FOTL: Plaintext Editing and Collaboration

Release 1.0

Andrew Harrington, Konstantin Läufer, George K. Thiruvathukal

August 16, 2012

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Abstract

Creating Open Courseware Using Free and Open Source (FOSS) Publishing Tools

George K. Thiruvathukal, Andrew Harrington, and Konstantin Läufer: Department of Computer Science

The presenters will provide a hands-on tutorial for how to organize course materials using the methods created by the FOSS community. We will cover an emerging class of documentation tools that are based entirely on simple plaintext authoring and look at simple authoring languages. We'll demonstrate that although these tools were created for software developers, they in fact are general enough to support writing across all disciplines (e.g. humanities, arts, social sciences, life/physical sciences, and mathematics) and are straightforward to learn. These tools also facilitate version control and shared repositories for works in progress with multiple authors.

1.2 Topics

- Plain text authoring
  - Rationale for using plain text authoring tools
  - The tool we use: Sphinx
  - Example (this talk!)
- Collaboration tools
  - Version control (Mercurial is our example)
  - Free online repositories for joint work (Bitbucket)

1.3 History of Plaintext Authoring

Plaintext authoring has a long history. Before there were word processors, there were much simpler systems that were focused only on text, and formatted as a second step. These systems are very much still with us today.

- text formatting programs such as nroff (and groff)
- generalized markup (HTML, XML, and SGML)
- typesetting software: TeX, LaTeX
- emerging alternatives to markup: reStructuredText and Markdown

The emerging alternatives are the subject of this demo and tutorial.

1.4 Rationale

Plaintext editing represents a rather ironic direction for text editing.

Word processors were supposed to simplify writing. There are a couple of issues with them that have developed over time:

- Word processors (e.g. Word et al) became more about features than the core task of writing.
- "Simple" editing is generally interpreted as WYSIWYG (What you see is what you get).

WYSIWYG may be OK if you are only looking to generate a document in a single format, but if you want to generate multiple formats simultaneously (html, pdf, epub for ereaders, ...), it is important to separate

- the content and the classifications of parts
- the way they end up being displayed in (one of several) final products

For instance with a WYSIWYG editor you do not necessarily use the right abstractions: you may use concrete styles (e.g., italics, large bold) instead of abstract uses (e.g., emphasis, section heading).

In the end, we're not proposing that folks don't use word processors, but we will focus on what is *gained* by keeping the final product in plain text.

For us, there are many advantages:

- lightweight
- easy to keep in a version control system and collaborate
- no licensing fees
- automated indexing and cross-referencing
- supports transformation to a myriad of formats–including Microsoft .docx itself!
- supports proper targeting of different devices (desktop, web, mobile, e-readers) with nearly zero effort!

The list goes on...

CHAPTER

TWO

RESTRUCTUREDTEXT AND SPHINX

Your content has parts with different uses in the text: titles, paragraphs, emphasized text, URL's with hyperlinks, index references. *Markup* is added to basic text content to identify these specific functions. It is better to identify logical function than the final appearance: The formatting for a mobile device is quite different than for a regular large html web page, which is different than the formatting for a page-oriented pdf file.

In a WYSIWYG editor like MS-Word, the markup is largely hidden from you, and you mostly look at a version formatted for a page.

If you are going to generate multiple final forms it is easier to see the functional markup directly. The trick is to keep the plain text concise and easily readable, so as to not obscure the underlying text your are editing.

In the early days of the web there were no WYSIWYG editors, and html was edited in its raw form. Html markup is specifically designed for web pages and can get rather verbose.

Later markup formats were structured text, and then its successor restructured text for single files, and then Sphinx extends restructured text to handle multiple file organization with indexing and cross-referencing.

There are other markup systems, like *markdown*, but here we cover the system we used.

In the html version of the output of this document, you should see the link for **source** at the top. We use this document to show off some of the basic source markup.

The only markup above in this page has been

- the main section heading
- paragraphs
- italics and boldface
- and now this bulleted list

Look at the **source** to see the markup (link in the left frame of html output):

- The chapter heading is set off with a separate line of equal signs.
- Paragraphs are left justified and separated by a blank line. They are wrapped in the final formatting.
- Italics is bracketed in *single asterisks*. Boldface is bracketed in **double asterisks**.
- Bulleted lists look close to normal. See that the continuation lines like this one is indented to line up with the bullet text, not the bullet.

All of this markup is very easy to recognize and enter in the plaintext version, though it looks better in the final formatted version.

Using chapter headings for small parts of this presentation is overkill, and wastes space in the pdf version, but we get to see the format.

2.1 This is a subsection

I chose dashes to indicate subsections. You need to use a consistent symbol, but there are many symbol characters that you can use to mark sub-sections.

2.2 This is another subsection

And inside subsections we can have sub-sub-sections, etc.

2.2.1 This is a sub-sub-section

I chose tildas to identify sub-sub-sections.

> This is a set-off block of text. See the blank lines surrounding a block with *consistent* indentations, further than in the surrounding section.
>
> There may be several paragraphs in a set-off block.

From the sub-sub-section back to a sub-section:

2.3 Fancier Markup

There are limits to this fairly obvious spatial markup. To mark a wider diversity of components, some explicit labels are needed. They come in two basic forms labeling

- outside a paragraph, on a line starting with ".. "
- inside the flow of a paragraph

2.3.1 Indexing

One of our favorite additions made by Sphinx is easy indexing.

