Automated citations in Wikipedia: Citoid and the technology behind it

Sebastian Karcher

Wikimedia Tech Talk, February 29, 2016

Citoid is based on Zotero

zotero

- Reference manager developed out of GMU
- Mozilla based
- Free and open source

Setting up your development environment

- Firefox: https://www.mozilla.org/en-US/firefox/new/
- Zotero: https://www.zotero.org/download/
- Scaffold:
  https://www.zotero.org/support/dev/translators/scaffold

Zotero translators

- Zotero translator: individual file, written in javascript
- Four types of translators: Web, Import, Search, Export
- Some web translators, like the JSTOR one, call on an import translators (e.g. RIS)
- Other web translators “scrape” data from the page
- Full list https://github.com/zotero/translators/

Tools: Xpaths are “directions” on a website

- xpaths are basically “directions” to a part of a webpage
- A webpage is built up from a number of nested nodes
- A super-simple webpage:

```
<html>
  <head>
    <title>A Basic Webpage</title>
  </head>
  <body>
    <div itemprop="headline">Title</div>
    <div class="text">Content</div>
  </body>
</html>
```

The most basic xpath

- Give directions: at every corner/node, tell Zotero where to go:
- We want to go go to “The title of the webpage”
- “Take the HTLM road, take a left at ‘body’, then take the ‘div’ street, or in Xpath:

```
/html/body/div
```

Making xpaths more precise

- But we're still "lost" - which of the two "div" streets do we go down?
- Option 1: Take the first `<div>`

  ```
  /html/body/div[1]
  ```

- Option 2: Take the `<div>` that has "headline" as an itemprop

  ```
  /html/body/div[@itemprop="headline"]
  ```

Making xpaths more efficient

- Full xpath can be *very* long, so we'd like to shorten them.
- Use // to start anywhere in html tree, e.g "the `<div>` with 'headline' as an 'itemprop' anywhere on the site":

```
//div[@itemprop="headline"]
```

- Match only part of an attribute using contains() as in

```
//div[contains(@itemprop, "head")]
```

Basic translators structure

Always:

- Target (URL)
- detectWeb(doc, url) – is there something to translate?
- doWeb(do, url) – run the translator

Normally:

- scrape (does the actual work)
- getSearchResults (check for multiples)

Use existing structure

- Common code from experienced dev: https://github.com/zuphilip/translators/wiki/Common-code-blocks-for-translators
- Existing translators
  - Scrape: e.g. FAZ.NET.js
  - Call Metadata: e.g. PLoS Journals.js

Putting this to work

- Página12 (pure scrape)
- El País (call metadata, then enhance)

Getting help

- Documentation: https://www.zotero.org/support/dev/translators/coding
- Help on xpaths: http://archive.oreilly.com/pub/a/perl/excerpts/system-admin-with-perl/ten-minute-xpath-utorial.html (mostly xml)
- javascript: you don't need much. Codeacademy's javascript course is good, e.g.
- regular expressions "regex": e.g. http://regexone.com/ is one option. There are many...
- Zotero developer group: https://groups.google.com/forum/#!forum/zotero-dev (make your code available)
- Or just submit your code as a pull request to https://github.com/zotero/translators/ – we'll work with you