




Integration Studio Overview



What is Integration Studio?

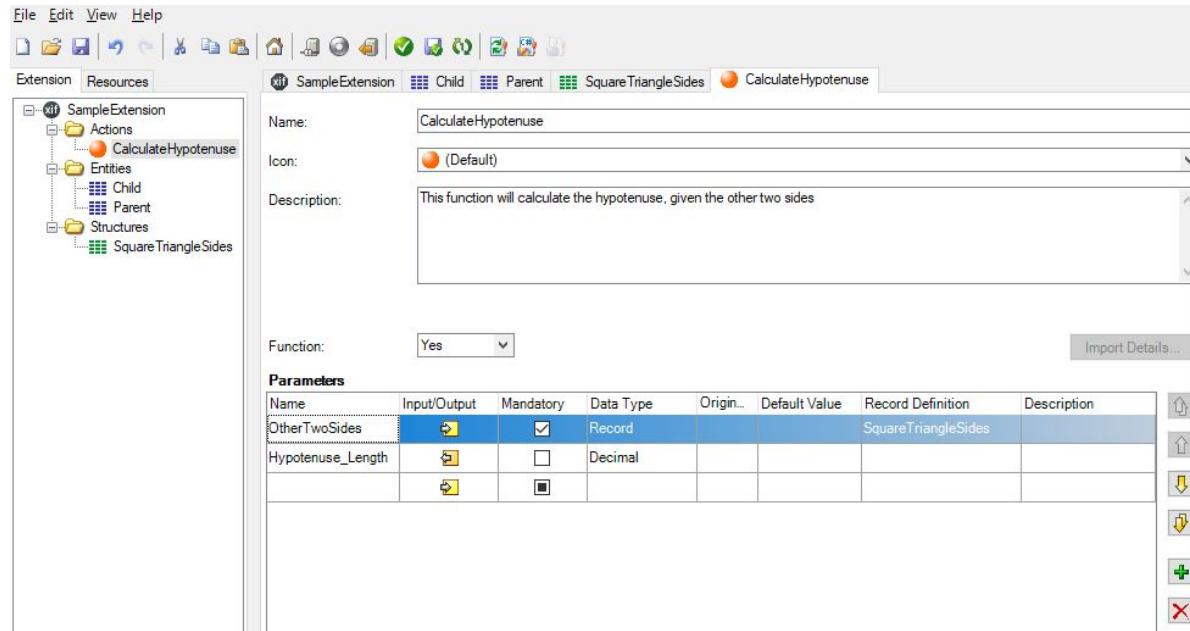
Integration Studio is part of the OutSystems Development suite. It enables the **extension** of the OutSystems platform through the creation of custom adapters to integrate with existing systems, databases and code

The developer defines and develops these adapters, that are published to the server as  **Extension modules**, ready to be **consumed** by Service Studio modules for use in web or mobile applications

What is Integration Studio? (cont.)

Most common scenarios in Extension modules

- Importing external database tables, *as Entities*
- Writing custom native code, *as Actions*



Importing External DB Tables

Declare and detail the database connection inside **Service Center**

Create an **Extension module** in **Integration Studio**, connect to the OutSystems server and import the desired tables over that connection

(Optionally) Tweak the introspected model e.g.

- Add descriptions to Entities and Attributes
- Fine-tune the Attributes' data types

These representations of external tables are published to server to be used in Service Studio Modules (no data is copied)

Importing External DB Tables (cont.)

The screenshot displays the 'Connect to External Table or View Wizard' in three overlapping windows. The background window is Step 1 of 6. The middle window is Step 4 of 6, titled 'Select the Tables and Views', showing a list of available tables and views on the left, including 'TypeMapping', 'TypeMappingBinary', 'TypeMappingNoBinaryData', 'TypeMappingTimestamp', and 'TypeMappingVarBinary'. The 'Tables' pane on the right shows a tree structure with 'DBImport' as the root, containing 'Actions', 'Entities', 'Child', 'Parent', 'TypeMappingImage', and 'Structures'. The 'Child' entity is selected. The foreground window shows the configuration for the 'Child' entity, with fields for Name, Table or View Name, Logical Database, Identifier, and Description. The 'Attributes' table lists 'cid' and 'pid' with their original names and types. A '1-Click Publish' dialog is open in the bottom right, showing a progress bar with steps 1 through 6, and a 'Publish Done' message indicating successful publication to p10training.outsystems.net.

Connect to External Table or View Wizard - Step 1 of 6

Connect to External Table or View Wizard - Step 4 of 6

Select the Tables and Views
Select one or more tables or views to use in the OutSystems Platform.

Available Tables and Views:

- TypeMapping
- TypeMappingBinary
- TypeMappingNoBinaryData
- TypeMappingTimestamp
- TypeMappingVarBinary

Add >> << Remove

Tables:

- Child
- Parent
- TypeMapping

Extension Resources

DBImport

- Actions
- Entities
- Child
- Parent
- TypeMappingImage
- Structures

Name: Child

Table or View Name: [[INTEGRATION]].[Integra].[dbo].[Child]

Logical Database:

Identifier: cid

Description: A child node

Default Value behavior: No conversion to/from D

Attributes

Name	Original Name	Original Type
cid	cid	Int
pid	pid	Int

1-Click Publish

- Verifying
- Updating Source Code
- Compiling
- Compiler Output
- Saving
- Uploading
- Publishing
- Publish Done

Publish Done
The Extension was successfully published to p10training.outsystems.net.