On a set off line by itself:

```
.. index::  index entry; simple
```

makes an index entry. Here the primary index entry is "index entry", and the secondary entry (after the semicolon) is "simple".

Often we want to mark a place with two related idea, and want two index entries, reversing which is primary. You can do both simultaneously with the markup:

```
.. index::
   double: double entry; index entry
```

Look at the index and check.

2.3.2 Footnotes

Here we have a footnote. [1]

Automatically numbered footnotes have a marking in place in the source paragraph text like

```
[#doubleColon]_
```

and at the end of the section set off footnote text like

```
.. [#doubleColon]
   If you look at the source, you see that we are using a
   double colon :: before the set off text to make it literal.
```

The original footnote reference inside the paragraph is bracketed by `[#` and `]_`. This follow the basic pattern in restructured text to use concise symbol combinations inside a paragraph that are unlikely to appear in normal text.

There is also simple markup to do hyperlinks, both to web URL's and as cross references within a Sphinx document, like to *Transformation*.

More elaborate documents by some of the authors are http://introcs.cs.luc.edu and Dr. Harrington's Python Tutorial.

2.4 Multiple Files

In many situations it is useful to break up a long document into separate files. This is particularly useful if you are collaborating. Separating and rearranging parts is very easy in Sphinx. Multi-file documents can easily be nested to any level with the toctree directive, like in the source for the initial file for this document.

File names listed under the toctree directive can also include a toctree directive. Just by moving single lines, you can rearrange sections. Numbering is reworked. Cross references still work....

This section just gave some of the flavor of Sphinx. Look at its web site, http://sphinx.pocoo.org/, for full documentation on installation, initiating a project, editing, and generating output formats.

[1] If you look at the source, you see that we are using a double colon :: before the set off text to make it literal.

CHAPTER

THREE

COLLABORATION WITH BITBUCKET

If you have edited and erased large sections that you later wish you could go back to, a version control system may be useful to you. Version control systems retain old versions (very efficiently with plaintext) and allow you to easily access earlier versions and to compare versions (of plaintext) easily.

There are many flavors. You can use one totally on a single machine, but with the Internet widely available, with traveling between multiple machines, and with other possible remote collaborators, a better system is likely to have a repository online. A very handy option in http://bitbucket.org. It allow free access for higher ed users with any number of private contributors, and you can chose a repository to be publicly viewable or not.

If you are collaborating, you will want to upload your changes frequently, so your collaborator has your latest work. Even if you are not so conscientious, if you are working on separate files, updating is immediate, and even if you make simultaneous separate edits to the same plaintext file, comparison and merging changes is facilitated.

Bitbucket allows two version control systems, both free downloads. We have been using Mercurial and there is also git. The basic Mercurial client work from the command line, but if that is foreign to you, there are also graphical, mouse-driven interfaces for most operating systems and now mobile apps.

The source files for this presentation are at https://bitbucket.org/gkthiruvathukal/fotl.

CHAPTER

FOUR

TRANSFORMATION

Of course with plaintext editing systems, there must be a separate step to create the final output. A big advantage is that it takes no more work to create multiple formats: They all are automatically generated from your single plaintext source.

For example we can generate final documents from the source for this talk using the program sphinx-build....

4.1 Demonstration

- This presentation itself is written in Sphinx with source at http://bitbucket.org/gkthiruvathukal/fotl and with fairly recent formatted output at http://anh.cs.luc.edu/fotl
- http://introcs.cs.luc.edu (a work in progress for Comp 170)
- http://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml (for Comp 150)

Demo the whole edit process:

- Clone the repository (typically for a new collaborator or new machine, but it can be into a separate folder on the *same* machine for a demo).
- Add a change to FOTL source in one repository.
- Make and view html and pdf versions.
- Commit changes to the local repository.
- Push changes to Bitbucket.
- Look in Bitbucket for the diff.
- In the second local repository, pull from Bitbucket and update.

4.2 Markdown

There are alternate systems for markup and transformation:

- http://daringfireball.net/projects/markdown/
- http://johnmacfarlane.net/pandoc/

Note: Pandoc can also transform individual files between multiple formats, for instance translate restructured text, markdown, hmtl or latex to another one of these formats or into docx, epub, pdf, or various slide show formats like S5 or Slidy.

CHAPTER

FIVE

REFERENCES

5.1 Resources

All are free downloads:

- Sphinx download and documentation, http://sphinx.pocoo.org/
- Python 2.7 required to run Sphinx (comes with a Mac, not Windows), http://python.org
- For pretty pdf output from sphinx: Latex, http://www.latex-project.org/
- Web Repository, http://bitbucket.org
- Version Control System Usable on Bitbucket, http://mercurial.selenic.com/

This presentation:

- Bitbucket source: http://bitbucket.org/gkthiruvathukal/fotl
- As web pages: http://anh.cs.luc.edu/fotl/html
- As pdf: http://anh.cs.luc.edu/fotl/FOTL.pdf

5.2 Presenters

- Dr. Andrew N. Harrington <aharrin@luc.edu>
- Dr. Konstantin Läufer <klaufer@luc.edu>
- Dr. George K. Thiruvathukal <gthiruv@luc.edu>

INDEX