Step 4 : use the label tag to set extra properties

Each label can have a different purpose: a header, a footer, a table, a resultset, a database field, a text, an image and so on. To set and change specific properties of a label the tag property of the label is used.

Define a Type variable named "**tagdata**" to hold the extra properties.

Changes in the Main page:

- Add the Type variable **tagdata** in the Process_Globals subroutine:

```
Type tagdata (labeltype As String, row As Int, col As Int, numrows As Int, numcols As Int, _
                imgdir As String, imgfile As String, _
                dbfile As String, dbpath As String, dbname As String, _
                tblname As String, fldname As String, strsql As String, text As String)
```

Each label that is placed on the panemain pane will have its tag property set to the tagdata variable and its contents.

Changes in the **B4XMainPage** page:

- Create a subroutine for the default label properties. These properties are almost the same as you have in the btnnewlbl_Click subroutine. Did you notice the addition of a TooltipText property?

```
Private Sub set_default_label_properties(text As String, bordercolor As Paint,
backgroundcolor As Paint) As Label
    Dim lbl As Label
    lbl.Initialize("lbl")
    lbl.Text = text
    lbl.TextSize = 15
    lbl.WrapText = True
    lbl.Alignment = "TOP_LEFT"
    lbl.TooltipText = text
    lbl.Tag = lbl.Text
    CSSUtils.SetStyleProperty(lbl,"-fx-padding","3 3 3 3")
    CSSUtils.SetBorder(lbl,1,bordercolor,5)
    CSSUtils.SetBackgroundColor(lbl,backgroundcolor)
    Return lbl
End Sub
```

- You can now replace the lines in the btnnewlbl_Click subroutine with this line:
  `Dim lbl As Label = set_default_label_properties(talabeltext.text, fx.Colors.Green, fx.Colors.ARGB(255,255,255,187))`
- Below this line (in the btnnewlbl_Click subroutine) you can add code to fill the tagdata and set the label tag to this tagdata. Then add the label to the panemain pane.

```
Dim td As tagdata
td.Initialize
td.labeltype = "TXT"
td.row = lbl.Top/gridsize
td.col = lbl.Left/gridsize
```

```
td.text = lbl.Text
td.numrows = lblrows.text
td.numcols = lblcols.Text + 2
lbl.Tag = td
lbl.TooltipText = lbl.Text
panemain.AddNode(lbl,0,0,lbl.prefWidth,lbl.prefHeight)
```

- The labels **lblrows** and **lblcols** are added to the layout below the talabeltext textarea.

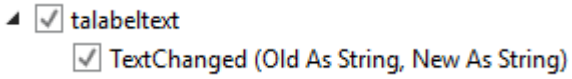
width: 70 height: 30

Note: adjust the call to setup_dialog: change 300 to 340!

Generate the members lblrows and lblcols and set the variables to public.

Make sure to also check the TextChanged method of the talabeltext textarea!

The labels texts will be adjusted in this subroutine.



- Fill the TextChanged subroutine. An estimate of 3 characters per column (20 pixels) is used. The number of lines is determined by the number of CRLF's in the textarea.

```
Private Sub talabeltext_TextChanged (Old As String, New As String)
    If New <> "" Then
        Dim lines As List = Regex.Split(CRLF,New)
        cols = 1
        lblcols.Text = cols
        For i = 0 To lines.Size -1
            Dim strlen As String = lines.Get(i)
            cols = Round(strlen.Length / 3)
            If cols > lblcols.Text Then lblcols.Text = cols
        Next
        lblrows.Text = lines.size
    End If
End Sub
```

- The lblrows and lblcols variables are initialized before the call to show the dialog (labeldialog.ShowCustom).

```
lblrows.Text = 1
lblcols.Text = 1
```

You can run the application now and test adding a label. The rows and cols labels should change when you type text in the textarea. A tooltip text should appear when you hover over a label in the grid.

- Add some code to show the tagdata in the tooltip. For this you can write the following subroutine:

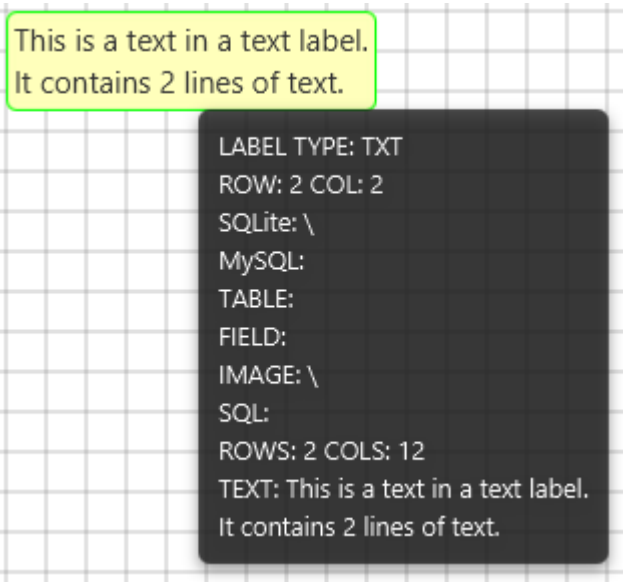
```
Private Sub set_tooltip_text(tds As tagdata) As String
    Dim sb As StringBuilder
    sb.Initialize
```

```
        sb.Append("LABEL TYPE: ").Append(tds.labeltype).Append(CRLF)
        sb.Append("ROW: ").Append(tds.row).Append(" COL:
").Append(tds.col).Append(CRLF)
        sb.Append("SQLite: ").Append(tds.dbpath & "\" & tds.dbfile).Append(CRLF)
        sb.Append("MySQL: ").Append(tds.dbname).Append(CRLF)
        sb.Append("TABLE: ").Append(tds.tblname).Append(CRLF)
        sb.Append("FIELD: ").Append(tds.fldname).Append(CRLF)
        sb.Append("IMAGE: ").Append(tds.imgdir & "\" & tds.imgfile).Append(CRLF)
        sb.Append("SQL: ").Append(tds.strsql).Append(CRLF)
        sb.Append("ROWS: ").Append(tds.numrows).Append(" COLS:
").Append(tds.numcols).Append(CRLF)
        sb.Append("TEXT: ").Append(tds.text).Append(CRLF)
        Return sb.ToString
End Sub
```

- For this new tooltip you need to add some code to the lbl_MouseEntered subroutine. Retrieve the contents of the view.tag property, call the set_tooltip_text subroutine and set the TooltipText property to the resulting string.

```
Private Sub lbl_MouseEntered (EventData As MouseEvent)
        Dim view As Label = Sender
        If view.IsInitialized Then
                Dim tdvw As tagdata = view.Tag
                Dim strtooltip As String = set_tooltip_text(tdvw)
                view.TooltipText = strtooltip
                set_tooltip_style(view)
        End If
        mousedrag = False
End Sub
```

- You can also change the style of the tooltip. By default it looks like a black label with white text.



- Create a CSS file and name it “tooltipstyle.css”. Add the file to the Files Manager in the IDE. Here’s an example of the contents of that file:

```
{

    -fx-show-delay: 50ms;
```

```
    -fx-show-duration: 5000ms;
    -fx-hide-delay: 50ms;
    -fx-background-radius: 5 5 5 5;
    -fx-background-color: blue;
    -fx-text-fill: white;
    -fx-font-size: 15px;
    -fx-font-family: 'Verdana';
    -fx-font-weight: bold;
}
```

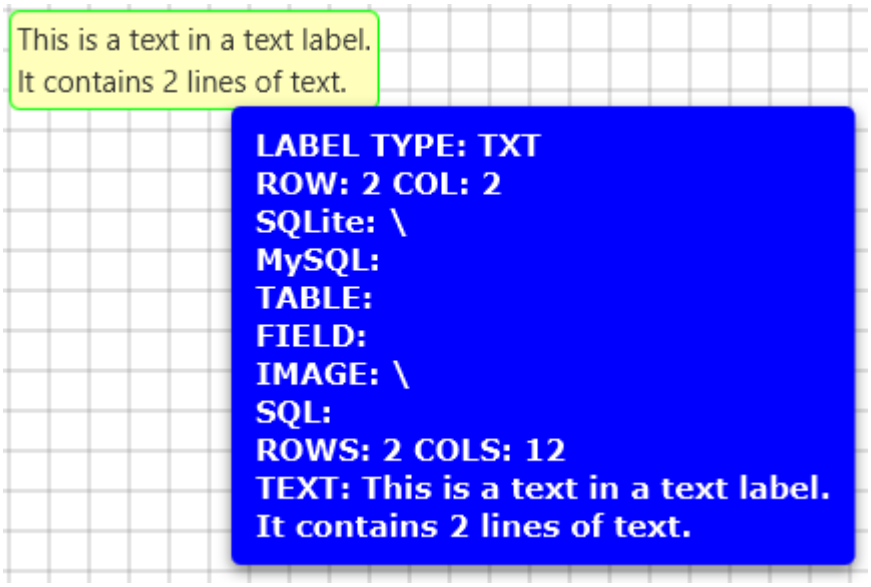
- To use the new style you have to add a line of code in the B4XPage_Created subroutine and add a public variable tooltipstyle as a string in the Class_Globals subroutine.

```
tooltipstyle = File.ReadString(File.DirAssets,"tooltipstyle.css")
```

- The link between the stylesheet file and the tooltipstyle variable can be set using a javaobject in the new set_tooltip_style subroutine.

```
Private Sub set_tooltip_style(vw As B4XView)
    Dim jo As JavaObject = vw
    Dim tooltip As JavaObject = jo.RunMethodJO("getTooltip",Null)
    tooltip.RunMethod("setStyle",Array As Object(tooltipstyle))
End Sub
```

