Die Tafelstube

Lena-Marie Hoppe

2024-06-14

Table of contents

Chapter 1

Katalog zur Ausstellung: Die Tafelstube (Belagerungsszenen des Langen Türkenkriegs an der Decke)

Ein Katalog mit Kunstwerken aus der CbDD-Sammlung. Textteil: 6e73f774-4b7f-4e37-937b-e11cc35c5bc8

Die Tafelstube (Belagerungsszenen des Langen Türkenkriegs an der Decke) [Raum]

Chapter 2

Die Tafelstube

```
from datetime import datetime
import sys
import time
from SPARQLWrapper import SPARQLWrapper, JSON
import requests
from PIL import Image
import html

endpoint_url = "https://computational-publishing-service.wikibase.cloud/query/sparql"
#where the sparql queries come from

query_txt = """PREFIX cps: <https://computational-publishing-service.wikibase.cloud/entity/>
PREFIX cpss: <https://computational-publishing-service.wikibase.cloud/entity/statement/>
PREFIX cpsv: <https://computational-publishing-service.wikibase.cloud/value/>
PREFIX cpspt: <https://computational-publishing-service.wikibase.cloud/prop/direct/>
PREFIX cpsp: <https://computational-publishing-service.wikibase.cloud/prop/>
PREFIX cpsps: <https://computational-publishing-service.wikibase.cloud/prop/statement/>
PREFIX cpspq: <https://computational-publishing-service.wikibase.cloud/prop/qualifier/>

SELECT ?textItem ?kuratorLabel ?textUrl
WHERE
{
  <placeholder>
  ?textItem cpsp:P46 ?kuratorStatement.
  ?kuratorStatement cpsps:P46 ?kuratorItem.
  ?kuratorItem rdfs:label ?kuratorLabel.
  ?textItem cpsp:P57 ?urlstatement.
  ?urlstatement cpsps:P57 ?textUrl.
}"""
```

```
query_img = """PREFIX cps: <https://computational-publishing-service.wikibase.cloud/ent
PREFIX cpss: <https://computational-publishing-service.wikibase.cloud/entity/statement/
PREFIX cpsv: <https://computational-publishing-service.wikibase.cloud/value/>
PREFIX cpspt: <https://computational-publishing-service.wikibase.cloud/prop/direct/>
PREFIX cpsp: <https://computational-publishing-service.wikibase.cloud/prop/>
PREFIX cpsps: <https://computational-publishing-service.wikibase.cloud/prop/statement/>
PREFIX cpspq: <https://computational-publishing-service.wikibase.cloud/prop/qualifier/>

SELECT DISTINCT ?itemLabel ?itemDescr ?imgItem ?imgUrl ?publishDate
WHERE
{
  ?imgItem cpsp:P107 ?urlStatement.
  ?urlStatement cpsps:P107 ?imgUrl.
  ?imgItem cpsp:P60 ?dateStatement.
  ?dateStatement cpsps:P60 ?publishDate.
  ?imgItem cpsp:P6 ?partOfStatement.
  ?partOfStatement cpsps:P6 ?partOfItem.
  <placeholder>
  SERVICE wikibase:label {
      bd:serviceParam wikibase:language "de,en".
      ?imgItem rdfs:label ?itemLabel.
      ?imgItem schema:description ?itemDescr.
    }
}"""
query_graph = """PREFIX cps: <https://computational-publishing-service.wikibase.cloud/e
PREFIX cpss: <https://computational-publishing-service.wikibase.cloud/entity/statement/
PREFIX cpsv: <https://computational-publishing-service.wikibase.cloud/value/>
PREFIX cpspt: <https://computational-publishing-service.wikibase.cloud/prop/direct/>
PREFIX cpsp: <https://computational-publishing-service.wikibase.cloud/prop/>
PREFIX cpsps: <https://computational-publishing-service.wikibase.cloud/prop/statement/>
