Book Api – full steps

אפיון השירות

סעיף 1:

יש ליצור טבלאות מתאימות ב db עבור שמירת נתוני ה authors וה books

- Author
  - Name -string
  - Age - int
  - Image -string
  -
- book
  - BookName - string
  - Price - decimal
  - Author - number

דגשים:

- יש ליצור מפתח ראשי של id לכל טבלה - המפתח יקודם בצורה אוטומטית בכל הוספת רשומה.
- יש ליצור את ה Name של ה Author עם הגדרה של ערך ייחודי
- יש ליצור את ה BookName של Book עם הגדרה של ערך ייחודי

סעיף 2:

יש ליצור את שכבת ה DAL על ידי EF כך שיוכל לגשת ל DB - מסעיף 1

סעיף 3:

יש ליצור dll בשם **BO** המכיל את המחלקות הבאות:

- מחלקה המייצגת Author
  - Name - (string - min 3 max 20)
  - Age - (int between 18-120)
  - Image - (string - min 5 chars)

- מחלקה המייצגת book

- BookName - (string - min 2 max 15)
- Price - (decimal - between 30 -200)
- Author - (Author class object)

סעיף 4:

יש ליצור dll בשם BLL המכיל קישור ל DAL (מסעיף 2) ול BO (מסעיף 3),

בתוך ה BLL יש ליצור שני מחלקות:

- AuthorManager - with crud to Author table (use EF from DAL)
- BookManager - with crud to Book table (use EF from DAL)

סעיף 5:

יש ליצור פרוייקט של web-api המכיל קישור ל BLL (מסעיף 4) ול BO (מסעיף 3),

בתוך ה web-apiיש ליצור שני controllers :

AuthorController - with the following actions:

- Get - without parameters - will return array of Author (the class is defined in the BO),

  The web-api will call the AuthorManager in the BLL

  The BLL will create an object of the EF from the DAL, and pass all the authors to the web-api

- Get - with id parameter (int) - will return an Aouthor (the class is defined in the BO),

  The web-api will call the AuthorManager in the BLL

  The BLL will create an object of the EF from the DAL, and pass the author with the required id to the web-api

- Post - with an Aouthor parameter (the class is defined in the BO) - will return a boolean value.

  The web-api will call the AuthorManger in the BLL

  The BLL will create an object of the EF from the DAL

  The BLL will try to add the author to the db, and return to the web api a boolean value that indicates if the action has completed successfully or not.

- put - with id parameter (int) and an Aouthor parameter (the class is defined in the BO) - will return a boolean value

  The web-api will call the AuthorManger in the BLL

  The BLL will create an object of the EF from the DAL

  The BLL will try to edit the author in the db, and then return to the web api a boolean value that indicates if the action has completed successfully or not.

- delete - with id parameter (int) - will return a boolean value

  The web-api will call the AuthorManger in the BLL

  The BLL will create an object of the EF from the DAL

  The BLL will try to delete the author from the db, and then return to the web api a boolean value that indicates if the action has completed successfully

BookController - with the following actions:

- Get - without parameters - will return array of Book (the class is defined in the BO),

  The web-api will call the BookManager in the BLL

  The BLL will create an object of the EF from the DAL, and pass all the books to the web-api

- Get - with id parameter (int) - will return an Book (the class is defined in the BO),

  The web-api will call the BookManager in the BLL

  The BLL will create an object of the EF from the DAL, and pass the book with the required id to the web-api

- Post - with an Book parameter (the class is defined in the BO) - will return a boolean value.

  The web-api will call the BookManger in the BLL,

  The BLL will create an object of the EF from the DAL

  The BLL will try to add the book to the db, and then return to the web api a boolean value that indicates if the action has completed successfully
- put - with id parameter (int) and an Book parameter (the class is defined in the BO) - will return a boolean value

  The web-api will call the BookManger in the BLL

  The BLL will create an object of the EF from the DAL

  The BLL will try to edit the book in the db, and then return to the web api a boolean value that indicates if the action has completed successfully

- delete - with id parameter (int) - will return a boolean value

  The web-api will call the BookManger in the BLL

The BLL will create an object of the EF from the DAL

The BLL will try to delete the book from the db, and then return to the web api a boolean value that indicates if the action has completed successfully

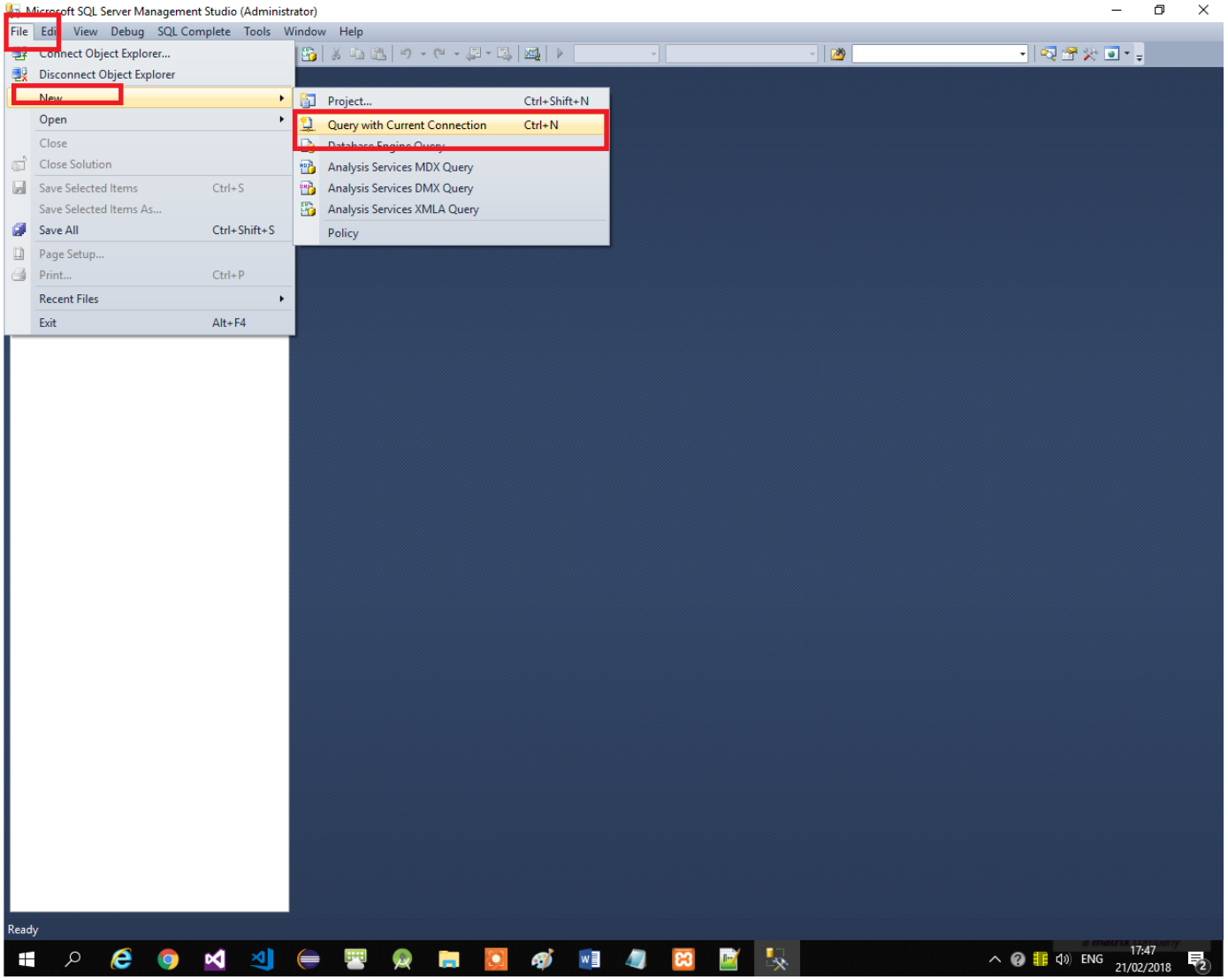
הערה: בפעולות put ו post יש לוודא ב web-api שהModel תקין - ורק אם הוא תקין לבצע קריאה לBLL

Part 1- create the DB

Step 1 – create the script to add a new db to mssql

```
USE master
GO
CREATE DATABASE BookStore
GO
```

Step 2 – run the script from step 1 in mssql



Step 3– create the script to add new tables to the db from step 1

```sql
USE BookStore
GO
CREATE TABLE [dbo].[Authors] (
    [AuthorID]   INT           IDENTITY (1, 1) NOT NULL,
    [AuthorAge] INT  NOT NULL,
    [AuthorName] NVARCHAR (20) NOT  NULL UNIQUE,
    [AuthorImage] NVARCHAR (MAX) NOT  NULL,
    CONSTRAINT [PK_Author] PRIMARY KEY ([AuthorID])
);


CREATE TABLE [dbo].[Books] (
    [BookID]     INT           IDENTITY (1, 1) NOT NULL,
    [BookName]   NVARCHAR (15) NOT NULL UNIQUE,
    [BookPrice]  DECIMAL           NOT NULL,
    [AuthorID]   INT           NOT NULL,
    CONSTRAINT [PK_Books] PRIMARY KEY ([BookID]),
    CONSTRAINT [FK_Books_ToTable] FOREIGN KEY ([AuthorID]) REFERENCES
[dbo].[Authors]([AuthorID])

);
```

Step 4 – run the script from step 3 in mssql

SQLQuery1.sql - localhost.BookStore (303_INS\jbt)

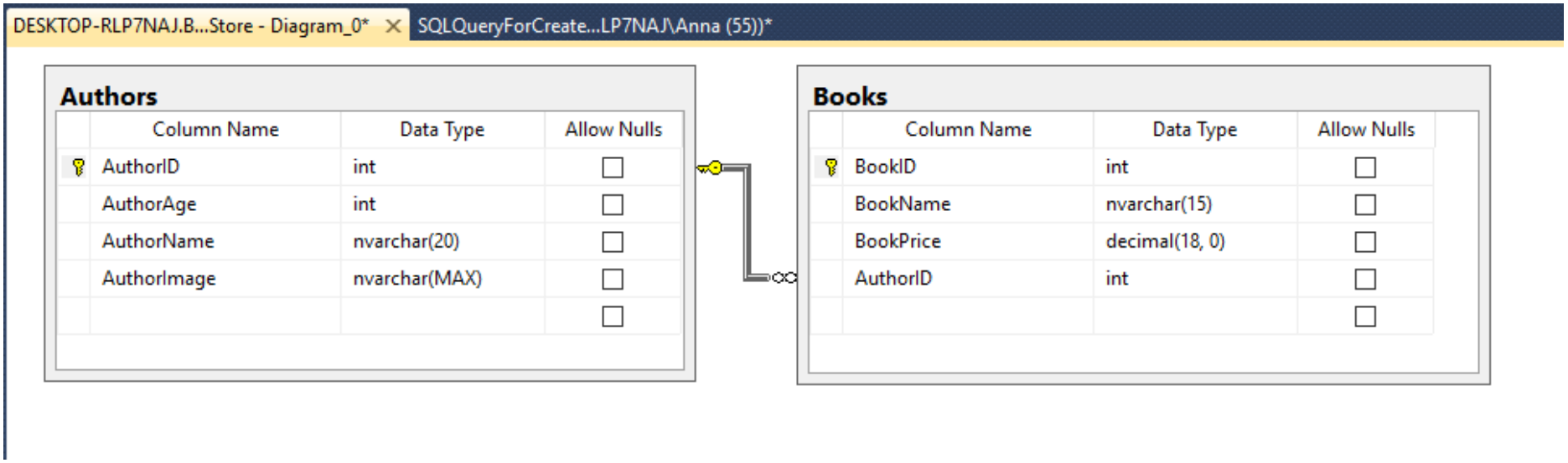
```
USE BookStore
GO
CREATE TABLE [dbo].[Authors] (
    [AuthorID]   INT           IDENTITY (1, 1) NOT NULL,
    [AuthorAge] INT  NOT NULL,
    [AuthorName] NVARCHAR (20) NOT  NULL UNIQUE,
    [AuthorImage] NVARCHAR (MAX) NOT  NULL,
    CONSTRAINT [PK_Author] PRIMARY KEY ([AuthorID])
);


CREATE TABLE [dbo].[Books] (
    [BookID]     INT           IDENTITY (1, 1) NOT NULL,
    [BookName]   NVARCHAR (15) NOT NULL UNIQUE,
    [BookPrice]  DECIMAL           NOT NULL,
    [AuthorID]   INT           NOT NULL,
    CONSTRAINT [PK_Books] PRIMARY KEY ([BookID]),
    CONSTRAINT [FK_Books_ToTable] FOREIGN KEY ([AuthorID]) REFERENCES [dbo].

);
```

Messages

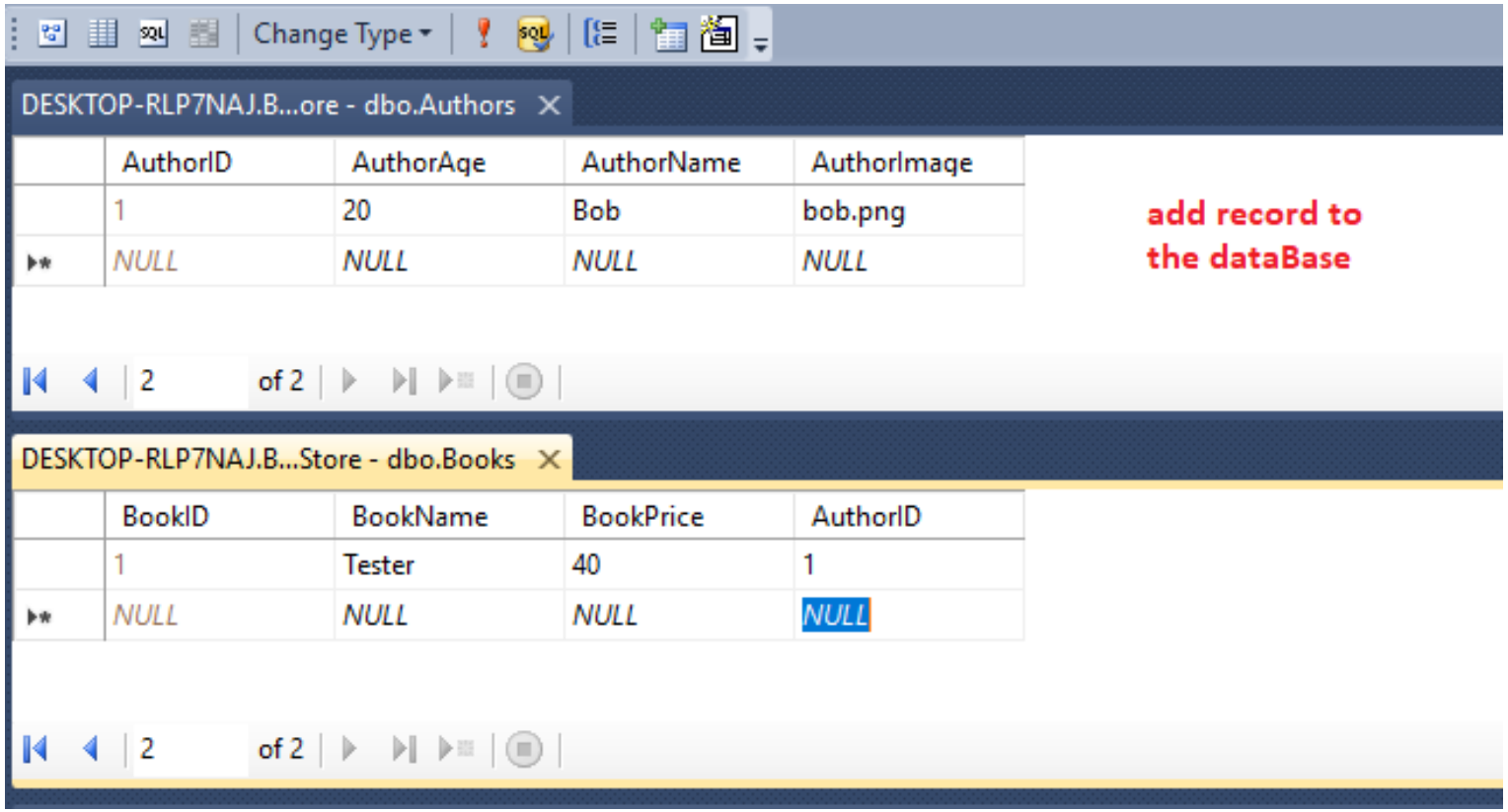
```
Command(s) completed successfully.
```

Query executed successfully. | localhost (12.0 RTM) | 303_INS\jbt (55) | BookStore | 00:00:00 | 0 rows

