# Simio API Note: ExcelGrid DataProvider

Aug 2019 (D.Houck)

Contents

[Simio API Note: ExcelGrid DataProvider 1](#_Toc19799878)

[Overview 2](#_Toc19799879)

[Using the Data Provider 3](#_Toc19799880)

[Running the Test Sample 7](#_Toc19799881)

[Run-Time Contents and Preparation 7](#_Toc19799882)

[Source Contents 8](#_Toc19799883)

[Appendix – Using the DevExpress Excel Spreadsheet Library 9](#_Toc19799884)

[Objects 9](#_Toc19799885)

[Cell Referencing 10](#_Toc19799886)

[Appendix – Using the EPPlus Excel Spreadsheet 11](#_Toc19799887)

[Components 11](#_Toc19799888)

[Cell Referencing 11](#_Toc19799889)

# Overview

This API Note describes how a Grid DataProvider, and specifically an Excel Grid DataProvider.is constructed. Note: This describes the Classic version of the DataProvider, and the version of that uses the Open Source Excel EPPlus Excel library (the version used by Simio internally may use a different version).

A Simio GridDataProvider that uses files implements the Interfaces:

1. IGridDataProvider
2. IGridDataProviderWithFiles

The IGridDataProvider requires properties of Name, Description, Icon, and UniqueID, plus methods GetDataSettings(), GetDataSummary, and OpenData.

The IGridDataProviderWithFiles requires a single method of GetFileNamesIfAny() that takes a single parameter of “dataSettings”, which is a byte array, and returns a string array.

This provider assumes a data location from an Excel file that is one of the following types:

1. Worksheet (a page within Excel), e.g. “Sheet1”
2. A specified Range (a ‘rectangular’ row/column section with the file
3. An Excel “Named Range”.

You specify which type using the popup form which the Extension displays.

The program defines a class ExcelGridDataRecords, which implements IGridDataRecords and thereby enumerates over a collection of ExcelGridDataRecord.

The constructor of ExcelGridDataRecords does the opening of the Excel file (using the parameter settings) putting the file contents (package) into memory (and into a local memory cache to prevent unnecessary file I/O).

# Using the Data Provider

This section describes an example of using this Data Provider.

Once Simio launches, open the data file which is located under the Data folder.

C:\(test)\ExcelReadWrite

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Navigate to the Data tab, and then to the Content.

Press Create Binding. This will load the DataProvider. The ExcelGridDataSettings FileName property getter is invoked to return the FileName and invoke the dialog form.

When the file is selected, the OpenData method returns the IGridDataRecords.

A close up of text on a white background

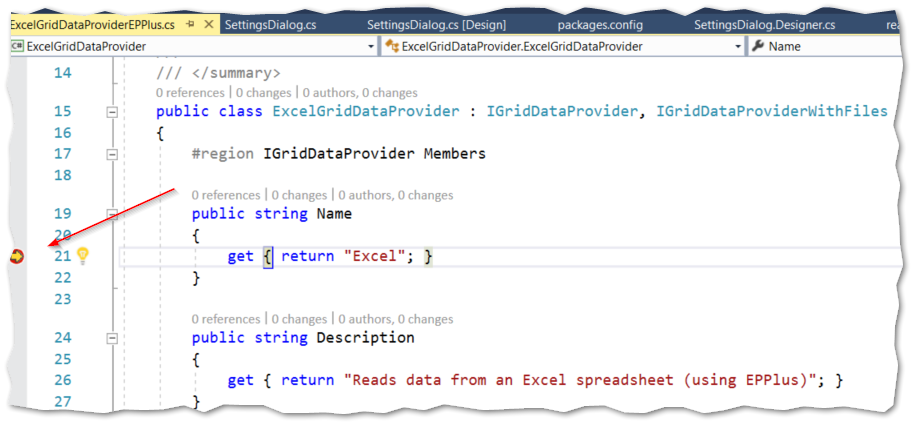
Description automatically generated

Once the file is found, OpenData returns the records using theSettings and openContext:

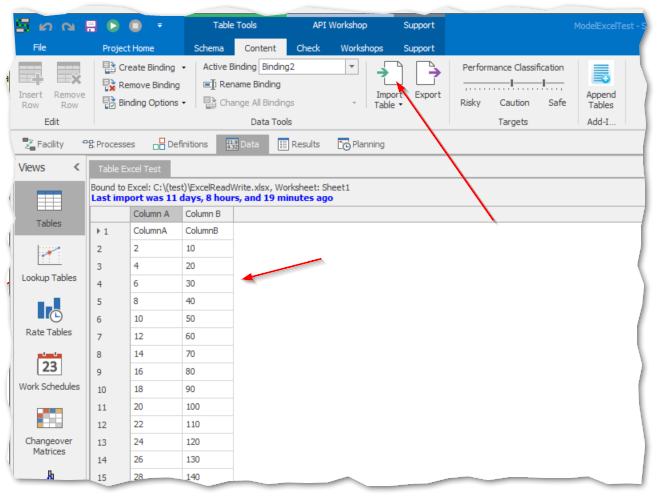
A screenshot of a social media post

Description automatically generated

GetDataSummary is called…



Pressing continue results in:



Then press Import Table

This put the code in the constructor for ExcelGridDataRecords. “Sheet1” is read in and its dimension is A1:B31. And lastRowIndex is 31 and lastColumnIndex is 2.

Figure 1 - Emulator Controller for Adjusting Speed

This Note describes some complex programming topics. It assumes that the reader is familiar with C# and .NET technologies such as threading, asynchronous operations, and the Singleton pattern of programming.

# Running the Test Sample

This sample tests several of the use cases for the Excel DataProvider, including

* Worksheet
* Named

This sample is run by placing the files in the correct location and then running the included Simio test model.

The test cases are:

* Data from Worksheet Range
* Data from Named Range
* Data from Specific Range

And the data columns are types supported by the provider, which are:

A screenshot of a cell phone

Description automatically generated

## Run-Time Contents and Preparation

## Source Contents

# Appendix – Using the DevExpress Excel Spreadsheet Library

If you have the (paid) license for it, the Spreadsheet library provided by DevExpress can also be used as an Excel interface. Here are its main characteristics.

See Also: <https://documentation.devexpress.com/OfficeFileAPI/DevExpress.Spreadsheet.Worksheet.Cells.property>

## Objects

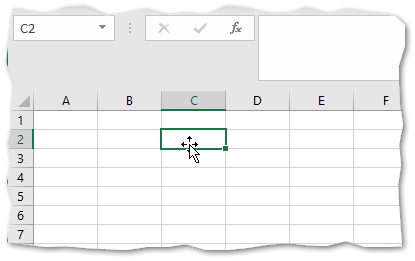
The objects are:

Workbook – The spreadsheet document

Worksheet – Pages within the Workbook

Cells and Cell Ranges

Rows and Columns: 1,048,576 rows and 16,384 columns.



## Cell Referencing

DevExpress has a variety of ways to reference cells.

***Note that array referencing is 0-based, so Column “A” is index 0, and the first row is index 0.***

Cell referencing for Cell A1

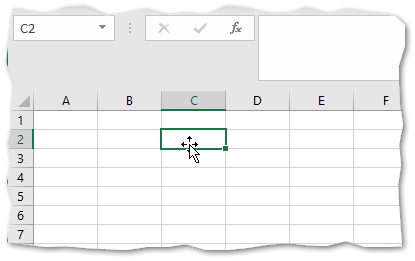
* Worksheet.cells[“A1”] // where the letter (“A”) is the column
* Worksheet.cells[0,0] // This is in row, column format
* Worksheet.cells.Columns[0],[“1”]

Cell referencing for Cell C2. Column “C” is index 2, “2” is index 1

* Worksheet.cells[“C2”]
* Worksheet.cells[1,2] // Note that row is first
* Worksheet.cells.Columns[“C”],[1]
* Worksheet.cells.Rows[1][“C”]

# Appendix – Using the EPPlus Excel Spreadsheet

The EPPlus Excel interface is a free Open Source API.



## Components

## Cell Referencing

The method “Cells()” for a Worksheet retrieves a CellRange, which is 1 or more cells.

Cells can be referenced in multiple ways.

An array reference is always [row, column]

***Note that EPPLUS referencing is 1-based. So, the column A is index 1, and the first row is index 1***

Cell Referencing for cell “A1”:

Worksheet.Cells[“A1”].Value

Worksheet.Cells[1,1].Value

Cell Referencing for cell “B1”:

Worksheet.Cells[“B1”].Value

Worksheet.Cells[1,2].Value

Cell Referencing for “B3”:

Worksheet.Cells[“B3”]

Worksheet.Cells[3,2]

References:

<https://github.com/JanKallman/EPPlus/wiki/Getting-Started>