# RBAC คืออะไร? พร้อมสอนวิธีใช้ RBAC DB

AUGUST 20, 2015

ในการพัฒนา Web Application เราจำเป็นต้องมีการควบคุมการเข้าใช้งานในส่วนต่างๆ ของ application ถ้าหาก application ของเรามีเพียงแค่การ login และ logout ผมคิดว่าคงไม่พอ เพราะในการใช้งานปกติ ผู้ใช้งานแต่ละคนอาจจะมีสิทธิ์ในการเข้าใช้งานที่แตกต่างกัน อย่างน้อยก็เพื่อเป็นการปกป้องข้อมูลหรือการใช้งานในแต่ละส่วนให้ถูกต้องเหมาะสม ถ้าหากระบบที่เราได้พัฒนาขึ้นแล้ว แต่ยังไม่มีการจัดการ RBAC ผมคิดว่าคุณควรต้องคิดใหม่..เพราะตรงนี้ Yii2 ได้เตรียมไว้ให้เราครบ พร้อมเสริฟ รอเพียงแค่กดน้ำร้อน ปรุงรสตามใจชอบ เท่านั้นเอง ดี้ดี ^ ^

เนื้อหาในเรื่องนี้ผมจะอธิบายตั้งแต่การออกแบบ RBAC ไปจนถึงตัวอย่างการใช้งานกับโปรเจ็คง่ายๆ เพื่อให้เราสามารถออกแบบและใช้งาน RBAC กับ Application ที่เราพัฒนาได้อย่างเหมาะสม

## เนื้อหามีอะไรบ้าง

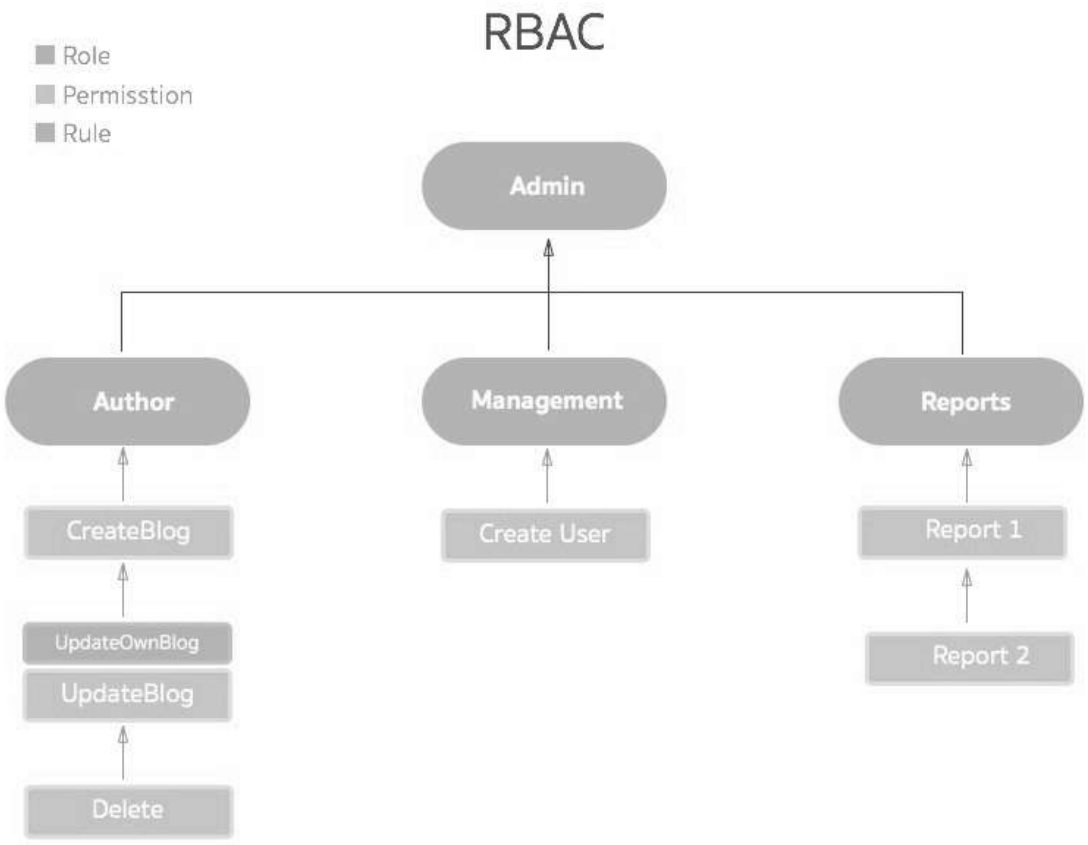
- การใช้งาน RBAC DB แบบเก็บข้อมูลตาม url

# RBAC คืออะไร?

RBAC คือ การจัดการสิทธิ์การเข้าใช้งานหรือที่เรียกว่า Role Based Access Control (RBAC) ซึ่งเป็นการควบคุมการเข้าใช้งานในส่วนต่างๆ ของ application ที่เราได้สร้างขึ้น โดยเมื่อมีการเรียกใช้งานในแต่ละส่วน RBAC จะคอยเช็คผู้ใช้งานแต่ละคนว่า มีสิทธิ์ที่จะเข้าใช้งานในส่วนนี้หรือไม่ หากมีก็จะยอมให้เข้าใช้งาน แต่ถ้าหากไม่มีก็จะทำการแสดง error forbidden ออกไป เพื่อให้ผู้ใช้งานทราบว่าไม่สามารถเข้าใช้งานในส่วนนี้ได้

# ส่วนประกอบของ RBAC

RBAC ประกับไปด้วย Role,Permission,Rule ลองดูตามภาพ

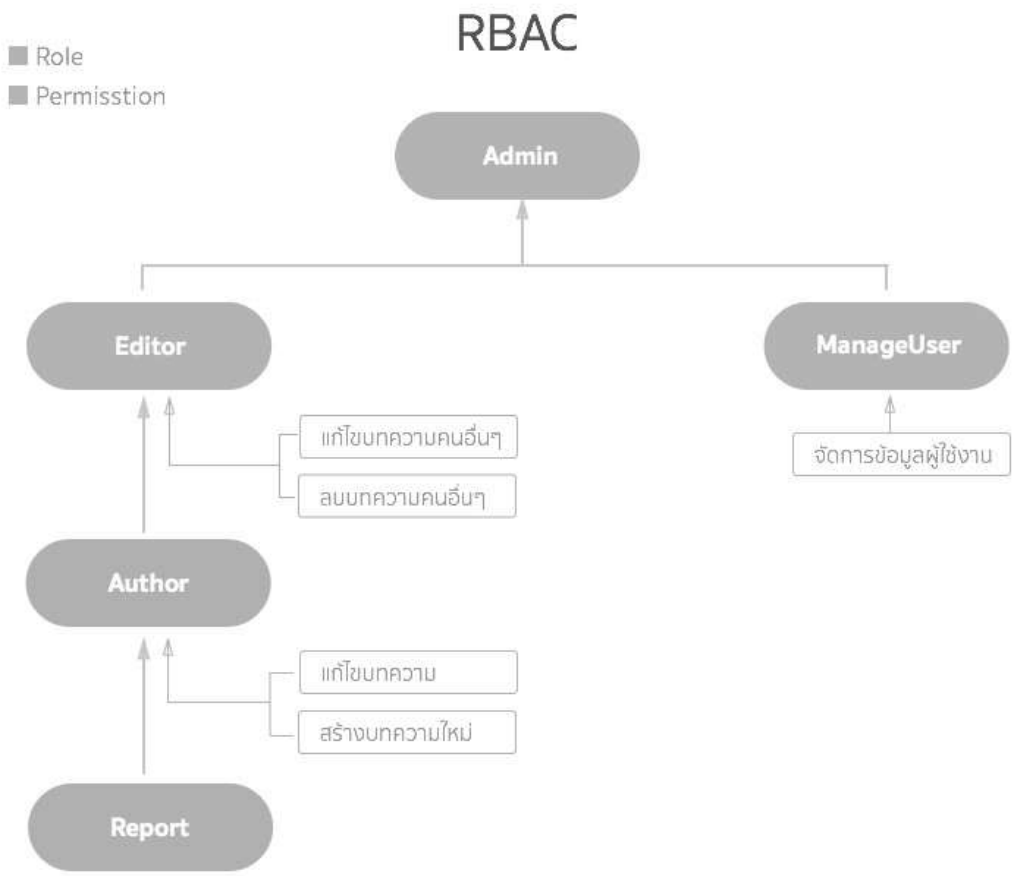



ส่วนประกอบหลักๆ ของ rbac ใน yii ที่เราสามารถสร้างได้จะประกอบไปด้วย 3 ส่วน คือ

- Permission ก็เปรียบเสมือนใบอนุญาติที่บอกว่าเราสามารถทำอะไรได้บ้าง
- Role เป็นกลุ่มของ Permission หลายๆ ตัว ที่เกี่ยวข้องกัน รวมตัวกัน เพื่อให้สามารถเรียกใช้งานได้สะดวก ไม่ต้องเรียกใช้งานที่ Permission ทีละอัน
- Rule คือตัวช่วยที่ทำให้ Permission หรือ Role ธรรมดาสามารถเขียนฟังก์ชันเพิ่มเติมเพื่อตรวจสอบค่าบางอย่างได้

# ตัวอย่าง RBAC แบบต่างๆ

โดยปกติแล้ว ผู้ใช้งานแต่ละคนจะมีสิทธ์ในการเข้าใช้งานแตกต่างกัน เพราะฉะนั้นเราจึงจำเป็นต้องแยก role ออกเป็นกลุ่มๆ โดยให้แต่ละกลุ่มมีสิทธิ์ในการเข้าใช้งานที่ไม่เหมือนกัน เมื่อผู้ใช้งานได้รับ role ใหนก็จะสามารถเข้าใช้งานได้ตามสิทธิ์ที่ role นั้นมี ทำให้เราสามารถควบคุมการเข้าใช้งานในส่วนต่างๆ ได้อย่างเหมาะสมและยืดยุ่น




ถ้าดูจากภาพด้านบนจะเห็นว่ามี Role ทั้งหมด 5 ตัว ซึ่งแต่ละตัวมีความสามารถที่แตกต่างกัน ภายใน Role แต่ละตัวอาจมี Role หรือ Permission อยู่ภายใต้สังกัดได้ ส่วน Permission จะไม่อยู่ภายใต้ Role ก็ได้ แต่ในความเป็นจริง เมื่อใช้งานมันอาจจะไม่สะดวก เพราะหากมี Permission ทั้งหมด 10 ตัวแล้วเราต้องการใช้ User ใช้งานได้เราต้อง add Permission ทั้งหมด 10 ตัวให้ user คนนั้น แต่ถ้าหากเราย้ายให้มันสังกัดที่ Role แล้วเวลาให้สิทธิ์เราก็ให้ Role แค่ตัวเดียว ผู้ใช้งานก็จะได้สิทธิ์การเข้าใช้งานทั้ง 10 ตัว ซึ่งจะสะดวกและยืดยุ่นกว่ามากๆ

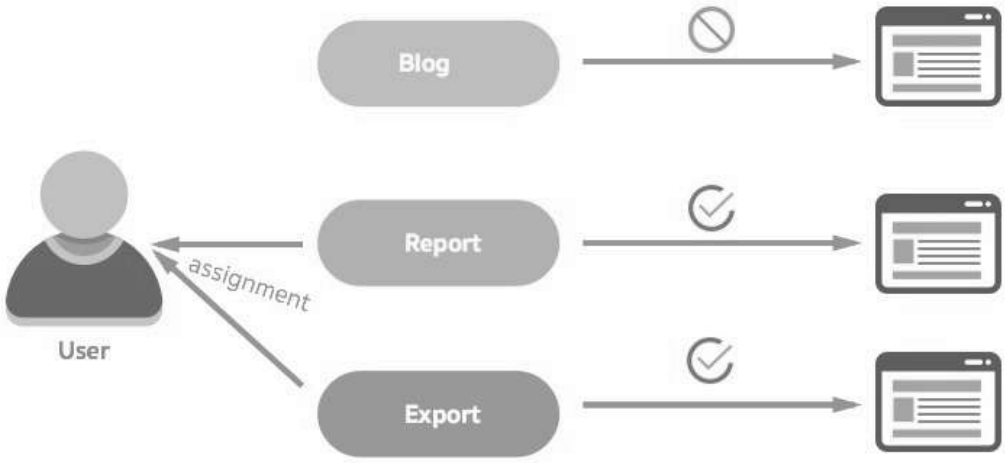
เพื่อให้สามารถเข้าใจได้ง่ายๆ ผมจะยกตัวอย่าง โดยใช้ Role ที่กำหนดขึ้นมาใหม่ 3 ตัว คือ

- Blog การใช้งาน blog ทั้งหมดเช่น create,update,delete
- Report การดูรายงานต่างๆ
- Export การ export ข้อมูลในรูปแบบต่างๆ

**ตัวอย่างที่ 1**

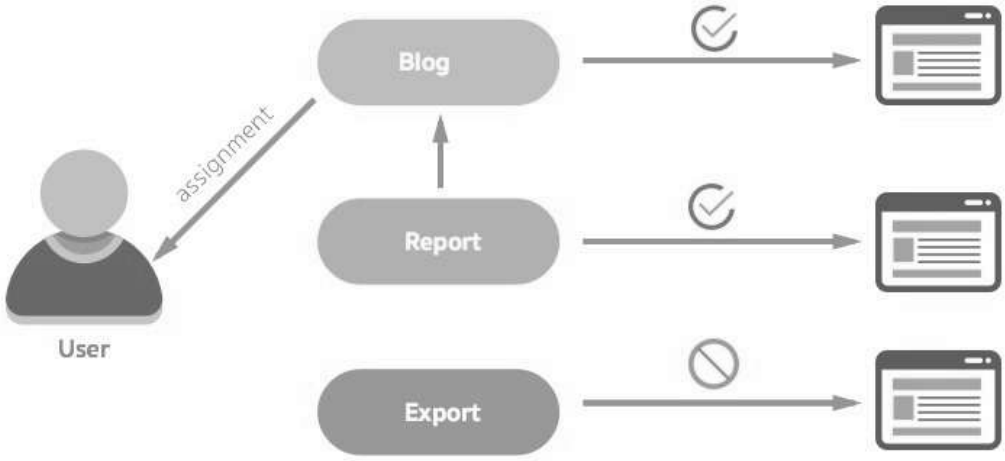

เมื่อเราดูตามภาพแล้วจะเห็นว่า มีการมอบ role `Blog` ให้กับ User เมื่อ User ได้รับก็จะสามารถเข้าใช้งานในส่วนของ blog ได้แต่ในส่วนของ role `Report` และ `Export` จะไม่สามารถเข้าใช้งานได้เพราะ user ไม่ได้รับ role นั้น

**ตัวอย่างที่ 2**



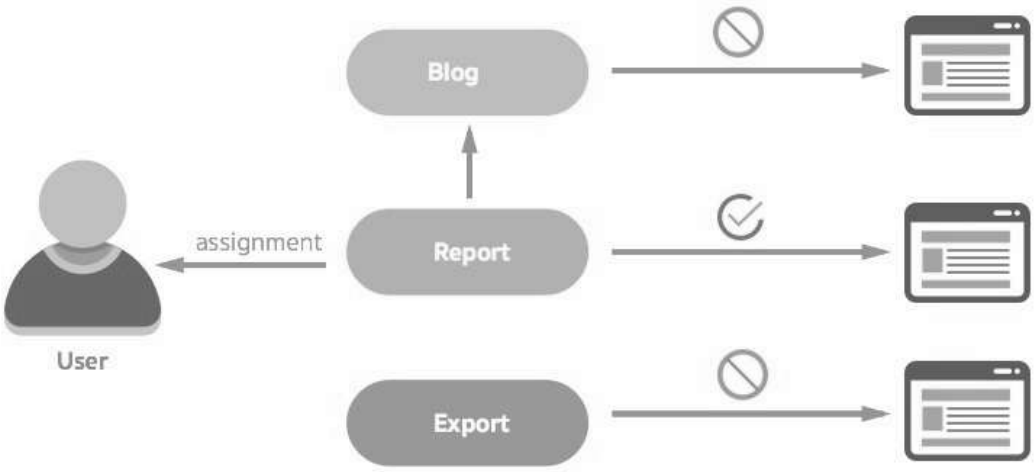

User ได้รับมอบ Role `Report` และ `Export` เพราะฉะนั้นก็จะเข้าใช้งาน report,export ได้ ยกเว้น blog เพราไม่ได้รับ role `Blog` มา

**ตัวอย่างที่ 3**



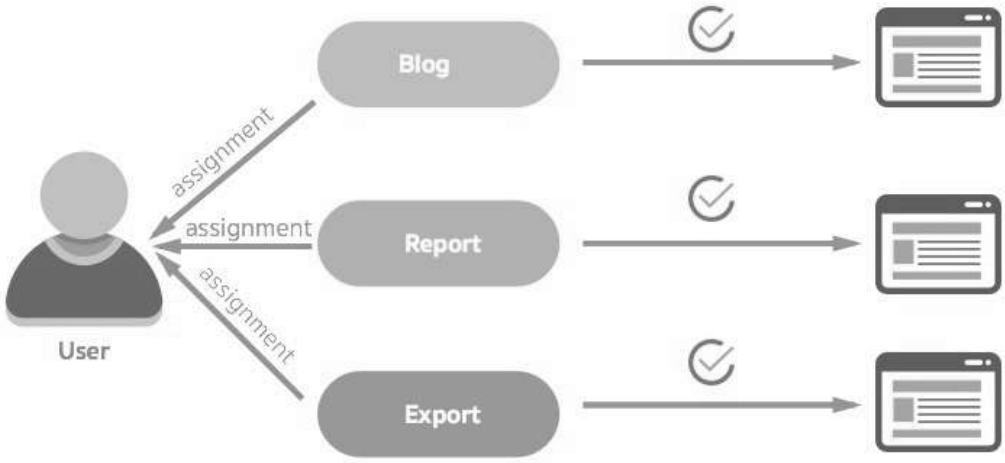

สังเกตว่า User จะได้รับมอบในส่วนของ Role `Blog` เพียงอย่างเดียว แต่ User สามารถเข้าใช้งานทั้ง `Blog` และ `report` ได้ เพราะว่า report ในอยู่ภายใต้ Role `Blog` เพราะฉะนั้นใครก็ตามที่ได้รับ role `Blog` ก็จะสามารถใช้งาน reprot ได้ด้วย

**ตัวอย่างที่ 4**



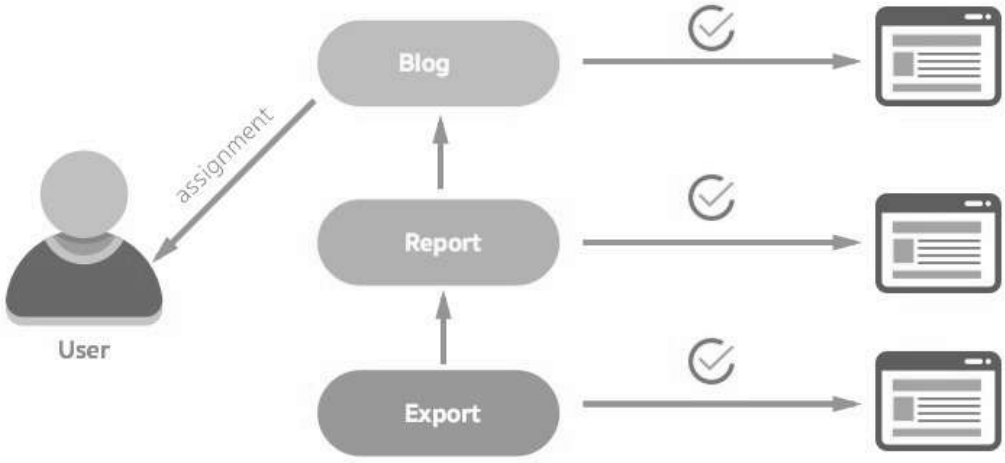

User ได้รับมอบ role `Report` ก็จะสามารถเข้าใช้งาน report ได้เพียงอย่างเดียว แม้ว่า role `Report` จะอยู่ภายใต้ `Blog` ก็ตาม แต่ถ้าหามี role อะไรก็ตามอยู่ภายใต้ role `Report` มันก็จะสามารถเข้าใช้งานได้ตามไปด้วย

**ตัวอย่างที่ 5**




จากภาพด้านบนจะเห็นว่า หากเราต้องการให้ผู้ใช้งานสามารถเข้าใช้งานได้ทั้งหมด ต้อง assignment role ทั้ง 3 ตัว ให้กับผู้ใช้งานแต่ละคน คนละ 3 role มีกี่คนก็คูน 3 เข้าไปซึ่งมันจะเยอะมาก จึงไม่ค่อยสะดวกเท่าไหร่

```
+-----------+---------+
| item_name | user_id |
+-----------+---------+
| Blog      | 1       |
| Report    | 1       |
| Export    | 1       |
| Blog      | 2       |
| Report    | 2       |
| Export    | 2       |
| Blog      | 3       |
| Report    | 3       |
| Export    | 3       |
+-----------+---------+
```




