API Documentation

API Documentation

November 9, 2007

Contents

1 Package curator

1.1 Modules

2 Package curator.etc

2.1 Modules

- **MARC**: Importer and exporter for the subset of MARC format covered by infoscience.
(Section 3, p. 4)

3 Module curator.etc.MARC

Importer and exporter for the subset of MARC format covered by infoscience.

3.1 Functions

mkmap(*db*, *group*)

Convenience function to manipulate enumerated fields.
For instance, to manipulate a document's type, you can do:

```
doctype = mkmap (db, 'doctype')

record ['doctype'] = doctype ['ARTICLE']
```

Parameters

`db:` the database containing the records
(type=instance of `Pyblio.Store.Database`)

`group:` the name of a taxonomy managed by the database
(type=string)

Return Value

a mapping between names and instances of `Pyblio.Attribute.Txo`.

parse(*fd*, *db*=`None`, *quiet*=`False`)

3.2 Variables

Name	Description
root	**Value:** '/usr/lib/python2.5/site-packages/framework-1.0-py2.5.egg...
mapping	**Value:** {1: 'record-id', (13, '', '', 'a'): 'patent-nr', (20, '',...

3.3 Class Reader

object
└ Pyblio.Parsers.Syntax.XMLMARC.Reader
└ Pyblio.Parsers.Syntax.XMLMARC.SimpleReader
└ **curator.etc.MARC.Reader**

MARC parser for infoscience's specific tags.

This MARC parser handles the MARC fields used by infoscience, including the non-standard local tags.

Usually, this class is instanciated by the `parse` function.

3.3.1 Methods

__init__(*self*, *map*={1: 'record-id', (13, '', '', 'a'): 'patent-nr', (20, '',..., *quiet*=False)
x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: Pyblio.Parsers.Syntax.XMLMARC.SimpleReader.__init__

parse(*self*, *fd*, *db*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.SimpleReader.parse

record_begin(*self*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.Reader.record_begin

record_end(*self*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.Reader.record_end

do_980(*self*, *maj*, *ind1*, *ind2*, *values*)

do_024(*self*, *maj*, *ind1*, *ind2*, *values*)

do_773(*self*, *maj*, *ind1*, *ind2*, *values*)

do_520(*self*, *maj*, *ind1*, *ind2*, *values*)

do_440(*self*, *maj*, *ind1*, *ind2*, *values*)

do_260(*self*, *maj*, *ind1*, *ind2*, *values*)

do_245(*self*, *maj*, *ind1*, *ind2*, *values*)

do_973(*self*, *maj*, *ind1*, *ind2*, *values*)

do_700(*self*, *maj*, *ind1*, *ind2*, *values*)

do_852(*self*, *maj*, *ind1*, *ind2*, *values*)

do_856(*self*, *maj*, *ind1*, *ind2*, *values*)

person_add(*self*, *field*, *value*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.SimpleReader.person_add

text_add(*self*, *field*, *value*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.SimpleReader.text_add

date_add(*self*, *field*, *value*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.SimpleReader.date_add

do_unknown(*self*, *tag*, *ind1*, *ind2*, *key*, *value*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.SimpleReader.do_unknown

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__new__(*T*, *S*, ...)
Return Value

```
a new object with type S, a subtype of T
```

__reduce__(...)

helper for pickle

__reduce_ex__(...)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(...)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

do_control(*self*, *field*, *value*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.Reader.do_control

do_default(*self*, *tag*, *ind1*, *ind2*, *values*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.Reader.do_default

id_add(*self*, *field*, *value*)

skip(*self*, *field*, *value*)

url_add(*self*, *field*, *value*, *q*={})

3.3.2 Properties

Name	Description
__class__	**Value:** <attribute '__class__' of 'object' objects>

3.3.3 Class Variables

Name	Description
log	**Value:** logging.getLogger('pyblio.import.xmlmarc')

3.4 Class Writer

object
Pyblio.Parsers.Syntax.XMLMARC.Writer
curator.etc.MARC.Writer

Export to XML MARC format.

