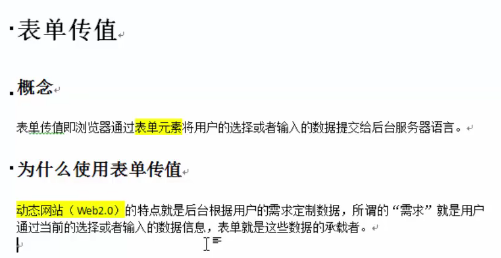
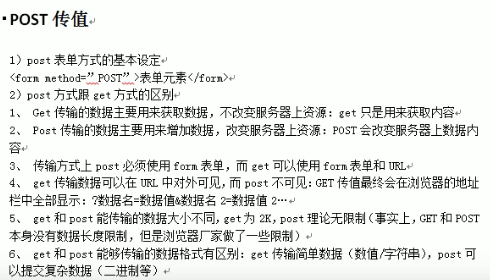
表單傳值



GET傳值



POST傳值 , POST與GET方式的區別



PHP接收數據的三種方式

不管是$\_GET , $\_POST , $\_REQUEST, 三個都是php超全局(沒有範圍限制)的預定義數組, 表單元素form內的’name’屬性的值 作為數組的下標, 而value屬性對應的值就是數組的元素值。

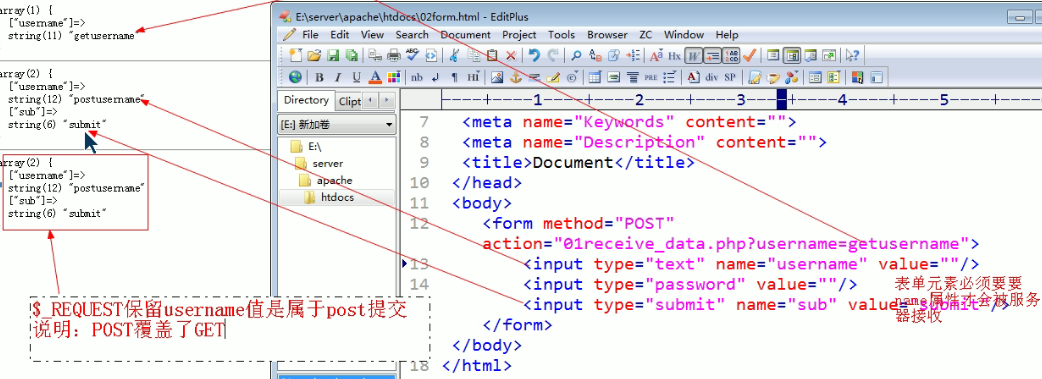
$\_GET 接收get方式的提交數據

$\_POST接收post方式的提交數據

$\_REQUEST接收get / post方式的提交數據

表單元素要有name屬性, 服務端才會接收到。

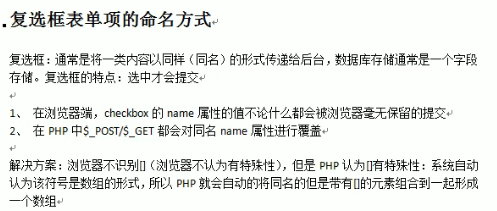
在$\_REQUEST中, 如果post與get上傳的name(鍵key)是相同, post會複蓋get的數據。



php處理複選框數據

複選框表單項 命名方式

name = 值[] name = “hobby[]”



<form method="POST" action="04-checkbox.php">

        <input type="checkbox" name="hobby[]" value="basketball">basketball

        <input type="checkbox" name="hobby[]" value="football">football

        <input type="checkbox" name="hobby[]" value="swimming">swimming

        <input type="checkbox" name="hobby[]" value="running">running

        <input type="checkbox" name="hobby[]" value="fighting">fighting

        <input type="submit" value="submit">

    </form>

$\_POST的數組內, 會有另一個數組(二維數組),放hobby[]的內容。

checkbox傳post數據時, name要加[] 。 以便php接收時以數組接收。

radio傳post數據時, 就不需要加[], 因為radio是單選。

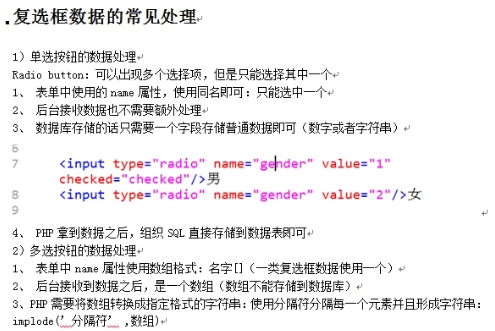
implode(連接方式, 目標數組) 數組轉換為字符串

複選框數據的常見處理

checkbox傳post數據時, name要加[] 。 以便php接收時以數組接收。

radio傳post數據時, 就不需要加[], 因為radio是單選。

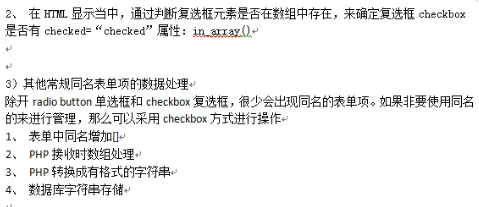
implode(連接方式, 目標數組) 數組轉換為字符串



php組織sql直接存儲到數據庫



var\_dump(explode(' , ',$hobby\_string));



in\_array();

複選框細節

如果複選框沒有複,會報錯, 因為沒有$\_POST[‘hobby]。

利用isset,

isset():判斷數據存儲的變量本身是否存在, 存在變量返回true, 不存在返回false;

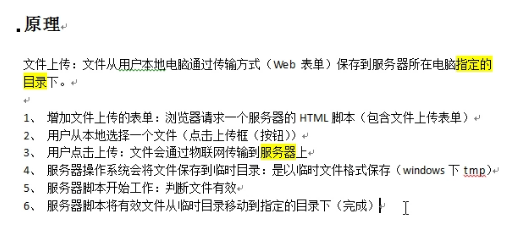
$hobby = isset($\_POST['hobby']) ? $\_POST['hobby'] : array();

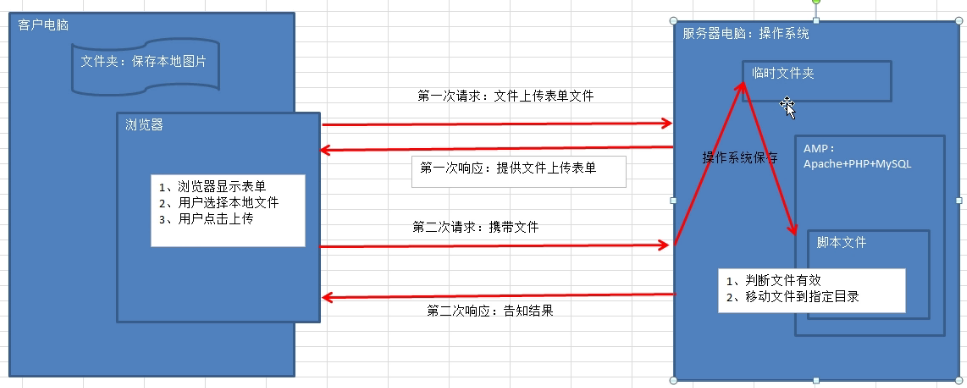
三目運算, 如果$\_POST[‘hobby]存在, 就$hobby = $\_POST[‘hobby’];

如果不存在, 就返回空數組, 這樣就不會報錯。

文件上傳

文件上傳原理





文件上傳表單寫法

method屬性: 文件上傳表單必須為post, 文件是二進制

enctype屬性: form表單屬性, 主要是規範表單數據的編碼方式。



multipart/form-data 不對字符編碼, 在使用包含文件上傳控件的表單時, 必須使用的值。

input type=”file”

<form method="POST" enctype="multipart/form-data" action="06-file-upload.php">

        <input type="file" name="image" />

        <input type="submit" name="submitBTN" value="submit">

</form>

$\_FILES變量

在php中,有一個預定義變量$\_FILES是專門用來存儲用戶上傳的文件的。

1. name: 文件在用戶端(瀏覽器端)電腦上實際存在的名字(實際用來保存文件的後綴名)
2. tmp\_name: 文件上傳到服務器後操作系統保存的temp路徑 (實際用來給php處理)
3. type: MIME類型(多功能互聯網郵件擴展) , 用來在計算機中客戶端識別文件類型 (用來確定軟件)
4. error: 文件上傳的代號, 用來告知應用軟件(php)文件接收過程中出現了什麼問題(php文檔手冊>特點>文件上傳處理>錯誤信息說明)

