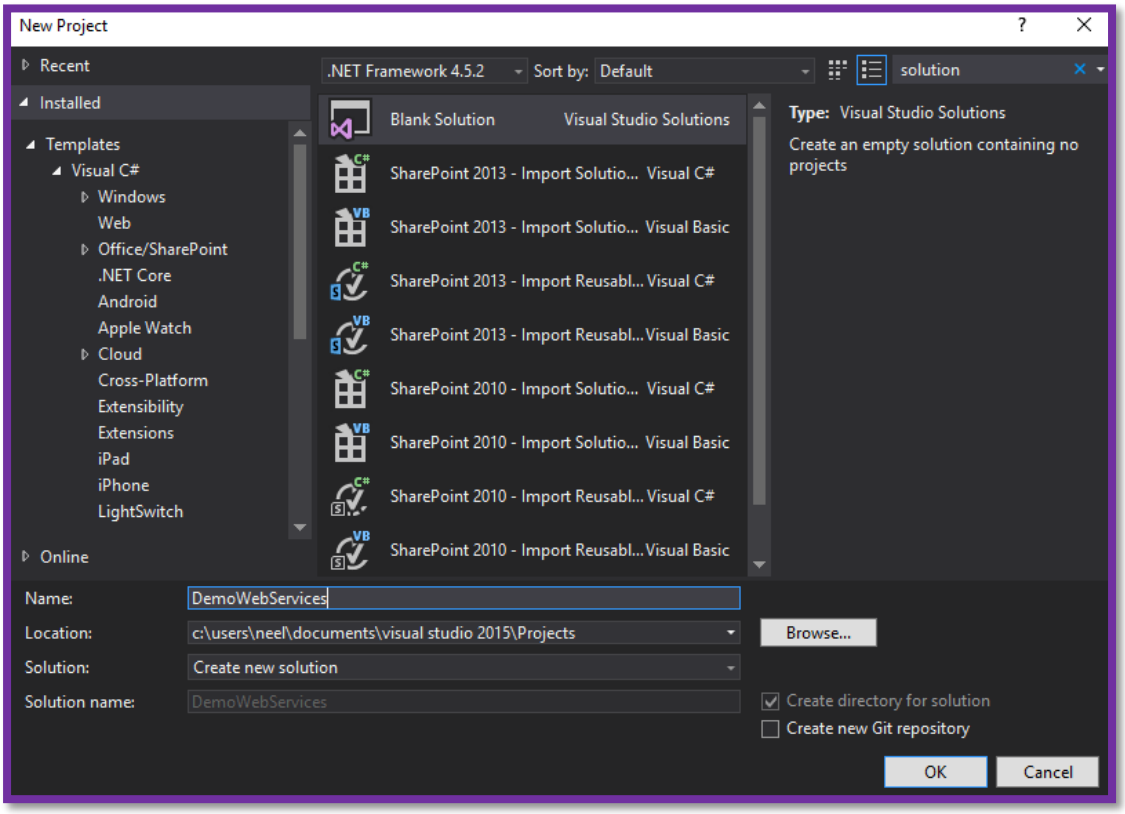
Web Services with WCF – Part 1/2

WCF (Windows Commutation Foundation)

WCF stands for Windows Communication Foundation. It is a framework for building, configuring, and deploying network-distributed services. Earlier known as Indigo, it enables hosting services in any type of operating system process.

