UT3 - Programación basada en lenguajes de marcas con código embebido

Sumario

Curriculum

- Sentencias. Tipos. Bloques.
- Comentarios.
- Tomas de decisión.
- Bucles.

- Tipos de datos compuestos: Arrays.
- Funciones. Parámetros.
- Recuperación y utilización de información proveniente del cliente Web.
- Procesamiento de la información introducida en un formulario. Métodos POST y GET.

Contenidos

3.1. Sentencias. Tipos. Bloques

Las sentencias en PHP irán separadas por ; como en Java
Podremos crear bloques de sentencias con las llaves { } que agruparán varias sentencias.
Dentro de los bloques { } podremos incluir tanto código PHP como texto en HTML, abriendo y cerrando en modo PHP con las etiquetas `<?php ?>`

3.2. Comentarios

Ya visto en la unidad anterior - Comentarios

3.3. Tomas de decisión

Recordad que la sentencia afectada por una estructura de control puede ser una unica sentencia o varias sentencias agrupadas en un bloque con { }

Estructuras de Control

- Introducción
- if
- else
- elseif/else if
- switch
- Sintaxis alternativa de estructuras de control

Consejo de ISW: Aunque no sea necesario sintácticamente siempre es preferible incluir entre { } las sentencias afectadas por una condición.

3.4. Bucles.

Bucles

- while
- do-while
- for
- foreach
- break
- continue

Bloque 1 – Estructuras de control - Ejercicios propuestos

1. Realiza una página web que muestre la tabla de multiplicar del 5, utilizando bucles en PHP.
2. Realiza una página web que muestre todas las tablas de multiplicar del 1 al 10.
3. La función *rand()* genera un número aleatorio. Realizar una página que muestre de forma aleatoria una tabla de multiplicar.

4. Utilizando la construcción *switch* realiza un programa que genere un número aleatorio entre 1 y 10 y muestre en la página el número en letra.
5. Realizar una página en php que muestre todos los números entre 1 y 1000 que son múltiplos de 3, 4.
6. Realizar una página en php que muestre todos los números entre 1 y 1000 que son múltiplos de 3, 5 y 7. Avanzará de línea cada vez que se cambie de decena.
7. Utilizando el bucle do..while realiza un programa que muestre números aleatorios por pantalla entre 1 y 100 hasta que salga el 15.
8. La función *time()* devuelve la fecha en número de segundos desde el 1 de enero de 1970. Realiza una página web que muestre en vuestro equipo todos los números múltiplos de 5 que le de tiempo a mostrar en 5 segundos. Mostraréis solamente 10 números por línea y Después de 10 líneas incluireis un separador (raya, salto de línea, caracteres ...)

3.5 Funciones. Parámetros.

Funciones

- Funciones definidas por el usuario
- Argumentos de funciones
- Devolver valores
- Funciones variables
- Funciones internas (incluidas)
- Funciones anónimas

Ámbito de variables, variables globales
Funciones predefinidas del lenguaje

- Funciones de Fecha/Hora
- Funciones de strings – Se reseñan aquí algunas que os pueden resultar útiles
  - addslashes — Escapa un string con barras invertidas
  - chr — Devuelve un caracter específico
  - sprintf — Devuelve un string formateado
  - htmlentities — Convierte todos los caracteres aplicables a entidades HTML
  - explode — Divide un string en varios string
  - implode — Une elementos de un array en un string
  - join — Alias de implode
  - lcfirst — Pasa a minúscula el primer caracter de un string
  - nl2br — Inserta saltos de línea HTML antes de todas las nuevas líneas de un string
  - number_format — Formatear un número con los millares agrupados
  - ord — devuelve el valor ASCII de un caracter
  - parse_str — Convierte el string en variables
  - print — Mostrar una cadena
  - str_replace — Reemplaza todas las apariciones del string buscado con el string de reemplazo
  - str_ireplace — Versión insensible a mayúsculas y minúsculas de str_replace
  - str_pad — Rellena un string hasta una longitud determinada con otro string
  - str_repeat — Repite un string
  - str_shuffle — Reordena aleatoriamente una cadena
  - str_split — Convierte un string en un array

- strcasecmp — Comparación de string segura a nivel binario e insensible a mayúsculas y minúsculas
- strip_tags — Retira las etiquetas HTML y PHP de un string
- stripcslashes — Desmarca la cadena marcada con addcslashes
- strpos — Encuentra la posición de la primera ocurrencia de un substring en un string
- stripos — Encuentra la posición de la primera aparición de un substring en un string sin considerar mayúsculas ni minúsculas
- stripslashes — Quita las barras de un string con comillas escapadas
- strlen — Obtiene la longitud de un string
- strrchr — Encuentra la última aparición de un caracter en un string
- stristr — strstr insensible a mayúsculas y minúsculas
- strrev — Invierte una string
- strripos — Encuentra la posición de la última aparición de un substring insensible a mayúsculas y minúsculas en un string
- strrpos — Encuentra la posición de la última aparición de un substring en un string
- strspn — Averigua la longitud del segmento inicial de un string que consista únicamente en caracteres contenidos dentro de una máscara dada.
- strstr — Encuentra la primera aparición de un string
- strtolower — Convierte una cadena a minúsculas
- strtoupper — Convierte un string a mayúsculas
- strtr — Convierte caracteres o reemplaza substrings
- substr_compare — Comparación segura a nivel binario de dos o más strings desde un índice hasta una longitud de caracteres dada
- substr_count — Cuenta el número de apariciones del substring
- substr_replace — Reemplaza el texto dentro de una porción de un string
- substr — Devuelve parte de una cadena
- trim — Elimina espacio en blanco (u otro tipo de caracteres) del inicio y el final de la cadena
- ltrim — Retira espacios en blanco (u otros caracteres) del inicio de un string
- rtrim — Retira los espacios en blanco (u otros caracteres) del final de un string
- ucfirst — Convierte el primer caracter de una cadena a mayúsculas
- ucwords — Convierte a mayúsculas el primer caracter de cada palabra de una cadena
- wordwrap — Ajusta un string hasta un número dado de caracteres

