Markdown2Pdf

A simple library to convert markdown to pdf, **using Java**

Why

Sometimes, it might come in handy to create a pdf version of a markdown file. This library combines several other libraries to leverage the hassle of converting everything by yourself.

The codebase is kept fairly small, while still providing a fluent API.

Prerequisites

Version 1.0.x

This library only relies on the fact that you have at least Java JDK 1.6 installed. We support Java 1.8, and as of version 1.1 (which is still being worked on) we will make great use of the new features Java 8 has to offer.

```
<dependency>
    <groupId>eu.de-swaef.pdf</groupId>
    <artifactId>Markdown2Pdf</artifactId>
    <version>2.0.1</version>
</dependency>
```

Version 1.1.x and up

This version of the library will only support Java 1.8.

Building

```
> git clone https://github.com/Qkyrie/Markdown2Pdf.git
> mvn clean install
```

API examples

Instead of just providing you with examples here, we created an entire Class dedicated on teaching you the basics of the API. The class can be found in the repository, but for the current version, basically, this is it:

```
/**
 * Welcome to the tutorial for Markdown2Pdf.
 * We tried to create a page to demonstrate how to use our API. But in the end, we found
 * it was a lot easier to see how to use it when effectively using our code in a working
 * <p/>
 * If you look at the examples listed below, you should get the hang of how you'll be ab
 * and integrate this library in one of your own projects
 */
public class Tutorial {

    @Test
    public void basicExample() throws Markdown2PdfLogicException, ConversionException {
        /*
            Let's create an instance of the markdown2PdfConverter to work with
         */
        Markdown2PdfConverter markdown2PdfConverter =
                Markdown2PdfConverter.newConverter();

        /*
            We'll have to specify where our converter has to get his markdown data.
            With Java 8 in mind, we chose for a more generic approach, where we only have
            implement a class defined by Markdown2PdfReader
         */
        markdown2PdfConverter.readFrom(() -> "***Test***");

        /*
            Same thing goes for our writer. When the reading, cleaning and converting is
            we need to do something with the resulted bytes. What you do with it is enti
         */
        markdown2PdfConverter.writeTo(out -> {
            //do something with it here
        });

        /*
            What has the API done at this point? The simple answer: nothing. It only set
            for what it will be doing. It's lazily executed, which means that you'll have
            work in order for it to read, clean and convert.

            Actually making the API do the work can be done as follows:
         */
        markdown2PdfConverter.doIt();
    }

    /**
     * Our API has been implemented in such a way, that you can use it as a oneliner.
     * The above example can be rewritten, as shown in the this example
     */
    @Test
    public void basicExampleAsOneLiner() throws Markdown2PdfLogicException, ConversionExc
        Markdown2PdfConverter
                .newConverter()
                .readFrom(() -> "***Test***")
                .writeTo(out -> {
                    //here you can just do something with the bytes, like write it to a f
                    //for example.
                })
                .doIt();
    }

    /**
     * It might seem tedious, implementing two interfaces in order to make the execution
     * we say that it's for the best. With Java 8 in mind, these interfaces will be funct
     * replaceable with Lambda expressions.
     *
     * Fortunately, we provided some Simple implementations which can serve as replacemen
     * for your own implementation of the interfaces. Just follow the
     */
    @Test
    public void additionalUtilityClasses() {
       utilityReaders();
       utilityWriters();
    }

    /**
     * UtilityReaders are simple implementations of the
     * Markdown2PdfReader interface
     */
    private void utilityReaders() {
        simpleStringMarkdown2PdfReader();
    }

    /**
     * UtilityWriters on the other hand are simple implementation
     * of the Markdown2PdfWriter interface
     */
    private void utilityWriters() {
        simpleFileMarkdown2PdfWriter();
    }

    /**
     * the SimpleStringMarkdown2PdfReader can be used to read from a String. It's the mos
     * of implementations, which you'll probably find using a lot.
     */
    private void simpleStringMarkdown2PdfReader() {
        Markdown2PdfConverter
                .newConverter()
                .readFrom(new SimpleStringMarkdown2PdfReader("***Test***"));
    }

    /**
     * The SimpleFileMarkdown2PdfWriter implements the Markdown2PdfWriter in such a way,
     * thing you need to provide is a file. It will write the resulted bytes to the file
     */
    private void simpleFileMarkdown2PdfWriter() {
        Markdown2PdfConverter
                .newConverter()
                .writeTo(new SimpleFileMarkdown2PdfWriter(new File("pathname_comes_here")

}
```