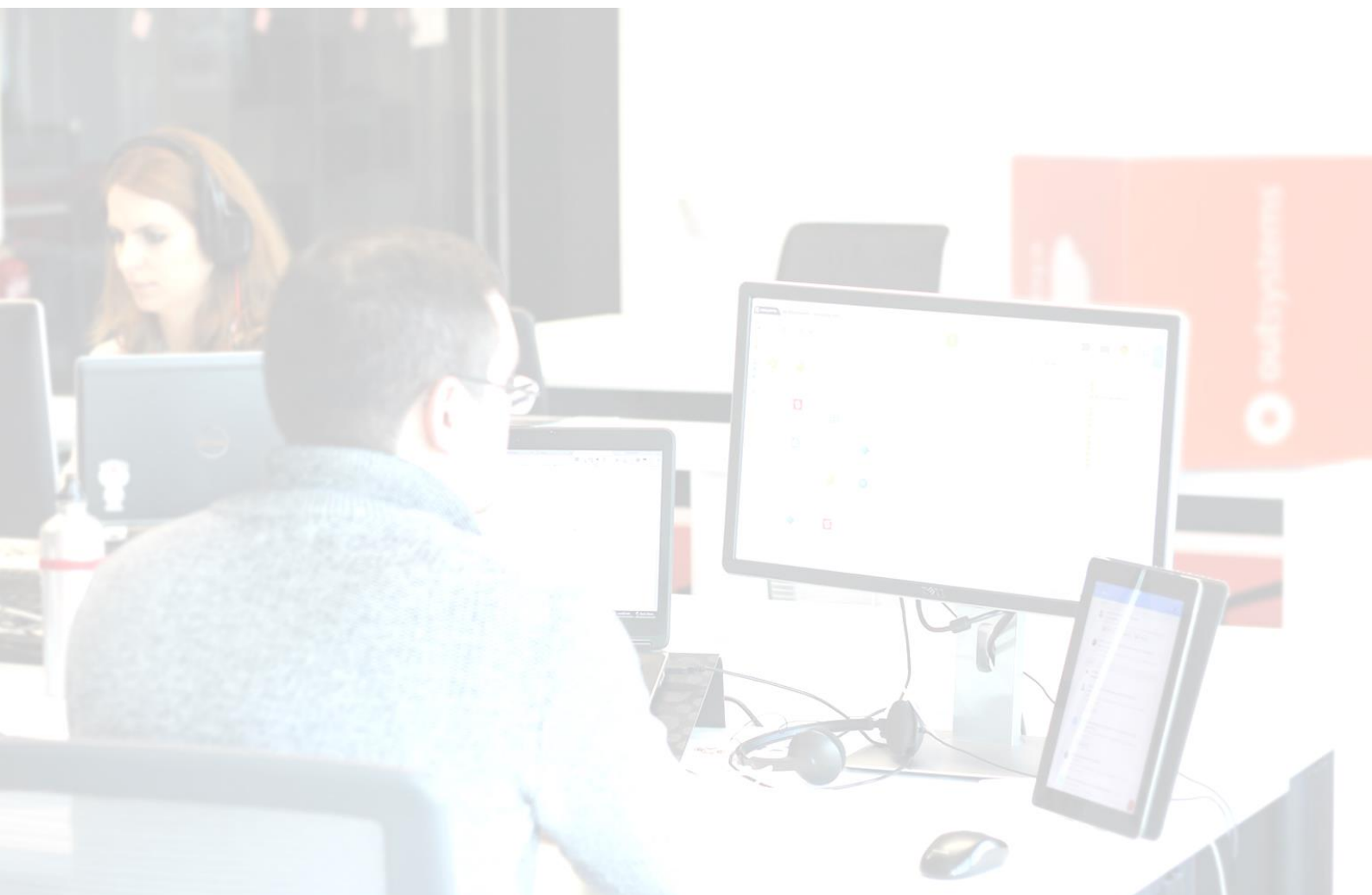




DEVELOPING OUTSYSTEMS MOBILE APPS

Create Data Model



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 To Do 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 and bootstrap Entities
- Create Static Entities, and their associated records

Table of Contents

Introduction.....	2
Table of Contents.....	3
Part 1: Create Entities	4
Part 2: Create Static Entities	11
Part 3: Publish the application module	18
End of Lab	19
List of Figures.....	20

Part 1: Create Entities

In this part of the exercise, you will create the initial Entities (**ToDo** and **Category**) off the application's data model, in the **ToDo_Core** module. An Entity requires a Name, an Id (identifier) and at least one other attribute. Entities can be initialized with data from an Excel spreadsheet. This process is called Bootstrapping.

