Trainer Preparation

Suggested schedule (1-1.5 hr)

- ❏ Introduction & MediaWiki Action API overview (15 min)
- ❏ Hands-on activity related to the MediaWiki Action API (15 min)
- ❏ Wikidata API overview (15 min)
- ❏ Hands-on activity related to the Wikidata API (15 min)
- ❏ Commons API & Closing (10 min)

Trainer notes

Delete this slide when you have completed the preparation.

Wikimedia APIs

Srishti Sethi & Lucas Werkmeister
[[User:SSethi_(WMF)]] & [[User:Lucas_Werkmeister]]

WIKIMANIA
STOCKHOLM

Wikimedia APIs

Introduction to the APIs of Wikimedia projects

Wikipedia

Wikidata

Wikimedia Commons

Wikiquote

Wikisource

...

Trainer notes

Give a topic introduction: There are a few Wikimedia APIs, using which you can access data of so many Wikimedia projects (e.g., Wikipedia, Wikidata, Wikimedia Commons, Wikiquote, Wikisource, etc.). In this workshop, we will be taking a closer look at the MediaWiki Action API and the ones based on it, such as the Commons and the Wikibase API.

MediaWiki Action API
(Wikipedia)

Trainer notes

You can use the Action API to get data from the Wikimedia sites (powered on MediaWiki) and access to wiki features. In some cases, use of some modules may have been disabled, and write access might only be available to specific users: https://www.mediawiki.org/wiki/API:Restricting_API_usage.

In this tutorial, we will be using the MediaWiki Action API to fetch data from Wikipedia.

the free knowledge base with
5,896,873 articles that anyone can edit

https://en.wikipedia.org

Short Introduction

→ web service of type REST
→ allows access to wiki features related but not limited to:
  → authentication
  → page operations
  → search
  → accounts & users
→ used by third-party and MediaWiki extension developers, and site administrators

Trainer notes

Explain what a web service means - you can request data from client side to the server over HTTP and get response back in a standard format. REST means Representational State Transfer. In case of a Restful API, the server will transfer to the client representation of the state of the requested resource.

You can point to these resources on simple explanation of APIs:
https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-1-introduction-b4a072f8740f
https://restful.io/an-introduction-to-api-s-cee90581ca1b

More examples of features accessible via the API
https://www.mediawiki.org/wiki/Template:API

Supported Data Formats

Input

→ Encoding: Valid UTF-8
→ Multivalue parameters: separated by a “|” sign
→ Boolean parameters: `someParam=` for true and no parameter for false value
→ Timestamp formats: ISO 8601, MediaWiki’s internal timestamp, MySql, Unix, etc.

Output

→ JSON format

Trainer notes

Get a full overview of supported data formats here:
https://www.mediawiki.org/wiki/API:Data_formats

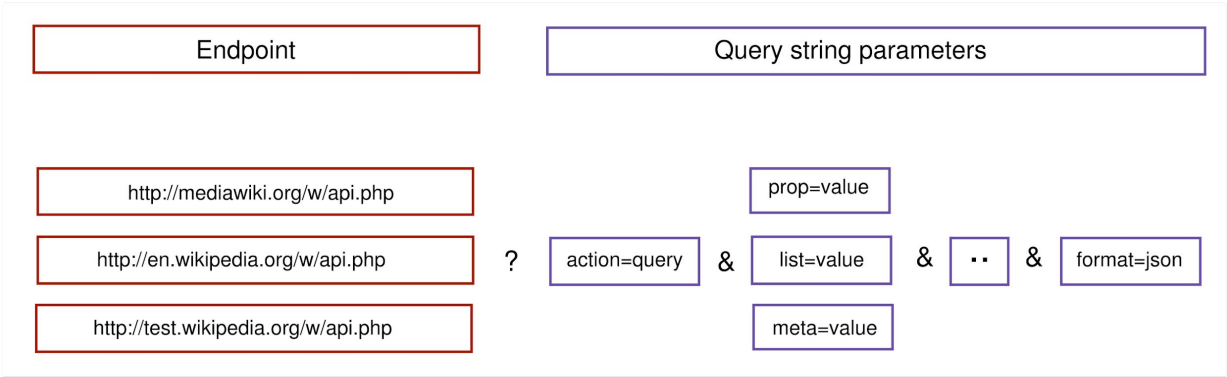
Best Practices

→ Request limit
→ Parsing of revisions
→ The maxlag parameter
→ The User-Agent header
→ Data formats
→ Performance

