**一、XdocReport的基本概念**

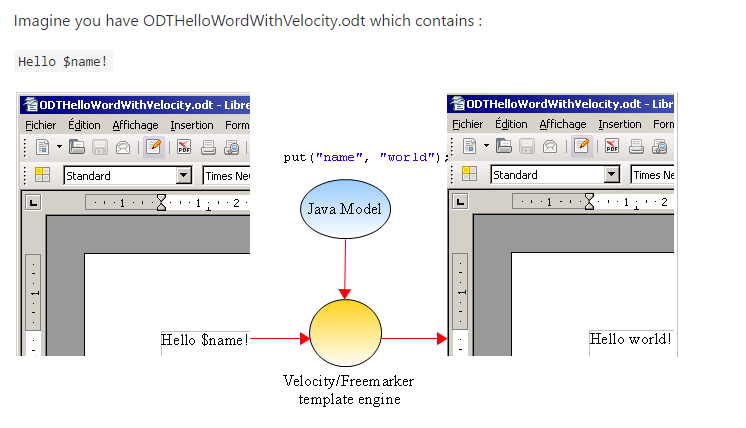
**XdocReport是模块化、轻量级基本用于XML的文档报告，对word文档的操作，java API合并MS OFFICE创建XML文档(docx、ppt)或OPENOFFICE(ODT),LibreOffice(ODT)与Java模型生成报告并转换为PDF、xhtml等形式。应用XdocReport作用于：**

**1、格式之间的转换,应用XdocReport提供的转换器，如word转换为pdf，word --🡪 xhtml**

**2、格式的支持：如OpenOffice的ODT、ODS，MS Word(Microsoft word)的docx(2007后)、doc（2007前）.**

**3、常见支持的语法：  
Freemark:Hello ${name}!**

**Velocity:Hello $name!**



**4、应用在OSGI环境的开发,更多用于报表**

**二、以下说的是MS 2007以及以后,可以根据后缀名docx简单的就辨认。XdocReport基于Apache POI的应用**

**Apache POI：提供API给Java操作MS WORD、MS EXCEL、MSPOWERPOINT文件，主要的结构：  
HSSF － 提供读写Microsoft Excel XLS格式档案的功能。**

**XSSF － 提供读写Microsoft Excel OOXML XLSX格式档案的功能。**

**HWPF － 提供读写Microsoft Word DOC格式档案的功能。**

**HSLF － 提供读写Microsoft PowerPoint格式档案的功能。**

**HDGF － 提供读Microsoft Visio格式档案的功能。**

**HPBF － 提供读Microsoft Publisher格式档案的功能。**

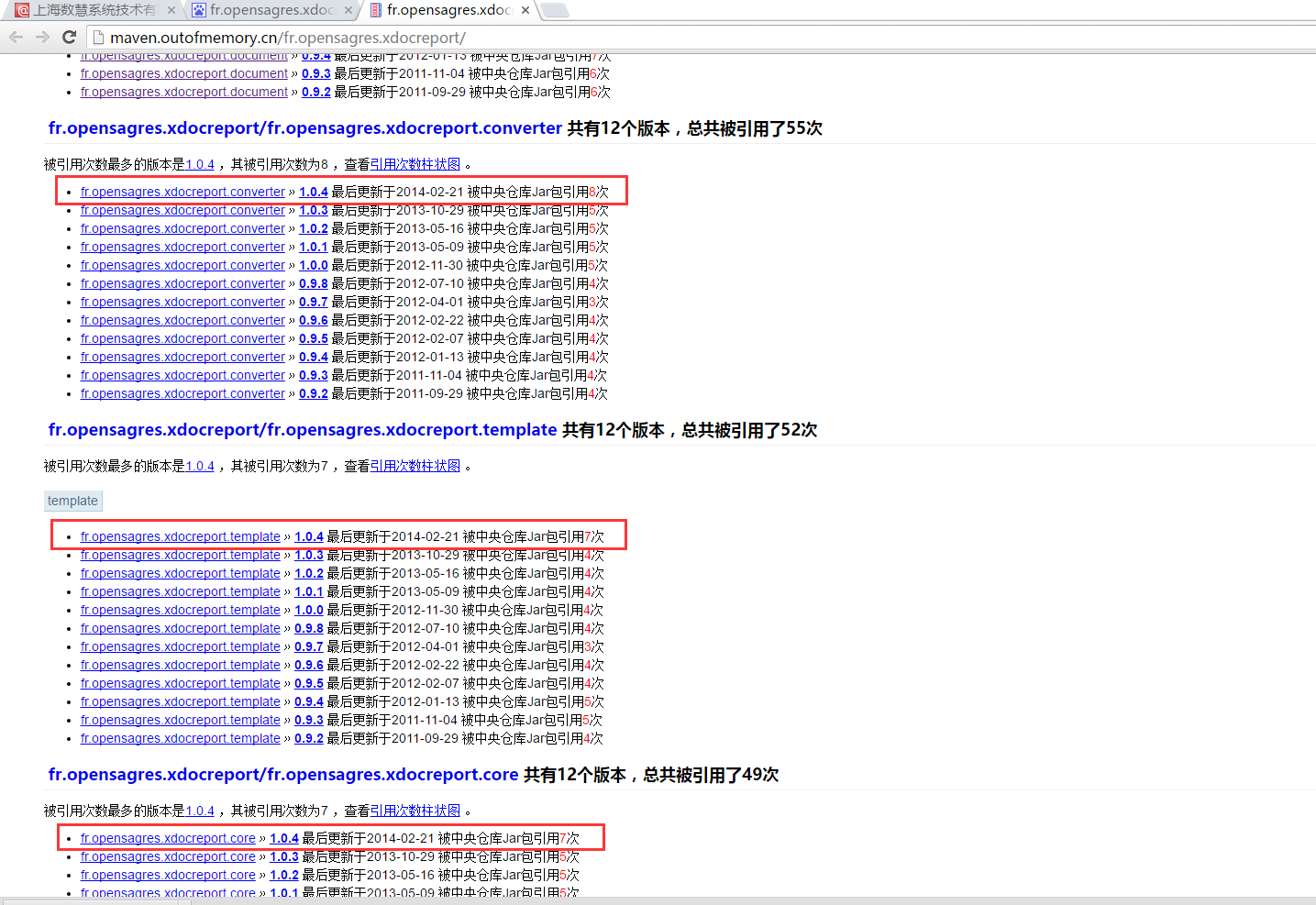
**HSMF － 提供读Microsoft Outlook格式档案的功能。**

**2007以前：Apache POI HWPF操作Microsoft Word文件,应用的jar包：poi-scratchpad-xxx.jar，HWPF 是 POI 支持 Word(97-2003) 的**[Java](http://lib.csdn.net/base/javaee)**组件，支持读写Word文档，但是写功能目前只实现一部分**