Step 5 – you are done!!! The DB is ready

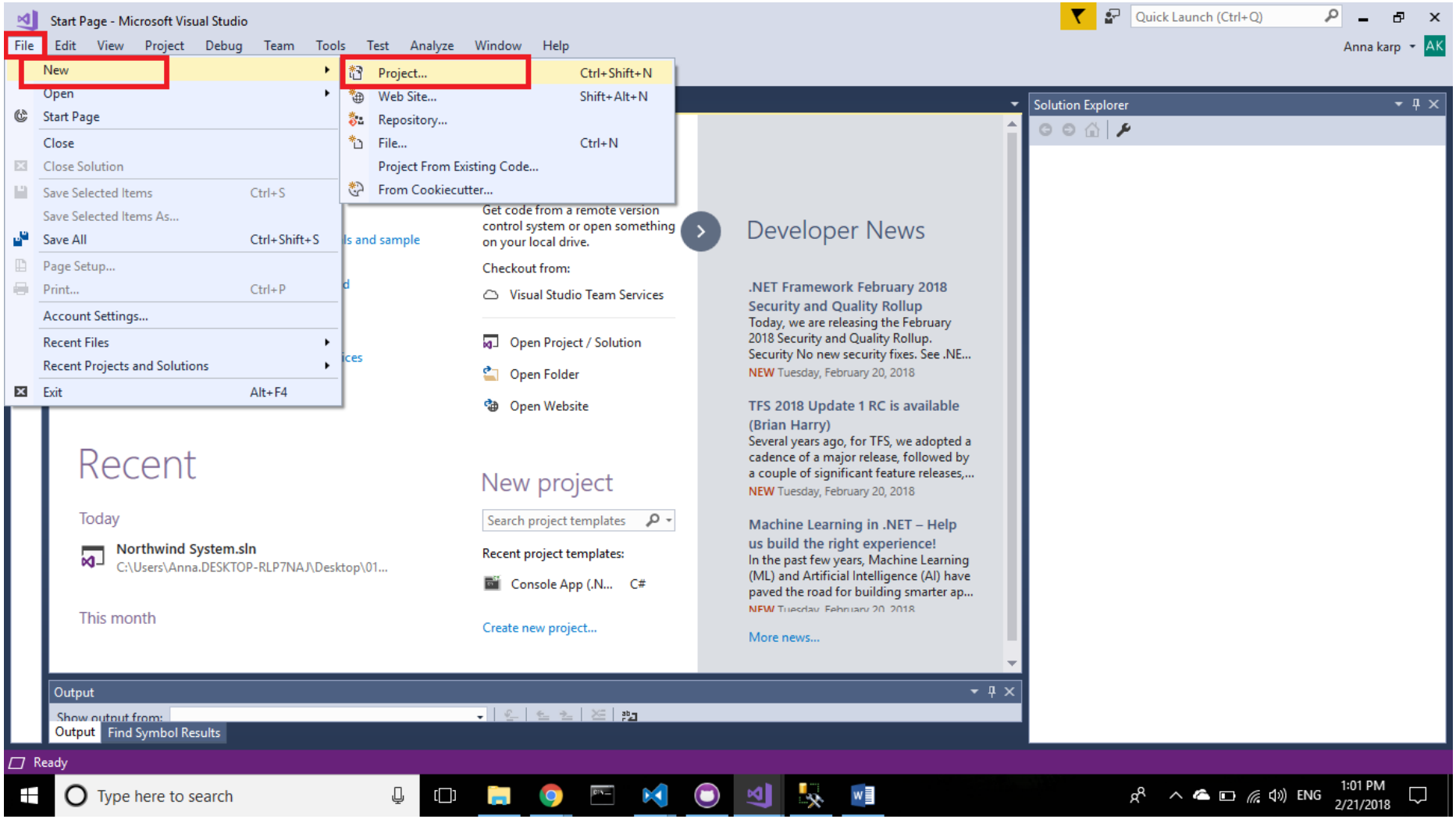


Step 6– add records to the DB

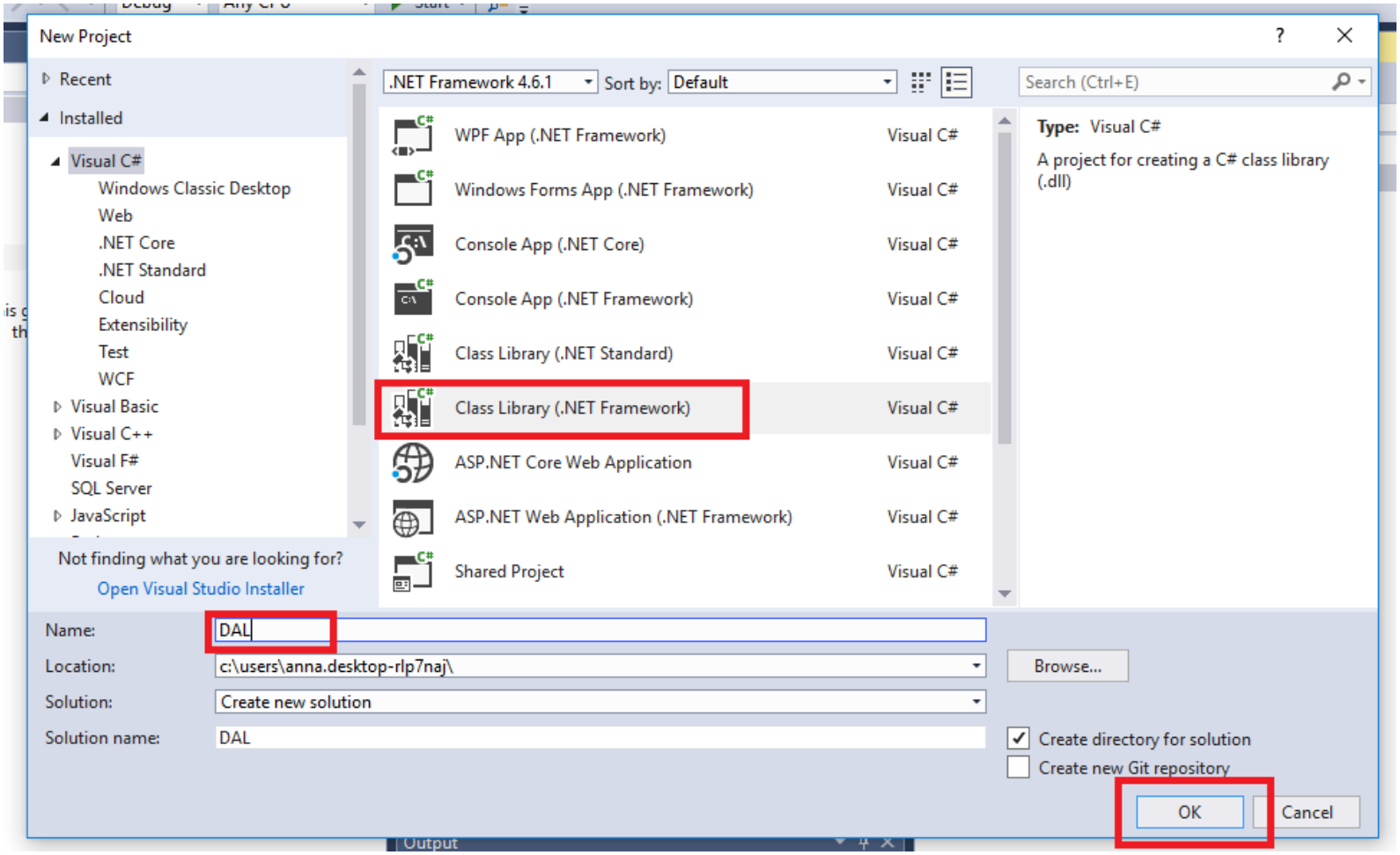


Part 2- the DAL

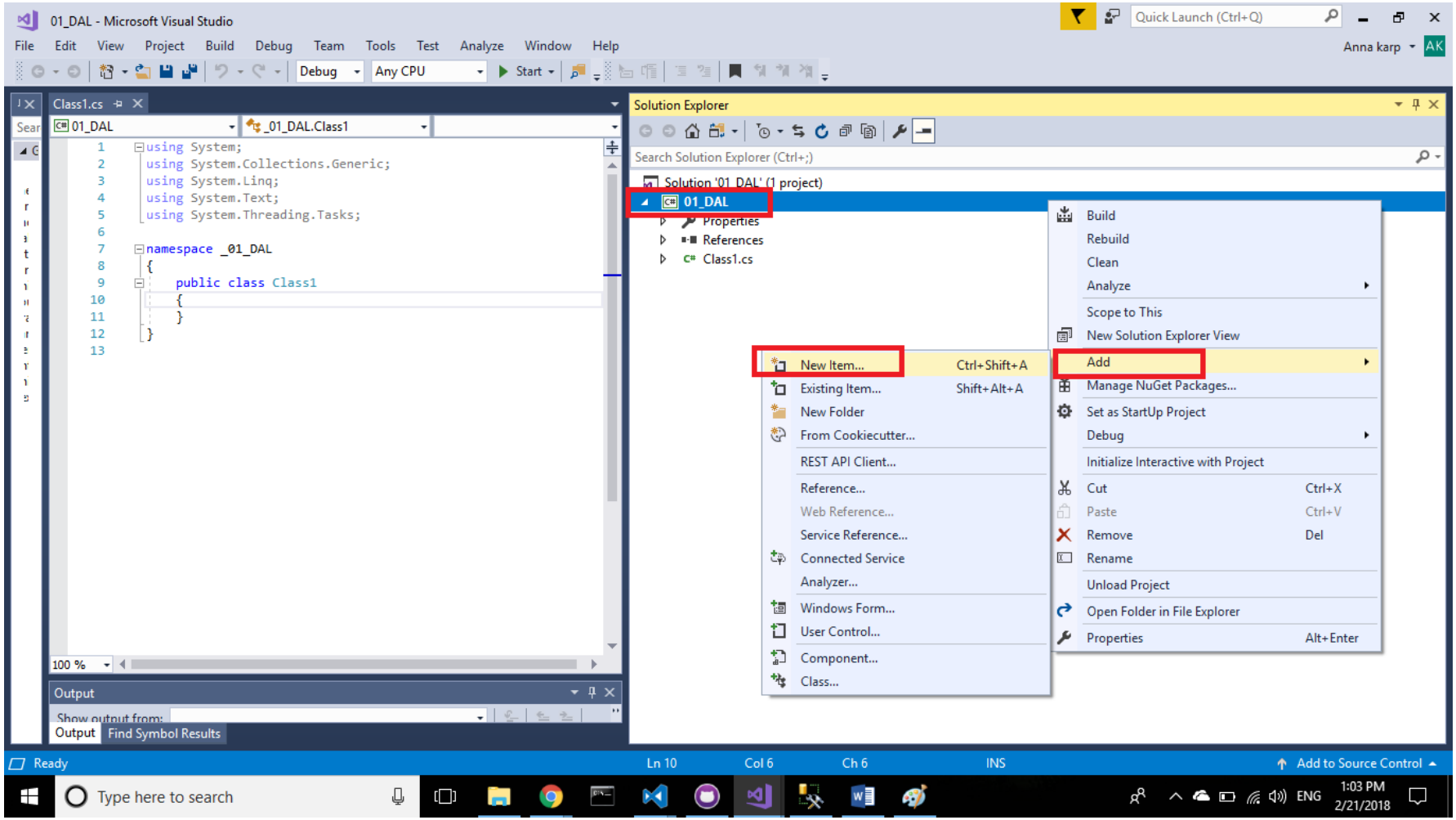
Step 1 – open a new project in visual studio



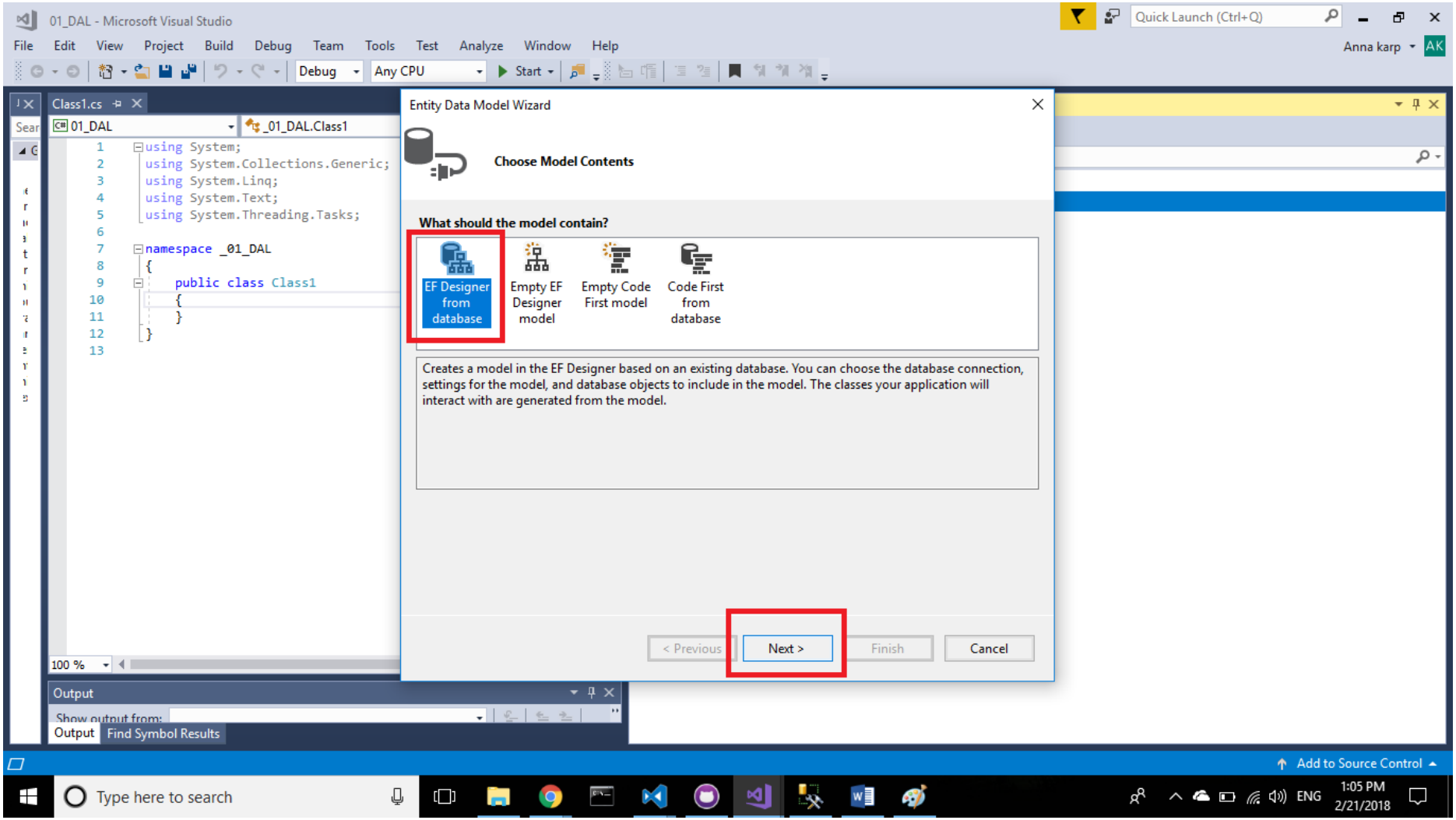
Step 2– open a new Class library project



