docx4j in depth learning

Posted by **bogdan** on *Fri, 06 Mar 2020 05:31:58 +0100*

I. Preface

In my work, I often encounter the processing and operation of word, the most commonly used is the export of word documents, and I often encounter a variety of complex formats of word. At the beginning, I used poi, but I struggled for a period of time, modfa, instead of using docx4j, which is very powerful. Well, I like it, and it took a long time to find the Chinese documents about docx4j on the Internet It's still too few. We are all fragmented blog articles. I think it's time to organize and share, so that more people can enjoy the convenience of docx4j.

docx4j is an open source (ASLv2) Java library for creating and processing Microsoft Open XML (Word docx, Powerpoint pptx and Excel xlsx) files.

Official website of docx4j: https://www.docx4java.org/trac/docx4j

docx4j example code gitHub address: https://github.com/plutext/docx4j

2, Understanding docx files

1. Understand Open XML

docx4j is a Microsoft Open XML file that operates on docx files.

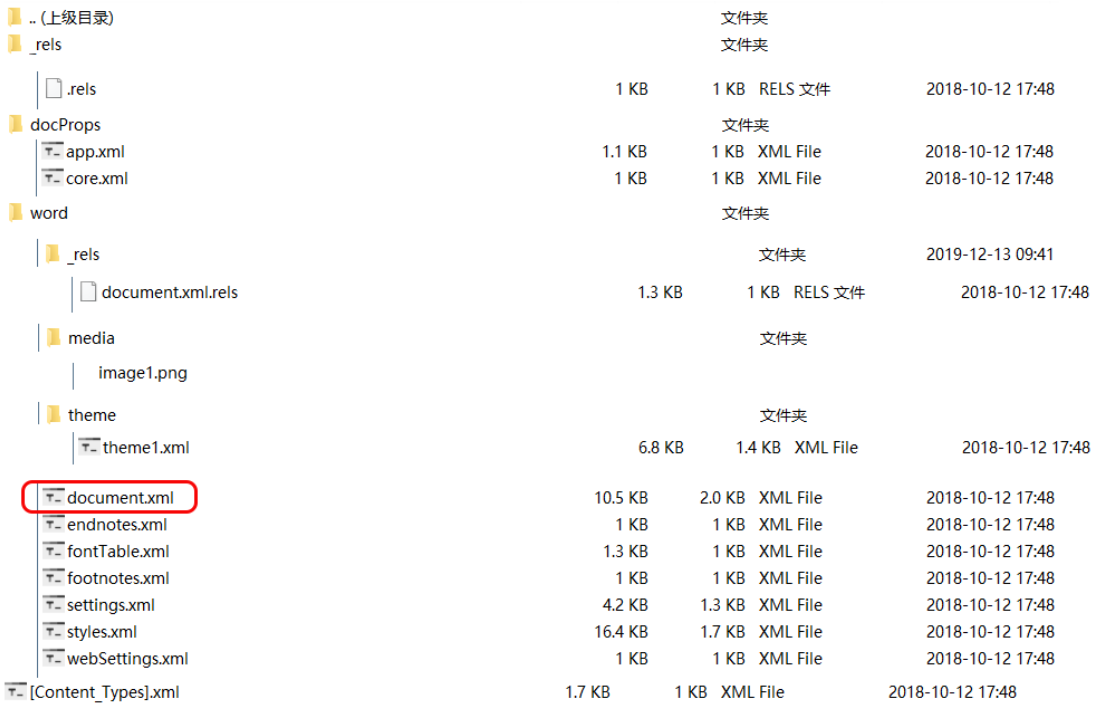
What is Microsoft Open XML?

It's still the old rule. If you don't understand it, go to the official website: http://officeopenxml.com/anatomyofOOXML

Office Open XML, also known as OpenXML or OOXML, is the basic XML format of office documents, which is applied to word, execl, ppt, chart, etc. The specification was developed by Microsoft and adopted by ISO and IEC. It is now the default format for all Microsoft Office documents (. docx,. xlsx, and. pptx).

