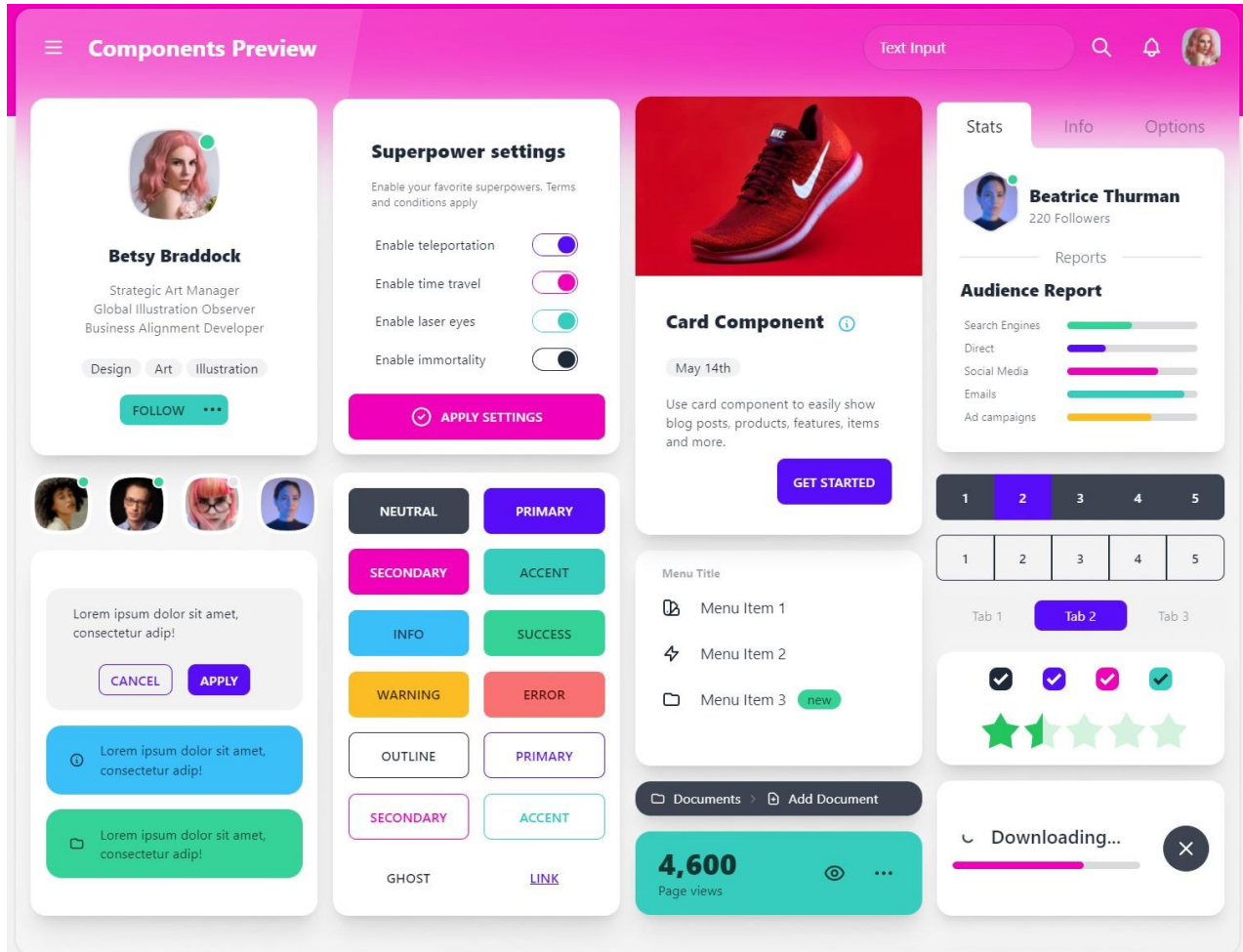


TailwindCSS WebApps



using B4X

Table of contents

Introduction	3
Getting Started	5
New to B4x?	11
Project Templates	11
SithasoDrawer	12
SithasoCanvas.....	13
Creating WebApps.....	16
Creating a Project	16
Creating a Page	21
Creating a Grid.....	26

Introduction

Welcome to the **SithasoDaisy** world.

SithasoDaisy is a library of components built on top of [TailwindCSS](#) and the [DaisyUI](#) frameworks to help you create WebApps, WebSites, Single Page Application (SPA) and Progressive Web Apps (PWA) with the power of the b4x programming language.

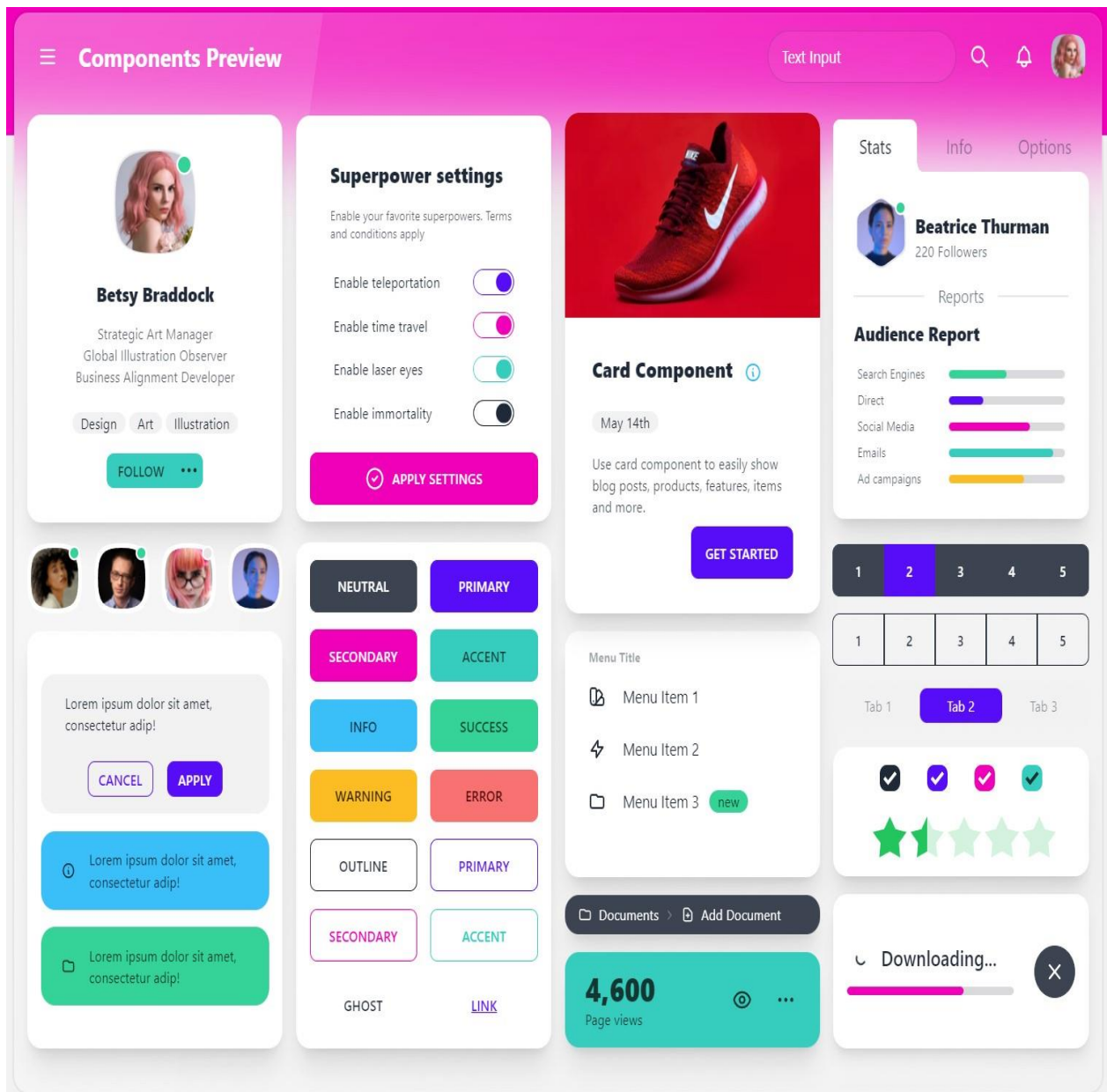
When it comes to developing anything that works on the internet browser, whether it is a WebApp or a WebSite, one has to use HTML (**H**yper **T**ext **M**arkup **L**anguage), CSS (**C**ascading **S**tyle **S**heet) and JavaScript (a dynamic programming language used for web development).

SithasoDaisy works on top of a programming language called **b4x**. It is not JavaScript, and for SithasoDaisy to produce web applications, a code transpiler is used. A transpiler converts source code from one programming language to another. For example, when one uses Flutter for web, they use a programming language called Dart. When they build their application, their source code is transpiled / converted to JavaScript for it to work on the interweb. There are many other programming languages that target JavaScript, the Top 10, being:

1. Scala.js
2. Haxe
3. Dart
4. Elm
5. Imba
6. Nim
7. ClojureScript
8. ReasonML
9. Kotlin
10. TypeScript

B4X is a set of programming tools that is developed by [Anywhere Software](#) that uses [Visual Basic](#) like syntax so that anyone who wants to, can create apps. The developed apps are able to run on Windows, Linux, Mac, Apple Phones, Android Phones and Arduino IoT devices, mostly from the same code base. The family product we will use here is called b4j i.e., Basic4Java. There is also b4a (basic4android), b4i (basic4ios), b4r (basic4arduino).

Our b4x to JavaScript transpiler is called BANano. It is penned by Alain Bailleul, that is the **BA** in BANano, whilst Nano, you guessed it right, nanotechnology. When creating your web projects with SithasoDaisy, one can use the Abstract Designer and or write b4x code. We will show you how. To show you an idea of the stuff we will be building, let's take a look at this image, directly from the DaisyUI website.



