

Course : COMP6821001 – Web
Programming
Effective Period : 2023

Localization

Session 10

Learning Outcomes

- **LO-3 : Identify a proper web programming technic to build web based application**
- **LO-4 : Create web-based applications using PHP Framework to solve problems that occur in the IT field**

Outline

- Definition of Localization
- Dynamic Localization Setting
- The resources lang folder
- Lang commands and functions
- Localization with Route Parameters
- Make 2 Language Localization
- Localization for Form Validation

Definition of Localization

- **Localization** is a programming term that refers to how to translate an application into a particular language. In this material, the application in question is in the form of a website.
- For example, some websites in Indonesia have pages in Indonesian such as www.traveloka.com/id-id/, as well as pages in English at www.traveloka.com/en/. These two websites can be made separately, one team makes an Indonesian website and the other team makes an English website.
- But the problem is, if the Indonesian team wants to add a new menu, the English team must also match the structure of the program code for the menu (including HTML, CSS, PHP, etc.). Things like this are certainly not efficient, it would be better if the web structure could be shared for both websites. This is what we can do from Laravel's **Localization** feature.

Definition of Localization

- The working principle of the localization process in Laravel is as follows:
- First, create a kind of "dictionary" containing words or sentences for each language. This dictionary is stored in a separate file but uses the same key name (in the form of an associative array). For example, we want to design a **"Daftar" / "Register"** button, then we can create a file called indonesia.php which contains "button" => "Daftar", and an english.php file which contains an array of "buttons" => "Register".
- Then, in the view write a <button> tag that refers to the name of the key that has been prepared (instead of writing the text directly). Like the following example:

```
<button>{{ __('tombo1') }}</button>
```

- The __() sign, which is a double underscore character, is a special function in the blade that serves to display the localization result text.

The resources \ lang folder

- The "dictionary" location of localization is in the resources\lang folder. Default Laravel, this folder already contains 1 en folder in which there are 4 files.

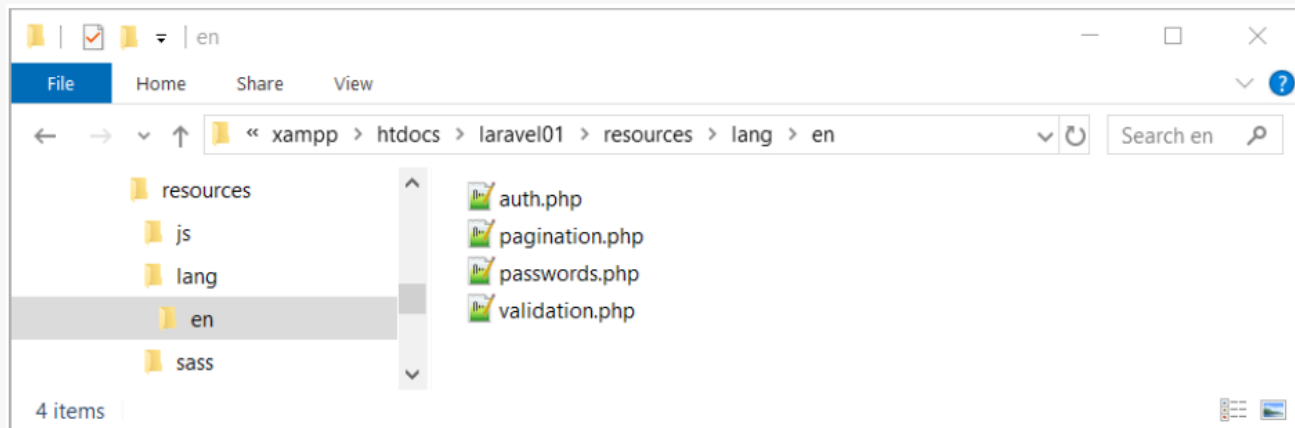


Fig 1. Contents folder resources\lang\en.

- The name of the **en** folder is the language code used by Laravel. If you want to create localization for another language, place it in a separate folder with the appropriate language code. For example, for Indonesian, the code is **id** (to be created).

