Actualización de una aplicación Symfony 3 a Symfony 4

No hay ninguna herramienta que automatice el proceso, por lo que hay que hacerlo de forma manual siguiendo estos pasos:

1. Instalar Flex como una dependencia de nuestro proyecto

> *composer require symfony/flex*

- Si el fichero composer.json de nuestro proyecto contiene la dependencia *symfony/symfony* debemos quitarla previamente:

> *composer remove symfony/symfony*

Añadimos el paquete *symfony/symfony* a la sección *conflict* fichero *composer.json* tal como se muestra a continuación para que no se vuelva a instalar otra vez:

```
{
    "require": {
        "symfony/flex": "^1.0",
    },
    "conflict": {
        "symfony/symfony": "*"
    }
}
```

Lo siguiente es añadir las dependencias de nuestro proyecto. Podemos añadir todas las dependencias que teníamos y luego ir quitando las que no necesitemos. Por ejemplo:

> *composer require annotations asset orm-pack twig logger mailer form security translation validator*

> *composer require --dev dotenv maker-bundle orm-fixtures profiler*

En caso de que no tuviéramos dependencias con symfony/symfony, únicamente necesitaríamos reinstalar las dependencias para que flex genere los ficheros de configuración en *config/*

> *rm -rf vendor/**

> *composer install*

2. Restaurar la configuración

Sea cual sea el camino hasta aquí, ahora tendremos muchos ficheros nuevos en *config/*. Estos ficheros contienen la configuración por defecto definida por Symfony, por lo que debemos comprobar cada uno de los ficheros y configurarlo de acuerdo a la configuración que teníamos en app/config.

NOTA: Los ficheros no coinciden exactamente 1 a 1, ni en número de

ficheros, ni en nombre, ni en ubicación. Por ejemplo, el contenido del antiguo *app/config/config_dev.yml* debe ir en los ficheros .yaml de config/packages/dev/.

El fichero services.yaml

El fichero más relevante en este proceso de "migración" es *app/config/services.yml*, que ahora está en *config/services.yaml*. Se debe respetar el contenido generado por defecto por Flex y añadir después nuestra propia configuración.

Sin embargo, gracias al *autowiring* es posible que podamos eliminar la mayor parte de los servicios configurados.

3. Recolocar ficheros en los directorios correspondientes

- app/Resources/views/ -> templates/
- app/Resources/translations/ -> translations/
- app/Resources/<BundleName>/views/ -> templates/bundles/<BundleName>/
- El resto de los ficheros de app/Resources/ -> src/Resources/
- src/AppBundle/* -> src/

Después de mover los ficheros, actualiza las secciones autoload y autoload-dev del fichero composer.json para utilizar los namespaces App\ y App\Tests\ .

Si utilizaste varios bundles para organizar el código, debes reorganizar

todos los bundles en la única carpeta src/.

src/AppBundle/Resources/public/ -> public/

4. Crear el fichero index.php

Crea un fichero public/index.php copiando el contenido del mismo fichero de cualquier proyecto Symfony 4.

5. Cambiar el script bin/console

Cambia el contenido del script bin/console por el de symfony 4.

Actualización de una aplicación Symfony 4 a Symfony 5

No hay ninguna herramienta que automatice el proceso, por lo que hay que hacerlo de forma manual siguiendo estos pasos:

1. Actualizar nuestro proyecto hasta la versión 4.4
2. Eliminar todo el código obsoleto (deprecated)
3. Actualizar el proyecto a la versión 5