Python SMART on FHIR API

Documentation: http://docs.smarthealthit.org/client-py/index.html

Class structure for patient: https://www.hl7.org/fhir/patient.html

* Replace ‘patient’ in the URL with ‘encounter’, ‘observation’, or ‘practitioner’ to see those class structures

Installation instructions:

- Download the repo https://github.com/smart-on-fhir/client-py.git
- In the root directory, run the following command
  `% python setup.py install`
- This should install fhirclient.4.0.0 to your python environment
  * If instead you decide to run `pip install fhirclient`, the version pip installs is fhiclient.3.2.0, which doesn’t support FHIR R4. Going to the github page for fhirclient states that they haven’t updated the pip installer yet.

Code Examples:

- Initial client setup:

```
from fhirclient import client
settings = {
    'app_id': 'edts',
    'api_base': 'https://r4.smarthealthit.org'
}
smart = client.FHIRClient(settings=settings)
```

- Get patient resource (the victim):

```
from fhirclient.models.patient import Patient
# Get patient by patient ID
patient = Patient.read('326b4675-0bc8-4dbd-b406-a5564c282401', smart.server)
print(patient.birthDate.isostring)
print(smart.human_name(patient.name[0]))

# Get a list of patients with the name 'Bradly Douglas'
search = Patient.where(struct={'given': 'Bradly', 'family': 'Douglas'})
patients = search.perform_resources(smart.server)
for p in patients:
    print(smart.human_name(p.name[0]))
```

- Get practitioner resource (the doctor or nurse):

```
from fhirclient.models.practitioner import Practitioner
practitioner = Practitioner.read('626133', smart.server)
print(smart.human_name(practitioner.name[0]))
print(practitioner.telecom[0].value)
```

- Set encounter resource (the place and time where the patient is seen, this updates as the patient moves through the system):

```
from fhirclient.models.encounter import Encounter
from fhirclient.models.encounter import EncounterParticipant
from fhirclient.models.coding import Coding
from fhirclient.models.codeableconcept import CodeableConcept
from fhirclient.models.fhirreference import FHIRReference
from fhirclient.models.period import Period
encounter = Encounter()
# Status of the patient, like if they have arrived or been triaged (this will change as time goes on)
encounter.status = 'triaged'
# Where the encounter takes place, like in the ER or in the field
coding = Coding({'system': 'http://terminology.hl7.org/CodeSystem/v3-ActCode', 'code': 'EMER', 'display': 'emergency'})
encounter.class_fhir = coding
# The patient being seen
encounter.subject = FHIRReference({'reference': 'Patient/326b4675-0bc8-4dbd-b406-a5564c282401'})
# The practitioner seeing the patient (this may change as time goes on, and I think should be empty when the patient first arrives)
participant = EncounterParticipant()
participant.individual = FHIRReference({'reference': 'Practitioner/03dfaa2f-a54b-4acf-bd54-80defef6ed51'})
encounter.participant = [participant]
# Why the patient came in (I think this should always be for Emergency room admission for this project)
concept = CodeableConcept()
coding = Coding({'system': 'http://snomed.info/sct', 'code': '50849002', 'display': 'Emergency room admission (procedure)'})
concept.coding = [coding]
concept.text = 'Emergency room admission (procedure)'
encounter.type = [concept]
# When the encounter started and ended (times should be in ISO format)
encounter.period = Period({'start': '2020-10-16T06:37:03-04:00'})
# Which hospital the encounter took place in
encounter.serviceProvider = FHIRReference({'reference': 'Organization/d5117822-5756-389d-9547-891a372d580f'})

# Create the encounter on the server and get the ID
status = encounter.create(smart.server)
if status is not None:
    print(status['id'])
```

