BridgeLabz

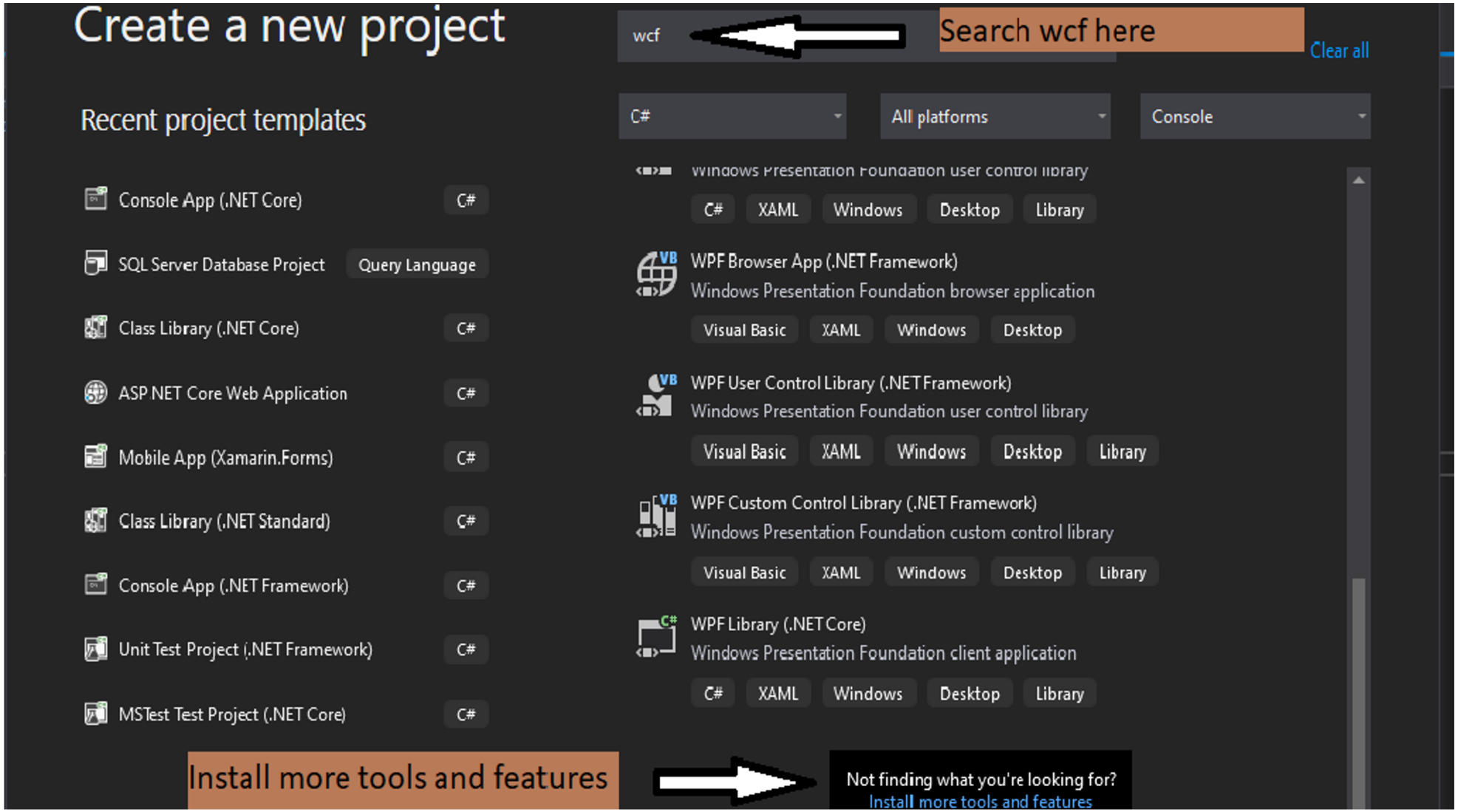
Employability Delivered

WCF - Windows Communication Foundation

Windows Communication Foundation

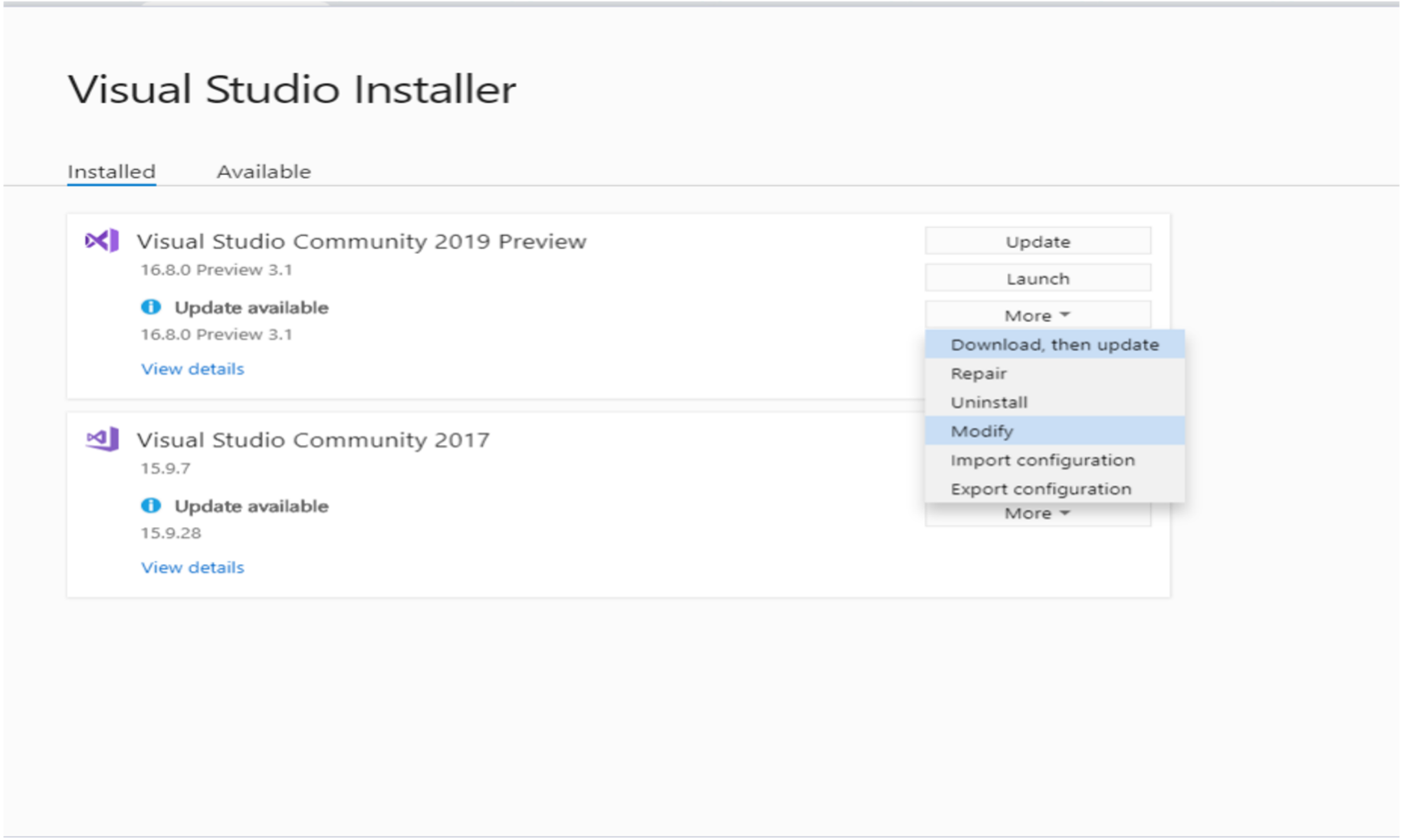
- WCF stands for Windows Communication Foundation.
- It is a framework for building, configuring, and deploying network-distributed services.
- Send data as asynchronous messages from one service endpoint to another.
- An endpoint can be a client of a service that requests data from a service endpoint.
- Earlier known as Indigo, it enables hosting services in any type of operating system process

Install WCF on Visual studio 2019



Search wcf here, if it is not found in Project templates than you need to install, so for that click Install more tools and features and refer next slides for further instructions

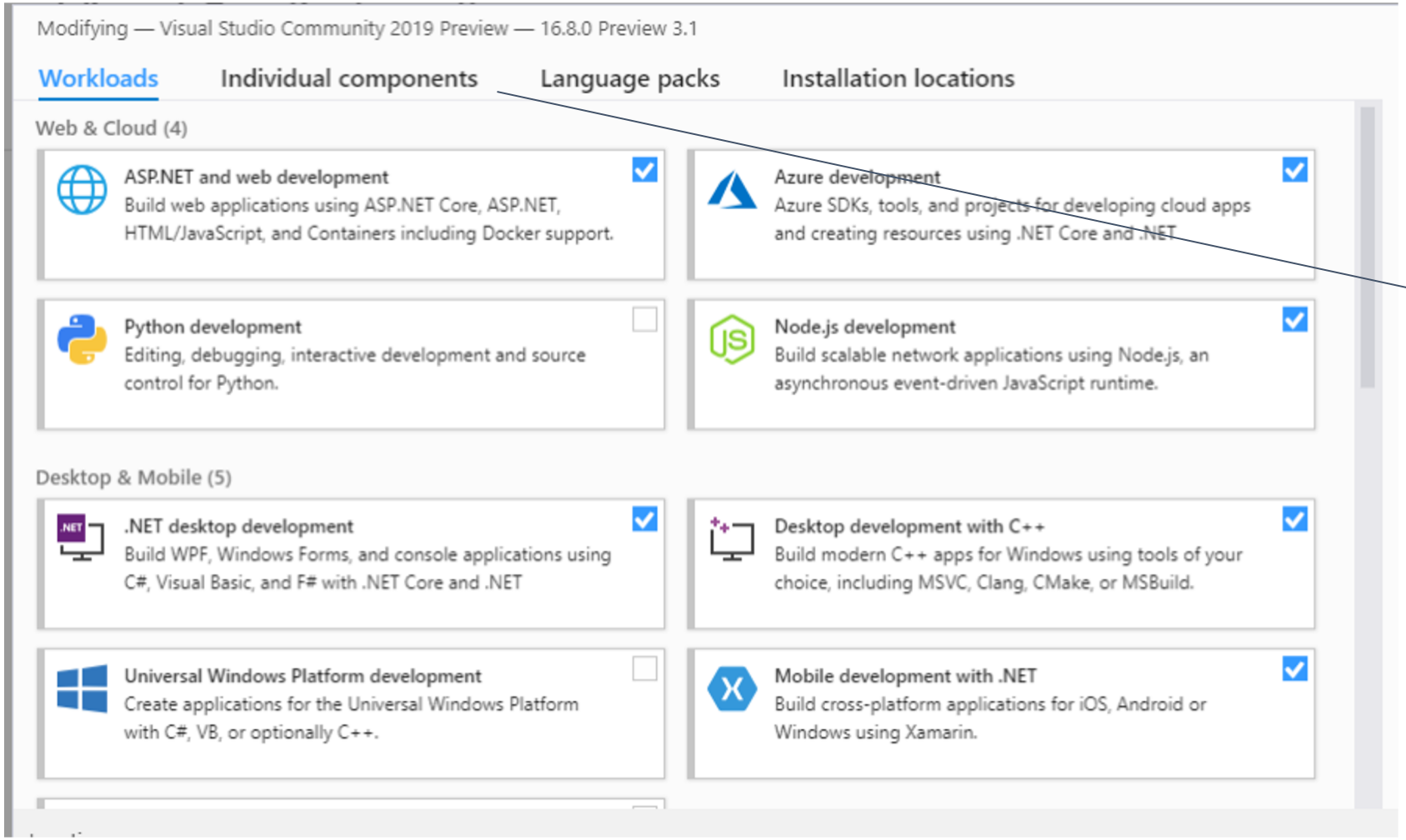
Install WCF in Visual Studio 2019



If you are not able to find install more tools and features, you can go to Visual studio installer and search for modify in drop down.

