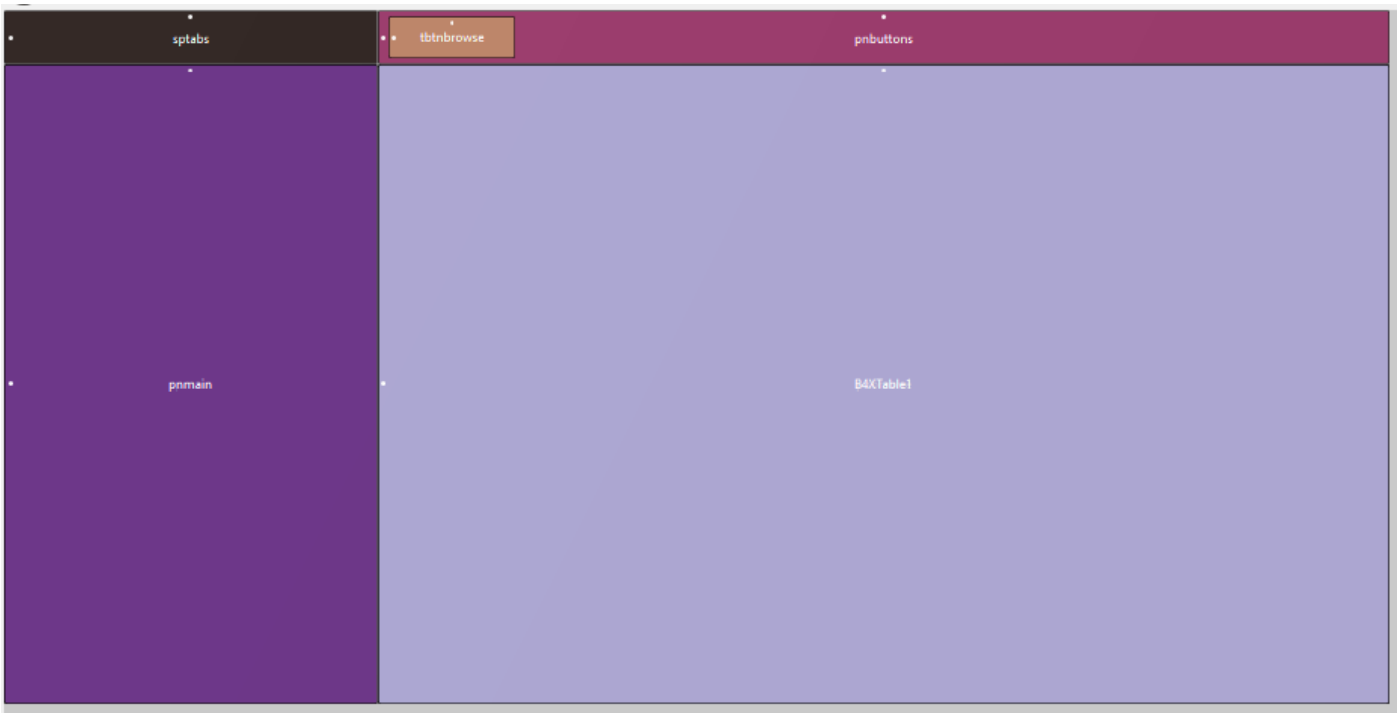
Report writer steps.

In this tutorial you can learn how to create a simple report writer application step-by-step.

Step 1 : transform the DBField_selector application into a database browser

Changes to the MainPage layout:

- Variants change: add a new variant 1300 by 650 and remove the old one (350 by 650).
- Remove the anchors from the sptabs, spmain and pnmain views and set their width to 350.
- Add a pane next to the sptabs view, name it **pnbuttons** and set the width at 950 and the height at 50.
- Add a togglebutton into the pnbuttons pane, name it **tbtnbrowse** and set the width at 120 and the height at 40. Set the text to “Browse data”.
- Add a pane under the pnbuttons pane, name it **pnbrowse** and fill the rest of the layout (950 by 600).
- Add a B4Xtable into the pnbrowse pane, name it **B4Xtable1** and give it the same size as the pnbrowse. Don’t forget to add the B4Xtable library to the Library manager panel in the IDE.
- Generate the pnbuttons, pnbrowse, tbtnbrowse and B4Xtable1 members. Select the SelectedChange event from the tbtnbrowse togglebutton.



