LEARNING apache-poi

#apache-poi

Table of Contents

About

Chapter 1: Getting started with apache-poi

Remarks

The Apache POI Project is a Java APIs for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). In short, you can read and write MS Excel, Word, and Powerpoint files using Java.

Apache POI became a top level project in June 2007 and POI 3.0 artifacts were re-released. Prior to that date POI was a sub-project of Apache Jakarta.

Versions

Version Name	Release Date
3.11	2014-12-17
3.12	2015-05-09
3.13	2015-09-22
3.14	2016-03-02
3.15	2016-09-19
3.16	2017-04-19

Examples

Installation or Setup

Detailed instructions on getting apache-poi set up or installed.

Read Getting started with apache-poi online: https://riptutorial.com/apache-poi/topic/5516/getting-started-with-apache-poi

Chapter 2: Getting started with NPOI

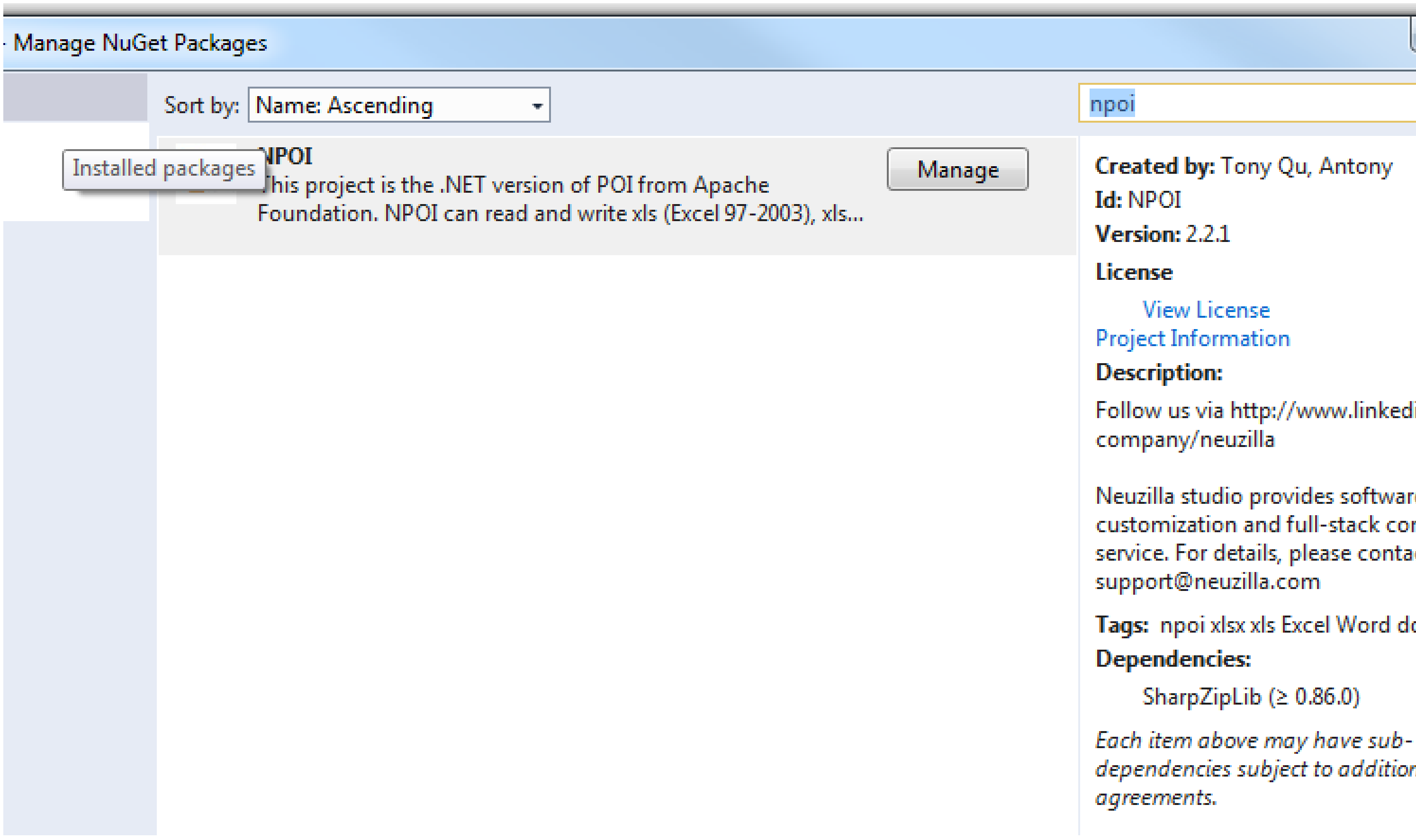
Introduction

It is .NET version of POI Java project. it allows to read/write xls, doc, ppt files without Microsoft Office installed. Details about documentation is available here:https://github.com/tonyqus/npoi

Examples

Installing NPOI

Best way to include all library related to NPOI is NUGet Package Manager. Search for NPOI on NUGet package manager window.

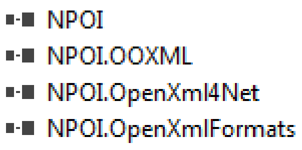


NPOI
This project is the .NET version of POI from Apache Foundation. NPOI can read and write xls (Excel 97-2003), xls...

Created by: Tony Qu, Antony
Id: NPOI
Version: 2.2.1
License
View License
Project Information
Description:
Follow us via http://www.linked
company/neuzilla

Neuzilla studio provides softwar
customization and full-stack co
service. For details, please conta
support@neuzilla.com
Tags: npoi xlsx xls Excel Word d
Dependencies:
SharpZipLib (≥ 0.86.0)
*Each item above may have sub-
dependencies subject to additio
agreements.*

Once it is successfully installed all needed library will appear in reference section of your current project



- NPOI
- NPOI.OOXML
- NPOI.OpenXml4Net
