Retrieving Data from a REST API (GET)

Table of Contents

knowledge@outsystems.com

www.outsystems.com

Introduction

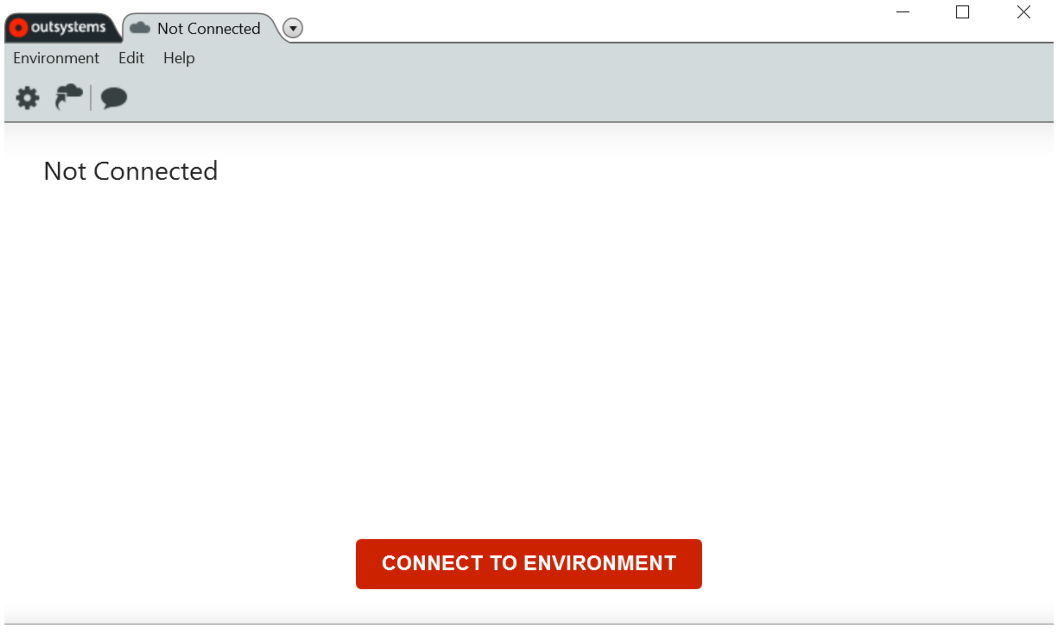
In this lab, we are going to integrate with a REST Web Service to retrieve data from an external system. We are going to use the GET method to read information of the Contacts in the external system and then display that data in our application.

Connect to an Environment

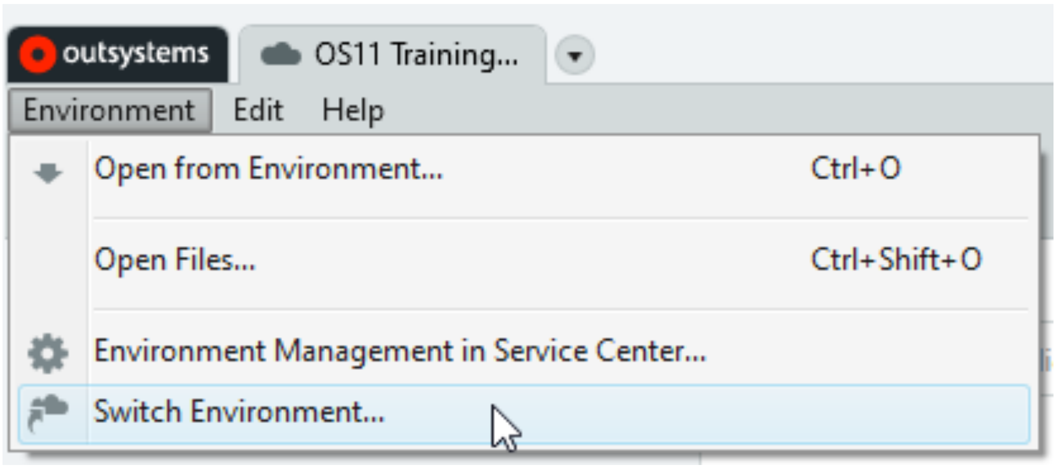
When we open Service Studio for the first time, we will need to connect to an **environment** where the OutSystems platform server generates, optimizes, compiles, and deploys OutSystems applications.

1) Open Service Studio and access the **Connect to Environment** dialog. This can be done in two ways.

   a) If you are not logged in to any environment, click on **Connect to Environment**.



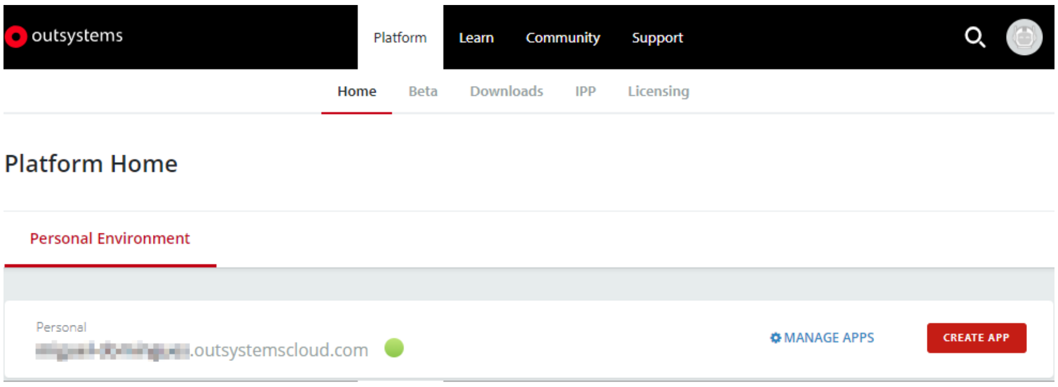
b) If you are already logged in to an environment, select the **Switch Environment...** option from the Environment menu at the top.


outsystems
OS11 Training...
Environment
Edit
Help
Open from Environment... Ctrl+O
Open Files... Ctrl+Shift+O
Environment Management in Service Center...
Switch Environment...

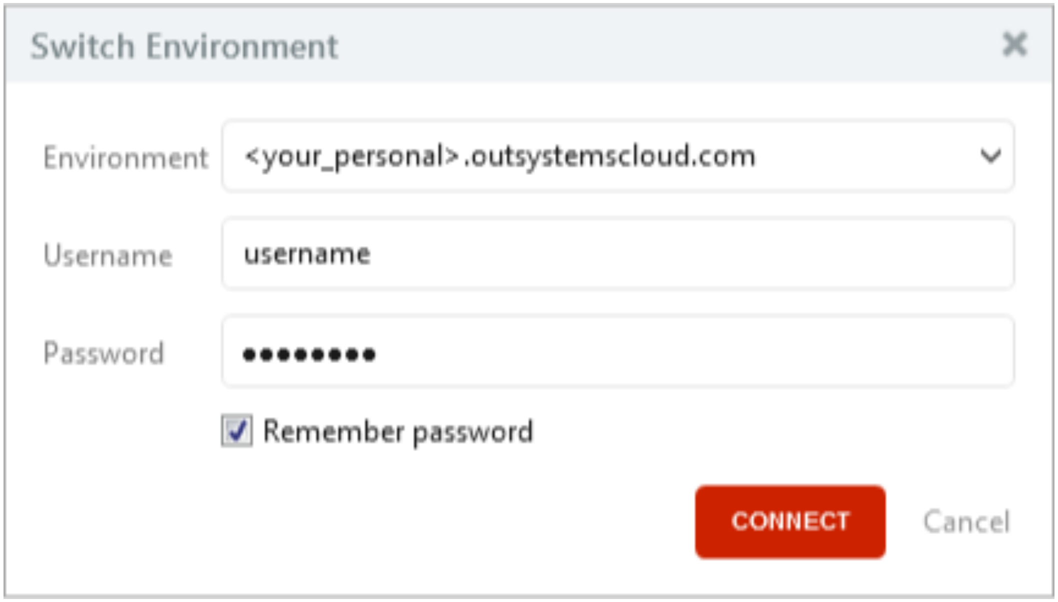
2) Connect to your OutSystems personal environment.

a) If you are using your Personal Environment, you can find its address in the OutSystems website and log in.

b) Under the **Platform** tab and then under the **Personal Environment** tab the environment address (or **Server Address**) can be found.



c) Back in Service Studio, use that Environment and login with your OutSystems community email (username) and password.