Changes in the Main page:

- In the region Project Attributes set the #MainFormWidth to 1300.

```
#Region Project Attributes
    #MainFormWidth: 1300
```

```
    #MainFormHeight: 650
    #AdditionalJar: sqlite-jdbc-3.7.2
    #AdditionalJar: mysql-connector-java-5.1.27-bin.jar
#End Region
```

Changes in the B4XMainPage page:

- Make the pnbuttons, pnbrowse, tbtnbrowse and B4Xtable1 variables Public (this will remove the warnings)

```
Public pnbrowse As Pane
Public pnbuttons As Pane
Public tbtnbrowse As ToggleButton
Public B4XTable1 As B4XTable
```

- Add code to the tbtnbrowse_SelectedChange subroutine:
  - A try catch structure to get the errors logged.
  - An if test for the Selected state of the togglebutton.
  - An if test for the current tab in use: currtab = 0 for SQLite or currtab = 1 for MySQL
  - An if test on the contents of the tasqlitefile.text: if empty give a message and return
  - An if test on the contents of the tamysqldb.text: if empty give a message and return
  - An if test on the current tab = 0 to call the fill_B4XTable1_from_sqlite subroutine
  - An if test on the current tab = 1 to call the fill_B4XTable1_from_mysql subroutine

```
Private Sub tbtnbrowse_SelectedChange(Selected As Boolean)
    Try
        If Selected Then
            If currtab = 0 Then
                If tasqlitefile.Text = "" Then
                  xui.MsgboxAsync("Open a SQLite database file first.","Browse data")
                  Return
                End If
            End If
            If currtab = 1 Then
                If tamysqldb.text = "" Then
                  xui.MsgboxAsync("Connect to the MySQL server first.","Browse data")
                  Return
                End If
            End If
            If currtab = 0 Then
                fill_B4XTable1_from_sqlite(tapath.text, tasqlitefile.text,tatable.text)
            End If
            If currtab = 1 Then
                fill_B4XTable1_from_mysql(tamysqldb.text, tamysqltable.text)
            End If
        End If
    Catch
        Log(LastException)
    End Try
End Sub
```

- Add the private subs fill_B4XTable1_from_sqlite and fill_B4XTable1_from_mysql.
- Add code to the fill_B4XTable1_from_sqlite subroutine:
    - Provide the arguments for the subroutine:
      (fpath As String, fname As String,tname As String).
    - Clear the B4Xtable1 variable.
    - Loop over the fields list (clvfields0) and add a column for each field. The field types are text.
    - Initialize the data list (data) and the fieldtypes list (fldtypes).
    - Run a dbsqlite query to get the fieldtypes from the table and add the type to the fieldtypes list.
    - Run a dbsqlite query to get all the records from the table.
    - Loop over the resultset, initialize a record list (rec), loop over the fields (clvfields0) and add the content to the record list if the fieldtype is not a BLOB, is not a GRAPHIC, is not Null. Otherwise add the fieldtype text as the content to the record list.
    - Add the record list to the data list.
    - After the loop close the resultset.
    - Use the B4Xtable1 method SetData(data) to add the data list to the B4Xtable1.

```
Private Sub fill_B4XTable1_from_sqlite(fpath As String, fname As String,tname As String)
	B4XTable1.Clear
	For i = 0 To clvfields0.Size -1
		B4XTable1.AddColumn(clvfields0.GetValue(i),B4XTable1.COLUMN_TYPE_TEXT)
	Next
	Private data As List
	data.Initialize
	Dim fldtypes As List
	fldtypes.Initialize
	Private rs As ResultSet = dbsqlite.ExecQuery("PRAGMA table_info(" & tname & ")")
	Do While rs.NextRow
		Dim fld As String = rs.GetString("type").Trim
		fldtypes.Add(fld.Trim)
	Loop
	rs.Close
	Private rs1 As ResultSet = dbsqlite.ExecQuery("SELECT * FROM " & tname)
	Do While rs1.NextRow
		Dim rec As List
		rec.Initialize
		For i = 0 To clvfields0.Size -1
			If fldtypes.Get(i) <> "BLOB" And fldtypes.Get(i) <> "GRAPHIC" Then
				If rs1.GetString2(i) <> Null Then
					rec.Add(rs1.GetString2(i))
				Else
					rec.Add("NULL")
				End If
			Else
				rec.Add(fldtypes.Get(i))
			End If
		Next
		data.Add(rec)
	Loop
```

```
        rs1.Close
        B4XTable1.SetData(data)
End Sub
```

