Zenodio Documentation

Release 0.1.0.dev

AURA/LSST

December 29, 2015

Contents

CHAPTER 1

Zenodo I/O

Zenodio is a simple Python interface getting data into and out of Zenodo, the digital archive developed by CERN. Zenodo is an awesome tool for modern scientists to archive the products of research, including datasets, codes, and documents. Zenodio adds a layer of mechanization to Zenodo, allowing you to grab metadata about records in a Zenodo collection, or upload new artifacts to Zenodo with a smart Python API.

We're still designing the upload API, but metadata harvesting is ready to go.

Zenodio is built by SQuaRE for the Large Synoptic Survey Telescope. The code's on GitHub.

CHAPTER 2

Install Zenodio

Zenodio is built for Python 3.4+. You can install the latest release via:

```
pip install zenodio
```

Or you can get the latest version from GitHub:

```
pip install git+git://github.com/lsst-sqre/zenodio.git
```

Developers will want to read the Developer Guide.

CHAPTER 3

User Guide

3.1 Harvesting Zenodo Metadata with OAI-PMH (Datacite 3)

Zenodio's `zenodio.harvest` module provides a Pythonic interface to record metadata in a Zenodo community collection. Zenodio uses the standard OAI-PMH harvesting protocol (and specifically retrieves Datacite3-flavored XML; it's the best and latest metadata standard used by Zenodo).

3.1.1 Quick Start

Download metadata for a Zenodo collection by giving its identifier. For example, we'll get

```
import zenodio.harvest
collection = harvest_collection('lsst-dm')
```

`collection` is a *zenodio.harvest.Datacite3Collection* instance for the Zenodo community's record collection. Use its `records()` method to generate *Datacite3Record* instances for each record stored in the Zenodo community:

```
for record in collection.records():
   print(record.title)
```

Or you can get a list of all records:

```
records = [r for r in collection.records()]
```

With these *Datacite3Record* instances you can access information about individual artifacts on Zenodo through simple class attributes. For example:

```
record = records[0]
print(record.title)
print(record.issue_date)
print(record.doi)
print(record.abstract_html)
```

For information about authors, Zenodio provides an *Author* class. For example:

```
authors = record.authors
print(','.join([a.last_name for a in authors]))
```

3.1.2 API Reference

Convenience Functions

zenodio.harvest.**harvest_collection**(*community_name*)
Harvest a Zenodo community's record metadata.

Examples

You can harvest record metadata for a Zenodo community by its identifier name. For example, the identifier for LSST Data Management's Zenodo collection is `'lsst-dm'`:

```
>>> import zenodio.harvest import harvest_collection
>>> collection = harvest_collection('lsst-dm')
```

`collection` is a *Datacite3Collection* instance. Use its *records()* method to generate *Datacite3Record* objects for individual records in the Zenodo collection.

Parameters **community_name** (*str*) – Name of the community.

Returns **collection** – The *Datacite3Collection* instance with record metadata downloaded from Zenodo.

Return type *zenodio.harvest.Datacite3Collection*

Metadata Classes

class zenodio.harvest.**Datacite3Collection**(*xml_records*)
Zenodo metadata for a Community collection derived from Datacite v3 metadata.

Use the *from_collection_xml()* classmethod to build a *Datacite3Collection* from XML obtained from the Zenodo OAI-PMH API. Most likely, users should use *harvest_collection()* to build a *Datacite3Collection* for a Community.

classmethod **from_collection_xml**(*xml_content*)
Build a *Datacite3Collection* from Datecite3-formatted XML.

Users should use *zenodio.harvest.harvest_collection()* to build a *Datacite3Collection* for a Community.

Parameters **xml_content** (*str*) – Datacite3-formatted XML content.

Returns **collection** – The collection parsed from Zenodo OAI-PMH XML content.

Return type Datacite3Collection

records()
Yield records from the collection.

Yields **record** (Datacite3Record) – The Datacite3Record for an individual resource in the Zenodo collection.

class zenodio.harvest.**Datacite3Record**(*xml_dict*)
Zenodo metadata for a single record.

Use *Datacite3Record*s to access metadata about a record though a convient object properties.

Parameters **xml_dict** (collections.OrderedDict) – A *dict*-like object mapping XML content for a single record (i.e., the contents of the `record` tag in OAI-PMH XML). This dict is typically generated from `xmltodict`.

abstract_html
Abstract text, marked up with HTML (*str*).

authors
List of *Author*s (*zenodio.harvest.Author*).

Authors correspond to *creators* in the Datacite schema.

doi
Digital object identifier *str*.

issue_date
Date when the DOI was issued (`datetime.datetime.Datetime`).

title
Title of resource (*str*).

If there are multiple titles, the first title is returned.

class `zenodio.harvest.`**Author**(*last_first*, *orcid=None*, *affiliation=None*)
Metadata about an author.

Author instances are typically built by `Datacite3Record.authors()`.

Parameters

- **last_first** (*str*) – Author's name, formatted as *'Last, First'*.
- **orcid** (*str, optional*) – Author's ORCiD.
- **affiliation** (*str, optional*) – Author's affiliation.

last_first
str – Author's name, formatted as *'Last, First'*.

orcid
str – Author's ORCiD.

affiliation
str – Author's affiliation.

first_name
Author's first name (*str*).

classmethod from_xmldict(*xml_dict*)
Create an *Author* from a datacite3 metadata converted by *xmltodict*.

Parameters **xml_dict** (`collections.OrderedDict`) – A *dict*-like object mapping XML content for a single record (i.e., the contents of the `record` tag in OAI-PMH XML). This dict is typically generated from `xmltodict`.

last_name
Author's last name (*str*).

3.2 Developer Guide

Zenodio is built for Python 3.4+.

3.2.1 Development Environment

Fork the Zenodio repository, and clone:

```
git clone https://github.com/<username>/zenodio.git
cd zenodio
git remote add upstream https://github.com/lsst-sqre/zenodio.git
```

Setup a virtual environment, and install a development version of the code:

```
pip install -r requirements.txt
python setup.py develop
```

3.2.2 Style Guide

Our code style is unadulterated PEP8 (*not* the LSST DM Python code style). Use a Flake8 to make sure your code is up to snuff.

3.2.3 Testing

For testing we use pytest. Don't use `unittest`.

Run tests via:

```
py.test
```

You can find tests in the `tests/` directory. If you need to include a sample dataset, put that data in the `data/` directory. Use setuptools's `pkg_resources` to read that data.

CHAPTER 4

License

License: MIT.

A

D

F

H

I

L

O

R

T