TOM Toolkit

Architecture

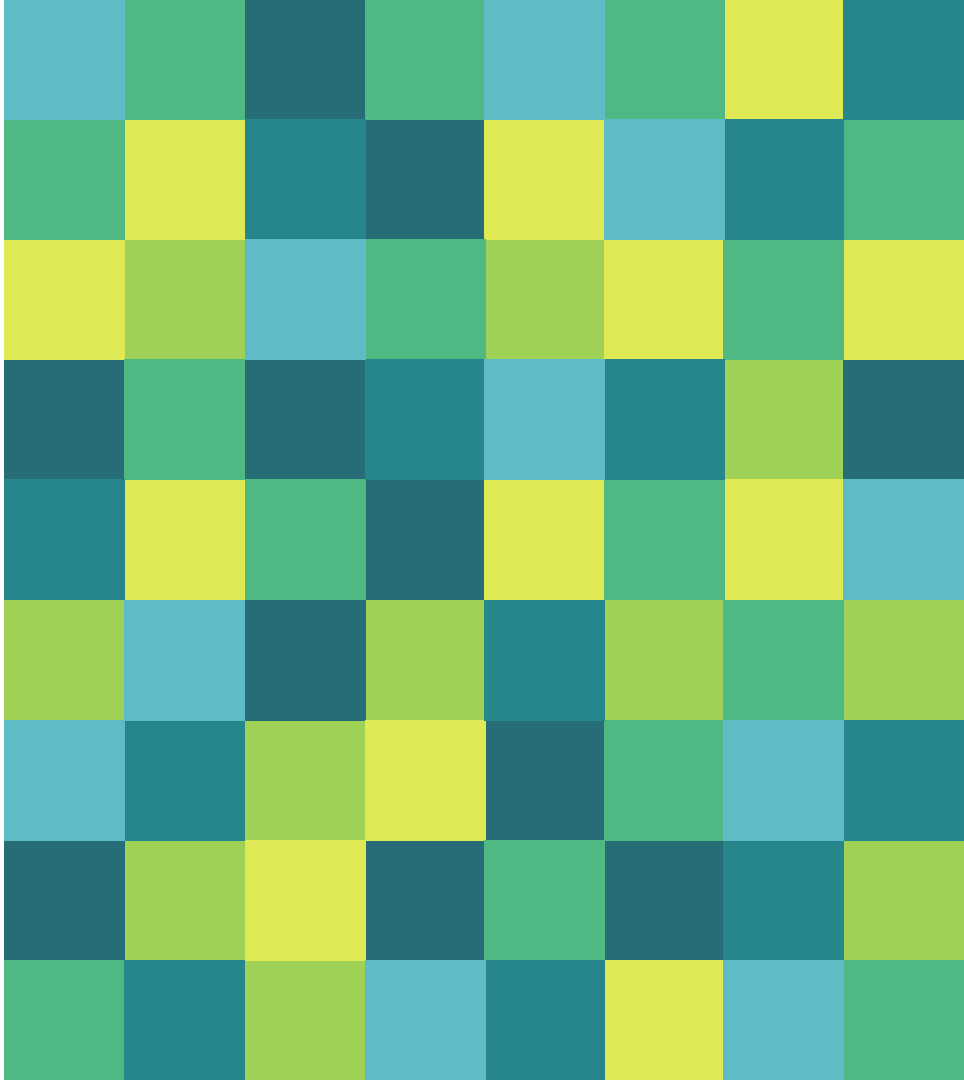
Engineering Goals

- Provide common TOM functionality
- Easy to customize
- Minimize software knowledge required

1.

Overview

Language choice and frameworks



Built on the Web

- Easy to develop
- Easy to distribute
- Runs in web browsers
- Near native performance/experience

People have come to expect web applications for nearly everything now.

Python

- Popular in the sciences
- Excellent library support
- Great for web development

An obvious choice as the main language in the TOM Toolkit.

