

11. Login

Yii heeft al een (beperkte) ingebouwde login. In deze les wordt besproken hoe je dit systeem kan gebruiken.

Role based security

In *role based security* heeft elke gebruiker een naam en wachtwoord en heeft elke gebruiker ook een rol (role in het Engels). De rol bepaald dan wat de gebruiker kan. Je kunt de rol 'admin' hebben die bijvoorbeeld alles kan en mag. Daarnaast kun je de rol 'user' hebben die bijvoorbeeld alleen bepaalde overzichten kan maken. Een andere rol zou bijvoorbeeld 'docent' kunnen zijn en de docent kan dan bijvoorbeeld cijfers aanpassen. De rol leerling kan dan bijvoorbeeld alleen de cijfers raadplegen.

Model

De file in model 'User.php' bevat de data waarin de users staan. De users staan dus niet in de database. Dat kan wel maar dat kost iets meer tijd en voor het ontwikkelen is het lekker 'makkelijk' om de gebruikers even lokaal aan te maken.

Standaard staat dit in de *User.php* file

```
private static $users = [  
    '100' => [  
        'id' => '100',  
        'username' => 'admin',  
        'password' => 'admin',  
        'authKey' => 'test100key',  
        'accessToken' => '100-token',  
        'role' => 'admin',  
    ],  
    '101' => [  
        'id' => '101',  
        'username' => 'demo',  
        'password' => 'demo',  
        'authKey' => 'test101key',  
        'accessToken' => '101-token',  
    ],  
];
```

```
        'role' => 'user'
    ],
};
```

Regel 8 en 16 (de 'role') zijn toegevoegd aan de standaard inhoud van de file. In dit voorbeeld zijn er dus twee rollen.

Dan dien je nog een attribuut/property toe te voegen aan de User Model. Door deze als volgt aan te maken:

Voeg `public $role` toe bovenaan in de class User

```
class User extends \yii\base\BaseObject implements \yii\web\IdentityInterface
{
    public $id;
    public $username;
    public $password;
    public $authKey;
    public $accessToken;

    public $role; //Deze dien je nog toe te voegen aan de User.php
```

Dit is alles en hiermee kan je aanloggen in het Yii systeem.

Functionaliteiten afschermen

In elke controller staat bovenin een function behaviors(). Deze functie regelt de toegang tot de functies. Bijvoorbeeld:

```
public function behaviors()
{
    return [
        'verbs' => [
            'class' => VerbFilter::className(),
            'actions' => [
                'delete' => ['POST'],
            ],
        ],
        'access' => [
            'class' => AccessControl::className(),
            'rules' => [
                // when logged in, any user but yo need to be logged in!
```

```

        [ 'actions' => ['student','create', 'update-status'],
          'allow' => true,
          'roles' => ['@'],
        ],

        // when logged in and role = user
        [ 'actions' => [ 'rolspeler', 'update', 'call-student' ],
          'allow' => true,
          'roles' => ['@'],
          'matchCallback' => function ($rule, $action) {
            return (Yii::$app->user->identity->role == 'user');
          }
        ],

        // when logged in and role = admin
        [ 'actions' => [],
          'allow' => true,
          'roles' => ['@'],
          'matchCallback' => function ($rule, $action) {
            return (Yii::$app->user->identity->role == 'admin');
          }
        ],
      ],
    ],
  ],
];
}

```

Toelichting

Het eerste gedeelte is standaard en daarin wordt bepaald dat een delete alleen mag worden aangeroepen via een POST (en dus niet via een GET). Dit is niet van belang voor onze role based security.

```

'verbs' => [
  'class' => VerbFilter::className(),
  'actions' => [
    'delete' => ['POST'],
  ],
],

```

Het gedeelte daaronder is wel van belang. We zien drie blokjes (elk blokje begint met 'actions').

In het eerste blokje wordt bepaald dat als je bent aangemeld dat je dan de functie *student*, *create* en *update-status* mag gebruiken.

```
// when logged in, any user but yo need to be logged in!  
[ 'actions' => ['student','create', 'update-status'],  
  'allow' => true,  
  'roles' => ['@'],  
],
```

De regel `'roles'=> ['@']` betekent iedereen die is ingelogd.

Je kunt ook `'roles'=>['?']` gebruiken en dat betekent iedereen die niet is aangemeld (*guest user*).