1. In the **ToDo_Core** module, create the **ToDo** Entity. Add **Title**, **UserId**, **IsStarred**, **Notes**, **DueDate**, **CreatedDate**, **CompletedDate** attributes to the Entity. The following attributes are non-mandatory: **IsStarred**, **Notes**, **DueDate** and **CompletedDate**. **Notes** length should be set to 250. **CreatedDate** default value should be set to the current date.

a) Open the **ToDo_Core** module.

b) Click the **Data** tab in the upper right corner of the workspace to switch the Elements area to the Data Elements.

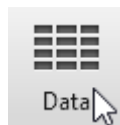


Figure 1. Switch to the Data tab

c) In the elements area, under the **Entities** folder right click **Database** and select **Add Entity**.

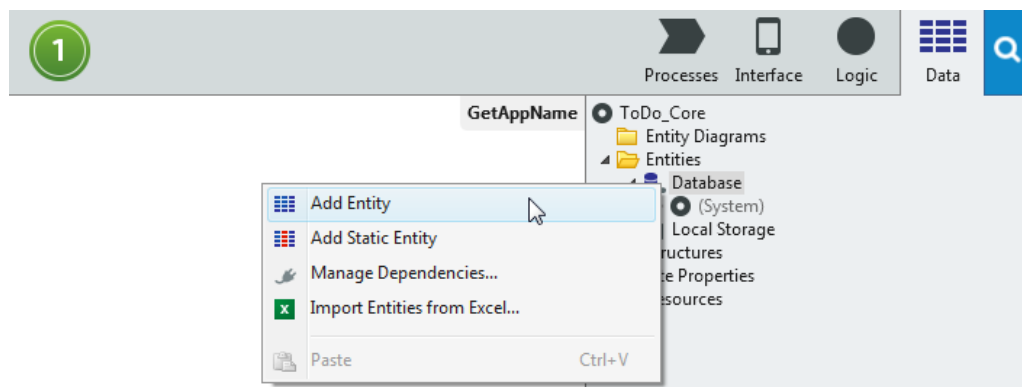


Figure 2. Add Entity

d) Enter 'ToDo' for the name of the Entity.

NOTE: Notice that the **ToDo** Entity is underlined in red and that the TrueChange tab has changed from a green 'check' to a red 'X'. Entities cannot be made up of a single Auto Numbered attribute.

- e) Click the triangle to the left of **ToDo** to expand it.
- f) Notice the six **Entity Actions** that were created to provide typical CRUD functionality.

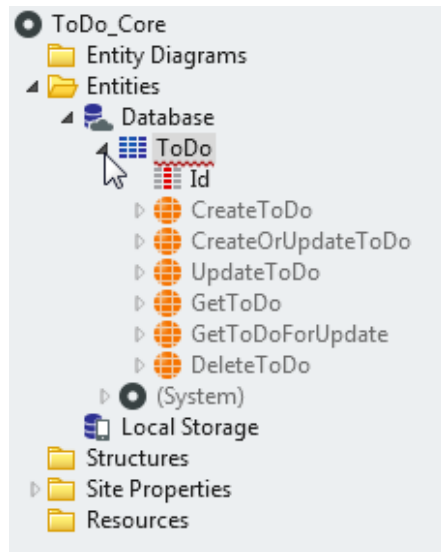


Figure 3. Expanded Entity with its Id attribute and its Entity Actions

- g) Since you will be using this Entity on our UI module, in the properties editor at the bottom right, change the **Public** property to **Yes** and the **Expose Read Only** to **No**.

