biblio.webquery api

API Documentation

May 6, 2009

Contents

1 Package biblio.webquery

Querying webservices for bibliographic information. **Version:** 0.4b

1.1 Modules

2 Module biblio.webquery.basewebquery

A base class for querying webservices.

2.1 Class BaseWebquery

object

biblio.webquery.impl.ReprObj

biblio.webquery.basewebquery.BaseWebquery

Known Subclasses: biblio.webquery.basewebquery.BaseKeyedWebQuery, biblio.webquery.xisbn.XisbnQuery, biblio.webquery.worldcat.WorldcatQuery, biblio.webquery.loc.LocQuery

A base class for querying webservices.

This serves as a foundation for other web-query classes, centralising a small amount of functionality and providing a common interface.

2.1.1 Methods

__init__(*self*, *root_url*, *timeout*=`5.0`, *limits*=`[]`)

Ctor, allowing the setting of the webservice, timeout and limits on use. **Parameters**

`root_url`: The url to be used as the basis for all requests to this service. It should be the common "stem" that does not vary for any request. *(type=string)*

`timeout`: How many seconds to wait for a response. *(type=int or float)*

`limits`: A list of QueryThrottles to impose upon the use of this webservice. *(type=iterable)*

Overrides: object.__init__

request(*self*, *sub_url*)

Send a request to the webservice and return the response.

This is the low-level calls that checks any throttling, send the request and actually fetches the response data. For consistency, all service access should be placed through here.

Parameters

`sub_url`: This will be added to the root url set in the c'tor and used as the actual url that is requested. *(type=string)*

Return Value

The data in the webservice response.

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

2.1.2 Properties

Name	Description
Inherited from object	
__class__	

2.2 Class BaseKeyedWebQuery

object

biblio.webquery.impl.ReprObj

biblio.webquery.basewebquery.BaseWebquery

biblio.webquery.basewebquery.BaseKeyedWebQu

Known Subclasses: biblio.webquery.isbndb.IsbndbQuery

A Webquery that requires an access key.

2.2.1 Methods

__init__(*self*, *root_url*, *key*, *timeout*=`5.0`, *limits*=`[]`)

Ctor, allowing the setting of a webservice access key. **Parameters**

`root_url:`	See `BaseWebquery`. Either this or the sub_url passed to `request` must include a keyword formatting for the access key, i.e. `%(key)s`.
`key:`	The access or PAI key to be passed to the webservice for access. *(type=string)*
`timeout:`	See `BaseWebquery`.
`limits:`	See `BaseWebquery`.

Overrides: object.__init__

Inherited from biblio.webquery.basewebquery.BaseWebquery(Section 2.1)

request()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

2.2.2 Properties

Name	Description
Inherited from object	
__class__	

3 Module biblio.webquery.bibrecord

Classes for representing bibliographic records and authors.

These are not the intended major function of this module, but are necessary for translation between formats.

Variously based upon:

- pymarc
- bibconverter
- bibliograph.core and bibliograph.parsing

3.1 Variables

Name	Description
SHORT_TITLE_SPLIT_R-E	**Value:** `re.compile(r'[:\?]')`

3.2 Class BibRecord

```
object ─┐
        │
biblio.webquery.impl.ReprObj ─┐
                              │
                    biblio.webquery.bibrecord.BibRecord
```

3.2.1 Methods

__init__(*self*)

C'tor. Overrides: object.__init__

add_ext_references(*self*, *key*, *val*)

get_short_title(*self*)

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

3.2.2 Properties

Name	Description
short_title	
Inherited from object	
__class__	

3.3 Class PersonalName

object

biblio.webquery.impl.ReprObj

biblio.webquery.bibrecord.PersonalName

A name, as used for authors and editors.

The terms 'given', 'other' and 'family' are used in preference to other schemes, as they are more culture-neutral and do not assume any particular ordering.

given The first / christian or forename, e.g. 'John'.

other Any middle names, e.g. 'James Richard'.

family surname, last name, e.g. 'Smith'.

