|  |  |
| --- | --- |
| **Materi** | **Nilai** |
| PHP FRAMEWORK CI minggu ke 20 (video 34-52 Tutorial Codeigniter-4) | 95 |

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

1. Tutorial CodeIgniter-4 pada bagian Order Restoran (2) yaitu Sql Select. Di dalam controller order ini tidak perlu menggunakan models lagi untuk menyambungkan dengan database nya tetapi menggunakan sql seperti ini

$db = \Config\Database::connect();

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

Yang berarti objek $db ini menyambungkan dengan database lalu $sql ini untuk memilih data yang akan diambil yaitu di vorder dan mengurutkan data nya dari bawah. Setelah itu

$result = $db->query($sql);

$row = $result->getResult('array');

$data = [

'judul' => 'DATA ORDER',

'order' => $row

];

echo view('order/select', $data);

yang berarti $result ini untuk menyambungkan $db dengan $sql lalu $row nya untuk menjadikan data yang diambil agar berbentuk array dan $data untuk membuat variable yang bisa digunakan nanti di dalam view nya.

2. Tutorial CodeIgniter-4 pada bagian Order Restoran(3) yaitu Coding Paging. Untuk menggunakan pager maka di dalam controller harus ditambahkan ini terlebih dahulu $pager = \Config\Services::pager(); kemudian di dalam file view nya yaitu select.php dari order ditambahkan <?= $pager->makeLinks(1, $perPage, $total, 'bootstrap') ?> yang berarti php ini untuk menampilkan $perPage, $total, yang sebelumnya sudah diatur di controller. Untuk itu di dalam controller nya dibuat seperti ini

 $total = count($row);

$tampil = 3;

        if (isset($\_GET['page'])) {

            $page = $\_GET['page'];

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

            $sql = "SELECT \* FROM vorder ORDER BY status ASC LIMIT $mulai, $tampil";

        } else {

            $sql = "SELECT \* FROM vorder ORDER BY status ASC LIMIT 0, $tampil";

        }

Jadi $total untuk menotal jumlah data yang ada, $tampil sebagai tempat menentukan berapa data yang akan ditampilkan. Setelah itu di if isset, di dalam if isset ini untuk mengatur apabila sudah memilih salah satu page maka $page adalah page yang dipilih misalkan page 2 maka $page adalah 2, $mulai adalah hasil dari $tampil dikali $page dikurangi $tampil yang nantinya digunakan di dalam sql untuk mengurutkan dari $mulai dengan jumlah data sebanyak $tampil. Setelah itu ditambahkan

$result = $db->query($sql);

$row = $result->getResult('array');

Untuk menjalankan sql nya dan di dalam $data nya. lalu ditambahkan

'pager' => $pager,

'perPage' => $tampil,

'total' => $total

Agar bisa digunakan nanti di dalam view yaitu dengan menggunakan php yang sudah dipasang tadi.

3. Tutorial CodeIgniter-4 pada bagian Order Restoran yaitu Coding Pembayaran Order-1.

Pada bagian ini adalah untuk mengatur apabila ada pelanggan yang statusnya belum bayar maka di dalam if else view select dari ordernya yang else diubah menjadi

else {

     $status = "<a href='" . base\_url("/admin/order/find") . "/" . $value['idorder'] . "'>BAYAR</a>";

} Agar tulisan BAYAR ini menjadi hyperlink yang ketika ditekan dia akan mengarah ke controller order bagian function find lalu mengambil idorder dari pelanggan yang belum bayarnya. Di dalam function baru yaitu function find ditambahkan $db untuk menyambungkan dengan database dan 2 sql yang menyambungkan dengan vorder dan vorderdetail lalu untuk nama $result nya adalah $row dan $detail. Setelah itu membuat

$data = [

            'judul' => 'PEMBAYARAN PELANGGAN',

            'order' => $row,

            'detail' => $detail

        ];

