

Web Framework Programming

Topic 8

Submissin Form

WEEK 08

Informatics Engineering
Universitas Surabaya



Submission Data

<https://laravel.com/docs/10.x/eloquent#inserts>

UI/UX provided for the required data

Gentelella Alela!

Welcome, John Doe

GENERAL

Home

Forms

General Form

Advanced Components

Form Validation

Form Wizard

Form Upload

Form Buttons

UI Elements

Tables

Data Presentation

Layouts

LIVE ON

Additional Pages

Form Validation

Search for... Go!

Form validation

For alternative validation library `jquery.validate` check out in the form page

Personal Info

Name * both name(s) e.g Jon Doe

Email * colorlibauthor email address is invalid

Confirm Email *

Number *

Website URL * www.website.com

Occupation *

Password *****

Repeat Password

Telephone *

Textarea *

Cancel Submit

Not all field are provided for user input.

- Some fields maybe are auto increment
- Some fields are maybe are taken from session
- Some fields maybe are not this user authority

Example – Validation & Submission Process

```
web.php x DosenController.php biodata.blade.php x
1 <?php
2
3 /*
4 |-----
5 | Web Routes
6 |-----
7 |
8 | Here is where you can register web routes for your applicat
9 | routes are loaded by the RouteServiceProvider within a grou
10 | contains the "web" middleware group. Now create something g
11 |
12 */
13
14 Route::get('/', function () {
15     return view('welcome');
16 });
17
18 Route::get('halo', function () {
19     return "Halo, Selamat datang di tutorial laravel www.ma
20 });
21
22 Route::get('blog', function () {
23     return view('blog');
24 });
25
26
27 Route::get('dosen', 'DosenController@index');
```

```
public function store(Request $request)
{
    $sekreteraris = new Sekreteraris;

    $sekreteraris->dari = $request->dari;
    $sekreteraris->tanggal_surat = $request->tanggal_surat;
    $sekreteraris->tanggal_diterima = $request->tanggal_diterima;
    $sekreteraris->no_agenda = $request->no_agenda;
    $sekreteraris->no_surat = $request->no_surat;
    $sekreteraris->image = $request->file('image')->store('image');

    $sekreteraris->save();

    return redirect('sekreteraris');
```

Exercise #1 : Creating Form

Example – Bootstrap Form

```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp">
    <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</div>
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

<https://getbootstrap.com/docs/5.0/forms/overview/>

Create – a New Category

In this example, user can create a new category with a form.

Make sure :

1. You already have Category's Controller and Model
2. Make sure you already create a table for model

Step :

1. Prepare a routing
2. Create a new file view with html form

Prepare a Routing

Open web.php and check a new routing called listkategori (line 26).

```
24 Route::resource('daftarfoto',PhotoController::class);
25 Route::resource('listmakanan',FoodController::class);
26 Route::resource('listkategori',CategoryController::class);
27
```

Note: Review Routing Topics in Week 2-4

The implication of 'route resource' can be displayed at this syntax

GET HEAD	listkategori	listkategori.index	> CategoryController@index
POST	listkategori	listkategori.store	> CategoryController@store
GET HEAD	listkategori/create	listkategori.create	> CategoryController@create
GET HEAD	listkategori/{listkategori}	listkategori.show	> CategoryController@show
PUT PATCH	listkategori/{listkategori}	listkategori.update	> CategoryController@update
DELETE	listkategori/{listkategori}	listkategori.destroy	> CategoryController@destroy
GET HEAD	listkategori/{listkategori}/edit	listkategori.edit	> CategoryController@edit

Prepare a Routing (2)

In `CategoryController` you will see :

