Python

Managing and Querying Stardog with Python

Taught by:

Joseph Hayes
Senior Software Engineer

Learning Objectives

- Create a Python virtualenv and install pystardog
- Manage the Stardog server with pystardog
- Query Stardog
- Demonstrate pystardog in a Jupyter notebook

Getting Started

Introduction

- Stardog's functionality is exposed via an HTTP API. pystardog wraps the Stardog HTTP API to make it easier for Python developers to interact with the server
- pystardog uses the popular Python requests library to communicate with the Stardog server
- pystardog aims to have feature parity with the Stardog HTTP API
- For more information about the underlying HTTP API visit this link: https://stardog-union.github.io/http-docs/

Install Stardog

- In order to use pystardog, you will need a local copy of Stardog to run the commands against
- You can follow the Install and Setup instructions located at https://www.stardog.com/get-started/
- Make sure the server is running with this command:
  - $ stardog-admin server status

Setup pystardog

- Python virtualenvs allow many projects to have their own isolated Python environments with specific versions of dependencies installed
- More information can be found at https://docs.python.org/3/tutorial/venv.html

Setup pystardog

- Create a virtualenv to contain pystardog dependencies
  - $ python3 -m venv ~/.virtualenvs/tutorial
- Activate the virtualenv
  - $. ~/.virtualenvs/tutorial/bin/activate
- Install pystardog and dependencies
  - $ pip install pystardog jupyterlab pandas seaborn

Overview

- As a reminder, pystardog wraps the functionality of the Stardog DBMS
- There are two main classes used to interact with the server:
  - stardog.admin.Admin
    - Administer a Stardog server
    - https://pystardog.readthedocs.io/en/latest/source/stardog.html#stardog.admin.Admin
  - stardog.connection.Connection
    - Connect to Stardog databases
    - https://pystardog.readthedocs.io/en/latest/source/stardog.html#stardog.connection.Connection

Examples

Preparing to Administer the Server

- Below is an example of how to create an Admin object
- The Admin object has methods that allow the server to be administered, for example creating databases, or configuring the server settings.

```
import stardog

connection_details = {
  'endpoint': 'http://localhost:5820',
  'username': 'admin',
  'password': 'admin'
}
admin = stardog.Admin(**connection_details)
```

Stored Queries

- Create a new stored query

```
admin.new_stored_query(
    'all_triples',
    'select * where { ?s ?p ?o . }',
    { 'database': database_name }
)
```

- List stored queries on the server

```
In [1]: [sq.name for sq in admin.stored_queries()]
Out[1]: ['all_triples']
```

- Run a stored query

```
conn = stardog.Connection(database_name)
conn.select('all_triples')
```

Virtual Graphs

- https://docs.stardog.com/virtual-graphs/

```
example_vg_options = {
        "jdbc.driver": "com.mysql.jdbc.Driver",
        "jdbc.username": "admin",
        "jdbc.password": "admin",
        "jdbc.url": "jdbc:mysql://localhost/my_db"
}

admin.new_virtual_graph('vg', content.File('path/to/mapping_file.ttl'), example_vg_options)
```

Managing Databases on the Server

- Database objects have a variety of methods such as online/offline, backup, drop, copy, repair, and set_options.
- Complete documentation is located here: https://pystardog.readthedocs.io/en/latest/source/stardog.html#stardog.admin.Database

```
db = admin.database(database_name)
```

  - Retrieve an object representing a database

Listing Databases on the Stardog Server

- Retrieve a list of objects, each representing a database

```
In [7]: admin.databases()

Out[7]: [<stardog.admin.Database at 0x104aacd30>,
         <stardog.admin.Database at 0x104a69c40>]
```

- Get the names of the databases

```
In [1]: # output will likely vary on your system
        [db.name for db in admin.databases()]

Out[1]: ['test', 'pystardog-tutorial']
```

Creating/Dropping Database

- Create a new database

```
db = admin.new_database(database_name, {'search.enabled': True})
```

- Drop an existing database

```
admin.database('pystardog-tutorial').drop()
```

- Copy an existing database

```
admin.database('pystardog-tutorial').copy('new_db_name')
```

