SolPython

Python Solr clients

This is a part of the Solr Clients list. As with the main list, the latest source update is listed - where possibly - as a proxy for level of relevance.

SolrClient

SolrClient An actively-developed client based on Python 3 and targeting Solr 5.

Last update: November 2015

solrcloudpy

solrcloudpy is a library designed specifically for interacting with SolrCloud. It also comes with an interactive console.

Last update: April 2015

solrpy

solrpy is available at The Python Package Index so you should be able to:

```
easy_install solrpy
```

Or you can check out the source code and:

```
python setup.py install
```

Last update: April 2015

pysolr

pysolr - lightweight python wrapper for Solr.

Last update: October 2015

sunburnt

Sunburnt is a Solr library, both for inserting and querying documents. Its development has aimed particularly at making the Solr API accessible in a Pythonic style.

Last Release: version 0.6 in Jan 2012. Last code update November 2015 (has lots of forks though by other groups)

Others

- Scorched - a fork of sunburnt - June 2014
- PySolarized (Docs) - May 2014
- Solar (Russian docs) - July 2014
- Pysolr4 - June 2013
- mysolr - Last updated sometime in 2012
- Solr command line client - October 2012
- txSolr is a Twisted-based asynchronous library - October 2011
- Django-Solr ORM - August 2012

Using Solr's Python output

Solr has an optional Python response format that extends its JSON output in the following ways to allow the response to be safely eval'd by Python's interpreter:

- true and false changed to True and False
- Python unicode strings used where needed
- ASCII output (with unicode escapes) for less error-prone interoperability
- newlines escaped
- null changed to None

Here is a simple example of how one may query Solr using the Python response format:

```
from urllib2 import *
conn = urlopen('http://localhost:8983/solr/collection/select?q=iPod&wt=python')
rsp = eval( conn.read() )

print "number of matches=", rsp['response']['numFound']

#print out the name field for each returned document
for doc in rsp['response']['docs']:
  print 'name field =', doc['name']
```

With Python 2.6 you can use the literal_eval function instead of eval. This only evaluates "safe" syntax for the built-in data types and not any executable code:

```
import ast
rsp = ast.literal_eval(conn.read())
```

Using normal JSON

Using `eval` is generally considered bad form and dangerous in Python. In theory if you trust the remote server it is okay, but if something goes wrong it means someone can run arbitrary code on your server (attacking eval is very easy).

It would be better to use a Python JSON library like simplejson. It would look like:

```
from urllib2 import *
import simplejson
conn = urlopen('http://localhost:8983/solr/collection/select?q=iPod&wt=json')
rsp = simplejson.load(conn)
...
```

Safer, and as you can see, easy.