CBIRES Development standard

Summary

PHP

Variable names

1. Corresponding to data from databases: $my_var
2. Corresponding to algorithm: $my_var
3. The visibility of a member variable does not affect its name: private $my_var

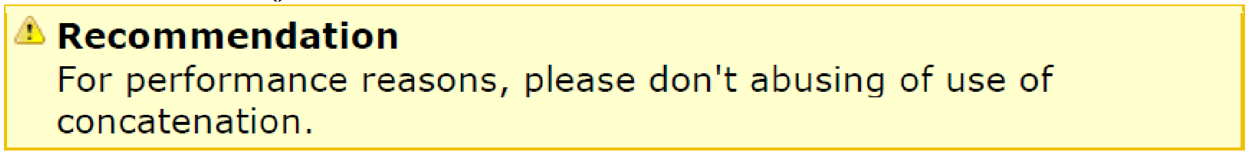
Assignments

1. There should be a space between variable and operators:
2.
3. $my_var = 17;
4. $a = $b;

Operators

1. "+", "-", "*", "/", "=" and any combination of them (e.g. "/=") need a space between left and right members
2.

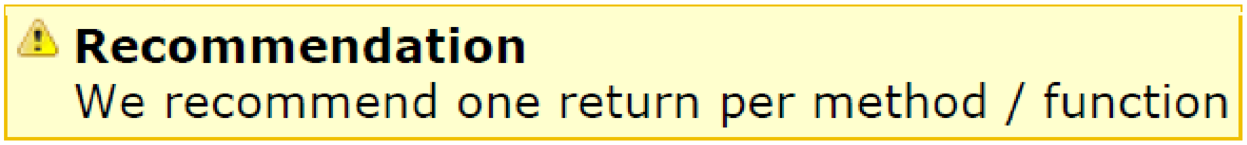
3. $a + 17;
4. $result = $b / 2;
5. $i += 34;
6. "." don't have space between left and right members
7.
8. echo $a.$b;
9. $c = $d.$this->foo();


Recommendation
For performance reasons, please don't abusing of use of concatenation.

10. ".=" need a space between left and right members
11.
12. $a .= 'Debug';

Statements

1. if, elseif, while, for: presence of a space between the if keyword and the bracket
2.
3. if (<condition>)
4. while (<condition>)
5. When a combination of if and else are used and that they should both return a value, the else has to be avoided.
6.
7. if (<condition>)
8. return false;
9. return true;


Recommendation
We recommend one return per method / function

10. When a method/function returns a boolean and the current method/function return depends on it, the if statement has to be avoided
11.
12. public aFirstMethod()
13. {
14.   return $this->aSecondMethod();
15. }
16.   Tests must be grouped by "entity"
17.
18. if ($price AND !empty($price))
19.   [...]
20. if (!Validate::$myObject OR $myObject->id === NULL)
21.   [...]

Visibility

1. The visibility must be defined everytime, even when it is a public method.
2. The order of the method properties should be: visibility static function name()
3.
4. private static function foo()

Method / Function names

1. Method and function name always begins with a lowercase character and each following words must begin with an uppercase character (CamelCase)
2.
3. public function myExempleMethodWithALotOfWordsInItsName()
4. Braces introducing method code have to be preceded by a carriage return
5.
6. public function myMethod($arg1, $arg2)
7. {
8.   [...]
9. }
10. Method and function names must be explicit, so such function names as "b()" or "ef()" are completly forbidden.

Enumeration

Commas have to be followed (and only followed) by a space.
protected function myProtectedMethod($arg1, $arg2, $arg3 = null)

Objects / Classes

1. Object name must be singular
2.
3. class Customer
4. Class name must follow the CamelCase practice except that the first letter is uppercase
5.
6. class MyBeautifulClass

Defines

1. Define names must be written in uppercase
2. Define names have to be prefixed by "CB_" inside the core
3.
4. define('CB_DEBUG', 1);
6. Define names does not allow none alphabetical characters. Except "_".

