outsystems

Creating a Mobile Application

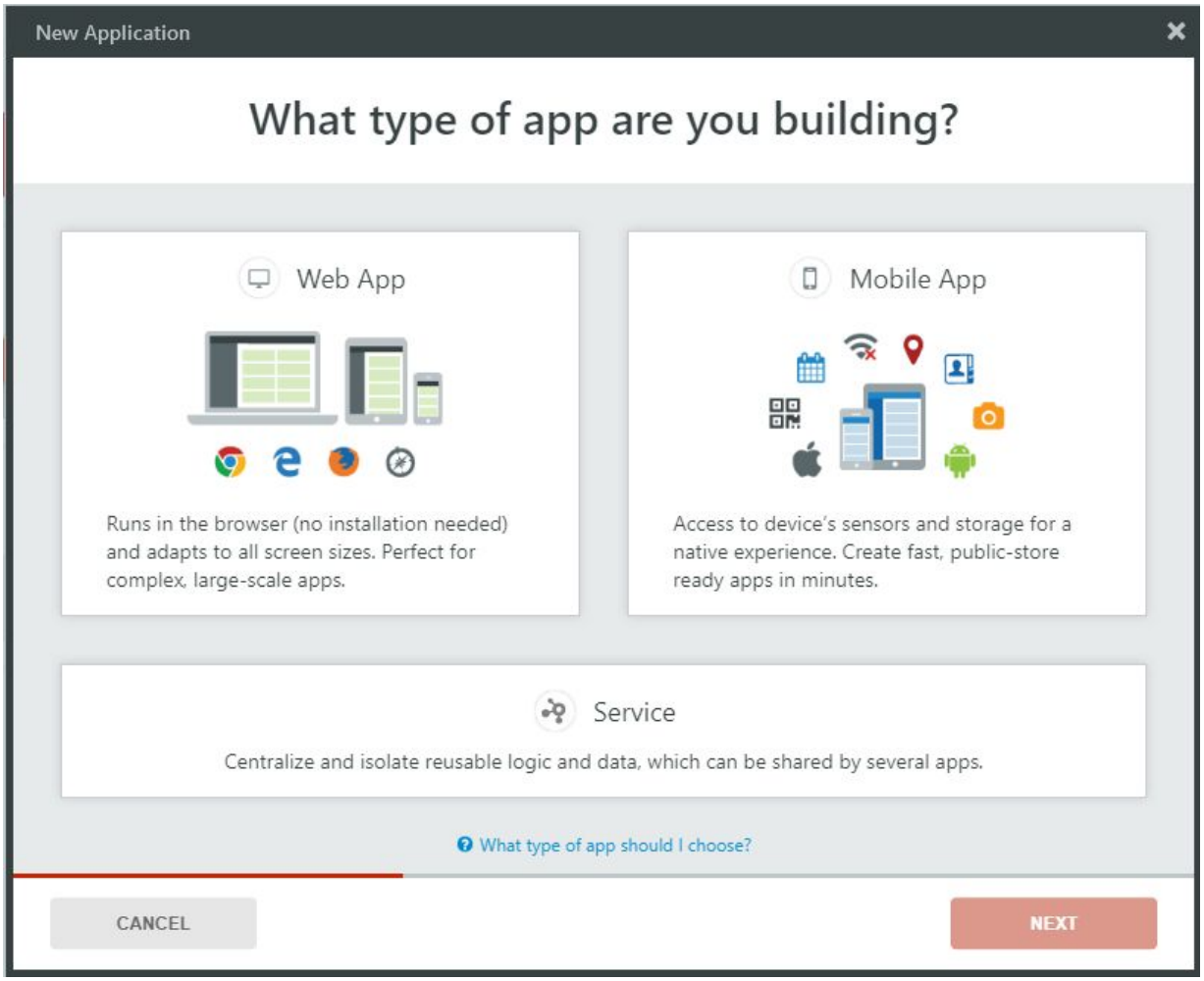


Table of Contents

Introduction

Over the course of this set of exercise labs, you will create a mobile application. The application will focus on creating and managing To Dos. The To Dos will be persisted in a database so they can be accessed from and shared across multiple devices. To Dos will have attributes such as category, priority (low, medium or high), due date and they can be marked as important (starred) by the user.

Users of the application will be able to access all of this information regardless of whether the device is online or offline. When offline, users will still be able to keep interacting with the application and changes will be saved locally in the device local storage. When the device returns to online mode, changes made while offline will automatically be synced to the server.

You constantly will be expanding your application, publishing it to the server and testing it in your mobile device. Throughout the process you will be learning and applying new OutSystems concepts.

At the end of this set of exercise labs, you will have a small, but well-formed application, spanning multiple screens and concepts that you can easily access from your mobile device.

In this specific exercise lab, you will:

- Create an application
- Create the Core module of the application
- Create the User Interface module of the application
- Create a Client Action that works as a function
- Use that function in another module
- Create an end-user of the application

Create the ToDo application

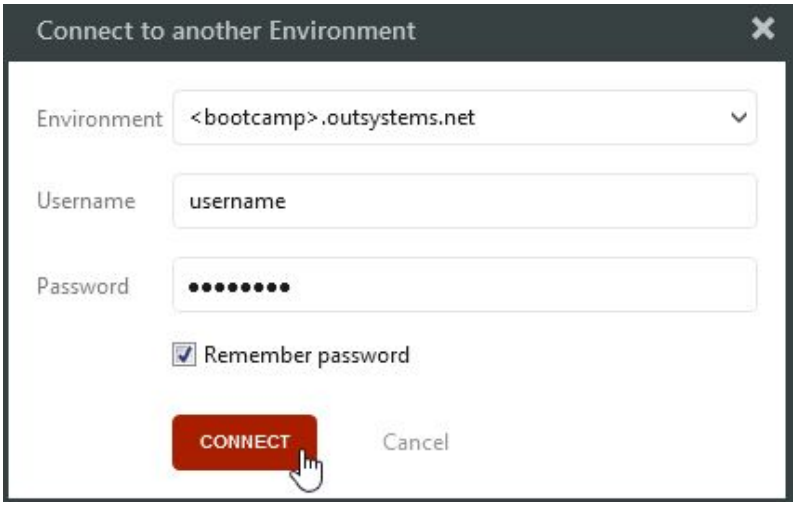
In this part of the exercise, you will create the **ToDo** application. This application will have two modules, the Core (data) module, and a User Interface (Mobile) module. The **ToDo** application will also have an icon and a description.

