Local Storage Exercise

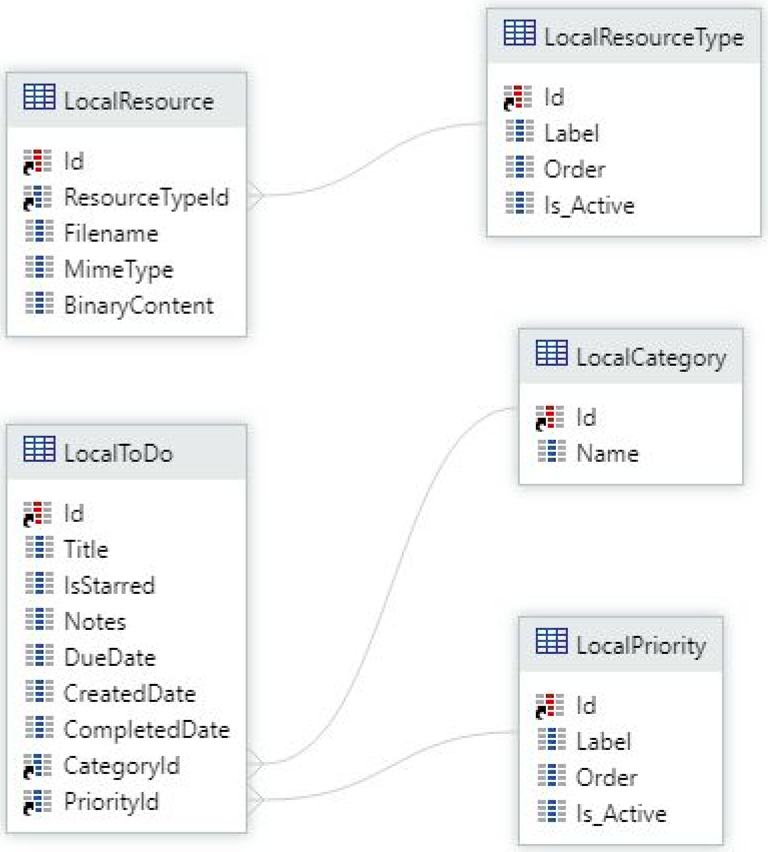


Table of Contents

Introduction

In this exercise lab, we will introduce Local Storage Entities to the ToDo application. Up to this point, all the data from the application was stored in the Database, thus requiring the application to always be online to be able to view or edit the ToDos.

To make sure the app accesses data while offline, we will start by creating five Local Storage Entities, with a similar structure of the existing Database Entities. The main difference in the Entities will be that there is no need to store the User Identifier of the ToDos locally, since the only ToDos the user will see are its own.

Then, to make sure that the Screens use the data from local storage, we will replace the Aggregates to fetch data from the database for new ones, with the new Local Storage Entities as sources. With this change, we will also adjust the Screens and logic to use the new Local Storage Aggregates.

Finally, we will also add some logic to create and update ToDos in local storage, using the Entity Actions of the newly created Entities.

As a summary, in this specific exercise lab, we will:

- Create Local Storage Entities
- Replace the usage of Entities for Local Storage Entities in the application Screens
- Add and update ToDos in local storage

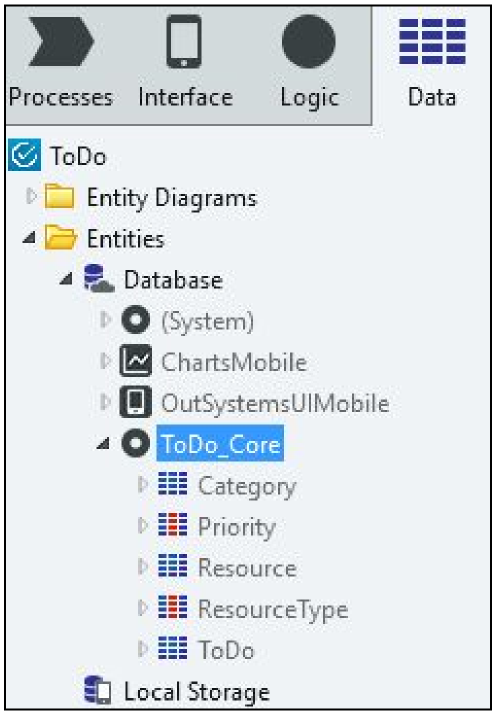
Create Local Entities

We will start this exercise lab by creating the Local Entities. We will do that by leveraging the Database Entities that already exist.

