

---

# *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-api יש 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 Aauthor (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 Aauthor 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 Aauthor 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 post ו put 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*

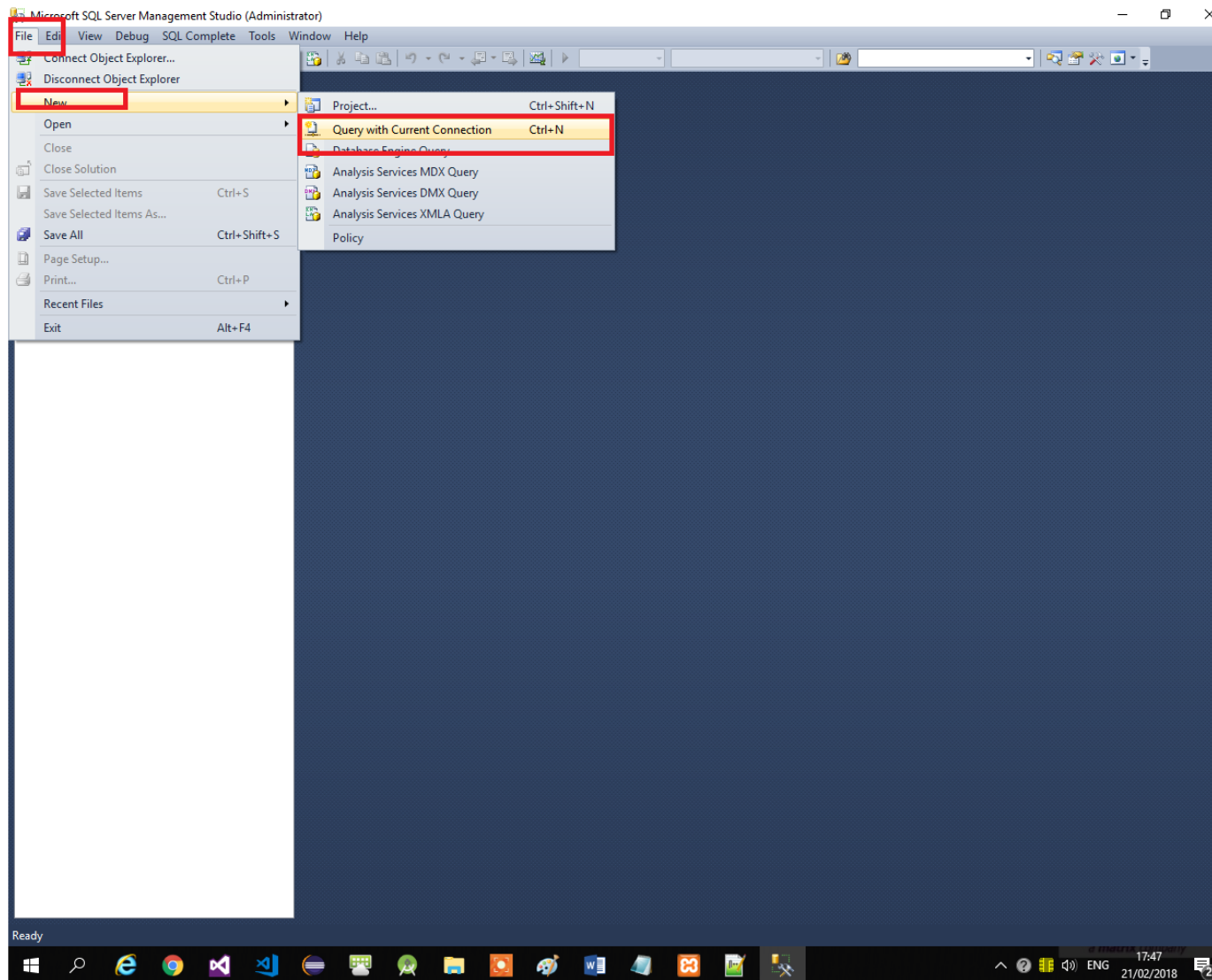
---

```
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*

---

```
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

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'localhost (SQL Server 12.0.2000 - 303\_INS\jbt)', including 'Databases', 'System Databases', 'Database Snapshots', 'AdventureWorks2014', 'asfas', 'BookStore', 'Database Diagrams', 'Tables', 'System Tables', 'FileTables', 'dbo.Authors', 'dbo.Books', 'Views', 'Synonyms', 'Programmability', 'Service Broker', 'Storage', 'Security', 'Northwind', 'pubs', 'Security', 'Server Objects', 'Replication', 'AlwaysOn High Availability', 'Management', 'Integration Services Catalogs', and 'SQL Server Agent'.

The main window shows the 'SQLQuery1.sql - loc...Store (303\_INS\jbt)\*' script. The script contains the following SQL code:

```
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].
);
```

The 'Messages' pane at the bottom shows the command(s) completed successfully. The status bar at the bottom indicates 'Query executed successfully.' and 'localhost (12.0 RTM) | 303\_INS\jbt (55) | BookStore | 00:00:00 | 0 rows'.

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

DESKTOP-RLP7NAJ.B...Store - Diagram\_0\* × SQLQueryForCreate...LP7NAJ\Anna (55))\*

### Authors

	Column Name	Data Type	Allow Nulls
🔑	AuthorID	int	<input type="checkbox"/>
	AuthorAge	int	<input type="checkbox"/>
	AuthorName	nvarchar(20)	<input type="checkbox"/>
	AuthorImage	nvarchar(MAX)	<input type="checkbox"/>
			<input type="checkbox"/>



### Books

	Column Name	Data Type	Allow Nulls
🔑	BookID	int	<input type="checkbox"/>
	BookName	nvarchar(15)	<input type="checkbox"/>
	BookPrice	decimal(18, 0)	<input type="checkbox"/>
	AuthorID	int	<input type="checkbox"/>
			<input type="checkbox"/>

---

*Step 6– add records to the DB*

---

DESKTOP-RLP7NAJ.B...ore - dbo.Authors

	AuthorID	AuthorAge	AuthorName	AuthorImage
	1	20	Bob	bob.png
▶*	NULL	NULL	NULL	NULL

add record to the dataBase

DESKTOP-RLP7NAJ.B...Store - dbo.Books

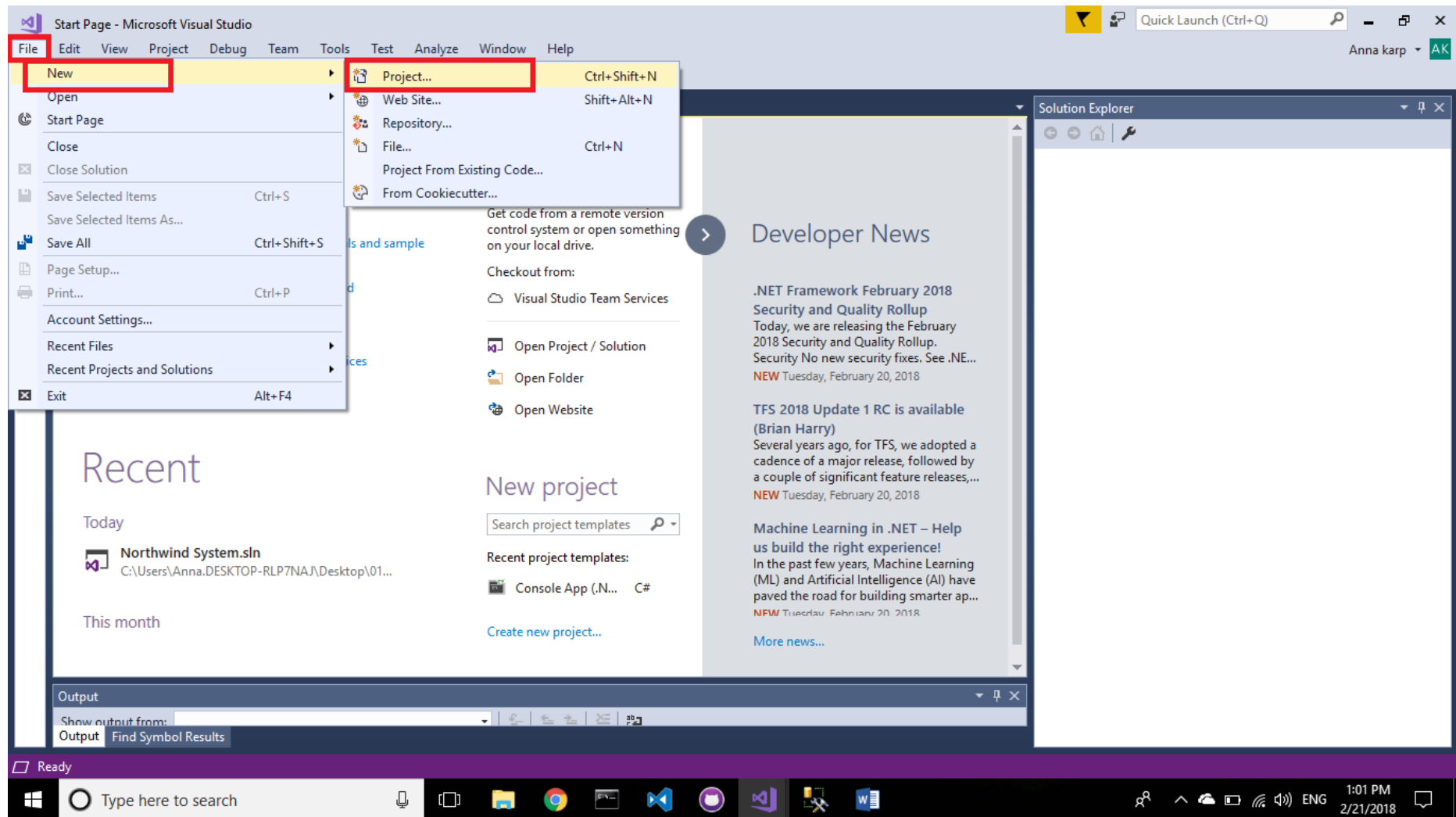
	BookID	BookName	BookPrice	AuthorID
	1	Tester	40	1
▶*	NULL	NULL	NULL	NULL

