ABMaterial 2.17 Update/Upgrade CheckList

# Introduction

This document is based on the Template project of ABMaterial 2.17.

## 1.1. The flow in ABMaterial 2.17+

1. You start compiled .jar file on your server (e.g. java -jar MyApp.jar)
2. A user loads the app in his browser (e.g. <http://localhost:51042/MyApp>)
3. The WebApp redirects the user to **InitialPage** defined in ABMApplication
4. If no session exists for this user, it enters the ABMSessionCreator class.
5. A new instance of your page class is created (Initialize is called, we call the BuildPage() method containing the GRID structure of our page).
6. WebSocket\_Connected is called when the DOM of the page in the browser is fully loaded:

If it is a **new session**, We initialize the page object and save everything in the **cache**.

If it is an **existing session**, the page and all global variables are restored from the **cache**.

1. Everything is send to the Browser. We finish this method by doing a **page.Refresh** and tell the browser we’re done by calling **page.FinishedLoading**
2. When the user leaves the page, a **beforeunload** event is raised.
3. WebSocket\_Disconnected is called.
4. If no reconnection happens after some time, the session is destroyed by the server. De cache scavenger will do some cleanup.

If a user comes back to the app while the page is still in cache, it will continue with the cached page. If the cached page is gone, the user is redirected to the entry point of the app. Note that while the user is still connected with the internet, the browser will periodically send a ‘heartbeat’ to the server to keep the session ‘Alive’.

An ABMaterial project has the following components (RED means things that **NEED** to be changed to work with/upgrade to ABMaterial 2.17+, or are very important).

## 1.2. The components of an ABMaterial project

### 1.2.1. Main

The B4J app starting point. Additional to normal B4J initializations, this is the place where you create the ABMApplication and add your own WebApp pages. Finally, you start the Server.

'Non-UI application (console / server application)

#Region Project Attributes

#CommandLineArgs:

#MergeLibraries: True

#End Region

Sub Process\_Globals

Public srvr As Server

End Sub

Sub AppStart (Args() As String)

' the user needs to login

'ABMShared.NeedsAuthorization = True

' Build the Theme

ABMShared.BuildTheme("mytheme")

' create the app

Dim myApp As ABMApplication

myApp.Initialize

' create the pages

Dim myPage As ABMPageTemplate

myPage.Initialize

' add the pages to the app

myApp.AddPage(myPage.Page)

' start the server

myApp.StartServer(srvr, "srvr", 51042)

' optional if you want to redirect the logs to a file in release mode

ABMShared.RedirectOutput(File.DirApp, "logs.txt")

StartMessageLoop

End Sub

### 1.2.2. ABMApplication

The ABMaterial application. Here you can set general properties for the WebApp. In general, this does not need much changes and can be left as is.

**IMPORTANT**

1. Set the Initial starting page using the **InitialPage** variable in Class\_Globals
2. Set the Application name (do not use spaces or special characters!). This is the entry point to your WebApp and all pages will be created under this. You can use **ABMShared.AppName** in the initialize() method.

### 1.2.3. ABMSessionCreator

The ABMSessionCreator is a B4J filter which regulates a new websocket connection. In general, this does not need much changes and can be left as is.

### 1.2.4. ABMUploadHandler

The ABMUploadHandler is a B4J Server Handler that handles the uploads from the user. In general, this does not need much changes and can be left as is.

### 1.2.5. ABMCacheScavenger

Its job is taking care of **cleaning the cached pages** to preserve memory in the server app.

### 1.2.6. ABMShared

A B4J module where code is shared between all users. It can be useful to write general methods shortcuts to create a header, themes, or the navigation bar. It also must contain the NavigateToPage method.

### 1.2.7. ABMPages (your own pages)

These are your actual WebApp pages containing your own ABMaterial components and logic. An ABM Web Page NEEDS a certain structure and methods. You can use (copy) the ABMPageTemplate and ABMPageTemplateAlternative (see further) classes to get started with your own pages.

# Preparation to upgrade/update from ABMaterial versions BEFORE 2.17

If you are new, read them too as it contains some explaining on how the inner workings of ABMaterial work.

## ABMShared

Sub Process\_Globals

Public MyTheme As ABMTheme

Private ABM As ABMaterial 'ignore

Public NeedsAuthorization As Boolean = False

Public AppVersion As String = DateTime.now

Public AppPublishedStartURL As String = ""

Public AppName As String = ""

Public CachedPages As Map

Public CacheScavengePeriodSeconds As Int = 15\*60 ' 15 minutes ' 10 minutes

Public SessionMaxInactiveIntervalSeconds As Int = 30\*60 ' 30 minutes

'1\*60\*60 ' one hour ' -1 = immortal but beware! This means the cache is NEVER emptied!

