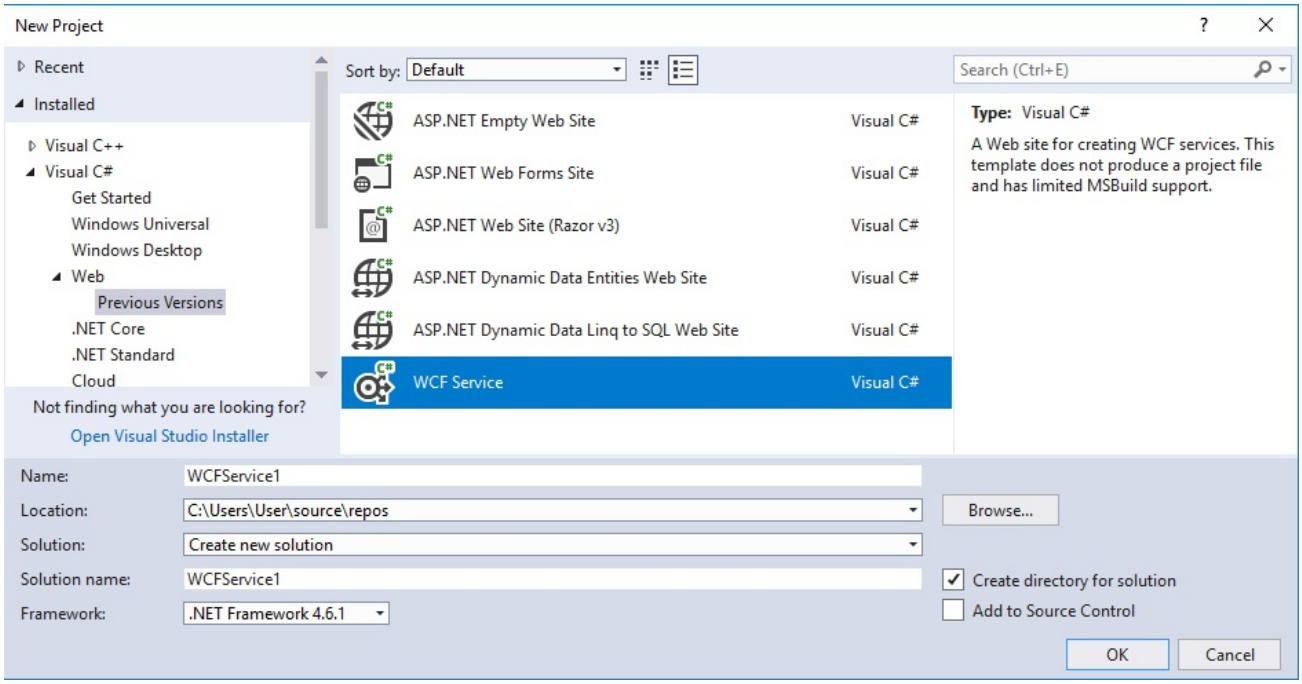
Build your First WCF Service

This is a small guide with excercises in order to learn a little bit more about WCF Services

Create a new WCF Service

- Open Visual Studio
- Go to New Project
- C#
- Previous Version
- **WCF Service** (*Update the location path if needed*)