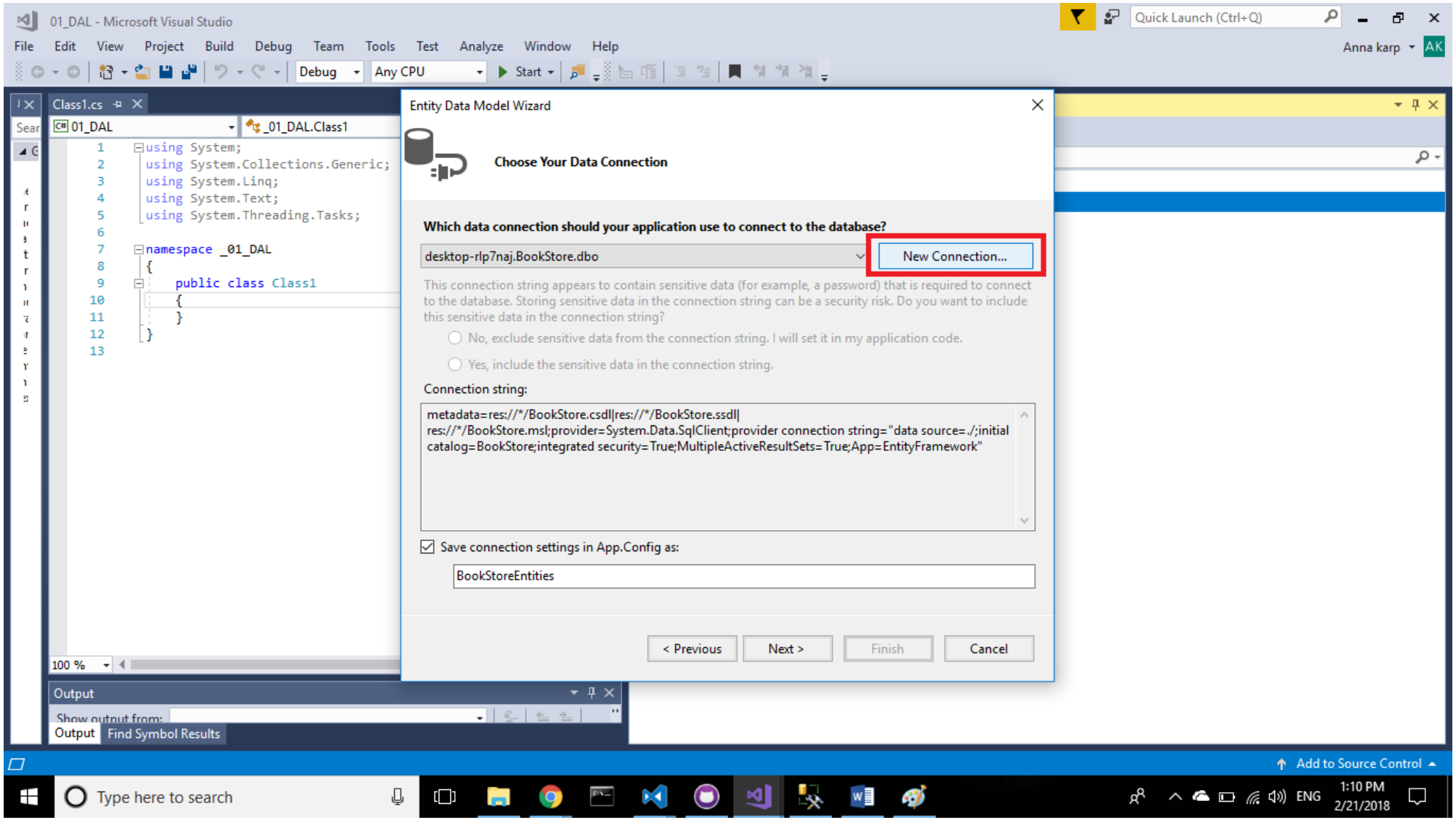
Step 3 – start adding the EF



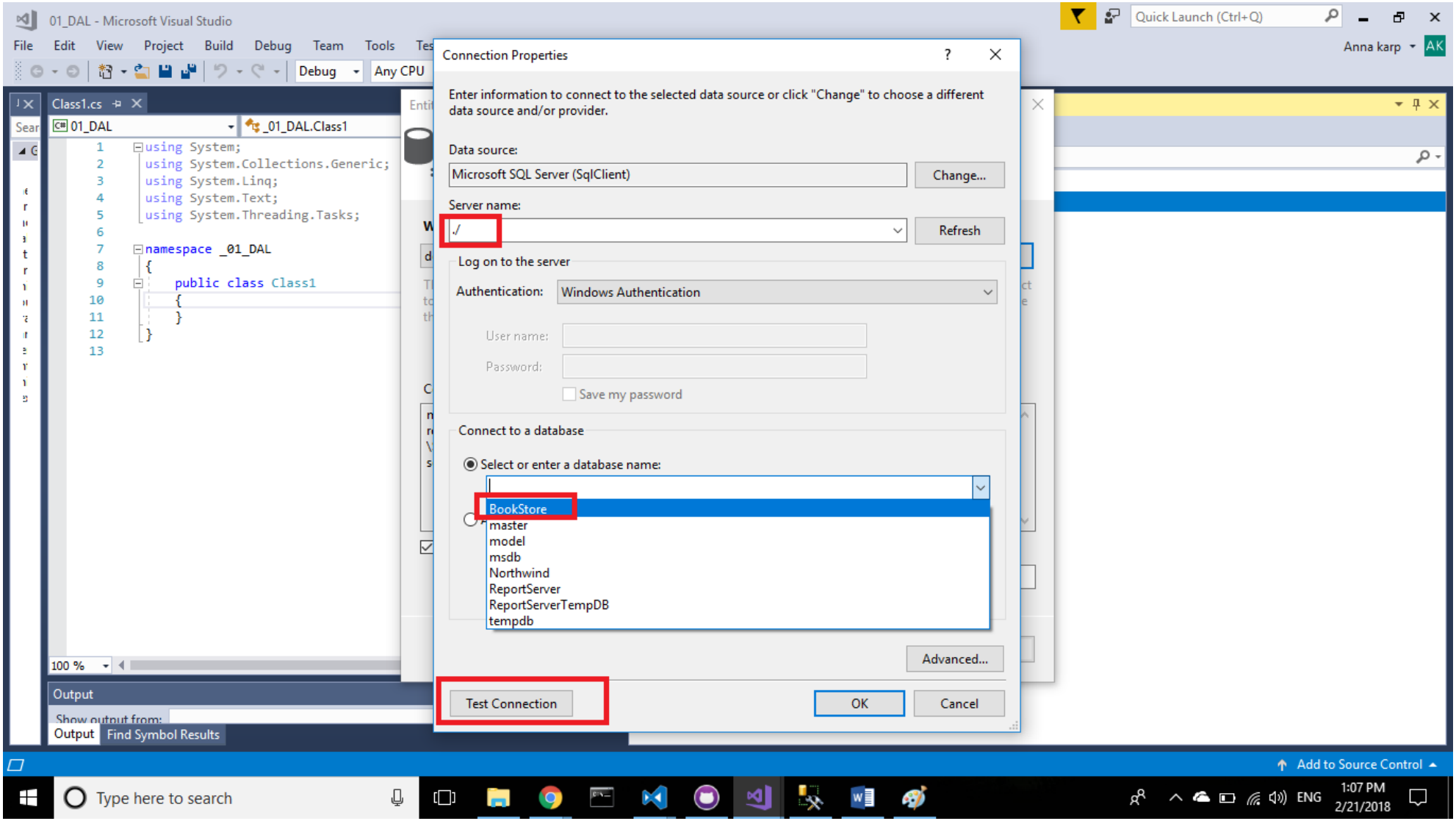
Step 4– select ef from db



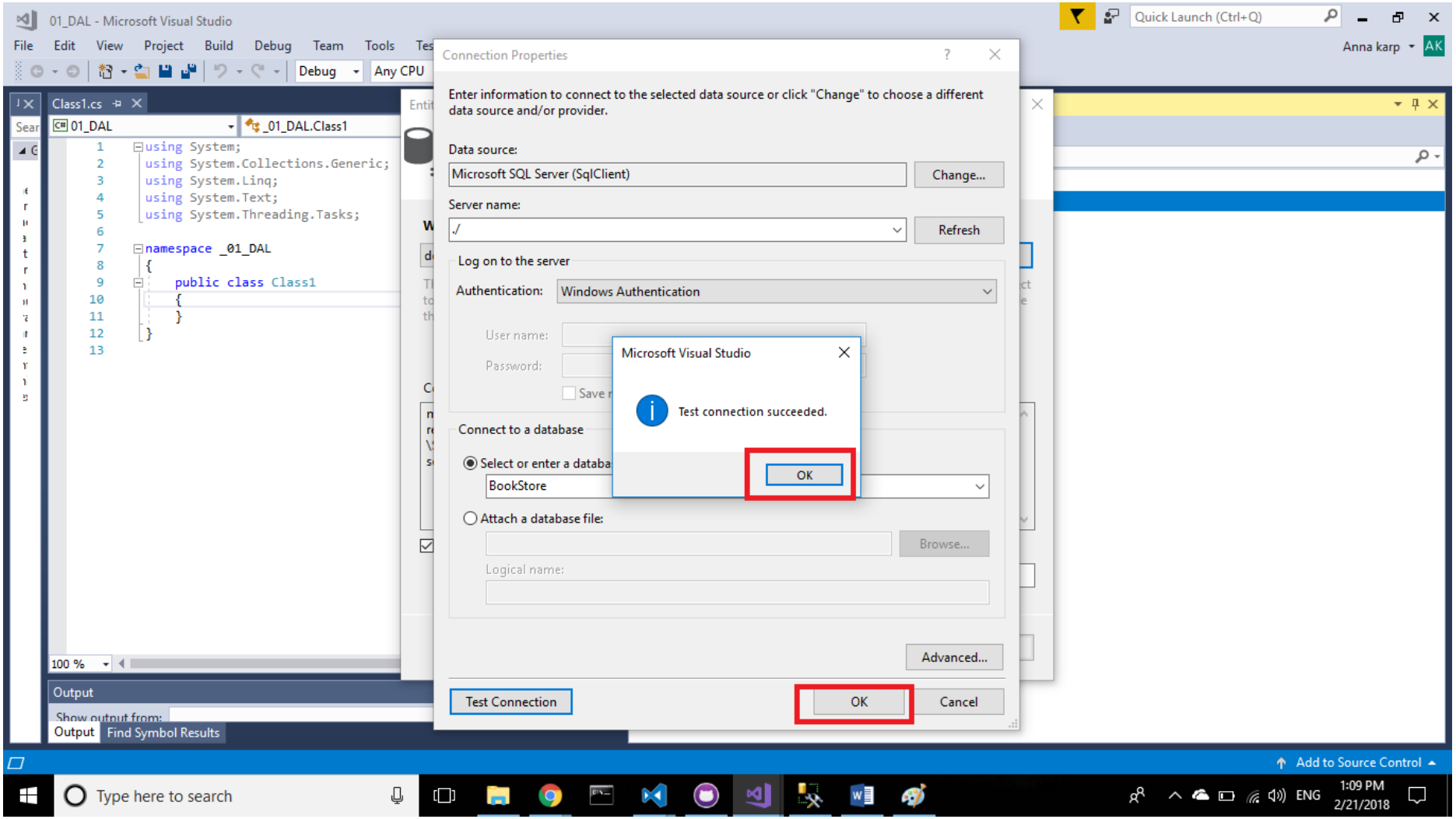
Step 5- add a new sql connection



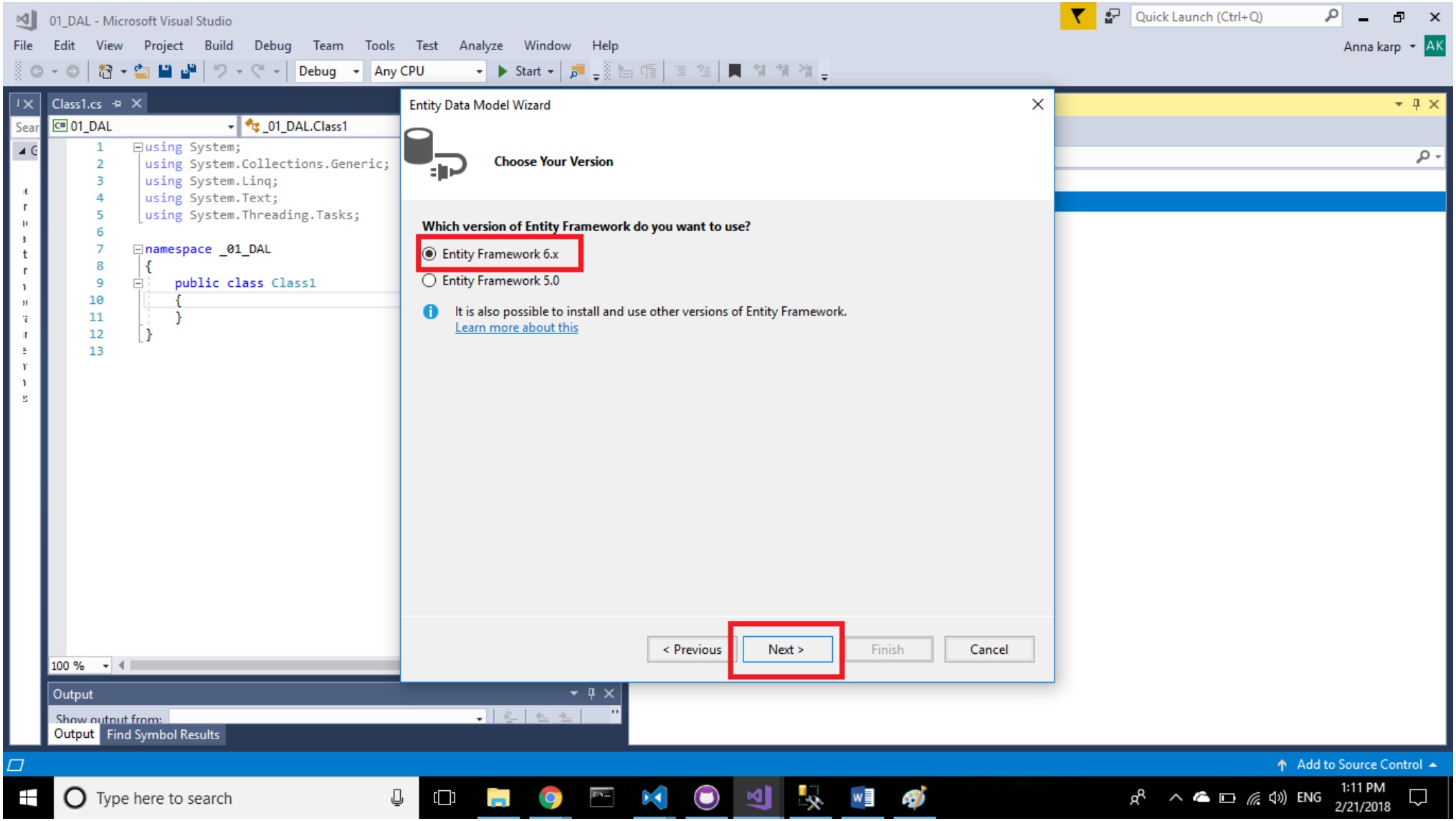
Step 6– choose your server and then select the "BookStore" DB