แต่ถ้าเราปรับใหม่โดยให้ role ที่เกี่ยวข้องกันจัดเป็นลำดับชั้นไว้ตามภาพด้านบน หรือจะสร้าง Role อีกตัวขึ้นมาเพื่อให้ role อื่นๆ ไปอยู่ภายใต้ก็ได้ ตัวอย่างนี้จะให้อยู่ภายใต้ blog โดยจะเรียงลำดับตามนี้ Blog->Report->Export โดยให้ `Blog` เป็น Role

สูงสุด ถ้าหากอยากให้ผู้ใช้งานสามารถเข้าใช้งานได้ทั้งหมด 3 role ก็แค่มอบ role Blog ให้กับผู้ใช้งานหรือ User นั้นๆ เมื่อดูข้อมูล

```
+-----------+---------+
| item_name | user_id |
+-----------+---------+
| Blog      | 1       |
| Blog      | 2       |
| Blog      | 3       |
+-----------+---------+
```

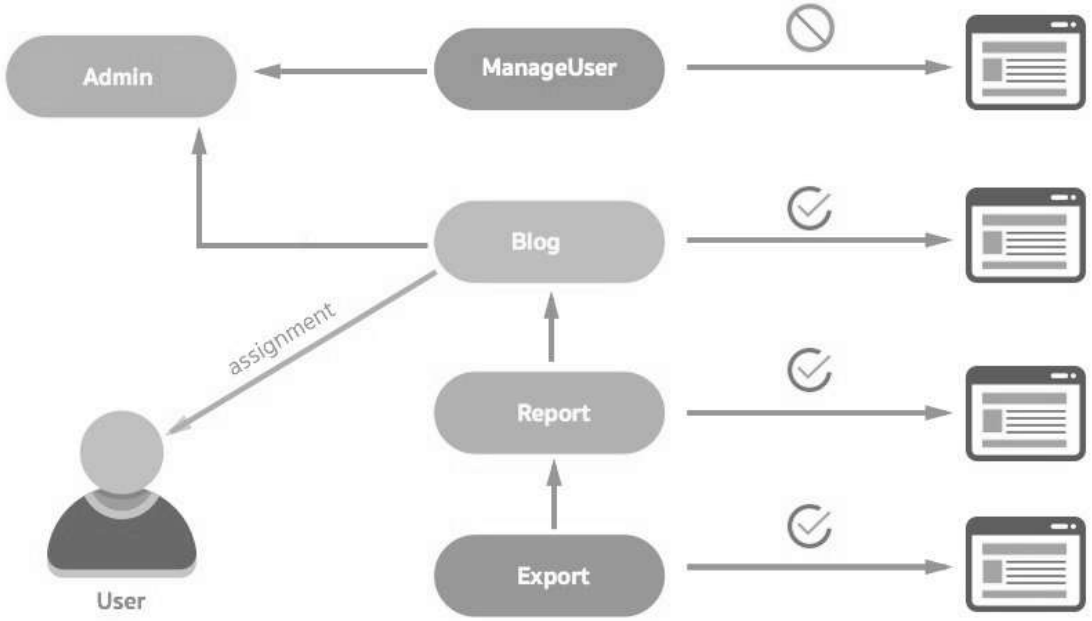
ผู้ใช้งานหรือ User ก็จะสามารถเข้าใช้งานได้ทั้ง 3 ส่วนคือ Blog,Report,Export

> ตารางผมแค่ทำให้มองเห็นภาพว่าเวลาที่มันเก็บข้อมูลมันเก็บยังไง

**ตัวอย่างที่ 6**

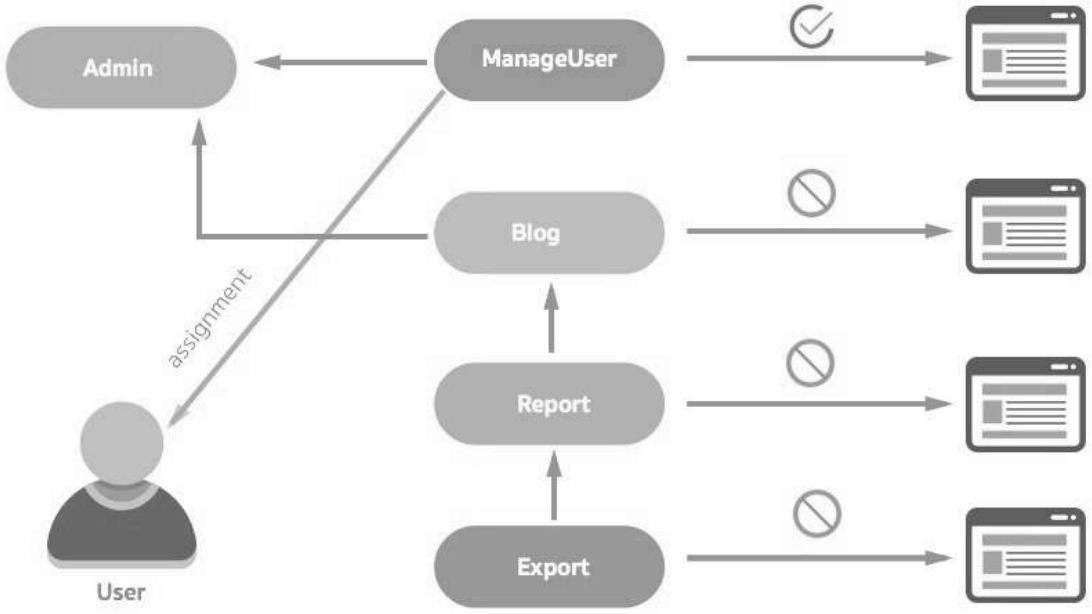

ตัวอย่างนี้ได้เพิ่ม role `Admin` ขึ้นมาเพื่อเป็นตัวควบคุมสูงสุด เพื่อที่จะทำให้เราสามารถมอบ role ต่างๆ ได้อิสระมากขึ้น สังเกตุว่า `Admin` จะมี role ที่เอาไว้จัดการข้อมูลผู้ใช้งานคือ role `ManageUser` เพื่อให้ `Admin` สามารถจัดการข้อมูลผู้ใช้ได้ จึงให้ role `ManageUser` ไปอยู่ภายใต้ role `Admin`

**ตัวอย่างที่ 7**



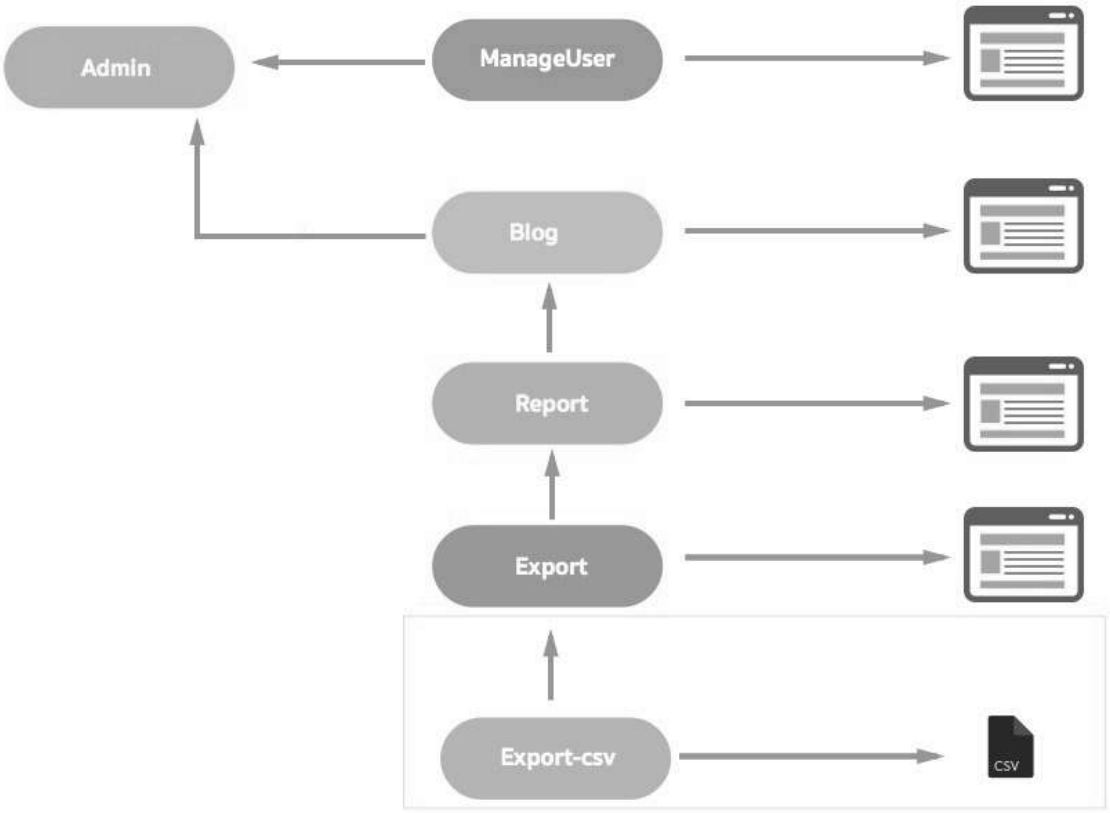

ถ้าเรามอบ role `Blog` ให้กับ user แล้ว user ก็ใช้งานได้แค่ role `Blog,Report,Export`

**ตัวอย่างที่ 8**



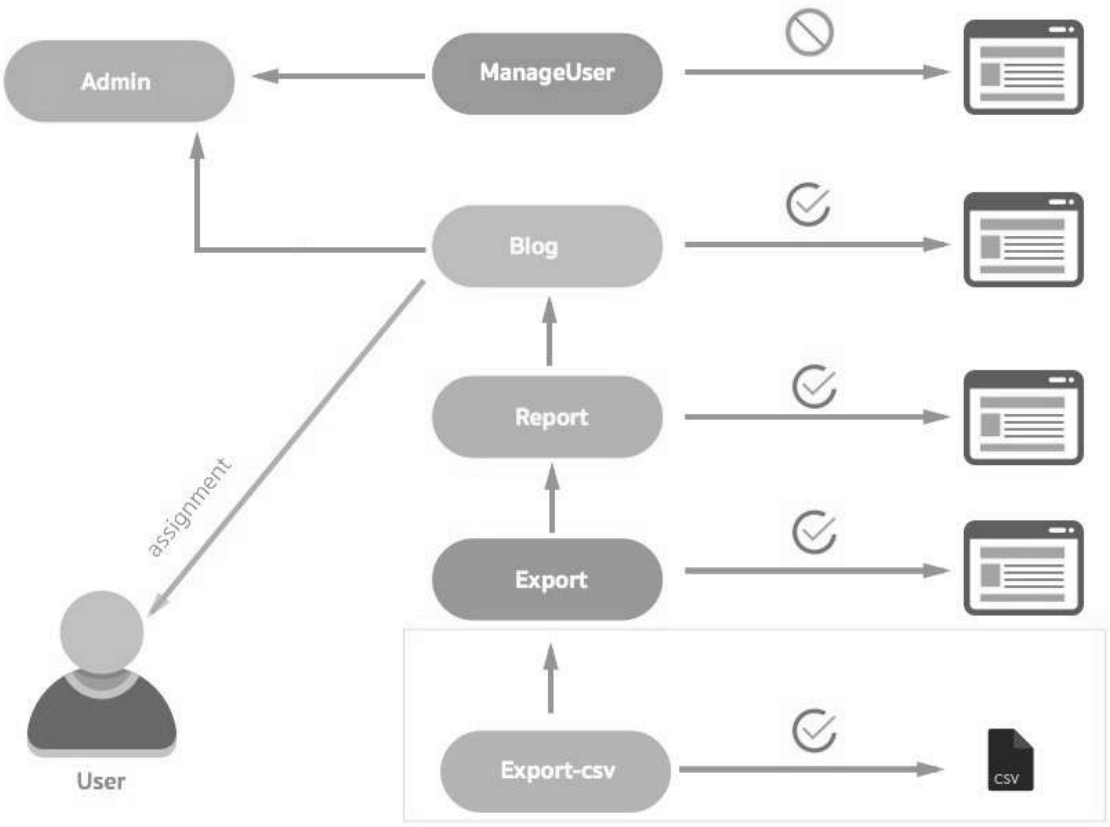

ถ้าหาก user ได้รับมอบ role `ManageUser` แล้วก็จะสามารถใช้งานได้แค่ส่วนจัดการข้อมูลผู้ใช้งานเท่านั้น

**ตัวอย่างที่ 9**




ถ้าหากต่อไปมีการเพิ่ม role ใหม่เข้ามาชื่อ `Export-csv` ใครก็ตามที่อยู่บน role Export-csv ก็จะสามารถใช้งานได้ทันทีโดยไม่ต้องแก้ไขข้อมูลการ assignment ใหม่เลย อย่างเช่น ถ้าได้รับ role `Blog` ก็จะสามารถใช้งาน Blog->Report->Export->Export-csv ได้เลย เพราะ Export-csv อยู่ภายใต้ Blog อยู่แล้ว

**ตัวอย่างที่ 10**




ถ้า User ได้รับ role `Blog` ก็จะสามารถใช้งาน Blog->Report->Export->Export-csv ได้เลย เพราะ Export-csv อยู่ภายใต้ Blog อยู่แล้ว

ตัวอย่างทั้งหมดคิดว่าคงพอทำให้เราเข้าใจและสามารถออกแบบ rbac ที่เหมาะสมกับงานของตัวเองได้ ต่อไปเราจะทำการลงมือสร้าง rbac จริงๆ เพื่อใช้งานกับ application ของเรา

# ออกแบบ RBAC

หลักๆ ในการสร้าง RBAC นั้น เราจะต้องทำการวิเคราะห์ว่า application ของเรา มีการทำงานหลักๆ อะไรบ้าง ในแต่ละส่วนต้องทำอะไรบ้าง เพื่อให้สามารถสร้าง rbac ได้อย่างง่ายๆ ผมจะยกตัวอย่างการสร้างระบบ Blog ตามตัวอย่าง doc-2.0 ได้เขียนอธิบายไว้ เพราะเข้าใจง่าย

การออกแบบจะใช้ตัวอย่างการสร้าง Blog แบบง่ายๆ ไม่ซับซ้อนเพื่อให้เราสามารถทำความเข้าใจได้อย่างรวดเร็ว ก่อนอื่นเราจะต้องแบ่งกลุ่มผู้ใช้งานว่ามีอะไรบ้างหลักๆ แยกออกมา หลังจากนั้นให้เราเขียนความสามารถของระบบว่าทำอะไรได้บ้างหลักๆ ออกเป็นข้อๆ ดังนี้

## กลุ่มผู้ใช้งาน

ระบบแบ่งผู้ใช้งานเป็น 3 กลุ่มใหญ่ๆ คือ

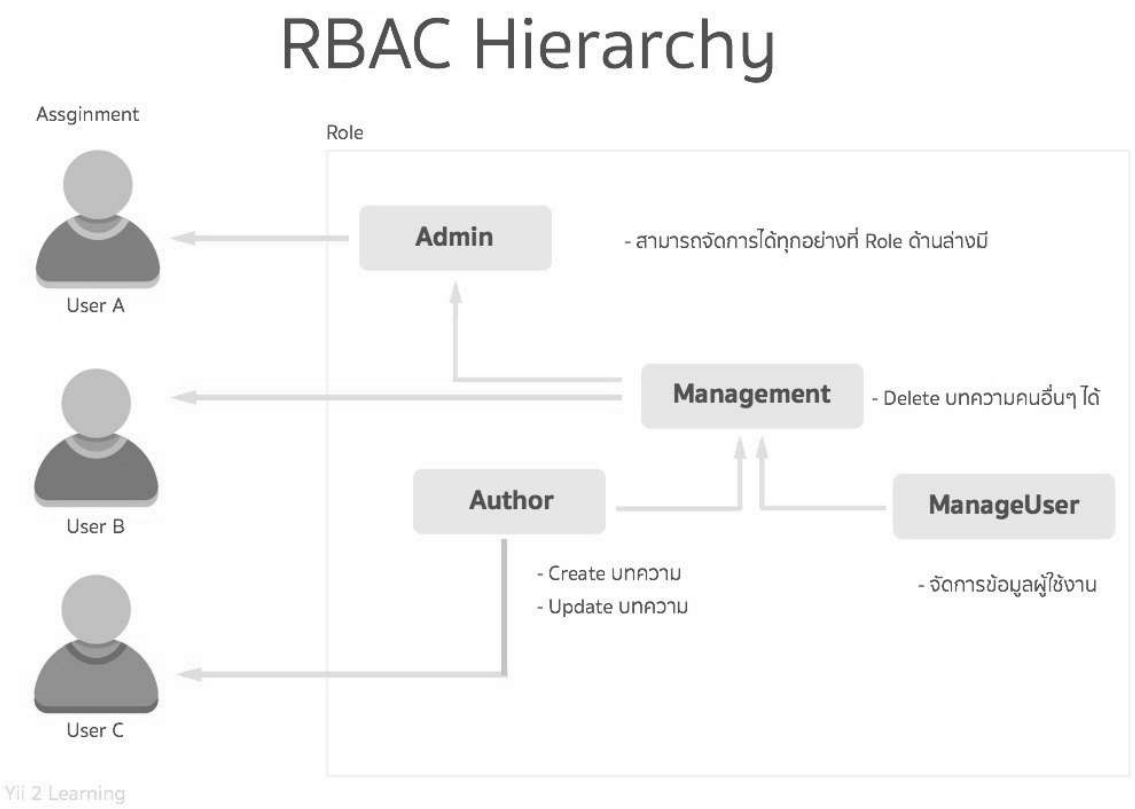
- Author ผู้ใช้งานที่เขียนบทความ
- ManageUser จัดการข้อมูลผู้ใช้งาน
- Management จัดการข้อมูลบทความ
- Admin ผู้ดูแลระบบ

ซึ่งในส่วนนี้เราจะเรียกมันว่า Role

## สิทธิ์การเข้าใช้งาน (Permission)

- สามารถ create บทความได้
- สามารถ update บทความได้
- สามารถ delete บทความได้
- สามารถ จัดการข้อมูลผู้ใช้งาน

เมื่อเราทำการวิเคราะห์และออกแบบระบบหลักๆ ได้แล้วเราจะต้องทำการให้สิทธิ์ แก่ Role หรือกลุ่มผู้ใช้งานที่เราได้ทำการออกแบบไว้ว่า กลุ่มใหนสามารถทำอะไรได้บ้าง ดังนี้




- Author
    - create บทความ
    - update บทความ

- Management
    - delete บทความ
- ManageUser
    - สามารถจัดการข้อมูลผู้ใช้งานได้
- Admin
    - เป็นผู้ดูแลระบบ

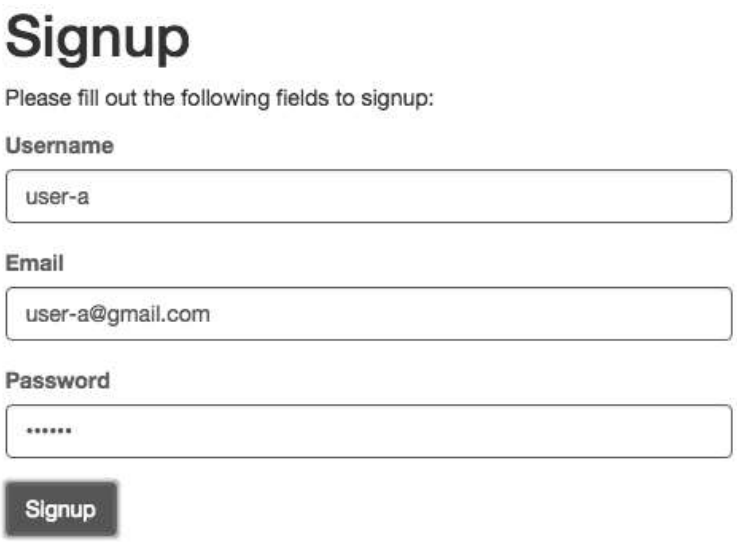
เมื่อเราออกแบบและเข้าใจโครงสร้างระบบโดยรวมแล้ว จะทำให้เราสามารถสร้าง RBAC ได้ง่าย และไม่สับสน

# เตรียม Project และเปิดใช้งาน RBAC

ก่อนอื่นเราจะต้องเตรียม project ให้พร้อม โดยตัวอย่างนี้เราจะใช้ Advanced Project Template เพราะมีระบบ login มาให้อยู่แล้ว ให้ทำการสร้าง Advanced Project Template ดูวิธีการสร้างได้ที่นี่ หลังจากติดตั้งเสร็จแล้วให้ทำตามขั้นตอนต่อไปนี้

- php yii/init
- สร้างฐานข้อมูลไว้ ชื่อ yii2-workshop-rbac-db
- ทำการ config ฐานข้อมูลที่ common/config/main-local.php ใส่ชื่อฐานข้อมูลและ username & password เปลี่ยน localhost เป็น 127.0.0.1 ให้ถูกต้อง
- ทำการนำเข้าฐานข้อมูล user โดยใช้คำสั่ง `php yii migrate`
- ลงทะเบียนผู้ใช้งาน โดยกำหนด user,password,email เพื่อใช้สำหรับเข้าทดสอบการใช้งาน rbac ที่เราจะสร้างขึ้น มีข้อมูล username,email,password ดังนี้
    - user-a, user-a@gmail.com, 123456
    - user-b, user-b@gmail.com, 123456
    - user-c, user-c@gmail.com, 123456
    - user-d, user-d@gmail.com, 123456