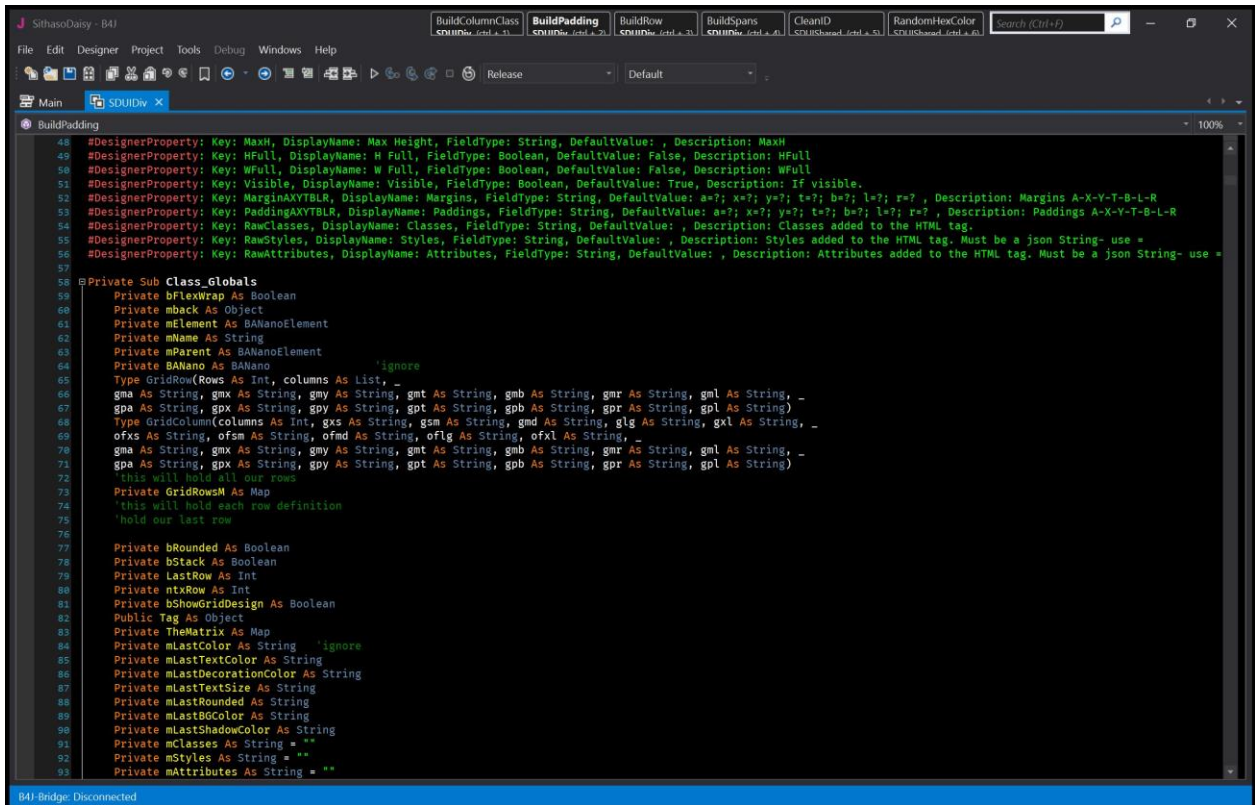
You can explore the complete demo of what and how SithasoDaisy works from this [Netlify](#) Deployment. The source code of the demo is in this [Github repo](#).

With a 64-bit Windows PC, let's get started.

Getting Started

To be able to start developing using B4X, one needs the b4j IDE (Integrated Development Environment. This at the moment only runs on a Windows PC. One also needs the Java SDK. Figure 1 below depicts how the B4X IDE looks like. We have a menu, a toolbar, the coding area and a module listing area, to mention the few.

1. The B4J IDE



2. Creating Folders

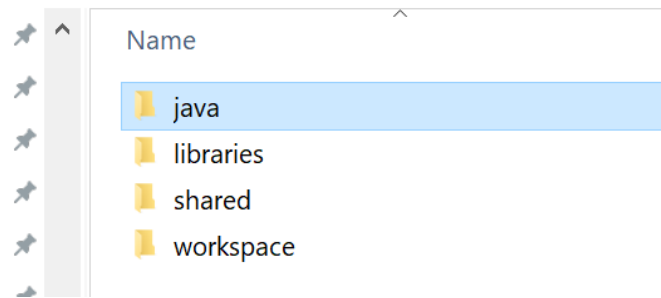
Let's set up our PC for development. We need to set up a folder structure first.

1. In your Windows PC, create the following folder structure:

- (a) c:\b4j\libraries
- (b) c:\b4j\shared
- (c) c:\b4j\workspace
- (d) c:\b4j\java

This should look like:

Local Disk (C:) > b4j



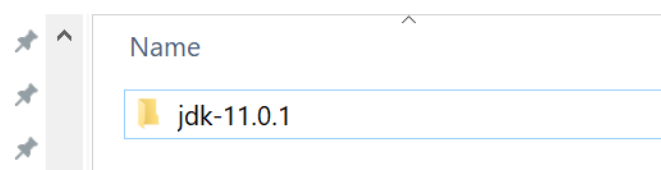
3. Downloading and installing B4J

1. Head over to Anywhere Software Website and download b4j. You can click [here](#) to do that.
2. Click on Download B4J Full Version (64-BIT). After you download, ensure you install the application.

**DOWNLOAD B4J FULL VERSION
(64-BIT)**

3. Also download the recommended OpenJDK 11. You can get it [here](#). Unpack it to **c:\b4j\java**, you should have

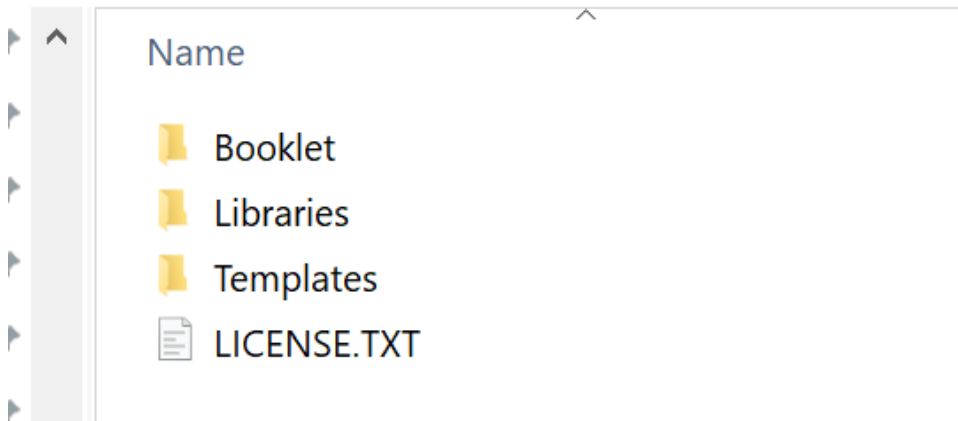
Local Disk (C:) > b4j > java >



4. Download BANano

You will also need BANano. This is a b4j plugin. Click [here](#) to get it and unpack it. It should have this content.







Mashy > Downloads > BANano7.37



In the BANano download copy the contents of the **Libraries** folder to **c:\b4j\libraries**.

5. Download SithasoDaisy

From your **SithasoDaisy** download.

Name	↑
	PocketBase Expense Tracker
	ReadMe First.rtf
	SithasoCanvas.zip
	SithasoDaisy.b4xlib
	SithasoDaisyDemo.zip
	SithasoDrawer.zip

5.1 Copy the **SithasoDaisy.b4xlib** to **c:\b4j\libraries**. You now should have. Also copy the contents of the **b4xtemplates** folder to **c:\libraries**.

< (C:) > b4j > libraries

Name	Date modified	Type	Size
BANano.jar	2022/04/08 12:38	JAR File	691 KB
BANano.xml	2022/04/08 12:31	XML Source File	361 KB
BANanoBase64.b4xlib	2022/02/18 12:41	B4XLIB File	3 KB
BANanoDragula.b4xlib	2022/02/18 12:41	B4XLIB File	10 KB
BANanoLeaflet.b4xlib	2022/02/18 12:43	B4XLIB File	9 KB
BANanoMediaRecorder.b4xlib	2022/02/18 12:43	B4XLIB File	4 KB
BANanoPeer.b4xlib	2022/02/18 12:44	B4XLIB File	46 KB
BANanoServer.b4xlib	2022/02/17 12:26	B4XLIB File	20 KB
BANanoSkeleton.b4xlib	2022/04/08 12:05	B4XLIB File	1 950 KB
BANanoSweetAlert.b4xlib	2022/02/18 12:45	B4XLIB File	4 KB
bcprov-jdk15on-154.jar	2016/04/25 11:59	JAR File	3 201 KB
HikariCP.jar	2018/04/03 12:36	JAR File	3 KB
HikariCP.xml	2018/04/03 12:36	XML Source File	4 KB
HikariCP-4.0.3.jar	2022/03/14 09:04	JAR File	156 KB
HikariCP.txt	2022/02/27 12:49	Text Source File	1 KB
mysql-connector-java-8.0.23.jar	2021/02/21 10:04	JAR File	2 359 KB
servlet-api-3.1.jar	2017/05/10 14:42	JAR File	94 KB
SithasoDaisy.b4xlib	2022/11/06 00:00	B4XLIB File	1 648 KB
slf4j-api-1.7.32.jar	2021/09/02 08:55	JAR File	41 KB
slf4j-simple-1.7.32.jar	2021/09/02 09:35	JAR File	15 KB

5.1 Unpack **SithasoCanvas.zip**, **SithasoDaisyDemo.zip** and **SithasoDrawer.zip** to **c:\b4j\workspace**. You now should have.

Local Disk (C:) > b4j > workspace >

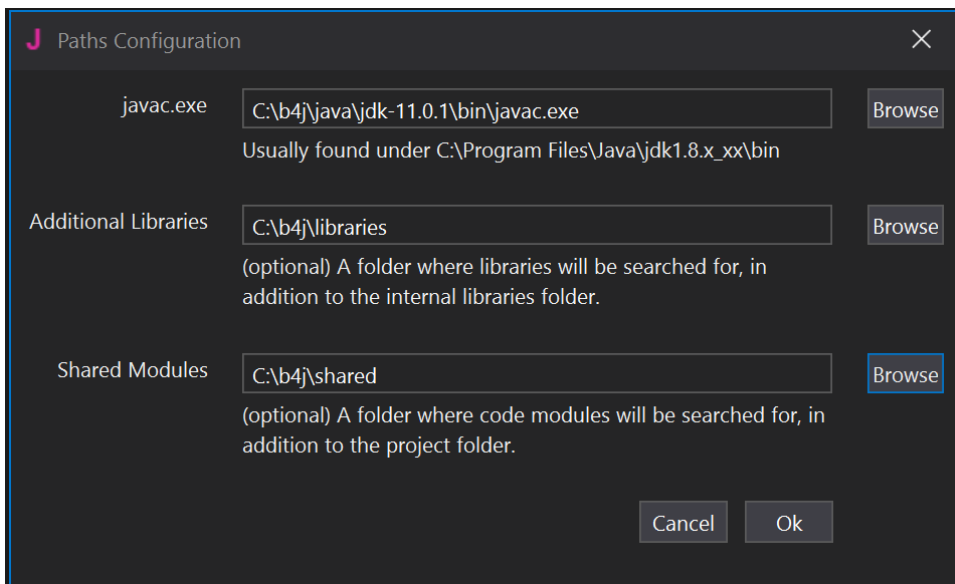
x +

Name
SithasoCanvas
SithasoDaisyDemo
SithasoDrawer

6. B4J Paths Configuration.

Start B4J, in the menu, click on 6.1 **Tools** then **Configure Paths**.

