UT 3: FORMULARIOS

1. **Envio de datos desde un formulario**
2. Métodos de envío
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. Subir un fichero al servidor
6. Validar formularios
7. Uso de arrays en formularios
8. Redirigir a otra página

1. ENVÍO DE DATOS DESDE UN FORMULARIO

Los formularios son el elemento de las páginas web que permiten recoger valores del usuario y enviarlos a una dirección URL para que se les procese. En nuestro caso la URL será una página PHP que recogerá los valores introducidos en el formulario y actuará en consecuencia.

HTML dispone de numerosos controles para almacenar información que se pide al usuario: cuadros de texto, cuadros numéricos, botones de radio, cuadros combinados, etc.

Estos elementos tienen un atributo **name** en el que se indica el nombre de la variable que se envía al servidor y un atributo **value** que indica el valor que ha tomado esa variable, excepto en el type="text".

1. ENVIO DE DATOS DESDE UN FORMULARIO

INPUT TYPE TEXT

```
<label for="nombre">Nombre:</label>
<input type="text" id="nombre" name="nombre">
```

INPUT TYPE RADIO (Recordad que todas las opciones tienen el mismo name)

```
<label for="colorRojo">Rojo:</label>
<input type="radio" id="colorRojo" name="color" value="rojo">
<label for="colorVere">Verde:</label>
<input type="radio" id="colorVerde" name="color" value="verde">
```

1. ENVIO DE DATOS DESDE UN FORMULARIO

INPUT TYPE CHECKBOX

```
<label><input type="checkbox" name="bici" value="bici">
  Bicicleta</label><br>
<label><input type="checkbox" name="andar" value="andar">
  Andar</label><br>
```

INPUT TYPE HIDDEN

```
<input type="hidden" name="alumId" value="3487">
```

1. ENVIO DE DATOS DESDE UN FORMULARIO

INPUT TYPE SUBMIT

```
<input type="submit" name="enviar" value="Enviar">
```

SELECT

```
<select name="estilo">
    <option value="rock">Rock</option>
    <option value="pop">Pop</option>
</select>
```

1. ENVIO DE DATOS DESDE UN FORMULARIO

Todos los controles del formulario deben estar dentro de un elemento form, el cual nos permite configurar estos atributos:

- **method:** Puede ser get o post y se refiere a la forma en la que se pasan los datos de formulario. Por defecto se toma corno método, get.
- **action:** Que indica la URL a la dirección del servicio que recoge los datos.

```
<form action="/procesar.php" method="post">
</form>
```

UT 3: FORMULARIOS

1. Envio de datos desde un formulario
2. **Métodos de envío y recepción de datos**
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. Subir un fichero al servidor
6. Validar formularios
7. Uso de arrays en formularios
8. Redirigir a otra página

2. MÉTODOS DE ENVÍO Y RECEPCIÓN DE DATOS

PHP permite recibir los datos pasados por los parámetros usando dos variables, que son en realidad dos *arrays* que contendrán todos los valores del formulario. Los arrays se llaman **$_GET** y **$_POST,** cada uno de ellos dedicado a almacenar los datos enviados con el método correspondiente especificado en el **method** del form.

Hay otro array, de uso menos recomendable, llamado **$_REQUEST** que recoge los valores sin importar si se han pasado por GET o por POST.

UT 3: FORMULARIOS

1. Envio de datos desde un formulario
2. **Métodos de envío y recepción de datos**
    1. **Paso de datos mediante get**
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. Subir un fichero al servidor
6. Validar formularios
7. Uso de arrays en formularios
8. Redirigir a otra página

2.1 PASO DE DATOS MEDIANTE GET

El método GET lo que hace es añadir a la **URL** destinataria del formulario los nombres y valores recogidos en el formulario.