UPLOAD\_ERR\_INT\_SIZE , UPLOAD\_ERR\_OK等等…. 每種錯誤信息都有一種INT對應值返回到$\_FILES.error, OK的值是int(0)。 (實際用來給php根據error類型進行文件判斷)

1. size: 文件大小 (php根據實際需求來確定是否該保留)

\*\*\*

如果表單form沒有enctype=’ multipart/form-data’, 就沒有$\_FILES變量了。

<input type="file" name="image" />

 var\_dump($\_FILES);

array(1) {

["image"]=>

array(5) {

["name"]=>

string(10) "959294.jpg"

["type"]=>

string(10) "image/jpeg"

["tmp\_name"]=>

string(45) "C:\Users\Admin\AppData\Local\Temp\phpFBB3.tmp"

["error"]=>

int(0)

["size"]=>

int(960892)

}

}

移動temp文件到目標位置

1. 判斷是否為上傳的文件: is\_uploaded\_file()
2. 移動文件: move\_uploaded\_file()

文件上傳後保存到$\_FILES, 訪問文件就是: $\_FILES[表單name屬性值][元素信息]

// 那一個$\_FILES , name為'image'的$\_FILES

$file = $\_FILES['image'];

// temp文件的路徑位置

$file['tmp\_name']

move\_uploaded\_file($tmp\_name, "$uploads\_dir/$name");

$\_FILES變量:

$file = $\_FILES['image'];

$file['tmp\_name'] temp文件的路徑位置

$file['name'] 文件名

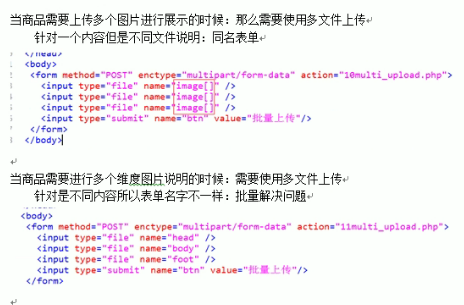
$file['type'] MIME類型

$file['error'] 報錯類型 int(0)為成功

$file['size'] 文件大小

多文件上傳

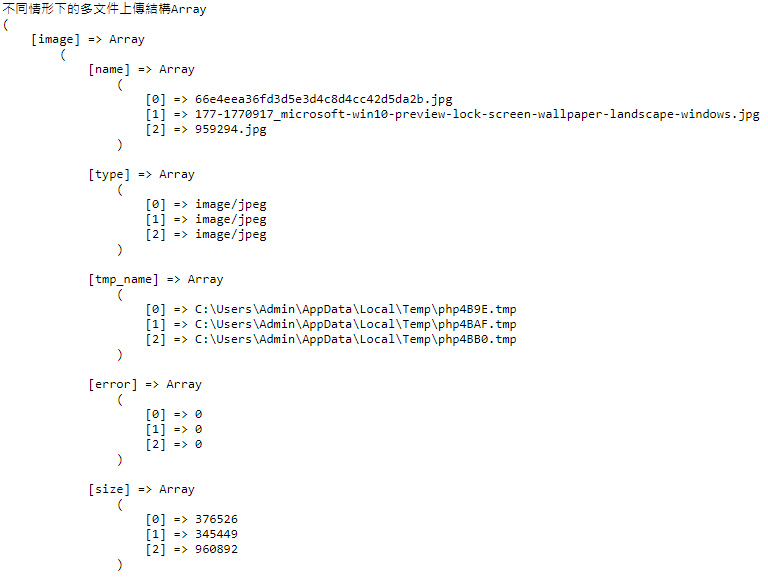
2種多文件上傳



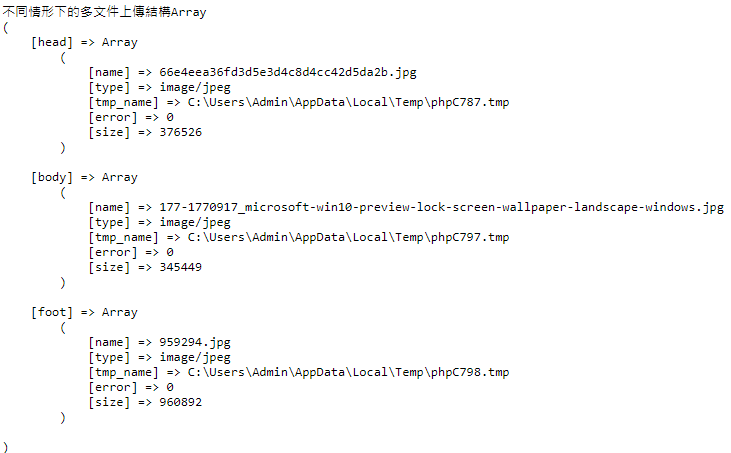
(批量上傳)同名上傳(name[]數組)表單

每個元素都有對應數量的數組, 每個文件上傳對應數組的元素索引。





(批量上傳)不同名(name不同名)上傳表單: 每個文件都會形成一個屬性自己獨立的5個元素(name,type,tmp\_name,error,name)的數組。



以上就是多文件上傳的$\_FILES變量的數據結構形式。

同名與不同名。

對多文件信息的編歷讀取和處理 (不同名文件 與 同名文件(name[]數組))

*// 不同名文件編歷處理方式*

    foreach($\_FILES as $file){

*//$file 就是一個完整的上傳文件信息*

        if(is\_uploaded\_file($file['tmp\_name'])){

            move\_uploaded\_file($file['tmp\_name'],'uploads/'.$file['name']);

        }

    };

*// 同名文件編歷處理方式*

if(isset($\_FILES['image']['name']) && is\_array($\_FILES['image']['name'])){ *//判斷存在而且是array*

*//編歷構造數組元素*

        $images = array();

        foreach($\_FILES['image']['name'] as $k => $file){

            $images[] = array(

                'name' => $file,

                'tmp\_name' => $\_FILES['image']['tmp\_name'][$k],

*// \*\*\*返回上一層數組找 再給了$k 索引*

                'type' => $\_FILES['image']['type'][$k],

                'error' => $\_FILES['image']['error'][$k],

                'size' => $\_FILES['image']['size'][$k]

            );

        };

    };

    foreach($images as $file){

*//$file 就是一個完整的上傳文件信息*

        if(is\_uploaded\_file($file['tmp\_name'])){

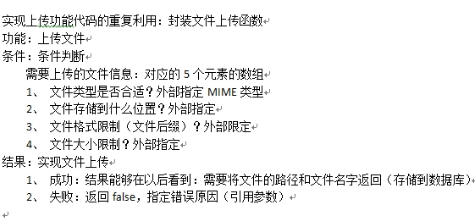
            move\_uploaded\_file($file['tmp\_name'],'uploads/'.$file['name']);

        }

    };

文件上傳後續問題

封裝文件上傳函數



09-file-upload-function.php

*// 判斷文件是否有效*

        if(!is\_array($file) || !isset($file['error'])){

*// 判斷文件存儲路徑 是否有效*

        if(!is\_dir($path)){

*// 判斷 MIME 類型*

        if(!in\_array($file['type'],$allow\_type)){

獲取後綴名

$ext = ltrim(strrchr($file['name'],'.'));

**strrchr** ( string $haystack , [mixed](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/language.types.declarations.html#language.types.declarations.mixed) $needle ) : string

该函数返回 haystack 字符串中的一部分，这部分以 needle 的最后出现位置开始，直到 haystack 末尾。

if(!empty($allow\_format) && !in\_array($ext,$allow\_format)){

*// 不允許的後綴名*

*// 構造文件名字: 類型\_年月日\_隋機字符串.ext*

因為不同用戶可能上傳同名的file, 使文件被覆蓋。

**strstr** ( string $haystack , [mixed](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/language.types.declarations.html#language.types.declarations.mixed) $needle , bool $before\_needle = **false** ) : string

返回 haystack 字符串从 needle 第一次出现的位置开始到 haystack 结尾的字符串。

haystack

输入字符串。

needle

如果 needle 不是一个字符串，那么它将被转化为整型并且作为字符的序号来使用。

before\_needle

若为 **true**，**strstr()** 将返回 needle 在 haystack 中的位置之前的部分。

\*\*\*\*\*

strstr() true:取字符串頭與needle之間 /

false:取needle與字符串尾之間(包含needle)。

strrchr() 取needle與字符串尾之間(包含needle)。

舊php的strrchr是不包含needle的

**chr** ( int $ascii ) : string

返回相对应于 ascii 所指定的单个字符。

*// 構造文件名字: 類型\_年月日\_隋機字符串.ext*

        $fullname = strstr($file['type'] , '/' ,true) . date('YYYYmmdd') ;

*// 產生隋機字符串*

        for($i =0;$i < 4 ; $i++){

            $fullname .= chr(mt\_rand(65,90)); *//大寫英文字母的ascii碼是65到90*

        }

MIME 類型

$allow\_type = array('image/jpg','image/jpeg','image/gif','image/pjpeg');

$ext = ltrim(strrchr($file['name'],'.'), '.');

ltrim 去除 取左側的 .

