1. Tomas de decisión

- Denominadas sentencias condicionales
- **if /elseif /else**: definir una expresión para ejecutar o no la sentencia o conjunto de sentencias siguientes

```
<?php
    if ($a<$b)
        print "a es menor que b";
    elseif ($a>$b)
        print "a es mayor que b";
    else
        print "a es igual a b";
?>
```

- No es necesario poner {} porque solo ejecuta una sentencia en cada caso
- **switch**: similar a enlazar varias sentencias if comparando una misma variable con diferentes valores

```
<?php
    switch ($a){
        case 0:
            print " a vale 0";
            break;
        case 1:
            print " a vale 1";
```

```
            break;
        default:
            print "a no vale 0 ni 1";
    }
?>
```

2. Bucles

Son sentencias repetitivas:

- while
- do ... while
- for
- **while**:define un bucle que se ejecuta mientras se cumpla una expresión. La expresión se evalúa antes de comenzar cada ejecución del bucle.

```
<?php
    $a=1;
    while ($a <8)
        $a+=3;
    print $a // el valor obenido es 10

?>
```

- No es necesario poner {} porque solo ejecuta una sentencia
- **do ... while**: similar al bucle while, pero la expresión se evalúa al final, con lo cúal se asegura que la sentencia/as del bucle se ejecutan al menos una vez.

```
<?php
    $a=1;
    do
        $a -=3;
    while ($a >10)
    print $a; // el valor obtenido es -2
?>
```

- No es necesario poner {} porque solo ejecuta una sentencia
- **for**:compuesto por tres expresiones: sintaxis

```
    for(expr1;expr2;expr3){
        sentencia o conjunto de sentencias
```

```
    }
```

- expr1: se ejecuta una vez al comienzo del bucle.
- expr2: se evalúa para saber si se debe ejecutar o no la/as sentencia/as
- expr3: se ejecuta tras ejecutar todas las sentencias del bucle

```php
<?php
    for ($a=5;$a<10;$a+=3){
        print $a // se muestran los valores 5 y 8
        print "<br/>";
    }
?>
```

> Hoja03_PHP_01

3. Funciones

- Permiten **asociar una etiqueta** (el nombre de la función) con un bloque de código a ejecutar.
- Al usar funciones estamos ayudando a **estructurar mejor el código**.
- Las funciones permiten crear variables locales que no serán visibles fuera del cuerpo de las mismas.

3.1 Inclusión de las funciones en ficheros externos

En ocasiones resulta más cómodo agrupar las funciones en ficheros externos al que se hará referencia.

Formas de incorporar ficheros externos:

- **include**: evalúa el contenido del fichero que se indica y lo incluye como parte del fichero actual en el punto de realización de la llamada. Se puede indicar la ruta de forma absoluta o de forma relativa.
- **include_once**: si por equivocación incluyes más de una vez un mismo fichero, lo normal es que obtengas algún tipo de error (por ejemplo, al repetir una definición de una función)
- **require**: si el fichero que queremos incluir no se encuentra, include da un aviso y continua la ejecución del guión. La diferencia más importante al usar require es que en ese caso, cuando no se puede incluir el fichero, se **detiene** la ejecución del guión.
- **require_once**: es la combinación de las dos anteriores.

Ejemplo:

```php
<?php
   include 'funciones.inc.php';
   print fecha();
?>
```

3.2 Creación y ejecución de funciones

Para crear tus propias funciones se usa la palabra **function**

```
<?php
   function precio_con_iva(){
        global $precio;
        $precio_iva=$precio*1.21;
        print "el precio con IVA es ".$precio_iva;
   }
   $precio=10;
   precio_con_iva();
?>
```

No es necesario definir las funciones antes de usarlas excepto cuando están definidas condicionalmente:

```
<?php
    $iva=true;
    $precio =10;
    precio_con_iva();// Da error, pues aquí aún no está definida
    if ($iva){
        function precio_con_iva(){
            global $precio;
            $precio_iva=$precio*1.21;
            print "el precio con IVA es ".$precio_iva;
        }
    }
    precio_con_iva(); // aquí ya no da error
?>
```

3.3 Argumentos en las funciones

- Siempre es mejor utilizar **argumentos o parámetros al hacer la llamada**. Además, en lugar de mostrar el resultado en pantalla o guardar el resultado en una variable global, las funciones pueden **devolver un valor usando la sentencia return**.
- Cuando en una función se encuentra una sentencia return, termina su procesamiento y devuelve el valor que se indica

```
<?php
    function precio_con_iva($precio){
        return $precio*1.21;
    }
    $precio=10;
    $precio_iva=precio_con_iva($precio);
    print "el precio con IVA es ".$precio_iva;
?>
```

