



DEVELOPING OUTSYSTEMS MOBILE APPS

Upload and Download



Introduction

Over the course of this set of exercise labs, you will create a web 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 To Do application will be able to access all of this information. This back-office application will allow administrators to manage all existing To Dos.

You constantly will be expanding your application, publishing it to the server and testing your application to the server while learning and applying new OutSystems concepts.

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

In this specific exercise lab, you will:

- Enable users to upload Resources
- Enable users to download Resources

Table of Contents

Introduction.....	2
Table of Contents.....	3
Part 1: Add References to 'Resources'	4
Part 2: Add 'Upload Resource' feature.....	6
Part 3: Add 'Download Resource' feature	17
End of Lab	22
List of Figures.....	23

Part 1: Add References to 'Resources'

In this part of the exercise, you will add references to the **Resource** Entity and **ResourceType** Static Entity, as well as to another Server Action built-in with the platform.

1. Add references to the **Resource** Entity and **ResourceType** Static Entity.

- a) Click the **Manage Dependencies...** button from the toolbar to open the Manage Dependencies dialog.

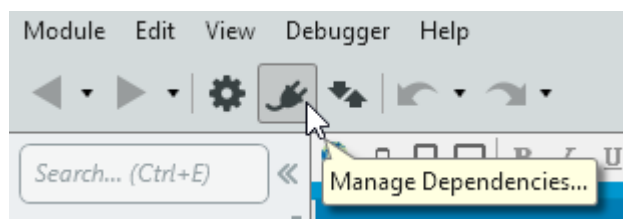


Figure 1. Open Manage Dependencies... dialog

- b) In the **Manage Dependencies** dialog, select 'Show All' in both drop downs of Producers and Public elements, then select the your **ToDo_Core** module, and tick the checkboxes for the **Resource** Entity and **ResourceType** Static Entity.

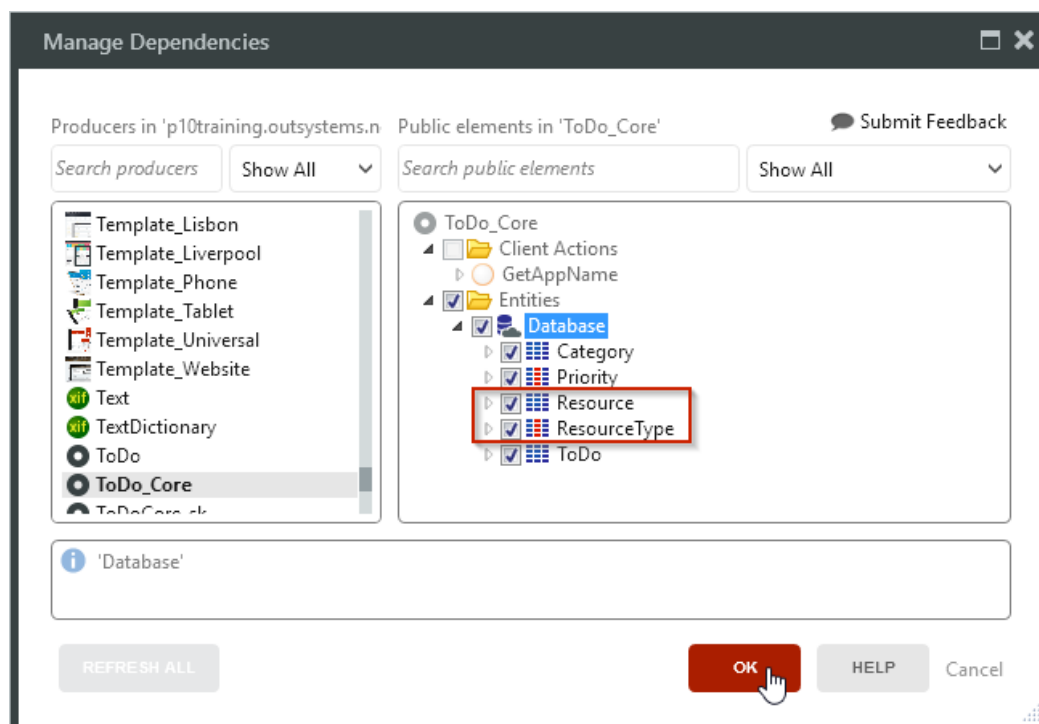


Figure 2. Add References to Resource and ResourceType Entities

- c) Since we will be dealing with the upload file mechanism, we will also need an extra Server Action already built-in with the Platform. In the list of **Producers** on the left, select **BinaryData**. Then, tick the checkbox of the **BinaryDataSize** Server Action.

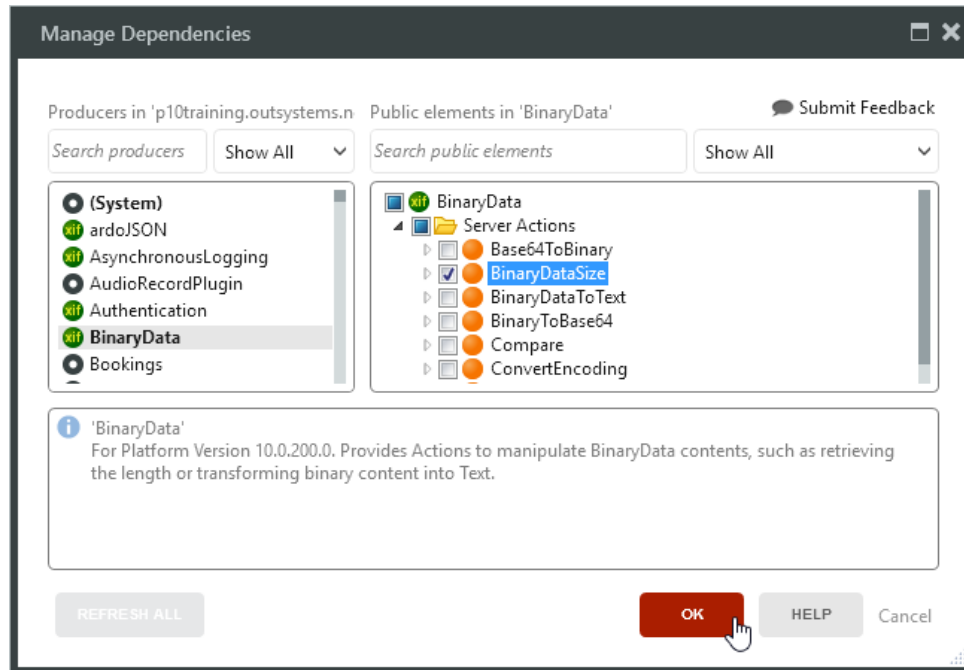


Figure 3. BinaryDataSize reference

NOTE: The **Binary** extension module provides several Server Actions, to manipulate and obtain information from Binary Data values.

