

## OutSystems / docs-product

Public

[Code](#) [Issues 13](#) [Pull requests 5](#) [Actions](#) [Wiki](#) [Security](#) [Insights](#)[master](#) 

...

[docs-product / src / getting-started / create-reactive-web.md](#)

OutSystemsAMM adding version tags

 History

7 contributors



250 lines (133 sloc) | 10.9 KB

...

	summary	tags	locale	guid	app_type	platform-version
	Follow this tutorial to learn how to create a Reactive Web App to manage tasks.	runtime-reactiveweb;	en-us	795332a5-e1f3-40c0-886d-75b7bddf48af	reactive web apps	o11

# Create Your First Reactive Web App

Check our training [Becoming a Reactive Web Developer](#) for a guided introduction into Reactive Web App.

Developing Reactive Web Apps with OutSystems is fast. In this example, we will use a spreadsheet to create some database entries and then add user interface and logic to connect everything - into a ToDo app.

This is the overview of what we are about to do:

1. Create a Reactive Web App, name it, and choose the primary color.
2. Automatically create a database model by importing data from Excel.
3. Create a Screen that lists the data from the database.
4. Create a Screen to add and update records.
5. Implement functionality to delete records.
6. Test the application in a browser.

## Pre-requisites

---

You should satisfy the following requirements to develop, run, and deploy a Reactive App.

- Service Studio 11.53.7 or later
- Platform Server 11 - Release Oct.2019.CP1 or later
- LifeTime Management Console - Release Sep.2019 version 11.0.321.0 or later

We also recommend that you update the following components:

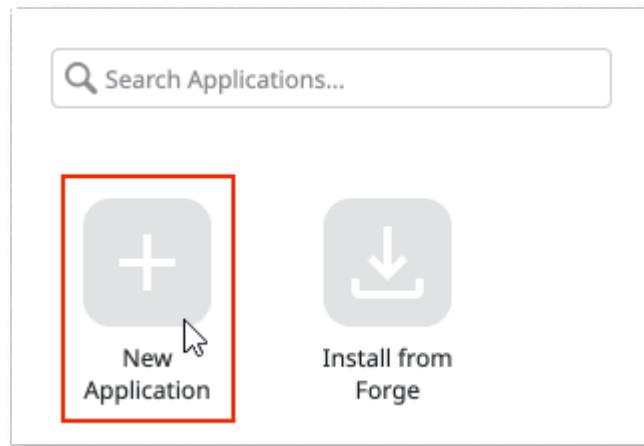
- OutSystems UI
- OutSystems UI Templates Reactive

## Create a Reactive Web App { #new-app }

---

Let's create a sample "ToDo" app.

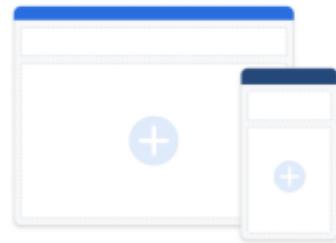
1. In Service Studio, select New Application.



2. In the New Application window, choose From scratch, then click Next.

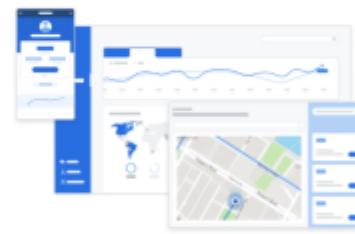
New Application

## How do you want to start?



### From scratch

Build your app using a blank canvas



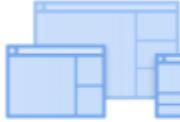
### From an app

Use a ready-made app

3. Choose **Reactive Web App**, then click **Next**.

New Application

## What are you building?



**Reactive Web App**

Responsive interface that can be run on all browsers and devices



**Tablet App**

Optimized for tablets. Create native mobile apps or PWAs

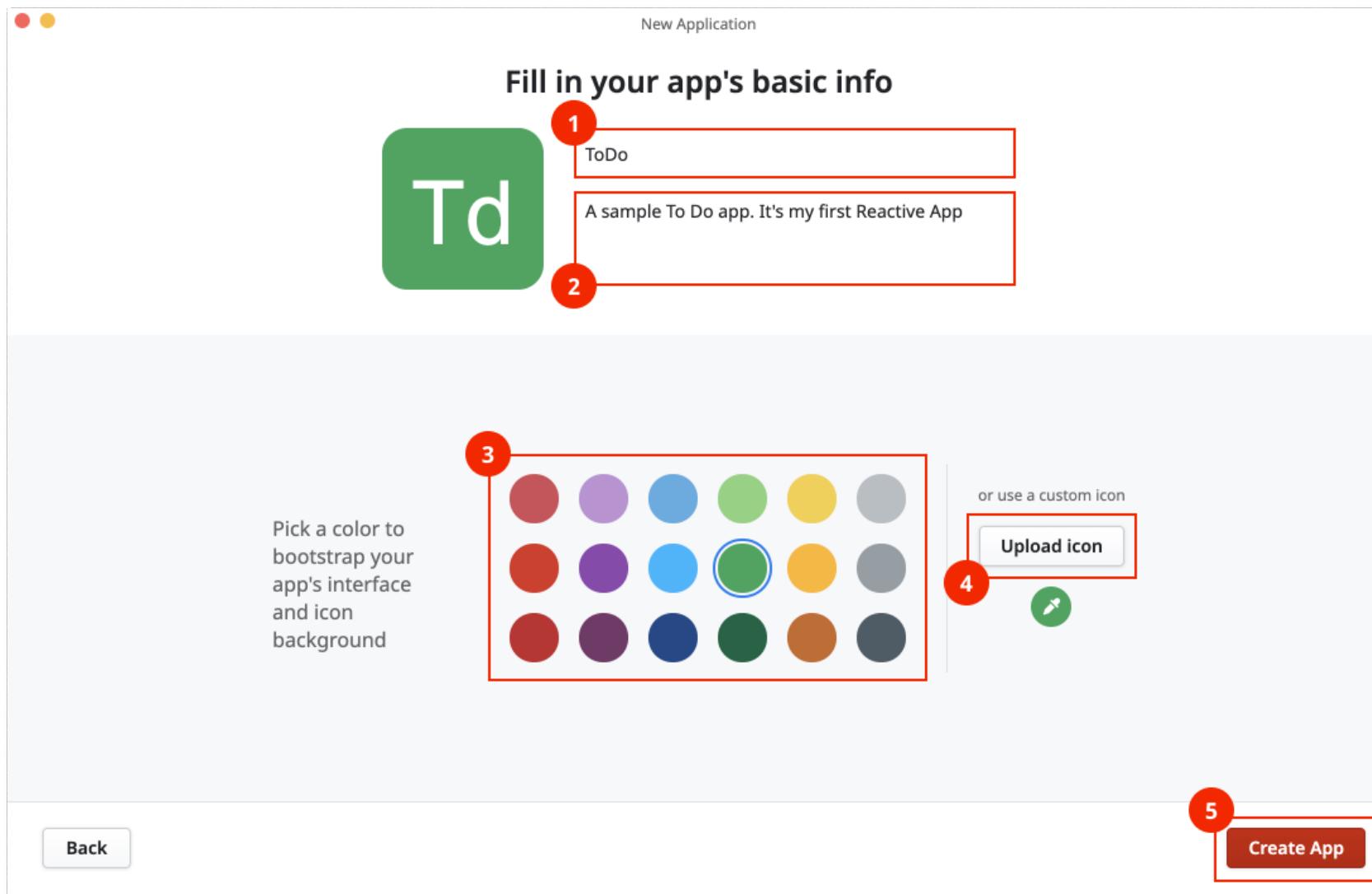


**Phone App**

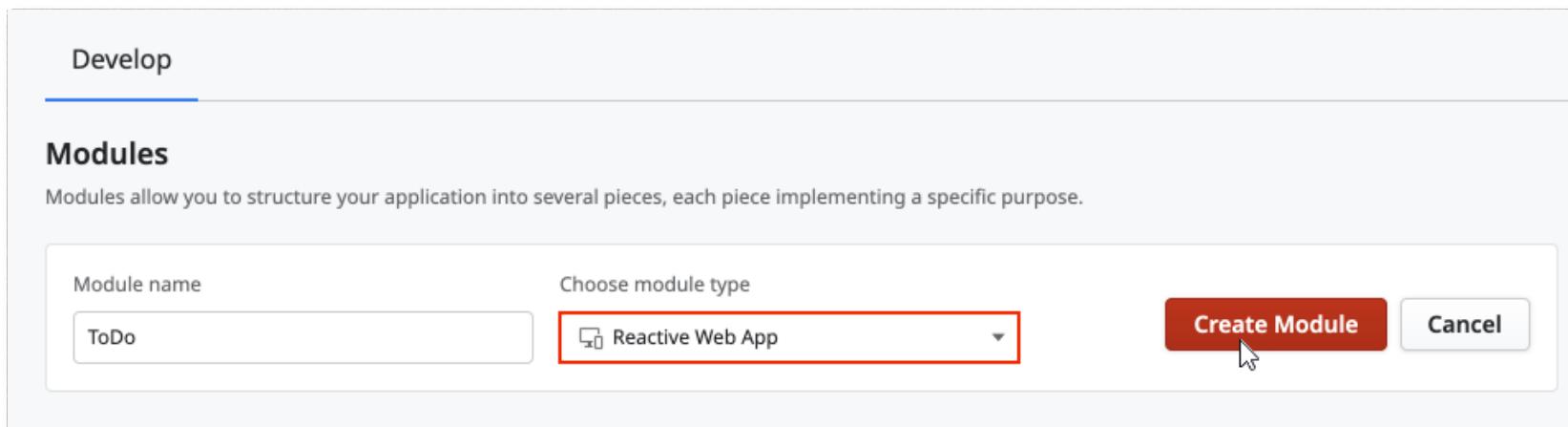
Optimized for phones. Create native mobile apps or PWAs

4. In the properties for your new app, set up the following:

- i. Name your app "ToDo".
- ii. Add a description.
- iii. Change the primary color of your app by picking one of the suggested colors, or using the color picker.
- iv. Upload an icon by clicking **Upload icon**.
- v. Click **Create App** to advance to the next step.



5. In the application properties screen, make sure **Reactive Web App** is selected in the **Choose module type** dropdown. Click **Create Module** to create the first module and open it for editing.



The screenshot shows the 'Develop' section of the OutSystems interface. Under 'Modules', there is a dialog box for creating a new module. The 'Module name' field is filled with 'ToDo'. The 'Choose module type' dropdown menu is open, and the option 'Reactive Web App' is selected and highlighted with a red border. To the right of the dropdown are two buttons: a red 'Create Module' button with a white cursor icon over it, and a grey 'Cancel' button.

