PyXNAT 2016 Advanced usage and examples

In [1]:
```
import pyxnat
import os
import time
from lxml import etree
```

In [2]:
```
from contextlib import contextmanager

@contextmanager
def timeit_context(context_name):
    startTime = time.time()
    yield
    elapsedTime = time.time() - startTime
    print('[{0}] finished in {1} ms'.format(context_name, int(elaps
edTime * 1000)))
```

In [3]:
```
xnat = pyxnat.Interface('https://central.xnat.org','nosetests','nos
e2016')
nosetests = xnat.select.project('nosetests')
subject = nosetests.subject('pyxnat2016')
```

Using xnat.inspect to take a look at the columns avaiable for an xnat:subjectData object.

The columns for any given data type come from xnat's display documents plus some auto-generated columns. The columns that start with XNAT_COL_X are auto-generated. (exactly how that happens is not fully documented.)

```
In [4]: xnat.inspect.datatypes('xnat:subjectData')

Out[4]: ['xnat:subjectData/INSERT_DATE',
         'xnat:subjectData/INSERT_USER',
         'xnat:subjectData/GENDER_TEXT',
         'xnat:subjectData/HANDEDNESS_TEXT',
         'xnat:subjectData/DOB',
         'xnat:subjectData/EDUC',
         'xnat:subjectData/SES',
         'xnat:subjectData/INVEST_CSV',
         'xnat:subjectData/PROJECTS',
         'xnat:subjectData/PROJECT',
         'xnat:subjectData/SUB_GROUP',
         'xnat:subjectData/ADD_IDS',
         'xnat:subjectData/RACE',
         'xnat:subjectData/ETHNICITY',
         'xnat:subjectData/XNAT_COL_SUBJECTDATALABEL',
         'xnat:subjectData/XNAT_COL_DEMOGRAPHICDATAAGE',
         'xnat:subjectData/XNAT_COL_MRSESSIONDATALABEL']
```

```
In [5]: #If an object does not have a display document, it will not have an
        y columns avaiable.
        xnat.inspect.datatypes('xnat:mrScanData')

Out[5]: []
```

Uploading files to deep paths

It is possible to upload files to deep paths within a resource to recreate directory structures underneath. However, It seems impossible to retrive multiple files at once. You must retrive files individually in the same manor that they were uploaded.

```
In [61]: files_resource=subject.resource('FILES')
         file1=files_resource.file('deep/folder/path/file1.txt')
         file1.put('/Users/m085077/file1.txt')
         file2=files_resource.file('deep/folder/path/file2.txt')
         file2.put('/Users/m085077/file2.txt')
```

```
In [62]: #This does NOT work to retrive multiple files even though the direc
         tory structure exists.
         file_folder=files_resource.file('deep')
         file_folder.exists()
         #file_folder.get('/Users/m085077/Downloads/')

Out[62]: False
```

```
In [67]: file2.get('/Users/m085077/Downloads/file2_downloaded.txt')

Out[67]: '/Users/m085077/Downloads/file2_downloaded.txt'
```

```
In [68]: #Files.zip is a zip file containing two files:
         #/deep/path/file1.txt
         #/deep/path/file2.txt
         zip_resource=subject.resource('ZIPS')
         zip_resource.put_zip('files.zip')
         file1=zip_resource.file('/deep/path/file1.txt')
         file1.get('/Users/m085077/Downloads/file1_downloaded.txt')

Out[68]: '/Users/m085077/Downloads/file1_downloaded.txt'
```

Using JSON to set multiple parameters at once.

```
In [52]: with timeit_context('using set'):
             subject.attrs.set('xnat:subjectData/demographics/gender','femal
         e')
             subject.attrs.set('xnat:subjectData/demographics/race', 'Minion
         2')
             subject.attrs.set('xnat:subjectData/demographics/ethnicity', 'S
         teve')

         [using set] finished in 1127 ms
```

```
In [51]: with timeit_context('using mset'):
             subject.attrs.mset({
                     'xnat:subjectData/demographics/gender': 'female',
                     'xnat:subjectData/demographics/race': 'Minion2',
                     'xnat:subjectData/demographics/ethnicity': 'Steve',
                     })

         [using mset] finished in 488 ms
```

```
In [ ]:
```

```
In [7]: ##Using json to set multiple parameters at once.

        URL = '/data/archive/projects/%s/subjects/%s' % ( 'nosetests', 'pyx
        nat2016')
        payload = {'xsiType': 'xnat:subjectData',
                   'xnat:subjectData/demographics/gender': 'female',
                   'xnat:subjectData/demographics/race': 'Minion',
                   'xnat:subjectData/demographics/ethnicity': 'Steve',
                   }
        with timeit_context('using mset'):
            r = xnat.put(URL, params=payload )
            print r

        <Response [200]>
        [using mset] finished in 111 ms
```