2. Structure of docx file

Docx can be understood as a compression package of a front-end project. There are style files and xml files in dom format. Let's unzip a docx file and see its directory structure:



Hot Topics
Java - 5554
Python - 2199
Javascript - 1563
Database - 1497
Linux - 1333
Spring - 1284
Back-end - 1194
Algorithm - 1142
Android - 1039
Programming - 1004
Front-end - 919
MySQL - 912
C++ - 905
Attribute - 865
network - 855
xml - 720
data structure - 713
less - 696
github - 666
JSON - 632

Let's first look at the [content [u types]. XML file in the root directory. This is the configuration file of the content component of the entire docx file. The files used in the entire compression package are configured in it. It can be simply understood that when we write the front end, the head part of the html file is about js,css refers to the parts, but this understanding is a bit unclear. Imagine that the total page of each part of the jsp file include s is a bit similar.

_The rels directory is used to configure and define the relationship between various parts,

Looking at the core word directory, you can see that the directory structure is very similar to the html and css structures of our front-end projects,

media directory places multimedia elements, pictures and so on. Just understand

Theme directory, as the name implies, the theme of word, just understand

The most important file in the word directory is the document.xml file

For other word files, settings.xml and styles.xml are configuration and style files of docx files, footTable is font table, footnotes is glossary, other files include footTable at the beginning of foot, head er file and so on

3. Core OpenXML file format of docx

We mainly look at the document.xml file, which is the skeleton of the whole docx, similar to the html file of the front-end page, and is the most basic and important file.

Understand the text structure of document.xml, which is similar to html format

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<w:document>
    <w:body>
        <w:p></w:p>
        <w:tbl></w:tbl>
        <w:sectPr></w:sectPr>
    </w:body>
</w:document>
```

(suddenly I don't want to write about this....., so I'll record it casually.)

3, Using docx4j

1. jar used

maven import

```
        <dependency>
            <groupId>org.docx4j</groupId>
            <artifactId>docx4j</artifactId>
            <version>6.1.2</version>
        </dependency>
```

2. tool class

```
package com.zb.hello.util;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.xml.bind.JAXBElement;
```

```
import javax.xml.bind.JAXBElement;
import javax.xml.bind.JAXBException;

import org.docx4j.XmlUtils;
import org.docx4j.dml.wordprocessingDrawing.Inline;
import org.docx4j.jaxb.XPathBinderAssociationIsPartialException;
import org.docx4j.openpackaging.exceptions.Docx4JException;
import org.docx4j.openpackaging.packages.WordprocessingMLPackage;
import org.docx4j.openpackaging.parts.WordprocessingML.BinaryPartAbstractImage;
import org.docx4j.wml.ContentAccessor;
import org.docx4j.wml.Drawing;
import org.docx4j.wml.ObjectFactory;
import org.docx4j.wml.P;
import org.docx4j.wml.R;
import org.docx4j.wml.Tbl;
import org.docx4j.wml.Tc;
import org.docx4j.wml.Text;
import org.docx4j.wml.Tr;

public class Docx4jUtil {

    public static Builder Builder;

    public static Builder of(String path) throws FileNotFoundException, Docx4JException {
        return new Builder(path);
    }

    public static class Builder {
        private WordprocessingMLPackage template = null;
        private Iterator<Text> texts = null;

        // Placeholder parameter map
        private Map<String, String> params = new HashMap<>();

        private Builder(String path) throws FileNotFoundException, Docx4JException {
            if (path != null && !path.isEmpty()) {
                this.template = WordprocessingMLPackage.load(new FileInputStream(new File(path))
                this.texts = getAllPlaceholderElementFromObject(template.getMainDocumentPart()).
            }
        }

        /**
         * Add text placeholder parameter (one)
         *
         * @param key   key
         * @param value value
         * @return Builder object
         */
        public Builder addParam(String key, String value) {
            Builder builder = this;
            if (key != null && !key.isEmpty()) {
                                /*while (texts.hasNext()) {
                                        Text text = texts.next();
                                        String temp = text.getValue();
                                        if (temp.equals("${" + key + "}")) {
                                                text.setValue(value);
                                                texts.remove();
                                                return builder;
                                        }
                                }*/
                params.put(key, value);
            }
            return builder;
        }
```

```
}

/**
 * Add parameters (multiple)
 *
 * @param params map of multiple parameters
 * @return Builder object
 */
public Builder addParams(Map<String, String> params) {
    this.params = params;
    return this;
}

/**
 * Add a table
 *
 * @param tablePlaceholder Find placeholders for tables
 * @param placeholderRows  Number of template lines
 * @param list             Replace data for template placeholders
 * @return Builder object
 * @throws JAXBException JAXBException
 * @throws Docx4JException Docx4JException
 */
public Builder addTable(String tablePlaceholder, int placeholderRows, List<Map<String, St
        throws Docx4JException, JAXBException {
    List<Object> tables = getAllElementFromObject(template.getMainDocumentPart(), Tbl.cla

