What I need from IntelliJ and what I deeply miss when I'm not using Emacs

Dr. Arne Babenhauserheide

<2019-10-28 Mo>

At work I'm using IntelliJ for Java development, but I'm not happy with the interface. It forces me out of my concentration and regularly breaks my flow by having stuff jump around and stealing focus.

But I cannot switch to something that works better for me, because there are features of IntelliJ that I require to work efficiently.

Update: I also need ëxtract interface and replace all usageänd data flow analysis.

What I really need from IntelliJ

- inspection
  - Where is this called? — all callers
  - Where is this implemented? Where is it declared? Or overridden?
  - Visual indicator whether a method is overridden or whether it overrides
  - Where is this defined (base method or concrete method)?
  - Which Symbol with "these letters" is available for use here? Add imports as necessary.
- refactor
  - rename symbol,
  - change signature (with base method and overrides and callers),

  - extract method from selection,
  - extract variable / store selected expression in variable
  - extract interface and replace all possible usages of the class with the interface.

- run
  - Run tests in changed modules or in file
  - re-run test, restart current program
  - re-build incrementally
  - hot-swap without restart

- debug
  - set breakpoint and see breakpoints, set conditional breakpoint
  - run project via ~~eclipse run config~~ main method (we replaced the eclipse stuff extracted with Eclipser by main methods)
  - inspect stack and state at break point
  - step over / in / out / continue
  - Where does the value from this variable come from (data flow analysis)

- other
  - Jump to definition / caller (also with mouse CTRL-click), even for xml, so colleagues can do it when working at the same box
  - show all methods in file
  - VCS: ignore changes in some files
  - run Sonar Qube on changed files

What I deeply miss when not using Emacs

- keyboard shortcuts
  - mnemnonic keybindings: When I type C-x r t, I thing x-rectacngle-text. That is why it works accross different keyboard layouts.

  - staying on the letter row.
  - ad-hoc shortcuts limited to the current file with `local-bind-key`.

- editing
  - killing to the end of the line with C-k (I actually added that to IntelliJ now)
  - cycling through the cut-paste list with M-y: Often I don't need the last kill, but the one before. Yes, I can reach for the mouse and use klipper, but that slows me down and breaks my concentration. ——— **C-S-v in IntelliJ uses paste from history**
  - storing and retrieving multiple values with registers.
  - Completion which replaces the suffic, or at least M-d (Alt-d): kill world or rest of word. ——— **You can remap Alt-d in IntelliJ to kill to word end**
  - Activate selection mark, navigate, kill all code in-between mark and current point. The Emacs live plugin is close, but not good enough.

- windows
  - Commands with M-x, fuzzy matched, and without settings-window-names getting in the way. I half-ways replace that with C-S-a
  - closing other windows with x1 (actual "x1thanks to key-chords-mode). Deeper: Natural use of multiple windows.
  - storing a window configuration in a register and retrieving it later
  - Truly having two windows side-by-side with two points and switching with xo or xö (C-x o).

- files
  - Fuzzy matching in buffer-list with bf (as chord or with C-x b).

- interop
  - Linking to code files from my org-mode planning file.

- movement

  - dumb-jump to test
  - Navigation with C-n / C-p / M-b / M-f. That avoids having to move to the arrow keys.
  - back to last edit which stays in the buffer. I can switch between buffers with bf, and after I just want to go back to where I last edited this buffer. Multi-file back-to-last-change is also nice (as IntelliJ provides it), but it's not complete.

- Feeling fast
  - Somehow all the things I need to do in IntelliJ feel slow. Maybe that's because a million lines of code is a lot. Maybe because it keeps a huge amount of state. Or because Maven is slow. But it feels like I'm regularly waiting for something to refresh itself.
  - IntelliJ feels slow, because it often opens dialogs before they accept keyboard input. To reproduce: start a global search (with CTRL-shift-F) and start typing. It misses my first keystrokes. Emacs takes all keystrokes.

- other
  - Having the shell just an M-! away, in the same folder as the code file.
  - ripgrep
  - A colleague said today "I wish we had tabs grouped by type". I could not suppress saying "emacs does this — with tabbar-mode".
  - Inline merge-conflict highlighting (I actually switch from IntelliJ to Emacs for that).
  - glasses-mode to highlight capital letters in camelCase.