Django OAuth 2.0 App Documentation

Release 0.1.0

John Wehr

July 01, 2015

Contents

CHAPTER 1

About

- See http://hiidef.github.com/oauth2app for documentation.
- See https://github.com/hiidef/oauth2app for source code.
- Based on http://code.google.com/p/django-oauth2
- Support for OAuth 2.0 draft 16, http://tools.ietf.org/html/draft-ietf-oauth-v2-16

CHAPTER 2

Installation

If easy_install is available, you can use:

```
easy_install https://github.com/hiidef/oauth2app/tarball/master
```

Introduction

The oauth2app module helps Django site operators provide an OAuth 2.0 interface. The module is registered as an application.

In settings.py, add ‘oauth2app’ to INSTALLED_APPS.

```
INSTALLED_APPS = (
    ...,
    'oauth2app'
)
```

Sync the DB models.

```
python manage.py syncdb
```

In urls.py, add /oauth2/authorize and /oauth2/token views to a new or existing app.

```
urlpatterns += patterns('',
    (r'^oauth2/missing_redirect_uri/?$',   'mysite.oauth2.views.missing_redirect_uri'),
    (r'^oauth2/authorize/?$',                'mysite.oauth2.views.authorize'),
    (r'^oauth2/token/?$',                    'oauth2app.token.handler'),
)
```

Create client models.

```
from oauth2app.models import Client

Client.objects.create(
    name="My Sample OAuth 2.0 Client",
    user=user)
```

Create authorize and missing_redirect_uri handlers.

```
from django.shortcuts import render_to_response
from django.http import HttpResponseRedirect
from django.template import RequestContext
from django.contrib.auth.decorators import login_required
from oauth2app.authorize import Authorizer, MissingRedirectURI, AuthorizationException
from django import forms

class AuthorizeForm(forms.Form):
    pass

@login_required
def missing_redirect_uri(request):
    return render_to_response(
        'oauth2/missing_redirect_uri.html',
        {},
        RequestContext(request))

@login_required
def authorize(request):
    authorizer = Authorizer()
    try:
        authorizer.validate(request)
    except MissingRedirectURI, e:
        return HttpResponseRedirect("/oauth2/missing_redirect_uri")
    except AuthorizationException, e:
        # The request is malformed or invalid. Automatically
        # redirects to the provided redirect URL.
        return authorizer.error_redirect()
    if request.method == 'GET':
        template = {}
        # Use any form, make sure it has CSRF protections.
        template["form"] = AuthorizeForm()
        # Appends the original OAuth2 parameters.
        template["form_action"] = '/oauth2/authorize?%s' % authorizer.query_string
        return render_to_response(
            'oauth2/authorize.html',
            template,
            RequestContext(request))
    elif request.method == 'POST':
        form = AuthorizeForm(request.POST)
        if form.is_valid():
            if request.POST.get("connect") == "Yes":
                # User agrees. Redirect to redirect_uri with success params.
                return authorizer.grant_redirect()
            else:
                # User refuses. Redirect to redirect_uri with error params.
                return authorizer.error_redirect()
    return HttpResponseRedirect("/")
```

Authenticate requests.

```
from oauth2app.authenticate import Authenticator, AuthenticationException
from django.http import HttpResponse

def test(request):
    authenticator = Authenticator()
    try:
        # Validate the request.
        authenticator.validate(request)
    except AuthenticationException:
        # Return an error response.
        return authenticator.error_response(content="You didn't authenticate.")
    username = authenticator.user.username
    return HttpResponse(content="Hi %s, You authenticated!" % username)
```

If you want to authenticate JSON requests try the JSONAuthenticator.

```
from oauth2app.authenticate import JSONAuthenticator, AuthenticationException

def test(request):
    authenticator = JSONAuthenticator()
    try:
        # Validate the request.
        authenticator.validate(request)
    except AuthenticationException:
        # Return a JSON encoded error response.
        return authenticator.error_response()
    username = authenticator.user.userame
    # Return a JSON encoded response.
    return authenticator.response({"username":username})
```

CHAPTER 4

Examples

An example Django project demonstrating client and server functionality is available in the repository.

https://github.com/hiidef/oauth2app/tree/develop/examples/mysite

Usage

5.1 Authorization

5.1.1 Authorizer

The authorizer grants access tokens and authentication codes via query string parameters and URI fragments sent to redirect URIs. Optionally a “scope” kwarg of one or more AccessRange objects can be passed to verify that granted tokens can only be used to access specific scopes.

