API Documentation

API Documentation

January 31, 2013

Contents

1 Package changemonitor

Changemonitor – monitoring changes on web

TODO: docstring. 1) koncepce, casti (checking, availability, versioning, differ) 2) user-view vs global-view 3) usage

The very basic and most probable usage:

```
>>> from rrslib.web.changemonitor import Monitor
>>> monitor = Monitor(user_id="rrs_university")
>>> resource = monitor.get("http://www.google.com")
>>> # if the page changed
>>> if resource.check():
>>>     print res.get_diff(start='last', end='now')
```

Date: $21.6.2012 16:08:11$

Author: Stanislav Heller

1.1 Modules

- **_http**: Maly privatni modul zajistujici HTTP pozadavky pro changemonitor. *(Section 2, p. 13)*
- **diff** *(Section 3, p. 17)*
- **errors**: Exceptions raised by rrslib.web.changemonitor package *(Section 4, p. 21)*
- **model**: Model for changemonitor – interface for accessing the data *(Section 5, p. 27)*
- **resolver** *(Section 6, p. 38)*

1.2 Class HTTPDateTime

object
└ **changemonitor._http.HTTPDateTime**

Datetime class for manipulating time within HTTP enviroment.

Usage: >>> h = HTTPDateTime() >>> h HTTPDateTime(Thu, 01 Jan 1970 00:00:00 GMT) >>> h.now() >>> h HTTPDateTime(Sat, 30 Jun 2012 16:09:43 GMT) >>> h.to_httpheader_format() 'Sat, 30 Jun 2012 16:09:43 GMT'

FIXME: solve the problem with GMT: >>> h = HTTPDateTime() >>> h.to_timestamp() -7200.0 # WTF?

1.2.1 Methods

___init___(*self*, *year*=1970, *month*=1, *day*=1, *hour*=0, *minute*=0, *second*=0, *microsecond*=0)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

to_httpheader_format(*self*)

Converts this object into date and time in HTTP-header format.

Return Value

date and time in HTTP format, i.e. 'Wed, 31 Aug 2011 16:45:03 GMT'.

(type=str)

from_httpheader_format(*self*, *timestr*)

Parse http header datetime format and save into this object.

Parameters

`timestr:` date and time in format which is used by HTTP protocol

(type=str)

Return Value

HTTPDateTime object equivalent to date and time of the timestr

(type=HTTPDateTime)

to_timestamp(*self*)

Convert into UNIX timestamp. (seconds since start of the UNIX epoch).

Return Value

unix timestamp

(type=float)

from_timestamp(*self*, *timestamp*)

Set date and time from timestamp.

Parameters

`timestamp:` time since start of the unix epoch

(type=float)

Return Value

HTTPDateTime object representing date and time of the timestamp

(type=HTTPDateTime)

to_datetime(*self*)

Convert the date and time from this object into python's datetime.datetime.

Return Value

datetime object equivalent to date and time of this object

(type=datetime.datetime)

from_datetime(*self*, *datetimeobj*)

from_gridfs_upload_date(*self*, *upload_date*)

Convert the date and time from grid_file.update_date string to HTTPDateTime object.

Parameters

`update_date:` time string from grid_file.update_date

Return Value

HTTPDateTime object representing date and time of update_date

(type=HTTPDateTime)

now(*self*)

Set the time of this object as current time (time.time())

Return Value

HTTPDateTime object representing current date and time.

(type=HTTPDateTime)

___repr___(*self*)

repr(x)

Overrides: object.___repr___ extit(inherited documentation)

___lt___(*self*, *other*)

___le___(*self*, *other*)

___eq___(*self*, *other*)

___ne___(*self*, *other*)

___gt___(*self*, *other*)

___ge___(*self*, *other*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___setattr___(), ___sizeof___(), ___str___(),

___subclasshook___()

1.2.2 Properties

Name	Description
Inherited from object	
___class___	

1.3 Class MonitoredResource

object
└
changemonitor.MonitoredResource

Monitored resource (URL). The ressource is generally any document in any format, but most often it will be HTML code.

This class wraps the URL content and metadata.

The contents can be manipulated within the time so it can provide information about how the content changed in different versions of the document.

Warning: Application developers should generally not need to instantiate this class directly. The only correct way how to get this object is through Monitor.get() method.