JSON object to access unbounded children by using the child's ID attribute

In [39]:
```
##Using json to set multiple parameters at once referencing child e
lements.

URL = '/data/archive/projects/%s/subjects/%s/experiments/%s' % ( 'n
osetests', 'pyxnat2016', 'Session_1')
payload = {'xsiType': 'xnat:MRSession',
           'xnat:MRSession/scans/scan[ID=1]/fieldStrength': '3.0',
           'xnat:MRSession/scans/scan[ID=2]/fieldStrength': '3.0',

           }
r = xnat.put(URL, params=payload )
print r
print r.content
```

```
<Response [200]>
CENTRAL_E08003
```

Example subject object XML with two custom fields

In [ ]:
```
<?xml version="1.0" encoding="UTF-8"?>
<xnat:Subject ID="CENTRAL_S04728" project="nosetests" label="pyxnat
2016" >
<xnat:fields>
    <xnat:field name="customfield1">
        <!--hidden_fields[xnat_subjectData_field_id="629",fields_fi
eld_xnat_subjectData_id="CENTRAL_S04728"]-->
        testValue1</xnat:field>
    <xnat:field name="customfield2">
        <!--hidden_fields[xnat_subjectData_field_id="630",fields_fi
eld_xnat_subjectData_id="CENTRAL_S04728"]-->
        testValue2</xnat:field>
    </xnat:fields>
</xnat:Subject>
```

Setting custom fields

Any number of custom fields can be set using Object.attrs.set(), but retriving them requires some finesse.

In [6]:

```
xnat = pyxnat.Interface('https://central.xnat.org','nosetests','nos
e2016')
nosetests = xnat.select.project('nosetests')
subject = nosetests.subject('pyxnat2016')
if subject.exists():
    subject.delete()
    time.sleep(5)
subject.create()
print "Setting and getting first attribute:"
subject.attrs.set("xnat:subjectData/fields/field[name='customfield
1']/field", 'testValue1')

print "Value:" , subject.attrs.get("xnat:subjectData/fields/field[n
ame='customfield1']/field")

print "Setting and getting second attribute:"

subject.attrs.set("xnat:subjectData/fields/field[name='customfield
2']/field", 'testValue2')
print "Value:" , subject.attrs.get("xnat:subjectData/fields/field[n
ame='customfield2']/field")

print '''
Using: subject.xpath("/xnat:Subject/xnat:fields/xnat:field[@name='c
ustomfield2']/text()[2]")[0]
xpath explanation:
    * /xnat:Subject/xnat:fields/xnat:field
        = Find me all elements of type xnat:field that are children
of xnat:Subject/xnat:fields
    * [@name='customfield2']
        = of all xnat:field elements, give me only those with an at
tribute 'name' equal to 'customfield2'
    * /text() = give me all text nodes of the found elements.
    * [2]  = we know that each xnat:field has a blank text element
before the main data element.
             So grab the 2nd element of the list.
'''
print "Value:" , subject.xpath("/xnat:Subject/xnat:fields/xnat:fiel
d[@name='customfield2']/text()[2]")[0]
```

```
Setting and getting first attribute:
Value: testValue1
Setting and getting second attribute:
Value: ['testValue2', 'testValue1']

Using: subject.xpath("/xnat:Subject/xnat:fields/xnat:field[@name
='customfield2']/text()[2]")[0]
xpath explanation:
    * /xnat:Subject/xnat:fields/xnat:field
        = Find me all elements of type xnat:field that are childre
n of xnat:Subject/xnat:fields
    * [@name='customfield2']
        = of all xnat:field elements, give me only those with an a
ttribute 'name' equal to 'customfield2'
    * /text() = give me all text nodes of the found elements.
    * [2]  = we know that each xnat:field has a blank text element
before the main data element.
             So grab the 2nd element of the list.

Value: testValue2
```

