Costasur reglas y buenas prácticas

Qui-gon Jinn (aka Paco Vela)

Capítulo 1
Introducción

Cuando escribas código para el proyecto de *Costasur*, debes seguir estas normas. La regla de oro será: Imitar el código que ya existe en *Costasur*. La mayoría de los paquetes de open-sourcec y las bibliotecas utilizadas por Symfony2 / Costasur también siguen las mismas directrices, y tú también.

Recuerda que la principal ventaja de los estándares es que cada pieza de código se muestra de forma homogénea. Symfony se ajusta a las normas establecidas en el PSR-0, PSR-1 y el PSR-2 documentos. Puesto que una imagen - o algo de código - vale más que mil palabras, aquí hay un pequeño ejemplo que contiene la mayoría de las funciones que se describen a continuación.

Estos estándares son los mismos que los de Symfony2, y para documentarlo hemos copiado el siguiente codigo this site:

```
<?php
/*
 * This file is part of the Symfony package.
 *
 * (c) Fabien Potencier <fabien@symfony.com>
 *
 * For the full copyright and license information,
 * please view the LICENSE file that was distributed
 * with this source code.
 */
namespace Acme;
/**
 * Coding standards demonstration.
 */
class FooBar
{
    const SOME_CONST = 42;
    private $fooBar;
    /**
     * @param string $dummy Some argument description
     */
    public function __construct($dummy)
    {
        $this->fooBar = $this->transformText($dummy);
    }
    /**
```

```
     * @param string $dummy Some argument description
     * @param array $options
     *
     * @return string|null Transformed input
     */
    private function transformText($dummy, $options = array())
    {
        $mergedOptions = array_merge(
            $options,
            array(
                'some_default' => 'values',
                'another_default' => 'more values',
            )
        );
        if (true === $dummy) {
            return;
        }
        if ('string' === $dummy) {
            if ('values' === $mergedOptions['some_default']) {
                $dummy = substr($dummy, 0, 5);
            } else {
                $dummy = ucwords($dummy);
            }
        }
        return $dummy;
    }
}
```

Capítulo 2
Standars PSR para nuestro código

Se pretende aplicar un formato homogeneo al todo el codigo que generemos. Para este proposito se necesita una herramienta que realize este propósito.

Actualmente existe un grupo de pogramadores que se están ocupando de normalizar estos “habitos” y formalizarlos en una serie de reglas. Estas reglas se titulan PSR-0, PSR-1, PSR-2, PSR-3 y podemos encontrar información sobre ellas aquí.

Cada regla inplica cumplir con sus predecesoras, esto quiere decir, que un codigo en PSR-2, tiene tambien que cumplir PSR -1 y PSR-0. Estas son las que están aprobadas actualmente ENE-2113.

PSR-0: https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md.

PSR-1: https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md.

PSR-2: https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md.

PSR-3: https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-3-logger-interface.md.

RESUMEN DE PSR-0/PSR-2: http://www.nireleku.com/2012/10/convenciones-en-codigo-symfony-2/.

2.1 Que nos ofrece Symfony

De nuevo de mano de Fabian, nos llega el "PHP-CS-Fixer" https://github.com/fabpot/PHP-CS-Fixer , este bundle nos ayuda a aplicar automaticamente estas reglas a nuestro código. En su versión actual, solo llega hasta PSR-2, si bien, es importante destacar que no nos aplica PSR-2 al completo, Instalación:

Para instalarlo debemos añadir al require de nuestro composer.json la siguiente linea:

```
[php]
"fabpot/php-cs-fixer": "dev-master"
```

Los comandos están accesibles desde Proyecto/vendor/bin, algunos de estos comandos son:

```
// Para mostrar los archivos que deben fixearse, sin hacer nada, solo los anal
izalos:
php-cs-fixer fix /ruta –dry-run
```

```
// Para fixear los archivos de esa carpeta indicandole que es código de Symfon
y 2.1. y que nos llegue hasta el nivel 2 de PSR:
php-cs-fixer fix /ruta  --config=sf21  --level=psr2
```

NOTA: --level=(psr0,psr1,psr2,all) con all aplicamos todo.

MUY IMPORTANTE : Tener cuidado con lo que se fixea, porque puede que deje de funcionar.