To develop any OutSystems application, we need the OutSystems Development Environment, **Service Studio**, and an OutSystems server (or **environment**).

1) Open Service Studio and login in your personal environment (when following the online class) or the bootcamp environment (when following the classroom training).

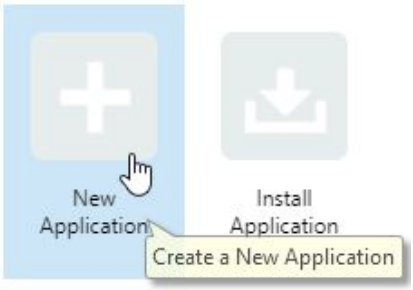
    a) Open Service Studio from the Start Menu or by double clicking the icon.

    b) In the **Connect to Environment** or **Switch Environment** dialog, enter the environment address, username and password you will be using to carry out the exercises, and click **Connect**.

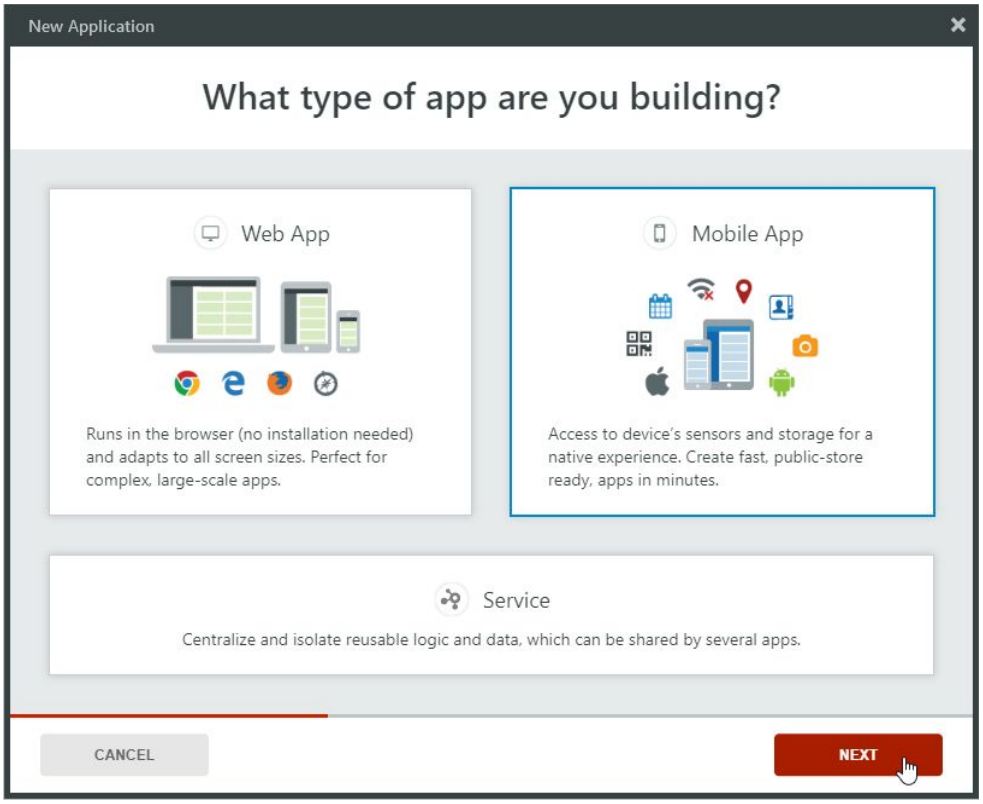


2) Create a new mobile application named *ToDo_<your_Initials>*, with a *ToDo_Core Mobile* **Blank** module, where we will later define our data model. Give the application a simple description and change the application icon to **todo-icon.png**.

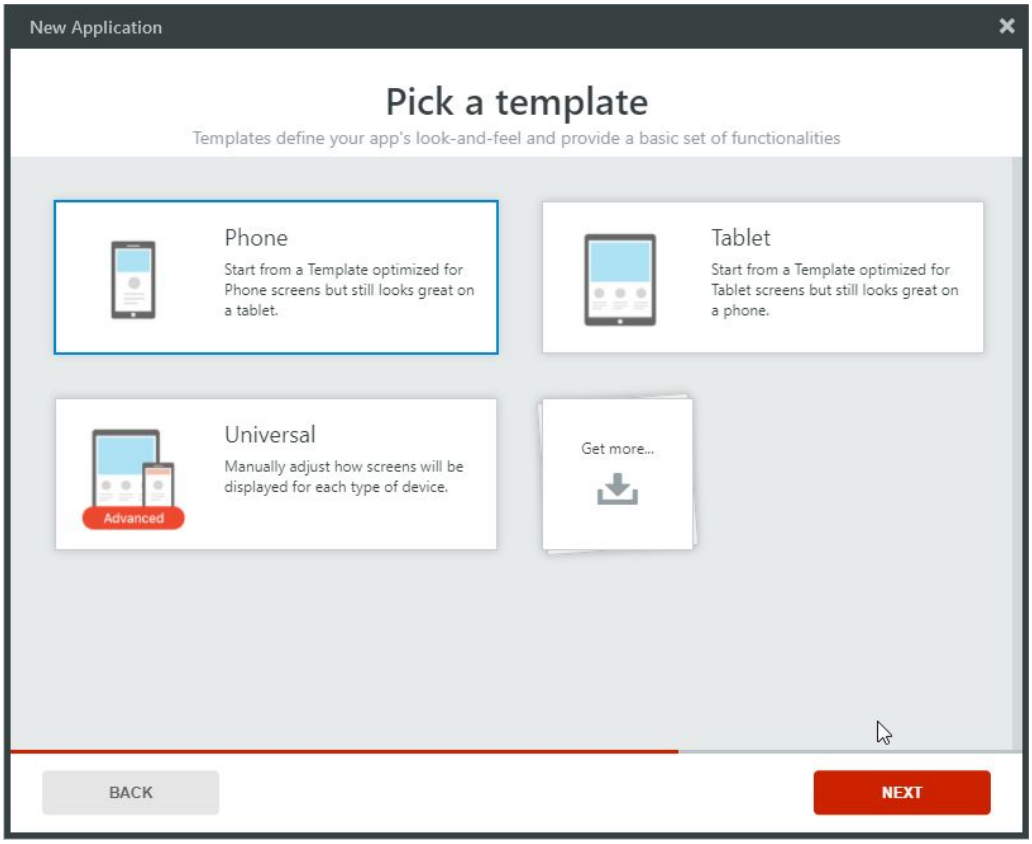
a) In the 'Applications in Development' area, click **New Application**.



b) In the **New Application** dialog, select **Mobile App**, and then click **Next.**



c) Select the **Phone** template, and then click **Next**. The templates provide a starting point for the application, containing the layout structure for the web pages. In this case, we are choosing pages with the menu on top.