- d) Click **Ok** to update the module references.
- e) In the **Data** tab, you should now see the newly added Entity references under the **ToDo_Core** module, and in the **Logic** tab you should see the **BinaryDataSize** Server Action, under the **BinaryData** Extension.

Part 2: Add 'Upload Resource' feature

In this part of the exercise, you will add a file Upload Widget to the **ToDoDetail** Screen, that will enable end-users to upload resources to attach to the To Dos. Resource files will be stored in the database **Resource** Entity.

1. Add a File **Upload** Widget to the **ToDoDetail** Screen.

- a) Switch to the **Interface** tab, and open the **ToDoDetail** screen.
- b) Drag a new **Container** Widget and drop it between **Notes** and **Due Date**.

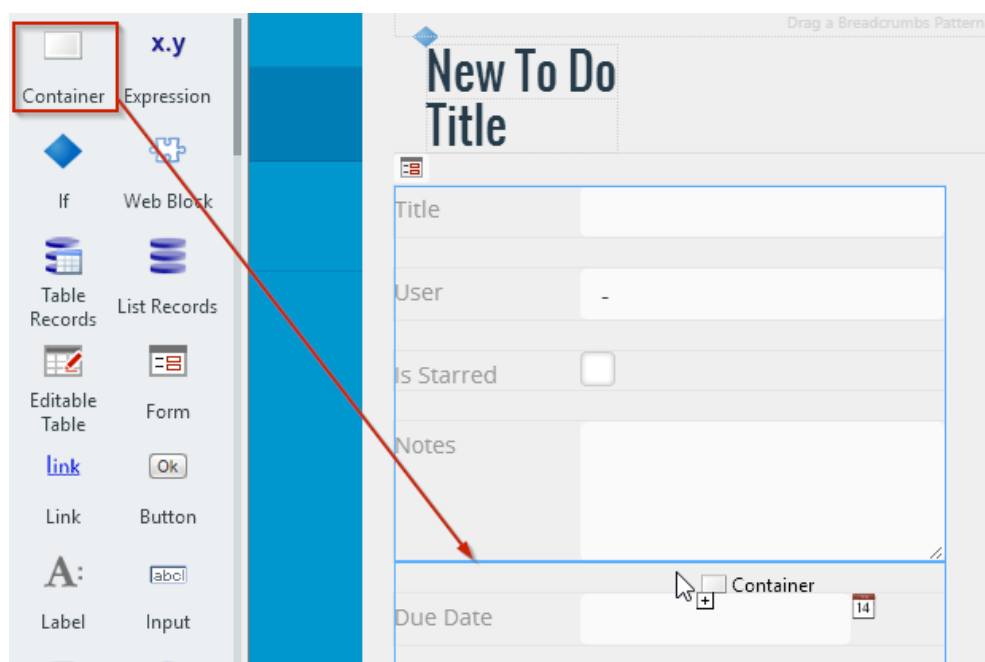


Figure 4. Drag and drop a new Container Widget

- c) Drag a new **Label** Widget, and drop it inside the Container created in the previous step.

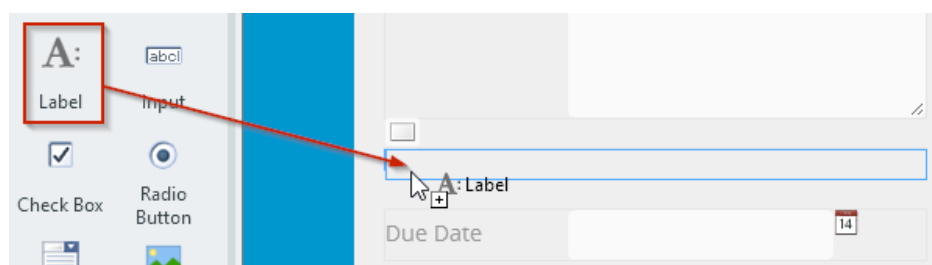


Figure 5. Drag and drop a new Label Widget

- d) Drag an **Upload** Widget, and drop it to the right of the **Label** created in the previous step, but still inside the same Container.

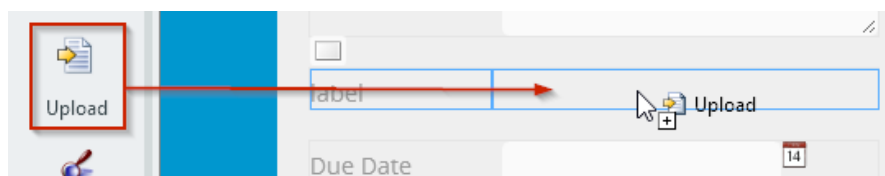


Figure 6. Drag and drop a new Upload Widget

- e) Select the **Label** created previously, and set the **Value** property to “Resource”.
- f) From the **Input Widget** property drop down select **(Select Value...)**.

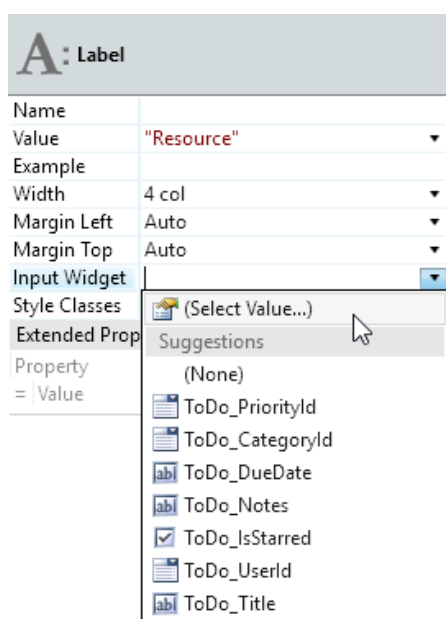


Figure 7. Label Input Widget property drop down

- g) In the **Select Input Widget** dialog, select **Upload1** and click **Ok**.