End Sub

**CacheScavengePeriodSeconds**: this is the interval the Scavenger will try to clean up disconnected pages. They will not be cleaned up as long as the websocket has a 'working' session. (Default 15 minutes)

**SessionMaxInactiveIntervalSeconds**: this is an important one where you have to decide how you want your app to work. In case of an app with little new connection, you can put this up rather high (days). But if you have a lot of new/different connections, you better set this lower to free up meomory in the server app. (Default 30 minutes)

***The new NavigateToPage method***

As we are leaving the page here, we can remove the cached version here instead of waiting for the Scavenger.

Public Sub NavigateToPage(ws As WebSocket, PageId As String, TargetUrl As String)

If AppVersion <> "" Then

TargetUrl = TargetUrl & "?" & AppVersion

End If

ABM.RemoveMeFromCache(CachedPages, PageId)

If ws.Open Then

ws.Eval("window.location = arguments[0]", Array As Object(TargetUrl))

ws.Flush

End If

End Sub

***Or an alternative so the user can open multiple tabs of the same page, except on mobiles/tablets.***

Public Sub NavigateToPageNewTab(ws As WebSocket, PageId As String, TargetUrl As String, OpenInNewTab As Boolean)

If AppVersion <> "" Then

TargetUrl = TargetUrl & "?" & AppVersion

End If

ABM.RemoveMeFromCache(CachedPages, PageId)

If ws.Open Then

If OpenInNewTab Then

Dim s As String

' check if a mobile phone only

' s = $"var check = false;

' (function(a){if(/(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|compal|elaine|fennec|hiptop|iemobile|ip(hone|od)|iris|kindle|lge |maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm( os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)0|symbian|treo|up\.(browser|link)|vodafone|wap|windows ce|xda|xiino/i.test(a)||/1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a wa|abac|ac(er|oo|s\-)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s )|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\/|capi|ccwa|cdm\-|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|\_)|g1 u|g560|gene|gf\-5|g\-mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-| |\_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac( |\-|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt( |\/)|klon|kpt |kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-w])|libw|lynx|m1\-w|m3ga|m50\/|ma(te|ui|xo)|mc(01|21|ca)|m\-cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-| |o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-g|qa\-a|qc(07|12|21|32|60|\-[2-7]|i\-)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-|oo|p\-)|sdk\/|se(c(\-|0|1)|47|mc|nd|ri)|sgh\-|shar|sie(\-|m)|sk\-0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-|v\-|v )|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-|tdg\-|tel(i|m)|tim\-|t\-mo|to(pl|sh)|ts(70|m\-|m3|m5)|tx\-9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3]|\-v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| )|webc|whit|wi(g |nc|nw)|wmlb|wonu|x700|yas\-|your|zeto|zte\-/i.test(a.substr(0,4))) check = true;})(navigator.userAgent||navigator.vendor||window.opera);

' if (check) {

' window.location = arguments[0];

' } else {

' window.open(arguments[0],'\_blank');

' }"$

' check if a mobile phone or a tablet

s = $"var check = false;

(function(a){if(/(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|compal|elaine|fennec|hiptop|iemobile|ip(hone|od)|iris|kindle|lge |maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm( os)?|phone|p(ixi|re)\/|plucker|pocket|psp|series(4|6)0|symbian|treo|up\.(browser|link)|vodafone|wap|windows ce|xda|xiino|android|ipad|playbook|silk/i.test(a)||/1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a wa|abac|ac(er|oo|s\-)|ai(ko|rn)|al(av|ca|co)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|attw|au(di|\-m|r |s )|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-(n|u)|c55\/|capi|ccwa|cdm\-|cell|chtm|cldc|cmd\-|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-s|devi|dica|dmob|do(c|p)o|ds(12|\-d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-7]0|os|wa|ze)|fetc|fly(\-|\_)|g1 u|g560|gene|gf\-5|g\-mo|go(\.w|od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-|hi(pt|ta)|hp( i|ip)|hs\-c|ht(c(\-| |\_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac( |\-|\/)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt( |\/)|klon|kpt |kwc\-|kyo(c|k)|le(no|xi)|lg( g|\/(k|l|u)|50|54|\-[a-w])|libw|lynx|m1\-w|m3ga|m50\/|ma(te|ui|xo)|mc(01|21|ca)|m\-cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-| |o|v)|zz)|mt(50|p1|v )|mwbp|mywa|n10[0-2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0(0|1)|10)|ne((c|m)\-|on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-g|qa\-a|qc(07|12|21|32|60|\-[2-7]|i\-)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/|sa(ge|ma|mm|ms|ny|va)|sc(01|h\-|oo|p\-)|sdk\/|se(c(\-|0|1)|47|mc|nd|ri)|sgh\-|shar|sie(\-|m)|sk\-0|sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-|v\-|v )|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-|tdg\-|tel(i|m)|tim\-|t\-mo|to(pl|sh)|ts(70|m\-|m3|m5)|tx\-9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3]|\-v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-| )|webc|whit|wi(g |nc|nw)|wmlb|wonu|x700|yas\-|your|zeto|zte\-/i.test(a.substr(0,4))) check = true;})(navigator.userAgent||navigator.vendor||window.opera);

if (check) {

window.location = arguments[0];