Design pattern: Active Record

Example of checking new version of document:

```
>>> from rrslib.web.changemonitor import Monitor
>>> monitor = Monitor(user_id="myuid")
>>> monitor
Monitor(conn=Connection('localhost', 27017), dbname='webarchive', uid='myuid')
>>> resource = monitor.get("http://www.myusefulpage.com/index.html")
>>> resource
<MonitoredResource(url='http://www.myusefulpage.com/index.html', uid='myuid') at 0xb7398accL>
>>> resource.check()
True
>>> # the resource has changed
```

Checking availability of the document on the URL

```
>>> from rrslib.web.changemonitor import HTTPDateTime
>>> resource = monitor.get("http://www.nonexistentpage.com")
>>> resource.available()
False
```

```
>>> resource = monitor.get("http://www.myusefulpage.com/index.html")
>>> resource.available(HTTPDateTime(2012, 6, 30, 15, 34))
True
```

Example of getting last available version of the document on the URL

```
>>> resource = monitor.get("http://www.myusefulpage.com/index.html")
>>> content = resource.get_last_version()
>>> print content.data
<html><head>
...
>>> resource = monitor.get("http://www.crazynonexistentpage.com")
>>> content = resource.get_last_version()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
DocumentNotAvailable: The content of this URL is not available and it
is not in the storage.
```

Example of getting version of the document in exact time

```
>>> resource = monitor.get("http://www.myusefulpage.com/index.html")
>>> content = resource.get_version(HTTPDateTime(2012, 6, 30, 15, 34))
```

Getting the last time when the document was checked:

```
>>> resource = monitor.get("http://www.crazynotexistentpage.com")
>>> resource.last_checked()
HTTPDateTime(Thu, 01 Jan 1970 00:00:00 GMT)
```

1.3.1 Methods

___init___(*self*, *url*, *uid*, *storage*)

x.___init___(...) initializes x; see help(type(x)) for signature

Parameters

`url:` monitored URL
(type=basestring (str or unicode))

`uid:` user identifier. This ID has to be unique for each one, who is using changemonitor.
(type=str)

`storage:` storage of monitored-resource data
(type=model.Storage)

Overrides: object.___init___

check(*self*)

Check the resource URL and load the most recent version into database.

TODO: consider using @lazy decorator. Most of use cases use this method so we have to insure that it will be called only once.

Return Value

True if the document has changed since last check.

get_last_version(*self*)

Get last available content of the document. If the document is available at this time, returns most recent version which is on the web server.

Return Value

Last available content of this resource.

(type=Content)

get_version(*self*, *time_or_version*)

Get content of this document in specified time or version. If the document was not available in given time, returns last available content. If there is no available content until given time, raises exception.

Parameters

`time_or_version`: Time or version of the content we want to retrieve. Version numbering is a convenience atop the GridFS API provided by MongoDB. version "-1" will be the most recently uploaded matching file, "-2" the second most recently uploaded, etc. Version "0" will be the first version uploaded, "1" the second version, etc. So if three versions have been uploaded, then version "0" is the same as version "-3", version "1" is the same as version "-2", and version "2" is the same as version "-1".

(type=HTTPDateTime or int)

get_diff(*self*, *start*, *end*)

Parameters

`start`: start time or version to be diffed

(type=HTTPDateTime or int)

`end`: end time or version to be diffed

(type=HTTPDateTime or int)

Return Value

either textual or binary diff of the file (if available). If contents are equal (document did not change within this time range) returns None.

(type=unicode)

available(*self*, *httptime*=`None`)

last_checked(*self*)

Get information about the time of last check of this resource.

Return Value

time of last check or None if the resource was never checked (or the HTTP requests timed out)

(type=HTTPDateTime or None)

___repr___(*self*)

str(x)

Overrides: object.___repr___ extit(inherited documentation)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

1.3.2 Properties

Name	Description
Inherited from object	
__class__	

1.4 Class Monitor

object
└ **changemonitor.Monitor**

Monitor is main class representing web change monitor. It serves as factory for creating MonitoredResource objects.

Usage:

```
>>> from rrslib.web.changemonitor import Monitor
>>> monitor = Monitor(user_id="rrs_university")
>>> resource = monitor.get("http://www.google.com")
>>> # if the page changed
>>> if resource.check():
>>>     print res.get_diff(start='last', end='now')
```

1.4.1 Methods

___init___(*self*, *user_id*, *db_host*=`'localhost'`, *db_port*=`27017`, *db_name*=`'webarchive'`, *http_proxy*=`None`)

Create a new monitor connected to MongoDB at *db_host:db_port* using database db_name.

