ABMaterial For Dummies
(I resemble that remark)

Lesson 1 - The ABM Template Project

Alwaysbusy (aka ab) graciously provides many fully built apps with each version release of ABMaterial. Each has been updated to the latest requirements of the release.

However, as has been often noted by many newcomers, these marvels of code engineering are no place to start learning about ABMaterial. They seem frightfully complex to the novice! We agree.

The simplest of all these apps is the Template project. Even though it provides everything to make an ABM web app, it essentially does and shows nothing! It was left up to us to “fill in the blanks” and make something (anything) of it. That is what we are about to do (show).

Project Files

Every ABM project consists of many files. In this lesson, we shall concern ourselves with the following two:

ABMPageTemplate.bas
ABMShared.bas

These others play significant roles, but that is for another lesson.

ABMApplication.bas
ABMCacheControl.bas
ABMCacheScavenger.bas
ABMErrorHandler.bas
ABMRootFilter.bas
AboutPage.bas
DBM.bas
Template.b4j (your project Main file)

There are other files required - very many. These are located in the www folder - and found in their own folders… (js, font, css).

Also, the act of compiling creates the required files that allow the app to run (.jar, .html, css, .js, etc). We won’t touch on those now. As non-traditional web app developers - we shouldn’t need to. This is **EXACTLY** the reason ABMaterial was created in the first place. **It shields us from the madness of coupling the usual technologies required to make functional web apps!**

B4J - The best things in life are free…

I can only say this about B4J - know it, love it. Without B4J - ABMaterial is Jack Squat. **ABM is nothing more than the (ultra fancy - material design) GUI for B4J.** Those words were spoken by the creator of ABM (alwaysbusy).

ABMPageTemplate - Our page file

Pages contain what the user sees and interacts with. Each page will usually (ok, always) contain some identical methods - the ones that actually make it connect to a web socket and process user actions (component clicks).

Let us review these required methods most briefly.

Class_Globals - not ABM specific - required in all class modules

```
' name of the page, must be the same as the class name (case sensitive!)
Public Name As String = "ABMPageTemplate"
```

Initialize - again, required by class modules.

```
' build the local structure IMPORTANT!    The app inits all pages before starting
```

WebSocket_Connected (WebSocket1 As WebSocket)

```
' handles the web socket “stuff” (browser to server) Too complex for now..
```

Page_ParseEvent(Params As Map)

```
' the place where magic happens…Interprets user actions (clicks) and
processes them - through events.
```

BuildTheme()

```
' start with the base theme defined in ABMShared Code Module
' themes “style” your components…
```

The following 2 methods are where you will actually layout the page (with rows and cells) and add various ABM components to each row / cell. You will often need to create event handlers for components (the component ID name and _CLicked()) to make things happen!

BuildPage()
' Each Functional Page Must have THIS (BuildPage())!!!!!!!
' This is called from: "Public Sub Initialize" near the top of this form..
‘ Many things go on in here, but most important is adding the rows and cells for your components.
‘ Use this model / ordered structure for all future pages you will create

ConnectPage()
‘ This is where you declare and init components that are added to Rows/Cells on your page.

Page_NavigationbarClicked(Action As String, Value As String)
‘ This is where the app is directed when the navigation (items) are clicked

_Clicked(Target As String)
‘ These are YOUR event handlers. Do what you must here…

ABMShared - The Shared Code Module

Pages share common methods. This is a place to put them - just like you would in any other app you build.

BuildTheme(themeName As String)
' themes are used to style your components

' init the theme
MyTheme.Initialize(themeName)

' create themes for various components with MyTheme.Add (component type)

' the right TEXTALIGN_RIGHT label theme
MyTheme.AddLabelTheme("lbltheme1")
MyTheme.Label("lbltheme1").ForeColor = ABM.COLOR_BLACK
MyTheme.Label("lbltheme1").FontWeight = "BOLD"
MyTheme.Label("lbltheme1").ZDepth = ABM.ZDEPTH_4
MyTheme.Label("lbltheme1").Align = ABM.TEXTALIGN_RIGHT

ConnectNavigationBar(page As ABMPage)
' add NavigationBar items
' direct the nav to an associated page in the last parameter - "../AboutPage"
page.NavigationBar.AddSideBarItem("About","About/Help",
“mdi-action-dashboard", "../AboutPage")

Now, open these files in the B4X IDE and study the comments. Hopefully you will gain some understanding of the logic that makes a page (ABM) work. It is, after all and essentially, a place where you put pre-built (ABM) components in a formal structure that creates a remarkable web (browser) interface - **WITH VERY LITTLE EFFORT!**

Yes, we have just scratched the surface here in Lesson 1. That was the objective. Without knowing this, you would not be able to move forward - to more advanced topics. That said, “more advanced” implies more of the same - only slightly different.

Further Study

Go back and visit AB’s demo web site and review all of what is presented there.

Start from the top item and work your way down. He has done an excellent job of showing and explaining as you shall agree - ***NOW that you have some freakin idea about what the heck he is talking about!***

See you in Lesson 2 - Login…