CODEWORKS 2010

Five tools every PHP programmer should know (and love)

SEATTLE • PORTLAND • AUSTIN • BALTIMORE • ORLANDO

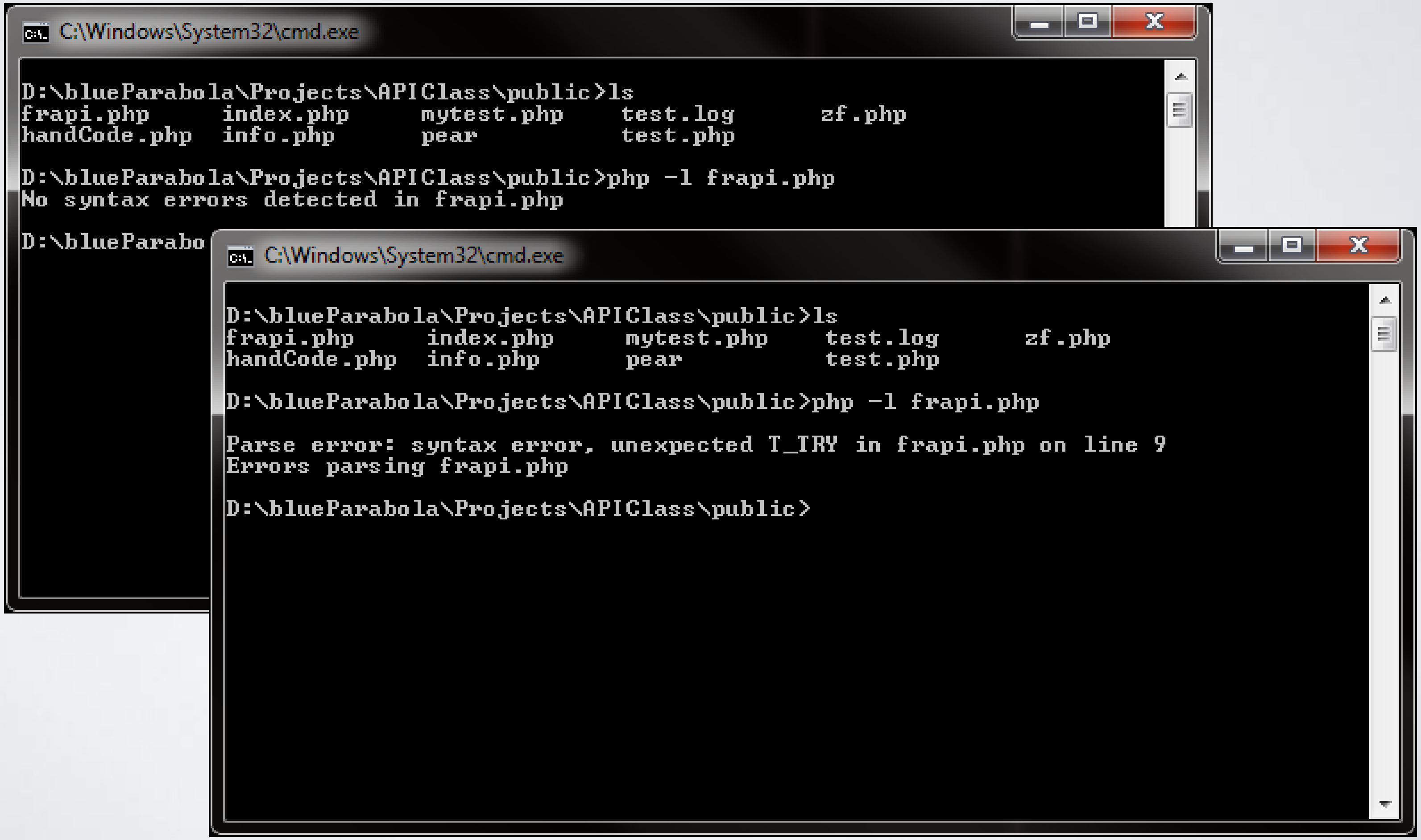
Cal Evans

cal@blueparabola.com

0:php lint

- Author/Project Lead:
  PHP Core Team
- Website:
  http://www.php.net/manual/en/features.commandline.options.php
- Installation:
  Included in PHP
- MYLALH Scale:
  0 (Actually Helpful)