Switch Environment

Environment <your_personal>.outsystemscloud.com

Username username

Password ●●●●●●●●●

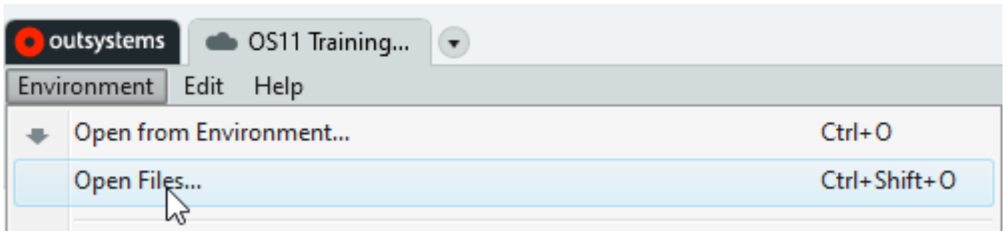
[✓] Remember password

Get to know the scenario

Install the Contacts application

Open and publish the **REST Contacts - GET.oap** in your personal environment. The oap file can be found in the Resources folder.

1) In the Applications, open the Environment menu and select **Open Files..**



2) In the Open dialog, change the File Type dropdown option to **OutSystems Application Pack (*.oap)** and then open the REST Contacts - GET.oap.

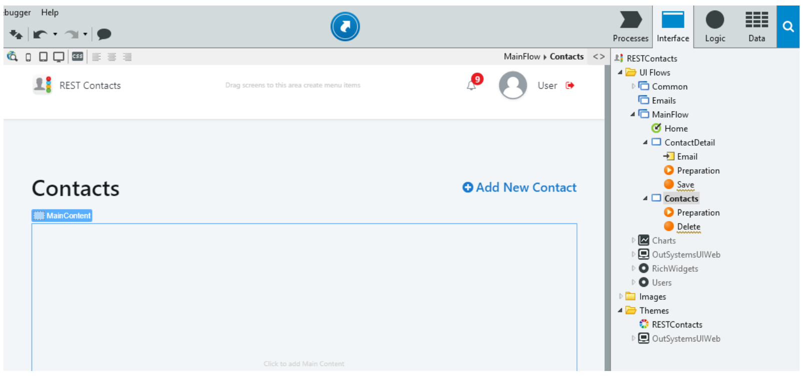
3) Click Proceed when asked.

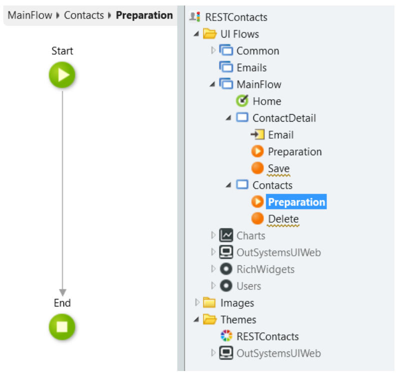
4) Wait for the installation to complete and then proceed.

Business Case Overview

The REST Contacts application is a simple application, only containing a couple empty Screens, Preparation and some Screen Actions, also empty. These elements is where we'll create the logic to get the Contacts and display the list on the screen.

The quick start application is simply to speed up the setup part and start right away working with the external REST web service.





Define the REST API Method - GET

In OutSystems, integration with REST Services can be straightforward. OutSystems helps us to generate all the methods and data structures needed to integrate with an external system.

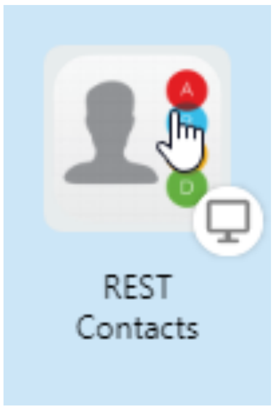
The GET request is known as a an idempotent and safe method as there are no changes in the resource when calling it and the results are always the same for identical requests.

Before you consume any REST API it's important to gather all the information you need from the REST API documentation. Information such as the expected structures and some examples maybe be really useful when consuming an external service. In this lab, the documentation for the external REST Web Service is available here.

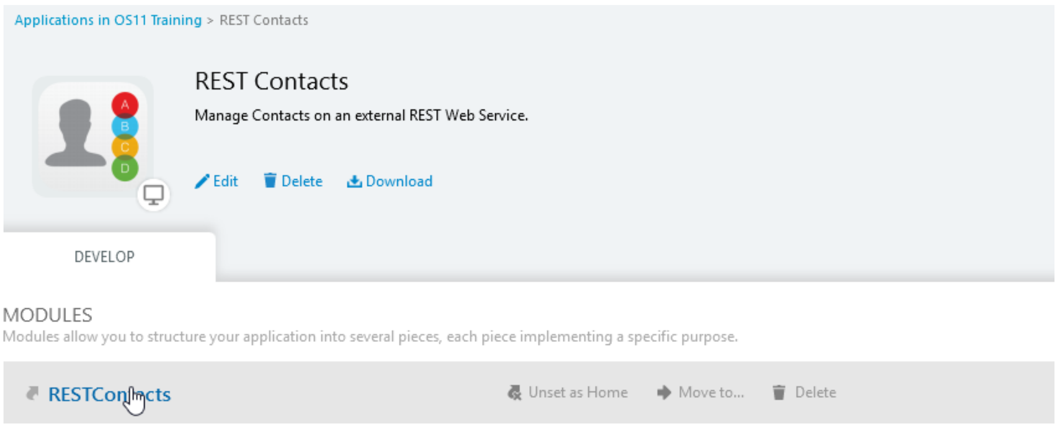
In this section we will create the integration with the external REST Web Service. The data that is retrieved from this Web Service will later on be used and displayed on a Web Screen.

1) Open the Contacts module

   a) In the Applications list, locate the REST Contacts Web application and open it.