You could run the application at this point to test it with a SQLite database. Open a SQLite database and click on the tbtnbrowse togglebutton ("Browse data").

- Add code to the fill_B4XTable1_from_mysql subroutine:
  - Provide the arguments for the subroutine:
    (dbname As String,tname As String)
  - Clear the B4Xtable1 variable.
  - Loop over the fields list (clvmysqlfields) and add a column for each field. The field types are text.
  - Initialize the data list (data) and the fieldtypes list (fldtypes).
  - Run the dbmysql query to use the database.
  - Run a dbmysql query to get the fieldtypes from the table and add the type to the fieldtypes list.
  - Run a dbmysql query to get all the records from the table.
  - Loop over the resultset, initialize a record list (rec), loop over the fields (clvmysqlfields) and add the content to the record list if the fieldtype is not a date, is not Null. Otherwise if the fieldtype is a blob then set the content to "BLOB" and if it's not a blob then set the content to "NULL".
    If the type is a date and the content is empty then MySQL can convert that field to "null". You have to change the initialization string from the dbmysql object in the get_schemas subroutine to the following:

    ```
    dbmysql.Initialize2("com.mysql.jdbc.Driver",
    "jdbc:mysql://localhost?characterEncoding=utf8&zeroDateTimeBehavior=convertToNull","root","root")
    ```

    The first type = "date" test automatically turns the content to "null" and since the content is not a blob in the last test then the content will be "NULL".
  - Add the record list to the data list.
  - After the loop close the resultset.
  - Use the B4Xtable1 method SetData(data) to add the data list to the B4Xtable1.

```
Private Sub fill_B4XTable1_from_mysql(dbname As String,tname As String)
        B4XTable1.Clear
        For i = 0 To clvmysqlfields.Size -1
                B4XTable1.AddColumn(clvmysqlfields.GetValue(i),B4XTable1.COLUMN_TYPE_TEXT)
        Next
        Private data As List
        data.Initialize
        dbmysql.ExecQuery("USE " & dbname)
        Dim fldtypes As List
        fldtypes.Initialize
        Dim rstypes As ResultSet = dbmysql.ExecQuery2("SELECT column_type FROM
INFORMATION_SCHEMA.COLUMNS WHERE table_name = ? AND TABLE_SCHEMA = ?",Array As
String(tname,dbname))
```

```
        Do While rstypes.NextRow
                fldtypes.Add(rstypes.GetString2(0))
        Loop
        Dim rs As ResultSet = dbmysql.ExecQuery("SELECT * FROM " & tname)
        Do While rs.NextRow
                Dim rec As List
                rec.Initialize
                For i = 0 To clvmysqlfields.Size -1
                        If fldtypes.Get(i) = "date" Then
                                Dim fldvalue As String = rs.GetString2(i)
                        End If
                        If fldtypes.Get(i) <> "date" Then
                                Dim fldvalue As String = rs.GetString2(i)
                        End If
                        If fldvalue <> Null Then
                                rec.Add(fldvalue)
                        Else
                                If fldtypes.Get(i) = "blob" Then
                                        rec.Add("BLOB")
                                Else
                                        rec.Add("NULL")
                                End If
                        End If
                Next
                data.Add(rec)
        Loop
        rs.Close
        B4XTable1.SetData(data)
End Sub
```

Now you can run the application again.

When you use a MySQL database you first need to connect to the database server (localhost). Select the database and table and click on the tbtnbrowse togglebutton ("Browse data").