```
21      * Show the form for creating a new resource.
22      */
23      public function create()
24      {
25          //
26      }
27
28      /**
29       * Store a newly created resource in storage.
30       */
31      public function store(Request $request)
32      {
33          //
34      }
35
```

Create method() inside your Controller

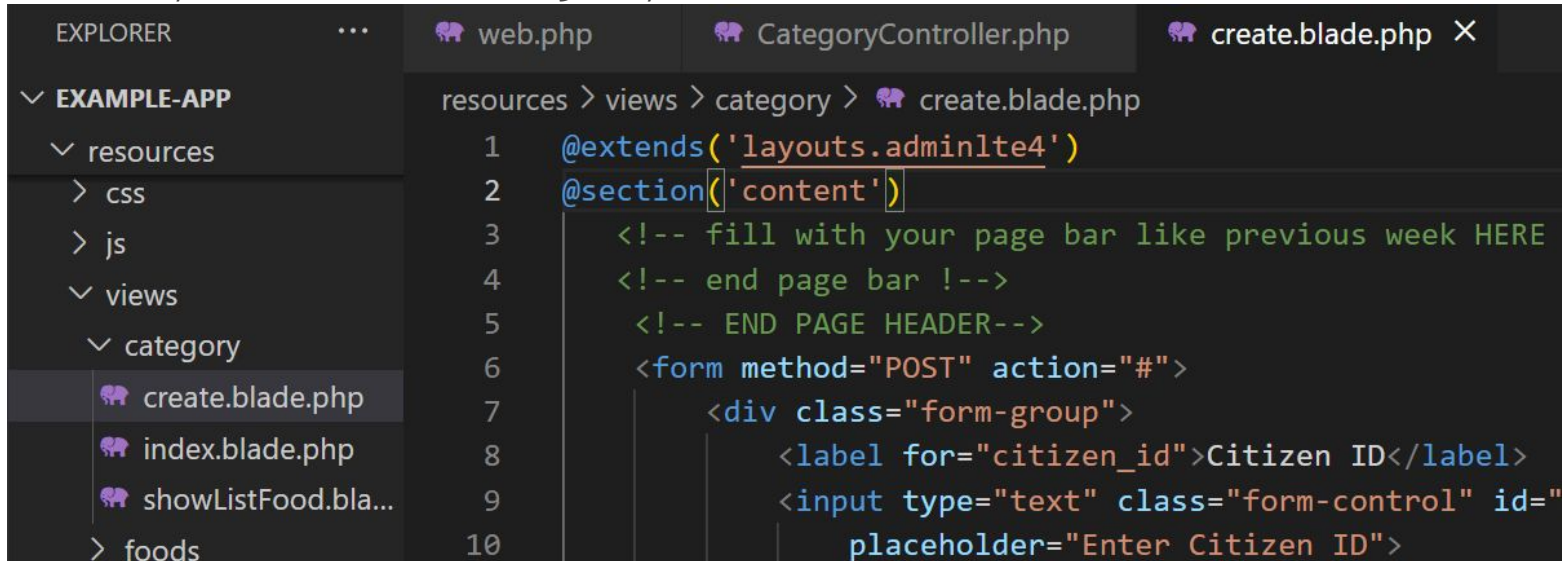
- The create() function shows the form to create a new Type data.
- If the form must collect other data from other tables, you can construct the query statement before the “return view” statement and parse it.
- This is a simple example of our case study (Type Form)
- This method will direct into directory /resources/views/category and formcreate.blade.php file.
- The name of ‘formcreate.blade.php’ is depends on analysis and naming file of your company

```
23     public function create()  
24     {  
25         return view('category.create');  
26     }
```

Create a New View :

category/formcreate.blade

- Put the file view in the folder category.
- Implement your include, extend and section of your template
- Implement your form create based on the bootstrap form.
- Modify the form according to your database



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 project structure: **EXAMPLE-APP** > **resources** > **views** > **category** > **create.blade.php**. The code editor shows the content of **create.blade.php** with line numbers 1 to 10. The code is as follows:

```
1 @extends('layouts.adminlte4')
2 @section('content')
3     <!-- fill with your page bar like previous week HERE
4     <!-- end page bar !-->
5     <!-- END PAGE HEADER-->
6     <form method="POST" action="#">
7         <div class="form-group">
8             <label for="citizen_id">Citizen ID</label>
9             <input type="text" class="form-control" id="
10             placeholder="Enter Citizen ID">
```

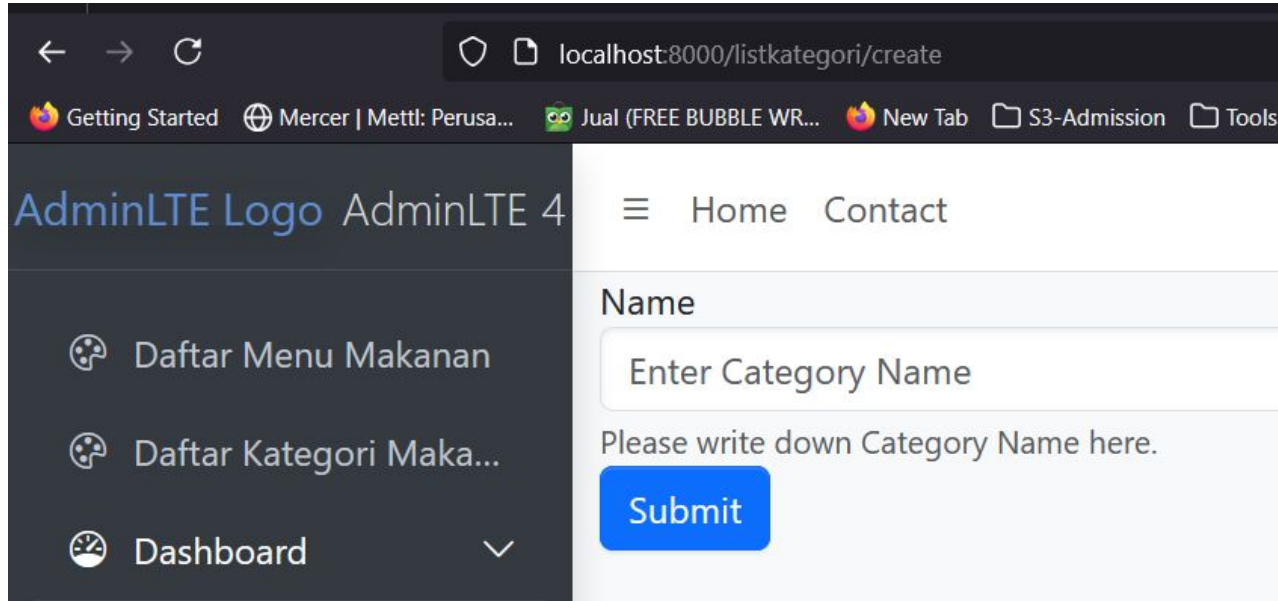
Create a New View : category/formcreate.blade (1)

```
@extends('layouts.adminlte4')
@section('content')
    <!-- fill with your page bar like previous week HERE !-->
    <!-- end page bar !-->
    <!-- END PAGE HEADER-->
    <form method="POST" action="#">

        <div class="form-group">
            <label for="name">Name</label>
            <input type="text" class="form-control" id="name" name="name" aria-describedby="name"
                placeholder="Enter Category Name">
            <small id="name" class="form-text text-muted">Please write down Category Name here.</small>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
@endsection
```

Test your view

Test your creation form by accessing from browser with this URL ☐
{{BASEURL}}/listkategori/create



The screenshot shows a web browser window with the address bar displaying `localhost:8000/listkategori/create`. The browser's tab bar includes several tabs: "Getting Started", "Mercer | Mettl: Perusa...", "Jual (FREE BUBBLE WR...", "New Tab", "S3-Admission", and "Tools".

The page content is divided into two main sections. On the left is a dark sidebar for "AdminLTE 4" containing a logo and three menu items: "Daftar Menu Makanan", "Daftar Kategori Maka...", and "Dashboard" with a dropdown arrow. On the right is the main content area, which features a top navigation bar with "Home" and "Contact" links. Below this is a form titled "Name" with a text input field containing the placeholder "Enter Category Name". A message below the input field reads "Please write down Category Name here." At the bottom of the form is a blue "Submit" button.

Exercise #2 : Saving Data into DB

Button Submit

- In the form, an element with `type="Submit"` will trigger an action or processing the form data
- The process is directed to a method in a Controller. Which inside `Resource Controller`, it already provided with the `store()` method.

```
<button type="submit" class="btn btn-primary">Submit</button>
```

Defining Target Action

- We need to define action with a target URL.
- In order to connect with the store() method, we have to follow the convention of Resource Controller

```
5      <!-- END PAGE HEADER-->
6      <form method="POST" action="#">
7
8          <div class="form-group">
9              <label for="name">Name</label>
10             <input type="text" class="form-control" id="name" name="na
11                 placeholder="Enter Category Name">
12             <small id="name" class="form-text text-muted">Please write
13         </div>
14         <button type="submit" class="btn btn-primary">Submit</button>
```


Defining Target Action (2)

- Check the list of routes `php artisan route:list`,
- (1) what is the URL or the route name
- (2) what HTTP method used to get into `store() / CategoryController@store` method

