Índice

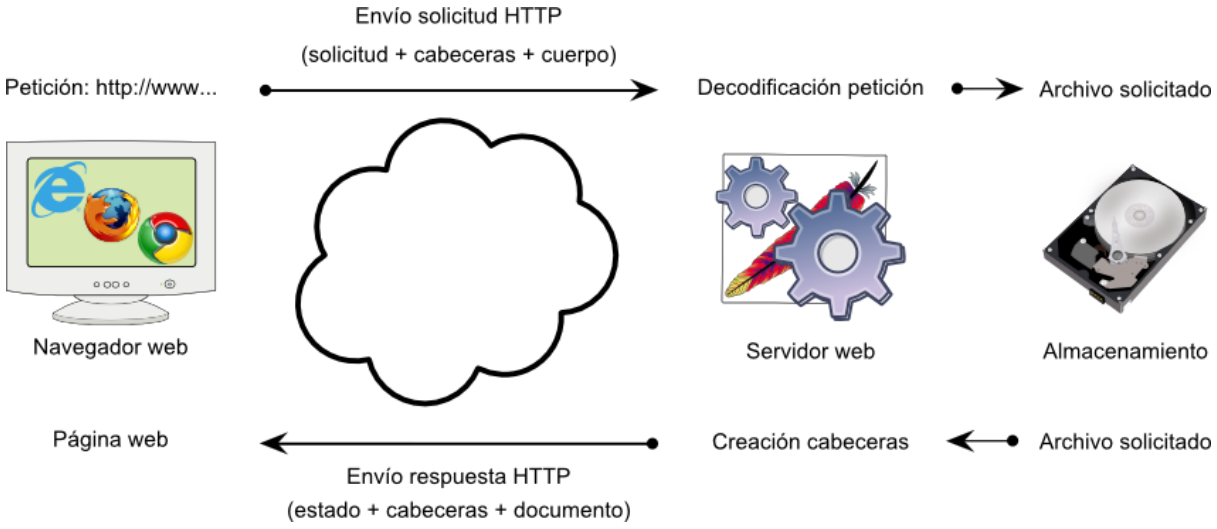
El protocolo HTTP

Vamos a comentar algunos aspectos básicos del protocolo HTTP.

Comunicación entre el cliente y el servidor

Cuando un usuario solicita una página web a un servidor, el proceso se puede describir (de forma simplificada) en cuatro pasos:

- El navegador solicita al servidor el documento mediante una solicitud HTTP.
- El servidor prepara el documento.
- El servidor envía el documento al navegador mediante una respuesta HTTP.
- El navegador muestra el documento al usuario.



Solicitudes HTTP

La solicitud HTTP está formada por varias líneas de texto:

- La línea de solicitud, que incluye:
  - El método que se va a utilizar (GET, POST, TRACE, etc.).
  - La ubicación del documento solicitado (URI).
  - La versión del protocolo HTTP.
- Los campos de cabecera, que pueden incluir entre otros:
  - Referer: dirección de donde se ha obtenido la dirección del documento solicitado (si una página A contiene un enlace a otra página B, el navegador solicita en la línea de petición la página B, pero también envía la dirección de la página A como referer).
  - User-agent: información sobre el navegador (nombre, versión, etc).
  - Accept: tipos MIME, juegos de caracteres, codificaciones, idiomas, etc. admitidos por el navegador.
  - Cookies: Si el servidor ha almacenado previamente cookies en el cliente, estas se incluyen en las peticiones.
- Una línea en blanco.
- El cuerpo del mensaje, que incluye texto opcional como por ejemplo:

- datos de un formulario cuyo método sea POST

Respuestas HTTP

La respuesta HTTP está formada por varias líneas de texto:

- La línea de estado, que incluye:
  - La versión del protocolo HTTP.
  - El código de status (403, 404, 500, etc.).
  - El texto asociado al código de status (403=Forbidden, 404=Not found, 500=Internal Server Error, etc.).
- Los campos de cabecera, que pueden incluir entre otros:
  - Location:permite decirle al cliente que solicite otro documento en lugar del documento solicitado inicialmente.
  - Content: tipo MIME, juego de caracteres, codificación, idioma, etc. del documento enviado.
  - Set-Cookie: permite decirle al cliente que cree una cookie.
- Una línea en blanco.
- El cuerpo del mensaje, que incluye el documento solicitado por el cliente.

Cabeceras: la función header()

La función header()

Cuando un servidor envía una página web al navegador, no sólo envía la página web, sino también información adicional (el estado y los campos de cabecera). Tanto el estado como los campos de cabecera se envían antes de la página web.
Normalmente, un programa PHP sólo genera la página web y es el servidor el que genera automáticamente la información de estado y los campos de cabecera y los envía antes de enviar el contenido generado por el programa. Pero un programa PHP también puede generar la información de estado y los campos de cabecera, mediante la función **header()**.

