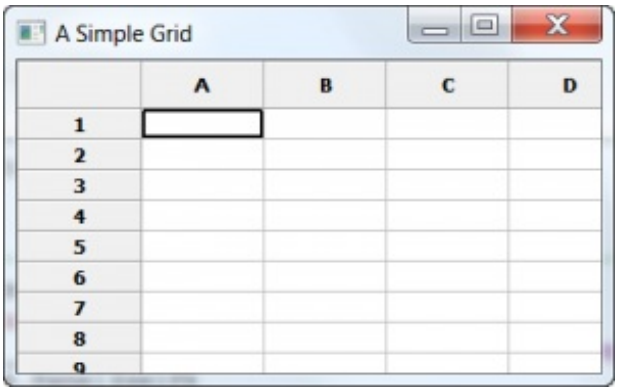
wxPython: An Introduction to Grids

The grid widget in wxPython is one of the most complex GUI elements that you'll work with in this toolkit. In this article you will learn the basics of grid creation and usage. One of the major uses for a grid is to display tabular data. Another use is to create some kind of spreadsheet. If you need something where you need a lot of cells that can be edited easily, then the grid widget is probably what you want. The ListCtrl in Report Mode is similar in appearance to the grid and can be used as a replacement for the grid depending on your needs.

Creating a Simple Grid

Let's take a look at how to actually create a grid:

```
import wx
import wx.grid as gridlib
```




```
########################################################################
class MyForm(wx.Frame):
    """"""

    #----------------------------------------------------------------------
    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, parent=None, title="A Simple Grid")
        panel = wx.Panel(self)

        myGrid = gridlib.Grid(panel)
        myGrid.CreateGrid(12, 8)

        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(myGrid, 1, wx.EXPAND)
        panel.SetSizer(sizer)

if __name__ == "__main__":
    app = wx.PySimpleApp()
    frame = MyForm().Show()
    app.MainLoop()
```

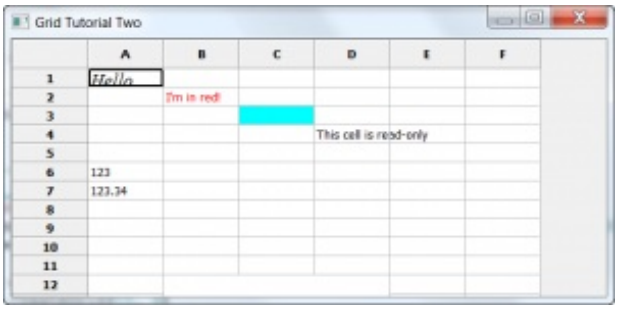
This is one of the easiest ways to create a grid widget. All we did here was instantiate a grid object and then call its CreateGrid method to tell it how many rows and columns we wanted. We put it in a sizer mainly because that's just the proper way to contain widgets and makes adding additional widgets easier. Now you should have a functional grid!

Introducing Some Grid Methods

There is a problem with the previous example: this grid doesn't really do anything useful! We need a way to put

data into the grid and get the data out. Let's find out how to do that and a whole lot more.

```
import wx
import wx.grid as gridlib
```




```
##########################################################################
class MyForm(wx.Frame):
    """"""

    #----------------------------------------------------------------------
    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, parent=None, title="Grid Tutorial Two",
size=(650,320))
        panel = wx.Panel(self)

        myGrid = gridlib.Grid(panel)
        myGrid.CreateGrid(15, 6)

        myGrid.SetCellValue(0,0, "Hello")
        myGrid.SetCellFont(0, 0, wx.Font(12, wx.ROMAN, wx.ITALIC, wx.NORMAL))
        print myGrid.GetCellValue(0,0)

        myGrid.SetCellValue(1,1, "I'm in red!")
        myGrid.SetCellTextColour(1, 1, wx.RED)

        myGrid.SetCellBackgroundColour(2, 2, wx.CYAN)

        myGrid.SetCellValue(3, 3, "This cell is read-only")
        myGrid.SetReadOnly(3, 3, True)

        myGrid.SetCellEditor(5, 0, gridlib.GridCellNumberEditor(1,1000))
        myGrid.SetCellValue(5, 0, "123")
        myGrid.SetCellEditor(6, 0, gridlib.GridCellFloatEditor())
        myGrid.SetCellValue(6, 0, "123.34")
        myGrid.SetCellEditor(7, 0, gridlib.GridCellNumberEditor())

        myGrid.SetCellSize(11, 1, 3, 3)
        myGrid.SetCellAlignment(11, 1, wx.ALIGN_CENTRE, wx.ALIGN_CENTRE)
        myGrid.SetCellValue(11, 1, "This cell is set to span 3 rows and 3 columns")

        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(myGrid)
        panel.SetSizer(sizer)

if __name__ == "__main__":
    app = wx.PySimpleApp()
    frame = MyForm()
```

```
    frame.Show()
    app.MainLoop()
```

