Creating Database Storage Exercise

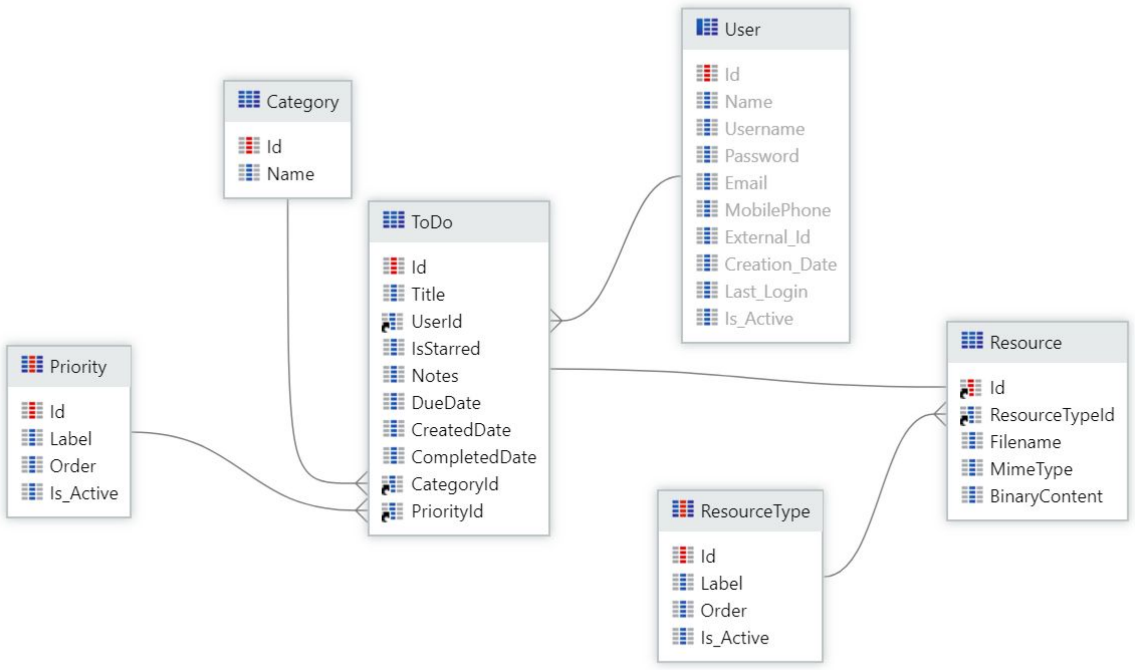


Table of Contents

Introduction

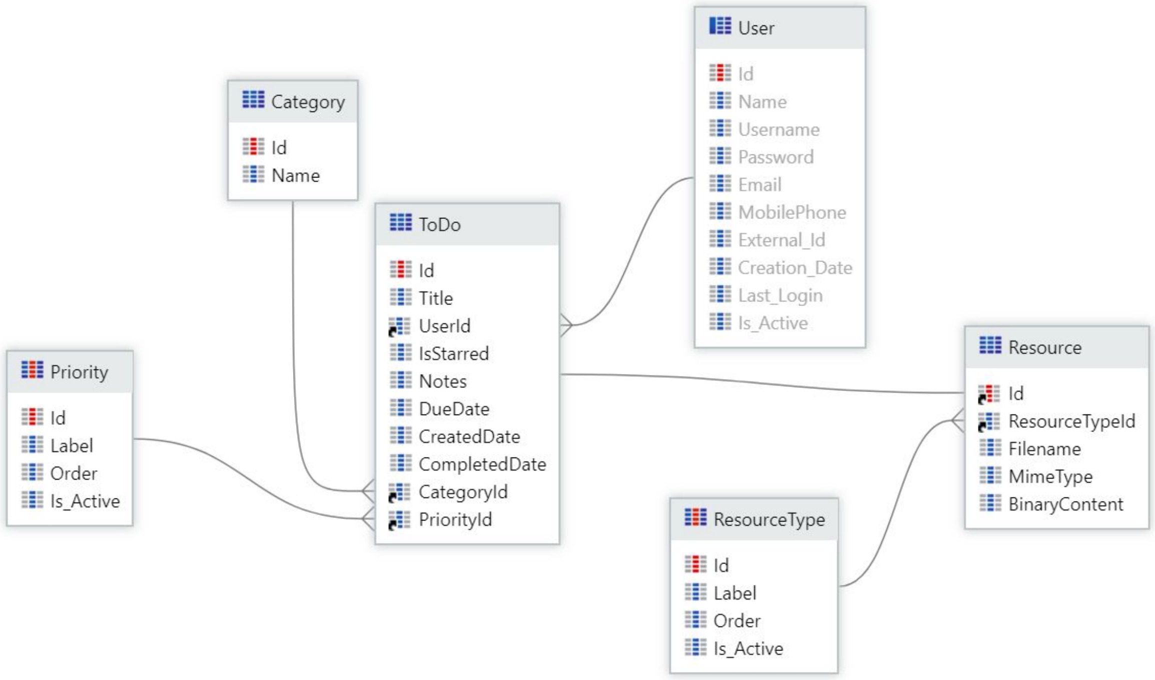
In this exercise lab, we already have the ToDo application created, so we will start to progressively build our app, by creating the data model.

