Experiência de programador

Joao Pedro Schmitt

A Java-based application used to fill Word (docx) files.

 schmittjoaopedro . Java 3 03+00:00 Março 03+00:00 20183 03+00:00 Março 03+00:00 2018 3 Minutes

Introduction

Companies are every time demanding more automation tools for documents generation, these tools are used to speed up contracts generation and client documentation. To support this necessity, tools like Word have been widely used and integrated with different technologies to fill data in template files. In this context, the Java programming language combined with the library DOCX4J allow, by programming, to generate Word documents filled with data from XML files. However, besides the extensive library documentation and community support available on the internet, there is still a lack for simple guides, and this article proposes to offer a very simple guide (step-by-step) to generate Word documents using the DOCX4J library.

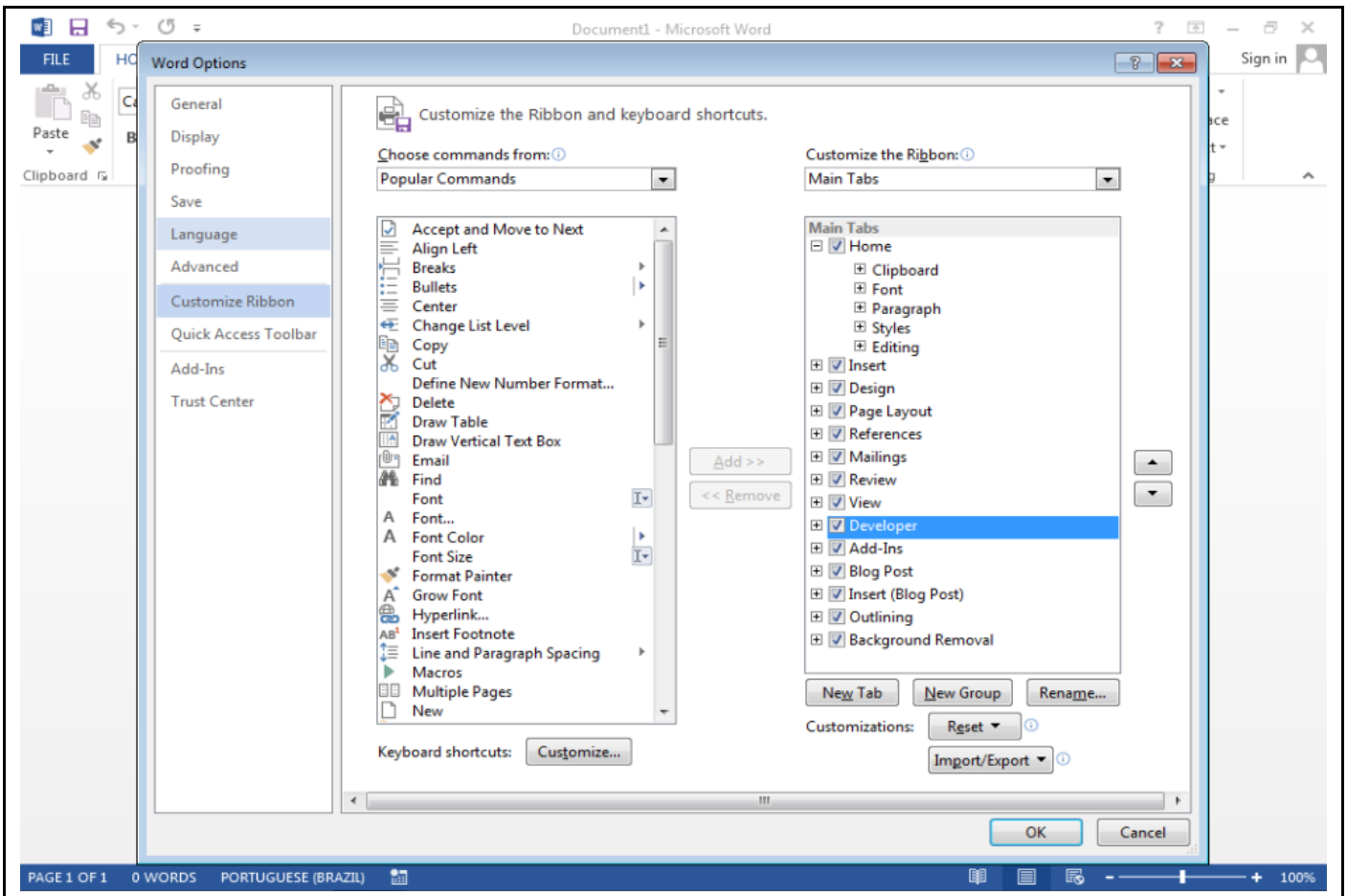
(https://github.com/schmittjoaopedro/java-word-template-filler#guide-to-generate-word-files-with-java)Guide to generate Word files with Java

