Achieving High-Quality Code Standards

Patrick Schwisow

Madison PHP

June 25, 2015

Why do we need standards?

```
function convert($data){
$l = strlen($data)-2;
$str = substr($data,1,$l);
$a = explode(",",$str);
$a2 = array();
for ($i=0;$i<sizeof($a);$i++) {
$l2 = strlen($a[$i])-2;
$a2[] = substr($a[$i],1,$l2);
}
return $a2;
}
```

Why do we need standards?

```
function jsonStringsToArray($jsonData)
{
    // remove first and last []
    $length = strlen($jsonData) - 2;
    $jsonString = substr($jsonData, 1, $length);

    // tokenize, using , as a divider
    $newArray = explode(",", $jsonString);

    $lastArray = array();
    for ($i = 0; $i < sizeof($newArray); $i++) {
        $thisLength = strlen($newArray[$i]) - 2;
        $lastArray[] = substr($newArray[$i], 1, $thisLength);
    }
    return $lastArray;
}
```

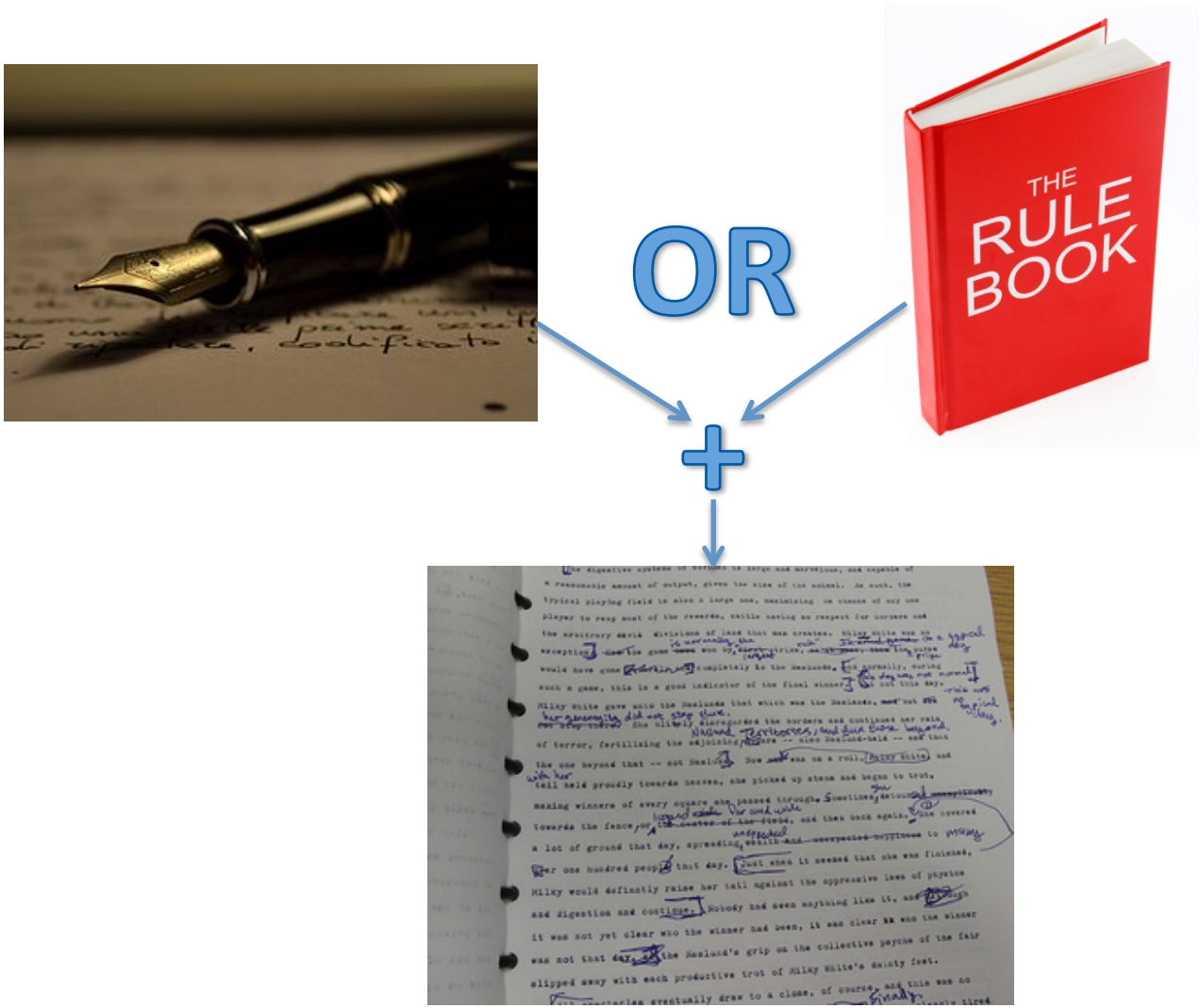
Why do we need standards?

```
/**
 * Converts JSON data string to PHP array.
 *
 * Assumes this format: ["Some text","Even more text"]
 *
 * @param string $jsonData
 * @return array
 */
function jsonStringsToArray($jsonData)
{
    ...
}
```