- Set observation resource (the results of the practitioner’s assessment):

```
from fhirclient.models.observation import Observation
from fhirclient.models.codeableconcept import CodeableConcept
from fhirclient.models.coding import Coding
from fhirclient.models.fhirreference import FHIRReference
from fhirclient.models.fhirdate import FHIRDate
observation = Observation()
# Status of the observation (specific values found at
https://www.hl7.org/fhir/codesystem-observation-status.html)
observation.status = 'preliminary'
# What the observation is about
categoryconcept = CodeableConcept()
categoryconcept.coding = [Coding({'system':
'http://terminology.hl7.org/CodeSystem/observation-category', 'code':
'survey', 'display': 'survey'})]
observation.category = [categoryconcept]
codeconcept = CodeableConcept()
codeconcept.coding = [Coding({'system': 'http://loinc.org', 'code': '75636-
1', 'display': 'Emergency severity index'})]
codeconcept.text = 'Emergency severity index'
observation.code = codeconcept
# The patient being observed
observation.subject = FHIRReference({'reference': 'Patient/326b4675-0bc8-
4dbd-b406-a5564c282401'})
# The encounter that this observation is part of
observation.encounter = FHIRReference({'reference': 'Encounter/ce115934-fe9d-
43cf-a2e7-241d16b6d839'})
# The time that this observation was first made (times should be in ISO
format)
observation.effectiveDateTime = FHIRDate('2020-10-16T06:37:03-04:00')
# The time that this observation was reviewed and approved (times should be
in ISO format) (I think this also updates every time this observation is
updated)
observation.issued = FHIRDate('2020-10-16T06:38:21-04:00')
# The results of the observation (the field used for this changes depending
on how the observation was measured)
valueconcept = CodeableConcept()
valueconcept.coding = [Coding({'system': 'http://loinc.org', 'code':
'LA21753-1', 'display': 'Needs 2 or more resources'})]
valueconcept.text = 'Needs 2 or more resources'
observation.valueCodeableConcept = valueconcept

# Create the observation on the server and get the ID
status = observation.create(smart.server)
if status is not None:
    print(status['id'])
```

TIPS

- Use https://www.postman.com to see your posts and updates to the server in real time.

- When initializing a field in a resource (like CodeableConcept or FHIRReference), the constructor can take in a dictionary, and any elements in that dictionary with the same name as elements in that field will be automatically filled in:

```
from fhirclient.models.coding import Coding

c = Coding()
c.code = 'mycode'
c.system = 'mysystem'
```

  is equivalent to

```
c = Coding({'code': 'mycode', 'system': 'mysystem'})
```

- Be careful when entering fields to see if it only takes a single value or an array of values. For example:

  In an Encounter there needs to only be one subject and would be declared:

```
encounter.subject = mypatient
```

  In an Encounter there needs to be an array of participants; even if there is only one participant, it needs to be in an array (or you could use a list) and would be declared:

```
encounter.participant = [ mypractitioner ]
```

  You can find out if your resource field needs one value or an array by going to https://www.hl7.org/fhir/encounter.html (change 'encounter' to the resource type of your choice).

- Observations have a field called body site (which is coded using SNOMED) to specify if the injury is affecting a specific part of the body.

- Some codes for observations based on injuries from mass casualty events:

Code	**Name**	**Value**
75636-1 (loinc)	Emergency Severity Index	LL3056-0 (loinc)
9269-2 (loinc, though each part of the survey has its own loinc code)	Glasgow coma score total	386557006 (snomed)
423234004 (snomed)	Injury to respiratory system due to inhaled substance	??? (snomed)
72300-7 (loinc, useful for just assessing a wound by type and specifying body site)	Wound type	LL2215-3 (loinc)

- To code a body site, use the syntax:

```
codeconcept = CodeableConcept()
codeconcept.coding = [Coding({'system': 'http://snomed.info/sct', 'code': 'codeID', 'display':
```

```
'description of codeID'})]
codeconcept.text = 'description of codeID'
observation.bodySite = codeconcept
```

Body Site	**Code**
Face *(this might need to be split into more specific regions of the face)*	302549007
Scalp	181480002
Front of neck	362649006
Back of neck	362654002
Left shoulder	368107006
Right shoulder	368106002
Left upper arm	72098002
Right upper arm	59126009
Left lower arm	368225008
Right lower arm	368224007
Left hand	368456002
Right hand	368455003
Chest	302551006
Abdomen	302553009
Back *(this might need to be split into more specific regions of the back)*	727234005
Groin	243962005
Buttock	362677002
Left upper leg	209672000
Right upper leg	209570001
Left lower leg	213384005
Right lower leg	213289002
Left foot	239919000
Right foot	239830003

- The bodySite field is not ideal, since there do exist codes for specific injuries of specific body parts (for example, there exists a code for burn on right ear), but this is an exponentially large input set, so it is easier to code an Observation with a generic injury and a bodySite.
- The bodySite codes are all children of the SNOMED code 91722005 (typically they are children of code 362889002, which is a child of 91722005). You can find the children of this code at https://browser.ihtsdotools.org/?perspective=full&conceptId1=91722005&edition=MAIN/2020-07-31&release=&languages=en.