

Episode 3 - Setting up the Role Based Access Control for our blog

Hello,

We have already set up our authManager component in previous episode. So now we can start creating roles, permissions and their mutual connections.

We want to have 3 roles: user, author and admin. The “user” role is the default User role after registration. “Author” is the privileged User role which can add new Posts or edit own Posts. And “admin” is the User role which can do everything what “author” can but also can update ALL Posts.

Roles and Permissions

To create roles, permissions and their mutual connections we need to create new `console/controllers/RbacController.php` class. With this class we will later initialize our RBAC rules. Create new `RbacController.php` file with this content:

```
<?php
namespace console\controllers;

use Yii;
use yii\console\Controller;

class RbacController extends Controller
{
    public function actionInit()
    {
        $auth = Yii::$app->authManager;

        $createPost = $auth->createPermission('createPost');
        $createPost->description = 'User can create a post';
```

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

```
$updatePost = $auth->createPermission('updatePost');
```

```
$updatePost->description = 'User can update post';
```

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

```
$user = $auth->createRole('user');
```

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

```
$author = $auth->createRole('author');
```

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

```
$admin = $auth->createRole('admin');
```

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

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

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

```
$auth->addChild($admin, $updatePost);
```

```
}
}
```

If do you want more informations about RBAC Configuration, I recommend you to take a look to official Yii guide [1](#).

Now we can initialize our RBAC configuration by running this command:

```
php yii rbac/init
```

After this, we should check our database if `auth_item` and `auth_item_child` tables are filled with rules. You should see (`auth_item`

<div><div></div><div></div><div></div></div>			name	type	description	rule_name	data	created_at	updated_at				
<input type="checkbox"/>		Edit		Copy		Delete	admin	1	NULL	NULL	NULL	1441374017	1441374017
<input type="checkbox"/>		Edit		Copy		Delete	author	1	NULL	NULL	NULL	1441374017	1441374017
<input type="checkbox"/>		Edit		Copy		Delete	createPost	2	User can create a post	NULL	NULL	1441374017	1441374017
<input type="checkbox"/>		Edit		Copy		Delete	updatePost	2	User can update post	NULL	NULL	1441374017	1441374017
<input type="checkbox"/>		Edit		Copy		Delete	user	1	NULL	NULL	NULL	1441374017	1441374017

table):

Also, we need to automatically assign “user” role to every new User whose registered to our blog. To do this, we need to update `frontend\models\SignupForm`s action `signup()`. We just need to add 3 new lines:

```
$auth = \Yii::$app->authManager;
$userRole = $auth->getRole('user');
$auth->assign($userRole, $user->getId());
```

Entire `signup()` method now should looks like:

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

    $user = new User();
    $user->username = $this->username;
    $user->email = $this->email;
```

```

        $user->setPassword($this->password);

        $user->generateAuthKey();

        if ($user->save()) {

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

            return $user;
        }

        return null;
    }
}

```

Rules

Rules add additional constraint to roles and permissions. A rule is a class extending from `yii\rbac\Rule`. It must implement the `execute()` method. In the hierarchy we've created previously author cannot edit his own post. Let's fix it. First we need a rule to verify that the user is the post author. To do this, we need to create `console/rbac/AuthorRule.php` file. Also, you will need to create the `rbac` folder in the console directory. `AuthorRule.php` should contain:

```

<?php

namespace console\rbac;

use yii\rbac\Rule;

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

```

```

public function execute($user, $item, $params)
{
    return isset($params['model']) ? $params['model']->createdBy->id == $user : false;
}
}

```

The rule above checks if the post is created by \$user. We'll create new permission `updateOwnPost` and associate the new rule with it.

To do this, we will create new `actionCreateAuthorRule()` method in our `RbacController` class (`console\controllers\RbacController.php`).