3.3.1 Methods

__init__(*self*, *given*, *other*=`None`, *family*=`None`, *title*=`None`, *prefix*=`None`, *suffix*=`None`)

C'tor, requiring only the given name.

Note that the only required argument is the given name, allowing single names (e.g. 'Madonna'). Also the order of positional arguments allows a a regular name to be passed as 'John', 'James', 'Smith'. Overrides: object.__init__

__unicode__(*self*)

Return a readable formatted version of the name. Overrides: biblio.webquery.impl.ReprObj.__unicode__

__repr__(*self*)

Return a representation of this object. Overrides: object.__repr__

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__str__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

3.3.2 Properties

Name	Description
Inherited from object	
__class__	

4 Module biblio.webquery.errors

Various errors thrown by the the module.

4.1 Class QueryError

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
          exceptions.StandardError ─┐
                   exceptions.ValueError ─┐
                                 biblio.webquery.errors.QueryError
```

Raised when there is an problem with a queries reply.

4.1.1 Methods

__init__(*self*, *msg*)

C'tor. Overrides: object.__init__

Inherited from exceptions.ValueError

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__()

Inherited from object

__hash__(), __reduce_ex__()

4.1.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

4.2 Class ParseError

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
          exceptions.StandardError ─┐
                  exceptions.ValueError ─┐
                                  biblio.webquery.errors.ParseError
```

Thrown when parsing webservice formats.

4.2.1 Methods

__init__(*self*, *msg*)

C'tor. Overrides: object.__init__

Inherited from exceptions.ValueError

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__()

Inherited from object

__hash__(), __reduce_ex__()

4.2.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	
Inherited from object	
__class__	

4.3 Class QueryThrottleError

```
object ─┐
exceptions.BaseException ─┐
          exceptions.Exception ─┐
          exceptions.StandardError ─┐
                exceptions.RuntimeError ─┐
                                         biblio.webquery.errors.QueryThrottleError
```

An exception to throw when a query limit has been exceeded.

It serves little purpose except to distinguish failures caused by exceeding query limits.

4.3.1 Methods

__init__(*self*, *msg*=`None`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.RuntimeError

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__()

Inherited from object

__hash__(), __reduce_ex__()

4.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

5 Module biblio.webquery.impl

Various (fragile) implementation details and utilities.

Don't reply on these because they may go away.

5.1 Functions

assert_or_raise(*cond*, *error_cls*, *error_msg*=`None`)

If a condition is not met, raise a assertion with this message.

5.2 Class ReprObj

object ─┐
biblio.webquery.impl.ReprObj

Known Subclasses: biblio.webquery.querythrottle.BaseQueryThrottle, biblio.webquery.basewebquery.Ba
biblio.webquery.bibrecord.BibRecord, biblio.webquery.bibrecord.PersonalName

A class with an simple and consistent printable version.

5.2.1 Methods

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

__unicode__(*self*)

__repr__(*self*)

repr(x)

Overrides: object.__repr__ extit(inherited documentation)

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

5.2.2 Properties

Name	Description
Inherited from object	
__class__	

6 Module biblio.webquery.isbndb

Querying the ISBNDb for bibliographic information.

6.1 Functions

isbndb_xml_to_bibrecords(*xml_txt*)

6.2 Class IsbndbQuery

```
object ─┐
        │
biblio.webquery.impl.ReprObj ─┐
                              │
biblio.webquery.basewebquery.BaseWebquery ─┐
                                           │
biblio.webquery.basewebquery.BaseKeyedWebQuery ─┐
                                                │
                                                biblio.webquery.isbndb.IsbndbQuery
```

6.2.1 Methods

__init__(*self*, *key*, *timeout*=`5.0`, *limits*=`None`)

C'tor, accepting an access key. **Parameters**

`root_url:`	See `BaseWebquery`. Either this or the sub_url passed to `request` must include a keyword formatting for the access key, i.e. `%(key)s`.
`key:`	The access or PAI key to be passed to the webservice for access.
`timeout:`	See `BaseWebquery`.
`limits:`	See `BaseWebquery`.

