CLASE 08

CONCEPTOS INICIALES

- Repaso:
  - Formularios: diferentes tipos de elementos o controles
  - Uso de variables globales $_POST y $_GET
  - Métodos de envío
- Escritura de archivos con PHP
  - Usamos los datos de un formulario para almacenarlos en un archivo

OBJETIVOS

- Repaso general de temas vistos en clase anterior
- Conocer y usar las funciones para manejo de archivos , de PHP
- Usar los scripts de ejemplo y ejercicios ubicados dentro del archivo Clase_08_ Ejemplos.zip
- Aplicar conocimientos de clases anteriores para integrar con estas estructuras.
- Desarrollar código para ser visualizado en el navegador.

DESARROLLO

Introducción

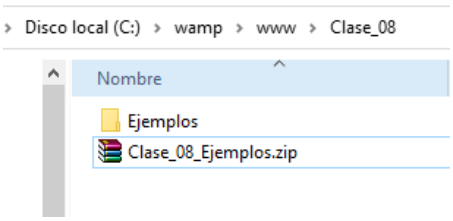
Recordemos los pasos previos para todas nuestras prácticas:

a) Debemos tener activo el WampServer:

Para comenzar a probar recuerden que el servidor local debe estar activado [en verde], tras doble click en el icono del Wamp del escritorio, o tras buscarlo en los Programas instalados:

b) Abriremos nuestro editor de texto, para trabajar con el código de los ejemplos que iremos viendo a continuación.

c) Tendremos que descargar el archivo **Clase_08_Ejemplos.zip** y ubicar su contenido en el directorio de alojamiento, creando las carpetas correspondientes para que quede de esta manera:



d) Abriremos el navegador [Chrome recomendado] y nos ubicaremos en nuestro server y la carpeta recién creada. De allí podremos ir revisando los ejemplos.

http://localhost/Clase_08/Ejemplos

Recordemos nuevamente:

La **ruta física a los archivos** o scripts de nuestra carpeta de alojamiento de nuestro servidor (local), será **C:\wamp\www**

La **ruta para ejecutar** esos archivos o scripts, alojadas en nuestro servidor (local), será **desde el navegador**, indicando primero el servidor a donde apuntamos, y luego las carpetas hasta llegar al script: http://localhost/Clase_xx/carpeta....

- Noten que envolvemos nuestro código con las etiquetas <?php y ?>
- La sentencia **echo** es la que muestra en pantalla el mensaje que se escribe entre comillas [dobles o simples].
- Recordar siempre terminar las sentencias con punto y coma ;
- Cada cambio que se genere en el script desde el editor de texto, guardarlo. Para ver el cambio en el navegador, refrescar la página con F5.[1]
- Cada script se guarda con la extensión .php para que pueda ser interpretado correctamente.

Ahora si, seguimos adelante!

[1] Deshabiliten el caché para que el navegador fuerce a traer los cambios recientes en cada cambio que hagan: https://amerikanistik.org/es/software/930-how-to-completely-disable-cache-in-google-chrome.html

HTML

Seguimos manipulando los valores enviados a través de un formulario de estructura HTML, y recogiendo esos datos mediante las variables **super globales de PHP: $_GET o $_POST según sea el método que tengamos definido en el formulario.**

Veremos algunos ejemplos más, usando estos valores, y también cómo generar un archivo "tipo log" en el servidor, tras el envío de estos datos.

Ya estamos próximos a poder guardar todo lo necesario en una base de datos.

Escritura de archivos con PHP

Vamos a crear un **archivo de texto plano (log)**, en alguna carpeta del servidor, y sus datos serán líneas con los **datos** que le enviemos desde el **formulario**, mas algunos otros que seguramente serán de utilidad.

Este proceso de captar ciertos datos en archivos, tambien nos sive para ir "capturando" cierta actividad que se va generando en nuestro sistema o aplicación. Lo que necesitemos, podemos ir generando un registro en estos archivos. Generalmente se les llama archivos de **logs**.