## Create a database table from an Excel file { #create-entity-from-excel }

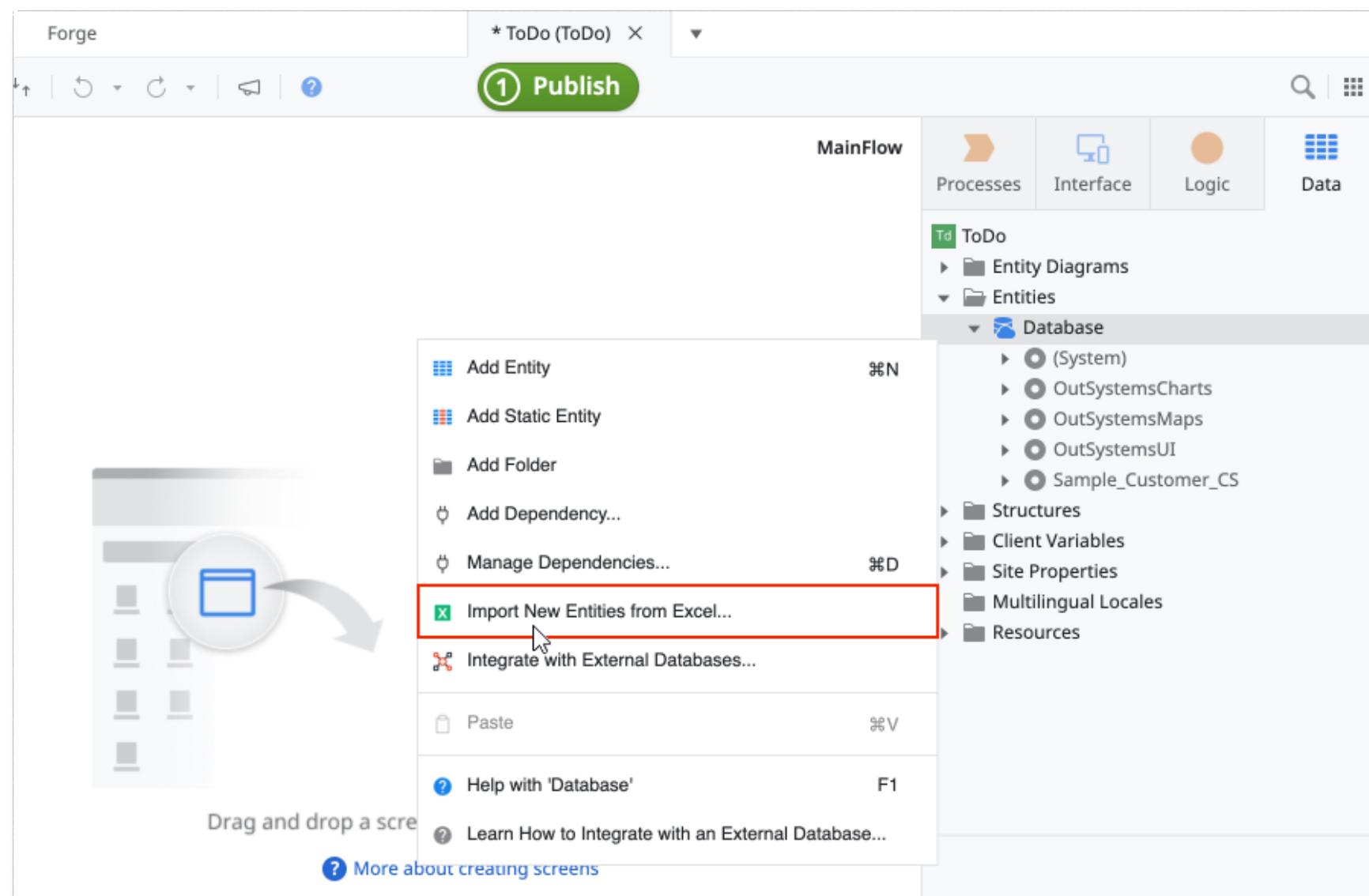
In OutSystems, application data can be stored in a relational database. This means that the first step in creating an app is defining the data model.

To do this, we are going to use an Excel file that already contains the following task information:

- Description
- Due Date
- Is Active

Download the [tutorial Excel file](#) to your computer.

In the **ToDo** module, open the **Data** tab on the top right-hand corner, right-click the **Database** folder, choose **Import New Entities from Excel**, and select the `TutorialResource.xlsx` Excel file. Click **Import** in the dialog to confirm.



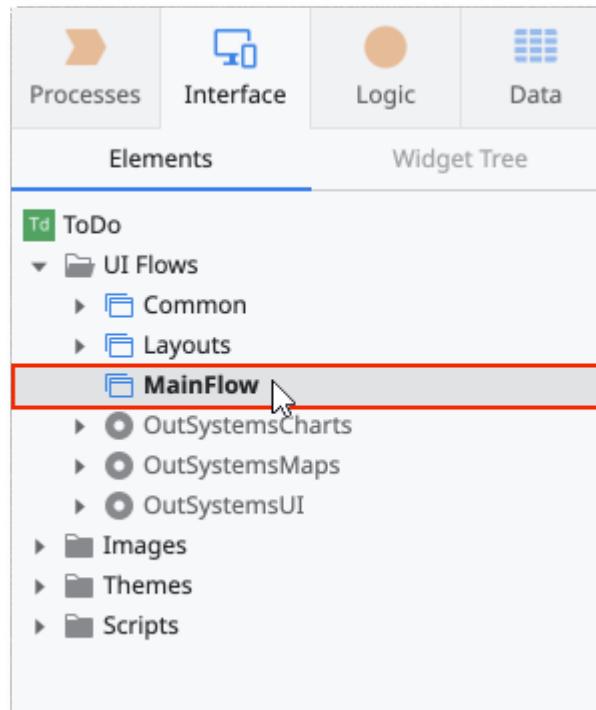
When importing an Excel file, OutSystems creates a database table (called an Entity in OutSystems) with the necessary columns (called Attributes in OutSystems) to store the data in the database.

Behind the scenes, OutSystems also creates logic to import each row in the Excel file into a corresponding database record. After publishing your application, the background logic populates (bootstraps) your database with the data from the Excel file. In this tutorial, we're only storing the data in the server database.

## Create a Screen to show tasks

Now we can create a Screen that shows all of the tasks.

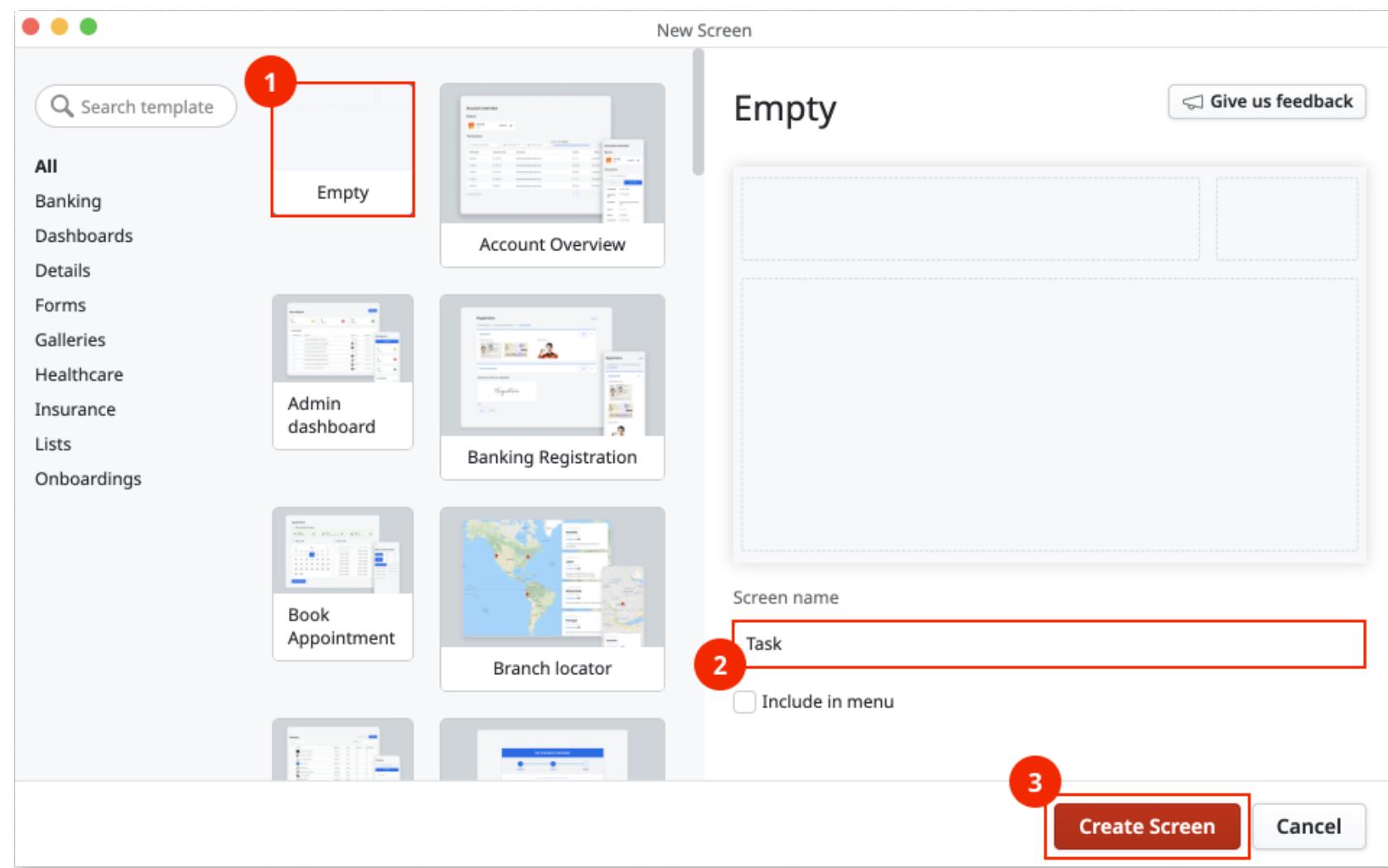
1. Switch to the **Interface** tab on the top right-hand corner, and double-click **MainFlow** under **UI Flows**.



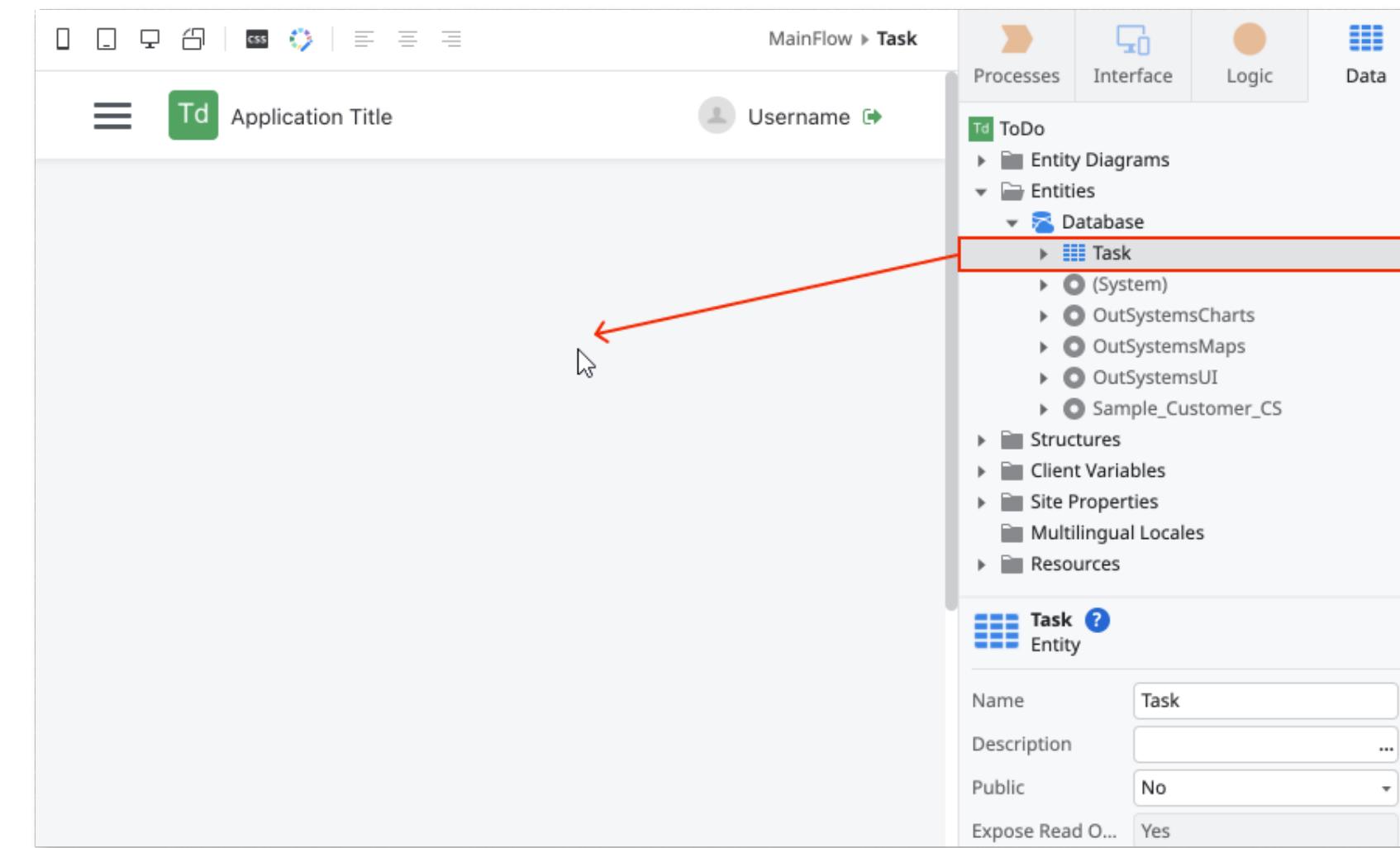
2. Drag a **Screen** from the Toolbox to an empty area in the Main Editor window.