El ejemplo siguiente muestra un ejemplo de cabecera HTTP, concretamente una redirección. En el ejemplo, la página 1 contiene un enlace a la página 2. Pero como la página genera una cabecera HTTP de redirección a la página 3, al hacer clic en el enlace se muestra directamente la página 3.

Página 1 (cabeceras-header-1-1.php)

```
<p>Esta es la página 1.</p>

<p><a href="cabeceras-header-1-2.php">Enlace a la página 2 (que
redirige a la página 3)</a></p>
```

Página 2 (cabeceras-header-1-2.php)

```
<?php
```

```
header("Location:cabeceras-header-1-3.php");

print "<p>Esta es la página 2</p>";

print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";

print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
?>
```

Página 3 (cabeceras-header-1-3.php)

```
<p>Esta es la página 3.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
```

Nota: Una situación que puede provocar el fallo de la función **header()** es cuando el programa contiene un error antes de la función **header()**, ya que PHP generará un aviso de error que es también parte de la salida del programa y eso provocaría el fallo de la función **header()** posterior. Esta situación suele darse al desarrollar el programa, ya que normalmente los servidores de desarrollo están configurados para mostrar el mayor número posible de avisos de error, pero es menos probable en los servidores de producción, que se suelen configurar para no enviar al usuario ningún mensaje de error.

Página 1 (cabeceras-header-1-1.php)

```
<p>Esta es la página 1.</p>

<p><a href="cabeceras-header-1-2.php">Enlace a la página 2 (que
redirige a la página 3)</a></p>
```

Página 2 (cabeceras-header-1-2.php)

```
<?php
print "<p>Error antes del header</p>"
header("Location:cabeceras-header-1-3.php");

print "<p>Esta es la página 2</p>";

print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";

print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
```

```
?>
```

Página 3 (cabeceras-header-1-3.php)

```
<p>Esta es la página 3.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
```

No se redirecciona a la página 3 y se muestra el siguiente error:

```
Parse error: syntax error, unexpected 'header' (T_STRING) in
cabeceras-header-1-2.php on line 3
```

Redirecciones: header("Location:...")

La función **header()** se puede utilizar para redirigir automáticamente a otra página, enviando como argumento la cadena Location: seguida de la dirección absoluta o relativa de la página a la que queremos redirigir.

```
<?php
  header("Location:https://www.google.com/");
?>
```

Si además de redirigir a una página, se quieren enviar controles a dicha página, se pueden añadir a la cadena separando los controles con el carácter &.

Página 1 (cabeceras-header-1-1.php)

```
<p>Esta es la página 1.</p>

<p><a href="cabeceras-header-1-2.php">Enlace a la página 2 (que
redirige a la página 3)</a></p>
```

Página 2 (cabeceras-header-1-2.php)

```
<?php
header("Location:cabeceras-header-1-3.php?nombre=Pepe&edad=25");

print "<p>Esta es la página 2</p>";

print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";
```

```
print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
?>
```

Página 3 (cabeceras-header-1-3.php)

```
<?php
print_r ($_REQUEST);
?>
<p>Esta es la página 3.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
?>
```

```
Array
(
    [nombre] => Pepe
    [edad] => 25
)
```

Si el valor a enviar contiene espacios, se pueden escribir caracteres **+** o espacios para separar las palabras:

```
<?php
header("Location:cabeceras-header-1-3.php?nombre=Pepe+Sol&edad=25
");
?>
```

El nombre de la cabecera (location) se puede escribir en mayúsculas o minúsculas y entre los dos puntos y la dirección puede haber espacios en blanco.

```
<?php
  header("LoCaTiOn: https://www.google.com/");
?>
```

Resto del programa tras la redirección

La ejecución de un programa no se detiene al encontrar una redirección, sino que PHP ejecuta el programa hasta el final. Dependiendo de las instrucciones que haya tras la dirección, el resultado puede ser relevante o no.

Si se escriben varias redirecciones, se aplicará la última. En el ejemplo siguiente, la página redirigirá a “cabeceras-header-1-4.php”.

Página 1 (cabeceras-header-1-1.php)

```
<p>Esta es la página 1.</p>
<p><a href="cabeceras-header-1-2.php">Enlace a la página 2 (que
redirige a la página 4)</a></p>
```

Página 2 (cabeceras-header-1-2.php)

```
<?php
header("Location:cabeceras-header-1-3.php");
header("Location:cabeceras-header-1-4.php");

print "<p>Esta es la página 2</p>";

print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";

print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
?>
```

Página 3 (cabeceras-header-1-3.php)

```
<p>Esta es la página 3.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
```

Página 4 (cabeceras-header-1-4.php)

```
<p>Esta es la página 4.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
```

Si después de la redirección se genera texto, ese texto no llegará al usuario, puesto que la redirección le llevará antes a otra página.
Si el programa realiza acciones como modificar registros en una base de datos, borrar archivos, etc., esas acciones sí que se realizan.
Si tras una redirección no queremos que se ejecute el resto del programa, tenemos dos opciones:

- podemos utilizar una expresión **if** ... **else** …

```
<?php
  if (condicion) {
    header("Location:https://www.google.com");
  } else {
    …
  }
?>
```

- podemos utilizar la instrucción **exit**, que detiene el programa en ese punto.