0:php lint



```
D:\blueParabola\Projects\APIClass\public>ls
frapi.php      index.php      mytest.php     test.log       zf.php
handCode.php   info.php       pear           test.php

D:\blueParabola\Projects\APIClass\public>php -l frapi.php
No syntax errors detected in frapi.php

D:\blueParabo
```

```
D:\blueParabola\Projects\APIClass\public>ls
frapi.php      index.php      mytest.php     test.log       zf.php
handCode.php   info.php       pear           test.php

D:\blueParabola\Projects\APIClass\public>php -l frapi.php

Parse error: syntax error, unexpected T_TRY in frapi.php on line 9
Errors parsing frapi.php

D:\blueParabola\Projects\APIClass\public>
```

Overview

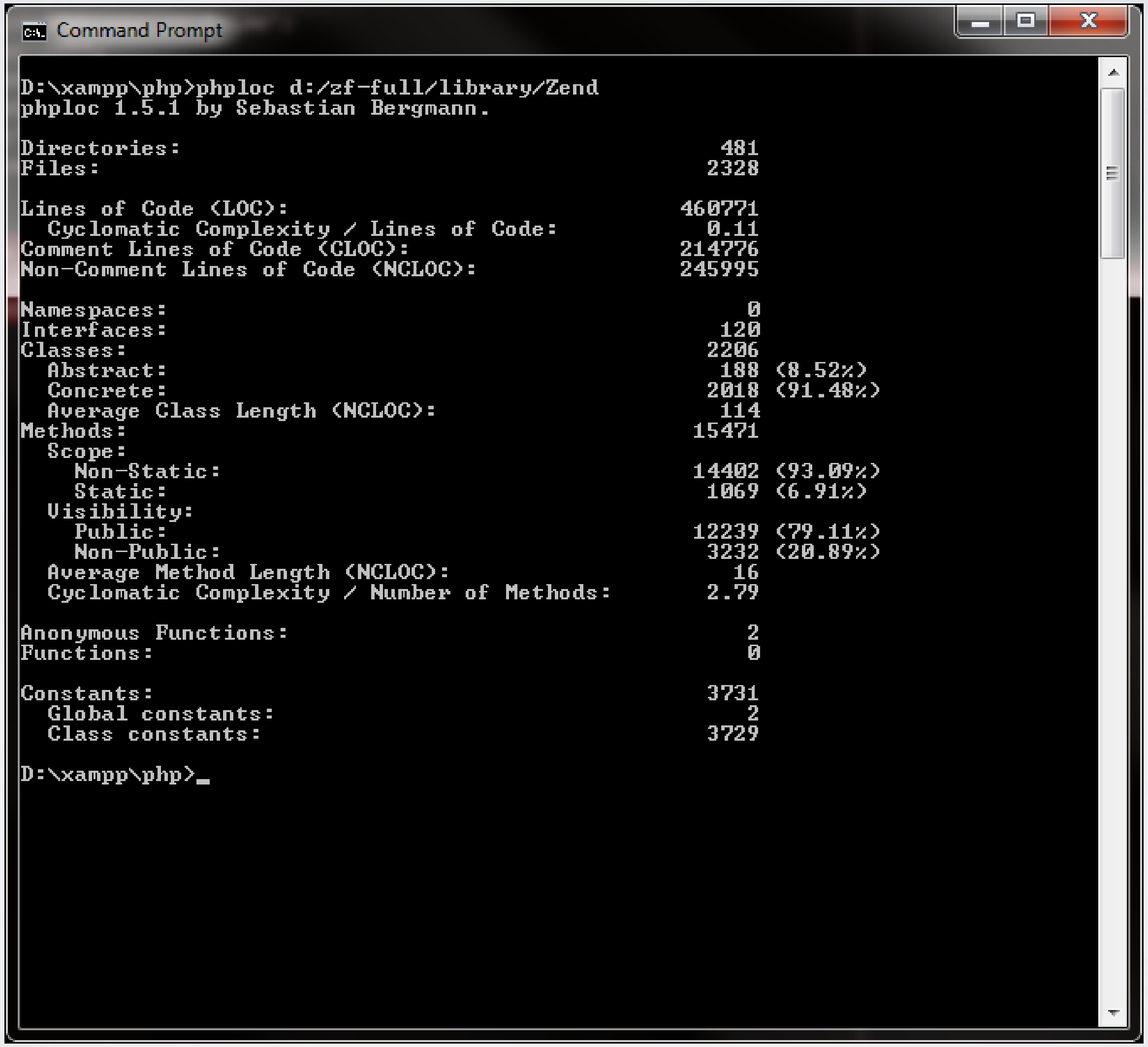
- Built into PHP
- Not true lint, just a syntax checker
- Life saver if you are in a hurry