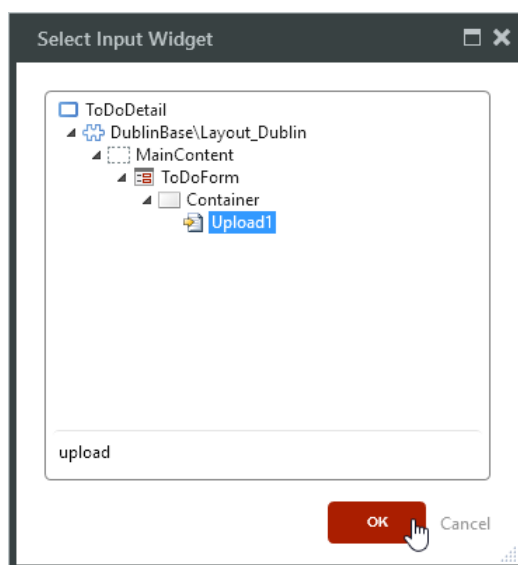


Figure 8. Select Upload1 as Input Widget

NOTE: The association between **Labels** and **Input** Widgets, allow users to move the focus of the browser easily to **Input** widgets. When an end-user clicks the text in the **Label**, the browser will automatically move the focus to the associated **Input** widget improving usability.

- h) Rename the **Upload1** Widget to 'UploadResource', by selecting it first and then editing the **Name** property.

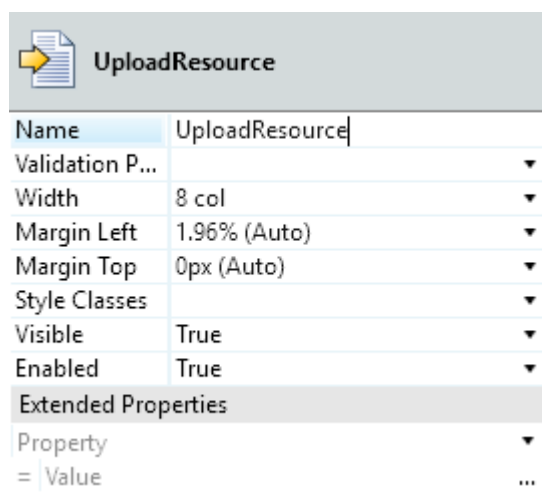


Figure 9. Upload Widget properties

2. Change the **Save** Screen Action logic to also store the uploaded resource.

- a) Open the **Save** Screen Action, either by double clicking the **Save** Button, or by expanding the **ToDoDetail** Screen, in the **Interface** tab, and double clicking the **Save** Screen Action.
- b) Recall that this Screen Action was created automatically by the Scaffolding process, when you created the **ToDoDetail** Screen. The flow of the **Save** Screen Action should look like this

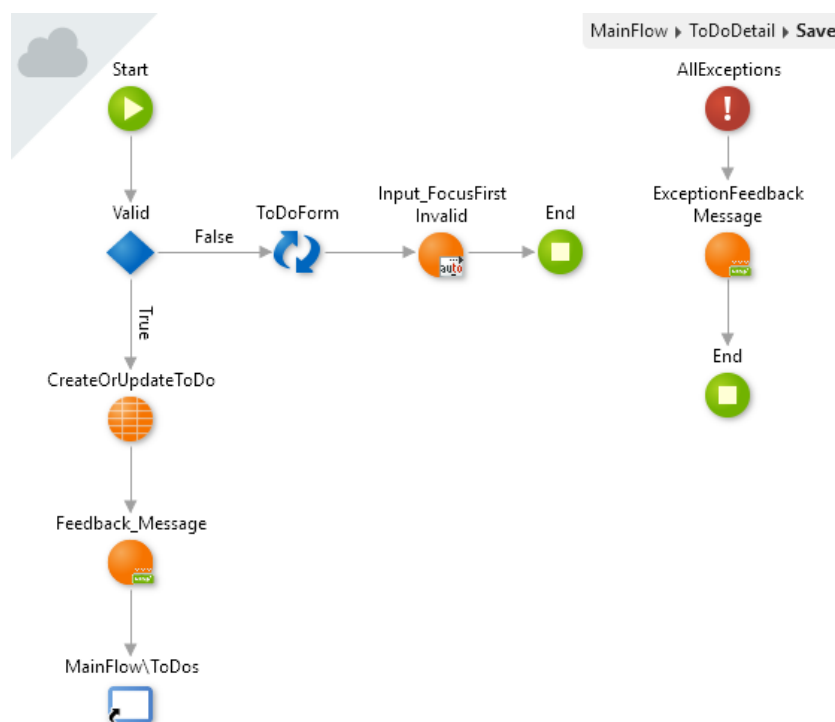


Figure 10. Save Screen Action created by Scaffold

NOTE: All logic and Screens created by Scaffolding patterns can be modified to better fit your application requirements.

- c) Select the **Feedback_Message** and **Destination** node and move them down to create some space between the **Feedback_Message** and **CreateOrUpdateToDo** nodes, where you will add more logic.
- d) Drag an **If** statement and drop it after the **CreateOrUpdateToDo**.

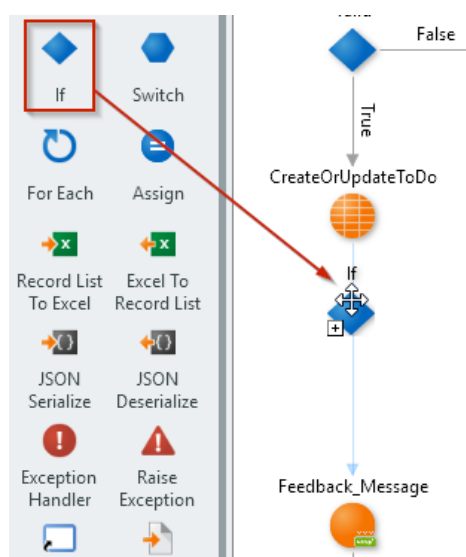


Figure 11. Drag and drop an If statement

- e) Double click the **If** statement to open the Expression Editor and set the **Condition** property to

```
BinaryDataSize (UploadResource.Content) > 0
```

- f) Drag an **Assign** statement and drop it to the right of the **If** statement.