Parameters

`user_id`: identification string of user/module who uses monitor. If user_id is given None, the monitor switches to 'global-view' mode and all requests to storage don't care about >>who checked this resource<<. On the other hand, if user_id is given a string, the monitor switches to 'user-view' mode and all operations are oriented to the user. Most of the reasonable use cases are using user_id, because a user/module almost everytime ask about >>what changed since I have been here for the last time<<, not >>what changed since somebody has been here for the last time<<...

(type=str or None)

`db_host`: (optional) hostname or IP address of the instance to connect to, or a mongodb URI, or a list of hostnames / mongodb URIs. If db_host' is an IPv6 literal it must be enclosed in '[' and ']' characters following the RFC2732 URL syntax (e.g. '[::1]' for localhost)

`db_port`: (optional) port number on which to connect

(type=int)

`db_name`: name of database which is used to store information about monitored documents and their versions.

(type=str)

`http_proxy`: (FUTURE USE) proxy server where to send requests

(type=unknown)

Overrides: object.___init___

get(*self*, *url*)

Creates new MonitoredResource instance which represents document on *url*.

Parameters

`url`: URL of monitored resource

(type=str)

Return Value

monitored resource object bound to URL *url*.

(type=MonitoredResource

Design pattern: factory method.)

allow_large_documents(*self*)

Allow large objects to be stored in the storage. Large document is defined as file larger than 4096KB. Tis constant is defined in this module named as LARGE_DOCUMENT_SIZE representing size of the file in kilobytes.

check_uid(*self*)

Check if user id given in constructor is a valid user id within the Monitor storage system. If the UID is occupied, returns False, True otherwise.

If user_id is None, an exception UidError is raised.

Return Value

True if the UID is free

(type=bool)

check_multi(*self*, *urls*=`[]`)

Check list of urls, start new thread for each one.

Parameters

`urls`: *(type=list)*

Return Value

list of MonitoredResource objects, each with actual data

(type=list<MonitoredResource>)

___repr___(*self*)

str(x)

Overrides: object.___repr___ extit(inherited documentation)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

1.4.2 Properties

Name	Description
Inherited from object	
___class___	

2 Module changemonitor._http

Maly privatni modul zajistujici HTTP pozadavky pro changemonitor. Obsahuje maly middleware, ktery slouzi k odstineni changemonitoru od nizsi vrstvy HTTP a ktery poskytne moznost pohodlnejsiho testovani nebo cachovani.

Date: $22.6.2012 13:01:57$

Author: Stanislav Heller

2.1 Variables

Name	Description
___modulename___	**Value:** `'_http'`
___email___	**Value:** `'xhelle03@stud.fit.vutbr.cz'`
___package___	**Value:** `'changemonitor'`

2.2 Class HTTPDateTime

object
└
changemonitor._http.HTTPDateTime

Datetime class for manipulating time within HTTP enviroment.

Usage: >>> h = HTTPDateTime() >>> h HTTPDateTime(Thu, 01 Jan 1970 00:00:00 GMT) >>> h.now() >>> h HTTPDateTime(Sat, 30 Jun 2012 16:09:43 GMT) >>> h.to_httpheader_format() 'Sat, 30 Jun 2012 16:09:43 GMT'

FIXME: solve the problem with GMT: >>> h = HTTPDateTime() >>> h.to_timestamp() -7200.0 # WTF?

2.2.1 Methods

___init___(*self*, *year*=`1970`, *month*=`1`, *day*=`1`, *hour*=`0`, *minute*=`0`, *second*=`0`, *microsecond*=`0`)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

to_httpheader_format(*self*)

Converts this object into date and time in HTTP-header format.

Return Value

date and time in HTTP format, i.e. 'Wed, 31 Aug 2011 16:45:03 GMT'.

(type=str)

from_httpheader_format(*self*, *timestr*)

Parse http header datetime format and save into this object.

Parameters

`timestr`: date and time in format which is used by HTTP protocol

(type=str)

Return Value

HTTPDateTime object equivalent to date and time of the timestr

(type=HTTPDateTime)

to_timestamp(*self*)

Convert into UNIX timestamp. (seconds since start of the UNIX epoch).

Return Value

unix timestamp

(type=float)

from_timestamp(*self*, *timestamp*)

Set date and time from timestamp.