Trainer notes

Get a full overview of best practices for API use here:
https://www.mediawiki.org/wiki/API:Etiquette

Request Format



Trainer notes

A query string consists of an endpoint and query string parameters.

The endpoint would be of the URL to the API on a Wikimedia Wiki. For English Wikipedia, it would be https://en.wikipedia.org/w/api.php. The query string consists of:

* An action parameter that tells the API which action to perform. The most popular action is `query` that allows fetching data from a wiki.

* Second parameter tells the type of information to query. It is one of the three query submodule types: prop (get properties of pages), list (get a list of pages) and meta (information about the wiki and user).

* Some additional parameters to be a bit more specific about the nature of query.

* Final parameter to specify the format in which you would like to retrieve the results. The recommended format is JSON.

GET Request

Parse the content of a Wikipedia Page

→ as Wikitext

https://en.wikipedia.org/w/api.php?action=parse&prop=wikitext&page=Wikimania&format=json

→ as HTML

https://en.wikipedia.org/w/api.php?action=parse&page=Wikimania&format=json

Trainer notes

Explain each parameter in the URL. For reference, point to the “parse” documentation here: https://www.mediawiki.org/wiki/API:Parsing_wikitext#API_documentation.

Response


```
{
    "parse": {
        "title": "Wikimania",
        "pageid": 19660380,
        "wikitext": {
            "*": "{{pp-move-indef}}\n{{Use mdy dates|date=April 2012}}\n{{Infobox recurring event\n| name
= Wikimania\n| logo         = Wikimania logo with text.svg\n| logo_alt     = \n| logo_caption = The
Wikimania logo\n| logo_size    = 150px\n| image        = \n| image_size   =\n| alt          = \n| caption
= \n| status       = Active\n| genre        = Conference\n| frequency    = Annually\n| location   = Various
\n| first      = {{Start date and age|2005|8|5}}\n| last         = July 18\u201322, 2018<!-- Date of most
recent event; if the event will not be held again, use {{End date|YYYY|MM|DD|df=y}} -->\n| next         =
\n| participants = \n| organizer  = Local volunteer teams\n| filing     = Non-profit\n| website    =
{{URL|https://wikimania.wikimedia.org}}\n}}\n'''Wikimania''' is the official annual conference of the
[[Wikimedia Foundation]]. Topics of presentations and discussions include Wikimedia projects such as
[[Wikipedia]], other wikis, [[open-source software]], free knowledge and free content, and social and
technical aspects related to these topics.\n\nSince 2011, the winner of …
        }
    }
}
```


Trainer notes

Explain the key-value pairs in the JSON response.

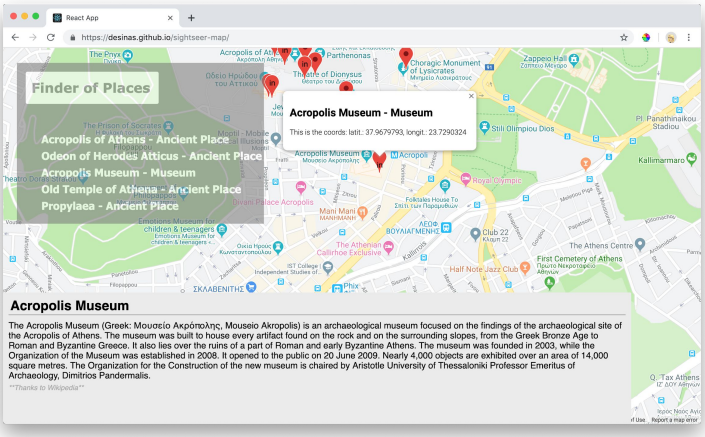
Sample code (Python)

```python
#!/usr/bin/python3

import requests

S = requests.Session()

URL = "https://en.wikipedia.org/w/api.php"

TITLE = "Wikimania"

PARAMS = {
    'action': "parse",
    'prop': "wikitext",
    'page': TITLE,
    'format': "json"
}

R = S.get(url=URL, params=PARAMS)
DATA = R.json()

print(DATA)
```

