Web Framework Programming

Topic 9:

Update & Delete

WEEK 9

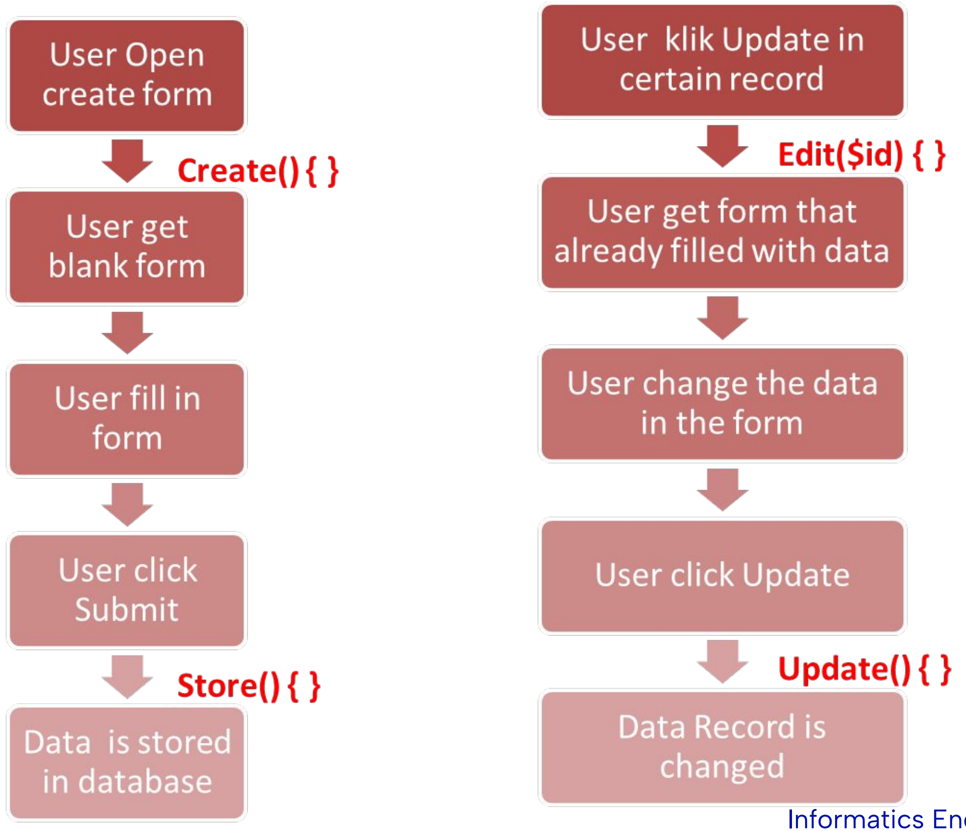
Informatics Engineering

Universitas Surabaya

Update Data

https://laravel.com/docs/10.x/eloquent#updates

Create vs. Update



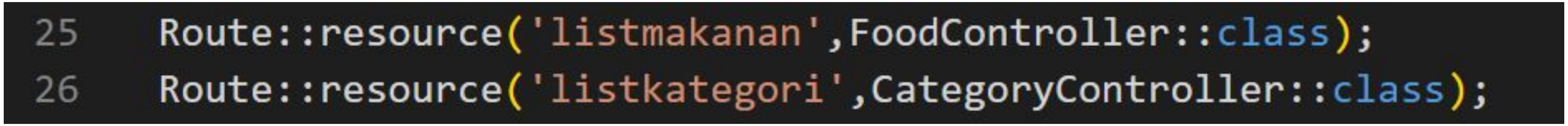
Exercise #1: create an Edit Form

Steps

1. Prepare a routing for the update form
2. Prepare the link/button in the records then check the routing first.
3. Create a new view. can re-use the existing code in ‘create form’. Fill the input field with the selected record. Call it in controller.

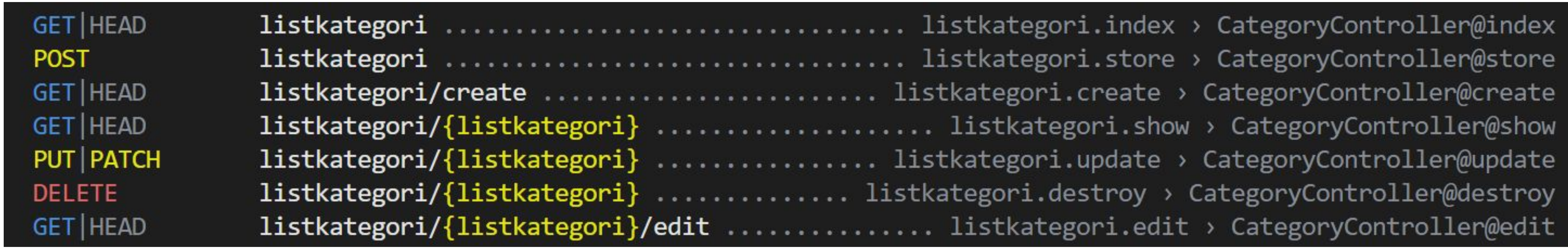
Prepare routing and Controller

If we use Resource Controller



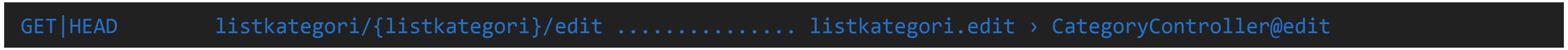
```
Route::resource('listmakanan',FoodController::class);
Route::resource('listkategori',CategoryController::class);
```

we can follow the existing method.
Check **php artisan route:list** 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
```

Focus on ‘edit’ function



```
GET|HEAD       listkategori/{listkategori}/edit ............... listkategori.edit › CategoryController@edit
```

GET|HEAD is a HTTP Method

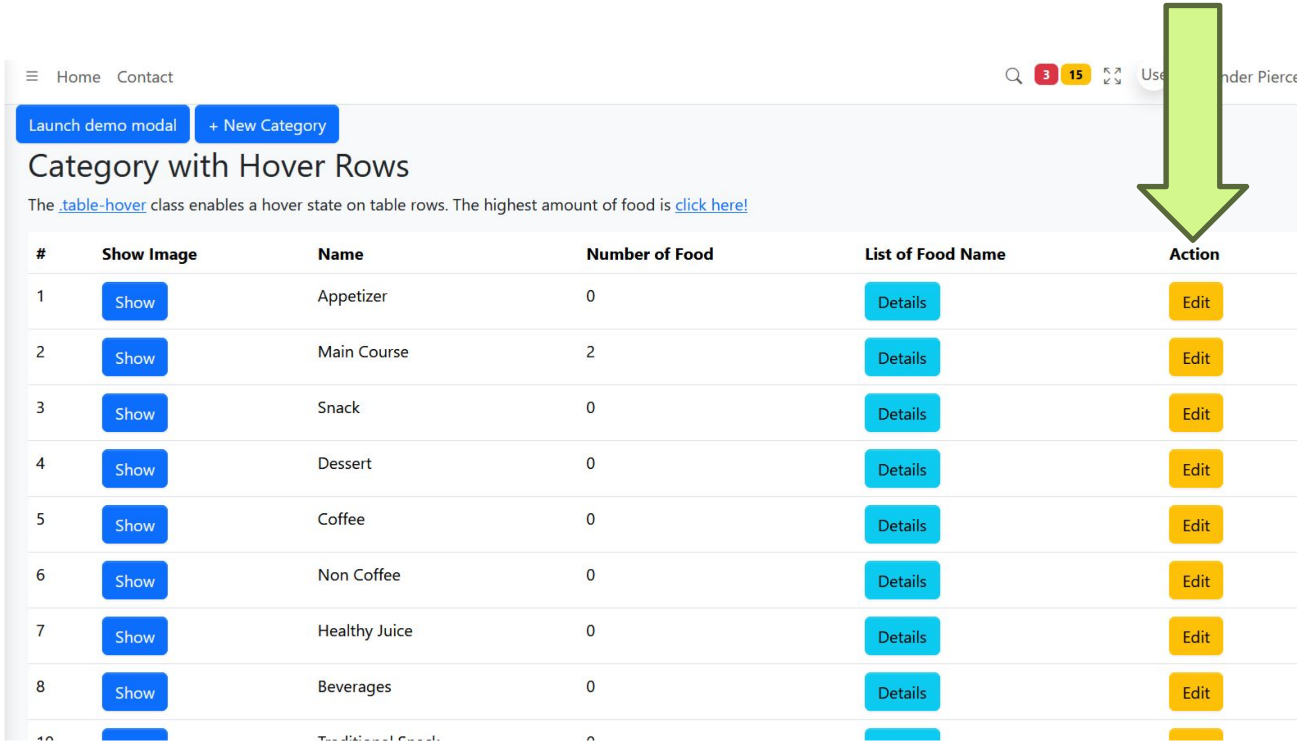
listkategori/{listkategori}/edit is a format URI for showing edit form and its existing data
{listkategori} -> will be replaced with number/string as the ID / Primary key of category table
Example :
http://localhost:8000/category/2798880729352657/edit
user will got edit form with the existing data for category with id = 2798880729352657

In Laravel, calling the edit form can use these 2 syntax:
With url : **url (‘category/’.$category->id.’/edit’)**
With route_name : **route(‘category.edit’,$category ->id)**

Example of Category Edit Form Action

For 10 minutes,

1. open your index-view of your Category
2. Add action “Edit”
3. Hover your mouse inside “Edit” button and look at the link. Is it similar to your routes?



Category with Hover Rows

The .table-hover class enables a hover state on table rows. The highest amount of food is click here!

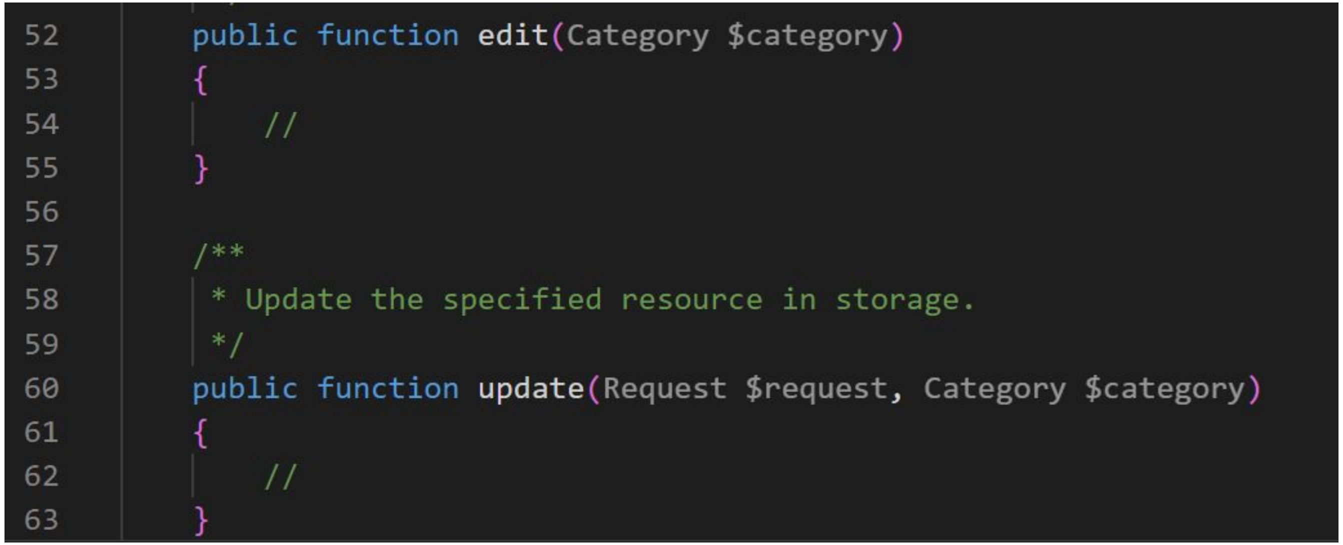
#	Show Image	Name	Number of Food	List of Food Name	Action
1	Show	Appetizer	0	Details	Edit
2	Show	Main Course	2	Details	Edit
3	Show	Snack	0	Details	Edit
4	Show	Dessert	0	Details	Edit
5	Show	Coffee	0	Details	Edit
6	Show	Non Coffee	0	Details	Edit
7	Show	Healthy Juice	0	Details	Edit
8	Show	Beverages	0	Details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

Prepare the link/button in your citizen/index.blade.php file

- Add new column for “action”
- Add new link for “show edit form”

