Multilingua

Sidrit Trandafili

@sidis405

@strandafili

- Sviluppatore PHP 15+ anni
- Laravel da 4 anni
- Back-end
  - php
  - java
  - node
  - python
- Front-end
  - VueJS
  - Vanilla JS
  - TailwindCSS
  - Sass
- Devops + Sysops da 13 anni

Considerazioni Multilingua e localization

Multilingua

“ Laravel ha delle funzionalità di gestione traduzioni (localisation) che permettono in modo semplice e veloce di visualizzare traduzioni in varie lingue, permettendoci di aggiungere versioni multilingua al nostro progetto.

“ I file delle traduzioni si trovano in resources/lang. Dentro questa cartella devono essere messe sottocartelle per ogni singola lingua supportata dall'applicazione:

```
/resources
    /lang
        /en
            messages.php
        /it
            messages.php
        /fr
            messages.php
```

Ogni file lingua deve contenere un array con chiavi e traduzione:

```php
<?php

return [
    'welcome' => 'Benvenuti al nostro sito'
];
```

Multilingua

Configurare la Locale.

La lingua di default della nostra applicazione si trova nel file config/app.php. Possiamo modificarla in ogni momento. Possiamo anche cambiarla a runtime tramite la funzione setLocale della Facade App:

```
Route::get('{locale}/home', function ($locale) {
    App::setLocale($locale);

    //
});
```

Per verificare la locale attiva:

```
$locale = App::getLocale();

if (App::isLocale('it')) {
    //
}
```

Multilingua

Stampare le stringhe della lingua attiva

Possiamo reperire stringhe tradotte nei nostri file traduzione tramite la funzione helper __ (due underscore). Questa funzione accetta due parametri. Il primo è la chiave della stringa tradotta nel file resources/lang/messages.php della nostra lingua impostata:

```
{{ __('messages.welcome') }}

@lang('messages.welcome')
```

Se la stringa non viene trovata, questa funzione restituisce la chiave.
È possibile passare parametri a queste stringhe tradotte:

```
<?php

return [
    'welcome' => 'Ciao :utente'
];
```

```
{{ __('messages.welcome', ['utente' => 'sid'])}}
```

Multilingua

Progetto: impostiamo un sistema multilingua per il nostro progetto

Struttureremo l'url nel seguente modo: http://esempio.dev/{lingua}

Imposteremo inglese e italiano e in base al parametro :lingua cambieremo la lingua

dell'applicazione stampando una stringa proveniente da un file traduzioni di quella lingua

Creiamo un controller e un model velocemente (se non presenti)

```
php artisan make:controller PostsController -r -m Post
```

Impostiamo una route

```
Route::get('/', 'PostsController@index');
```

Nel metodo index del PostsController inseriamo la seguente

```
return __('messages.welcome', ['utente' => 'sid', 'locale' => app()->getLocale()]);
```

Multilingua

Progetto: impostiamo un sistema multilingua per il nostro progetto

```
/resources
    /lang
        /en
            messages.php
        /it
            messages.php
```

Creiamo un file messages.php per l'inglese e uno per l l'italiano

```php
<?php // resources/lang/it/messages.php

return [
    'welcome' => 'Ciao :utente! Locale: :locale'
];
```

```php
<?php // resources/lang/en/messages.php

return [
    'welcome' => 'Hello :utente! Locale: :locale'
];
```

Multilingua

Progetto: impostiamo un sistema multilingua per il nostro progetto

Creiamo un middleware per gestire la lingua

```
php artisan make:middleware Lingua // app/Http/Middleware/Lingua.php
```

```
public function handle($request, Closure $next)
{
    // Ci assicuriamo che la locale esiste
    $locale = $request->segment(1);

    //se non esiste mettiamo quella di default e reidirizziamo
    if (!array_key_exists($locale, config('app.locales'))) {
        $segments = $request->segments();
        $segments[0] = config('app.fallback_locale');

        return redirect(implode('/', $segments));
    }

    //La lingua è stata trovata. La impostiamo

    app()->setLocale($locale);

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

Multilingua

Progetto: impostiamo un sistema multilingua per il nostro progetto

Modifichiamo il provider di laravel in modo da prefissare tutte le nostre routes con la locale (it, o en)

```
protected function mapWebRoutes()
{
    $locale = request()->segment(1);
    app()->setLocale($locale);

    Route::middleware('web')
         ->namespace($this->namespace)
         ->prefix($locale)
         ->group(base_path('routes/web.php'));
}
```

Estraiamo il primo segmento dell'url e lo usiamo per definire la locale del sistema

Prefissiamo tutte le web routes con la stringa della locale

Multilingua

Risultato