

Item 270

git_comments:

git_commits:

1. **summary:** [AIRFLOW-1076] Add get method for template variable accessor (#6793)
message: [AIRFLOW-1076] Add get method for template variable accessor (#6793) (cherry-picked from 0426e30a71d0f06ba80ddc49784bfac676178956)

github_issues:

github_issues_comments:

github_pulls:


1. **title:** [AIRFLOW-1076] Add get method for template variable accessor
body: Support getting variables in templates by string. This is necessary when fetching variables with characters not allowed in a class attribute name. We can then also support returning default values when a variable does not exist. Original PR went stale, <https://github.com/apache/airflow/pull/2223>. Make sure you have checked `_all_` steps below. **### Jira** - [x] My PR addresses the following [Airflow Jira] (<https://issues.apache.org/jira/browse/AIRFLOW/>) issues and references them in the PR title. For example, "[AIRFLOW-XXX] My Airflow PR" - <https://issues.apache.org/jira/browse/AIRFLOW-1076>
Description - [x] Here are some details about my PR, including screenshots of any UI changes: - See above.
Tests - [x] My PR adds the following unit tests `__OR__` does not need testing for this extremely good reason: - Added unit tests for calling ``var.value.get()`` and ``var.json.get()``, with or without default
Commits - [x] My commits all reference Jira issues in their subject lines, and I have squashed multiple commits if they address the same issue. In addition, my commits follow the guidelines from "[How to write a good git commit message]"(<http://chris.beams.io/posts/git-commit/>):
1. Subject is separated from body by a blank line
1. Subject is limited to 50 characters (not including Jira issue reference)
1. Subject does not end with a period
1. Subject uses the imperative mood ("add", not "adding")
1. Body wraps at 72 characters
1. Body explains "what" and "why", not "how"
Documentation - [x] In case of new functionality, my PR adds documentation that describes how to use it.

github_pulls_comments:

1. # [Codecov](<https://codecov.io/gh/apache/airflow/pull/6793?src=pr&el=h1>) Report > Merging [6793] (<https://codecov.io/gh/apache/airflow/pull/6793?src=pr&el=desc>) into [master] (<https://codecov.io/gh/apache/airflow/commit/702005fe35dc5b996a5c5b8d349ed36036472f00?src=pr&el=desc>) will ****decrease**** coverage by `0.06%`. > The diff coverage is `92.3%`. **[Impacted file tree graph]**(<https://codecov.io/gh/apache/airflow/pull/6793/graphs/tree.svg?width=650&token=WdLKLKHOAU&height=150&src=pr>)
(<https://codecov.io/gh/apache/airflow/pull/6793?src=pr&el=tree>) ``diff @@ Coverage Diff @@ ## master #6793 +/- ## ===== - Coverage 84.58% 84.51% -0.07% ===== Files 672 673 +1 Lines 38220 38869 +649 ===== + Hits 32329 32851 +522 - Misses 5891 6018 +127 `` | [Impacted Files](<https://codecov.io/gh/apache/airflow/pull/6793?src=pr&el=tree>) | Coverage Δ | | ---|---|---| | [airflow/models/taskinstance.py] (<https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9tb2RlbHMvdGFza2luc3RhbmNlLnB5>) | `93.8% <92.3%> (-0.21%)` | :arrow_down: | | [airflow/kubernetes/volume_mount.py](<https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9rdWJlcm5ldGVzL3ZvbHVtZV9tb3VudC5weQ==>) | `44.44% <0%> (-55.56%)` | :arrow_down: | | [airflow/kubernetes/volume.py] (<https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9rdWJlcm5ldGVzL3ZvbHVtZS5weQ==>) | `52.94% <0%> (-47.06%)` | :arrow_down: | | [airflow/kubernetes/pod_launcher.py](<https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9rdWJlcm5ldGVzL3BvZF9sYXVvY2hlcj5weQ==>) | `45.25% <0%> (-46.72%)` | :arrow_down: | | [airflow/kubernetes/refresh_config.py] (<https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9rdWJlcm5ldGVzL3JlZnJlc2hfY29uZmlnLnB5>) | `50.98% <0%> (-23.53%)` |

