Bonnes Pratiques de Développement PHP

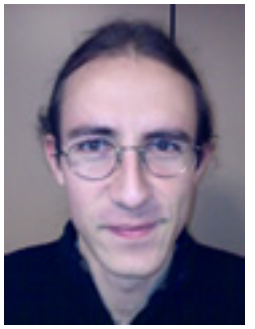
Pascal MARTIN – SQLI

Forum PHP 2009, Paris

- Quelques mots
- Environnement de travail
- Contrôle de source
- Développer [en PHP]
- Normes de codage
- Tests Automatisés
- Documentation
- Intégration continue
- Déploiement
- Encore un peu ?

afup MySQL User Group

A Propos de moi



- Pascal MARTIN
- Expert Technique PHP chez SQLI
  - Membre de la Cellule Architecture PHP
    - Capitalisation
    - Veille Technologique
    - Interventions au lancement de projets
- Blog Perso
  - http://blog.pascal-martin.fr/
- @pascal_martin

A Propos de vous ?

- Développeurs PHP
  - Curieux / Passionnés
  - Voulant / pouvant améliorer les processus
- Vos / Nos Projets ?
  - Maintenables ?
  - Organisés ?
  - Documentés ?
  - Et si vous partez en vacances ?

A Propos de cette présentation

- Quelques notes
  - Expérience pro / perso
  - Retours d'autres projets
- Pas une vérité absolue !
- Ni un ensemble de solutions toute prêtes !
- Plein d'autres choses fantastiques à découvrir ;-)

Plusieurs niveaux

- Bonnes pratiques : plusieurs niveaux
- Sur tout le projet
  - Relation avec le client
  - Configuration des serveurs
  - Code PHP
  - Processus de développement
  - Environnement de travail

Environnement de travail

afup MySQL User Group

« Ton fichier PHP est pourri, notepad m'affiche tout sur une seule ligne ! »

Utiliser un IDE – 1

- J'admets
  - Machine puissante
  - Temps d'adaptation / prise en main
  - « Trop » dans certains cas

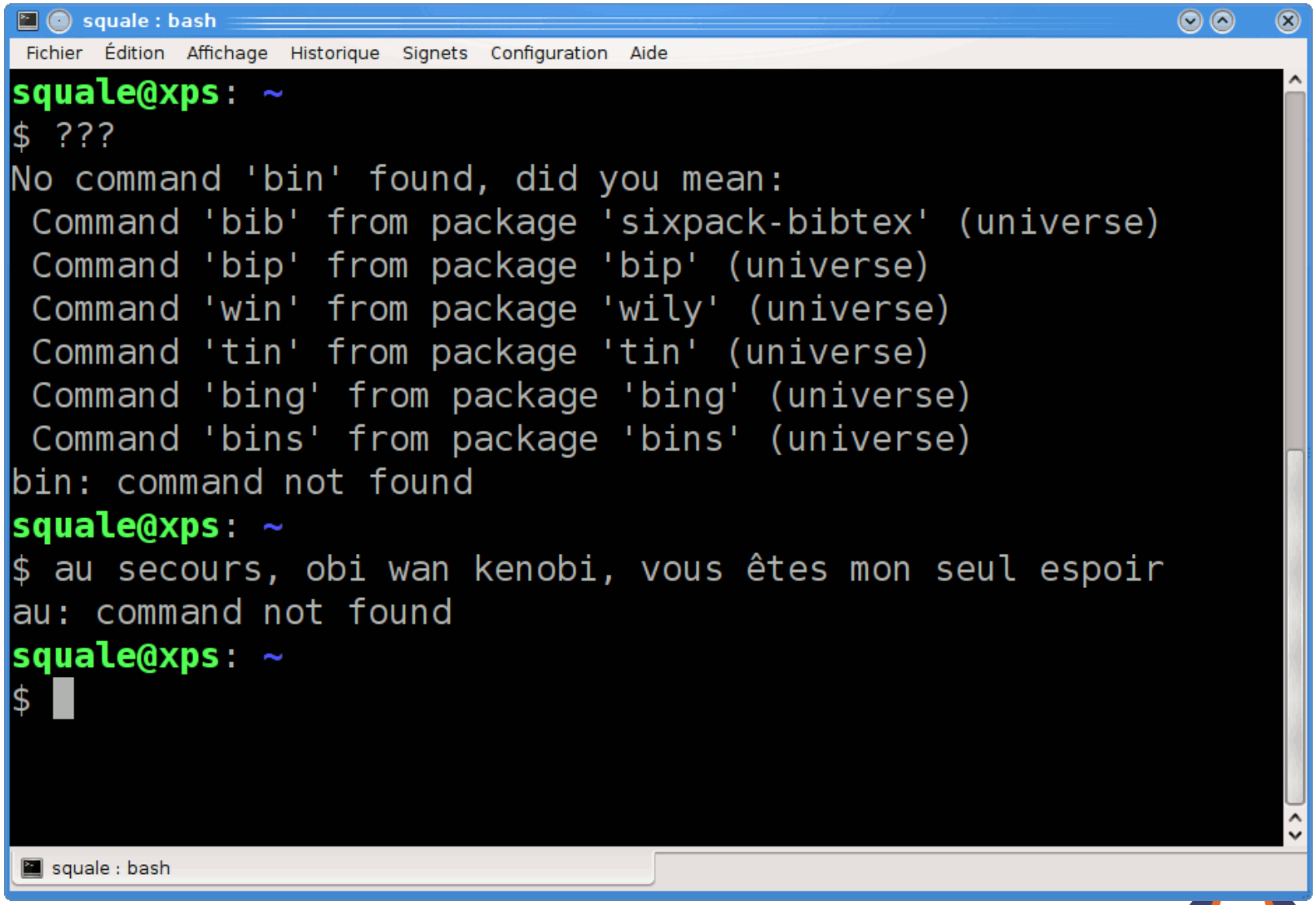
Utiliser un IDE – 2

- Mais
  - Gros projet, en équipe
  - Framework / bibliothèques
    - Plein de code peu connu
- Quels IDE ?
  - Eclipse PDT / Zend Studio
  - Netbeans
- Débugger intégré ;-)

Linux – 1

- LAMP => Linux
  - Serveurs de production
- Ligne de commande
  - « Heu... Je clique où ??? »
  - « WTF ??? »
- Différences de comportement de PHP !

Linux – « Heu ??? Au secours ! »



Linux – 3

- Quelques notions
  - Peut être salvateur
    - Incident sur un serveur de production
    - « Gars qui connait » absent ?
  - Mais aussi pour du développement
    - Suivi logs
    - Notions de configuration
- Solution possible : Virtualisation
  - Développement sous Windows
  - Serveur LAMP : machine virtuelle

Contrôle de source

« Quand quelqu'un veut commencer à modifier un fichier, il le *« réserve »* sur MSN »

Contrôle de source

- Quelques questions
  - Est-ce que quelqu'un a modifié quelque chose ?
  - Comment synchroniser entre développeurs ?
  - Comment obtenir la version du 14 avril 2008 ?
  - Qu'est-ce qui a changé entre le 19 mai et le 23 juillet ?
- Deux types de contrôles de source
  - Centralisés
  - Distribués

VCS Centralisé

- Subversion – SVN
- Principe
  - Travail local sur checkout du repository
  - Automatisations basées sur les commits
- Mais
  - Serveur nécessaire !
  - Peu de travail sans connexion
- Excellent support !
  - Forte intégration

VCS Décentralisé

- Git, Bazaar, Mercurial, Darcs, …
- Principe
  - Travail sur une branche / copie locale
  - Partage des modifications entre repositories
- Quelques avantages
  - Pas besoin de serveur centralisé
  - Facilite le fork local
- Moins intégré ? Moins répandu ?
  - Moins de processus d'automatisation ?

