Pulp CLI - Task #3836

Convert from coreapi to pyswagger

07/06/2018 10:21 PM - daviddavis

Status:	CLOSED - WONTFIX	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:	Proof of concept		
Platform Release:		**Tags:**	
Groomed:	Yes	**Sprint:**	
Sprint Candidate:	No	**Quarter:**	

Description

Why do we need to convert to pyswagger?

The schema that we use for documentation and bindings are openapi based. PySwagger is a python swagger client that reads in schema, and makes calls to the REST API based on that schema.

Pyswagger makes it so we don't need to typecheck body parameters, and we don't need to format query and path parameters in URLs.

Creating the nested Click commands.

Every openapi operation has its own operationId. These OperationIds can be accessed from pyswagger like so:

```
app = App.create(DOCUMENT_PATH)
app.op.keys()
```

This returns something like:

```
['repositories!##!repositories_versions_removed_content',
'repositories!##!repositories_versions_content',
'repositories!##!repositories_versions_added_content',
'repositories!##!repositories_versions_delete',
'repositories!##!repositories_versions_read',
'repositories!##!repositories_versions_create',
'repositories!##!repositories_versions_list',
'repositories!##!repositories_partial_update',
'repositories!##!repositories_delete',
'repositories!##!repositories_update',
'repositories!##!repositories_read',
'repositories!##!repositories_create',
'repositories!##!repositories_list',
'remotes!##!remotes_create',
...
]
```

Individual operations are separated via '_', this is less than ideal, since some of our operations contain '_' like 'added_content'.

A solution would be to separate this list with \ \#\#\, group all the common operations together (things to the left of \ \#\#\), and find the longest common startswith substring for any subset of the list (care should be taken that this substing has to be separated by a _, since we don't want repo_versions, and repositories to return repo as a substring).

This longest common substring should be a 'group' that other commands nest under. This substring should then be removed from the rest of the items in the the subset, and if there is no more shared substrings, everything can be added as a command.

Getting indivudual operation parameters

Individual parameters can be found on

```
app.op['repositories!##!repositories_update'].parameters
```

Usually this is a data parameter (for body params) and whatever path params are in the URL.

The data param has a schema reference that can be accessed

```
getattr(app.op['repositories!##!repositories_create'].parameters[0].schema, '$ref')
```

And a pyswagger Resolver can be used to load up that reference and get the necessary parameters

```
>>> from pyswagger.resolve import Resolver
>>> res = Resolver()
>>> res.resolve(getattr(app.op['repositories!##!repositories_create'].parameters[0].schema, '$ref'
))
{
    'required':[
        'name'
    ],
    'type':'object',
    'properties':{
        'id':{
            'title':'Id',
            'type':'string',
            'format':'uuid',
            'readOnly':True
        },
        '_href':{
            'title':' href',
            'type':'string',
            'format':'uri',
            'readOnly':True
        },
        'created':{
            'title':'Created',
            'description':'Timestamp of creation.',
            'type':'string',
            'format':'date-time',
            'readOnly':True
        },
        '_versions_href':{
            'title':' versions href',
            'type':'string',
            'format':'uri',
            'readOnly':True
        },
        '_latest_version_href':{
            'title':' latest version href',
            'type':'string',
            'format':'uri',
            'readOnly':True
        },
        'name':{
            'title':'Name',
```

```
            'description':'A unique name for this repository.',
            'type':'string',
            'minLength':1
        },
        'description':{
            'title':'Description',
            'description':'An optional description.',
            'type':'string'
        },
        'notes':{
            'title':'Notes',
            'description':'
A mapping of string keys to string values, for storing notes on this object.',
            'type':'object',
            'additionalProperties':{
                'type':'string',
                'minLength':1
            }
        }
    }
}
```

Related issues:

Related to Pulp CLI - Issue #3839: Handle file parameters	**CLOSED - WONTFIX**

History

#1 - 07/06/2018 10:21 PM - daviddavis

- Description updated

#2 - 07/06/2018 10:24 PM - daviddavis

- Sprint/Milestone set to Proof of concept

#3 - 07/10/2018 09:55 PM - bizhang

- Description updated

#4 - 07/10/2018 10:05 PM - bizhang

- Sprint Candidate changed from No to Yes

#5 - 07/10/2018 10:08 PM - bizhang

- Related to Issue #3839: Handle file parameters added

#6 - 07/10/2018 11:35 PM - CodeHeeler

- Groomed changed from No to Yes

#7 - 07/10/2018 11:42 PM - daviddavis

A couple concerns with the nesting. Suppose I have a path like "a/b_c/d" with it's the only endpoint under b_c. The operation id would be "a ##b_c_d". How would the CLI know to form the command pulp a b_c d and not pulp a b_c_d?

Also, if you had the paths "/z/a/b_c/" and "/z/a_b/d" which would form the operations "z ##a_b_c" and "a ##a_b_d", I think they would incorrectly get nested together under pulp z a_b.

I guess these issues aren't a big deal if we assume nesting is based on commonalities in operation names and not paths in the REST API.

Alternatively though, I think we could potentially look at the path property (e.g. /repositories/{repository_pk}/versions/{number}/added_content/) on app.op values to decide how to nest commands.

#8 - 07/11/2018 05:17 AM - bizhang

The operation id for POSTing to /artifacts/ is artifacts_create, and the operation id for GET /artifacts/ is artifacts_list, and the operation id for GET /artifacts/{id}/ is artifacts_read.
If we go off the path id, we also have to take available REST verbs into account (map POST-> create, GET->{list, read} depending on context).

The best solution is probably use a combination of both the path and the operation ids.

Path can be split on '/' and used to construct the nested CLI groups, and the operation ids can be used to get the commands (read, list, delete, update, etc). available to these groups.

There is some complexity in the case of 'repositories_versions_added_content' since it doesn't have any additional commands so added_content would have to be a command on repositories versions instead of an additional group.

#9 - 07/11/2018 05:18 AM - bizhang

- Description updated

#10 - 07/11/2018 12:47 PM - daviddavis

Sounds good. +1.

#11 - 07/16/2018 01:12 AM - dkliban@redhat.com

- Sprint set to Sprint 40

#12 - 07/23/2018 08:37 PM - bizhang

- Status changed from NEW to ASSIGNED

- Assignee set to bizhang

#13 - 08/06/2018 03:16 PM - rchan

- Sprint changed from Sprint 40 to Sprint 41

#14 - 08/24/2018 03:20 PM - bizhang

- Status changed from ASSIGNED to NEW

- Assignee deleted (bizhang)

Unassigning since the pulp3 CLI work has been deprioritized

#15 - 08/24/2018 03:47 PM - daviddavis

- Sprint deleted (Sprint 41)

#16 - 08/24/2018 04:04 PM - daviddavis

- Sprint Candidate changed from Yes to No

#17 - 04/29/2020 09:23 PM - daviddavis

- Status changed from NEW to CLOSED - WONTFIX