

Funzioni HTTP

Introduzione

Queste funzioni permettono di modificare l'output inviato verso un browser attraverso manipolazioni a livello di protocollo HTTP.

Requisiti

Non sono necessarie librerie esterne per utilizzare questo modulo.

Installazione

Non è necessaria nessuna installazione per usare queste funzioni, esse fanno parte del core di PHP.

Configurazione di Runtime

Questa estensione non definisce alcuna direttiva di configurazione in `php.ini`

Tipi di risorse

Questa estensione non definisce alcun tipo di risorsa.

Costanti predefinite

Questa estensione non definisce alcuna costante.

Sommario

[header](#) -- Spedisce un header HTTP

[headers_list](#) -- Returns a list of response headers sent (or ready to send)

[headers_sent](#) -- Restituisce TRUE se gli header sono stati trasmessi.

[setcookie](#) -- Spedisce un cookie

[setrawcookie](#) -- Send a cookie without urlencoding the cookie value

header

(PHP 3, PHP 4, PHP 5)

header -- Spedisce un header HTTP

Descrizione

int **header** (string string [, bool replace])

header() si utilizza per inviare header HTTP. Per maggiori informazioni riguardanti gli header HTTP si veda la risorsa [HTTP 1.1 specification](#).

L'argomento opzionale *replace* indica se l'header inviato deve sostituirne uno simile spedito precedentemente, o accordarsi al primo dello stesso tipo. Per default la funzione sostituisce l'header precedente, ma se viene passato **FALSE** come secondo argomento vengono forzate intestazioni multiple. Per esempio:

```
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM', false);
```

Ci sono due casi speciali di chiamate di header. Il primo è "Location". Location non trasmette solo un header al browser, ma anche un *REDIRECT* con codice di stato (302).

```
header("Location: http://www.php.net/"); /* Ridireziona il browser
                                         al sito di PHP */
exit;                                   /* Assicura che il codice sottostante
                                         non sia eseguito dopo il redirezionamento. */
```

Nota: HTTP/1.1 richiede un URI assoluto come argomento di [Location](#): composto da schema, hostname, e path assoluto, ma alcuni clients possono accettare anche URI relativi. E' possibile usare `$HTTP_SERVER_VARS['HTTP_HOST']`, `$HTTP_SERVER_VARS['PHP_SELF']` e [dirname\(\)](#) per creare URI assoluti da URI relativi in modo automatico:

```
header ("Location: http://".$HTTP_SERVER_VARS['HTTP_HOST']
        ."/".dirname($HTTP_SERVER_VARS['PHP_SELF'])
        ."/". $relative_url);
```

Il secondo caso speciale è esemplificato dalle intestazioni che iniziano con la stringa, "HTTP/" (le maiuscole non sono discriminanti), che è usato per inviare codici di stato HTTP. Per esempio, se si è configurato Apache per usare script PHP per la manipolazione di richieste fallite (usando la direttiva *ErrorDocument*), potete desiderare di assicurarvi che il vostro script generi il codice adeguato.

```
header ("HTTP/1.0 404 Not Found");
```

Nota: In PHP 3, questo funziona solo se PHP è compilato come modulo Apache. Potete ottenere lo stesso effetto usando l'header *Status*.

```
header("Status: 404 Not Found");
```

Spesso gli scritti PHP generano contenuti dinamici, se volete evitare che i contenuti vengano mantenuti nella cache di browser o proxy, potete forzare il loro comportamento con questa direttiva:

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // Data passata
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
// sempre modificato
header("Cache-Control: no-store, no-cache, must-revalidate"); // HTTP/1.1
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache"); // HTTP/1.0
```

Nota: E' possibile che alcune pagine rimangano in cache anche dopo l'uso degli header descritti sopra. Ci sono delle opzioni che l'utente può settare dal browser, capaci di modificare i comportamenti di default del caching. Per trasmettere efficacemente gli header descritti, bisogna che sia inattiva ogni regolazione che può forzare comportamenti contrari.

Inoltre, [session cache limiter\(\)](#) e la configurazione *session.cache_limiter* possono essere usate per generare automaticamente i corretti header relativi al caching durante l'uso delle sessioni.

Bisogna ricordare che la funzione **header()** va chiamata prima di qualsiasi output HTML o PHP (anche righe o spazi vuoti). E' un errore comune leggere files con funzioni [include\(\)](#), o [require\(\)](#) (o altre funzioni capaci di accedere a files), che possano emettere in output spazi o linee vuote prima di una chiamata della funzione **header()**. Lo stesso problema esiste nell'utilizzare file PHP/HTML.

```
<?php require("user_logging.inc") ?>

<?php header ("Content-type: audio/x-pn-realaudio"); ?>
// Non funziona, notate le linee vuote sopra
```

Nota: In PHP 4, potete usare il buffering dell'output per aggirare questo problema, evitando ogni output al browser trattenendolo al server fino a che non gli si impone l'invio. Si può fare questa operazione chiamando [ob_start\(\)](#) e [ob_end_flush\(\)](#) nello script, o settando la direttiva di configurazione *output_buffering* nel file *php.ini* o nel file di configurazione del server.

Se desiderate che l'utente sia spinto a salvare i dati trasmessi per esempio utilizzando un file PDF, potete usare l'header [Content-Disposition](#), che vi permette di dare un nome al file e forzare il browser a mostrare la finestra di dialogo save.

```
<?php
header("Content-type: application/pdf");
header("Content-Disposition: attachment; filename=downloaded.pdf");
```

```
/* ... manda in output un file pdf ... */
```

Nota: Per un bug di Microsoft Internet Explorer 4.01 questo sistema non funziona. Non ci sono soluzioni. C'è un altro bug in Microsoft Internet Explorer 5.5 che impedisce il giusto funzionamento, ma è possibile risolverlo con l'upgrade del Service Pack 2 o superiore.

Vedi anche [headers_sent\(\)](#), [setcookie\(\)](#), e la sezione [Autenticazione HTTP usando PHP](#).

headers_list

(PHP 5)

headers_list -- Returns a list of response headers sent (or ready to send)

Description

array headers_list (void)

headers_list() will return a numerically indexed array of headers to be sent to the browser / client. To determine whether or not these headers have been sent yet, use [headers_sent\(\)](#).

Esempio 1. Examples using headers_list()

```
<?php

/* setcookie() will add a response header on its own */
setcookie('foo', 'bar');

/* Define a custom response header
   This will be ignored by most clients */
header("X-Sample-Test: foo");

/* Specify plain text content in our response */
header('Content-type: text/plain');

/* What headers are going to be sent? */
var_dump(headers_list());

?>
```

this will output :

```
array(4) {
  [0]=>
  string(12) "X-Powered-By"
  [1]=>
  string(10) "Set-Cookie"
  [2]=>
  string(13) "X-Sample-Test "
```

