FHIR Parser

Release 0.1.5

Greenfrogs

Mar 18, 2020

CONTENTS

FHIR Parser is an elegant and simple FHIR library for Python, built for human beings.

The power of the FHIR Parser:

```
>> from fhir_parser import FHIR
>> fhir = FHIR()
>> patient = fhir.get_patient('8f789d0b-3145-4cf2-8504-13159edaa747')
>> patient.full_name()
'Ms. Abby752 Beatty507'
>> patient.marital_status
'Never Married'
>> patient.age()
21.514

>> observation = fhir.get_observation('4a064229-2a40-45f4-a259-f4eedcfd525a')
>> observation.type
'vital-signs'
>> observation.components[1].display
'Diastolic Blood Pressure'
>> observation.components[1].quantity()
'76.0 mm[Hg]'
```

CHAPTER

ONE

GETTING STARTED

Many core ideas & API calls are explained in the tutorial/getting-started document:

1.1 Getting Started

Welcome! This tutorial highlights FHIR Parser's core features; for further details, see the links within, or the documentation index which has links to conceptual and API doc sections.

1.1.1 Imports

The key class to import is FHIR inside fhir_parser.fhir.

```
from fhir_parser import FHIR, Patient, Observation
from fhir_parser.patient import Patient, Name, Telecom, Communications, Extension,␣
↪Identifier
from fhir_parser.observation import Observation, ObservationComponent
```

1.1.2 Patient Objects

Patients can either be retrieved via their uuid (e.g. 8f789d0b-3145-4cf2-8504-13159edaa747) or a full list of patients can be retrieved. Patients can also be requested one page at a time.

```
fhir = FHIR()
patients = fhir.get_all_patients()
specific_patient = fhir.get_patient('8f789d0b-3145-4cf2-8504-13159edaa747')
first_page_patients = fhir.get_patient_page(1)
```

The patient object contains:

- *uuid* (str)
- *name* (patient.Name object)
- *telecoms* (list of patient.Telecom objects)
- *gender* (str)
- *birth_date* (datetime.date object)
- *addresses* (list of patient.Address objects)
- *marital_status* (patient.MaritalStatus object)

- *multiple_birth* (bool)
- *communications* or languages spoken (patient.Communications object)
- *extensions* (list of patient.Extension objects)
- *identifiers* (list of patient.Identifier objects)

UUID

The unique identifier for a patient.

```
> patient.uuid
'8f789d0b-3145-4cf2-8504-13159edaa747'
```

Name

The name for a patient stored in a patient.Name object. The Name object contains the full name as a string and a list of strings for given names and prefixes. The full name can be accessed with the name.full_name property.

```
> name = patient.name
> name.full_name
'Mr. John Smith'
> name.family
'Smith'
> name.given
['John']
>name.prefix
['Mr.']
```

Telecoms

The telecommunication method for a patient stored a list of patient.Telecom objects.

```
> telecom = patient.telecoms
> first_telecom = telecom[0]
> first_telecom.system
'phone'
> first_telecom.number
'555-118-9003'
> first_telecom.use
'home'
```

Gender

The gender string of the patient.

```
> patient.gender
'female'
```

Birth Date

The birth date of the patient stored in a datetime.date object.

```
> patient.birth_date
datetime.date(1967, 2, 25)
```

Addresses

A list of addresses of a patient stored in a patient.Address object. If the latitude and longitude of the location is known then it can be accessed with the latitude and longitude properties. The full postal address can be retrieved with the full_address property.

```
> address = patient.addresses[0]
> address.full_address
'''506 Herzog Byway Apt 99
Barre, Massachusetts
01005, US'''
> address.lines
['506 Herzog Byway Apt 99']
> address.city
'Barre'
> address.state
'Massachusetts'
> address.postal_code
'01005'
> address.country
'US'
> address.latitude
42.459058557265024
> address.longitude
-72.081489014917324
```