ไปที่ frontend และทำการสมัครสมาชิกที่หน้า `site/signup` ด้วย username,email, password ที่กำหนดให้ในด้านบน




หลังจากลงทะเบียนเสร็จเรียบร้อยให้ทำการทดสอบ login ว่าสามารถใช้งานได้จริงหรือไม่ เมื่อสมัครสมาชิกเสร็จ ให้เราทำการตรวจสอบข้อมูลที่ตาราง user จะพบรายการดังนี้

```
mysql> select username,email from user;
+----------+------------------+
| username | email            |
+----------+------------------+
| user-a   | user-a@gmail.com |
| user-b   | user-b@gmail.com |
| user-c   | user-c@gmail.com |
| user-d   | user-d@gmail.com |
+----------+------------------+
3 rows in set (0.00 sec)
```

# เปิดการใช้งาน RBAC

Yii 2 มี RBAC ให้เลือกใช้งาน 2 แบบ คือ `yii\rbac\PhpManager` and `yii\rbac\DbManager` ทั้ง 2 ตัวนี้การทำงานหลักๆ ไม่ต่างกัน ต่างกันที่การเก็บข้อมูลเท่านั้นแล้วแต่จะเลือกใช้งาน ตามตัวอย่างวันนี้จะเลือกใช้งาน `yii\rbac\DbManager` เพราะสามารถจัดการได้ง่าย สามารถดูข้อมูลได้ที่ database ของเรา

## เปิดใช้งาน yii\rbac\DbManager

เปิดไฟล์ common/config/main.php เพิ่ม `authManager` เข้าไป

```
return [
    // ...
    'components' => [
        'authManager' => [
            'class' => 'yii\rbac\DbManager',
```

```
            ],
            // ...
        ],
    ];
```

เนื่องจาก `DbManager` เก็บข้อมูลที่ db ดังนั้นเราจะต้องนำเข้าโครงสร้างตารางที่ yii เตรียมไว้ให้แล้วโดยใช้ migration ให้เปิด command line แล้วเข้าไปที่ root โปรเจ็คของเราจากนั้นทำการรันคำสั่ง

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

ก็จะพบข้อความ

```
Yii Migration Tool (based on Yii v2.0.6)

Total 1 new migration to be applied:
m140506_102106_rbac_init

Apply the above migration? (yes|no) [no]:yes
*** applying m140506_102106_rbac_init
  > create table  ... done (time: 0.050s)
  > create table  ... done (time: 0.019s)
  > create index idx-auth_item-type on  (type) ... done (time: 0.025s)
  > create table  ... done (time: 0.027s)
  > create table  ... done (time: 0.024s)
*** applied m140506_102106_rbac_init (time: 0.174s)


Migrated up successfully.
```

และทำการเช็คดูในฐานข้อมูลของเราจะพบว่ามีตารางใหม่ขึ้นมา 4 ตาราง ดังนี้

```
mysql> show tables;
+---------------------------------+
| Tables_in_yii2-workshop-rbac-db |
+---------------------------------+
| auth_assignment                 |
| auth_item                       |
| auth_item_child                 |
| auth_rule                       |
| migration                       |
| user                            |
+---------------------------------+
6 rows in set (0.01 sec)
```

ตารางของ RBAC จะมี prefix นำหน้าว่า `auth_` เสมอ

> ถ้าหากพบ error อาจต้องตรวจสอบคอนฟิกในส่วนฐานข้อมูล ว่าถูกต้องหรือไม่ หากยังไม่ได้ให้ลองเปลี่ยน localhost เป็น 127.0.0.1 ดู

เมื่อ migration เสร็จเรียบร้อย ให้เราตรวจสอบฐานข้อมูลจะพบว่ามีตารางใหม่ขึ้นมา 4 ตารางคือ

- *itemTable* คือข้อมูลรายการ role และ permission ทั้งหมดซึ่งจะเก็บไว้ที่ตาราง `auth_item`
- *itemChildTable* เป็นข้อมูลที่บอกว่า role ตัวใหนอยู่ภายใต้ role อะไรบ้างซึ่งเก็บไว้ที่ตาราง `auth_item_child`
- *assignmentTable* เป็นตารางที่ระบุว่า user แต่ละคนมี role เป็นอะไรหรือมีสิทธิ์ทำอะไรบ้าง ตรงนี้จะเป็นตัวระบุ เก็บไว้ที่ตาราง `auth_assignment`
- *ruleTable* : เป็นตารางที่เก็บข้อมูลให้กับ Rule ต่างๆที่สร้างขึ้น เก็บไว้ที่ตาราง `auth_rule`

เมื่อตั้งค่าเปิดการใช้งาน authManager และเตรียมฐานข้อมูลเสร็จเรียบร้อย เราก็จะสามารถเรียกใช้งาน `authManager` ผ่าน Yii::$app ได้ ดังนี้

```
$auth = Yii::$app->authManager;
```

สามารถดูรายละเอียดเพิ่มเติมว่า DbManager ทำอะไรได้บ้าง ได้ที่นี่

# การสร้าง RBAC

หลังจากที่เราออกแบบและเปิดการใช้งาน authManager ไว้แล้ว ขั้นตอนต่อไปคือเราจะทำการสร้าง RbacController ซึ่งเป็น Controller แบบ console คือต้องใช้ผ่าน command line เท่านั้น ที่ต้องทำผ่าน command line เพราะปกติการสร้าง rbac เราก็จะทำแค่ครั้งเดียว ยกเว้นว่ามีการแก้ไข role ต่างๆ

เราจะทำการสร้างไฟล์ RbacController ไว้ที่ console/controllers/RbacController.php แล้วสร้างโค้ดตามนี้

```
<?php
namespace console\controllers;

use Yii;
use yii\helpers\Console;

class RbacController extends \yii\console\Controller {

  public function actionInit(){
     Console::output('Yii 2 Learning.');
  }
```

```
}
?>
```

เราสร้าง `RbacController` ขึ้นมาเพื่อใช้ในการสร้าง Rbac หากสังเกตจะพบว่า controller extends ด้วย yii\console\Controller ซึ่งจะไม่ใช่ controller ปกติที่เราใช้งานกัน ตอนนี้ RbacController มี 1 action คือ Init โดยให้ echo ข้อความออกมาเป็น "Yii 2 Learning"

ทดลองเรียกใช้งาน โดยเปิด command แล้ว cd เข้าไปที่โปรเจคของเรา จากนั้นรันคำสั่ง

```
php yii
```

จะพบกับข้อความประมาณนี้ ให้สังเกตุด้านล่างจะมี rbac/init ที่เราได้สร้างไว้ขึ้นมาแล้ว

```
This is Yii version 2.0.6.

The following commands are available:

- asset                         Allows you to combine and compress your
                                JavaScript and CSS files.
    asset/compress (default)    Combines and compresses the asset files a
                                to the given configuration.
    asset/template              Creates template of configuration file fo
                                [[actionCompress]].
// ......

- rbac
    rbac/init


To see the help of each command, enter:

  yii help <command-name>
```

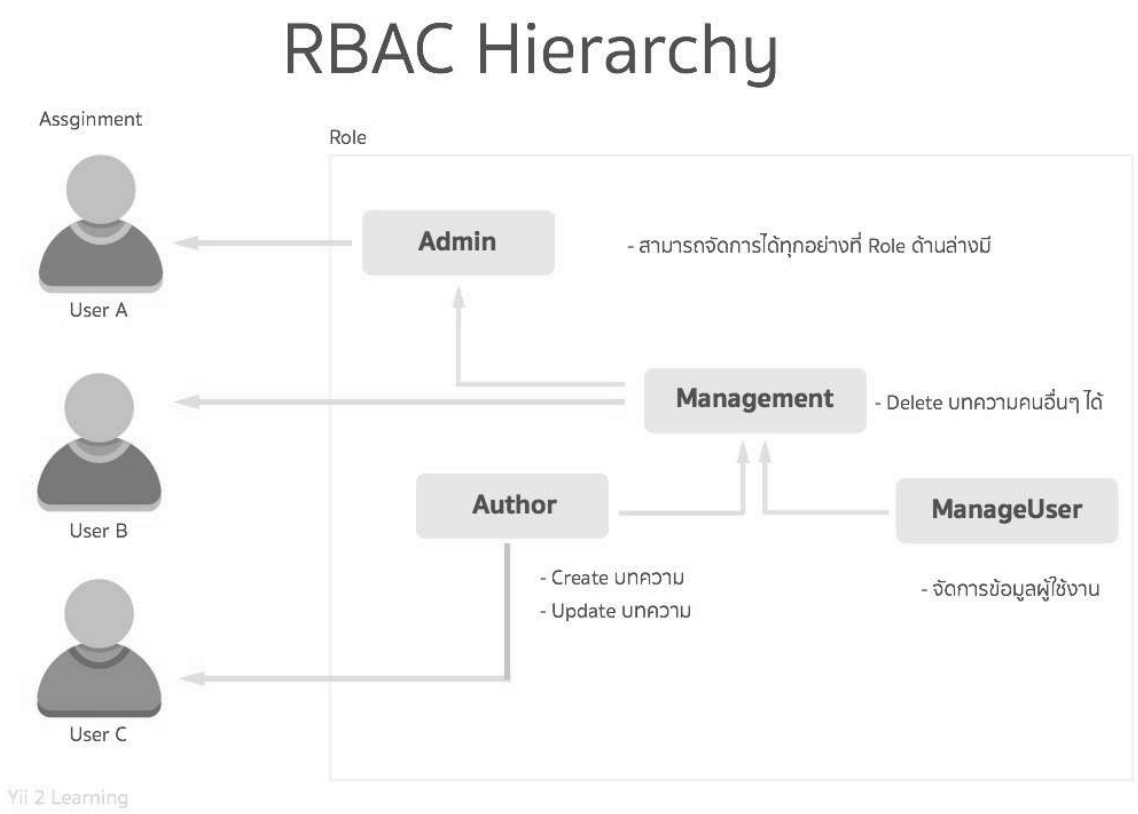
เราสามารถเรียกใช้งานได้เลย

```
 php yii rbac/init
```

จะพบข้อความดังนี้

```
Yii 2 Learning
```

หลังจากทดแล้วแล้วว่าใช้งานได้ ถึงเวลาที่เราจะสร้าง rbac จริงๆ ซึ่งเราได้ออกแบบไว้แล้ว ดูตามภาพ




ตามที่ออกแบบไว้เราจะแบ่ง Role เป็น 4 ประเภท และเพิ่ม ManageUser มาอีก 1 role

- Author สำหรับผู้ที่เขียนบทความ
- Management สำหรับจัดการข้อมูลบทความ
- ManageUser สำหรับจัดการข้อมูลผู้ใช้งาน
- Admin สำหรับผู้ดูแลระบบ

ซึ่งหากดูตามภาพแล้ว admin จะอยู่สูงสุดและให้ role ต่างๆ ไปอยู่ภายใต้ admin เพื่อทีจะให้ admin สามารถทำทุกอย่างได้ ตามที่ role ที่อยู่ภายใต้มีสิทธิ์ทำและเราได้ทำการเพิ่ม ManageUser เพื่อเอาไว้เป็นตัวคอยจัดการผู้ใช้งาน ที่ต้องแยกออกมากจาก Admin เพื่อให้เราสามารถให้สิทธิ์ในการจัดการข้อมูลกับคนอื่นๆ ที่ไม่ใช่ admin ได้ ระบบจะได้ยืดยุ่นขึ้น

หลังจากนั้นเราจะทำการ assignment ค่า role ให้กับ user ต่างๆ ดังนี้

- user-a : admin
- user-b : Management
- user-c : author
- user-d : ไม่ต้องให้สิทธิ์ เอาไว้ทดสอบ

> สังเกตว่าผมไม่ได้เรียกใช้งาน role ManageUser เลย เพราะ ManageUser อยู่ภายใต้ Management อยู่แล้วเมื่อเรียก Management ก็จะสามารถใช้งาน ManageUser ได้เลย

ในการสร้าง role เราจะเรียกใช้งานผ่าน `Yii::$app->authManager` เพื่อทำการสร้าง rbac โดยมีฟังก์ชันหลักๆ ที่ใช้งานดังนี้

- `removeAll()` เป็นฟังก์ชัน clear ข้อมูล rbac ทั้งหมด โปรดระวังการใช้งานด้วยครับ
- `createRole()` เป็นคำสั่งที่เอาไว้สร้าง role
- `add()` เป็นคำสั่งบันทึก role,rule,permission ตามที่กำหนด
- `addChild()` สามารถกำหนดให้ role,permission ที่ต้องการอยู่ภายใต้ role ใหนก็ได้
- `assign()` เป็นคำสั่งที่เอาไว้มอบ role ให้กับ user ที่ต้องการ

ไปที่ไฟล์ `console/controllers/RbacController.php` ให้ทำการสร้าง rbac ตามโค้ดด้านล่างนี้

```
<?php
//...
public function actionInit(){

  $auth = Yii::$app->authManager;
  $auth->removeAll();
  Console::output('Removing All! RBAC.....');

  $manageUser = $auth->createRole('ManageUser');
  $manageUser->description = 'สำหรับจัดการข้อมูลผู้ใช้งาน';
  $auth->add($manageUser);

  $author = $auth->createRole('Author');
  $author->description = 'สำหรับการเขียนบทความ';
  $auth->add($author);

  $management = $auth->createRole('Management');
  $management->description = 'สำหรับจัดการข้อมูลผู้ใช้งานและบทความ';
  $auth->add($management);

  $admin = $auth->createRole('Admin');
  $admin->description = 'สำหรับการดูแลระบบ';
  $auth->add($admin);

  $auth->addChild($management, $manageUser);
  $auth->addChild($management, $author);
  $auth->addChild($admin, $management);

  $auth->assign($admin, 1);
  $auth->assign($management, 2);
  $auth->assign($author, 3);
```

```
    Console::output('Success! RBAC roles has been added.');

}
```

จากนั้นทำการรันคำสั่งเพื่อสร้าง rbac ตาที่ได้ออกแบบไว้

```
php yii rbac/init
```

จะได้ผลลัพธ์ดังนี้

```
Removing All! RBAC.....
Success! RBAC roles has been added.
```

หลังจากนั้น ให้เราทำการตรวจสอบข้อมูลตารางที่ `auth_item` จะพบว่ามีข้อมูลขึ้นมาแล้ว 4 แถวซึ่งก็คือ Role ที่เราได้สร้างไว้นั่นเอง

```
mysql> select name,description from auth_item;
+------------+-----------------------------------------+
| name       | description                             |
+------------+-----------------------------------------+
| Admin      | สำหรับการดูแลระบบ                       |
| Author     | สำหรับการเขียนบทความ                    |
| Management | สำหรับจัดการข้อมูลผู้ใช้งานและบทความ    |
| ManageUser | สำหรับจัดการข้อมูลผู้ใช้งาน              |
+------------+-----------------------------------------+
4 rows in set (0.00 sec)
```

ในส่วนของ auth_item_child ที่เป็นตารางที่บอกว่า role ใหนอยู่ภายใต้ role อะไร

```
mysql> select * from auth_item_child;
+------------+------------+
| parent     | child      |
+------------+------------+
| Management | Author     |
| Admin      | Management |
| Management | ManageUser |
+------------+------------+
3 rows in set (0.00 sec)
```

และสุดท้าย `auth_assignment` จะเป็นตารางที่บอกว่า user คนใหนมีสิทธิ์อะไรหรือได้รับ role อะไร

```
mysql> select * from auth_assignment;
+------------+---------+------------+
| item_name  | user_id | created_at |
+------------+---------+------------+
| Admin      | 1       | 1440917151 |
```

```
| Author     | 3         | 1440917151 |
| Management | 2         | 1440917151 |
+------------+---------+------------+
3 rows in set (0.00 sec)
```

> สังเกตุ เราไมได้ให้สิทธิ์ user-d เพราะจะเอาไว้ทดสอบเข้าใช้งานในกรณีคนที่ไม่มีสิทธิ์

เสร็จเรียบร้อย การสร้าง RBAC DB มันง่ายๆ แค่นี้เอง (แต่กว่าจะเข้าใจก็เล่นเอาเหนื่อยเหมือนกัน อาจจะด้วยภาษาอังกฤษอันอ่อนด้อยของผม) หลังจากนี้ก็เป็นการตั้งค่าให้ controller ต่างๆ และการสร้างฟอร์ม assignment role ให้กับ user เพื่อให้เราจัดการสิทธิ์ได้ง่ายๆ

# รู้จักกับ Access Control Filter

ก่อนที่เราจะทำการเรียกใช้งาน rbac ที่เราได้สร้างขึ้น เราต้องทำความเข้าใจกับ Access Control Filter ใน Controller ก่อน Access Control ในส่วนของ controller จะเป็นตัวตรวจสอบการเข้าใช้งานในแต่ละ action โดยที่เราสามารถกำหนดเองได้ว่าแต่ละ role สามารถให้เข้าใช้งาน action อะไรได้บ้างเช่น `Author` สามารถเข้าใช้งาน index,view,update ถ้าเป็น role `Management` สามารถเข้าใช้งาน index,view,update,delete เป็นต้น และหากไม่มีสิทธิ์ก็จะไม่สามารถเข้าใช้งานได้ ซึ่งปกติค่า role default อยู่ 2 ตัว ที่เราสามารถเรียกใช้ได้เลยคือ

- ผู้ใช้งานที่ล็อกอิน (@)
- ผู้ใช้งานที่ไม่ได้ล็อกอิน (?)

โดยทั้ง 2 ตัวนี้เราสามารถกำหนดค่าในส่วน Access Control ของ Controller ได้ลองดูตัวอย่าง

```
<?php
use yii\web\Controller;
use yii\filters\AccessControl;

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

```
                    [
                        'allow' => true,
                        'actions' => ['logout'],
                        'roles' => ['@'],
                    ],
                ],
            ],
        ];
    }
    // ...
}
```

Access Control Filter เป็นส่วนของการกำหนดค่าให้แต่ละ action ว่า Role ไหนสามารถเข้าใช้งาน action อะไรได้บ้าง เราสามารถกำหนดค่าเพื่อเปิดใช้งานผ่าน `function behaviors()` ของ Controller ได้เลยโดยกำหนดภายใน property `access` หลังจากนั้นก็สามารถระบุได้เลยว่า action ไหนให้เข้าใช้งานยังไง ตัวอย่างเช่น action `logout` ต้อง login ก่อนถึงจะทำการเรียกใช้งาน logout ได้ จะสังเกตว่าเราจะใช้ roles เป็นเครื่องหมาย `@` แปลว่าต้องเป็นคนที่ login มาแล้วเท่านั้น ซึ่งในส่วนของ `actions` เราจะระบุกี่ action ก็ได้

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

ส่วนถ้ายังไม่ login ก็ใช้งานได้ 2 action คือ 'login', 'signup' และตรง roles ก็กำหนดค่าเป็น `?` เพื่อระบุว่าเป็นใครก็ได้ที่ยังไม่ได้ login