3. Choose the **Empty** template (1), name your screen **Task** (2) and click **Create Screen** (3).



4. Drag the **Task Entity** from the **Data** tab to the Content placeholder of the screen.



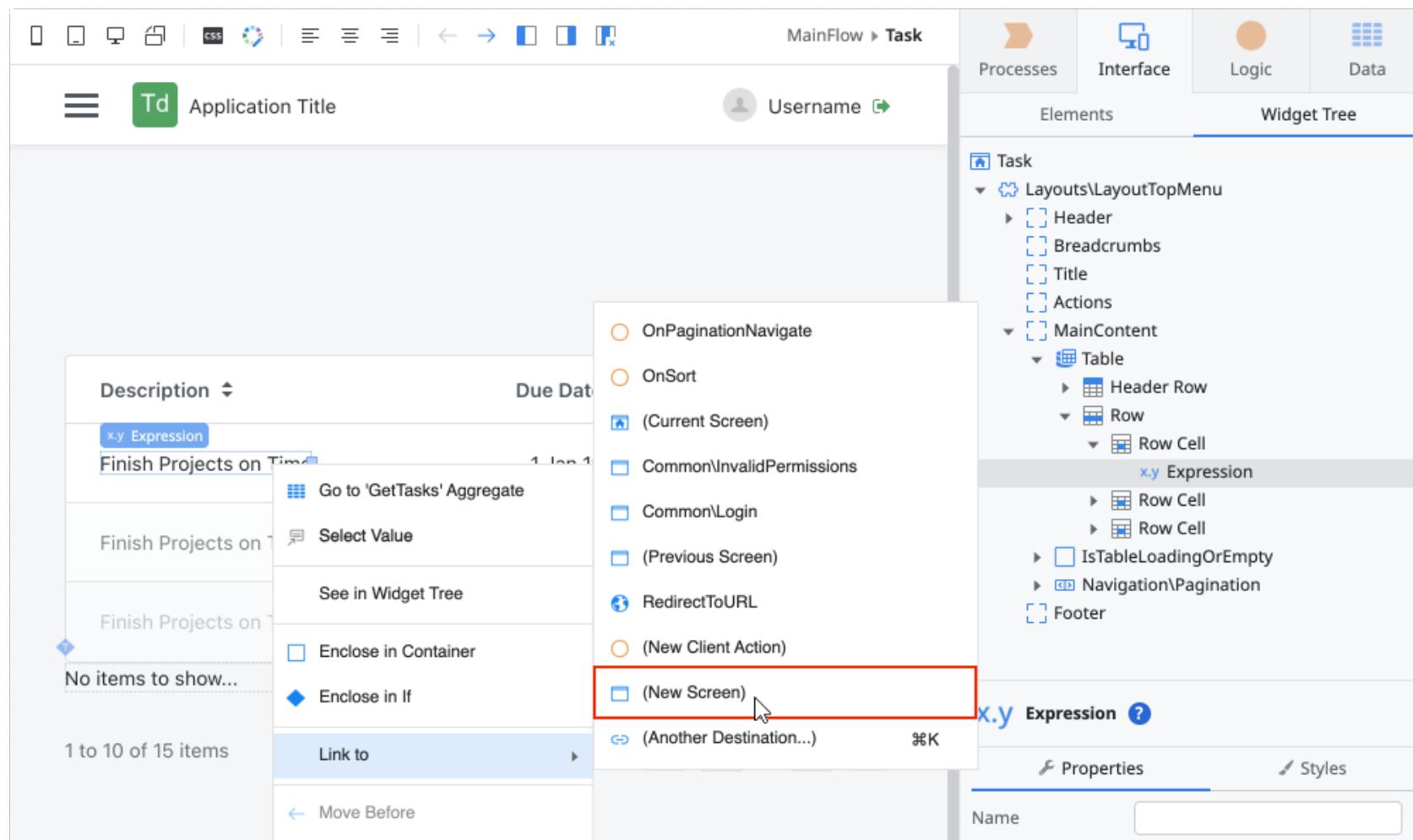
This automatically creates a Table with pagination support.

The screenshot shows the OutSystems Studio interface. On the left, there's a navigation bar with icons for Home, Project, Application, CSS, and Entity Diagrams. The main title is "MainFlow > Task". Below the title, there's a header with "Application Title" and a "Username" dropdown. The main content area displays a table with three rows, each containing a task description, due date, and an active status indicator. The table has columns for "Description", "Due Date", and "Is Active". The first two rows have identical data: "Finish Projects on Time", "1 Jan 1900", and a checked checkbox. The third row also has the same data. A message "No items to show..." is displayed below the table. At the bottom, there's a pagination control showing "1 to 10 of 15 items" and a set of navigation arrows. On the right side, there's a sidebar titled "ToDo" which includes sections for "Entity Diagrams", "Entities", "Database", and "Task". The "Task" section is expanded, showing various task-related entities like "Id", "Description", "DueDate", etc., and actions like "CreateTask", "CreateOrUpdateTask", etc. Below this, there's a section for "Task Entity" with fields for "Name" (set to "Task") and "Description". The top right of the interface has tabs for "Processes", "Interface", "Logic", and "Data".

## Create a Screen to edit tasks

Creating a Screen to edit the records is as fast as creating a Table. Follow these steps:

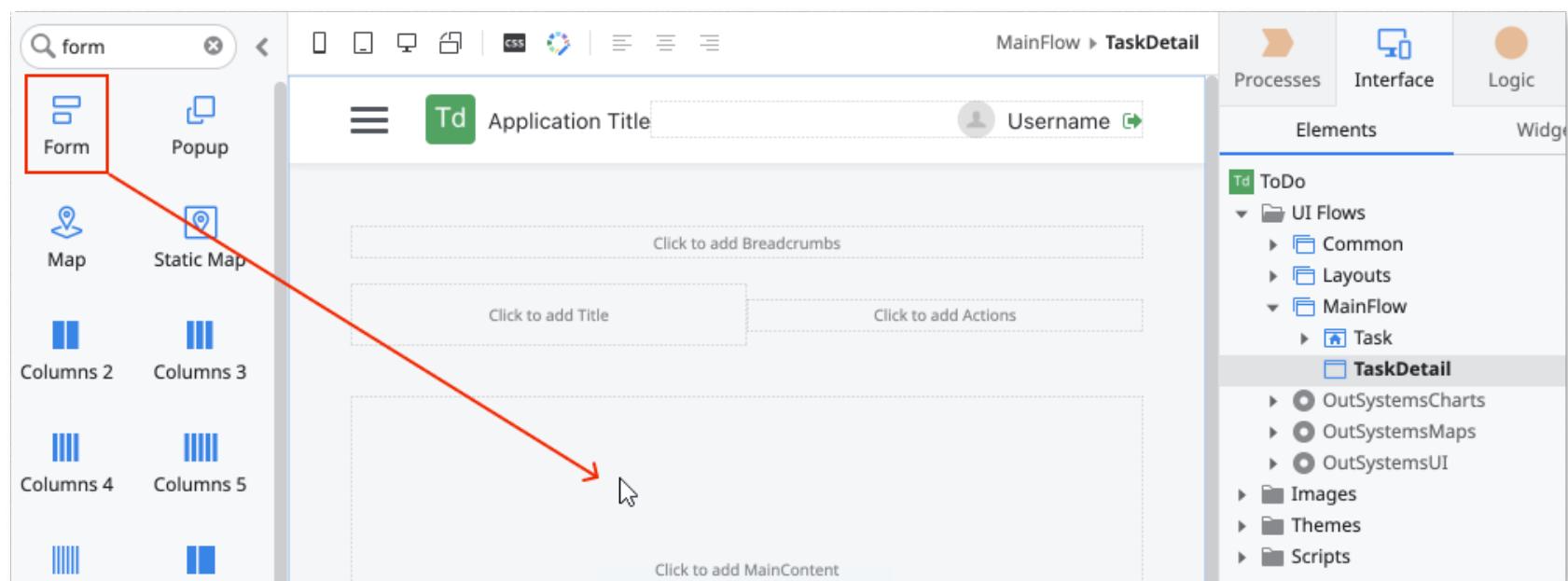
1. Right-click the title of the first task in the row and select **Link to > (New Screen)**.