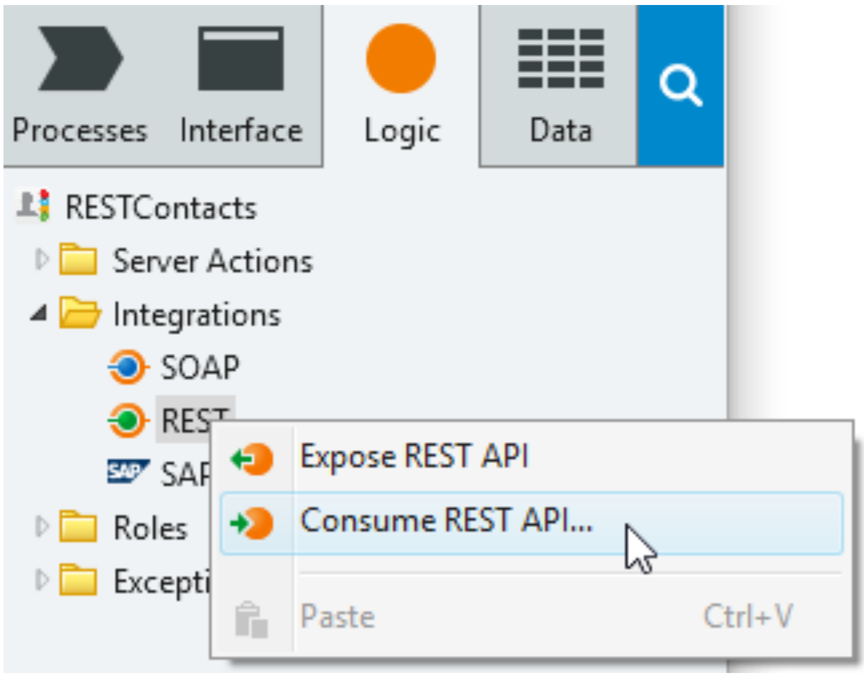


b) Open the **RESTContacts** module

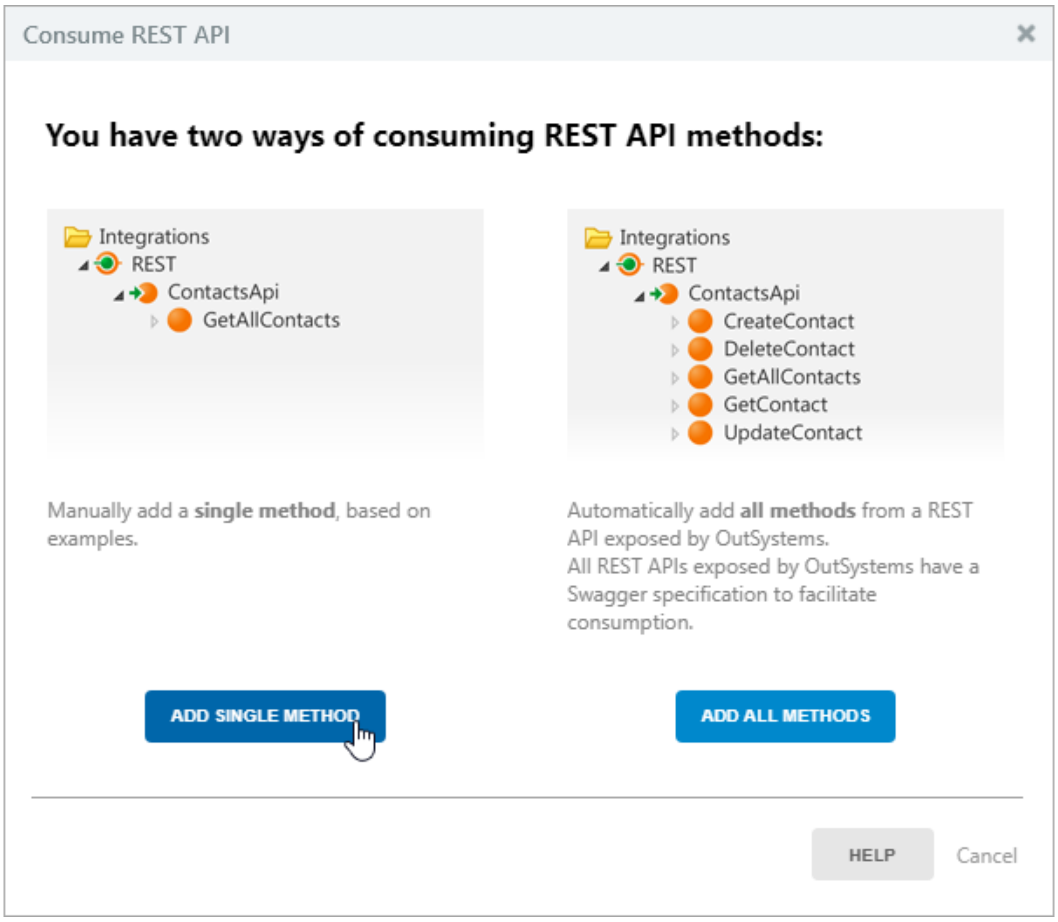

Applications in OS11 Training > REST Contacts
REST Contacts
Manage Contacts on an external REST Web Service.
Edit
Delete
Download
DEVELOP
MODULES
Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.
RESTContacts
Unset as Home
Move to...
Delete

2) Consume the GET method of the external REST Web Service.

a) Switch to the Logic tab and in the Integrations folder, right-click the REST integration element and select *Consume REST API...*. This opens a window to configure the REST method that you want to consume.



b) On the new window, click the **Add Single Method** button.


Consume REST API

You have two ways of consuming REST API methods:

Manually add a **single method**, based on examples.

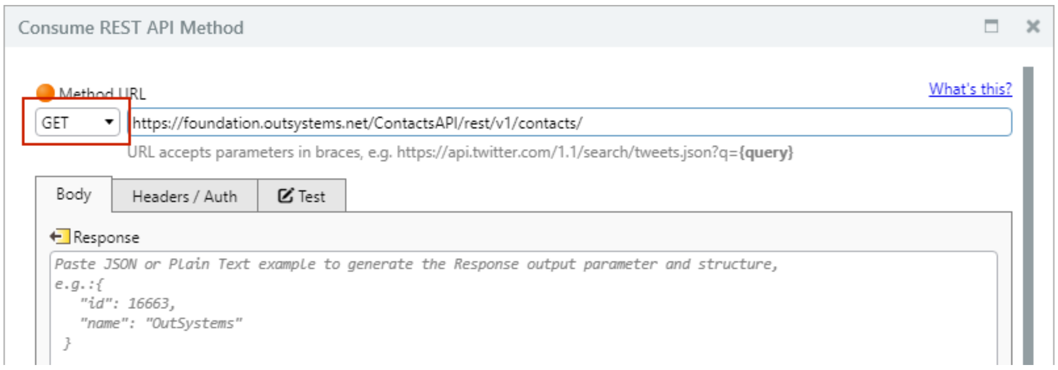
Automatically add **all methods** from a REST API exposed by OutSystems.
All REST APIs exposed by OutSystems have a Swagger specification to facilitate consumption.

NOTE: The Add All Methods option allows to integrate with REST Web Services that provide a Swagger definition file. This speeds up the integration process a lot, however not all Web Services provide this. Regardless if Swagger is available or not, you can use the Single Method option.

c) Set the Method URL to

```
https://foundation.outsystems.net/ContactsAPI/rest/v1/contacts/
```

d) Keep the **HTTP Method** to *GET*.

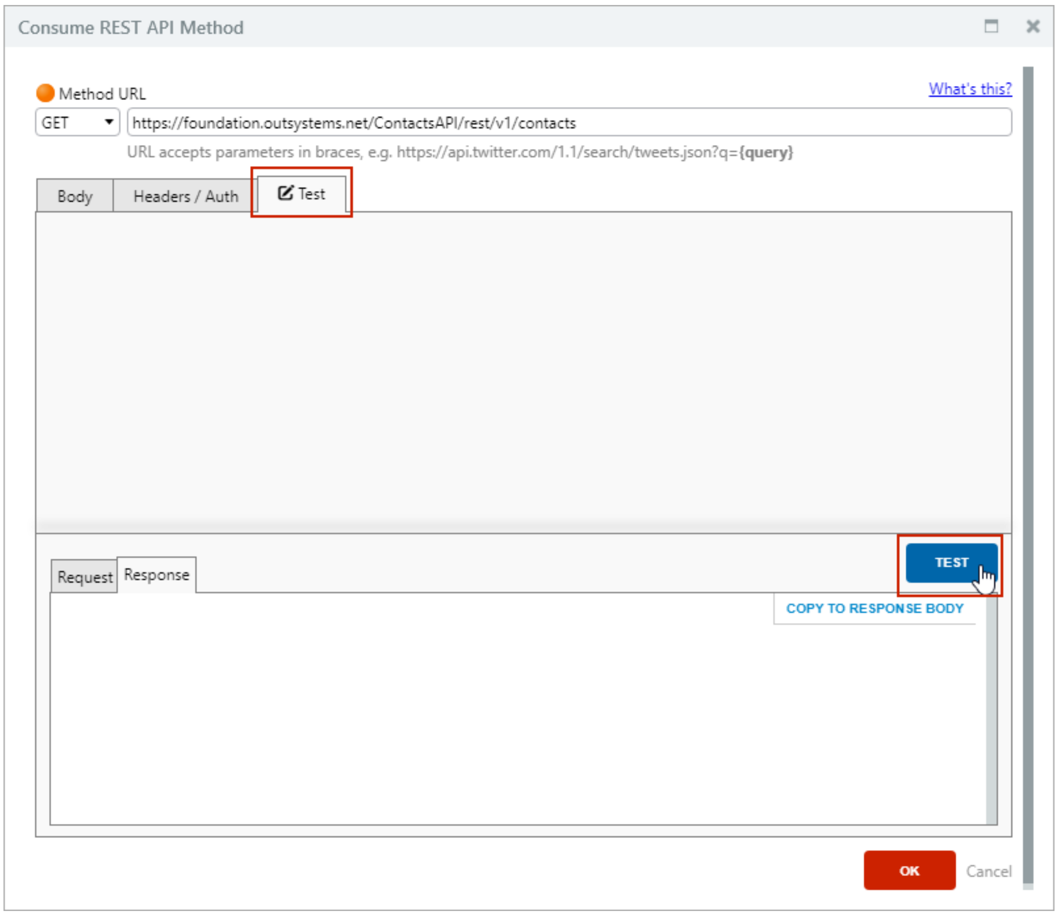


https://foundation.outsystems.net/ContactsAPI/rest/v1/contacts/

URL accepts parameters in braces, e.g. https://api.twitter.com/1.1/search/tweets.json?q={query}

```
Paste JSON or Plain Text example to generate the Response output parameter and structure,
e.g.:{
   "id": 16663,
   "name": "OutSystems"
 }
```