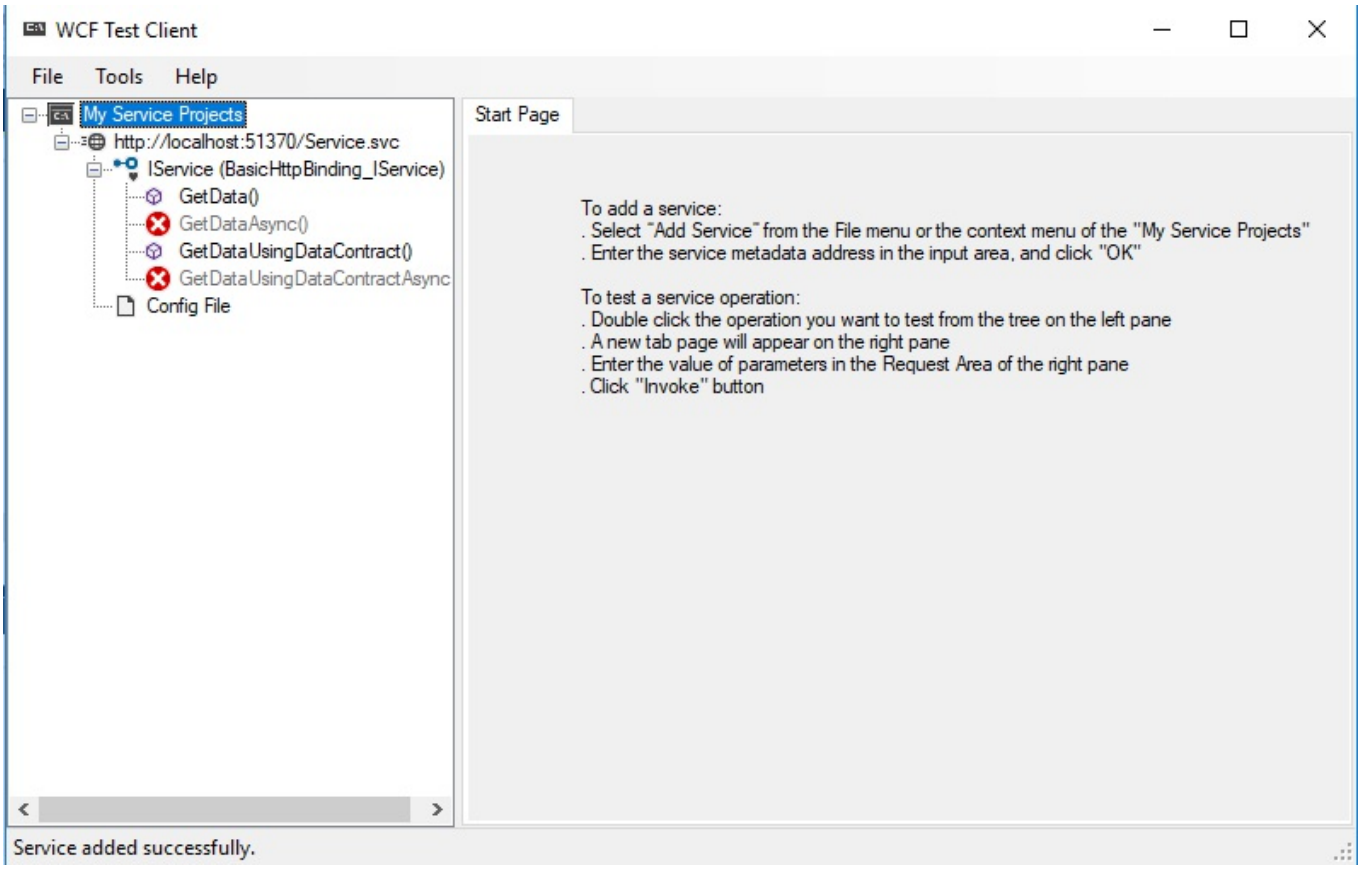


Once the service is created, you will be located on Service.cs, **hit F5 to run the Service**

If a popup appers saying: *The page cannot be run on debug mode because is not enable*, **select the option Modify the Web.config**

> *Visual Studio incorporate a tool to execute/test the WCF Services (internal client tool) that will allow us to execute all the service call as needed (for now)*

Now you can click the operations available on the service

> *Notes: Since we don't have any asynch method develop at this time some methods will be displayed as unavailable*
>
> 
>

To use the Tool: Input a value on the value field (Request Container) and click Invoke button.

On the Response you will see on the Value field "You entered: #" the number of your choice.

- Now close the Window to stop the service.

- Review the GetData method in the `Service.cs`, is there something missing regarding the past documentation?

Let's try out something, create a new method in the `Service.cs` with whatever signature you want and make it return a `string` value.

- Run the service again (F5)
- Did you see the operation displayed? No right?
- If you don't see the operation is because we are missing some required decorator to make our service discoverable (exposed in the service)
- Close the Window

Now go to the `IService.cs`

> *This interface contains the signature of the methods that we want to expose in the service, the reason of an Interface is because is a best practice, but it is not something required all the time.*

```
[ServiceContract]
public interface IService
{
    [OperationContract]
    string GetData(int value);

    [OperationContract]
    CompositeType GetDataUsingDataContract(CompositeType comp
osite);
```

```
}
```

Add the whatever signature of the method that you created in the previous step

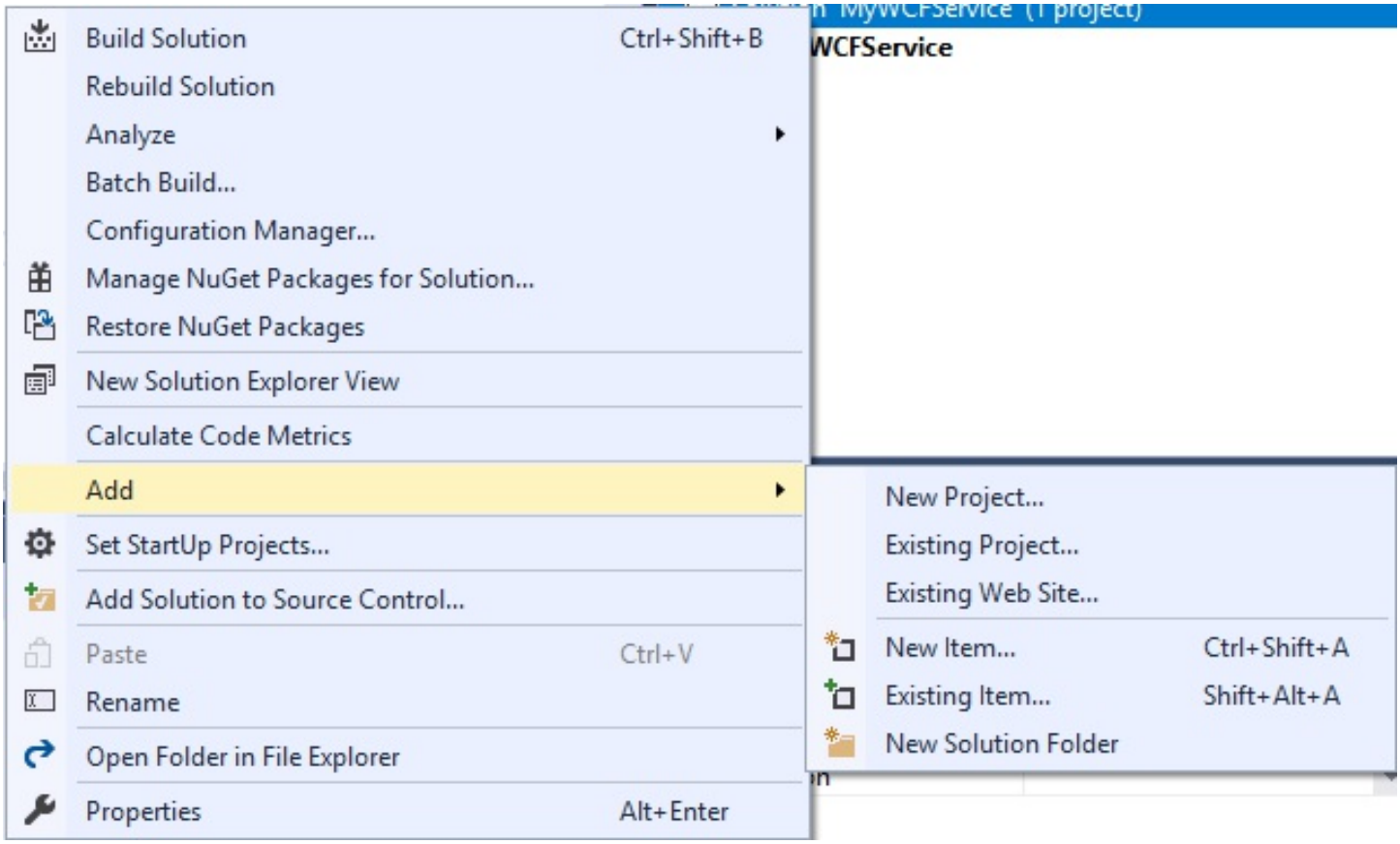
```
[OperationContract]
string MyWhateverMethod();
```