---

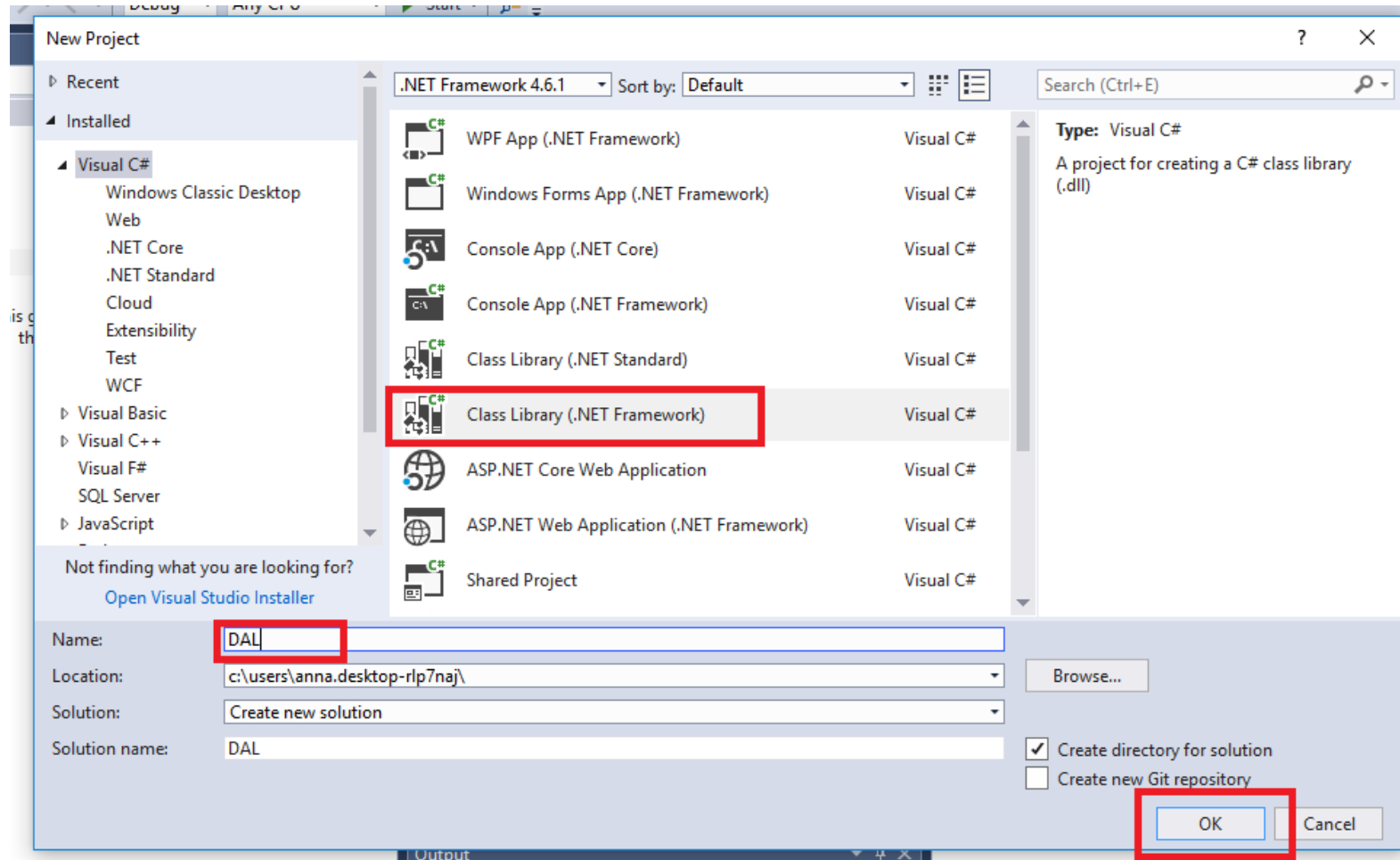
# *Part 2- the DAL*

---

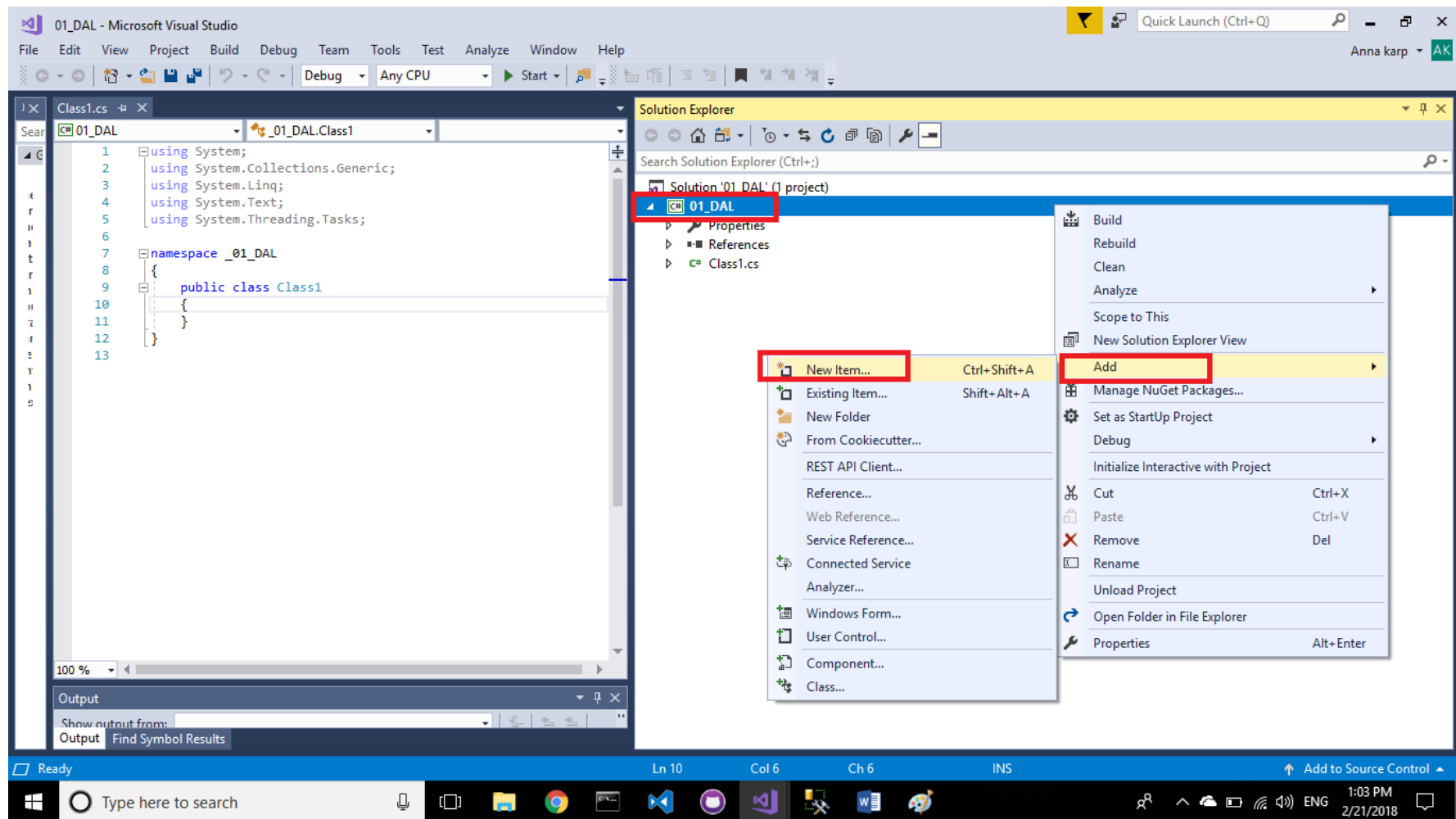
## Step 1 – open a new project in visual studio



