Yii2. rbac

sd@freematiq.com
https://github.com/freematiq/php

isGuest

Первая проверка на доступ в шаблонах или коде:

```
if (Yii::$app->user->isGuest)
    // неавторизован
else
    // авторизован
```

Hooks

beforeAction($action)

Исполняется перед исполнением действия (action) контроллера. Метод должен вернуть boolean, означающий, можно ли запускать само действие

afterAction($action, $result)

Вызывается после исполнения действия, но перед тем как выводится View.

Простая проверка

```
public function beforeAction($action)  {
        // базовая проверка на csrf
          $parentAllowed = parent::beforeAction($action);
          $meAllowed = !Yii::$app->user->isGuest;
          return $parentAllowed and $meAllowed;
    }
```

HttpExceptions

BadRequestHttpException
400 Bad Request

ConflictHttpException
409 Conflict

ForbiddenHttpException
403 Forbidden

GoneHttpException
410 Gone

MethodNotAllowedHttpException
405 Method Not Allowed

NotAcceptableHttpException
406 Not Acceptable

NotFoundHttpException
404 Not Found

TooManyRequestsHttpException
429 Too Many Requests

UnauthorizedHttpException
401 Unauthorized

UnsupportedMediaTypeHttpException
415 Unsupported Media Type

Простая проверка

```
if (Yii::$app->user->isGuest)
        throw new ForbiddenHttpException;
```

```
Yii::$app->user->can('student');
```

В контроллере

```
public function actionUpdate($id)
{
   $model = $this->findModel($id);
   if (Yii::$app->user->id == $model->user_id || \Yii::$app->user->can('manageArticles')) {
        // ...
   } else {
        throw new ForbiddenHttpException('You are not allowed to edit this article.');
       }
 }
```

Behaviours

Метод behaviors() должен вернуть массив конфигураций Поведений.

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

Action delete выполняется только для POST запроса

VerbFilter

Запрещает или разрешает доступ к action на основе HTTP метода запроса. Например, можно разрешить только POST запросы

http://www.yiiframework.com/doc-2.0/yii-filters-verbfilter.html

AccessControl

Фильтр предотвращает или разрешает доступ к действиям на наборе правил. Имеет очень гибкий и выразительный синтаксис и может быть использован только он один. Позволяет проверять доступ на основании HTTP метода, статуса авторизации пользователя или его роли, IP. Можно написать свою функцию проверки. В случае отказа 403.

http://www.yiiframework.com/doc-2.0/yii-filters-accesscontrol.html

Правила

'allow' => true, // или false
'actions' => ['действия', 'как', 'массив''],
'roles' => ['Роли', '?', '@'],
'ips' => ['IP', '*'],
'verbs' => ['HTTP Method'],

@ - аутентифицированный пользователь
? - гость (неаутентифицированный)

RateLimiter

Запрещает доступ пользователям, кто превысил свой лимит запросов в интервал времени. Чтобы работало - пользователю нужно реализовать интерфейс \yii\filters\RateLimitInterface.

http://www.yiiframework.com/doc-2.0/yii-filters-ratelimitinterface.html

Role Based Access Control, RBAC

Пользователям присваиваются роли, на основании которых осуществляется проверка их действий

rbac migrations

в config/web.php, config/console.php настраиваем компонент

```
'authManager' => [
    'class' => 'yii\rbac\DbManager',
    'defaultRoles' => ['guest'],
],
```

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

Домашнее задание

Добавить RBAC в проект