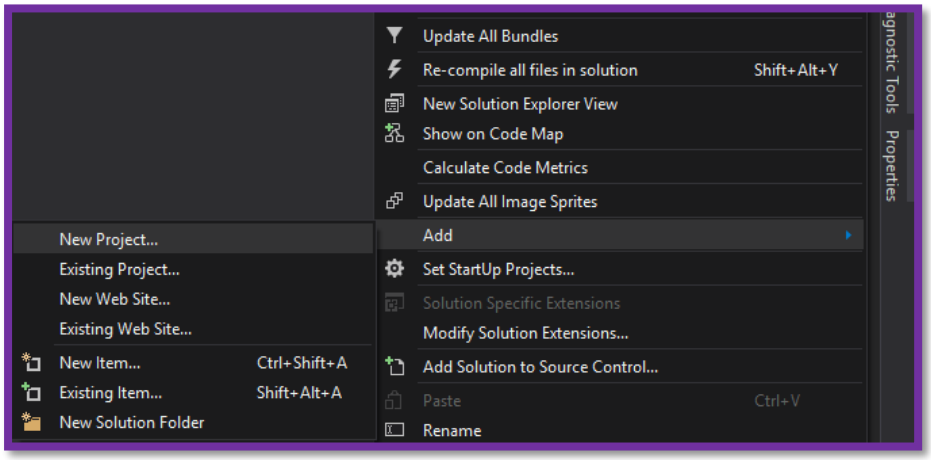
This tutorial explains the fundamentals of WCF and is conveniently divided into various sections. Every section of this tutorial has adequate number of examples to explain different concepts of WCF.

For this example, need add a new solution with the name "DemoWebServices"

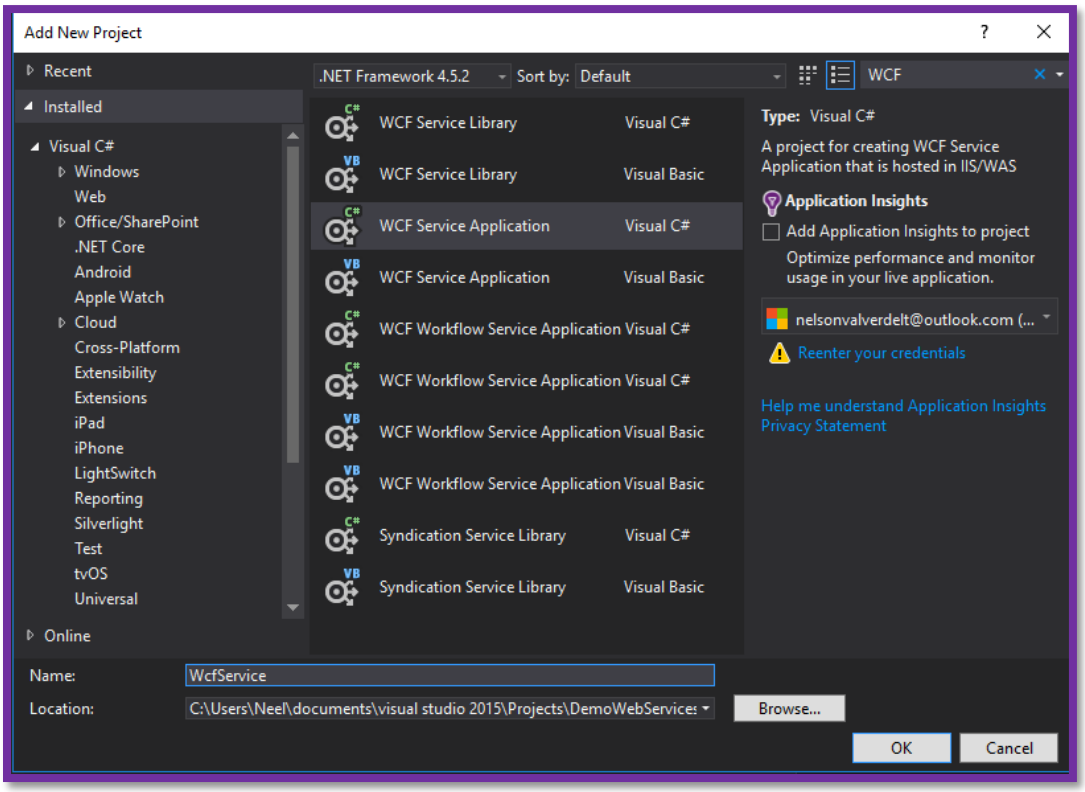


Now we create two projects:

- Right click in our solution and new project



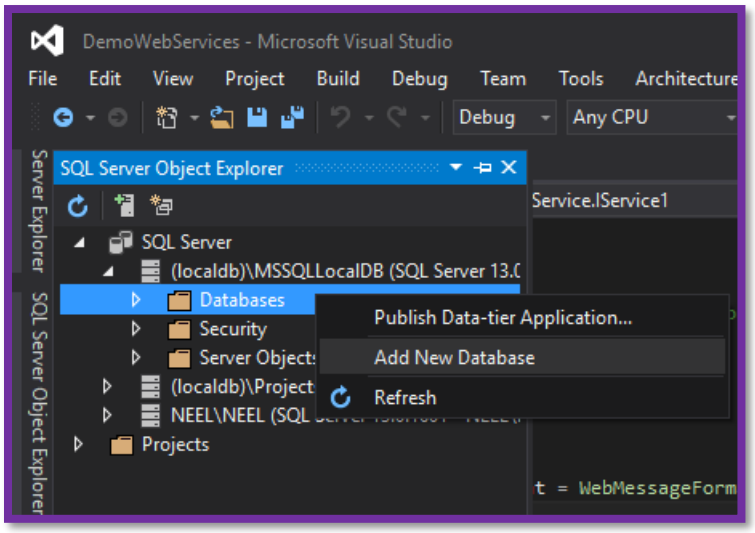
- Search and select “WCF Service Application”, then add the name “WcfService” and “OK”



Now we follow these steps for our project "WcfService"

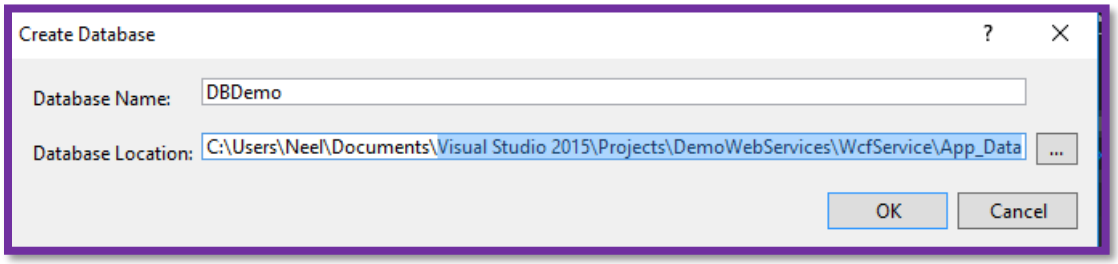
STEP 1

· Select Sql Server Object Explorer and create your local Database.



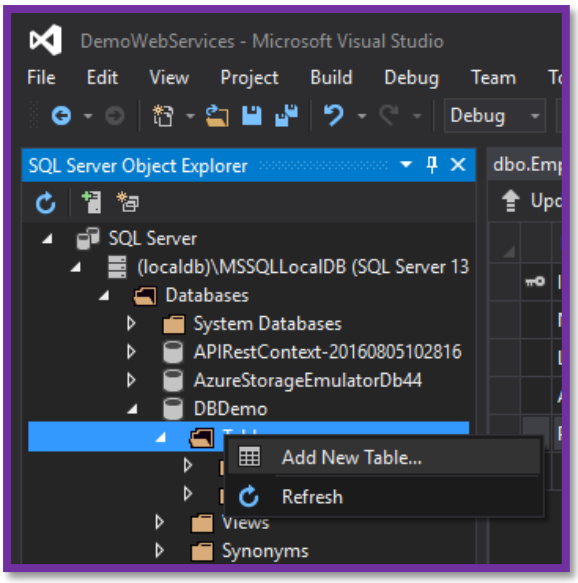
STEP 2

✓ Select your Database location, this having objective create a file local database with extension ".mdf"

