Excel File Processing Based on NPOI Package

Li-hua MA*, Sheng-ming LI, Xiao-lan WANG and Feng PANG
National Astronomical Observatories, Chinese Academy of Sciences, Beijing, China
*Corresponding author

Keywords: Excel file, NPOI package, Visual studio.

Abstract. With the development of modern technology, there are more and more measurement data in various industries. Excel file is one of commonly used computer file formats, and its processing has been received increasing attention. Many data analysis software can directly call and process Excel file. In the C# project of the Visual Studio 2013, integrated development environment, with the newly released NPOI package version 2.3.0, in this present paper, routine operations for the Excel file are implemented and corresponding example codes are provided. This work can be used for reference by related users.

Introduction

Microsoft Excel is an important part of Microsoft Office (referred to as Excel). It is a spreadsheet software written by Microsoft for computer users using Windows and Apple Macintosh operating systems. The Excel provides a user-friendly interface and retains the features of VisiCalc, the first spreadsheet software. It is the first spreadsheet software (including fonts, text attributes, cell formats) that allows users to customize the interface. Excel introduces the “smart recalculation” function, that is, when the cell data changes, only the data related to it will be updated and the original tabulation software can only recalculate all the data or wait for the next instruction. It also has powerful graphics capabilities. In addition, Excel-supported Visual Basic for Applications (VBA) is a powerful tool that enables Excel to form an independent programming environment. Manual steps can be automated using VBA and macros. The Excel can be used for data processing, statistical analysis and calculation, simple database management, charting, check and delete macro viruses, and share resources with the Internet. With the VBA, the Excel can be used to develop application-specific programs. Since the release of version 5.0 as a Microsoft Office component in 1993, Excel has become the dominant software for spreadsheets on the applicable operating platforms. It is widely used in management, statistics, finance, finance and many other fields [1]. Many data analysis software, including Python and R, provide a toolkit for reading and outputting Excel files directly.

Visual Studio Integrated Development Environment

Visual Studio is a series of development kits released by Microsoft Corporation of the United States. It is a development toolset that includes most of the tools needed throughout the software lifecycle. The target code written is for all platforms supported by Microsoft. It is the integrated development environment for the most popular Windows platform applications. In 2002, with the release of the .NET slogan and the release of Windows XP/Office XP, Microsoft released Visual Studio .NET (build number 7.0). In this edition, Microsoft stripped Visual FoxPro as a separate development environment sold separately in Visual FoxPro 7.0, while canceling Visual InterDev. At the same time, Microsoft introduced a managed code mechanism built on the .NET framework (version 1.0) and a new language C#. The C# is a modern language based on C++ and Java. It is a language for writing .NET frameworks. .NET's Common Language Runtime (CLR), which aims to support components developed in different languages in the same project. All code supported by the CLR will be interpreted as machine code executable by the CLR and then run [2]. Since its release, Microsoft has
released one version of Visual Studio for an average of 1-2 years. Visual Studio 2013 released in October 2013, is widely used and this work is exemplified with the version.

**Processing Method of Excel Files with C#**

Using the Visual Studio to generate C# projects, there are following four common methods operating the computer’s local Excel files.

1. Using OleDB to read Excel files. The entire Excel file is treated as a data source for data reading. Specific action: Namespace using System.Data.OleDb. The advantage of this method is simple and fast. Disadvantage is not flexible. When the amount of Excel data is large, the memory usage is large, which may cause memory overflow.

2. Utilization of the com component Microsoft.Office.Interop.Excel.dll. Firstly, getting Excel.dll, copying the Excel file in the Office installation directory to the bin directory of DotNet, cmd to the directory, run TlbImp Excel.exe Excel.dll to get the Dll file. Secondly, adding the reference to the dll file in the project. Its advantage is very flexible, and it can realize various data processing functions of Excel. However, since access to data is based on cell mode, reading and writing data is slower. Especially when the amount of data is large, the access efficiency problem is more prominent.