In the event of an error the Authorizer:error_response() method will return a redirect response to the client’s redirect_uri with information on the error passed as query string parameters.

If a request is authorized, Authorizer:grant_response() will serialize an object into a JSON response will return a redirect response to the client’s redirect_uri with information on the authorization code passed as query string parameters (response_type CODE) or access token passed as URI fragments.

```
from oauth2app.authorize import Authorizer, MissingRedirectURI, AuthorizationException
from oauth2app.models import AccessRange

@login_required
def authorize(request):
    scope = AccessRange.objects.get(key="last_login")
    authorizer = Authorizer(scope=scope)
    try:
        # Validate the request.
        authorizer.validate(request)
    except MissingRedirectURI, e:
        # No redirect_uri was specified.
        return HttpResponseRedirect("/oauth2/missing_redirect_uri")
    except AuthorizationException, e:
        # The request is malformed or invalid. Redirect to redirect_uri with error params.
        return authorizer.error_redirect()
    if request.method == 'GET':
        template = {}
        # Use any form, make sure it has CSRF protections.
        template["form"] = AuthorizeForm()
        # Appends the original OAuth2 parameters.
        template["form_action"] = '/oauth2/authorize?%s' % authorizer.query_string
        return render_to_response(
            'oauth2/authorize.html',
            template,
            RequestContext(request))
    elif request.method == 'POST':
        form = AuthorizeForm(request.POST)
        if form.is_valid():
            if request.POST.get("connect") == "Yes":
                # User agrees. Redirect to redirect_uri with success params.
                return authorizer.grant_redirect()
            else:
                # User refuses. Redirect to redirect_uri with error params.
                return authorizer.error_redirect()
    return HttpResponseRedirect("/")
```

5.1.2 Module Reference

5.2 Authentication

5.2.1 Authenticator

The Authenticator object verifies that a request has proper authentication credentials. Optionally a "scope" kwarg of one or more AccessRange objects can be passed to verify that tokens used to access this resource are authorized to access the specific scope.

In the event of an error the Authenticator:error_response() method will wrap an error response with the appropriate OAuth2 headers.

```
from oauth2app.authenticate import Authenticator, AuthenticationException
from oauth2app.models import AccessRange
from django.http import HttpResponse

def test(request):
    scope = AccessRange.objects.get(key="test_scope")
    authenticator = Authenticator(scope=scope)
    try:
        # Validate the request.
        authenticator.validate(request)
    except AuthenticationException:
        # Return an error response.
        return authenticator.error_response(content="You didn't authenticate.")
    username = authenticator.user.username
    return HttpResponse(content="Hi %s, You authenticated!" % username)
```

5.2.2 JSONAuthenticator

The JSONAuthenticator adds convenience methods and supports an optional callback request parameter for use with JSONP requests.

In the event of an error the JSONAuthenticator:error_response() method will return a JSON formatted error HttpResponse.

JSONAuthenticator:response() will serialize an object and return a formatted HttpResponse.

```
from oauth2app.authenticate import JSONAuthenticator, AuthenticationException

def test(request):
    authenticator = JSONAuthenticator()
    try:
        # Validate the request.
        authenticator.validate(request)
    except AuthenticationException:
        # Return a JSON encoded error response.
        return authenticator.error_response()
    username = authenticator.user.userame
    # Return a JSON encoded response.
    return authenticator.response({"username":username})
```

5.2.3 Module Reference

OAuth 2.0 Authentication

exception `oauth2app.authenticate.`**`AuthenticationException`**
Authentication exception base class.

class `oauth2app.authenticate.`**`Authenticator`**(*scope=None*, *authentication_method=1*)
Django HttpRequest authenticator. Checks a request for valid credentials and scope.

