Arvados - Story #9498

[Workbench] Show top-level container_requests in project jobs&pipelines tab and omit jobs from this tab.

06/28/2016 03:07 PM - Radhika Chippada

Status:	Resolved	**Start date:**	06/28/2016
Priority:	Normal	**Due date:**	
Assigned To:	Radhika Chippada	**% Done:**	100%
Category:	Workbench	**Estimated time:**	0.00 hour
Target version:	2016-07-06 sprint		

Description

Update the "jobs and pipelines" tab display:

- Display pipelines and top-level container_requests
- Do not include jobs in this tab anymore
- Rename the tab as "Pipelines and processes" similar to dashboard.

Subtasks:

Task # 9509: Review branch 9498-containers-in-project-tab	**Resolved**

Associated revisions

Revision 05b38a78 - 07/04/2016 09:23 PM - Radhika Chippada

closes #9498
Merge branch '9498-containers-in-project-tab'

Revision 4786aea0 - 09/29/2016 03:09 PM - Radhika Chippada

refs #9498 Fix failing workbench integration test due to containers fixture update.

History

#1 - 06/28/2016 03:09 PM - Radhika Chippada

- Description updated

#2 - 06/28/2016 04:54 PM - Radhika Chippada

- Category set to Workbench

- Status changed from New to In Progress

- Assigned To set to Radhika Chippada

- Target version set to 2016-07-06 sprint

#3 - 06/28/2016 04:55 PM - Radhika Chippada

- Story points set to 1.0

#4 - 06/29/2016 02:45 PM - Tom Clegg

I'm not sure about the change to load_searchable_objects. It looks convenient for Workbench's current needs ("only show top-level container requests"), but what about when Workbench or some other client wants to search all items in the project, including the "child" CRs?

I can think of two ways to make this explicit:

- Filter out the child CRs on the client side (but this will be slow if there are lots of child CRs)
- Accept filters like ["container_requests.requesting_container_uuid","=",nil], at least in GroupsController#index: load_searchable_objects could make a table-specific @filters before calling apply_where_limit_order_params, omitting any filters that specify a table other than klass.table_name.

Thoughts?

Two minor things

apps/workbench/app/views/projects/_show_pipelines_and_processes.html.erb has some forbidden tab characters.

Comment in infinite_scroll.js should probably say "container requests and pipeline instances" instead of "pipelines and processes".

#5 - 06/29/2016 11:21 PM - Radhika Chippada

> I'm not sure about the change to load_searchable_objects. It looks convenient for Workbench's current needs ("only show top-level container requests"), but what about when Workbench or some other client wants to search all items in the project, including the "child" CRs? I can think of two ways to make this explicit:
>
> > Filter out the child CRs on the client side (but this will be slow if there are lots of child CRs)

Yes, I do not want to do something that is bad for performance knowingly

> > Accept filters like ["container_requests.requesting_container_uuid","=",nil], at least in GroupsController#index: load_searchable_objects could make a table-specific @filters before calling apply_where_limit_order_params, omitting any filters that specify a table other than klass.table_name.

Added this in groups_controller. Currently this supports only '=' operator. That is, ['table_name.column_name', '=', 'some value']

> apps/workbench/app/views/projects/_show_pipelines_and_processes.html.erb has some forbidden tab characters.

Corrected this

> Comment in infinite_scroll.js should probably say "container requests and pipeline instances" instead of "pipelines and processes".

Updated accordingly

- I am assuming we need to document the fact that we will support filters such as ['table_name.column_name', '=', 'some value'] ?

#6 - 06/30/2016 06:13 PM - Tom Clegg

Radhika Chippada wrote:

> Added this in groups_controller. Currently this supports only '=' operator. That is, ['table_name.column_name', '=', 'some value']

It looks like ["table_name.column_name","!=","foo"] will be taken to mean ["table_name.column_name","=","foo"], which doesn't seem right (if we can't do what the client asks, we should send an error rather than just doing something different).