```
GET|HEAD listkategori ..... listkategori.index > CategoryController@index
POST listkategori ..... listkategori.store > CategoryController@store
GET|HEAD listkategori/create ..... listkategori.create > CategoryController@create
GET|HEAD listkategori/{listkategori} ..... listkategori.show > CategoryController@show
PUT|PATCH listkategori/{listkategori} ..... listkategori.update > CategoryController@update
DELETE listkategori/{listkategori} ..... listkategori.destroy > CategoryController@destroy
GET|HEAD listkategori/{listkategori}/edit ..... listkategori.edit > CategoryController@edit
```

- It shown that it need a POST method and has a route name = "listkategori.store" or url = "listkategori"

Defining Target Action (3)

```
<form method="POST" action="{{ route('list') }}">
```

App\Http\Controllers\CategoryControlle... ×

[POST] listkategori

```
<input type="text" class="form-control" placeholder="Enter Category Name" value="" />
```

```
placeholder="Enter Category Name" value="" />
```

```
<small id="name" class="form-control" value="" />
```

```
</div>
```

- listkategori.create
- listkategori.destroy
- listkategori.edit
- listkategori.index
- listkategori.show
- listkategori.store**
- listkategori.update

Defining Target Action (3)

- Therefore, the action parameter in the form is filled with :

```
<!-- END PAGE HEADER-->  
<form method="POST" action="{{ route('listkategori.store') }}">
```

- OR

```
6 <form method="POST" action="{{ url('listkategori') }}">  
7
```

Check Convention

- Check whether the your table already follows the convention or not.
- If your Category table doesn't have "created_at" and "update_at" fields, you need to do an overriding to the timestamp field (see the image) .

```
10 class Category extends Model
11 {
12     use HasFactory;
13     protected $table = 'categories';
14     public $timestamps = false;
15
16     public function foods(): HasMany
17     {
18         return $this->hasMany(Food::class, 'category_id', 'id');
19     }
20 }
```

Modify : CitizenController - Store

We need to code for save data into database

```
/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    $data= new Category();
    $data->name = $request->get('name');
    $data->save();
}
```

```
<form method="POST" action="{{ route('listkategori.store') }}">

    <div class="form-group">
        <label for="name">Name</label>
        <input type="text" class="form-control" id="name" name="name" aria-describedby="
            placeholder="Enter Category Name">
        <small id="name" class="form-text text-muted">Please write down Category Name he
    </div>

    <button type="submit" class="btn btn-primary">Submit</button>
```

Fill and see the result

4

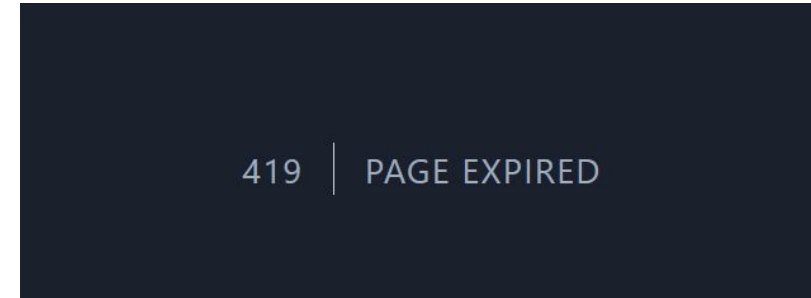
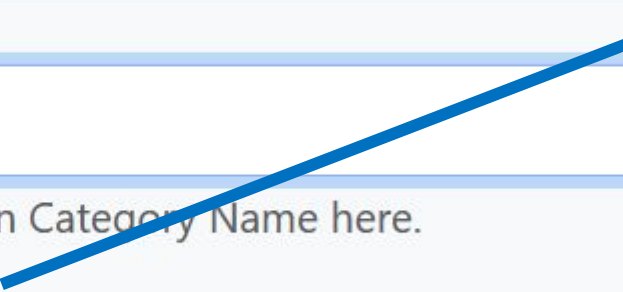
≡ Home Contact

Name

Beverages

Please write down Category Name here.

Submit



WHY ?

Error 419

- Error 419 is “a CSRF Token is missing or expired”
- CSRF Token is one of the security features. It overcomes an injection of ‘Cross-Site Request Forgery’ (CSRF) Script.

(https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

CSRF Protection

- Laravel will generate a 'token' in each managed session
- It can be done by using syntax @csrf in the request (both ajax or form) in the POST method