Bloque 2 – Funciones - Ejercicios propuestos

9. Crea la función *EsPrimo(numero)* que devuelva un booleano que indique si el número pasado como parámetro es primo. Utilizando dicha función mostrar en una página los números primos menores de 100 que existen.
10. Crea la función *DiasMes(num_mes)* que devolverá un entero que será el número de días que tiene un mes. Utilizando dicha función realizar un programa que imprima las fechas existentes entre el 1 de enero de 1999 y el 31 de diciembre de 2012. Las fechas se mostrarán separadas por una coma y cada mes aparecerá en una línea diferentes.

11. Crea la función *NombreMes(num_mes)* que devolverá una cadena que será el nombre de mes que corresponde al parámetro. Modifica el ejercicio anterior para que en cada línea aparezca el nombre de més y el año y a continuación solo aparezca el número de día.
12. Crea la función *EstNoSeDebeHacer()* -sin parámetros, que hará uso de la palabra reservada global- que modifica la variable *$num* asignándole el doble de su valor. La variable está iniciada fuera de la función. Crea una página que cree y pruebe el funcionamiento de la función.
13. Crea la función *Intercambia(v1, v2)* la cual intercambiará el valor de las dos variables. Realizar una página en la que se pruebe el funcionamiento de dicha función intercambiando el valor de dos variables. Mostrar las variables antes y después de la invocación de la función.
14. Utilizando la función predefinida *date()*, realiza una página en la que se muestre la fecha y hora actual.
15. Utilizando la función *date()* y *time()* escribe una página que muestre la fecha que será dentro de 50 segundos, y dentro de 2 horas, 4 minutos y 3 segundos.
16. Utilizando la función *date()* y la función *NombreMes()* creada anteriormente, muestra el nombre del mes en el que estamos.
    Crea la función en el fichero "*funciones_fecha.php*" que luego incluirás (*include* o *require*) en la solución.
17. Crea la función *MuestraFecha(dia, mes, anyo)* que mostrará la fecha que se le pase como parámetro en el formato "dia _semana (lunes...), num_dia de nombre_mes de num_anyo".

    Ejemplo: MuestraFecha(9,10,2018) mostrará
    martes 9 de octubre de 2018 / o / Tuesday 9 october 2018

    Prueba la función.
    Utiliza el fichero "funciones_fecha.php" creado anteriormente.

Podéis obtener información y ejemplos de las funciones de fecha y hora en PHP 5 Date and Time

3.6 Tipos de datos compuestos: Arrays.

Arrays en PHP (PHP.net)

Véase PHP 5 Arrays de W3C

Definición y acceso

Un array en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves. Este tipo se optimiza para varios usos diferentes; se puede emplear como un array, lista (vector), tabla asociativa (tabla hash - una implementación de un mapa), diccionario, colección, pila, cola, y posiblemente más. Ya que los valores de un array pueden ser otros arrays, también son posibles árboles y arrays multidimensionales.

Diferencias con java y otros lenguajes:

- La clave (indice) puede ser un **integer** o un **string**.
- El valor puede ser de cualquier tipo.

Otras características:

- Para acceder a elementos de array se utiliza la sintaxis de corchete. Igual que Java

- Añadir y borrar elementos
  Totalmente diferente de Java y otros lenguajes fuertemente tipados donde el espacio de un array de fija al definirlo o crearlo.
  Recordad que en PHP un array es un mapa de java
  - Añadir elementos:
    - Con clave: `$array['clave']=valor`
    - Sin clave: `$array[]=valor` *`Añade al final y calcula su indice`*
  - Liberar elementos: *`unset($array[key]);`*
- Para crear e inicializar un array se puede utilizar indistintamente la palabra *array(...)* o los corchetes `[]` a partir de la versión 5.3.
  Ejemplos:
  ```
  $lista=[1,2,3,4];
  $lista=array(1,2,3,4);
  ```

Algoritmos asociados al procesamiento de un array

Recorrer array:

Construcción que se debe utilizar: Véase bucle *foreach*:

Construcción obsoleta: Recorrido con indice numérico :

Se utiliza la función *Count(array)* para saber el número de elementos que tiene el array

http://lineadecodigo.com/php/recorrer-un-array-en-php/

```php
//
// Esta forma de recorrer el array está obsoleta y producirá error si el
// array no tiene indices numéricos
//
$a=array('a'=>'ta', 'b'=>'tb', 1=>'t1');

for($i=1; $i<count($a); $i++)
{
	echo "<p>$i - [{$a[$i]}]</p>";
}
// Los arrays son mapas
```

Disponemos de funciones para realizar otro tipo de operaciones habituales, aunque también podremos prograrmarlas directamente

- Buscar un elemento: *array_search()*
- Buscar una clave: *array_key_exists()*. Se puede también preguntar con *`isset($array[key])`*
- Ordenando arrays: http://www.php.net/manual/es/array.sorting.php

Funciones de arrays: http://www.php.net/manual/es/ref.array.php

Ejemplos – Aquí se tratan los arrays, como estamos habitualmente acostumbrados, indexando por un número. Recordad que para que esto sea posible, debemos crearlos indexados por un número.