Dan $data ini nanti digunakan di dalam view update.php yang ada di dalam folder order. Dan di dalam file ini dibuat judul menggunakan $judul yang diambil dari $data, lalu tulisan pelanggan dengan mengambil $order[0][‘pelanggan’] yang berarti dari $order dipilih indeks ke 0 dan diambil yang array asosiatif pelanggan. Lalu tulisan tanggal seperti ini

<p>Tanggal : <?= date("d-m-Y", strtotime($order[0]['tglorder'])) ?></p> dengan menggunakan date “d-m-Y” yang berarti day-month-year ututannya dan diambil dari tglorder yang dipilih. Lalu untuk tulisan total order nya menggunakan bold agar tulisannya tebal dan number\_format agar angka nya otomatis menjadi seperti sebuah harga yang tiap 3 angka dari belakang tertambah koma <p>Total : <b><?= number\_format($order[0]['total']) ?></b></p>

4. Tutorial CodeIgniter-4 pada bagian Order Restoran(5) yaitu Coding Pembayaran Order-2. Untuk membuat rincian order di dalam table, maka didalam table itu sebelum tr td nya ditambahkan foreach dari $detail seperti ini

<?php $no = 1; ?>

<?php foreach ($detail as $value) : ?>

Yang berarti dia akan melooping dari $detail yang diambil dari vorderdetail sebagai $value nya, jadi jika ingin mengambil data dan dimasukkan di dalam kolom table nya menggunakan $value lalu nama kolom yang ingin dipilih datanya [‘menu] misalkan, jadinya seperti ini $value[‘menu]. Setelah itu membuat form untuk pembayaran, yang pertama input type number untuk kolom bayar, kedua ada

<div class="form-group">

<input type="hidden" name="total" value="<?= $order[0]['total'] ?>" required class="form-control">

<input type="hidden" name="idorder" value="<?= $order[0]['idorder'] ?>" required class="form-control">

</div>

Yang digunakan untuk mengambil total dari pembelian dan idorder nya berapa, untuk fungsinya digunakan nanti.

5. Tutorial CodeIgniter-4 pada bagian Order Restoran(6) yaitu Coding Pembayaran Order-3. Karena form tadi setelah submit diarahkan ke dalam function update dalam controller order. Maka di dalam funcntion update dibuatkan seperti ini

public function update()

    {

        $idorder = $\_POST['idorder'];

        $total = $\_POST['total'];

        $bayar = $\_POST['bayar'];

        if ($bayar < $total) {

            session()->setFlashdata('info', 'PEMBAYARAN KURANG');

            $this->find($idorder);

        } else {

            $kembali = $bayar - $total;

            $sql = "UPDATE tblorder SET bayar=$bayar, kembali=$kembali, status=1 WHERE idorder=$idorder";

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

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

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

        }

    }

Yang berarti $idorder, $total, $bayar akan dipost hasilnya nanti dengan ketentuan jika $bayar kurang dari $total atau uang yang dibayarkan kurang dari total pembelian maka dia akan menampilkan flashdata info dengan tulisan PEMBAYARAN KURANG. Lalu menjalankan ulang function find. Untuk itu di dalam update.php nya agar flashdata berjalan ditambahkan

<div class="row">

    <div class="col">

        <?php

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

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

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

            echo '</div>';

        }

        ?>

    </div>

</div>

Jika pembayaran tidak kurang dari total, maka dia akan menghitung $kembali dulu yaitu $bayar dikurangi $total lalu mengupdate data-data yang sudah diinput ke dalam tblorder sesuai dengan idordernya, setelah itu dijalankan menggunakan sql seperti biasa dan jika sudah berhasil update langsung kembali ke dalam tampilan order awal.

6. Tutorial CodeIgniter-4 pada bagian Order Detail(1) yaitu Coding Order Detail. Sama seperti sebelumnya, untuk orderdetail di dalam models nya menyambungkan dengan vorderdetail, controller nya sama seperti yang ada di dalam folder admin dan functionnya baru index yang isinya sama dengan function read yang ada di dalam kategori, lalu untuk tabelnya bisa disesuaikan dengan mengambil beberapa kolom yang ada di vorderdetail yang dipilih untuk ditampilkan.

