

# Exposing Application Data using REST (GET)

## Table of Contents

<b>Introduction.....</b>	<b>2</b>
<b>Connect to an Environment.....</b>	<b>3</b>
<b>Get to know the scenario.....</b>	<b>5</b>
Install the Employees application	5
Business Case Overview	5
<b>Expose the List of Employees.....</b>	<b>6</b>
<b>Expose an Employee Information.....</b>	<b>20</b>
<b>Expose the Employee Picture.....</b>	<b>28</b>
<b>End Lab.....</b>	<b>35</b>

# Introduction

In this session, we are going to start implementing a REST API to expose existing data in our OutSystems application to an external system.

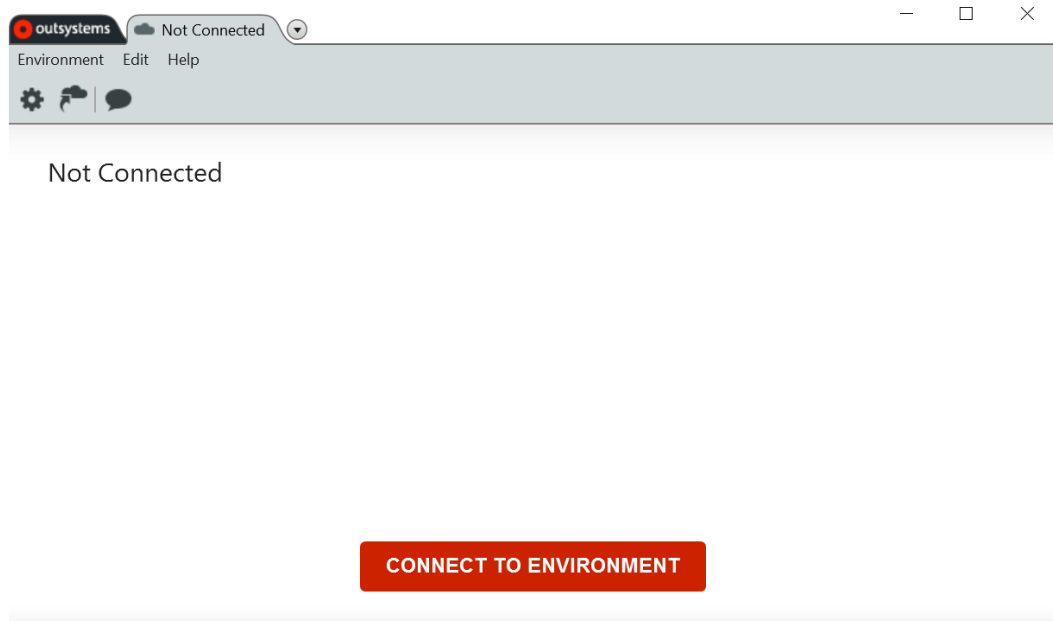
We will start by defining the API, and then move on to the implementation of the actual read methods. These will consist in methods that retrieve a list of Employees information; information of a single employee given an email as input; and the profile picture of the employee given its email.

To do this, we will use Service Studio to create the REST API methods, and the flows just like regular server actions. In these methods, input and output parameters will be defined and data fetched from the database entities that will then be sent via the REST API.

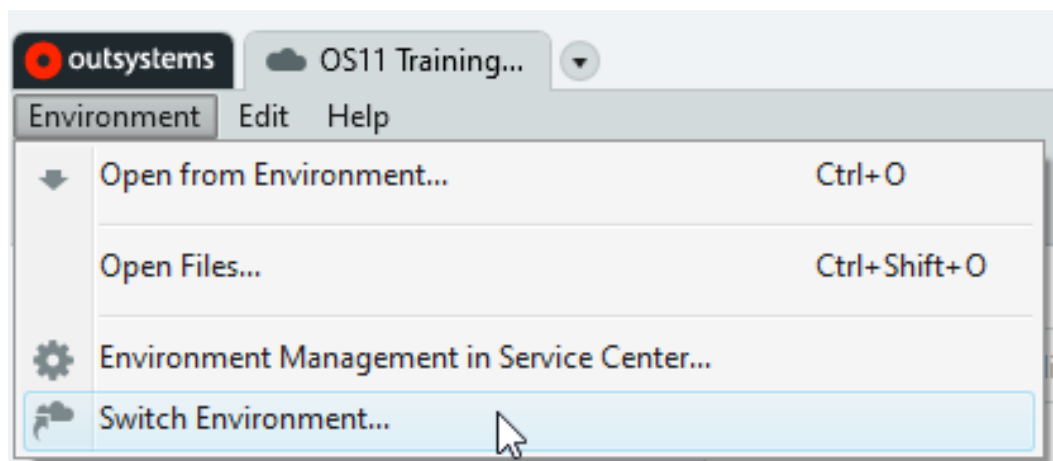
# Connect to an Environment

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.

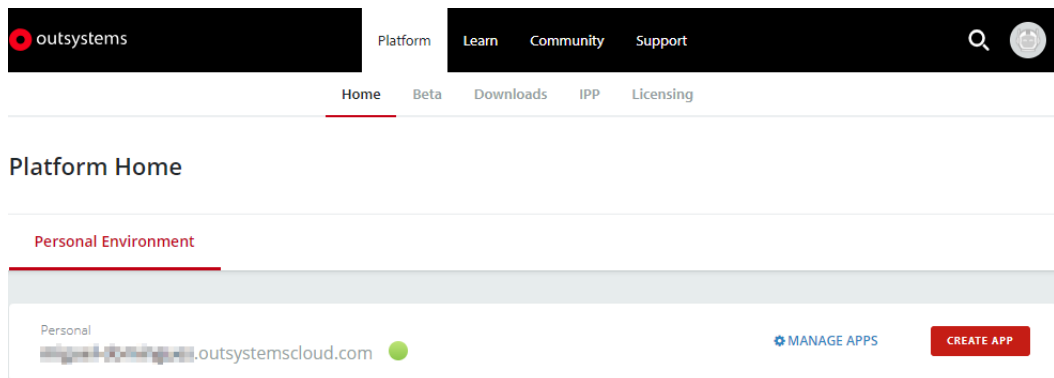
- 1) Open Service Studio and access the **Connect to Environment** dialog. This can be done in two ways.
  - a) If you are not logged in to any environment, click on **Connect to Environment**.



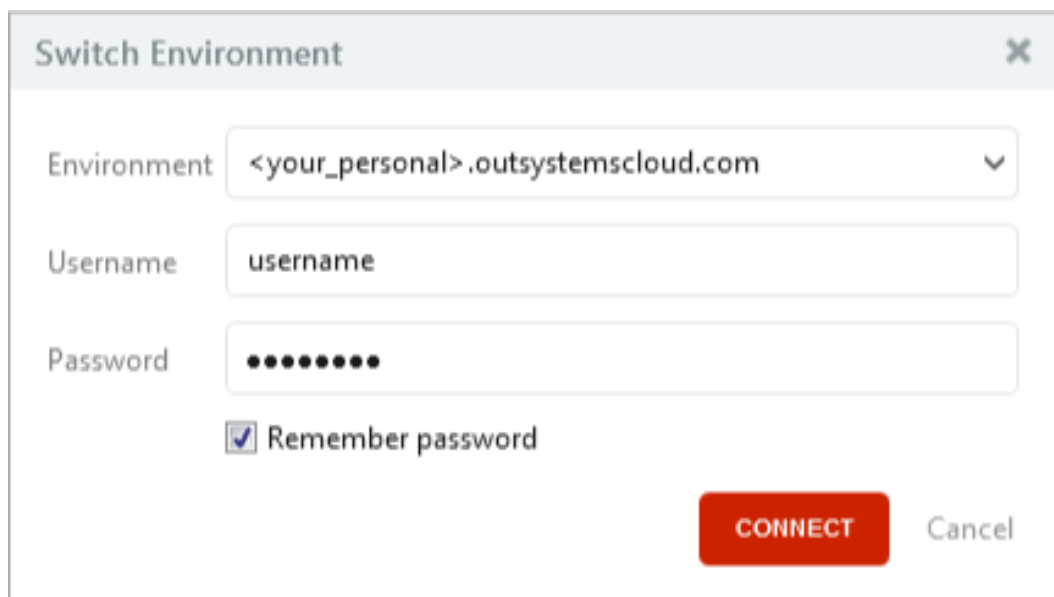
- b) If you are already logged in to an environment, select the **Switch Environment...** option from the Environment menu at the top.



- 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 **Platform Home** and then under **Personal Environment** the environment address (or **Server Address**) can be found.



- c) Back to Service Studio, use that Environment and **Connect** with your OutSystems community email (username) and password.

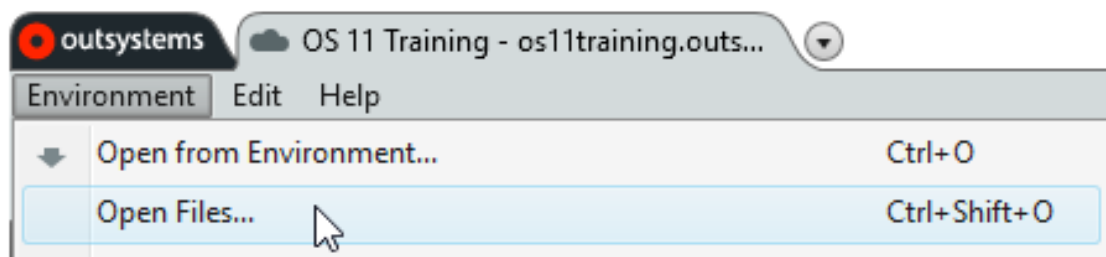

 The screenshot shows a 'Switch Environment' dialog box. It has a title bar with a close button (X). The dialog contains three input fields: 'Environment' with a dropdown menu showing '<your\_personal>.outsystemscloud.com', 'Username' with the text 'username', and 'Password' with masked characters. Below these fields is a checkbox labeled 'Remember password' which is checked. At the bottom right, there are two buttons: a red 'CONNECT' button and a 'Cancel' button.

# Get to know the scenario

## Install the Employees application

Open and publish the **Employees API - 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 Employees API - GET.oap.
- 3) Click Proceed when asked.
- 4) Wait for the installation to complete and then proceed.

## Business Case Overview

The Employees application centralizes employee information and makes it available to the entire organization. The application has three modules, EmployeesAPI\_Core, EmployeesAPI\_Demo and EmployeesAPI\_REST. These three modules contain the skeleton of the Employees Application.

The EmployeesAPI\_Core module contains the definition of the database entities that store the Employee information.

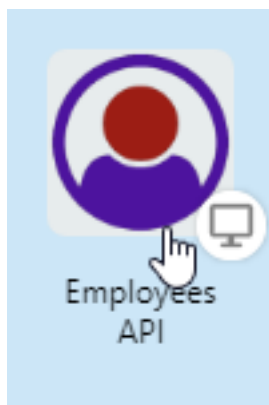
The EmployeesAPI\_Demo module contains a screen that displays the Employee information. This can be accessed using the address [https://<your-server>/EmployeesAPI\\_Demo/Employees.aspx](https://<your-server>/EmployeesAPI_Demo/Employees.aspx)

Finally, the EmployeesAPI\_REST is the module where we will be working in this session.


# Expose the List of Employees

We will be starting by opening the `EmployeesAPI_REST` module and create the REST API. Then we'll create our first method of the API, named `List`. This method will allow external systems to retrieve the list of Employees using a REST endpoint and in JSON format.