:arrow_down: | | [...rflow/contrib/operators/kubernetes_pod_operator.py](https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9jb250cmliL29wZXJhdG9ycy9rdWJlcm5ldGVzX3BvZF9vcGVyYXRvci5weQ==) | `78.2% <0%> (-20.52%)` | :arrow_down: | | [airflow/contrib/hooks/gcp_kms_hook.py](https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9jb250cmliL2hvbm2tL2djF9rbXNfaG9vay5weQ==) | `100% <0%> (0)` | :arrow_up: | | [airflow/executors/debug_executor.py](https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY9leGVjdXRvcnMvZGVidWdfZXhlY3V0b3IucHk=) | `92.06% <0%> (0)` | | [airflow/www/views.py](https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree#diff-YWlyZmxvdY93d3c3d3MucHk=) | `81.86% <0%> (+5.93%)` | :arrow_up: | | ... and [1 more](https://codecov.io/gh/apache/airflow/pull/6793/diff?src=pr&el=tree-more) | | -----
[Continue to review full report at Codecov](https://codecov.io/gh/apache/airflow/pull/6793?src=pr&el=continue). > **Legend** - [Click here to learn more](https://docs.codecov.io/docs/codecov-delta) > `Δ` = absolute <relative> (impact), `∅` = not affected, `?` = missing data` > Powered by [Codecov](https://codecov.io/gh/apache/airflow/pull/6793?src=pr&el=footer). Last update [702005f...af35c7a](https://codecov.io/gh/apache/airflow/pull/6793?src=pr&el=lastupdated). Read the [comment docs](https://docs.codecov.io/docs/pull-request-comments).

github_pulls_reviews:

1. Instead of needing a code change can we instead use Jinja's built in `default` filter? `{ { var.value['my.var'] | default 'fallback' } }`? <https://jinja.palletsprojects.com/en/2.10.x/templates/#default>
2. It is impossible to call this with anything other than this default value except by doing `var.__getattr__(name, default_value)` so we should just remove these changes to `__getattr__`
3. This is a static method, there is no `self` in local scope, so `self` is actually the TaskInstance. This is not what you want. ``suggestion return Variable.get(item, default_var=default_var, deserialize_json=True)``
4. Running the operator is overkill (yes I see you copied this existing pattern). Instead of putting these tests here look at the tests in tests/models/test_baseoperator.py which calls `task.render_template` more directly.
5. ``suggestion ``{{ var.value.get('variable_name', 'fallback') }}``. ``
6. ``suggestion templates by using ``{{ var.value.variable_name }}`` or ``
7. ``suggestion templates by using ``{{ var.json.variable_name }}`` or ``
8. ``suggestion ``{{ var.json.get('variable_name', {'fall': 'back'}) }}``. ``
9. Yep that looks much cleaner, plus parametrized them as well. Also moved to `tests/models/test_taskinstance.py`, but let me know if it belongs elsewhere or we want to keep it in core.
10. This resolves my original reason for adding this method (periods in variable keys), but that default logic doesn't work because it throws a KeyError first when it's missing. `Error rendering template: 'Variable my.var does not exist` is the error I get from your example.
11. 🙄 blindly copy pasted from `__getattr__`, fixed up.
12. yep had thought I was actually using `__getattr__` when I was doing `self.var` in `get()`, removed.
13. Ah okay, yeah I thought Jinja might capture a KeyError and handle it for the default filter, but I guess not. Cool, just wanted to check if this PR was needed, and it looks like it is. :+1:
14. Yup, nice one! 

jira_issues:

1. **summary:** Support getting variable by string in templates
description: Currently, one can fetch variables in templates with `{ { var.value.foo } }`. But that doesn't work if the variable key has a character you can't use as an attribute, like ":" or "-". Should provide alternative method of `{ { var.value.get('foo:bar') } }`. Can then also supply a default value if the variable is not found. This also allows you to fetch the variable specified in another jinja variable (probably not common use case).

jira_issues_comments:

1. User 'dhuang' has created a pull request for this issue: <https://github.com/apache/incubator-airflow/pull/2223>

2. stale[bot] closed pull request #2223: [AIRFLOW-1076] Add get method for template variable accessor
URL: <https://github.com/apache/incubator-airflow/pull/2223> This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/airflow/models.py b/airflow/models.py index edb3b67a40..75e2238c67 100755 --- a/airflow/models.py +++ b/airflow/models.py @@ -1699,20 +1699,31 @@ def get_template_context(self, session=None): class VariableAccessor: """ - Wrapper around Variable. This way you can get variables in templates by using - {var.variable_name}. + Wrapper around Variable. This way you can get variables in + templates by using {{ var.value.variable_name }} or + {{ var.value.get('variable_name', 'backup') }}. """ def __init__(self): self.var = None - def __getattr__(self, item): + def __getattr__(self, item, default_var=None): self.var = Variable.get(item) return self.var def __repr__(self): return str(self.var) + @staticmethod + def get(item, default_var=None): + self.var = Variable.get(item, default_var=default_var) + return self.var + class VariableJsonAccessor: + """ + Wrapper around Variable. This way you can get variables in + templates by using {{ var.json.variable_name }} or + {{ var.json.get('variable_name', 'backup') }}. + """ def __init__(self): self.var = None @@ -1723,6 +1734,12 @@ def __getattr__(self, item): def __repr__(self): return str(self.var) + @staticmethod + def get(item, default_var=None): + self.var = Variable.get(item, default_var=default_var, + deserialize_json=True) + return self.var + return { 'dag': task.dag, 'ds': ds, diff --git a/docs/code.rst b/docs/code.rst index d74d00ec56..ef5fa1ab2a 100644 --- a/docs/code.rst +++ b/docs/code.rst @@ -193,6 +193,11 @@ UI. You can access them as either plain-text or JSON. If you use JSON, you are also able to walk nested structures, such as dictionaries like: ``{{ var.json.my_dict_var.key1 }}`` +It is also possible to fetch a variable by string if +needed with ``{{ var.value.get('my_var', 'fallback') }}`` or +``{{ var.json.get('my_dict_var', {'key1': 'val1'}).key1 }}``. Defaults can be +supplied in case the variable does not exist. + Macros """ Macros are a way to expose objects to your templates and live under the diff --git a/tests/core.py b/tests/core.py index f25d0e7ff2..6322a9fd05 100644 --- a/tests/core.py +++ b/tests/core.py @@ -520,7 +520,54 @@ def verify_templated_field(context): some_templated_field='{{ var.value.a_variable }}', on_success_callback=verify_templated_field, dag=self.dag) - t.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE, ignore_ti_state=True) + t.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE, + ignore_ti_state=True) + self.assertTrue(val['success']) + + def test_template_with_variable_get(self): + """ + Test the availability of variables in templates using get() method + """ + val = { + 'success': False, + 'test_value': 'a test value' + } + Variable.set('a_variable', val['test_value']) + + def verify_templated_field(context): + self.assertEqual(context['ti'].task.some_templated_field, + val['test_value']) + val['success'] = True + + t = OperatorSubclass(+ task_id='test_complex_template', + some_templated_field='{{ var.value.get("a_variable") }}', + on_success_callback=verify_templated_field, + dag=self.dag) + t.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE, + ignore_ti_state=True) + self.assertTrue(val['success']) + + def test_template_with_variable_get_with_default(self): + """ + Test the availability of variables in templates using get() method with + a default value + """ + val = { + 'success': False, + } + + def verify_templated_field(context): + self.assertEqual(context['ti'].task.some_templated_field, 'N/A') + val['success'] = True + + t = OperatorSubclass(+ task_id='test_complex_template', + some_templated_field='{{ var.value.get("bad_variable", "N/A") }}', + on_success_callback=verify_templated_field, + dag=self.dag) + t.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE, + ignore_ti_state=True) + self.assertTrue(val['success']) + + def test_template_with_json_variable(self): @@ -543,7 +590,58 @@ def verify_templated_field(context): some_templated_field='{{ var.json.a_variable.obj.v2 }}', on_success_callback=verify_templated_field, dag=self.dag) - t.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE, ignore_ti_state=True) + t.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE, + ignore_ti_state=True) + self.assertTrue(val['success']) + + def test_template_with_json_variable_get(self): + """ + Test the availability of variables (serialized as JSON) in templates + using get() method + """ + val = { + 'success': False, + 'test_value': {'foo': 'bar', 'obj': {'v1': 'yes', 'v2': 'no'}} + } + + Variable.set('a_variable', val['test_value'], serialize_json=True) + + def verify_templated_field(context): + self.assertEqual(context['ti'].task.some_templated_field, + val['test_value']['obj']['v2']) + val['success'] = True + + t = OperatorSubclass(+ task_id='test_complex_template', + some_templated_field='{{ var.json.get("a_variable").obj.v2 }}', + on_success_callback=verify_templated_field, + dag=self.dag) + t.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE, + ignore_ti_state=True) + self.assertTrue(val['success']) + + def test_template_with_json_variable_get_with_default(self): + """ + Test the availability of variables (serialized as JSON) in templates + using get() method with a default value + """ + val = { + 'success':

```
False, + } + + def verify_templated_field(context): +
self.assertEqual(context['ti'].task.some_templated_field, + 'unknown') + val['success'] = True + + t =
OperatorSubclass( + task_id='test_complex_template', + some_templated_field=( + '{ {
var.json.get("bad_variable", { "obj": { "v2": "unknown" } })' + '.obj.v2 } }'), +
on_success_callback=verify_templated_field, + dag=self.dag) + t.run(start_date=DEFAULT_DATE,
end_date=DEFAULT_DATE, + ignore_ti_state=True) self.assertTrue(val['success']) def
test_template_with_json_variable_as_value(self): -----
```