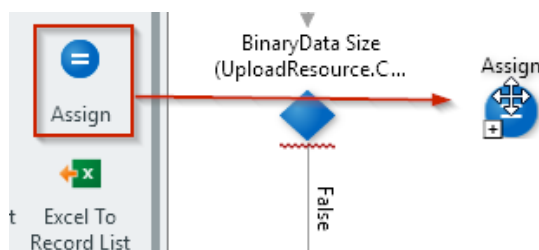


Figure 12. Drag and drop an Assign statement

- g) Click the **If** statement and create the **True** connector into the **Assign** statement, defining the flow of the **True** branch of the **If**.
- h) From the **Data** tab, expand the **Resource** Entity to view the six available Entity Actions.
- i) From the **Data** tab, drag the **CreateOrUpdateResource** Entity Action located under the **Resource** Entity of the **ToDo_Core** module, and drop it on the right of the **Assign** statement.

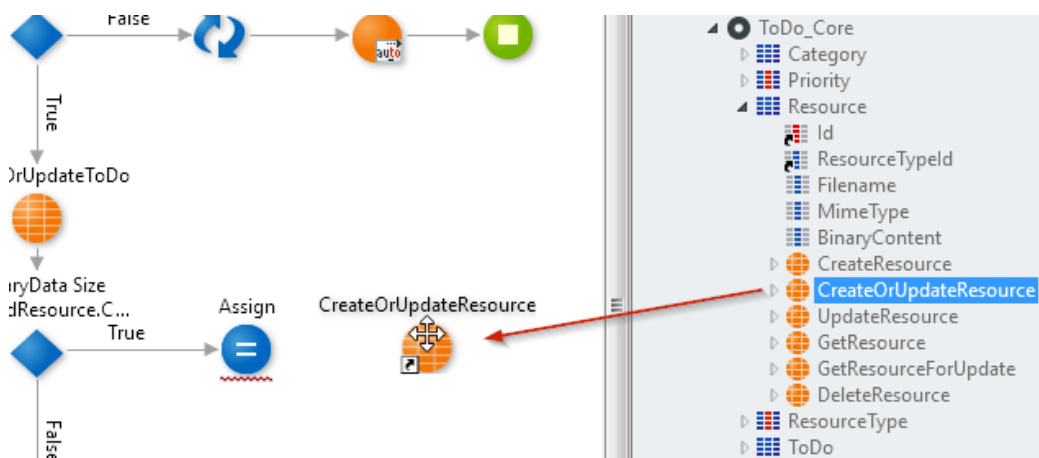


Figure 13. Drag and drop CreateOrUpdateResource Entity Action

- j) Create the connector from the **Assign** to the **CreateOrUpdateResource** statement, and another from the **CreateOrUpdateResource** to the **Feedback_Message**.

NOTE: The logic inserted into the **Save** Action will verify if a file has been uploaded (**If**). If no file was uploaded, then the logic executes as before, through the **False** branch. When a file is uploaded, then we need to store it into the **Resource** Database Entity. To do this, you will assign the required data into a variable that will then be used in the **CreateOrUpdateResource** Entity Action.

- k) In the **Interface** tab, right click the **Save** Action from the **ToDoDetail** Screen, and select **Add Local Variable**.
- l) Set the Local Variable **Name** to 'Resource' and verify that the **Data Type** is set to **Resource** (Entity).

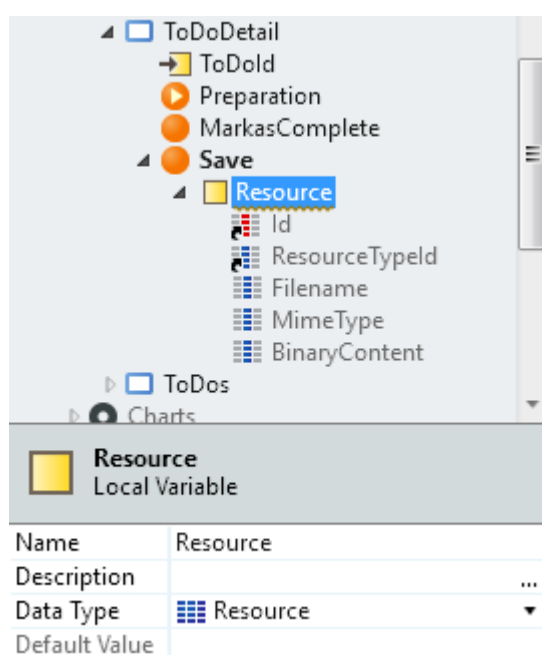


Figure 14. Resource Local Variable properties

- m) Select the **Assign** statement inside the **Save** Screen Action, then define the following assignments

```
Resource.Id = CreateOrUpdateToDo.Id
Resource.Filename = UploadResource.Filename
Resource.MimeType = UploadResource.Type
Resource.BinaryContent = UploadResource.Content
```

NOTE: Recall that the **Save** Action is used when you are creating new To Dos or updating existing ones.

When you are creating new To Dos, the **ToDoId** Input Parameter and **ToDoForm.Record.ToDo.Id** will both have the `NullIdentifier()` value. When editing an existing To Do, both variables will have the Identifier of the To Do being edited.

The **CreateOrUpdateToDo.Id** value will have, when creating a new To Do the new To Do Identifier, and when editing an existing To Do will have the Identifier of the To Do being edited.

- n) Add another assignment for the variable 'Resource.ResourceTypeId', and from the drop down select **(Expression Editor...)**.

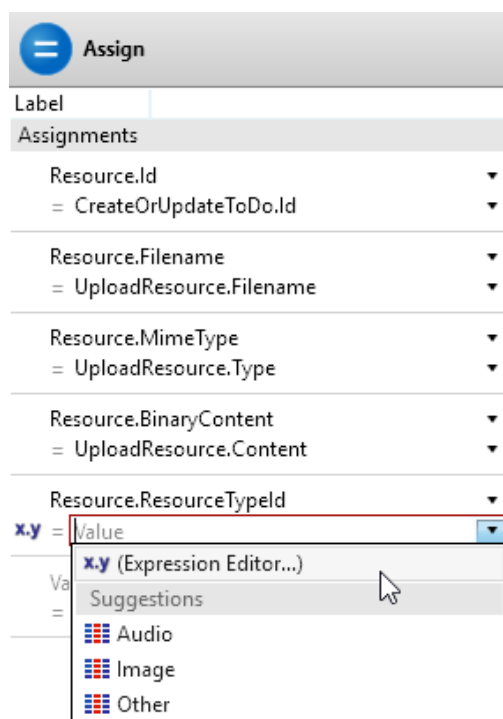


Figure 15. Assign statement assignments

- o) Set the expression to

```
If(Index(UploadResource.Type, "image") <> -1,
Entities.ResourceType.Image, If(Index(UploadResource.Type,
"audio") <> -1, Entities.ResourceType.Audio,
Entities.ResourceType.Other))
```

NOTE: The **UploadResource.Type** variable contains the **MimeType** (e.g. audio/wav, image/jpeg) of the uploaded file. The expression evaluates **UploadResource.Type**, and “converts” the **MimeType** into a Identifier of the **ResourceType** Static Entity.