    Tbl tempTable = getTemplateTable(tables, tablePlaceholder);
    if (tempTable != null && list != null && !list.isEmpty()) {
        List<Object> trs = getAllElementFromObject(tempTable, Tr.class);
        int rows = trs.size();

        if (rows > placeholderRows) {
            List<Tr> tempTrs = new ArrayList<>();
            for (int i = rows - placeholderRows; i < rows; i++) {
                tempTrs.add((Tr) trs.get(i));
            }

            for (Map<String, String> trData : list) {
                for (Tr tempTr : tempTrs) {
                    addRowToTable(tempTable, tempTr, trData);
                }
            }

            for (Tr tempTr : tempTrs) {
                tempTable.getContent().remove(tempTr);
            }
        }
    }
    return this;

}

private void loadImg(Tbl tempTable, byte[] decodeBuffer, int maxWidth) {
    Inline inline = createInlineImage(template, decodeBuffer, maxWidth);
    P paragraph = addInlineImageToParagraph(inline);
    List<Object> rows = getAllElementFromObject(tempTable, Tr.class);
    Tr tr = (Tr) rows.get(0);
    List<Object> cells = getAllElementFromObject(tr, Tc.class);
    Tc tc = (Tc) cells.get(0);
    tc.getContent().clear();
    tc.getContent().add(paragraph);
}
```

```
/**
 * Determine the loading position of the picture through the stop sign. The position of t
 *
 * @param placeholder  placeholder
 * @param decodeBuffer Byte stream of pictures
 * @return Current object
 * @throws Docx4JException Docx4JException
 * @throws JAXBException   JAXBException
 */
public Builder addImg(String placeholder, byte[] decodeBuffer) throws Docx4JException, JA
    addImg(placeholder, decodeBuffer, 0);
    return this;
}

/**
 * Determine the loading position of the picture through the stop sign. The position of t
 *
 * @param placeholder  placeholder
 * @param decodeBuffer Byte stream of pictures
 * @param maxWidth     The maximum width of the picture. The original width of the pictu
 * @return Current object
 * @throws Docx4JException Docx4JException
 * @throws JAXBException   JAXBException
 */
public Builder addImg(String placeholder, byte[] decodeBuffer, int maxWidth) throws Docx4
    List<Object> tables = getAllElementFromObject(template.getMainDocumentPart(), Tbl.cla
    Tbl tempTable = getTemplateTable(tables, placeholder);
    loadImg(tempTable, decodeBuffer, maxWidth);
    return this;
}

/**
 * Determine the position of the loaded image through the position array of int, and the
 *
 * @param wz           int Type array, length must be 3, the first value is the table, t
 * @param decodeBuffer Byte stream of pictures
 * @param maxWidth     The maximum width of the picture. The original width of the pictu
 * @return Current object
 */
public Builder addImg(int[] wz, byte[] decodeBuffer, int maxWidth) {
    Tc tc = getTcByWz(wz);
    Tbl tempTable = (Tbl) getAllElementFromObject(tc, Tbl.class).get(0);
    loadImg(tempTable, decodeBuffer, maxWidth);
    return this;
}

/**
 * Determine the position of the loaded image through the position array of int, and the
 *
 * @param wz           int Type array, length must be 3, the first value is the table, t
 * @param decodeBuffer Byte stream of pictures
 * @return Current object
 */
public Builder addImg(int[] wz, byte[] decodeBuffer) {
    addImg(wz, decodeBuffer, 0);
    return this;
}

/**
 * Add paragraphs
 *
 * @param list Data set
 * @param wz   The position of the template paragraph. The length is three (the first ta
```

```
     * @return Builder object
     */
    public Builder addParagrash(List<Map<String, String>> list, int[] wz) {
        Tc tc = getTcByWz(wz);
        List<Object> paraList = getAllElementFromObject(tc, P.class);
        tc.getContent().clear();
        for (Map<String, String> item : list) {
            paraList.forEach((tempPara) -> {
                P workingPara = (P) XmlUtils.deepCopy(tempPara);
                repaleTexts(workingPara, item);
                tc.getContent().add(workingPara);
            });
        }
        return this;
    }