7. Tutorial CodeIgniter-4 pada bagian Order Detail(2) yaitu Coding Pencarian Order Detail. Di dalam select.php di views ditambahkan form yang berfungsi untuk mencari tanggal order detail yang ingin ditampilkan, jadi form nya ada input date untuk tanggal awal dan tanggal akhir lalu submit nya diberi nama cari. Setelah menekan submit, form ini akan diarahkan ke controller orderdetail di function cari, di dalam function cari isinya seperti ini

public function cari()

    {

        if (isset($\_POST['awal'])) {

            $awal = $\_POST['awal'];

            $sampai = $\_POST['sampai'];

            $sql = "SELECT \* FROM vorderdetail WHERE tglorder BETWEEN '$awal' AND '$sampai'";

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

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

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

            $data = [

                'judul' => 'DATA RINCIAN ORDER',

                'orderdetail' => $row,

            ];

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

        }

    }

Yang berarti apabila mengisset tanggalnya maka tanggal awal yang dipilih akan diletakkan di dalam $awal dan tanggal akhir yang dipilih akan diletakkan di dalam $sampai, selanjutnya $sql akan mencari data yang tanggalnya berada di antara tanggal awal yang dipilih sampai dengan tanggal akhir yang dipilih. Setelah itu $sql akan dijalankan oleh $db $result dan $row lalu ada $data yang mengambil data yang berisi judul dan orderdetail yang berisi $row yang nantinya $data ini akan digunakan di dalam file cari.php yang ada di dalam folder orderdetail di views. File cari.php ini berisi code untuk menampilkan table dari vorderdetail dengan data yang diambil dari $row nya. Jadi setelah mengatur tanggal awal dan sampai lalu dicari dia akan menampilkan data-data order detail berdasarkan tanggal yang sudah dipilih.

8. Tutorial CodeIgniter-4 pada bagian User Restoran(1) yaitu Select User. Sama seperti sebelumnya, untuk user di dalam models nya menyambungkan dengan tbluser, controller nya sama seperti controller lain yang ada di dalam folder admin dan functionnya baru index yang isinya sama dengan function read yang ada di dalam kategori yaitu untuk menyambungkan dengan models dan menggunakan models untuk menampilkan data dari tbluser, lalu untuk tabelnya bisa disesuaikan dengan mengambil beberapa kolom yang ada di tbluser yang dipilih untuk ditampilkan melalui file select di folder user views.

9. Tutorial CodeIgniter-4 pada bagian User Restoran(2) yaitu Insert User. Untuk menginsert suatu data menggunakan coeigniter, pertama adalah harus menambahkan $allowedFields ke dalam models nya. seperti ini protected $allowedFields = ['user', 'email', 'password', 'level', 'aktif']; yang berarti dari tbluser, nama-nama kolom yang disebutkan di dalam array $allowedFields itu adalah kolom yang bisa diubah, selanjutnya ke dalam controller dibuatkan 1 function yaitu create, jadi button tambah data yang ada di select.php di dalam folder user views akan mengarahkan untuk menggunakan function create, function create nya seperti ini

public function create()

    {

        $data = [

            'level' => ['Admin', 'Koki', 'Kasir', 'Gudang'],

        ];

        return view("user/insert", $data);

    }

Yang berarti $data ini bisa digunakan di dalam file insert.php yang ada di folder user views. Setelah itu di dalam file folder views dibuatkan sebuah form seperti biasanya yaitu input text untuk user, input email untuk email, input password untuk password, dan select option untuk memilih level yang diambil dari function create yaitu $level. Code nya seperti ini

