HucitLib

Matteo Romanello

Nov 16, 2021

CONTENTS:

hucitlib is a knowledge base about classical (Greek and Latin) texts, as well as a Python library to query and modify its contents.

The main goal of hucitlib is to support the automatic extraction of bibliographic references to primary sources in the domain of Classics. The hucitlib knowledge base contains:

- names (and abbreviations) of ancient authors;
- titles (and abbreviations) of ancient works;
- resolvable URIs and unique identifiers (CTS URNs) for authors, works and citable passages;
- links to external resources (Perseus Catalog, Wikidata, Wikipedia);
- information about the canonical citation structure of ancient works.

hucitlib relies on SuRF , a Python Object RDF Mapper library, so as to In order to make the knowledge base as much as possible easy to use programmatically (read more here).

If you are using hucitlib as part of your research, please cite the following paper:

```
  @inproceedings{DBLP:conf/semweb/RomanelloP17,
  author    = {Matteo Romanello and
               Michele Pasin},
  editor    = {Alessandro Adamou and
               Enrico Daga and
               Leif Isaksen},
  title     = {Using Linked Open Data to Bootstrap a Knowledge Base of Classical
               Texts},
  booktitle = {Proceedings of the Second Workshop on Humanities in the Semantic Web
               (WHiSe {II)} co-located with 16th International Semantic Web Conference
               {(ISWC} 2017), Vienna, Austria, October 22, 2017},
  series    = {{CEUR} Workshop Proceedings},
  volume    = {2014},
  pages     = {3--14},
  publisher = {CEUR-WS.org},
  year      = {2017},
  url       = {http://ceur-ws.org/Vol-2014/paper-01.pdf},
  timestamp = {Wed, 12 Feb 2020 16:44:52 +0100},
  biburl    = {https://dblp.org/rec/conf/semweb/RomanelloP17.bib},
  bibsource = {dblp computer science bibliography, https://dblp.org}
}
```

CHAPTER

ONE

SETUP

1.1 Installation

```
pip install hucitlib
```

1.2 Default triple store

By default, when initialising a *hucitlib.KnowledgeBase* instance, RDF data are read from a read-only public triple store, which runs on the CLARIAH Druid infrastructure:

```
>>> from hucitlib import KnowledgeBase
>>> kb = KnowledgeBase()
>>> kb.settings
{
  'reader': 'sparql_protocol',
  'writer': 'sparql_protocol',
  'endpoint': 'https://api.druid.datalegend.net/datasets/mromanello/hucit/services/hucit/
↪sparql',
  'default_context': 'https://druid.datalegend.net/mromanello/hucit/graphs/default'
}
```

Note: When connecting to the default triple store, all methods that modify entries in the knowledge base (e.g. *hucitlib.surfext.HucitAuthor.set_urn()*) won't work!

1.3 Connecting to a local triple store

The RDF data that power `hucitlib` can be stored in any triple store that supports the SPARQL 1.1 API. `hucitlib` comes with scripts to install and load/clear/dump data from a Virtuoso triples store.

If you prefer to use another triple store, after having it set up and loaded the data into it, just create a new configuration file

```
# content of virtuoso_local.ini
[surf]
reader=sparql_protocol
```

(continues on next page)

(continued from previous page)

```
writer=sparql_protocol
server=localhost
endpoint=http://localhost:8890/sparql
port=8890
default_context=http://purl.org/hucit/kb
```

and pass the path to this file when initialising the knowledge base

```
>>> from hucitlib import KnowledgeBase
>>> kb = KnowledgeBase('virtuoso_local.ini')
>>> kb.settings
{
  'reader': 'sparql_protocol',
  'writer': 'sparql_protocol',
  'server': 'localhost',
  'endpoint': 'http://localhost:8890/sparql',
  'port': 8890,
  'default_context': 'http://purl.org/hucit/kb'
}
```

