Traverse complex JSON structures with live feedback

200ok GmbH

Alain M. Lafon, 2020-11-29

alain@200ok.ch

Outline

Introduction

About me

Alain M. Lafon

CV

- CEO 200ok GmbH (200ok.ch)
- Zen Monk, abbot of Lambda Zen Temple (zen-temple.net)

Contact
alain@200ok.ch

Proposition

- Most work on the computer is based on either text processing or text consumption.
- Often, the text that needs to be processed, is in a structured format, such as JSON, even if your job is not programming per se.
- Reading through a bigger chunk of JSON is non-trivial, but essential to get an understanding of the data.

Proposition

Here is an example from the Github API:

`github_issue_comment.json`

Let us quickly check that out.

Prior art

Processing JSON

> *jq is like sed for JSON data - you can use it to slice and filter and map and transform structured data with the same ease that sed, awk, grep and friends let you play with text.*

Quoted USP from `https://stedolan.github.io/jq/`

Demo

jq

- jq is written in portable C, and it has zero runtime dependencies.
- Let's explore an example JSON file with it.

jq for JSON exploration

Ok, so let’s go to the shell and check out what keys are available on the top level:

```
jq 'keys' github_issue_comment.json
```

```
[
  "action",
  "comment",
  "issue",
  "organization",
  "repository",
  "sender"
]
```

jq for JSON exploration

Let's find more information about that issue:

```
jq '.issue' github_issue_comment.json
```

```
{
  "url": "https://api.github.com/repos/200ok-ch/organi
  "repository_url": "https://api.github.com/repos/200o
  "labels_url": "https://api.github.com/repos/200ok-ch
  "comments_url": "https://api.github.com/repos/200ok-
  "events_url": "https://api.github.com/repos/200ok-ch
  "html_url": "https://github.com/200ok-ch/organice/pu
...
```

jq for JSON exploration

This is where it gets tedious:

- The output can be humongous whilst a shell is not the best place to read through big output (eshell being one of the better shells for this, because pretty much 'just' a regular Emacs buffer).
- The command needs to be repeated and repeated and repeated until finally the right query is built.

It would be so much nicer to have live feedback. When working with Emacs, we're quite used to that, so there should be an option!

Prior art

Completion in Emacs

> *Ivy is a generic completion mechanism for Emacs. While it operates similarly to other completion schemes such as icomplete-mode, Ivy aims to be more efficient, smaller, simpler, and smoother to use yet highly customizable.*

Quoted USP from `https://github.com/abo-abo/swiper`

counsel-jq for JSON exploration

- counsel-jq is a package with which you can quickly test queries and traverse a complex JSON structure whilst having live feedback.
- Just call M-x counsel-jq in a buffer containing JSON, then start writing your jq query string and see the output appear live in the message area.
- Whenever you're happy, hit RET and the results will be displayed to you in the buffer jq-json.

References

Links

- jq: `https://stedolan.github.io/jq/`
- Ivy: `https://github.com/abo-abo/swiper`
- counsel-jq:
  - `https://melpa.org/#/counsel-jq`
  - `https://github.com/200ok-ch/counsel-jq`
- organice (Org mode for mobile and desktop browsers): `https://github.com/200ok-ch/organice`