YII FRAMEWORK 2.0

The Fast, Secure & Professional Framework

#5
User Authorization

Let’s start it!

Outline

- User Authorization
  - definisi
  - *ACF* vs *RBAC*
  - implementasi authorization
  - penggunaan extension authorization

User Authorization

Pengaturan hak akses *user* terhadap aplikasi

Definisi

User authorization (otorisasi) adalah pengaturan hak akses *user* terhadap aplikasi. Hal ini berbeda dengan otentikasi yang hanya mengecek apakah *user* sudah *login* atau belum saja, *authorization* mengatur apa yang boleh dan tidak boleh dilakukan oleh *user* yang telah *login* atau belum pada aplikasi.

Authorization cocok diterapkan untuk aplikasi yang memiliki beberapa level *user*, misalnya level *guest* yang hanya bisa *view*, level *member* bisa men-*download*, dan level *admin* yang bisa melakukan apapun yang bisa dilakukan oleh dua level user sebelumnya.

Yii menyediakan dua metode *authorization* yang sederhana, yaitu *Access Control Filter* (ACF) dan *Role-Based Access Control* (RBAC). Pada dasarnya keduanya sama-sama *access control*.

Access Control Filter

Definisi

ACF adalah metode *authorization* yang sederhana yang cocok digunakan untuk aplikasi yang membutuhkan pengaturan hak akses yang *simple*. Sesuai namanya, ACF bekerja dengan mem-*filter* setiap *action* pada suatu *controller* ketika user melakukan *request* terhadap suatu *action*.

Access Control Filter

ACF Dasar

Contoh penggunaan ACF pada suatu *controller* kode sebagai berikut :

Access Control Filter

```
<?php
namespace app\controllers;
...
use yii\filters\AccessControl;
...

class SiteController extends Controller
{
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['logout', 'signup'],
                        'rules' => [
                                    [
                        'actions' => ['signup'],
                        'allow' => true,
                        'roles' => ['?'],
                    ],
                    [
                        'actions' => ['logout'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
                        ...
        ];
    }
```

Access Control Filter

Pada kode tersebut ACF diimplementasikan menggunakan fungsi *behaviors access* dengan *class* yii\filters\AccessControl

```
'access' => [
    'class' => AccessControl::classname(),
    'only' => ['logout', 'signup'],
    ...
```

Only berisi daftar fungsi *action* yang dikenai *filter access* ini, yaitu *logout* (actionLogout), dan *signup* (actionSignup). Jika parameter ini tidak didefinisikan artinya *filter access* akan diimplementasikan pada semua fungsi *action*.

Access Control Filter

```
'rules' => [
  [
    'actions' => ['signup'],
    'allow' => true,
    'roles' => ['?'],
  ],
  [
     'actions' => ['logout'],
     'allow' => true,
     'roles' => ['@'],
  ],
],
```

Rules berisi daftar aturan *filter action* tentang boleh atau tidaknya *user* tertentu mengakses suatu *action*.

- Actions berisi daftar fungsi *action* yang dikenai suatu aturan
- Allow bernilai *boolean* yang menunjukkan boleh tidaknya suatu *access*.
- Roles berkaitan dengan siapa yang dikenai aturan tersebut. Tanda ? menunjukkan *user* yang belum *login* atau *guest*, tanda @ menunjukkan *user* yang sudah *login / authenticated user*. Disamping itu kita bisa menggunakan teks lain (nama: *role: admin, staff*) yang akan men-*trigger* pemanggilan fungsi yii\web\User::can(), dimana masuk ke ranah RBAC.

Access Control Filter

Customize ACF

Lalu bagaimana jika kita ingin membuat *action* yang boleh diakses oleh *user* yang sudah *login* maupun yang belum *login*? Maka kita perlu gunakan dua *roles* sekaligus.

```
'rules' => [
    [
            'actions' => ['index'],
            'allow' => true,
            'roles' => ['?','@'],
    ],
],
```

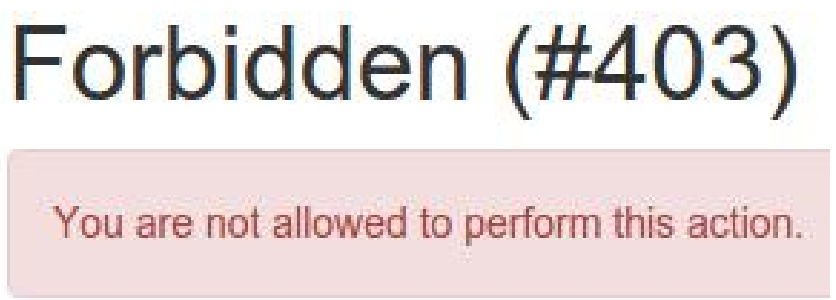
Access Control Filter

Ada dua hal yang akan terjadi ketika *user* mengakses suatu *action* yang mana dia tidak memiliki hak akses akan *action* tersebut.

