Instruction for Lab3

The purpose of this Lab assignment is to provide a hands-on experience in developing basic Laravel functionality. It will guide you through basic steps of creating a website, adding routes, controllers, views and models to the application. You will also implement CRUD functionality of a model.

The sample application will be hosted in address `http://www.lab3.dev`.

Prerequisites

PHP

In this lab, it is assumed that PHP command-line executable is installed in the computer and is added to PATH. That is, the following command outputs php version and does not display an error message:

```
php --version
```

Composer

The Lab requires that **Composer** is installed in your computer.

To check, if Composer is available, run following command line in terminal app of your computer:

```
composer --version
```

It should output the actual version number, not an error message. If it is not there, follow the guidelines:

- Windows: https://getcomposer.org/doc/00-intro.md#installation-windows
- Linux & Mac: https://getcomposer.org/doc/00-intro.md#installation-linux-unix-macos

Configuring the web server

In this step, you have to create a web site that runs locally on your computer and can be accessed using local web browser at http://www.lab3.dev.

Wamp.Net configuration

- launch Wamp.Net Manager
- make sure that
  - PHP 8.1.9. is installed and running
  - Nginx web server is installed and running
  - MySQL 10.0.30 is installed and running
- in the "Sites" section of Wamp.Net Manager add a new website:
  - Domain name: lab3.dev
  - Aliases: www
  - Document root: [leave the default value]
  - Web server: pick nginx 1.23.1
  - PHP: pick PHP 8.1.9
  - Click "create".
- Edit the newly created site and change its document root to `C:\Wamp.NET\sites\lab3.dev\public` (i.e., append the word `public`)

- Note: right now, the site will not open, the specified document root does not exist yet.
- This document further references the location `C:\Wamp.NET\sites\lab3.dev\` (without the `public` part) as *Laravel installation directory*.

XAMPP / MAMP configuration

- update the Windows `hosts` file to include www.lab3.dev by adding a new line at the end:

  ```
  127.0.0.1   www.lab3.dev
  ```

  For instructions on updating hosts file see this tutorial.
- Create a directory in your computer that will be your *Laravel installation directory* for this Lab (e.g. `C:\xampp\htdocs\lab3.dev\` in XAMPP, or `/Applications/MAMP/htdocs/lab3.dev` in MAMP)
- Update Apache configuration to add a new virtual host. See a tutorial on how to do this. When setting up the virtualhost, remember to set the document root as [*Laravel installation directory*]**/public**.

Creating a Laravel site

Now that you have a web server configured, you can set up Laravel. In the command line terminal, navigate to your Laravel installation directory (this and the following samples assume that you have Wamp.Net configuration where *Laravel installation directory* is `C:\Wamp.NET\sites\lab3.dev\`):

```
cd C:\Wamp.NET\sites\lab3.dev\
```

Install Laravel using Composer `create-project` command. Mind the trailing `.` in the command; it instructs Composer to install the files in current directory.

```
composer create-project laravel/laravel .
```

Now you should be able to open www.lab3.dev in your web browser and see the default installation page.

Creating the project database

Database and users

Connect to the local MySQL server using the same tool you used in Lab2 - for example MySQL Workbench or PhpMyAdmin.

Using the following script, create a user "lab3" and a database "lab3":

```
CREATE USER 'lab3'@'%' IDENTIFIED WITH mysql_native_password BY 'p@assword@lab3';

GRANT USAGE ON *.* TO 'lab3'@'%';

CREATE DATABASE IF NOT EXISTS `lab3`;

GRANT ALL PRIVILEGES ON `lab3`.* TO 'lab3'@'%';
```

