

# HTTP Headers

# Introduction

- ❖ When a web browser receives a web page it sends an HTTP request to the web server.

## Typical HTTP Request

```
GET / index.html HTTP/1.1
```

```
Host: www.jacado.com
```

```
Accept-Encoding: UTF-8
```

```
User-Agent: Firefox/1.0
```

# Introduction

- ❖ When the web server receives the browser's request, it processes the request and sends the HTTP response.

## Typical HTTP Response

```
HTTP/1.1 200 OK
Server: Apache
Content-Type: text/html
Content-Encoding: UTF-8
Content-Length: 232332
[Data]
```

# Introduction

- ❖ In addition to the core returned data (e.g. a requested HTML document), the server reply includes a set of response headers that contain information about the core returned data.
- ❖ An HTTP header is a simple string in the form of key:value.
- ❖ Each one of the headers is sent in a separated line.
- ❖ New line characters separate each one of the headers from the rest of the headers. A new line character separates the headers from the core returned data.

# Introduction

- ❖ The PHP engine and the web server automatically take care of sending out the HTTP headers.

# The header () Function

- ❖ The header() function allows sending back a raw of an HTTP header.

```
void header(String $string [,bool $replace  
                [,int $http_response_code]]
```

Usually, the \$string can be one of the following three options:

## The Content-type Option

We will use this option to set the mime type of the file our script sends back to the web browser.

```
<?PHP  
header("Content-type: text/html");  
?>
```

# The header () Function

## The “Location:” Option

We will use this option to send back a specific URL address we want to redirect the user to. When using this option it is highly important to ensure that there won't be any execution of any additional PHP code.

```
<?PHP  
header("Location: http://www.jacado.com");  
?>
```



## The “HTTP/” Option

We will use this option to set a specific HTTP status code we want to return.

```
<?PHP  
header("HTTP/1.0 404 Not Found");  
?>
```

# The `header()` Function

The `$replace` optional parameter indicates whether the new header we set should replace a previous header that was already set... or add a second header of the same type (with the same name). The default value of this parameter is “true”.

The `$http_response_code` optional parameter allows us forcing a specific status code value.

The `header()` function must be called before any other output... including any white spaces characters outside of the PHP tags. Avoiding this rule might result in a PHP error or a failure to complete our `header()` call successfully.



# The header () Function

```
<?php
header("Content-Type: text/xhtmll");
header("Expires: Mon, 22 Jul 2047 05:00:00 GMT");
echo "If you want to see the headers try to browse this
page via one of the following web based sniffers:"
echo "<BR>";
echo "http://webtools.mozilla.org/web-sniffer/";
echo "<BR>";
echo "http://www.delorie.com/web/headers.html";
echo "<BR>";
echo "http://www.rexswain.com/httpview.html";
?>
```

# HTTP Compression

- ❖ The HTTP protocol supports the possibility to send data in a compressed form (using the gzip algorithm).
- ❖ The compression level (1..9) is configurable. The default level is 6.
- ❖ When a compressed data is sent back from the server, the Content-Encoding header is returned with the “gzip” value.

# HTTP Compression

- ❖ The the following is a typical server reply that includes compressed data:

```
HTTP/1.1 200 OK
Server: Apache
Content-Type: text/html
Content-Encoding: gzip
Content-Length: 26395

[GZIP COMPRESSED DATA]
```

# The `ob_start()` Function

- ❖ Calling the `ob_start()` function will turn on the output buffering.

```
bool ob_start ([ callback $output_callback  
               [, int $chunk_size [, bool $erase ]]])
```

Each one of the three parameters is optional. The `$output_callback` parameter allows us setting the name of another function that will receive one string parameter and return a string. That other function will be called before the data is sent back to the client. The reply of that other function will be the one that eventually the client will get.

# The `ob_start()` Function

Passing a value to the `$chunk_size` parameter will cause the buffer to be flushed after any output call that causes its buffer's length to be equal or to exceed the `$chunk_size` value. Passing `$chunk_size` the value 0 will cause the buffer to be sent back to the user only in the end. Passing 1 will set `$chunk_size` to be 4096.