```
<form method="POST" action="/profile">  
    @csrf  
    ...  
</form>
```

<https://laravel.com/docs/10.x/csrf>

Add CSRF

In your view, add @csrf below <form>

```
<!-- End page bar -->
<!-- END PAGE HEADER-->
<form method="POST" action="{{ route('listkategori.store') }}">
    @csrf
    <div class="form-group">
        <label for="name">Name</label>
        <input type="text" class="form-control" id="name" name="na
```

Observe – Inspect Element Browser

Name

Please write down Category Name here.

Search HTML

```
<!--END PAGE HEADER-->
<form method="POST" action="http://localhost:8000/1"
  <input type="hidden" name="_token"
    value="My00FJfy1PHeQ0P3MFYcewPXnFAEt5xhSAGHxHrg"
  <div class="form-group">
    <label for="name">Name</label>
    <input id="name" class="form-control" type="text"
      placeholder="Enter Category Name"
v.app-wrapper > main.app-main > div.container > form > input >
```

Filter Styles

:hov .cls + ☀ 🌙 📄

element { inline

Layout Computed Change

Flexbox

Select a Flex container or item to

[Issue #1] There are specific required parameters that not complete



localhost:8000/listkategori

g Started



Mercer | Mettl: Perusa...



Jual (FREE BUBBLE WR...



New Tab



S3-Admission



Tools for Research

STACK



CONTEXT



FLARE

Illuminate \ Database \ QueryException

SQLSTATE[HY000]: General error: 1364 Field 'image' doesn't have a default value

