Traducciones

Instalación

Para instalar el componente de traducciones mediante Flex, se utiliza el siguiente comando:

> *composer require translator*

Configuración

El comando anterior crea un archivo de configuración en el que se pueden definir el *locale* por defecto y los *fallback locales* que se utilizarán si Symfony no encuentra la traducción buscada.

```
# config/packages/translation.yaml
framework:
    default_locale: 'es'
    translator:
        fallbacks: ['es','en']
        # ...
```

El locale utilizado en la traducción es el que se indique en la request. Si no se indica ninguno, entonces será el indicado en la configuración como *default_locale*.

Traducciones básicas

Las traducciones de texto en twig se realizan a través de las etiquetas **trans** y **transchoice**.

```
{% trans %}mensaje{% endtrans %}

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

No obstante, es posible utilizar filtros en lugar de etiquetas:

```
{{ mensaje|trans }}

{{ mensaje|transchoice(5) }}
```

Las traducciones de texto en el código se realizan a través del servicio **Translator** y su método **trans()**.

```
// ...
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Translation\TranslatorInterface;

public function index(TranslatorInterface $translator)
```

```
{
    $translated = $translator->trans('Symfony es fantástic
o');

    // ...
}
```

Las traducciones se almacenan en ficheros con formato XML, YAML o PHP, siendo el recomendado el XML, concretamente el formato XLIFF, por se un formato estándar de traducciones.

```
<!-- translations/messages.en.xlf -->
<?xml version="1.0"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:docum
ent:1.2">
    <file source-language="es" datatype="plaintext" origin
al="file.ext">
        <body>
            <trans-unit id="symfony_es_fantastico">
                <source>Symfony es fantástico</source>
                <target>Symfony is great</target>
            </trans-unit>
        </body>
    </file>
</xliff>
```

El proceso de traducción

Para traducir un mensaje, Symfony utiliza el siguiente proceso:

1. Determinar el **locale** del usuario actual a partir de la request

2. Cargar el catálogo de mensajes traducidos (diccionario) correspondiente al *locale* del usuario. También se añaden los mensajes traducidos faltantes del catálogo correspondiente al locale de fallback. El resultado final es un gran diccionario de traducciones.

3. Si el mensaje está en el catálogo, se devuelve la traducción. Si no, se devuelve el mensaje original.

Message Placeholders

Muchas veces los mensajes que queremos traducir, contienen variables.

Las traducciones de estos mensajes se realizan mediante los denominados **message placeholders**.

```
// ...
$translated = $translator->trans(
    'Hola %nombre%',
    array('%nombre%' => $nombre)
);

dump($translated);
```

O en una plantilla con la etiqueta trans:

```
{% trans with {'%nombre%': nombre} %}Hola %nombre%{% endtr
ans %}
```

o con el filtro trans:

```
{{ message|trans({'%nombre%': nombre}) }}
```

Si necesitáramos utilizar el símbolo del porcentaje en una cadena, podemos escaparla doblándola:

```
{% trans %}Porcentaje: %porcentaje%%%{% endtrans %}
```

En el fichero de traducciones quedaría así:

```
<?xml version="1.0"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2">
    <file source-language="es" datatype="plaintext" original="file.ext">
        <body>
            <trans-unit id="1">
                <source>Hola %nombre%</source>
                <target>Bonjour %nombre%</target>
            </trans-unit>
        </body>
    </file>
</xliff>
```

Manejo de plurales

Otras traducciones difíciles de manejar son los plurales o textos que dependen de un número variable.

Por ejemplo:

```
No hay ninguna manzana.
Hay 5 manzanas.
```

Para manejar estos casos, en las plantillas se utiliza la etiqueta o el filtro **transChoice**.

```
{% transchoice 10 %}
    {0} No hay ninguna manzana|{1} Hay una manzana|]1,Inf[
 Hay %count% manzanas
{% endtranschoice %}
```

Para traducir los textos en el código, se utiliza el método **transChoice()** del servicio.

```
$translator->transChoice(
    '{0} No hay ninguna manzana|{1} Hay una manzana|]1,Inf
[ Hay %count% manzanas',
    10
);
```

Desde Symfony 3.2, el tercer argumento de tranchoice() es opcional si el único placeholder es %count%. Symfony asigna automáticamente se asigna 10 a %count%. Si hubiera otros placeholders, ya sí que sería obligatorio.

```
$translator->transChoice(
    '{0} No hay ninguna manzana|{1} Hay una manzana|]1,Inf[ Hay %count% manzanas',
    10,
    array()
);
```

Los intervalos siguen la notación ISO 31-11. Ejemplos:

```
{1,2,3,4}

[1, +Inf[

]-1,2[
```

Los ficheros de traducciones

El comando translation:update

La mayor parte del tiempo dedicado a traducciones se lo lleva la gestión y mantenimiento de los ficheros de traducciones.

El componente de traducciones tiene un comando para ayudarnos en esta tarea: **translation:update**.

