Word 2 Markdown README

7/3/2016 1:04:01 PM

This document describes the procedure to convert Microsoft Word documents into Markdown documents. There are many variants of markdown, but it is a rather limited set.

# Instructions

The code is a C# application. Follow these instructions to generate a readme.md and images folder.

1. Double click the Word2Markdown.exe file to run the application
2. select a Word document (docx),
3. program generates a copy (Readme.docx) in the same folder as the original document,
4. the program locates and saves all inline pictures in the document under the folder images/ in the name format images/image#.jpg and will cross reference to this image in the markdown
5. The program generates a Readme.md document with all the mapings.

# Conversion Mappings

The Github web site uses markdown as its readme format to describe a repository. The initial goal of this executable was to produce readable Readme.md for github that included images.

Below are the conversion mappings from Microsoft Word document into some flavor (generally github) markdown.

|  |  |
| --- | --- |
| Word | Markdown |
| Heading 1, Heading 2, … styles | Corresponding number of # |
| Image | Save image into images/image#.jpg  Insert markdown:  ![Figure #](./images/image#.jpg?raw=true) |
| Bold Font | \*\*text\*\* |
| Underline Font | \_ text \_ |
| Code Style | \t code line1  \t code line2 |
| Table | Unclear, just used html to represent table,  Remaining problems with paragraph count using table  Github:  First Header | Second Header  ------------ | -------------  Content from cell 1 | Content from cell 2  Content in the first column | Content in the second column |
| List | Unordered (bullet) - \*  Numbered – #. Etc. |
| Hyperlink | Hyperlink![Text](URL)  e.g., http://github.com - automatic!  [GitHub](<http://github.com>) |
| Task Lists | Unhandled, should be:  - [x] @mentions, #refs, [links](), \*\*formatting\*\*, and <del>tags</del> supported  - [x] list syntax required (any unordered or ordered list supported)  - [x] this is a complete item  - [ ] this is an incomplete item |
| Strikethrough | Unhandled, should be:  ~~this~~ - this appears crossed out. |

# Modification

The program is a C# windows application. The program uses windows office word interoperability to do the word document manipulation. Originally the program was a VBA program, but saving the images appeared impossible for vba. So the program was rewritten into visual studio C# 2010, in which saving images was quite simple.

There is only one C# class to perform the Word to Markdown conversion. This class is called WordAutomation and does all the word automation and conversions. Of interest is that Word styles can vary from document to document. Under the WordAutomation class definition are the arrays that define the styles to search for to map heading, code, etc. into the corresponding Markdown equivalent. These straing arrays are currently defines as:

public string [] ListStyle = {"List Paragraph"};

public string [] CodeStyle = {"BoxedCode"};

public string [] TitleStyle = {"Title"};

public string [] Heading1 = {"Heading 1", "Heading1", "H1"};

public string [] Heading2 = {"Heading 2"};

public string [] Heading3 = {"Heading 3"};

You can modify these string and recompile the program to effect the changes.. Obviously an ini file or .Net config file could be used to modify these mappings.

The C# execution program is a normal C# windows Form application without the actual form. So the class Form1 and many of the Resources are meaningless. Instead, an instance of the WordAutomation is created and the the Init() method is called, and a dialog is popped up to select the word document, and the conversion is underway. For simplicity, only the Visual Studio comment extractor from the source code was used. Visual Studio will generate XML some XML documentation for entities with /// at the beginning of the method, variable, etc Below is the class diagram for the Wor2Markup program. Following is the Visual Studio XML comment extraction that was processed with a XSLT transformation to extract the comments into a readable format.



# WordAutomation Class Description

Handles the word automation and conversion into Markdown.

## Fields

#### TablesRanges

List of all tables and their ranges.

#### ImageRanges

List of all images and their ranges.

#### oWordDoc

Word application COM variable.

#### oWord

Word document COM variable.

#### filename

Filename of the current word file undergoing conversione.

#### foldername

Foldername of the current word file undergoing conversion.

## Methods

#### GetTableRanges

Extracts all the table ranges into the TablesRanges data structure .

#### GetImageRanges

Extracts all the image ranges into the ImageRanges data structure .

#### SaveAllImages(System.String)

Saves all the image ranges into the folder images under the current filename folder. Uses clipboard to copy and paste into image handler, which saves AS JPG.

#### InImage(Microsoft.Office.Interop.Word.Paragraph)

Determine if given paragraph p in found in the ImageRanges.

#### InTable(Microsoft.Office.Interop.Word.Paragraph,System.Int32@)

Determine if given paragraph p in found in the TablesRanges.

#### Words(Microsoft.Office.Interop.Word.Paragraph)

Looks at each word in a paragraph and formats font if word is bold or underlined.

#### Init

Pops dialog to retrieve word file to convert. Saves images, all image and table ranges, and then processes each paragraph. If image or table, handles. Tables are currently handled as HTML. If other style, mapping is performed. Output is streamed to Readme.md in the same folder as the oringal work file.

#### SaveClipboardImage(System.String)

Saves the clipboard into the given filename as a jpg. Uses System.Drawing.Image