UT2 - Inserción de código en páginas Web

Curriculum

- Mecanismos de generación de páginas Web. Lenguajes embebidos en HTML.
- Tecnologías asociadas: PHP, ASP, JSP, «Servlets», entre otras.
- Obtención del lenguaje de marcas para mostrar en el cliente.
- Sintaxis del lenguaje.
- Etiquetas para inserción de código.
- Directivas.
- Tipos de datos. Conversiones entre tipos de datos.
- Variables. Tipos. Ámbito de una variable.

Recursos

- Referencia del lenguaje:
  - http://php.net/
  - http://php.net/manual/es/index.php
- Tutoriales de PHP:
  - PHP Ya: https://www.tutorialesprogramacionya.com/phpya/ Tutorial con ejemplos y ejercicios
  - Programación en PHP: http://www.desarrolloweb.com/manuales/12/
  - Manual de PHP 5: http://www.desarrolloweb.com/manuales/58/
  - Aprende PHP en 12 capítulos: http://tutorialphp.net/
  - Páginas web con PHP http://www.mclibre.org/consultar/php/index.html
  - PHP Tutorial (Inglés): http://www.w3schools.com/php/
  - Tutorial de PHP: https://informaticapc.com/tutorial-php/
  - Libro: *Programador PHP Tomo 1 - Eugenia Bahit* (documento en pdf disponible en aula virtual)

Instalar el curso “PHP en la enseñanza” si se desea tener una guía rápida en nuestro equipo.

Contenidos

Instalación del servidor Web utilizando Xammp

- Qué es XAMPP - Cuadros cronológicos
- Instalación en GNU/Linux
- Instalación en Windows
- Configuración

2.1. Inserción de código en Páginas Web

Introducción
- ¿Qué es PHP?
- ¿Qué puede hacer PHP?

Un tutorial sencillo
- ¿Qué necesito?
- Su primera página con PHP

Nota: Recordad que nuestra aplicación generará código HTML que luego se mostrará en el navegador. Dicha visualización no siempre se ajustará a lo que vosotros esperáis, por lo que deberías revisar vuestro código con el propósito de ver que código HTML estáis generando. Para esto en el navegador tenéis la opción:

```
Ver código fuente
```

2.2. Etiquetas para Inserción de Código

Sintaxis básica
- Etiquetas de PHP
- Salir de HTML
- Separación de instrucciones
- Comentarios

Comentarios
Inclusión de código en páginas HTML
Funciones **echo** y **print**

2.3. Variables y tipos de datos

Utilizaremos la referencia del lenguaje

Definición y uso
Tipos
- Introducción
- Booleanos
- Números enteros (Integers)
- Números de punto flotante
- Cadenas de caracteres (Strings)
- Arrays
- Iterables
- Objetos
- Recursos
- NULO
- Llamadas de retorno (Callbacks / Callables)
- Seudotipos y variables usadas en esta documentación
- Manipulación de tipos

Conversiones entre tipos de datos - Manipulación de tipos
Ámbito de las variables

Operadores, expresiones, precedencia de operador:
Operadores
- Precedencia de operadores
- Operadores aritméticos
- Operadores de asignación
- Operadores bit a bit
- Operadores de comparación
- Operadores de control de errores
- Operadores de ejecución
- Operadores de incremento/decremento
- Operadores lógicos
- Operadores para strings
- Operadores para arrays
- Operadores de tipo

Variables variables

Variables Predefinidas

Estado de una variable: isset() , unset()
Constantes
- Sintaxis
- Constantes predefinidas

Expresiones

Operadores comparación y tipo: ===, !==
http://php.net/manual/es/language.operators.comparison.php

PHP es un lenguaje interpretado, por ese motivo debe procesar todo el código que vayamos a usar. El lenguaje incorporá funciones que permiten realizar un “copia y pega” en tiempo de ejecución. Las funciones son:
- require
- include
- require_once
- include_once

Las versiones modernas incluyen mecanismos más sofisticados para la inclusión de código.

Ejercicios propuestos

Notas:

- A todas las preguntas que se realizan debéis anotar vuestras respuestas como comentario.
- Llamad a los ficheros “XX_ejercicio.php”. Si en el enunciado se indican otros nombres añadid el prefijo “XX_”, donde XX será el número de ejercicio.
- Estas tareas son de obligada realización y deberán ser subidas a la moodle, donde se calificarán a *ojo de buen cubero*, es decir se revisarán superficialmente para verificar que váis trabajando y comprendiendo.
- Ante cualquier duda consulta al profesor o tu compañero. Crea un entorno colaborativo en el que todos os ayudéis.