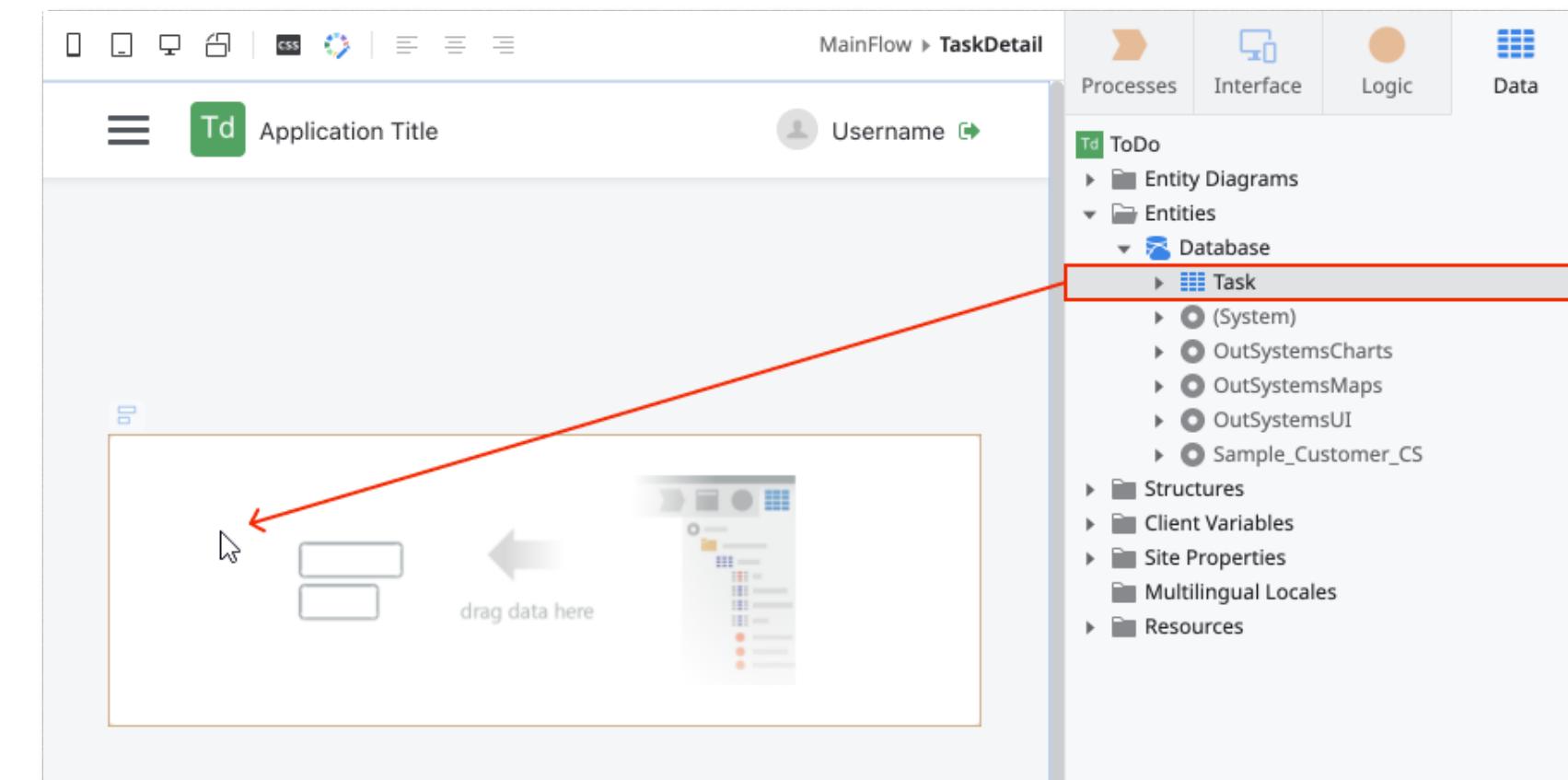
2. Choose the **Empty** template, name your screen `TaskDetail` and click **Create Screen**.

This links the title of the tasks to a newly created screen. We will use this new screen to edit the tasks. For that, we will need a form.

3. Drag a **Form** widget from the Toolbox to the Content placeholder in the `TaskDetail` screen.

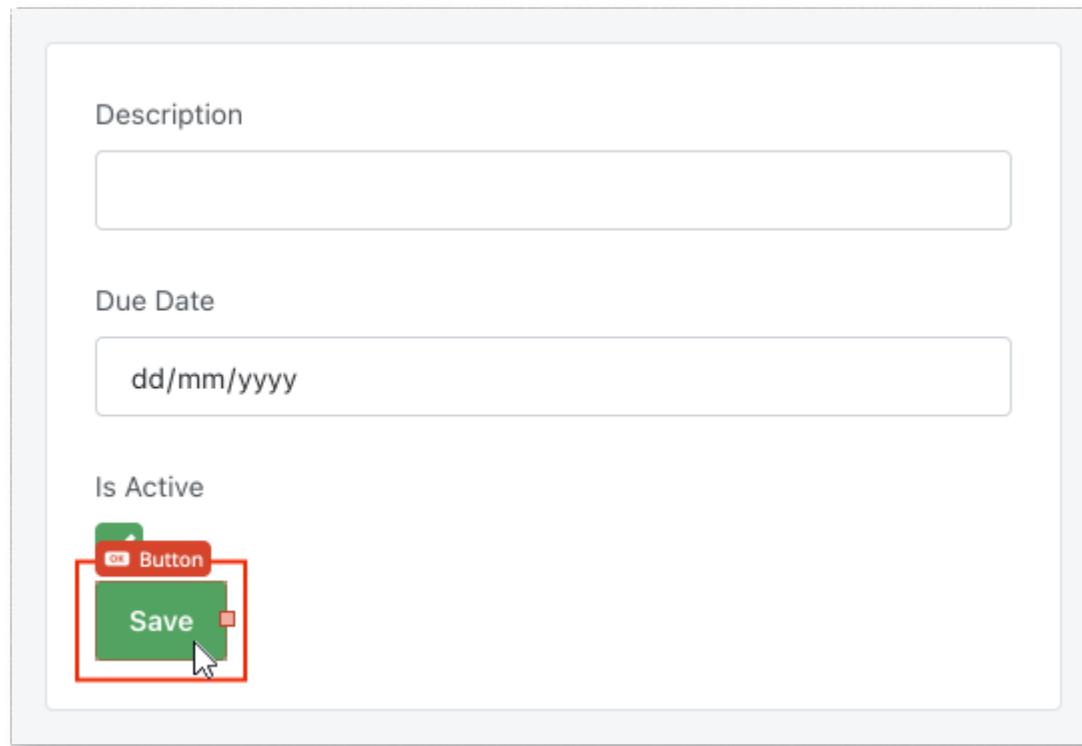


4. Drag the **Task** entity from the **Data** tab to the previously created Form.

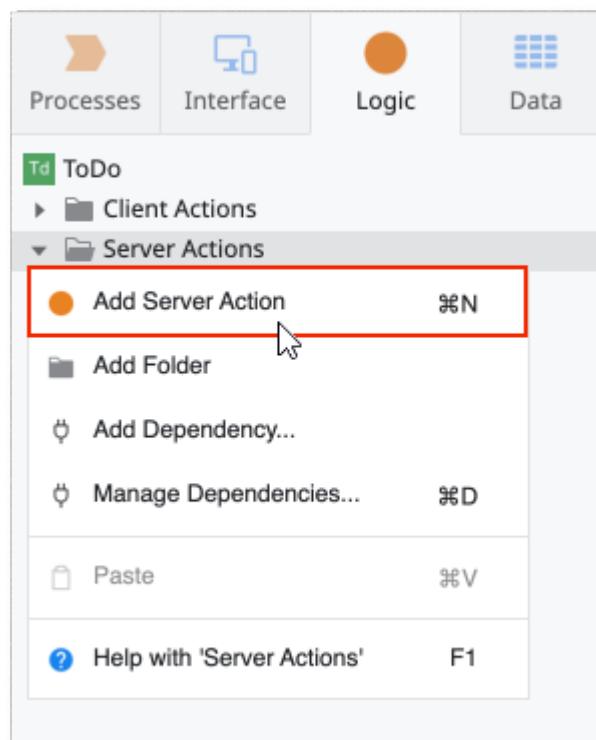


Now we will define the logic that runs when the end users press the **Save** button:

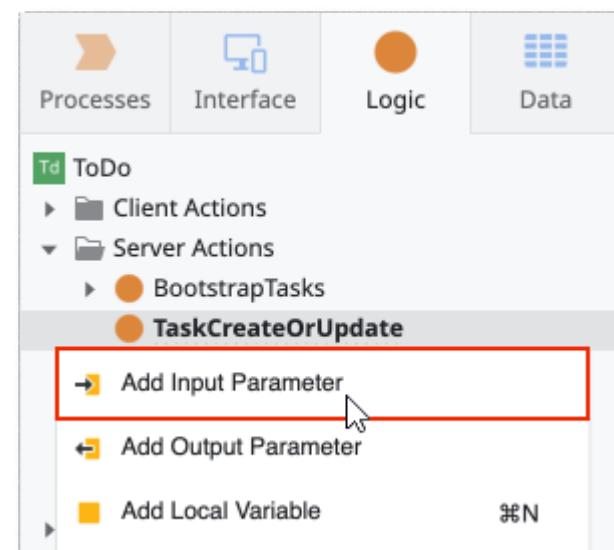
5. Double-click an empty area of the **Save** button to define the logic associated with the button. This will create a new screen action named **SaveOnClick**.



6. In the Logic tab, right-click Server Actions and select Add Server Action. Set its name to **TaskCreateOrUpdate**.



7. Right-click the newly created action and select Add Input Parameter. Set its name to Task.



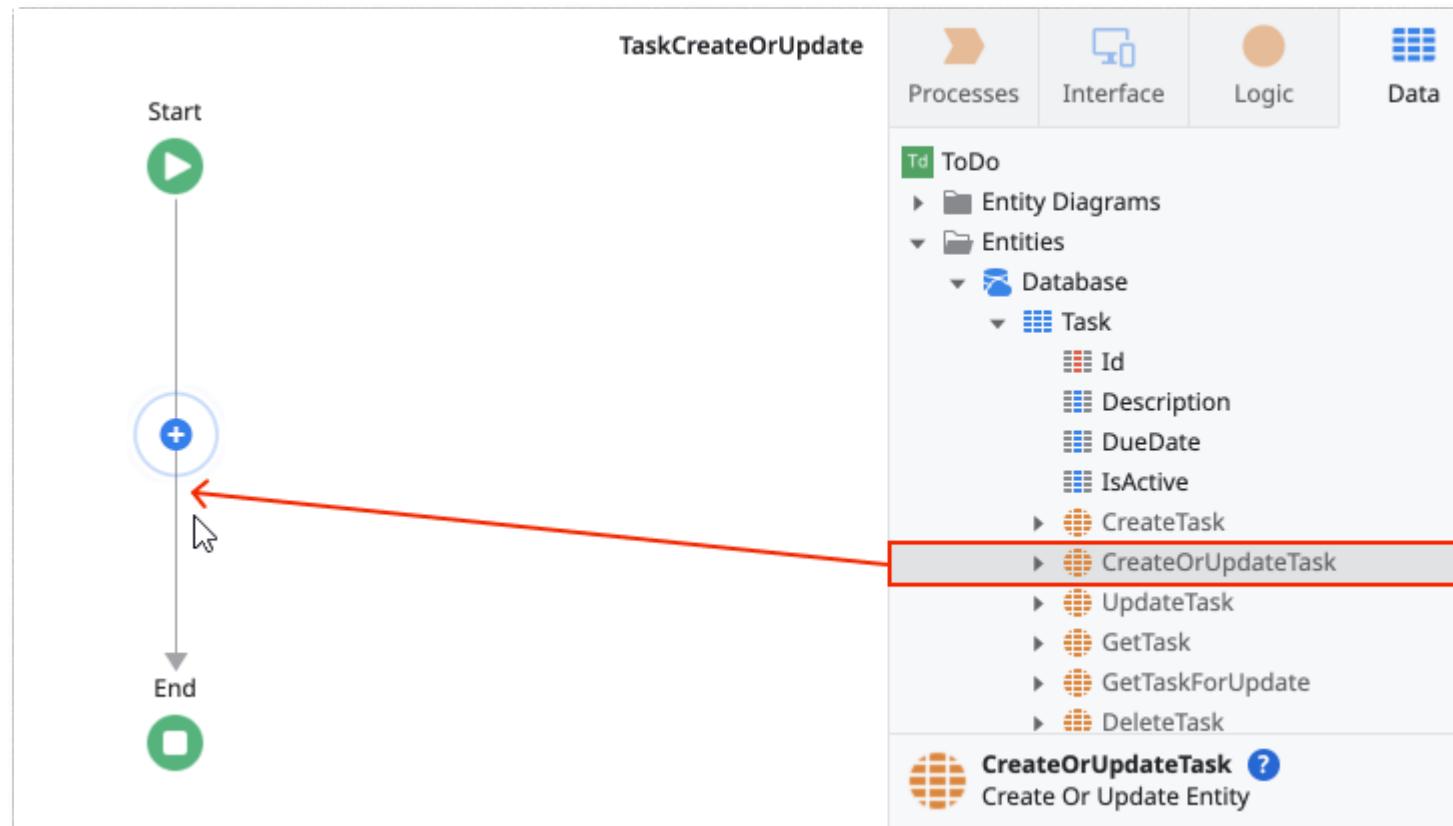
8. In the Input Parameter properties, set the Data Type to Task.