Todos los servers cuentan con ciertos *logs*, de accesos, de errores, así mismo nosotros podemos generar algunos cuando nos hagan falta.

Para el **primer ejemplo**, vamos a revisar un **formulario de Login**, repasando los elementos/controles y sus propiedades para ser utilizados con PHP. El proceso de envío de datos será trabajado por Post, se validarán los datos enviados, y **al enviar todos sus datos requeridos se escribirá un archivo con esos datos**.

Ubiquen entonces el script del primer ejemplo: C:\wamp\www\Clase_08\Ejemplos\Ejemplo1.php

Y desde el navegador: http://localhost/Clase_08/Ejemplos/Ejemplo1.php

Repaso de los controles del formulario

- El **formulario** procesará mediante el método ***post***.
- Existen 3 elementos/controles del formulario:
  - Un **cuadro de tipo text** para el ingreso de login, llamado *Login* Su valor será capturado por PHP en la variable $_POST[‘Login’].
  - Un **cuadro de tipo password** para el ingreso de la clave, llamado *Clave*. Su valor será capturado por PHP en la variable $_POST[‘Clave].
  - Un **botón de tipo *submit*** para el envío de los datos del formulario, llamado *Ingresar*. Su valor será capturado por PHP en la variable $_POST[‘Ingresar].

- En este caso, el valor del Login es recordado cada vez que se pulsa el botón, mediante la codificación en el *value*. Noten la forma de usar el **Operador Ternario [2]de PHP**. Esto es muy usado, pues es una forma abreviada de usar el IF, pueden usarlo cuando lo deseen.

[2] https://www.neoguias.com/if-abreviado-en-php-el-operador-ternario/

Es una **manera "corta"** que nos permite utilizar PHP para cuando tenemos un **if** simple, con **una acción si se cumple la condición**, y **una acción si no se cumple**. Revisen el link en la referencia.

```
<div class="form-group">
    <label>Ingresa tu login</label>
    <input class="form-control" type="text" name="Login"
        value="<?php echo !empty($_POST['Login']) ? $_POST['Login'] : ''; ?>" />
</div>
```

Aquí pueden ver el "IF largo", y tiene el mismo efecto que si lo hacemos con el Operador Ternario.

```
<div class="form-group">
    <label>Ingresa tu login</label>
    <input class="form-control" type="text" name="Login"
        value="<?php if (!empty($_POST['Login'])) {
                    echo $_POST['Login'] ; //tiene algun valor
                } else {
                    echo  '' ;  //está vacío
                } ?>"
    />
</div>
```

Procesamiento del formulario de login para escribir el log.

❖ Al presionar el botón de Ingresar, las líneas 4 a 12 muestran las **validaciones** de cada campo de texto.

- Si la longitud del Login es menor a 3 caracteres, mensaje de error.

- Si la Clave llega vacía, mensaje de error.

```
4   if (!empty($_POST['Ingresar'])) {
5       //esto va a ejecutar cuando el boton llegue con valor, es decir se haya pulsado
6       //validacion de la caja de texto del Login
7       if (strlen($_POST['Login']) < 3) {
8           //su longitud es menor a 3 caracteres
9           $Mensaje = "Debe ingresar un Login de 3 caracteres o mas. <br />";
10      } else if (empty($_POST['Clave'])) {
11          //validacion de la caja del Password
12          $Mensaje = "Debe ingresar su Clave. <br />";
13      } else {
14          //creo la carpeta de logs, donde voy a crear mi archivo
15
16          if (!is_dir('logs')) {      //pregunto si no existe el directorio
17              mkdir('logs');          //si no existe, lo creo
18              chmod('logs', 0777);    //le doy permisos de lectura/escritura
19          }
20          //variables a utilizar
21          $FechaHoy = date('Ymd'); //para nombrar el archivo
22          $FechaHoraHoy = date('Y-m-d H:i:s'); //para contenido del archivo
23
24          //mensaje a escribir en el archivo:
25          $TextoLog = "$FechaHoraHoy : access {$_POST['Login']}  \n";
26
27          //genero el archivo
28          $ArchivoLog = fopen("logs/access_$FechaHoy.log", 'a+');
29          //lo escribo con el texto
30          fwrite($ArchivoLog, $TextoLog);
31          //cierro el archivo
32          fclose($ArchivoLog);
33          $Mensaje = "<strong>Se ha guardado el log de acceso.</strong>
34          Puedes verlo <a href='logs/access_$FechaHoy.log' target='_blank'>aqui</a> ";
35      }
36  }
```