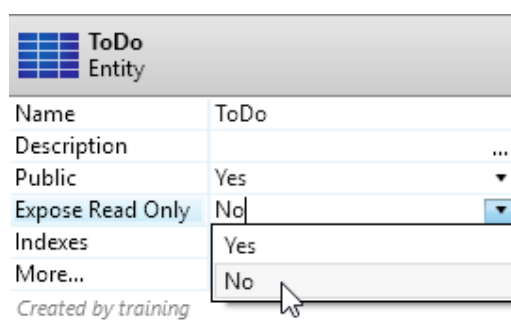


Figure 4. Entity set to public and writable

- h) Double click the **More...** property to open the advanced Entity editor.

Figure 5. ToDo Entity editor

NOTE: The Entity editor allows you to configure more advanced settings relating to your Entity, including behaviors related to its database representation and UI defaults while constructing your Screens.

- i) Click **More Options**, then set the Entity **Label (plural)** to 'ToDos', and then click **Close**.

Figure 6. ToDo Entity "More options"

NOTE: Label properties will be used whenever Service Studio needs to suggest the default name of any module element, or operation that involves multiple instances of **ToDo**. It will save manual adjustments later, if you want to maintain grammar correctness.

- j) Right click the **ToDo** Entity and select 'Add Entity Attribute'.

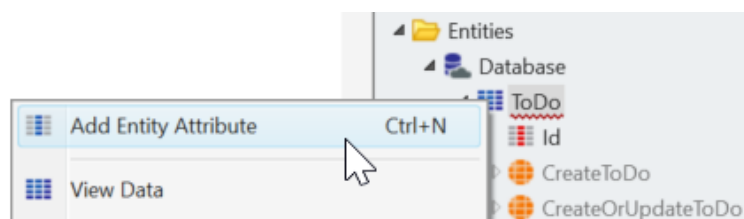


Figure 7. Add an attribute to an Entity

- k) Enter 'Title' for the name of the attribute. Notice that the error indicators disappear.
- l) In the properties editor at the bottom right, change the **Is Mandatory** property to 'Yes'.

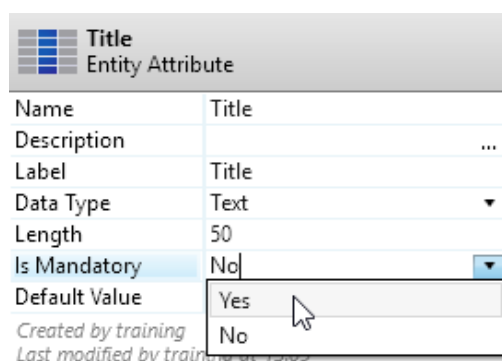



Figure 8. Change the Is Mandatory property to Yes

- m) Notice the default **Data Type** is **Text** with a **Length** of '50' characters.
- n) Right click the **ToDo** Entity and select **Add Entity Attribute**.
- o) Enter 'UserId' for the name of the attribute and make it mandatory. Notice the **Data Type** is automatically set to **User Identifier**.

NOTE: The **User** Entity is built-in with the platform. By having the **UserId** attribute data type set to **User Identifier**, you are creating a database constraint between both Entities. In this case, it will be a many to one relationship between **ToDo** and **User** Entities. Attributes that are foreign keys to other Entities are highlighted with a different icon .

The **Delete Rule** property allows to specify the behavior when you, in this case, delete a User. When set to **Protect**, if a To Do exists for a particular user, it is not possible to delete the User without first deleting all To Dos. When set to **Delete**, deleting a User also deletes all To Dos for that particular user. When set to **Ignore**, deleting a user will leave the To Dos orphaned in the **ToDo** Entity.

- p) Create an 'IsStarred' attribute and verify if the **Data Type** is set to **Boolean**. Leave the attribute as non-mandatory.
- q) Create a 'Notes' attribute, with **Text** as **Data Type**, and set its **Length** to '250' characters. Leave the attribute as non-mandatory.
- r) Create a 'CreatedDate' attribute. Verify the **Data Type** is set to **Date**. Make it mandatory and enter `CurrDate()` for its **Default Value**.

CreatedDate Entity Attribute	
Name	CreatedDate
Description	...
Label	Created Date
Data Type	Date ▼
Is Mandatory	Yes ▼
Default Value	<code>CurrDate()</code>

Figure 9. CreatedDate attribute properties

NOTE: By setting a **Default Value** when a new record is added to the Entity, and the attribute value is undefined, the default value is used.