The screenshot shows the OutSystems Studio interface. On the left, a tree view under the 'ToDo' category shows various actions and tasks. The 'TaskCreateOrUpdate' action is selected, indicated by a blue highlight. In the main panel, the properties of this action are displayed. The 'Data Type' field is highlighted with a red border and contains the value 'Task'. Other visible properties include 'Name' (Task), 'Description' (empty), 'Is Mandatory' (Yes), and 'Default Value' (empty).

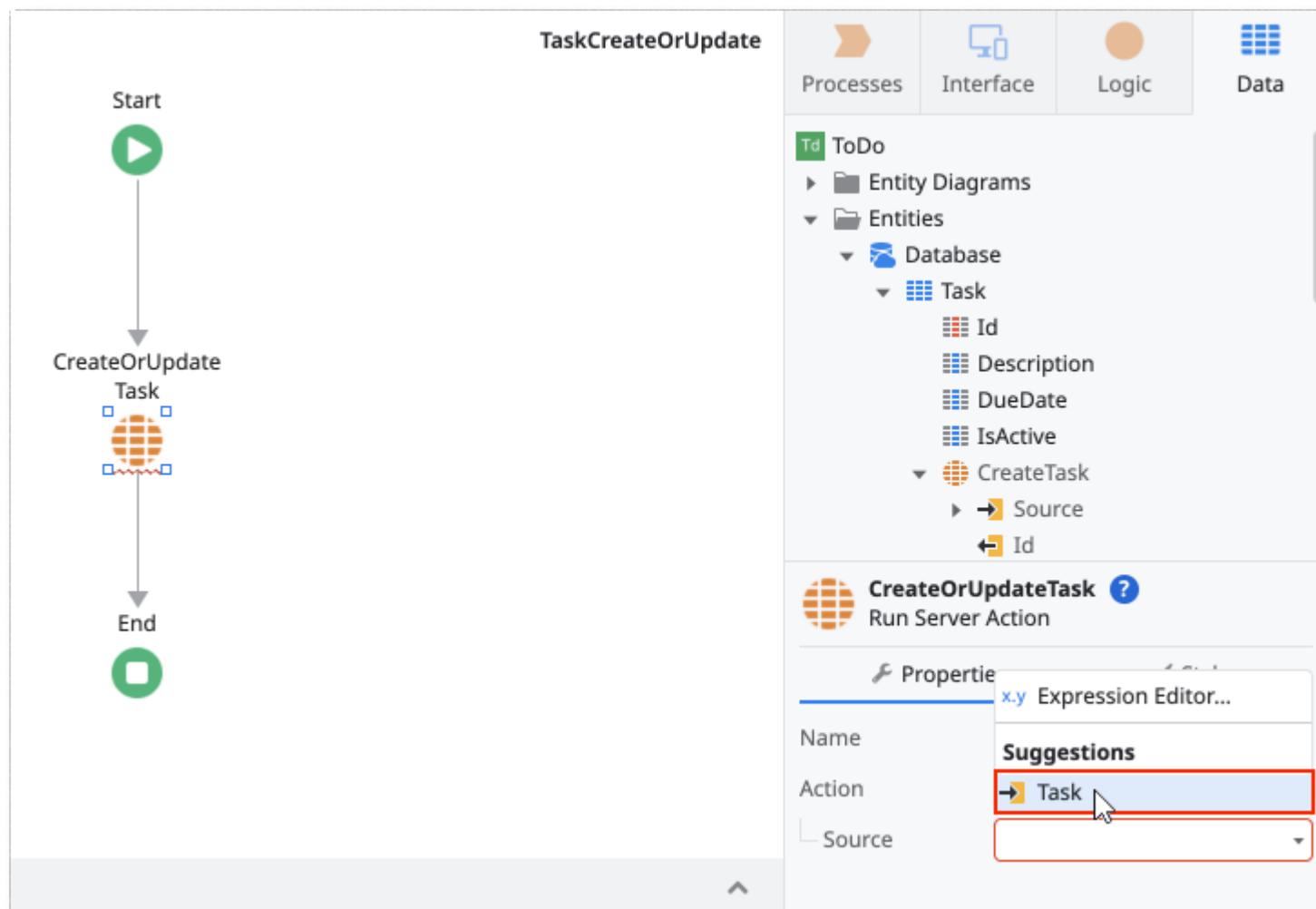
9. Right-click the **TaskCreateOrUpdate** action and select **Add Output Parameter**. Set its name to **TaskId**.
10. In the Output Parameter properties, set the Data Type to **Task Identifier**. This will be the task ID returned by the **CreateOrUpdateTask** that we'll need to pass on to the **SaveOnClick** action.

The screenshot shows the OutSystems Studio interface with the Data tab selected. In the left sidebar, under the 'Server Actions' section, the 'TaskCreateOrUpdate' action is expanded, and its 'Task' entity is selected. A specific output parameter, 'TaskId', is highlighted with a blue selection bar. Below the sidebar, a properties panel is open for the 'TaskId' parameter. The 'Properties' tab is active. The 'Name' field contains 'TaskId'. The 'Data Type' dropdown is set to 'Task Identifier' and is highlighted with a red box. The 'Default Value' field is empty.

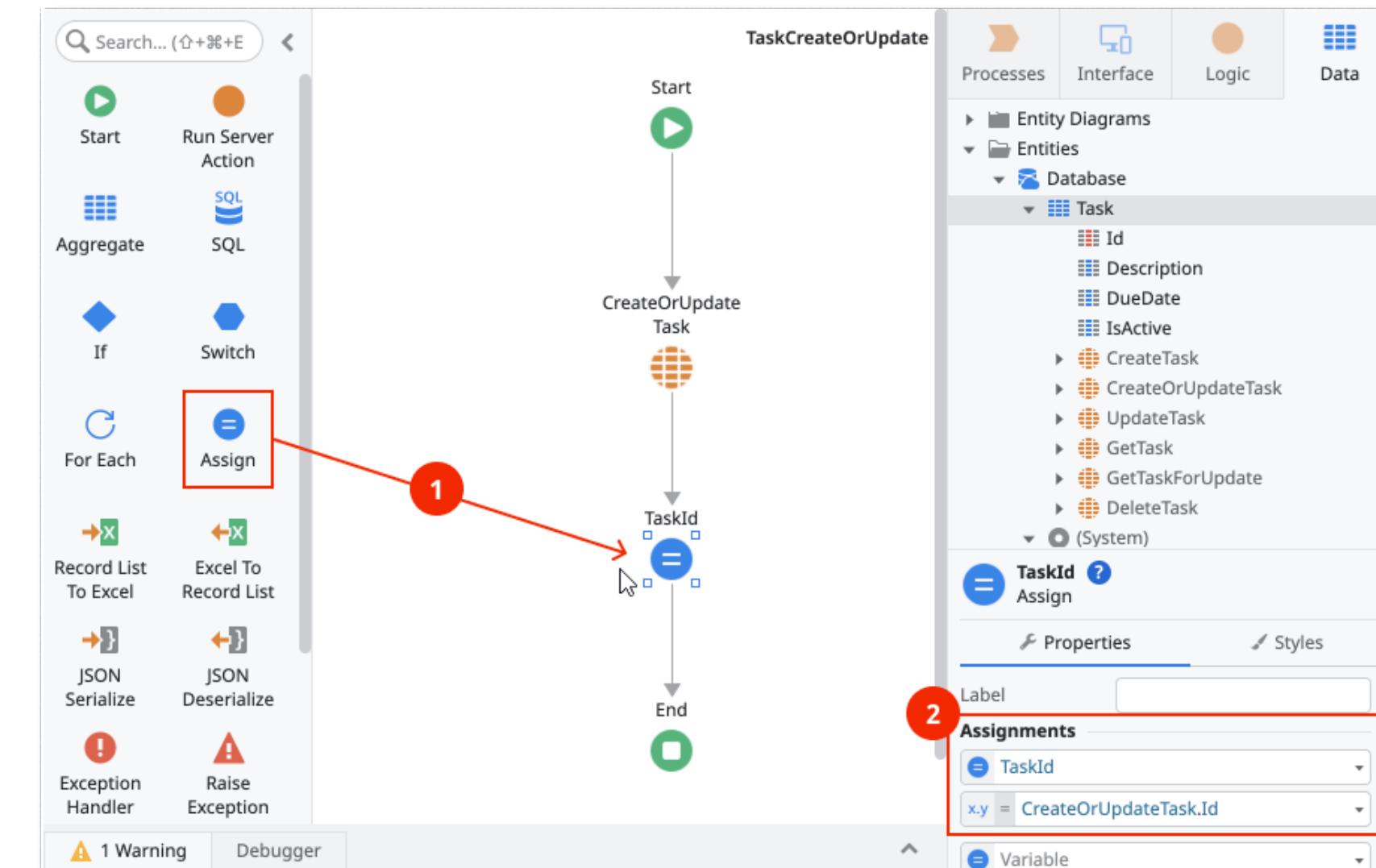
11. In the Data tab, expand the **Task** entity and drag the **CreateOrUpdateTask** entity action to the flow of the **TaskCreateOrUpdate** server action.