3. Converting the Excel file into a CSV (comma-separated) file and reading it with the file stream. This method is equivalent to reading the txt text file.

4. Implement Excel file operations through the NPOI package. NPOI uses the .NET version of the Java project POI, an open source project for reading and writing Microsoft OLE2 component documentation. With the method, the data can be read faster and its operation is flexible. Its developers encapsulate many useful operations of Excel. Someone only need to know how to use a function without knowing the specific implementation of this function [3].

**NPOI Package**

The NPOI package is created and maintained by the Neuzilla. It can operate Office objects without installing the Office, which is the .NET version of the POI. The package updates in a timely manner and provides convenience for .NET developers. It is mainly used to generate Excel reports, extract text from Excel file in search engine modules, batch generation of Excel files and generate new Excel files based on some templates. In March 2017, the Neuzilla released a new version 2.3.0. The package consists of four dynamic link libraries (DLL), such as NPOI.DLL, NPOI.OOXML.DLL, NPOI.OpenXml4Net.dll and NPOI.OpenXmlFormats.DLL. The modules and functions of the DLL files are as follows:

1. NPOI.DLL contains modules
   - NPOI. Util: basic auxiliary library.
   - NPOI.POIFS OLE2: format read and write library, mainly responsible for processing Document Information.
   - NPOILDF: Microsoft Drawing: format read and write library.
   - NPOI.SS: an interface library shared by Excel 2003 and Excel 2007 operation libraries and a common module implementation, such as a formula calculation library.
   - NPOI.HPSF: OLE2’s summary information and document summary information attribute read and write libraries.
   - NPOI.HSSF: Excel BIFF format read and write library, which is the most used namespace for users.

2. NPOI.OOXML contains modules
   - NPOI.OOXML: Excel 2007 operation library, most of the objects implement the NPOI.SS interface.
   - NPOI.XWPF: Word 2007 operations library.

3. NPOI.OpenXml4Net.DLL contains modules
Ported from POI's sub-project OpenXml4j, its function is similar to NPOI. POIFS, that it operates in OOXML format. This module is identical to the System. Packaging feature provided by Microsoft. The System. Packaging supports .Net 3.0 and above, it supports .NET 2.0.

(4) NPOI.OpenXmlFormats.DLL contains modules defining the mapping between all OOXML objects and C# objects, and assisting in serialization and deserialization to separate the file read and write layers from the logical layer.

NPOI Package Processing Excel File

NPOI Package Installation

The NuGet is a management tool of extension packages for Visual Studio. When using Visual Studio to develop applications based on .NET Framework, NuGet makes it easier and faster for programmers to add, remove and update references in projects. NuGet can easily be used to create and publish packages, which can be used to quickly install packages. The package should be firstly installed. Someone launches “Visual Studio Extension Manager” from the “TOOLS” | “Extensions and Updates” menu option, clicks on the “Online” tab to view useful the Visual Studio extension tool and the NuGet can be found. Someone can use the management to install third-party packages. The installation process of the NPOI package is described in the following two scenarios.

(1) Two ways to install NPOI package online
(a) Launching “Package Manager Console” from “TOOLS” | “Library Package Manager” menu option and typing in the command line:
   PM> install-package NPOI -Version 2.3.0
(b) Starting “Manage NuGet Packages for Solution” from “TOOLS” | “Library Package Manager” menu option. The NPOI package can be found.
   During installation of the NPOI package, the dependency of the package, SharpZipLib, and the corresponding file compression function library, ICSharpCode.SharpZipLib.DLL, are needed.

