1. PyVO

Stefan Becker *(sbecker@ari.uni-heidelberg.de)*
Hendrik Heinl *(heinl@ari.uni-heidelberg.de)*

Agenda

- The simple ones: SCS, SIAP, SSAP
- Registry
- SAMP
- TAP - UWS
- DALI (and SODA)

This is a talk given at the AppsWG session at the Interop in College Park in November 2018

2. What's PyVO?

PyVO is the Python API to the VO standards and protocols. It's an affiliated astropy package. Currentlly it's developed and maintained by GAVO in Heidelberg, in particular by

Stefan Becker *(sbecker@ari.uni-heidelberg.de)*

What it's not: beer (Apologies to Speakers of Slavian tongues)

So far, PyVO supports SCS, SIAP, SSAP, SAMP, TAP and DALI. SODA currently is in the pipeline. Coming up (possibly): VO redux, operating simple data access services, multi-service queries, sending results around on the desktop, parameter discovery, TAP queries, async services, UCDs, ObsTAP, Registry, VO for the solar system, remote manipulation. . .

3. SCS, SIAP, SSAP

When querying "simple" remote services (image, spectral, cone search; *not* directly TAP), PyVO has a consistent pattern:

```
# <prot> is SIA, SSA, SCS, SLA...
import pyvo

# construct a service object with a service's endpoint URL
service = pyvo.dal.<prot>Service(access_url)

#call the search method with the protocol's parameters
for result in service.search(<parameters>):
  ...work on dict-like object result...
```


4. Registry

```
for svc_rec in pyvo.registry.search(datamodel="obscore"):
  svc = pyvo.dal.TAPService(svc_rec.access_url)
  result = svc.run_sync("SELECT DISTINCT obs_collection
    FROM ivoa.obscore")
  print("\n>>>>{}\n{}\n".format(
    svc_rec.short_name,
    "\n".join(
      r["obs_collection"] for r in result))
```

5. SAMP

```
  with vohelper.SAMP_conn() as conn:
    ... (search) ...
    vohelper.send_image_to(conn, None, image.acref)
```

See PDF attachment(s): globalsiapsamp.py

See PDF attachment(s): vohelper.py

6. TAP

```
QUERIES = [
  ("twomass", "http://dc.zah.uni-heidelberg.de/tap",
    """SELECT TOP 1000000 raj2000, dej2000, jmag, hmag, kmag
    ...
  ("allwise", "http://tapvizier.u-strasbg.fr/TAPVizieR/tap",
    """SELECT raj2000, dej2000, w1mag, w2mag, w3mag, w4mag...]
    ...

with vohelper.SAMP_conn() as conn:
  topcat_id = vohelper.find_client(conn, "topcat")
  for short_name, access_url, query in QUERIES:
    service = pyvo.dal.TAPService(access_url)
    result = service.run_sync(query.format(**locals()), maxrec=90000)
    vohelper.send_table_to(conn, topcat_id, result.table, short_name)
```

See PDF attachment(s): fetch2.py

7. TAP - UWS

```
jobs = set()
for short_name, access_url, query in QUERIES:
  job = pyvo.dal.TAPService(access_url).submit_job(
    query.format(**locals()), maxrec=9000000)
  job.run()
  jobs.add((short_name, job))

 while jobs:
  time.sleep(5)
  for short_name, job in list(jobs):
    if job.phase not in ('QUEUED', 'EXECUTING'):
      jobs.remove((short_name, job))
      vohelper.send_table_to(conn, topcat_id,
          job.fetch_result().table, short_name)
      job.delete()
```

See PDF attachment(s): fetch2˙async.py

8. Magic : UCDs build SEDs

```
UCD_TO_WL = {
  "phot.mag;em.opt.u": 3.5e-7,
  "phot.mag;em.opt.b": 4.5e-7,
  "phot.mag;em.opt.v": 5.5e-7,
  "phot.mag;em.opt.r": 6.75e-7, ...

  for row in rows:
    for index, col in enumerate(row):
      ucd = row.columns[index].meta.get("ucd", "").lower())
      if ucd.startswith("phot.mag"):
        if ucd in UCD_TO_WL:
          phots.append((UCD_TO_WL[ucd], col))
```

See PDF attachment(s): fetch3˙cluster.py

9. DALI

```
for dl in result.iter_datalinks():
  for link in dl: # multiple links per dataset
    print link
```

Each link has a URL, a description, and machine-readable semantics[1]. E.g., to load previews:

```
for dl in matches.iter_datalinks():
    prev_url = dl.bysemantics("#preview").next()["access_url"]
    im = Image.open(io.BytesIO(requests.get(prev_url).content))
    ...
```

See PDF attachment(s): datalink-previews.py

10. Get it!

Get PyVO from:

- https://pypi.python.org/pypi/pyvo
- or try `apt-get install python-pyvo`
- or try `pip install pyvo`
- or try `conda install pyvo`

11. Credits and References

Thanks to Stefan Becker *(sbecker@ari.uni-heidelberg.de)* for developing and maintaining PyVO

All examples taken from the PyVO course at http://docs.g-vo.org/pyvo/html/[2]

[1] http://www.ivoa.net/rdf/datalink/core
[2] http://docs.g-vo.org/pyvo/html/