The tbtnbrowse is a togglebutton so you can show or hide the pnbrowse pane. You need to change the code in the tbtnbrowse_SelectedChange subroutine. Before the second test for the currtab = 0 you can add:

```
                pnbrowse.Visible = True
```

Before the last End If you can add an else statement and in it hide the pnbrowse pane:

```
        Else
                pnbrowse.Visible = False
```

Run the application again and now you can show or hide the pnbrowse pane and the B4Xtable1 that's in it.

When you hide the browse pane you can use the space in the layout for other things…

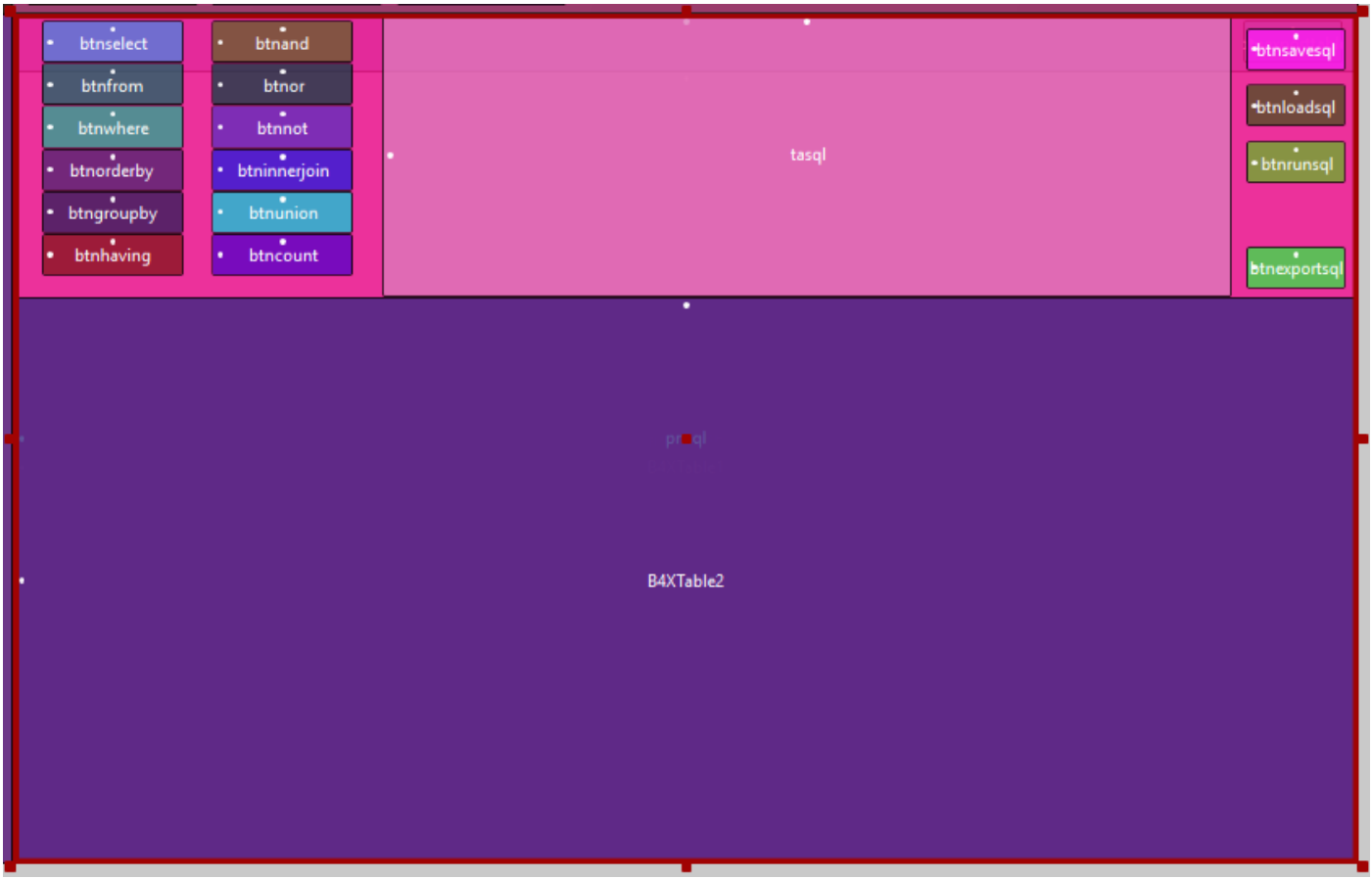
More about that in the next step.

