

MODUL 13 MENGELOLA DATABASE PADA CODE IGNITER

13.1. Deskripsi Singkat

CodeIgniter adalah salah satu framework PHP yang populer dan mudah digunakan untuk pengembangan aplikasi web. Salah satu fitur utama CodeIgniter adalah kemudahan dalam mengelola dan berinteraksi dengan database. Dalam pengembangan web modern, manajemen database merupakan aspek kunci yang harus dipahami dan dikuasai. CodeIgniter menyediakan berbagai fitur dan utilitas yang memudahkan dalam melakukan manajemen database, mulai dari membuat koneksi, melakukan query database, hingga mengelola struktur database.

13.2. Tujuan Praktikum

Setelah menyelesaikan praktikum pada modul ini, mahasiswa diharapkan dapat melakukan koneksi dan konfigurasi database pada CodeIgniter, serta mengelola dan berinteraksi dengan data dalam database menggunakan framework CodeIgniter. Dengan pemahaman yang baik tentang manajemen database pada CodeIgniter, mahasiswa diharapkan dapat mengembangkan aplikasi web yang handal dan efisien.

13.3. Material Praktikum

Praktikum menggunakan editor teks seperti Notepad, Notepad++, Visual Studio Code, atau aplikasi editor teks lainnya. Pada praktikum ini juga menggunakan tool XAMPP yang telah mem-bundle PHP dengan teknologi pendukungnya. Selain itu juga diperlukan framework CodeIgniter yang proses instalasinya sudah dijelaskan pada modul 12.

13.4. Kegiatan Praktikum

13.4.1. Command Line

Pada materi sebelumnya untuk menjalankan aplikasi Codeigniter kita menggunakan perintah:

```
php spark serve
```

yang merupakan salah satu perintah dari command line yang tersedia pada Codeigniter 4. Untuk melihat command line lain silahkan ketikkan perintah:

```
php spark
```

maka akan muncul daftar command line yang tersedia pada Codeigniter 4.

Beberapa perintah dikelompokkan ke dalam masing-masing grup seperti Cache, CodeIgniter, Database, dsb. Jika ingin mengetahui penjelasan detail dari masing-masing command line, Anda bisa menggunakan perintah dengan format `php spark` diikuti dengan nama command line, seperti contoh berikut.

```
php spark help db:create
```

Pada output akan muncul cara penggunaan, deskripsi, dan opsi argumen yang tersedia. Kita akan menggunakan beberapa command line dan mengetahui kegunaannya khususnya yang berkaitan dengan pengelolaan database yang akan kita bahas pada bagian berikutnya.

13.4.2. Membuat Project Baru dan Konfigurasi

1. Pada materi ini mahasiswa akan membuat aplikasi CRUD sederhana. Silakan membuat aplikasi CodeIgniter baru khusus untuk materi ini. Silakan buka terminal/cmd dan arahkan ke direktori `htdocs` lalu ketikkan perintah di bawah ini.

```
composer create-project codeigniter4/appstarter portfolio-app
```

2. Setiap kali Anda membuat project baru di CodeIgniter maka Anda perlu melakukan beberapa pengaturan konfigurasi. Silakan copy file `env` dan rename dengan nama `.env` lalu pada bagian `ENVIRONMENT` ubah menjadi seperti berikut ini:

```
#-----  
# ENVIRONMENT  
#-----  
  
CI_ENVIRONMENT = development
```

Kode di atas akan mengatur variabel `CI_ENVIRONMENT` dengan value `development` untuk mengaktifkan debugging sehingga jika terdapat error akan ditampilkan detail errornya.

3. Selanjutnya tambahkan variabel baru pada bagian APP seperti berikut:

```
#-----  
# APP  
#-----  
  
app.indexPage = ''
```

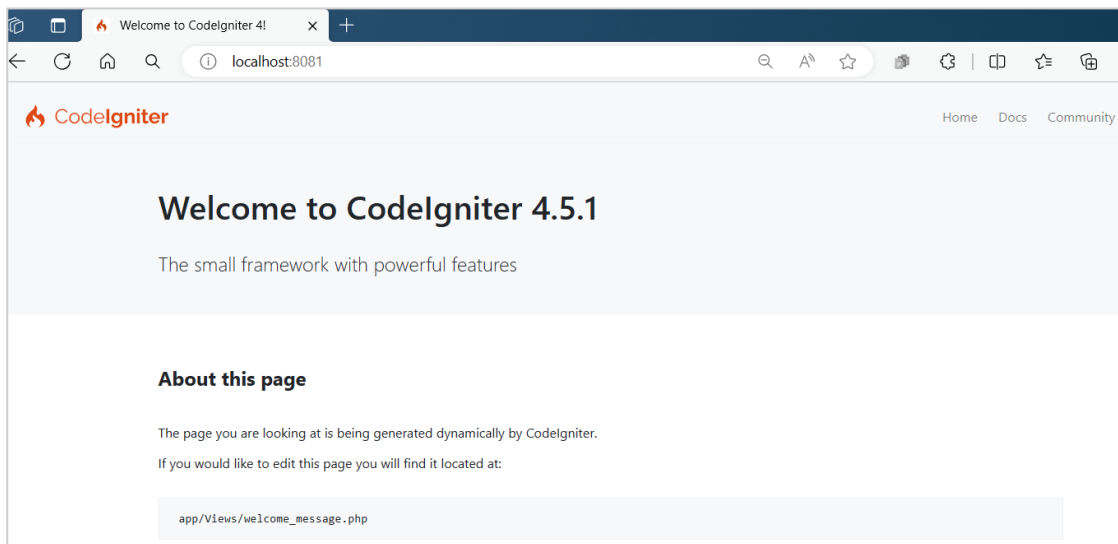
4. Pada bagian DATABASE silahkan mengatur group defaultnya menggunakan MySQL sebagai databasenya, isikan nama database, username, dan password yang Anda gunakan. Pastikan Anda sudah membuat terlebih dahulu database di MySQL dengan nama database portfolio_app.db

```
#-----  
# DATABASE  
#-----  
  
database.default.hostname = localhost  
database.default.database = portfolio_app.db  
database.default.username = root  
database.default.password =  
database.default.DBDriver = MySQLi  
database.default.DBPrefix =  
database.default.port      = 3306
```

