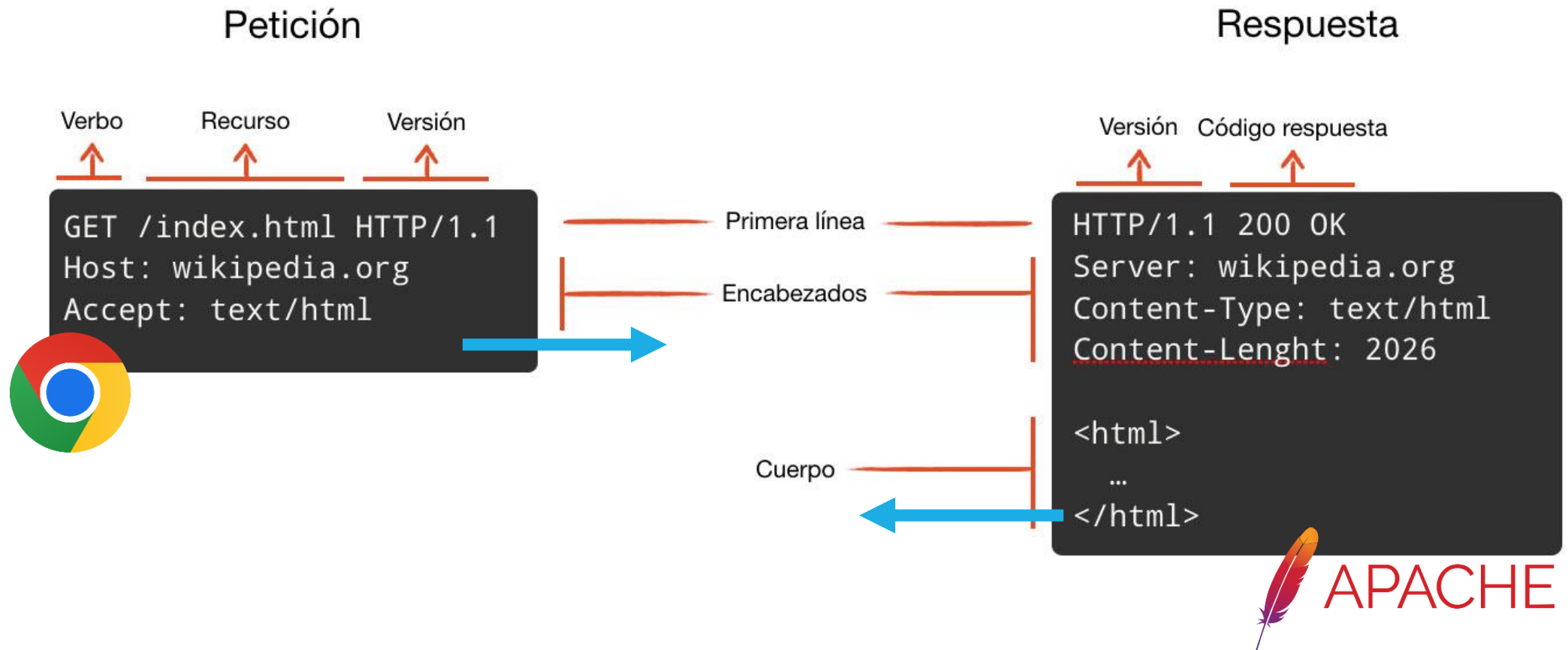


UD3. Desarrollo de Aplicaciones Web con PHP

DESARROLLO WEB EN ENTORNO SERVIDOR

PASO DE PARÁMETROS

Métodos de acceso



Métodos de acceso

Hay muchos métodos/verbos HTTP.

Ten en cuenta:

- El servidor (normalmente) sirve la página que buscamos ignorando el método.
- Esto significa que una web puede ser accedida con cualquiera.
- La página es la encargada de qué hacer con el método de acceso.
- ¿Cómo sé el método con el que accede el cliente?

```
$_SERVER["REQUEST_METHOD"];
```

Métodos de acceso

Los navegadores principalmente utilizan 2 métodos: GET y POST.

GET:

¿Cuándo se emplea?

- Se visita un enlace.
- Se introduce la URL a mano en la barra de direcciones.
- Se envía un formulario de tipo GET.
- La web a la que hemos accedido redirecciona a otra.

POST:

¿Cuándo se emplea?:

- Se envía un formulario de tipo POST (lo habitual).

Paso de parámetros

El navegador puede enviar parámetros al servidor web.

Estos parámetros tienen el formato: `nombre=valor` (el valor es opcional)

-
-
-
-

Paso de parámetros

El navegador puede enviar parámetros al servidor web.

Estos parámetros tienen el formato: `nombre=valor` (el valor es opcional)

Dependiendo del método de acceso, se hace de una manera u otra:

Con GET:

- Van codificados en la URL que se solicita al servidor:
- `http://URL?parametro1=nombre¶metro2`

Con POST:

- Van codificados en las cabeceras que se envían al servidor (inspector de elementos).
- El navegador lo hace automáticamente al enviar los datos de un formulario POST.

