How do I retrieve an extremely large range of cells from Excel if the clipboard functions all time out?

devblogs.microsoft.com/oldnewthing/20220610-00

June 10, 2022

Raymond Chen

We've been looking at what happens when a program is slow to generate delay-rendered clipboard data, and what you can do to give the program more time. But we got distracted by the question and lost sight of the problem.

The original customer problem was extracting a large range of cells, and their solution was to launch Excel, programmatically select a bunch of cells, and then copy them to the clipboard, so that another program could copy them out.

If you look at it, you'll realize that this is using a global solution to a local problem. This automated process is using the shared clipboard as a way to transfer data between processes, which means that it will damage whatever was on the clipboard previously. And the user can't use the clipboard while this automated process is running.

My guess is that the automated process is running on a dedicated computer with no interactive user present, so these problems are deemed inconsequential. But it also means that you want to run two copies of this automation, you need two computers!

I think a better way to solve this problem is to use the Excel object model to load the spreadsheet, select the range, and then copy the values from the range. You can then take those values and put them into your own data structures to do whatever it was that you wanted to do.

```
using Microsoft.Office.Interop;

var excel = new Excel.Application();
excel.Workbooks.Open("awesome.xls", ReadOnly: true);
var sheet = excel.ActiveWorkbook.ActiveSheet;
var cellA1 = sheet.Cells[1, 1];
Console.WriteLine(cellA1.Text);
excel.Quit();
```

The advantage of this approach is that you are no longer subject to the whims of the clipboard. You are talking directly to Excel to get the data you want.

Another option is to use the object model to copy the desired range to a new worksheet, and then save the worksheet as CSV (say), and then parse the CSV file, which will be much easier to parse than a full spreadsheet.

Bonus chatter: I suspect their system came together when somebody wanted to automate a tedious business process and cobbled together something that “worked on my machine”, back when the spreadsheet was relatively small, like, maybe a few thousand rows. This lunch-hour hack became popular and eventually became part of the company’s critical workflow. Now that the spreadsheet has grown from a few thousand rows to hundreds of thousands of rows, the “quick and dirty hack” doesn’t quite hold up, and the customer will have to invest in a solution that has better scalability.

Raymond Chen

Follow