- **Clase 1:** 10/08/2016

Introducción al curso

Abordar los temas a través del desarrollo de una aplicación, en donde vamos a ir a medida que avanzamos utilizando los temas a cubrir.

Herramientas

Xampp:

Apache: El servidor HTTP Apache es un servidor web HTTP de código abierto

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa.

PHP:

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy, lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico, como Facebook, para optar por el mismo como tecnología de servidor.

MySQL:

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo

IDE:

Netbeans – Aptana - Sublime Text

Bootstrap:

es un framework o conjunto de herramientas de Código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones,

cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales

Javascript: para programas eventos.

Modelo Vista Controlador- MVC

El modelo–vista–controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.1 2 Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento

De manera genérica, los componentes de MVC se podrían definir como sigue:

- El **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.[12]
- El **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo' (véase ***Middleware***).
- La **Vista**: Presenta el 'modelo' (información y *lógica de negocio*) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Estilos de codificación

Documentación del código

En proyectos medianos y grandes, participan muchos desarrolladores, aportando modulos y funcionalidades, siendo cada uno responsable de la desarrollado, dentro de esta responsabilidad se encuentra la de mantener el código en buena forma, hacerlo entendible, en lo posible, a otros desarrolladores en el futuro.

En principio se debería redactar código limpio, siguiendo entre otras las siguientes reglas:

Uso de identación («sangrado». Este término significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores, para así separarlo del margen izquierdo y mejor distinguirlo del texto adyacente, utilizado para mejorar la legibilidad del código fuente)

Ancho razonable de las líneas de código.

Utilización de espacios cuando corresponda.

Continuación del estilo inicial de programación (orientado a objetos, procedural, etc)

La documentación del código se divide en dos partes: la inclusión de comentarios dentro de los archivos que forman parte de nuestra aplicación y la escritura de manuales o guías por fuera del código.

- **Clase 2:** 17/08/2016

PHP Básico

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje "open source" interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

Directivas Generales de Configuración

display_errors boolean
Determina si los errores se visualizan en pantalla como parte de la salida en HTML o no.

include_path string
Especifica una lista de directorios en los que las funciones require(), include() y fopen_with_path()buscan los archivos. El formato es similar a la variable de entorno de sistema.
PATH: una lista de directorios separados por dos puntos en UNIX o por punto y coma en Windows.

Ejemplo include_path en UNIX
include_path=.:/home/httpd/php-lib
Ejemplo include_path en Windows
include_path=".;c:\www\phplib"
El valor por defecto para esta directiva es.(sólo el directorio actual).

max_execution_time integer
Fija el tiempo máximo en segundos que se le permite usar a un script antes de ser finalizado por el intérprete. Así se evita que scripts mal escritos puedan bloquear el servidor.

memory_limit integer

Fija el tamaño máximo de memoria en bytes que se permite reclamar a un script. Así se evita que script mal escritos se coman toda la memoria dispomible de un servidor.

short_open_tag boolean
Indica si se debe permitir el formato corto (<? ?>) de la etiqueta de apertura del PHP. Si desea utilizar PHP en combinación con XML, deberá desactivar esta opción. Si está desactivada, deberá utilizar el formato largo de la etiqueta de apertura (<?php ?>).

upload_tmp_dir string
El directorio temporal utilizado para almacenar archivos cuando se envían al servidor. Debe tener permiso de escritura para el usuario bajo el que corra el PHP.

extension_dir string
En qué directorio debe buscar el PHP las extensiones cargables dinámicamente.

Directivas de Configuración de MySQL

mysql.allow_persistent boolean
Si permitir o no conexiones MySQL persistentes.

mysql.max_persistent integer
El número máximo de conexiones persistentes de MySQL por proceso

Sintaxis básica

Para interpretar un archivo, php simplemente interpreta el texto del archivo hasta que encuentra uno de los caracteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Tipos de etiquetas

Etiqueta estándar (Standard Tags): Son estas las etiquetas de apertura y cierre de facto; son la mejor solución para la portabilidad y compatibilidad hacia atrás, ya que están garantizados para estar disponible y no se pueden desactivar cambiando el archivo de configuración de PHP.

```
<?php echo 'Estas etiquetas permiten el uso de PHP en codigo XML como XHTML'; ?>
```

Etiquetas cortas (Short Tags): fueron por mucho tiempo el estándar en el mundo de PHP; Sin embargo, tienen el inconveniente principal en el conflicto con instrucciones de procesamiento XML (por ejemplo, <? xml) y por tanto, ya no se recomiendan y se han reducido un poco en el camino, hoy son consideradas una mala práctica.

```
<? echo "Hola Mundo!!" ?>
```

Echo Tags: Desde la introducción de etiquetas cortas, también conocidas como “etiquetas de eco”, permiten imprimir el resultado de una expresión directamente a la salida de la secuencia de comandos. Con el lanzamiento de PHP 5.4, las etiquetas cortas y etiquetas de eco se dividieron, y las etiquetas de eco están siempre habilitadas.

```
<?= 'mostrar este string' ?>
```

Si el archivo php solo contiene código php, es una buena práctica omitir la etiqueta de cierre php para prevenir salidas accidentales.