CHAPTER

TWO

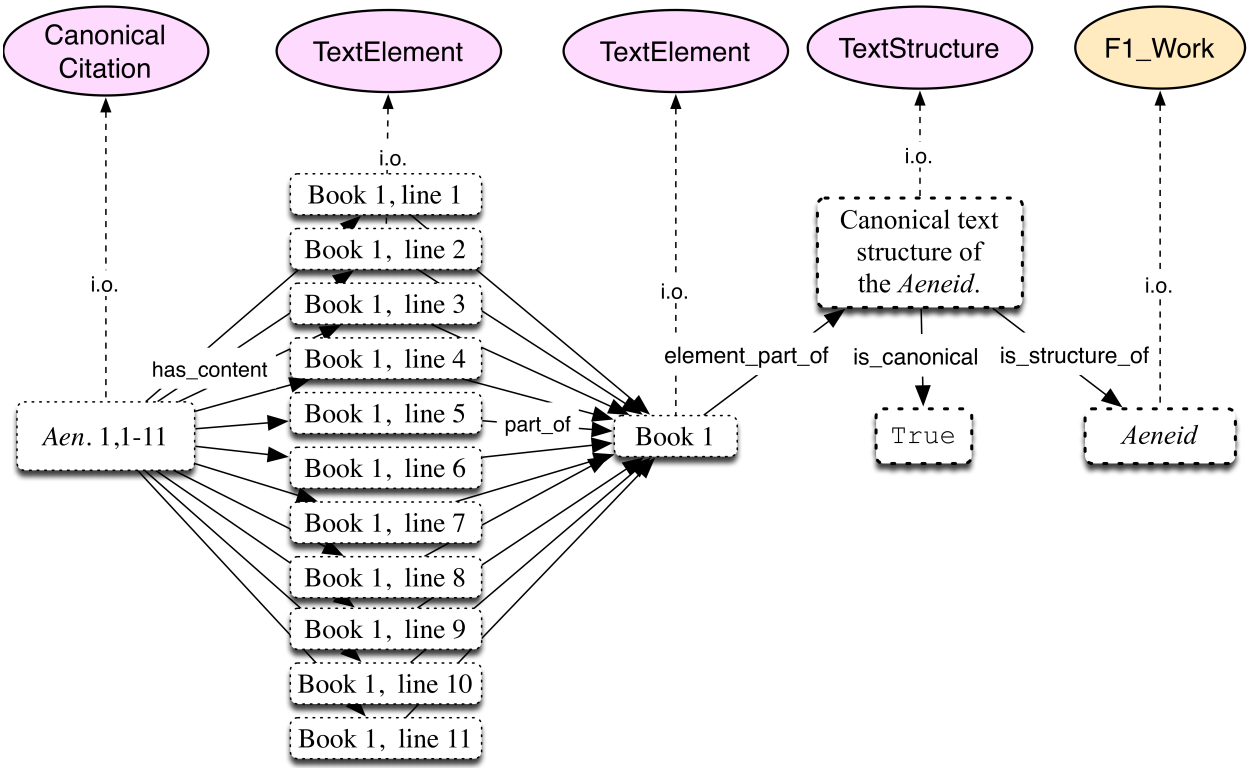
DATA MODEL

The data model of `hucitlib`'s knowledge base is based on the following ontologies:

1. CIDOC-CRM
2. FRBRoo
3. HuCit

In fact, the rationale for developing this data model was to re-use as much as possible already existing and widely adopted ontologies, and to extend them by means of new classes and properties only when absolutely necessary.

The first two ontologies that form the backbone of the HuCit knowledge base are CIDOC-CRM and FRBRoo. The CIDOC-CRM is a conceptual model that was born as a metadata standard for the archive and museum world, and proved to be suitable to represent information in many different domains. The subset of CIDOC-CRM classes and properties used by the knowledge base is limited to those that represent things like names, titles, and abbreviations for ancient authors and works. It is worth noting, however, that we try as much as possible to harmonise our use of CIDOC-CRM with the adoption of other essential standards, like the CTS protocol, that exist outside of the CRM world. For instance, we make extensive use of CTS URNs, which are declared as instances of CIDOC-CRM's `E42 Identifier` having a specific `E55 Type`.



