

Java 使用 FreeMarker 自动生成 Word 文档（带图片和表单）

- 1 背景
- 2 目标效果
- 3 创建 Word 模板
 - 3.1 创建模板文档
 - 3.2 转换模板文档
 - 3.3 处理模板文档中的占位符
 - 3.4 处理模板文档中的图片
 - 3.5 处理模板文档中的表单
 - 3.6 重命名模板文档
- 4 创建 Java 程序
 - 4.1 版本说明
 - 4.2 创建项目
- 5 测试
 - 5.1 准备
 - 5.2 生成 Word 文档
- 6 踩坑
 - 6.1 特殊符号
 - 6.2 换行符
 - 6.3 内容是 xml 格式
 - 6.4 后缀是 .doc

1 背景

近期工作中需要编写大量格式相同但数据不同的 Word 文档，因此研究了一下 Java 自动生成 Word 文档的技术。

Java 自动生成 Word 文档的技术方案较多，本文使用的是 Java + FreeMarker 的方案，该方案分为两个步骤：创建 FreeMarker 格式的 Word 模板、FreeMarker 根据模板生成 Word 文档。

2 目标效果


本文要使用 Java + FreeMarker 自动生成的 Word 文档的效果如下图所示：

Java 使用 FreeMarker 生成 Word 文档

FreeMarker 普通文本处理：

姓名：用户 1，性别：男，出生日期：1991-01-01！

FreeMarker 表格和图片处理：

姓名	用户 2	
性别	女	
出生日期	1992-02-02	

FreeMarker 表单处理：

姓名	性别	出生日期
用户 3	男	1993-03-03
用户 4	女	1994-04-04

https://blog.csdn.net/weixin_44516305

3 创建 Word 模板


3.1 创建模板文档

Java 使用 FreeMarker 生成 Word 文档

FreeMarker 普通文本处理：

姓名：\${name}，性别：\${sex}，出生日期：\${ birthday}！

FreeMarker 表格和图片处理：

姓名	\${userObj.name}	
性别	\${userObj.sex}	
出生日期	\${userObj.birthday}	

FreeMarker 表单处理：

姓名	性别	出生日期
\${user.name}	\${user.sex}	\${user.birthday}

https://blog.csdn.net/weixin_44516305

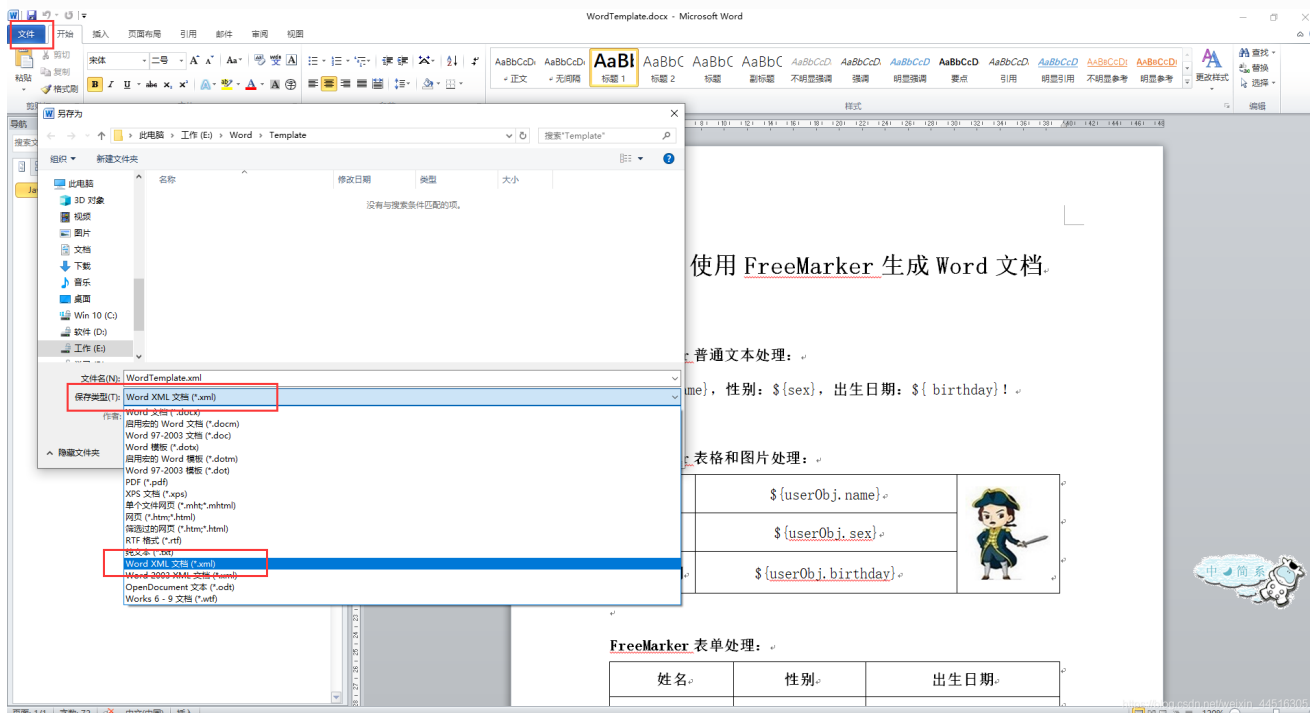
说明：

说明 1：模板文档中 \${} 是占位符，即生成 Word 文档时占位符会被真实的数据替换。例如 \${name} 在生成文档时会被 name 这个属性的值替换，\${userObj.name} 在生成文档时会被 userObj 这个对象的 name 属性的值替换。