## Step 2– open a new Class library project

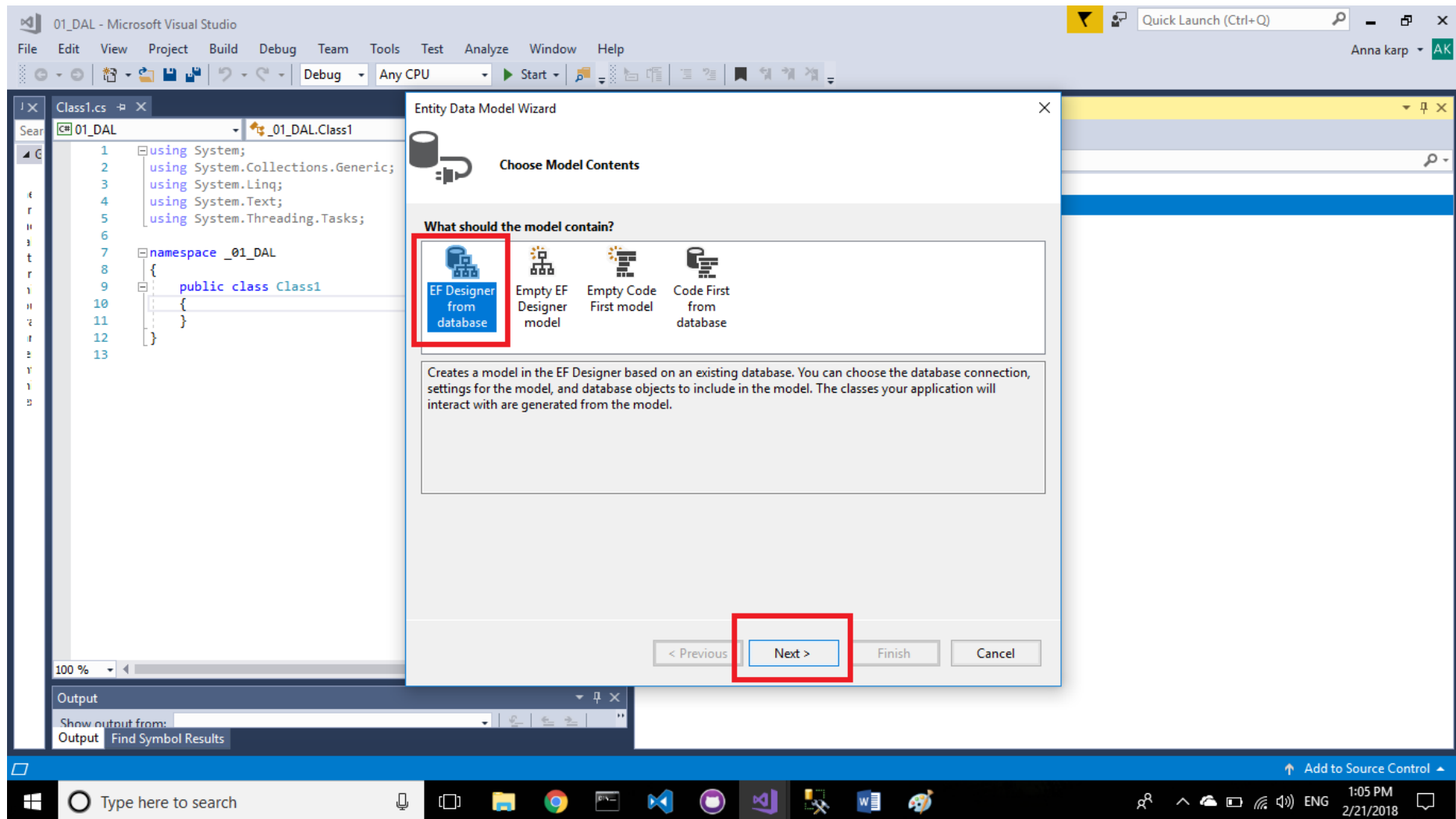


### Step 3 – start adding the EF

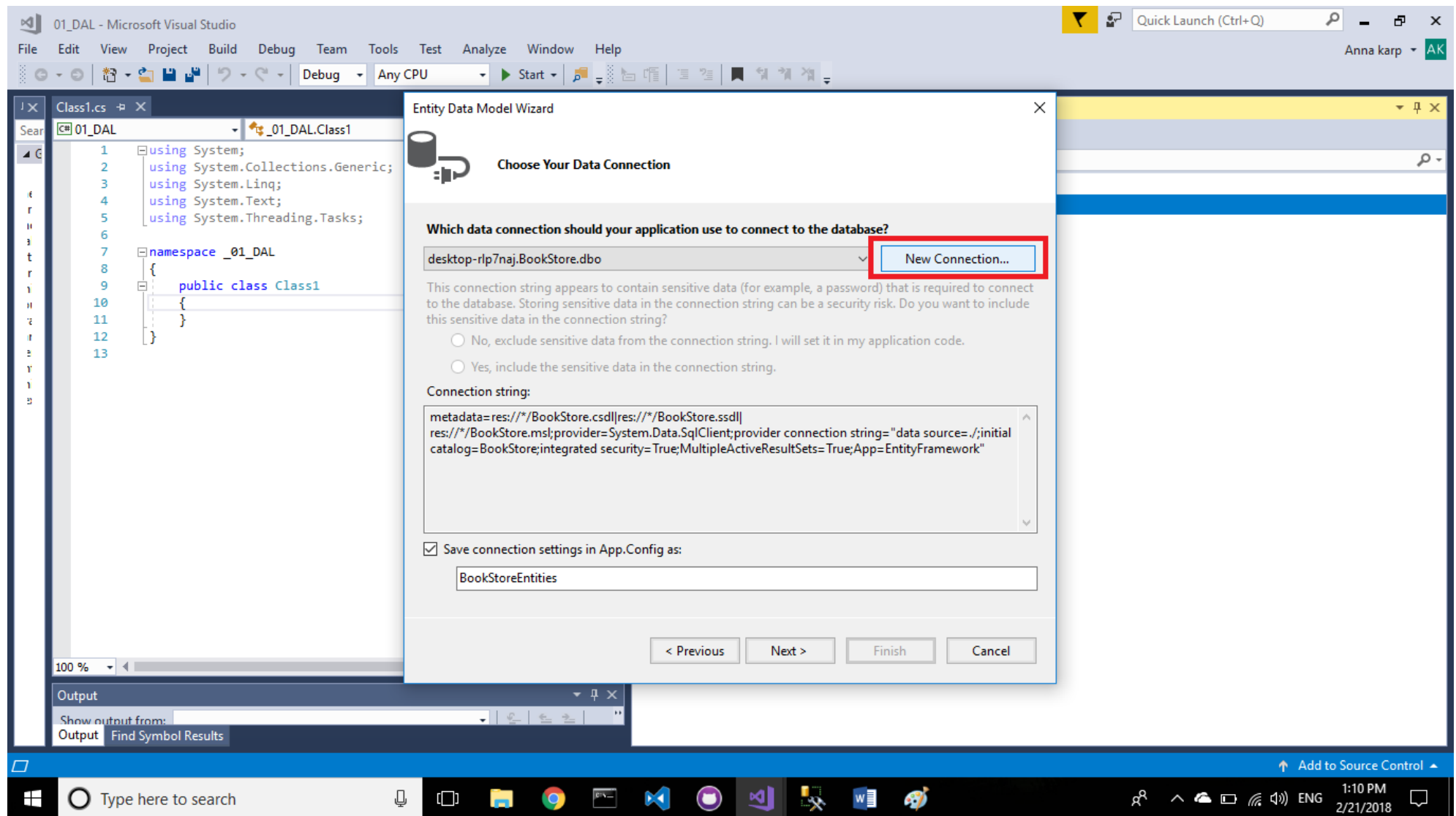




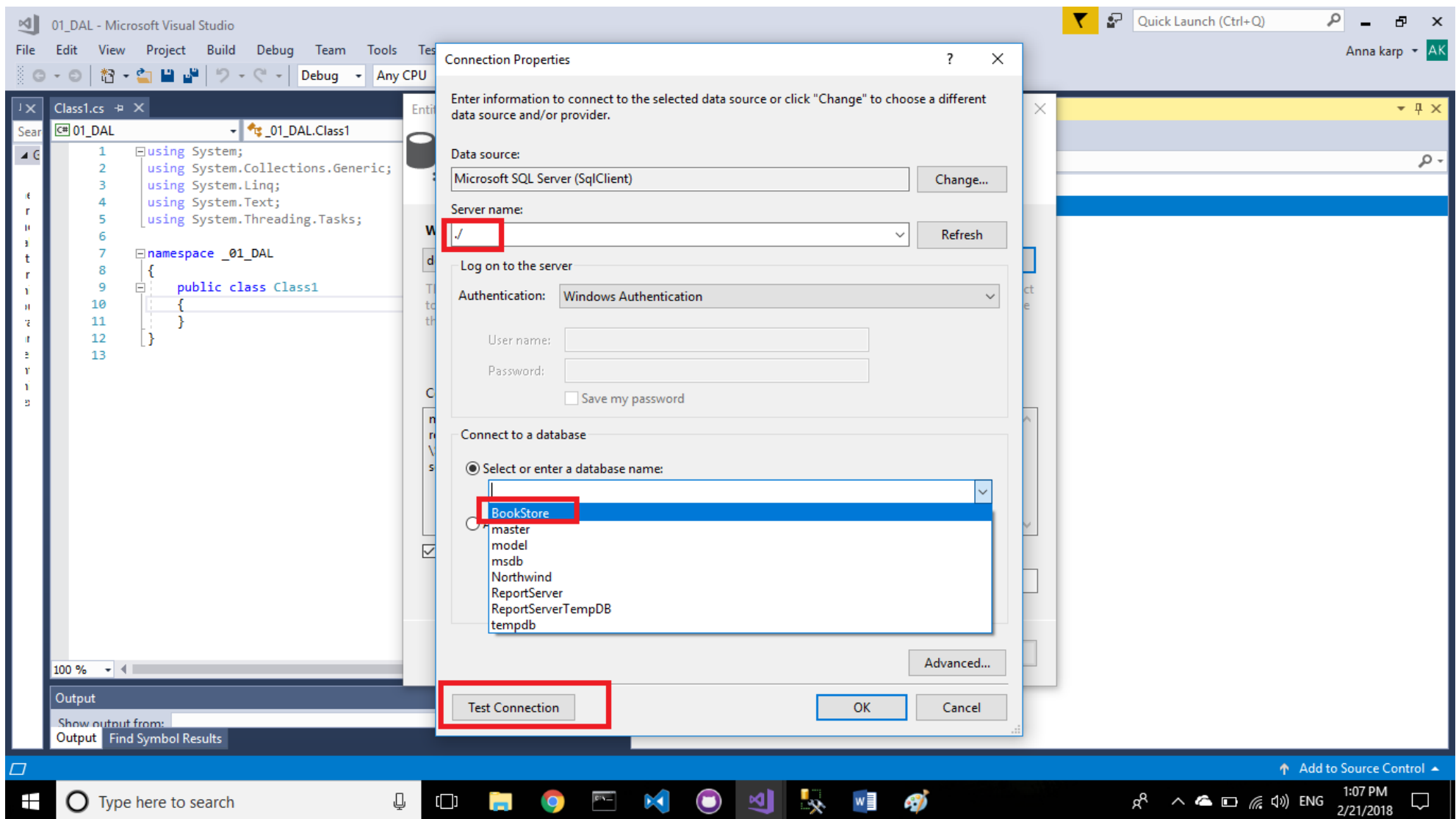
