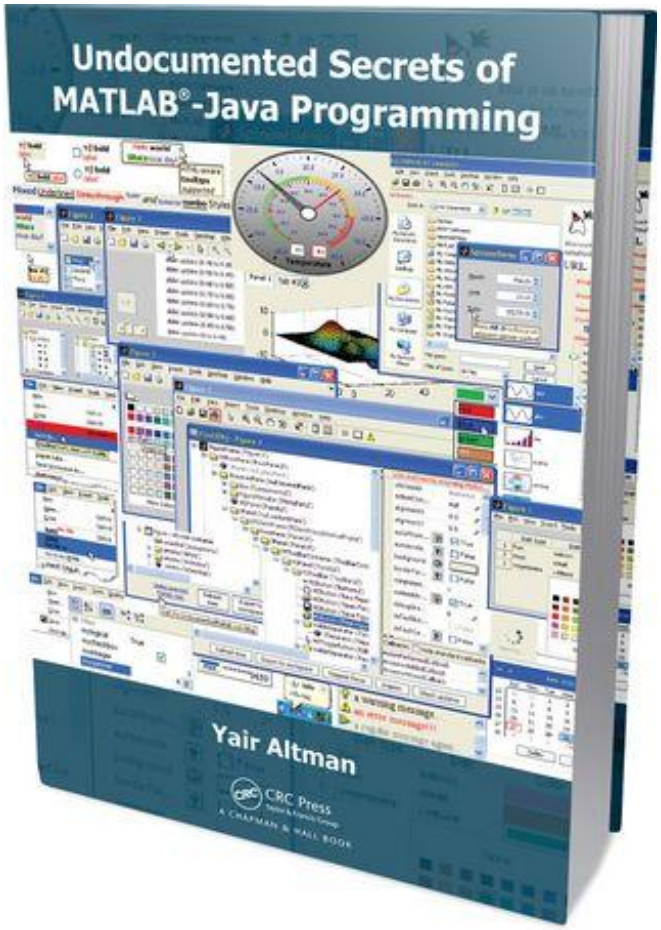
Filtering Matlab's Uitable Control

© Yair M. Altman

www.UndocumentedMatlab.com



MATLAB Table data filtering can be done in several ways. Matlab includes built-in filtering support using the `GlazedList` package, and JIDE includes built-in filtering support for its grids (tables), as shall be shown below.

However, I believe that a much better table data filter package is the open-source `TableFilter` package by Luis Pena.[1] All we need to do to implement filtering on Matlab ***uitable***s is the following:

1. Download the latest `TableFilter` jar file (it will be named something such as tablefilter-swing-4.2.0.jar)[2]

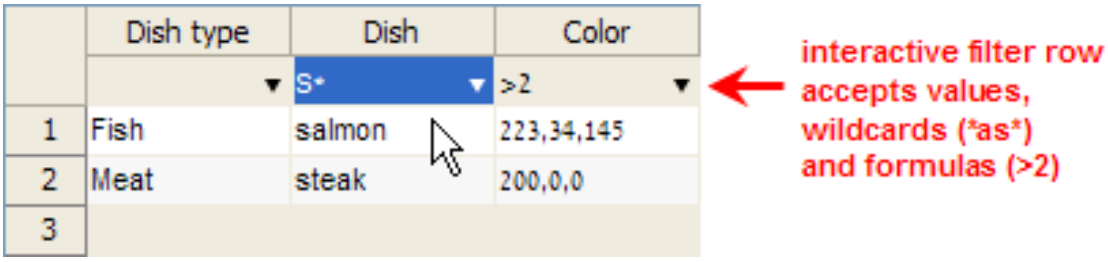
2. Add this zip file to Matlab's Java classpath, either statically (in *classpath.txt*) or dynamically (using the ***javaaddpath*** function)

3. In Matlab, use the ***findjobj*** utility[3] to get the Matlab ***uitable's*** underlying Java reference handle (`jtable`):

   ```
   htable = uitable(...);
   jScrollPane = findjobj(htable);
   jtable = jScrollPane.getViewport.getView;
   ```

4. Then in Matlab, add the filter to the table's Java reference by calling `net.coderazzi.filters.gui.TableFilterHeader(jtable)`:

   ```
   javaaddpath('tablefilter-swing-4.1.4.jar');
   filter = net.coderazzi.filters.gui.TableFilterHeader(jtable);
   ```



Automatic table filtering using `TableFilter`

`TableFilter` provides the ability to automatically populate filter drop-down (combo-box, popup-menu) values from the list of distinct data values in each table column. Note that enabling this may take some time to run, depending on the amount of data. For this reason, this feature is turned off by default. To turn it on, run this:

```
filter.setAutoChoices(net.coderazzi.filters.gui.AutoChoices.ENABLED);
```

---

[1] http://coderazzi.net/tablefilter (or: http://bit.ly/lRqhgL); http://code.google.com/p/tablefilter-swing/ (or: http://bit.ly/iUrzKW)

[2] http://www.coderazzi.net/tablefilter/download.html (or: http://bit.ly/kyh4ZD); http://repo2.maven.org/maven2/net/coderazzi/tablefilter-swing/ (or: http://bit.ly/jZUC2C)

[3] http://UndocumentedMatlab.com/blog/findjobj-find-underlying-java-object/ (or: http://bit.ly/8YFRnE)

