Baroque TOC

Team Computational Publishing

4/28/23

Table of contents

1 Baroque TOC: Automating Exhibition Catalogue Creation **1**
1.0.1 Desccription 1
1.0.2 Cite as 2

2 Colophon **3**

3 Catalogue Essay **5**

4 Bavarian State Painting Collection **7**

5 Embedded video and 3D **19**
5.1 3D model embedding 19

Chapter 1

Baroque TOC: Automating Exhibition Catalogue Creation

Step-by-step guide: Automating Exhibition Catalogue Creation — A Guide

2023-04-28 v1.0

1.0.1 Desccription

An exhibition catalogue prototype, created using an open-source computational [publishing toolset}(https://copim.pubpub.org/pub/scholarled-catalogue/release/1). The objective was to test automatic retrieval of remote media and linked open data sources and then auto-typeseting the collated publication as multi-format. The prototype is available for community reuse to enable others to make their own publications and is accompanied by a step-by-step guide.

A collaboration between Open Science Lab TIB, NFDI4Culture, and COPIM:

- NFDI4Culture Task Area 4: Which is looking at which initiatives are enhancing their publications for open scholarship. Its aim is to establish a guideline for scholars to create publications and their associated data with a focus on long-term digital preservation.
- COPIM's Computational Book Publishing Pilot Project: WP6 brings together publishers, technologists, researchers, and authors to devise strategies to promote experimental book publishing and the reuse of, and engagement with, open access books.

Example workshop publication: toc Baroque /toc

1.0.2 Cite as

Document DOI: 10.5281/zenodo.7876062

Book cover: Reworking of Baroque pearl with enamelled gold mounts set with rubies. Creative Commons CC0 1.0 Universal Public Domain Dedication. This file was donated to Wikimedia Commons as part of a project by the Metropolitan Museum of Art. And, Venus and Cupid, Heinrich Bollandt, between circa 1620 and circa 1630. Bavarian State Painting Collections. This work is in the public domain.

Chapter 2

Colophon

Sample colophon data retrieved from Thoth.pub API book publication metadata system.

```
from thothlibrary import ThothClient
from datetime import datetime
import json

work ='9bf5f52e-6d99-4d62-bcc5-c57b07efa3f0'

# calling the Thoth GraphQL API
thoth = ThothClient()
response = thoth.work_by_id(work_id=work, raw='true')
data = json.loads(response)

print(data['data']['work']['title'].upper())
print(data['data']['work']['subtitle'].upper())
print('\r')
print(data['data']['work']['imprint']['publisher']['publisherName'])
print(data['data']['work']['place'])
print('\r')
print('First published ' + data['data']['work']['publicationDate'])
print('\r')
print('Copyright © ' + data['data']['work']['copyrightHolder'] + ' ' + data['data']['work']['p
print('Licensed as ' + data['data']['work']['license'])
print('\r')
print('DOI: ' + data['data']['work']['doi'])
```

PUBLISHING FROM COLLECTIONS USES OF COMPUTATIONAL PUBLISHIGN AND LINKEDOPEN DATA

Open Science Lab - TIB Hannnover

First published 2023-03-30

DOI: https://doi.org/10.5281/zenodo.7701161

Chapter 3

Catalogue Essay

Article taken from Wikipedia: Baroque

The Baroque (UK: /bə r k/, US: /bə ro k/; French: [ba k]) is a style of architecture, music, dance, painting, sculpture, poetry, and other arts that flourished in Europe from the early 17th century until the 1750s. In the territories of the Spanish and Portuguese empires including the Iberian Peninsula it continued, together with new styles, until the first decade of the 19th century. It followed Renaissance art and Mannerism and preceded the Rococo (in the past often referred to as "late Baroque") and Neoclassical styles. It was encouraged by the Catholic Church as a means to counter the simplicity and austerity of Protestant architecture, art, and music, though Lutheran Baroque art developed in parts of Europe as well.

The Baroque style used contrast, movement, exuberant detail, deep colour, grandeur, and surprise to achieve a sense of awe. The style began at the start of the 17th century in Rome, then spread rapidly to France, northern Italy, Spain, and Portugal, then to Austria, southern Germany, and Poland. By the 1730s, it had evolved into an even more flamboyant style, called rocaille or Rococo, which appeared in France and Central Europe until the mid to late 18th century.

In the decorative arts, the style employs plentiful and intricate ornamentation. The departure from Renaissance classicism has its own ways in each country. But a general feature is that everywhere the starting point is the ornamental elements introduced by the Renaissance. The classical repertoire is crowded, dense, overlapping, loaded, in order to provoke shock effects. New motifs introduced by Baroque are: the cartouche, trophies and weapons, baskets of fruit or flowers, and others, made in marquetry, stucco, or carved.