} else {

window.open(arguments[0],'\_blank');

}"$

ws.Eval(s, Array As Object(TargetUrl))

Else

ws.Eval("window.location = arguments[0]", Array As Object(TargetUrl))

End If

ws.Flush

End If

End Sub

***Handy method to redirect the log output to a file in release mode:***

Sub RedirectOutput (Dir As String, FileName As String) 'ignore

#if RELEASE

Dim out As OutputStream = File.OpenOutput(Dir, FileName, False) 'Set to True to append the logs

Dim ps As JavaObject

ps.InitializeNewInstance("java.io.PrintStream", Array(out, True, "utf8"))

Dim jo As JavaObject

jo.InitializeStatic("java.lang.System")

jo.RunMethod("setOut", Array(ps))

jo.RunMethod("setErr", Array(ps))

#end if

End Sub

If you use LogOff, you also will have to change this method (You can use page.GetPageID as the pageID parameter)

Sub **LogOff**(page As ABMPage)

' do whatever you have to do to log off your user

page.ws.Session.SetAttribute("IsAuthorized", "")

NavigateToPage(page.ws, page.GetPageID, "../")

End Sub

## ABMSessionCreator

Worth noticing is that we no **NOT use ABMNewSession** any more.

'Filter class

Sub Class\_Globals

End Sub

Public Sub Initialize

End Sub

'Return True to allow the request to proceed.

Public Sub Filter(req As ServletRequest, resp As ServletResponse) As Boolean

DateTime.DateFormat = "dd/MM/yyyy"

DateTime.TimeFormat = "HH:mm"

Log("In filter: " & DateTime.Date(DateTime.Now) & " " & DateTime.Time(DateTime.now))

req.GetSession 'a new session will be created if a session doesn't exist.

Return True

End Sub

## ABMCacheScavenger (New Class)

Its job is taking care of **cleaning the cached pages** to preserve memory in the server app.

'CacheScavenger

Sub Class\_Globals

Private scavengeTimer As Timer

Private ABM As ABMaterial

End Sub

Public Sub Initialize

scavengeTimer.Initialize("ScavengeTimer", ABMShared.CacheScavengePeriodSeconds \* 1000)

scavengeTimer.Enabled = True

StartMessageLoop '<- don't forget!

End Sub

Sub ScavengeTimer\_Tick

'do the work required

ABM.ScavengeCache(ABMShared.CachedPages)

End Sub

## ABMApplication

The **Page\_Ready** event does **not exist any more** (and will never be raised!). Instead move this code (without the call to connectPage() else we loop), at the end of your Connect\_Page() method. There is little need to make changes in this class, except if you are more of an exprert and want to add your own login system for example.

'Main application

Sub Class\_Globals

' change to match you app

Private InitialPage As String = "ABMPageTemplate" '<-------- First page to load

' NOTE: Once you've set the above parameters, run the App once. That way, the complete folder structure for your app will be created

' /appname/

' /appname/images/

' /appname/uploads/

' You can then put the images you need in the pages into the /appname/images/ folder and start using them.

' other variables needed

Private AppPage As ABMPage

Private theme As ABMTheme

Private ws As WebSocket 'ignore

Private ABM As ABMaterial 'ignore

Private Pages As List

Private PageNeedsUpload As List

Private ABMPageId As String = ""

End Sub

Public Sub Initialize

Pages.Initialize

PageNeedsUpload.Initialize

ABM.AppVersion = ABMShared.AppVersion

ABM.AppPublishedStartURL = ABMShared.AppPublishedStartURL

ABMShared.AppName = "template" '<-------------------------------------------------------- IMPORTANT

' add your icons

' ABM.AddAppleTouchIcon("", "")

' ABM.AddMSTileIcon("", "")

' ABM.AddFavorityIcon("", "")

#If RELEASE

ABM.PreloadAllJavascriptAndCSSFiles=True ' NEW

ABM.ActivateGZip("DONATORKEY", 1000) ' NEW

ABM.AppDefaultPageCSSInline=True ' NEW

ABM.AppDefaultPageJSInline=True ' NEW

Dim folders As List ' NEW

folders.Initialize

folders.Add(File.DirApp & "/www/" & ABMShared.AppName & "/images")

ABM.ActivatePNGOptimize("DONATORKEY", folders, False , 9, False, True )

#End If

' build the local structure IMPORTANT!

BuildPage

End Sub

At this point the initial page (what you have done in the BuildPage() has been loaded in the browser. Before 2.17, a second event called Page\_Ready was needed, but this was not a very reliable way. **So now when WebSocket\_Connected() called, we know we have a html file and all javascript libraries are loaded.**

**In essence, this is ALL you do in WebSocket\_Connected! NO additional SetWebSockets, Page.Prepares or Page.Refreshes here**. They have all been handled.