- Start wizard
- Pick the DB and tables
- Tweak imported model
- Publish to server

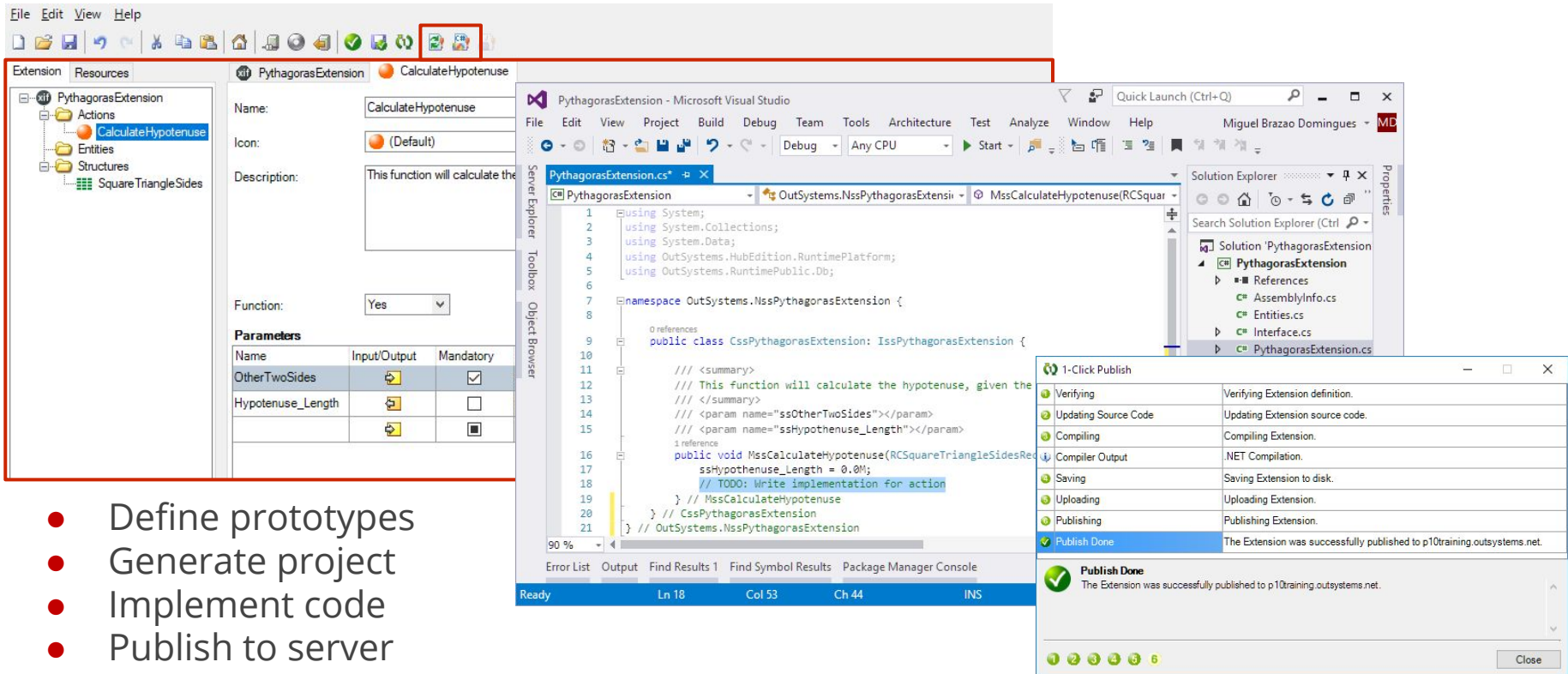
Integrating Custom Code

Create an **Extension module** in **Integration Studio** and define the Action and Structure prototypes it will export

Integration Studio generates the code stubs that developers then flesh out in Visual Studio for the .NET stack.

Compiled Extension code and any dependencies developers add to it are published to server to be used in Service Studio Modules

Integrating Custom Code (cont.)



The screenshot displays the Visual Studio IDE with the 'PythagorasExtension' project open. The 'CalculateHypotenuse' action is selected in the 'Actions' folder. The 'PythagorasExtension.cs' file is open, showing the implementation of the 'CalculateHypotenuse' function. The 'Publish Done' dialog box is visible, indicating that the extension was successfully published to p10training.outsystems.net.

PythagorasExtension - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help Miguel Brazao Domingues MD

PythagorasExtension.cs* x

```
1 using System;
2 using System.Collections;
3 using System.Data;
4 using OutSystems.HubEdition.RuntimePlatform;
5 using OutSystems.RuntimePublic.Db;
6
7 namespace OutSystems.NssPythagorasExtension {
8
9     0 references
10     public class CssPythagorasExtension: IssPythagorasExtension {
11
12         /// <summary>
13         /// This function will calculate the hypotenuse, given the
14         /// </summary>
15         /// <param name="ssOtherTwoSides"></param>
16         /// <param name="ssHypotenuse_Length"></param>
17
18         1 reference
19         public void MssCalculateHypotenuse(RCSquareTriangleSidesRec
20             ssHypotenuse_Length = 0.0M;
21             // TODO: Write implementation for action
22         } // MssCalculateHypotenuse
23     } // CssPythagorasExtension
24 } // OutSystems.NssPythagorasExtension
```

1-Click Publish

Step	Task
1	Verifying
2	Updating Source Code
3	Compiling
4	Compiler Output
5	Saving
6	Uploading
7	Publishing
8	Publish Done

Publish Done

The Extension was successfully published to p10training.outsystems.net.

Close

- Define prototypes
- Generate project
- Implement code
- Publish to server



Integration Studio Overview

Thank You!