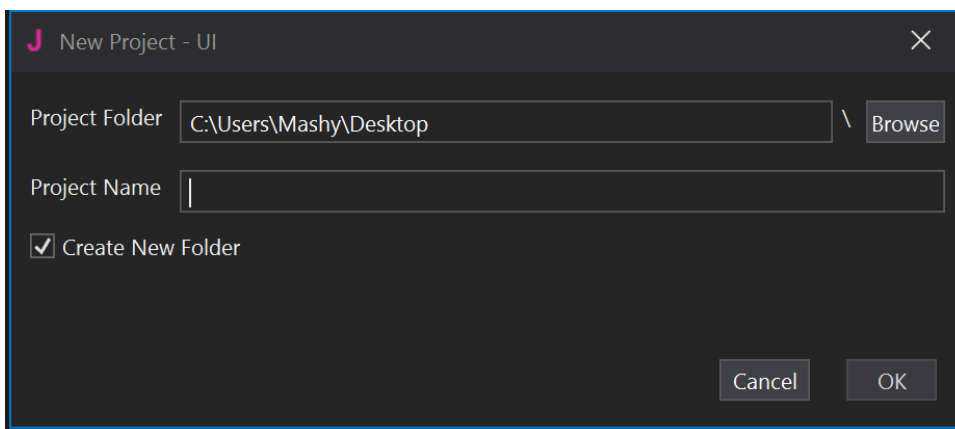
Click on the browse buttons to select the respective file and paths specified below.



When click click Ok, to save your configuration.

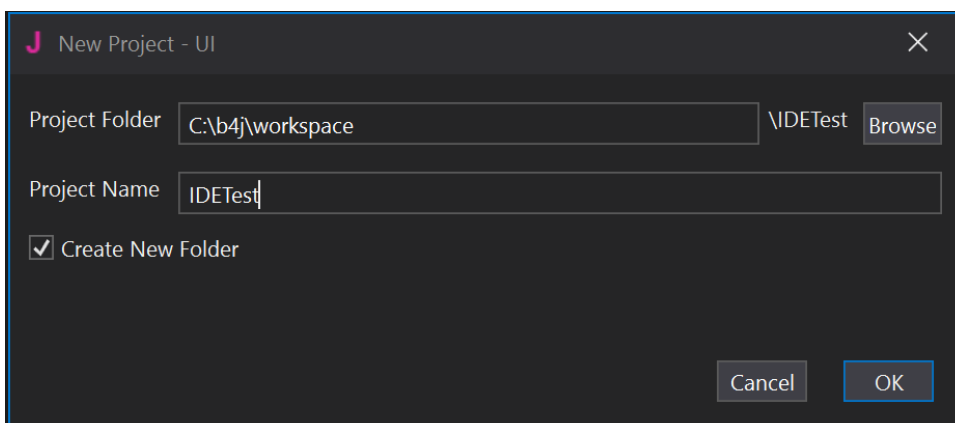
7. Testing B4J IDE readiness

Click on File > New > UI

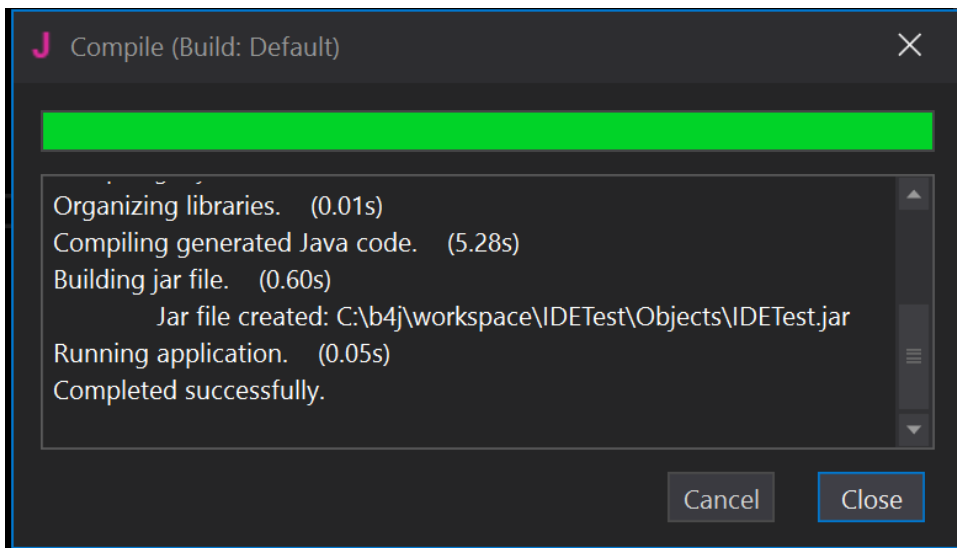


Click on Browse and ensure that the Project folder is **C:\b4j\workspace**

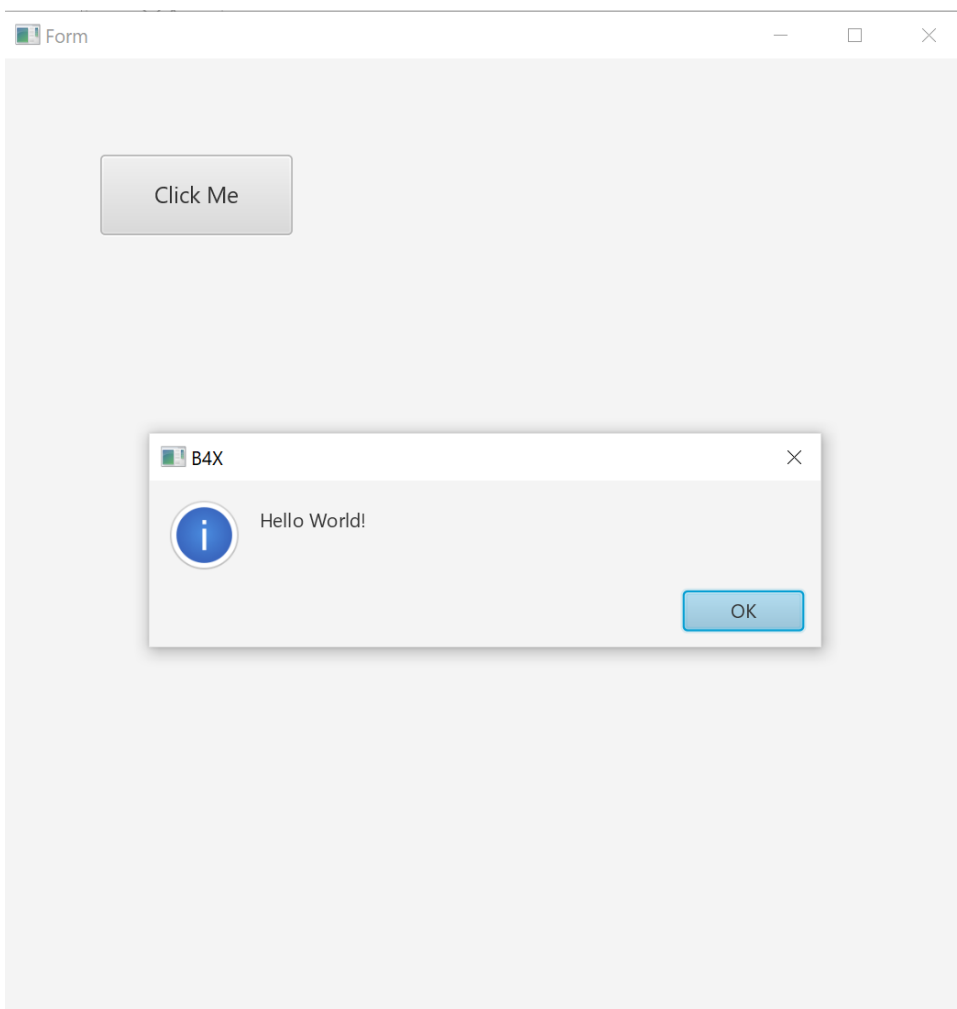
Type in a project name, for example, IDETest and click Ok.



This should open up the IDE with some template code. Press **F5**, this should compile you app and show a screen.



Click on the **Click Me** button. It should show a Message Box.

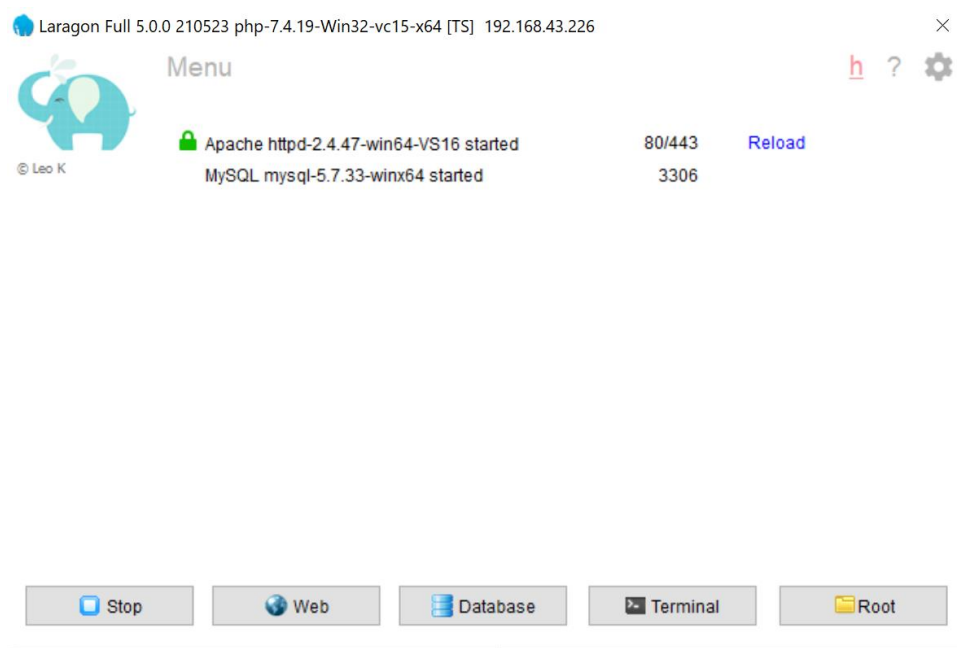


Congratulations, you have just ran your first b4j developed java application with b4x. You can close the App and the IDE.

