Notaciones en la Programación PHP

A fines de lograr una mayor organización, legibilidad y productividad del código fuente final, se pone a disposición las siguientes convenciones de escritura, que facilitarán no sólo el orden del mismo, sino también una mejor puesta en escena del trabajo en equipo.

Teniendo en cuenta el significado de convención, se definirán a continuación un conjunto de reglas que permitirá no sólo determinar la secuencia de caracteres que enunciarán las variables o métodos del código fuente, sino que también indicarán la manera de trabajar las estructuras de control que componen al mismo.

1. Algunas aclaraciones

Notación CamelCase: *Estilo de escritura que aplica frases compuestas. Existen dos tipos de CamelCase:*

- ***UpperCamelCase, CamelCase o PascalCase:*** *en esta variante la primera letra es mayúscula.*
- ***lowerCamelCase, camelCase o dromedaryCase****: la primera letra es minúscula.*

En nuestro caso se adoptará el UpperCamelCase.

2. Declaración de Clases, Métodos, Propiedades y Variables

Declaración de una Clase:

- Las mismas deben ser definidas bajo la notación *CamelCase o UpperCamelCase.*
- Recordar que las clases representan *Cosas/Objetos* por lo que se deben evitar los verbos en la designación de la misma.
- Por último es importante destacar que siempre las clases llevan su definición en singular.

```
class NombreClase {
    ......
}
```

Instancia de una Clase:

- A la hora de instanciar una clase, siempre definir el espacio para sus argumentos, indistintamente si los tiene o no.
- Incluir siempre los paréntesis, incluso aun cuando el constructor no incluya argumentos.

```
$foo = new NombreClase();
$foo = new NombreClase($arg1, $arg2);
```

Métodos:

- Los nombres de los métodos se describen mediante verbos, ya que los mismos detallan una acción.
- Es importante también que los métodos dentro de las clases a las cuales pertenecen, declaren su visibilidad: privada, protegida o pública:

```
public function agregar() {
    ......
}
```

- En caso de trabajar con argumentos en la declaración de un método, si se quiere predefinir el tipo de argumento se destaca que los mismos se deben igualar al tipo de dato que reciben

```
public function agregar($array=array(), $string='') {
    ......
}
```

- En caso de que un método se defina por una composición de palabras, las mismas deben estar separadas por un guión bajo:

```
public function agregar_foo() {
    ......
}
```

Propiedades

- En caso de definir propiedades de una clase, recordar siempre utilizar el identificador de propiedad o pseudovariable this:

```
$this->denominación  = ' ';
$this->edad = 0;
```

Variables

- Las variables deben ser nombradas con minúsculas, y las palabras deben estar separadas por un guión bajo en caso de estar referenciadas por la composición de dos o más palabras.

```
$foo = ' ';
$foo_bar = ' ';
```

3. Operadores Binarios

- Todos los operadores binarios que se encuentran entre dos valores, indefectiblemente deben llevar un espacio antes y después de los mismos.
- Nótese que en la definición de argumentos en una clase o en un método esta regla no aplica.
- Téngase en cuenta algunos ejemplos de operadores binarios como **+, -, *, /, ==, !=, <, >, entre otros:**

```
if($a>$b || $c<$b) {
      $resultado_foo = $a + $b;
}
```

4. Indentación y espacios en blanco

- Siempre para un mayor orden del código fuente, es importante llevar a cabo una correcta indentación del mismo. A continuación un breve ejemplo aplicando clase, método y estructura de control:

```
class NombreClase {
      public function agregar_foo() {
            if($a>$b || $c<$b) {
                  $resultado_foo = $a + $b;
            }
      }
}
```

- Cabe destacar que con respecto al manejo de espacios en blanco, se debe tener como prioridad el no dejar los mismos luego de finalizada una línea de código fuente.

5. Estructuras de Control

- Considérese como estructura de control a: **if, foreach, while, switch**, entre otros.
- Las sentencias de control deben tener un espacio entre la palabra clave de control y su paréntesis de apertura, para distinguirlas de las llamadas a funciones.
- Se recomienda utilizar siempre llaves, incluso en situaciones en las que técnicamente son opcionales. Aumenta la legibilidad y disminuye la probabilidad de errores lógicos que se presentan cuando se añaden líneas nuevas.
- Recordar también dejar un espacio entre la finalización del condicional en su paréntesis, con la llave que hace apertura de dicha estructura de control.

```
if($a != $b) {
      ......
}
```

6. Variables de tipo Constantes

- Las constantes siempre deben estar definidas en mayúsculas, con caracteres de guión bajo para separar aquellas que se encuentran especificadas con más de una palabra:

```
DB_HOST = 'localhost';
DB_NAME = 'db_example';
DB_USER = 'root';
DB_PASS = '123456';
```