- Arrays asociativos en PHP. Concepto y formas de declaración y uso. Ejercicios resueltos. (CU00825B)
- Función count. Recorrido de arrays multidimensionales en PHP con for foreach. Ejemplos. (CU00826B)

Bloque 3 – Arrays - Ejercicios propuestos

18. Utilizando arrays crea la función *DiasMes(num_mes)* que devolverá un entero que será el número de días que tiene un mes.
    Utilizando dicha función realiza un programa que imprima el número de días que tienes los distintos meses. El nombre del mes se almacenará en una array igualmente.
19. Realizar una página que verifique si un dni/nif es correcto. (solo hace falta para los que tienen el formato NúmeroLetra).
    Formato NIF: http://es.wikipedia.org/wiki/N%C3%BAmero_de_identificaci%C3%B3n_fiscal
20. Crea la función *Max(array)* que nos devolverá el valor máximo de un array. Realiza una página que pruebe dicha función.

Nota: En PHP muchas veces utilizaremos los arrays como objetos para almacenar información. Dichos objetos no tendrán una estructura predeterminada y todos serán iguales como tenemos en las clases de Java. Este mecanismo se utilizará también en Javascript

21. Se desea almacenar información sobre coches, para cada coche se almacenaran las siguientes características (atributos):
    - matrícula
    - color
    - modelo
    - marca

    Realiza un array que almacene información de 5 o más coches.
    Crea la función `MuestraCoche($coche)`, donde `$coche` será un array que contiene los atributos indicados anteriormente que c
    Rea liza la función `MuestraCoches($lista)` que mostrará por pantalla información de los coches almacenados .
    Añade dos coches adicionales al array, después de mostrar, y vuelve a mostrar toda la lista.

    Nota: Se utilizará un array para almacenar la información de cada coche. Los indices, serán el nombre de los atributos que deseamos almacenar. Esto se puede hacer también utilizando objetos (clases).

22. Comprueba si los arrays se pasan por valor o referencia como parámetros de una función. Modifica los datos de una array pasado como parámetro a una función y comprueba si se han modificado al salir de esta.
23. Realiza la función `FechaActual()` que devuelva la fecha en un array con el formato [‘dia’=>dia, ‘mes’=>mes, ‘año’=>año].
    Obtén la fecha actual llamando a la función muestra la fecha en el formato dd/mm/aa.

3.7 Recuperación y utilización de información proveniente del cliente Web.

En este apartado describiremos los mecanismos que tiene el protocolo HTTP para enviar y recibir información. Para este propósito necesitaremos conocer un poco la estructura de dicho protocolo. Si bien no es necesario conocer los mecanismos que se utilizan para realizar una aplicación web, pues los lenguajes/frameworks modernos los ocultan, siempre es interesante saber como se organizan las peticiones con objeto de tener mayor control sobre la situación.

Los siguientes artículos abordan como se envía la información en una aplicación web que utiliza el protocolo HTTP

- Véase El protocolo HTTP
- Véase Sending form data
- Véase Cabeceras HTTP para Dummies

En un primer momento solo es preciso que entendáis como se envía información al servidor con el protocolo HTTP, aunque no está de más que los leáis de forma completa pues tratan temas interesantes que más adelantes utilizaréis.

Puntos clave a mostrar:
- F12 en navegadores
- Ver cabeceras en navegadores
- Ver datos enviados por POST y GET en el navegador

Bloque 4 – Formularios - Ejercicios propuestos

24. Explora la web y observa pulsando F12 en el navegador la estructura de cabeceras y cuerpo.
25. Usando alguna de las siguientes páginas, o estas otras , explora las cabeceras cuando se envían datos. Intenta encontrar páginas que hagan uso de POST y de GET. Si de GET no encontráis ningúna observad la siguiente https://www.google.es/

3.8 Procesamiento de la información introducida en un formulario. Métodos POST y GET.

Repaso de HTML

Para trabajar como formularios en PHP es preciso tener los conocimientos básicos de HTML que nos permiten gestionarlos:

- Léase el artículo "*Todos sobre formularios html.odt*"
- Léase el artículo "*Formularios HTML W3C.odt*"
- Léase el artículo "*Etiquetas HTML5 formularios e inputs.odt*"
- Léase el artículo "*Nuevas características de formularios en HTML5.odt*"
- Véase las etiquetas disponibles para la creación de formularios en HTML:
  - HTML Forms and Input
  - HTML5 Input Types
  - HTML5 New Form Attributes

Procesamiento de los datos recibidos en nuestro script PHP

Véase la presentación: "UT3 – Formularios" para hacerse una idea rápida sobre el procesamiento de formularios.

Métodos GET y POST: Léase el artículo "Métodos GET y POST.odt"

Recuperación de información con GET: El contenido se pasa en la URL. Se puede pasar fácilmente información.
http://php.net/manual/es/reserved.variables.get.php

Recuperación de información con POST
- El contenido se pasa anexo en la petición HTTP, no es visible por el usuario.
- Léase el artículo: "Uso de las variables en formularios (GET y POST).odt"

Recuperación de información con REQUEST:

- http://php.net/manual/es/reserved.variables.request.php
- Léase el artículo "Diferencias entre REQUEST, GET, POST.odt"

