Estándares de desarrollo

Tabla de contenidos

Estándares de desarrollo PrestaShop

Resumen

#PHP	#PHP	#SQL
#Variable names	#Strings	#Table names
#Assignments	#Comments	#SQL query
#Operators	#Return values	
#Statements	#Call	
#Visibility	#Tags	
#Method / Function names	#Indentation	
#Enumeration	#Array	
#Objects / Classes	#Bloc	
#Defines	#Security	
#Keywords	#Limitations	
#Constants	#Other	
#Configuration variables		

PHP

Nombres de variables

1. Correspondiente a la información de las bases de datos: $my_var
2. Correspondiente al algoritmo: $my_var
3. La visibilidad de una variable miembro no afecta su nombre: private $my_var

Asignaciones

1. Debe haber un espacio entre la variable y los operadores:

```
$my_var = 17;
$a = $b;
```

Operadores

1. "+", "-", "*", "/", "=" y cualquier combinación de ellos (ej. "/=") necesitan un espacio entre miembros de la izquierda y la derecha

```
$a + 17;
$result = $b / 2;
$i += 34;
```

2. "." no llevan espacio entre los miembros de la izquierda y la derecha

```
echo $a.$b;
$c = $d.$this->foo();
```

> ⚠ **Recomendación**
>
> Por motivos de rendimiento, no exceda el uso de la concatenación.

3. ".=" necesita un espacio entre los miembros de la izquierda y la derecha

```
$a .= 'Debug';
```

Declaraciones

1. if, elseif, while, for:deben presentar un espacio entre la palabra clave if y el paréntesis

```
if (<condition>)
while (<condition>)
```

2. Cuando una combinación de if y else else es utilizada y si ambas deben ofrecer un valor, el else debe ser evitado.

```
if (<condition>)
        return false;
return true;
```

> ⚠ **Recomendación**
>
> Le recomendamos un resultado por método / función

3. Cuando un método / función ofrece un valor booleano y el valor actual del método / función depende de ello, la declaración if debe ser evitada

```
public aFirstMethod()
{
        return $this->aSecondMethod();
}
```

4. Las pruebas deben ser agrupadas por "entidad"

```
if ($price AND !empty($price))
        [...]
if (!Validate::$myObject OR $myObject->id === NULL)
        [...]
```

Visibilidad

1. La visibilidad debe definirse en cada momento, incluso cuando se trata de un método público.
2. El orden de las propiedades del método debe ser: visibility static function name()

```
private static function foo()
```

Nombres de Método / Función

1. Los nombres del método y la función comienzan siempre con un carácter en minúscula y cada una de las palabras siguientes deben comenzar con un carácter en mayúsculas (CamelCase)

```
public function myExempleMethodWithALotOfWordsInItsName()
```

2. Los corchetes que se presentan en el código del método tienen que ser precedidos por un retorno

```
public function myMethod($arg1, $arg2)
{
        [...]
}
```

3. Los nombres de los métodos y las funciones deben ser nombres explícitos, por lo tanto nombres de función como "b()" o "ef()" están completamente prohibidos.

> **Excepciones**
>
> Las únicas excepciones son la función de traducción llamada "l()" y las funciones de depuración "p()", "d()".

Enumeración

Las comas deben estar seguidas (y sólo seguidas) por un espacio.

```
protected function myProtectedMethod($arg1, $arg2, $arg3 = null)
```

Objetos / Clases

1. El nombre del objeto debe colocarse en singular

```
class Customer
```

2. El nombre de la clase debe seguir el modelo CamelCase, excepto que la primera letra sea mayúscula

```
class MyBeautifulClass
```

Definiciones

1. Los nombres de las definiciones deben ser ingresados en mayúsculas
2. Los nombres de las definiciones deben contar con el prefijo "PS_" dentro del núcleo y el módulo

```
define('PS_DEBUG', 1);
define('PS_MODULE_NAME_DEBUG', 1);
```

3. Los nombres de las definiciones no permiten caracteres alfabéticos. Excepto “_”.

Palabras claves

Todas las palabras clave deben colocarse en minúsculas
ej. as, case, if, echo, null

Constantes

Las constantes deben estar en mayúsculas, excepto "verdadero" y "falso" y "nulo", que debe estar en minúsculas
ej. "ENT_NOQUOTE", "true"

Variables de configuración

Las variables de configuración siguen las mismas reglas que las definiciones

Cadenas

Las Cadenas deben ir entre comillas simples, nunca dobles

```
echo 'Debug';
$myObj->name = 'Hello '.$name;
```

Comentarios

1. Dentro de las funciones y métodos, sólo la etiqueta "//" de comentario es permitida
2. Después de la etiqueta "//" de comentario, un espacio “// Comment“ es necesario

```
// My great comment
```

