Contributing to Galaxy

Dannon Baker, Nate Coraor, John Chilton

Overview

- Why contribute?
- How to contribute code
- Contributing without coding

Why Contribute?

- Galaxy core team is a small group with big goals
  - We have to prioritize our development efforts
- We have experience with usegalaxy.org
  - Which is completely unlike almost all other Galaxy instances
- You want things

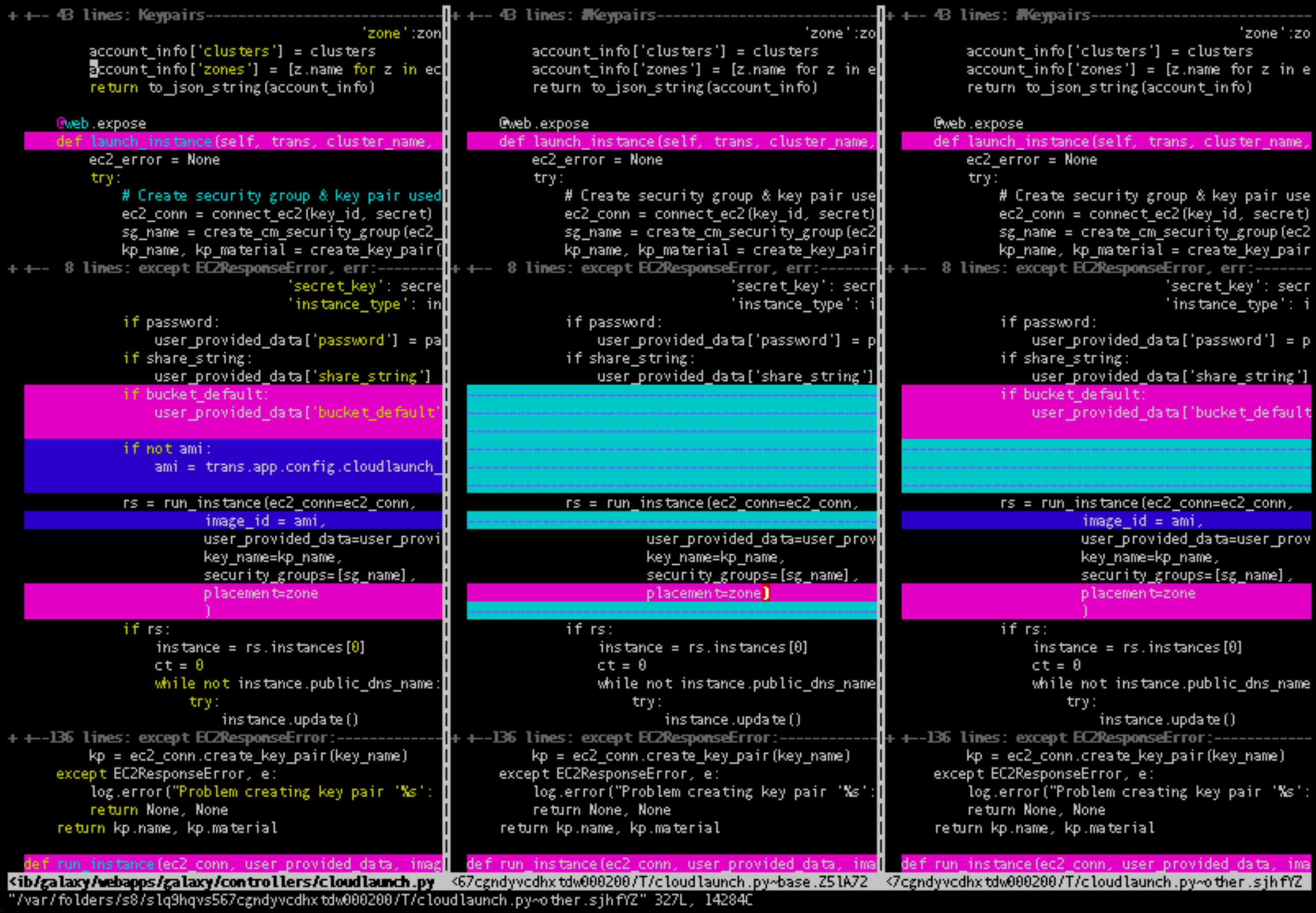
Divergence

Image: Kevin Pluck https://commons.wikimedia.org/wiki/File:Lion_waiting_in_Namibia.jpg

I was warned to avoid biological analogies

Image: Kenny Louie https://commons.wikimedia.org/wiki/File:Kitten_in_Rizal_Park,_Manila.jpg

