Desarrollo de aplicaciones con Symfony 4

Traducciones

Traducciones: Instalación

En la aplicaciones que dispongan de symfony flex y que no hayan sido instaladas con **symfony/website-skeleton,** donde este componente viene por defecto, ejecutaremos:

```
$  composer require symfony/translation
```

Traducciones: Configuración

En confir/packages deberíamos tener el archivo translation.yaml donde aplicaremos la configuración que deseemos. Por defecto:

```
# config/packages/translation.yaml
framework:
    default_locale: '%locale%'
    translator:
        default_path: '%kernel.project_dir%/translations'
        fallbacks:
            - '%locale%'
```

Traducciones: Traducción básica

Symfony nos provee del servicio traslator para realizar las traducciones, tanto en un controlador como en un servicio bastará con inyectar el servicio y llamar a la funcions trans():

```
use Symfony\Contracts\Translation\TranslatorInterface;

public function index(TranslatorInterface $translator)
{
    $translated = $translator->trans('Symfony is great');
    // ...
}
```

Traducciones: Traducción básica

Cuando se ejecuta este código, Symfony intentará traducir el mensaje según la configuración regional del usuario (el locale). Para que esto funcione, debes decirle a Symfony cómo traducir el mensaje a través de un "recurso de traducción", que generalmente es un archivo que contiene una colección de traducciones para un locale determinada.

Este "diccionario" de traducciones se puede crear en varios formatos diferentes, siendo XLIFF el formato recomendado por symfony.

translations/messages.es.yaml

Symfony is great: Symfony es genial

Traducciones: Traducción básica

¿Donde colocaremos esos archivos de traducciones?

Tenemos varias posibilidades:

- En la carpeta **translations/** (en el raiz de nuestro proyecto);
- En la carperta **src/Resources/<bundle name>/translations/** **#Esta deprecated desde la 4.2**
- En la carpeta **Resources/translations/** dentro de cualquier bundle

Traducciones: Traducción básica

¿Cual es el proceso de traducció?

1. Se determina la configuración regional del usuario actual, que se almacena en la solicitud.
2. Un catálogo (por ejemplo, una gran colección) de mensajes traducidos se carga de los recursos de traducción definidos para la configuración regional (por ejemplo, fr_FR). Los mensajes de la configuración regional alternativa también se cargan y se agregan al catálogo si aún no existen. El resultado final es un gran "diccionario" de traducciones.
3. Si el mensaje se encuentra en el catálogo, se devuelve la traducción. Si no, el traductor devuelve el mensaje original.

Traducciones: Message Placeholders

A veces, un mensaje que contiene una variable necesita ser traducido:

```
$translated = $translator->trans('Hello '.$name);
```

¡Esto no funcionaria!

La forma correcta sería:

```
$translated = $translator->trans( 'Hello %name%', ['%name%' => $name]);
```

y en el archivo de traducciones:

```
'Hello %name%': Hola %name%
```

Traducciones: Pluralización

Otra complicación es cuando tiene traducciones que pueden o no ser plurales.

Por ejemplo, en nuestro código podríamos querer mostrar:

- Hay una incidencia pendiente.
- Hay 5 incidencias pendientes.

¿como soluciona esto Symfony?

Traducciones: Pluralización

Cuando una traducción tiene diferentes formas debido a la pluralización, puede proporcionar todas las formas como una cadena separada por una tubería (|) y utilizar el método transChoice():

```
$translator->transChoice(
    'There is one issue|There are %count% issues,
    5
);
```

Traducciones: Pluralización

¿Como sabe symfony cual elegir en cada caso?

Sobre la base del número dado, el traductor elige la forma plural correcta. En inglés, la mayoría de las palabras tienen una forma singular cuando hay exactamente un objeto y una forma plural para todos los demás números (0, 2, 3 ...). Por lo tanto, si cuenta es 1, el traductor usará la primera cadena (There is one issue) cómo traducción. De lo contrario usará There are %count% issues

Traducciones: Pluralización

Pero esto puede variar según el idioma, por ejemplo, en francés se usará la forma singular cuando tanto cuando sea 1 como cuando sea 0.

