Simon GAUTIER

France

i18n internationalisation

Paramétrage / Utilisation des traductions

Gestion des variables et pluriels

Debug et génération de traductions

i18n – principes généraux

- Définition : fr.wikipedia.org/wiki/Internationalisation_(informatique)

- Locale = Langue + Pays
  - Codes de langues : fr.wikipedia.org/wiki/Liste_des_codes_ISO_639-1
  - Codes des pays : fr.wikipedia.org/wiki/ISO_3166-1_alpha-2
  - Exemples : en_US, en_GB, fr_FR, fr_CA, …
  - Souvent, on gère dans une application la langue sans la décliner par pays
    ➔ on utilise alors uniquement le code de langue. Exemple : en, fr, …

i18n – Paramétrage global

- Installation : **`composer require translation`**

- Configuration : fichier **`config/packages/translation.yaml`**

```yaml
framework:
    # default_locale : prise en compte uniquement si aucune locale n'est
    # explicitement définie par l'utilisateur => un seul choix possible
    default_locale: '%locale%'
    translator:
        paths:
            - '%kernel.project_dir%/translations'

        # Lorsqu'une traduction demandée dans la locale courante n'est pas
        # disponible, on utilise les fallbacks
        # Possible d'en définir plusieurs, Exemple : fr, sinon en, sinon ...
        fallbacks:
            - '%locale%'
```

Traduction de termes

- Dans un contrôleur, via le service « **`translator`** » :

```php
public function indexAction()
{
    $title = $this->get('translator')->trans('One term to translate');

    //...
}
```

- Dans un contrôleur, en utilisant l'injection de dépendance :
  - Cf. chapitre sur les services

```php
public function indexAction(TranslatorInterface $translator)
{
    $title = $translator->trans('One term to translate');

    // ...
}
```

Traduction de termes

- Avec Twig :

```
{% trans %}One term to translate{% endtrans %}

{{ 'One term to translate'|trans }}
```

- Attention : avec le tag **`{% trans %}`**, le comportement est différent de celui du filtre « **`trans`** » :

```
{% trans %}<span>One term to translate</span>{% endtrans %}

{{ '<span>One term to translate</span>'|trans }}
```


`<span>One term to translate</span>`
`<span>One term to translate</span>`

Domain	Times used	Message ID
messages	1	One term to translate
messages	2	One term to translate

Symfony considère qu'il s'agit de 2 termes bien distincts

Les fichiers de traduction

- Symfony sait gérer plusieurs formats de traduction : YAML, XLIFF, PHP Arrays, CSV, ICU (Data & RES), INI, MO / PO, Plain PHP, QT, JSON

- Emplacement des fichiers de traduction : répertoire « **`translations`** »
  - Les fichiers présents ici sont prioritaires ➔ permettent de surcharger les traductions que peuvent proposer certaines bundles dans **`vendor`** par exemple

- Nom des fichiers : [DOMAINE].[LOCALE].[FORMAT]
  - Domaine : classiquement « messages » mais il est possible de définir d'autres domaines
  - Ex : **`messages.fr_FR.yaml`**, **`messages.fr.yaml`**, **`messages.en.xlf`**, …

Quel format de traduction choisir ?

- Symfony recommande le XLIFF
  - Avantage : c'est un format standard dans le domaine de la traduction
  - Inconvénient : très verbeux car basé sur du XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<xliff xmlns="urn:oasis:names:tc:xliff:document:1.2" version="1.2">
    <file source-language="en" target-language="fr" datatype="plaintext" original="file.ext">
        <body>
            <trans-unit id="1">
                <source>One term to translate</source>
                <target>Un terme à traduire</target>
            </trans-unit>
            <!-- ... -->
        </body>
    </file>
</xliff>
```

Quel format de traduction choisir ?

- La communauté utilise en majorité le YAML
  - Avantage : format très synthétique
  - Inconvénient : le format n’est pas standard et la syntaxe est stricte

```
One term to translate: 'Un terme à traduire'
# ...
```

Terme à traduire *versus* code à traduire

- Le terme à traduire peut être rédigé par exemple en anglais
  - Avantage : cela permet d'avoir la source plus facile à lire
  - Inconvénient : si on change le terme, il faut aller le modifier dans tous les fichiers de traduction (à priori un par langue)
- Le terme à traduire peut être un code ➔ préconisé par Symfony !
  - Avantage : on peut en changer la valeur traduite sans avoir à le changer :

```
header.welcome: "Bienvenue sur notre site !"
```

  - Autre avantage si on fait du YAML : regroupement possible :

