SMW and WikiBase: blasphemy or feasible?

SMWCon2020 | 2020-11-25

CC BY SA

lucamauri.com

Why this presentation

- Hello and good afternoon
  - Who I am
- I am here to tell you a story
  - Why I decided to use WikiBase over Semantic MediaWiki
  - How I implemented WikiBase
  - Finally, how I got back to square one, understanding that, after all, Semantic MediaWiki was more than needed

Where I started

- For our small Mediawiki-based project we needed to be able to organize data and be able to access them in a structured way
  - Originally Wikidata seemed to be the right choice
  - Then I discovered that the only way to access data from our own MediaWiki instance
    - was to install our own WikiBase as well

«The access to Wikidata data is currently restricted to the Wikimedia projects, because of technical limitations. If you have your own instance of MediaWiki, you can't use Wikidata's data using these features. However, you can set up your own Wikibase instance and use data from there in the same way.»

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

How the system was setup

- I went for the same layout as Wikipedia is doing
  - So the setup consisted in two separate MediaWiki instances
  - So one was clearly the data provider
  - And the other one was the consumer
    - The Repository instance is multi-lingual and might potentially become a provider of open-data for other wikis as well

- **Server** instance with
  - Wikibase Repository
  - Wikibase Client

- **Client** instance with
  - Wikibase client
  - Later, Semantic MediaWiki

Wikibase advantages

- After the configuration was sorted-out, getting data from Repo to Client is **easy** both
  - using magic words and
  - using LUA scripting
- Wikibase is widely known because of the popularity of **Wikidata**
  - So chances are the users are **already** familiar with how the system works and how to contribute
- API is enough documented to be used proficiently
  - Same endpoint and similar use to the MW **standard** API
  - It can be handled with **existing** tools as they are, or with minimum modification

The problem

- After using Wikibase on the test environment for a few weeks, a few shortcomings became clear
  - Wikibase is surprisingly lacking in querying capabilities
  - In particular, reverse queries are impossible to get
- Practical example: we work on items representing television episodes and, in each episode, we maintain the couple performer – character
  - While with Wikibase is easy to get the info for the **single** episode
  - It is basically impossible to get a list of episodes and characters starting **from** the performer
- There are proposals to solve the issue by implementing a function `mw.wikibase.entity:getBacklinks` as explained here https://phabricator.wikimedia.org/T185313
  - But the discussion is at an early stage and it is unclear when it will be implemented, if ever

Enter Semantic MediaWiki

- I needed to completely change perspective to overcome the issue that seemed unsolvable **within** Wikibase boundaries
  - I went back considering Semantic MediaWiki that I originally discarded in my evaluation as it seemed less structured than WB for my purpose
- The problem was now to decide if to throw away all the job done with Wikibase and to start from **scratch** in SMW
  - With the uncertainty to discover some drawback on SMW side as well
- But were my assumptions right?

Do I really needed to choose?

- Then the thought struck me I really didn't need to **choose** between to two things
- Wikibase and SMW are often listed as **alternative** tools to obtain the same result
  - Add structured information to a MediaWiki content
- But, on second thought, this is **not** actually the case
- While you can use them for **similar** purpose
  - One act as a database
  - The other one add semantic meaning to the data themselves *within* pages

The Best of Both Worlds

- So the idea was to **not** choose between one and the other, but rather think on how making them work together, better than alone
- The basic plan was to
  - Still **store** the relevant data in Wikibase
  - Make the appropriate modifications to templates and LUA modules to use the **data** into pages and, at the same time, add **properties** that are meaningful to SMW in order to reuse them with its instruments
- The data can be shown to the reader in a human-readable form and, at the same **time**, they can be interpreted by SMW
  - but this is not a burden to the contributor
  - the user only add data in Wikibase, everything else is (or will be) automatic

How the plan looks like

Structure

- The first step was to make the Wikibase properties meaningful to SMW
  - This should **not** be done manually to
    - simplify the management of the properties in the two places
    - Reduce manual work
    - avoid human error

Content

- Users can add their own SMW annotation manually, of course, if and where necessary, but
- Wherever possible, LUA scripting should insert them
  - By taking not only **data** from Wikibase, but **properties** too

