Request PHP. Diferencias, ventajas y problemas de GET, POST, REQUEST.

Ejercicios ejemplos.

RECUPERAR DATOS DE FORMULARIO. $_REQUEST

Cuando un usuario pulsa el botón enviar de un formulario, la información que contenían sus campos es enviada a una dirección URL desde donde tendremos que recuperarla para tratarla de alguna manera. Por ejemplo, si realiza una compra, tendremos que recuperar los datos para completar el proceso de pago. La información del formulario "viaja" almacenada en variables que podremos recuperar y utilizar mediante PHP. Una de las formas de recuperación consiste en usar $_REQUEST.

RECUPERAR VARIABLES CON REQUEST

REQUEST nos permite capturar variables enviadas desde formularios con los métodos GET o POST. Vamos a ver dos ejemplos de formularios (ejemplo1.html y ejemplo2.html), que en un caso se enviarán usando GET y en otro usando POST. Ambos formularios enviarán la información (action) a una página común desde donde recuperaremos los datos usando $_REQUEST.

El código de los archivos html sería el siguiente para ejemplo1.html y ejemplo2.html. Escribe el código en un editor de texto como Notepad++ y visualízalos en tu navegador:

```
<form name="formulario" method="get" action="ejemploRequest.php"
>
        Nombre: <input type="text" name="nombre" value="">
        <input type="submit" />
</form>
```

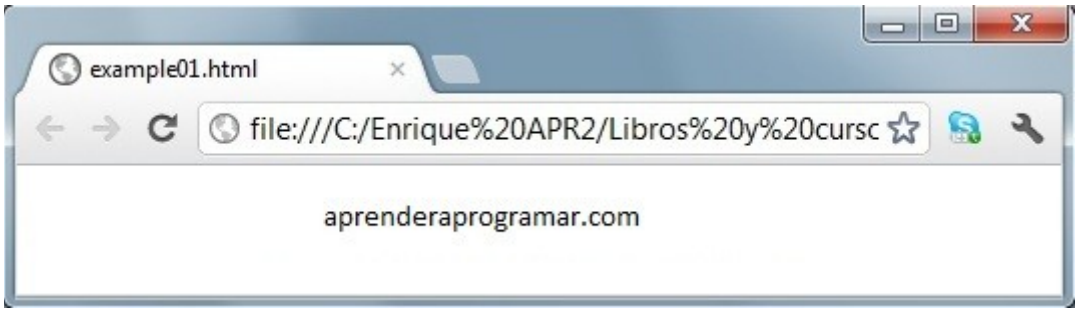
```
<form name="formulario" method="post" action="ejemploRequest.php
">
        Nombre: <input type="text" name="nombre" value="">
        <input type="submit" />
</form>
```

Como podemos observar, el ejemplo1.html envía los datos por GET mientras que el ejemplo2.html envía los datos por POST. Ahora bien, la acción o destino donde se enviarán los datos es la misma en los dos casos, la dirección ejemploRequest.php.

Escribe este código y guárdalo con un nombre de archivo como ejemploRequest.php. A continuación, sube el fichero al servidor en la misma carpeta donde subiste el ejemplo1.html y ejemplo2.html

```php
<?php //Ejemplo aprenderaprogramar.com
	$nombre = $_REQUEST['nombre'];
	echo $nombre;
?>
```

A continuación, observa el resultado obtenido al introducir el nombre tanto en el ejemplo1.html como en el ejemplo2.html, y verás que es el mismo.



Vamos a explicar el proceso que ha tenido lugar. El primer archivo es un documento HTML. Para ser más correctos, deberíamos haberlo escrito de esta manera:

```html
<!DOCTYPE html>
<html>
<head>
	<title>Ejemplo aprenderaprogramar.com</title>
	<meta charset="utf-8">
</head>
<body>
	<form name="formulario" method="get" action="ejemploRequest.p
	hp">
		Nombre: <input type="text" name="nombre" value="">
		<input type="submit" />
	</form>
</body>
</html>
```

Sin embargo, comprobamos que los navegadores tratan de interpretar y mostrar el resultado de un código HTML (o PHP) incluso cuando la sintaxis o la definición del documento no es del todo correcta. Esto debemos conocerlo, sin embargo recomendamos que siempre se trate de ser lo más correctos posibles a la hora de escribir código web.