```
<table class="table">
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Column X</th>
      ......
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
@foreach ($data as $d)
  <tr>
    <td>{{ $d->id }}</td>
      <td>{{ $d->name }}</td>
      <td>{{ $d->dataX }}</td>
              .........
      <td>
        <a class="btn btn-warning" href="{{ route('listkategori.edit’, $d->id) }}">Edit</a>
      </td>
  </tr>
@endforeach
  </tbody>
```

Check the routing & Controller



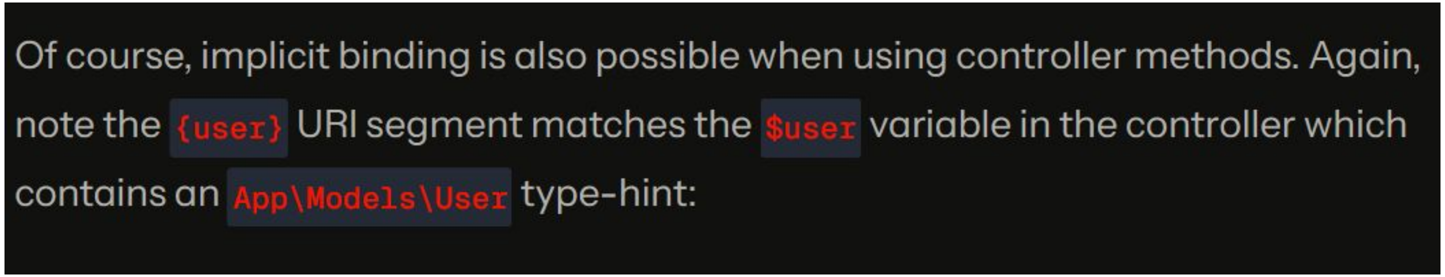
```
    public function edit(Category $category)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, Category $category)
    {
        //
    }
```

Method “edit” provides a parameter of Model Class. We can use it to retrieve the data that will be updated.

Method “update” provides two parameters, first $request from Form and second Model. We can use it to retrieve the data that will be updated from Form.

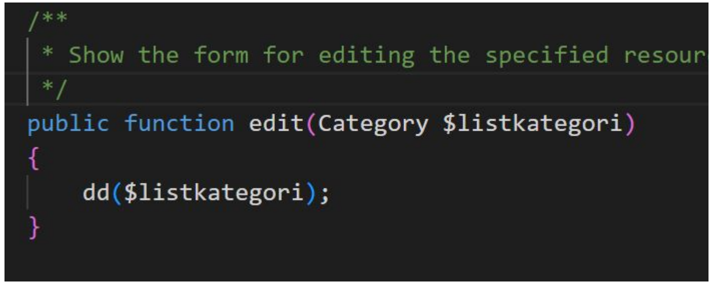
Check first your “Route Model Binding”

https://laravel.com/docs/10.x/routing#route-model-binding

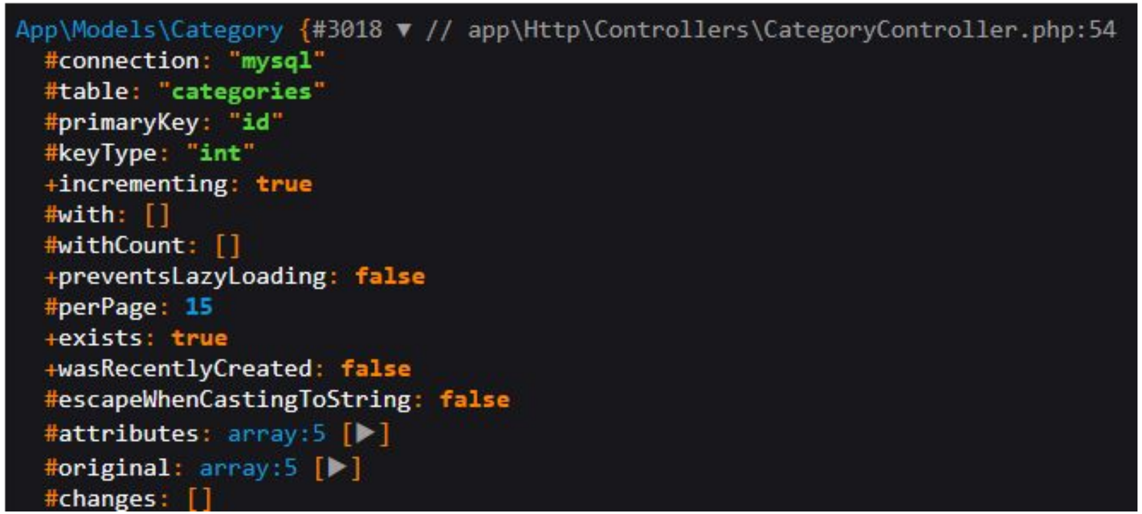


Of course, implicit binding is also possible when using controller methods. Again, note the `{user}` URI segment matches the `$user` variable in the controller which contains an `App\Models\User` type-hint:

Change $category to $listkategori for compile “route model binding” rule