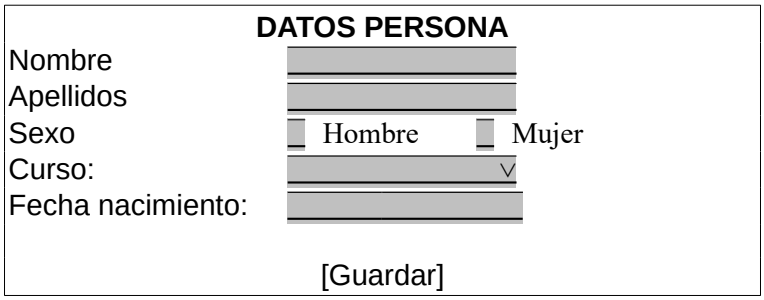
Procesar un formulario en la misma página / Autollamada de páginas

Véase el artículo "Form-Autollamada de páginas.odt"
Véase el artículo "Formularios en PHP"

Bloque 4 – Formularios - Ejercicios propuestos

26. Realiza una web que lea un número y muestre su tabla de multiplicar. El formulario estará en el fichero "*ejercicio_xx.html*" y el procesado del formulario se hará en "*ejercicio_xx.php*".
27. Realiza una web que lea el nombre y apellidos de una persona y los muestre en mayúsculas. Véanse las función *strtoupper().*
    En la página que muestra el resultado tendremos la opción de volver a mostrar el formulario para pedir más datos mediante un enlace.
28. Utilizando una sola URI / URL "*ejercicio_xx.php*" realiza el ejercicio anterior. *Véase el artículo "Autollamada de páginas".*
    Nota: esto se puede realizar utilizando un único fichero, o utilizando más de un fichero. Con las funciones *require* o *include* podemos incluir todos estos ficheros de forma que se muestren en una única URL como si fuese un único fichero.
29. Realiza una web que lea un número y muestre su tabla de multiplicar. La aplicación debe cumplir los siguientes requisitos:
    - El formulario se mostrará y procesará en una única URL (un solo fichero, o varios con *include*/*require*).
    - Si el valor introducido no es un número, o si no está en el rango 1..10 mostraremos un mensaje de error notificando dicha incidencia y dando la opción de introducir de nuevo el valor.
    - Se mostrarán en los campos el valor erróneo para que el usuario pueda ver el valor introducido.

30. Realizar una página que lea un formulario en el que se lean los siguientes campos:
    - Nombre:
    - Apellidos:
    - Sexo: H o M (botón radio)
    - Curso: 1º SMR, 2º SMR, 1º ASIR, 2º ASIR, 1º DAW, 2º DAW (select/combobox). Por defecto aparecerá en blanco
    - Fecha de nacimiento



Una vez enviado el formulario mostrará la información que se ha enviado en formato de tabla.

Nota: Cuando se muestran campos de tipo `<select>` en un formulario una opción habitual en las aplicaciones web es utilizar una función que nos generé dinámicamente las opciones disponibles. Véase ejemplos de código.

3.9. Filtrado de datos recibidos de un formulario:

Principio a seguir siempre

> *Todo valor recibido del cliente es **sospechoso** y debe ser **filtrado siempre** en el **servidor**. Aunque ya se haya filtrado en el cliente (javascript o nuevos atributos de HTML5), nada nos garantiza que el cliente está haciendo un uso correcto de la aplicación pues recordad que se puede modificar la petición en cualquier momento con el navegador u otras aplicaciones.*

El procedimiento genérico a seguir en el proceso de filtrado de un formulario es el siguiente:

```
** FILTRADO DE DATOS DE UN FORMULARIO **
Si han enviado el formulario
   Comprobar que campos recibidos cumplen las condiciones impuestas
Fin Si

Si hay errores o es la primera vez entonces
   Mostrar página de formulario (include)
   En cada campo del formulario mostraremos el valor enviado
Sino
   Realizamos las operaciones que procedan
   Mostramos página resultado

Fin Si
```

Para incluir el valor enviado en un campo tan solo tenemos que inicializar el control con el campo enviado. Para ello puede resultarnos interesante crear una función auxiliar

```php
<?php
/**
 * Devuelve el valor de un campo enviado por POST. Si no existe devuelve el valor por defecto
 * @param string $nombreCampo
 * @param mixed $valorPorDefecto
 * @return string
 */
function ValorPost($nombreCampo, $valorPorDefecto='')
{
	if (isset($_POST[$nombreCampo]))
		return $_POST[$nombreCampo];
	else
		return $valorPorDefecto;
}

// Ejemplo de uso en una etiqueta de tipo INPUT
?>
<!--  Versión si función auxiliar -->
<input type="text" name="nombre"
	value="<?=isset($_POST['nombre']) ? $_POST['nombre'] : '' ?>"/>
<!--  Versión con función auxiliar -->
<input type="text" name="nombre" value="<?=ValorPost('nombre')?>"/>
```

- Véase la presentación *Seguridad Web*: Centrándose en los apartados de Validación y escapado.http://www.slideshare.net/flaiwebnected/bloque-1-php-y-seguridad-web
- Véase el artículo: Filtrar entradas de un formulario en PHP
- Véase el artículo "Expresiones regulares en PHP" o el o el minimanual Explicaciones y ejemplos para el manejo de expresiones regulares.
-

Funciones de filtrado de datos en PHP

- Funciones de filtrado de datos PHP
  - W3Schoools - PHP filter_var() Function
  - GeeksForGeeks - PHP | filter_var() Function - Examples
  - Prueba on-line funciones de filtrado

Separar lógica de presentación

Siempre que realicemos una aplicación web que mezcle código PHP y HTML es conveniente, en la medida de lo posible separar la lógica de la presentación de forma que la secuencia de ejecución se entienda claramente sin necesidad de estar investigando el código en profundidad.

