Italian Art

Class participants

2023-04-27

Table of contents

Chapter 1

Part of the series: Italian Art

Programme instructions

2023-03-17 v1.0

Dieses SPARQL-Query wird verwendet, um eine Liste von Gemälden abzurufen, die von italienischen Künstlern geschaffen wurden und deren Ursprungsort in Italien liegt.

Example publications:

- Exhibition Catalogue (Work in progress) - https://nfdi4culture.github.io/catalogue-003/ (content from the current repo)
- Exhibition catalogue demo: toc Baroque /toc from Experimental Books – Re-imagining Scholarly Publishing, COPIM. Workshop URL: https://experimentalbooks.pubpub.org/programme-overview
- Publishers catalogue demo: ScholarLed A catalogue of ScholarLed presses built on a Quarto / Jupyter Notebook model for computational publishing. The publication is automatically updated daily to reflect any new books added by the publishers.
- Proof of concept #1 - Computational Publication: Computational Publishing for Collections - ADA CP Prototype #1 - Nov 22
- Proof of concept #2 - To be confirmed, completion for end of April 2023. This contains all parts fully rendered: Cover, colophon, essay, collection, graph, TIB AV Portal, Semantic Kompakkt
- semanticClimate: To be confirmed - customised research papers readers made for regional climate change action plans based on IPCC reports and

sourcing content from open research repositories.

- FSCI Summer School - publishing from collections class: To be confirmed, July 2023

Chapter 2

Colophon

Fork title - Publishing Task

Author - Tuku

ORCID -

Date - April 27, 2023

DOI - https://doi.org/10.5281/zenodo.7872070

Repository URL - https://github.com/mloran/catalogue-003

Chapter 3

Catalogue Experiment: Italian Paitings

`Nextcloud Markdown document link: https://tib.eu/cloud/s/qBx8SbqiPBBedye`

3.1 Part of the series: Italian Paintings

- Class instructions and all links: https://nfdi4culture.github.io/class-ADA-CP-pipeline/
- Demo publication: https://nfdi4culture.github.io/catalogue-003/
- Repo link: https://github.com/NFDI4Culture/catalogue-003

3.2 Author:

- Memo Loran Tuku

3.3 Topic

Dieses SPARQL-Query wird verwendet, um eine Liste von Gemälden abzurufen, die von italienischen Künstlern geschaffen wurden und deren Ursprungsort in Italien liegt. Es gibt fünf Variablen, die zurückgegeben werden:

?painting: die URI des Gemäldes ?paintingLabel: der Label (Name) des Gemäldes in Deutsch ?artist: die URI des Künstlers ?artistLabel: der Label (Name) des Künstlers in Deutsch ?originLabel: der Label (Name) des Ursprungsortes des Gemäldes in Deutsch Das Query wird auf Wikidata ausgeführt und nutzt die Wikidata Query Service-API. Um die Labels in Deutsch

anzuzeigen, wird der Service wikibase:label verwendet, mit dem Parameter wikibase:language auf “de” gesetzt.

https://openai.com/blog/chatgpt

https://www.perplexity.ai/

Chapter 4

Activity: Paintings catalogue in Jupyter Notebook

Objective: Make a selection of nine paintings for the exhibition catalogue to be selected from Wikidata and rendered multi-format in Quarto.

```
KeyError: 'item'
```

The below Python code uses SPARQLWrapper to retrieve data from Wikidata based on a SPARQL query.

Chapter 5

pip install sparqlwrapper

```
# https://rdflib.github.io/sparqlwrapper/

import sys
from SPARQLWrapper import SPARQLWrapper, JSON

from PIL import Image
import requests

endpoint_url = "https://query.wikidata.org/sparql"

query = """SELECT ?painting ?paintingLabel ?artist ?artistLabel ?originLabel ?image WHERE {
  ?painting wdt:P31 wd:Q3305213;
    wdt:P170 ?artist;
    wdt:P276 ?origin.
  ?artist wdt:P27 wd:Q38.
  ?origin wdt:P17 wd:Q38.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "de". }
  OPTIONAL { ?painting wdt:P18 ?image. }
}
LIMIT 9"""

# SUBROUTINES

def get_delay(date):
    try:
        date = datetime.datetime.strptime(date, '%a, %d %b %Y %H:%M:%S GMT')
        timeout = int((date - datetime.datetime.now()).total_seconds())
    except ValueError:
```

```
        timeout = int(date)
    return timeout

def get_image(url, headers):
    r = requests.get(url, headers=headers, stream=True)
    if r.status_code == 200:
        im = Image.open(r.raw)
        return im
    if r.status_code == 500:
        return None
    if r.status_code == 403:
        return None
    if r.status_code == 429:
        timeout = get_delay(r.headers['retry-after'])
        print('Timeout {} m {} s'.format(timeout // 60, timeout % 60))
        time.sleep(timeout)
        get_image(url, headers)

def get_results(endpoint_url, query):
    user_agent = "WDQS-example Python/%s.%s" % (sys.version_info[0], sys.version_inf
    # TODO adjust user agent; see https://w.wiki/CX6
    sparql = SPARQLWrapper(endpoint_url, agent=user_agent)
    sparql.setQuery(query)
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()

# MAIN PROGRAM

results = get_results(endpoint_url, query)

for result in results["results"]["bindings"]:
    print('Wikidata link: ' + '[' + result['painting']['value'] + ']' + '(' + result
    print('Title: ' + result['paintingLabel']['value'] + '\n')
    print('Creator: ' + result['artistLabel']['value'] + '\n')

    if 'image' in result:
      # get image from image URL and display resized version
      image_url=result['image']['value']
      headers = {'User-Agent': 'Ex_Books_conference_bot/0.0 (https://github.com/Simo
      im = get_image(image_url, headers)
      im.thumbnail((500, 500), Image.Resampling.LANCZOS)
      display(im)
    print('\n\n')
```

```
Wikidata link: [http://www.wikidata.org/entity/Q28803748](http://www.wikidata.org/entit
```

Title: Q28803748

Creator: Giovanni Benedetto Castiglione

AttributeError: module 'PIL.Image' has no attribute 'Resampling'