
Capitains.Nautilus Documentation

Release 0.0.1

Thibault Clérice

Jul 10, 2018

Contents

1	Capitains Nautilus	1
1.1	Documentation	1
1.2	Trying Nautilus with a test dataset example	1
1.3	Running Nautilus from the command line	2
1.4	Source for the tests	2
1.5	Contents	2
1.6	Indices and tables	6
	Python Module Index	7

CHAPTER 1

Capitains Nautilus

1.1 Documentation

CapiTainS Nautilus provides a Flask extension to build upon MyCapytain resolver. The finale goal of the application, built upon [MyCapytain](#), is to serve either as a Web-API provider (Currently supporting CTS, partly DTS. OAI-PMH and a Sparql endpoint are scheduled.) These API can be used to access portion of or complete texts using standards. Metadata are exposed as well.

A second goal of Nautilus is to serve as a cache wrapper for resolver, in order to speed up serving texts for user interfaces such as [Nemo](#) .

A known implementation can be found at [the University of Leipzig](#) . You can find the set-up files on [Github](#)

1.2 Trying Nautilus with a test dataset example

With Python 3 only !

```
git clone https://github.com/Capitains/Nautilus.git
virtualenv -p /usr/bin/python3 venv
source venv/bin/activate
python app.py
```

Now go to <http://localhost:5000> and check out <http://localhost:5000/api/cts> , <http://localhost:5000/api/dts/collections>, <http://localhost:5000/api/cts?request=GetValidReff>

1.3 Running Nautilus from the command line

This small tutorial takes that you have one or more Capitains formatted repositories (such as <http://github.com/PerseusDL/canonical-latinLit>) in the folders /home/USERNAME/repository1 where USERNAME is your user session name.

1. (Advised) Create a virtual environment and source it : `virtualenv -p /usr/bin/python3 env`,
`source env/bin/activate`
2. **With development version:**
 - Clone the repository : `git clone https://github.com/Capitains/Nautilus.git`
 - Go to the directory : `cd Nautilus`
 - Install the source with develop option : `python setup.py develop`
2. **With production version (not available for now):**
 - Install from pip : `pip install capitains-nautilus`
3. You will be able now to call `capitains nautilus help` information through `capitains-nautilus --help`
4. Basic setting for testing a directory is `capitains-nautilus --debug /home/USERNAME/repository1`. This can take more than one repository at the end such as `capitains-nautilus --debug /home/USERNAME/repository1 /home/USERNAME/repository2`. You can force host and port through `-host` and `-port` parameters.

1.4 Source for the tests

Textual resources and inventories are owned by Perseus under CC-BY Licences. See <https://github.com/PerseusDL/canonical-latinLit> and <https://github.com/PerseusDL/canonical-farsiLit>

1.5 Contents

1.5.1 CapiTainS Nautilus API Documentation

Resolvers

Resolver provides a system to retrieve a text file and an inventory from local resources for example.

CapiTainS formatted repository

```
class capitains_nautilus.cts.resolver.NautilusCTSResolver(resource, name=None,
                                                         logger=None,
                                                         cache=None,      dis-
                                                         patcher=None)
```

XML Folder Based resolver.

Parameters

- **resource** (*[str]*) – Resource should be a list of folders retaining data as Capitains Guidelines Repositories
- **name** – Key used to make cache key

- **cache** (*BaseCache*) – Cache object to be used for the inventory
- **logger** (*logging.logger*) – Logging object

Variables

- **inventory_cache_key** – Werkzeug Cache key to get or set cache for the TextInventory
- **texts_cache_key** – Werkzeug Cache key to get or set cache for lists of metadata texts objects
- **texts_parsed** – Werkzeug Cache key to get or set cache for lists of parsed texts objects
- **texts** – List of Text Metadata objects
- **source** – Original resource parameter