**2007及以上：Apache POI XWPF操作Microsoft Word文件，应用的jar包： poi-ooxml-xxx.jar，XWPF是 POI 支持 Word 2007+ 的 Java组件，提供简单文件的读写功能；**

**三、开始XdocReport对MS WORD操作的讲解，一下都是在2007及以上：  
1、将MS WORD转换为PDF**

**（1）、首先是maven依赖的引入，主要是fr.opensagres.xdocreport下的操作文档依赖，如：**



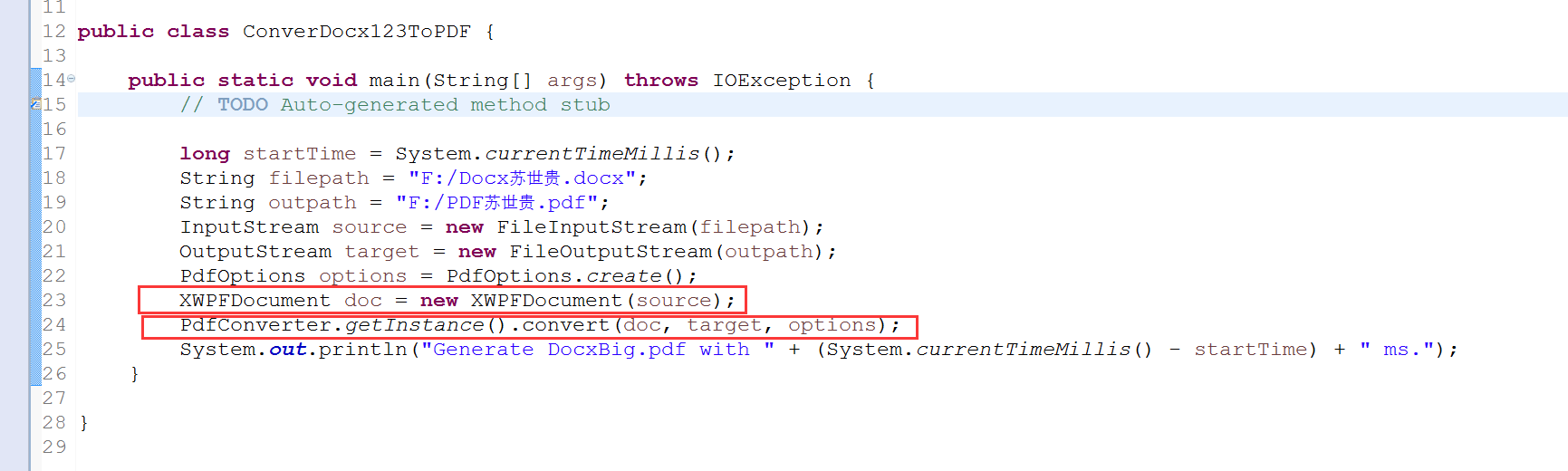
**ODT: 由openoffice软件产生的文档格式，一种开放文档格式，基于XML的文件格式，主要用于图表文字的处理，高版本的word（2007及以上）也可以打开**