Trainer notes

Execute the python script on PAWS: https://paws-public.wmflabs.org/paws-public/User:SSethi_(WMF)/mediawiki-api-parse.ipynb

Demo App: Sightseer Map



https://github.com/desinas/sightseer-map

Trainer notes

Briefly explain the demo app, how it works, and API modules it uses: https://github.com/desinas/sightseer-map

Resources

→ API overview
https://www.mediawiki.org/wiki/API:Main_page
→ Tutorial
https://www.mediawiki.org/wiki/API:Tutorial
→ List of supported modules
https://www.mediawiki.org/w/api.php?action=help
→ Sandbox to experiment with the API
https://www.mediawiki.org/wiki/Special:ApiSandbox

Trainer notes

Database dumps are available as well so you can work with them locally https://dumps.wikimedia.org/ and recommended client libraries are listed here: https://www.mediawiki.org/wiki/API:Client_code.

Activity 1

Use any programming language to create a page in your Sandbox on Test Wiki. Add in it, contents from one of the sections here: **https://en.wikipedia.org/wiki/Wikimania**

Hint:
Use the "parse" & "edit" modules documentation:
→ **https://www.mediawiki.org/wiki/API:Parsing_wikitext**
→ **https://www.mediawiki.org/wiki/API:Edit**

Download the code snippets for exploring the API modules further:
https://github.com/wikimedia/mediawiki-api-demos

Trainer notes

Explain a bit about the POST requests first, and then give ~10-15 minutes to participants to get comfortable with making API calls. Before moving on to the next step, demonstrate how one can create a page in their Sandbox on test wiki and add in it contents from the Wikimania page via the “edit” and “parse” modules. Execute the Python scripts on PAWS: https://paws-public.wmflabs.org/paws-public/User:SSethi_(WMF)/mediawiki-api-edit.ipynb

Wikibase API (Wikidata)