- NPOI.OpenXmlFormats

Then include the NPOI into your file like this

```
using NPOI.SS.UserModel;
using NPOI.SS.Util;
using NPOI.XSSF.UserModel
```

Create a Excel file

```
MemoryStream excelMS =  GetExcelFile();

 //Using Resposne Stream to Make File Available for User to Download;
 Response.Clear();
 Response.ContentType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";
 Response.AddHeader("Content-Disposition", string.Format("attachment;filename={0}", @"Excel_"
+ DateTime.Now.ToString("yyyy-dd-M-HH-mm") + ".xlsx"));

 Response.BinaryWrite(excelMS.ToArray());
 Response.End();
```

> be careful while using MIME type. you'll have select different MIME types for different file formats. For EX xltm extension it will be "application/vnd.ms-excel.template.macroEnabled.12"; xlxs extension it will be "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"; Different MIME types supported MS is at this link.

```
public MemoryStream GetExcelFile()
{
    //Excel File Stream
    MemoryStream ms = null;
    // create workbook
    XSSFWorkbook workbook = new XSSFWorkbook();
    // the table named mySheet
    //Assigning New Sheet Name /
    XSSFSheet sheet = (XSSFSheet)workbook.CreateSheet("WorkSheet");

    // Assuming FreezePaneRow  = 10, FreezePaneColumn = 11 in the config file
    int freezeRow = Convert.ToInt32(ConfigurationManager.AppSettings["FreezePaneRow"]);
    int freezeCol = Convert.ToInt32(ConfigurationManager.AppSettings["FreezePaneCol"]);
    try
    {
        // Freeze Created Excel Sheet Row;
            sheet.CreateFreezePane(freezeCol, freezeRow, freezeCol, freezeRow);
        //Freezing only the Header Row;
            sheet.CreateFreezePane(0, 1);

         using (ms = new MemoryStream())
            {
                logger.Info("Using Memory Stream to Create New WorkBook");
                workbook.Write(ms); // Write to memory stream for download through browser

            }
    }
    catch (Exception Ex)
    { ... }
    return ms;
}
```