**ODS：开放性的文档格式，类似于xls文件**

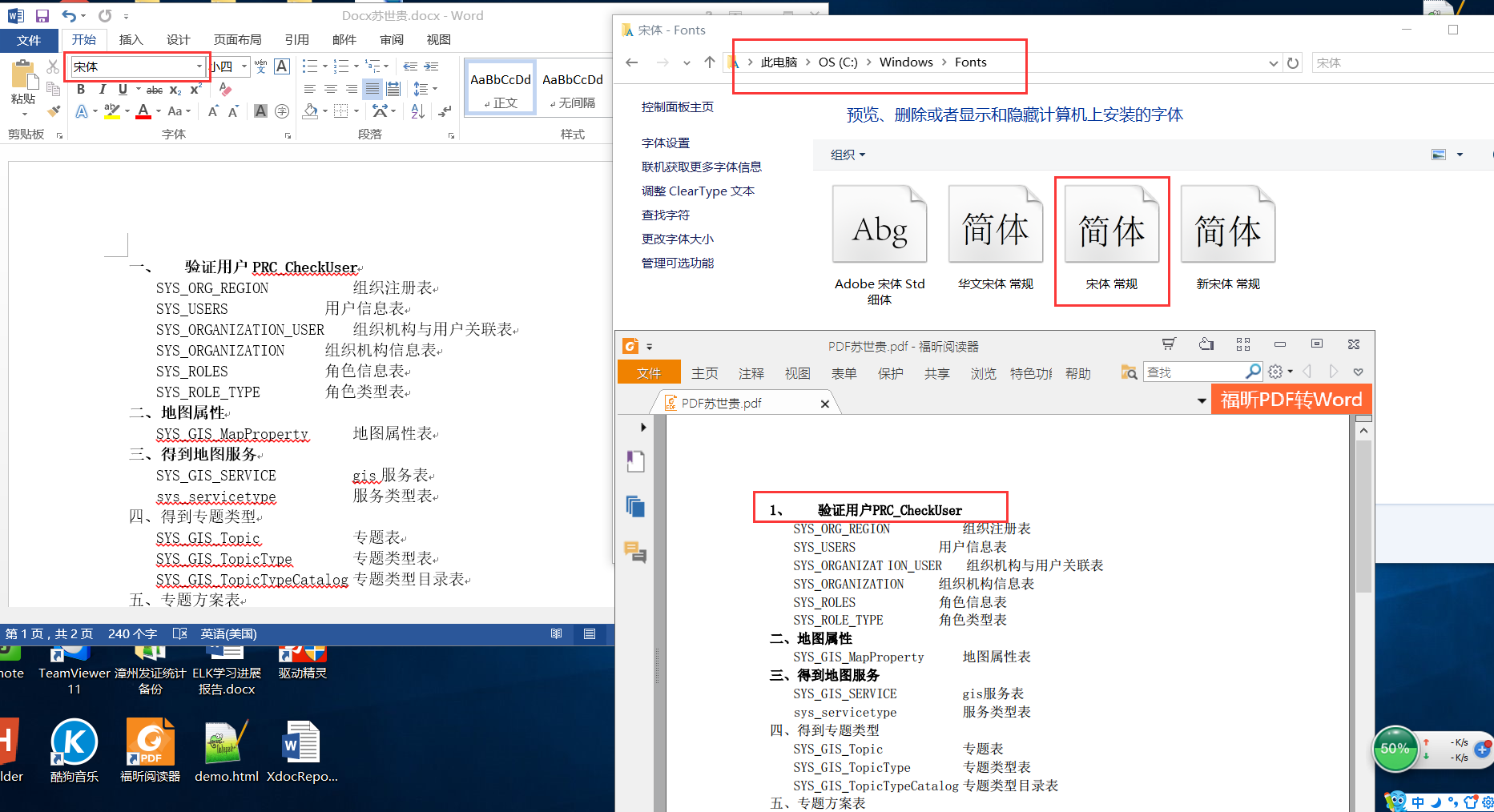
**ODP：一种数据文件，目前由Apache OpenOffice打开**

**（2）应用XWPF将MS WORD转换为PDF，PDF较word安全性高，PDF可以将文字、字型、格式、颜色以及独立于设备和分辨率的图形图像封装在一个文件中，集成度和安全度高**

**在Apache POI中，MS WORD 2007以后，通过XWPFDocument对象来解释docx文件，并在构造传入docx文件的读取流完成解释过程。支持大数据、有表格图像、目录树、ooxml等的文档转换，此时并未应用FreeMarker或者是Velocity的模板，纯粹的转换格式。**