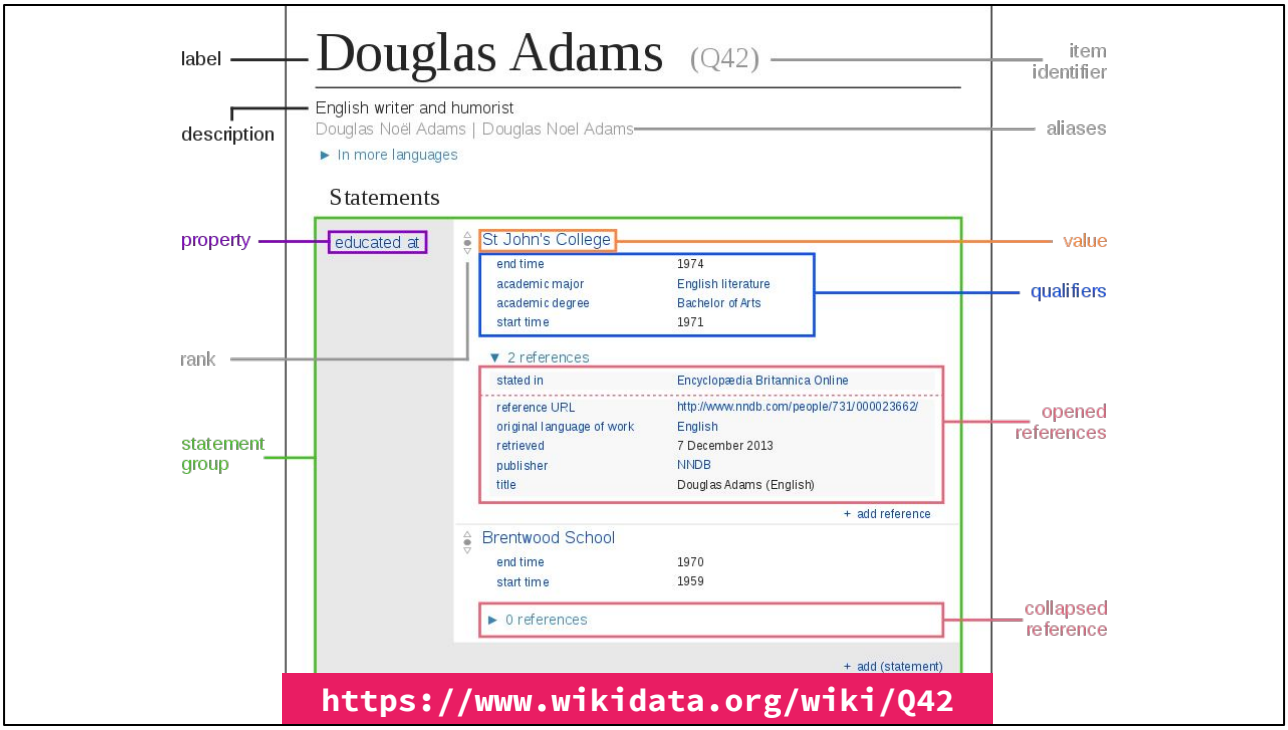
Trainer notes

You can use the Wikibase API to get data from Wikidata. A short introduction:
* Wikidata is a free knowledge base by the Wikimedia movement.
* Data collected is in the structured format that makes it easier to reuse it on Wikimedia sites.
* Both humans and bots edit Wikidata.
* Data is free and available for reuse under Creative Commons Public Domain dedication (CC0 1.0 Universal (CC0 1.0)) which allows copying, distributing, and modifying the data without asking for any permission.
* Data items are related to people, places, events, scientific topics, etc.
* It is an example of Linked Data, meaning the data items are connected. Interesting talk on this topic by Tim Berners Lee: https://www.ted.com/talks/tim_berners_lee_on_the_next_web
* Powered by Wikibase which is the software consisting of MediaWiki extensions allowing storing and accessing of structured data.

You can learn more here: https://www.wikidata.org/wiki/Wikidata:Introduction.

the free knowledge base with
58,604,264 data items that anyone can
edit

https://www.wikidata.org



Trainer notes

Wikidata repository consists of a lot of data items. Each data item has its own page on Wikidata. It is structured, and the data model is made of:
* Label: Name of the item.
* Item identifier: A "Q" followed by a number.
* Description: Short explanation of the item.
* Aliases: Alternative names of the data item.
* Statements: Describe characteristics of an item and consists of a set of property and value pairs.
** Property: A data type which is assigned an identifier with a "P" followed by a number.
** Values: Describes the data value. There can be multiple values for a property, and each value can have its own item in Wikidata.
** Qualifiers: Further describe the value of a property such as when the values date from, how they were determined, what exactly they refer to, etc.
** References: Sources that back up the data provided in the statement. Two options: a) publications and media b) websites and online databases.
** Rank: Interpret multiple values for a property in a statement. Three types: deprecated, preferred and normal.
** Claims: Form part of statement on item pages.

Accessing Wikidata's Data

https://www.wikidata.org/wiki/Wikidata:Data_access

Per-item Access

- Linked Data Interface
- SPARQL Endpoints
- Wikibase API

Dumps

Trainer notes

There are many ways to access Wikidata's data. We will take a high-level overview of different methods and focus more on the Wikibase API.