The data model of this application will exclusively be created on the ToDo_Core module, and will consist at this stage on three Entities, ToDo, Category and Resource, and on two Static Entities, Priority and ResourceType.

These Entities will represent the tasks to do (ToDo) in the database, with their Categories and Priorities. Also, to associate a User to each task, we will use the System Entity User, already available in Service Studio to be used.

Finally, we will create a Resource Entity to represent an attachment (audio, image or other) to the ToDos. The type of Resources will be defined in the ResourceType Static Entity.

At the end of the Lab, we should have our data model looking like this



Create the Database Entities

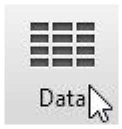
In this exercise lab, we will create the initial Entities (**ToDo** and **Category**) off the application's data model, in the **ToDo_Core** module. An Entity in OutSystems requires a **Name**, an **Id** (identifier) attribute and at least one other attribute. Entities can be initialized with data from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

Since our application will have two modules, and the UI module will reference these Entities, they will be defined as **Public**.

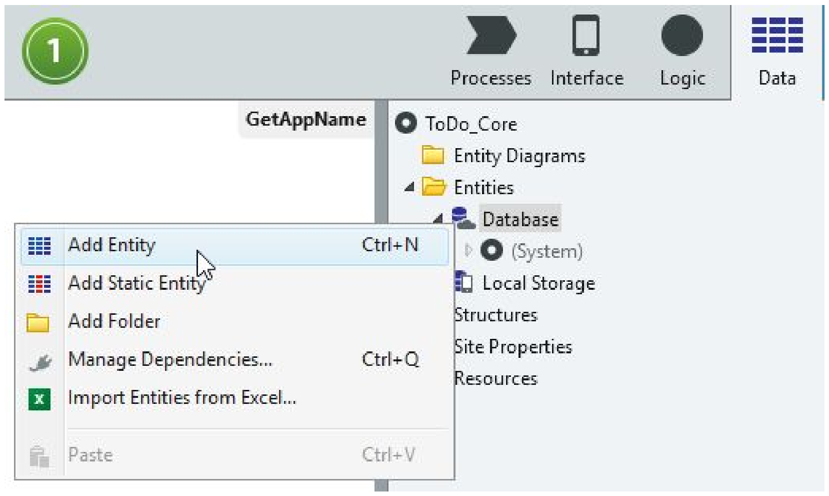
1) Create the ToDo Entity in the ToDo_Core module. Make sure that the Entity is set to **Public** and the **Expose Read Only** property to *No.* Also, change the ToDo **plural Label** to *ToDos*.

    a) Open the **ToDo_Core** module.

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



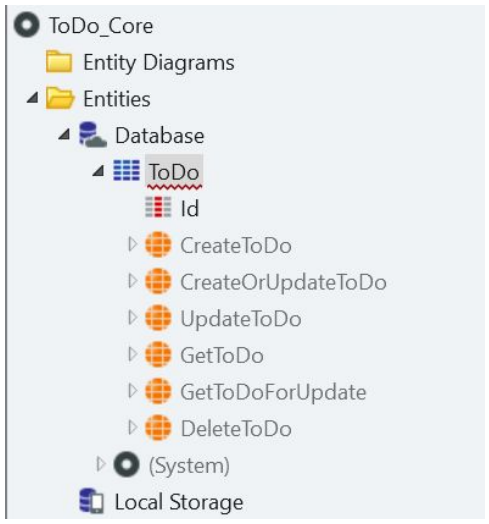
    c) Under the Entities folder, right click the section Database and select **Add Entity**.



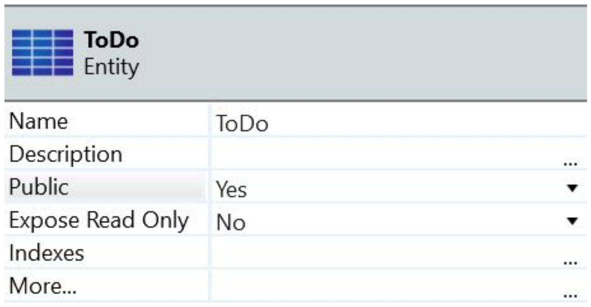
    d) Enter *ToDo* for the name of the Entity.

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'. This was caused by the creation of the Entity, which cannot be made up of a single Auto Numbered attribute (Id). This error will be fixed later.