Note : This is just alternative for the process from previous slide

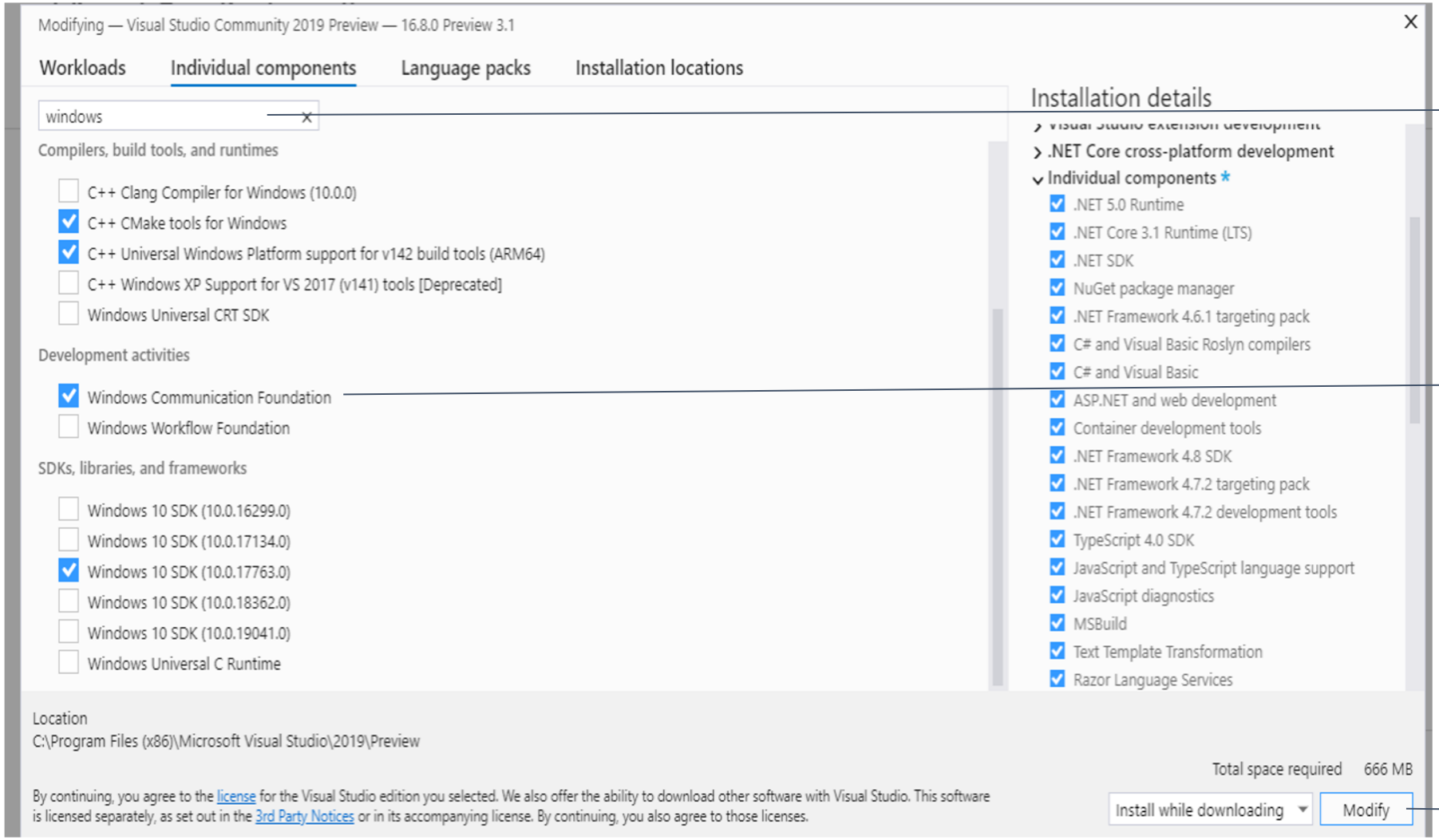
Install WCF on Visual studio 2019



Select individual components

.

Install WCF on Visual studio 2019



1. **Search for windows communication foundation**

2. Mark Windows communication foundation

3. Click on Modify

REST Services

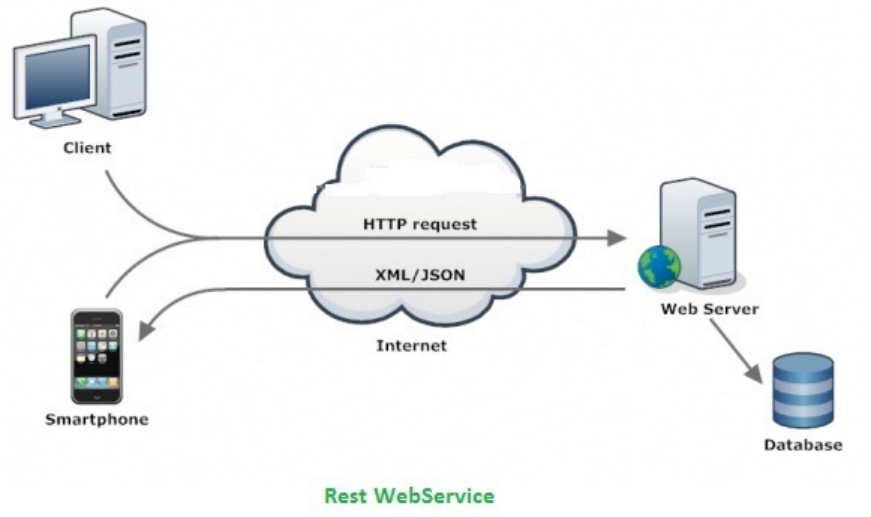
- Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services.
- Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the internet.
- RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations.
- Other kinds of Web services, such as SOAP Web services, expose their own arbitrary sets of operations.

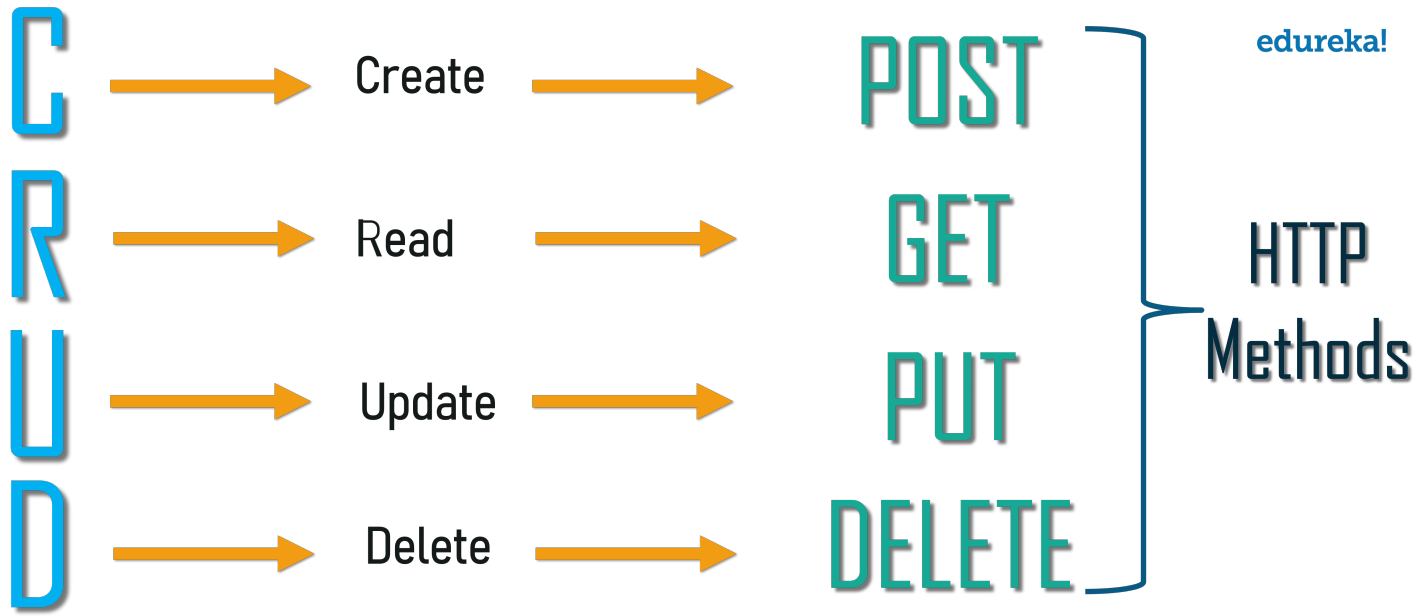
CRUD operations in WCF REST service

To Perform CRUD operations in WCF REST service we need to use the following HTTP methods:

- **GET :** Get the resource (Records) from a particular source such as SQL database.
- **POST :** Used to insert the records into a particular source such as SQL database.
- **PUT :** Used to modify the resource or records.
- **DELETE :** used Delete the specific resource or record from a particular source.

RESTFUL SERVICES - Architecture





References: SOAP Vs REST

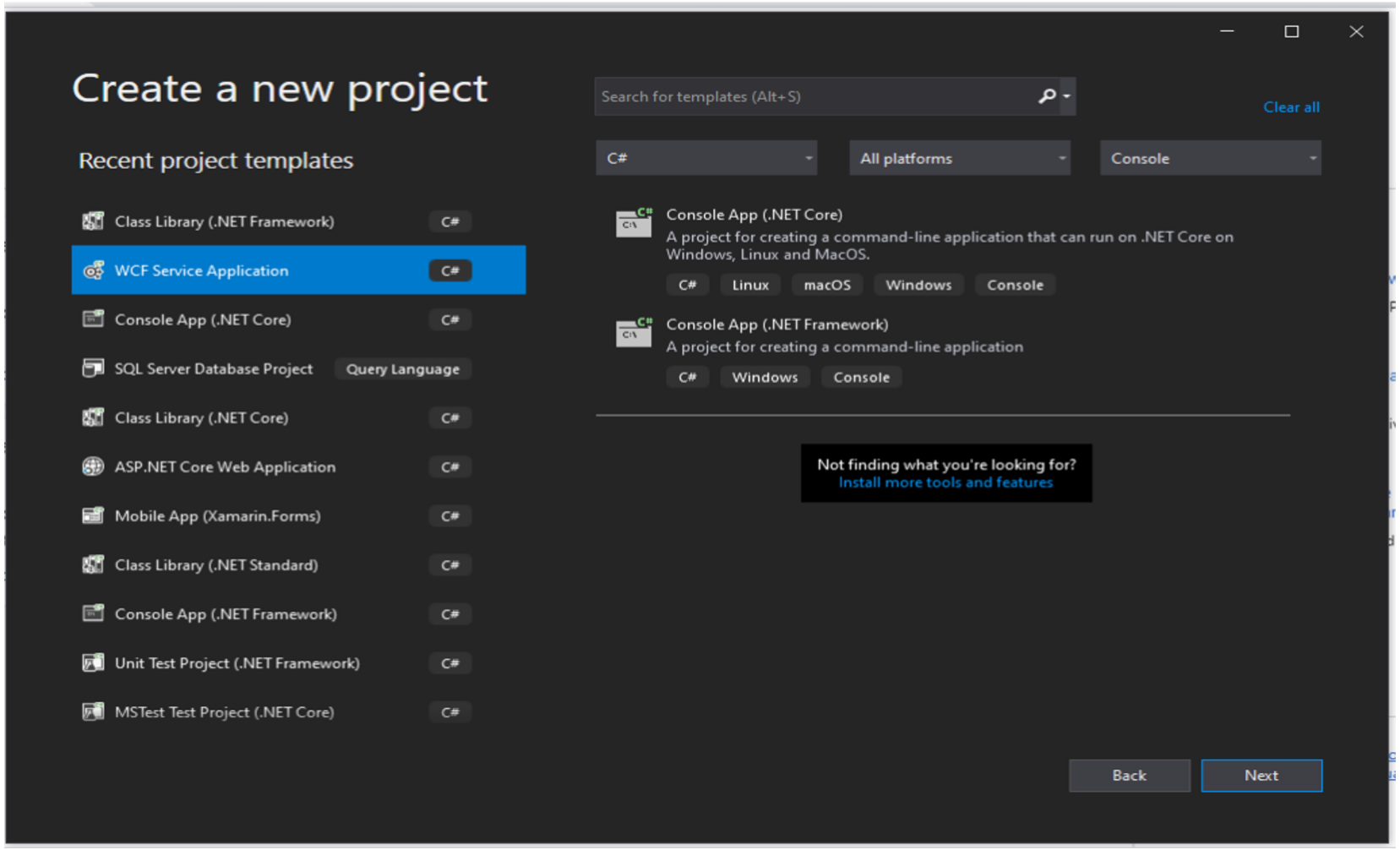
3 Tier Architecture

We will have 3 layers -

- WCF Project -> the layer which can act as REST services.
- BusinessManager -> layer to write business logic for our application.
- RepositoryManager -> layer which will interact with our database.