Django

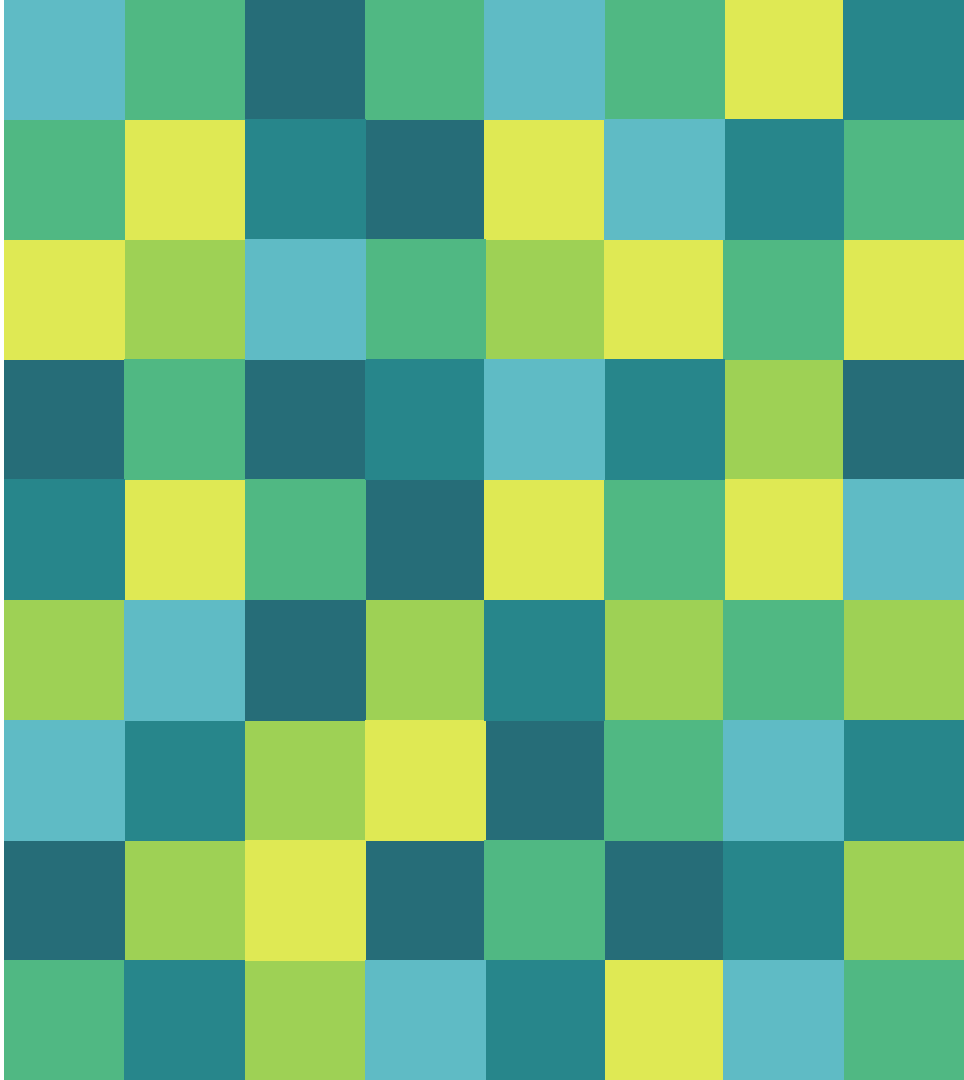
- Web + Python
- Mature
- Extensible

The TOM Toolkit isn't the first framework built on top of Django.

2.