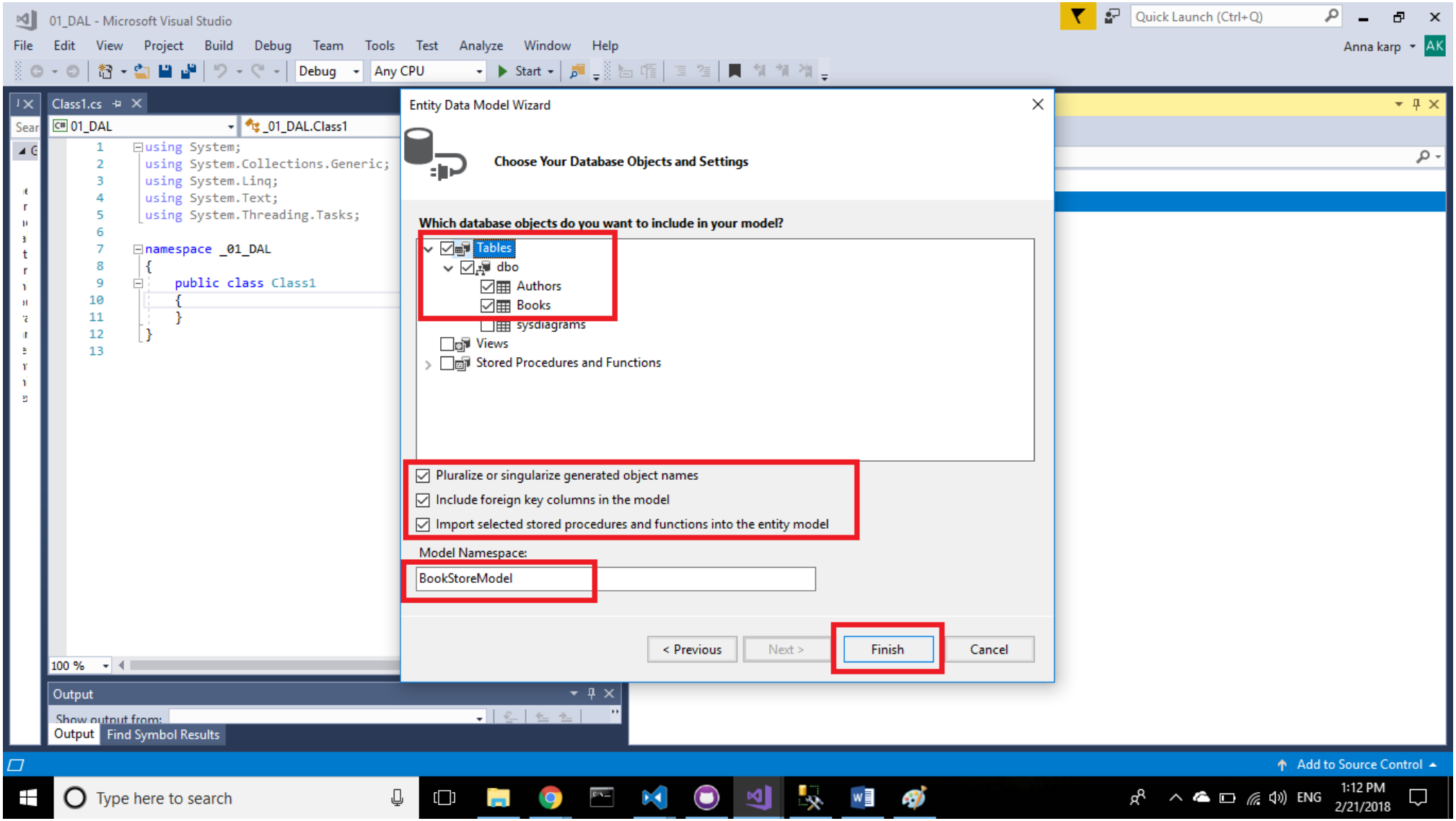
Step 7– test the connection



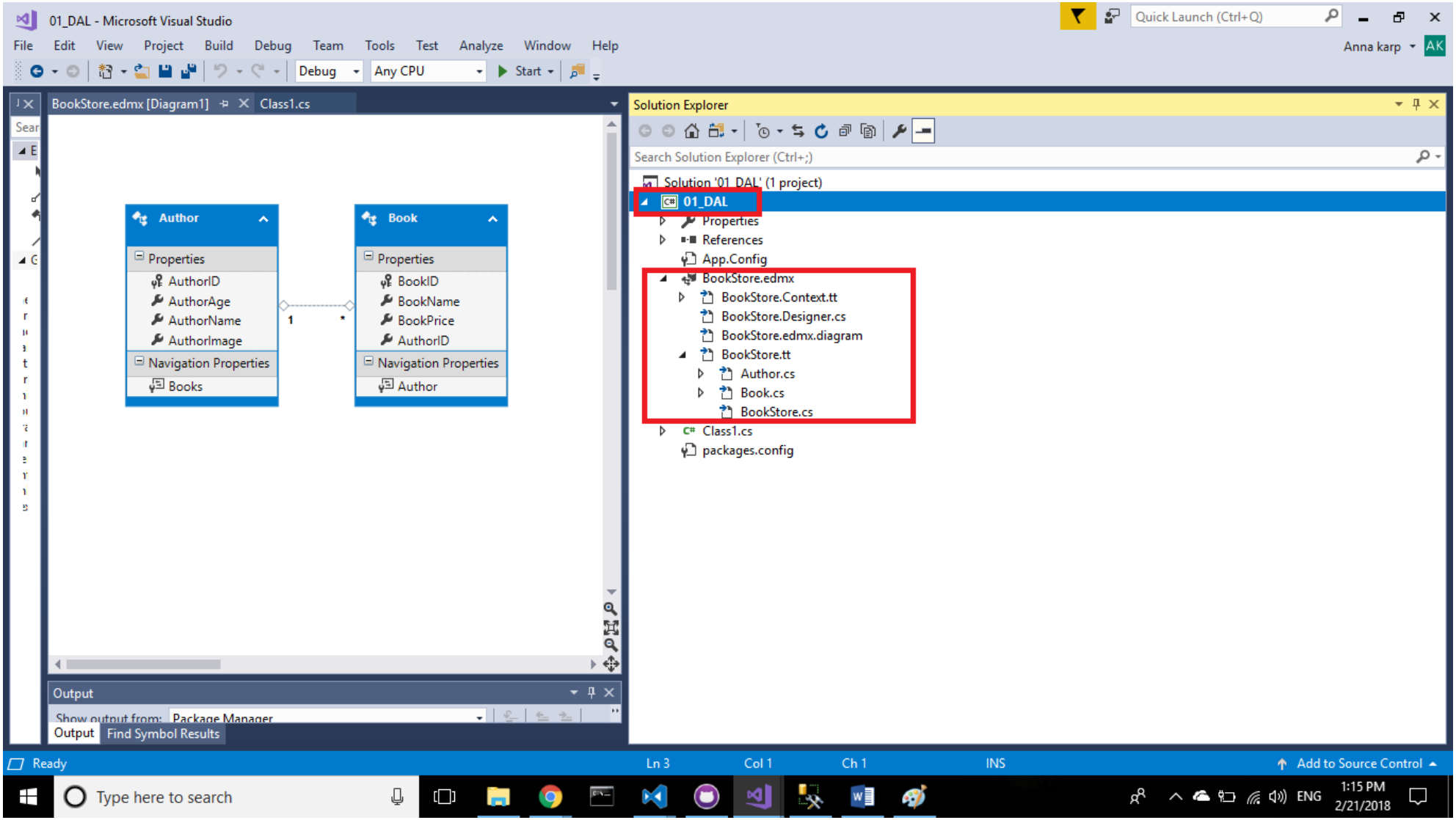
Step 8– select the ef version



Step 9– select the relevant tables

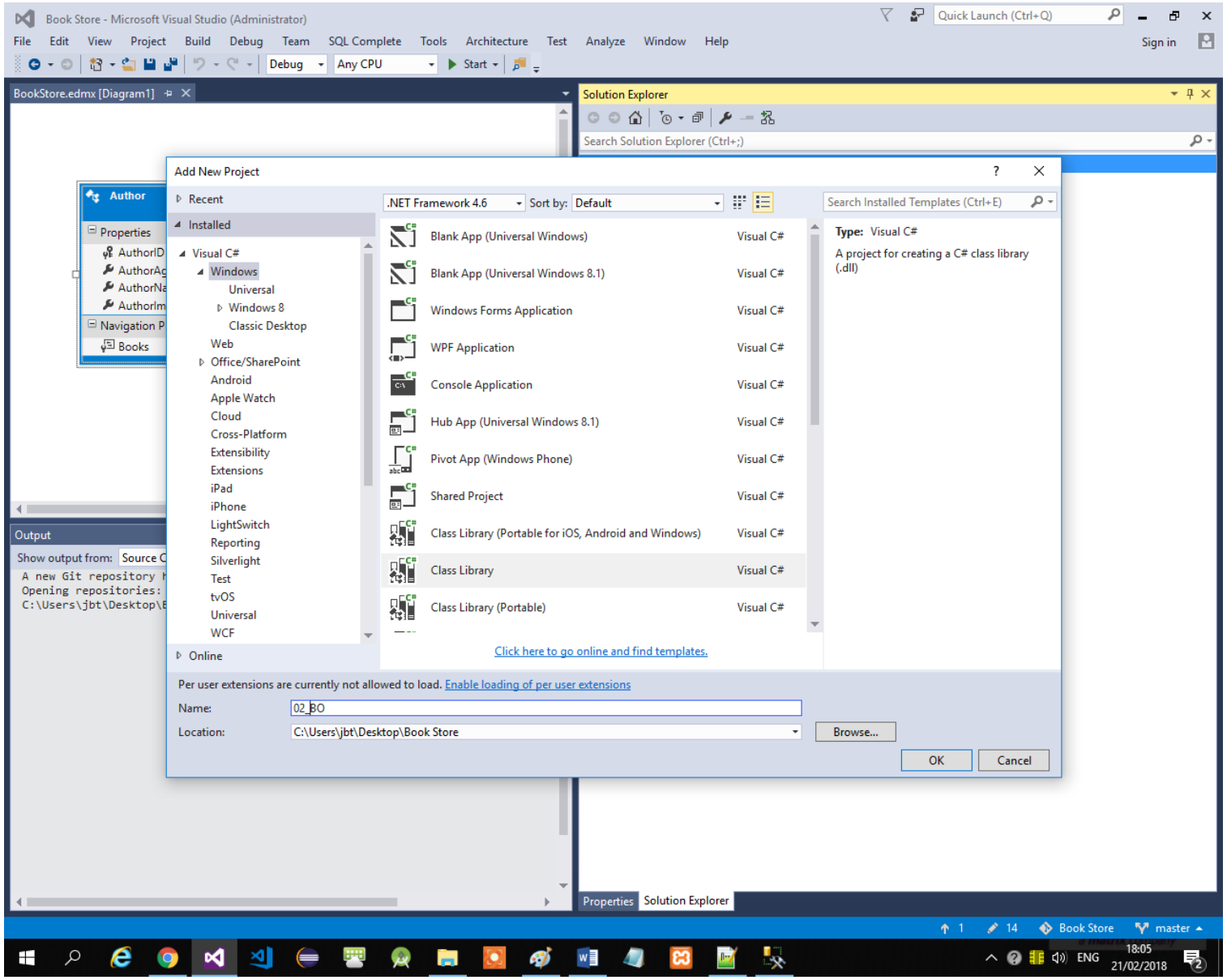


Step 10– you added the ef successfully

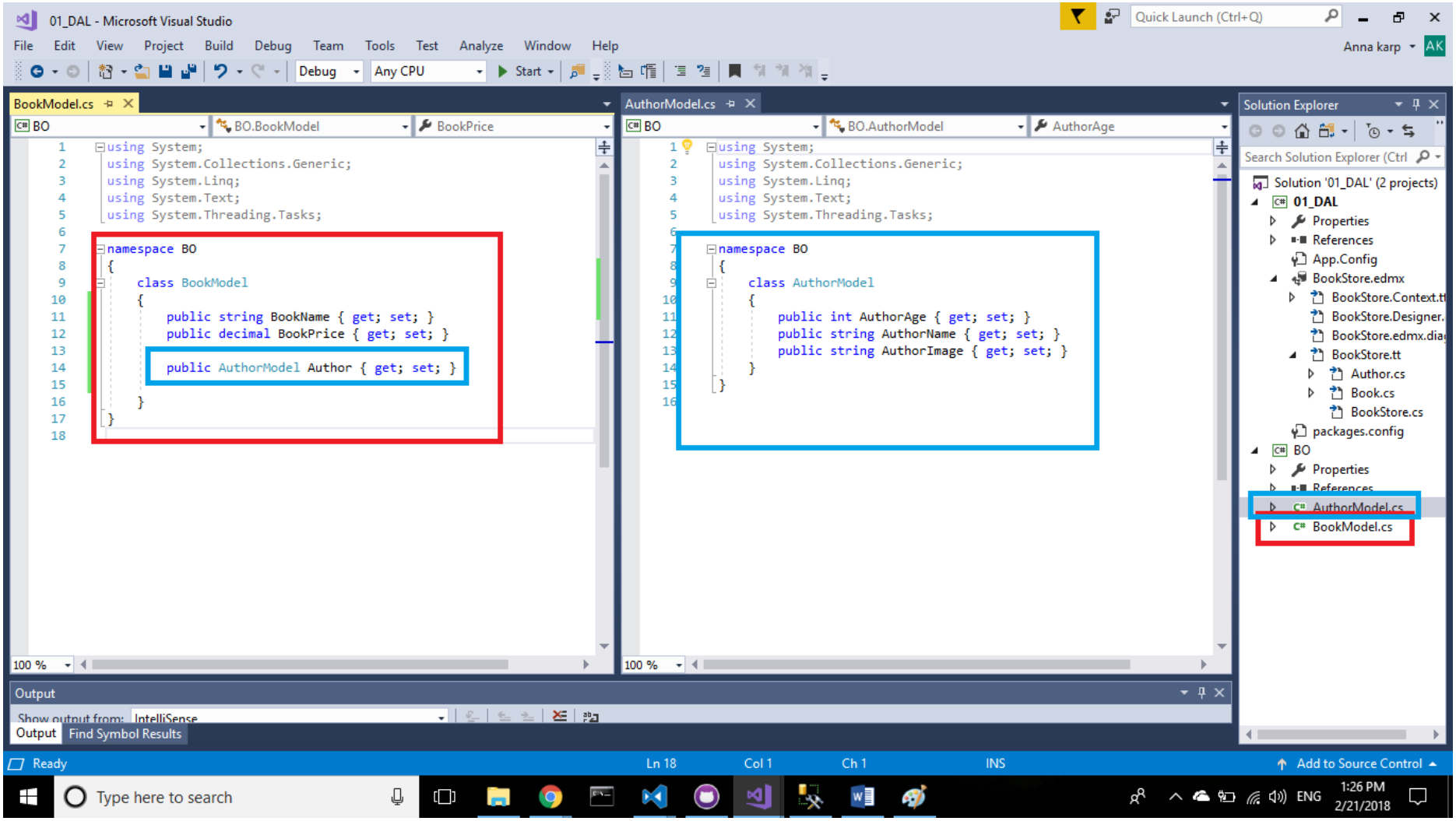


Part 3- create the BOL

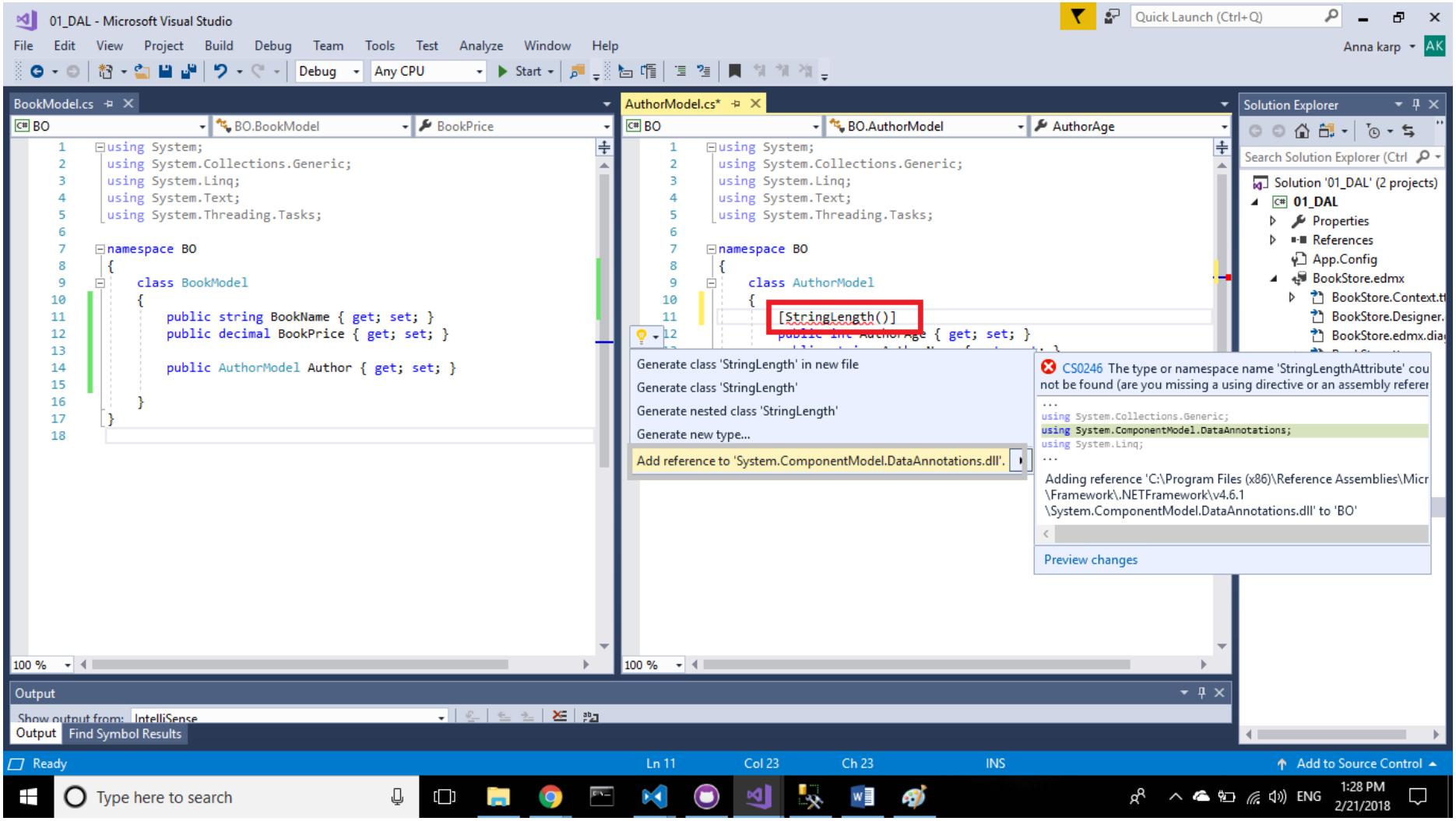
Step 1– add to the current solution a new Class library project



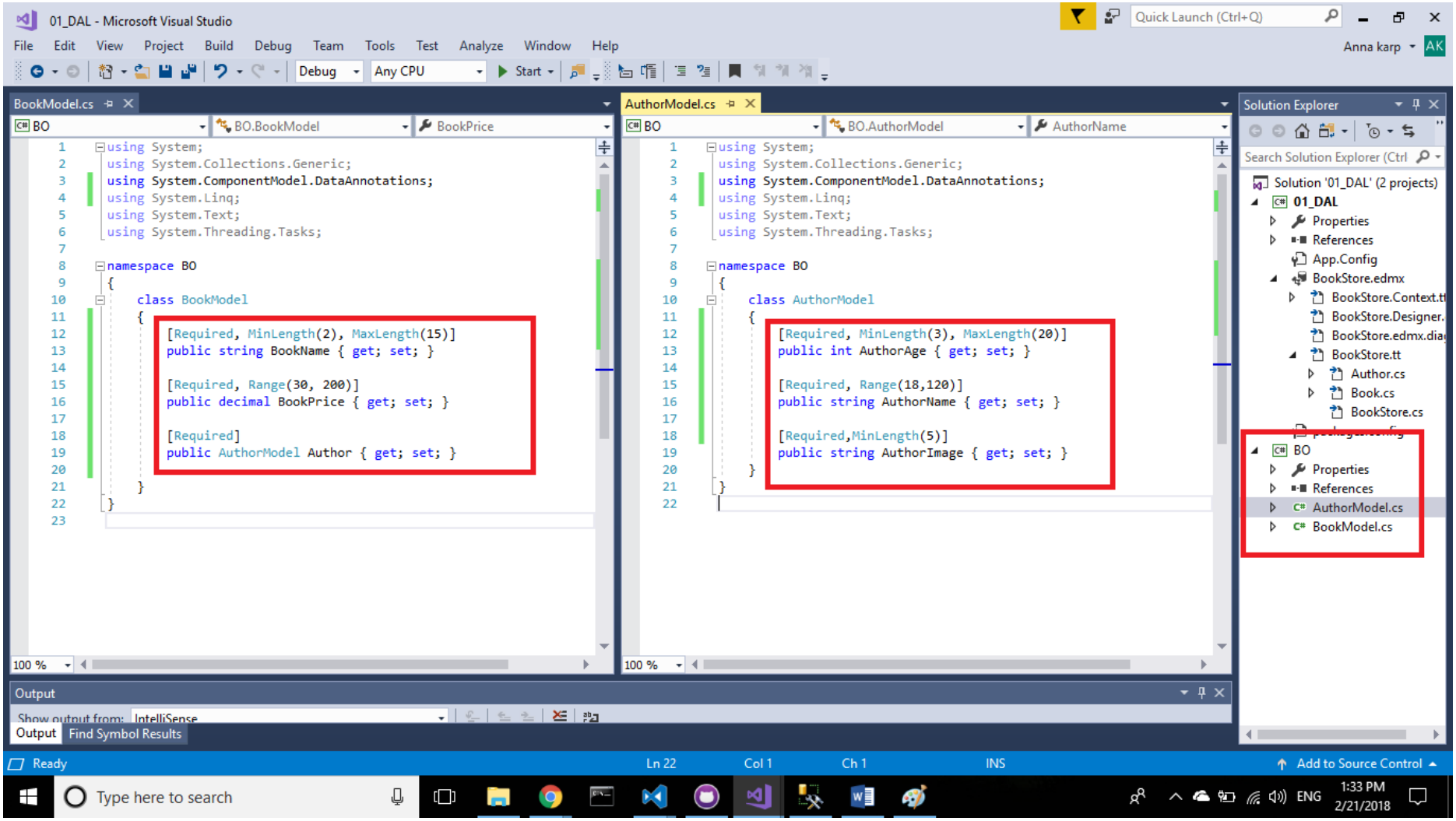
Step 2– create 2 classes with the relevant names and properties (according to the DB tables)



Step 3– add using System.ComponentModel.DataAnnotations



Step 4– add data annotation to specify the validation for the properties



Step 5– set the access modifiers to "public" in order to let other DLL's to use this classes

01_DAL - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug | Any CPU | Start

AuthorModel.cs — BO | BO.AuthorModel | AuthorAge

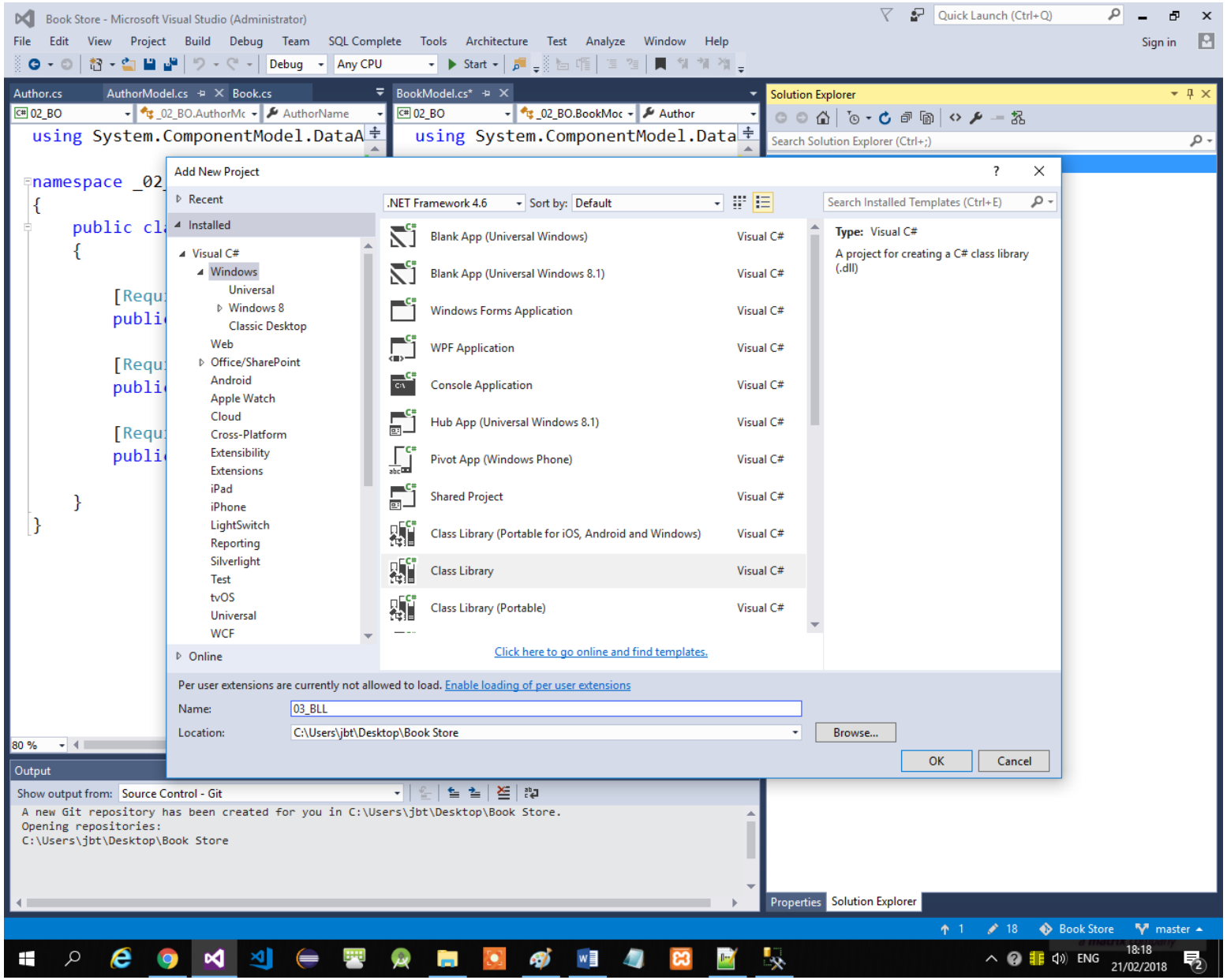
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BO
{
    public class AuthorModel
    {
        [Required, MinLength(3), MaxLength(20)]
        public int AuthorAge { get; set; }

        [Required, Range(18,120)]
        public string AuthorName { get; set; }

        [Required,MinLength(5)]
        public string AuthorImage { get; set; }
    }
}
```

BookModel.cs — BO | BO.BookModel | BookPrice

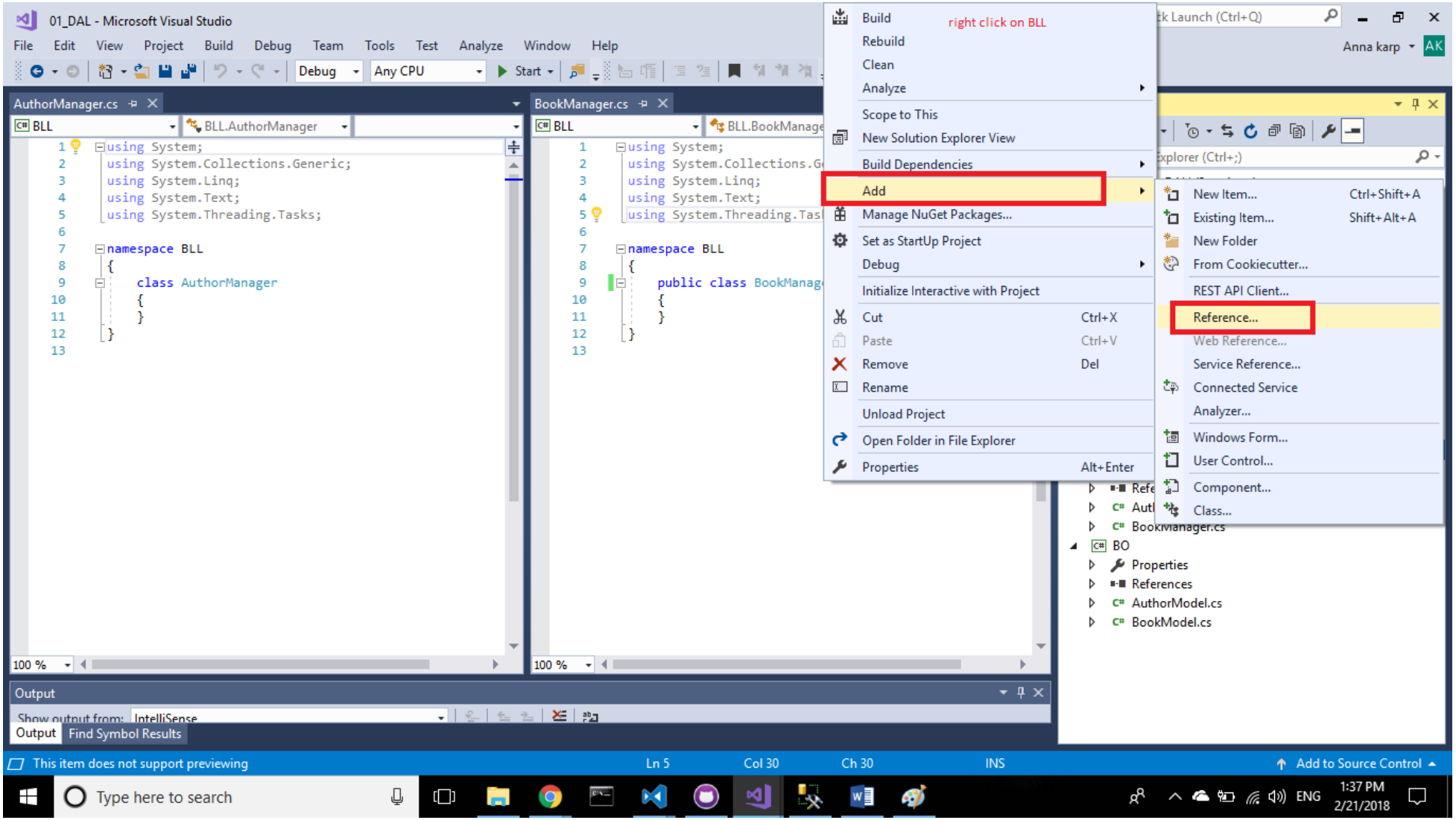
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BO
{
    public class BookModel
    {
        [Required, MinLength(2), MaxLength(15)]
        public string BookName { get; set; }

        [Required, Range(30, 200)]
        public decimal BookPrice { get; set; }

        [Required]
        public AuthorModel Author { get; set; }

    }
}
```

Part 4- create the BLL

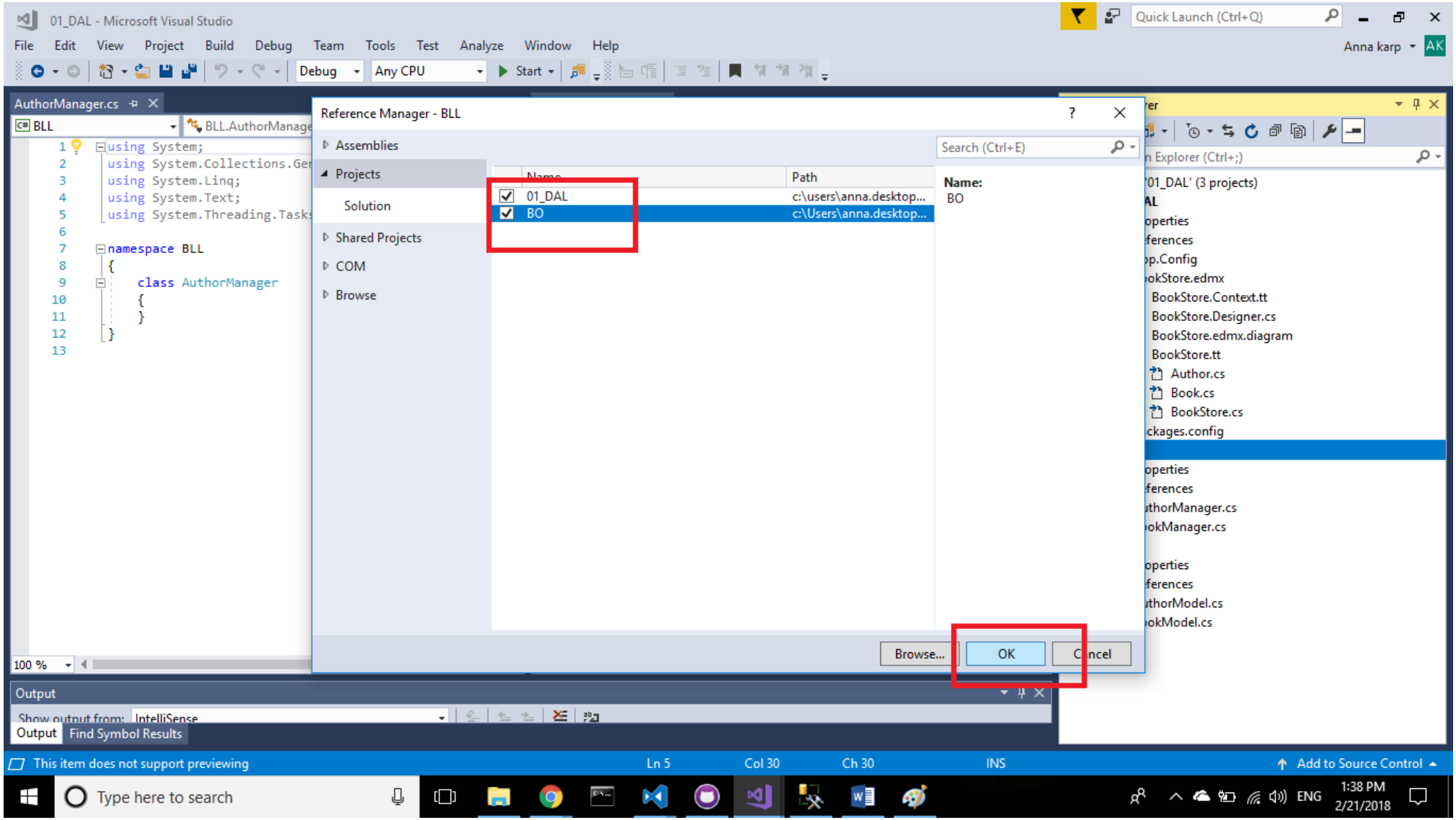
Step 1– add to the current solution a new Class library project



