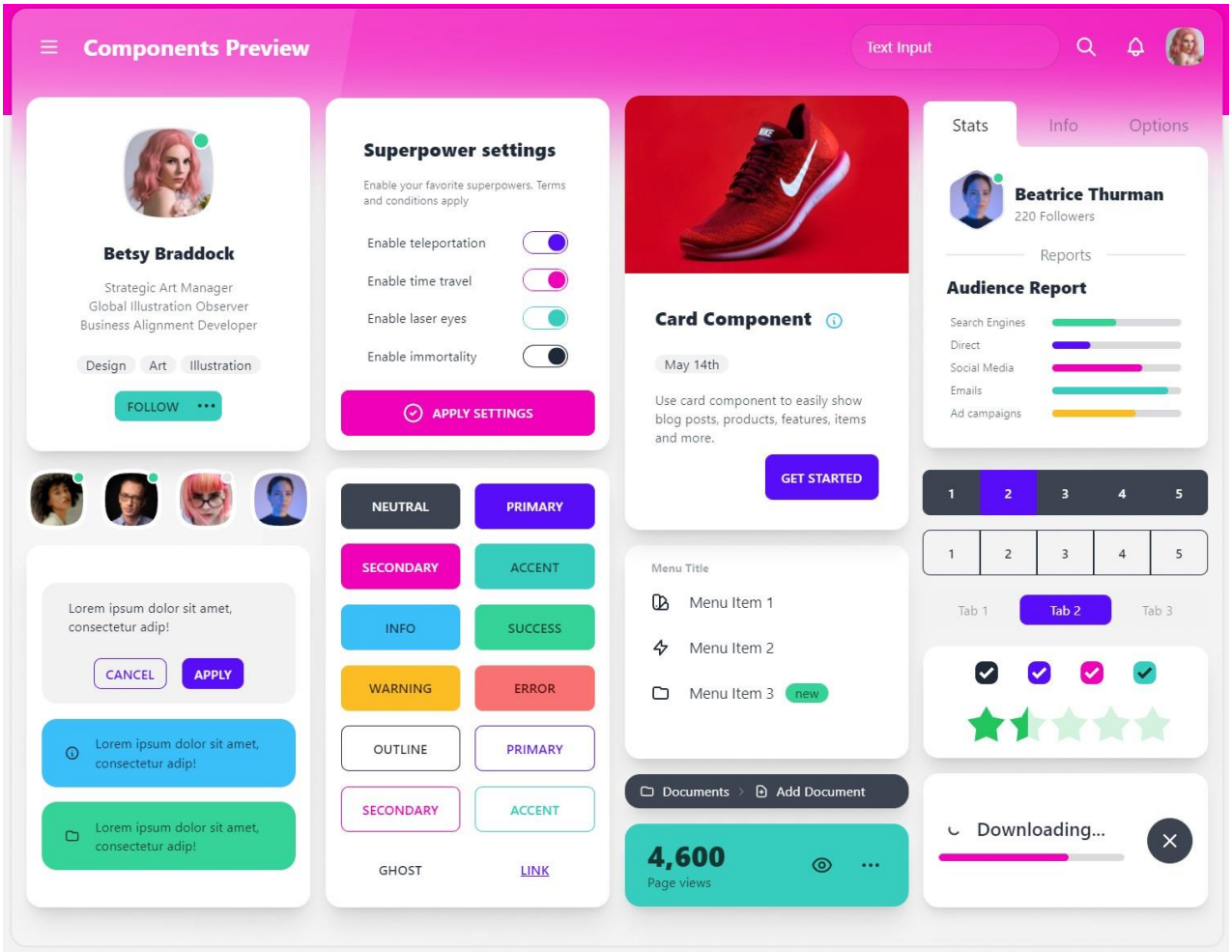
TailwindCSS WebApps



using B4X

By Anele 'Mashy' – November 2022 – Draft 0.01

Table of contents

Introduction 3
Getting Started 5
Project Templates 11
    SithasoDrawer 11
    SithasoCanvas 13
New to B4x? 15
Creating WebApps 16
    Creating a Page 16

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.

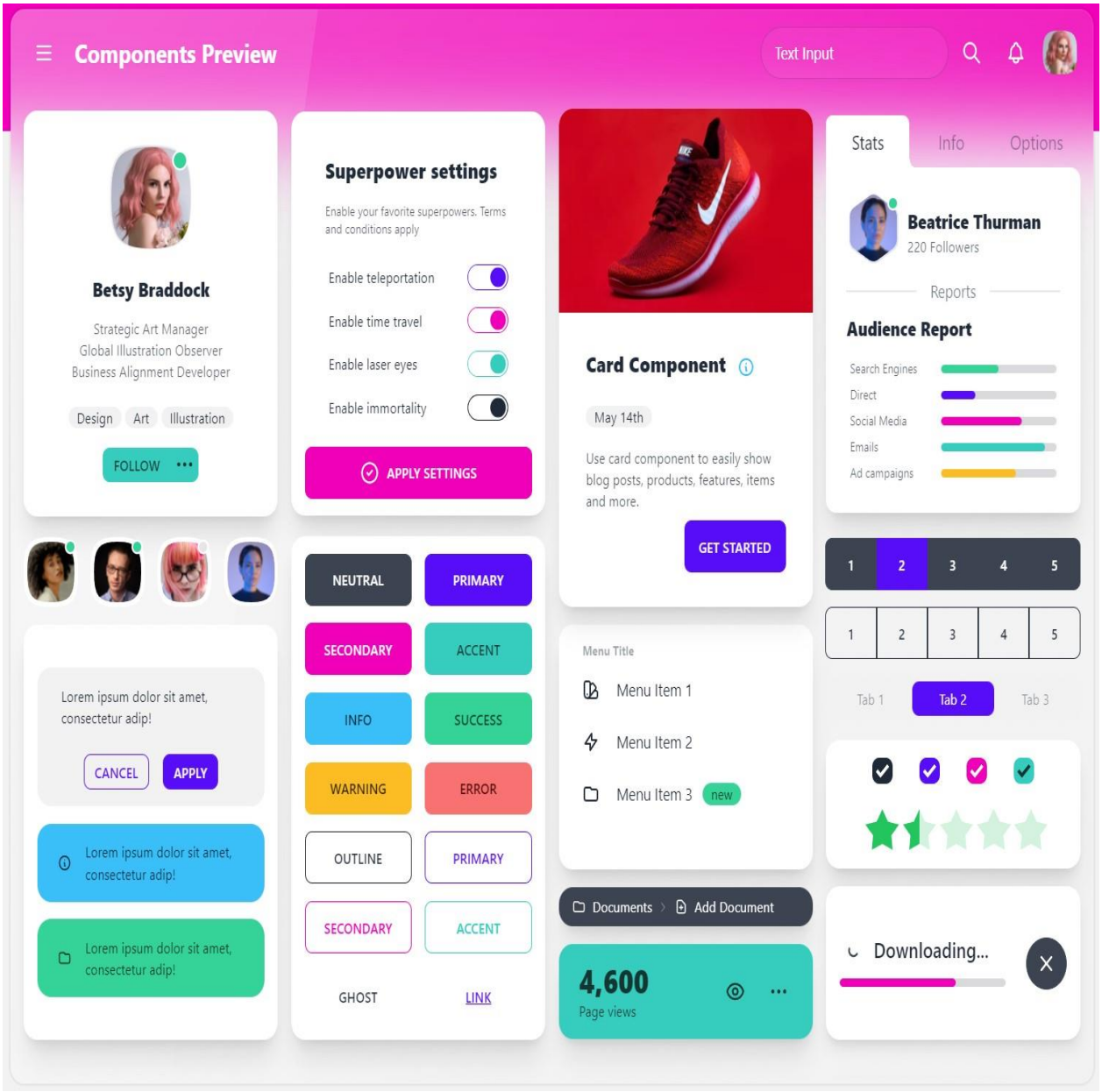
When it comes to developing anything that works on the internet browser, whether it is a WebApp or a WebSite, one has to use HTML (**H**yper **T**ext **M**arkup **L**anguage), CSS (**C**ascading **S**tyle **S**heet) and JavaScript (a dynamic programming language used for web development).