Word documents generation is based on XML source files. In this example, the XML structure used to fill the Word template is presented below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<root>
    <person>
        <name></name>
        <city></city>
        <phone></phone>
        <marriedYes></marriedYes>
        <marriedNo></marriedNo>
        <male></male>
        <female></female>
    </person>
</root>
```

With the XML structure defined, open the Word and enable the developer mode following the steps below (Figure 1):

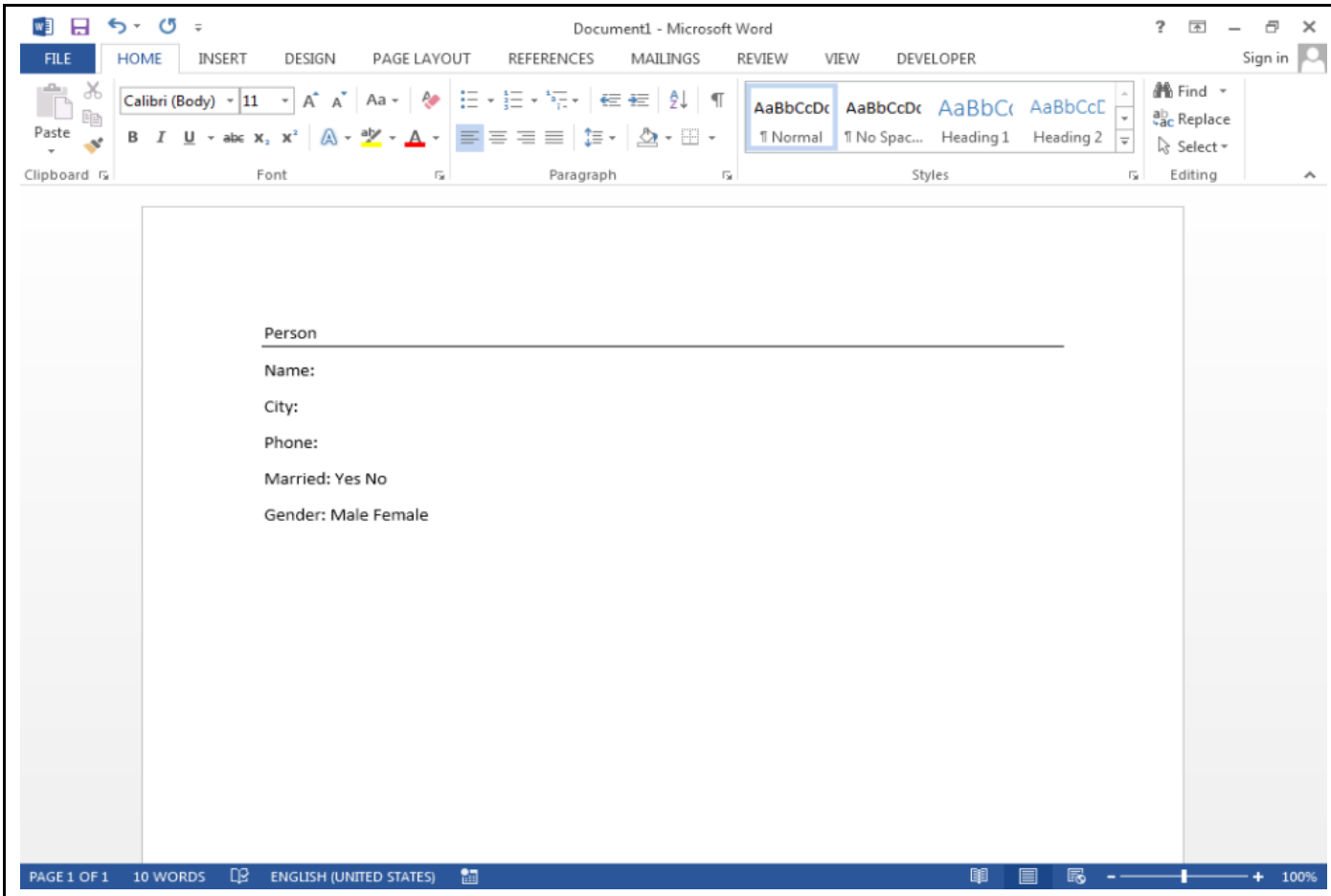
- Select File > Options > Customize Ribbon
- On the right side of the panel select “Main Tabs” and mark the “Developer” checkbox.
- Click OK.



(https://github.com/schmittjoaopedro/java-word-template-filler/blob/master/images/DeveloperTab.png)

Figure 1: Word 2013 Developer Tab

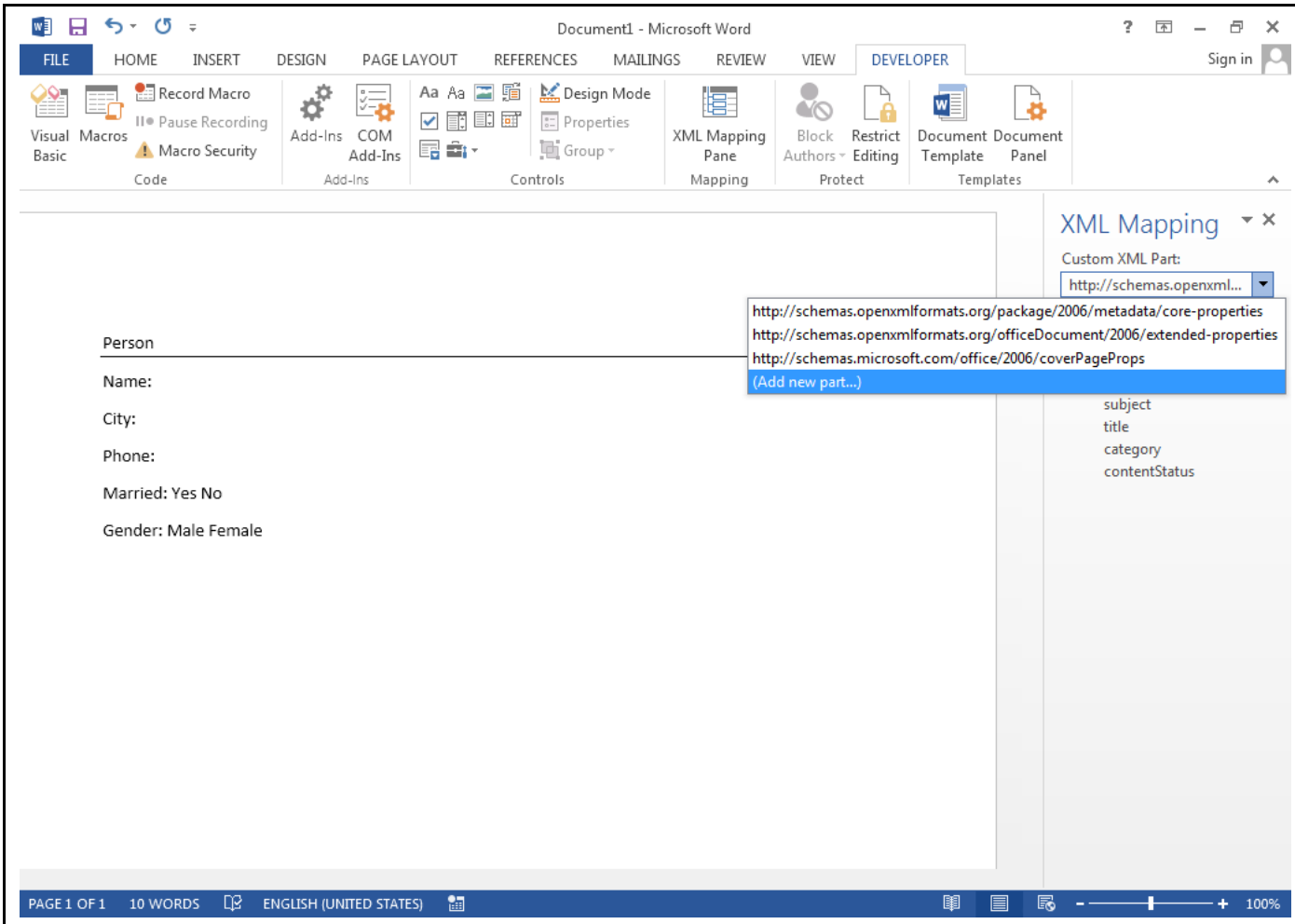
