Documentation

Release 0.3

Mathias Loesch

April 17, 2013

CONTENTS

Sickle is lightweight OAI-PMH client library written in Python. It has been designed for conveniently retrieving data from OAI interfaces the Pythonic way:

```
>>> sickle = Sickle('http://elis.da.ulcc.ac.uk/cgi/oai2')
>>> records = sickle.ListRecords(metadataPrefix='oai_dc')
```

Most importantly, Sickle lets you iterate through OAI records without having to deal with things like result batches or `resumptionTokens` yourself:

```
>>> records.next()
<Record oai:eprints.rclis.org:4088>
```

Sickle maps the OAI records to Python objects:

```
>>> record = records.next()
>>> record.header
<Header oai:eprints.rclis.org:4088>
>>> record.header.identifier
'oai:eprints.rclis.org:4088'
```

The metadata payload is stored as a dictionary:

```
>>> record.metadata
{'creator': ['Melloni, Marco'],
 'date': ['2000'],
 'description': [u'A web site for...
```

CHAPTER

ONE

IMPORTANT LINKS

- Sickle @ PyPI
- Sickle @ GitHub

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Installation

Sickle requires requests and lxml.

Installation using pip:

```
pip install sickle
```

Installation using easy_install:

```
easy_install sickle
```

2.2 Tutorial

This section gives a brief overview on how to use Sickle for querying OAI interfaces.

2.2.1 OAI-PMH Primer

This section gives a basic overview of the Open Archives Protocol for Metadata Harvesting (OAI-PMH). For more detailed information, please refer to the protocol specification.

Glossary of Important OAI-PMH Concepts

Repository A *repository* is a server-side application that exposes metadata via OAI-PMH.

Harvester OAI-PMH client applications like Sickle are called *harvesters*.

record A *record* is the XML-encoded container for the metadata of a single publication item. It consists of a *header* and a *metadata* section.

header The record *header* contains a unique identifier and a datestamp.

metadata The record *metadata* contains the publication metadata in a defined metadata format.

set A structure for grouping records for selective harvesting.

harvesting The process of requesting records from the repository by the harvester.

OAI Verbs

OAI-PMH features six main API methods (so-called "OAI verbs") that can be issued by harvesters. Some verbs can be combined with further arguments:

`Identify` Returns information about the repository. Arguments: None.

GetRecord Returns a single record. Arguments:

- `identifier` (the unique identifier of the record, *required*)
- `metadataPrefix` (the prefix identifying the metadata format, *required*)

ListRecords Returns the records in the repository in batches (possibly filtered by a timestamp or a `set`). Arguments:

- `metadataPrefix` (the prefix identifying the metadata format, *required*)
- `from` (the earliest timestamp of the records, *optional*)
- `until` (the latest timestamp of the records, *optional*)
- `set` (a set for selective harvesting, *optional*)
- `resumptionToken` (used for getting the next result batch if the number of records returned by the previous request exceeds the repository's maximum batch size, *exclusive*)

ListIdentifiers *Like* `ListRecords` *but returns only the record headers.*

ListSets Returns the list of sets supported by this repository. Arguments: None

ListMetadataFormats Returns the list of metadata formats supported by this repository. Arguments: None

Metadata Formats

OAI interfaces may expose metadata records in multiple metadata formats. These formats are identified by so-called "metadata prefixes". For instance, the prefix `oai_dc` refers to the OAI-DC format, which by definition has to be exposed by every valid OAI interface. OAI-DC is based on the 15 metadata elements specified in the Dublin Core Metadata Element Set.

Note: Sickle only supports the OAI-DC format out of the box. See section XXX for how to extend Sickle for retrieving metadata in other formats.