Structure

- Custom Property page should contain, as a bare minimum, a declaration of the type
  - To automate the task, the related property in Wikibase, in turn, has another property indicating the SMW type
  - As shown on the right, as an example

Original airdate (P2)

No description defined

In more languages

Configure

Language	Label	Description
English	Original airdate	No description defined
Italian	Data di trasmissione originale	No description defined

Data type

Point in time

Statements

SMWHasType	Date 0 references

Structure – code behind

- The function shown here is a simple constructor of `[[Has type::`
  - getting the information from the related Property on wikibase

```
function p.TypeFromDT(frame)
        local Item
        local Type

        Item = mw.wikibase.getEntity()

        [some error checking]

        if (not Item['claims']) or (not Item['claims']['P49']) then
                return "ERROR"
        else
                Type = Item['claims']['P49'][1].mainsnak.datavalue.value
                return "[[Has type::" .. Type .. "|''" .. Type .. "'']]"
        end
end
```

Content

- Within the LUA functions used to generate content, I implemented a **option** telling if the output should be produced to comply with SMW syntax
  - That allows to **selectively** decide where to use SMW or not to use when is not necessary
  - In the best-case scenario, all elements that strongly rely on data are SMW **enabled**
  - For instance all the data within the InfoBoxes are…

Content - example

- taken from WikiBase
- presented in graphical and user-readable way to the reader
- Finally they includes properties labels, so they are interpreted by SMW and they can be queried on

Numero di produzione: 305

Date e emittenti della prima TV: giovedì 12 novembre 2020 su *CBS All Access* (USA) · venerdì 13 novembre 2020 su *Netflix* (Resto del Mondo)

Durata: 56min

Titolo italiano: Provarci fino alla morte

Personaggi e interpreti DT

Attori co-protagonisti

- Tenente Keyla Detmer: Emily Coutts
- Tenente Gen Rhys: Patrick Kwok-Choon
- Tenente Joann Owosekun: Oyin Oladejo
- Tenente R.A. Bryce: Ronnie Rowe Jr.
- Tenente Nilsson: Sara Mitich
- Eli: Brendan Beiser
- Dottor Attis: Jake Epstein
- Tenente Audrey Willa: Vanessa Jackson
- Mamma Barzani: Ana Sani
- Figlia Barzani 1: Ava McKinnon
- Figlia Barzani 2: Shazdeh Kapadia

Attori ospiti

- Ammiraglio Charles Vance: Oded Fehr
- Adira Tal: Blu del Barrio
- Kovich: David Cronenberg
- Jett Reno: Tig Notaro

Attori in ruoli non accreditati

- Linus: David Benjamin Tomlinson

Attori ospiti straordinari

- Philippa Georgiou (specchio): Michelle Yeoh

Navigatore episodi

< Precedente	Successivo >
Non ti scordar di me	Sfruttatori

Modifica i dati nella pagina della entità su *DataTrek*

Content – code behind

- The fragment of code shown here takes data from the Wikibase property and then prepare it for publishing into wiki page
  - at the same time it adds SMW property

```
for _, Statement in pairs(Statements) do
        local Result
        local DateLabel
        if AddSemantic then
            DateLabel = "[[Data di trasmissione::" ..
Statement['mainsnak'].datavalue['value'].time .. "|" ..
frame:expandTemplate{ title = 'TimeL', args = {Tipo='ITEstesa',
Istante=Statement['mainsnak'].datavalue['value'].time} } .. "]]"
        else
            DateLabel = frame:expandTemplate{ title = 'TimeL',
args = {Tipo='ITEstesa',
Istante=Statement['mainsnak'].datavalue['value'].time} }
        end


    Result = '<li>' .. DateLabel .. " su '''" ..
Statement['qualifiers']['P4'][1].datavalue['value'] .. "''' (" ..
Statement['qualifiers']['P34'][1].datavalue['value'] .. ")</li>"
    Results[#Results + 1] = Result
end
```

Still lot to do

- More complete `SMWProperty` template
  - Automate the data gathering by pairing name of the property with the P item in Wikibase
  - To get more information from Wikibase
- To reduce manual work
  - Automatically relate Wikibase properties to SMW types
  - Automatically add the proper SMW property according to the upstream Wikibase one

Thanks, your turn now

CC BY SA