FABIO CIRRUTO

CONFIGURARE YII PER GLI OAUTH2 TOKENS

INSTALLARE LA DEPENDENCY

- Raggiungendo questo link https://github.com/filsh/yii2-oauth2-server troverete le informazioni per aggiungere la dependency al composer json sotto la chiave require.
- Collegarsi da terminare alla root del progetto ed eseguire un composer update

PREPARARE IL CONFIG

- A l’indice 0 del config settare il time zone Europeo, per far si che i token abbiano la durata corretta, inserendo questa chiave: 'timeZone' => “Europe/Rome"
- Assicurati che sotto la chiave “components”->”user” l’identityClass abbia lo stesso model che usi per generare i token altrimenti non funzionerà l’autentificazione
- Segui le restanti indicazioni per terminare la configurazione

```
    'db[rm' => $db[rm,
    'urlManager' => [
        'enablePrettyUrl' => true,
        'showScriptName' => false,
        'rules' => [
            'POST oauth2/<action:\w+>' => 'oauth2/rest/<action>',
        ],
    ],
],
'params' => $params,
'modules' => [
    'oauth2' => [
        'class' => 'filsh\yii2\oauth2server\Module',
        'tokenParamName' => 'accessToken',
        'tokenAccessLifetime' => 3600 * 24,
        'storageMap' => [
            'user_credentials' => 'app\models\Customers',
        ],
        'grantTypes' => [
            'user_credentials' => [
                'class' => 'OAuth2\GrantType\UserCredentials',
            ],
            'refresh_token' => [
                'class' => 'OAuth2\GrantType\RefreshToken',
                'always_issue_new_refresh_token' => true
            ]
        ],
        'components' => [
            'request' => function () {
                return \filsh\yii2\oauth2server\Request::createFromGlobals();
            },
            'response' => [
                'class' => \filsh\yii2\oauth2server\Response::class,
                                'format' => yii\web\Response::FORMAT_JSON,
                                'charset' => 'UTF-8',
            ],
        ],
    ]
],
```

PREPARARE IL DB

- Da terminale collegati alla root del progetto e lancia il comando per la migrazione del database, il comando sarà simile a “yii migrate –migrationPath=@vendor/filsh/yii2-oauth2-server/migrations”
- Probabilmente il comando andrà in errore quindi apri il file della migrazione che si trova in vendor/filsh/yii2-oauth2-server/migrations e inserisci alla funzione primaryKey a null il parmaetro che riceve in input e esegui il comando
- Accedi al db e nella tabella “oauth_clients” modifica i campi “client_id” e “client_secret” per inserire valori che si addicano al tuo progetto

```
11
12  public function primaryKey($columns = null) {
13      return 'PRIMARY KEY (' . $this->db->getQueryBuilder()->buildColumns($columns
            ) . ')';
14  }
15
```

COFNIGURARE IL MODEL

- Nel model scelto per l'autenticazione importa sotto Yii anche il modulo IdentityInterface inserendo queste riga: "use yii\web\IdentityInterface;"
- Estendi il modulo a tutto il model inserendo: "extends \yii\db\ActiveRecord implements IdentityInterface, \OAuth2\Storage\UserCredentialsInterface"
- Importa tutti i metodi necessari a rendere il tuo model confermo alla Identity interface. Ricorda di adeguare i metodi in base ai campi della tua tabella di riferimento

```php
public static function findIdentityByAccessToken($token, $type = null){
    /** @var \filsh\yii2\oauth2server\Module $module */
    $module = Yii::$app->getModule('oauth2');
    $token = $module->getServer()->getResourceController()->getToken();
    return !empty($token['user_id'])
                ? static::findIdentity($token['user_id'])
                : null;
}

public function checkUserCredentials($username, $password){
    $user = static::findByUsername($username);
    if (empty($user)) {
        return false;
    }
    return $user->validatePassword($password);
}

public function getUserDetails($username){
    $user = static::findByUsername($username);
    return ['user_id' => $user->getId()];
}

public static function findByUsername($username){
    return static::findOne(['customerEmail' => $username]);
}

public function getAuthKey() {
    return $this->auth_key;
}

public function getId(){
    return $this->getPrimaryKey();
}

public function validateAuthKey($authKey) {
    return $this->getAuthKey() === $authKey;
}

public static function findIdentity($id) {
    return static::findOne(['customerid'=>$id]);
}

public function validatePassword($password){
    return $this->getPassword() == md5($password);
}

public function getPassword(){
    return $this->customerPasswd;
}
```

