Crafting Good good-first-issue’s

Nisha Kumar
Open Source Engineer @ VMware
Open Source Summit North America 2019
Audience

Open Source Project Maintainers
Open Source Contributors
GitHub
Agenda

good-first-issue’s and where to find them
How to prepare for creating good-first-issue’s
How to create good-first-issue’s
good-first-issue’s wins
good-first-issue’s fails
A little about Tern

Container image inspection tool for Open Source Software compliance

Originated by VMware and donated to the Linux Foundation under the ACT project

Mostly written in Python3

2 years and some months old

2 maintainers (Rose and Nisha) who both identify as women
Caveat:

I can’t speak for other Open Source projects

All examples come from Tern’s GitHub page
What exactly is a good-first-issue?

Eg: https://github.com/vmware/tern/issues

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue

Go to Labels
Where can one find good-first-issue’s?
Where can one find good-first-issue’s?

Empty search? Look at the list of labels
Other labels like “good first issue”

- up-for-grabs
- low-hanging-fruit
- easy-fix
- difficulty/newcomer
- good-first-issue
- beginner-friendly
- first timers only
- starter-bug
- jump-in
Why (take the time to) create good-first-issue’s?

Contributors:
Makes your project approachable
Reduces the barriers to contribution - especially for groups underrepresented in Open Source
Make contributors interested in your project (potentially)
Why (take the time to) create good-first-issue’s?

Maintainers:

Create maintainable code

Take care of the important stuff (coding is the easy part)

Work towards growing the project
How to lay the groundwork for good-first-issue’s
Plan for accepting a Pull Request
This is probably not news but bears repeating

README, CONTRIBUTING, LICENSE, NOTICE, CODE_OF_CONDUCT

GitHub’s Issue Template

GitHub’s Pull Request Template

Getting Started Tutorials (You’re welcome to use mine: https://github.com/nishakm/puns)

Tests - linters, unit tests, functional tests

Continuous Integration (Tern uses CircleCI and TravisCI and looking to try GitHub Actions)

You don’t have to have *all* of these in place….but it helps
Break a feature down into small tasks

Big Feature: Structured JSON output

For each class, implement function to return a JSON object

Implement a function to return collated JSON objects

Class 1

Class 2

Class 3

Call class’s JSON function

Check for corner cases
Document changes that are needed to complete the task

Task Goal

In `classes/origins.py` add a `to_dict(self)` method to get an object. The resulting object should look like this:

```python
[{\'origin_str\': \'<string>', \'notices\': [...]}, {\'origin_str\': ...}]
```

For reference, look at `classes/notice.py` and `classes/notice2.py` methods do.

Implementation could look like this:

```python
origin_list = []
for origin in self.origins:
    origin_list.append(origin.to_dict())
return origin_list
```

This is work towards [#70](#70)
Use stubs

A stub is a placeholder for some yet-to-be-developed functionality

```python
def is_ready(some_value):
    # Things need to be implemented here but I'll
    # do it later
    return False

def do_the_thing(some_value):
    # Things need to be implemented here but I'll
    # do it later
    if is_ready(some_value):
        pass
    else:
        print("Not Ready")
```
Pick changes in the modular part of the code

```python
def get_pkg_attr_list(shell, attr_dict, package_name=''): chroot=True, override=''

def check_library_key(listing, key):
    '''Given the command library listing, check to see if a key is present. If the key is in the list of keys that should be in there then provide a note'''
    try:
        return listing[key], ''
    except KeyError as e:
        if e in base_keys and e not in package_keys:
            return {}, errors.no_listing_for_base_key.format(listing_key=e)
        if e in package_keys and e not in base_keys:
            return {}, errors.no_listing_for_package_key.format(listing_key=e)
        return errors.unsupported_key_for_key.format(listing_key=e)

def get_image_shell(base_image_list):
    '''Given the base image listing, return the image shell. If there is no shell listing, return a default string'''
    shell, msg = check_library_key(base_image_listing, 'shell')
    if not shell:
        shell = ''
    return shell, msg

def get_package_listing(command_name):
    '''Given a command name, return the package listing from the snippet library.'''
    return get_command_listing(command_name)['packages']
```
good-first-issue wins
Supporting the contributor

This isn’t a technical interview

Update the `load_from_cache` def in common.py to take an argument called `redo` (means the default operation should be to not redo the cache)

```python
35 def load_from_cache(layer):  # add new argument redo=False
36     '''Given a layer object, check against cache to see if that layer
37        if yes then get the package list and load it in the layer and
38        If it doesn't exist return false  # add documentation for the
39        Add notices to the layer's origins matching the origin_str'''
40        loaded = False
41        origin_layer = 'Layer: ' + layer.fs_hash[:10]
42        if not layer.packages:  # add and not redo
43            # there are no packages in this layer  # and we are not re
44            # try to get it from the cache
```
Tracking each “good first issue” to an impactful change

Big Feature: Structured JSON output #100

- implement function to return a JSON object (done!)
- function to return collated JSON objects (done!)

#101
#102
#103
#104
#105
Participating in group events

1. Add test for to_dict method in test_class_notice.py
   - Assigned to nishakm
   - Created on Oct 9, 2018

2. Add to_dict method to docker_image class
   - Assigned to nishakm
   - Created on Oct 3, 2018

3. Fill to_dict method in Image class
   - Assigned to nishakm
   - Created on Oct 5, 2018

4. Complete to_dict method in image_layer class
   - Assigned to nishakm
   - Created on Oct 4, 2018

5. Add to_dict method to Origins class
   - Assigned to nishakm
   - Created on Oct 1, 2018
good-first-issue fails
Assigning dependent issues to new contributors

DO NOT put a new contributor in the critical path

Contributors are totally OK with you taking over

Contributors are NOT OK with you forcing them to adhere to your schedule
Picking a task of unreasonably large scope

Create abstract class Template #184

Closed nishakm opened this issue on Mar 19 · 1 comment

Looks like a small task on the surface
Poorly documenting a good-first-issue

First pass implementation of dict_from_template() method in a Package class. See #180 for an overview. This is a sub-issue of #163 and #174. Is dependent on #184 and #185

Lots of references for a contributor to read through

Dependent on two other issues

No guidance on implementation

It’s just sloppy
Takeaways

Impact:
Fosters a more inclusive community
Forces more disciplined code and project management
Grows the project and the community sustainably

Tern currently has 28 contributors so far (not including the maintainers)
6 of them are women
That’s 15% more than the rest of Open Source
Takeaways

Costs:

More work up-front

More planning than coding

Not a steady source of good-first-issues

Not guaranteed of retaining contributors
Thank You!

Tern: http://github.com/vmware/tern
github: @nishakm
twitter: @nishakmr, @vmwopensource
website: https://nishakm.github.io

Come see me at VMware’s Expo booth #31