### Step 4- select ef from db



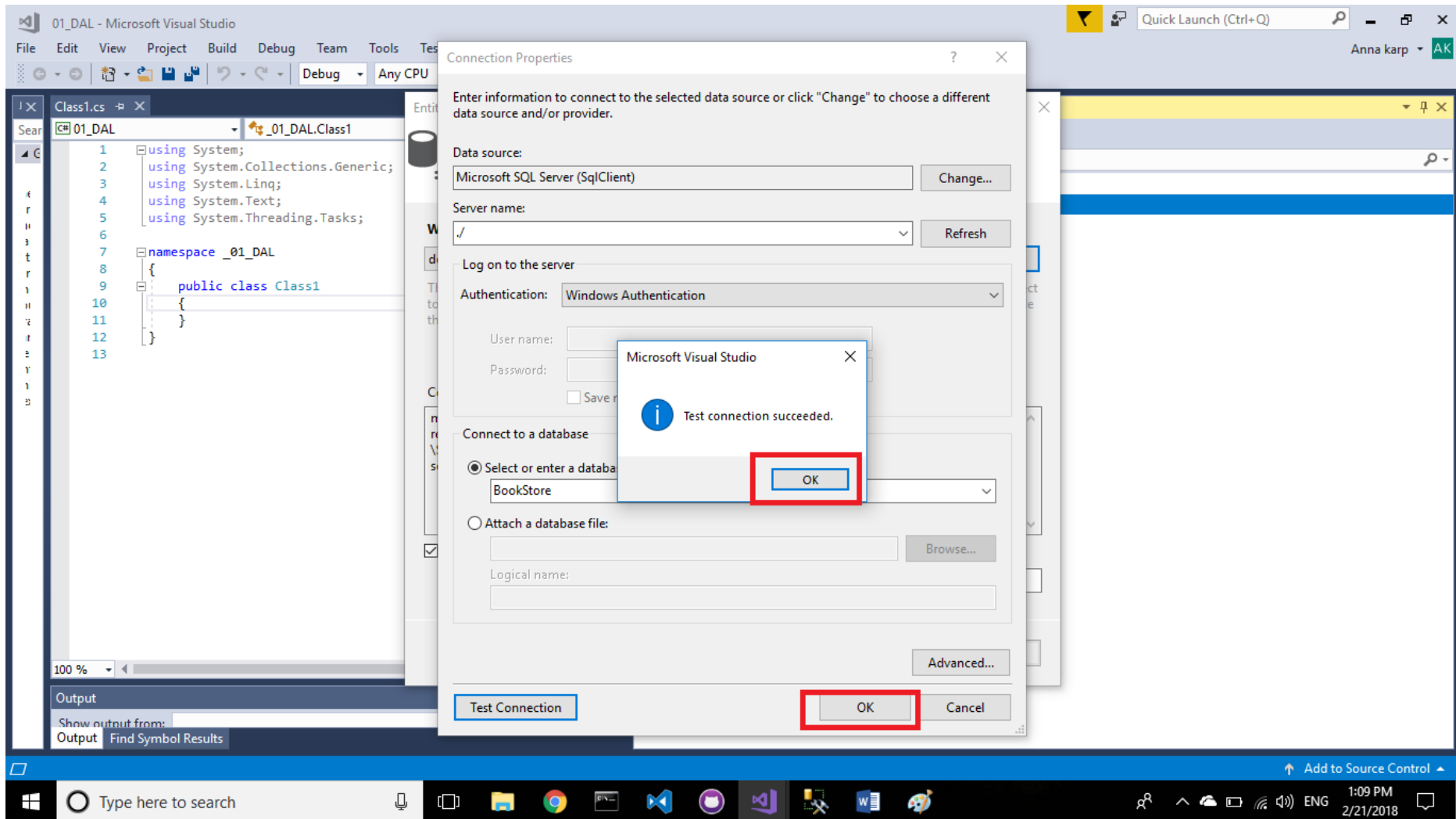
### Step 5- add a new sql connection



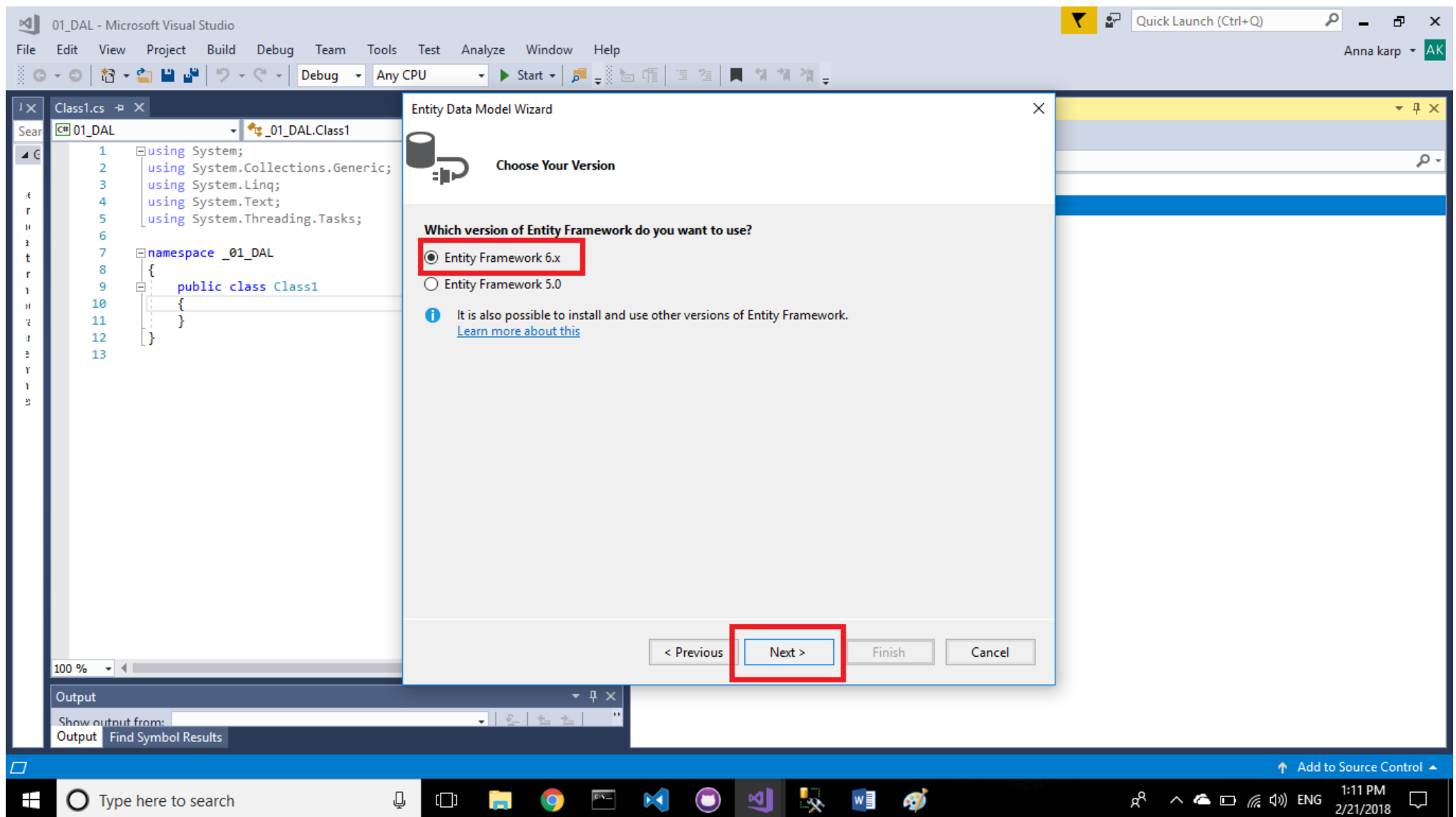
*Step 6– choose your server and then select the “BookStore” DB*



