Markdown Preprocessor (MarkdownPP)

The Markdown Preprocessor is a Python module designed to add extended features on top of the excellent Markdown syntax defined by John Gruber. These additions are mainly focused on creating larger technical documents without needing to use something as heavy and syntactically complex as Docbook.

MarkdownPP uses a set of selectable modules to apply a series of transforms to the original document, with the end goal of generating a new Markdown document that contains sections or features that would be laborious to generate or maintain by hand.

Documents designed to be preprocessed by MarkdownPP should try to follow the convention of naming files with a .mdpp extension, so that MarkdownPP can generate a document with the same name, but with the standard .md extension. As an example, this document in raw format is named "readme.mdpp", and the generated document from MarkdownPP is named "readme.md" so that GitHub can find and process that document when viewing the repository.

1. Installation and Usage

Currently, you'll need to download the source code from GitHub or clone the repository, and the run the installation script manually.

```
pip install MarkdownPP
```

There are two components to the project: a Python module, `MarkdownPP`, and a Python script that acts as a simple command line interface to the module, `markdown-pp`.

Assuming you have a file named `foo.mdpp`, you can generate the preprocessed file `foo.md` by running the following command:

```
$ markdown-pp foo.mdpp -o foo.md
```

If you do not specify an output file name, the results will be printed to stdout, enabling them to be piped to another command.

By default, all available modules are enabled. You can specify a list of modules to exclude:

```
$ markdown-pp foo.mdpp -o foo.md -e latexrender,youtubembed
```

To see usage instructions, including a list of enabled modules, supply the -h or --help arguments:

```
$ markdown-pp --help
```

2. Modules

2.1. Includes

In order to facilitate large documentation projects, MarkdownPP has an Include module that will replace a line of the form `!INCLUDE "path/to/filename"` with the contents of that file, recursively including other files as needed.

File `foo.mdpp`:

```
Hello
```

File `bar.mdpp`:

```
World!
```

File `index.mdpp`:

```
!INCLUDE "foo.mdpp"
!INCLUDE "bar.mdpp"
```

Compiling `index.mdpp` with the Include module will produce the following:

```
Hello
World!
```

Furthermore, the Include module supports the shifting of headers in the file to be included. For example,

File `foo.mdpp`:

```
# Foo
## Bar
```

File `index.mdpp`:

```
# Title
## Subtitle
!INCLUDE "foo.mdpp", 2
```

Compiling `index.mdpp` with the Include module and using `2` as shift parameter will yield:

```
# Title
## Subtitle
### Foo
#### Bar
```

2.2. IncludeURLs

Facilitates the inclusion of remote files, such as files kept in a subversion or GitHub repository. Like Include, the IncludeURL module can replace a line of the form `!INCLUDEURL "http://your.domain/path/to/filename"` with the contents returned from that url, recursively including additional remote urls as needed.

IncludeURL runs immediately after the Include module finishes executing. This means that is it possible to include local files that then require remote files, but impossible parse !INCLUDE statements found in remote files. This is prevent ambiguity as to where the file would be located.

Remote file `http://your.domain/foo.mdpp`:

```
Hello
```

Remote file `http://your.domain/bar.mdpp`:

```
Remote World!
```

Local file `index.mdpp`:

```
!INCLUDEURL "http://your.domain/foo.mdpp"
!INCLUDEURL "http://your.domain/bar.mdpp"
```

Compiling `index.mdpp` with the IncludeURL module will produce the following:

```
Hello
Remote World!
```

2.3. IncludeCode

Facilitates the inclusion of local code files. GFM fences will be added around the included code.

Local code file `hello.py`:

```
def main():
    print "Hello World"


if __name__ == '__main__':
    main()
```

Local file `index.mdpp`:

```
# My Code

!INCLUDECODE "hello.py"
Easy as that!
```

Compiling `index.mdpp` with IncludeCode module wil produce the following:

```
# My Code

```
def main():
    print "Hello World"

if __name__ == '__main__':
    main()
```
Easy as that!
```

Furthermore the IncludeCode module supports line extraction and language specification. The line extraction is like python list slicing (e.g. 3:6; lines three to six). Please note that line counting starts at one, not at zero.

Local file `index.mdpp`:

```
# My Code

!INCLUDECODE "hello.py" (python), 1:2
Easy as that!
```

Compiling `index.mdpp` with IncludeCode module will produce the following:

```
# My Code

```python
def main():
    print "Hello World"
```
Easy as that!
```

2.4. Table of Contents

The biggest feature provided by MarkdownPP is the generation of a table of contents for a document, with each item linked to the appropriate section of the markup. The table is inserted into the document wherever the preprocessor finds `!TOC` at the beginning of a line. Named `<a>` tags are inserted above each Markdown header, and the headings are numbered hierarchically based on the heading tag that Markdown would generate.

2.5. Reference

Similarly, MarkdownPP can generate a list of references that follow Markdown's alternate link syntax, eg `[name]: <url> "Title"`. A list of links will be inserted wherever the preprocessor finds a line beginning with `!REF`. The generated reference list follows the same alternate linking method to ensure consistency in your document, but the link need not be referenced anywhere in the document to be included in the list.

2.6. LaTeX Rendering

Lines and blocks of lines beginning and ending with $ are rendered as LaTeX, using QuickLaTeX.

For example,

```
$\displaystyle \int x^2 = \frac{x^3}{3} + C$
```

becomes

2.7. YouTube Embeds

As GitHub-flavored Markdown does not allow embed tags, each line of the form `!VIDEO "[youtube url]"` is converted into a screenshot that links to the video, roughly simulating the look of an embedded video player.

For example,

```
!VIDEO "http://www.youtube.com/embed/7aEYoP5-duY"
```

becomes

3. Examples

Example file.mdpp:

```
# Document Title

!TOC

## Header 1
### Header 1.a
## Header 2

!REF

[github]: http://github.com "GitHub"
```

The preprocessor would generate the following Markdown-ready document file.md:

```
# Document Title

1\. [Header 1](#header1)
1.1\. [Header 1.a](#header1a)
2\. [Header 2](#header2)

<a name="header1"></a>
## Header 1
<a name="header1a"></a>
### Header 1.a
<a name="header2"></a>
## Header 2

*   [GitHub][github]

[github]: http://github.com "GitHub"
```

4. Support

If you find any problems with MarkdownPP, or have any feature requests, please report them to GitHub, and I will respond when possible. Code contributions are *always* welcome, and ideas for new modules, or additions to existing modules, are also appreciated.

5. References

- Markdown Preprocessor on GitHub