Overrides: object.__init__

query_service(*self*, *index*, *value*, *results*)

A generalised query for ISBNdb.

This serves a general way of accessing all the methods available for ISBNdb. It also normalises the ISBN to a suitable form for submission. Note that it is probably possible to form a bad query with the wrong combination of parameters. **Parameters**

- `index`: The index to search in ISBNdb. *(type=string)*
- `value`: The value to search for in the index.. *(type=string)*
- `results`: A list of the data to include in the response. *(type=iterable)*

Return Value

The response received from the service.

query_bibdata_by_isbn(*self*, *isbn*, *format*=`'bibrecord'`)

Return publication data based on ISBN. **Parameters**

- `isbn`: An ISBN-10 or ISBN-13. *(type=string)*
- `format`: The desired format for the results. *(type=string)*

Return Value

Publication data in Xisbn XML format.

query_author_by_name(*self*, *name*, *fields*=`None`)

Search author data based on name. **Parameters**

- `name`: The name to search for. *(type=string)*
- `fields`: What result blocks to return.. *(type=iterable)*

Return Value

Publication data in ISBNdb XML format.

query_author_by_id(*self*, *auth_id*, *fields*=`None`)

Search author data based on ID. **Parameters**

`auth_id`: The ISBN “person_id” to search for. *(type=string)*

`fields`: What result blocks to return.. *(type=iterable)*

Return Value

Publication data in ISBNdb XML format.

Inherited from biblio.webquery.basewebquery.BaseWebquery(Section 2.1)

request()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

6.2.2 Properties

Name	Description
Inherited from object	
__class__	

7 Module biblio.webquery.loc

Querying the Library of Congress for bibliographic information.

7.1 Variables

Name	Description
LOC_ROOTURL	**Value:** `'http://z3950.loc.gov:7090/voyager?operation=searchRetrie...`

7.2 Class LocQuery

object

biblio.webquery.impl.ReprObj

biblio.webquery.basewebquery.BaseWebquery

biblio.webquery.loc.LocQuery

7.2.1 Methods

__init__(*self*, *timeout*=`5.0`, *limits*=`None`)

C'tor. **Parameters**

`root_url`:	The url to be used as the basis for all requests to this service. It should be the common “stem” that does not vary for any request.
`timeout`:	How many seconds to wait for a response.
`limits`:	A list of QueryThrottles to impose upon the use of this webservice.

Overrides: object.__init__

query_bibdata_by_isbn(*self*, *isbn*, *format*=`'MODS'`)

Return the metadata for a publication specified by ISBN.

Inherited from biblio.webquery.basewebquery.BaseWebquery(Section 2.1)

request()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

7.2.2 Properties

Name	Description
Inherited from object	
__class__	

8 Module biblio.webquery.querythrottle

Classes for throttling web-queries, so as to stay within limits.

8.1 Variables

Name	Description
FAIL_AND_RAISE	**Value:** `'RAISE'`
FAIL_AND_WAIT	**Value:** `'WAIT'`

8.2 Class BaseQueryThrottle

object
biblio.webquery.impl.ReprObj
biblio.webquery.querythrottle.BaseQueryThrottle

Known Subclasses: biblio.webquery.querythrottle.AbsoluteNumberThrottle, biblio.webquery.querythrot

A limit upon query usage.

Often webservices will request that users restrict themselves to a request every second, or no more than 1000 a day, etc. This is a base class for implementing those limits. Different restrictions can be implemented in derived classes.

Limits are constructed with set behaviour

8.2.1 Methods

__init__(*self*, *fail_action*=`None`, *wait_duration*=`1.0`)

Ctor, allowing the polling period and failure behaviour to be set. Overrides: object.__init__

check_limit(*self*, *wquery*)

Has the query exceeded its limit?

This is a primarily internal method for testing whether a limit has been reached. Handling that circumstance is left to the calling method `check_limit`. This should be overridden in derived class to implement different throttling methods. **Parameters**

`wquery`: The object or service to be throttled. This allows the same limit to service several objects in different ways (e.g. by having them share the same limit, or be handled independently).

