Technical White Paper

Automating Excel Reports

Five Approaches for Java Developers

Table of Contents

Java Developers Asked to Automate Excel Reports

Java developers are often asked to provide company personnel with timely reports needed for business decisions and operations. There may be hundreds of users who rely on the information in the reports, across different departments, functions and locations. The data required for the reports may be stored across multiple systems and files. Since the majority of business users are already comfortable manipulating data in Excel, a common request is for the reports to be delivered as Excel files.

In some cases the developer is asked to replace an existing application because the current report output is limited or inflexible. In other cases the new application, when delivered, will replace tedious manual processes including querying data sources, cutting and pasting results into Excel files, and manipulating formats.

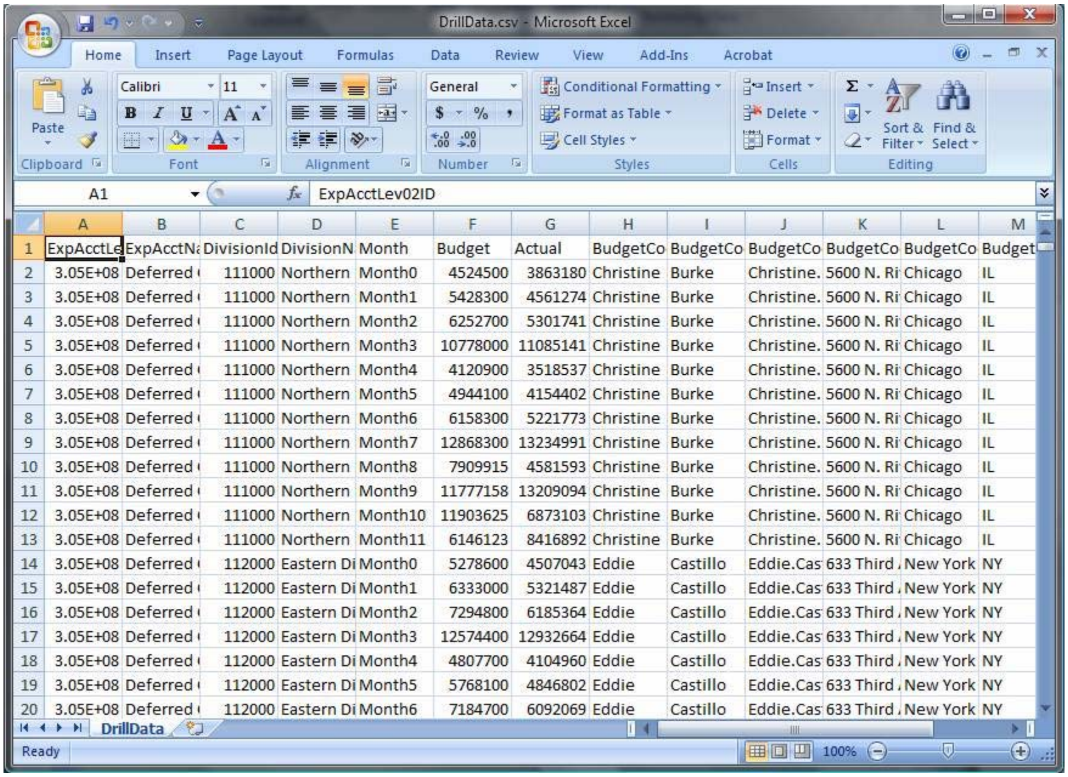
Five alternative approaches are available for automating the delivery of Excel-based reports, each with its own set of advantages and disadvantages. This paper describes the five approaches and where each can be best applied.

Approach #1: Hand-Code Delimited Output

A common first approach to consider is to write code that outputs delimited data (e.g. tab or comma) to Excel by setting the content (MIME) type to that of Excel and separating the columns with a tab or comma character. Excel then knows how to load the data and where to start each new column. Alternatively, the developer can create a file and set the file extension to one known by Excel. Excel will then read the file and display the information in the appropriate rows and columns.

Hand-coding is the fastest way to get started. There is no external tool or API to source or learn, there are no purchase approvals to deal with, and Java developers already have the required expertise to begin work immediately.