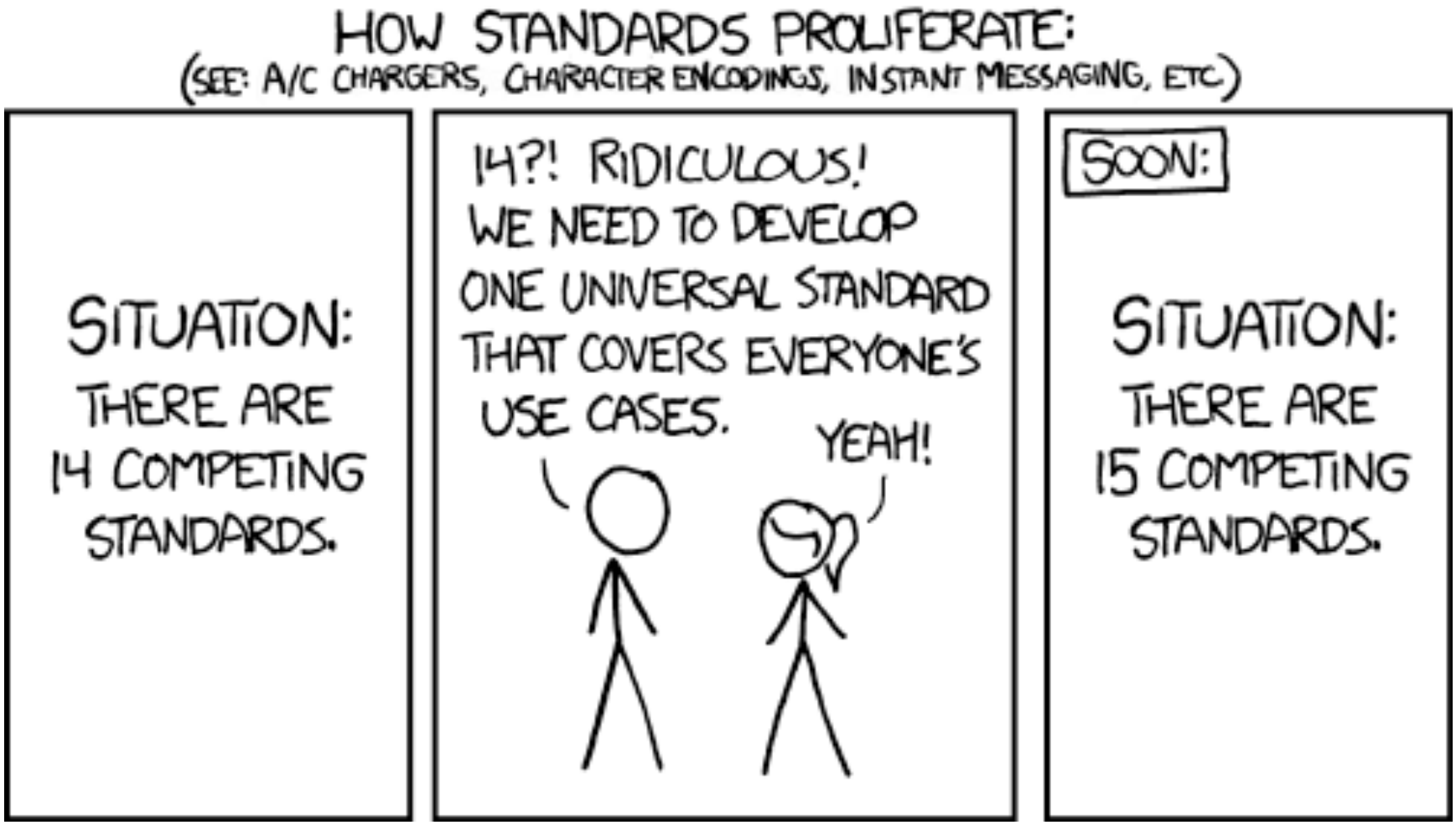
What areas should standards address?