- 1) Open the *EmployeesAPI\_REST* module and define the basics to expose a REST API for Employees information.
  - a) In the applications list, open the Employees API web application



- b) Open the **EmployeesAPI\_REST** module




**Employees API**  
Small Employee app that exposes functionality via REST

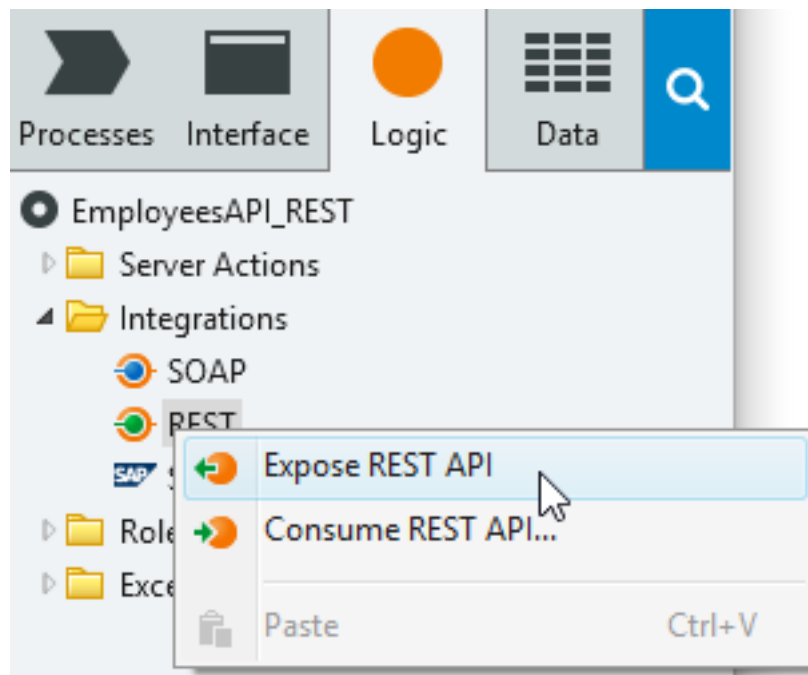
[Edit](#) [Delete](#) [Download](#)

DEVELOP

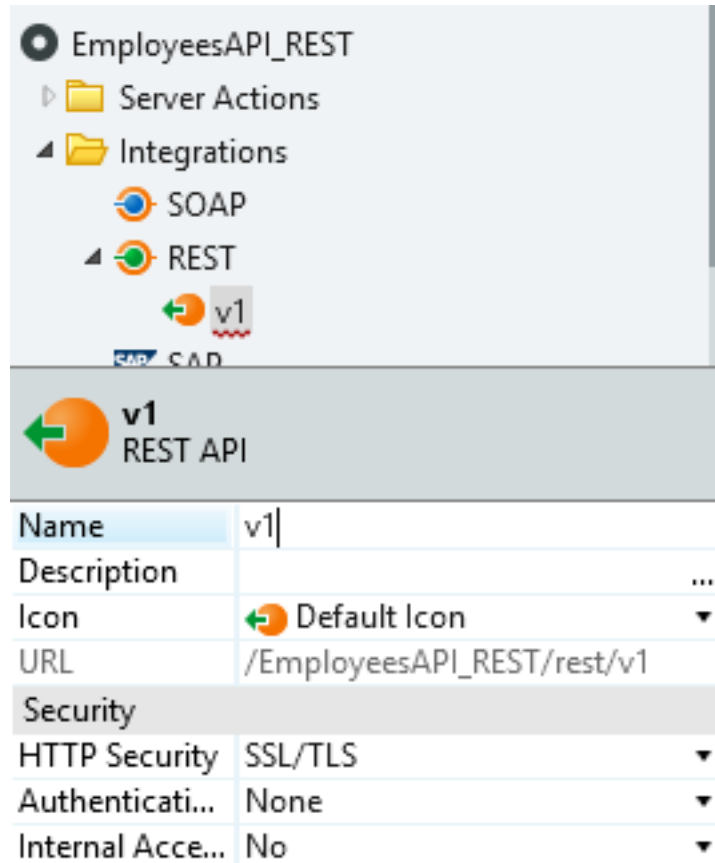
**MODULES**  
Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

<a href="#">EmployeesAPI_Core</a>	Changed yesterday 12:10 by Miguel Domingues
 <a href="#">EmployeesAPI_Demo</a>	Changed 14:10 by Miguel Domingues
<a href="#">EmployeesAPI_REST</a>	<a href="#">Set as Home</a> <a href="#">Move to...</a> <a href="#">Delete</a>

- c) In the Logic tab, right-click the REST element under the Integrations folder and select **Expose REST API**



- d) Rename the API to v1



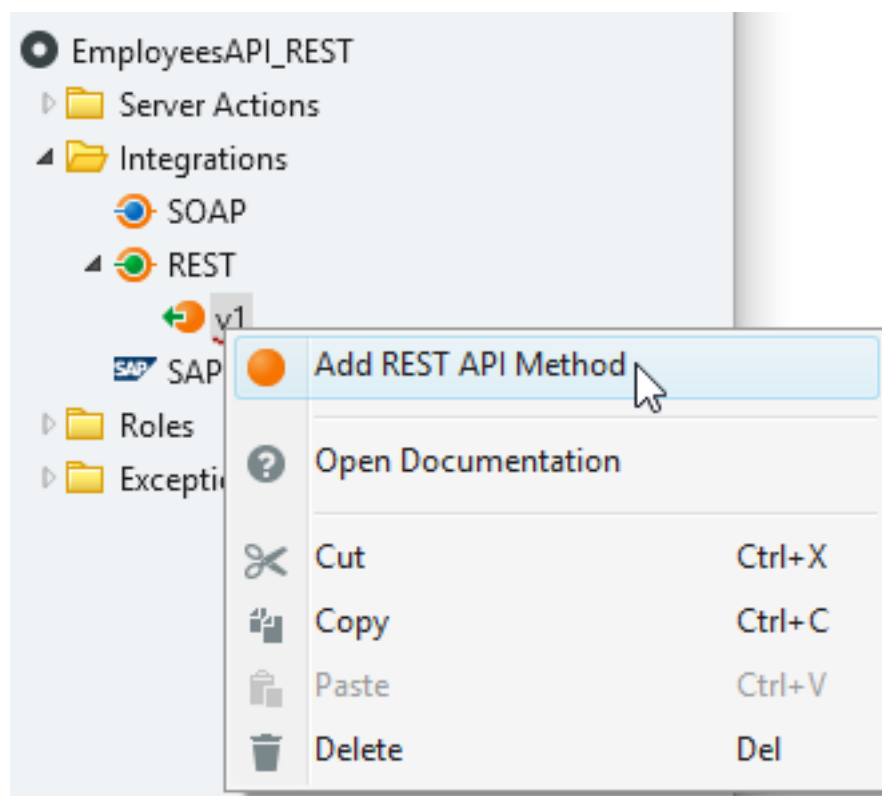
**NOTE:** The name of the API influences the generated URL. In this case, we are naming as *v1* to mark the first version of the API. It is common, that throughout the lifecycle of the applications that the API changes. Depending on the changes that are made, they may affect the compatibility with external systems. For instance, if a method is removed, the external system may enter in an error state because of that. Therefore, it is usually recommended to version APIs, and whenever a breaking change occurs, a new version of the API should be implemented.

- e) Set the **Send Default Values** property to *Yes*.

**NOTE:** This property ensures that default values in OutSystems (e.g. *False*) are always sent in the responses of the API. Depending on how the API is defined this may be relevant.

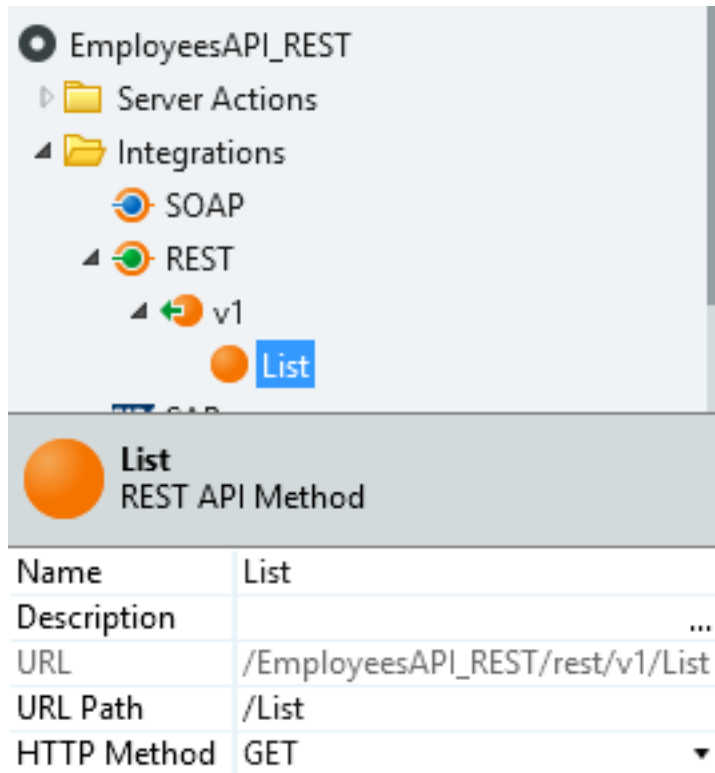
- 2) Create the *List* REST API method to return a list of employees. The output of the method should contain only a subset of attributes of the Employee entity.

- a) Right-click the *v1* REST API and select **Add REST API Method**





b) Set the name of the method to *List*



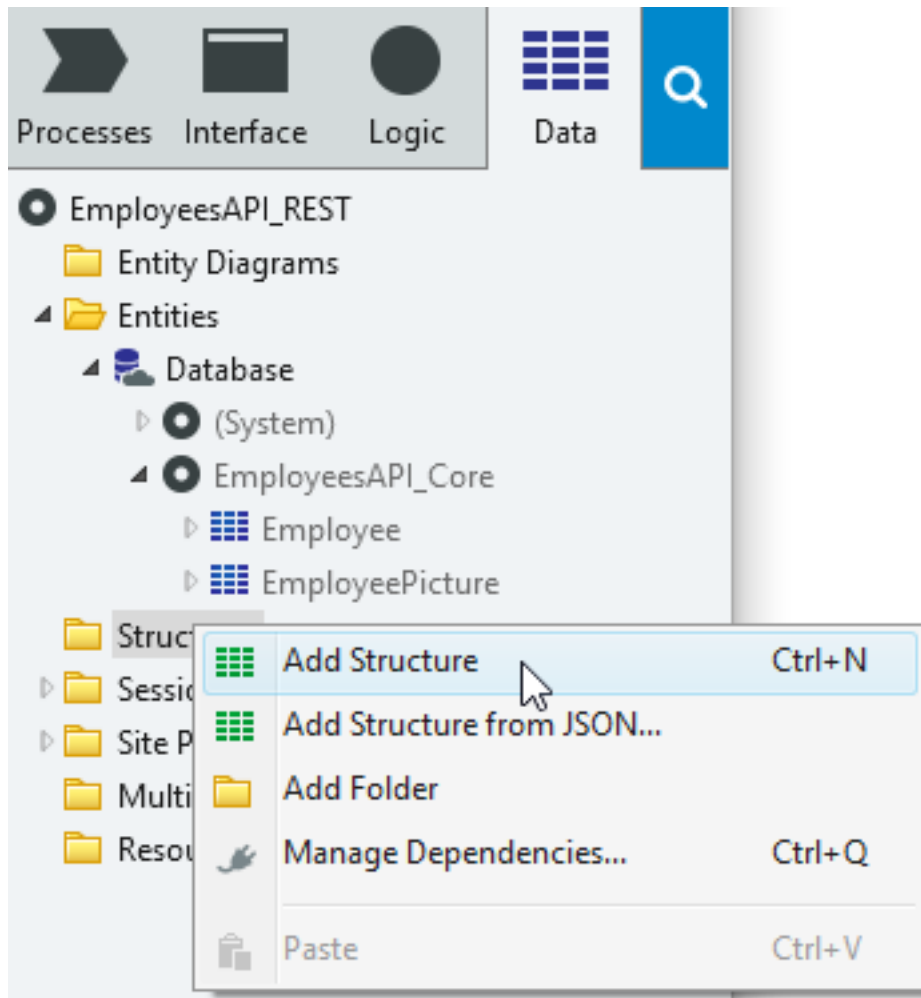
c) Change the **URL Path** to `/employees/`

**NOTE:** The URL paths can be customized to follow standards or best practices. In this case we are going to use the following convention:

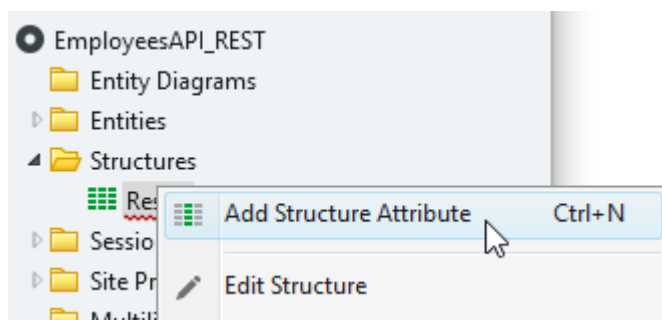
- GET `/employees/`
- GET `/employees/{Email}/`
- POST `/employees/`
- PUT `/employees/`
- DELETE `/employees/{Email}/`

- 4) Define the auxiliary data structures that will be used in the output result of the method. The List method should return a single record output containing the list employees and a result (success and error message).