The resources \ lang folder

- In the **en** folder, here is an example of one of the files, namely **auth.php**:

```
resources/lang/en/auth.php

1  <?php
2
3  return [
4
5      /*
6      |-----
7      | Authentication Language Lines
8      |-----
9      |
10     | The following language lines are used during authentication for various
11     | messages that we need to display to the user. You are free to modify
12     | these language lines according to your application's requirements.
13     |
14     */
15
16     'failed' => 'These credentials do not match our records.',
17     'throttle' => 'Too many login attempts. Please try again in :seconds ...',
18
19 ];
```

Fig 2. auth.php.

The resources \ lang folder

- This file in Figure 2 basically only contains an associative array, which is of the following format:

```
return [  
    'key1' => 'value1',  
    'key2' => 'value2',  
    'key3' => 'value3',  
    //...  
];
```

- The format of writing the dictionary for localization as later in the view, this key will be accessed.

Lang commands and functions

- In this topic we will try the `__()` function and the Lang command. For example, please create a new file called **test.php** in the `resources\lang\en` folder, then fill it with the following code:

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

return [
    'title' => 'Learn Laravel',
];
```

- Here we create a dictionary with the key 'title' which when accessed will display the text 'Learn Laravel'.

Fig 2. auth.php.

Lang commands and functions

- Then please open the **welcome.blade.php** view, which is Laravel's default view. Scroll to the opening `<body>` tag and add the line below:

```
<h1 style="text-align: center"> {{ __('test.title') }}</h1>
```

- Save, and run it in a web browser and the result, the text 'Learn Laravel' appears as written in the dictionary.
- Function `__()` is a blade specific function for displaying localization text. As the argument of this function is the combination of the localization filename without the .php suffix, and the name of the dictionary key you want to access.
- If you use the `@lang` command, you don't need to add curly braces twice like the `__()` function.

Make 2 Language Localization

- In the previous topic, only accessing localization dictionaries for 1 language only. In this topic, we will try to access 2 languages, namely English and Indonesian.
- For this example, we will translate the student registration form as we used in the previous topic (in Session 8). Please create a file called **form.php** in the resources\lang\en folder and fill it with the following

```
resources/lang/en/form.php
1  <?php
2
3  return [
4      'judul' => 'Student Registration',
5      'input' => [
6          'nim' => 'NIM',
7          'nama_lengkap' => 'Full Name',
8          'email' => 'Email',
9          'jenis_kelamin' => 'Gender',
10         'pilihan_jenis_kelamin' => [
11             'laki_laki' => 'Male',
12             'perempuan' => 'Female',
13         ],
14         'jurusan' => 'Major',
15         'alamat' => 'Address',
16         'tombol' => 'Register',
17     ]
18 ];
```

Fig 3. localization dictionary

Make 2 Language Localization

- As an exercise, please try modifying the view **form-pendaftaran.blade.php** from **session 8**, then add the localization code for the form title and input the form name from the **form.php** file above in Fig 3.

```
12 <div class="container pt-4 bg-white">
13   <div class="row">
14     <div class="col-md-8 col-xl-6">
15       <h1>{{ __('form.judul') }}</h1>
16       <hr>
```

```
12 <div class="container pt-4 bg-white">
13   <div class="row">
14     <div class="col-md-8 col-xl-6">
15       <h1>Pendaftaran Mahasiswa</h1>
16       <hr>
```

```
<div class="form-group">
  <label for="nim">{{ __('form.input.nim') }}</label>
```

```
<div class="form-group">
  <label for="nim">NIM</label>
```

Fig 4. The image on the left adds the Localization code

Make 2 Language Localization

- Create a route to access the "form-registration" view with the following code:

routes/web.php

```
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\MahasiswaController;
5
6  Route::get('/form-pendaftaran', [MahasiswaController::class, 'formPendaftaran']);
7  Route::post('/proses-form', [MahasiswaController::class, 'prosesForm']);
```

Fig 5. Route for access view form-pendaftaran

- The first route is to display the form, and the second route is to process the form input. Since we don't have a MahasiswaController yet, please create it from the command:

```
php artisan make:controller MahasiswaController
```

Make 2 Language Localization

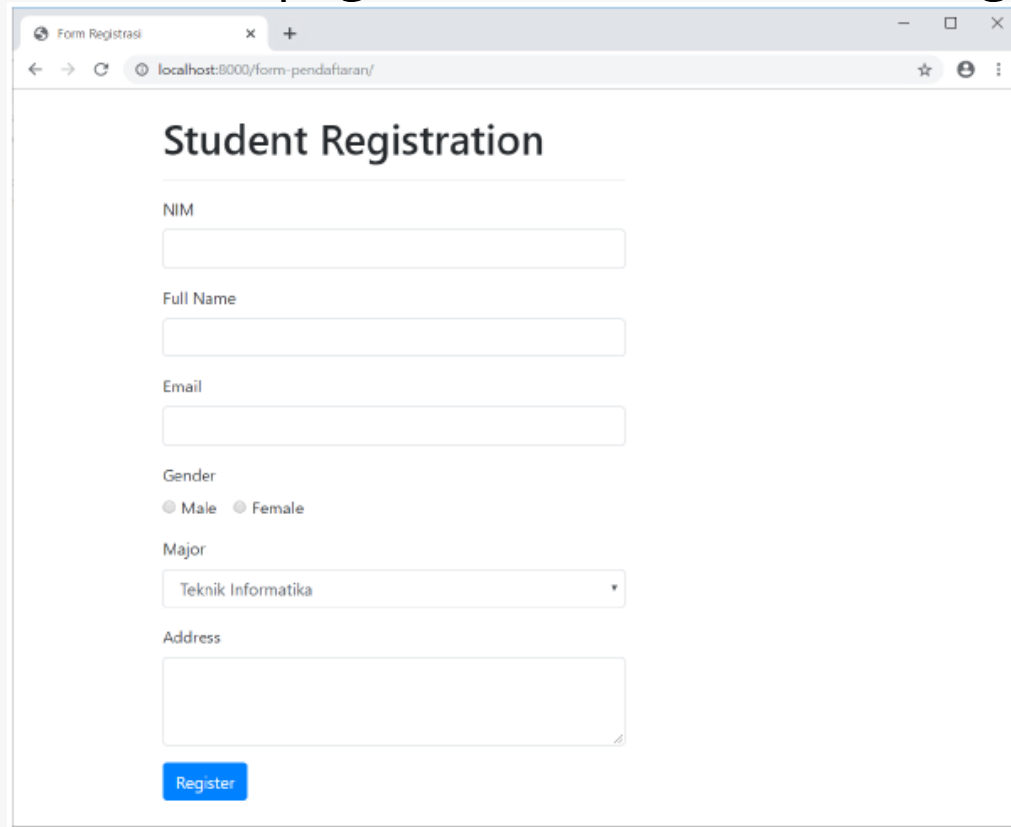
- Here is the contents of this controller:

```
app/Http/Controllers/MahasiswaController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class MahasiswaController extends Controller
8  {
9      public function formPendaftaran()
10     {
11         return view('form-pendaftaran');
12     }
13
14     public function prosesForm(Request $request)
15     {
16         $validateData = $request->validate([
17             'nim'          => 'required|size:8',
18             'nama'         => 'required|min:3|max:50',
19             'email'        => 'required|email',
20             'jenis_kelamin' => 'required|in:P,L',
21             'jurusan'      => 'required',
22             'alamat'       => '',
23         ]);
24
25         dump($validateData);
26     }
27 }
```

Fig 5. Content of MahasiswaController.php

Make 2 Language Localization

- Please access the page localhost:8000/form-registration/.



The screenshot shows a web browser window with the title 'Form Registrasi'. The address bar displays 'localhost:8000/form-pendaftaran/'. The main content area is titled 'Student Registration' and contains the following fields and controls:

- NIM:
- Full Name:
- Email:
- Gender: ☐ Male ☐ Female
- Major:
- Address:
- Register:

Fig 6. The form appears in English.

Make 2 Language Localization

- It can be seen that the title of the form and the name of the input have appeared in English. Now, how to make the Indonesian version?
- Please create a new folder with the name **id** in `resources\lang\`, then in this folder create a **form.php** file with the following code:

```
resources/lang/id/form.php
1  <?php
2
3  return [
4      'judul' => 'Pendaftaran Mahasiswa',
5      'input' => [
6          'nim' => 'NIM',
7          'nama_lengkap' => 'Nama Lengkap',
8          'email' => 'Email',
9          'jenis_kelamin' => 'Jenis Kelamin',
10         'pilihan_jenis_kelamin' => [
11             'laki_laki' => 'Laki-Laki',
12             'perempuan' => 'Perempuan',
13         ],
14         'jurusan' => 'Jurusan',
15         'alamat' => 'Alamat',
16         'tombol' => 'Daftar',
17     ]
18 ];
```

Fig 7. Localization code for id.

Make 2 Language Localization

- The next step is to change Laravel's localization settings. To do this, open the config\app.php file, then look for the line: 'locale' => 'en'. So that we can use localization in the language indonesia, exchange the value