IL FILE HTACCESS

- Yii potrebbe aver problemi a leggere le informazioni dal header della tua richiesta
- Collegati alla root del tuo progetto e configurare il file htaccess della root come nella foto che vedi acconto

```
SetEnvIf Authorization .+ HTTP_AUTHORIZATION=$0

<IfModule mod_rewrite.c>
    Options +FollowSymlinks
    RewriteEngine On
</IfModule>

<IfModule mod_rewrite.c>
    RewriteCond %{REQUEST_URI} ^/.*
    RewriteRule ^(.*)$ web/$1 [L]

    RewriteCond %{REQUEST_URI} !^/web/
    RewriteCond %{REQUEST_FILENAME} !-f [OR]
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^.*$ web/index.php
</IfModule>
```

CUSTOMIZZARE I MESSAGGI DI ERRORE

- Crea una cartella util e importa la classe ForbiddenFilter
- Questa classe serve per customizzare i messaggi di errore

```php
<?php


namespace app\util;

use Yii;
use yii\base\Controller;
use filsh\yii2\oauth2server\Module;
use filsh\yii2\oauth2server\exceptions\HttpException;

class ForbiddenFilter extends \yii\base\Behavior
{
    /**
     * @inheritdoc
     */
    public function events()
    {
        return [Controller::EVENT_AFTER_ACTION => 'afterAction'];
    }

    /**
     * @param ActionEvent $event
     * @return boolean
     * @throws HttpException when the request method is not allowed.
     */
    public function afterAction($event)
    {
        $response = Yii::$app->response;

        $isValid = !$response->isForbidden;
        if (!$isValid) {
            throw new HttpException(403, "Permesso negato", []);
        }
    }

    protected function getErrorMessage(\OAuth2\Response $response)
    {

        $message = Module::t('common', $response->getParameter('error_description'));
        if ($message === null) {
            $message = Module::t('common', 'An internal server error occurred.');
        }
        return $message;
    }
}
```

IL CONTROLLER PROTETTO DA TOKEN

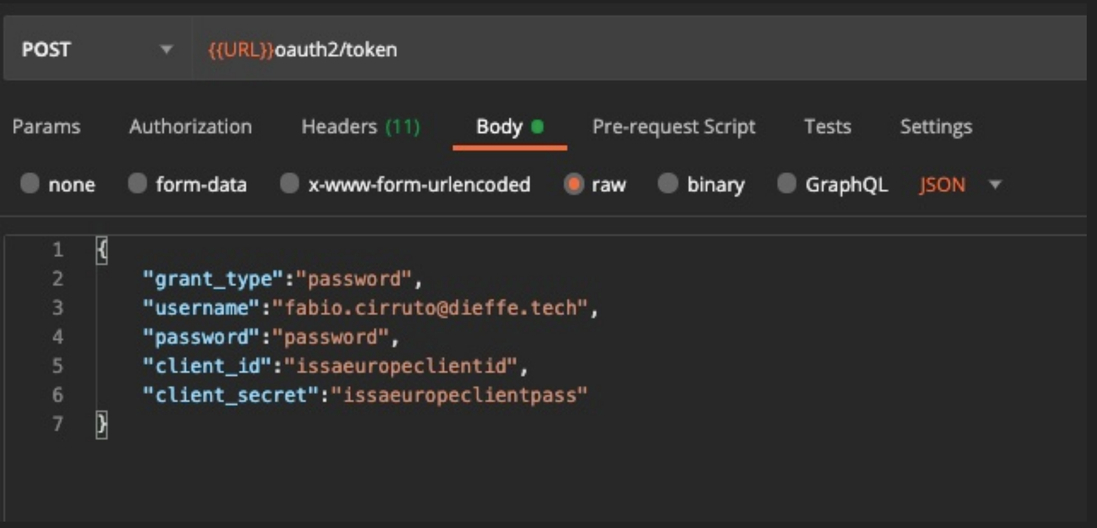
- Nel controller che vuoi progettare importa le dipendenze necessarie ad abilitare il tutto e configura il behaviors come nella foto che vedi accanto
- Nei metodi puoi usare questo metodo: “$customer = Customers::findIdentityByAccessToken(“”);” per avere un’istanza della classe che gestisce la login per l’utente che effettua la richiesta

```php
use yii\filters\auth\HttpBearerAuth;
use yii\filters\auth\QueryParamAuth;
use filsh\yii2\oauth2server\filters\ErrorToExceptionFilter;
use filsh\yii2\oauth2server\filters\auth\CompositeAuth;

class AuthController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return ArrayHelper::merge(parent::behaviors(), [
            'authenticator' => [
                'class' => CompositeAuth::className(),
                'authMethods' => [
                    ['class' => HttpBearerAuth::className()],
                    ['class' => QueryParamAuth::className(), 'tokenParam' => 'accessToken'],
                ]
            ],
            'exceptionFilter' => [
                'class' => ErrorToExceptionFilter::className()
            ],
        ]);
    }
```