Vemos que hemos definido un formulario en cuya cabecera hemos puesto que el método de envío de los datos va a ser el método GET y que la URL de destino va a ser ejemploRequest.php. A su vez, el formulario tiene un campo cuyo atributo name es “nombre”. Ese atributo define el nombre de la variable que vamos a poder recuperar en la URL de destino. En el archivo php hemos incluido la línea $nombre = $_REQUEST['nombre']; que significa que creamos una variable php denominada $nombre donde almacenamos la información del campo ‘nombre’ que recibe la URL a través del método GET proviniente del formulario. Si tuviéramos otros campos que hubiéramos definido como apellidos, teléfono, edad, el tratamiento sería similar. Por ejemplo:

```
$apellidos = $_REQUEST['apellidos'];

$telefono = $_REQUEST['celular'];

$edadPersona = $_REQUEST['edad'];
```

Fíjate en que una cosa es la variable en la que almacenamos la información recuperada, y otra cosa es el nombre del campo del formulario de donde proviene. Por ejemplo en $apellidos = $_REQUEST['apellidos']; coinciden el nombre de la variable que utilizamos con el nombre del campo del formulario. Sin embargo, en $telefono = $_REQUEST['celular']; no coinciden. En este caso, el campo que proviene del formulario se llama ‘celular’ mientras que la información que venga en ese campo la almacenamos en una variable a la que hemos llamado $telefono. Finalmente, en $edadPersona = $_REQUEST['edad']; estamos almacenando en una variable a la que hemos llamado $edadPersona la información proviniente de un campo del formulario denominado ‘edad’.

Con frecuencia los nombres de las variables y de los campos del formulario se hacen coincidir, pero en otras ocasiones no. Esto queda a elección del programador.

EJERCICIO RESUELTO Nº 1

Diseñar un formulario web que pida la altura y el diámetro de un cilindro en metros. Una vez el usuario introduzca los datos y pulse el botón calcular, deberá calcularse el volumen del cilindro y mostrarse el resultado en el navegador. El envío de datos debe hacerse por GET y la recuperación con REQUEST.

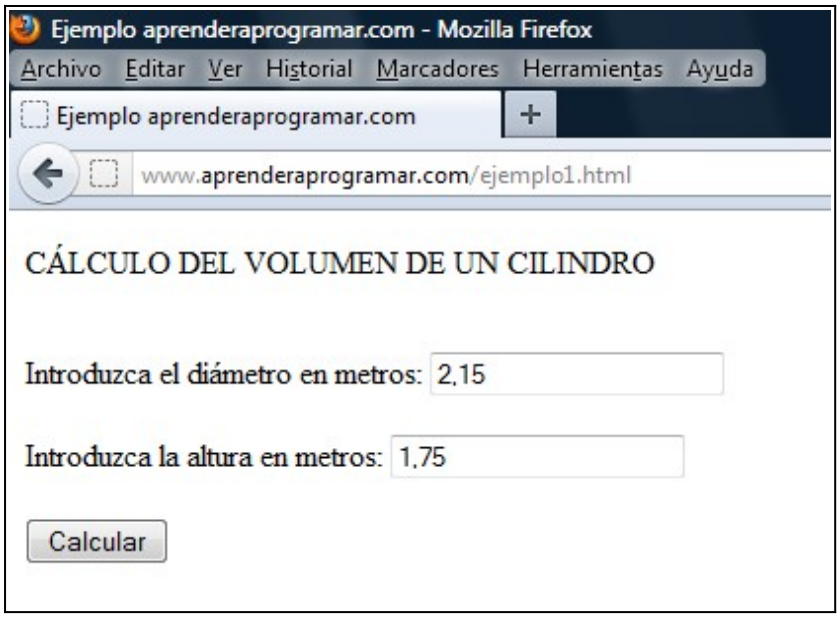
SOLUCIÓN

La solución esquematizada en pseudocódigo es la siguiente:

```
1. Inicio
2. Mostrar “Introduzca el diámetro, en metros” : Pedir D
3. Mostrar “Introduzca la altura, en metros” : Pedir H
4. R = D/2 : Pi = 3,141593
5. V = Pi * (R ^ 2) * H
6. Mostrar “El volumen del cilindro es de”, V, “metros cúbicos”
7. Fin
```