```
[
    'allow' => true,
    'actions' => ['login', 'signup'],
    'roles' => ['?'],
],
```

นี่เป็นการใช้งาน Access Control Filter แบบปกติทั่วไป เราจะสามารถควบคุมการเข้าใช้งานได้แค่ ล็อกอิน,ไม่ล็อกอิน แต่ถ้าหากเราจะควบคุมการเข้าใช้งานหลังจากล็อกอินแล้ว เช่น ผู้ใช้งานบางคนอาจมีสิทธิ์การเข้าใช้งานที่แตกต่างกัน จะไม่สามารถทำได้ ซึ่ง RBAC จะมาช่วยตรงนี้

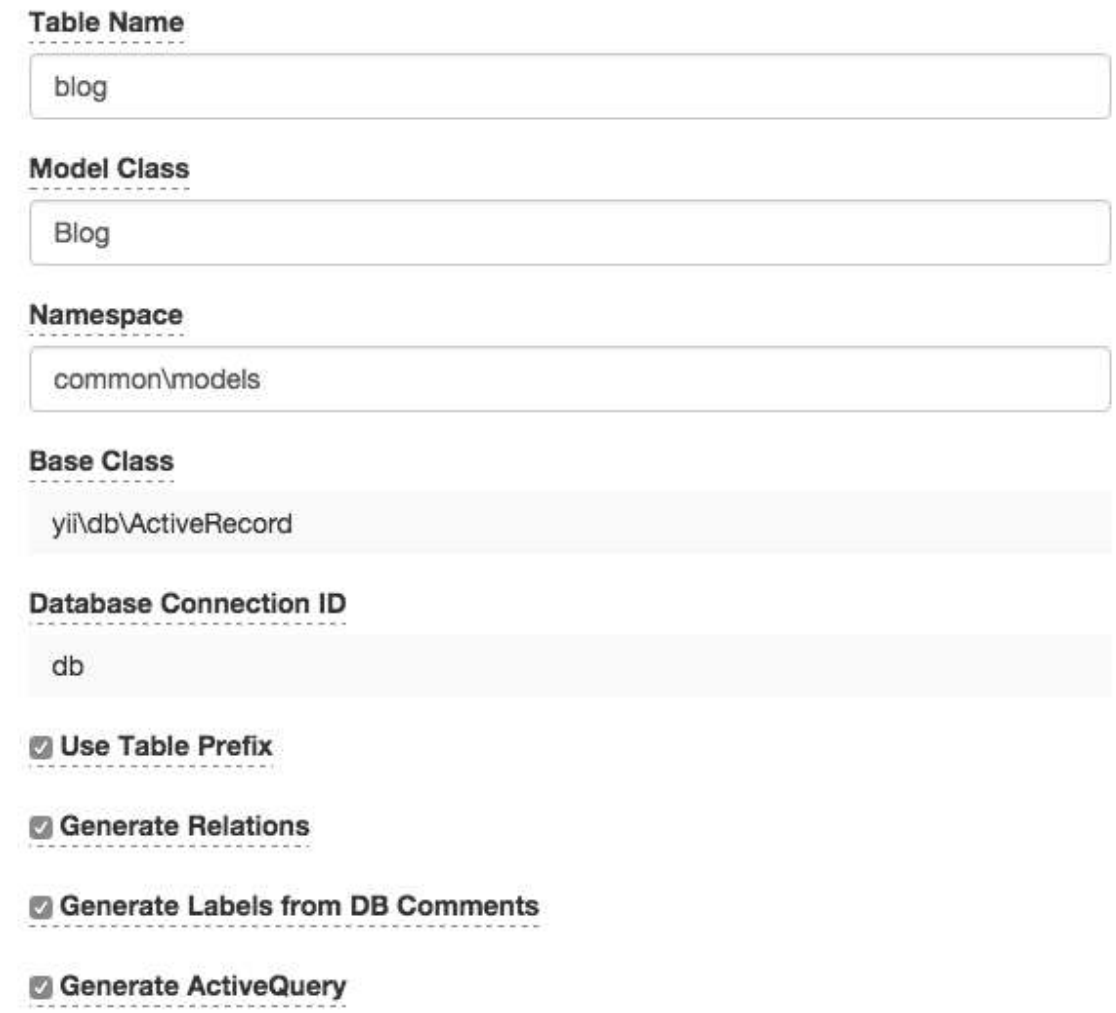
RBAC ที่เราจะใช้งานก็แค่เปลี่ยน role default `@` , `?` ให้เป็นชื่อ Role ตามที่เราได้ตั้งขึ้น เช่น Author,MangeUser,Management

# สร้าง Blog และเรียกใช้งาน RBAC

เราจะสร้าง Controller ขึ้นมาใหม่ เพื่อทดลองใช้งาน RBAC ที่เราสร้างขึ้น โดยจะทำระบบ blog ง่ายๆ ให้ทำการสร้างตาราง blog ดังนี้

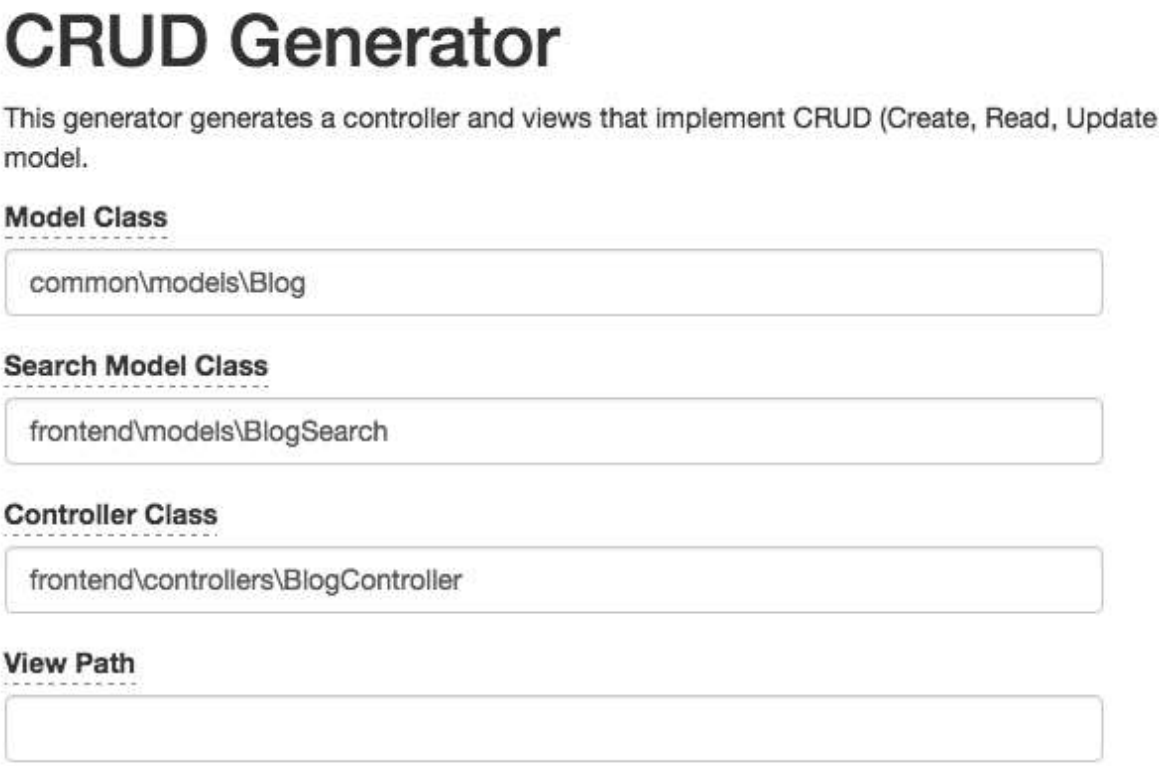
```
CREATE TABLE `blog` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(255) DEFAULT NULL COMMENT 'ชื่อเรื่อง',
  `content` text COMMENT 'เนื้อหา',
  `category` int(11) DEFAULT NULL COMMENT 'หมวดหมู่',
  `tag` varchar(255) DEFAULT NULL COMMENT 'คำค้น',
  `created_at` int(11) DEFAULT NULL COMMENT 'สร้างวันที่',
  `created_by` int(11) DEFAULT NULL COMMENT 'สร้างโดย',
  `updated_at` int(11) DEFAULT NULL COMMENT 'แก้ไขวันที่',
  `updated_by` int(11) DEFAULT NULL COMMENT 'แก้ไขโดย',
  PRIMARY KEY (`id`),
  KEY `title` (`title`),
  FULLTEXT KEY `content` (`content`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

จากนั้นทำการ Gii model ไว้ที่ common/models เพื่อให้สามารถใช้งานได้ทั้ง backend,frontend




---

เมื่อ Gii model Blog เสร็จเรียบร้อยให้เราทำการ Gii CRUD Blog โดยให้ gii ไว้ที่ frontend




> สังเกตผมจะไม่เอา BlogSearch ไว้ที่ common นะครับ เพราะในการใช้งาน search มันอาจมีการตั้งค่าหรือมีการเรียกใช้ที่แตกต่างกัน เงื่อนไขในการค้นหาต่างๆ ไม่เหมือนกัน ถ้าใช้ที่ frontend ก็ gii ไว้ที่ frontend เลย ถ้า backend ก็ gii ไว้ที่ backend เลยจะให้เราทำงานง่ายกว่า เพราะไม่ต้องใช้ไฟล์เดียวกัน

ปรับแต่ง model ที่ common/models/Blog.php เพื่อให้บันทึกรหัส created_by,updated_by โดยจะใช้ BlameableBehavior ดูรายละเดียดเพิ่มเติมที่นี่ ส่วน created_at,updated_at ใช้ TimestampBehavior ดูรายละเอียดได้ที่นี่ ทั้ง 2 ตัวนี้ระบบจะทำให้อัตโนมัติเราไม่ต้องทำอะไรเลยแค่ตั้งค่าเรียกใช้งานที่ model เลย

เปิดไฟล์ Blog.php ขึ้นมา แล้วทำการเรียกใช้งาน BlameableBehavior,TimestampBehavior จากนั้นตั้งค่าเปิดใช้งานที่ฟังก์ชัน behaviors()

```
<?php

use yii\behaviors\BlameableBehavior;
use yii\behaviors\TimestampBehavior;

// .....

class Blog extends \yii\db\ActiveRecord
{

    public function behaviors(){
      return [
```

```
            BlameableBehavior::className(),
            TimestampBehavior::className()
        ];
    }

    //...
}
```

จากนั้นไปที่ /frontend/views/blog/_blog.php ลบฟิวด์ที่ไม่ได้ใช้ออกไปคือ created_at,created_by,updated_at,updated_by เพราะเราใช้ behaviors เป็นตัวกรอกข้อมูลให้เราเอง ฟอร์มก็จะเหลือ 4 ฟิวด์ดังนี้

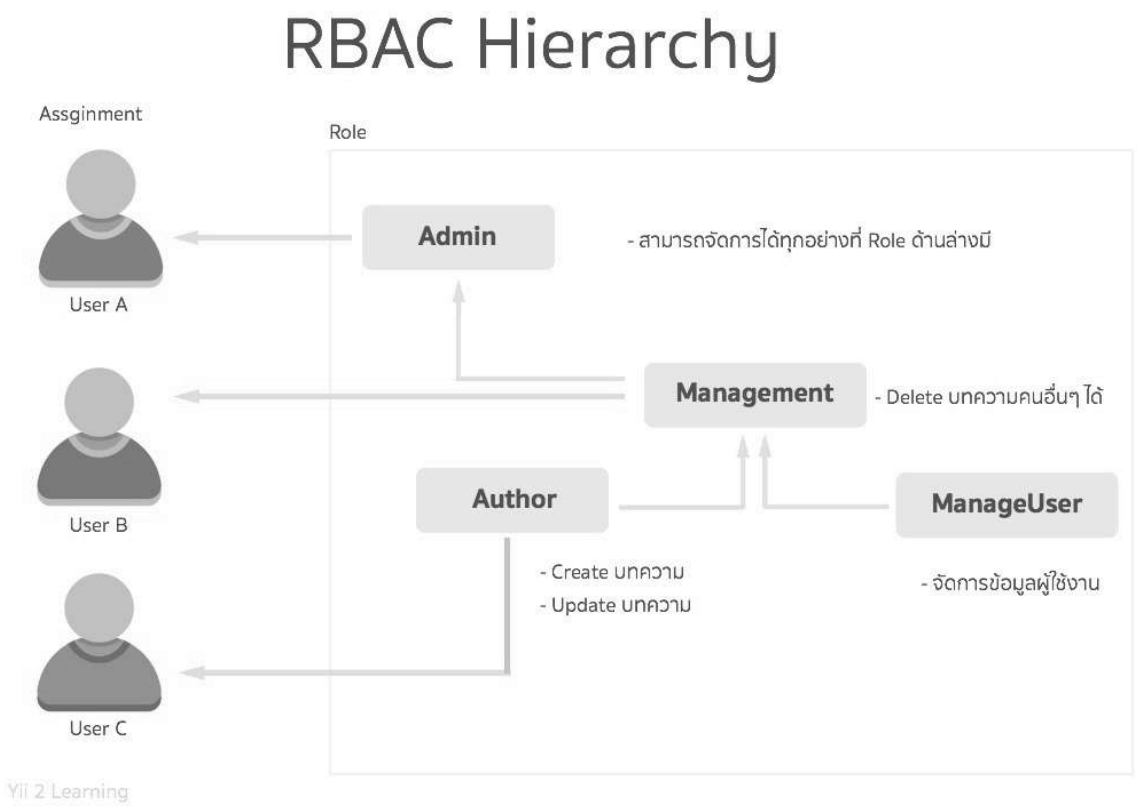
```
<?php

use yii\helpers\Html;
use yii\widgets\ActiveForm;

/* @var $this yii\web\View */
/* @var $model common\models\Blog */
/* @var $form yii\widgets\ActiveForm */
?>

<div class="blog-form">

    <?php $form = ActiveForm::begin(); ?>

    <?= $form->field($model, 'title')->textInput(['maxlength' => true])

    <?= $form->field($model, 'content')->textarea(['rows' => 6]) ?>

    <?= $form->field($model, 'category')->textInput() ?>

    <?= $form->field($model, 'tag')->textInput(['maxlength' => true]) ?>

    <div class="form-group">
        <?= Html::submitButton($model->isNewRecord ? Yii::t('app', 'Crea
    </div>

    <?php ActiveForm::end(); ?>

</div>
```

ให้ทดลอง login แล้วเข้าใช้งานที่ blog ว่าเข้าใช้งานได้หรือไม่ ทดลองเพิ่ม,ลบ,แก้ไขข้อมูลดู หากสังเกตุจะมีข้อมูล created_at,created_by ขึ้นมาแล้ว

```
mysql> select * from blog;
+----+-------+---------+----------+------+------------+------------+----
| id | title | content | category | tag  | created_at | created_by | upd
+----+-------+---------+----------+------+------------+------------+----
|  1 | sdf   | sdf     |        1 | 1    | 1440302545 |          3 | 144
```

```
+----+-------+---------+----------+------+-----------+-----------+----
1 row in set (0.00 sec)
```

ให้ไปที่ไฟล์ BlogCongroller.php เราจะตั้งค่า rbac โดยใช้ข้อมูลจากที่เราได้ออกแบบไว้ข้างต้นคือ




เปิดไฟล์ BlogCongroller.php ซึ่งของเดิมจะยังไม่มีการเรียกใช้งาน Access Control Filter

```
<?php
//...

class BlogController extends Controller
{
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
        ];
    }

    //...
```

```
}
```

ให้ทำการ use yii\filters\AccessControl; เพิ่มใช้กำหนดค่าในการเข้าใช้งาน action ต่างๆ ว่า role ใหนให้เค้า action อะไรบ้าง

```
<?php

namespace frontend\controllers;

use Yii;
use common\models\Blog;
use frontend\models\BlogSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl; //<<------ Use

/**
 * BlogController implements the CRUD actions for Blog model.
 */
class BlogController extends Controller
{
  public function behaviors()
{
    return [
        'verbs' => [
            'class' => VerbFilter::className(),
            'actions' => [
                'delete' => ['post'],
            ],
        ],
        'access'=>[
          'class'=>AccessControl::className(),
          'rules'=>[
            [
              'allow'=>true,
              'actions'=>['index','view','create','update'],
              'roles'=>['Author']
            ],
            [
              'allow'=>true,
              'actions'=>['delete'],
              'roles'=>['Management']
            ]
          ]
        ]
    ];
}

    //...

}
```

ถ้าดูตามโค้ดจะเห็นว่า เรากำหนดให้ index,view,create,update ให้เฉพาะ `Author` ส่วน action delete ให้เฉพาะ role `Management` ขึ้นไป ซึ่งก็จะเป็นลำดับตามภาพที่เราได้กำหนดค่าไว้

- user-a : Admin
- user-b : Management
- user-c : Author
- user-d : ไม่ได้กำหนด

## ลองใช้งาน user-d

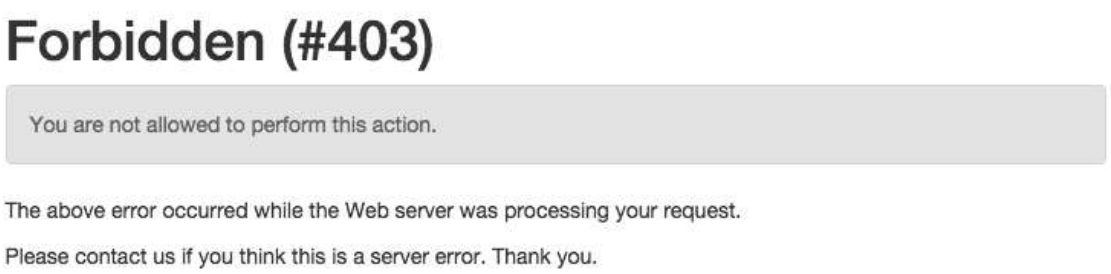
ลอง login ด้วย user-d และลองเข้า blog/index ดูจะพบว่าเข้าไม่ได้ เพราะว่าไม่ได้รับสิทธิ์ในการใช้งาน ไม่ว่าจะเข้า action อะไรก็ไม่สามารถเข้าได้ แต่อย่าลืม rbac จะรู้จักแค่ action ที่เรากำหนดไว้ใน actions=>['xxx','xxx'] เท่านั้นนะครับ

> อย่าลืมกำหนด property `'action'=>[....]` ใน rules ให้ครบทุกตัวด้วยนะครับ มี action กี่ตัวก็ระบุให้ครบ


## ลองใช้งาน user-c

ต่อไปให้ลอง login ด้วย user-c จะเข้าได้ปกติ action index,create,view,update ตามที่เรากำหนดไว้ใน rules ของ controller

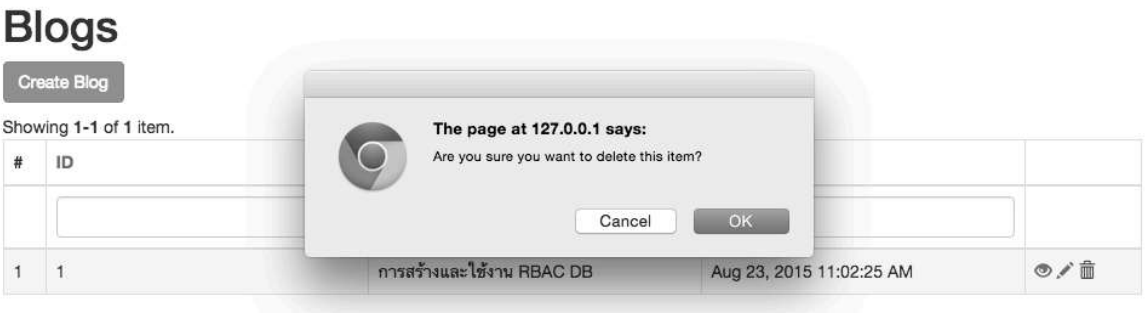

แต่เราจะไม่สามารถเข้าใช้งาน action Delete ได้ เพราะเรากำหนดว่าต้องเป็น role Management ขึ้นไปถึงจะสามารถลบได้




## ลองใช้งาน user-b,user-a

ส่วน user-a,user-b ก็จะสามารถเข้าได้เช่นกันและยังสามารถลบได้อีกด้วย เพราะเราระบุว่า user ที่จะลบได้ต้องเป็น สิทธิ์ Management ขึ้นไป ถึงแม้ว่าในส่วน access ของ Controller จะไม่ได้ระบุ role Admin,Management ไว้ เพราะ Author นั้นอยู่ภายใต้ Management และ Admin อีกทีตามลำดับมันจึงสามารถเข้าใช้งานได้นั่นเอง


สามารถลบข้อมูลได้





ในการตั้งค่า Access Control Filter และ RBAC กับ Controller ก็จะมีประมาณนี้

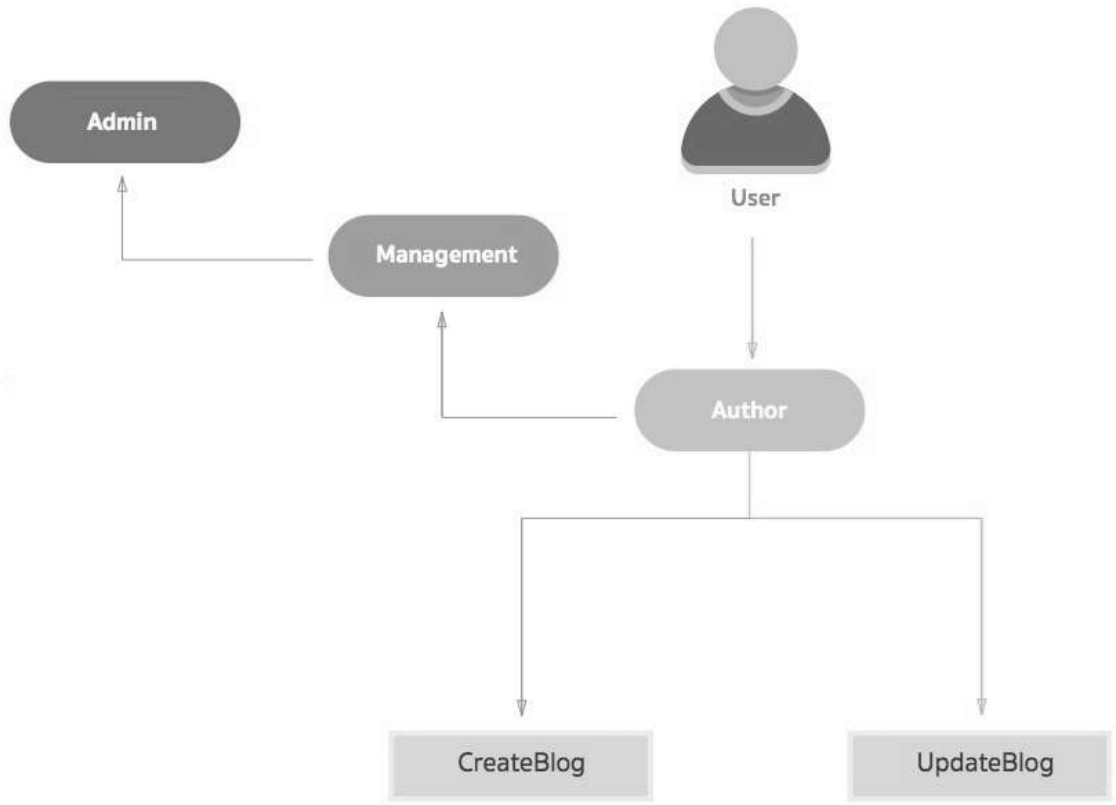
# Rule คืออะไร?