With the Developer tab activated, create a simple document template (Figure 2).



(https://github.com/schmittjoaopedro/java-word-template-filler/blob/master/images/DocSimple.png)

Figure 2: Simple Document

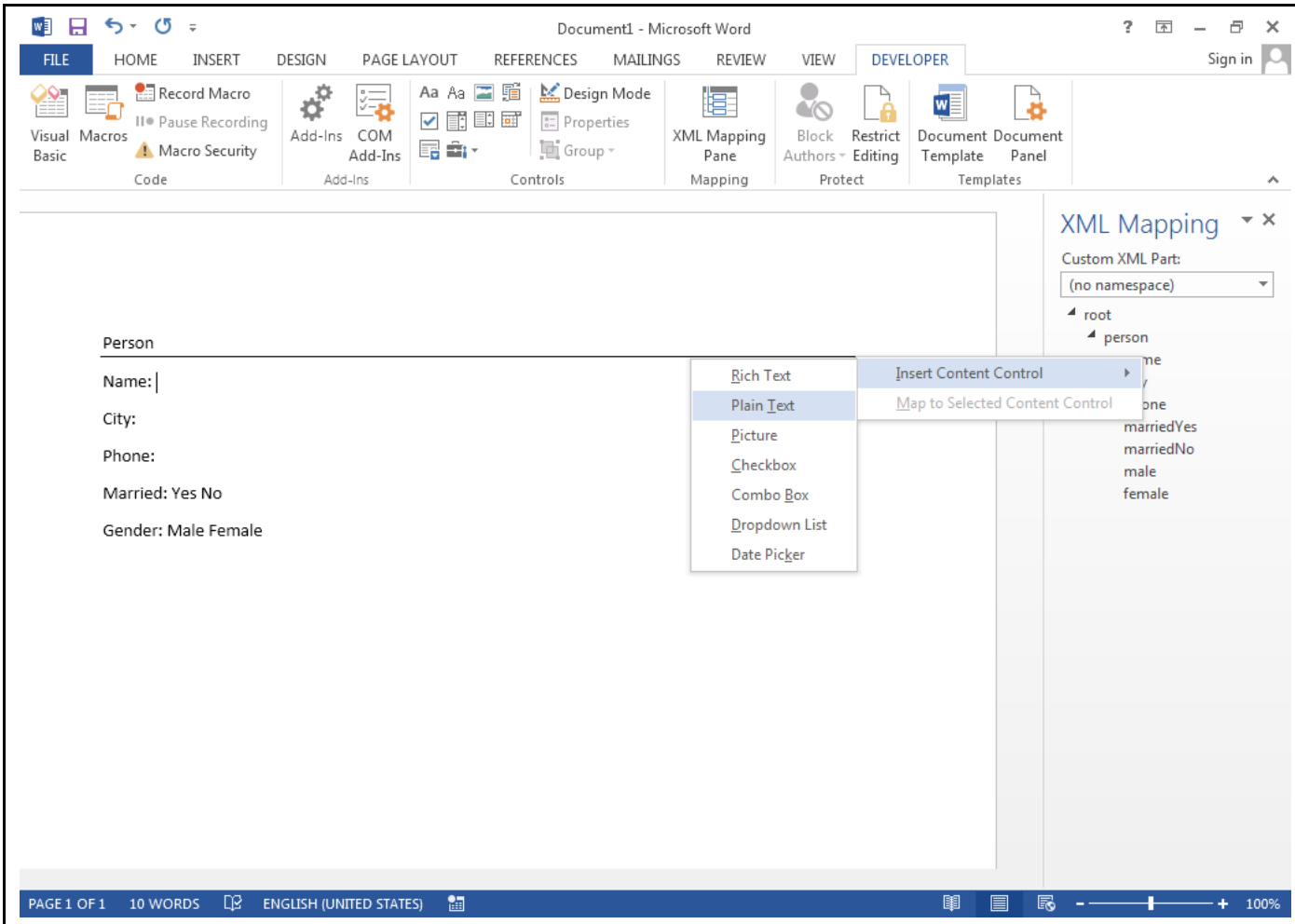
After that, go to the Developer Tab and click in the "XML Mapping Pane" button to open XML data configuration, then in the "Custom XML Part" combo select "Add a new part" (Figure 3).



(https://github.com/schmittjoaopedro/java-word-template-filler/blob/master/images/AddXmlStructure.png)