```
/**
 * Show the form for editing the specified resour
 */
public function edit(Category $listkategori)
{
    dd($listkategori);
}
```



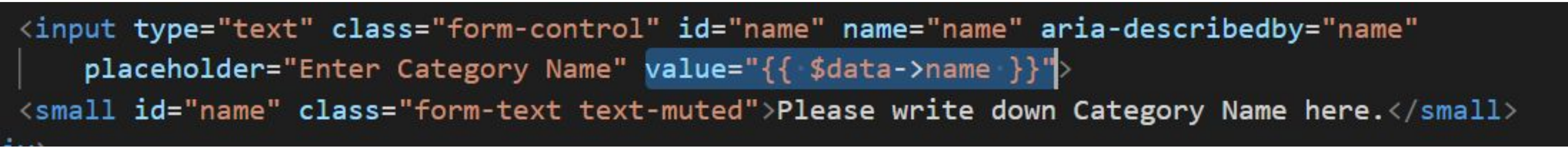
```
App\Models\Category {#3018 ▼ // app\Http\Controllers\CategoryController.php:54
  #connection: "mysql"
  #table: "categories"
  #primaryKey: "id"
  #keyType: "int"
  +incrementing: true
  #with: []
  #withCount: []
  +preventsLazyLoading: false
  #perPage: 15
  +exists: true
  +wasRecentlyCreated: false
  #escapeWhenCastingToString: false
  #attributes: array:5 [▶]
  #original: array:5 [▶]
  #changes: []
```

Create a view, fill with current record

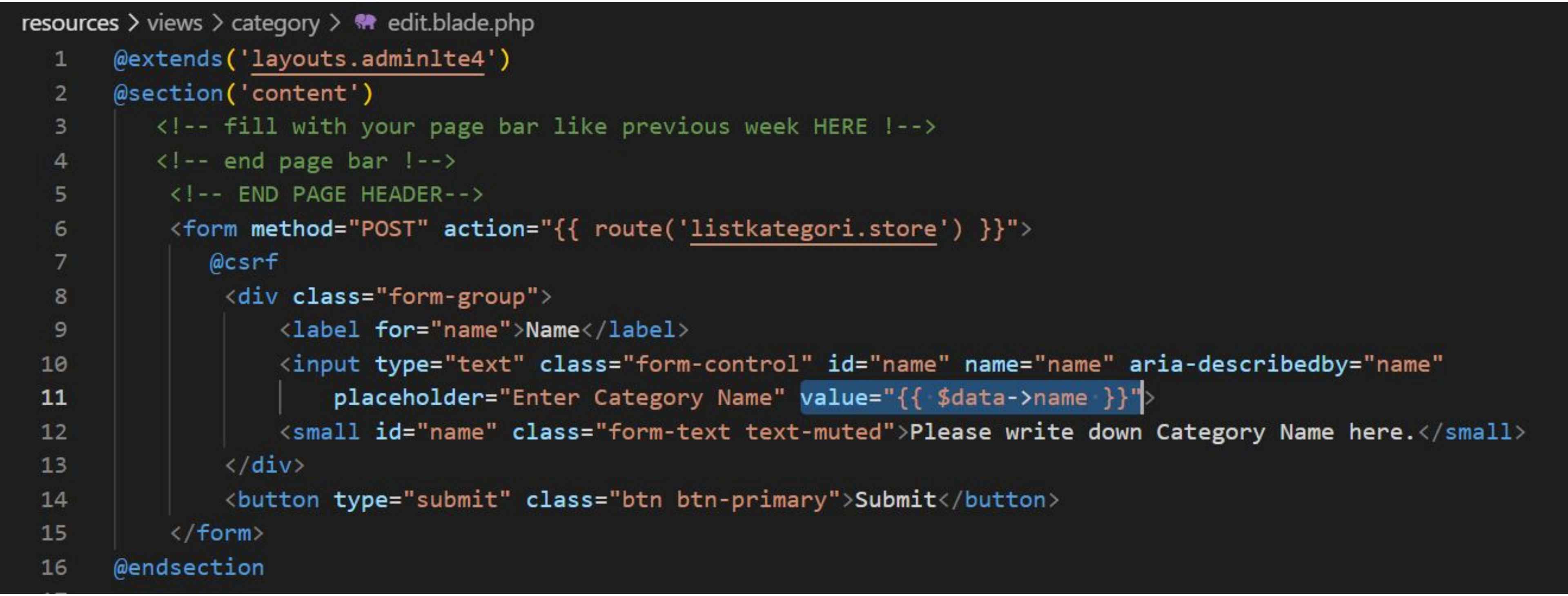
Create a new view named category/edit.blade.php. Re-use and modify code in category/create.blade.php and modify the edit method

```
public function edit(Category $listkategori)
{
  return view(‘category.edit’,  compact(‘listkategori’));
}
```

Edit your HTML Blade with “value” parameter inside input tag



```
<input type="text" class="form-control" id="name" name="name" aria-describedby="name"
    placeholder="Enter Category Name" value="{{ $data->name }}">
<small id="name" class="form-text text-muted">Please write down Category Name here.</small>
```


resources > views > category > edit.blade.php

```
@extends('layouts.adminlte4')
@section('content')
    <!-- fill with your page bar like previous week HERE !-->
    <!-- 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="name" aria-describedby="name"
                 placeholder="Enter Category Name" value="{{ $data->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
```

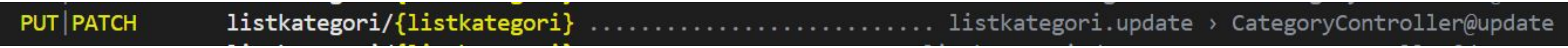
Exercise #2: Processing the updated data

The Update() method

The process for saving the updated data is directed to some method in controller via submit button. In resource controller, it is already prepared in method update()

Routing

Check the route:list,
what is the URL and the route name ?
what the HTTP method use to get into update() method ?



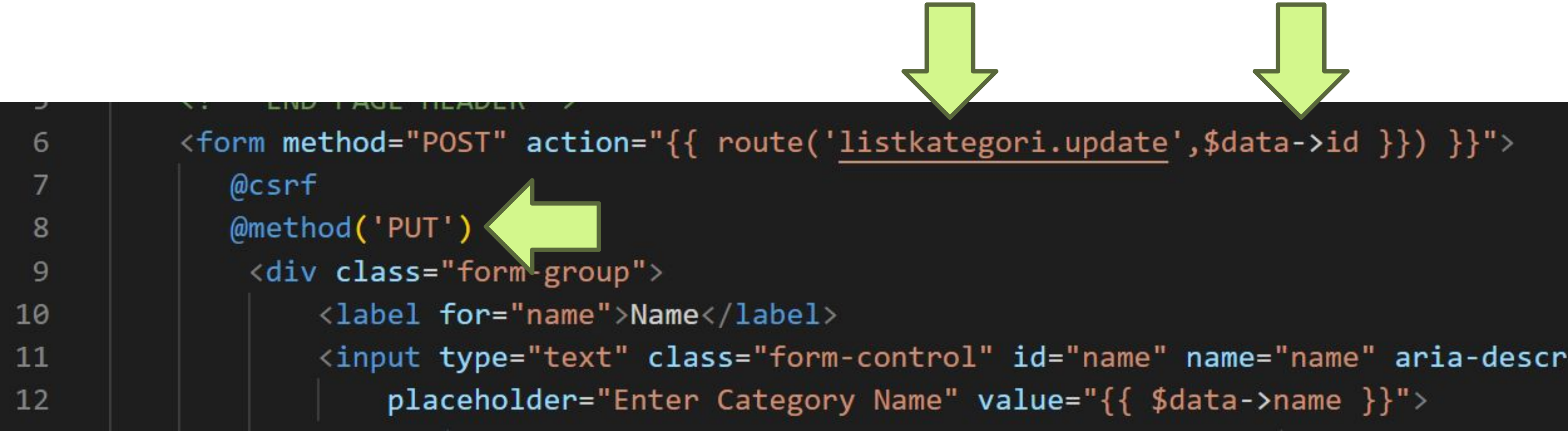
```
PUT|PATCH       listkategori/{listkategori} ........................ listkategori.update › CategoryController@update
```

To make HTTP PUT method, we use POST method and adding this syntax:

    @method("PUT")

- The {listkategori} must be replaced with the ID of the current record

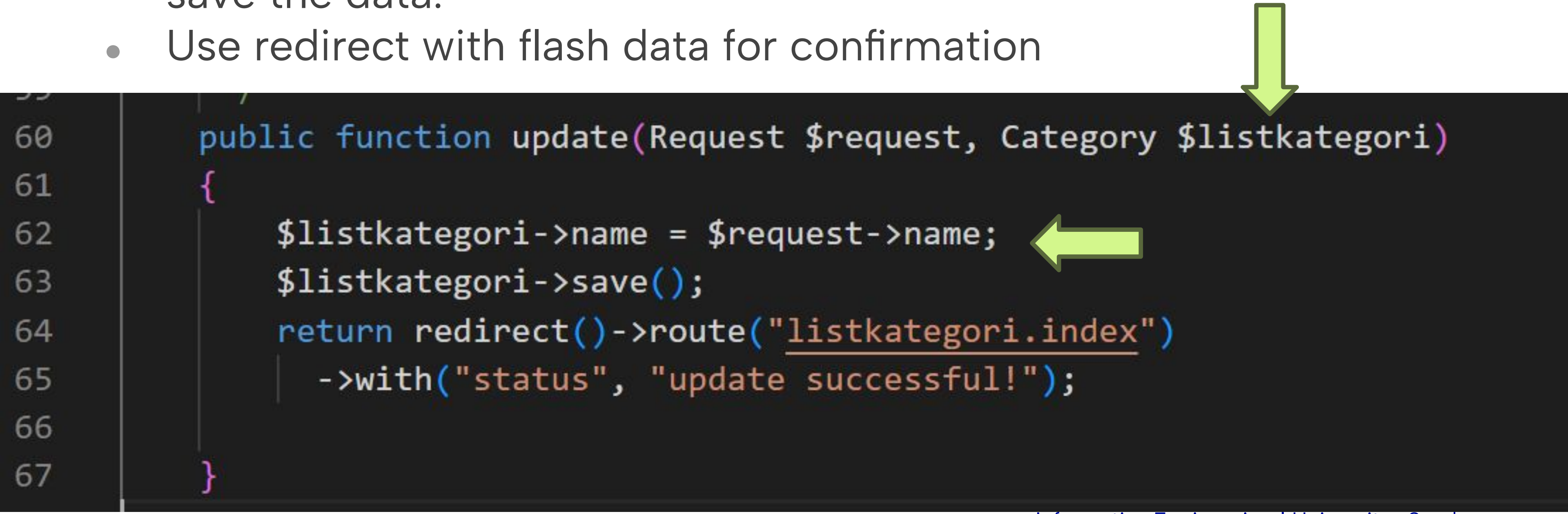
Modify the category/edit.blade.php



```
    <form method="POST" action="{{ route('listkategori.update',$data->id }}) }}">
      @csrf
      @method('PUT')
       <div class="form-group">
          <label for="name">Name</label>
          <input type="text" class="form-control" id="name" name="name" aria-descr
             placeholder="Enter Category Name" value="{{ $data->name }}">
```

Code the update() method

- Fill the table column with the corresponded request data, then save the data.
- Use redirect with flash data for confirmation



```
    public function update(Request $request, Category $listkategori)
    {
        $listkategori->name = $request->name;
        $listkategori->save();
        return redirect()->route("listkategori.index")
          ->with("status", "update successful!");

