Using RSS feeds in computational chemistry programs

by V.Ganesh, Research student, University of Pune

Abstract:
This article describes an aspect of using RSS feeds in computational chemistry programs, by giving an exemplar with implementation in GAMESS.

Introduction:

RSS or Really Simple Syndication has recently become popular and is extensively used as news aggregator. RSS is basically a small well formed XML file that contains minimal information (like headlines) and links to additional information (detailed report). It has become so ubiquitous that recent browsers like Firefox and mail clients like Thunderbird have in-built RSS readers.

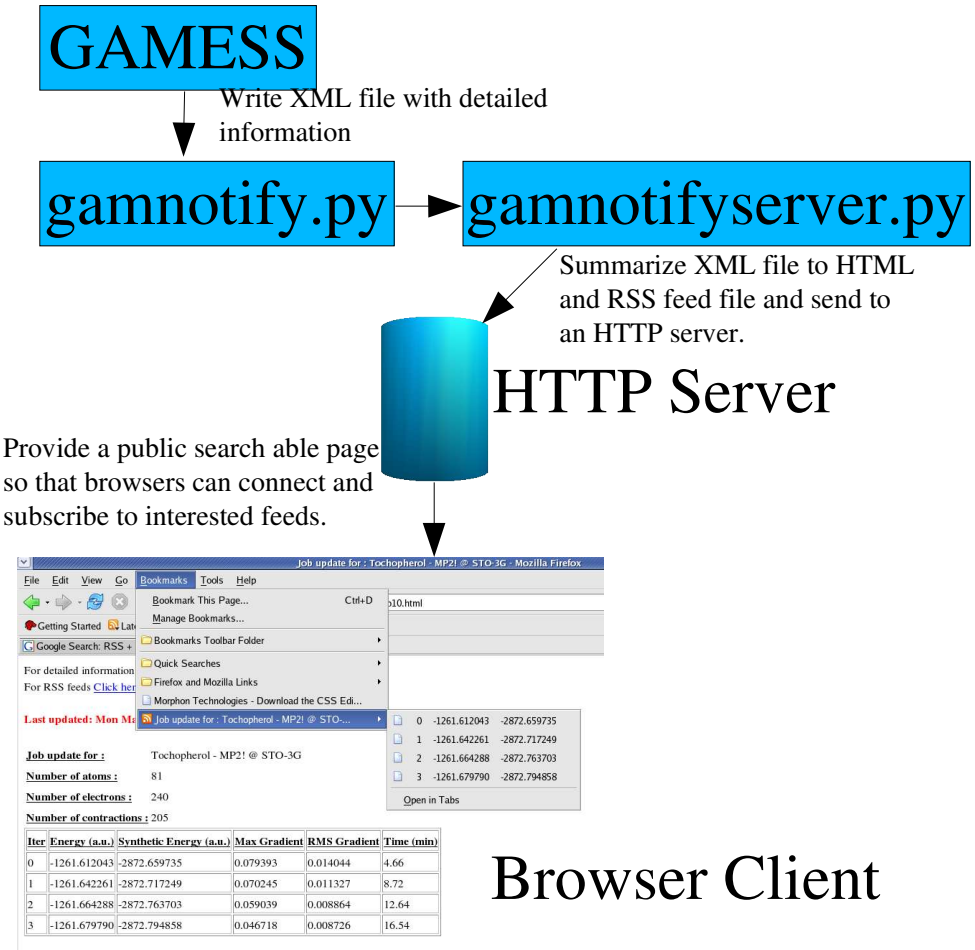
In an seemingly unconnected paradigm, computational chemistry today has become more and more oriented towards "*black-box*" approach or getting some results out of canned software. Though this form of compotenization of scientific software is good indication in the eye of a computer scientist, it takes away lot of creativity form budding computational chemists. Whether you like it or not, this segment of software is more and more going to be like todays canned OSs like Windows and OS X.

Most of the "*real*" computations performed by a computational chemist take up lot of computational resources. At a given time one might be interested in many systems and thus performing a couple of calculations on different disconnected machines. In such a case it become difficult and cumbersome to handle and keep track of the status of each computation, not only because of the number, but also because of the disconnected nature of the compute servers.

