prov Documentation

Release 2.0.0

Trung Dong Huynh

Nov 01, 2020

Contents

1 Introduction **3**
1.1 Features . . . 3

2 Installation **5**

3 Usage **7**
3.1 Simple PROV document . . . 7
3.2 PROV document with a bundle . . . 8
3.3 More examples . . . 8

4 Contributing **9**
4.1 Types of Contributions . . . 9
4.2 Get Started! . . . 10
4.3 Pull Request Guidelines . . . 11

5 prov **13**
5.1 prov package . . . 13

6 Credits **17**
6.1 Development Lead . . . 17
6.2 Contributors . . . 17

7 History **19**
7.1 2.0.0 (2020-11-01) . . . 19
7.2 1.5.3 (2018-11-20) . . . 19
7.3 1.5.2 (2018-02-06) . . . 19
7.4 1.5.1 (2017-07-18) . . . 19
7.5 1.5.0 (2016-10-19) . . . 20
7.6 1.4.0 (2015-08-13) . . . 20
7.7 1.3.2 (2015-06-17) . . . 20
7.8 1.3.1 (2015-02-27) . . . 20
7.9 1.3.0 (2015-02-03) . . . 20
7.10 1.2.0 (2014-12-19) . . . 21
7.11 1.1.0 (2014-08-21) . . . 21
7.12 1.0.1 (2014-08-18) . . . 21
7.13 1.0.0 (2014-07-15) . . . 21

8 Indices and tables 23

Python Module Index 25

Index 27

Contents:

CHAPTER 1

Introduction

A library for W3C Provenance Data Model supporting PROV-O (RDF), PROV-XML, PROV-JSON import/export

- Free software: MIT license
- Documentation: http://prov.readthedocs.io/.
- Python 3 only.

1.1 Features

- An implementation of the W3C PROV Data Model in Python.
- In-memory classes for PROV assertions, which can then be output as PROV-N
- Serialization and deserialization support: PROV-O (RDF), PROV-XML and PROV-JSON.
- Exporting PROV documents into various graphical formats (e.g. PDF, PNG, SVG).
- Convert a PROV document to a Networkx MultiDiGraph and back.

1.1.1 Uses

See a short tutorial for using this package.

This package is used extensively by ProvStore, a free online repository for provenance documents.

CHAPTER 2

Installation

At the command line:

```
$ easy_install prov
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv prov
$ pip install prov
```

CHAPTER 3

Usage

3.1 Simple PROV document

```
import prov.model as prov
import datetime

document = prov.ProvDocument()

document.set_default_namespace('http://anotherexample.org/')
document.add_namespace('ex', 'http://example.org/')

e2 = document.entity('e2', (
    (prov.PROV_TYPE, "File"),
    ('ex:path', "/shared/crime.txt"),
    ('ex:creator', "Alice"),
    ('ex:content', "There was a lot of crime in London last month"),
))

a1 = document.activity('a1', datetime.datetime.now(), None, {prov.PROV_TYPE: "edit"})
# References can be qnames or ProvRecord objects themselves
document.wasGeneratedBy(e2, a1, None, {'ex:fct': "save"})
document.wasAssociatedWith('a1', 'ag2', None, None, {prov.PROV_ROLE: "author"})
document.agent('ag2', {prov.PROV_TYPE: 'prov:Person', 'ex:name': "Bob"})

document.get_provn() # =>

# document
#   default <http://anotherexample.org/>
#   prefix ex <http://example.org/>
#
#   entity(e2, [prov:type="File", ex:creator="Alice",
#               ex:content="There was a lot of crime in London last month",
#               ex:path="/shared/crime.txt"])
#   activity(a1, 2014-07-09T16:39:38.795839, -, [prov:type="edit"])
```

(continues on next page)

(continued from previous page)

```
#   wasGeneratedBy(e2, a1, -, [ex:fct="save"])
#   wasAssociatedWith(a1, ag2, -, [prov:role="author"])
#   agent(ag2, [prov:type="prov:Person", ex:name="Bob"])
# endDocument
```

3.2 PROV document with a bundle

```
import prov.model as prov

document = prov.ProvDocument()

document.set_default_namespace('http://example.org/0/')
document.add_namespace('ex1', 'http://example.org/1/')
document.add_namespace('ex2', 'http://example.org/2/')

document.entity('e001')

bundle = document.bundle('e001')
bundle.set_default_namespace('http://example.org/2/')
bundle.entity('e001')

document.get_provn() # =>

# document
#   default <http://example.org/0/>
#   prefix ex2 <http://example.org/2/>
#   prefix ex1 <http://example.org/1/>
#
#   entity(e001)
#   bundle e001
#     default <http://example.org/2/>
#
#     entity(e001)
#   endBundle
# endDocument

document.serialize() # =>

# {"prefix": {"default": "http://example.org/0/", "ex2": "http://example.org/2/", "ex1
↪": "http://example.org/1/"}, "bundle": {"e001": {"prefix": {"default": "http://
↪example.org/2/"}, "entity": {"e001": {}}}}, "entity": {"e001": {}}}
```

3.3 More examples

See prov/tests/examples.py

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/trungdong/prov/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

We could always use more documentation, whether as part of the official prov docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/trungdong/prov/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *prov* for local development.