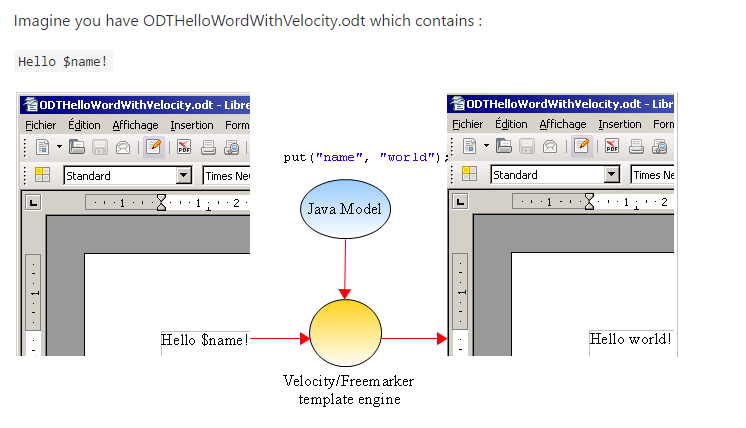


**注意，如果内容存在中文，平常的设置字体编码如：PdfOptions.create().fontEncoding("UTF-8")，是没用的，要想支持中文的话，就必须在电脑上有相应的字体包。如：**



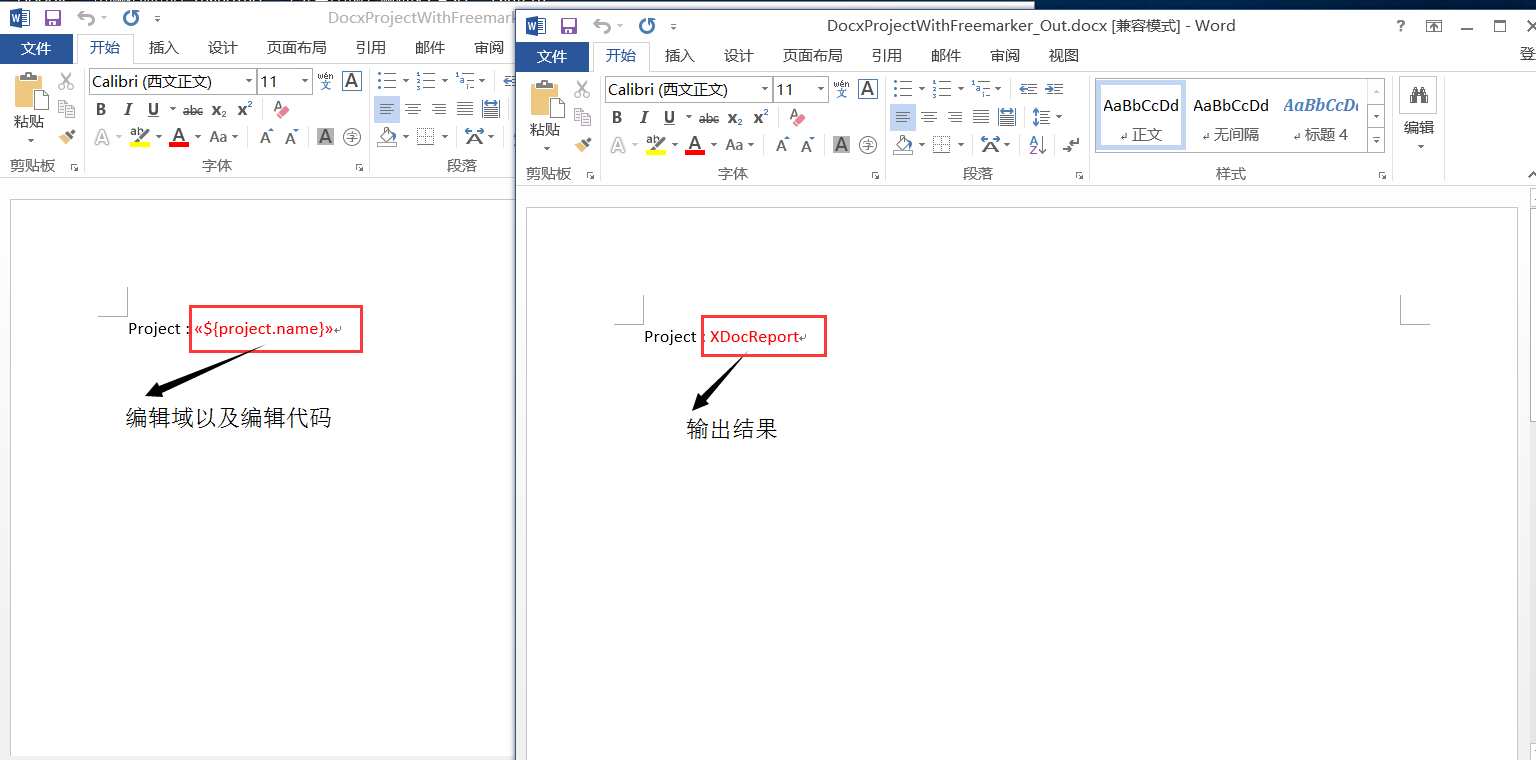
**（3）、转换为html,原理和转换为pdf一样，只是转换后的格式不同：**

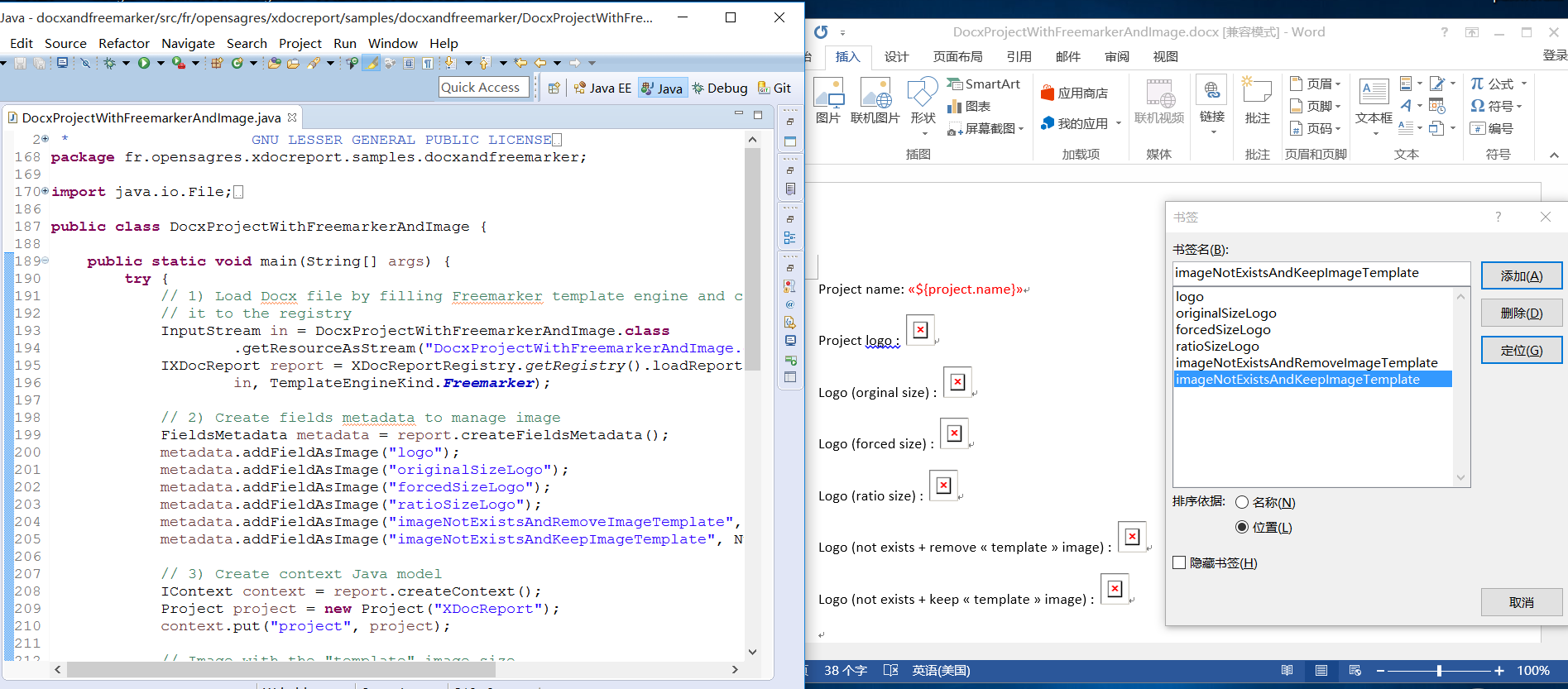
**2、支持FreekMarker的语法，原理如下图：**



**具体流程：**

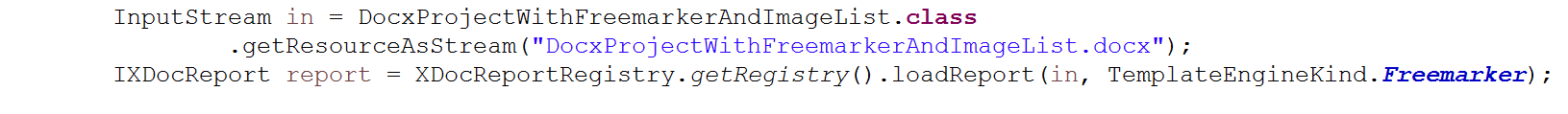
1. **加载所需操作的文档docx，文档中已建立好动态变量，如编辑域和编辑代码，并且注册加载FreeMarker模板，主要用fr.opensagres.xdocreport.document-1.0.4.jar下的IXDocReport类**
2. **创建java的上下文模型**
3. **利用java模型与docx合成报告并输出**