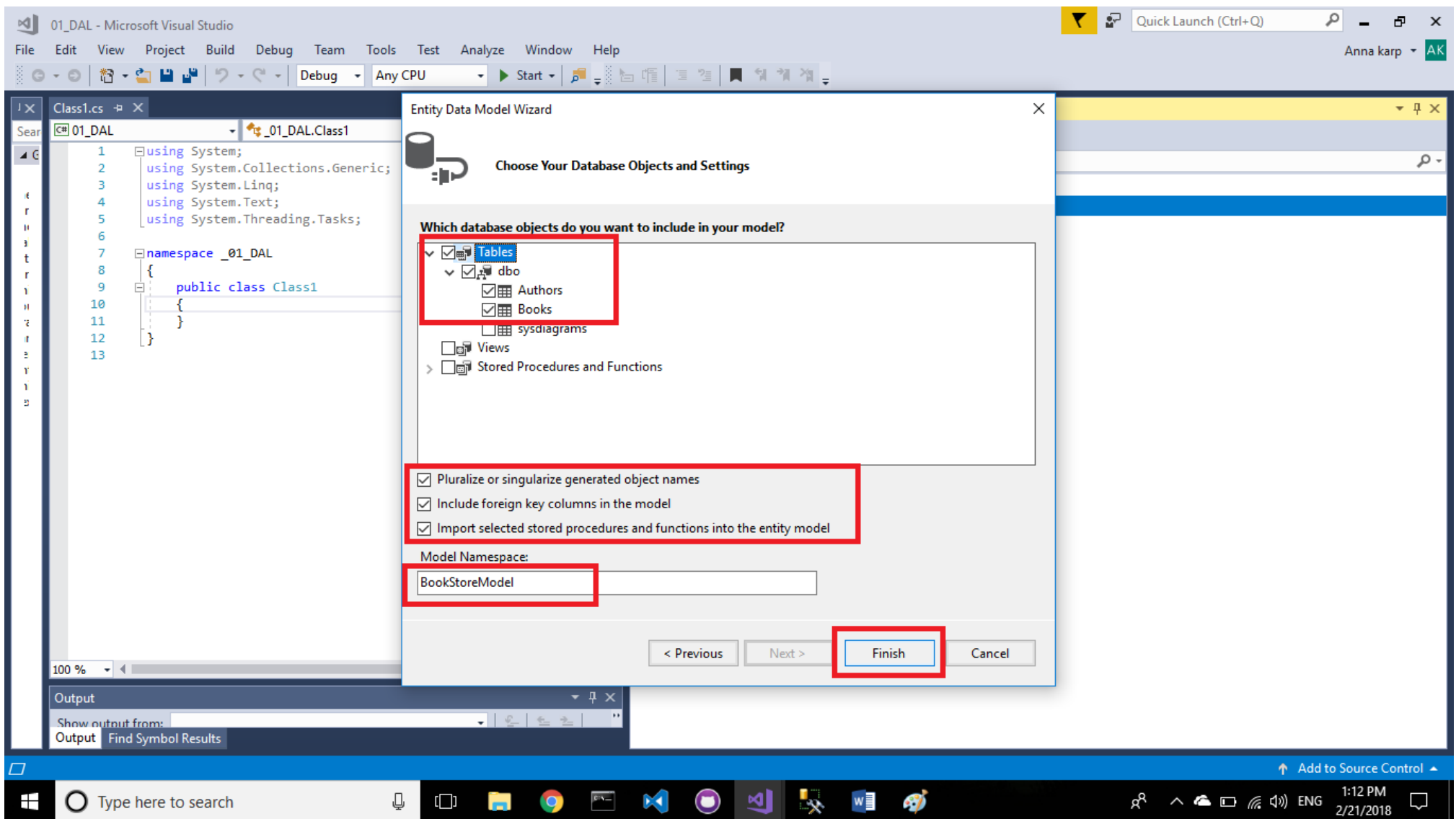
## Step 7—test the connection



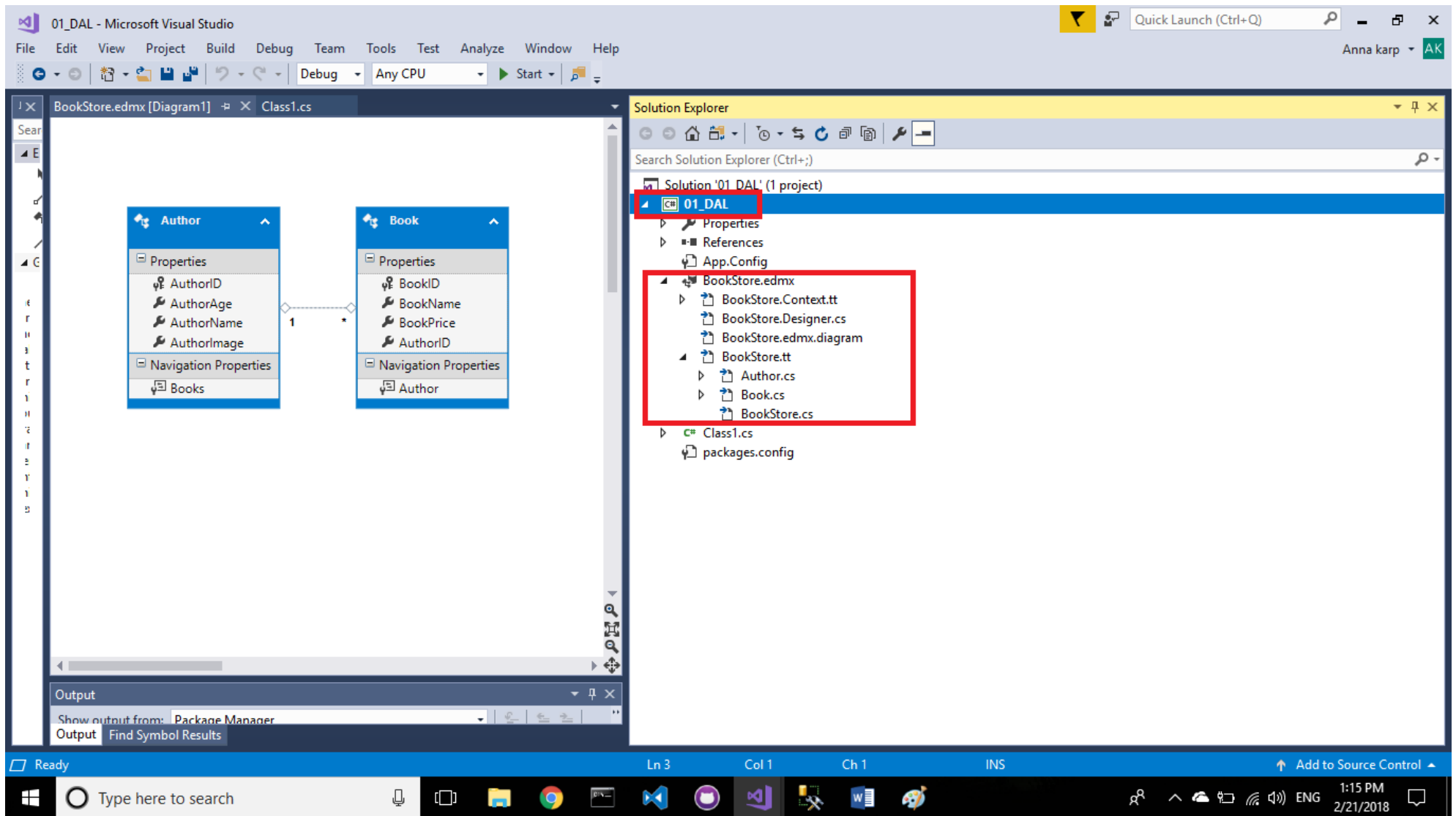
### Step 8— select the ef version



### Step 9– select the relevant tables



*Step 10— you added the ef successfully*



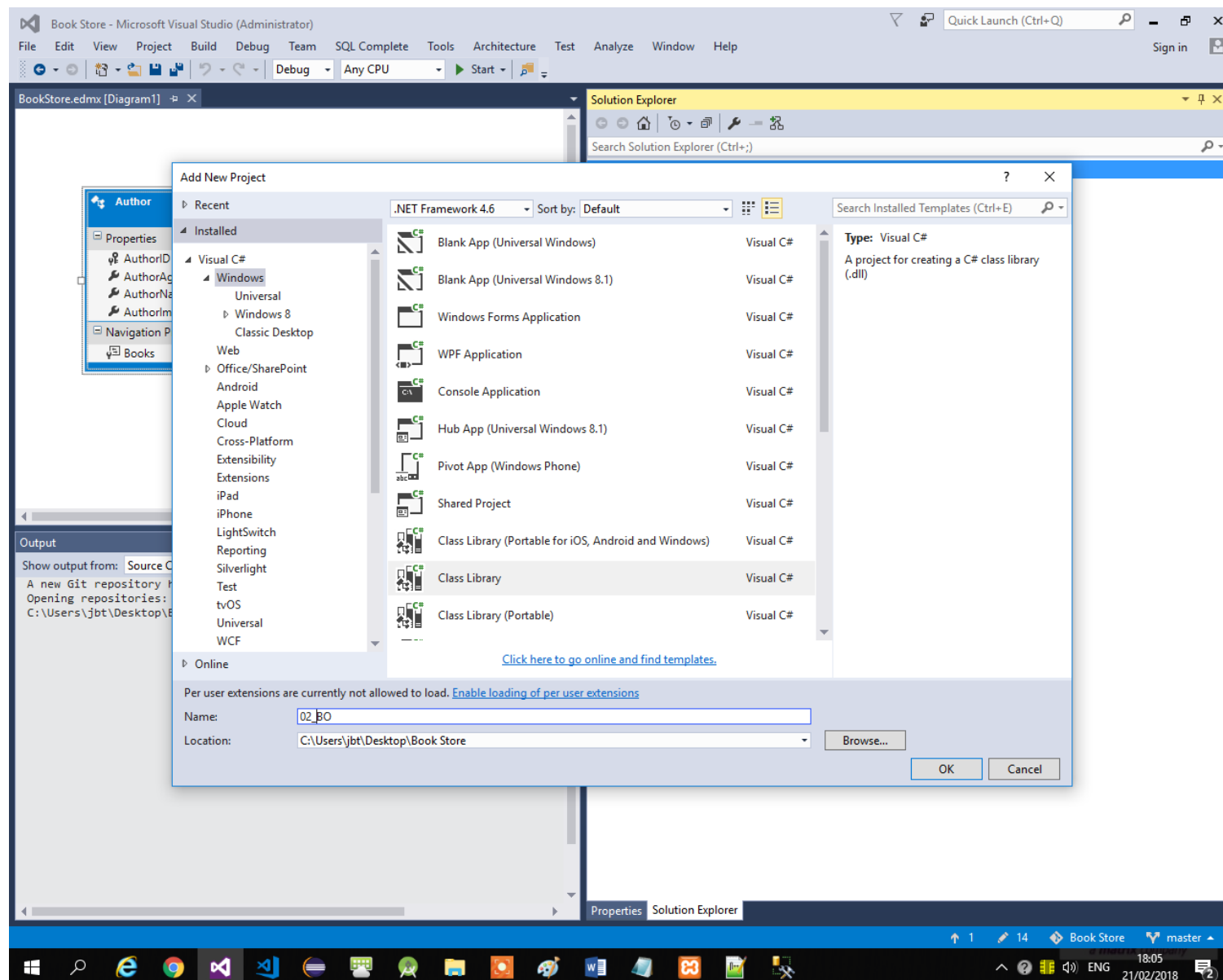
---

# *Part 3- create the BOL*

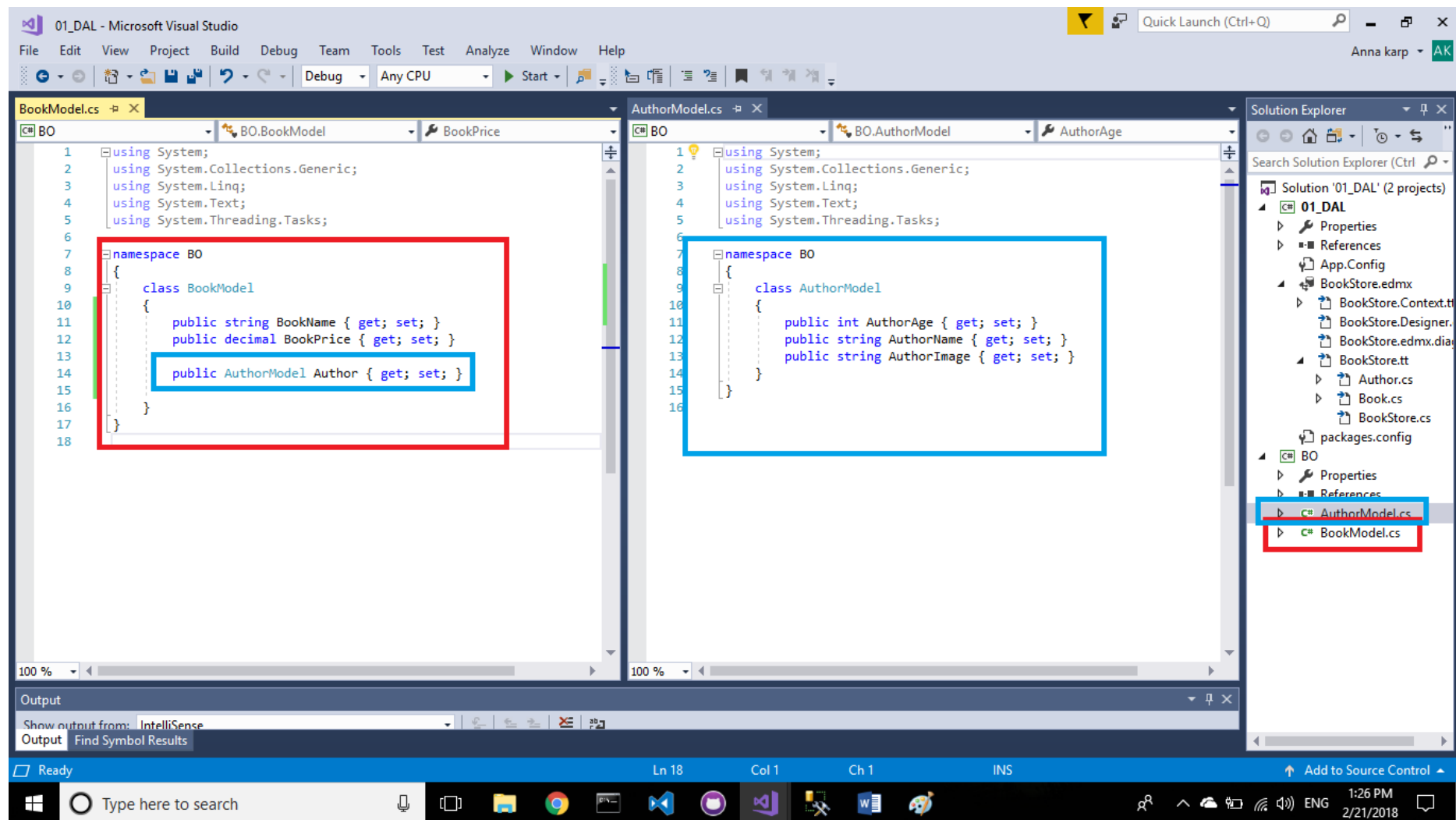
