Tutorial CRUD (Create, Read, Update & Delete) Codeigniter 4 dengan Bootstrap

Tutorial CRUD (Create, Read, Update & Delete) Codeigniter 4 dengan Bootstrap

Admin

Admin · 14 Oktober 2021

27 min read · 10030 views

Series: Codeigniter 4 Show

CRUD Codeigniter 4 – Halo gaess !👋 Di artikel ini saya akan share tutorial membuat CRUD atau Create, Read, Update dan Delete di codeigniter 4. Artikel ini merupakan lanjutan dari artikel codeigniter 4 yang sebelumnya telah saya bagikan. Disini, saya akan simulasikan membuat CRUD di codeigniter 4 untuk membuat contact management. Jadi, kita bisa melihat data contact, tambah data contact, update data

Oke, mari kita langsung saja ke koding. 👇 🧑‍💻

Persiapan

Karena kita ingin membuat contact management (CRUD), tentunya kita juga butuh model dan table contact. Untuk itu, sebagai langkah persiapan, mari kita siapkan dulu file model dan table contacts.

Step 1: Buat Model

```
php spark make:model Contact
```

Yang pertama, kita butuh model Contact. Untuk membuat model di codeigniter 4, kita bisa membuatnya dengan menjalankan perintah seperti di atas atau membuat file secara manual di dalam direktori app/Models dengan nama file Contact.php.

```
<?php

namespace App\Models;

use CodeIgniter\Model;

 ass Contact extends Model
{
```

kita menekankan bahwa model Contact ini merupakan model untuk table contacts dengan primaryKey id dan mempunyai field-field name, email, phone dan address.

Step 2: Buat Migration

```
php spark make:migration Contact
```

Ok, selanjutnya kita buat file migration untuk table contacts. Untuk membuat file migration di codeigniter 4, bisa dengan menjalankan perintah seperti di atas atau membuat file migration secara manual di dalam direktori app/Database/Migrations.

```
<?php

namespace App\Database\Migrations;

use CodeIgniter\Database\Migration;

class Contact extends Migration
{
    public function up()
    {
        $this->forge->addField([
            'id'          => [
                'type'           => 'INT',
                'constraint'     => 5,
                'unsigned'       => true,
                'auto_increment' => true
            ],
            'name'        => [
                'type'           => 'VARCHAR',
                'constraint'     => '255'
            ],
            'email'       => [
                'type'           => 'VARCHAR',
                'constraint'     => 100,
            ],
            'phone' => [
                'type'           => 'VARCHAR',
                'constraint'     => 100,
            ],
            'address'       => [
                'type'           => 'VARCHAR',
                'constraint'     => 255,
```

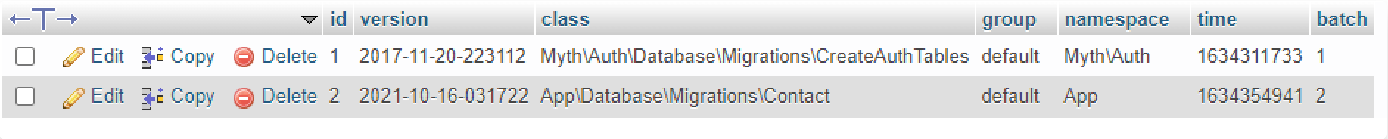
Beyond
with
Codes

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

Selanjutnya buka file contact migration yang baru saja dibuat, lalu sesuaikan kode yang ada menjadi seperti kode di atas. Disini kita akan membuat table contacts dengan field id, name, email, phone dan address.

```
php spark migrate
```

Sudah selesai membuat file migration untuk contact ? Lanjutkan dengan menjalankan perintah **php spark migrate** seperti di atas.



←T→	id	version	class	group	namespace	time	batch
□ Edit Copy Delete	1	2017-11-20-223112	Myth\Auth\Database\Migrations\CreateAuthTables	default	Myth\Auth	1634311733	1
□ Edit Copy Delete	2	2021-10-16-031722	App\Database\Migrations\Contact	default	App	1634354941	2

Jika berhasil melakukan migrate, seharusnya jika kamu membuka table migrations di database, akan terlihat class Contact telah berhasil di migrasi seperti gambar di atas.

Membuat Fitur Create Data

Oke, setelah selesai di langkah-langkah persiapan, mari kita masuk ke langkah-langkah membuat fitur create data dan menampilkan data-data yang telah berhasil dibuat ke tampilan table.