Rule คืออะไร? ถ้าจะให้ผมนิยาม Rule ง่ายๆ ผมคิดว่ามันคือตัวช่วยที่ทำให้ Role หรือ Permission ธรรมดาๆ ให้สามารถเขียนฟังก์ชันเพิ่มเติมเพื่อตรวจสอบค่าบางอย่างตามเงื่อนไขของเราได้ เพราะ Role หรือ permission มันทำหน้าที่เพียงเช็คว่าเรามีสิทธิ์หรือไม่เท่านั้น ถ้าจะตรวจสอบด้วยเงื่อนไขอื่นๆ Rule คือคำตอบ

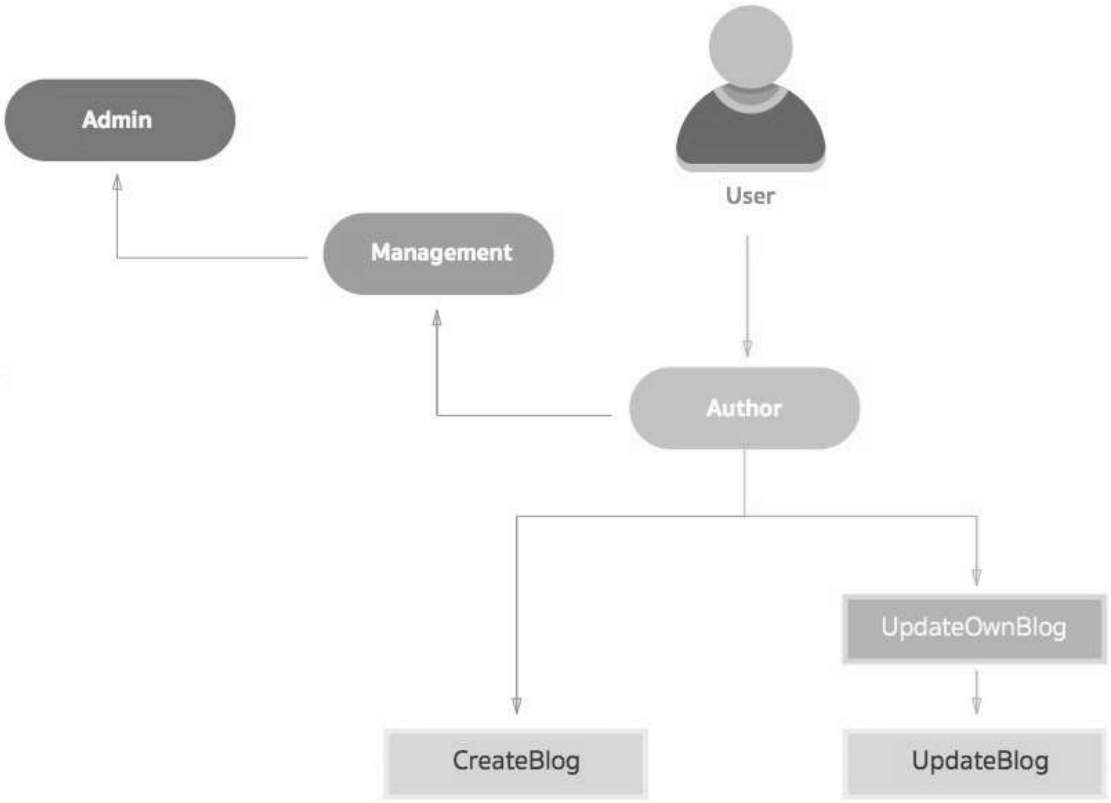
อย่างเช่น การใช้งานจริงๆ ในระบบ Blog ถ้าหากเราไม่ใช่คนสร้างบทความ ก็ไม่ควรจะไปแก้ไขบทความคนอื่นๆ ได้ ควรจะทำได้เฉพาะของตนเอง ดังนั้นเราจะสร้าง Rule ขึ้นมา 1 ตัวเพื่อเอาไว้เช็คว่าเป็นเจ้าของบทความจริงหรือไม่ โดยตรวจสอบ user_id ที่ login เข้ามาเทียบกับฟิวด์ created_by ของในตาราง blog หากเป็นคนๆ เดียวกันก็อนุญาติให้แก้ไขได้

> ไม่รู้ว่าผมเข้าใจถูกหรือปล่าวนะ แต่เท่าที่ทำมามันประมาณนี้ ผิดถูกยังไงแนะนำด้วยนะครับ

ลองดูที่ภาพด้านล่างนี้ ถ้าแบบปกตยังไม่มี Rule จะเห็นว่า ผู้ใช้งานจะสามารถ updateBlog ได้เลยไม่ว่าจะเป็นของใครก็ตามก็จะสามารถแก้ไขได้หมด เพราะไม่มี rule คอยตรวจสอบ




เพื่อแก้ไขปัญหาข้างต้น เราจะทำการสร้าง Rule ขึ้นมาเพื่อเป็นตัวตรวจสอบว่าบทความที่เขียนขึ้นมาใช่ของตัวเองหรือไม่ ถ้าใช่ถึงจะอนุญาติให้แก้ไขได้




Rule จะเป็นตัวช่วยให้เราสามารถตรวจสอบค่าบางอย่างเกี่ยวกับผู้ใช้งานได้ ซึ่งก็แล้วแต่ว่าเราจะตรวจสอบอะไรและใช้เงื่อนไขอะไร เพราะเราสามารถสร้างมันได้เอง

# สร้าง Rule เพื่อตรวจสอบเงื่อนไข

เราจะทำการสร้าง Rule ขึ้นมา 1 ตัว ชื่อ AuthorRule เอาไว้ตรวจสอบว่าบทความที่เรากำลังจะแก้ไขเป็นของเราจริงหรือไม่ ถ้าใช่ก็จะอนุญาติให้แก้ไขบทความ ถ้าไม่ใช่ก็จะ error forbidden แจ้งให้ทราบว่าไม่มีสิทธิ์เข้าใช้งาน

ให้ทำการสร้างไฟล์ ชื่อ AuthorRule.php ไว้ที่ common/rbac/AuthorRule.php หากไม่มีโฟลเดอร์ rbac ให้สร้างได้เลย

จากนั้นทำการสร้าง Rule ตามโค้ดด้านล่าง

```
<?php
namespace common\rbac;

use yii\rbac\Rule;

class AuthorRule extends Rule
{
    public $name = 'isAuthor';

    public function execute($user_id, $item, $params)
    {
        return isset($params['model']) ? $params['model']->created_by ==
    }
```

```
}
```

Rule นี้มีหน้าที่รับค่าเข้า model Blog เข้ามา ผ่านตัวแปร $params โดยใช้ฟิวด์ created_by เปรียบเทียบกับ $userId ว่าตรงกันหรือไม่ ถ้าใช่ก็ return เป็น true ซึ่งเท่ากับยอมให้เข้าใช้งาน แต่ถ้าหากเป็น false ก็จะ แสดง error forbidden ออกไป ส่วนค่า $userId นั้นมันส่งเข้ามาให้อัตโนมัติเราแค่ตั้งชื่อแล้วก็เรียกใช้งานได้เลย

ในการเรียกใช้งานง่ายมากๆ แค่ใช้ `Yii::$app->user->can()` เป็นตัวตรวจสอบให้ว่ามีสิทธ์หรือไม่

- ถ้าใช่ จะ return เป็น true
- ถ้าไม่จะ return เป็น false

และตัวฟังชัน `Yii::$app->user->can()` จะทำการรับค่า params 2 ตัวคือ

- ลำดับแรก เป็นชื่อ Rule
- ลำดับที่สอง เป็นค่า parameter ที่เราจะส่งเข้าไปเพื่อตรวจสอบ

ต่อไปทำการแก้ไข rbac ใหม่ โดยเพิ่ม permission ใหม่เข้าไป ใน `RbacController` ใหม่ดังนี้

```
<?php
namespace console\controllers;

use Yii;
use yii\helpers\Console;

class RbacController extends \yii\console\Controller {

  public function actionInit(){

    $auth = Yii::$app->authManager;
    $auth->removeAll();
    Console::output('Removing All! RBAC.....');

    $createPost = $auth->createPermission('createBlog');
    $createPost->description = 'Create a application';
    $auth->add($createPost);

    $updatePost = $auth->createPermission('updateBlog');
    $updatePost->description = 'Update application';
    $auth->add($updatePost);

    $admin = $auth->createRole('Admin');
    $auth->add($admin);

    $author = $auth->createRole('Author');
    $auth->add($author);

    $management = $auth->createRole('Management');
```

```
        $auth->add($management);

        // เรียกใช้งาน AuthorRule
        $rule = new \common\rbac\AuthorRule;
        $auth->add($rule);

        // สร้าง permission ขึ้นมาใหม่เพื่อเอาไว้ตรวจสอบและนำ AuthorRule มาใช้งานกับ upda
        $updateOwnPost = $auth->createPermission('updateOwnPost');
        $updateOwnPost->description = 'Update Own Post';
        $updateOwnPost->ruleName = $rule->name;
        $auth->add($updateOwnPost);

        $auth->addChild($author,$createPost);

        // เปลี่ยนลำดับ โดยใช้ updatePost อยู่ภายใต้ updateOwnPost และ updateOwnPost
        $auth->addChild($updateOwnPost, $updatePost);
        $auth->addChild($author, $updateOwnPost);

        $auth->addChild($management, $author);
        $auth->addChild($admin, $management);

        $auth->assign($admin, 1);
        $auth->assign($management, 2);
        $auth->assign($author, 3);
        $auth->assign($author, 4);

        Console::output('Success! RBAC roles has been added.');
      }

    }
```

> createBlog ผมสร้างเป็น permission ให้ดูเฉยๆ ไม่ได้เรียกใช้งาน

ตัวอย่างการเรียกใช้งานง่ายๆ แค่ใช้ if และอย่างที่บอกมันต้องรับค่า Blog มาด้วย เพื่อนำไปตรวจเช็คใน AuthorRule อีกที

```
<?php
if (\Yii::$app->user->can('updatePost', ['blog' => $model])) {
    // update post
}
```

เมื่อนำไปใช้ใน Controller เราจะต้องนำ `Yii::$app->user->can()` ไปครอบใน actionUpdate() เพื่อตรวจสอบก่อนจะให้เข้าใช้งาน จากปกติ actionUpdate() ก็จะประมาณนี้

```
<?php

// ...
public function actionUpdate($id)
{
```

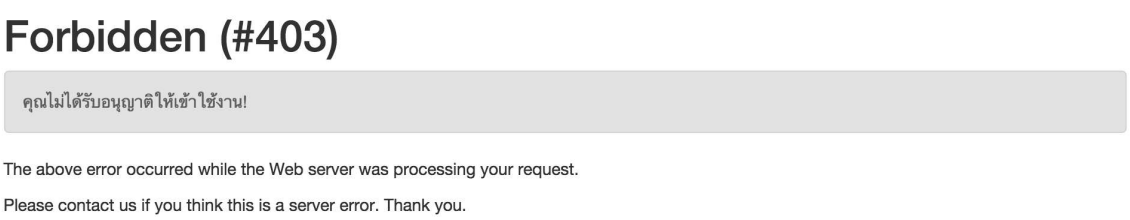
```
        $model = $this->findModel($id);
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        } else {
            return $this->render('update', [
                'model' => $model,
            ]);
        }
    }
```

เมื่อเรียกใช้งาน Yii::$app->user->can() ด้วยก็จะประมาณนี้

```
<?php
use ForbiddenHttpException;

// ...
public function actionUpdate($id)
{
  $model = $this->findModel($id);
  if (\Yii::$app->user->can('updateBlog', ['model' => $model])) {

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }

  }else{
     throw new ForbiddenHttpException('คุณไม่ได้รับอนุญาติให้เข้าใช้งาน!');
 }
}
```

ให้ลอง create,update ดู ถ้าหากเป็นบทความของตัวเองก็จะสามารถแก้ไขได้เลย


แต่ถ้าหากไม่ใช้ก็จะแสดง eror forbidden ออกมาเพื่อแจ้งให้ผู้ใช้ทราบว่าไม่สามารถเข้าใช้งานได้




# เรียกใช้งาน Rule ใน AccessControl

ในการใช้งานการตรวจสอบสิทธิ์ ถ้าหากเราใช้ `Yii::$app->user->can()` ไป if ใน action ดูไม่ค่อยสะดวกเท่าไหร่ เราจะเปลี่ยนมาเรียกใช้งานที่ AccessControl โดยใช้ `matchCallback` เป็นที่ตรวจสอบข้อมูล role ให้เรา

## จากปกติ

```
public function behaviors()
{
    return [
        'verbs' => [
            'class' => VerbFilter::className(),
            'actions' => [
                'delete' => ['post'],
            ],
        ],
        'access'=>[
          'class'=>AccessControl::className(),
          'rules'=>[
            [
              'allow'=>true,
              'actions'=>['index','view','create','update'],
              'roles'=>['Author']
            ],
            [
              'allow'=>true,
              'actions'=>['delete'],
              'roles'=>['Admin']
            ]
          ]
        ]
    ];
}
```

## เปลี่ยนเป็น

```
<?php

public function behaviors()
  {
      return [
          'verbs' => [
              'class' => VerbFilter::className(),
              'actions' => [
```

```
                'delete' => ['post'],
            ],
        ],
        'access'=>[
          'class'=>AccessControl::className(),
          'rules'=>[
            [
              'allow'=>true,
              'actions'=>['index','view','create'],
              'roles'=>['Author']
            ],
            [
              'allow'=>true,
              'actions'=>['update'],
              'roles'=>['Author'],
              'matchCallback'=>function($rule,$action){
                $model = $this->findModel(Yii::$app->request->get('id'
                if (\Yii::$app->user->can('UpdateBlog',['model'=>$mode
                        return true;
                }
              }
            ],
            [
              'allow'=>true,
              'actions'=>['delete'],
              'roles'=>['Admin']
            ]
          ]
        ]
    ];
```

สังเกตว่า rule แรกปกติจะมี `index` , `view` , `create` , `update` เราจำเป็นจะต้องแยก update ออกมาเพราะถ้าหากไม่แยก มันจะทำการเช็คข้อมูลทุก action ซึ่งเราอยากให้เช็คแค่ update เพราะฉะนั้นเราจึงแยก update ออกมาต่างหากนอกนั้นเหมือนเดิม

ส่วนที่ actionUpdate() ให้เราลบ `Yii::$app->user->can()` ออกให้เหลือเหมือนเดิม

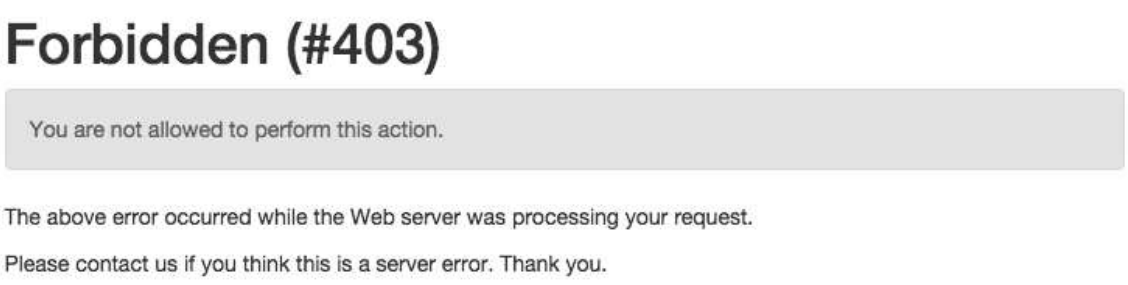
```
public function actionUpdate($id)
{
    $model = $this->findModel($id);
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}
```

จากนั้นทดลอง create,update ดูก็จะพบว่าสามารถใช้งานได้เหมือนเดิมแต่เราไม่ต้องไป if ครอบที่ actionUpdate แล้ว

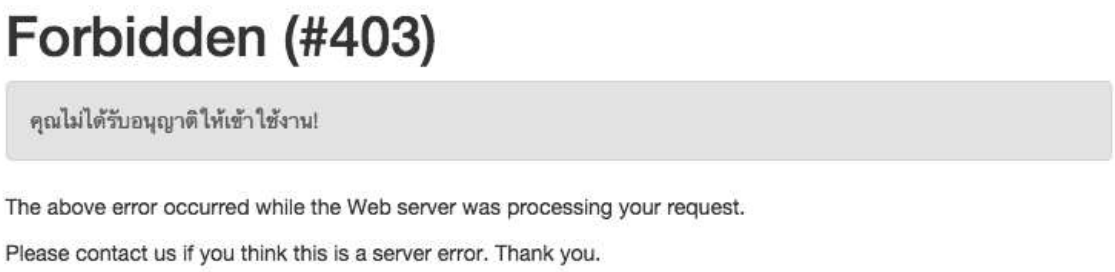
ปัญหาอีกอย่างคือถ้าหากเปิดแก้ไข blog แล้วไม่มีสิทธิ์มันจะแจ้งข้อความเป็น `You are not allowed to perform this action.` ซึ่งเราสามารถเปลี่ยนเป็นภาษาไทยได้เพียงเพิ่ม `denyCallback` เข้าไปดังนี้

```
<?php
use yii\web\ForbiddenHttpException;
// ...

public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
            'access'=>[
              'class'=>AccessControl::className(),
              'denyCallback' => function ($rule, $action) {
                  throw new ForbiddenHttpException('คุณไม่ได้รับอนุญาติให้เข้าใช้ง
              },
              'rules'=>[
                [
                  'allow'=>true,
                  'actions'=>['index','view','create'],
                  'roles'=>['Author']
                ],
                [
                  'allow'=>true,
                  'actions'=>['update'],
                  'roles'=>['Author'],
                  'matchCallback'=>function($rule,$action){
                    $model = $this->findModel(Yii::$app->request->get('i
                    if (\Yii::$app->user->can('UpdateBlog',['model'=>$mo
                              return true;
                    }
                  }
                ],
                [
                  'allow'=>true,
                  'actions'=>['delete'],
                  'roles'=>['Admin']
                ]
              ]
            ]
        ];
    }
```

ปกติหากไม่ใส่ `denyCallback` ก็จะเป็นภาษาอังกฤษแบบนี้




เมื่อกำหนดข้อความ error แล้วแล้วก็จะได้แบบนี้




# ปรับปรุงระบบ Registration เพื่อใส่ค่า default Role

ในขั้นตอนของการ Registration ตอนนี้จะยังไม่มี Role default ให้เพราะเรายังไม่ได้ทำการตั้งค่า หากทำการ Registration ในตอนนี้จะ Registration ได้และ login ได้ปกติ แต่จะใช้งานในส่วนที่เราเปิดใช้งาน RBAC ไม่ได้ เพราะฉะนั้นเราจะต้องทำการใส่ค่า default role ให้ระบบในตอน Registration ก่อน

- ไปที่ frontend/controllers/SiteController.php
- ไปที่ ActionSignUp() จะสังเกตว่ามัน ได้ทำการเรียกใช้งาน Model SignupForm อยู่
- ให้เราเปิดไปที่ไฟล์ frontend/models/SignupForm.php ไปที่ `actionSignup()` เพิ่มโค้ดตามนี้

ของเดิม

```
<?php

//...
public function signup()
{
    if ($this->validate()) {
        $user = new User();
        $user->username = $this->username;
        $user->email = $this->email;
        $user->setPassword($this->password);
        $user->generateAuthKey();
        if ($user->save()) {
            return $user;
```