Step 2 : add a SQL editor pane with buttons and a textarea

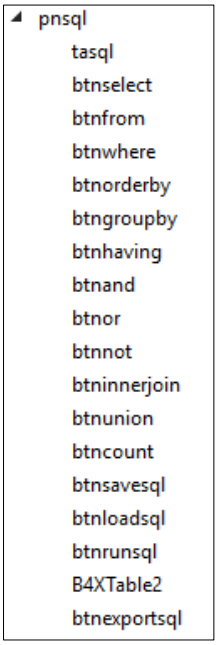
Changes to the MainPage layout:

- Add a togglebutton into the pnbuttons pane, name it **tbtnsql** and set the text to “SQL” Generate the tbtnsql togglebutton member and select the SelectedChange event.

- Add a pane under the pnbuttons pane, name it **pnsql** and set the width to 950 and the height to 600.



- Add the buttons, the textarea and the B4Xtable2 into the pnsql pane.
- Set the text of each button left of the textarea to the keyword(s) from it’s name and a space: **btnselect** text = SELECT , **btnfrom** text = FROM , **btnwhere** text = WHERE , **btnorderby** text = ORDER BY , **btngroupby** text = GROUP BY , **btnhaving** text = HAVING , **btnand** text = AND , **btnor** text = OR , **btnnot** text = NOT , **btninnerjoin** text = INNER JOIN , **btnunion** text = UNION , **btncount** text = COUNT(.
- Set the text of the buttons right of the textarea: **btnsavesql** text = Save, **btnloadsql** text = Load, **btnrunsql** text = Run and **btnexportsql** text = Export
- Set the textarea **tasql** width to 600 and the height to 200.
- Set the **B4Xtable2** width to 950 and the height to 400.
- Generate the members from the pnsql, select the click event for the buttons, select the textarea **tasql** and the B4Xtable2.



In the Main page you might want to set the mainform resizable to false below the mainform.show statement:

```
MainForm.Resizable = False
```

Changes in the B4XMainPage page:

- Make all the generated variables public
- To make the textarea (tasql) a drag-and-drop target you have to add some code in the B4XPage_Created subroutine below the first drag-and-drop initialization:

```
DragAndDrop2.Initialize(Me)
DragAndDrop2.MakeDragTarget(tasql,"sqlTarget")
```

- Create a region SQL to collect all the pnsql related subroutines:

```
#Region SQL
```

- Add a sqlTarget_DragOver subroutine to set the transfer mode and add a sqlTarget_DragDropped subroutine to get the string from the Dragboard into the textarea tasql.
    - Remove the brackets ({}, []) from the string
    - if the tasql text is not empty and the text doesn't contains the word "FROM" then you can remove the trailing comma from the text that contains the select fields string just before you click on the btnfrom button.

```
Sub sqlTarget_DragOver(e As DragEvent)
	If e.GetDragboard.HasString Then e.AcceptTransferModes(TransferMode.MOVE)
End Sub
Sub sqlTarget_DragDropped(e As DragEvent)
	Dim Db As Dragboard = e.GetDragboard
	If Db.HasString Then
		Log(Db.GetString)
		Dim strsrc As String = Db.GetString
		strsrc = strsrc.Replace("{","")
		strsrc = strsrc.Replace("}","")
		strsrc = strsrc.Replace("[","")
		strsrc = strsrc.Replace("]","")
		If tasql.Text <> "" And tasql.Text.Contains("FROM") = False Then
			tasql.Text = tasql.Text & strsrc & ","
		Else
			tasql.Text = tasql.Text & strsrc
		End If
		e.SetDropCompleted(True)
		Return
	End If
End Sub
```