Esquematizar la solución en pseudocódigo es una buena idea antes de realizar la programación, pues nos permite definir conceptualmente cómo va a ser nuestro código antes de escribirlo. Es sobre todo adecuado para personas que se están iniciando en la programación. Crearemos el archivo html:

```
<!DOCTYPE html>
<html>
	<head>
	<title>Ejemplo aprenderaprogramar.com</title>
	<meta charset="utf-8">
	</head>
<body>
	<form name="formularioDatos" method="get" action="ejemploRequest1.p
	hp">
		<p> CÁLCULO DEL VOLUMEN DE UN CILINDRO </p>
		<br/>
		Introduzca el diámetro en
		metros: <input type="text" name="diam" value="">
		<br/> <br/>
		Introduzca la altura en
		metros: <input type="text" name="altu" value="">
		<br/> <br/>
		<input value="Calcular" type="submit" />
	</form>
</body>
</html>
```

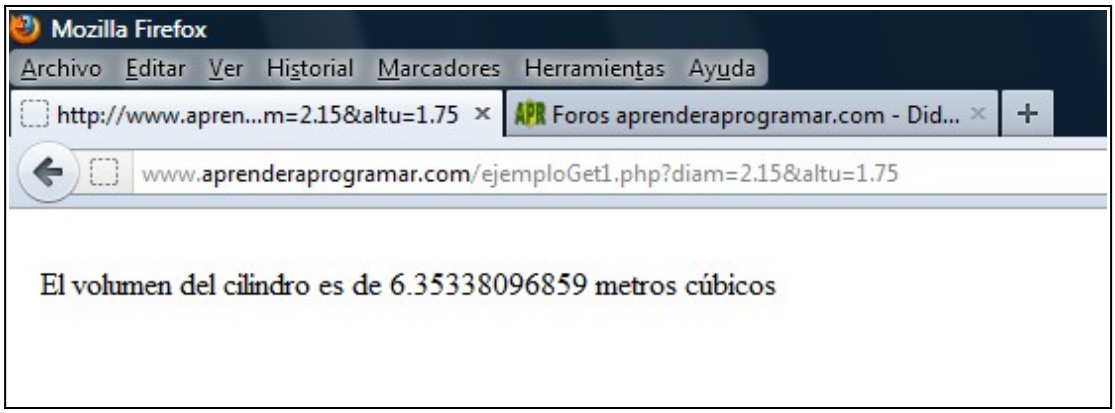


Por otro lado, crearemos el archivo php con el tratamiento de datos:

```
<?php //Ejemplo aprenderaprogramar.com
        $diametro = $_REQUEST['diam'];
        $altura = $_REQUEST['altu'];
        $radio = $diametro/2;
        $Pi = 3.141593;
        $volumen = $Pi*$radio*$radio*$altura;
        echo "<br/>   ; El volumen del cilindro es de". $volumen. "metros
        cúbicos";
?>
```

Fíjate que hemos escrito la potencia del radio como $radio * $radio. En otros lenguajes existe el operador de exponenciación, pero en php esta operación se tiene que realizar recurriendo a una función matemática. Esta función la estudiaremos en otro momento.

Finalmente obtenemos un resultado. Haz pruebas introduciendo como valores de diámetro y altura 2,15 y 1,75 en vez de 2.15 y 1.75. Posiblemente no obtengas un resultado adecuado si usas las comas, ya que en PHP el separador de la parte decimal de un número es el punto.



EJERCICIO RESUELTO Nº 2

Diseñar un desarrollo web simple con php que pida al usuario el precio de tres productos en tres establecimientos distintos denominados “Tienda 1”, “Tienda 2” y “Tienda 3”. Una vez se introduzca esta información se debe calcular y mostrar el precio medio del producto. El envío de datos debe hacerse por POST y la recuperación con REQUEST.