Parameters

`timestamp`: time since start of the unix epoch

(type=float)

Return Value

HTTPDateTime object representing date and time of the timestamp

(type=HTTPDateTime)

to_datetime(*self*)

Convert the date and time from this object into python's datetime.datetime.

Return Value

datetime object equivalent to date and time of this object

(type=datetime.datetime)

from_datetime(*self*, *datetimeobj*)

from_gridfs_upload_date(*self*, *upload_date*)

Convert the date and time from grid_file.update_date string to HTTPDateTime object.

Parameters

`update_date:` time string from grid_file.update_date

Return Value

HTTPDateTime object representing date and time of update_date

(type=HTTPDateTime)

now(*self*)

Set the time of this object as current time (time.time())

Return Value

HTTPDateTime object representing current date and time.

(type=HTTPDateTime)

___repr___(*self*)

repr(x)

Overrides: object.___repr___ extit(inherited documentation)

___lt___(*self*, *other*)

___le___(*self*, *other*)

___eq___(*self*, *other*)

___ne___(*self*, *other*)

___gt___(*self*, *other*)

___ge___(*self*, *other*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

2.2.2 Properties

Name	Description
Inherited from object	
___class___	

3 Module changemonitor.diff

Date: $25.6.2012 12:12:44$

Author: Stanislav Heller

3.1 Variables

Name	Description
___modulename___	**Value:** `'diff'`
___email___	**Value:** `'xhelle03@stud.fit.vutbr.cz'`
___package___	**Value:** `'changemonitor'`

3.2 Class DocumentDiff

object
└ **changemonitor.diff.DocumentDiff**

Known Subclasses: changemonitor.diff.BinaryDiff, changemonitor.diff.HtmlDiff, changemonitor.diff.PlainTextDiff

Zakladni interface sjednocujici pristup k diffovani dokumentu.

Zdedena trida musi implementovat tridu diff, ktera se stara o diffnuti dvou dokumentu stejnych typu.

3.2.1 Methods

diff(*cls*, *obj1*, *obj2*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

3.2.2 Properties

Name	Description
Inherited from object	
___class___	

3.3 Class PlainTextDiff

object
changemonitor.diff.DocumentDiff
changemonitor.diff.PlainTextDiff

This class is a diff wrapper around classical gnu diff. Using this differ we can process every text documents (plaintext, html, css etc.)

3.3.1 Methods

diff(*cls*, *obj1*, *obj2*)

Parameters

`obj1`: first text to be diffed

(type=string or unicode)

`obj2`: second text to be diffed

(type=string or unicode)

Return Value

unicode diff

(type=unicode)

Overrides: changemonitor.diff.DocumentDiff.diff

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

3.3.2 Properties

Name	Description
Inherited from object	
___class___	

3.4 Class BinaryDiff

object
changemonitor.diff.DocumentDiff
changemonitor.diff.BinaryDiff

Tato trida bude diffovat binarni dokumenty - prevazne pdf, odt, doc atp.

3.4.1 Methods

diff(*cls*, *obj1*, *obj2*)

Parameters

`obj1`: first object to be diffed

(type=)

`obj2`: second object to be diffed

(type=)

Return Value

diff from xdelta and metainfo about diff

(type=dictionary {'diff': binaryDiff , 'metainfo': human-readable metainfo})

Overrides: changemonitor.diff.DocumentDiff.diff

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

3.4.2 Properties

Name	Description
Inherited from object	
___class___	

3.5 Class HtmlDiff

object

changemonitor.diff.DocumentDiff

changemonitor.diff.HtmlDiff

Html diff, which shows pieces of code, which was added to the page. Uses output of GNU diff and its interface PlainTextDiff.

Returns generator object HtmlDiffChunk Usage: >>> # r is resource >>> d = r.get_diff(-2,-1) #get diff of last two versions >>> print d.next() HtmlDiffChunk(position=u'line_info_from_diff', removed=u'this was removed',added=u'this was added')

3.5.1 Methods

htmldiff(*cls*, *raw_diff*)

diff(*cls*, *obj1*, *obj2*)

Overrides: changemonitor.diff.DocumentDiff.diff

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

3.5.2 Properties

Name	Description
Inherited from object	
___class___	

4 Module changemonitor.errors

Exceptions raised by rrslib.web.changemonitor package

Date: $22.6.2012 13:01:57$

Author: Stanislav Heller

4.1 Variables

Name	Description
___modulename___	**Value:** `'errors'`
___email___	**Value:** `'xhelle03@stud.fit.vutbr.cz'`
___package___	**Value:** `None`

4.2 Class ChangeMonitorError

Known Subclasses: changemonitor.errors.DocumentHistoryNotAvaliable, changemonitor.errors.Docume changemonitor.errors.DocumentTooLarge, changemonitor.errors.NotSupportedYet, changemonitor.errors.UidError

Base error class for all exceptions within this package.