说明 2：由于要在生成的 Word 文档中自动插入一张图片，因此，需要在模板文档中插入一张图片作为占位符，如上图所示。

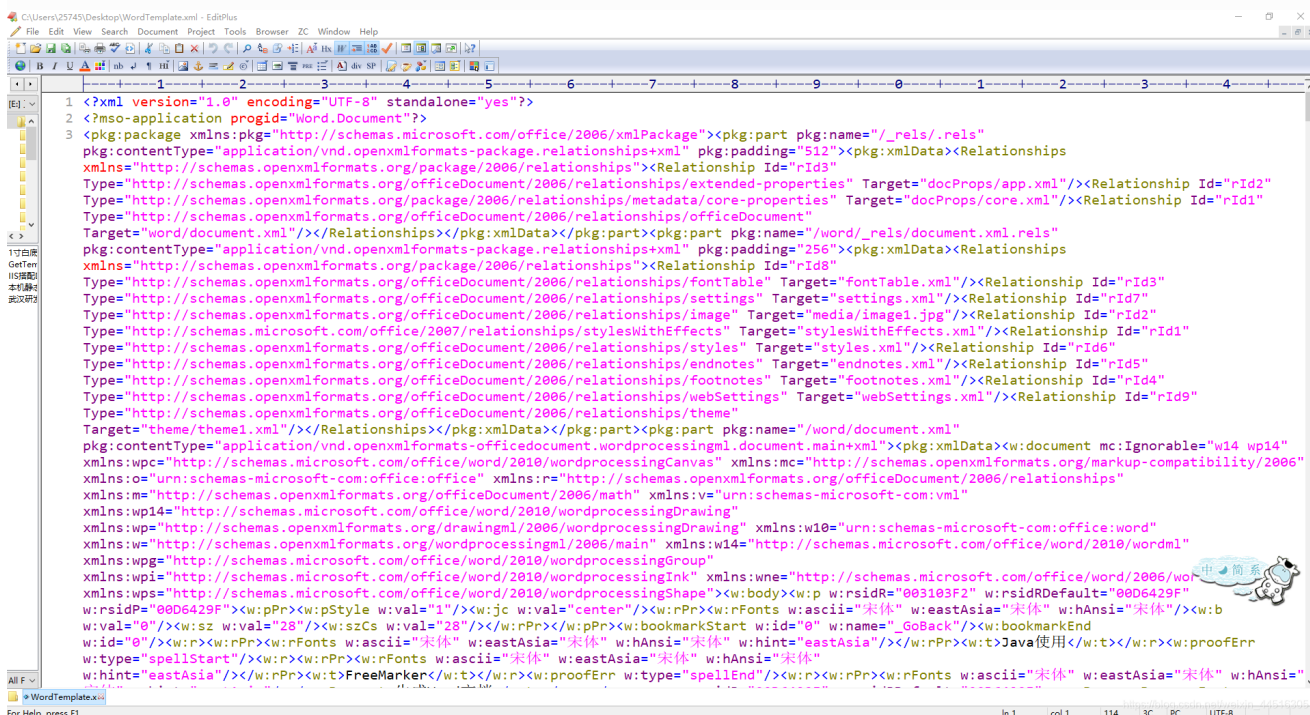
3.2 转换模板文档

使用 Word 将模板文档另存为 Word XML 文档 (*.xml) 格式，如下图所示：

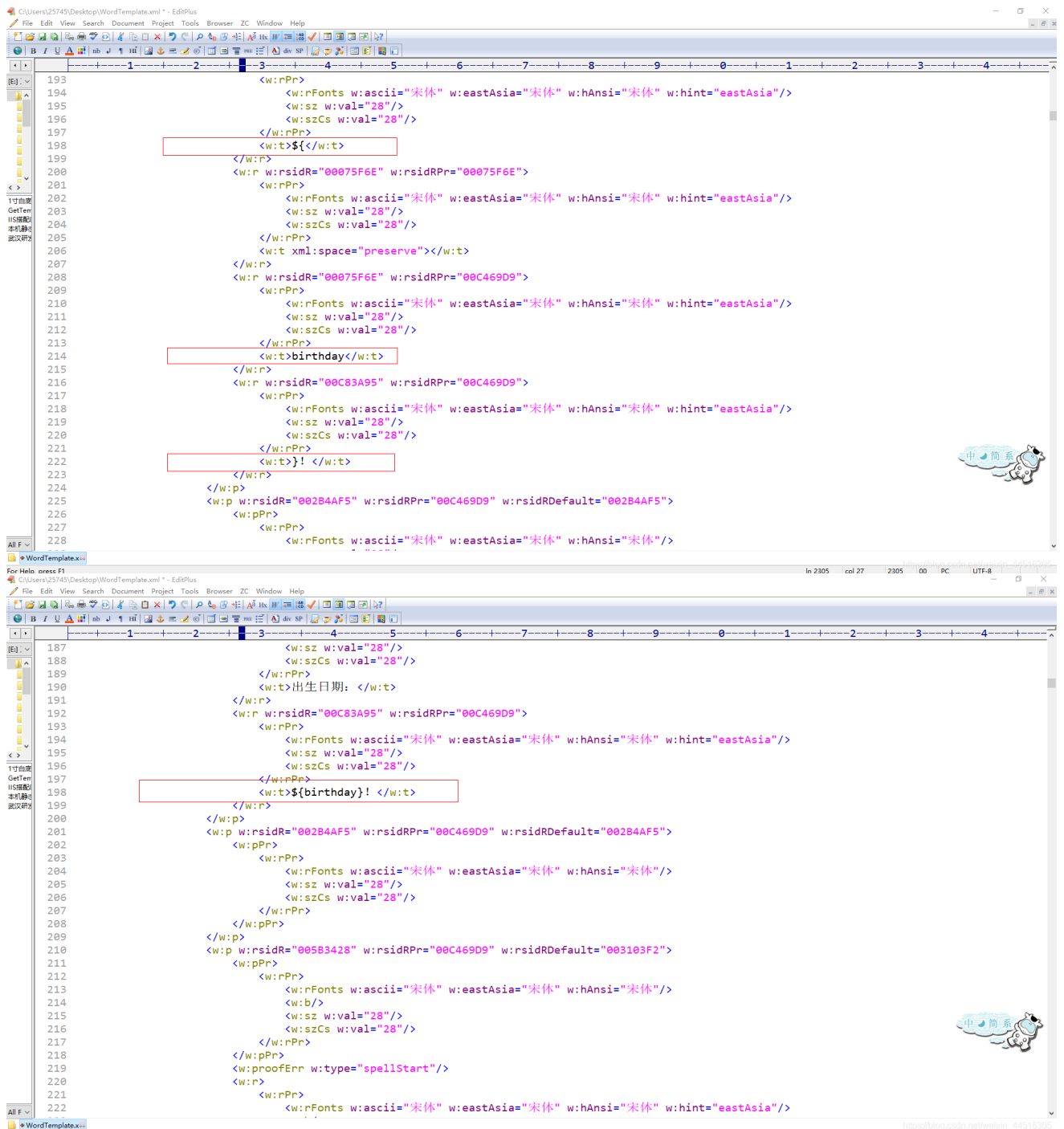