Step 2– add to this project references



Step 3– add to this project references to the DAL and BOL

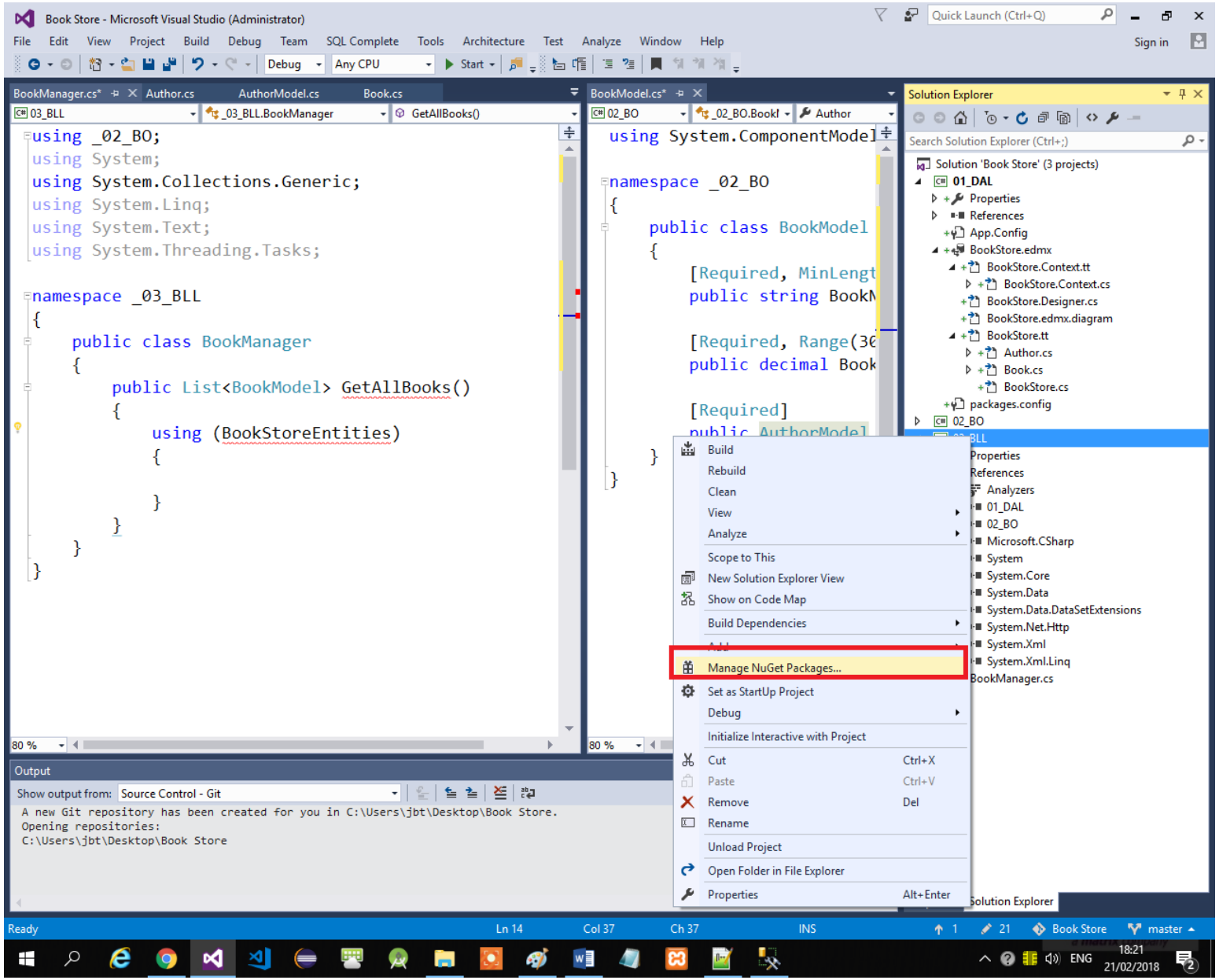


Step 4– create a BookManager class – note: in the next step you will add ef and fix the error

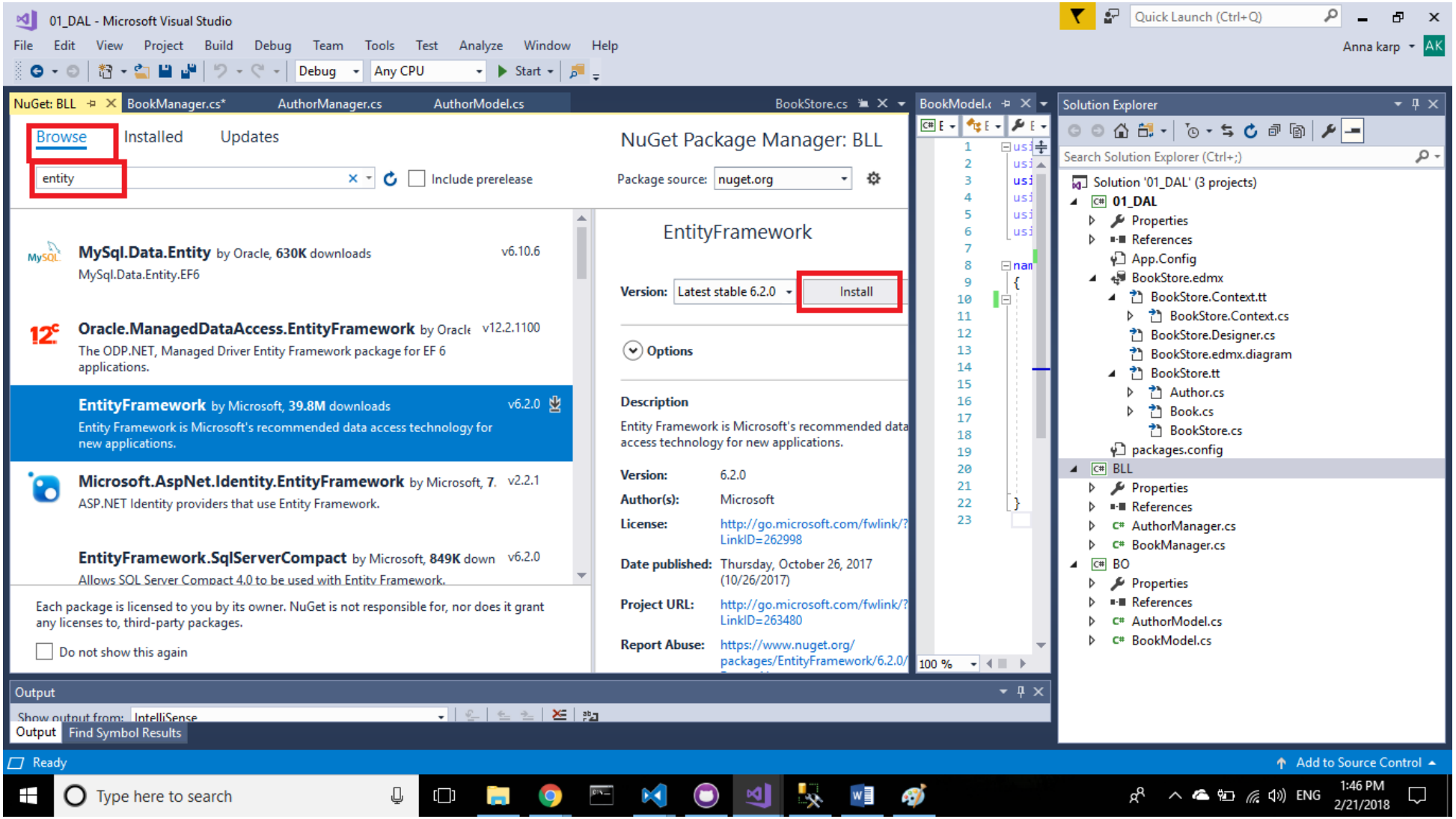
```
using _01_DAL;
using BO;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BLL
{
    public class BookManager
    {

        // Get all books:
        public List<BookModel> GetAllOrders()
        {
            using(BookStoreEntities db= new BookStoreEntities())
            {
                ret
                new
                {
```

Add reference to 'EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyTo...

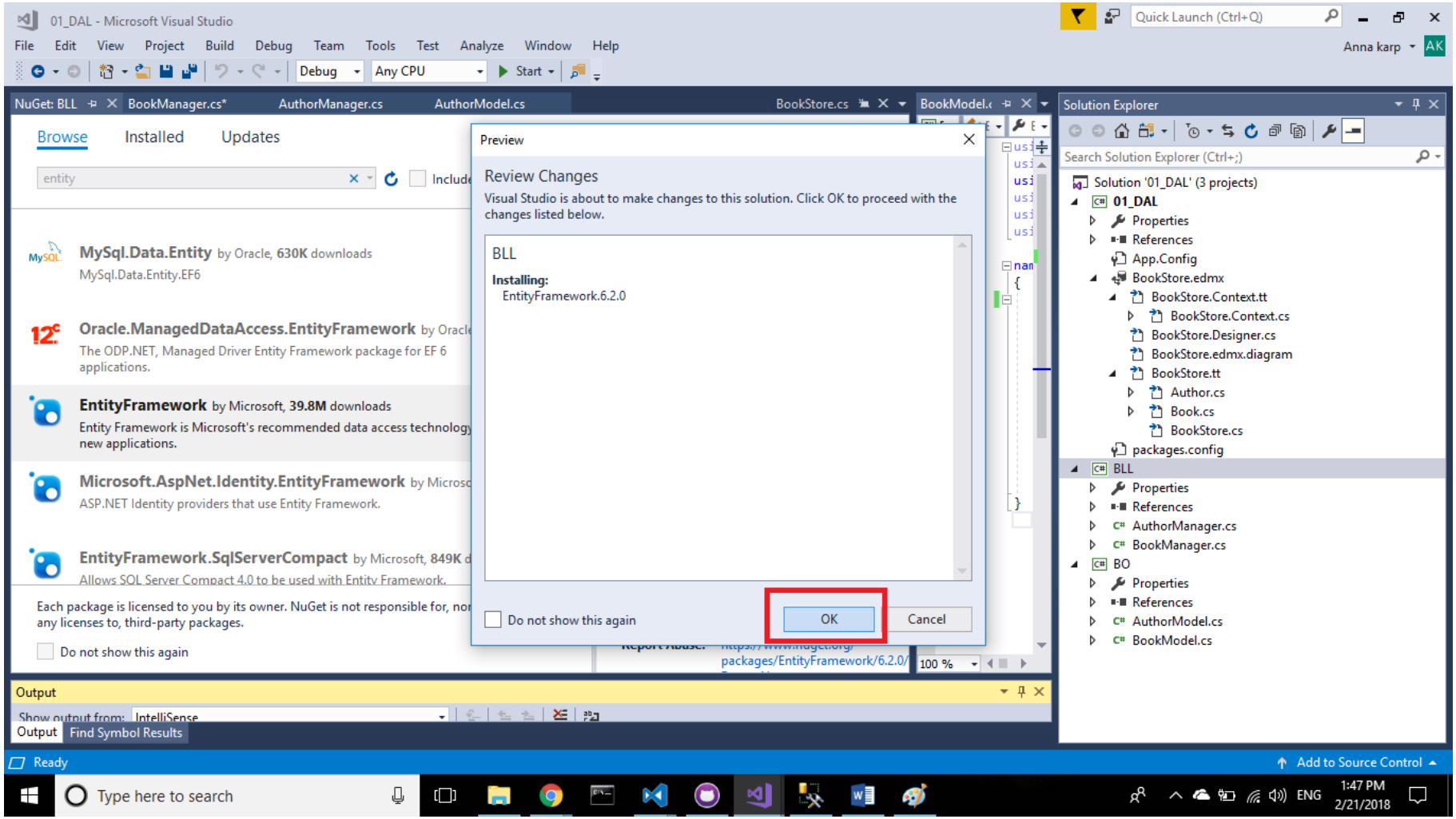
Step 5– add ef to the BLL with the nuget packages manager



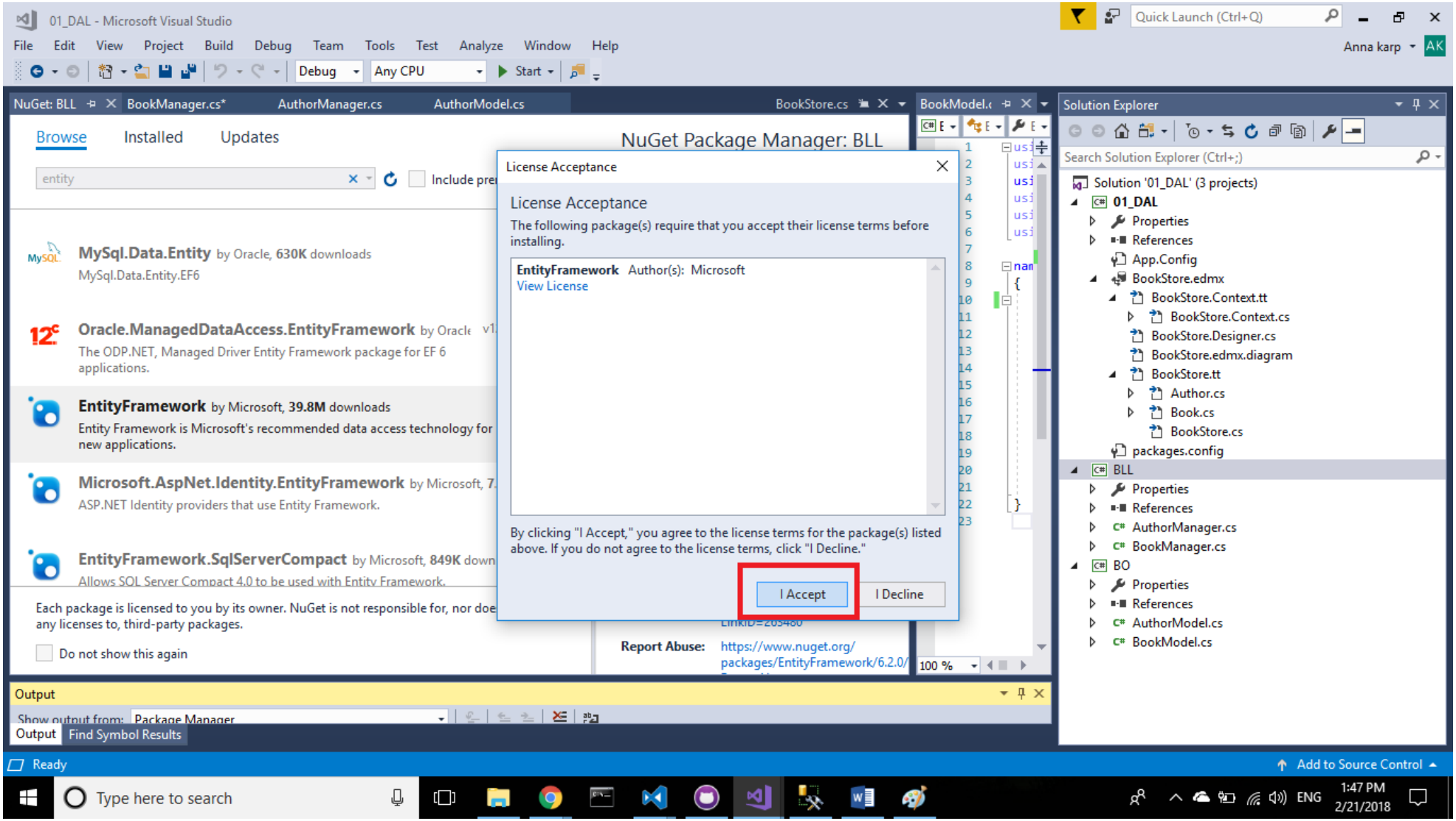
Step 6– browse entity package



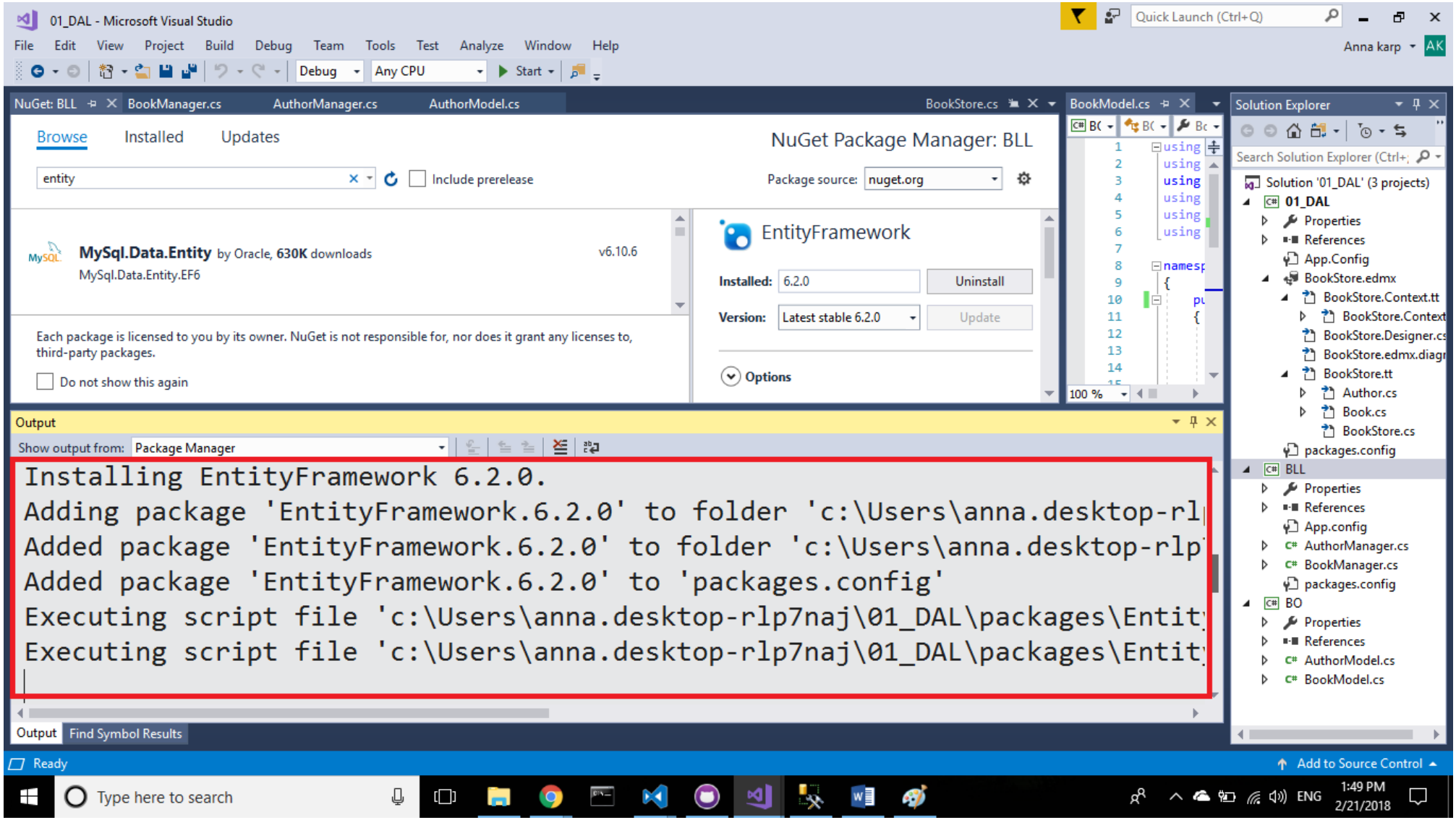
Step 7– start adding the ef package



Step 8– add the ef package



Step 9– the ef package has added successfully



Step 10– add the relevant logic to the BookManager and AouthorManager classes

01_DAL - Microsoft Visual Studio

BookManager.cs*

```
using _01_DAL;
using BO;
using System.Collections.Generic;
using System.Linq;

namespace BLL
{
    public class BookManager
    {

        // Get all books:
        public List<BookModel> GetAllBooks()
        {
            using (BookStoreEntities db = new BookStoreEntities())
            {
                return db.Books.Select(b =>
                new BookModel
                {
                    BookName = b.BookName,
                    BookPrice = b.BookPrice,
                    Author = new AuthorModel()
                    {
                        AuthorName = b.Author.AuthorName,
                        AuthorAge = b.Author.AuthorAge,
                        AuthorImage = b.Author.AuthorImage
                    }
                }).ToList();
            }

        }

    }
}
```