- Al definir la función, puedes indicar **valores por defecto** para los argumentos, de forma que cuando hagas una llamada a la función puedes no indicar el valor de un argumento; en este caso se toma el valor por defecto indicado.

```
<?php
    function precio_con_iva($precio,$iva=0.21){
        return $precio*(1+$iva);
    }
    $precio=10;
    $precio_iva=precio_con_iva($precio);
    print "el precio con IVA es ".$precio_iva;
?>
```

- Los argumentos anteriores se pasaban **por valor**. Esto es, cualquier cambio que se haga dentro de la función a los valores de los argumento no se reflejará fuera de la función.
- Si quieres que esto ocurra debes definir el parámetro para que su valor se pase **por referencia**, añadiendo el símbolo **& antes de su nombre**.

```
<?php
    function precio_con_iva(&$precio,$iva=0.21){
        return $precio*(1+$iva);
    }
    $precio=10;
    precio_con_iva($precio);
    print "el precio con IVA es ".$precio."<br/>";
?>
```

> Hoja03_PHP_02

4. Tipos de datos compuestos

Un tipo de datos compuesto es aquel que te permite almacenar más de un valor. En PHP puedes utilizar dos tipos de datos compuestos: el array y el objeto.

5. Arrays y cadenas

Un array es un tipo de datos que nos permite almacenar varios valores. Cada miembro del array se almacena en una posición a la que se hace referencia utilizando un valor clave. Las claves pueden ser numéricas o asociativas.

```
    //array numérico
    $modulos1=[0=>"Programación",1=>"Base de datos",....,9=>"Desarrollo web
en entorno servidor"];
```

```
   //$modulos1=array(0=>"Programación",1=>"Base de
datos",....,9=>"Desarrollo web en entorno servidor");

   //array asociativo
   $modulos2=["PR"=>"Programación","BD"=>"Base de
datos",...,"DWES"=>"Desarrollo web en entorno servidor"];
```

- La función **print_r**, nos muestra todo el contenido del array que le pasamos.
- Para hacer referencia a los elementos almacenados en un array, hay que utilizar el valor clave entre corchetes:

```
$modulos1[9];
$modulos2["DWES"];
```

- En PHP se pueden crear arrays de varias dimensiones almacenando otro array en cada uno de los elementos de un array.

```
   $ciclos=[
    "DAW"=>["PR"=>"Programación","BD"=>"Base de
datos",...,"DWES"=>"Desarrollo web en entorno servidor"],
    "DAM"=>["PR"=>"Programación","BD"=>"Base de datos",...,"AD"=>"Acceso a
datos"]
   ];
```

- Para hacer referencia a los elementos almacenados en un array multidimensional, hay que indicar las claves para cada una de las dimensiones:

```
$ciclos["DAW"]["DWES"];
```

- No hay que indicar el tamaño del array para crearlo. Ni siquiera es necesario indicar que una variable concreta es de tipo array:

```
$modulos1[0]="Programación";
$modulos1[1]="Base de datos";
$modulos1[9]="Desarrollo web en entorno servivor";
```

- Ni siquiera es necesario especificar el valor de la clave. Si se omite, el array se irá llenando a partir de la última clave numérica existente, o de la posición 0 si no existe ninguna:

```
$modulos1[]="Programación";
$modulos1[]="Base de datos";
```

```
$modulos1[]="Desarrollo web en entorno servivor";
```

5.1 Recorrer un array

Las cadenas de texto o strings se pueden tratar como arrays en los que se almacena una letra en cada posición, siendo 0 el índice correspondiente a la primera letra, 1 el de la segunda, etc.

```
$modulo = "Desarrollo web en entorno servidor";
// $modulo[3] == "a";
```

Para recorrer un array se puede utilizar **el bucle foreach**. Utiliza una variable temporal para asignarle en cada iteración el valor de cada uno de los elementos del arrays. Puedes usarlo de dos formas.

- Recorriendo sólo los elementos:

```
foreach ($modulos1 as $modulo)
        echo $modulo . "<br/>";
```

- O recorriendo los elementos y sus valores clave de forma simultánea:

```
foreach ($modulos2 as $codigo => $modulo)
        echo "El código del módulo ".$modulo." es ".$codigo."<br/";
```

> Hoja03_PHP_03

6. Recuperación y utilización de información proveniente del cliente Web

6.1 Métodos GET y POST

Son métodos del protocolo HTTP para intercambio de información entre cliente y servidor.