3. La etiqueta "//" de comentario es permitida al final de una línea de código

```
$a = 17 + 23; // A comment inside my exemple function
```

4. Funciones y métodos externos, sólo las etiquetas "/**" y "*/" de comentario son permitidas

```
/* This method is required for compatibility issues */
public function foo()
{
// Some code explanation right here
[...]
}
```

5. Comentario PHP Doc Element es necesario antes de las declaraciones de método

```
/**
 * Return field value if possible (both classical and multilingual fields)
 *
 * Case 1 : Return value if present in $_POST / $_GET
 * Case 2 : Return object value
 *
 * @param object $obj Object
 * @param string $key Field name
 * @param integer $id_lang Language id (optional)
 * @return string
 */
protected function getFieldValue($obj, $key, $id_lang = NULL)
```

> **Para más información**
>
> Para más información acerca de la regla PHP Doc: vea este enlace útil

Devolución de valores

1. Las declaraciones de devolución no necesitan paréntesis, excepto cuando se trata de una expresión compuesta

```
return $result;
return ($a + $b);
return (a() - b());
return true;
```

2. Desarmar una función

```
return;
```

Llamada

La función de llamada precedida por una "@" está prohibida pero tenga cuidado con la llamada a la función / método con login / contraseña o ruta de argumentos.

```
myfunction()
// En el siguiente ejemplo colocamos una @ por motivos de seguridad
@mysql_connect([...]);
```

Etiquetas

1. Una línea en blanco debe dejarse después de la etiqueta de apertura de PHP

```
<?php

require_once('my_file.inc.php');
```

2. La etiqueta final de PHP está prohibida

Indentación

1. El carácter de tabulación ("\t") es el único caracter de indentación que se permite
2. Cada nivel de indentación debe ser representado por un solo caracter de tabulación

```
function foo($a)
{
        if ($a == null)
                return false;
        [...]
}
```

Matriz

1. La palabra clave de la matriz no debe ser seguida por un espacio

```
array(17, 23, 42);
```

2. La indentación debe ser de la siguiente forma cuando muchos datos se encuentran dentro de una matriz:

```
$a = array(
        36 => $b,
        $c => 'foo',
        $d => array(17, 23, 42),
        $e => array(
                0 => 'zero',
                1 => $one
        )
);
```

Bloque

Los corchetes están prohibidos cuando se define solamente una instrucción o una combinación de declaración

```
if (!$result)
        return false;

for ($i = 0; $i < 17; $i++)
        if ($myArray[$i] == $value)
                $result[] = $myArray[$i];
        else
                $failed++;
```

Seguridad

1. Toda la información de los usuarios (ingresada por ellos) debe ser moldeada.

```
$data = Tools::getValue('name');

$myObject->street_number = (int)Tools::getValue('street_number');
```

2. Todos los parámetros de método / función deben ser ingresados cuando son recibidos (con Array u Object).

```
public myMethod(Array $var1, $var2, Object $var3)
```

3. Todos los demás parámetros, deben ser moldeados cada vez que se usen, pero no cuando son enviados a otros métodos / funciones

```
protected myProtectedMethod($id, $text, $price)
{
        $this->id = (int)$id;
        $this->price = (float)$price;
        $this->callMethod($id, $price);
}
```

Limitaciones

1. Líneas de código fuente son limitadas a 120 caracteres
2. Líneas de funciones y métodos están limitadas a 80 con buenas justificaciones.

Otros

1. Está prohibido el uso de un ternario en otro ternario
2. Recomendamos utilizar && y || en sus condiciones
3. Por favor, no use los parámetros de referencia

SQL

Nombres de tablas

1. Los nombres de tablas deben comenzar con el prefijo de PrestaShop "*DB_PREFIX*"

```
[...] FROM `'. _DB_PREFIX_.'customer` [...]
```

2. Los nombres de tablas deben tener el mismo nombre que el objeto que reflejan
   por ejemplo "ps_cart"
3. Los nombres de tablas deben permanecer en singular
   por ejemplo "ps_order"
4. Información de idioma debe ser almacenada en una tabla denominada exactamente como uno de los objetos y con el sufijo "_lang" ej. "ps_product_lang"

Consulta SQL

1. Las palabras clave deben ser escritas en mayúsculas.

```
SELECT `firstname`
FROM `'. _DB_PREFIX_.'customer`
```

2. Las comillas inversas ("`") deben ser utilizadas alrededor de nombres de campo y tabla

```
SELECT p.`foo`, c.`bar`
FROM `'. _DB_PREFIX_.'product` p, `'. _DB_PREFIX_.'customer` c
```