Lets now run the other applications in our **workspace** folder which are **TailwindCSS** based, for this we will need a development **WebServer**. I like the ease of use of Laragon. It comes with MySQL and other lovely stuff.

8. Installing a WebServer (optional)

Download it from [here](#) and run it. After installation run it, it should look similar to this screen. You can also set it up to use SSL and different ports.



New to B4x?

If you are new to B4x, going through the guidelines would help you a great deal. There is also [Video Material](#) done by Erel, who is the author of the b4x ecosystem.

1. Get all the guides [here](#), these speak to:

- 1.1. B4x language
- 1.2. B4x IDE
- 1.3. B4x Visual Designer / Abstract Designer

and many other useful information.

2. To understand how the BANano transpiler works, read the **BANano Essentials Booklet**

3. You can also join the wonderful community of other coders like you.

Now lets get back to our topic, Creating WebApps with SithasoDaisy. We will first explain the project templates.

Project Templates

SithasoDaisy comes with 2 source code templates for you to create WebApps. These are **SithasoDrawer** and **SithasoCanvas**.

- Use SithasoDrawer as a base for projects with a top navigation bar and a drawer.
- Use SithasoCanvas as a base for projects that you will start from scratch.

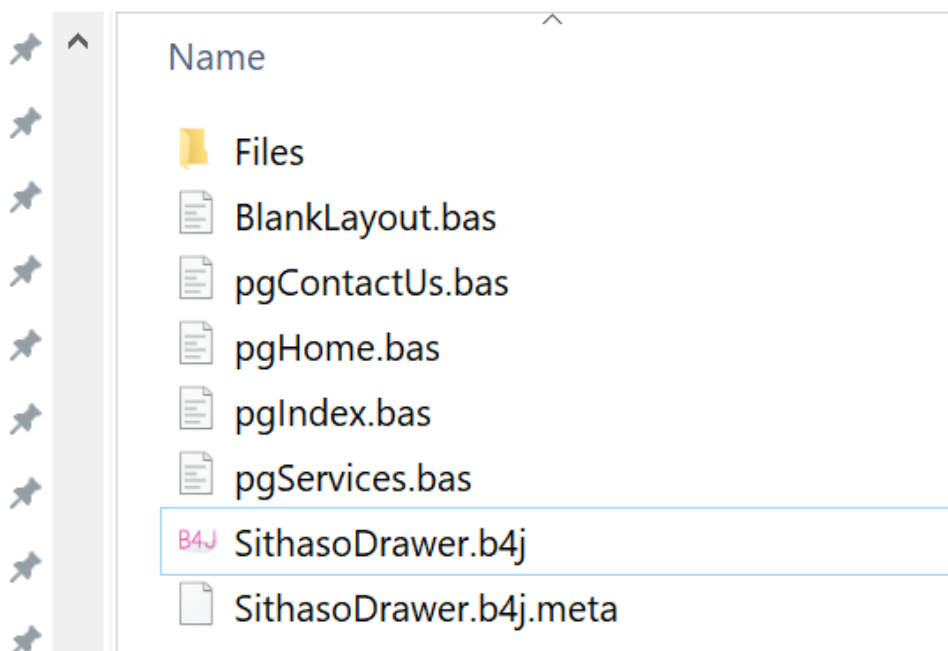
These are set to run on localhost. This is done in the **config.properties** in each of the projects. The projects are also set up to save the transpiled javascript, css and html files to **c:\www\laragon**.

These need to be changed to suit your needs if you are not using laragon and also using different port numbers.

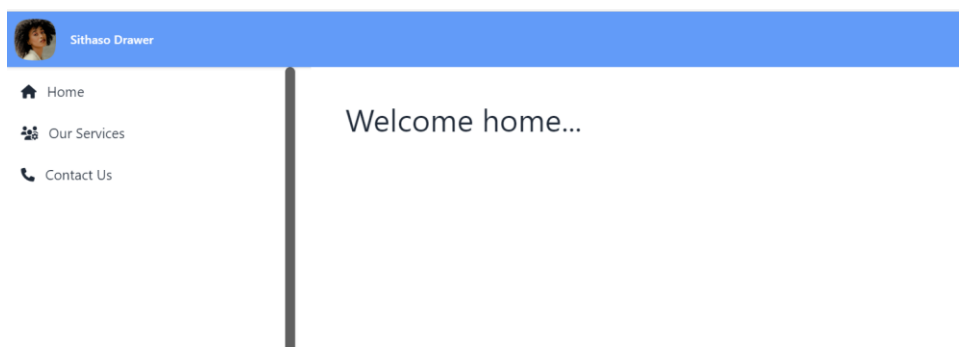
SithasoDrawer

1. Open the c:\b4j\workspace\SithasoDrawer folder
2. Double click the SithasoDrawer.b4j file. This is a b4j project file. This will activate b4j.
3. Press F5 to run the application. This will also transpile your code to JavaScript, CSS, HTML etc

workspace > SithasoDrawer



After compilation, you should see this app in action on your default web browser.



The name of this app is "**sithasodrawer**". This is defined in the **Main** code module.

```

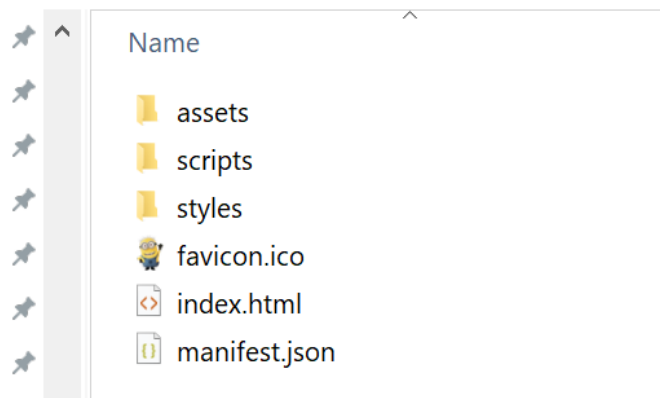
55 #IgnoreWarnings:12, 15
56 Sub Process_Globals
57     Public BANano As BANano 'ignore
58     'the name of the application &
59     'this is the folder on your development server.
60     Public AppName As String = "sithasodrawer"
61     Public AppTitle As String = "Sithaso Drawer"
62     'whe the app should
63     Private Publish As String = "C:\laragon\www"
64     Public Version As String = "0.01"
65     Public ServerIP As String
66 End Sub

```

To access the transpiled source code (javascript, css and html) that resulted with what you see in the browser, head over to the **c:\laragon\www\sithasodrawer** folder.

This contains all the stuff that you can deploy to your public webserver when you are finished developing you webapp.

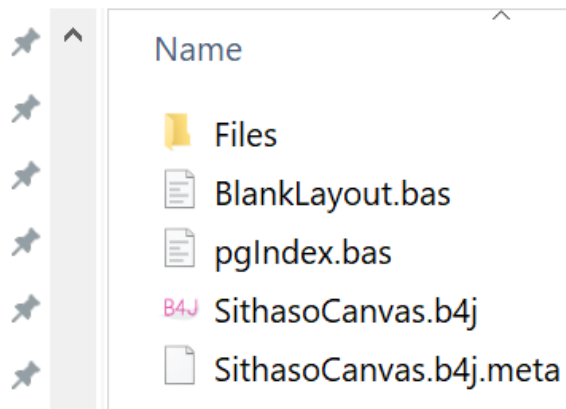
Local Disk (C:) > laragon > www > sithasodrawer



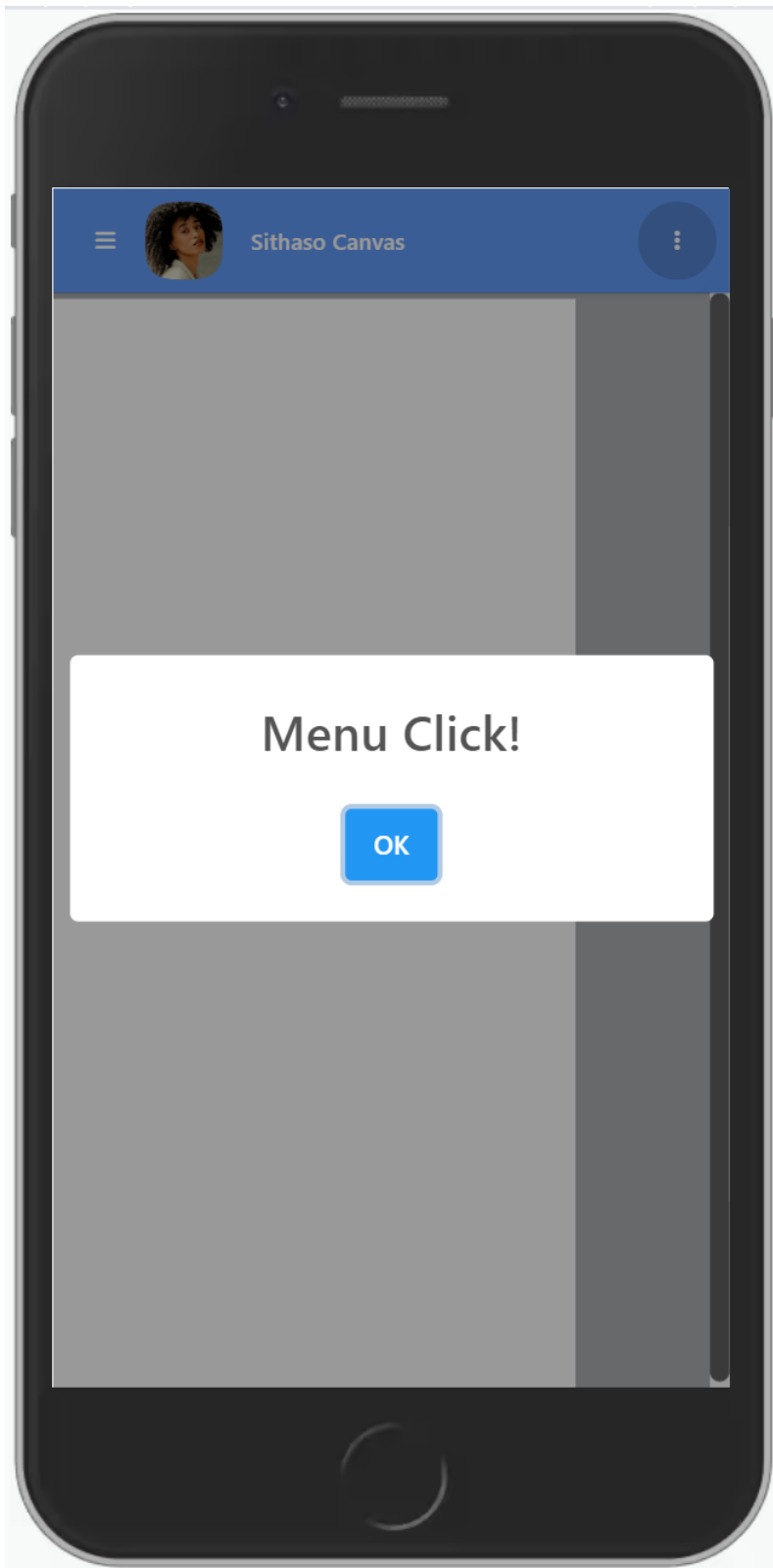
SithasoCanvas

1. Open the c:\b4j\workspace\SithasoCanvas folder
2. Double click the SithasoCanvas.b4j file. This is a b4j project file. This will activate b4j.
3. Press F5 to run the application. This will also transpile your code to JavaScript, CSS, HTML etc

b4j > workspace > SithasoCanvas >



After compilation, you should see this app in action on your default web browser. Click the hamburger or menu.