- s) Create a 'DueDate' attribute. Verify that the **Data Type** is set to **Date**, and leave the attribute as non-mandatory.
 - t) Create a 'CompletedDate' attribute. Verify that the **Data Type** is set to **Date**, and leave the attribute as non-mandatory.
2. Bootstrap the **Category** Entity. Use the **Catergories.xlsx** file in the Resources.

- a) Right click the **Database** element under **Entities**, and then select **Import Entities from Excel...**

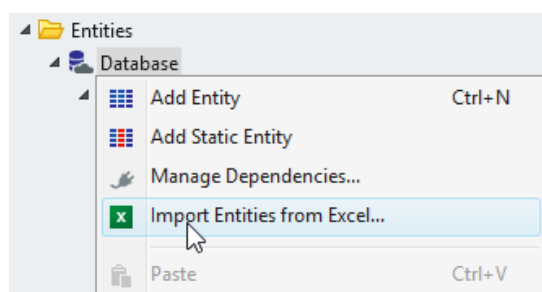


Figure 10. Import an Entity from Excel

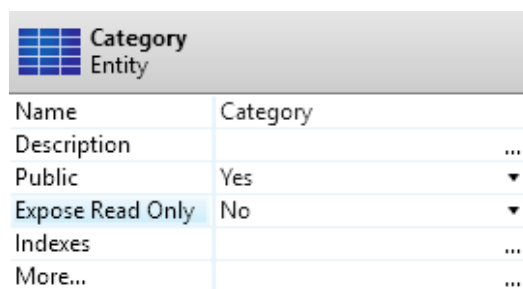
- b) Browse to the **Resources** folder in the Boot Camp materials, select **Categories.xlsx** and click **Open**.

NOTE: A timer named **BootstrapCategories**, which is scheduled to run when published, will be created. The **BootstrapCategories** timer calls the **BootstrapCategories** Action. Timers can be found in the **Processes** tab, so stars will appear on the Processes tab.

The **BootstrapCategories** Action will also be created. It checks if any Categories currently exist. If not, it imports the Categories from the Excel spreadsheet and creates a Category for each row in the spreadsheet. Actions can be found in the Logic tab, so stars will also appear on the Logic tab as well.

The Excel file will be saved in the Resources folder, in the **Data** tab.

- c) Select the **Category** Entity, and set the **Public** property to 'Yes', and **Expose Read Only** to 'No'.



Category Entity	
Name	Category
Description	...
Public	Yes ▼
Expose Read Only	No ▼
Indexes	...
More...	...

Figure 11. Category Entity set as public and writable

- d) The **Category** Entity was automatically created as part of the bootstrapping action. Make sure that the **Name** attribute has Data Type **Text**, and a Length of '50' characters.
3. Add a new attribute to the **ToDo** Entity of type **Category Identifier**.
- a) Right-click the **ToDo** Entity and select **Add Entity Attribute**.
- b) Enter 'CategoryId' for the name of the attribute. Make sure the **Data Type** is **Category Identifier**, and set it as mandatory.

The screenshot shows the 'ToDo' entity in the OutSystems Data Model Designer. The entity attributes are listed on the left, and the 'CategoryId' attribute is highlighted. Below the entity list, the 'CategoryId' attribute is detailed in a table.

Name	CategoryId
Description	
Label	Category
Data Type	Category Identifier
Is Mandatory	Yes
Delete Rule	Protect

Figure 12. CategoryId foreign key in the ToDo Entity

Part 2: Create Static Entities

In this part of the exercise, you will create a Static Entity (**Priority**). A Static Entity is created with a few initial attributes (**Label**, **Order** and **Is_Active**) that support its operation as an enumerate, but other attributes can be added. Unlike normal Entities, the Static Entity data is defined and initialized at design time. Each of the possible values of a Static Entity is called a Record.

1. Create the **Priority** Static Entity. Add three records to the Entity: **Low**, **Medium** and **High**.

- a) Click the **Data** tab in the upper right corner of the workspace to switch to the Data Elements.

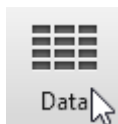


Figure 13. Switch to the Data elements

- b) Under the **Entities** folder, right click **Database** and select **Add Static Entity**.