- Naming practices
- Code formatting
- File structure
- Autoloading
- Application file structure
- Separation of concerns
- Consistent architectural patterns
- Documentation standards
- Testing standards
- Version control standards
- Process standards

How do we define our standards?



Published Standards



Published Standards

- PEAR
- Zend Framework (v1)
- symfony (v1)
- WordPress
- Drupal
- PHP-FIG (PSRs)

PHP-FIG Standards

- ~~PSR-0 Autoloading Standard~~ DEPRECATED
- PSR-1 Basic Coding Standard
- PSR-2 Coding Style Guide
- PSR-4 Improved Autoloading

PSR-0/-4 Autoloading

- Filenames end in .php
- One class per file
- Classes must be under a vendor namespace (at least)
- Classname (after namespace) must match filename
- `\Symfony\Core\Request` ➔ /path/to/project/lib/vendor/Symfony/Core/Request.php
- `\Zend_Controller_Front` ➔ /path/to/project/lib/vendor/Zend/Controller/Front.php (PSR-0 only)
- `\Aura\Web\Response\Status` ➔ /path/to/aura-web/src/Response/Status.php (PSR-4 with prefix of “Aura\Web”)

PSR-1 Basic Coding Standard

- Must follow PSR-0 or PSR-4 autoloading requirements
- Side effects: “A file SHOULD declare new symbols (classes, functions, constants, etc.) and cause no other side effects, or it SHOULD execute logic with side effects, but SHOULD NOT do both.”
- Namespace and class names: `StudlyCaps`
- Class constants: `ALL_CAPS`
- Method names: `camelCase()`

PSR-2 Coding Style Guide

- Must follow PSR-1
- Omit closing ?> tag from PHP-only files
- Indents: 4 spaces (per level)
- PHP keywords and constants `true`, `false`, and `null` must be lower case
- Opening braces:
  - Classes and methods: next line
  - Control structures: same line
- Visibility: properties and methods must be declared `public`, `protected`, or `private`

Checking and Enforcing Standards

- Code Reviews
- PHP_CodeSniffer (phpcs)
- PHP Code Beautifier and Fixer (phpcbf)

