**How to add new function to LS Omni Service**

This sample describes how to add new function to Omni Service that can be called from a Mobile Loyalty App or added to the eCommerce API Interface.

The code for the Omni Service and Mobile Apps is stored in separate Git repositories.

The code can be joined together if you are working with Loyalty App, as the code in the Common folder is same in both Solutions. For eCommerce, Loyalty App code is not needed.

Open *Server\WcfService\LSOmniService.sln*

Set Service Project as StartUp Project

Set correct connection data into *Service\AppSettings.config*

You can get this data from an already installed and running Omni Service.

**Omni Service uses these interfaces:**

|  |  |  |
| --- | --- | --- |
| **Json Interface** | **SOAP Interface** | **Usage** |
| IBOJson | IBOService | LS Central to communicate back to Omni Service |
| IUCJson | IUCService | Ecommerce and Loyalty API Interface |

**The main code blocks are:**

|  |  |
| --- | --- |
| Commons\Domain.DataModel.Base | Common Data objects used all apps and Web sites and common code used by the Common Data Object |
| Commons\Domain.DataModel.Loyalty | Loyalty Data objects |
| DataAccess\BOConnection.CentrAL | LS Central 15 and later Data communication |
| DataAccess\BOConnection.NavSQL | LS Central and NAV 14 and older Data communication |
| DataAccess\BOConnection.NavWS | LS Central in Cloud Data communication |
| DataAccess\Data.SQLServer | Local Omni Database communication |
| DataAccess\Interfaces | Interface files for DataAccess projects |
| BusinessLayer | Business Logic that takes care of calling right DataAccess code. |
| Service | Omni Service Interface code |

The most common code path is following:

The interface calls the Business Layer, that then take decision on where the code is located and calls the appropriated DataAccess code that then get the data from its source.

**Sample MyCustomFunction**

In the code, there is already a sample of a new function, which both Web Sites and Loyalty app can use. It is best to start from the end of the code path when adding a new function.

Depending on what NAV or BC Central platform is used, then your code should be placed in the appropriated DataAccess Project, see table above for which project to use.

This sample has code for both LS Central 14 and 15.

The main database processing code for LS Central 14 is done in this place

*DataAccess\BOConnection.NavSQL\Dal\MyCustomRepository.cs*

public string GetMyData(string data)

{

string returndata = string.Empty;

using (SqlConnection connection = new SqlConnection(connectionString))

{

using (SqlCommand command = connection.CreateCommand())

{

connection.Open();

// Write your database selection code here

connection.Close();

}

}

return returndata;

}

The code that takes care of deciding if should call web service or go directly into BC Database is done here. In this sample we will do direct Database call.

*DataAccess\BOConnection.NavSQL\NavCustom.cs*

public virtual string MyCustomFunction(string data)

{

MyCustomRepository rep = new MyCustomRepository(config, NAVVersion);

return rep.GetMyData(data);

}

For LS Central 15, you do the same as above, except it is done in CentrAL project.

*DataAccess\BOConnection.CentrAL*

We need to add new interface for our function, which defines our function for both LS Central 14 and 15

*Interfaces\BOConnection\ICustomBO.cs*

string MyCustomFunction(string data);

Business Layer will open up correct BO object where our new code is and call it

*BusinessLayer\CustomBLL.cs*

public virtual string MyCustomFunction(string data)

{

logger.Debug(config.LSKey.Key, "Debug data: " + data);

return BOCustom.MyCustomFunction(data);

}

All Omni Interfaces call this code, which then calls the function in Business Layer.

*Service\Common\LSOmniBaseCustom.cs*

public virtual string MyCustomFunction(string data)

{

CustomBLL myBLL = new CustomBLL();

return myBLL.MyCustomFunction(data);

}

Public interfaces

*Interface\IUCService.cs*

[OperationContract]

string MyCustomFunction(string data);

*Interface\IUCJson.cs*

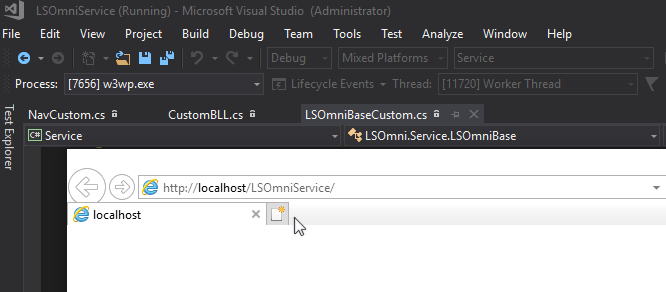
[OperationContract]

[WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Wrapped, ResponseFormat = WebMessageFormat.Json)]

string MyCustomFunction(string data);

**Testing MyCustomFunction**

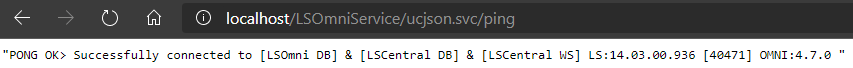
Run the solution and Internet Explorer will start up and wait like this:



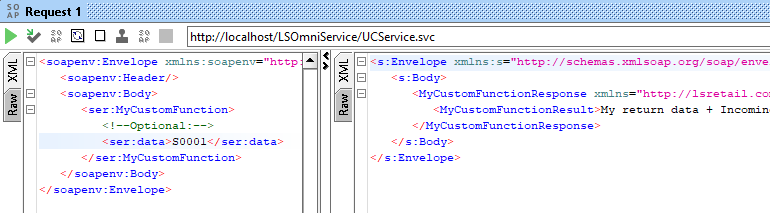
To test if the connection is working, type this into a browser

<http://localhost/LSOmniService/ucjson.svc/Ping>

If all is ok then you get this answer:



SoapUI is a free tool that can be used to test SOAP (service) interface, there we can find our new function and put in some data, run it and get the result.



**Call MyCustomFunction from the App Code**

To add this to the Json code so Loyalty App can use the function, open up the Loyalty solution.

*Loyalty\Android\LSRetail.Mobile.Loyalty.Android.sln*

**App Code blocks are:**

Loyalty App code uses same Core as Omni Service.

Commons Data Objects are located under Domain Folder.

Infrastructure folder includes Code to talk to the Omni Service Json interface and the Local App Database.

Presentation.Android has the main App code.

This document will not go into detail on how to do the App coding, just where you would add the new function that we created in Omni Service and how to call it.

The new functions goes in to Services library, which has Interface and Service code that app, uses to call the new function. First, we add the function to the Interface

*Common\Core\Loyalty\Domain.Services.Loyalty\Custom\ICustomRepository.cs*

string MyCustomFunction(string data);

Service code has two versions of the function, one for direct call and other to call in Async mode, which the App should usually use when calling the function.

*Common\Core\Loyalty\Domain.Services.Loyalty\Custom\CustomService.cs*

public string MyCustomFunction(string data)

{

return iCustomRepository.MyCustomFunction(data);

}

public async Task<string> MyCustomFunctionAsync(string data)

{

return await Task.Run(() => MyCustomFunction(data));

}

Infrastructure then does the actual call to Omni Service Json Interface.

*Common\Core\Loyalty\Infrastructure.Data.Omniservice.Loyalty\Custom\CustomRepository.cs*

public string MyCustomFunction(string data)

{

string methodName = "MyCustomFunction";

var jObject = new { data = data };

return base.PostData<string>(jObject, methodName);

}

In the Presentation code, you would call the new function like this

private CustomService customService;

public async Task<string> MyCustomFunction(string data)

{

BeginWsCall();

return await customService.MyCustomFunctionAsync(data);

}