|  |  |
| --- | --- |
| Materi | Nilai |
| Restoran-ci4 20-34 #19 | 93 |

**Saya Sudah Belajar dan Mengerti dan Saya BISA**

1. Menambahkan kolom ke halaman admin page di kategori yaitu pelanggan, order, orderdetail, dan user.
2. Mengedit halaman kategori pada file select.php yaitu menambahkan 1 kolom yg diisi 2 baris dan menambah datu kolom lagi untuk membungkus isi select kategori dari tabel sampai bawah

<div class="row">

    <div class="col-3">

        <a class="btn btn-primary mb-2" href="<?= base\_url('/admin/kategori/create') ?>" role="button">Tambah Data</a>

    </div>

    <div class="col-6">

        <h2><?php echo $judul; ?></h2>

    </div>

</div>

1. Mengedit tombol delete dan update menggunakann icon yg disediakan pada bootstrap yaitu gambar trash dan pencil yg bertipe file .svg lalu memasangkan pada update dan delete

<img src="<?= base\_url('/icon/trash.svg') ?>">

<img src="<?= base\_url('/icon/pencil.svg') ?>">

1. Mengedit front end form insert dan form update menambahkan form-group dan form control serta label pada masing masing form
2. Mengedit controller update agar dapat dibaca oleh database dan agar menjalankan perintahnya lebih akurat.

if ($model->save($\_POST) === false) {

            $error = $model->errors();

$id = $\_POST['idkategori'];

            session()->setFlashdata('info', $error['kategori']);

            return redirect()->to(base\_url("/admin/kategori/find/$id"));

        } else {

            return redirect()->to(base\_url("/admin/kategori"));

        }

Artinya yaitu ($model->save($\_POST) === false) yaitu untuk menyimpan data dari form yang telah kita ketik dan jika tipe data dan value nya salah maka akan menjalankan $error dan menjalankan fungsi setflashdata dan menampilkan informasi error pada halaman yang di redirect() yaitu /admin/find/$id. Jika benar maka akan diambil datanya melalui input submit dan akan otomatis kembal ke halaman /admin/kategori

1. Lalu menambahkan

<?php

echo session()->getFlashdata('info');

?>

Agar jika saat update dan kita tidak mengiisi apa apa atau isi dari update yg belum kita rubah jika kita submit maka akan keluar informasi yg ada dalam setflashdata().

1. Mengedit nama dari pagging group menjadi page.
2. Mengedit paging agar tiap halaman kategori nomornya bisa urut dan akurat

<?php

if (isset($\_GET['page\_page'])) {

    $page = $\_GET['page\_page'];

    $jumlah = 2;

    $no = ($jumlah \* $page) - $jumlah + 1;

} else {

    $no = 1;

}

?>

Artinya jika page nya di isset atau ditekan akan mengambil url page\_page lau akan menjalankan $no yg isinya $jumlah yg isinya jumlah isi kategori kita tiap page kita dikali oleh $page yg isinya page yg kita klik adalah page ke berapa. Lalu dikurangi $jumlah dan +1 agar awalannya tidak angka 0 tapi angka satu. Lalu jika page\_page tidak ada atau belum di isset maka akan menampilkan else yg berisi $no=1; yg diawali angka 1 pada halaman kategori yg page nya belumm di isset

1. Mengedit bagian info jika error menggunakan bootstrap

if (!empty(session()->getFlashdata('info'))) {

    echo '<div  class="col">

    <div class="alert alert-danger col-8" role="alert">';

    echo session()->getFlashdata('info');

    echo "</div>";

}

?>

Artinya jika halaman not empty atau tidak kosomg maka akan diisi dengan session()->getflashdata(‘info’). Maka akan menjalankan class bootstrap lalu text nya diisi dengan session()->getflashdata(‘info’) dan ditutup

1. Membuat controller menu yg baru dan memenghubungkannya dengan name space

namespace App\Controllers\Admin;

use App\Controllers\BaseController;

1. Membuat file select sebagai view dari controller menu lalu diisi file yg hampir sama dengan select dari kategori. Tetapi di edit menjadi select menu
2. Menghubungkan menu ke dalam database dgn cara membuat file baru dalam folder model yaitu Menu\_M.php. yang diisi dengan class Menu\_M protected $table=’tblmenu’; yang menghubungkan menu ke database tblmenu

