Lexpin

¿Qué es PHP?

PHP es un lenguaje de programación de código abierto que se utiliza principalmente para crear aplicaciones web dinámicas. PHP significa "Hypertext Preprocessor" y se ejecuta en el servidor web, lo que significa que el código PHP se procesa en el servidor antes de enviar el resultado al navegador del usuario.

¿Cómo utilizarlo?

Para utilizarlo, se necesita un servidor web que sea compatible con PHP, como Apache. Además, también se necesita un editor de código para escribir y editar el código. Una vez que se tienen estos elementos, se puede empezar a escribir código PHP.

Sintaxis

Su sintaxis se basa en bloques de código rodeados por las etiquetas "**<?php**" y "**?>**" que indican al servidor web que el código contenido en ellas debe ser procesado por el motor de PHP. La sintaxis básica es similar a la de otros lenguajes de programación por lo que es un lenguaje de fácil aprendizaje.

- **Variables**: Las variables en PHP comienzan con el símbolo "$" seguido del nombre de la variable.

```php
<?php
    $variable1 = 10;
    $variable2 = 5;
?>
```

Tipos de datos:
PHP admite varios tipos de datos que se utilizan para almacenar diferentes tipos de información. Los tipos de datos más comunes en PHP son los siguientes:

- **Integer** (entero): representa un número entero sin punto decimal, positivo o negativo.
  Ejemplo: $edad = 35;
- **Float** (flotante): representa un número decimal, positivo o negativo.
  Ejemplo: $precio = 12.50;
- **String** (cadena de texto): representa una secuencia de caracteres.

Ejemplo: $nombre = "Gabriel";

- **Boolean** (booleano): representa un valor verdadero o falso.
  Ejemplo: $activo = true;
- **Array** (matriz): representa una colección de elementos.
  Ejemplo: $numeros = array(1, 2, 3, 4, 5);
- **Object** (objeto): representa una instancia de una clase.
  Ejemplo:

```
class Persona {
  public $nombre;
  public $edad;
}
$persona = new Persona();
$persona->nombre = "Gabriel";
$persona->edad = 30;
```

Null: representa una variable sin valor.

Ejemplo: $variable = null;

- **Operadores**: PHP incluye operadores para realizar operaciones matemáticas, comparaciones y concatenación de cadenas.
  1. Suma (+)
  2. Resta (-)
  3. Multiplicación (*)
  4. División (/)
  5. Módulo (%)
  6. Igualdad (==)
  7. Desigualdad (!=)
  8. Concatenación (.)

- **Estructuras** de control: PHP incluye estructuras de control como:

1. If)

```php
<?php
    $variable1 = 10;
    $variable2 = 5;

    if($variable1 > $variable2){
        echo("La variable 1 es mayor que la variable 2");
    }
    else{
        echo("La variable 1 es menor que la variable 2");
    }
?>
```

2. While)

```php
<?php
    $i = 0;
    while ($i < 10) {
        echo $i." ";
        $i++;
    }
?>
```

3. For)

```php
<?php
    for ($i = 1; $i <= 10; $i++) {
        echo "Numero ".$i."<br/>";
    }
?>
```

4. Foreach)

```php
<?php
    $frutas = ["manzana","Pera","Uva","Mango"];
    foreach($frutas as $fruta){
        echo $fruta."<br>";
    }
?>
```

5. Switch)

```php
<?php
    $color = "verde";
    switch ($color) {
        case "rojo":
            echo "El color es rojo";
            break;
        case "verde":
            echo "El color es verde";
            break;
        default:
            echo "El color no es ni rojo ni verde";
    } <- #36-45 switch ($color)
?>
```

- **Funciones**: PHP incluye muchas funciones integradas que se pueden utilizar para realizar tareas comunes, aunque también es posible definir funciones personalizadas para realizar tareas específicas.

Funciones para trabajar con strings:

- **strlen():** Esta función devuelve la longitud de una cadena de texto.
- **substr():** Esta función devuelve una parte de una cadena de texto, según los índices especificados.
- **strpos():** Esta función devuelve la posición de la primera ocurrencia de una subcadena de texto dentro de una cadena de texto.
- **str_replace():** Esta función reemplaza todas las ocurrencias de una subcadena de texto dentro de una cadena de texto con otra subcadena de texto.
- **strtolower():** Esta función convierte una cadena de texto en minúsculas.
- **strtoupper()**: Esta función convierte una cadena de texto en mayúsculas.