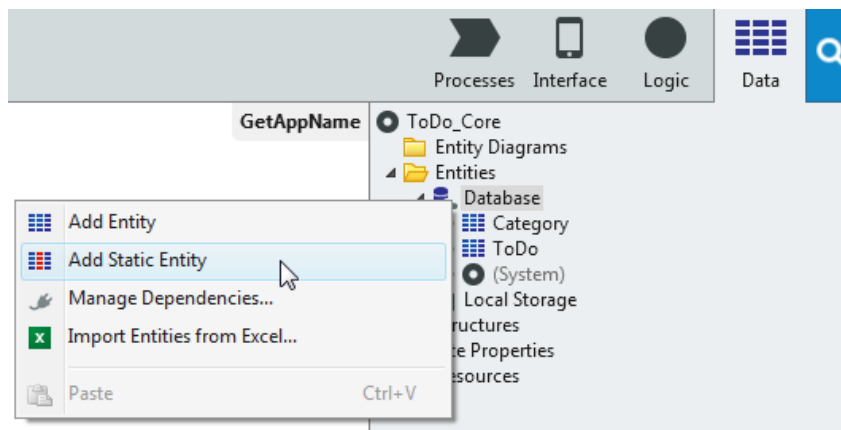


Figure 14. Add a Static Entity

- c) Enter 'Priority' for the name of the Static Entity.
- d) Click the triangle to the left of **Priority** to expand it.
- e) Notice the Static Entity only has one **Entity Action**, as it is impossible to dynamically create or update records in runtime.

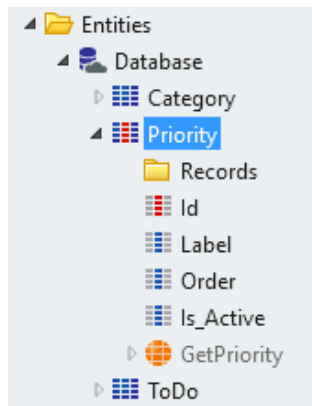


Figure 15. Static Entities have one Entity Action

- f) Since you will be using this Static Entity on your UI module, in the properties editor, change the **Public** property to 'Yes'.

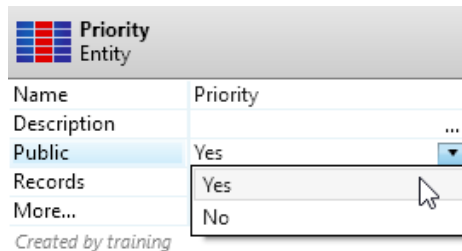


Figure 16. Static Entity set to public

- g) Right click the **Records** folder and select **Add Record**.

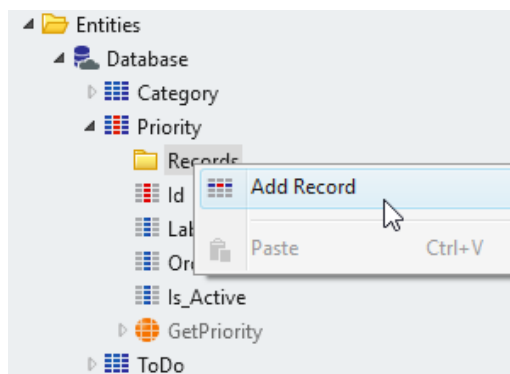


Figure 17. Add Record to a Static Entity

- h) Enter 'Low' for the name of the record.
- i) Repeating the 2 previous steps, add two more records: 'Medium' and 'High'. The Static Entity should look like this.

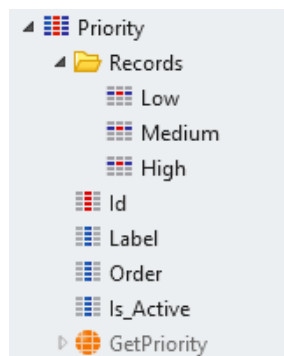


Figure 18. The Priority Static Entity

2. Add a new attribute to the **ToDo** Entity of type **Priority Identifier**.

- a) Right-click the **ToDo** Entity and select **Add New Attribute**.
- b) Enter 'PriorityId' for the name of the attribute. Make sure the **Data Type** is **Priority Identifier**, and set it as mandatory.