Marital Status

The marital status of a patient stored in the patient.MarialStatus object, the str method can be used to convert the char into a meaningful string.

```
> marital = patient.marital_status
> marital.martial_status
'S'
> str(marital)
'Never Married'
```

Multiple Birth

The multiple birth argument stored as a bool, defaults to false when not available.

```
> patient.multiple_birth
False
```

Communications

The communication methods or languages spoken by the patient, stored in a single patient.Communications object.

```
> communications = patient.communications
> communications.languages
['English']
> communications.codes
['en-US']
```

Extensions

The extensions available for a patient, most commonly: us-core-race, us-core-ethnicity, patient-mothersMaidenName, us-core-birthsex, patient-birthPlace, disability-adjusted-life-years, and quality-adjusted-life-years. They can be accessed directly by retrieving the list of patient.Extension objects or more easily by using the get_extension method.

```
> patient.get_extension('us-core-ethnicity')
'Not Hispanic or Latino'
> patient.extensions
['us-core-race: White', 'us-core-ethnicity: Not Hispanic or Latino', ...]
```

Identifiers

The identifiers available for a patient, most commonly: Medical Record Number (MR), Social Security Number (SS), and Driver's License (DL). They can be accessed directly by retrieving the list of patient.Identifier objects or more easily by using the get_identifier method.

```
> patient.get_identifier('DL')
'S99995899'
> patient.identifiers
['Driver\'s License: S99995899', 'Social Security Number: 999-58-8677', ...]
```

1.1.3 Observation Objects

A single observation can be retrieved with the it's uuid or a list of observations for a single patient. Observations for a patient can also be retrieved one page at a time.

```
fhir = FHIR()
observations = fhir.get_patient_observations('8f789d0b-3145-4cf2-8504-13159edaa747')
specific_observation  = fhir.get_observation('4a064229-2a40-45f4-a259-f4eedcfd525a')
first_page_observations = fhir.get_patient_observations_page('8f789d0b-3145-4cf2-8504-
↪13159edaa747', 1)
```

The observation object contains:

- *uuid* (str)
- *type* (str)
- *status* (str)
- *patient_uuid* (str)
- *encounter_uuid* (str)
- *effective_datetime* (datetime.datetime object)
- *issued_datetime* (datetime.datetime object)
- *components* (list of observation.ObservationComponents)

UUID

The unique identifier for an observation.

```
> observation.uuid
'4a064229-2a40-45f4-a259-f4eedcfd525a'
```

Type

The type of the investigation, typically: vital-signs, survey, or laboratory.

```
> observation.type
'vital-signs'
```

Status

The status of the investigation.

```
> observation.status
'final'
```

Patient UUID

The unique identifier of the patient tied to the observation.

```
> observation.patient_uuid
'8f789d0b-3145-4cf2-8504-13159edaa747'
```

Encounter UUID

The unique identifier of the encounter tied to the observation.

```
> observation.encounter_uuid
'04090f8c-076e-4af1-9582-98d8cae66764'
```

Effective Datetime

The effective datetime of the observation returned as a datetime.datetime object.

```
> observation.effective_datetime
datetime.datetime(2011, 9, 20, 21, 27, 12, tzinfo=tzoffset(None, 3600))
```

Issued Datetime

The issued datetime of the observation returned as a datetime.datetime object.

```
> observation.issued_datetime
datetime.datetime(2011, 9, 20, 21, 27, 12, 215000, tzinfo=tzoffset(None, 3600))
```

Components

Each observation consists of multiple observation components, for example 'Diastolic Blood Pressure' and 'Systolic Blood Pressure' as part of a 'Blood Pressure' vital signs observation. Stored as a list of observation.ObservationComponent objects.

```
> component = observation.components[0]
> component.system
'http://loinc.org'
> component.code
'8462-4'
> component.display
'Diastolic Blood Pressure'
> component.value
76.0
> component.unit
mm[Hg]
> component.quantity()
'76.0 mm[Hg]'
```

CHAPTER

TWO

EXAMPLES

A collection of examples using the FHIR API:

2.1 Examples

Welcome! THis collection of examples helps to highlight FHIR Parser's core features; for further details, see the getting started guide, or the documentation index which has links to the API doc sections.