```
INSERT INTO `categories` (`name`) VALUES (Beverages)
```

Columns: + Add ✕ Remove ▲ Up ▼ Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default
 1	id	BIGINT	20	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREME...
2	created_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	updated_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	name	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default
5	image	VARCHAR	1000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default

“image” column have to complete one value where you execute your new data ☐ based on “Allow Null” and “Default value”

Solution #1:

- Change specific column with NULLABLE or DEFAULT VALUE

Solution #2:

- You have to encourage user to set specific value with hidden type OR explicit user input (if possible)

In this case “category”, we use “solution 1” with Default Image from storage

Get from open source, Google Images, Canva or your image repository

default image



All

Images

Short videos

Shopping

Videos

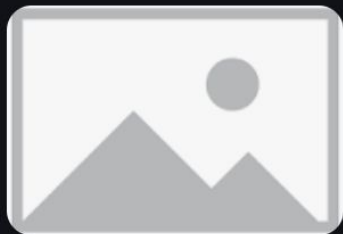
Forums

Web

More

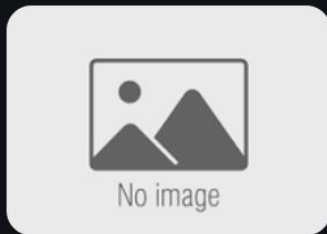
Tools

Images



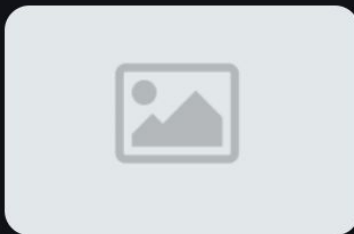
39+ Thousand Default Image Ic...

Shutterstock



Default image icon vector Mis...

Freepik



default-featured-image.png - Co...

Collective



Image Default Icons - F...

The Noun Project



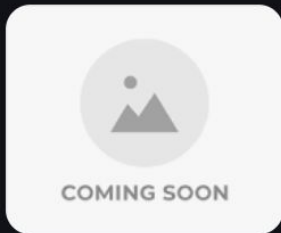
80+ Default Image Stock Illu...

iStock



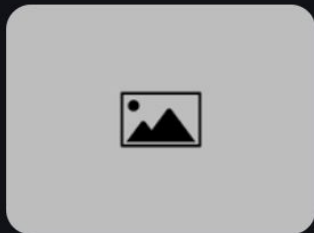
Default Image Image...

Adobe Stock



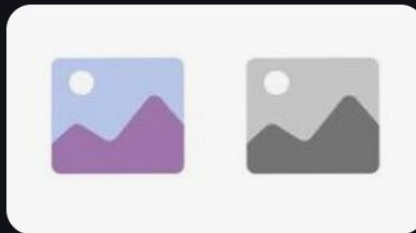
80+ Default Image Stock I...

iStock



default-image-5-1 | Cámara ...

Cámara de Comercio e In...



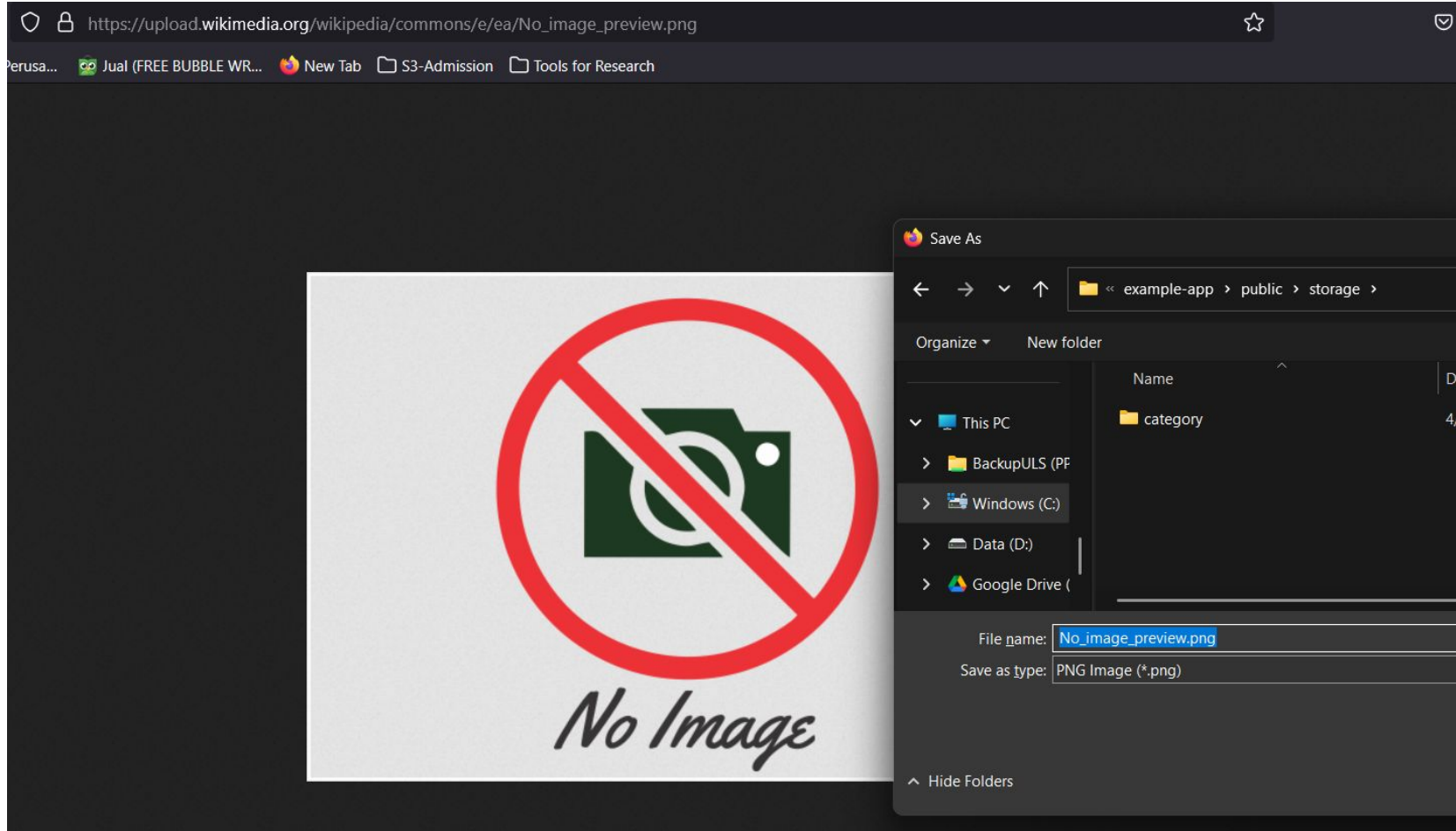
37 ribu Ilustrasi, Foto Stok, dan Gambar ...

Shutterstock



Default Picture | Features

TheWALL 360



Save your image into: `Laravel/public/storage/category/.....`

Name	Date modified
category	4/10/2025
.gitignore	2/13/2024
no_image_preview.png	5/3/2025 4

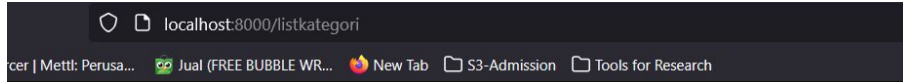
Get the filename of your image and insert into your database structure

Columns: + Add ✖ Remove ▲ Up ▼ Down								
#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Comment
1	id	BIGINT	20	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREME...	
2	created_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
3	updated_at	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
4	name	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default	
5	image	VARCHAR	1000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/> No default value <input checked="" type="radio"/> Custom text: <input type="text" value="no_image_preview.png"/> <input type="radio"/> NULL	

