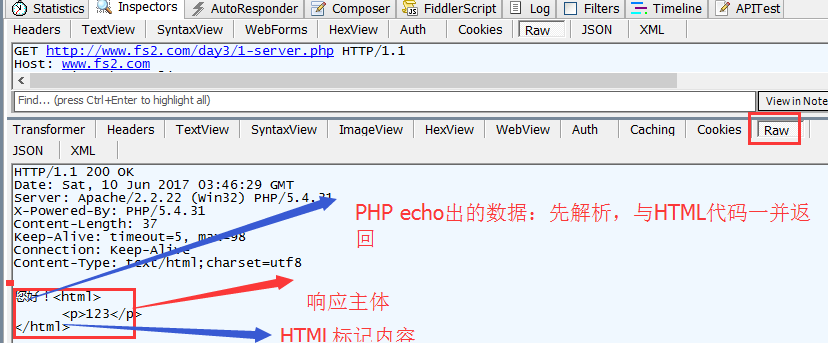
### 常见的协议项



* + Date:被请求页面的最后更新时间
  + Server:服务器版本信息
  + Content-Length:内容长度，单位为字节。
  + Content-Type: 内容类型。解决乱码。

## 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数据。



# 常见HTTP响应设置

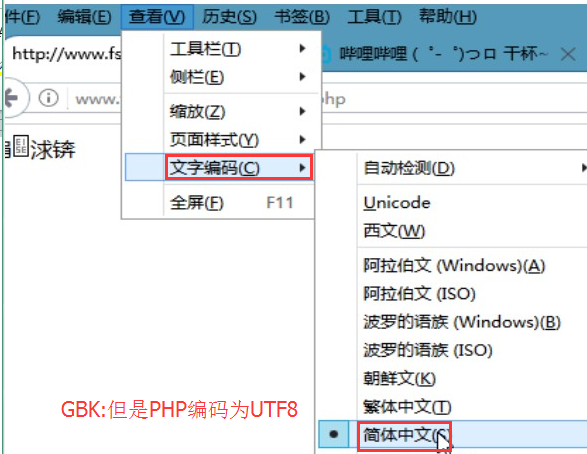
响应设置：通过模拟HTTP响应协议（响应头），告知浏览器该怎么做。

## header函数

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。

header设置响应头信息。

乱码：



## 跳转

通知浏览器，从一个页面跳转其他页面



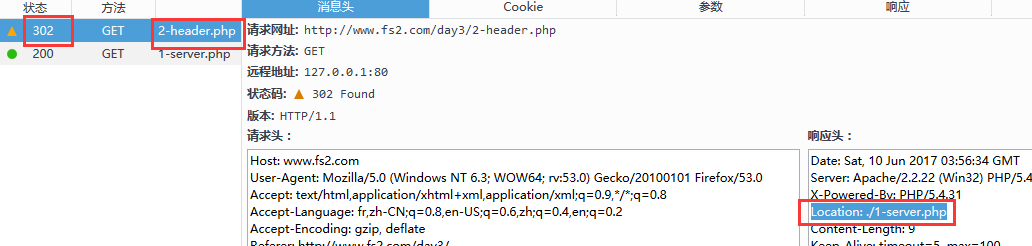
### 协议项

location：地址；

通知浏览器如何跳转。

### 状态码

302 重定向



后续代码：不执行。

## 刷新



### 跳转与刷新的区别

刷新：需要设置等待时间。等待时间之内，执行后续代码。时间到，它就跳。

跳转：立即跳转，后续代码不执行。

## 内容类型设置

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

通知浏览器，数据类型是什么，需要使用哪种方式解析处理数据。

text/html:浏览器使用HTML解析器解析。

header(‘content-type:image/png’); //返回的是图片

也可以实现下载功能。

# PHP模拟HTTP请求

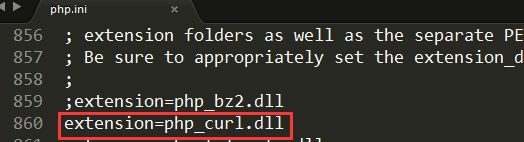
## CURL库介绍

URL：统一资源定位符。Uniform resource locator.