**ltrim** ( string $str , string $character\_mask = ? ) : string

删除字符串开头的空白字符（或其他字符）

**参数**

str

输入的字符串。

character\_mask

通过参数 character\_mask，你也可以指定想要删除的字符，简单地列出你想要删除的所有字符即可。使用..，可以指定字符的范围。

mysql擴展



連庫基本操作

php7 開始, 要用mysqli

需要在php.ini 配置文件, 開放extension=mysqli;

連接數據庫服務器

$conn = new mysqli(‘localhost’, ‘user’, ‘password’,’data\_base’); //使用new操作符，最後一個參數是直接指定數據庫

OOP Style usage

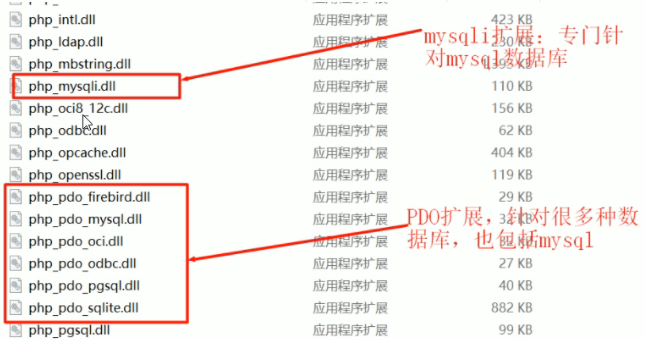
$mysqli = new mysqli('localhost', 'my\_user', 'my\_password', 'my\_db');

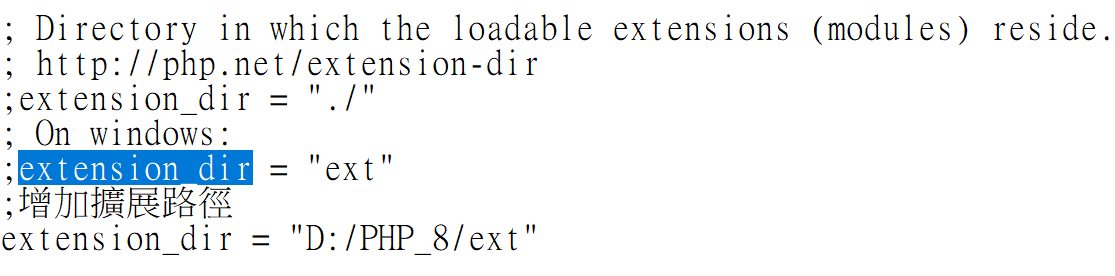
Procedural Style usage

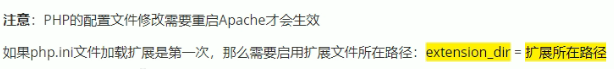
$link = mysqli\_connect('localhost', 'my\_user', 'my\_password', 'my\_db');

連接資源$link 是超全局的。

pdo擴展, 針對行多種數據庫, 也包括mysql。





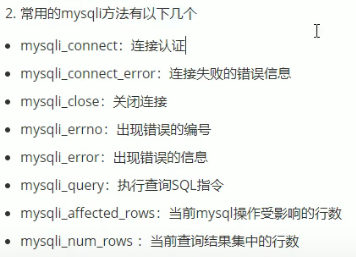


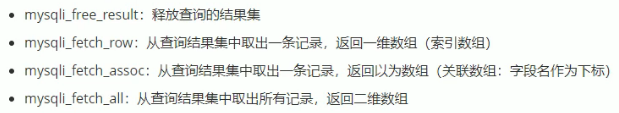
加載php.ini文件擴展。

mysqli\_ 的函數是 面向過程

mysqli:: 的函數是 面向對象

常用的mysqli方法 (面向過程)





mysqli\_connect 連接認證

mysqli\_conncet\_error 連接失敗的錯誤信息

mysqli\_close 關閉連接

mysqli\_errno 出現錯誤編號

mysqli\_error 出現錯誤信息

mysqli\_query 執行查詢SQL指令

mysqli\_affected\_rows 當前mysql操作受影響的行數 (寫操作)

mysqli\_num\_rows 當前查詢結果集 中的行數

mysqli\_free\_result 釋放查詢的結果集

mysqli\_fetch\_row 從查詢結果集 中取出一條記錄,

返回一維數組(索引數組)

mysqli\_fetch\_assoc 從查詢結果集 中取出一條記錄,

返回一維數組(關聯數組:字段名作為下標)

mysqli\_fetch\_all 從查詢結果集 中取出所有記錄,

返回二維數組

\*\*\*\*\*

mysqli\_fetch\_assoc和\_row都是有指針, 用了一次會使原數組的指針指向下一個索引。

mysqli::query -- mysqli\_query — 对数据库执行一次查询

在文檔中沒有 mysqli\_query 就查mysqli::query 因為是一樣的。

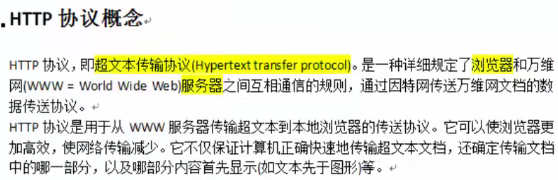
**mysqli\_query** ( [mysqli](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/class.mysqli.html) $link , string $query , int $resultmode = MYSQLI\_STORE\_RESULT ) : [mixed](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/language.types.declarations.html#language.types.declarations.mixed)

\*\*\*\*\*

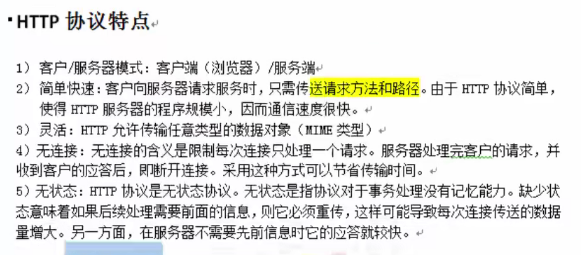
*// create table用utf8  客戶端用set names big5;*

http協議

http協議概念



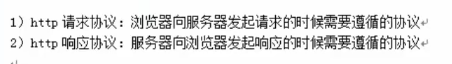
http協議特點



無狀態: 可能使每次連接傳送的數據量增大。(連續處理)

http協議分類

請求, 響應



HTTP請求 (請求行/頭/空行/體)

請求行

1. 形式: 請求方式 資源路徑 協議版本號
2. get /index.php http/1.1

現在多數都用http/1.1

第一行, 獨占一行

請求頭

請求頭就是各項協議的內容, 具體協議的內容不會每次都使用全部(有多種);

1. Host: 接收請求的主機地址(必須)
2. Accept: 當前請求能夠接收服務器返回的類型 (MIME類型)
3. Accept-Language: 接收的語言
4. User-Agent: 客戶瀏覽器所在點的一些信息

請求頭不固定數量, 每個請求協議也是獨占一行, 最後會有一行空行。(用來區分請求頭和請求體)

請求體

只有post請求會有請求體, get請求所有數據都是跟在URL後(query), 會在請求行中的資源路徑上體現。

get的request.query格式: ?資源名字=資源值&資源名字=資源值…



http響應

響應行

1. 形式 協議版本號 狀態碼 狀態消息

http/1.1 200 OK 獨占一行

常見狀態碼

2. 200 ok :成功

3. 403 forbidden: 沒權限訪問

4. 404 not found : 未找到頁面

5. 500 server internal error: 服務器內部錯誤

6. 302 Found 重定向

1XX: 信息

2XX: 成功

3XX: 重定向

4XX: 客戶端錯誤

5XX: 服務器錯誤

響應頭

具體協議內容

1. 時間:Date: Thu, 04 Nov 2021 14:23:10 GMT
2. 服務器: Server: Apache/2.4.51 (Win32) OpenSSL/1.1.1l PHP/8.0.12
3. 內容長度: Content-Length: 1571 , 數據具體的字節數 (響應體)
4. 內容類型: Content-Type: text/html , 告訴瀏覽器對應的數據格式

