第四章 高级文件处理

XML/docx/xls/csv/pdf

XML简介

基本概念

• 可扩展标记语言：意义+数据
• 标签可自行定义，具有自我描述性
• 纯文本表示，跨系统/平台/语言
• W3C标准(1998年，W3C发布了XML1.0，包括几乎所有的
Unicode字符)

语法

– 任何的起始标签都必须有一个结束标签。
– 简化写法，例如，可以写为。
– 大小写敏感，如和不一样。
– 每个文件都要有一个根元素。
– 标签必须按合适的顺序进行嵌套，不可错位。
– 所有的特性都必须有值，且在值的周围加上引号。
– 需要转义字符，如“<”需要用<代替。
– 注释：

XML解析

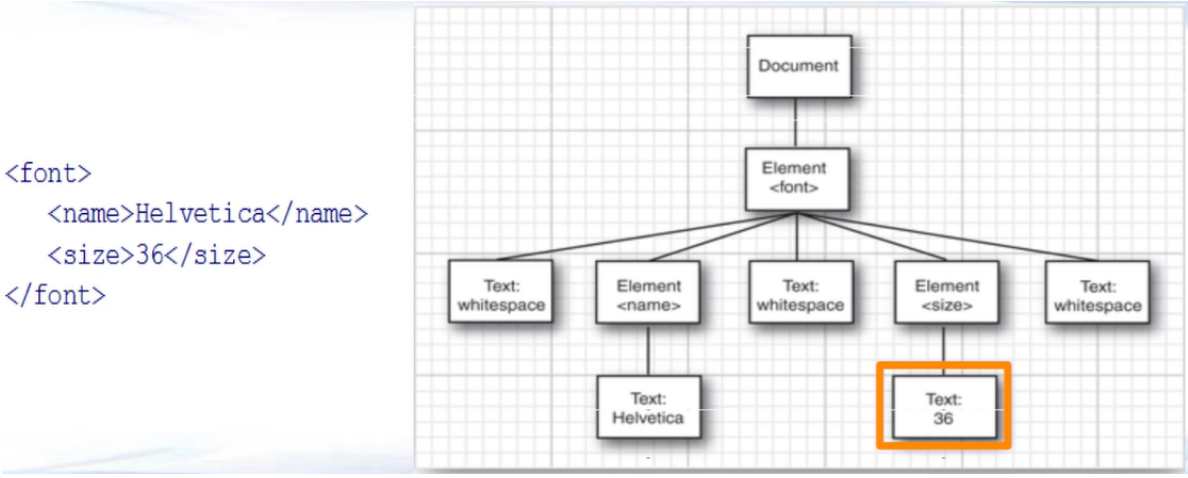
方式1 DOM解析

特征

DOM 是 W3C 处理 XML 的标准 API
– 直观易用。
– 其处理方式是将 XML 整个作为类似树结构的方式读入内存中以便操作及解析，方便修改。
– 解析大数据量的 XML 文件，会遇到内存泄露及程序崩溃的风险。

结构示例

标签与标签之间的空格也会被上级父节点视为子元素



Dom类

- DocumentBuilder 解析类，parse方法
- Node 节点主接口，getChildNodes返回一个NodeList
- NodeList 节点列表，每个元素是一个Node
- Document 文档根节点
- Element 标签节点元素 (每一个标签都是标签节点)
- Text节点 (包含在XML元素内的，都算Text节点)
- Attr节点(每个属性节点)

程序实现

import

```
//导入包
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

//
public class DomReader
{
    public static void main(String[] a)
    {
        recursiveTraverse(); //自上而下进行访问
        System.out.println("========华丽丽的分割线==========");
        traverseBySearch();     //根据名称进行搜索
    }
}
```

自上而下访问

```
public static void recursiveTraverse()
{

    //采用Dom解析xml文件
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

```
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document document = db.parse("users.xml");

    //获取所有的一级子节点
    NodeList usersList = document.getChildNodes();
    System.out.println(usersList.getLength());  //1

    for (int i = 0; i < usersList.getLength(); i++)
    {
        Node users = usersList.item(i);         //1  users

        NodeList userList = users.getChildNodes(); //获取二级子节点user的列表
        System.out.println("==" + userList.getLength()); //9

        for (int j = 0; j < userList.getLength(); j++) //9
        {
            Node user = userList.item(j);
            if (user.getNodeType() == Node.ELEMENT_NODE)
            {
                NodeList metaList = user.getChildNodes();
                System.out.println("====" + metaList.getLength()); //7

                for (int k = 0; k < metaList.getLength(); k++) //7
                {
                    //到最后一级文本
                    Node meta = metaList.item(k);
                    if (meta.getNodeType() == Node.ELEMENT_NODE)
                    {
                        System.out.println(metaList.item(k).getNodeName()+ ":"
+metaList.item(k).getTextContent());
                    }