Passing the 'false' value to the 'erase' optional parameter ensures the buffer will not be deleted until the script ends.

The `ob_start()` function returns 'true' if it succeeded in its work, and 'false' if it fails. (e.g. couldn't find the `callable_function`).

# The `ob_gzhandler()` Function

- ❖ The `ob_gzhandler()` is the `ob_start()` callback function we should use to gzip the output buffer.
- ❖ Calling the `ob_start("ob_gzhandler")` will turn on the output buffering and cause the script's output to be compressed before it returns back to the client.

```
string ob_gzhandler ( string $buffer , int $mode )
```

If according to the client request headers the PHP engine understands that the client browser is not capable of receiving compressed data this method returns false.

# The `ob_gzhandler()` Function

```
<?php
ob_start("ob_gzhandler");
?>
<html>
<body>
<p>
If your browser supports receiving compressed data
then this page was received in a compressed format.</p>
</html>
<body>
```

# The `ob_gzhandler()` Function

- ❖ We can use `ob_gzhandler()` separately without using the `ob_start()` function.
- ❖ The `ob_gzhandler()` receives a string and returns its compressed version in accordance with the client support for getting compressed data.



# The ob\_gzhandler () Function

```
<?php
$str = "a                                     bba";
echo "strlen(str)=" . strlen($str);
echo "<br/>";
$temp = ob_gzhandler($str, 9);
echo "strlen(temp) " . strlen($temp);
?>
```



# The `ob_gzhandler()` Function

```
localhost:8888/samples/zipdemo.php  
strlen(str)=52  
strlen(temp)=27
```

The screenshot shows the Network tab of a web browser's developer tools. The filter is set to 'All'. A request for 'zipdemo.php' is selected, and the 'Headers' sub-tab is active. The response headers are expanded, showing the following information:

- General:**
  - Connection: Keep-Alive
  - Content-Length: 34
  - Content-Type: text/html; charset=UTF-8
  - Date: Thu, 02 Jul 2015 07:52:16 GMT
  - Keep-Alive: timeout=5, max=100
  - Server: Apache/2.2.29 (Unix) mod\_fastcgi/2
  - X-Powered-By: PHP/5.6.2
- Request Headers:**
  - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
  - Accept-Encoding: gzip, deflate, sdch
  - Accept-Language: en-US,en;q=0.8,he;q=0.6
  - Cache-Control: max-age=0
  - Connection: keep-alive
  - Cookie: SQLiteManager\_currentLangue=2
  - Host: localhost:8888
  - User-Agent: Mozilla/5.0 (Macintosh; Intel

# The `php.ini` Compression Setting

- ❖ The `zlib` extension offers the option to transparently compress your pages on-the-fly, if the requesting browser supports this.
- ❖ We can change the compression behavior of the `zlib` extension through the `php.ini` setting file.

# Browser's Caching

- ❖ Most browsers cache as much of the content they download as possible. This way the user receives the requested content in a faster way. The user experience improves.
- ❖ Using the “Cache-Control” and the “Expires” headers it is possible to instruct the browser how to cache the output our script returns.

```
header("Cache-Control: no-cache, must-revalidate");  
header("Expires: Mon, 31 Jan 2008 02:55:22 GMT");
```

# Browser's Cookies

- ❖ A browser cookie is a small amount of textual data the browser has received from a specific web server.
- ❖ Each cookie has a name and a value. In addition, it includes various meta data.
- ❖ Setting a cookie within a specific web browser is done using specific HTTP header, the web server includes in its reply.

# Browser's Cookies

- ❖ Each time a web browser sends an HTTP request to a web server (the one that has set up the cookie/s) and assuming that cookie is still alive, the browser will include that cookie (cookies) in its request.

