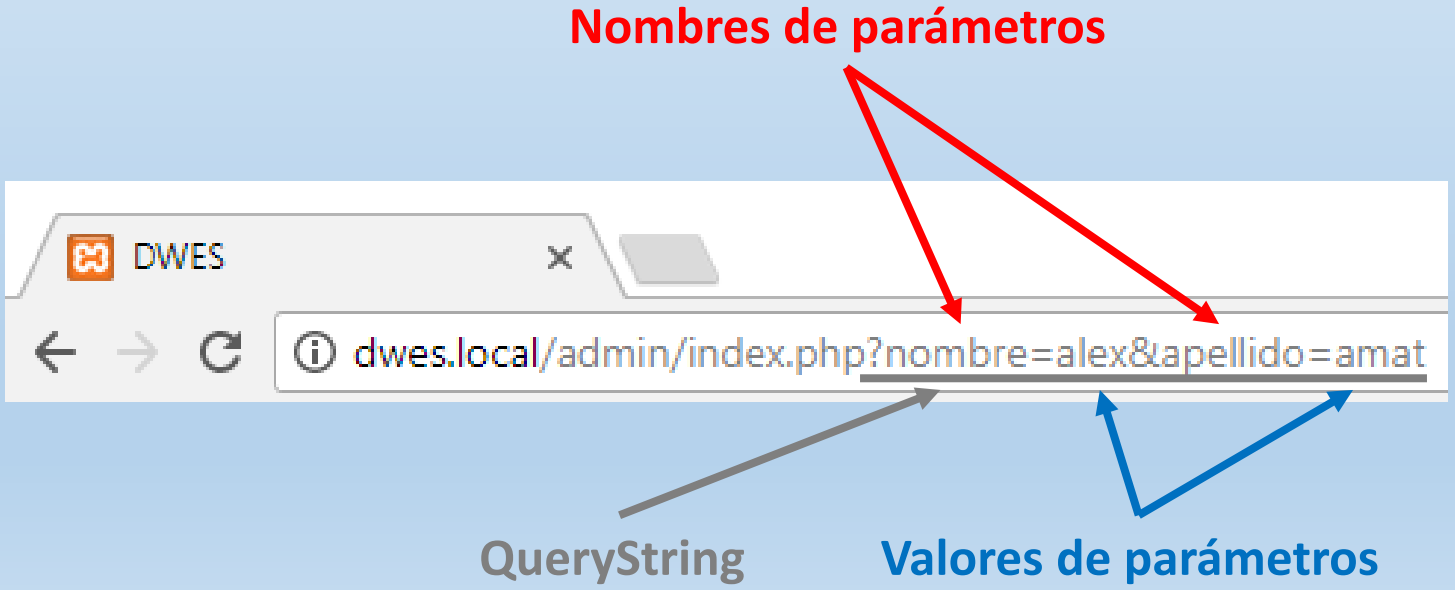
Peticiones HTTP en el lenguaje PHP

Peticiones HTTP

- Es una **solicitud de un recurso** al servidor.
- Se realiza a través de una url.
- Se pueden pasar **parámetros** con la petición.
- Hay distintos métodos (**METHOD**) de realizar una petición (GET, POST, PUT, DELETE, PATCH, etc.).
- Los más habituales son **GET** y **POST**