5. Simpan perubahan di atas. Melalui command line arahkan direktori ke portfolio-app lalu coba jalankan aplikasi perintah berikut

```
php spark serve
```

Buka browser dengan alamat <http://localhost:8080> atau 8081 sesuai dengan port aplikasi CodeIgniter yang berjalan di sistem Anda. Halaman default akan terlihat seperti gambar di bawah ini.



13.4.3. Membuat Migration dan Model

Pada bagian ini Anda akan membuat satu tabel dengan nama **portfolios** yang strukturnya seperti berikut.

Field Nama	Type	Key
id	INT (5) autoincrement	Primary
title	VARCHAR (100)	
description	TEXT	
image	VARCHAR (255)	
created_at	DATETIME	
updated_at	DATETIME	
deleted_at	DATETIME	

1. Buat terlebih dahulu file Migration untuk tabel **portfolios**. Ketikkan command line berikut pada terminal/cmd untuk membuat file Migration baru. Jangan lupa arahkan terlebih dahulu ke direktori portfolio-app.

```
php spark make:migration create_portfolios_table
```

2. Buka file yang dibuat pada langkah 1 yang terdapat pada folder app/Database/Migrations dan lakukan perubahan menjadi seperti baris kode di bawah ini.

```
<?php

namespace App\Database\Migrations;

use CodeIgniter\Database\Migration;

class CreatePortfoliosTable extends Migration
{
    public function up()
    {
        $this->forge->addField([
            'id' => [
                'type' => 'INT',
                'constraint' => 5,
                'unsigned' => true,
                'auto_increment' => true,
            ],
            'title' => [
                'type' => 'VARCHAR',
                'constraint' => '100',
            ],
            'description' => [
                'type' => 'TEXT',
                'null' => true,
            ],
            'image' => [
                'type' => 'VARCHAR',
```

```

        'constraint' => '255',
    ],
    'created_at DATETIME DEFAULT CURRENT_TIMESTAMP',
    'updated_at DATETIME DEFAULT CURRENT_TIMESTAMP',
    'deleted_at DATETIME DEFAULT NULL',
]);
$this->forge->addKey('id', true);
$this->forge->createTable('portfolios');
}

public function down()
{
    $this->forge->dropTable('portfolios');
}
}

```

Pada method up Anda menambahkan kode untuk membuat tabel **portfolios** dengan field-field yang sesuai dengan struktur tabel di atas. Sedangkan pada method down Anda menambahkan kode untuk menghapus tabel **portfolios**.

3. Jalankan migration dengan command line berikut.

```
php spark migrate
```

Output:

```

CodeIgniter v4.5.1 Command Line Tool - Server Time: 2024-05-03
02:17:06 UTC+00:00

Running all new migrations...
    Running: (App) 2024-05-03-
020432_App\Database\Migrations\CreatePortfoliosTable
Migrations complete.

```

4. Berikutnya Anda perlu membuat class Model dan Entity untuk tabel **portfolios**. Silakan buat dulu kedua filenya dengan command line dengan mengetikkan perintah berikut.

```
php spark make:model PortfolioModel
```

Output:

```

CodeIgniter v4.5.1 Command Line Tool - Server Time: 2024-05-03
02:24:42 UTC+00:00

File created: APPPATH\Models\PortfolioModel.php

```

5. Selanjutnya silakan membuat file class Entity baru dengan menggunakan perintah command line berikut ini.

```
php spark make:entity Portfolio
```

Output:

```
CodeIgniter v4.5.1 Command Line Tool - Server Time: 2024-05-03  
02:28:19 UTC+00:00
```

```
File created: APPPATH\Entities\Portfolio.php
```