                }
                System.out.println();
            }
        }
    }

}
```

根据名称搜索

```
    public static void traverseBySearch()
    {
        try
        {
            //采用Dom解析xml文件
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document document = db.parse("users.xml");

            Element rootElement = document.getDocumentElement();

            NodeList nodeList = rootElement.getElementsByTagName("name");
            if(nodeList != null)
            {
               for (int i = 0 ; i < nodeList.getLength(); i++)
```

```
                {
                    Element element = (Element)nodeList.item(i);
                    System.out.println(element.getNodeName() + " = " +
element.getTextContent());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

方式2 SAX

Simple API for XML

– 采用事件/流模型来解析 XML 文档，更快速、更轻量。
– 有选择的解析和访问，不像 DOM 加载整个文档，内存要求较低。
– SAX 对 XML 文档的解析为一次性读取，不创建/不存储文档对象，很难同时访问文档中的多处数据。
– 推模型。当它每发现一个节点就引发一个事件，而我们需要编写这些事件的处理程序。

程序实现

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;

public class SAXReader {
    public static void main(String[] args) throws SAXException, IOException {
        XMLReader parser = XMLReaderFactory.createXMLReader();
        BookHandler bookHandler = new BookHandler();
        parser.setContentHandler(bookHandler);
        parser.parse("books.xml");
        System.out.println(bookHandler.getNameList());
    }
}

class BookHandler extends DefaultHandler {
    private List<String> nameList;
    private boolean title = false;

    public List<String> getNameList() {
        return nameList;
    }
```

```java
    // xml文档加载时
    public void startDocument() throws SAXException {
        System.out.println("Start parsing document...");
        nameList = new ArrayList<String>();
    }

    // 文档解析结束
    public void endDocument() throws SAXException {
        System.out.println("End");
    }

    // 访问某一个元素
    public void startElement(String uri, String localName, String qName,
Attributes atts) throws SAXException {

        if (qName.equals("title")) {
            title = true;
        }
    }

    // 结束访问元素
    public void endElement(String namespaceURI, String localName, String qName)
throws SAXException {
        // End of processing current element
        if (title) {
            title = false;
        }
    }

    // 访问元素正文
    public void characters(char[] ch, int start, int length) {

        if (title) {
            String bookTitle = new String(ch, start, length);
            System.out.println("Book title: " + bookTitle);
            nameList.add(bookTitle);
        }
    }

}
```

方式3 Stax

Streaming API for XML

– 流模型中的拉模型
– 在遍历文档时，会把感兴趣的部分从读取器中拉出，不需要引发事件，允许我们选择性地处理节点。这大大提高了灵活性，以及整体效率。
– 两套处理API
• 基于指针的API， XMLStreamReader
• 基于迭代器的API，XMLEventReader

程序实现

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Iterator;

import javax.xml.stream.XMLEventReader;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamReader;
import javax.xml.stream.events.Attribute;
import javax.xml.stream.events.EndElement;
import javax.xml.stream.events.StartElement;
import javax.xml.stream.events.XMLEvent;


public class StaxReader {

    public static void main(String[] args) {
        StaxReader.readByStream();
        System.out.println("========华丽丽的分割线==========");
        StaxReader.readByEvent();
    }

    //流模式
    public static void readByStream() {
        String xmlFile = "books.xml";
        XMLInputFactory factory = XMLInputFactory.newFactory();
        XMLStreamReader streamReader = null;
        try {
            streamReader = factory.createXMLStreamReader(new
FileReader(xmlFile));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (XMLStreamException e) {
            e.printStackTrace();
        }

        // 基于指针遍历
        try {
            while (streamReader.hasNext()) {
                int event = streamReader.next();
                // 如果是元素的开始
                if (event == XMLStreamConstants.START_ELEMENT) {
                    // 列出所有书籍名称
                    if ("title".equalsIgnoreCase(streamReader.getLocalName())) {
                        System.out.println("title:" +
streamReader.getElementText());
                    }
                }
            }
            streamReader.close();
        } catch (XMLStreamException e) {
            e.printStackTrace();
        }
    }
```

```java
    // 事件模式
    public static void readByEvent() {
        String xmlFile = "books.xml";
        XMLInputFactory factory = XMLInputFactory.newInstance();
        boolean titleFlag = false;
        try {
            // 创建基于迭代器的事件读取器对象
            XMLEventReader eventReader = factory.createXMLEventReader(new
FileReader(xmlFile));
            // 遍历Event迭代器
            while (eventReader.hasNext()) {
                XMLEvent event = eventReader.nextEvent();
                // 如果事件对象是元素的开始
                if (event.isStartElement()) {
                    // 转换成开始元素事件对象
                    StartElement start = event.asStartElement();
                    // 打印元素标签的本地名称

                    String name = start.getName().getLocalPart();
                    //System.out.print(start.getName().getLocalPart());
                    if(name.equals("title"))
                    {
                        titleFlag = true;
                        System.out.print("title:");
                    }

                    // 取得所有属性
                    Iterator attrs = start.getAttributes();
                    while (attrs.hasNext()) {
                        // 打印所有属性信息
                        Attribute attr = (Attribute) attrs.next();
                        //System.out.print(":" + attr.getName().getLocalPart() +
"=" + attr.getValue());
                    }
                    //System.out.println();
                }
                //如果是正文
                if(event.isCharacters())
                {
                    String s = event.asCharacters().getData();
                    if(null != s && s.trim().length()>0 && titleFlag)
                    {
                        System.out.println(s.trim());
                    }
                }
                //如果事件对象是元素的结束
                if(event.isEndElement())
                {
                    EndElement end = event.asEndElement();
                    String name = end.getName().getLocalPart();
                    if(name.equals("title"))
                    {
                        titleFlag = false;
```

```
                    }
                }
            }
            eventReader.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (XMLStreamException e) {
            e.printStackTrace();
        }
    }
}
```