Métodos avanzados de escape

```
<?php
if ($expression) {
?>
<strong>This is true.</strong>
<?php
} else {
?>
<strong>This is false.</strong>
<?php
}
?>
```

Para escribir bloques grandes de texto generamente es más eficiente separalos del código PHP que enviar todo el texto mediante las funciones echo(), print() o similares.

Comentarios

```
/*
     Escribir texto entre una barra-asterisco y asterisco-barra
     crea un comentario multilínea.
*/

// Dos barras comienzan un comentario de una línea.
```

Tipos

PHP soporta los siguientes tipos:

El tipo de una variable normalmente no lo indica el programador; en su lugar, lo decide PHP en tiempo de ejecución dependiendo del contexto en el que se utilice esa variable.

Array

PHP considera que todos los arrays son asociativos, por lo que no hay ninguna función PHP disponible para comprobar si un array es realmente asociativo o escalar.

```
// $a[] = valor
$arrayEscalar = array('apple', 'orange', 'tomato', 'carrot');

// $a['indice'] = valor

$arrayAsociativo = array(
    'fruit1' => 'apple',
    'fruit2' => 'orange',
    'veg1' => 'tomato',
    'veg2' => 'carrot'
);

// Funciona con todas las versiones de php
$asociativo = array('Uno' => 1, 'Dos' => 2, 'Tres' => 3);

// PHP 5.4 introdujo una nueva sintaxis
$asociativo = ['Uno' => 1, 'Dos' => 2, 'Tres' => 3];

echo $asociativo['Uno']; // imprime 1
```

Arrays unidimensionales

Se puede crear una array usando las funciones list() o array(), o se puede asignar el valor de cada elemento del array de manera explícita.

```
$a[0] = "apple";
$a[1] = "orange";
$b["trece"] = 13;
```

También se puede crear un array simplemente añadiendo valores al array.

Cuando se asigna un valor a una variable array usando corchetes vacíos [ ], el valor se añadirá al final del array.

```
$a[] = "hola"; // $a[2] == "hola"
$a[] = "mundo"; // $a[3] == "mundo"
// o
array_push($array, 'Cinco');
```

Los arrays se pueden ordenar usando las funciones asort(), arsort(), ksort(), rsort(), sort(), uasort(),

usort(), y uksort() dependiendo del tipo de ordenación que se desee.

Arrays Multidimensionales

Los arrays multidimensionales son bastante simples actualmente. Para cada dimensión del array, se puede añadir otro valor [clave] al final:

```
$a[1] = $f; # ejemplos de una sola dimensión
$a["foo"] = $f;
$a[1][0] = $f; # bidimensional
$a["foo"][2] = $f; # (se pueden mezclar índices numéricos y
asociativos)
$a[3]["bar"] = $f; # (se pueden mezclar índices numéricos y
asociativos)
```

Es posible referirse a arrays multidimensionales directamente dentro de cadenas encerrando la referencia al array (dentro de la cadena) entre llaves:

```
$a[3]['bar'] = 'Bob';
echo "Esto va a funcionar: {$a[3][bar]}";
```

La función array() se puede anidar para arrays multidimensionales:

```
<?
$a = array(
    "manzana" => array(
        "color" => "rojo",
        "sabor" => "dulce",
        "forma" => "redondeada"
    ),
    "naranja" => array(
       "color" => "naranja",
```

```
       "sabor" => "ácido",
       "forma" => "redondeada"
    ),
    "plátano" => array(
       "color" => "amarillo",
       "sabor" => "paste-y",
         "forma" => "aplatanada"
      )
);
echo $a["manzana"]["sabor"]; # devolverá "dulce"
?>
```

Para eliminar un elemento de un array se utiliza

```
unset($array[3]);
```

Números en punto flotante

Los números en punto flotante ("double") se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
$a = 1.234; $a = 1.2e3;
```

Entero

Los enteros se pueden especificar usando una de las siguientes sintaxis:

```
$a = 1234; # número decimal
$a = -123; # un número negativo
$a = 0123; # un 0 al comienzo declara un número octal (equivalente
al 83 decimal)
$a = 0x12; # un 0x al comienzo declara un hexadecimal (equivalente
al 18 decimal)
```

Objeto

Para inicializar un objeto, se usa la sentencia new para instanciar el objeto a una variable.