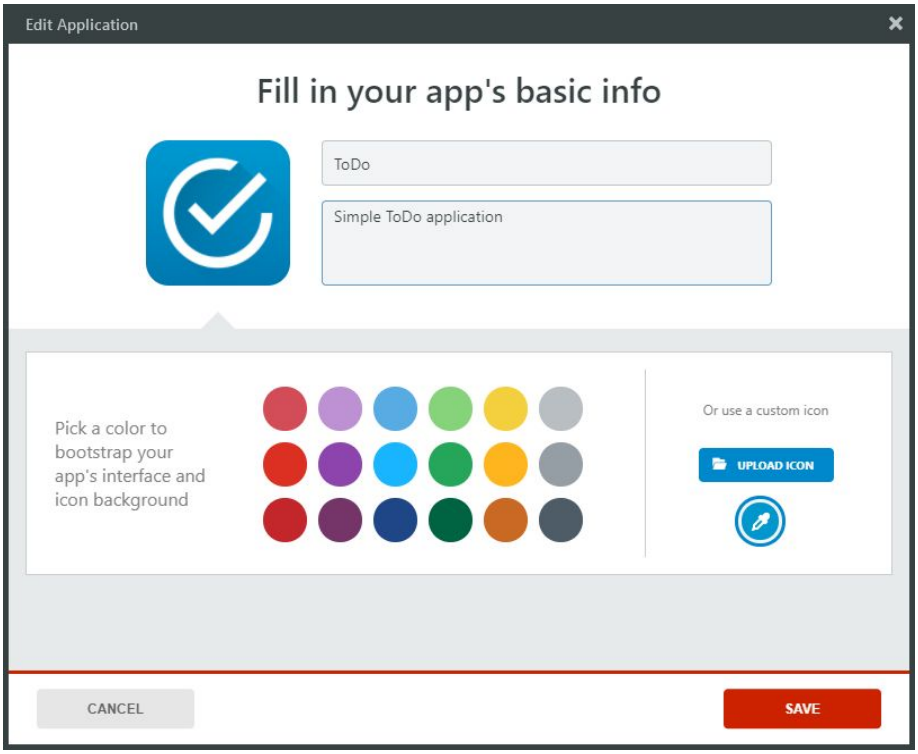
d) Set the Application Name to *ToDo_<yourinitials>*.

NOTE: From now on, all screenshots or references to the app will use the *ToDo* name, for simplicity.

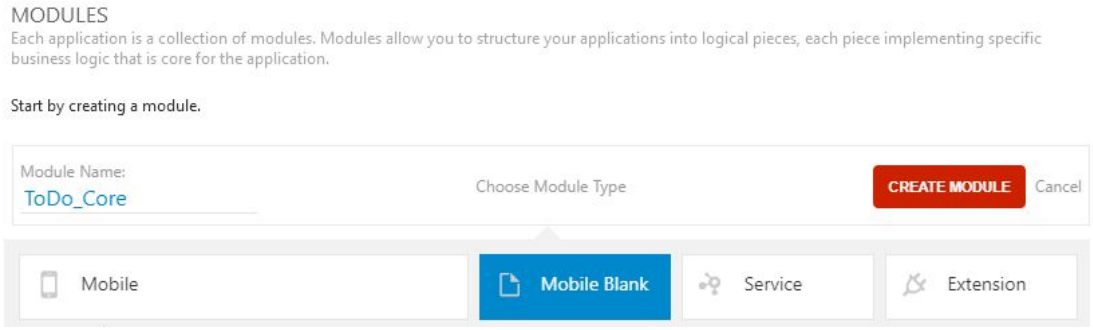
---

e) Type in a simple description for the application.

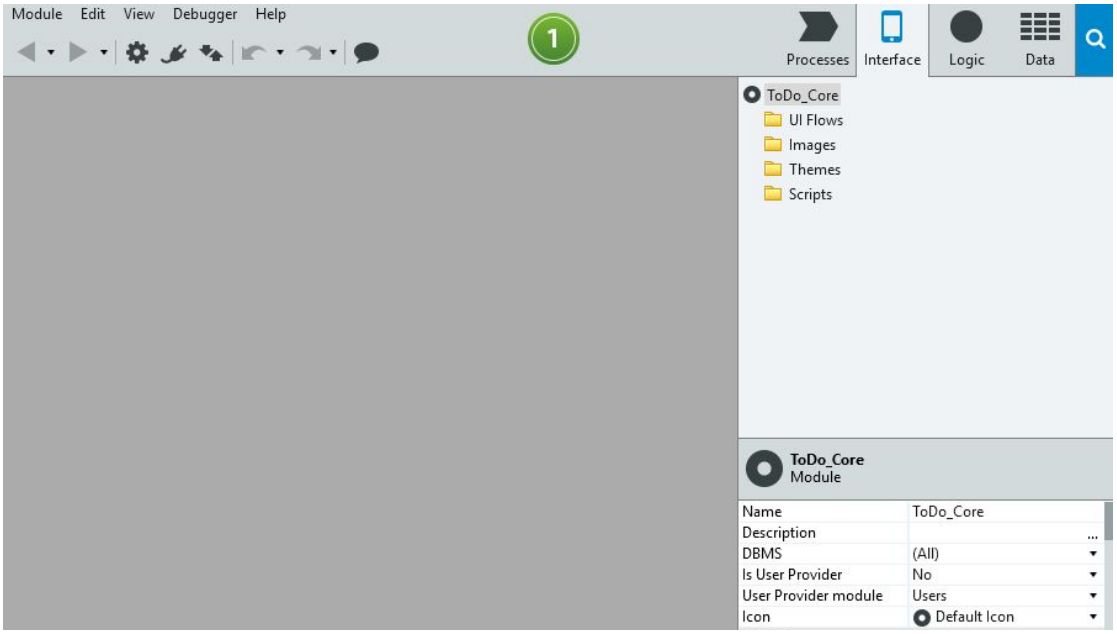
f) Click the **Upload Icon** button, select the **todo-icon.png** from the Resources folder on the Boot Camp materials. Select **Create App**.



g) In the **MODULES** area, the list of modules of the ToDo application can be found. Since we are creating a core data module, change the **Module Name** to *ToDo_Core_<yourinitials>* and select the **Mobile Blank** module type. Click **Create Module** to create the module.



h) Notice that the application is created and the module is opened. Since you selected a Mobile Blank module, you should see nothing in the workspace.