3. Los alias de tablas deben ser realizados utilizando la primera letra de cada palabra y deben estar en minúsculas

```
SELECT p.`id_product`, pl.`name`
FROM `'. _DB_PREFIX_.'product` p
NATURAL JOIN `'. _DB_PREFIX_.'product_lang` pl
```

4. Cuando ocurran conflictos entre los alias de tablas, el segundo carácter también debe ser tomado.

```
SELECT ca.`id_product`, cu.`firstname`
FROM `'.DB_PREFIX.'cart` ca, `'. _DB_PREFIX_.'customer` cu
```

5. La indentación se debe realizar para cada cláusula

```
$query = 'SELECT pl.`name`
FROM `'.PS_DBP.'product_lang` pl
WHERE pl.`id_product` = 17';
```

6. Está prohibido realizar una combinación en la cláusula WHERE

Instalación del validador de código

Este es un breve tutorial sobre cómo instalar un validador de código en su PC y utilizarlo para validar sus archivos. El validador de código utiliza PHP CodeSniffer, el cual es un paquete de PEAR. El código estándar de PrestaShop fue creado específicamente para CodeSniffer y utiliza muchas reglas tomadas de las reglas existentes, agregándole reglas personalizadas con el fin de adaptarse mejor a nuestro proyecto.

Puede acceder a la regla del código PrestaShop utilizando SVN: http://svn.prestashop.com/branches/norm/ (debe realizar este paso antes de continuar con este tutorial)

Integración eclipse

Enlaces rápidos:

- Guía oficial de instalación
- Guía oficial de configuración

Si utiliza *Eclipse*, puede integrar la validación de código en el editor de texto usando un plugin, que es muy fácil de instalar.

La configuración del plugin también es muy simple. En la lista de paquetes disponibles, sólo seleccione PHP CodeSniffer y PEAR si aún no cuenta con ellos.

A continuación, tendrá que agregar el código estándar PrestaShop a las preferencias de Eclipse, para lograr esto tiene que dirijirse a "PHP Tools" y elegir el estándar PS que ha descargado anteriormente (vea el enlace arriba).

Consejo: si el archivo no es validado automáticamente, como debe ser, puede configurarlo en el menú "Preferences", "Validation". De lo contrario, haga clic derecho en la carpeta / archivo en la lista de archivos y seleccione "PHP Tools" en el menú contextual (el cual puede establecer como un acceso directo).

Línea de Comandos (Linux)

Puede instalar PHP CodeSniffer sin tener que utilizar Eclipse, utilizando la línea de comandos.

```
apt-get install php-pear
pear install PHP_CodeSniffer
svn co http://svn.prestashop.com/branches/norm/ /usr/share/php/PHP/CodeSniffer/Standards/Prestashop
phpcs --config-set default_standard Prestashop
```

Utilización del programa

Para instalar el validador como un programa que puede lanzar desde la línea de comandos, siga estos pasos:

1. Instalar PEAR: http://pear.php.net/
2. Instalar PHP CodeSniffer en PEAR: http://pear.php.net/package/PHP_CodeSniffer
3. Añadir la regla PrestaShop que ha descargado desde SVN antes, y colocarla en la carpeta "Standards" de PHP CodeSniffer.

Las distintas opciones para este comando están bien explicadas en su documentación. Por ahora, aquí está la manera fácil de ponerla en marcha:

```
$> phpcs --standard=/chemin/vers/norme/Prestashop /fichier/ou/dossier/a/valider/
```

Para mostrar sólo errores, no advertencias:

```
$> phpcs --standard=/chemin/vers/norme/Prestashop --warning-severity=99 /fichier/ou/dossier/a/valider/
```

Si ya ha instalado manualmente PHP CodeSniffer, el programa debe estar en la carpeta "scripts" de PEAR.

⚠ Usuarios de Windows: si bien el archivo phpcs.bat debe esta localizado en la carpeta "scripts", tal vez tenga que editarlo para que funcione correctamente (reemplazar las rutas con la suya):

```
path/to/php.exe \-d auto_apprend_file="" \-d auto_prepend_file{color} {color:#339966}\-d include_path="
path/to/PEAR/" path/to/pear/scripts/phpcs %\*
```

Integración del programa para la consola de Eclipse (opcional)

1. Haga clic en el botón "External tools" en la barra de iconos (una flecha verde que apunta a una carpeta roja pequeña).
2. Haga clic en la pestaña "External tools configuration".
3. Haga doble clic en "Program" con el fin de crear una configuración:
    a. Ubicación: ruta de acceso al programa de phpcs (o phpcs.bat para usuarios de Windows).
    b. Argumentos: los argumentos de la línea de comandos, por ejemplo --standard=Prestashop ${selected_resource_loc}