Divergence - This is not fun



```
+ +-- 43 lines: Keypairs-------------------------------
                                          'zone':zon
        account_info['clusters'] = clusters
        account_info['zones'] = [z.name for z in ec
        return to_json_string(account_info)

    @web.expose
    def launch_instance(self, trans, cluster_name,
        ec2_error = None
        try:
            # Create security group & key pair used
            ec2_conn = connect_ec2(key_id, secret)
            sg_name = create_cm_security_group(ec2_
            kp_name, kp_material = create_key_pair(
+ +-- 8 lines: except EC2ResponseError, err:---------
                                 'secret_key': secre
                                 'instance_type': in
            if password:
                user_provided_data['password'] = pa
            if share_string:
                user_provided_data['share_string']
            if bucket_default:
                user_provided_data['bucket_default'

            if not ami:
                ami = trans.app.config.cloudlaunch_

            rs = run_instance(ec2_conn=ec2_conn,
                      image_id = ami,
                      user_provided_data=user_provi
                      key_name=kp_name,
                      security_groups=[sg_name],
                      placement=zone
                      )
            if rs:
                instance = rs.instances[0]
                ct = 0
                while not instance.public_dns_name:
                    try:
                        instance.update()
+ +--136 lines: except EC2ResponseError:--------------
        kp = ec2_conn.create_key_pair(key_name)
    except EC2ResponseError, e:
        log.error("Problem creating key pair '%s':
        return None, None
    return kp.name, kp.material

def run_instance(ec2_conn, user_provided_data, imag
<ib/galaxy/webapps/galaxy/controllers/cloudlaunch.py
```

```
+ +-- 43 lines: #Keypairs------------------------------
                                          'zone':zo
        account_info['clusters'] = clusters
        account_info['zones'] = [z.name for z in e
        return to_json_string(account_info)

    @web.expose
    def launch_instance(self, trans, cluster_name,
        ec2_error = None
        try:
            # Create security group & key pair use
            ec2_conn = connect_ec2(key_id, secret)
            sg_name = create_cm_security_group(ec2
            kp_name, kp_material = create_key_pair
+ +-- 8 lines: except EC2ResponseError, err:--------
                                 'secret_key': secr
                                 'instance_type': i
            if password:
                user_provided_data['password'] = p
            if share_string:
                user_provided_data['share_string']
            rs = run_instance(ec2_conn=ec2_conn,
                      user_provided_data=user_prov
                      key_name=kp_name,
                      security_groups=[sg_name],
                      placement=zone)
            if rs:
                instance = rs.instances[0]
                ct = 0
                while not instance.public_dns_name
                    try:
                        instance.update()
+ +--136 lines: except EC2ResponseError:-------------
        kp = ec2_conn.create_key_pair(key_name)
    except EC2ResponseError, e:
        log.error("Problem creating key pair '%s':
        return None, None
    return kp.name, kp.material

def run_instance(ec2_conn, user_provided_data, ima
<67cgndyvcdhxtdw000200/T/cloudlaunch.py~base.Z51A72
```

```
+ +-- 43 lines: #Keypairs------------------------------
                                          'zone':zo
        account_info['clusters'] = clusters
        account_info['zones'] = [z.name for z in e
        return to_json_string(account_info)

    @web.expose
    def launch_instance(self, trans, cluster_name,
        ec2_error = None
        try:
            # Create security group & key pair use
            ec2_conn = connect_ec2(key_id, secret)
            sg_name = create_cm_security_group(ec2
            kp_name, kp_material = create_key_pair
+ +-- 8 lines: except EC2ResponseError, err:--------
                                 'secret_key': secr
                                 'instance_type': i
            if password:
                user_provided_data['password'] = p
            if share_string:
                user_provided_data['share_string']
            if bucket_default:
                user_provided_data['bucket_default
            rs = run_instance(ec2_conn=ec2_conn,
                      image_id = ami,
                      user_provided_data=user_prov
                      key_name=kp_name,
                      security_groups=[sg_name],
                      placement=zone
                      )
            if rs:
                instance = rs.instances[0]
                ct = 0
                while not instance.public_dns_name
                    try:
                        instance.update()
+ +--136 lines: except EC2ResponseError:-------------
        kp = ec2_conn.create_key_pair(key_name)
    except EC2ResponseError, e:
        log.error("Problem creating key pair '%s':
        return None, None
    return kp.name, kp.material

def run_instance(ec2_conn, user_provided_data, ima
<7cgndyvcdhxtdw000200/T/cloudlaunch.py~other.sjhfYZ
```

```
"/var/folders/s8/slq9hqvs567cgndyvcdhxtdw000200/T/cloudlaunch.py~other.sjhfYZ" 327L, 14284C
```

So you want to be a Hero (contributor)

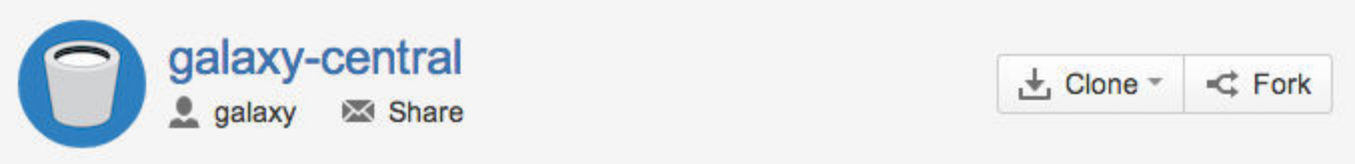
1. Awesome.