 <select class="form-control" name="level" id="user">

                <?php foreach ($level as $key) : ?>

                    <option value="<?= $key ?>">

                        <td><?= $key ?></td>

                    </option>

                <?php endforeach; ?>

</select>

Jadi select ini melooping data yang diambil dari $level, karena isi data $level ini diambil key nya saja maka $level as $key lalu option nya menampilkan $key nya. Setelah mengisi form dan menekan submit, selanjutnya akan diarahkan ke function insert di controller. Di dalam function insert isinya hanya seperti ini

public function insert()

    {

        $model = new User\_M();

        $model->insert($\_POST);

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

    }

Yang berarti $model sebagai wadah untuk menggunakan User\_M, lalu dari model akan mempost data yang diinsert tadi dan kembali ke dalam tampilan data user awal. Dengan begitu data-data yang diinsert tadi sudah bisa dilihat melalui table karna proses insert berhasil.

10. Tutorial CodeIgniter-4 pada bagian User Restoran(3) yaitu Delete User. Untuk menghapus suatu data, maka di dalam models User\_M nya harus ditambahkan protected $primaryKey = 'iduser'; untuk mengambil primarykey yang ada di dalam tbluser nya. Setelah itu, karena ketika menghapus ini mengarahkan ke dalam function delete, maka di controller dibuatkan function delete nya dulu seperti ini

public function delete($id = null)

    {

        $model = new User\_M();

        $model->delete($id);

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

    }

Jadi dia akan mengambil id dari data yang dipilih untuk dihapus, setelah itu $model untuk wadah dari model User\_M dan akan menjalankan delete berdasarkan id yang sudah dipilih, jika delete berhasil dia akan kembali ke tampilan awal data user dengan perubahan yang sudah terjadi yaitu data yang dihapus jadi hilang. Setelah itu untuk mengaktifkan banned dan membanned kan aktif, maka di dalam select.php nya user dibagian td dari status diubah menjadi

<td>

<a href="<?= base\_url() ?>/admin/user/update/<?= $value['iduser'] ?>/<?= $value['aktif'] ?>"><?= $aktif ?></a>

</td>

Dengan begitu dia akan berbentuk sebuah hyperlink yang tulisannya sesuai dengan statusnya aktif atau tidak karena sebelumnya sudah di if else dan ketika sudah ditekan hyperlinknya dia akan mengarahkan ke function update dan mengambil iduser dari data yang akan diupdate, maka di dalam function update isinya seperti ini

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

    {

        $model = new User\_M();

        if ($isi == 0) {

            $isi = 1;

        } else {

            $isi = 0;

        }

        $data = [

            'aktif' => $isi

        ];

        $model->update($id, $data);

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

    }

Jadi dia mengambil id nya lalu membuat variable isi yang berisi 1, kolom aktif ini akan dilihat dengan menggunakan $isi yang apabila sebelumnya 0 menjadi 1, dan sebelumnya 1 menjadi 0. Setelah itu diupdate berdasarkan id nya dengan $data nya dan kembali menampilkan ke tampilan awal data user dengan perubahan status yang tinggal pencet langsung berubah.

11. Tutorial CodeIgniter-4 pada bagian User Restoran(4) yaitu Validation. Yang pertama adalah menambahkan $validation dulu di dalam models nya. Seperti ini

protected $validationRules = [

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

        'email' => 'valid\_email'

    ];

    protected $validationMessages = [

        'user' => [

            'alpha\_numeric\_space' => 'Tidak boleh menggunakan simbol',

            'min\_length' => 'Minimal 3 huruf',

            'is\_unique' => 'Data sudah digunakan'

        ],

        'email' => [

            'valid\_email' => 'Email Salah'

        ]

    ];

Jadi ada 2 $validation yaitu $validationRules untuk mengatur kolom mana yang akan di validasi dan model validasi apa yang akan digunakan lalu $validationMessages untuk memberikan pesan apa yang akan ditampilkan apabila ada error. Selanjutnya di dalam controller pada bagian function insert diubah seperti ini

public function insert()