The name of this app is "**sithasocanvas**". This is defined in the **Main** code module.

```

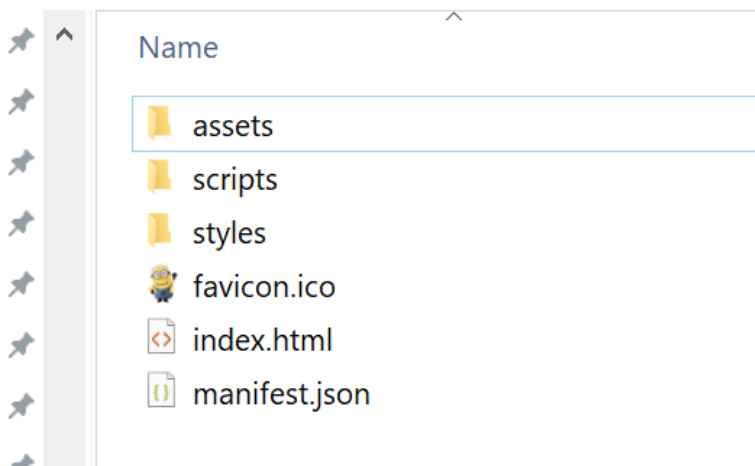
55 #IgnoreWarnings:12, 15
56 Sub Process_Globals
57     Public BANano As BANano 'ignore
58     'the name of the application &
59     'this is the folder on your development server.
60     Public AppName As String = "sithasocanvas"
61     Public AppTitle As String = "Sithaso Canvas"
62     'whe the app should
63     Private Publish As String = "C:\laragon\www"
64     Public Version As String = "0.01"
65     Public ServerIP As String
66 End Sub

```

To access the transpiled source code (javascript, css and html) that resulted with what you see in the browser, head over to the `c:\laragon\www\sithasocanvas` folder.

This contains all the stuff that you can deploy to your public webserver when you are finished developing you webapp.

< (C:) > laragon > www > sithasocanvas



Creating WebApps

To create a webapp, use either **SithasoDrawer** or **SithasoCanvas** project as your base. As your WebApp will possibly have a number of pages, you will use **Code Modules** to create the pages. Each page should be unique, including its name, title, layout and possibly its icon.

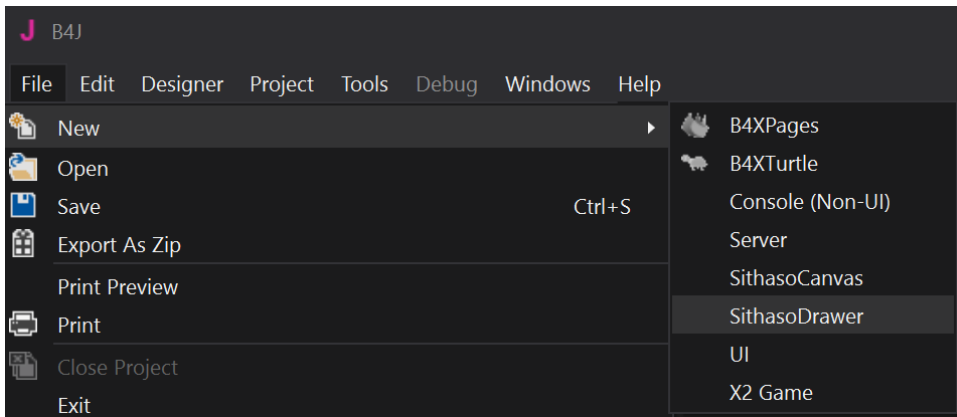
As you saw previously, both these templates provide different structured apps. Let's create an app using these.

Creating a Project

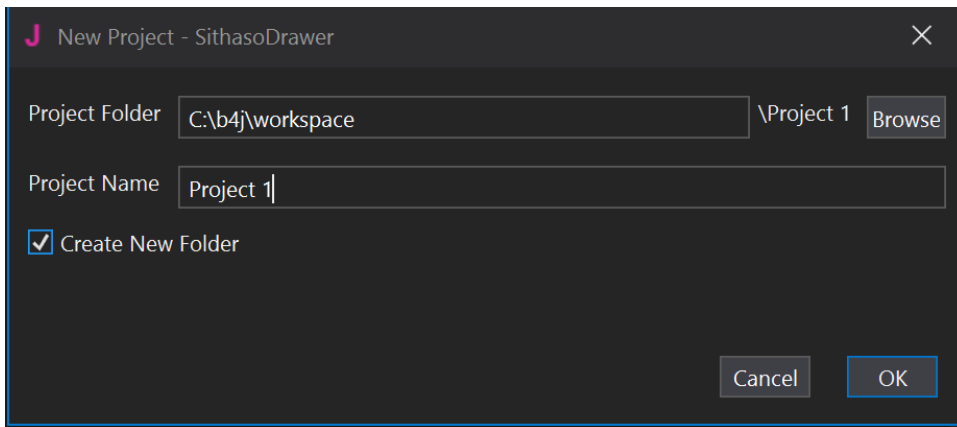
1. Start B4J.
2. Click File

3. Click New

4. Click SithasoDrawer / SithasoCanvas. SithasoDrawer is a 3-page project whilst SithasoCanvas has no additional pages.



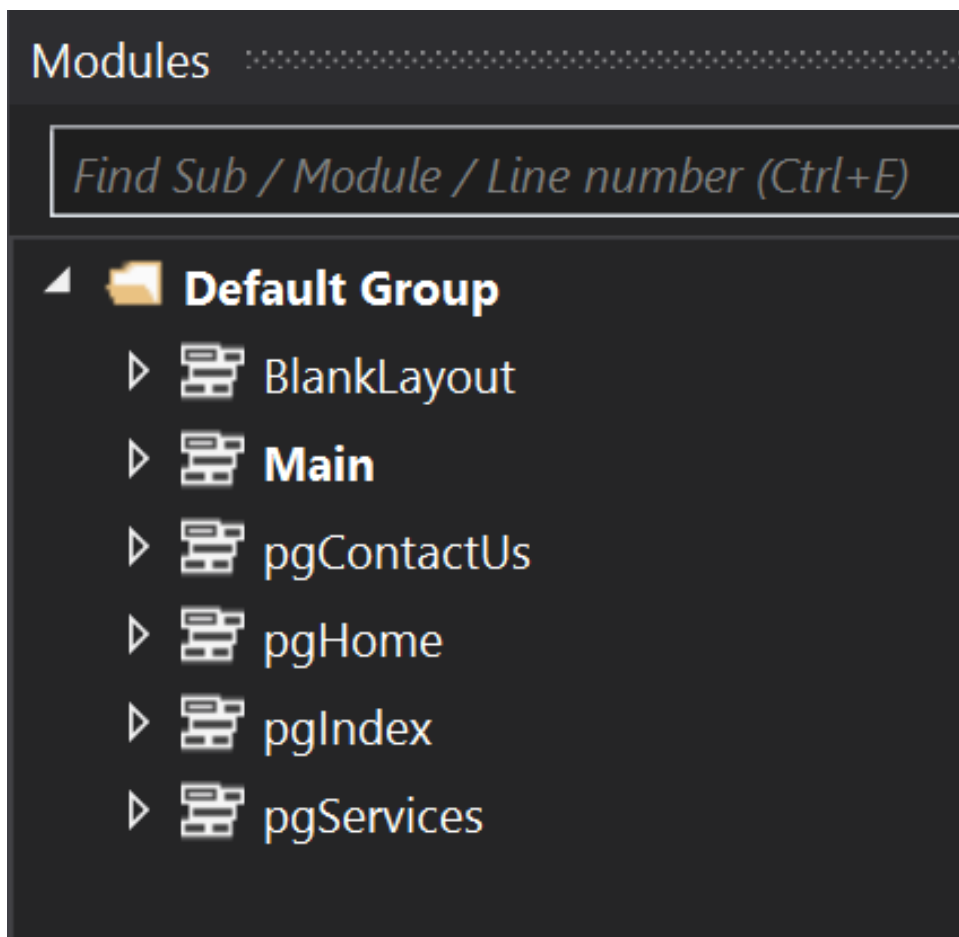
A prompt screen will appear. Type in the Project Name and click Ok



The main starting point of the app when it runs is the **Main** code module. As soon as you press F5 to run the app, the **AppStart** sub routine will fire. This will then fire the **BANano_Ready** method.

Banano_Ready executes code **pgIndex.Initialize**, which is where our app structure is built. This involves building our pages, the navigation bar, drawers and other things. One can then use these templates to add more pages to the app and the provided templates will provide guidance.

Basically, in the code modules, we have a BlankLayout template and other pages. The **BlankLayout** is used to create other pages. You can double click any of the pages to activate it.