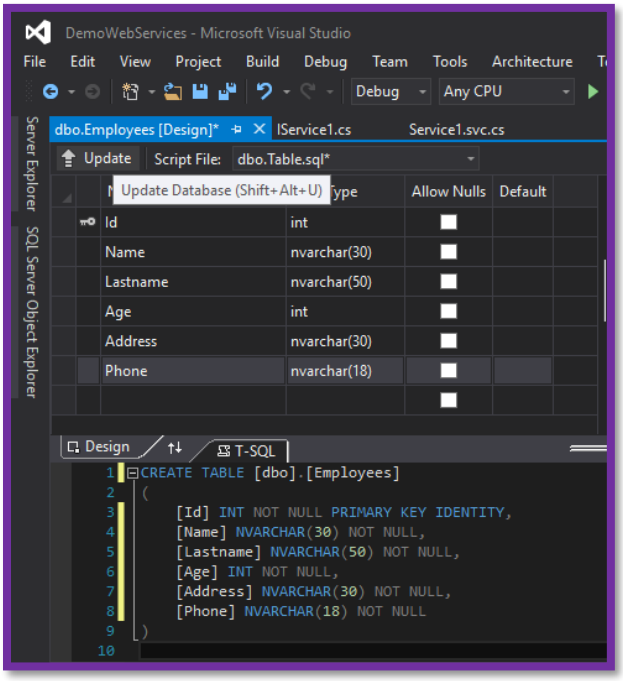


STEP 3

- ✓ Select our table file and add new table

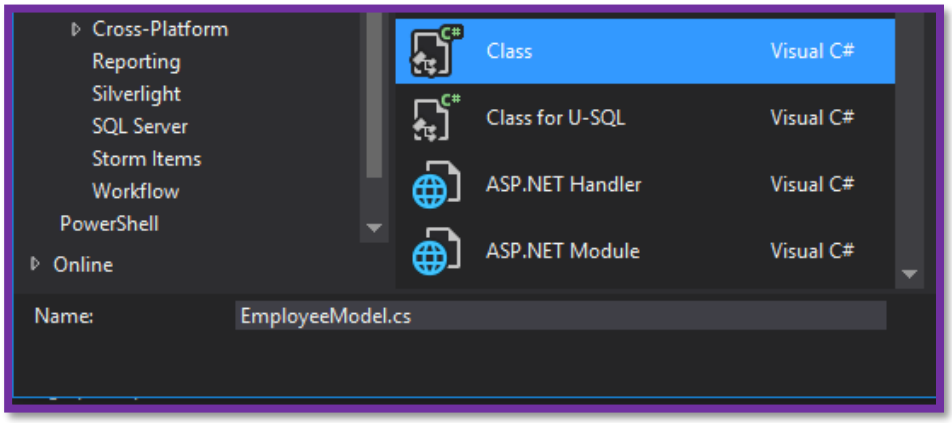


- ✓ Add a table "Employees" for this example, Add our attributes: Id, Name, Lastname, Address, Phone and "update" database



STEP 4

- ✓ Create a file “Models” and create a new class with the name “EmployeeModel”.



- ✓ Add the attributes at our class EmployeeModel: Id, Name, Lastname, Address, Phone

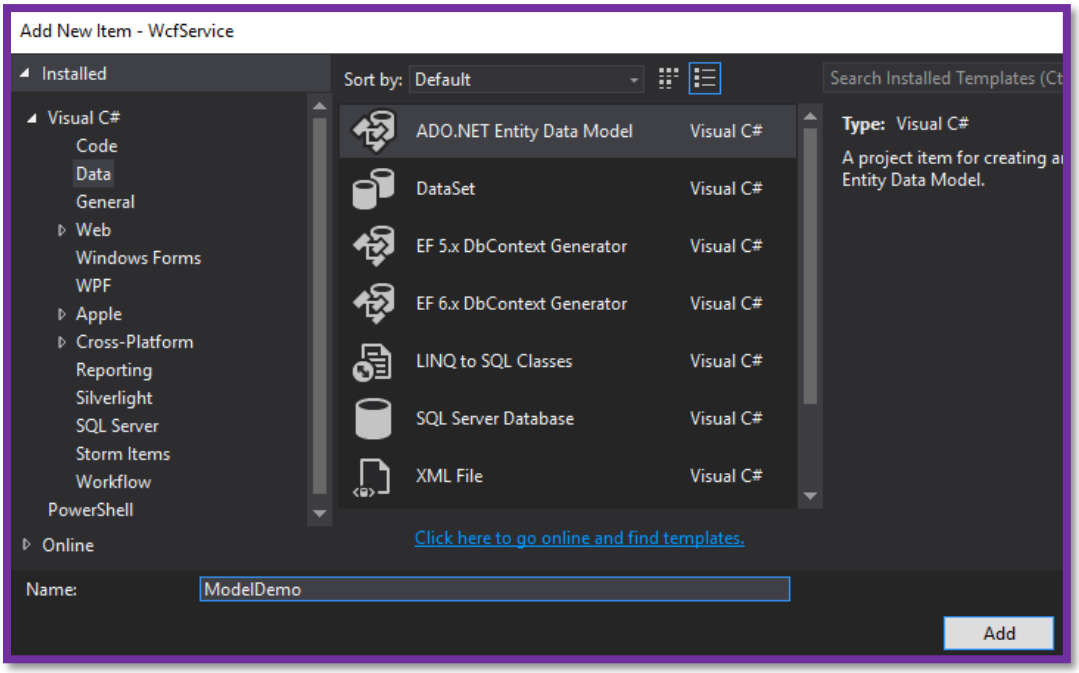
```
using System.Runtime.Serialization;

namespace WcfService.Models
{
    [DataContract]
    0 references
    public class EmployeeModel
    {
        [DataMember]
        0 references
        public int Id { get; set; }
        [DataMember]
        0 references
        public string Name { get; set; }
        [DataMember]
        0 references
        public string Lastname { get; set; }
        [DataMember]
        0 references
        public int Age { get; set; }
        [DataMember]
        0 references
        public string Phone { get; set; }
    }
}
```

