django-scim2 Documentation

Release 0.5.2

Paul Logston

Dec 22, 2018

Modules

This is a partial provider-side implementation of the SCIM 2.0[1] specification for use in Django.

Note that currently the only supported database is Postgres.

[1] http://www.simplecloud.info/, https://tools.ietf.org/html/rfc7644

CHAPTER 1

Installation

Install with pip:

```
$ pip install django-scim2
```

Then add the `django_scim` app to `INSTALLED_APPS` in your Django's settings:

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

Add the appropriate middleware to authorize or deny the SCIM calls:

```
MIDDLEWARE_CLASSES = (
    ...
    'django_scim.middleware.SCIMAuthCheckMiddleware',
    ...
)
```

Make sure to place this middleware after authentication middleware as this middleware simply checks *request.user.is_anonymous()* to determine if the SCIM request should be allowed or denied.

Add the necessary url patterns to your root urls.py file. Please note that the namespace is mandatory and must be named *scim*:

```
# Django 1.11
urlpatterns = [
    ...
    url(r'^scim/v2/', include('django_scim.urls', namespace='scim')),
]

# Django 2+
urlpatterns = [
    ...
```

(continues on next page)

(continued from previous page)

```
    path('scim/v2/', include('django_scim.urls')),
]
```

Finally, add settings appropriate for you app to your settings.py file:

```
SCIM_SERVICE_PROVIDER = {
    'NETLOC': 'localhost',
    'AUTHENTICATION_SCHEMES': [
        {
            'type': 'oauth2',
            'name': 'OAuth 2',
            'description': 'Oauth 2 implemented with bearer token',
        },
    ],
}
```

Other SCIM settings can be provided but those listed above are required.

CHAPTER 2

PyPI

https://pypi.python.org/pypi/django-scim2

CHAPTER 3

Source

https://github.com/15five/django-scim2

CHAPTER 4

Documentation

http://django-scim2.readthedocs.io/

CHAPTER 5

Tests

https://travis-ci.com/15five/django-scim2

CHAPTER 6

License

This library is released under the terms of the **MIT license**. Full details in `LICENSE.txt` file.

CHAPTER 7

Extensibility

This library was forked and developed to be highly extensible. A number of adapters can be defined to control what different endpoints do to your resources. Please see the documentation for more details.

PLEASE NOTE: This app does not implement authorization and authentication. Such tasks are left for other apps such as Django OAuth Toolkit to implement.

CHAPTER 8

Credits

This project was forked from https://bitbucket.org/atlassian/django_scim

CHAPTER 9

Contents

9.1 Adapters

Adapters are used to convert the data model described by the SCIM 2.0 specification to a data model that fits the data provided by the application implementing a SCIM api.

For example, in a Django app, there are User and Group models that do not have the same attributes/fields that are defined by the SCIM 2.0 specification. The Django User model has both `first_name` and `last_name` attributes but the SCIM speicifcation requires this same data be sent under the names `givenName` and `familyName` respectively.

An adapter is instantiated with a model instance. Eg:

```
user = get_user_model().objects.get(id=1)
scim_user = SCIMUser(user)
...
```

class django_scim.adapters.**SCIMGroup**(*obj*, *request=None*)

Adapter for adding SCIM functionality to a Django Group object.

This adapter can be overriden; see the `GROUP_ADAPTER` setting for details.

display_name

Return the displayName of the group per the SCIM spec.

from_dict(*d*)

Consume a `dict` conforming to the SCIM Group Schema, updating the internal group object with data from the `dict`.

Please note, the group object is not saved within this method. To persist the changes made by this method, please call `.save()` on the adapter. Eg:

```
scim_group.from_dict(d)
scim_group.save()
```

handle_add(*operation*)

Handle add operations.

handle_remove(*operation*)
Handle remove operations.

handle_replace(*operation*)
Handle the replace operations.

members
Return a list of user dicts (ready for serialization) for the members of the group.

Return type list

meta
Return the meta object of the group per the SCIM spec.

classmethod resource_type_dict(*request=None*)
Return a `dict` containing ResourceType metadata for the group object.

to_dict()
Return a `dict` conforming to the SCIM User Schema, ready for conversion to a JSON object.

class django_scim.adapters.**SCIMUser**(*obj*, *request=None*)
Adapter for adding SCIM functionality to a Django User object.

This adapter can be overriden; see the `USER_ADAPTER` setting for details.

display_name
Return the displayName of the user per the SCIM spec.

emails
Return the email of the user per the SCIM spec.

from_dict(*d*)
Consume a `dict` conforming to the SCIM User Schema, updating the internal user object with data from the `dict`.

Please note, the user object is not saved within this method. To persist the changes made by this method, please call `.save()` on the adapter. Eg:

```
scim_user.from_dict(d)
scim_user.save()
```

groups
Return the groups of the user per the SCIM spec.

handle_replace(*operation*)
Handle the replace operations.

meta
Return the meta object of the user per the SCIM spec.

classmethod resource_type_dict(*request=None*)
Return a `dict` containing ResourceType metadata for the user object.

to_dict()
Return a `dict` conforming to the SCIM User Schema, ready for conversion to a JSON object.