Nota: **exit** no es una función, sino una palabra reservada del lenguaje, pero también se puede utilizar seguida de paréntesis como una función: **exit()**. Se puede incluso incluir un parámetro que puede ser un entero entre 0 y 255 o una cadena:

- Si el parámetro es un valor numérico, este valor se utilizará como estado de salida y no se imprimirá. Los estados de salida deben estar en el rango de 0 a 254, el estado de salida 255 está reservado por PHP y no se utilizará. El estado 0 se utiliza para finalizar el programa con éxito.
- Si el valor es una cadena, se imprime su valor antes de terminar.

El ejemplo anterior se podría escribir entonces de la siguiente manera, sin necesidad de incluir las instrucciones posteriores en un bloque else ...:

Para el ejemplo anterior se podría escribir entonces **exit** antes de redireccionar a la página “cabeceras-header-1-4.php”

Página 1 (cabeceras-header-1-1.php)

```
<p>Esta es la página 1.</p>
<p><a href="cabeceras-header-1-2.php">Enlace a la página 2 (que
redirige a la página 3)</a></p>
```

Página 2 (cabeceras-header-1-2.php)

```
<?php
header("Location:cabeceras-header-1-3.php");
exit;
header("Location:cabeceras-header-1-4.php");

print "<p>Esta es la página 2</p>";

print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";

print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
```

```
?>
```

Página 3 (cabeceras-header-1-3.php)

```
<p>Esta es la página 3.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
```

Página 4 (cabeceras-header-1-4.php)

```
<p>Esta es la página 4.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
```

Pero hay que tener en cuenta que **exit** detiene no sólo la ejecución del programa, sino también la de los programas que hubieran llamado a esos programas, como se muestra a continuación, que se redireccionará la página “cabeceras-header-1-4.php” en lugar de a “cabeceras-header-1-3.php”.

Página 1 (cabeceras-header-1-1.php)

```
<p>Esta es la página 1.</p>
<p><a href="cabeceras-header-1-2.php">Enlace a la página 2 (que
redirige a la página 4)</a></p>
```

Página 2 (cabeceras-header-1-2.php)

```
<?php
include "cabeceras-header-1-4.php";
exit;
header("Location:cabeceras-header-1-3.php");

print "<p>Esta es la página 2</p>";

print "<p>La redirección <strong>NO</strong> se ha
realizado</p>";

print "<p><a href=\"cabeceras-header-1-1.php\">Volver a la página
1</a></p>";
?>
```

Página 3 (cabeceras-header-1-3.php)

```
<p>Esta es la página 3.</p>

<p>La redirección <strong>SÍ</strong > se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
?>
```

Página 4 (cabeceras-header-1-4.php)

```
<p>Esta es la página 4.</p>

<p>La redirección <strong>SÍ</strong> se ha realizado.</p>

<p><a href="cabeceras-header-1-1.php">Volver a la página
1</a></p>
?>
```

Crear con PHP otros tipos de archivos

Con instrucciones print, PHP puede generar archivos con cualquier contenido (archivos de texto o incluso binarios), pero para que el navegador acepte esos archivos y los interprete correctamente, en las cabeceras se debe incluir el tipo MIME correspondiente. Existen muchos tipos MIME (véase lista de tipos MIME en MDN), por ejemplo:

extensión del archivo	tipo de archivo	tipo MIME
.html	página web HTML	text/html
.css	hoja de estilo CSS	text/css
.js	JavaScript	application/js
.svg	SVG	image/svg+xml
.json	JSON	application/json

El tipo MIME se indica en la cabecera, por lo que un programa PHP puede declarar el tipo MIME mediante la función **header()**, enviando como argumento la cadena **Content-type**: seguida del tipo MIME correspondiente.