    /**
     * Remove Tr with placeholder
     * @param placeholder placeholder
     * @return Builder object
     */
    public Builder removeTrByPlaceholder(String placeholder) {
        //This way of getting is normal, but the get() method cannot replace the text normal
        //List<Object> trs = template.getMainDocumentPart().getJAXBNodesViaXPath("//w:tr", t
        List<Object> trs = getAllElementFromObject(template.getMainDocumentPart(), Tr.class)
        Tr tr = (Tr) getTemplateObj(trs,placeholder,false);
        if(tr != null){
            Tbl tbl = (Tbl) tr.getParent();
            tbl.getContent().remove(tr);
        }
        return this;
    }

    /**
     * Remove Tr with placeholder
     * @param placeholders Collection of placeholders
     * @return Builder object
     */
    public Builder removeTrByPlaceholder(List<String> placeholders) {
       /* List<Object> trs = template.getMainDocumentPart().getJAXBNodesViaXPath("//w:tr", t
        List<Object> trs = getAllElementFromObject(template.getMainDocumentPart(), Tr.class)
        List<Object> list = getTemplateObjs(trs,placeholders);
        for (Object o:list) {
            Tr tr = (Tr) o;
            if(tr != null){
                Tbl tbl = (Tbl) tr.getParent();
                tbl.getContent().remove(tr);
            }
        }
        return this;
    }

    /**
     * Get file byte stream
     *
     * @return File byte stream
     * @throws Docx4JException docx abnormal
     */
    public byte[] get() throws Docx4JException {
        if (!params.isEmpty()) {
            while (texts.hasNext()) {
                Text text = texts.next();
                String temp = text.getValue();
```

```
                for (Entry<String, String> param : params.entrySet()) {
                    if (temp.equals("${" + param.getKey() + "}")) {
                        text.setValue(param.getValue());
                    }
                }
            }
        }

        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        template.save(outputStream);
        return outputStream.toByteArray();
    }

    /**
     * Get the specified cell
     *
     * @param temp The position of the template paragraph. The length is three (the first ta
     * @return tc
     */
    private Tc getTcByWz(int[] temp) {
        List<Object> tables = getAllElementFromObject(template.getMainDocumentPart(), Tbl.cl
        Tbl wzTable = (Tbl) tables.get(temp[0]);
        Tr tr = (Tr) getAllElementFromObject(wzTable, Tr.class).get(temp[1]);
        return (Tc) getAllElementFromObject(tr, Tc.class).get(temp[2]);
    }

}

/**
 * Create an inline object that contains a picture
 *
 * @param wordMLPackage WordprocessingMLPackage
 * @param bytes Picture byte stream
 * @param maxWidth Maximum width
 * @return Inline objects for pictures
 */
private static Inline createInlineImage(WordprocessingMLPackage wordMLPackage, byte[] bytes,
    Inline inline = null;
    try {
        BinaryPartAbstractImage imagePart = BinaryPartAbstractImage.createImagePart(wordMLPa
        int docPrId = 1;
        int cNvPrId = 2;
        if (maxWidth > 0) {
            inline = imagePart.createImageInline("Filename hint", "Alternative text", docPrI
        } else {
            inline = imagePart.createImageInline("Filename hint", "Alternative text", docPrI
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return inline;
}

/**
 * Create an object factory and use it to create a paragraph and a runnable block R. then ad
 * Object to add to the drawing and return the paragraph object
 *
 * @param inline Inline object containing the picture
 * @return Paragraph with picture
 */
private static P addInlineImageToParagraph(Inline inline) {
    // Add inline objects to a paragraph
    ObjectFactory factory = new ObjectFactory();
```

```
        P paragraph = factory.createP();
        R run = factory.createR();
        paragraph.getContent().add(run);
        Drawing drawing = factory.createDrawing();
        run.getContent().add(drawing);
        drawing.getAnchorOrInline().add(inline);
        return paragraph;
    }

    // Found text nodes with placeholders in docx documents
    private static List<Text> getAllPlaceholderElementFromObject(Object obj) {
        List<Text> result = new ArrayList<>();
        Class<Text> toSearch = Text.class;
        Text textPlaceholder;
        if (obj instanceof JAXBElement) {
            obj = ((JAXBElement<?>) obj).getValue();
        }
        if (obj.getClass().equals(toSearch)) {
            textPlaceholder = (Text) obj;
            if (isPlaceholder(textPlaceholder.getValue())) {
                result.add((Text) obj);
            }
        } else if (obj instanceof ContentAccessor) {
            List<?> children = ((ContentAccessor) obj).getContent();
            for (Object child : children) {
                result.addAll(getAllPlaceholderElementFromObject(child));
            }
        }
        return result;
    }