1. Realiza una página web que muestre información sobre el interprete php que tienes funcionando (versión y librerías). Busca en la ayuda (php.net) lo que hace la función *phpinfo()*
   Identifica:
   - Versión de PHP
   - Servidor web que estás utilizando
   - Comprueba si alguna de las opciones que tienes configuradas en el fichero php.ini se corresponden con la información mostrada en la sección “Core”. Busca por el nombre “Directive”
2. Crea un nuevo script en en php en el que inicializaras distintas variables con diferentes tipos de datos y luego mostrarás sus valores.
   - Variables de tipo entero: expresa el número en decimal, octal, hexadecimal
   - Variable de tipo float: expresa el número con punto, en notación flotante, etc.
   - Variable de tipo cadena
   - Variable tipo booleano
3. Utilizando la función gettype() muestra el tipo que tiene cada una de las variables creadas.
4. Realiza una página en php en la que la misma variable la inicialices con distintos tipos de datos y vayas mostrando su contenido.
5. Realiza una página en la que realices distintas operaciones con los operadores disponibles. Muestra el resultado de distintas formas, comprueba el orden de evaluación.
   - Realiza suma/resta/multiplicació/división
   - Muestra el resto de una división
   - Divide dos enteros donde el numerador es menor que el denominador. Justifica la respuesta
   - Crea operaciones compuestas donde combines los operadores. Demuestra que con los paréntesis cambia el resultado al alterar la prioridad.
   - Concatena cadenas
   - Muestra operaciones con los operadores binarios
   - Muestra operaciones con los operadores lógicos. ¿Que diferencia existe entre los lógicos y binarios?
6. Realiza una página que muestre el mensaje “Hola Mundo”. Para ello concatena dos variables que contendrán el mensaje.
7. ¿Qué sucede si intentas mostrar una variable que no ha sido inicializada?
8. ¿Que sucede si sumas una variable inicializada y una no inicializada?
9. Realiza un programa que sume dos variables. Prueba su ejecución dando a las variables valores de distintos tipos de datos (entero, real, cadena, etc). Como por ejemplo:
   - 5 y 6.0
   - “7” y “9.0”
   - 2 y “hola”
   - 2 y “3lola”
10. A la hora de interpretar una página existen distintos niveles de errores. Lee el documento “Niveles de errores en PHP” para obtener más información.
11. Estudia la diferencia de comportamiento del operador “==” y el operador “===”. Para ello inicializa dos variables $v1, $v2 a los valores 1 y “1”, y luego comprueba utilizando la expresión echo $v1==$v2 y echo $v1===$v2 si dan el mismo resultado. Si lo deseas puedes utilizar el bloque if que seguro no tienes dificultad aunque no se haya explicado.
12. Crea un fichero “uno.php” en el que definirás 2 variables. En un segundo fichero “dos.php” incluirás el fichero “uno.php” y usarás dichas variables como si ya estuviesen declaradas en uno.
13. Utilizando la función *include* o *require* crea el fichero “layout.php” el cual contendrá la estructura de una página formada por varios bloques que estarán en distintos ficheros y que tendrá una presentación similar a la siguiente:

layout.php – Contiene código HTML con una tabla <table>

<table>
<tr><td colspan="2">Include titulo.php</td></tr>
<tr><td>Include menu.php</td><td>Include cuerpo.php</td></tr>
</table>

Incluye el contenido que quieras en todos los ficheros y prueba que la página para ver que se une todo en una misma página.

Nota: está forma de trabajar es la que utilizan las aplicaciones para rellenar el contenido de las páginas web, obteniendo el contenido de cada bloque de ficheros o bases de datos. Aunque actualmente ya no se utilizan tablas para maquetar los resultados.

UT3 - Programación basada en lenguajes de marcas con código embebido

Sumario

*** En construcción ***

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"

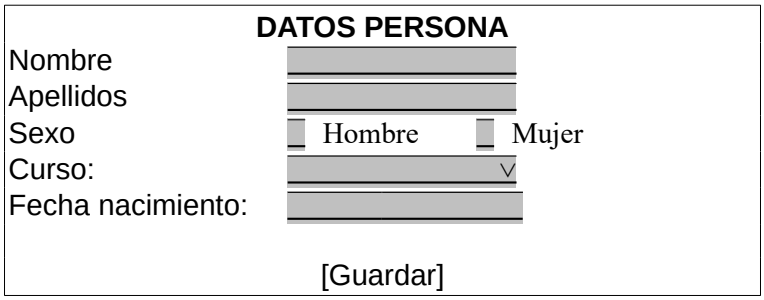
Procesar un formulario en la misma página / Autollamada de páginas