2. Think about:
   - Security
   - Performance
   - General Applicability
   - Communicate Early and Often
     - Trello
     - Mailing List
     - IRC

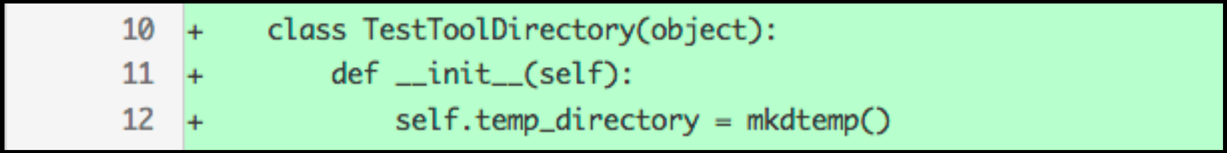
Using Trello For R&D

HOWTO in a nutshell

Fork it.

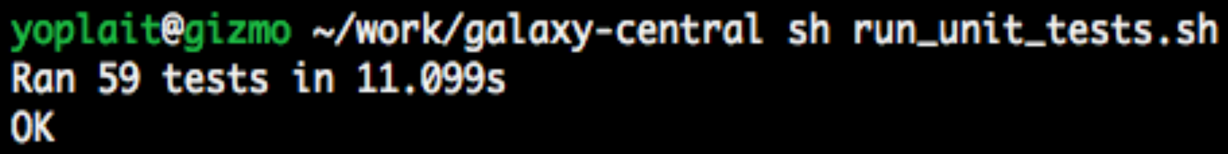


Tweak it.



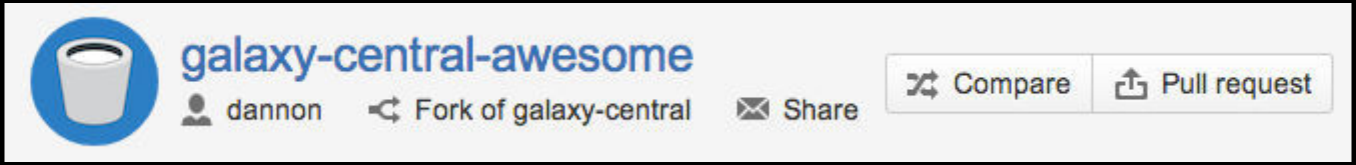
```
+    class TestToolDirectory(object):
+        def __init__(self):
+            self.temp_directory = mkdtemp()
```

Test it.



```
yoplait@gizmo ~/work/galaxy-central sh run_unit_tests.sh
Ran 59 tests in 11.099s
OK
```

Pull it.



Writing code that's easy to integrate

- Readable
- Commented
- Detailed commit messages
- Preserves existing functionality
  - Config conditional!
- Includes unit and/or functional tests

```
           globals()
      .update({   "______":
   lambda x:         globals()
   .update((         dict([[x]
      *2]))}),   ______(((
            "Just")))
      ,______((   "another"
   )),______         ("Python"
),______(               "Hacker")
];print (               " ".join(
   [(Just),(         (another)
      ),(Python   ),Hacker]
            ));______
```

credit: http://c2.com/cgi/wiki?ObfuscatedPython

A foolish consistency is the hobgoblin of little minds

Galaxy mostly follows PEP8, readability is the ultimate goal

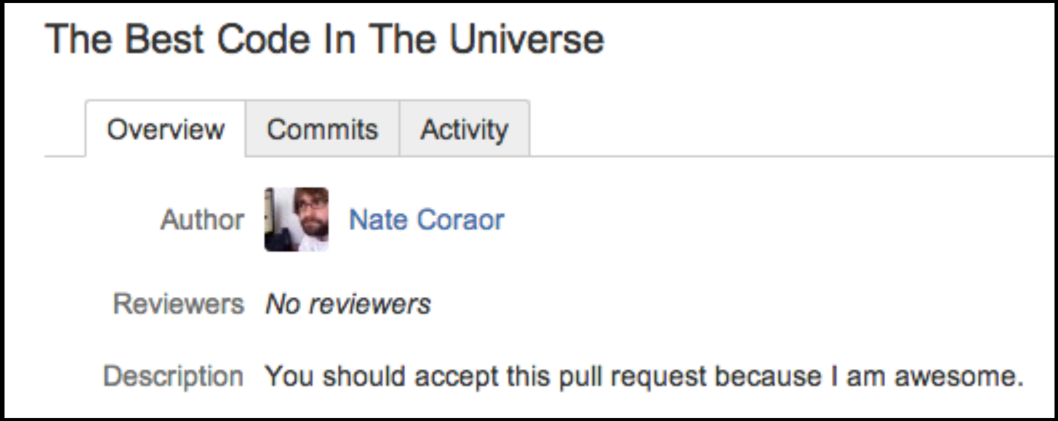
- Avoid `from module import *`
- thisIsNotJava underscores_strongly_preferred
- Comment lines should be under 79 characters, code lines can be up to 200 characters
- Docstrings need to be reStructured Text and Sphinx markup compatible
- Whitespace: whatever is most readable, both for blank lines and space around operators

Writing code that is easy to reject

- Tools
- Hard-to-read code
- Code that negatively impacts existing functionality to solve your specific case
- Suboptimal implementations that inhibit comprehensive work in the future