```
72  ... /*
73  ... |-----
74  ... | Application Locale Configuration
75  ... |-----
76  ... |
77  ... | The application locale determines the default locale that will be used
78  ... | by the translation service provider. You are free to set this value
79  ... | to any of the locales which will be supported by the application.
80  ... |
81  ... */
82
83  ... 'locale' => 'en',
```




Fig 8. Locale settings in app.php

- Save the app.php file, then reopen the form-pendaftaran:

Dynamic Localization Setting

- The localization setting of the config/app.php file is global and must be done manually.
- But Laravel also allows us to dynamically set localization, even displaying a different language for each view. For this purpose, we need to access the static method `App::setLocale()` from within the controller.
- First, create 2 new routes to access the registration form:

routes/web.php

```
1 Route::get('/form-pendaftaran/id', [MahasiswaController::class,  
2   'formPendaftaranId']);  
3  
4 Route::get('/form-pendaftaran/en', [MahasiswaController::class,  
5   'formPendaftaranEn']);
```

Fig 9. Routes for access registration form page

Dynamic Localization Setting

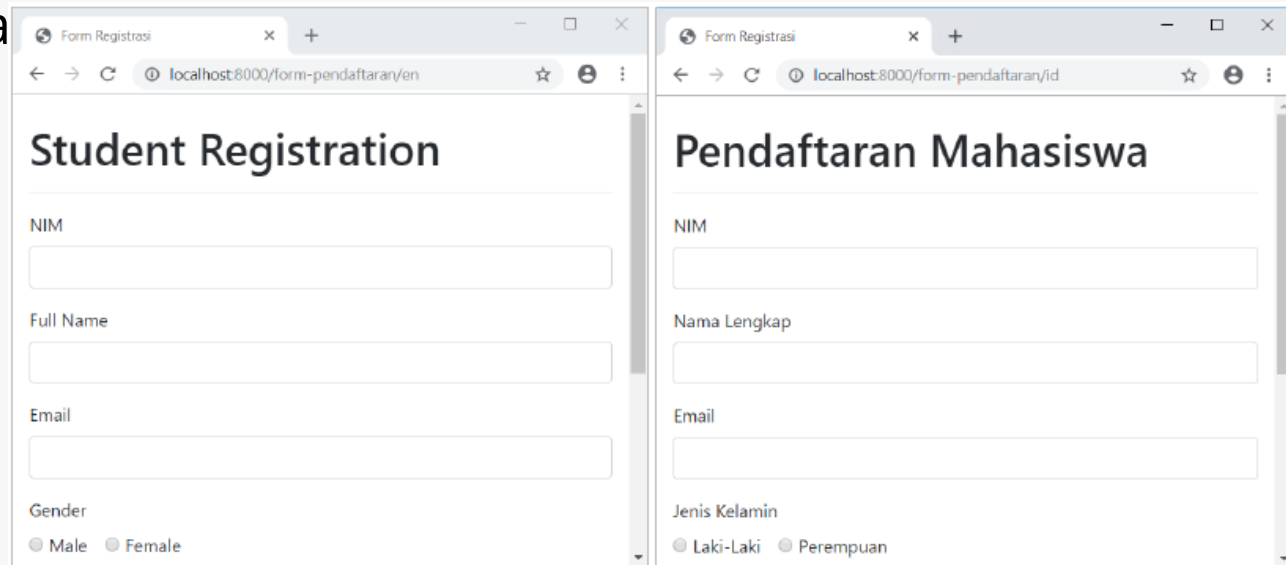
- The goal is that when the `/form-pendaftaran/id` page is accessed, the form appears in Indonesian, while when the `/form-pendaftaran/en` page is accessed, the form will appear in English. These two routes will also call different methods, namely `formPendaftaranId()` and `formPendaftaranEn()`.
- The following is the code for the `MahasiswaController`:

```
app/Http/Controllers/MahasiswaController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App;
7
8  class MahasiswaController extends Controller
9  {
10     //...
11
12     public function formPendaftaranId()
13     {
14         App::setLocale('id');
15         return view('form-pendaftaran');
16     }
17
18     public function formPendaftaranEn()
19     {
20         App::setLocale('en');
21         return view('form-pendaftaran');
22     }
23
24     //...
25     //...
26 }
```

Fig 10. MahasiswaController

Dynamic Localization Setting

- In the MahasiswaController there are commands `App::setLocale('id')` and `App::setLocale('en')`. Here's how to dynamically adjust locale settings.
- Please try to open the page **localhost:8000/form-pendaftaran/id** and **localhost:8000/form-pendaftaran/en**, the same form will appear in 2 types of language.



The image displays two browser windows side-by-side, both showing a registration form. The left window is titled 'Form Registrasi' and shows the English version of the form at 'localhost:8000/form-pendaftaran/en'. The form title is 'Student Registration'. It contains input fields for 'NIM', 'Full Name', and 'Email', and a 'Gender' section with radio buttons for 'Male' and 'Female'. The right window is also titled 'Form Registrasi' and shows the Indonesian version of the form at 'localhost:8000/form-pendaftaran/id'. The form title is 'Pendaftaran Mahasiswa'. It contains input fields for 'NIM', 'Nama Lengkap', and 'Email', and a 'Jenis Kelamin' section with radio buttons for 'Laki-Laki' and 'Perempuan'.

Fig 11. Registration form in English (left) and in Indonesian (right)

Localization with Route Parameters

- In the previous topic, we have succeeded in displaying the form in 2 languages, but in fact it is still not efficient because there is duplication of program code in the `formPendaftaranId()` and `formPendaftaranEn()` methods. It would be better if the code to display the registration form view is only one. As a solution, we can use the **route parameter**.
- First, we modify the writing of the previous 2 routes to only 1 route:

routes/web.php

```
Route::get('/form-pendaftaran/{locale?}', [MahasiswaController::class, 'formPendaftaran']);
```

- **Form-pendaftaran** routes can now accept optional route parameters, which will be accommodated by the **locale** variable.

Localization with Route Parameters

- Next, inside the controller can be created with the following code:

```
app/Http/Controllers/MahasiswaController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App;
7
8  class MahasiswaController extends Controller
9  {
10     public function formPendaftaran($locale = 'id')
11     {
12         App::setLocale($locale);
13         return view('form-pendaftaran');
14     }
15
16     //...
17     //...
18 }
```

Fig 12. MahasiswaController

- The value inputted from the route parameter will be captured by the `$locale` argument which is also filled with the default value of `'id'`. The goal is that when the registration form page is accessed without arguments, this `'id'` value will be used.

Localization with Route Parameters

- The `$locale` argument then becomes the input value into the `App::setLocale()` method, as in line 12 : `App::setLocale($locale)`.
- Please access the **localhost:8000/form-pendaftaran/id** page, and the **localhost:8000/form-pendaftaran/en** page. The appearance of the form will remain as before, namely in Indonesian and English, depending on the address written.

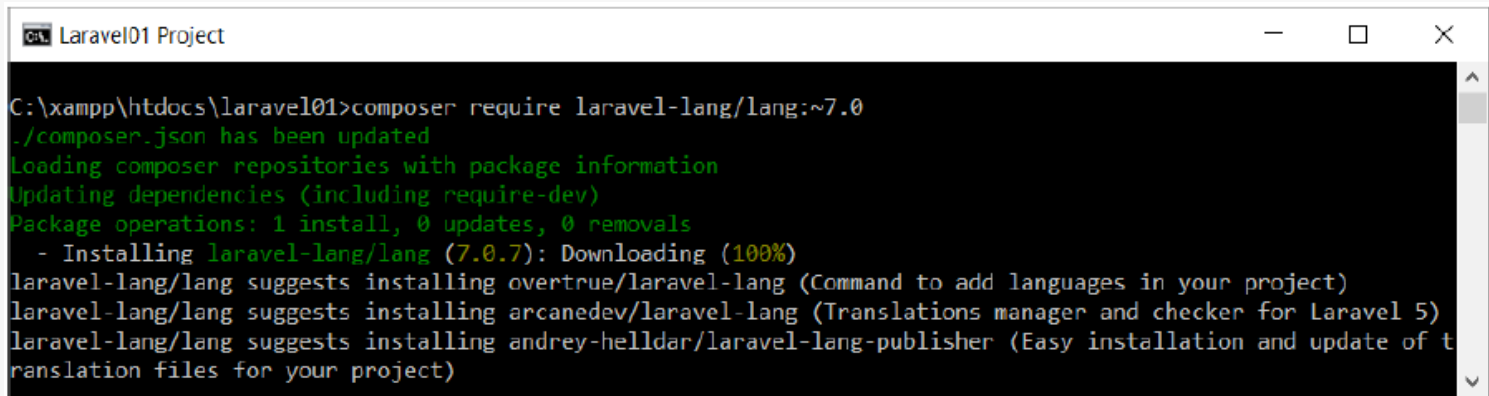
Localization for Form Validation

- One of the most common uses of localization is to display form error messages. By default, form error messages appear in English. Original file found in **resources\lang\en\validation.php**.
- Now, how to make the validation message appear in Indonesian? The trick, copy this validation.php file into the **id** folder, then translate one by one the text in the array.
- However, considering the large number of arrays, we can use Laravel's ready-to-use localization libraries, one of which is github.com/LaravelLang/lang. In this library, Laravel translation files are available for dozens of languages, including Indonesian.

Localization for Form Validation

- Please open cmd, go to **laravel** folder and type the following command:

```
composer require laravel-lang/lang:~7.0
```



```
C:\xampp\htdocs\laravel01>composer require laravel-lang/lang:~7.0
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing laravel-lang/lang (7.0.7): Downloading (100%)
laravel-lang/lang suggests installing overtrue/laravel-lang (Command to add languages in your project)
laravel-lang/lang suggests installing arcanedev/laravel-lang (Translations manager and checker for Laravel 5)
laravel-lang/lang suggests installing andrey-helldar/laravel-lang-publisher (Easy installation and update of t
translation files for your project)
```

Fig 13. Installation Process Lib Laravel-lang

- When finished, please open the `vendor\laravel-lang\lang\src` folder. Inside there are dozens of folders with language codes.

Localization for Form Validation

- Please open the id folder, copy the 4 files in it, then paste it into the resources\lang\id folder.

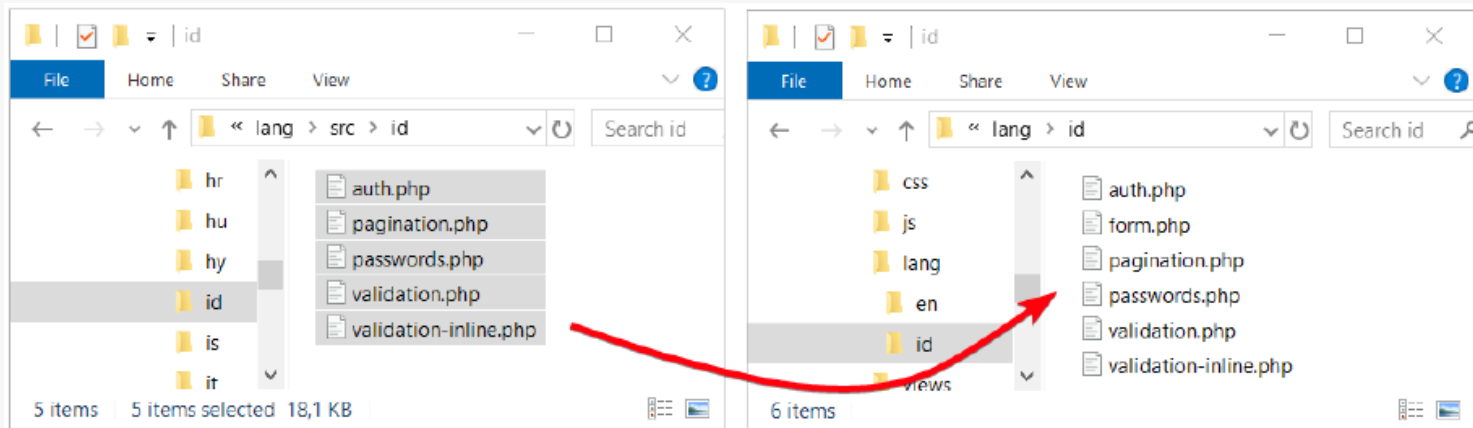
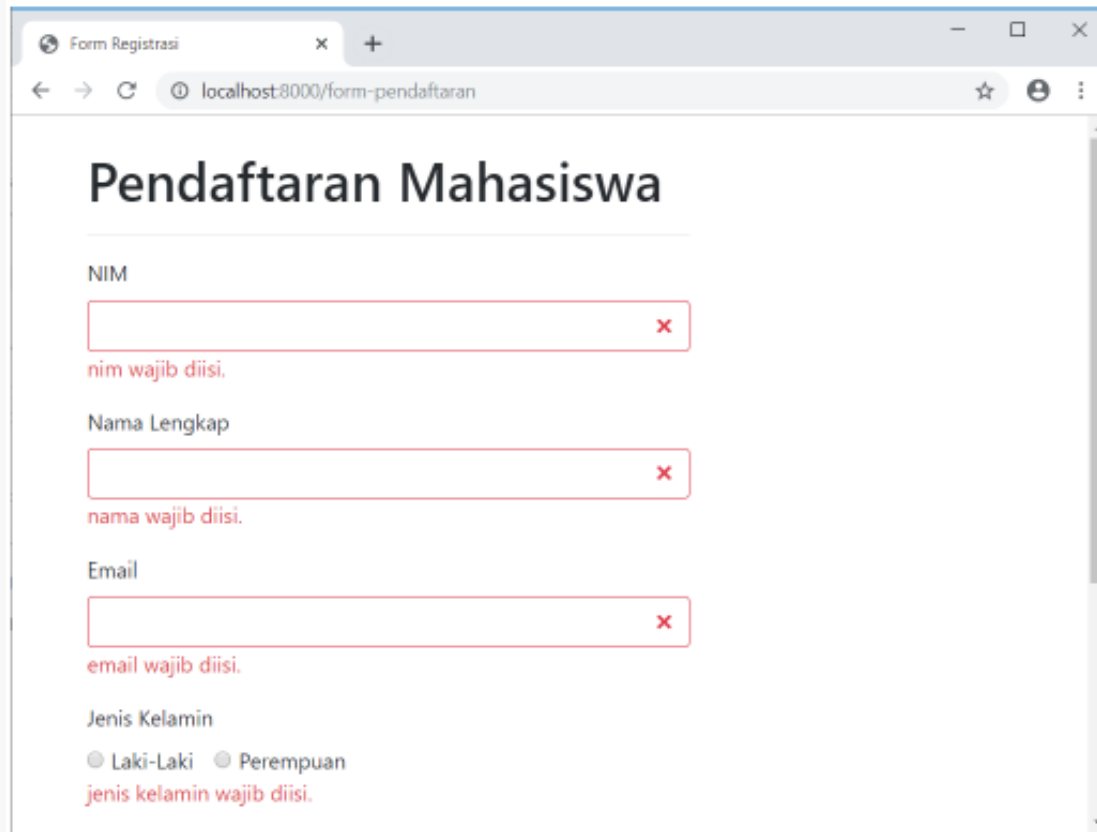


Fig 14. Copy the contents of the vendor\laravel-lang\lang\src\id folder and paste it into resources\lang\id

- Now please check the locale setting in the config\app.php file, make sure the value is id. Also check in the controller whether there is a prosesForm() method that will process the registration form validation. Next, run the registration form and just click the **Register** button:

Localization for Form Validation



The screenshot shows a web browser window with the title "Form Registrasi" and the address bar displaying "localhost:8000/form-pendaftaran". The page content is titled "Pendaftaran Mahasiswa". It contains four form fields, each with a red border and a red 'x' icon indicating a validation error:

- NIM**: The input field is empty. Below it, the error message "nim wajib diisi." is displayed in red.
- Nama Lengkap**: The input field is empty. Below it, the error message "nama wajib diisi." is displayed in red.
- Email**: The input field is empty. Below it, the error message "email wajib diisi." is displayed in red.
- Jenis Kelamin**: The field contains two radio buttons labeled "Laki-Laki" and "Perempuan". Below them, the error message "jenis kelamin wajib diisi." is displayed in red.

Fig 15. Validation error display

References

- Andy Turner. (2022). Laravel 9.x | PHP Learning Laravel with Easiest Way: The book will teach you Laravel 9.x step by step.
- Andre Pratama. (2019). Laravel Uncover. Duniallkom.
- <https://www.youtube.com/watch?v=ytuQ8oezLDw>

The background is a solid blue color with a gradient. On the left side, there are two large, overlapping circles. The top circle is a lighter shade of blue, and the bottom circle is a slightly darker shade. They overlap in the center-left area.

Thank You