Excel will use defaults for all formatting. It will use defaults for items like column width, font, style, alignment, and word wrap. Any text that is longer than the default column width will not be displayed beyond the right edge of the cell, and numbers longer than the cell width will be displayed in scientific notation. Headings will not appear in bold; totals will not be separated from the other cells with borders. This approach limits the spreadsheet to a single worksheet with no charts, and use of formulas is extremely cumbersome. The output appears as a data dump. A sample report is displayed in this example:



A1 | ExpAcctLev02ID

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ExpAcctLe	ExpAcctNa	DivisionId	DivisionN	Month	Budget	Actual	BudgetCo	BudgetCo	BudgetCo	BudgetCo	BudgetCo	Budget
2	3.05E+08	Deferred	111000	Northern	Month0	4524500	3863180	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
3	3.05E+08	Deferred	111000	Northern	Month1	5428300	4561274	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
4	3.05E+08	Deferred	111000	Northern	Month2	6252700	5301741	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
5	3.05E+08	Deferred	111000	Northern	Month3	10778000	11085141	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
6	3.05E+08	Deferred	111000	Northern	Month4	4120900	3518537	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
7	3.05E+08	Deferred	111000	Northern	Month5	4944100	4154402	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
8	3.05E+08	Deferred	111000	Northern	Month6	6158300	5221773	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
9	3.05E+08	Deferred	111000	Northern	Month7	12868300	13234991	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
10	3.05E+08	Deferred	111000	Northern	Month8	7909915	4581593	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
11	3.05E+08	Deferred	111000	Northern	Month9	11777158	13209094	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
12	3.05E+08	Deferred	111000	Northern	Month10	11903625	6873103	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
13	3.05E+08	Deferred	111000	Northern	Month11	6146123	8416892	Christine	Burke	Christine.	5600 N. Ri	Chicago	IL
14	3.05E+08	Deferred	112000	Eastern Di	Month0	5278600	4507043	Eddie	Castillo	Eddie.Cas	633 Third	New York	NY
15	3.05E+08	Deferred	112000	Eastern Di	Month1	6333000	5321487	Eddie	Castillo	Eddie.Cas	633 Third	New York	NY
16	3.05E+08	Deferred	112000	Eastern Di	Month2	7294800	6185364	Eddie	Castillo	Eddie.Cas	633 Third	New York	NY
17	3.05E+08	Deferred	112000	Eastern Di	Month3	12574400	12932664	Eddie	Castillo	Eddie.Cas	633 Third	New York	NY
18	3.05E+08	Deferred	112000	Eastern Di	Month4	4807700	4104960	Eddie	Castillo	Eddie.Cas	633 Third	New York	NY
19	3.05E+08	Deferred	112000	Eastern Di	Month5	5768100	4846802	Eddie	Castillo	Eddie.Cas	633 Third	New York	NY
20	3.05E+08	Deferred	112000	Eastern Di	Month6	7184700	6092069	Eddie	Castillo	Eddie.Cas	633 Third	New York	NY

Figure 1. Example .csv report.

Because of the formatting limitations, for this approach to work users need to be tolerant of reports that look like a raw data dump. As the application becomes more successful and a broader set of users begins to rely on the reports, the number of users dissatisfied with the formatting may grow. In addition, the requirements for what once sufficed as a very simple, single worksheet may grow over time, with requests for more sophistication causing the developer to seek out a more full-featured solution. Because of this, Excel reports hand-coded in Java typically have a limited lifespan.

In summary, the hand-coding approach can be successful when speed of delivery for the first report is the most important consideration, and when the solution is considered temporary.

Approach #2: Use an OS Library to Hand-Code Excel Output

Another approach is to use an open source Java tool that produces Excel files. These libraries are readily available, and are free. They allow the developer to produce Excel files with multiple worksheets, and with formulas. They also give the developer control over basic Excel formatting elements such as column width and font treatment. With these controls, open source libraries can be used to deliver reports that are easy to read and intuitive.

When using an open source library, the developer reads in or creates an Excel file using the library, then programmatically iterates over a data set to drop the correct data fields into the correct cells in the pre-defined spreadsheet. The developer programmatically controls groupings, sub-totals and totals, labels, page breaks and so forth, using control break logic. Once into this task, if the developer discovers that a new row or column or a different grouping is needed, it can be very time consuming to make those changes.

While it can take some effort to get the control break logic in place and working, a nicely formatted Excel output file is attainable.

There are two limitations to the open source library approach that should be considered. The first has to do with limitations in the report output. While the open source libraries support the basic Excel features, they do not support the richer features such as pivot tables, auto filters, and outlines. They also do not provide a charting API. The more successful the application becomes, the more need there may be for reports with these richer Excel features, and with charts.

The other important consideration is the time allocation required for ongoing maintenance. When business needs change, the application must be changed. When the reports are hand-coded, this means the code has to be changed. Every change in report layout or input data typically requires a code change. Complex reports that have multi-level control breaks can be very time consuming to change; for example, a customer report grouped and sub-totaled by country, customer, and orders. The more complex the report, the more time-consuming it is to change the code.

In summary, open source libraries are an effective means for delivering a nicely formatted Excel file and can be a viable alternative when the reports are relatively simple (single-level groupings, etc) and the number of reports is small. With this, the effort for ongoing code changes will be manageable.