- Add code to the tbtnsql_SelectedChange subroutine:
    - If the button is in the selected state then clear the B4Xtable2 B4Xtable and make the pnsql pane visible.
    - You can then select the current tab and fill in the **lbldbinfo** text with the database and table info. This label was added in the designer into the pnbuttons on the

right side. Set the lbldbinfo width to 540 and the height to 30.


```
Private Sub tbtnsql_SelectedChange(Selected As Boolean)
	If Selected Then
		B4XTable2.Clear
		pnsql.Visible = True
		Select currtab
			Case 0
				lbldbinfo.Text = tapath.Text & "\" & tasqlitefile.text & " : " & tatable.text
			Case 1
				lbldbinfo.Text = tamysqldb.text & " : " & tamysqltable.text
		End Select
	Else
		pnsql.Visible = False
	End If
End Sub
```

- Add code to the button click events to compose a sql statement. The statement is divided into several lines: the SELECT line, the FROM line, the WHERE line and so on.
  - The buttons on the left of the textarea:
  - The SELECT button code clears the B4Xtable2 first and starts a new sql statement
  - The FROM button code strips off the last comma from the select line and adds the button text after adding a CRLF
  - The rest of the buttons code add the button text to the textarea after adding a CRLF

```
Private Sub btnselect_Click
	B4XTable2.Clear
	tasql.Text = btnselect.Text
End Sub
Private Sub btnfrom_Click
	If tasql.Text.EndsWith(",") Then tasql.Text = tasql.Text.SubString2(0,tasql.Text.Length-1)
	tasql.Text = tasql.Text & CRLF & btnfrom.Text
End Sub
Private Sub btnwhere_Click
	tasql.Text = tasql.Text & CRLF & btnwhere.Text
End Sub
Private Sub btnorderby_Click
	tasql.Text = tasql.Text & CRLF & btnorderby.Text
End Sub
Private Sub btngroupby_Click
	tasql.Text = tasql.Text & CRLF & btngroupby.Text
End Sub
Private Sub btnhaving_Click
	tasql.Text = tasql.Text & CRLF & btnhaving.Text
End Sub
Private Sub btnand_Click
	tasql.Text = tasql.Text & CRLF & btnand.Text
End Sub
Private Sub btnor_Click
```

```
    tasql.Text = tasql.Text & CRLF & btnor.Text
End Sub
Private Sub btnnot_Click
    tasql.Text = tasql.Text & btnnot.Text
End Sub
Private Sub btninnerjoin_Click
    tasql.Text = tasql.Text & CRLF & btninnerjoin.Text
End Sub
Private Sub btnunion_Click
    tasql.Text = tasql.Text & CRLF & btnunion.Text
End Sub
Private Sub btncount_Click
    tasql.Text = tasql.Text & btncount.Text
End Sub
```

- Add code to the btnsavesql_Click subroutine:
  - If the textarea tasql only contains the “SELECT “ string then there is no statement to save
  - Using the current tab selection you can set the initial file name to start with “SQLite_” or “MySQL_”
  - For the initial directory you can set the xui default folder in the B4XPage_Created subroutine to xui.SetDataFolder("report_writer_steps").
  - Set the extension filter to “*.sql”: the default file extension for SQL statements.
  - Show the dialog and if the file name is not empty then write the tasql text to the file
  - Show a message where the file is saved.

```
Private Sub btnsavesql_Click
    If tasql.Text.Length < 8 Then Return
    Select currtab
        Case 0
            Dim dbtype As String = "SQLite"
            fchoose.InitialFileName = dbtype & "_"
        Case 1
            Dim dbtype As String = "MySQL"
            fchoose.InitialFileName = dbtype & "_"
    End Select
    fchoose.InitialDirectory = xui.DefaultFolder
    fchoose.setExtensionFilter("SQL", Array As String("*.sql"))
    Dim fname As String = fchoose.ShowSave(B4XPages.GetNativeParent(B4XPages.MainPage))
    If fname = "" Then Return
    Dim path As String = fname.SubString2(0,fname.LastIndexOf("\")+1)
    Dim filename As String = fname.SubString(fname.LastIndexOf("\")+1)
    File.WriteString(path,filename,tasql.text)
    xui.MsgboxAsync("File " & fname & " saved.","Save SQL")
End Sub
```