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 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');
      }
}
```

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

// Suponemos que no habrá errores
$errores=array();

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 4 – Formularios - Ejercicios propuestos

31. Amplia el ejercicio anterior 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:

    CALCULADORA

    Operador 1: 1
    Operador 2: 2
    Operación: Sumar
    Resultado: 3

    Seleccione operación
    [+ Sumar] [- Restar] [* Multiplicar] [/ Dividir]

    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 4 – 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 5 – 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 6 – 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

3.10 Orientación a objetos PHP

Lecturas

- 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”: http://www.phpya.com.ar/poo/index.php
- *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 7 – 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. 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.

43. 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.

44. 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

Nota: Solamente se abordan los entornos Netbeans y eclipse. Otros entornos soportan igualmente la depuración.

UT4 - Desarrollo de aplicaciones Web utilizando código embebido

Sumario

Curriculum

- Mantenimiento del estado. Sesiones. Cookies.
- Seguridad: usuarios, perfiles, roles.
- Mecanismos de autentificación de usuarios.
- Adaptación a aplicaciones Web: Gestores de contenidos y tiendas virtuales entre otras.
- Pruebas y depuración. Herramientas y entornos.

Contenidos

- Mantenimiento del estado. Sesiones. Cookies.
- Seguridad: usuarios, perfiles, roles.
- Mecanismos de autentificación de usuarios.
- Adaptación a aplicaciones Web: Gestores de contenidos y tiendas virtuales entre otras.
- Pruebas y depuración. Herramientas y entornos.

Estado de una aplicación - Sesiones en PHP

Véase el artículo: ¿Cómo Funciona El Sistema Web?

Cabeceras protocolo HTTP

En el protocolo HTTP, antes de enviar el cuerpo (HTML habitualmente) se envían cabeceras. Dichas cabeceras transportarán información de control que se puede utilizar de múltiples formas.

Entra la información que transmiten podemos destacar:

- Códigos de estado de las peticiones (Wikipedia).
- Envío de información sobre el tipo de datos que contiene el cuerpo.
- Envío de cookies.

Estas cabeceras se envían / reciben antes de que se envíe el cuerpo.

La función *header()* nos va a permitir poder modificar las cabeceras si nos resulta preciso. Entre las utilidades posibles destacamos:

- Redirigir a otra página
- Forzar descarga de archivos: Artículo que nos indica como conseguir que desde un script php se degargue un fichero, indicando el tipo, el tamaño y el nombre que nos mostrará.
- Generar un error 404 desde PHP: Si queremos podemos simular que al ejecutar un script devuelva el error 404 (recurso no encontrado).
- Códigos de ejemplo para redirigir tu sitio web

Si no funcionan los enlaces véase fichero "Cabeceras PHP.odt"

Trabajando con sesiones

Las sesiones utilizarán las cabeceras del protocolo HTTP para su funcionamiento. Por este motivo no se podrá haber enviado "nada" antes de incializar la sesión.

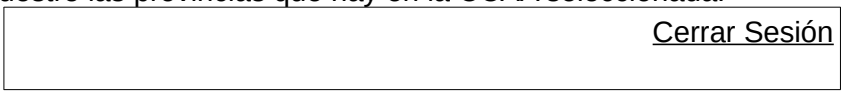
La sesión es la herramienta que nos permite simular la conexión en un protocolo desconectado como es HTTP.

- Manejo de sesiones en PHP. Tutorial sencillo:
- Funciones de sesión, manual PHP:
- Manejo de sesiones
- Tutorial básico para el manejo y control de sesiones en PHP: Contador de visitas a página
- Validación de usuario para acceder a una página: Control de sesiones en PHP:

Si no funcionan los enlaces véase fichero "Sesiones – Manejo de Sesiones.odt"

Ejercicios propuestos - Sesiones

1. Realiza una página que cuente el número de veces que ha sido visitada desde un navegador y que lo muestre por pantalla.
2. Cuestión teórica: ¿Qué cuenta el ejercicio anterior, el número de visitas o el número de visitas desde un ordenador con un navegador?. Para responder a tu pregunta consulta la página utilizando diferentes navegadores. Intenta igualmente acceder desde otro equipo (utilizando la IP de tu equipo), para ver el resultado. (Puedes también utilizar navegación privada o ventana de incógnito).
3. Realiza una página que permita seleccionar la comunidad autónoma en un formulario y nos muestre las provincias que hay en la CCAA seleccionada.



Seleccione CCAA: [Ver provincias]

Después de enviar el formulario se mostrará el nombre de la comunidad autónoma seleccionada y las provincias que la conforman.
