UWS validation

Kristin Riebe

GAVO, AIP
UWS service at AIP

- "Daiquiri" web service for data publication (catalogues)
- SQL queries on database tables
- with UWS interface (not full TAP)
- UWS = universal worker service, for asynchronous, job-oriented web services
  - user creates job, job waits in queue until executed
  - results not returned immediately
  - UWS was recently updated to version 1.1
UWS 1.1 features

- latest version at volute-repository
- example url for job list:
  - https://gaia.aip.de/uws/query
- new job list filtering: append keywords:
  - by phase: ?PHASE=EXECUTING&PHASE=COMPLETED
  - latest jobs: ?LAST=100
  - latest with phase: ?LAST=100&PHASE=EXECUTING
  - jobs created after given date: ?AFTER=2016-01-01
- wait-blocking behaviour for jobs:
  - just wait: ?WAIT=100
  - with phase: ?WAIT=100&PHASE=QUEUED
- many combinations possible, a lot of options to test
UWS tests

- created external, stand-alone test suite, using python
- uses `behave`: python module for functional tests
  - called from the command line:
    ```
    behave <options> <feature-file>
    ```
  - define test cases with Gherkin syntax for human-readable text descriptions (features and scenarios, similar to Cucumber), e.g.:
    ```
    Scenario: Ensure user can access UWS endpoint
    When I make a GET request to base URL
    Then the response status should be "200"
    ```
- even allows looping over parameters
Scenario Outline: PHASE filter

When I make a GET request to "?PHASE=<phase>"
Then the response status should be "200"
And all UWS elements "phase" should be "<phase>"

Examples: Valid phases

<table>
<thead>
<tr>
<th>phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>PENDING</td>
</tr>
<tr>
<td>QUEUED</td>
</tr>
<tr>
<td>EXECUTING</td>
</tr>
<tr>
<td>COMPLETED</td>
</tr>
<tr>
<td>ERROR</td>
</tr>
<tr>
<td>ABORTED</td>
</tr>
<tr>
<td>ARCHIVED</td>
</tr>
<tr>
<td>HELD</td>
</tr>
<tr>
<td>SUSPENDED</td>
</tr>
<tr>
<td>UNKNOWN</td>
</tr>
</tbody>
</table>
UWS tests (continued)

- **behave** Python module
  - allows using tags for filtering tests for different uses, e.g.:
    - use tags for UWS1.1, slow jobs, etc. for being able to exclude them
  - takes care of collecting error messages

- => only needed to define test cases and implement the steps:

**uws-validator**

https://github.com/kristinriebe/uws-validator
UWS tests – user configuration

- parameters added via command line or config file
- parameters are:
  - server's url, e.g. `https://gaia.aip.de`
  - base URL for UWS service: `uws/query`
  - user credentials for authentication
  - job details for jobs of different (estimated) duration:
    - veryshort: finishes immediately
    - short: < 30 seconds
    - long: a couple of minutes
    - error: a job that will return with an error
Examples: testing uws

- Check basic access and authentication:
  - `behave -D configfile="userconfig-gaia.json" features/account.feature`

- Test job list, creating veryshort job:
  - `behave [...] --tags=basics`

- For UWS 1.0, exclude all 1.1 tests:
  - `behave [...] --tags=-uws1_1`

- Do fast tests first (exclude slow and neverending jobs):
  - `behave [...] --tags=-slow --tags=-neverending`
When I create and start a user-defined "veryshort" job Then the response status should be "200"
And the UWS element "phase" should be one of "QUEUED, EXECUTING, COMPLETED"

@slow
Scenario: Create a job with error
Given I set base URL to user-defined value
And I set BasicAuth username and password to user-defined values
When I create a user-defined "error" job
  POST request to URL: https://gaia.aip.de/uws/query
When I create a user-defined "error" job
And I send PHASE="RUN" to the phase of the same job
  POST request to URL: https://gaia.aip.de/uws/query/14627943672458386135/phase
And I send PHASE="RUN" to the phase of the same job
And I check the same job every "2" seconds until it is in a final state
  Then the UWS element "phase" should be "ERROR"
And the UWS element "startTime" should exist
And the UWS element "endTime" should exist

@slow @longjob @queue
Scenario: Create a long job and abort
Given I set base URL to user-defined value
And I set BasicAuth username and password to user-defined values
When I create a user-defined "long" job
  POST request to URL: https://gaia.aip.de/uws/query
When I create a user-defined "long" job
And I send PHASE="RUN" to the phase of the same job
  POST request to URL: https://gaia.aip.de/uws/query/1462794367216545838/phase
And I send PHASE="RUN" to the phase of the same job
And I check the same job every "2" seconds until it starts or is aborted/deleted
And I send PHASE="ABORT" to the phase of the same job
  POST request to URL: https://gaia.aip.de/uws/query/1462794367216545838/phase
And I send PHASE="ABORT" to the phase of the same job
And I wait for "2" seconds
  Then the UWS element "phase" should be "ABORTED"
And the UWS element "endTime" should exist

Clean-up: removing the created test jobs

The removed jobIds are: ['146279436853934385', '146279436074323857', '1462794368922912618', '1462794361424657893', '1462794361782985351', '1462794362198905356', '1462794362458386135', '1462794367216545838']
1 feature passed, 0 failed, 0 skipped
8 scenarios passed, 0 failed, 8 skipped
56 steps passed, 0 failed, 0 skipped, 8 undefined
 Took 68.580s
kristin@torado:~/E-Science/Daiquiri/test-uws/uws-validator$
Discussion I

- Some decisions needed to be made
- Job list checks:
  - Only use existing job list?
    - But then cannot test anything if no previous jobs
  - Use "fresh" test account with no previous jobs?
    - But no guarantee that jobs finish soon (time in queue uncertain)
- Reuse previously created jobs?
  - Would save time for some tests
  - But then tests depend on each other
  - So: rather create new jobs each time I need them and clean up afterwards
Discussion II

- **WAIT checks:**
  - useful for jobs in active phases (PENDING, QUEUED, EXECUTING)
  - but time until phase changes is uncertain (e.g. if WAIT=10 really waits 10 seconds is difficult to check for QUEUED jobs)
  - only PENDING phase change is controlled by user
  - server may return anytime sooner (allowed by standard)
Discussion III

- Divide into different use cases?
  1. "own" services, where I have full control:
     - need complete feature tests for services (can use short jobs, influence time in queue)
  2. validation of external services:
     - only check required features (job list)
     - no reliable possibility to check WAIT, unless further information is given, like:
       - "returned early because server is busy"
       - "server max. wait time exceeded"

- should print out validation report with features that could/could not be tested