<?php

namespace App\Models;

use CodeIgniter\Model;

class Menu\_M extends Model

{

    protected $table = 'tblmenu';

}

1. Menambahkan kolom yang diperlukan ke dalam menu dan juga menghubungkannya dari database agar dapat membaca dri database

<td><?= $value['menu'] ?></td>

                    <td><?=  $value['gambar']  ?></td>

                    <td><?= number\_format($value['harga'])  ?></td>

1. Menambahkan foto pada kolom foto dan dihubungkan ke database kolom gambar

 <td><img style="width:70px;" src="<?= base\_url('upload/' . $value['gambar'] . '') ?>"></td>

1. Menambahkan view cell pada select menu agar dapat menampilkan option kategori

<div class="row">

    <div class="col-6">

        <form action="" method="get">

            <?= view\_cell('\App\Controllers\Admin\Menu::option') ?>

        </form>

    </div>

</div>

Artinya yaitu membuat form action method get yang diisi dari view cell yang ada pada controller menu

1. Membuat method option dalam controller menu agar saat dipanggil oleh view cell dalam select menu dapat diakses. Didalam option ini diisi dengan dihhubungkannya dulu pada database kategori lalu mengisi data yang berisi kategori lalu mereturn view dari html template/optionn, $data

public function option()

    {

        $model = new Kategori\_M;

        $kategori = $model->findAll();

        $data = [

            'kategori' => $kategori

        ];

        return view('template/option', $data);

    }

onchange="this.form.submit()

digunakan untuk mengganti halaman jika form options di submit atau di pilih

1. Membua method read pada controller menu agar dapat diakses oleh onchange yang ada pada option jika sudah di submit

public function read()

    {

        if (isset($\_GET['idkategori'])) {

            $id = $\_GET['idkategori'];

            $model = new Menu\_M;

            $menu = $model->where('idkategori', $id)->findAll();

            $data = [

                'judul' => 'DATA PENCARIAN MENU',

                'menu' => $menu,

                'pager' => $model->pager

            ];

            return view("Menu/cari", $data);

        }

    }

Artinya jika option dipilih dan ditekan akan mengambil idkategori dari url, lalu membuat $menu yg berisi model yg memanggil idkategori dan $id yg dipilih lalu mengggunakan function findALL(). Lalu membuat $data yg berisi judul menu dan pager untuk pagination dan me return view Menu/cari untuk ditampilkan ke halaman read.

1. Membuat file cari yang hampirsama pada select karena gunanya hanya untuk menampilkan select atau menu dari option kategori yang dipilih dan sudah di atur oleh method read

<?= $pager->makeLinks(1, $tampil, $total, 'bootstrap') ?>

Digunakan untuk mengatur penempatan paging yang ditaruh pada file cari. Yang bagian akhir yaitu adalah nama paging kita. Yang awal yaitu untuk berapa jumlah page yg mau kita tambahkan

1. Mengedit page agar nomernya tiap halaman page tetap urut

$tampil = 3;

            $mulai = 0;

if (isset($\_GET['page'])) {

                $page = $\_GET['page'];

                $mulai = ($tampil \* $page) - $tampil;

            }

Artinya yaitu jika page nya di klik maka akan mengambil nilai ‘page’ dari url lalu di $mulai yg artinya (3 dikali jumlah angka page yg di klik) lalu kurangi 3 yaitu page yg ingin kita tampilkan

Lalu setelah itu akan dijalankan

$menu = $model->where('idkategori', $id)->findAll($tampil, $mulai);

1. Menambahkan fuungsi update pada controller menu lalu ditaruh pada select menu

public function delete($id = null)

    {

        $model = new Menu\_M;

        $model->delete($id);

        return redirect()->to(base\_url("/admin/menu"));

    }

Artinya delete akan mengambil $id dari model yaitu dari pilihan yg ingi kita delete lalu jika sudah dihapus ia akan mengembalikan halaman ke /admin/menu agar ke refresh otomatis. Sebelum menambahkan method delete pada controller menu, keita harus menambahkan dalam model menu yaitu

 protected $primaryKey = 'idmenu';

itu adalah property dari ci4 nya untuk menghapus data atau mengupdate data

1. Menambahkan function create untuk membuat fungsi insert pada controller menu

public function create()