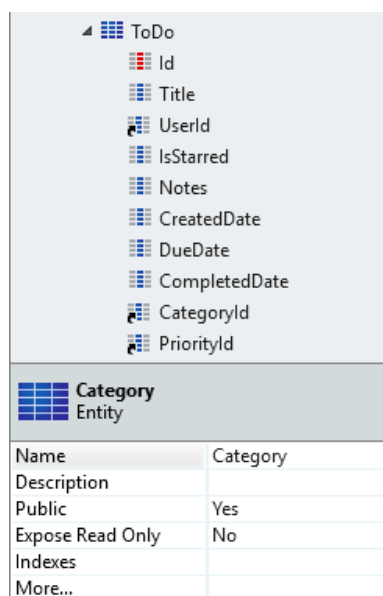


Figure 19. PriorityId foreign key in ToDo Entity

3. Create a new Static Entity named **ResourceType** to specify the different types of resources: **Audio**, **Image** and **Other**.

- a) Right-click the **Database** element and select **Add Static Entity**.

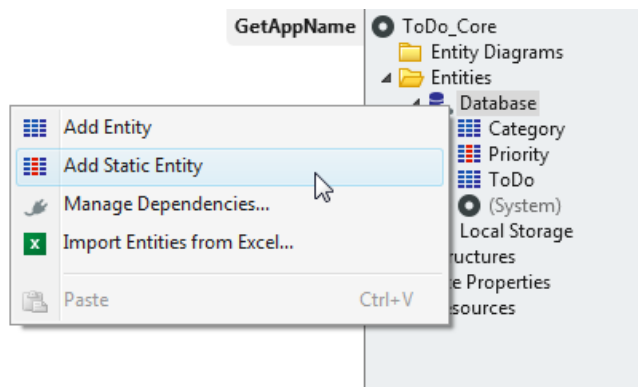


Figure 20. Add Static Entity

- b) Set the name of the Static Entity to 'ResourceType'.
- c) Set its **Public** property to 'Yes'.

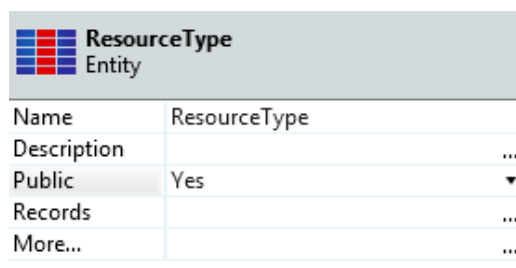


Figure 21. ResourceType Static Entity properties

- d) Expand the **ResourceType** Static Entity by clicking the triangle icon.
 - e) Right click the **Records** folder and select **Add Record**.
 - f) Enter 'Audio' for the record identifier.
 - g) Add 2 more records to ResourceType: 'Image' and 'Other'.
4. Create the **Resource** extension Entity to store resources' information.
- a) Right click the **Database** element and select **Add Entity**.

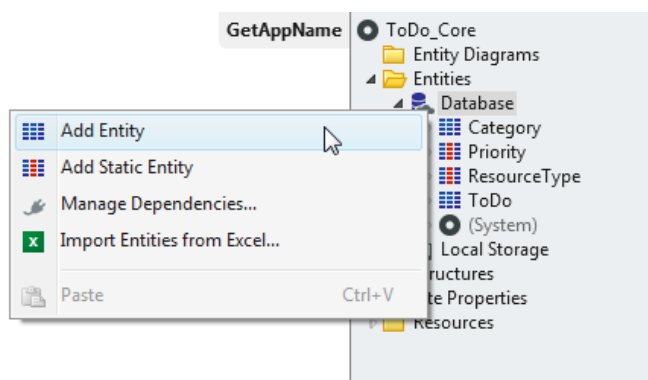


Figure 22. Add Entity

- b) Set the Name of the Entity to 'Resource'.
- c) Change the **Public** property to 'Yes' and the **Expose Read Only** to 'No'.
- d) Expand the **Resource** Entity to view existing attributes and the six Entity Actions.

