1. Our existing implementation using pegdown for all the markdown parsing used the following logic where we do have our custom class i.e ConfluenceWikiLinkRenderer.java to render the Confluence WikiLInks. This class implementation I have already sent in the earlier comments.

PegDownProcessor pegDownProcessor = new PegDownProcessor(Extensions.ALL - (headerLinks ? 0 : Extensions.ANCHORLINKS)

- (hardwrap ? 0 : Extensions.HARDWRAPS) + (allowHtml ? 0 : Extensions.SUPPRESS\_ALL\_HTML));

try {

processed = pegDownProcessor.markdownToHtml(data, new **ConfluenceWikiLinkRenderer**(info, xhtmlContent));

// log.debug("pegdown returned HTML: {}", processed);

} catch (Exception exception) {

exception.printStackTrace();

throw new MacroExecutionException("Conversion of Markdown markup failed: " + exception.toString());

}

1. After I started migrating to flexmark the above code is changed to the following where in I have not used the custom class as I did not find a way to fit into flexmark API and because of not using the class wikilinks are not working with the below implementation. And getting compilation errors as well for my custom class as we are using pegdown APIs

MutableDataHolder OPTIONS = PegdownOptionsAdapter.flexmarkOptions(

Extensions.ALL - (headerLinks ? 0 : Extensions.ANCHORLINKS) - (hardwrap ? 0 : Extensions.HARDWRAPS)

+ (allowHtml ? 0 : Extensions.SUPPRESS\_ALL\_HTML)).toMutable()

// set additional options here:

// .set(HtmlRenderer.FENCED\_CODE\_LANGUAGE\_CLASS\_PREFIX,"")

;

**final** ArrayList<Extension> extensions = **new** ArrayList<Extension>();

// add your extra extensions here

// extensions.add(ConfluenceWikiLinkExtension.create());

**for** (Extension ext : OPTIONS.get(Parser.EXTENSIONS)) {

extensions.add(ext);

}

OPTIONS.set(Parser.EXTENSIONS, extensions);

**final** Parser PARSER = Parser.builder(OPTIONS).build();

**final** HtmlRenderer RENDERER = HtmlRenderer.builder(OPTIONS).build();

// use the PARSER to parse and RENDERER to render with pegdown compatibility

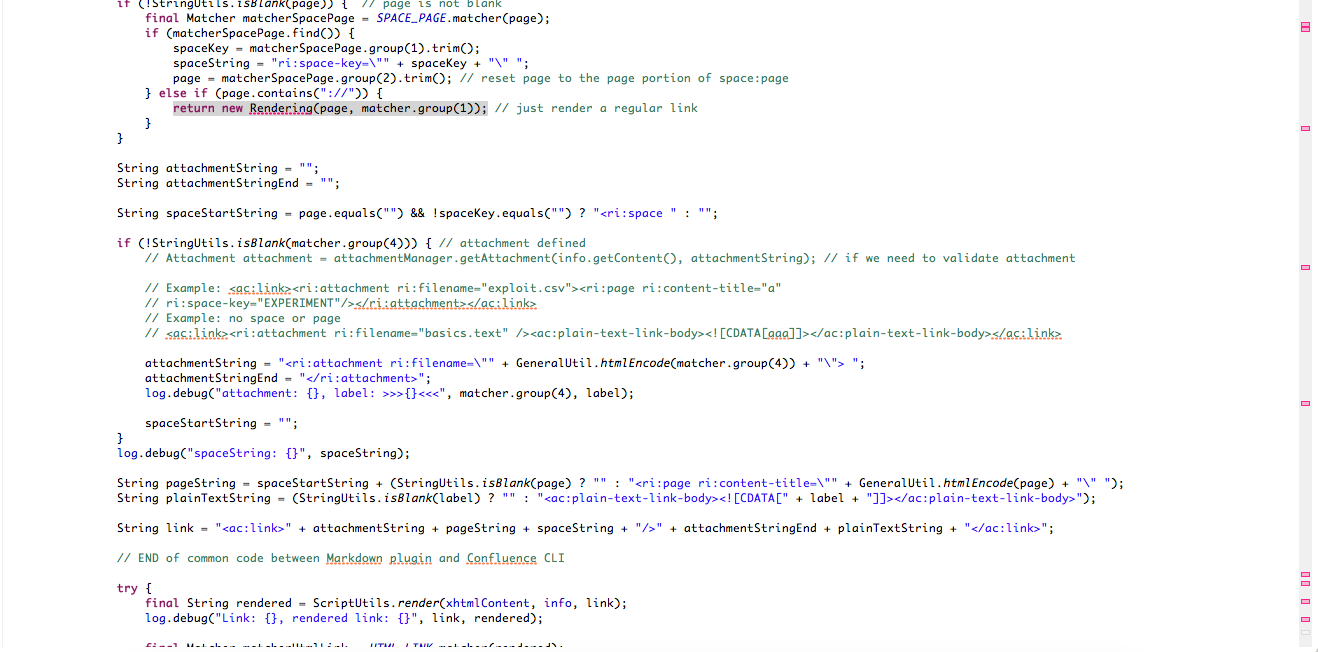
Node document = PARSER.parse(data);

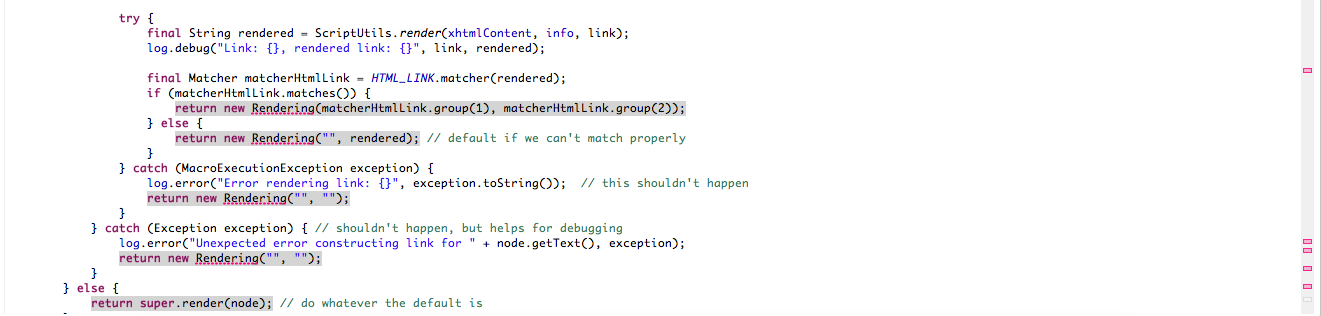
processedHtml = RENDERER.render(document);

*Please find the below screen shot for compilation errors in my custom class.* ***LinkRenderer, WikiLinkNode, Rendering***

****

****

****

****

Now our problem is

1. Wikilinks are not working
2. How we can find the replacements for the pegdown API’s errors which we used in our custom implementation
3. How we can incorporate this custom class along with other parsing objects in the flexmark current implementation.