ALSO: Do **NOT** add ABMComponents to your page here. Use ConnectPage for that (see further). The only thing you do here is things like Initializing timers, setting/getting other session variables, getting setting variables from the local storage.

Private Sub WebSocket\_Connected (WebSocket1 As WebSocket)

Log("Connected")

ws = WebSocket1

ABMPageId = ABM.GetPageID(AppPage, ABMShared.AppName,ws)

Dim session As HttpSession = ABM.GetSession(ws, ABMShared.SessionMaxInactiveIntervalSeconds)

' Prepare the page IMPORTANT!

AppPage.Prepare

' Run ConnectPage here in ABMApplication

ConnectPage

' navigate to the first page

If ABMShared.NeedsAuthorization Then

If Session.GetAttribute2("IsAuthorized", "") = "" Then

AppPage.ShowModalSheet("login")

Return

End If

End If

ABMShared.NavigateToPage(ws, "","./" & InitialPage)

End Sub

**WebSocket\_Disconnected()** is a good place to stop your timers.

Private Sub WebSocket\_Disconnected

Log("Disconnected")

End Sub

In ABMaterial 216+, a new special event “**beforeunload**” is called.

There is no need to add/change anything in Page\_ParseEvent. Leave it as described below.

Sub Page\_ParseEvent(Params As Map)

Dim eventName As String = Params.Get("eventname")

Dim eventParams() As String = Regex.Split(",",Params.Get("eventparams"))

If eventName = "beforeunload" Then

Log("preparing for url refresh")

ABM.RemoveMeFromCache(ABMShared.CachedPages, ABMPageId)

Return

End If

If SubExists(Me, eventName) Then

Params.Remove("eventname")

Params.Remove("eventparams")

Select Case Params.Size

Case 0

CallSub(Me, eventName)

Case 1

CallSub2(Me, eventName, Params.Get(eventParams(0)))

Case 2

If Params.get(eventParams(0)) = "abmistable" Then

Dim PassedTables As List = ABM.ProcessTablesFromTargetName(Params.get(eventParams(1)))

CallSub2(Me, eventName, PassedTables)

Else

CallSub3(Me, eventName, Params.Get(eventParams(0)), Params.Get(eventParams(1)))

End If

Case Else

' cannot be called diretly, to many param

CallSub2(Me, eventName, Params)

End Select

End If

End Sub

Adds and writes your pages to disk (everything you have written in a BuildPage() method.

public Sub AddPage(Page As ABMPage)

Pages.Add(Page.Name)

PageNeedsUpload.Add(ABM.WritePageToDisk(Page, File.DirApp & "/www/" & ABMShared.AppName & "/" & Page.Name & "/", Page.PageHTMLName, ABMShared.NeedsAuthorization))

End Sub

Starts the server (two methods: HTTP1.1 and HTTP2). In 2.17, we also have to add our Scavenger as a BackgroundWorker (jServer 2.70+ is needed!, see the B4J forum). We also need to set the IdleTimout on all the connections.

Finally, we must initialize the CachedPages map as a ThreadSafeMap which will hold our cached pages.

public Sub StartServer(srvr As Server, srvrName As String, srvrPort As Int)

ABM.WriteAppLauchPageToDisk(AppPage, File.DirApp & "/www/" & ABMShared.AppName, "index.html", ABMShared.NeedsAuthorization)

' start the server

srvr.Initialize(srvrName)

srvr.AddFilter("/js/b4j\_ws.min.js", "ABMSessionCreator", False)

srvr.AddWebSocket("/ws/" & ABMShared.AppName, "ABMApplication")

For i =0 To Pages.Size - 1

srvr.AddWebSocket("/ws/" & ABMShared.AppName & "/" & Pages.Get(i) , Pages.Get(i))

If PageNeedsUpload.Get(i) Then

srvr.AddHandler("/" & ABMShared.AppName & "/" & Pages.Get(i) & "/abmuploadhandler", "ABMUploadHandler", False)

End If

Next

srvr.AddBackgroundWorker("ABMCacheScavenger")

srvr.Port = srvrPort

#If RELEASE

srvr.SetStaticFilesOptions(CreateMap("cacheControl": "max-age=604800,public","gzip":True,"dirAllowed":False))

#Else

srvr.SetStaticFilesOptions(CreateMap("cacheControl": "max-age=604800,public","gzip":False,"dirAllowed":False))

#End If

srvr.Start

Dim jo As JavaObject = srvr

Dim connectors() As Object = jo.GetFieldJO("server").RunMethod("getConnectors", Null)

Dim timeout As Long = ABMShared.SessionMaxInactiveIntervalSeconds\*1000

For Each c As JavaObject In connectors

c.RunMethod("setIdleTimeout", Array(timeout))

Next

ABMShared.CachedPages = srvr.CreateThreadSafeMap

End Sub

public Sub StartServerHTTP2(srvr As Server, srvrName As String, srvrPort As Int, SSLsvrPort As Int, SSLKeyStoreFileName As String, SSLKeyStorePassword As String, SSLKeyManagerPassword As String)