# The `setcookie()` Function

- ❖ The `setcookie()` function allows us setting a cookie on the client's browser.

```
bool setcookie ( string $name [, string $value [, int $expire  
                [, string $path [, string $domain  
                [, bool $secure [, bool $httponly ]]]]] )
```

As with other headers related functions it is a **MUST** to ensure that this function is called before any output (including HTML tags and white spaces) is written back from our script. All parameters (except for `$name`) are optional.

# The `setcookie()` Function

```
setcookie("id","1000232");
```

This code sets in client's browser a cookie with the name "id" and the value "1000232". That cookie will live as long as the session is alive. Once the session ends, that cookie will be deleted. In order to have a cookie that continue to live after the session ends and persists between sessions we need to pass an expiration date in our call to the `setcookie` function.

```
setcookie("id","1000232",time()+86400*3);
```

This code will instruct the browser to try and keep the cookie for 3 days.



# The `setcookie()` Function

The `$path` argument allows us setting a path on our website (relative to our website's root director). When setting that path, the cookie will be accessible in that path only. The browser will send a cookie to pages within that path only.

The `$secure` boolean parameter allows us ensuring the browser will send the new created cookie when communicating via HTTPS only.

# Accessing Cookies

- ❖ When a web browser sends a request to a web server, the PHP engine automatically separates the cookies from the HTTP headers and places them within the `$_COOKIE` super global array.

```
$id_val = $_COOKIE['id'];
```

This code returns puts the value of the 'id' cookie within the `id_val` variable.

# Cookie Array Values

- ❖ Though the cookie value must be a scalar value (one value only), it is possible to create a cookie its value is an array via the same technique used for creating parameters with a value that is an array of other values.

```
setcookie("names[0]","moshe");  
setcookie("names[1]","david");  
setcookie("names[2]","john");
```

This code creates a cookie that its value is an array that contains three values: "moshe", "david" and "john".

# Cookie Array Values

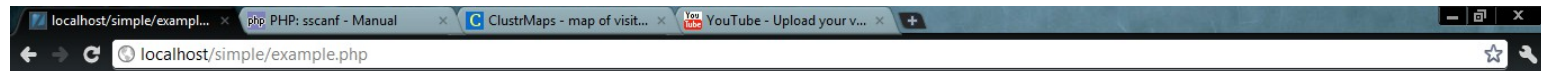
- ❖ An alternative way for setting an array values as a cookie value includes the usage of the `implode` and the `explode` functions.
- ❖ The `implode` function receives two arguments.. an array of string elements and a string that will be used as the delimiter when creating a new string based on the array elements.
- ❖ The `explode` function performs the reverse action taking a string composed of multiple string elements.

# Cookie Array Values

```
<?php
$vec=array("michael","david","john","maria");
$delimiter=",";
$str=implode($delimiter,$vec);
echo "\$str=$str";
echo "<br>";
$array=explode($delimiter,$str);
foreach($array as $k => $v)
{
    echo "<br>\$v=$v";
}
?>
```



# Cookie Array Values



`$str=michael,david,john,maria`

`$v=michael`

`$v=david`

`$v=john`

`$v=maria`



# Cookie Deletion

- ❖ There is no way to delete a cookie. There is no way to instruct the browser to delete a specific cookie.
- ❖ Passing a negative value to the `$expire` parameter will effectively kill the cookie once the session ends.

# HttpOnly Cookie

- ❖ HttpOnly cookies cannot be accessed using JavaScript. This type of cookies was introduced by Microsoft and today it is supported in most web browsers.
- ❖ The HttpOnly is an additional flag included in the Set-Cookie HTTP response header.

```
Set-Cookie: <name>=<value>[; <Max-Age>=<age>]  
[; expires=<date>][; domain=<domain_name>]  
[; path=<some_path>][; secure][; HttpOnly]
```



# HttpOnly Cookie

- ❖ Calling the `setcookie` method we just need to pass over the value `true` to the `httponly` parameter. The default value of this parameter is `false`.

```
bool setcookie ( string $name
                [, string $value
                [, int $expire = 0
                [, string $path
                [, string $domain
                [, bool $secure = false
                [, bool $httponly = false ]]]]] )
```