    {

        $model = new Kategori\_M;

        $kategori = $model->findAll();

        $data = [

            'kategori' => $kategori

        ];

        echo view("menu/insert", $data);

    }

1. Membuat file insert pada view menu yg copasan dari kategori tapi diedit sesuai halaman menu.
2. Membat input type yg bertipe file untuk mengupload gamber

 <div class="form-group">

            <label for="harga">Gambar : </label>

            <input type="file" name="gambar" required >

            <br>

        </div>

1. Menambahkan

protected $allowedFields = ['idkategori', 'menu', 'gambar', 'harga'];

untuk mengisi tabel idkategori menu gambar dan harga ke data base

1. Membuat method insert untuk menambahkan fungsi tambah data yg ada pada select menu tambah data dan pada file insert

public function insert()

    {

        $request = \Config\Services::request();

        $File = $request->getFile('gambar');

        $name = $File->getName();

        $data = [

            'idkategori' => $request->getPost('idkategori'),

            'menu' => $request->getPost('menu'),

            'gambar' => $name,

            'harga' => $request->getPost('harga')

        ];

        $model = new Menu\_M();

        $model->insert($data);

        return redirect()->to(base\_url("/admin/menu"));

    }

Membuat $request agar permintaaan yg diinginkan bisa dijalankan. $file digunakan untuk mengambikl file dari directory untu mengambil gambar yg kta pilih. $name untuk mengambil file name yg kita isi . membuat $data agar diketahi di dalam tabel ada kolom apa saja.

Memnbuat $ model agar terhubung pada data base dan tabelnya dicocokkan dari $data

Lalu menambahkan =$data->insert($data) untuk melakukan insert data sesuai pilihan yang di isi.

Setelah itu akan ke redirect ke admin/menu agar ke refresh otomatis.

1. Membuat file update pada folder menu yg ada dalam view
2. Membuat method yg namanya update untuk menjalnkan fungsi update

public function update()

    {

        $id = $this->request->getPost('idmenu');

        $File = $this->request->getFile('gambar');

        $name = $File->getName();

        if (empty($name)) {

            $name = $this->request->getPost('gambar');

        } else {

            $File->move('./upload');

        }

        $data = [

            'idkategori' => $this->request->getPost('idkategori'),

            'menu' => $this->request->getPost('menu'),

            'gambar' => $name,

            'harga' => $this->request->getPost('harga')

        ];

        $model = new Menu\_M();

        $model->update($id, $data);

        return redirect()->to(base\_url("/admin/menu"));

    }

Artinya yaitu jika tidak ada nama dari folder yg kita ambil jadi file akan mengambil gambar yang awal, tetapi jika ada fle baru yg dimasukan lalu file nya akan ditimpa dengan file baru jadi bisa di update bisa tidak. Lalu membuat $data untuk mengirim data yg dibuat. Lalu membuat $model agar terhubung ke model Menu\_M dan memberi funsi update dan mengirim $id dan $data lalu mengirim redirect ke admin/menu

1. Mengatur error pada update data.

Cth :

Mengatur pesan apabila terjadi error pada model Menu\_M :

protected $validationRules = [

'menu' => 'alpha\_numeric\_space|min\_length[3]|is\_unique[tblmenu.menu]',

'harga' => 'numeric'

];

protected $validationMessages = [

'menu' =>[

'alpha\_numeric\_space' => 'Mohon maaf, di dalam pengisian menu, tidak boleh menggunakan simbol,

harus diisi dengan huruf, angka, dan spasi.',

'min\_length' => 'Isi minimal dari data yang dimasukkan adalah 3 huruf',

'is\_unique' => 'Nama menu sudah terdaftar, silahkan menggunakan nama yang lain'

],

'harga' =>[

'numeric' => 'Mohon maaf, di dalam pengisian harga, tidak boleh menggunakan simbol,

harus diisi dengan angka.'