Customization

Make the TOM work for you



Plugins

- Generic & common functions abstracted
- Standalone, shareable modules
- Easy to develop

The gemini.py module is an example of a plugin written by Bryan Miller and contributed to the TOM Toolkit codebase.

```
class GEMFacility(GenericObservationFacility):
    name = 'GEM'
    form = GEMObservationForm

    def submit_observation(self, observation_payload):
        obsids = []
        for payload in observation_payload:
            response = make_request(
                'POST', PORTAL_URL[get_site(payload['prog'])] + '/too',
                verify=False, params=payload)
            # Return just observation number
            obsid = response.text.split('-')
            obsids.append(obsid[-1])
        return obsids
```

Templating Engine

The TOM Toolkit uses the powerful Django Templating engine in which any template can be overridden without modifying the framework's code.

- Easy to change layouts using html & css
- No python necessary
- Override parts of or entire pages

M42

Update Target | Delete Target

Identifier	M42
Name	Messier 42
Name 2	NGC 1976
Name 3	Orion Nebula
Target Type	SIDEREAL
Right Ascension	83.822
	05:35:17.304
Declination	-5.391
	-05:23:27.960
Proper Motion (Ra)	1.670
Proper Motion (Declination)	-0.300
Distance	0.500

Survey View

Observe | Observations | Data

Observe

LCO | GEM

Plan

Start Time

Start Time

End Time

End Time

Maximum Airmass

Maximum Airmass

Plan

Comments

Austin Riba 2019-05-16

Everyone's favorite nebula!

Comment

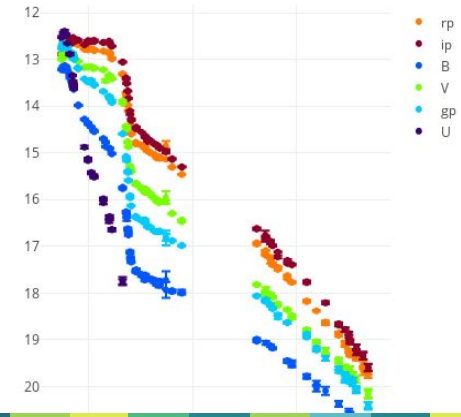
Comment

at2017eaw

Update Target | Delete Target

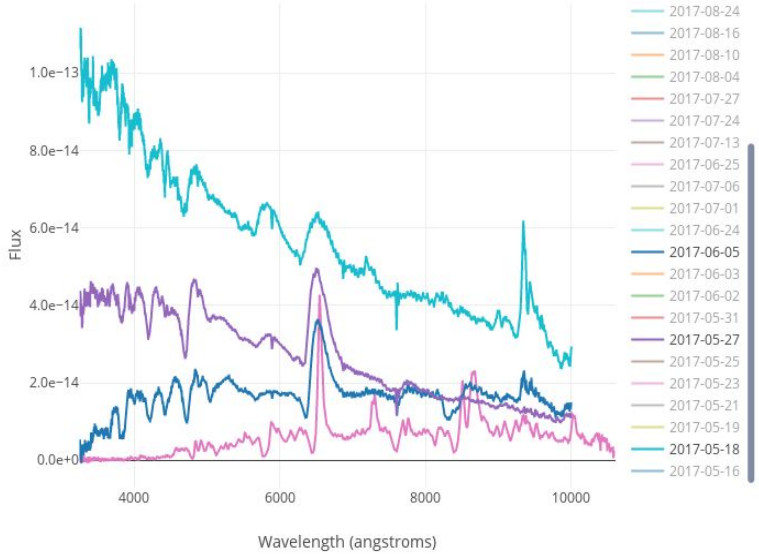
Right Ascension	308.6843333333
	20:34:44.240
Declination	60.1933055556
	+60:11:35.900
Redshift	0.0001330000
Classification	SN II
Also known as	at2017eaw
	SN 2017eaw
	SN2017eaw

Check for new data

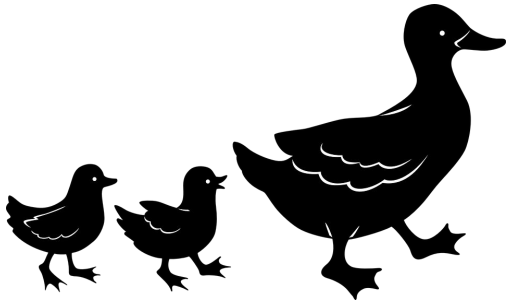


Overview | Spectra | Images | Data | Cross-Reference | Schedule Observations

Spectra



OOP



Tweaking the internals of TOM Toolkit functionality can usually be accomplished via inheritance. Django heavily relies on this pattern.