- Author/Project Lead:
  Sebastian Bergman
- Website:
  http://github.com/sebastianbergmann/phploc
- Installation:
  PEAR
- MYLALH Scale:
  1 (Not painful)

phploc is a tool for quickly measuring the size of a PHP project.

The goal of **phploc** is not to replace more sophisticated tools such as phpcs, pdepend, or phpmd, but rather to provide an alternative to them when you just need to get a quick understanding of a project's size

1:phploc



```
D:\xampp\php>phploc d:/zf-full/library/Zend
phploc 1.5.1 by Sebastian Bergmann.

Directories:                                              481
Files:                                                   2328

Lines of Code (LOC):                                   460771
  Cyclomatic Complexity / Lines of Code:                 0.11
Comment Lines of Code (CLOC):                          214776
Non-Comment Lines of Code (NCLOC):                     245995

Namespaces:                                                 0
Interfaces:                                               120
Classes:                                                 2206
  Abstract:                                               188 (8.52%)
  Concrete:                                              2018 (91.48%)
  Average Class Length (NCLOC):                           114
Methods:                                                15471
  Scope:
    Non-Static:                                         14402 (93.09%)
    Static:                                              1069 (6.91%)
  Visibility:
    Public:                                             12239 (79.11%)
    Non-Public:                                          3232 (20.89%)
  Average Method Length (NCLOC):                           16
  Cyclomatic Complexity / Number of Methods:             2.79

Anonymous Functions:                                        2
Functions:                                                  0

Constants:                                               3731
  Global constants:                                         2
  Class constants:                                       3729

D:\xampp\php>_
```

- Measures
  - Raw Lines of Code (LOC)
  - Lines of comments (CLOC)
  - Non-Comment Lines of Code (NCLOC)
  - Namespaces
  - Classes
  - Methods

Cyclomatic Complexity

“Cyclomatic complexity (or conditional complexity) is a software metric (measurement). It was developed by Thomas J. McCabe, Sr. in 1976 and is used to indicate the complexity of a program. It directly measures the number of linearly independent paths through a program's source code.”

http://en.wikipedia.org/wiki/Cyclomatic_complexity

Overview

- Easy to use tool
- Provides quick overview of your poject’s size and complexity
- Great stats

2:phpdepend

- Author/Project Lead:
  Manuel Pichler
- Website:
  http://pdepend.org
- Installation:
  PEAR *
- MYLALH Scale:
  1 (Benign)