    {

        if (isset($\_POST['password'])) {

            $data = [

                'user' => $\_POST['user'],

                'email' => $\_POST['email'],

                'password' => password\_hash($\_POST['password'], PASSWORD\_DEFAULT),

                'level' => $\_POST['level'],

                'aktif' => 1

            ];

            $model = new User\_M();

            if ($model->insert($data) === false) {

                $error = $model->errors();

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

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

            } else {

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

            }

        }

    }

Yang berarti jika password sudah diisi atau sudah dijalankan, maka dia akan membuat sebuah $data yang berfungsi sebagai tempat menyimpan data dari data yang sudah diinsert untuk dipost sebelumnya, lalu menggunakan $model sebagai wadah dari models dan jika data yang diinsert kan terjadi kesalahan, maka dia akan menjalankan errors dengan menampilkan pesan error nya lalu kembali ke tampilan awal insert data kosong. Jika data yang dimasukkan benar, maka proses insert berhasil dan kembali ke tampilan data user. Disini password sudah di password\_hash, jadi password yang diinsert dan dipost akan otomatis terenkripsi sebagai keamanan pengguna.

12. Tutorial CodeIgniter-4 pada bagian User Restoran(5) yaitu Update Data. Untuk melakukan update, yang bisa diganti hanya email dan level saja. Caranya yang pertama yaitu membuat function find karena ketika menekan aksi bertanda pensil dia akan menuju ke function find, di dalam function find isinya seperti ini

public function find($id = null)

    {

        $model = new User\_M();

        $user = $model->find($id);

        $data = [

            'judul' => 'UPDATE DATA',

            'user' => $user,

            'level' => ['Admin', 'Koki', 'Kasir', 'Gudang'],

        ];

        return view("user/update", $data);

    }

Yang berarti dia mengambil id sesuai data yang dipilih lalu menggunakan models untuk mengambil semua data dari id yang dipilih dan membuat variable data untuk menyimpan data judul, user, level, yang nanti bisa digunakan di dalam file view nya yaitu update.php, pada file ini, form nya hanya disediakan untuk update email dan level saja, jadi password dan user nya dihilangkan. Judulnya bisa menggunakan $judul dan untuk mengambil id nya maka yang letaknya form user tadi diubah menjadi iduser namanya dan value nya seperti ini value="<?= $user['iduser'] ?>" lalu untuk type nya hidden karena ini untuk mengambil id nya saja. Setelah itu value pada email nya juga seperti ini value="<?= $user[email] ?>" dan untuk optionnya dia menggunakan if ($user[‘level’]==$key){echo “selected”;} yang berarti dia akan menampilkan level sesuai dengan id yang dipilih, misalkan memilih id milik koki, maka ketika diupdate yang muncul di select optionnya adalah koki. Jika sudah melakukan submit, selanjutnya akan diarahkan ke function ubah yang berisi seperti ini

public function ubah()

    {

        $id = $\_POST['iduser'];

        $data = [

            'email' => $\_POST['email'],

            'level' => $\_POST['level']

        ];

        $model = new User\_M();

        $model->update($id, $data);

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

    }

Yang berarti $id yang akan dipost adalah id dari iduser data yang dipilih, lalu $data berisi untuk kolom email akan diisi dengan input bernama email, begitu juga levelnya. Setelah itu dia akan menjalankan update dari $id yang dipilih dengan membawa $data nya juga untuk diupdate. Setelah update berhasil akan diarahkan ke tampilan awal data user.