docx

简述

• 以Microsoft Office的doc/docx为主要处理对象。
• Word2003（包括）之前都是doc，文档格式不公开。
• Word2007（包括）之后都是docx，遵循XML路线，文档格式公开。
• docx为主要研究对象
– 文字样式
– 表格
– 图片
– 公式

常见功能

– docx解析
– docx生成（完全生成，模板加部分生成：套打）
• 处理的第三方库
– Jacob, COM4J (Windows 平台)
– POI, docx4j, OpenOffice/Libre Office SDK (免费)
– Aspose (收费)
– 一些开源的OpenXML的包。

Apache POI

– Apache 出品，必属精品， poi.apache.org
– 可处理docx, xlsx, pptx, visio等office套件
– 纯Java工具包，无需第三方依赖
– 主要类
• XWPFDocument 整个文档对象
• XWPFParagraph 段落 //以回车来衡量，并非文本意义上的段落
• XWPFRun 一个片段(字体样式相同的一段)
• XWPFPicture 图片
• XWPFTable 表格

读文档

```
package docx;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.HashMap;
import java.util.List;

import javax.xml.namespace.QName;

import org.apache.poi.POIXMLDocumentPart;
import org.apache.poi.poifs.filesystem.POIFSFileSystem;
import org.apache.poi.util.IOUtils;
import org.apache.poi.xwpf.usermodel.BodyElementType;
import org.apache.poi.xwpf.usermodel.Document;
import org.apache.poi.xwpf.usermodel.IBodyElement;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;
import org.apache.poi.xwpf.usermodel.XWPFPicture;
import org.apache.poi.xwpf.usermodel.XWPFRun;
import org.apache.xmlbeans.XmlCursor;
import org.apache.xmlbeans.XmlObject;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.CTObject;

public class TextRead {

    public static void main(String[] args) throws Exception {
        readDocx();
    }

    public static void readDocx() throws Exception {
        InputStream is;
        is = new FileInputStream("test.docx");
        XWPFDocument xwpf = new XWPFDocument(is);

        //获取docx中的所有子元素
        List<IBodyElement> ibs= xwpf.getBodyElements();
        for(IBodyElement ib:ibs)
        {
            BodyElementType bet = ib.getElementType();
            if(bet== BodyElementType.TABLE)
            {
                //表格
                System.out.println("table" + ib.getPart());
            }
            else
            {
                //段落
                XWPFParagraph para = (XWPFParagraph) ib;
                System.out.println("It is a new paragraph....The indention is "
                         + para.getFirstLineIndent() + "," +
para.getIndentationFirstLine() );
                //System.out.println(para.getCTP().xmlText());
```

```
                List<XWPFRun> res = para.getRuns();
                //System.out.println("run");
                if(res.size()<=0)
                {
                    System.out.println("empty line");
                }
                for(XWPFRun re: res)
                {
                    if(null == re.text()||re.text().length()<=0)
                    {
                        if(re.getEmbeddedPictures().size()>0)
                        {
                            System.out.println("image***" +
re.getEmbeddedPictures().size());
                        } else
                        {
                            System.out.println("objects:" +
re.getCTR().getObjectList().size());
                            if(re.getCTR().xmlText().indexOf("instrText") > 0) {
                                System.out.println("there is an equation
field");
                            }
                            else
                            {
                                //System.out.println(re.getCTR().xmlText());
                            }
                        }
                    }
                    else
                    {
                        System.out.println("==="+ re.getCharacterSpacing() +
re.text());
                    }
                }
            }
        }
        is.close();
    }
}
```