 **如果需要输出图片形式，则需要设置相应的书签，如**

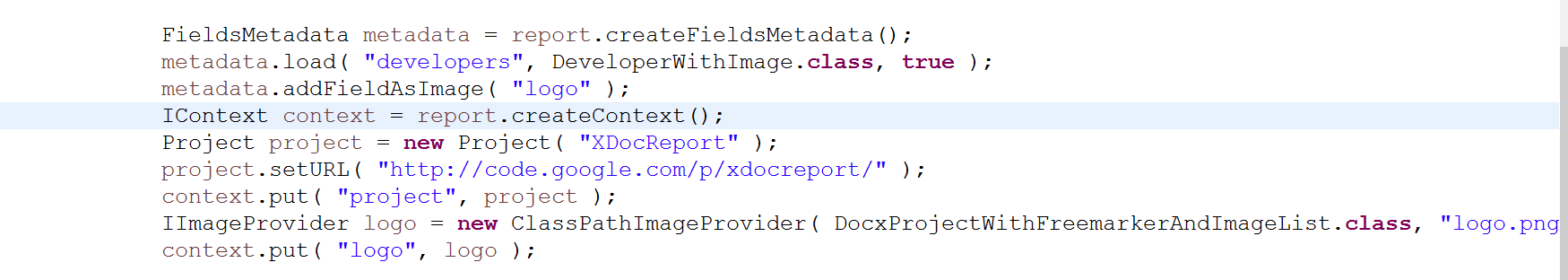


**实例分析一个XdocReport支持FreeMarker语法的过程：**

1. **加载所需操作的文档docx，文档中已建立好动态变量，如编辑域和编辑代码，并且注册加载FreeMarker模板**



1. **创建字段元数据管理([#列表FreeMarker]),创建Java对象模型，承载输出输出信息，Java对象模型：DeveloperWithImage**

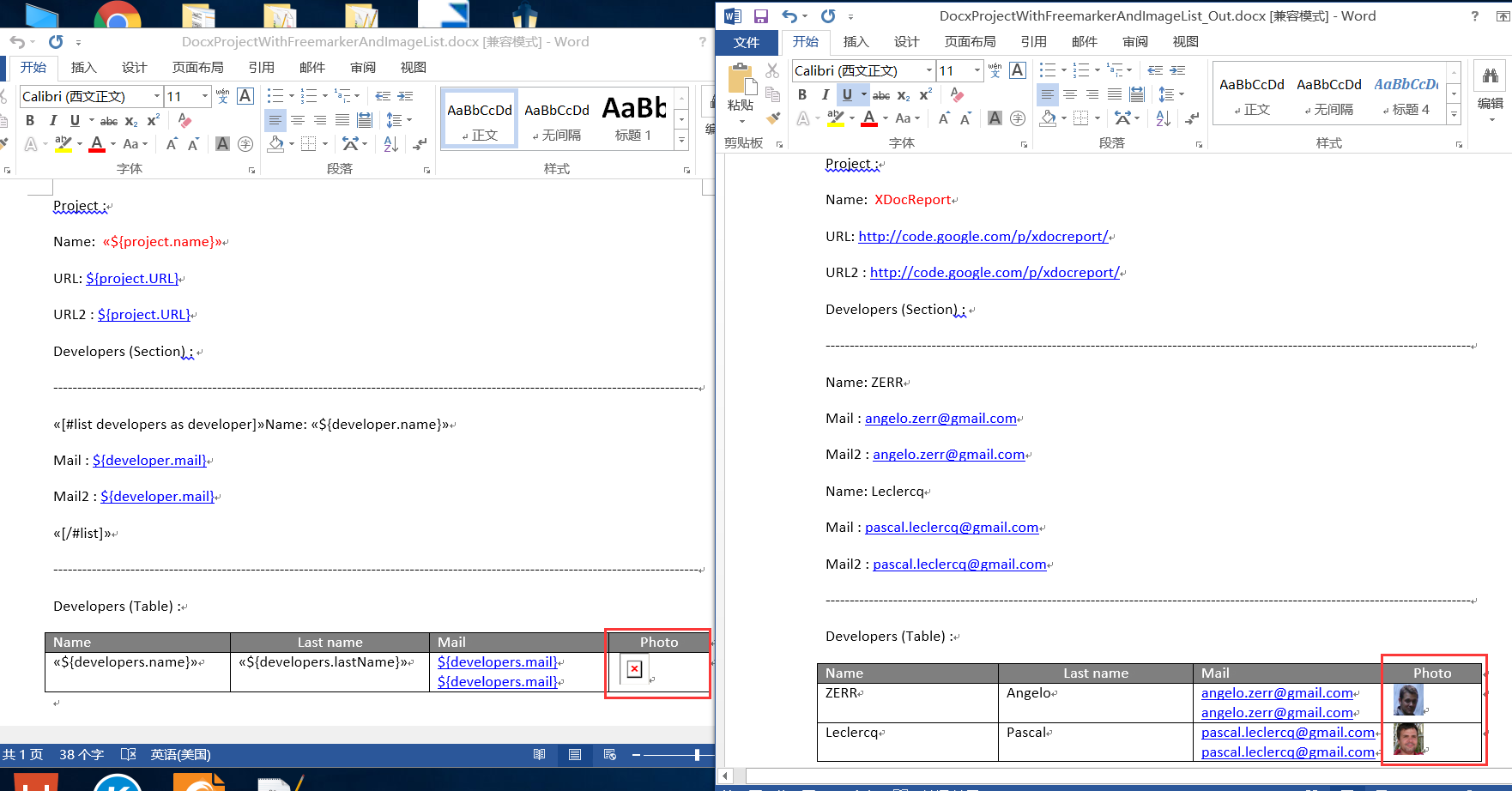


**利用java模型与docx合成报告并输出，FreeMarker列表输出的简单语法：**Name:

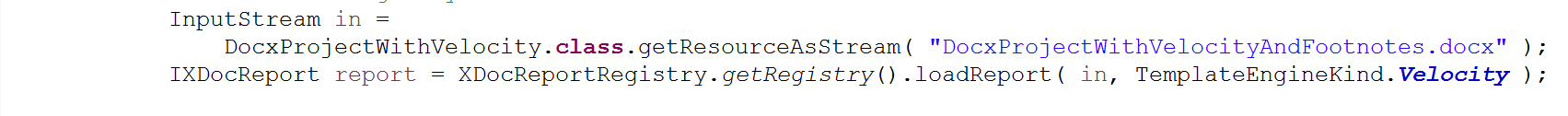
Mail : [${developer.mail}](mailto:$%7bdeveloper.mail%7d)

Mail2 : [${developer.mail}](mailto:$%7bdeveloper.mail%7d)