Kwargs:

- *scope:* An iterable of oauth2app.models.AccessRange objects representing the scope the authenticator will authenticate. *Default None*
- *authentication_method:* Accepted authentication methods. Possible values are: oauth2app.consts.MAC, oauth2app.consts.BEARER, oauth2app.consts.MAC | oauth2app.consts.BEARER, *Default oauth2app.consts.BEARER*

`access_token` = None

`attempted_validation` = False

`auth_type` = None

`auth_value` = None

`client`
The client associated with the valid access token.

oauth2app.models.Client object

`error` = None

`error_response`(*content=''*, *mimetype=None*, *content_type='text/html'*)
Error response generator. Returns a Django HttpResponse with status 401 and the approproate headers set. See Django documentation for details. https://docs.djangoproject.com/en/dev/ref/request-response/#django.http.HttpResponse.__init__

Kwargs:

- *content:* See Django docs. *Default ''*
- *mimetype:* See Django docs. *Default None*
- *content_type:* See Django docs. *Default DEFAULT_CONTENT_TYPE*

`scope`
The client scope associated with the valid access token.

QuerySet of AccessRange objects.

`user`
The user associated with the valid access token.

django.auth.User object

`valid` = **False**

`validate`(*request*)
Validate the request. Raises an AuthenticationException if the request fails authentication.

Args:

- *request:* Django HttpRequest object.

Returns None

exception `oauth2app.authenticate.`**`InsufficientScope`**
The request requires higher privileges than provided by the access token.

`error` = **'insufficient_scope'**

exception `oauth2app.authenticate.`**`InvalidRequest`**
The request is missing a required parameter, includes an unsupported parameter or parameter value, repeats the same parameter, uses more than one method for including an access token, or is otherwise malformed.

`error` = **'invalid_request'**

exception `oauth2app.authenticate.`**`InvalidToken`**
The access token provided is expired, revoked, malformed, or invalid for other reasons.

`error` = **'invalid_token'**

class `oauth2app.authenticate.`**`JSONAuthenticator`**(*scope=None*)
Wraps Authenticator, adds support for a callback parameter and JSON related. convenience methods.

Args:

- *request:* Django HttpRequest object.

Kwargs:

- *scope:* A iterable of oauth2app.models.AccessRange objects.

`callback` = **None**

`error_response`()
Returns a HttpResponse object of JSON error data.

`response`(*data*)
Returns a HttpResponse object of JSON serialized data.

Args:

- *data:* Object to be JSON serialized and returned.

`validate`(*request*)

exception `oauth2app.authenticate.`**`UnvalidatedRequest`**
The method requested requires a validated request to continue.

5.2.4 To Do

Todo

MAC Authentication

5.3 Access Token Generation

5.3.1 TokenGenerator

The TokenGenerator is used by the oauth2app.token.handler method to generate access tokens. It responds to several grant types, specified through the grant_type request parameter.

- **authorization_code:** Grants an access token based on an authorization code issued via Authorization.
- **refresh_token:** Refreshes an access token.
- **password:** Grants an access token based on a POST containing a username and password.
- **client_credentials:** Grants an access token based specific to the client to access internal resources.

Connect the handler method to the access endpoint.

```
from django.conf.urls.defaults import patterns

urlpatterns = patterns('',
    (r'^oauth2/token/?$',  'oauth2app.token.handler'),
)
```

To set token handler parameters, you can also use the TokenGenerator callable.

```
from django.conf.urls.defaults import patterns
from oauth2app.token import TokenGenerator
from oauth2app.consts import MAC

oauth2_token_generator = TokenGenerator(authentication_method=MAC, refreshable=False)

urlpatterns = patterns('',
        (r'^token/?$',  oauth2_token_generator)
)
```

5.3.2 Module Reference

OAuth 2.0 Token Generation

exception `oauth2app.token.`**`AccessTokenException`**
Access Token exception base class.

exception `oauth2app.token.`**`InvalidClient`**
Client authentication failed (e.g. unknown client, no client credentials included, multiple client credentials included, or unsupported credentials type).

`error` **= 'invalid_client'**

exception `oauth2app.token.`**`InvalidGrant`**
The provided authorization grant is invalid, expired, revoked, does not match the redirection URI used in the authorization request, or was issued to another client.

`error` **= 'invalid_grant'**

exception `oauth2app.token.`**`InvalidRequest`**
The request is missing a required parameter, includes an unsupported parameter or parameter value, repeats a

parameter, includes multiple credentials, utilizes more than one mechanism for authenticating the client, or is otherwise malformed.