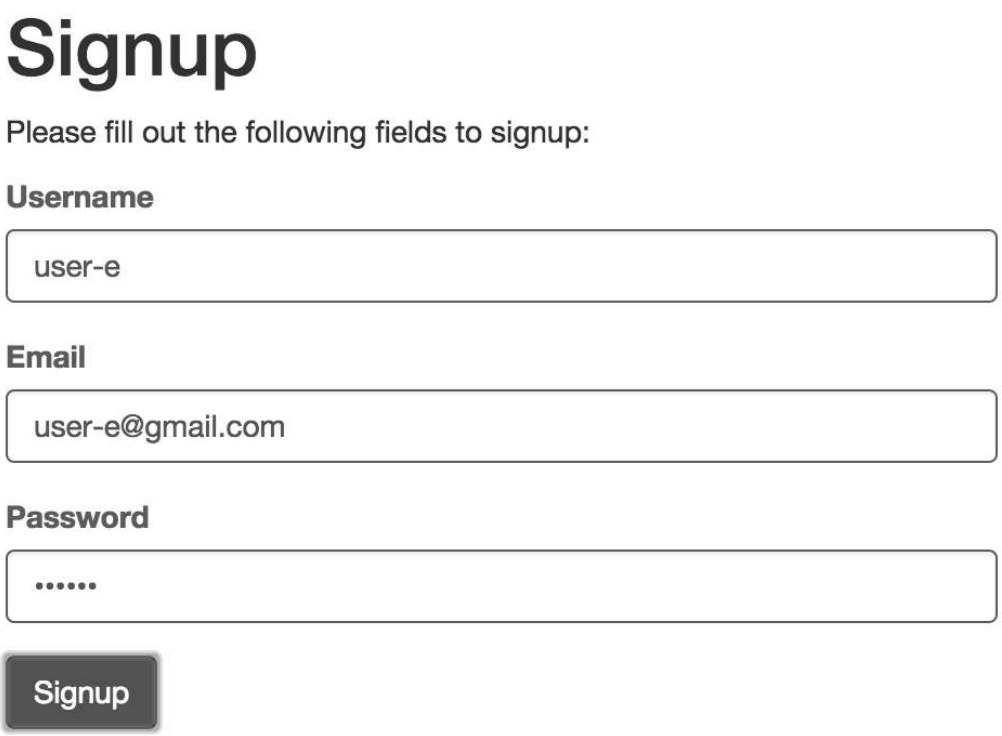
```
            }
        }

        return null;
    }
```

เพิ่มส่วนนี้เข้าไป เพื่อใส่ค่า role `Author` เป็น role default ในตอน registration

```
<?php

//...

public function signup()
{
    if ($this->validate()) {
        $user = new User();
        $user->username = $this->username;
        $user->email = $this->email;
        $user->setPassword($this->password);
        $user->generateAuthKey();
        if ($user->save()) {

            $auth = Yii::$app->authManager;
            $authorRole = $auth->getRole('Author');
            $auth->assign($authorRole, $user->getId());

            return $user;
        }
    }

    return null;
}
```

เราจะเรียกตัวจัดการ RBAC คือ `Yii::$app->authManager` เพื่อทำการเรียก Role ที่เราจะใส่เป็น Role Default ให้ตอนนี้เราจะเลือก `Author` หลังจากนั้นก็ทำการ assignment ให้ user ที่ Registration เข้ามา ผู้ใช้งานก็จะได้รับ Role Author ทันที

ทดลอง Registration เพิ่ม user-e เข้าไป




ลองตรวจสอบข้อมูลที่ Mysql ดูที่ตาราง user ก่อน จะเห็น user-e เข้ามาแล้ว

```
mysql> select id,username,email from user;
+----+----------+------------------+
| id | username | email            |
+----+----------+------------------+
|  1 | user-a   | user-a@gmail.com |
|  2 | user-b   | user-b@gmail.com |
|  3 | user-c   | user-c@gmail.com |
|  4 | user-d   | user-d@gmail.com |
|  5 | user-e   | user-e@gmail.com |
+----+----------+------------------+
5 rows in set (0.00 sec)
```

ตรวจสอบที่ item_assign เพื่อเช็คว่ามีการ assignment Role ชื่อ Author ให้หรือยัง

```
mysql> select * from auth_assignment;
+------------+---------+------------+
| item_name  | user_id | created_at |
+------------+---------+------------+
| Admin      | 1       | 1440917151 |
| Author     | 3       | 1440917151 |
| Author     | 5       | 1440933014 |
| Management | 2       | 1440917151 |
+------------+---------+------------+
4 rows in set (0.00 sec)
```

จะพบว่ามี user_id = 5 ที่ได้รับค่า role = Author แล้ว ซึ่ง user_id = 5 คือ user-e ที่เรา Registration นั้นเอง จากนั้นก็ทดสอบเข้าใช้งาน Blog ก็จะใช้งานได้ตามสิทธิ์ที่ได้รับ

# สร้างระบบจัดการผู้ใช้งานและระบบจัดการสิทธิ์

เราจะสร้างระบบจัดการผู้ใช้งานเพื่อใช้ในการ เพิ่ม,ลบ,แก้ไข ข้อมูลผู้ใช้งานทั้งหมด และการจัดการสิทธิ์ให้กับผู้ใช้งานแต่ละคน ซึ่งใน controller นี้เราจะใช้งานแค่ผู้ที่ได้รับ role `ManageUser` เท่านั้น

ซึ่งเราจะสร้างระบบจัดการข้อมูลผู้ใช้งานไว้ที่ Backend โดยทำการ gii CRUD ตาราง user ขึ้นมาใหม่ โดยใช้ model User เดิมที่อยู่ไหน `common/models/User.php` ซึ่งเราจะต้องทำการปรับปรุง role() ใหม่ ดังนี้

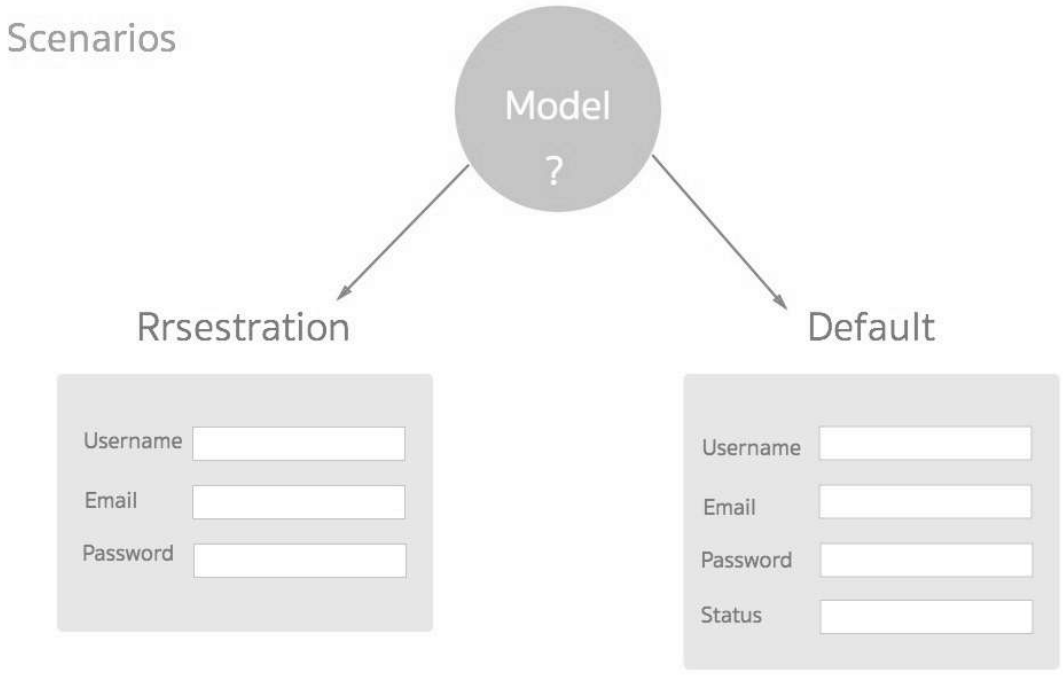
ของเดิม

```
<?php

// ...

public function rules()
{
    return [
        ['status', 'default', 'value' => self::STATUS_ACTIVE],
        ['status', 'in', 'range' => [self::STATUS_ACTIVE, self::STATUS_D
    ];
}
```

เพิ่มการกำหนด validation ในส่วนของ role ใหม่และเพิ่ม getStatusItem(),getStatusName() เพื่อใช้แสดงผลสถานะตอนแสดงที่ gridview

อีกส่วนที่สำคัญคือเราได้เพิ่มฟังก์ชัน scenarios() เพื่อให้เราสามารถเรียกใช้งาน model user ในตอน registration และ create & update ในส่วน ManageUser ได้ scenarios คือส่วนที่เราสามารถแยกส่วนของการ validation ได้เช่นในหน้า ฟอร์ม registration นั้นเรามีการรับค่าแค่ 3 ฟิวด์คือ username,email,password ถ้าหาเราไม่ได้แยก scenario ไว้ก็จะทำการ validation ทุกฟิวด์ที่มี เพราฉะนันเราจะแยกเป็น 2 scenario คือ

Registration จะใช้ฟิวด์ username,email ส่วนในหน้าบันทึกเราจะใช้ default คือ validation ทุกฟิวด์




ด้วยวิธีนี้จึงทำให้เราสามารถใช้ model ตัวเดียวแต่สามารถสร้างฟอร์มกรอกข้อมูลหลายๆ แบบในตารางเดียวเช่น registration ใช้แค่ username,password แต่ในฟอร์ม create อาจจะใช้ username,email,password,confirm_password,status เป็นต้น

เวลาเรียกใช้งาน เช่นตอน $model = new User(['scenario'=>'registration']);

หรือถ้าเป็น default ก็ $model = new User(); หรือจะเรียกผ่าน method ก็ได้เช่น

```
 $model = new User();
 $model->scenario = 'registration';
```

```
<?php

class User extends ActiveRecord implements IdentityInterface
{
    const STATUS_DELETED = 0;
    const STATUS_ACTIVE = 10;

    public $password;
    public $confirm_password;
    public $roles;

//...

    public function scenarios()
    {
        $scenarios = parent::scenarios();
        $scenarios['registration'] = ['username','email'];
        return $scenarios;
    }

    public function rules()
```

```
{
    return [
        //['status', 'required'],
        ['status', 'default', 'value' => self::STATUS_ACTIVE],
        ['status', 'in', 'range' => [self::STATUS_ACTIVE, self::STAT

        ['username', 'filter', 'filter' => 'trim'],
        ['username', 'required'],
        ['username', 'unique', 'targetClass' => '\common\models\User
        ['username', 'string', 'min' => 2, 'max' => 255],

        ['email', 'filter', 'filter' => 'trim'],
        ['email', 'required'],
        ['email', 'email'],
        ['email', 'unique', 'targetClass' => '\common\models\User',

        ['password', 'required'],
        ['password', 'string', 'min' => 6],

        ['confirm_password', 'required'],
        ['confirm_password', 'string', 'min' => 6],
        ['confirm_password', 'compare','compareAttribute'=>'password

        ['roles', 'safe']

    ];
}
```

```
  // ...

  public function getItemStatus(){
    return [
      self::STATUS_ACTIVE => 'Active',
      self::STATUS_DELETED => 'Deleted'
    ];
  }
  public function getStatusName()
  {
    $items = $this->getItemStatus();
    return array_key_exists($this->status, $items) ? $items[$this->
  }
```

หลังจากที่ปรับปรุง model User เสร็จเรียบร้อยให้ทำการ Gii CRUD model `User` และปรับปรุงโค้ดในแต่ละส่วนดังต่อไปนี้

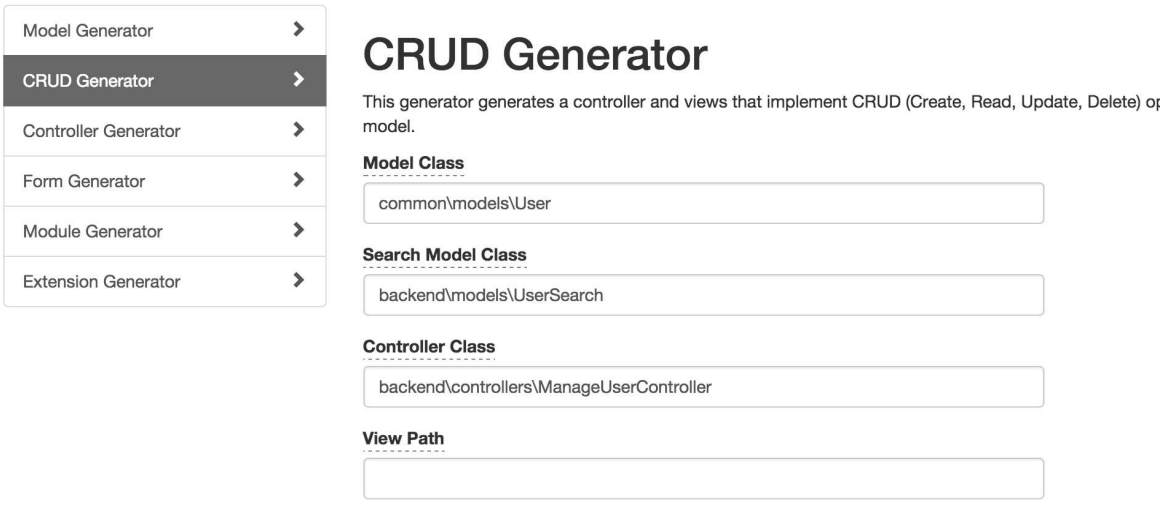
> เนื่องจากเราทำการเพิ่ม scenarios() เข้าไปจำเป็นต้องระบุ scenario = registration ในตอน signup ด้วย

ให้เราเปิดไปที่ไฟล์ frontend/models/SignupForm.php ไปที่ `actionSignup()` เพิ่มโค้ดตามนี้

```php
<?php

//...

public function signup()
{
    if ($this->validate()) {
        $user = new User(['scenario'=>'registration']); // <<<<------
        $user->username = $this->username;
        $user->email = $this->email;
        $user->setPassword($this->password);
        $user->generateAuthKey();
        if ($user->save()) {

            $auth = Yii::$app->authManager;
            $authorRole = $auth->getRole('Author');
            $auth->assign($authorRole, $user->getId());

            return $user;
        }
    }

    return null;
}
```

### Gii CRUD User

ไปที่ backend แล้ว gii CRUD



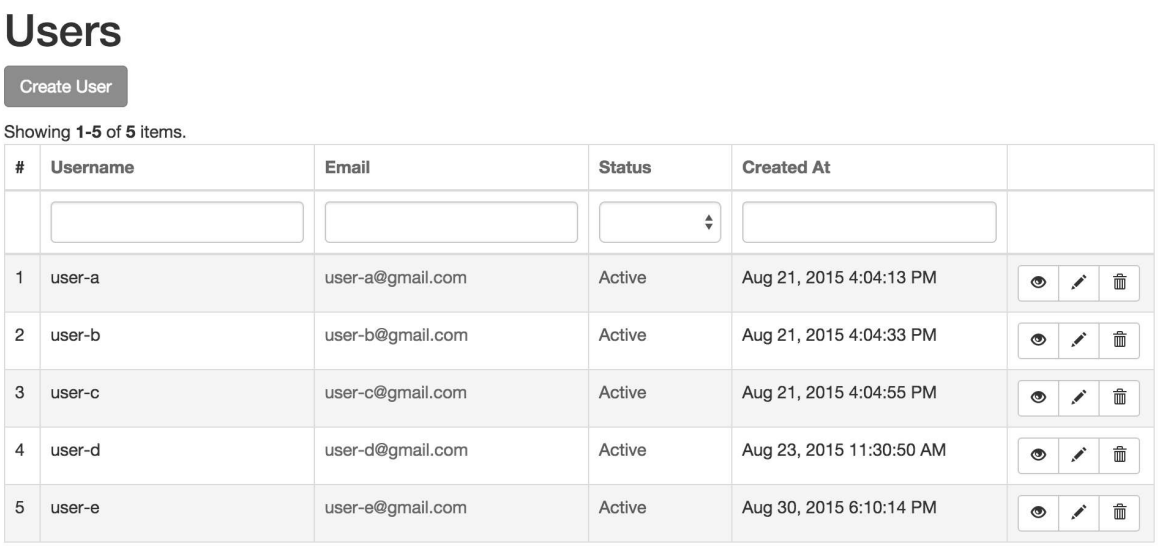

### views/manage-user/index.php

มีการปรับแต่ง columns และใส่ filter ให้กับ status เป็น dropdownList เพื่อให้สามารถค้นหาได้ง่าย

```php
<?php

use yii\helpers\Html;
use yii\grid\GridView;
```

```
/* @var $this yii\web\View */
/* @var $searchModel backend\models\UserSearch */
/* @var $dataProvider yii\data\ActiveDataProvider */

$this->title = Yii::t('app', 'Users');
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="user-index">

    <h1><?= Html::encode($this->title) ?></h1>
    <?php // echo $this->render('_search', ['model' => $searchModel]); ?

    <p>
        <?= Html::a(Yii::t('app', 'Create User'), ['create'], ['class' =
    </p>

    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'filterModel' => $searchModel,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],

            //'id',
            'username',
            //'auth_key',
            //'password_hash',
            //'password_reset_token',
             'email:email',
             //'status',
             [
               'attribute'=>'status',
               'format'=>'html',
               'filter'=>$searchModel->itemStatus,
               'value'=>function($model){
                 return $model->statusName=='Active' ?'<span class="text
               }
             ],
             'created_at:dateTime',
            // 'updated_at',

            [
              'class' => 'yii\grid\ActionColumn',
              'options'=>['style'=>'width:120px;'],
              'buttonOptions'=>['class'=>'btn btn-default'],
              'template'=>'<div class="btn-group btn-group-sm text-cente
           ],
        ],
    ]); ?>
```

**เมื่อทดสอบรันดูก็จะได้หน้าตาประมาณนี้**




## ปรับปรุง model User.php

ปรับปรุง model User เพื่อสร้างฟังก์การจัดการสิทธิ์เกี่ยวกับ user

ไปที่ไฟล์ `common/models/User.php` แล้วสร้างฟังก์ชันตามโค้ดด้านล่าง เพื่อใช้จัดการสิทธิ์ต่างๆ ของผู้ใช้งาน

```
<?php

//...

public function getAllRoles(){
  $auth = $auth = Yii::$app->authManager;
  return ArrayHelper::map($auth->getRoles(),'name','name');
}

public function getRoleByUser(){
  $auth = Yii::$app->authManager;
  $rolesUser = $auth->getRolesByUser($this->id);
  $roleItems = $this->getAllRoles();
  $roleSelect=[];

  foreach ($roleItems as $key => $roleName) {
    foreach ($rolesUser as $role) {
      if($key==$role->name){
        $roleSelect[$key]=$roleName;
      }
    }
  }
  $this->roles = $roleSelect;
}

public function assignment(){
    $auth = Yii::$app->authManager;
    $roleUser = $auth->getRolesByUser($this->id);
    $auth->revokeAll($this->id);
    foreach ($this->roles as $key => $roleName) {
      $auth->assign($auth->getRole($roleName),$this->id);
```

```
    }
}
```

- getAllRoles() ดึงข้อมูล role items ที่มีทั้งหมดใน rbac db ที่เราได้สร้างไว้
- getRoleByUser() ดึงข้อมูล role ที่ user ได้รับ
- assignment() ทำการ assignment role ที่ได้เลือกในฟอร์ม ให้กับ user

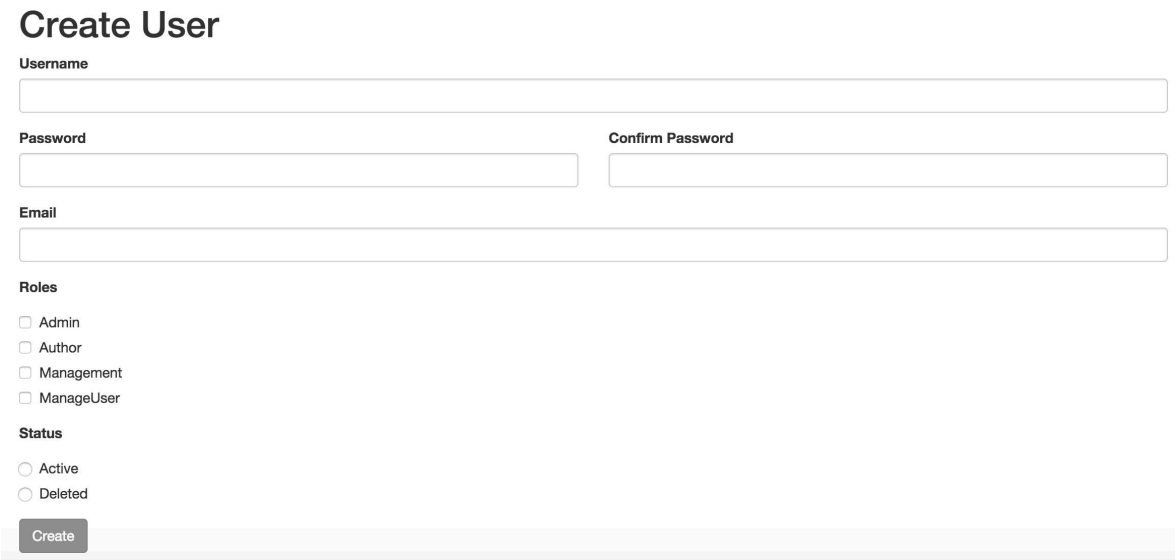
**ปรับปรุง _form.php**

ปรับ layout และสร้างฟอร์มเพื่อแสดงผล role โดยใช้ฟังก์ชัน getAllRoles() ที่เราได้สร้างไว้ใน model user เพื่อดึงข้อมูล role ที่มีทั้งหมดให้สามารถเลือกสิทธิ์ให้แต่ user ได้

```
<?php

use yii\helpers\Html;
use yii\bootstrap\ActiveForm;

/* @var $this yii\web\View */
/* @var $model common\models\User */
/* @var $form yii\widgets\ActiveForm */
?>

<div class="user-form">

    <?php $form = ActiveForm::begin(); ?>

    <?= $form->field($model, 'username')->textInput(['maxlength' => true
    <div class="row">
      <div class="col-lg-6">
          <?= $form->field($model, 'password')->passwordInput(['maxlengt
      </div>
      <div class="col-lg-6">
        <?= $form->field($model, 'confirm_password')->passwordInput(['ma
      </div>
    </div>
    <?= $form->field($model, 'email')->textInput(['maxlength' => true])

    <?= $form->field($model, 'roles')->checkboxList($model->getAllRoles(

    <?= $form->field($model, 'status')->radioList($model->getItemStatus(

    <div class="form-group">
        <?= Html::submitButton($model->isNewRecord ? Yii::t('app', 'Crea
    </div>

    <?php ActiveForm::end(); ?>

</div>
```