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.

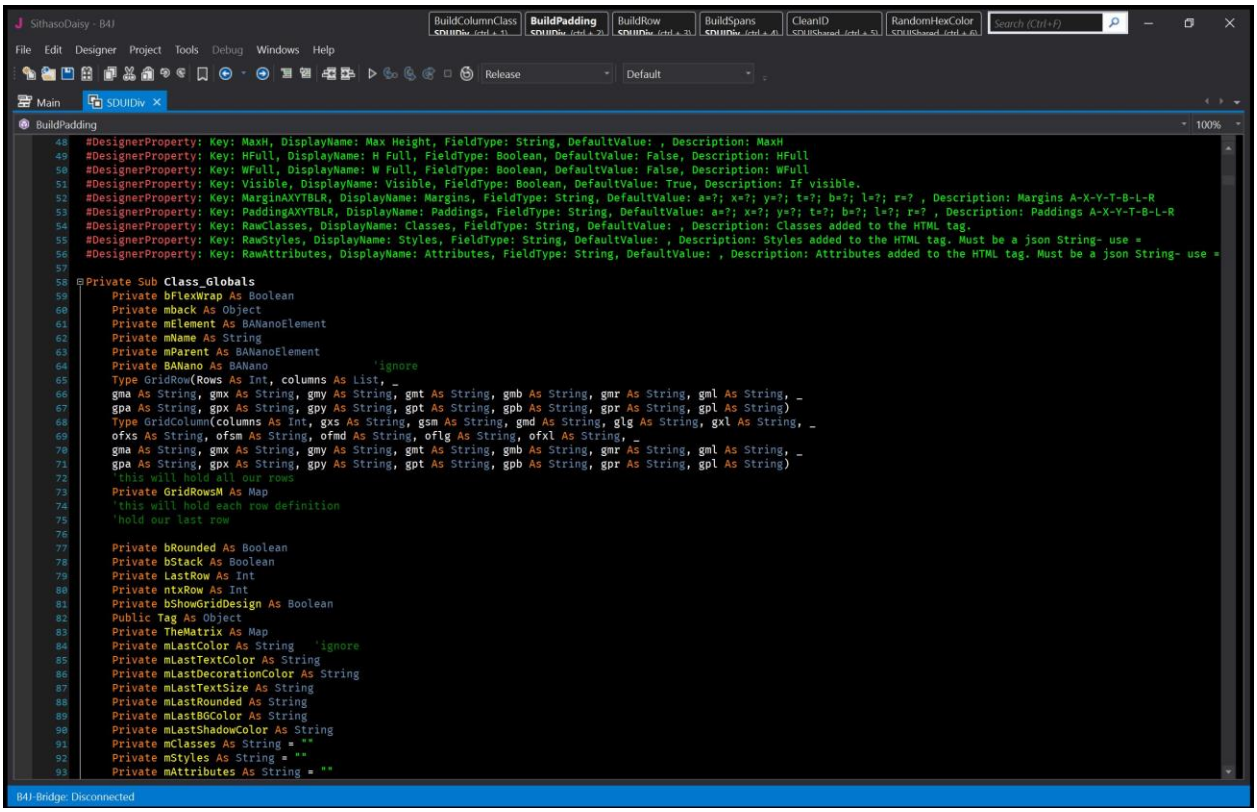


With a 64-bit Windows PC, lets get started.

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 menus, a toolbar, the coding area and a module listing area, to mention the few.

1. The B4J IDE



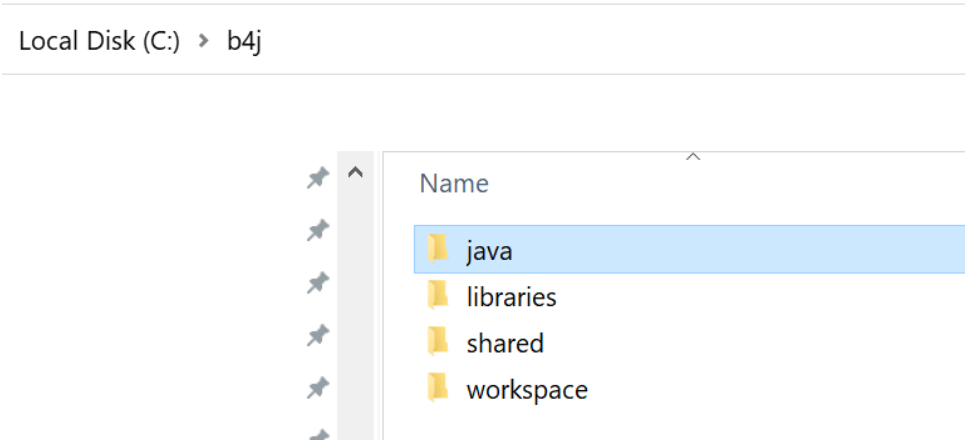
2. Creating Folders

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

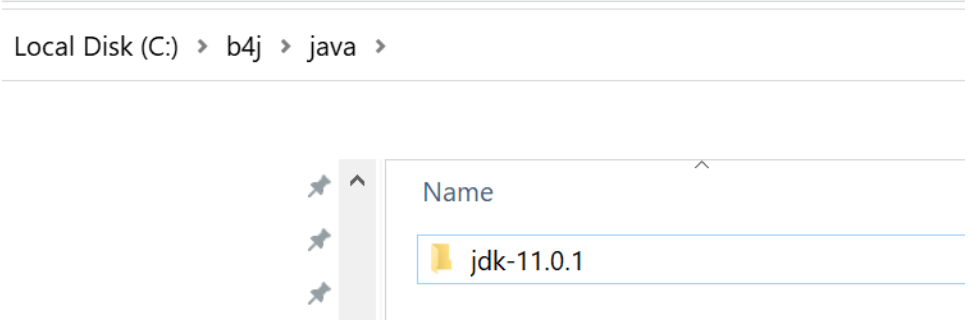
This should look like:



3. Downloading and installing B4J

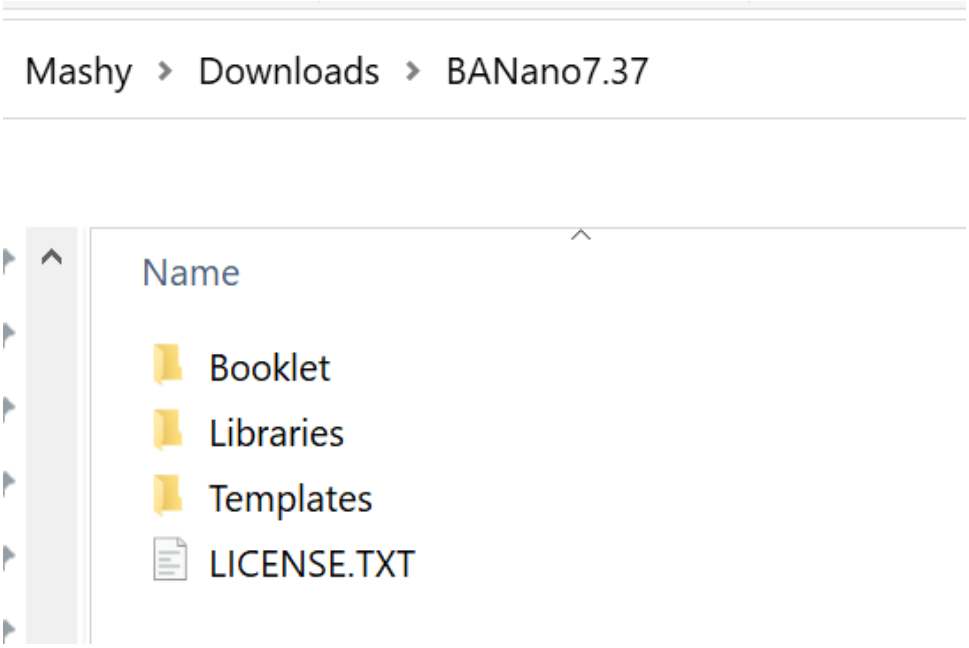
1. Head over to Anywhere Software Website and download b4j. You can click here to do that.
2. Click on Download B4J Full Version (64-BIT). After you download, ensure you install the application.

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



4. Download BANano

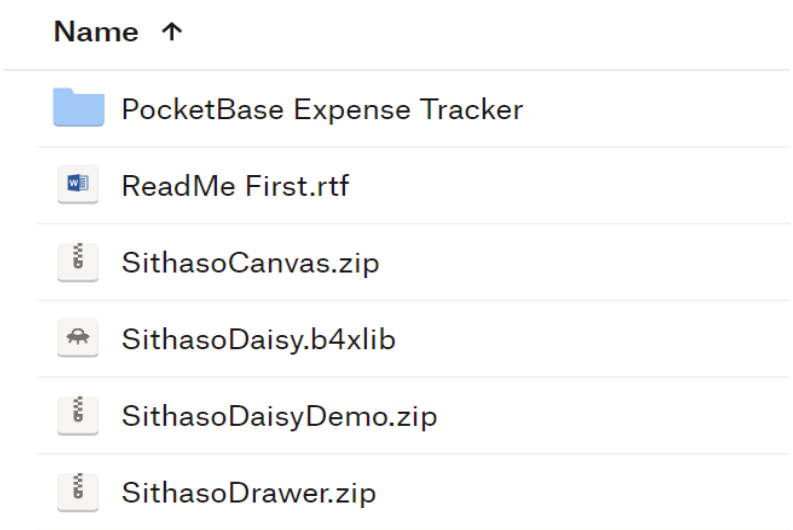
You will also need BANano. This is a b4j plugin. Click here to get it and unpack it. It should have this content.



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

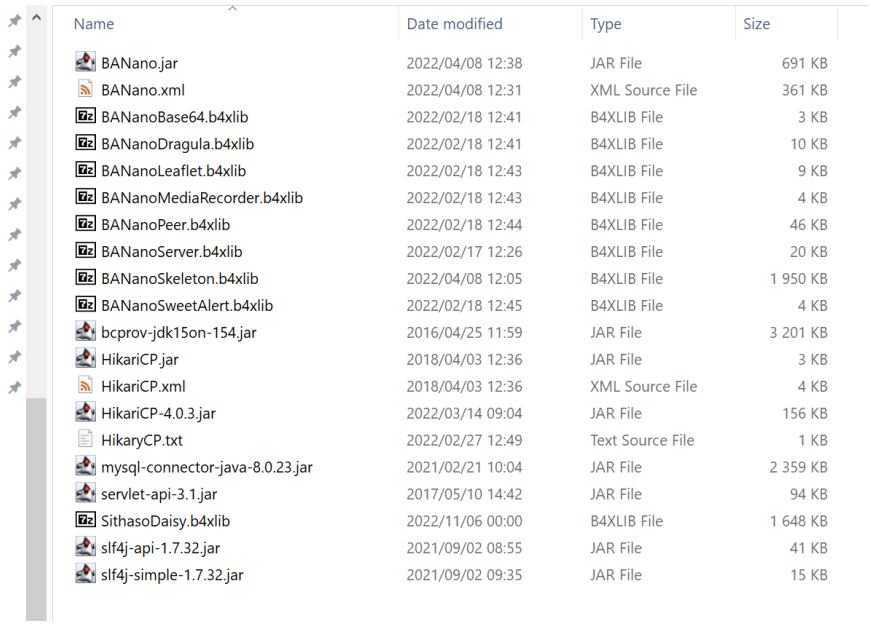
5. Download SithasoDaisy

From your **SithasoDaisy** download.



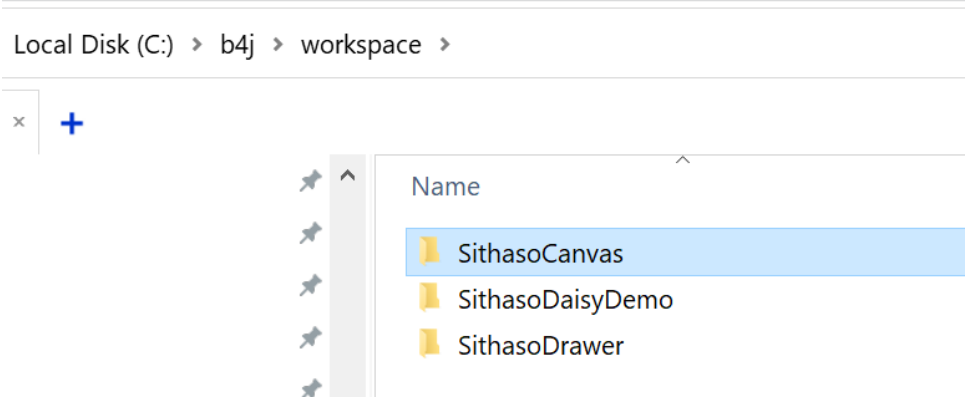
5.1 Copy the **SithasoDaisy.b4xlib** to **c:\b4j\libraries.** You now should have.