La primera vez que se muestre la página se nos preguntará el nombre de usuario y clave con los que nos conectaremos a la base de datos. Las siguientes no se solicitará dicha información. Solamente, después de pulsar el enlace “Cerrar sesión” se solicitará de nuevo el usuario y clave.

Cookies

Una cookie es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario.

Sus principales funciones son:

- Llevar el control de usuarios: cuando un usuario introduce su nombre de usuario y contraseña, se almacena una cookie para que no tenga que estar introduciéndolas para cada página del servidor. Sin embargo, una cookie no identifica solo a una persona, sino a una combinación de computador-navegador-usuario.
- Conseguir información sobre los hábitos de navegación del usuario, e intentos de spyware (programas espía), por parte de agencias de publicidad y otros. Esto puede causar problemas de privacidad y es una de las razones por la que las cookies tienen sus detractores.

Las cookies son creadas y gestionadas por el navegador. Se pueden crear y modificar desde el servidor, enviando las cabeceras apropiadas. Desde Javascript también podremos crear y consultar las cookies que tengamos almacenadas en nuestro navegador.

Contenido

- Cookies Wikipedia
- Manual PHP
  - setcookie(): http://php.net/manual/es/function.setcookie.php
  - $_COOKIE: http://www.php.net/manual/es/reserved.variables.cookies.php
- Explicaciones completas sobre cookies en PHP, con todos los parámetros de la función setcookie() y el array $_COOKIE
  http://www.desarrolloweb.com/articulos/cookies-en-php.html
- Cookies con PHP y Javascript:
  http://copstone.com/2010/03/cookies-con-php-y-javascript/
- PHP Cookies W3Schools
- Cookies en javascript
  - Como utilizar cookies en JavaScript
  - W3Schools - JavaScript Cookies
  - MDN Web Docs - HTTP cookies
- Autenticar usuario y guardar en una cookie con PHP

- Sistema de autenticación de usuarios en PHP con cookies

Ejercicios propuestos - Cookies

4. Realiza una página que cuente el número de veces que ha sido visitada desde un navegador y que lo muestre por pantalla. Para llevar la cuenta utilizará cookies.
5. Cuestión teórica: ¿Qué cuenta el ejercicio anterior, el número de visitas o el número de visitas desde un ordenador con un navegador?. Para responder a tu pregunta consulta la página utilizando diferentes navegadores. Intenta igualmente acceder desde otro equipo (utilizando la IP de tu equipo), para ver el resultado.
6. Realiza una página que muestre un mensaje de bienvenida la primera vez que se visita. Las siguientes veces que se acceda a esa página automáticamente mostrará la hora del sistema.

1ª vez	Siguientes
Bienvenido a la página XXXX. Pulse aquí para continuar.	Hora del sistema HH:MM:SS

7. Realiza una página que muestre un mensaje de bienvenida la primera vez que se visita con la opción de omitir o no el mensaje en visitas posteriores. Las siguientes veces en función de si desea mostrar el mensaje de bienvenida o si desea omitirlo se mostrará la página de bienvenida o no.

1ª vez	Siguientes
Bienvenido a la página XXXX. ☐ Seleccione la opción si no desea volver a ver el mensaje de bienvenida. [Continuar]	Hora del sistema HH:MM:SS

Documentar aplicaciones PHP

Conforme va creciendo una aplicación, se va complicando el manejo del proyecto, por ejemplo para que sirva cada clase, cada método, cada atributo, y no solo para el desarrollador, sino para todos los colaboradores y así ellos puedan entender partes del código sin necesidad de analizar el código línea a línea.

El estándar de programación, define las reglas para escribir y documentar código, además de cómo se comunican las diferentes piezas de código desarrolladas por diferentes equipos. El objetivo de esto es que parezca que el código ha sido escrito por una única persona.
De allí la importancia de documentar el código.

¿Qué es PHPDoc? (Wikipedia)

Wikipedia dice: es una adaptación de javadoc para php que define un estándar oficial para comentar código php, con las siguientes características:

- •Hace comentarios que pueda leerse en un método estándar para animar a los programadores a definir y comentar los aspectos del código que normalmente se ignoran.
- •Permite que los generadores de documentos externos como phpDocumentor puedan crear la documentación API en buen formato y fácil de entender.
- •Permite que algunos IDEs como Zend Studio, NetBeans y Aptana Studio interpreten los tipos de variables y otras ambigüedades en el lenguaje de programación.

Todo esto a través de Docblocks, un Docblock no es más que un bloque para documentar un bloque de código. Comienza con / ** y tiene un asterisco al principio de cada línea. Ejemplo:

```
/**
 * Saluda al visitante
 *
 * Une la palabra hola con el nombre del visitante
 *
 * @param string $nombre nombre del visitante
 * @return string saludo completo
 */
function saludo($nombre) {
    return 'hola ' . $nombre;
}
```

¿Como generamos la documentación?

Al igual que java con JavaDoc para generar la documentación de nuestro proyecto utilizaremos una herramienta auxiliar que nos genere automáticamente la documentación a partir de nuestros comentarios en el código fuente.

La herramienta que utilizaremos será APIGen. Aunque si tenemos problemas con ella podremos usar DoxyGen que soporta la generación de documentación en múltiples lenguajes.

En la carpeta Documentar en PHP podéis ver como se puede ejecutar APIGen de forma autonoma o arrancandolo desde nuestro IDE (Netbeans), aunque ahora sin duda alguna la forma más sencilla de incorporarlo a nuestro proyecto es usando composer, como puede ver apigen/apigen, o en GitHub

Seguridad: usuarios, perfiles, roles.

Seguridad en una aplicación -Control de acceso

Con el control de acceso pretendemos impedir que usuarios no autenticados accedan a las páginas reservadas de nuestra aplicación.

- Sistema de autentificación PHP
  http://www.desarrolloweb.com/manuales/37/

- **1.- Funcionamiento del sistema de autentificación en PHP**
  Descripción de las distintas páginas que forman el sistema de autentificación y su funcionamiento, basado en usuario y contraseña.
- **2.- Página inicial con el formulario de autentificación en PHP**
  Página que muestra el formulario donde el visitante debe introducir su nombre de usuario y contraseña, necesarios para acceder a la aplicación segura.
- **3.- Control de los datos de autentificación en PHP**
  Aquí veremos si los datos de autentificación son correctos y dependiendo de si lo son o no, se redirigirá al navegador a la aplicación segura o al formulario inicial, respectivamente.
- **4.- Capa de seguridad en PHP**
  Este capítulo nos explicará el funcionamiento del módulo de seguridad.
- **5.- Archivos de la aplicación con acceso restringido en PHP**
  Mostraremos un ejemplo del código de una de las páginas web que formaría la aplicación segura.
- **6.- Salir de la aplicación segura en PHP**
  En este capítulo veremos con un sencillo ejemplo como salir de la aplicación de accesi restringido.
- **7.- Diferentes formas de cerrar sesión en PHP**
  Cómo cerrar una sesión autenticada correctamente, por inactividad o por cierre del navegador por parte del usuario.
- **8.- Cierre de sesión al cerrar el navegador en PHP**
  Código en PHP para cerrar sesiones cuando se cierra el navegador.
- **9.- Autentificación PHP para múltiples usuarios usando MySQL**
  Página PHP que necesitaríamos para realizar un acceso restringido por clave y contraseña para múltiples usuarios, donde cada uno tenga unos datos de acceso propios, que se guardan en la base de datos.
- **10.- Autenticar usuario y guardar en una cookie con PHP**
  Sistema de autenticación de usuarios en PHP, donde se ofrece la opción de memorizar su usuario en el ordenador y guardar una cookie para recordar el usuario y no tener que volverse a autenticar. Entrar

Evitar ataques

- Encriptar y guardar contraseñas en una base de datos
- Como Almacenar Passwords En Una Base De Datos
- Login mediante cookies

UT6 - Utilización de técnicas de acceso a datos

Sumario

Curriculum

- Establecimiento de conexiones.
- Recuperación y edición de información.
- Utilización de conjuntos de resultados.
- Ejecución de sentencias SQL.
- Transacciones.
- Utilización de otros orígenes de datos.

Contenidos

En PHP podremos acceder a múltiples tipos de base de datos (MySQL, MariaDB, Oracle, PostgresSQL, etc). Cada base de datos tendrá su correspondiente controlador/conector que nos proporcionará las funciones para acceder a la misma.

Contenidos generales

- *Véase la presentación "ut6 - Acceso a BBDD - mysql"*

- *Véase el apartado "Bases de datos en PHP" del curso "Principios básicos para la programación en PHP" – Nota: Aquí se mezcla la presentación y el código, algo que deberemos evitar más adelante*
- *Véase los temas "Trabajando con Bases de Datos MySQL" y "Trabajando con MySQL desde PHP" del libro "Programador PHP Tomo 1 - Eugenia Bahit.pdf"*

Herramientas

Las herramientas siguientes nos permitirán interactuar con el sistema gestor de base de datos, con el objeto de crear esquemas, usuarios, tablas, realizar consultas, etc.

- *PhpMyAdmin*: Aplicación en PHP que permite realizar operaciones sobre bases de datos en MySQL.
  http://www.phpmyadmin.net/
- *MySQLWorkBrench*: Aplicación de MySQL que permite administrar una base de datos.
  https://www.mysql.com/products/workbench/
  A partir de la versión 8.0.19 de Workbench puede que requiera una conexión segura a través de SSL algo que no viene configurado por defecto en XAMPP, puede que resulte interesante instalar esta versión, la cual podremos obtener en
  https://downloads.mysql.com/archives/workbench/

Existen otros productos en el mercado que permiten realizar igualmente estas operaciones. Se han seleccionado los siguientes por ser el primero el que se incluye en la mayoría de los hosting de internet, y el segundo ser la aplicación diseñada por el propio fabricante que incorpora además opciones de modelado.

Por motivos didácticos, y para tener conocimiento de aplicaciones antiguas, aunque ya obsoletas se explicará el acceso a la base de datos utilizando la API MySQL original "*mysql_*" que actualmente está obsoleta y no se recomienda su utilización.

Fácilmente se puede utilizar la nueva extensión MySQL mejorada (desde versión MySQL 4.1) añadiendo el prefijo "mysqli_" a las mismas funciones que obsoletas.

La nueva extensión aporta mejoras que debemos también considerar.

Nota: Debéis observar que en las nuevas funciones siempre es preciso pasar como parámetro el enlace "$link" a la base de datos con la que estáis trabajando y que obtendréis en la conexión.