Keywords

All keywords have to be lowercase e.g. as, case, if, echo, null

Constants

Constants must be uppercase except for "true" and "false" and “null” which must be lowercase e.g. "ENT_NOQUOTE", "true"

Configuration variables

Configuration variables follow same rules as defines

Strings

Strings have to be surrounded by simple quotes, never double ones
echo 'Debug';
$myObj->name = 'Hello '.$name;

Comments

```
1. Inside functions and methods, only the "//" comment tag is allowed
2. After the "//" comment tag, a space “// Comment“ is required
3.
4. // My great comment
5. The "//" comment tag is tolerated at the end of a code line
6.
7. $a = 17 + 23; // A comment inside my exemple function
8. Outside funcions and methods, only the "/" and "/" comment tags are allowed
9.
10. /* This method is required for compatibility issues */
11. public function foo()
12. {
13.      // Some code explanation right here
14.       [...]
15. }
16. PHP Doc Element comment is required before the method declarations
17.
18. /**
19. * Return field value if possible (both classical and multilingual fields)
20. *
21. * Case 1 : Return value if present in $_POST / $_GET
22. * Case 2 : Return object value
23. *
24. * @param object $obj Object
25. * @param string $key Field name
```

26. * @param integer $id_lang Language id (optional)
27. * @return string
28. */
29. protected function getFieldValue($obj, $key, $id_lang = NULL)

For more informations
For more informations about the PHP Doc norm:
http://manual.phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tuto

Return values

1. Return statement does not need brackets except when it deals with a composed expression
2.
3. return $result;
4. return ($a + $b);
5. return (a() - b());
6. return true;
7. Break a function
8.
9. return;

Call

Function call preceded by a "@" is forbidden but beware with function / method call with login / password or path argmuments.

myfunction()
// In the following exemple we put a @ for security reasons
@pgsql_connect([...]);

Tags

1. An empty line has to be left after the PHP opening tag
2.
3. <?php
4.
5. require_once('my_file.inc.php');
6. The PHP ending tag is forbidden in files which contain only php code (e.g. not a mix of php with html)

Indentation

1. The tabulation character ("\t") is the only indentation character allowed
2. Each indentation level must be represented by a single tabulation character
3.
4. function foo($a)

```
5. {
6.         if ($a == null)
7.                 return false;
8.          [...]
9. }
```

Array

```
1. The array keyword must not be followed by a space
2.
3. array(17, 23, 42);
4. The indentation when too much datas are inside an array has to follow the following
5.
6. $a = array(
7.    36 => $b,
8.    $c => 'foo',
9.    $d => array(17, 23, 42),
10.             $e => array(
11.                         0 => 'zero',
12.                         1 => $one
13.               )
14. );
```

Bloc

Brasses are prohibited when they define only one instruction or a statement combination

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

Security

```
1. All user datas (datas entered by users) have to be casted.
2.
3. $data = Tools::getValue('name');
4.
5. $myObject->street_number = (int)Tools::getValue('street_number');
6. All method/function's parameters must be typed (when Array or Object) when
received.
7.
8. public myMethod(Array $var1, $var2, Object $var3)
```

9. For all other parameters they have to be casted each time they are used, but not when sent to other methods/functions
10.
11. protected myProtectedMethod($id, $text, $price)
12. {
13. $this->id = (int)$id;
14. $this->price = (float)$price;
15. $this->callMethod($id, $price);
16. }

Limitations

1. Source code lines are limited to 120 characters (with some exceptions)
2. Functions and methods lines are limited to 80 with good justifications

Other

1. It's forbidden to use a ternary into another ternary
2. We recommend to use && and || into your conditions
3. Please don't use reference parameters

SQL

Table names

1. Table names must begin with the CBIRES "DB_PREFIX" prefix
2.
3. [...] FROM `'. _DB_PREFIX_.'images` [...]
4. Table names must have the same name as the object they reflect e.g. "tbl_histogram" (if that is the case)

SQL query

1. Keywords must be written in uppercase.
2.
3. SELECT `firstname`
4. FROM `'. _DB_PREFIX_.'users`
5. Back quotes ("`") must be used around field names and table names
6.
7. SELECT p.`foo`, c.`bar`
8. FROM `'. _DB_PREFIX_.'images` p, `'. _DB_PREFIX_.'histograms` c
9. Table aliases have to be make by taking the first letter of each word, and must be lowercase
10.
11. SELECT p.`id_username`, pl.`name`
12. FROM `'. _DB_PREFIX_.'users` p

13. NATURAL JOIN `'. _DB_PREFIX_.'users` pl
14. When conflicts between table aliases occur, the second character has to be taken too
15.
16. SELECT ca.`id_feature`, cu.`firstname`
17. FROM `'.DB_PREFIX.'features` ca, `'. _DB_PREFIX_.'user` cu
18. Indentation has to be done for each clause
19.
20. $query = 'SELECT pl.`name`
21. FROM `'.CB_DBP.'features` pl
22. WHERE pl.`id_product` = 17';
23. It’s forbidden to make a join in WHERE clause