6. Anda tidak perlu melakukan perubahan pada class Entity. Cukup lakukan perubahan pada Model. Buka file `app/Models/PortfolioModel.php` lalu hapus beberapa property yang tidak diperlukan dan sesuaikan beberapa properti lainnya.

```
<?php  
  
namespace App\Models;  
  
use CodeIgniter\Model;  
  
class PortfolioModel extends Model  
{  
    protected $table          = 'portfolios';  
    protected $primaryKey     = 'id';  
    protected $useAutoIncrement = true;  
  
    protected $returnType     = 'App\Entities\Portfolio';  
    protected $useSoftDeletes = true;  
    protected $allowedFields  = ['title', 'description', 'image'];  
  
    // Dates  
    protected $useTimestamps = true;  
    protected $dateFormat    = 'datetime';  
    protected $createdField   = 'created_at';  
    protected $updatedField   = 'updated_at';  
    protected $deletedField   = 'deleted_at';  
}
```

Pada kode di atas Anda sudah menghapus beberapa property yang tidak diperlukan kemudian mengubah property seperti `$returnType`, `$useSoftDeletes`, `$allowedFields` dan `$useTimestamps`.

13.4.4. Membuat View Layout

1. Pada bagian ini Anda akan membuat view layout yang terdiri dari tiga bagian yaitu: navbar, content, dan footer. Buat folder baru dengan nama layouts di dalam folder app/Views.
2. Tambahkan file baru dengan nama main.php pada folder app/Views/layouts dan tuliskan kode di bawah ini.

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <!-- Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstra
p.min.css" rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">

    <title>CI4 Portfolio App</title>

    <style>
      body {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <?php echo $this->include('partials/navbar'); ?>

      <!-- Section content-->
      <?php echo $this->renderSection('content') ?>
      <!-- end Section Content -->

      <div class="row mt-4">
        <?php echo $this->include('partials/footer'); ?>
      </div>

      <?php echo $this->renderSection('scripts') ?>
    </body>
  </html>
```

Pada kode di atas Anda membuat layout dengan struktur HTML dan mengimpor file css nya Bootstrap v5 pada bagian header dan beberapa style css lainnya. Di

bagian body, Anda memasukkan view navbar dan footer serta menambahkan bagian untuk section content dan script.

3. Selanjutnya Anda perlu membuat view navbar dan footer yang sudah dimasukkan pada bagian layout. Buat folder dengan nama partials di dalam folder app/Views dan tambahkan dua file yaitu masing-masing dengan nama navbar.php dan footer.php.

app/Views/partial/navbar.php

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">

    <div class="collapse navbar-collapse"
id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page"
href="#">My Portfolio</a>
        </li>
        <li class="nav-item">
          <a class="btn btn-outline-primary"
href="#">Create New</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

app/Views/partial/footer.php

```
<hr>
<div class="col text-center">
  <footer class="text-muted">stispbw &copy; 2024</footer>
</div>
```

13.4.5. CRUD: Menambah Data Portfolio

1. Pada bagian ini Anda akan membuat halaman untuk menambahkan data portfolio. Silakan membuat file Controller baru. Buka terminal/cmd pastikan direktori sudah mengarah ke direktori project Anda. Selanjutnya jalankan perintah berikut pada command line.

```
php spark make:controller PortfolioController
```

Output:

```
CodeIgniter v4.5.1 Command Line Tool - Server Time: 2024-05-03
02:54:39 UTC+00:00
```

```
File created: APPPATH\Controllers\PortfolioController.php
```


2. Buka file `app/Controllers/PortfolioController.php` dan tambahkan method `create` di dalam class `PortfolioController`.

```
public function create()
{
    echo view('portfolio/create');
}
```

Pada kode tersebut Anda mengembalikan response dari method `create` dalam bentuk View.

3. Selanjutnya tambahkan route untuk halaman form tambah portfolio. Buka file `app/Config/Routes.php` pada bagian Route Definitions tambahkan route baru seperti di bawah ini.

```
$routes->get('/', 'Home::index');
$routes->get('portfolio/create', 'PortfolioController::create');
```

Simpan perubahan di atas dan buka browser Anda dengan alamat <http://localhost:8080/portfolio/create>

Ternyata muncul error seperti ini.

```
CodeIgniter\View\Exceptions\ViewException
Invalid file: portfolio/create.php
```

4. Error di atas muncul karena Anda belum membuat file View `portfolio/create.php`, sekarang silahkan membuatnya. Tambahkan folder baru dengan nama `portfolio` di dalam folder `app/Views` lalu buat file `create.php` dan tuliskan baris kode di bawah ini.

```
<?php echo $this->extend('layouts/main') ?>

<?php echo $this->section('content'); ?>
<h1>Create Portfolio</h1>

<form method="post" action="/portfolio/create"
enctype="multipart/form-data">
    <?php echo csrf_field() ?>
    <div class="mb-3">
        <label class="form-label">Title</label>
        <input type="text" class="form-control" name="title"
value="<?php echo set_value('title', '');?>">
    </div>
    <div class="mb-3">
        <label class="form-label">Description</label>
        <textarea name="description" class="form-control" rows="4"><?php
echo set_value('description', '');?></textarea>
    </div>
    <div class="mb-3">
        <label class="form-label">Image</label>
        <input type="file" class="form-control" name="image">
    </div>
```

```

        <button type="submit" class="btn btn-primary">Save</button>
        <a class="btn btn-secondary" href="{<?php echo
site_url('portfolio');?>}">Back</a>
    </form>
    <?php echo $this->endSection(); ?>

```

Pada kode di atas, Anda membuat sebuah form HTML dengan input text untuk field title, textarea untuk field description, input file untuk field image dan dua button di dalam section content. Pada View di atas Anda juga menggunakan layout dari file View app/Views/layouts/main.php.

Buka kembali browser Anda dan ketikkan alamat <http://localhost:8080/portfolio/create> dan ternyata masih muncul error seperti ini.