```
class foo {
  function do_foo () {
     echo "Doing foo.";
  }
}
$bar = new foo;
$bar->do_foo();
```

Cadena

Las cadenas de caracteres se pueden especificar usando uno de dos tipos de delimitadores.

Si la cadena está encerrada entre dobles comillas ("), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación).

El carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

Caracteres protegidos

secuencia	significado
\n	Nueva línea
\r	Retorno de carro
\t	Tabulación horizontal
\\	Barra invertida
\$	Signo del dólar
\”	Comillas dobles

Se puede acceder a los caracteres dentro de una cadena tratándola como un array de caracteres indexado numéricamente

Ejemplos de cadenas

```
<?php
/* Asignando una cadena. */
$str = "Esto es una cadena";

/* Añadiendo a la cadena. */
$str = $str . " con algo más de texto";

/* Esta cadena terminará siendo '<p>Número: 9</p>' */
$num = 9;
$str = "<p>Número: $num</p>";

/* Esta será '<p>Número: $num</p>' */
$num = 9;
$str = '<p>Número: $num</p>';

/* Obtener el primer carácter de una cadena */
$str = 'Esto es una prueba.';
$first = $str[0];
```

```
/* Obtener el último carácter de una cadena. */
$str = 'Esto es aún una prueba.';
$last = $str[strlen($str)-1];

?>
```

Variables

Las variables comienzan con el símbolo $.

Una variable válida comienza con una letra o guion bajo, seguida de cualquier cantidad de letras, números o guiones bajos.

Las variables booleanas no distinguen entre mayúsculas o minúsculas

```
$boolean = true;  // o TRUE o True
$boolean = false; // o FALSE o False
```

Asignar variables por referencia significa que la nueva variable simplemente referencia (en otras palabras, "se convierte en un alias de" o "apunta a") la variable original. Los cambios a la nueva variable afectan a la original, y viceversa.

Para asignar por referencia, simplemente se antepone un ampersand (&) al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo, el siguiente trozo de código produce la salida 'Mi nombre es Bob' dos veces:

```
<?php
$foo = 'Bob'; // Asigna el valor 'Bob' a $foo
$bar = &$foo; // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $foo; // $foo también se modifica.
echo $bar;
?>
```

A veces es conveniente tener nombres de variables variables.

```
$a = "hola"

$$a = "mundo";    // variable hola

echo "$a ${$a}";

//produce el mismo resultado que:
```

```
echo "$a $hello";
```

Constantes

Una constante se define utilizando define() y nunca puede ser cambiada en tiempo de ejecución, un nombre válido para una constante debe comenzar con una letra o guion bajo, seguido por cualquier número de letras, números o guiones bajos.

```
define("FOO",     "algo");
```

El acceso a una constante se puede realizar llamando a la variable elegida sin un símbolo de $

```
echo FOO; // Devuelve 'algo'
echo 'Esto imprime '.FOO;  // Devuelve 'Esto imprime algo'
```

Operadores

Aritméticos

ejemplo	nombre	resultado
$a + $b	Adición	Suma de $a y $b.
$a - $b	Substracción	Diferencia entre $a y $b.
$a * $b	Multiplicación	Producto de $a and $b.
$a / $b	División	Cociente de $a entre $b.
$a % $b	Módulo	Resto de $a dividido entre $b.

Comparación

ejemplo	nombre	resultado
$a == $b	Igualdad	Cierto si $a es igual a $b.
$a === $b	Identidad	Cierto si $a es igual a $b y si son del mismo tipo (sólo PHP4)
$a != $b	Desigualdad	Cierto si $a no es igual a $b.
$a < $b	Menor que	Cierto si $a es estrictamente menor que $b.
$a > $b	Mayor que	Cierto si $a es estrictamente mayor que $b.
$a <= $b	Menor o igual que	Cierto si $a es menor o igual que $b.
$a >= $b	Mayor o igual que	Cierto si $a es mayor o igual que $b.

Operador ternario

El operador ternario evalúa una condición y retorna un valor dependiendo si la condición es verdadera (true) o falsa (false).