- Add code to the btnloadsql_Click subroutine:
  - Set the initial directory to the default folder
  - Set the extension filter to “*.sql”
  - If the file name is not empty then read the string from the file and set the tasql text to the content of the file

```
Private Sub btnloadsql_Click

    fchoose.InitialDirectory = xui.DefaultFolder
```

```
    fchoose.setExtensionFilter("SQL", Array As String("*.sql"))
    Dim fname As String = fchoose.ShowOpen(B4XPages.GetNativeParent(B4XPages.MainPage))
    If fname = "" Then Return
    Dim path As String = fname.SubString2(0,fname.LastIndexOf("\")+1)
    Dim filename As String = fname.SubString(fname.LastIndexOf("\")+1)
    tasql.text = File.ReadString(path,filename)
End Sub
```

- Add code to the btnrunsql_Click subroutine:
  - If the textarea is empty then return from the subroutine
  - Clear the B4Xtable2
  - Get the sql statement lines and get the fields list from the SELECT line
  - Set the B4Xtable2 columns using the fields list
  - Initialize the data list
  - Select the current tab, execute the query, loop over the resultset and set the data list
  - Use the setdata method from the B4Xtable2 and use the B4Xtable2 RefreshNow method to show the data

```
Private Sub btnrunsql_Click

    Try
        If tasql.Text = "" Then Return
        B4XTable2.Clear
        Dim sqllines As List = Regex.Split(CRLF,tasql.Text)
        Dim fldstring As String = sqllines.Get(0)
        fldstring = fldstring.SubString(fldstring.IndexOf("SELECT ")+7)
        Dim fldslst As List = Regex.Split(",",fldstring)
        For i = 0 To fldslst.Size -1
            B4XTable2.AddColumn(fldslst.Get(i),B4XTable2.COLUMN_TYPE_TEXT)
        Next
        Private data As List
        data.Initialize
        Select currtab
            Case 0
                Private rs1 As ResultSet = dbsqlite.ExecQuery(tasql.text)
                Do While rs1.NextRow
                    Dim rec As List
                    rec.Initialize
                    For i = 0 To fldslst.Size -1
                        Dim fldvalue As String = rs1.GetString2(i)
                        If fldvalue <> Null Then
                            rec.Add(fldvalue)
                        Else
                            rec.Add("NULL")
                        End If
                    Next
                    data.Add(rec)
                Loop
                rs1.Close
            Case 1
                Dim dbname As String = tamysqldb.text
                dbmysql.ExecQuery("USE " & dbname)
                Dim rs As ResultSet = dbmysql.ExecQuery(tasql.text)
                Do While rs.NextRow
```

```
                Dim rec As List
                rec.Initialize
                For i = 0 To fldslst.Size -1
                    Dim fldvalue As String = rs.GetString2(i)
                    If fldvalue <> Null Then
                        rec.Add(fldvalue)
                    Else
                        rec.Add("NULL")
                    End If
                Next
                data.Add(rec)
            Loop
            rs.Close
        End Select
        B4XTable2.SetData(data)
        B4XTable2.RefreshNow
    Catch
        Log(LastException)
        xui.MsgboxAsync("There is a possible syntax error in the query" & CRLF & _
                "Or the database is not active.","Run Query")
    End Try
End Sub
```

- Add code to the btnexportsql_Click subroutine:
  - If the table is not initialized then return from the subroutine
  - Use the get_datalist_from_B4Xtable(B4Xtable2) subroutine to collect the data from the B4Xtable.
  - Set a Word table string using the create_word_table subroutine with the parameters B4Xtable2, data, data size and B4Xtable column size.
  - Set the report title as the SQL query from the textarea.
  - Create a Word document (download and set the **XLUtils library** if you don't have it yet) with the Word table string and the title as parameters

```
Private Sub btnexportsql_Click
    Try
        If B4XTable2.IsInitialized = False Then Return
        Private data As List = get_datalist_from_B4XTable(B4XTable2)
        Dim strtable As String =
create_word_table(B4XTable2,data,data.Size,B4XTable2.Columns.Size)
        Dim title As String = "SQL query:" & CRLF & tasql.text
        Dim strtable As String =
create_word_table(B4XTable2,data,data.Size,B4XTable2.Columns.Size)
        create_word_document(strtable,title)
    Catch
        Log(LastException)
        xui.MsgboxAsync("There was an error in collecting data from the table","Export SQL")
    End Try
End Sub
```