1. Jika user belum *login* atau *guest*, maka ACF akan memanggil fungsi yii\web\User::loginRequired() untuk me-*redirect browser* ke halaman *login.*

2. Jika *user* sudah *login* atau *authenticated user* maka ACF akan menjalankan yii\web\ForbiddenHttpException

Access Control Filter



Kita juga bisa memodifikasi *pesan* error diatas dengan menggunakan parameter *dennyCallback.*

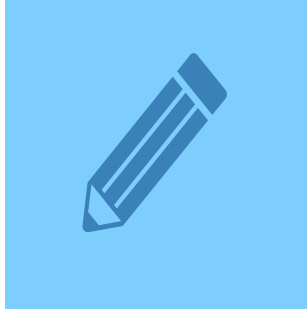
```
<?php
public function behaviors()
{
    return [
        'access' => [
            'class' => AccessControl::className(),
            ...
            'denyCallback' => function ($rule, $action) {
                throw new \yii\web\ForbiddenHttpException('Anda tidak diizinkan untuk mengakses halaman
'.$action->id.' ini!');
            }
        ],
```

Access Control Filter

Kita juga bisa membuat *controller* ini agar hanya bisa diakses oleh *user* dengan ID

```
'access' => [
    'class' => AccessControl::className(),
    'rules' => [
        [
            'allow' => true,
            'roles' => ['@',1],
            'matchCallback' => function ($rule, $action) {
                $user = \Yii::$app->user;
                if(in_array($user->id, $rule->roles))
                                                    return true;
            }
        ],
    ],
    ...
],
```

Access Control Filter



Dynamic ACF

Untuk membuat *Dynamic ACF* kita perlu media penyimpan semisal *database* untukk menyimpan data *route (module, controller, action)* dan ID user yang diperbolehkan mengakses *action* tersebut. Sebagai contoh, kita buat tabel auth sebagai berikut.

```
auth
- module : varchar(60)
- controller : varchar(60)
- action : varchar(60)
- user_id : int(11)
```

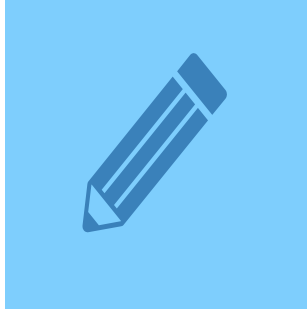
Menggunakan Gii, *generate model* tabel auth.

Access Control Filter

Berikut ini contoh implementasinya pada matchCallback.

```
'rules' => [
    [
        'allow' => true,
        'matchCallback' => function ($rule, $action) {
            $moduleID = $action->controller->module->id;
            $controllerID = $action->controller->id;
            $actionID = $action->id;
            $userID = \Yii::$app->user->id;
            $auth = \app\models\Auth::find()
                ->where([
                    'module' => $moduleID,
                    'controller' => $controllerID,
                    'action' => $actionID,
                    'user_id' => $userID,
                ])
                ->count();
            if($auth>0) return true;
        }
    ],
],
```

Access Control Filter



Sebelum mengujinya, kita perlu masukkan secara *manual* data *action* yang diizinkan untuk userID tertentu ke dalam *database* tabel auth.

module	controller	action	user_id
basic	category	create	2
basic	category	index	2

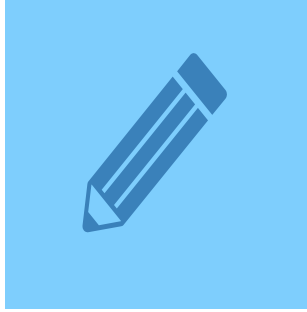
Setelah itu kita bisa testing dengan mengakses fungsi *action* pada CategoryController.

