



## 5. PHP - Peticiones HTTP (Get y Post)

---

Profesor: Alejandro Amat Reina



# ¿Qué es una petición HTTP?

---

- Solicitud de un recurso al servidor
- Se realiza a través de una url
- Se pueden pasar parámetros con la petición
- Hay distintos metodos (METHOD) de realizar una petición (GET, POST, PUT, DELETE, PATCH, etc.)
- Los más habituales son GET y POST
- El resto son más utilizados en REST

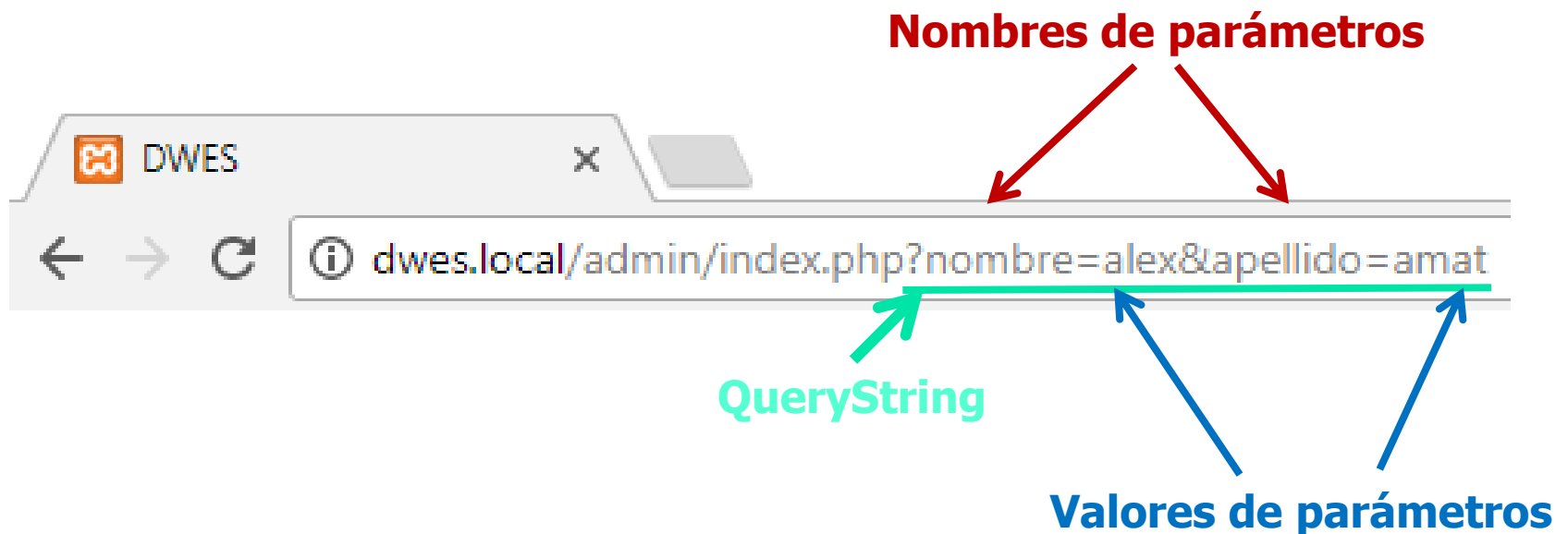


# Petición GET


---

- Se utiliza para solicitar datos de un recurso
- Muestran los parámetros que se envían en la url
- Se pueden utilizar directamente en enlaces
- El resultado se puede almacenar en cache
- Permanecen en el historial del navegador
- El tamaño de los parámetros está limitado a 255 caracteres

# Ejemplo de petición GET



# Inspección de la petición



X Headers Preview Response Timing

▼ General

Request URL: `http://dwes.local/admin/index.php?nombre=alex&apellido=amat`

Request Method: GET ← Usamos el método GET

Status Code: ● 200 OK

Remote Address: 127.0.0.1:80

Referrer Policy: no-referrer-when-downgrade

► Response Headers (7)

► Request Headers (8)

▼ Query String Parameters view source view URL encoded

nombre: alex  
apellido: amat ← Parámetros de la petición



# Acceder a los datos de la petición GET

---

- Usamos la variable superglobal **\$\_GET**
- Es un array asociativo
- Las claves del array coincidirán con los nombres que le hemos dado a los parámetros.
- Para acceder a los parámetros de la petición anterior:

```
echo $_GET['nombre'] . ' ' . $_GET['apellido'];
```



# Ejercicio

---

- Crea una página que reciba como parámetro un nombre y muestre el texto 'Bienvenido nombre!!!'



# Petición POST

---

- Se utiliza para enviar datos a un recurso
- Los parámetros van en el cuerpo de la petición, no son visibles para el usuario
- La petición no se guarda en cache
- No se puede utilizar en un enlace
- No permanece en el historial
- Se suelen utilizar en los formularios
- No tenemos la limitación de tamaño de los parámetros





# Formulario web

---

El formulario enviará los datos a index.php

Utiliza el método post

```
<form action="index.php" method="post">
  <label for="nombre">Nombre</label>
  <input type="text" name="nombre" value="">
  <br>
  <label for="apellido">Apellido</label>
  <input type="text" name="apellido" value="">
  <br>
  <br>
  <input type="submit" value="Enviar">
</form>
```

Nombres con los que podremos acceder a los parámetros en el servidor

# Inspección de la petición

× Headers Preview Response Timing

▼ General

Request URL: `http://dwes.local/admin/index.php`

Request Method: `POST`

Status Code: ● 200 OK

Remote Address: `127.0.0.1:80`

Referrer Policy: `no-referrer-when-downgrade`

▶ Response Headers (7)