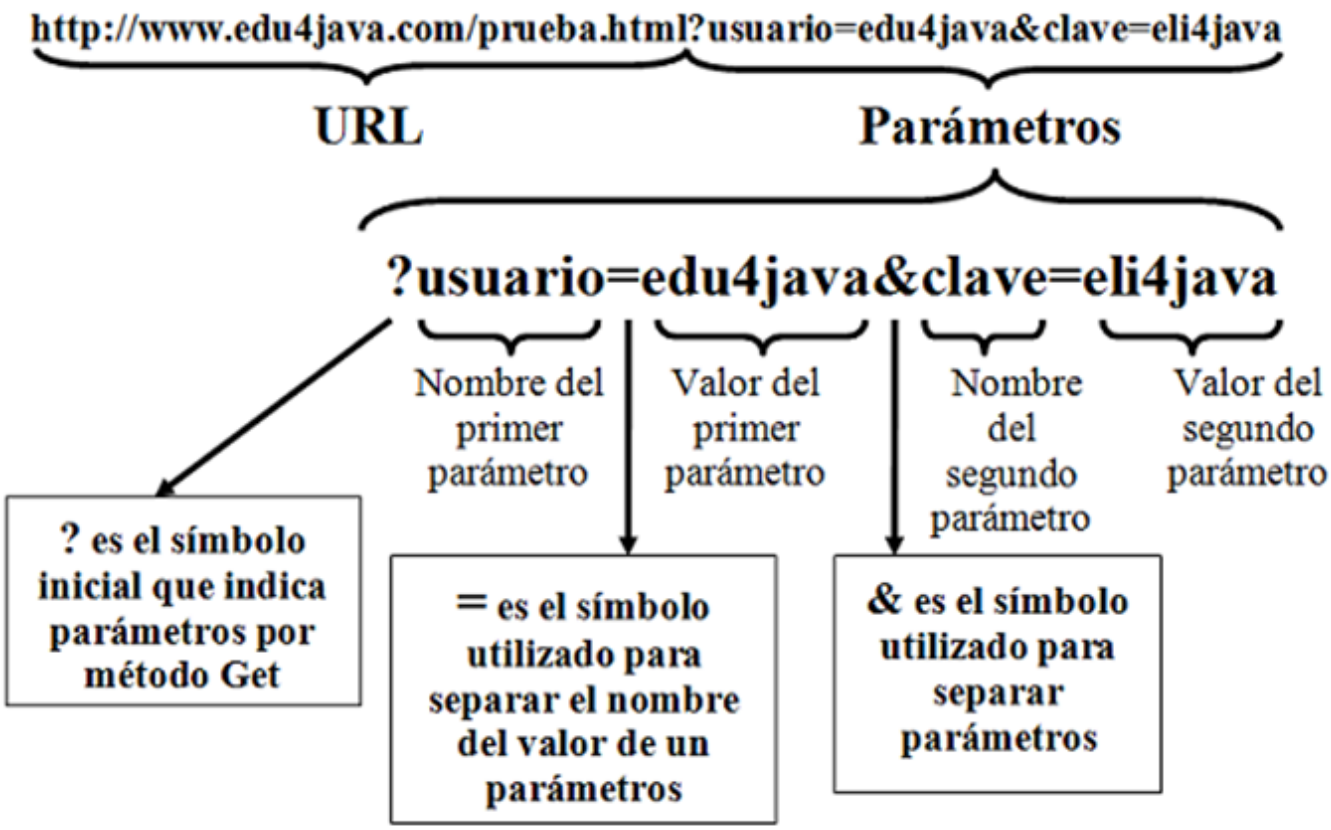
- **get**: el método más usado, pero no se usa en formularios. Pide al servidor que le devuelva al cliente la información identificada en la URI.
- **post**: se usa para enviar información a un servidor. Se utiliza básicamente en formularios.

La principal diferencia radica en la codificación de la información.

6.2 Método GET

Utiliza la dirección URL que está formada por:

- **Protocolo**: especifica el protocolo de comunicación
- **Nombre de dominio**: nombre del servidor donde se aloja la información.
- **Directorios**: secuencia de directorios separados por / que indican la ruta en la que se encuentra el recurso
- **Fichero**: nombre del recurso al que acceder.
- Detrás de la URL se coloca el símbolo **?** Para indicar el comienzo de las variables con valor que se enviarán, separadas cada una de ellas por **&**.
- **Ejemplo**: http://www.example.org/file/example1.php?v1=0&v2=3



6.3 Método POST

La información va codificada en el cuerpo de la petición HTTP y por tanto viaja oculta.