Access Control Filter

Implementasi ACF

Implementasi ACF bisa melalui *event* beforeAction, melalui *behavior access control*, serta bisa meletakkannya pada Bootstrap.

Access Control Filter

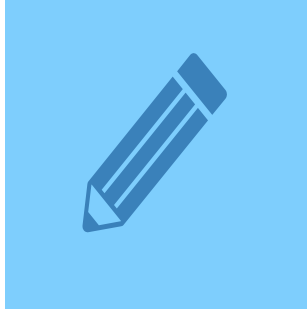


Implementasi ACF pada *Application event* di Config

```
<?php
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'on beforeAction' => function($event){
        $action = $event->action;
        $moduleID = $action->controller->module->id;
        $controllerID = $action->controller->id;
        $actionID = $action->id;
        $user = \Yii::$app->user;
        $userID = $user->id;
        if(!in_array($controllerID,['default','site'])){
            $auth = \app\models\Auth::find()
                ->where([
                    'module' => $moduleID,
                    'controller' => $controllerID,
                    'action' => $actionID,
                    'user_id' => $userID,
                ])
                ->count();
```

```
            if($auth==0) {
                if (!$action instanceof \yii\web\ErrorAction) {
                    if ($user->getIsGuest()) {
                        $user->loginRequired();
                    } else {
                        throw new \yii\web\ForbiddenHttpException('Anda
tidak diizinkan untuk mengakses halaman ' . $action->id . ' ini!');
                    }
                }
            }
        }
    },
```

Access Control Filter

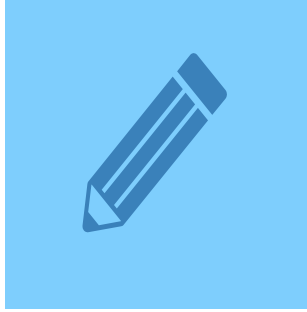


Implementasi ACF pada *Application event* di Bootstrap
Buat class Auth di @app/components/Auth.php

```
<?php
namespace app\components;
use Yii;
class Auth implements \yii\base\BootstrapInterface
{
    public function bootstrap($app)
    {
        $app->on(\yii\base\Application::EVENT_BEFORE_ACTION,
function ($event) {
            $action = $event->action;
            $moduleID = $action->controller->module->id;
            $controllerID = $action->controller->id;
            $actionID = $action->id;
            $user = \Yii::$app->user;
            $userID = $user->id;
            if(!in_array($controllerID,['default','site'])){
                $auth = \app\models\Auth::find()
                    ->where([
                        'module' => $moduleID,
                        'controller' => $controllerID,
                        'action' => $actionID,
                        'user_id' => $userID,
                    ])
                    ->count();
```

```
                if($auth==0) {
                    if (!$action instanceof \yii\web\ErrorAction) {
                        if ($user->getIsGuest()) {
                            $user->loginRequired();
                        } else {
                            throw new \yii\web\ForbiddenHttpException('Anda
tidak diizinkan untuk mengakses halaman ' . $action->id . ' ini!');
                        }
                    }
                }
            }
        });
    }
}
```

Access Control Filter



Kemudian pada *config* cukup menambahkan *class* yang tadi dibuat pada *aplication bootstrap*

```
<?php
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => [
            'log'
            'app\components\Auth',
    ],
    'components' =>[
```

Access Control Filter

Implementasi ACF pada *Application Behaviour* di Config
Buat class *behaviour* di @app/components/AccessControl.php

```
<?php
namespace app\components;
use Yii;
class AccessControl extends \yii\base\ActionFilter
{
    public function beforeAction($action)
    {
        $moduleID = $action->controller->module->id;
        $controllerID = $action->controller->id;
        $actionID = $action->id;
        $user = \Yii::$app->user;
        $userID = $user->id;
        if(!in_array($controllerID,['default','site'])){
            $auth = \app\models\Auth::find()
                ->where([
                    'module' => $moduleID,
                    'controller' => $controllerID,
                    'action' => $actionID,
                    'user_id' => $userID,
                ])
                ->count();
            if($auth==0) {
                if (!$action instanceof \yii\web\ErrorAction) {
                    if ($user->getIsGuest()) {
                        $user->loginRequired();
                    } else {
                        throw new \yii\web\ForbiddenHttpException('Anda
tidak diizinkan untuk mengakses halaman ' . $action->id . ' ini!');
                    }
                }
            }
        }
        return true;
    }
}
```