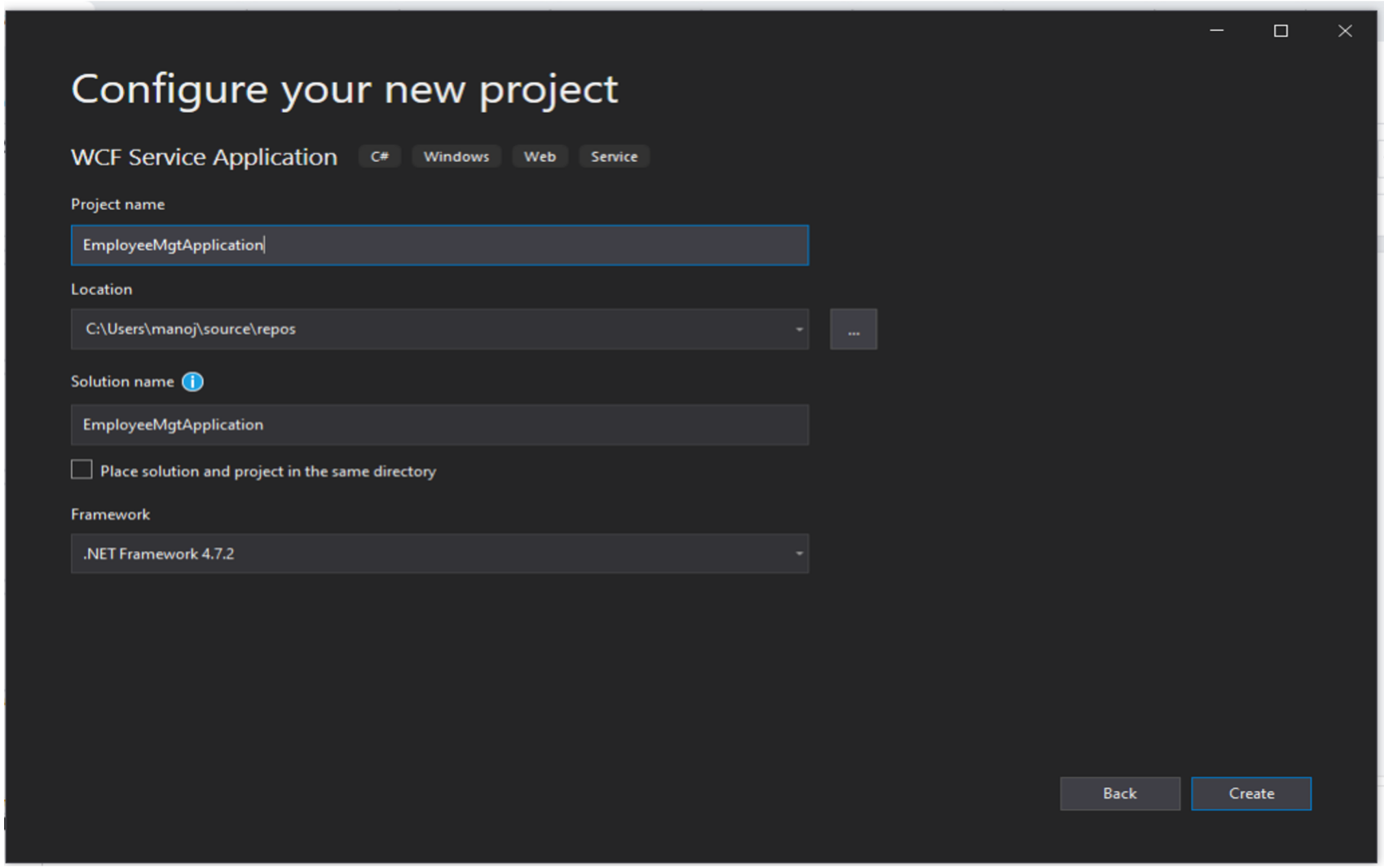
Common - > Here we can include Model class, constant error or success messages…

Let’s create a WCF project



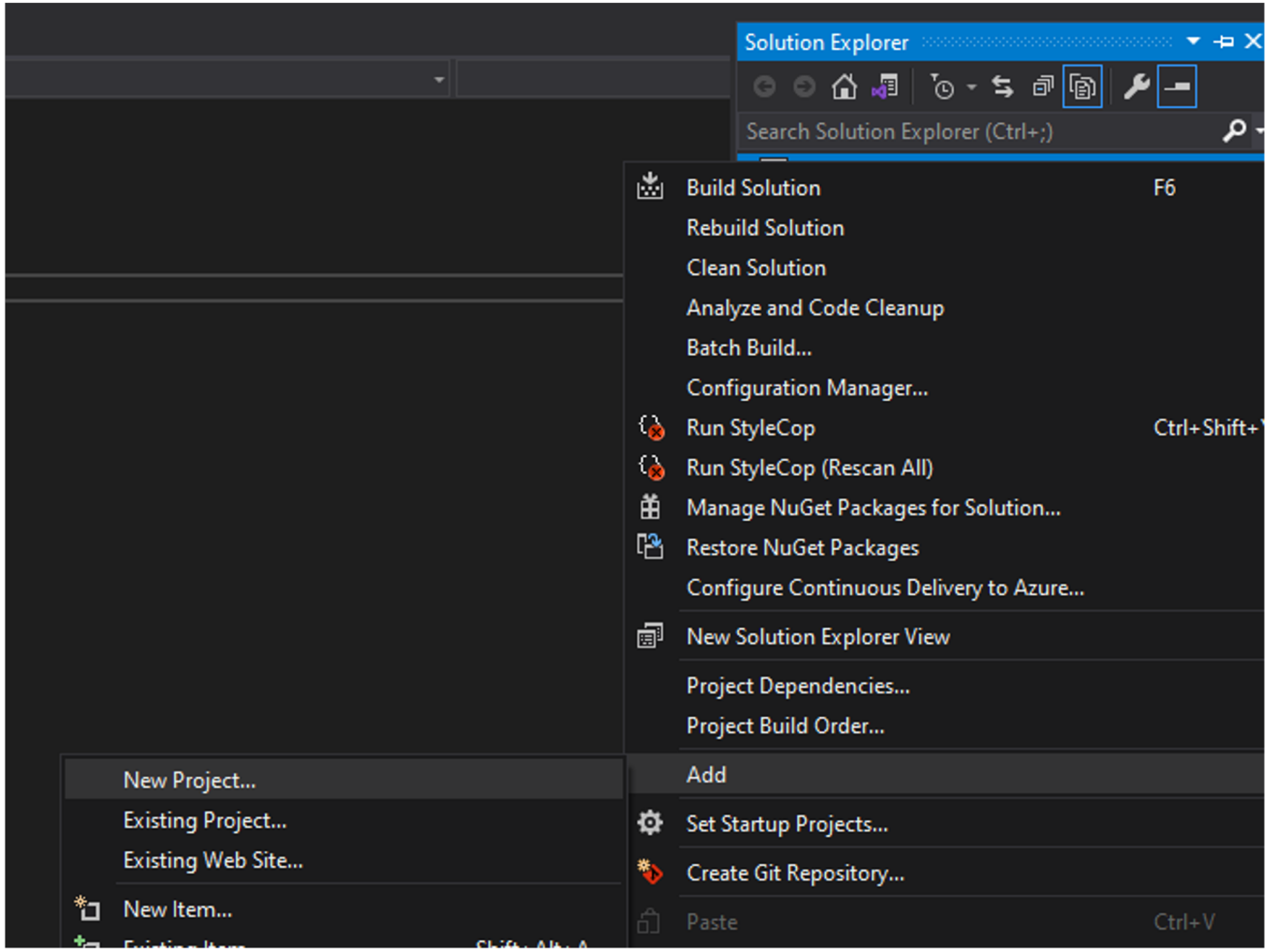
Open Visual studio 2019, click on Create a new project, select WCF Service Application and click Next.

Create project in WCF



Give Project name and create Project

Let’s add layers in Project



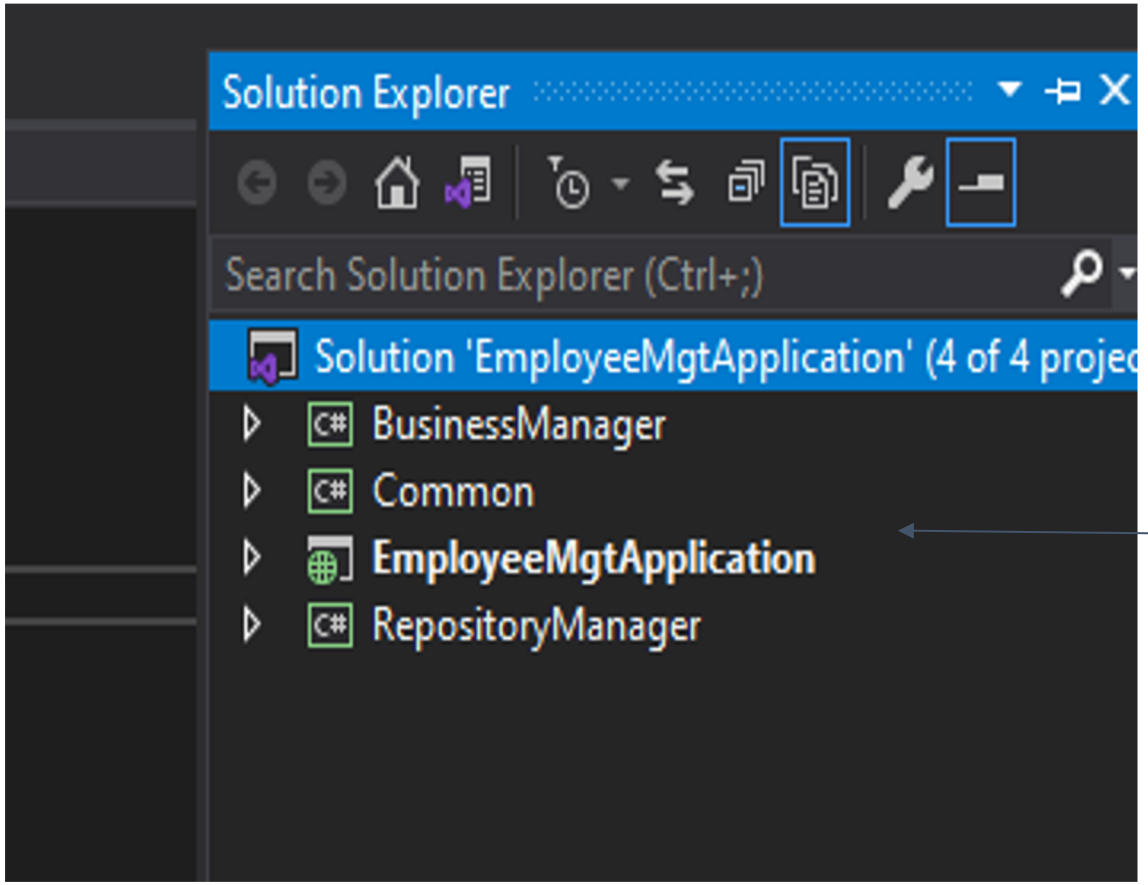
Right click on Solution of your project from Solution explorer.

Now click on ->Add -> New Project

Select Class Library(.Net framework), from project template and click on Next

Give Project Name as BusinessManager and Create Project.

