User Interface - Feature #12061

emulate the Excel.Application automation object

10/20/2026 06:24 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 10/20/2026 06:25 PM - Greg Shah

Emulate the Excel.Application automation object

Project: User Interface

Problem

Applications drive Microsoft Excel through CREATE "Excel.Application" for several purposes:

- importing report text into a sheet through Workbooks:OpenText or a QueryTable, then leaving the workbook open for the user — by far the most common case;
- formatted exports: Range:Value, Font:Bold, Interior:Color, Borders, Merge, PageSetup;
- filling templates through bookmarks or defined names;
- native Excel charts;
- dumping data with SaveAs, and generating blank templates that users fill in and save as CSV for re-import;
- pasting the clipboard into the active sheet.

In a converted system there is no Excel on the server, ComServer.create cannot reach Excel on Linux, and the user is at a web browser.

Work

1. Inventory the Excel object model members used: Workbooks, Worksheets, Range, Cells, QueryTables, Charts, ActiveSheet, Selection, the formatting members and the constants passed to them.
2. Implement a Java emulation of that subset, registered for CREATE "Excel.Application" behind ComServer.create, which builds the workbook with Apache POI on the server.
3. Reproduce the type detection Excel applies when text is imported, which reports depend on: dates by locale, numbers, and leading zeros.
4. Where the program saves the workbook, write the file on the server.
5. Where the program leaves the workbook open for the user (Visible = TRUE), deliver the generated workbook to the user's browser. The default is a download of an .xlsx file; opening it in the FWD SPREADSHEET widget is an option to be decided per project.
6. Native Excel charts: implement them with POI only if a project needs them.
7. Update the gap marking for the class.

Testcases

Each with a Windows OpenEdge and Excel baseline, comparing the resulting workbook cell by cell (values, types and formats):

- a report text file imported with tabs and fixed-width columns, including dates, numbers with leading zeros and negative numbers;
- a formatted export using every formatting member in the inventory;
- a template filled through defined names;
- SaveAs in .xlsx and CSV form.