



WORDCAMP
BOLOGNA 2013
DESIGNER+DEVELOPER

SABATO / 2013
9 FEBBRAIO
EVENTO DEDICATO A
DESIGNER E DEVELOPER

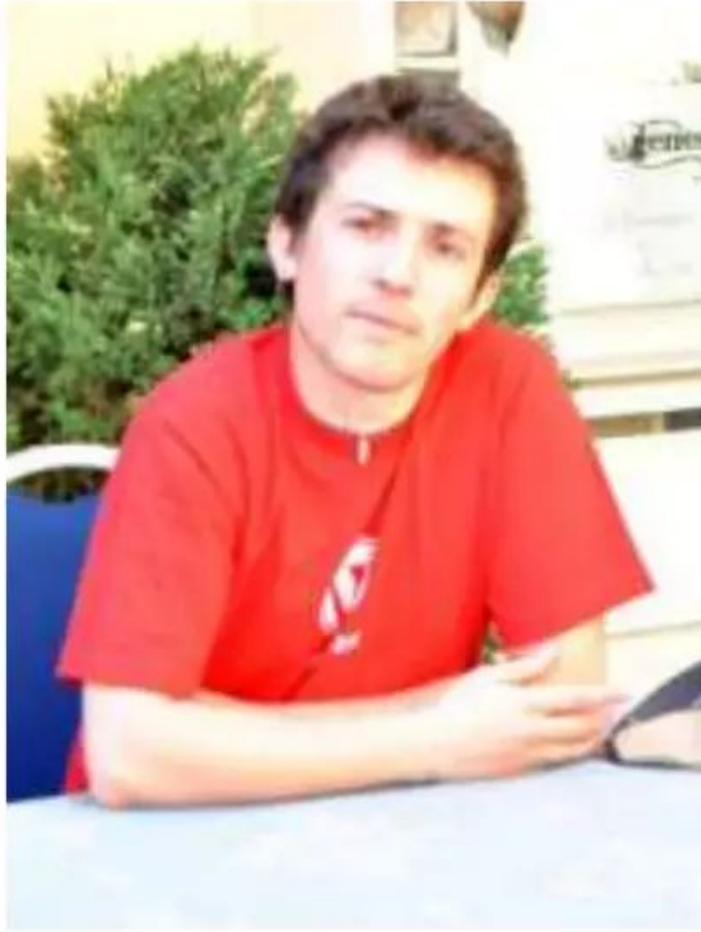


WordPress

Sviluppo e Qualità

di MAURIZIO PELIZZONE

WORDCAMP BOLOGNA - 9 FEBBRAIO 2013
@WORDCAMPBOLOGNA #WPCAMPBO13



Maurizio Pelizzone

<http://maurizio.mavida.com>

Twitter:

@miziomon

Linkedin:

<http://www.linkedin.com/in/mauriziopelizzone>

BIO



Dal 2001 co-titolare di una **mavida**, webagency che offre servizi e consulenza su tutto il territorio nazionale. Dal 2008 sviluppa plugin e template personalizzati con WordPress.

Nel tempo libero socializza con i giochi da tavolo.

GESTIRE PROGETTI COMPLESSI

Nello sviluppo di progetti complessi sono richiesti **efficienza, efficacia ed appropriatezza**.

Questi 3 aspetti possono rappresentare il cuore di un **modello di qualità** capace di favorire la gestione delle complessità.

Considerando come “macro-obiettivo” la pubblicazione di un sito saranno presentati i requisiti generalmente considerati di “**efficienza**” e gli strumenti che possono rendere il lavoro più “**efficace**”

LA QUALITÀ AIUTA A GESTIRE LA COMPLESSITÀ

EFFICACIA

EFFICACIA

“raggiungere un obiettivo entro scadenza”

EFFICIENZA

EFFICIENZA

***“raggiungere un obiettivo entro scadenza
con un consumo appropriato di risorse”***

- **Disponibilità** (accessibilità / user experience)
- **Sicurezza**
- **Performance**
- **Valore** (contenuti e comunicazione)

- Disponibilità (accessibilità / user experience)
- **Sicurezza**
- **Performance**
- Valore (contenuti e comunicazione)