3.3 处理模板文档中的占位符

转换后生成的是一个 WordTemplate.xml 模板文档，使用 EditPlus 等软件打开 WordTemplate.xml 文档（如下图所示），可以发现很不方便阅读，可以借助 xml 在线格式化工具（<http://www.bejson.com/otherformat/xml/>）将 WordTemplate.xml 文档的内容进行格式化。

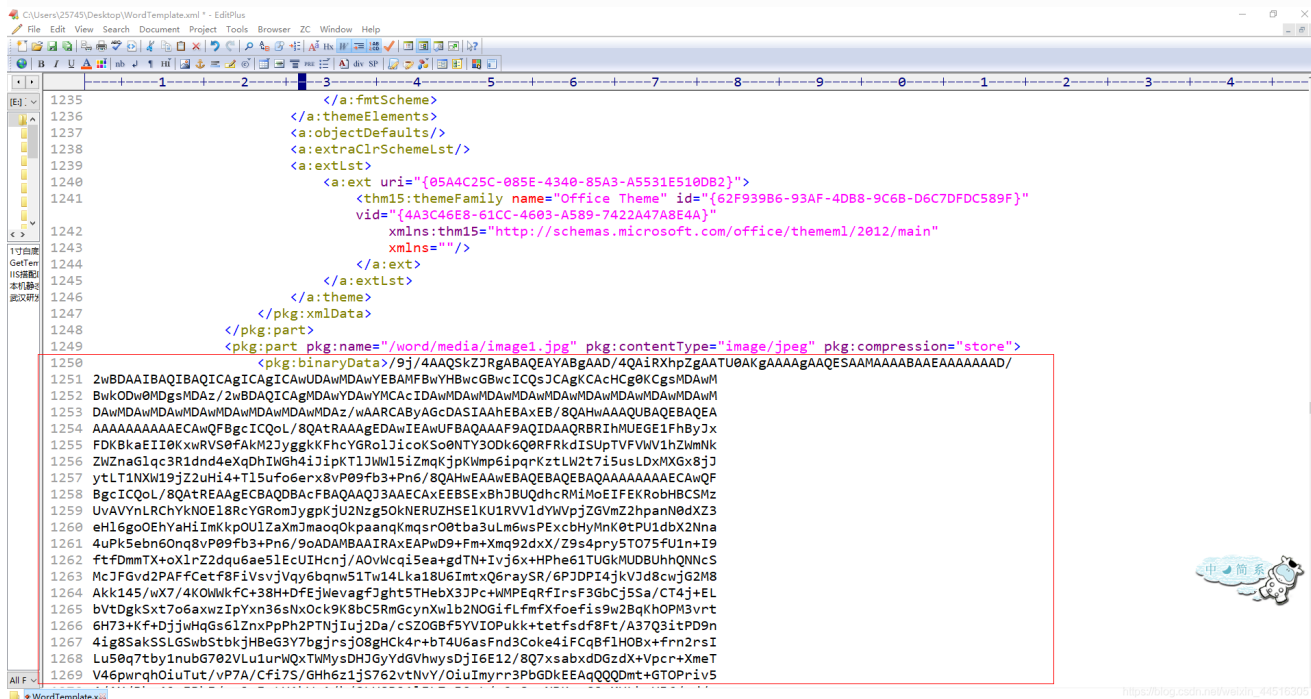


打开格式化后的 WordTemplate.xml 文档可以发现，Word 在转换时会自动的将占位符分开（如下图 1），因此需要把占位符之间多余的部分删除掉（如下图 2）。每个占位符如果被分开了，就需要进行这样的处理，但是如果是一段文字被分开了，就不需要进行处理：