Para llevar esto a cabo siempre conviene tener presente las siguientes reglas:

- La página que gestiona la lógica de la interacción deberá ser un fichero de tipo PHP que solo contendrá código.
- La página que muestre el formulario y resultado, tan solo contendrán el código en PHP imprescindible para mostrar la información.

Visto con un ejemplo quedaría algo parecido a lo siguiente:

- *pagina.php***:** Fichero que controla la lógica
- *pagina_form.php*: Fichero que contiene el formulario de entrada.
- *pagina_form_procesado.php*: Fichero que contiene la página que muestra el resultados

Este esquema puede ser ampliando incluyendo el número de fichero que estimemos oportuno.

El contenido de *pagina.php* sería algo parecido a

```php
<?php
//
// ¿Han enviado datos? POST o GET dependerá
//
if ($_POST)
{     // Datos enviados
      $hay_errores=FALSE;
      //
      // Filtramos los datos.
      // Si hay errores activamos $hay_errores
      //
      if ($hay_errores)
      {     // Hay errores - Debemos mostrarlos y preguntar
            include('pagina_form.php');
      }
      else
      {
            // Realizamos operaciones que correspondan
```

```
            include('pagina_form_procesada.php');
      }
}
else
{     // Mostramos formulario por primera vez
      include('pagina_form.php');
}
```

Más adelante veremos que esto es una primera aproximación simple al patrón de diseño MVC (Modelo-Vista-Controlador)

Control de errores

Cuando estamos depurando un formulario enviado, muchas veces debemos controlar los errores que se producen en los diferentes campos para luego ver como proceder.

Un patrón muy útil en PHP es almacenar los errores en un array desde el que luego podremos recuperar los mensajes.

```
<?php
// ...
//
// Filtramos datos de formulario

// Suponemos que no habrá errores
$errores=[];

if ( /* Es erroneo CAMPO1 */ )
{
      $errores['CAMPO1']='Texto de error';
}
// ...
if ($errores) // Evaluamos N° elementos
{     // Hay errores
      // Lo que proceda
}
else
{
      // Lo que proceda
}

// La siguiente función se utilizará en la vista
/**
 * Muestra el texto de error si el campo es erroneo
 * @param string $campo Nombre campo
 */
function VerError($campo)
{
      global $errores;
      if (isset($errores[$campo]))
      {
            echo $errores[$campo];
      }
}
```

Bloque 5 – Formularios – Validación - Ejercicios propuestos

En estos ejercicios utiliza las funciones de validación y filtrado que proporciona PHP - Funciones de filtrado de datos PHP con el objetivo de hacer código más seguro.

31. Amplia el ejercicio anterior (30) para que filtre los datos con las siguientes condiciones:
    - Nombre: No puede estar vacío
    - Apellidos: No puede estar vacío
    - Sexo: Debe estar seleccionado alguno
    - Curso: Debe tener seleccionado alguno
    - Fecha de nacimiento: Debe contener una fecha válida. Véase en la ayuda: *checkdate*(), *explode()*

    Si alguno de los campos tiene error se mostrará el formulario de nuevo, con los valores que hemos introducido e indicando con algún mensaje de error o efecto visual (poner en rojo) que hay campos erróneos.

    Si los campos son válidos se mostrará también la edad en la tabla resultado.

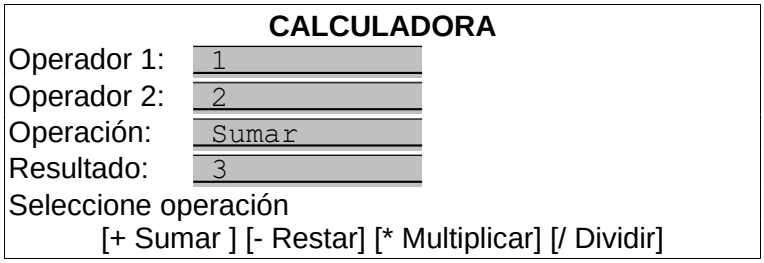
32. Amplia el ejercicio anterior para que incluya un campo observaciones, de tipo <*textarea*>. Al mostrar las observaciones, después de enviar el formulario, se deberá respetar el salto de línea introducido.
    Véase las siguientes funciones que pueden resultar interesantes:
    - *str_replace()*. Sustituir el caracter **\n** por la etiqueta `<br/>`.
    - Nl2br(): Función que realiza directamente la sustitución

    Probad con ambas funciones.

33. Realiza un formulario que contendrá un campo de texto en el que se almacenará un número y un botón [+1] . Al pulsar el botón [+1] se mostrará de nuevo la página y se incrementará en uno el contenido del campo de texto.

34. Realiza el ejercicio anterior, pero en lugar de utilizar un cuadro de texto el número lo mostraras en un párrafo.
    Nota: Puede que os resulta interesante utilizar un campo oculto.

35. Realiza un programa que implemente una calculadora en una página web que tendrá el siguiente formato:



    La página tendrá los siguientes botones que permitirán realizar la suma, resta, multiplicación o división de los números introducidos en los campos *operador 1* y *operador 2*. Una vez realizada la operación seguirán mostrando el valor del campo en los campos operadores y mostrarán el resultado en el campo “Resultado”. Los campos Operación y Resultado serán de solo lectura

Campos como array en el formulario

Cuando en un formulario el nombre de campo en HTML va seguido por corchetes, con o sin indice, este es procesado en PHP como si fuese una variable de tipo array.