12. Set the Source to the Input Parameter Task.

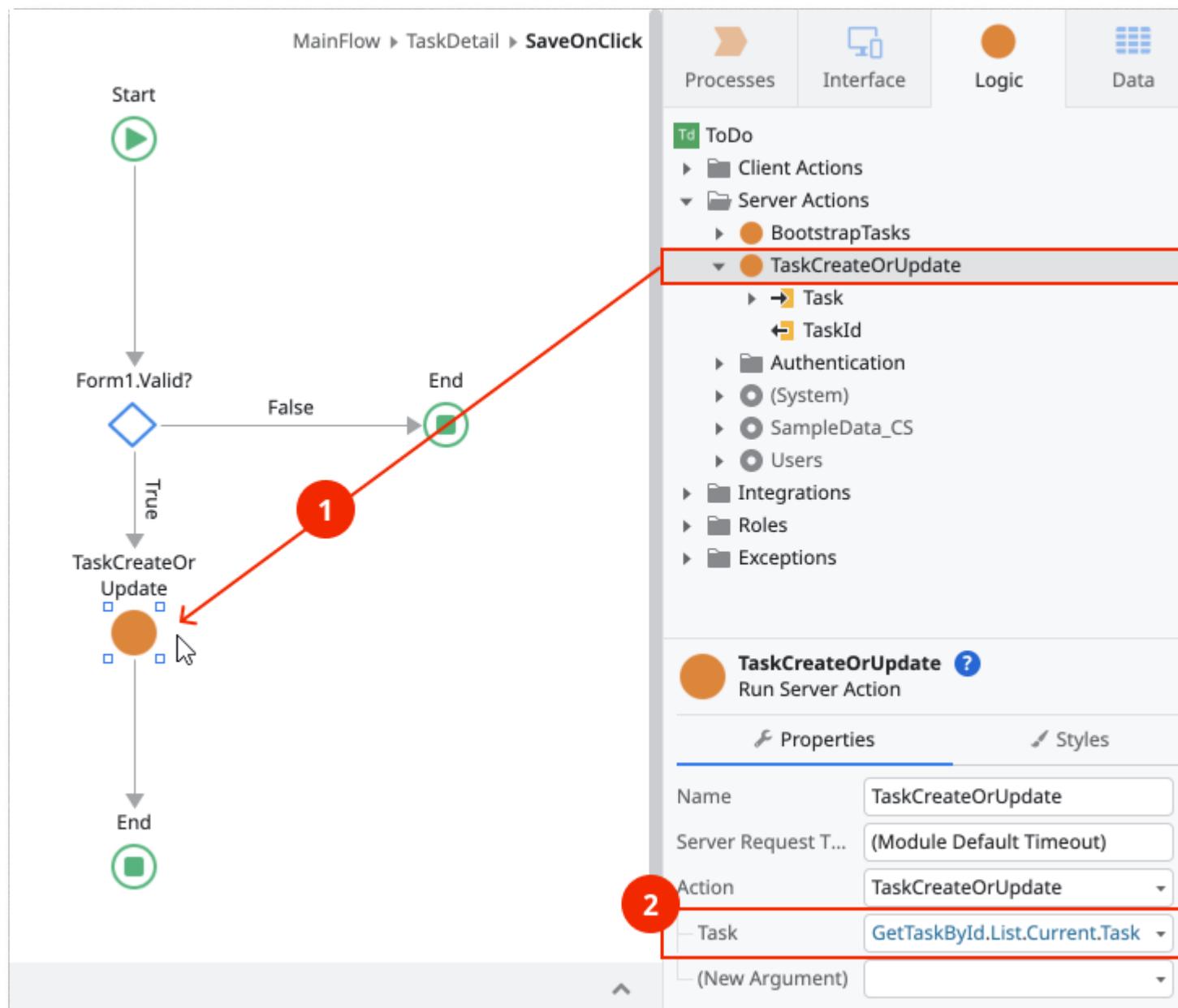


13. Next, we'll need to assign the value of the Output Parameter **TaskId** to the **CreateOrUpdateTask**. Drag an **Assign** node from the toolbox to the flow (1) and set the **Variable** to **TaskId**, and the **Value** to `CreateOrUpdateTask.Id` (2).

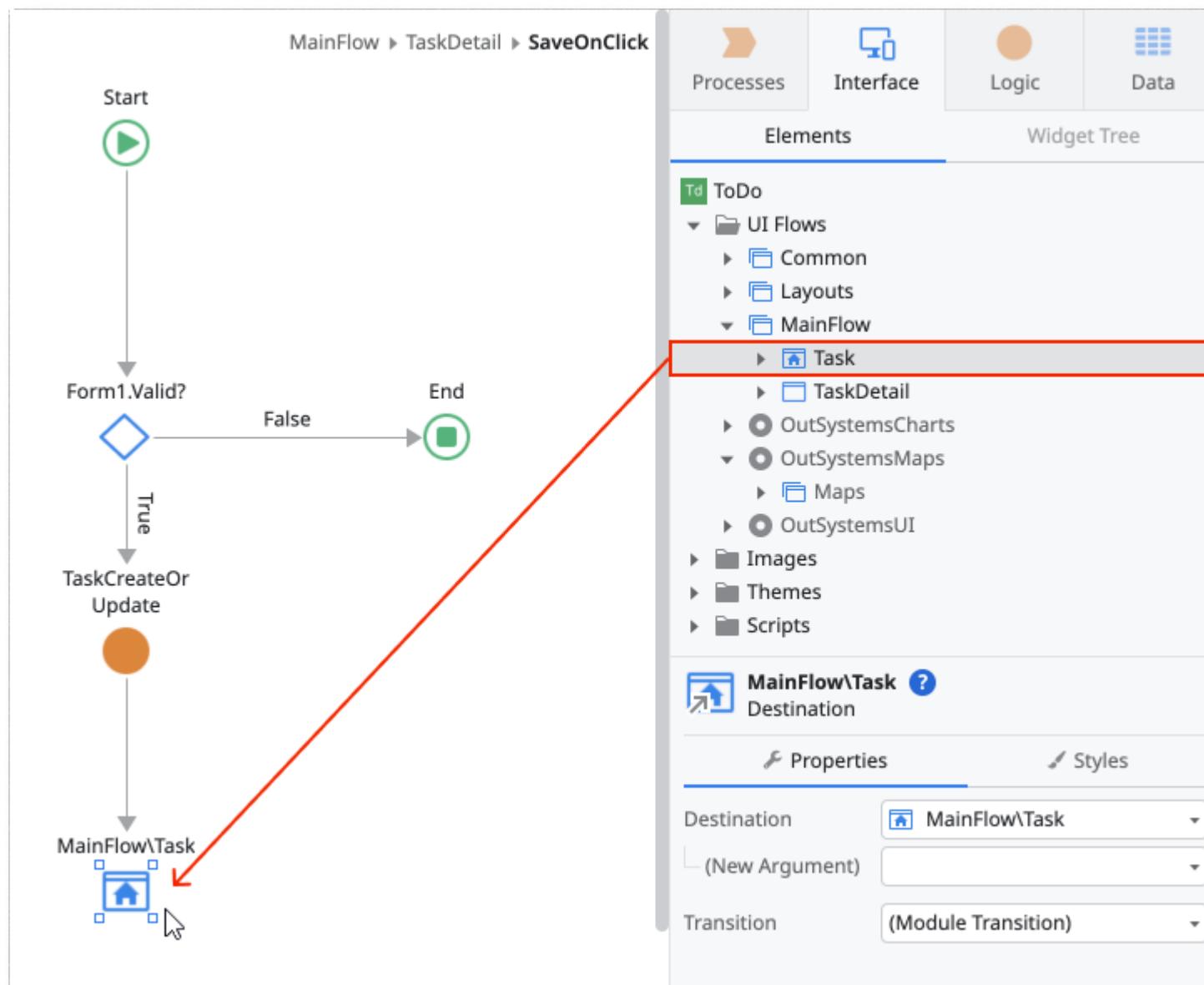


14. In the **Interface** tab, double-click the **SaveOnClick** action.

15. Navigate to the **Logic** tab and drag the **TaskCreateOrUpdate** server action to the **True** branch of the **If** (1). Set the **Task** property to `GetTaskById.List.Current.task` (2).



16. Drag the **Task Screen** from the **Interface** tab to the **End** node so that the user is redirected back to the main screen after saving a task.



## Allow completing tasks

Now let's add the functionality to mark tasks as complete. We can implement that by adding a feature to delete the completed task:

1. In the **Interface** tab, double-click the **Task Screen**.
2. Right-click the Checkbox in the **Is Active** column and select **Delete**.

The screenshot shows the OutSystems Studio interface for creating a reactive web application. The main area displays a table with three rows, each representing a task. The columns are labeled "Description", "Due Date", and "Is Active". The first two rows have the same data: "Finish Projects on Time" in the Description column, "1 Jan 1900" in the Due Date column, and a checked checkbox in the Is Active column. The third row also has the same data. Below the table, a message says "No items to show...". At the bottom, there are navigation buttons for "1 to 10 of 15 items" and a page number "1".

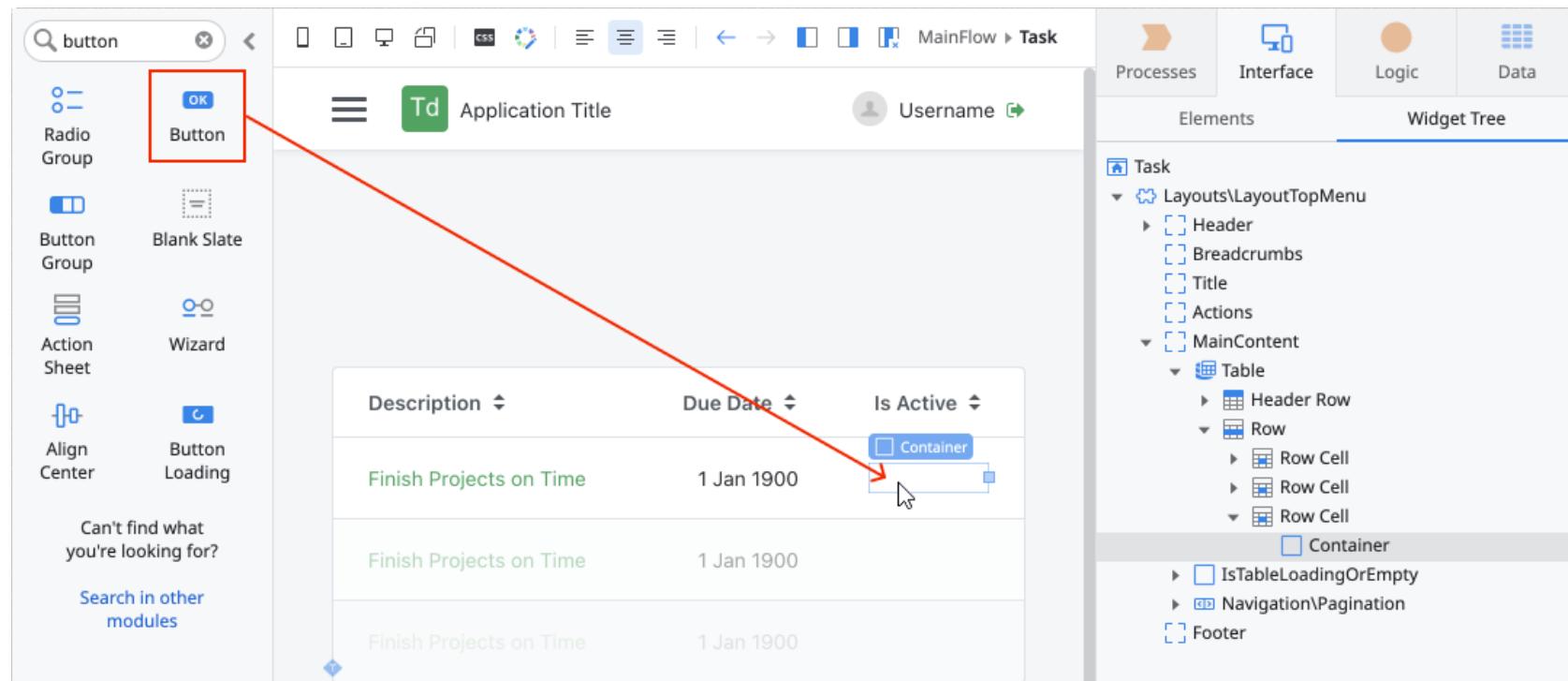
The top navigation bar includes icons for file operations (New, Open, Save, etc.), CSS, and database. The title bar shows "MainFlow > Task". To the right, there are tabs for "Processes", "Interface", "Logic", and "Data", with "Elements" currently selected. The "Widget Tree" tab is also visible.

A context menu is open over the third row of the table, listing options such as "Select Icon", "Remove Enclosing Container", "Enclose in Container", and "Enclose in If". The "Delete" option at the bottom of the menu is highlighted with a red box.

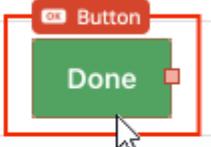
The "Elements" panel on the right shows a hierarchical tree structure of the application's components:

- Task
  - Layouts\LayoutTopMenu
    - Header
    - Breadcrumbs
    - Title
    - Actions
  - MainContent
    - Table
      - Header Row
      - Row
        - Row Cell
        - Row Cell
        - Row Cell
    - Container
      - Icon

3. Drag a **Button** widget to the same Container where the Checkbox was, and enter **Done** in the Text property of the button.