3.4.1 Methods

add(*self*, *code*, ***kval*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.Writer.add

record_parse(*self*, *rec*)
Overrides: Pyblio.Parsers.Syntax.XMLMARC.Writer.record_parse

__delattr__(...)

x.__delattr__('name') <==> del x.name

__getattribute__(...)

x.__getattribute__('name') <==> x.name

__hash__(*x*)

hash(x)

__init__(...)

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

__new__(*T*, *S*, *...*)
Return Value
`a new object with type S, a subtype of T`

__reduce__(*...*)

helper for pickle

__reduce_ex__(*...*)

helper for pickle

__repr__(*x*)

repr(x)

__setattr__(*...*)

x.__setattr__('name', value) <==> x.name = value

__str__(*x*)

str(x)

begin(*self*)

control_add(*self*, *code*, *val*)

end(*self*)

single(*self*, *rec*, *field*)

write(*self*, *fd*, *rs*, *db*)

3.4.2 Properties

Name	Description
__class__	**Value:** `<attribute '__class__' of 'object' objects>`

3.4.3 Class Variables

Name	Description
parameters	**Value:** `()`
log	**Value:** `logging.getLogger('pyblio.export.xmlmarc')`

4 Package curator.lib

4.1 Modules

4.2 Functions

newdb(*store*=`None`, *file*=`None`, *args*={})

Create a new empty database.
The database uses the infoscience schema of data.

Return Value

a new database
(type=an instance of `Pyblio.Store.Database`)

5 Module curator.lib.authentification

5.1 Class authentification

Known Subclasses: curator.lib.authentification_invenio.authentification_invenio

Abstract generic class for the authentification on Curator

5.1.1 Methods

__init__(*self*)

login(*self*)

Abstract generic method for the login on Curator

6 Module curator.lib.authentification_invenio

6.1 Class authentification_invenio

curator.lib.authentification.authentification
└ **curator.lib.authentification_invenio.authentification_invenio**

Authentification_invenio is the derivated class of authentification. It's specialised it and offers tools to log a user on Invenio.

6.1.1 Methods

__init__(*self*, *logger*)

Parameters

`logger:` Define a logger to store all events
`username:` The username of the user of Curator to log in on Invenio
`password:` The password of the user of Curator to log in on Invenio
`right:` The right of the user of Curator which is log in on Invenio

Overrides: curator.lib.authentification.authentification.__init__

login(*self*, *login*, *password*)

The login method use a web service to connect on Invenio. After this, it checks the right of the user with the username and the password given.

Parameters

`login:` The username that the user has given in the forms
`password:` The password that the user has given in the forms

Return Value

The right of this user. Admin, curateur... or maybe any right

Overrides: curator.lib.authentification.authentification.login

6.1.2 Class Variables

Name	Description
logger	**Value:** `None`
password	**Value:** `None`
right	**Value:** `None`
username	**Value:** `None`

7 Module curator.lib.exporter

7.1 Class exporter

Known Subclasses: curator.lib.exporter_invenio.exporter_invenio

Abstract generic class to export data from Curator to another plateform like Invenio

7.1.1 Methods

__init__(*self*)

insert(*self*)

Abstract generic method to insert data on the distant plateform

update(*self*)

Abstract generic method to update data on the distant plateform

delete(*self*)

Abstract generic method to delete data on the distant plateform

8 Module curator.lib.exporter_invenio

8.1 Class exporter_invenio

curator.lib.exporter.exporter
└ **curator.lib.exporter_invenio.exporter_invenio**

This class provide some method to export data from Curator to Invenio. These exports can be an insert, an update or a delete.

The connexion with Invenio is made with a web service. The method give a "bibupload" order to Invenio through the web service. With specificaly parameters and the XML MARC, Invenio update his database.