`error` **= 'invalid_request'**

exception `oauth2app.token.`**`InvalidScope`**

The requested scope is invalid, unknown, malformed, or exceeds the scope granted by the resource owner.

`error` **= 'invalid_scope'**

class `oauth2app.token.`**`TokenGenerator`**(*scope=None*, *authentication_method=1*, *refreshable=True*)

Token access handler. Validates authorization codes, refresh tokens, username/password pairs, and generates a JSON formatted authorization code.

Args:

- *request:* Django HttpRequest object.

Kwargs:

- *scope:* An iterable of oauth2app.models.AccessRange objects representing the scope the token generator will grant. *Default None*
- *authentication_method:* Type of token to generate. Possible values are: oauth2app.consts.MAC and oauth2app.consts.BEARER *Default oauth2app.consts.BEARER*
- *refreshable:* Boolean value indicating whether issued tokens are refreshable. *Default True*

`access_token` **= None**

`client` **= None**

`code` **= None**

`error` **= None**

`error_response`()

In the event of an error, return a Django HttpResponse with the appropriate JSON encoded error parameters.

Returns HttpResponse

`grant_response`()

Returns a JSON formatted authorization code.

`request` **= None**

`user` **= None**

`valid` **= False**

`validate`()

Validate the request. Raises an AccessTokenException if the request fails authorization.

Returns None

exception `oauth2app.token.`**`UnauthorizedClient`**

The client is not authorized to request an authorization code using this method.

`error` **= 'unauthorized_client'**

exception `oauth2app.token.`**`UnsupportedGrantType`**

The authorization grant type is not supported by the authorization server.

`error` **= 'unsupported_grant_type'**

exception `oauth2app.token.`**`UnvalidatedRequest`**

The method requested requires a validated request to continue.

`oauth2app.token.`**`handler`**`(`**args*, ***kwargs*`)`

Token access handler. Conveneince function that wraps the Handler() callable.

Args:

- *request:* Django HttpRequest object.

5.4 Models

5.4.1 Module Reference

OAuth 2.0 Django Models

class `oauth2app.models.`**`AccessRange`**`(`**args*, ***kwargs*`)`

Stores access range data, also known as scope.

Args:

- *key:* A string representing the access range scope. Used in access token requests.

Kwargs:

- *description:* A string representing the access range description. *Default None*

exception **`DoesNotExist`**

exception `AccessRange.`**`MultipleObjectsReturned`**

`AccessRange.`**`accesstoken_set`**

`AccessRange.`**`code_set`**

`AccessRange.`**`objects`** **= <django.db.models.manager.Manager object>**

class `oauth2app.models.`**`AccessToken`**`(`**args*, ***kwargs*`)`

Stores access token data.

Args:

- *client:* A oauth2app.models.Client object
- *user:* A Django User object

Kwargs:

- *token:* A string representing the access key token. *Default 10 character random string*
- *refresh_token:* A string representing the access key token. *Default 10 character random string*
- *mac_key:* A string representing the MAC key. *Default None*
- *expire:* A positive integer timestamp representing the access token's expiration time.
- *scope:* A list of oauth2app.models.AccessRange objects. *Default None*
- *refreshable:* A boolean that indicates whether this access token is refreshable. *Default False*

exception **`DoesNotExist`**

exception `AccessToken.`**`MultipleObjectsReturned`**

`AccessToken.`**`client`**

`AccessToken.`**`macnonce_set`**

`AccessToken.`**`objects`** **= <django.db.models.manager.Manager object>**

`AccessToken.`**`scope`**

`AccessToken.`**`user`**

class `oauth2app.models.`**`Client`**`(`**args*, ***kwargs*`)`

Stores client authentication data.

Args:

- *name:* A string representing the client name.
- ***user:*** **A Django User object representing the client** owner.

Kwargs:

- *description:* A string representing the client description. *Default None*
- *key:* A string representing the client key. *Default 30 character random string*
- *secret:* A string representing the client secret. *Default 30 character random string*
- *redirect_uri:* A string representing the client redirect_uri. *Default None*

exception **`DoesNotExist`**

exception `Client.`**`MultipleObjectsReturned`**

`Client.`**`accesstoken_set`**

`Client.`**`code_set`**

`Client.`**`objects`** **= <django.db.models.manager.Manager object>**

`Client.`**`user`**

class `oauth2app.models.`**`Code`**`(`**args*, ***kwargs*`)`

Stores authorization code data.