```
mixed mysqli_query(mysqli $link,string $query[,int
        $resultmode= MYSQLI_STORE_RESULT] )
```

Puede consultar la lista de funciones disponibles en:
http://www.w3schools.com/php/php_ref_mysqli.asp

Accediendo a MySQL (MaríaDB) de forma nativa

Véase PHP.NET: Extensión MySQL mejorada

- Interfaz dual: procedimental y orientada a objetos
- Conexiones

- Ejecutar sentencias
- Sentencias Preparadas

También hay ejemplos en W3Schools.com - PHP MySQL Database

Ejercicios propuestos

Bloque Consultas

1. Crea la base de datos “bdprovincias” en la que incluiréis los datos contenidos en el fichero “provincias.sql”.
2. Realiza una página en PHP que muestre las provincias que hay en España leyendo los datos de la base de datos.
3. Realiza una página en PHP que nos diga cuantas provincias tiene cada CCAA.
4. Realiza una página en PHP que muestre las provincias que tiene cada comunidad autónoma, con el siguiente formato:

   CCAA1 : prov1, prov2, …

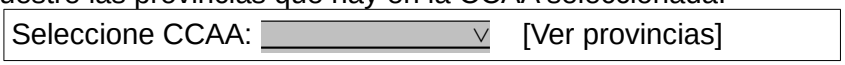
   CCAA2: …..

5. Realiza una página en PHP que muestre las provincias que tiene cada comunidad autónoma, en una tabla con el formato:

CCAA	Provincias
Andalucía	Almería
	Cadiz
	...
	Sevilla

> Cuando resolvemos problemas siempre es conveniente separar la lógica de negocio (obtener los datos de la base de datos, en este caso) de la presentación (mostrar la página web). Por este motivo es conveniente que por un lado obtengáis los datos y los almacenéis en un array, y por otro generéis el código HTML que muestre los datos.

6. Realiza una página que permita seleccionar la comunidad autónoma en un formulario y nos muestre las provincias que hay en la CCAA seleccionada.

   Después de enviar el formulario se mostrará el nombre de la comunidad autónoma seleccionada y las provincias que la conforman.

Bloque Modificaciones

> Habitualmente nos estamos conectando desde múltiples páginas a la misma base de datos en una aplicación, por lo que estaremos incluyendo redundantemente los datos de conexión en cada una de las páginas en las que nos conectamos. Por este motivo puede ser conveniente la creación de un fichero en php que incluya toda la información sobre la conexión
>
> **-- bbdd_conex.inc.php –**
> <?php