    // Discover nodes in docx documents
    private static List<Object> getAllElementFromObject(Object obj, Class<?> toSearch) {
        List<Object> result = new ArrayList<>();
        if (obj instanceof JAXBElement) {
            obj = ((JAXBElement<?>) obj).getValue();
        }
        if (obj.getClass().equals(toSearch)) {
            result.add(obj);
        } else if (obj instanceof ContentAccessor) {
            List<?> children = ((ContentAccessor) obj).getContent();
            for (Object child : children) {
                result.addAll(getAllElementFromObject(child, toSearch));
            }
        }
        return result;
    }

    // This method only checks whether the table contains our placeholders, and if so, returns t
    private static Tbl getTemplateTable(List<Object> tables, String templateKey) {
        return (Tbl) getTemplateObj(tables,templateKey,false);
    }

    /**
     * This method only checks whether the dom contains our placeholders, and if so, returns the
     *
     * @param objects     dom elements to find
     * @param placeholder placeholder
     * @param f           Whether to search all or not. When it is true, it will search all. Whe
     * @return Elements found
     */
    private static Object getTemplateObj(List<Object> objects, String placeholder, boolean f) {
        List<Object> objectList = new ArrayList<>();
```

```
        for (Object o : objects) {
            List<?> textElements = getAllElementFromObject(o, Text.class);
            for (Object text : textElements) {
                Text textElement = (Text) text;
                if (textElement.getValue() != null && textElement.getValue().equals("${" + place
                    if (!f) {
                        return o;
                    } else {
                        objectList.add(o);
                    }
                }
            }
        }
        return objectList.isEmpty()?null:objectList;
    }

    /**
     *  This method only checks whether the dom contains our placeholders, and if so, returns th
     * @param objects The collection of dom elements to find
     * @param placeholders Placeholder collection
     * @return Collection of elements found
     */
    private static List<Object> getTemplateObjs(List<Object> objects, List<String> placeholders)
        List<Object> objectList = new ArrayList<>();
        for (Object o : objects) {
            List<?> textElements = getAllElementFromObject(o, Text.class);
            for (Object text : textElements) {
                Text textElement = (Text) text;
                if (textElement.getValue() != null && placeholders.contains(getPlaceholderStr(te
                    objectList.add(o);
                }
            }
        }
        return objectList;
    }

    /**
     * Copy template lines
     *
     * @param reviewtable  form
     * @param templateRow  Template row
     * @param replacements Fill in data for template lines
     */
    private static void addRowToTable(Tbl reviewtable, Tr templateRow, Map<String, String> repla
        Tr workingRow = XmlUtils.deepCopy(templateRow);
        repaleTexts(workingRow, replacements);
        reviewtable.getContent().add(workingRow);
    }

    /**
     * Replace all placeholders in the working object
     *
     * @param working      Working object
     * @param replacements map data object
     */
    private static void repaleTexts(Object working, Map<String, String> replacements) {
        List<?> textElements = getAllElementFromObject(working, Text.class);
        for (Object object : textElements) {
            Text text = (Text) object;
            String keyStr = getPlaceholderStr(text.getValue());
            if (keyStr != null && !keyStr.isEmpty()) {
                String replacementValue = replacements.get(keyStr);
                if (replacementValue != null) {
```

```
                    text.setValue(replacementValue);
                } else {
                    text.setValue("--");
                }
            }
        }
    }

    /**
     * Determine if the string has ${} placeholders
     *
     * @param str String to judge
     * @return Whether the string has ${} placeholders
     */
    private static boolean isPlaceholder(String str) {
        if (str != null && !str.isEmpty()) {
            Pattern pattern = Pattern.compile("([$]\\{\\w+\\})");
            Matcher m = pattern.matcher(str);
            return m.find();
        }
        return false;
    }

    /**
     * Get text in placeholder ${}
     *
     * @param str String to judge
     * @return Text in placeholder ${}
     */
    private static String getPlaceholderStr(String str) {
        if (str != null && !str.isEmpty()) {
            Pattern p = Pattern.compile("\\$\\{(.*?)\\}");
            Matcher m = p.matcher(str);
            if (m.find()) {
                return m.group(1);//m.group(0) includes these two characters
            }
        }
        return null;
    }
}
```

3. Export word file for use