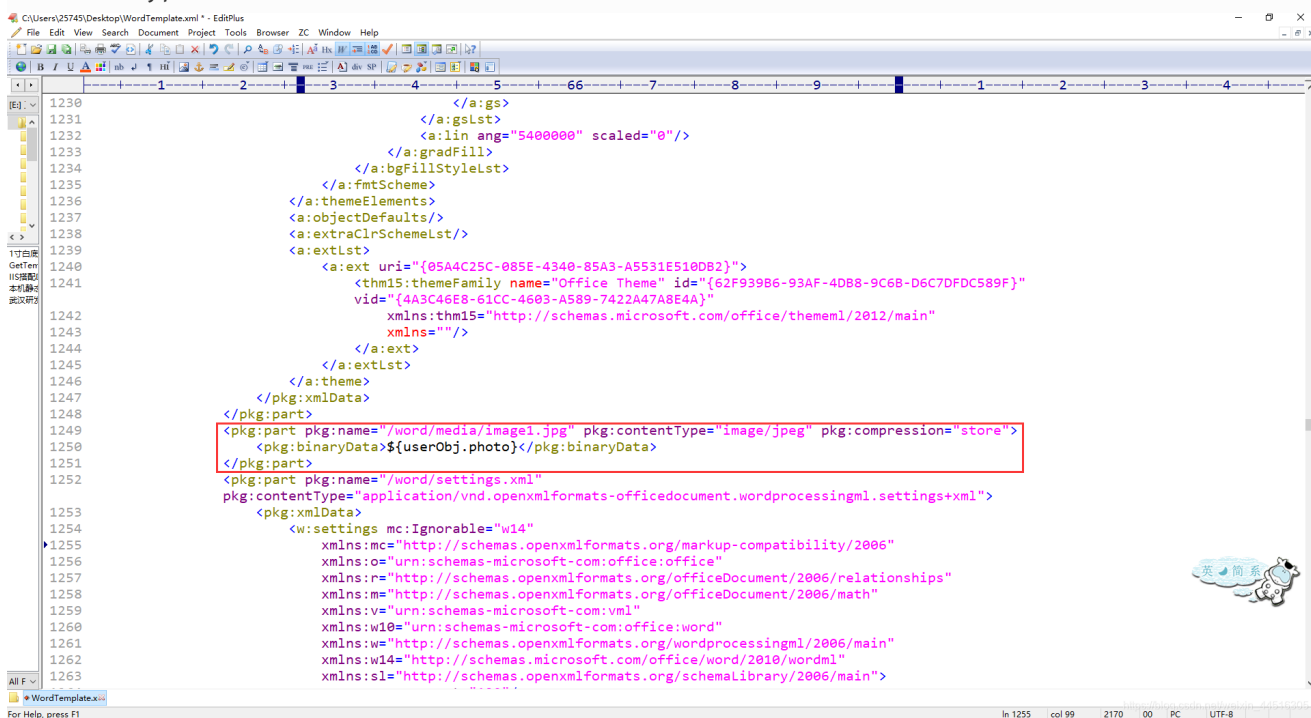


3.4 处理模板文档中的图片

模板文档在转换成 xml 格式时，图片的内容会被转换成很长的 16 进制的字符串，如下图所示：



将 `<pkg:binaryData>` 标签中 16 进制字符串形式的图片内容替换成 `$(userObj.photo)` 占位符（这里的 `userObj.photo` 是与 Java 程序中保持一致），替换后的效果如下图所示：