The structure of each page is almost the same. These methods / subs are COMPULSOY

BuildPage - code to build the HTML of the page

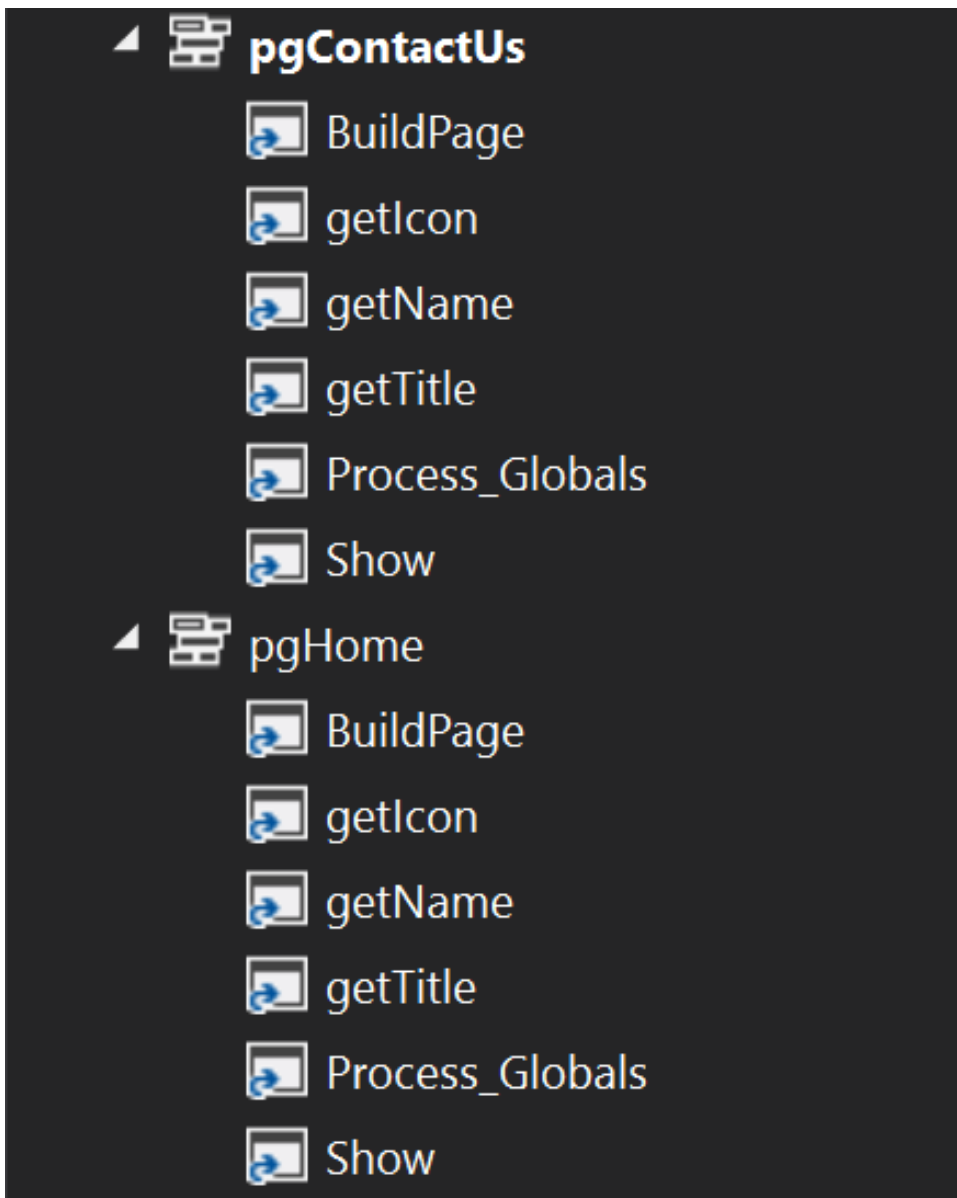
getIcon - the icon you specify for the page.

getName - the name you specify for the page

getTitle - the name you specify for the page

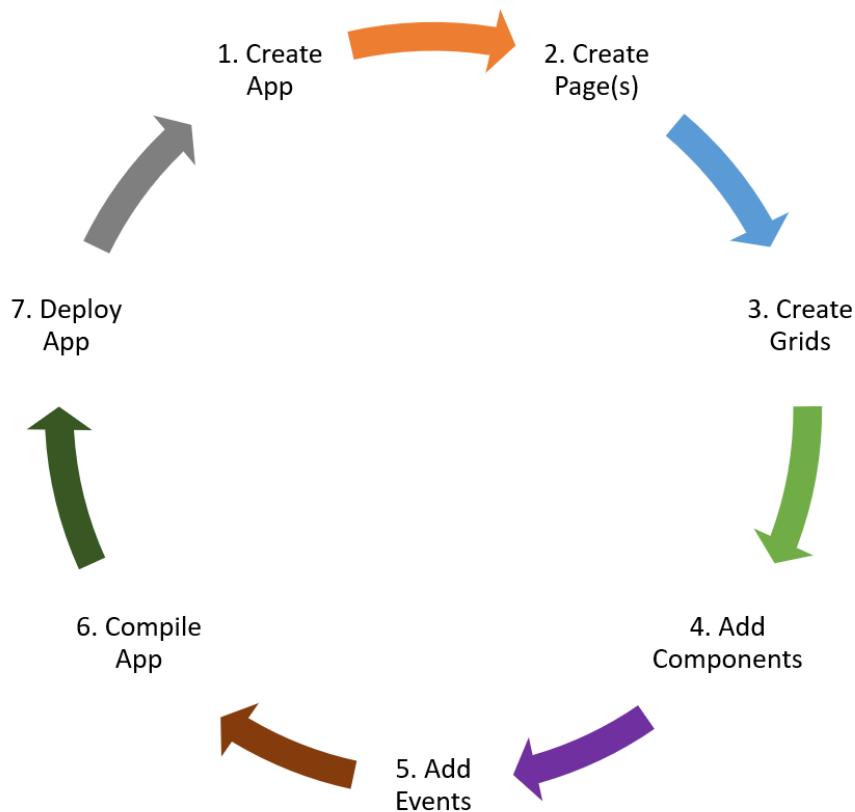
Process_Globals - definition of global variables for the page

Show - method to show the page



You can see the [Creating a Page](#) section on how to add Pages to your project.

Anyway, in creating each of our web-apps, we will follow this methodology.



One can use the abstract designer to place components or create them via code. The main controller when it comes to our app is the pgIndex code module. In it:

1. The base layout is loaded. This might have the main navigation bar and drawer of your app.
2. Other pages added on the app are linked to the drawer in this module.
3. The toggling of the drawer is done via the hamburger on this page.
4. Other nav-bar functionality can be added on this page.
5. The footer / bottom nav of your app can be added on this page.

If for example a page you add will be accessed via the drawer, it needs to be added in this method in pgIndex.

```

34 'define the menu items fo daw
35 Sub CreatedrawerMenu
36   'clear the menu
37   appdrawer.Clear("")
38   'add a page link to the drawer
39   appdrawer.AddItemPage(pgHome)
40   appdrawer.AddItemPage(pgServices)
41   appdrawer.AddItemPage(pgContactUs)
42 End Sub

```

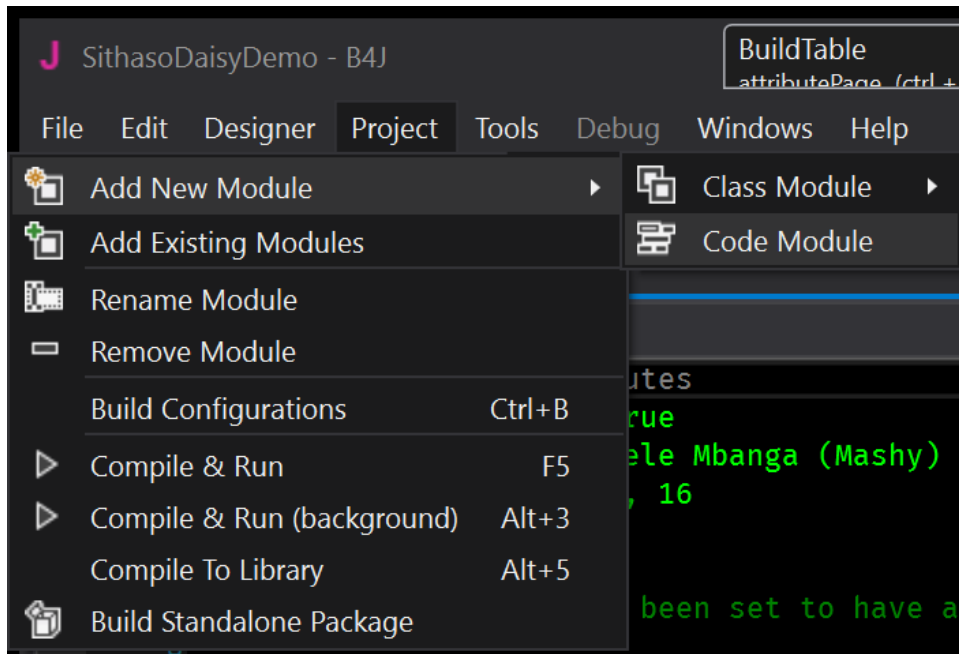
You can explore the SithasoDemo source code on how most of this coding was done for more understanding.

Creating a Page

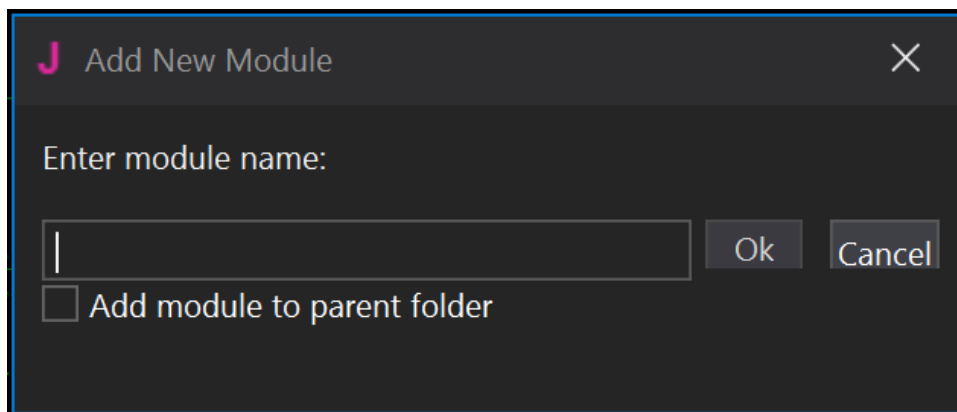
With your b4j project opened. If not opened, double click the **.bjl** file of your project.