Acceso a los parámetros


hola_nombre.php
hola_comprobacion.php

PHP guarda todos los parámetros en arrays superglobales:

- `$_GET`
- `$_POST`
- `$_REQUEST` (combina los datos de los dos anteriores, **no lo usamos**).



`http://URL?parametro1=nombre¶metro2`



```
echo $_GET["parametro1"];  
echo $_GET["parametro2"];
```



Acceso a los parámetros

hola_nombre.php
hola_comprobacion.php

En general:

1. Miro el rol que está tomando mi página (si me interesa).
 - `$_SERVER["REQUEST_METHOD"] == "GET"`
2. Compruebo que el parámetro que espero me llega (y controlo qué ocurre si no).
 - `isset($_GET["usuario"])`
3. Compruebo que el parámetro tenga algún valor dentro (si me interesa).
 - `empty($_GET["usuario"])`
4. Hago lo que tenga que hacer con el valor.
 - `echo "Bienvenido " . $_GET["usuario"];`

Formularios

login_basico.php

Cada vez que hagas un formulario, ten en cuenta:

- El atributo **method** determina según el valor si el navegador envía al servidor los parámetros como POST o GET.

Si está en blanco → el navegador usará GET (**evítalo**).

- El atributo **action** determina a qué página PHP del servidor se envía la petición.

Si está en blanco → la misma página que imprime el formulario lo recibe.

Formularios

login_basico.php

Ya dentro del formulario, para cada control que haya (`input`, `textarea`, `select`, etc.), ten en cuenta:

- **El atributo `name`** es obligatorio para que pueda recuperarse en el servidor, es el nombre del parámetro.

Si está en blanco → el navegador **no** enviará el control.

- **No esperes que todos funcionen igual:** hay campos que, si el usuario no selecciona, no se envían al servidor; otros que, aunque estén en blanco, se envían siempre; etc.

En el AV:



Anexo - paso de parámetros y formularios

Formularios - ejemplo

Formulario

```
<!DOCTYPE html>
<html>
  <head>
    <title>Formulario de login</title>
    <meta charset = "UTF-8">
  </head>
  <body>
    <form action = "login.php" method = "POST">
      <input name = "usuario" type = "text">
      <input name = "clave" type = "password">
      <input type = "submit">
    </form>
  </body>
```

Procesamiento (login.php)

```
<?php
echo "Usuario introducido: " .
  $_POST['usuario']. "<br>";
echo "Clave introducida: " .
  $_POST['clave'];
```

Formularios - una o dos páginas

un_login.php
(en otros ejemplos)

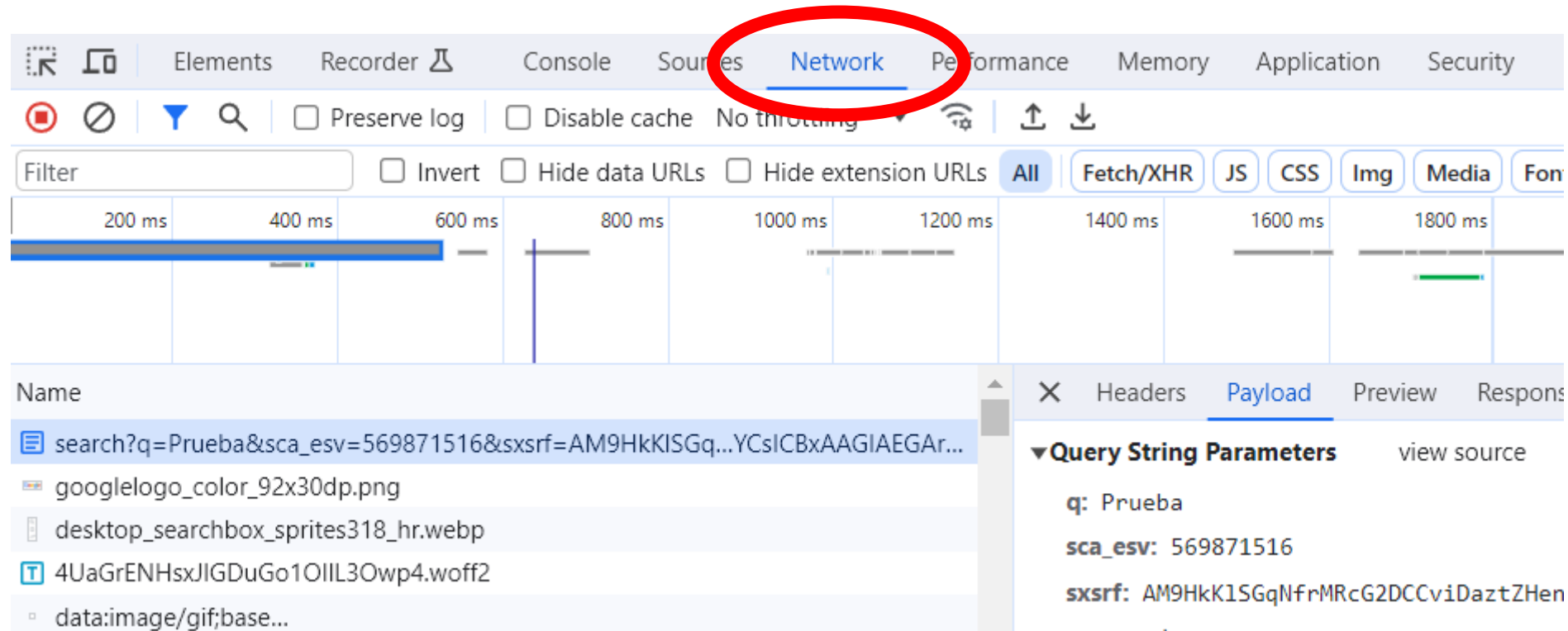
Si te fijas, necesitamos un trozo de código que envíe el formulario al usuario, y otro trozo que pueda procesar los parámetros de vuelta.

