

# SQLFusion LLC coding standards

v1.3

Philippe Lewicki

## Table of Contents

PHP tags.....	2
Comments.....	2
Source code.....	3
Indent.....	3
Control Structure.....	3
Function calls.....	4
Function Definitions.....	4
Naming Convention .....	5
Class.....	5
Functions and Methods.....	5
Constants.....	5
Objects.....	6
Files and directories.....	6
Includes.....	6
Events.....	7
Classes.....	7
Reports.....	7
Forms.....	7
SavedQuery .....	7
Registry.....	7
File Formats.....	7
Database naming convention.....	8
HTML tag case convention.....	9

## PHP tags

The preferred PHP tags to use are `<?php ?>`.  
`<% %>` are also possible in development, but for all production code they need to be converted using the PAS asp2phptags scripts.

## Comments

Each file needs a header comment with the following:

```
<?php
// Copyright 2005 SQLFusion LLC           info@sqlfusion.com
// All rights reserved
```

Then a second comment describing the scripts.

All comments need to follow the phpDocumenter syntax.

Start with `/**` and ends with `*/`

Must contain a phrase with a title and then the description.

The `@package` is required and must contain the package name.

Example:

```
<?php
// Copyright 2005 SQLFusion LLC           info@sqlfusion.com
// All rights reserved
/**
 * Display Class
 *
 * It manage parameters from an internal array
 * and built a full url with the getUrl method.
 *
 * @package PASClass
 * @author Philippe Lewicki <phil@sqlfusion.com>
 * @version 3.3.0
 * @date 2005-02-18
 */
?>
```

Comments within structural code should be avoided. The mix of `//` comments with procedural code is not allowed.

Explanation about the code should be written in plain text english on the top of the Page, Class, Function, Event or section.

The comments will generate the documentation, so it should describe the code without the need to see it.

Comments can be long and descriptive even with examples.

## Source code

Our coding standards are similar to the Pear coding standards.

<http://pear.php.net/manual/en/standards.php>

## Indent

Use an indent of 4 spaces, with no tabs.

Here are vim rules for the same thing:

```
set expandtab
set shiftwidth=4
set softtabstop=4
set tabstop=4
```

## Control Structure

These include if, for, while, switch, etc. Here is an example if statement, since it is the most complicated of them:

```
<?php
if ((condition1) || (condition2)) {
    action1;
} elseif ((condition3) && (condition4)) {
    action2;
} else {
    defaultaction;
}
?>
```

Control statements should have one space between the control keyword and opening parenthesis, to distinguish them from function calls.

You are strongly encouraged to always use curly braces even in situations where they are technically optional. Having them increases readability and decreases the likelihood of logic errors being introduced when new lines are added.

For switch statements:

```
<?php
switch (condition) {
case 1:
    action1;
    break;

case 2:
    action2;
    break;

default:
    defaultaction;
    break;
}
?>
```

## ***Function calls***

Functions should be called with no spaces between the function name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon. Here's an example:

```
<?php
$var = foo($bar, $baz, $quux);
?>
```

## ***Function Definitions***

```
<?php
function fooFunction($arg1, $arg2 = '') {
    if (condition) {
        statement;
    }
    return $val;
}
?>
```

Arguments with default values go at the end of the argument list. Always attempt to return a meaningful value from a function if one is appropriate.

Here is a slightly longer example:

```
<?php
function connect(&$dsn, $persistent = false) {
    if (is_array($dsn)) {
        $dsninfo = &$dsn;
    } else {
        $dsninfo = DB::parseDSN($dsn);
    }

    if (!$dsninfo || !$dsninfo['phptype']) {
        return $this->raiseError();
    }

    return true;
}
?>
```

## Naming Convention

### ***Class***

Classes should be given descriptive names. Avoid using abbreviations where possible. Class names should always begin with an uppercase letter.