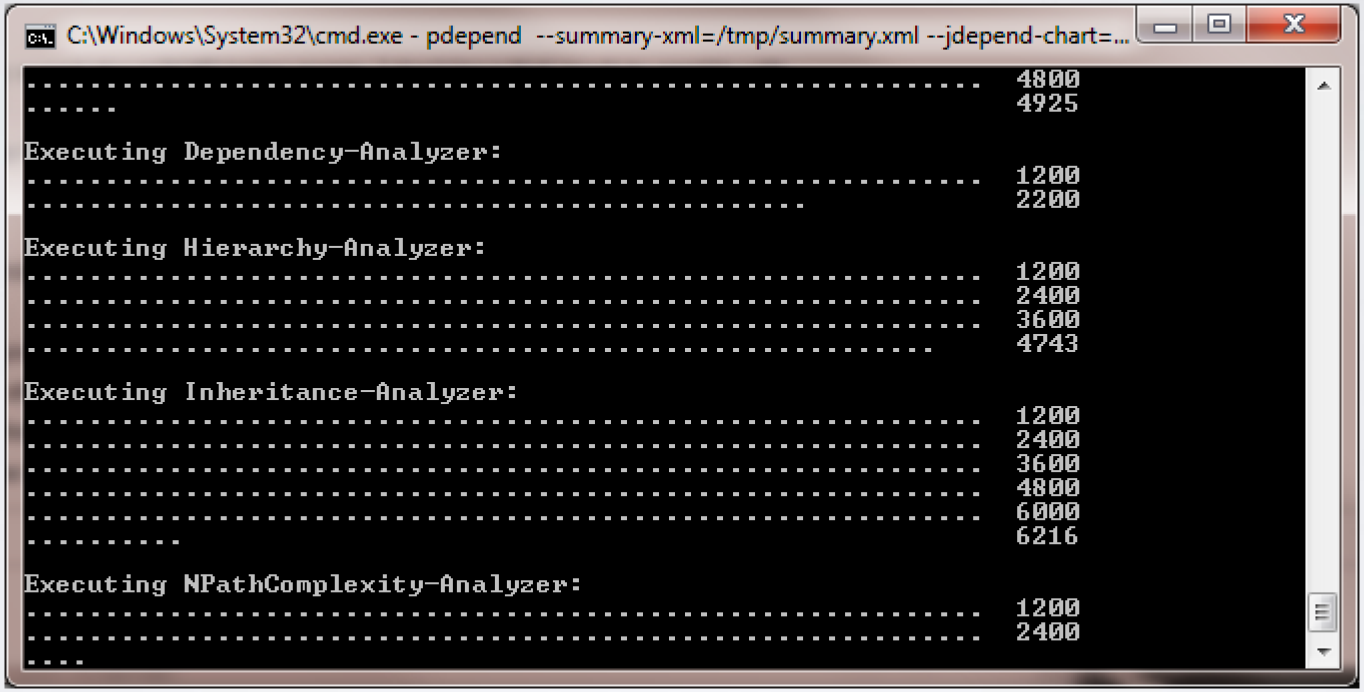
2:phpdepend

- PHP_Depend is a small program that performs static code analysis on a given source base.
  - PHP_Depend first takes the source code and parses it into an easily processable internal data structure called an AST (Abstract Syntax Tree)
  - Then it takes the generated AST and measures several values, the so called software metrics.
    - Cyclomatic Complexity
    - Npath Complexity
    - CodeRank
    - Lines of Code

Overview

- Duplicates some of phploc
- Gives a slightly different view of the code view
- Generates pretty graphics if you can get *&^% imageick installed.
- Because it is more complete, it is more complex and takes much longer to run.
- Great cron job tool to add some nice graphics and info to your project’s web page.
- Documentation is very incomplete. If you understand what it does and like it, then use it. Otherwise, it may not be the best tool for your project.