- Only change what you need to
- Reusable code

mytom.views.MyCustomTomTargetCreateView

tom_targets.views.TargetCreateView

django.views.generic.edit.CreateView

django.views.generic.edit.BaseCreateView

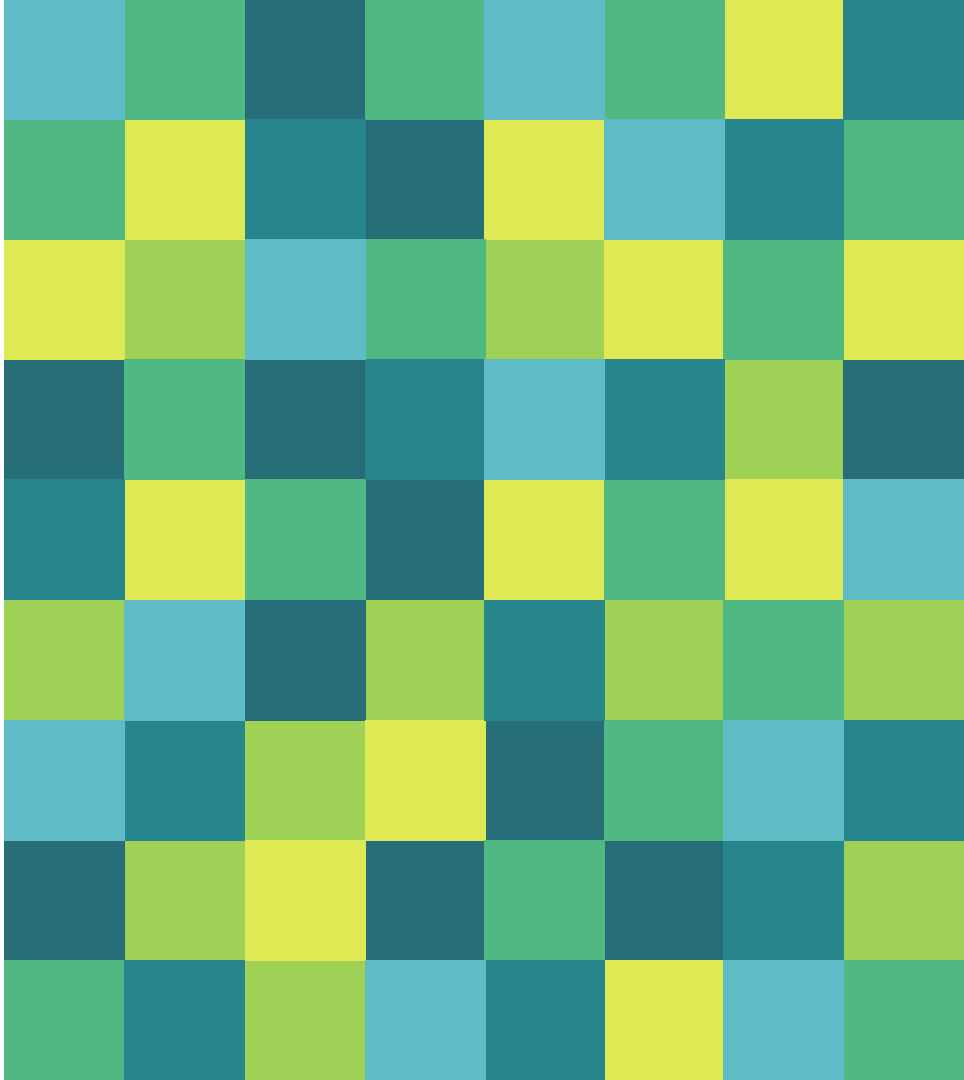
django.views.generic.edit.ProcessFormView

django.views.generic.base.View

3.