2.2.2 Initialize an OAI Interface

To make a connection to an OAI interface, you need to import the Sickle object:

```
>>> from sickle import Sickle
```

Next, you can initialize the connection by passing it the basic URL. In our example, we use the OAI interface of the ELIS repository:

```
>>> sickle = Sickle('http://elis.da.ulcc.ac.uk/cgi/oai2')
```

2.2.3 Issuing Requests

Now you are set to issue some requests. Sickle provides methods for each of the six OAI verbs (ListRecords, GetRecord, Idenitfy, ListSets, ListMetadataFormats, ListIdentifiers). Start with a ListRecords request:

```
>>> records = sickle.ListRecords(metadataPrefix='oai_dc')
```

Note that all keyword arguments you provide to this function are passed to the OAI interface as HTTP parameters. Therefore our example request results in `verb=ListRecords&metadataPrefix=oai_dc`. Consequently, we can add additional parameters, like `set` for example:

```
>>> records = sickle.ListRecords(metadataPrefix='oai_dc', set='driver')
```

2.2.4 Using the `from` Parameter

If you need to perform selective harvesting by date using the `from` parameter, you will run into problems though, since `from` is a reserved word in Python:

```
>>> records = sickle.ListRecords(metadataPrefix='oai_dc', from="2012-12-12")
  File "<stdin>", line 1
    records = sickle.ListRecords(metadataPrefix='oai_dc', from="2012-12-12")
                                                            ^
SyntaxError: invalid syntax
```

Fortunately, you can circumvent this problem by using a dictionary together with the `**` operator:

```
>>> records = sickle.ListRecords(
...     **{'metadataPrefix': 'oai_dc',
...       'from': '2012-12-12'
...       }
... )
```

2.2.5 Iterative Harvesting

Sickle lets you conveniently iterate through resumption batches without having to deal with `resumptionTokens` yourself:

```
>>> records = sickle.ListRecords(metadataPrefix='oai_dc')
>>> records.next()
<Record oai:eprints.rclis.org:4088>
```

Note that this works with all requests that return more than one element. These are: `ListRecords()`, `ListIdentifiers()`, `ListSets()`, and `ListMetadataFormats()`.

Iterating through the headers returned by `ListIdentifiers`:

```
>>> headers = sickle.ListIdentifiers(metadataPrefix='oai_dc')
>>> headers.next()
<Header oai:eprints.rclis.org:4088>
```

Or through the sets returned by `ListSets`:

```
>>> sets = sickle.ListSets()
>>> sets.next()
<Set Status = In Press>
```

2.2.6 Getting a Single Record

OAI-PMH allows you to get a single record by using the `GetRecord` verb. And so does Sickle:

```
>>> sickle.GetRecord(identifier='oai:eprints.rclis.org:4088',
...                  metadataPrefix='oai_dc')
<Record oai:eprints.rclis.org:4088>
```

2.2.7 Ignoring Deleted Records

The `ListRecords()` and `ListIdentifiers()` methods take an optional parameter `ignore_deleted`. If it is set to `True`, the returned `OAIIterator` will skip deleted records/headers:

```
>>> records = sickle.ListRecords(metadataPrefix='oai_dc', ignore_deleted=True)
```

2.3 API

2.3.1 The Sickle Client

class sickle.app.**Sickle**(*endpoint*, *http_method='GET'*, *protocol_version='2.0'*, *max_retries=5*, *timeout=None*, *class_mapping=None*, *auth=None*)

Client for harvesting OAI interfaces.

Use it like this:

```
>>> sickle = Sickle('http://elis.da.ulcc.ac.uk/cgi/oai2')
>>> records = sickle.ListRecords(metadataPrefix='oai_dc')
>>> records.next()
<Record oai:eprints.rclis.org:3780>
```

Parameters

- **endpoint** (*str*) – The endpoint of the OAI interface.
- **http_method** (*str*) – Method used for requests (GET or POST, default: GET).
- **protocol_version** (*str*) – The OAI protocol version.
- **max_retries** (*int*) – Number of retries if HTTP request fails.
- **timeout** (*int*) – Timeout for HTTP requests.
- **class_mapping** (*dict*) – A dictionary that maps OAI verbs to classes representing OAI items. If not provided, `sickle.app.DEFAULT_CLASS_MAPPING` will be used.
- **auth** (*tuple*) – An optional tuple ('username', 'password') for accessing protected OAI interfaces.

last_response
Contains the last response that has been received.