Configuring Laravel to use Lab3 database

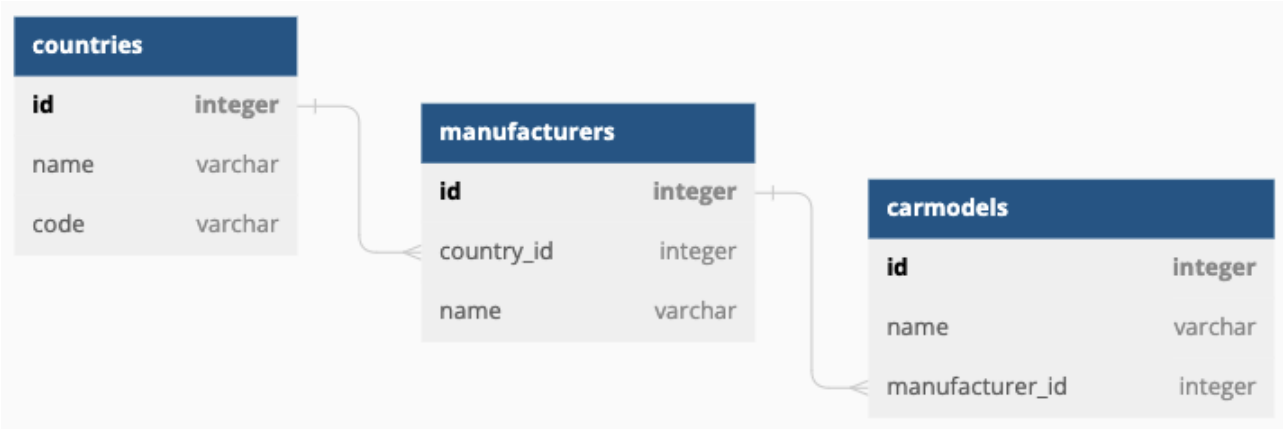
Update the .env file in the root of your *Laravel installation folder*. Locate configuration section that starts with keys `DB_CONNECTION`, and set the following keys:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lab3
DB_USERNAME=lab3
DB_PASSWORD=p@assword@lab3
```

Now your Laravel application will be able to connect to the database.

Creating the model

The lab database will have the following data entities, similar to ones we had in Lab2: `country`, `manufacturer`, `model` (for the sake of clarity, we'll use the term `carmodel` to avoid confusion with Laravel *models*).



Generation of model classes

Use `artisan` command line tool to create the models using the following commands:

```
php artisan make:model Country --migration
php artisan make:model Manufacturer --migration
php artisan make:model Carmodel --migration
```

Each command creates several new items:

- model class in *Laravel installation folder*/app/models
- migration file in *Laravel installation folder*/database/migrations

Adding fields to DB migrations

Update the migration files (found in *Laravel installation directory*/database/migrations) created by `artisan` to include data columns:

- file `2023_..._create_countries_table.php` Edit the `up()` method, add the following after `$table->id()`:

```
public function up(): void
{
    Schema::create('countries', function (Blueprint $table) {
```

```
            $table->id();
            $table->string('name');
            $table->string('code');
            $table->timestamps();
        });
    }
```

- file `2023_..._create_manufacturers_table.php` Edit the `up()` method, add the following after `$table->id()`:

```
public function up(): void
{
    Schema::create('manufacturers', function (Blueprint $table) {
        $table->id();
        $table->foreignId('country_id')->constrained()->cascadeOnDelete();
        $table->string('name');
        $table->timestamps();
    });
}
```

- file `2023_..._create_carmodels_table.php` Edit the `up()` method, add the following after `$table->id()`:

```
public function up(): void
{
    Schema::create('carmodels', function (Blueprint $table) {
        $table->id();
        $table->foreignId('manufacturer_id')->constrained()->cascadeOnDelete();
        $table->string('name', 100);
        $table->timestamps();
    });
}
```

Using the command line `artisan` tool, execute the migrations to create the initial version of your database:

```
php artisan migrate
```

Check the structure of your database. Besides the standard Laravel tables (" failed_jobs", " migrations", " password_reset_tokens"...), you will also find newly created tables "carmodels", "countries" ," manufacturers".

Foreign key relationships in models

Update your model classes to include references to other models. Edit files in *Laravel installation directory*/App/models:

- file `Country.php`

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Database\Eloquent\Model;

class Country extends Model
{
    use HasFactory;
    protected $fillable = ['name'];
    public function manufacturers(){
        return $this->hasMany(Manufacturer::class);
    }
}
```