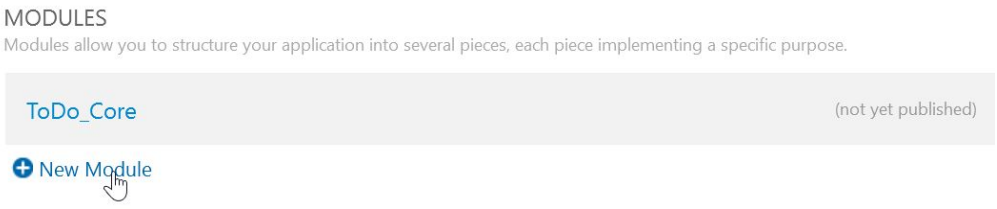


NOTE: If you see something visual in the workspace, it is because you created a **Mobile** module. In that case, you should delete the module and create a new **Mobile Blank** module.

3) Create the *ToDo_<your_initials>* module, the second module of the application. This should be a **Mobile** module, where the UI will be implemented. Don't forget to add your initials in front of the module's name, specially if you are following a classroom training.

   a) Go back to the Application tab.

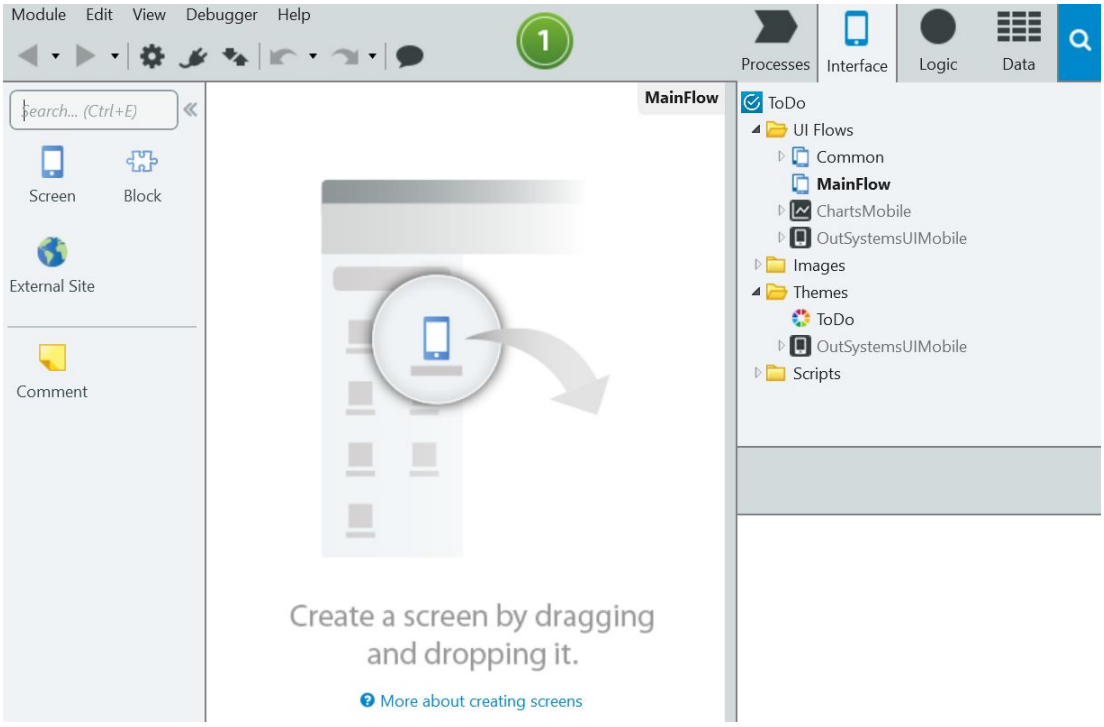
   b) Click on the **New Module** button to create a new module.



   c) Call it *ToDo_<your_initials>*.

   d) Select the **Mobile** module option. Click on the **Create Module** button.

   e) Since this module will have UI, you should see something like this

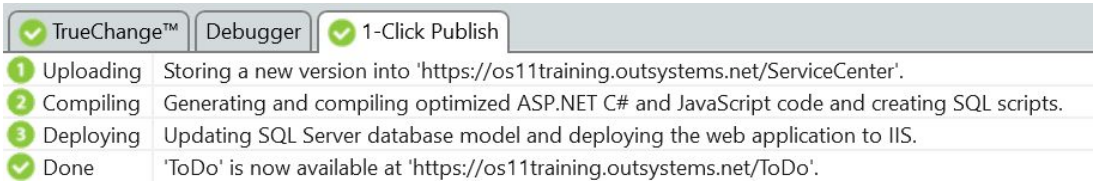

Module
Edit
View
Debugger
Help
1
Processes
Interface
Logic
Data
Search... (Ctrl+E)
Screen
Block
External Site
Comment
MainFlow
ToDo
UI Flows
Common
MainFlow
ChartsMobile
OutSystemsUIMobile
Images
Themes
ToDo
OutSystemsUIMobile
Scripts
Create a screen by dragging and dropping it.
More about creating screens

Publish the application modules

In this section, we will publish the modules in the server for the first time. This will create the first version of the modules in the application server. From now on, every new publish creates a new version of the module.

The publishing process uploads the module's information to the server. Then, it proceeds to generate and compile the necessary code and create the required database scripts. Finally, it uses the scripts to update the database and then deploys the application to the server.

1) Click the 1 **1-Click Publish** button to publish the **ToDo** module to the server.
2) Notice the **1-Click Publish tab** that appears near the bottom. This tab provides progress updates on the publishing process.



NOTE: Once published, module Screens become available at the URL displayed in the step **Done**. In general, that URL will be *https://hostname/ModuleName*

---

3) Open the *ToDo_Core (ToDo)* tab and publish the **ToDo_Core** module to the server.