e) Switch to the **Test** tab, then click the Test button



https://foundation.outsystems.net/ContactsAPI/rest/v1/contacts

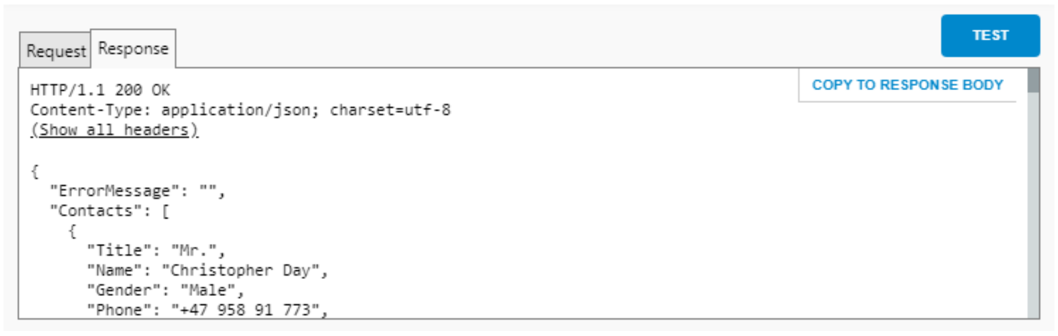
URL accepts parameters in braces, e.g. https://api.twitter.com/1.1/search/tweets.json?q={query}

NOTE: Service Studio allows you to Test REST API methods. This feature will also speed up the integration since the data type of the response can be automatically inferred. In situations where Test is not available or not possible you can set a sample Body response in the Body tab. Usually documentation provide some sample responses.

For this particular case, the sample response found in the documentation is like this:

```
{
"ErrorMessage": "string",
"Contacts": [
{
"Title": "string",
"Name": "string",
"Gender": "string",
"Phone": "string",
"Birthday": "2014-12-31",
"Occupation": "string",
"City": "string",
"State": "string",
"Country": "string",
"Email": "string",
"Company": "string"
}
],
"Success": true
}
```

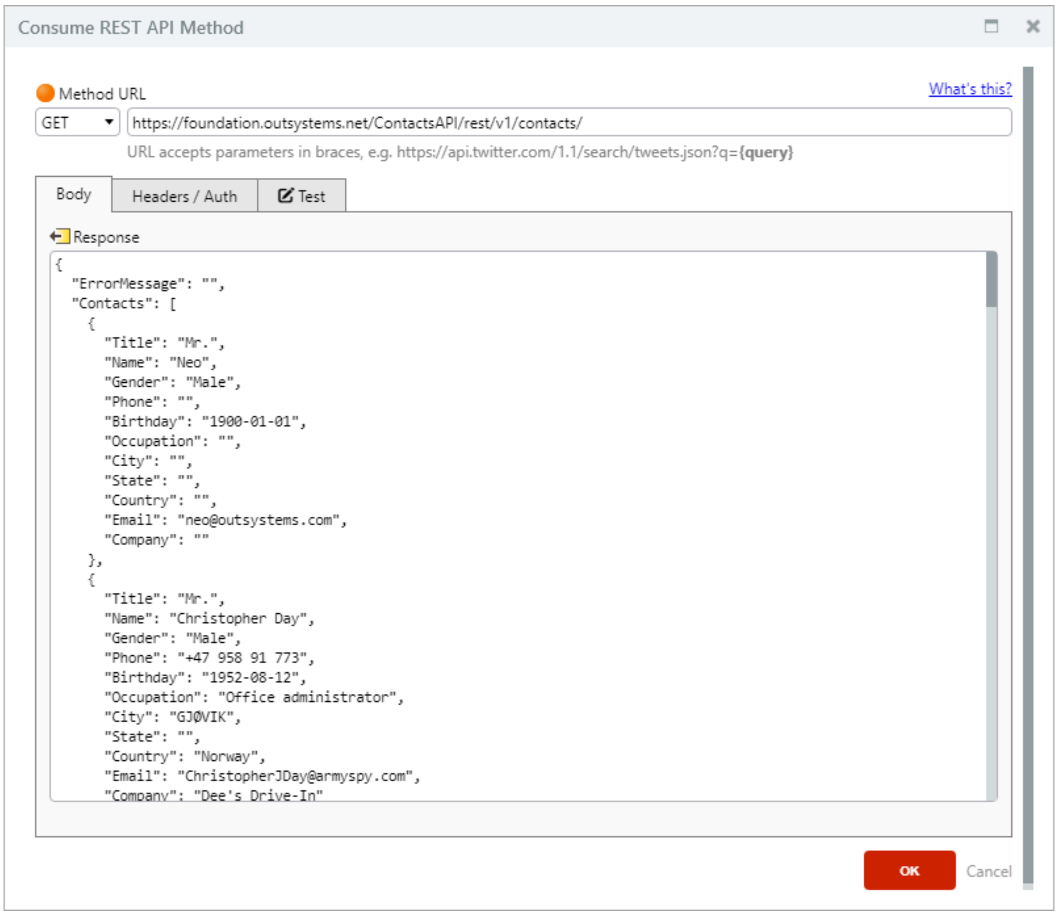
f) After clicking the Test button, the Response tab at the bottom should be populated with the data retrieved from invoking the REST API method.



```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
(Show all headers)

{
  "ErrorMessage": "",
  "Contacts": [
    {
      "Title": "Mr.",
      "Name": "Christopher Day",
      "Gender": "Male",
      "Phone": "+47 958 91 773",
```

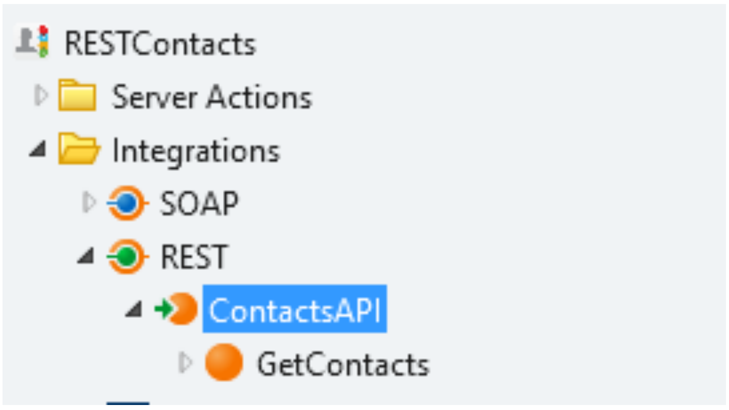
g) Click the **Copy to Response Body** button. The Body tab should now have the sample response obtained in the previous step.