‹ (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
HikaryCP.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

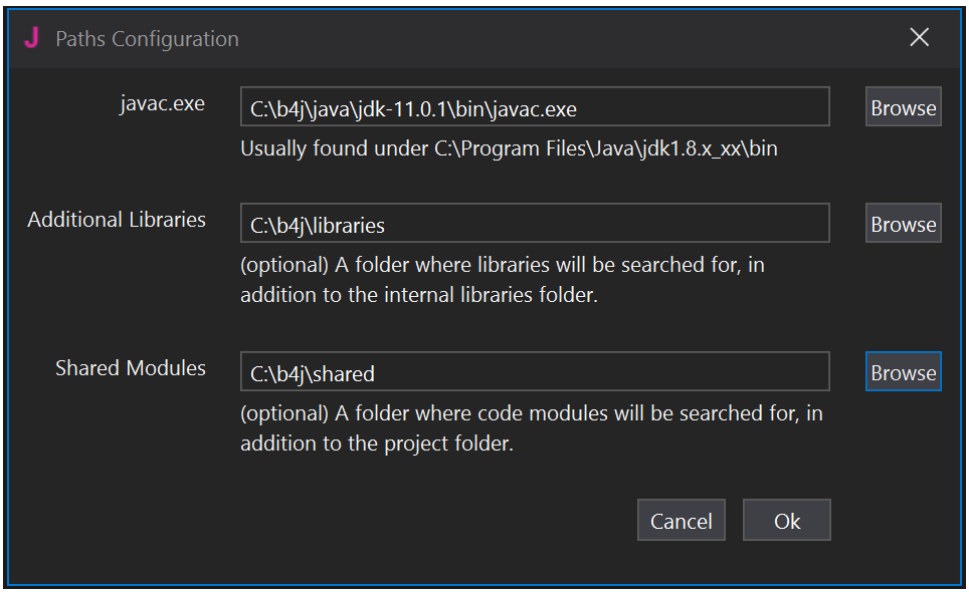
5.1 Unpack **SithasoCanvas.zip**, **SithasoDaisyDemo.zip** and **SithasoDrawer.zip** to **c:\b4j\workspace.** You now should have.



6. B4J Paths Configuration.

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

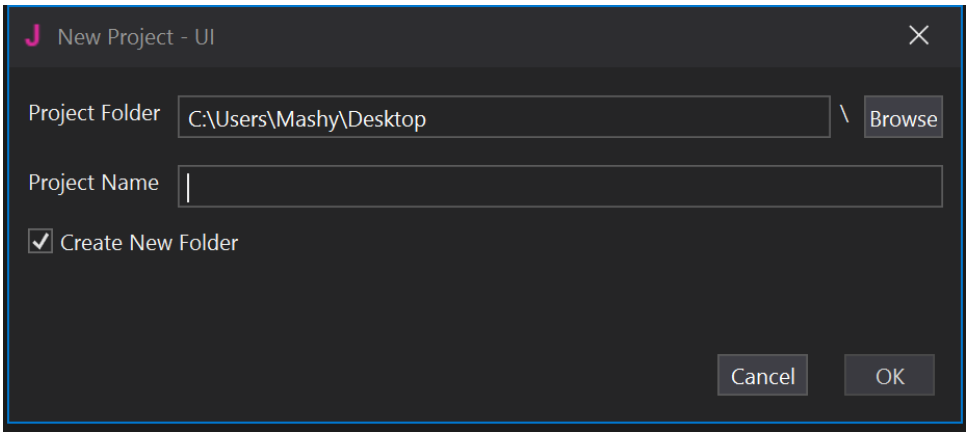
Click on the browse buttons to select the respective file and paths specified below.



Then click Ok, to save your configuration.