- Véase Handling checkbox in a PHP form processor
- Véase PHP: Get Values of Multiple Checked Checkboxes

Bloque 5 – Formularios Avanzado- Ejercicios propuestos

36. Este ejercicio es complejo, pero si lo completáis os ayudará a comprender como a partir de las estructuras de datos internas del lenguaje de programación (PHP en este caso), podremos generar HTML dinámico para capturar datos. Además estas estructuras permitirán/facilitarán procesar la información recibida cuando se envía el formulario.

    La empresa de encuestas SIGMA3 nos ha encargado que realicemos una página en la que deseamos obtener información sobre los gustos de los usuarios. Para ello tendremos distintas preguntas las cuales tendrás opciones que pueden ser multiselección (checbox) o de una única opción. Las preguntas que desea que realicemos son las siguientes

    1. Sexo: Hombre / Mujer
    2. Aficiones (múltiple)
        - Deporte - Cine -Teatro
    3. Estudios que tiene (múltiple)
        - ESO -C.F.G.Medio - C.F.G. Superior - Grado
    4. Lugar al que le gustaría ir de vacaciones (una sola opción)
        - Mediterráneo
        - Caribe
        - EEUU
        - Centro europa

    Se desea realizar una página que lea los datos del usuario en un formulario y luego muestre los resultados en la página que procese dicho formulario. Una vez enviado el formulario se mostrará por pantalla los valores seleccionados.

    Para la realización de este ejercicio, en lugar de escribir el código HTML del formulario directamente, lo que haremos será crear una función en PHP que nos genere dicho código a partir de los datos recibidos en su parámetro.

    Diseñar la función *GetHTMLPregunta($pregunta /* array */)*, la cual creará el código HTML que *devolverá como cadena.* El parámetro de la función será un array que tendrá el siguiente formato:

```
$pregunta=array(
      'texto_pregunta'=>'....'
      'tipo'=>MULTI_OPCION | UNA_OPCION,
      'campo'=>nombre_campo,
      'respuestas'=>array(
            array('etiqueta'=>'...', 'valor'=>VALOR),
            ...
      )
);
```

    Donde:

    - texto_pregunta: Será el texto de la pregunta.
    - tipo: El tipo de control a crear Radio o Checkbox

- campo: El nombre del campo, atributo *name*
- respuestas: Las respuestas posibles para la pregunta. Donde cada respuesta es un array que contiene la siguiente información:
  - etiqueta: Etiqueta que aparecerá junto al botón radio o checkbox. Debe estar asociada con la etiqueta HTML label.
    Nota: Si es un checkbox cada campo debe tener un nombre diferente. Recordad que podéis utilizar el formato *nombre[]*
  - valor: Atributo value del campo.

Por ejemplo para la primera pregunta 1. Sexo

```
$pregunta1=[
      'texto_pregunta'=>'Sexo'
      'tipo'=>UNA_OPCION,
      'campo'=>'sexo',
      'respuestas'=>[
            ['etiqueta'=>'Hombre', 'valor'=>'H'],
            ['etiqueta'=>'Mujer', 'valor'=>'M']
      ]
);
```

Lo que pretende este ejercicio es que las preguntas las almacenéis en un array de preguntas y luego con la función *CreaPregunta()* generéis el formulario que se mostrará.

Nota: Recordad que el nombre de los campos de los controles puede ser un array.

37. Amplia el ejercicio anterior de forma que en función de las selecciones genere una nueva página en la que se solicite más información.
    - Si en afición hay “deporte” nos pregunte también el tipo de deporte que nos gusta a elegir entre las siguientes opciones: (ciclismo, tenis, futbol, baloncesto, voleibol, etc).
    - Si en estudios hay ESO, preguntar el año que obtuvo el título.
    - Si el destino es Mediterráneo, seleccionar alguna o varias de las siguientes opciones: Cataluña, Valencia, Andalucía, Tunez, Turquía, Italia, Francia,...

    Una vez seleccionadas las opciones se mostrará en la página resultado todas las respuestas seleccionadas.
    Nota: Puede que necesitéis utilizar campos ocultos para transmitir la información entre los formularios.

Funciones interesantes para el manejo de formularios

- *nl2br*: Convierte saltos de línea en etiquetas <*br/*>, útil para mostrar el valor de un TextArea
- *htmlentities / htmlspecialchars*: Convierte tildes y caractéres extraños en entidades HTML. Útil para leer datos en codificaciones distintas de de UTF-8
- *urlencode / urldecode*: para generar parámetros con GET

Subiendo ficheros

- Subida con el método POST
- Véase artículo “How to Upload a File in PHP (With Example)”

Bloque 6 – Formularios Envío ficheros -Ejercicios propuestos

38. Realiza una página que nos permita subir un fichero a través de un formulario. Una vez subido generará una página que contendrá un enlace al fichero subido. El fichero lo

guardaremos en la misma carpeta que está el script PHP. Véanse las constante predefinida _FILE_ y similares.

39. Modifica el programa anterior para que una vez subido el fichero muestre su contenido en el navegador.
    Véase la función *readfile()*
40. Se desea realizar un visor de fotos "cutre" que nos permita ver en el navegador fotos/imágenes que subiremos con un formulario. En la pantalla 1 pediremos el fichero que deseamos mostrar, y en la pantalla 2 mostraremos el fichero que hemos seleccionado

Pantalla 1
Visor de fotos Fichero: [Ver imagen]
Pantalla 2
Visor de fotos    Ver otra imagen

Nota: Este ejercicio podría realizarse en el cliente completamente utilizando javascript, y dada la funcionalidad que tiene sería lo más apropiado. Aquí pretendemos trabajar el envío de ficheros en PHP y además ver como los enlazaremos luego.