Return Value

A boolean, giving whether the query is within limit or not.

within_limit(*self*, *wquery*)

Has the query exceeded its limit?

This should be called by services to test whether a limit has been reached. Handling that circumstance is left to the calling method `check_limit`. This should be overridden in derived class to implement different throttling methods. **Parameters**

`wquery`: See `check_limit`

log_success(*self*, *wquery*)

Sucessful queries will probably effect the success of future ones, so here is a place to log them.

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

8.2.2 Properties

Name	Description
Inherited from object	
__class__	

8.3 Class WaitNSecondsThrottle

```
object ─┐
        │
biblio.webquery.impl.ReprObj ─┐
                              │
biblio.webquery.querythrottle.BaseQueryThrottle ─┐
                                                 │
                                   biblio.webquery.querythrottle.WaitNSecondsT
```

Known Subclasses: biblio.webquery.querythrottle.WaitOneSecondThrottle

Limit a query to every N seconds at most.

By default this throttle holds the query until the wait period is over. Note that this throttle can be used across a set of queries, so that the limit applies for the set. In this case the waiting behaviour could be undesirable, with a large population of queries on hold.

8.3.1 Methods

__init__(*self*, *wait*, *fail_action*=`'WAIT'`)

C'tor, allowing the wait period and failure behaviour to be set. **Parameters**

`wait:`	The period to enforce between queries. *(type=int or float)*
`fail_action:`	See `BaseQueryThrottle`.

Overrides: object.__init__

within_limit(*self*, *wquery*)

Has it been longer than the wait period since the last query? **Parameters**

`wquery:` See `check_limit`

Overrides: biblio.webquery.querythrottle.BaseQueryThrottle.within_limit

log_success(*self*, *wquery*)

Sucessful queries will probably effect the success of future ones, so here is a place to log them. Overrides: biblio.webquery.querythrottle.BaseQueryThrottle.log_success extit(inherited documentation)

Inherited from biblio.webquery.querythrottle.BaseQueryThrottle(Section 8.2)

check_limit()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

8.3.2 Properties

Name	Description
Inherited from object	
__class__	

8.4 Class WaitOneSecondThrottle

```
object ─┐
        │
biblio.webquery.impl.ReprObj ─┐
                              │
biblio.webquery.querythrottle.BaseQueryThrottle ─┐
                                                 │
  biblio.webquery.querythrottle.WaitNSecondsThrottle ─┐
                                                      │
                                                      biblio.webquery.querythrottle.WaitOneSe
```

Limit a query to once every second at most.

This is a common limit, and so is provided as a convenience.

8.4.1 Methods

__init__(*self*, *fail_action*=`'WAIT'`)

C'tor, allowing the wait period and failure behaviour to be set. **Parameters**

`wait:`	The period to enforce between queries.
`fail_action:`	See `BaseQueryThrottle`.

Overrides: object.__init__ extit(inherited documentation)

Inherited from biblio.webquery.querythrottle.WaitNSecondsThrottle(Section 8.3)

log_success(), within_limit()

Inherited from biblio.webquery.querythrottle.BaseQueryThrottle(Section 8.2)

check_limit()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

8.4.2 Properties

Name	Description
Inherited from object	
__class__	

8.5 Class AbsoluteNumberThrottle

```
object ─┐
        │
biblio.webquery.impl.ReprObj ─┐
                              │
biblio.webquery.querythrottle.BaseQueryThrottle ─┐
                                                 │
                                    biblio.webquery.querythrottle.AbsoluteNumb
```

Limit a query to a maximum number.

Many web-services have a per-day query limit (e.g. 500 per day for ISBNdb). It is difficult to implement this across multiple invocations of the query objects and Python interpreter, but this can serve as a crude implementation. By default, it raises an exception if the limit is reached.

8.5.1 Methods

__init__(*self*, *max*, *fail_action*=`'RAISE'`)

C'tor, allowing the maximum queries and failure behaviour to be set.