8.1.1 Methods

__init__(*self*, *session*)
Overrides: curator.lib.exporter.exporter.__init__

insert(*self*, *xml_marc*)

Insert one or more new notice in Invenio through the web service. The data are given in XML MARC format. The order which is passed to Invenio is a "bibupload -i"

Parameters
`xml_marc:` It's the XML MARC dataflows that will be sent to Invenio

Return Value
It's an acknowledgment for the deroulment of the process at side of Invenio

Overrides: curator.lib.exporter.exporter.insert

update(*self*, *xml_marc*)

Update one or more new notice in Invenio through the web service. The data are given in XML MARC format. The order which is passed to Invenio is a "bibupload -?"

Parameters
`xml_marc:` It's the XML MARC dataflows that will be sent to Invenio

Return Value
It's an acknowledgment for the deroulment of the process at side of Invenio

Overrides: curator.lib.exporter.exporter.update

delete(*self*, *xml_marc*)

Delete one or more new notice in Invenio through the web service. The data are given in XML MARC format. The order which is passed to Invenio is a "bibupload -r"

Parameters
`xml_marc:` It's the XML MARC dataflows that will be sent to Invenio

Return Value
It's an acknowledgment for the deroulment of the process at side of Invenio

Overrides: curator.lib.exporter.exporter.delete

8.1.2 Class Variables

Name	Description
url	**Value:** `'https://path_to_Invenio:XX/path_to_webservice'`
namespace	**Value:** `'namespace_of_webservice'`

9 Module curator.lib.importer

9.1 Class importer

Known Subclasses: curator.lib.importer_invenio.importer_invenio

Abstract generic class to import data on Curator

9.1.1 Methods

__init__(*self*)

get_data(*self*)

Abstract generic method for search data on the plateform who's connect with Curator

10 Module curator.lib.importer_invenio

10.1 Class importer_invenio

curator.lib.importer.importer
└ **curator.lib.importer_invenio.importer_invenio**

This class provide a method to import data from Invenio to Curator.

The connexion with Invenio is made with a web service.

10.1.1 Methods

__init__(*self*, *session*)

Parameters

`db:` It's a temporary database of Pybliographer's type. With it, Pybliographer can make some traitments on the data which is imports from Invenio.

Overrides: curator.lib.importer.importer.__init__

search(*self*, *field*, *pattern*)

Search data on Invenio for any Curator modules and return it in XML MARC format.

Parameters

`field:` It's the field on which we want make the search.

`pattern:` It's the pattern which the field must be according to.

Return Value

A Pybliographer database (XML MARC(with the notices corresponding to the search.

xml_marc_to_pyblio(*self*, *xml_marc*)

Temporary method to tranform the imported XML MARC. It take car to put the XML MARC in a Pybliographer database et return this db.

Parameters

`xml_marc:` It's the result from the request to Invenio. It's in XML MARC format.

Return Value

Pybliographer database who contains notice from XML MARC.

get_data(*self*)

Abstract generic method for search data on the plateform who's connect with Curator

10.1.2 Class Variables

Name	Description
db	**Value:** `newdb()`
url	**Value:** `'https://path_to_Invenio:XX/path_to_webservice'`
namespace	**Value:** `'namespace_of_webservice'`

11 Module curator.lib.logger

11.1 Functions

my_logger()

This method is used to save all events that pass in Curator. These events are store in a log file which is in /opt/curator/var/curator.log.

Parameters

`logger:` It's the instance of logging class of Python.
`hdlr:` It's used to configure the logging
`formatter:` It's used to format the logging

Return Value

The instance of logging

12 Module curator.lib.main

12.1 Functions

choice_content(*req*, *content*, *args*, *html*, *session*)

The choice_content method choose the content of the page according to the GET variable names "content". His goal it's to load the right page, the right module of Curator.

Parameters

`req:` see index() method from Main.
`content:` It's a variable that comes from GET parameters. She's extract from args. She used to know which content must be appear.
`args:` see index() method from Main.
`html:` see index() method from Main.
`session:` see index() method from Main.

handler(*req*)

This method handle the authentification session of user on Curator

Parameters

`req:` see index() method from Main.
`session:` The authentification session to check the rights and the state of the user

index(*req*, ***args*)

Fonction index must be define for the mod_python knows this is a webpage. Without this method, The browser can't show the page.