Consume REST API Method
Method URL
What's this?
GET
https://foundation.outsystems.net/ContactsAPI/rest/v1/contacts/
URL accepts parameters in braces, e.g. https://api.twitter.com/1.1/search/tweets.json?q={query}
Body
Headers / Auth
Test
Response
OK
Cancel

```
{
  "ErrorMessage": "",
  "Contacts": [
    {
      "Title": "Mr.",
      "Name": "Neo",
      "Gender": "Male",
      "Phone": "",
      "Birthday": "1900-01-01",
      "Occupation": "",
      "City": "",
      "State": "",
      "Country": "",
      "Email": "neo@outsystems.com",
      "Company": ""
    },
    {
      "Title": "Mr.",
      "Name": "Christopher Day",
      "Gender": "Male",
      "Phone": "+47 958 91 773",
      "Birthday": "1952-08-12",
      "Occupation": "Office administrator",
      "City": "GJØVIK",
      "State": "",
      "Country": "Norway",
      "Email": "ChristopherJDay@armyspy.com",
      "Company": "Dee's Drive-In"
```

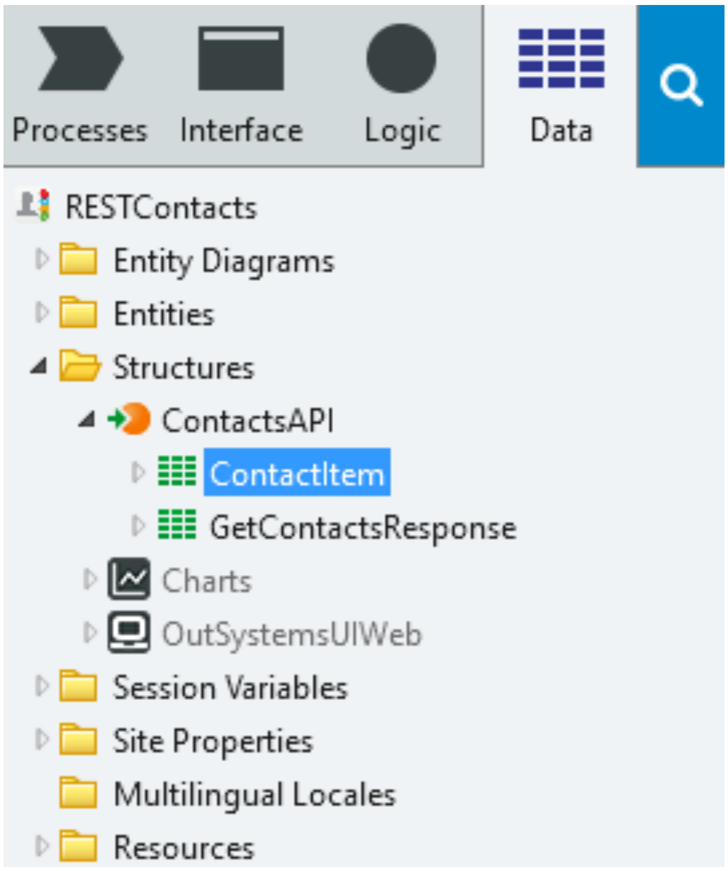
h) Click the **OK** button to close the Consume REST API Method.

i) Finally, change the name of the Integration from *FoundationOutSystems* to *ContactsAPI*.



NOTE: This step is not mandatory, but it allows developers to have custom or more familiar names in their integrations. You may also change the names of the methods which can be useful when the auto-generated names are more unfamiliar.

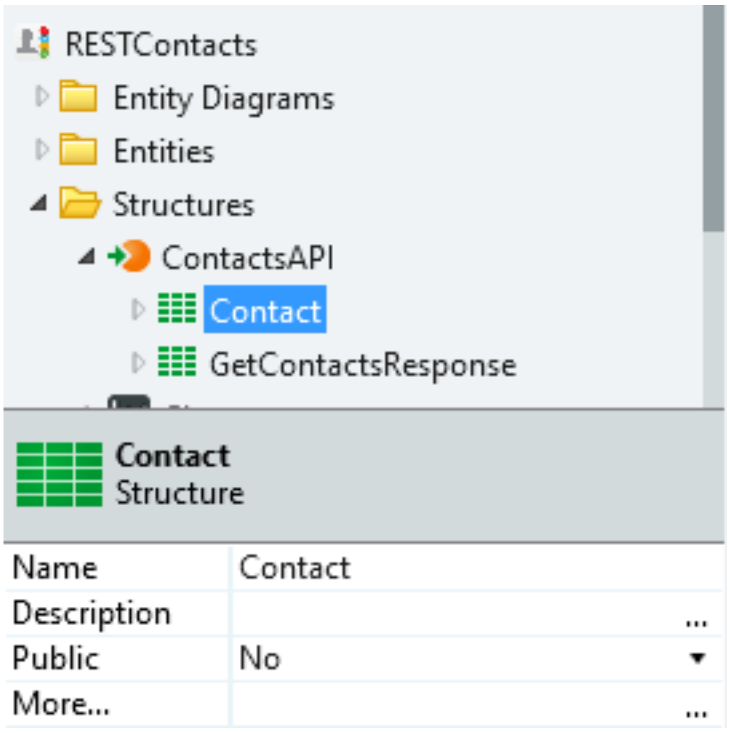
j) Switch to the Data tab, and locate the *ContactItem* Structure under the *ContactsAPI*.



The *ContactItem* and *GetContactsResponse* structures were automatically created by Service Studio when the API Method was added in previous steps.

Expanding the *ContactItem* structure allows you to see all attributes of Contact (Title, Name, Gender, ...). These attributes were also automatically inferred by Service Studio from the sample response of the API method.

k) Rename the structure to Contact.



Name	Contact
Description	...
Public	No
More...	...

NOTE: Just like renaming the API method, this step is also not mandatory. However, renaming it makes development easier since names become more familiar. It also allows for Service Studio to automatically set variables data types based on their names. For Instance, when creating a variable named Contact, Service Studio will automatically set its type to the Contact structure.

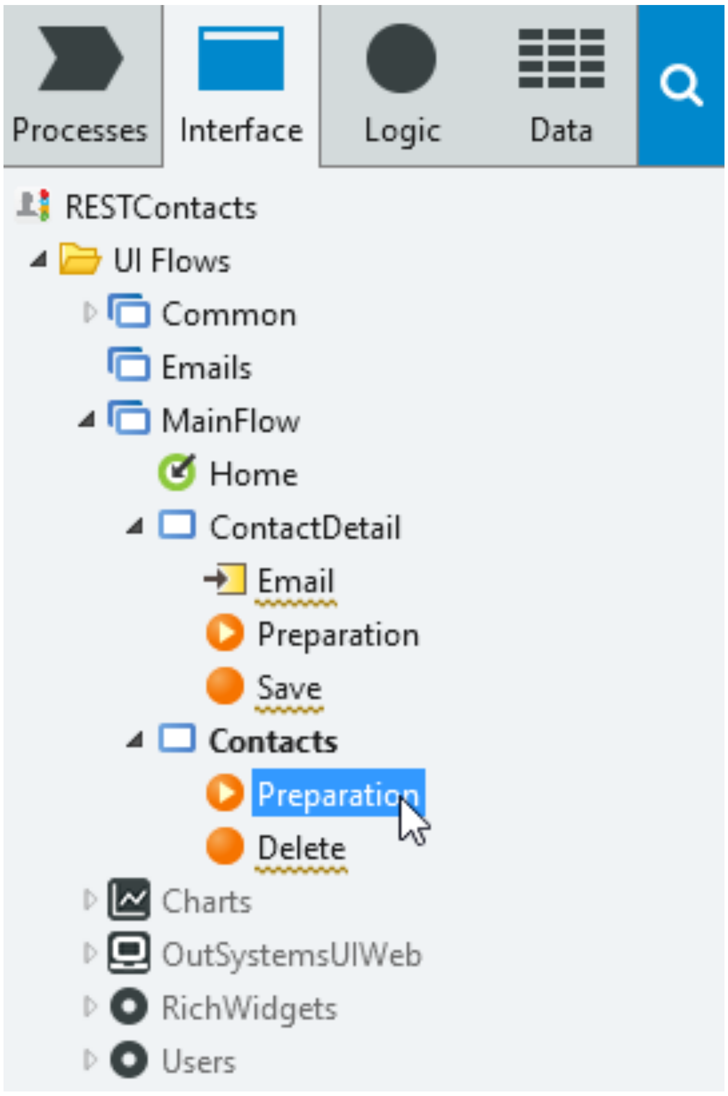
List Contacts on the Screen

According to the Screen lifecycle of OutSystems web applications, when the user is redirected to the Contacts Screen, firstly, the Preparation will be executed. Therefore, the Preparation of the Contacts Screen is where we will request the list of contacts from the external REST Service.