2:phpdepend

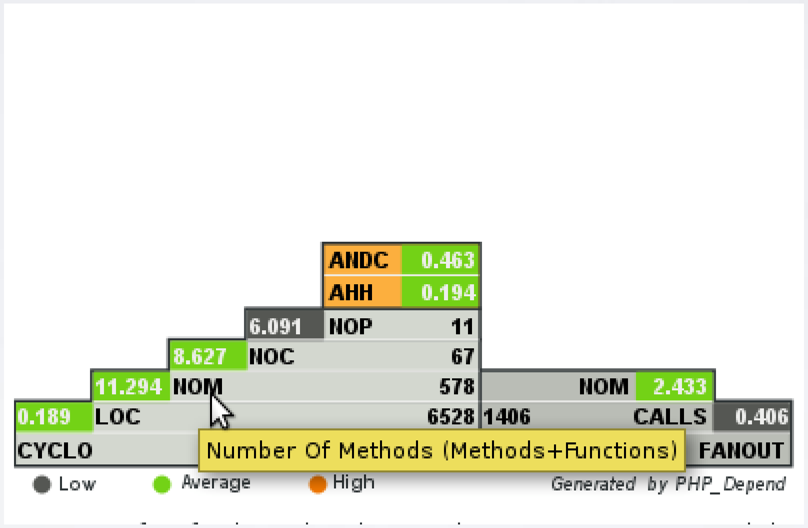


C:\Windows\System32\cmd.exe - pdepend --summary-xml=/tmp/summary.xml --jdepend-chart=...

```
............................................................    4800
......                                                          4925

Executing Dependency-Analyzer:
............................................................    1200
..................................................              2200

Executing Hierarchy-Analyzer:
............................................................    1200
............................................................    2400
............................................................    3600
..........................................................      4743

Executing Inheritance-Analyzer:
............................................................    1200
............................................................    2400
............................................................    3600
............................................................    4800
............................................................    6000
..........                                                      6216

Executing NPathComplexity-Analyzer:
............................................................    1200
............................................................    2400
....
```