Parameters

`req:` The "req" param is a predefined object which is passed by Apache with the method index. For more information about req, see : http://www.modpython.org/live/current/doc-html/pyapi-mprequest.html
`session:` The authentification session to check right of user
`html:` Instance of default html page. Use to show header, menu, content & footer.
`args:` It's an automaticaly generated varibale which contents the GET and POST variable
`argd:` It's the GET and POST variable from args after parsing and typing

12.2 Variables

Name	Description
auth	**Value:** `<curator.lib.authentification_invenio.authentification_in...`
logger	**Value:** `<logging.Logger instance at 0x8a22b8c>`
sample	**Value:** `'<?xml version="1.0" encoding="UTF-8"?>\n<collection xmln...`

13 Package curator.lib.modules

13.1 Modules

14 Package curator.lib.modules.load_notice

14.1 Modules

15 Module curator.lib.modules.load_notice.importer

15.1 Functions

import_notices(*logger*, *session*, *req*)

This is the entry point of module "load notice", in french "Chargement d'un fichier de notice" This method go to search data from laboratory passed in form and export the xml marc file in Invenio in the good laboratory.

Parameters

`logger:` Define a logger to store all events
`session:` see index() method from Main.
`req:` see index() method from Main.

Return Value

Return the html content to view the result of execution of the xml marc data export.

xml_marc_to_pyblio(*logger*, *req*)

Temporary method to tranform the imported XML MARC. It take car to put the XML MARC in a Pybliographer database et return this db.

Parameters

`logger:` Define a logger to store all events
`req:` see index() method from Main.

Return Value

Pybliographer database who contains notice.

do_form()

Little method that create html form to choose a labo and a xml marc file to export

view(*db*)

Method use for view the xml marc file which was upload whit form. Not necessary for the process, it's just for view if the result is correct.

Parameters

`db:` Pybliographer database which contain xml marc from file from form

Return Value

Return the html content for view the xml marc field

16 Module curator.lib.url_handler

16.1 Functions

wash_urlargd(*form*, *content*)

Wash the complete form based on the specification in content. Content is a dictionary containing the field names as a key, and a tuple (type, default) as value.
'type' can be list, str, int, tuple, dict or mod_python.util.Field (for file uploads).
The specification automatically includes the 'ln' field, which is common to all queries.
Arguments that are not defined in 'content' are discarded.

Return Value

a dictionary that can be used for passing function parameters by keywords.

17 Module curator.sample

17.1 Variables

Name	Description
sample	**Value:** '<?xml version="1.0" encoding="UTF-8"?>\n<collection xmln...

18 Package curator.www

18.1 Modules

- **default** *(Section 19, p. 25)*
- **modules** *(Section 20, p. 26)*
  - **load_notice** *(Section 21, p. 27)*
    - **show_notices** *(Section 22, p. 28)*

19 Module curator.www.default

19.1 Class homepage

This class is used to construct the html principal interface of Curator. It contains only some methods who are print html.

19.1.1 Methods

__init__(*self*, *req*, *session*)
Parameters
`req:` see index() method from Main.
`session:` see index() method from Main.

set_session(*self*, *session*)

Update the local session for instance of homepage

do_header(*self*)

Print header for the principal page of Curator

do_footer(*self*)

Print footer for the principal page of Curator

do_head(*self*)

Print title for the principal page of Curator

do_menu(*self*)

Print menu for the principal page of Curator

do_smenu(*self*)

Print submenu for the principal page of Curator

do_content(*self*, *html_content*)

Print content for the principal page of Curator

20 Package curator.www.modules

20.1 Modules

- **load_notice** *(Section 21, p. 27)*
  - **show_notices** *(Section 22, p. 28)*

21 Package curator.www.modules.load_notice

21.1 Modules

- **show_notices** *(Section 22, p. 28)*

22 Module curator.www.modules.load_notice.show_notices

22.1 Class show_notices

This is the html class for load_notice module of Curator. It takes care of print content for this module.

22.1.1 Methods

__init__(*self*)
Parameters
`req`: see index() method from Main.

do_form(*self*)

Print the html form the laboratory and to choice a file of notice to load in.