The third and last ontology involved is the Humanities Citation Ontology (HuCit). This ontology was developed as a lightweight extension of CIDOC-CRM and FRBRoo aimed specifically at formalising the canonical text structures that are used to cite classical texts. This ontology allows us to instantiate any single citable unit of a canonical text (e.g. all lines in all books of Homer's Iliad), an ability of essential importance when representing canonical citations.

CHAPTER

THREE

COMMAND LINE INTERFACE

HucitLib comes with a command-line interface to query and modify the knowledge base's contents.

```
$ hucit --help

RDFLib Version: 5.0.0
Command line interface for a HuCit knowledge base.

Usage:
    knowledge_base/cli.py find <search_string> [--config-file=<path>]
    knowledge_base/cli.py add (name|abbr|title|sameas) --to=<cts_urn> <string_to_add> [--
↪config-file=<path>]
    knowledge_base/cli.py (-h | --help)

Options:
    --to=<cts_urn> CTS URN of the author/work to edit.
    --config-file=<path> Path to the configuration file (overwrites default␣
↪configuration).
```

3.1 Search

```
$ hucit find Homer

Searching for "homer" yielded 2 results

1) urn:cts:cwkb:498 Homerus Latinus                                    (Matched: 'Homer.
↪')


2) urn:cts:greekLit:tlg0012 Homer                                      (Matched: 'Homer')
```

Wildcard search works as well:

```
$ hucit find Homer*

...
```

3.2 Display

```
$ hucit find urn:cts:greekLit:tlg0011

Sophokles :: urn:cts:greekLit:tlg0011 (http://purl.org/hucit/kb/authors/1090)

7 works by this author:
 - Sophokles, Ajax :: urn:cts:greekLit:tlg0011.tlg003 ␣
↪(http://purl.org/hucit/kb/works/3896)
 - Sophokles, Antigone :: urn:cts:greekLit:tlg0011.tlg002 ␣
↪(http://purl.org/hucit/kb/works/3897)
 - Sophokles, Electra :: urn:cts:greekLit:tlg0011.tlg005 ␣
↪(http://purl.org/hucit/kb/works/3898)
 - Sophokles, Oedipus at Kolonos :: urn:cts:greekLit:tlg0011.tlg007 ␣
↪(http://purl.org/hucit/kb/works/3899)
 - Sophokles, King Oedipus :: urn:cts:greekLit:tlg0011.tlg004 ␣
↪(http://purl.org/hucit/kb/works/3900)
 - Sophokles, Philoctetes :: urn:cts:greekLit:tlg0011.tlg006 ␣
↪(http://purl.org/hucit/kb/works/3901)
 - Sophokles, The Women of Trachis :: urn:cts:greekLit:tlg0011.tlg001 ␣
↪(http://purl.org/hucit/kb/works/3902)


Related resources:
 - http://cwkb.org/author/id/1090/turtle
 - http://data.perseus.org/catalog/urn:cts:greekLit:tlg0011
 - http://viaf.org/viaf/101760867
 - http://www.wikidata.org/wiki/Special:EntityData/Q7235
```

3.3 Edit

Note: Editing via CLI is not yet fully implemented/tested.

3.4 Changing configuration

CHAPTER

FOUR

KNOWLEDGE BASE

The class *KnowledgeBase* is the main access point to all resources described in the knowledge base (e.g. *HucitAuthor*, *HucitWork*, etc.). Its methods can be divided into the following high-level groups:

- methods that concern globally the knowledge base:
  - *search()*
  - *to_json()*
  - *get_statistics()*
- methods to access top-level resources:
  - *get_resource_by_urn()*
  - *get_authors()*
  - *get_author_label()*
  - *get_works()*
  - *get_work_label()*
  - *get_opus_maximum_of()*
  - *get_textelement_type()*
  - *get_textelement_types()*
- "factory methods", i.e. methods that create new objects (i.e. entries):
  - *create_cts_urn()*
  - *create_text_element()*
  - *add_textelement_type()*
  - *add_textelement_types()*

class hucitlib.**KnowledgeBase**(*config_file: Optional[str] = None*)

`KnowledgeBase` is a class that allows for accessing a HuCit knowledge base in an object-oriented fashion. The abstraction layer it provides means that you can use, search and modify its content without having to worry about the underlying modelling of data in RDF.