Access Control Filter

Kemudian pada *config* definisikan *behaviour* yang telah dibuat

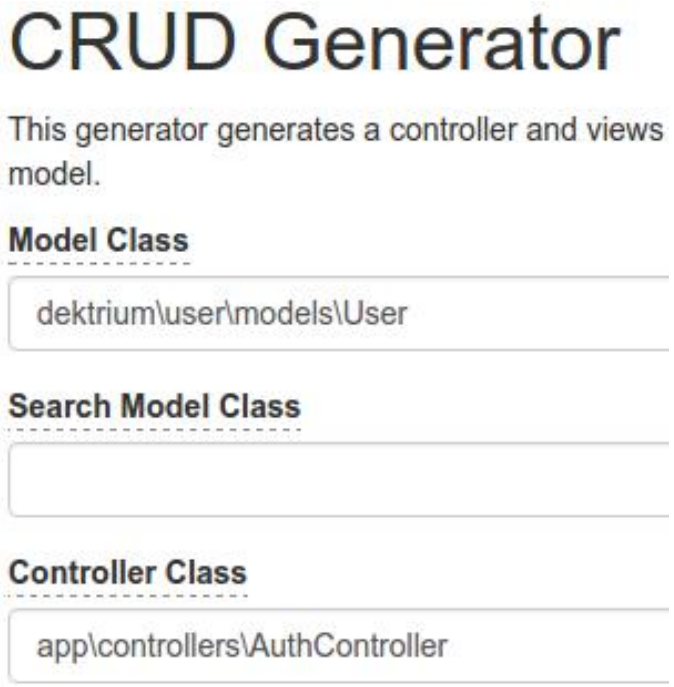
```
<?php
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'as access' => [
            'class' => 'app\components\AccessControl',
    ],
```

Access Control Filter

User Interface ACF

Menyediakan *user interface* (UI) dari ACF merupakan sebuah keharusan pada aplikasi yang menerapkan ACF.

Mari kita buat menggunakan *tools* Gii, CRUD *Generator*.



Access Control Filter

Klik tombol *Preview* lalu *Generate*

Preview Generate

Click on the above `Generate` button to generate the files selected below:

Create Unchanged Overwrite

Code File	Action	
controllers/AuthController.php	create	☑
views/auth/_form.php	create	☐
views/auth/create.php	create	☐
views/auth/index.php	create	☑
views/auth/update.php	create	☐
views/auth/view.php	create	☑

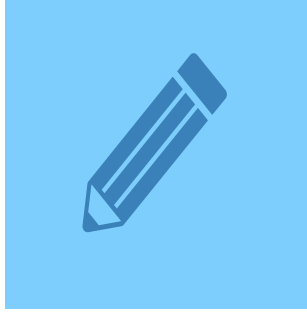
Access Control Filter

Modifikasi file @app/views/auth/index.php

```
<?php
use yii\helpers\Html;
use yii\grid\GridView;
$this->title = 'Authorization';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="auth-index">
    <h1><?= Html::encode($this->title) ?></h1>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],
            'username',
```

```
            [
                'header' => 'Auth',
                'format' => 'raw',
                'value' => function ($model){
                    return Html::a("<i class='glyphicon
glyphicon-lock'></i>",
                        ['view','id'=>$model->id],
                        ['class'=>'btn btn-danger btn-xs']
                    );
                }
            ]
        ],
    ]); ?>
</div>
```

