Pandoc to Orgmode with Babel

Rohit Goswami · 2020-05-02 · tools · emacs · workflow · orgmode

Background

One of the best things about writing in `orgmode` is that we can embed and execute arbitrary code snippets. However, not all languages have an exporter, for obvious reasons. Somewhat surprisingly, there is no way to call pandoc on embedded snippets, which feels like a waste, especially when a whole bunch of documentation formats can be converted to `orgmode` with it.

Consider the following beautifully highlighted snippet of an `rst` (ReStructured Text) list table.

```
.. list-table:: Title
   :widths: 25 25 50
   :header-rows: 1

   * - Heading row 1, column 1
     - Heading row 1, column 2
     - Heading row 1, column 3
   * - Row 1, column 1
     -
     - Row 1, column 3
   * - Row 2, column 1
     - Row 2, column 2
   - Row 2, column 3
```

Trying to run this will generate the sort of obvious error:

```
org-babel-execute-src-block: No org-babel-execute function for rst!
```

Writing an Exporter

For this post, I will be focusing on `rst`, but this can be defined for any of the `pandoc` back-ends. The approach was inspired by ob-markdown.

```
(defun org-babel-execute:rst (body params)
  "Execute a block of rst code with org-babel.
This function is called by `org-babel-execute-src-block'."
  (let* ((result-params (split-string (or (cdr (assoc :results params)) "")))
         (in-file (org-babel-temp-file "rst-"))
         (cmdline (cdr (assoc :cmdline params)))
         (to (cdr (assoc :to params)))
         (template (cdr (assoc :template params)))
         (cmd (concat "pandoc"
                      " -t  org"
                      " -i " (org-babel-process-file-name in-file)
                      " -f rst "
                      " " cmdline)))
    (with-temp-file in-file (insert body))
    (message cmd)
    (shell-command-to-string cmd))) ;; Send to results

(defun org-babel-prep-session:rst (session params)
  "Return an error because rst does not support sessions."
  (error "rst does not support sessions"))
```

Trying it out

With that done, it is pretty trivial to re-run the above example.

```
.. list-table:: Title
   :widths: 25 25 50
   :header-rows: 1

   * - Heading row 1, column 1
     - Heading row 1, column 2
     - Heading row 1, column 3
   * - Row 1, column 1
     -
     - Row 1, column 3
   * - Row 2, column 1
     - Row 2, column 2
   - Row 2, column 3
```

Heading row 1, column 1	Heading row 1, column 2	Heading row 1, column 3
Row 1, column 1		Row 1, column 3
Row 2, column 1	Row 2, column 2	

Note that we have used `rst :exports both :results raw` as the header argument.

Conclusions

Will probably follow this up with an actual package, which should handle the entire spectrum of `pandoc` back-ends.