(2) Two ways to install NPOI package offline
The premise of this installation is that the local computer has saved the latest version of the NPOI package file (npoi.2.3.0.nupkg). The offline installation method is:
(a) Launching “Package Manager Console” from “TOOLS” | “Library Package Manager” menu option and typing in the command line:
   PM> Install-Package NPOI -Source .nupkg file directory
(b) Launching “Package Manager Settings” from “TOOLS” | “Library Package Manager” menu option. At “Package Sources” place, adding the path to the .nupkg file; then, starting “Manage NuGet Packages for Solution” from “TOOLS” | “Library Package Manager” menu option to find the added NPOI package. Finally the package can be installed.

Of course, if someone has already obtained the above NPOI dynamic linkable library DLL file. He can also use “Project” | “Add Reference” to achieve a reference to the NPOI package.

Excel File Processing Example Based on NPOI Package

The following example illustrates creation and operation of an Excel file in a C# project with the NPOI package.

(1) Adding a reference to the NPOI package in the project
   First complete installation of the NPOI package according to the previous section in a new C# Project.
(2) C# code introducing namespace
   Using System.IO; //class operation
   Using NPOI.HSSF. UserModel; //commonly used reference, someone can operate the .xls file of Excel 2003
   Using NPOI. XSSF. UserModel; //Operating the .xlsx file of Excel 2007

340
Using NPOI. SS. User Model; //Implementation of interface library and common mode shared by .xls and .xlsx

(3) Creating workbooks, worksheets, rows and cells
IWorkbook book1 = new HSSFWorkbook(); //Creating one workbook with .xls format
IWorkbook book2 = new XSSFWorkbook(); //Creating one workbook with .xlsx format
ISheet sheet1 = book1.CreateSheet(); //Creating one worksheet
ISheet sheet2 = book1.CreateSheet("sheet1"); //Creating one worksheet with some name
IRow row = sheet1.CreateRow(0); //Creating one row in the worksheet
ICell cell = row.CreateCell(0); //Creating the first cell of the first row in the worksheet

(4) Reading the data in the Excel file
Using (FileStream fs = File.OpenRead(@"Excel filename (including path)"); //Opening Excel file
HSSFWorkbook wk = new HSSFWorkbook(fs); //Writing the data of the Excel file into the workbook wk
ISheet sheet = wk.GetSheetAt(i); //Geting the ith worksheet
IRow row = sheet.GetRow(0); //Geting the first row of the worksheet
ICell cell = row.GetCell(0); //Geting the value of the first cell of one row of data

(5) Condition screening
for (int j = sheet.LastRowNum; j >= 0; j--)  //looping from the last line of worksheet to the first line
{
    IRow row = sheet.GetRow(j); // Reading the (j+1)th row of data
    ICell cell = row.GetCell(k); // Reading the (k+1)th cell data of (j+1)th row
    if (determining whether the cell data meets the filter criteria)
    {
        sheet.ShiftRows(j + 1, sheet.LastRowNum + 1, -1); //Moving 1 line to delete the current line that doesn’t satisfy the condition
    }
}

Summary
The NPOI package is introduced in this work. Based on the version 2.3.0 of the package in C# project of the Visual Studio 2013 integrated development environment, some routine operations of the Excel file are implemented. The powerful package can be used to perform most of operations related to the Excel file. This article only takes condition screening as an example. It is different from the Excel formula in the normal operation or by VBA programming. Here the C# project calls the NPOI package to implement corresponding processing. This operation can be embedded into more complex post-processing data programs.

Acknowledgements
This research was financially supported by the National Natural Science Foundation of China (11573041, 11473045). Thanks to the NPOI package provided by the Neuzilla. During this work, a large amount of network materials were referred, only several references were listed for the length of the article was limited. The authors were grateful to the providers of relevant materials.

References
[1] Wenjie College, Office 2013 Computer Office Basic Course, Tsinghua University Press, Beijing, 2016.
[2] J. Sharp, Microsoft Visual C# 2013 Step by Step, Microsoft Press, 2013.
[3] Working with Excel files in C# projects - using the NPOI library on https://blog.csdn.net/dcrmg/article/details/52356236