```

Error
Call to undefined function set_value()

```

5. Ternyata Anda menggunakan salah satu helper dari Form helper yaitu `set_value` untuk mengeset value pada input text dan textarea. Namun perlu diingat kembali bahwa helper tidak autoload secara default. Untuk mengatasinya Anda cukup meload Form Helper dengan menambahkannya pada property `$helpers` di dalam Controller seperti berikut ini.

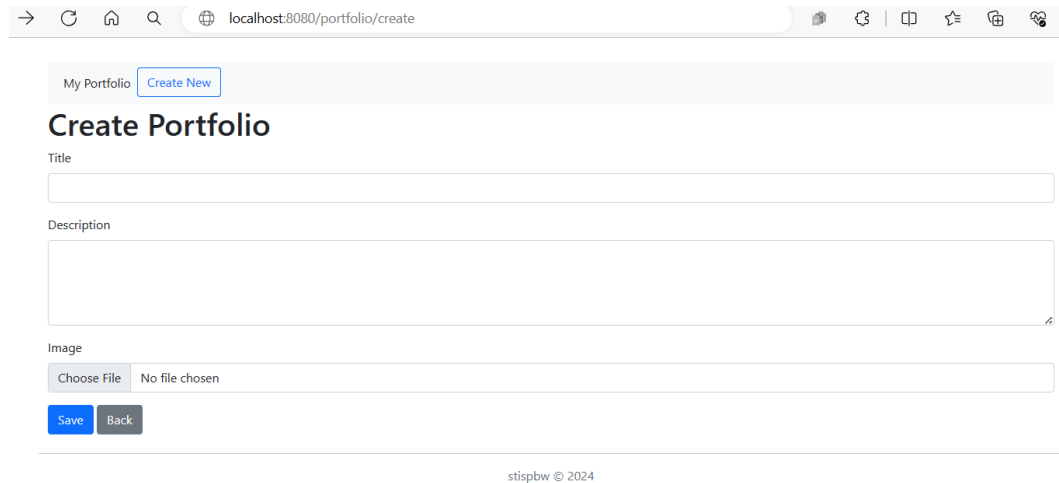
app/Controllers/PortfolioController.php

```

class PortfolioController extends BaseController
{
    public $helpers = [
        'form'
    ];
}

```

Refresh kembali halaman <http://localhost:8080/portfolio/create> dan seharusnya akan muncul tampilan form create portfolio.



6. Berikutnya, Anda akan mengubah route dan method create pada PortfolioController untuk menangani request ketika button save pada form di atas diklik. Pada app/Config/Routes.php lakukan perubahan pada baris kode ini:

```
$routes->get('/portfolio/create', 'PortfolioController::create');
```

menjadi seperti ini:

```
$routes->match(['get', 'post'], '/portfolio/create',  
'PortfolioController::create');
```

Maksud kode di atas adalah ketika Anda mengirimkan request untuk menampilkan form atau ketika form disubmit maka akan menggunakan route yang sama yaitu /portfolio/create dan diarahkan ke Controller dan method yang sama juga tetapi dengan HTTP method yang berbeda (get atau post). Dengan kata lain baris kode di atas seperti Anda menggabungkan dua route dengan satu definisi route saja.

```
GET /portfolio/create -> Class PortfolioController, method create  
POST /portfolio/create -> Class PortfolioController, method create
```

7. Pada bagian Controller, kita juga perlu mengubah isi dari method create pada class PortfolioController.

app/Controllers/PortfolioController.php

```
public function create()  
{  
    // jika HTTP method adalah "POST" dan data yang dikirim  
    valid, maka akan melakukan proses selanjutnya  
    if ($this->request->getMethod() === 'POST' && $this->  
>validate([  
        'title' => 'required',
```

```

        'description' => 'required',
        'image' => 'uploaded[image]|ext_in[image,png,jpg,jpeg]'
    ))) {
        // menyimpan file gambar dari input file
        $imageFile = $this->request->getFile('image');
        $imageFileName = $imageFile->getRandomName();
        $imageFile->move(ROOTPATH . 'public/images/',
$imageFileName);

        // mengambil data entity portfolio dari request form
        $portfolio = new \App\Entities\Portfolio();
        $portfolio->title = $this->request-
>getPost('title');
        $portfolio->description = $this->request-
>getPost('description');
        $portfolio->image = "images/" . $imageFileName;

        // menyimpan data entity portfolio ke tabel
portfolios

        $model = model('App\Models\PortfolioModel');
        $model->save($portfolio);