- Ambiente di sviluppo in locale
- IDE (NetBeans, Eclipse, PhpStorm, ZendStudio)
- Coding Standards (WordPress, Zend, PSR)
- Controllo di versione (git, svn)
- Unit Testing
- Continuous integration

AMBIENTE DI SVILUPPO LOCALE

AMBIENTE DI SVILUPPO LOCALE



- Creare un **VirtualHost** su apache
- Gestire il file **hosts**
- Aggiungere database a MySQL
- Modifica wp-config.php per la gestione di ambienti diversi (locale,stage,produzione)

<VirtualHost *:80>

ServerName website.local

ServerAlias www.website.local

DocumentRoot "C:/path/to/your/site"

<Directory "c:\path\to\your\site">

Options Indexes FollowSymLinks MultiViews

RewriteEngine On

AllowOverride all

Require all granted

</Directory>

</VirtualHost>

...

```
if (file_exists(dirname(__FILE__) . '/wp-config-local.php')) {  
  
    include( dirname(__FILE__) . '/wp-config-local.php' );  
} else {  
  
    define('DB_NAME', 'dbname');  
    define('DB_USER', 'username');  
    define('DB_PASSWORD', 'password');  
    define('DB_HOST', 'localhost');  
    define('DB_CHARSET', 'utf8');  
    define('DB_COLLATE', '');  
  
} // end if  
  
define('WP_CONTENT_URL', 'http://' . $_SERVER['HTTP_HOST']);  
define('WP_SITEURL', 'http://' . $_SERVER['SERVER_NAME']);  
define('WP_HOME', 'http://' . $_SERVER['SERVER_NAME']);
```

INTEGRATED DEVELOPMENT ENVIRONMENT

- Gestione progetti
- Gestione sintassi
- Autocompletamento codice
- Navigazione classi e funzioni
- Integrazione (ftp,git,phing,unit test)

CODING STANDARS

WordPress Coding Standards

Languages: **English** • [日本語](#) • [Português do Brasil](#) • [中文简体](#) • ([Add your language](#))

Some parts of the WordPress code structure for PHP markup are inconsistent in their style. WordPress is working to gradually improve this by helping users maintain a consistent style so the code can become clean and easy to read at a glance.

Keep the following points in mind when writing code for WordPress, whether for core programming code, [Plugins](#), or [WordPress Themes](#). The guidelines are similar to [Pear standards](#) in many ways, but differ in some key respects.

See also [this post on the wp-hackers list](#). There is also a page on proposed [Inline Documentation](#) standards.

PHP

Single and Double Quotes

Use single and double quotes when appropriate. If you're not evaluating anything in the string, use single quotes. You should almost never have to escape quotes in a string, because you can just alternate your quoting style, like so:

```
echo '<a href="/static/link" title="Yeah yeah!">Link name</a>';  
echo "<a href='$link' title='$linktitle'>$linkname</a>";
```

An exception to this is JavaScript, which sometimes requires double or single quotes. Text that goes into attributes should be run through `esc_attr()` so that single or double quotes do not end the attribute value and invalidate the HTML and cause a security issue. See [Data Validation](#) for further details.

Indentation

Contents

[hide]

- [1 PHP](#)
 - [1.1 Single and Double Quotes](#)
 - [1.2 Indentation](#)
 - [1.3 Brace Style](#)
 - [1.4 include_once vs require_once](#)
 - [1.5 Regular Expressions](#)
 - [1.6 No Shorthand PHP tags](#)
 - [1.7 Remove Trailing Spaces](#)
 - [1.8 Space Usage](#)
 - [1.9 Formatting SQL statements](#)
 - [1.10 Database Queries](#)
 - [1.11 Naming Conventions](#)
 - [1.12 Self-Explanatory Flag Values for Function Arguments](#)

[Home Page](#)[WordPress Lessons](#)[Getting Started](#)[Working with WordPress](#)[Design and Layout](#)[Advanced Topics](#)[Troubleshooting](#)[Developer Docs](#)[About WordPress](#)

Codex Resources

[Community portal](#)[Current events](#)[Recent changes](#)[Random page](#)[Help](#)

Select a version: **1.12** ▼

[« Naming Conventions](#)
[Zend Framework Documentation Standards»](#)

Coding Style

PHP Code Demarcation

PHP code must always be delimited by the full-form, standard PHP tags:

1. `<?php`
2. `<?PHP`
3. `?>`

Short tags are never allowed. For files containing only PHP code, the closing tag must always be omitted (See [General standards](#)).