GetRecord(***kwargs*)
Issue a ListSets request.

Return type sickle.models.Record

Identify()
Issue an Identify request.

Return type sickle.models.Identify

ListIdentifiers(*ignore_deleted=False*, ***kwargs*)
Issue a ListIdentifiers request.

Parameters **ignore_deleted** – If set to `True`, the resulting sickle.app.OAIIterator will skip records flagged as deleted.

Return type sickle.app.OAIIterator

ListMetadataFormats(***kwargs*)
Issue a ListMetadataFormats request.

Return type sickle.app.OAIIterator

ListRecords(*ignore_deleted=False*, ***kwargs*)
Issue a ListRecords request.

Parameters **ignore_deleted** – If set to `True`, the resulting `sickle.app.OAIIterator` will skip records flagged as deleted.

Return type `sickle.app.OAIIterator`

`ListSets`(***kwargs*)
Issue a ListSets request.

Return type `sickle.app.OAIIterator`

`harvest`(***kwargs*)
Make HTTP requests to the OAI server.

Parameters **kwargs** – The OAI HTTP arguments.

Return type `sickle.app.OAIResponse`

2.3.2 Working with OAI Responses

class `sickle.app.`**`OAIResponse`**(*response*, *params*)
A response from an OAI server.

Provides access to the returned data on different abstraction levels.

Parameters

- **response** – The original HTTP response.
- **params** (*dict*) – The OAI parameters for the request.

`raw`
The server's response as unicode.

`xml`
The server's response as parsed XML.

2.3.3 Iterating through OAI Items

class `sickle.app.`**`OAIIterator`**(*oai_response*, *sickle*, *ignore_deleted=False*)
Iterator over OAI records/identifiers/sets transparently aggregated via OAI-PMH.

Can be used to conveniently iterate through the records of a repository.

Parameters

- **oai_response** (`sickle.app.OAIResponse`) – The first OAI response.
- **sickle** (`sickle.app.Sickle`) – The Sickle object that issued the first request.
- **ignore_deleted** (*bool*) – Flag for whether to ignore deleted records.

`sickle`
The `sickle.app.Sickle` instance used for making requests to the server.

`verb`
The OAI verb used for making requests to the server.

`element`
The name of the OAI item to iterate on (`record`, `header`, `set` or `metadataFormat`).

`resumption_token`
The content of the XML element `resumptionToken` from the last request.

`ignore_deleted`
Flag for whether to skip records marked as deleted.

`next`()
Return the next record/header/set.

2.3.4 Classes for OAI Items

The following classes represent OAI-specific items like records, headers, and sets. All items feature the attributes `raw` and `xml` which contain their original XML representation as unicode and as parsed XML objects.

Identify Object

The Identify object is generated from Identify responses and is returned by `sickle.app.Sickle.Identify()`. It contains general information about the repository.

class `sickle.models.`**Identify**(*identify_response*)

Represents an Identify container.

This object differs from the other entities in that is has to be created from a `sickle.app.OAIResponse` instead of an XML element.

Parameters **identify_response** (`sickle.app.OAIResponse`) – The response for an Identify request.

Note: As the attributes of this class are auto-generated from the Identify XML elements, some of them may be missing for specific OAI interfaces.