2.1.1 Average Patient Age

Calculate the average age of all patients in the FHIR database

```
from fhir_parser import FHIR

fhir = FHIR()
patients = fhir.get_all_patients()

ages = []
for patient in patients:
    ages.append(patient.age())

print(sum(ages)/len(ages))
```

2.1.2 Most Common Patient Observations

Calculate the most common patient observations, a single observation comprises of multiple observation components which is normally the actual investigation (the observation might be vital-signs which consists of two observation components for blood pressure tests).

```
from fhir_parser import FHIR

fhir = FHIR()
patients = fhir.get_all_patients()
observations = []

for patient in patients:
    observations.extend(fhir.get_patient_observations(patient.uuid))

print("Total of {} observations".format(len(observations)))
```

(continues on next page)

(continued from previous page)

```
observation_types = [observation.type for observation in observations]
most_frequent_observation_type = max(set(observation_types), key=observation_types.
↪count)
print("Most common observation type: {}".format(most_frequent_observation_type))

observation_components = []
for observation in observations:
    observation_components.extend(observation.components)

print("Total of {} observation components".format(len(observation_components)))

observation_component_types = [observation_component.display for observation_
↪component in observation_components]
most_frequent_observation_component_type = max(set(observation_types),␣
↪key=observation_types.count)
print("Most common observation component type: {}".format(most_frequent_observation_
↪component_type))
```

2.2 Graphing

The following examples use MatPlotLib to graph FHIR data.

2.2.1 Marital Status

A bar chart of the marital status of the dataset.

```
import matplotlib.pyplot as plt
from fhir_parser import FHIR

fhir = FHIR()
patients = fhir.get_all_patients()

marital_status = {}
for patient in patients:
    if str(patient.marital_status) in marital_status:
        marital_status[str(patient.marital_status)] += 1
    else:
        marital_status[str(patient.marital_status)] = 1


plt.bar(range(len(marital_status)), list(marital_status.values()), align='center')
plt.xticks(range(len(marital_status)), list(marital_status.keys()))
plt.show()
```

2.2.2 Languages Spoken

A bar chart of all languges spoken in the dataset.

```
import matplotlib.pyplot as plt
from fhir_parser import FHIR

fhir = FHIR()
patients = fhir.get_all_patients()

languages = {}
for patient in patients:
    for language in patient.communications.languages:
        languages.update({language: languages.get(language, 0) + 1})


plt.bar(range(len(languages)), list(languages.values()), align='center')
plt.xticks(range(len(languages)), list(languages.keys()), rotation='vertical')
plt.show()
```

CHAPTER

THREE

API

Know what you're looking for & just need API details? View our auto-generated API documentation:

3.1 FHIR

3.1.1 FHIR

The primary component of the FHIR parser with the FHIR class for handling all FHIR endpoint calls

class fhir_parser.fhir.**FHIR**(*endpoint: str = 'https://localhost:5001/api/'*, *verify_ssl: bool = False*, *ignore_errors: bool = True*)

Create the FHIR endpoint to retrieve patient and observation data

get_all_patients() → List[fhir_parser.patient.Patient]

Returns: A list of patients

get_observation(*id: str*) → fhir_parser.observation.Observation

Parameters **id** – Observation ID or UUID string

Returns: A single observation

get_patient(*id: str*) → fhir_parser.patient.Patient

Parameters **id** – Patient ID or UUID string

Returns: A single patient

get_patient_observations(*id: str*) → List[fhir_parser.observation.Observation]