* cURL (Client Uniform Resource Locator) 是一个利用URL语法规定来传输文件和数据的客户端工具，客户端向服务器请求资源的工具。
* 支持很多协议，如HTTP、FTP、TELNET等。
* 功能：
  + 所有的网络资源都可以使用CURL访问及下载
  + 上传文件、下载文件
  + 获取接口数据：验证码

## 操作步骤

需要在PHP.ini中，开启PHP的CURL扩展库。



### 初始化一个CURL资源

curl\_init();

成功返回curl资源，失败返回false。

### 设置CURL执行的相关选项

curl\_setopt(resource $res,选项，选项值)；

有许多选项可供设置，这些选项用来指定URL请求的各个细节。

选项都是以“CURLOPT\_”开头，为常量。

* CURLOPT\_URL

可以设置对应的请求URL地址。

* CURLOPT\_RETURNTRANSFER ：是否以文件流的形式返回。

### 执行curl会话并获取结果

curl\_exec(resource $res);

参数必须为CURL资源。

### 关闭资源

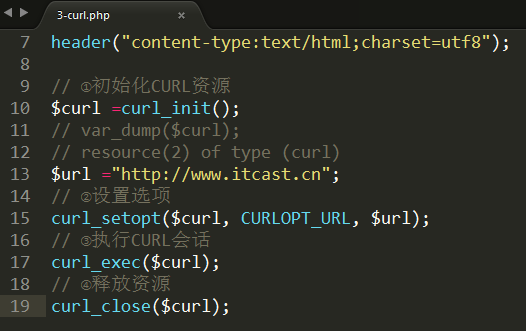
资源：fileinfo资源、数据库连接资源、查询结果集、CURL资源

对于资源来说，一般是占据的内存比较大。不使用时，需要立即关闭，以释放内存。

curl\_close(resource $res);

## 案例

### 请求传智首页页面



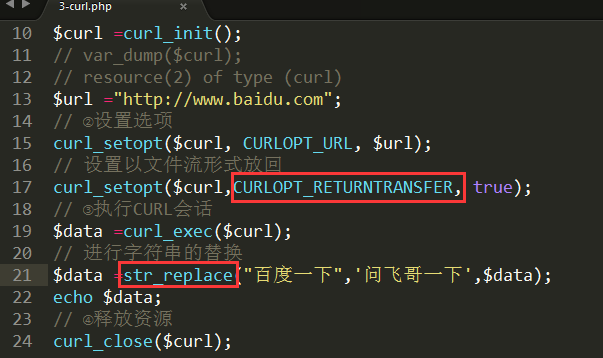


### 案例：请求百度页面，将‘百度一下’修改为‘问飞哥一下’

请求回来的数据，不是直接输出至浏览器，需要进行后期加工。

**CURLOPT\_RERTURNTRANSFER**：数据以文件流的形式返回。

str\_replace(要查的内容，要替换成的内容，目标字符串);





# 文件编程

一个问题 ：如何读取存储在TXT文本中的文件内容？

## 介绍

### 文件编程的必要性

* 统计文件数量、大小等
* 创建文件夹（判断是否存在该文件夹，不存在则创建）、删除文件夹等
* 文件上传及下载

### 文件编程操作分类

文件系统：

目录（文件夹），英文为：directory/dir

文件:英文为file.

文件编程操作：

目录操作及文件操作

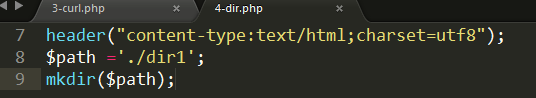
## 目录操作

### 创建目录相关

* mkdir

mkdir(路径名[，0777,是否递归创建=false])；

普通创建。对于已经存在的目录，再次创建，返回警告信息



**递归创建**

0777:Linux中的用户权限管理比较严格。权限可以分为读、写、执行权限。同时，需要定义该用户、该组其他用户、组外用户的权限。

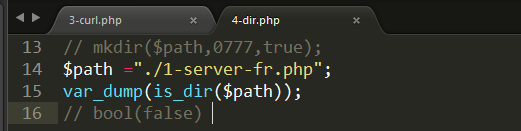
0777表示该用户、该组的其他用户、其他组的用户都是可读、可写、可执行权限。



* is\_dir

is\_dir($path);

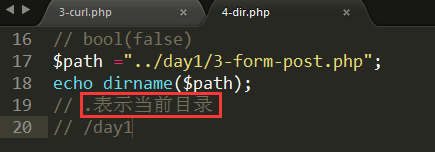
判断是否是目录。



* dirname

dirname($path);

返回路径的父目录（去除文件名之后的目录）



* realpath

realpath($path);