Linked Data Interface

→ data of single item can be retrieved via

http://www.wikidata.org/wiki/Special:EntityData/Q7251

→ to get the data of the item in **.json** format

http://www.wikidata.org/wiki/Special:EntityData/Q7251.json

→ other possible formats:

.rdf, **.ttl** or **.nt**

Trainer notes

Entity is the content of a Wikidata page, such as an item or a property. Every entity is identified by an ID (starts with prefix Q for an item and P for a property). Entities have a URI of the format: http://www.wikidata.org/Special:EntityData/*ID* where *ID* is the entity ID.

More on supported file formats:
https://en.wikipedia.org/wiki/N-Triples (nt)
https://en.wikipedia.org/wiki/Turtle_(syntax) (ttl)
https://en.wikipedia.org/wiki/Resource_Description_Framework (rdf)

SPARQL Endpoints

→ RDF query language for databases

→ expression format: item-property-value

→ query service: https://query.wikidata.org/

→ query example: humans of Sweden who are either writers or teachers

```
select ?item ?itemLabel ?itemDescription WHERE
{
  ?item wdt:P31 wd:Q5 . # humans
  ?item wdt:P27 wd:Q34 . # citzenship
  { ?item wdt:P106 wd:Q36180 . } # occupation - writer
  UNION
  { ?item wdt:P106 wd:Q37226 .} # occupation - teacher

  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en" }
}
```

https://w.wiki/6zu

Trainer notes

SPARQL is a query language recommended by the W3C. It is a language of databases that allows you to retrieve and manipulate data stored in an RDF format. The SPARQL query is in a triple form: item-property-value. This format is analogous to RDF where you can construct statements about resources in subject-predicate-object format. For example, one way to represent the statement "Alan Turing is an instance of human race" in RDF/SPARQL would be – an item denoting "Alan Turing," a property denoting "instance of," and a value indicating "humans."

Give an example of a sample query like in https://w.wiki/6m7 via the Wikidata Query Service. https://query.wikidata.org.

Tutorials on SPARQL:
https://www.mediawiki.org/wiki/Wikidata_Query_Service/User_Manual
Querying Wikidata with SPARQL for absolute beginners
https://www.youtube.com/watch?v=kJph4q0Im98 (recommended)

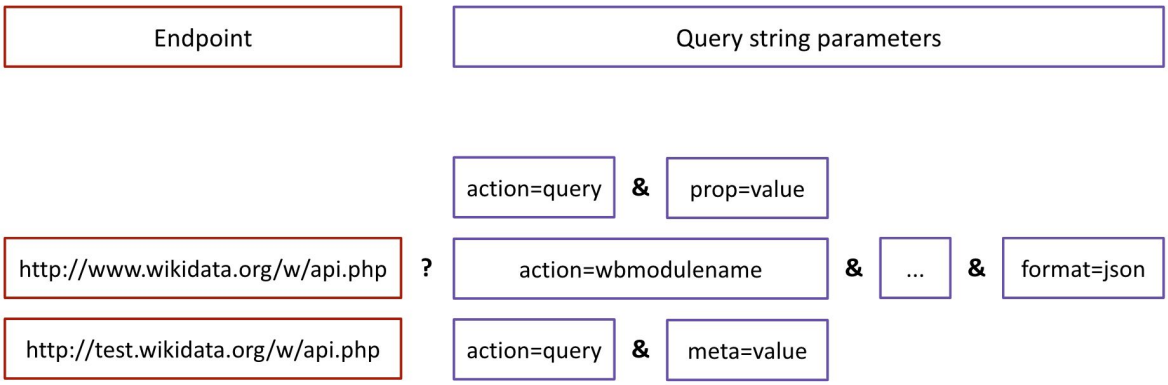
Wikibase API: Short Introduction

→ Wikibase
  → MediaWiki extension
  → Powers Wikidata
  → store or access structured data
→ API
  → Similar to the MediaWiki Action API
  → Allows querying, adding, removing and editing Wikidata information

Trainer notes

More on overview of Wikibase and Wikibase API:
https://www.mediawiki.org/wiki/Wikibase
https://www.mediawiki.org/wiki/Wikibase/API#What_is_the_Wikibase_API?