4.2.1 Methods

Inherited from exceptions.Exception

___init___(), ___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___str___(), ___unicode___()

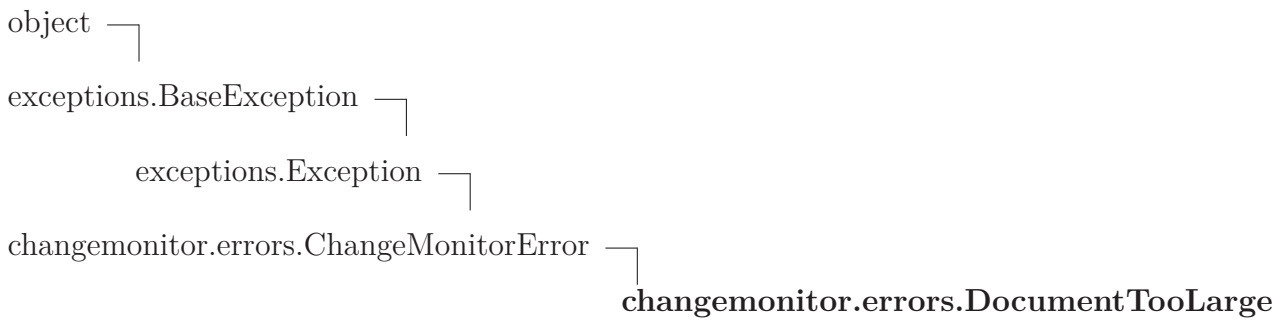
Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

4.2.2 Properties

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

4.3 Class DocumentTooLarge



Raised when monitored document's size exceeds the LARGE_DOCUMENT_SIZE constant.

4.3.1 Methods

Inherited from exceptions.Exception

___init___(), ___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___str___(), ___unicode___()

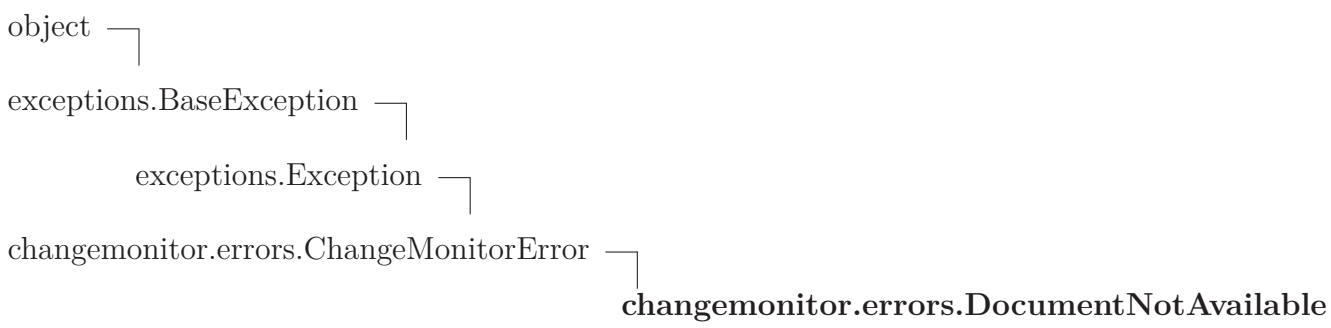
Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

4.3.2 Properties

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

4.4 Class DocumentNotAvailable



Raised when document is not available on the URL or on the storage.

