Django-SCIM2

User Provisioning At Scale

Paul Logston
https://github.com/logston
https://pycon.hk/sessions-2020-spring/django-scim2-user-provisioning-at-scale/

PyCon HK 2020 Spring

The Problem Being Solved

- Mass user provisioning to service
- Ongoing updates to those users
- Automated removal of those users from said service

Disclaimer:

- This talk is about a tool created to help abstract boilerplate code.
- This talk is not about distributing the tool over N hosts to handle massive numbers of requests per second.

What Existed Before SCIM?

Some proprietary solutions and attempts at an open standard:

- SPML
- PortableContacts
- vCards
- LDAP directory services

```
BEGIN:VCARD
VERSION:4.0
N:Gump;Forrest;;Mr.;
FN:Forrest Gump
ORG:Bubba Gump Shrimp Co.
TITLE:Shrimp Man
PHOTO;MEDIATYPE=image/gif:http://www.example.com/dir_photos/my_photo.gif
TEL;TYPE=work,voice;VALUE=uri:tel:+1-111-555-1212
TEL;TYPE=home,voice;VALUE=uri:tel:+1-404-555-1212
ADR;TYPE=WORK;PREF=1;LABEL="100 Waters Edge\nBaytown\, LA 30314\nUnited States of America":;;100
Waters Edge;Baytown;LA;30314;United States of America
ADR;TYPE=HOME;LABEL="42 Plantation St.\nBaytown\, LA 30314\nUnited States of America":;;42 Plantation
St.;Baytown;LA;30314;United States of America
EMAIL:forrestgump@example.com
REV:20080424T195243Z
X-qq:21588891
END:VCARD
```

2015 - SCIM 2.0


```
{
  "schemas":
    ["urn:ietf:params:scim:schemas:core:2.0:User",
      "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"],

  "id": "2819c223-7f76-453a-413861904646",
  "externalId": "701984",

  "userName": "bjensen@example.com",
  "name": {
    "formatted": "Ms. Barbara J Jensen, III",
    "familyName": "Jensen",
    "givenName": "Barbara",
    "middleName": "Jane",
    "honorificPrefix": "Ms.",
    "honorificSuffix": "III"
  },
  ...
```

https://tools.ietf.org/html/rfc7643 (Schema)
https://tools.ietf.org/html/rfc7644 (Protocol)
http://www.simplecloud.info/
https://en.wikipedia.org/wiki/User_provisioning_software
https://en.wikipedia.org/wiki/System_for_Cross-domain_Identity_Management

Django-SCIM2

Forked from [django_scim](#) around November 2016.

- Implements generic views
- Behavior of views is customized via adapters (wrappers):
  - User
  - Group
- Middleware to allow SCIM operations
  - Request must already be authenticated

https://bitbucket.org/atlassian/django_scim/
https://github.com/15five/django-scim2

```
➜ src git:(master) tree .
.
└── django_scim
    ├── __init__.py
    ├── __pycache__
    ├── adapters.py
    ├── constants.py
    ├── exceptions.py
    ├── filters.py
    ├── middleware.py
    ├── models.py
    ├── schemas
    │   ├── README.rst
    │   ├── __init__.py
    │   ├── __pycache__
    │   ├── core
    │   │   ├── Group.json
    │   │   ├── ResourceType.json
    │   │   ├── Schema.json
    │   │   ├── ServiceProviderConfig.json
    │   │   └── User.json
    │   └── extension
    │       └── Enterprise-User.json
    ├── settings.py
    ├── urls.py
    ├── utils.py
    └── views.py

6 directories, 19 files
```

Django-SCIM2: views.py

```
class PostView(object):
    def post(self, request, *args, **kwargs):
        obj = self.model_cls()
        scim_obj = self.scim_adapter(obj, request=request)

        body = self.load_body(request.body)

        if not body:
            raise exceptions.BadRequestError('POST call made with empty body')

        scim_obj.validate_dict(body)
        scim_obj.from_dict(body)
        ...
```

```
...
try:
    scim_obj.save()
except db.utils.IntegrityError as e:
    raise exceptions.IntegrityError(str(e))

content = json.dumps(scim_obj.to_dict())
response = HttpResponse(content=content,
                        content_type=constants.SCIM_CONTENT_TYPE,
                        status=201)
response['Location'] = scim_obj.location
return response
```

Django-SCIM2: views.py

```
class UsersView(FilterMixin, GetView, PostView, PutView, PatchView, DeleteView, SCIMView):

    http_method_names = ['get', 'post', 'put', 'patch', 'delete']

    scim_adapter_getter = get_user_adapter
    model_cls_getter = get_user_model
    parser_getter = get_user_filter_parser
```

Django-SCIM2: adapters.py

```
class SCIMUser(SCIMMixin):
    def from_dict(self, d):
        super().from_dict(d)

        username = d.get('userName')
        self.obj.username = username or ''

        self.obj.scim_username = self.obj.username

        first_name = d.get('name', {}).get('givenName')
        self.obj.first_name = first_name or ''

        last_name = d.get('name', {}).get('familyName')
        self.obj.last_name = last_name or ''
```

PATCH Requests

```
{
  "schemas":
   ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
    "Operations":[{
      "op":"add",
      "value":{
        "emails":[
          {
            "value":"babs@jensen.org",
            "type":"home"
          }
        ],
        "name.familyName": "Jensen"
    }]
 }
```