Args:

- *client:* A oauth2app.models.Client object
- *user:* A Django User object

Kwargs:

- *key:* A string representing the authorization code. *Default 30 character random string*
- *expire:* A positive integer timestamp representing the access token's expiration time.
- *redirect_uri:* A string representing the redirect_uri provided by the requesting client when the code was issued. *Default None*
- *scope:* A list of oauth2app.models.AccessRange objects. *Default None*

exception **`DoesNotExist`**

exception `Code.`**`MultipleObjectsReturned`**

`Code.`**`client`**

`Code.`**`objects`** **= <django.db.models.manager.Manager object>**

`Code.`**`scope`**

`Code.`**`user`**

class `oauth2app.models.`**`KeyGenerator`**(*length*)
Callable Key Generator that returns a random keystring.

Args:

- *length:* A integer indicating how long the key should be.

Returns str

class `oauth2app.models.`**`MACNonce`**(**args*, ***kwargs*)
Stores Nonce strings for use with MAC Authentication.

Args:

- *access_token:* A oauth2app.models.AccessToken object
- *nonce:* A unique nonce string.

exception **`DoesNotExist`**

exception `MACNonce.`**`MultipleObjectsReturned`**

`MACNonce.`**`access_token`**

`MACNonce.`**`objects`** **= <django.db.models.manager.Manager object>**

class `oauth2app.models.`**`TimestampGenerator`**(*seconds=0*)
Callable Timestamp Generator that returns a UNIX time integer.

Kwargs:

- *seconds:* A integer indicating how many seconds in the future the timestamp should be. *Default 0*

Returns int

5.5 Settings

The following settings can be specified in Django settings.py:

5.5.1 Client Key Length

```
OAUTH2_CLIENT_KEY_LENGTH
```

Length of the client key.

Default 30

5.5.2 Client Secret Length

```
OAUTH2_CLIENT_SECRET_LENGTH
```

Length of the client secret.

Default 30

5.5.3 Code Key Length

```
OAUTH2_CODE_KEY_LENGTH
```

Length of the code key.

Default 30

5.5.4 MAC Key Length

```
OAUTH2_MAC_KEY_LENGTH
```

Length of the MAC authentication key. Only used when the authentication method is set to oauth2app.consts.MAC. See *Authentication method*.

Default 20

5.5.5 Access Token Length

```
OAUTH2_ACCESS_TOKEN_LENGTH
```

Length of the access token.

Default 10

5.5.6 Refresh Token Length

```
OAUTH2_REFRESH_TOKEN_LENGTH
```

Length of the refresh token.

Default 10

5.5.7 Refreshable Tokens

```
OAUTH2_REFRESHABLE
```

Issue refreshable tokens.

Default True

5.5.8 Authorization Code Expiration

```
OAUTH2_CODE_EXPIRATION
```

Number of seconds in which an authorization code should expire.

Default 120

5.5.9 Access Token Expiration

```
OAUTH2_ACCESS_TOKEN_EXPIRATION
```

Number of seconds in which an access token should expire.

Default 3600

5.5.10 Authentication method

```
OAUTH2_AUTHENTICATION_METHOD
```

Authentication method. Possible values are oauth2app.consts.MAC and oauth2app.consts.BEARER.

For Bearer see http://tools.ietf.org/html/draft-ietf-oauth-saml2-bearer-03 and http://tools.ietf.org/html/draft-ietf-oauth-v2-bearer-04

For MAC see http://tools.ietf.org/html/draft-ietf-oauth-v2-http-mac-00

Default oauth2app.consts.BEARER

5.5.11 Realm

:: OAUTH2_REALM

Authentication realm

Default “”

CHAPTER 6

Tests

- See https://github.com/hiidef/oauth2app/tree/develop/tests/testsite

The test site uses django.db.backends.sqlite3 and requires minimal configuration.

```
git clone git@github.com:hiidef/oauth2app.git oauth2app
cd oauth2app/tests/testsite
git checkout master
pip install https://github.com/hiidef/oauth2app/tarball/master django-test-coverage
python manage.py test api
```

CHAPTER 7

To Do

Todo

MAC Authentication

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/oauth2app/checkouts/latest/docs/authenticate.rst, line 65.)

Index