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

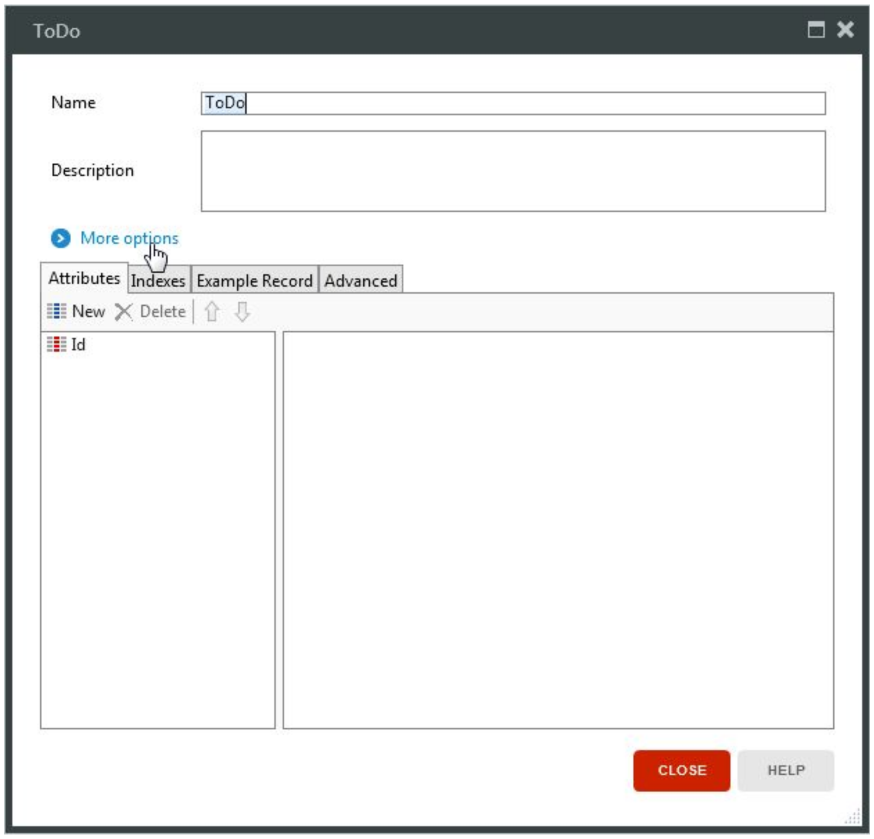


f) Since we will be using this Entity on our UI module, in the properties editor at the bottom right, change the **Public** property to *Yes* and the **Expose Read Only** to *No*.


ToDo
Entity

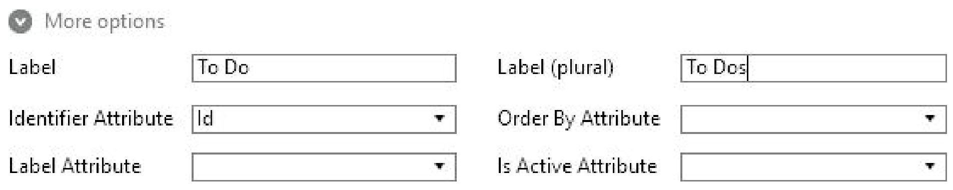
Name	ToDo
Description	...
Public	Yes
Expose Read Only	No
Indexes	...
More...	...

g) Double click the **More...** property to open the advanced 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.

h) Click **More Options**, then set the Entity **Label (plural)** to *ToDos***,** and then click **Close**. This will change the plural Label of the Entity to *To Dos*, which by default was created as *To Does*.

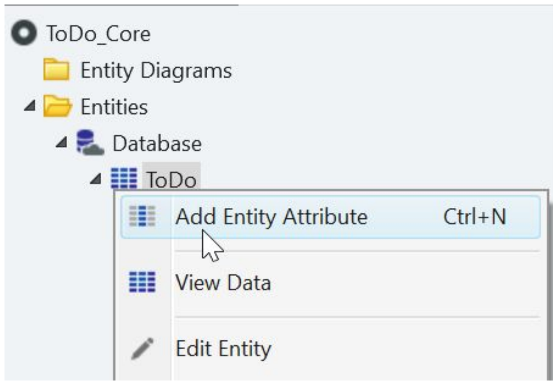


Label	To Do	Label (plural)	To Dos
Identifier Attribute	Id	Order By Attribute	
Label Attribute		Is Active Attribute	

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.