返回磁盘上的真实路径（绝对路径）

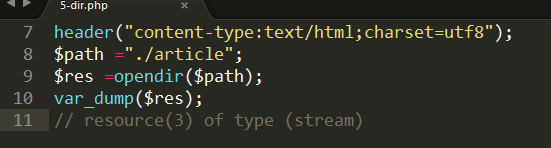


### 打开、读取目录

读取目录的前提是已经打开目录，建立起与外部数据的引用。

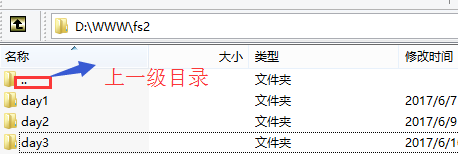
* openDir($path)

成功返回资源型（文件流），供后续的读取目录，删除、关闭目录使用。失败，返回false.



* readDir($res)

读取已经打开的目录资源，返回当前目录中的子目录名称及或者文件名称。指针移动一次，指向下一个文件或者子目录。





**指针移动的原理：**

每次执行读取一个文件或者目录，指针移动一次。指针非法，返回false。

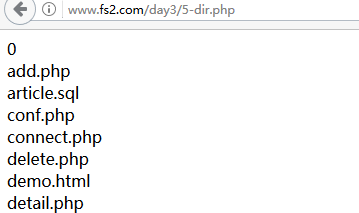
### 目录操作应用：读取文件夹

* 循环读取，排除.及..



* 排除特殊文件名：如0及字符串false。





### 封装读取目录的函数

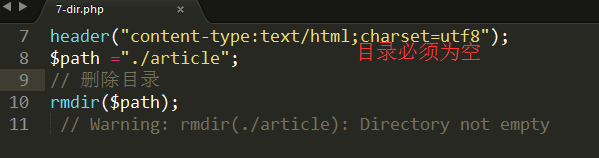
只需要一个路径参数。



### 删除目录

rmdir($path);

目录必须为空，才能删除。



### 关闭目录

释放内存（资源）。

需要关闭：数据库连接、CURL资源、目录资源。

closedir($res);



## 文件操作（PHP4）

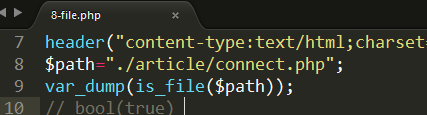
如何读取文件中的信息？

### 打开文件相关函数

* is\_file

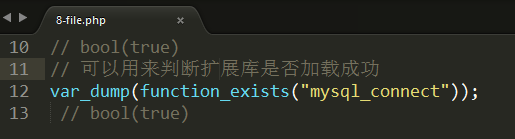
is\_file($path);

判断是否是文件。

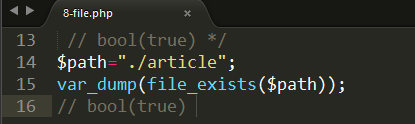


* file\_exists

function\_exists(函数名)



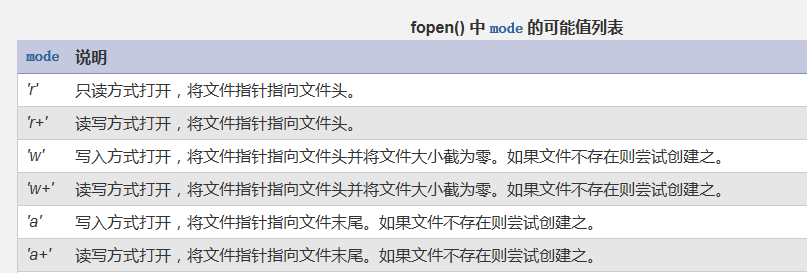
判断路径（目录及文件）存在与否。



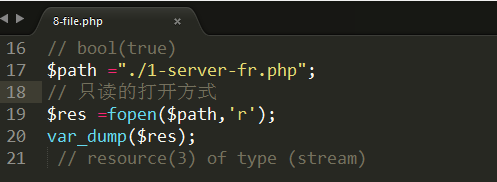
* fopen：正确的打开方式

fopen($path,模式);