Reading a Excel File

There are various approach to read a excel file. I'll provide a example with file path parameter. You can get various way to read a file at this post.

```
public void ReadExcel(string path)
        {
            // Write data in workbook from xls document.
            XSSFWorkbook workbook = new XSSFWorkbook(path);
            // Read the current table data
            XSSFSheet sheet = (XSSFSheet)workbook.GetSheetAt(0);
            // Read the current row data
            XSSFRow headerRow = (XSSFRow)sheet.GetRow(0);
            // LastCellNum is the number of cells of current rows
            int cellCount = headerRow.LastCellNum;
            DataTable dt = new DataTable();
            bool isBlanKRow = false;

            try
            {
                if (dt.Rows.Count == 0)
                {
                    //Reading First Row as Header for Excel Sheet;
                    try
                    {
                        for (int j = headerRow.FirstCellNum; j < cellCount; j++)
                        {
                            // get data as the column header of DataTable
                            DataColumn column = new
DataColumn(headerRow.GetCell(j).StringCellValue);
                            dt.Columns.Add(column);
                        }
                    }
                    catch (Exception Ex)
                    { }
                }

                for (int sheetindex = 0; sheetindex < workbook.NumberOfSheets; sheetindex++)
                {
                    sheet = (XSSFSheet)workbook.GetSheetAt(sheetindex);
                    if (null != sheet)
                    {
                        // LastRowNum is the number of rows of current table
                        int rowCount = sheet.LastRowNum + 1;
                        //Reading Rows and Copying it to Data Table;
                        try
                        {
                            for (int i = (sheet.FirstRowNum + 1); i < rowCount; i++)
                            {
                                XSSFRow row = (XSSFRow)sheet.GetRow(i);
                                DataRow dataRow = dt.NewRow();
                                isBlanKRow = true;
                                try
                                {
                                    for (int j = row.FirstCellNum; j < cellCount; j++)
                                    {
                                        if (null != row.GetCell(j) &&
!string.IsNullOrEmpty(row.GetCell(j).ToString()) &&
!string.IsNullOrWhiteSpace(row.GetCell(j).ToString()))
                                        {
                                            dataRow[j] = row.GetCell(j).ToString();
                                            isBlanKRow = false;
                                        }

                                    }
                                }
                                catch (Exception Ex)
                                { }
                                if (!isBlanKRow)
                                {
                                    dt.Rows.Add(dataRow);
                                }
                            }
                        }
                        catch (Exception Ex)
                        { }
                    }
                }
            }
            catch (Exception Ex)
            { }
            finally
            {
                workbook.UnlockStructure();
                workbook.UnlockRevision();
                workbook.UnlockWindows();
                workbook = null;
                sheet = null;
            }
        }
```

Read Getting started with NPOI online: https://riptutorial.com/apache-poi/topic/10761/getting-started-with-npoi

Chapter 3: NPOI: Data validation approach for XSSF(.xslx) excel file using c#

Introduction

Data validation allows user to create a drop-down list and restrict values in the cell to these entries. Due to limitation Excel can't bind more than 256 characters programmatically. To bind more than 256 characters one can follow explained approach.

Examples

When Sum of all list item's total character count less than 256

You can read all items either from any config file or type it inline.

Considering if its saved in Config File

```
// Read all list items from config file
string[] countryDV = ConfigurationManager.AppSettings["countryDV"].Split(',').Select(s =>
s.Trim().ToUpper()).ToArray();
int DVRowLimit = (Int16.MaxValue);
CellRangeAddressList countryDVAddList = new CellRangeAddressList(1, DVRowLimit, 0, 0);
dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateExplicitListConstraint(countryDV);
// In case of Inline list values
// use this approach:  dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateExplicitListConstraint(new string[] {
"USA", "CANADA"});
dataValidation = (XSSFDataValidation)validationHelper.CreateValidation(dvConstraint,
countryDVAddList);
dataValidation.ShowErrorBox = true;
dataValidation.SuppressDropDownArrow = true;
dataValidation.ErrorStyle = 0;
dataValidation.CreateErrorBox("InvalidValue", "Select Valid country.");
dataValidation.ShowErrorBox = true;
dataValidation.CreatePromptBox("country Data Validation", "Enter country.");
dataValidation.ShowPromptBox = true;
sheet.AddValidationData(dataValidation);
```

When Sum of all list item's total character count more than 256.

Approach in this case will be different than previous example because Excel file supports the Data validation for list of items with total character count less than 256 if all items are binded directly as in previous example. But in many situation list items can be longer than 256 characters and in that case direct binding will not work.