The code above starts out much the same as the code in the first example. However, you'll quickly see that here we have many grid specific methods that we use to set various attributes of the grid widget. For example, we use *SetCellValue(0,0, "Hello")* to set the value of the top right cell to the string, "Hello". Next, we set the font size and style for that same cell by calling the following:

SetCellFont(0, 0, wx.Font(12, wx.ROMAN, wx.ITALIC, wx.NORMAL))

You can create wx.Font objects that use the fonts on your system as well. To set the font color, we use SetCellTextColour and to set the cell's background color we use SetCellBackgroundColour. As you can see, the grid's methods are pretty straightforward and have intuitive names. Of course, if you're American, then you'll notice that the naming is tends to be the British spelling (i.e. Colour instead of Color), so you need to watch out for that.

If you need to make a cell read only, you can do the following: SetReadOnly(row, col, isReadOnly=True). If you need to make the entire grid read-only, then use EnableEditing(False). Finally, if you need to set entire rows or columns to read only, then you'll want to use cell attribute function. Something like this should get you started:

GetCellAttr(row, col).SetReadOnly(isReadOnly)

Then use SetRowAttr() and SetColAttr() to set the respective row or column to read-only. I got this information from Robin Dunn's wonderfully helpful book: *wxPython in Action.* Be sure to also take note of SetCellAlignment which will set the cell's content's alignment.

UPDATE (2013-09-12): I was trying to figure out how to actually use the above information to make an entire row or column read-only, but I don't think it works or it doesn't work that way any longer. Instead, you should do something like this:

```
attr = gridlib.GridCellAttr()
attr.SetReadOnly(True)
myGrid.SetRowAttr(0, attr)
```

This code will make the first row (i.e. row zero) read-only.

The last section of importance in this example is how to set up custom cell editors. It's pretty simple in that all you need to do is call SetCellEditor and pass in the (row, col) tuple along with your editor of choice. In our example, we use GridCellNumberEditor, GridCellFloatEditor and GridCellNumberEditor. See the official documentation for other choices.