4. Double-click an empty area of the button to define the logic associated with the click.

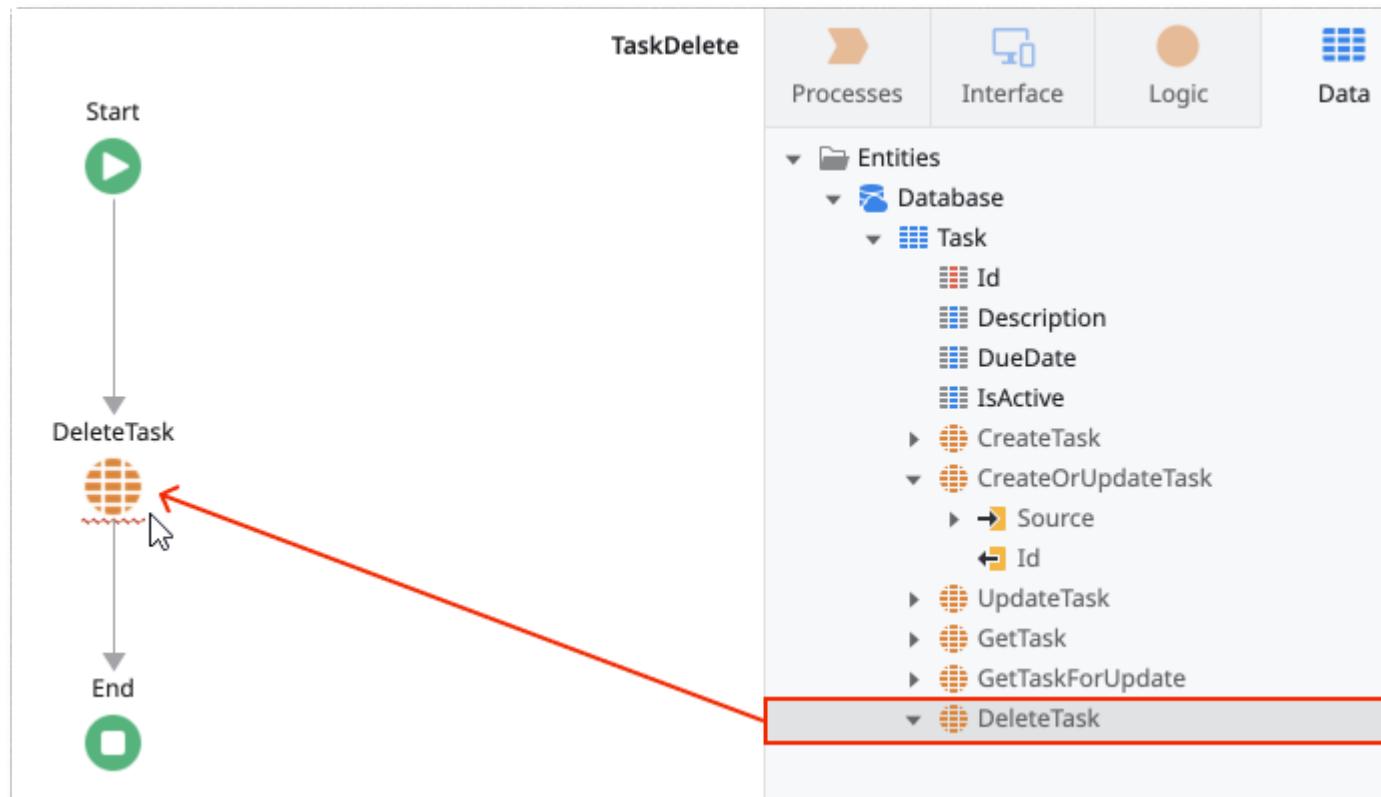
Description	Due Date	Is Active
Finish Projects on Time	1 Jan 1900	
Finish Projects on Time	1 Jan 1900	
Finish Projects on Time	1 Jan 1900	
No items to show...		

5. In the Logic tab, right-click the Server Actions and select Add Server Action. Name it TaskDelete.
6. Add an Input Parameter to the TaskDelete to receive the Task identifier. Set its name to TaskId and the Data Type to Task Identifier.

The screenshot shows the OutSystems Studio interface with the Data tab selected. The navigation bar at the top includes icons for Processes, Interface, Logic, and Data. Below the navigation bar, the 'ToDo' section is visible, followed by a tree view of actions. A red box highlights the 'TaskDelete' action under the 'Server Actions' category. The 'TaskId' input parameter is shown below, with its properties panel open. A second red box highlights the 'DataTask Identifier' data type for the 'TaskId' parameter. The properties panel also shows the parameter is mandatory.

Properties	
Name	TaskId
Description	...
Data Type	DataTask Identifier
Is Mandatory	Yes

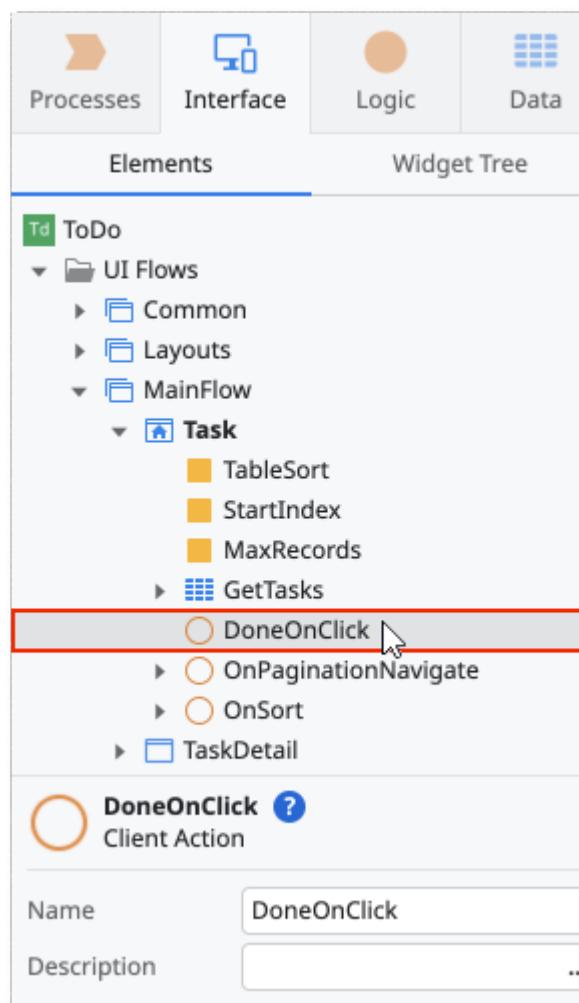
7. In the Data tab, expand the Task Entity. Drag the DeleteTask Entity Action to the flow.



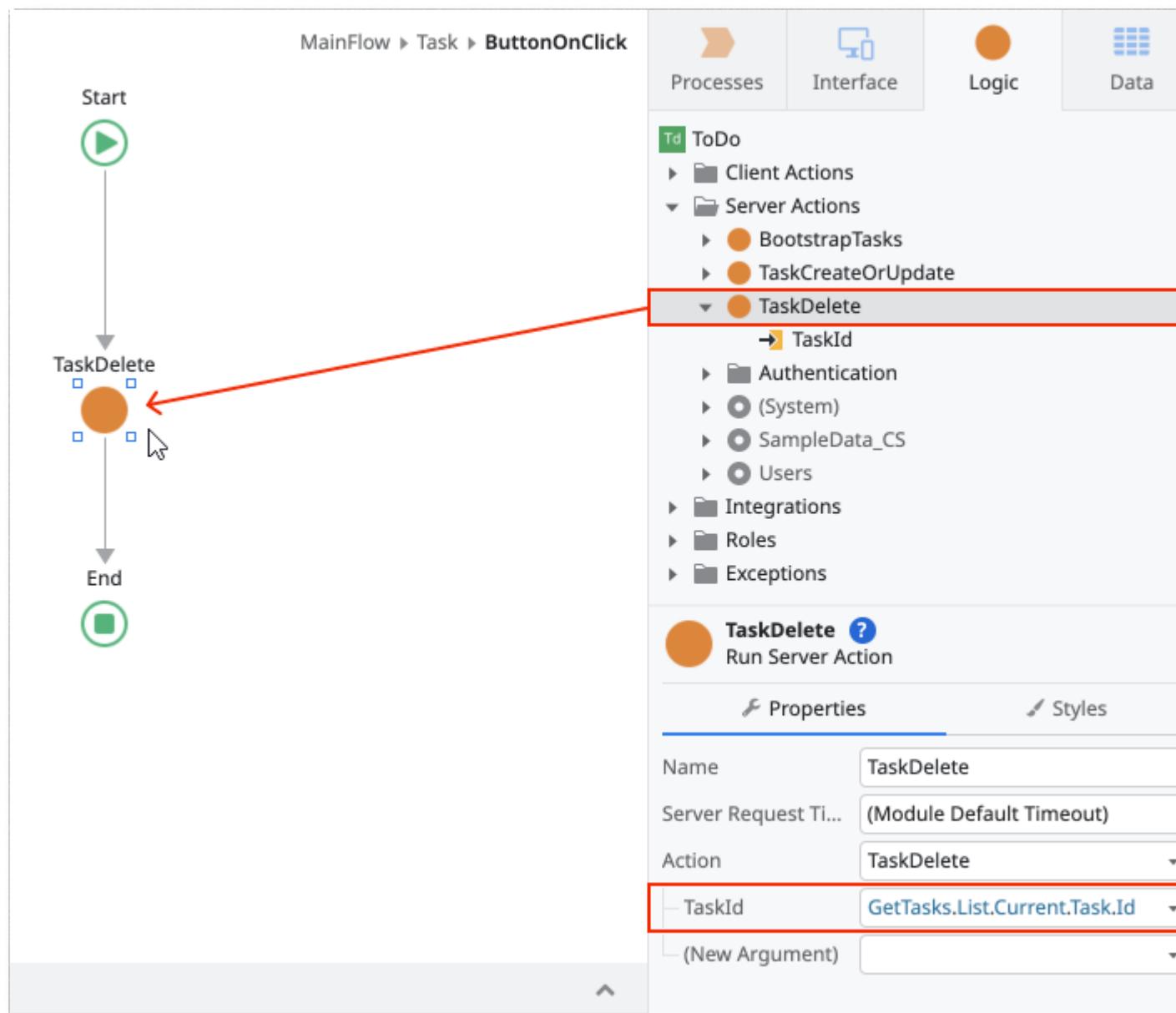
8. Set the **Id** property to the Input Parameter **TaskId**.

The screenshot shows the OutSystems Studio interface with the Data tab selected. On the left, a tree view displays the structure of entities and their fields. Under the 'Task' entity, the 'DeleteTask' action is selected. The properties panel shows the 'DeleteTask' action with an 'Id' property highlighted by a red border. A dropdown menu is open over the 'Id' property, listing suggestions such as 'TaskId' and 'NullIdentifier()'. The 'Properties' tab is currently active.