### ***Functions and Methods***

Functions and methods should be named using the "studly caps" style (also referred to as "bumpy case" or "camel caps"). Functions should in addition have the package name as a prefix, to avoid name collisions between packages. The initial letter of the name (after the prefix) is lowercase, and each letter that starts a new "word" is capitalized. Some examples:

connect()	getData()	buildSomeWidget()
-----------	-----------	-------------------

### ***Constants***

Constants should always be all-uppercase, with underscores to separate words. Prefix constant names with the uppercased name of the class/package they are used in.

## **Objects**

Instance of PAS Object follow the suggested naming convention:  
<first\_letter\_off\_the\_class>\_<name\_of\_the\_object>.

For example an event object will be named:

```
$e_myEvent = new Event("mypackage.myevent");
```

A report object:

```
$r_myReport = new Report("mypackage.myreport");
```

A form object:

```
$r_myForm = new ReportForm("mypackage.myform");
```

A query object:

```
$q_myQuery = new sqlQuery($conx);
```

A savedQuery object:

```
$sq_myQuery = new sqlSavedQuery($conx);
```

## **Files and directories**

Files in a PAS project follow a strict naming convention. PAS applications generate a lot of files, it is strongly suggested to follow a naming convention.

Most the files are lower case without spaces. "\_" or "." can be used for filenames based on multiple words.

Only the files that contain classes can use upper and lowercase.

There is 3 main type of files, the regular php files (.php), the includes php files (.inc.php) and xml files (.xml).

Includes files and xml files also have for prefix the name of the package they are part of.

For example for a report from the mailing tool package:  
report/mailingtools.list\_all\_emails.report.xml

All the regular php files are in root directory the have for extention .php.

## **Includes**

The includes are php files never displayed directly but included.

They are stored in the include directory.

They ends with : .inc.php

## **Events**

Are in the events directory and ends with .inc.php

## **Classes**

In the class directory are all the class files and they all end with:  
.class.php

## **Reports**

Are in the report directory and ends with .report.xml

## **Forms**

Are in the form directory and ends with .form.xml

## **SavedQuery**

Are in the savedquery directory and ends with .sq.xml

## **Registry**

Are in the registry directory and ends with .reg.xml

## **File Formats**

All scripts must:

- Be stored as ASCII text
- Use ISO-8859-1 character encoding
- Be Unix formatted

"Unix formatted" means two things:

- 1) Lines must end only with a line feed (LF). Line feeds are represented as ordinal 10, octal 012 and hex 0A. Do not use carriage returns (CR) like Macintosh computers do or the carriage return/line feed combination (CRLF) like Windows computers do.
- 2) There should be no line feed after the closing PHP tag (?>). This means that when the cursor is at the very end of the file, it should be on the right of the >, >\_ with no additional characters or line feed.

## Database naming convention

Database names, database tables and database fields should be in lower cases with no space or special characters, the “\_” is used to separate words.

All table should have a default primary key that is an autoincrement. The default primary key name is : id<tablename> (no space or “\_” between the id and <tablename>).

Using reserved words as database,table or fields name should be avoid.

Example:

```
SELECT `type`,
       SUM(`amount`)+SUM(`taxes`)-SUM(`discount`) AS total
FROM `expense`, `expensetype`
WHERE `expense`.`type`=`expensetype`.`idexpensetype`
      AND YEAR(`datepayed`) = 2003
GROUP BY `type`
ORDER BY `expensetype`.`name`
```

## HTML tag case convention

The HTML tags are lowercase. The tags attributes are lower case.

The values of the attributes are inside quote.

The HTML documents or PHP generated HTML should comply to XHTML 1.0 Transitional.

Go to <http://validator.w3.org/> to check your generated code.

Example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Intranet - SQLFusion</title>
    <link rel="stylesheet" type="text/css"
href="includes/dtree.css"/>
    <script type="text/javascript" language="javascript1.2"
src="includes/dtree.js"></script>
    <meta name="author" content="Philippe Lewicki"/>
    <meta name="copyright" content="SQLFusion LLC"/>
  </head>
  <body>
    <table>
      <tr>
        <td valign="top">
          <div class="dtree">
            Test here
          </div>
        </td>
      </tr>
    </table>
  </body>
</html>
```