```
public byte[] downloadWord() throws Docx4JException, JAXBException, FileNotFoundException {
                //Template file path
                String path = this.getClass().getClassLoader().getResource("template/test.docx")
                //Data in template to generate table
                List<Map<String, String>> list = new ArrayList<>();
                for (int i = 0; i < 3; i++) {
                        Map<String, String> m = new HashMap<>();
                        m.put("name", "Full name"+i);
                        m.put("sex", "Gender"+i);
                        m.put("age", "Age"+i);
                        m.put("bz", "Remarks"+i);
                        m.put("xx", "detailed"+i);
                        list.add(m);
                }
                list.stream();

                //Data to insert picture into template
                byte[] img = null;
                try (InputStream input = new FileInputStream(this.getClass().getClassLoader().ge
```

```
                try (InputStream input = new FileInputStream(this.getClass().getClassLoader().ge
                        img = new byte[input.available()];
                        input.read(img);
                } catch (Exception e) {
                        e.printStackTrace();
                }

                //map data to insert
                Map<String, String> m = new HashMap<>();
                m.put("today", LocalDate.now().toString());
                m.put("active", "Swimming");

                //Super simple call after data processing
                return Docx4jUtil.of(path)
                                        .addParam("title", "Test document title")
                                        .addParam("user", "Tester")
                                        .addParams(m)
                                        .addTable("name", 2, list)
                                        .addImg("img", img)
                                        .get();
        }
```

Four, expand

1.word to html

maven using poi

```
<!-- word turn html Of poi -->
        <dependency>
            <groupId>fr.opensagres.xdocreport</groupId>
            <artifactId>xdocreport</artifactId>
            <version>2.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>${poi-version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-scratchpad</artifactId>
            <version>${poi-version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>${poi-version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml-schemas</artifactId>
            <version>${poi-version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>ooxml-schemas</artifactId>
            <version>1.4</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
```

```
            <artifactId>commons-compress</artifactId>
            <version>1.18</version>
        </dependency>
```

2. tool class

```
package com.zb.hello.util;

import fr.opensagres.poi.xwpf.converter.xhtml.Base64EmbedImgManager;
import fr.opensagres.poi.xwpf.converter.xhtml.XHTMLConverter;
import fr.opensagres.poi.xwpf.converter.xhtml.XHTMLOptions;
import org.apache.commons.codec.binary.Base64;
import org.apache.poi.hwpf.HWPFDocumentCore;
import org.apache.poi.hwpf.converter.PicturesManager;
import org.apache.poi.hwpf.converter.WordToHtmlConverter;
import org.apache.poi.hwpf.converter.WordToHtmlUtils;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.w3c.dom.Document;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class WordHtmlUtil {
    //doc conversion html
    public static String docToHtml(File file) throws IOException, ParserConfigurationException, 
        HWPFDocumentCore wordDocument = WordToHtmlUtils.loadDoc(new FileInputStream(file));
        WordToHtmlConverter wordToHtmlConverter = new WordToHtmlConverter(
                DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument()
        );
        PicturesManager pictureRunMapper = (bytes, pictureType, s, v, v1) -> "data:image/png;bas
        wordToHtmlConverter.setPicturesManager(pictureRunMapper);
        wordToHtmlConverter.processDocument(wordDocument);
        Document htmlDocument = wordToHtmlConverter.getDocument();
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        DOMSource domSource = new DOMSource(htmlDocument);
        StreamResult streamResult = new StreamResult(out);
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer serializer = transformerFactory.newTransformer();
        serializer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        serializer.setOutputProperty(OutputKeys.INDENT, "yes");
        serializer.setOutputProperty(OutputKeys.METHOD, "html");
        serializer.transform(domSource, streamResult);
        out.close();
        return new String(out.toByteArray());
    }

    //docx transform html
    public static String docxToHtml(File file) throws IOException {
        XWPFDocument docxDocument = new XWPFDocument(new FileInputStream(file));
        XHTMLOptions options = XHTMLOptions.create();
        //Picture to base64
        options.setImageManager(new Base64EmbedImgManager());
        // Transform html11
```

```
        ByteArrayOutputStream htmlStream = new ByteArrayOutputStream();
        XHTMLConverter.getInstance().convert(docxDocument, htmlStream, options);
        htmlStream.close();
        return new String(htmlStream.toByteArray());
    }
}
```

Topics: Programming xml Java Apache github