---



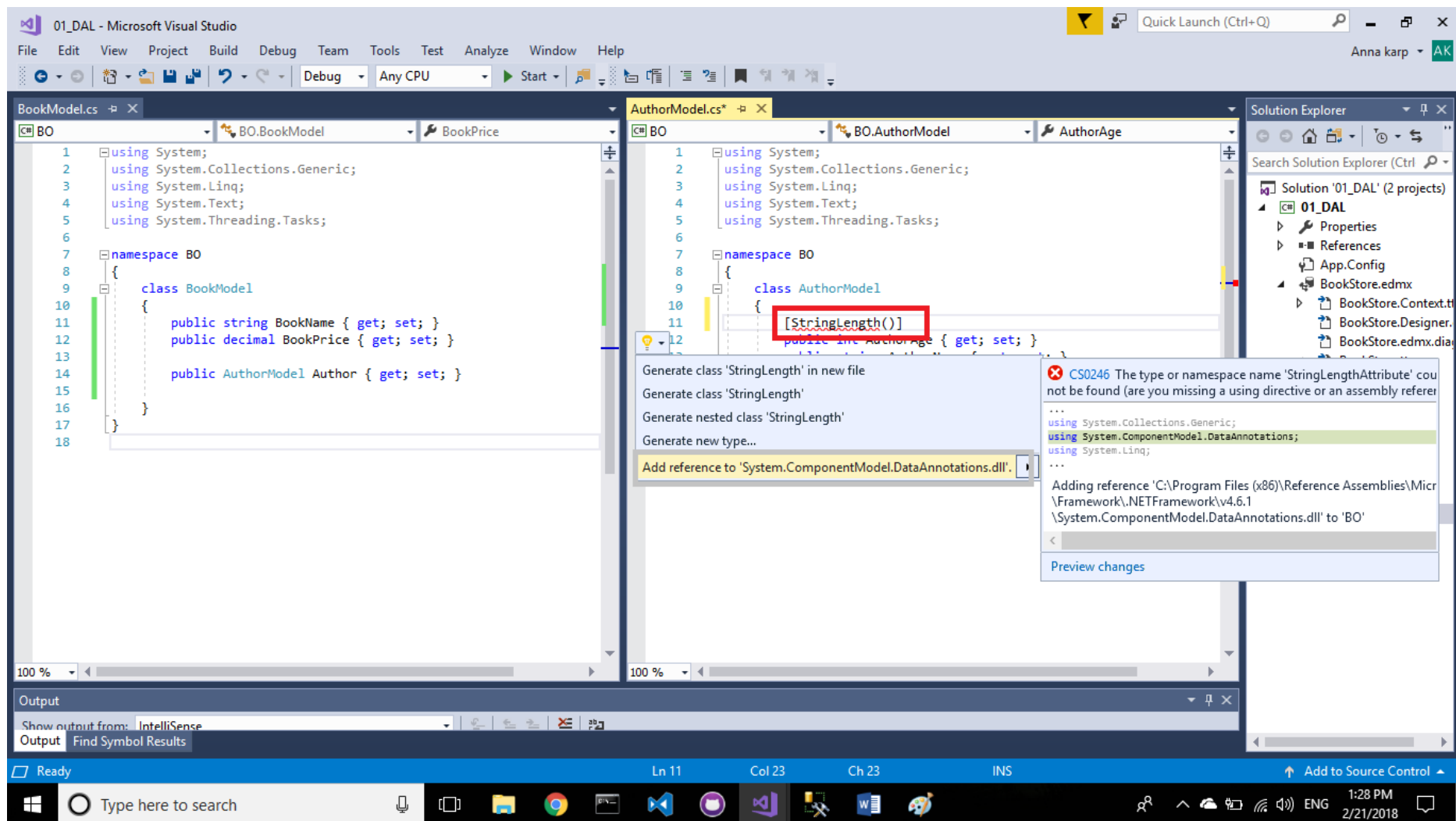
## Step 1—add to the current solution a new Class library project



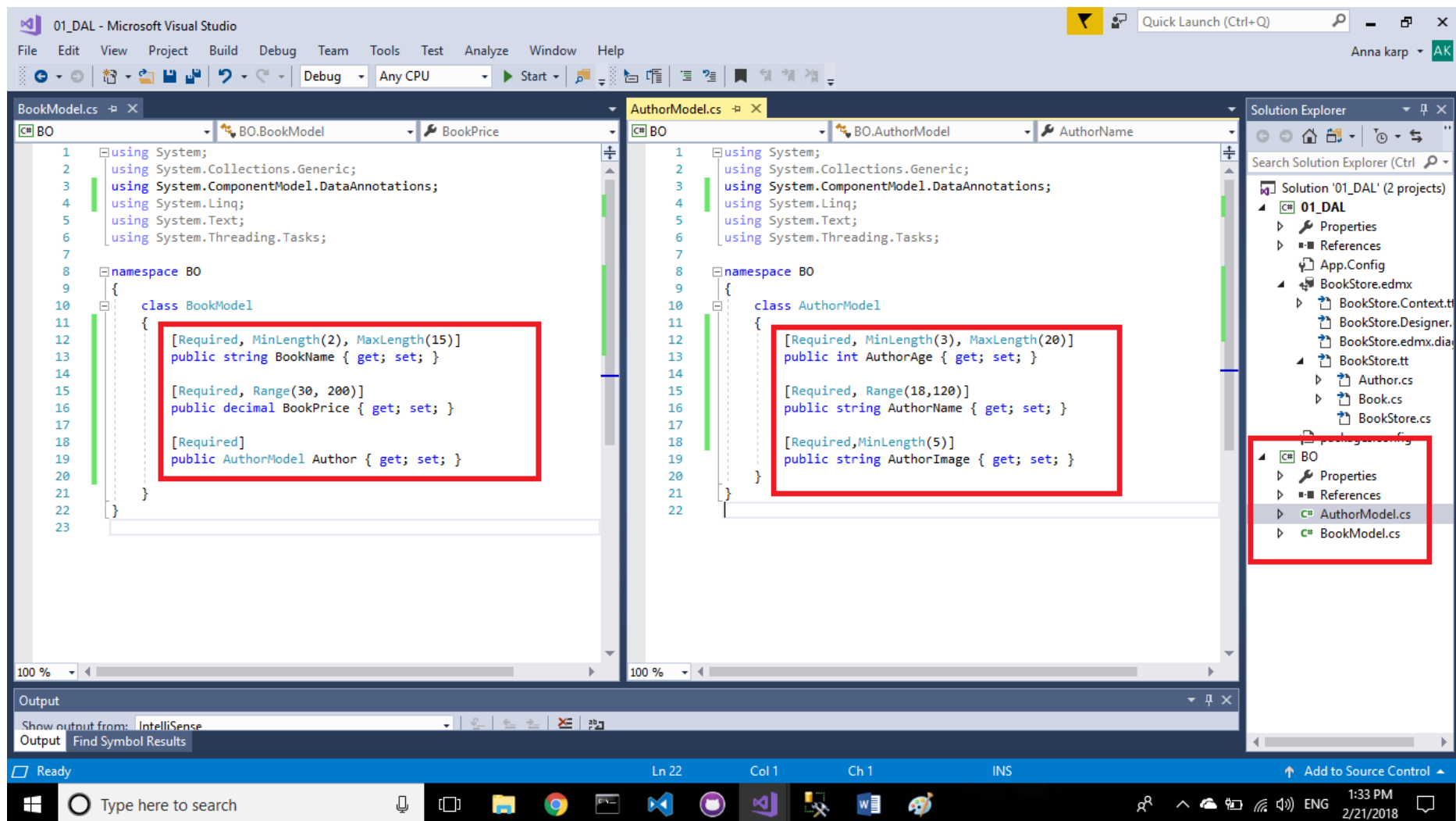
*Step 2— create 2 classes with the relevant names and properties (according to the DB tables)*



### 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*

The screenshot shows the Microsoft Visual Studio IDE with two C# files open: AuthorModel.cs and BookModel.cs. Both files are part of a project named '01\_DAL' and are in the 'BO' namespace. The 'public' keyword is highlighted with a red box in both files, indicating the step to set access modifiers to 'public' for other DLLs to use these classes.