- a) Switch to the Data tab, right click the Structures folder and select **Add Structure**




- b) Set the structure name to *Result*.
- c) Right-click the *Result* structure and select **Add Structure Attribute...**



- d) Set the attribute **Name** to Success and its **Data Type** to *Boolean*.

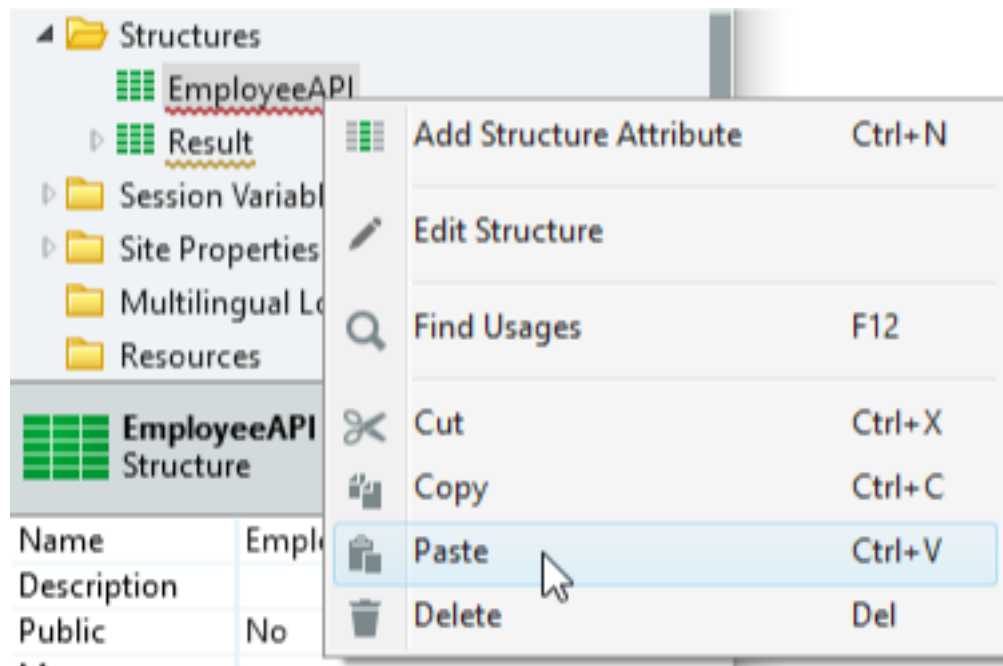
 <b>Success</b> Structure Attribute	
Name	Success
Description	...
Label	Success
Data Type	Boolean ▼
Is Mandatory	No ▼
Default Value	▼
JSON...	...

- e) Add another attribute named *ErrorMessage* with the **Data Type** set to *Text*.

 <b>ErrorMessage</b> Structure Attribute	
Name	ErrorMessage
Description	...
Label	Error Message
Data Type	Text ▼
Length	
Is Mandatory	No ▼
Default Value	...
JSON...	...

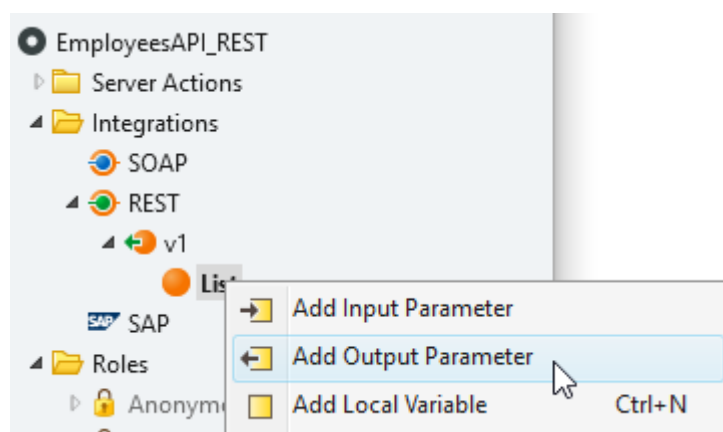
- f) Right-click the Structures folder and select **Add Structure**, then name it *EmployeeAPI*.
- g) Expand the Entities folder, and locate the *Employee* entity under the *EmployeesAPI\_Core*.
- h) Select the *Name*, *Email*, *Gender*, *City*, *State*, *Country* and *Occupation* attributes and copy them using the right click menu or CTRL+C.

- i) Right-click the *EmployeeAPI* structure created before and select *Paste*



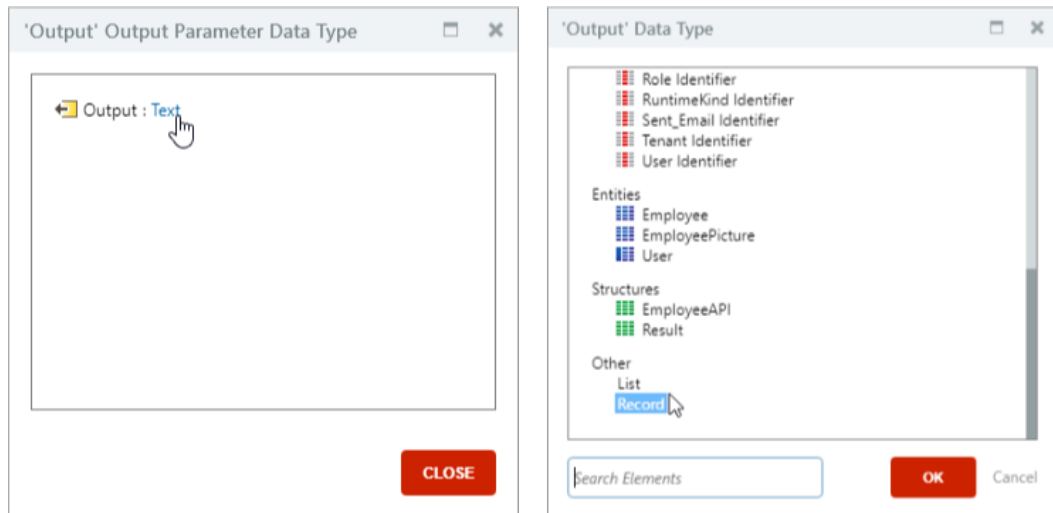
**NOTE:** This copy and paste operation allows to quickly create attributes in structures based on existing entities' attributes. It is common for APIs to expose a subset of the information when compared to the complete set of attributes that the data model may have.

- 5) Define the output of the *List* API method as a record containing the result (success and error message) and a list of employees.
- a) Switch to the Logic tab, then right-click the *List* method and select **Add Output Parameter**

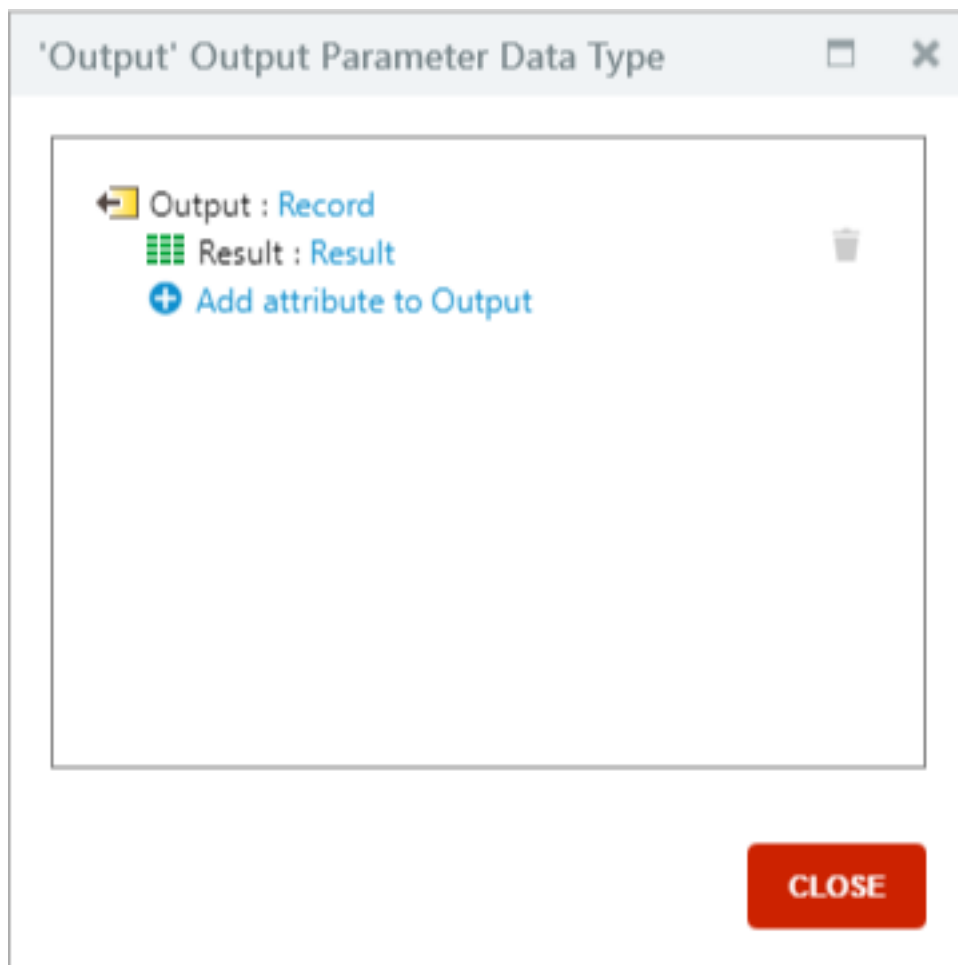


- b) Set the name of the output parameter to *Output*.
- c) Double-click the **Data Type** property to open the Data Type Editor.

d) Click the *Text* type, then select *Record*.