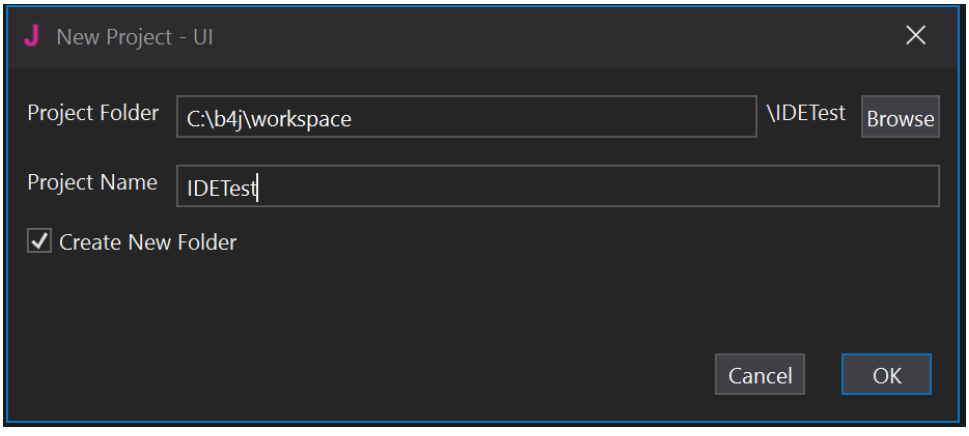
7. Testing B4J IDE readiness

Click on File > New > UI

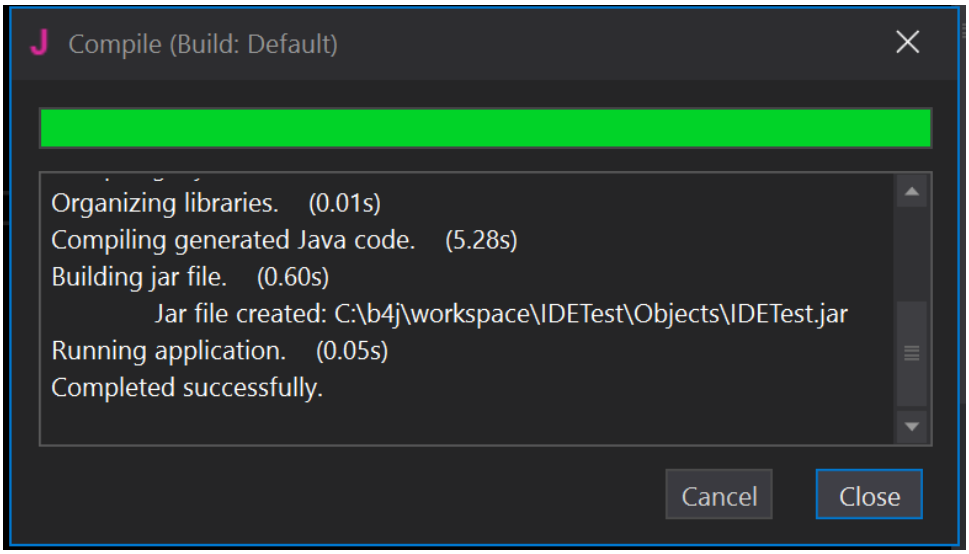


Click on Browse and ensure that the Project folder is **C:\b4j\workspace**

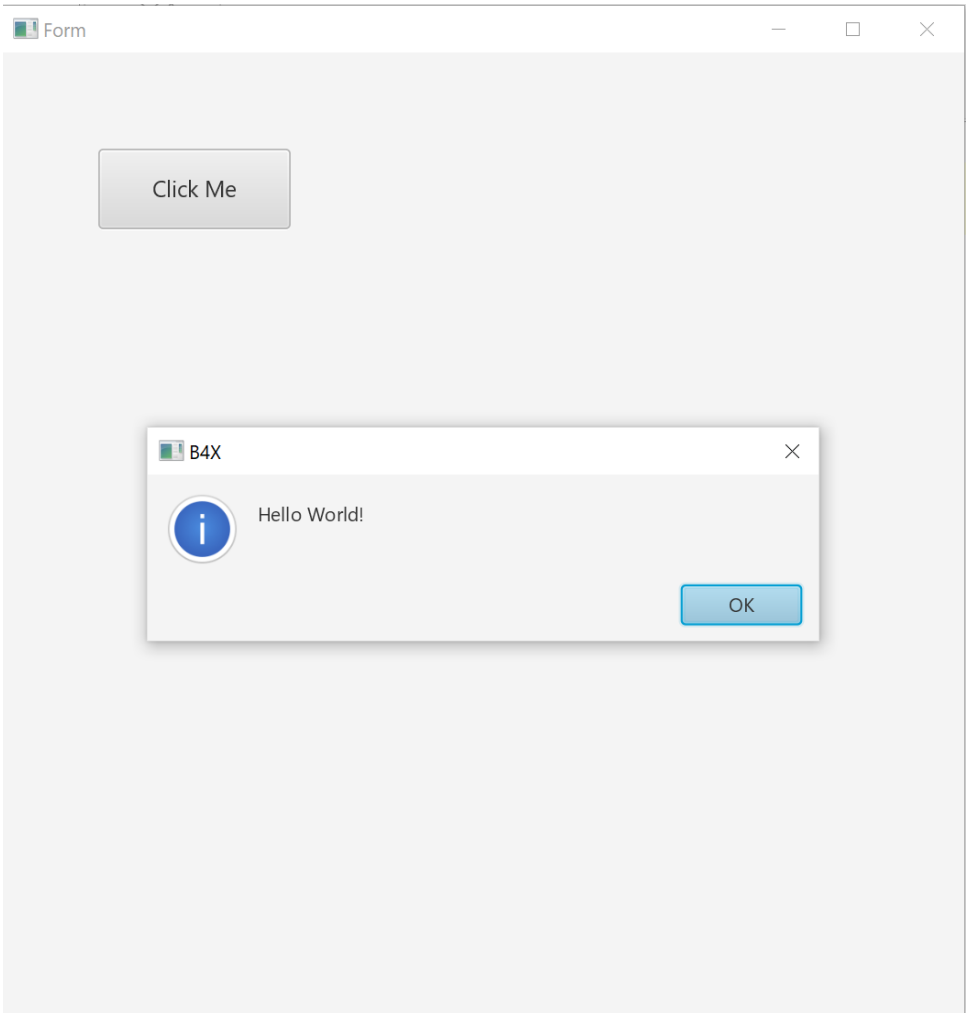
Type in a project name, for example, IDETest and click Ok.



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



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

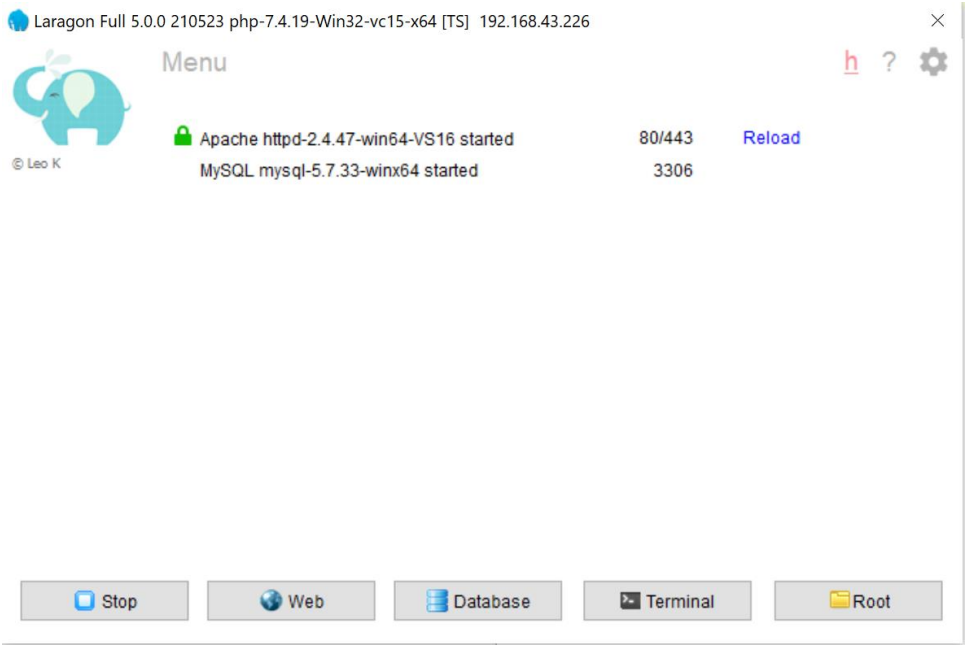


Congratulations, you have just run your first b4j developed java application with b4x. You can close the App and the IDE.

Let's 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.



Project Templates

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

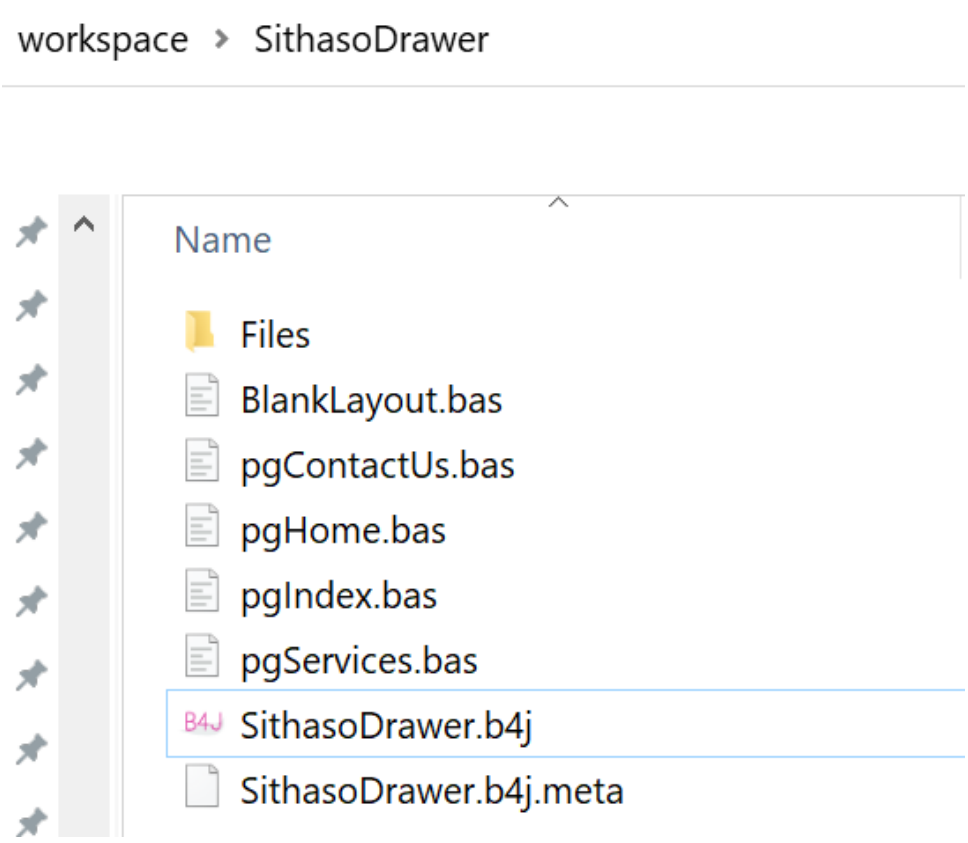
- Use SithasoDrawer as a base for projects with a top navigation bar and a drawer.
- Use SithasoCanvas as a base for projects that you will start from scratch.