BASIC TOKEN

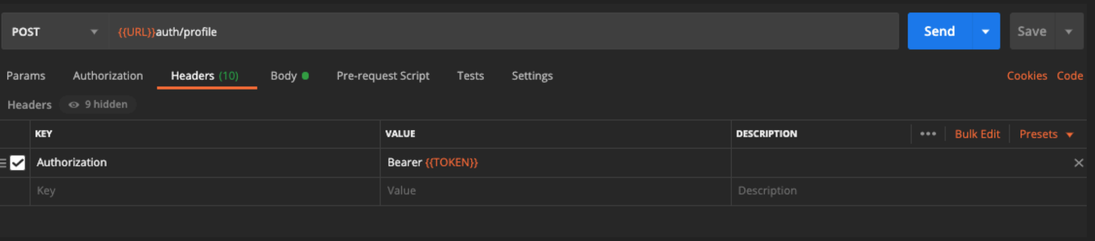
- Se hai delle chiamate che non neccitano la login ma vuoi proteggerle comunque con il token puoi fare una chiamata basic
- Il token è generato dal base 64 di TUA_USERNAME:TUA_PASSWORD

```php
use yii\filters\auth\CompositeAuth;
use yii\filters\auth\HttpBearerAuth;
use yii\filters\auth\QueryParamAuth;
use filsh\yii2\oauth2server\filters\ErrorToExceptionFilter;


class TrainersController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors() {
        return ArrayHelper::merge(parent::behaviors(), [
                    'authenticator' => [
                        'class' => CompositeAuth::className(),
                        'authMethods' => [
                            [
                                'class' => \yii\filters\auth\HttpBasicAuth::className(),
                                'auth' => function ($username, $password) {

                                    if ($username == Yii::$app->params["WEBSERVICE_USERNAME"] && $password == Yii::$app->params["WEBSERVICE_PASSWORD"]) {
                                        return new \app\models\User();
                                    }

                                    return null;
                                }
                            ]
                        ]
                    ],
                    'exceptionFilter' => [
                        'class' => ForbiddenFilter::className()
                    ],
        ]);
    }
```



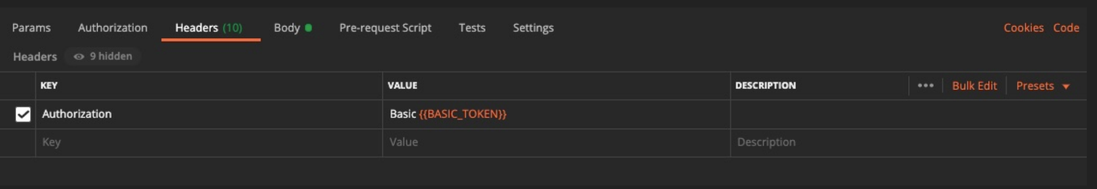
FABIO CIRRUTO

LA GENERAZIONE DEL TOKEN



FABIO CIRRUTO

COME PASSARE IL TOKEN UTENTE



FABIO CIRRUTO

UTILIZZO DI UN TOKEN STATICO