Die Tafelstube

Lena-Marie Hoppe

2024-06-14

Table of contents

# 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](https://www.deckenmalerei.eu/42d06165-58e7-4653-bfe4-3d5f7091fc33#6e73f774-4b7f-4e37-937b-e11cc35c5bc8)

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

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 International License.

# 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/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 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/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 ?x ?y  
WHERE  
{  
 ?a cpsp:P2 ?c.  
 ?c cpsps:P2 ?d. #also quasi überall wo sparql ist, muss was angepasst werden  
 ?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 python example code)  
 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']['value'] + ')' + '\n')  
 print('Kurator: ' + item['kuratorLabel']['value'] + '\n')  
 headers = {'User-Agent': 'Ex\_Books\_conference\_bot/0.0 (https://github.com/SimonXIX/Experimental\_Books\_workshop; ad7588@coventry.ac.uk)'}  
 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:"+partOfItem\_id+".")  
 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['imgItem']['value'] + ')' + '\n')  
 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/SimonXIX/Experimental\_Books\_workshop; ad7588@coventry.ac.uk)'}  
 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](https://computational-publishing-service.wikibase.cloud/wiki/Special:NewItem) 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)

1. check if your new text item occurs in the list of selected text items: [Link to wikibase query service](https://computational-publishing-service.wikibase.cloud/query/#PREFIX%20cps%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fentity%2F%3E%0APREFIX%20cpss%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fentity%2Fstatement%2F%3E%0APREFIX%20cpsv%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fvalue%2F%3E%0APREFIX%20cpspt%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2Fdirect%2F%3E%0APREFIX%20cpsp%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2F%3E%0APREFIX%20cpsps%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2Fstatement%2F%3E%0APREFIX%20cpspq%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2Fqualifier%2F%3E%0A%0ASELECT%20%3FtextItem%20%3FkuratorLabel%20%3FtextUrl%0AWHERE%0A%7B%0A%20%20%3FtextItem%20cpsp%3AP46%20%3FkuratorStatement.%20%0A%20%20%3FkuratorStatement%20cpsps%3AP46%20%3FkuratorItem.%20%0A%20%20%3FkuratorItem%20rdfs%3Alabel%20%3FkuratorLabel.%0A%20%20%3FtextItem%20cpsp%3AP57%20%3Furlstatement.%20%0A%20%20%3Furlstatement%20cpsps%3AP57%20%3FtextUrl.%20%0A%7D)
2. 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](https://computational-publishing-service.wikibase.cloud/query/#PREFIX%20cps%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fentity%2F%3E%0APREFIX%20cpss%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fentity%2Fstatement%2F%3E%0APREFIX%20cpsv%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fvalue%2F%3E%0APREFIX%20cpspt%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2Fdirect%2F%3E%0APREFIX%20cpsp%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2F%3E%0APREFIX%20cpsps%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2Fstatement%2F%3E%0APREFIX%20cpspq%3A%20%3Chttps%3A%2F%2Fcomputational-publishing-service.wikibase.cloud%2Fprop%2Fqualifier%2F%3E%0A%0ASELECT%20DISTINCT%20%3FpartOfItem%20%3FpartOfItemLabel%0AWHERE%0A%7B%0A%20%20%3FimgItem%20cpsp%3AP107%20%3FurlStatement.%20%0A%20%20%3FurlStatement%20cpsps%3AP107%20%3FimgUrl.%20%0A%20%20%3FimgItem%20cpsp%3AP60%20%3FdateStatement.%20%0A%20%20%3FdateStatement%20cpsps%3AP60%20%3FpublishDate.%20%0A%20%20%3FimgItem%20cpsp%3AP6%20%3FpartOfStatement.%0A%20%20%3FpartOfStatement%20cpsps%3AP6%20%3FpartOfItem.%0A%20%20SERVICE%20wikibase%3Alabel%20%7B%0A%20%20%20%20%20%20bd%3AserviceParam%20wikibase%3Alanguage%20%22de%2Cen%22.%0A%20%20%20%20%20%20%3FpartOfItem%20rdfs%3Alabel%20%3FpartOfItemLabel.%0A%20%20%20%20%20%20%3FpartOfItem%20schema%3Adescription%20%3FpartOfItemDescr.%0A%20%20%20%20%7D%0A%7D%20GROUP%20BY%20%3FpartOfItem%20%3FpartOfItemLabel)
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()

# 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/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 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/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 ?x ?y  
WHERE  
{  
 ?a cpsp:P2 ?c.  
 ?c cpsps:P2 ?d. #also quasi überall wo sparql ist, muss was angepasst werden  
 ?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 python example code)  
 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']['value'] + ')' + '\n')  
 print('Kurator: ' + item['kuratorLabel']['value'] + '\n')  
 headers = {'User-Agent': 'Ex\_Books\_conference\_bot/0.0 (https://github.com/SimonXIX/Experimental\_Books\_workshop; ad7588@coventry.ac.uk)'}  
 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:"+partOfItem\_id+".")  
 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['imgItem']['value'] + ')' + '\n')  
 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/SimonXIX/Experimental\_Books\_workshop; ad7588@coventry.ac.uk)'}  
 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