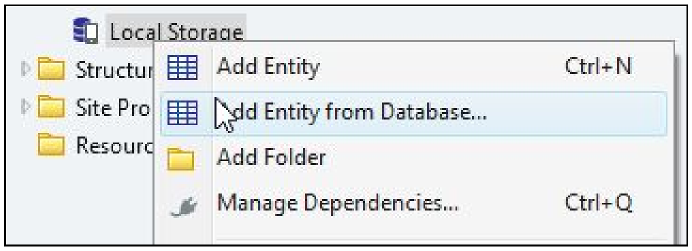
In general, we want to create the Local Storage Entities at the image of the Database Entities, meaning that they will keep the same structure. There is only one difference. The Local Storage Entity for the ToDos will not need the UserId, since that can be later assigned server-side, when adding a ToDo to the Database. This way, the Local Storage for the ToDos will only hold the ToDos of the currently logged in user.

These Local Storage Entities will be created in the ToDo module.

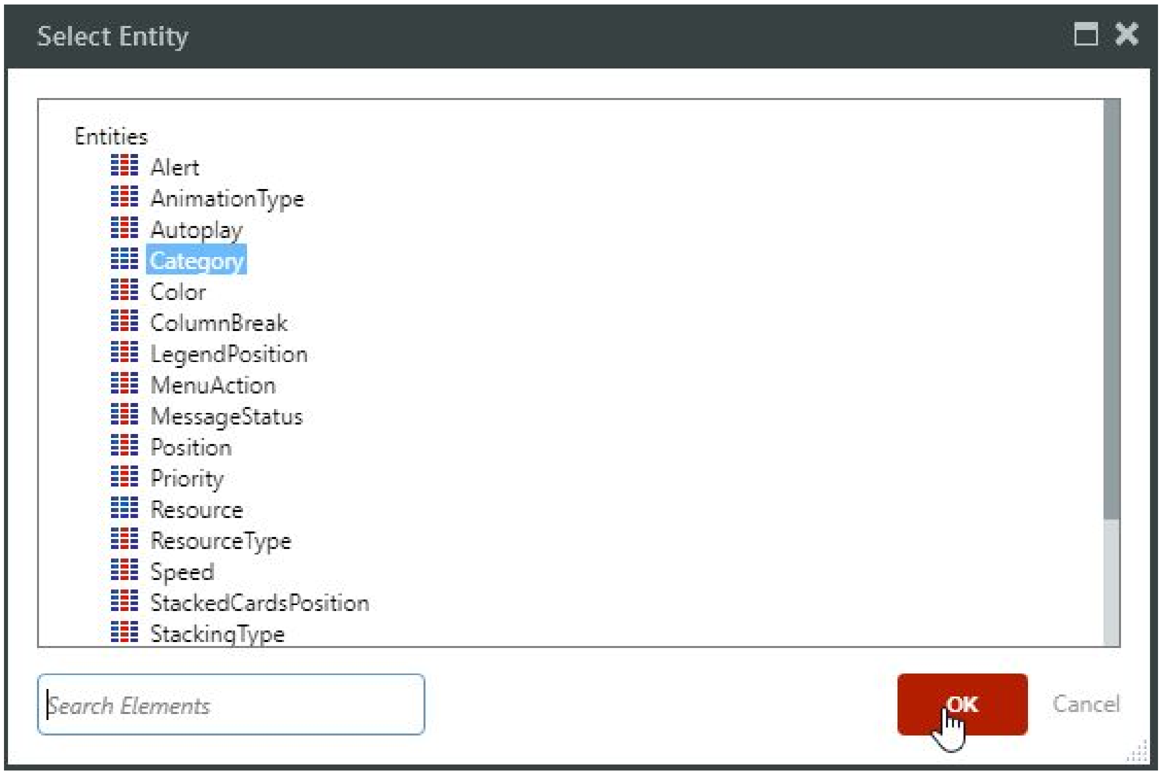
1) In the ToDo module, switch to the Data tab and expand the **ToDo_Core** module to view the existing Database Entities.