- File `Manufacturer.php`

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Manufacturer extends Model
{
    use HasFactory;

    protected $fillable = ['name'];

    public function country(){
        return $this->belongsTo(Country::class);
    }
    public function carmodels(){
        return $this->hasMany(Carmodel::class);
    }
}
```

-File `Carmodel.php`

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Carmodel extends Model
{
    use HasFactory;

    protected $fillable = ['name'];

    public function manufacturer()
    {
        return $this->belongsTo(Manufacturer::class);
    }
}
```

Filling the model with sample data

Update the *Laravel installation directory*/Database/seeders/DatabaseSeeder.php file to the following. See how there are different approaches for handling relations, but the result is similar:

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Country;
use App\Models\Manufacturer;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        Country::create(['name' => 'Germany', 'code'=>'DE']);
        Country::create(['name' => 'Italy', 'code'=>'IT']);
        Country::create(['name' => 'France', 'code'=>'FR']);

        #approach #1 - create instance of manufacturer, call save on collection
        $france = Country::where('name', 'France')->first();
        $renault = new Manufacturer();
        $renault->name = 'Renault';
        $france->manufacturers()->save($renault);

        #approach #2 - use "create"  shortcut of collection
        $france->manufacturers()->create(['name' => 'Peugeot']);

        #approach #3 - calling "associate" on sub-object
        $germany = Country::where('name', 'Germany')->first();
        $audi = new Manufacturer();
        $audi->name = 'Audi';
        $audi->country()->associate($germany);
        $audi->save();
    }
}
```

Task #1

Extend the code of database seeder to:

- add Spain and Japan to list of countries
- add Seat (Spain), Toyota (Japan), Volkswagen (Germany) to list of manufacturers
- add the following car models to `carmodel`:
    - Passat, Golf, Multivan to manufacturer Volkswagen
    - A4, A6, Q3, Q4, Q5 to manufacturer Audi
    - Toledo, Ibiza to manufacturer Seat

Execute the seeder with the following command (keep in mind that the `:fresh` mode means deleting all current data and re-creating data structures):

```
php artisan migrate:fresh --seed
```

The web application

The navigation in Lab3 web application will start from a list of countries, then provide links to related objects (manufacturers, models) and related functionality (e.g., edit country name).

Controller for `countries`

Using artisan, create a resource controller:

```
php artisan make:controller CountryController --resource
```

It creates a new file `CountryController.php` in *Laravel installation folder*\app\Http\Controllers. Edit the file change its beginning accordingly:

```
<?php

namespace App\Http\Controllers;

use App\Models\Country;
use Illuminate\Http\Request;

class CountryController extends Controller
{
    /**
     * Display a listing of the countries.
     */
    public function index()
    {
        $countries = Country::all();
        return view('countries', compact('countries'));
    }
```

Inside this `index` method, a full list of countries is loaded from the database and passed to a view called `countries.

View for listing all countries

In the folder *Laravel installation folder*\resources\views, create a file `countries.blade.php` with the following contents:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Countries</title>
</head>

<body>
    <h1>Pick a country you like</h1>
    @if (count($countries) == 0)
```

```
        <p class='error'>There are no records in the database!</p>
    @else
        <ul>

            @foreach ($countries as $country)
                <li>
                    {{ $country->code }} - {{ $country->name }}
                </li>
            @endforeach

        </ul>
    @endif
</body>

</html>
```

Routing requests to a controller

To see a list of countries, routing has to be changed. Edit the file *Laravel installation folder*/routes/web.php

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\CountryController;

// Route::get('/', function () {
//     return view('welcome');
// });

Route::redirect('/', 'country');
Route::resource('country', CountryController::class);
```

1. requests looking for site root (/) are redirected to /country
2. requests looking for /country/... are served by controller `CountryController`.

After this change, you can open http://www.lab3.dev and see a list of countries.

Implementing "CRUD" in the application

Deleting a country

Edit the `CountryController.php` file (in *Laravel installation folder*\app\Http\Controllers\ folder) and change the `destroy()` method accordingly:

```
public function destroy(string $id)
{
    Country::findOrfail($id)->delete();
    return redirect('country/');
}
```

In the migration it was set to `cascadeOnDelete()` which means that deleting a country will immediately delete all the manufacturers in that country (and deleting a manufacturer will delete all of its car models).

To cause the `destroy()` operation, an action button is needed.

Update the web.php file by adding Routes to GET and DELETE requests.

The first route is a GET request that maps to the index method of the ManufacturerController class, which retrieves a list of manufacturers.

The second route is a DELETE request that maps to the destroy method of the CountryController class, which deletes a specific country based on its ID.