ABM.WriteAppLauchPageToDisk(AppPage, File.DirApp & "/www/" & ABMShared.AppName, "index.html", ABMShared.NeedsAuthorization)

Dim ssl As SslConfiguration

ssl.Initialize

ssl.SetKeyStorePath(File.DirApp, SSLKeyStoreFileName) 'path to keystore file

ssl.KeyStorePassword = SSLKeyStorePassword

ssl.KeyManagerPassword = SSLKeyManagerPassword

srvr.SetSslConfiguration(ssl, SSLsvrPort)

' start the server

srvr.Initialize(srvrName)

srvr.AddFilter("/js/b4j\_ws.min.js", "ABMSessionCreator", False)

srvr.AddWebSocket("/ws/" & ABMShared.AppName, "ABMApplication")

For i =0 To Pages.Size - 1

srvr.AddWebSocket("/ws/" & ABMShared.AppName & "/" & Pages.Get(i) , Pages.Get(i))

If PageNeedsUpload.Get(i) Then

srvr.AddHandler("/" & ABMShared.AppName & "/" & Pages.Get(i) & "/abmuploadhandler", "ABMUploadHandler", False)

End If

Next

srvr.AddBackgroundWorker("ABMCacheScavenger")

srvr.Port = srvrPort

srvr.Http2Enabled = True

#If RELEASE

srvr.SetStaticFilesOptions(CreateMap("cacheControl": "max-age=604800,public","gzip":True,"dirAllowed":False))

#Else

srvr.SetStaticFilesOptions(CreateMap("cacheControl": "max-age=604800,public","gzip":False,"dirAllowed":False))

#End If

srvr.Start

Dim jo As JavaObject = srvr

Dim connectors() As Object = jo.GetFieldJO("server").RunMethod("getConnectors", Null)

Dim timeout As Long = ABMShared.SessionMaxInactiveIntervalSeconds\*1000

For Each c As JavaObject In connectors

c.RunMethod("setIdleTimeout", Array(timeout))

Next

ABMShared.CachedPages = srvr.CreateThreadSafeMap

End Sub

Always load the base theme from ABMShared. You can add extra page specific themes to it. This is called when the server is started, and is **the only place in the page where themes can be defined**!

public Sub BuildTheme()

' start with the base theme defined in ABMShared

theme.Initialize("pagetheme")

theme.AddABMTheme(ABMShared.MyTheme)

' add additional themes specific for this page

End Sub

Here we build the initial STRUCTURE of the page. We can set some page properties and we have to build our GRID. You add ABMcomponents here, but it is **not advisable**! Components defined here are static. If you want dynamic components, it is better to define them in ConnectPage. We can also add the ModalSheets here.

**NOTE: the page Initialize methods have a new PARAMETER:** ABMShared.SessionMaxInactiveIntervalSeconds

public Sub BuildPage()

' initialize the theme

BuildTheme

' initialize this page using our theme

AppPage.InitializeWithTheme(ABMShared.AppName, "/ws/" & ABMShared.AppName, False, ABMShared.SessionMaxInactiveIntervalSeconds , theme)

AppPage.ShowLoader=True

AppPage.PageTitle = "Template"

AppPage.PageDescription = "Template for ABMaterial, a Material UI Framework for B4J"

AppPage.PageHTMLName = "index.html"

AppPage.PageKeywords = ""

AppPage.PageSiteMapPriority = "0.00"

AppPage.PageSiteMapFrequency = ABM.SITEMAP\_FREQ\_YEARLY

' adding a navigation bar

' create the page grid

AppPage.AddRows(1,True, "").AddCells12(1,"")

AppPage.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

' add a modal sheet template to enter contact information

AppPage.AddModalSheetTemplate(BuildLoginSheet)

' add a error box template if the name is not entered

AppPage.AddModalSheetTemplate(BuildWrongInputModalSheet)

End Sub

Here we define all our components! Note that any definition of a component you make here, is purely **a ‘mirror’ object** of what will be seen in the users browser. This is the reason we do not put dims om ABMComponents in Global\_Class as if you do, they are not ‘synced’ with the users browser. Only when you use a method like Page.Component(), the sync happens between the browser and the server. (e.g. in an ABMInput component, the servers object will receive the actual Text value the users entered in the browser). It is using the JQueryElement and Future mechanism forseen by B4J.

**As Page\_Ready() is depreciated, we also have to add its code to the ConnectPage method!**

public Sub ConnectPage()

' you dynamic stuff

AppPage.Refresh ' IMPORTANT

' Tell the browser we’ finished loading

AppPage.FinishedLoading 'IMPORTANT

AppPage.RestoreNavigationBarPosition 'used to be in the depreciated Page\_Ready() event!