Parameters

`max:`	The total number of queries allowed. *(type=int)*
`fail_action:`	See `BaseQueryThrottle`.

Overrides: object.__init__

within_limit(*self*, *wquery*)

Have fewer queries been posted than the limit?

Note that if multiple queries simulatanoeusly test via this function, exceeding the limit is possible. **Parameters**

`wquery:` See `check_limit`

Overrides: biblio.webquery.querythrottle.BaseQueryThrottle.within_limit

log_success(*self*, *wquery*)

Sucessful queries will probably effect the success of future ones, so here is a place to log them. Overrides: biblio.webquery.querythrottle.BaseQueryThrottle.log_success extit(inherited documentation)

Inherited from biblio.webquery.querythrottle.BaseQueryThrottle(Section 8.2)

check_limit()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

8.5.2 Properties

Name	Description
Inherited from object	
__class__	

8.6 Class Max500Throttle

```
object ─┐
        │
        biblio.webquery.querythrottle.Max500Throttle
```

Limit a query to 500 attempts at most.

This is a common limit, and so is provided as a convenience.

8.6.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __str__()

8.6.2 Properties

Name	Description
Inherited from object	
__class__	

9 Package biblio.webquery.scripts

Scripts that use biblio.webquery.

9.1 Modules

- **common**: Function shared between the scripts.
  (Section 10, p. 28)
- **config**: Constants and definitions for scripts.
  (Section 11, p. 29)
- **queryisbn**: Retreive bibliographic information for a given ISBN.
  (Section 12, p. 30)
- **renamebyisbn**: Rename files as by the ISBN buried in their original name.
  (Section 13, p. 31)

10 Module biblio.webquery.scripts.common

Function shared between the scripts.

10.1 Functions

add_shared_options(*optparser*)

check_shared_options(*options*, *optparser*)

construct_webquery(*service*, *key*)

10.2 Variables

Name	Description
script_version	**Value:** `'0.4b'`

11 Module biblio.webquery.scripts.config

Constants and definitions for scripts. **Version:** 0.4b

11.1 Variables

Name	Description
WEBSERVICES	**Value:** `[{'ctor': <class 'biblio.webquery.xisbn.XisbnQuery'>, 'id...`
DEFAULT_WEBSERVIC-E	**Value:** `{'ctor': <class 'biblio.webquery.xisbn.XisbnQuery'>, 'id'...`
WEBSERVICE_LOOKU-P	**Value:** `{'isbndb': {'ctor': <class 'biblio.webquery.isbndb.Isbndb...`

12 Module biblio.webquery.scripts.queryisbn

Retreive bibliographic information for a given ISBN.

12.1 Functions

parse_args()

main()

12.2 Variables

Name	Description
PRINT_FIELDS	**Value:** `['title', 'authors', 'publisher', 'year', 'lang']`

13 Module biblio.webquery.scripts.renamebyisbn

Rename files as by the ISBN buried in their original name.

13.1 Functions

parse_args()

dir_base_ext_from_path(*fpath*)

Return a files base name and extension from it's path.

rename_file(*oldpath*, *newname*)

Rename a file, while keeping it in the same location.

extract_isbn_from_filename(*fname*)

generate_new_name(*bibrec*, *options*)

postprocess_name(*name*, *options*)

main()

13.2 Variables

Name	Description
ISBN10_PAT	**Value:** `'(\\d{9}[\\d|X])'`
ISBN13_PAT	**Value:** `'(\\d{13})'`
ISBN_PATS	**Value:** `['\\(ISBN([^\\)]+)\\)', '^(\\d{13})$', '^(\\d{13})[\\b|_|...`
ISBN_RE	**Value:** `[re.compile(r'(?i)\(ISBN([^\)]+)\)'), re.compile(r'(?i)^(...`
DEF_NAME_FMT	**Value:** `'%(auth)s%(year)s_%(short_title)s_(isbn%(isbn)s)'`
DEF_STRIP_CHARS	**Value:** `':!,\'".?()'`

continued on next page

Name	Description
DEF_BLANK_CHARS	**Value:** `’’`
STRIP_CHARS_RE	**Value:** `re.compile(r’[\’":,!\.\?\(\)]’)`
COLLAPSE_SPACE_RE	**Value:** `re.compile(r’\s+’)`
CASE_CHOICES	**Value:** `[’orig’, ’upper’, ’lower’]`
p	**Value:** `’\\D(\\d{9}[\\d|X])$’`

14 Module biblio.webquery.utils

Various utilities.