Step 1: Buat Route

```
$routes->get('contact', 'ContactController::index');
$routes->add('contact', 'ContactController::create');
```

Langkah yang pertama, kita perlu membuat atau menambahkan route baru yang nantinya akan dipanggil saat kita ingin melihat data contact dan ingin menambahkan contact. Buka file app/Config/Routes.php, lalu tambahkan kode seperti di atas ke dalam route group.

```
$routes->group('', ['filter' => 'login'], function($routes){
    $routes->get('dashboard', 'Home::dashboard');
    $routes->get('contact', 'ContactController::index');
    $routes->add('contact', 'ContactController::create');
});
```

bisa menggunakan cara seperti di atas yaitu menjalankan perintah **php spark make:controller ContactController** pada terminal.

Kita juga bisa membuat controller di codeigniter 4 secara manual, yaitu dengan membuat file baru dengan nama ContactController.php di dalam direktori app/Controllers.

```
 <?php

namespace App\Controllers;
use App\Models\Contact;

class ContactController extends BaseController
{
    protected $contact;

    function __construct()
    {
        $this->contact = new Contact();
    }

    public function index()
    {
    $data['contacts'] = $this->contact->findAll();

        return view('contacts/index', $data);
    }

    public function create()
    {
        $this->contact->insert([
            'name'  => $this->request->getPost('name'),
            'email'  => $this->request->getPost('email'),
            'phone'  => $this->request->getPost('phone'),
            'address'  => $this->request->getPost('address'),
        ]);

        return redirect('contact')->with('success', 'Data Added Successfully');
    }

}
```

udian buka file ContactController.php yang baru saja dibuat, lalu sesuaikan kode yang ada menjadi seperti kode di atas. Disini kita punya, 3 function yaitu constract, index dan create.

and Beyond with Codes

insert ke tables contacts. Setelah melakukan insert, user akan diarahkan kembali ke halaman contact dan menampilkan pesan success dengan text "Data Added Successfully.

Step 3: Buat Contact View

```
 <?= $this->extend('layouts/admin') ?>
<?php $this->section('styles') ?>
<!-- Custom styles for this page -->
<link href="<?= base_url('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?> " rel="styl
<?= $this->endSection() ?>
<?= $this->section('content') ?>
<div class="container-fluid">
    <!-- Page Heading -->
    <h1 class="h3 mb-2 text-gray-800">Contacts</h1>
    <?php
        if(session()->getFlashData('success')){
        ?>
    <div class="alert alert-success alert-dismissible fade show" role="alert">
        <?= session()->getFlashData('success') ?>
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
        <span aria-hidden="true">&times;</span>
        </button>
    </div>
    <?php
        }
        ?>
    <!-- DataTales Example -->
    <div class="card shadow mb-4">
        <div class="card-header py-3">
            <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#addMo
            Add Contact
            </button>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
                    <thead>
                        <tr>
                            <th>#</th>
                            <th>Name</th>
                            <th>Email</th>
                            <th>Phone</th>
                            <th>Address</th>
                            <th>Action</th>
```

and
Beyond
with
Codes

```
                            <td><?= $contact['email'] ?></td>
                            <td><?= $contact['phone'] ?></td>
                            <td><?= $contact['address'] ?></td>
                            <td>
                                NA
                            </td>
                        </tr>
                        <?php endforeach; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<!-- Add Contact Modal -->
<div class="modal fade" id="addModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidd
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Add Contact</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <form action="<?= base_url('contact') ?>" method="post">
            <?= csrf_field(); ?>
                <div class="modal-body">
                    <div class="form-group">
                        <label for="name">Name</label>
                        <input type="text" name="name" class="form-control" id="name" placeholder
                    </div>
                    <div class="form-group">
                        <label for="email">Email</label>
                        <input type="email" name="email" class="form-control" id="email" placehol
                    </div>
                    <div class="form-group">
                        <label for="phone">Phone</label>
                        <input type="text" name="phone" class="form-control" id="phone" placehold
                    </div>
                    <div class="form-group">
                        <label for="address">Address</label>
                        <input type="text" name="address" class="form-control" id="address" place
                    </div>
                </div>
```

```
</div>
<?= $this->endSection() ?>
<?php $this->section('scripts')?>
<!-- Page level plugins -->
<script src="<?= base_url('assets/vendor/datatables/jquery.dataTables.min.js') ?>"></script>
<script src="<?= base_url('assets/vendor/datatables/dataTables.bootstrap4.min.js') ?>"></script>
<!-- Page level custom scripts -->
<script>
    // Call the dataTables jQuery plugin
    $(document).ready(function() {
      $('#dataTable').DataTable();
    });
</script>
<?php $this->endSection()?>
```