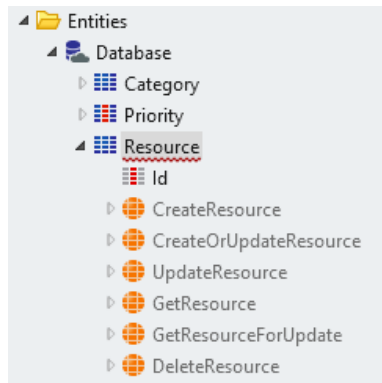


Figure 23. Resource Entity attributes and Entity Actions

- e) Select the **Id** attribute and set its **Data Type** to **ToDo Identifier**.

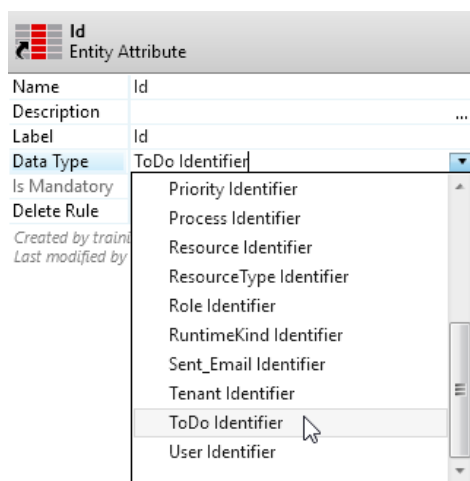


Figure 24. Change Resource Id Attribute Data Type

NOTE: Setting an Entity Identifier **Data Type** to another Entity Identifier creates a one-to-one Entity relationship. This also changes the **Auto Number** property of the attribute to 'No'.

- f) Right click the **Resource** Entity and select **Add Entity Attribute**.

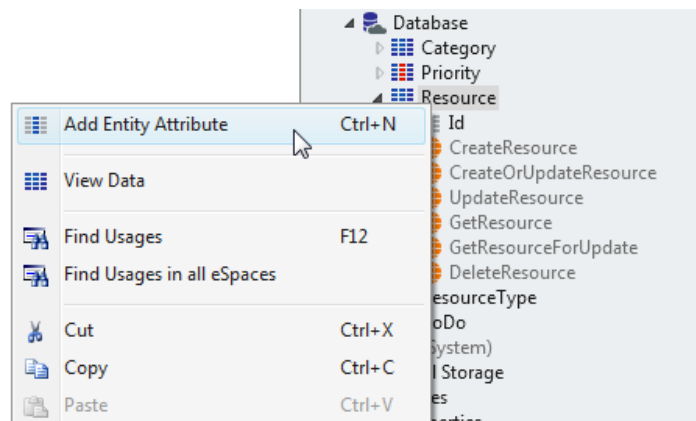


Figure 25. Add Entity Attribute to Resource Entity

- g) Set the attribute Name to 'ResourceTypeId'.
- h) In the properties area of the **ResourceTypeId** attribute, set the **IsMandatory** property to 'Yes' and verify that the **Data Type** was automatically set to **ResourceType Identifier**.

ResourceTypeId Entity Attribute	
Name	ResourceTypeId
Description	...
Label	Resource Type
Data Type	ResourceType Identifier ▼
Is Mandatory	Yes ▼
Delete Rule	Protect ▼

Figure 26. ResourceTypeId attribute properties

- i) Add a new attribute to the **Resource** Entity and name it 'Filename'.
- j) Verify that the attribute **Data Type** is set to **Text**, and make it mandatory.

Filename Entity Attribute	
Name	Filename
Description	...
Label	Filename
Data Type	Text ▼
Length	50
Is Mandatory	Yes ▼
Default Value	

Figure 27. Filename attribute properties

- k) Add a new attribute to the **Resource** Entity and name it 'MimeType'.
- l) Verify that the attribute **Data Type** is set to **Text**, and make it mandatory.

MimeType Entity Attribute	
Name	MimeType
Description	...
Label	Mime Type
Data Type	Text ▼
Length	50
Is Mandatory	Yes ▼
Default Value	

Figure 28. MimeType attribute properties

- m) Add a new attribute to the **Resource** Entity and name it 'BinaryContent'.
- n) Verify that the attribute **Data Type** is set to **Binary Data**, and make it mandatory.