Fichero uno.php envia los datos a dos.php

```
<form action="dos.php" method="get">
    <label for="nombre">Escriba su nombre</label>
    < input type="text" name="nombre" /> <br />
    <label for="nornbre">Escriba sus teléfonos: </label>

    <input type="text" name="telefono1" maxlength="9" pattern="[0-9]{9}" />
    <input type="text" name="telefono2" maxlength="9" pattern="[0-9]{9}" />
    <input type="submi t" value="enviar"/> <br />
</form>
```

2.1 PASO DE DATOS MEDIANTE GET

La forma de una URL tipo GET es: http://urlpágina?nornbrer=valorr&nombre2=valor2&.....

En la URL, los caracteres no ASCII, por ejemplo, letras con tildes, se recodifican usando el valor de la letra en UTF-8 y añadiendo antes el %.

Ej: el carácter é en UTF8 se convierte en C3 A9, en la URL se representa %C3%A9

https://www.w3schools.com/tags/ref_urlencode.ASP

Entonces, cuando se envía la URL que contiene los pares nombre/valor rellenados en el formulario está a la vista por lo que se debe pasar de esta forma contraseñas y otros datos críticos.

Por otro lado, la depuración de nuestros programas es más cómoda si usamos GET, porque vemos lo que realmente llega del formulario en la propia URL sin tener que acudir a otras herramientas.

2.1 PASO DE DATOS MEDIANTE GET

Al enviar los datos se reciben en un array ($_GET).

Ejemplo:

```
<?php
   print_r($_GET);
?>
```

Mostrará

```
 Array( "nombre" => XXXX, "telefono1" => 666666666, "telefono2"
=> 999999999)
```

2.1 PASO DE DATOS MEDIANTE GET

Para acceder a un elemento podemos hacerlo así:

```
$_GET["name"]
```

Si intentamos acceder a un nombre que no hemos recibido, entonces nos daría un error: `Notice : Undefined index.`

Para evitar este error, antes de acceder al nombre debemos comprobar si existe mediante la función isset:

```
if(isset($_GET ["nombre" ] ) {
}
```

UT 3: FORMULARIOS

1. Envio de datos desde un formulario
2. **Métodos de envío y recepción de datos**
    1. Paso de datos mediante get
    2. **Paso de datos mediante post**
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. Subir un fichero al servidor
6. Validar formularios
7. Uso de arrays en formularios
8. Redirigir a otra página

2.2 PASO DE DATOS MEDIANTE POST

Los formularios se crean igual en ambos casos, pero el atributo **method** de la etiqueta form toma el valor POST.

Mediante POST, los nombres y valores del formulario no viajan en la URL, sino que se colocan en la cabecera del paquete http; por lo que les dotamos de una menor visibilidad. Naturalmente podemos verlos utilizando herramientas que nos permitan examinar las cabeceras http del paquete que se envía por la red . Hoy en día es sencillo, ya que desde los propios navegadores disponemos de herramientas para ello.

```
if(isset($_POST ["name" ] ) {
}
```

UT 3: FORMULARIOS

1. Envio de datos desde un formulario
2. Métodos de envío
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. **Usar la misma página para el formulario y la recepción**
4. Codificación
5. Subir un fichero al servidor
6. Validar formularios
7. Uso de arrays en formularios
8. Redirigir a otra página

3. USAR LA MISMA PÁGINA PARA EL FORMULARIO Y LA RECEPCIÓN

Una forma habitual de trabajar con formularios en PHP es utilizar un único programa que procese el formulario o lo muestre según haya sido o no enviado, respectivamente.

Para ello el action del form es una cadena vacia: `action=""`.

```
if (se ha enviado el formulario) {
   Procesar formulario
} else {
   Mostrar formulario
}
```

3. USAR LA MISMA PÁGINA PARA EL FORMULARIO Y LA RECEPCIÓN

Para saber si se ha enviado el formulario, una forma es usar la variable correspondiente al botón de envío.
Por ejemplo, si este botón es así:

```
<input type="submit" name="enviar" value="Enviar">
```

entonces la condición para saber si el formulario se ha mandado sería así

```
if (isset($_POST[‘enviar’])) {

}
```