14.1 Functions

normalize_isbn(*isbn*)

Remove formatting from an ISBN, making it suitable for web-queries.

parse_single_name(*name_str*)

Clean up an indivdual name into a more consistent format.

parse_names(*name_str*)

Clean up a list of names into a more consistent format.

Xisbn data can be irregularly formatted, unpredictably including ancillary information. This function attempts to cleans up the author field into a list of consistent author names.

For example:

```
>>> n = parse_names ("Leonard Richardson and Sam Ruby.")
>>> print (n[0].family == 'Richardson')
True
>>> print (n[0].given == 'Leonard')
True
>>> print (not n[0].other)
True
>>> n = parse_names ("Stephen P. Schoenberger, Bali Pu-
lendran")
>>> print (n[0].family == 'Schoenberger')
True
>>> print (n[0].given == 'Stephen')
True
>>> print (n[0].other == 'P.')
True
>>> n = parse_names ("Madonna")
>>> print (not n[0].family)
True
>>> print (n[0].given == 'Madonna')
True
>>> print (not n[0].other)
True
```

Parameters

`name_str`: The "author" attribute from a Xisbn record in XML.
(type=string)

Return Value

A list of the authors in "reverse" format, e.g. "['Smith, A. B.', 'Jones, X. Y.']"

parse_editing_info(*name_str*)

Detect whethers names are editors and returns

Returns: Whether editing information was recognised and the name with that editing information removed.

For example:

```
>>> parse_editing_info ("Leonard Richard-
son and Sam Ruby.")
(False, 'Leonard Richardson and Sam Ruby.')
>>> parse_editing_info ("Ann Thomson.")
(False, 'Ann Thomson.')
>>> parse_editing_info ("Stephen P. Schoen-
berger, Bali Pulendran, editors.")
(True, 'Stephen P. Schoenberger, Bali Pulendran')
>>> print parse_editing_info ("Madonna")
(False, 'Madonna')
```

parse_publisher(*pub_str*)

Parse a string of publisher information.

As with author names, publication details are often inconsistently set out, even in bibliographic data. This function attempts to parse out and normalise the details.

For example:

```
>>> parse_publisher ('New York: Asia Pub. House, c1979.')
('Asia Pub. House', 'New York', '1979')
>>> parse_publisher ('New York : LearningExpress, 1999.')
('LearningExpress', 'New York', '1999')
>>> parse_publisher ('HarperTorch')
('HarperTorch', '', '')
>>> parse_publisher ('Berkeley Heights, NJ: Enslow Pub-
lishers, c2000.')
('Enslow Publishers', 'Berkeley Heights, NJ', '2000')
```

Parameters

`pub_str`: text giving publisher details. *(type=string)*

Return Value

A tuple of strings, being (<publisher>, <city of publication>, <year of publication>). If no value is available, an empty string returned.

14.2 Variables

Name	Description
EDITOR_PATS	**Value:** `[re.compile(r'(?iu)^edited by\s+'), re.compile(r'(?iu)\s*...`
STRIP_PATS	**Value:** `[re.compile(r'(?iu)^by\s+'), re.compile(r'(?iu)\s*;\s+wit...`
AND_PAT	**Value:** `re.compile(r'\s+and\s+')`
COLLAPSE_SPACE_RE	**Value:** `re.compile(r'\s+')`
PUBLISHER_RES	**Value:** `[re.compile(r'(?iu)^(?P<city>.*)\s*:\s*(?P<pub>.*)\s*,\s*...`
p	**Value:** `'^(?P<pub>.*)\\.?$'`
x	**Value:** `'\\s*;.*$'`

15 Module biblio.webquery.worldcat

Querying WorldCat for bibliographic information and normalising the results.

