Leibniz-Institut für
Astrophysik Potsdam

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

UWS tests – loop example

```
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
    | phase     |
    | PENDING   |
    | QUEUED    |
    | EXECUTING |
    | COMPLETED |
    | ERROR     |
    | ABORTED   |
    | ARCHIVED  |
    | HELD      |
    | SUSPENDED |
    | UNKNOWN   |
```

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 I

- 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`

Screenshot

```
    When I create and start a user-defined "veryshort" job                    # features/steps/steps_jobs.py:20 0.249s
    Then the response status should be "200"                                  # features/steps/steps_http.py:62 0.000s
    And the UWS element "phase" should be one of "QUEUED, EXECUTING, COMPLETED" # features/steps/steps_xml.py:57 0.000s

  @slow
  Scenario: Create a job with error                                           # features/job_basics.feature:63
    Given I set base URL to user-defined value                                # features/steps/steps_http.py:81 0.001s
    And I set BasicAuth username and password to user-defined values          # features/steps/steps_http.py:77 0.000s
    When I create a user-defined "error" job                                  # features/steps/steps_jobs.py:41
      POST request to URL:  https://gaia.aip.de/uws/query
    When I create a user-defined "error" job                                  # features/steps/steps_jobs.py:41 0.188s
    And I send PHASE="RUN" to the phase of the same job                       # features/steps/steps_jobs.py:89
      POST request to URL:  https://gaia.aip.de/uws/query/1462794362458386135/phase
    And I send PHASE="RUN" to the phase of the same job                       # features/steps/steps_jobs.py:89 0.251s
    And I check the same job every "2" seconds until it is in a final state # features/steps/steps_jobs.py:131 4.314s
    Then the UWS element "phase" should be "ERROR"                            # features/steps/steps_xml.py:64 0.000s
    And the UWS element "startTime" should exist                              # features/steps/steps_xml.py:36 0.000s
    And the UWS element "endTime" should exist                                # features/steps/steps_xml.py:36 0.000s

  @slow @longjob @queue
  Scenario: Create a long job and abort                                                # features/job_basics.feature:80
    Given I set base URL to user-defined value                                         # features/steps/steps_http.py:81 0.000s
    And I set BasicAuth username and password to user-defined values                   # features/steps/steps_http.py:77 0.000s
    When I create a user-defined "long" job                                            # features/steps/steps_jobs.py:41
      POST request to URL:  https://gaia.aip.de/uws/query
    When I create a user-defined "long" job                                            # features/steps/steps_jobs.py:41 0.183s
    And I send PHASE="RUN" to the phase of the same job                                # features/steps/steps_jobs.py:89
      POST request to URL:  https://gaia.aip.de/uws/query/1462794367216545838/phase
    And I send PHASE="RUN" to the phase of the same job                                # features/steps/steps_jobs.py:89 0.255s
    And I check the same job every "2" seconds until it starts or is aborted/deleted # features/steps/steps_jobs.py:99 4.291s
    And I send PHASE="ABORT" to the phase of the same job                              # features/steps/steps_jobs.py:89
      POST request to URL:  https://gaia.aip.de/uws/query/1462794367216545838/phase
    And I send PHASE="ABORT" to the phase of the same job                              # features/steps/steps_jobs.py:89 0.199s
    And I wait for "1" seconds                                                         # features/steps/steps_jobs.py:268 1.001s
    Then the UWS element "phase" should be "ABORTED"                                   # features/steps/steps_xml.py:64 0.000s
    And the UWS element "endTime" should exist                                         # features/steps/steps_xml.py:36 0.000s
Clean-up: removing the created test jobs

The removed jobIds are: ['1462794360553334385', '1462794360744323057', '1462794360922912618', '1462794361424657893', '14627943617
82996351', '1462794362198905356', '1462794362458386135', '1462794367216545838']
1 feature passed, 0 failed, 0 skipped
8 scenarios passed, 0 failed, 0 skipped
56 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m12.580s
kristin@dorado:~/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