EJEMPLO

```
<!DOCTYPE html>
<html>
   <head>
     <meta charset="UTF-8">
     <title>Respuesta</title>
   </head>
<body>
 <?php
  if(isset($_POST["enviar"])) {
     if(isset($_POST["nombre"])){
        echo "Nombre:" . $_POST["nombre"];
     }
  } else {
?>
   <form action="" method="post">
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre">
    <input type="submit" name="enviar" value="Enviar">
   </form>
<?php
  }
?>
</body>
</html>
```

UT 3: FORMULARIOS

1. Envio de datos desde un formulario
2. Métodos de envío
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. **Codificación**
5. Subir un fichero al servidor
6. Validar formularios
7. Uso de arrays en formularios
8. Redirigir a otra página

4. CODIFICACIÓN

UTF-8 es un alfabeto con el que se puede representar casi todos los caracteres de todos los idiomas del mundo. Cada carácter, ya sea una letra del alfabeto inglés, un kanji japonés o un emoji, tiene un código único en UTF-8.

Buenas prácticas para trabajar con UTF8:

Cuando se envían datos a través de un formulario HTML, esos datos se envían utilizando la codificación especificada en el formulario. Se deben de configurar para usar UTF-8.

```
<form accept-charset="UTF-8">
```

UT 3: FORMULARIOS

1. Envio de datos desde un formulario
2. Métodos de envío
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. **Subir un fichero al servidor**
6. Validar formularios
7. Uso de arrays en formularios
8. Redirigir a otra página

5. SUBIR UN FICHERO AL SERVIDOR

Para subir un fichero al servidor se utiliza el elemento de entrada FILE

Hay que tener en cuenta una serie de consideraciones importantes:

- El elemento FORM debe tener el atributo **ENCTYPE="multipart/form-data"** y debe ser por método POST
- El elemento de HTML es de tipo file

```
<input type="file" name="fichero">
```

5. SUBIR UN FICHERO AL SERVIDOR

- El fichero tiene un límite en cuanto a su tamaño. Este límite se fija de dos formas diferentes:
  - En el fichero de configuración php.ini
  - En el propio formulario.

En el propio formulario con el atributo **size**, que indica el tamaño en bytes.

```
<input type="file" id="archivo" name="archivo" accept=".pdf,
   .doc,.docx" size="2097152">
```

5. SUBIR UN FICHERO AL SERVIDOR

En el fichero c:\windows\php.ini:

- **upload_max_filesize**: Tamaño máximo del archivo a recibir por el servidor.
- **file_uploads**: On para activar el recibo de ficheros.
- **upload_tmp_dir**: Directorio donde se alojan los ficheros recibidos de forma temporal

5. SUBIR UN FICHERO AL SERVIDOR

Una vez enviado el formulario, la información se almacena en **$_FILES** (anteriormente se utilizaba $HTTP_POST_FILES):

```
<input type="file" name="archivo">
```

- $_FILES['archivo']['name']: Nombre original del fichero en la máquina cliente
- $_FILES['archivo']['type']: Tipo mime del fichero. Por ejemplo, "image/gif"
- $_FILES['archivo']['size']: Tamaño en bytes del fichero subido
- $_FILES['archivo']['tmp_name']: Nombre del fichero temporal en el que se almacena el fichero subido en el servidor
- $_FILES['archivo']['error']: Código de error asociado al fichero subido.

5. SUBIR UN FICHERO AL SERVIDOR

El fichero se recibe en el directorio temporal, si se especifica en el archivo de configuración.

Para copiar el archivo, hay dos opciones:

- Función **copy(origen, destino)**
- Función **move_uploaded_file**(origen, destino): Función más segura ya que comprueba:
  - Fichero enviado con método POST
  - Si la opción **safe mode=on** comprueba que la transferencia ha sido realizada por el mismo usuario que realizo la petición de ejecución de script.
  - **destino** debe ser ruta absoluta.