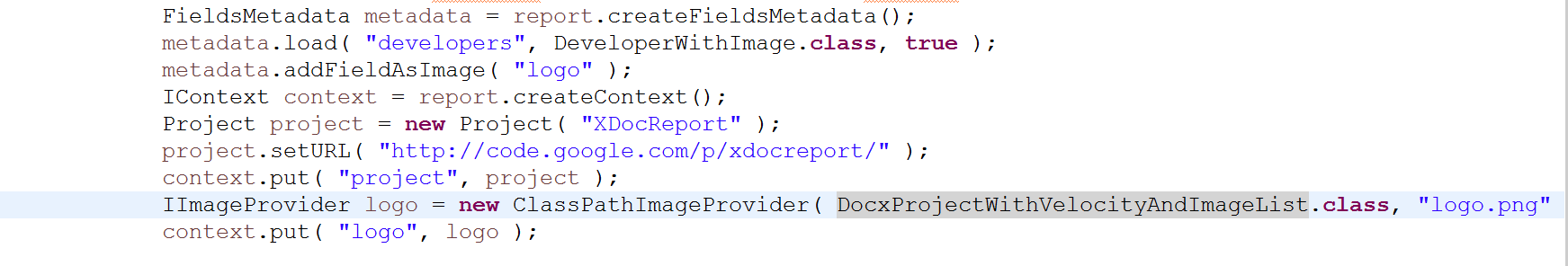


**（3）运行结果和被操作的文档进行对比：**

**3、XdocReport支持Velocity语法，用法和FreeMarker区别不大，主要在于加载的模板是Velocity。在这里就不详细介绍了，举个简单的例子：  
（1）加载被操作的文档以及Velocity模板**



**（2）创建元数据和Java模型对象**



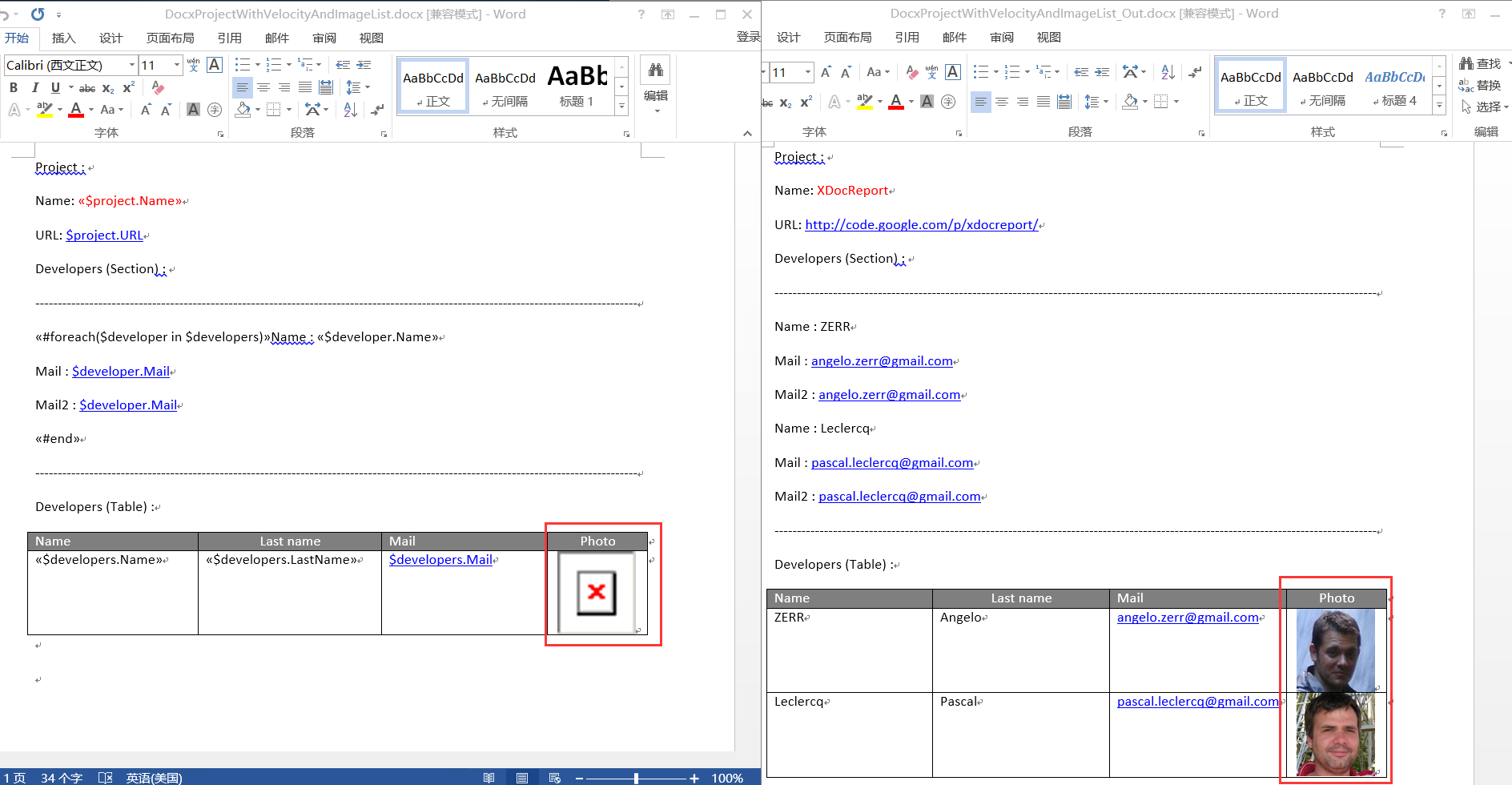
**（3）迭代输出信息，Velocity迭代输出的简单信息：**Name :

Mail : [$developer.Mail](mailto:$developer.Mail)

Mail2 : [$developer.Mail](mailto:$developer.Mail)