**ปรับปรุง actionCreate()**

เราจะปรับปรุง `actionCreate()` เพื่อให้สามารถกำหนด role ให้กับ user ได้ โดยเราจะใช้ checkBoxList เป็นตัวเลือกและนำรายการ role ที่เลือกไปบันทึกในส่วนของ assignment ของ RBAC

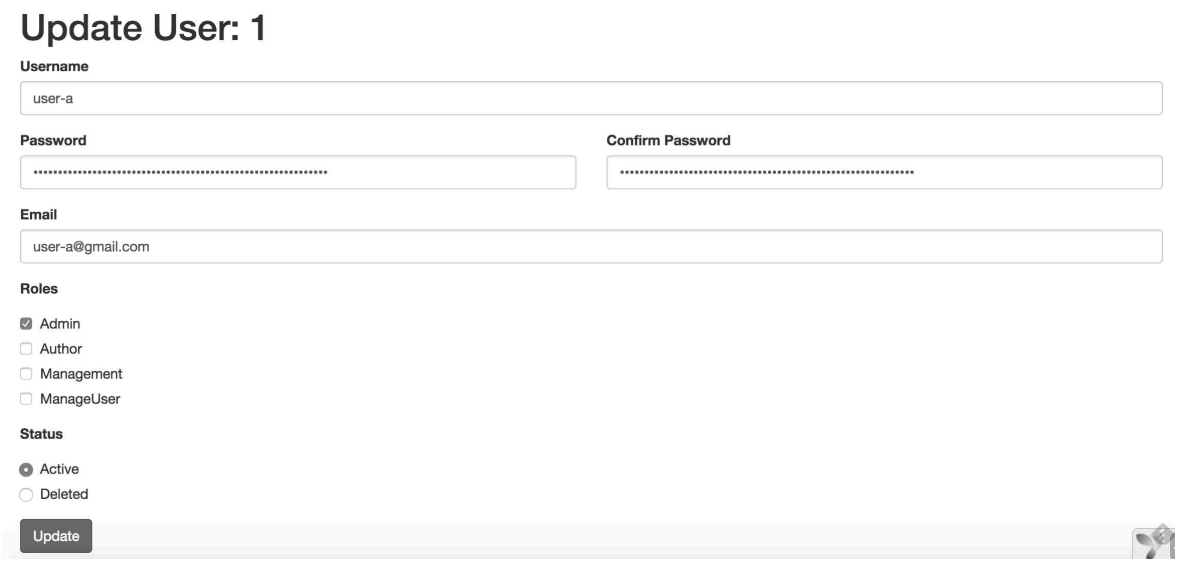



```php
<?php

//...

public function actionCreate()
{
    $model = new User();

    if ($model->load(Yii::$app->request->post()) && $model->validate())
        $model->setPassword($model->password);
        $model->generateAuthKey();
        if($model->save()){
          $model->assignment();
        }
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}
```

ฟังก์ชัน assignsent() จะนำ role ที่เราได้เลือกในฟอร์มไปบันทึกว่าผู้ใช้งานมีสิทธิ์อะไร

### ปรับปรุง actionUpdate()

เราจะเพิ่มฟังก์ชันเพื่อเรียก role ที่เค้าได้รับมาแสดงในฟอร์ม จากนั้นมีการอัพเดทข้อมูลในฟอร์ม ก็จะทำการ assignsent role ตามทีได้เลือกไว้




```php
<?php
public function actionUpdate($id)
{
    $model = $this->findModel($id);
    $model->getRoleByUser();
    $model->password = $model->password_hash;
    $model->confirm_password = $model->password_hash;
    $oldPass = $model->password_hash;

    if ($model->load(Yii::$app->request->post()) && $model->validate())
      if($oldPass!==$model->password){
        $model->setPassword($model->password);
      }
      if($model->save()){
        $model->assignment();
      }

        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}
```

ใน `actionUpdate()` เราจะใช้ฟังก์ชัน getRoleByUser() เพื่อดึงข้อมูล role ที่ user ได้รับเพื่อเอามาแสดงผลในฟอร์มว่า user ได้รับสิทธิ์อะไรบ้าง และมีการเช็ค password ว่ามีการกรอกข้อมูลเข้ามาใหม่หรือไม่หากมีก็ให้นำ password ไปเข้ารหัสใหม่ก่อนทำการบันทึก

### ปรับปรุง view.php

ทำการเรียกฟิวด์ที่ต้องการเพื่อแสดงผล

```
<?php

<?php

use yii\helpers\Html;
use yii\widgets\DetailView;

/* @var $this yii\web\View */
/* @var $model common\models\User */

$this->title = $model->id;
$this->params['breadcrumbs'][] = ['label' => Yii::t('app', 'Users'), 'ur
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="user-view">

    <h1><?= Html::encode($this->title) ?></h1>

    <p>
        <?= Html::a(Yii::t('app', 'Update'), ['update', 'id' => $model->
        <?= Html::a(Yii::t('app', 'Delete'), ['delete', 'id' => $model->
            'class' => 'btn btn-danger',
            'data' => [
                'confirm' => Yii::t('app', 'Are you sure you want to del
                'method' => 'post',
            ],
        ]) ?>
    </p>

    <?= DetailView::widget([
        'model' => $model,
        'attributes' => [
            'id',
            'username',
            // 'auth_key',
            // 'password_hash',
            // 'password_reset_token',
            'email:email',
            'statusName',
            'created_at:dateTime',
            'updated_at:dateTime',
        ],
    ]) ?>

</div>
```

**AccessControl**

ในส่วนสุดท้ายที่จะปรับปรุงคือ Access Control Filter เพื่อกำหนดสิทธิ์เพื่อระบุให้เฉพาะ role ที่เท่ากับ ManageUser เท่านั้นที่สามารถใช้งานได้

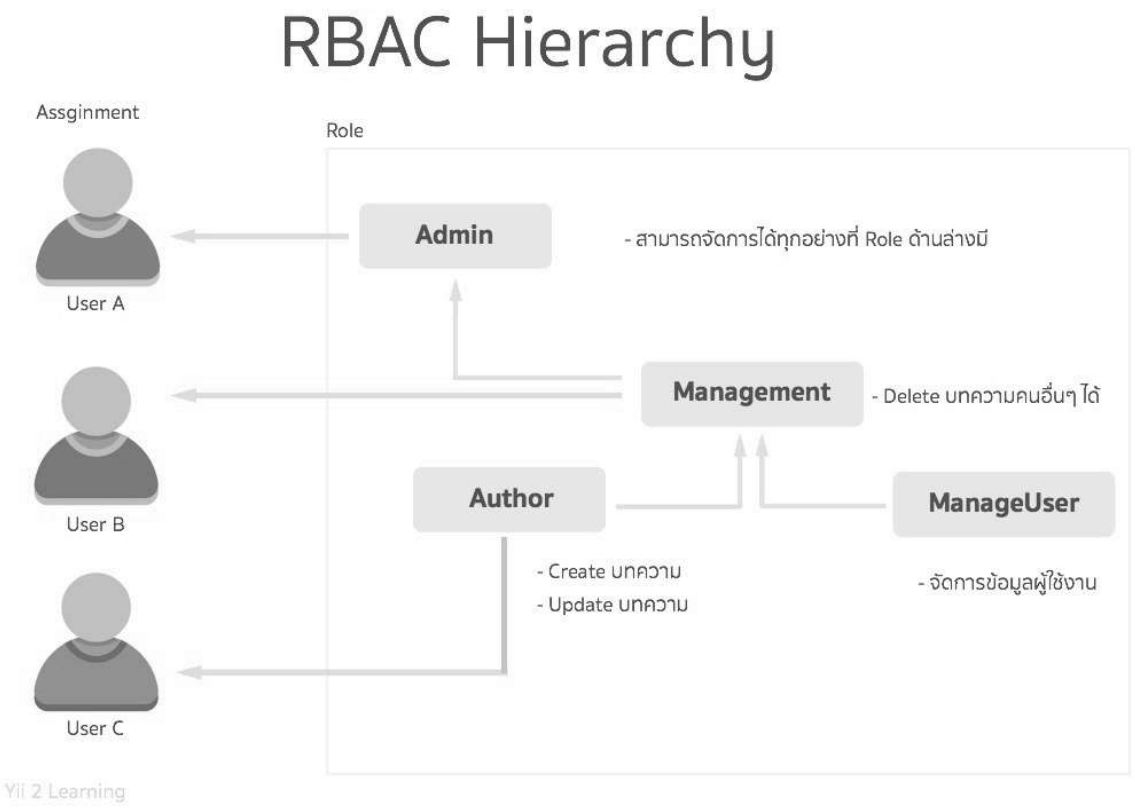
ไปที่ไฟล์ ManageUserController.php เพื่อทำการกำหนดสิทธิ์การเข้าใช้งาน ในฟังก์ชัน behaviors() สังเกตว่าเราจะยังไม่ได้กำหนด access เข้าไป

```php
<?php

// ...

public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
        ];
    }
```

ให้เราทำการเพิ่ม access Control เข้าไปและระบุ `roles` ให้เฉพาะ `ManageUser` เท่านั้น

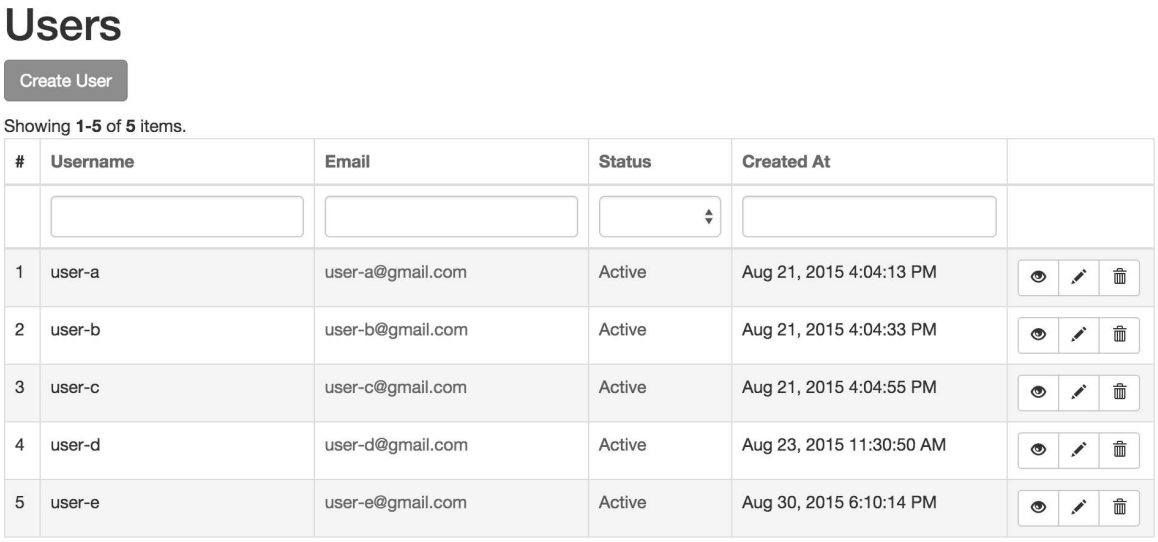
```php
<?php

use yii\filters\AccessControl;

// ...

public function behaviors()
    {
        return [
            'access'=>[
              'class'=>AccessControl::className(),
              'rules'=>[
                  [
                      //'actions'=>['index','create','view','update','de
                      'roles'=>['ManageUser'],
                      'allow'=>true
                  ]
              ]
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
        ];
    }
```

> สังเกตว่าผมไม่ได้ระบุ action เพราะให้มัน check ทุก action




คนที่ได้รับ role ManageUser หรือ Management ขึ้นไปจะสามารถเข้าใช้งาน Manage Users ได้

ทดลองสร้าง user ใหม่แล้วทดสอบ login & logout ดู




# สร้าง Role เพื่อจำกัดการ login ส่วน Backend

ในตอนนี้การใช้งาน ผู้ที่ได้รับ role Author จะยังสามารถล็อกอินเข้าหลังบ้านได้ แม้จะไม่สามารถ ทำอะไรได้ แต่ในความเป็นจริง ไม่ควรให้ user อื่นๆ ที่ไม่มีสิทธิ์ในส่วน

backend สามารถ login หลังบ้านได้เลย เพราะฉะนั้น เราจะทำการสร้าง role ขึ้นมาใหม่ 1 ตัวเพื่อเอาไว้เช็คสิทธิ์การเข้าใช้งาน backend

## สร้าง role loginToBackend

ทำการแก้ไข RBAC ใหม่โดยเพิ่ม permission `loginToBackend` เข้าไปที่ไฟล์ `console\controllers\RbacController.php`

```
<?php

public function init()
    {
      $auth = Yii::$app->authManager;
      $auth->removeAll();
      Console::output('Removing All! RBAC.....');

      $createPost = $auth->createPermission('createBlog');
      $createPost->description = 'สร้าง blog';
      $auth->add($createPost);

      $updatePost = $auth->createPermission('updateBlog');
      $updatePost->description = 'แก้ไข blog';
      $auth->add($updatePost);

      // เพิ่ม permission loginToBackend <<<-------------------------
      $loginToBackend = $auth->createPermission('loginToBackend');
      $loginToBackend->description = 'ล็อกอินเข้าใช้งานส่วน backend';
      $auth->add($loginToBackend);

      $manageUser = $auth->createRole('ManageUser');
      $manageUser->description = 'จัดการข้อมูลผู้ใช้งาน';
      $auth->add($manageUser);

      $author = $auth->createRole('Author');
      $author->description = 'การเขียนบทความ';
      $auth->add($author);

      $management = $auth->createRole('Management');
      $management->description = 'จัดการข้อมูลผู้ใช้งานและบทความ';
      $auth->add($management);

      $admin = $auth->createRole('Admin');
      $admin->description = 'สำหรับการดูแลระบบ';
      $auth->add($admin);

      $rule = new \common\rbac\AuthorRule;
      $auth->add($rule);

      $updateOwnPost = $auth->createPermission('updateOwnPost');
      $updateOwnPost->description = 'แก้ไขบทความตัวเอง';
      $updateOwnPost->ruleName = $rule->name;
      $auth->add($updateOwnPost);
```

```
        $auth->addChild($author,$createPost);
        $auth->addChild($updateOwnPost, $updatePost);
        $auth->addChild($author, $updateOwnPost);

        // addChild role ManageUser <<<-------------------------
        $auth->addChild($manageUser, $loginToBackend);

        $auth->addChild($management, $manageUser);
        $auth->addChild($management, $author);

        $auth->addChild($admin, $management);

        $auth->assign($admin, 1);
        $auth->assign($management, 2);
        $auth->assign($author, 3);
        $auth->assign($author, 4);

        Console::output('Success! RBAC roles has been added.');
    }
```

## ปรับปรุง actionLogin เพื่อเช็คสิทธิ์

ในส่วนของ login จะใช้ model ที่ชื่อว่า `LoginForm` อยู่ที่ `common\models\LoginForm.php` ให้เราดูที่ฟังก์ชัน login() ซึ่งของเดิมจะเป็นแบบนี้

```
<?php

// ...

public function login()
{
    if ($this->validate()) {
        return Yii::$app->user->login($this->getUser(), $this->rememberM
    } else {
        return false;
    }
}
```

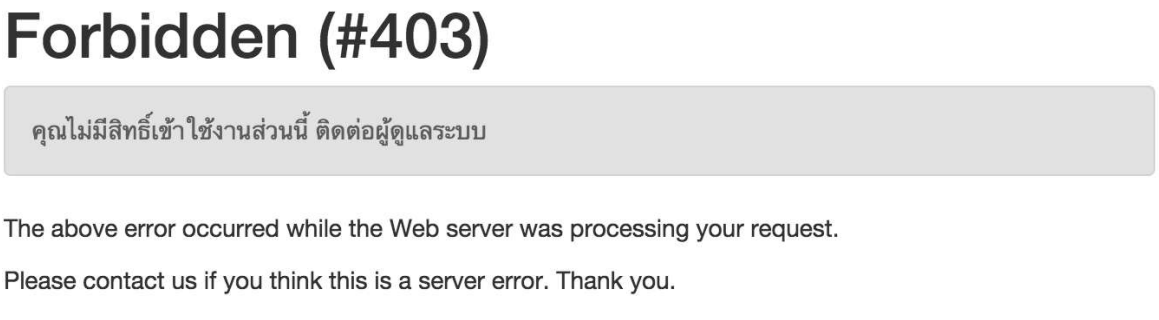
เราจะเปลี่ยนใหม่และเรียกใช้งาน `Yii::$app->user->can('loginToBackend')` เพื่อตรวจสอบว่า ผู้ใช้งานมีสิทธิ์เข้าใช้งานหลังบ้านหรือไม่

```
<?php
use yii\web\ForbiddenHttpException;

// ...
```

```
public function login()
{
    if (!$this->validate()) {
          return false;
    }

      if(Yii::$app->user->login($this->getUser(), $this->rememberMe ? 36
        if(Yii::$app->id=='app-backend' && !Yii::$app->user->can('loginT
            Yii::$app->user->logout();
            throw new ForbiddenHttpException('คุณไม่มีสิทธิ์เข้าใช้งานส่วนนี้ ติดต่อผู้ดู
        }
        return true;
      }
      return false;
```

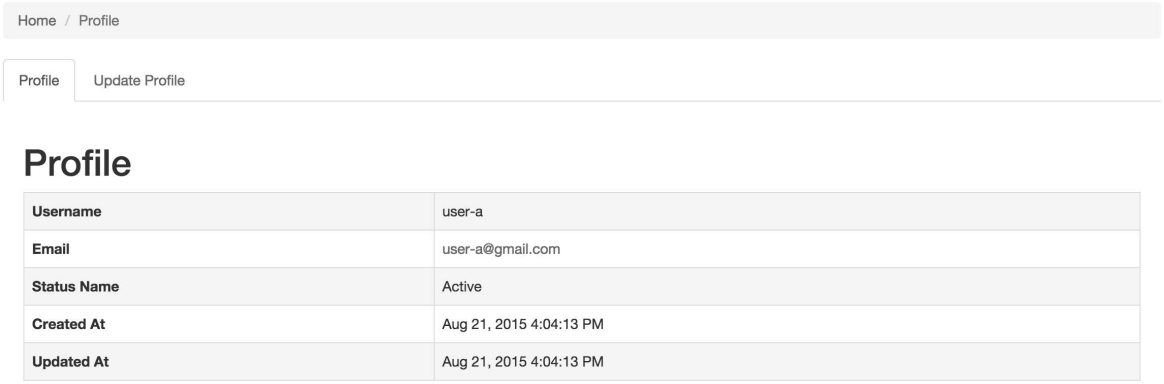
> `Yii::$app->id=='app-backend'` คือชื่อของ backend ถ้าหากเปลี่ยนต้องแก้ไขส่วนนี้ด้วย

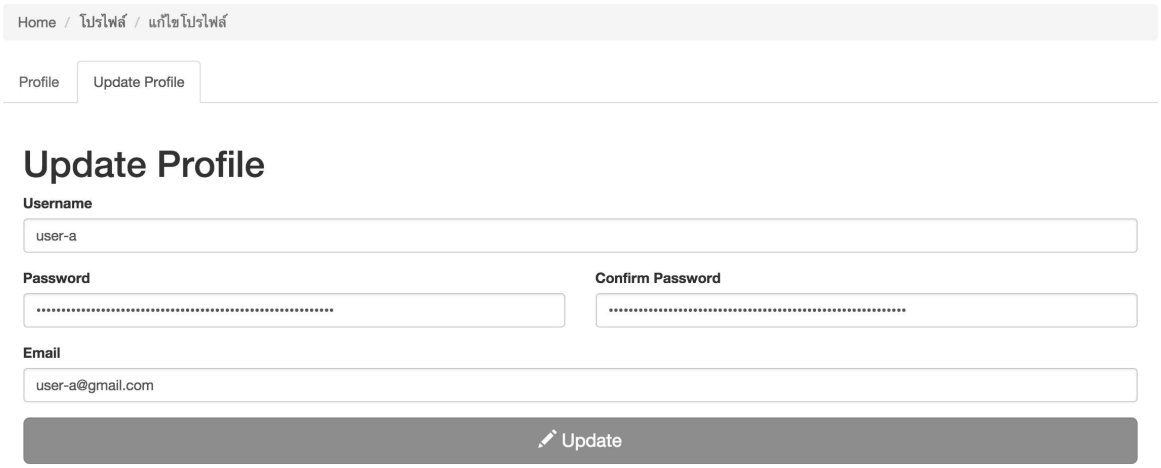
ให้ทดลอง login ด้วย user-c,user-d เมื่อทำการ login เข้า backend หากไม่มีสิทธิ์จะไม่สามารถเข้าใช้งานได้เลย ส่วน user-a,user-b จะเข้าใช้งานได้ปกติ