3.5 处理模板文档中的表单

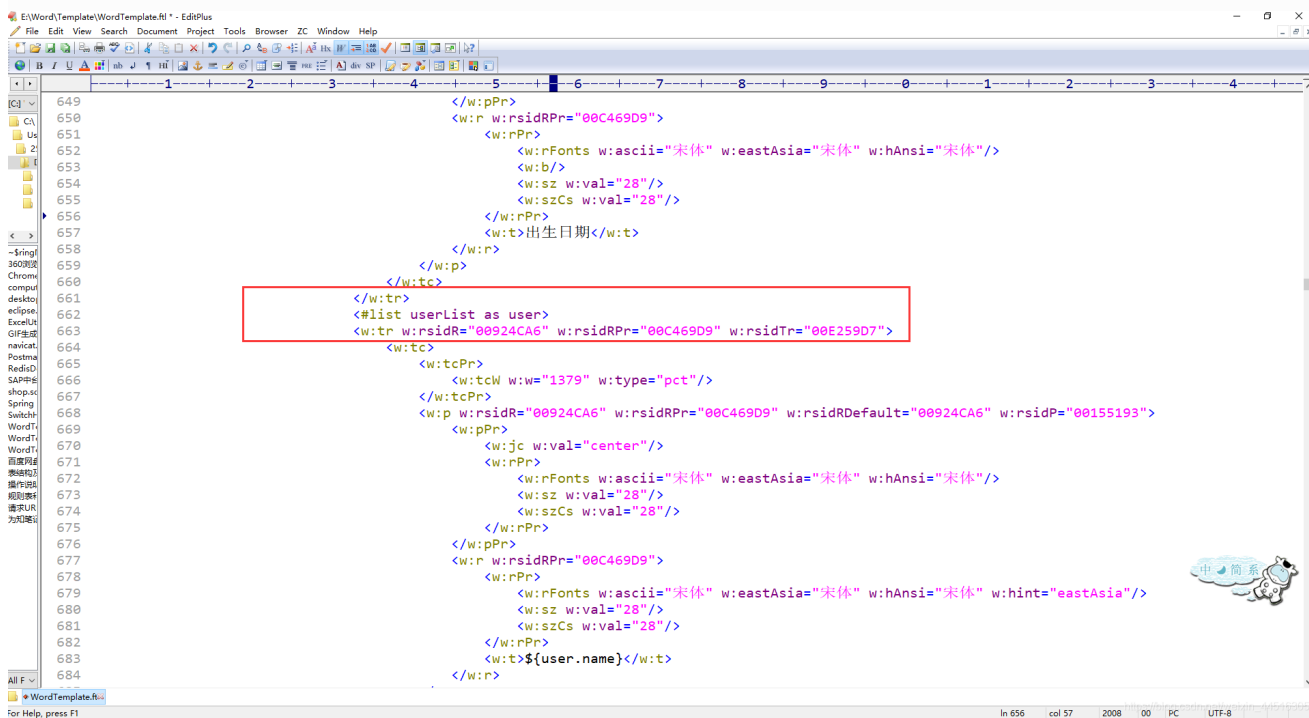
在自动生成 Word 文档中的表单时，由于表头那一行只生成一次，而表单中的数据是循环生成的，因此，需要在 xml 格式的模板文档中表头那一行的后面添加如下的内容：

```
<#list userList as user>
```

并在对应的地方添加如下的内容：

```
</#list>
```

这里的 userList 是和 Java 程序中保存一致，而 user 是和 xml 模板文档中的占位符保持一致，处理后的效果如下图所示：



3.6 重命名模板文档

经过上述处理后，将 WordTemplate.xml 模板文档进行保存，并直接修改后缀名为 ftl，即：WordTemplate.ftl。

至此，Word 模板文档已经创建完成。

4 创建 Java 程序

4.1 版本说明

1. Spring Boot: 2.1.13
2. Freemarker: 2.3.28
3. IDE: IDEA
4. JDK: 1.8

4.2 创建项目

The screenshot shows the IntelliJ IDEA interface with the 'Word' project selected. The Project Explorer on the left displays the following structure:

- 1: Project
 - Word G:\workspace\idea\Word
 - .idea
 - src
 - main
 - java
 - com.office.word
 - model
 - User
 - util
 - ImageUtil
 - WordUtil
 - WordApplication
 - resources
 - test
 - java
 - target
 - .gitignore
 - pom.xml
 - Word.iml

