

Laravel - Localization

Topics : [Laravel](#)

Written on [December 25, 2023](#)

Localization in Laravel allows you to provide support for multiple languages in your application. This feature enables you to display content in different languages based on the user's preferences. Laravel provides a straightforward way to manage localization through language files and the `trans` helper function. Here's a guide on how to use localization in Laravel:

1. Configuration:

Ensure that the `config/app.php` file has the `locale` set to the default language you want for your application.

```
'locale' => 'en',
```

2. Language Files:

Create language files for each supported language in the `resources/lang` directory. Laravel uses a nested array structure to organize translations. For example:

- `resources/lang/en/messages.php` for English:

```
return [  
    'welcome' => 'Welcome to our website!',  
    'greeting' => 'Hello, :name!',  
];
```

- `resources/lang/es/messages.php` for Spanish:

```
return [  
    'welcome' => '¡Bienvenido a nuestro sitio web!',  
    'greeting' => 'Hola, :name.',  
];
```

3. Using Translations:

In your Blade views or controllers, use the `trans` function or `@lang` directive to fetch translations.

```
// Using trans() function
echo trans('messages.welcome');
```

```
// Using @lang directive in Blade
{{ __('messages.welcome') }}
```

4. Pluralization:

Laravel provides support for pluralization in translations.

```
// resources/lang/en/messages.php
return [
    'apples' => '{0} No apples|{1} :count apple|[2,*] :count apples',
];
```

Use it in your code:

```
echo trans_choice('messages.apples', 5, ['count' => 5]);
// Output: 5 apples
```

5. Replacing Variables:

You can replace variables within translation strings using the `:variable` syntax.

```
// resources/lang/en/messages.php
return [
    'greeting' => 'Hello, :name!',
];
```

```
echo trans('messages.greeting', ['name' => 'John']);
// Output: Hello, John!
```

6. Changing the Locale Dynamically:

You can change the application's locale dynamically during the request lifecycle.

```
app()->setLocale('es');
```

7. Falling Back to the Default Locale:

If a translation key is not found in the requested language, Laravel falls back to the default locale.

8. Detecting User's Preferred Language:

You can use middleware to detect and set the user's preferred language based on their preferences or browser settings.

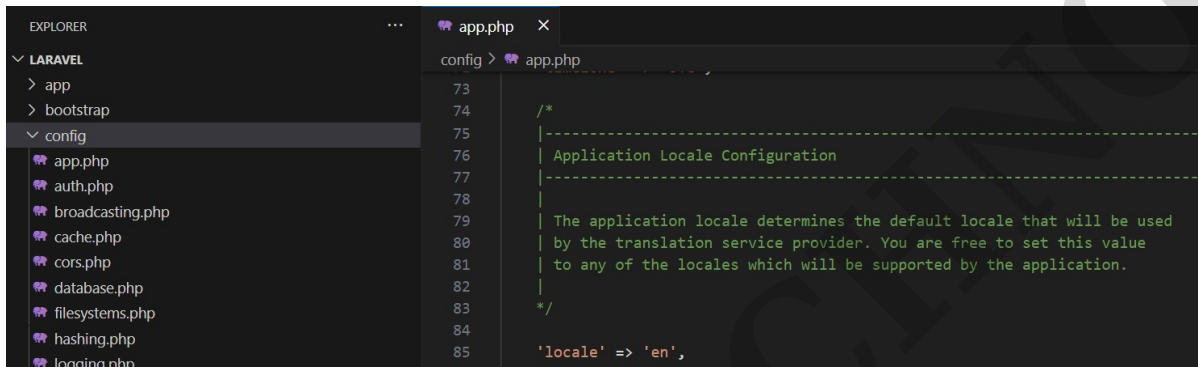
9. Language Files for Validation Messages:

Laravel also allows you to customize validation error messages using language files. These can be placed in the `resources/lang` directory under a directory named `en` or `es`, etc.

10. URL Localization:

Laravel provides a convenient way to generate localized URLs using the `URL::to` method.

```
$url = URL::to('/page', [], 'es');  
// Output: http://example.com/es/page
```



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the Laravel project structure, with the `config` directory expanded. The code editor shows the `app.php` configuration file, with the following content:

```
73  
74  
75  
76 |-----  
76 | Application Locale Configuration  
77 |-----  
78  
79 | The application locale determines the default locale that will be used  
80 | by the translation service provider. You are free to set this value  
81 | to any of the locales which will be supported by the application.  
82 |-----  
83  
84  
85 'locale' => 'en',
```

© Copyright **Aryatechno**. All Rights Reserved. Written tutorials and materials by [Aryatechno](#)