---

adminEmail
The content of the element `adminEmail`. Normally the repository's administrative contact.

baseURL
The content of the element `baseURL`, which is the URL of the repository's OAI endpoint.

respositoryName
The content of the element `repositoryName`, which contains the name of the repository.

deletedRecord
The content of the element `deletedRecord`, which indicates whether and how the repository keeps track of deleted records.

delimiter
The content of the element `delimiter`.

description
The content of the element `description`, which contains a description of the repository.

earliestDatestamp
The content of the element `earliestDatestamp`, which indicates the datestamp of the oldest record in the repository.

granularity
The content of the element `granularity`, which indicates the granularity of the used dates.

oai_identifier
The content of the element `oai-identifier`.

Note: `oai-identifier` is not a valid name in Python.

---

protocolVersion
The content of the element `protocolVersion`, which indicates the version of the OAI protocol implemented by the repository.

repositoryIdentifier
The content of the element `repositoryIdentifier`.

sampleIdentifier
The content of the element `sampleIdentifier`, which usually contains an example of an identifier used by this repository.

scheme
The content of the element `scheme`.

raw
The original XML as unicode.

Record Object

Record objects represent single OAI records.

class `sickle.models.`**Record**(*record_element*, *strip_ns=True*)
Represents an OAI record.

Parameters

- **record_element** – The XML element 'record'.
- **strip_ns** – Flag for whether to remove the namespaces from the element names.

header
Contains the record header represented as a `sickle.models.Header` object.

deleted
A boolean flag that indicates whether this record is deleted.

raw
The original XML as unicode.

Header Object

Header objects represent OAI headers.

class `sickle.models.`**Header**(*header_element*)
Represents an OAI Header.

Parameters **header_element** – The XML element 'header'.

raw
The original XML as unicode.

Set Object

class `sickle.models.`**Set**(*set_element*)
Represents an OAI set.

Parameters **set_element** – The XML element 'set'.

setName
The name of the set.

setSpec
The identifier of this set used for querying.

raw
The original XML as unicode.

MetadataFormat Object

class sickle.models.**MetadataFormat**(*mdf_element*)
Represents an OAI MetadataFormat.

Parameters **mdf_element** – The XML element 'metadataFormat'.

metadataPrefix
The prefix used to identify this format.

metadataNamespace
The namespace URL for this format.

schema
The URL to the schema file of this format.

raw
The original XML as unicode.

2.4 Harvesting Custom Metadata Formats

By default, Sickle's unpacking of the metadata into a Python dictionary is taylored to work only with Dublin-Core-encoded metadata payloads. Other formats most probably won't be unpacked correctly, especially if they are more hierarchically structured than Dublin Core.

In case your want to harvest these more complex formats, you have to write your own record model classes by subclassing the default implementation that unpacks the metadata XML:

```
from sickle.models import Record

class MyRecord(Record):
    # Your XML unpacking implementation goes here.
    pass
```

Note: Take a look at the implementation of `sickle.models.Record` to get an idea of how to do this.

Next, associate your implementation with OAI verbs in the `Sickle` object. In this case, we want the `Sickle` object to use our implementation to represent items returned by ListRecords and GetRecord responses:

```
sickle = Sickle('http://...')
sickle.class_mapping['ListRecords'] = MyRecord
sickle.class_mapping['GetRecord'] = MyRecord
```

If you need to rewrite *all* item implementations, you can also provide a complete mapping to the `Sickle` object at instantiation:

```
my_mapping = {
    'ListRecords': MyRecord,
    'GetRecord': MyRecord,
    # ...
}

sickle = Sickle('http://...', class_mapping=my_mapping)
```

2.5 Development

2.5.1 Get the Code

Sickle is developed on GitHub.

2.5.2 Testing

Sickle is tested with nose.

To run the tests, type:

```
python setup.py nosetests
```

Since the tests should not rely on an external OAI server, static OAI responses stored in files are used instead. To this end, a mock version of the `sickle.app.Sickle.harvest()` method is created that reads the stored responses:

2.6 Credits

- pyoai provided valueable inspiration
- Sickle logo: Free Valentina typeface by Pedro Arilla and public domain image by Pearson Scott Foresman.