1. Fork the *prov* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/prov.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv prov
$ cd prov/
$ pip install -r requirements-dev.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 prov tests
$ python setup.py test
$ tox
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6+ and for PyPy3. Check https://travis-ci.org/trungdong/prov/pull_requests and make sure that the tests pass for all supported Python versions. (See pyenv for help on setting up multiple versions of Python locally for testing.)

CHAPTER 5

prov

5.1 prov package

5.1.1 Subpackages

prov.serializers package

Module contents

`prov.serializers.`**`get`**(*format_name*)

Returns the serializer class for the specified format. Raises a DoNotExist

class `prov.serializers.`**`Serializer`**(*document=None*)

Bases: `object`

Serializer for PROV documents.

`deserialize`(*stream*, ***kwargs*)

Abstract method for deserializing.

Parameters **`stream`** – Stream object to deserialize the document from.

`document = None`

PROV document to serialise.

`serialize`(*stream*, ***kwargs*)

Abstract method for serializing.

Parameters **`stream`** – Stream object to serialize the document into.

prov.serializers.provjson module

prov.serializers.provn module

class prov.serializers.provn.**ProvNSerializer**(*document=None*)

Bases: *prov.serializers.Serializer*

PROV-N serializer for ProvDocument

deserialize(*stream*, ***kwargs*)

Abstract method for deserializing.

Parameters **stream** – Stream object to deserialize the document from.

serialize(*stream*, ***kwargs*)

Serializes a `prov.model.ProvDocument` instance to a PROV-N.

Parameters **stream** – Where to save the output.

prov.serializers.provrdf module

prov.serializers.provxml module

5.1.2 Submodules

5.1.3 prov.constants module

5.1.4 prov.dot module

5.1.5 prov.graph module

5.1.6 prov.identifier module

class prov.identifier.**Identifier**(*uri*)

Bases: `object`

Base class for all identifiers and also represents xsd:anyURI.

provn_representation()

PROV-N representation of qualified name in a string.

uri

Identifier's URI.

class prov.identifier.**Namespace**(*prefix*, *uri*)

Bases: `object`

PROV Namespace.

contains(*identifier*)

Indicates whether the identifier provided is contained in this namespace.

Parameters **identifier** – Identifier to check.

Returns bool

prefix

Namespace prefix.

qname(*identifier*)

Returns the qualified name of the identifier given using the namespace prefix.

Parameters **identifier** – Identifier to resolve to a qualified name.

Returns *QualifiedName*

uri
Namespace URI.

class prov.identifier.**QualifiedName**(*namespace*, *localpart*)
Bases: *prov.identifier.Identifier*

Qualified name of an identifier in a particular namespace.

localpart
Local part of qualified name.

namespace
Namespace of qualified name.

provn_representation()
PROV-N representation of qualified name in a string.

5.1.7 prov.model module

5.1.8 Module contents

exception prov.**Error**
Bases: exceptions.Exception

Base class for all errors in this package.

prov.**read**(*source*, *format=None*)
Convenience function returning a ProvDocument instance.

It does a lazy format detection by simply using try/except for all known formats. The deserializers should fail fairly early when data of the wrong type is passed to them thus the try/except is likely cheap. One could of course also do some more advanced format auto-detection but I am not sure that is necessary.

The downside is that no proper error messages will be produced, use the format parameter to get the actual traceback.

CHAPTER 6

Credits

6.1 Development Lead

- Trung Dong Huynh (@trungdong)

6.2 Contributors

- Satrajit Ghosh (`prov.serializers.provrdf` module)
- Lion Krischer (`prov.serializers.provxml` module and Python 3 support)
- Sam Millar

CHAPTER 7

History

7.1 2.0.0 (2020-11-01)

- Removed support for EOL Python 2
- Testing against Python 3.6+ and Pypy3

7.2 1.5.3 (2018-11-20)

- Reorganised source code to /src
- Added Python 3.7 support
- Removed Python 3.3 support due to end-of-life
- plus minor improvements and bug fixes

7.3 1.5.2 (2018-02-06)

- Fixed association relation in RDF serialisation
- Fixed compatibility with networkx 2.0+

7.4 1.5.1 (2017-07-18)

- Replaced pydotplus with pydot (see #111)
- Fixed datetime and bundle error in RDF serialisation
- Tested against Python 3.6

- Improved documentation

7.5 1.5.0 (2016-10-19)

- Added: Support for PROV-O (RDF) serialization and deserialization
- Added: *direction* option for `prov.dot.prov_to_dot()`
- Added: `prov.graph.graph_to_prov()` to convert a MultiDiGraph back to a `ProvDocument`
- Testing with Python 3.5
- Various minor bug fixes and improvements