However, excel file supports more than 256 character of list item if it is referred from different column of same excel file. So as a workaround once can read all values from database or settings file and keep it hidden from current view into one of the distant column and data validation can read this hidden column through formula to create list item. Below code will show this approach by reading data values through setting file.

```
// Read all list items from config file
string[] countryDV = ConfigurationManager.AppSettings["countryDV"].Split(',').Select(s =>
s.Trim().ToUpper()).ToArray();
// Get the column name where you want to hide the list items, assume distant  column is "ZZ"
string countryDVDataCellColumn =
ConfigurationManager.AppSettings["countryDVDataCellColumn"].Trim().ToString();


int DVRowLimit = (Int16.MaxValue);
// Creating and Assigning Settings for Data Validation
CreateDropDownUsingCellReference(workbook, countryDV, "CountryConstraint",
countryDVDataCellColumn);
CellRangeAddressList countryDVAddList = new CellRangeAddressList(1, DVRowLimit,
targetFirstCol, targetLastCol);
dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateFormulaListConstraint("=CountryConstraint");

dvConstraint.Validate();
dataValidation = (XSSFDataValidation)validationHelper.CreateValidation(dvConstraint,
countryDVAddList);
dataValidation.ShowErrorBox = true;
dataValidation.SuppressDropDownArrow = true;
dataValidation.ErrorStyle = 0;
dataValidation.CreateErrorBox("InvalidValue", "Select Valid country.");
dataValidation.ShowErrorBox = true;
dataValidation.CreatePromptBox("country Data Validation", "Enter country.");
dataValidation.ShowPromptBox = true;
sheet.AddValidationData(dataValidation);



private void CreateDropDownUsingCellReference(XSSFWorkbook wb, string[] csvListOfValues,
string listName, string headerName)
{
    int columnIndex = CellReference.ConvertColStringToIndex(headerName);
    try
    {
    XSSFName namedCell = (XSSFName)wb.CreateName();
    namedCell.NameName = listName;
    //Creating Cell and Assigning Values from CSVListOfValues;
    for (int i = 0; i < csvListOfValues.Length; i++)
    {
            var namedRow = wb.GetSheetAt(0).CreateRow(i + 1);
            namedRow.CreateCell(columnIndex).SetCellValue(csvListOfValues[i]);
    }

    //Assigning the Reference for sheet 0 With Cell Range, where list items iscopied
    String reference = wb.GetSheetAt(0).SheetName + "!$" + headerName + "$2:$" + headerName +
"$" + (csvListOfValues.Length + 1).ToString();
    namedCell.RefersToFormula = reference;

    //Hiding the Column  now;
    wb.GetSheetAt(0).SetColumnHidden(columnIndex, true);
    }
    catch (Exception Ex)
    { }
}
```

Read NPOI: Data validation approach for XSSF(.xslx) excel file using c# online:
https://riptutorial.com/apache-poi/topic/10753/npoi--data-validation-approach-for-xssf--xslx--excel-file-using-csharp

Chapter 4: NPOI: Data validation constraint approach for Date, Time , List Item , email etc. for XSSF(.xslx) excel file using c#

Introduction

Creating the data validation constraint can be tricky and time taking in NPOI. I have shared some of my workable approach. These approach will give good idea to customize your own constraint types.

Examples

Set the Date Constraints for Date Field Values Between 01/01/1900 To 12/31/2119 with Date Format mm/dd//yyyyy;

```
  int DVRowLimit = (Int16.MaxValue);
        CellRangeAddressList cellRangeFieldsType1 = new CellRangeAddressList(1, DVRowLimit,
targetFirstCol, targetLastCol);
        XSSFDataValidationConstraint dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateDateConstraint(OperatorType.BETWEEN,
"=DATE(1900,1,1)", "=DATE(2119,12,31)", "mm/dd/yyyyy");
    //dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateDateConstraint(OperatorType.IGNORED, "",
"", "m/d/yy h:mm");
        XSSFDataValidation dataValidation =
(XSSFDataValidation)validationHelper.CreateValidation(dvConstraint, cellRangeFieldsType1);
        dataValidation.ShowErrorBox = true;
        dataValidation.ErrorStyle = 0;
        dataValidation.CreateErrorBox("InvalidDate", "Allowed Format is MM/DD/YYYY");
        dataValidation.ShowErrorBox = true;
        dataValidation.CreatePromptBox("Date Data Validation", "Enter Date in Format
MM/DD/YYYY.");
        dataValidation.ShowPromptBox = true;
        sheet.AddValidationData(dataValidation);
```