Create different Layers



Repeat same steps from previous slides and create 2 new projects in same solution, name them “Common” and “RepositoryManager”

So now your project structure will look like this

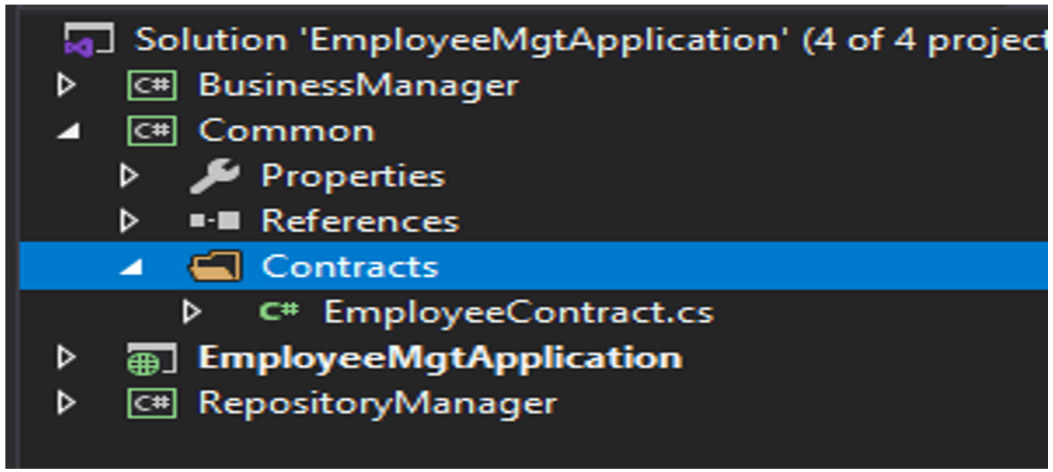
Employee Management system

Here in this application we will perform CRUD operation and for database mapping we will use Entity Framework (Database first approach)

We will start implementing from RepositoryManager -> BusinessManager -> Wcf

Add EmployeeContract in Common Project

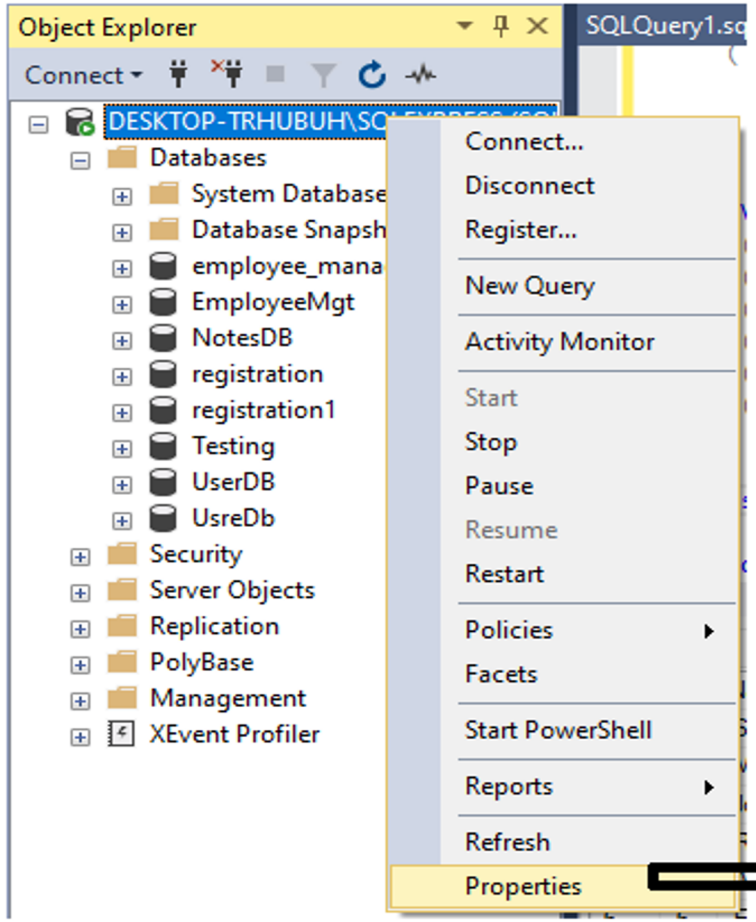
```
[DataContract]
0 references
public class EmployeeContract
{
    [DataMember]
    0 references
    public int Id { get; set; }

    [DataMember]
    0 references
    public string Name { get; set; }

    [DataMember]
    0 references
    public int Salary { get; set; }

    [DataMember]
    0 references
    public string Email { get; set; }
}
```

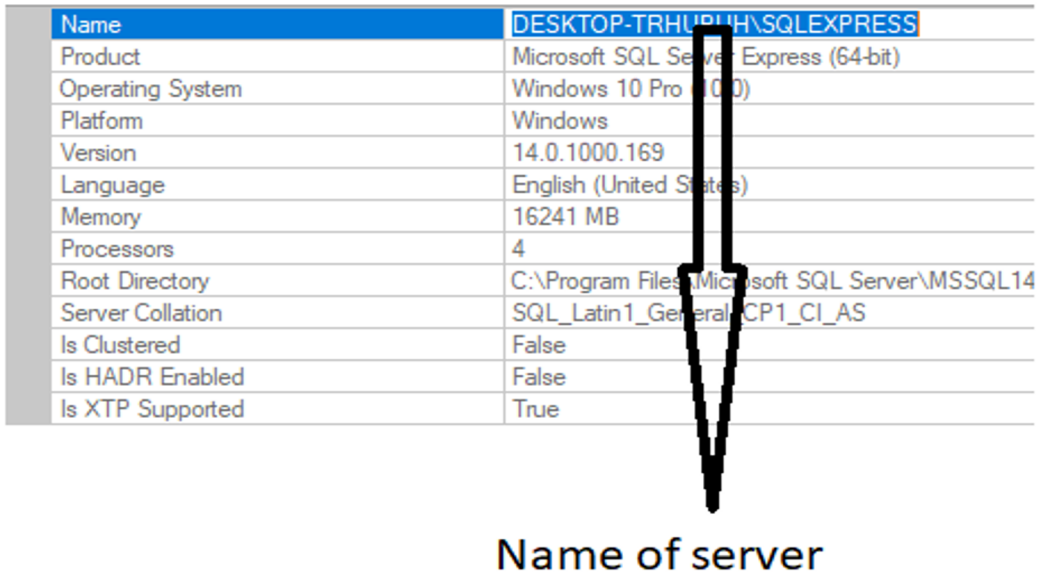


Create Contracts folder in Common Project, right click on Contracts folder and Add New Class and name it as EmployeeContract.cs.

Once class is added add code in it as shown on left side, for DataContract and DataMember it will show error so add namespace “using System.Runtime.Serialization”

Get server name from SSMS



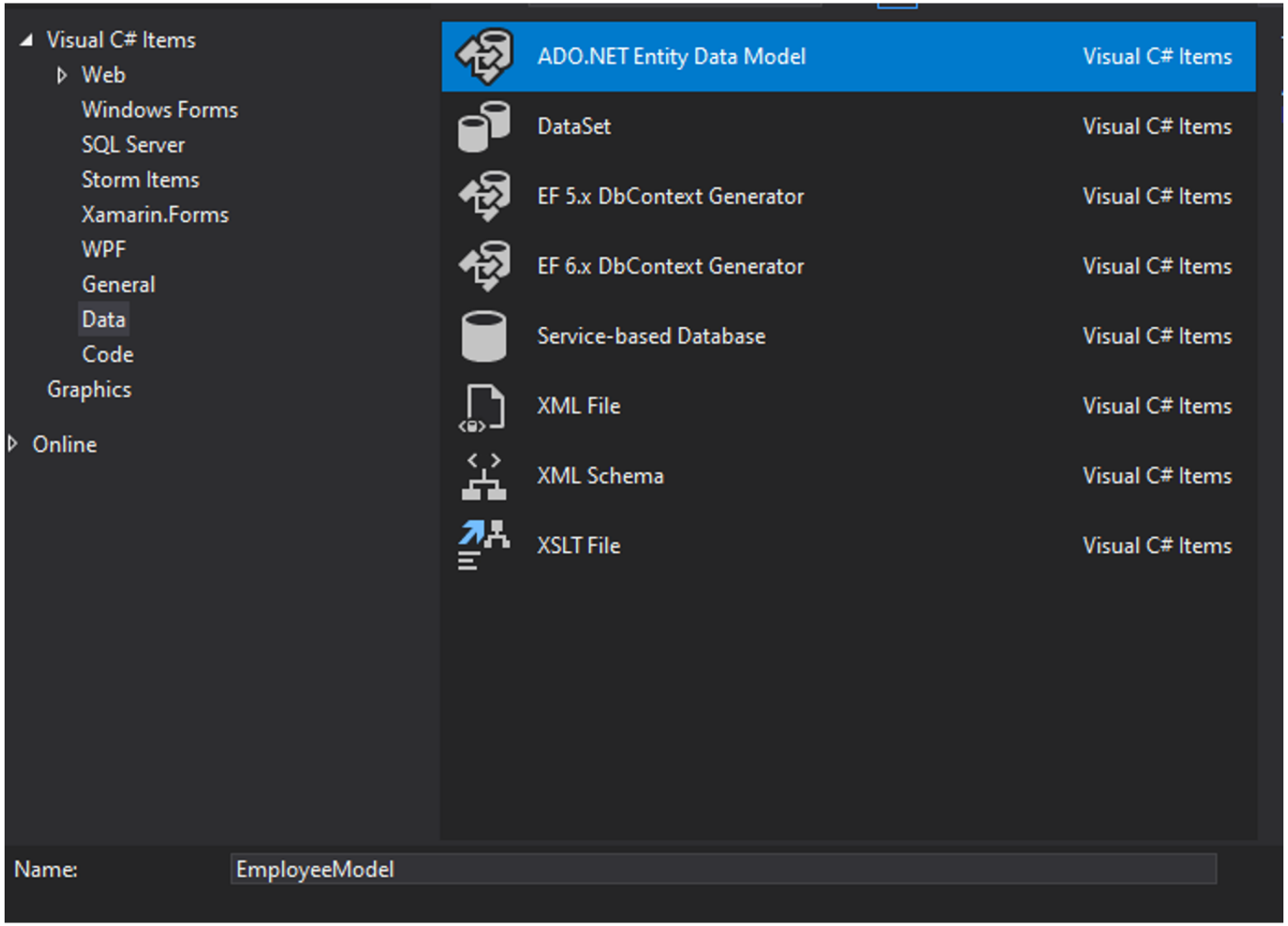


Name of server

Click on properties and copy Name

Go to SSMS, click on properties and copy Name of server

Database Mapping in RepositoryManager



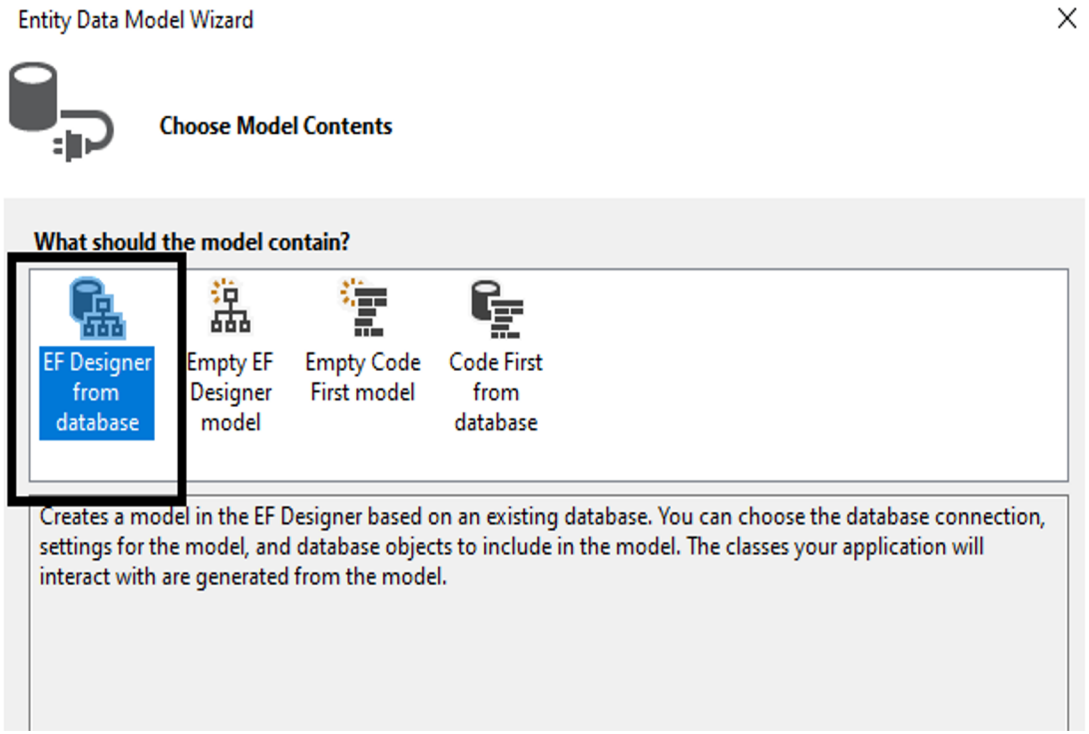
Create Model folder in RepositoryManager.
Right click on Model and click on
Add -> New Item.