```
ALTER TABLE `categories`
CHANGE COLUMN `image` `image` VARCHAR(1000) NOT NULL DEFAULT 'no_image_preview.png';
```


Run and Submit again

There is no error message, but it shows blank page.
Check in the type table in database.



Status	Method	Domain	File	Initiator
200	POST	localhost:8000	listkategori	document
200	GET	localhost:8000	favicon.ico	FaviconLoader.sys.mjs:175 ...

New Record was Inserted

5	(NULL)	(NULL)	Coffee	
6	(NULL)	(NULL)	Non Coffee	
7	(NULL)	(NULL)	Healthy Juice	
8	(NULL)	(NULL)	Beverages	no_image_preview.png

Exercise #3 : Confirmation

Redirect

- For confirmation, we can re-use or re-call `index()` function in `TypeController`.
- It is not recommended to make new query then returning view, because it already exists in `index()`.
- Therefore, we will use **Redirect**.

Redirect Laravel

- Redirect in Laravel is similar with the common Redirect function which changing a page to another page based on the URL
- Redirect can be placed in the controller or action in routing file.
- The advantages of redirect in laravel
 - Can use redirect back() to going back in page before the active page. Usually use after a validation process.
 - Can be redirect based on route name and can be parameterized
 - Can be redirect to a controller function (redirect action)
 - Can be redirect to another URL
 - Can be redirect with bringing flash message (stored in the session)

<https://laravel.com/docs/10.x/responses#redirects>

Redirect : Example Route Name & Controller

```
return redirect()->route('login');
```

```
// For a route with the following URI: profile/{id}
```

```
return redirect()->route('profile', ['id' => 1]);
```

```
return redirect()->action('HomeController@index');
```

```
return redirect()->action(  
    'UserController@profile', ['id' => 1]  
);
```

Redirect : Example with FlashData

```
Route::post('user/profile', function () {  
    // Update the user's profile...  
  
    return redirect('dashboard')->with('status', 'Profile updated!');  
});
```

To get “profile update!” data, it use session()

```
@if (session('status'))  
    <div class="alert alert-success">  
        {{ session('status') }}  
    </div>  
@endif
```

Redirect : Confirmation

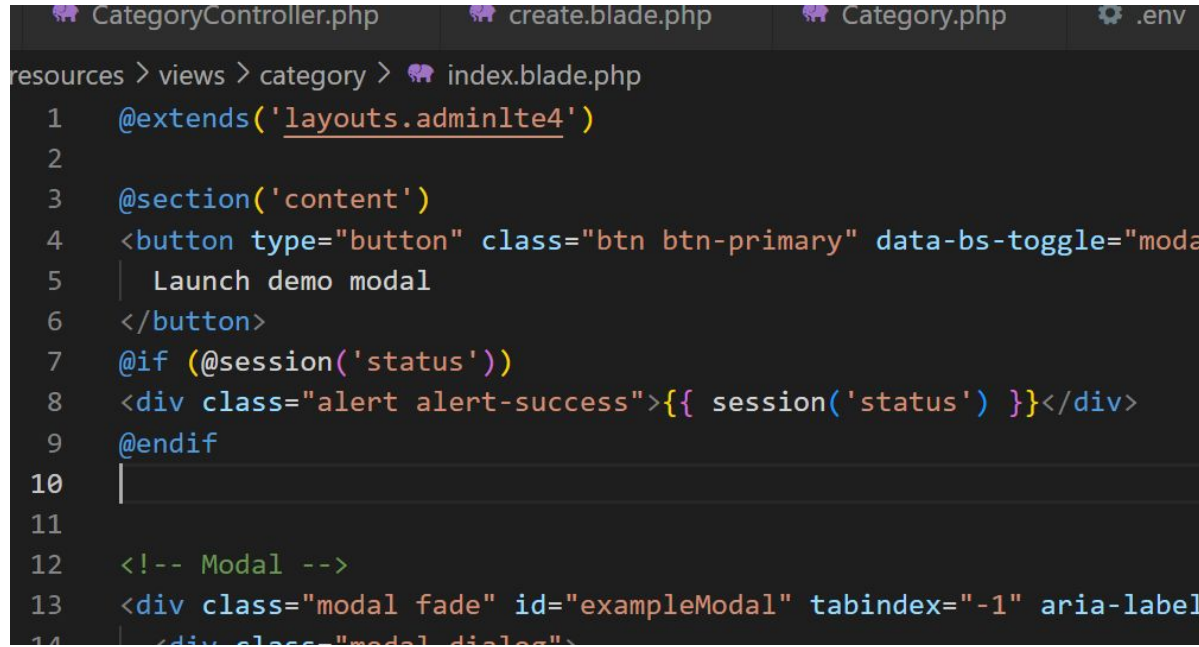
Using Flash after saving data