Step 1 - Creating a new Code Module

1. Click on **Project** in the Menu
2. Click **Add New Module**
3. Click on **Code Module**



4. Type in the code module name and click Ok. The name should not have spaces or special characters.



We typed in **demoADLottiePlayer** as a code module name (example), the code module is then created.

```

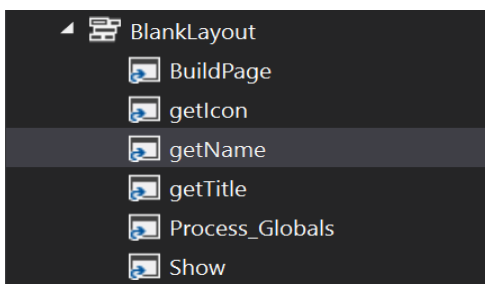
1 | Static code module
2 | Sub Process_Globals
3 |     Private fx As JFX
4 | End Sub
5 |

```

Now what we need to do is copy the page template to this code module

Step 2 - Copying the Page Template code from "BlankLayout"

In the Modules tab, locate the "BlankLayout" code module. This has the structure of the code needed for any page you can create in the app. Double click the Module to activate it.



```

1 | '***** DO NOT DELETE OR CHANGE THIS FILE *****
2 | #IgnoreWarnings:12, 9
3 | Sub Process_Globals
4 |     'this is the name of the page
5 |     Public name As String = "adblank"
6 |     Public title As String = "AD Blank"
7 |     Public icon As String = "fa-solid fa-swatchbook"
8 |     'this variable holds the page controller
9 |     Public page As SDUIPage
10 |     'this variable holds reference to the app
11 |     'usually for constants and other things
12 |     Public app As SDUIApp
13 |     'the variable referencing banano lib
14 |     Private banano As BANano 'ignore
15 | End Sub
16 |
17 | 'sub to show the page
18 | Sub Show(duiapp As SDUIApp) 'ignore
19 |     'get the reference to the app
20 |     app = duiapp
21 |     banano.LoadLayout(app.PageViewer, "adblanklayout")
22 |     'build the page, via code or loadlayouts
23 |     BuildPage
24 | End Sub
25 |

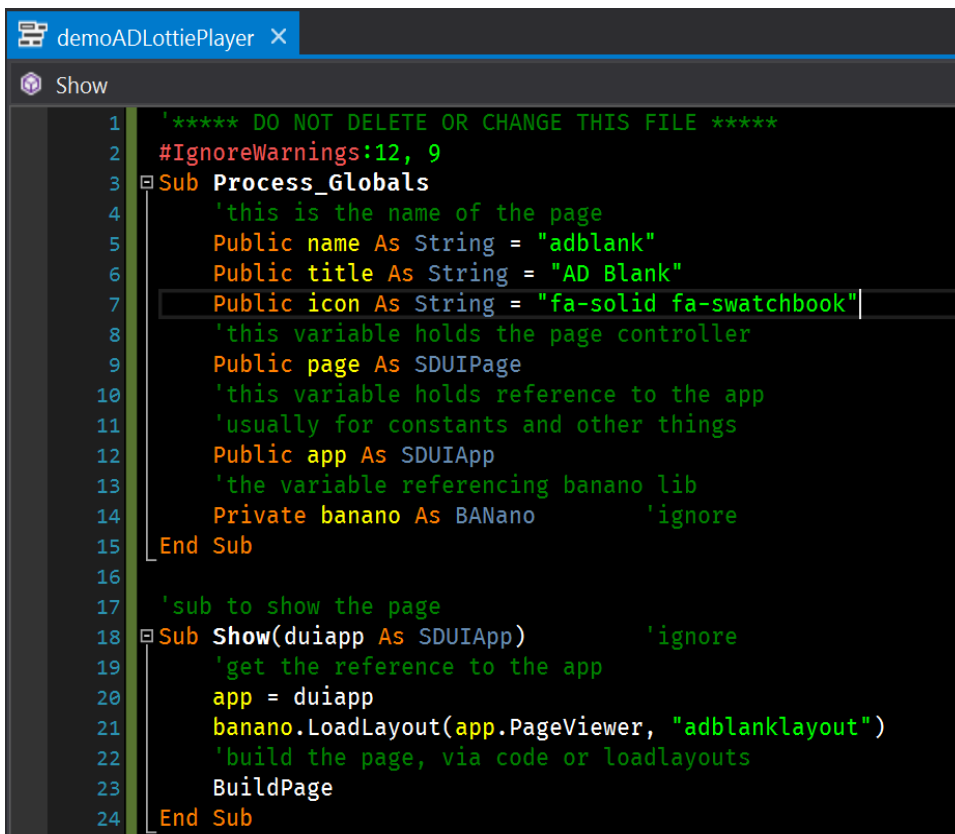
```

Select and copy all this code (Ctrl + A) as is to the newly created code module. Do not change anything on the BlankLayout code module. Paste the code to our new code module.

Step 3 - Giving the Page a Name, Title, Icon & Layout to load.

On the newly created code module, we need to make the page **unique**. To do this we will change 4 items in it on the code we pasted. This data is compulsory per page in your WebApp.

1. *name* - change the name string from "adblank" to be your unique page name.
2. *title* - change the title of the page from "AD Blank" to be something more catchier.
3. *icon* - change the icon also to be unique. FontAwesome is the default integrated font family. You can search for an icon there, <https://fontawesome.com/>
4. In the **Show** sub-routine, change the layout name from adblanklayout to be your unique layout name. Usually, I just use the name + "layout" here.



```

1  '***** DO NOT DELETE OR CHANGE THIS FILE *****
2  #IgnoreWarnings:12, 9
3  Sub Process_Globals
4      'this is the name of the page
5      Public name As String = "adblank"
6      Public title As String = "AD Blank"
7      Public icon As String = "fa-solid fa-swatchbook"
8      'this variable holds the page controller
9      Public page As SDUIPage
10     'this variable holds reference to the app
11     'usually for constants and other things
12     Public app As SDUIApp
13     'the variable referencing banana lib
14     Private banana As BANano 'ignore
15 End Sub
16
17 'sub to show the page
18 Sub Show(duiapp As SDUIApp) 'ignore
19     'get the reference to the app
20     app = duiapp
21     banana.LoadLayout(app.PageViewer, "adblanklayout")
22     'build the page, via code or loadlayouts
23     BuildPage
24 End Sub

```

As an example, below, we have updated the code for our page to be like this:

```

1  '***** DO NOT DELETE OR CHANGE THIS FILE *****
2  #IgnoreWarnings:12, 9
3  Sub Process_Globals
4      'this is the name of the page
5      Public name As String = "adlottieplayer"
6      Public title As String = "AD Lottie Player"
7      Public icon As String = "fa-solid fa-play"
8      'this variable holds the page controller
9      Public page As SDUIPage
10     'this variable holds reference to the app
11     'usually for constants and other things
12     Public app As SDUIApp
13     'the variable referencing banana lib
14     Private banana As BANano 'ignore
15 End Sub
16
17 'sub to show the page
18 Sub Show(duiapp As SDUIApp) 'ignore
19     'get the reference to the app
20     app = duiapp
21     banana.LoadLayout(app.PageViewer, "adlottieplayerlayout")
22     'build the page, via code or loadlayouts
23     BuildPage
24 End Sub

```

Note the following:

1. The code module name has been named in such a way that we know which page it is.
2. The name of the page on the code, title, icon and layout name on the **BANano.LoadLayout** code line has been named clearly and properly.

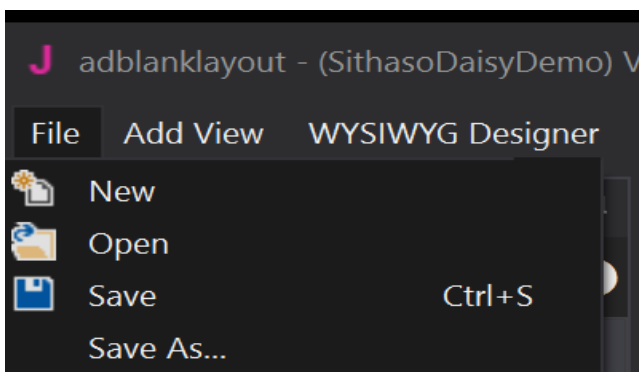
Step 4 - Copying the Page Layout from "adblanklayout"

Now we need to ensure that the view/layout of our page exist. We will create it from an existing .bjl file.

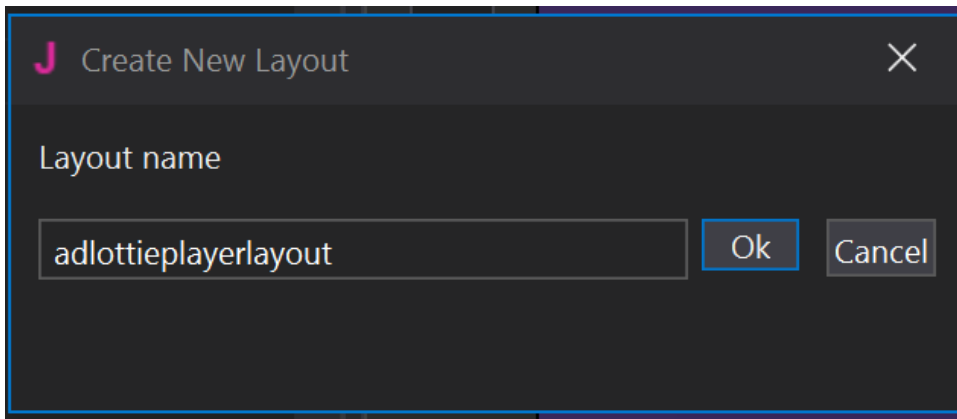
Copy the layout name e.g., "adlottieplayerlayout" from the show sub.
In the Files tab, locate the adblanklayout.bjl file and open it.



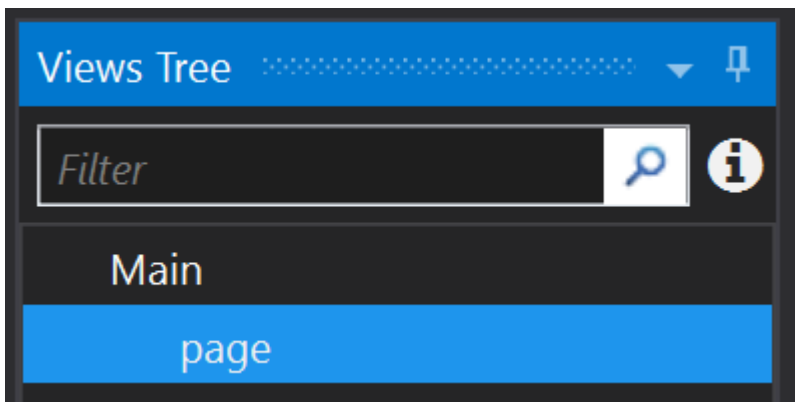
Click on File > Save As



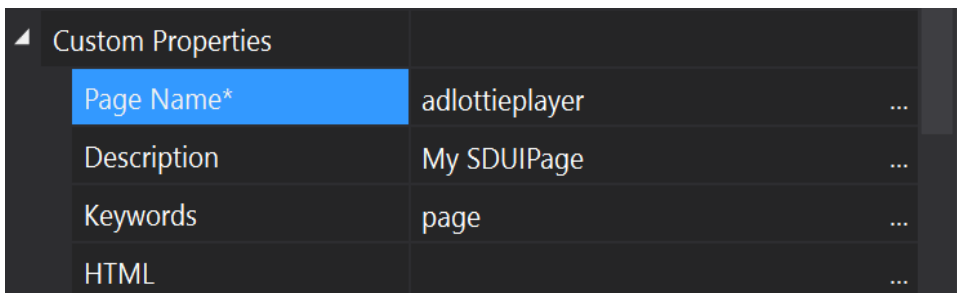
In the prompt that follows, paste the layout name you copied and click Ok.