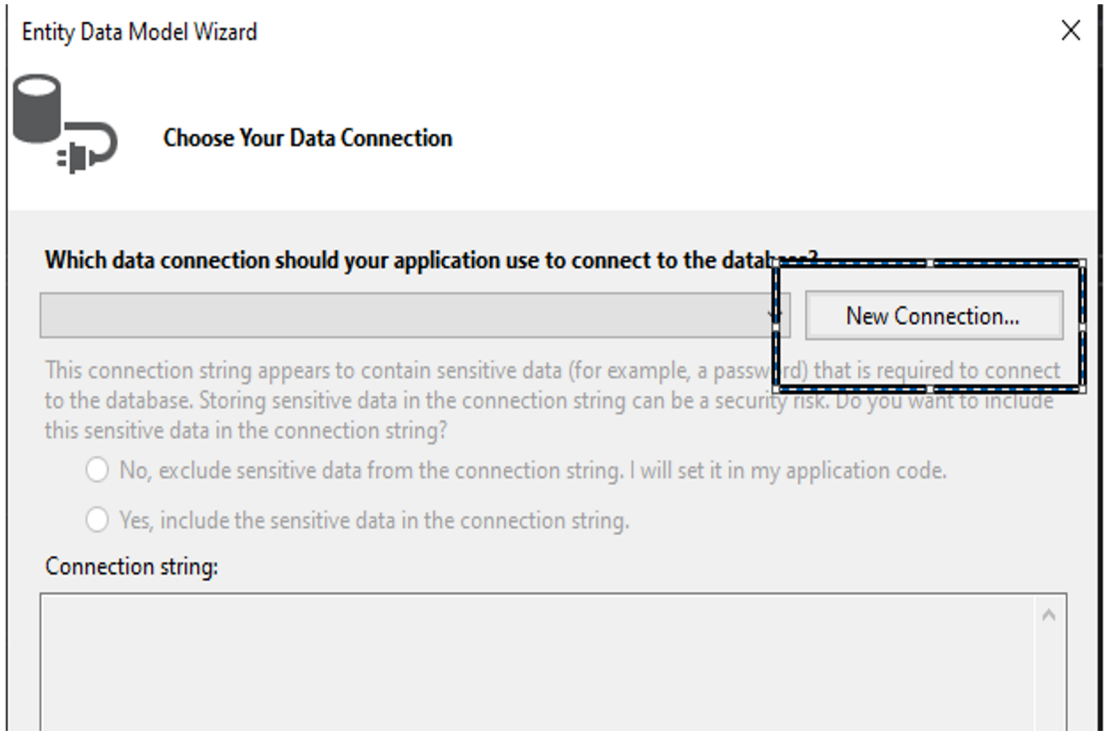
Now dialogue box will appear as shown in image.

Select Data from Visual c# items and Select ADO.Net Entity Data Model.

Name it EmployeeModel and click on Add button.

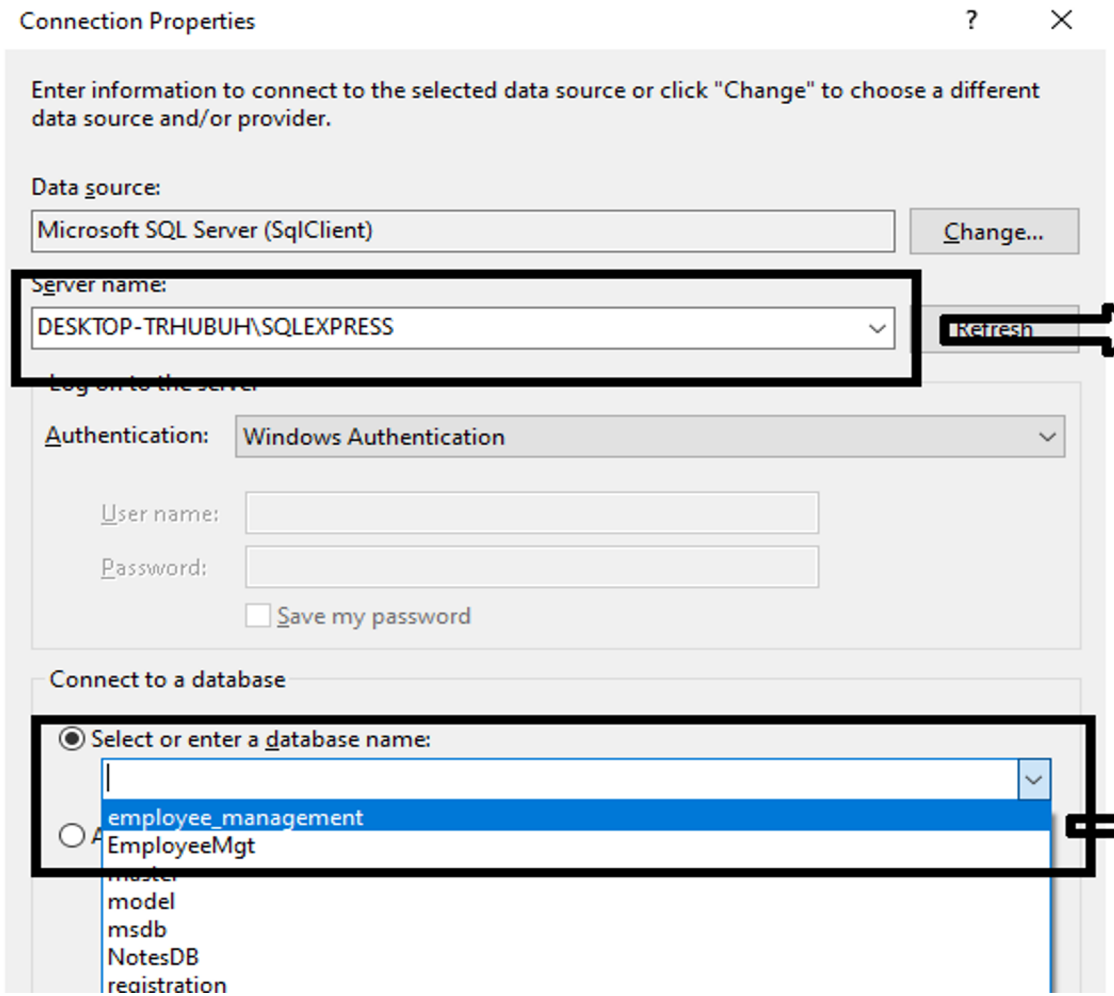
Entity Data model Configuration





Click on EF Designer From Database and click Next -> than click on New Connection
(Continue from last slide)

Configure Database



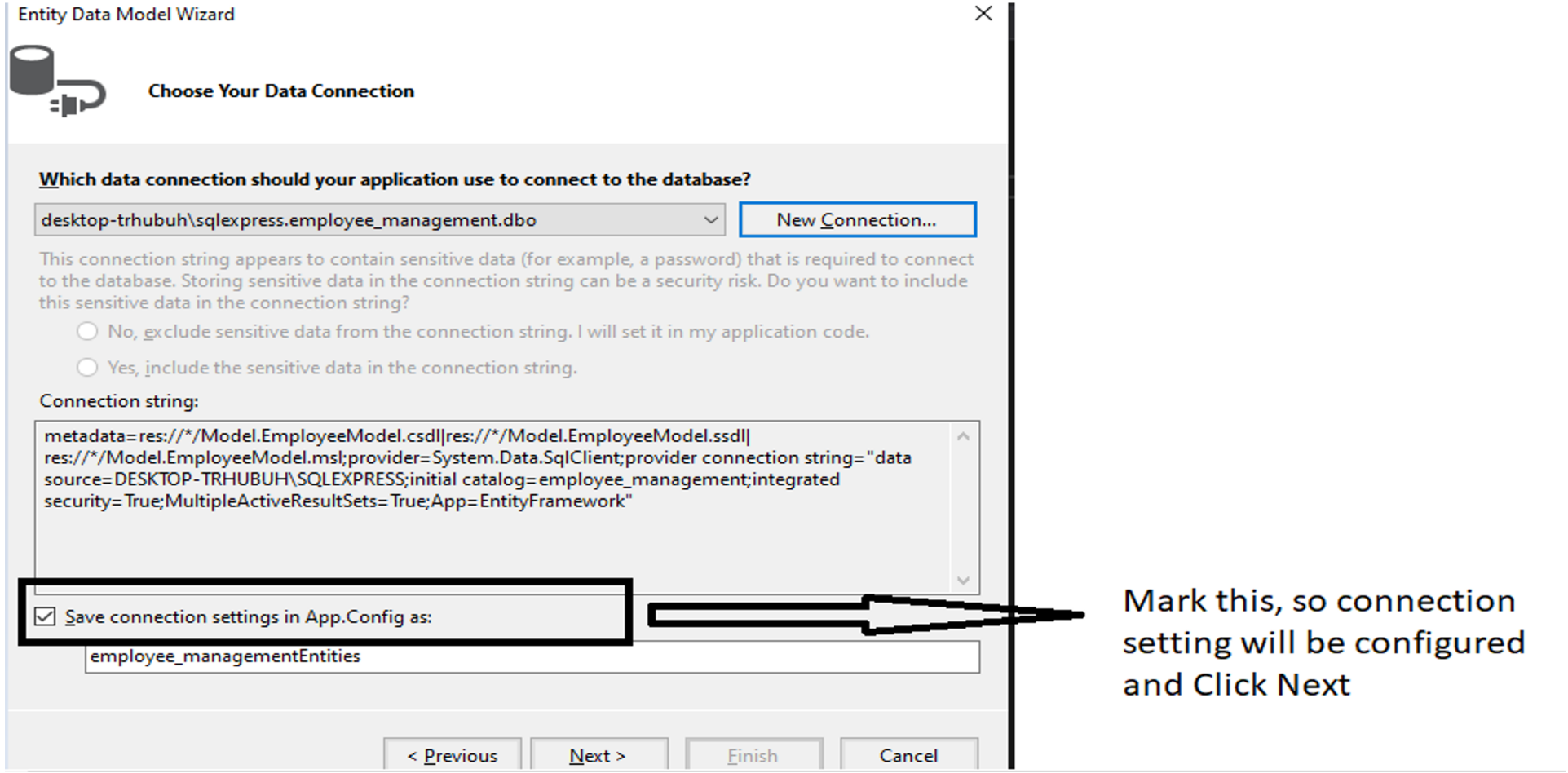
Paste server name which you copied earlier