CODEWORKS 2010

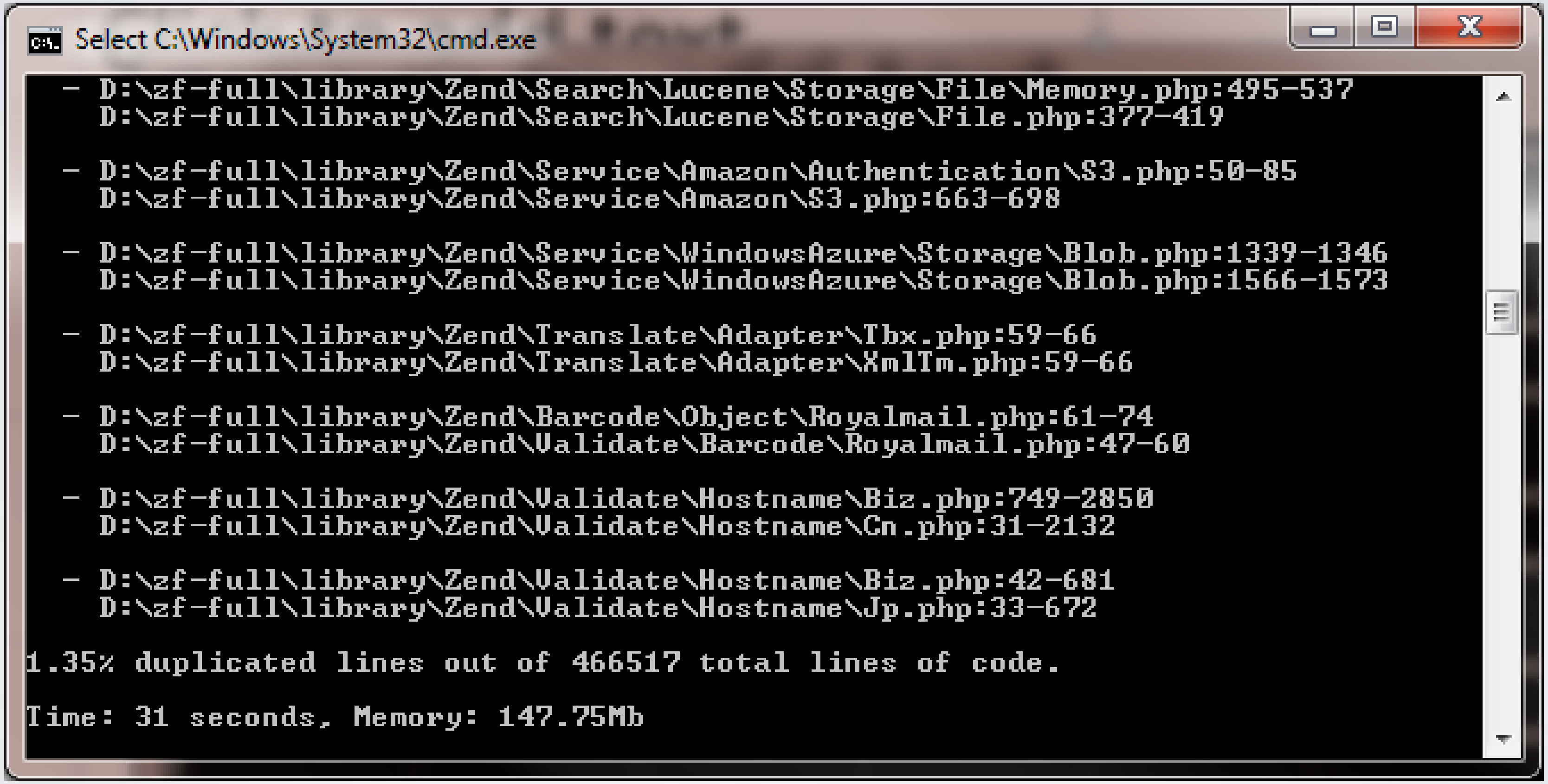
2:phpdepend



- Author/Project Lead:
  Sebastian Bergman
- Website:
  http://github.com/sebastianbergmann/phpcpd
- Installation:
  PEAR
- MYLALH Scale:
  2 (PEAR Hell)

The goal of phpcpd is not to replace more sophisticated tools such as phpcs, pdepend, or phpmd, but rather to provide an alternative to them when you just need to get a quick overview of **duplicated code** in a project. (Copy and Paste Detector)

3:phpcpd



```
  - D:\zf-full\library\Zend\Search\Lucene\Storage\File\Memory.php:495-537
    D:\zf-full\library\Zend\Search\Lucene\Storage\File.php:377-419

  - D:\zf-full\library\Zend\Service\Amazon\Authentication\S3.php:50-85
    D:\zf-full\library\Zend\Service\Amazon\S3.php:663-698

  - D:\zf-full\library\Zend\Service\WindowsAzure\Storage\Blob.php:1339-1346
    D:\zf-full\library\Zend\Service\WindowsAzure\Storage\Blob.php:1566-1573

  - D:\zf-full\library\Zend\Translate\Adapter\Tbx.php:59-66
    D:\zf-full\library\Zend\Translate\Adapter\XmlTm.php:59-66

  - D:\zf-full\library\Zend\Barcode\Object\Royalmail.php:61-74
    D:\zf-full\library\Zend\Validate\Barcode\Royalmail.php:47-60

  - D:\zf-full\library\Zend\Validate\Hostname\Biz.php:749-2850
    D:\zf-full\library\Zend\Validate\Hostname\Cn.php:31-2132

  - D:\zf-full\library\Zend\Validate\Hostname\Biz.php:42-681
    D:\zf-full\library\Zend\Validate\Hostname\Jp.php:33-672

1.35% duplicated lines out of 466517 total lines of code.

Time: 31 seconds, Memory: 147.75Mb
```