Développer [en PHP]

« Votre citation, ici ;-) »

afup MySQL User Group

Ne développez pas !

- Utilisez ce qui existe déjà
  - Vous n'êtes pas le seul à vouloir une application qui fasse X, Y, et Z
- Applications entières
  - CMS, CRM, E-Commerce,
  - Libres ?
- Adaptez
  - Ne développez que des modules
    - Là où vos besoins sont spécifiques

Framework

- Grand sujet de débat
- Ne pas réinventer la roue
  - Framework ~~*perso*~~
    - Zend Framework, symfony, Cake, ...
  - Bibliothèques
    - Composants de Frameworks
    - PEAR
- Ou pas ?

afup MySQL User Group

Normes de codage

afup MySQL User Group

« Il est illisible, son code,
je comprends rien ! »

afup MySQL User Group

Pourquoi une norme de codage ?

- Uniformité
- Consistance
- Lisibilité
  - Se concentrer sur le code
  - Ne pas perdre du temps à le formater mentalement
- Facilite la collaboration

Quoi ? Comment ?

- Indentation
- Position des parenthèses / accolades
- Retours à la ligne
- Nommage de variables / classes / méthodes
- Nommages / organisation des fichiers
- Longueurs de lignes
- Documentation
- ...

afup MySQL User Group

Quelques exemples – 1

- Classes
  - MonNomDeClasse
- Méthodes, variables, propriétés
  - monNomDeMethode
- Propriétés privées / protégées
  - _monNomDePropriete
- Constantes
  - MA_CONSTANTE

Quelques exemples – 2

- Fichiers et répertoires
  - Classe
    - Societe_Generateur_Document_Pdf
  - Répertoires / Fichier
    - Societe/Generateur/Document/Pdf.php
  - Une classe par fichier
- Conversion des « _ » en « / »
  - Facilite l'autoload
  - PHP 5.3 : Généralisable aux namespaces

Quelques exemples – 3

- Indentation
  - Espaces vs tabulations
  - 2 ou 4 espaces
- Position des accolades ?

```
public static function delete($idAccount)
{
    try {
        $db->beginTransaction();
        if (...) {
            throw new Exception("...");
        }
        // ...
        $db->commit();
    } catch (Exception $e) {
        $db->rollback();
    }
} // delete
```

Quelques plus qu'exemples

- Soyons raisonnables
  - Classes : 10-20 méthodes
  - Méthodes : 30-50 lignes
  - Fichiers : pas 3000+ lignes !
- Respect de la structuration
  - MVC : ne pas mélanger les couches
  - Pas de requête SQL dans les templates

Standards connus et reconnus

- N'inventez pas votre propre standard
- Utilisez un standard connu, répandu, accepté
  - Celui de votre Framework / Application
    - http://framework.zend.com/manual/en/coding-standard.html
    - http://trac.symfony-project.org/wiki/HowToContributeToSymfo
    - http://drupal.org/coding-standards
  - Normes de codage PEAR
    - http://pear.php.net/manual/en/standards.php
- Configurez votre IDE

Moyen de contrôle ?

- Une norme de codage
  - C'est bien
- Qu'elle soit respectée
  - C'est mieux !
- Toute l'équipe doit utiliser le même standard
- Moyen de contrôle nécessaire
  - (semi) Automatisé

PHP_CodeSniffer

- PHP_CodeSniffer
  - http://pear.php.net/package/PHP_CodeSniffer/
  - Analyse d'un code source + rapport

```
$ phpcs -s /path/to/code/myfile.php

FILE: /path/to/code/myfile.php
----------------------------------------------------------------------
FOUND 5 ERROR(S) AND 1 WARNING(S) AFFECTING 5 LINE(S)
----------------------------------------------------------------------
  2 | ERROR   | Missing file doc comment (PEAR.Commenting.FileComment)
 20 | ERROR   | PHP keywords must be lowercase; expected "false" but found
    |         | "FALSE" (Generic.PHP.LowerCaseConstant)
 47 | ERROR   | Line not indented correctly; expected 4 spaces but found 1
    |         | (PEAR.WhiteSpace.ScopeIndent)
 47 | WARNING | Equals sign not aligned with surrounding assignments
    |         | (Generic.Formatting.MultipleStatementAlignment)
 51 | ERROR   | Missing function doc comment
    |         | (PEAR.Commenting.FunctionComment)
 88 | ERROR   | Line not indented correctly; expected 9 spaces but found 6
    |         | (PEAR.WhiteSpace.ScopeIndent)
----------------------------------------------------------------------
```

Mise en place

- Dès le début du projet
  - Sinon : énormément de travail
- Accepté / reconnu
  - Par toute l'équipe
  - Y compris le management
- Qualité !
- Et si le code est horrible ?
  - http://pear.php.net/package/PHP_Beautifier

Tests Automatisés

afup MySQL User Group

« On a testé il y a 6 mois, quand on a mis en prod ; depuis, on n'a pas le temps »

« Des fois, ça plante »

Fatal Error on line X

- Code peut casser
  - Changement dans un fichier...
  - … Quel impact sur le reste de l'application ?
- Nécessité de tester !
- « Pas le temps »
  - Tester manuellement = long !
    - Rarement fait
    - Uniquement sur quelques pages
  - Tests automatiques

Tests Unitaires

- Tester une « unité »
  - Un module / Une classe / Une méthode
  - Indépendamment de tout le reste
- Tests Automatisés
  - Détection de régressions
  - Joués après chaque modification
  - Rapides – quelques secondes
- Code Coverage

Tests Unitaires

- PHPUnit
  - Framework de Tests Unitaires
  - http://www.phpunit.de/
- Difficultés ?
  - Temps / coût
  - Code existant « non testable »
- Avantages
  - Facilite le rework, les évolutions
  - Confiance !