7.6 1.4.0 (2015-08-13)

- Changed the type of qualified names to prov:QUALIFIED_NAME (fixed #68)
- Removed XSDQName class and stopped supporting parsing xsd:QName as qualified names
- Replaced pydot dependency with pydotplus
- Removed support for Python 2.6
- Various minor bug fixes and improvements

7.7 1.3.2 (2015-06-17)

- Added: prov-compare script to check equivalence of two PROV files (currently supporting JSON and XML)
- Fixed: deserialising Python 3's bytes objects (issue #67)

7.8 1.3.1 (2015-02-27)

- Fixed unicode issue with deserialising text contents
- Set the correct version requirement for six
- Fixed format selection in prov-convert script

7.9 1.3.0 (2015-02-03)

- Python 3.3 and 3.4 supported
- Updated prov-convert script to support XML output
- Added missing test JSON and XML files in distributions

7.10 1.2.0 (2014-12-19)

- Added: `prov.graph.prov_to_graph()` to convert a `ProvDocument` to a MultiDiGraph
- Added: PROV-N serializer
- Fixed: None values for empty formal attributes in PROV-N output (issue #60)
- Fixed: PROV-N representation for xsd:dateTime (issue #58)
- Fixed: Unintended merging of Identifier and QualifiedName values
- Fixed: Cloning the records when creating a new document from them
- Fixed: incorrect SoftwareAgent records in XML serialization

7.11 1.1.0 (2014-08-21)

- Added: Support for PROV-XML serialization and deserialization
- A `ProvRecord` instance can now be used as the value of an attributes
- Added: convenient assertions methods for `ProvEntity`, `ProvActivity`, and `ProvAgent`
- Added: `prov.model.ProvDocument.update()` and `prov.model.ProvBundle.update()`
- Fixed: Handling default namespaces of bundles when flattened

7.12 1.0.1 (2014-08-18)

- Added: Default namespace inheritance for bundles
- Fixed: `prov.model.NamespaceManager.valid_qualified_name()` did not support `XSDQName`
- Added: Convenience *`prov.read()`* method with a lazy format detection
- Added: Convenience `plot()` method on the `ProvBundle` class (requiring matplotlib).
- Changed: The previous `add_record()` method renamed to `new_record()`
- Added: `add_record()` function which takes one argument, a `ProvRecord`, has been added
- Fixed: Document flattening (see `flattened()`)
- Added: `__hash__()` function added to `ProvRecord` (**at risk**: to be removed as `ProvRecord` is expected to be mutable)
- Added: `extra_attributes` added to mirror existing `formal_attributes`

7.13 1.0.0 (2014-07-15)

- The underlying data model has been rewritten and is **incompatible** with pre-1.0 versions.
- References to PROV elements (i.e. entities, activities, agents) in relation records are now QualifiedName instances.
- A document or bundle can have multiple records with the same identifier.
- PROV-JSON serializer and deserializer are now separated from the data model.

- Many tests added, including round-trip PROV-JSON encoding/decoding.
- For changes pre-1.0, see CHANGES.txt.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

prov, 15
prov.constants, 14
prov.identifier, 14
prov.serializers, 13
prov.serializers.provn, 14

Index

C

`contains()` (*prov.identifier.Namespace method*), 14

D

`deserialize()` (*prov.serializers.provn.ProvNSerializer method*), 14
`deserialize()` (*prov.serializers.Serializer method*), 13
`document` (*prov.serializers.Serializer attribute*), 13

E

`Error`, 15

G

`get()` (*in module prov.serializers*), 13

I

`Identifier` (*class in prov.identifier*), 14

L

`localpart` (*prov.identifier.QualifiedName attribute*), 15

N

`Namespace` (*class in prov.identifier*), 14
`namespace` (*prov.identifier.QualifiedName attribute*), 15

P

`prefix` (*prov.identifier.Namespace attribute*), 14
`prov` (*module*), 15
`prov.constants` (*module*), 14
`prov.identifier` (*module*), 14
`prov.serializers` (*module*), 13
`prov.serializers.provn` (*module*), 14
`provn_representation()` (*prov.identifier.Identifier method*), 14
`provn_representation()` (*prov.identifier.QualifiedName method*), 15
`ProvNSerializer` (*class in prov.serializers.provn*), 14

Q

`qname()` (*prov.identifier.Namespace method*), 14
`QualifiedName` (*class in prov.identifier*), 15

R

`read()` (*in module prov*), 15

S

`serialize()` (*prov.serializers.provn.ProvNSerializer method*), 14
`serialize()` (*prov.serializers.Serializer method*), 13
`Serializer` (*class in prov.serializers*), 13

U

`uri` (*prov.identifier.Identifier attribute*), 14
`uri` (*prov.identifier.Namespace attribute*), 15