Aprende a documentar

Costasur

Capítulo 1
Requisistos previos

La documentacion de Costasur, la estamos realizando con un Bundle de Symfony llamado Easybook.

Con este Bundle podremos generar documentos en format .pub .pdf o .html a partir de un fichero en texto plano con extensión .md.

A continuación pasaremos a describir cuales son los requisitos previos que necesitamos tener instalados en nuestro sistema para que todo funcione correctamente.

1.1 Symfony 2.x

Como hemos dicho anteriormente, Easybook, es un Bundle de Symfony, por lo tanto para empezar deberemos de tener instalado el framework de Symfony (versión 2.1 en adelante).

Para realizar dicha instalación, antes tendremos que leer y aplicar el primer capítulo de la documentación principal de Costasur. En este capítulo, instalaremos el proyecto de Costasur y de esta forma tendremos Symfony instalado correctamente, ya que es el framework que usamos en el proyecto.

1.2 Bundle Easybook

El Bundle Easybook, ya lo tendremos configurado correctamente gracias al paso anterior, por lo tanto solamente tendremos que verificar que esté instalado correctamente en la siguiente ruta.

```
.../vendor/easybook/easybook
```

Y por otro lado nos aseguramos que está la documentación actual.

```
.../src/Costasur/DocumentationBundle/Doc/
```

1.3 Sami

Sami es un bundle de nuestro amigo Fabien Potencier. En este articulo nos explica en que consiste y porque lo creó.

Básicamente hace lo mismo que phpdocumentor, pero de una forma mas sencilla y con menos añadidos. Recorre toda nuestra aplicaciñon para para analizarla y documentarla. Utiliza plantillas twig para presentar los resultados y finalmente nos genera una aplicación html con los resultados.

Para la instalacion solo agregamos a require de nuestro composer:

```
"sami/sami": "dev-master"
```

Para comprobar que todo ha ido bien lo probamos desde /vendor/bin ejecutando sami.php sin argumentos.

Disponemos de un fichero de configuración que soporta incluso varias versiones, temporalmente este fichero se encuentra en:

```
app/config/samicfg.php
```

Para genera el codemap de pnuestro src nos vamos a vendor/bin y ejecutamos:

```
./sami.php update  ../../app/config/samicfg.php
```

NEW: Para facilitar el uso de este bundle hemos creado un comando propio:

```
app/console costasur:generate:document
```

Este comando realiza varias operaciones, las veremos mas adelante. Nos interesa usarlo, ya que una de estas operaciones es ejecutar Sami.

1.4 KnpSnappyBundle

Este bundle se utiliza para convertir ficheros htm en formato pdf o imagen. Para ello tira de “wkhtmltopdf”.

Se habla de él de forma detallada en el Capitulo 5 de libro "Instalacion del Entorno de Trabajo".

Capítulo 2
Documentación

Una vez que ya tenemos nuestro sistema configurado correctamente, vamos a ver donde se encuentran los directorios de la documentación.

Abrimos el proyecto con nuestro IDE o editor de texto, y abrimos la siguiente ruta:

`.../src/Costasur/DocumentationBundle/Doc/*`

2.1 Organización de los documentos

Dentro del directorio que vimos en el apartado anterior, podemos ver como las documentaciones se estructuran de la siguiente forma:

```
root/
    Contents/
        chapter1.md
        chapter2.md
        appendix1.md
    Output/
        print/
            book.pdf
        html/
    config.yml
```

En el directorio Contents, están los distintos capítulos, donde cada uno de ellos están englobados en un fichero de extensión .md

En el directorio Output, están los distintos ficheros generados por Easybook, recordemos que pueden ser PDF, HTML o .Epub.

Por otro lado el `config.yml` es el fichero de configuración, donde podremos definir los parámetros generales de la documentación, como es el título, autor, el numero de capítulos que tiene, qué capitulos están incluidos en el libro, etc ...

Para terminar, el fichero generate.sh, es el fichero que compilamos y generamos el pdf. Una vez que tenemos el fichero abierto, con el entorno Sublime Text 2 por ejemplo, lo compilamos con el atajo de teclado *ctrl + b* y nos generará un PDF, en el directorio Output/print con el nombre book.

De esta forma ya sabemos donde se encuentra los distintos ficheros de la documentación y para qué sirve cada uno.

2.2 Como crear un libro

Una vez que sabemos como está estructurado Easybook, vamos a comenzar a crear un nuevo libro.

En primer lugar para crear un nuevo libro, hacemos los siguientes pasos:

- Nos situamos en la raíz de nuestro proyecto:

```
cd /var/www/costasur/
```

- Escribimos la siguiente línea en nuestra terminal.

```
vendor/easybook/  vendor/easybook/easybook/book new "Test" --dir="src/Costasur
/DocumentationBundle/Doc"
```

- Una vez creado el libro, se nos habrá creado una nueva estructura de libro en la ruta definida por --dir con el nombre Test en la siguiente ruta:

```
/var/www/costasur/src/DocumentationBundle/Doc/Test
```

2.3 Configuración de un libro

Una vez creado el libro, el siguiente paso es configurarlo. Para ello abrimos el fichero config.yml que está en el raiz del directorio del libro y lo más importante que tenemos que configurar son los siguientes campos.

