

# CABECERAS

- 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()**.

# Función header

- La función `header()` debe ser llamado antes de mostrar nada por pantalla:
  - Ni etiquetas HTML (por ejemplo, con un `print`),
  - Ni líneas en blanco desde un fichero o desde PHP.
  - Ni errores generados en el programa
- Es un error muy común leer código con funciones como `include` o `require`, u otro tipo de funciones de acceso de ficheros que incluyen espacios o líneas en blanco que se muestran antes de llamar a la función **`header()`**. Sucede el mismo problema cuando se utiliza un solo fichero PHP/HTML.

Hay que tener en cuenta que una simple línea en blanco antes del bloque PHP sería suficiente para impedir el envío de la cabecera:

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

**Warning:** Cannot modify header information  
- headers already sent by (output started at  
ejemplo.php:2) in **ejemplo.php** on line 3

Incluso la **etiqueta** <html>

```
1 <html>  
2 <?php  
3 header("Location: http://www.google.com/"); // Error
```

# Función header

- El motivo es que en cuanto un programa genera contenido HTML, el servidor genera automáticamente la información de estado y los campos de cabecera, y a continuación envía el contenido generado.
- Si después el programa contiene una instrucción `header()`, se produce un error porque las cabeceras ya se han enviado, y esta cabecera no se puede generar, y que las cabeceras solo se envían 1 sola vez.
- Lo mismo ocurre 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.

# Función header

- Esta función 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.
- Se puede escribir "location" o "Location" y entre los dos puntos y la dirección puede haber espacios en blanco.

# Función header

- Redirigir al cliente a otra dirección.
  - Esta puede ser absoluta, si está en nuestro servidor o relativa si esta en otro, en este caso hay que poner la URL

```
<?php  
header("Location: http://www.example.com");  
exit;  
?>
```

# Función header

- 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 &.
  - En este caso no se debe utilizar la entidad de carácter **&amp**

```
<?php
header("Location: http://www.example.com?nombre=elena&edad=18");
exit;
?>
```

# Redireccionar una página a una URL después de esperar X segundos en PHP.

- **Refresh** redirige a los usuarios al igual que Location, pero se puede añadir un retardo antes de que se redirija al usuario.

```
<?php
header ( 'Refresh: 10; url=http://www.mysite.com/otherpage.php' );
echo 'vas a ser redirigido en 10 segundos' ;
?>
```




# 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.
  - a) Si se escriben varias redirecciones, se aplicará la última.

```
<?php  
  
header("Location: http://www.example0.com")  
  
header("Location:http://www.example1.com/");  
header("Location:http://www.example2.org/");  
// Esta página redirige a www.example2.org  
  
exit;  
?>
```


# Resto del programa tras la redirección

b) Si después de la redirección se genera texto, ese texto no llegará al usuario, puesto que la redirección le llevará antes a otra página.

```
<?php  
  
header("Location: http://www.example.com")  
  
print "<p>esto no se visualiza</p>\n";  
  
exit;  
?>
```

# Resto del programa tras la redirección

c) Si el programa realiza acciones como modificar registros en una base de datos, borrar archivos, etc., esas acciones sí que se realizarán. Por ejemplo, el siguiente programa redirige a una página, pero también se borra a sí mismo.

```
 <?php  
  
header("Location: http://www.example.org/");  
  
unlink("ejemplo.php");  
?>
```

# Control del programa tras la redirección

- Si tras una redirección no queremos que se ejecute el resto del programa, tenemos dos opciones:
  - Discriminar sentencias con if ... else ...

```
<?php
if (condicion) {
    header("Location: http://www.example.org/");
} else {
    ...
}
?>
```

# Resto del programa tras la redirección

- Deteniendo el programa en un punto determinado:
  - La instrucción `exit`, detiene el programa en el punto en que aparece.
  - `Exit` no es una función, sino una palabra reservada del lenguaje, pero si se utiliza seguida de un paréntesis incluyendo un valor numérico en los paréntesis este valor lo devuelve al programa, sirviendo de dato para evaluar en un control.

```
<?php
if (condicion) {
    header("Location:http://www.example.org/");
    exit;
}
// ...
?>
```

# Resto del programa tras la redirección

- 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.

```
<?php
// ejemplo-1.php
include "ejemplo-2.php";
header("Location: http://www.exaple1.org/");
?>
```

```
<?php
// ejemplo-2.php
header("Location: http://www.example2.com/");
?>
```

```
// el resultado final será una redirección a la página web de example1.
```

# Resto del programa tras la redirección

```
<?php
// ejemplo-3.php
include "ejemplo-4.php";
header("Location: http://www.example2.org/");
?>
```

```
<?php
// ejemplo-4.php
header("Location: http://www.example.com/");
exit;
?>
```

// el resultado final será una redirección a la página <http://www.example.com/>.