Select database and click on OK

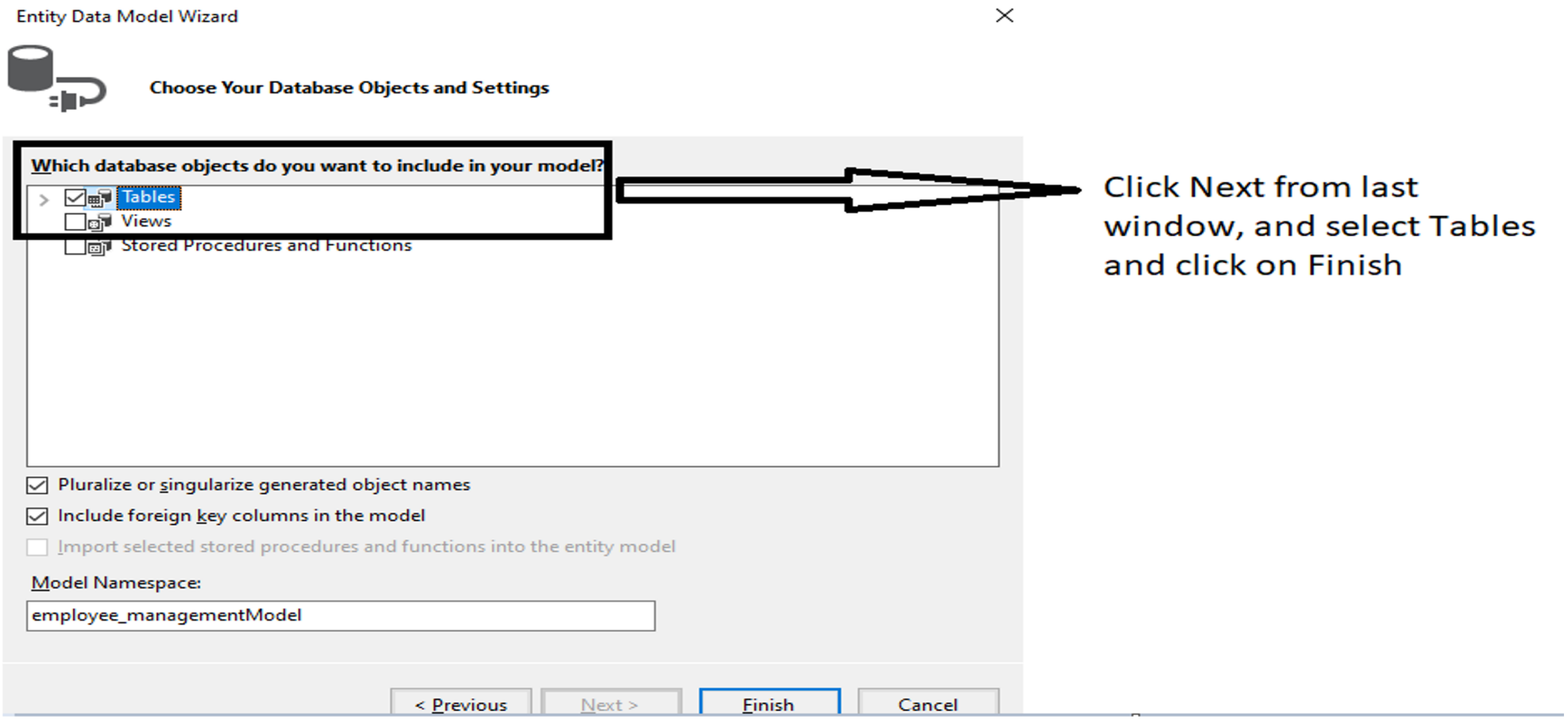
Paste server name, which was copied from SSMS.

Database will be loaded, select appropriate database

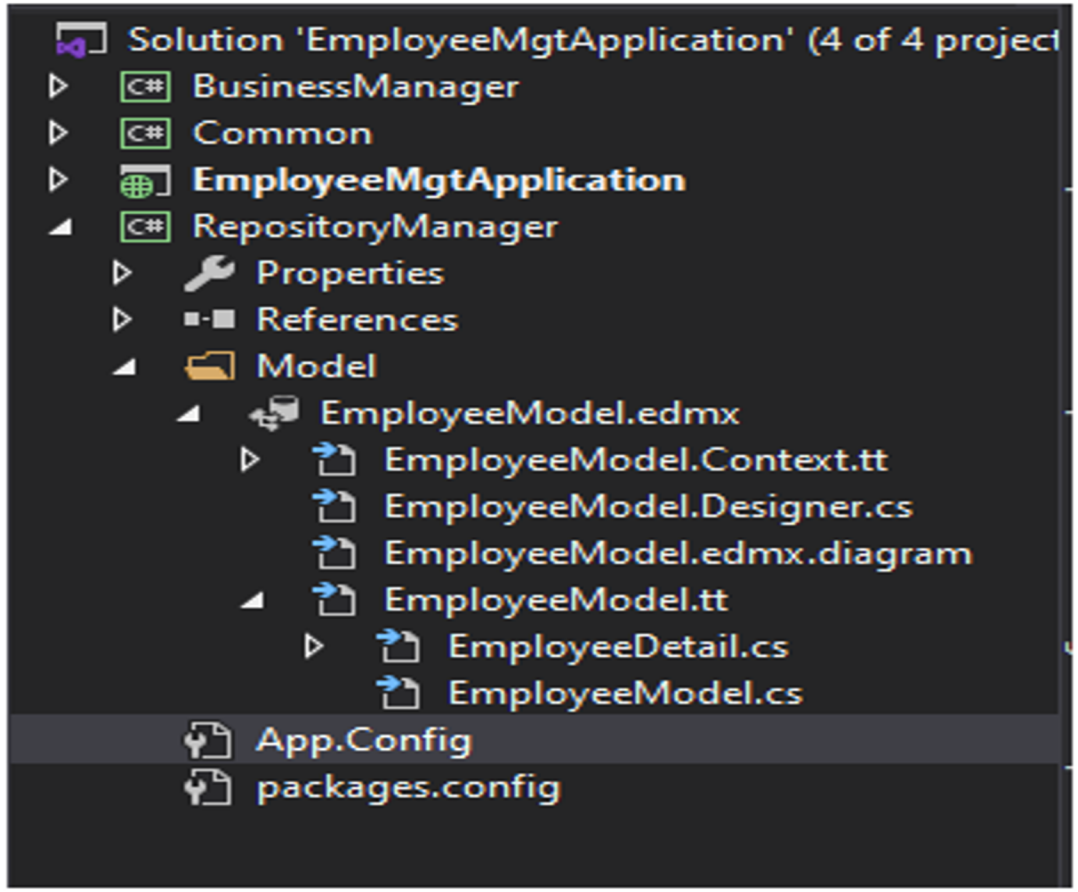
Entity data model mapping



Choose your database objects and settings

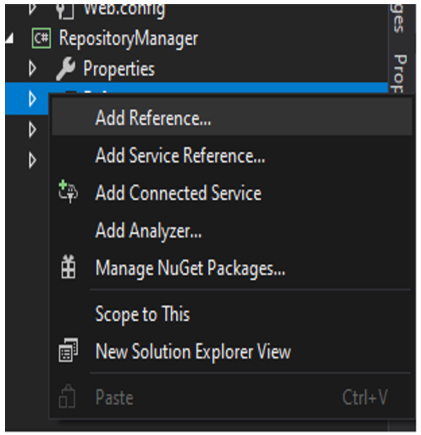


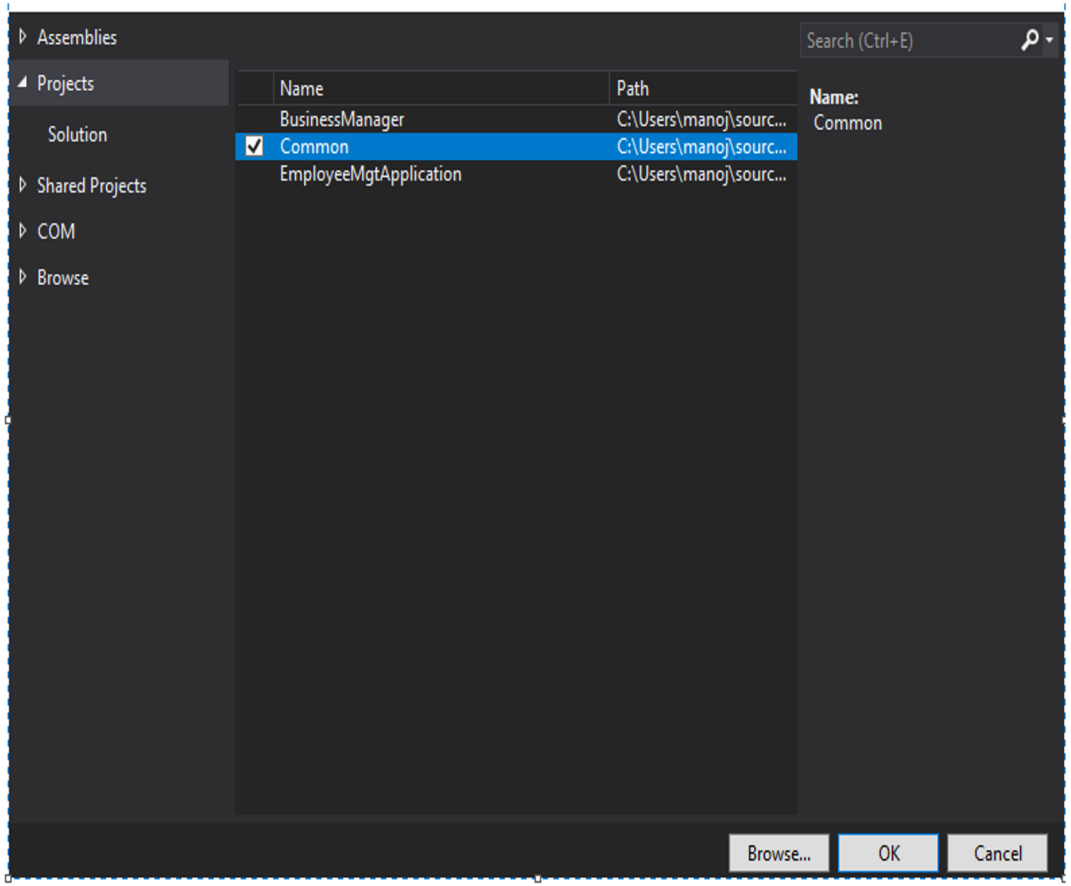
Folder structure after Database mapping



After adding Model in Repository manager, folder structure will look like this, and new App.Config file will be Added, which will have database connection settings

Add Projects as a reference





Right click on Reference of Project, Add reference, mark project which you want to add as reference and click OK.

Repository Manager will have Common project in reference.

BusinessManager will have Common and RepositoryManager in reference

WCF project(Here EmployeeMgtApplication) will have BusinessManager and Common in Refernce

BridgeLabz
Employability Delivered

UC 1

Ability to Retrieve all Employees in EmployeePayroll DB

- Use WCF and LINQ to save the EmployeePayroll Data of id, name, and salary
- Retrieve all the Employee records using WCF REST API

BridgeLabz
Employability Delivered

UC 2

Ability to Retrieve single Employee in EmployeePayroll DB

- Use WCF and LINQ to save the EmployeePayroll Data of id, name, and salary
- Retrieve an Employee records by employee id

BridgeLabz
Employability Delivered

UC 3

Ability to add a new Employee to the EmployeePayroll DB

- Use WCF and LINQ to save the EmployeePayroll Data of id, name, and salary
- Ability to add using WCF REST API and LINQ and then on success add to Employee Payroll
- Validate with the successful Count

BridgeLabz
Employability Delivered

UC 4

Ability to add multiple Employee to the EmployeePayroll DB

- Use WCF and LINQ to add multiple Employees to Payroll
- Ability to add using WCF REST API and then on success add to EmployeePayrollService
- Validate with the successful Count

BridgeLabz
Employability Delivered

UC 5

Ability to Update Salary in Employee Payroll DB

- Update the Salary in DB using WCF REST API and LINQ
- Update the salary by using employee id

BridgeLabz
Employability Delivered

UC 6

Ability to Delete Employee from Employee Payroll DB

- Use WCF REST API and LINQ to delete the employee by using ID

Start implementing RepositoryManager

```
2 references
public interface IEmployeeRepository
{
    2 references
    IList<EmployeeContract> GetAllEmployee();

    2 references
    int AddEmployee(EmployeeContract employeeContract);

    2 references
    int UpdateEmployee(EmployeeContract employeeContract, int EmpId);

    2 references
    EmployeeContract GetById(int empId);

    2 references
    int DeleteEmployee(int empId);

    2 references
    EmployeeContract GetEmployeeByEmail(string email);
}
```

Add Folder in RepositoryManager and Name it as Interface, right click and new item as interface and name it as IEmployeeRepository and declare all methods as mentioned

Implement IEmployeeRepository interface

```
employee_managementEntities employeeManagementEntitiesObj;
public EmployeeRepository()
{
    employeeManagementEntitiesObj = new employee_managementEntities();
}
```