Parameters **id** – Patient ID or UUID string

Returns: A list of observations for a patient

get_patient_observations_page(*id: str*, *page: int*) → List[fhir_parser.observation.Observation]

Parameters

- **id** – Patient ID or UUID string
- **page** – Page number int

Returns: A list of observations for a patient up to the specified page

get_patient_page(*page: int*) → List[fhir_parser.patient.Patient]

Parameters **page** – Page number int

Returns: A list of patients up to the specified page

3.2 Observation

3.2.1 Observations

The Observation and ObservationComponent objects

class fhir_parser.observation.**Observation**(*uuid: str, type: str, status: str, patient_uuid: str, encounter_uuid: str, effective_datetime: datetime.datetime, issued_datetime: datetime.datetime, components: List[fhir_parser.observation.ObservationComponent]*)

An observation object holding either one or more observation components

class fhir_parser.observation.**ObservationComponent**(*system: str, code: str, display: str, value: Union[str, float, None], unit: Optional[str]*)

An observation component object containing the details of a part of the observation

quantity() → str

Pretty print of the value and unit for an observation Returns: Value and unit, '76.0 mm[Hg]'

3.3 Patient

3.3.1 Patient

The Patient object containing information about a specific patient

class fhir_parser.patient.**Address**(*lines: List[str], city: str, state: str, postal_code: str, country: str, extensions: List[fhir_parser.patient.Extension]*)

The address consisting of multiple lines, city, state, postal code, country, and if applicable an extension containing the latitude and longitude

property full_address

The full postal address

Type Returns

property latitude

The latitude if available

Type Returns

property longitude

The longitude if available

Type Returns

class fhir_parser.patient.**Communications**(*communication: List[Tuple[str, str]]*)

The known languages and communication methods

property codes

A list of language codes

Type Returns

property languages

A list of languages

Type Returns

class fhir_parser.patient.**Extension**(*url: str, value: Union[str, float]*)
An extension consisting of a url and a value

class fhir_parser.patient.**Identifier**(*system: str*, *code: str*, *display: str*, *value: str*)
An identifier consiting of a system, code, display, and value

class fhir_parser.patient.**MaritalStatus**(*martial_status: str*)
The marital status, stored as a 1 length string, str() can be used to get the full definition

class fhir_parser.patient.**Name**(*family: str, given: List[str], prefix: List[str]*)
The name consisting of family, given (can be multiple), and prefix (can be multiple)

property full_name
The full name of a patient

Type Returns

property given
The given names of a patient joined with a space

Type Returns

property prefix
The prefix's of a patient joined with a space

Type Returns

class fhir_parser.patient.**Patient**(*uuid: str, name: fhir_parser.patient.Name, telecoms: List[fhir_parser.patient.Telecom], gender: str, birth_date: datetime.date, addresses: List[fhir_parser.patient.Address], marital_status: fhir_parser.patient.MaritalStatus, multiple_birth: bool, communications: fhir_parser.patient.Communications, extensions: List[fhir_parser.patient.Extension], identifiers: List[fhir_parser.patient.Identifier]*)
The patient object consisting of a uuid, name, telecoms, gender, birth_date, addresses, marial_status, multiple_birth, communications, extensions, and identifiers

age() → float
Returns: The age of a patient as a float

full_name() → str
Returns: The full name for a patient

get_extension(*extension: str*) → Union[str, float, None]
Gets an extension value (either str or float) based on the extension url, returns None if not available. :param extension: url of the extension

Returns: str or float of the extension or None if not found

get_identifier(*identifier: str*) → Union[str, float, None]
Gets an identifier value (str) based on the identifier code, returns None if not available. :param identifier: code of the identifier

Returns: str value of the or None if not found

class fhir_parser.patient.**Telecom**(*system: str*, *number: str*, *use: str*)
The telecommunication method consisting of the system, phone number, and use (for example home/work)

PYTHON MODULE INDEX

f

INDEX