5. SUBIR UN FICHERO AL SERVIDOR

- Consideraciones
  - Debe darse al fichero un nombre que evite coincidencias con ficheros ya subidos. Por ello, y como norma general, debe **descartarse el nombre original** del fichero y crear uno nuevo que sea único, con el método time().
  - El fichero subido se almacena en un directorio temporal y hemos de moverlo al directorio de destino usando la función move_uploaded_file().
  - La carpeta donde queremos subir ese archivo debe tener **permisos** de lectura y escritura.

```
chmod 777 carpeta_subida
```

5. SUBIR UN FICHERO AL SERVIDOR

- Procedimiento:
  - si se ha subido correctamente el fichero:
    - Asignar un nombre al fichero
    - Mover el fichero a su ubicación definitiva
  - si no:
    - Mostrar un mensaje de error

EJEMPLO DE FORMULARIO PARA MANDAR UN FICHERO

```
<form action="procesar.php" method="post"
enctype="multipart/form-data">

<label for="imagen">Selecciona una imagen:</label>
<input type="file" name="imagen">


<input type="submit" value="Subir Imagen">
</form>
```

EJEMPLO DE CÓDIGO PHP PARA RECIBIR UN FICHERO (PROCESAR.PHP)

```php
<?php
   if (isset($_FILES['imagen']) &&
is_uploaded_file($_FILES['imagen']['tmp_name'])) {
      //se ha enviado un archivo con ese nombre y se ha cargado
correctamente
      $nombreTemporal = $_FILES['imagen']['tmp_name'];
      if (is_file($nombreTemporal)) {
         // Se mueve el archivo a su ubicación deseada
         $directorioDestino = "img/";
         $nombreArchivo = time()."-".$_FILES['imagen']['name'];
         $rutaCompleta = $directorioDestino . $nombreArchivo;
         if (move_uploaded_file($nombreTemporal,
$rutaCompleta)) {
             echo "El archivo se ha subido y movido con
éxito.";
         } else {
             echo "Error al mover el archivo.";
         }
      } else {
         echo "Error: El archivo no es válido.";
      }
    } else {
       echo "Error: No se recibió un archivo válido.";
    }
?>
```

UT 5: FORMULARIOS

1. Envio de datos desde un formulario
2. Métodos de envío
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. Subir un fichero al servidor
6. **Validar formularios**
7. Uso de arrays en formularios
8. Redirigir a otra página

6. VALIDAR FORMULARIOS

Toda la información proveniente de un formulario debe considerarse por norma como contaminada, y hay que validarla antes de darla por buena y procesarla.

Para ello conviene comprobar que:

- los elementos existen con la función **`isset()`.**
- Los elementos existen y tienen valor: con la función **`empty()`.**

  Hay que tener cuidado con el uso de empty porque el valor 0 en un entero, cadena vacía... se considera que la variable está vacía.

Además, se puede hacer comprobaciones más específicas sobre un número que debe estar dentro de un rango, por ejemplo.

6. VALIDAR FORMULARIOS

- Una manera de gestionarlo sería mostrar los errores que ha habido en una página específica.
- Otra manera de gestionarlo es mostrar los errores sobre el propio formulario para facilitar su corrección. Si se ha enviado el formulario:

```
if(se ha enviado el formulario) {
    if(hay errores) {
      Mostrar formulario con errores
    }else{
      Procesar formulario
    }
} else {
   Mostrar formulario
}
```

EJEMPLO DE VALIDAR FORMULARIO

```php
<?php
// Variables para almacenar los errores
$errores = array();

// Verifica si se ha enviado el formulario
if(isset($_POST["enviar"])) {
  // Validamos el campo llamado "nombre"
  if (empty($_POST["nombre"])) {
      $errores[] = "El campo 'Nombre' es obligatorio.";
  }

  // Verifica si hay errores
  if (count($errores) > 0) {
    echo "<h2>Formulario con Errores:</h2>";
    echo "<ul>";
    foreach ($errores as $error) {
      echo "<li>$error</li>";
    }
    echo "</ul>";
    // Mostrar el formulario
    include("formulario.php");
  } else {
     // Procesar el formulario si no hay errores
  }
} else {
   // Mostrar el formulario
   include("formulario.php");
}
?>
```