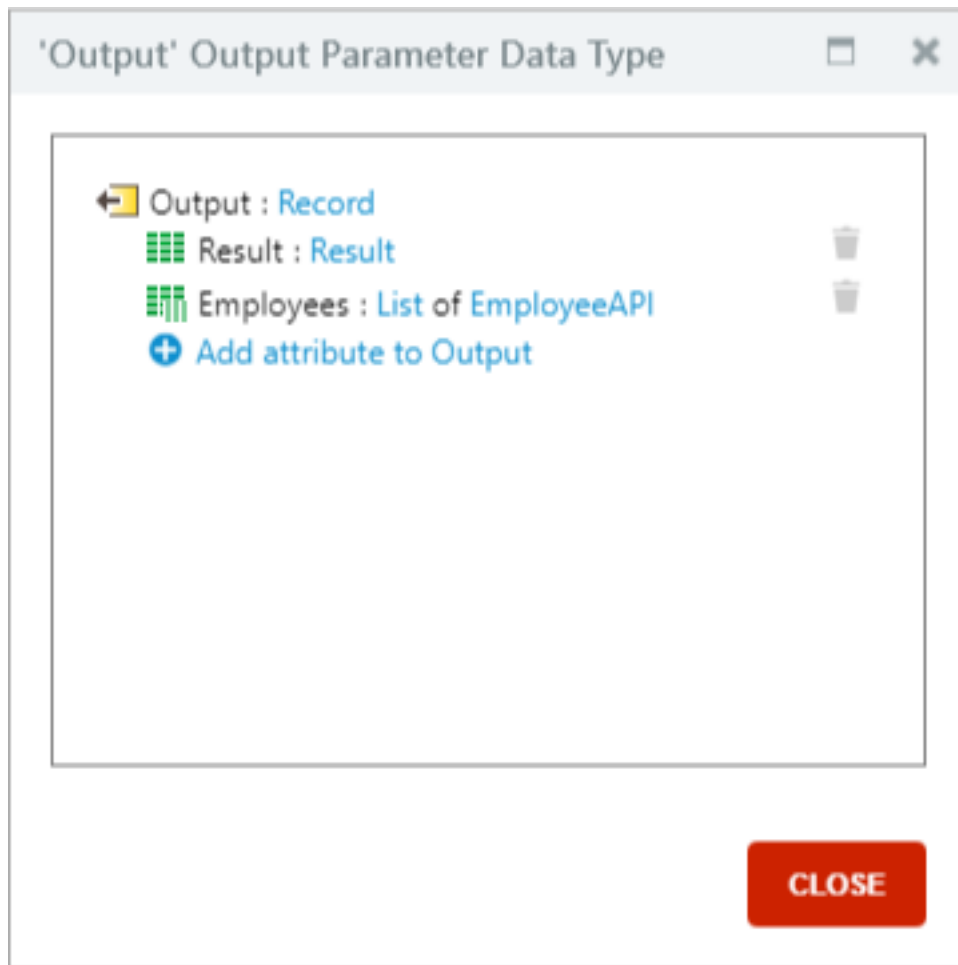


e) Change the name of the existing attribute to *Result* with type *Result*.



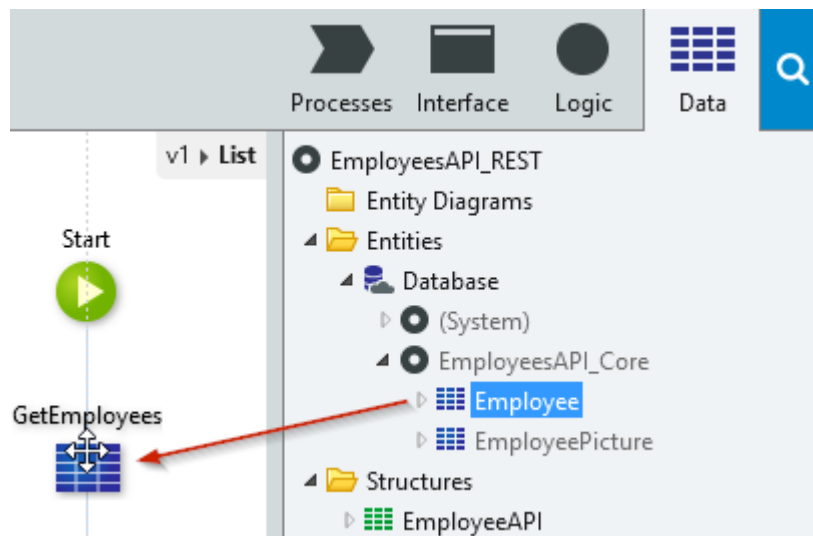
f) Add another attribute and name it *Employees*.

- g) Make sure the Data Type of the *Employees* attribute is set to *List of EmployeeAPI*



- h) Click **Close** to close the Data Type editor.
- 6) Define the Logic to retrieve the employees from the *Employee* entity and set the result of the *List* method.
- a) Double-click the *List* method to open its flow.

- b) From the Data tab, drag the **Employee** entity and drop it after the Start to create the *GetEmployees* aggregate



- c) Drag an **If** and drop it in the flow after the GetEmployees aggregate, then set its Condition to

```
GetEmployees.List.Empty
```

- d) Drag an **Assign** and drop it on the *False* branch connector of the If.

- e) In the Assign, define the following assignments

Assign	
Label	
Assignments	
Output.Result.Success	▼
= True	▼
Output.Result.ErrorMessage	▼
= ""	▼
Output.Employees	▼
= GetEmployees.List	▼
Mapping to EmployeeAPI	
Name	Employee.Name ▼
Email	Employee.Email ▼
Gender	Employee.Gender ▼
City	Employee.City ▼
State	Employee.State ▼
Country	Employee.Country ▼
Occupat...	Employee.Occupation ▼
Assignments	
Variable	▼
= Value	▼

**NOTE:** Since the output of the List method has a different data type (EmployeeAPI) from the Employee entity, the mapping is required. In this case it is straightforward, but some scenarios may require extra fine tuning.

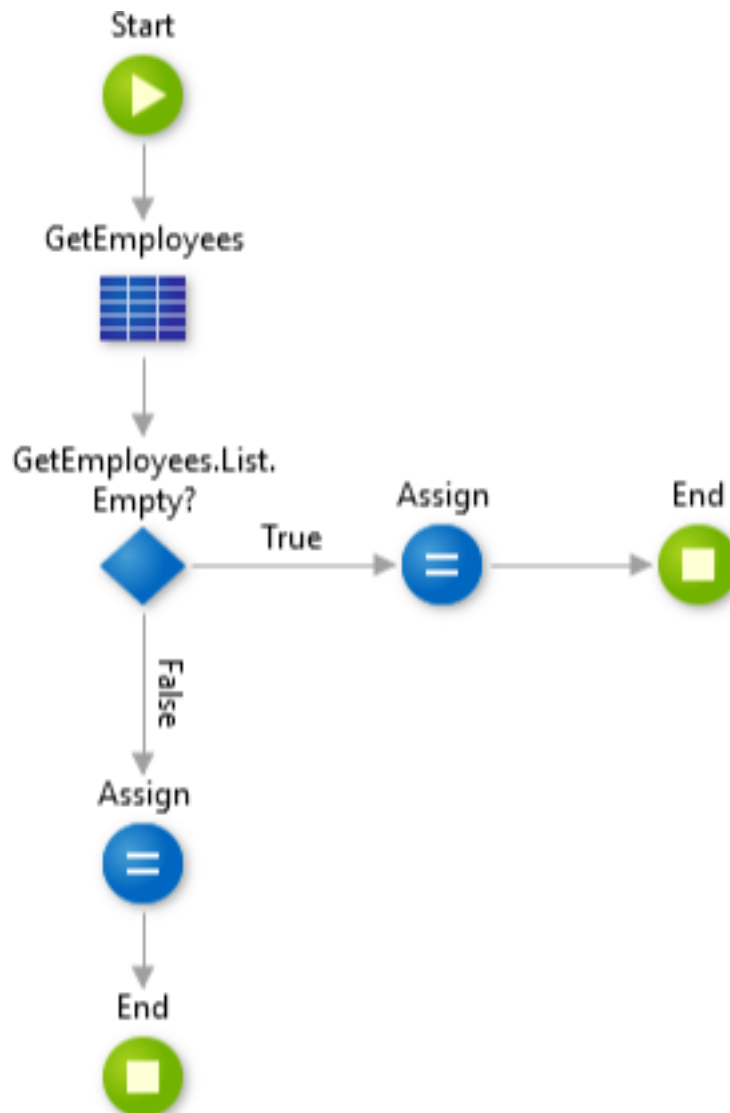
- f) Drag another **Assign**, and drop it to the right of the If, then create the True branch connector from the If to the new Assign.
- g) Define the following assignments

```
Output.Result.Success = False
Output.Result.ErrorMessage = "No Employees found."
```

- h) Drag an **End** and drop it to the right of the Assign, then connect both.



i) The flow should look like this

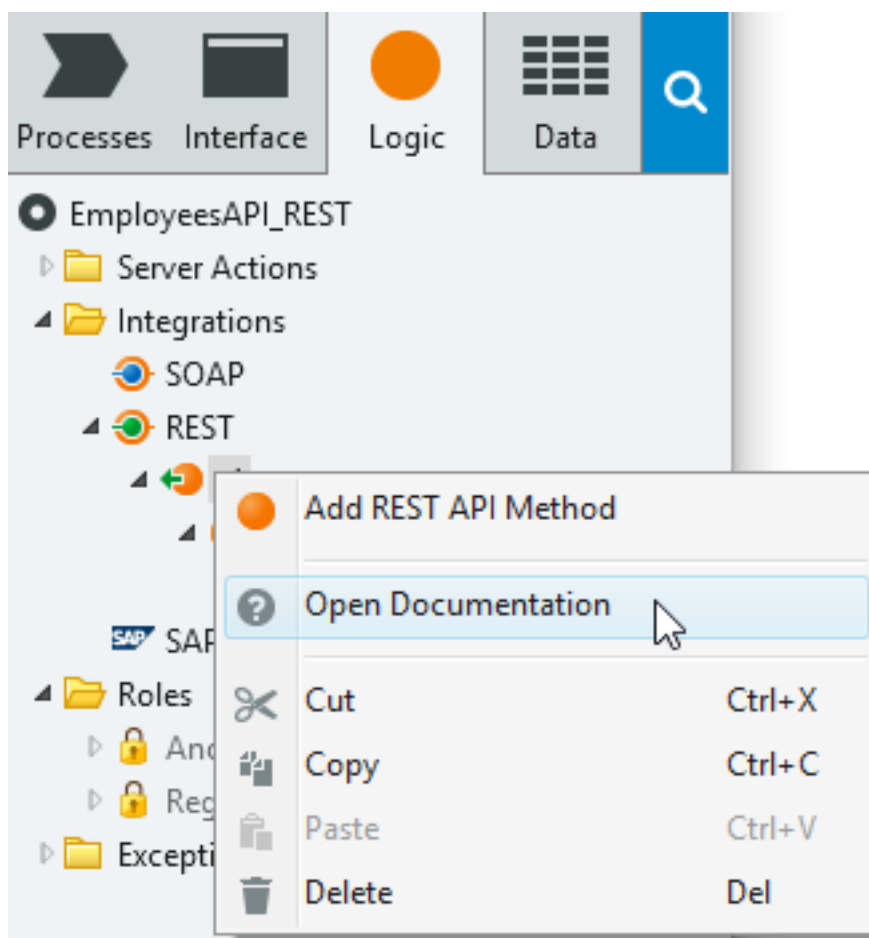


7) Publish the application using 1-Click Publish button and verify that the publish completed successfully in the 1-Click Publish tab.

- 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.

TrueChange™	Debugger	1-Click Publish
1	Uploading	Storing a new version into 'https://os11training.outsystems.net/ServiceCenter'.
2	Compiling	Generating and compiling optimized ASP.NET C# code and creating SQL scripts.
3	Deploying	Updating SQL Server database model and deploying the web application to IIS.
✓	Done	'EmployeesAPI_REST' is now available at 'https://os11training.outsystems.net/EmployeesAPI_REST'.