Crear hojas de estilo CSS

El programa siguiente genera el texto correspondiente a una hoja de estilo CSS, concretamente dando color al fondo y al texto de forma aleatoria.

```
<?php
  $color = rand(0, 360);
  print "body";
  print "  background-color: hsl($color, 50%, 90%);";
  print "  color: hsl($color, 50%, 30%);";
  print "}";
?>
```

Pero si llamamos a este programa en la etiqueta **<link>** de una página HTML, podemos comprobar que el navegador no aplica la hoja de estilo, como en el ejemplo siguiente. El problema se debe a que el servidor asigna por defecto el tipo MIME **text/html** a los ficheros generados por un programa PHP, pero los navegadores exigen que las hojas de estilo lleguen con el tipo MIME **text/css**. Aunque el contenido corresponda a una hoja de estilo, al no recibir el tipo MIME esperado, el navegador no interpreta su contenido.

```
<link rel="stylesheet" href="mime-css.php" title="Color">
```

```
<?php
  $c1 = rand(0, 255);
  $c2 = rand(0, 255);
  $c3 = rand(0, 255);
  print "body{";
  print "  background-color: rgb($c1, $c2, $c3);";
  print "  color: rgb($c2, $c3, $c1);";
  print "}";
?>
```

Para que el navegador aplique la hoja de estilo, el programa debe generar una cabecera que declare el tipo MIME de hoja de estilos **text/css**, como muestra el siguiente ejemplo.

```
<link rel="stylesheet" href="mime-css.php" title="Color">
```

```
<?php
  header("Content-type: text/css");
  $c1 = rand(0, 255);
  $c2 = rand(0, 255);
  $c3 = rand(0, 255);
  print "body{";
  print "  background-color: rgb($c1, $c2, $c3);";
  print "  color: rgb($c2, $c3, $c1);";
  print "}";
?>
```

Crear imágenes SVG

El programa siguiente crea las etiquetas correspondientes a una imagen SVG, concretamente a un cuadrado de color aleatorio. Si escribimos directamente la url de ese programa en el navegador, el navegador muestra el contenido esperado.

```
<?php
  $c1 = rand(0, 255);
  $c2 = rand(0, 255);
  $c3 = rand(0, 255);
  print "<svg version=\"1.1\"
xmlns=\"http://www.w3.org/2000/svg\" "   . "    width=\"100\"
height=\"100\" viewBox=\"-5 -5 100 100\">";
  print " <rect fill=\"rgb($c1, $c2, $c3)\" x=\"0\" y=\"0\"
width=\"90\" height=\"90\" />";
  print "</svg>";
?>
```

Pero aunque el navegador consiga mostrar el cuadrado, realmente no estamos haciendo bien las cosas. Una imagen SVG debería entregarse al navegador con el tipo MIME **image/svg+xml**. En este caso, el programa no declara ningún tipo MIME, así que el servidor lo entrega con el tipo MIME predeterminado de los programas PHP, el tipo MIME **text/html** (es decir, como página web). Aunque estrictamente hablando el documento no es una página web válida porque le faltan las etiquetas básicas como <html>, <head>, <body>, etc. los navegadores aceptan las páginas inválidas e intentan interpretar el contenido de la mejor manera posible. En este caso, el navegador consigue mostrar la imagen.

Pero en otras circunstancias, este mismo programa no daría el resultado esperado. Como muestra el siguiente ejemplo, si el programa se llama en una etiqueta <img>, el navegador no puede mostrar la imagen. El problema es que en una etiqueta el navegador tiene que recibir obligatoriamente algún tipo MIME de imagen (jpg, png, svg, etc.). Como pasaba en el ejemplo anterior de la hoja de estilo, si el documento se sirve con un tipo MIME incorrecto, los navegadores no intentan siquiera mostrar el contenido.

```
<p><img src="svg-sin-mime.php" alt="ERROR No puedo mostrar la
imagen"></p>
```

```
<?php
  $c1 = rand(0, 255);
  $c2 = rand(0, 255);
  $c3 = rand(0, 255);
```

```
  print "<svg version=\"1.1\"
xmlns=\"http://www.w3.org/2000/svg\" "   . "    width=\"100\"
height=\"100\" viewBox=\"-5 -5 100 100\">";
  print " <rect fill=\"rgb($c1, $c2, $c3)\" x=\"0\" y=\"0\"
width=\"90\" height=\"90\" />";
  print "</svg>";
?>
```

```
ERROR No puedo mostrar la imagen
```

Para que el navegador muestre correctamente la imagen, el programa debe generar una cabecera que declare el tipo MIME de imagen **image/svg+xml**, como muestra el siguiente ejemplo.

```
<p><img src="svg-con-mime.php" alt="ERROR No puedo mostrar la
imagen"></p>
```

```
<?php
  header("Content-type: image/svg+xml");
  $c1 = rand(0, 255);
  $c2 = rand(0, 255);
  $c3 = rand(0, 255);
  print "<svg version=\"1.1\"
xmlns=\"http://www.w3.org/2000/svg\" "   . "    width=\"100\"
height=\"100\" viewBox=\"-5 -5 100 100\">";
  print " <rect fill=\"rgb($c1, $c2, $c3)\" x=\"0\" y=\"0\"
width=\"90\" height=\"90\" />";
  print "</svg>";
?>
```