Funciones para trabajar con arrays:

- **array_push():** Esta función inserta uno o más elementos al final de un array.
- **array_pop()**: Esta función elimina el último elemento de un array.
- **array_shift():** Esta función elimina el primer elemento de un array.
- **array_unshift():** Esta función inserta uno o más elementos al inicio de un array.
- **array_merge():** Esta función combina dos o más arrays en uno solo.
- **array_unique():** Esta función elimina los elementos duplicados de un array.

Funciones de base de datos:

- **mysqli_connect** para establecer una conexión con una base de datos MySQL.
- **mysqli_query** para ejecutar una consulta SQL en una base de datos MySQL.
- **mysqli_fetch_assoc** para obtener los resultados de una consulta como un array asociativo.

Funciones Personalizadas:

```php
<?php
    function tablaMultiplicar() {
        $num = 5;
            echo "<h2>Tabla de multiplicar del $num:</h2>";
            for ($i = 1; $i <= 10; $i++) {
                $producto = $num * $i;
                echo "<p>$num x $i = $producto</p>";
            }
    } <- #71-78 function tablaMultiplicar()

    tablaMultiplicar();
?>
```

Regex

Puedes utilizar expresiones regulares (regex) a través de funciones nativas. Estas funciones te permiten buscar, reemplazar y dividir cadenas de texto utilizando patrones de búsqueda que se ajusten a tus necesidades.

- **preg_match($patron, $cadena, &$coincidencias)**: esta función busca una cadena de texto $cadena para ver si coincide con el patrón de expresión regular $patron. Si se encuentra una coincidencia, devuelve 1(true), de lo contrario devuelve 0(false). Puedes utilizar el tercer parámetro &$coincidencias para obtener un array que contiene la coincidencia encontrada en la cadena de texto.

- **preg_match_all($patron, $cadena, &$coincidencias)**: esta función busca todas las coincidencias en la cadena de texto $cadena utilizando el patrón de expresión regular definido en $patron. También puedes utilizar el tercer parámetro &$coincidencias para obtener un array que contiene el total de coincidencias encontradas en la cadena de texto.

- **preg_replace($patron, $reemplazo, $cadena)**: esta función reemplaza todas las coincidencias encontradas del patrón de expresión regular almacenado en $patron, en la cadena de texto $cadena con el valor de reemplazo $reemplazo y devuelve una nueva cadena de texto con las sustituciones realizadas.

- **preg_split($patron, $cadena):** esta función divide la cadena de texto $cadena utilizando el patrón de expresión regular almacenado en $patron y devuelve un array que contiene las partes de la cadena de texto divididas.

CRUD

Es un acrónimo que significa **Create, Read, Update y Delete,** que se refiere a las cuatro operaciones básicas de las bases de datos y la gestión de datos en general. Estas operaciones se realizan comúnmente en aplicaciones web y programas de software para interactuar con los datos almacenados en una base de datos.

- **Create (Crear)**: se refiere a la creación de nueva informacion en la base de datos. Por ejemplo, en una aplicación de registro de usuarios, la operación de crear se utilizaría para agregar un nuevo usuario a la base de datos.
- **Read (Leer):** se refiere a la obtención de información de la base de datos. Por ejemplo, en una aplicación de registro de usuarios, la operación de leer se utilizaría para recuperar la información de un usuario existente.
- **Update (Actualizar):** se refiere a la actualización de registros existentes en la base de datos. Por ejemplo, en una aplicación de registro de usuarios, la operación de actualizar se utilizaría para cambiar la información de un usuario existente.
- **Delete (Borrar):** se refiere a la eliminación de información en la base de datos. Por ejemplo, en una aplicación de registro de usuarios, la operación de borrar se utilizaría para eliminar la información de un usuario existente.

Trabajar con una BD MySQL

Conectar

```php
<?php
    /*Variables de conexion */
    $servername = "localhost";
    $username = "root";
    $password = "";

    //Abrimos conexion con localhost
    $connection = new mysqli($servername,$username,$password);

    //Se valida coneccion, en caso de error se detiene lo demas
    if($connection->connect_error) {
        die("Connection failed: "). $connection->connect_error;
    }
    else{
    echo ("<script>alert(Conectado a LocalHost);</script>");   //Se conecto correctamente
    }

    //Crear base de datos
    //Se crea la variable que contiene el codigo SQL que ejecutaremos
    $sql = "CREATE DATABASE mi_db";

    //Ejecutamos el SQL y validamos si se ejecuta correctamente
    if($connection->query($sql) === true) {
        echo ("<script>aletr(La base de datos se creo!);</script>");
    } else {
        echo ("<script>aletr(No se creo la base de datos!);</script>".$connection->error);
    }

    date_default_timezone_set("America/Santiago"); //Definicion de la zona horaria
    echo ("<br><p>".date('d-M-Y')." ".date('h:i:s')."</p>");

    //Cerramos conexion por seguridad
    $connection->close();
?>
```