]

];

Artinya, di dalam $validationRules yang mengatur 2 kolom, yaitu pada kolom menu, rulesnya adalah, alpha\_numeric\_space artinya, menu tidak boleh menggunakan simbol, hanya boleh menggunakan alpha / huruf, numeric / nomor, space /spasi. Lalu min\_length['3'], artinya isi minimal adalah 3. is\_unique artinya, unik dan tidak boleh sama, yang diatur adalah pada tblmenu kolom menu. Dan mengatur kolom harga dengan peraturan numeric, yaitu kolom harus diisi dengan angka, apabila diisi selain angka, maka akan terjadi error.

Di dalam $validationMessages berisi pesan – pesan yang ditampilkan apabila update data yang dimasukkan tidak sesuai dengan peraturan, $validastionMessages mengatur kolom menu dan kolom harga, dimana pesan di dalam kolom menu adalah, apabila data yang dimasukkan menggunakan symbol, maka pesan yang ditampilkan adalah pesan pada key alpha\_numeric\_space, lalu apabila data yang dimasukkan berisi kurang dari 3, maka akan menampiilkan pesan error pada min\_length, dan apabila data yang dimasukkan sudah terdaftar, maka akan menampilkan pesan error yang ada pada is\_unique. Dan di dalam kolom harga, apabila data yang dimasukkan tidak berupa angka, maka akan menampilkan pesan error pada numeric, yaitu data yang dimasukkan harus berupa angka.

1. Mengatur error di dalam function update pada Menu :

if ($model -> update($id, $data)===false) {

$error = $model->errors();

session() -> setFlashdata('info', $error);

return redirect()->to(base\_url("/Admin/Menu/find/$id"));

}else {

return redirect()->to(base\_url("/Admin/Menu"));

}

Artinya, terdapat pengujian yaitu apabila $model perintah update data pada $id dan update pada $data yang dilakukan salah / tidak mengikuti peraturan, maka akan menampilkan $error yang berisi $model menjalankan perintah errors(), lalu di dalam session() berisi perintah untuk mengatur pesan yang ditampilkan ketika melakukan proses dengan key info dan menjalankan $error. Tetapi apabila update yang dijalankan mengikuti peraturan, maka akan mengarahkan halaman ke pada base\_url() dengan alamat localhost/restoran-ci4 lalu disambungkan dengan /Admin/Menu.

1. Mengatur pemanggilan dari error pada file update.php di dalam folder views :

if (!empty(session() -> getFlashdata('info'))) {

echo '<div class="alert alert-danger" role="alert">';

$error = session() -> getFlashdata('info');

foreach ($error as $key => $value) {

echo $key." : " .$value;

echo "<br>";

}

echo '</div>';

}

Artinya, apabila session() getFlashdata yang menjalankan key info tidak kosong, karena data pada key info yang akan diambil sudah diatur, jadi ketika mau mengambil data key info, datanya sudah ada, jadi tidak kosong, maka akang menjalankan tampilan pesan yang mempunyai background merah, lalu menjalankan $error dengan session() yang berisi perintah untuk mengambil data / getFlashdata yang menjalankan key info, karena kolom yang diatur apabila terjadi error lebih dari satu, maka untuk menampilkan pesan error menggunakan foreach dengan $error sebagai key dan menampilkan $value.

1. Membuat file Pelanggan.php di dalam folder admin lalu di dalam function index berisi $model yang memanggil Pelanggan\_M yang di dalam nya terdapat $table tblpelanggan, lalu $data berisi data – data yang akan dikirimkan ke view, dan menjalankan view pelanggan/select. Di dalam view pelanggan/select terdapat pengaturan tabel yang akan ditampilkan yang datanya diambil dari tblpelanggan.
2. Melakukan pengambilan data apabila akun aktif, maka akan menampilkan tulisan aktif dan apabila akun tidak aktif, maka akan menampilkan tulisan non-aktif.