```
Route::get('/manufacturers', [App\Http\Controllers\ManufacturerController::class,
'index'])->name('manufacturers.index');
Route::delete('/countries/{id}', [App\Http\Controllers\CountryController::class,
'destroy'])->name('countries.destroy');
```

Update the `countries.blade.php` file by adding a "Delete" button in all rows:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Countries</title>
</head>

<body>
    <h1>Pick a country you like</h1>
    @if (count($countries) == 0)
        <p style="color: red;"> There are no records in the database!</p>
    @else
        <ul>

            @foreach ($countries as $country)
                <li>
                    {{ $country->code }} -
                    <a href="{{ route('manufacturers.index', ['countryslug' =>
$country->code]) }}">{{ $country->name }}</a>

                    <form method="POST"
                        action="{{ route('countries.destroy', $country->id) }}">
                        @csrf
                        @method('DELETE')
                        <button type="submit" value="delete">Delete</button>
                    </form>
                </li>
            @endforeach

        </ul>
    @endif
</body>
</html>
```

Open the list of countries at http://www.lab3.dev/country and click the "Delete" button next to France. See if France disappears from the list.

Viewing manufacturers in a particular country

Using `artisan` command, create a controller for manufacturers:

```
php artisan make:controller ManufacturerController --resource
```

Change the routes (*Laravel installation folder*/routes/web.php) to route *most* requests to standard controller methods. The 'index' and 'create' methods will be have specific routes - ones including country code.

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\CountryController;
use App\Http\Controllers\ManufacturerController;

Route::redirect('/', 'country');
Route::resource('country', CountryController::class);

Route::resource('manufacturer', ManufacturerController::class, ['except' =>
['index', 'create']]);
Route::get('{countryslug}/manufacturer', [ManufacturerController::class, 'index']);
```

Change the beginning of `ManufacturerController.php` (found in *Laravel installation folder*/app/HTTP/controllers/) controller file to create the specific index method

```
<?php

namespace App\Http\Controllers;

use App\Models\Country;
use App\Models\Manufacturer;
use Illuminate\Http\Request;

class ManufacturerController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index($countryslug)
    {
        //look up the country by its 2-letter code
        $country = Country::where('code','=', $countryslug)->first();

        #use Eloquent relations to find all manufacturers in that country
        $manufacturers = $country->manufacturers()->get();

        return view('manufacturers', ['country' => $country, 'manufacturers' =>
$manufacturers]);
    }
```

Create a view file `manufacturers.blade.php` in (*Laravel installation folder*/resources/views), with the following contents:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Manufacturers in {{$country->name}}</title>
</head>

<body>
    <h1>Manufacturers in {{$country->name}}</h1>
    @if (count($manufacturers) == 0)
        <p style="color: red;"> There are no records in the database!</p>
    @else
        <ul>

            @foreach ($manufacturers as $manufacturer)
                <li>
                   {{ $manufacturer->name }}
                </li>
            @endforeach
        </ul>
    @endif
</body>

</html>
```

Try to access the address http://www.lab3.dev/de/manufacturer. It should look up the "de" fragment and recognize it as Germany, then display all manufacturers in that country.

Adding a new manufacturer

When the users visit the page of manufacturers in particular country, they might want to add yet another manufacturer to that list. For instance, to register a new manufacturer in Germany, the URL address of the form might be http://www.lab3.dev/de/manufacturer/create/

To achieve that, first edit the routes file (*Laravel installation folder*/routes/web.php) and add another routing rule:

```
Route::get('{countryslug}/manufacturer/create', [ManufacturerController::class,
'create']);
```

Change the method `create()` in `ManufacturerController.php` (found in *Laravel installation folder*/app/HTTP/controllers/) controller file:

```
public function create($countryslug)
{
    $country = Country::where('code','=', $countryslug)->first();
    return view('manufacturer_new', compact('country'));
}
```