```
    $host='localhost';
    $user='root';
    $passwd='';
    $conex=mysql_connect($host, $user, $passwd) or die('Error BBDD');
-- otro_fichero.php –
<?php
    include ('bbdd_conex.inc.php');
    // resto del código
```

Más adelante se abordará este problema desde un enfoque de POO, lo que nos creará soluciones más elegantes.

7. Realiza una página que nos permita añadir nuevas provincias a nuestra base de datos. Las provincias creadas estarán ubicadas en una nueva comunidad autónoma que se llamará "Nuevas provincias".

   La entrada será filtrada, de forma que no se permitirá que introduzcan provincias repetidas o que se introduzcan cadenas en blanco. Igualmente el nombre de la provincia no deberá contener ningún dígito. Véase el artículo "Expresiones regulares en PHP" o el o el minimanual Explicaciones y ejemplos para el manejo de expresiones regulares.
   El programa tendrá un formulario similar al siguiente:

   Provincia: _____ [Añadir] Ver todas las provincias

   Después de cambiar, filtrando que los datos sean correctos, se seguirá preguntando para seguir cambiando el nombre.

   Donde desde el enlace "Ver todas las provincias" mostraremos todas las provincias y comunidades autónomas que tenemos almacenadas.

8. Realiza una página que nos permita modificar el nombre de las provincias.
   El programa tendrá un formulario similar al siguiente:

   Provincia: _____ Nuevo nombre:_____[Cambiar]
   Ver todas las provincias

   Donde desde el enlace "Ver todas las provincias" mostraremos todas las provincias.

Para la creación de los campos *select* resulta interesante crear una función en php que cree el campo en html utilizando los datos de un array. La función podría tener el siguiente formato :
*function creaSelect($nombreCampo, $valores /*array*/, $seleccionado='') {…}*

Luego la usaríais

```
<html>
…
<?php echo creaSelect('numeros', array('1'=>'uno', '2'=>'dos',...), $_POST['numeros']); ?>
```

9. Cuestión teórica: ¿Qué pasa si en el nuevo nombre de provincia incluís comillas simples o dobles en la cadena? Haz la prueba.
   Busca información sobre "inyección de código en SQL" (Wikipedia), para obtener una información más precisa sobre el problema.

Lee información sobre las funciones interesantes para el manejo de cadenas en SQL del que se incluye información mas adelante.

10. Realizar una página que nos permita borrar una provincia de la tabla. Cuando borramos es conveniente realizar una confirmación del borrado, por lo tanto el proceso se hará de la siguiente manera:

Formulario 1	Provincia: [▽] [Borrar] Ver todas las provincias
Formulario 2	¿Desea borrar la provincia XXXXXX? [Sí] [No]
Si pulsan [No] volvemos a Formulario 1, Sino eliminamos y vamos a Formulario 3	
Formulario 3	Se ha procedido a borrar la provincia XXXXXXX Borrar otra provincia - Ver todas las provincias

Vais a necesitar campos ocultos para solucionar el problema.

11. Ahora deseamos realizar el ejercicio anterior pero con otra presentación. El objetivo es poder borrar o modificar una provincia pero sin tener que seleccionarla en ningún campo del formulario. El procedimiento de trabajo será el siguiente:

    Se mostrarán todas las provincias y se dará la opción de borrarla o modificarla

LISTADO DE PROVINCIAS - Provincia 1 Modificar / Borrar - Provincia 2 Modificar / Borrar ...

    Para realizar esto deberéis utilizar parámetros de tipo GET en los que notificaréis la provincia con la que deseáis trabajar. En la operación de borrado se precisa confirmación igual que en el ejercicio anterior.

Seguridad en las consultas – Inyección de SQL

Cuando realizamos aplicaciones que trabajan con base de datos, debemos contemplar la posibilidad de que los datos que vengan del cliente no tengan el formato que esperamos. Debiendo ser muy cuidadosos sobre como trasladamos dichos datos a nuestra consulta SQL. No olvidéis que a fin de cuentas una consulta SQL no deja de ser una cadena de texto.

Véase:

- Inyección SQL - Wikipedia
- PHP.net Inyección de SQL
- Un ataque de inyección de SQL
- Inyecciones SQL – Otro ejemplo
- *Véase el ejemplo "bbdd_sql_inyeccion_de_codigo.php" e intente que se muestren todas las provincias incluyendo texto en el cuadro de búsqueda*

Las siguientes funciones resultan interesantes para evitar inyección de código

- *Addslashes*: Añade barras invertidas a una cadena
- *mysqli_escape_string*: Escapa una cadena para incluirla en una sentencia SQL

Por eficiencia y seguridad es mejor utilizar sentencias preparadas.

Sentencias preparadas

Las bases de datos MySQL soportan sentencias preparadas. Una sentencia preparada o una sentencia parametrizada se usa para ejecutar la misma sentencia repetidamente con gran eficiencia.

Flujo de trabajo básico

La ejecución de sentencias preparadas consiste en dos etapas: la preparación y la ejecución. En la etapa de preparación se envía una plantilla de sentencia al servidor de bases de datos. El servidor realiza una comprobación de sintaxis e inicializa los recursos internos del servidor para su uso posterior.

El servidor de MySQL soporta el uso de parámetros de sustitución posicionales anónimos con ?.

Léase los siguientes enlaces

- "Sentencias preparadas en PHP"
- PHP Prepared Statements
- PHP.NET Sentencias Preparadas

Véase

- Sentencias Preparadas en PHP
- Sentencias preparadas de MySQL en PHP (ejemplos orientados a objetos)
  Consultas preparadas y escapado
  http://programandolo.blogspot.com.es/2013/06/extension-mysqli-parte-3-consultas.html

Ejercicios propuestos – Bloque sentencias preparadas

12. Modifica varios de los ejercicios que has realizado anteriormente de forma que se utilicen consultas preparadas. Realiza un ejemplo de consulta, otro de inserción y otro de actualización.

Limitar los resultados de una consulta SQL

Véase el artículo: SQL-Limitando registros recuperados en una Consulta

Paginación de resultados de una base de datos.

La paginación de resultados nos permite mostrar grandes cantidades de datos en una aplicación web. El objetivo mostrar los datos que devuelve una operación, normalmente una consulta sobre una base de datos, de forma organizada, tratando en todo momento de evitar que se realicen cálculos o movimientos de datos innecesarios.

Se tendrán las siguientes variables:

- Nº de resultados por página: Puede ser un valor fijo en programación o configurable de diferentes formas.
- Nº de página a mostrar

Enlaces

- Véase el artículo: "Paginación de resultados" en el que se incluyen tres artículos sobre creación de páginación.

- Paginación de resultados con PHP y MySQL:
  En este tutorial **se calcula improductivamente el número de elementos** que hay que mostrar, deberíamos utilizar una consulta "*select count(*)...*" en lugar de traer todos los registros para saber cuantos son.
  http://www.desarrolloweb.com/articulos/1035.php

> No es conveniente utilizar la función *mysqli_num_rows(..)* o el atributo del mismo nombre si estámos utilizando la versión O.O. debido a que es improductiva y puede devolver resultados imprecisos.
> Para saber el número de resultados devueltos por una consulta deberiamos realizar un:
> `select count(*) where -condición-`

- Buscar en Google más artículos...

Cuando estamos paginando resultados al mostrar una tabla de una base de datos es muy poco productivo, casi podría considerarse un error, traer de la base de datos todos los registros para luego descartarlos