```
header:
    welcome: "Bienvenue sur notre site !"
    account:
        login: "Se connecter"
        logout: "Se déconnecter"
        dashboard: "Mon compte"
```

Gestion de la locale utilisateur

- Accéder à la locale courante :

```php
public function indexAction(Request $request)
{
    // Récupération de la locale courante
    $locale = $request->getLocale();
    // ...
}
```

- Définir la locale via un paramètre GET spécial « **`_locale`** » :

```php
/**
 * @Route("/{_locale}", name="home")
 */
public function indexAction(Request $request)
{
    // Récupération de la locale transmise en GET
    // Possibilité de restreindre les valeurs possibles (cf. requirements pour la route)
    $locale = $request->getLocale();
    // ...
}
```

- Rendre la locale persistante sur la session utilisateur : cf. doc : symfony.com/doc/current/session/locale_sticky_session.html

Traduction – forcer la locale

- Dans un contrôleur :

```php
public function indexAction(Request $request)
{
    $value = $this->get('translator')->trans(
        'header.welcome', // message à traduire
        array(), // paramètres pour la traduction => vu plus loin dans le cours
        'messages', // domaine de traduction => vu plus loin dans le cours
        'fr_FR' // locale souhaitée pour la traduction
    );
    // ...
}
```

- Dans un template Twig :

```twig
{{ 'header.welcome'|trans({}, 'messages', 'fr_FR') }}

{% trans from 'messages' into 'fr_FR' %}
header.welcome
{% endtrans %}
```

- Attention : à éviter au maximum, préférer la définition de la locale via une des méthodes présentées précédemment

Domaines de traduction

- Un domaine par défaut : « **`messages`** »
- Créer d'autres domaines permet de « cloisonner » les traductions par thématiques (ex : admin, customer, …)
- Les fichiers de traduction sont nommés avec le nom du domaine :
  - messages.fr.yaml, admin.fr.yaml, customer.fr.yaml, …
- Il est obligatoire de préciser le domaine lors d'une traduction sauf si le domaine est « **`messages`** » ➔ cf. exemples précédents
- Dans un template Twig, possibilité de préciser le domaine à utiliser dans tout le template (uniquement le template courant, ne concerne pas les éventuels templates inclus) :

```
{# Dans tout le template excepté ses fils, le domaine de traduction sera "customer" #}
{% trans_default_domain 'customer' %}

{# ... Contenu du template ... #}
```

Traductions avec variables

- Utilisation :

```php
public function indexAction()
{
    // Par convention, entourer les variables par "%" => %my_var% (ce n'est qu'une convention)
    $value = $this->get('translator')->trans('My name is %name%', array('%name%' => 'John'));
    // ...
}
```

```
{{ 'My name is %name%'|trans({'%name%': 'John'}) }}

{% trans with {'%name%': 'John'} %}My name is %name%{% endtrans %}
```

- Définition dans le fichier de traduction (ex en YAML) :

```yaml
'My name is %name%': "Je m'appelle %name%"
```

Gestion des pluriels

- Utilisation :

```
// Les messages dans transChoice sont séparés par un |
// Le premier message est affiché si le second paramètre vaut 0 ou 1
// Sinon, le second message est affiché
$msg = $this->get('translator')->transChoice('There is one product left|There are %count% products left', 5);
```

```
{% transchoice 5 %}There is one product left|There are %count% products left{% endtranschoice %}

{{ 'There is one product left|There are %count% products left'|transchoice(5) }}
```

- Définition dans le fichier de traduction (ex en YAML) :

```
'There is one product left|There are %count% products left': "Il ne reste qu'un seul produit|Il reste %count% produits"
```

Gestion des pluriels – utilisation des intervalles

- Utilisation de la norme ISO 31-11 pour la définition des intervalles :
  en.wikipedia.org/wiki/Interval_(mathematics)#Notations_for_intervals

```
{{ 'my_msg'|transchoice(3) }}
```

```
'my_msg': ']-Inf,0[ Négatif|{0} Vide|{1} Plus qu''un|{2,3} Plus que 2 ou 3|]3,Inf[ Encore %count%'
```

- Cf. doc pour les détails :
  symfony.com/doc/current/components/translation/usage.html#explicit-interval-pluralization
- Remarque : toutes les langues n'ont pas l'obligation de définir les mêmes intervalles
  - Les règles sur les pluriels ne sont d'ailleurs pas les mêmes d'une langue à l'autre

Synthèse des paramètres possibles

```php
$value = $this->get('translator')->trans(
    'My name is %name%', // Message
    array('%name%' => 'John'), // Paramètres
    'messages', // Domaine
    'fr_FR' // Locale
);
```