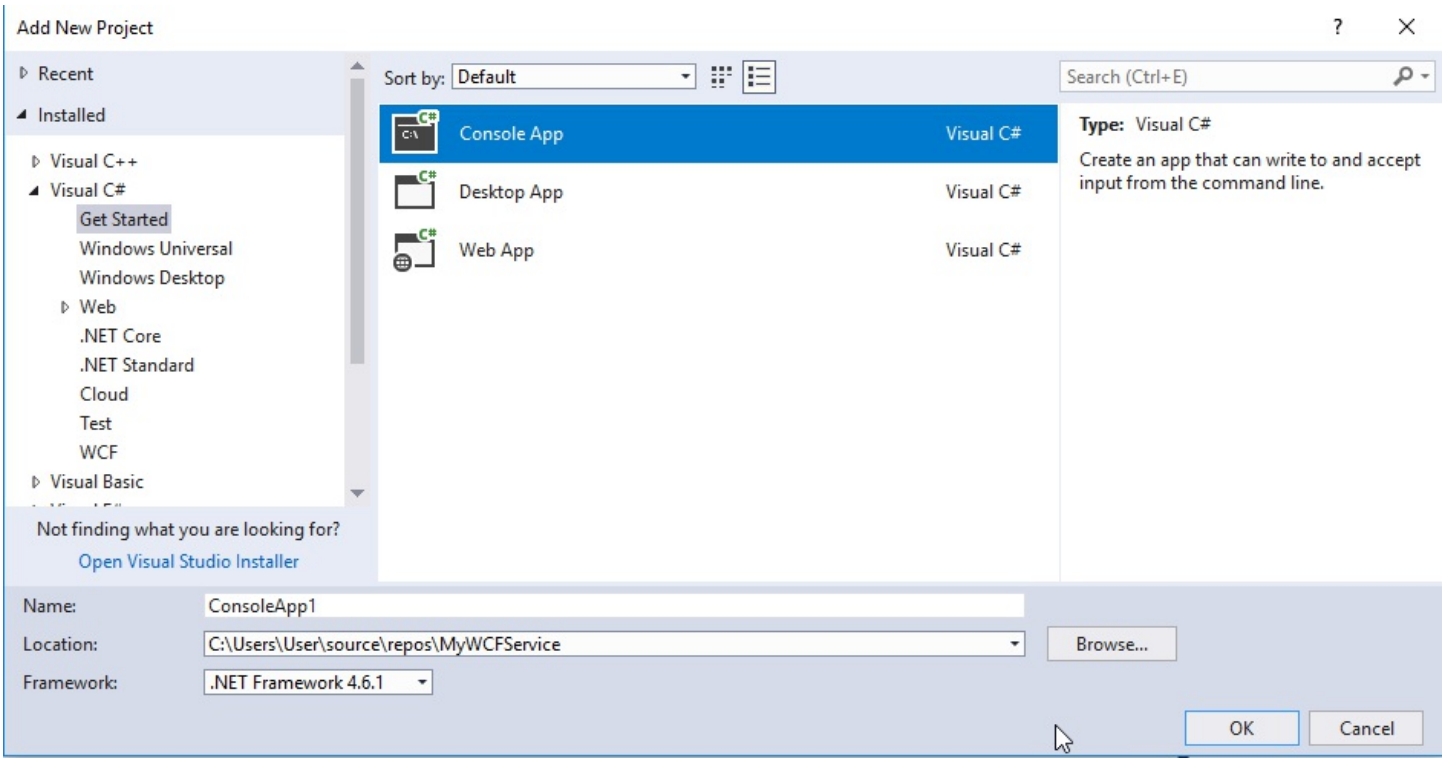
> *Don’t forget to make sure your service match exaclty the signature of your interface (Visual Studio will throw compilation errors if this happens)*

> *Every time that you update the Service, you have to right clikc on the Service Reference and perform and **Update Service Reference** under **Connected Services****

Create a WCF Client App

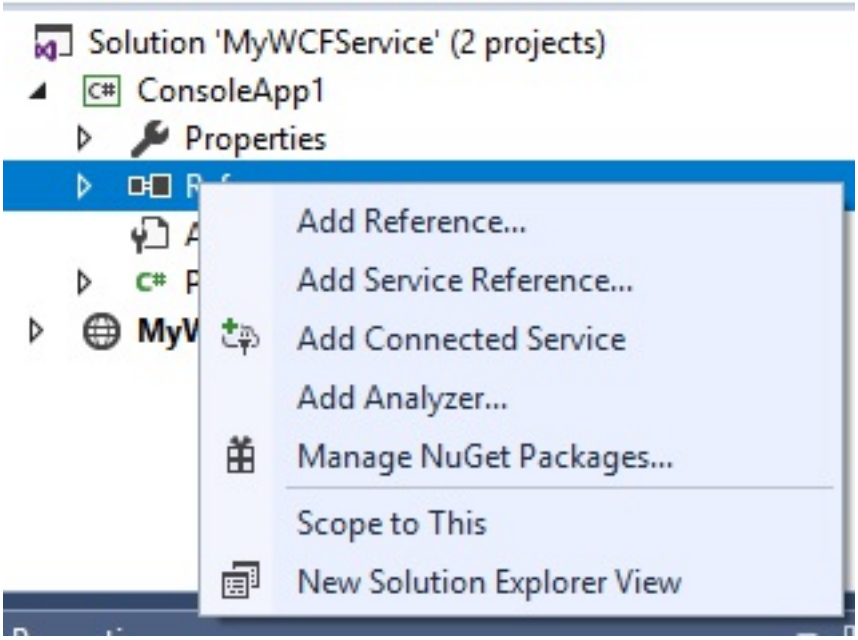
Add to the existing solution a new Console Application