```
$ssql = "select * from pais " . $criterio;
$rs = mysqli_query($conn,$ssql);
$num_total_registros = mysql_num_rows($rs);
```

Luego ...

```
$ssql="select * from pais ".$criterio." limit ".$ini.",".TAM_PAG;
$rs = mysqli_query($conn, $ssql);
while ($fila = mysqli_fetch_object($rs)){
      echo $fila->nombre_pais . "<br>";
}
```

Cada vez que invocamos a `mysqli_query()` traemos todos los registros, pero la primera vez los ignoramos, pues solo nos importa el n.º de registros. Es mucho mejor, y menos costoso realizar

```
$ssql = "select count(*) as total from pais " . $criterio;
$rs = mysqli_query($conn,$ssql);
$reg= mysqli_fech_assoc($rs);
$num_total_registros = $reg['total'];
```

Ejercicios propuestos – Bloque paginación

13. Muestra la lista de países existentes en la base de datos países, de tal forma que solo se muestren 20 países de una vez. Crea en tu página un sistema de paginación que te permitirá avanzar o retroceder para mostrar los siguientes países.

Abstracción de la base de datos utilizando objetos

Cuando accedemos a la base de datos nos interesa abstraer y abordar con una metodología orientada a objetos. Esto puede ser realizado de forma simple, con una clase que nos abstraiga de un proveedor determinado.

Léanse los documentos ubicados en la carpeta *artículos*

- *BD-1-Crear una clase para conectar a base de datos con php (En el documento se ha modificado el original)*
- *BD-1-Crear una capa de conexión abstracta a base de datos con PHP*
- *BD-3-Creando_una_capa_de_abstracción_con_PHP_y_mysqli-Eugenia_Bahit*

Estos mecanismos los podréis encontrar en muchas aplicaciones antiguas. En las modernas la extensión PDO, de la cual hablaremos más adelante, y que es estándar en PHP consigue los mismos resultados.

Patrones de diseño

Patrón de diseño Singleton

Wikipedia: El patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

Este patrón será muy útil en nuestro propósito de crear una única conexión a la base de datos.

- El patrón singleton con PHP
- El patrón de diseño Singleton (implementado en PHP)

Ejercicios propuestos – Patrón singletón

14. Modifica varios de los ejercicios que has realizado anteriormente de forma que utilicen la nueva clase de acceso a datos. Esta clase estará definida en el fichero “db.php”. Se añadirá a la clase nueva funcionalidad si se precisa..

    En alguno de los artículos o ejemplos posteriores ya está codificada la clase “db”

Patrón de diseño DAO

Normalmente como hemos visto en el ejemplo anterior nos interesa desacoplar el acceso a la base de datos de la lógica de nuestro programa. De esta forma ante posibles cambios de sistema gestor de base de datos, de MySQL a PostgresSQL por ejemplo, tendremos localizado el código que hay que modificar, lo que facilitará la migración.

Otro de los patrones que nos pueden ayudar en esta labor es el patrón de diseño DAO (Data Access Object – Wikipedia)

Léase el artículo “Patrón de diseño DAO“ el cual contiene enlaces las siguientes páginas:

- Descripción conceptual:
  - DAO
  - ABSTRACCIÓN DE DATOS (I): EL PATRÓN DAO
- En PHP
  - DAO (Data Access Object) – PHP
  - Patrón DAO en PHP (Data Access Object in PHP): Observad que en PHP no es preciso crear la clase VO (Value Object) o TO (Transport object), pues el lenguaje crear dichas estructuras al vuelo.

Otro patrón de diseño para acceder a los datos es el Active Record del cual podréis ver una comparativa con el DAO en este artículo: Patrones de Diseño (Active Record vs DAO)

Videotutoriales:

- Implementación de Patrón de diseño DAO con PHP
- El patrón DAO (Data Access Object). Manteniendo la persistencia de datos | | UPV
-

PDO - Objetos de Datos de PHP

Dentro de PHP existen múltiples librerías que nos permiten abstraer la conexión a la base de datos, permitiendo que nuestra aplicación pueda cambiar de base de datos sin que haya que apenas realizar modificaciones en nuestro código.

Una de las formas es utilizando la librería PDO que viene incluida en PHP desde la versión 5.

Eche un vistazo al manual de referencia - Objetos de datos de PHP (PHP.net)- o véase el documento "PDO - Acceso a base de datos" que incluye varios artículos que tratan el acceso de es forma.

- Tutorial de PDO
- PHP Data Objects (PDO) – Ejemplo de conexión a varias bases de datos diferentes
- Consultas a bases de datos desde PHP con PDO (ejemplo simple)
- Acceso a bases de datos con PDO – Muy completo
- PDO PHP ejemplos con clases – Aquí podéis ver un ejemplo que utiliza clases para acceder a la BBDD

Manejo de errores - Mecanismo de excepciones

PDO utiliza las excepciones para indicar cualquier tipo de error. El procesamiento de las excepciones en PHP es similar a otro tipo de lenguajes como Java, C#, etc.

- Fundamentos de programación/Manejo de errores
- php.net – Excepciones
- MANEJO DE EXCEPCIONES EN PHP
- Excepciones en PHP

Transacciones

Cuando tenemos un conjunto de consultas de inserción o borrado, si no podemos realizar alguna de ellas, que las anteriores que sí se hayan podido realizar no tengan efecto o no se registren en la base de datos. En otras palabras, un conjunto de sentencias se comportan como una sola unidad, o se ejecutan todas o ninguna. Esto es más o menos la definición de transacción.

Este comportamiento sobre acciones en la base de datos lo podemos conseguir mediante el uso de transacciones, las cuales tendrán tresoperaciones básicas en cualquier SGBD:

- Comienzo de la transacción (Begin Trannsaction)
- Fin de la transacción correcto (Commit Trannsaction) Todas las operaciones quedan en firme.

- Fin de la transacción incorrecto (Rollback Trannsaction) Se deshacen todas las modificaciones que se hayan realizado.

MySQL y otras bases de datos nos permite realizar una gestión manual de transacciones.

- TRANSACCIONES MYSQL CON PHP
- Transacciones en MySQL en PHP
- Soporte de la API para transacciones MySQLi (PHP.net)
- Transacciones PDO (PHP.net)

Ultimo identificador insertado

Es una práctica recomendable crear un campo llamado “id” de tipo entero y con el modificador autoincremento para que sea la clave principal de nuestra tabla. Independientemente de que tengamos otros campos candidatos a ser clave principal.

El motivo que justifica esta decisión es

```
CREATE TABLE Persons (
      ID int NOT NULL AUTO_INCREMENT,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int,
      PRIMARY KEY (ID)
);
```

El motivo que justifica esta decisión de diseño es que dicho campo, al no ser un atributo del dominio que estamos modelando nunca cambiará y nos permitirá identificar los registros de las tablas.

Pero, que pasa cuando insertamos un nuevo registros

```
INSERT INTO Persons (FirstName,LastName) VALUES ('Lars','Monsen');
```

¿Que valor toma el campo id?

En MySQL existe una sentencia ***LAST_INSERT_ID()*** que nos devuelve el último identificador insertado en la tabla relacionada. La librería MySQLI nos proporciona tanto una función como un atributo de la clase MySQLI que nos permiten obtener dicho identificador. La sintaxis es la siguiente:

Estilo por procedimientos:

```
mixto mysqli_insert_id (mysqli $identificador_de_conexion)
```

Ejemplo de uso:

```
$fk_pedidos = mysqli_insert_id($db);
```

Estilo orientado a objetos (atributo):

Ejemplo anterior:

```
$fk_pedidos = $db->insert_id;
```

Más información en el manual de PHP: http://docs.php.net/manual/es/mysqli.insert-id.php

Esta funcionalidad estará disponible también para otras bases de datos con una función similar. PDO (PDO::lastInsertId) incluye también esta funcionalidad

En las tablas que definamos en nuestros problemas/trabajos es conveniente poner siempre como clave principal un identificador (id) de tipo autonumérico.

Aunque conceptualmente fuese aconsejable utilizar otro campo clave, según la teoria del diseño de BBDD, por cuestiones de simplicidad en la creación de la aplicación es preferible usar como clave principal un campo que no tenga ningún significado en el dominio del problema.

Las claves principales “conceptuales” les asociaremos un indice único para garantizar su unicidad.