Warning: This resolver does not support inventories

Errors

exception capitains_nautilus.errors.CTSError

Bases: *capitains_nautilus.errors.NautilusError*

CODE = None

exception capitains_nautilus.errors.InvalidContext

Bases: *capitains_nautilus.errors.CTSError*

Invalid value for context parameter in GetPassage or GetPassagePlus request

CODE = 5

exception capitains_nautilus.errors.InvalidLevel

Bases: *capitains_nautilus.errors.CTSError*

Invalid value for level parameter in GetValidReff request

CODE = 4

exception capitains_nautilus.errors.InvalidURN

Bases: *capitains_nautilus.errors.CTSError*, *MyCapytain.errors.InvalidURN*

Syntactically valid URN refers in invalid value

CODE = 3

exception capitains_nautilus.errors.InvalidURNSyntax

Bases: *capitains_nautilus.errors.CTSError*

Invalid URN syntax

CODE = 2

exception capitains_nautilus.errors.MissingParameter

Bases: *capitains_nautilus.errors.CTSError*

Request missing one or more required parameters

CODE = 1

exception capitains_nautilus.errors.NautilusError

Bases: `BaseException`

An error has occurrence

CODE = `None`

exception capitains_nautilus.errors.UndispatchedTextError

Bases: `capitains_nautilus.errors.CTSError`, `MyCapytain.errors.UndispatchedTextError`

A Text has not been dispatched

CODE = `7`

exception capitains_nautilus.errors.UnknownCollection

Bases: `MyCapytain.errors.UnknownCollection`, `capitains_nautilus.errors.CTSError`

Resource requested is not found

CODE = `6`

Flask Extension

```
class capitains_nautilus.flask_ext.FlaskNautilus (prefix="", app=None,
                                                name=None, resolver=None,
                                                flask_caching=None, access_Control_Allow_Origin=None,
                                                access_Control_Allow_Methods=None,
                                                logger=None)
```

Bases: `object`

HTTP API Interfaces for MyCapytains resolvers

Parameters

- **prefix** – Prefix on which to install the extension
- **app** – Application on which to register
- **name** – Name to use for the blueprint
- **resolver** (*Resolver*) – Resolver
- **flask_caching** (*Cache*) – HTTP Cache should be a FlaskCaching Cache object
- **logger** (*logging.Logger*) – Logging handler.

Variables

- **access_Control_Allow_Methods** – Dictionary with route name and allowed methods over CORS
- **access_Control_Allow_Origin** – Dictionary with route name and allowed host over CORS or “*”
- **ROUTES** – List of triple length tuples
- **Access_Control_Allow_Methods** – Dictionary with route name and allowed methods over CORS
- **Access_Control_Allow_Origin** – Dictionary with route name and allowed host over CORS or “*”

- **LoggingHandler** – Logging handler to be set for the blueprint
- **logger** – Logging handler
- **resolver** – CapiTainS resolver

Access_Control-Allow-Methods = {'r_cts': 'OPTIONS, GET', 'r_dts_collection': 'OPTION

Access_Control-Allow-Origin = '*'

CACHED = ['_r_GetCapabilities', '_r_GetPassage', '_r_GetPassagePlus', '_r_GetValidReff

LoggingHandler

alias of `logging.StreamHandler`

ROUTES = [('/cts', 'r_cts', ['GET']), ('/dts/collections', 'r_dts_collection', ['GET',

cts_error (*error_name*, *message=None*)

Create a CTS Error reply

Parameters

- **error_name** – Name of the error
- **message** – Message of the Error

Returns CTS Error Response with information (XML)

dts_error (*error_name*, *message=None*)

Create a DTS Error reply

Parameters

- **error_name** – Name of the error
- **message** – Message of the Error

Returns DTS Error Response with information (JSON)

flaskcache

init_app (*app*)

Initiate the extension on the application

Parameters **app** – Flask Application

Returns Blueprint for Flask Nautilus registered in app

Return type Blueprint

init_blueprint ()

Properly generates the blueprint, registering routes and filters and connecting the app and the blueprint

Returns Blueprint of the extension

Return type Blueprint

r_cts ()

Actual main route of CTS APIs. Transfer typical requests through the ?request=REQUESTNAME route

Returns Response

r_dts_collection (*objectId=None*)

DTS Collection Metadata reply for given objectId

Parameters **objectId** – Collection Identifier

Returns JSON Format of DTS Collection

r_dts_collections (*objectId*)