    }
```

Delete Data

https://laravel.com/docs/10.x/eloquent#deleting-models

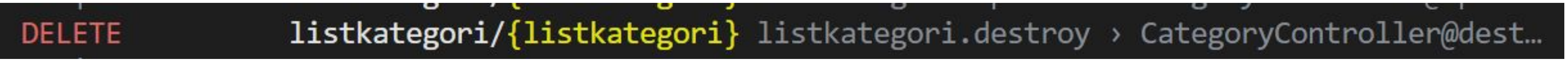
Steps

1. Prepare a routing for the delete
2. Prepare the link/button in the records then check the routing first.
3. Code the controller method to delete the data

Exercise #1: Adding delete button

The destroy() method

In resource controller, deleting record can use the provided method: the destroy() method.



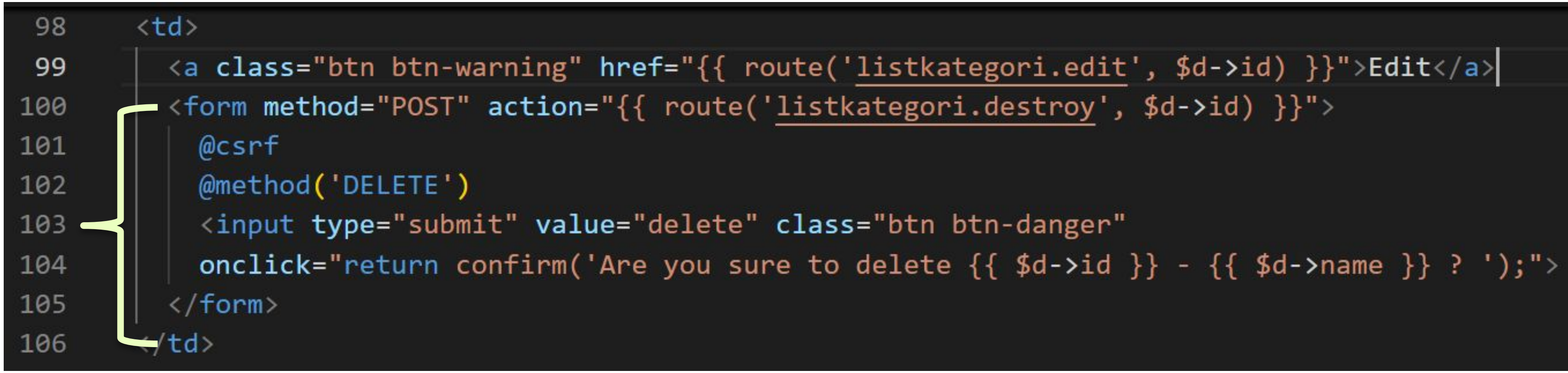
```
DELETE          listkategori/{listkategori} listkategori.destroy › CategoryController@dest…
```

The method for removing data is DELETE. It can be used POST method and replaced with DELETE by using
@method('DELETE')

It use url 'listkategori/{listkategori}'
Or route name listkategori.destroy

The Delete Button

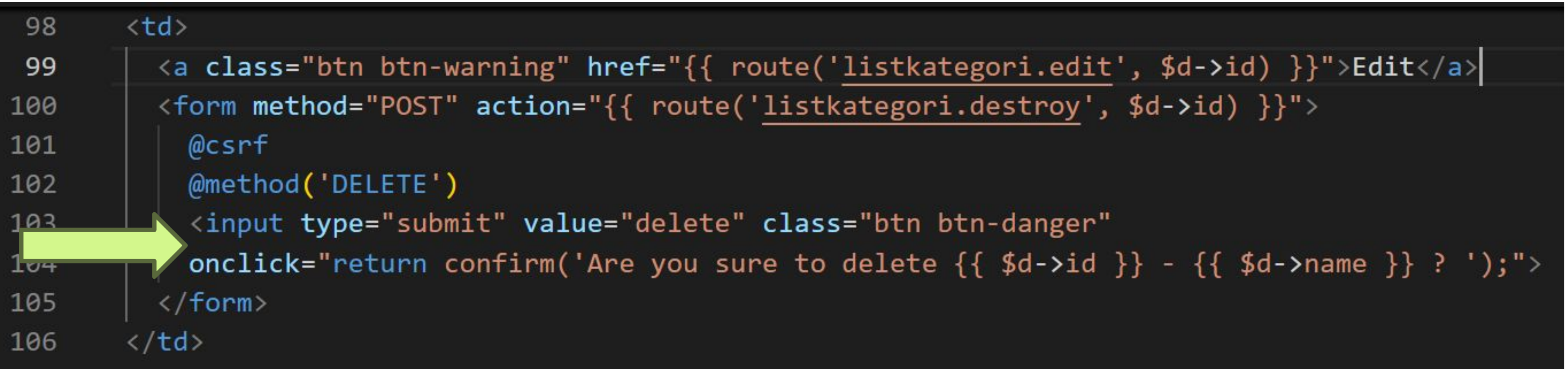
The delete button is not open any form. It directly contacting the controller. Therefore, the delete button must be wrapped in a Form element and the delete button typed is a submit button.



```
<td>
    <a class="btn btn-warning" href="{{ route('listkategori.edit', $d->id) }}">Edit</a>
    <form method="POST" action="{{ route('listkategori.destroy', $d->id) }}">
      @csrf
      @method('DELETE')
      <input type="submit" value="delete" class="btn btn-danger"
      onclick="return confirm('Are you sure to delete {{ $d->id }} - {{ $d->name }} ? ');">
    </form>
</td>
```

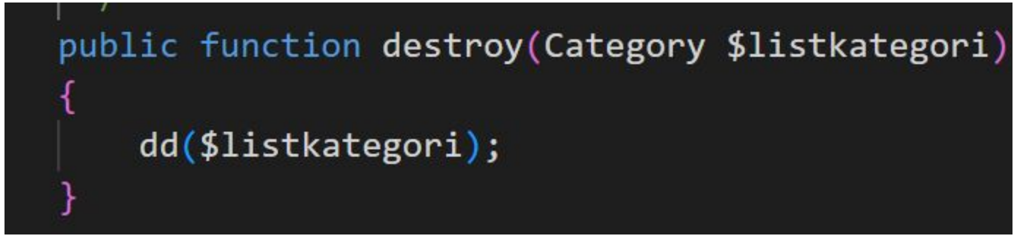
Additional

The “Delete” button **is very crucial** and the action is **immediately executed**. If someone accidentally clicks it, the data will be deleted directly without any confirmation.
It is fully recommended to provide a confirmation script in the delete button.

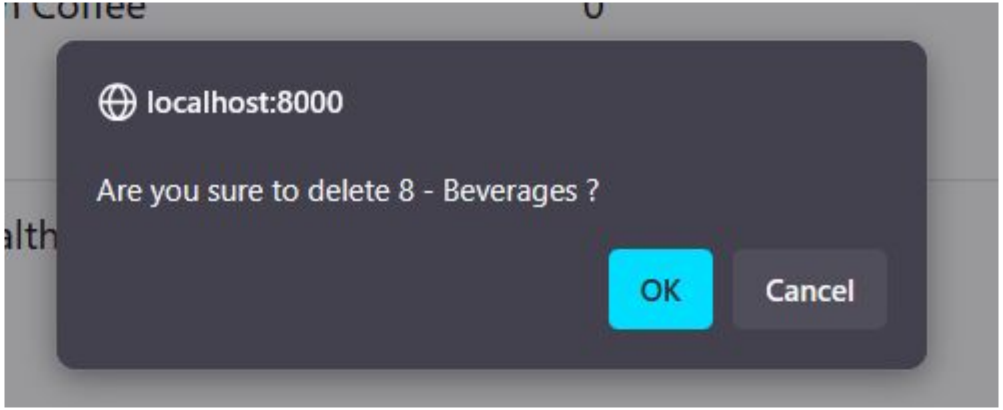


```
<td>
  <a class="btn btn-warning" href="{{ route('listkategori.edit', $d->id) }}">Edit</a>
  <form method="POST" action="{{ route('listkategori.destroy', $d->id) }}">
    @csrf
    @method('DELETE')
    <input type="submit" value="delete" class="btn btn-danger"
    onclick="return confirm('Are you sure to delete {{ $d->id }} - {{ $d->name }} ? ');">
  </form>
</td>
```

Try your delete route

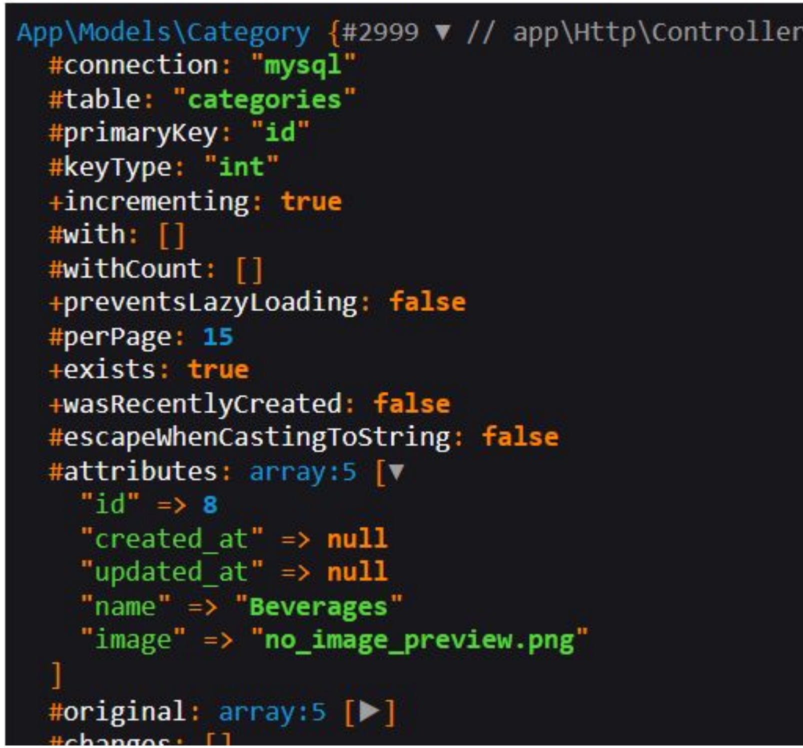