9.2 Filters

Filter transformers are used to convert the SCIM query and filter syntax into valid SQL queries.

class django_scim.filters.**SCIMGroupFilterTransformer**
Transforms a PlyPlus parse tree into a tuple containing a raw SQL query and a dict with query parameters to go with the query.

join()
Returns join expressions. E.g.

JOIN bb_userprofile p ON p.user_id = u.id

classmethod search(*query*, *request=None*)
Takes a SCIM 1.1 filter query and returns a Django *QuerySet* that contains zero or more group model instances.

Parameters **query** (*unicode*) – a *unicode* query string.

class django_scim.filters.**SCIMUserFilterTransformer**
Transforms a PlyPlus parse tree into a tuple containing a raw SQL query and a dict with query parameters to go with the query.

join()
Returns join expressions. E.g.

JOIN bb_userprofile p ON p.user_id = u.id

classmethod search(*query*, *request=None*)
Takes a SCIM 1.1 filter query and returns a Django *QuerySet* that contains zero or more user model instances.

Parameters **query** (*unicode*) – a *unicode* query string.

9.3 Models

class django_scim.models.**SCIMServiceProviderConfig**(*request=None*)
A reference ServiceProviderConfig. This should be overridden to describe those authentication_schemes and features that are implemented by your app.

9.4 Utilities

django_scim.utils.**default_base_scim_location_getter**(*request=None*, **args*, ***kwargs*)
Return the default location of the app implementing the SCIM api.

django_scim.utils.**default_get_extra_model_exclude_kwargs_getter**(*model*)
Return a **method** that will return extra model exclude kwargs for the passed in model.

Parameters **model** –

django_scim.utils.**default_get_extra_model_filter_kwargs_getter**(*model*)
Return a **method** that will return extra model filter kwargs for the passed in model.

Parameters **model** –

django_scim.utils.**get_all_schemas_getter**()
Return a function that will, when called, returns the base location of scim app.

django_scim.utils.**get_base_scim_location_getter**()
Return a function that will, when called, returns the base location of scim app.

django_scim.utils.**get_extra_model_exclude_kwargs_getter**(*model*)
Return a function that will, when called, returns the base location of scim app.

django_scim.utils.**get_extra_model_filter_kwargs_getter**(*model*)
Return a function that will, when called, returns the base location of scim app.

django_scim.utils.**get_group_adapter**()
Return the group model adapter.

django_scim.utils.**get_group_model**()
Return the group model.

django_scim.utils.**get_service_provider_config_model**()
Return the Service Provider Config model.

django_scim.utils.**get_user_adapter**()
Return the user model adapter.

9.5 Views

class django_scim.views.**GroupsView**(***kwargs*)

get_extra_exclude_kwargs(*request*, **args*, ***kwargs*)
Return extra exclude kwargs for the given model. :param request: :param args: :param kwargs: :rtype: dict

get_extra_filter_kwargs(*request*, **args*, ***kwargs*)
Return extra filter kwargs for the given model. :param request: :param args: :param kwargs: :rtype: dict

model_cls
alias of django.contrib.auth.models.Group

parser
alias of *django_scim.filters.SCIMGroupFilterTransformer*

scim_adapter
alias of *django_scim.adapters.SCIMGroup*

class django_scim.views.**ResourceTypesView**(***kwargs*)

class django_scim.views.**SCIMView**(***kwargs*)

get_object()
Get object by configurable ID.

status_501(*request*, **args*, ***kwargs*)
A service provider that does NOT support a feature SHOULD respond with HTTP status code 501 (Not Implemented).

class django_scim.views.**SchemasView**(***kwargs*)

class django_scim.views.**SearchView**(***kwargs*)

get_extra_exclude_kwargs(*request*, **args*, ***kwargs*)
Return extra exclude kwargs for the given model. :param request: :param args: :param kwargs: :rtype: dict

get_extra_filter_kwargs(*request*, **args*, ***kwargs*)
Return extra filter kwargs for the given model. :param request: :param args: :param kwargs: :rtype: dict

class django_scim.views.**ServiceProviderConfigView**(***kwargs*)

class django_scim.views.**UsersView**(***kwargs*)

get_extra_exclude_kwargs(*request*, **args*, ***kwargs*)
Return extra exclude kwargs for the given model. :param request: :param args: :param kwargs: :rtype: dict

get_extra_filter_kwargs(*request*, **args*, ***kwargs*)
Return extra filter kwargs for the given model. :param request: :param args: :param kwargs: :rtype: dict

model_cls
alias of `django.contrib.auth.models.User`

parser
alias of *django_scim.filters.SCIMUserFilterTransformer*

scim_adapter
alias of *django_scim.adapters.SCIMUser*

- genindex
- modindex
- search

Python Module Index

d

Index

P

R

S

T

U