Test Driven Development

- Écriture d'un test qui échoue
  - Car implémentation non écrite
  - Définit le comportement attendu
- Puis, écriture du code
  - Qui passe le test
- Éventuellement
  - Rework du code
  - Sécurité : il est testé

Tests d'Intégration

- Tester
  - Non plus des composants unitaires
  - Mais l'assemblage des briques
  - Voire toutes les couches de l'application
- Deux possibilités
  - « Pseudo » requêtes HTTP
    - Rapide, facile à mettre en place
  - Utilisation d'un « vrai » navigateur
    - Teste aussi l'interface utilisateur
    - Support Javascript

Zend_Test

- Classe Zend Framework
  - Test via le MVC – sans navigateur

```
public function testLoggingInShouldBeOk()
{
    $this->dispatch('/login/login');
    $csrf = $this->_getLoginFormCSRF();
    $this->resetResponse();
    $this->request->setPost(array(
        'login'  => 'pmartin',
        'password' => '123456',
        'csrfLogin' => $csrf,
        'ok' => 'Login',
    ));
    $this->request->setMethod('POST');
    $this->dispatch('/login/login');
    $this->assertRedirectTo('/');
    $this->assertTrue(Zend_Auth::getInstance()->hasIdentity());
}
```

Selenium RC

- Pilotage d'un navigateur
  - Intégré à PHPUnit !

```
class functionnal_Front_Selenium_ArchiveTest
  extends functionnal_TestsUtils_SeleniumTestCase {
  public function testNavigationArchive() {
    $this->open("/");
    $this-
>click("//div[@id='archiveWidget']/ul/li[1]/a/span[1]");
    $this->waitForPageToLoad("30000");
    $this->assertEquals("Juin 2008 - Mon premier blog",
                        $this->getTitle());
    $this->assertTrue($this->isTextPresent("Septième post"));
    $this-
>click("//div[@id='archiveWidget']/ul/li[2]/a/span[1]");
    $this->waitForPageToLoad("30000");
    $this->assertTrue($this->isElementPresent("link=Troisième
post"));
  }
}
```

Quoi tester ?

- Écrire des tests prend du temps
- Tout ne peut pas être testé
- Il n'est pas utile / indispensable de tout tester
  - Ne pas (re-) tester le Framework !
- Tester
  - Ce qui est important
  - Ce qui risque de casser
  - Ce qu'on modifie souvent ?
  - Ce que l'on va re-worker ?

Maintenir les Tests

- Suivre le déroulement des tests
- Tests non maintenus
  - Inutiles
  - Dangereux : « fausse sécurité »
- Maintenance
  - Au fil de l'eau
  - Quand modifications du comportement
  - Requiert du temps

Documentation

afup MySQL User Group

« Comment ça marche ? »

« Heu...Vas voir dans le code »

Plusieurs documentations

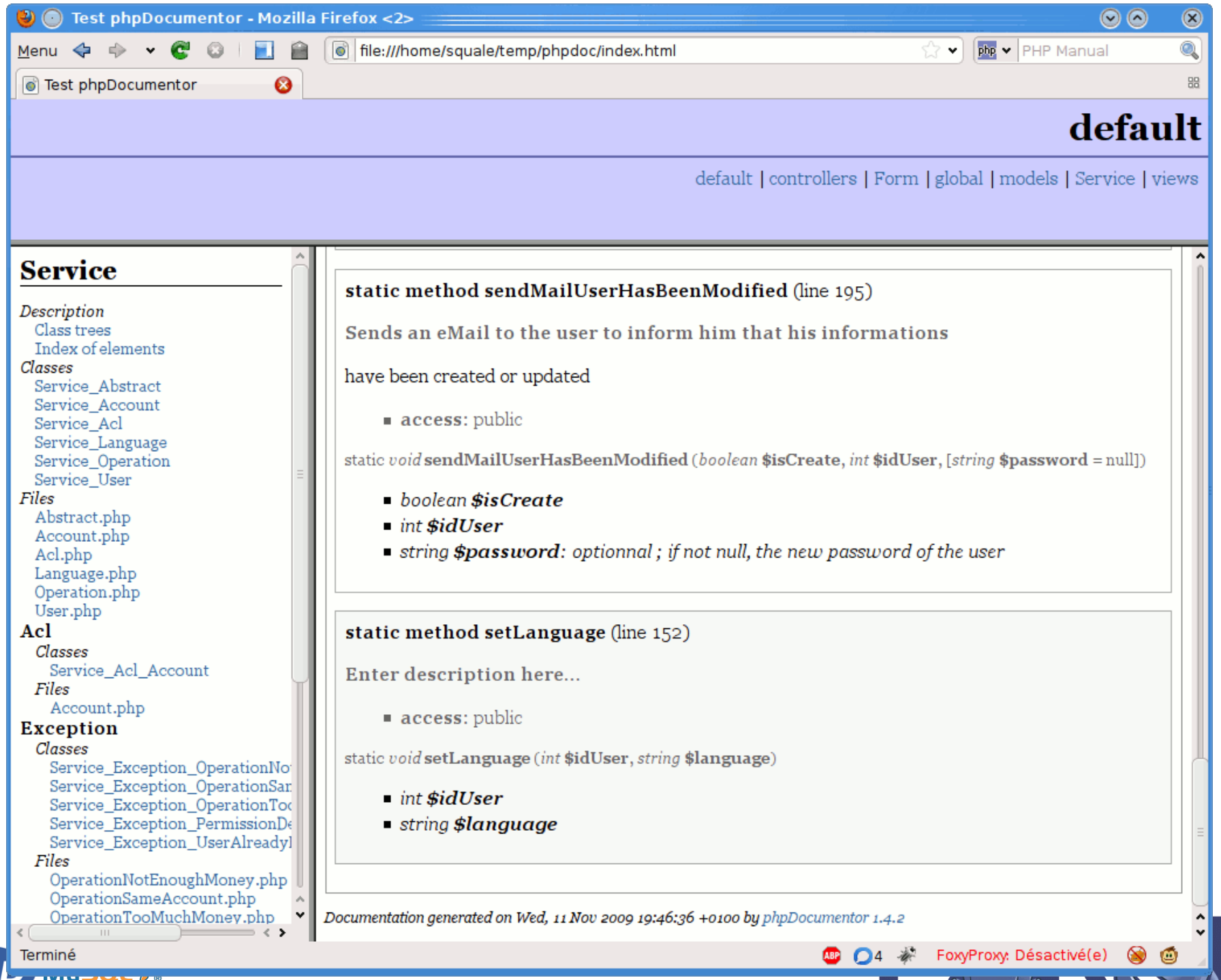
- Documentation d'API
  - Technique : comment développer
  - Classes, méthodes, paramètres, retours, ...
- Documentation d'application
  - Fonctionnel : comment utiliser l'application
  - Scénarios d'utilisation, captures d'écrans, ...
- Procédures
  - Comment déployer en production
  - Reprise sur incident

Documentation d'API

- Docblocks
  - Fichiers, Classes, Méthodes
- Quelques tags utiles
  - @param, @return, @throws
  - @package, @subpackage
  - @author, @see, @todo
- Génération
  - PhpDocumentor
  - Doxygen