- ✓ **DataContract:** Uses a serialization engine called the Data Contract Serializer by default to serialize and deserialize data.

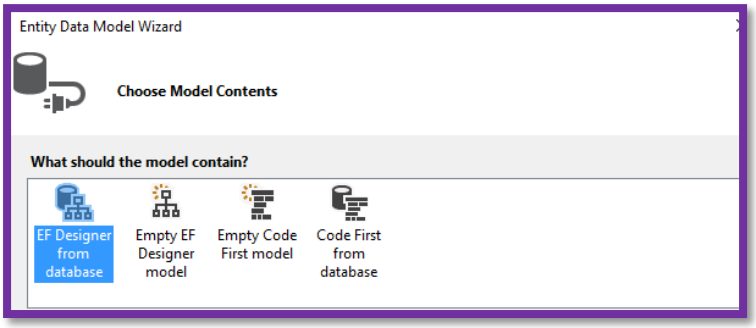
- ✓ **DataMember:** The attribute must then be applied to each member of the data contract type to indicate that it is a data member, that is, it should be serialized.

STEP 5

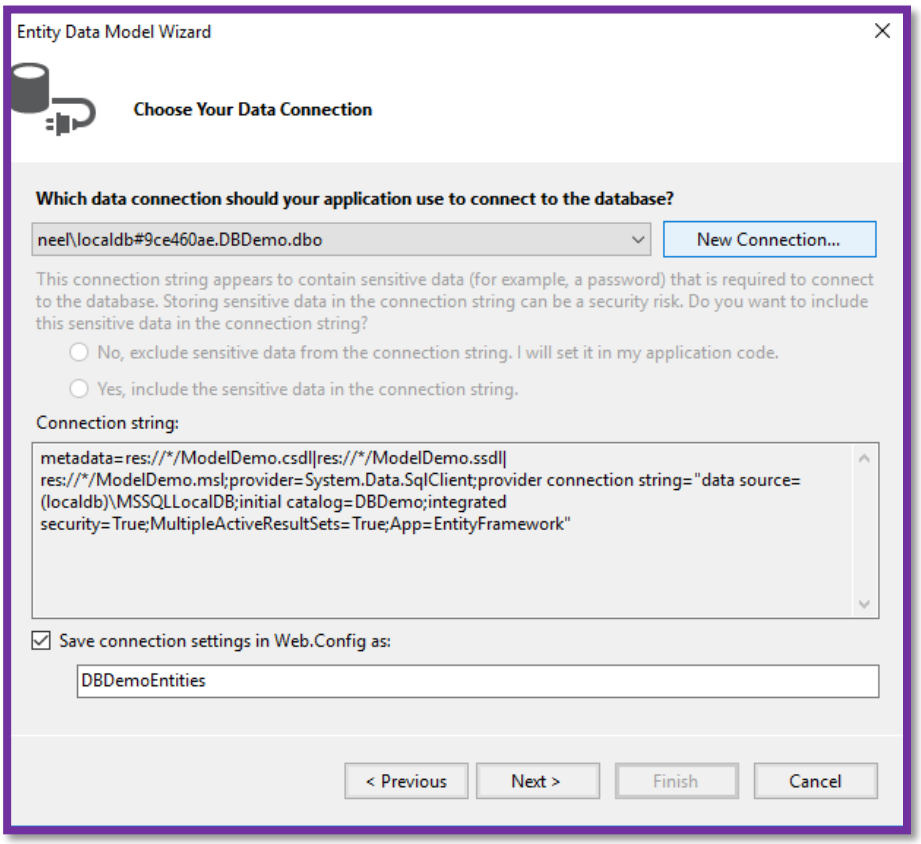
- ✓ Select a new item, select data and Create a ADO.NET Entity Data Model with the name "ModelDemo"