        // redirect ke halaman dengan route `/portfolio`
dengan mengesetflashdata untuk menampilkan pesan sukses
        return redirect()->to('/portfolio')-
>with('success_message', 'Successfully create a new portfolio.');
```

```

    }
    // jika HTTP method adalah "POST" dan data yang dikirim
tidak valid, maka akan mengirimkan pesan error
    elseif ($this->request->getMethod() === 'POST') {
        return redirect()->to('/portfolio/create')-
>withInput()->with('errors', $this->validator->getErrors());
    }

    // handle untuk request dengan method "get"
    echo view('portfolio/create');
}

```

Silakan Anda cermati dan pelajari maksud dari kode di atas.

8. Selanjutnya, pada bagian View Anda juga perlu menambahkan baris kode untuk menampilkan pesan error ketika data yang dikirim dari form tidak valid. Buka file `app/Views/portfolio/create.php` dan tambahkan baris kode berikut sebelum pembuka tag form (`<form>`).

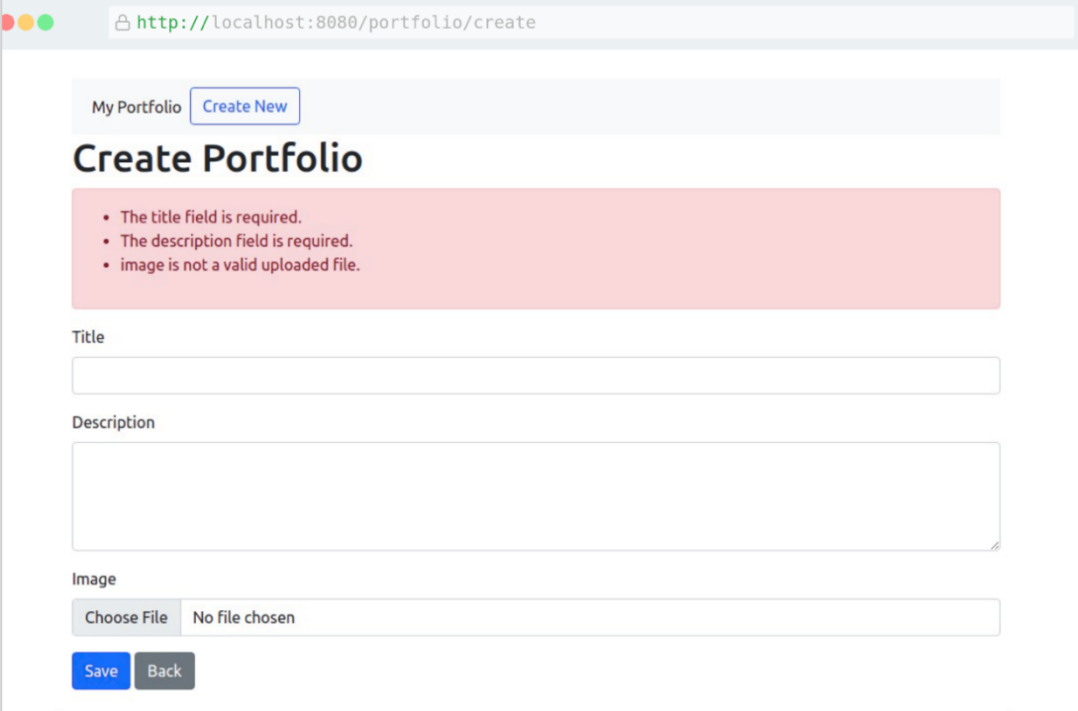
```

<!-- Menampilkan pesan error ketika data tidak valid -->
<!-- cek apakah ada errors -->
<?php if (session()->has('errors')) : ?>
<div class="alert alert-danger" role="alert">
    <ul><!-- dilakukan perulangan pesan error -->
        <?php foreach (session('errors') as $error) : ?>
            <li><?= $error ?></li>
        <?php endforeach ?></ul>
    </div>

```