```php
$value = $this->get('translator')->transChoice(
    'my_msg', // Message
    5, // Valeur permettant de choisir le message
    array(), // Paramètres
    'messages', // Domaine
    'fr_FR' // Locale
);
```

```twig
{{ 'My name is %name%'|trans({'%name%': 'John'}, 'messages', 'fr_FR') }}

{{ 'my_msg'|transchoice(5, {}, 'messages', 'fr_FR') }}
```

```twig
{% trans with {'%name%': 'John'} from 'messages' into 'fr_FR' %}My name is %name%{% endtrans %}

{% transchoice 5 with {} from 'messages' into 'fr_FR' %}my_msg{% endtranschoice %}
```

Traduction en base de données ?

- N’est pas fourni de base dans Symfony, car lié à Doctrine

- Une extension Doctrine permet de gérer les traductions en base de données :
  - github.com/stof/StofDoctrineExtensionsBundle
  - Utiliser l’extension « Translatable » ➔ voir cours sur les extensions Doctrine

Debug : Activer les logs liés à la traduction

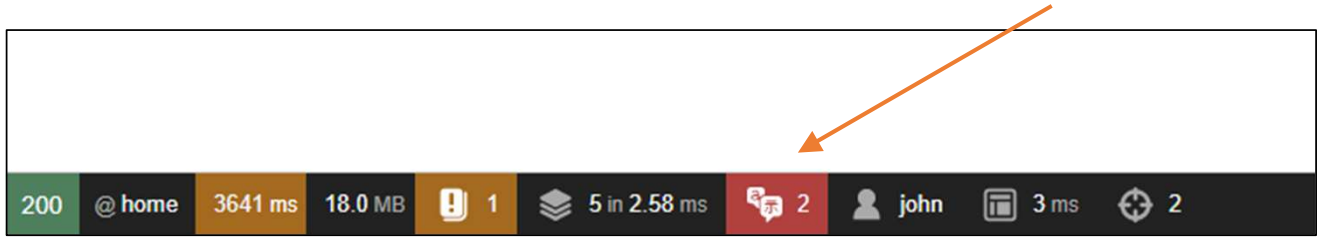
- A activer dans **`config/packages/translation.yaml`** :

```yaml
framework:
    translator:
        logging: true
```

- Exemple de logs générés dans **`var/log/dev.log`** :

```
[201x-yy-zz …] translation.DEBUG: Translation use fallback catalogue. {"id":"My name is %name%","domain":"messages","locale":"fr_FR"} []
[201x-yy-zz …] translation.WARNING: Translation not found. {"id":"toto","domain":"messages","locale":"fr"} []
```

Debug : Utiliser le profiler



Debug : Utiliser la console

- Afficher les traductions d'un catalogue :

```
php bin/console debug:translation fr
```

```
---------- ---------- ------------------------------------------------------------ ----------------------------------------
 State      Domain     Id                                                           Message Preview (fr)
---------- ---------- ------------------------------------------------------------ ----------------------------------------
            messages   One term to translate                                        Un terme à traduire
            messages   My name is %name%                                            Je m'appelle %name%
  unused    messages   There is one product left|There are %count% products left   Il ne reste qu'un seul produit|Il res...
  unused    messages   my_msg                                                       ]-Inf,0[ Négatif|{0} Vide|{1} Plus qu...
  unused    messages   header.welcome                                               Bienvenue sur notre site !
  unused    messages   header.account.login                                        Se connecter
  unused    messages   header.account.logout                                       Se déconnecter
  unused    messages   header.account.dashboard                                    Mon compte
---------- ---------- ------------------------------------------------------------ ----------------------------------------
```

- Générer les traductions dans une langue :

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

```
'My name is %name%': '__My name is %name%'
my_msg: __my_msg
```

Ne prend en compte que les messages présents dans les templates
Ajoute « __ » en préfixe pour facilement identifier les traductions à reprendre (possibilité de définir un autre préfixe)
Plusieurs options disponibles, utiliser **`--help`**

TP i18n

- Paramétrer Symfony pour activer 2 langues (français et anglais)

- Pour pouvoir accéder à une langue ou l'autre, plusieurs possibilités. Par exemple :
  - Modifier toutes les routes pour ajouter la locale en paramètre de la route (paramètre « _locale »)
  - Stocker la locale courante en session pour éviter de changer toutes les routes
    ➔ rappel : symfony.com/doc/current/session/locale_sticky_session.html

- Prévoir un sélecteur de langue dans le menu principal de votre site

- Traduire tous les termes déjà définis dans les contrôleurs et les templates dans les 2 langues