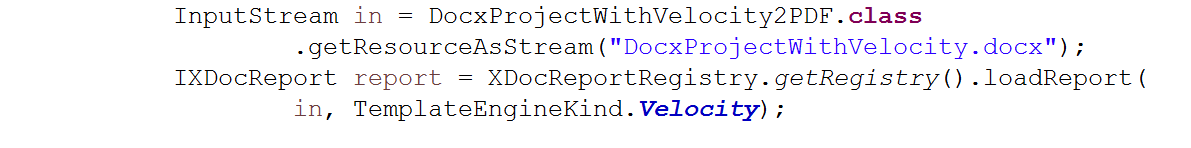


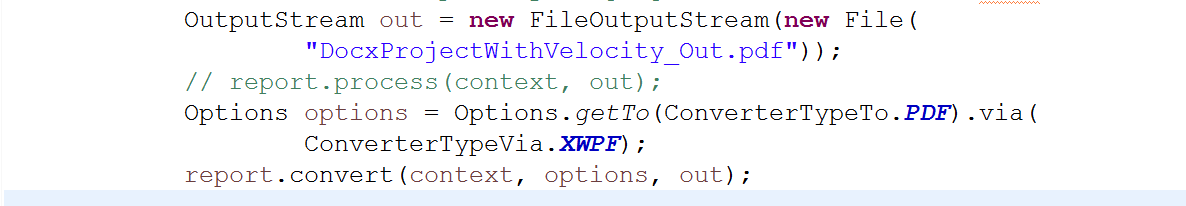
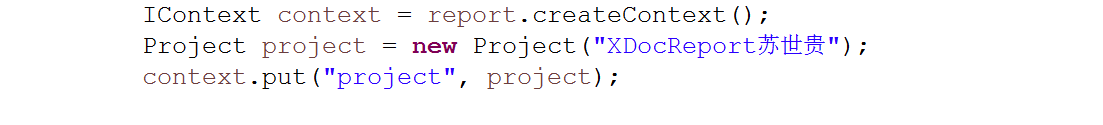
**被操作的文档和运行结果对比：**

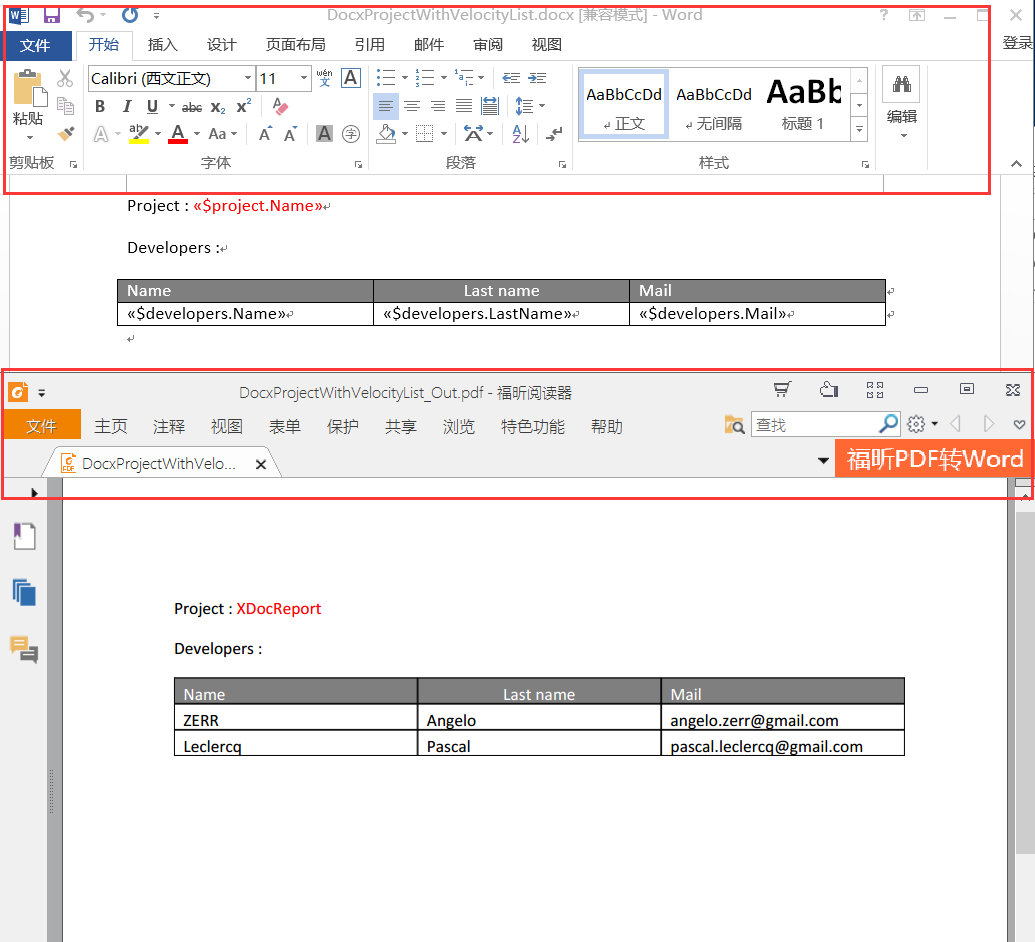


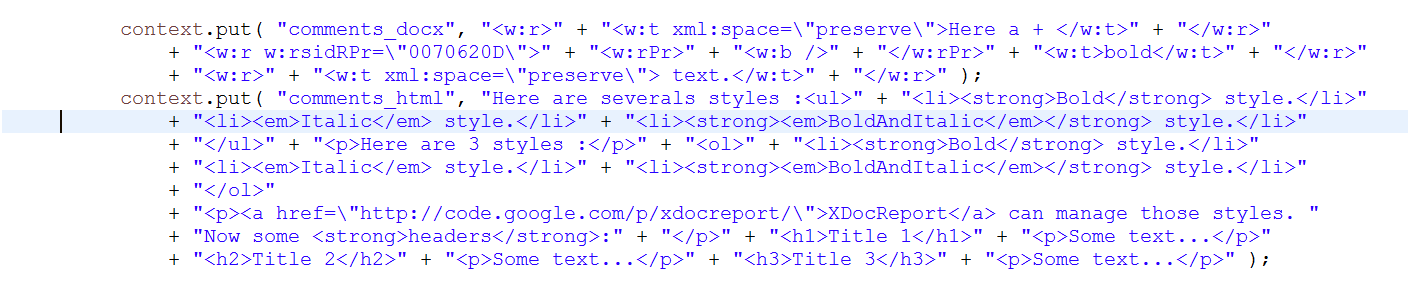
**4、应用Velocity模板转化为pdf或html，原理和应用Velocity的语法将docx的信息输出是一样的，只是在这里是将格式转换：**

