Basic Antelope Programming

Juan Reyes reyes@ucsd.edu
Nov 6, 2013
Antelope User Group Meeting
Guanacaste, Costa Rica

Abstraction

In general we want to:

1. Connects to a database
2. Perform some operation
3. Extract the information
4. Export results in some format

For Today

Build Python script that:

1. Connects to the demo database
2. Subset in time (if needed)
3. Extract *origin* information
4. Export in JSON format

Requirements

Every library used in the Python script is included in Antelope's Python distribution. The client code will include HTML and Javascript code. The popular jQuery library will be included in the example folder.

https://db.tt/lVueTLdb

General Development

* Sourced Antelope environment
* Local copy of antelope_contrib
* Write access to /opt/antelope
* git software in PATH

Example Directory

```
volcano{reyes}% ls
Makefile                index.html
event2json.xpy          jquery-2.0.3.min.js
```

event2json.xpy

```
#!/opt/antelope/python2.7.2/bin/python

"""
Output event data in JSON format

We will read the event/origin database and
output the information as a JSON text.

Juan Reyes
reyes@ucsd.edu
11/2013
"""

import os
import sys
import signal

signal.signal(signal.SIGINT, signal.SIG_DFL)

sys.path.append(os.environ['ANTELOPE'] + "/data/python")
```

event2json.xpy

```
"""
Output event data in JSON format

We will read the event/origin database and
output the information as a JSON text.

Juan Reyes
reyes@ucsd.edu
11/2013
"""
```

event2json.xpy

```
import re

try:
    from optparse import OptionParser
except Exception,e:
    print "Problem loading Python's json library. (%s)" % e
    sys.exit()

try:
    import json
except Exception,e:
    print "Problem loading Python's json library. (%s)" % e
    sys.exit()

try:
    import antelope.datascope as datascope
    import antelope.stock as stock
except Exception,e:
    print "Problem loading Python's json library. (%s)" % e
    sys.exit()
```

event2json.xpy

```
usage = "Usage: event2json [-v] [-o] [-n] [-t time] [-f outputfile]
database"

parser = OptionParser(usage=usage)
parser.add_option("-v", action="store_true",
                dest="verbose", default=False,
                help="set verbose output")
parser.add_option("-o", action="store_true",
                dest="origin", default=False,
                help="Use origin database instead")
parser.add_option("-n", action="store_true",
                dest="netmag", default=False,
                help="Join with netmag database")
parser.add_option("-t", action="store",default=0,
                dest="time", type="int",
                help="subset evids after this time")
parser.add_option("-f", action="store",default='',
                dest="file", type="string",
                help="output file to write into")
```

event2json.xpy

```
try:
    (options, args) = parser.parse_args()
except Exception,e:
    print "Problems parsing command-line arguments. (%s)" % e
    sys.exit()

if options.file:
    file = os.path.abspath(options.file)
    if os.path.exists(file):
        try:
            os.remove(file)
        except Exception,e :
            print "Cannot remove  copy of (%s) %s" % (file,e)
            sys.exit()

    # Open file object
    f = open(file,'w')
else:
    file = False
```

event2json.xpy

```
if options.time:
    time = stock.str2epoch(options.time)
else:
    time = 0
```

event2json.xpy

```
# Running on event or origin tables?
if options.origin:
    table = 'origin'
else:
    table = 'event'

if len(args) == 1:
    database = args[0]

    # Cleanup db file path if needed.
    if not re.search('.*\.%s$' % table, database):
        databaseTable = '%s.%s' % (database,table)

    # Test if db file exists.
    if not os.path.exists(databaseTable):
        print "Error looking for table (%s)" % databaseTable
        sys.exit()

else:
    print usage
    sys.exit()
```

event2json.xpy

```
# opend the databse read-only
db = datascope.dbopen(database,'r')

# look for the origin table
db = db.lookup( table=table )

# look for total records in table
nrecs = db.nrecs()

if verbose: print "Records in %s table: %s" % (table,nrecs)
```