```
</div>  
<?php endif ?>
```

9. Sekarang silahkan melakukan pengujian. Buka browser dan ketikkan alamat <http://localhost:8080/portfolio/create> dan klik button save tanpa mengisi data apapun pada form. Maka seharusnya akan muncul pesan error pada halaman tersebut seperti gambar di bawah ini.



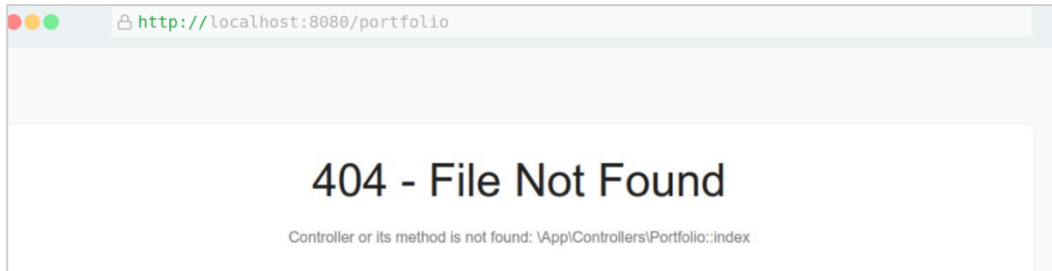
The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/portfolio/create`. The page has a header with 'My Portfolio' and a 'Create New' button. The main heading is 'Create Portfolio'. Below the heading, a red error box contains the following messages:

- The title field is required.
- The description field is required.
- image is not a valid uploaded file.

The form consists of three fields: 'Title' (a single-line text input), 'Description' (a multi-line text area), and 'Image' (a file upload control with a 'Choose File' button and 'No file chosen' text). At the bottom of the form are two buttons: 'Save' (blue) and 'Back' (grey).

10. Lakukan pengujian kedua dengan mengisi data yang valid pada form dan klik button save. Maka seharusnya data akan tersimpan dan halaman akan dialihkan ke alamat <http://localhost:8080/potfolio>. Tentunya halaman yang tampil akan seperti di bawah ini karena Anda belum membuat halaman tersebut. Pada bagian

selanjutnya Anda akan memperbaikinya dengan membuat halaman untuk menampilkan data portfolio.



Apabila pengujian yang Anda lakukan tidak sesuai dengan hasil yang seharusnya, lakukan penelusuran kembali terhadap kode yang Anda buat dan gunakan referensi di internet jika diperlukan.

13.4.6. CRUD: Menampilkan Data Portfolio

Pada bagian sebelumnya Anda sudah berhasil membuat halaman untuk menambahkan data portfolio. Selanjutnya Anda akan membuat halaman untuk menampilkan data portfolio dari tabel portfolios.

1. Tambahkan route pada file app/Config/Routes.php. Tambahkan route baru di atas route yang sebelumnya sudah Anda tambahkan.

```
$routes->get('/portfolio', 'PortfolioController::index');  
$routes->match(['get', 'post'], '/portfolio/create',  
    'PortfolioController::create');
```

2. Selanjutnya ubah method index pada class PortfolioController. Buka file app/Controllers/PortfolioController.php dan ubah pada bagian method index sebagai berikut.

```
public function index()  
{  
    // mengambil semua data portfolio dari tabel portfolios  
    $model = new PortfolioModel();  
    $portfolios = $model->findAll();  
  
    $data = [  
        'portfolios' => $portfolios  
    ];  
  
    // mengirimkan data portfolio ke View  
    echo view('portfolio/index', $data);  
}
```

3. Tambahkan juga kode untuk mengimport class PortfolioModel pada file app/Controllers/PortfolioController.php

```
use App\Models\PortfolioModel;
```

```
class PortfolioController extends BaseController
```

4. Pada method index di atas Anda memanggil View pada path portfolio/index. Oleh karena itu Anda perlu membuat file View pada folder app/Views/portfolio dengan nama index.php dan tuliskan kode awal seperti berikut.

```
<?php echo $this->extend('layouts/main');?>

<?php echo $this->section('content');?>
<!-- menampilkan pesan sukses dari flashdata -->

<!-- menampilkan list portfolio -->

<?php echo $this->endSection(); ?>
```

Pada kode di file tersebut Anda menggunakan View layout dari file app/Views/layouts/main.php dan di bagian section terdapat dua bagian yaitu bagian untuk menampilkan pesan sukses yang tersimpan pada flashdata dan menampilkan data portfolio yang sebelumnya sudah dikirim dari Controller ke View. Untuk bagian pertama, tambahkan baris kode ke bagian tersebut hingga menjadi seperti ini.

```
<!-- menampilkan pesan sukses dari flashdata -->
<?php if(session()->getFlashData('success_message')): ?>
<div class="alert alert-success" role="alert">
    <?php echo session()->getFlashData('success_message'); ?>
</div>
<?php endif;?>
```

Pada kode tersebut Anda mengecek ketika terdapat data pada flashdata dengan key success_message maka akan ditampilkan ke dalam div bootstrap alert. Untuk bagian kedua tambahkan baris kode sebagai berikut.

```
<!-- menampilkan list portfolio -->
<div class="row gx-5 gy-5 mt-2">
    <?php foreach($portfolios as $portfolio):?>
        <div class="col col-6">
            <div class="card">
                title;?>">
                <div class="card-body">
                    <h5 class="card-title"><?php echo $portfolio-
>title;?></h5>
                    <p class="card-text"><?php echo $portfolio-
>description;?></p>
                    <br>
                    <a href="#" class="btn btn-primary">Edit</a>
```

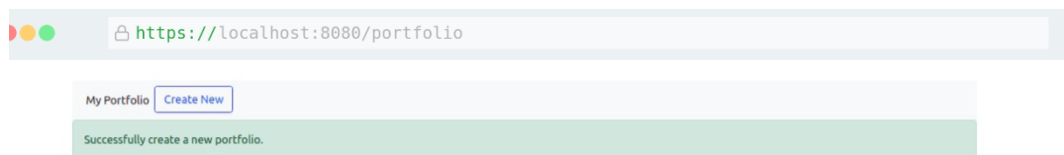
```