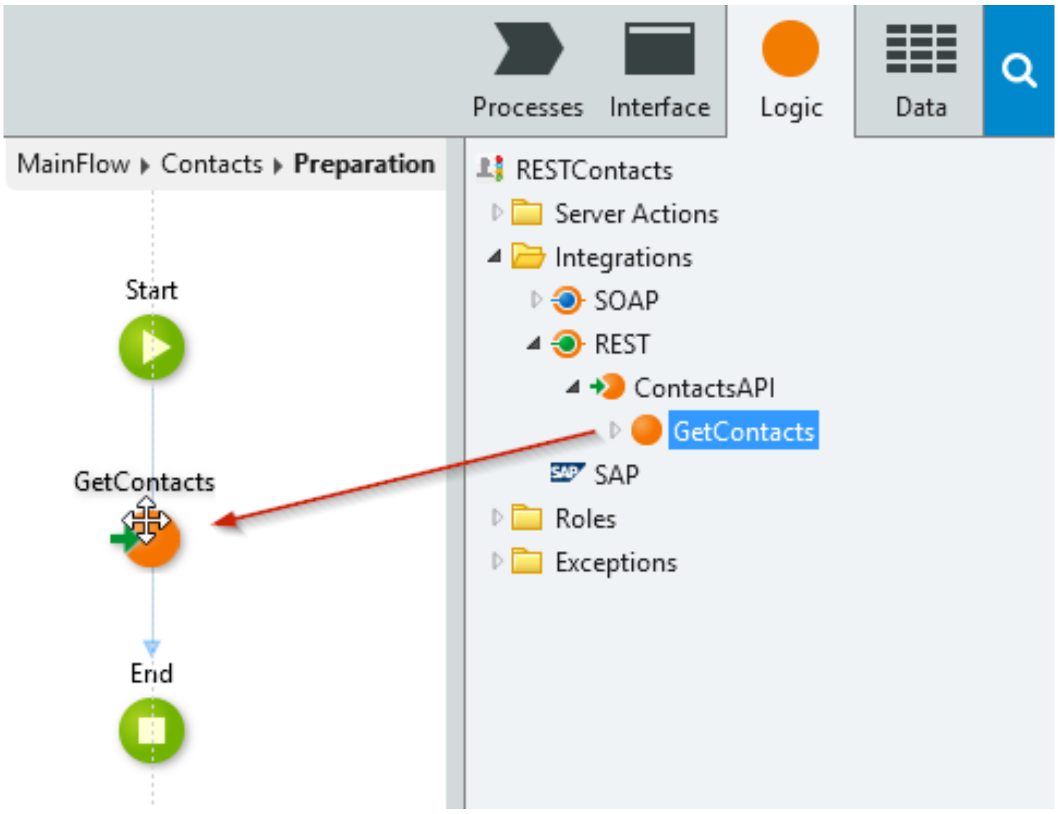
In our sample app, the Contacts screen has already been created. Let's now implement the Preparation that will retrieve the list of Contacts.

1) In the Preparation of the Contacts screen, invoke the ContactsAPI to retrieve the list of contacts from the External REST Web Service.

   a) In the Interface tab, double-click the Preparation of the **Contacts** Web Screen to open it in the canvas.



b) Switch to the Logic tab and locate the **GetContacts** REST method that we have just defined. Drag and drop the method on the Preparation.



2) Validate the result of the REST API method and show the error message on fail.

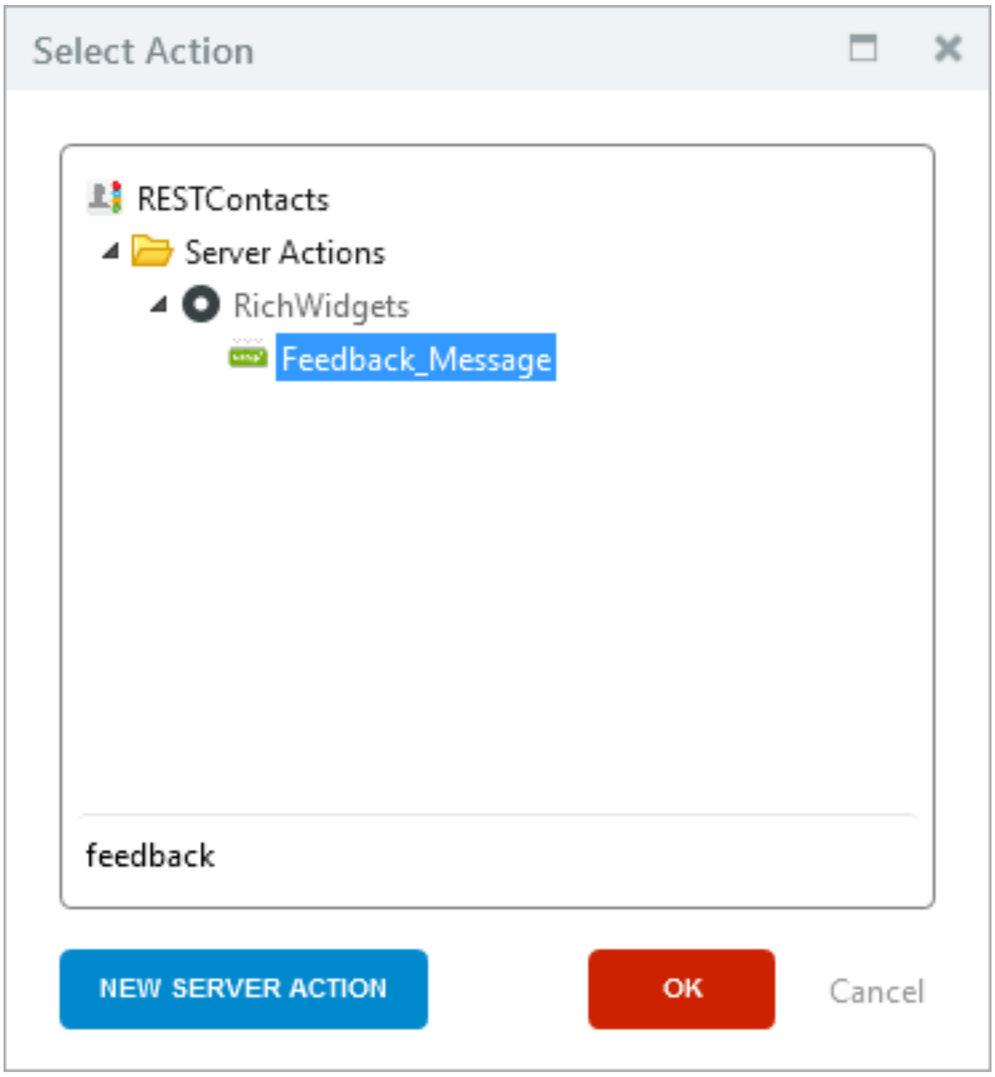
a) Drag an **If** and drop it after the GetContacts, then set its **Condition** to

```
GetContacts.Response.Success
```

b) Drag an **End** and drop it to the right of the If, then create the True branch connector from the If to the new End.

c) Drag a **Run Server Action** and drop it on the False branch connector of the If.

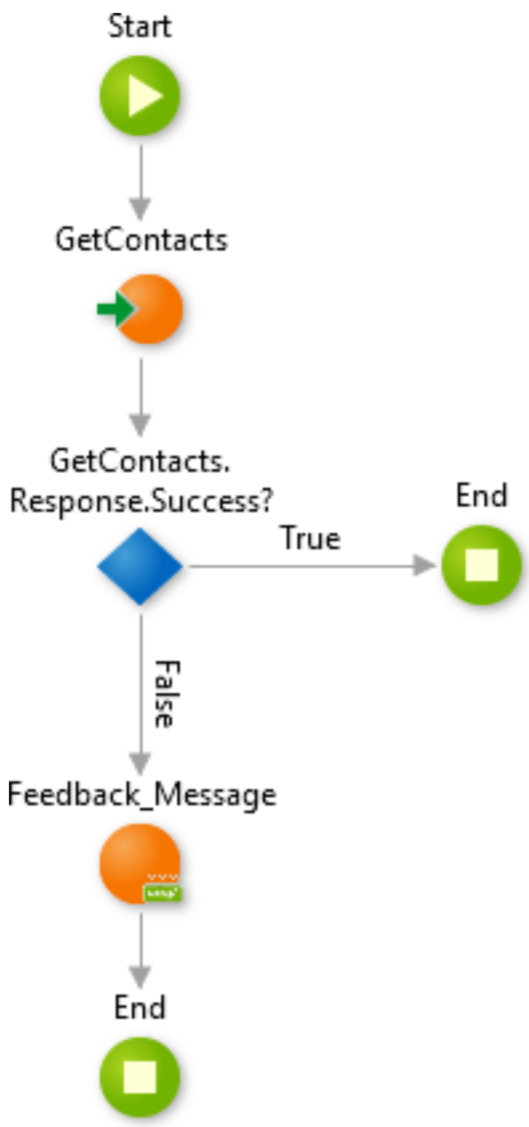
d) In the **Select Action** dialog, choose the *Feedback_Message* action.