event2json.xpy

```
if table == 'event':
    db = db.join(table='origin')
    db = db.subset('orid == preforid')
    if verbose: print "Records in table after join origin: %s" % nrecs
```

event2json.xpy

```
if netmag:
    db = db.join( 'netmag' )

    nrecs = db.nrecs()

    if verbose: print "Records after join of netmag: %s" % nrecs
```

event2json.xpy

```
if time:
    if verbose: print "Subset for time > %s" % time
    # look for events after time
    db = db.subset( "time > %s" % time )

    nrecs = db.nrecs()

    if verbose: print "Records after time subset: %s" % nrecs
```

event2json.xpy

```
for x in range(db.nrecs()):

    db[3] = x

    try:
        if netmag:
            id, lat, lon, depth, time, mag, magtype = \
                    db.getv( 'orid', 'lat', 'lon', 'depth', \
                      'time', 'magnitude','magtype' )
        else:
            id, lat, lon, depth, time = db.getv( 'orid', 'lat',
                                                 'lon', 'depth', 'time')
            mag = '-'
            magtype = '-'

    except Exception,e:
        print 'dbgetv error: %s' % e
        continue

    strTime = stock.strtime(time)
```

event2json.xpy

```
jsonOutput.append({'event':id,
        'longtime':strTime,
        'time':time,
        'lat':lat,
        'lon':lon,
        'depth':depth,
        'mag':mag,
        'magtype':magtype})
```

event2json.xpy

```
if file:
    f.write( json.dumps(jsonOutput) )
    f.close()

else:
    print json.dumps(jsonOutput)
```

Makefile

```
PF=
MAN1=
BIN=event2json

include $(ANTELOPEMAKE)
```

make/make install

```
volcano{reyes}% vi event2json.xpy
volcano{reyes}% make
produce event2json
volcano{reyes}% make install
deposit event2json /opt/antelope/5.3/bin
```

Test

```
volcano{reyes}%./event2json -nov -t "5/17/1992  21:44:00" -f event.json /
opt/antelope/data/db/demo/demo

Reading database: /opt/antelope/data/db/demo/demo.origin
Subset after start-time: 706139040.0
Records in origin table: 1351
Records after join of netmag: 5
Subset for time > 706139040.0

Records after time subset: 3
    1) lat:41.288 lon:72.409 depth:0.0 time:706139071.742 mag:2.84 type:mb
    2) lat:41.385 lon:72.358 depth:0.0 time:706139099.29 mag:3.66 type:mb
    3) lat:41.608 lon:72.293 depth:0.0 time:706139710.007 mag:2.2 type:mb

Done
```

JSON output


```
[
    {
        "depth" : 0.0,
        "longtime" : " 5/17/1992  21:44:31.742",
        "mag" : 2.84,
        "time" : 706139071.74195,
        "lat" : 41.288,
        "magtype" : "mb",
        "lon" : 72.409,
        "event" : 643
    },
    {
        "depth" : 0.0,
        "longtime" : " 5/17/1992  21:55:10.007",
        "mag" : 2.2,
        "time" : 706139710.007,
        "lat" : 41.608,
        "magtype" : "mb",
        "lon" : 72.293,
        "event" : 645
    }
]
```

index.html

```
<html>
    <head>
    ...
    </head>
    <body>
        <h2>Events from database</h2>
        <table id='eventTable'>
            <thead>
            <th>Time</th><th>Lat</th><th>Lon</th><th>Depth</
th><th>Magnitude</th>
            </thead>
            <tbody id='eventTableBody'>
            </tbody>
        </table>
        <p id='lastUpdateTime'></p>
    </body>
</html>
```

index.html HEAD

```
<html>
    <head>
    …
   <style>
      table { margin: 1em; border-collapse: collapse; }
      td, th { padding: .3em; border: 1px #ccc solid; }
      thead { background: #fc9; }
   </style>
    …
    </head>
</html>
```