PREFIX cpspq: <https://computational-publishing-service.wikibase.cloud/prop/qualifier/>

SELECT ?x ?y
WHERE
{
  ?a cpsp:P2 ?c.
  ?c cpsps:P2 ?d.                        #also quasi überall wo sparql ist, muss was ang
  ?a rdfs:label ?x.
  ?d rdfs:label ?y.

}LIMIT 1"""

query_graph2 = """
SELECT ?a ?b ?c
WHERE
```

```
{
  ?a rdfs:label ?c
}LIMIT 100"""


def run_query(endpoint_url, query):
    user_agent = "WDQS-example Python/%s.%s" % (sys.version_info[0], sys.version_info[1])
    # TODO adjust user agent; see https://w.wiki/CX6
    sparql = SPARQLWrapper(endpoint_url, agent=user_agent)
    sparql.setQuery(query)
    sparql.setMethod("POST") #this NEEDS to be added to get results (not included in the wikibase
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()

def get_text(textitem_id):
    q = ""
    if textitem_id:
        q = query_txt.replace("<placeholder>", "cps:"+textitem_id+" cpsp:P46 ?kuratorStatement.")
    else:
        q = query_txt.replace("<placeholder>","")

    results_txt = run_query(endpoint_url, q)
    for item in results_txt["results"]["bindings"]:
        # print(item)
        print('Wikibase link: ' + '[' + item['textItem']['value'] + ']' + '(' + item['textItem'][
        print('Kurator: ' + item['kuratorLabel']['value'] + '\n')
        headers = {'User-Agent': 'Ex_Books_conference_bot/0.0 (https://github.com/SimonXIX/Experi
        r = requests.get(item['textUrl']['value'], headers=headers, stream=True)
        text = str(r.content)
        text = text.replace("ä","&auml;")
        text = text.replace("Ä","&Auml;")
        text = text.replace("ö","&ouml;")
        text = text.replace("Ö","&Ouml;")
        text = text.replace("ü","&uuml;")
        text = text.replace("Ü","&Uuml;")
        text = text.replace("ß","&szlig;")
        text = text.replace('\\n',"<br>")
        text = str(text)
        text = text.removeprefix("b'<!DOCTYPE html>").removesuffix("'")
        print(text)

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 fetch_image_by_url(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)
        fetch_image_by_url(url, headers)

def get_img(partOfItem_id):
    q = ""
    if partOfItem_id:
        q = query_img.replace("<placeholder>", "?partOfStatement cpsps:P6 cps:"+partOf
    else:
        q = query_img.replace("<placeholder>","")
    results_img = run_query(endpoint_url, q)
    for item in results_img["results"]["bindings"]:
      #print(item)
      print('Wikibase link: ' + '[' + item['imgItem']['value'] + ']' + '(' + item['img
      print('Title: ' + item['itemLabel']['value'] + '\n')
      print('Year: ' + item['publishDate']['value'] + '\n')
      print('Description: ' + html.unescape(item['itemDescr']['value']) + '\n')

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

def get_graph():
    import VizKG.visualize as vkg
    results_graph1 = run_query(endpoint_url, query_graph)
    #print(results_graph1)
```

```
    #print('---')
    results_graph2 = run_query(endpoint_url, query_graph2)
    #print(results_graph2)

    chart = vkg(sparql_query=query_graph2, sparql_service_url=endpoint_url, chart='wordcloud')
    chart.plot()
```

