Créer un site web multilingue avec Laravel Localization en utilisant un middleware

1. Introduction :

Les fonctionnalités de localisation de Laravel offrent un moyen pratique de récupérer des chaînes dans différentes langues, ce qui vous permet de prendre facilement en charge plusieurs langues dans votre application.

Etape 1 : configuration

Par défaut, la langue locale de notre application est l'anglais ou **en**.

Vous pouvez la trouver dans le fichier **config** >> **app.php**.

```
|--------------------------------------------------------------------------
|
| The application locale determines the default locale that will be used
| by the translation service provider. You are free to set this value
| to any of the locales which will be supported by the application.
|
*/

'locale' => 'fr',

/*
|--------------------------------------------------------------------------
| Application Fallback Locale
|--------------------------------------------------------------------------
|
| The fallback locale determines the locale to use when the current one
| is not available. You may change the value to correspond to any of
| the language folders that are provided through your application.
|
*/

'fallback_locale' => 'en',
```

Ici, la locale est définie sur **fr**.

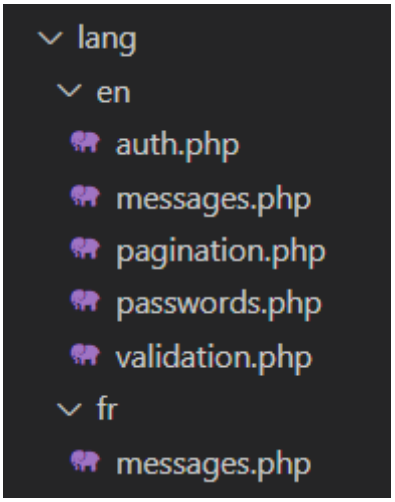
Maintenant, si vous vérifiez le dossier **lang**, il y a un dossier appelé **en**, dans lequel il y a quelques fichiers comme auth.php ,messages.php , pagination.php, passwords.php, validation.php.

Ces fichiers sont donc des fichiers de traduction. Mais, tout d'abord, vérifions le fichier **lang/fr/messages.php**.

```
<?php
return [

    /*
    |--------------------------------------------------------------------------
-
    | Pagination Language Lines
    |--------------------------------------------------------------------------
-
    |
    | The following language lines are used by the paginator library to build
    | the simple pagination links. You are free to change them to anything
    | you want to customize your views to better match your application.
    |
    */

    'name required' => 'nom est obligatoire',
    'phone integer' => 'téléphone non valide',
    'welcome' => 'Bienvenue dans notre application',
    'home' => 'Acceuil',
    'about' => 'A propos de nous',
    'title' => 'Comment créer un site web multilingue dans Laravel',
    'Select Language'=>'Sélectionner une langue',

];
```

Ce fichier renvoie donc un tableau contenant une paire clé-valeur. Vous pouvez donc définir n'importe quelle clé, mais les valeurs dépendent de votre langue.

Le nom du dossier racine d'un fichier **app.php** est en, donc la traduction des instructions de ce fichier est en anglais ou en, si le dossier racine est **fr**, ce qui signifie une traduction en français, le fichier **app.php** pourrait ressembler à ce qui suit.



Étape 2 : Création de fichiers de traduction

D'accord, nous allons maintenant passer une langue. L'anglais est déjà là.

1- French

Créons donc un dossier **fr** dans le dossier **lang**.

Maintenant, créez un fichier appelé **messages.php** et ajoutez le code suivant dans les deux dossiers.

Pour le fichier **fr>> messages.php**

```
<?php
// messages.php
return [
     'welcome' => 'Bienvenue dans notre application',
    'home' => 'Acceuil',
    'about' => 'A propos de nous',
    'title' => 'Comment créer un site web multilangage dans Laravel 9',
    'Select Language'=>'Sélectionner une langue',
];
```

Pour le fichier **en>> messages.php**

```
<?php
// messages.php
return [
```

```
    'welcome' => 'Welcome to our application!',
    'home' => 'Home',
    'about' => 'About',
    'title' => 'How To Create Multi Language Website In Laravel 9',
    'Select Language'=>'Select Language',
];
```

Étape 3 : Configurer la vue.

Tout d'abord, créez un dossier "layouts" dans le dossier "resources >> views", et dans ce dossier "layouts", créez un fichier "master.blade.php" et ajoutez le code suivant.

```
<html>
 <head>
 <title>@yield('title')</title>
 @vite('resources/js/app.jsx')
 </head>
 <body>

    <div class="container">
        <div class="row" style="text-align: center;margin-top: 40px;">
            <h2>{{__('messages.title')}}</h2><br>
            <div class="col-md-2 col-md-offset-3 text-right">
                <strong>{{__('messages.Select Language')}}: </strong>
            </div>
            <div class="col-md-2">

                <a
href="{{route(\Illuminate\Support\Facades\Route::currentRouteName(), ['locale'
=> 'en'])}}"> <img src="{{asset('storage/english.png')}}" width="30px"
height="20x"> English</a>
                <a
href="{{route(\Illuminate\Support\Facades\Route::currentRouteName(), ['locale'
=> 'fr'])}}"><img src="{{asset('storage/french.png')}}" width="30px"
height="20x">French</a>

            </div>
          <nav>
            <a href="{{route('home')}}">{{__('messages.home')}}</a>
            <a href="{{route('about')}}">{{__('messages.about')}}</a>
          </nav>
        </div>
    </div>

 <div class="container">
@yield('content')
```

```
 </div>
 </body>
</html>
```