Cth :

<?php

if ($value['aktif']==1) {

$aktif = "AKTIF";

}else {

$aktif = "NON-AKTIF";

}

?>

<td>

<a href="<?= base\_url()?>/Admin/Pelanggan/update/<?= $value['idpelanggan']?> /<?= $value['aktif']?>"><?= $aktif?></a>

</td>

Artinya, terdapat pengujian yaitu, apabila $value[‘aktif’] nilai nya sama dengan 1, maka akan menjalankan $aktif = AKTIF, tetapi apabbila $value[‘aktif’] bernilai 0 atau tidak bernilai 1, maka akan menjalankan $aktif = NON-AKTIF. Terdapat td berisi hyperlink yang memanggil base\_url() dengan alamat localhpst/restoran-ci4 lalu disambungkan dengan /Admin/Pelanggan/update/dengan mengambil value dari $value[‘idpelanggan’]/lalu disambungkan dengan mengambil nilai status aktif. , jadi artinya apabila link di klik, maka akan mengarahkan ke tampilan method update di dalam file pelanggan pada folder Admin, dan di dalam hyperlink memanggil nilai dari $aktif dan menampilkan status akun dari setiap data pelanggan.

1. Melakukan update data pada kolom aktif.

Cth :

public function update($id = null, $isi = 1)

{

$model = new Pelanggan\_M();

if ($isi == 0) {

$isi = 1;

} else {

$isi = 0;

}

$data = [

'aktif' => $isi

];

$model -> update($id, $data);

return redirect() -> to(base\_url("/Admin/Pelanggan"));

}

Artinya, function update terdapat $id yang bernilai kosong dan $isi yang bernilai 1, lalu di dalam function update terdapat objek $model yang memanggil Pelanggan\_M, yang di dalamnya terdapat $table tblpelanggan, artinya tabel yang dihubungkan oleh model ke dalam database adalah tblpelanggan, dan di dalam Pelanggan\_M terdapat $allowedFields yaitu kolom [‘aktif’], lalu terdapat pengujian, yaitu apabila $isi bernilai 0, maka $isi akan diubah nilai nya menjadi 1, dan apabila $isi bernilai 1, maka nilainya akan dirubah menjadi 0. Lalu $dasta berbentuk array digunakan untuk mengirimkan data yang akan diupdate, $data berisi key aktif dengan value mengambil nilai dari $isi. $model menjalankan perintah update pada $data sesuai dengan $id. Setelah melakukan update data, maka halaman akan dikembalikan ke dalam halaman base\_url() dengan alamat localhost/restoran-ci4 lalu disambungkan dengan /Admin/Pelanggan.

1. Melakukan delete data di dalam pelanggan.php

Cth :

public function delete($id = null)

{

$model = new Pelanggan\_M();

$model -> delete($id);

return redirect() -> to(base\_url("/Admin/Pelanggan"));

}

Artinya, function delete terdapat $id dengan nilai null berarti $id bernilai kosong, lalu objek $model yang memanggil Pelanggan\_M yang berisi $table tblpelanggan, artinya table yang dihubungkan oleh model ke dalam database adalah tblpelanggan, lalu terdapat $primaryKey [‘idpelanggan’], $primaryKey digunakan untuk melakukan delete data. $model berisi perintah untuk delete data pada $id. Apabila selesai melakukan delete data, maka halaman akan dikembalikan ke dalam halaman base\_url() dengan alamat localhost/restoran-ci4 lalu disambungkan dengan /Admin/Pelanggan.

1. Query builder adalah class yang disediakan oleh codeigniter yang digunakan untuk melakukan perintah pada database, seperti select, update, insert, dan delete.
2. Perintah koneksi ke database.

Cth :

$db = \Config\Database::connect();

1. Memilih data dari tblorder menggunakan perintah select pada query.

Cth :

public function index()

{

$db = \Config\Database::connect();

$sql = "SELECT \* FROM tblorder ORDER BY status ASC";

$result = $db -> query($sql);

$row = $result -> getResult();

echo "<pre>";

print\_r($row);

echo "</pre>";

}

Artinya, function index berisi $db yang digunakan untuk menyambungkan ke dalam database, lalu $sql yang berisi perintah untuk memilih data / SELECT data dari tblorder yang diurutkan dari status dan diurutkan dengan metode ASC / dari kecil ke besar. Lalu $result berisi $db yang menjalankan perintah query pada sql. $row berisi $result yang menjalankan perintah getResult / perintah untuk mengambil data. $row ditampilkan menggunakan print\_r dan dirapikan tampilannya menggunakan echo “pre”.