Set the Time Constraints for Field Values Between 00:00 To 23:59 with Date Format HH:MM;

```
  dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateTimeConstraint(OperatorType.BETWEEN,
"=TIME(0,0,0)", "=TIME(23,59,59)");
```

Create a list item Constraint

```
 dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateExplicitListConstraint(new string[] {
```

```
"MON", "TUE" , "WED", "THU", "FRI"});
```

Phone Number Constraint

//try same approach for currency types with format like "$#,###.00"and date "m/d/yy h:mm"

```
XSSFFont defaultFont = (XSSFFont)workbook.CreateFont();
defaultFont.FontHeightInPoints = (short)10;
defaultFont.FontName = "Arial";
XSSFCellStyle phoneCellStyle = (XSSFCellStyle)workbook.CreateCellStyle();
XSSFDataFormat phoneDataFormat = (XSSFDataFormat)workbook.CreateDataFormat();
phoneCellStyle.SetDataFormat(phoneDataFormat.GetFormat("000-000-0000"));
phoneCellStyle.FillBackgroundColor = IndexedColors.LightYellow.Index;
dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateintConstraint(OperatorType.BETWEEN,
"1000000000", "9999999999");
sheet.AddValidationData(dataValidation);
sheet.SetDefaultColumnStyle(headerCount, phoneCellStyle);
```

Email Address constraint For Email Columns

```
 string emailValidationFormula = GetEmailValidationFormula(targetFirstCol);
 CellRangeAddressList cellRangeFieldsType5 = new CellRangeAddressList(1, DVRowLimit,
targetFirstCol, targetLastCol);
     dvConstraint =
(XSSFDataValidationConstraint)validationHelper.CreateCustomConstraint(emailValidationFormula)
```

// get the email address validation pattern private string GetEmailValidationFormula(int targetColumn) {

```
    int div = headerCount + 1;
    string colLetter = String.Empty;
    int mod = 0;

    try
    {
        while (div > 0)
        {
            mod = (div - 1) % 26;
            colLetter = (char)(65 + mod) + colLetter;
            div = (int)((div - mod) / 26);
        }
    }
    catch (Exception Ex)
    {
        logger.Error("Error ", Ex);
    }

    return "=AND(FIND(\"@\"," + colLetter + "2),FIND(\".\"," + colLetter + "2),ISERROR(FIND(\"
\"," + colLetter + "2)))";

}
```

Read NPOI: Data validation constraint approach for Date, Time , List Item , email etc. for XSSF(.xslx) excel file using c# online: https://riptutorial.com/apache-poi/topic/10754/npoi--datavalidation-constraint-approach-for-date--time---list-item---email-etc--for-xssf--xslx--excel-file-using-csharp

Chapter 5: Simple Excel (XLSX) creation

Examples

Basic excel

```
String fileName = "Fruit.xlsx";

String sheetName = "Apples";

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

for (int r=0;r < 3; r++ )
{
    XSSFRow row = sheet.createRow(r);

    //iterating c number of columns
    for (int c=0;c < 3; c++ )
    {
        XSSFCell cell = row.createCell(c);

        cell.setCellValue("Nice apple, in row: "+r+" and col: "+c);
    }
}

try(FileOutputStream fos = new FileOutputStream(fileName))
{
    wb.write(fos);
}
```

Read Simple Excel (XLSX) creation online: https://riptutorial.com/apache-poi/topic/6058/simple-excel--xlsx--creation

Credits

S. No	Chapters	Contributors
1	Getting started with apache-poi	Community, jmarkmurphy
2	Getting started with NPOI	kumar chandraketu
3	NPOI: Data validation approach for XSSF(.xslx) excel file using c#	kumar chandraketu
4	NPOI: Data validation constraint approach for Date, Time , List Item , email etc. for XSSF(.xslx) excel file using c#	kumar chandraketu
5	Simple Excel (XLSX) creation	RobAu