At this point a simple technology like RSS comes to our rescue. If the program that is being used by the chemist can be suitably modified so as to generate a small RSS file that can be put on an HTTP server, then your job is done! You can get updates of the job sitting from any where albeit an access to a machine having a descent browser with RSS feed (like Firefox). I must caution you that it is not always possible to get the source of the program that you use, this is especially true if you are using some commercial packages. The need to do so also brings out, why "*black-box*" approach is not always good as also the importance of

open source.

Methodology:
I follow up a very simple approach of getting RSS feeds from GAMESS program (used for *ab initio* electronic structure calculations). The basis idea is outlined in the following diagram:

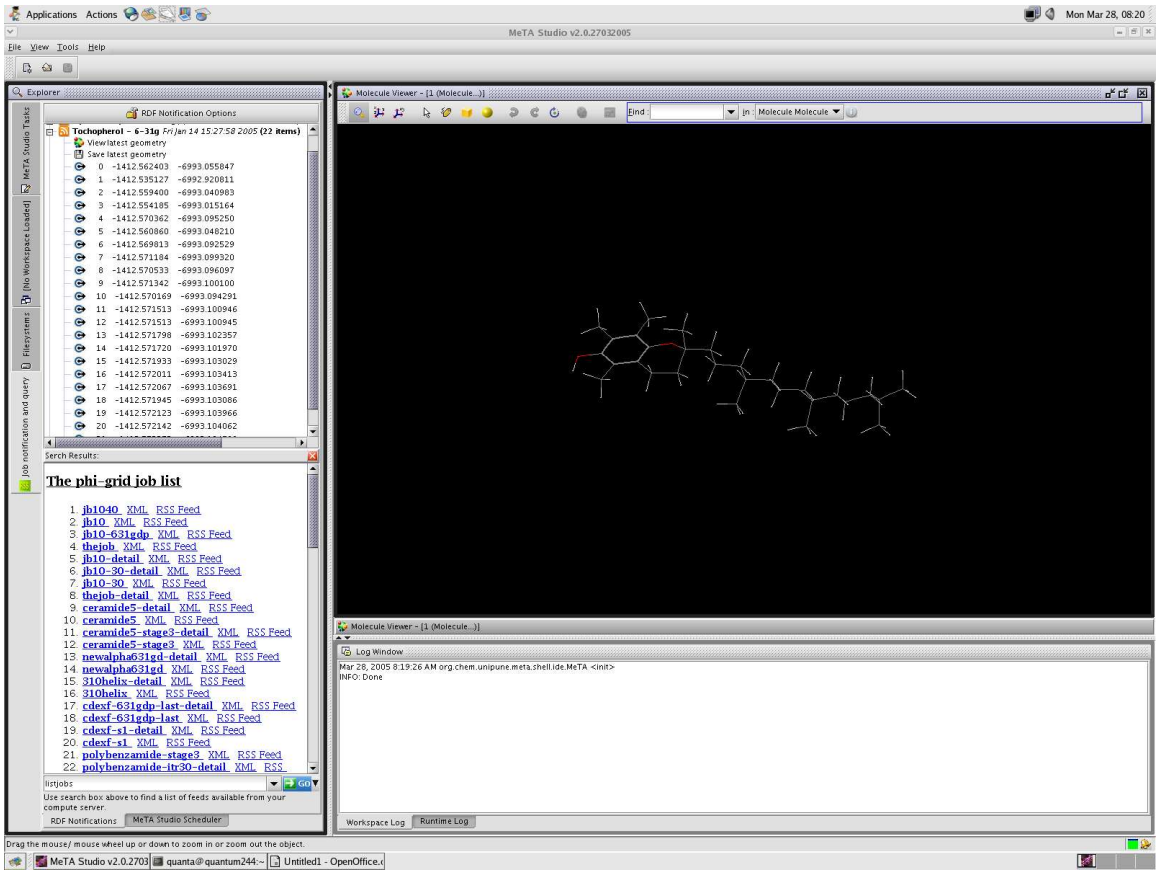


The basic programing tools used for this purpose were Python, C and FORTRAN. The GAMESS package is suitably modified to generate appropriate XML file containing current status of the job executing on a compute server. The XML file is then send to a predefined notification server (`gamnotifyserver.py`) using a python script `gamnotify.py`. A notification server, which too is written in python