NOTE: Since the current module has no UI, you won't be able to navigate to it using your browser, thus the 1-Click Publish will still be green after publishing.

Display a “Hello from the ToDo app” message

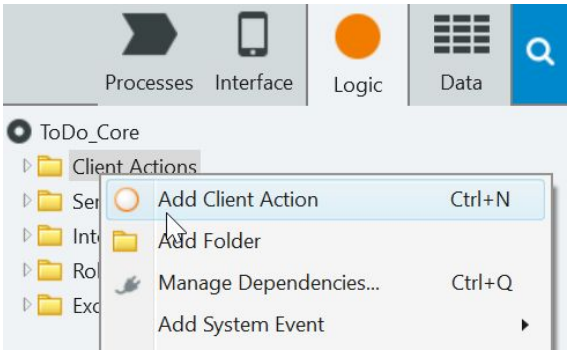
Now that we have the application created, with a Mobile and a Mobile Blank module, we want to display a simple message to the end-user that opens the application in the browser.

We will create a simple **Action** that returns the name of the application, **in the Core module**, and then will display it next to a “Hello World” message in the Home Screen of the application. This section will require an element from one module to be used in a different module. For that, we need to create a dependency between the two modules.

1) In the Core module, create a *GetAppName* Client Action that returns the application name (**ToDo**), using an Output Parameter *AppName* of **Type** *Text*. The Action should be set to *Public*, to make it reusable in other modules, and its **Function** property should be set to *Yes*, to make it usable on Expressions on Screens.

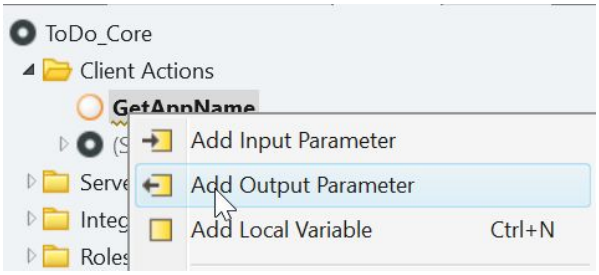
    a) Switch to the **Logic** tab.

    b) Right-click the **Client Actions** folder and select the *Add Client Action* options.



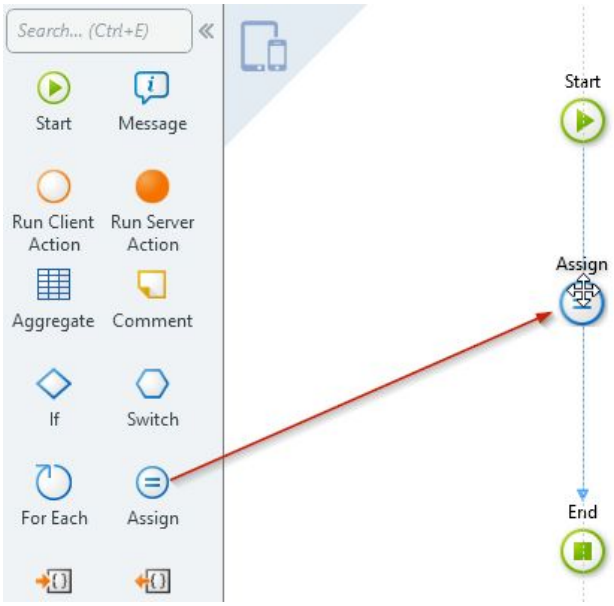
    c) Type in *GetAppName* to change the name of the Action from **Action1**.

    d) Right-click the **GetAppName** Action and select *Add Output Parameter*

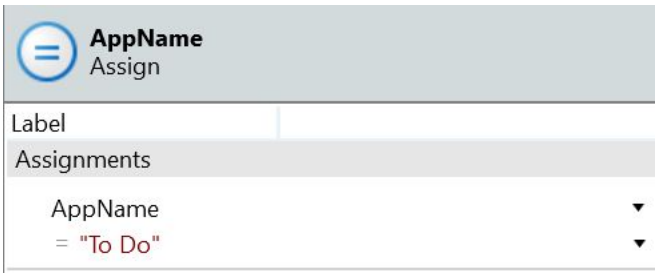


e) Type *AppName* as the **Name** of the Output Parameter.

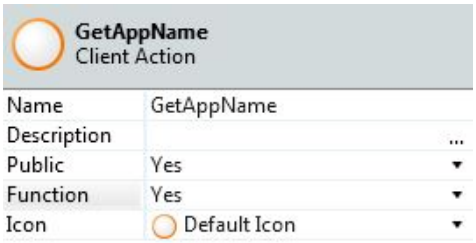
f) Drag an **Assign** from the toolbox and drop it on the action flow between the Start and End nodes.



g) In the Assign, set the **AppName** variable equal to "To Do"



h) In the properties area of the Client Action, set the **Public** property to *Yes* and the **Function** property to *Yes*.

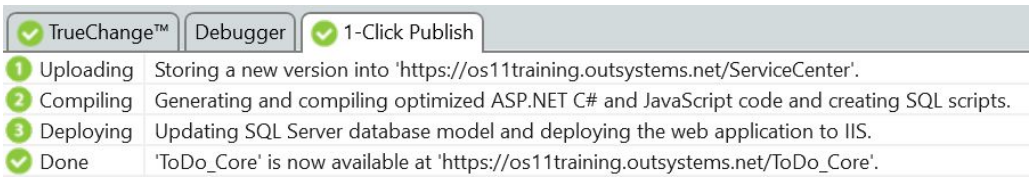


NOTE: By setting the **Function** property to *Yes,* enables the GetAppName Action to be used in an Expression, to be evaluated at runtime. A Function cannot have more than one Output Parameter.

2) Publish the **ToDo_Core** module to the server.