Oke, selanjutnya buatlah folder baru di dalam direktori app/Views dengan nama contacts. Kemudian, masuk ke folder contacts yang baru saja dibuat tersebut dan buatlah file baru dengan nama index.php. Selanjutnya, copy semua kode di atas, lalu paste pada file index.php yang baru saja dibuat tersebut.

```
<?= $this->extend('layouts/admin') ?>
```

Di file view index.php ini, kita menggunakan layout dari file view admin.php.

```
<?php $this->section('styles') ?>
<!-- Custom styles for this page -->
<link href="<?= base_url('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?> " rel="styl
<?= $this->endSection() ?>
```

Kemudian kita tambahkan style untuk datatables yang kita gunakan untuk menampilkan data-data dari table contacts.

```
<?php
    if(session()->getFlashData('success')){
?>
<div class="alert alert-success alert-dismissible fade show" role="alert">
    <?= session()->getFlashData('success') ?>
    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
    </button>
</div>
<?php
```

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#addModal">
Add Contact
</button>
```

Kita juga menambahkan button Add contact dengan data-target #addModal. Jika button ini diklik, maka akan menampilkan pop up modal yang berisikan form untuk tambah data contact.

```
<table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
    <thead>
        <tr>
            <th>#</th>
            <th>Name</th>
            <th>Email</th>
            <th>Phone</th>
            <th>Address</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>
        <?php foreach ($contacts as $key => $contact) : ?>
        <tr>
            <td><?= ++$key ?></td>
            <td><?= $contact['name'] ?></td>
            <td><?= $contact['email'] ?></td>
            <td><?= $contact['phone'] ?></td>
            <td><?= $contact['address'] ?></td>
            <td>
                NA
            </td>
        </tr>
        <?php endforeach; ?>
    </tbody>
</table>
```

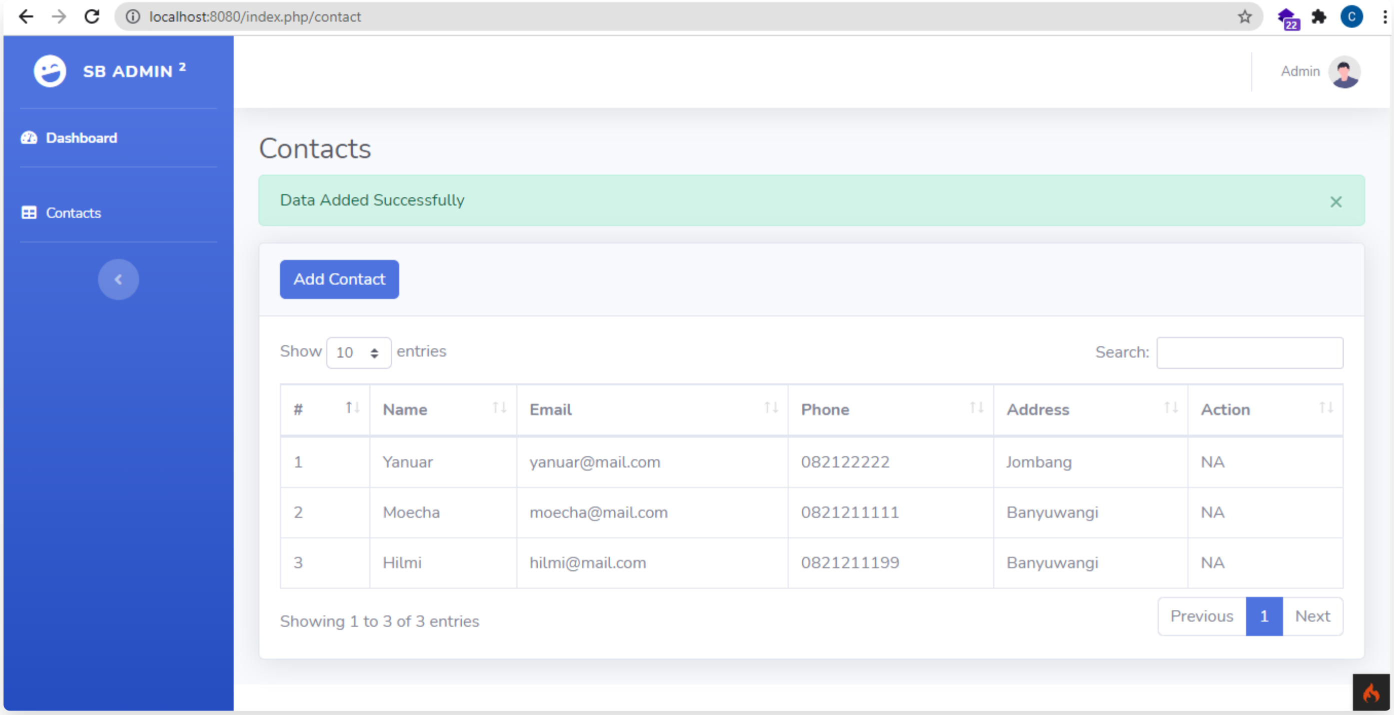
Kode di atas, berfungsi untuk membuat tampilan table dengan menampilkan data dari variable contacts yang telah dibuat pada ContactController.php.