(`gamnotifyserver.py`) then parses the XML and generates an HTML and RSS feed file and places it in appropriate directory so that the web server can read it. A special python script (`search.py`) is provided which gets a list of all RSS feeds available on the server and presents it to the browser. For example you can view a list of jobs currently running on our hypothetical "*phi-grid*" by pointing your browser to the following URL:

```
http://[phi-grid-server-name]/search.py?q=listjobs
```

This provides a very elegant and handy way of monitoring a job. What I want to further exemplify is that the client is not restricted to a minimal browser but it can well be a very rich desktop client as is in the case of MeTA Studio, which even allows you to view the latest geometry as shown in the following screen shot:



Conclusion:

RSS is a powerful way of notification, not only for news sites but for any process for which constant monitoring is required. I have exemplified it for GAMESS but this can be easily used for any kind of computational

chemistry package. Further, this article should give you at least an indication as to why we should not take every thing as a "*black-box*" ... because you loose the power of innovation altogether.

Code listing:
In the end, I provide code listing of the python files I used, but I don not guarantee correctness of any piece of code. In case you want to do a similar setup, you may mail me at tovganesh@yahoo.co.in.

PS: I am not an computational chemist, but a computer science student, so please avoid asking any chemistry questions to me ;)

Listing 1

```
#!/usr/bin/python

#
# search.py
#
# A specially written search script for giving a list
# of GAMESS jobs running on the chemistry quantum grid!
# Its a demo of how to write cgi based search tools for
# compute grids which can be directly integrated into
# the MeTA Studio (TM).
#
# (c) V.Ganesh
# @author V.Ganesh
# 29th Nov 2004
#

import os
import cgi
import sys
import string

from time import *

# list of supported options, at present only one!
LIST_JOBS = "listjobs"

# job list directory
JOB_LIST_DIR = "/home/tcg/public_html/gamess/"

# job URL base
JOB_URL_BASE = "http://chem.unipune.ernet.in/~tcg/gamess/"

# generate the job list
def generateJobList():
  fList = os.listdir(JOB_LIST_DIR)

  jobList = []
  newJob  = " "

  for fil in fList:
     if (string.find(fil, "frag") < 0):
        newJob = fil[0:string.rfind(fil, ".")]
```

```
        if (jobList.count(newJob) == 0):
           jobList.append(newJob)

  return jobList

# generat a valid request HTML
def generateHTML(option, jobList):
  print 'content-type: text/html \n'
  print '<html><head><title>The phi-grid</title></head><body>'
  print '<h2><u>The phi-grid job list</u></h2>'
  print '<ol>'
  for job in jobList:
       print '<li> <a href="' + JOB_URL_BASE + job + '.html" >'
       print '<b><u>' + job + '</u></b> </a>'
       print '  <a href="' + JOB_URL_BASE + job + '.xml" >XML</a>'
       print '  <a href="' + JOB_URL_BASE + job + '.rdf" >RSS Feed</a>'
       print '</li>'
  print '</ol>'

  print '</body></html>'

# there was an error! do not revel much, must be a robot!
def generateErrorHTML():
  print 'content-type: text/html \n'
  print '<html><head></head><body>'
  print '<h1> Donot Do Things Which You Donot Know! </h1>'
  print '</body></html>'

# the main search entry point
def main():
  form = cgi.FieldStorage()

  if (not form.has_key("q")):
     generateErrorHTML()

  if (LIST_JOBS == form["q"].value):
     generateHTML(LIST_JOBS, generateJobList())
  else:
     generateErrorHTML()

# the main entry!
main()
```