AuthorManager.cs*

```
using BO;
using System.Collections.Generic;
using System.Linq;

namespace BLL
{
    class AuthorManager
    {


        public List<AuthorModel> GetAllAuthors()
        {
            using (BookStoreEntities db = new BookStoreEntities())
            {
                return db.Authors.Select(a =>
                new AuthorModel()
                    {
                        AuthorName = a.AuthorName,
                        AuthorAge = a.AuthorAge,
                        AuthorImage = a.AuthorImage

                    }).ToList();
                }

            }
        }
}
```

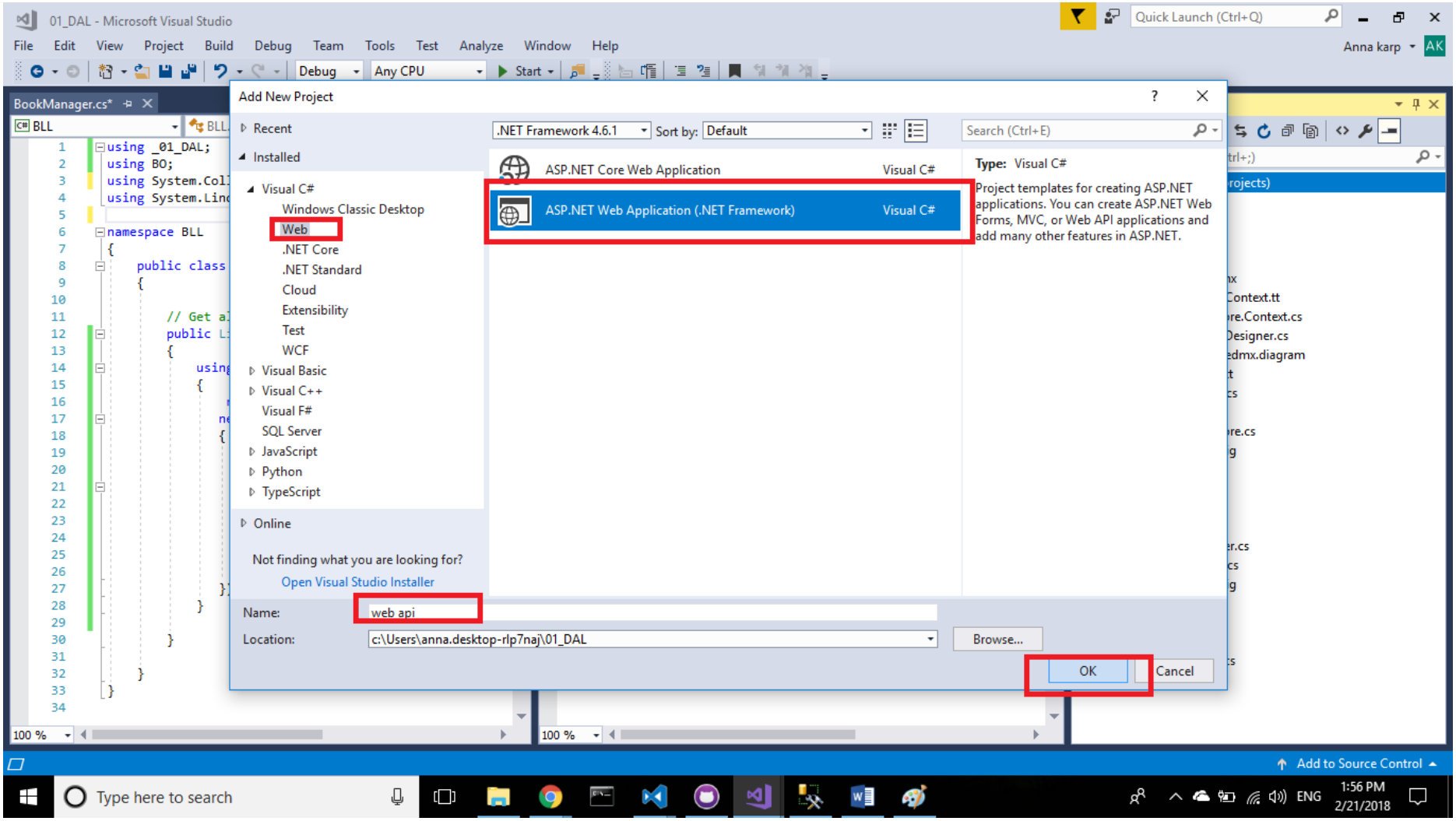
Solution Explorer

- Solution '01_DAL' (3 proje
- 01_DAL
  - Properties
  - References
  - App.Config
  - BookStore.edmx
    - BookStore.Con
      - BookStore.(
      - BookStore.Desi
      - BookStore.edm
    - BookStore.tt
      - Author.cs
      - Book.cs
      - BookStore.c
  - packages.config
- BLL
  - Properties
  - References
  - App.config
  - AuthorManager.cs
  - BookManager.cs
  - packages.config
- BO
  - Properties
  - References
  - AuthorModel.cs
  - BookModel.cs

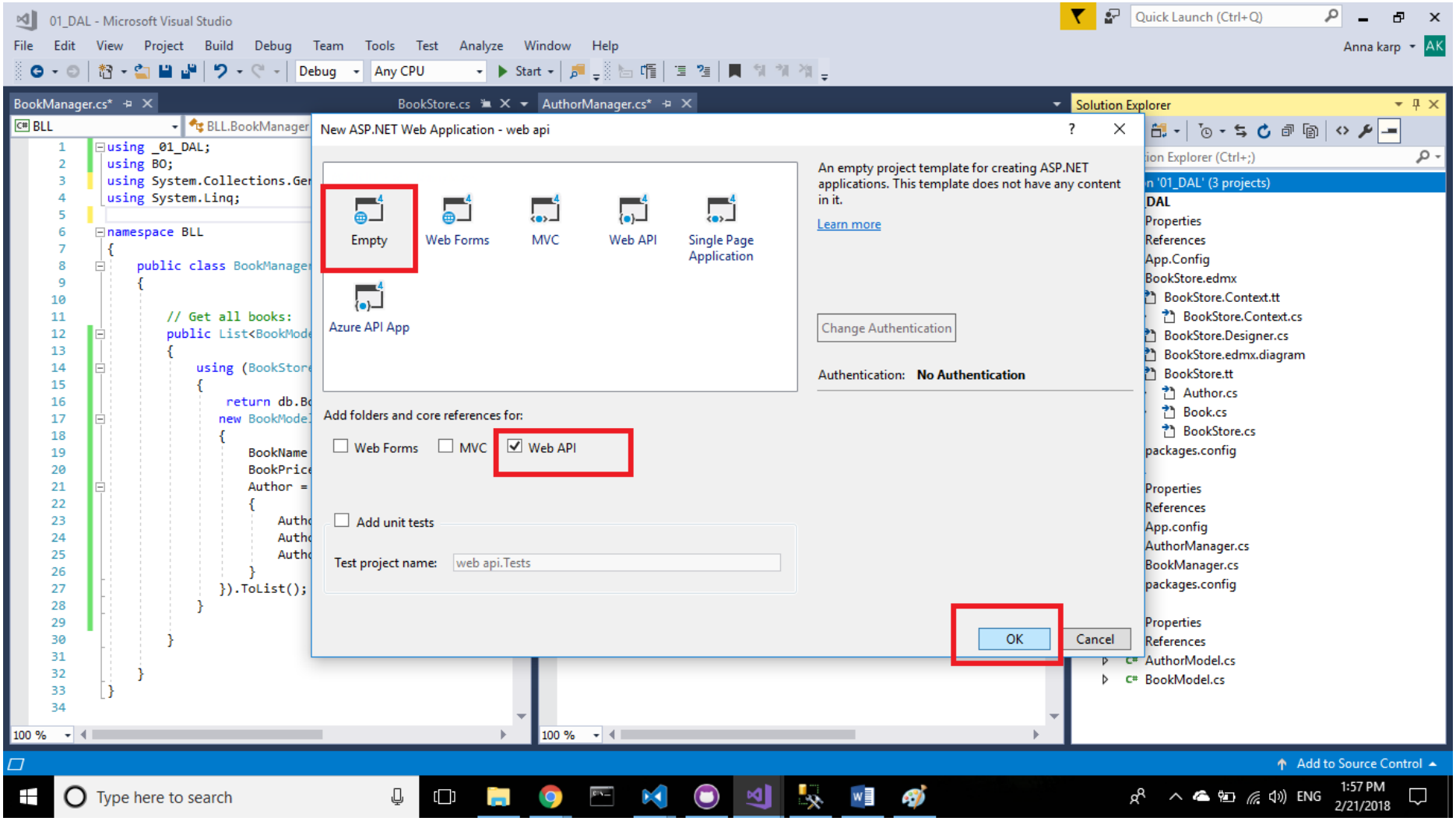
Ready | Ln 29 | Col 1 | Ch 1 | INS | Add to Source Control

Part 5- create the Web-Api

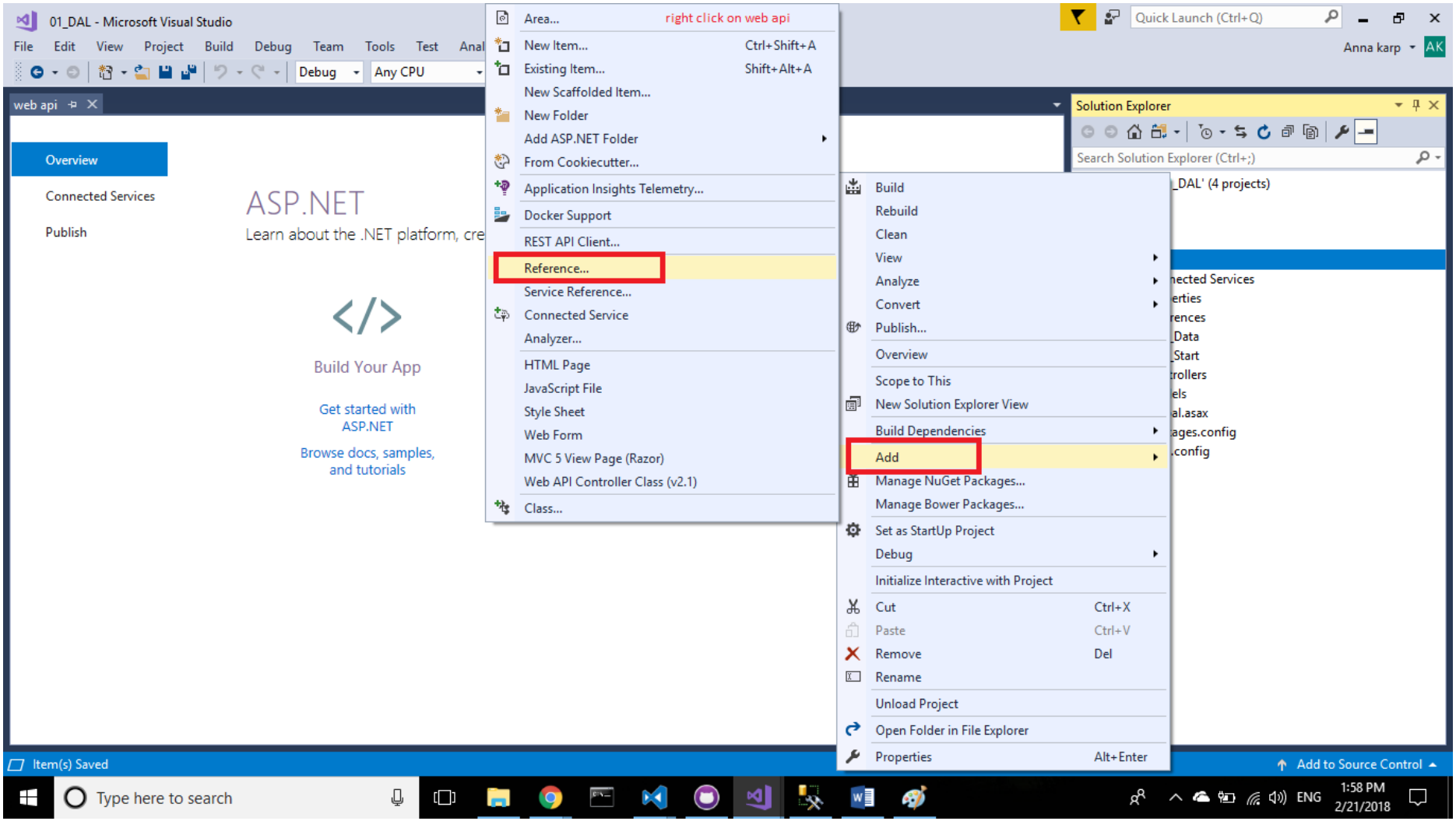
Step 1– add to the current solution an ASP.NET project