How to use your own text for processing

1. Add a new Text item to the wikibase. link to wikibase new item the item should contain the following statements:

- P57 (external link): link to the html file containing the new text
- P46 (kurator): Item of the curator. you may use an existing item like Q210 (Ulrike seeger) for test purposes
- P53 (license): Item of a license for the text. e.g Q203 (CC BY-NC-ND 4.0 DEED)
- P6 (is part of): set value to Q218 (Schlossanlage Weikersheim)

2. check if your new text item occurs in the list of selected text items: Link to wikibase query service

3. set parameter of get_text() to the id of your new text item e.g.: get_text("Q209")

```
get_text("Q232")
#Text zur Tafelstube
```

How to select images for processing

Images are selected via the sparql query. The method get_img() is capable of using a wikibase item id as parameter to select images with the property P6 (is part of) linking to the given item id.

1. select a valid location id from the query result: Link to wikibase query service

2. set parameter of get_img() to the id of your selected location item e.g.: get_img("Q217")

```
get_img("Q231")
#Bild Tafelstube
```

```
get_text("Q278")
#Belagerungsszenen des Langen Türkenkrieges
```

```
#Q252-Q263 = Texte Belagerungsszenen
```

```
get_text("Q252")
#Belagerungsszene 1
```

```
get_text("Q253")
#Belagerungsszene 2
```

```
get_text("Q254")
#Belagerungsszene 3
```

```
get_text("Q255")
#Belagerungsszene 4
```

```
get_text("Q256")
#Belagerungsszene 5
```

```
get_text("Q257")
#Belagerungsszene 6
```

```
get_text("Q258")
#Belagerungsszene 7
```

```
get_text("Q259")
#Belagerungsszene 8
```

```
get_text("Q260")
#Belagerungsszene 9
```

```
get_text("Q261")
#Belagerungsszene 10
```

```
get_text("Q262")
#Belagerungsszene 11
```

```
get_text("Q263")
#Belagerungsszene 12
```

```
get_text("Q264")
#Programm und Synthese der einstigen Tafelstube
```

```
get_graph()
```

Chapter 3

“ “

title: “Die Tafelstube” jupyter: python3 code-fold: true execute: echo: true output: asis

---

:::