Cada locale tiene su propio conjunto de reglas, y algunas tienen hasta seis formas plurales diferentes con reglas complejas detrás de las cuales los números se asignan a cada forma plural. Las reglas son bastante simples para el inglés y el francés, pero, por ejemplo, para el ruso, es posible que desee una pista para saber qué regla coincide con qué cadena. Para ayudar a los traductores, puede "etiquetar" opcionalmente cada cadena:

'one: There is one issue|some: There are %count% issues'

'none_or_one: Il y a %count% issue|some: Il y a %count% issues'

Traducciones: Pluralización explícita

La forma más fácil de pluralizar un mensaje es dejar que el Traductor use la lógica interna para elegir qué cadena usar en función de un número dado. Pero ya hemos visto que en algunos idiomas puede ser una locura o simplemente podemos querer más control. Para ello:

'{0} There are no issues|{1} There is one issue|]1,19] There are %count% issues|[20,Inf[There are many issues'

Más información:
https://symfony.com/doc/current/components/translation/usage.html#explicit-interval-pluralization

Traducciones: Forzando el locale

Cuando se traduce un mensaje, el Traductor utiliza la configuración regional especificada o la configuración regional alternativa si es necesario. También puede especificar manualmente la configuración regional que se utilizará para la traducción:

```
$translator->trans(
    'Symfony is great',
    [],
    'messages',
    'fr_FR'
);
```

Traducciones: Forzando el locale

```
$translator->transChoice(
    '{0} There are no issues|{1} There is one issue|]1,Inf[ There are %count% issues,
    10,
    [],
    'messages',
    'fr_FR'
);
```

Traducciones: Forzando el locale

```
$translator->transChoice(
    '{0} There are no issues|{1} There is one issue|]1,Inf[ There are %count% issues,
    10,
    [],
    'messages',
    'fr_FR'
);
```

Traducciones: Traduciendo templates

Podemos usar las etiquetas trans y transchoice:

{% trans %}Hello %name%{% endtrans %}

{% transchoice count %}

{0} There are no apples|{1} There is one apple|]1,Inf[There are %count% apples

{% endtranschoice %}

Traducciones: Traduciendo templates

O los flitros:

{{ message|trans }}

{{ message|transchoice(5) }}

El uso de etiquetas o filtros de traducción tiene el mismo efecto, pero con una diferencia sutil: el escape de salida automático solo se aplica a las traducciones que usan un filtro. En otras palabras, si necesitamos asegurarnos de que su mensaje traducido no se haya escapado, debe aplicar el filtro en bruto después del filtro de traducción:

Traducciones: Traduciendo templates

```
{# text translated between tags is never escaped #}
{% trans %}
    <h3>foo</h3>
{% endtrans %}

{% set message = '<h3>foo</h3>' %}
{# strings and variables translated via a filter are escaped by default #}
{{ message|trans|raw }}
{{ '<h3>bar</h3>'|trans|raw }}
```

Traducciones: Creando/actualizando catalogos

La tarea que requiere más tiempo al traducir una aplicación es extraer todo el contenido de la plantilla a traducir y mantener todos los archivos de ta sincronizados. Symfony incluye un comando llamado translation:update que te ayuda con estas tareas:

```
$ php bin/console translation:update --dump-messages --force fr
```

Traducciones: Los arvhivos de traducción

Ya hemos visto que tenemos varias posibilidades:

- En la carpeta **translations/** (en el raiz de nuestro proyecto);
- En la carperta **src/Resources/<bundle name>/translations/** **#Esta deprecated desde la 4.2**
- En la carpeta **Resources/translations/** dentro de cualquier bundle

Traducciones: Los archivos de traducción

- Estas ubicaciones están listadas con la prioridad más alta primero. Es decir, puede anular los mensajes de traducción de un paquete en cualquiera de los dos directorios principales.
- El mecanismo de anulación funciona a nivel de clave: solo las claves anuladas deben incluirse en un archivo de mensajes de mayor prioridad. Cuando no se encuentra una clave en un archivo de mensajes, el traductor retrocederá automáticamente a los archivos de mensajes de prioridad más baja.

Traducciones: Los archivos de traducción

El nombre de archivo de los archivos de traducción también es importante: cada archivo de mensaje debe nombrarse de acuerdo con la siguiente ruta: domain.locale.loader:

- **domain**: una forma opcional de organizar los mensajes en grupos (por ejemplo, admin, navegación o los mensajes predeterminados). Consulte Uso de dominios de mensajes.
- **locale**: la configuración regional para la que se realizan las traducciones (por ejemplo, en_GB, en, etc.);
- **loader**: cómo Symfony debe cargar y analizar el archivo (por ejemplo, xlf, php, yaml, etc.).

Traducciones: Los archivos de traducción

¿Y qué loaders son soportados?

- xlf: XLIFF file; # Recomendado por Symfony.
- php: PHP file;
- yaml: YAML file. #El más sencillo para aplicaciones pequeñas.

Traducciones: Los archivos de traducción

También podemos definir rutas propias donde guardar archivos de traducción:

```
# config/packages/translation.yaml
framework:
    translator:
        paths:
            - '%kernel.project_dir%/custom/path/to/translations'
```

Traducciones: ¡Ojo con la cache!