Figure 3: Add XML Structure

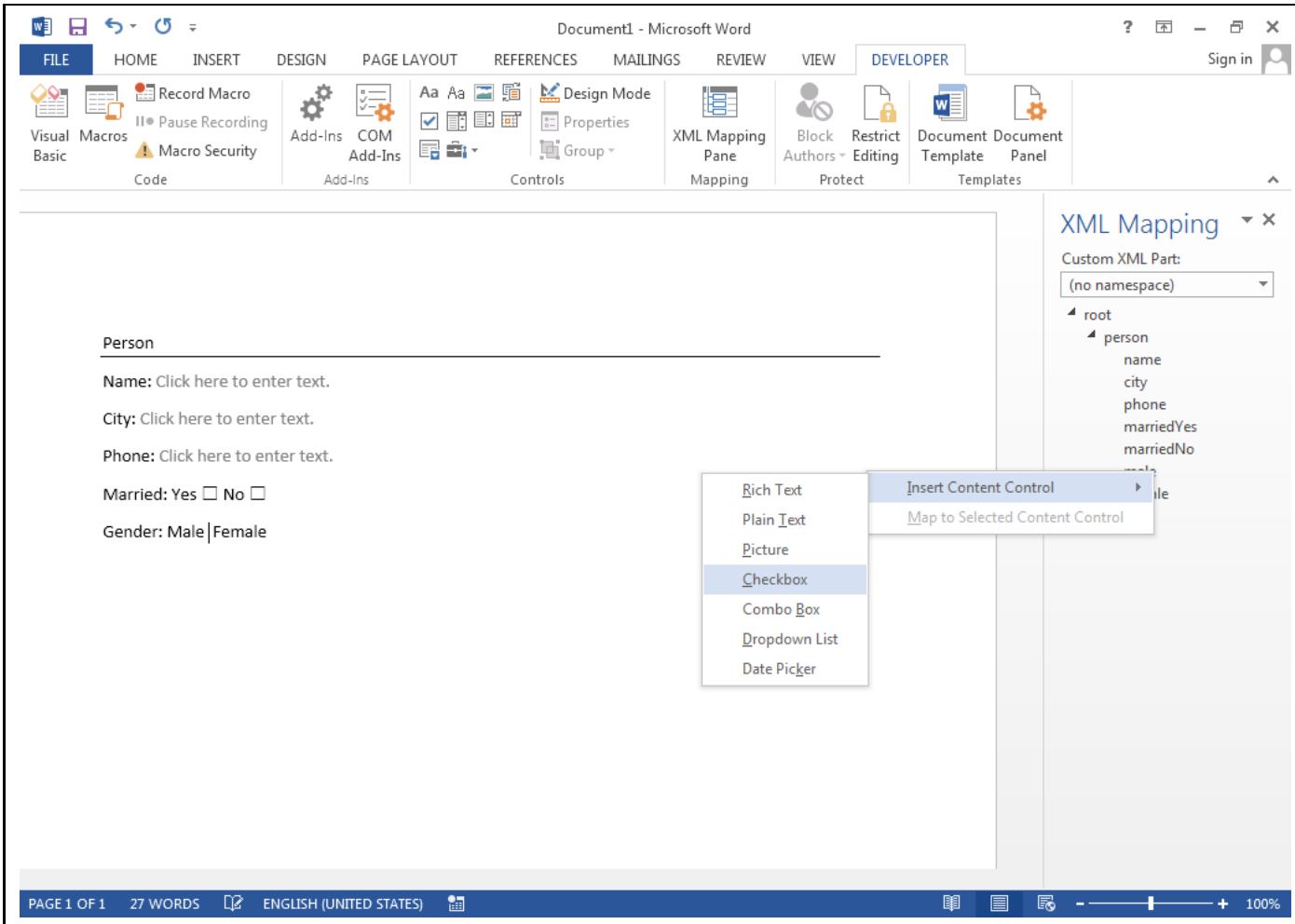
Select our new XML structure with the name "(no namespace)", and use the text cursor to select a position for the control. After that, add a new Plain Text control using the XML Structure. An example is presented in figure 4.



(https://github.com/schmittjoaopedro/java-word-template-filler/blob/master/images/AddControls.png)

Figure 4: Add Input Text Control