Request Format



Trainer notes

Request format is similar to the Action API. A query string consists of an endpoint and query string parameters. The endpoint would be of the URL to the API on Wikidata Wiki: https://www.wikidata.org/w/api.php.

There are three types of modules: API, Meta, and Property submodules. You could either use `action=wgmodule` or meta and property submodule in combination with `action=query`. It is possible to get or set data from/of entities.

Give a high-level overview of different modules and what they support: https://www.mediawiki.org/wiki/Wikibase/API#API_documentation_and_Wikibase_modules.

Get Request

→ get the data of a single item

https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q7251&format=json

→ wbgetentities, wbgetclaims, wbsearchentites

Trainer notes

Demonstrate use of additional parameters supported by the ‘wbgetentities’ module like in: https://www.mediawiki.org/wiki/Wikibase/API#GET_request.

Response


```
{
    "entities": {
        "Q7251": {
            "pageid": 8455,
            "ns": 0,
            "title": "Q7251",
            "lastrevid": 992109551,
            "modified": "2019-08-05T03:19:16Z",
            "type": "item",
            "id": "Q7251",
            "labels": {},
            "descriptions": {},
            "aliases": {},
            "claims": {
                "P91": [
                    {
                        "mainsnak": { },
                        "type": "statement",
                        "id": "q7251$152AF535-A578-403A-8E4E-FA8BA1AA3AFB",
                        "rank": "normal",
                        "references": [ ]
                    }
                ],
            },
            "sitelinks":  {}
        }
    },
    "success": 1
}
```


Trainer notes

Explain the key-value pairs in the JSON response.

Sample code (Python)

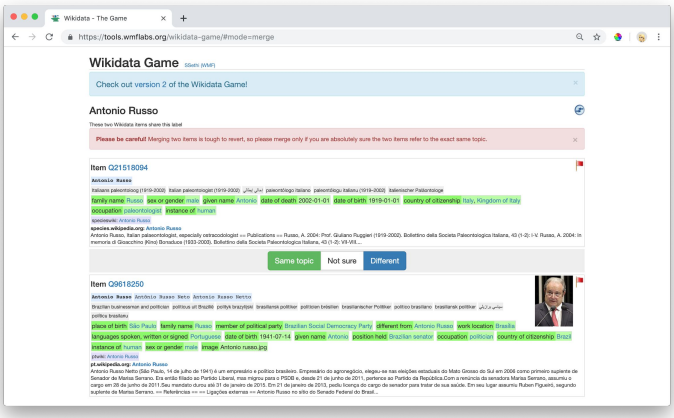
```python
#!/usr/bin/python3

import requests

S = requests.Session()

URL = "https://www.wikidata.org/w/api.php"

ID = "Q7521"

PARAMS = {
    'action':"wbgetentities",
    'ids': ID,
    'format':"json"
}

R = S.get(url=URL, params=PARAMS)
DATA = R.json()

print(DATA)
```

Trainer notes

Show how one can fetch entities via the ‘wbgetentities’ module. Execute the Python script on PAWS: https://paws-public.wmflabs.org/paws-public/User:SSethi_(WMF)/wikidata-sample-code.ipynb

Show how one can access the Wikidata SPARQL endpoint and retrieve the query with python by passing it as a parameter to the GET Request. Execute the Python script on PAWS: https://paws-public.wmflabs.org/paws-public/User:SSethi_(WMF)/humans-of-sweden.ipynb. Discuss when one would prefer using the API over SPARQL query and vice-versa (e.g., for data modifying operations).