Ouvrez maintenant votre fichier **home.blade.php**, qui se trouve dans le dossier resources >> views>>pages, et placez le code ci-dessous :

```
@extends('layouts.master')
@section('title')
    Home page
@endsection
@section('content')

<h1>{{__('messages.home')}}</h1>
{{__('messages.welcome')}}



@endsection
```

Ouvrez maintenant votre fichier **about.blade.php**, qui se trouve dans le dossier resources >> views>>pages, et placez le code ci-dessous :

```
@extends('layouts.master')
@section('title')
    About page
@endsection
@section('content')

<h1>{{__('messages.about')}}</h1>
{{__('messages.welcome')}}



@endsection
```

Étape 4 : Préfixer les routes avec la locale

Ensuite, nous allons ajouter **Route::prefix()** pour toutes les URLs possibles - ainsi toutes les pages du projet auront un préfixe de **/[locale]/[any_page]**. Voici ce que cela donne dans le fichier **routes/web.php** :

```
Route::prefix('{locale}')
    ->where(['locale' => '[a-zA-Z]{2}'])
    ->middleware('locale')
```

```
    ->group(function () {
 Route::get('/home',[LocalizationController::class,'index'])->name('home');
Route::get('/about',[LocalizationController::class,'about'])->name('about');

});

Route::get('/', function () {
    return redirect()->route('home',app()->getLocale());
});
```

Toutes les routes ci-dessus ont été pré-générées, nous les avons simplement placées dans notre groupe **Route::prefix().**

Ce groupe couvrira des URLs telles que /en/home ou /en/about.

Maintenant, à des fins de validation, définissons le **{locale}** pour qu'il ne contienne que deux lettres, comme **en** ou **fr** ou **de**. Nous ajoutons une règle basée sur des expressions régulières à l'intérieur du groupe :

Étape 5 : Définir la localisation de l'application à l'aide d'un middleware

Les gens pensent souvent que les classes Middleware servent à restreindre certains accès. Mais il est possible d'effectuer d'autres actions à l'intérieur de ces classes. Dans notre exemple, nous allons définir **app()->setLocale()**.

```
php artisan make:middleware SetLocale
```

Cette commande générera app/Http/Middleware/SetLocale.php, où nous devons ajouter seulement deux lignes de code :

```
class SetLocale
{
    public function handle(Request $request, Closure $next)
    {
        app()->setLocale($request->segment(1));

        URL::defaults(['locale' => $request->segment(1)]);

        return $next($request);
    }
```

```
}
```

La variable **$request->segment(1)** contiendra notre partie {locale} de l'URL, de sorte que nous définissons la locale de l'application sur toutes les requêtes. Et **URL::defaults()** définira que chaque route aura la locale par défaut.

Bien sûr, nous devons enregistrer cette classe dans app/Http/Kernel.php dans le tableau $routeMiddleware :

```
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    //
    'locale' => \App\Http\Middleware\SetLocale::class,
```

Enfin, nous appliquerons ce middleware à l'ensemble du groupe de locales que nous avons créé ci-dessus :

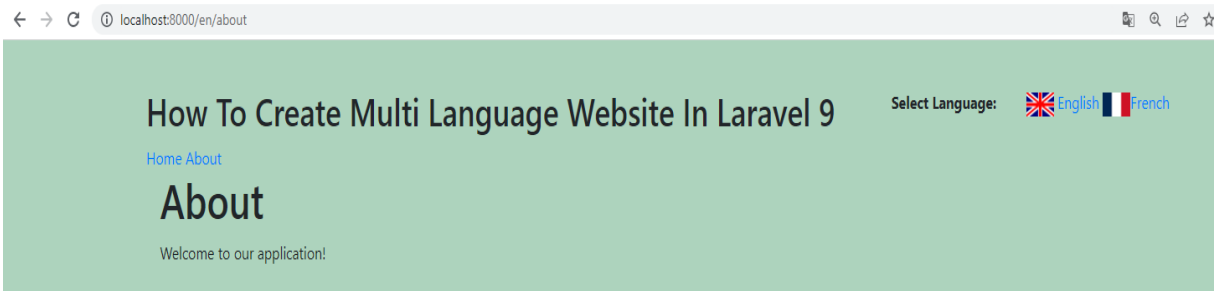
```
Route::prefix('{locale}')
    ->where(['locale' => '[a-zA-Z]{2}'])
    ->middleware('locale')
    ->group(function () {
    // ...
```

Étape 6 : Redirection automatisée de la page d'accueil

Nous devons ajouter une autre ligne à **routes/web.php** - pour rediriger l'utilisateur de la page d'accueil non localisée vers la page d'accueil /fr/. Cette route sera en dehors de la Route::prefix() que nous avons créée ci-dessus :

```
Route::get('/', function () {
    return redirect()->route('home',app()->getLocale());
}) ;
```

Ceci redirigera l'utilisateur de votredomaine.com vers votredomaine.com/en/home, qui affichera la page d'accueil par défaut.


localhost:8000/en/about
How To Create Multi Language Website In Laravel 9
Select Language:
English
French
Home About
About
Welcome to our application!