In [10]:

```
subject.attrs.set("xnat:subjectData/fields/field[name='customfield
2']/field", 'testValue2')
subject.xpath("/xnat:Subject/xnat:fields/xnat:field[@name='customfi
eld2']/text()[2]")
```

Out[10]:

```
['testValue2']
```

Xpath with attributes

Attributes are accessed with pyxnat and xpaty by prefixing the attribute with an '@' symbol

In [40]:

```
s= subject.xpath("/xnat:Subject")[0]
print s.xpath('@ID')
print s.xpath('@label')
```

```
['CENTRAL_S04768']
['pyxnat2016']
```

Examples of querying by different methods.

Each report that you write will probably require testing varous mthods to find which technique is the fastest.

In [43]:
```
print "="*80, '\n'
with timeit_context('select(/projects/nosetests/subjects/*/experiments/*'):
    print "xnat.select('/projects/nosetests/subjects/*/experiments/*')"
    results = xnat.select('/projects/nosetests/subjects/*/experiments')
    experiments=[e for e in results if e.datatype() == 'xnat:mrSessionData']
    print len(experiments)
print "="*80, '\n'

with timeit_context('select(xnat:mrSessionData).with(constraints)'):
    print "xnat.select('xnat:mrSessionData').where(contraints)"
    contraints = [('xnat:mrSessionData/project', '=', 'nosetests')]
    results = xnat.select('xnat:mrSessionData').where(contraints)
    experiments=[e for e in results]
    print len(experiments)
print "="*80, '\n'
```

```
================================================================================

xnat.select('/projects/nosetests/subjects/*/experiments/*')
6
[select(/projects/nosetests/subjects/*/experiments/*] finished in 4773 ms
================================================================================

xnat.select('xnat:mrSessionData').where(contraints)
6
[select(xnat:mrSessionData).with(constraints)] finished in 484 ms
================================================================================
```