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.**

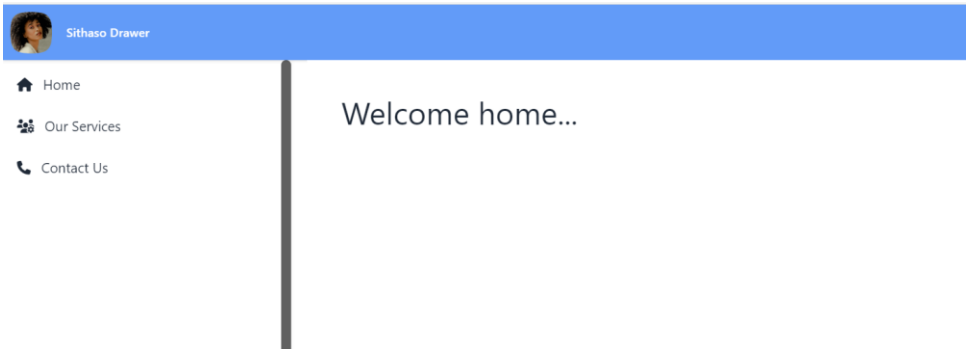
These need to be change to suit your needs if you are not using laragon and also using different port numbers.

SithasoDrawer

1. Open the c:\b4j\workpace\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



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

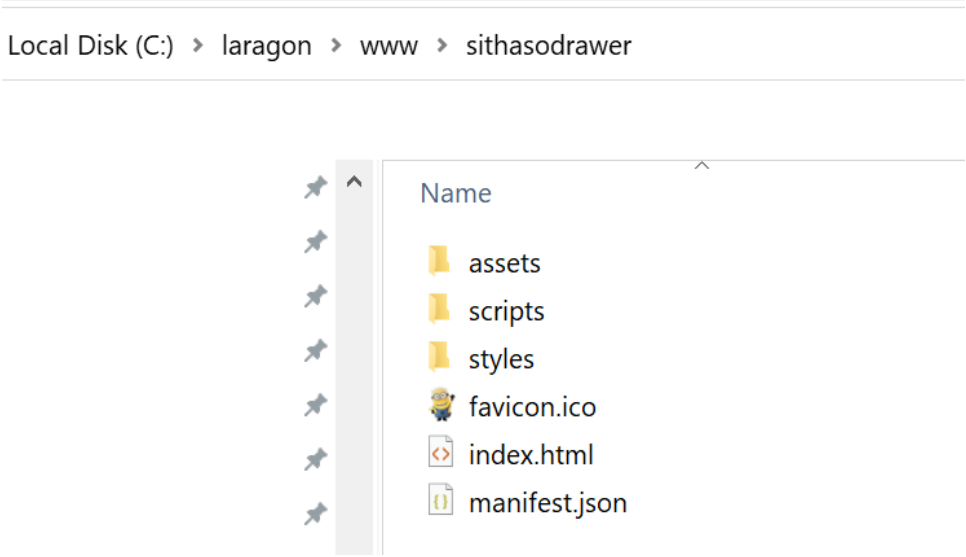


The name of this app is "**sithasodrawer**". This is defined in the **Main** code module.

```
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
```

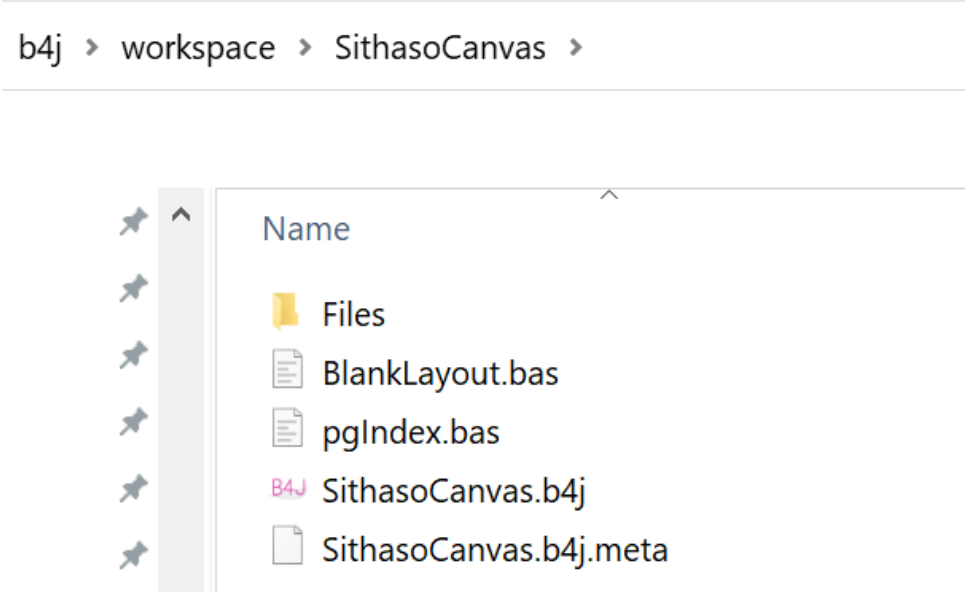
To access the transpiled source code (javascript, css and html) that resulted with what you see in the browser, head over to the **c:\laragon\www\sithasodrawer** folder.

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



SithasoCanvas

1. Open the c:\b4j\workpace\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



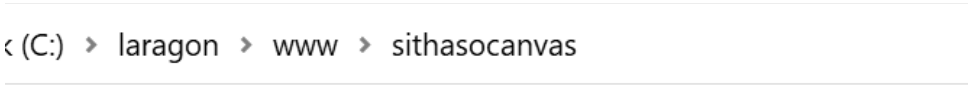
After compilation, you should see this app in action on your default webbrowser. Click the hamburger or menu.