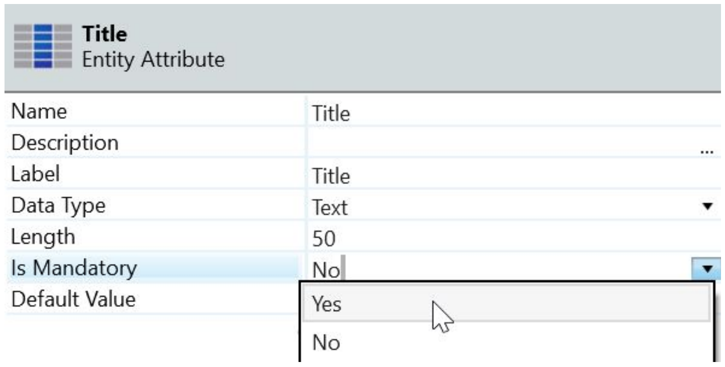
2) The ToDo Entity should have the following attributes: *Title* (Text), *UserId* (User Identifier), *IsStarred* (Boolean), *Notes* (Text), *DueDate* (Date), *CreatedDate* (Date) and *CompletedDate* (Date). The **Title**, **UserId** and **CreatedDate** are mandatory fields. The **Notes** attribute needs to have *250* characters of length and the default value of the **CreatedDate** should be set to the current date.

   a) Right click the **ToDo** Entity and select **Add Entity Attribute**.



   b) Enter *Title* for the name of the attribute. Notice that the error indicators disappear.

   c) In the properties editor at the bottom right, change the **Is Mandatory** property to *Yes*. Notice the default **Data Type** is *Text* with a **Length** of *50* characters.



Title
Entity Attribute

Name	Title
Description	...
Label	Title
Data Type	Text
Length	50
Is Mandatory	No
Default Value	

   d) Right click the **ToDo** Entity and select **Add Entity Attribute**.

   e) 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*, we 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 .

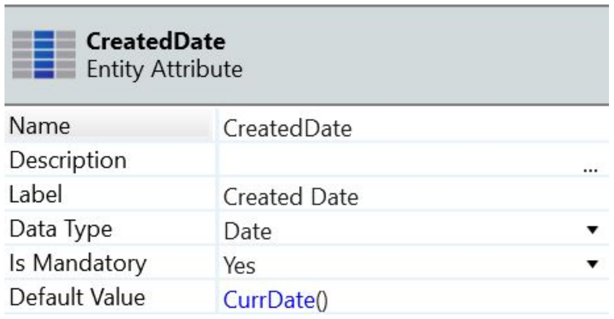
The **Delete Rule** property allows to specify the behavior when we, in this case, delete a User. When set to *Protect*, if a To Do exists for a particular user, it is not possible to delete the User without first deleting all its To Dos. When set to *Delete*, deleting a User also deletes all To Dos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

f) Create an *IsStarred* attribute and verify if the **Data Type** is set to *Boolean*. Leave the attribute as **non-mandatory**.

g) Create a *Notes* attribute, with **Data Type** set to *Text*, and set its **Length** to *250* characters. Leave the attribute as **non-mandatory**.

h) Create a *DueDate* attribute. Verify that the **Data Type** is set to *Date*, and leave the attribute as **non-mandatory**.

i) Create a *CreatedDate* attribute. Verify the **Data Type** is set to *Date*. Make it **mandatory** and enter *CurrDate()* as its **Default Value**.

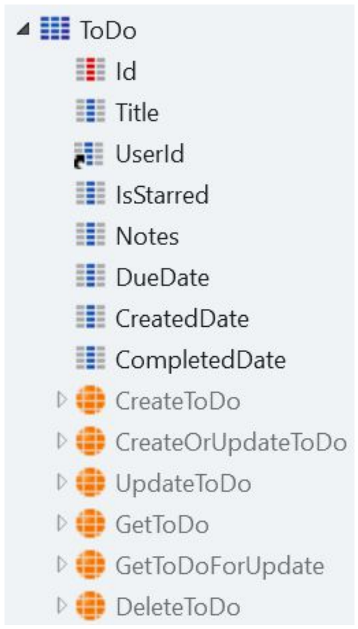

CreatedDate
Entity Attribute

Name	CreatedDate
Description	...
Label	Created Date
Data Type	Date
Is Mandatory	Yes
Default Value	CurrDate()

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.

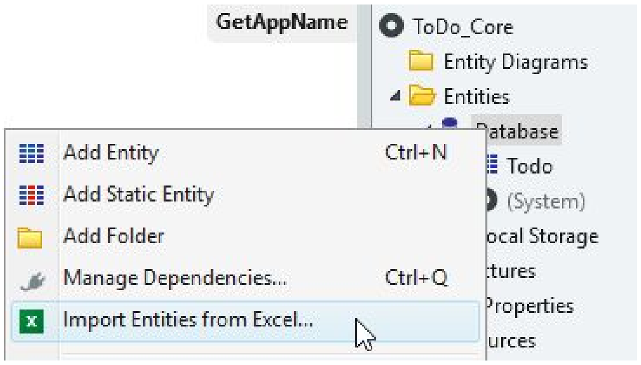
j) Create a *CompletedDate* attribute. Verify that the **Data Type** is set to *Date*, and leave the attribute as **non-mandatory**.

k) The ToDo Entity should look like the following screenshot



3) Bootstrap the **Category** Entity from the **Catergories.xlsx** Excel file. This file can be found in the Resources folder of the Exercise. Set it to **Public** and **expose it with write permissions**.