```
if (condición) {
variable = valor-cuando-es-verdadera;
} else {
variable = valor-cuando-es-falsa;
}
```

Utilizando el operador ternario se simplifica de esta manera:

variable = (condición) ? valor-cuando-es-verdadera : valor-cuando-es-falsa;

Un ejemplo práctico:

```
<?php
if (date('G') < 12) {
$mensaje = 'Buenos días';
} else {
$mensaje = 'Buenas tardes';
}

echo $mensaje;
?>
```

Incremento/decremento

ejemplo	nombre	efecto
++$a	Preincremento	Incrementa $a en uno y después devuelve $a.
$a++	Postincremento	Devuelve $a y después incrementa $a en uno.
--$a	Predecremento	Decrementa $a en uno y después devuelve $a.
$a--	Postdecremento	Devuelve $a y después decrementa $a en uno.

Operadores de cadenas

El primero es el operador de concatenación (’.’)

```
$a = "Hola ";
$b = $a . "Mundo!"; // ahora $b contiene "Hola Mundo!"
```

El segundo es el operador de concatenación y asignación (’.=’).

```
$a = "Hola ";
$a .= "Mundo!"; // ahora $a contiene "Hola Mundo!"
```

Lógicos

ejemplo	nombre	resultado
$a and $b	Y	Cierto si tanto $a como $b son ciertos.
$a or $b	O	Cierto si $a o $b son ciertos.
$a xor $b	O exclusiva	Cierto si $a es cierto o $b es cierto, pero no ambos a la vez.
! $a	Negación	Cierto si $a no es cierto.
$a && $b	Y	Cierto si tanto $a como $b son ciertos.
$a || $b	O	Cierto si $a o $b son ciertos.

Estructuras de control

La construcción if permite la ejecución condicional de fragmentos de código, las sentencias if se pueden anidar indefinidamente dentro de otras sentencias if .

else extiende una sentencia if para ejecutar una sentencia en caso de que la expresión en la sentencia if se evalúe como FALSE

elseif, como su nombre sugiere, es una combinación de if y else

```
if ($a > $b) {
print "a es mayor que b";
} elseif ($a == $b) {
print "a es igual que b";
} else {
print "a es mayor que b";
}
```

Switch

La sentencia switch es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia switch

```
switch ($x) {
    case '0':
```

```
        print 'Switch does type coercion';
        break; // Debes incluir un break para no seguir con los
casos 'Dos' y 'Tres'
    case 'Dos':
    case 'Tres':
        // Hacer algo si la variables es 'Dos' o 'Tres'
        break;
    default:
        // Hacer algo por defecto
}
```

Los bucles while son los tipos de bucle más simples en PHP. Le dice a PHP que ejecute la(s) sentencia(s) anidada(s) repetidamente, mientras la expresión while se evalúe como TRUE. La forma básica de una sentencia while es:

while (expr) sentencia

```
$i = 1;
while ($i <= 10) {
print $i++; /* el valor impreso sería $i antes del incremento
(post-incremento) */
}
```

Los bucles do..while son muy similares a los bucles while, excepto que las condiciones se comprueban al final de cada iteración en vez de al principio. La principal diferencia frente a los bucles regulares while es que se garantiza la ejecución de la primera iteración de un bucle do..while (la condición se comprueba sólo al final de la iteración)

```
$i = 0;
do {
print $i;
} while ($i > 0);
```

Los bucles for son los bucles más complejos en PHP. Se comportan como su contrapartida en C. La sintaxis de un bucle for es:

for (expr1; expr2; expr3) sentencia

La primera expresión (expr1) se evalúa (ejecuta) incondicionalmente una vez al principio del bucle. Al comienzo de cada iteración, se evalúaexpr2. Si se evalúa como TRUE, el bucle continúa y las sentencias anidadas se ejecutan. Si se evalúa como FALSE, la ejecución del bucle finaliza.

Al final de cada iteración, se evalúa (ejecuta) expr3.

```
for ($i = 1; $i <= 10; $i++) {
print $i;
}
```

Los bucles foreach pueden iterar por arrays

```
$ruedas = ['bicicleta' => 2, 'coche' => 4];

foreach ($ruedas as $numero_ruedas) {
    echo $numero_ruedas;
} // Imprime "24"

echo "\n";
```

También se puede iterar sobre las claves, así como sobre los valores

```
foreach ($ruedas as $vehiculo => $numero_ruedas) {
    echo "Un $vehiculo tiene $numero_ruedas ruedas";
}

echo "\n";
```

break

break escapa de la estructuras de control iterante (bucle) actuales for, while o switch

```
$i = 0;
while ($i < 5) {
    if ($i === 3) {
        break; // Sale fuera del bucle while
    }
    echo $i++;
} // Imprime "012"
```

continue

continue se usa dentro de la estructura del bucle para saltar el resto de la iteración actual del bucle y continuar la ejecución al comienzo de la siguiente iteración.

```
for ($i = 0; $i < 5; $i++) {
    if ($i === 3) {
        continue; // Se salta esta iteración del bucle
    }
```

```
    echo $i;
} // Imprime "0124"
```