Approach #3: Use an Excel Plug-In

A number of report server vendors provide an Excel plug-in which embeds a custom toolbar in Excel that allows users to access and query data sources directly or through metadata and create their own reports. While this is not a Java-based approach, it is included here as it is an approach that is often considered by Java developers. This approach gives users flexibility in trying different “what if” scenarios, spreadsheet models and layouts. The user controls the report format and the developer does not need to be involved at all.

Using a plug-in, the user can create reports and store them on a server, thereby providing shared access to the report files. But since it is not a server-generated solution it does not produce dynamic reports.

Policies related to users accessing corporate data sources may come into play. The user will need to learn how to configure and access data sources and write queries, and will need to be supplied with the right local database drivers.

When creating an Excel report, users tend to logically group information across different tabs, creating a tab for each product, for each region, etc. With a plug-in, users will not be able to dynamically create new tabs based on values in the data. Plug-ins operate on one worksheet at a time and depend on a fixed number of worksheets instead of dynamically bursting worksheet pages as dictated by the data.

To run a report, plug-ins use an Excel template and populate each pre-defined cell with a value. Because the data fields must be fixed in the template ahead of time, the report cannot dynamically expand across the page at run time based on the actual data. For example, a report cannot dynamically fill in data for those periods that have transpired to date without having “placeholder” columns that will appear blank for those periods not yet transpired. Likewise, Excel templates cannot expand to accommodate new periods for subsequent years without those years being pre-determined. An Excel-based template that summarizes by the dimension of 'time' will need to be modified every time the time period exceeds the preset boundaries of the document, else the report will “break.”

While plug-ins support some advanced features such as charting and pivot tables, other features such as outlines are not supported.

Some plug-ins impose further limitations to the report layout. Many require that all data fields in a group be listed together in the report. So for example, if the data set groups cost data by product, the report could not separately group hardware costs, selling costs or other costs. Some plug-ins limit the operations that can be used to one type of operation per report, so if the report needs to SUM some data, the only operation that can be used with other data is SUM.

Plug-ins also tend to limit the data source options. Typically, the data source is limited to that provided by the report server vendor (and, of course, there must be a report server in place). Some plug-ins that connect to alternate data sources require those sources be a flat file structure. In these cases, hierarchical data structures will have to be normalized by creating a separate data warehouse, or building a data mart for the application using complex SQL queries.

A related option is to use Excel 2007 to create reports, as Microsoft has added an ODBC query editor in the 2007 release of Excel. This option is very similar to the plug-in solution described here, except it is not tied to a particular data source. With Excel 2007, users have the option of specifying columns within the data query itself, or creating a pivot table.

Approach #4: Use Excel Services & Microsoft SharePoint Server

While Excel Services do not represent a Java-based approach, it is natural for a Java developer to give them consideration and as such they will be considered in this paper. Excel Services are part of Microsoft Office SharePoint Server 2007 and provide a server-based Excel option. With this option a spreadsheet is created using Excel 2007 then converted to HTML in the SharePoint portal server. The portal server creates interactive, web-page versions of spreadsheets and dynamically refreshes them with up-to-date data when opened. The advantage of using Excel Services over an Excel plug-in is that it is a server-based solution, and it does not restrict access to a single data source.

Since this approach is based on an Excel report template, it has the data layout limitations of the Excel plug-in approach outlined above. It should also be noted that Excel Services does not yet support all the features of Excel. If an unsupported feature is used when building the report there may be problematic behavior when a user accesses the report. In some cases the file will not load, in other cases the unsupported aspects of the file will not be displayed or the user will be unable to manipulate the file in certain ways. (For example, report users cannot add or rearrange fields in a pivot table report). So developers will need to be thorough in testing the user experience and carefully set user expectations.

Approach #5: Use a Report Designer and Deployment Engine

The final approach to consider is to use an Excel report designer that is purpose-built for creating Excel reports, and use it with a report generation engine for deploying the reports. Excel report designers pair a visual development environment with a scripting engine, providing a means for even the most sophisticated reports to be built quickly and easily, without coding. Since the reports are built without writing code, they can be changed and updated without modifying code. Excel report designers slash the time needed to create reports but more importantly, they slash the developer time needed to maintain reports.

Because Excel report designers are built to create virtually any report, the developer has complete freedom with the data layout. The developer is free to design reports with any layout including crosstab formats that expand dynamically across columns and worksheets, and financial statement formats with inter-row and inter-column calculations.

Key characteristics of an Excel report designer include the following:

- ***Purpose-built for Excel report development.*** A true Excel report designer will include a rich array of capabilities specific to report development, such as the ability to burst data dynamically down and across a page and across worksheets, without writing or maintaining any code; the ability to set up dynamic, cascading user prompts for parameter input; and dashboards that provide a quick overview of the report infrastructure.