Create a view file `manufacturer_new.blade.php` in *Laravel installation folder*/resources/views:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>New manufacturer in {{ $country->name }}</title>
</head>

<body>
    <h1>New manufacturer in {{ $country->name }}</h1>
    <form method="POST" action={{
action([App\Http\Controllers\ManufacturerController::class, 'store']) }}>
        @csrf
        <input type="hidden" name="country_id" value="{{ $country->id }}">
        <label for='manufacturer_name'>Manufacturer name</label>
        <input type="text" name="manufacturer_name" id="manufacturer_name">
        <button type="submit" value="Add">Save</button>
    </form>
</body>
</html>
```

You can now test that the new input form can be open at https://www.lab3.dev/de/manufacturer/create (however, it does not work yet).

To actually save the new manufacturer data in database, edit the `store()` method of `ManufacturerController.php` (found in *Laravel installation folder*/app/HTTP/controllers/) controller file:

```
 public function store(Request $request)
    {
        $manufacturer = new Manufacturer();
        $manufacturer->name = $request->manufacturer_name;
        $manufacturer->country_id = $request->country_id;
        $manufacturer->save();

        #to perform a redirect back, we need country code from ID
        $country = Country::findOrFail($request->country_id);
        $action = action([ManufacturerController::class, 'index'], ['countryslug' =>
$country->code]);
        return redirect($action);
    }
```

To create a link to the form where a new manufacturer can be registered, edit the `manufacturers.blade.php` file in *Laravel installation folder*/resources/views to add a link after the list of current manufacturers:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Manufacturers in {{$country->name}}</title>
</head>

<body>
    <h1>Manufacturers in {{$country->name}}</h1>
    @if (count($manufacturers) == 0)
```

```
        <p style="color: red;"> There are no records in the database!</p>
    @else
        <ul>
            @foreach ($manufacturers as $manufacturer)
                <li>
                   {{ $manufacturer->name }}
                </li>
            @endforeach
        </ul>
        <a href="{{ action([App\Http\Controllers\ManufacturerController::class,
'create'],['countryslug' => $country->code])}}">Add new manufacturer</a>
    @endif
</body>

</html>
```

Task #2

Implement the DELETE functionality in Manufacturer controller.

Editing existing resources

To allow users to edit an existing manufacturer object (change its 'name' attribute), change the `ManufacturerController` class (found in *Laravel installation folder*/app/HTTP/controllers/) by setting the body of `edit` and `update` functions.

```
 /**
     * Show the form for editing the specified resource.
     */
    public function edit(string $id)
    {
        $manufacturer = Manufacturer::findOrFail($id);
        return view('manufacturer_edit', compact('manufacturer'));
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, string $id)
    {
        $manufacturer = Manufacturer::findOrFail($id);
        $manufacturer->name = $request->manufacturer_name;
        $manufacturer->save();
        return redirect(action([ManufacturerController::class, 'index'],
['countryslug' => $manufacturer->country->code]));
    }
```

Then, create a new view `manufacturer_edit.blade.php` in *Laravel installation folder*/resources/views:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
    <title>Editing manufacturer {{ $manufacturer->name }}</title>
  </head>

  <body>
    <h1>Editing manufacturer {{ $manufacturer->name }}</h1>
    <form method="POST"
        action={{ action([App\Http\Controllers\ManufacturerController::class,
  'update'], [ 'manufacturer' => $manufacturer]) }}>
        @csrf
        @method('put')
        <label for='manufacturer_name'>Manufacturer name</label>
        <input type="text" name="manufacturer_name" id="manufacturer_name" value="{{
  $manufacturer->name }}">
        <button type="submit" value="Update">Save changes</button>
    </form>
  </body>

  </html>
```

Now you can see that manufacturer objects can be changed by accessing https://www.lab3.dev/manufacturer/3/edit

Task #3

Modify the controllers and routes, add view files as needed to provide the following functionality:

- When viewing the list of manufacturers in particular country, each item in the list has a link to http://www.lab3.dev/manufacturer/{id}/models - this will open a list of all car models of particular manufacturer
- When viewing all models of particular manufacturer, there is a link "Add new model" which opens a 'new item' form and allows saving the data.