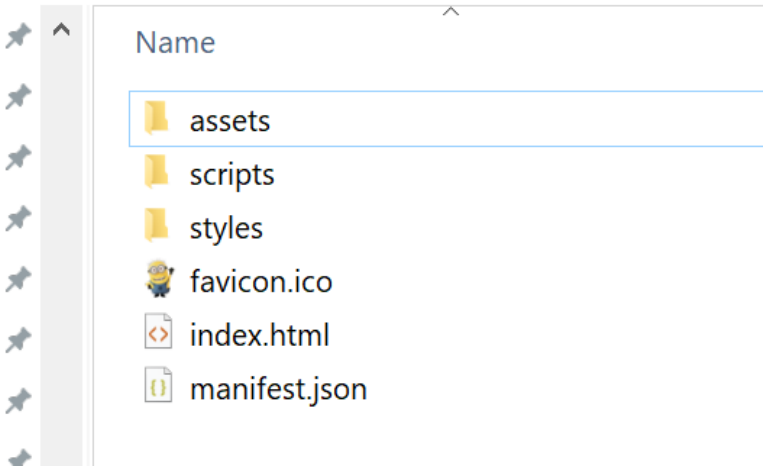
The name of this app is "**sithasocanvas**". This is defined in the **Main** code module.

```
#IgnoreWarnings:12, 15
Sub Process_Globals
    Public BANano As BANano 'ignore
    'the name of the application &
    'this is the folder on your development server.
    Public AppName As String = "sithasocanvas"
    Public AppTitle As String = "Sithaso Canvas"
    'whe the app should
    Private Publish As String = "C:\laragon\www"
    Public Version As String = "0.01"
    Public ServerIP As String
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 you public webserver when you are finihed developing you webapp.





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

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.

Creating WebApps

To create a webapp, use either SithasoDrawer or SithasoCanvas project as your base. You can copy the folder contents, rename the folder and also rename the .b4j file to your new project name. In the future this will be available from the IDE.

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 an possibly its icon.

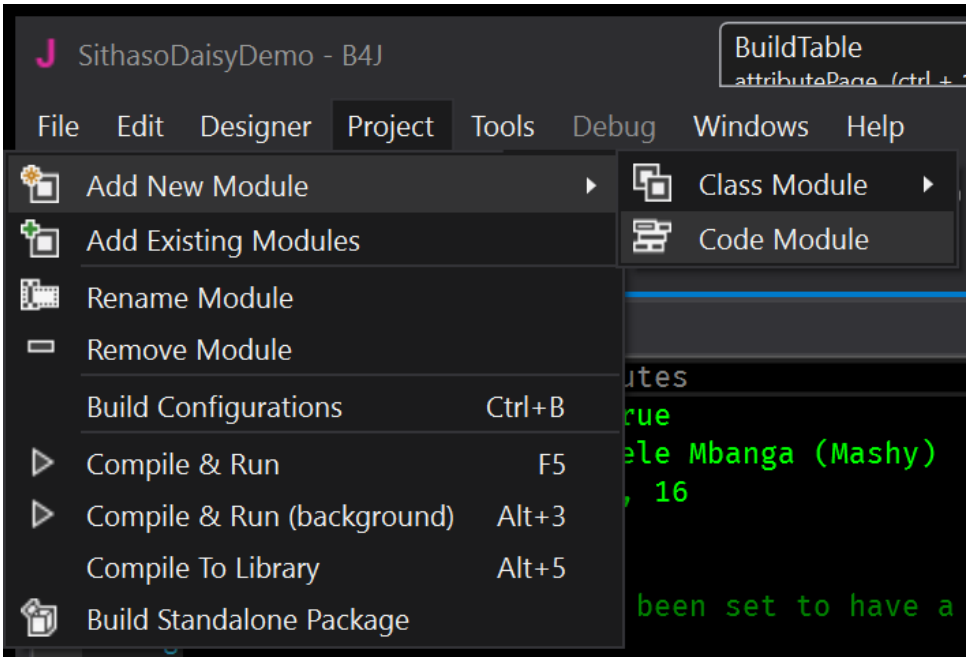
Creating a Page

Creating a Page

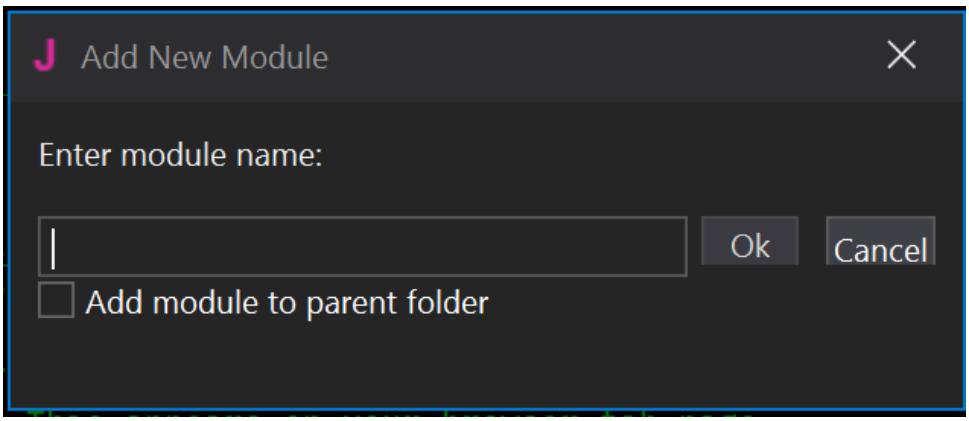
With your b4j project opened.

Step 1 - Creating a new Code Module

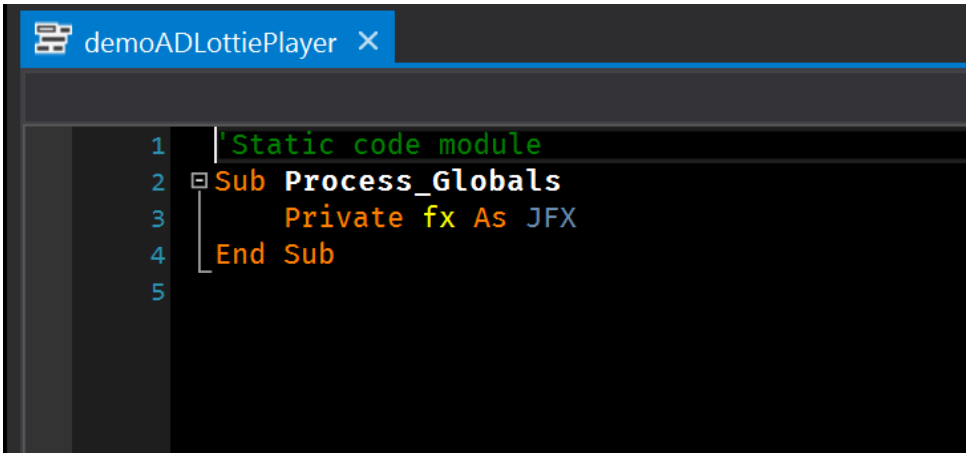
1. Click on **Project** in the Menu
2. Click **Add New Module**
3. Click on **Code Module**



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



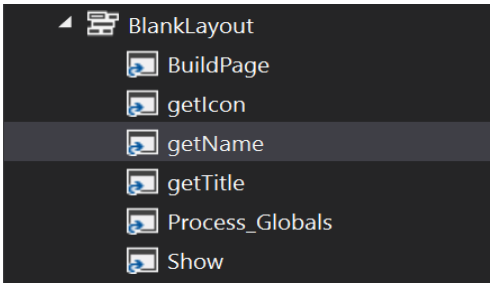
We typed in **demoADLottiePlayer** as a code module name (example), the code module is then created.



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.



BlankLayout

getIcon

```
'***** DO NOT DELETE OR CHANGE THIS FILE *****
#IgnoreWarnings:12, 9
Sub Process_Globals
    'this is the name of the page
    Public name As String = "adblank"
    Public title As String = "AD Blank"
    Public icon As String = "fa-solid fa-swatchbook"
    'this variable holds the page controller
    Public page As SDUIPage
    'this variable holds reference to the app
    'usually for constants and other things
    Public app As SDUIApp
    'the variable referencing banano lib
    Private banano As BANano        'ignore
End Sub

'sub to show the page
Sub Show(duiapp As SDUIApp)         'ignore
    'get the reference to the app
    app = duiapp
    banano.LoadLayout(app.PageViewer, "adblanklayout")
    'build the page, via code or loadlayouts
    BuildPage
End Sub
```

Select and copy all this code (Ctr + 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 you unique page name.
2. *title* - change the title of the page fom "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.

demoADLottiePlayer

Show