Listing 2

```
##
## gamnotify.py
##
## GAMESS notification client, accept the name and send in the XML file
## to the GAMESS notification server.
##
## 27th Nov 2004
##
## @author: V.Ganesh
##

import os
import sys
import string
import socket
```

```
from gamio import *
from gamconstants import *

#
# send the XML file
def sendXML(xmlFileName):
    # connect to the notification server
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((GAMESS_NOTIFY_SERVER, GAMESS_NOTIFY_SERVER_PORT))

    # open o/p stream connection
    fos = s.makefile("w")

    # send in the file name
    fos.write(xmlFileName + "\n")
    fos.flush()

    # and then the contants
    writeFile(xmlFileName, fos)

    # close the streams and connection
    fos.close()
    s.close()

# the main function
if __name__ == "__main__":
   if (not (len(sys.argv) == 2)):
       print "The i/p XML file not given: " + sys.argv[1]
       sys.exit(10)

   sendXML(sys.argv[1])
```

Listing 3

```
##
## gamnotifyserver.py
##
## GAMESS notification server
##
## 27th Nov 2004
##
## @author: V.Ganesh
##

import os
import sys
import string
import socket

import xml.dom.minidom
from xml.dom.minidom import Node

from time import *

from gamio import *
from gamconstants import *

#
# update the RSS file
def updateRSSAndHTML(xmlFileName):
    rssFileName = xmlFileName[0 : string.rfind(xmlFileName, ".")] + ".rdf"
```

```
    htmlFileName = xmlFileName[0 : string.rfind(xmlFileName, ".")] + ".html"

    # open and parse the GAMESS xml file
    gamessDoc = xml.dom.minidom.parse(GAMESS_NOTIFY_OP_DIR + xmlFileName)

    # now using this write a short update syndication file
    gamessJob = gamessDoc.getElementsByTagName("agamessjob")[0]

    rssFile  = open(GAMESS_NOTIFY_OP_DIR + rssFileName, "w")
    htmlFile = open(GAMESS_NOTIFY_OP_DIR + htmlFileName, "w")

    rssFile.write("<?xml version=\"1.0\" encoding=\"iso-8859-1\" ?>")
    rssFile.write("<rss " + \
                  "xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\" "
+ \
                  "xmlns:dc=\"http://purl.org/dc/elements/1.1/\" " + \
                  "xmlns=\"http://purl.org/rss/1.0/\" version=\"2.0\"> ")

    rssFile.write("<channel>")
    rssFile.write("<title>" + gamessJob.getAttribute("title") + "</title>")
    htmlFile.write("<html><head><title>Job update for : " +
gamessJob.getAttribute("title") + "</title></head>")
    rssFile.write("<description>Job update for : " +  gamessJob.getAttribute
("title") + "</description>")
    rssFile.write("<language>en-us</language>")
    rssFile.write("<pubDate>" + asctime() + "</pubDate>")
    rssFile.write("<lastBuildDate>" + asctime() + "</lastBuildDate>")
    rssFile.write("<docs>" + GAMESS_NOTIFY_URL_ROOT + "</docs>")
    rssFile.write("<generator>GAMXML addons</generator>")
    rssFile.write
("<managingEditor>sysadmin@chem.unipune.ernet.in</managingEditor>")
    rssFile.write("<webMaster>sysadmin@chem.unipune.ernet.in</webMaster>")

    htmlFile.write("<link rel=\"alternate\" title=\"" + \
                   "Job update for : " + gamessJob.getAttribute("title") + \
                   "\" href=\"" + GAMESS_NOTIFY_URL_ROOT + rssFileName + "\"
type=\"application/rss+xml\">")

    htmlFile.write("<body>")
    htmlFile.write("For detailed information look into the <a href=\"" + \
                   GAMESS_NOTIFY_URL_ROOT + xmlFileName + \
                   "\" > XML file </a> <br>")
    htmlFile.write("For RSS feeds <a href=\"" + \
                   GAMESS_NOTIFY_URL_ROOT + rssFileName + \
                   "\" > Click here </a>")
    htmlFile.write("<h4><font color=\"red\">Last updated: " + asctime() +
"</font></h4>")
    htmlFile.write("<table border=\"0\">")
    htmlFile.write("<tr><td><b><u>Job update for :</b></u></td><td>" +
gamessJob.getAttribute("title") + "</td></tr>")
    htmlFile.write("<tr><td><b><u>Number of atoms :</b></u></td><td>" +