- ***Allow reports to be changed or updated on the fly, without having to modify code.*** With an Excel report designer, the developer can create or modify a data set or add a new grouping level in a few clicks. Using the visual development environment and scripting engine, changes that could take hours, days or longer with other approaches can be accomplished in a few clicks. Excel report designers are extremely usability focused, providing the ability, for example, to visually build a query and see the relationships between tables, right click for controls, drag and drop data elements, and use control key shortcuts.

- ***Easily support any data layout, and provide exact control over formatting.*** A true Excel report designer will support all the data layouts that may be requested by users or dictated by the data, which are virtually unlimited. One commonly requested data layout is a report with inter-row and inter-column calculated values, typically used for financial statements. Another commonly requested layout is one that shows “sibling” data side by side, such as budget and actual figures. Multi-sheet workbooks with a summary sheet on top are also extremely common. A true Excel report designer will easily support any data layout. It will also provide complete formatting freedom with support of text cells, merged cells, rotated text, hyperlinks, and the like. It will allow the data to dictate the rows, columns, and worksheets, never requiring “placeholder” rows or columns for empty data.

- ***Works with a pure Java spreadsheet engine*** that generates spreadsheet reports in a way that is scalable.

About Actuate e.Spreadsheet Designer

Actuate e.Spreadsheet Designer provides a visual report development environment coupled with a powerful scripting engine for quickly building full-featured Excel report templates without coding. Purpose-built as a true Excel report designer, it connects to any number of hierarchical data sources and provides a host of powerful report development features such as:

- A built-in understanding of date relationships so that report data can be grouped automatically into months and quarters.

- The ability to easily set up dynamic cascading user prompts that filter the data or affect the structure of the spreadsheet that is rendered. For example, a prompt that asks the user for State, then presents the appropriate City selections based on the user’s answer. The prompt screen might also ask the users for viewing preferences, for example, if they want to view the data in a chart or matrix format, or if they want the data displayed on just one worksheet or divided into many.

- A dashboard that provides a quick overview of data sources, data sets, parameters and security settings, as well as easy access to definitions for each scripting macro.

With a powerful package of usability features, this pure Java application provides an extremely fast and dependable way to create and maintain even the most sophisticated reports. Because the reports are fully data-driven, the developer has exact control over the layout and never needs empty "placeholder" columns. e.Spreadsheet Designer supports a virtually unlimited variety of report formats, including:

- Three-dimensional multi-sheet crosstab reports that include cross-sheet references such as a table of contents or a summary worksheet that subtotals the values on other worksheets. The developer even has exact control over the ordering of the tabs.
- Spreadsheet reports with complex inter-row and inter-column calculations.
- Grid-style reports with irregular formatting, where selected cells are formatted differently than others or use a different data source.
- Reports with drill-to-detail using Excel's ability to expand-to-detail and collapse-to-summary; reports that contain Visual Basic code, used for Excel Macros; 2D and 3D charts; pivot tables; and reports that pull together data from multiple hierarchical data sources.

With e.Spreadsheet Designer the developer can deliver charting and complex reports with drill down and interactive capabilities faster than it would take to deliver reports that are much more basic using another approach. Coupled with one of Actuate's deployment solutions, e.Spreadsheet Designer allows the Java developer to create, deploy and maintain any number of complex reports in a fraction of the time that would be required using other approaches. For more information on e.Spreadsheet Designer:

e.Spreadsheet Designer tutorial (*basic*):
http://www.birt-exchange.com/modules/wfdownloads/singlefile.php?cid=4&lid=213

e.Spreadsheet Designer tutorial *(advanced, including conditional formatting, charting from data ranges, using dynamic parameters):*
http://www.birt-exchange.com/modules/wfdownloads/singlefile.php?cid=4&lid=94

Using VBA (Excel Macros) with e.Spreadsheet Designer:
http://www.birt-exchange.com/modules/wfdownloads/singlefile.php?cid=4&lid=178

Sheet bursting with e.Spreadsheet:
http://www.birt-exchange.com/modules/wfdownloads/singlefile.php?cid=4&lid=140

e.Spreadsheet Data Range Mechanics (*a tutorial focused on data manipulation):*
http://www.birt-exchange.com/modules/wfdownloads/singlefile.php?cid=4&lid=144

To download a free 90-day trial of e.Spreadsheet Designer:
http://www.birt-exchange.com/modules/wfdownloads/staticdls_index.php

ACTUATE

Email: essinfo@actuate.com
1-800-884-8665
www.actuate.com

WP-Automating Excel Reporting-V3