Writing Pull Requests that are easy to integrate

- Informative pull request message



The Best Code In The Universe

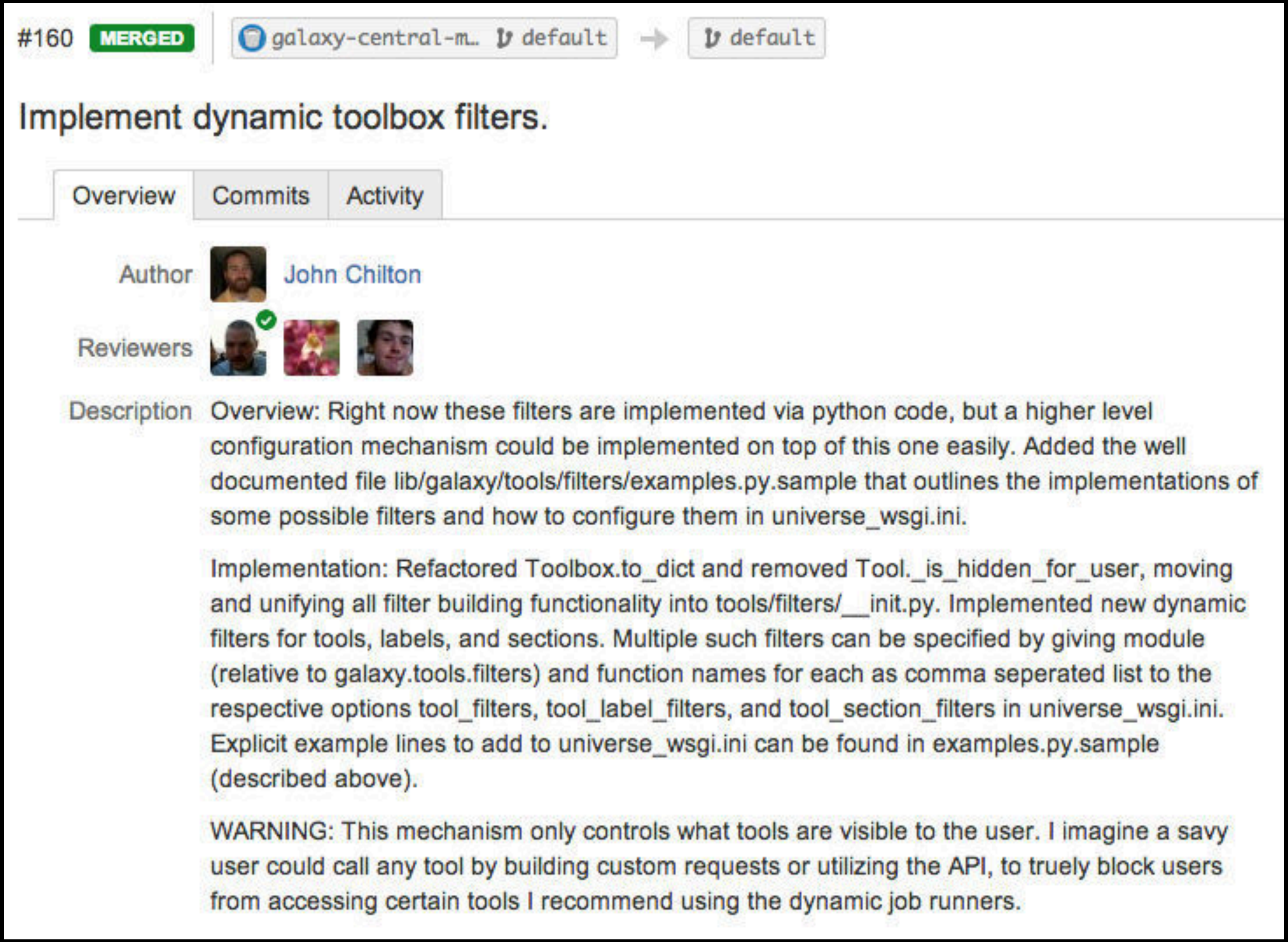
Author Nate Coraor

Reviewers *No reviewers*

Description You should accept this pull request because I am awesome.

- Context
  - New functionality: test cases
  - Bug fixes: how to reproduce
- Links to Trello cards as applicable
- Tag specific individuals as reviewers

An Almost Perfect Example



Implement dynamic toolbox filters.

Author John Chilton

Reviewers

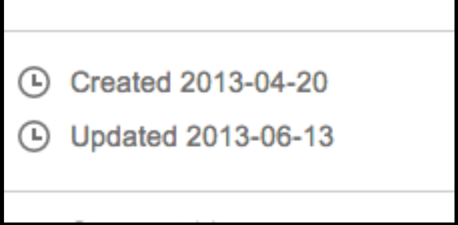
Description Overview: Right now these filters are implemented via python code, but a higher level configuration mechanism could be implemented on top of this one easily. Added the well documented file lib/galaxy/tools/filters/examples.py.sample that outlines the implementations of some possible filters and how to configure them in universe_wsgi.ini.

Implementation: Refactored Toolbox.to_dict and removed Tool._is_hidden_for_user, moving and unifying all filter building functionality into tools/filters/__init.py. Implemented new dynamic filters for tools, labels, and sections. Multiple such filters can be specified by giving module (relative to galaxy.tools.filters) and function names for each as comma seperated list to the respective options tool_filters, tool_label_filters, and tool_section_filters in universe_wsgi.ini. Explicit example lines to add to universe_wsgi.ini can be found in examples.py.sample (described above).

WARNING: This mechanism only controls what tools are visible to the user. I imagine a savy user could call any tool by building custom requests or utilizing the API, to truely block users from accessing certain tools I recommend using the dynamic job runners.