以上只是列舉常見的響應頭,不是全部。

每個響應頭, 獨占一行。

響應體

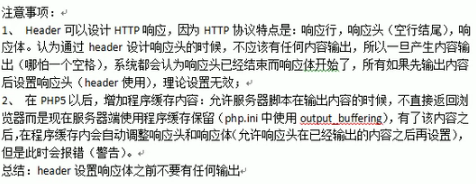
實際服務器響應給瀏覽器的內容。



HTTP響應

常見HTTP響應設置與使用

php 可以利用 header() 來實現修改 http響應頭



header() 應該放在最前面。

直接在header中設置重定向 location。

Location: 重定向,立即轉跳(響應體不用解析)



header('Location:13-location.html');

Refresh: 重定向, 定時轉跳 (響應體會解析)



header('Refresh:3;url=13-location.html');

Content-Type: 內容類型, MIME類型

通過內容告知(MIME類型), 瀏覽器正確解析內容。

header('Content-Type:text/html;charset=utf8');

Content-disposition: 內容類型, MIME類型擴展, 激活瀏覽器文件下載對話框。

瀏覽器在解析內容的時候, 默認是直接解析, 有時候需要瀏覽器 不解析, 當做內容下載成文件。

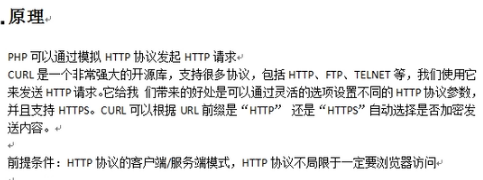
attachment 附件(下載) , filename 文件名

header('Content-disposition:attachment;filename=girl.jpg');

看到content-disposition , 瀏覽器不解析, 直接下載文件。

php 模擬 http請求

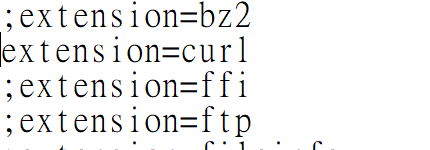
原理 利用curl擴展



CURL 開源庫

php.ini中

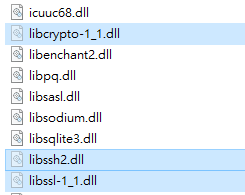
開放extension=curl



使用curl 這3個文件要放到windows

php官方說明

https://www.php.net/manual/en/curl.installation.php



使用curl

1. 建立連接: curl\_init()初始化 cURL 会话 (激活一個curl連接功能)
2. 設置請求選項: curl\_setOpt()

为 cURL 并行处理设置一个选项(設定連接選項)

curl\_setopt($ch , 連接選項 , 連接選項value) 設置連接選項

**curl\_setopt** ( resource $ch , int $option , [mixed](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/language.types.declarations.html#language.types.declarations.mixed) $value ) : bool

CURLOPT\_URL: 連接對象

CURLOPT\_RETURNTRANSFER: 將服務器執行的結果(響應) 以文件流的形式

返回給請求界面 (PHP腳本)

CURLOPT\_POST: 是否使用post請求。(默認get)

CURLOPT\_POSTFIELDS: 用來傳遞post提交的數據, 分為兩種方式: 字符串與數組。

字符串:(name=abc&password=123)

數組:(array(‘name’=>’abc’….))

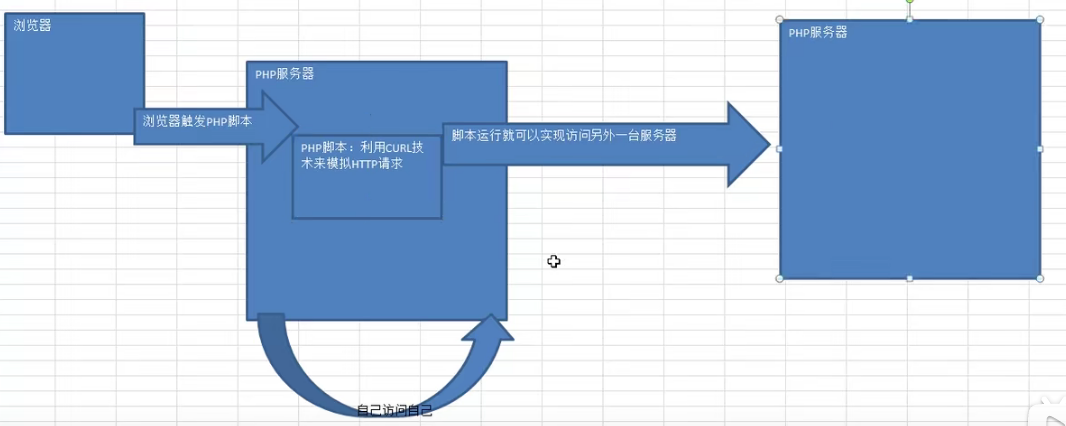
CURLOPT\_HEADER: 是否得到響應的header信息(響應頭),默認不獲取。

1. 執行請求: curl\_exec()

运行当前 cURL 句柄的子连接 (執行選項 (與服務器發起請求) 得到服務器返回的內容)

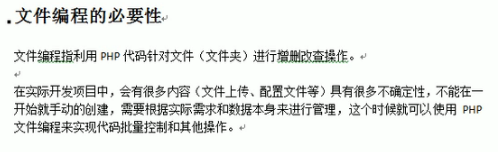
1. 關閉連接: curl\_close() 关闭 cURL 会话 (關閉資源)

curl 的實現 自己(php)訪問自己(php)

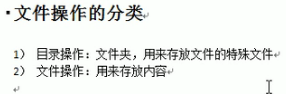


文件編程

文件編程的必要性



文件編程的分類



文件編程 - 路徑操作

文件操作創建目錄結構

mkdir() make directory 創建成功返回true, 失敗返回false;

$res = @mkdir('directory');

抑制錯誤@, 如果本身就有directory目錄, 就不用報錯了。

刪除目錄

rmdir() remove directory

@rmdir('dir');

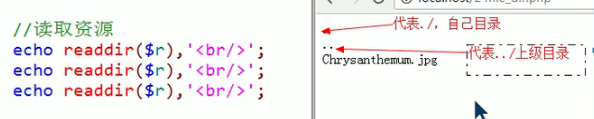
讀取目錄

讀取方式: 將文件夾(路徑)按照資源方式打開

opendir() 返回一個路徑目錄, 包含指定目錄下所有文件(不含文件夾)

$opendir = opendir('uploads');

readdir() 從資源中讀取指針所在位置的文件名字, 然後指針下移, 直到指針移出資源 (利用opendir路徑目錄)



*// 循環編歷輸出*

    while($readdir = readdir($opendir)){

        echo $readdir;

        echo '<br>';

    }

關閉目錄

closedir() 關閉資源

 closedir();

其他目錄操作

dirname(path) : 得到的是path上一層的路徑

realpath(path): 得到真實路徑, 路徑不存在時 ,返回false;

is\_dir(): 判斷路徑是不是目錄, 最終路徑為文件的不是目錄, 返回false

scandir(): 封裝版的opendir + readdir + closedir ,獲取一個指定路徑下的所有文件信息 , 以數組形式返回。

目錄操作

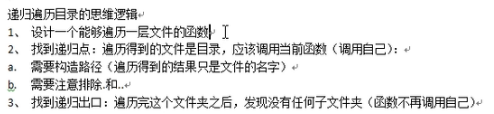
遞歸編歷目錄:

指定一個目錄的情況下 , 將其下所有文件和目錄 , 及其目錄內部的所有內部都輸出出來。

遞歸算法: 將大問題 切成相似的 小問題(最小單位), 之後可以調用解決大問題的方法來解決小問題。

遞歸函數: 函數如果自己內部調用自己 , 該函數稱之為遞歸函數。

遞歸編歷目錄的思維邏輯



dir(path);

**dir** ( string $directory , resource $context = ? ) : [Directory](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/class.directory.html)

成功的话，返回一个 [Directory](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/class.directory.html) 类实例, 参数错误的情况下返回 **null** ， 其它错误情况返回 **false。 同時回報錯誤信息。**