::: {.cell}
``` {.python .cell-code}
from datetime import datetime
import sys
import time
from SPARQLWrapper import SPARQLWrapper, JSON
import requests
from PIL import Image
import html

endpoint_url = "https://computational-publishing-service.wikibase.cloud/query/sparql"
#where the sparql queries come from

query_txt = """PREFIX cps: <https://computational-publishing-service.wikibase.cloud/entity/>
PREFIX cpss: <https://computational-publishing-service.wikibase.cloud/entity/statement/>
PREFIX cpsv: <https://computational-publishing-service.wikibase.cloud/value/>
PREFIX cpspt: <https://computational-publishing-service.wikibase.cloud/prop/direct/>
PREFIX cpsp: <https://computational-publishing-service.wikibase.cloud/prop/>
PREFIX cpsps: <https://computational-publishing-service.wikibase.cloud/prop/statement/>
PREFIX cpspq: <https://computational-publishing-service.wikibase.cloud/prop/qualifier/>

SELECT ?textItem ?kuratorLabel ?textUrl
```

```
WHERE
{
  <placeholder>
  ?textItem cpsp:P46 ?kuratorStatement.
  ?kuratorStatement cpsps:P46 ?kuratorItem.
  ?kuratorItem rdfs:label ?kuratorLabel.
  ?textItem cpsp:P57 ?urlstatement.
  ?urlstatement cpsps:P57 ?textUrl.
}"""

query_img = """PREFIX cps: <https://computational-publishing-service.wikibase.cloud/ent
PREFIX cpss: <https://computational-publishing-service.wikibase.cloud/entity/statement/
PREFIX cpsv: <https://computational-publishing-service.wikibase.cloud/value/>
PREFIX cpspt: <https://computational-publishing-service.wikibase.cloud/prop/direct/>
PREFIX cpsp: <https://computational-publishing-service.wikibase.cloud/prop/>
PREFIX cpsps: <https://computational-publishing-service.wikibase.cloud/prop/statement/
PREFIX cpspq: <https://computational-publishing-service.wikibase.cloud/prop/qualifier/

SELECT DISTINCT ?itemLabel ?itemDescr ?imgItem ?imgUrl ?publishDate
WHERE
{
  ?imgItem cpsp:P107 ?urlStatement.
  ?urlStatement cpsps:P107 ?imgUrl.
  ?imgItem cpsp:P60 ?dateStatement.
  ?dateStatement cpsps:P60 ?publishDate.
  ?imgItem cpsp:P6 ?partOfStatement.
  ?partOfStatement cpsps:P6 ?partOfItem.
  <placeholder>
  SERVICE wikibase:label {
      bd:serviceParam wikibase:language "de,en".
      ?imgItem rdfs:label ?itemLabel.
      ?imgItem schema:description ?itemDescr.
    }
}"""
query_graph = """PREFIX cps: <https://computational-publishing-service.wikibase.cloud/e
PREFIX cpss: <https://computational-publishing-service.wikibase.cloud/entity/statement/
PREFIX cpsv: <https://computational-publishing-service.wikibase.cloud/value/>
PREFIX cpspt: <https://computational-publishing-service.wikibase.cloud/prop/direct/>
PREFIX cpsp: <https://computational-publishing-service.wikibase.cloud/prop/>
PREFIX cpsps: <https://computational-publishing-service.wikibase.cloud/prop/statement/
PREFIX cpspq: <https://computational-publishing-service.wikibase.cloud/prop/qualifier/

SELECT ?x ?y
WHERE
{
  ?a cpsp:P2 ?c.
```

```
  ?c cpsps:P2 ?d.                        #also quasi überall wo sparql ist, muss was angepasst wer
  ?a rdfs:label ?x.
  ?d rdfs:label ?y.

}LIMIT 1"""

query_graph2 = """
SELECT ?a ?b ?c
WHERE
{
  ?a rdfs:label ?c
}LIMIT 100"""


def run_query(endpoint_url, query):
    user_agent = "WDQS-example Python/%s.%s" % (sys.version_info[0], sys.version_info[1])
    # TODO adjust user agent; see https://w.wiki/CX6
    sparql = SPARQLWrapper(endpoint_url, agent=user_agent)
    sparql.setQuery(query)
    sparql.setMethod("POST") #this NEEDS to be added to get results (not included in the wikibase
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()

def get_text(textitem_id):
    q = ""
    if textitem_id:
        q = query_txt.replace("<placeholder>", "cps:"+textitem_id+" cpsp:P46 ?kuratorStatement.")
    else:
        q = query_txt.replace("<placeholder>","")

    results_txt = run_query(endpoint_url, q)
    for item in results_txt["results"]["bindings"]:
        # print(item)
        print('Wikibase link: ' + '[' + item['textItem']['value'] + ']' + '(' + item['textItem'][
        print('Kurator: ' + item['kuratorLabel']['value'] + '\n')
        headers = {'User-Agent': 'Ex_Books_conference_bot/0.0 (https://github.com/SimonXIX/Experi
        r = requests.get(item['textUrl']['value'], headers=headers, stream=True)
        text = str(r.content)
        text = text.replace("ä","&auml;")
        text = text.replace("Ä","&Auml;")
        text = text.replace("ö","&ouml;")
        text = text.replace("Ö","&Ouml;")
        text = text.replace("ü","&uuml;")
        text = text.replace("Ü","&Uuml;")
        text = text.replace("ß","&szlig;")
        text = text.replace('\\n',"<br>")
```

```
        text = str(text)
        text = text.removeprefix("b'<!DOCTYPE html>").removesuffix("'")
        print(text)

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 fetch_image_by_url(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)
        fetch_image_by_url(url, headers)

def get_img(partOfItem_id):
    q = ""
    if partOfItem_id:
        q = query_img.replace("<placeholder>", "?partOfStatement cpsps:P6 cps:"+partOf
    else:
        q = query_img.replace("<placeholder>","")
    results_img = run_query(endpoint_url, q)
    for item in results_img["results"]["bindings"]:
      #print(item)
      print('Wikibase link: ' + '[' + item['imgItem']['value'] + ']' + '(' + item['img
      print('Title: ' + item['itemLabel']['value'] + '\n')
      print('Year: ' + item['publishDate']['value'] + '\n')
      print('Description: ' + html.unescape(item['itemDescr']['value']) + '\n')

      # get image from image URL and display resized version
      image_url=item['imgUrl']['value']
      headers = {'User-Agent': 'Ex_Books_conference_bot/0.0 (https://github.com/SimonX
      im = fetch_image_by_url(image_url, headers)
      im.thumbnail((500, 500), Image.Resampling.LANCZOS)
```

```
      display(im)
      print('\n\n')

def get_graph():
    import VizKG.visualize as vkg
    results_graph1 = run_query(endpoint_url, query_graph)
    #print(results_graph1)
    #print('---')
    results_graph2 = run_query(endpoint_url, query_graph2)
    #print(results_graph2)

    chart = vkg(sparql_query=query_graph2, sparql_service_url=endpoint_url, chart='wordcloud')
    chart.plot()
```

```
get_text("Q278")
#Belagerungsszenen des Langen Türkenkrieges
```