Creating Connections

- Below is an example of how to connect to a database on a Stardog server
- The Connection object is configured with server details. It has methods that allow the database to be interacted with

```
conn = stardog.Connection(
    'pystardog-tutorial', # database name
    endpoint='http://localhost:5820',
    username='admin',
    password='admin'
)
```

Transactions

- The Connection object has methods to control transactions

```
conn.begin()
# ...changes to the database
conn.commit()
# alternately conn.rollback() to discard the changes
```

Adding Data

- In this example, data from Turtle files is added to a database

```
conn.begin()
conn.add(stardog.content.File('music_schema.ttl'))
conn.add(stardog.content.File('music.ttl.gz'))
conn.commit()
```

- Triples can also be inserted from strings

```
conn.begin()
conn.add(stardog.content.Raw(':Beatles a :Band', 'text/turtle')
conn.commit()
```

Querying

- In this example, we query the database using the connection

In [1]:
```
query = """
SELECT ?date WHERE {
  ?s a <http://stardog.com/tutorial/Album> ;
    <http://stardog.com/tutorial/date> ?date .
}
"""
csv_results = conn.select(query, content_type=stardog.content_types.CSV)
pd.read_csv(io.BytesIO(csv_results)).head()
```

Out[1]:

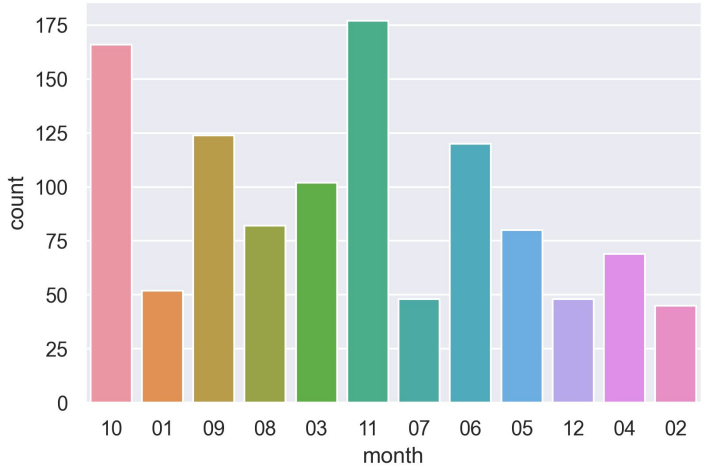
	date
0	1977-10-14
1	2006-01-01
2	2011-09-09
3	1988-08-25
4	1978-03-24

Make a Chart

- Import needed libraries and plot the query results in a chart
  - Pandas is a data analysis and manipulation module
  - Seaborn is a library for making charts

```
import io
import pandas as pd
import seaborn as sns

df = pd.read_csv(io.BytesIO(csv_results))
# make a new column in the dataset with the months
df['month'] = df['date'].str.split('-').str[1]
sns.set(style="darkgrid")
sns.countplot(x='month', data=df)
```



Parameterized Queries

- Use the bindings argument to safely parameterize a query. This example demonstrates how to pass a parameter to a Full-Text search query

```
query = """prefix fts: <tag:stardog:api:search:>

SELECT ?score ?source ?class ?result WHERE {
  service fts:textMatch {
      [] fts:query ?query ;
         fts:score ?score ;
         fts:result ?result ;
  }
  ?source a ?class ;
      ?predicate ?result .
}
order by desc(?score)"""
csv_results = conn.select(
    query,
    content_type=stardog.content_types.CSV,
    bindings={'query': '"star AND dog"'}
)
```

Query Result Content Types

- Stardog can return the results in formats besides CSV including SPARQL_XML and SPARQL_JSON. Here is an example of a query that returns JSON

```
conn.select(
    query,
    content_type=stardog.content_types.SPARQL_JSON,
    bindings={'query': '"star AND dog"'}
)
```

Demo

Learning Objectives

Learning Objectives

- Create a Python virtualenv and install pystardog
- Manage the Stardog server with pystardog
- Query Stardog
- Demonstrate pystardog in a Jupyter notebook

Thank you