**根據層級來實現縮進: str\_repeat()**

**str\_repeat** ( string $input , int $multiplier ) : string

返回 input 重复 multiplier 次后的结果

常見文件操作

php讀寫文件

php5操作

file\_get\_contents(filepath) — 将整个文件读入一个字符串

file\_put\_contents(filepath , content) — 将一个字符串写入文件

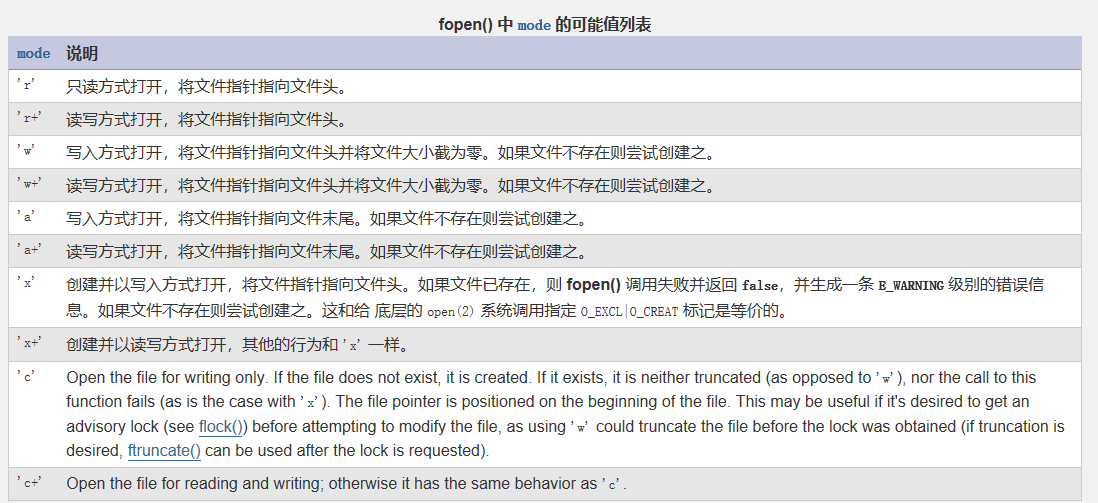
如果路徑下沒有文件, file\_put\_contents 會創建一個這個名字的文件,做寫入操作。

寫入操作是 把文件清空,再寫入。

php4操作

fopen(文件路徑,打開模式) — 打开文件或者 URL 返回一個resource(資源)

mode 打開模式



r模式 read文件

r+模式, 可read可write , 寫入是以a模式。

w模式 清空再寫入

a模式 不清空,指針指向文件尾, 再寫入

fread(資源,長度) — 读取文件（可安全用于二进制文件）

**fread** ( resource $handle , int $length ) : string

fwrite(資源, 內容 , [長度]) — 写入文件（可安全用于二进制文件）

fclose — 关闭一个已打开的文件指针

其他文件操作函數

is\_file() 判斷文件路徑是否正確(不識別路徑)

filesize() 獲取文件大小 bytes

file\_exists() 判斷文件是否在存(同時識別文件與路徑)

unlink() 取消連接(取消文件名字與磁盤地址的連接) (刪除文件)

filemtime() 獲取文件最後一次修改的時間

fseek() 設定fopen打開文件的指針位置 (聯合fopen(a模式)使用,可在中間插入文字)

fgetc() c是char 一次獲取一個字符

fgets($resource , [length]) 一次獲取一個字符串(默認一行)

file() 讀取整個文件, 類似file\_get\_contents, 區別是以 (行) 做單位, 返回一個數組。

文件操作

文件下載

從服務器將文件通過http協議傳輸到瀏覽器, 瀏覽器不解析保存成相應的文件。

可以使用HTML中的 a標簽: <a href=”URLpath”>下載</a>

但a標簽有二個缺點:

1. a標簽能夠讓瀏覽器自動下載的內容有限, 瀏覽器是發現如果解析不了才會啟動下載
2. a標簽的url地址會暴露我們服務器存儲數據的位置(不安全)

php下載: 讀取文件內容, 以文件流的形式傳遞給瀏覽器, 在響應頭中告知瀏覽器不要解析, 激活下載框實現下載。

1. 指定瀏覽器解析字符集
2. 設定響應頭

設定文件返回類型: image/jpg||Content-Type:application/octem-stream

設定返回文件計算方式: Accept-ranges: bytes

設定下載提示: Content-disposition: attachment:filename=’文件名字’

設定文件大小: Accept-length: 文件大小(字節)

1. 讀取文件

$f = @fopen($filepath ,'r') or die();

1. 輸出文件

while(!feof($f)){

        echo fread($f,1024);

    }

  fclose($f);

**iconv** ( string $in\_charset , string $out\_charset , string $str ) : string

将字符串 str 从 in\_charset 转换编码到 out\_charset。

feof() 測驗文件指針是否到了文件結束位置。 到了返回true

*// 下載響應頭*

    header('Content-Type:application/octem-stream'); *//以文件流形式傳輸數據給瀏覽器*

    header('Accept-ranges:bytes');                   *//以字節方式計算*

    header("Content-disposition: attachment;filename=" . $filename); *//下載,指定命名*

    header('Accept-length:' . filesize($filepath));  *//設定文件大小*

*// 如果文件名字是從文件夾讀取出來, 而且有中文字, 如果直接使用名字作為下載 名字會出現亂碼 使用iconv*

*// iconv( string $in\_charset, string $out\_charset, string $str) : string*

*// 将字符串 str 从 in\_charset 转换编码到 out\_charset。*

*// 輸出文件*

*// php5: 小文件*

*// echo file\_get\_contents($file);*

*// php4: 大文件*

    $f = @fopen($filepath ,'r') or die();

*// 方式1: 直接讀, 輸出*

*// while($row = fread($f,1024)){*

*//     echo $row;*

*// };*

*// 方式2: 判定是否可讀, 之後再讀*

    while(!feof($f)){

        echo fread($f,1024);

    }

    fclose($f);

die exit 函數

die() == exit()

exit — 输出一个消息并且退出当前脚本

中止脚本的执行。 尽管调用了 **exit()**， [Shutdown函数](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/function.register-shutdown-function.html) 以及 [object destructors](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/language.oop5.decon.html#language.oop5.decon.destructor) 总是会被执行。

exit 是个语法结构，如果没有 status 参数要传入，可以省略圆括号。

//检查文件是否可读

if(!is\_file($filePath) || !is\_readable($filePath)) exit('Can not access file '.$filename);

php實現下載2

<?php

//接收需要下载的文件名称

if(!isset($\_GET['file'])) exit('Filename is empty');

if(empty($\_GET['file'])) exit('Filename not valid');

ob\_clean();//清除一下缓冲区

//获得文件名称

$filename = basename(urldecode($\_GET['file']));

//文件完整路径（这里将真实的文件存放在temp目录下）

$filePath = \_\_DIR\_\_."/temp/".$filename;

//将utf8编码转换成gbk编码，否则，文件中文名称的文件无法打开

$filePath = iconv('UTF-8','gbk',$filePath);

//检查文件是否可读

if(!is\_file($filePath) || !is\_readable($filePath)) exit('Can not access file '.$filename);

// 这里应该加上安全验证之类的代码，例如：检测请求来源、验证UA标识等等

//以只读方式打开文件，并强制使用二进制模式

$fileHandle=fopen($filePath,"rb");

if($fileHandle===false){

exit("Can not open file: $filename");

}

//文件类型是二进制流。设置为utf8编码（支持中文文件名称）

header('Content-type:application/octet-stream; charset=utf-8');

header("Content-Transfer-Encoding: binary");

header("Accept-Ranges: bytes");

//文件大小

header("Content-Length: ".filesize($filePath));

//触发浏览器文件下载功能

header('Content-Disposition:attachment;filename="'.urlencode($filename).'"');

//循环读取文件内容，并输出

while(!feof($fileHandle)) {

//从文件指针 handle 读取最多 length 个字节（每次输出10k）

echo fread($fileHandle, 10240);

}

//关闭文件流

fclose($fileHandle);

urlencode()

urlencode — 编码 URL 字符串

**urlencode** ( string $str ) : string

此函数便于将字符串编码并将其用于 URL 的请求部分，同时它还便于将变量传递给下一页。

//获得文件名称

$filename = basename(urldecode($\_GET['file']));