Overview

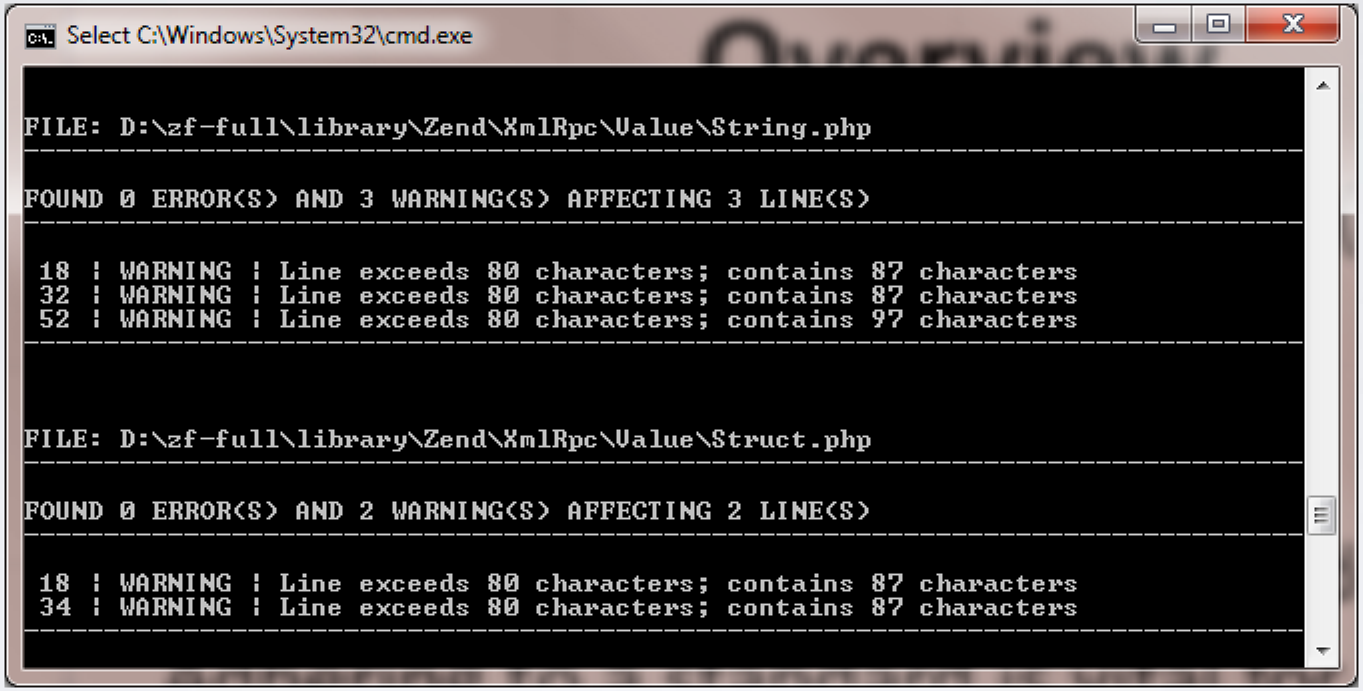
- Quick tool but one you won’t run often
- Memory hog
- Useful for team leads and project managers to keep an eye on things.

- Author/Project Lead:
  Greg Sherwood
- Website:
  http://pear.php.net/package/PHP_CodeSniffer
- Installation:
  PEAR
- MYLALH Scale:
  7 (Curly Brace Hell)

4:CodeSniffer

PHP_CodeSniffer is a PHP5 script that tokenises and “sniffs” PHP, JavaScript and CSS files to detect violations of a defined coding standard.

- 5 coding standards
  - PEAR
  - PHPCS
  - Squiz
  - WordPress
  - Zend
- You can create your own



```
FILE: D:\zf-full\library\Zend\XmlRpc\Value\String.php
--------------------------------------------------------------------------------
FOUND 0 ERROR(S) AND 3 WARNING(S) AFFECTING 3 LINE(S)
--------------------------------------------------------------------------------
 18 | WARNING | Line exceeds 80 characters; contains 87 characters
 32 | WARNING | Line exceeds 80 characters; contains 87 characters
 52 | WARNING | Line exceeds 80 characters; contains 97 characters
--------------------------------------------------------------------------------


FILE: D:\zf-full\library\Zend\XmlRpc\Value\Struct.php
--------------------------------------------------------------------------------
FOUND 0 ERROR(S) AND 2 WARNING(S) AFFECTING 2 LINE(S)
--------------------------------------------------------------------------------
 18 | WARNING | Line exceeds 80 characters; contains 87 characters
 34 | WARNING | Line exceeds 80 characters; contains 87 characters
--------------------------------------------------------------------------------
```

4:CodeSniffer

```
FILE: D:\zf-full\library\Zend\Amf\Parse\Resource\MysqlResult.php
--------------------------------------------------------------------------------
FOUND 8 ERROR(S) AFFECTING 7 LINE(S)
--------------------------------------------------------------------------------
 51 | ERROR | Opening brace should be on a new line
 54 | ERROR | Variable "fields_transform" is not in valid camel caps format
 55 | ERROR | Expected "for (...) {\n"; found "for(...) {\n"
 57 | ERROR | Expected "if (...) {\n"; found "if(...) {\n"
 58 | ERROR | Variable "fields_transform" is not in valid camel caps format
 62 | ERROR | Expected "while (...) {\n"; found "while(...) {\n"
 63 | ERROR | Expected "foreach (...) {\n"; found "foreach(...) {\n"
 63 | ERROR | Variable "fields_transform" is not in valid camel caps format
--------------------------------------------------------------------------------
```