Exemple – code

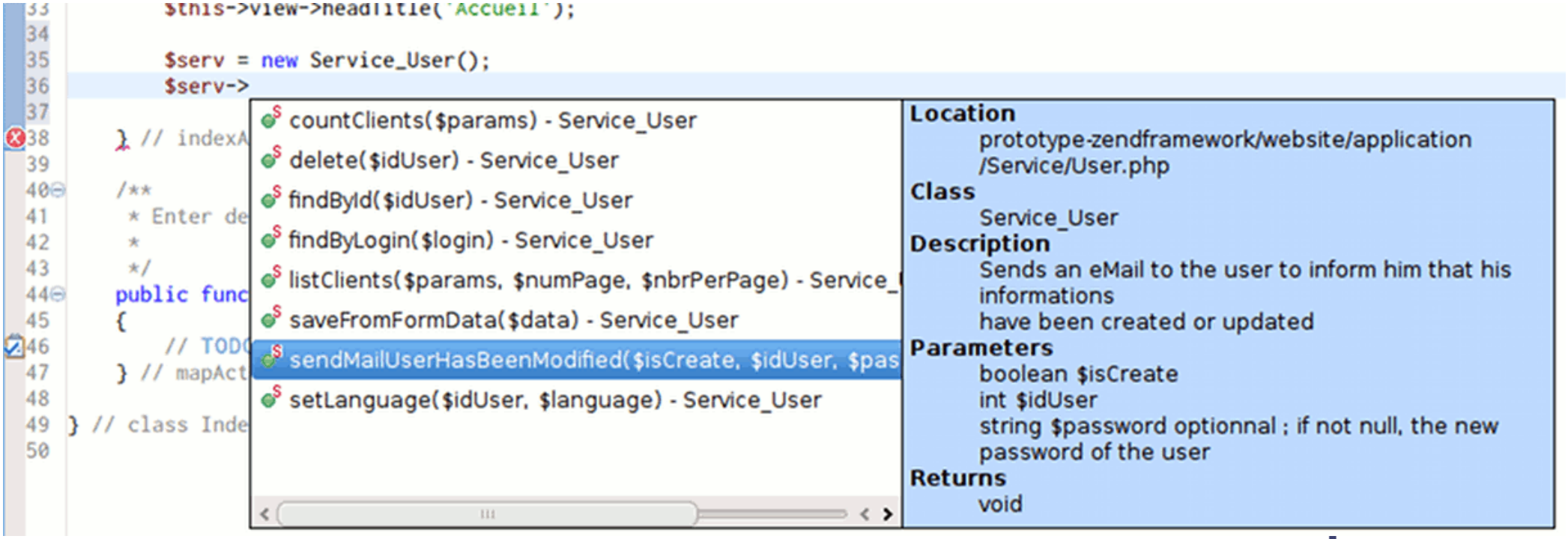
```
/**
 * Some description here
 *
 * @package Service
 * @author Pascal MARTIN
 * @todo Use some real templating mecanism
 */
class Service_User
{
    /**
     * Sends an eMail to the user to inform him that his informations
     * have been created or updated
     *
     * @param boolean $isCreate
     * @param int $idUser
     * @param string $password optionnal ; if not null, the new password of
the user
     * @return void
     */
    public static function sendMailUserHasBeenModified($isCreate, $idUser,
$password=null) {
        // ...
    } // sendMailUserHasBeenModified
}
```


Test phpDocumentor - Mozilla Firefox <2>
Menu
file:///home/squale/temp/phpdoc/index.html
PHP Manual
Test phpDocumentor
default
default | controllers | Form | global | models | Service | views
Service
Description
Class trees
Index of elements
Classes
Service_Abstract
Service_Account
Service_Acl
Service_Language
Service_Operation
Service_User
Files
Abstract.php
Account.php
Acl.php
Language.php
Operation.php
User.php
Acl
Classes
Service_Acl_Account
Files
Account.php
Exception
Classes
Service_Exception_OperationNo
Service_Exception_OperationSar
Service_Exception_OperationToc
Service_Exception_PermissionDe
Service_Exception_UserAlreadyI
Files
OperationNotEnoughMoney.php
OperationSameAccount.php
OperationTooMuchMoney.php
static method sendMailUserHasBeenModified (line 195)
Sends an eMail to the user to inform him that his informations
have been created or updated
access: public
static void sendMailUserHasBeenModified (boolean $isCreate, int $idUser, [string $password = null])
boolean $isCreate
int $idUser
string $password: optionnal ; if not null, the new password of the user
static method setLanguage (line 152)
Enter description here...
access: public
static void setLanguage (int $idUser, string $language)
int $idUser
string $language
Documentation generated on Wed, 11 Nov 2009 19:46:36 +0100 by phpDocumentor 1.4.2
Terminé
FoxyProxy: Désactivé(e)

Pourquoi une doc d'API ?

- Intégration IDE
- Documentation Technique
- Utilisable en PHP
- Mais
  - N'empêche pas d'écrire du code « auto-documenté »
    - Noms de méthodes / classes / variables clairs
    - Significatifs
  - Facilite la compréhension

Doc d'API et IDE



afup MySQL User Group

FORUM

Documentation utilisateurs

- Pensez-y !
- Dépend
  - Du type d'application
  - Du public visé
- Comment est-ce que votre logiciel sera utilisé, sinon ?
- Vous êtes aussi utilisateurs ;-)

Doc utilisateur : Docbook

- Format XML
  - Fréquemment utilisé pour les logiciels OSS
  - Documentation de PHP !
- Plusieurs formats de sortie
  - HTML, PDF, …
- WYSIWYG
  - XMLmind XML Editor
    - http://www.xmlmind.com/xmleditor/
    - Gratuit en version personnelle

Docbook – exemple

- http://fr.php.net/manual/fr/functions.user-defined.php

```
<?xml version="1.0" encoding="utf-8"?>
 <chapter xml:id="language.functions" xmlns="http://docbook.org/ns/docbook">
  <title>Les fonctions</title>
  <sect1 xml:id="functions.user-defined">
   <title>Les fonctions définies par l'utilisateur</title>
   <para>
    Une fonction peut être définie en utilisant la syntaxe suivante :
   </para>
   <para>
    <example>
    <title>Pseudo code pour illustrer l'usage d'une fonction</title>
     <programlisting role="php">
<![CDATA[
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Exemple de fonction.\n";
    return $retval;
}
?>
]]>
     </programlisting>
    </example>
   </para>
```

Procédures

- Documentation « technique »
  - Procédure d'installation
  - Procédure de mise à jour
- Doivent être
  - Connues de tous
  - Maintenues à jour !
- Quelle forme ?
  - Fiches wiki ?

Intégration continue

« Ça fait une semaine qu'on essaye de scotcher les composants ensemble ; y'a rien qui marche ;-(»

Intégration Continue ?

- Intégration fréquente
  - Du travail de tous les membres de l'équipe
  - Au moins une fois par jour
- Build automatique
  - Tests Automatisés
  - Vérifications
    - Normes de codage
  - Construction
    - Documentation
    - Archives

Pourquoi de l'IC ?