```
<?php $this->section('scripts')?>
<!-- Page level plugins -->
<script src="<?= base_url('assets/vendor/datatables/jquery.dataTables.min.js') ?>"></script>
<script src="<?= base_url('assets/vendor/datatables/dataTables.bootstrap4.min.js') ?>"></script>
```

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

Kemudian, di file index.php ini kita juga menambahkan script untuk datatables yang kita gunakan pada project codeigniter 4 ini.

Step 4: Test Add Data



Contacts

Data Added Successfully

Show 10 entries

Search:

#	Name	Email	Phone	Address	Action
1	Yanuar	yanuar@mail.com	082122222	Jombang	NA
2	Moecha	moecha@mail.com	0821211111	Banyuwangi	NA
3	Hilmi	hilmi@mail.com	0821211199	Banyuwangi	NA

Showing 1 to 3 of 3 entries

Oke, sampai disini, mari kita coba terlebih dahulu fitur create data yang telah kita buat. Jalankan project codeigniter 4 ini dengan menjalankan perintah php spark serve, lalu buka pada browser. Coba login atau register akun terlebih dahulu. Jika berhasil login, coba masuk ke menu contact dan tambahkan data contact dengan klik button Add Contact. Jika berhasil menambahkan data, maka kita akan diarahkan kembali ke halaman contact dan tampilan pesan "Data Added Successfully".

Membuat Fitur Update Data

Fitur selanjutnya yang akan kita buat pada project latihan CRUD di codeigniter 4 ini yaitu fitur update data.

Step 1: Tambah Route

```
$routes->add('contact/edit/(:segment)', 'ContactController::edit/$1');
```

Oke, pertama kita perlu membuat atau menambahkan route baru untuk edit atau update data. Buka file 'Config/Routes.php, lalu tambahkan route dengan kode seperti di atas.

Sehingga, route di dalam route group sekarang menjadi seperti kode di atas.

Step 2: Update Controller

```
    public function edit($id)
        {

            $this->contact->update($id, [
                    'name' => $this->request->getPost('name'),
                    'email' => $this->request->getPost('email'),
                    'phone' => $this->request->getPost('phone'),
                    'address' => $this->request->getPost('address'),
                ]);

                return redirect('contact')->with('success', 'Data Updated Successfully');
        }
```

Kemudian selanjutnya kita update file ContactController.php dengan menambahkan function edit seperti kode di atas. Fungsi dari function di atas, untuk menangkap request dan melakukan update data berdasarkan id data yang diupdate.

Step 3: Setup View

```
    <td>
          NA
    </td>
```

OK, kemudian kita update file view index.php. Cari kode seperti di atas, lalu ubah menjadi seperti kode di bawah ini.