Demo App: Wikidata Game

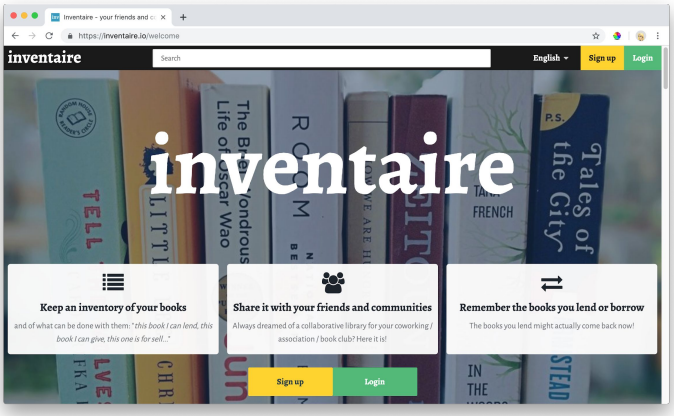


https://tools.wmflabs.org/wikidata-game/#mode=merge

Trainer notes

Briefly explain the demo app, how it works, and API modules it might be using.

Demo App: Inventaire



https://inventaire.io

Trainer notes

Briefly explain the demo app, how it works, and API modules it might be using.

Resources

→ API tutorial
https://www.mediawiki.org/wiki/Wikibase/API
→ List of supported modules
https://www.mediawiki.org/wiki/Wikibase/API#API_documentation_and_Wikibase_modules
→ Sandbox to experiment with the API
https://www.wikidata.org/wiki/Special:ApiSandbox

Activity 2

Use any programming language to create a new item on the test instance of Wikidata:
https://test.wikidata.org/w/api.php (api endpoint)

Hint:
Use the "wbeditentity" module:
→ **https://www.wikidata.org/w/api.php?action=help&modules=wbeditentity**

For exploring this API module further:
Use the same module to create and remove a claim associated with the item you just created.

Trainer notes

Explain a bit about the POST requests first, and then give ~10-15 minutes to participants to get comfortable with making API calls. Before moving on to the next step, demonstrate how one can create an item, a claim, and remove a claim using the 'wbeditentity' module. Execute the Python scripts on PAWS:

https://paws-public.wmflabs.org/paws-public/User:SSethi_(WMF)/wikidata-edit-item.ipynb (edits an item)
https://paws-public.wmflabs.org/paws-public/User:SSethi_(WMF)/wikidata-create-claim.ipynb (creates a claim)
https://paws-public.wmflabs.org/paws-public/User:SSethi_(WMF)/wikidata-remove-claim.ipynb (removes a claim)

Wikimedia Commons API

a database of 55,247,754 freely usable media files to which anyone can contribute

https://commons.wikimedia.org

GET Request

→ get request to fetch files uploaded by a particular user

https://commons.wikimedia.org/w/api.php?action=query&list=allimages&aiuser=Srishti%20Sethi&aisort=timestamp&ailimit=100&format=json

→ documentation
https://commons.wikimedia.org/wiki/Commons:API/MediaWiki

Trainer notes

Explain each parameter in the URL. You could also show a real use case of the GET Request mentioned in the slide above, a photo carousel on a personal website: http://srishtisethi.com/photos.html :)

Trainer notes

Highlight the use of MediaWiki Action API to fetch data from other Wikimedia sites such as Wikisource and Wikiquotes, not covered in this presentation. Share an example of the project "Quote Mashup": http://natetyler.github.io/.

Resource for all Wikimedia APIs (needs documentation improvements):
https://www.mediawiki.org/wiki/Web_APIs_hub

Other useful APIs in the Wikimedia universe:
https://wikitech.wikimedia.org/wiki/Event_Platform/EventStreams
https://wikitech.wikimedia.org/wiki/Analytics/AQS/Pageviews

Credits

Template and content of this toolkit are based on the **Wikimedia APIs at the Wikimedia Game Jam 2015** presentation by **Lucie-Aimée Kaffee** / [[User:Frimelle]].

Special thanks to Lucie for sharing the source of the slides of the presentation!

Thank you!

Presentation:
http://tinyurl.com/wikimedia-apis
Srishti Sethi

numbers as of Aug 06,2019

Contact me!

ssethi@wikimedia.org

@srish_aka_tux