4:CodeSniffer

```
public function getIncludedSniffs()
 {
       return array(
 'Generic/Sniffs/Functions/OpeningFunctionBraceBsdAllmanSniff.php
 ',
 'Generic/Sniffs/PHP/DisallowShortOpenTagSniff.php',
 'Generic/Sniffs/WhiteSpace/DisallowTabIndentSniff.php',
 'PEAR/Sniffs/Classes/ClassDeclarationSniff.php',
 'PEAR/Sniffs/ControlStructures/ControlSignatureSniff.php',
 'PEAR/Sniffs/Files/LineEndingsSniff.php',
 'PEAR/Sniffs/Functions/FunctionCallArgumentSpacingSniff.php',
 'PEAR/Sniffs/Functions/FunctionCallSignatureSniff.php',
 'PEAR/Sniffs/Functions/ValidDefaultValueSniff.php',
 'PEAR/Sniffs/WhiteSpace/ScopeClosingBraceSniff.php',
 'Squiz/Sniffs/Functions/GlobalFunctionSniff.php',
 );
 }//end getIncludedSniffs()
```

Overview

- Powerful tool in the right hands. (Instrument of torture in the wrong hands)
- Run early and run often
- The output is not just so your code is pretty. Adhering to a standard is vital for any project
- You can create your own standards file.

5:phing

- Author/Project Lead:
  Hans Lellelid, Michiel Rook/Community
- Website:
  http://phing.info/trac/wiki/Development/Contributors
- Installation:
  PEAR
- MYLALH Scale:
  1 (XML Quagmire)

Phing is a project build system based on Apache ant [ant]. You can do anything with Phing that you could do with a traditional build system like Gnu make [gnumake], and Phing's use of simple XML build files and extensible PHP "task" classes make it an easy-to-use and highly flexible build framework.

Snippet from a sample build.xml

```
<!-- ======================================= -->
<!-- Target: build                           -->
<!-- ======================================= -->
<target name="build" depends="prepare">
    <echo msg="Copying files to build directory..." />

    <echo msg="Copying ./about.php to ./build directory..." />
    <copy file="./about.php" tofile="./build/about.php" />

    <echo msg="Copying ./browsers.php to ./build directory..." />
    <copy file="./browsers.php" tofile="./build/browsers.php" />

    <echo msg="Copying ./contact.php to ./build directory..." />
    <copy file="./contact.php" tofile="./build/contact.php" />
</target>
```

Secret Sauce

`Phing/Phing.php:1087(ish)`

```
$dataDir = dirname(__FILE__).
           DIRECTORY_SEPARATOR .
           '..\' .
           DIRECTORY_SEPARATOR .
           'data';
```

Extending Phing: A new task

```
<?php

require_once "phing/Task.php";

class CheckTwitter extends Task {

    /**
     * The message passed in the buildfile.
     */
    protected $hashtag = null;
    protected $url = 'http://search.twitter.com/search.json';

    /**
     * The setter for the attribute "message"
     */
    public function setHashtag($str) {
        $this->hashtag = $str;
    }
```

Extending Phing: A new task

```
  public function init() {
      // nothing to do here
    }

    /**
     * The main entry point method.
     */
    public function main() {
        $json = file_get_contents($this->url .
               '?q='.
               urlencode('#'.$this->hashtag));

        $myFeed = json_decode($json);

        foreach ($myFeed->results as $stuff=>$tweet) {
            echo $tweet->text."\n---\n";
        }
    }

}
```

Overview

- PHP doesn’t really need make or ant, however phing is useful for a lot of repetitive tasks like packaging and distribution.
- Surprisingly easy to use tool.
- Somewhat difficult to get working on Windows

CWX
CODEWORKS 2010

Bringing it all together

- Small tools make big tools
- It’s fun when everything works together.

Who Am I?

Cal Evans

Chief Marketing Officer, Blue Parabola

http://phparch.com

http://blog.calevans.com

@calevans

cal@blueparabola.com

cal@calevans.com