读表格

```
package docx;

/**
 * 本类完成docx的表格内容读取
 */
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.Iterator;
import java.util.List;

import org.apache.poi.poifs.filesystem.POIFSFileSystem;
import org.apache.poi.xwpf.usermodel.BodyElementType;
import org.apache.poi.xwpf.usermodel.IBodyElement;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
```

```java
import org.apache.poi.xwpf.usermodel.XWPFParagraph;
import org.apache.poi.xwpf.usermodel.XWPFRun;
import org.apache.poi.xwpf.usermodel.XWPFTable;
import org.apache.poi.xwpf.usermodel.XWPFTableCell;
import org.apache.poi.xwpf.usermodel.XWPFTableRow;


public class TableRead {
    public static void main(String[] args) throws Exception {
        testTable();
    }

    public static void testTable() throws Exception {
        InputStream is = new FileInputStream("simple2.docx");
        XWPFDocument xwpf = new XWPFDocument(is);
        List<XWPFParagraph> paras = xwpf.getParagraphs();
        //List<POIXMLDocumentPart> pdps = xwpf.getRelations();

        List<IBodyElement> ibs= xwpf.getBodyElements();
        for(IBodyElement ib:ibs)
        {
            BodyElementType bet = ib.getElementType();
            if(bet== BodyElementType.TABLE)
            {
                //表格
                System.out.println("table" + ib.getPart());
                XWPFTable table = (XWPFTable) ib;
                List<XWPFTableRow> rows=table.getRows();
                 //读取每一行数据
                for (int i = 0; i < rows.size(); i++) {
                    XWPFTableRow  row = rows.get(i);
                    //读取每一列数据
                    List<XWPFTableCell> cells = row.getTableCells();
                    for (int j = 0; j < cells.size(); j++) {
                        XWPFTableCell cell=cells.get(j);
                        System.out.println(cell.getText());
                        List<XWPFParagraph> cps = cell.getParagraphs();
                        System.out.println(cps.size());
                    }
                }
            }
            else
            {
                //段落
                XWPFParagraph para = (XWPFParagraph) ib;
                System.out.println("It is a new paragraph....The indention is "
                   + para.getFirstLineIndent() + "," +
para.getIndentationFirstLine() + ","
                   + para.getIndentationHanging()+"," +
para.getIndentationLeft() + ","
                   + para.getIndentationRight() + "," + para.getIndentFromLeft()
+ ","
                   + para.getIndentFromRight()+"," +
para.getAlignment().getValue());

                //System.out.println(para.getAlignment());
                //System.out.println(para.getRuns().size());
```

```
                List<XWPFRun> res = para.getRuns();
                System.out.println("run");
                if(res.size()<=0)
                {
                    System.out.println("empty line");
                }
                for(XWPFRun re: res)
                {
                    if(null == re.text()||re.text().length()<=0)
                    {
                        if(re.getEmbeddedPictures().size()>0)
                        {
                            System.out.println("image***" +
re.getEmbeddedPictures().size());

                        }
                        else
                        {
                            System.out.println("objects:" +
re.getCTR().getObjectList().size());
                            System.out.println(re.getCTR().xmlText());

                        }
                    }
                    else
                    {
                        System.out.println("===" + re.text());
                    }
                }

            }
        }
        is.close();
    }
}
```