Custom choice sets can be prepared. For example:[4]

```
data = {'option #1', 'option #2', 'option #3'};
choices = net.coderazzi.filters.gui.CustomChoice.createSet(data);
filter.getFilterEditor(colIdx).setCustomChoices(choices);

disabled = net.coderazzi.filters.gui.AutoChoices.DISABLED;
filter.getFilterEditor(colIdx).setAutoChoices(disabled);
```

You may also want to similarly modify other filter properties, for example the maximal number of displayed drop-down options:

```
filter.setMaxVisibleRows(12);  % default=8
```

Note that the filter also installs its own sorting functionality in the table's header. Therefore, if you have enabled table sorting by any other mechanism (e.g., standard JIDE sorting), you will have duplicate sorting functionality when clicking the column header. For this reason you may wish to disable the JIDE table sorting, and just use the filter's sorting. Doing the reverse, i.e. disabling the filter's sorting and keeping the standard table sorting, is also possible, although a bit more difficult:

```
% Disable the standard JIDE sorting
jtable.setSortable(false);

% Disable the filter sorting
for colIdx = 0 : jtable.getColumnCount-1
   jtable.getRowSorter.setSortable(colIdx,false);
end
```

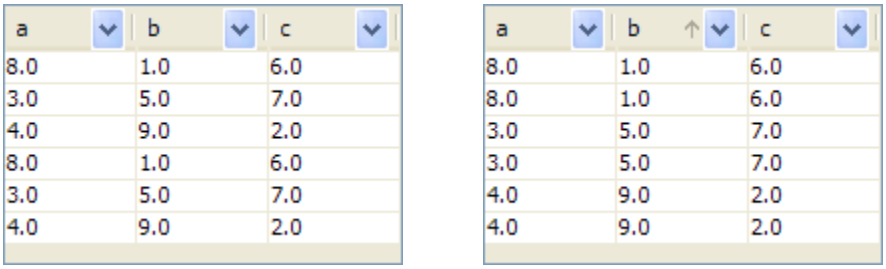
To retrieve the actual data in a filtered table, use the table's `RowSorter` object. For example, the following code returns the selected (filtered) row's data upon selection a specific table cell with the mouse:

```
set(jtable.getSelectionModel, 'ValueChangedCallback', ...
    {@selectionCallbackFcn,jtable});

function selectionCallbackFcn(jModel, jEventData, jtable)
   rowIdx = get(jModel,'LeadSelectionIndex');
   try
      rowIdx = jtable.getRowSorter.convertRowIndexToModel(rowIdx);
   catch
      % never mind: no filtering is used so stay with existing rowIdx
   end

   % Now do something useful with the selected row's data…
   data = jtable.getModel.getValueAt(rowIdx,1);  % column #2

   % Alternate way to get the selected row's data…
   data = jtable.getActualRowAt(rowIdx);

   % ...
end  % selectionCallbackFcn
```

[4] This functionality has been enhanced in version 4.2 of `TableFilter`, making Matlab integration even easier

An altogether different method for table data filtering relies on the built-in filtering functionality of JIDE grids, on which the new ***uitable*** is based. Note that JIDE grids, with its associated filtering, can also be used in older Matlab releases that do not use JIDE grids for ***uitable***. A code snippet that demonstrates this functionality follows:

```
% An example for a simple (???) JIDE filter
data = num2cell([magic(3);magic(3)]);
jtable = com.jidesoft.grid.SortableTable(data,{'a','b','c'});
tableHeader = com.jidesoft.grid.AutoFilterTableHeader(jtable);
tableHeader.setAutoFilterEnabled(true)
tableHeader.setShowFilterName(true)
tableHeader.setShowFilterIcon(true)
jtable.setTableHeader(tableHeader)
installer = com.jidesoft.grid.TableHeaderPopupMenuInstaller(jtable);
pmCustomizer1=com.jidesoft.grid.AutoResizePopupMenuCustomizer;
installer.addTableHeaderPopupMenuCustomizer(pmCustomizer1);
pmCustomizer2=com.jidesoft.grid.TableColumnChooserPopupMenuCustomizer;
installer.addTableHeaderPopupMenuCustomizer(pmCustomizer2);
jScrollPane = javax.swing.JScrollPane(jtable);
[hjtable,hjcontainer]=javacomponent(jScrollPane,[20,20,200,150],gcf);
```



a	b	c
8.0	1.0	6.0
3.0	5.0	7.0
4.0	9.0	2.0
8.0	1.0	6.0
3.0	5.0	7.0
4.0	9.0	2.0

a	b	c
8.0	1.0	6.0
8.0	1.0	6.0
3.0	5.0	7.0
3.0	5.0	7.0
4.0	9.0	2.0
4.0	9.0	2.0

JIDE's built-in table filtering: unsorted (left) and sorted (right)

JIDE's filtering in integrated in the table header, taking up less space than Pena's `TableFilter` (which uses a separate filtering row). It is also built-in within JIDE, meaning that it is automatically available in Matlab, without requiring any installation (unlike Pena's `TableFilter`).

However, JIDE filtering is relatively difficult to set up, as shown in the complex snippet above that was required to produce even this simple example. Pena's `TableFilter` is obviously easier to set up. I also believe that Pena's `TableFilter` is more powerful and usable.

Read more about Matlab's ***uitable*** customization, and Matlab-Java programming in general, in the book *Undocumented Secrets of MATLAB-Java Programming* (CRC Press 2011, ISBN 9781439869031).[5]

[5] http://UndocumentedMatlab.com/matlab-java-book/