SOLUCIÓN

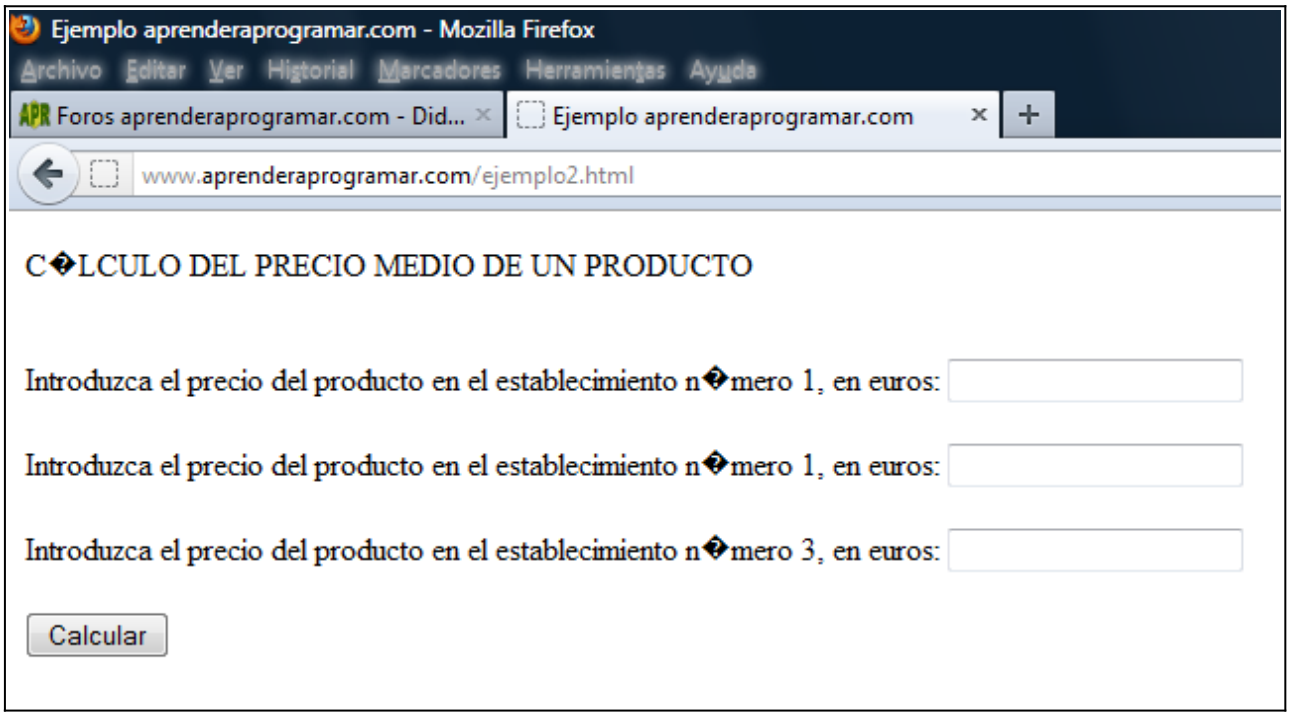
La solución esquematizada en pseudocódigo es la siguiente:

1. Inicio
2. Mostrar “Introduzca el precio del producto en el establecimiento número 1, en euros” : Pedir Precio1
3. Mostrar “Introduzca el precio del producto en el establecimiento número 2, en euros” : Pedir Precio2
4. Mostrar “Introduzca el precio del producto en el establecimiento número 3, en euros” : Pedir Precio3
5. Media = (Precio1 + Precio2 + Precio3) / 3
6. Mostrar “El precio medio del producto es”, Media, “euros”
7. Fin