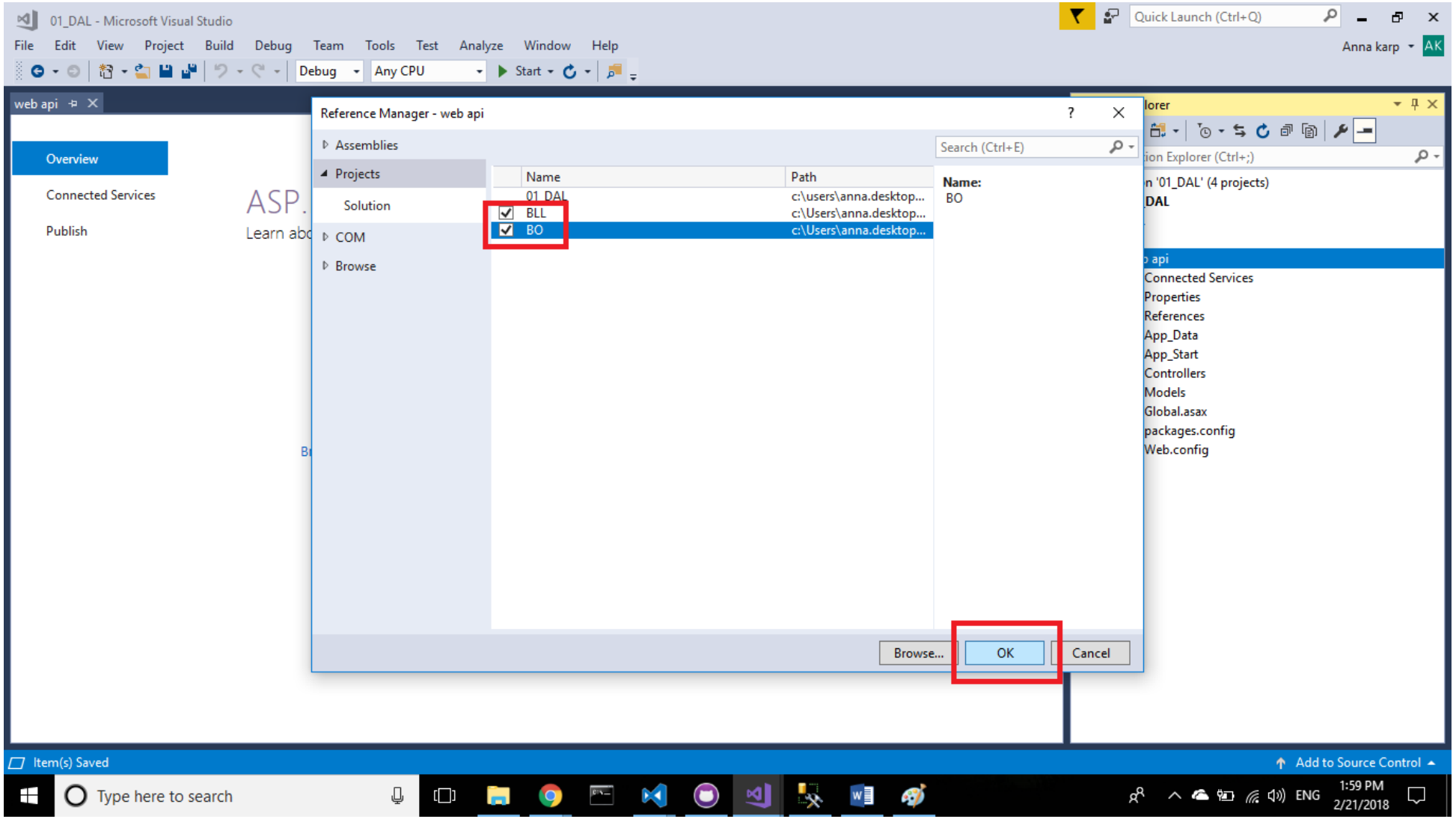
Step 2– select an Empty web api project



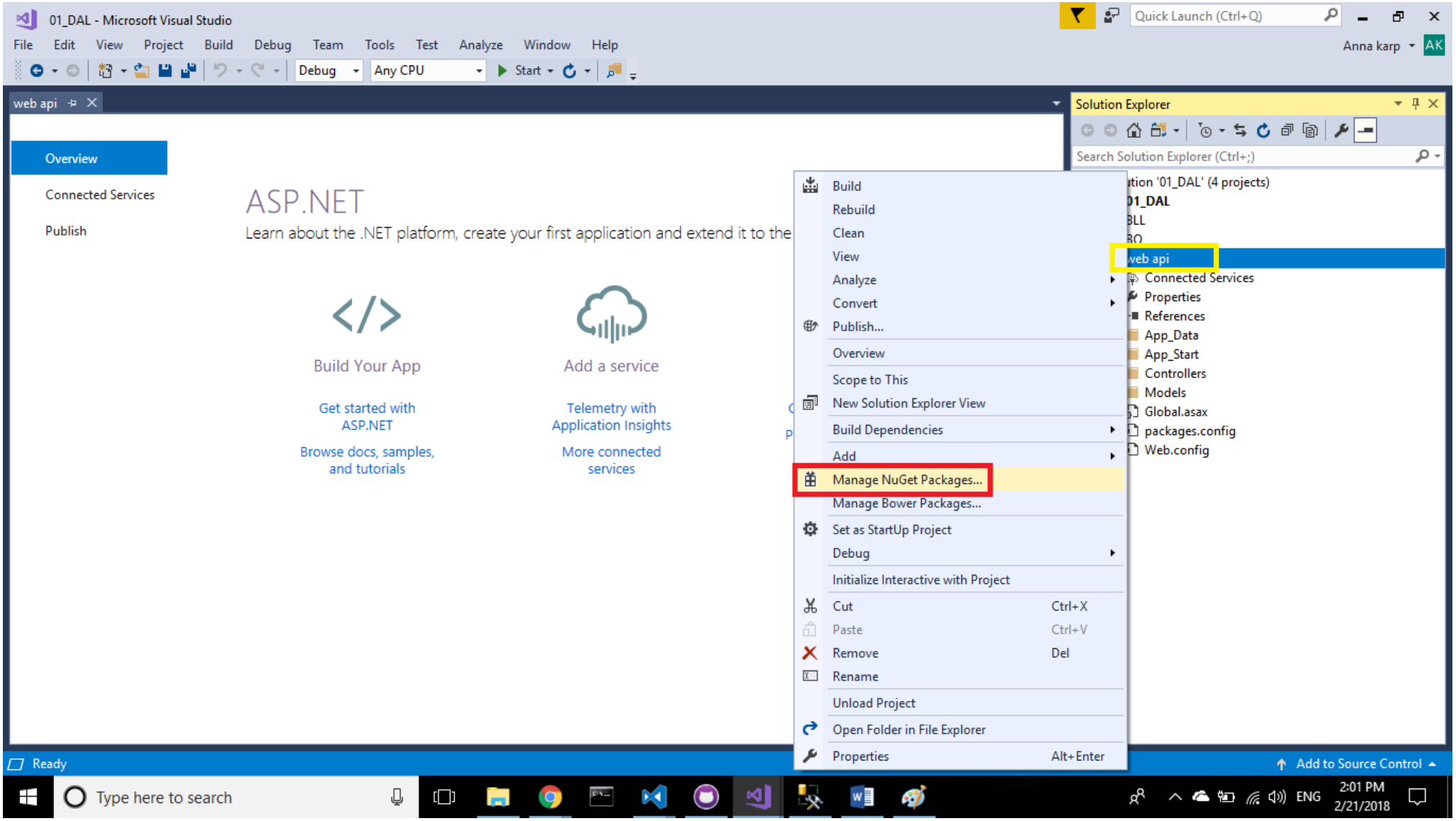
Step 3– add references



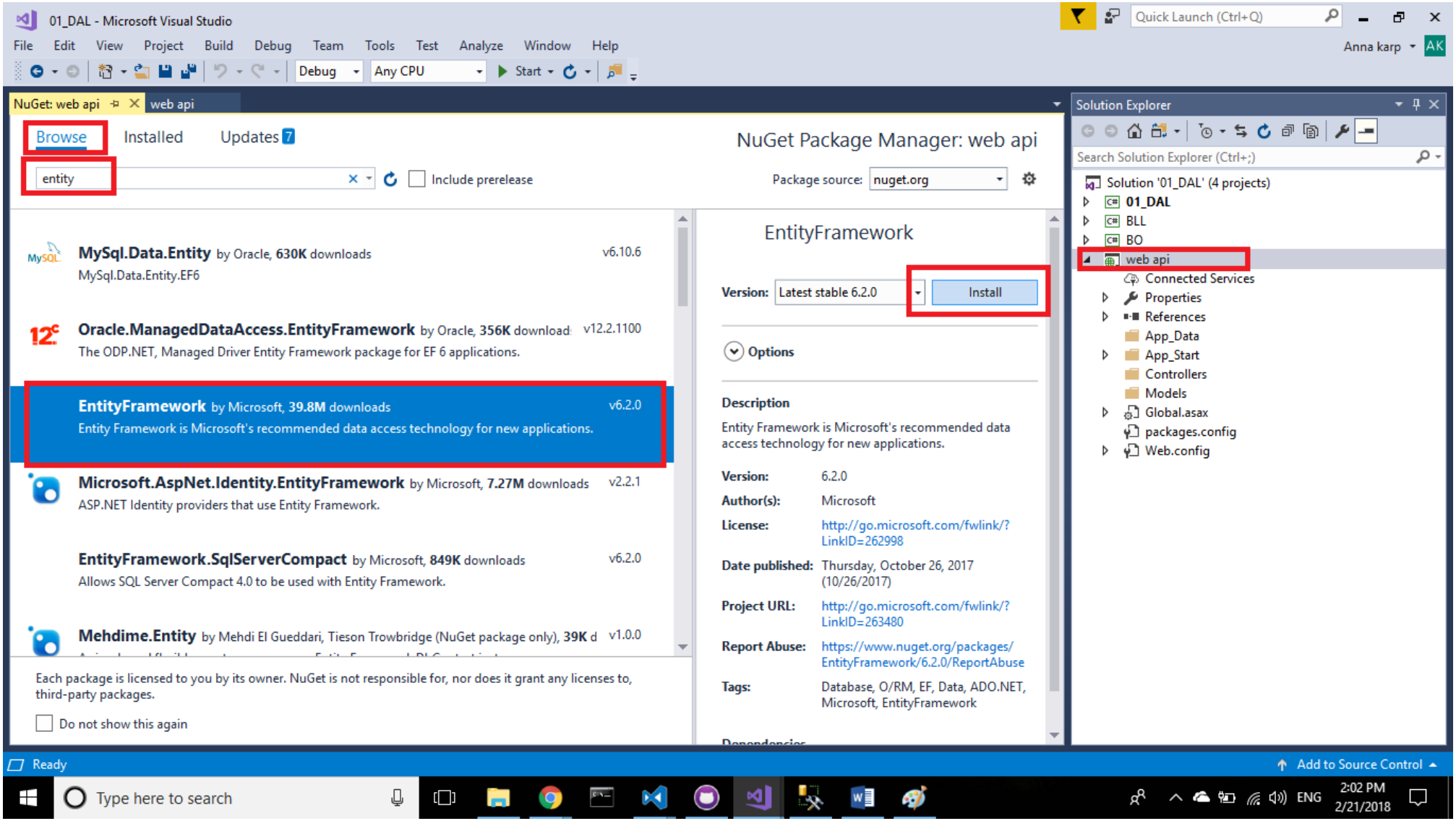
Step 4– add references to the BOL and BLL



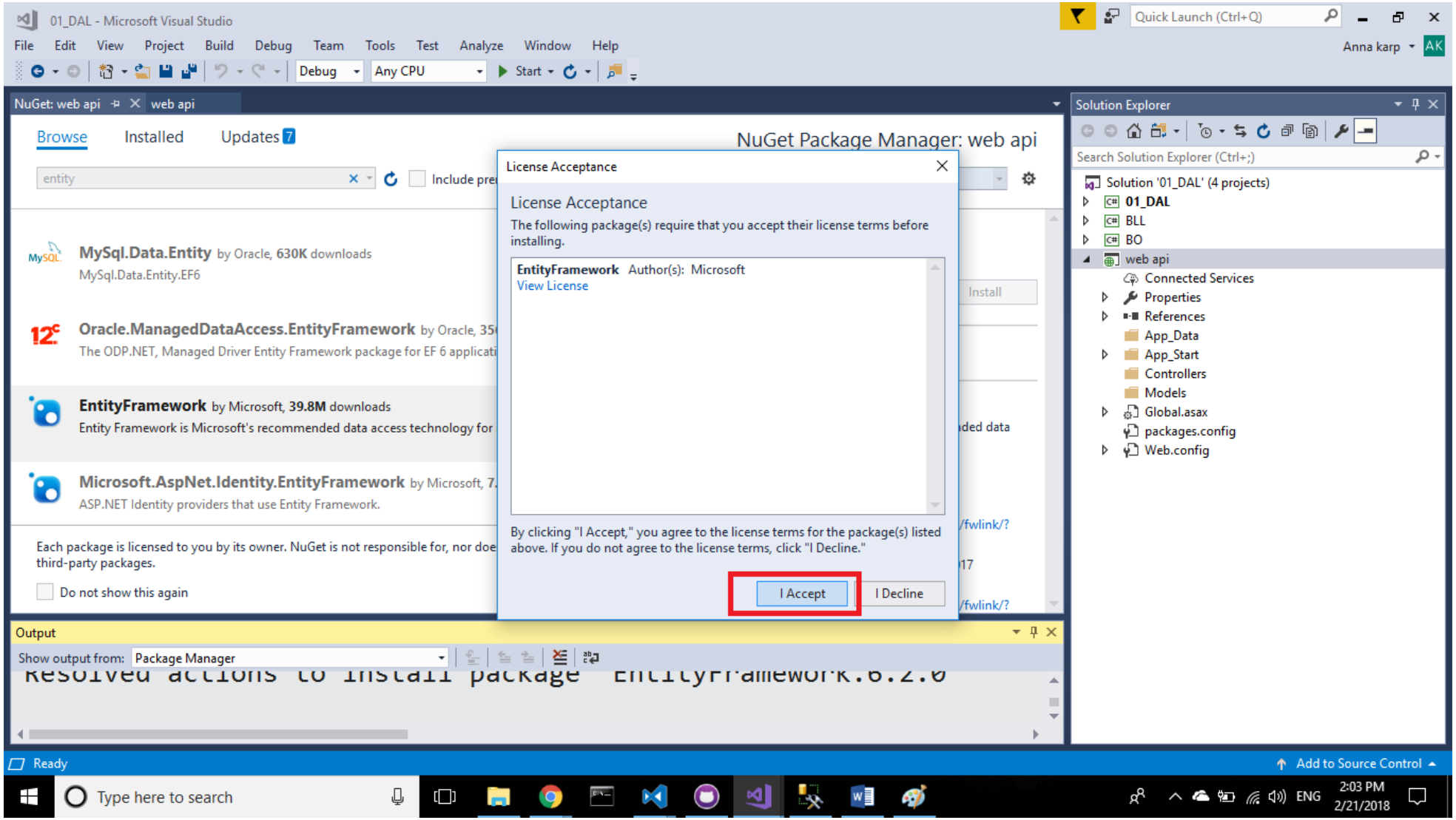
Step 5– add a reference to the ef package with the nuget package manager



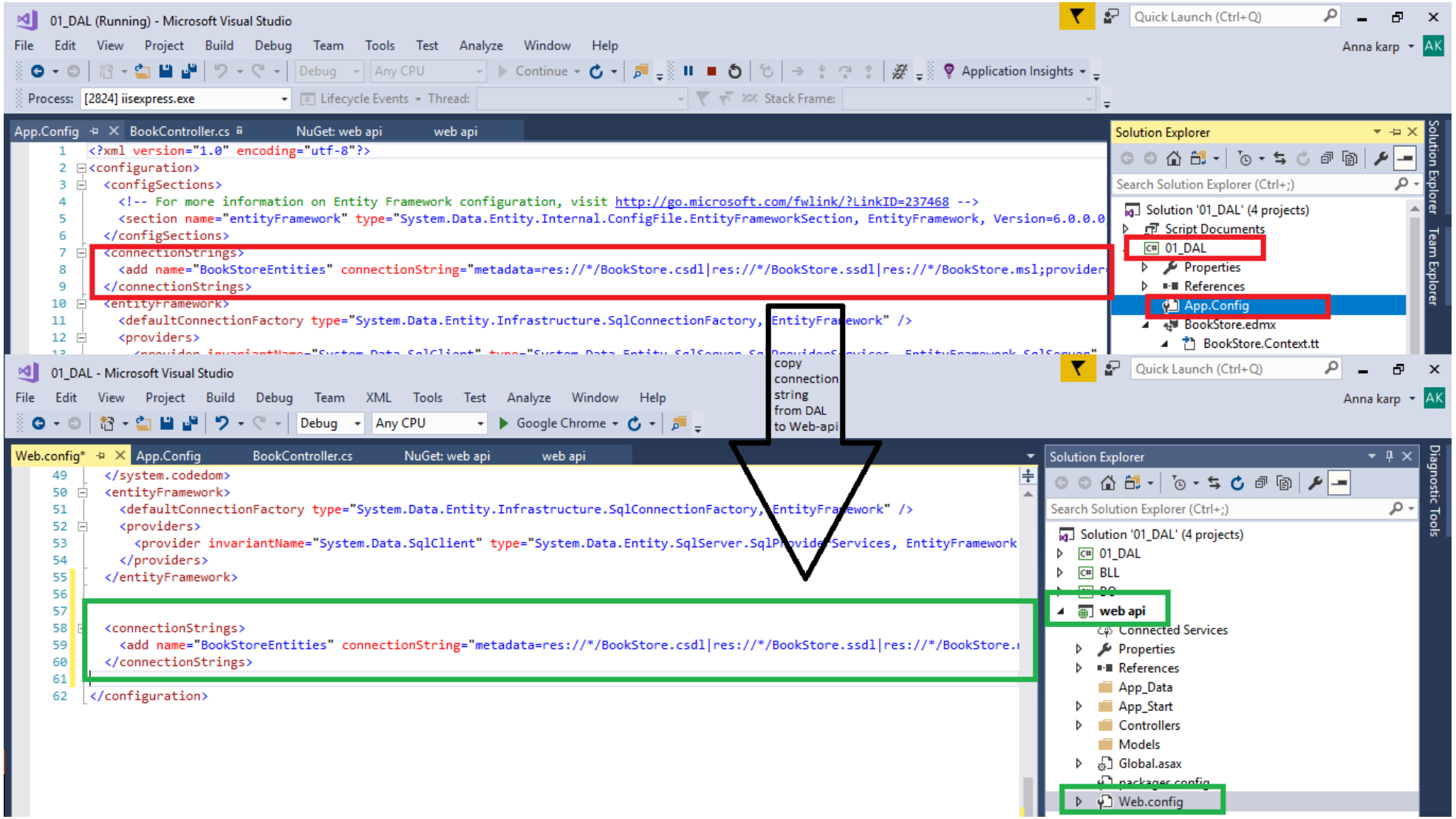
Step 6– browse the entity option



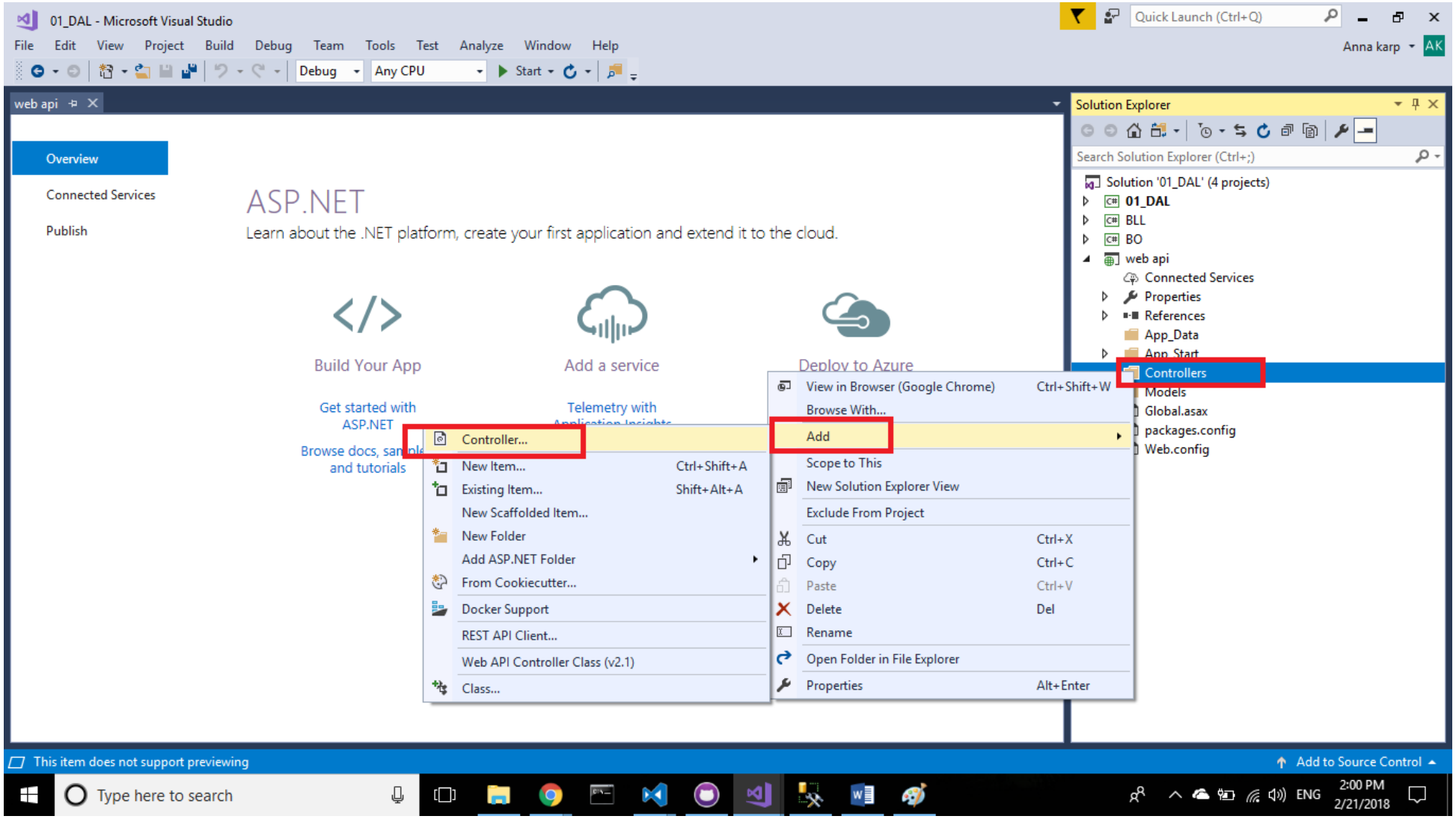
Step 7– install the ef package



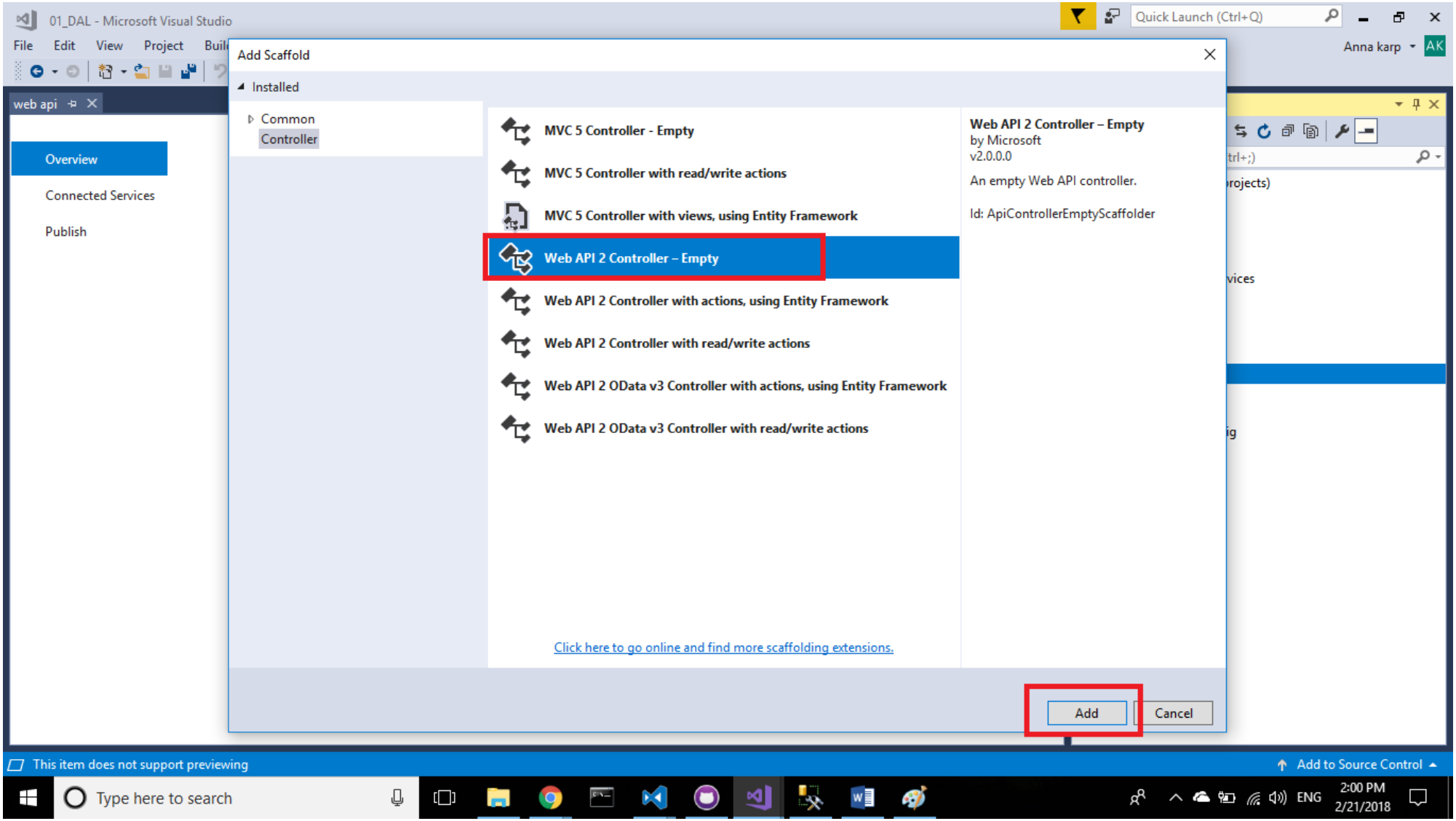
Step 8– copy the connection string from the DAL to the web api – web config