                <button onclick="" class="btn btn-
danger">Delete</button>
            </div>
        </div>
    </div>
<?php endforeach;?>
</div>

```

Pada kode tersebut Anda menggunakan function foreach untuk menampilkan data portfolio satu persatu dengan beberapa elemen seperti image, title, description serta dua buah button (Edit dan Delete) yang nantinya akan digunakan untuk mengubah dan menghapus data portfolio.

5. Jalankan kembali aplikasi dengan membuka browser dan ketikkan alamat <http://localhost:8080/portfolio>
6. Jika kita tambahkan lagi data portfolio dari alamat <http://localhost:8080/portfolio/create> maka akan dialihkan ke alamat <http://localhost:8080/portfolio> dengan menampilkan pesan sukses seperti gambar di bawah ini.



7. Anda perlu memperbaiki link pada file `app/Views/partials/navbar.php` seperti berikut.

```

<li class="nav-item">
    <a class="nav-link active" aria-current="page" href="<?php echo
site_url('portfolio');?>">My Portfolio</a>
</li>
<li class="nav-item">
    <a class="btn btn-outline-primary" href="<?php echo
site_url('portfolio/create');?>">Create New</a>
</li>

```

Pada menu MyPortfolio Anda set attribute hrefnya dengan URL helper agar mengarah ke URL <http://localhost:8080/portfolio> dan melakukan hal yang sama untuk menu Create New dengan mengarahkan ke URL <http://localhost:8080/portfolio/create>

13.4.7. CRUD: Mengubah Data Portfolio

1. Pada bagian ini Anda akan membuat halaman untuk mengubah data portfolio. Lakukan modifikasi pada file `app/Views/portfolio/index.php` dengan mengeset

attribut href button Edit dengan URL helper dan mengarahkannya ke URL portfolio/update/:id

```
<a href="<?php echo site_url('portfolio/update/'.$portfolio->id);?>"  
class="btn btn-primary">Edit</a>
```

2. Tambahkan route baru ke file app/Config/Routes.php. Tambahkan di bagian bawah dari route untuk halaman menambah data portfolio.

```
$routes->match(['get', 'post'], '/portfolio/create',  
'PortfolioController::create');  
$routes->match(['get', 'post'], '/portfolio/update/(:num)',  
'PortfolioController::update/$1');
```

3. Selanjutnya buka file `app/Controllers/PortfolioController.php` dan tambahkan method `update` pada class `PortfolioController`

```
public function update($id)
{
    $model = new PortfolioModel();
    //mengambil data portfolio berdasarkan parameter $id
    if (! $portfolio = $model->find($id)) {
        // jika data portfolio tidak ditemukan, maka akan dialihkan
        ke halaman 404
        throw
        \CodeIgniter\Exceptions\PageNotFoundException::forPageNotFound();
    }

    // handle untuk request dengan method "post"

    // handle untuk request dengan method "get"
}
```

Di bagian bawah Anda akan menambahkan dua bagian untuk handle request dengan method `get` maupun `post` karena sebelumnya Anda mengeset route untuk path `/portfolio/update/(:num)` dengan dua HTTP method yaitu `get` dan `post`.

4. Untuk yang pertama, tambahkan baris kode untuk handle request dengan http method `put` pada `app/Controllers/PortfolioController.php`

```
// handle untuk request dengan method "post"
// mengecek apakah request http method = "post" dan apakah data yang
dikirim valid atau tidak
if ($this->request->getMethod() === 'POST' && $this->validate([
    'title' => 'required',
    'description' => 'required',
    'image' => 'ext_in[image,png,jpg,jpeg]'
])) {
    $imageFile = $this->request->getFile('image');
    $imageName = $imageFile->getRandomName();

    if($imageFile->isValid()) {
        //menghapus file gambar sebelumnya
        unlink(ROOTPATH.'public/'.$portfolio->image);

        //menyimpan file gambar baru dari input file
        $imageFile->move(ROOTPATH.'public/images/', $imageName);
        $portfolio->image = "images/".$imageName;
    }

    $portfolio->title = $this->request->getPost('title');
    $portfolio->description = $this->request-
>getPost('description');
    // jika terdapat perubahan, simpan perubahan data portfolio ke
    tabel portfolios
    if ($portfolio->hasChanged()) {
        $model->save($portfolio);
    }
}
```

```

        return redirect()->to('/portfolio')->with('success_message',
'Successfully update portfolio.');
```

```

    }

    // jika HTTP method adalah "POST" dan data yang dikirim
    tidak valid, maka akan mengirimkan pesan error
    elseif ($this->request->getMethod() === 'POST') {
        return redirect()->to('/portfolio/update/'.$portfolio-
>id)->withInput()->with('errors', $this->validator->getErrors());
    }

```

Pelajari dan pahami baris kode di atas dengan baik.

5. Selanjutnya, Anda akan menambahkan baris kode pada bagian kedua untuk menampilkan halaman form untuk mengubah data portfolio. Tambahkan kode berikut ini pada file `app/Controllers/PortfolioController.php`