```
public function destroy(Category $listkategori)
{
    dd($listkategori);
}
```



localhost:8000

Are you sure to delete 8 - Beverages ?

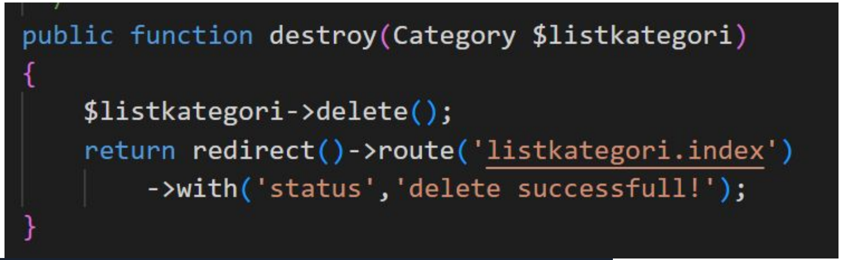


```
App\Models\Category {#2999 ▼ // app\Http\Controller
  #connection: "mysql"
  #table: "categories"
  #primaryKey: "id"
  #keyType: "int"
  +incrementing: true
  #with: []
  #withCount: []
  +preventsLazyLoading: false
  #perPage: 15
  +exists: true
  +wasRecentlyCreated: false
  #escapeWhenCastingToString: false
  #attributes: array:5 [▼
    "id" => 8
    "created_at" => null
    "updated_at" => null
    "name" => "Beverages"
    "image" => "no_image_preview.png"
  ]
  #original: array:5 [▶]
  #changes: []
```

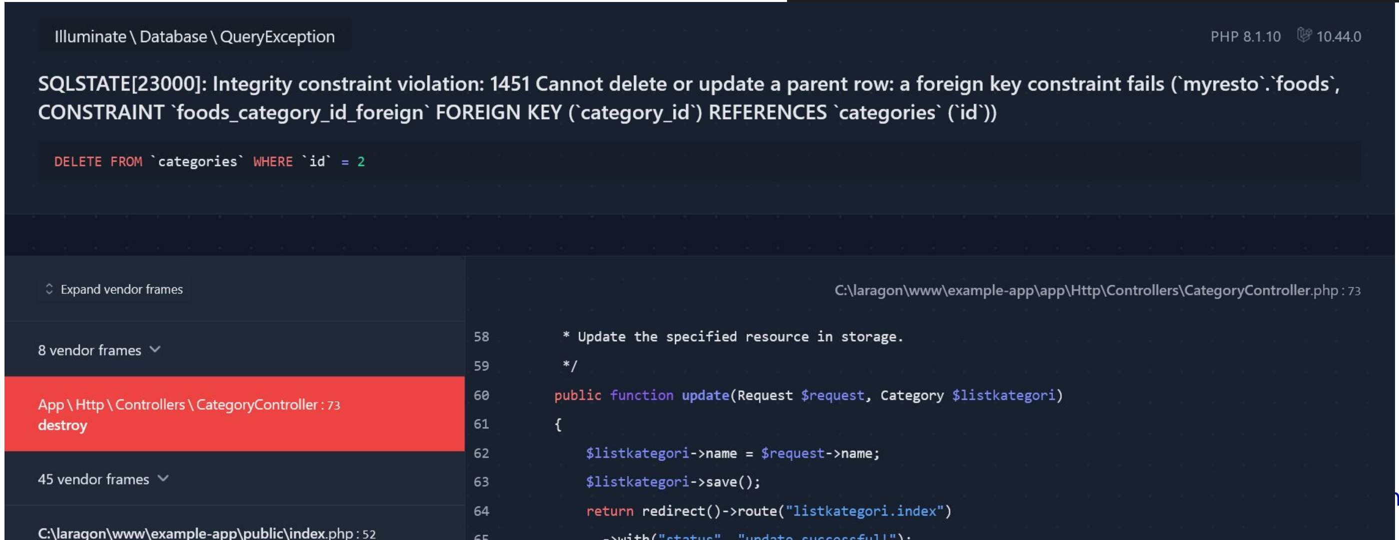
Exercise #2: Deleting data in database

Known Issue

Try to delete type data which is related with some data in hotel table. You will get this error.
-> handle it with try and catch



```php
public function destroy(Category $listkategori)
{
    $listkategori->delete();
    return redirect()->route('listkategori.index')
        ->with('status','delete successfull!');
}
```



Illuminate \ Database \ QueryException

PHP 8.1.10 10.44.0

SQLSTATE[23000]: Integrity constraint violation: 1451 Cannot delete or update a parent row: a foreign key constraint fails (`myresto`.`foods`, CONSTRAINT `foods_category_id_foreign` FOREIGN KEY (`category_id`) REFERENCES `categories` (`id`))

```sql
DELETE FROM `categories` WHERE `id` = 2
```

Expand vendor frames

8 vendor frames

App \ Http \ Controllers \ CategoryController : 73
destroy

45 vendor frames

C:\laragon\www\example-app\public\index.php : 52

C:\laragon\www\example-app\app\Http\Controllers\CategoryController.php : 73

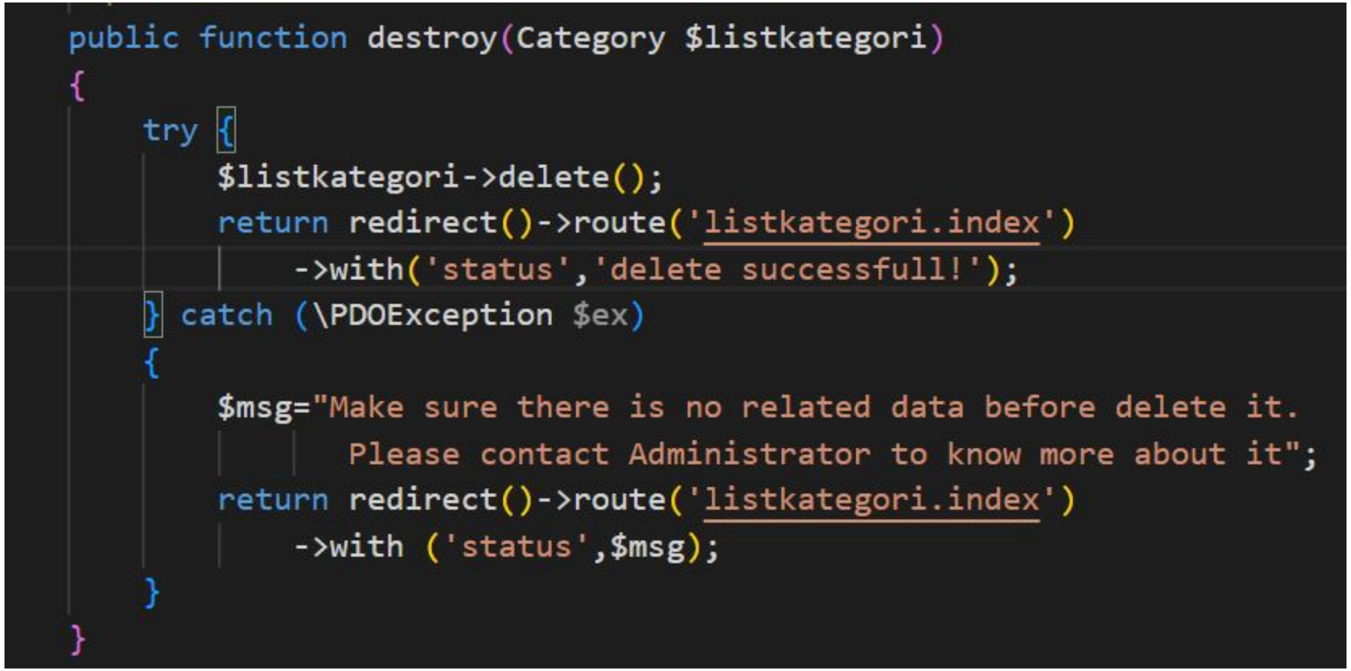
```php
     * Update the specified resource in storage.
     */
    public function update(Request $request, Category $listkategori)
    {
        $listkategori->name = $request->name;
        $listkategori->save();
        return redirect()->route("listkategori.index")
            ->with("status", "update successfull!");
```

Delete() Function

Modify the destroy() method

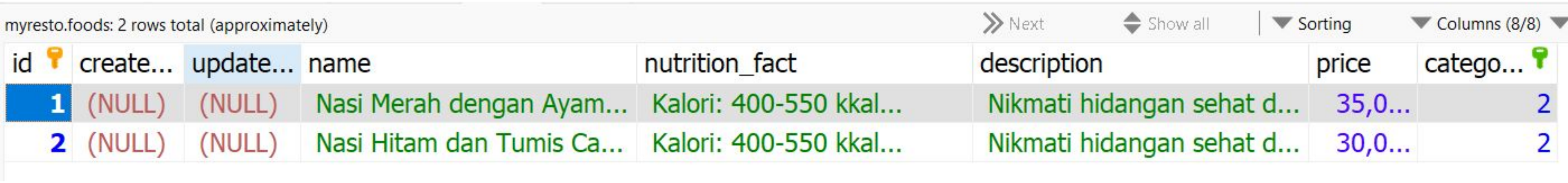
Success handler

Failed handler