- p) Select the **CreateOrUpdateResource** Entity Action from the **Save** Screen Action flow, and from the **Source** property drop down select the **Resource** Local Variable.

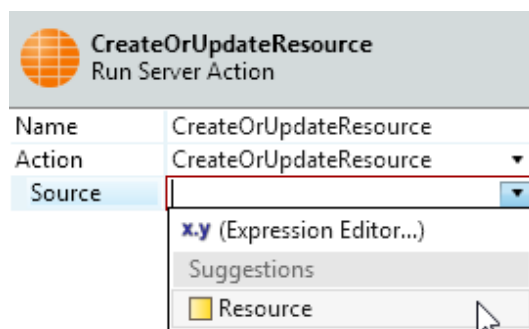


Figure 16. Set CreateOrUpdateResource Source property

- q) Open to the **ToDoDetail** Screen, then select the **Save** Button and change the **Method** property to 'Submit'.

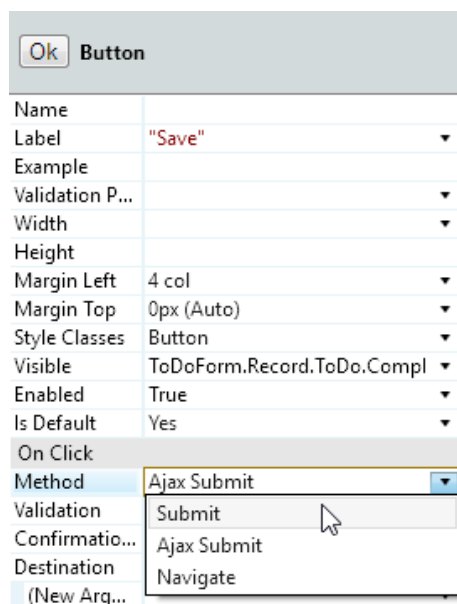


Figure 17. Method property of Save Button

NOTE: When uploading files, the **Submit Method** should be set to 'Submit' instead of 'Ajax Submit'. When the Screen was initially created by Scaffold, the method was automatically set to 'Ajax Submit'.

- r) Double click the **Save** Button to return to the **Save** Action.
- s) To fix the existing warning delete the 'ToDoForm' **Ajax Refresh** statement.

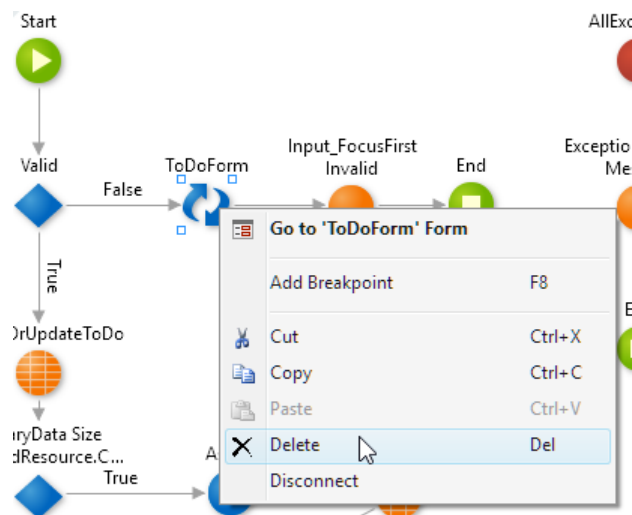


Figure 18. Delete Ajax Refresh statement

NOTE: Recall that the **Submit Method** of the **Save Button** is set to 'Submit', and therefore the **Ajax Refresh** statement will have no effect.