- Title: Título del libro
- Author: Autor
- Contents (element): Tendremos que definir el número de capitulos que va a tener el libro, en función de los capitulos que tenga tendremos que ir añadiendo líneas a este array.

Por defecto se nos crea dos capítulos como podemos ver a continuación:

```
contents:
    # available content types: acknowledgement, afterword,
    # appendix, author, chapter, conclusion, cover,
    # dedication, edition, epilogue, foreword,
    # glossary, introduction, license, lof (list of figures),
    # lot (list of tables), part, preface, prologue,
    # title, toc (table of contents)
    - { element: cover }
    - { element: toc   }
    - { element: chapter, number: 1, content: chapter1.md }
    - { element: chapter, number: 2, content: chapter2.md }
```

Si quisiesemos crear un nuevo capítulo más , añadiriamos la siguiente línea, donde chapter3.md es el nombre del fichero que contendrá el capitulo en sí dentro del directorio del libro dentro de Contents.

```
   - { element: chapter, number: 3, content: chapter3.md }
```

2.4 Escribir un libro

Una vez que tenemos el libro configurado y creado, nos faltaría empezar a crear el contenido de los capítulos. Para ello abrimos el fichero md del capitulo, para recordar, este fichero debe estar en:

```
/var/.../DocumentationBundle/.../*Contents*/chapter1.md
```

El libro se escribe en un fichero de texto plano, el cual usa la sintaxis llamada Markdown.

Referencia de sintaxis. Enlace

*Es **indispensable** leer la documentación de Markdown. Y usar 4 espacios para indentar...*

2.5 Etiquetas más utilizadas

En este apartado iremos añadiendo las sintaxis más utilizadas a lo largo de la documentación.

2.5.1 Títulos

```
# Documentacion --> título h1
## Primera entrada --> subtítulo h2
### Primera Subentrada --> h3
...
```

2.5.2 Listas

Con markdown es muy fácil crear listas *desordenadas* (ul)

```
* Test 1
* Test 2
```

y *ordenadas* (ol)

```
1. Test 1
```

```
2. Test 2
```

2.5.3 Código de programación embebido

Por otro lado podemos incrustar o pegar código html en nuestro libro, simplemente indentándolo. Si queremos que se resalte la sintaxis apropiada lo indicaremos entre corchetes.

Así este código:

```
[php]
<?php
    echo "Hola mundo";
?>
```

Genera el contenido:

```php
<?php
    echo "Hola mundo";
?>
```

Entre las etiquetas que hay disponibles están [bash], [php], [apache](para configuración), [ini] y casi la que os haga falta, ya que easybook internamente usa GeSHi http://qbnz.com/highlighter/ para el resaltado.

2.6 Tools

El codigo Markdown no es muy complejo, pero aún así, hechamos en falta un buen editor visual en Linux, que nos ayude en nuestra tarea de documentar.

Para Windows existe *MarkdownPad*, que es aceptable.

En Linux tenemos *reText*, que parece bastante básico.

Para Sublime text 2, tenemos un paquete interesante:

```
*Markdown sublime text preview*
```

Nos facilita, de forma sencilla (Alt+M), una vista previa de como quedará nuestro texto. Esta vista se actualiza al salvar cambios en nuestro fichero y tan solo tendremos que resfrescar el navegador, para ver el nuevo contenido.

2.6.1 Publicando la documentación

Para generar toda la documentación HTML-PHP hemos creado un comando de consola:

```
app/console costasur:generate:documentation
```

Este comando, sin argumentos, realiza los siguientes pasos:

- Genera html de los books creados en easybook.
- Convierte este html en pdf.
- Ejecuta "Sami" para generar el codemap del proyecto en html.
- Traduce este contenido a pdf.

- Mueve los ficheros al directorio public.
- Elimina ficheros imnecesarios.
- Regenera los assetts.
- Borra la cache.

Este comando encuentra dentro de DocumentationBundle "DocumentationCommand.php".

Para nas información sobre este comando ejecutenlo con el argumento --help.

Todo lo que se generase manda a la carpeta 'public' de este bundle. A su vez esta carpeta está linkeada en el directorio web de Symfony.

Para acceder a la documentación lo haremos desde

> `documentation.tratewolocal.com`

Por ultimo y de manera *temporal* disponemos de este fichero:

> `/src/Costasur/DocumentationBundle/Resources/public/index.html`

Debemos editarlo a mano, para añadir los nuevos libros que generemos.

Capítulo 3
Documenta tu código

Ni que decir tiene, la importantia de este punto, nos ayudará a comprenderlo mejor.

Seguiremos una serie de reglas y metodos para unificar criterios.