Access Control Filter



Selanjutnya tambahkan fungsi getRoutes pada *controller* untuk mendapatkan semua *route* (*module, controller, action*) dari aplikasi kita.

```
public function getRoutes($app, $moduleID, $namespace, $user_id)
{
    $routes = [];
    $path = @Yii::getAlias('@' . str_replace('\\', '/', $namespace));
    foreach (scandir($path) as $file) {
        if (strcmp(substr($file, -14), 'Controller.php') === 0) {
            $controllerID =
\yii\helpers\Inflector::camel2id(substr(basename($file), 0, -14));
            $className = $namespace .
\yii\helpers\Inflector::id2camel($controllerID) . 'Controller';
            $controller = Yii::createObject($className, [$controllerID, $app]);
            $controllerID = $controller->uniqueId;
            foreach ($controller->actions() as $actionID => $value) {
                $auth = \app\models\Auth::find()->where([
                    'module'=>$moduleID,
                    'controller'=>$controllerID,
                    'action'=>$actionID,
                    'user_id'=>$user_id,
                ])->count();
                $routes[] = [
                    'module'=>$moduleID,
                    'controller'=>$controllerID,
                    'action'=>$actionID,
                    'auth'=>$auth,
                ];
            }
```

```
            $class = new \ReflectionClass($controller);
            foreach ($class->getMethods() as $method) {
                $name = $method->getName();
                if ($method->isPublic() && !$method->isStatic() && strpos($name,
'action') === 0 && $name !== 'actions') {
                    $actionID = \yii\helpers\Inflector::camel2id(substr($name, 6));
                    $auth = \app\models\Auth::find()->where([
                        'module'=>$moduleID,
                        'controller'=>$controllerID,
                        'action'=>$actionID,
                        'user_id'=>$user_id,
                    ])->count();
                    $routes[] = [
                        'module'=>$moduleID,
                        'controller'=>$controllerID,
                        'action'=>$actionID,
                        'auth'=>$auth,
                    ];

                }
            }
        }
    }
    return $routes;
}
```

Access Control Filter

Modifikasi fungsi actionView

```
public function actionView($id)
{
    $app = \Yii::$app;
    $moduleID = $app->id;
    $namespace = trim($app->controllerNamespace, '\\') . '\\';
    $routes = $this->getRoutes($app, $moduleID, $namespace, $id);
    foreach ($app->getModules() as $moduleID => $child) {
        if (($module = $app->getModule($moduleID)) !== null) {
            $namespace = trim($module->controllerNamespace, '\\') . '\\';
            $routes = array_merge($routes, $this->getRoutes($module, $moduleID, $namespace,
$id));
        }
    }

    return $this->render('view', [
        'model' => $this->findModel($id),
        'routes' => $routes,
    ]);
}
```

Access Control Filter

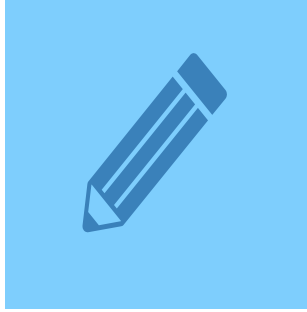
Modifikasi *view* yang akan menampilkan *routes* dan *checklist*

```
<?= DetailView::widget([
    'model' => $model,
    'attributes' => [
        'id',
        'username',
    ],
]) ?>

<table class="table table-striped">
    <tr>
        <th>Modules</th>
        <th>Controllers</th>
        <th>Actions</th>
        <th>Auth</th>
    </tr>
```

```
<?php
    foreach($routes as $row) {
        ?>
        <tr >
            <td><?= $row['module']?></td >
            <td><?= $row['controller']?></td >
            <td><?= $row['action']?></td >
            <td><?=
Html::checkbox('auth[]',$row['auth']); ?></td >
        </tr >
        <?php
    }
    ?>
</table>
```

Access Control Filter



Buat fitur untuk menentukan hak akses melalui *checklist*, lalu tambahkan javascript untuk melakukan *request* ke fungsi pemrosesan auth secara Ajax

```
<td><?= Html::checkbox('auth[]',$row['auth'],[
    'class' => 'processAuth',
    'data-module' => $row['module'],
    'data-controller' => $row['controller'],
    'data-action' => $row['action'],
]); ?></td>
```

```
<?php
$this->registerJs('
    $(".processAuth").on("click", function (e) {
        module = $(this).attr("data-module");
        controller = $(this).attr("data-controller");
        action = $(this).attr("data-action");
        user_id = '.$model->id.';
        checked = $(this).prop("checked");
        var link = "'.Url::to(['process-auth']).'?module="+module+
            "&controller="+controller+"&action="+action+"&user_id="+user_id+
            "&checked="+checked;
        $.get(link, function(data) {
            alert(data)
        });
    });
');
```

Access Control Filter

Selanjutnya buat fungsi action untuk memproses auth

```
use app\models\Auth;

public function actionProcessAuth($module,$controller,$action,$user_id,$checked)
{
    $params = [
        'module'=>$module,
        'controller'=>$controller,
        'action'=>$action,
        'user_id'=>$user_id,
    ];
    $auth = Auth::find()->where($params)->count();
    if($checked){
        if($auth==0){
            $model = new Auth($params);
            $model->save();
        }
        return "success inserted";
    }
    else{
        if($auth>0) {
            Auth::find()->where($params)->delete();
        }
        return "success deleted";
    }
}
```

Role Based Access Control

RBAC adalah mekanisme pengaturan hak akses berdasarkan *role* pada aplikasi. Mudahnya *role* bisa diartikan sebagai *group* dari user meskipun kenyataanya *role* lebih tepat diartikan sebagai *group* dari hak akses. Contoh role: *admin, member, author, dst*.