Retriving a subject's XML data

In [6]:
```
import pprint
pprint.pprint(subject.get())
```

```
'<?xml version="1.0" encoding="UTF-8"?>\n<xnat:Subject ID="CENTRAL
_S04769" project="nosetests" label="pyxnat2016" xmlns:sapssans="ht
tp://nrg.wustl.edu/sapssans" xmlns:cnda="http://nrg.wustl.edu/cnd
a" xmlns:arc="http://nrg.wustl.edu/arc" xmlns:val="http://nrg.wust
l.edu/val" xmlns:pipe="http://nrg.wustl.edu/pipe" xmlns:genetics
="http://nrg.wustl.edu/genetics" xmlns:neurocog="http://nrg.wustl.
edu/neurocog" xmlns:fs="http://nrg.wustl.edu/fs" xmlns:sf="http://
nrg.wustl.edu/sf" xmlns:wrk="http://nrg.wustl.edu/workflow" xmlns:
scr="http://nrg.wustl.edu/scr" xmlns:xdat="http://nrg.wustl.edu/se
curity" xmlns:cat="http://nrg.wustl.edu/catalog" xmlns:nunda="htt
p://nrg.wustl.edu/nunda" xmlns:prov="http://www.nbirn.net/prov" xm
lns:xnat="http://nrg.wustl.edu/xnat" xmlns:xnat_a="http://nrg.wust
l.edu/xnat_assessments" xmlns:xsi="http://www.w3.org/2001/XMLSchem
a-instance" xsi:schemaLocation="http://nrg.wustl.edu/fs https://ce
ntral.xnat.org/schemas/fs/fs.xsd http://nrg.wustl.edu/workflow htt
ps://central.xnat.org/schemas/pipeline/workflow.xsd http://nrg.wus
tl.edu/catalog https://central.xnat.org/schemas/catalog/catalog.xs
d http://nrg.wustl.edu/pipe https://central.xnat.org/schemas/pipel
ine/repository.xsd http://nrg.wustl.edu/scr https://central.xnat.o
rg/schemas/screening/screeningAssessment.xsd http://nrg.wustl.edu/
nunda https://central.xnat.org/schemas/nunda/nunda.xsd http://nrg.
wustl.edu/arc https://central.xnat.org/schemas/project/project.xsd
http://nrg.wustl.edu/cnda https://central.xnat.org/schemas/cnda_xn
at/cnda_xnat.xsd http://nrg.wustl.edu/sapssans https://central.xna
t.org/schemas/sapssans/sapssans.xsd http://nrg.wustl.edu/sf http
s://central.xnat.org/schemas/subjforms/subjforms.xsd http://nrg.wu
stl.edu/val https://central.xnat.org/schemas/validation/protocolVa
lidation.xsd http://nrg.wustl.edu/xnat https://central.xnat.org/sc
hemas/xnat/xnat.xsd http://nrg.wustl.edu/neurocog https://central.
xnat.org/schemas/neurocog/neurocog.xsd http://nrg.wustl.edu/geneti
cs https://central.xnat.org/schemas/genetics/genetics.xsd http://n
rg.wustl.edu/xnat_assessments https://central.xnat.org/schemas/ass
essments/assessments.xsd http://www.nbirn.net/prov https://centra
l.xnat.org/schemas/birn/birnprov.xsd http://nrg.wustl.edu/security
https://central.xnat.org/schemas/security/security.xsd">\n<xnat:de
mographics>\n<!--hidden_fields[xnat_abstractDemographicData_id="42
72"]-->\n</xnat:demographics>\n<xnat:fields>\n<xnat:field name="cu
stomfield1">\n<!--hidden_fields[xnat_subjectData_field_id="674",fi
elds_field_xnat_subjectData_id="CENTRAL_S04769"]-->testValue3</xna
t:field>\n<xnat:field name="customfield2">\n<!--hidden_fields[xnat
_subjectData_field_id="675",fields_field_xnat_subjectData_id="CENT
RAL_S04769"]-->testValue2</xnat:field>\n<xnat:field name="test3">
\n<!--hidden_fields[xnat_subjectData_field_id="676",fields_field_x
nat_subjectData_id="CENTRAL_S04769"]-->42</xnat:field>\n</xnat:fie
lds>\n<xnat:experiments>\n<xnat:experiment ID="CENTRAL_E08016" pro
ject="nosetests" label="pyxnat101" xsi:type="xnat:mrSessionData">
\n<xnat:subject_ID>CENTRAL_S04769</xnat:subject_ID>\n<xnat:scans>
\n<xnat:scan ID="ScanOne" xsi:type="xnat:mrScanData">\n<!--hidden_
fields[xnat_imageScanData_id="60321"]-->\n<xnat:image_session_ID>C
ENTRAL_E08016</xnat:image_session_ID>\n<xnat:quality>good</xnat:qu
ality>\n<xnat:series_description>The description goes here</xnat:s
eries_description>\n<xnat:file label="DICOM" file_count="24" file_
size="3287640" URI="/data/xnat_central/archive/nosetests/arc001/py
xnat101/SCANS/ScanOne/DICOM/DICOM_catalog.xml" xsi:type="xnat:reso
```

```
urceCatalog">\n<!--hidden_fields[xnat_abstractResource_id="1232323
37",xnat_imageScanData_xnat_imagescandata_id="60321"]-->\n</xnat:f
ile>\n<xnat:file label="NIFIT" file_count="1" file_size="917856" U
RI="/data/xnat_central/archive/nosetests/arc001/pyxnat101/SCANS/Sc
anOne/NIFIT/NIFIT_catalog.xml" xsi:type="xnat:resourceCatalog">\n
<!--hidden_fields[xnat_abstractResource_id="123232338",xnat_imageS
canData_xnat_imagescandata_id="60321"]-->\n</xnat:file>\n<xnat:par
ameters>\n<xnat:imageType>myType</xnat:imageType>\n</xnat:paramete
rs>\n</xnat:scan>\n</xnat:scans>\n</xnat:experiment>\n</xnat:exper
iments>\n</xnat:Subject>\n'
```

Using the subject's xml to build a dom tree and querying it

```
In [45]: root = etree.fromstring(subject.get())
         root.xpath("/xnat:Subject/xnat:fields/xnat:field[@name='customfield
         2']/text()", namespaces=root.nsmap)

Out[45]: ['\n', 'testValue2']
```