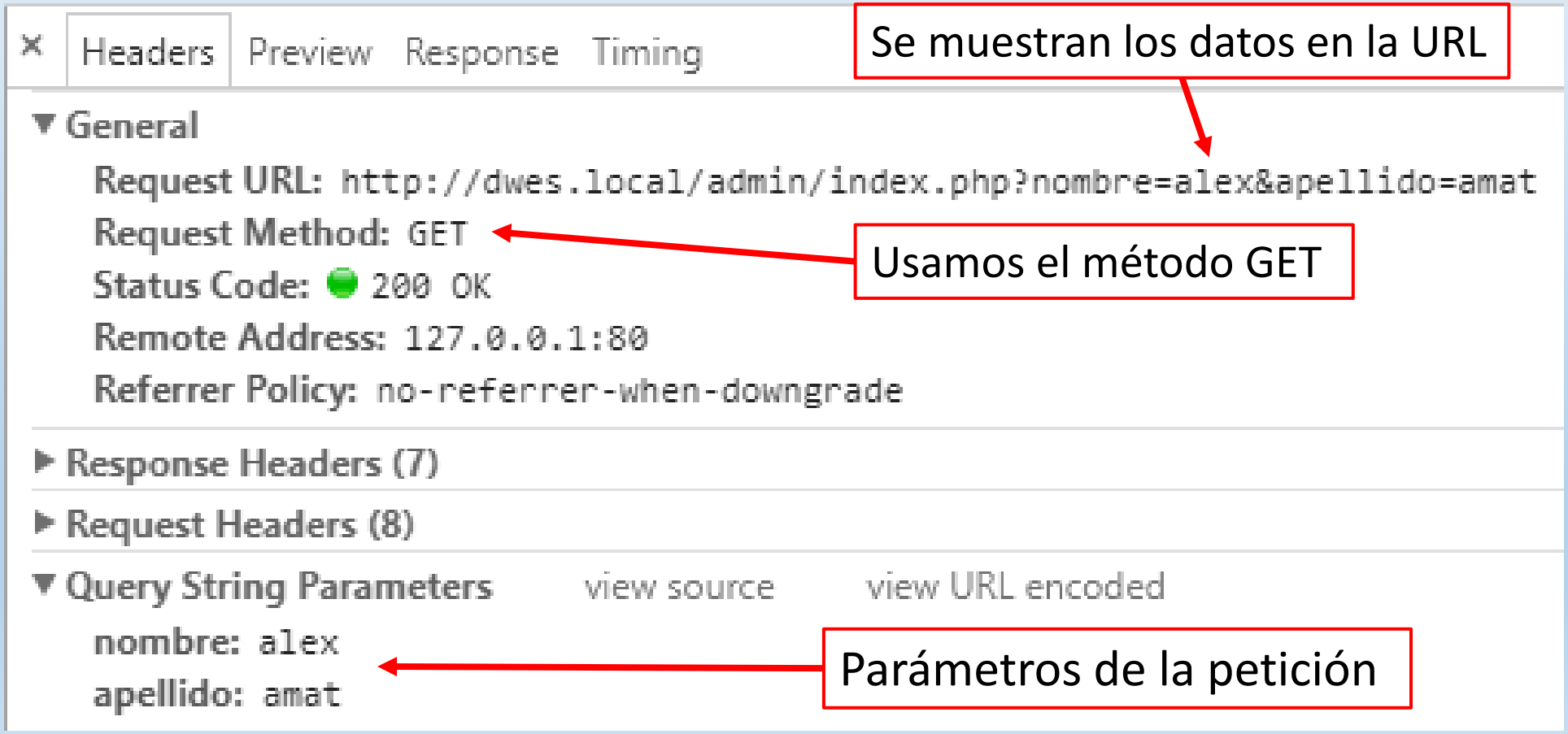
Peticiones HTTP: GET

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



Peticiones HTTP: Inspección petición GET

Pulsando **F12** en el navegador:



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:

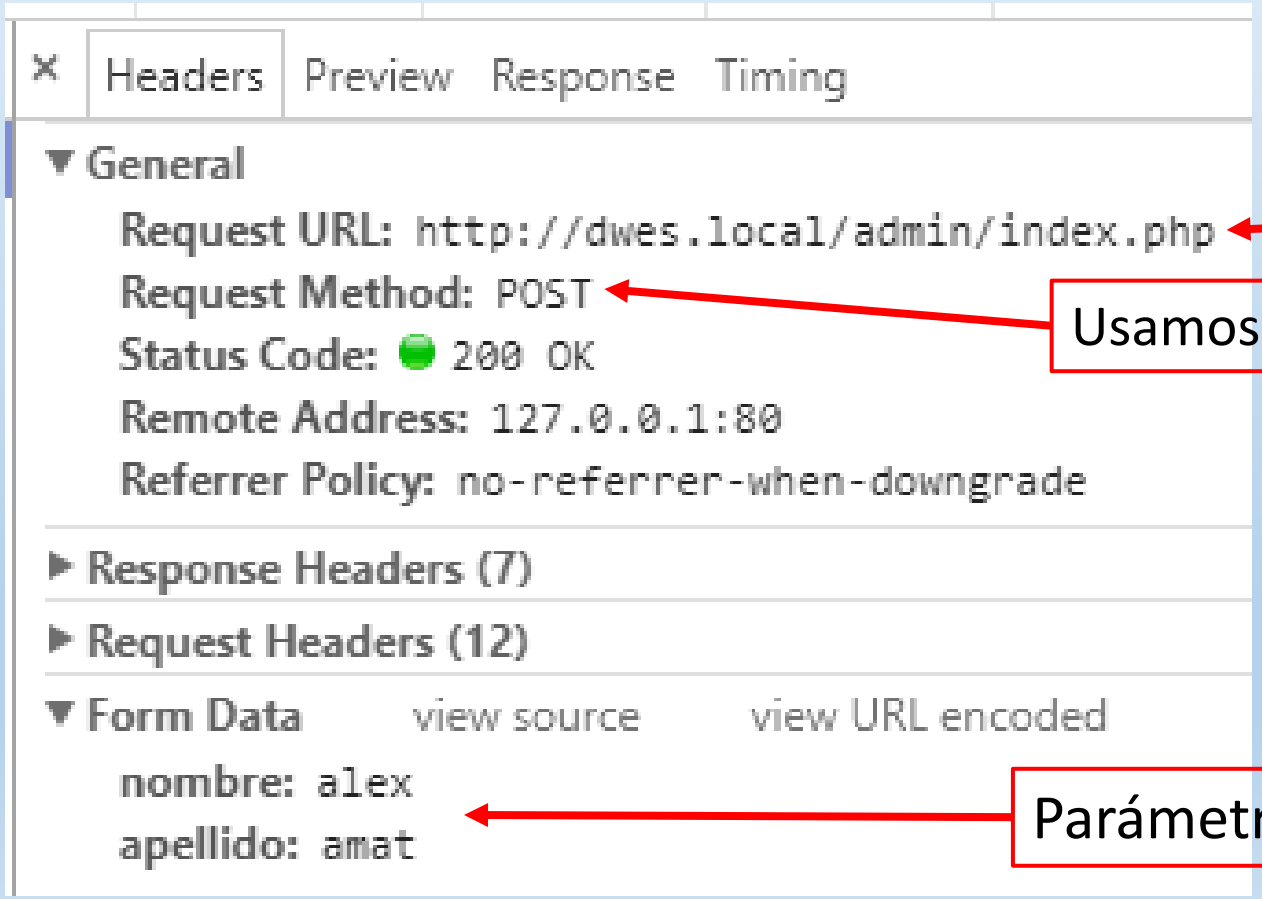
```
echo $_GET['nombre']. ' ' . $_GET['apellido'];
```

Peticiones HTTP: POST

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

Peticiones HTTP: Inspección petición POST

Pulsando **F12** en el navegador:



No se muestran los datos en la URL

Usamos el método POST

Parámetros enviados

Acceder a los datos de la petición GET

- Usamos la variable superglobal **$_POST**
- Funciona igual que **$_GET**, 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'];
```

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

Enviar los datos al mismo script que muestra el formulario

- La variable **$_SERVER** contiene datos relacionados con el entorno del servidor de la petición HTTP.
- Una de los datos que contiene es el **script php que se está ejecutando** ($_SERVER[**'PHP_SELF'**]).
- Si indicamos el “action” del formulario así:

  action="<?= $_SERVER[**'PHP_SELF'**]; **?>"**

- Será la propia página del formulario la que procese los datos del mismo.

Verificar que el formulario se ha enviado

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

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

Validación del formulario

- Normalmente siempre debemos comprobar que los datos del formulario son correctos.
- Validaciones a realizar:
  - Los campos requeridos **no** deben quedar **vacíos** (función "**empty()**").
  - Eliminar los **espacios en blanco del principio y final** de los campos (función "**trim()**").
  - Todos los campos de entrada se deben **filtrar con "htmlspecialchars"** antes de mostrar el campo con echo o similar.
  - Los campos **email** deben tener el **formato esperado** (función "**filter_var($email, FILTER_VALIDATE_EMAIL**)").
  - Los campos **fecha** deben tener el **formato esperado**.

Validación del formulario: Fechas

- En PHP las fechas se almacenan como números.
- Se utiliza la clase **DateTime** para representarlas.
- Al igual que ocurre con las cadenas no tiene sentido estudiar las funciones relacionadas una a una.
- Las puedes consultar en: http://php.net/manual/es/ref.datetime.php