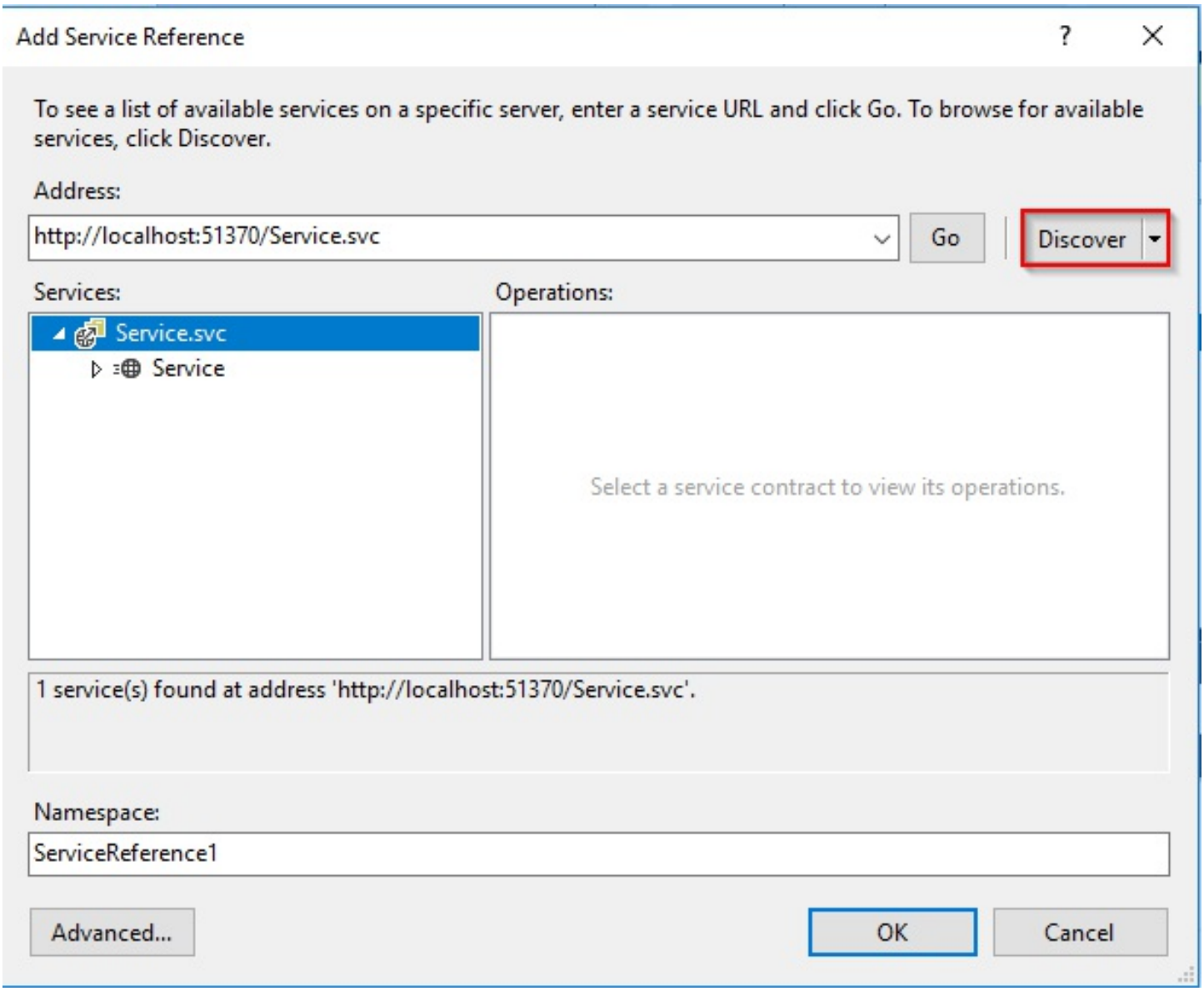


Now Add the **Service Reference** of the WCF Service to the Console

Application



Once the Dialog Window open, **click Discover** to get the Service Reference and **click Ok**



> *You can change the name of the Service Reference to something more meaningful*

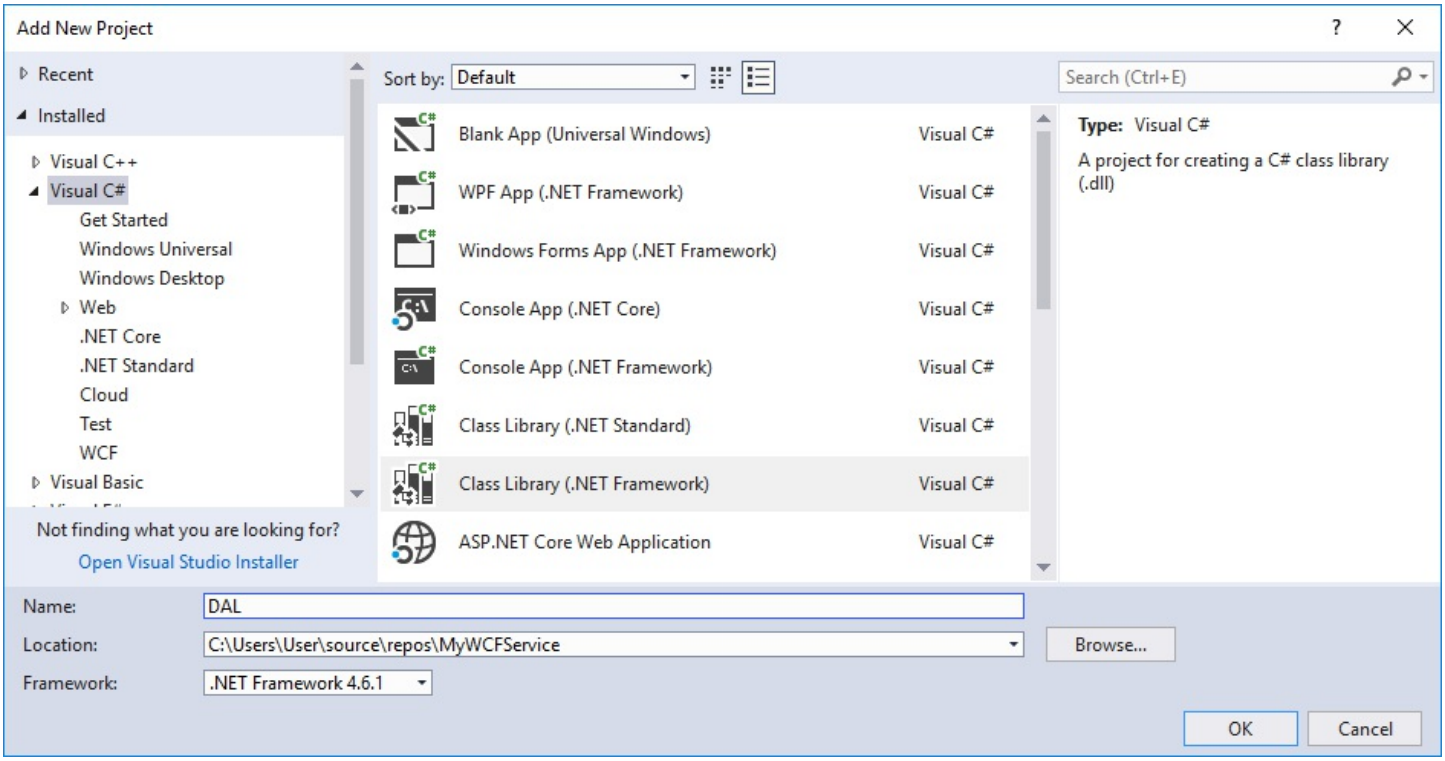
Now open the `Program.cs` and type the following code, this will create the client to consume the Service.

```
Console.WriteLine("Test Service Method");
//Build a Service Client Reference
var client = new <replacewithyourserviceref>.ServiceClient();
```