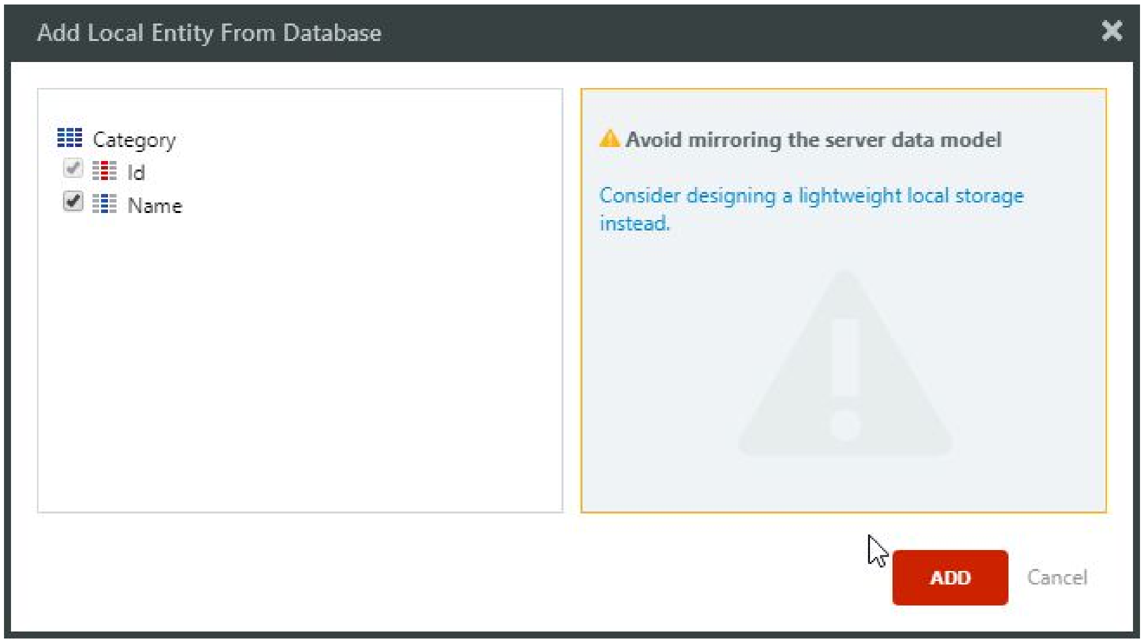
2) Right-click the **Local Storage** element and choose the option **Add Entity from Database**.



3) In the **Select Entity** dialog, choose the **Category** Entity and then click **OK**.



4) In the next dialog, select all the Entity attributes and click the **Add** button.



Avoid mirroring the server data model

Consider designing a lightweight local storage instead.

5) Notice that a new *LocalCategory* Entity was created and has the same structure as the **Category** Database Entity.

NOTE: Notice that there is a warning to avoid mirroring the server data model in local storage. This is to raise awareness for performance issues that may arise from synchronizing large amounts of data between the server and local storage. In this case, we need to have both attributes in Local Storage as well.

---

6) Repeat the previous steps to create the *LocalPriority*, *LocalResource*, *LocalResourceType* and *LocalToDo* Local Storage Entities. During this process, select **all** the attributes from the Entities, except for the **UserId** attribute of the **ToDo** Entity.

NOTE: The **Data Type** of the Entity Identifiers from Local Storage have the Database Entity Identifier type.

Local Entities exist in the device of each user, therefore there is no need to also have the **UserId** attribute, as it would match the identifier of the logged in user. In the **LocalToDo** Entity, we will only store data from the current logged in user.

Fetch Data from Local Storage

Now that we have created the Local Storage Entities, in this section we will start using them on Screens. For that, we will replace the Aggregates in the ToDos and ToDoDetail Screens by Local Storage Aggregates, to fetch data from local storage, instead of from the Database.

This will create several errors in the application, since the Screens were built using the Database Entities, and now we will fetch data from Local Storage. These errors can be easily fixed using the TrueChange tab to guide us.