But more generally: rather than writing more code to translate filters to ActiveRecord queries, why not just put the desired set of filters in @filters and let apply_where_limit_order_params translate them with the full set of features? That way, we'd support all of the operators using the same code we use for other queries. I thought something like this would work:

```
request_filters = @filters
[...].each do |klass|
  ...
  @filters = request_filters.map do |col, op, val|
    if !col.index('.')
      [col, op, val]
    elsif (col = col.split('.', 2))[0] == klass.table_name
      [col[1], op, val]
    else
      nil
    end
  end.compact
  apply_where_limit_order_params klass
  ...
end
```

(I think provided_column == klass.table_name is more direct than (@provided_column.classify.constantize rescue nil), and avoids using the client-provided string for anything other than a string comparison...)

> - I am assuming we need to document the fact that we will support filters such as ['table_name.column_name', '=', 'some value'] ?

Yes, we should mention this at http://doc.arvados.org/api/methods/groups.html

#7 - 06/30/2016 08:25 PM - Radhika Chippada

> rather than writing more code to translate filters to ActiveRecord queries, why not just put the desired set of filters in @filters and let apply_where_limit_order_params translate them with the full set of features?

Used this. This is so much better than previous version with no constraint on the operator. Thanks.

> Doc update: http://doc.arvados.org/api/methods/groups.html

Done

#8 - 07/01/2016 01:33 PM - Tom Clegg

This test case seems to protect us from accidentally *fixing* a bug. It's good to test that we don't crash or anything, but wouldn't it be even better if this request was rejected with an error like "invalid attribute no_such_table.uuid"? Perhaps we should add a TODO in GroupsController?

```
[['no_such_table.uuid', '!=', 'zzzzz-tpzed-xurymjxw79nv3jz'], 200]
```

I think it would also be good to test that the filters are actually being applied, not just that the response isn't empty. Perhaps each test case could specify one UUID that should be included in the results, and one that should not?

#9 - 07/01/2016 02:33 PM - Radhika Chippada

> [['no_such_table.uuid' ...] wouldn't it be even better if this request was rejected with an error like "invalid attribute no_such_table.uuid"? Perhaps we should add a TODO in GroupsController?

Gandhi said, "Do tomorrow's work today and today's work now," and so I added this rather than add a TODO

> I think it would also be good to test that the filters are actually being applied, not just that the response isn't empty. Perhaps each test case could specify one UUID that should be included in the results, and one that should not?

Yes, I enhanced the tests and also added more failing tests with invalid attributes.

#10 - 07/01/2016 02:58 PM - Tom Clegg

I still don't think we should do the "classify.constantize" thing; it's better to go the other way around, with table_name. We should also make sure a filter like ["users.uuid","=","foo"] is rejected: "users" is a real table, but we can't filter groups.contents on users.uuid. Similarly, "time.uuid" passes the classify.constantize test, so it will also be silently ignored.

How about leaving the request_filters.map the way it was, and just validate all filters once, above the loop, using something like this:

```
klasses = [Group, Job, PipelineInstance, ...]
tables = klasses.map(&:table_name)
request_filters.each do |col, op, val|
  if col.index('.') && !tables.include?(col.split('.', 2)[0])
    raise ....
  end
end
klasses.each do |klass|
  ...
```

#11 - 07/01/2016 04:06 PM - Radhika Chippada

> I still don't think we should do the "classify.constantize" thing; it's better to go the other way around, with table_name ...How about leaving the request_filters.map the way it was, and just validate all filters once, above the loop, using something like this: tables = klasses.map(&:table_name)

This is absolutely a better alternative to my classify.constantize. Updated accordingly. Thanks.

#12 - 07/04/2016 08:39 PM - Tom Clegg

LGTM @ b9f14b6, thanks!

#13 - 07/04/2016 08:46 PM - Tom Clegg

One minor thing: shouldn't these "get contents with '#{filter}' filter" tests be functional tests instead of integration tests, since they each make just one

API call?

#14 - 07/04/2016 09:27 PM - Radhika Chippada

> One minor thing: shouldn't these "get contents with '#{filter}' filter" tests be functional tests instead of integration tests, since they each make just one API call?

You are right. Converted the test(s) into functional tests. Thanks.

#15 - 07/04/2016 09:35 PM - Radhika Chippada

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

Applied in changeset arvados|commit:05b38a78504bfa8955a70d50fa3c073206f6e780.