Cuando se suben ficheros a nuestro servidor normalmente estos se alojarán en carpetas que no son accesibles desde el servidor web, es decir, se ubicarán en carpetas a las que no podremos acceder con una URL.

Enviando parámetros de tipo GET mediante un enlace

Los formularios de tipo GET envían el valor de los campos en la dirección URL con el siguiente formato:

```
http://dominio/recurso?campo1=valor1&campo2=valor2...
```

En nuestras páginas, cuando creamos enlaces con la etiqueta `<a>` vamos a poder simular el envío de datos de un formulario creando direcciones URL que incluyan los campos del formulario en la dirección

Envío.php

Creamos un enlace que nos apunta a un recurso que procesa los datos

```
...
<a href="recibe.php?Nombre=Jose&Apellidos=Lopez%20Jimenez">Envío datos GET</a>ç
...
```

recibe.php

El recurso que recibe los campos los procesa como si fuese el resultado del envío de un formulario GET

```
...<ul><?php
	foreach($_GET as $campo=>$valor)
	{
		echo "<li>$campo = $valor</li>";
	}
?></ul>...
```

Para poder incluir el valor de los campos en los parámetros de la URL deberemos transformarlos previamente para evitar que estos contengan caracteres no válidos. PHP proporciona funciones como *urlencode / urldecode* que nos ayudan con la tarea.

- *urlencode*: transforma una cadena a valores válidos en una URL
- *urldecode*: transforma el valor de un campo recibido en una URL

Bloque 7 – Parámetros con GET - Ejercicios propuestos

41. Utilizando dos páginas simula el envío de información con un enlace. Envía los siguientes campos:
    - *nombre:* Puede tener espacios y caracteres especiales para la URL, por lo que habrá que codificarlos con *urlencode*();
    - *edad:* Será un número

    Nuestra página 1º mostrará una lista de nombres, cada uno de los cuales será un enlace

Juan
María
Pedro
...

    Al pulsar el enlace del nombre en la página 1, saltará a la página 2 mostrando los datos
    El enlace tendrá la forma
    pagina2.php?nombre=Juan%20Lopez%20Perez&edad=23

Datos del seleccionado Nombre: Juan Lopez Perez Edad: 23

3.10 Orientación a objetos PHP

Lecturas

- Programación Orientada a Objetos en PHP para Principiantes
- Introducción a la programación a objetos, que ya deberíais conocer.
  http://www.desarrolloweb.com/manuales/teoria-programacion-orientada-objetos.html

- Léase el capítulo “9. PHP orientado a objetos” del libro “ Aprende PHP con ejercicios” para una introducción rápida a la sintaxis de PHP.
- Véase el tutorial “POO (Programación Orientada a Objetos) con PHP”: https://www.tutorialesprogramacionya.com/phpya/poo/index.php?inicio=0
- *Léase el módulo 2 del libro “Laboratorio PHP y MySQL” - Taller UOC*
- *Léase el libro. PHP Orientado a objetos*: Este libro más que una guía de la sintaxis de PHP es un libro que aborda la programación orientada a objetos utilizando el lenguaje PHP.
- Léase los capítulos 1 y 2 del libro “POO y MVC en PHP”

Clases y Objetos (Referencia del lenguaje)

Introduciremos a continuación una referencia rápida sobre como usar la sintaxis del lenguaje PHP. Vosotros deberéis profundizar por vuestra cuenta en los conceptos importantes de POO.

Muchos de los conceptos vistos en Java son extrapolables a este lenguaje, a veces con las mismas construcciones, otras con pequeñas variaciones en las palabras reservadas. Nuestas primeras aplicaciones apenas usaran objetos. Más adelante cuando avancemos, veremos que al igual que en Java se utilizaran continuamente objetos para realizar nuestros programas.

Indice que se incluye en el apartado “Referencia del lenguaje / Clases y Objetos”

Apartados importantes

- Introducción
- Lo básico
- Propiedades
- Constantes de Clases
- Constructores y destructores
- Visibilidad
- Herencia de Objetos
- Operador de Resolución de Ámbito (::)
- La palabra clave 'static'
- Abstracción de clases
- Interfaces de objetos
- Palabra clave Final
- Clonación de Objetos
- Comparación de Objetos
- Implicación de Tipos
- Objetos y referencias

Apartados que interesa leer pero que puede que no usemos habitualmente

- Sobrecarga
- Métodos mágicos
- Autocarga de clases
- Serialización de objetos

Autocarga de clases

La autocarga de clases nos permitirá referenciar clases definidas en ficheros que no hayamos incluido explícitamente mediante “*include*” o “*require*”. Se utilizará habitualmente en frameworks y librerias.

- Autocarga de clases
  http://php.net/manual/es/language.oop5.autoload.php¶
- Función: spl_autoload_register()
  http://php.net/manual/es/function.spl-autoload-register.php

La autocarga simplifica nuestros programas pues no precisaremos estar continuamente incluyendo sentencias *include* para referenciar a las definiciones de las clases.

Bloque 8 – POO - Ejercicios propuestos

41. Hemos visto en los ejercicios anteriores como podíamos utilizar los arrays para guardar los errores que se producían en los campos. Lo propio sería crear un objeto que se encargase de llevar dicho registro y nos permitiese realizar las operaciones básicas. Por ejemplo la clase `GestorErrores` que tendría los siguientes métodos:

```
/**
 * Anota un error para un campo en nuestro gestor de errores
 * @param type $campo
 * @param type $descripcion
 */
AnotaError($campo, $descripcion)

/**
 * Indica si hay errores
 * @return boolean
 */
HayErrores()

/**
 * Indica si hay error en un campo
 * @return boolean
 */
HayError($campo)

/**
 * Devuelve la descripción de error para un campo o una cadaena vacia
si no
 * hay
 * @param type $campo
 * @return string
 */
Error($campo)

/**
 * Devuelve la descripción del error o cadena vacia si no hay
 * @param type $campo
 * @return string
 */
public function ErrorFormateado($campo)
```

En su constructor se inicializarían los datos que fuesen precisos para que el objeto permita anotar los errores que se producen en el filtrado de datos de un formulario.