```
# actualiza el archivo de traducciones a Inglés con los me
nsajes que faltan encontrados en las plantillas app/Resour
ces/
 bin/console translation:update --dump-messages --force en

# actualiza el archivo de traducciones a Francés con los m
ensajes que faltan encontrados en el bundle AppBundle
 bin/console translation:update --dump-messages --force fr
 AppBundle
```

Si se ejecuta el comando sin la opción *–force*, aparecerán en la consola los mensajes que faltan por incluir en el fichero, pero no se incluirán.

Este comando no extrae mensajes de otras fuentes como controladores, formularios, mensajes flash, servicios… Si necesitáramos esta utilidad, Symfony nos recomienda utilizar el bundle TranlationBundle.

Nombre y ubicación de los ficheros de traducción

Symfony busca por defecto los ficheros de traducción en las siguientes ubicaciones, por orden de prioridad:

- El directorio **translations/** en el raíz del proyecto
- El directorio **src/Resources/{nombredelbundle}/translations/**
- El directorio **Resources/translations/** dentro de un bundle

Los mensajes que no se encuentren en la primera ubicación se buscarán en la siguiente, y así sucesivamente. Por lo tanto, para sobreescribir mensajes de un bundle, solamente es necesario sobrescribir los mensajes deseados, y no todo el fichero.

El nombre del fichero de traducción también es importante. Los ficheros se buscan con el siguiente nombre:
domain.locale.loader

domain: Podemos dividir los mensajes en diversos ficheros de traducción. El utilizado por defecto es **messages**.

locale: El locale al que se quiere traducir un mensaje (es_ES, es, etc)

loader: El formato del fichero de traducciones (xlf, php, yaml, etc).

Symfony puede manejar 3 formatos de loader:

- **xlf**: para archivos XLIFF
- **php**: para archivos PHP
- **yaml**: para archivos YAML

La opción recomendada por Symfony es XLIFF, pero no deja de ser una cuestión de gustos.

Nota
Se pueden añadir otros directorios en la opción **framework.translator.paths** de la configuración:

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

Cada vez que creemos un fichero de traducción nuevo debemos limpiar la caché para que symfony vuelva a revisar todas las ubicaciones y descubra el nuevo fichero.

> *bin/console cache:clear*

Tratamiento del *locale*

Imaginemos que el *locale* configurado por defecto es *es* y el *locale* del usuario (petición) actual es *en_GB* y que se quiere presentar la traducción de “Symfony es fantástico” en el *domain messages* y con el *loader xlf*.

Symfony buscará la traducción en los siguientes ficheros:

1º) En un fichero con nombre messages.en_GB.xlf

2º) En un fichero messages.en.xlf

3º) En un fichero con el locale configurado por defecto, en este caso messages.es.xlf.

Si Symfony no es capaz de encontrar la traducción buscada, lo indicará en el fichero de log, en la barra de debug y en el profiler.

Traducción del contenido de la base de datos

Symfony no proporciona soporte para la traducción del contenido de la base de datos.

Si necesitamos alguna librería que nos lo facilite, debemos buscar en las propias librerías de los ORM. Por ejemplo, en Doctrine, la *Translatable Extension* o el *Translatable Behaviour* o en Propel el *Translatable Behaviour*

Como traducir los mensajes de los Validadores

La traducción de los mensajes de validación es muy sencilla: basta con incluir la traducción en un fichero de traducciones con el *domain* **validators**.

Supongamos el siguiente mensaje de validación:

```
// src/Entity/Author.php
use Symfony\Component\Validator\Constraints as Assert;

class Autor
{
    /**
     * @Assert\NotBlank(message="autor.nombre.not_blank")
     */
    public $nombre;
}
```

Habría que incluir la traducción en un fichero *translations/validators.es.xlf*

```
<!-- translations/validators.es.xlf -->
<?xml version="1.0"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:docum
```

```
ent:1.2">
    <file source-language="es" datatype="plaintext" origin
al="file.ext">
        <body>
            <trans-unit id="autor.nombre.not_blank">
                <source>autor.nombre.not_blank</source>
                <target>El nombre del autor es obligatorio
.</target>
            </trans-unit>
        </body>
    </file>
</xliff>
```

Depuración de traducciones

Durante el desarrollo y el mantenimiento de una aplicación y/o de un bundle, continuamente se añaden, eliminan o modifican traducciones, y es muy fácil tener desactualizados los ficheros de traducción.

El comando **debug:translation** nos ayuda a depurar estas situaciones.

> *bin/console debug:translation fr*

```
+----------+------------------+----------------------+----
---------------------------+
| State(s) | Id               | Message Preview (fr) | Fal
lback Message Preview (en) |
+----------+------------------+----------------------+----
---------------------------+
| o        | Symfony is great | J'aime Symfony       | Sym
fony is great              |
+----------+------------------+----------------------+----
---------------------------+

Legend:
  x Missing message
  o Unused message
  = Same as the fallback message
```

Este comando nos informa de mensajes no traducidos, de mensajes traducidos pero no utilizados, y de mensajes cuya traducción es el mismo texto que el mensaje sin traducir (indica un posible olvido de traducirlo).

Podemos indicar que inspeccione únicamente un dominio

> *bin/console debug:translation en --domain=messages*

Que nos informe únicamente de los mensajes encontrados en el catálogo pero no en las plantillas

> *bin/console debug:translation en --only-unused*

O que nos informe únicamente de los mensajes encontrados en las plantillas pero no en los catálogos

> *bin/console debug:translation en --only-missing*