- c) In the Logic tab, right-click the exposed API, and select **Open Documentation**



**NOTE:** When modules containing exposed REST APIs are published, the OutSystems platform server automatically generates the Documentation page. This page contains essential information that can be provided to whoever will integrate with the defined APIs. Although we have not written Descriptions for the API, its methods and the outputs, defining them will have influence in the generated Documentation. Descriptions, HTTP Methods, URL Paths, and others are used when the Documentation is generated. The generated documentation is available at <https://<your-server-address>/<module-name>/rest/<api-name>/>

d) Expand the List method, and copy the URL

**v1**

[ API DEFINITION URL: [https://os11training.outsystems.net/EmployeesAPI\\_REST/rest/v1/swagger.json](https://os11training.outsystems.net/EmployeesAPI_REST/rest/v1/swagger.json) ]

GET /employees/

Request URL

`https://os11training.outsystems.net/EmployeesAPI_REST/rest/v1/employees/`

Response Messages

HTTP Status Code	Description	Content-Type	Data Type ( Model   Example )
200		application/json	<pre>{   "Employees": [     {       "Name": "string",       "Email": "string",       "Gender": "string",       "City": "string",       "State": "string",       "Country": "string",       "Occupation": "string"     }   ],   "": 1 }</pre>

e) Open a new tab in your browser and navigate to <https://reqbin.com/>

f) In the URL field paste the URL copied above

## Send HTTP Requests Online

Send custom HTTP requests to a server and check server responses

GET

g) Click the **Send** button. On the right-hand side of the page the sample response should appear

### Send HTTP Requests Online

Send custom HTTP requests to a server and check server responses

GET

Status: 200 (OK) Time: 48 ms Size: 8.7 kb

Content Headers Raw

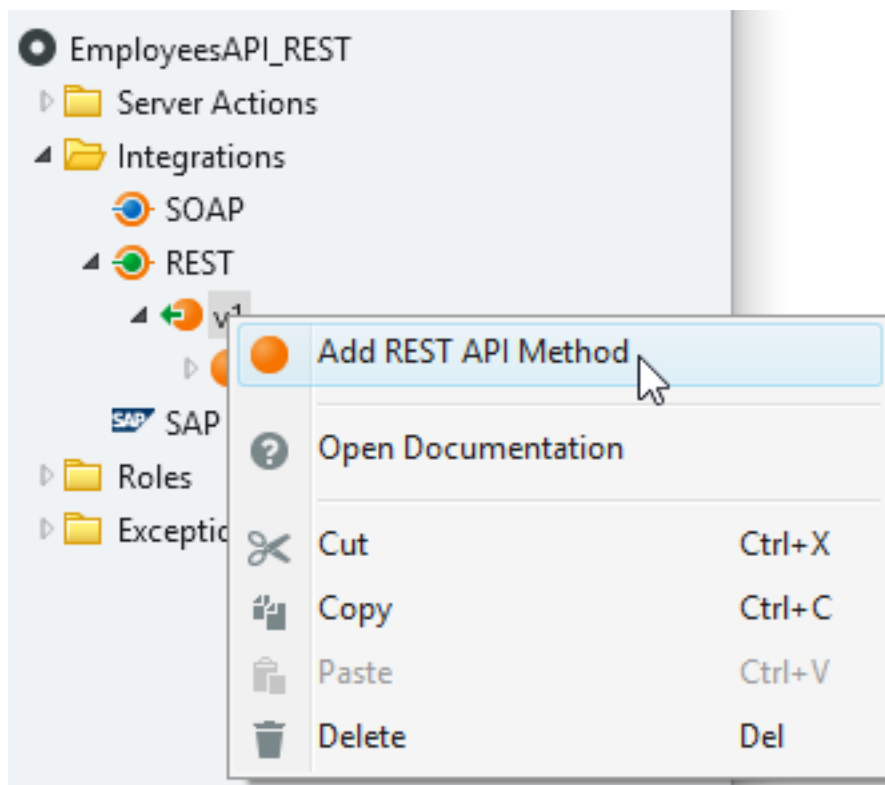
application/json; charset=utf-8

```
1 {
2   "Employees": [
3     {
4       "Name": "Amy Peters",
5       "Email": "Amy2Peters@teleworm.us",
6       "Gender": "female",
7       "City": "Wattsburg",
8       "State": "Pennsylvania",
9       "Country": "United States",
10      "Occupation": "Monetary economist"
11    }, {
12      "Name": "Bette Bauer",
13      "Email": "BetteBauer@jourrapide.com",
14      "Gender": "female",
15      "City": "Baywood Park",
16      "State": "California",
17      "Country": "United States",
18      "Occupation": "Loan closer"
19    }, {
20      "Name": "Brandie Lahey",
21      "Email": "BrandieLahey@fleckens.hu",
22      "Gender": "female",
23      "City": "Moultrie",
24      "State": "Georgia",
25      "Country": "United States",
26      "Occupation": "Nuclear technician"
27    }, {
28      "Name": "Carol Hightower",
29      "Email": "CarolCHightower@einet.com",
30    }
31  ],
32  "": 1
33 }
```

## Expose an Employee Information

In this section we will now work on a new API method. This new method will allow to retrieve the information of a single Employee based on a given Email address. Therefore, we will need to have one input parameter, and then based on that search for the employee information in the Database, and retrieve it.

- 1) Create a new API method named *GetEmployee*, and define the Email input parameter and an Output parameter containing a Result value and the Employee information.
  - a) In the Logic tab, right-click the v1 exposed API and select **Add REST API Method**

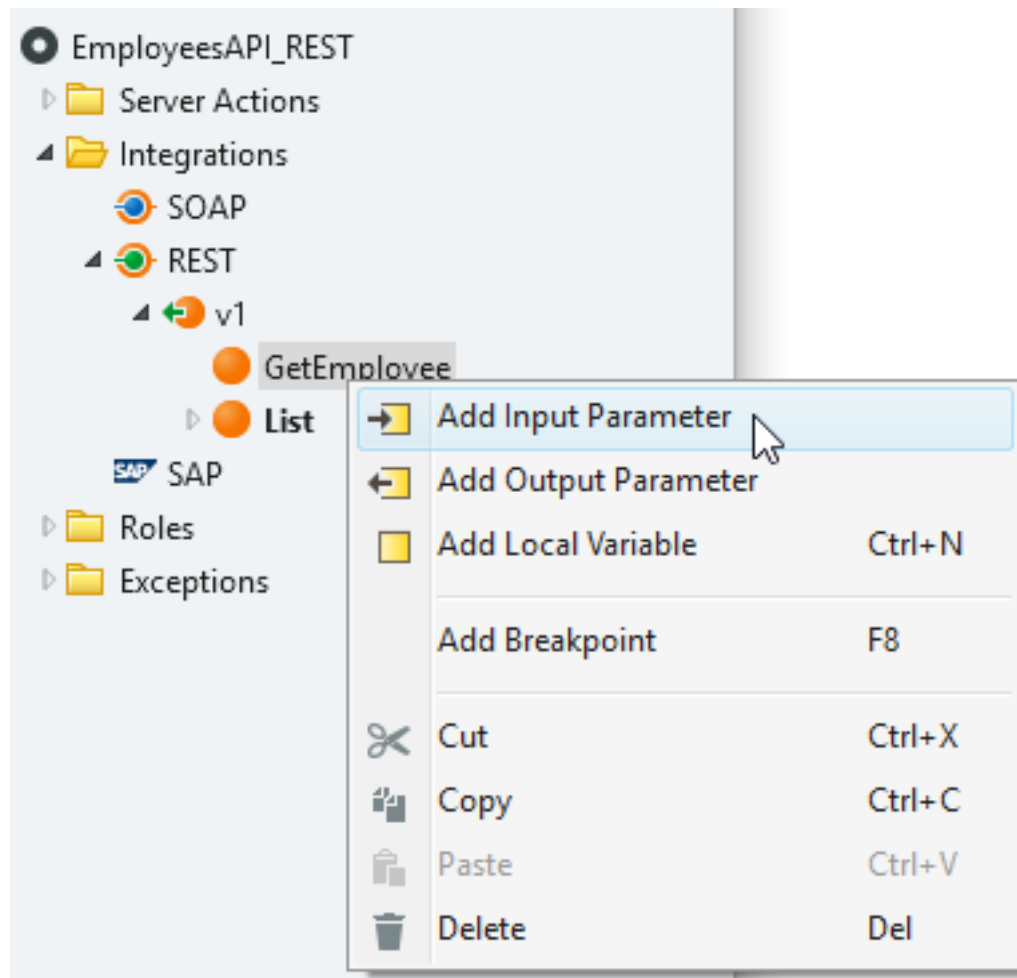


- b) Change the **Name** of the method to *GetEmployee*, then set the **URL Path** to

```
/employees/{Email}/
```

**NOTE:** At this point an error will be detected. This happens because we are specifying the Email input parameter, but haven't yet defined one at the method level.

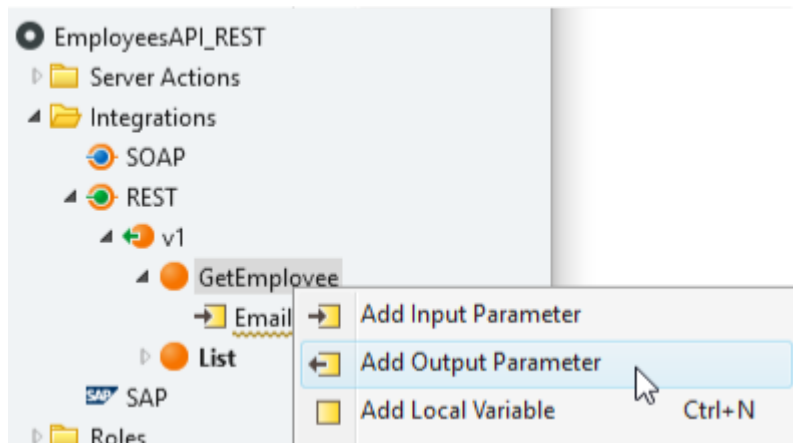
- c) Right-click the new method and select **Add Input Parameter**



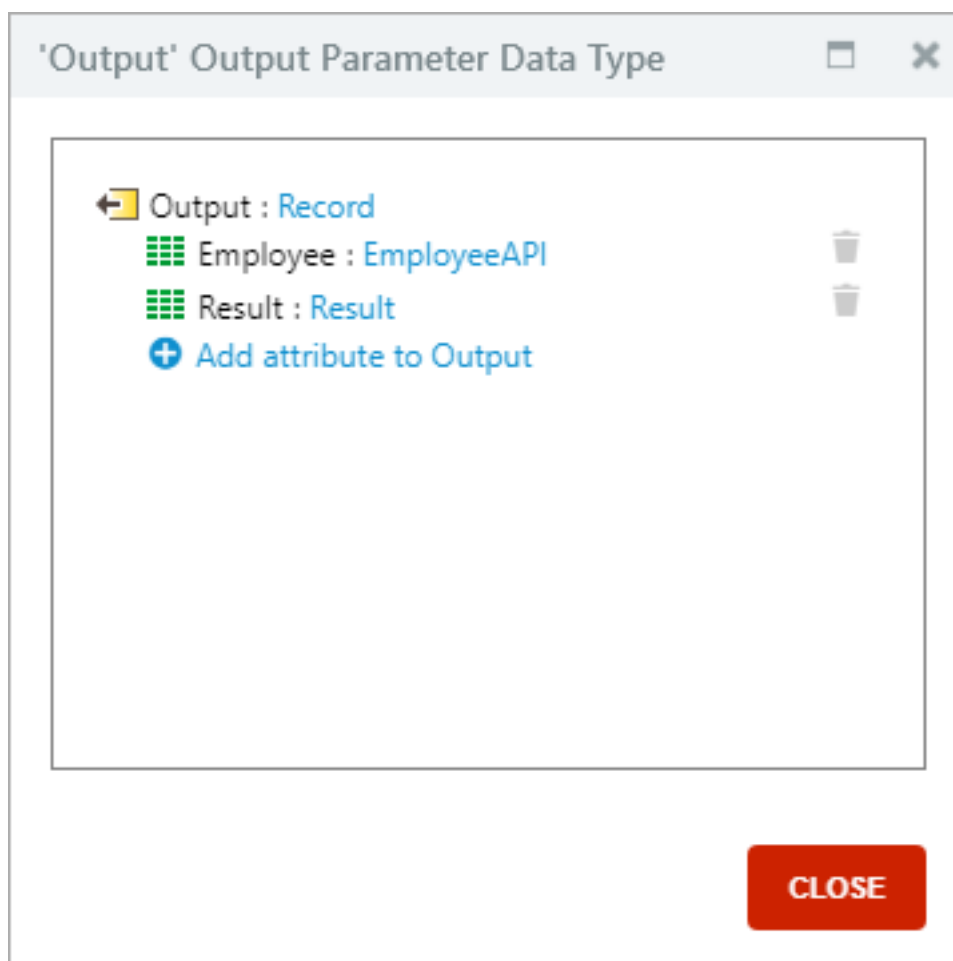
- d) Change the name of the input to *Email* and make sure its Data Type is set to *Email*

Email Input Parameter	
Name	Email
Description	...
Data Type	Email ▼
Is Mandatory	Yes ▼
Advanced	
Receive In	URL ▼
Default Value	
Name in Req...	