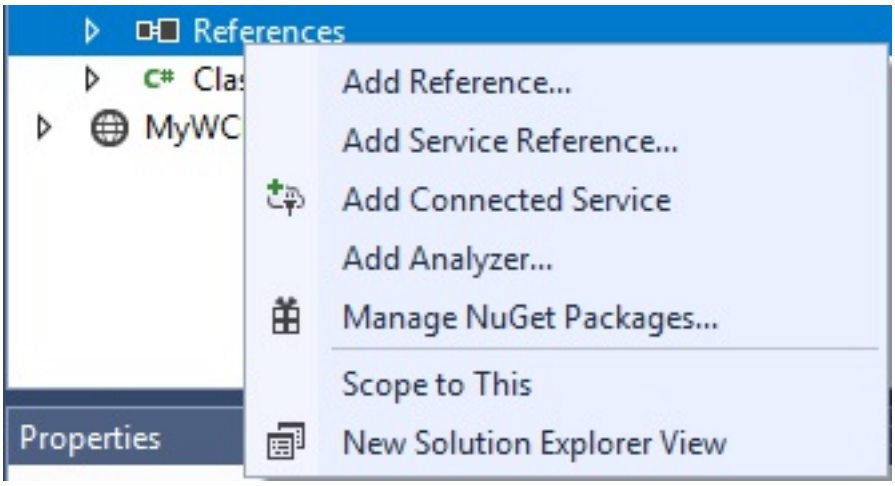
```
//Perform the Call to the Desired Method
var myServiceResult = client.<replacewithyourmethod>();

Console.WriteLine(myServiceResult);
Console.ReadKey();
```

Connect WCF With Entity Framework

Add a new Class Library to modularize the application, name the project `DAL`

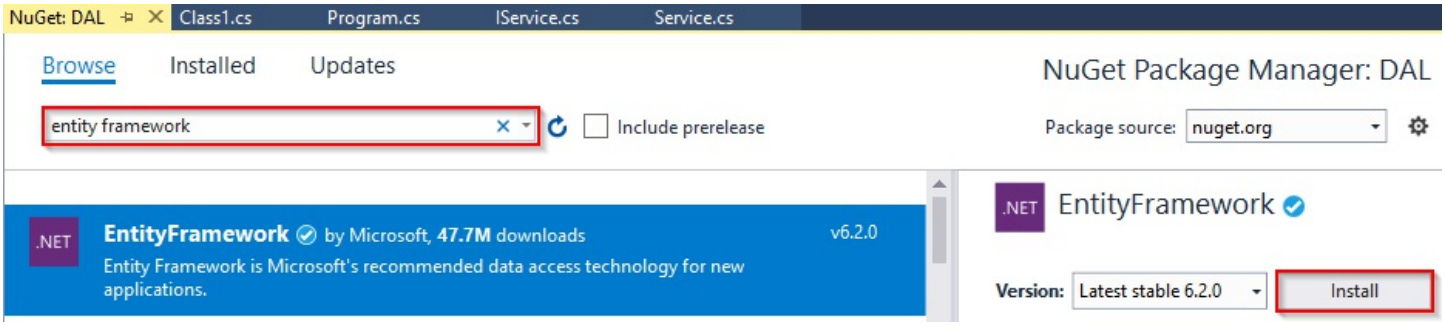


Now lets add the Entity Framework reference from Nuget (**Manage Nuget Packages...**)