The 'WordApplication' class is highlighted in the 'util' package under the 'com.office.word' package.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.office.word</groupId>
  <artifactId>office-word</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Word</name>
  <description>Java使用FreeMarker自动生成Word文档示例</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.3.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
```



```

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.freemarker</groupId>
            <artifactId>freemarker</artifactId>
            <version>2.3.28</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>

```

User 类程序如下所示：

```

package com.office.word.model;

/**
 * 用户信息封装类
 */
public class User {

    private String name;
    private String sex;
    private String photo;
    private String birthday;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSex() {
        return sex;
    }

    public void setSex(String sex) {
        this.sex = sex;
    }
}

```

```

    }

    public String getPhoto() {
        return photo;
    }

    public void setPhoto(String photo) {
        this.photo = photo;
    }

    public String getBirthday() {
        return birthday;
    }

    public void setBirthday(String birthday) {
        this.birthday = birthday;
    }
}

```

ImageUtil 类程序如下所示：

```

package com.office.word.util;

import org.springframework.util.StringUtils;
import sun.misc.BASE64Encoder;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

/**
 * 图片工具类
 */
public class ImageUtil {

    /**
     * 将图片内容转换成Base64编码的字符串
     * @param imageFile 图片文件的全路径名称
     * @return 转换成Base64编码的图片内容字符串
     */
    public static String getImageBase64String(String imageFile) {
        if (StringUtils.isEmpty(imageFile)) {
            return "";
        }
        File file = new File(imageFile);
        if (!file.exists()) {
            return "";
        }
        InputStream is = null;
        byte[] data = null;
        try {
            is = new FileInputStream(file);
            data = new byte[is.available()];
            is.read(data);
            is.close();
        } catch (IOException e) {

```

```

        e.printStackTrace();
    }

    BASE64Encoder encoder = new BASE64Encoder();
    return encoder.encode(data);
}

}

```

WordUtil 类程序如下所示：

```

package com.office.word.util;

import freemarker.template.Configuration;
import freemarker.template.Template;
import freemarker.template.Version;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.util.Map;

/**
 * Word文档工具类
 */
public class WordUtil {

    /**
     * 使用FreeMarker自动生成Word文档
     * @param dataMap 生成Word文档所需要的数据
     * @param fileName 生成Word文档的全路径名称
     */
    public static void generateWord(Map<String, Object> dataMap, String fileName) throws
Exception {
        // 设置FreeMarker的版本和编码格式
        Configuration configuration = new Configuration(new Version("2.3.28"));
        configuration.setDefaultEncoding("UTF-8");

        // 设置FreeMarker生成Word文档所需要的模板的路径
        configuration.setDirectoryForTemplateLoading(new File("E:/Word/Template/"));
        // 设置FreeMarker生成Word文档所需要的模板
        Template t = configuration.getTemplate("WordTemplate.ftl", "UTF-8");
        // 创建一个Word文档的输出流
        Writer out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(new
File(fileName)), "UTF-8"));
        //FreeMarker使用Word模板和数据生成Word文档
        t.process(dataMap, out);
        out.flush();
        out.close();
    }

}

```

WordApplication 主启动类程序如下所示：

```

package com.office.word;

```

```

import com.office.word.model.User;
import com.office.word.util.ImageUtil;
import com.office.word.util.WordUtil;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Java使用FreeMarker生成Word文档主程序
 */
@SpringBootApplication
public class WordApplication {

    public static void main(String[] args) throws Exception {
        SpringApplication.run(WordApplication.class, args);

        /**
         * 自动生成Word文档
         * 注意：生成的文档的后缀名需要为doc，而不能为docx，否则生成的Word文档会出错
         */
        WordUtil.generateWord(getWordData(), "E:/Word/Document/User.doc");
    }

    /**
     * 获取生成Word文档所需要的数据
     */
    private static Map<String, Object> getWordData() {
        /**
         * 创建一个Map对象，将Word文档需要的数据都保存到该Map对象中
         */
        Map<String, Object> dataMap = new HashMap<>();