3.1 Las Cabeceras

En ellas guardaremos mucha información. Debén seguir este diseño según su tipo:

- Fichero:

```
/**
* ▌▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀
▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀

*  ▌     □
*  ▌  © COSTASUR.com     @version 1.0 @author Nobre y Apellidos <correo@costasu
r.com>
* ▌         □
* ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄ */
```

- Clase:

```
/**
*  » Clase creada generar los assets como links.
*     ┌────────────────────────────────────────────────────────────────────────
```

```
┌──────────────────────────────────────────┐
*      - @see http://documentation.costa.com/codemap/Costasur/fichero.html
*      - @version 1.0 @author Nobre y Apellidos <correo@costasur.com>
*    └──────────────────────────────────────────────────────────────────────────┘
└──────────────────────────────────────────┘
*/
```

- Metodo:

```
/**
*  »  Función que sirve para
*
*      @param Nombre  $var  Describe el parametro
*/
```

- Variable:

  [php] /**

  - » @var La utilizamos para recoger el objeto xx */

3.1.1 CONSIDERACIONES:

- Al generar un nuevo archivo nos identificaremos como su autor en la cabecera del fichero.
- No es necesario añadir el autor a cada metodo, funcion o clase, basta con hacerlo en el fichero .
- Si otro usuario agrega un metodo a este fichero lo documenta poniendose como autor de este nuevo método.
- Si otro usuario lo moficica, se generará una nueva versión y se añadirá como autor del mismo
- Las anotaciones @see apuntarán a la documentación oficial de costasur.
- No es obligatorio comentar todas las variables o parametros. No es obligatorio pero si recomendable si se considera importante para entender el funcionamiento del código.

3.2 Snippet para Sublime Text

Para agilizar la labor de generar las cabeceras hemos creado varios snippets.

3.2.1 Instalación

Para que el snippet obtenga nuestros datos debemos crear este fichero y poner nuestro nombre y correo correctamente. Debemos crearnos este fichero con nuestros datos:

```
/home/Usuario/.config/sublime-text-2/Packages/User/Default.tmPreferences
```

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com
/DTDs/PropertyList-1.0.dtd">
    <plist version="1.0">
    <dict>
        <key>name</key>
```

```
        <string>Globals</string>
        <key>scope</key>
        <string></string>
        <key>settings</key>
        <dict>
            <key>shellVariables</key>
            <array>
                <dict>
                    <key>name</key>
                    <string>TM_FULLNAME</string>
                    <key>value</key>
                    <string><![CDATA[NUESTRO NOMBRE Y APPELIDOS]]></string>
                </dict>
                <dict>
                    <key>name</key>
                    <string>TM_EMAIL</string>
                    <key>value</key>
                    <string><![CDATA[NUESTRO NOMBRE @costasur.com]]></string>
                </dict>
            </array>
        </dict>
    </dict>
    </plist>
```

Para los snippers, basta con generar estos ficheros, copiando el contenido:

```
/home/Usuario/.config/sublime-text-2/Packages/User/cabeceraFichero.sublime-snippet
```

```
<snippet>
    <content><![CDATA[
/**
* ▌▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬
*  ▌      □
*  ▌  © COSTASUR.com     @version 1.0 @author ${global:$TM_FULLNAME} <${global:$TM_EMAIL}>
* ▌         □
* ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ */
```

```
]]>
    </content>
    <tabTrigger>/</tabTrigger>
    <scope>source.php</scope>
</snippet>
```

```
/home/Usuario/.config/sublime-text-2/Packages/User/cabeceraClase.sublime-snippet
```

```
<snippet>
    <content><![CDATA[
/**
*  » ${1:Descripción de la Clase}.
*    ┌──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────┐
*      - @see http://documentation.costa.com/codemap/Costasur/${2:Ruta de la d
ocumentación}.
*      - @version 1.0 @author ${global:$TM_FULLNAME} <${global:$TM_EMAIL}>
*    └──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────┘
*/

]]>
    </content>
    <tabTrigger>/</tabTrigger>
    <scope>source.php</scope>
</snippet>
```

```
/home/Usuario/.config/sublime-text-2/Packages/User/cabeceraMetodo.sublime-snip
pet
```

```
<snippet>
<content><![CDATA[
/**
*  » ${1:Descripción del Método}.
*
*/

]]>
    </content>
    <tabTrigger>/</tabTrigger>
    <scope>source.php</scope>
</snippet>
```

```
/home/Usuario/.config/sublime-text-2/Packages/User/cabeceraVariable.sublime-sn
ippet
```

```
<snippet>
    <content><![CDATA[
/**
*  » @var ${1:Descripción de la Variable}.
*/

]]>
    </content>
    <tabTrigger>/</tabTrigger>
    <scope>source.php</scope>
</snippet>
```

3.2.2 Su uso

Para usarlos basta con escribir "/" y posteriormente pulsar la tecla "Tabulador". Sublime nos mostrará una lista con los snippers disponibles para ese comando.

Debemos hacer uso de las etiquetas @, para documentar nuestro código (@param, @return, @thows ...), si bien, será el PSR el que nos marcará el contenido mínimo que deben tener nuestras cabeceras.