index.html HEAD

```
<html>
    <head>
    …
    <script type="text/javascript" src="jquery-2.0.3.min.js"></script>
    …
    </head>
</html>
```

index.html HEAD

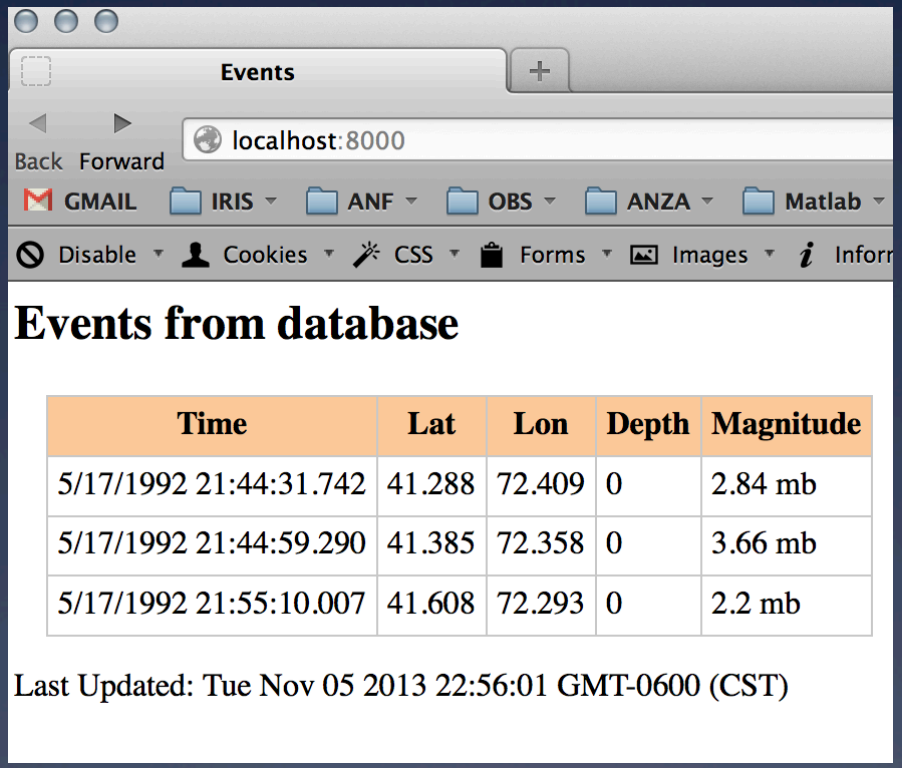
```
<script>
 function loadEvents() {
    $("#eventTableBody").empty();
    $.ajax({ url: "event.json",
       success: function(data) {
          for (var i = 0; i < data.length; i++) {
             $("#eventTableBody")
             .append($('<tr>')
             .append($('<td>').text(data[i]['longtime']))
             .append($('<td>').text(data[i]['lat']))
             .append($('<td>').text(data[i]['lon']))
             .append($('<td>').text(data[i]['depth']))
             .append($('<td>').text(
                data[i]['mag'] + ' ' + data[i]['magtype']))
             );
          }
          $("#lastUpdateTime").html('Last Updated: ' + new Date());
       }
    });
}
</script>
```

Test script

```
volcano{reyes}% ls
Makefile*          event2json*        index.html
event.json         event2json.xpy*    jquery-2.0.3.min.js

volcano{reyes}% python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
1.0.0.127.in-addr.arpa - - [05/Nov/2013 22:53:20] "GET /event.json HTTP/1.1" 200 –
...
```

Browser



Events from database

Time	Lat	Lon	Depth	Magnitude
5/17/1992 21:44:31.742	41.288	72.409	0	2.84 mb
5/17/1992 21:44:59.290	41.385	72.358	0	3.66 mb
5/17/1992 21:55:10.007	41.608	72.293	0	2.2 mb

Last Updated: Tue Nov 05 2013 22:56:01 GMT-0600 (CST)