End Sub

Some methods to handle the login system in the application:

Sub msbtn1\_Clicked(Target As String)

Dim mymodal As ABMModalSheet = AppPage.ModalSheet("login")

Dim inp1 As ABMInput = mymodal.Content.Component("inp1")

Dim inp2 As ABMInput = mymodal.Content.Component("inp2")

' here check the login a page against your login database

If inp1.Text <> "demo" Or inp2.Text <> "demo" Then

AppPage.ShowModalSheet("wronginput")

Return

End If

ws.Session.SetAttribute("IsAuthorized", "true")

ABMShared.NavigateToPage(ws, "", "./" & InitialPage)

End Sub

Sub BuildLoginSheet() As ABMModalSheet

Dim myModal As ABMModalSheet

myModal.Initialize(AppPage, "login", False, False, "")

myModal.Content.UseTheme("modalcontent")

myModal.Footer.UseTheme("modalfooter")

myModal.IsDismissible = False

' create the grid for the content

myModal.Content.AddRows(2,True, "").AddCells12(1,"")

myModal.Content.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

' add paragraph

myModal.Content.Cell(1,1).AddComponent(ABMShared.BuildParagraph(AppPage,"par1","Please login: (login and password are 'demo')") )

' create the input fields for the content

Dim inp1 As ABMInput

inp1.Initialize(AppPage, "inp1", ABM.INPUT\_TEXT, "Login", False, "")

myModal.Content.Cell(2,1).AddComponent(inp1)

Dim inp2 As ABMInput

inp2.Initialize(AppPage, "inp2", ABM.INPUT\_TEXT, "Password", False, "")

myModal.Content.Cell(2,1).AddComponent(inp2)

' create the grid for the footer

' we add a row without the default 20px padding so we need to use AddRowsM(). If we do not use this method, a scrollbar will appear to the sheet.

myModal.Footer.AddRowsM(1,True,0,0, "").AddCellsOS(1,9,9,9,3,3,3, "")

myModal.Footer.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

' create the button for the footer

Dim msbtn1 As ABMButton

msbtn1.InitializeFlat(AppPage, "msbtn1", "", "", "Login", "transparent")

myModal.Footer.Cell(1,1).AddComponent(msbtn1)

Return myModal

End Sub

Sub BuildWrongInputModalSheet() As ABMModalSheet

Dim myModalError As ABMModalSheet

myModalError.Initialize(AppPage, "wronginput", False, False, "")

myModalError.Content.UseTheme("modalcontent")

myModalError.Footer.UseTheme("modalcontent")

myModalError.IsDismissible = True

' create the grid for the content

myModalError.Content.AddRows(1,True, "").AddCells12(1,"")

myModalError.Content.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

Dim lbl1 As ABMLabel

lbl1.Initialize(AppPage, "contlbl1", "The login or password are incorrect!",ABM.SIZE\_PARAGRAPH, False, "")

myModalError.Content.Cell(1,1).AddComponent(lbl1)

Return myModalError

End Sub

' clicked on the navigation bar

Sub Page\_NavigationbarClicked(Action As String, Value As String)

AppPage.SaveNavigationBarPosition

End Sub

**NOTE: At the end, do a search for Page\_Ready, as I sometimes forgot that part too to remove this code while upgrading!**

# For each of your own pages.

## The classic ABMaterial way using BuildPage() and ConnectPage()

The Page\_Ready event does not exist any more (and will never be raised!). Instead move this code (without the call to connectPage() else we loop), at the end of your Connect\_Page() method.

'Class module

Sub Class\_Globals

Private ws As WebSocket 'ignore

' will hold our page information

Public page As ABMPage

' page theme

Private theme As ABMTheme

' to access the constants

Private ABM As ABMaterial 'ignore

' name of the page, must be the same as the class name (case sensitive!)

Public Name As String = "ABMPageTemplate" '<-------------------------------------------------------- IMPORTANT

' will hold the unique browsers window id

Private ABMPageId As String = ""

' your own variables

End Sub

This is needed for ABMApplication to write the initial HTML/JavaScript and CSS files.

'Initializes the object. You can add parameters to this method if needed.

Public Sub Initialize

' build the local structure IMPORTANT!

BuildPage

End Sub

At this point the initial page (what you have done in the BuildPage() has been loaded in the browser. Before 2.17, a second event called Page\_Ready was needed, but this was not a very reliable way. **So now when WebSocket\_Connected() called, we know we have a html file and all javascript libraries are loaded.**

**In essence, this is ALL you do in WebSocket\_Connected!** **NO additional SetWebSockets, Page.Prepares or Page.Refreshes here**. They have all been handled.

ALSO: Do **NOT** add ABMComponents to your page here. Use ConnectPage for that (see further). The only thing you do here is things like Initializing timers, setting/getting other session variables, getting setting variables from the local storage.