On this section, we will not yet add data to Local Storage. We will only replace the source of data from database to local storage.

1) Adapt the ToDos Screen to fetch data from the Local Storage, instead of using the Database. Fix all errors that will arise from this change, by adapting the UI and logic to use the new Local Storage Aggregates.

    a) Switch to the Interface tab and expand the **ToDos** Screen.

    b) Delete the **GetToDosByUserId** Aggregate.

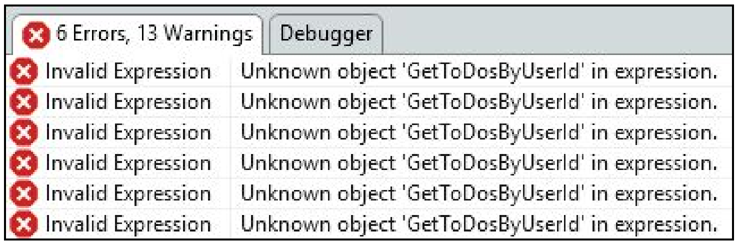
    NOTE: At this point, you will see several errors in the **TrueChange** tab. We will fix them later during this exercise.

    c) Right-click the ToDos Screen and choose *Fetch Data from Local Storage*.

    d) From the Data tab, drag the **LocalToDo** Entity and drop it inside the Aggregate editor.

    NOTE: The Local Storage Aggregate is created with the name *GetToDos*. This happens because the **LocalToDo** Entity was created based on the **ToDo** Database Entity, which has a customized plural label *ToDos*. As it is a customized label, the new LocalToDo Entity inherits it. Entities with automatic / default plural labels will have the Local prefixed to the plural, e.g. Local Categories.

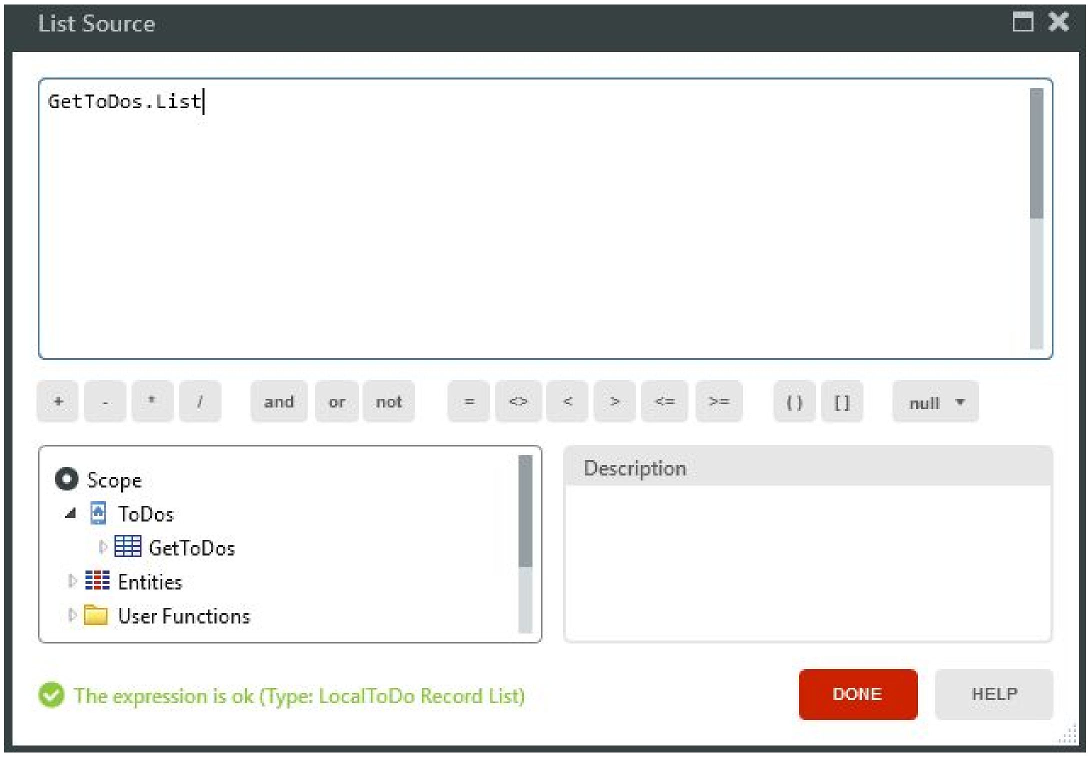
    e) Open the TrueChange tab to view the existing errors.