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

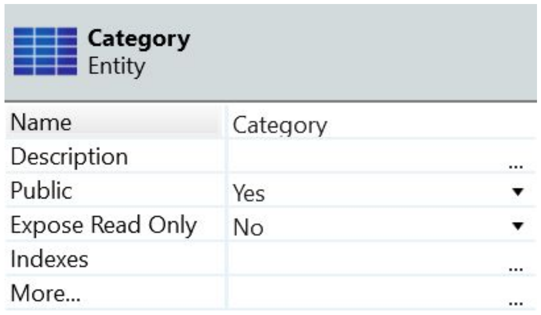


   b) Browse to the Resources folder in the Boot Camp materials, select **Categories.xlsx** and click **Open**.

NOTE: Stars will appear in the interface when the previous step is completed. The stars indicate the areas where elements are being created.

The logic for fetching the data from the Excel file and add it to the database is created in the Action **BootstrapCategories,** under the Logic tab. It checks if any Category currently exist. If not, it imports the Categories from the Excel spreadsheet and creates a Category in the database, for each row in the spreadsheet. The Excel file will be saved inside the module, in the Resources folder under the Data tab. This Action runs when the module is published.

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



Category
Entity

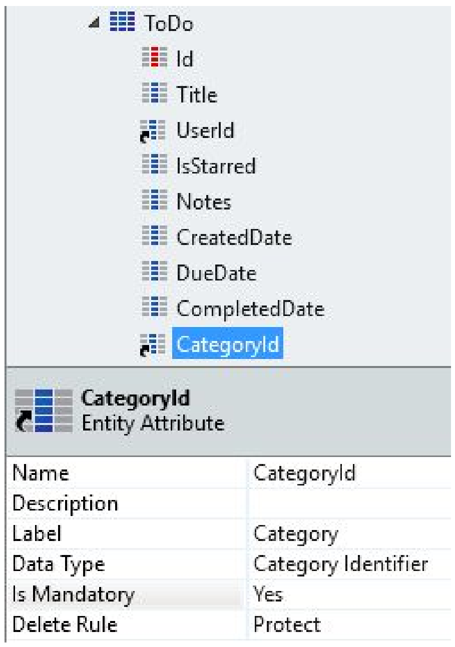
Name	Category
Description	...
Public	Yes ▾
Expose Read Only	No ▾
Indexes	...
More...	...

The **Category** Entity was automatically created as part of the bootstrapping action. Make sure that the **Name** attribute has **Data Type** set to *Text*, and a Length of *50* characters.

4) Add a new attribute to the **ToDo** Entity of type *Category Identifier*, to create a 1-to-many relationship between the **ToDo** and the **Category** Entities.

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

e) Enter *CategoryId* for the name of the attribute. Make sure the **Data Type** is *Category Identifier*.



- ToDo
  - Id
  - Title
  - UserId
  - IsStarred
  - Notes
  - CreatedDate
  - DueDate
  - CompletedDate
  - CategoryId

CategoryId
Entity Attribute

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

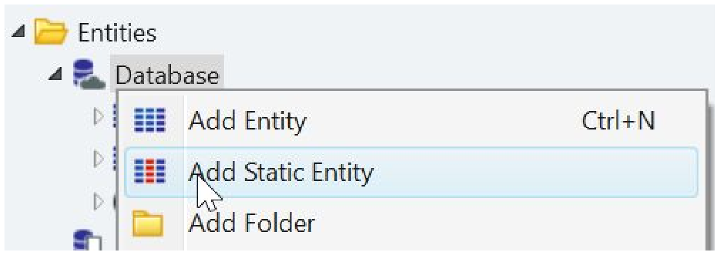
Create the Priority Static Entity

After creating the first Entities, we will create one Static Entity: **Priority**. This Entity will represent the Priorities of the ToDos in the database. A Static Entity is initially created with a few attributes: **Label**, **Order** and **Is_Active**, but other attributes can be added.

Unlike regular 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.

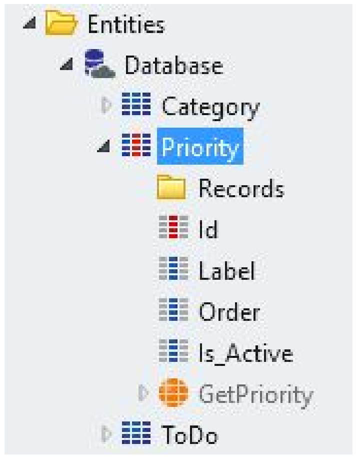
1) Create the *Priority* Static Entity and set it to **Public**. Add three records to the Entity: *Low*, *Medium* and *High*.