gamessJob.getAttribute("noOfAtoms") + "</td></tr>")
    htmlFile.write("<tr><td><b><u>Number of electrons :</b></u></td><td>" +
gamessJob.getAttribute("noOfElectrons") + "</td></tr>")
    htmlFile.write("<tr><td><b><u>Number of contractions :</b></u></td><td>" +
gamessJob.getAttribute("noOfContractions") + "</td></tr>")
    htmlFile.write("</table>")
    htmlFile.write("<table border=\"1\">")
    htmlFile.write("<tr>")
    htmlFile.write("<td><b><u>Iter</u></b></td>")
```

```
    htmlFile.write("<td><b><u>Energy (a.u.)</u></b></td>")
    htmlFile.write("<td><b><u>Synthetic Energy (a.u.)</u></b></td>")
    htmlFile.write("<td><b><u>Max Gradient</u></b></td>")
    htmlFile.write("<td><b><u>RMS Gradient</u></b></td>")
    htmlFile.write("<td><b><u>Time (min)</u></b></td>")
    htmlFile.write("</tr>")
    for geom in gamessJob.getElementsByTagName("geom"):
        energy   = geom.getElementsByTagName("energy")[0]
        gradient = geom.getElementsByTagName("gradient")[0]
        tscf     = geom.getElementsByTagName("time")[0]

        rssFile.write("<item>")
        rssFile.write("<title>" + geom.getAttribute("number") + " " +
energy.getAttribute("value") \
                      + " " + energy.getAttribute("syntheticEnergy") +
"</title>")
        rssFile.write("<link>" + GAMESS_NOTIFY_URL_ROOT + htmlFileName +
"</link>")
        rssFile.write("<description>")

        rssFile.write("Energy: " + energy.getAttribute("value") +
energy.getAttribute("units") + \
                      " , Synthetic Energy: " + energy.getAttribute
("syntheticEnergy") + energy.getAttribute("units") + \
                      ", Maximum gradient : " + gradient.getAttribute("max") +
", RMS gradient : " + gradient.getAttribute("rms") + \
                      ", Time : " + tscf.getAttribute("scftime") +
tscf.getAttribute("units"))

        htmlFile.write("<tr>")
        htmlFile.write("<td>" + geom.getAttribute("number") + "</td>")
        htmlFile.write("<td>" + energy.getAttribute("value") + "</td>")
        htmlFile.write("<td>" + energy.getAttribute("syntheticEnergy") +
"</td>")
        htmlFile.write("<td>" + gradient.getAttribute("max") + "</td>")
        htmlFile.write("<td>" + gradient.getAttribute("rms") + "</td>")
        htmlFile.write("<td>" + tscf.getAttribute("scftime") + "</td>")

        htmlFile.write("</tr>")

        rssFile.write("</description>")
        rssFile.write("<pubDate>" + asctime() + "</pubDate>")
        rssFile.write("</item>")

    rssFile.write("</channel>")
    rssFile.write("</rss>")
    htmlFile.write("</table></body></html>")
    rssFile.close()
    htmlFile.close()

#
# the server starts here
def startServer():

    # start the service
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((GAMESS_NOTIFY_SERVER, GAMESS_NOTIFY_SERVER_PORT))

    # start an endless loop of
    while 1:
```

```
      try:
        # listen till we get some client
        s.listen(1)
        conn, addr = s.accept()

        # open i/p stream connection
        fin = conn.makefile("r")

        # read the file name
        xmlFileName = string.strip(fin.readline())

        # ensure UNIX style filenames
        xmlFileName = string.replace(xmlFileName, "\\", "/")
        xmlFileName = xmlFileName[string.rfind(xmlFileName, "/")+1 : len
(xmlFileName)]

        print "The XML will be logged to " +  xmlFileName + " at " + asctime()

        # write the file contents
        readFile(GAMESS_NOTIFY_OP_DIR + xmlFileName, fin, "w")

        # close the stream and connection
        fin.close()
        conn.close()

        # update the RSS (Really Simple Syndication) file
        updateRSSAndHTML(xmlFileName)
      except Exception, err:
        print asctime() + " : Unable to write XML : ", err.__str__()

    s.close()


# the main function
if __name__ == "__main__":
   startServer()
```