13. Tutorial CodeIgniter-4 pada bagian Login Admin(1) yaitu Form Login. Untuk membuat login maka harus dibuatkan controllernya dulu yaitu Login.php yang diletakkan di dalam admin, controller ini menggunakan class Login lalu disambungkan ke basecontroller sama seperti class lainnya yang ada di dalam admin controller, lalu disambungkan juga dengan models User\_M karena login ini berhubungan dengan tbluser. Di dalam class login ada function index yang apabila nanti ketika url nya diketikkan localhost/restoran-ci4/admin/login dia akan menjalankan controller login ini. Yang pertama dijalankan adalah function index seperti ini

public function index()

    {

        if ($this->request->getMethod() == 'post') {

            $email = $this->request->getPost('email');

            $password = $this->request->getPost('password');

            $model = new User\_M();

            $user = $model->where(['email' => $email, 'password' => $password, 'aktif' => 1])->first();

            echo "<pre>";

            print\_r($user);

            echo "</pre>";

            $this->setSession($user);

        } else {

            # code...

        }

        return view('template/login');

    }

Disitu digunakan $this-> bukannya $\_POST tujuannya untuk melindungi keamanan pada saat login, jadi apabila form bermethod post yang ada di view nya nanti diisi form email dan password nya yang berarti atas request an dari input user nanti akan diambil data yang dimasukkan misalkan data emailnya maka nanti dengan model dia akan mencari data yang memiliki email seperti yang diinputkan lalu untuk passwordnya juga seperti itu, dan apabila statusnya tidak aktif maka tidak bisa login, lalu untuk first nya berarti dia mengambil data yang ditemukan terlebih dahulu yang sesuai dengan data yang diinputkan dan akan digunakan untuk login. Lalu ada $this->setSession($user) sebagai wadah untuk menyimpan data yang diambil dari session $user. Untuk itu ada function seperti ini

public function setSession($user)

    {

        $data = [

            'user' => $user['user'],

            'email' => $user['email'],

            'level' => $user['level'],

            'loggedIn' => true

        ];

        session()->set($data);

    }

Dari $data ini berarti dia akan menyimpan kolom user dengan user yang dipilih, kolom email dengan email yang dipilih, begitu juga passwordnya apabila login benar dan data yang dibuat login itu memang ada di database nya. Lalu controller ini akan digunakan di dalam file login.php yang ada di dalam view, untuk view sebagai tempat membuat form, jadi ketika membuka url setelah admin/login maka yang akan tampil adalah form dari view nya yang menggunakan controller login. Jika login sudah berhasil, bisa dicek lagi ke dalam admin nya dengan menambahkan div yang berisi php dulu di dalam view admin.php nya, letak div ini berada di bawah navbar-brand Admin Page, codenya

<div>

<?php

   if (!empty(session()->get('user'))) {

echo session()->get('user');

}

?>

</div>

<div>

<?php

if (!empty(session()->get('email'))) {

echo session()->get('email');

}

?>

</div>

<div>

<?php

if (!empty(session()->get('level'))) {

echo session()->get('level');

}

?>

</div>

Yang berarti apabila user, email, dan level tidak kosong atau sudah melakukan login dan mengetahui data yang diinputkan mengambil data yang mana, maka dia akan menampilkan data user, email, dan level dibawah navbar.brand admin page.

14. Tutorial CodeIgniter-4 pada bagian Login Admin(2) yaitu Menampilkan View dan Mengatur Link. Di dalam controller login pada bagian function index di dalam if nya yang pertama ditambahkan

if (empty($user)) {

$data = [

'info' => "Email atau Password Salah"

];

} else {

$this->setSession($user);

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

}

Yang berarti apabila login kosong atau tidak cocok dengan data yang ada di dalam database, maka dia akan menjalankan info atau error yang sudah dipasang di models sebelumnya dan bisa langsung digunakan ke dalam $data. Dan apabila tidak kosong dan data yang dimasukkan untuk login berhasil karena sesuai dengan yang di database, maka dia akan menjalankan tampilan admin page. Agar info berjalan atau keluar tanda info error nya, maka di dalam login.php view ditambahkan

<?php

if (!empty($info)) {

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

echo $info;

echo '</div>';

}

?>

Yang berarti apabila ketentuan dari info berjalan atau ada error, maka akan keluar isi dari info nya dibagian atas dari form loginnya. Setelah itu tampilan navbar-brand pada admin.php bisa diganti dengan mengcopas contoh navbar yang ada di bootstrap dan menyesuaikan dengan tampilan sesuai keinginan. Lalu di dalam routes.php pada bagian $routes->get('/admin', 'Admin\Login::index'); ini dijalankan atau dibuka comment nya agar ketika di url nya hanya mengetikkan localhost/restoran-ci4/admin saja dia akan menjalankan controller login di function index nya tanpa harus masuk url /login nya.

