Pérez Molinito Cristian José
Programación Web

Conceptos básicos de PHP

- **Sintaxis PHP**: Dentro de un archivo PHP, un script comienza con **<?php** y termina con **?>**.
- **Variables en PHP**:
  Las variables en el lenguaje PHP comienzan con el carácter **$**, seguidas de un identificador.
  $txt = “Hola Mundo”;
  $Num = 42;
  - Una variable siempre comienza con el carácter **$**, seguido de un identificador.
  - Un identificador siempre debe comenzar con una letra o el caracter **_**.
  - Un identificador no puede comenzar con un número.
  - Un identificador sólo puede contener valores alfanuméricos.
  - Los identificadores son **case-sensitive**, es decir, se toman en cuenta mayúsculas y minúsculas ($Doge != $doge).
  - Una variable declarada afuera de una función se considera una variable global (**GLOBAL SCOPE**) que sólo podrá ser utilizada fuera de las funciones.
  - Existe una forma de acceder a los valores de un global scope dentro de alguna función usando una **Global Keyword** antes de las variables:
    ```
    $x = 10;
    function hola(){
     global $x;
    }
    ```
  - PHP también almacena todas las variables globales en un arreglo llamado $GLOBALS[]. El índice de dicho arreglo mantiene el nombre de la variable.
  - PHP utiliza distintos tipos de datos que pueden ser almacenados en las variables:
    - String
    - Integer
    - Float
    - Boolean
    - Array
    - Object
    - NULL
    - Resource
  - En el caso del tipo de dato **Object**, se trata de un tipo de datos que puede almacenar datos y las información necesaria para poder procesar esos datos. En PHP **un objeto debe ser declarado explícitamente**.

- **Funciones en PHP**:
  Una función puede ser definida empleando una sintaxis como la siguiente:
  ```
  function Hola(Parametros){
    //Todo el código necesario aquí
  }
  ```
  Todas las funciones y clases de PHP tienen un ámbito global, es decir, se pueden

llamar desde fuera de una función, incluso si fueron definidas dentro y viceversa.
En PHP no se permite la sobrecarga de funciones, no es posible modificar o redefinir funciones previamente declaradas.
Los identificadores de las funciones no toman en cuenta mayúsculas o minúsculas, sin embargo, es una buena práctica el llamar a las funciones tal y como fueron declaradas originalmente.

- **Estructuras de control en PHP**:
  Como es debido, PHP cuenta con las estructuras básicas de control de flujo de ejecución:
  - **if**
    Estructura de control fundamental en muchos lenguajes de programación.
    ```
    if(expr)
     #CodeToDo
    ```
  - **else**
    Permite ejecutar una secuencia de código alternativa si la expresión de la condición **if** no se cumplió.
    ```
    if(expr_1)
     #ToDoHere
    else
     #SomethingElseToDo
    ```
  - **while**
    Se trata de una estructura cíclica, es muy simple y fácil de usar, el ciclo de ejecución continuará mientras se cumpla la condición definida para dicha estructura, y dejará de ejecutarse hasta que esa condición cambie.
    ```
    while(expr)
     #ToDoHere
    ```
  - **do-while**
    Es muy similar a la estructura **while**, la diferencia se encuentra en que para **do-while** primero se ejecuta el código dentro del ciclo y posterior a ello se evalua la condición definida, de esta forma se determinará si se repite nuevamente el código o continua con las demás líneas del programa. En el ciclo **while**, primero se evalúa la condición antes de ejecutar el código que contiene, y en ocasiones puede no ejecutarse nunca si la condición así lo dicta.
  - **For**
    Las estructuras **for** son más complejas, se componen de 3 condiciones que se deben evaluar o ejecutar, el ciclo termina cuando se cumple la condición del a segunda expresión.
    ```
    for(exp1;exp2;exp3)
     #ToDoHere
    ```
  - **foreach**
    La estructura **foreach** provee de una sencilla forma de ejecutar iteraciones sobre un arreglo de datos, por lo tanto, la estructura **foreach** sólo funcionará en arreglos y objetos, si se intenta ejecutar esta estructura sobre un tipo de dato que no corresponda a un arreglo o un objeto, la ejecución del código generará un error.
    ```
    foreach(array_expression as $value)
    ```

#ToDoHere

- **break**
  El uso de **break** detendrá la ejecución de alguna estructura cíclica (**while**, **do-while**, **for**, **foreach** o **switch**). **Break** también acepta argumentos numéricos para indicar cuantas estructuras anidadas deberá detener, el valor por defecto de este argumento es **1**, por lo que si no tiene un argumento definido, detendrá la ejecución de la estructura inmediata donde se encuentra.
- **Continue**
  Se usa con estructuras cíclicas para “saltar” el resto del código dentro de la iteración actual del ciclo y continuar la ejecución desde la evaluación de la condición del mismo ciclo y comenzar de nuevo la ejecución del código. Así como **break**, también **continue** puede aceptar argumentos numéricos que definirán cuantas estructuras anidadas deberá “saltar”, y el valor de este argumento por defecto también es **1**.
- **switch**
  La estructura **switch** es similar a un conjunto de estructuras **if** anidadas en una sola expresión. En ocasiones será necesario comparar varias veces la misma variable o expresión con diferentes valores y ejecutar distintos bloques de código dependiendo de los resultados de las comparaciones realizadas.

```
switch(exp){
 case 0:
  #ToDoHere;
 break;
 case 1:
  #DoSomethingDifferentHere;
 break;
 case …
}
```

- **declare**
  La estructura **declare** se usa para definir un conjunto de directivas para un bloque de código. La sintaxis de esta estructura es similar a casi cualquier otra sentencia de control de flujo.

```
declare(directive)
 #Statement
```

  La sección de las directivas permite al comportamiento del bloque de **declare** ser establecido. Existen tres tipos de directivas: **thick**, **encoding** y **strict**.
- **Return**
  Esta expresión “regresa” un programa de control al módulo que lo mandó llamar. La ejecución del código continuará la invocación del módulo que llamó a alguna función.
- **Require**
  La declaración **require** es similar a **include** por la excepción de que si existe alguna falla en la compilación, enviará un error de compilación que impedirá completar el proceso, en otras palabras, detendrá el **script**, mientras que **include** sólo emitirá una advertencia pero permitirá la ejecución del **script**.

Pérez Molinito Cristian José
Programación Web

Patrón de diseño MVC

El patrón de diseño MVC permite organizar el código basándose ensu función, es decir, lo separa en capas que separan las funciones de cada código y faciliten aspectos como la modificación y mantenimiento posteriores al desarrollo del proyecto.

- **Capa de Modelo**:
  Define la lógica del negocio (Una base de datos pertenece a esta capa).
- **Capa de Vista**:
  Define los elementos que permiten la interacción del sistema con el usuario final (Un gestor de plantillas pertence a esta capa).
- **Capa de Controlador**:
  Se trata de un bloque de código que realiza llamadas al modelo para obtener datos, y los envía a la vista para mostrarlos como resultados al usuario.