International
Virtual
Observatory
Alliance

Using Markdown for IVOA standards

Version 1.0-201506xx

IVOA Note 2016-09-02

Working group

This version
http://www.ivoa.net/documents/IvoaMarkdown/
Latest version
http://www.ivoa.net/documents/IvoaMarkdown
Previous versions
Version xx
Author(s)
Omar Laurino
Editor(s)
Omar Laurino

Abstract

Markdown is a lightweight markup language. This note introduces a set of `pandoc` templates and filters meant to make writing IVOA standards in Markdown possible in a convienient, portable, versionable way.

Status of This Document

This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification.

A list of current IVOA Recommendations and other technical documents can be found at [http://www.ivoa.net/Documents/].

Contents

1 Goals

This project aims at providing IVOA authors and editors with a versatile tool to edit and collaborate on IVOA documents.

The goals could be summarized as follows:

- enable documents to be properly versioned and authored concurrently;
- adopt a lightweight markup language, thus focusing on the content rather than on the presentation;
- require as few dependencies as possible;
- be cross-platform;
- produce high-quality typesetting renderings of the same master document in different formats.

NOTE: Renderings of this document produced with this project are available in the output folder.

2 Comparison with IVOATeX

This project shares the goals of IVOATeX (Demleitner 2016) and reuses its templates as much as possible. However, it significantly departs from IVOATeX in that it does not require LaTeX to be used to author documents. In `cereal` LaTeX is an optional dependency required by editors in order to produce a `PDF` version of the document.

Also, `cereal` is designed to be cross-platform and easy to set up with as few dependencies as possible.

A potential solution could have been to create `pandoc` templates that would render Markdown as IVOATeX LaTeX files to be compiled with IVOATeX itself.

However, this would have not removed the hard dependency on LaTeX and increased the number of dependencies required to produce documents, which is already significant in IVOATeX.

On the other hand, IVOATeX is probably more complete and far more tested that `cereal`, at the time of this writing.

3 Dependencies

In order to edit files you only need a text editor and whatever source control system is required by your collaboration.

In order to build `html` documents only `pandoc` is required. It is also recommended to install the `pandoc cross-ref` filter.

`Pandoc` is open source and available for Linux, OSX, and Windows.

The `cross-ref` filter is available as source code and can be built from sources. For Windows, pre-compiled binaries are available.

More advanced filters require a `Python` interpreter to run, along with the `pandocfilters` package. These filters allow to:

- include external source files.
- eventually, to process special elements like todo lists, but this is not yet implemented.
- Python filters could be created specifically for integrating documents with specific tools, for instance for creating `graphviz` diagrams from or into document elements.

Finally, if a `LaTeX` distribution is installed, `PDF` documents can be produced.

NOTE: On Windows, it should be possible to install all dependencies as native packages with regular installers. It is also possible to build many if not all the dependencies from sources, if desired.

The project is currently in an early development stage, so the user interface is simple and not necessarily pretty. That will come with time.

4 Quick start

The very first goal of this project is to allow proper versioning of standard documents. In the following instructions we will thus assume that a versioning system is used. `Git` or `Subversion` are recommended. It should be

easy to adapt these instructions to cases where a versioning system is not available. Since `cereal` is hosted on GitHub, one can download tarballs at any revision directly from the GitHub pages.

With both systems, the general idea is to create a project for the document to be written, and then include the rendering framework as a dependency, either as a `git submodule` or as an `svn external`.

Since `cereal` is hosted on GitHub both `git` and `svn` can be used as clients.

4.1 Git

The following instructions, valid for both *nix and Windows systems, will create a new folder, initialize an empty `git` repository in it, and then clone the `cereal` project as a submodule.

```
1 mydoc
2 cd mydoc
3 git init
4 git commit --allow-empty -m "init commit"
5 git submodule add https://github.com/olaurino/cereal cereal
```

A submodule is a pointer to a different repository. It is cloned in the user's working directory and can be used as any other local directory. However, `git` internally simply refers to it with the commit ID of the current `HEAD`.

`Git` is a distributed system, so you can start your project locally and then publish it to a public or private server where it can be forked and shared.

4.2 Subversion

The following instructions, valid for both *nix and Windows systems (make sure you change the paths, though), will create a new folder, initialize an empty `svn` repository in it, and then include the `cereal` project as an external.

Since `svn` is not distributed, if you want the repository to reside on a server you will need to create the repo on that server first, and then checkout a working directory. Alternatively you can `relocate` or `switch`. For simplicity, below we create the repository locally and we checkout the working directory in a different folder.

```
1 svnadmin create mydoc_repo
2 svn co file:///C:\Documents\mydoc_repo mydoc
3 cd mydoc
4 svn propset svn:externals "cereal https://github.com/olaurino/cereal/trunk" .
5 svn update
```

4.3 Hello World

Open your favorite text editor and type the following text:

```
Hello World
===========

This is my first Markdown document[^1] written in my favorite text editor.

[^1]: powered by [`cereal`](https://github.com/olaurino/cereal).
```

Save this file as `content.md` in the project's folder. If you are using an editor that renders Markdown, or if you have a browser extension that does the same, you can see that the `Hello World` text is being rendered as a header, while the page ends with a link to the `cereal` repository on GitHub.

The footnote is recognizable as such, but since it is a `pandoc`-specific feature, it is not translated to appropriate `html`.