Invalid Expression	Unknown object 'GetToDosByUserId' in expression.
Invalid Expression	Unknown object 'GetToDosByUserId' in expression.
Invalid Expression	Unknown object 'GetToDosByUserId' in expression.
Invalid Expression	Unknown object 'GetToDosByUserId' in expression.
Invalid Expression	Unknown object 'GetToDosByUserId' in expression.
Invalid Expression	Unknown object 'GetToDosByUserId' in expression.

These errors were caused by the deletion of the **GetToDosByUserId** Aggregate. To fix them, we need to replace the references to this Aggregate by referencing the new one, **GetToDos**.

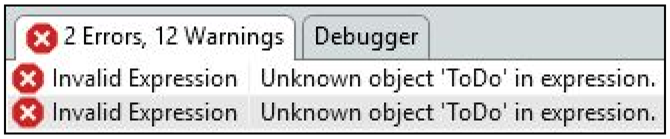
f) Double-click each one of the errors and replace the **GetToDosByUserId** for *GetToDos*, in all Expressions.



```
GetToDos.List
```

The expression is ok (Type: LocalToDo Record List)

g) After fixing the errors, the **TrueChange** tab should display two new errors.



Invalid Expression	Unknown object 'ToDo' in expression.
Invalid Expression	Unknown object 'ToDo' in expression.

NOTE: These new TrueChange errors were caused because the Source of the **GetToDos** Aggregate is the Local Storage Entity **LocalToDo**, while the Aggregate **GetToDosByUserId** had as Source the Database Entity **ToDo**. This means that replacing the Aggregate in the Expressions is not enough, since we also need to change the ToDo references to LocalToDo.

h) Double-click each one of these new errors and replace the **ToDo** by *LocalToDo*. At this point there are a lot of warning listed. Those will be dealt with later.

2) Modify the **ToDoDetail** Screen to fetch data from the Local Storage instead of using the Database.

   a) Under the MainFlow, expand the **ToDoDetail** Screen.

   b) Delete the **GetCategories** Aggregate.

   c) Right-click the **ToDoDetail** Screen and choose *Fetch Data from Local Storage.*

   d) From the Data tab, drag the **LocalCategory** and drop it inside the Aggregate Editor.

   e) Rename the **ToDoId** Input Parameter of the ToDoDetail Screen to *LocalToDoId* and change the **Data Type** to *LocalToDo Identifier*.

   f) Delete the **GetToDoById** Aggregate.

   g) Right-click the **ToDoDetail** Screen and choose *Fetch Data from Local Storage*.

   h) Drag the **LocalToDoId** Input Parameter to the Aggregate Editor.

   i) Fix the errors that appear by replacing the **GetToDoById** references by *GetLocalToDoById*, **ToDo** with *LocalToDo*, **GetCategories** with *GetLocalCategories* and **Category** by *LocalCategory*.