- ✓ Select **EF Designer From Database** and Next

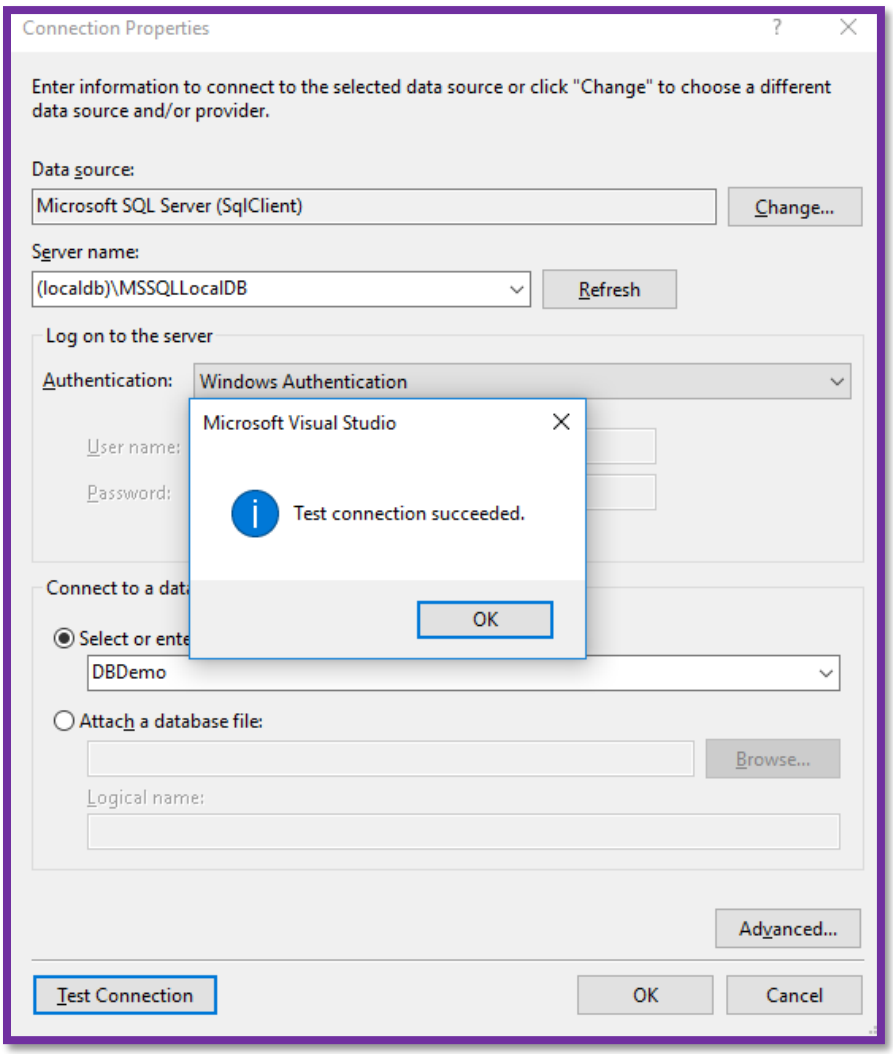


✓ Select New Connection



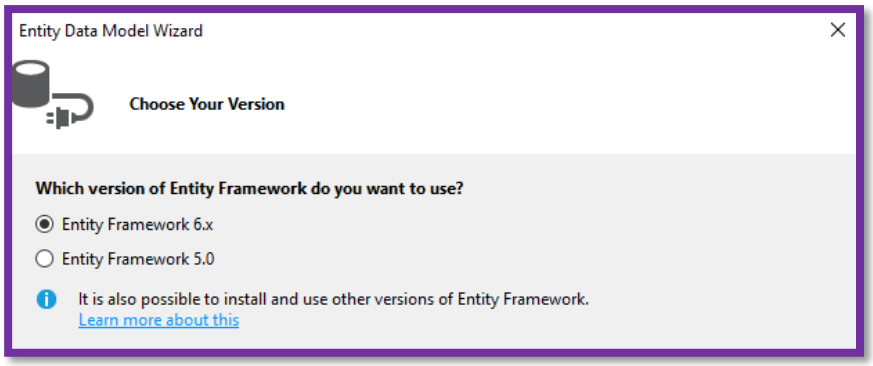
✓ In Connection Properties add a new connection