t) The **Save Screen Action flow** should look like this

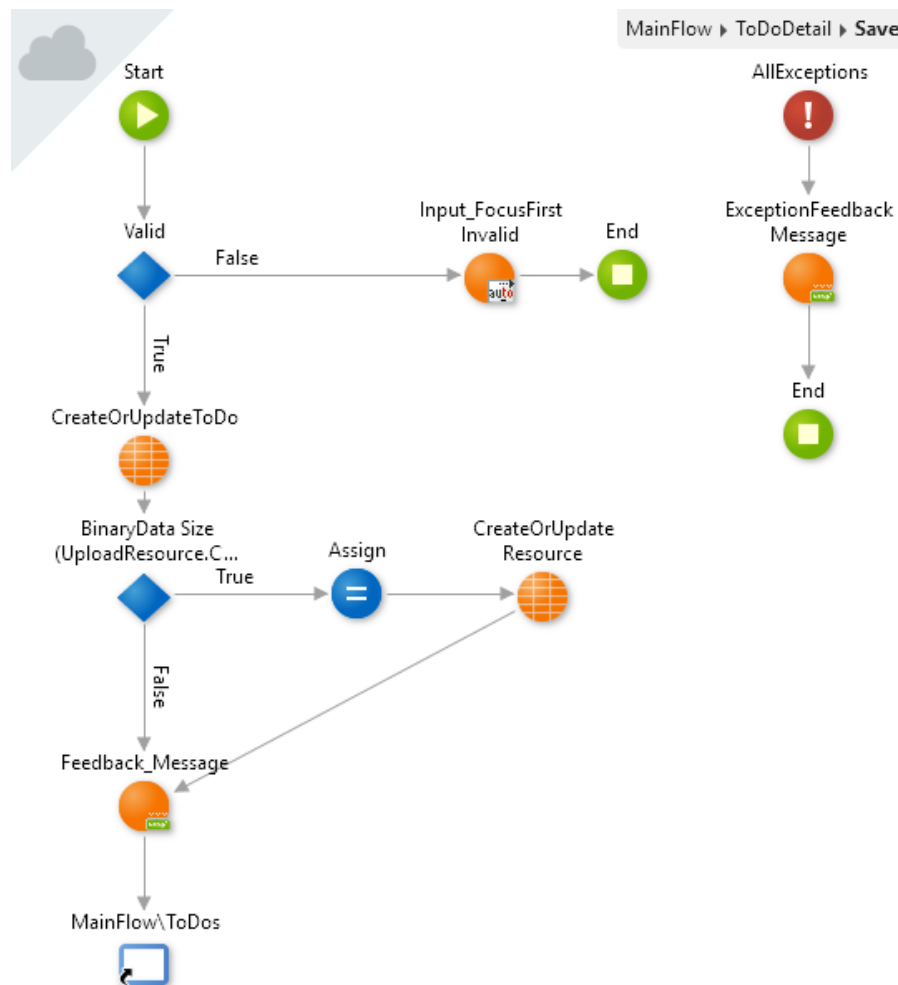
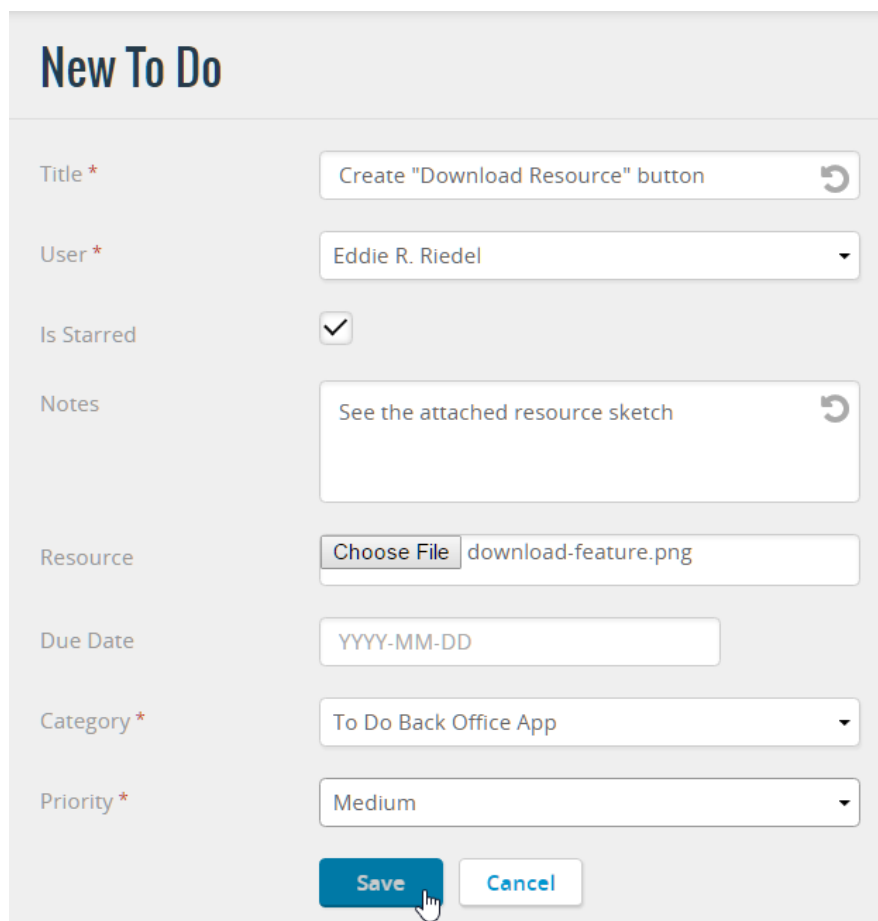


Figure 19. Save Screen Action flow

3. Publish and test the Upload Resource by creating a new To Do.

- a) Click the **1-Click Publish** button to publish the module.
- b) Click the **Open in Browser** button to open the application. If required, login again with the same credentials as before.
- c) Click the **Create New ToDo** Link from the **Menu** on the left side.
- d) Create a new To Do with the information based on the image below. The **download-feature.png** Resource is located in the **Resources** folder.



New To Do

Title * Create "Download Resource" button

User * Eddie R. Riedel

Is Starred ☒

Notes See the attached resource sketch

Resource Choose File download-feature.png

Due Date YYYY-MM-DD

Category * To Do Back Office App

Priority * Medium

Save Cancel

Figure 20. 'Download Resource' Button in action

- e) After clicking **Save**, you should see the green feedback message informing that the To Do was successfully created, and new To Do will appear in the **ToDos** Screen.
- f) Notice that you haven't yet added a way to view or download Resources. This will be done in the next part of this exercise.

Part 3: Add 'Download Resource' feature

In this part of the exercise, you will add a 'Download Resource' Button to the **ToDoDetail** Screen. This Button will only be visible when the To Do has a Resource attached, enabling end-users to download and view the Resource.

1. Fetch the To Do Resource from the database.

a) Open the **Preparation** of the **ToDoDetail** Screen by double clicking it in the elements area of the **Interface** tab.

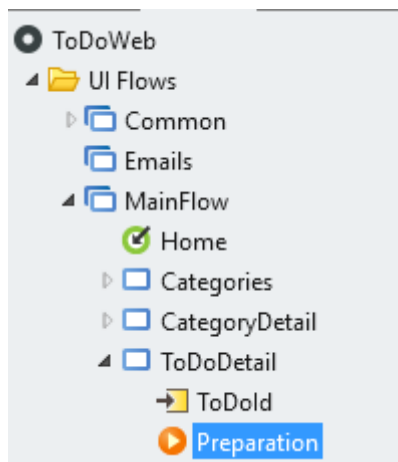


Figure 21. ToDoDetail Preparation

b) Double click the **Aggregate** in the Preparation to open it.

c) Open the **Sources** tab and select **Add Source** to open the **Select Source** dialog.

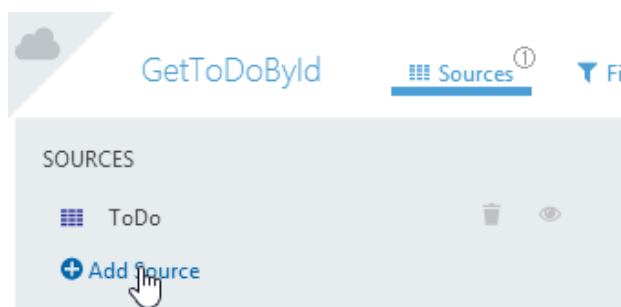


Figure 22. Add Source to GetToDoById Aggregate

d) In the **Select Source** dialog, pick the **Resource** Entity and click **Ok**.