```
    <td>
        <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#editModal-<?=
        Edit
        </button>
    </td>
```

Disini, kita menambahkan button edit dengan data target editMode-{id}. Artinya, jika button ini diklik, maka akan menampilkan pop up atau modal dengan data contact berdasarkan id data yang dipilih.

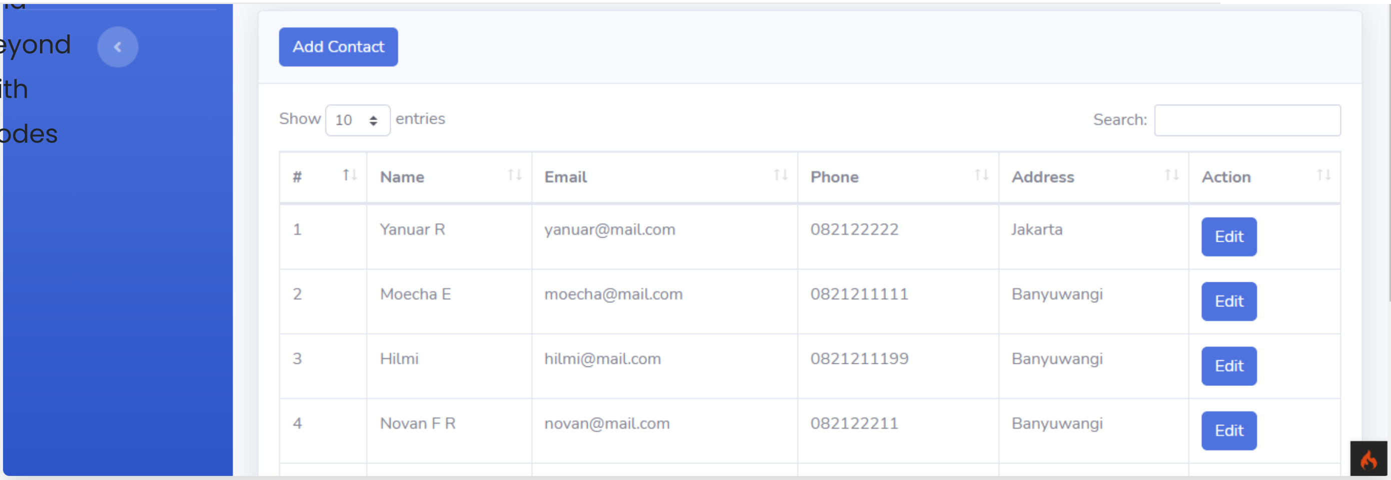
```
    -- Edit Contact Modal -->
    iv class="modal fade" id="editModal-<?= $contact['id'] ?>" tabindex="-1" aria-labelledby="examp
        <div class="modal-dialog">
```

and
Beyond
with
Codes

```
            <form action="<?= base_url('contact/edit/'.$contact['id']) ?>" method="post">
                <?= csrf_field(); ?>
                <div class="modal-body">
                    <div class="form-group">
                        <label for="name">Name</label>
                        <input type="text" name="name" class="form-control" id="name" value="<?= 
                    </div>
                    <div class="form-group">
                        <label for="email">Email</label>
                        <input type="email" name="email" class="form-control" id="email" value="<
                    </div>
                    <div class="form-group">
                        <label for="phone">Phone</label>
                        <input type="text" name="phone" class="form-control" id="phone" value="<?
                    </div>
                    <div class="form-group">
                        <label for="address">Address</label>
                        <input type="text" name="address" class="form-control" id="address" value
                    </div>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</b
                    <button type="submit" class="btn btn-primary">Save</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

Kemudian, tambahkan kode edit modal seperti di atas dan tempatkan di atas kode `<?php endforeach; ?>`.

Step 4: Test Update Data



#	Name	Email	Phone	Address	Action
1	Yanuar R	yanuar@mail.com	082122222	Jakarta	Edit
2	Moecha E	moecha@mail.com	0821211111	Banyuwangi	Edit
3	Hilmi	hilmi@mail.com	0821211199	Banyuwangi	Edit
4	Novan F R	novan@mail.com	082122211	Banyuwangi	Edit

Oke, sekarang waktunya kita coba fitur edit yang telah kita buat di project CRUD codeigniter 4 ini. Silahkan refresh halaman, lalu klik button edit. Nanti akan tampil modal yang berisikan data berdasarkan dari id data yang dipilih. Cobalah untuk mengubah data, lalu klik save, maka data akan diubah sesuai dengan request dan akan menampilkan flash message dengan pesan "Data Updated Successfully".

Membuat Fitur Delete Data

Delete data merupakan fitur terakhir yang akan kita buat pada latihan project CRUD codeigniter 4 ini. Dengan fitur ini nanti, kita bisa melakukan penghapusan data contact berdasarkan id data yang dipilih.

