2021 - All about COOL days Collabora Online

Recent performance improvements for Collabora Office and Online

Tor Lillqvist

Software Engineer Collabora Online

@TorLillqvist

Profiling: the old tooling

- The so-called ProfileZone class (in core)
  - Logged one line when object was constructed and one when it was destructed (typically at the end of the scope of a local ProfileZone variable)
  - Output was a home-grown textual format
  - We had some Perl script to manipulate it

Desire for better profile tooling

- Desirable to be able to use existing viewer(s) for a suitable standardised format
- Choice: Google’s Trace Event format, as used by Chrome/ium
- Human-readable text (JSON)
- Public specification from Google, stability unclear
- Chrome/ium also acts as a viewer for files in that format

Modifications to ProfileZone

- Make it output Trace Event format data
- Re-factor to enable other types of data in the Trace Event specification to be generated, too
- Eventually we noticed that the Trace Event viewer in Chrome/ium doesn’t actually support all the types described in the specification

ProfileZone for Online, too

- Write a class with the same name and same use cases for Online, too
  - Could not re-use the core ProfileZone code as it uses LibreOffice-specific types like OUString etc
  - It was not that much code anyway so easier to re-implement using std::string etc

Trace Event plumbing in Online

- It is the WSD process that actually opens the Trace Event output file and writes to it
- But most of the interesting data would be generated in LibreOffice core, or in the Kit process code
- Send collected Trace Event data from core in a callback to Kit (the same process), and from Kit with with a WebSocket message to WSD

ProfileZones in JavaScript, too

- No scope-based construction and destruction, so ProfileZone API different: Need to explicitly call a function to finish the event
- Data sent to WSD process as a WebSocket message

Trace Event results

- There are already some improvements based on results from the generation of these Trace Events
  - Message handling in the JavaScript greatly improved by so-called slurping, a kind of buffering
- In many cases, though, functions that were expected to be performance heavy turned out to show up as extremely short duration events

JavaScript improvements

- Large speedups of the JavaScript thanks to fixing bottlenecks noticed with other tools
- Doing things in JavaScript the way we were doing caused much slowness
- Be more clever with buffering of messages
- Avoid lots of string re-creation by appending one character at a time

JavaScript improvements

- Be more clever in reacting to messages from the server
  - If the server sends us repeated essentially identical messages, don’t mutate the DOM, i.e. destroy and re-create HTML the exact same elements
  - Don’t mutate the DOM immediately but wait until all buffered (“slurped”) messages have been seen, only then make the page look how it should according to accumulated messages

JavaScript improvements

- Don’t use external JavaScript libraries of questionable performance
  - Our use of the select2 library caused 800 ms delay when loading the document editor

2021 - All about COOL days Collabora Online

Thanks!

Collabora Online

By Tor Lillqvist

@CollaboraOffice
hello@collaboraoffice.com
Collaboraoffice.com