By design, Markdown and its extensions focus on readability by the human eye. Even without any rendering, the above text makes sense.

This is important because unless you are a document editor, you don't need any special requirements installed in order to contribute on a document.

The following section shows how, with `pandoc` installed, one can properly render specific extensions using `cereal`.

4.4 Hello Pandoc

Assuming you have `pandoc` installed and available from the command line, you can run the following command (showing both Windows and *nix versions):

```
.\cereal\bin\simple.bat .\content.md
./cereal/bin/simple.sh content.md
```

The rendered `html` will be produced in the `output` folder as `simple.html`.

You can open it with any web browser. You should see several differences now.

First, there is some IVOA branding. There is also a very simple table of contents. The footnote is now rendered as a couple of links referring to each other.

This result can be achieved by simply installing `pandoc`, which is available on most platforms: on Windows, it is provided as a regular application installer.

The IVOA branding is not particularly rich. We need to provide `cereal` with some metadata in order to fill some information in.

4.5 Metadata block

Create a new file `metadata.yaml` with the following content:

```
---
title: My First Document with `Cereal`
name: FirstDoc
version: '1.0'
# supported types: WD, PR, NOTE, REC, DERR
# custom types may be provided by creating files
# in `style/messages`, following the examples.
type: NOTE
date: '2016-09-13'
datecode: '20160913'
author:
    - name: Omar Laurino
      url: http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/OmarLaurino
editor:
    - name: Omar Laurino
group: Data Models
previousVersion:
    - ver: xx
      url: http://www.ivoa.net/documents/xx

abstract: |
    My very first Markdown document rendered as an IVOA standard document by
    `cereal`.

references:
  - id: ivoatex
    type: article
    author:
      - family: Demleitner
        given: Markus
    title: 'The IVOATEX Document Preparation System'
    issued:
      year: 2016
      month: 04
      day: 30
    URL: 'http://www.ivoa.net/documents/Notes/IVOATeX/index.html'
link-citations: true
---
```

Now, run again the `simple` script, adding the metadata file to the command line:

```
.\cereal\bin\simple.bat .\metadata.yaml .\content.md
./cereal/bin/simple.sh metadata.yaml content.md
```

Refresh the `html` page and notice how the metadata is used to render a familiar IVOA front page. Notice that the `Status of this document` section is automatically filled with the correct description for the kind of document you specified.

If you changed `type: NOTE` to `type: WD` you would get a different message.

Remember that unless you are using an editor that automatically updates the `html` rendering, you have to run the `simple.sh|bat` script to see the document change.

Did you notice that there is a reference defined in the metadata header? You can use the ID to refer to it in your document. For instance, add anywhere in your `content.md` file the following text:

```
Can you do this with IVOATeX [@ivoatex]? Yes, you can, if you prefer LaTeX over
Markdown!
```

Process the file with the `simple` script again: the citation is properly and automatically rendered as you would expect in a LaTeX document. The reference itself is also added at the add of the document.

NOTE: `pandoc` supports the usual LaTeX bibliography files as well. It also provide great control over the way citations and references are rendered. However, this is outside of the scope of this document. Refer to the `pandoc` documentation for more details.

The `simple` script only requires `pandoc` and as such it is very portable.

However, this script does not allow you to render some advanced features. You can still author a document, but you won't see some special constructs properly rendered.

For instance, modify `content.md` to look like this:

````
Hello World
===========

This is my first Markdown document[^1] written in my favorite text editor.

[^1]: powered by [`cereal`](https://github.com/olaurino/cereal).

Can you do this with IVOATeX [@ivoatex]? Yes, you can, if you prefer LaTeX over
Markdown!

~~~~ {include=sample.xml .xml .numberLines}
An xml snippet with a list of books
~~~~

References
==========
````

Then, create a file named `sample.xml` with the following content:

```
<xml>
  <book>
    <title>The Hitchhiker's Guide To The Galaxy</title>
    <author>Douglas Adams</author>
  </book>
</xml>
```

Now, run the `simple.bat|sh` script again, and reload the page. Unimpressively, only the placeholder text is displayed.

4.6 Hello Python

If Python is available on your system, you can install the `pandocfilters` package[1].

Then, you can run the `full.sh|bat` script just as you did with the `simple` script (you also need to have the `pandoc-crossref` add-on installed):

[1] We include some instructions on how to install Python packages in sec. **??**, especially for Windows users. Linux and OSX users probably won't need any instructions.

```
.\cereal\bin\full.bat .\metadata.yaml .\content.md
./cereal/bin/full.sh metadata.yaml content.md
```

Now, you should see the `xml` snippet rather than the placeholder.

4.7 Hello PDF

If you have a LaTeX distribution installed, then you can run the `pdf.sh|bat` script to render the current document:

```
.\cereal\bin\pdf.bat .\metadata.yaml .\content.md
./cereal/bin/pdf.sh metadata.yaml content.md
```

The output will be printed to `output/full.pdf`.

It is important to stress that the production of a PDF is the terminal step, and that LaTeX is not required if not at the end of the pipeline.

As such, LaTeX can be used by editors rather than by authors, and, unlike IVOATeX, it is not required to produce `html` renderings.

References

Demleitner, Markus. 2016. "The IVOATEX Document Preparation System." http://www.ivoa.net/documents/Notes/IVOATex/index.html.