```
public function destroy(Category $listkategori)
{
    try {
        $listkategori->delete();
        return redirect()->route('listkategori.index')
            ->with('status','delete successfull!');
    } catch (\PDOException $ex)
    {
        $msg="Make sure there is no related data before delete it.
                Please contact Administrator to know more about it";
        return redirect()->route('listkategori.index')
            ->with ('status',$msg);
    }
}
```

Explanation


myresto.foods: 2 rows total (approximately)

id	create...	update...	name	nutrition_fact	description	price	catego...
1	(NULL)	(NULL)	Nasi Merah dengan Ayam...	Kalori: 400-550 kkal...	Nikmati hidangan sehat d...	35,0...	2
2	(NULL)	(NULL)	Nasi Hitam dan Tumis Ca...	Kalori: 400-550 kkal...	Nikmati hidangan sehat d...	30,0...	2

The current citizen has a correspondence with 2 contributions. If you want to delete this data, the product owner must move contribution with ID 1 and 2 to another food

Soft Delete

https://laravel.com/docs/10.x/eloquent#soft-deleting

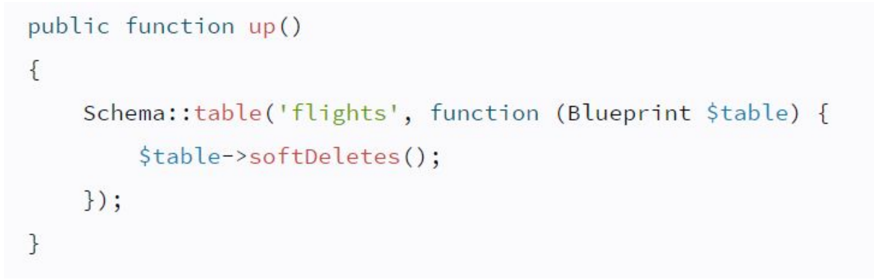
Soft Delete

Soft Delete is a mechanism in Laravel that is used to visually “temporarily” delete a record.
Benefits:

- Functions as a "recycle bin" of a data
- Avoid foreign key constraint errors when deleting a transaction data

Soft Delete Key

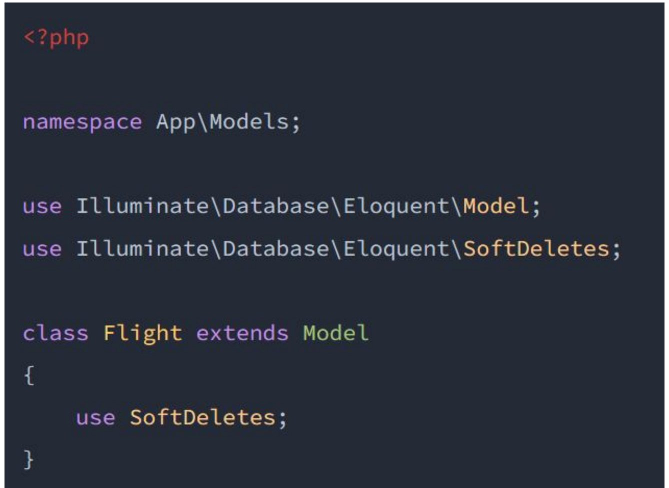
The existence of the "deleted_at" attribute in a table that is given the softdelete feature



```
public function up()
{
    Schema::table('flights', function (Blueprint $table) {
        $table->softDeletes();
    });
}
```

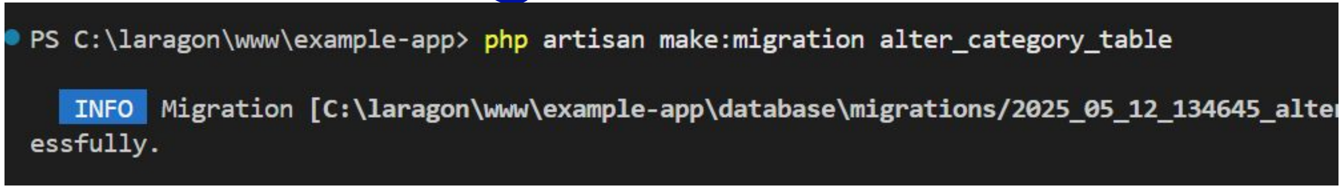
Will form 1 column deleted_at in the database table

There is a keyword use SoftDeletes on php Model

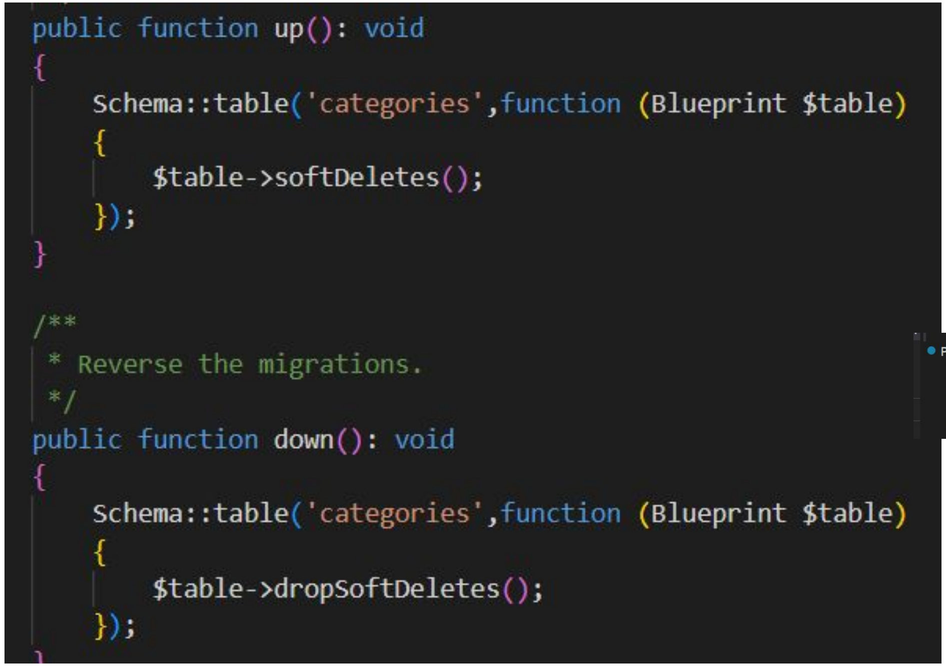


```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Flight extends Model
{
    use SoftDeletes;
}
```

Create a Migration File

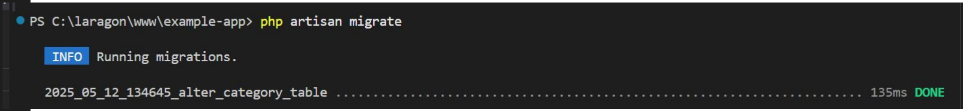


```
PS C:\laragon\www\example-app> php artisan make:migration alter_category_table

   INFO  Migration [C:\laragon\www\example-app\database\migrations/2025_05_12_134645_alte
essfully.
```



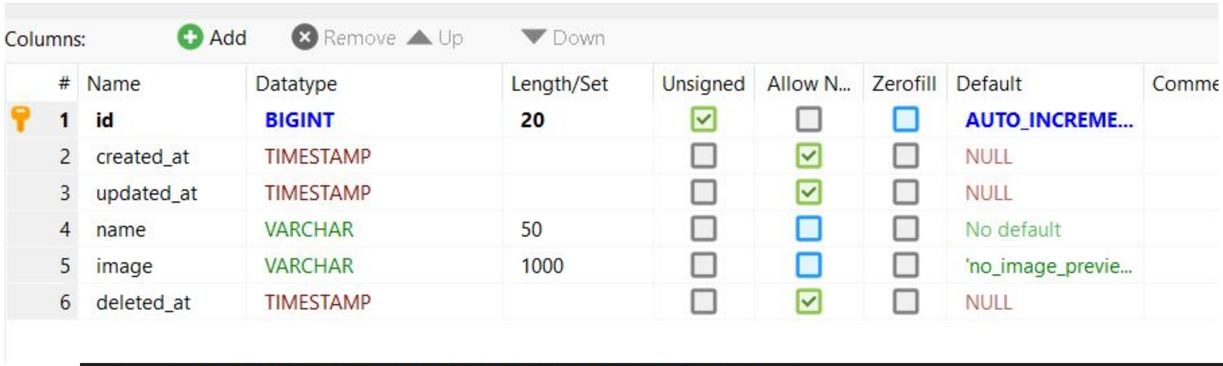
```
public function up(): void
{
    Schema::table('categories',function (Blueprint $table)
    {
        $table->softDeletes();
    });
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::table('categories',function (Blueprint $table)
    {
        $table->dropSoftDeletes();
    });
```

Add softDeletes() and dropSoftDeletes() in your new migration file



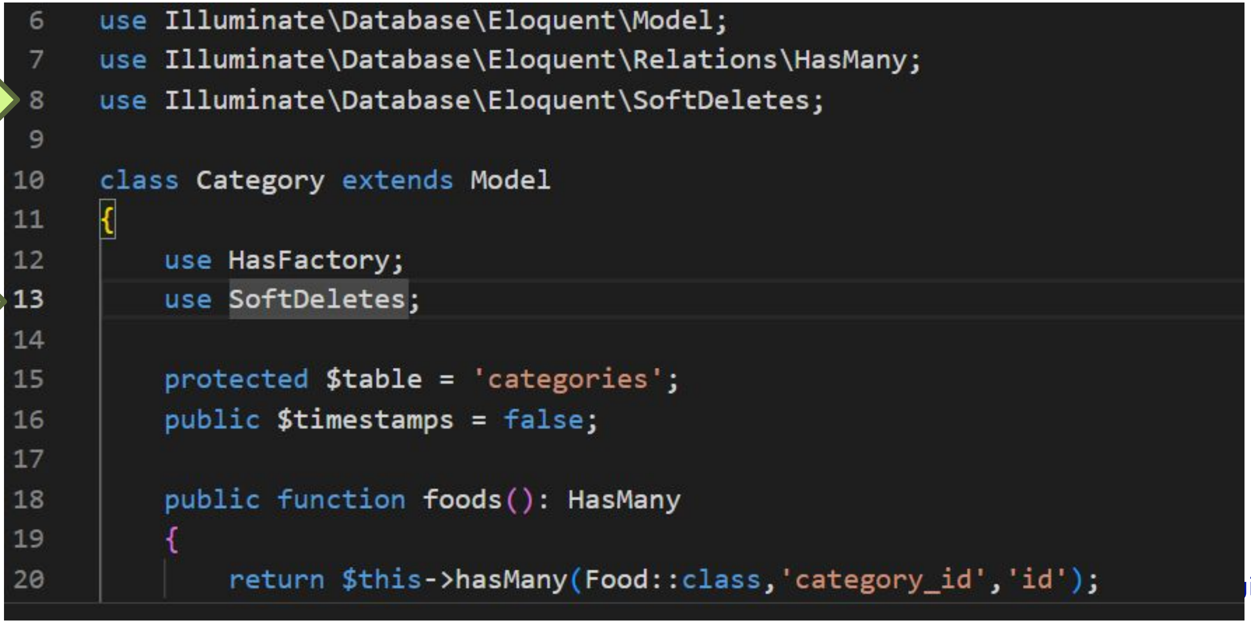
```
PS C:\laragon\www\example-app> php artisan migrate

   INFO  Running migrations.