Meer uitleg kan je vinden op: <https://www.yiiframework.com/doc/guide/2.0/en/security-authorization>

Het tweede blokje bepaald dat als je bent aangemeld met de rol *user*, je de functie *rolspeler*, *update* en *call-student* mag gebruiken.

```
// when logged in and role = user  
[ 'actions' => [ 'rolspeler', 'update', 'call-student' ],  
  'allow' => true,  
  'roles' => ['@'],  
  'matchCallback' => function ($rule, $action) {  
    return (Yii::$app->user->identity->role == 'user');  
  }  
],
```

Het derde blokje bepaald dat je alle acties (dus alle functies) kan gebruiken als he de rol *admin* hebt.

```
// when logged in and role = admin  
[ 'actions' => [],  
  'allow' => true,  
  'roles' => ['@'],  
  'matchCallback' => function ($rule, $action) {  
    return (Yii::$app->user->identity->role == 'admin');  
  }  
],
```

Opgave 1

Maak allereerst een menu voor jouw *coffee* app.

[image-1621708382447.png](#)

Kijk in [les 3](#) als je niet meer weet hoe je een menu moet maken.

Opgave 2

Maak nu een drie logins: één admin en twee medewerkers Adam en jane.

Gebruik de volgende gegevens.

id	username	password	authKey	accessToken	role
100	admin	admin123	admin-key	admin-token	admin
200	Adam	geheim	adam-key	adam-token	user
201	Jane	wachtwoord	jane-key	jane-token	user

Pas de *User.php* in *models* aan. Kijk naar het voorbeeld dat in deze les is beschreven.

Je kunt de template die hieronder staat gebruiken.

```
private static $users = [  
    '100' => [  
        'id' => '100',  
        'username' => '',  
        'password' => '',  
        'authKey' => '',  
        'accessToken' => '',  
        'role' => '',  
    ],  
    '200' => [  
        'id' => '200',  
        'username' => '',  
        'password' => '',  
        'authKey' => '',  
        'accessToken' => '',  
        'role' => ''  
    ],  
    '201' => [  
        'id' => '201',  
        'username' => '',  
        'password' => '',  
        'authKey' => '',  
        'accessToken' => '',  
        'role' => ''  
    ],  
]
```

```

'id' => '201',
'username' => '',
'password' => '',
'authKey' => '',
'accessToken' => '',
'role' => ''
],
];

```

Opgave 3

Wie mag wat? Pas de rechten aan aan de hand van het volgende schema.

	Create	Read (index)	Update	Delete
Medewerkers	admin	medewerker/admin ²⁾	admin	admin
Menus	admin	iedereen ¹⁾	admin	admin
Bestelling	medewerker	iedereen ¹⁾	medewerker/admin ²⁾	medewerker/admin ²⁾

¹⁾*Iedereen* betekent admin, medewerker of iemand die *niet* is aangemeld.

²⁾*medewerker/admin*, kan ook worden gezien als iemand die is aangemeld.

Bovenstaande tabel kun je ook zo vertalen:

Een **admin** kan

Medewerker	Read (index)	Create	Update	Delete
Menus	Read (index)	Create	Update	Delete
Bestelling	Read (index)	Create	Update	Delete

Een **medewerker/user** kan

Medewerker	Read (index)			
Menus	Read (index)			
Bestelling	Read (index)	Create	Update	Delete

Een **niet-ingelogde gebruiker/guest** kan

Medewerker				
Menus	Read (index)			
Bestelling	Read (index)			

We gaan de toegang regelen voor de *create*, *index*, *update* en *delete* van de *medewerker*.

Ga naar de file *MedewerkerController.php* in de *controllers* en voeg de volgende code toe:

```
class MedewerkerController extends Controller

{
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
            'access' => [
                'class' => AccessControl::className(),
                'rules' => [

                    // when logged in as admin
                    [ 'actions' => ['create', 'update', 'delete'],
                      'allow' => true,
                      'roles' => ['@'],
                      'matchCallback' => function ($rule, $action) {
                          return (Yii::$app->user->identity->role == 'admin' );
                      }
                    ],

                    // when logged in, any user but you need to be logged in!
                    [ 'actions' => ['index'],
                      'allow' => true,
                      'roles' => ['@'],
```

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

....

....

Maak nu zelf de toegangscontrole voor de *MenuController.php* en de *BestellingController* af.

More info

In

<https://www.roc.ovh/books/yii/page/login-via-db>

staat kort beschreven hoe je de gebruikers in de database kan vastleggen. Voor het examen is dit niet noodzakelijk.

--

Revision #13

Created 17 May 2021 19:32:59 by Max

Updated 7 July 2021 14:48:37 by Max