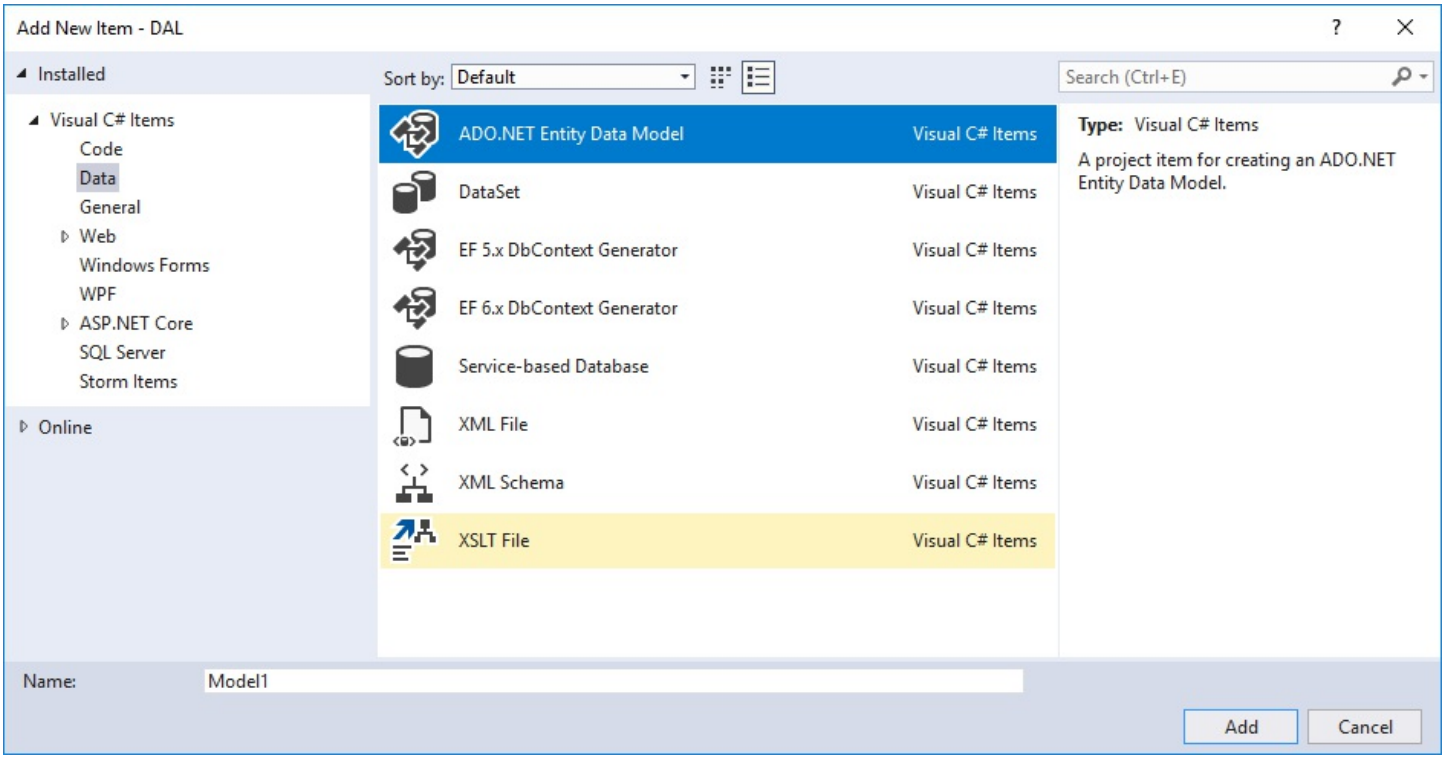
Select the Browse Tab and type **Entity Framework** on the input field