Insertar Información

```php
<?php
    /*Variables de conexion */
    $servername = "localhost";
    $username = "root";
    $password = "";
    $db = "mi_bd";

    //crear conexion
    $connection = new mysqli($servername,$username,$password,$db);

    //Validamos conexion, si hay error no se ejecutara el resto del codigo
    if($connection->connect_error) {
        die("Connection failed: "). $connection->connect_error;
    }
    else{
        //Si imprime esto en consola, entonces se conecto correctamente
        echo ("<script>console.log('Conectado')</script>");
    }

    //Cargar datos en tabla
    //Se crea la variable que contiene el codigo SQL que ejecutaremos
    $sql = "INSERT INTO users(Nombre, Email, Contraseña) VALUES ('$nombre','$correo','$claveEncriptada')";

    //query() es una funcion que realiza una consulta de SQL en este caso en la conexion creada y guardada en $connection
    try{$connection->query($sql) === true;
        echo ("Felicidades ".$nombre." te has registrado exitosamente.");
    }

    //catch exception
    catch(Exception $e) {
        echo ("Ocurrio un error al registrar los datos ".$e->getMessage());
    }

    //Cerramos conexion por seguridad
    $connection->close();
?>
```

Leer Información

```php
<?php
    /*Variables de conexion */
    $servername = "localhost";
    $username = "root";
    $password = "";
    $db = "mi_bd";

    //Crear conexion
    $conn = new mysqli($servername,$username,$password,$db);

        //Validamos conexion, si hay error no se ejecutara el resto del codigo
        if($conn->connect_error) {
            die("Connection failed: "). $conn->connect_error;
        }
        else{
            //Si imprime esto, se conecto correctamente
            echo ("<script>console.log('Conectado');</script>");
        }

        //Preparar la orden SQL
        $sql= "SELECT * FROM users";

        //Ejecutar la consulta y almacenar los datos en $result
        $result = $conn->query($sql);

        //Si el numero de resultados es mayor a 0 se muestran con un ciclo while
        if ($result->num_rows > 0) {
            //Se muestran datos de cada row (Fila)
            //fetch_assoc() es un método en PHP que se utiliza para obtener una fila de resultados
            //de una consulta SQL en forma de arreglo asociativo.
            while($row = $result->fetch_assoc()) {
                $Nombre = $row["Nombre"];
                $Email = $row["Email"];
                echo
                    ("Nombre: ".$Nombre. " - Email: ".$Email);
            }
        } else {
            echo "0 resultados";
        }
    //Cerramos conexion por seguridad
    $conn->close();
?>
```

Actualizar Información

```php
<?php
    /*Variables de conexion */
    $servername = "localhost";
    $username = "root";
    $password = "";
    $db = "mi_bd";
    //crear conexion
    $connection = new mysqli($servername,$username,$password,$db);
    //Validamos conexion, si hay error. No se ejecutara el resto del codigo
    if($connection->connect_error) {
        die("Connection failed: "). $connection->connect_error;
    }
    else{
        //Si imprime esto en consola, se conecto correctamente
        echo ("<script>console.log('Conectado)</script>");
    }
    $suario=strtolower($_POST['IUser']);
    $correo=strtolower($_POST['IEmail']);
    //Se crea la variable $sql que contiene la consulta SQL que ejecutaremos
    $sql = "UPDATE users SET Usuario = $usuario WHERE Email = $correo" ;
    //query() es una funcion que realiza una consulta de SQL en este caso en la conexion creada y guardada $connection
    try{$connection->query($sql) === true;
        echo ("<div class='flex justify-center'><p>Usuario actualizado.<br>Nuevo nombre de usuario: ".$usuario."</p></div>");
    }
    catch(Exception $e) {
        echo ("<div class='flex justify-center'><p>Error al actualizar los datos.<br>".$e->getMessage()."</p></div>");
    }
    //Cerramos conexion por seguridad
    $connection->close();
?>
```