Parameters **config_file** (*str*) – Path to the configuration file containing the parameters to connect to the triple store whose data will be accessible via the `KnowledgeBase` object.

Returns Description of returned object.

Return type None

Note: By default (i.e. when no configuration file is specified) a new `KnowledgeBase` instance will be created that reads data directly from the triple store hosted at Druid. **NB**: please note that all methods that *modify* entries in the KB won't work as that triple store is *read-only*.

Example of usage:

```
>>> from hucit_kb import KnowledgeBase
>>> kb = KnowledgeBase()
>>> homer = kb.get_resource_by_urn('urn:cts:greekLit:tlg0012')
>>> print(homer.rdfs_label.one)
```

add_textelement_type(*label: str*, *lang: str = 'en'*) → Optional[surf.resource.Resource]
Adds a new TextElementType to the Knowledge base if not yet present.

Parameters

- **label** (*str*) – Description of parameter *label*.
- **lang** (*str*) – Description of parameter *lang*.

Returns Description of returned object.

Return type Optional[surf.resource.Resource]

```
# this will work only when connecting to a triples store
# where you have access in writing mode
>>> from hucit_kb import KnowledgeBase
>>> kb = KnowledgeBase()
>>> element_type_obj = kb.add_textelement_type("book")
```

add_textelement_types(*types: List[str]*) → None
Adds the text element type in case it doesn't exist.

Parameters **types** (*List[str]*) – a list of strings (e.g. ["book", "poem", "line"])

Returns Description of returned object.

Return type None

```
# this will work only when connecting to a triples store
# where you have access in writing mode
>>> from hucit_kb import KnowledgeBase
>>> kb = KnowledgeBase()
>>> kb.add_textelement_types(["book", "line"])
```

property author_names: Dict[str, str]
Returns a dictionary like this:

```
{
    "urn:cts:greekLit:tlg0012$$n1" : "Homer"
    , "urn:cts:greekLit:tlg0012$$n2" : "Omero"
    , ...
}
```

create_cts_urn(*resource: surf.resource.Resource*, *urn_string: str*) → Optional[surf.resource.Resource]
Creates a CTS URN object and assigns it to a given resource.

Parameters

- **resource** (*surf.resource.Resource*) – KB entry to be identified by the CTS URN.
- **urn_string** (*str*) – CTS URN identifier (e.g. `urn:cts:greekLit:tlg0012`)

Returns The newly created object or `None` if it already existed.

Return type Optional[surf.resource.Resource]

create_text_element(*work: surf.resource.Resource*, *urn_string: str*, *element_type: surf.resource.Resource*, *source_uri: str = None*)

Short summary.

Parameters

- **urn** (*str*) – Text element's URN.
- **element_type** (*surf.resource.Resource*) – Text element type.

Returns The newly created text element.

Return type type

```
>>> iliad = kb.get_resource_by_urn("urn:cts:greekLit:tlg0012.tlg001")
>>> etype_book = kb.get_textelement_type("book")
>>> ts = iliad.structure
>>> ts.create_element(
    "urn:cts:greekLit:tlg0012.tlg001:1",
    element_type=type_book,
    following_urn="urn:cts:greekLit:tlg0012.tlg001:2"
)
```

get_author_label(*urn*)

Get the label corresponding to the author identified by the CTS URN.

try to get an lang=en label (if multiple labels in this lang pick the shortest) try to get a lang=la label (if multiple labels in this lang exist pick the shortest) try to get a lang=None label (if multiple labels in this lang exist pick the shortest)

returns None if no name is found

get_authors() → List[*hucitlib.surfext.HucitAuthor*]

Lists all authors contained in the knowledge base.