Chapter 4

Bavarian State Painting Collection

An example notebook retreiving a sample of nine paintings via Wikidata from the Bavarian State Painting Collection. Here is the SPARQL query used in the Code section below. https://w.wiki/6VCz.

The notebooks is a sample 9 paintings from the Baroque period.

The complete collection is here on Wikidata.

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

```
from SPARQLWrapper import SPARQLWrapper, JSON
from PIL import Image
import requests


sparql_endpoint_url = 'https://query.wikidata.org/bigdata/namespace/wdq/sparql'
wikibase_url = 'https://www.wikidata.org'
api_url = '/w/api.php'

# Wikidata requires a user agent header to prevent spam requests
user_agent = 'Ex_Books_conference_bot/0.0 (https://github.com/SimonXIX/Experimental_Books_work

# SPARQL query
# see in Wikidata's Query Service GUI at:
# https://w.wiki/6VCz
query = """
```

```
#defaultView:ImageGrid
SELECT ?item ?itemLabel ?inceptionyear ?creator ?creatorLabel ?copyright ?copyrightL
WHERE
{
  # find items which:
  # are instances of (wdt:P31) paintings (wd:Q3305213)
  # have the property (wdt:P195) of being in collection wd:Q812285 (Bavarian State P
  ?item wdt:P31 wd:Q3305213 .
  ?item wdt:P195 wd:Q812285 .
  # get the item's creator property (wdt:P170)
  ?item wdt:P170 ?creator .
  # get the item's image property (wdt:P18)
  ?item wdt:P18 ?image .
  # get the item's copyright status (wdt:P6216)
  ?item wdt:P6216 ?copyright .
  {
    ?item wdt:P571 ?inception.
    BIND(YEAR(?inception) AS ?inceptionyear)
  }

  # filter out all paintings not created between the years 1600 and 1700
  FILTER((1600  <= ?inceptionyear) && (?inceptionyear < 1700 ))
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". } }
# limit to nine results
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)

# MAIN PROGRAM

# create SPARQL query
sparql = SPARQLWrapper(sparql_endpoint_url, agent=user_agent)

# retrieve results and convert to JSON format
sparql.setQuery(query)
sparql.setReturnFormat(JSON)
result = sparql.query().convert()

# for each result, print various data fields
for item in result['results']['bindings']:
    print('Wikidata link: ' + '[' + item['item']['value'] + ']' + '(' + item['item']['value']
    print('Title: ' + item['itemLabel']['value'] + '\n')
    print('Year: ' + item['inceptionyear']['value'] + '\n')
    print('Creator: ' + item['creatorLabel']['value'] + '\n')
    print('Copyright: ' + item['copyrightLabel']['value'] + '\n')

    # get image from image URL and display resized version
    image_url=item['image']['value']