Private Sub WebSocket\_Connected (WebSocket1 As WebSocket)

Log("Connected")

ws = WebSocket1

ABMPageId = ABM.GetPageID(page, Name,ws)

Dim session As HttpSession = ABM.GetSession(ws, ABMShared.SessionMaxInactiveIntervalSeconds)

If ABMShared.NeedsAuthorization Then

If session.GetAttribute2("IsAuthorized", "") = "" Then

ABMShared.NavigateToPage(ws, ABMPageId, "../")

Return

End If

End If

ABM.UpdateFromCache(Me, ABMShared.CachedPages, ABMPageId, ws)

If page.ComesFromPageCache Then

' refresh the page

page.Refresh

' Tell the browser we finished loading

page.FinishedLoading

Else

' Prepare the page

page.Prepare

' load the dynamic content

ConnectPage

End If

Log(ABMPageId)

End Sub

**WebSocket\_Disconnected()** is a good place to stop your timers.

Private Sub WebSocket\_Disconnected

Log("Disconnected")

End Sub

In ABMaterial 216+, a new special event “**beforeunload**” is called.

There is no need to add/change anything in Page\_ParseEvent. Leave it as described below.

Sub Page\_ParseEvent(Params As Map)

Dim eventName As String = Params.Get("eventname")

Dim eventParams() As String = Regex.Split(",",Params.Get("eventparams"))

If eventName = "beforeunload" Then

Log("preparing for url refresh")

ABM.RemoveMeFromCache(ABMShared.CachedPages, ABMPageId)

Return

End If

If SubExists(Me, eventName) Then

Params.Remove("eventname")

Params.Remove("eventparams")

Select Case Params.Size

Case 0

CallSub(Me, eventName)

Case 1

CallSub2(Me, eventName, Params.Get(eventParams(0)))

Case 2

If Params.get(eventParams(0)) = "abmistable" Then

Dim PassedTables As List = ABM.ProcessTablesFromTargetName(Params.get(eventParams(1)))

CallSub2(Me, eventName, PassedTables)

Else

CallSub3(Me, eventName, Params.Get(eventParams(0)), Params.Get(eventParams(1)))

End If

Case Else

' cannot be called directly, to many param

CallSub2(Me, eventName, Params)

End Select

End If

End Sub

Always load the base theme from ABMShared. You can add extra page specific themes to it. This is called when the server is started, and is **the only place in the page where themes can be defined**!

public Sub BuildTheme()

' start with the base theme defined in ABMShared

theme.Initialize("pagetheme")

theme.AddABMTheme(ABMShared.MyTheme)

' add additional themes specific for this page

End Sub

Here we build the initial STRUCTURE of the page. We can set some page properties and we have to build our GRID. You add ABMcomponents here, but it is **not advisable**! Components defined here are static. If you want dynamic components, it is better to define them in ConnectPage. We can also add the ModalSheets here.

**NOTE: the page Initialize methods have a new PARAMETER:** ABMShared.SessionMaxInactiveIntervalSeconds

public Sub BuildPage()

' initialize the theme

BuildTheme

' initialize this page using our theme

page.InitializeWithTheme(Name, "/ws/" & ABMShared.AppName & "/" & Name, False, ABMShared.SessionMaxInactiveIntervalSeconds, theme)

page.ShowLoader=True

page.PageHTMLName = "index.html"

page.PageTitle = ""

page.PageDescription = ""

page.PageKeywords = ""

page.PageSiteMapPriority = ""

page.PageSiteMapFrequency = ABM.SITEMAP\_FREQ\_YEARLY

page.ShowConnectedIndicator = True

' adding a navigation bar

' create the page grid

page.AddRows(2,True, "").AddCells12(1,"")

page.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

End Sub

Here we define all our components! Note that any definition of a component you make here, is purely **a ‘mirror’ object** of what will be seen in the users browser. This is the reason we do not put dims om ABMComponents in Global\_Class as if you do, they are not ‘synced’ with the users browser. Only when you use a method like Page.Component(), the sync happens between the browser and the server. (e.g. in an ABMInput component, the servers object will receive the actual Text value the users entered in the browser). It is using the JQueryElement and Future mechanism forseen by B4J.

**As Page\_Ready() is depreciated, we also have to add its code to the ConnectPage method!**

public Sub ConnectPage()

' connecting the navigation bar

' init all your own variables (like a List, Map) and add your components

' page.Cell(1,1).AddComponent(ABMShared.BuildParagraph(page, "lbl", "This is a test"))

'

' Dim btn As ABMButton

' btn.InitializeRaised(page, "btn", "", "", "Press me", "")

' page.Cell(2,1).AddComponent(btn)

' refresh the page

page.Refresh

' Tell the browser we finished loading

page.FinishedLoading

' restoring the navigation bar position

page.RestoreNavigationBarPosition 'used to be in the depreciated Page\_Ready() event!