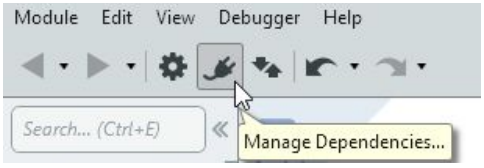
   a) Click the **1-Click Publish** button to publish the module to the serve

   b) In the 1-Click Publish tab you should see something similar to this

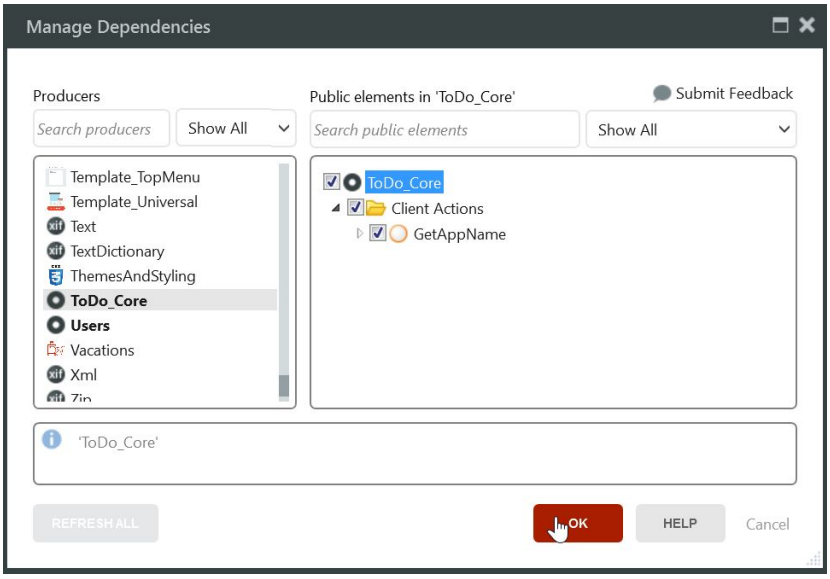


3) Reference the **GetAppName** Action in the **ToDo** Mobile module, to make it available in the module.

   a) Open the ToDo module and click the **Manage Dependencies...** icon.



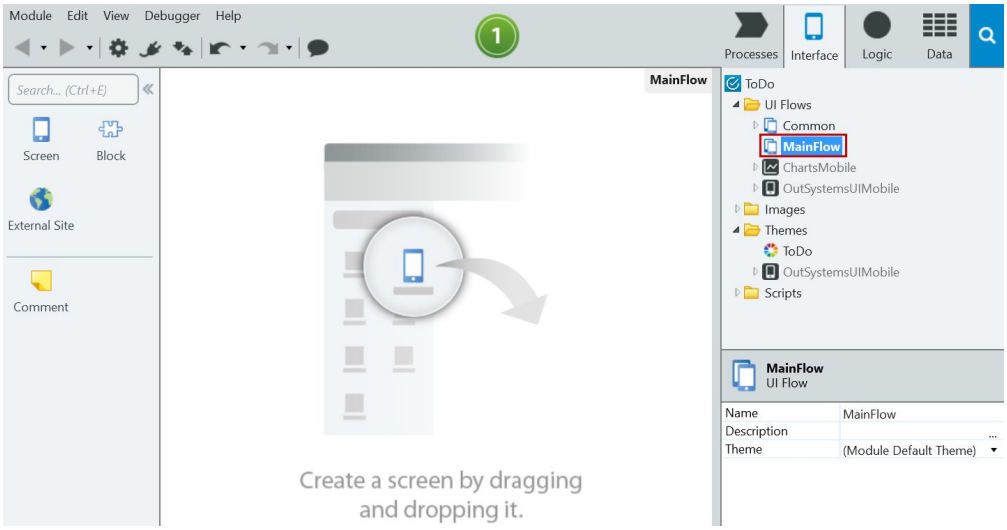
   b) In the **Manage Dependencies** dialog, select the **ToDo_Core** module on the left, and then select the **GetAppName** Client Action on the right.



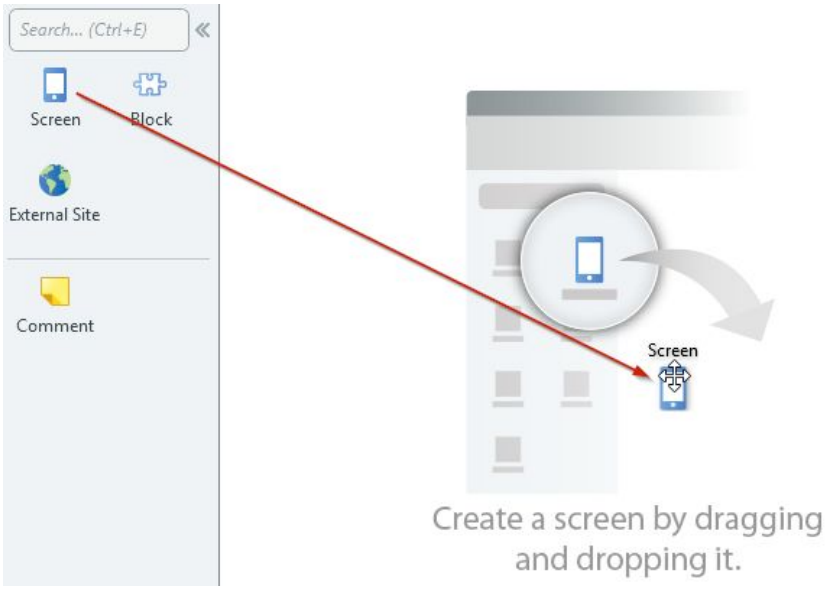
c) Click **Ok** to add the new dependency. This makes the Action available in the ToDo module.

4) Create a new Screen called *HomeScreen,* using the **Empty** template Screen.

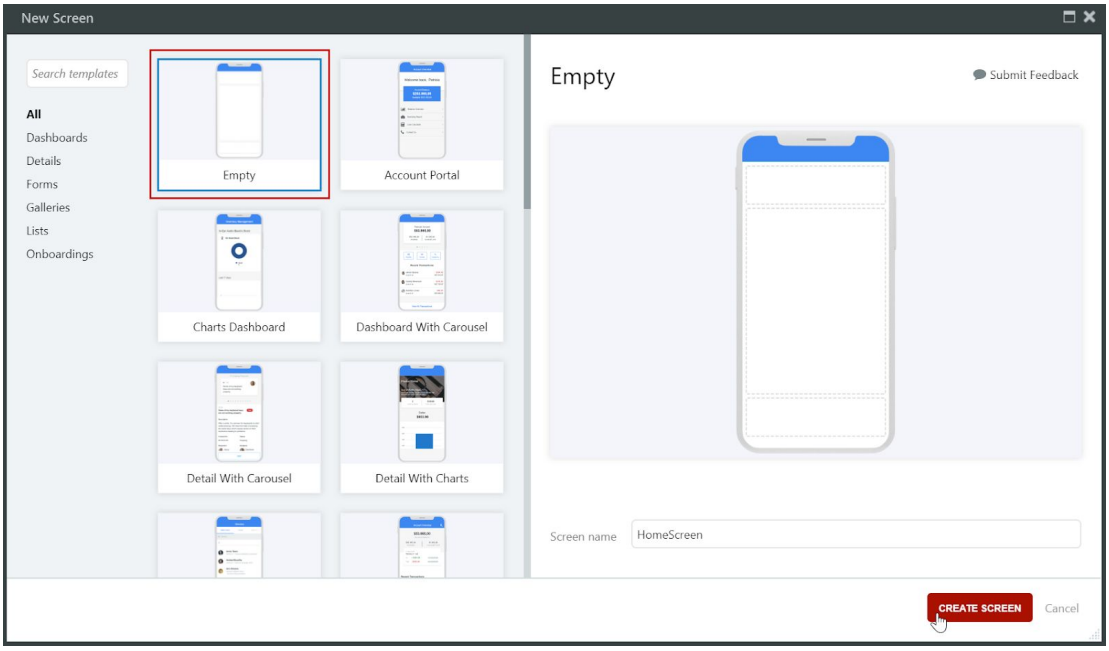
a) Still in the ToDo module, make sure that you see the workspace just like in the following screenshot, with the message "Create a screen by dragging and dropping it". If not, switch to the Interface tab and open the MainFlow, by double-clicking on it.