Returns A list of authors.

Return type List[*HucitAuthor*]

get_opus_maximum_of(*author_cts_urn*)

Return the author's opux maximum (None otherwise).

Given the CTS URN of an author, this method returns its opus maximum. If not available returns None.

Parameters **author_cts_urn** – the author's CTS URN.

Returns an instance of *surfext.HucitWork* or None

get_resource_by_urn(*urn*)

Fetch the resource corresponding to the input CTS URN.

Currently supports only HucitAuthor and HucitWork.

Parameters **urn** – the CTS URN of the resource to fetch

Returns either an instance of *HucitAuthor* or of *HucitWork*

`get_statistics`() → Dict[str, int]
Gather basic stats about the Knowledge Base and its contents.

Note: This method currently has some performances issues.

Returns a dictionary

`get_textelement_type`(*label: str*) → Optional[surf.resource.Resource]
Returns a TextElementType (instance of E55_Type) if present.

Note: `label` (lowercased) is used to create the URI (http://purl.org/hucit/kb/types/{label}).

Parameters `label` (*str*) – Description of parameter *label*.

Returns Description of returned object.

Return type surf.resource.Resource

`get_textelement_types`() → List[surf.resource.Resource]
Returns all TextElementTypes defined in the knowledge base.

Returns Description of returned object.

Return type List[surf.resource.Resource]

`get_work_label`(*urn*)
Get the label corresponding to the work identified by the input CTS URN.

try to get an lang=en label try to get a lang=la label try to get a lang=None label

returns None if no title is found

`get_works`()
Return the author's works.

Returns a list of *HucitWork* instances.

`search`(*search_string: str*) → List[Tuple[str, surf.resource.Resource]]
Searches for a given string through the resources' labels.

Parameters `search_string` (*str*) – Description of parameter *search_string*.

Returns Description of returned object.

Return type List[Tuple[str, Resource]]

`to_json`()
Serialises the content of the KnowledgeBase as JSON.

Returns TODO

CHAPTER

FIVE

KNOWLEDGE BASE POPULATION

Command line interface for populating the HuCit knowledge base.

Usage: hucitlib/populate.py –work=<cts_urn> –log-file=<path> –kb-config-file=<path> [–verbose]

Options:

--work=<cts_urn> CTS URN of the work whose citation structure should be populated

–kb-config-file=<path> Path to the configuration file (overwrites default configuration). –log-file=<path> Path to the log file –verbose Turn on verbose logging

Example:

python hucitlib/populate.py –work=urn:cts:greekLit:tlg0011.tlg004 –log-file=hucitlib/data/tests/populate-tlg0011.tlg004.log –kb-config-file=hucitlib/config/virtuoso_local.ini

hucitlib.populate.**download_text_structure**(*urn: str*, *basedir: str = '/home/docs/checkouts/readthedocs.org/user_builds/hucitlib/checkouts/latest/huc*, *sample_size: Optional[int] = None*) → None

Example:

```
>>> download_text_structure('urn:cts:greekLit:tlg0012.tlg001')
```

hucitlib.populate.**fetch_text_structure**(*urn: str*, *endpoint: str = 'http://cts.perseids.org/api/cts'*, *stop_at: int = -1*) → Dict[str, object]

Fetches the text structure of a given work from a CTS endpoint.

Parameters

- **urn** (*string*) – the work's CTS URN (at the work-level!, e.g."urn:cts:greekLit:tlg0012.tlg001")
- **endpoint** (*string*) – the URL of the CTS endpoint to use (defaults to Perseids')

Returns a dict with keys "urn", "provenance", "valid_reffs", "levels"

Return type dict

hucitlib.populate.**populate_text_structure**(*kb: hucitlib.kb.KnowledgeBase*, *work: surf.resource.Resource*, *ts: Dict*) → None

Short summary.