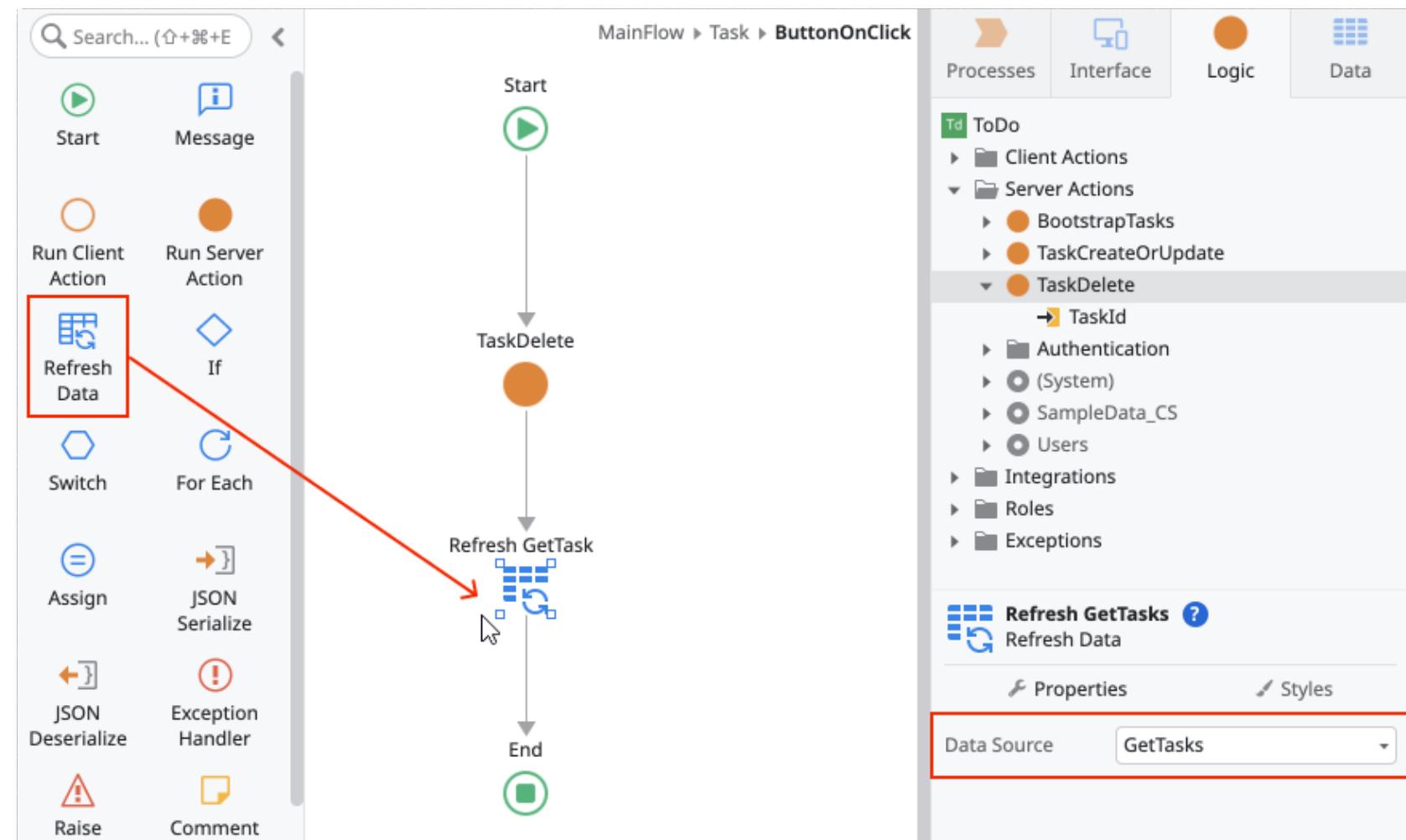
9. Go back to the **Interface** tab and double-click the **DoneOnClick** action under the **Task** screen.



10. Select the **Logic** tab and drag the **TaskDelete** server action to the flow of the **DoneOnClick** action. Set the **TaskId** property to `GetTasks.List.Current.Task.Id`.



11. Drag Refresh Data from the Toolbox to the action flow, after the TaskDelete action, and select the aggregate GetTasks to refresh the available tasks on the screen.

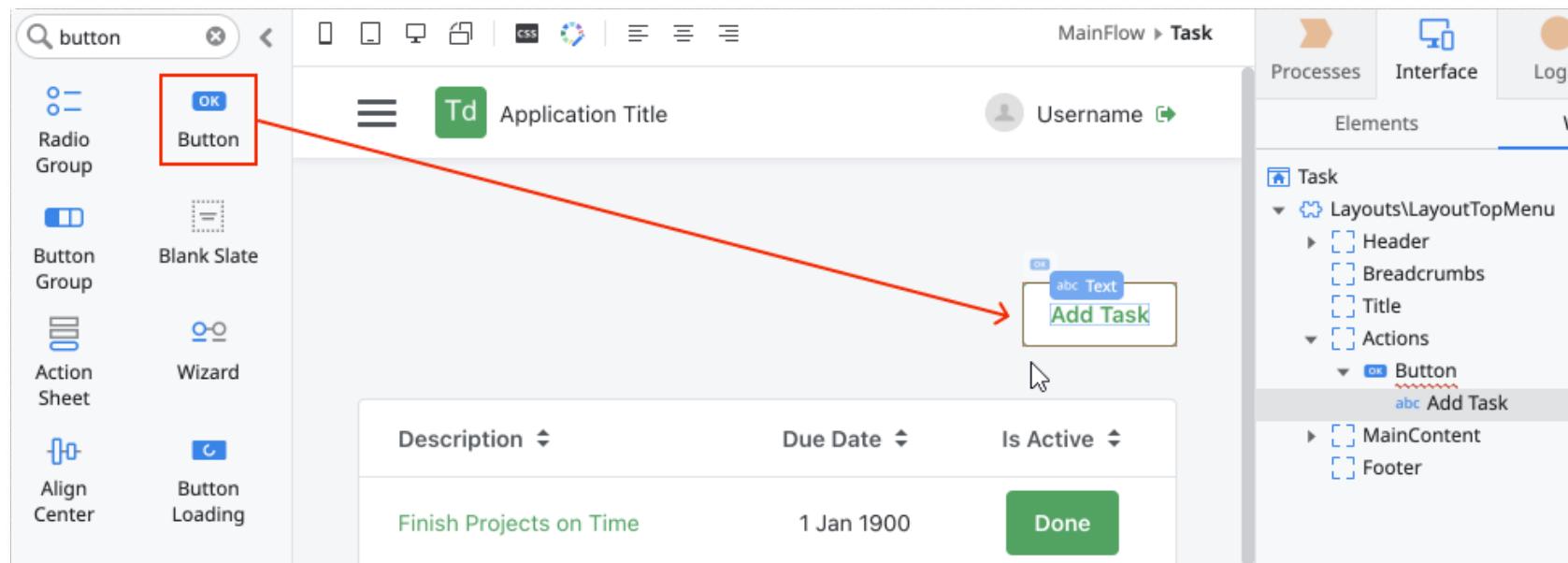


## Allow adding tasks

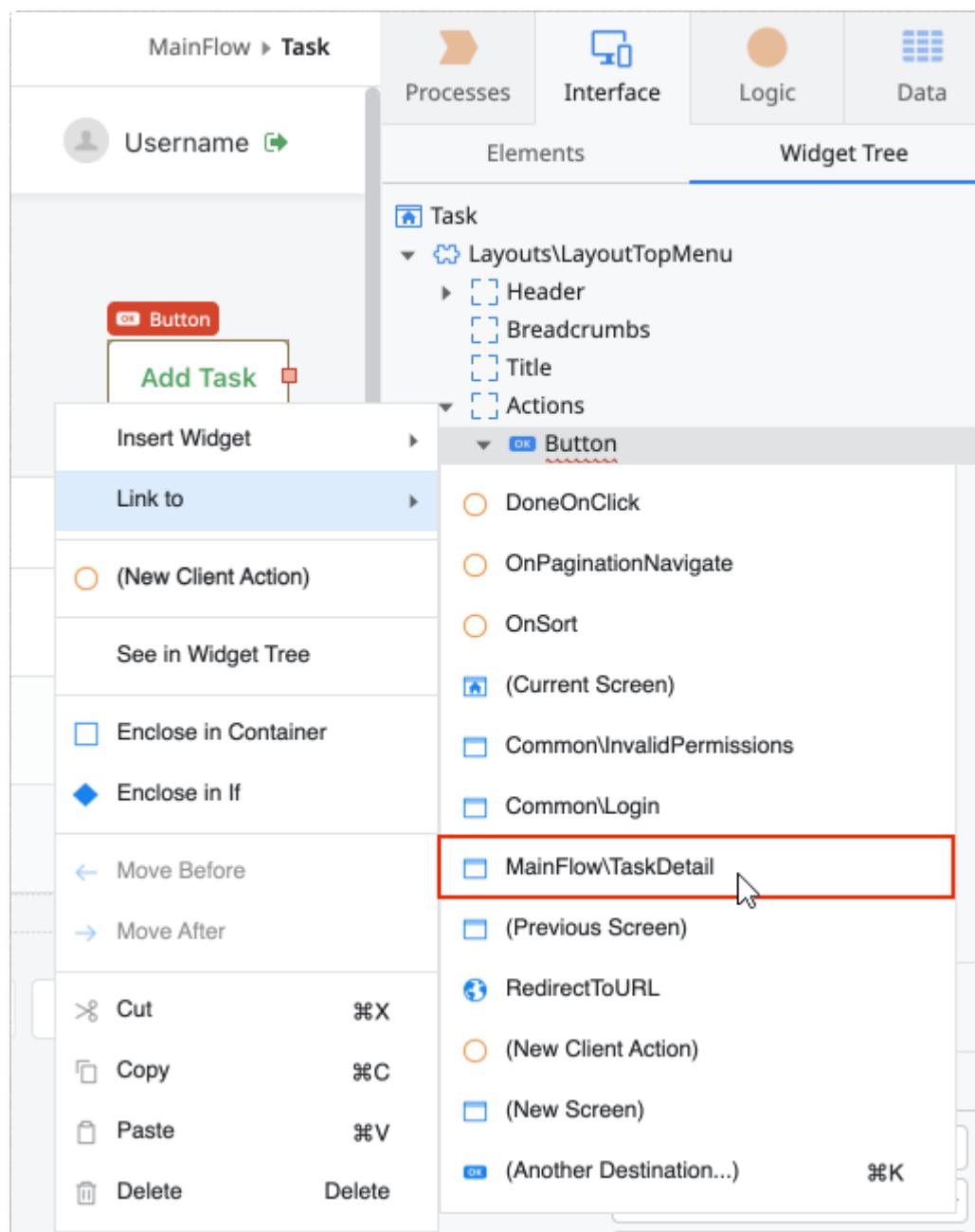
We also want to enable the end users to add new tasks from the screen with all tasks by linking to the screen that is already used to edit tasks:

1. Go to the **Interface** tab > **UI Flows** > **MainFlow**, and double-click the "Task" Screen to open the screen with all tasks in the main editor.

2. Drag a **Button** widget from the toolbox to the Actions placeholder in the top right-hand corner of the screen. Change the label of the button to **Add Task**.



3. Right-click an empty area of the button and choose **Link > MainFlow\TaskDetail**.



## Test your Reactive Web App

At this stage, you can test your Reactive Web App. Click the **1-Click Publish** button to publish the application to your environment. When the application is deployed, click the **Open in Browser** button to test your application in a browser.

The screenshot shows a web-based application interface for managing tasks. At the top left, there is a green button labeled "Td" and the word "ToDo". To the right of the button are two icons: a user profile icon and a circular arrow icon. On the far right, there is a green "Add Task" button. Below this header, there is a table with four columns: "Description", "Due Date", and "Is Active" (each with a dropdown arrow), followed by a "Done" button. The table contains four rows of task data:

Description	Due Date	Is Active
Finish Tutorial	6 Feb 2021	Done
Go On Vacation & Relax	4 Mar 2021	Done
Learn OutSystems	31 Jan 2021	Done
Finish Projects on Time	1 Mar 2021	Done

At the bottom left of the main content area, there is a message indicating "1 to 4 of 4 items".