Grid Events

For our last example in our whirlwind Grid tour, we'll look at the grid's special events. Here's some code to get us started:

```
# gridEvents.py

import wx
import wx.grid as gridlib

########################################################################
class MyGrid(gridlib.Grid):
    """"""

    #----------------------------------------------------------------------
    def __init__(self, parent):
```

```
        """Constructor"""
        gridlib.Grid.__init__(self, parent)
        self.CreateGrid(12, 8)


        # test all the events
        self.Bind(gridlib.EVT_GRID_CELL_LEFT_CLICK, self.OnCellLeftClick)
        self.Bind(gridlib.EVT_GRID_CELL_RIGHT_CLICK, self.OnCellRightClick)
        self.Bind(gridlib.EVT_GRID_CELL_LEFT_DCLICK, self.OnCellLeftDClick)
        self.Bind(gridlib.EVT_GRID_CELL_RIGHT_DCLICK, self.OnCellRightDClick)

        self.Bind(gridlib.EVT_GRID_LABEL_LEFT_CLICK, self.OnLabelLeftClick)
        self.Bind(gridlib.EVT_GRID_LABEL_RIGHT_CLICK, self.OnLabelRightClick)
        self.Bind(gridlib.EVT_GRID_LABEL_LEFT_DCLICK, self.OnLabelLeftDClick)
        self.Bind(gridlib.EVT_GRID_LABEL_RIGHT_DCLICK, self.OnLabelRightDClick)

        self.Bind(gridlib.EVT_GRID_ROW_SIZE, self.OnRowSize)
        self.Bind(gridlib.EVT_GRID_COL_SIZE, self.OnColSize)

        self.Bind(gridlib.EVT_GRID_RANGE_SELECT, self.OnRangeSelect)
        self.Bind(gridlib.EVT_GRID_CELL_CHANGE, self.OnCellChange)
        self.Bind(gridlib.EVT_GRID_SELECT_CELL, self.OnSelectCell)

        self.Bind(gridlib.EVT_GRID_EDITOR_SHOWN, self.OnEditorShown)
        self.Bind(gridlib.EVT_GRID_EDITOR_HIDDEN, self.OnEditorHidden)
        self.Bind(gridlib.EVT_GRID_EDITOR_CREATED, self.OnEditorCreated)

    def OnCellLeftClick(self, evt):
        print "OnCellLeftClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                 evt.GetCol(),
                                                 evt.GetPosition())
        evt.Skip()

    def OnCellRightClick(self, evt):
        print "OnCellRightClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                  evt.GetCol(),
                                                  evt.GetPosition())
        evt.Skip()

    def OnCellLeftDClick(self, evt):
        print "OnCellLeftDClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                  evt.GetCol(),
                                                  evt.GetPosition())
        evt.Skip()

    def OnCellRightDClick(self, evt):
        print "OnCellRightDClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                   evt.GetCol(),
                                                   evt.GetPosition())
        evt.Skip()

    def OnLabelLeftClick(self, evt):
        print "OnLabelLeftClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                  evt.GetCol(),
```

```
                                                evt.GetPosition())
        evt.Skip()

    def OnLabelRightClick(self, evt):
        print "OnLabelRightClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                   evt.GetCol(),
                                                   evt.GetPosition())
        evt.Skip()

    def OnLabelLeftDClick(self, evt):
        print "OnLabelLeftDClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                   evt.GetCol(),
                                                   evt.GetPosition())
        evt.Skip()

    def OnLabelRightDClick(self, evt):
        print "OnLabelRightDClick: (%d,%d) %s\n" % (evt.GetRow(),
                                                    evt.GetCol(),
                                                    evt.GetPosition())
        evt.Skip()

    def OnRowSize(self, evt):
        print "OnRowSize: row %d, %s\n" % (evt.GetRowOrCol(),
                                           evt.GetPosition())
        evt.Skip()

    def OnColSize(self, evt):
        print "OnColSize: col %d, %s\n" % (evt.GetRowOrCol(),
                                           evt.GetPosition())
        evt.Skip()

    def OnRangeSelect(self, evt):
        if evt.Selecting():
            msg = 'Selected'
        else:
            msg = 'Deselected'
        print "OnRangeSelect: %s  top-left %s, bottom-right %s\n" % (msg,
evt.GetTopLeftCoords(),

evt.GetBottomRightCoords())
        evt.Skip()



    def OnCellChange(self, evt):
        print "OnCellChange: (%d,%d) %s\n" % (evt.GetRow(), evt.GetCol(),
evt.GetPosition())

        # Show how to stay in a cell that has bad data.  We can't just
        # call SetGridCursor here since we are nested inside one so it
        # won't have any effect.  Instead, set coordinates to move to in
        # idle time.
        value = self.GetCellValue(evt.GetRow(), evt.GetCol())

        if value == 'no good':
```

```
            self.moveTo = evt.GetRow(), evt.GetCol()

    def OnSelectCell(self, evt):
        if evt.Selecting():
            msg = 'Selected'
        else:
            msg = 'Deselected'
        print "OnSelectCell: %s (%d,%d) %s\n" % (msg, evt.GetRow(),
                                                 evt.GetCol(),
evt.GetPosition())

        # Another way to stay in a cell that has a bad value...
        row = self.GetGridCursorRow()
        col = self.GetGridCursorCol()

        if self.IsCellEditControlEnabled():
            self.HideCellEditControl()
            self.DisableCellEditControl()

        value = self.GetCellValue(row, col)

        if value == 'no good 2':
            return  # cancels the cell selection

        evt.Skip()


    def OnEditorShown(self, evt):
        if evt.GetRow() == 6 and evt.GetCol() == 3 and \
           wx.MessageBox("Are you sure you wish to edit this cell?",
                         "Checking", wx.YES_NO) == wx.NO:
            evt.Veto()
            return

        print "OnEditorShown: (%d,%d) %s\n" % (evt.GetRow(), evt.GetCol(),
                                               evt.GetPosition())
        evt.Skip()


    def OnEditorHidden(self, evt):
        if evt.GetRow() == 6 and evt.GetCol() == 3 and \
           wx.MessageBox("Are you sure you wish to  finish editing this cell?",
                         "Checking", wx.YES_NO) == wx.NO:
            evt.Veto()
            return

        print "OnEditorHidden: (%d,%d) %s\n" % (evt.GetRow(),
                                                evt.GetCol(),
                                                evt.GetPosition())
        evt.Skip()


    def OnEditorCreated(self, evt):
        print "OnEditorCreated: (%d, %d) %s\n" % (evt.GetRow(),
```

```
                                                evt.GetCol(),
                                                evt.GetControl())

#----------------------------------------------------------------------
class MyForm(wx.Frame):
    """"""

    #----------------------------------------------------------------------
    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, parent=None, title="An Eventful Grid")
        panel = wx.Panel(self)

        myGrid = MyGrid(panel)

        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(myGrid, 1, wx.EXPAND)
        panel.SetSizer(sizer)

if __name__ == "__main__":
    app = wx.PySimpleApp()
    frame = MyForm().Show()
    app.MainLoop()
```