PATCH Requests

```
class SCIMUser(SCIMMixin):
    def handle_replace(self,
                       path: Optional[AttrPath],
                       value: Union[str, list, dict],
                       operation: dict):
        if not isinstance(value, dict):
            # Restructure for use in loop below.
            value = {path: value}

        if not isinstance(value, dict):
            raise exceptions.NotImplementedError(
                f'PATCH replace operation with value type of '
                f'{type(value)} is not implemented'
            )
        ....
```

```
...
 for path, value in (value or {}).items():
     if path.first_path in self.ATTR_MAP:
         setattr(self.obj, self.ATTR_MAP.get(path.first_path), value)

     elif path.first_path == ('emails', None, None):
         self.parse_emails(value)

     else:
         raise exceptions.NotImplementedError('Not Implemented')

 self.save()
```

scim2-filter-parser

```
Example Path: 'emails[type eq "work" and value co "@example.com"] or ims[type eq "xmpp" and value co "@foo.com"]'

$ sfp-parser 'emails[type eq "work" and value co "@example.com"] or ims[type eq "xmpp" and value co "@foo.com"]'

Filter(expr=LogExpr, negated=False, namespace=None)
    LogExpr(op='or', expr1=Filter, expr2=Filter)
        Filter(expr=Filter, negated=False, namespace=None)
            Filter(expr=Filter, negated=False, namespace=AttrPath)
                Filter(expr=LogExpr, negated=False, namespace=None)
                    LogExpr(op='and', expr1=Filter, expr2=Filter)
                        Filter(expr=AttrExpr, negated=False, namespace=None)
                            AttrExpr(value='eq', attr_path=AttrPath, comp_value=CompValue)
                                AttrPath(attr_name='type', sub_attr=None, uri=None)
                                CompValue(value='work')
                        Filter(expr=AttrExpr, negated=False, namespace=None)
                            AttrExpr(value='co', attr_path=AttrPath, comp_value=CompValue)
                                AttrPath(attr_name='value', sub_attr=None, uri=None)
                                CompValue(value='@example.com')
                ...
```

scim2-filter-parser

```
Example Path: 'emails[type eq "work" and value co "@example.com"] or ims[type eq "xmpp" and value co "@foo.com"]'
```

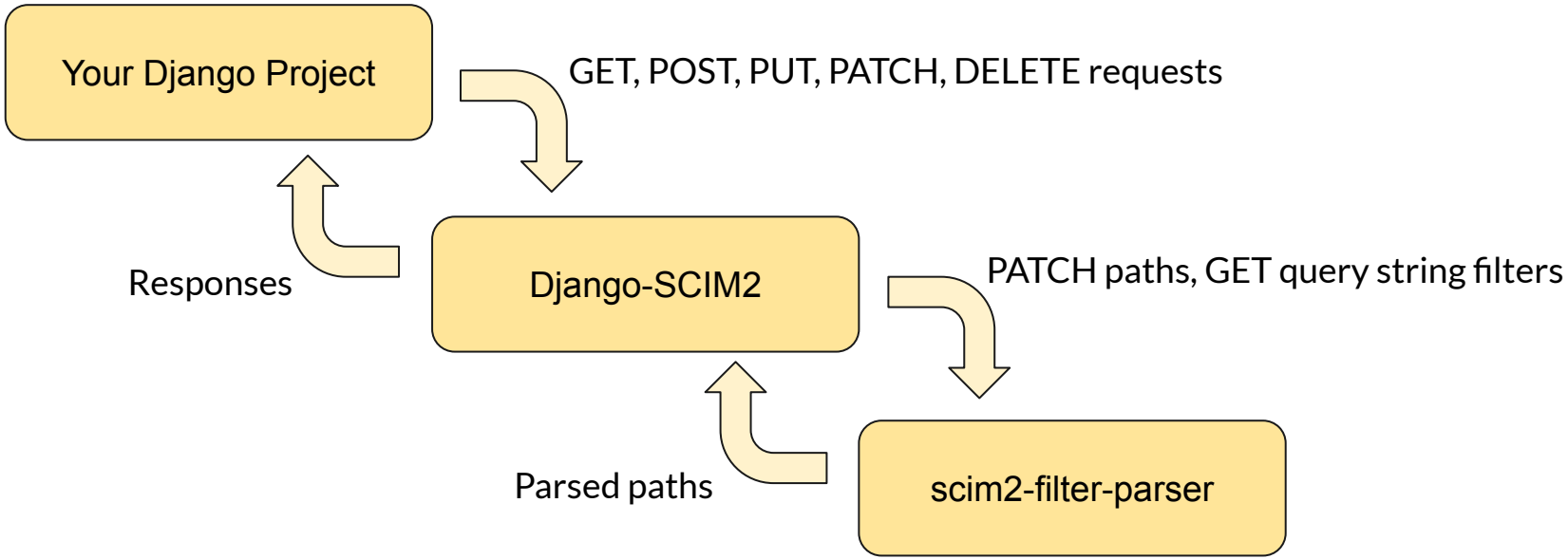
```
$ sfp-query 'emails[type eq "work" and value co "@example.com"] or ims[type eq "xmpp" and value co "@foo.com"]'
```

```
    SELECT DISTINCT users.*
    FROM users
    LEFT JOIN emails ON emails.user_id = users.id
    LEFT JOIN schemas ON schemas.user_id = users.id
      WHERE ((emails.type = work) AND (emails.value LIKE %@example.com%)) OR
            ((ims.type = xmpp) AND (ims.value LIKE %@foo.com%));
```

** SQL injection protection is handled via transpiler

https://github.com/15five/scim2-filter-parser

Django-SCIM2: Putting It All Together



Thanks & Questions

Thanks for your time!

Thanks to 15Five for time to build these open source tools.

Questions?

Libraries:

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