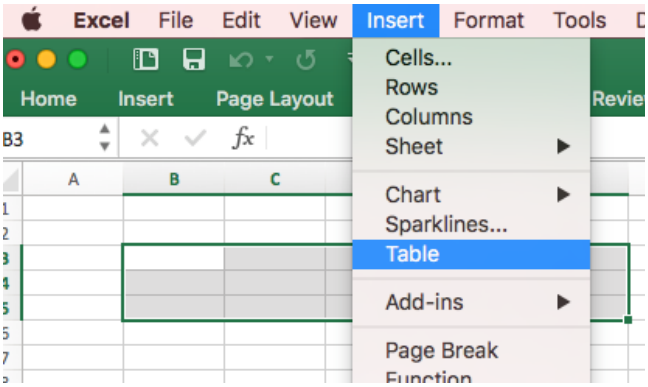
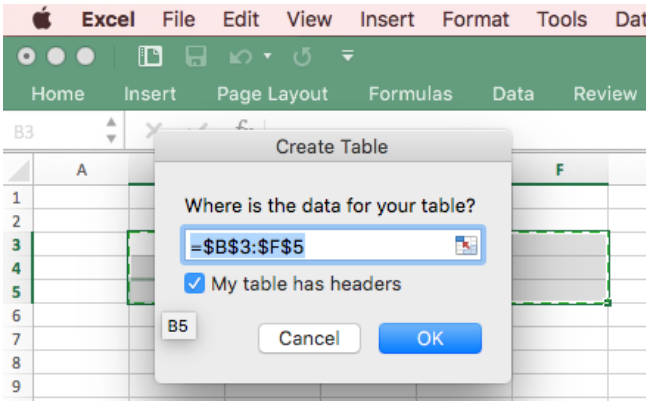
ExcelInjector

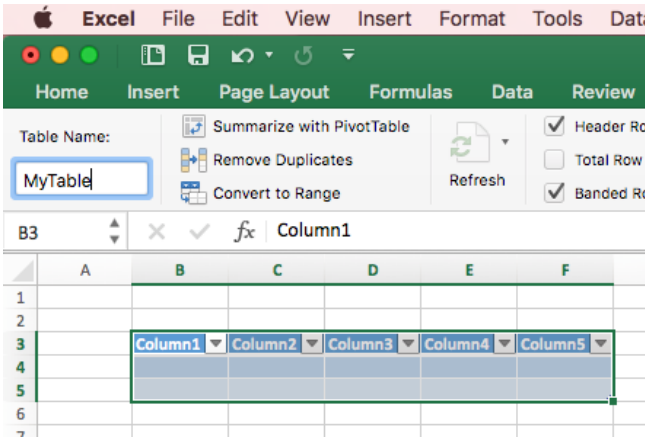
ExcelInjector is a free, java command-line utility that allows you to inject merge-formatted data tables and individual cell data into a pre-existing Excel file.

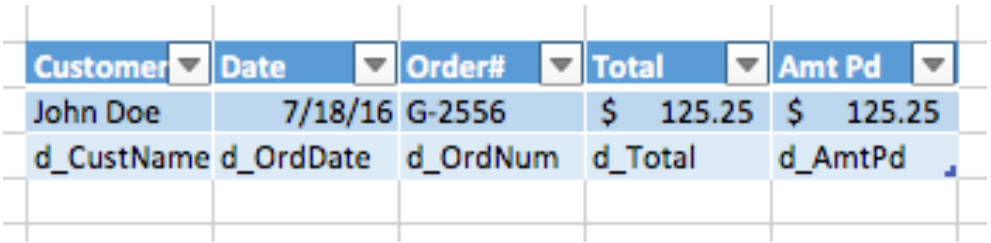
In order to inject cell data, you simply include instructions as specified below in your instruction list when calling the command-line utility.

In order to inject merge-formatted data, you must create a Table object in Excel with headers and must make exactly three rows for the table when creating it. The first row holds the column headers. The second row contains example data formatted as desired. The third row contains the names of the columns in the merge file. The table itself should also be named something specific, as this too will be referenced in the instruction list when calling the utility.









The above pictures illustrate the process of creating a Table object in Excel. First select a range of rows. Second select Insert > Table. Third, pick “My table has headers” from Create Table dialog. Fourth, select the Table tab and provide a name for the table. Finally, change the default “Column1” column headers to anything you would like, input sample data on the first data row and Merge file column headers on the second column row.

Calling ExcelInjector from Command-line

To call the utility from the command-line while working in the same directory as the utility:

java -jar ExcelInjector.jar <parameters>

<perameters> are explained below.

