Symfony et Internationalisation (i18n)

Introduction

Je vous donne ici une recette simple à appliquer. Elle ne couvre pas tout le sujet, loin de là, mais elle est facile à déployer et donne de bons résultats.

Traduction des chaînes de caractères d'un programme

Dans le fichier .env ajoutez une nouvelle variable d'environnement appelée « LOCALE ». Donnez-lui un code pour la valeur de la langue souhaitée (par exemple LOCALE=en)

Créez une classe **Traducteur** dans le fichier …/src/Classe/Traducteur.php.

Dans ce fichier on utilisera (« use ») les classes **Translator** et **PhpFileLoader**:

```
<?php
// . . . /src/Classe/Traducteur.php


namespace App\Classe;

use Symfony\Component\Translation\Translator;
use Symfony\Component\Translation\Loader\PhpFileLoader;

class Traducteur
{
    private $trad;

    public function __construct()
    {
       // instanciation d'un objet de la classe Translator
       $this->trad = new Translator($_ENV['LOCALE']);

       //  On indique la méthode qu'on utilisera pour charger le catalogue des
       //  chaînes de caractères du language souhaité
       //  Ici on choisit la méthode fichier PHP
       $this->trad->addLoader('php', new PhpFileLoader());

       // On construit le nom du fichier à charger
       $fichierTrad =
__DIR__ . '/../../translations/messages.' . $_ENV['LOCALE'] . '.php';

       // On teste l'existence du fichier. S'il existe on le charge dans notre
       // objet Translator
       if (file_exists($fichierTrad))
           $this->trad->addResource('php', $fichierTrad , $_ENV['LOCALE']);
    }
```

```
    //-------------------------------------------------------------
    // Fonction qui servira à aller chercher la bonne chaîne dans la langue
    // souhaitée à l'aide d'une clé
    //-------------------------------------------------------------
    public function TR($clé)
    {
        return $this->trad->trans($clé);
    }
}
```

La correspondance entre les chaînes de caractères du langage désiré se fera par l'entremise de « clés ». Ces clés seront identiques dans toutes les langues mais pointeront sur des chaînes de caractères différentes et adaptées à chaque LOCALE.

Chaque langue supportée aura son propre fichier catalogue dans lequel on établira la correspondance entre les clés et la chaîne adaptée.

Ces « fichiers catalogue » seront déposés dans le dossier **translations** à la racine du projet. Dans l'exemple suivant on crée trois fichiers pour trois langues supportées (anglais, français et espagnol):

Messages.es.php

Messages.fr.php

Messages.en.php

Chacun de ces fichiers aura le contenu respectif suivant :

```
<?php
// . . . /translation/messages.es.php
return [
       'connexionErreur' => 'Esta conetivo moelas',
       'connexionOK' => 'Benvinito',
       'deconnexion' => 'A la ciudad',
        'creation' => 'generatos moy bien',
       'InfoModif' => 'Informations bueno modifica',
       'MdePModif' => 'Secreto modo modifica',
       'MdeP_Actuel_Incorrect' => 'Secreto modo moelas',
  ];
```

```
<?php
// . . . /translation/messages.fr.php
return [
       'connexionErreur' => 'Erreur de connexion',
       'connexionOK' => 'Bienvenue',
       'deconnexion' => 'Au revoir',
        'creation' => 'créée avec succès',
        'InfoModif' => 'Informations modifiées',
       'MdePModif' => 'Mot de passe modifié',
       'MdeP_Actuel_Incorrect' => 'Mot de passe actuel incorrect',
  ];
```

```
<?php
// . . . /translation/messages.en.php
return [
      'connexionErreur' => 'Error while connecting',
      'connexionOK' => 'Welcome',
      'deconnexion' => 'See you soon',
       'creation' => 'create successfully',
      'InfoModif' => 'informations successfully modified',
      'MdePModif' => 'Password successfully modified',
      'MdeP_Actuel_Incorrect' => 'Actual password is wrong',
  ];
```

À tous les endroits où l'on affiche du texte à traduire on appliquera la technique suivante:

```
#[Route('/creationEntreprise')]
public function creationEntreprise(ManagerRegistry $doctrine,
                                   Request $request): Response
{
  //Création de l'entité
  $entreprise = new Entreprise;
  .  . .
   $trad = new Traducteur;

   $this->addFlash('succes',
                "Entreprise " . $entreprise->getNom() . $trad->TR("creation");
. . .
 }
```

Nous venons d'effleurer le sujet de l'adaptation culturelle d'un site Symfony. Il y a encore une foule de détails à explorer dans ce domaine. Pour un complément d'info voici la référence:

https://symfony.com/doc/current/translation.html

Traduction des chaînes de caractères d'une base de données

Pour traduire le contenu d'une base de données on peut ajouter autant de colonnes pour les champs textes que de langues supportées. Imaginons un champ « description » d'une table « produit ». On pourrait tripler cette colonne et les appeler ainsi :

description_fr,

description_en,

description_es

Quand on fait la requête d'accès on regarde la LOCALE de la configuration et on va chercher la bonne colonne. On aurait ainsi un seul endroit à modifier, le fichier .env, et notre site devient multilingue!