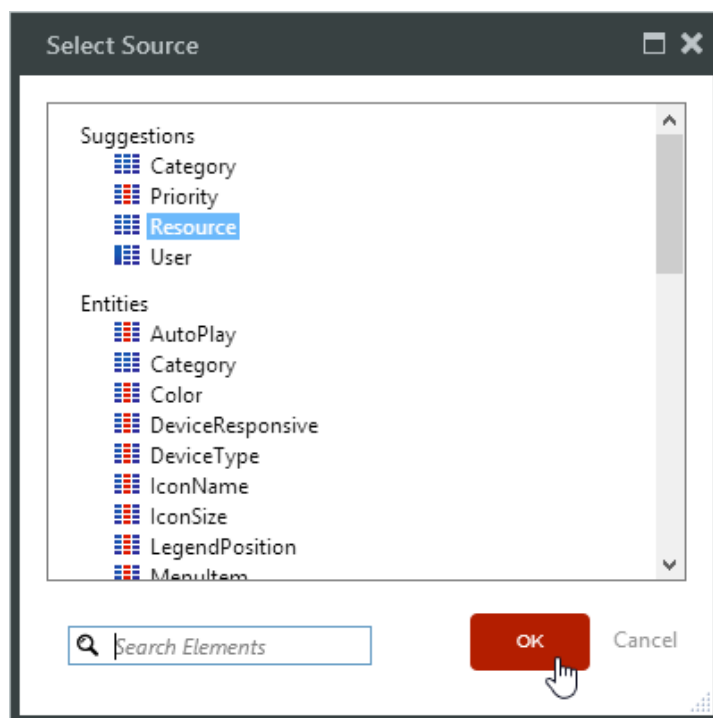


Figure 23. Select Source dialog

- e) Notice that a new Join Condition between the **ToDo** and **Resource** Entities was automatically created.

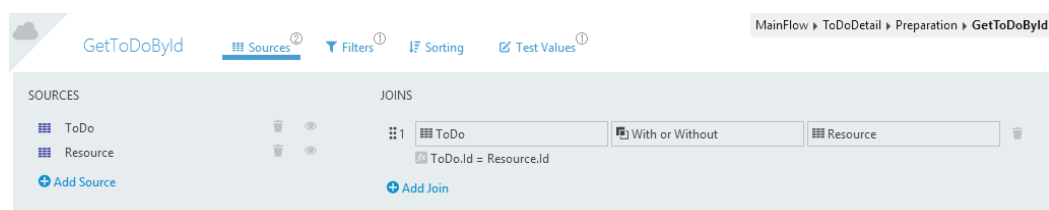


Figure 24. Join Condition between ToDo and Resource Entities

2. Add a 'Download Resource' Button to the **ToDoDetail** Screen.

- a) Use the breadcrumbs to navigate to the context of the **ToDoDetail** Screen.



Figure 25. Navigate to ToDoDetail Screen, using breadcrumbs

- b) Drag a **Container** and drop it on the right side of the **ToDoForm**.

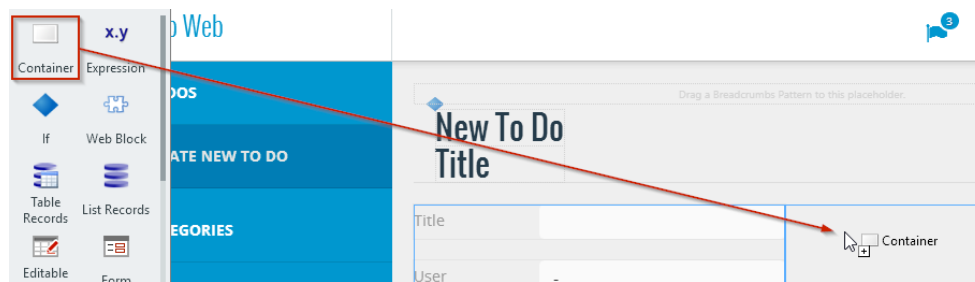


Figure 26. New Container in ToDoDetail Screen

NOTE: Notice that the Container width was set automatically to '6 col' (half of the page width). This happened because you dropped the new Container on the right of a Widget (**ToDoForm**) that also has a width of '6 col'.

- c) Drag a new **Button** Widget and drop it inside the Container created in the previous step.

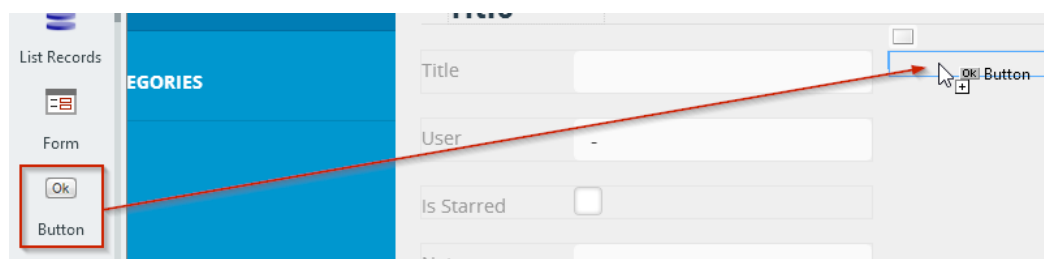


Figure 27. Add Button to Container

- d) With the focus on the Button, click the **Align Center** from the toolbar, to center the Button in the parent Container.



Figure 28. Align Button to center

- e) Set the new Button **Label** property to "Download Resource".
- f) Set the **Visible** property of the Button to
- ```
GetToDoById.List.Current.Resource.Id <> NullIdentifier()
```
- g) Open the **Destination** property drop down and pick (**New Screen Action**) to create a new Screen Action and open it.
- h) Drag a **Refresh Data** statement and drop it between the Start and End.