Note: In our Server name I do reference a database local

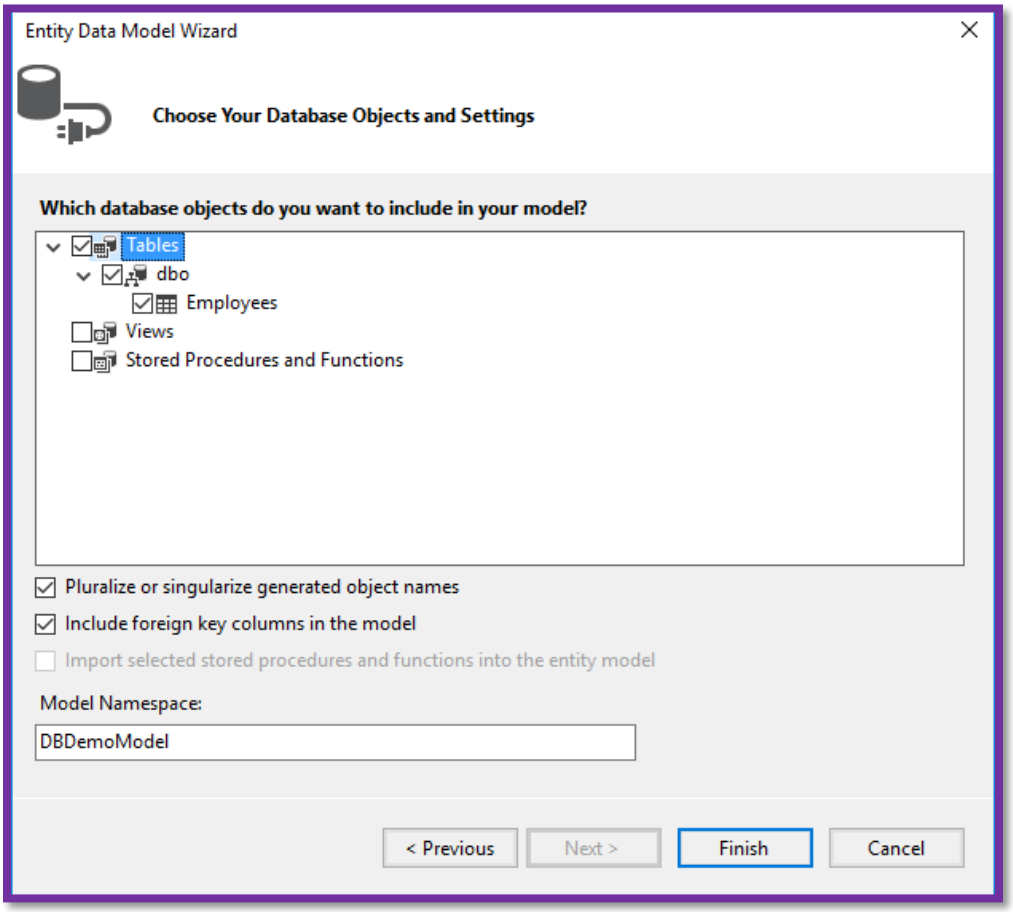


✓ Select Entity Framework 6.x and Next,

Entity Framework: Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write.