End Sub

Method to handle the navigation through the app from page to page

' clicked on the navigation bar

Sub Page\_NavigationbarClicked(Action As String, Value As String)

' saving the navigation bar position

page.SaveNavigationBarPosition

If Action = "LogOff" Then

ABMShared.LogOff(page)

Return

End If

ABMShared.NavigateToPage(ws, ABMPageId, Value)

End Sub

**NOTE: At the end, do a search for Page\_Ready, as I sometimes forgot that part too to remove this code while upgrading!**

## The alternative: ABMaterial without using BuildPage() and ConnectPage(), more classic B4J style

'Class module

Sub Class\_Globals

Private ws As WebSocket 'ignore

' will hold our page information

Public page As ABMPage

' page theme

Private theme As ABMTheme

' to access the constants

Private ABM As ABMaterial 'ignore

' name of the page, must be the same as the class name (case sensitive!)

Public Name As String = "ABMPageTemplateAlternative" '<-------------------------------------------------------- IMPORTANT

' will hold the unique browsers window id

Private ABMPageId As String = ""

End Sub

**NOTE: the page Initialize methods have a new PARAMETER:** ABMShared.SessionMaxInactiveIntervalSeconds

'Initializes the object. You can add parameters to this method if needed.

Public Sub Initialize

' build the local structure IMPORTANT!

' start with the base theme defined in ABMShared

theme.Initialize("pagetheme")

theme.AddABMTheme(ABMShared.MyTheme)

' add additional themes specific for this page

' initialize this page using our theme

page.InitializeWithTheme(Name, "/ws/" & ABMShared.AppName & "/" & Name, False, ABMShared.SessionMaxInactiveIntervalSeconds, theme)

page.ShowLoader=True

page.PageHTMLName = "index.html"

page.PageTitle = ""

page.PageDescription = ""

page.PageKeywords = ""

page.PageSiteMapPriority = ""

page.PageSiteMapFrequency = ABM.SITEMAP\_FREQ\_YEARLY

page.ShowConnectedIndicator = True

' adding a navigation bar

' create the page grid

page.AddRows(2,True, "").AddCells12(1,"")

page.BuildGrid 'IMPORTANT once you loaded the complete grid AND before you start adding components

End Sub

Private Sub WebSocket\_Connected (WebSocket1 As WebSocket)

Log("Connected")

ws = WebSocket1

ABMPageId = ABM.GetPageID(page, Name,ws)

Dim session As HttpSession = ABM.GetSession(ws, ABMShared.SessionMaxInactiveIntervalSeconds)

If ABMShared.NeedsAuthorization Then

If session.GetAttribute2("IsAuthorized", "") = "" Then

ABMShared.NavigateToPage(ws, ABMPageId, "../")

Return

End If

End If

ABM.UpdateFromCache(Me, ABMShared.CachedPages, ABMPageId, ws)

If page.ComesFromPageCache Then

' refresh the page

page.Refresh

' Tell the browser we finished loading

page.FinishedLoading

Else

' Prepare the page

page.Prepare

' load the dynamic content

' connecting the navigation bar

' init all your own variables (like a List, Map) and add your components

' page.Cell(1,1).AddComponent(ABMShared.BuildParagraph(page, "lbl", "This is a test"))

'

' Dim btn As ABMButton

' btn.InitializeRaised(page, "btn", "", "", "Press me", "")

' page.Cell(2,1).AddComponent(btn)

' refresh the page

page.Refresh

' Tell the browser we finished loading

page.FinishedLoading

' restoring the navigation bar position

page.RestoreNavigationBarPosition 'used to be in the depreciated Page\_Ready() event!

End If

Log(ABMPageId)

End Sub

Private Sub WebSocket\_Disconnected

Log("Disconnected")

End Sub

Sub Page\_ParseEvent(Params As Map)

Dim eventName As String = Params.Get("eventname")

Dim eventParams() As String = Regex.Split(",",Params.Get("eventparams"))

If eventName = "beforeunload" Then

Log("preparing for url refresh")

ABM.RemoveMeFromCache(ABMShared.CachedPages, ABMPageId)

Return

End If

If SubExists(Me, eventName) Then

Params.Remove("eventname")

Params.Remove("eventparams")

Select Case Params.Size

Case 0

CallSub(Me, eventName)

Case 1

CallSub2(Me, eventName, Params.Get(eventParams(0)))

Case 2

If Params.get(eventParams(0)) = "abmistable" Then

Dim PassedTables As List = ABM.ProcessTablesFromTargetName(Params.get(eventParams(1)))

CallSub2(Me, eventName, PassedTables)

Else

CallSub3(Me, eventName, Params.Get(eventParams(0)), Params.Get(eventParams(1)))

End If

Case Else

' cannot be called directly, to many param

CallSub2(Me, eventName, Params)

End Select

End If

End Sub

' clicked on the navigation bar

Sub Page\_NavigationbarClicked(Action As String, Value As String)

' saving the navigation bar position

page.SaveNavigationBarPosition

If Action = "LogOff" Then

ABMShared.LogOff(page)

Return

End If

ABMShared.NavigateToPage(ws, ABMPageId, Value)

End Sub

**NOTE: At the end, do a search for Page\_Ready, as I sometimes forgot that part too to remove this code while upgrading!**