❖ Si pasa todas estas validaciones, procedo a escribir el log. [Líneas 14 a 33]

- Si no existe ya una **carpeta de logs**, la crea dándole todos los permisos para poder escribir dentro, mi archivo de log. (Linea 16 a 19)
- Genero **variables** con la fecha actual, y otra con fecha y hora actuales. (Líneas 21 y 22)
- Genero la **variable** con el texto a escribir en el **log**: uso la fecha actual, concateno con la palabra “access” y uso el valor del login que está queriendo ingresar a mi aplicación. (Linea 25)
- Luego genero la **apertura, escritura, y cierre de un archivo** ubicado en la carpeta *logs,* y llamado “access_fechaActual.log” (Linea 28 a 32)
- Genero una **variable de mensaje** de Ok, brindando posibilidad de ver el log recién creado desde un link que apunta a su ubicación. (Linea 33)

❖ Cuando la variable del mensaje tenga un valor para ser mostrado, se mostrará, rodeada de la caja con estilos correspondiente.

```
<form role="form" method="post" >
    <div class="row">
        <div class="col-lg-4" style="text-align: left;">
            <img src="dist/img/login2.png" class="img-thumbnail" />
        </div>
        <div class="col-lg-6">
            <!-- seccion central  -->
            <?php if (!empty($Mensaje)) { ?>
            <div class="alert alert-warning">
                <?php echo $Mensaje; ?>
            </div>
            <?php } ?>
            <div class="form-group">
                <label>Ingresa tu login</label>
```

Para probar el funcionamiento general

Deberán probar entonces el Ejemplo1, presionando el **botón** sin valores en las cajas de texto, para ver los mensajes de error.

Cuando ingresen ambos valores correctamente, verán el mensaje de ok con el link al log de accesos recientemente generado.

La carpeta física de logs se habrá creado en: C:\wamp\www\Clase_08\Ejemplos\logs

Y el archivo de logs del día actual se irá actualizando cada vez que realicen un proceso de "acceso correcto".

Como verán, **es muy fácil generar un script de log con PHP**, con los valores que necesiten.

Algo muy útil, es exportar información de nuestras aplicaciones, en **formato CSV**[3]. Aquí también pueden utilizar esto de generar el archivo, con extensión "csv". Lo veremos cuando usemos base de datos, asi podemos exportar la info desde allí. De momento, ya conocen cómo crear el archivo.

Si no conocen respecto de **archivos CSV,** pueden consultar la referencia al pie de página.

A continuación, sigue una tarea de Análisis de otro script con un comportamiento similar. Deberán comprender el procesamiento y las secciones del formulario involucradas.