15. Tutorial CodeIgniter-4 pada bagian Login Admin(3) yaitu Menggunakan Filter. Yang pertama adalah membuat sebuah file dulu di dalam folder filters, file ini bernama Auth.php. Didalam file Auth.php ini berisi php yang didalamnya ada

namespace App\Filters;

use CodeIgniter\HTTP\RequestInterface;

use CodeIgniter\HTTP\ResponseInterface;

use CodeIgniter\Filters\FilterInterface;

Agar dia bisa menjalankan filter, jadi harus ada bagian itu, setelah itu ada class yang diberi nama sama seperti file nya yaitu Auth dengan tambahan implements FilterInterface karena dia berupa class di filter. Di dalam class nya ada function before yang berisi

public function before(RequestInterface $request, $arguments = null)

    {

        if (!session()->get('loggedIn')) {

            return redirect()->to(base\_url('/login'));

        }

    }

Jadi apabila session loginnya masih belum terisi, maka dia akan menjalankan login terus sampai session loginnya terisi. Jika file ini sudah jadi, selanjutnya ke dalam folder Config dan masuk ke dalam file Filters.php. di dalam Filters.php pada bagian public $aliases didalamnya tambahkan 'Auth' => \App\Filters\Auth::class agar dia menjalankan class Auth sebagai filter. Selanjutnya di dalam Routes.php pada bagian routes groupnya ditambahkan ['filter' => 'Auth'] agar dia menggunakan filter dari class Auth. Setelah itu isi dari routes group ini ada $routes->add('kategori', 'Admin\Kategori::read'); yang berarti ketika diketikkan localhost/restoran-ci4/admin/kategori dia akan menjalankan controller Kategori dan menjalankan function read nya, tetapi karena ini sudah diberi filter dari class Auth, maka dia akan menjalankan filter nya dulu yaitu apabila belum login maka tidak bisa masuk ke dalam kategori itu. Agar menu-menu lainnya sama seperti kategori maka harus ditambahkan seperti itu juga, dengan nama class nya lalu koma lalu Admin/ nama class nya :: nama function pertama yang ada di dalamnya. Seperti ini

$routes->add('menu', 'Admin\menu::index');

$routes->add('pelanggan', 'Admin\pelanggan::index');

$routes->add('order', 'Admin\order::index');

$routes->add('orderdetail', 'Admin\orderdetail::index');

$routes->add('user', 'Admin\user::index');

Dengan begitu apabila belum melakukan login, maka untuk menjalankan controller diatas tidak akan bisa. Setelah itu di dalam admin.php pada views ditambahkan

<li class="nav-item mt-2 ml-2">

<a href="<?= base\_url('admin/login/logout') ?>">Logout</a>

</li>

Agar ada navbar untuk logoutnya, dan ketika ditekan logout, dia akan menjalankan controller login dan masuk ke dalam function logout. Maka di dalam controller login ditambahkan function logout seperti ini

public function logout()

    {

        session()->destroy();

        return redirect()->to(base\_url('/login'));

    }

Jadi ketika ditekan logout, dia akan mendestroy atau mematikan semua session yang sebelumnya digunakan untuk login, setelah itu menampilkan tampilan login lagi untuk bisa mengisi session lagi.