Para agilizar el proceso hemos creado un comando:

```
[bash]
app/console costasur:psr:fixer
```

Para mas informacion, ejecutad el comando con el argumento *--help* .

La configuración y los parametros se pueden cambiar en /vendor/fabpot/php-cs-fixer/.php_cs . Concretametne los directorios que queremos excluir.

Otra opción que podemos tomar es no instalarlo como bundle. Se puede instalar de forma independiente con los siguientes comandos:

```
sudo curl http://cs.sensiolabs.org/get/php-cs-fixer.phar -o /usr/local/bin/php
-cs-fixer
sudo chmod a+x /usr/local/bin/php-cs-fixer
```

Como se indicó anteriormente, solo nos corrige algunas cosas, aquí os dejo la lista de

cambios en la versión actual y a que nivel de PSR pertenece cada uno:

- Indentation [PSR-2] Code must use 4 spaces for indenting, not tabs.
- Linefeed [PSR-2] All PHP files must use the Unix LF (linefeed) line ending.
- Trailing_spaces [PSR-2] Remove trailing whitespace at the end of lines.
- Unused_use [all] Unused use statements must be removed.
- Php_closing_tag [PSR-2] The closing ?> tag MUST be omitted from files containing only PHP.
- Short_tag [PSR-1] PHP code must use the long tags or the short-echo tags; it must not use the other tag variations.
- Return [all] An empty line feed should precede a return statement.
- Visibility [PSR-2] Visibility must be declared on all properties and methods; abstract and final must be declared before the visibility; static must be declared after the visibility.
- Braces [PSR-2] Opening braces for classes and methods must go on the next line, and closing braces must go on the next line after the body. Opening braces for control structures must go on the same line, and closing braces must go on the next line after the body.
- Phpdoc_params [all] All items of the @param phpdoc tags must be aligned vertically.
- Eof_ending [PSR-2] A file must always end with an empty line feed.
- Extra_empty_lines [all] Removes extra empty lines.
- Include [all] Include and file path should be devided with single space. File path should not be placed under brackets.
- Psr0 [PSR-0] Classes must be in a path that matches their namespace, be at least one namespace deep, and the class name should match the file name.
- Controls_spaces [all] A single space should be between: the closing brace and the control, the control and the opening parenthese, the closing parenthese and the opening brace.
- Elseif [PSR-2] The keyword elseif should be used instead of else if so that all control keywords looks like single words.

2.2 Como puedo añadir todo esto a SublimeTex.

Necesitaremos para ello:

```
- Tener el fixer instalado en symfony o por comando.
- Paquete PhpCs en Sublime.
- Requiere:
    - Pyton
    - Php_CodeSniffer 1.3+:
        pear install PHP_CodeSniffer-1.4.3
    - PHPMD -PHP Mess Detector
        sudo pear channel-discover pear.phpmd.org
        sudo pear channel-discover pear.pdepend.org
        sudo pear install --alldeps phpmd/PHP_PMD
```

```
    - El bunble php-cs_fixer o su versión como comando. (ya está instalado)
- Tenemos que configurarle en Preferences-Package Settings- PHP Code Snifer -S
etings Default.
    - Añadir el ejecutable del "php-cs-Fix" (o la ruta del vendor) en la confi
g del php snyfer (user).
       {
        "show_debug": true,
        "phpcs_executable_path": "/usr/local/bin/php-cs-fixer",
       }
    -
    - Poner el debug de la consola en true, para ver los errores en grande en
la consola.
    - Poner el resto de path.
```

Ya está, tenemos lo básico para funcionar, con boton derecho "PHP Code Sniffer" => sniff o fix this file. nos muestra los errores o fixea.

2.3 Como me adapto a los estandares de sf2

Debemos configurar correctamente el standard que utilizaremos para nuestro proyecto "sf2 en nuestro caso", y lo debemos hacer tanto en el sniffer como en jenkins. Estos estandares estan disponibles en la ruta:

```
/usr/share/php/PHP/CodeSniffer/Standars
```

Para Sublime lo indicamos en la configuración del usuario del paquete.

```
...
"phpcs_additional_args": {
    "--standard": "Symfony2",
    "-n": ""
},
...
```