15.1 Functions

parse_authors(*auth_str*)

Clean up Worldcat author information into a more consistent format.

Worldcat data can be irregularly formatted, unpredictably including ancillary information. This function attempts to cleans up the author field into a list of consistent author names.

For example:

```
>>> parse_authors ("Leonard Richardson and Sam Ruby.")
['Richardson, Leonard', 'Ruby, Sam']
>>> parse_authors ("Ann Thomson.")
['Thomson, Ann']
>>> parse_authors ("Stephen P. Schoenberger, Bali Pulen-
dran, editors.")
['Schoenberger, Stephen P.', 'Pulendran, Bali']
>>> parse_authors ("Madonna")
['Madonna']
```

Parameters

`auth_str`: The "author" attribute from a Worldcat record in XML. *(type=string)*

Return Value

A list of the authors in "reverse" format, e.g. "['Smith, A. B.', 'Jones, X. Y.']"

parse_metadata(*mdata_xml*)

Retrieve fields from metadata and return and cleanup in a sensible form.

Parameters

`mdata_xml`: An Worldcat record in XML. *(type=string)*

Return Value

A dictionary with keys "year", "title" and "authors" parsed from the Worldcat record. If a field is not present or parseable, neither is the key.

parse_title(*title*)

Clean up Worldcat title information into a more consistent format.

Althogh this currently does nothing, in the future it will normalise the titles, e.g. by stripping out subtitle and edition information.

15.2 Variables

Name	Description
AND_PAT	**Value:** `re.compile(r'\s+and\s+')`
STRIP_PATS	**Value:** `[re.compile(r'(?iu)^((edited )?by\s+)'), re.compile(r'(?i...`
WORLDCAT_ROOTURL	**Value:** `'http://xisbn.worldcat.org/webservices/xid/isbn/'`
x	**Value:** `'\\([^\\)]+\\)'`

15.3 Class WorldcatQuery

object

biblio.webquery.impl.ReprObj

biblio.webquery.basewebquery.BaseWebquery

biblio.webquery.worldcat.WorldcatQuery

15.3.1 Methods

__init__(*self*)

C'tor. **Parameters**

`root_url`:	The url to be used as the basis for all requests to this service. It should be the common "stem" that does not vary for any request.
`timeout`:	How many seconds to wait for a response.
`limits`:	A list of QueryThrottles to impose upon the use of this webservice.

Overrides: object.__init__

query_mdata_by_isbn(*self*, *isbn*)

Return publication data based on ISBN. **Parameters**

`isbn`: An ISBN-10 or ISBN-13. *(type=string)*

Return Value

Publication data in Worldcat XML format.

Inherited from biblio.webquery.basewebquery.BaseWebquery(Section 2.1)

request()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

15.3.2 Properties

Name	Description
Inherited from object	
__class__	

16 Module biblio.webquery.xisbn

Querying WorldCat xISBN service for bibliographic information.

16.1 Functions

xisbn_py_to_bibrecord(*pytxt*)

Translate the Python text returned by xISBN to a series of BibRecords.

Parameters

`pytxt`: An Xisbn record in Python. *(type=string)*

Return Value

A list of BibRecords.

16.2 Class XisbnQuery

```
object ─┐
        │
biblio.webquery.impl.ReprObj ─┐
                              │
biblio.webquery.basewebquery.BaseWebquery ─┐
                                           │
                                  biblio.webquery.xisbn.XisbnQuery
```

16.2.1 Methods

__init__(*self*, *timeout*=`5.0`, *limits*=`None`)

C'tor. **Parameters**

`root_url`: The url to be used as the basis for all requests to this service. It should be the common "stem" that does not vary for any request.

`timeout`: How many seconds to wait for a response.

`limits`: A list of QueryThrottles to impose upon the use of this webservice.