▶ Request Headers (12)

▼ Form Data    view source    view URL encoded

nombre: alex

apellido: amat

← No se muestran los datos en la url

← Se utiliza el método post

← Parámetros enviados



# Acceder a los datos de la petición POST

---

- Usamos la variable superglobal `$_POST`
- Funciona igual que `$_POST`, pero con los nombres que le hemos dado a los campos del formulario.
- Mostrar todos los datos recibidos:

```
var_dump($_POST);
```

- Mostrar los datos individualmente:

```
echo $_POST['nombre'];  
echo $_POST['apellido'];
```



# Ejercicio

---

- Crea un formulario que tenga los siguientes campos:
  - Nombre
  - teléfono
  - E-mail
  - Fecha de nacimiento
- En el action del formulario pondremos lo siguiente:  
**action="<?= \$\_SERVER['PHP\_SELF']; ?>"**
- Esto hará que sea la propia página del formulario la que procese los datos del mismo.
- Al pulsar enviar deben aparecer debajo del formulario los datos que se han introducido en el mismo.

# Acceder a parámetros no existentes

**(!)** Notice: Undefined index: date in /var/www/php/app/public/index.php on line 2

Call Stack

#	Time	Memory	Function	Location
1	0.0008	357024	{main}()	.../index.php:0
2	0.0067	362152	require( 'var/www/php/app/public/index.php' )	.../index.php:3

**(!)** Notice: Undefined index: email in /var/www/php/app/public/index.php on line 3

Call Stack

#	Time	Memory	Function	Location
1	0.0008	357024	{main}()	.../index.php:0
2	0.0067	362152	require( 'var/www/php/app/public/index.php' )	.../index.php:3

**(!)** Notice: Undefined index: desc in /var/www/php/app/public/index.php on line 4

Call Stack

#	Time	Memory	Function	Location
1	0.0008	357024	{main}()	.../index.php:0
2	0.0067	362152	require( 'var/www/php/public/index.php' )	.../index.php:3

Esto nos dice que la clave date no existe en el array `$_POST`



# Verificar que el formulario se ha enviado

---

- Antes de mostrar los datos verificaremos que se haya enviado el formulario:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST')  
{  
    ...  
}
```



# Ejercicio

---

- Soluciona el problema de los parámetros no enviados del ejercicio anterior



# Validación del formulario

---

- Debemos comprobar que los datos del formulario son correctos
- Validaciones a realizar:
  - Los campos requeridos no deben quedar vacíos
  - Los campos email y fecha deben tener el formato esperado
  - Todos los campos se deben filtrar con `htmlspecialchars`





# Valores vacíos

---

- Los campos requeridos no deberían quedarse vacíos.
- Para verificar que un valor no queda vacío podemos utilizar la función empty de PHP.
  - <http://php.net/manual/es/function.empty.php>



# Espacios en blanco

---

- Debemos eliminar los espacios en blanco del principio y final de los campos
- Se utiliza la función trim



# Filtrar la entrada

---

- Siempre debemos filtrar la entrada con `htmlspecialchars` antes de mostrar el campo con `echo` o similar



# Comprobar el email

---

- Para verificar si un email es correcto podemos utilizar la función `filter_var`
- `filter_var($email, FILTER_VALIDATE_EMAIL)`
  - <http://php.net/manual/es/function.filter-var.php>



# Comprobar la fecha

---

- Para comprobar la fecha debemos de crear una función a tal efecto
- Podemos utilizar el método `createFromFormat` de la clase `DateTime`
  - <http://php.net/manual/es/datetime.createfromformat.php>



# Ejercicio

---

- Modifica el ejercicio anterior realizando las validaciones oportunas

# Ejercicio (I)

- Debes crear un script que muestre y visualice los datos de un formulario



The screenshot shows a Mozilla Firefox browser window with the title 'Búsqueda de canciones. - Mozilla Firefox'. The address bar displays 'http://localhost:8'. The page content includes a search form with the following elements:

- Búsqueda de canciones** (Title)
- Texto a buscar:** A text input field containing the word 'amor'.
- Buscar en:** Three radio buttons for search criteria: 'Títulos de canción', 'Nombres de álbum', and 'Ambos campos' (which is selected).
- Género musical:** A dropdown menu currently showing 'Pop'.
- Buscar** (Submit button)



## Ejercicio (II)

---

- Como se puede observar en la imagen anterior, el formulario pide datos para la búsqueda de canciones, donde los botones de selección 'Títulos de canción', 'Nombres de álbum' y 'Ambos campos', devuelven los valores título, álbum y ambos respectivamente.
- EL pop-up de género musical me ofrecerá las siguientes posibilidades: Todos (seleccionado por defecto), Blues, Jazz, Pop y Rock
- Las canciones las tendremos almacenadas en un array asociativo
- Tras pulsar buscar, el mismo script mostrará los resultados de la búsqueda
- Para realizar la búsqueda en el array utilizaremos la función `array_filter`





## Ejercicio (III)

---

- Vamos a añadir dos opciones más a las que accederemos mediante dos enlaces:
  - La primera mostrará en pantalla toda la información del array de canciones. Para ello utilizaremos la función `array_map` para generar un array de strings con las líneas html que se mostrarán en la página
  - La segunda mostrará en pantalla los distintos géneros de canciones (sin duplicados) que tenemos en el array ordenados alfabéticamente. Para ello utilizaremos las funciones `array_column`, `array_unique` y `sort`