b) Drag a **Screen** to the workspace.



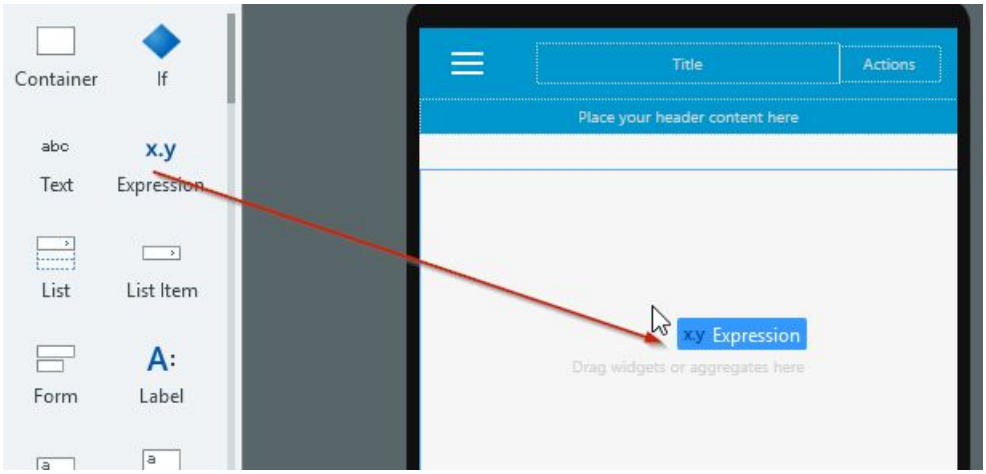
c) In the new window, select the **Empty** Template and name the Screen as *HomeScreen.* Click on the **Create Screen** button.



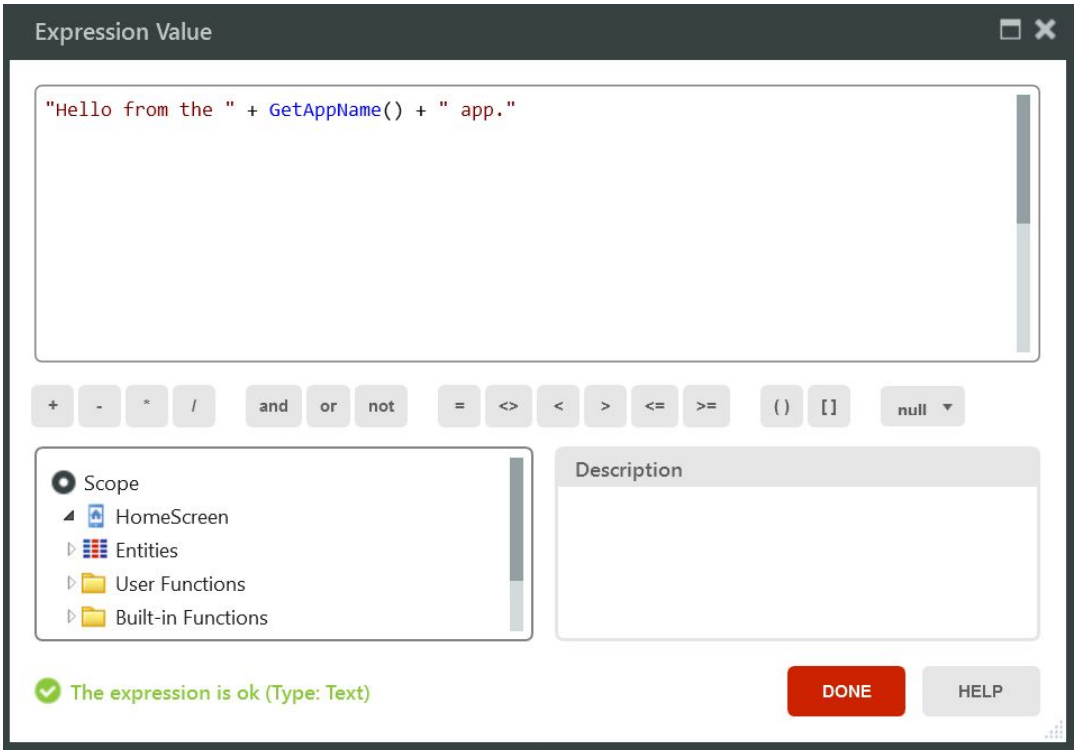
d) On the right, in the elements area, we can find the HomeScreen under the **MainFlow**.

5) Add an **Expression** to the **HomeScreen** to display the results from the **GetAppName** Action to produce a *"Hello from the To Do app"* message on the main content area.

a) Drag and drop an **Expression** from the toolbox to the main content area



b) Set the Expression to *"Hello from the "* + *GetAppName()* + *" app."*



Create a new User and test the application

Now that we have the message defined in the HomeScreen of the application, it is time to test it and make sure that it is properly displayed.