- Create the code for the get_datalist_from_B4Xtable subroutine. The sql1 object from the B4XTable is used to query the data list from the B4Xtable:

```
Private Sub get_datalist_from_B4XTable(B4Xtbl As B4XTable) As List
    Private data As List
    data.Initialize
    If B4Xtbl.Size = 0 Then Return Null
    If B4Xtbl.IsInitialized = False Then Return Null
    Dim rs As ResultSet = B4Xtbl.sql1.ExecQuery("SELECT * FROM data")
```

```
	Do While rs.NextRow
		Dim rec As List
		rec.Initialize
		For i = 0 To B4Xtbl.Columns.Size -1
			rec.Add(rs.GetString2(i))
		Next
		data.Add(rec)
	Loop
	rs.Close
	Return data
End Sub
```

- Add the code for the Word subroutines (maybe put them in a #region Word):
  - A StringBuilder object is used to assemble all the literal string pieces
  - The Word table has a header row that is repeated on each page.
  - The blue_line.png can be found as an attachment in the thread on the website.
  - Create a variable Wutils in the Class_Globals subroutine and initialize it in the B4XmainPage subroutine:

    ```
    Public Wutils As WordUtils
    Wutils.Initialize
    ```
  - You can remove the comment from the line with the Wutils.PowerShellConvertToPdf to have a copy in a PDF format.
  - The Word application is opened with the report document.

```
Private Sub create_word_table(B4Xtbl As B4XTable, datalst As List,numrows As Int, numcols As Int) As String
	Dim sb As StringBuilder
	sb.Initialize
	sb.Append($"[table Alignment=Center rows=${numrows+1} cols=${numcols}]"$)
	sb.Append($"[row RepeatHeader=True]"$)
	For i = 0 To numcols-1
		Dim tblcol As B4XTableColumn = B4Xtbl.Columns.Get(i)
		sb.Append($"[cell color=blue width=100][p alignment=center][color=white][b]${tblcol.title}[/b][/color][/p][/cell]"$)
	Next
	sb.Append($"[/row]"$)
	For i = 0 To datalst.Size -1
		sb.Append($"[row]"$)
		Dim rec As List = datalst.Get(i)
		For j = 0 To rec.Size -1
			Dim fldvalue As String = rec.Get(j)
			If fldvalue = "" Then fldvalue = "null"
			sb.Append($"[cell][p alignment=center]${fldvalue}[/p][/cell]"$)
		Next
		sb.Append($"[/row]"$)
	Next
	sb.Append("[/table]")
	Log(sb.ToString)
	Return sb.ToString
End Sub

Private Sub create_word_document(strtable As String,title As String)
	Dim Document As WordDocument = Wutils.CreateDocument
	Document.Append($"
	[header]
```

```
        [p][FieldCode=DATE/]${TAB}${TAB}${TAB}${TAB}${TAB}${TAB}${TAB}${TAB}${TAB
}${TAB}[FieldCode=PAGE/] of [FieldCode=NUMPAGES/][/p]
        [/header]
        [p alignment=left]${title}[/p]
        ${strtable}
        [footer]
        [p][img dir=assets filename="blue_line.png" width=470 height=2/]Generated by report writer
application...${TAB}${TAB}${TAB}${TAB}${TAB}${TAB}${TAB}[FieldCode=PAGE/] of
[FieldCode=NUMPAGES/][/p]
        [/footer]
        "$)
        Dim f As String = Document.SaveAs(xui.DefaultFolder, "Report.docx", True)
'       Wait For (Wutils.PowerShellConvertToPdf(f, File.Combine(xui.DefaultFolder, "Report.pdf"), True))
Complete (Success As Boolean)
        Wait For (Wutils.OpenWord(f)) Complete (Success As Boolean)
End Sub
```

In the next step the focus will be on the design aspect of the report…