- Environnement stable
  - Savoir où en est le projet
    - Qualité / Quantité
    - Composants développés
  - Diminution des risques liés à l'Intégration
    - Devient un « non-événement »
  - Détection de régressions
    - Facilitée par les Tests Automatisés

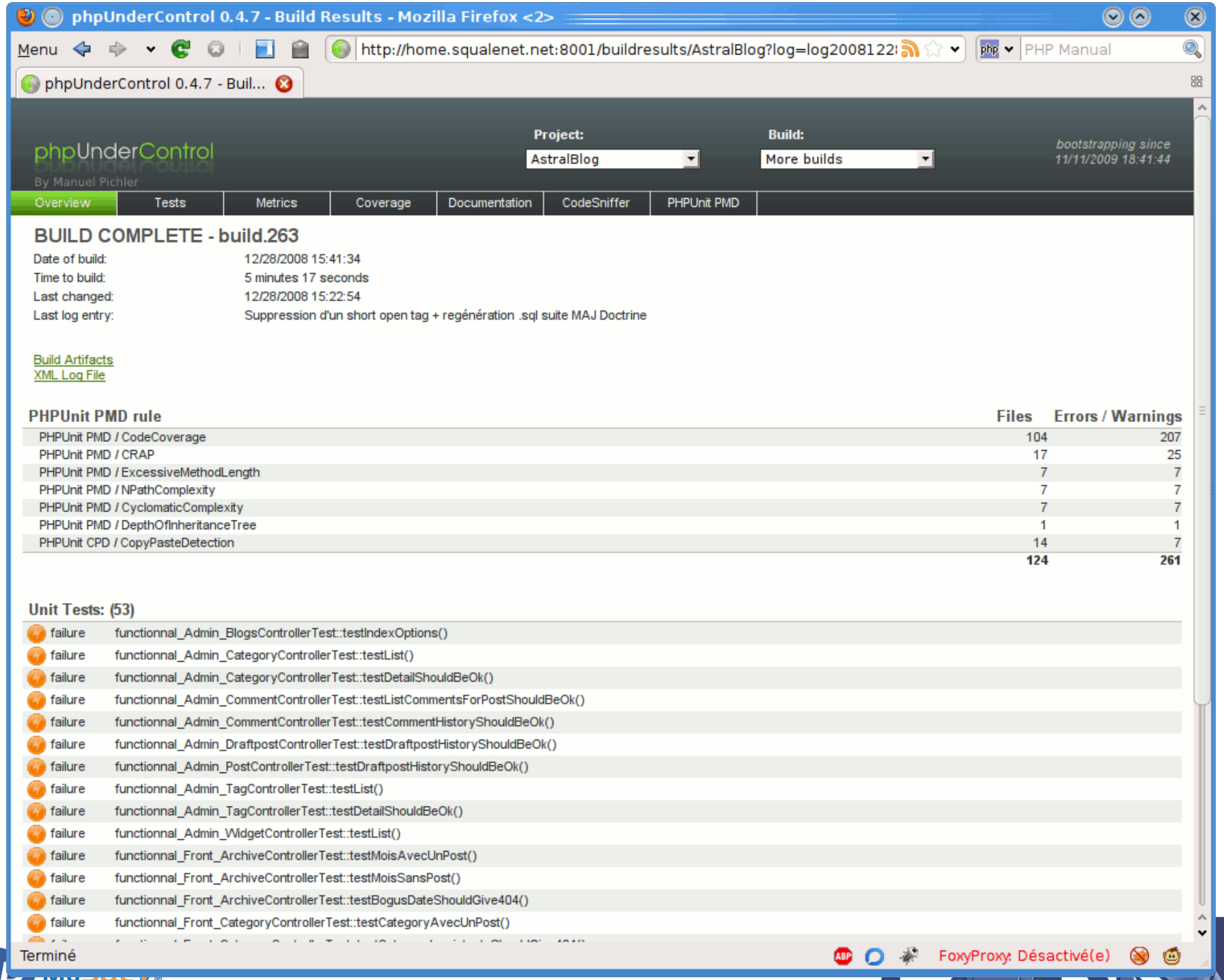
Plateforme d'IC

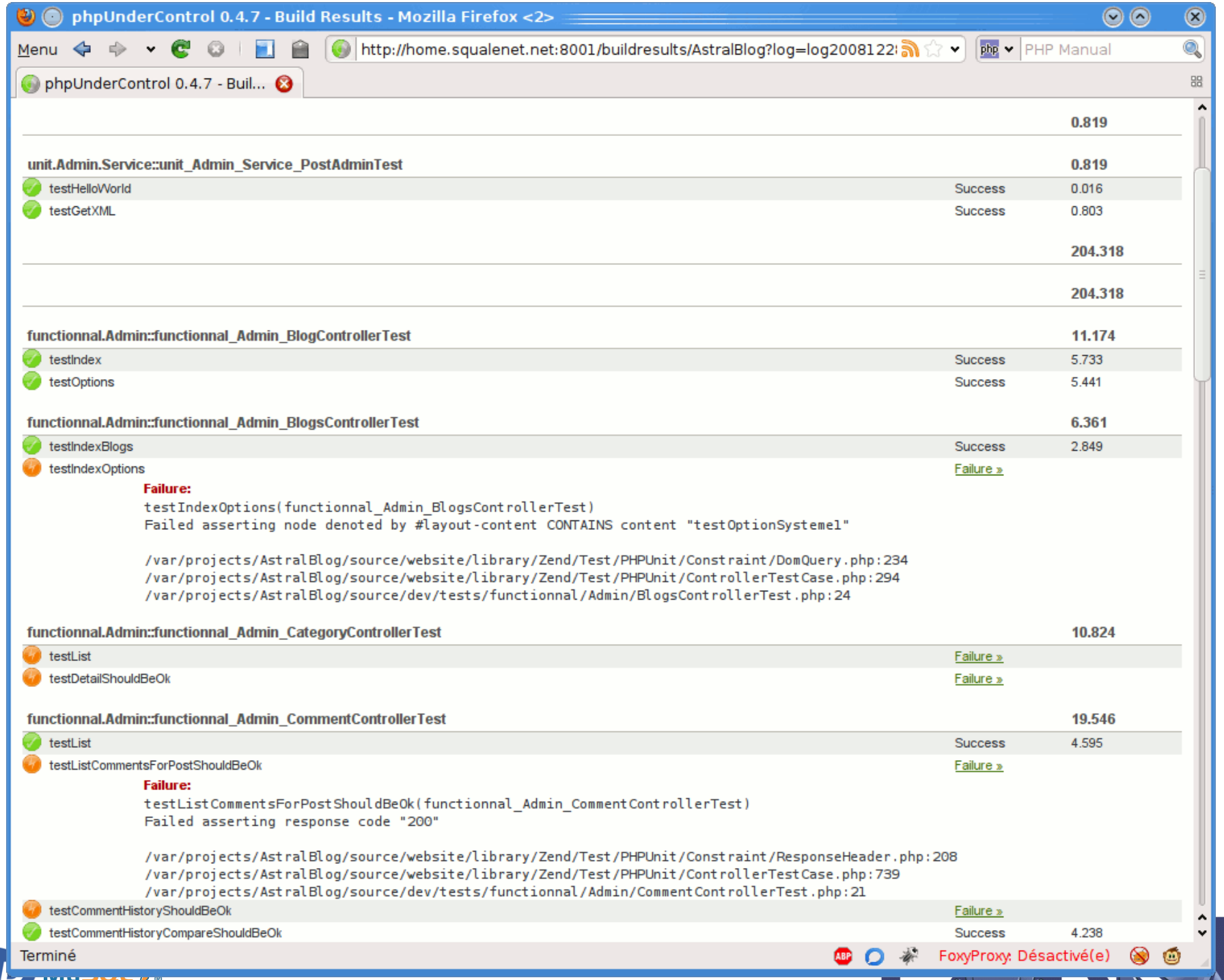
- Outil réalisant le build
  - Interface « user-friendly »
  - Historisation
    - De tous les builds
    - Y compris documentation, rapports, ...
  - Reporting
    - Via l'interface
    - Par mails
- Pas besoin de « tout » mettre en place !