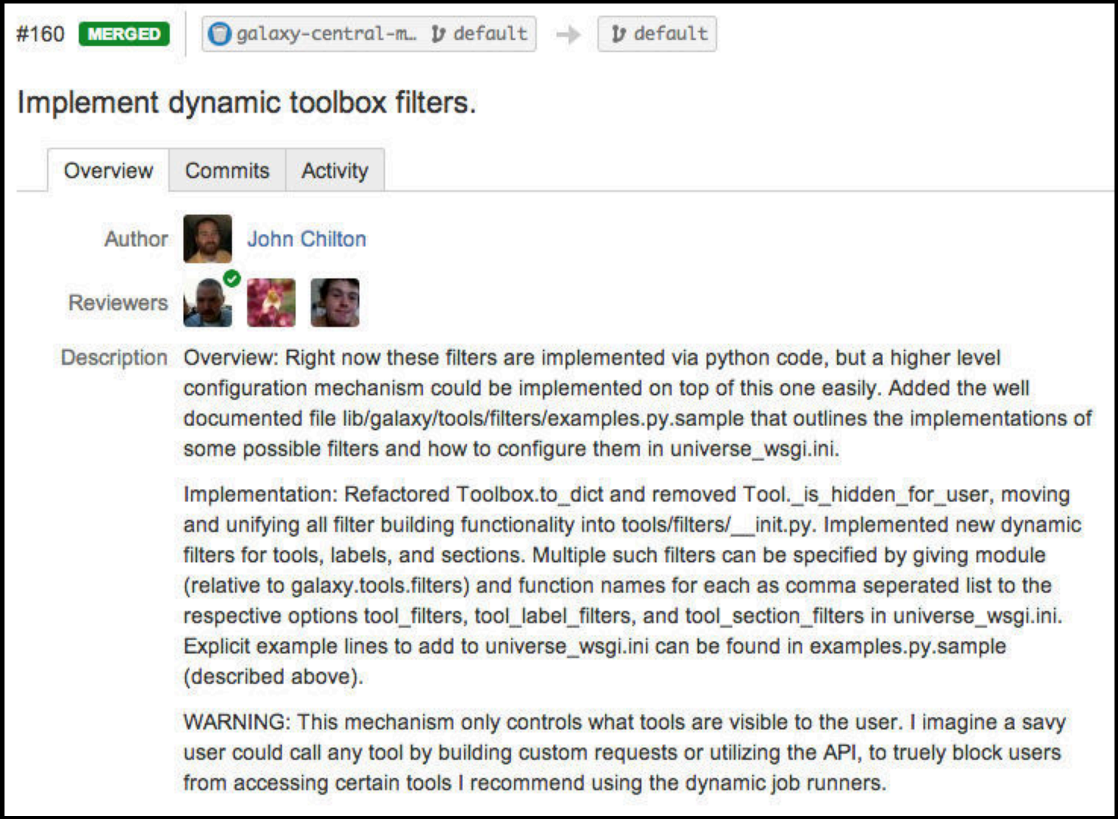
An Almost Perfect Example

The imperfection ->



Created 2013-04-20

Updated 2013-06-13



Implement dynamic toolbox filters.