写表格

```
package docx;

/*
 * 本类测试写入表格
 */

import java.io.FileOutputStream;
import java.io.OutputStream;
import java.math.BigInteger;
import java.util.List;

import org.apache.poi.xwpf.usermodel.*;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.*;

public class TableWrite {
```

```java
 public static void main(String[] args) throws Exception {
    try {
        createSimpleTable();
    }
    catch(Exception e) {
        System.out.println("Error trying to create simple table.");
        throw(e);
    }
 }

 public static void createSimpleTable() throws Exception {
     XWPFDocument doc = new XWPFDocument();

     try {
         XWPFTable table = doc.createTable(3, 3);

         //第二行的第二列
         table.getRow(1).getCell(1).setText("表格示例");

         XWPFParagraph p1 = table.getRow(0).getCell(0).getParagraphs().get(0);

         XWPFRun r1 = p1.createRun();
         r1.setBold(true);
         r1.setText("The quick brown fox");
         r1.setItalic(true);
         r1.setFontFamily("Courier");
         r1.setUnderline(UnderlinePatterns.DOT_DOT_DASH);
         r1.setTextPosition(100);

         table.getRow(2).getCell(2).setText("only text");

         OutputStream out = new FileOutputStream("simpleTable.docx");
         try {
             doc.write(out);
         } finally {
             out.close();
         }
     } finally {
         doc.close();
     }
 }
}
```

读图

```java
package docx;

/**
 * 本类 完成docx的图片读取工作
 */
import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.openxml4j.opc.OPCPackage;
import org.apache.poi.xwpf.usermodel.*;

import java.awt.image.BufferedImage;
```

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Iterator;

import javax.imageio.ImageIO;

public class ImageRead {


    public static void imageRead() throws IOException, InvalidFormatException {
        File docFile = new File("simple.docx");

        XWPFDocument doc = new XWPFDocument(OPCPackage.openOrCreate(docFile));

        int i = 0;
        for (XWPFParagraph p : doc.getParagraphs()) {
            for (XWPFRun run : p.getRuns()) {
                System.out.println("a new run");
                for (XWPFPicture pic : run.getEmbeddedPictures()) {
                    System.out.println(pic.getCTPicture().xmlText());
                    //image EMU(English Metric Unit)

System.out.println(pic.getCTPicture().getSpPr().getXfrm().getExt().getCx());

System.out.println(pic.getCTPicture().getSpPr().getXfrm().getExt().getCy());
                    //image 显示大小  以厘米为单位

System.out.println(pic.getCTPicture().getSpPr().getXfrm().getExt().getCx()/36000
0.0);

System.out.println(pic.getCTPicture().getSpPr().getXfrm().getExt().getCy()/36000
0.0);
                    int type = pic.getPictureData().getPictureType();
                    byte [] img = pic.getPictureData().getData();
                    BufferedImage bufferedImage=
ImageRead.byteArrayToImage(img);
                    System.out.println(bufferedImage.getWidth());
                    System.out.println(bufferedImage.getHeight());
                    String extension = "";
                    switch(type)
                    {
                        case Document.PICTURE_TYPE_EMF: extension = ".emf";
                                                        break;
                        case Document.PICTURE_TYPE_WMF: extension = ".wmf";
                            break;
                        case Document.PICTURE_TYPE_PICT: extension = ".pic";
                            break;
                        case Document.PICTURE_TYPE_PNG: extension = ".png";
                            break;
                        case Document.PICTURE_TYPE_DIB: extension = ".dib";
                            break;
                            default: extension = ".jpg";
                    }
                    //outputFile = new File ( );
```

```
                    //BufferedImage image = ImageIO.read(new File(img));
                    //ImageIO.write(image , "png", outputfile);
                    FileOutputStream fos = new FileOutputStream("test" + i +
extension);
                    fos.write(img);
                    fos.close();
                    i++;
                }
            }
        }
    }
    public static BufferedImage  byteArrayToImage(byte[] bytes){
        BufferedImage bufferedImage=null;
        try {
            InputStream inputStream = new ByteArrayInputStream(bytes);
            bufferedImage = ImageIO.read(inputStream);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
        return bufferedImage;
}
    public static void main(String[] args) throws Exception {
        imageRead();
    }
}
```

写图

```
package docx;

/**
 * 本类完成docx的图片保存工作
 */
import org.apache.poi.util.Units;
import org.apache.poi.xwpf.usermodel.*;

import java.io.FileInputStream;
import java.io.FileOutputStream;


public class ImageWrite {

    public static void main(String[] args) throws Exception {
        XWPFDocument doc = new XWPFDocument();
        XWPFParagraph p = doc.createParagraph();

        XWPFRun r = p.createRun();