        /**
         * 直接在map里保存一个用户的各项信息
         * 该用户信息用于Word文档中FreeMarker普通文本处理
         * 模板文档占位符${name}中的name即指定使用这里的name属性的值"用户1"替换
         */
        dataMap.put("name", "用户1");
        dataMap.put("sex", "男");
        dataMap.put("birthday", "1991-01-01");

        /**
         * 将用户的各项信息封装成对象，然后将对象保存在map中，
         * 该用户对象用于Word文档中FreeMarker表格和图片处理
         * 模板文档占位符${userObj.name}中的userObj即指定使用这里的userObj属性的值(即user2对象)替换
         */
        User user2 = new User();
        user2.setName("用户2");
        user2.setSex("女");
        user2.setBirthday("1992-02-02");
        // 使用FreeMarker在Word文档中生成图片时，需要将图片的内容转换成Base64编码的字符串
        user2.setPhoto(ImageUtil.getImageBase64String("E:/Word/Images/photo.jpg"));
        dataMap.put("userObj", user2);
    }
}

```

```

/*
 * 将多个用户对象封装成List集合，然后将集合保存在map中
 * 该用户集合用于Word文档中FreeMarker表单处理
 * 模板文档中使用<#list userList as user>循环遍历集合，即指定使用这里的userList属性的值(即
userList集合)替换
 */
List<User> userList = new ArrayList<>();
User user3 = new User();
user3.setName("用户3");
user3.setSex("男");
user3.setBirthday("1993-03-03");
User user4 = new User();
user4.setName("用户4");
user4.setSex("女");
user4.setBirthday("1994-04-04");
userList.add(user3);
userList.add(user4);
dataMap.put("userList", userList);

return dataMap;
}
}

```

5 测试

5.1 准备

将要插入到 Word 文档中的图片（photo.jpg）复制到 E:/Word/Images 目录下，并将模板文件 WordTemplate.ftl 复制到 E:/Word/Template 目录下。

5.2 生成 Word 文档

直接运行 WordApplication 主启动类，程序运行成功后，即可以在 E:/Word/Document 目录下自动生成一个名为 User.doc 的文档，打开该文档，即为第 2 节中所示的目标效果。

6 踩坑

6.1 特殊符号

使用 FreeMarker 模板生成 Word 文档时，如果填充的数据字符串中含有特殊字符 <、>、&，那么生成的 Word 文档是无法打开的。因为这些字符在生成 Word 文档时被认为是 FreeMarker 模板的标签，如果这些字符不经过处理就直接用于生成 Word 文档，使用 Word 打开生成的文档就会报错，但以 xml 的方式打开，却会发现所有内容都是完整的，唯独上面三个特殊字符出问题。因此，在处理数据时需要在这三个特殊字符进行处理。

6.2 换行符

使用 FreeMarker 模板生成 Word 文档时，如果填充的数据字符串过长且当中使用 "\n" 进行换行，则生成的 Word 文档中并没有起到换行的作用。需要先将 "\n" 全部替换成 "w:p/w:p"，然后使用替换后的字符串数据生成 Word 文档，才能达到换行的效果。

6.3 内容是 xml 格式

使用本文方法生成的 Word 文档的内容实质上是 xml 格式的，因此，生成的 Word 文档即可以使用 Word 打开，也可以使用 xml 文档工具打开。如果使用 Java 程序去读取生成的 Word 文档的内容，则读出来的也是 xml 格式的内容。如果想要将内容转换成 Word 格式，则可以使用 Word 打开生成的文档然后另存为 Word 文档格式。

6.4 后缀是.doc

使用本文方法生成的 Word 文档的后缀必须是.doc 格式，而不能是.docx 格式，否则生成的 Word 文档无法打开。如果想要将后缀转换成.docx 格式，则可以使用 Word 打开生成的文档然后另存为 Word 文档格式。

如果觉得本文对您有帮助，请关注博主的微信公众号，会经常分享一些 Java 和大数据方面的技术案例！

