sgapi Documentation

Release 0.1

Mike Boers

July 14, 2015

Contents

1 Installation **3**

2 Basic Usage **5**

3 Python API **7**

3.1 `sgapi` 7

3.2 `sgapi.filters` 7

Python Module Index **9**

This project is a lower-level Python implementation of the Shotgun API. The canonical Python API (`shotgun_api3`) can be found on Shotgun's GitHub page

This project exists because we wanted to have a little more control over the details of the connection, threading, time handling, etc., but monkey-patching the `shotgun_api3` was deemed too unsafe. Ergo, we have a minimal API that does the few things that we need it to do at a low level.

Extra things that we implement include:

- forgiving filters which understand any of the 3 filter dialects;
- asynchronous paging during find via *threads*=<*number of threads*>.

CHAPTER 1

Installation

From GitHub:

```
pip install -e git+git@github.com:westernx/sgapi#egg=sgapi
```

Basic Usage

```
>>> from sgapi import Shotgun

>>> # Basic instantiation is the same:
>>> sg = Shotgun(server_url, script_name, api_key)

>>> # Info is the same:
>>> sg.info()
{
    's3_uploads_enabled': True,
    'totango_site_id': '123',
    'version': [6, 0, 3],
    'totango_site_name': 'com_shotgunstudio_example'
}

>>> # Finding can be the same:
>>> sg.find_one('Task', [('id', 'is', 1234)])
{'type': 'Task', 'id': 1234}

>>> # You can also iterate over entities while requests run in a thread:
>>> for e in sg.find('Task', [...], threads=3, per_page=100):
...     process_entity(e)

>>> # Or you can manually construct requests:
>>> sg.call('find', {...})
```

CHAPTER 3

Python API

3.1 sgapi

class sgapi.**Shotgun**(*base_url*, *script_name*, *api_key*)

call(*method_name*, *method_params=None*, *authenticate=True*)
Make a raw API request.

Parameters

- **method_name** (*str*) – The remote method to call, e.g. `"info"` or `"read"`.
- **method_params** (*dict*) – The parameters for that method.
- **authenticat** (*bool*) – Pass authentication info along?

Raises ShotgunError if there is a remote error.

Returns the API results.

find(**args*, ***kwargs*)
Same as Shotgun's find

If `threads` is set to an integer, that many threads are used to make consecutive page requests in parallel.

find_iter(**args*, ***kwargs*)
Like `find()`, but yields entities as they become available.

find_one(*entity_type*, *filters*, *fields=None*, *order=None*, *filter_operator=None*, *retired_only=False*, *include_archived_projects=True*)
Same as Shotgun's find_one

info()
Basic `info` request.

3.2 sgapi.filters

There are 3 different filter syntaxes we tend to see:

1. Lists of tuples: these simple filters are lists of individual filter tuples, usually "and"ed together (although the Python API does take a `filter_operator`). These tuples are of the form:

```
(path, relation, *values)
```

e.g.:

```
('id', 'is', 1234)
```

2. Python dicts: complex logic can be expressed via dicts. They typically have the form:

```
{
    'filter_operator': 'all', # or 'any',
    'filters': [
        # a list of filters go here
    ],
}
```

The filters can either be of the simple tuples above, or additional dicts of the same form.

3. API dicts: the format that is seen by the remote API is all dicts. They look like:

```
{
    'logical_operator': 'and', # or 'or'
    'conditions': [
        # individual filters, such as:
        {
            'path': 'id',
            'relation': 'is',
            'values': [1234]
        },
        # or sub filters:
        {
            'logical_operator': 'and',
            'conditions': [
                # ... and so on.
            ]
        }
    ]
}
```

Here we offer functions to adapt any of the above syntaxes into the RPC version.

`sgapi.filters.`**`adapt_filters`**(*filters*, *operator=None*)

Given any of the 3 filter dialects, translate into the remote condition syntax.

Python Module Index

S

sgapi, 7
sgapi.filters, 7

Index

A

adapt_filters() (in module sgapi.filters), 8

C

call() (sgapi.Shotgun method), 7

F

find() (sgapi.Shotgun method), 7
find_iter() (sgapi.Shotgun method), 7
find_one() (sgapi.Shotgun method), 7

I

info() (sgapi.Shotgun method), 7

S

sgapi (module), 7
sgapi.filters (module), 7
Shotgun (class in sgapi), 7