Parameters

- **kb** (KnowledgeBase) – Description of parameter *kb*.
- **work** (*Resource*) – Description of parameter *work*.
- **ts** (*Dict*) – Description of parameter *ts*.

Returns Description of returned object.

Return type None

CHAPTER

SIX

SURF MAPPINGS

hucitlib relies on SuRF , a Python Object RDF Mapper library, so as to In order to make the knowledge base as much as possible easy to use programmatically. SuRF works similarly to Object-relation mappers (such as SQLAlchemy) with the main difference that Python objects are mapped to contents of a triples store rather than of a database.

A set of *SuRF <https://pythonhosted.org/SuRF/>* mappings is defined in order to ease the programmatic interaction with the knowledge base, and to away certain complexities of the underlying *Data model*.

Mappings are defined for the following classes:

- frbroo:F10_Person -> *HucitAuthor*
- frbroo:F1_Work -> *HucitWork*
- hucit:TextElement -> *HucitTextElement*
- hucit:TextStructure -> *HucitTextStructure*

6.1 Authors

class hucitlib.surfext.**HucitAuthor**

Object mapping for class frbroo:F10_Person.

add_abbreviation(*new_abbreviation*) → bool

Adds a new name abbreviation to an author's name.

Parameters **new_abbreviation** – the abbreviation to be added

Returns *True* if the abbreviation is added, *False* otherwise (the abbreviation is a duplicate)

add_name(*name: str*, *lang: Optional[str] = None*) → bool

Adds a new name variant to an author's name.

Parameters

- **name** (*str*) – The name variant to be added.
- **lang** (*str*) – The language of the name variant.

Returns *True* if the name is added, *False* otherwise (the name is a duplicate)

Return type bool

get_abbreviations() → List[str]

Get abbreviations of the names of the author.

Returns A list of known abbreviations.

Return type List[str]

Example:

```
>>> kb = KnowledgeBase()
>>> homer = kb.get_resource_by_urn('urn:cts:greekLit:tlg0012')
>>> homer.get_abbreviations()
['Hom.']
```

get_names() → Dict[str, str]
Returns a list of author's name variants.

Returns A dictionary where key is the language and value is the name in that language.

Return type Dict[str]

Example:

```
>>> homer = kb.get_resource_by_urn('urn:cts:greekLit:tlg0012')
>>> homer.get_names()
[('en', 'Homer'),
(None, 'Homeros'),
('la', 'Homerus'),
('fr', 'Homère'),
('it', 'Omero')]
```

get_urn() → Optional[pyCTS.CTS_URN]
Returns the author's CTS URN.

Note: It is assumed that each HucitAuthor has only one CTS URN.

Returns Description of returned object.

Return type Optional[CTS_URN]

get_works() → List[*hucitlib.surfext.HucitWork*]
Returns a list of the works (intances of *surf.Resource* and *HucitWork*) attributed to a given author.

set_urn(*urn: str*) → Optional[pyCTS.CTS_URN]
Changes the CTS URN of the author or adds a new one (if no URN is assigned).

Parameters **urn** (*str*) – The new CTS URN.

Returns Description of returned object.

Return type Optional[CTS_URN]

to_json() → None
Serialises a HucitAuthor to a JSON formatted string.

Note: This method will probably be deprecated in the near future.

Example:

```
>> homer = kb.get_resource_by_urn("urn:cts:greekLit:tlg0012")
>> homer.to_json()
{
  "name_abbreviations": [
```

(continues on next page)

(continued from previous page)


```
    "Hom."
  ],
  "urn": "urn:cts:greekLit:tlg0012",
  "works": [
    {
      "urn": "urn:cts:greekLit:tlg0012.tlg001",
      "titles": [
        {
          "language": "it",
          "label": "Iliade"
        },
        {
          "language": "la",
          "label": "Ilias"
        },
        {
          "language": "en",
          "label": "Iliad"
        },
        {
          "language": "de",
          "label": "Ilias"
        },
        {
          "language": "fr",
          "label": "L'Iliade"
        }
      ],
      "uri": "http://purl.org/hucit/kb/works/2815",
      "title_abbreviations": [
        "Il."
      ]
    },
    ...
  ],
  "uri": "http://purl.org/hucit/kb/authors/927",
  "names": [
    {
      "language": "fr",
      "label": "Homère"
    },
    {
      "language": "la",
      "label": "Homerus"
    },
    {
      "language": null,
      "label": "Homeros"
    },
    {
      "language": "en",
      "label": "Homer"
```