PHP_CodeSniffer

```
$ ./vendor/bin/phpcs --standard=PSR2 src/Controller/IndexController.php

FILE: src/Controller/IndexController.php
----------------------------------------------------------------------
FOUND 9 ERRORS AFFECTING 9 LINES
----------------------------------------------------------------------
  2 | ERROR | [x] There must be one blank line after the namespace
    |       |     declaration
 11 | ERROR | [x] Multi-line function call not indented correctly;
    |       |     expected 12 spaces but found 0
 12 | ERROR | [x] Multi-line function call not indented correctly;
    |       |     expected 12 spaces but found 0
 13 | ERROR | [x] Multi-line function call not indented correctly;
    |       |     expected 12 spaces but found 0
 31 | ERROR | [x] Multi-line function call not indented correctly;
    |       |     expected 12 spaces but found 0
...
----------------------------------------------------------------------
PHPCBF CAN FIX THE 9 MARKED SNIFF VIOLATIONS AUTOMATICALLY
----------------------------------------------------------------------

Time: 74ms; Memory: 4.25Mb
```

PHP Code Beautifier and Fixer

```
$ ./vendor/bin/phpcbf --standard=PSR2 src/Controller/IndexController.php
Changing into directory /Users/patrickschwisow/code/standards-checker/src/Controller
Processing IndexController.php [PHP => 274 tokens in 42 lines]... DONE in 15ms (9 fixable
violations)
        => Fixing file: 0/9 violations remaining [made 2 passes]... DONE in 32ms
Patched 1 file
Time: 105ms; Memory: 4.75Mb

$ git diff
@@ -1,16 +1,15 @@
 <?php
 namespace PSchwisow\StandardsChecker\Controller;

-
 class IndexController extends BaseController
 {
     public function index()
     {
         $files = array_map(
             function ($file) {
-//                $return = explode('-', substr($file, 0, -5));
+            //                $return = explode('-', substr($file, 0, -5));
```

CodeSniffer Standards

Recommended

- PEAR
- PSR1
- PSR2

Outdated / Not Recommended

- MySource
- PHPCS
- Squiz
- Zend

Custom Standards

In your project root phpcs.xml

```
<?xml version="1.0"?>
<ruleset name="Foo_Project">
 <description>The coding standard for the Foo Project.</description>

 <!-- directories and files to check -->
 <file>src</file>
 <file>public/index.php</file>

 <!-- directory to exclude -->
 <exclude-pattern>*/Tests/*</exclude-pattern>

 <!-- command-line arguments -->
 <arg name="report" value="summary"/>
 <arg value="np"/>

 <!-- use all of PSR-1 -->
 <rule ref="PSR1"/>

 <!-- All PHP files MUST use the Unix LF (linefeed) line ending. -->
 <rule ref="Generic.Files.LineEndings">
  <properties>
   <property name="eolChar" value="\n"/>
  </properties>
 </rule>
</ruleset>
```

See https://github.com/squizlabs/PHP_CodeSniffer/wiki/Coding-Standard-Tutorial for more details.

Automation!