```

public function actionCreateAuthorRule()
{
    $auth = Yii::$app->authManager;

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

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

    $updatePost = $auth->getPermission('updatePost');

    $author = $auth->getRole('author');
}

```

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

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

```
}
```

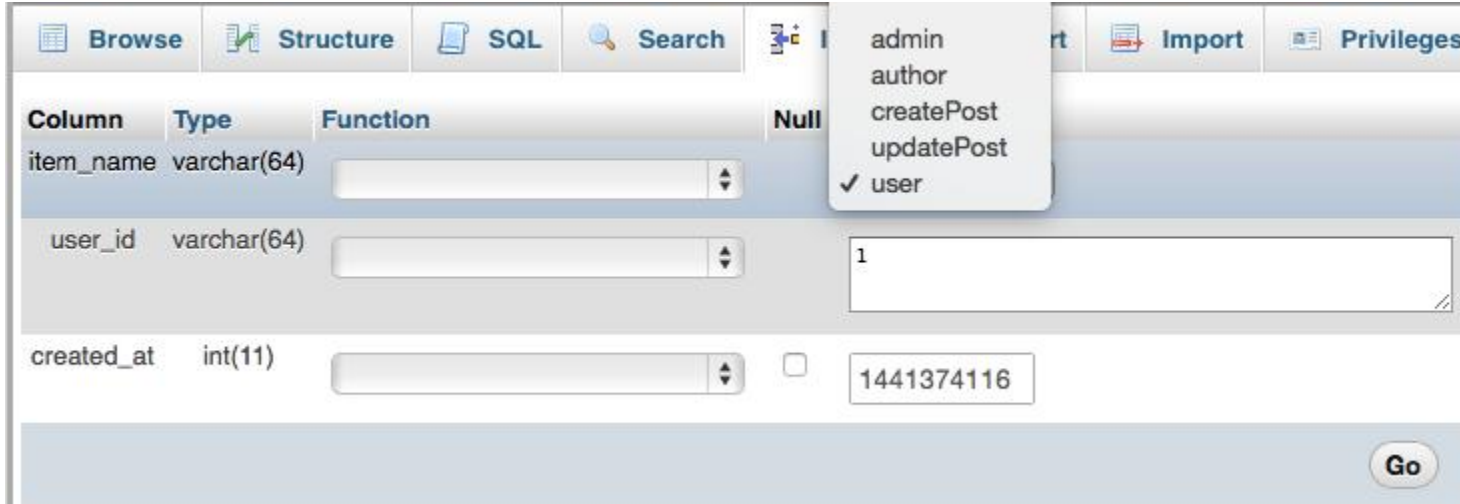
Then run:

```
php yii rbac/create-author-rule
```

Now we are done!

Trying it out

Now is the good time to sign up on your blog. After registration you should be automatically logged in and you should have assigned the “user” role in the `auth_assignment` table. Manually change your role to “admin”. You can do it by changing value “user” to “admin” in `item_name` column.



The screenshot shows the 'auth_assignment' table configuration in the Yii2 RBAC web interface. The table has columns for 'Column', 'Type', 'Function', and 'Null'. The 'item_name' column is set to 'varchar(64)' and has a dropdown menu open showing the following options: 'admin', 'author', 'createPost', 'updatePost', and 'user' (which is selected with a checkmark). The 'user_id' column is set to 'varchar(64)' and has a text input field containing the value '1'. The 'created_at' column is set to 'int(11)' and has a text input field containing the value '1441374116'. A 'Go' button is located at the bottom right of the form.

Access Check

To check if user is able to create new Post:

```
if (\Yii::$app->user->can('createPost')) {
```

```
}
```

To check if a user can update a post, we need to pass an extra parameter that is required by `AuthorRule` described before:

```
if (\Yii::$app->user->can('updatePost', ['model' => $post])) {
```

```
}
```

We will continue building our blog in next episode. If do you have any questions regarding to this episode, please write them below to the comments section.

Download files from this episode: [episode_03.zip](#).