        String[] imgFiles = new String[2];

        imgFiles[0] = "c:/temp/ecnu.jpg";
        imgFiles[1] = "c:/temp/shida.jpg";

        for(String imgFile : imgFiles) {
            int format;
```

```
            if(imgFile.endsWith(".emf")) format = XWPFDocument.PICTURE_TYPE_EMF;
            else if(imgFile.endsWith(".wmf")) format =
XWPFDocument.PICTURE_TYPE_WMF;
            else if(imgFile.endsWith(".pict")) format =
XWPFDocument.PICTURE_TYPE_PICT;
            else if(imgFile.endsWith(".jpeg") || imgFile.endsWith(".jpg"))
format = XWPFDocument.PICTURE_TYPE_JPEG;
            else if(imgFile.endsWith(".png")) format =
XWPFDocument.PICTURE_TYPE_PNG;
            else if(imgFile.endsWith(".dib")) format =
XWPFDocument.PICTURE_TYPE_DIB;
            else if(imgFile.endsWith(".gif")) format =
XWPFDocument.PICTURE_TYPE_GIF;
            else if(imgFile.endsWith(".tiff")) format =
XWPFDocument.PICTURE_TYPE_TIFF;
            else if(imgFile.endsWith(".eps")) format =
XWPFDocument.PICTURE_TYPE_EPS;
            else if(imgFile.endsWith(".bmp")) format =
XWPFDocument.PICTURE_TYPE_BMP;
            else if(imgFile.endsWith(".wpg")) format =
XWPFDocument.PICTURE_TYPE_WPG;
            else {
                System.err.println("Unsupported picture: " + imgFile +
                        ". Expected
emf|wmf|pict|jpeg|png|dib|gif|tiff|eps|bmp|wpg");
                continue;
            }

            r.setText(imgFile);
            r.addBreak();
            r.addPicture(new FileInputStream(imgFile), format, imgFile,
Units.toEMU(200), Units.toEMU(200)); // 200x200 pixels
            r.addBreak(BreakType.PAGE);
        }

        FileOutputStream out = new FileOutputStream("images.docx");
        doc.write(out);
        out.close();
    }


}
```

pdf

PDF 处理和第三方包

• 常见功能处理
– 解析PDF
– 生成PDF(转化)
• 第三方包
– Apache PDFBox (免费)
– iText (收费)
– XDocReport (将docx转化为pdf)

Apache PDFBox

– 纯Java类库
– 主要功能：创建，提取文本，分割/合并/删除，...
– 主要类
• PDDocument pdf文档对象
• PDFTextStripper pdf文本对象
• PDFMergerUtility 合并工具

常见功能

抽取PDF文档 / 写PDF / 合并PDF / 删除某页PDF

抽取PDF文档

```
package pdfbox;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import org.apache.pdfbox.io.RandomAccessBuffer;
import org.apache.pdfbox.pdfparser.PDFParser;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.encryption.AccessPermission;
import org.apache.pdfbox.text.PDFTextStripper;

/**
 * PdfReader 抽取pdf的文本
 * @author Tom
 *
 */
public class PdfReader {

    public static void main(String[] args){

        File pdfFile = new File("simple.pdf");
        PDDocument document = null;
        try
        {
            document=PDDocument.load(pdfFile);

            AccessPermission ap = document.getCurrentAccessPermission();
```

```
            if (!ap.canExtractContent())
            {
                throw new IOException("你没有权限抽取文本");
            }
            // 获取页码
            int pages = document.getNumberOfPages();

            // 读文本内容
            PDFTextStripper stripper=new PDFTextStripper();
            // 设置按顺序输出
            stripper.setSortByPosition(true);
            stripper.setStartPage(1);  //起始页
            stripper.setEndPage(pages);//结束页
            String content = stripper.getText(document);
            System.out.println(content);
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

写PDF

```
package pdfbox;

import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;
import org.apache.pdfbox.pdmodel.PDPageContentStream;
import org.apache.pdfbox.pdmodel.font.PDFont;
import org.apache.pdfbox.pdmodel.font.PDType1Font;

public class PdfWriter {

    public static void main(String[] args) {
        createHelloPDF();
    }
    public static void createHelloPDF() {
        PDDocument doc = null;
        PDPage page = null;

        try {
            doc = new PDDocument();
            page = new PDPage();
            doc.addPage(page);
            PDFont font = PDType1Font.HELVETICA_BOLD;
            PDPageContentStream content = new PDPageContentStream(doc, page);
            content.beginText();
            content.setFont(font, 12);
            content.moveTextPositionByAmount(100, 700);
            content.showText("hello world");


            content.endText();
```

```
            content.close();
            doc.save("test.pdf");
            doc.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

合并pdf

```
package pdfbox;

import java.io.File;
import java.io.IOException;

import org.apache.pdfbox.multipdf.PDFMergerUtility;
import org.apache.pdfbox.pdmodel.PDDocument;

public class MergePdfs {
       public static void main(String[] args) throws IOException {