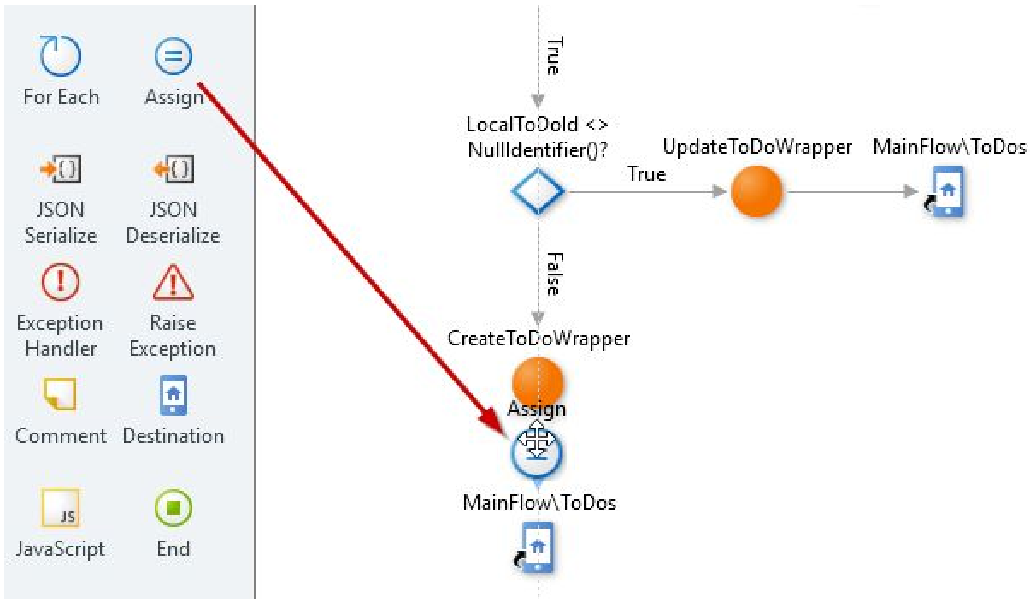
3) Publish the ToDo module and notice that no ToDos appear on the Screen. That happens because there are no ToDos stored in local storage yet, but we are already fetching data from local storage. We will add ToDos to the local storage on the next section.

Modify Data in Local Storage

1) Modify the **SaveOnClick** Action in the ToDoDetail Screen to create / update a ToDo in the Local Storage (LocalToDo Entity).

   a) Open the **SaveOnClick** Client Action of the ToDoDetail Screen.

   b) Drag an Assign statement and drop it between the CreateToDoWrapper and the Destination statement.



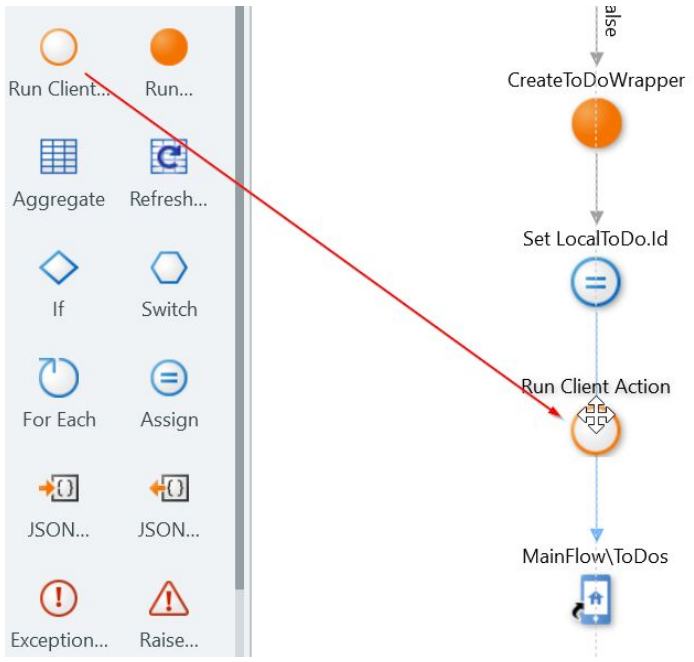
   c) Define the following assignment in the new Assign statement

