Production Operations Management Service (POMS) - Feature #12934

Code Cleanup -- Factor out poms_service.py

06/15/2016 10:59 AM - Marc Mengel

Status:	Closed	**Start date:**	06/15/2016
Priority:	Normal	**Due date:**	
Assignee:	Andres Alba hernandez	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	v1_1_0	**Spent time:**	0.00 hour
Scope:	Internal	**Stakeholders:**	
Experiment:	-		

Description

Nearly all the methods in poms_service should look like:

```
   def method_name(arguments...):
        vars = self.blah.method_name(cherrypy.request.db, arguments...)
        template = self.jinja_env.get_template('method_name.html')
        vars.update( {
            'current_experimenter': cherrypy.session.get('experimenter'),
            'pomspath':self.path,
            'help_page':"MethodNameHelp"
           })
        return template.render(**vars)
```

with their guts factored out into other classes, which should have unit tests.

Related issues:

Related to Production Operations Management Service (POMS) - Task #14708: Imp...	**Closed**	**11/29/2016**

History

#1 - 06/28/2016 09:51 AM - Marc Mengel

So here's a potential grouping.

leave in/split up

- def *init*(self): * def headers(self):

security

- def can_create_task(self): * def can_report_data(self): * def can_db_admin(self):

downtimes

- def index(self): * def calendar_json(self, start, end, timezone, _): * def calendar(self): * def add_event(self, title, start, end): * def edit_event(self, title, start, new_start, end, s_id): #even though we pass in the s_id we should not rely on it because they can and will change the service name * def service_downtimes(self): * def update_service(self, name, parent, status, host_site, total, failed, description): * def new_task_for_campaign(self, campaign_name, command_executed, experimenter_name, dataset_name = None): * def service_status(self, under = 'All'): * def service_status_hier(self, under = 'All', depth = 0):

raw_tables

- def make_admin_map(self): * def raw_tables(self): * def list_generic(self, classname): * def edit_screen_generic(self, classname, id = None): * def update_generic(self, classname, *args, **kwargs): * def update_for(self, classname, eclass, primkey, *args , **kwargs): * def edit_screen_for(self, classname, eclass, update_call, primkey, primval, valmap): * def make_list_for(self,eclass,primkey):

experimenters

- def user_edit(self, *args, **kwargs): * def experiment_members(self, *args, **kwargs): * def experiment_edit(self, message=None): * def experiment_authorize(self, *args, **kwargs):

fancy_tables

- def launch_template_edit(self, *args, **kwargs): * def campaign_definition_edit(self, *args, **kwargs): * def campaign_edit(self, *args, **kwargs): * def campaign_edit_query(self, *args, **kwargs): * def create_task(self, experiment, taskdef, params, input_dataset, output_dataset, creator, waitingfor = None):

job_task_interface

- def active_jobs(self): * def report_declared_files(self, flist): * def output_pending_jobs(self): * def wrapup_tasks(self): * def compute_status(self, task): * def update_job(self, task_id = None, jobsub_job_id = 'unknown', **kwargs): * def get_task_id_for(self, campaign, user = None, experiment = None, command_executed = "", input_dataset = "", parent_task_id=None): * def register_poms_campaign(self, experiment, campaign_name, version, user = None, campaign_definition = None, dataset = "", role = "Analysis", params = []): * def kill_jobs(self, campaign_id=None, task_id=None, job_id=None, confirm=None): * def get_dataset_for(self, camp): * def set_job_launches(self, hold): * def launch_dependents_if_needed(self, task_id): * def launch_recovery_if_needed(self, task_id): * def launch_jobs(self, campaign_id, dataset_override = None, parent_task_id = None): * def list_launch_file(self, campaign_id, fname): * def schedule_launch(self, campaign_id): * def update_launch_schedule(self, campaign_id, dowlist = None, domlist = None, monthly = None, month = None, hourlist = None, submit = None , minlist = None, delete = None): * def snapshot_parts(self, campaign_id):