Apa perbedaanya dengan ACF?

Perbedaan mendasar antara RBAC dan ACF meskipun keduanya sama-sama *access control,* yaitu RBAC menggunakan konsep *role* dalam pengaturan hak aksesnya sedangkan ACF hanya menggunakan konsep *on off* saja.

Role Based Access Control

- Apa kelebihannya dibandingkan dengan ACF?

RBAC memungkinkan dan memudahkan kita membuat hak akses *user* secara bertingkat.

- Konfigurasi

Yii mempunyai dua jenis *class* untuk mengontrol *authorization,* yaitu yii\rbac\PhpManager dan yii\rbac\DbManager

Jika kita menggunakan jenis class PhpManager, maka Yii akan menggunakan file PHP biasa untuk menyimpan data *authorization*, ini cocok untuk RBAC yang sederhana. Sedangkan *class* DbManager, menyimpan data *authorization*-nya menggunakan *database*.

Role Based Access Control

PhpManager

Konfigurasi dengan mendefinisikannya pada *component* authManager

```
$config = [
    ...
    'components' => [
            ...
            'authManager' => [
                    'class'=> 'yii\rbac\PhpManager',
            ],
            ...
    ],
    ...
];
```

Role Based Access Control

DbManager

Konfigurasi dengan mendefinisikannya pada *component* authManager

```
$config = [
    ...
    'components' => [
            ...
            'authManager' => [
                    'class'=> 'yii\rbac\DbManager',
            ],
            ...
    ],
    ...
];
```

Role Based Access Control

Berbeda dengan PhpManager, DbManager menyimpan data RBAC pada *database* dengan menggunakan empat tabel, yaitu :

- itemTable
- itemChildTable
- assignmentTable
- ruleTable

Yii telah menyiapkan *database migration* untuk keempat tabel tersebut.

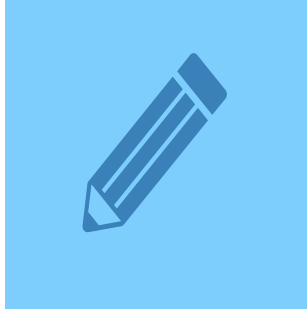
```
yii migrate --migrationPath=@yii/rbac/migrations
```

Role Based Access Control

Menggunakan *Extension* RBAC

Untuk memudahkan kita mengimplementasikan RBAC maka kita bisa menggunakan *extension* RBAC Manager.

mdmsoft/yii2-admin



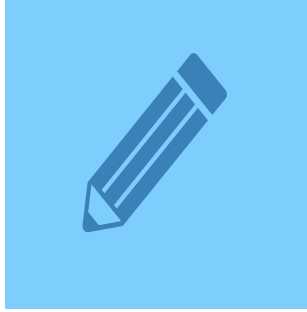
Extension ini mengeksplorasi semua fitur RBAC yang dimiliki Yii. Untuk kasus pengaturan hak akses yang kompleks maka *extension* ini sangat cocok.

```
https://github.com/mdmsoft/yii2-admin
```

Instalasi

```
composer require mdmsoft/yii2-admin "~2.0"
```

hscstudio/yii2-mimin



Extension ini versi mini atau *simplify* dari yii2-admin. Tidak semua fitur RBAC bisa digunakan pada *extension* ini.

https://github.com/hscstudio/yii2-mimin

Instalasi

```
composer require --prefer-dist hscstudio/yii2-mimin "~1.1"
```

Thanks!

Any questions?

You can find me at:
@edoriansyah
edo@nurulfikri.co.id

Credits

Special thanks to :

- Presentation template by SlidesCarnival
- "Membangun Aplikasi Profesional Berbasis Web Menggunakan Yii Framework" Book by Hafid Mukhlasin