- The tooltip should now look like this:



- Now you can write a subroutine to change the text of the text label. The setup_dialog subroutine is used to show the label dialog. The tagdata of the current node (label) is retrieved from the tag property. The lblrows and lblcols label texts in the dialog are set to the info from the tagdata. The set_default_label_properties subroutine is reused to create a new label with the properties of the current node. The tagdata and label properties are adjusted to the new text. The currnode is removed from the panemain pane and the new label is added.

```
Private Sub change_label_text
```

```
	Dim title As String = " Change label text"
	Dim title_color As Paint = fx.Colors.ARGB(255,186,255,158)
	Dim pnl1 As Pane = setup_dialog("labeldialog_layout",title,title_color,340)
	talabeltext.Text = currnode.text
	Dim tdo As tagdata = currnode.Tag
	lblrows.text = tdo.numrows
	lblcols.Text = tdo.numcols
	Dim rsub1 As ResumableSub =  labeldialog.ShowCustom(pnl1, "OK", "",
"CANCEL")
	Wait For (rsub1) Complete (Result As Int)
	If Result = xui.dialogResponse_Positive Then
		Dim lbl As Label =
set_default_label_properties(talabeltext.text,fx.Colors.Green,fx.Colors.ARGB(255,255,255,
187))
		lbl.Tag = currnode.Tag
		lbl.Left = currnode.Left
		lbl.Top = currnode.Top
		lbl.PrefWidth = currnode.Width
		lbl.PrefHeight = currnode.Height
		Dim td As tagdata = lbl.tag
		td.row = lbl.Top/gridsize
		td.col = lbl.Left/gridsize
		td.text = lbl.Text
		td.numrows = lblrows.text
		td.numcols = lblcols.Text + 2
		lbl.Tag = td
		lbl.TooltipText = lbl.Text
		currnode.RemoveViewFromParent
		panemain.AddNode(lbl,lbl.Left,lbl.Top,lbl.prefWidth,lbl.prefHeight)
	End If
End Sub
```

- The change_label_text is called in the lbl_MouseClicked subroutine. If the label is not dragged and the labeltype is “TXT” then the change_label_text subroutine is called.

```
Private Sub lbl_MouseClicked(EventData As MouseEvent)
	Dim view As B4XView = Sender
	currnode = view
	If mousedrag = False Then
		Dim tdl As tagdata = currnode.Tag
		If tdl.labeltype = "TXT" Then
			change_label_text
		End If
	End If
	EventData.Consume
End Sub
```

- When a label is moved the tagdata need to be adjusted to indicate the position of the label on the grid. This is done in the lbl_MouseReleased subroutine.

```
Private Sub lbl_MouseReleased (EventData As MouseEvent)
	Dim view As B4XView = Sender
```

```
    snap_to_grid(view)
    Dim td As tagdata = view.Tag
    td.row = view.Top/gridsize
    td.col = view.Left/gridsize
    view.Tag = td
    EventData.Consume
End Sub
```

- Let's add an image to the grid. The image will be the background of a label. Add a button next to the new label button in the layout for this:

width: 80 height: 30 text: Image

You can also adjust the width from the btnnewlbl to 80 because there will be more views added to the pndesign pane.

- The code in the B4XMainPage page looks like this:

```
Private Sub btnnewimage_Click
    fchoose.SetExtensionFilter("Image files",Array As String("*.jpg","*.png"))
    Dim filename As String =
fchoose.ShowOpen(B4XPages.GetNativeParent(B4XPages.MainPage))
    If filename <> "" Then
        Dim fpath As String = filename.SubString2(0,filename.LastIndexOf("\"))
        Dim fname As String = filename.SubString(filename.LastIndexOf("\")+1)
        Dim lbl As Label
        lbl.Initialize("lbl")
        lbl.Text = ""
        lbl.Tag = lbl.Text
        lbl.prefWidth = 100
        lbl.prefHeight = 100
        CSSUtils.SetBorder(lbl,1,fx.Colors.Blue,5)
        CSSUtils.SetBackgroundImage(lbl,fpath,fname)
        Dim td As tagdata
        td.Initialize
        td.row = lbl.Top/gridsize
        td.col = lbl.Left/gridsize
        td.numrows = 100 / 20
        td.numcols = 100 / 20
        td.imgdir = fpath
        td.imgfile = fname
        td.labeltype = "IMG"
        td.text = ""
        lbl.Tag = td
        panemain.AddNode(lbl,0,0,lbl.prefWidth,lbl.prefHeight)
    End If
End Sub
```

- The image is resized to 100 by 100 pixels (5 rows and 5 columns) for now.

That's it for step 4. In step 5 you can read about adding special tags.