```
31 public function store(Request $request)
32 {
33     $data= new Category();
34     $data->name = $request->get('name');
35     $data->save();
36
37     return redirect()->route('listkategori.index')
38         ->with('status','Success updated data!');
39 }
```

Modify your index.blade.php

Put if session before the table data and do not forget to modify index method

```
@extends('layouts.adminlte4')
@section('content')
    @if (@session('status'))
        <div class="alert alert-success">
            {{ session('status') }}
        </div>
    @endif
    <table class="table">
        ...
    </table>
@endsection
```



```
resources > views > category > index.blade.php
1  @extends('layouts.adminlte4')
2
3  @section('content')
4      <button type="button" class="btn btn-primary" data-bs-toggle="modal"
5          | Launch demo modal
6      </button>
7
8      @if (@session('status'))
9          <div class="alert alert-success">{{ session('status') }}</div>
10         @endif
11
12     <!-- Modal -->
13     <div class="modal fade" id="exampleModal" tabindex="-1" aria-label=
14         <div class="modal-dialog">
```


Result

Name

Traditional Snack

Please write down Category Name here.

Submit

Success updated data!

Category with Hover Rows

The [.table-hover](#) class enables a hover state on table rows. The highest amount of food is [click here!](#)

#	Show Image	Name	Number of Foc
1	Show	Appetizer	0
2	Show	Main Course	2
3	Show	Snack	0
4	Show	Dessert	0
5	Show	Coffee	0
6	Show	Non Coffee	0
7	Show	Healthy Juice	0
8	Show	Beverages	0
10	Show	Traditional Snack	0

Add button : New Citizen

```
CategoryController.php  create.blade.php  Category.php  .env

resources > views > category > index.blade.php
3  @section('content')
4  <button type="button" class="btn btn-primary" data-bs-toggle="modal"
5  |   Launch demo modal
6  </button>
7  @if (@session('status'))
8  <div class="alert alert-success">{{ session('status') }}</div>
9  @endif
10
11  <a href="{{ route('listkategori.create') }}"
12  |   class="btn btn-primary" >
13  |   + New Category
14  </a>
15
16  <!-- Modal -->
```

Add button : New Citizen (2)

≡ [Home](#) [Contact](#)

[Launch demo modal](#) [+ New Category](#)

Category with Hover Rows

The [.table-hover](#) class enables a hover state on table rows. The hi

#	Show Image	Name
1	Show	Appetizer
2	Show	Main Course
3	Show	Snack

Homework

Redirect

Until next week

1. Implement a submission form for new menus, and transactions / orders
2. In new menu Form, there are capabilities:
 - a. Use combo box for selecting the category_id (from categories table).
 - b. The value of category_id is taken from query that parsed when calling the view in create() method
3. In orders Form, there are capabilities :
 - a. Use combo box for selecting the menu_id and each number
 - b. Use Many to many relationship (please open <https://laravel.com/docs/10.x/eloquent-relationships#many-to-many>)

Please submit your homework with your project

Thanks.