Reusable Apps

Organized functionality in the TOM



Django Reusable Apps

- Django's way of grouping functionality
- Can be distributed as python packages
- Popular apps: Django Rest Framework, Django-debug-toolbar, django-filter

When you install tomtoolkit from pypi, you download several Django reusable apps.

TOM Targets

Handles the storage and logic surrounding the Target model, some visualization.

Most other apps depend on tom_targets.

TOM Observations

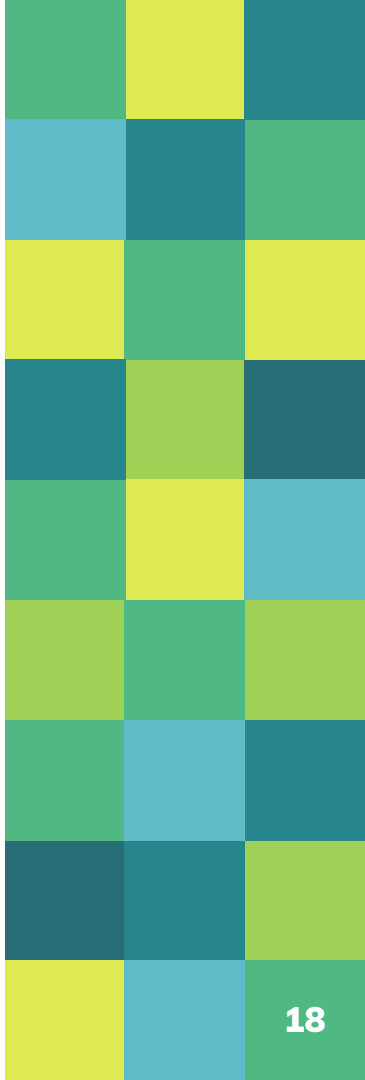
Logic for submitting and tracking observation requests. Contains the plugin framework for observatory plugins.

TOM Dataproducts

Allows users to upload and download data, locally or to cloud storage. Provides code hooks for pipelining/data processing.

TOM Alerts

Provides the plugin framework for broker modules. Handles the logic surrounding querying brokers and saving targets from alerts.



TOM Catalogs

Provides the plugin architecture for catalog plugins. Handles logic for creating targets from catalog search results.