UT 5: FORMULARIOS

1. Envio de datos desde un formulario
2. Métodos de envío
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. Subir un fichero al servidor
6. Validar formularios
7. **Uso de arrays en formularios**
8. Redirigir a otra página

7. USO DE ARRAYS EN FORMULARIOS

Una de las virtudes de un array, es su capacidad de **recoger** en una **sola variable**, valores procedentes de **diferentes controles** en un formulario. La manera de hacerlo es muy sencilla, basta con indicar un nombre de array como atributo name en los elementos del formulario. Para indicar que ese nombre es de un array, se añaden al nombre la apertura y cierre de corchetes [].

EJEMPLO 1 DE FORMULARIO CON INPUT CHECKBOX

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulario</title>
</head>
<body>
    <h2>Selecciona tus colores favoritos:</h2>
    <form method="post" action="procesar.php">
        <label><input type="checkbox"
name="colores[]" value="rojo"> Rojo</label><br>
        <label><input type="checkbox"
name="colores[]" value="verde">
Verde</label><br>
        <label><input type="checkbox"
name="colores[]" value="azul"> Azul</label><br>
        <label><input type="checkbox"
name="colores[]" value="amarillo">
Amarillo</label><br>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

EJEMPLO 1 DE PROCESAR.PHP

```
<!DOCTYPE html>
<html>
<head>
    <title>Resultado de la Selección de Colores</title>
</head>
<body>
    <h2>Colores Favoritos Seleccionados:</h2>
    <?php
    if (isset($_POST['colores']) &&
!empty($_POST['colores'])) {
        $coloresSeleccionados = $_POST['colores'];
        echo "<ul>";
        $numColores = count($coloresSeleccionados);
        for ($i = 0; $i < $numColores; $i++) {
            $color = $coloresSeleccionados[$i];
            echo "<li>$color</li>";
        }
        echo "</ul>";
    } else {
        echo "No has seleccionado ningún color
favorito.";
    }
    ?>
</body>
</html>
```

EJEMPLO 1 DE FORMULARIO CON INPUT CHECKBOX Y ARRAYS ASOCIATIVOS

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulario</title>
</head>
<body>
    <h2>Selecciona tus colores favoritos:</h2>
    <form method="post" action="procesar.php">
        <label><input type="checkbox"
name="colores['Color rojo']" value="rojo">
Rojo</label><br>
        <label><input type="checkbox"
name="colores['Color verde']" value="verde">
Verde</label><br>
        <label><input type="checkbox"
name="colores['Color verde']" value="azul">
Azul</label><br>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

EJEMPLO 1 DE PROCESAR.PHP

```
<!DOCTYPE html>
<html>
<head>
    <title>Resultado de la Selección de Colores</title>
</head>
<body>
    <h2>Colores Favoritos Seleccionados:</h2>
    <?php
    if (isset($_POST['colores']) &&
!empty($_POST['colores'])) {
        $colores = $_POST['colores'];
        echo "<ul>";
        foreach($colores as $nombre=>$valor){ {
           echo "<li>$valor</li>";
        }
        echo "</ul>";
    } else {
        echo "No has seleccionado ningún color
favorito.";
    }
    ?>
</body>
</html>
```

UT 5: FORMULARIOS

1. Envio de datos desde un formulario
2. Métodos de envío
    1. Paso de datos mediante get
    2. Paso de datos mediante post
3. Usar la misma página para el formulario y la recepción
4. Codificación
5. Subir un fichero al servidor
6. Validar formularios
7. Uso de arrays en formularios
8. **Redirigir a otra página**

8. REDIRIGIR A OTRA PAGINA

Es muy habitual al hacer aplicaciones web que ante una determinada condición, deseemos cargar otra página diferente a la que estamos. Para ello se utiliza la función header:

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