Select the Entity Framework Package and click **Install**



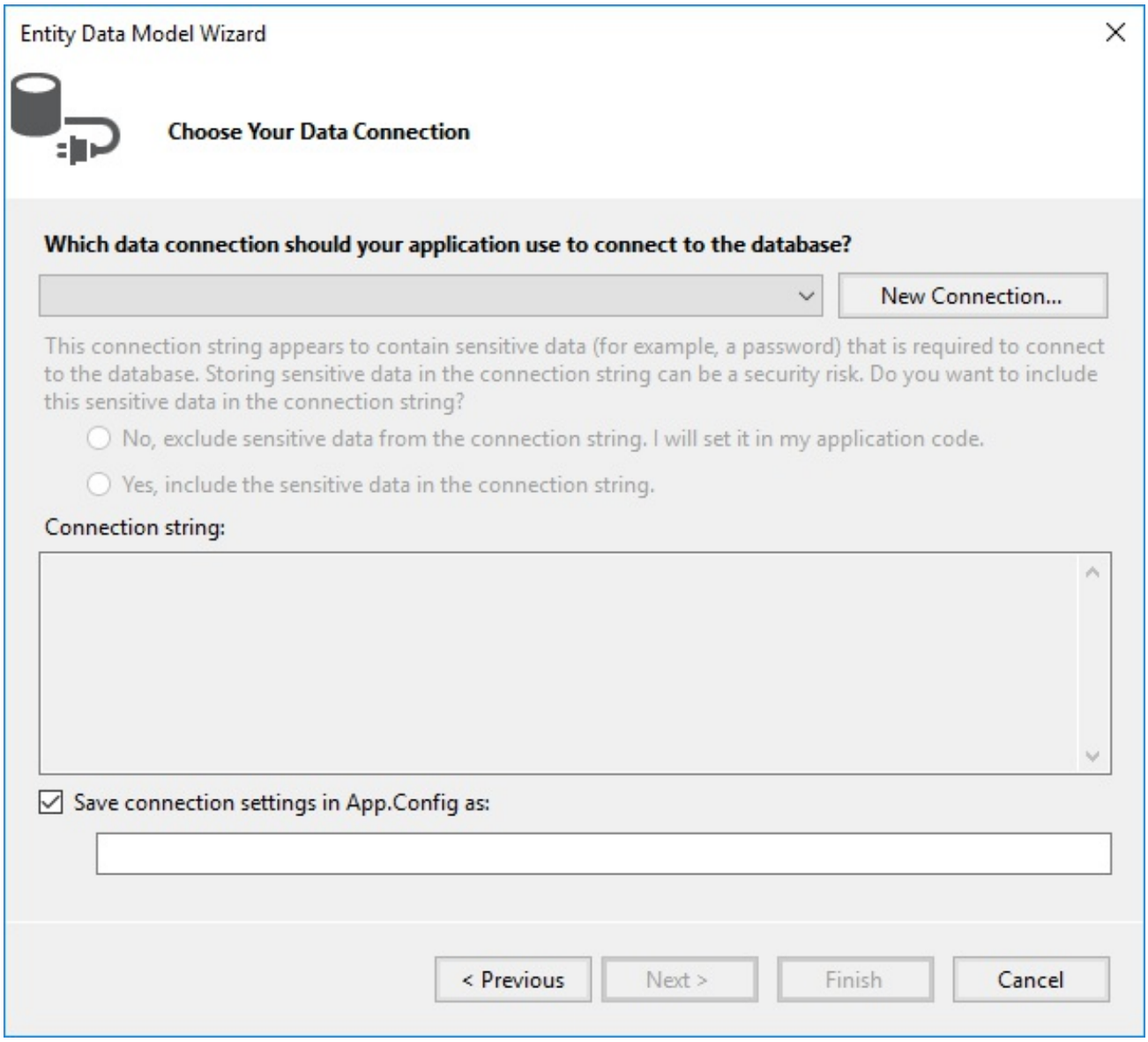
Modeling of Entities

Add a new Data Model



Select the EF Designer from Database

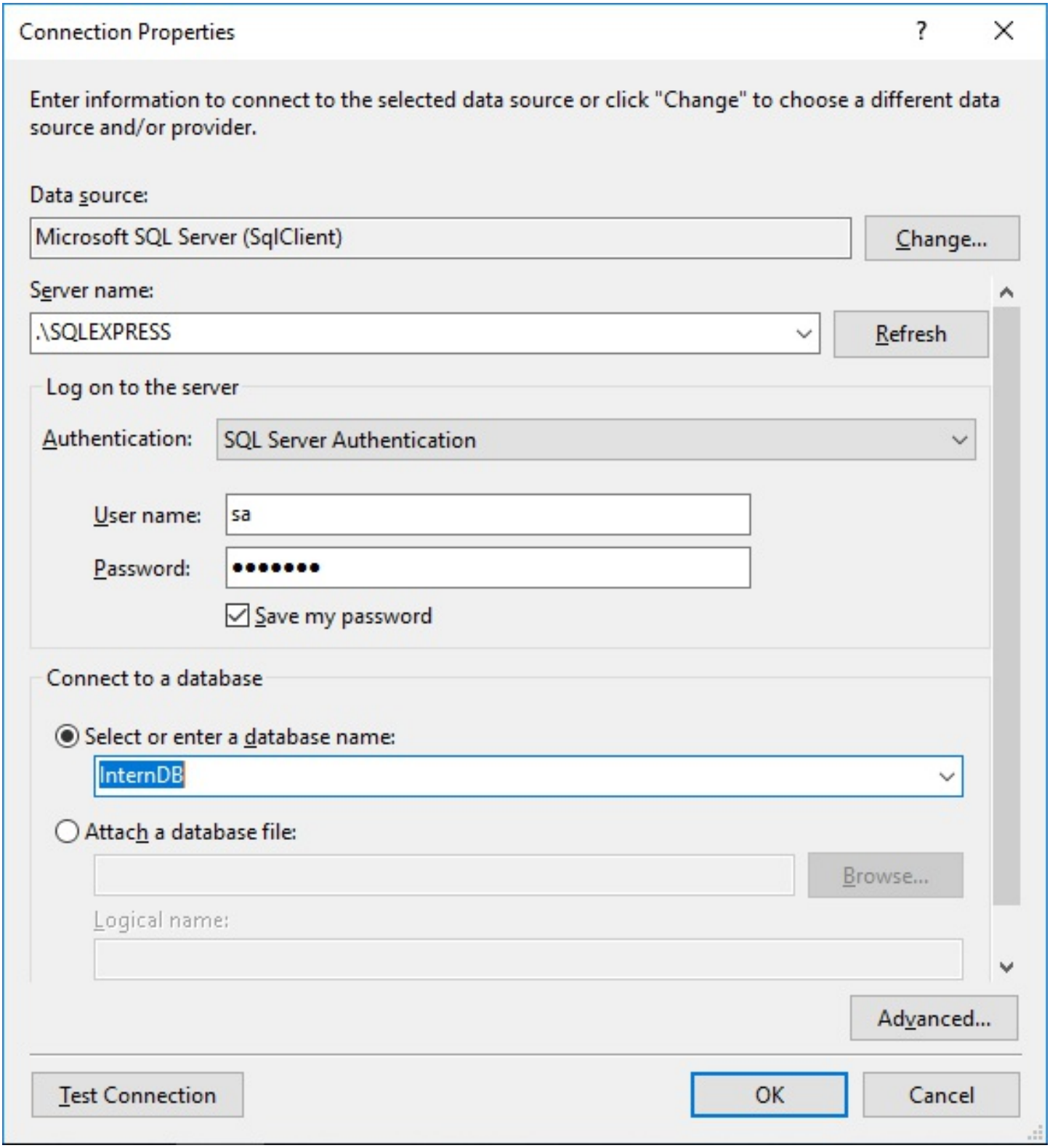
- Create a New Connection



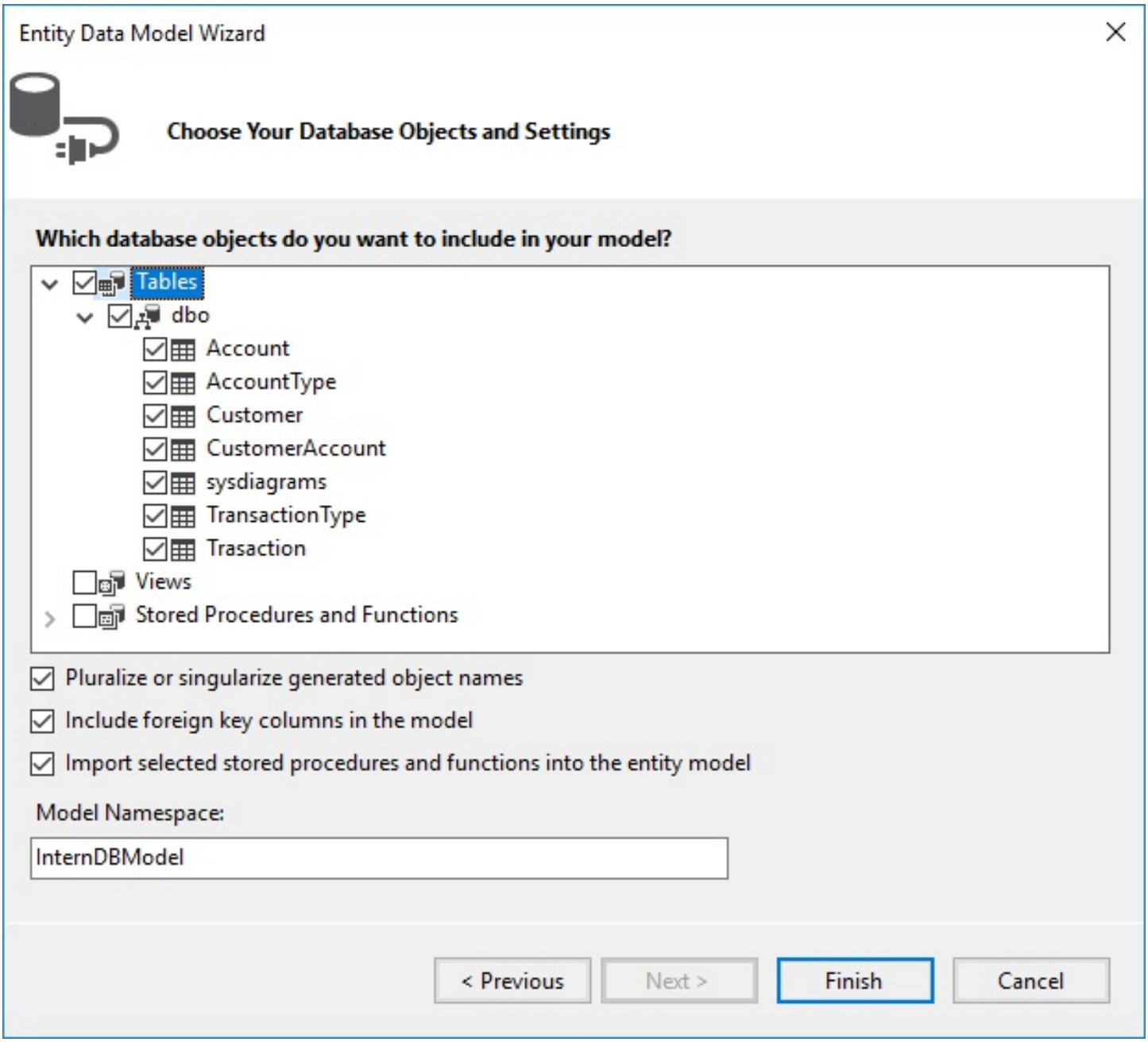
> *Make sure you check the:* ***Save connection setting in App.Config as:*** *and* ***Yes, include the sensitive data in the connection string.***