```
[3]=>
string(12) "Content-type"
}
```

See Also: [headers_sent\(\)](#), [header\(\)](#), and [setcookie\(\)](#).

headers_sent

(PHP 3 >= 3.0.8, PHP 4, PHP 5)

`headers_sent` -- Restituisce TRUE se gli header sono stati trasmessi.

Descrizione

bool **headers_sent** (void)

Questa funzione restituisce TRUE se gli header HTTP sono stati spedite correttamente, FALSE in caso contrario.

Vedi anche [header\(\)](#)

setcookie

(PHP 3, PHP 4, PHP 5)

`setcookie` -- Spedisce un cookie

Descrizione

int **setcookie** (string name [, string value [, int expire [, string path [, string domain [, int secure]]]])

setcookie() definisce un cookie da inviare insieme alle altre informazioni di header. I cookie devono essere spediti *prima* di qualsiasi altra intestazione (questa è una restrizione dei cookies, non di PHP). E' necessario perciò chiamare la funzione **setcookie()** *prima* di qualsiasi tags, anche `<html>` o `<head>`.

Tutti gli argomenti della funzione eccetto *name* sono opzionali. Se viene passato alla funzione solo l'argomento *name*, il cookie registrato con quel nome verrà cancellato dal client su cui è archiviato. E' possibile sostituire gli argomenti che non si intende specificare utilizzando una stringa vuota (`""`). Gli argomenti *expire* e *secure* che richiedono numeri interi, non possono essere omessi inserendo una stringa vuota, per questo scopo si usa (`0`). L'argomento *expire* è un normale intero Unix Timestamp ottenibile grazie alle funzioni [time\(\)](#) o [mktime\(\)](#). *secure* sta ad indicare che il

cookie dovrebbe essere trasmesso soltanto attraverso un collegamento sicuro di tipo HTTPS.

Errori comuni:

- I cookie diventano disponibili soltanto dalla pagina successiva a quella che li ha generati, o dopo il ricaricamento di questa.
- I cookie devono essere cancellati specificando gli stessi parametri con cui sono stati creati.

In PHP 3, chiamate successive di **setcookie()** nello stesso script sono eseguite in ordine inverso. Se state provando a cancellare un cookie prima dell' inserimento di un altro cookie, dovete creare il secondo prima della cancellazione del primo. In PHP 4, chiamate successive di **setcookie()** invece, sono eseguite secondo l'ordine di chiamata.

Alcuni esempi sul modo di spedire cookie:

Esempio 1. setcookie() esempi di spedizione/creazione

```
setcookie ("TestCookie", "Test Value");
setcookie ("TestCookie", $value,time()+3600); /* aspira in 1 ora */
setcookie ("TestCookie", $value,time()+3600, "/~rasmus/", ".utoronto.ca", 1);
```

Gli esempi mostrano come cancellare i cookie introdotti nell'esempio precedente:

Esempio 2. setcookie() esempi di cancellazione

```
setcookie ("TestCookie");
// set the expiration date to one hour ago
setcookie ("TestCookie", "", time() - 3600);
setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".utoronto.ca", 1);
```

Per cancellare un cookie dovete assicurarvi che la data di scadenza del cookie sia già trascorsa, in questo modo il cookie verrà rimosso dal client.

Si noti che i valori salvati nei cookies sono automaticamente codificati per la trasmissione via URL (urlencoded) quando il cookie viene inviato, e che al momento del richiamo sono automaticamente decodificati e assegnati ad una variabile che ha lo stesso nome del cookie. Per vedere il contenuto di un cookie in uno script, si usa una di queste due sintassi:

```
echo $TestCookie;
echo $_HTTP_COOKIE_VARS["TestCookie"];
```

Potete registrare array in un cookie usando la notazione degli array al posto del nome del cookie. Questo equivale alla spedizione di tanti cookie quanti sono gli elementi dell'array, ma si ha un vantaggio: quando il cookie è ricevuto, tutti i suoi valori sono ordinati in un singolo array che ha per nome il nome del cookie:

```
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");
if (isset ($cookie)) {
    while (list ($name, $value) = each ($cookie)) {
        echo "$name == $value<br>\n";
    }
}
```

```
}  
}
```

Per saperne di più sui cookies, Netscape's cookie specification è la risorsa giusta http://wp.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 con Service Pack 1 non crea correttamente cookie che hanno il parametro *path* specificato.

Netscape Communicator 4.05 e Microsoft Internet Explorer 3.x sembrano utilizzare in modo errato i cookie quando *path* ed *expire* non sono specificati.

setrawcookie

(PHP 5)

setrawcookie -- Send a cookie without urlencoding the cookie value

Description

bool **setrawcookie** (string name [, string value [, int expire [, string path [, string domain [, bool secure]]]])

setrawcookie() is exactly the same as [setcookie\(\)](#) except that the cookie value will not be automatically urlencoded when sent to the browser.

See also [header\(\)](#), [setcookie\(\)](#) and the [cookies section](#).