          //Loading an existing PDF document
          File file1 = new File("sample1.pdf");
          PDDocument doc1 = PDDocument.load(file1);

          File file2 = new File("sample2.pdf");
          PDDocument doc2 = PDDocument.load(file2);

          //Instantiating PDFMergerUtility class
          PDFMergerUtility PDFmerger = new PDFMergerUtility();

          //Setting the destination file
          PDFmerger.setDestinationFileName("merge2.pdf");

          //adding the source files
          PDFmerger.addSource(file1);
          PDFmerger.addSource(file2);

          //Merging the two documents
          PDFmerger.mergeDocuments();

          System.out.println("Documents merged");
          //Closing the documents
          doc1.close();
          doc2.close();
       }
    }
```

删除某页pdf

```
package pdfbox;

import java.io.File;

import org.apache.pdfbox.pdmodel.PDDocument;


public class RemovePdf {

    public static void main(String[] args) throws Exception {
        File file = new File("merge.pdf");
        PDDocument document = PDDocument.load(file);

        int noOfPages = document.getNumberOfPages();
        System.out.println("total pages: " + noOfPages);

        // 删除第1页
        document.removePage(1);  // 页码索引从0开始算

        System.out.println("page removed");

        // 另存为新文档
        document.save("merge2.pdf");

        document.close();
    }
}
```

XDocReport

• XDocReport
– 将docx文档合并输出为其他数据格式(pdf/html/...)
– PdfConverter
– 基于poi和iText完成

word转PDF

```
import java.awt.Color;
import java.io.FileInputStream;
import java.io.FileOutputStream;

import org.apache.poi.xwpf.usermodel.XWPFDocument;

import com.lowagie.text.Font;
import com.lowagie.text.pdf.BaseFont;

import org.apache.poi.xwpf.converter.pdf.PdfConverter;
import org.apache.poi.xwpf.converter.pdf.PdfOptions;
```

```java
import fr.opensagres.xdocreport.itext.extension.font.IFontProvider;
import fr.opensagres.xdocreport.itext.extension.font.ITextFontRegistry;

public class XDocReportTest {

    public static void main(String[] args) throws Exception {
        XWPFDocument doc = new XWPFDocument(new
FileInputStream("template.docx"));// docx
        PdfOptions options = PdfOptions.create();
        options.fontProvider(new IFontProvider() {
            // 设置中文字体
            public Font getFont(String familyName, String encoding, float size,
int style, Color color) {
                try {
                    BaseFont bfChinese = BaseFont.createFont(
                            "C:\\Program Files (x86)\\Microsoft
Office\\root\\VFS\\Fonts\\private\\STSONG.TTF",
                            BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
                    Font fontChinese = new Font(bfChinese, size, style, color);
                    if (familyName != null)
                        fontChinese.setFamily(familyName);
                    return fontChinese;
                } catch (Throwable e) {
                    e.printStackTrace();
                    return ITextFontRegistry.getRegistry().getFont(familyName,
encoding, size, style, color);
                }
            }
        });
        PdfConverter.getInstance().convert(doc, new
FileOutputStream("template.pdf"), options);// pdf
    }
}
```

Excel

表格文件

- xls/xlsx 文件 (Microsoft Excel)
- CSV文件 (Comma-Seperated Values文件)

xlsx(Excel)

- 与word类似，也分成xls和xlsx。
- xlsx以XML为标准，为主要研究对象
- 数据
  - sheet
- 行

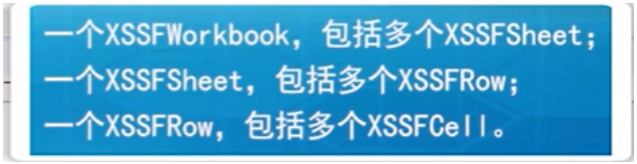
• 列
–单元格

xlsx(Excel)功能和第三方包

• 常见功能
– 解析
– 生成
• 第三方的包
– POI，JXL (免费)
– COM4J (Windows平台)
– Aspose等(收费)

Apache POI

– Apache 出品，必属精品， poi.apache.org
– 可处理docx, xlsx, pptx, visio等office套件
– 纯Java工具包，无需第三方依赖
– 主要类
• XSSFWorkbook 整个文档对象
• XSSFSheet 单个sheet对象
• XSSFRow 一行对象
• XSSFCell 一个单元格对象



依赖