- e) Right-click the GetEmployee method and select **Add Output Parameter**



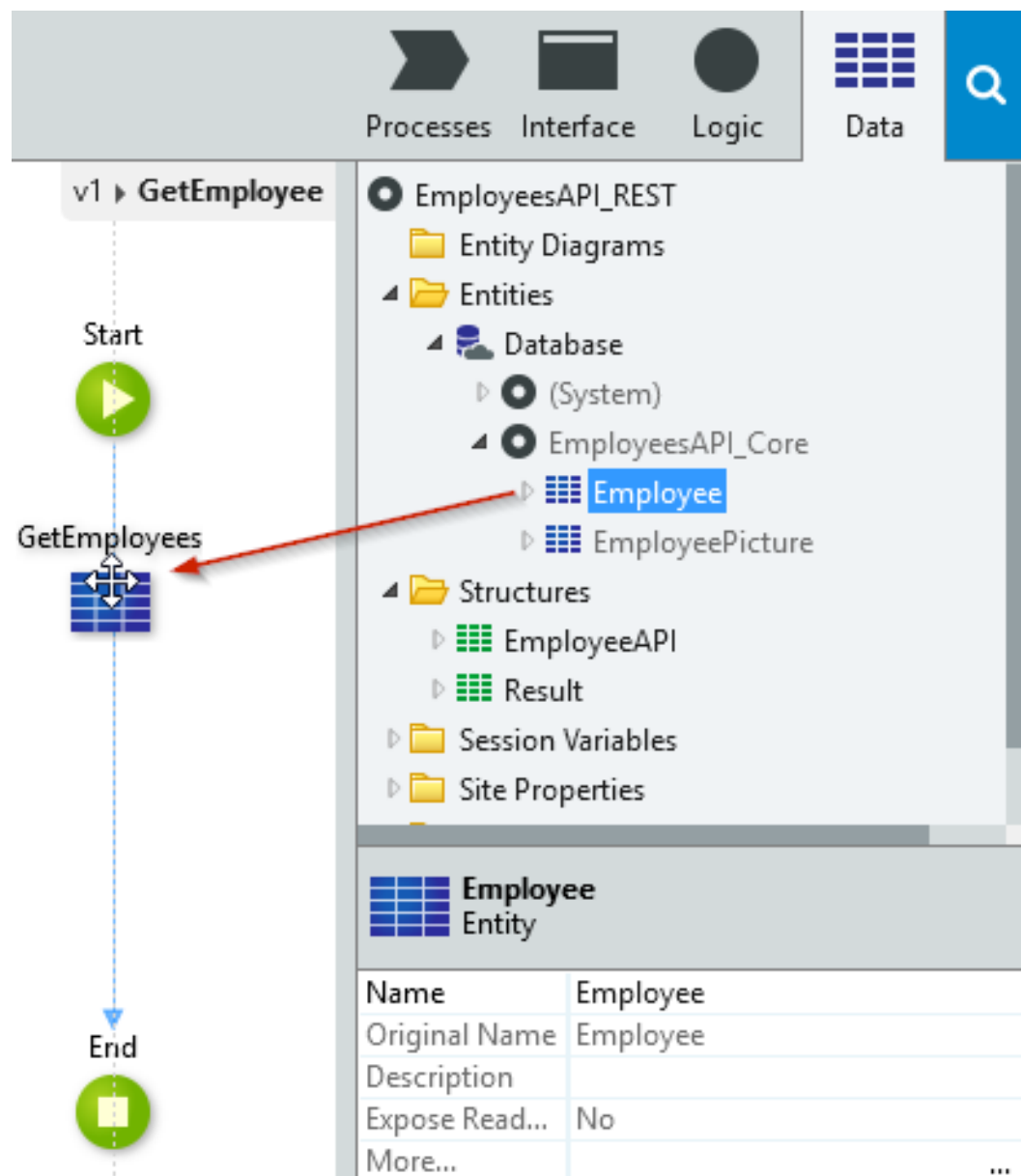
- f) Set the name of the output parameter to *Output*, then double-click the **Data Type** property and define the type as follows



**NOTE:** The types can be changed by clicking on them and selecting the desired type. On the Employee attribute make sure you select EmployeeAPI.

Remember that the information we are exposing via the API is a subset of the actual information we have stored in the Database.

- 2) Implement the *GetEmployee* API method to retrieve the Employee information from the database entity based on a given email.
  - a) Double-click the *GetEmployee* method to open its flow.
  - b) From the Data tab, drag the **Employee** entity and drop it in the flow to create the *GetEmployees* aggregate

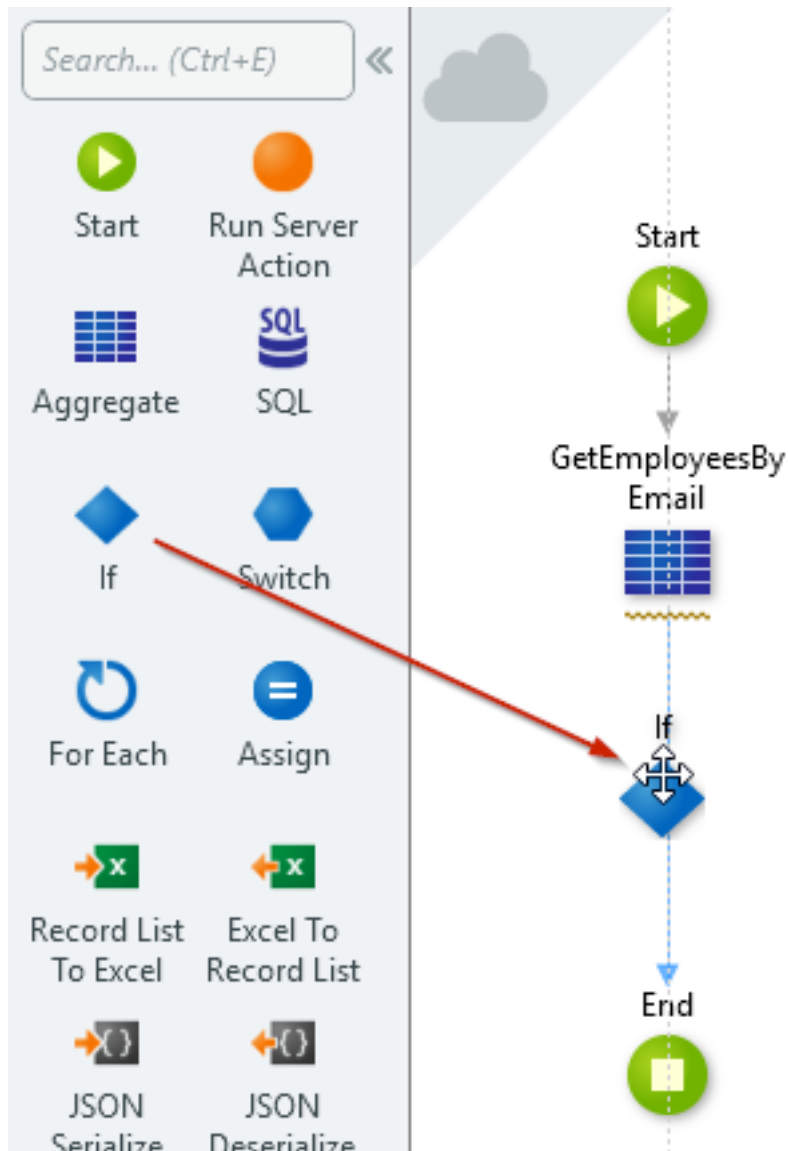


- c) Double-click the Aggregate to open it, then click on the Filters at the top to expand the Aggregate filters.

- d) Click the **+Add Filter**, and define its expression as

```
Employee.Email = Email
```

- e) Click **Done** to close the Expression Editor and return to the *GetEmployee* API method flow.
- f) Drag an **If** and drop it below the Aggregate



- g) Set the **Condition** of the If to

```
GetEmployeesByEmail.List.Empty
```

- h) Drag an **Assign** and drop it on the False branch connector, then define the following assignments

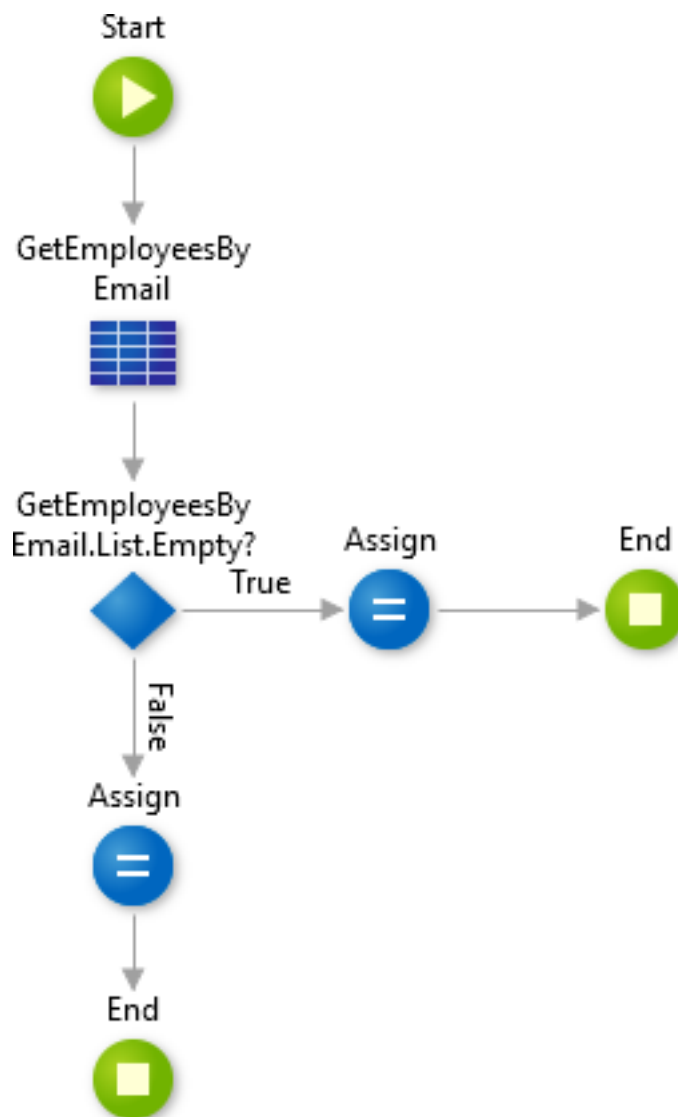


```
Output.Result.Success = True
Output.Employee = GetEmployeesByEmail.List.Current.Employee
```

- i) Drag another **Assign** and drop it to the right of the If, then create the *True* branch connector from the If to the new Assign.
- j) In the new **Assign** define the new assignments

```
Output.Result.Success = False
Output.Result.ErrorMessage = "Employee not found."
```

- k) Drag an **End** and drop it to the right of the Assign, then connect both.
- l) The flow should look like this



3) 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.