4.4.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

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

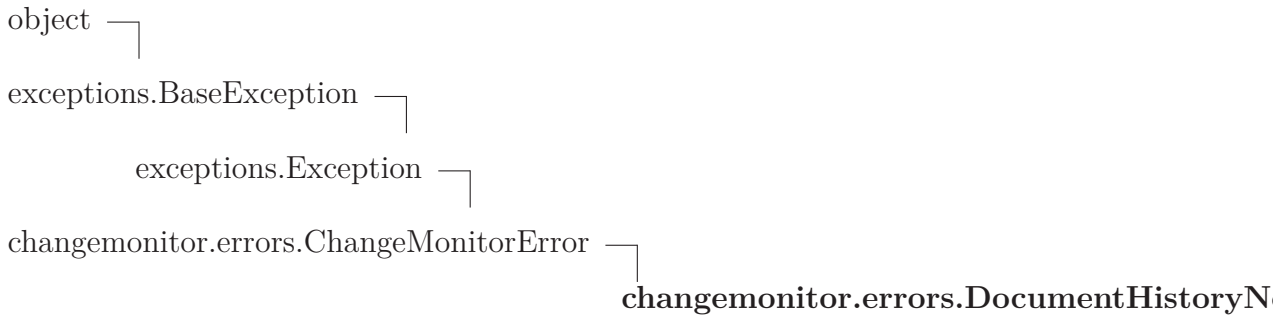
Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

4.4.2 Properties

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

4.5 Class DocumentHistoryNotAvaliable



Raised when trying to get version or diff of document, which version history is not stored in the storage.

4.5.1 Methods

Inherited from exceptions.Exception

___init___(), ___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___str___(), ___unicode___()

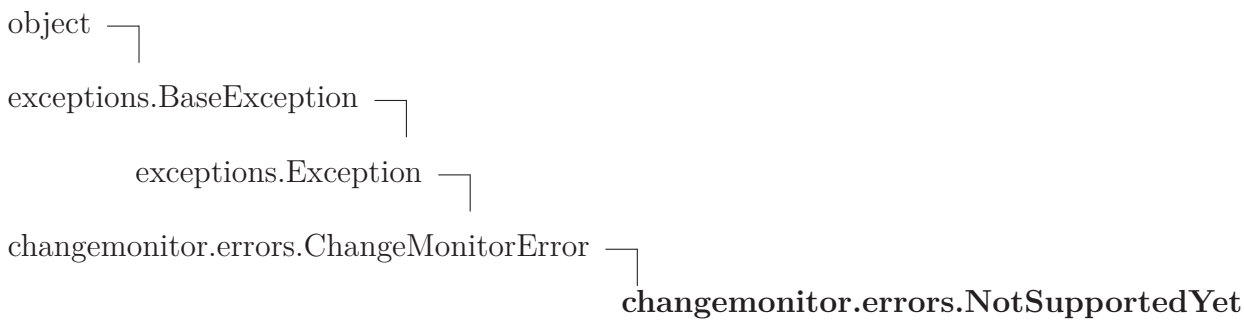
Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

4.5.2 Properties

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

4.6 Class NotSupportedYet



Raised when the method/class/function is not supported in this implementation.

4.6.1 Methods

Inherited from exceptions.Exception

___init___(), ___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___str___(), ___unicode___()

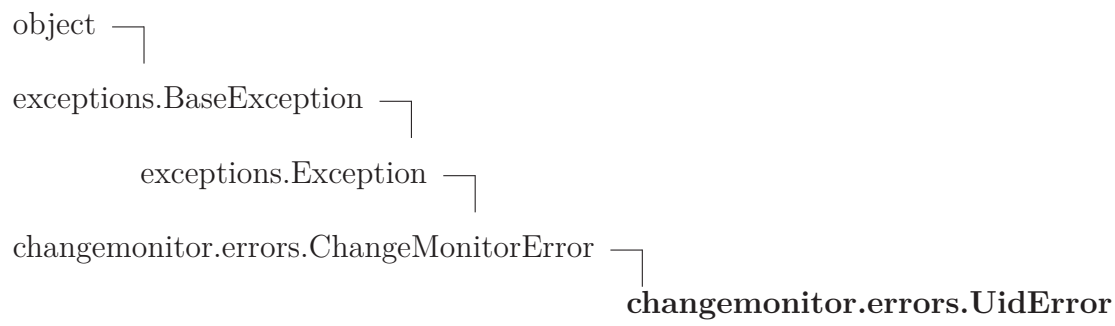
Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

4.6.2 Properties

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

4.7 Class UidError



Raised when some error connected with user id occured.

4.7.1 Methods

Inherited from exceptions.Exception

___init___(), ___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___str___(), ___unicode___()

Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

4.7.2 Properties

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

5 Module changemonitor.model

Model for changemonitor – interface for accessing the data

This module creates abstraction layer between data storage implementation (SQL/NoSQL database/filesystem).