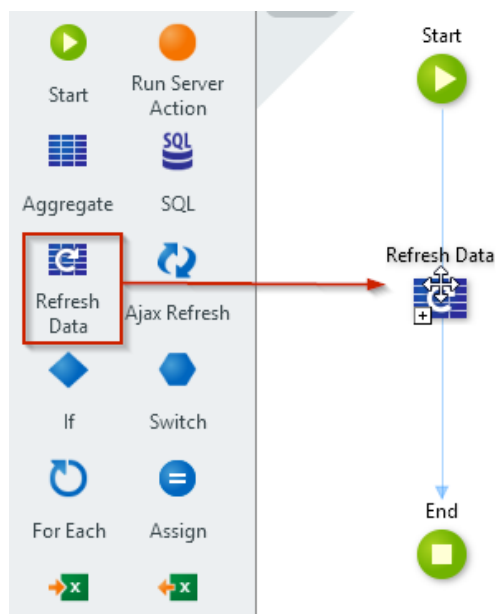


Figure 29. New Aggregate in DownloadResource Screen Action

i) In the **Select Data Source** dialog select the **GetToDoById** Aggregate.

---

**NOTE:** The **Refresh Data** statement will re-execute an existing Aggregate from the **Preparation**, avoiding you to recreate the same Aggregate inside the Screen Action.

---

j) Drag a **Download** node and drop it over the End node to replace it.

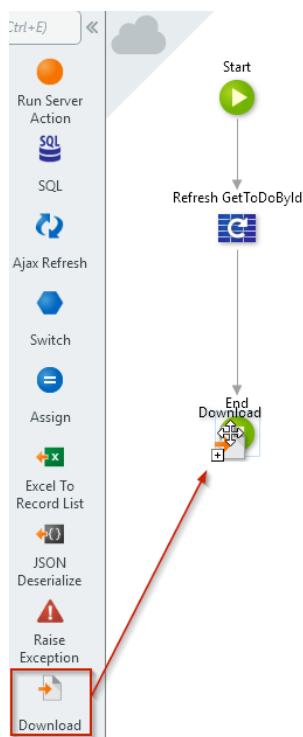


Figure 30. Replace End node by Download

- k) In the properties area of the **Download** statement, open the **File Content** property drop down, and pick

`GetToDoById.List.Current.Resource.BinaryContent`

- l) In the **File Name** property drop down, pick the

`GetToDoById.List.Current.Resource.Filename` option.

- m) For the **Mime-Type** property, pick 'GetToDoById.List.

`Current.Resource.Mimetype`' from the drop down.

- n) The properties of the **Download** statement should look like this

| Download     |                                                   |
|--------------|---------------------------------------------------|
| File Content | GetToDoById.List.Current.Resource.BinaryContent ▼ |
| File Name    | GetToDoById.List.Current.Resource.Filename ▼      |
| Mime-Type    | GetToDoById.List.Current.Resource.MimeType ▼      |
| Save to Disk | Yes ▼                                             |

Figure 31. Download statement properties

### 3. Publish and test the 'Download Resource' Button.

- a) Click the **1-Click Publish** button to publish the module.
- b) Click the **Open in Browser** button to open the application.
- c) From the list of To Dos, open the 'Create "Download Resource" button' To Do.

| TITLE                               | USER          | CATEGORY             | PRIORITY | COMPLETED DATE   |
|-------------------------------------|---------------|----------------------|----------|------------------|
| ★ Create "Download Resource" button | Administrator | ToDo Back Office App | Medium   | Mark as Complete |
| Implement "Complete" feature        | Administrator | ToDo Back Office App | High     | yesterday        |
| ★ Test To Do creation               | Administrator | ToDo Back Office App | High     | yesterday        |

Figure 32. List of To Dos

- d) On the right side of the **ToDoDetail** Screen, you should now see the 'Download Resource' Button. Click it to download the attached Resource.
- e) Verify that the downloaded file is a valid image.

## End of Lab

In this exercise, you have added new features to store **Resources** associated with To Dos.

The new feature allows users to upload a local file from their computer and attach it to a To Do. The **Resource** is stored in a Database Entity, to allow the user to also download the attached **Resource**.

## List of Figures

Here is the list of screenshots and pictures used in this exercise.

|                                                                      |    |
|----------------------------------------------------------------------|----|
| Figure 1. Open Manage Dependencies... dialog .....                   | 4  |
| Figure 2. Add References to Resource and ResourceType Entities ..... | 4  |
| Figure 3. BinaryDataSize reference .....                             | 5  |
| Figure 4. Drag and drop a new Container Widget .....                 | 6  |
| Figure 5. Drag and drop a new Label Widget .....                     | 6  |
| Figure 6. Drag and drop a new Upload Widget .....                    | 7  |
| Figure 7. Label Input Widget property drop down .....                | 7  |
| Figure 8. Select Upload1 as Input Widget .....                       | 7  |
| Figure 9. Upload Widget properties .....                             | 8  |
| Figure 10. Save Screen Action created by Scaffold .....              | 9  |
| Figure 11. Drag and drop an If statement .....                       | 9  |
| Figure 12. Drag and drop an Assign statement .....                   | 10 |
| Figure 13. Drag and drop CreateOrUpdateResource Entity Action .....  | 10 |
| Figure 14. Resource Local Variable properties .....                  | 11 |
| Figure 15. Assign statement assignments .....                        | 12 |
| Figure 16. Set CreateOrUpdateResource Source property .....          | 13 |
| Figure 17. Method property of Save Button .....                      | 13 |
| Figure 18. Delete Ajax Refresh statement .....                       | 14 |
| Figure 19. Save Screen Action flow .....                             | 15 |
| Figure 20. 'Download Resource' Button in action .....                | 16 |
| Figure 21. ToDoDetail Preparation .....                              | 17 |
| Figure 22. Add Source to GetToDoById Aggregate .....                 | 17 |
| Figure 23. Select Source dialog .....                                | 18 |
| Figure 24. Join Condition between ToDo and Resource Entities .....   | 18 |
| Figure 25. Navigate to ToDoDetail Screen, using breadcrumbs .....    | 18 |
| Figure 26. New Container in ToDoDetail Screen .....                  | 19 |
| Figure 27. Add Button to Container .....                             | 19 |
| Figure 28. Align Button to center .....                              | 19 |
| Figure 29. New Aggregate in DownloadResource Screen Action .....     | 20 |
| Figure 30. Replace End node by Download .....                        | 20 |
| Figure 31. Download statement properties .....                       | 21 |
| Figure 32. List of To Dos .....                                      | 21 |