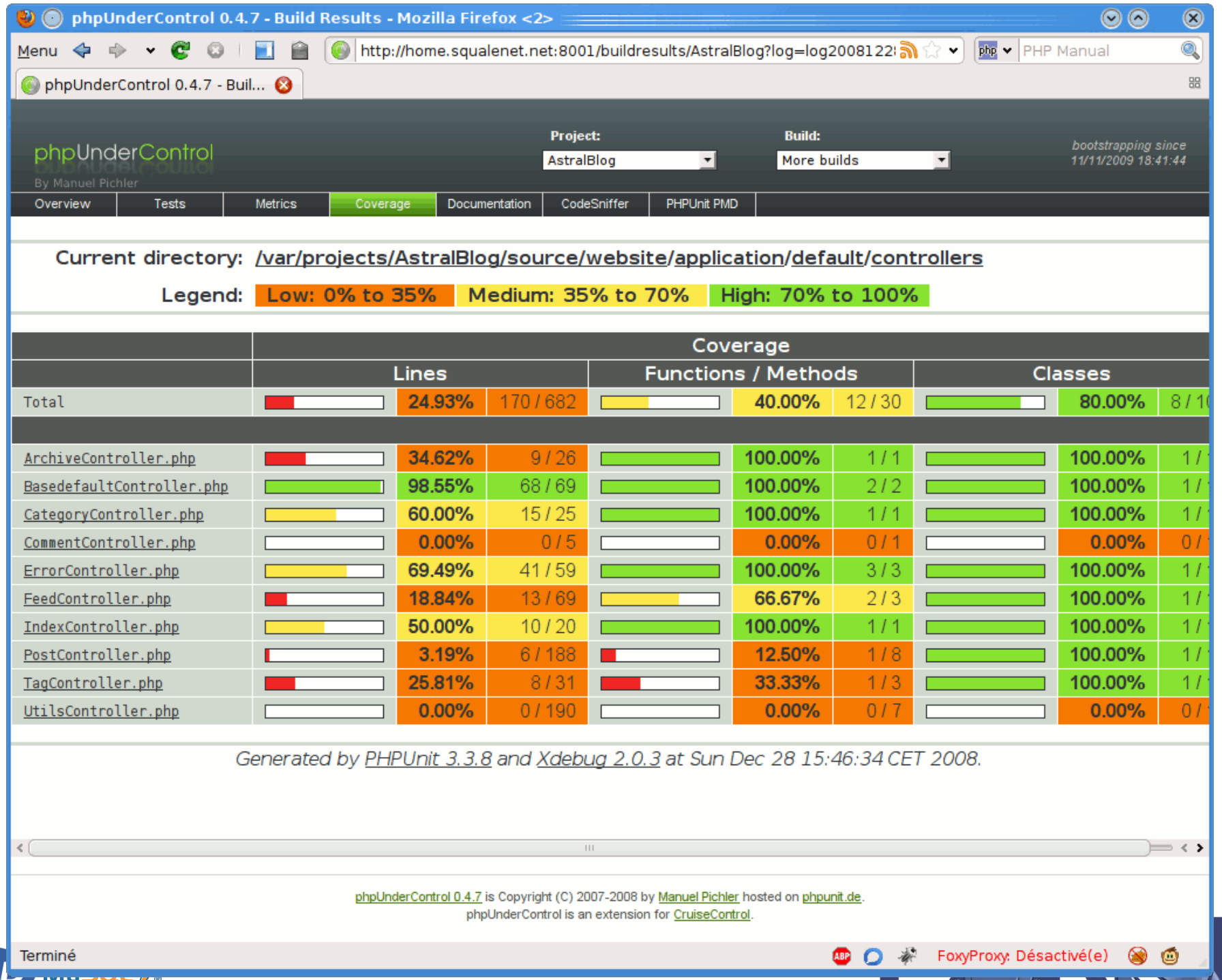
phpUnderControl

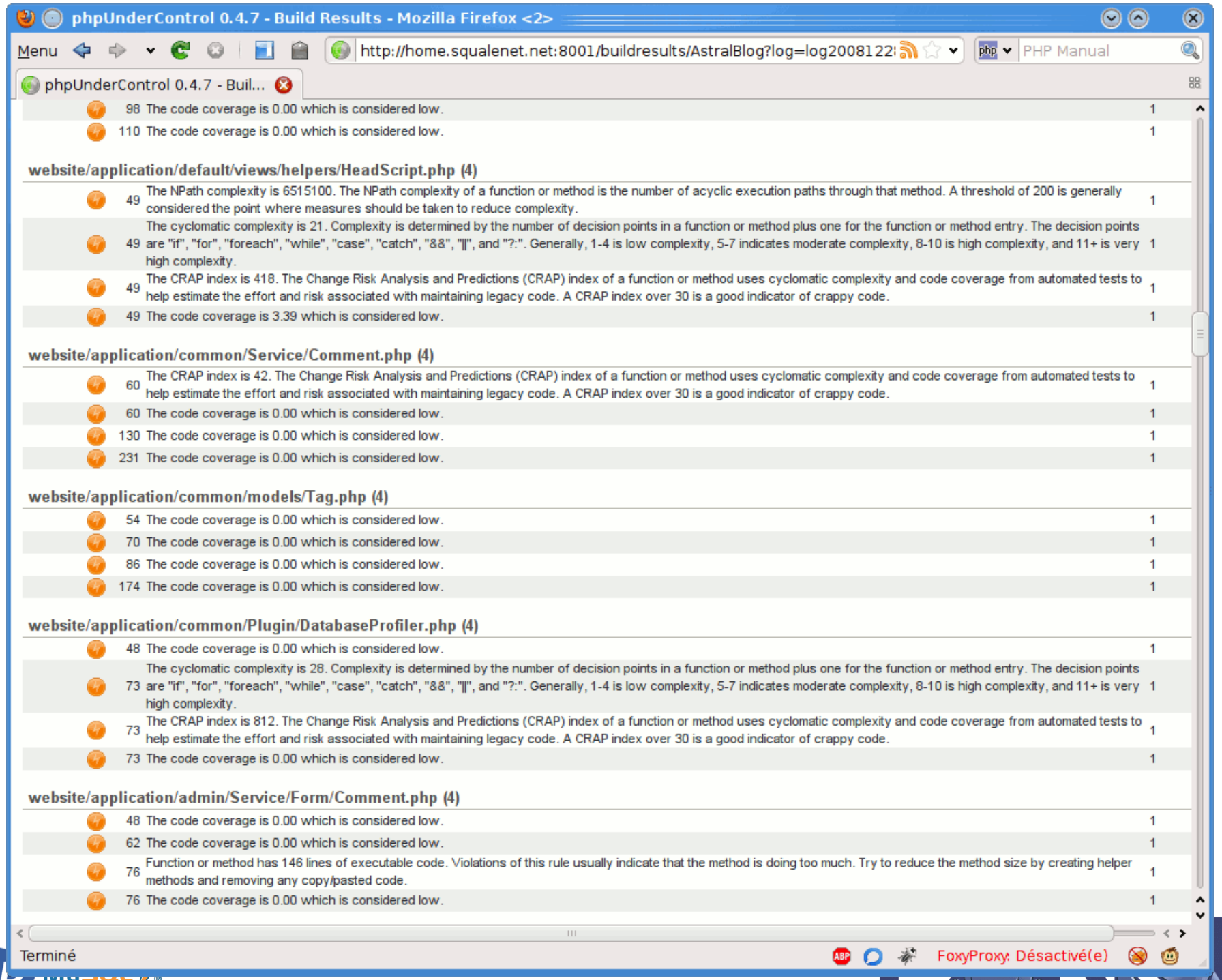
- phpUnderControl
  - http://www.phpundercontrol.org/
  - Basé sur CruiseControl / Ant
  - Intégration des outils communs en PHP
    - SVN
    - PHPUnit + Xdebug
    - PhpDocumentor
    - PHP_CodeSniffer

