Tabular Browser Design Documentation

Nicholas DiOrio, National Renewable Energy Laboratory

March 2, 2016

1 Introduction

The tabular results browser in SAM provides the user the capability to easily display and compare matrix data. This document is intended for internal use only to provide reference for the design of the tabular browser in case of future problems. Basic code paths will be listed, as well as intended behavior. For functions which have been particulary difficult to get correct, inputs, outputs, and additional details will be explained.

2 Code Paths

2.1 Running a simulation

The general code path which occurs when running a simulation is:

```
MainWindow::OnCaseMenu
   CaseWindow::RunBaseCase
      Simulation::DispatchThreads
      CaseWindow::UpdateResults
         CaseWindow::Setup
            TabularBrowser::Setup
               TabularBrowser::UpdateAll
         CaseWindow::GetTabularBrowser
            TabularBrowser::SelectVariables
               TabularBrowser::UpdateAll
```

As can be seen, the tabular browser is updated twice when a simulation is run. More investigation needs to be done to understand if this is correct. Within TabularBrowser, a new or updated simulation results in calls to UpdateAll. This function encapsulates checks for adding variables, removing variables, and updating variable sizes on the existing grids. The following illustrates the order of the calls within UpdateAll, with the understanding that some logic exists to choose when to call these functions or not.

```
TabularBrowser::UpdateAll
   PopulateSelectionList
   TabularBrowser::RemoveUnusedVariables
   TabularBrowser::ProcessRemoved
   TabularBrowser::ProcessAdded
   TabularBrowser::SetLastSelection
```

2.2 OnPageClosed

When a page in the tabular browser is closed, a page-closed event is generated, at which point internal data structures need to be cleaned up, and the existing grids need to be updated. The primary tasks which must occur are to remove all of the variables on the closed-grid and then redraw the existing grids.

```
TabularBrowser::OnPageClosed
   TabularBrowser::ProcessRemovedAll
   TabularBrowser::UpdateAll
```

2.3 OnPageChanged

Every time a user clicks on a notebook tab which results in a change of page from the previously selected tab to the new selection, a page-changed event is generated. The event handling for this is relatively simple. The code simply points local variables to reflect the currently selected grid.

2.4 OnVarSel

Every time a user clicks to add or remove a variable from the current grid, a variable-select event is generated. To handle this event, the code simply checks whether the check-box is selected or deselected and then processes the addition or removal:

```
TabularBrowser::OnVarSel
   TabularBrowser::ProcessAdded
   TabularBrowser::ProcessRemoved
   TabularBrowser::SetLastSelection
```

3 Data Structures

The internal data structures relevent to the TabularBrowser are:

`m_notebook`	the wxAuiNotebook which holds a wxExtGridCtrl on every tab
`m_grid`	the currently selected wxExtGridCtrl
`m_gridTable`	the underlying table containing the data for the currently selected grid
`m_gridMap`	a map which associates grid size to a pointer to that grid
`m_gridTableMap`	a map which associates grid size to a pointer to the underlying data table
`m_tabLabelsMap`	a map which associates grid size to the label of the tab
`m_selectedVars`	a string array containing names of every variable selected across all grids
`m_selectedVarsBySizeMap`	a map which associates variable name to the grid size
`m_selectedVarsMap`	a map which associates grid size to the array of variable names on that particular grid
`m_numberOfTabs`	the number of tabs (grids)
`m_lastSize`	the size of the last selected grid
`m_key`	an integer which increments when a matrix is added, allowing a way to keep track of grids of the same size

4 Main Functions

4.1 ProcessAdded

Inputs
`wxString name` - the name of the variable to add

Outputs
`none`

Description The `ProcessAdded` function is called for one variable at a time. The function checks whether or not the size of the grid associated with the variable has changed. If the size has changed, `ProcessRemoved` is called before proceeding. If `ProcessAdded` is called and a variable needs to be added, it will be added to `m_selectedVars`, `m_selectedVarsMap`, and `m_selectedVarsBySizeMap`. The variable size will be used to update `m_lastSize`. Finally, the notebook, grid, and case will be updated.

4.2 ProcessRemoved

Inputs
`wxString name` - the name of the variable to remove
`bool update_grid` - boolean choice on whether to call for grid update

Outputs
`none`

Description The `ProcessRemoved` function is called for one variable at a time. The variable is removed from `m_selectedVars`, `m_selectedVarsMap`, and `m_selectedVarsBySizeMap`. In the event that more variables exist on the current grid, the variable size will be used to update `m_lastSize`, and if `update_grid` is true, the grid and case will be updated. If no more variables exist on the current grid, a call is made to `ProcessRemovedAll`.

4.3 ProcessRemovedAll

4.4 GetVariableSize

4.5 CheckSizeChanged

4.6 RemoveUnusedVariables

4.7 UpdateNotebook

4.8 UpdateGridSpecific

4.9 UpdateCase

4.10 LoadData

5 UI Hints

6 Bugs and Pitfalls

6.1 Changing from single-year to lifetime mode