Eliminar Información

```php
<?php
    /*Variables de conexion */
    $servername = "localhost";
    $username = "root";
    $password = "";
    $db = "mi_bd";
    //crear conexion
    $connection = new mysqli($servername,$username,$password,$db);
    //Validamos conexion, si hay error. No se ejecutara el resto del codigo
    if($connection->connect_error) {
        die("Connection failed: "). $connection->connect_error;
    }
    else{
        //Si imprime esto en consola, se conecto correctamente
        echo ("<script>console.log('Conectado)</script>");
    }
    $suario=strtolower($_POST['IUser']);
    $correo=strtolower($_POST['IEmail']);
    //Se crea la variable $sql que contiene la consulta SQL que ejecutaremos
    $sql = "DELETE FROM users WHERE Email = $correo";
    try{$connection->query($sql) === true;
        echo ("<p>Se elimino el usuario: ".$usuario."</p>");
    }
    catch(Exception $e) {
        echo ("<p>Error al eliminar el usuario.<br>".$e->getMessage()."</p>");
    }
    //Cerramos conexion por seguridad
    $connection->close();
?>
```

Modularización

La modularización es un proceso mediante el cual se divide un código en partes más pequeñas y manejables llamadas módulos, cada módulo contiene una parte específica del código que realiza una tarea en particular.

La modularización tiene varios beneficios, como:

- **Facilitar la lectura**: Se facilita la lectura haciendo que el código sea más fácil de leer y entender, ya que los módulos pueden tener nombres que indiquen lo que hacen.
- **Facilita la depuración**: Aumenta la facilidad de depuración y mantenimiento del código, ya que los errores pueden aislarse y solucionarse en módulos específicos.
- **Ayuda a reutilizar el código**: Los módulos pueden utilizarse en diferentes partes del programa o en diferentes programas.

En **PHP**, se pueden modularizar diferentes partes del código, por ejemplo, la conexión a la base de datos, el manejo de formularios, el envío de correos electrónicos, entre otros. La modularización se puede realizar mediante la creación de archivos independientes que contienen funciones específicas o mediante la definición de funciones dentro del mismo archivo, que luego se pueden llamar desde otros archivos.

La modularización es una buena práctica de programación que permite crear código más organizado, mantenible y reutilizable.

¿Como se hace?

Existen dos formas principales de modularizar el código: include y require. Ambas funciones permiten incluir código de otros archivos **PHP** en el archivo actual.

Ejemplo: Tenemos la conexión de datos en un archivo dedicado:

```php
<?php
    /*Variables de conexion */
    $servername = "localhost";
    $username = "root";
    $password = "";
    $db = "mi_bd";
    //crear conexion
    $connection = new mysqli($servername,$username,$password,$db);
    //Validamos conexion, si hay error. No se ejecutara el resto del codigo
    if($connection->connect_error) {
        die("Connection failed: "). $connection->connect_error;
    }
    else{
        //Si imprime esto en consola, se conecto correctamente
        echo ("<script>console.log('Conectado)</script>");
    }
?>
```

Archivo llamado conexion.php

- **La función include**: incluye el archivo especificado y muestra una advertencia si no se encuentra el archivo.

```php
<?php
    include "./conexion.php";

    $suario=strtolower($_POST['IUser']);
    $correo=strtolower($_POST['IEmail']);
    //Se crea la variable $sql que contiene la consulta SQL que ejecutaremos
    $sql = "DELETE FROM users WHERE Email = $correo";
    try{$connection->query($sql) === true;
        echo ("<p>Se elimino el usuario: ".$usuario."</p>");
    }
    catch(Exception $e) {
        echo ("<p>Error al eliminar el usuario.<br>".$e->getMessage()."</p>");
    }
    //Cerramos conexion por seguridad
    $connection->close();
?>
```

Archivo llamado deleteUser.php

- **La función require:** también incluye el archivo especificado, pero muestra un error fatal si no se encuentra el archivo deteniendo todo proceso relacionado.

```php
<?php
    require "./conexion.php";

    $suario=strtolower($_POST['IUser']);
    $correo=strtolower($_POST['IEmail']);
    //Se crea la variable $sql que contiene la consulta SQL que ejecutaremos
    $sql = "DELETE FROM users WHERE Email = $correo";
    try{$connection->query($sql) === true;
        echo ("<p>Se elimino el usuario: ".$usuario."</p>");
    }
    catch(Exception $e) {
        echo ("<p>Error al eliminar el usuario.<br>".$e->getMessage()."</p>");
    }
    //Cerramos conexion por seguridad
    $connection->close();
?>
```

Archivo llamado deleteUser.php

En ambos casos, el archivo conexion.php contendría el código que se desea modularizar. Una vez que se incluye(include) o se requiere(require) el archivo, el código que contiene se ejecutará como si fuera parte del archivo actual.

La elección entre include y require depende del caso de uso. Si se trata de una función opcional que no es crítica para el funcionamiento del programa, include puede ser la mejor opción. Si, por otro lado, la función es crítica para el programa, require es más apropiado, ya que evita que se ejecuten errores fatales.