afup MySQL User Group


phpUnderControl 0.4.7 - Build Results - Mozilla Firefox <2>
Menu
http://home.squalenet.net:8001/buildresults/AstralBlog?log=log2008122
PHP Manual
phpUnderControl 0.4.7 - Buil...
phpUnderControl
By Manuel Pichler
Project:
AstralBlog
Build:
More builds
bootstrapping since
11/11/2009 18:41:44
Overview
Tests
Metrics
Coverage
Documentation
CodeSniffer
PHPUnit PMD
BUILD COMPLETE - build.263
Date of build: 12/28/2008 15:41:34
Time to build: 5 minutes 17 seconds
Last changed: 12/28/2008 15:22:54
Last log entry: Suppression d'un short open tag + regénération .sql suite MAJ Doctrine
Build Artifacts
XML Log File
PHPUnit PMD rule Files Errors / Warnings
PHPUnit PMD / CodeCoverage 104 207
PHPUnit PMD / CRAP 17 25
PHPUnit PMD / ExcessiveMethodLength 7 7
PHPUnit PMD / NPathComplexity 7 7
PHPUnit PMD / CyclomaticComplexity 7 7
PHPUnit PMD / DepthOfInheritanceTree 1 1
PHPUnit CPD / CopyPasteDetection 14 7
124 261
Unit Tests: (53)
failure functionnal_Admin_BlogsControllerTest::testIndexOptions()
failure functionnal_Admin_CategoryControllerTest::testList()
failure functionnal_Admin_CategoryControllerTest::testDetailShouldBeOk()
failure functionnal_Admin_CommentControllerTest::testListCommentsForPostShouldBeOk()
failure functionnal_Admin_CommentControllerTest::testCommentHistoryShouldBeOk()
failure functionnal_Admin_DraftpostControllerTest::testDraftpostHistoryShouldBeOk()
failure functionnal_Admin_PostControllerTest::testDraftpostHistoryShouldBeOk()
failure functionnal_Admin_TagControllerTest::testList()
failure functionnal_Admin_TagControllerTest::testDetailShouldBeOk()
failure functionnal_Admin_WidgetControllerTest::testList()
failure functionnal_Front_ArchiveControllerTest::testMoisAvecUnPost()
failure functionnal_Front_ArchiveControllerTest::testMoisSansPost()
failure functionnal_Front_ArchiveControllerTest::testBogusDateShouldGive404()
failure functionnal_Front_CategoryControllerTest::testCategoryAvecUnPost()
Terminé
FoxyProxy: Désactivé(e)


phpUnderControl 0.4.7 - Build Results - Mozilla Firefox <2>
Menu
http://home.squalenet.net:8001/buildresults/AstralBlog?log=log2008122
PHP Manual
phpUnderControl 0.4.7 - Buil...
0.819
unit.Admin.Service::unit_Admin_Service_PostAdminTest 0.819
testHelloWorld Success 0.016
testGetXML Success 0.803
204.318
204.318
functionnal.Admin::functionnal_Admin_BlogControllerTest 11.174
testIndex Success 5.733
testOptions Success 5.441
functionnal.Admin::functionnal_Admin_BlogsControllerTest 6.361
testIndexBlogs Success 2.849
testIndexOptions Failure »
Failure:
testIndexOptions(functionnal_Admin_BlogsControllerTest)
Failed asserting node denoted by #layout-content CONTAINS content "testOptionSysteme1"
/var/projects/AstralBlog/source/website/library/Zend/Test/PHPUnit/Constraint/DomQuery.php:234
/var/projects/AstralBlog/source/website/library/Zend/Test/PHPUnit/ControllerTestCase.php:294
/var/projects/AstralBlog/source/dev/tests/functionnal/Admin/BlogsControllerTest.php:24
functionnal.Admin::functionnal_Admin_CategoryControllerTest 10.824
testList Failure »
testDetailShouldBeOk Failure »
functionnal.Admin::functionnal_Admin_CommentControllerTest 19.546
testList Success 4.595
testListCommentsForPostShouldBeOk Failure »
Failure:
testListCommentsForPostShouldBeOk(functionnal_Admin_CommentControllerTest)
Failed asserting response code "200"
/var/projects/AstralBlog/source/website/library/Zend/Test/PHPUnit/Constraint/ResponseHeader.php:208
/var/projects/AstralBlog/source/website/library/Zend/Test/PHPUnit/ControllerTestCase.php:739
/var/projects/AstralBlog/source/dev/tests/functionnal/Admin/CommentControllerTest.php:21
testCommentHistoryShouldBeOk Failure »
testCommentHistoryCompareShouldBeOk Success 4.238
Terminé
FoxyProxy: Désactivé(e)