    headers = {'User-Agent': 'Ex_Books_conference_bot/0.0 (https://github.com/SimonXIX/Experim
    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/Q29474642

Title: The Birth of Benjamin

Year: 1650

Creator: Francesco Furini

Copyright: public domain

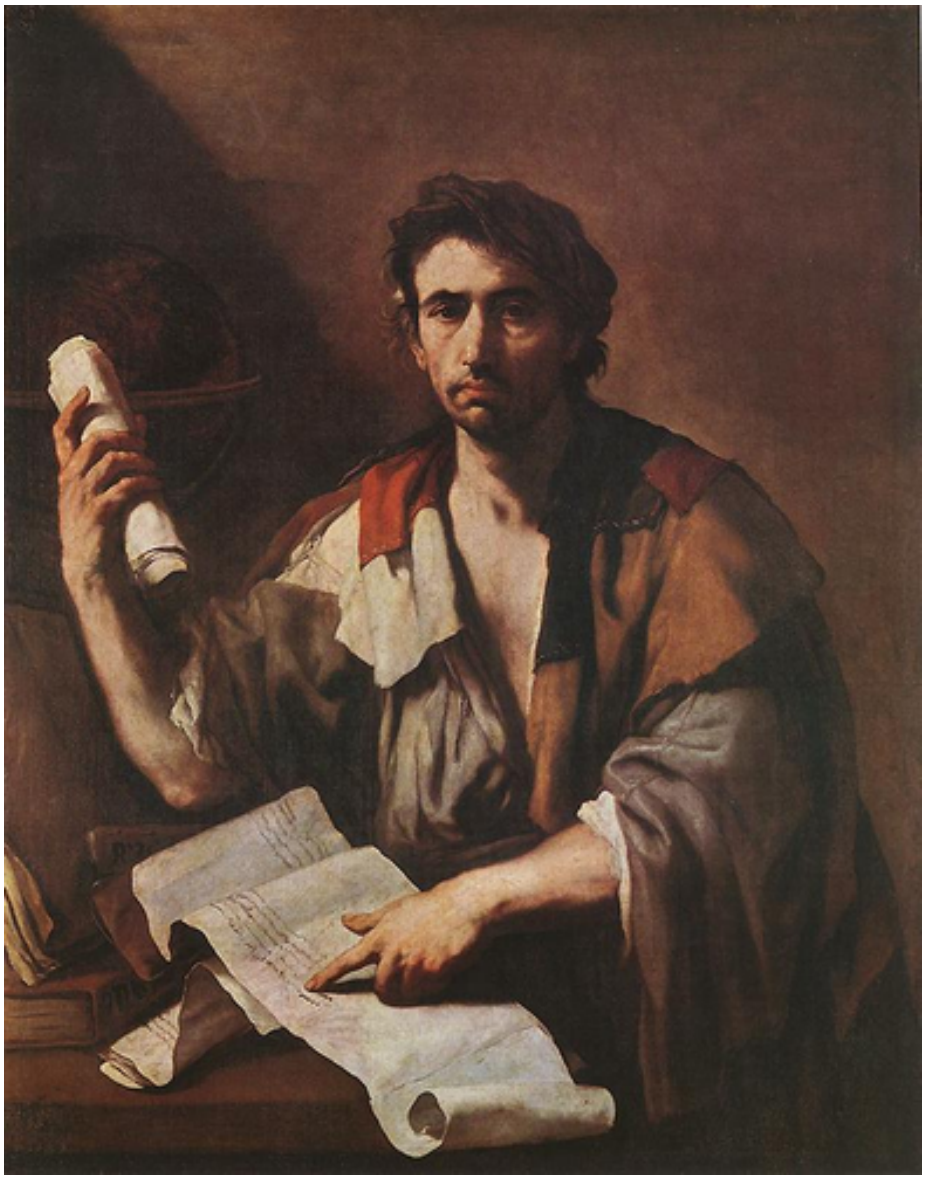
Wikidata link: http://www.wikidata.org/entity/Q29474649

Title: A Cynical Philosopher

Year: 1650

Creator: Luca Giordano

Copyright: public domain



Wikidata link: http://www.wikidata.org/entity/Q29474651

Title: Solomon and the Queen of Sheba

Year: 1697

Creator: Luca Giordano

Wikidata link: http://www.wikidata.org/entity/Q29477235

Title: Q29477235

Year: 1674

Creator: Antonio Triva

Copyright: public domain

Wikidata link: http://www.wikidata.org/entity/Q29477863

Title: Q29477863

Year: 1633

Creator: Guido Reni

Wikidata link: http://www.wikidata.org/entity/Q29477898

Title: Still-Life with Books

Year: 1628

Creator: Jan Lievens

Copyright: public domain

Wikidata link: http://www.wikidata.org/entity/Q29480557

Title: Feast of Herod

Year: 1630

Creator: http://www.wikidata.org/.well-known/genid/3f945710e81609ba4bae458b2820460a

Wikidata link: http://www.wikidata.org/entity/Q29480565

Title: Venus and Cupid

Year: 1625

Creator: Heinrich Bollandt

Wikidata link: http://www.wikidata.org/entity/Q29480594

Title: Still-life with Parrot

Year: 1630

Creator: Georg Flegel

Copyright: public domain

Chapter 5

Embedded video and 3D

The below Python code experiments with retrieving video data via iframe embedding. Video from TIB AV Portal.

```
from IPython.display import HTML
HTML("""
<iframe width="600" height="315" scrolling="no" src="https://av.tib.eu/player/56162" framebord
""")
```

```
<IPython.core.display.HTML object>
```

5.1 3D model embedding

The below Python code experiments with retrieving 3D data via iframe embedding. 3D models from demo instance of Semantic Kompakkt at kompakkt.wbworkshop.tibwiki.io

```
from IPython.display import HTML
HTML("""
<iframe
  width="600" height="315"
  name="Cube STL"
  src="https://kompakkt.wbworkshop.tibwiki.io/viewer/?entity=63e91957f3dcab9b642046ce&mode=ope
  allowfullscreen
  loading="lazy"
></iframe>
""")
```

```
<IPython.core.display.HTML object>
```

```
from IPython.display import HTML
HTML("""
<iframe
  width="600" height="315"
  name="Monopterus Schloss Monaise Trier Temple"
  src="https://kompakkt.wbworkshop.tibwiki.io/viewer/?entity=63e8d2fbfbc272b762204e1
  allowfullscreen
  loading="lazy"
></iframe>
""")
```

```
<IPython.core.display.HTML object>
```