TOM Common/Setup

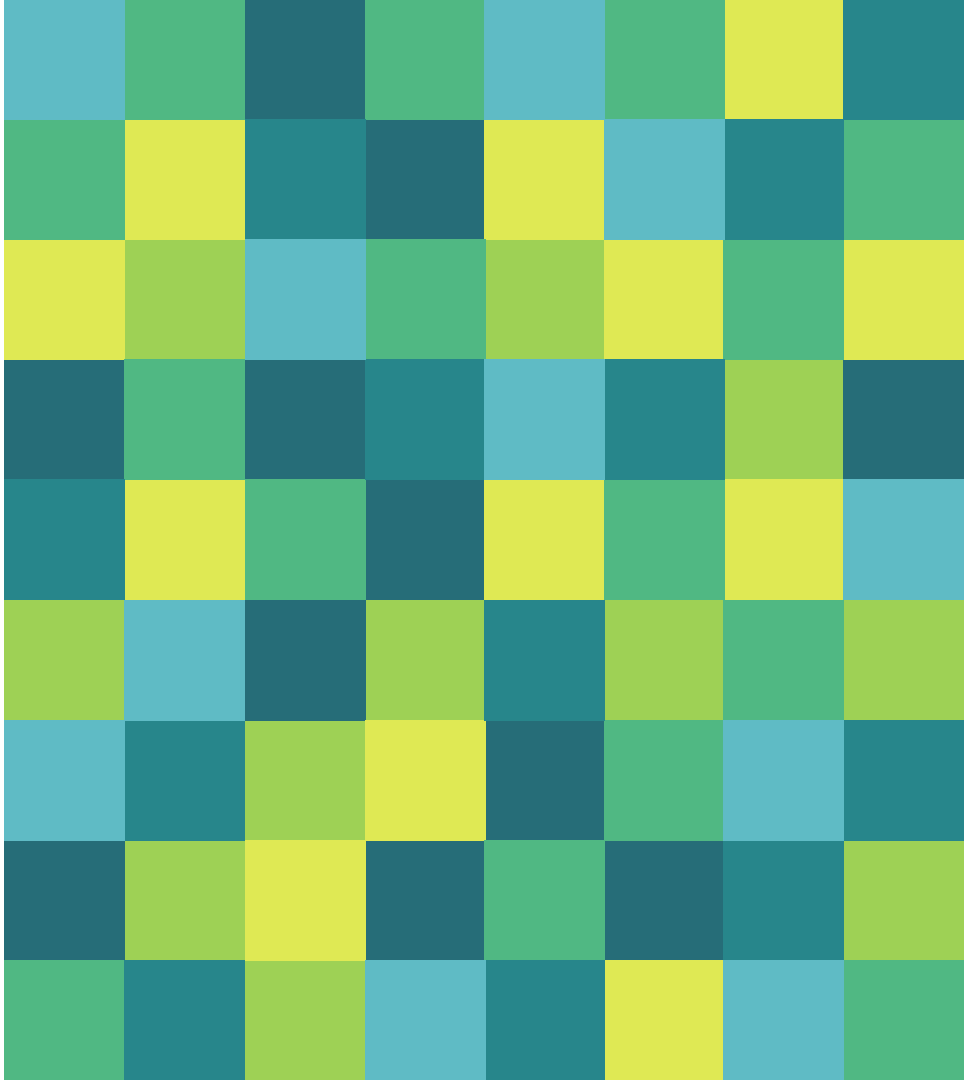
tom_common contains general logic that doesn't fit anywhere else.

tom_setup provides the tom_setup command, and is automatically uninstalled once a TOM is up and running.

4.

Tech Specifics

Because you probably want to have an opinion



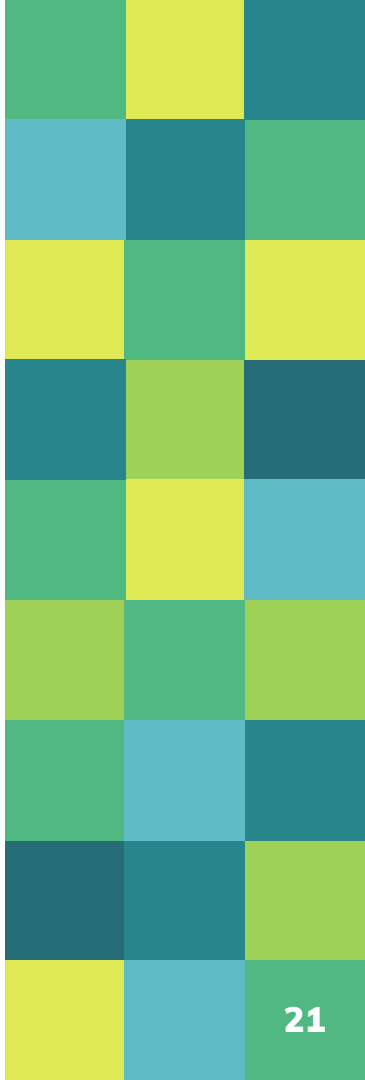
Languages: Python3, Javascript, HTML/CSS

Database: Relational: supports Sqlite, Postgres, MySql, Oracle.

Libraries: Django, bootstrap4, astropy, plotly, matplotlib, django-storages.

Tests: We got ‘em

SCM: https://github.com/TOMToolkit/



Thanks!

Any questions?

Reach out at:

- github.com/TomToolkit
- ariba@lco.global

Slide design by SlideCarnival