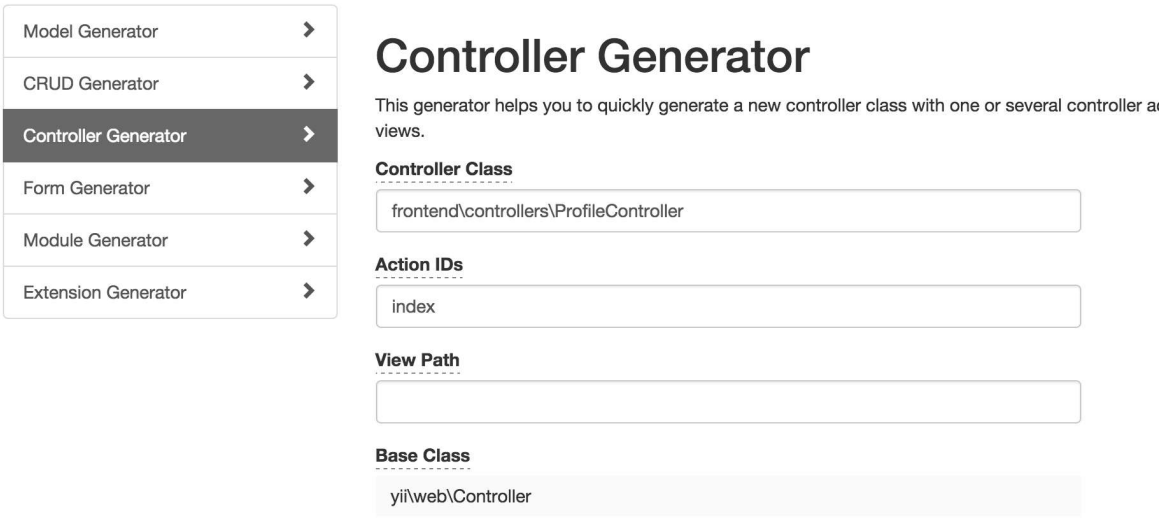

# สร้างหน้าแก้ไข Profile User

เราจะสร้างหน้าจัดการข้อมูลผู้ใช้งานเพื่อใช้ แก้ไขข้อมูลผู้ใช้งานและ user & password ดังนี้




เราจะสร้างไว้ที่ส่วน frontend ให้ไปที่ gii แล้วทำการ gii Controller ตามภาพด้านล่างนี้




# ปรับปรุง ProfileController.php

จากนั้นไปที่ไฟล์ ProfileController.php เราจะทำการสร้าง action ดังนี้

- findModel() เอาไว้ค้นข้อมูลผู้ใช้งานจาก model User ด้วย `Yii::$app->user->id`
- actionIndex() จะแสดงข้อมูลผู้ใช้งาน

- actionUpdate() เอาไว้แก้ไขข้อมูลผู้ใช้งาน
- behaviors() เราจะกำหนดสิทธิ์การเข้าใช้งานที่นี่ โดยระบุให้ user ที่มี role ตั้งแต่ Author ขึ้นไป สามารถเข้าใช้งานได้หมด

```
<?php

namespace frontend\controllers;

use Yii;
use common\models\User;
use frontend\models\ProfileSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;

/**
 * ProfileController implements the CRUD actions for User model.
 */
class ProfileController extends Controller
{
    public $layout = 'profile'; // set this to profile,profile2

    public function behaviors()
    {
        return [
          'access' => [
              'class' => AccessControl::className(),
              'rules' => [
                  [
                      'actions' => ['index','update','view'],
                      'allow' => true,
                      'roles' => ['@'],
                  ],
              ],
          ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['post'],
                ],
            ],
        ];
    }


    /**
     * Displays a single User model.
     * @param integer $id
     * @return mixed
     */
    public function actionIndex()
    {
        return $this->render('index', [
            'model' => $this->findModel(Yii::$app->user->id),
        ]);
```

```
}

/**
 * Updates an existing User model.
 * If update is successful, the browser will be redirected to the 'v
 * @param integer $id
 * @return mixed
 */
public function actionUpdate()
{
    $model = $this->findModel(Yii::$app->user->id);
    $model->password = $model->password_hash;
    $model->confirm_password = $model->password_hash;
    $oldPass = $model->password_hash;

    if ($model->load(Yii::$app->request->post()) && $model->validate

        if($oldPass!==$model->password){
          $model->setPassword($model->password);
        }

        if($model->save()){
          Yii::$app->getSession()->setFlash('success', 'บันทึกเสร็จเรียบร้อ
          return $this->redirect(['update']);
        }else{
          throw new NotFoundHttpException('พบข้อผิดพลาด!.');
        }

    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Finds the User model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param integer $id
 * @return User the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */
protected function findModel($id)
{
    if (($model = User::findOne($id)) !== null) {
        return $model;
    } else {
        throw new NotFoundHttpException('The requested page does not
    }
}
```

## สร้างไฟล์ view index,update,_form

## index.php

```
<?php

use yii\helpers\Html;
use yii\widgets\DetailView;

/* @var $this yii\web\View */
/* @var $model common\models\User */

$this->title = 'Profile';

$this->params['breadcrumbs'][] = $this->title;
?>
<div class="user-view">

    <h1><?= Html::encode($this->title) ?></h1>


    <?= DetailView::widget([
        'model' => $model,
        'attributes' => [

            'username',
            'email:email',
            'statusName',
            'created_at:dateTime',
            'updated_at:dateTime',
        ],
    ]) ?>

</div>
```

## update.php

```
<?php

use yii\helpers\Html;
use yii\helpers\ArrayHelper;

/* @var $this yii\web\View */
/* @var $model common\models\User */

$this->title = 'Update Profile';
$this->params['breadcrumbs'][] = ['label' => 'โปรไฟล์', 'url' => ['index']
$this->params['breadcrumbs'][] = Yii::t('app', 'แก้ไขโปรไฟล์');
?>

<div class="user-update">

    <h1><?= Html::encode($this->title) ?></h1>

    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
```

### _form.php

```
<?php

use yii\helpers\Html;
use yii\widgets\ActiveForm;

/* @var $this yii\web\View */
/* @var $model common\models\User */
/* @var $form yii\widgets\ActiveForm */
?>

<div class="user-form">

    <?php $form = ActiveForm::begin(); ?>

    <?= $form->field($model, 'username')->textInput(['maxlength' => true
    <div class="row">
      <div class="col-lg-6">
          <?= $form->field($model, 'password')->passwordInput(['maxlengt
      </div>
      <div class="col-lg-6">
        <?= $form->field($model, 'confirm_password')->passwordInput(['ma
      </div>
    </div>
    <?= $form->field($model, 'email')->textInput(['maxlength' => true])

    <div class="form-group">
        <?= Html::submitButton('<i class="glyphicon glyphicon-pencil"></
    </div>

    <?php ActiveForm::end(); ?>

</div>
```

## สร้าง layout profile

สร้าง layout ใหม่เพื่อใส่ tab ไว้ด้านบน สร้างไฟล์ไว้ที่ views/layouts/ profile.php

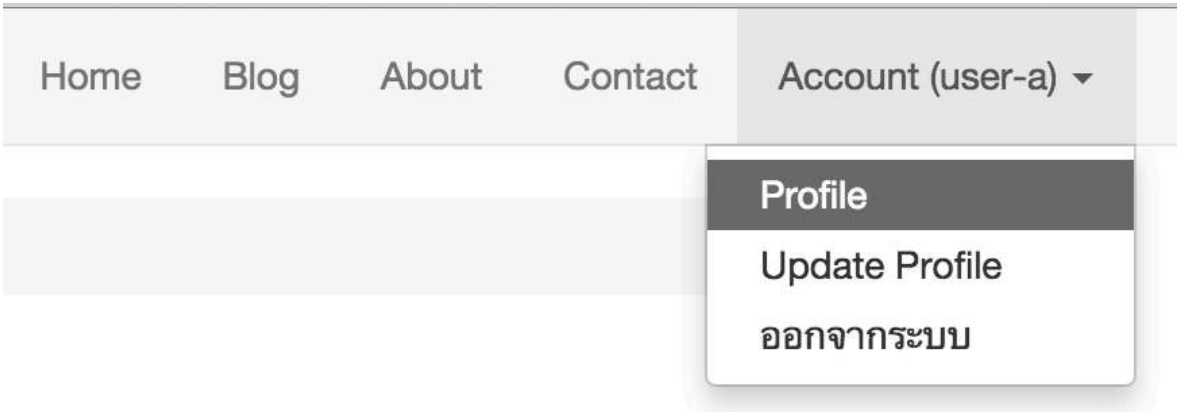
```
<?php
use yii\widgets\Breadcrumbs;
use common\widgets\Alert;
use yii\helpers\Html;
use yii\bootstrap\Nav;

 ?>
<?php $this->beginContent('@frontend/views/layouts/main.php'); ?>
```

```
<?php
echo Nav::widget([
    'items' => [
        [
            'label' => 'Profile',
            'url' => ['profile/index'],
        ],
        [
            'label' => 'Update Profile',
            'url' => ['profile/update'],
        ]
    ],
    'options' => ['class' =>'nav nav-tabs'], // set this to nav-tab to g
]);
 ?>

<div style="padding:20px;">

  <?php echo $content; ?>
</div>
```

> เราเรียกใช้งานใน layout profile ใน ProfileController `public $layout = 'profile';`

## เพิ่มเมนู Profile

เพิ่มเมนูในส่วนของเมนูหลักเพื่อให้สามารถคลิกดูรายละเอียดโปรไฟล์และแก้ไขโปรไฟล์ได้




ไปที่ไฟล์ views/layouts/main.php มองหาส่วน NavBar widget แล้วแก้ไข menu ของเดิม

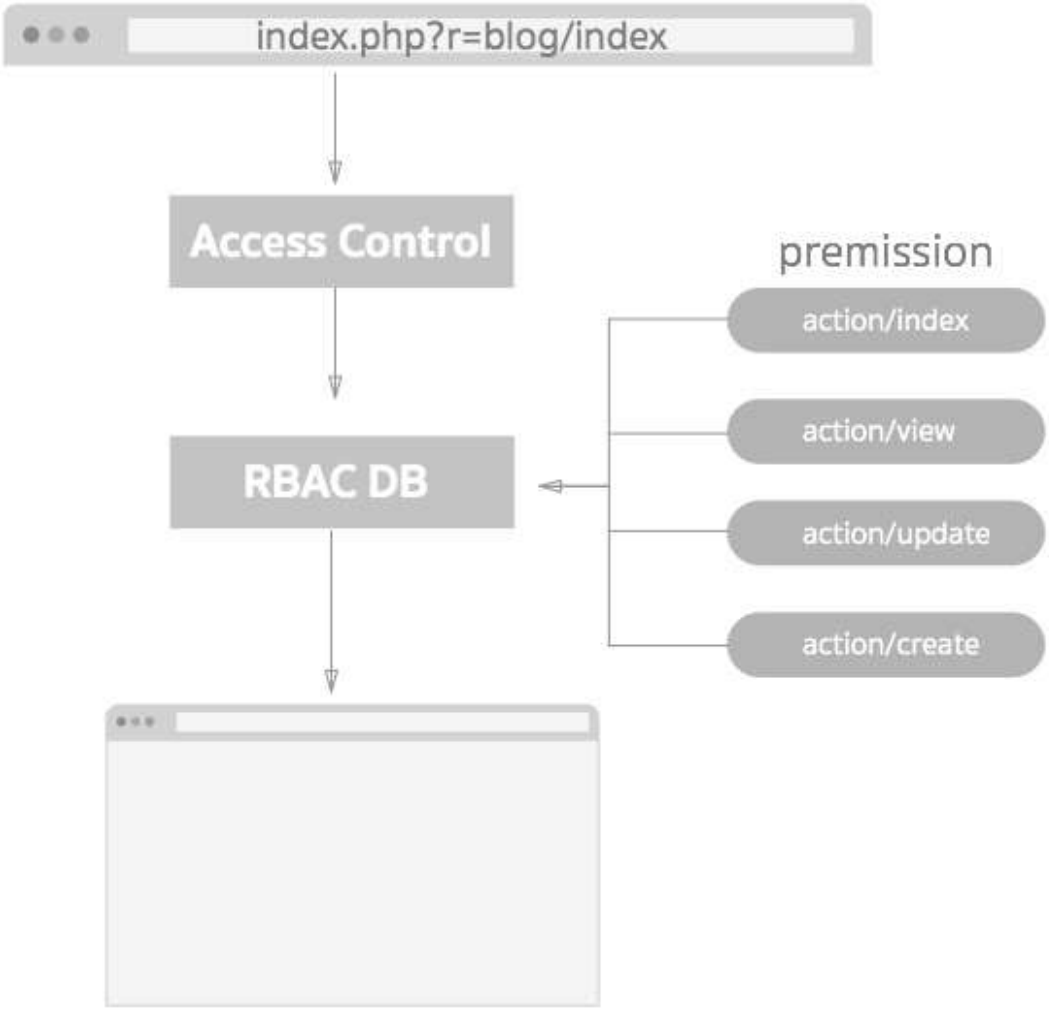
```
$menuItems[] = [
          'label' => 'Logout (' . Yii::$app->user->identity->username .
          'url' => ['/site/logout'],
          'linkOptions' => ['data-method' => 'post']
      ];
```

เปลี่ยนใหม่เป็น

```
$menuItems[] = [
'label' => 'Account (' . Yii::$app->user->identity->username . ')',
'items'=>[
    ['label' => 'Profile', 'url' => ['/profile/index']],
    ['label' => 'Update Profile', 'url' => ['/profile/update']],
    [ 'label'=>'ออกจากระบบ','url' => ['/site/logout'],'linkOptions' => ['
]
];
```

# การใช้งาน RBAC แบบเก็บข้อมูลตาม url

วิธีนี้เป็นวิธีที่ extension RBAC ส่วนใหญ่ใช้กันคือ ทำการสร้าง ชื่อ routing ที่มีทั้งหมดให้เป็น permission แล้ว ให้มาอยู่ภายใต้ role ที่เราได้สร้างไว้ เช่น `blog/index` , `blog/view` , `blog/update` , `blog/delete` ซึ่งจะใช้ชื่อนี้ในการสร้าง permission แล้วเวลาตรวจสอบก็เช็คว่า routing ปัจจุบันที่ใช้งาน มีสิทธิ์เข้าใช้งานหรือไม่




การตรวจสอบก็ใช้ Yii::$app->user->can() เหมือนเดิม ซึ่งนั่นหมายความว่าเราจะต้องมีจำนวน permission เท่ากับ action ทั้งหมดที่มี โดยส่วนตัวผมไม่ค่อยชอยวิธีนี้ ผมคิดว่าแค่ตัวอย่างที่เราได้ทำผ่านมาก็น่าจะเพียงพอ แต่จะขอแนะนำว่ามีวิธีการสร้างและใช้งานยังไง

## ปรับปรุง RBAC DB ใหม่

```
<?php

    public function up()
    {
      $auth = Yii::$app->authManager;
      $auth->removeAll();
      Console::output('Removing All! RBAC.....');

      $rule = new \common\rbac\AuthorRule;
      $auth->add($rule);

      // เพิ่มรายชื่อ action
      $index  = $auth->createPermission('blog/index');
```

```
$auth->add($index);
$view   = $auth->createPermission('blog/view');
$auth->add($view);
$create = $auth->createPermission('blog/create');
$auth->add($create);
$update = $auth->createPermission('blog/update');
$update->ruleName = $rule->name;
$auth->add($update);
$delete = $auth->createPermission('blog/delete');
$auth->add($delete);

$createPost = $auth->createPermission('createBlog');
$createPost->description = 'สร้าง blog';
$auth->add($createPost);

$updatePost = $auth->createPermission('updateBlog');
$updatePost->description = 'แก้ไข blog';
$auth->add($updatePost);

$loginToBackend = $auth->createPermission('loginToBackend');
$loginToBackend->description = 'ล็อกอินเข้าใช้งานส่วน backend';
$auth->add($loginToBackend);

$manageUser = $auth->createRole('ManageUser');
$manageUser->description = 'จัดการข้อมูลผู้ใช้งาน';
$auth->add($manageUser);

$author = $auth->createRole('Author');
$author->description = 'การเขียนบทความ';
$auth->add($author);

$management = $auth->createRole('Management');
$management->description = 'จัดการข้อมูลผู้ใช้งานและบทความ';
$auth->add($management);

$admin = $auth->createRole('Admin');
$admin->description = 'สำหรับการดูแลระบบ';
$auth->add($admin);

$updateOwnPost = $auth->createPermission('updateOwnPost');
$updateOwnPost->description = 'แก้ไขบทความตัวเอง';
$updateOwnPost->ruleName = $rule->name;
$auth->add($updateOwnPost);

// เพิ่มรายชื่อ action
$auth->addChild($author,$index);
$auth->addChild($author,$view);
$auth->addChild($author,$create);
$auth->addChild($author,$update);
$auth->addChild($management, $delete);

$auth->addChild($author,$createPost);
$auth->addChild($updateOwnPost, $updatePost);
$auth->addChild($author, $updateOwnPost);

$auth->addChild($manageUser, $loginToBackend);

$auth->addChild($management, $manageUser);
```

```
        $auth->addChild($management, $author);

        $auth->addChild($admin, $management);

        $auth->assign($admin, 1);
        $auth->assign($management, 2);
        $auth->assign($author, 3);
        //$auth->assign($author, 4);

        Console::output('Success! RBAC roles has been added.');
    }
```

## ปรับปรุง Access Controll

ในการเช็คว่าผู้ใช้งามมีสิทธิ์เข้าใช้งานหรือไม่ ก็ใช้ `Yii::$app->user->can()` เหมือนเดิม แต่ในส่วนนี้เราสร้าง permission ด้วยชื่อ ของ route เช่น `blog/index` มาสร้างเป็นชื่อ permission เลย เวลาตรวจสอบก็ใช้ `Yii::$app->controller->getRoute()` จะได้ค่า route ที่ใช้ปัจจุบันจากนั้นนำไปตรวจสอบกับ Yii::$app->user->can() อีกที

แก้ไขฟังก์ชัน behaviors() ใหม่ดังนี้

```
public function behaviors(){
      return [
                'verbs' => [
                    'class' => VerbFilter::className(),
                    'actions' => [
                        'delete' => ['post'],
                    ],
                ],
                'access'=>[
                    'class'=>AccessControl::className(),
                    'rules'=>[
                        [
                          'allow'=>true,
                          'actions'=>['update'],
                          'roles'=>['@'],
                          'matchCallback'=>function($rule,$action){
                            $model = $this->findModel(Yii::$app->request
                            if (\Yii::$app->user->can('blog/update',['mo
                                    return true;
                            }
                          }
                        ],
                        [
                            'allow'=>true,
                            'roles'=>['@'],
                            'matchCallback'=>function($rule,$action){
                              $currentRoute = Yii::$app->controller->get
                              if(Yii::$app->user->can($currentRoute)){
                                return true;
```

```
                                    }
                                }
                            ]
                        ]
                    ]
        ];
```

ทดลองเข้าใช้งาน

# สรุป

ผมคิดว่าคงจะทำให้เราเข้าใจหลักการสร้าง RBAC และสามารถนำไปใช้งานได้อย่างถูกต้องและมีประสิทธิภาพ สรุปขั้นตอนหลักๆ ของการสร้าง rbac มีดังนี้

- ออกแบบ rbac ว่ามีกี่กลุ่มอะไรบ้าง
- สร้าง rbac
- เรียกใช้งาน Access Control ที่ controller
- ทดสอบเข้าใช้งาน

แนะนำให้สร้างโปรเจคใหม่และค่อยทำตามที่ละ step จะเข้าใจและมองภาพออก หากติดปัญหาก็ฝากคำถามไว้ที่ comment ได้ครับ หรือจะไปที่ fanpage Yii2learning ก็ได้ครับ

ที่มา

- https://www.youtube.com/watch?v=7-jo8LKCnUk
- http://www.yiiframework.com/doc-2.0/guide-security-authorization.html
- http://www.satusoftware.com/frontend/web/site/post/7/Yii2-RBAC-Examples-Simple-Tutorial-with-User-Admin

Thanks

- อ. Prawee Wongsa

Download SourceCode @dixonsatit

ALSO ON **DIXONSATIT.GITHUB.IO**

### รวมคำสั่ง Query ใน Model ที่ใช้งานบ่อยๆ

7 years ago

รวมคำสั่ง Query ใน Model ที่ใช้งานบ่อยๆ | ...

### การ Register css,js ด้วย Client Script

2 years ago

การ Register css,js ด้วย Client Script | Yii 2 มีตัวช่วยในการจัดการพวก script file ...

### การใช้งาน CheckBo ...

a year ago

การใช้งาน CheckBoxLis การบันทึกข้อมูล | Yii2, Y Framework, Extension,

**เครือข่ายพันธมิตรของเรา**


---