Tarea Análisis

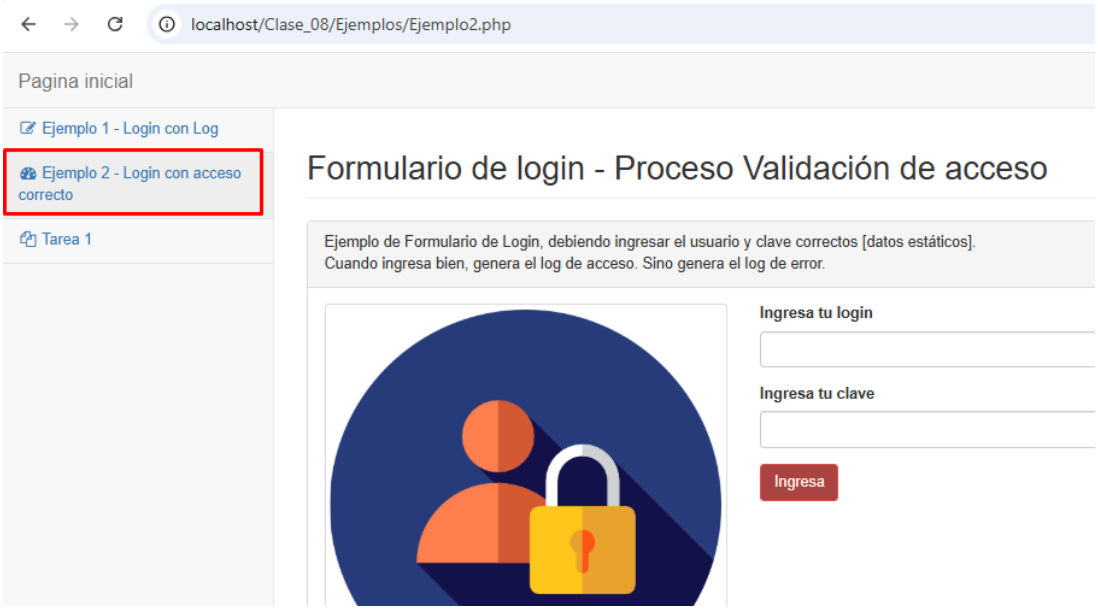
Script a trabajar: http://localhost/Clase_08/Ejemplos/Ejemplo2.php

Analizar:

- Método con el que trabaja el formulario
- Controles usados en el formulario

[3] https://www.geeknetic.es/Archivo-CSV/que-es-y-para-que-sirve
https://iosmac.es/el-formato-csv-y-lo-que-has-de-saber.html

- Nombres de cada control
- Validaciones realizadas a los controles y mensajes que se muestran cada vez.
- Aquí un acceso correcto se generará cuando en el login se ingrese la palabra alumno y en la clave se ingrese Alu_ISSD. [respetar mayúsculas y minúsculas].
- Noten que se escriben dos logs: uno de acceso correcto y otro de error.
- Visualicen cada log, son distintos, llevan distinto nombre, pero se guardan en la misma carpeta de logs.
- Visualizar la sección donde se muestran los mensajes al navegante. [Son distintas cajas para cada mensaje, de error o de Ok].



Tarea #1

Desde el siguiente script: C:\wamp\www\Clase_08\Ejemplos\Tarea1.php

En el navegador: http://localhost/Clase_08/Ejemplos/Tarea1.php

Desde la template asignada, deberán completar el HTML y codificar lo necesario en PHP para lograr:

- Procesar los datos mediante post.
- Generar el formulario con sus controles: tipos y propiedades correctos.
- Generar la validación de los controles:
  - Nombre y Apellido deben contener valores de 3 caracteres como mínimo.
  - El Email debe contener valores de 6 caracteres al menos.
- Una vez que pasan todas las validaciones, se debe escribir un log, llamado "registro_accesos.csv", con esta información: [Notar que se van agregando los valores por cada registración correcta].

  Fecha y hora actual | Valor del apellido | Valor del nombre | Email

  Fecha y hora actual | Valor del apellido | Valor del nombre | Email

  Fecha y hora actual | Valor del apellido | Valor del nombre | Email

  Noten que el separador de los valores es el *pipeline*, caracter especial ubicado generalmente al lado del 1 en el teclado numérico superior, o mediante Alt+124. Es muy utilizado.
- Recuerden codificar solo en PHP, la template no es necesario que le modifiquen su diseño.

Continuamos!!!