✓ Select the table **Employees** for this example and Finish



STEP 6

- ✓ Modify our Interface Service **"IService1"**

```
using System.Collections.Generic;
using System.IO;
using System.ServiceModel;
using System.ServiceModel.Web;
using WcfService.Models;

namespace WcfService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1"
    [ServiceContract]
    1 reference
    public interface IService1
    {

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, BodyStyle = WebMessageBodyStyle.Wrapped,
            UriTemplate = "ListEmployees")]
        1 reference
        List<EmployeeModel> ListEmployees();

        [OperationContract]
        [WebGet(ResponseFormat = WebMessageFormat.Json, UriTemplate = "FindEmployee/{id}")]
        1 reference
        EmployeeModel FindEmployee(string id);

        [OperationContract]
        [WebInvoke(Method = "POST", RequestFormat = WebMessageFormat.Json, UriTemplate = "CreateEmployee")]
        1 reference
        string CreateEmployee(Stream JsonStream);

    }
}
```

STEP 7

- ✓ Create an instance of our string connection, this allow to interact with our database, in this example is **"DBDemoEntities"**
- ✓ Then include this within our method "CreateEmployee"

```
//Instantance Database
DBDemoEntities db = new DBDemoEntities();
1 reference
public string CreateEmployee(Stream JsonStream)
{
    try
    {
        //initialise a new instance of the StreamReader class for the specified stream
        StreamReader reader = new StreamReader(JsonStream);

        //Read Contennt json
        string json = reader.ReadToEnd();

        //initialise Serializer of JSON
        JavaScriptSerializer jss = new JavaScriptSerializer();

        //Deserializaer json and save in our Model
        EmployeeModel employee = jss.Deserialize<EmployeeModel>(json);

        //Save our serialized data to Model Mapping of our database
        var emp = new Employee
        {
            Name = employee.Name,
            Lastname = employee.Lastname,
            Age = employee.Age,
            Address = employee.Address,
            Phone = employee.Phone
        };

        //Add a data entity emp
        db.Employees.Add(emp);

        //Save Changes to our database
        db.SaveChanges();

        return "OK";
    }
    catch (Exception ex)
    {
        return ex.Message;
    }

}
```

✓ Add this within our method **FindEmployee**

```
1 reference
public EmployeeModel FindEmployee(string id)
{
    // Use  Linq for database, Use "where" for an id existing and get data

    var query = from e in db.Employees
                where e.Id.ToString().Equals(id)
                select new EmployeeModel()
                {
                    Name = e.Name,
                    Lastname = e.Lastname,
                    Age = e.Age,
                    Address = e.Address,
                    Phone = e.Phone
                };

        return query.FirstOrDefault();
}
```

✓ Add this within our method **ListEmployees**

```
1 reference
public List<EmployeeModel> ListEmployees()
{
    // Use  Linq for database, save the data in our Models and get data

    var query = from e in db.Employees
                select new EmployeeModel()
                {
                    Name = e.Name,
                    Lastname = e.Lastname,
                    Age = e.Age,
                    Address = e.Address,
                    Phone = e.Phone
                };
    return query.ToList();
}
```

STEP 8

✓ Go to **web.config** and add this:

- Add this within of **<system.webServer>**

```
<httpProtocol>
    <customHeaders>
            <add name="Access-Control-Allow-Origin" value="*" />
            <add name="Access-Control-Allow-Headers" value="Content-Type, Accept" />
     </customHeaders>
</httpProtocol>
```

- Add this within of **<behaviors>**

```
<endpointBehaviors>
        <behavior name="WebService">
                <webHttp/>
        </behavior>
</endpointbehaviors>
```

- Add this within and below of **<system.serviceModel>**

```
<services>
        <service name="WcfService.Service1">
                <endpoint address="" binding="webHttpBinding"
                contract="WcfService.IService1"
                behaviorConfiguration="WebService" />
         </service>
</services>
```

Note:

WcfService: Is the name of our project

IServices1: Is our interface

Service1: Is our class with our methods