Author: John Chilton

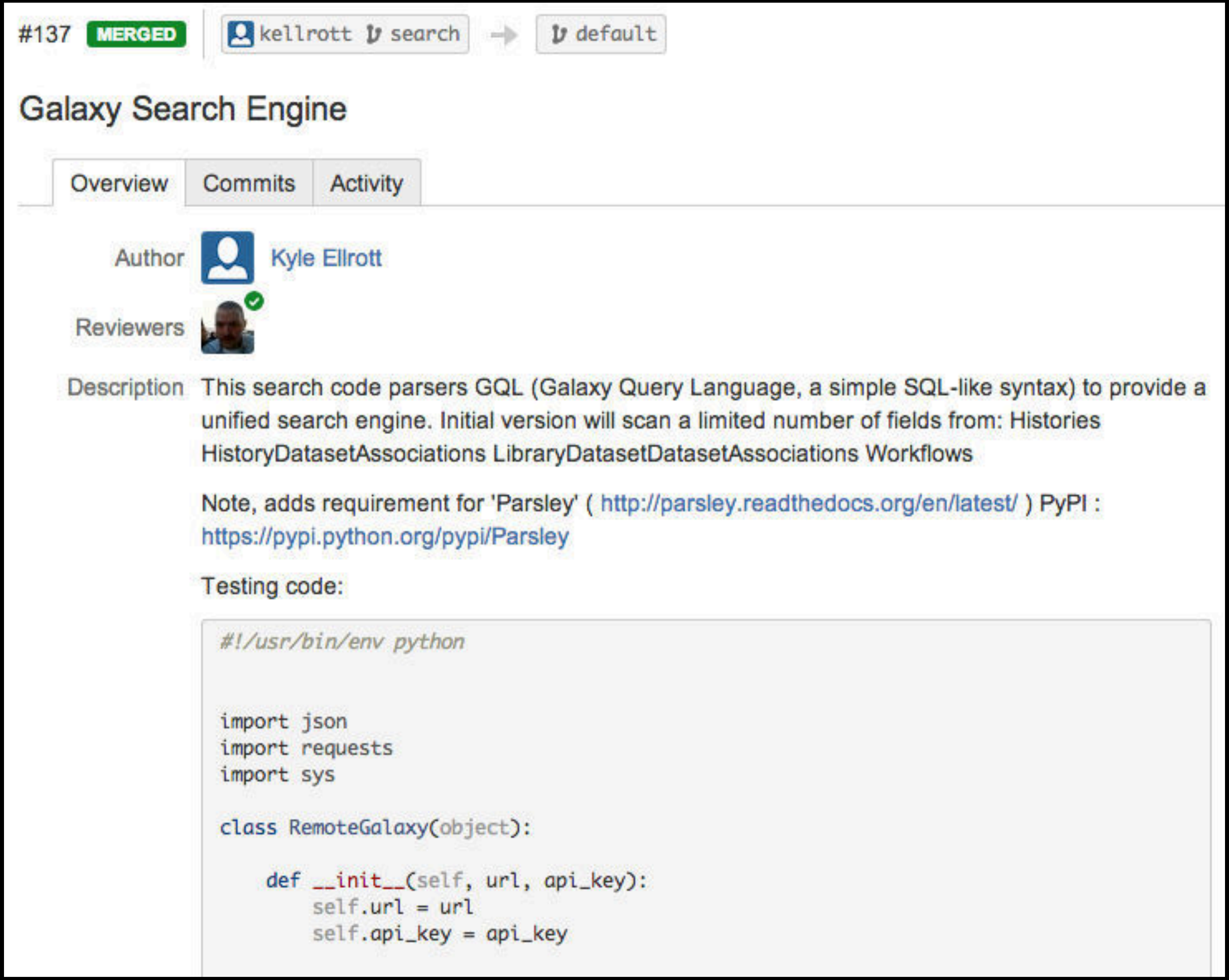
Reviewers

Description: Overview: Right now these filters are implemented via python code, but a higher level configuration mechanism could be implemented on top of this one easily. Added the well documented file lib/galaxy/tools/filters/examples.py.sample that outlines the implementations of some possible filters and how to configure them in universe_wsgi.ini.

Implementation: Refactored Toolbox.to_dict and removed Tool._is_hidden_for_user, moving and unifying all filter building functionality into tools/filters/__init.py. Implemented new dynamic filters for tools, labels, and sections. Multiple such filters can be specified by giving module (relative to galaxy.tools.filters) and function names for each as comma seperated list to the respective options tool_filters, tool_label_filters, and tool_section_filters in universe_wsgi.ini. Explicit example lines to add to universe_wsgi.ini can be found in examples.py.sample (described above).

WARNING: This mechanism only controls what tools are visible to the user. I imagine a savy user could call any tool by building custom requests or utilizing the API, to truely block users from accessing certain tools I recommend using the dynamic job runners.

Tests and Demo code



Galaxy Search Engine

Author Kyle Ellrott

Reviewers

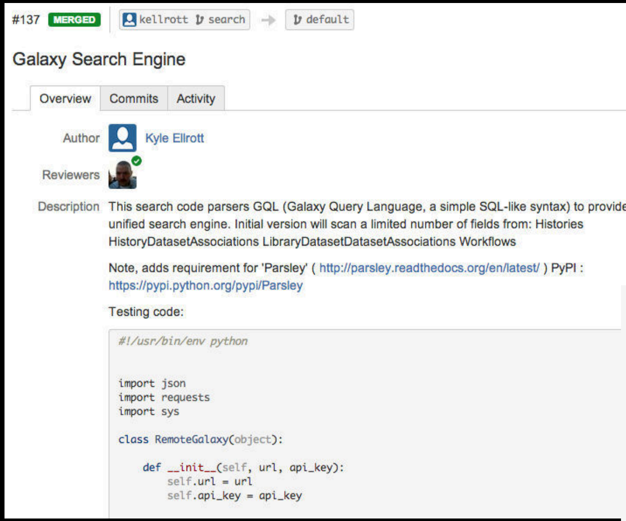
Description This search code parsers GQL (Galaxy Query Language, a simple SQL-like syntax) to provide a unified search engine. Initial version will scan a limited number of fields from: Histories HistoryDatasetAssociations LibraryDatasetDatasetAssociations Workflows

Note, adds requirement for 'Parsley' (http://parsley.readthedocs.org/en/latest/) PyPI : https://pypi.python.org/pypi/Parsley

Testing code:

```
#!/usr/bin/env python


import json
import requests
import sys

class RemoteGalaxy(object):

    def __init__(self, url, api_key):
        self.url = url
        self.api_key = api_key
```