Strings

String Literals

When a string is literal (contains no variable substitutions), the apostrophe or "single quote" should always be used to demarcate the string:

1. `$a = 'Example String';`

String Literals Containing Apostrophes

When a literal string itself contains apostrophes, it is permitted to demarcate the string with quotation marks or "double quotes". This is especially useful for SQL statements:

1. `$sql = "SELECT `id`, `name` from `people` "`
2. `. "WHERE `name`='Fred' OR `name`='Susan'";`

This syntax is preferred over escaping apostrophes as it is much easier to read.

Variable Substitution

Table Of Contents

[Zend Framework Coding Standard for PHP](#) ▶

[PHP Code Demarcation](#)
[Strings](#)
[String Literals](#)
[String Literals Containing Apostrophes](#)
[Variable Substitution](#)
[String Concatenation](#)
[Arrays](#)
[Numerically Indexed Arrays](#)
[Associative Arrays](#)
[Classes](#)
[Class Declaration](#)
[Class Member Variables](#)
[Functions and Methods](#)
[Function and Method Declaration](#)
[Function and Method Usage](#)
[Control Statements](#)
[If/Else/Elseif](#)
[Switch](#)
[Inline Documentation](#)

[Explore](#) [Gist](#) [Blog](#) [Help](#)[miziomon](#)

PUBLIC

[php-fig / fig-standards](#)[Watch](#)[★ Star](#)

1,972

[Fork](#)

331

Code

Network

Pull Requests 13

Issues 30

Wiki

Graphs

branch: **master**

Files

Commits

Branches 1

Tags

[fig-standards](#) / [accepted](#) / **PSR-1-basic-coding-standard.md** **pmjones** 8 months ago move PSR-1 and 2 to 'accepted'

1 contributor



file | 173 lines (120 sloc) | 4.066 kb

Edit

Raw

Blame

History

Basic Coding Standard

This section of the standard comprises what should be considered the standard coding elements that are required to ensure a high level of technical interoperability between shared PHP code.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#).

1. Overview

- Files MUST use only `<?php` and `<?=>` tags.
- Files MUST use only UTF-8 without BOM for PHP code.
- Files SHOULD *either* declare symbols (classes, functions, constants, etc.) *or* cause side-effects (e.g. generate output, change .ini settings, etc.) but SHOULD NOT do both.
- Namespaces and classes MUST follow [PSR-0](#).
- Class names MUST be declared in `StudlyCaps`.
- Class constants MUST be declared in all upper case with underscore separators.
- Method names MUST be declared in `camelCase`.

CONTROLLO DI VERSIONE

- Integrato con i più importanti IDE
- Client multiplatforma
- Funziona in locale (no server)
- Ampia documentazione
- Github Community (online hosting using Git.)

UNIT TESTING

1. Automating Tests

2. PHPUnit's Goals

3. Installing PHPUnit

[PEAR](#)

[Composer](#)

[PHP Archive \(PHAR\)](#)

[Optional packages](#)

[Upgrading](#)

4. Writing Tests for PHPUnit

[Test Dependencies](#)

[Data Providers](#)

[Testing Exceptions](#)

[Testing PHP Errors](#)

[Testing Output](#)

[Assertions](#)

[assertArrayHasKey\(\)](#)

[assertClassHasAttribute\(\)](#)

[assertClassHasStaticAttribute\(\)](#)

[assertContains\(\)](#)

[assertContainsOnly\(\)](#)

[assertContainsOnlyInstancesOf\(\)](#)

[assertCount\(\)](#)

[assertEmpty\(\)](#)

[assertEqualXMLStructure\(\)](#)

[assertEquals\(\)](#)

[assertFalse\(\)](#)

[assertFileEquals\(\)](#)

[assertFileExists\(\)](#)

[assertGreaterThan\(\)](#)

[assertGreaterThanOrEqual\(\)](#)

[assertInstanceOf\(\)](#)

[assertInternalType\(\)](#)

[assertJsonFileEqualsJsonFile\(\)](#)

[assertJsonStringEqualsJsonFile\(\)](#)

[assertJsonStringEqualsJsonString\(\)](#)

[assertLessThan\(\)](#)

[assertLessThanOrEqual\(\)](#)

[assertNull\(\)](#)

[Prev](#)

[Next](#)

Chapter 3. Installing PHPUnit

There are three supported ways of installing PHPUnit. You can use the [PEAR Installer](#) or [Composer](#) to download and install PHPUnit as well as its dependencies. You can also download a [PHP Archive \(PHAR\)](#) of PHPUnit that has all required (as well as some optional) dependencies of PHPUnit bundled in a single file.