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

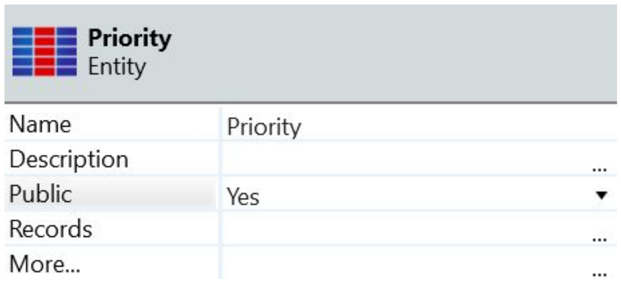


   b) Enter *Priority* for the name of the Static Entity.

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

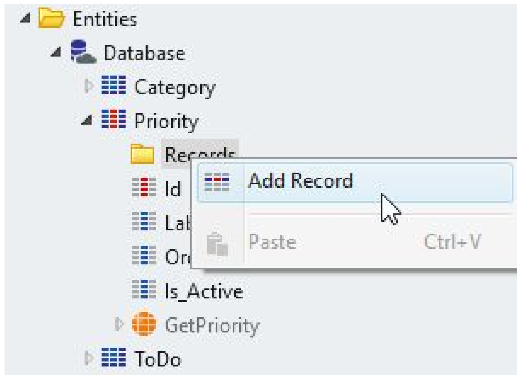


   d) Since we will be using this Static Entity on your UI module, in the properties editor, change the **Public** property to *Yes*.



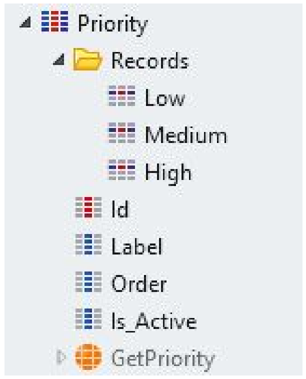
Priority Entity	
Name	Priority
Description	...
Public	Yes
Records	...
More...	...

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



f) Enter *Low* for the name of the record.

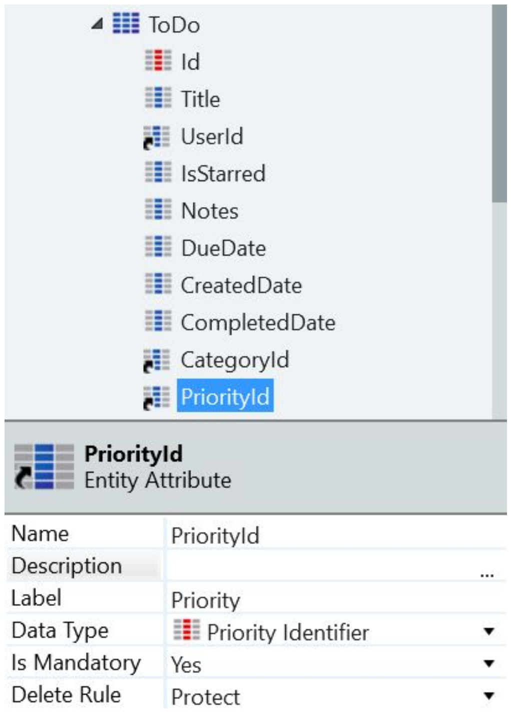
g) Repeating the 2 previous steps, add two more records: *Medium* and *High*. The Static Entity should look like this



2) Add a new attribute to the **ToDo** Entity of type *Priority Identifier*, to create the 1-to-many relationship between the **ToDo** and the **Priority** Entities.

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

   b) Enter *PriorityId* for the name of the attribute. Make sure the **Data Type** is set to *Priority Identifier*, and set it as **mandatory**.



PriorityId
Entity Attribute

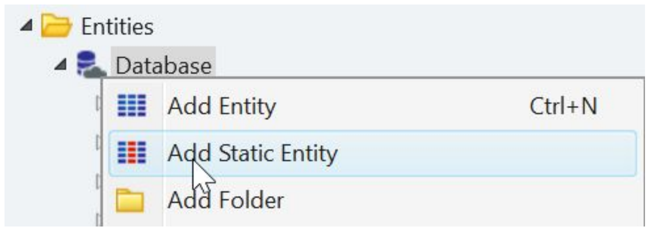
Name	PriorityId
Description	...
Label	Priority
Data Type	Priority Identifier
Is Mandatory	Yes
Delete Rule	Protect

Create Entities for representing Resources

In one of the future labs, we will allow end-users to take a picture and attach it to a ToDo. However, to support that, we need first to create two Entities to save that information in the database: Entity **Resource** and Static Entity **ResourceType**. The first represents the resource itself, with the file and its name, while the second will distinguish if the resource is a picture, audio, or other (e.g: video).

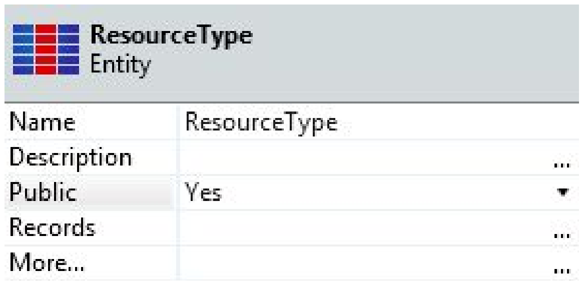
1) Create a new Static Entity named *ResourceType* and set it to **Public**. Specify the different types of resources as Records: *Audio*, *Image* and *Other*.