Parameter Listing:

-c Command to perform
    updateCell - update a specific cell with provided data
    injectTable - inject a single merge file into a single table
    injectMultiple - a list of table and cell injection instructions

-s Source Excel file path or, if -p is used, name of file within data folder. File must already exist.

-d Destination Excel file path, or if -p is used, name of file within data folder. If file exists, it will be overwritten.

-m Merge file path or name if -p is used. For use with injectTable function only (not injectMultiple).

-t Name of Excel table object to populate when using injectTable function. Not used with injectMultiple function.

-n Sheet number. Default is 0. The injectMultiple function provides an alternative means to encode sheet number per instruction.

-a Cell position. For use with updateCell function only. The specification of a specific cell to update.

-v Cell value. For use with updateCell function only. The value to insert into the specified cell.

-p An optional path provided to be applied to parameters -s, -d, -m and -b if said parameters do not contain a forward or backslash.

-b A list of instructions for use with the injectMultiple function. Each instruction is one of type types, contains three sub-parameters as defined below, and each should be separated by a ~ (tilde) character. See below for more detail on this parameter.

Details for injectMultiple -b parameter:

Syntax for each instruction:

Type:ExcelReference=DataReference

Type can be cell or table.

If type is cell, ExcelReference must be a valid cell reference, such as B2, and DataReference must be the data to insert.

If type is table, ExcelReference must be the name of a Table object within the Excel file and DataReference must be the name or path to a merge-formatted data file.

Either cell or table instructions can specify a specific sheet by including square brackets containing the zero-based sheet number within the ExcelReference sub-parameter. For example:

cell:B2[1]=Text
table:MyTable[3]=SomeMergeFile.mer

An example of multiple instructions submitted for the injectMultiple command:

-c injectMultiple -s ExcelSourceFile.xlsx -d ExcelDestinationFile.xlsx -p /users/jerickson/desktop -b table:My Table 1=Merge1.mer~cell:B2=Some Text~table:Another Table[1]=Merge2.mer

In the above, three instructions are provided to create a new Excel file called ExcelDestinationFile by updating an existing file called ExcelSourceFile. First a table called “My Table 1” on the fist sheet (sheet 0) is populated using a merge file called Merge1.mer. Second, cell B2 is updated to contain “Some Text” and third, a table called “Another Table” on the second sheet (sheet 1) is populated with the data from “Merge2.mer.” Both merge files and both the source and destination Excel files are to be found/created on the desktop.

Using ExcelInjector from within FileMaker

To use the ExcelInjector from within FileMaker, use either Perform AppleScript from the Mac or Send Event from Windows.

```
20
21 # POPULATE Instructions
22 Set Variable [ $instructions ; Value: Get(ScriptParameter) ]
23
24 # SETUP PARAMS FOR UTILITY
25 Set Variable [ $cmdParams ; Value:  " -c injectMultiple " &  " -s " & $excelName &  " -d " & $excelName &  " -p " & $$path &  " -b " & Substitute ( $instructions ; ¶ ; "~" ) ]
26
27 # INJECT DATA
28 If [ Get ( SystemPlatform ) = 1 ]
29     Set Variable [ $script ;
       Value: Let ( [  script = "do shell script \"java -jar -Xms500m -Xmx1G " & $jarPath &  $cmdParams  & "\""  ] ;  Substitute ( script ; Get ( SystemDrive ) ; "/" )  ) ]
30     Perform AppleScript [ $script ]
31 Else
32     Set Variable [ $script ; Value: Let ( [  script = "cmd.exe /c java -jar -Xms500m -Xmx1G " & $jarPath &  $cmdParams  ] ;  Substitute ( script ; Get ( SystemDrive ) ; "/" )  ) ]
33     Send Event [ “aevt” ; “odoc” ; $script ]
34 End If
35
```

In the above pictured utility script, a script parameter is passed containing the instructions for the -b parameter and then $cmdParams is created using additional parameters $excelName and $$path, which are not shown but would be the name of an Excel file and a defined data folder path. Then based on SystemPlatform, a final $script parameter is built to either tell AppleScript to “do shell script ‘java -jar…’ or Send Event ‘cmd.exe /c java -jar…’

These are the basics required to use the utility from within FileMaker. Obviously several other steps are necessary, such as collecting and exporting a set of records, writing out the utility and an Excel source file, defining instructions, etc. For a complete example of doing all of this in FileMaker, see http://github.com/linearblue/ExcelInjector for an example .fmp12 file.