```
'***** DO NOT DELETE OR CHANGE THIS FILE *****
#IgnoreWarnings:12, 9
Sub Process_Globals
    'this is the name of the page
    Public name As String = "adblank"
    Public title As String = "AD Blank"
    Public icon As String = "fa-solid fa-swatchbook"
    'this variable holds the page controller
    Public page As SDUIPage
    'this variable holds reference to the app
    'usually for constants and other things
    Public app As SDUIApp
    'the variable referencing banano lib
    Private banano As BANano        'ignore
End Sub

'sub to show the page
Sub Show(duiapp As SDUIApp)         'ignore
    'get the reference to the app
    app = duiapp
    banano.LoadLayout(app.PageViewer, "adblanklayout")
    'build the page, via code or loadlayouts
    BuildPage
End Sub
```

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

demoADLottiePlayer

Process_Globals

```
'***** DO NOT DELETE OR CHANGE THIS FILE *****
#IgnoreWarnings:12, 9
Sub Process_Globals
    'this is the name of the page
    Public name As String = "adlottieplayer"
    Public title As String = "AD Lottie Player"
    Public icon As String = "fa-solid fa-play"
    'this variable holds the page controller
    Public page As SDUIPage
    'this variable holds reference to the app
    'usually for constants and other things
    Public app As SDUIApp
    'the variable referencing banano lib
    Private banano As BANano        'ignore
End Sub

'sub to show the page
Sub Show(duiapp As SDUIApp)         'ignore
    'get the reference to the app
    app = duiapp
    banano.LoadLayout(app.PageViewer, "adlottieplayerlayout")
    'build the page, via code or loadlayouts
    BuildPage
End Sub
```

Note the following:

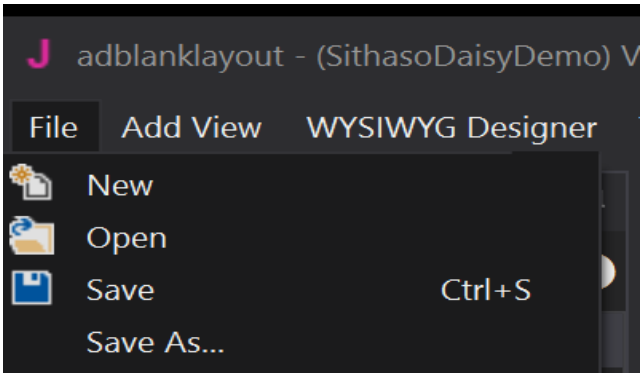
1. The code module name has been named in such a way that we know which page it is.
2. The name of the page on the code, title, icon and layout name on the **BANano.LoadLayout** code line has been named clearly and properly.

Step 4 - Copying the Page Layout from "adblanklayout"

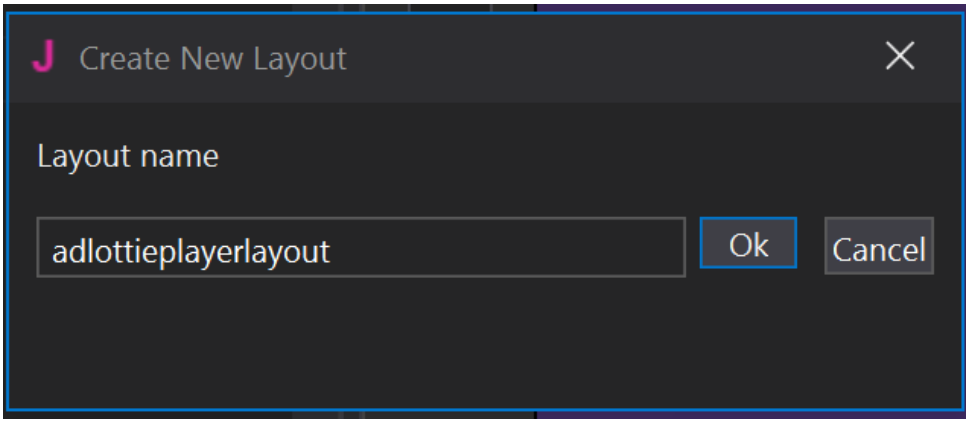
Now we need to ensue that the view/layout of our page exist. We will create it from an existing .bjl file.

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

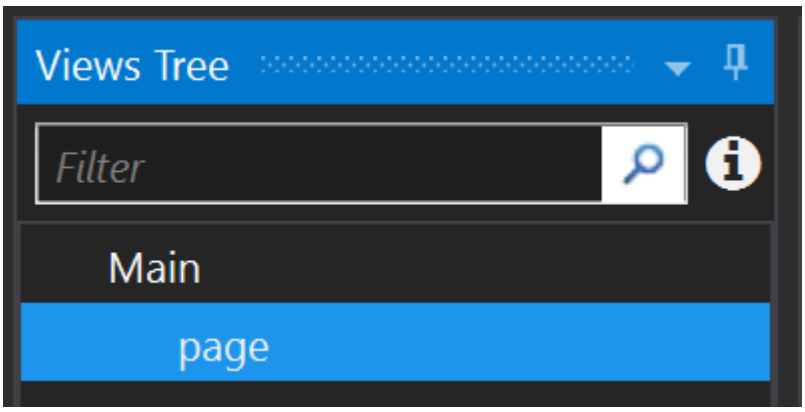
Click on File > Save As



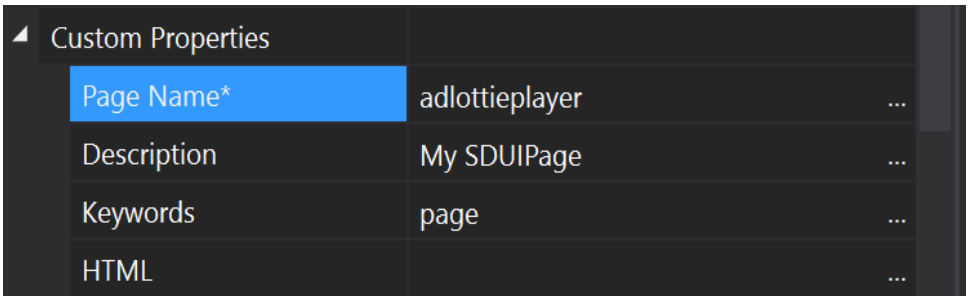
In the prompt that follows, paste the layout name you copied and click Ok.



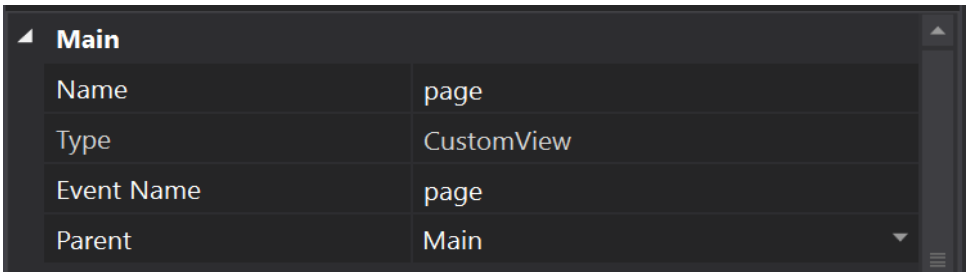
Now, on the Views Tree, click "page", to activate the custom view.



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.