Cada vez que creemos un nuevo recurso de traducción (o instalemos un paquete que incluya un recurso de traducción), hay que asegurarse de borrar tu caché para que Symfony pueda descubrir los nuevos recursos de traducción:

```
$ php bin/console cache:clear
```

Traducciones: Fallback

```
# config/packages/translation.yaml
framework:
//...
    fallbacks:
        - '%locale%'
```

- Primero, Symfony busca la traducción en un recurso de traducción fr_FR (por ejemplo, messages.fr_FR.xlf);
- Si no se encontró, Symfony busca la traducción en un recurso de traducción fr (por ejemplo, messages.fr.xlf);
- Si aún no se encuentra la traducción, Symfony usa el parámetro de configuración de reserva, que por defecto

Traducciones: Traduciendo el contenido de la base de datos

Para gestionar las traducciones del contenido de la base de datos tenemos tres opciones:

Utilizar Translatable extension de Doctrine:
http://atlantic18.github.io/DoctrineExtensions/doc/translatable.html

Utilizar Translatable Behavior de Doctrine:

https://github.com/KnpLabs/DoctrineBehaviors#translatable

Gestionarlo nosotros mismos.

Traducciones: Translatable extension

Para trabajar con las extensiones de Doctrine deberemos instalar un Bundle llamado StofDoctrineExtensionsBundle

```
$ composer require stof/doctrine-extensions-bundle
```

Este bundle nos proporciona todas las extensiones de Doctrine, no solo translatable.

Más información

https://symfony.com/doc/current/bundles/StofDoctrineExtensionsBundle/index.html

Traducciones: Translatable extension

Activamos las extensiones necesarias:

https://symfony.com/doc/current/bundles/StofDoctrineExtensionsBundle/configuration.html#add-the-extensions-to-your-mapping

Traducciones: Translatable extension

Ya podemos seguir la documentación de la extensión para hacer traducible nuestra entidad y trabajar con las diferentes traducciones:

https://github.com/Atlantic18/DoctrineExtensions/blob/v2.4.x/doc/translatable.md

Traducciones: Doctrine2 Behaviors

También vamos a necesitar instalar un bundle de terceros:

```
$ composer require knplabs/doctrine-behaviors:~1.1
```

Traducciones: Doctrine2 Behaviors

Ver documentación de bundle

https://github.com/KnpLabs/DoctrineBehaviors#translatable

Traducciones: Traduciendo rutas

En versiones anteriores de Symfony esto no venía de incluido en el propio symfony y había que usar librerias externas. La más conocida y descargada era JMSI18nRoutingBundle.

A partir de Symfonoy 4.1 esta funcionalidad ya viene incluida en el propio Symfony

Traducciones: Traduciendo rutas

En el caso de utilizar yaml para las traducciones:

```
contact:
    controller: App\Controller\ContactController::send
    path:
        en: /send-us-an-email
        nl: /stuur-ons-een-email
        es: /contactanos
```

Traducciones: Traduciendo rutas

Internamente Symfony crea una ruta por cada idioma (en el ejemplo anterior, se crean las rutas contact.en y contact.nl) pero puedes seguir usando el nombre original de la ruta para generar las URLs.

Cuando se genera una URL, se usa por defecto el idioma del usuario actual:

```
$url = $urlGenerator->generate('contact');
```

pero puedes pasar el idioma explícitamente para forzar la generación de una determinada URL:

```
$url = $urlGenerator->generate('contact', ['_locale' => 'es']);
```

Traducciones: Traduciendo rutas

Esto también funcionaría, pero no se recomienda:

```
$url = $urlGenerator->generate('contact.nl');
```

Traducciones: Traduciendo rutas

Los prefijos de las rutas también se pueden traducir (tanto en YAML/XML como en la anotación @Route) y puedes combinarlo tanto con rutas traducidas como con rutas sin traducir:

```
# config/routes/annotations.yaml
site:
    resource: '../src/Controller/'
    type: annotation
    prefix:
        en: '/site'
        es: '/sitio'
```

Traducciones: Traduciendo rutas

```
class DefaultController extends Controller {
    /**
     * @Route({"en": "/contact", "es": "/contacto"}, name="contact")
     */
    public function contact()  { ... }

    /**
     * @Route("/page/{slug}", name="page")
     */
    public function page($slug) { // ... }
}
```

AvaiBook BOOKING

Symfony

Traducciones: Traduciendo rutas

Lo anterior nos generaría:

Nombre de la ruta	**Path de la ruta**
contact.en	/site/contact
contact.es	/sitio/contacto
page.en	/site/page/{slug}
page.es	/sitio/page/{slug}

¿Preguntas?