Step 1: Tambah Route

```
$routes->get('contact/delete/(:segment)', 'ContactController::delete/$1');
```

Oke, pada langkah yang pertama kita perlu membuat atau menambahkan route baru untuk delete data. Buka file routes.php, lalu tambahkan kode route seperti di atas di dalam route group.

```
$routes->group('', ['filter' => 'login'], function($routes){
    $routes->get('dashboard', 'Home::dashboard');
    $routes->get('contact', 'ContactController::index');
    $routes->add('contact', 'ContactController::create');
    $routes->add('contact/edit/(:segment)', 'ContactController::edit/$1');
    $routes->get('contact/delete/(:segment)', 'ContactController::delete/$1');
});
```

Sehingga route group sekarang akan menjadi seperti kode di atas.

Step 2: Update Controller

```
public function delete($id)
{
```

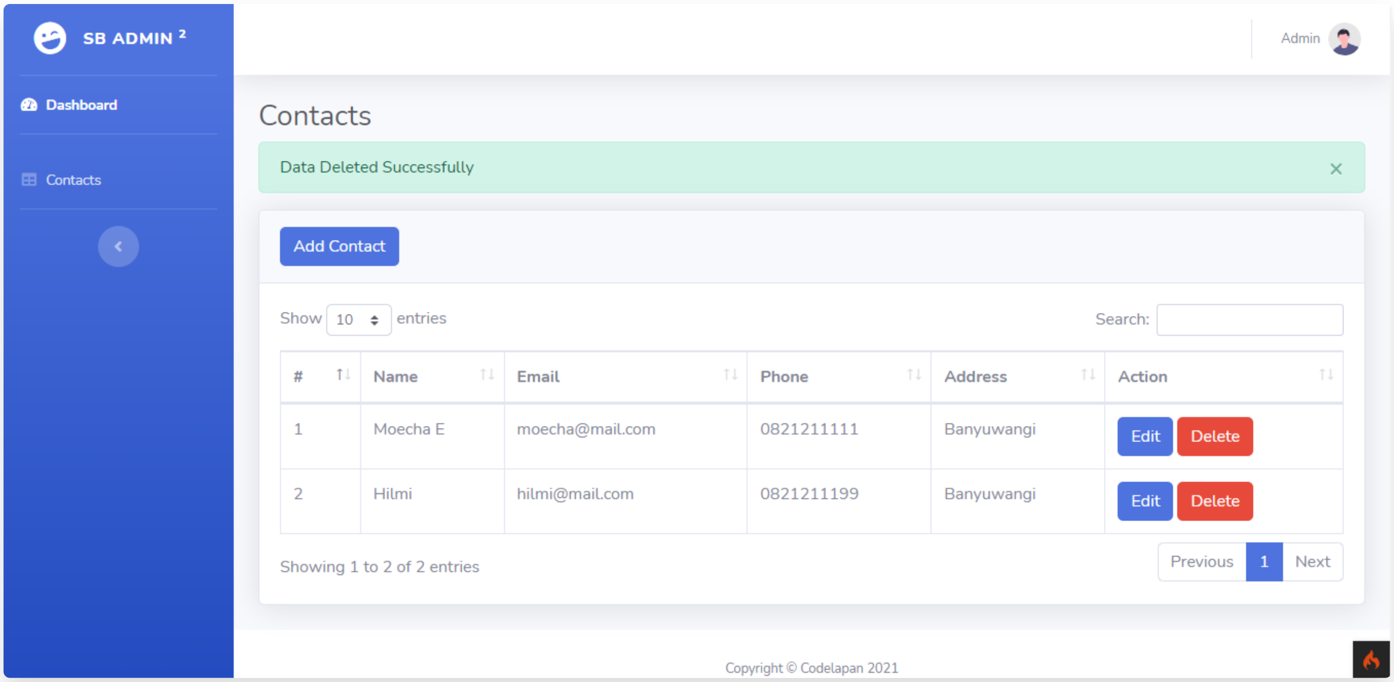
Dengan kode ini, akan melakukan proses delete data contact berdasarkan id data yang dipilih. Kemudian setelah berhasil menghapus data, maka akan menampilkan flash message dengat text pesan "Data Deleted Successfully".

Step 3: Setup View

```
<a href="<?= base_url('contact/delete/'.$contact['id']) ?>" class="btn btn-danger" onclick="retur
```

Selanjutnya, kita tambahkan button delete dengan kode seperti di atas. Letakkan kode di atas, di bawah button edit. Saat button Delete diklik, maka akan menampilkan pesan konfirmasi, jika diklik OK, maka data dengan id terpilih akan dihapus dari table contacts.