16. Tutorial CodeIgniter-4 pada bagian Login Admin(4) yaitu Mengatur Routing. Di dalam menu-menu seperti kategori, menu, pelanggan, order, dll. Menggunakan controller yang class nya sesuai dengan namanya tetapi juga ada function-function yang digunakan untuk menu itu tadi. Jika dituliskan ke dalam url, tanpa harus login, function ini masih bisa dijalankan karena belum di routes untuk setiap function yang digunakan disetiap menunya. Untuk itu di dalam routes group harus manual mengisi semua url yang masih bisa dijalankan tanpa harus login dulu agar ter filter oleh class Auth tadi. Misalkan di dalam menu untuk ‘Kategori’ ada url untuk menjalankan function create, maka di dalam routes nya ditambahkan $routes->add('kategori/create', 'Admin\Kategori::create'); agar ketika diketikkan url seperti itu, dia tidak akan langsung menuju create dari kategori tetapi melalui proses filter dulu yaitu login. Setelah itu ada update yang menjalankan function find maka didalam routes group nya ditambahkan $routes->add('kategori/find/(:any)', 'Admin\Kategori::find/$1'); alasannya juga sama seperti menambahkan routes untuk function create, setelah itu ada hapus yang menjalankan function delete, maka didalam routes groupnya ditambahkan $routes->add('Kategori/delete/(:any)', 'Admin\Kategori::delete/$1'); dan seterusnya. Hal ini juga berlak untuk menu-menu lainnya agar lebih aman apabila ada orang mengarang url maka dia tidak akan langsung menjalankan functionnya tetapi harus login dulu.

17. Tutorial CodeIgniter-4 pada bagian Login Admin(5) yaitu Mengatur Hak Akses & Verfikasi Password. Mengatur Hak Akses seperti apabila yang login admin, maka dia bisa melihat semua menu-menu yang disediakan, apabila yang login koki maka dia hanya bisa melihat order detail dan apabila yang login kasir maka dia hanya bisa melihat order dan order detail. Caranya di dalam admin.php pada views dibagian ul li untuk menunya ditambahkan seperti ini, misalkan untuk level kasir maka

<?php

if ($level === "Kasir") : {

?>

<li class="list-group-item"><a href=" <?= base\_url('/admin/order') ?>">Order</a></li>

<li class="list-group-item"><a href=" <?= base\_url('/admin/orderdetail') ?>">Order Detail</a></li>

<?php

endif;

?>

Yang berarti apabila level nya Kasir, maka dia akan menampilkan li itu saja. Nah $level ini ditambahkan di dalam php yang ada di navbar level. Ditambahkan seperti ini $level = session()->get('level'); jadi $level fungsinya untuk mengetahui level apa yang dimiliki oleh user yang login. Setelah itu dibuatkan juga seperti php diatas untuk koki dan admin. Selanjutnya agar password yang digunakan untuk login terenkripsi, maka

if (empty($user)) {

            $data = [

                'info' => "Email atau Password Salah"

            ];

            } else {

                $this->setSession($user);

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

            }

Diubah menjadi

if (empty($user)) {

                $data = [

                    'info' => "Email Salah"

                ];

            } else {

                if (password\_verify($password, $user['password'])) {

                    $this->setSession($user);

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

                } else {

                    $data = [

                        'info' => "Password Salah"

                    ];

                }

            }

Yang sebelumnya hanya mengoreksi $user biasa, diubah menjadi apabila $user untuk emailnya kosong atau data nya tidak ada dia akan menjalankan error dengan pesan “Email Salah” dan apabila data user nya ada maka dia menjalankan else nya yaitu jika passwordnya terverifikasi antara password yang digunakan untuk login dengan password yang sudah terenkripsi di dalam database nya maka dia berhasil login dan masuk ke admin page. Jika password yang dimasukkan dengan password yang ada didatabase salah, maka dia akan menjalankan else yang ada di dalam else nya yaitu $data info berisi “Password Salah”. Terakhir mengatur $routes->setDefaultController(‘Login'); menjadi $routes->setDefaultController(‘Home');

**SELESAI**

**Saya Belum Mengerti**

1.