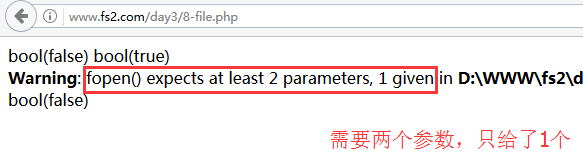
以指定的模式打开文件。



打开成功，返回资源型，供后续的读写操作使用。



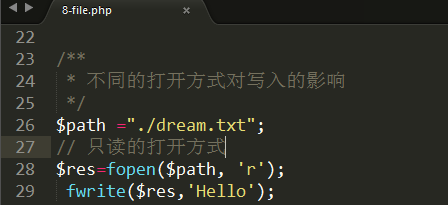
错误提示：



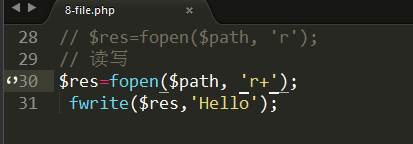
### 写入文件fwrite及不同打开方式对数据写入的影响

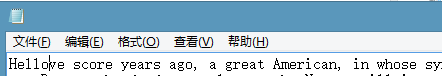
fwrite(文件资源，写入的字符串);

* r:只读，不能写入

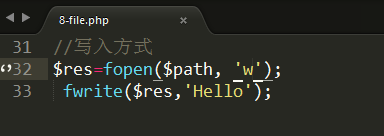


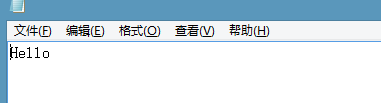
* r+：读写，指针指向文件的开头。从文件开头开始写入，后面的相应数据被覆盖。



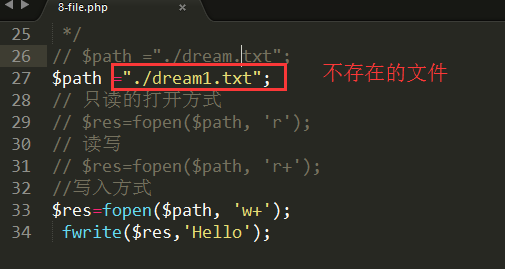


* w:以写入的方式打开，文件清空。指针指向文件的开头，从开头开始写入。





* w+:以读写的方式打开，文件清空。指针指向文件的开头，从开头开始写入。W及W+对于不存在的文件，先创建再操作。



* a:add,append写入方式打开。指针指向文件的末尾。对于不存在的文件，自动创建



* a+:读写方式打开，追加写入。不存在的文件自动创建。

### 读取文件fread

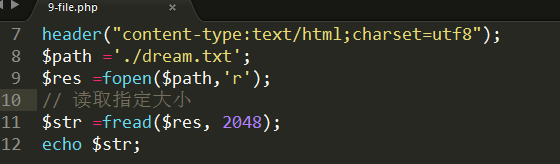
fread($res,长度);

读取指定长度（byte）的文件流。

读取一部分，指针移动一次。

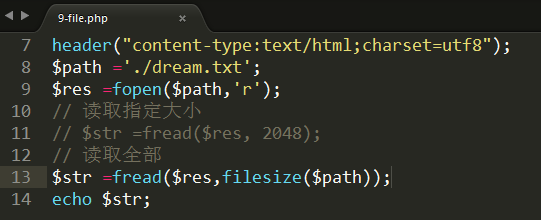
* 按照一定速度读取

建议使用该方法，用户占用的带宽少。



* 一次性读取

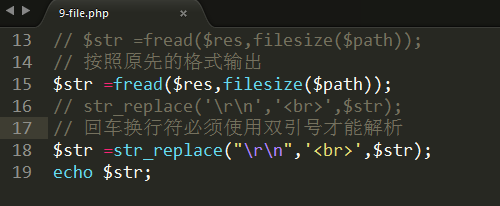
filesize(文件名)；



获取文件的大小。

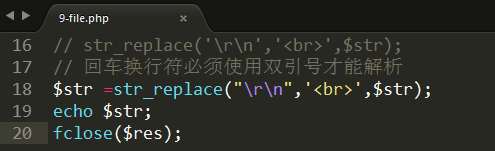
* 按照文件中数据的样式输出

文件系统回车换行符为\r\n，浏览器的<br>



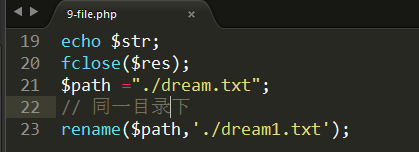
### 关闭文件fclose

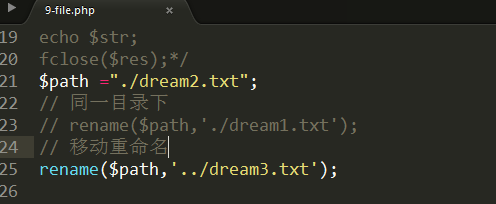
释放资源。



### 重命名

rename(原始名字，新名字)；

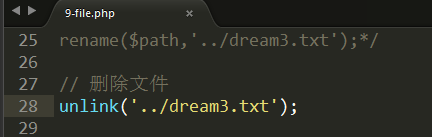




### 删除文件(物理文件)

unlink($path);