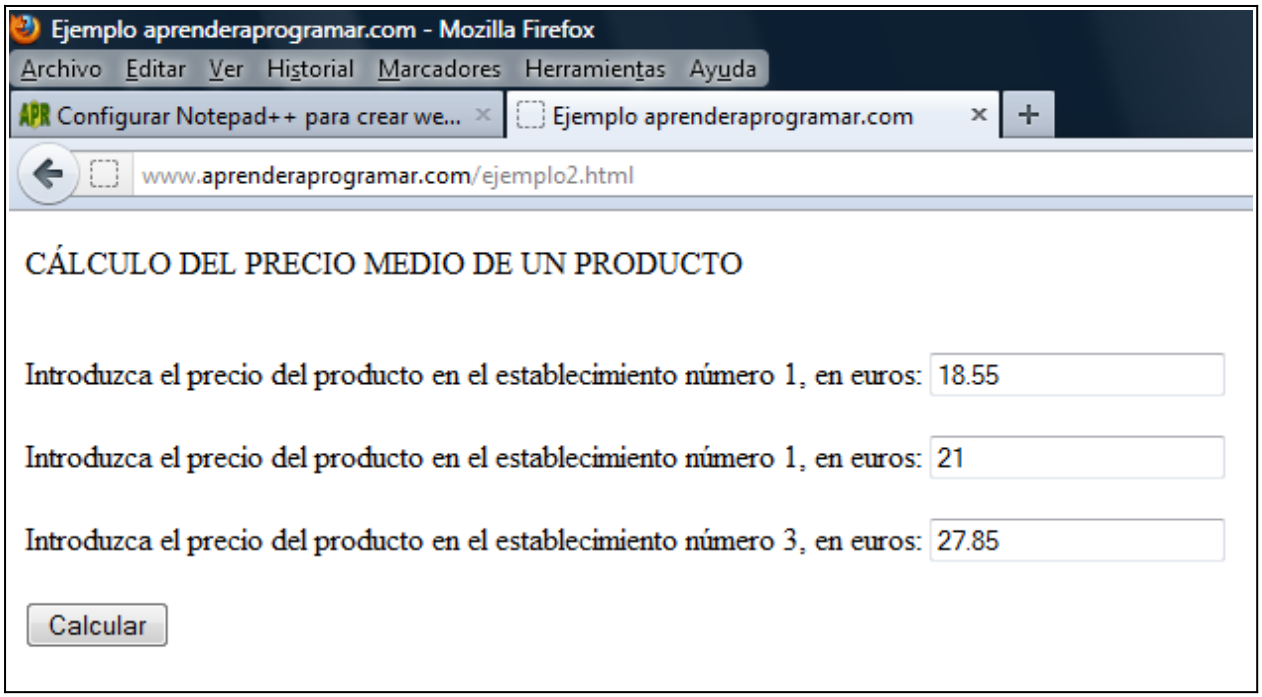
```
<!DOCTYPE html>
<html>
<head>
	<title>Ejemplo aprenderaprogramar.com</title>
	<meta charset="utf-8">
</head>
<body>
	<form name="formularioDatos" method="post" action="ejemploRequest2.php">
		<p> CÁLCULO DEL PRECIO MEDIO DE UN PRODUCTO </p>
		<br/>
		Introduzca el precio del producto en el establecimiento número 1, en euros: <input type="text" name="precio1" value="">
		<br/> <br/>
		Introduzca el precio del producto en el establecimiento número 1, en
```

```
            euros: <input type="text" name="precio2" value="">
            <br/> <br/>
            Introduzca el precio del producto en el establecimiento número 3, en
            euros: <input type="text" name="precio3" value="">
            <br/> <br/>
            <input value="Calcular" type="submit" />
        </form>
</body>
</html>
```

Es posible que durante el desarrollo del curso te encuentres visualizaciones de este tipo, donde podrás comprobar que los acentos o tildes no se ven bien.