Select Action

RESTContacts
Server Actions
RichWidgets
Feedback_Message

feedback

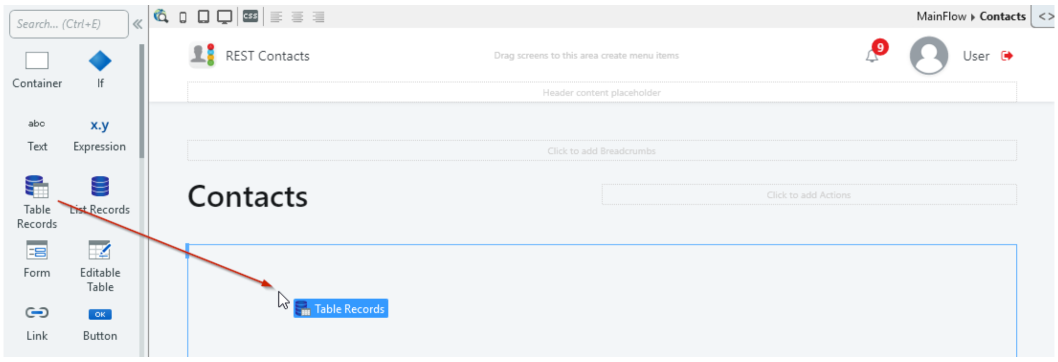
e) Set the **MessageText** parameter to `GetContacts.Response.ErrorMessage` and the **MessageType** to `Entities.MessageType.Error`.



3) Now that we have a way to retrieve a list of records we need to create the interface to display them to the user. For that we are going to show them in a Table Records widget.

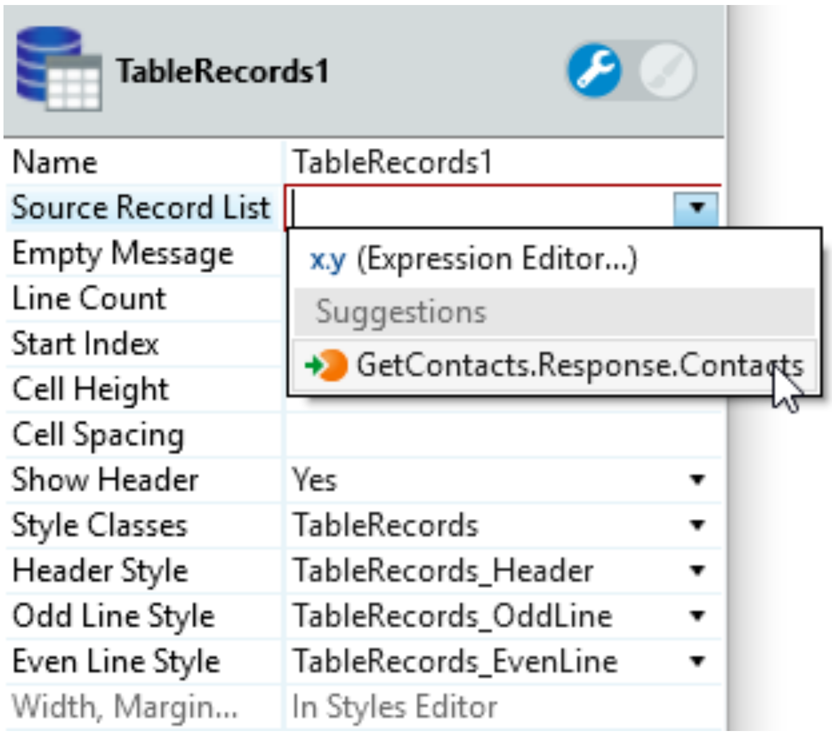
   a) Switching back to the Interface tab, double-click the **Contacts** Screen to open it in the canvas.

b) Drag and drop a **Table Records** to the Main Content.



c) Set the **Source Record List** property to

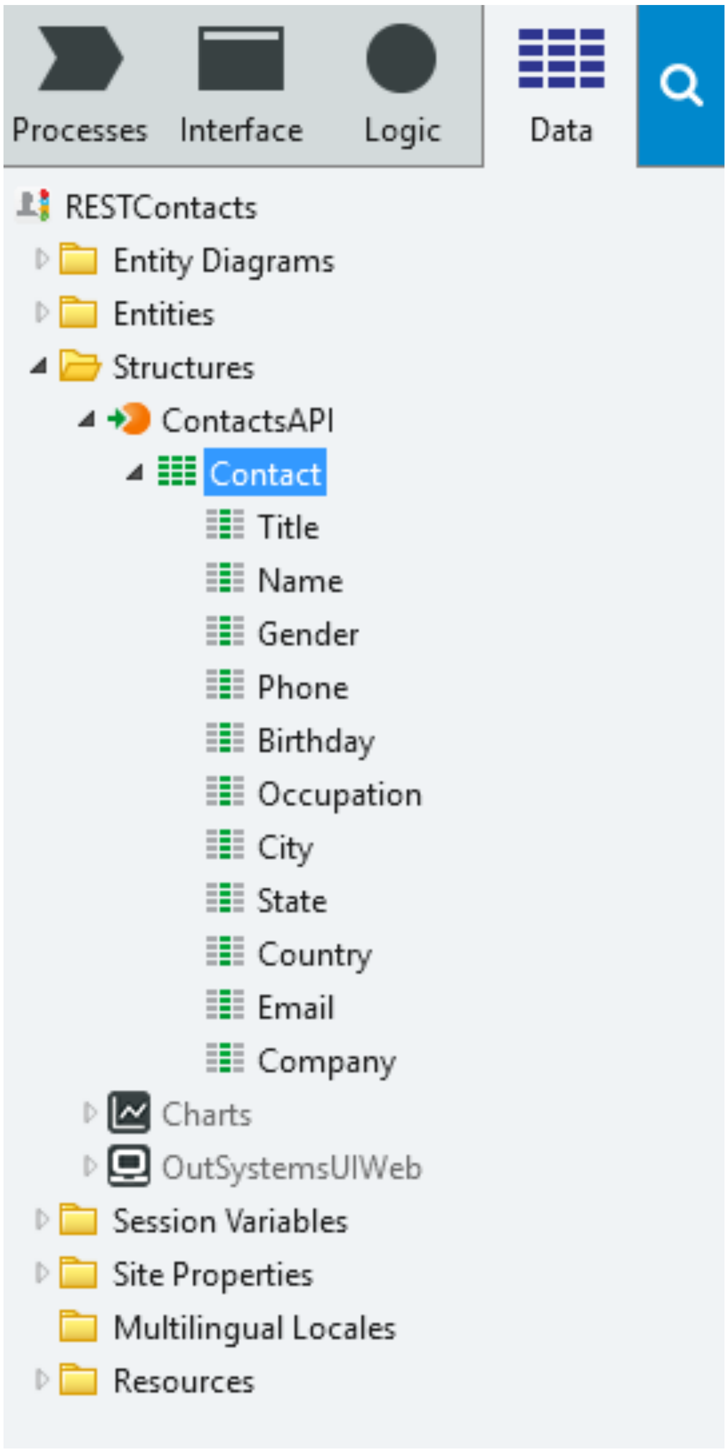
```
GetContacts.Response.Contacts
```



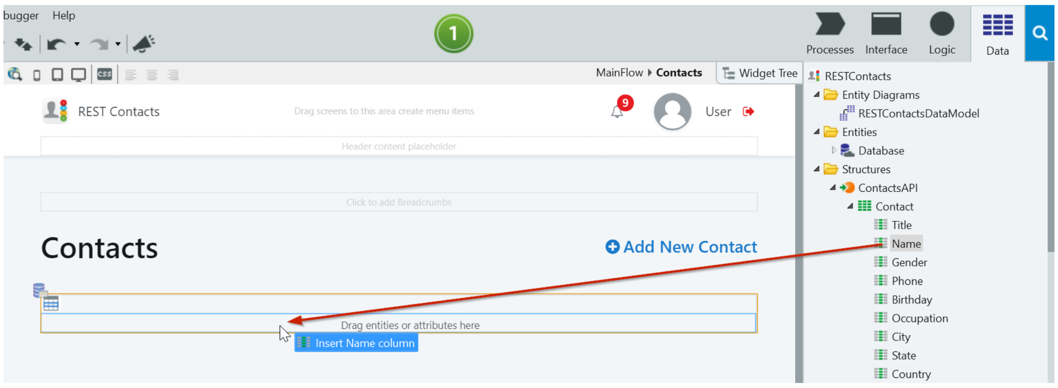
TableRecords1

Name	TableRecords1
Source Record List	
Empty Message	
Line Count	
Start Index	
Cell Height	
Cell Spacing	
Show Header	Yes
Style Classes	TableRecords
Header Style	TableRecords_Header
Odd Line Style	TableRecords_OddLine
Even Line Style	TableRecords_EvenLine
Width, Margin...	In Styles Editor