This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

3. dhuang commented on pull request #6793: [AIRFLOW-1076] Add get method for template variable accessor URL: <https://github.com/apache/airflow/pull/6793> Support getting variables in templates by string. This is necessary when fetching variables with characters not allowed in a class attribute name. We can then also support returning default values when a variable does not exist. Original PR went stale, <https://github.com/apache/airflow/pull/2223>. Make sure you have checked `_all_` steps below. ### Jira - [x] My PR addresses the following [Airflow Jira](<https://issues.apache.org/jira/browse/AIRFLOW/>) issues and references them in the PR title. For example, "[AIRFLOW-XXX] My Airflow PR" - <https://issues.apache.org/jira/browse/AIRFLOW-1076> #### Description - [x] Here are some details about my PR, including screenshots of any UI changes: - See above. ### Tests - [x] My PR adds the following unit tests `__OR__` does not need testing for this extremely good reason: - Added unit tests for calling ``var.value.get()`` and ``var.json.get()``, with or without default ### Commits - [x] My commits all reference Jira issues in their subject lines, and I have squashed multiple commits if they address the same issue. In addition, my commits follow the guidelines from "[How to write a good git commit message]" (<http://chris.beams.io/posts/git-commit/>): 1. Subject is separated from body by a blank line 1. Subject is limited to 50 characters (not including Jira issue reference) 1. Subject does not end with a period 1. Subject uses the imperative mood ("add", not "adding") 1. Body wraps at 72 characters 1. Body explains "what" and "why", not "how" #### Documentation - [x] In case of new functionality, my PR adds documentation that describes how to use it. ----- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
4. ashb commented on pull request #6793: [AIRFLOW-1076] Add get method for template variable accessor URL: <https://github.com/apache/airflow/pull/6793> ----- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
5. Commit 0426e30a71d0f06ba80ddc49784bfac676178956 in airflow's branch refs/heads/master from Daniel Huang [<https://gitbox.apache.org/repos/asf?p=airflow.git;h=0426e30>] [AIRFLOW-1076] Add get method for template variable accessor (#6793) Support getting variables in templates by string. This is necessary when fetching variables with characters not allowed in a class attribute name. We can then also support returning default values when a variable does not exist.
6. Commit 47f384b7d3a65b8ef77328daf41d6c73f567ceaf in airflow's branch refs/heads/v1-10-test from Daniel Huang [<https://gitbox.apache.org/repos/asf?p=airflow.git;h=47f384b>] [AIRFLOW-1076] Add get method for template variable accessor (#6793) (cherry-picked from 0426e30a71d0f06ba80ddc49784bfac676178956)