Step 4: Test Delete Data



Contacts

Data Deleted Successfully ×

Add Contact

Show 10 entries   Search:

#	Name	Email	Phone	Address	Action
1	Moecha E	moecha@mail.com	0821211111	Banyuwangi	Edit Delete
2	Hilmi	hilmi@mail.com	0821211199	Banyuwangi	Edit Delete

Showing 1 to 2 of 2 entries   Previous 1 Next

Copyright © Codelapan 2021

Sekarang, waktunya kita coba fitur delete data pada project latihan CRUD codeigniter 4 ini. Refresh halaman, kemudian coba klik button Delete. Setelah data berhasil dihapus, maka akan diarahkan kembali ke halaman contact dengan flash message "Data deleted Successfully".

Review

Dan berikut ini, kode-kode yang telah kita buat pada project latihan CRUD codeigniter 4.

Route.php

```
<?php
```

```
// can override as needed.
if (file_exists(SYSTEMPATH . 'Config/Routes.php')) {
    require SYSTEMPATH . 'Config/Routes.php';
}

/*
 * --------------------------------------------------------------------
 * Router Setup
 * --------------------------------------------------------------------
 */
$routes->setDefaultNamespace('App\Controllers');
$routes->setDefaultController('Home');
$routes->setDefaultMethod('index');
$routes->setTranslateURIDashes(false);
$routes->set404Override();
$routes->setAutoRoute(true);

/*
 * --------------------------------------------------------------------
 * Route Definitions
 * --------------------------------------------------------------------
 */

// We get a performance increase by specifying the default
// route since we don't have to scan directories.
$routes->get('/', 'Home::index');
$routes->group('', ['filter' => 'login'], function($routes){
    $routes->get('dashboard', 'Home::dashboard');
    $routes->get('contact', 'ContactController::index');
    $routes->add('contact', 'ContactController::create');
    $routes->add('contact/edit/(:segment)', 'ContactController::edit/$1');
    $routes->get('contact/delete/(:segment)', 'ContactController::delete/$1');
});

/*
 * --------------------------------------------------------------------
 * Additional Routing
 * --------------------------------------------------------------------
 *
 * There will often be times that you need additional routing and you
 * need it to be able to override any defaults in this file. Environment
 * based routes is one such time. require() additional route files here
 * to make that happen.
 *
```

and
Beyond
with
Codes

ContactController.php

```php
<?php

namespace App\Controllers;
use App\Models\Contact;

class ContactController extends BaseController
{
    protected $contact;

    function __construct()
    {
        $this->contact = new Contact();
    }

    public function index()
    {

        $data['contacts'] = $this->contact->findAll();

        return view('contacts/index', $data);
    }

    public function create()
    {
        $this->contact->insert([
            'name' => $this->request->getPost('name'),
            'email' => $this->request->getPost('email'),
            'phone' => $this->request->getPost('phone'),
            'address' => $this->request->getPost('address'),
        ]);

        return redirect('contact')->with('success', 'Data Added Successfully');

    }

    public function edit($id)
    {

        $this->contact->update($id, [
                'name' => $this->request->getPost('name'),
                'email' => $this->request->getPost('email'),
```

Beyond
with
Codes

```
    public function delete($id){
        $this->contact->delete($id);

        return redirect('contact')->with('success', 'Data Deleted Successfully');
    }

}
```