验证码过期或者处理后的缩略图均可以删除。

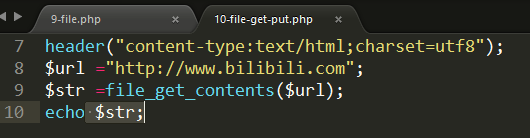


## PHP5文件操作函数

### file\_get\_contents($path/$url)

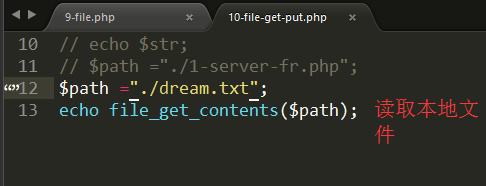
获取文件或者远程地址的全部内容。

本质对文件操作，为打开文件fopen及读取文件fread、关闭文件fclose的结合。

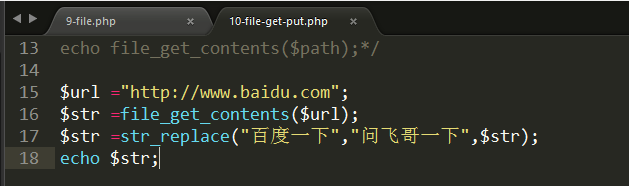




本地文件：



### 案例：抓取百度首页，将百度一下替换为问飞哥一下



### file\_put\_contents(目标名称，内容)

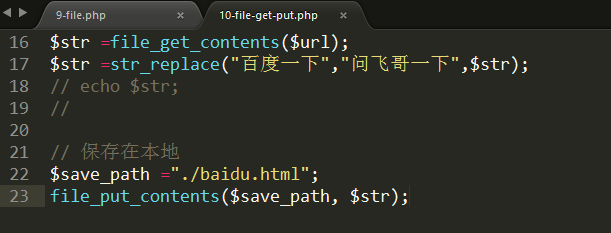
将数据存储为文件。

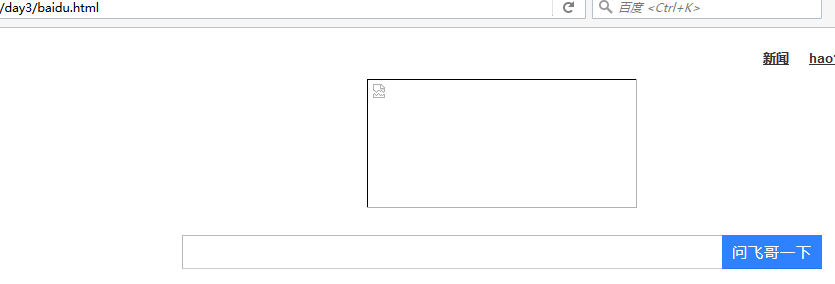
理解为：打开文件fopen及读取文件fwrite、关闭文件fclose的结合。

注意事项：任何类型数据，存储后再读取时类型全部为字符串类型。



### 案例：抓取百度首页，将百度一下替换为谷歌一下，并存储为baidu.html





# 遍历文件夹（读取所有的子目录及文件）（面试）

## 实现的逻辑

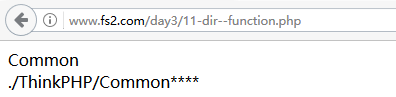
* 封装一个读取目录的函数f1。使用函数读取根目录。
* 读取到的文件及子目录先输出。
* 然后判断是文件还是目录。
* 是目录继续读取。调用f1.