  2025_05_12_134645_alter_category_table ................................................................ 135ms DONE
```

Check Categories Table and Model



Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Comme
1	id	BIGINT	20	✓	☐	☐	AUTO_INCREME...	
2	created_at	TIMESTAMP		☐	✓	☐	NULL	
3	updated_at	TIMESTAMP		☐	✓	☐	NULL	
4	name	VARCHAR	50	☐	☐	☐	No default	
5	image	VARCHAR	1000	☐	☐	☐	'no_image_previe...	
6	deleted_at	TIMESTAMP		☐	✓	☐	NULL	



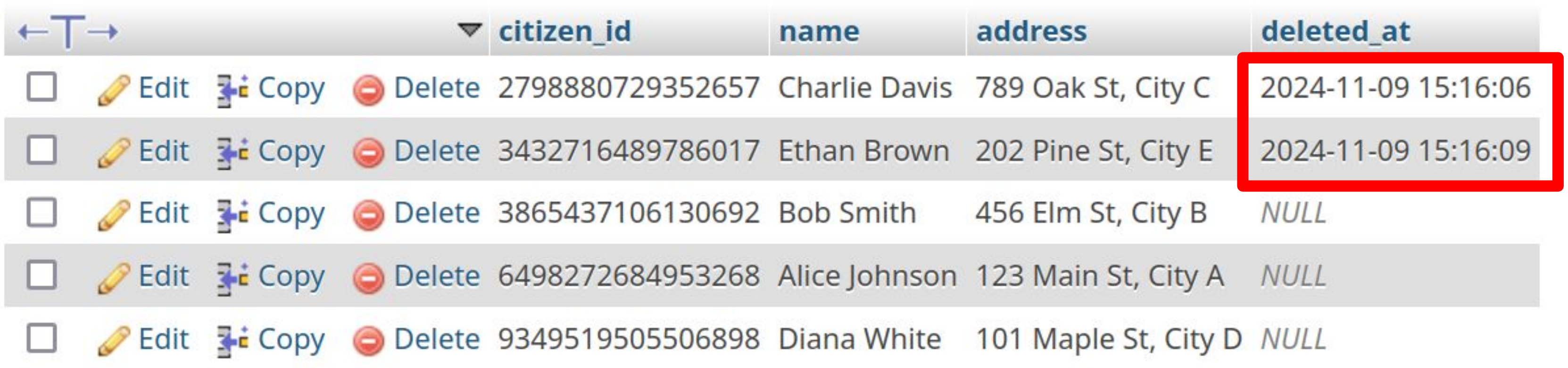
```php
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Database\Eloquent\SoftDeletes;

class Category extends Model
{
    use HasFactory;
    use SoftDeletes;

    protected $table = 'categories';
    public $timestamps = false;

    public function foods(): HasMany
    {
        return $this->hasMany(Food::class,'category_id','id');
```

Conclusion

By installing the two syntaxes above, the delete() method on Eloquent will function to fill the 'deleted_at' column with a delete time

Soft Delete Example

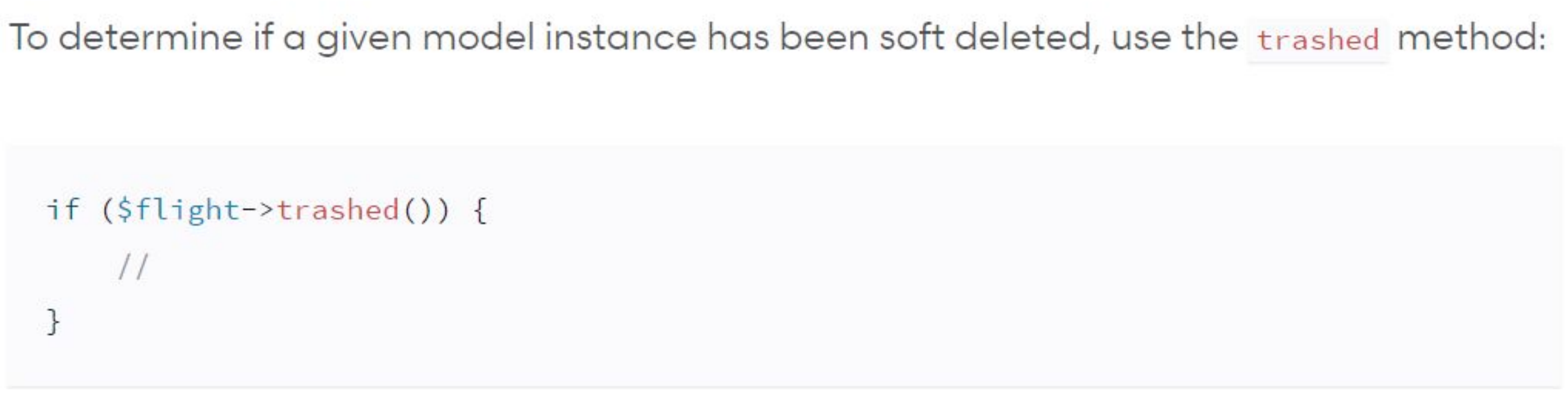
You only use softDelete, and directly your view with Eloquent::all() will automatically update with SoftDeletes Capabilites



←T→			citizen_id	name	address	deleted_at
☐ Edit	Copy	Delete	2798880729352657	Charlie Davis	789 Oak St, City C	2024-11-09 15:16:06
☐ Edit	Copy	Delete	3432716489786017	Ethan Brown	202 Pine St, City E	2024-11-09 15:16:09
☐ Edit	Copy	Delete	3865437106130692	Bob Smith	456 Elm St, City B	*NULL*
☐ Edit	Copy	Delete	6498272684953268	Alice Johnson	123 Main St, City A	*NULL*
☐ Edit	Copy	Delete	9349519505506898	Diana White	101 Maple St, City D	*NULL*

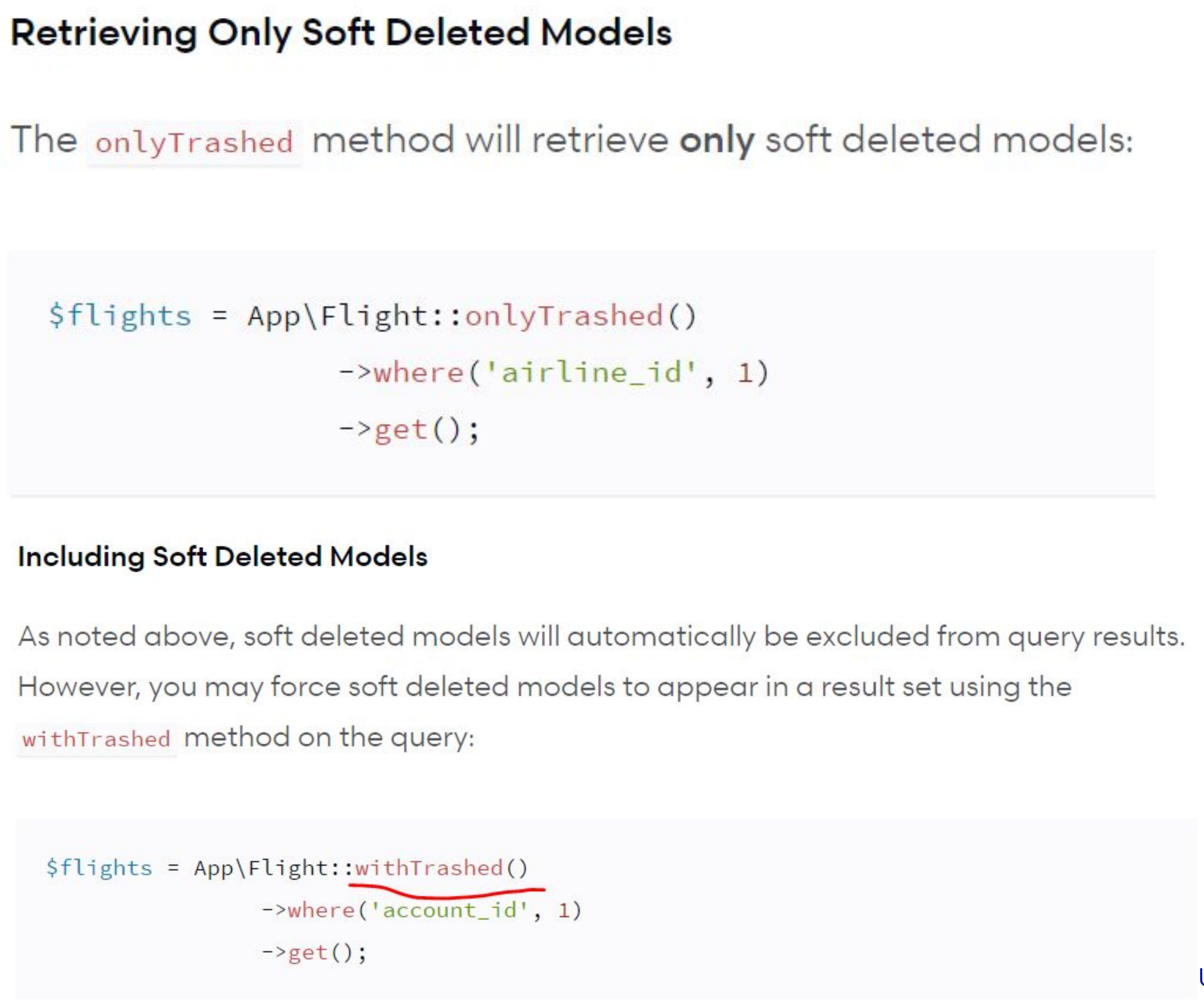
If you want to see the "recycle-bin"

If you want to display deleted data, use the trashed() method on your eloquent object.
Note variable $flight is an eloquent object



To determine if a given model instance has been soft deleted, use the `trashed` method:

```
if ($flight->trashed()) {
    //
}
```



Retrieving Only Soft Deleted Models

The `onlyTrashed` method will retrieve **only** soft deleted models:

```
$flights = App\Flight::onlyTrashed()
                ->where('airline_id', 1)
                ->get();
```

Including Soft Deleted Models

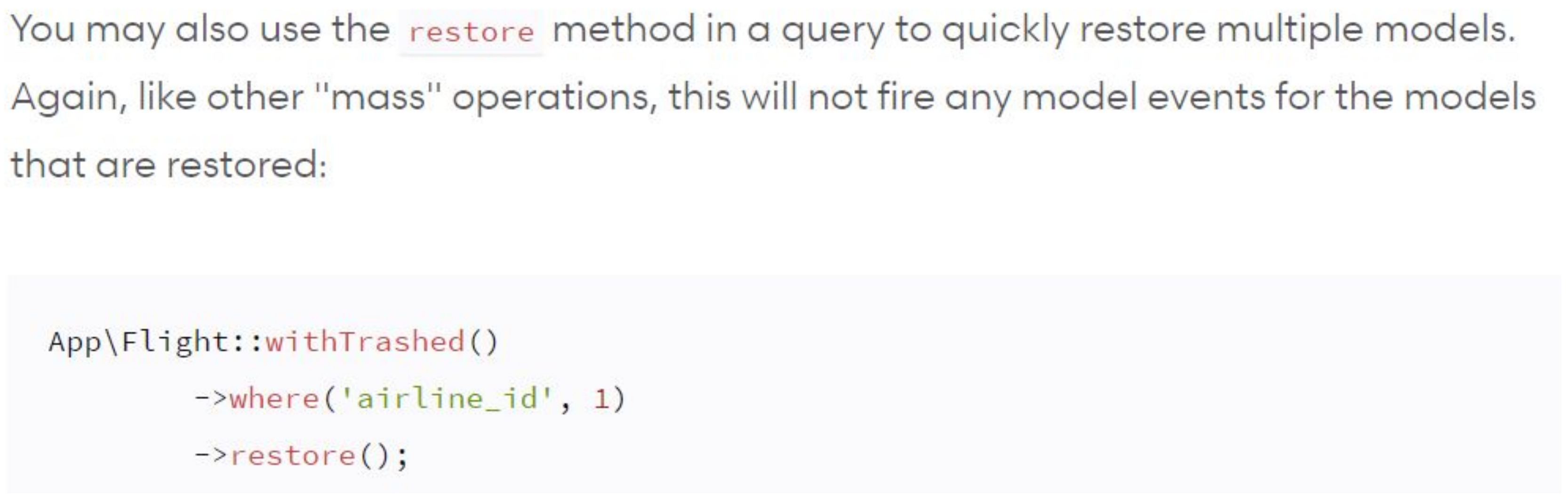
As noted above, soft deleted models will automatically be excluded from query results. However, you may force soft deleted models to appear in a result set using the `withTrashed` method on the query:

```
$flights = App\Flight::withTrashed()
                ->where('account_id', 1)
                ->get();
```

Universitas Surabaya

Restore Deleted Data

We use the restore() method

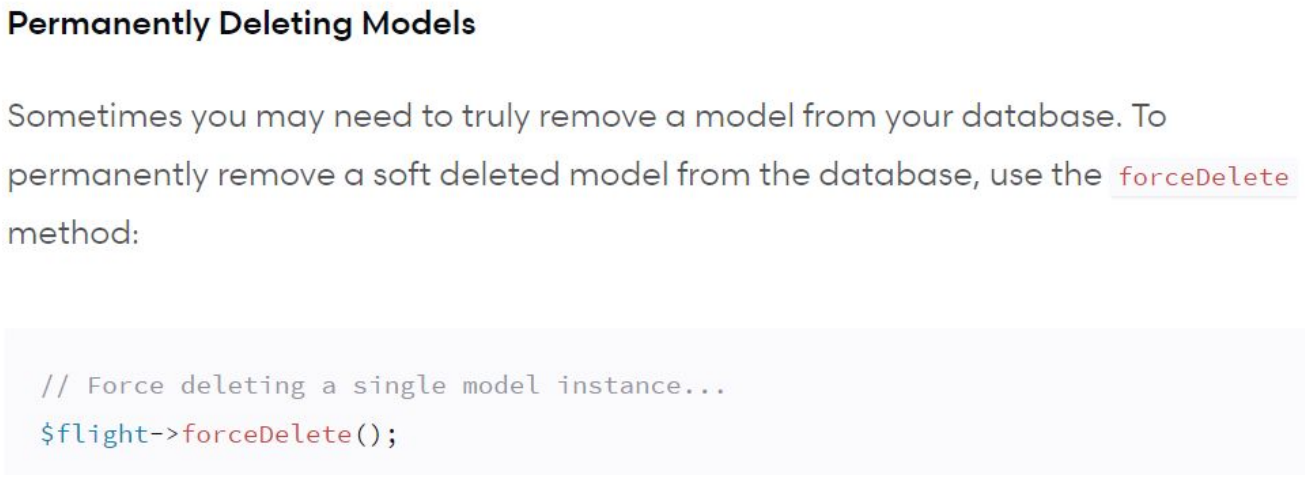


You may also use the `restore` method in a query to quickly restore multiple models. Again, like other "mass" operations, this will not fire any model events for the models that are restored:

```
App\Flight::withTrashed()
        ->where('airline_id', 1)
        ->restore();
```

If you want to empty the "recycle-bin"

If you want to completely delete data, use the force Delete() method.



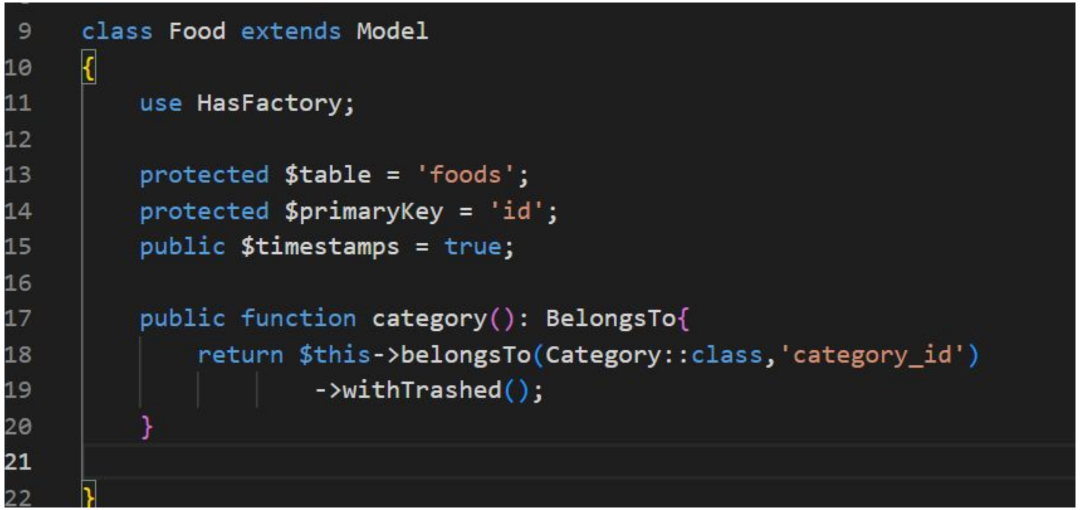
Permanently Deleting Models

Sometimes you may need to truly remove a model from your database. To permanently remove a soft deleted model from the database, use the `forceDelete` method:

```
// Force deleting a single model instance...
$flight->forceDelete();
```

This softDelete will impact into another Data

Try to access ‘food’ page after you finish delete something that has relationship with category. They show error about “non-property” of something.
This happened because your default Relationship will change to adapt the SoftDelete behavior. To fix this, you can use `withTrashed()`



```php
class Food extends Model
{
    use HasFactory;

    protected $table = 'foods';
    protected $primaryKey = 'id';
    public $timestamps = true;

    public function category(): BelongsTo{
        return $this->belongsTo(Category::class,'category_id')
                ->withTrashed();
    }

}
```

Homework

Implement Update and Delete function inside

- Food page
- Category page
- Order page

Your progress is to achieve your Final Project Goals
Please do it carefully and consistently!

Thank You.