No hay un método más seguro que otro, ambos tienen sus pros y sus contras.

Es conveniente usarlos GET para la recuperación y POST para el envío de información.

7. Formularios

Son la forma de hacer llegar los datos a una aplicación web.

Van encerrados en las etiquetas

```
<form>
    ....
</form>
```

Dentro de las etiquetas de formulario se pueden incluir elementos sobre los que puede actuar el usuario. Ejemplo:

```
<input>
<select>
<textarea>
<button>
```

En el **atributo action** del FORM se indica la página a la que se enviarán los datos del formulario

En el **atributo method** se especifica el método usado para enviar la información:

- **get**: los datos se envían en la URI utilizando el signo **?** como separador.
- **post**: los datos se incluyen en el cuerpo del formulario y se envían usando el protocolo HTTP

Ejemplo con GET

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo Formularios</title>
</head>
<body>
    <h1>Ejemplo de procesado de formularios</h1>
    <form action="ejemplo1.php" method="get">
        <label for="nombre">Introduzca su nombre:</label>
        <input type="text" id="nombre" name="nombre">
        <br/>
        <label for="apellido">Introduzca sus apellidos:</label>
        <input type="text" id="apellido" name="apellidos"><br/>
        <input type="submit" name="enviar" value="Enviar">
    </form>
</body>
</html>
```

Ejemplo con POST

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo Formularios</title>
</head>
```

```
<body>
    <h1>Ejemplo de procesado de formularios</h1>
    <form action="ejemplo1.php" method="post">
        <label for="nombre">Introduzca su nombre:</label>
        <input type="text" id="nombre" name="nombre">
        <br/>
        <label for="apellido">Introduzca sus apellidos:</label>
        <input type="text" id="apellido" name="apellidos"><br/>
        <input type="submit" name="enviar" value="Enviar">
    </form>
</body>
</html>
```

8. Procesamiento de la información

8.1 Con GET

En el caso del envío de información utilizando el método GET existe una variable especial $_GET, donde se almacenan todas las variables pasadas con este método.

La forma de almacenar la información es **un array en el que el índice es el nombre asignado al elemento del formulario**

```
$_GET['nombre'];
$_GET['apellidos'];
```

También se puede recuperar con la función **print_r** que muestra el array completo.

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

8.2 Con POST

Al igual que en el método GET, se utiliza una variable interna $_POST, donde se almacenan todas las variables pasadas con este método.

La forma de almacenar la información, también es **un array en el que el índice es el nombre asignado al elemento del formulario**

```
$_POST['nombre'];
$_POST['apellidos'];
```

También se puede recuperar con la función **print_r** que muestra el array completo.

Existe otra variable $_REQUEST que contiene tanto el contenido de $_GET como $_POST

> Hoja03_PHP_04 (ejercicios 1, 2 y 3)

8.3 Validación de datos

Siempre que sea posible, es preferible **validar los datos que se introducen en el navegador antes de enviarlos. Para ello deberás usar código en lenguaje Javascript**.

Si por algún motivo hay datos que se tengan que validar en el servidor, por ejemplo, porque necesites comprobar que los datos de un usuario no existan ya en la base de datos antes de introducirlos, **será necesario hacerlo con código PHP en la página que figura en el atributo action del formulario**.

```
<html>
    <head> <title>Desarrollo Web</title> </head>
    <body>
        <?php
            if (isset($_POST['enviar'])) {
                $nombre = $_POST['nombre'];
                $modulos = $_POST['modulos'];
                print "Nombre: ".$nombre."<br />";
                foreach ($modulos as $modulo) { print "Modulo: ".$modulo."
<br />"; }
            }
            else {
        ?>

        <form name="input" action="" method="post">
            <label for="nombre">Nombre del alumno: </label>
            <input type="text" name="nombre" id="nombre" />
             <br />
            <p>Módulos que cursa:</p>
            <input type="checkbox" name="modulos[]" value="DWES" />
            Desarrollo web en entorno servidor<br />
            <input type="checkbox" name="modulos[]" value="DWEC" />
            Desarrollo web en entorno cliente<br /> <br />
            <input type="submit" name="enviar" value="Enviar" />
        </form>
        <?php    }    ?>
    </body>
</html>
```

en el **atributo action** del formulario también se puede poner echo $_SERVER['PHP_SELF'];

> Hoja03_PHP_04

> Hoja03_PHP_05