util

- def handle_dates(self,tmin, tmax, tdays, baseurl): * def jump_to_job(self, jobsub_job_id, **kwargs): * def task_min_job(self, task_id): * def test_job_counts(self, task_id = None, campaign_id = None): * def quick_search(self, search_term):

campaign_task_monitoring

- def show_task_jobs(self, task_id, tmax = None, tmin = None, tdays = 1): * def *init*(self, **kwargs): * def show_campaigns(self,tmin = None, tmax = None, tdays = 1): * def campaign_info(self, campaign_id): * def campaign_time_bars(self, campaign_id, tmin = None, tmax = None, tdays = 1): * def jobs_eff_histo(self, campaign_id, tmax = None, tmin = None, tdays = 1):

file_status

- def list_task_logged_files(self, task_id): * def campaign_task_files(self,campaign_id, tmin = None, tmax = None, tdays = 1): * def *init* (self, **kwargs): * def job_file_list(self, job_id,force_reload = False): * def job_file_contents(self, job_id, task_id, file, tmin = None, tmax = None, tdays = None): * def actual_pending_files(self, count_or_list, task_id = None, campaign_id = None, tmin = None, tmax= None, tdays = 1): * def format_job_counts(self, task_id = None, campaign_id = None, tmin = None, tmax = None, tdays = 7, range_string = None): * def json_project_summary_for_task(self, task_id): * def project_summary_for_task(self, task_id): * def project_summary_for_tasks(self, task_list): * def show_dimension_files(self, experiment, dims): * def inflight_files(self, campaign_id=None, task_id=None): * def campaign_sheet(self, campaign_id, tmin = None, tmax = None , tdays = 7): * def get_inflight(self, campaign_id=None, task_id=None):

triage

- def job_counts(self, task_id = None, campaign_id = None, tmin = None, tmax = None, tdays = None): * def job_table(self, tmin = None, tmax = None, tdays = 1, task_id = None, campaign_id = None , experiment = None, sift=False, campaign_name=None, name=None,campaign_def_id=None, vo_role=None, input_dataset=None, output_dataset=None, task_status=None, project=None, jobsub_job_id=None, node_name=None, cpu_type=None, host_site=None, job_status=None, user_exe_exit_code=None, output_files_declared=None, campaign_checkbox=None, task_checkbox=None, job_checkbox=None, ignore_me = None, keyword=None, dataset = None, eff_d = None): * def jobs_by_exitcode(self, tmin = None, tmax = None, tdays = 1): * def failed_jobs_by_whatever(self, tmin = None, tmax = None, tdays = 1 , f = [], go = None): * def triage_job(self, job_id, tmin = None, tmax = None, tdays = None, force_reload = False):

tags

- def link_tags(self, campaign_id, tag_name, experiment): * def delete_campaigns_tags(self, campaign_id, tag_id, experiment): * def search_tags(self, q): * def auto_complete_tags_search(self, experiment, q):

#2 - 10/24/2016 04:52 PM - Marc Mengel

- Assignee set to Andres Alba hernandez

- % Done changed from 0 to 60

#3 - 12/05/2016 03:51 PM - Andres Alba hernandez

- % Done changed from 60 to 80

#4 - 12/13/2016 11:09 AM - Vladimir Podstavkov

- Related to Task #14708: Implement row limit + paging on POMS job_table page added

#5 - 12/14/2016 03:10 PM - Anna Mazzacane

- Status changed from New to Work in progress

#6 - 12/15/2016 11:33 AM - Andres Alba hernandez

- % Done changed from 80 to 90

The clean of the code is almost complete, the pending task are testing and define what to do with -- def service_status_hier

#7 - 12/16/2016 01:42 PM - Marc Mengel

- Status changed from Work in progress to Resolved

- % Done changed from 90 to 100

Refactored code is now running on dev instance on fermicloud045.
A this point I'm declaring the refactoring complete.

#8 - 12/21/2016 03:59 PM - Anna Mazzacane

- Status changed from Resolved to Closed