Overrides: object.__init__

query_service(*self*, *isbn*, *method*, *format*, *fields*=`['*']`)

A generalised query for xISBN.

This serves a general way of accessing all the methods available for xISBN. It also normalises the ISBn to a suitable form for submission. **Parameters**

`isbn`: A normalised ISBN-10 or -13. *(type=string)*

`method`: The request type to make of xISBN. *(type=string)*

`format`: The form for the response. *(type=string)*

`fields`: A list of the fields to include in the response. *(type=iterable)*

Return Value

The response received from the service.

query_bibdata_by_isbn(*self*, *isbn*, *format*=`'bibrecord'`)

Return publication data based on ISBN. **Parameters**

`isbn`: An ISBN-10 or ISBN-13. *(type=string)*

Return Value

Publication data in Xisbn XML format.

query_editions_by_isbn(*self*, *isbn*, *format*=`'xml'`)

Return the editions associated with an ISBN. **Parameters**

`isbn`: An ISBN-10 or ISBN-13. *(type=string)*

`format`: See `query_service`. *(type=string)*

Return Value

Publication data in Xisbn XML format.

query_isbn(*self*, *isbn*, *method*, *format*=`'string'`)

A generalised method for ISBN queries that return ISBNs.

This allows functionality to be shared among the ISBN conversion and checking methods.

query_isbn10_to_13(*self*, *isbn*, *format*=`'string'`)

query_isbn13_to_10(*self*, *isbn*, *format*=`'string'`)

query_fix_isbn_csum(*self*, *isbn*, *format*=`'string'`)

query_hyphenate_isbn(*self*, *isbn*, *format*=`'string'`)

Inherited from biblio.webquery.basewebquery.BaseWebquery(Section 2.1)

request()

Inherited from biblio.webquery.impl.ReprObj(Section 5.2)

__repr__(), __str__(), __unicode__()

Inherited from object

__delattr__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__()

16.2.2 Properties

Name	Description
Inherited from object	
__class__	

17 Module xml.etree.ElementTree

17.1 Functions

Comment(*text*=`None`)

Element(*tag*, *attrib*=`{}`, ***extra*)

PI(*target*, *text*=`None`)

ProcessingInstruction(*target*, *text*=`None`)

SubElement(*parent*, *tag*, *attrib*=`{}`, ***extra*)

XML(*text*)

dump(*elem*)

fromstring(*text*)

iselement(*element*)

parse(*source*, *parser*=`None`)

tostring(*element*, *encoding*=`None`)

17.2 Variables

Name	Description
VERSION	**Value:** `'1.2.6'`

17.3 Class ElementTree

17.3.1 Methods

__init__(*self*, *element*=`None`, *file*=`None`)

find(*self*, *path*)

findall(*self*, *path*)

findtext(*self*, *path*, *default*=`None`)

getiterator(*self*, *tag*=`None`)

getroot(*self*)

parse(*self*, *source*, *parser*=`None`)

write(*self*, *file*, *encoding*=`'us-ascii'`)

17.4 Class QName

17.4.1 Methods

__cmp__(*self*, *other*)

__hash__(*self*)

__init__(*self*, *text_or_uri*, *tag*=`None`)

__str__(*self*)

17.5 Class TreeBuilder

17.5.1 Methods

__init__(*self*, *element_factory*=`None`)

close(*self*)

data(*self*, *data*)

end(*self*, *tag*)

start(*self*, *tag*, *attrs*)

17.6 Class XMLTreeBuilder

17.6.1 Methods

__init__(*self*, *html*=`0`, *target*=`None`)

close(*self*)

doctype(*self*, *name*, *pubid*, *system*)

feed(*self*, *data*)

17.7 Class XMLTreeBuilder

17.7.1 Methods

__init__(*self*, *html*=`0`, *target*=`None`)

close(*self*)

doctype(*self*, *name*, *pubid*, *system*)

feed(*self*, *data*)

17.8 Class iterparse

17.8.1 Methods

__init__(*self*, *source*, *events*=`None`)

__iter__(*self*)

next(*self*)

Index