TrueChange™	Debugger	1-Click Publish
1 Uploading	Storing a new version into 'https://os11training.outsystems.net/ServiceCenter'.	
2 Compiling	Generating and compiling optimized ASP.NET C# code and creating SQL scripts.	
3 Deploying	Updating SQL Server database model and deploying the web application to IIS.	
✓ Done	'EmployeesAPI_REST' is now available at 'https://os11training.outsystems.net/EmployeesAPI_REST'.	

c) Open the Documentation page of the exposed REST API, then copy the URL of the *GetEmployee* method

GET /employees/{Email}/

Request URL

https://os11training.outsystems.net/EmployeesAPI\_REST/rest/v1/employees/{Email}/

Parameters

Parameter	Description	Parameter Type	Data Type ( Model Example )
Email		path	string

Response Messages

HTTP Status Code	Description	Content-Type	Data Type ( Model Example )
200		application/json	<pre>{  "Result": {    "Success": true,    "ErrorMessage": "string"  },  "Employee": {    "Name": "string",    "Email": "string",    "Gender": "string",    "City": "string",    "State": "string",    "Country": "string".  } }</pre>

d) Open a new tab in your browser and navigate to https://reqbin.com/

e) In the URL field paste the URL copied above, replacing the {Email} by one of the emails obtained in the previous section (e.g. BetteFBauer@jourrapide.com)

```
https://<your-server>/EmployeesAPI_REST/rest/v1/employees/{Email}/
```

f) Click the **Send** button.

g) The result text JSON should show something similar to

### Send HTTP Requests Online

Send custom HTTP requests to a server and check server responses

GET

https://os11training.outsystems.net/EmployeesAPI\_REST/rest/v1/emplc

Send

Status: 200 (OK) Time: 606 ms Size: 0.2 kb

Authorization

Headers

Content

☒ No Auth ☐ Bearer Token ☐ Basic Auth ☐ Custom

This request does not use any authorization.

About ReqBin

ReqBin is a free, online HTTP/REST/SOAP API client.  
With ReqBin you can send custom HTTP requests to your server and examine the server responses, test server performance and detect security problems by sending modified requests without authorization headers, cookies, etc.  
  
ReqBin is maintained by HTTP Debugger Team which provides a professional [Proxy-less HTTP sniffer](#) for developers to debug HTTP API calls to back-ends and between back-ends.  
  
**Free Tools by HTTP Debugger Team**  
[HttpReply](#) - simulate HTTP API server responses by replying with files from local disk.  
[NetThrottler](#) - slow down your network speed and test your website or desktop app on slow connections.  
[HostRemapper](#) - retarget HTTP requests from production server to your local machine.

Content

Headers

Raw

application/json; charset=utf-8


```
1 {
2   "Result": {
3     "Success": true,
4     "ErrorMessage": ""
5   },
6   "Employee": {
7     "Name": "Bette Bauer",
8     "Email": "BetteFBauer@jourrapide.com",
9     "Gender": "female",
10    "City": "Baywood Park",
11    "State": "California",
12    "Country": "United States",
13    "Occupation": "Loan closer"
14  }
15 }
```

## Expose the Employee Picture

In this section we will now work on a new API method. This new method will allow to retrieve a the picture of an Employee (Binary Data) based on a given Email address. Therefore, we will need to have one input parameter, and then based on that search for the employee picture in the Database, and retrieve it.

- 1) Create a new API method named `GetEmployeePicture`, and define the Email input parameter and an Output parameter containing a Result value and the Employee picture.
  - a) In the Logic tab, right click the v1 exposed API and select **Add REST API Method**.
  - b) Name the new method as `GetEmployeePicture` and set the URL Path to

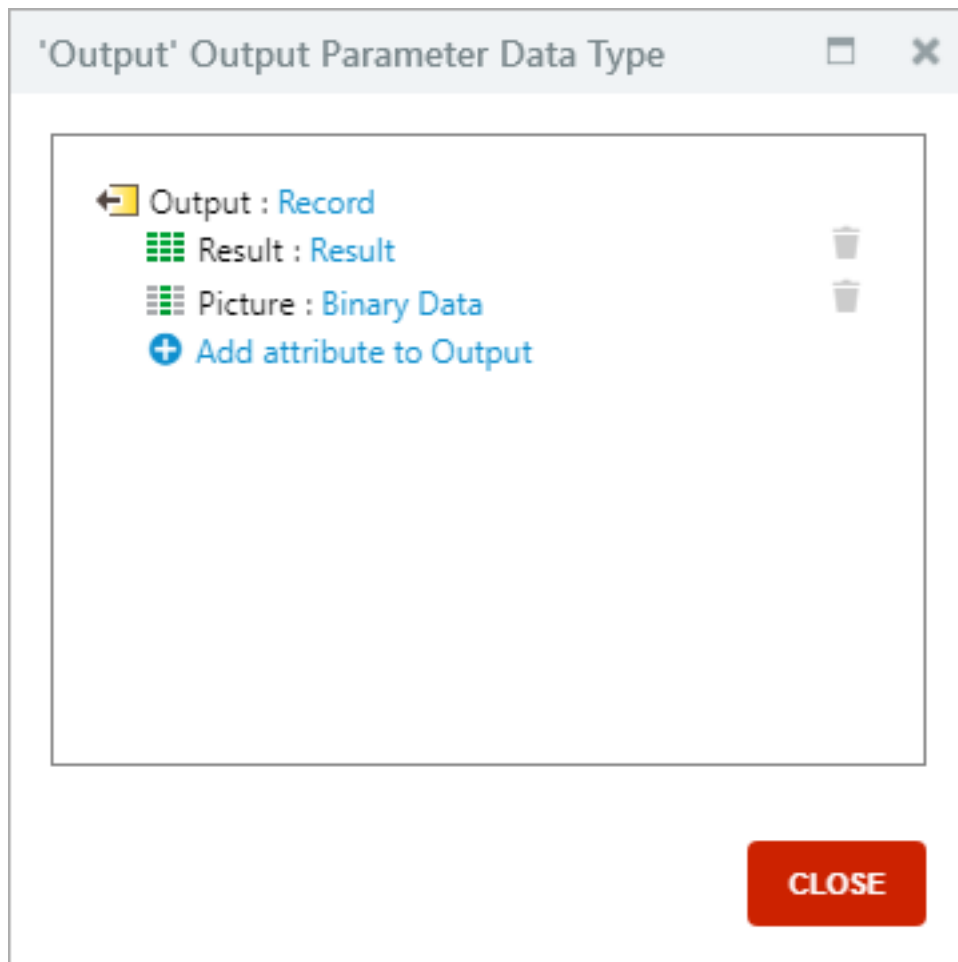
```
/employees/{Email}/picture/
```

 <b>GetEmployeePicture</b> REST API Method	
Name	GetEmployeePicture
Description	...
URL	/EmployeesAPI_REST/rest/v1/emp
URL Path	/employees/{Email}/picture/
HTTP Method	GET ▼

**NOTE:** At this point an error will be detected. This happens because we are specifying the Email input parameter, but haven't yet defined one at the method level.

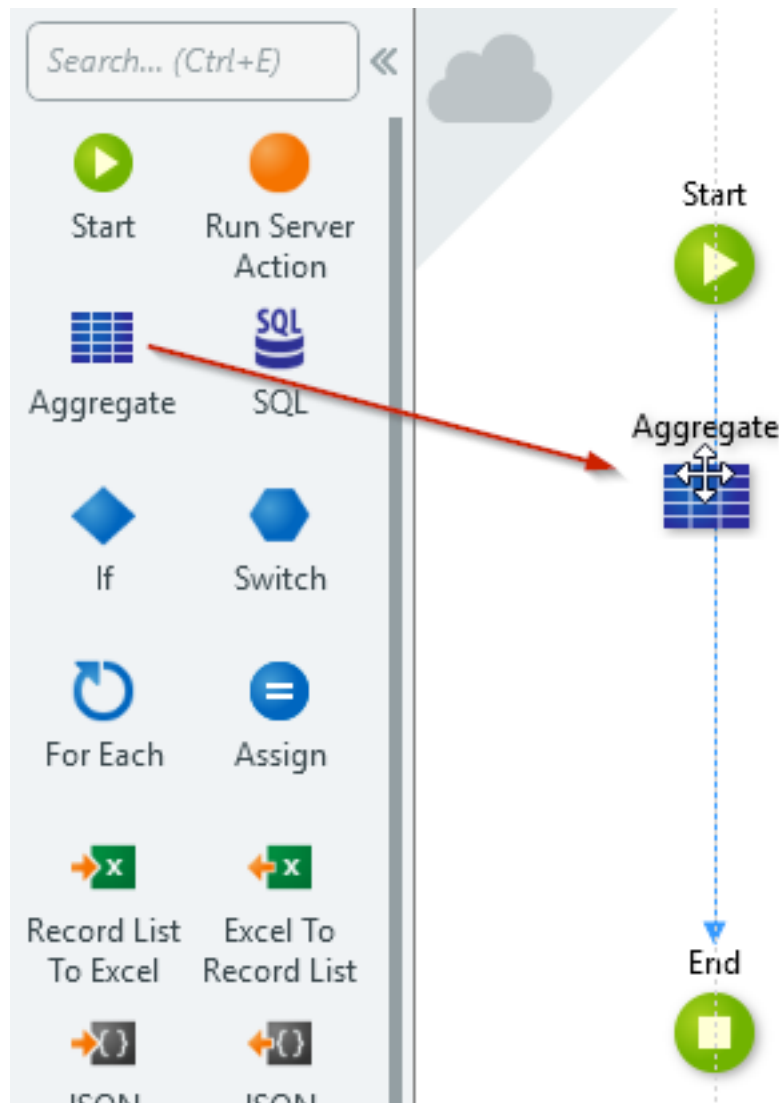
- c) Add an Input Parameter with the name *Email* and Data Type set to *Email*.

- d) Add an output parameter with the name *Output*, and the following type

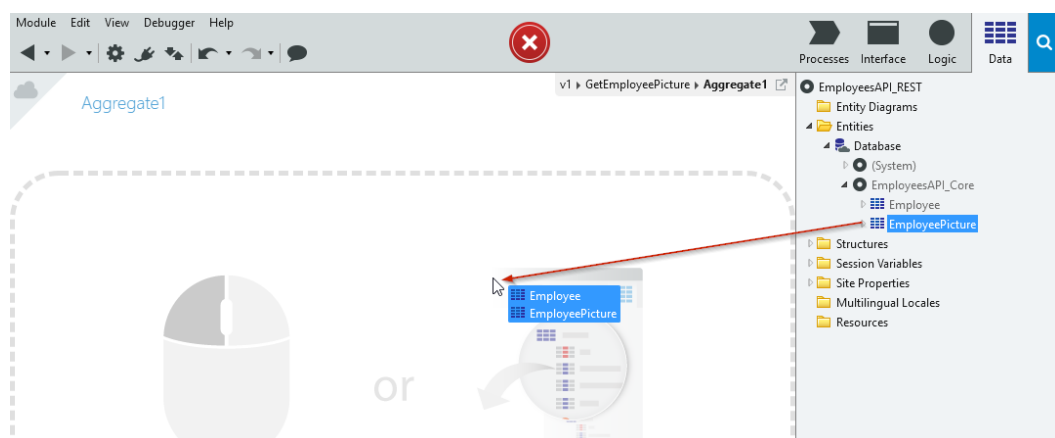


- 2) Define the flow of the *GetEmployeePicture* API method to retrieve the picture of the employee and return it in the output parameter.
- a) Double-click the *GetEmployeePicture* API method to open its flow.