Nota: Disponéis de este objeto creado en los ejemplos

Modifica alguno de los ejemplos anteriores para utilizar esta clase.

42. Usando la **clase** DateTime y sus **clases** asociadas realiza un script que muestre:
    - Cuantas horas, minutos y segundos quedan para que sean las 11 de la noche.
    - Cuantos minutos y segundos quedan para cambiar de hora
    - Qué fecha será dentro de 5 días
    - Qué fecha fue hace 5 días
    - Calcula la edad que tiene una persona a partir de su fecha de nacimiento

    Puedes utilizar la página https://diego.com.es/fecha-y-hora-en-php para ver como se manejan los objetos. Aunque también debes acostumbrarte a entender la documentación https://www.php.net/manual/es/book.datetime.php

43. El el fichero “hora.php” define la siguiente clase.

    Clase Hora

    Crea una clase Hora con atributos para las horas, los minutos y los segundos de la hora Incluye, al menos, los siguientes métodos:

    - Constructor predeterminado con el 00:00:00 como hora por defecto. En el constructor se podrán indicar horas, minutos y segundos.
    - Asigna(): Permitirá asignar una hora al objeto.
    - EsValida(): comprobará si la hora es correcta; si no lo es la ajustará. Será un método auxiliar (privado) que se llamará en el constructor parametrizado.
    - A_Segundos(): devolverá el número de segundos transcurridos desde la medianoche.
    - De_Segundos(int): hará que la hora sea la correspondiente a haber transcurrido desde la medianoche los segundos que se indiquen.
    - Segundos_Desde(Hora): devolverá el número de segundos entre la hora y la proporcionada.
    - Siguiente($nSegundos): Avanzará *nSegundos* la hora, por defecto será 1 si no se indica otra cosa.
    - Anterior($nSegundos): Retrocederá *nSegundos* la hora, por defecto será 1 si no se indica otra cosa.
    - Copia(): devolverá un clon de la hora.
    - CargaHoraSistema(): Cargará la hora del sistema.
    - ToString: Devolverá la hora en una cadena con formato HH:MM:SS.
    - EsIgual(Hora): indica si la hora es la misma que la proporcionada.
    - EsMenor(Hora): indica si la hora es anterior a la proporcionada.
    - EsMayor(Hora): indica si la hora es posterior a la proporcionada.

    Funciones útiles, estáticas, que nos ayudarán a trabajar con el objeto:
    - ConvierteASeguntos($hora, $min, $seg): Devuelve el número de seguntos que hay.
    - ConviertaAHora($segundos): Devuelve un array('hora'=>h, 'min'=>m, 'seg'=>s) que contiene las horas, minutos y segundos que se corresponde a una cantidad de segundos. Si el nº de segundos es mayor que 84600 (1 día), desprecia el día.

44. Utilizando la clase Hora, creada anteriormente, realiza un programa, que nos permita crear una tabla de tareas con el siguiente formato:

Hora	Tarea

Las tareas a mostrar serán las que tengáis almacenadas en un array, como “tarea 1”, “tarea 2”, etc.
La hora de inicio será las 9:00, la de finalización las 15:00, y se mostrarán las horas en intervalos de 45 min.

45. Amplia el ejercicio anterior para que mediante un formulario podamos escoger la hora de inicio, la hora de fin, y el intervalo.

Depuración de aplicaciones usando Xdebug

PHP al igual que la mayoría de los lenguajes actuales dispone de herramientas que permiten depurar nuestros scripts. Estas herramientas permitirán comunicar nuestro IDE / Editor con nuestro servidor de forma que podremos realizar un análisis y ejecución de nuestro código paso a paso.

En la carpeta “*Depurar con Xdebug*” podremos encontrar artículos que nos indican como configurar nuestro entorno.

Principios de funcionamiento.

Para depurar nuestra aplicación es preciso que se establezca una comunicación entre nuestro entorno y el interprete de PHP. Para ello precisamos añadir librerías adicionales a nuestro interprete, las cuales permitirán dicha comunicación. Utilizaremos Xdebug debido a que está es gratuita.

Para configurar nuestro entorno precisaremos:

- Instalar / Copiar librerías de depuración donde se encuentre nuestro interprete de PHP. Precisaremos modificar el fichero de configuración “php.ini”
- Activar las librerías y probar que funcionan ejecutando la función de php “phpinfo()”
- Configurar nuestro entorno para que se conecte con nuestro interprete. Deberemos indicar la ubicación de nuestro servidor y el puerto que utilizarán para comunicarse.

Requisitos:

•XAMPP for Windows: https://www.apachefriends.org/download.html

•The VC14 builds require to have the Visual C++ Redistributable for Visual Studio 2015 x86 or x64 installed

•The VC15 builds require to have the Visual C++ Redistributable for Visual Studio 2017 x64 or x86 installed

Depurando con Visual Studio Code

- Visual Studio Code para PHP
- Debuggin – VS Code
- Youtube - Como depurar código PHP con Visual Studio Code (Configuración de XDebug)