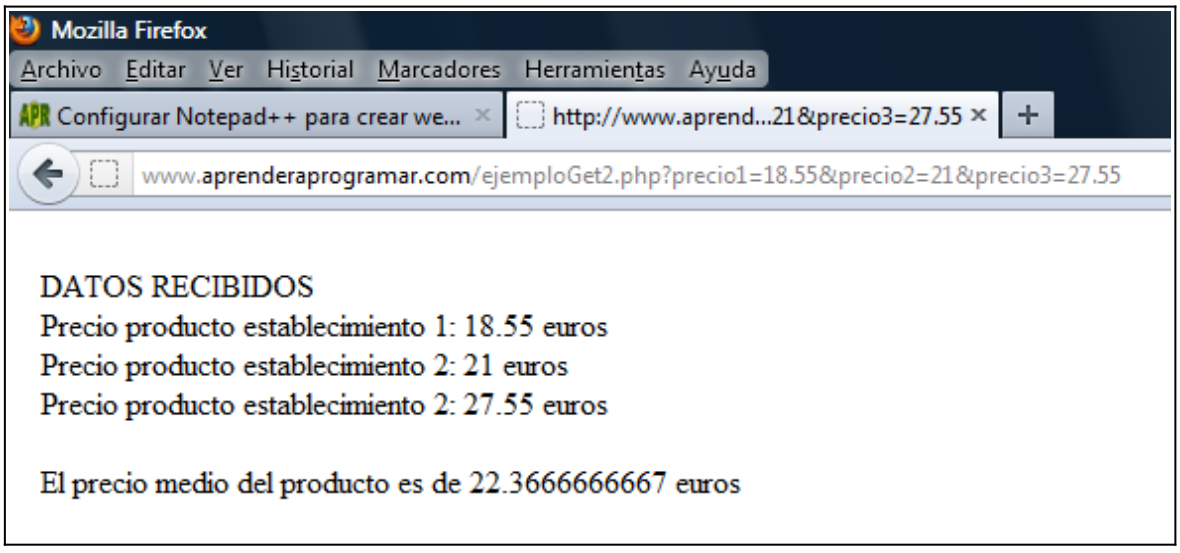


Recordarte que la solución a esto es, cuando estamos trabajando con Notepad++, elegir en el menú Formato la opción “Codificar en UTF-8 sin BOM”. En caso de que por error el archivo esté en otro formato, elige la opción “Convertir en UTF-8 sin BOM” para dejar correctamente la codificación del archivo. También puede ser necesario introducir la etiqueta: <meta charset="utf-8"> entre las etiquetas <head> ... </head> para lograr que la visualización sea correcta.



Por otro lado necesitamos el archivo php.

```php
<?php //Ejemplo aprenderaprogramar.com
$precio1 = $_REQUEST['precio1'];
$precio2 = $_ REQUEST ['precio2'];
$precio3 = $_ REQUEST ['precio3'];
$media = ($precio1+$precio2+$precio3)/3;
echo "<br/> & nbsp; DATOS RECIBIDOS";
echo "<br/>   ; Precio producto establecimiento 1: ". $precio1. " euros";
echo "<br/>   ; Precio producto establecimiento 2: ". $precio2. " euros";
echo "<br/>   ; Precio producto establecimiento 2: ". $precio3. " euros <br/>";
echo "<br/>   ; El precio medio del producto es de ". $media. " euros";
?>
```



EJERCICIO PROPUESTO

Diseñar un desarrollo web simple con PHP que dé respuesta a la necesidad que se plantea a continuación. Un operario de una fábrica recibe cada cierto tiempo un depósito cilíndrico de dimensiones variables, que debe llenar de aceite a través de una toma con cierto caudal disponible. Se desea crear una aplicación web que le indique cuánto tiempo transcurrirá hasta el llenado del depósito. El caudal disponible se considera estable para los tiempos que tardan los llenados de depósitos y lo facilita el propio operario, aportando el dato en litros por minuto.

ORIENTACIÓN PARA LA SOLUCIÓN

La solución esquematizada en pseudocódigo es la siguiente:

1. Inicio
2. Mostrar “Introduzca el caudal disponible en litros / minuto”: Pedir Q
3. Mostrar “Introduzca el diámetro del depósito, en metros” : Pedir D
4. Mostrar “Introduzca la altura del depósito, en metros” : Pedir H
5. Pi = 3,141593
6. R = D / 2
7. V = Pi * (R ^ 2) * H : Vlitros = V * 1000
8. tminutos = Vlitros / Q
9. Mostrar “El tiempo que transcurrirá hasta el llenado del depósito es de”, tminutos, “minutos”
10. Fin

Realiza el ejercicio y comprueba los resultados. Para que tengas una referencia, si el caudal disponible es de 125 litros por minuto y losvalores de diámetro y altura del cilindro son 2.15 y 1.75, el resultado que debes obtener es que el depósito tarda en llenarse será de aproximadamente 50 minutos.