Distinguimos dos situaciones:

- **Dos páginas:** una genera el formulario y otra recoge la respuesta.
 - Atributo `action` del `<form>` apunta a otra página.
- **Una página,** que genera y recoge la respuesta a la vez:
 1. Distingo el método de acceso, por ejemplo, con `$_SERVER["REQUEST_METHOD"]`;
 2. Si GET → muestro el formulario.
 3. Si POST → proceso el formulario.
 - Atributo `action` del `<form>` está vacío.

Inspector de elementos

Tu mejor amigo para depurar el envío de parámetros:



Redirecciones de servidor

La función `header` permite escribir crear una cabecera que enviará el servidor al cliente.

Principalmente se usa para indicar redirecciones al navegador:

```
header("Location: http://www.google.es");
```

Validación de formularios

Los input a veces generan el envío de parámetros al servidor y a veces no:

- Los campos de texto siempre se envían, hay que comprobar que no están vacíos.
- Los controles radio y checkbox solo están definidos si se marcan.

Además de esto, siempre hay que validar la información en el lado servidor:

- Comprobar que un número es un número (funciones `is_numeric`, `intval`, `floatval`, etc.)
- Comprobar que un string es correcto (funciones vistas).
- Eliminar espacios iniciales y finales: `trim()`
- Etc.

Validación de formularios

Entradas con caracteres especiales:

The diagram illustrates a security vulnerability in a web application's form validation. It shows a sequence of three browser windows:

- Window 1:** A form titled "Por favor, indique su nombre:" with a text input field containing the text `Juan|` and an "Enviar" button.
- Window 2:** The result of a successful submission, showing "Hola **Juan**".
- Window 3:** A malicious submission. The URL bar shows `localhost/CursoPHP/procesaform.php?cliente= <body+onload=%3D\"alert('Hola')\"`. The page content shows "Hola" and a modal dialog box with "Hola" and an "OK" button.

Validación de formularios

`strip_tags()` elimina las posibles etiquetas que haya introducido el usuario.

`htmlspecialchars()` convierte caracteres especiales en entidades HTML:

- `>` `>`;
- `<` `<`;
- `"` `"`;

Validación de formularios

Saneamiento de la entrada: elimina caracteres problemáticos en base a la constante (el filtro) que se indique.

```
filter_var($_POST["POST"], FILTER_SANITIZE_EMAIL)
```

<https://www.php.net/manual/es/filter.filters.sanitize.php>

Comprobación de la entrada: devuelve true o false dependiendo de si la entrada cumple con la constante (el filtro) que se indique.

```
filter_var($_POST["POST"], FILTER_VALIDATE_EMAIL)
```

<https://www.php.net/manual/es/filter.filters.validate.php>

Validación de formularios

Expresiones regulares:.

```
preg_match("/[A-Z]{4}/", $matricula)
```

Devuelve 1 si coincide y 0 si no.

Formularios: envío de ficheros

procesar_subida.php

Caso particular de un formulario:

```
<form action="procesar_subida.php" method="post" enctype="multipart/form-data">
  <input type="file" name="fichero">
  <input type="file" name="fichero_B">

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

</form>
```

- En este caso, accedemos al fichero subido con `$_FILES`:
 - `$_FILES["fichero_B"]`

Formularios: envío de ficheros

procesar_subida.php

Accedemos al fichero subido con el array superglobal `$_FILES`:

- `$_FILES["fichero_B"]`

Elementos del fichero:

- `$_FILES["fichero_B"]["name"]`: nombre del fichero en el cliente.
- `$_FILES["fichero_B"]["size"]`: tamaño en bytes.
- `$_FILES["fichero_B"]["type"]`: tipo MIME
- `$_FILES["fichero_B"]["tmp_name"]`: nombre temporal del fichero en el servidor.
- `$_FILES["fichero_B"]["error"]`: código de error en la subida.

Formularios: envío de ficheros

procesar_subida.php

El fichero se guarda temporalmente en el servidor (/tmp).

Si no se mueve, se borra cuando acabe el script.

Se puede mover con:

```
move_uploaded_file($fichero, $destino);
```