Date: $23.6.2012 16:33:31$

Author: Stanislav Heller

5.1 Variables

Name	Description
___modulename___	**Value:** `'model'`
___email___	**Value:** `'xhelle03@stud.fit.vutbr.cz'`
___package___	**Value:** `'changemonitor'`

5.2 Class BaseMongoModel

object
└ **changemonitor.model.BaseMongoModel**

Known Subclasses: changemonitor.model.HttpHeaderMeta, changemonitor.model.Storage

Serves as base class, which is inherited by every model class.

5.2.1 Methods

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

5.2.2 Properties

Name	Description
Inherited from object	
___class___	

5.3 Class Storage

object

changemonitor.model.BaseMongoModel

changemonitor.model.Storage

Abstraction of the storage. The purpose of this class is to create abstraction layer, which provides database-independent API for manipulation in the filesystem. The only requirement on the filesystem is that it has to support file versioning (or some workaround which implements versioning within the fs which does not support versioning natively).

The implementation is nowadays built on MongoDB.

Usage:

```
>>> from pymongo import Connection
>>> from model import Storage
>>> store = Storage(Connection(), "myuid", "webarchive")
>>> file = store.get("http://www.myfancypage.com/index.html")
>>> # get last version of the file, which is available in the storage
>>> c = file.get_last_content()
>>> # get the raw data
>>> c.data
"<html>
...
>>> # content type and content length
>>> print c.content_type, c.length
'text/html' 29481
```

Design pattern: Factory

5.3.1 Methods

___init___(*self*, *connection*, *uid*, *database='webarchive'*)

Initializes storage.

Parameters

`connection:` database connection
(type=pymongo.Connection)

`uid:` user id (see Monitor.___doc___ for more info)
(type=str)

`database:` if the storage is based on database, this param represents the name of database to be used within this instance.
(type=str)

Overrides: object.___init___

allow_large_documents(*self*)

Allow large objects to be stored in the storage.

get(*self*, *filename*)

Get file object by filename.

Parameters

`filename:` name of the file. In this case, it will be URL.

`filaname:` *(type=str)*

Return Value

File object representing file in many versions
(type=File)

check_uid(*self*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

5.3.2 Properties

Name	Description
Inherited from object	
__class__	

5.4 Class File

changemonitor.model.File

```
One file in filesystem. A file can contain more contents in various
versions. Main purpose of this class is to get rid of GridFS
and GridOut instances and replace it with file-like wrapper.

MongoDB record:
content = {
    filename: URL
    md5: str
    sha1: str
    content_type: str
    length: int
    urls = []
}

Design pattern: Active Record
```

5.4.1 Methods

___init___(*self*, *filename*, *fs*, *headermeta*)

Create new file instance.

Parameters

`filename:` name of the file

(type=basestring (str or unicode))

`fs:` filesystem object

(type=GridFS)

`headermeta:` http header metadata

(type=HttpHeaderMeta

WARNING: Application developers should generally not need to instantiate this class directly. The only correct way how to get this object is using Storage.get() method.)

Overrides: object.___init___

purge_cache(*self*)

Cleans the whole content cache.

refresh_cache(*self*)

Refreshes part of cache, which can potentionally change (version pointers). This is very useful if we expect the File object to live during more than one check() call. If so, the information about version has to be updated in the cache (version -1 becomes -2 etc.).

This method should be called after every check() call!

get_version(*self*, *timestamp_or_version*)

Get content of the file in specific version. Version can be specified by version number (convenience atop the GridFS API by MongoDB) or unix timestamp.

Parameters

`timestamp_or_version`: version or timestamp of version which we want to retrieve.

(type=int)

Return Value

content of the file in specified time/version

(type=Content)

get_last_version(*self*)

Loads the last version of the file which is available on the storage. If the monitor is in user-view mode, loads last version, which was checked by specified user.

Return Value

most recent content of the file which is on the storage.

(type=Content)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

5.4.2 Properties

Name	Description
Inherited from object	
___class___	

5.5 Class Diffable

object
└ **changemonitor.model.Diffable**

Known Subclasses: changemonitor.model.Content

Interface-like class. A class, which inherites this interface, has to implement the method for

diffing two objects and choose of a diff algorithm.

5.5.1 Methods

diff_to(*self*, *obj*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

5.5.2 Properties

Name	Description
Inherited from object	
___class___	

5.6 Class Content

object

changemonitor.model.Diffable

changemonitor.model.Content

Content of web document in one version.