Tests and Demo code



#137 MERGED kellrott search → default

Galaxy Search Engine

Author Kyle Ellrott

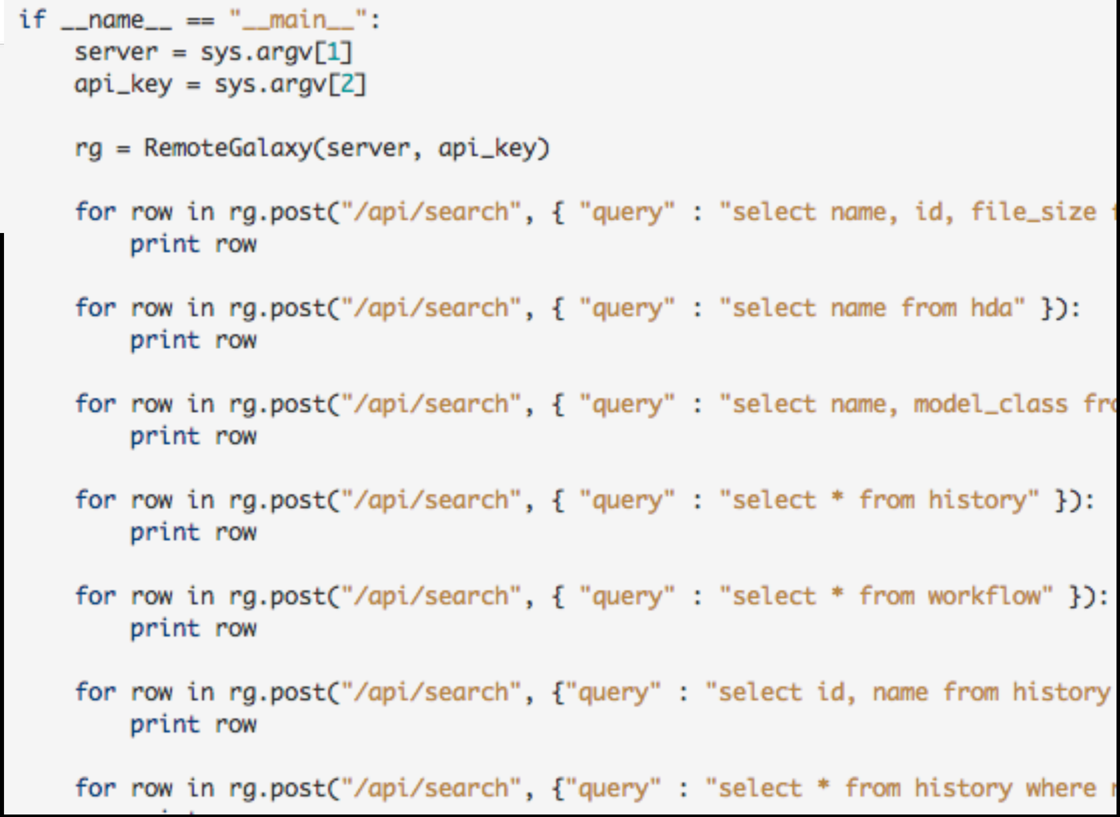
Reviewers

Description This search code parsers GQL (Galaxy Query Language, a simple SQL-like syntax) to provide a unified search engine. Initial version will scan a limited number of fields from: Histories HistoryDatasetAssociations LibraryDatasetDatasetAssociations Workflows

Note, adds requirement for 'Parsley' (http://parsley.readthedocs.org/en/latest/) PyPI : https://pypi.python.org/pypi/Parsley

Testing code:

```
#!/usr/bin/env python


import json
import requests
import sys

class RemoteGalaxy(object):

    def __init__(self, url, api_key):
        self.url = url
        self.api_key = api_key
```

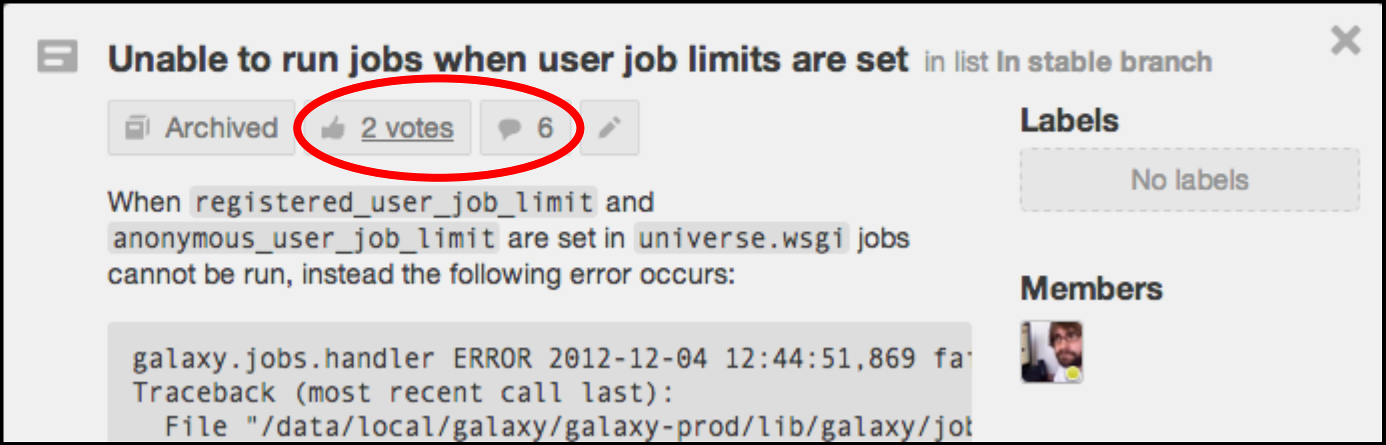


```
if __name__ == "__main__":
    server = sys.argv[1]
    api_key = sys.argv[2]

    rg = RemoteGalaxy(server, api_key)

    for row in rg.post("/api/search", { "query" : "select name, id, file_size f
        print row

    for row in rg.post("/api/search", { "query" : "select name from hda" }):
        print row

    for row in rg.post("/api/search", { "query" : "select name, model_class fr
        print row

    for row in rg.post("/api/search", { "query" : "select * from history" }):
        print row

    for row in rg.post("/api/search", { "query" : "select * from workflow" }):
        print row

    for row in rg.post("/api/search", {"query" : "select id, name from history
        print row

    for row in rg.post("/api/search", {"query" : "select * from history where r
```