Get connection string name from App.config file and create reference as mentioned

```
2 references
public IList<EmployeeContract> GetAllEmployee()
{
    var query = (from a in employeeManagementEntitiesObj.EmployeeDetails select a).Distinct();
    List<EmployeeContract> employeeData = new List<EmployeeContract>();

    query.ToList().ForEach(x =>
    {
        employeeData.Add(new EmployeeContract
        {
            Id = x.Id,
            Name = x.Name,
            Email = x.Email,
            Salary = x.Salary
        });
    });

    return employeeData;
}
```

Write GetAllEmployee method as mentioned

Implementation of EmployeeRepository

```
2 references
public int AddEmployee(EmployeeContract employeeContract)
{
    EmployeeDetail employee = new EmployeeDetail()
    {
        Name = employeeContract.Name,
        Email = employeeContract.Email,
        Salary = employeeContract.Salary
    };
    employeeManagementEntitiesObj.EmployeeDetails.Add(employee);
    return employeeManagementEntitiesObj.SaveChanges();

}
```

AddEmployee method in EmployeeRepository

```
2 references
public int UpdateEmployee(EmployeeContract employeeContract, int EmpId)
{
    EmployeeDetail employee = employeeManagementEntitiesObj
        .EmployeeDetails.Find(EmpId);
    if(employee != null)
    {
        emplcyee.Email = emplcyeeContract.Email;
        emplcyee.Name = employeeContract.Name;
        emplcyee.Salary = employeeContract.Salary;
        return employeeManagementEntitiesObj.SaveChanges();
    }
    else
    {
        throw new Exception("Employee do not exists");
    }
}
```

UpdateEmployee method in EmployeeRepository

Implementation of IEmployeeRepository

```
2 references
public EmployeeContract GetById(int empId)
{
    var Employee = employeeManagementEntitiesObj.EmployeeDetails
        .Find(empId);
    EmployeeContract employeeContract = new EmployeeContract()
    {
        Name = Employee.Name,
        Email = Employee.Email,
        Salary = Employee.Salary,
        Id = Employee.Id
    };
    return employeeContract;
}
```

GetById method in EmployeeRepository

```
2 references
public int DeleteEmployee(int empId)
{
    var employee = (from a in employeeManagementEntitiesObj.EmployeeDetails
                    where a.Id == empId select a).FirstOrDefault();
    if(employee!=null)
    {
        employeeManagementEntitiesObj.EmployeeDetails.Remove(employee);
        return employeeManagementEntitiesObj.SaveChanges();
    }
    else
    {
        return 0;
    }
}
```

Delete method in EmployeeRepository

Implementation of IEmployeeRepository

```
2 references
public EmployeeContract GetEmployeeByEmail(string email)
{
    EmployeeDetail employeeDetail = (from a in employeeManagementEntitiesObj.EmployeeDetails
                                     where a.Email == email select a).FirstOrDefault();
    if(employeeDetail != null)
    {
        EmployeeContract employeeContract = new EmployeeContract()
        {
            Name = employeeDetail.Name,
            Email = employeeDetail.Email,
            Salary = employeeDetail.Salary,
            Id = employeeDetail.Id
        };
        return employeeContract;
    }

    return null;
}
```

GetEmployeeByEmail method in EmployeeRepository

Start implementing BusinessManager

```
2 references
public interface IEmployeeBusiness
{
    2 references
    IList<EmployeeContract> GetAllEmployee();

    2 references
    string AddEmployee(EmployeeContract employeeContract);

    2 references
    string UpdateEmployee(EmployeeContract employeeContract, int EmpId);

    2 references
    EmployeeContract GetById(int empId);

    2 references
    string DeleteEmployee(int empId);
}
```

Add Folder in BusinessManager and Name it as Interface, right click and new item as interface and name it as IEmployeeBusiness and declare all methods as mentioned

Implementation of IEmployeeBusiness

```
private readonly IEmployeeRepository employeeRepository;

1 reference
public EmployeeBusiness()
{
    employeeRepository = new EmployeeRepository();
}
2 references
public IList<EmployeeContract> GetAllEmployee()
{
    IList<EmployeeContract> employeeContracts = employeeRepository.GetAllEmployee();
    if (employeeContracts != null)
    {
        return employeeContracts;
    }
    else
    {
        return new List<EmployeeContract>();
    }
}
2 references
```

Create Reference of IEmployeeRepository interface and initialize implementation of it

Write GetAllEmployee method in EmployeeBusiness which will interact with EmployeeRepository

Implementation of IEmployeeBusiness

```
public string AddEmployee(EmployeeContract employeeContract)
{
    if(employeeRepository.GetEmployeeByEmail(employeeContract.Email) != null )
    {
        return "Employee already registered, please try with other email id";
    }

    if (employeeRepository.AddEmployee(employeeContract) == 1)
    {
        return "Employee Added successfully";
    }
    else
    {
        return "Employee not able to add.";
    }
}

2 references
public string UpdateEmployee(EmployeeContract employeeContract, int EmpId)
{
    if (employeeRepository.UpdateEmployee(employeeContract, EmpId) == 1)
    {
        return "Employee updated successfully";
    }
    else
    {
        return "Employee not updated";
    }
}
```

Implement AddEmployee method in EmployeeBusiness which will interact with EmployeeRepository

Implement UpdateEmployee method in EmployeeBusiness which will interact with EmployeeRepository

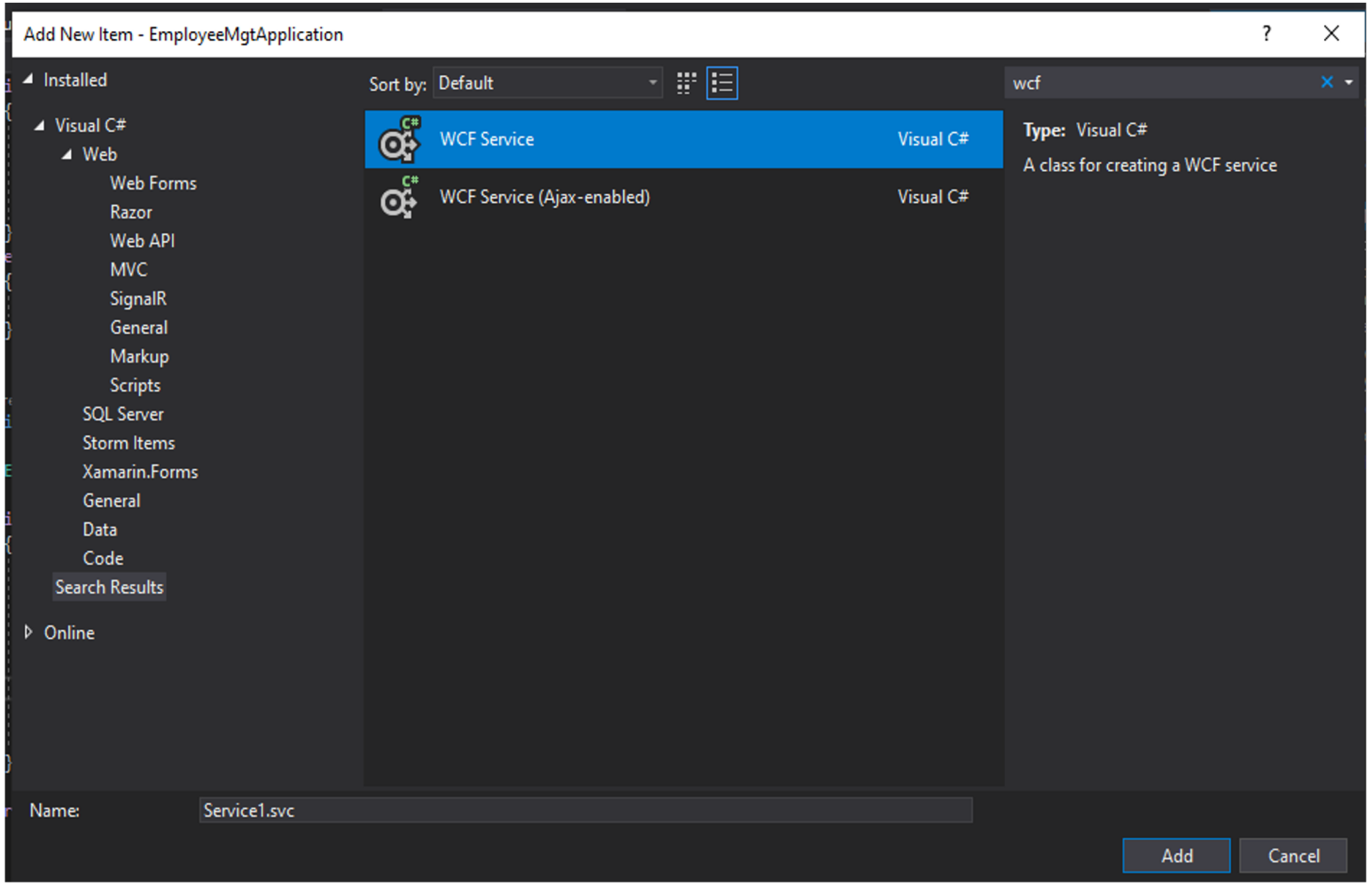
Implementation of IEmployeeBusiness

```
2 references
public EmployeeContract GetById(int empId)
{
    EmployeeContract employeeContract = employeeRepository.GetById(empId);
    if (employeeContract != null)
    {
        return employeeContract;
    }
    else
    {
        return new EmployeeContract();
    }
}

2 references
public string DeleteEmployee(int empId)
{
    if(employeeRepository.DeleteEmployee(empId) == 1)
    {
        return "Employee deleted successfuly";
    }
    else
    {
        return "Employee does not exists.";
    }
}
```

Implement GetById method in EmployeeBusiness

Implement DeleteEmployee method in EmployeeBusiness

Start implementing EmployeeMgtApplication(Wcf project)



Right click on WCF project and Add Wcf Service with name EmployeeService, it will add 2 new files IEmployeeService.cs and EmployeeService.svc

Interface for WCF project

```
[ServiceContract]
1 reference
public interface IEmployeeService
{
    [OperationContract]
    [WebInvoke(Method = "PUT",
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/Update/{EmpId}")]
    1 reference
    string UpdateEmployee(EmployeeContract employeeContract, string empId);

    [OperationContract]
    [WebInvoke(Method = "GET",
        RequestFormat = WebMessageFormat.Json,
        ResponseFormat = WebMessageFormat.Json,
        UriTemplate = "/GetAllEmployee/")]
    1 reference
    IList<EmployeeContract> GetAllEmployee();

    [OperationContract]
    [WebInvoke(Method = "POST",
        RequestFormat = WebMessageFormat.Json,
        ResponseFormat = WebMessageFormat.Json,
        UriTemplate = "/Add/")]
    1 reference
    string AddEmployee(EmployeeContract employeeContract);
```

```
[OperationContract]
[WebInvoke(Method = "GET",
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json,
UriTemplate = "/get/{EmpId}")]
1 reference
EmployeeContract GetById(string EmpId);

[OperationContract]
[WebInvoke(Method = "DELETE",
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json,
UriTemplate = "/delete/{EmpId}")]
1 reference
string DeleteEmployee(string EmpId);
```

Let's understand code including REST Template.

IEmployeeService: REST Service interface name

Method: HTTP methods types it can be GET,POST,PUT,DELETE and other

UriTemplate: To Define url structure that how can be service method accessed at client.

BodyStyle: Allows to define message body style format such as Bare, Wrapped etc.

RequestFormat: Defines in which message format does request come from client such xml or json.

ResponseFormat: Defines what message format does service return to the client as a part of response such as xml or json.