      GetLocalToDoById.List.Current.LocalToDo.Id = CreateToDoWrapper.ToDoId

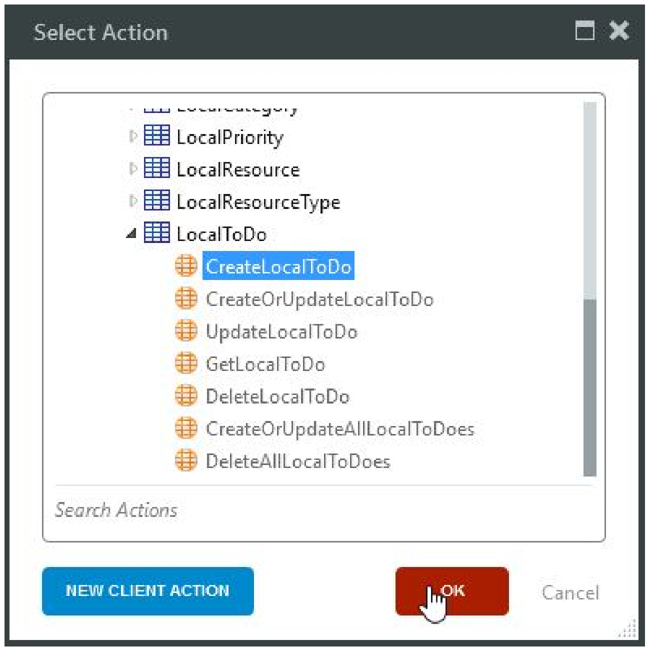
   To make sure that the LocalToDo has the same identifier as the ToDo created in the database, it is important to have this assignment that makes the Id of the LocalToDo equal to the Output Parameter of the CreateToDoWrapper Action. Going back to the creation of this Wrapper, its Output Parameter was assigned to hold the Id of the ToDo created in the database, thus giving us what we need for this assignment.

   d) Set the **Label** property of the Assign to *Set LocalToDo.Id*

   e) Drag a **Run Client Action** statement and drop it between the Assign and the Destination statements.

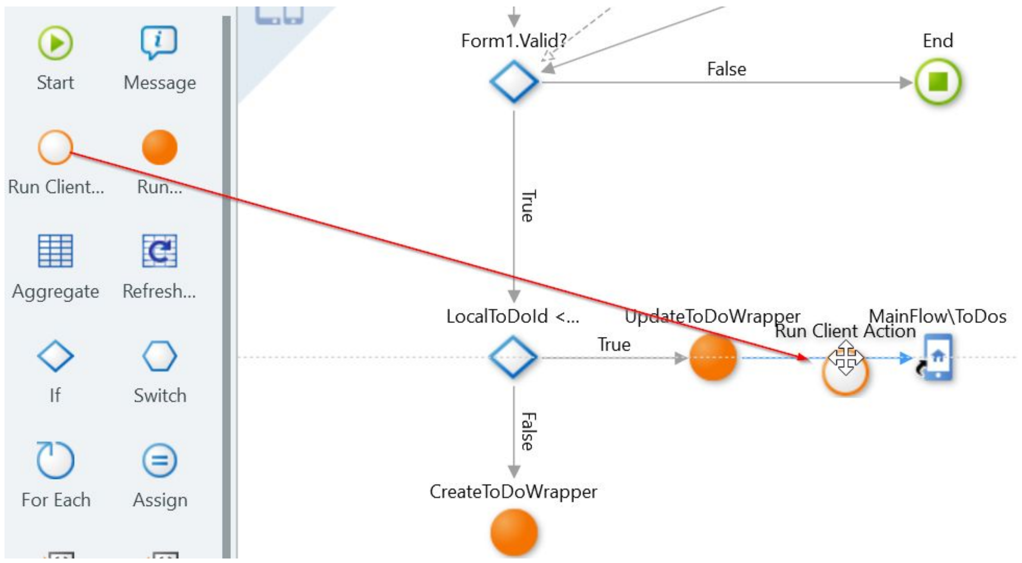


f) In the Select Action dialog, select the **CreateLocalToDo** Action.



g) Set the Source property to *GetLocalToDoById.List.Current*

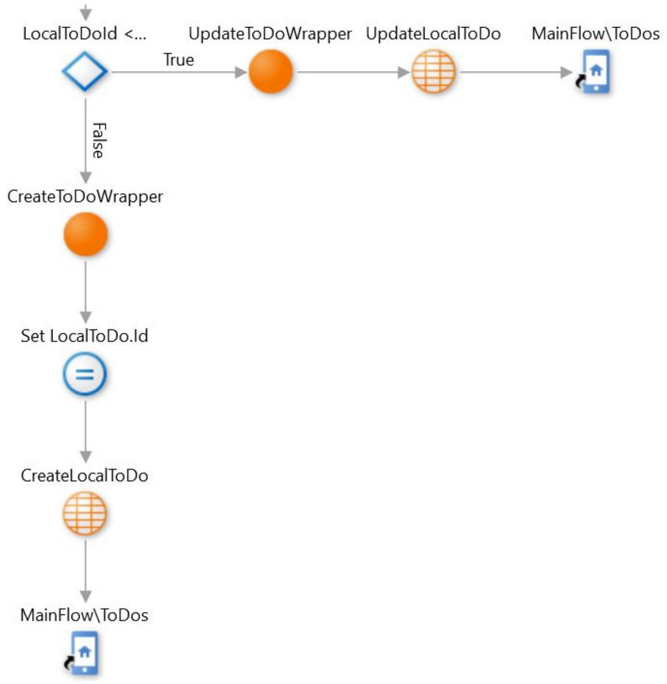
h) Drag a **Run Client Action** between the UpdateToDoWrapper and the Destination statement.