```
<dependencies>
  <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi</artifactId>
      <version>3.14</version>
  </dependency>
  <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>ooxml-schemas</artifactId>
      <version>1.3</version>
  </dependency>
  <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi-ooxml-schemas</artifactId>
      <version>3.14</version>
  </dependency>
  <dependency>
      <groupId>org.apache.xmlbeans</groupId>
      <artifactId>xmlbeans</artifactId>
      <version>2.6.0</version>
```

```
    </dependency>
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi-examples</artifactId>
        <version>3.14</version>
    </dependency>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-csv</artifactId>
        <version>1.6</version>
    </dependency>
  </dependencies>
```

import

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Iterator;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
```

读xsl文件

```
public static void readXLSXFile() throws IOException
  {
    InputStream ExcelFileToRead = new FileInputStream("Test.xlsx");
    XSSFWorkbook wb = new XSSFWorkbook(ExcelFileToRead);

    XSSFSheet sheet = wb.getSheetAt(0);
    XSSFRow row;
    XSSFCell cell;

    Iterator rows = sheet.rowIterator();

    while (rows.hasNext())
    {
      row = (XSSFRow) rows.next();
      Iterator cells = row.cellIterator();
      while (cells.hasNext())
      {
        cell = (XSSFCell) cells.next();

        if (cell.getCellType() == XSSFCell.CELL_TYPE_STRING)
        {
          System.out.print(cell.getStringCellValue() + " ");
```

```
        }
        else if (cell.getCellType() == XSSFCell.CELL_TYPE_NUMERIC)
        {
          System.out.print(cell.getNumericCellValue() + " ");
        }
        else
        {
          // U Can Handel Boolean, Formula, Errors
        }
      }
      System.out.println();
    }

  }
```

写xsl文件

```
  public static void writeXLSXFile() throws IOException
  {

    String excelFileName = "Test.xlsx";// name of excel file

    String sheetName = "Sheet1";// name of sheet

    XSSFWorkbook wb = new XSSFWorkbook();
    XSSFSheet sheet = wb.createSheet(sheetName);

    // iterating r number of rows
    for (int r = 0; r < 5; r++)
    {
      XSSFRow row = sheet.createRow(r);

      // iterating c number of columns
      for (int c = 0; c < 5; c++)
      {
        XSSFCell cell = row.createCell(c);

        cell.setCellValue("Cell " + r + " " + c);
      }
    }
    FileOutputStream fileOut = new FileOutputStream(excelFileName);

    // write this workbook to an Outputstream.
    wb.write(fileOut);
    fileOut.flush();
    fileOut.close();

  }
```

CSV文件

• 全称： Comma-Seperated Values文件(逗号分隔)
• 广义CSV文件，可以由空格/Tab键/分号/.../完成字段分隔
• 第三方包：Apache Commons CSV
– CSVFormat 文档格式
– CSVParser 解析文档
– CSVRecord 一行记录
– CSVPrinter 写入文档

读写CSV文件

```
package csv;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.time.LocalDate;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVPrinter;
import org.apache.commons.csv.CSVRecord;

public class CSVTest {

    public static void main(String[] args) throws Exception {
        readCSVWithIndex();
        System.out.println("===========华丽丽的分割线1===================");
        readCSVWithName();
        System.out.println("===========华丽丽的分割线2===================");
        writeCSV();
        System.out.println("write done");
    }

    //通过索引值读取
    public static void readCSVWithIndex() throws Exception {
        Reader in = new FileReader("c:/temp/score.csv");
        Iterable<CSVRecord> records = CSVFormat.EXCEL.parse(in);
        for (CSVRecord record : records) {
            System.out.println(record.get(0)); //0 代表第一列
        }
    }

    //通过列的名字读取
    public static void readCSVWithName() throws Exception {
        Reader in = new FileReader("c:/temp/score.csv");
        Iterable<CSVRecord> records = CSVFormat.RFC4180.withHeader("Name",
"Subject", "Score").parse(in);
        for (CSVRecord record : records) {
            System.out.println(record.get("Subject"));
        }
    }

    //写CSV
```

```
    public static void writeCSV() throws Exception {
        try (CSVPrinter printer = new CSVPrinter(new FileWriter("person.csv"),
CSVFormat.EXCEL)) {
            printer.printRecord("id", "userName", "firstName", "lastName",
"birthday");
            printer.printRecord(1, "john73", "John", "Doe", LocalDate.of(1973,
9, 15));
            printer.println();  //空白行
            printer.printRecord(2, "mary", "Mary", "Meyer", LocalDate.of(1985,
3, 29));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```