That's a lot of code, but it's mostly just event handlers. We'll only go over a couple of them. You're intelligent, so I'm sure you'll be able to figure out the rest! Notice that all the events in this sample are prepended with "gridlib". That means that these events only apply to Grid instances and will have no effect on any other widget. For our first event example, we'll look at EVT_GRID_CELL_LEFT_CLICK. When you bind to this event, it allows you to catch the event that is fired when the user left clicks their mouse pointer on a cell in your grid. As far as I can tell, this event gets fired before the select event (i.e. EVT_GRID_SELECT_CELL), so you could use it to veto a cell selection if you wanted to. Note that there is a right click and corresponding double-click events for both mouse buttons as well.

The EVT_GRID_LABEL_LEFT_CLICK event allows us to know when a user clicks on a grid column or row label. As with the previous event example, you can also catch double-clicking and right clicking. Note that there is no middle click event though. You would probably need to use the normal middle-click mouse events for this.

EVT_GRID_EDITOR_SHOWN is fired when you start editing a cell. You could use this event to create a custom editor to be displayed or veto the editing. See if you can figure out what the other events do on your own.

Now let's look at some of the handlers. A lot of them have the following code in them:

```
(evt.GetRow(),
 evt.GetCol(),
 evt.GetPosition())
```

This can be quite handy for troubleshooting as it tells us if we're in the cell we expect to be in and it also gives us pixel coordinates of where we clicked. The latter can be handy for contextual pop-up menus or tooltip positioning. The other interesting handler is *OnRangeSelect* which shows us how to tell what our selected range is. Notice that all we needed to do was call GetTopLeftCoords and GetBottomRightCoords to figure it out. Isn't this fun?

Now you know the basics for grid event handling. Go out there and build some cool applications with your new-found knowledge! Keep your eyes peeled for the next couple grid tutorials where you'll learn how to use virtual tables, change grid and row labels, add tooltips to columns and other fun stuff!

Note: The code in this post was tested on the following:

- Windows XP Professional, wxPython 2.8.10.1 (unicode), Python 2.5
- Windows 7 Professional, wxPython 2.8.10.1 (unicode), Python 2.6.4

Further Reading

- Grid Manual
- wx.Grid Wiki Page
- Official Grid Documentation