Symfony Flex

Flex es la herramienta que reemplaza al instalador de Symfony.

Symfony Flex automatiza las tareas más comunes de las aplicaciones symfony, como la propia instalación, las dependencias de composer…

Symfony Flex está disponible desde la versión 3.3 de Symfony. En Symfony 4.0 no es obligatorio todavía utilizarlo, pero es ya la herramienta utilizada por defecto.

Cómo funciona

Symfony Flex es un plugin de composer que modifica el comportamiento de los comandos *require*, *update* y *remove*.

Cuando se instalan o desinstalan dependencias en aplicaciones gestionadas con Flex, Symfony puede realizar tareas antes y después de la ejecución de las propias tareas de composer.

Veamos un ejemplo:

Si ejecutamos el siguiente comando:

> *composer require mailer*

Sin Flex, composer daría un error, porque *mailer* no es ningún nombre

de paquete válido. Sin embargo, con Flex, el comando acaba instalando y activando el paquete SwiftmailerBundle.

Cuando tenemos Flex instalado en nuestra aplicación y ejecutamos *composer require*, la aplicación realiza una petición al servidor de Symfony Flex para buscar allí el mejor paquete para el nombre indicado.

Si no hay información en el server, entonces se ejecutaría el proceso normal de Composer.

Si hay información sobre el paquete, Flex la devuelve en un fichero llamado “recipe”, y la aplicación la utiliza para decidir qué paquete instalar y qué tareas ejecutar después de la instalación.

En el ejemplo anterior, Symfony Flex pregunta sobre el package *mailer* y el servidor de Symfony Flex detecta que *mailer* es un alias de *SwiftmailerBundle* y devuelve el archivo “recipe” correspondiente.

Flex anota las “recipes” instaladas en el archivo symfony.lock. Este archivo debería subirse al repositorio de código.

Symfony Flex Recipes

Las Recipes se definen en un archivo manifest.json y pueden contener cualquier número de otros ficheros y directorios. Por ejemplo, este es el contenido del archivo manifest.json de SwiftmailerBundle:

```
{
    "bundles": {
        "Symfony\\Bundle\\SwiftmailerBundle\\SwiftmailerBu
ndle": ["all"]
    },
    "copy-from-recipe": {
        "config/": "%CONFIG_DIR%/"
    },
    "env": {
        "MAILER_URL": "smtp://localhost:25?encryption=&aut
h_mode="
    },
    "aliases": ["mailer", "mail"]
}
```

(Podemos encontrar esta recipe en https://github.com/symfony/recipes/blob/master/symfony/swiftmailer-bundle/2.5/manifest.json)

La opción **aliases** permite a Flex instalar paquetes utilizando nombres

cortos y fáciles de recordar. (El original en este caso sería composer require symfony/swiftmailer-bundle)

La opción **bundles** informa a Flex de en qué entornos se debe habilitar este bundle. En este caso, en todos ([“all”])

La opción **env** hace que Flex cree una variable de entorno en la aplicación.

La opción **copy-from-recipe** permite a la “recipe” copiar ficheros y directorios en la aplicación.

Las instrucciones definidas en este archivo manifest.json son utilizadas por Symfony Flex igualmente para desinstalar un paquete deshaciendo todas las operaciones realizadas en la instalación.

Las recipes de Symfony Flex se almacenan en dos repositorios públicos:

- **Main recipe** repository (https://github.com/symfony/recipes), es una lista de recipes de paquetes de alta calidad y con mantenimiento. Por defecto, Symfony Flex únicamente consulta este repositorio.

- **Contrib recipe** repository (https://github.com/symfony/recipes-contrib), contiene todas las recipes creadas por la comunidad. Todas tienen garantía de funcionamiento, pero no se asegura el mantenimiento.

Para utilizar este repositorio habría que ejecutar el comando:

> *composer config extra.symfony.allow-contrib true*

Actualizar aplicaciones a Flex

Para tener Flex, no basta con instalar la dependencia symfony/flex en nuestro proyecto. Symfony Flex necesita que las aplicaciones utilicen la siguiente estructura de directorios, que evidentemente, coincide con la estructura de directorios de Symfony 4:

```
miproyecto/
├── assets/
├── bin/
│   └── console
├── config/
│   ├── bundles.php
│   ├── packages/
│   ├── routes.yaml
│   └── services.yaml
├── public/
│   └── index.php
├── src/
│   ├── ...
│   └── Kernel.php
├── templates/
├── tests/
├── translations/
├── var/
└── vendor/
```

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*

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 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. Gracias al *autowiring* es posible que podamos eliminar la mayor parte de los servicios configurados.

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/

Crear el fichero index.php

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

Cambiar el script bin/console

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