d) Let's switch to the Data tab and under the Structures folder locate the *Contact* structure.

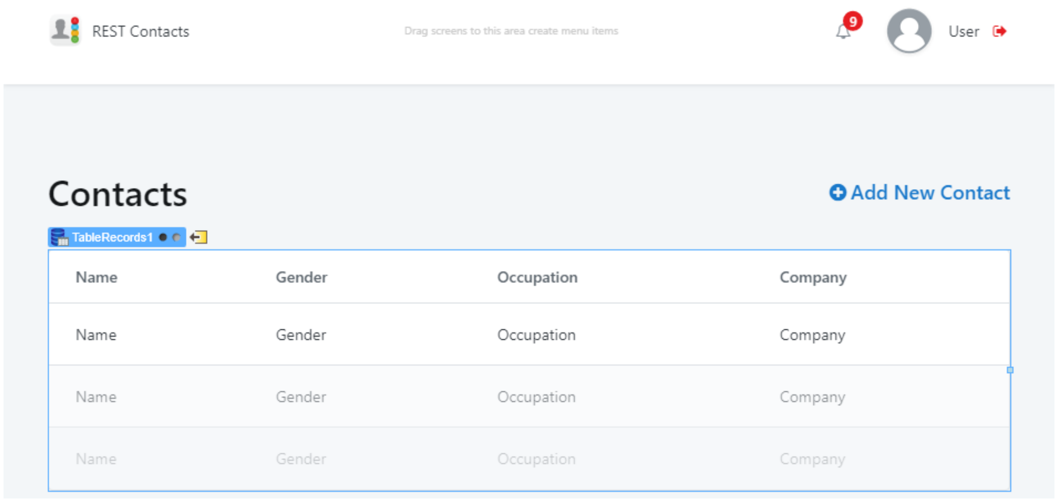


e) Drag the *Name* attribute and drop it in the Table Records.



f) Repeat the previous step for the *Gender*, *Occupation* and *Company* fields.

g) You should end up having the Contacts screen with a Table Records with 4 columns.

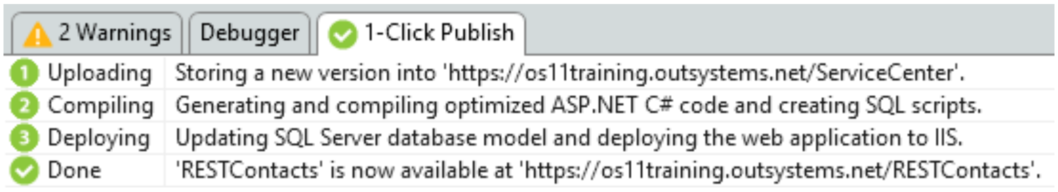


Name	Gender	Occupation	Company
Name	Gender	Occupation	Company
Name	Gender	Occupation	Company
Name	Gender	Occupation	Company

4) Publish the application using 1-Click Publish button and verify that the publish completed successfully in the 1-Click Publish Tab.

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

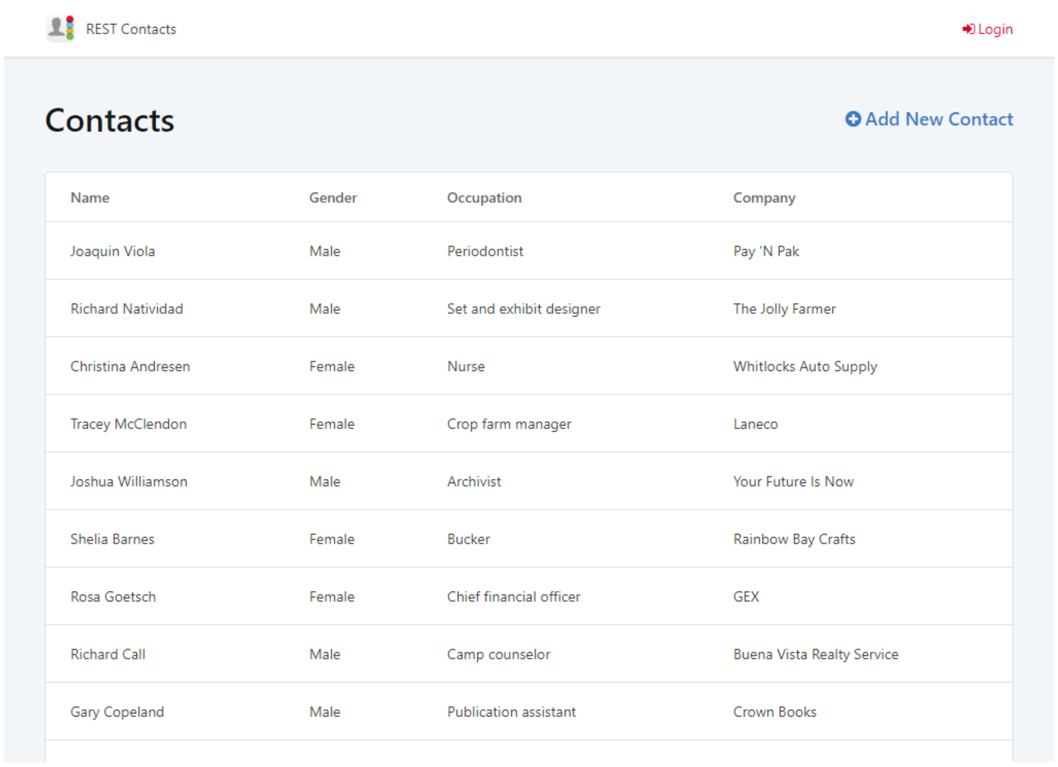
b) Verify in the 1-Click Publish tab that the publishing process was successful.



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

c) Preview the app in browser by clicking on the **Open in Browser** button.

d) The Contacts screen should be presented and display existing contacts.



Contacts

Name	Gender	Occupation	Company
Joaquin Viola	Male	Periodontist	Pay 'N Pak
Richard Natividad	Male	Set and exhibit designer	The Jolly Farmer
Christina Andresen	Female	Nurse	Whitlocks Auto Supply
Tracey McClendon	Female	Crop farm manager	Laneco
Joshua Williamson	Male	Archivist	Your Future Is Now
Shelia Barnes	Female	Bucker	Rainbow Bay Crafts
Rosa Goetsch	Female	Chief financial officer	GEX
Richard Call	Male	Camp counselor	Buena Vista Realty Service
Gary Copeland	Male	Publication assistant	Crown Books

NOTE: The contacts shown may vary depending on the data provided by the external system.

End Lab

In this lab, we integrated with a REST Web Service, namely a GET method to retrieve the list of Contacts provided by the external system.

To accomplish that, we have used the OutSystems visual interface to consume the REST method, that automatically infers the output structure of the method.

Once we consumed the API, we implemented the logic to display the information in the Contacts screen, firstly adding the method on the Preparation and then creating the interface to show some of the Contacts' data on the screen.

At the end of this exercise you should be able to integrate with a simple REST API to retrieve data and display it in your application.