RESUMEN DE LAS DIFERENCIAS ENTRE $GET, $POST Y $REQUEST

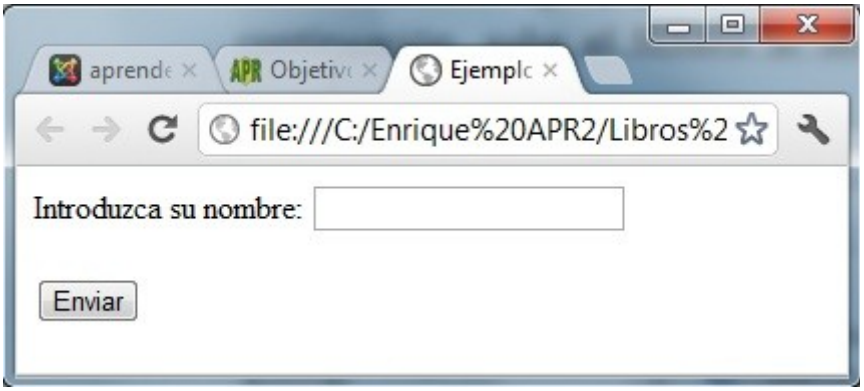
MÉTODO	CONCEPTO	OBSERVACIONES
GET	GET lleva los datos de forma "visible" al cliente (navegador web). El medio de envío es la URL. Para recoger los datos que llegan en la url se usa $_GET.	Los datos son visibles por la URL, por ejemplo: www.aprenderaprogramar.com/ action.php?nombre=pedro&apellidos1= gomez
POST	POST consiste en datos "ocultos" (porque el cliente no los ve) enviados por un formulario cuyo método de envío es post. Es ideal para formularios. Para recoger los datos que llegan por este método se usa $_POST.	La ventaja de usar POST es que estos datos no son visibles al usuario de la web. En el caso de usar get, el propio usuario podría modificar la URL escribiendo diferentes parámetros a los reales en su navegador, dando lugar a que la información tratada no sea la prevista.
REQUEST	No es un método de envío propiamente dicho, es decir, no podemos poner en un formulario HTML que el method sea REQUEST. Con la variable $_REQUEST recuperaremos los datos de los formularios enviados tanto por GET como por POST.	La ventaja principal de esta forma de recuperar los datos de un formulario es que no tenemos que saber con cual método fue enviado. La desventaja principal, como veremos en el siguiente ejemplo, es que no podremos diferenciar una variable enviada por GET o por POST.

Ahora nos podemos estar preguntado: ¿Qué pasaría si envío una variable nombre con un formulario por método post y además en la página de recuperación de datos también está definida dicha variable en la url y obtengo dicho variable con $_REQUEST? ¿Cuál me devolvería?

Pues bien, para contestar a esta cuestión planteemos el siguiente ejercicio.

Escribe este código y guárdalo con un nombre de archivo como ejemploPostGetRequest.html. A continuación, sube el fichero al servidor.

```
<!DOCTYPE html>
<html>
<head>
	<title>Ejemplo aprenderaprogramar.com</title>
	<meta charset="utf-8">
</head>
<body>
	<form name="formularioDatos" method="post" action="ejemploGetPostRequest.php?
	nombre=pepito">
		Introduzca su nombre: <input type="text" name="nombre" value="">
		<br/> <br/>
		<input value="Enviar" type="submit" />
	</form>
</body>
</html>
```



Introduciremos un nombre diferente a pepito, con lo que en la página de recuperación de datos, tendremos dos valores para 'nombre'. Y ahora comprobaremos cuál nos devuelve $_REQUEST['nombre']. Para ello escribe este código y guárdalo con un nombre de archivo como ejemploPostGetRequest.php. A continuación, sube el fichero al servidor en la misma carpeta donde subiste el ejemploPostGetRequest.html

```
<?php //Ejemplo aprenderaprogramar.com
echo "nombre = " . $_REQUEST['nombre'];
?>
```

Como podemos observar, el valor que toma $_REQUEST es primero el valor enviado por POST y después el enviado por GET (si no viniera en POST). Luego, como hemos observado, $_REQUEST da prioridad a los valores enviados por POST antes que a los enviados por GET.

Recuerda también que una URL es modificable por el usuario fácilmente. Por ejemplo si un formulario envía datos así: www.aprenderaprogramar.com/action.php?nombre=pedro, el usuario podría modificar la URL escribiendo directamente en su navegador www.aprenderaprogramar.com/action.php?nombre=barack. Esto puede tener cierta importancia, sobre todo en el caso de transmisión de datos relativos a precios, ya que si el precio se transmite por get será más fácilmente manipulable por el usuario (algo indeseable). Get tiene la ventaja de que los datos son visibles y más fáciles de seguir y localizar, y el inconveniente de que puede ser manipulado más fácilmente.