## 关键代码

### 打开目录，循环读取里面的文件及子目录

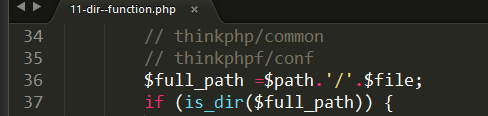
子目录的路径：



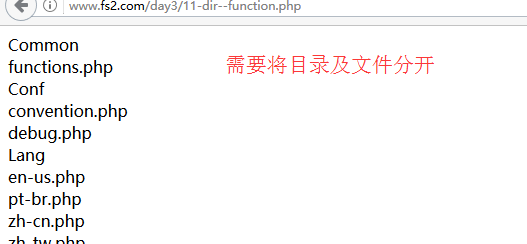


### 判断是否是目录，是目录，递归调用封装好的函数

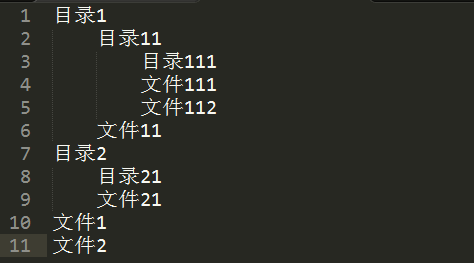
递归调用的核心：函数调用本身。



### 处理不同层级之间的缩进问题



最终效果：



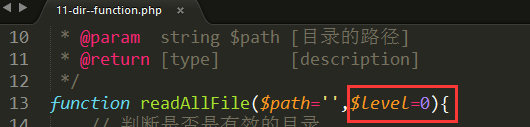
需要一个层级参数，记录当前的文件或者子目录的层级。按照层级进行缩进。

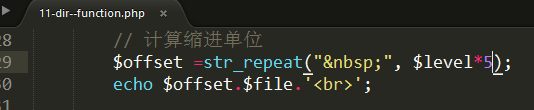
第0级 缩进0个单位

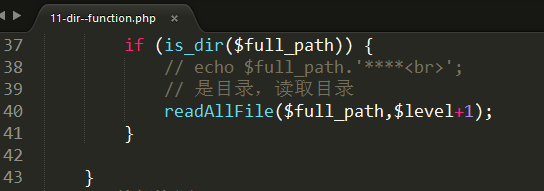
第1级 缩进1单位

使用str\_repeat函数;

str\_repeat(字符串，重复的次数)；







### 处理中文乱码的问题

乱码的本质是：字符集不一致。



* PHP编码为UTF8
* 文件系统的编码为GBK：如命令行
* 从磁盘上读取出来的文件名的编码：GBK
* echo 使用的字符集为：utf8.

即输出之前需要将字符集转换为UTF-8

string **iconv** ( string 原始的字符集, string 输出的字符集, string $str )

输出：

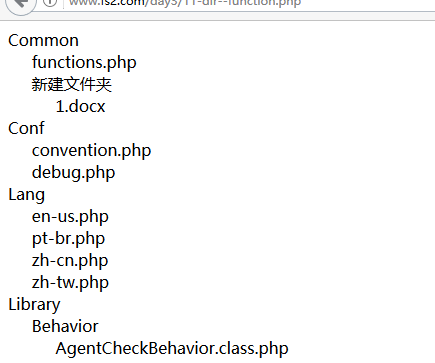




再次读取时的字符集转换：



### 最终效果：



# (了解)文件下载

结合header()函数。

## 实现方法介绍

* 可以通过a标签实现下载。

a标签实现下载的时候，浏览器会首先尝试解析（打开）所下载的文件，如果能打开，就直接在浏览器上显示了，不会给用户提供下载。





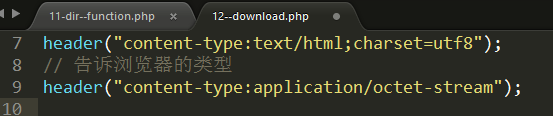
可以利用HTTP协议（header函数结合响应头），告诉浏览器，不要尝试打开文件，直接给用户下载。

## 步骤介绍

### 通知浏览器返回的数据类型

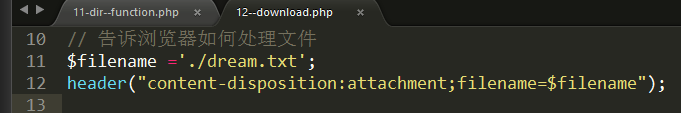
响应的内容：

header(“content-type:application/octet-stream”);



### 指导浏览器如何保存文件(附件方式，弹出下载窗口)

header(“content-disposition:attachment;filename=文件最终保存的名字”);



### 将文件传输给浏览器(传输下载的内容)：

先读取文件，然后输出给浏览器。

echo file\_get\_contents($path);