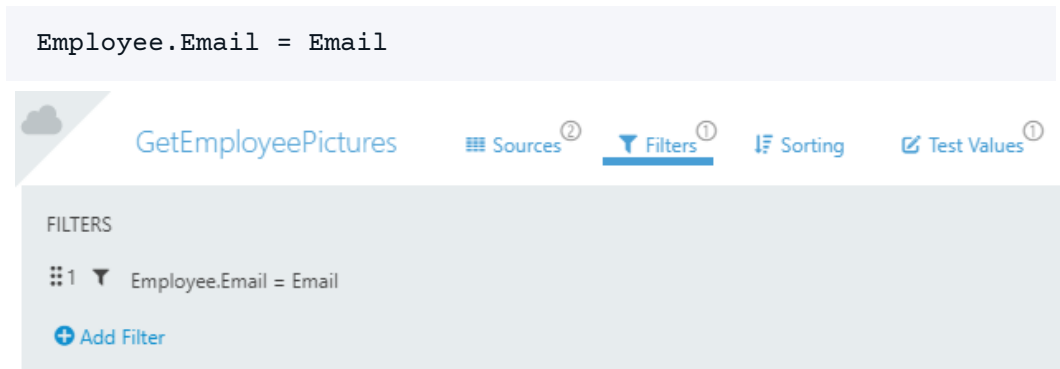
b) Drag an Aggregate to the flow, then open it



c) From the Data tab, drag the *EmployeePicture* entity, and drop it inside the Aggregate



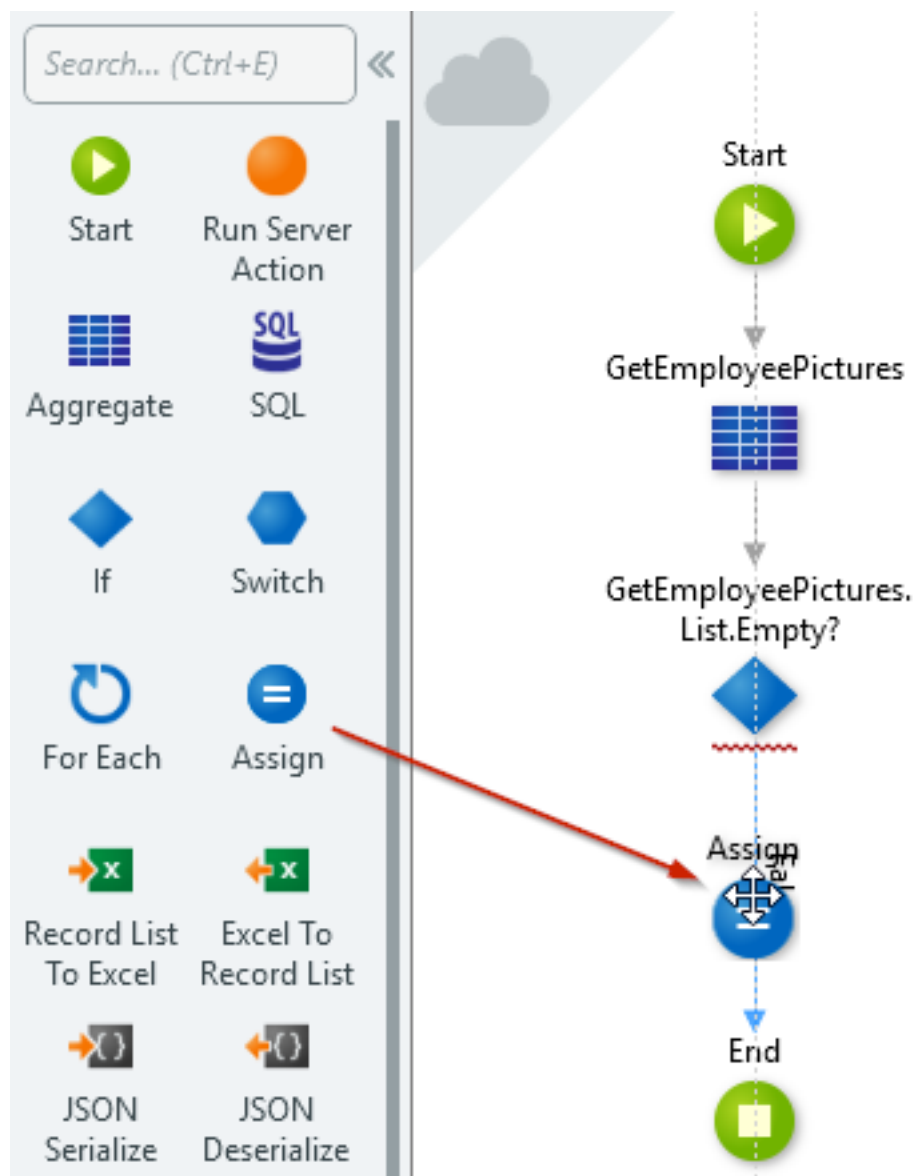
- d) Expand the Filters tab then click the **+Add Filter** and add the following filter



- e) Return to the *GetEmployeePicture* flow.
- f) Drag an **If** and drop it between the Aggregate and the End.
- g) Set the Condition property to

```
GetEmployeePictures.List.Empty
```

- h) Drag an **Assign** and drop it on the *False* branch of the If



- i) Define the following assignments in the newly created Assign

```
Output.Result.Success = True
Output.Picture =
GetEmployeePictures.List.Current.EmployeePicture.Picture
```

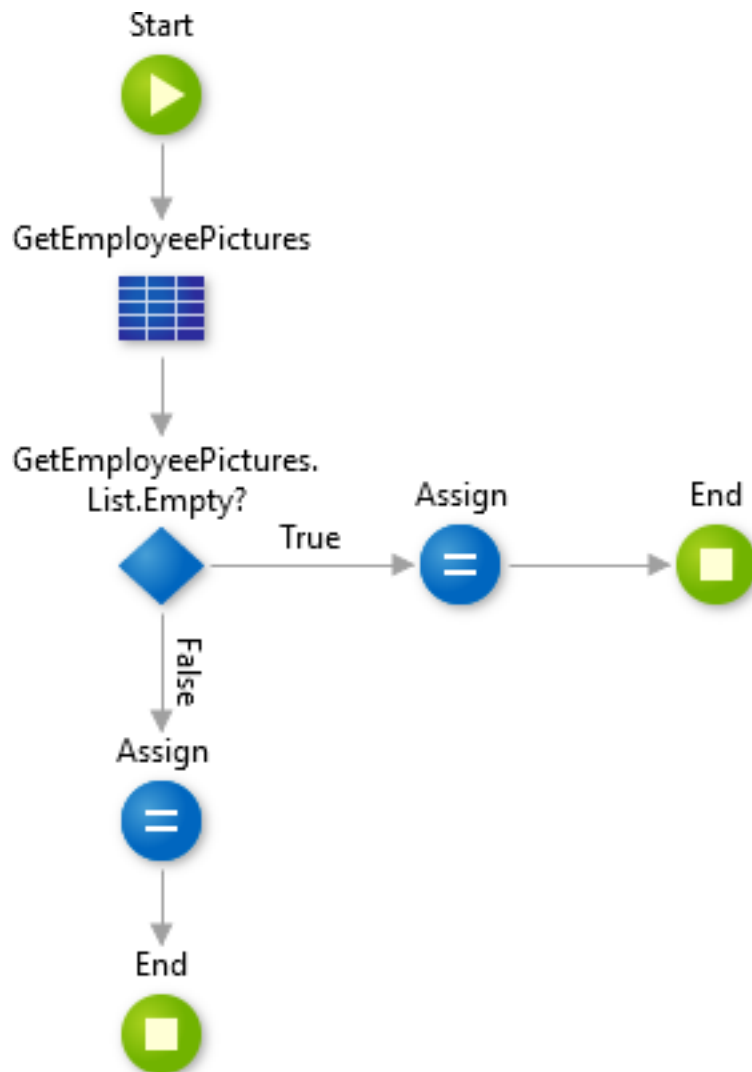
- j) Drag another **Assign** and drop it to the right of the If, then create the *True* branch connector from the If to the new Assign.
- k) In the Assign created in the previous step, define the following assignments

```
Output.Result.Success = False
Output.Result.ErrorMessage = "Employee picture not found."
```



l) Drag an **End** and drop it to the right of the Assigning, then connect both.

m) The flow should look like this



3) 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.

TrueChange™	Debugger	1-Click Publish
1 Uploading	Storing a new version into 'https://os11training.outsystems.net/ServiceCenter'.	
2 Compiling	Generating and compiling optimized ASP.NET C# code and creating SQL scripts.	
3 Deploying	Updating SQL Server database model and deploying the web application to IIS.	
✓ Done	'EmployeesAPI_REST' is now available at 'https://os11training.outsystems.net/EmployeesAPI_REST'.	

- c) Open the Documentation page of the exposed REST API, then copy the URL of the *GetEmployeePicture* method.
- d) Open a new tab in your browser and navigate to <https://reqbin.com/>
- e) In the URL field paste the URL copied above, replacing the *{Email}* by one of the emails obtained in the previous section (e.g. BetteFBauer@jourrapide.com)

```
https://<your-server>/EmployeesAPI_REST/rest/v1/employees/{Email}/picture/
```

- f) Click the **Send** button.

### Send HTTP Requests Online

Send custom HTTP requests to a server and check server responses

GET

https://os11training.outsystems.net/EmployeesAPI\_REST/rest/v1/emplc

Send

Status: 200 (OK) Time: 656 ms Size: 10.3 kb

Authorization

No Auth

Bearer Token

Basic Auth

Custom

This request does not use any authorization.

Content

application/json; charset=utf-8

```

1 {
2   "Picture": "1VB0RwKGG0AAAANSUjEgAAAGQAAABkCAHAAABPGVhAAABG8BTUEAAK/ITmMk6QAAAB10RV
3   "Result": {
4     "Success": true,
5     "ErrorMessage": ""
6   }
7 }

```

About ReqBin

ReqBin is a free, online HTTP/REST/SOAP API client.

With ReqBin you can send custom HTTP requests to your server and examine the server responses, test server performance and detect security problems by sending modified requests without authorization headers, cookies, etc.

ReqBin is maintained by HTTP Debugger Team which provides a professional [Proxy-less HTTP sniffer](#) for developers to debug HTTP API calls to back-ends and between back-ends.

Free Tools by HTTP Debugger Team

[HttpReply](#) - simulate HTTP API server responses by replying with files from local disk.

[NetThrottler](#) - slow down your network speed and test your website or desktop app on slow connections.

[HostRemapper](#) - retarget HTTP requests from production server to your local machine.

## End Lab

In this lab, we started the implementation of a REST API. This API allows external systems to access the data inside our application built in OutSystems, namely Employee information.

We started by implementing a List method that retrieves the information of existing employees. Then, moved on to the implementation of a GetEmployee method that given an Email input parameter allows to retrieve the information of a single employee. To complement this method, we created a third method named GetEmployeePicture. This time to retrieve a Binary Data (the employee profile picture).

Together, the three methods would allow an external system to display the information of the employees.