To exemplify another control example, figure 5 illustrates a checkbox configuration.



(https://github.com/schmittjoaopedro/java-word-template-filler/blob/master/images/AddCheckControl.png)

Figure 5: Add Checkbox Text Control

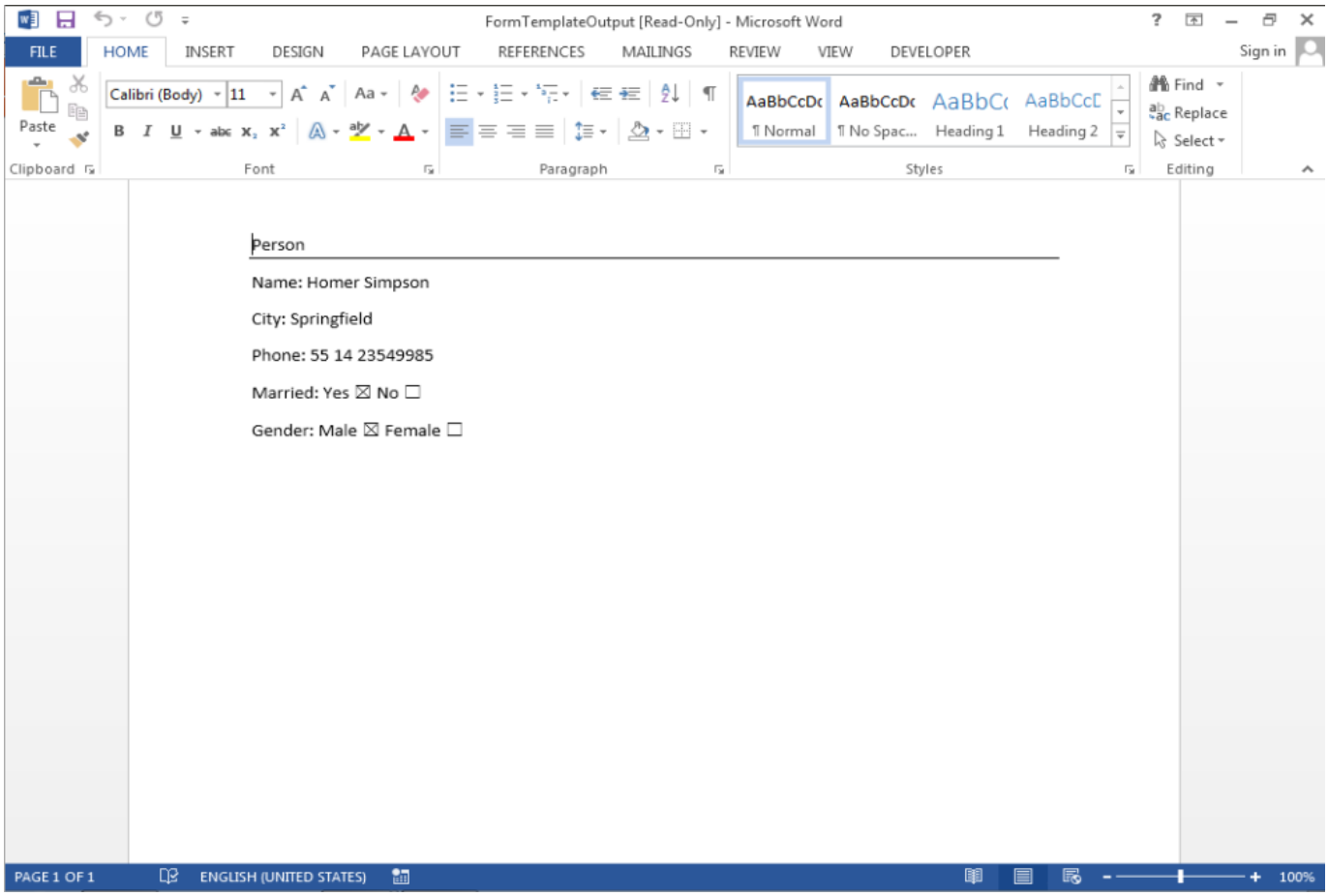
Finally, save the Word document and create a simple Java project using the DOCX4J (https://www.docx4java.org/) library. The code necessary to process the Word document is presented below.