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



    b) Set the **Name** of the Static Entity to *ResourceType*.

    c) Set its **Public** property to *Yes*.



ResourceType
Entity

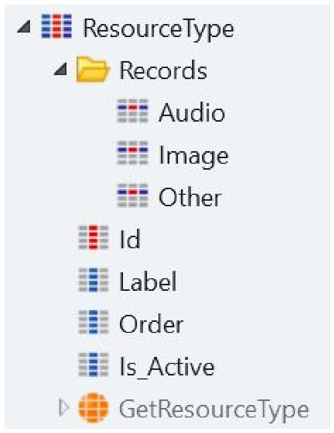
Name	ResourceType
Description	...
Public	Yes
Records	...
More...	...

    d) Expand the **ResourceType** Static Entity by clicking the triangle icon.

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

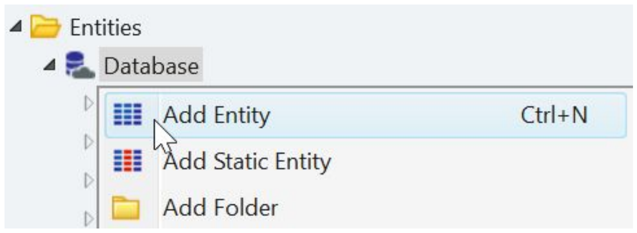
    f) Enter *Audio* for the record identifier.

    g) Add 2 more records to ResourceType: *Image* and *Other*. The Static Entity should look like the following screenshot



2) Create the **Resource** Entity to store information about ToDo resources. Set it as **Public** and **expose it with write permissions**. Make sure the **Data Type** of the **Id** attribute is set to *ToDo Identifier*. Add the following attributes: *ResourceTypeId* (Type: *ResourceType Identifier*), *Filename* (Type: *Text*) , *MimeType* (Type: *Text*) and *BinaryContent* (Type: *Binary Data*). All of these attributes should be set as **mandatory**.

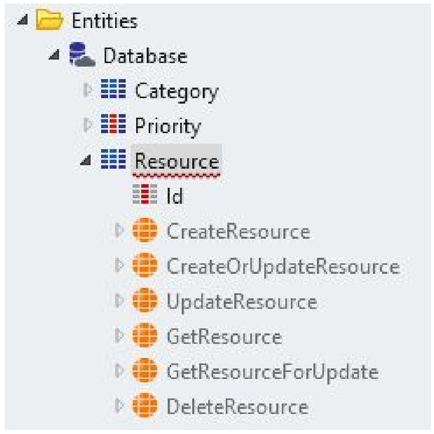
   c) Right click the **Database** element and select **Add Entity**.



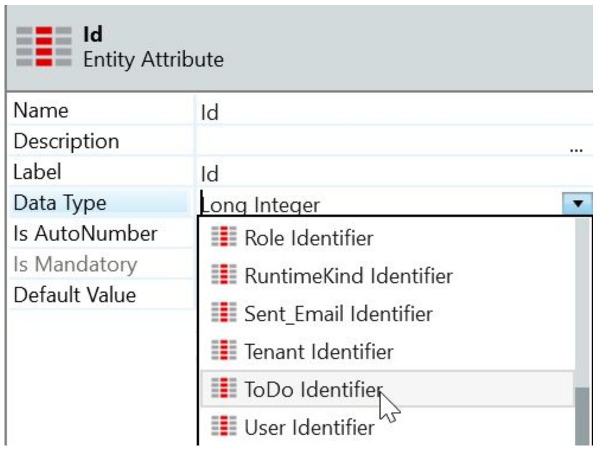
   d) Set the Name of the Entity to *Resource*.

   e) Change the **Public** property to *Yes* and the **Expose Read Only** to *No*.

   f) Expand the **Resource** Entity to view existing attributes and the six Entity Actions.



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

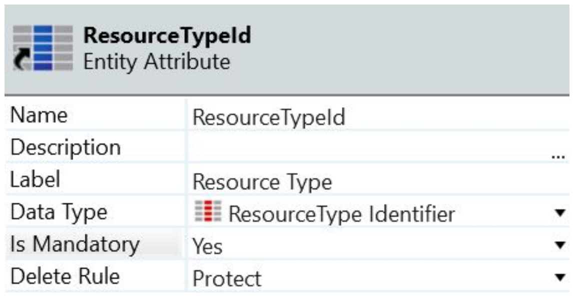


Id	Entity Attribute
Name	Id
Description	...
Label	Id
Data Type	Long Integer
Is AutoNumber	
Is Mandatory	
Default Value	

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*.

h) Add a new attribute and **Name** it *ResourceTypeId*.