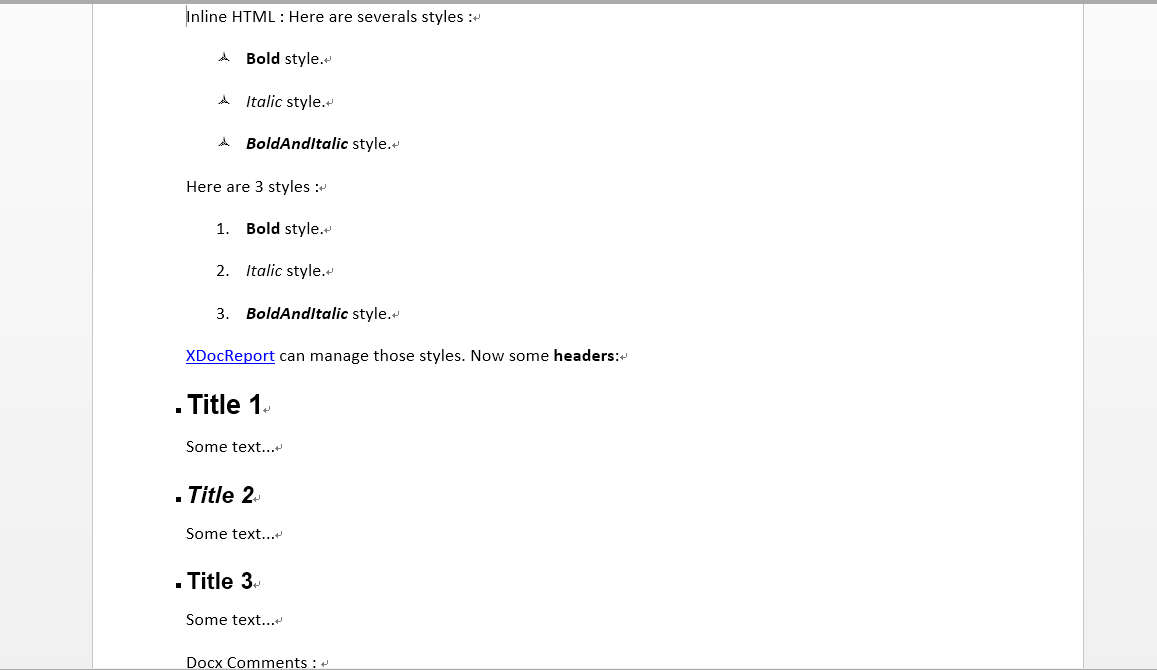
**加载被操作的文档以及Velocity模板、创建Java模型对象、利用java模型与docx合成报告以转换后的格式输出**





**在此可以知道，通过Velocity模板可以先完整对docx的数据信息输出，接着已pdf或是其他格式输出，如：**

**5、可以应用FreemMarker模板直接定义输出的信息:**

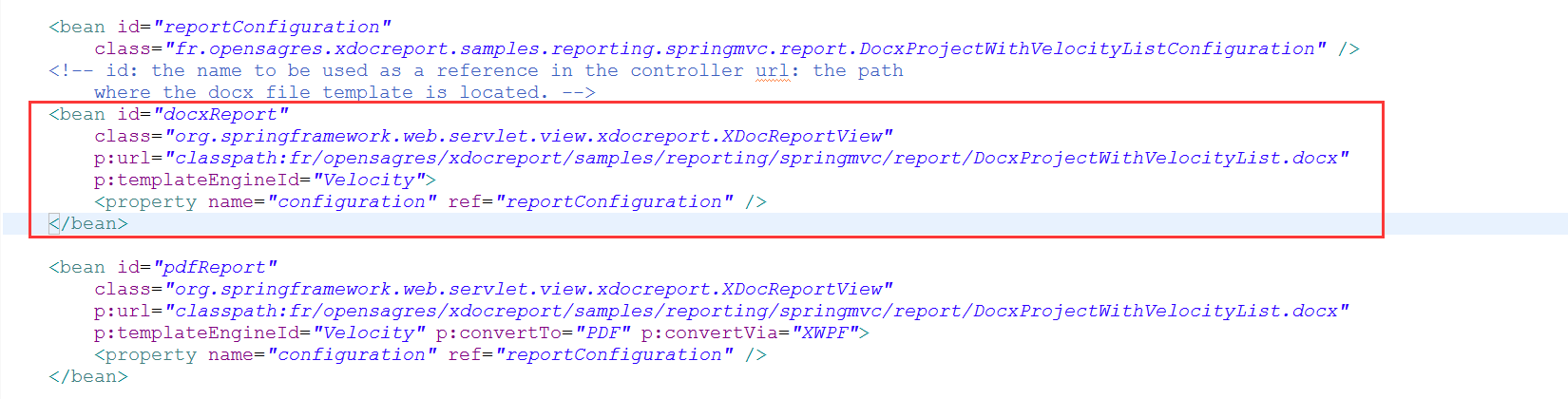
**运行输出文档的结果：**

**6、XdocReport操作odt、odp、ppt文件的原理过程基本和操作docx一样**

**四、web上应用XdocReport，比如应用SpringMVC**

**1、springmvc的环境搭建好，如maven引进springmvc的包、springmvc-servlet.xml的配置，web.xml配置好springmv，spring注入xdocreport等。**

**2、将docx文件应用Velocity模板输出Java模型信息的话，需要配置Xdocreport ，并被spring的配置文件引入**



**其实，这和传统的用Java项目结合Velocity模板操作docx文件的原理是一样的，只是这里是通过Xdocreport的配置信息加载docx文档、Velocity模板，在控制层里合成信息输出为docx或者转换为pdf等格式。**

**2、应用Servlet在web通过XdocReport操作，前提在web.xml配置好Servlet，此时，并没有应用Java模型对象，通过Velocity模板输出，原理是一样的。**

**前台jsp:**

**注意reportId,后台会根据这个ID输出操作后的文档，**

**后台：**