**Warning** 超全局变量 [*$\_GET*](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/reserved.variables.get.html) 和 [*$\_REQUEST*](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/reserved.variables.request.html) 已经被解码了。对 [*$\_GET*](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/reserved.variables.get.html) 或 [*$\_REQUEST*](mk:@MSITStore:D:\PHP_8\manual\php_enhanced_zh.chm::/res/reserved.variables.request.html) 里的元素使用 **urldecode()** 将会导致不可预计和危险的结果。

urldecode — 解码已编码的 URL 字符串

**urldecode** ( string $str ) : string

解码给出的已编码字符串中的任何 %##。 加号（'+'）被解码成一个空格字符。

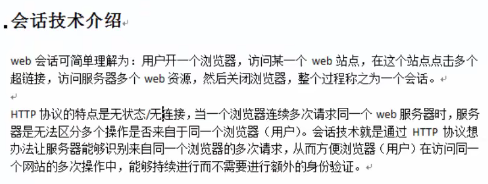
basename — 返回路径中的文件名部分

**basename** ( string $path , string $suffix = ? ) : string

给出一个包含有指向一个文件的全路径的字符串，本函数返回基本的文件名。

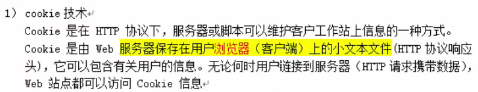
feof file End-Of-File

會話技術

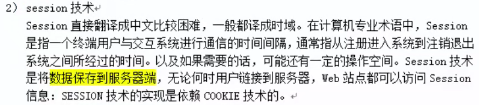


會話技術的分類

1. cookie技術



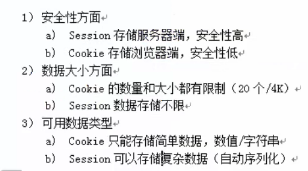
1. session技術



session的實現是依賴cookie的。

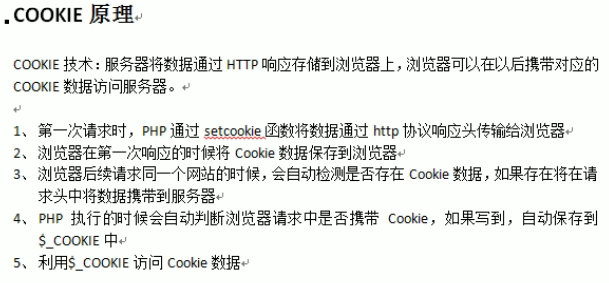
session與cookie的區別:

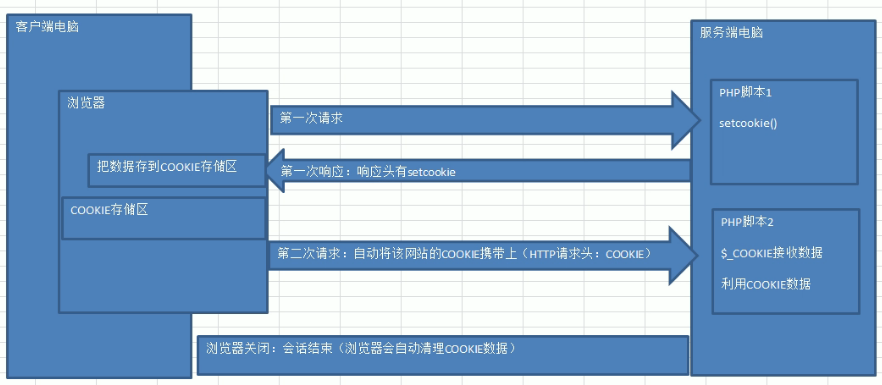
安全性, 數據大小, 可用數據類型, 保存位置





cookie原理:





cookie 基本使用

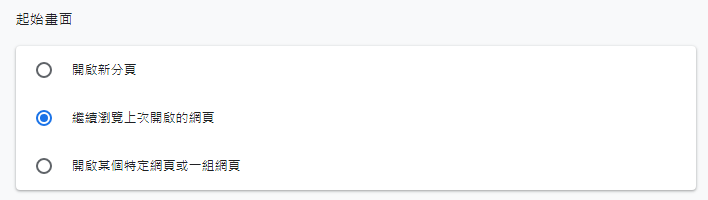
設置cookie信息

setcookie(名字,值) 函數用來設定cookie信息

setcookie 值的類型: 整數或字符串。

$\_COOKIE 讀取瀏覽器對應網站的cookie信息。

cookie高級使用



如果chrome 有以上設定, 可能會使 cookie在關閉瀏覽器後, 生命周期得不到結束。

cookie生命周期:

cookie在瀏覽器生存時間 (瀏覽器在下次訪問服務器的時候是否携帶對應的cookie)

1. 默認(不設定)時的生命周期: 不設定周期默認是關閉瀏覽器(會話結束)
2. 設定一個生命周期: setcookie的第3個參數可以限定生命周期, 是用時間戳。
3. **setcookie** ( string $name , string $value = "" , int $expires = 0 , string $path = "" , string $domain = "" , bool $secure = **false** , bool $httponly = **false** ) : bool

**第3個參數,默認為0。 就是普通生命周期,** 默認是關閉瀏覽器(會話結束),清除該cookie。

1. 刪除cookie。 服務器沒有權限去操作瀏覽器的cookie(無法刪除), 只能利用設定生命周期,讓瀏覽器自動判定cookie是否有效, 無效就清除。

4.1 cookie對應的名字給予空value 就是無效cookie。

*// 刪除cookie*

    setcookie('a4','');

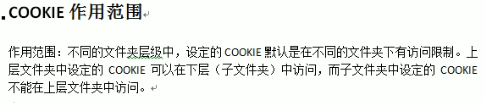
* 1. 設定無效的生命周期給對應的cookie名, 使它無效。

setcookie('a4','a4',1);

1970年的1秒後, 就清空該cookie。

cookie 高級

cookie 作用範圍



1. 默認(不設定)的範圍: 就是使用cookie默認的作用範圍。
2. 上層(father) 看不到 下層(son) 的cookie。
3. 所以我們設定 “/” 含意:

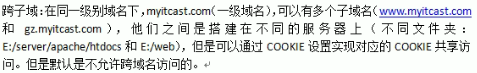
告知瀏覽器當前cookie作用範圍是網站根目錄。

setcookie(名字,值,生命周期,作用範圍);

 setcookie('global\_son','global\_son\_value',0,'/');

cookie 高級

cookie 跨子域



默認是不允許跨域名訪問的。

1. 設定cookie的有效域名: 不同的域名(包含主機)之間不能共享cookie
2. 不設定時的默認有效域名

setcookie() 第5個參數: 有效域名

1. 跨子域設定方法: 在設定域名訪問的時候用設定上級域名即可:

myitcast.com 這個是有所有以myitcast.com結尾的網站都可以共享cookie。

(同一個域名下的 不同的 子域名)。

*// 指定域名cookie*

    setcookie('local\_area','local\_area',0,'/','www.myitcast.com');

*// 設定一個允許子域名訪問的cookie*

    setcookie('local\_area','local\_area',0,'/','myitcast.com');

cookie 數組數據

cookie的值本身只支持(數字 / 字符串)。能夠保留的數據本身也有限。

我們想方法實現 cookie 保存數組數據。

1. 設置形式: setcookie(‘C1[k1]’, value); 在cookie名中, 給個下標索引k。
2. 實現偽裝數組
3. 讀取形式: $\_COOKIE[‘c1’][‘k1’];

*// 偽裝數組*

    setcookie('goods\_id[0]',001);

    setcookie('goods\_id[1]',002);

    setcookie('goods\_id[2]',003);

    setcookie('goods\_id[3]',004);

*// 讀取偽裝數組cookie*

    echo '<pre>';

    var\_dump($\_COOKIE);

    echo '<hr>';

    echo $\_COOKIE['goods\_id'][0];

    echo $\_COOKIE['goods\_id'][1];

    echo $\_COOKIE['goods\_id'][2];

    echo $\_COOKIE['goods\_id'][3];

會話技術:

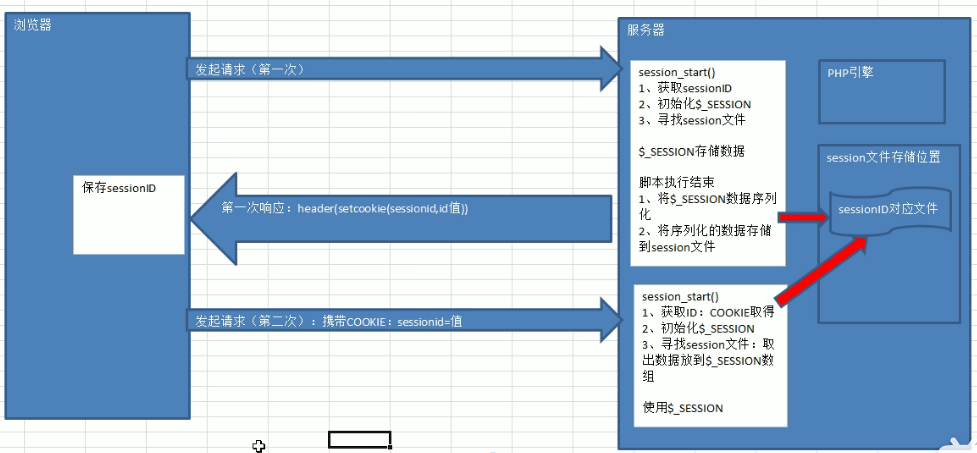
session技術

session原理:

session與瀏覽器無關, 但與cookie有關。

sessionID是 setcookie(名字,值)中的名字

1. php執行到session\_start()時開啟session會話, 會自動檢測sessionID
2. 如果cookie中存在, 使用現有的。
3. 如果cookie中不存在, 創建一個sessionID , 并通過響應頭以cookie形式保存到瀏覽器上。
4. 初始化超全局變量$\_SESSION為一個空數組。
5. php通過sessionID去指定位置(session文件存儲位置)匹配對應的文件。
6. 不存在該文件, 創建一個sessionID命名文件。(創建後就存在)
7. 存在該文件,讀取文件內容(反序列化), 將數據存儲到$\_SESSION中。
8. session\_start()腳本執行結束
9. 將$\_SESSION中保存的所有數據序列化存儲到sessionID對應的文件中。



session基本使用

啟用session, 任何時候都需要開啟session (腳本使用到$\_SESSION就開啟一次)

$\_SESSION 是通過session\_start()函數的調用才會定義的, 沒有直接定義。

session\_start();

設置session信息

如果想存儲數據到session中, 只要直接給$\_SESSION數組添加元素即可。

 $\_SESSION['name'] = 'JAck';

 $\_SESSION['hobby'] = array('running','fighting');

讀取session信息

$\_SESSION 是個數組, 就以數組形式訪問就可以讀取了。

*// 讀取session信息*

    echo '<pre>';

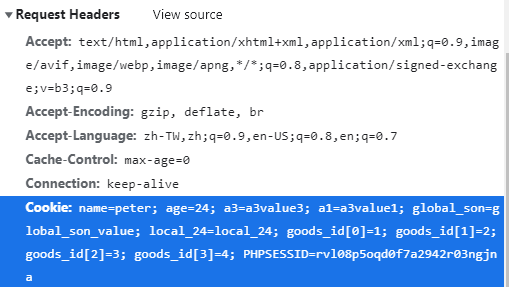
    echo $\_SESSION['name'];

    echo '<br>';

    var\_dump($\_SESSION['hobby']);

會話技術本質是為了實現 跨腳本共享數據:

在一個腳本中定義數據, 在另一個腳本中保存數據。



request header中 有一個PHPSESSID 就是

去請求cookie中的$\_SESSION內的sessionID內對應的數據。

session\_start() 🡪 $\_SESSION[‘sessionID’ , ‘value’] 🡪 瀏覽器session文件內

就會有[‘sessionID’ , ‘value’]的數據。

刪除session數據

刪除一個SESSION信息

unset($\_SESSION[元素索引]); 刪除指定session數據。

unset($\_SESSION['name']);

刪除全部session信息

$\_SESSION = array();

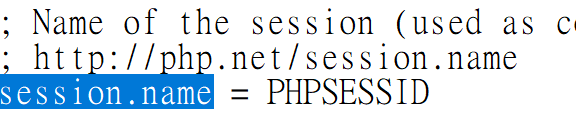
使$\_SESSION變成空數組,就是清空session數據了。

session 相關配置

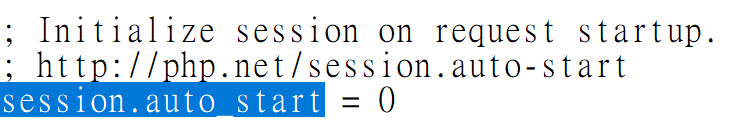
session 基礎配置

可在php.ini中查看

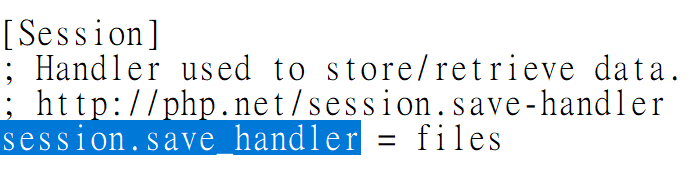
1. session.name: session名字 , 保存到cookie中sessionID對應的名字



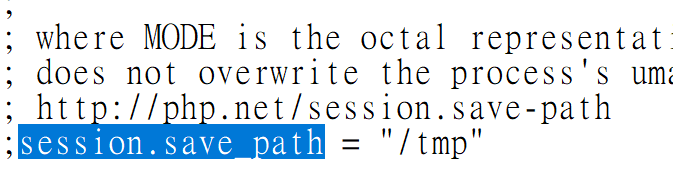
1. session.auto\_start: 是否自動開啟 session (如果開啟後無需手動session\_start(), 默認是關閉的。)



1. session.save\_handler: session數據的保存方式, 默認是文件形式

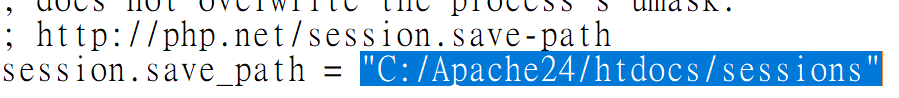


1. session.save\_path: session 文件默認存儲的位置



session.save\_path 默認是沒有開啟的。 說明借用操作系統的臨時文件存儲。

使用系統的臨時文件存儲不安全, 我們自己指定存儲路徑。

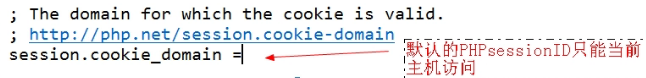


session 常用配置 php.ini

session.cookie\_lifetime: PHPsessionID 在瀏覽器對應cookie的生命周期, 默認是會話結束 (自動開啟的 session.cookie\_lifetime = 0;) 0就是默認是會話結束

session.cookie\_path: sessionID在瀏覽器存儲之後允許服務器訪問的路徑(cookie有作用範圍) 默認是session.cookie\_path = “/” 網站根目錄

session.cookie\_domain: cookie允許訪問的子域 (cookie可以跨子域)



配置的2種方式

1. php.ini中配置: 全局配置, 修改php.ini中的配置。
2. 腳本中配置: php可以用 ini\_set() 函數來在運行中設定某些配置項(只會對當前運行的腳本有效), 叫做項目級配置。

ini\_set(‘session.save\_path’,’path/path/…’);

銷毀session

之前刪除session是指刪除session數據, $\_SESSION中看不到而已。

銷毀session是刪除session對應的文件。

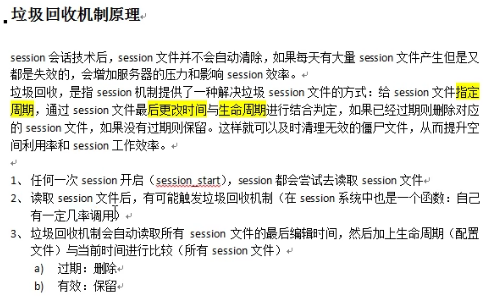
銷毀session函數:

session\_destroy(); 會自動根據 session\_start得到的sessionID去找到指定的session文件, 并把其刪除。

sleep(5); php休息5秒再執行

session 垃圾回收

垃圾回收機制原理



垃圾回收參數設置: php.ini中

session.gc\_maxlifetime = 1440

規定session文件最大的生命周期是1440秒,24分鐘。

session.gc\_probability = 1

垃圾回收函數執行的機率(分子)

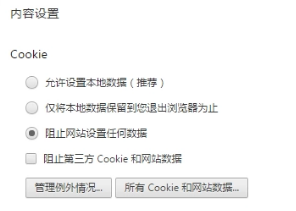
session.gc\_divisor = 1000

垃圾回收函數執行的機率(分母)

所以現在是 1/1000 的機率。

禁用cookie與如何使用session





如果禁用了cookie。

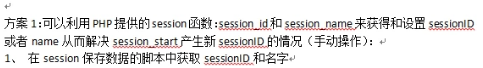
瀏覽器讀取不了session文件, 就會每次都開一個新的session文件。

解決思路: 最終讓session\_start在開啟之前拿到原來的sessionID(另外一個腳本的)

實現禁用cookie下使用session

有2種方法:

session\_id 與 session\_name



$id = session\_id();

    $name = session\_name();

    echo $name . '=' . $id;



echo "<a href='31-session\_nocookie2.php?{$name}={$id}'>click</a>";



*// 接收數據*

    $name = session\_name();

    $id = $\_GET[$name];