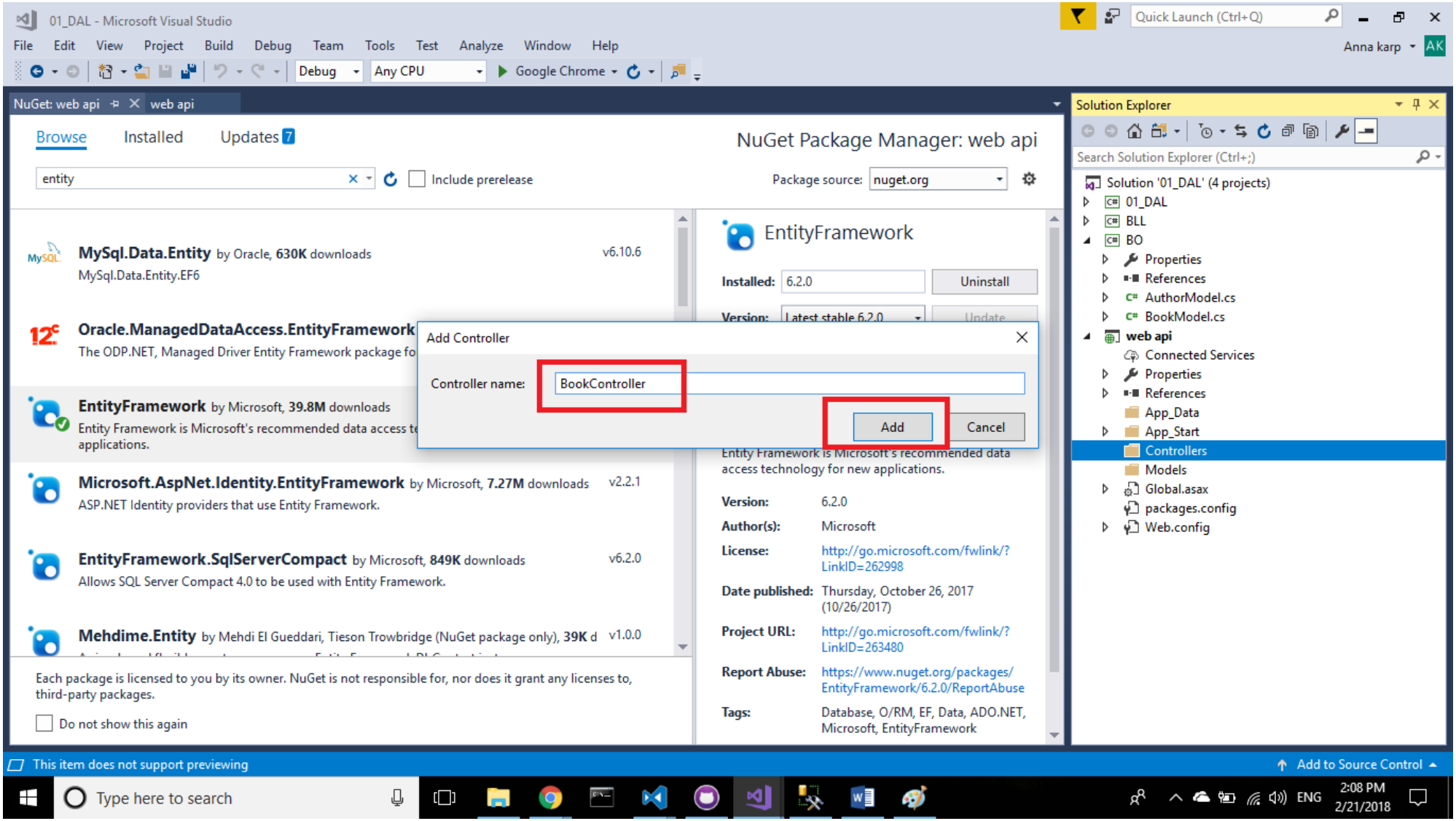
Step 9– add a new controller



Step 10– select a web api 2 controller



Step 11– name the new controller



Step 12– add the relevant content to the controller

01_DAL - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Web.config | App.Config | BookController.cs | NuGet: web api | web api

web api | web_api.Controllers.BookController | logic

```
using BLL;
using BO;
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace web_api.Controllers
{
    [RoutePrefix("api/book")]
    public class BookController : ApiController
    {
        BookManager logic = new BookManager();

        [HttpGet]
        [Route("all")]   // access link : /api/book/all
        public HttpResponseMessage GetAllBooks()
        {
            {
                try
                {
                    List<BookModel> books = logic.GetAllBooks();
                    return Request.CreateResponse(HttpStatusCode.OK, books);
                }
                catch (Exception ex)
                {
                    return Request.CreateErrorResponse(HttpStatusCode.InternalServerError,ex);
                }
            }
        }

    }
}
```

Solution Explorer

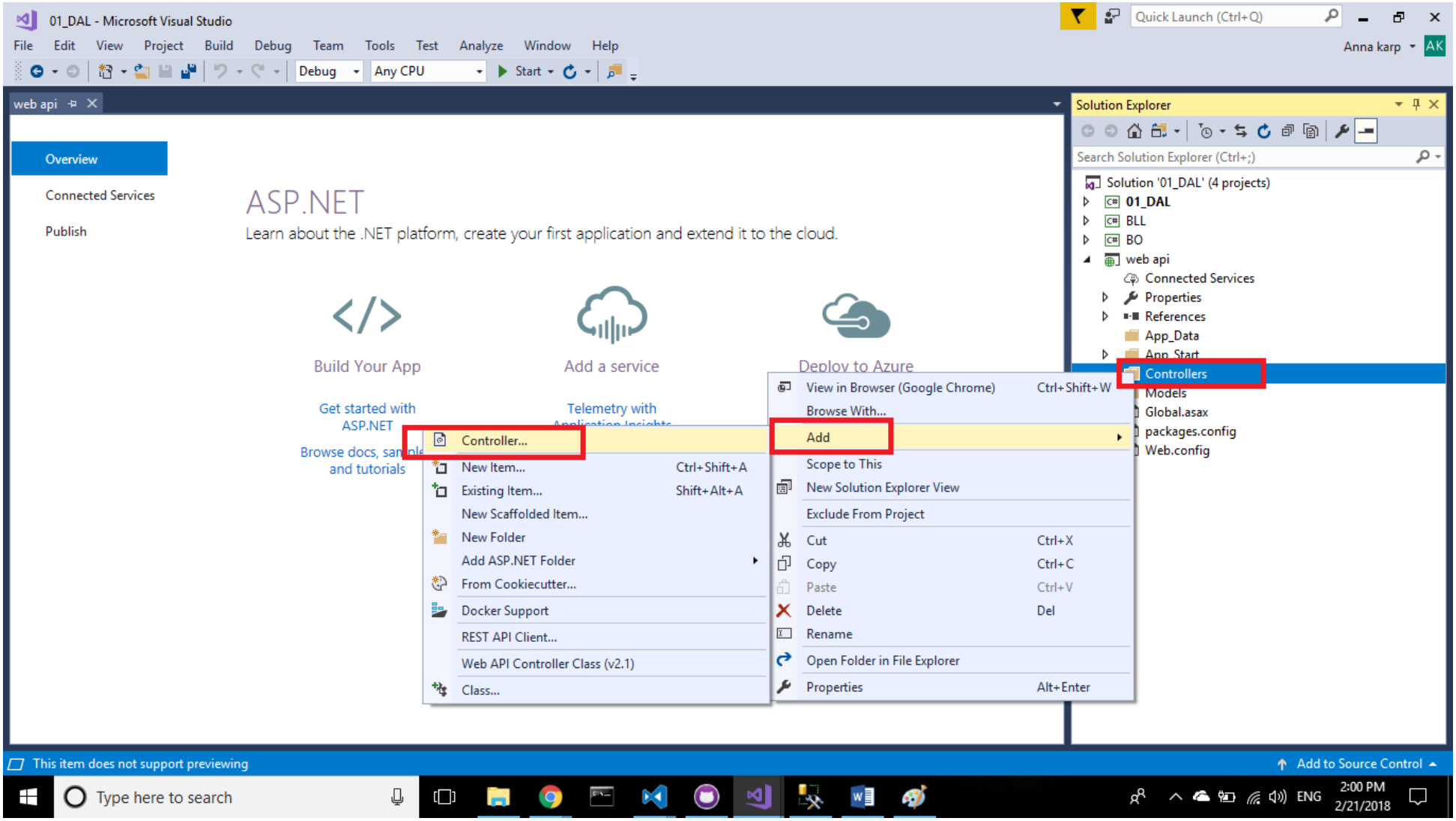
Search Solution Explorer (Ctrl+;)

- Solution '01_DAL' (4 projects)
  - 01_DAL
  - BLL
  - BO
  - web api
    - Connected Services
    - Properties
    - References
    - App_Data
    - App_Start
    - Controllers
      - BookController.cs
    - Models
    - Global.asax
    - packages.config
    - Web.config

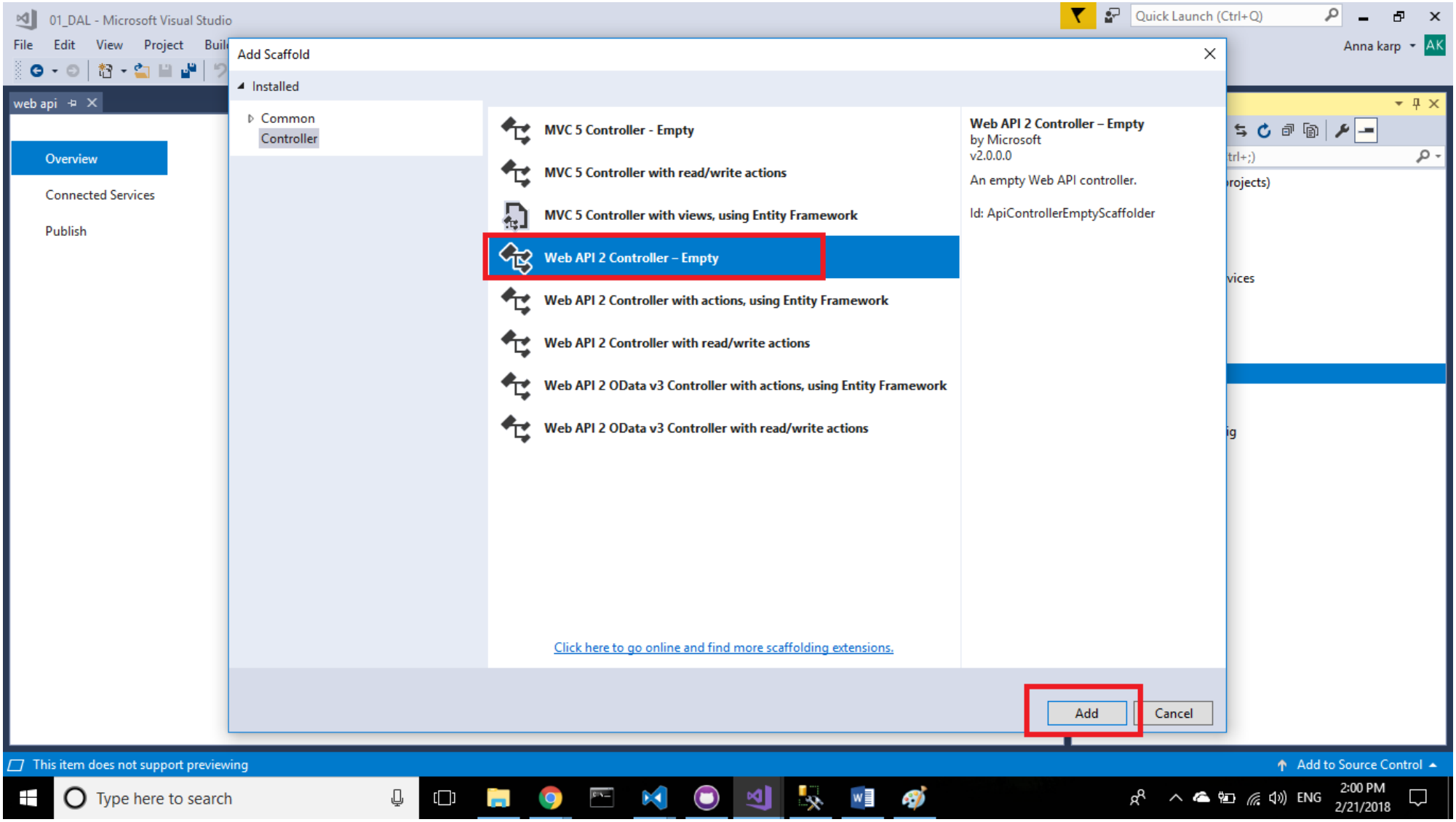
Ready | Ln 13 | Col 6 | Ch 6 | INS | Add to Source Control

Type here to search | ENG | 2:27 PM 2/21/2018

Step 13– add another controller



Step 14– select again the web api 2 controller



Step 15– name the controller

01_DAL - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug | Any CPU | Google Chrome

Web.config | App.Config | BookController.cs | NuGet: web api | web api

web api | web_api.Controllers.BookController | logic

```
using BLL;
using BO;
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace web_api.Controllers
{
    [RoutePrefix("api/book")]
    public class BookController : ApiContr
    {
        BookManager logic = new BookManage

        [HttpGet]
        [Route("all")]   // access link :
        public HttpResponseMessage GetAllB
        {
            {
                try
                {
                    List<BookModel> books = logic.GetAllBooks();
                    return Request.CreateResponse(HttpStatusCode.OK, books);
                }
                catch (Exception ex)
                {
                    return Request.CreateErrorResponse(HttpStatusCode.InternalServerError,ex);
                }
            }
        }

    }
}
```

Add Controller

Controller name: AuthorController

Add | Cancel

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution '01_DAL' (4 projects)
  - 01_DAL
  - BLL
  - BO
  - web api
    - Connected Services
    - Properties
    - References
    - App_Data
    - App_Start
    - Controllers
      - BookController.cs
    - Models
    - Global.asax
    - packages.config
    - Web.config

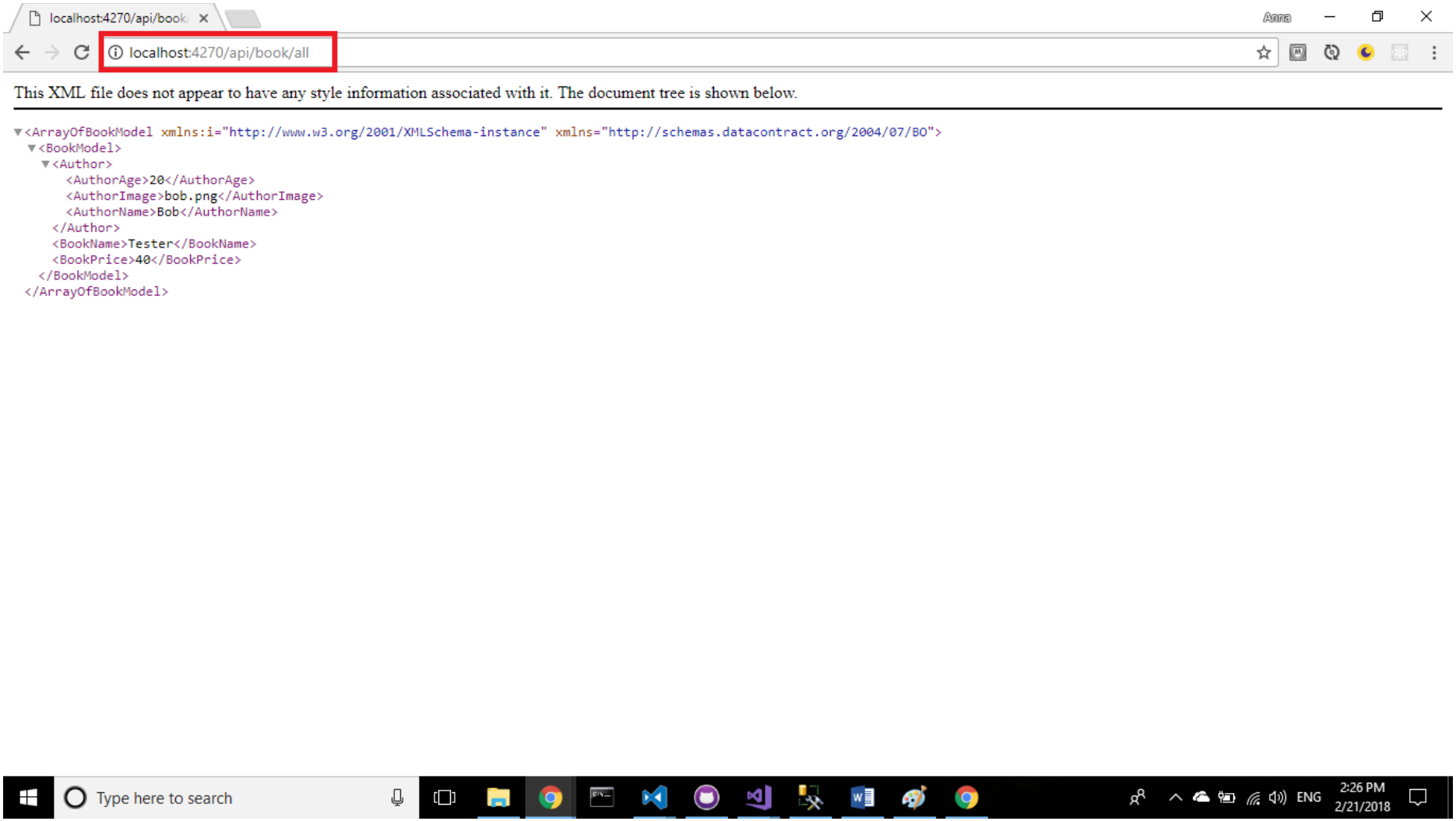
This item does not support previewing | Ln 35 | Col 1 | Ch 1 | INS | Add to Source Control

Type here to search | ENG | 2:28 PM 2/21/2018

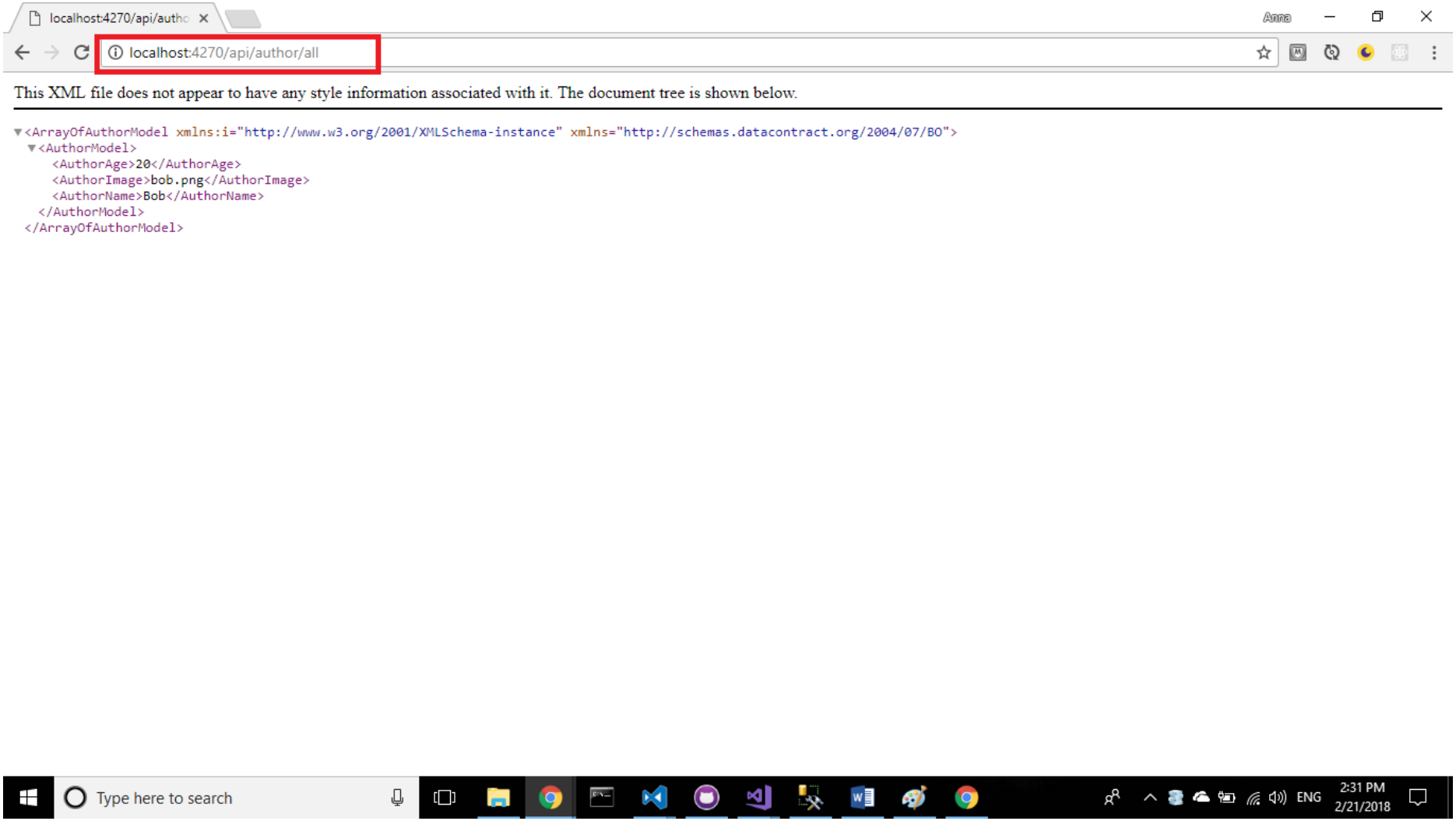
Step 16– add the relevant content to the controller

```
using BLL;
using BO;
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace web_api.Controllers
{
    [RoutePrefix("api/author")]
    public class AuthorController : ApiController
    {
        AuthorManager logic = new AuthorManager();

        [HttpGet]
        [Route("all")]   // access link : /api/author/all
        public HttpResponseMessage GetAllAuthors()
        {
            {
                try
                {
                    List<AuthorModel> authors = logic.GetAllAuthors();
                    return Request.CreateResponse(HttpStatusCode.OK, authors);
                }
                catch (Exception ex)
                {
                    return Request.CreateErrorResponse(HttpStatusCode.InternalServerError, ex);
                }
            }
        }

    }
}
```

Step 17– run the web api application, and write in the browser the url that gets all books



Step 18– run the web api application, and write in the browser the url that gets all authors



GOOD LUCK!!!