DTS Collection Metadata reply for given objectId

Parameters **objectId** – Collection Identifier

Returns JSON Format of DTS Collection

setLogger (*logger*)

Set up the Logger for the application

Parameters **logger** – logging.Logger object

Returns Logger instance

view (*function_name*)

Builds response according to a function name

Parameters **function_name** – Route name / function name

Returns Function

Command-line Interface

`capitains_nautilus.cmd.cmd()`

Cache Manager

`capitains_nautilus.manager.FlaskNautilusManager` (*resolver, flask_nautilus*)

Provides a manager for flask scripts to perform specific maintenance operations

Parameters

- **resolver** (`NautilusCTSResolver`) – Nautilus Extension Instance
- **flask_nautilus** (`FlaskNautilus`) – Flask Application

Returns CLI

Return type `click.group`

Import with

`capitains_nautilus.manager.read_levels` (*text*)

Read text and get there reffs

Parameters **text** – Collection (Readable)

Returns

1.6 Indices and tables

- [Importing Modules](#)
- [genindex](#)
- [modindex](#)
- [search](#)

C

`capitains_nautilus.cmd`, 6
`capitains_nautilus.errors`, 3
`capitains_nautilus.flask_ext`, 4
`capitains_nautilus.manager`, 6

A

Access_Control-Allow-Methods (capitains_nautilus.flask_ext.FlaskNautilus attribute), 5

Access_Control-Allow-Origin (capitains_nautilus.flask_ext.FlaskNautilus attribute), 5

C

CACHED (capitains_nautilus.flask_ext.FlaskNautilus attribute), 5

capitains_nautilus.cmd (module), 6

capitains_nautilus.errors (module), 3

capitains_nautilus.flask_ext (module), 4

capitains_nautilus.manager (module), 6

cmd() (in module capitains_nautilus.cmd), 6

CODE (capitains_nautilus.errors.CTSError attribute), 3

CODE (capitains_nautilus.errors.InvalidContext attribute), 3

CODE (capitains_nautilus.errors.InvalidLevel attribute), 3

CODE (capitains_nautilus.errors.InvalidURN attribute), 3

CODE (capitains_nautilus.errors.InvalidURNSyntax attribute), 3

CODE (capitains_nautilus.errors.MissingParameter attribute), 3

CODE (capitains_nautilus.errors.NautilusError attribute), 4

CODE (capitains_nautilus.errors.UndispatchedTextError attribute), 4

CODE (capitains_nautilus.errors.UnknownCollection attribute), 4

cts_error() (capitains_nautilus.flask_ext.FlaskNautilus method), 5

CTSError, 3

D

dts_error() (capitains_nautilus.flask_ext.FlaskNautilus method), 5

F

flaskcache (capitains_nautilus.flask_ext.FlaskNautilus attribute), 5

FlaskNautilus (class in capitains_nautilus.flask_ext), 4

FlaskNautilusManager() (in module capitains_nautilus.manager), 6

I

init_app() (capitains_nautilus.flask_ext.FlaskNautilus method), 5

init_blueprint() (capitains_nautilus.flask_ext.FlaskNautilus method), 5

InvalidContext, 3

InvalidLevel, 3

InvalidURN, 3

InvalidURNSyntax, 3

L

LoggingHandler (capitains_nautilus.flask_ext.FlaskNautilus attribute), 5

M

MissingParameter, 3

N

NautilusCTSResolver (class in capitains_nautilus.cts.resolver), 2

NautilusError, 3

R

r_cts() (capitains_nautilus.flask_ext.FlaskNautilus method), 5

r_dts_collection() (capitains_nautilus.flask_ext.FlaskNautilus method), 5

r_dts_collections() (capitains_nautilus.flask_ext.FlaskNautilus method), 5

read_levels() (in module capitains_nautilus.manager), 6

ROUTES (capitains_nautilus.flask_ext.FlaskNautilus attribute), [5](#)

S

setLogger() (capitains_nautilus.flask_ext.FlaskNautilus method), [6](#)

U

UndispatchedTextError, [4](#)

UnknownCollection, [4](#)

V

view() (capitains_nautilus.flask_ext.FlaskNautilus method), [6](#)