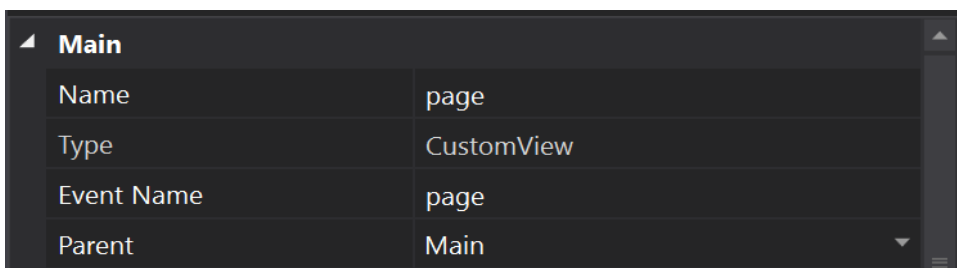
Now, on the Views Tree, click "page", to activate the custom view.



In the property bag for the page, change the **"Page Name"** to EXACTLY MATCH the **name** you used in the code module. For example.



This part below should remain UNCHANGED, i.e Name should ALWAYS be page



Save you changes.

Creating a Grid

A grid is used to place components on your page. These could be textboxes, images, labels etc. Before the components are placed, this grid needs to be defined. SithasoDaisy has a grid system that is based on the Bootstrap grid. This has "invisible" rows and columns. Each row can have 12 columns. Columns can span to a number of other columns. Below is a grid with 15 rows and in each row, we have 1 or more columns that span.

R1C1											
R2C1										R2C2	
R3C1										R3C2	
R4C1										R4C2	
R5C1								R5C2			
R6C1								R6C2			
R7C1						R7C2					
R8C1						R8C2					
R9C1					R9C2						
R10C1				R10C2							
R11C1			R11C2								
R12C1		R12C2									
R13C1						R13C2				R13C3	
R14C1			R14C2							R14C3	
R15C1		R15C2							R15C3		

Let's explain how this structure was created by looking at each row, based on the following b4x code.

```

50  page.Root.AddRows1.AddColumns12
51  page.Root.AddRows1.AddColumns11.AddColumns1
52  page.Root.AddRows1.AddColumns10.AddColumns2
53  page.Root.AddRows1.AddColumns9.AddColumns3
54  page.Root.AddRows1.AddColumns8.AddColumns4
55  page.Root.AddRows1.AddColumns7.AddColumns5
56  page.Root.AddRows1.AddColumns6.AddColumns6
57  page.Root.AddRows1.AddColumns5.AddColumns7
58  page.Root.AddRows1.AddColumns4.AddColumns8
59  page.Root.AddRows1.AddColumns3.AddColumns9
60  page.Root.AddRows1.AddColumns2.AddColumns10
61  page.Root.AddRows1.AddColumns1.AddColumns11
62  page.Root.AddRows1.AddColumns6.AddColumns3.Addcolumns3
63  page.Root.AddRows1.AddColumns2.AddColumns8.Addcolumns2
64  page.Root.AddRows1.AddColumns1.AddColumns9.Addcolumns2
65  'build the grid
66  page.Root.BuildGrid

```

Each row/column is allocated a unique id when the grid is built. Let's look at the rows.

R1 - In this row we have a row with 1 column named R1C1. This was created with **.AddRows1.AddColumns12**. This adds 1 row and within that row add a column that will span 12 spaces. So, all calls that start with AddRows? are for adding rows and all the ones with AddColumns? are for adding columns within a row.

R2 - In this row we have a row with 2 columns named R1C1 and R1C2. This was created with `.AddRows1.AddColumns11.AddColumns1` - this adds 1 row and within that row two columns. The first column R1C1 spans 11 spaces whilst the other column spans 1 space. $11+1=12$.

R3 - Here we have used `AddColumns10.AddColumns2`. This makes 1 column span 10 and the other 2. $10+2=12$.

etc

...

R13 - In this row we have 3 columns. We have used. `AddColumns6.AddColumn3.AddColumns3`. $6+3+3=12$

If the sum of columns exceeds 12, the columns that exceed 12 will be wrapped to the following row.

Adding Components to the Grid

To add components to the grid, we use a row & column reference. We call this a cell. For example, to refer to row 10 column 2, we will use **Cell(10, 2)**

In this example below, a button is added to **Cell(1, 1)** that is **R1C1**.

Round Full

```
Dim btnRounded As SDUIButton = page.Cell(1, 1).AddButton("btn2", "Round Full")
btnRounded.NormalCase = True
btnRounded.RoundedFull
```

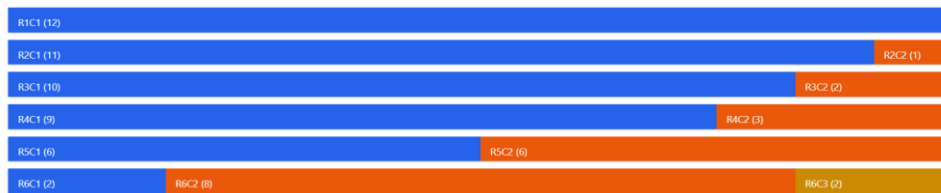
Components can also be designed using the abstract designer. Items in the abstract designer are placed inside others, establishing a parent child relationship. To create a grid in the abstract designer, one will use **SDUIRow** and **SDUIColumn** components.

Below is an example of a grid created with rows and columns.

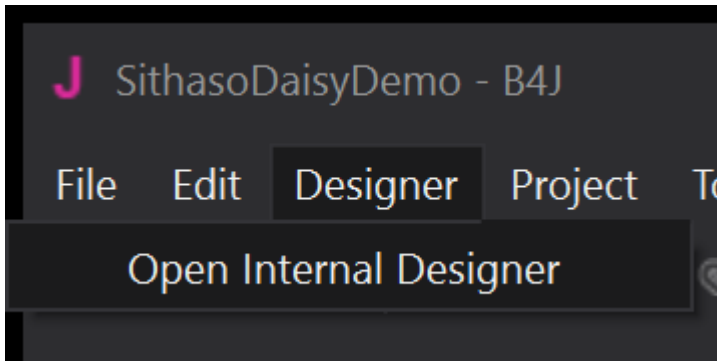
Abstract Designer



Outcome



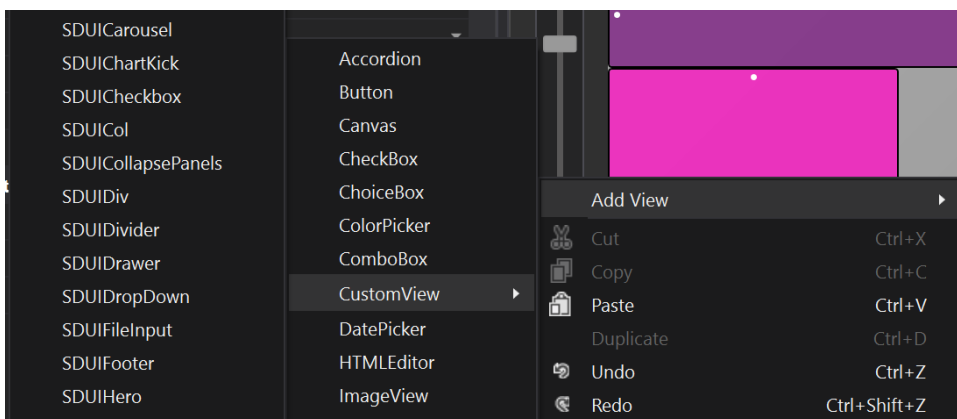
One can access the abstract designer by clicking Designer in the menu.



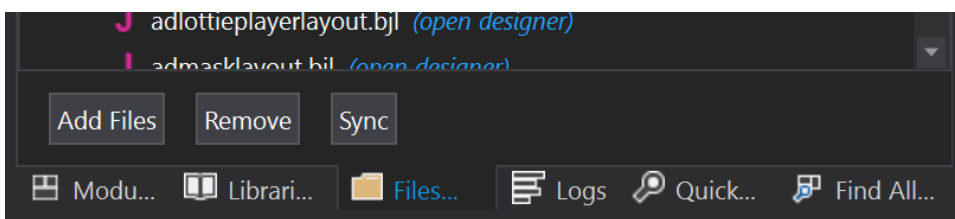
Adding Components to the Abstract Designer Layout

The components in the design above were added by:

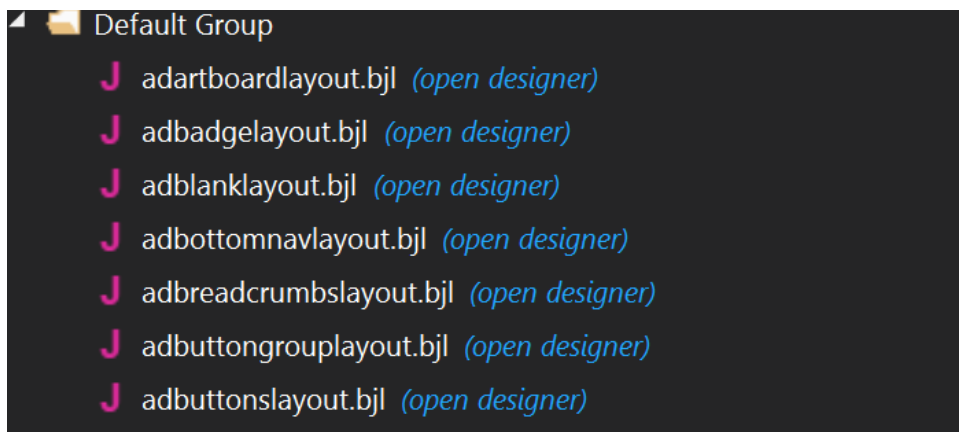
1. Right click on the empty abstract design canvas.
2. Click Add View
3. Click CustomView, then select an SDUI component from the list.



At the bottom right part of the b4j IDE, you will find **tabs**. These provide access to the resources of your project.



In the files tab, you will notice there a **bjl** files (SithasoDemo). These are the abstract designer files. You can click on Open Designer to open that file and drop elements to it.



Please refer to the [New to B4x?](#) section for more details about how the whole b4x ecosystem works as that is beyond the scope of this eBook.