SUBVERSION

git

Bamboo

Travis CI

Jenkins

Documentation Standards: DocBlocks

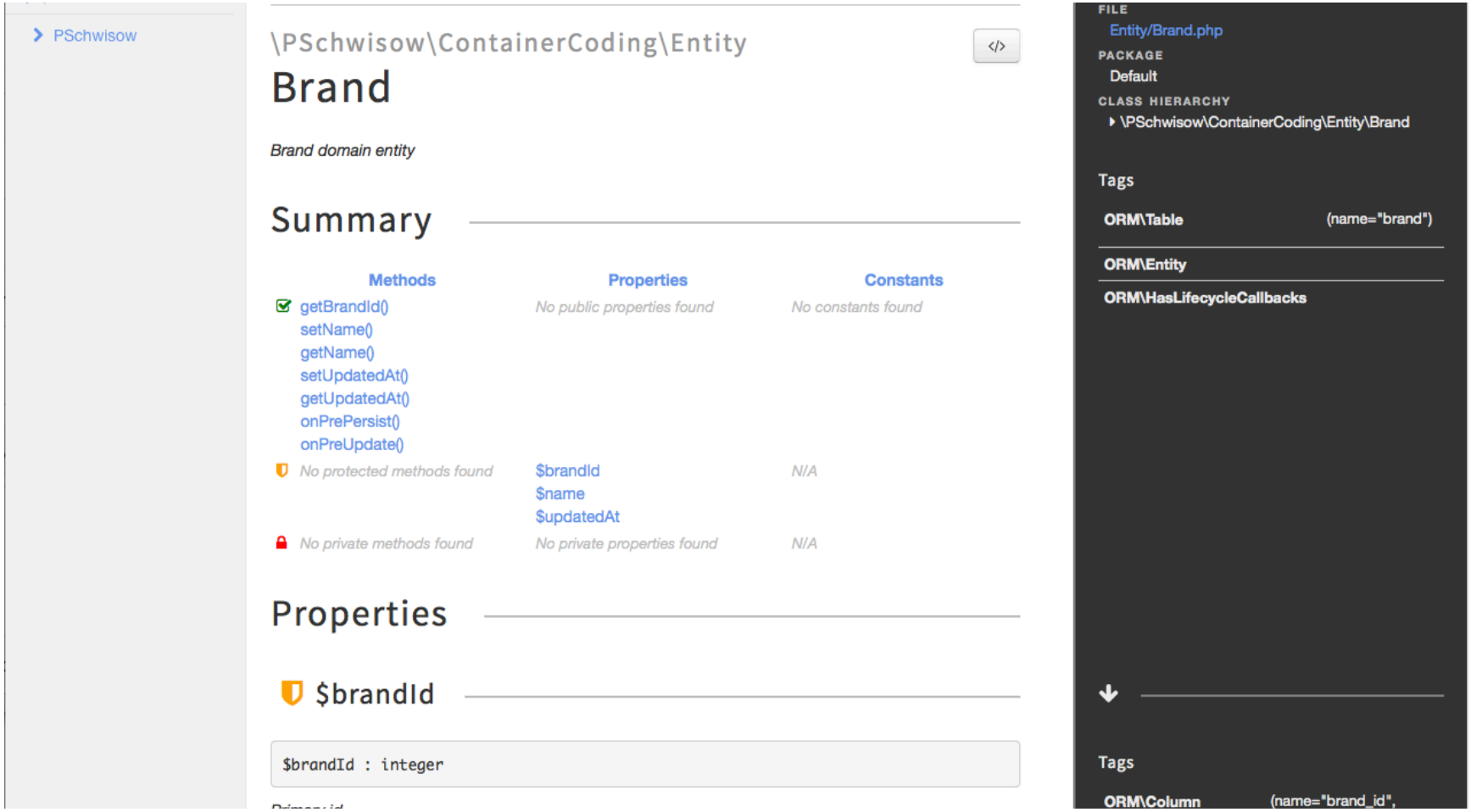
```
<?php
/**
 * A summary informing the user what the associated element does.
 *
 * A *description*, that can span multiple lines, to go _in-depth_
 * into the details of this element and to provide some background
 * information or textual references.
 *
 * @param string $myArgument With a *description* of this argument,
 *    these may also span multiple lines.
 * @return void
 */
function myFunction($myArgument)
{
}
```

Documentation Standards: DocBlocks

DocBlocks are attached to Structural Elements:

- Function
- Constant
- Class
- Interface
- Trait
- Class constant
- Property
- Method
- File
- require / include

Documentation Standards: phpDocumentor



Documentation Standards: PSR-5

Other Standards

- Testing
- Version Control
- Process

Who Am I?
Feedback / Contact / Slides

- Software Engineer at Shutterstock
- Zend Certified Engineer – PHP 5 & Zend Framework
- Founder / Organizer of Lake / Kenosha PHP
- Email: patrick.schwisow@gmail.com
- Twitter: @Pschwisow
- Joind.in: https://joind.in/14619
- Slides: https://github.com/PSchwisow/Miscellaneous/