Note

Support for Composer and PHP Archive (PHAR) was added in PHPUnit 3.7 (and is known to be stable since PHPUnit 3.7.5). Earlier releases of PHPUnit are not available through these distribution channels.

Note

PHPUnit 3.7 requires PHP 5.3.3 (or later) but PHP 5.4.7 (or later) is highly recommended.

PHP_CodeCoverage, the library that is used by PHPUnit to collect and process code coverage information, depends on Xdebug 2.0.5 (or later) but Xdebug 2.2.0 (or later) is highly recommended.

PEAR

The following two commands (which you may have to run as `root`) are all that is required to install PHPUnit using the PEAR Installer:

```
pear config-set auto_discover 1
pear install pear.phpunit.de/PHPUnit
```

Caution

Depending on your OS distribution and/or your PHP environment, you may need to install PEAR or update your existing PEAR installation before you can proceed with the instructions in this section.

`sudo pear upgrade PEAR` usually suffices to upgrade an existing PEAR installation. The [PEAR Manual](#) explains how to perform a fresh installation of PEAR.

[Explore](#) [Gist](#) [Blog](#) [Help](#)

miziomon



PUBLIC

nb / **wordpress-tests**

Unwatch

Unstar

94

Fork

28

Code

Network

Pull Requests 0

Wiki

Graphs

Improved WordPress tests (moved to core) — [Read more](#)<http://unit-tests.trac.wordpress.org/>

Clone in Windows



ZIP

HTTP

SSH

Git Read-Only

<https://github.com/nb/wordpress-tests.git>

Read-Only access



branch: master

Files

Commits

Branches 1

Tags 1

wordpress-tests /

67 commits

update build.xml



scribu authored 6 months ago

latest commit d983429727

bin

7 months ago

Use a sha1 hash of the db_version and config params for version check. [[gcorne](#)]

lib

7 months ago

remove test_add_filter(). closes [#26](#) [[scribu](#)]

mu-plugins

7 months ago

Quick proof of concept using mu-plugins to get plugins active by defa... [[johnpbloch](#)]

.gitignore

8 months ago

Added a Phing build file that is configured to run tests and generate... [[tierra](#)]

README.md

7 months ago

Update master [[scribu](#)]

all.php

7 months ago

Use is_subclass_of() to find tests. [[gcorne](#)]

bootstrap.php

7 months ago

rename init.php to bootstrap.php, to match unit-tests.trac [[scribu](#)]

build.xml

6 months ago

update build.xml [[scribu](#)]

phpunit.xml

7 months ago

rename init.php to bootstrap.php, to match unit-tests.trac [[scribu](#)]

test_go_to.php

a year ago

use phpunit.xml instead to load init.php [[scribu](#)]

test_test.php

a year ago

use phpunit.xml instead to load init.php [[scribu](#)]

test_texturize.php

a year ago

use phpunit.xml instead to load init.php [[scribu](#)]

unittests-config-sample.php

8 months ago

Allow configuration of PHP executable. [[maxcutler](#)]

Codex

Codex tools: [Log in](#)

Theme Unit Test

Languages: [English](#) • [日本語](#) • ([Add your language](#))

(Last Edited: 2012-11-29)

This step in [Theme testing](#) correlates to a WordPress export (WXR) file that you can import into a WordPress installation to test your Theme. To be clear, this is only one step in testing your Theme. See [Theme testing](#) for a full guide.