4. 阻止session\_start 產生新id (由於禁用cookie,無法讀取,所以會開新id與新session文件), 告訴它已經存在: session\_id($id);

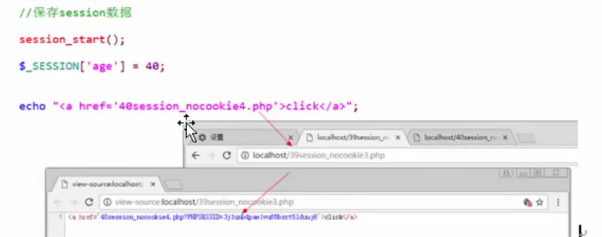
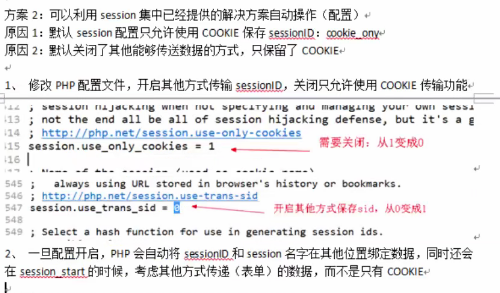
session\_id($id);

方案2:

php.ini 配置

session.use\_only\_cookies = 0

session.usse\_trans.sid = 1



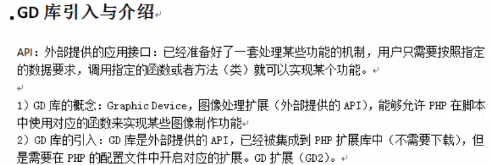


我們的php session 沒有保存到session Storage中, 而是保存到服務器中(通過cookie技術)。

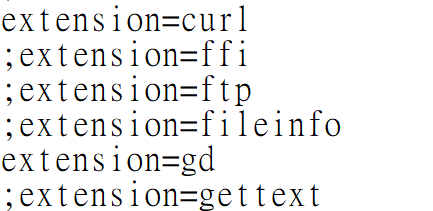
GD圖像處理技術

graphic device

GD庫引入與介紹

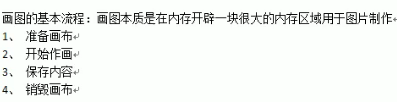


php8 是命名為gd。



到php.ini 開放gd庫的擴展。

畫圖基本流程



創建畫布資源 (imageCreate返回一個$resource)

1. imageCreate(x\_size , y\_size) 創建一個空白畫布 (background color白色)
2. imageCreateTrueColor(x\_size , y\_size) 創建一個真彩畫布(background color黑色, 需要填充)
3. imageCreateFromJpeg(path) 打開一個jpeg格式的圖片資源
4. imageCreateFromGif(path) 打開一個gif格式的圖片資源 (php中無法實現動態)
5. imageCreateFromPng(path) 打開一個png格式的圖片資源

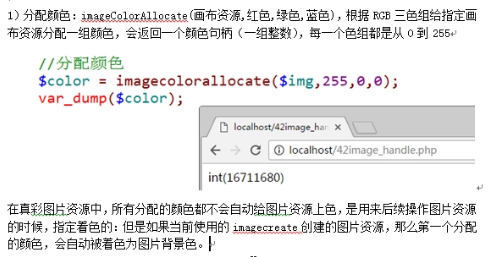
JPEG和JPG之間沒有任何區別。 JPG圖片格式 之所以存在，是因為早期的Windows版本需要3個字母的文件擴展名。 因此，.jpeg擴展名被縮短並變成了.jpg。

操作畫布資源

說明: 所有的畫布資源操作都是需要指定畫布資源(第一個參數)

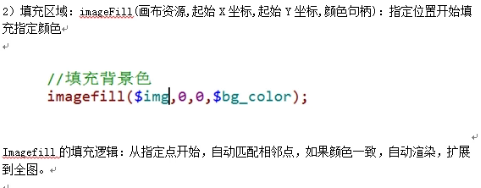
1. imageColorAllocate($resource , 255 ,255 ,255) 分配顏色 RGB三色給值

返回一個(顏色整數 $colorint)。



2. imageFill($resource,Xstart,Ystart,$colorint) 填充區域





3. 畫直線: imageLine($resource,Xstart,Ystart,Xend,Yend,$colorint)

4. 畫矩形: imageRectangle($resource,左上X,左上Y,右下X,右下Y,$colorint);

5. imagearc — 画椭圆弧

**imagearc** ( resource $image , int $cx , int $cy , int $w , int $h , int $s , int $e , int $color ) : bool

imagearc($resource , 軸點x, 軸點y, width , height, 弧度起點, 弧度終點,$color)

**imagearc()** 以 cx，cy（图像左上角为 0, 0）为中心在 image 所代表的图像中画一个椭圆弧。w 和 h 分别指定了椭圆的宽度和高度，起始和结束点以 s 和 e 参数以角度指定。0°位于三点钟位置，以顺时针方向绘画。

1. 畫布上寫字

imageString() 水平地画一行字符串

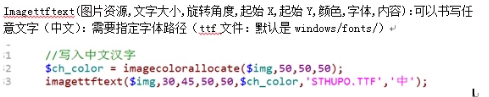
**imagestring** ( resource $image , int $font , int $x , int $y , string $s , int $col ) : bool

**imagestring()** 用 col 颜色将字符串 s 画到 image 所代表的图像的 x，y 坐标处（这是字符串左上角坐标，整幅图像的左上角为 0，0）。如果 font 是 1，2，3，4 或 5，则使用内置字体。

imageTTFtext() 用 TrueType 字体向图像写入文本

**imagettftext** ( resource $image , float $size , float $angle , int $x , int $y , int $color , string $fontfile , string $text ) : array

使用 TrueType 字体将 指定的 text 写入图像。



我們把window的fonts 複製到 htdocs/fonts內

輸出畫布資源

1. 輸出為圖片文件: 以圖片文件形式保存到本地文件夾
2. 輸出為網頁圖片: 給html(用戶)

imagejpeg() 保存成jpeg格式

imagepng()保存成png格式

imagegif()保存成gif格式

如果圖片只是提供了圖片資源, 不指定保存位置, 就會輸出給瀏覽器。

如果指定了位置, 就會保存到本地。(第二參數)

輸出到瀏覽器要加header() 因為這個圖片是binary的,要轉編碼。

*// 保存到本地*

    imagepng($img,'GD\_img/my.png');

*// 輸出到瀏覽器*

    header('Content-Type:image/png');

    imagepng($img);

\*\*\*\*\*\*

header前不能有任何輸出 比如echo var\_dump, 不然會影響到header, 無法正常執行。





imagecopyresized — 拷贝部分图像并调整大小

銷毀畫布資源 imagedestroy($resource);

*// 銷毀畫布資源*

    imagedestroy($img);

由內存中清理畫布資源 , 釋放內存。

獲取圖片信息

imagesx()獲取畫布x尺寸

imagesy() 獲取畫布y尺寸

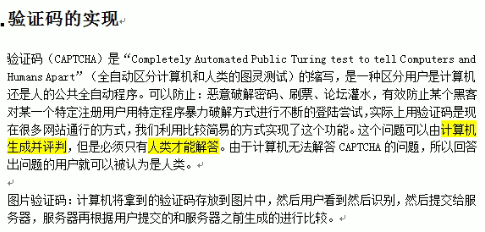
getimageSize() 獲取圖片尺寸

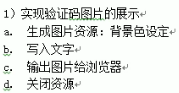
getimageSize 還能獲取圖片的格式,mime類型等…

GD圖片應用案例

驗證碼的實現

驗證碼 (captcha)





驗證碼如果用imagestring() 效果不好。

我們用imageTTFtext()

獲取random中文字

   $rand = mt\_rand(0,$c\_len - 1);

    $char = substr($str , $rand\* 3, 3);

**substr** ( string $string , int $start , int $length = ? ) : string

返回字符串 string 由 start 和 length 参数指定的子字符串。

實現驗證碼背景的干擾點(點,直線等等…)

imagesetpixel($resource,x,y,$color); 點生成 (但不能定大小)

我們用imagestring() 生成點(可以設置大小)

*// 干擾點 imagestring()*

    for($i=0;$i<90;$i++){

        $dot\_color = imagecolorallocate($img,mt\_rand(100,200),mt\_rand(100,200),mt\_rand(100,200));

        imagestring($img,mt\_rand(1,5),mt\_rand(0,200),mt\_rand(0,50),'\*',$dot\_color);

    };

實現click刷新驗證碼: 在瀏覽器中。

縮略圖,水印圖