```

// handle untuk request dengan method "get"
$data = [
    'portfolio' => $portfolio
];

echo view('portfolio/update', $data);

```

6. Sekarang Anda perlu membuat file baru dengan nama `update.php` pada folder `app/Views/portfolio`

```

<?php echo $this->extend('layouts/main') ?>

<?php echo $this->section('content'); ?>
<!-- title -->
<h1>Update Portfolio</h1>

<!-- alert validation errors -->
<!-- cek apakah ada errors -->
<?php if (session()->has('errors')) : ?>
<div class="alert alert-danger" role="alert">
    <ul><!-- dilakukan perulangan pesan error -->
        <?php foreach (session('errors') as $error) : ?>
            <li><?= $error ?></li>
        <?php endforeach ?></ul>
    </div>
<?php endif ?>

<!-- form update -->
<form method="post" action="<?php echo
site_url('portfolio/update/'.$portfolio->id);?>"
enctype="multipart/form-data">
    <?php echo csrf_field() ?>
    <div class="mb-3">
        <label class="form-label">Title</label>
        <input type="text" class="form-control" name="title"
value="<?php echo set_value('title', $portfolio->title);?>">
    </div>

```

```

<div class="mb-3">
    <label class="form-label">Description</label>
    <textarea name="description" class="form-control" rows="4"><?php
echo set_value('description', $portfolio->description);?></textarea>
</div>
<div class="mb-3">
    <label class="form-label">Image</label>
    <input type="file" class="form-control" name="image">
</div>
<div class="mb-3">
    <?php if($portfolio->image):?>
    title;?>">
    <?php endif;?>
</div>

    <button type="submit" class="btn btn-primary">Save</button>
    <a class="btn btn-secondary" href="<?php echo
site_url('portfolio');?>">Back</a>
</form>

<?php echo $this->endSection(); ?>

```

Pahami bagaimana perbedaan baris kode antara method put dengan post.

7. Lakukan pengujian kembali melalui browser Anda.

13.4.8. CRUD: Menghapus Data Portfolio

Berikutnya Anda akan menambahkan action untuk menghapus data portfolio. Pada bagian ini Anda tidak membuat halaman tersendiri namun hanya akan menambahkan route dan method pada class Controller tanpa View.

1. Untuk route silahkan tambahkan route baru di bawah route untuk mengubah data portfolio.

app/Config/Routes.php

```

$routes->match(['get', 'post'], '/portfolio/update/(:num)',
'PortfolioController::update/$1');

$routes->delete('/portfolio/delete/(:num)',
'PortfolioController::destroy/$1');

```

2. Selanjutnya tambahkan method destroy pada class PortfolioController. Buka file app/Controllers/PortfolioController.php lalu tambahkan kode berikut ini.

```

public function destroy($id)
{
    $model = new PortfolioModel();

```

```
//handle untuk request dengan method "delete"
if($this->request->getMethod() === 'DELETE') {
    if (! $portfolio = $model->find($id)) {
        // jika data portfolio tidak ditemukan, maka akan dialihkan
        // ke halaman 404
        throw
        \CodeIgniter\Exceptions\PageNotFoundException::forPageNotFound();
    }
    // hapus data portfolio dari tabel portfolios
    $model->delete($portfolio->id);
}

//handle untuk request dengan method selain "delete"
return redirect()->to('/portfolio');
}
```

3. Selanjutnya Anda perlu menambahkan action pada button Delete di halaman list portfolio. Buka file `app/Views/Portfolio/index.php` dan ubah attribute onclick pada button Delete seperti ini:

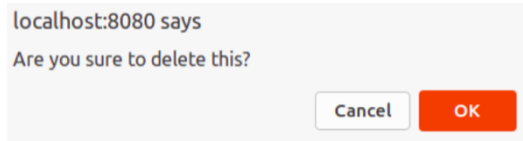
```
<button onclick="return deletePortfolio('<?php echo $portfolio-
>id;?>');" class="btn btn-danger">Delete</button>
```

4. Tambahkan script javascript untuk membuat function tersebut di baris paling bawah di dalam tag `<script>`

```
<!-- script untuk action hapus data portfolio -->
<?php echo $this->section('scripts');?>
<script>
    function deletePortfolio(id) {
        var confirmation = confirm("Are you sure to delete this?");

        if(confirmation == true) {
            fetch("<?php echo site_url('/portfolio/delete');?>/" +
id, {
                method: 'DELETE'
            }).then(data => {
                location.reload();
            });
        }
    }
</script>
<?php echo $this->endSection(); ?>
```

5. Jalankan kembali aplikasi melalui browser Anda. Klik button Delete maka akan muncul dialog box dan ketika Anda pilih OK maka data akan terhapus.



13.5. **Responsi**

Kerjakan sesuai dengan yang dijelaskan pada bagian Kegiatan Praktikum. Simpan tangkapan layar hasil pekerjaan Anda dalam file docx. Simpan ulang file tersebut dalam format pdf dan beri nama dengan format <<nim>>_modul13.pdf, contoh: 192191234_modul13.pdf. Kumpulkan file tersebut sebagai responsi melalui Google Classroom.