(continues on next page)

(continued from previous page)

```
    },
    {
      "language": "it",
      "label": "Omero"
    }
  ]
}
```

6.2 Works

class hucitlib.surfext.**HucitWork**

Object mapping for instances of *http://erlangen-crm.org/efrbroo/F1_Work*.

add_abbreviation(*new_abbreviation*)

Adds a new name variant to a work.

Parameters **new_abbreviation** – the abbreviation to be added

Returns *True* if the abbreviation is added, *False* otherwise (the abbreviation is a duplicate)

add_text_structure(*label: str*, *lang: str = 'en'*)

Adds a citable text structure to the work.

add_title(*title*, *lang=None*)

Adds a new title variant to a work.

Parameters

- **title** – the title to be added
- **lang** – the language of the title variant

Returns *True* if the title is added, *False* otherwise (the title is a duplicate)

property author: *hucitlib.surfext.HucitAuthor*

Returns the author to whom the work is attributed.

Returns an instance of *HucitWork* # TODO: check that's the case

get_abbreviations(*combine=False*)

TODO: if *combine==True*, concatenate with author abbreviation(s)

Get abbreviations of the titles of the work.

Returns a list of strings (empty list if no abbreviations available).

get_citation_structure() → List[hucitlib.surfext.CitationLevel]

Returns a sorted list of citation levels

[(1, 'book', ...), (2, 'line', ...),

]

get_titles()

TODO

get_top_elements()

TODO

get_urn()
Get the CTS URN that identifies the work.

Returns an instance of *pyCTS.CTS_URN* or None

has_text_structure()
Checks whether a citable text structure is defined.

Returns boolean

is_opus_maximum()
Check whether the work is the author's opus maximum.

Two cases: 1. the work is flagged as opus max 2. there is only one work by this author

Returns boolean

remove_text_structure(*text_structure*) → None
Remove any citable text structure to the work.

set_as_opus_maximum()
Mark explicitly the work as the author's opus maximum.

set_urn(*urn*)
Change the CTS URN of the author or adds a new one (if no URN is assigned).

to_json()
Serialises a HucitWork to a JSON formatted string.

6.3 Citable text structures and text elements

class hucitlib.surfext.**HucitTextStructure**
Object mapping for instances of *http://purl.og/net/hucit#TextStructure*.

property work
Returns the parent object (*HucitWork*).

class hucitlib.surfext.**HucitTextElement**
Object mapping for instances of *http://purl.og/net/hucit#TextElement*.

add_relations(*parent: surf.resource.Resource = None*, *previous: surf.resource.Resource = None*, *next: surf.resource.Resource = None*) → None
Short summary.

Parameters

- **parent** (*Resource*) – Description of parameter *parent*.
- **previous** (*Resource*) – Description of parameter *previous*.
- **next** (*Resource*) – Description of parameter *next*.

Returns Description of returned object.

Return type None

property children: List[surf.resource.Resource]
Returns the children text element(s) (if any).

get_type(*as_string: bool = True*) → Union[str, surf.resource.Resource]
Short summary.

Parameters **as_string** (*bool*) – Description of parameter *as_string*.

Returns Description of returned object.

Return type Union[str, Resource]

`get_urn()` → pyCTS.CTS_URN
Returns the TextElement's CTS URN.

`property next: Optional[surf.resource.Resource]`
Returns the following text element (if any).

`property parent`
Returns the parent (if any).

`property previous`
Returns the preceding text element (if any).

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- *Search*

PYTHON MODULE INDEX

h

INDEX

K

M

N

P

R

S

T

W