Contributing without Coding

- Mailing List (galaxy-dev)
  - wiki.galaxyproject.org/MailingLists
- Trello
  - galaxyproject.org/trello
- Wiki
  - wiki.galaxyproject.org
- Pull Request Review
  - bitbucket.org/galaxy/galaxy-central/pull-requests

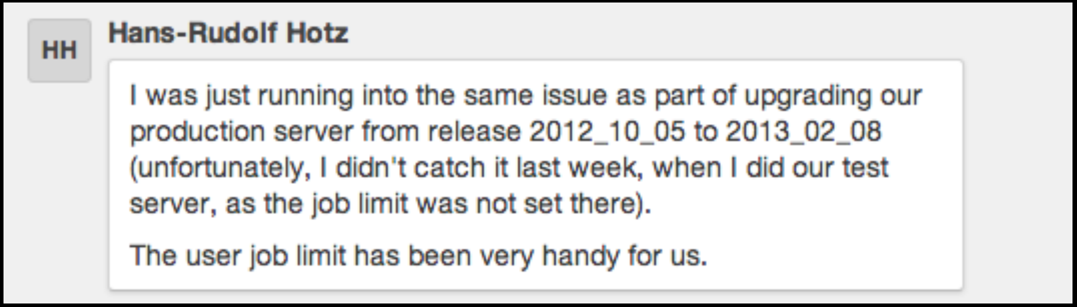
Using Trello For Friendly Coercion



Unable to run jobs when user job limits are set in list **In stable branch**

When `registered_user_job_limit` and `anonymous_user_job_limit` are set in `universe.wsgi` jobs cannot be run, instead the following error occurs:

```
galaxy.jobs.handler ERROR 2012-12-04 12:44:51,869 fa
Traceback (most recent call last):
  File "/data/local/galaxy/galaxy-prod/lib/galaxy/jo
```



Hans-Rudolf Hotz

I was just running into the same issue as part of upgrading our production server from release 2012_10_05 to 2013_02_08 (unfortunately, I didn't catch it last week, when I did our test server, as the job limit was not set there).

The user job limit has been very handy for us.

Vote!

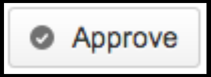
Comment!

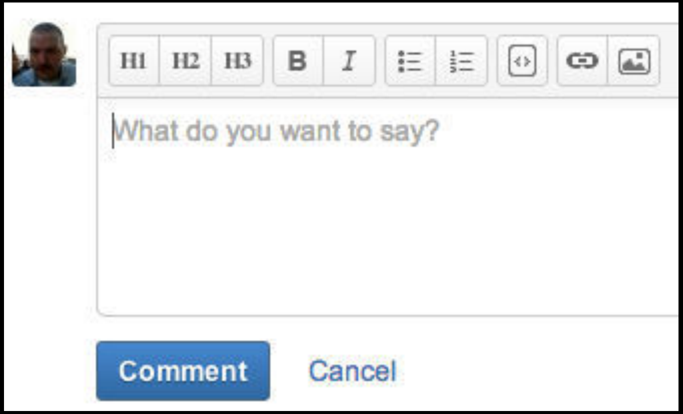
Make your impassioned plea!

Make sure we know what matters to you!

Reviewing Pull Requests

- Yes, please?
- Consider the utility of the PR
    - Is this something that, included in Galaxy, would benefit the community at large?
- Pull the code
- Test, look for corner cases
- Errors or Improvements?
- Comment
- Approve!





Wrapping Up

- Small team, large and diverse community
- Pull requests take time to review (help us!)
- Tell us how we can improve the process

Thank You

Florent Angly
Raj Ayyampalayam
Finn Bacall
James Boocock
Carlos Borroto
Brad Chapman
John Chilton
Ilya Chorny
Saket Choudhary
Peter Cock
Freek de Bruijn
John Duddy
Kyle Ellrott

Jaime Frey
Mike Gehring
Nuwan Goonasekera
Björn Grüning
MORITA Hideyuki
Rob Hooft
Gert Hulselmans
Joachim Jacob
Jim Johnson
Rory Kirchner
Brad Langhorst
Philip Mabon
Eric Paniagua

Richard Park
Lance Parsons
Andrea Sbardellati
Nick Semenkovich
Clare Sloggett
Nicola Soranzo
Roy Storey
Hanfei Sun
Tomithy Too
Peter van Heusden
Martijn Vermaat
Hiral Vora
Trevor Wennblom

This page intentionally left blank

So you found a bug (and want to fix it)

Saw the Cytoscape rendering, and you're not fazed? Great!

<<NEED CONTENT>>

(http://wiki.galaxyproject.org/Develop)