i) In the Select Action dialog, select the **UpdateLocalToDo** Action. In this case, we do not need the assignment because we are updating a LocalToDo that already exists, thus it has an Id already.

j) Set the Source property to *GetLocalToDoById.List.Current*

k) The final part of the flow should look like the following screenshot



2) Modify the **StarOnClick** Action of the ToDoDetail Screen to also update the Local Storage, following the same strategy of the previous step.

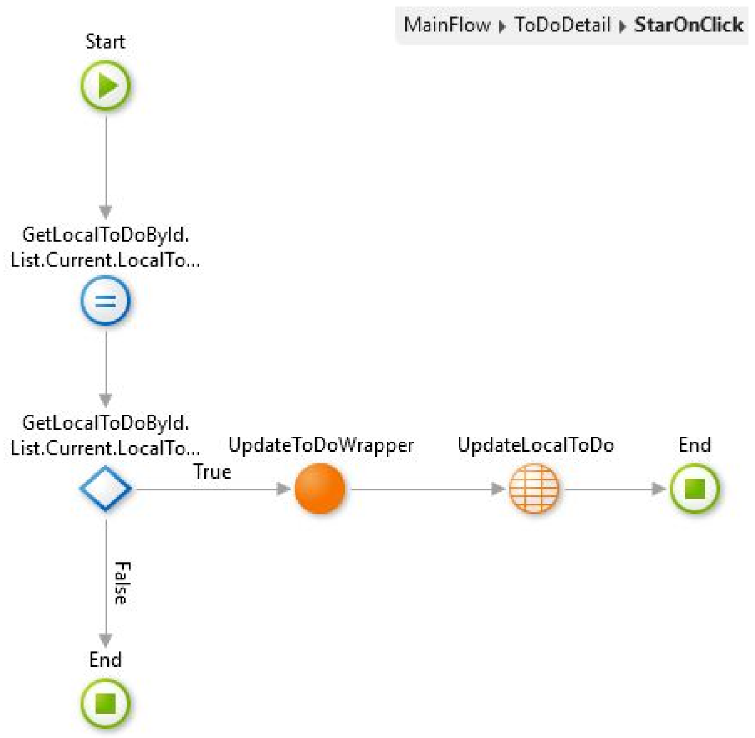
a) Open the **StarOnClick** Action of the ToDoDetail Screen.

b) Drag a **Run Client Action** statement and drop it between the UpdateToDoWrapper and the End statements.

c) In the Select Action dialog, select the **UpdateLocalToDo** Action.

d) Set the **Source** property of the UpdateLocalToDo to *GetLocalToDoById.List.Current*

e) The StarOnClick Action should look like this



3) Publish the module and open the application in the browser. Try to create a ToDo. Notice that no Categories appear on the Screen, since there are no Categories in Local Storage. Since the Category is a mandatory field, we are still not able to create a ToDo in local storage. On the next lab, we will implement the data synchronization that will solve this problem.

End of Lab

In this exercise, we started by creating Local Storage Entities, from the existing Database Entities, keeping almost exactly the same structure. This is an accelerator that Service Studio provides to help us easily map the structure of the Database Entities to the Local Storage Entities.

With the new Entities, we then changed the logic and UI of the Screens to use the Local Storage instead of the database Entities.

Finally, we added the logic to create or update ToDos in the local storage.

At this point, there are still no ToDos in the local storage, however in the next lab, we will implement the data synchronization logic, which will enable that the ToDos in the database are saved in the local storage and vice-versa.