# JMarkdown

A Java markdown library is created with an objective to provide an easy interface to convert Markdown syntax to other syntax. A simple Markdown to Html renderer is provided. Hope everyone can find it useful.

Since the MarkdownParser is still in development, there could be bugs.

## Command line usage

jmarkdown-cmdutil-vx.y.z.jar is released. And The general syntax for the command line is as follows. Since v2.0.0 release, command line options are added and everything is different. The old version will not be documented here.

|  |
| --- |
| jmarkdown-cmdutil-vx.y.z.jar <md2html / md2doc> [options]  -h print help  -o <file> output file  -i <file> input file  --styles <file> style file with .docx file extension  --toc create table of contents (for ooxml word only) |

Markdown to ooxml word is a little bit different, there is an option called --styles. The parser will open the file and read the pre-defined styles from the .docx file. Since the generator relies on pre-defined header styles, you need to named them in form of Header#. For example Header1, Header2, so on and so forth.

Let's say you have a style file named styles\_and\_themes.docx, then the command becomes:

|  |
| --- |
| java -jar jmarkdown-cmdutil-vx.y.z.jar md2doc -i in.md -o out.docx --styles styles\_and\_themes.docx |

## Library Usage

Compile your java project by including this JMarkdown library in your project's classpath.

## How to use HtmlRenderer or OoxmlWordRenderer

Some tests are written inside src/com/vincentcodes/tests/markdown/, you may make reference to those.

Within **TemplateParsingTest.java**

|  |
| --- |
| public class TemplateParsingTest {  public static void main(String[] args) throws IOException{  try(FileInputStream fis = new FileInputStream("Template.md")){  MarkdownParser parser = new MarkdownParser(new HtmlRenderer());  parser.parse(new String(fis.readAllBytes()));  HtmlRenderer renderer = (HtmlRenderer)parser.getRenderer();  System.out.println(renderer.getRenderedHtml());  }  }  } |

Styles I made may not be the finest, you may customize it by extending or modifying OoxmlWordRenderer class itself.

Within **TemplateParsingToWordTest.java**

|  |
| --- |
| public class TemplateParsingToWordTest {  public static void main(String[] args) throws IOException{  try(FileInputStream fis = new FileInputStream("Template.md")){  MarkdownParser parser = new MarkdownParser(new OoxmlWordRenderer(new File("out.docx"), new File("TemplateWithThemes.docx")));  parser.parse(new String(fis.readAllBytes()));  }  }  } |

## Creating your own Renderer

You can create your own Renderer to convert Markdown to languages. First, you need to implement the Renderer interface.

|  |
| --- |
| public class HtmlRenderer implements Renderer {  ...  } |

After that, let's take a look inside the Renderer interface. There are multiple methods like body(), h1(TextNode), etc. These functions are called by the parser when it starts parsing (by calling the parse() method).

For example, when the parser encounters a heading #, h1(TextNode) is called with the TextNode object. TextNode stores information of a group of texts TextGroup each text group has information on the style of the text it stores. Feel free to use TextNode.toString() to observe what they stores.

|  |
| --- |
| public interface Renderer {  /\*\*  \* Init  \*/  void body();  /\*\*  \* Close  \*/  void done();  // Headings  void h1(TextNode texts);  void h2(TextNode texts);  ...  } |

After that you can include it into your code.

|  |
| --- |
| MarkdownParser parser = new MarkdownParser(new CustomRenderer()); |

## Upcoming plans

* A command line util will be implemented for markdown to html conversion.
* Markdown to OpenXML renderer will be implemented