Configuration in Web.config file

```
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configurati
    <section name="entityFramework" type="System.Data.Entity.
  </configSections>
  <connectionStrings>
    <add name="employee_managementEntities" connectionString=
  </connectionStrings>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infras
    <providers>
      <provider invariantName="System.Data.SqlClient" type="S
    </providers>
  </entityFramework>
```

Copy this code from App.config file from RepositoryManager and paste in Web.config file of Wcf project

And Add EntityFramework nuget package

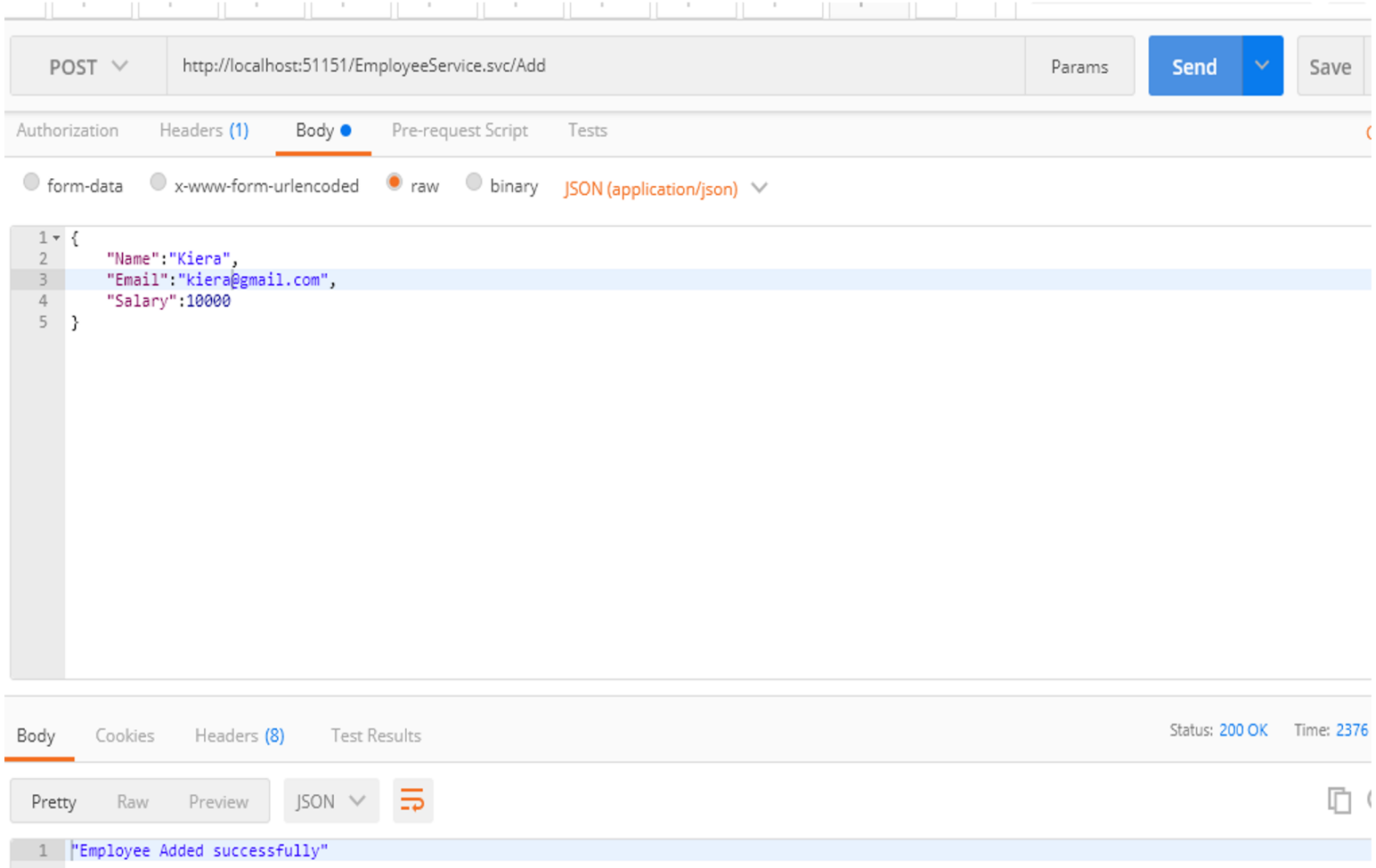
Configuration in web.config file

```
<behaviors>
  <serviceBehaviors>
    <behavior name="ServiceBehavior">
      <!-- To avoid disclosing metadata information, set the values below to false before deployment -->
      <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
      <!-- To receive exception details in faults for debugging purposes, set the value below to true.  Set to false bef
      <serviceDebug includeExceptionDetailInFaults="false"/>
    </behavior>
  </serviceBehaviors>

  <endpointBehaviors>
    <behavior name="web">
      <webHttp/>
    </behavior>
  </endpointBehaviors>

</behaviors>

<services>
  <service name="EmployeeMgtApplication.EmployeeService" behaviorConfiguration="ServiceBehavior">
    <endpoint binding="webHttpBinding" contract="EmployeeMgtApplication.IEmployeeService" behaviorConfiguration="web">
    </endpoint>
  </service>
</services>
```

Set True httpGetEnabled

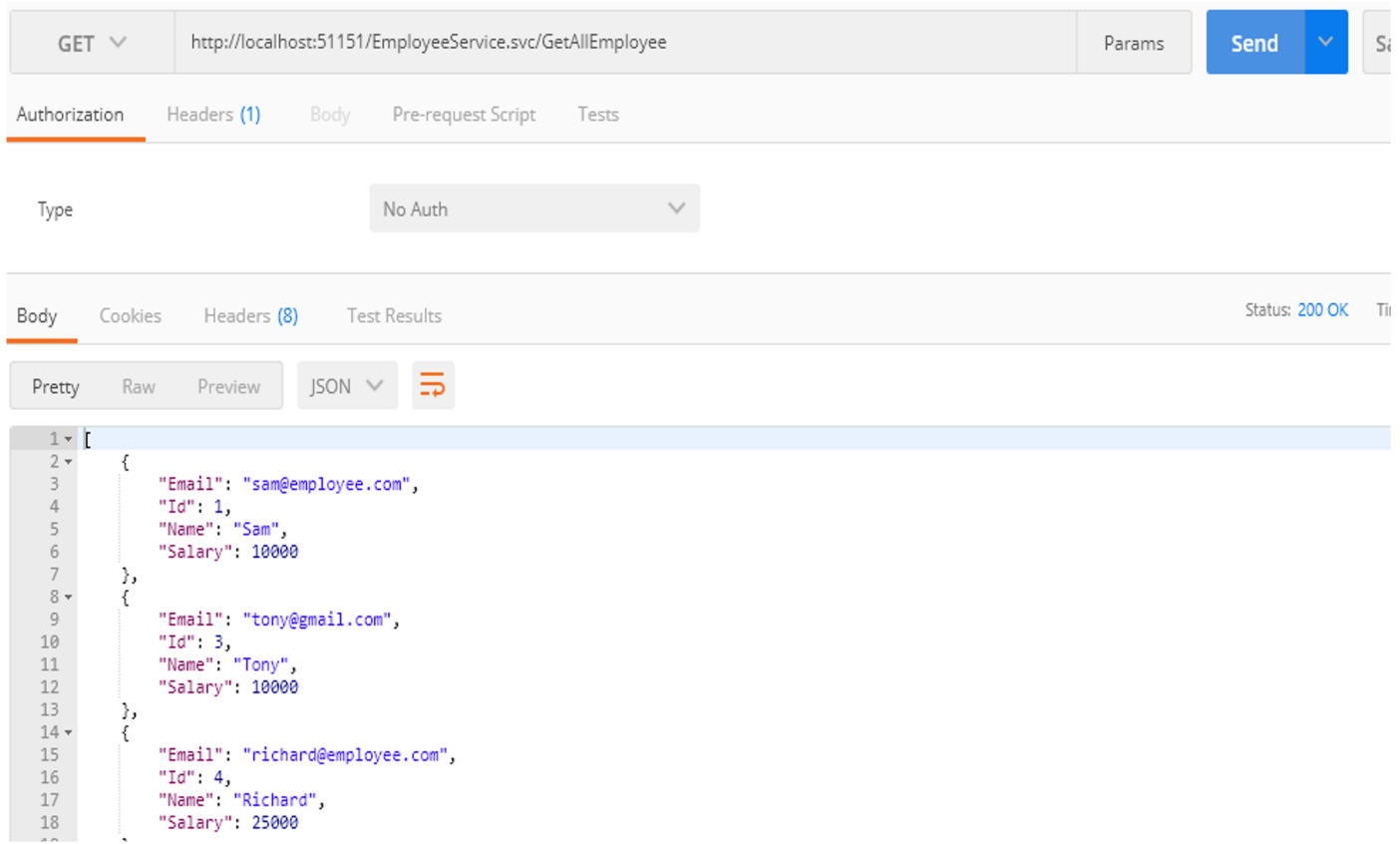
Configure End Point

Lets test using Postman - Add Functionality



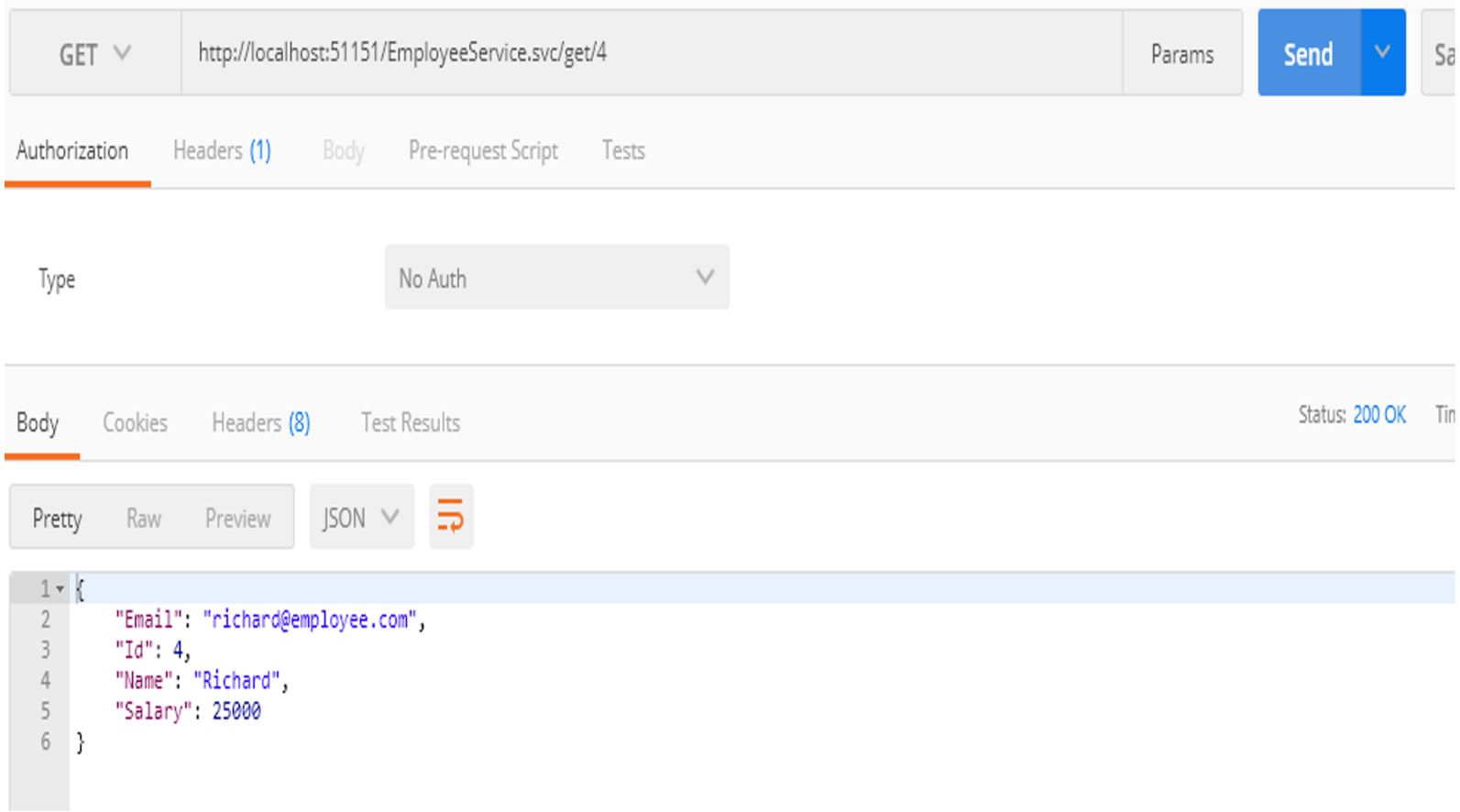
http://localhost:51151/EmployeeService.svc/Add

GetAllEmployees Functionality



http://localhost:51151/EmployeeService.svc/GetAllEmployee

GetById functionality



http://localhost:51151/EmployeeService.svc/get/4

BridgeLabz

Employability Delivered

Thank You