contact/index.php

```
 <?= $this->extend('layouts/admin') ?>

<?php $this->section('styles') ?>
<!-- Custom styles for this page -->
<link href="<?= base_url('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?> " rel="styl
<?= $this->endSection() ?>

<?= $this->section('content') ?>
<div class="container-fluid">
    <!-- Page Heading -->
    <h1 class="h3 mb-2 text-gray-800">Contacts</h1>
    <?php
        if(session()->getFlashData('success')){
    ?>
    <div class="alert alert-success alert-dismissible fade show" role="alert">
        <?= session()->getFlashData('success') ?>
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
        <span aria-hidden="true">&times;</span>
        </button>
    </div>
    <?php
        }
    ?>
    <!-- DataTales Example -->
    <div class="card shadow mb-4">
        <div class="card-header py-3">
            <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#addMo
            Add Contact
            </button>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
                            <th>Action</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($contacts as $key => $contact) : ?>
                        <tr>
                            <td><?= ++$key ?></td>
                            <td><?= $contact['name'] ?></td>
                            <td><?= $contact['email'] ?></td>
                            <td><?= $contact['phone'] ?></td>
                            <td><?= $contact['address'] ?></td>
                            <td>
                                <button type="button" class="btn btn-primary" data-toggle="modal"
                                    Edit
                                </button>
                                <a href="<?= base_url('contact/delete/'.$contact['id']) ?>" class
                            </td>
                        </tr>
                        <!-- Edit Contact Modal -->
                        <div class="modal fade" id="editModal-<?= $contact['id'] ?>" tabindex="-1
                            <div class="modal-dialog">
                                <div class="modal-content">
                                    <div class="modal-header">
                                        <h5 class="modal-title" id="exampleModalLabel">Edit Conta
                                        <button type="button" class="close" data-dismiss="modal"
                                        <span aria-hidden="true">&times;</span>
                                        </button>
                                    </div>
                                    <form action="<?= base_url('contact/edit/'.$contact['id']) ?>
                                    <?= csrf_field(); ?>
                                        <div class="modal-body">
                                            <div class="form-group">
                                                <label for="name">Name</label>
                                                <input type="text" name="name" class="form-contro
                                            </div>
                                            <div class="form-group">
                                                <label for="email">Email</label>
                                                <input type="email" name="email" class="form-cont
                                            </div>
                                            <div class="form-group">
                                                <label for="phone">Phone</label>
                                                <input type="text" name="phone" class="form-contr
                                            </div>
                                            <div class="form-group">
```

```
                                        </div>
                                    </form>
                                </div>
                            </div>
                        </div>
                        <?php endforeach; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<!-- Add Contact Modal -->
<div class="modal fade" id="addModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidd
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Add Contact</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <form action="<?= base_url('contact') ?>" method="post">
                <?= csrf_field(); ?>
                <div class="modal-body">
                    <div class="form-group">
                        <label for="name">Name</label>
                        <input type="text" name="name" class="form-control" id="name" placeholder
                    </div>
                    <div class="form-group">
                        <label for="email">Email</label>
                        <input type="email" name="email" class="form-control" id="email" placehol
                    </div>
                    <div class="form-group">
                        <label for="phone">Phone</label>
                        <input type="text" name="phone" class="form-control" id="phone" placehold
                    </div>
                    <div class="form-group">
                        <label for="address">Address</label>
                        <input type="text" name="address" class="form-control" id="address" place
                    </div>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</b
```

Beyond
with
Codes

```
<?php $this->section('scripts')?>
<!-- Page level plugins -->
<script src="<?= base_url('assets/vendor/datatables/jquery.dataTables.min.js') ?>"></script>
<script src="<?= base_url('assets/vendor/datatables/dataTables.bootstrap4.min.js') ?>"></script>
<!-- Page level custom scripts -->
<script>
    // Call the dataTables jQuery plugin
    $(document).ready(function() {
      $('#dataTable').DataTable();
    });
</script>
<?php $this->endSection()?>
```

Kesimpulan

Sampai disini kita telah berhasil membuat CRUD (Create, Read, Update dan Delete) sederhana di codeigniter 4 dengan menggunakan modal untuk create dan update data. Di artikel ini, kita juga banyak membahas tentang penggunaan method extend, section, flash data atau flash message dan perulangan di codeigniter 4 dengan menggunakan foreach.

Sekian artikel tentang membuat CRUD sederhana di codeigniter 4 kali ini, di artikel seri codeigniter 4 berikutnya mungkin saya akan share bagaimana membuat validasi di codeigniter 4. Selamat mencoba, semoga artikel ini bisa bermanfaat, dan sampai jumpa di artikel berikutnya. 👋 🚀

Bootstrap Codeigniter

https://codelapan.com/post/crud-codeigniter-4

←| Artikel Sebelumnya

and
Beyond
with
Codes

Tinggalkan Komentar

Loading Comments

Artikel Terkait

Bookmark

Windows

Linux

Mac

and
Beyond
with
Codes

Favorit

See All

Cara Mengatasi Error XAMPP: MySQL shutdown unexpectedly

23 Oktober 2021 22111 views

Membuat REST API CRUD di Laravel 8 dengan Sanctum

18 September 2021 18747 views

Laravel 8: REST API Authentication dengan Sanctum

17 September 2021 16128 views

Contoh Cara Menggunakan Sweet Alert di Laravel 8

and
Beyond
with
Codes