Fill the information as displayed here:

> *Username: sa*
>
> *Password: 1234567*



Select the Tables to be modeled for now



> *This process could take a while to be finish*

Now that everything is mapped you we can try to perform a query

On the project DAL, open the `Class1.cs` and renamed to `AccountTypeProvider.cs`

Update the class with this code:

```
public class AccountTypeProvider
{
```

```
    public List<AccountType> GetAccountTypes()
    {
        //New context for the query execution (open/close con
nection)
        using (InternDBEntities db = new InternDBEntities())
        {
            //This will be like a Select *
            List<AccountType> accountList = db.AccountTypes.T
oList();

            return accountList;
        }
    }
}
```

Now in order to consume this query we need to consume this from the Service

First add the connection string to the sevice

```
<connectionStrings>
...
</connectionStrings>
```

Go to my Service Project and open the `IService.cs`

Update the signature method to this:

```
 [OperationContract]
```

```
 List<AccountTypeResponse> GetAllAccountTypes();
```

Update the `Service.cs` to support the `IService` new method

```
public List<AccountTypeResponse> GetAllAccountTypes()
{
    var mapAccountTypeResponse = new List<AccountTypeResponse
>();
    //The method that we recently did !
    var accountTypeProvider = new DAL.AccountTypeProvider();
    var dataResult = accountTypeProvider.GetAccountTypes();

    //Manual Mapping, this sucks !
    foreach (var item in dataResult)
    {
        mapAccountTypeResponse.Add(new AccountTypeResponse
        {
            AccountTypeID = item.accountTypeID,
            AccountTypeName = item.accountTypeName
        });
    }

    return mapAccountTypeResponse;
}
```

Create a new folder inside App_Code called **Response**

Create the following class:

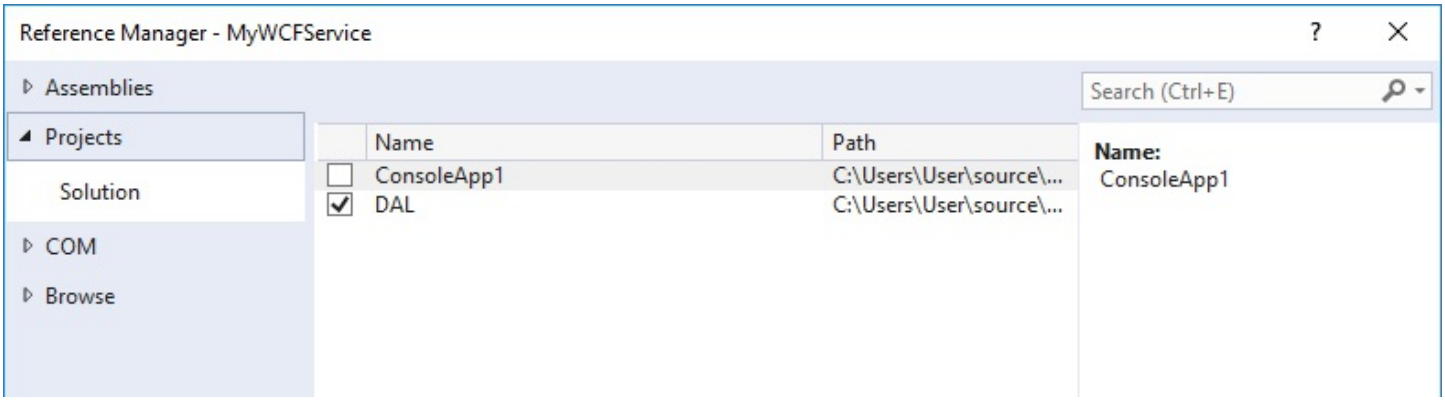
```
[DataContract]
public class AccountTypeResponse
{
    [DataMember]
    public int AccountTypeID { get; set; }

    [DataMember]
    public string AccountTypeName { get; set; }
}
```

> *We need to create a intermediate object to be able to transfer the information, look for DTO Concept instead of trying to pass the Entity Framework objects (which is a terrible practice)*

Build the project

There should be reference errors, this is happening because our DAL is not referenced

Go to the Service Project, click on any folder and Add References



Compile the Project

Update the Service Reference on the Console App

Now Update the `Program.cs` in the ConsoleApplication to return the Account Types

```
static void Main(string[] args)
{
    Console.WriteLine("Get All Account Types");

    var client = new MyServiceReference.ServiceClient();
    var myServiceResult = client.GetAllAccountTypes();
    foreach (var item in myServiceResult)
    {
        Console.WriteLine($"Account TypeId: {item.AccountType
ID} | Account Type Name: {item.AccountTypeName}");
    }

    Console.ReadKey();
}
```