phpUnderControl 0.4.7 - Build Results - Mozilla Firefox <2>
Menu
http://home.squalenet.net:8001/buildresults/AstralBlog?log=log2008122
PHP Manual
phpUnderControl 0.4.7 - Buil...
phpUnderControl
By Manuel Pichler
Project:
AstralBlog
Build:
More builds
bootstrapping since
11/11/2009 18:41:44
Overview
Tests
Metrics
Coverage
Documentation
CodeSniffer
PHPUnit PMD
Current directory: /var/projects/AstralBlog/source/website/application/default/controllers
Legend: Low: 0% to 35% Medium: 35% to 70% High: 70% to 100%
Coverage
Lines
Functions / Methods
Classes
Total 24.93% 170 / 682 40.00% 12 / 30 80.00% 8 / 10
ArchiveController.php 34.62% 9 / 26 100.00% 1 / 1 100.00% 1 / 1
BasedefaultController.php 98.55% 68 / 69 100.00% 2 / 2 100.00% 1 / 1
CategoryController.php 60.00% 15 / 25 100.00% 1 / 1 100.00% 1 / 1
CommentController.php 0.00% 0 / 5 0.00% 0 / 1 0.00% 0 / 1
ErrorController.php 69.49% 41 / 59 100.00% 3 / 3 100.00% 1 / 1
FeedController.php 18.84% 13 / 69 66.67% 2 / 3 100.00% 1 / 1
IndexController.php 50.00% 10 / 20 100.00% 1 / 1 100.00% 1 / 1
PostController.php 3.19% 6 / 188 12.50% 1 / 8 100.00% 1 / 1
TagController.php 25.81% 8 / 31 33.33% 1 / 3 100.00% 1 / 1
UtilsController.php 0.00% 0 / 190 0.00% 0 / 7 0.00% 0 / 1
Generated by PHPUnit 3.3.8 and Xdebug 2.0.3 at Sun Dec 28 15:46:34 CET 2008.
phpUnderControl 0.4.7 is Copyright (C) 2007-2008 by Manuel Pichler hosted on phpunit.de.
phpUnderControl is an extension for CruiseControl.
Terminé
FoxyProxy: Désactivé(e)


phpUnderControl 0.4.7 - Build Results - Mozilla Firefox <2>
Menu
http://home.squalenet.net:8001/buildresults/AstralBlog?log=log2008122
PHP Manual
phpUnderControl 0.4.7 - Buil...
98 The code coverage is 0.00 which is considered low. 1
110 The code coverage is 0.00 which is considered low. 1
website/application/default/views/helpers/HeadScript.php (4)
49 The NPath complexity is 6515100. The NPath complexity of a function or method is the number of acyclic execution paths through that method. A threshold of 200 is generally considered the point where measures should be taken to reduce complexity. 1
49 The cyclomatic complexity is 21. Complexity is determined by the number of decision points in a function or method plus one for the function or method entry. The decision points are "if", "for", "foreach", "while", "case", "catch", "&&", "||", and "?:". Generally, 1-4 is low complexity, 5-7 indicates moderate complexity, 8-10 is high complexity, and 11+ is very high complexity. 1
49 The CRAP index is 418. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
49 The code coverage is 3.39 which is considered low. 1
website/application/common/Service/Comment.php (4)
60 The CRAP index is 42. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
60 The code coverage is 0.00 which is considered low. 1
130 The code coverage is 0.00 which is considered low. 1
231 The code coverage is 0.00 which is considered low. 1
website/application/common/models/Tag.php (4)
54 The code coverage is 0.00 which is considered low. 1
70 The code coverage is 0.00 which is considered low. 1
86 The code coverage is 0.00 which is considered low. 1
174 The code coverage is 0.00 which is considered low. 1
website/application/common/Plugin/DatabaseProfiler.php (4)
48 The code coverage is 0.00 which is considered low. 1
73 The cyclomatic complexity is 28. Complexity is determined by the number of decision points in a function or method plus one for the function or method entry. The decision points are "if", "for", "foreach", "while", "case", "catch", "&&", "||", and "?:". Generally, 1-4 is low complexity, 5-7 indicates moderate complexity, 8-10 is high complexity, and 11+ is very high complexity. 1
73 The CRAP index is 812. The Change Risk Analysis and Predictions (CRAP) index of a function or method uses cyclomatic complexity and code coverage from automated tests to help estimate the effort and risk associated with maintaining legacy code. A CRAP index over 30 is a good indicator of crappy code. 1
73 The code coverage is 0.00 which is considered low. 1
website/application/admin/Service/Form/Comment.php (4)
48 The code coverage is 0.00 which is considered low. 1
62 The code coverage is 0.00 which is considered low. 1
76 Function or method has 146 lines of executable code. Violations of this rule usually indicate that the method is doing too much. Try to reduce the method size by creating helper methods and removing any copy/pasted code. 1
76 The code coverage is 0.00 which is considered low. 1
Terminé
FoxyProxy: Désactivé(e)

PIC chez SQLI

- Un des projets présentés au Forum cette année
  - Allez faire un tour sur le stand !
- Basée sur phpUnderControl
  - Avec plus de fonctionnalités
  - Intégration des rapports de Zend Platform
- Utilisée sur les projets au Forfait à Paris
  - Et composants utilisés sur certains projets dans d'autres agences

afup MySQL User Group

Autres solutions ?

- Liberté de choix ;-)
- Xinc
  - http://code.google.com/p/xinc/
- Hudson
  - http://hudson-ci.org/
- Sismo
  - http://ci.symfony-project.org/

Déploiement

« Pierre est en vacances ;
on ne peut pas livrer, on ne sait pas
comment il fait d'habitude »

afup MySQL User Group

Avant de commencer

- Suivez une procédure de déploiement clairement définie
- Ne bossez pas directement en production !
- Utilisez un serveur de tests, proche de la production
  - Mêmes versions de logiciels,
  - Même OS.
- Gardez trace de ce que vous livrez
  - Contrôle de sources : « tag »

Exemple de processus

- Développements
- Tests interne sur serveur d'intégration
- Tests + Validation interne
- Tag « production »
- Déploiement sur serveur de tests client
- Tests + Validation client
- Déploiement sur serveur de production
- Tests

afup MySQL User Group

Automatiser le déploiement

- Déploiement automatique
  - Moins de risque d'erreur de manipulation
  - Genre « oubli d'une étape »
- Réalisable par n'importe qui
  - Évite d'être bloqué en cas d'absence
- Documenté !
- Tests
  - Automatiques aussi !
  - Simples, rapides – mais nécessaires

Outils ?

- Simples shell-scripts
- Ant / phing
- Package pear
- .deb / .rpm
- Lien symbolique
  - Qui pointe sur la version actuelle
  - Objectif : retour arrière facilité
  - Ne pas écraser l'ancienne version

Et après ?

- Application en production
  - Mais ce n'est pas fini !
- Monitorer
  - Les serveurs
  - Le fonctionnement de l'application
  - Les logs

Encore un peu ?

afup MySQL User Group

« WTF ??? »

Veille Techno / curiosité

- Nous ne sommes pas seuls au monde
- Suivre l'actualité
  - De PHP
  - Des composants / logiciels
- Internet = source de savoir
  - Flux RSS
  - Twitter
  - Événements
- Participez !

Formation / Explications

- Plein de bonnes pratique
  - C'est bien !
- Expliquer comment / pourquoi
  - C'est mieux !
- Malgré « *la crise* »
  - Investissement
  - Qualité des projets
  - Gain à moyen / long terme

afup MySQL User Group

A Vous !

- Qualité
  - processus continu
- Mise en place : progressive !
  - Quelques bases
  - Puis d'autres
  - Et encore d'autres
- Laisser le temps
  - De se familiariser
  - D'apprécier

afup MySQL User Group

Merci !

Pascal MARTIN – SQLI
http://blog.pascal-martin.fr
contact@pascal-martin.fr

sqli GROUP

afup MySQL User Group