PROGRAMACIÓN WEB EN ENTORNO SERVIDOR UD-5

FORMULARIOS

ÍNDICE

[FORMULARIOS 3](#_Toc146895114)

[MÉTODOS DE PETICIÓN 3](#_Toc146895115)

[VALIDACIÓN 4](#_Toc146895116)

[PARÁMETRO MULTIVALOR 4](#_Toc146895117)

[RELLENADO DE FORMULARIOS 5](#_Toc146895118)

[CABECERAS DE RESPUESTA 6](#_Toc146895119)

[GESTIÓN DEL ESTADO 6](#_Toc146895120)

# FORMULARIOS

Los datos se envían vía URL con el formato var1=valor1&var2=valor2…. Por ejemplo: ejemplo.php?nombre=Pepe+apellido1=Perez

Se divide en dos pasos:

1. Generar un formulario con action='archivo.php' method='GET'
2. En el archivo .php leer los datos con $\_GET['nombreVar']

Vamos a separar siempre que podamos el código HTML del de PHP. Por ejemplo, el formulario lo colocamos en ejemplo.html y recogemos sus datos en ejemplo.php. Si queremos dejarlo en un solo archivo, preguntaremos si existe la variable de recogida, para ver si se ha hecho la llamada con un if(isset($\_GET(“dato”)).

# MÉTODOS DE PETICIÓN

Existe una gran cantidad de verbos HTTP, o métodos de petición, dentro del desarrollo web:

* GET
* POST
* PUT
* DELETE
* HEAD
* CONNECT
* OPTIONS
* TRACE
* PATH

Según el método que usemos marcaremos el uso que vamos a dar a la petición. El destinatario en función del método indicado tratará los datos de forma distinta.

A la hora de enviar un formulario, debemos tener claro cuando usar GET o POST

* GET: los parámetros se pasan en la URL
  + <2048 caracteres, sólo ASCII.
  + Permite almacenar la dirección completa (marcador / historial)
  + Idempotente: dos llamadas con los mismos datos siempre deben dar el mismo resultado.
  + El navegador puede cachear las llamadas.
* POST: parámetros ocultos (no encriptados)
  + Sin límite de datos, permite datos binarios.
  + No se pueden cachear
  + No idempotente → actualizar la BBDD

Así pues, para recoger los datos accederemos al array dependiendo del método del formulario que nos ha invocado:

Texto

Descripción generada automáticamente

Podemos ver con que método se ha hecho la petición usando la variable $\_SERVER



Si en action del de la etiqueta <form> de html no indicamos nada, al hacer un envío la petición se hará a la misma página donde está ubicado el formulario.

# VALIDACIÓN

Respecto a la validación, es conveniente siempre hacer validación doble:

* En el cliente mediante JS.
* En servidor, antes de llamar a negocio, es conveniente volver a validar los datos.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

# PARÁMETRO MULTIVALOR

Existen elementos HTML que envían varios valores:

* select multiple
* checkbox

Para recoger los datos, el nombre del elemento debe ser un array.

También nos podemos apoyar en las funciones serialize() y unserialize(). Nos convertirá los arrays en texto o texto en arrays. Estas funciones nos permiten mantener estructuras avanzadas como los diccionarios. Es inseguro serializar un string proviene de un cliente. Se práctica porque es cómodo y rápido para recuperar array u objetos, aunque lamentablemente debes evitarlo ya que es posible manipular el texto recibido para inyectar código malicioso en el backend. En su lugar recibe un JSON, ya que sus campos son muy limitados, y construye manualmente todos los elementos (se verá en temas posteriores)

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

# RELLENADO DE FORMULARIOS

Cada vez que pulsas en un submit la página se refresca (se realiza una petición) y con ello se reinicia el formulario. ¿Qué pasa si me he equivocado en algún campo? El usuario tendría que volver a rellenarlo. Un truco para solucionarlo sería comprobar si existe el dato, y si es así rellenar su value.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# CABECERAS DE RESPUESTA

Debe ser lo primero a devolver. Se devuelven mediante la función header(cadena). Mediante las cabeceras podemos configurar el tipo de contenido, tiempo de expiración, redireccionar el navegador, especificar errores HTTP, etc.



Se puede comprobar en las herramientas del desarrollador de los navegadores web mediante Developer Tools → Network → Headers.

Es muy común configurar las cabeceras para evitar consultas a la caché o provocar su renovación.

Texto

Descripción generada automáticamente

# GESTIÓN DEL ESTADO

HTTP es un protocolo stateless, sin estado. Por ello, se simula el estado mediante el uso de campos ocultos, cookies, tokens o la sesión. El estado es necesario para procesos tales como el carrito de la compra, operaciones asociadas a un usuario, etc...

Por ahora trabajaremos con campos ocultos hasta que veamos cookies en un tema posterior.

Texto

Descripción generada automáticamente