BinaryContent Entity Attribute	
Name	BinaryContent
Description	...
Label	Binary Content
Data Type	Binary Data ▼
Is Mandatory	Yes ▼

Figure 29. BinaryContent attribute properties

NOTE: **Binary Data** type allows to store binary content (e.g. image) in an attribute of a database record. In this application, this attribute will be used to store the resource binary data content, and also will allow the end-users to download the contents of resources.

Part 3: Publish the application module

In this part of the exercise, you will publish this application module to the server. The publishing process uploads the module's information to the server and then proceeds to generate and compile the necessary code and create the required database scripts. It uses the scripts to update the database server and then deploys the application to the server.

Only upon publication will the Entities and Static Entities you defined be committed as tables in the database.

1. Publish the **ToDo_Core** module to the server.

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

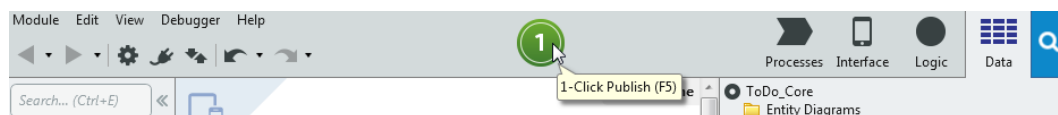


Figure 30. Publish a module with 1-Click Publish

b) Notice the **1-Click Publish** tab that appears near the bottom. This tab provides progress updates on the publishing process.

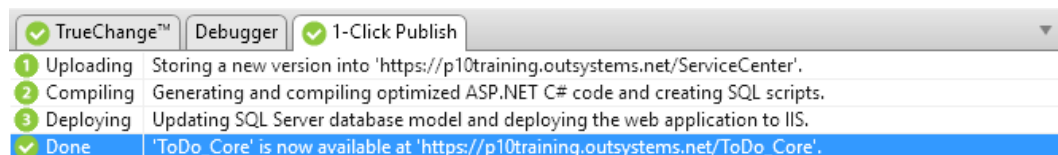


Figure 31. 1-Click Publish progress information

NOTE: Once published, module Screens become available at the URL displayed in Done. In general, that URL will be <http://hostname/ModuleName>. Since the current module has no UI, you won't be able to navigate to it using your browser.

End of Lab

In this exercise, you created an initial data model for the **ToDo** application. The main application concepts are **ToDo**s and **Categories** that were represented as Entities, which were later stored in a database.

In preparation for more advanced modelling of these Entities and their relationship, you created a Static Entity: **Priority**.

The application module was published to the server, thus creating the appropriate database tables for these Entities.

We will start visualizing these Entities in the following lab.

List of Figures

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

Figure 1. Switch to the Data tab	4
Figure 2. Add Entity	4
Figure 3. Expanded Entity with its Id attribute and its Entity Actions	5
Figure 4. Entity set to public and writable	5
Figure 5. ToDo Entity editor	6
Figure 6. ToDo Entity "More options"	6
Figure 7. Add an attribute to an Entity	7
Figure 8. Change the Is Mandatory property to Yes	7
Figure 9. CreatedDate attribute properties	8
Figure 10. Import an Entity from Excel	8
Figure 11. Category Entity set as public and writable	9
Figure 12. CategoryId foreign key in the ToDo Entity	10
Figure 13. Switch to the Data elements	11
Figure 14. Add a Static Entity	11
Figure 15. Static Entities have one Entity Action	12
Figure 16. Static Entity set to public	12
Figure 17. Add Record to a Static Entity	12
Figure 18. The Priority Static Entity	13
Figure 19. PriorityId foreign key in ToDo Entity	13
Figure 20. Add Static Entity	14
Figure 21. ResourceType Static Entity properties	14
Figure 22. Add Entity	14
Figure 23. Resource Entity attributes and Entity Actions	15
Figure 24. Change Resource Id Attribute Data Type	15
Figure 25. Add Entity Attribute to Resource Entity	16
Figure 26. ResourceTypeId attribute properties	16
Figure 27. Filename attribute properties	16
Figure 28. MimeType attribute properties	17
Figure 29. BinaryContent attribute properties	17
Figure 30. Publish a module with 1-Click Publish	18
Figure 31. 1-Click Publish progress information	18