Implements Diffable interface to get possibility to diff contents to each other. Differ algorithm is choosen automatically.

Implementation detail: wrapper of GridOut instance.

5.6.1 Methods

___init___(*self*, *gridout*)

```
Create new instance of content.

WARNING: Do not instantiate this class by yourself, this is done by
         File methods.
@param gridout: gridout instance which was retrieved by GridFS.
@type  gridout: gridfs.grid_file.GridOut
```

Overrides: object.___init___

___getattr___(*self*, *name*)

diff_to(*self*, *other*)

Creates diff of self and given Content object and returns unicode string representing the computed diff: $ diff-algo self obj

Parameters

`other`: diffed content

(type=Content)

Return Value

computed diff

(type=unicode)

Overrides: changemonitor.model.Diffable.diff_to

___repr___(*self*)

str(x)

Overrides: object.___repr___ extit(inherited documentation)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

5.6.2 Properties

Name	Description
Inherited from object	
___class___	

5.7 Class HttpHeaderMeta

object

changemonitor.model.BaseMongoModel

changemonitor.model.HttpHeaderMeta

```
Model for HTTP header metadata.

header = {
  timestamp: 1341161610.287
  response_code: 200
  last_modified: cosi
  etag: P34lkdfk32jrlkjdfpoqi3
  uid: "rrs_university"
  url+index: "http://www.cosi.cz"
  content: object_id
}
```

5.7.1 Methods

___init___(*self*, *connection*, *uid*, *database*)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

get_by_time(*self*, *url*, *timestamp*, *last_available*=`False`)

Parameters

`url:` url of resource to search for
(type=string)

`timestamp:` *(type=int)*

`last_available:` *(type=Bool)*

Return Value

http header metadata of 'url'/None if not found
(type=)

To Do: docstring Get record of 'url' with 'timestamp' from HeaderMeta database

get_by_version(*self*, *url*, *version*, *last_available*=`False`)

Parameters

`url:` url of resource to get from db
(type=string)

`version:` version number of record, ...TODO: x,-x,1,0,-1
(type=int)

`last_available:` *(type=bool)*

Return Value

http header metadata of 'url'/None if not found
(type=)

To Do: docstring Get 'version' of 'url' from HeaderMeta database

save_header(*self*, *url*, *response_code*, *fields*, *content_id*)

Save http header into HttpHeaderMeta database

Parameters

`url:` url of checked resource

`response_code:` response code of web server

`fields:` fields of http response

`content_id:` content-id field of http response

Return Value

saved object

last_checked(*self*, *url*)

```
Get time when 'url' was last checked
WARNING!
  if None is returned, then 'url' was never checked
  that should never happen, as 'url' is always checked
  in constructor of MonitoredResource
  but it's possible that the header is not saved
  because of an error, eg. timeout or other
@param url: url of resource checked
@type url: string
@returns: time of last check
@rtype: HTTPDateTime
```

check_uid(*self*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

5.7.2 Properties

Name	Description
Inherited from object	
___class___	

6 Module changemonitor.resolver

Date: $25.6.2012 12:12:44$

Author: Stanislav Heller

6.1 Variables

Name	Description
___modulename___	**Value:** `'resolver'`
___email___	**Value:** `'xhelle03@stud.fit.vutbr.cz'`
___package___	**Value:** `'changemonitor'`

6.2 Class Resolver

object
└ **changemonitor.resolver.Resolver**

System pro zajisteni funkcionality "zmenilo se neco na dane URL?". Zde se bude osetrovat mnoho vyjimek s http hlavickami, s md5, content-length atp.

Implikace: changed(HTTP:content-length) -> content changed changed(HTTP:md5) -> content changed changed(HTTP:response code) -> it depends.. changed(HTTP:last-modified) -> doesn't matter

6.2.1 Methods

___init___(*self*, *storage*, *timeout*=`10`)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

resolve(*self*, *url*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

6.2.2 Properties

Name	Description
Inherited from object	
___class___	

6.3 Class Rule

object
└
changemonitor.resolver.Rule

Pravidlo pro resolver. Mozna se bude hodit nejake takove rozdeleni tech pravidel... proste rozdel a panuj.

6.3.1 Methods

___call___(*self*)

Pokud ___call___ vraci true, pak rule matchl.

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

6.3.2 Properties

Name	Description
Inherited from object	
___class___	

Index