1. Menampilkan salah satu kolom.

Cth :

public function index()

{

$db = \Config\Database::connect();

$sql = "SELECT \* FROM tblorder ORDER BY status ASC";

$result = $db -> query($sql);

$row = $result -> getResult();

echo $row[0] -> tglorder;

}

Artinya, function index berisi $db yang digunakan untuk menyambungkan ke dalam database, lalu $sql yang berisi perintah untuk memilih data / SELECT data dari tblorder yang diurutkan dari status dan diurutkan dengan metode ASC / dari kecil ke besar. Lalu $result berisi $db yang menjalankan perintah query pada sql. $row berisi $result yang menjalankan perintah getResult / perintah untuk mengambil data, karena hasil dari pengambilan nilai adalah array yang terdapat objek di dalamnya, maka untuk menambil nilai salah satu kolom yaitu dengna memanggil index yang akan dipanggil dan kolom yang akan ditampilkan. $row memanggil index 0 dan memilih kolom tglorder untuk ditampilkan.

1. Menampilkan banyak data di dalam query menggunakan foreach.

Cth :

public function index()

{

$db = \Config\Database::connect();

$sql = "SELECT \* FROM tblorder ORDER BY status ASC";

$result = $db -> query($sql);

$row = $result -> getResult();

foreach ($row as $key ) {

echo $key -> total;

echo "<br>";

}

}

Artinya, function index berisi $db yang digunakan untuk menyambungkan ke dalam database, lalu $sql yang berisi perintah untuk memilih data / SELECT data dari tblorder yang diurutkan dari status dan diurutkan dengan metode ASC / dari kecil ke besar. Lalu $result berisi $db yang menjalankan perintah query pada sql. $row berisi $result yang menjalankan perintah getResult / perintah untuk mengambil data, lalu untuk menampilkan banyak data, maka menggunakan foreach, di dalam foreach $row sebagai $key, $key memanggil kolom total. Maka semua hasil dari kolom total akan ditampilkan. Karena hasil dari getResult adalah berbentuk object, maka untuk memanggil kolom diberi tanda panah / ->.

1. Menampilkan hasil dengan menggunakan array.

Cth :

public function index()

{

$db = \Config\Database::connect();

$sql = "SELECT \* FROM tblorder ORDER BY status ASC";

$result = $db -> query($sql);

$row = $result -> getResult('array');

echo $row[2]['idpelanggan'];

}

Artinya, function index berisi $db yang digunakan untuk menyambungkan ke dalam database, lalu $sql yang berisi perintah untuk memilih data / SELECT data dari tblorder yang diurutkan dari status dan diurutkan dengan metode ASC / dari kecil ke besar. Lalu $result berisi $db yang menjalankan perintah query pada sql. $row berisi $result yang menjalankan perintah getResult / perintah untuk mengambil data, hasil data akan berbentuk array karena di dalam getResult diberi array, maka hasilnya akan menjadi array, sehingga untuk pemanggilan dari data, maka memanggil index dan kolom yang ingin ditampilkan. Seperti, $row[2][‘idpelanggan’], maka $row akan menampilkan data dari index 2 dengan kolom idpelanggan.

1. Menampilkan hasil getResult berbentuk array menggunakan foreach.

Cth :

public function index()

{

$db = \Config\Database::connect();

$sql = "SELECT \* FROM tblorder ORDER BY status ASC";

$result = $db -> query($sql);

$row = $result -> getResult('array');

foreach ($row as $key) {

echo $key['tglorder'];

echo "<br>";

}

}

Artinya, function index berisi $db yang digunakan untuk menyambungkan ke dalam database, lalu $sql yang berisi perintah untuk memilih data / SELECT data dari tblorder yang diurutkan dari status dan diurutkan dengan metode ASC / dari kecil ke besar. Lalu $result berisi $db yang menjalankan perintah query pada sql. $row berisi $result yang menjalankan perintah getResult / perintah untuk mengambil data, di dalam foreach terdapat $row sebagai $key, lalu $key memanggil kolom tglorder dan ditampilkan menggunakan echo.

**Saya Belum Mengerti**