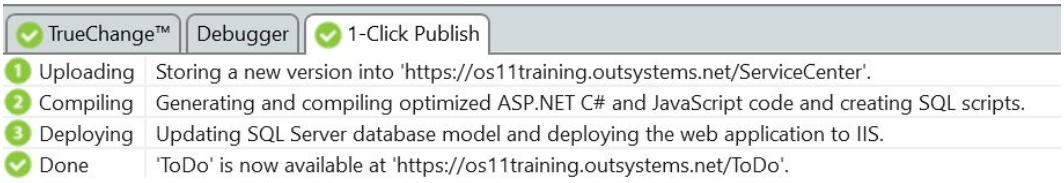
To test the application, we will use the browser's device emulator (Chrome or Safari only), which will open automatically with the Open in Browser Button in Service Studio, if you have one of the two as the default browser.

Also, to login to the application, we will need to also create an end-user in the server.

1) Publish the ToDo module to save the changes in the server.

   a) Click the **1-Click Publish** button to publish the module to the server.

   b) In the 1-Click Publish tab you should see something similar to this.

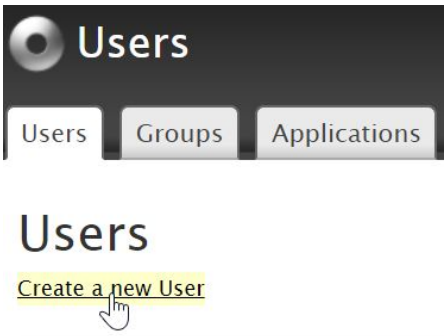


2) Create a new end-user in the server to test the application, using the **Users** application. Give it a username and password.

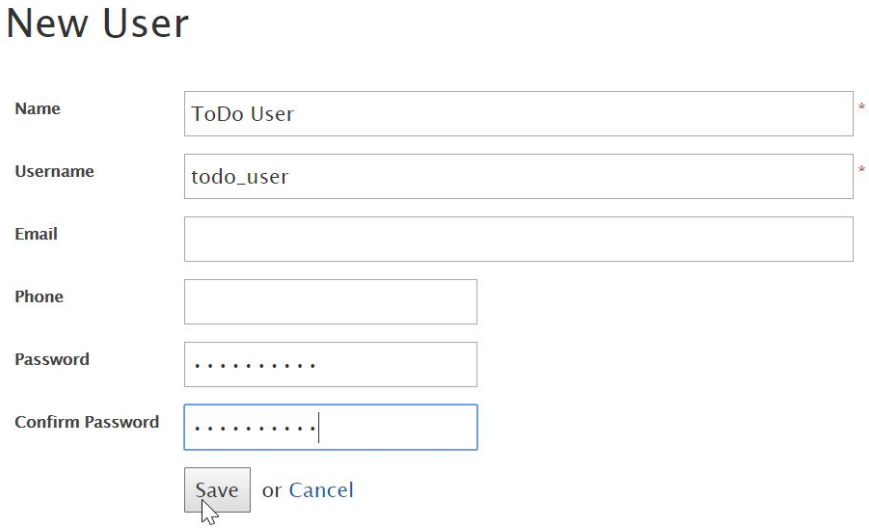
   a) Open a browser window and navigate to the following URL: *https://<server_address>/Users*

   b) When using a Personal Environment, you may login with your own credentials. In a classroom Boot Camp server, login with the user **admin** and password **admin**.

   c) Click the 'Create a new User' link.



d) Define the user name, username and password as you desire. Click **Save** to create the user.



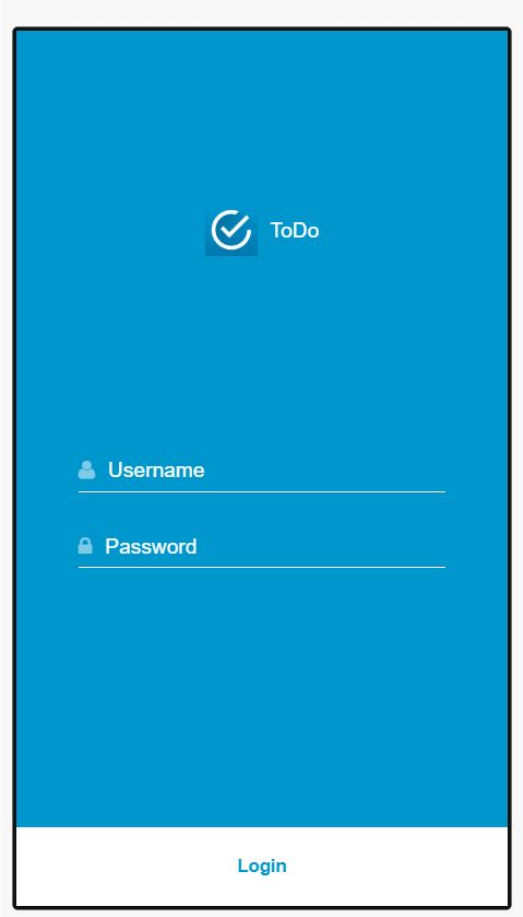
e) Logout from the Users application.

3) Test the **ToDo** application and make sure the message appear properly.

a) Return to Service Studio, and click the **Open in Browser** button.

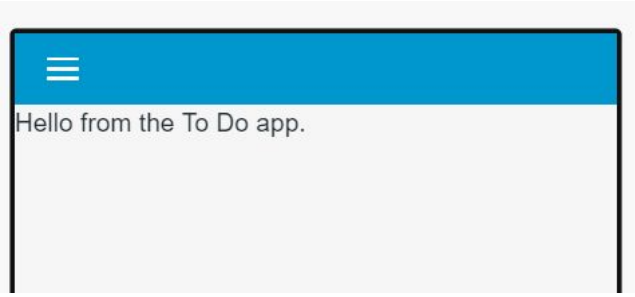
b) You should now see your application in a device emulator inside your browser.

c) After the load process completes, you will see something similar to this



d) Login with the user credentials that you have just created.

e) Verify that “Hello from the To Do app” appears in the main content area.



End of Lab

In this exercise lab, we created a mobile application, ToDo, to list and manage of tasks that a user has to do. Since this is the first lab, we just created the mobile app in Service Studio and two modules: one Mobile, to hold all the UI of the application, and one Mobile Blank, to hold the data model of the application.

After creating these modules, we published them for the first time to the server.

To test the app in the browser emulator for the first time, we created a Client Action that returns the name of the app, which is later used to display it in the HomeScreen of the application. Since the Action was created in the Blank module, and the Screen is present in the UI module, we created a dependency between the two modules. Finally, we created a new end-user in the server, to login to the app and see the message on the Screen.