```
package com.github.schmittjoaopedro;

import org.docx4j.Docx4J;
import org.docx4j.openpackaging.packages.WordprocessingMLPackage;

import java.io.File;
import java.io.FileInputStream;

public class WordTest {

    public static final String input_DOCX = "E:\\Temp5\\Word document templat

    public static final String input_XML = "E:\\Temp5\\Word document data.xml

    public static final String output_DOCX = "E:\\Temp5\\Word document output

    public static void main(String[] args) throws Exception {
        WordprocessingMLPackage wordMLPackage = Docx4J.load(new File(input_DO
        FileInputStream xmlStream = new FileInputStream(new File(input_XML));
        Docx4J.bind(wordMLPackage, xmlStream, Docx4J.FLAG_BIND_INSERT_XML | D
        Docx4J.save(wordMLPackage, new File(output_DOCX), Docx4J.FLAG_NONE);
        System.out.println("Saved: " + output_DOCX);
    }

}
```

Execute the above program with the XML data presented below.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<root>
    <person>
        <name>Homer Simpson</name>
        <city>Springfield</city>
        <phone>55 14 23549985</phone>
        <marriedYes>true</marriedYes>
        <marriedNo>false</marriedNo>
        <male>true</male>
        <female>false</female>
    </person>
</root>
```

If everything works fine, a Word output file will be generated, with the data presented in figure 6.



(https://github.com/schmittjoaopedro/java-word-template-filler/blob/master/images/Result.png)

Figure 6: Generated document

The project source code is available in the GitHub (https://github.com/schmittjoaopedro/java-word-template-filler).

Com as etiquetas :
Automation,
Data,
Fill,
java,
Office,
programming,
Template,
word,
XML

Publicado por schmittjoaopedro

Graduado como bacharel em Sistemas de Informação pelo Centro Universitário Católica de Santa Catarina campus Jaraguá do Sul. Formado no Ensino Médio pelo Senai com Técnico em Redes de Computadores Articulado. Atualmente desenvolvedor JEE/Web em Sistemas de Engenharia na WEG. Pesquisador no período de faculdade em Informática pela Católica de Santa Catarina. Contato 47 - 99615 2305 E-mail:

schmittjoaopedro@gmail.com Web page: https://joaoschmitt.wordpress.com/ Linkedin: https://www.linkedin.com/in/joao-pedro-schmitt-60847470/ Curriculum lattes: http://lattes.cnpq.br/9304236291664423 Twitter: @JooPedroSchmitt Ver todos os artigos de schmittjoaopedro