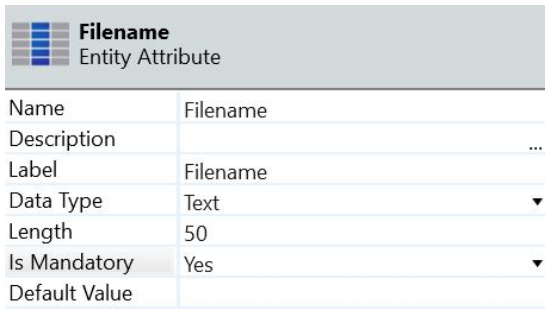
i) In the properties area of the **ResourceTypeId** attribute, set the **Is Mandatory** property to *Yes* and verify that the **Data Type** was automatically set to *ResourceType Identifier*.



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

j) Add a new attribute to the **Resource** Entity and name it *Filename*.

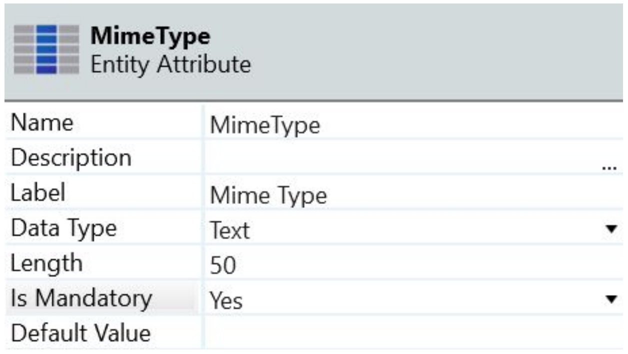
k) Verify that the attribute **Data Type** is set to *Text*, and make it **mandatory**.



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

l) Add a new attribute to the **Resource** Entity and name it *MimeType*.

m) Verify that the attribute **Data Type** is set to **Text**, and make it mandatory.

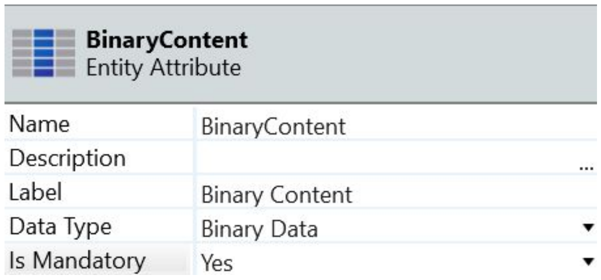


MimeType
Entity Attribute

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

n) Add a new attribute to the **Resource** Entity and name it *BinaryContent*.

o) Verify that the attribute **Data Type** is set to *Binary Data*, and make it mandatory.



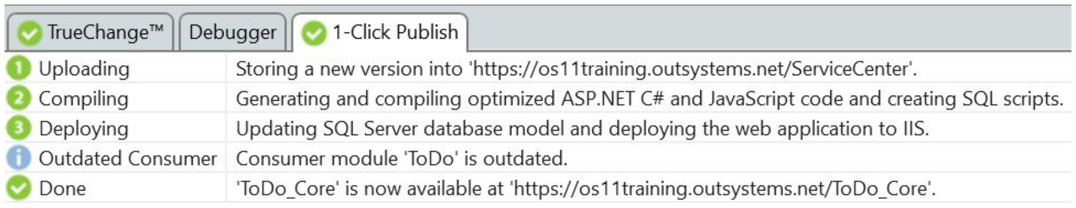
BinaryContent
Entity Attribute

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

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.

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

q) Make sure the module was published successfully, with a message indicating that the ToDo module is outdated. This will be addressed in the next Lab.



Uploading	Storing a new version into 'https://os11training.outsystems.net/ServiceCenter'.
Compiling	Generating and compiling optimized ASP.NET C# and JavaScript code and creating SQL scripts.
Deploying	Updating SQL Server database model and deploying the web application to IIS.
Outdated Consumer	Consumer module 'ToDo' is outdated.
Done	'ToDo_Core' is now available at 'https://os11training.outsystems.net/ToDo_Core'.

End of Lab

In this exercise, we created an initial data model for the ToDo mobile application, in its Core module. The main application concept is the **ToDo**, represented as an Entity, which will correspond to a table in the database.

The To Dos have a **Category**, a **Priority**, and a **User** associated with it. For that, the ToDo application also has a **Category** Entity, bootstrapped from an Excel File, a **Priority** Static Entity with the *Low*, *Medium* and *High* priorities, and user the System **User** Entity, to represent the users of the app.

We created 1-to-many relationships between the ToDo Entity and all the three other Entities, to store in the ToDo table its category, priority and the user that created it.

Finally, we created the **Resource** Entity and the **ResourceType** Static Entity to represent resource attachments that a ToDo may have, which can be *Audio*, *Image* or *Other*.

The application module was then published to the server, thus creating the appropriate database tables for these Entities. We will start visualizing these Entities, and their data, in the next lab.