Setup

Test Environment Setup

1. Download test data: <https://wpcom-themes.svn.automattic.com/demo/theme-unit-test-data.xml> (updated 11/29/12 18:11 UTC)
2. Import test data into your WordPress install. Do this by going to Tools -> Import.
3. Read [How To Join WPTRT](#), and follow all test-environment setup instructions

WordPress Settings

Adjust WordPress settings as follows:

- Settings -> General: set the Site Title to something fairly long, and set the Tagline to something even longer. These settings will facilitate testing how the Theme handles these values.
- Settings -> Reading: set "Blog pages show at most" to 5. This setting will ensure that index/archive pagination is triggered.
- Settings -> Discussion: enable Threaded Comments, at least 3 levels deep. This setting will facilitate testing of Theme comment list styling.
- Settings -> Discussion: enable Break comments into pages, and set 5 comments per

Contents

[\[hide\]](#)

- 1 Setup
 - 1.1 Test Environment Setup
 - 1.2 WordPress Settings
- 2 Template Hierarchy Index Pages
- 3 Static Front Page
- 4 404 Page
- 5 Search Results Page
- 6 Blog Posts Index Page
 - 6.1 Scheduled Post
 - 6.2 Draft Post
 - 6.3 Layout Test
 - 6.4 Readability Test
 - 6.5 Post Format Tests (All)
 - 6.6 Post Format Test Gallery
 - 6.7 Post Format Test Image (Linked)
 - 6.8 Post Format Test Image

[Home Page](#)[WordPress Lessons](#)[Getting Started](#)[Working with WordPress](#)[Design and Layout](#)[Advanced Topics](#)[Troubleshooting](#)[Developer Docs](#)[About WordPress](#)

Codex Resources

[Community portal](#)[Current events](#)[Recent changes](#)[Random page](#)[Help](#)

- Curva di apprendimento rigida
- Difficile scrivere test “*furbi*”
- Difficile scrivere test dopo...
- TDD non sempre è possibile
- “efficienza” dopo molto esercizio

CONTINUOUS INTEGRATION

Home[News](#)[Features](#)[License](#)[Download](#)[Installation](#)[Documentation](#)[Buzz about Phing](#)[Projects using
Phing](#)**Support**[Mailing Lists](#)[Reporting a Bug](#)[IRC](#)[Twitter](#)**Development**[Source](#)[How to
Contribute](#)[Contributors](#)[User
Contributions](#)

Welcome to the website of The Phing Project

PHing Is Not GNU make; it's a PHP project build system or build tool based on [Apache Ant](#). You can do anything with it that you could do with a traditional build system like GNU make, and its use of simple XML build files and extensible PHP "task" classes make it an easy-to-use and highly flexible build framework.

Features include running PHPUnit and SimpleTest unit tests (including test result and coverage reports), file transformations (e.g. token replacement, XSLT transformation, Smarty template transformations), file system operations, interactive build support, SQL execution, CVS/SVN/GIT operations, tools for creating PEAR packages, documentation generation (DocBlox, PhpDocumentor) and much, much more.

If you find yourself writing custom scripts to handle the packaging, deploying, or testing of your applications, then we suggest looking at the Phing. Phing comes packaged with numerous out-of-the-box operation modules (tasks), and an easy-to-use OO model to extend or add your own custom tasks.

Phing provides the following features:

- Simple XML buildfiles
- Rich set of provided tasks
- Easily extendable via PHP classes
- Platform-independent: works on UNIX, Windows, Mac OSX
- No required external dependencies
- Built for PHP5

To download, visit the [Download page](#).

For more info, refer to the following documentation links:

- [Documentation home](#)
- [User guide](#)
- [API documentation](#)

We try to follow the [Semantic Versioning](#) principle.

Recent Phing News

01/03/2013

- [Users/News/SVN deprecation \(diff\)](#)

11/29/2012

- [Users/News/Phing 2.4.14 released](#)

- Esecuzione Unit Test
- Backup automatici
- Caricamento FTP
- Gestione file (copia/sposta/cancella)
- Integrazione git / svn

COME CONTROLLARE LA QUALITÀ?

- Checklist
- Benchmark

CHECKLIST SICUREZZA

- Custom Setup
- Pianificato manutenzione
- Unit Testing
- Penetration test
- Backup
- Automazione deploy

BENCHMARK PERFORMANCE

- Tempo caricamento pagina
- Tempo di esecuzione
- Numero di Query
- Cache interna
- Cache html su parti di template



Grazie



Pelizzone Maurizio

@miziomon

maurizio@mavida.com

<http://www.mavida.com>

<http://maurizio.mavida.com>