**AuthorModel.cs**

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace BO
9 {
10     public class AuthorModel
11     {
12         [Required, MinLength(3), MaxLength(20)]
13         public int AuthorAge { get; set; }
14
15         [Required, Range(18, 120)]
16         public string AuthorName { get; set; }
17
18         [Required, MinLength(5)]
19         public string AuthorImage { get; set; }
20     }
21 }
22
```

**BookModel.cs**

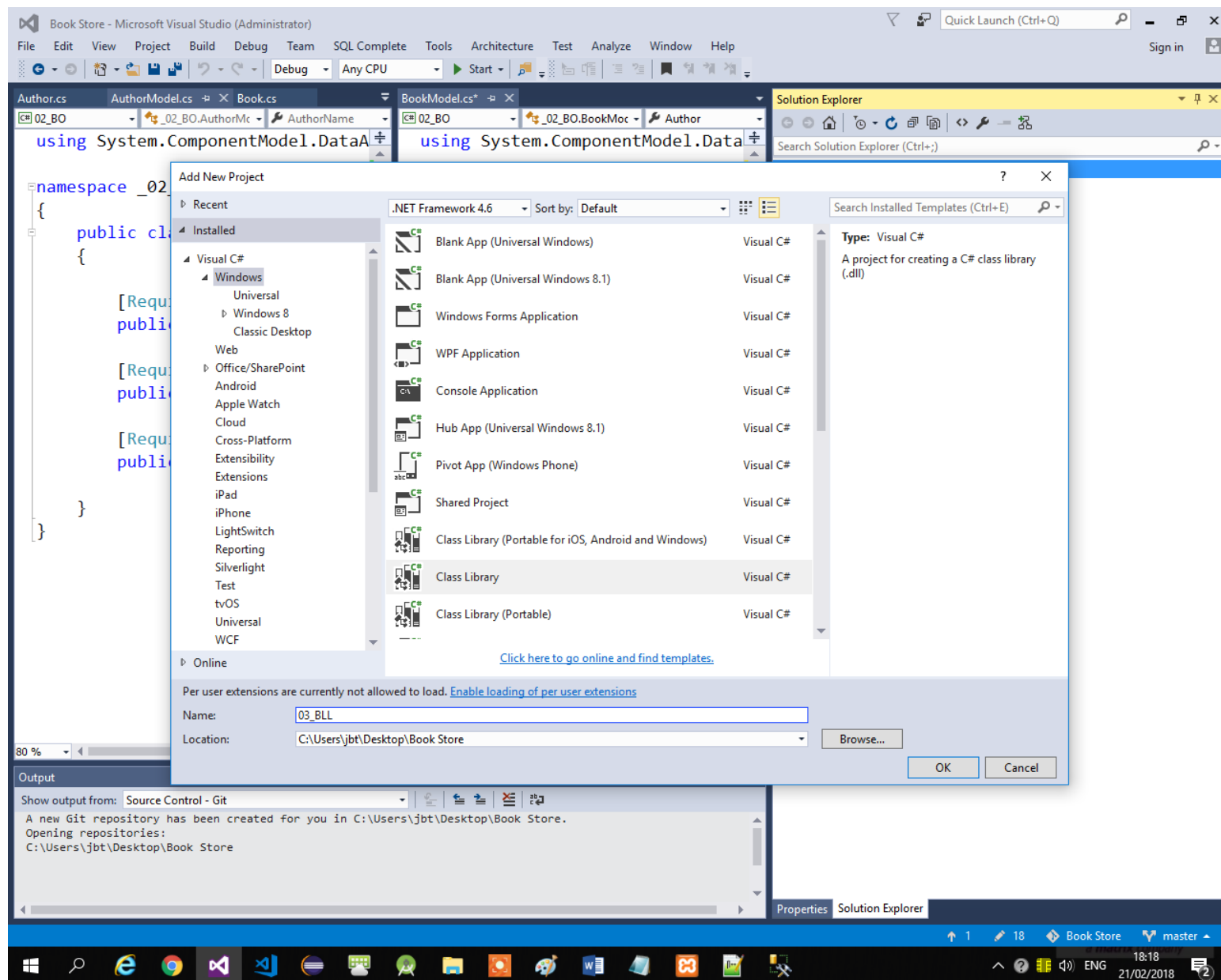
```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace BO
9 {
10     public class BookModel
11     {
12         [Required, MinLength(2), MaxLength(15)]
13         public string BookName { get; set; }
14
15         [Required, Range(30, 200)]
16         public decimal BookPrice { get; set; }
17
18         [Required]
19         public AuthorModel Author { get; set; }
20     }
21 }
22
23
```

---

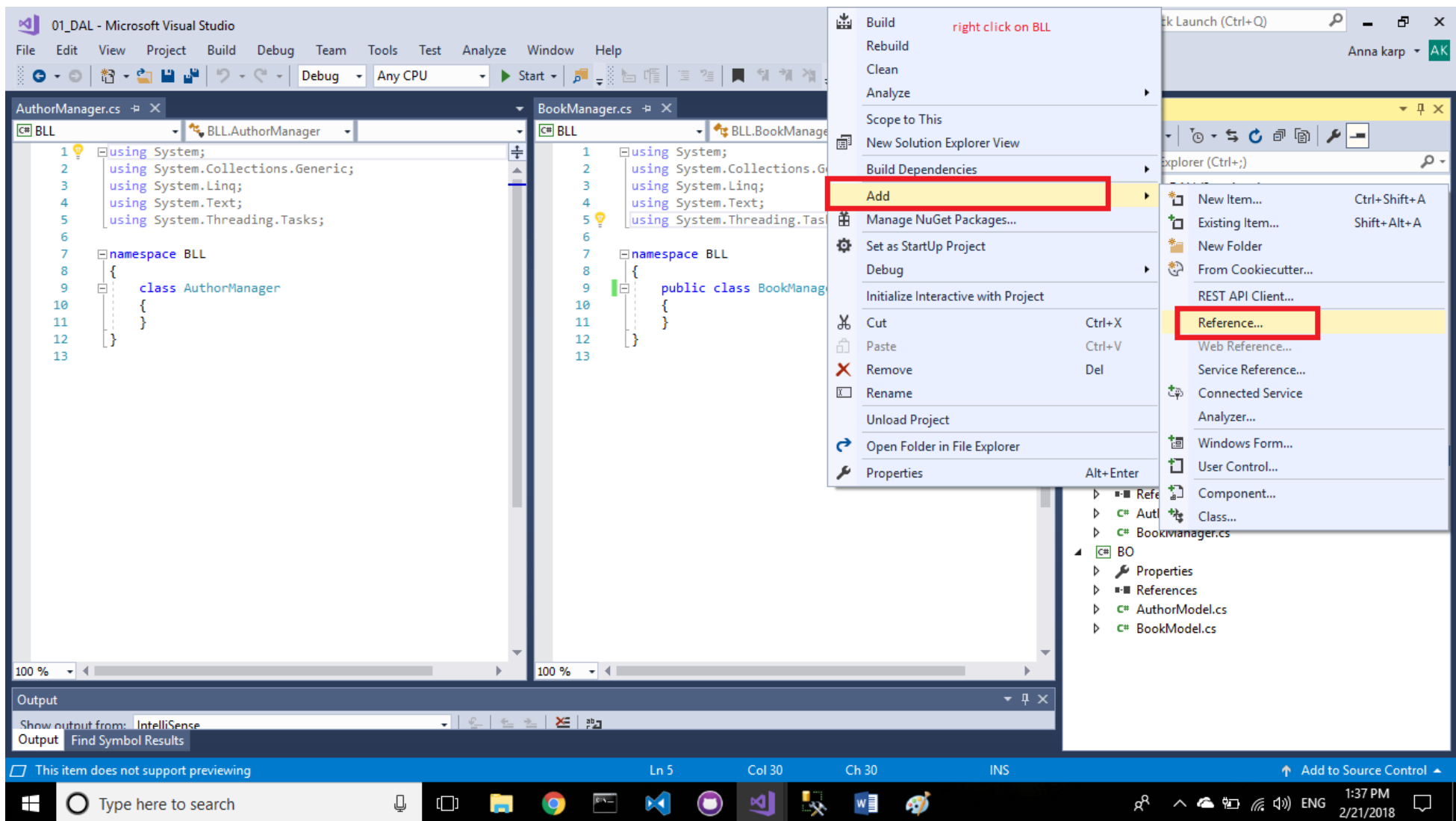
# *Part 4- create the BLL*

---

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

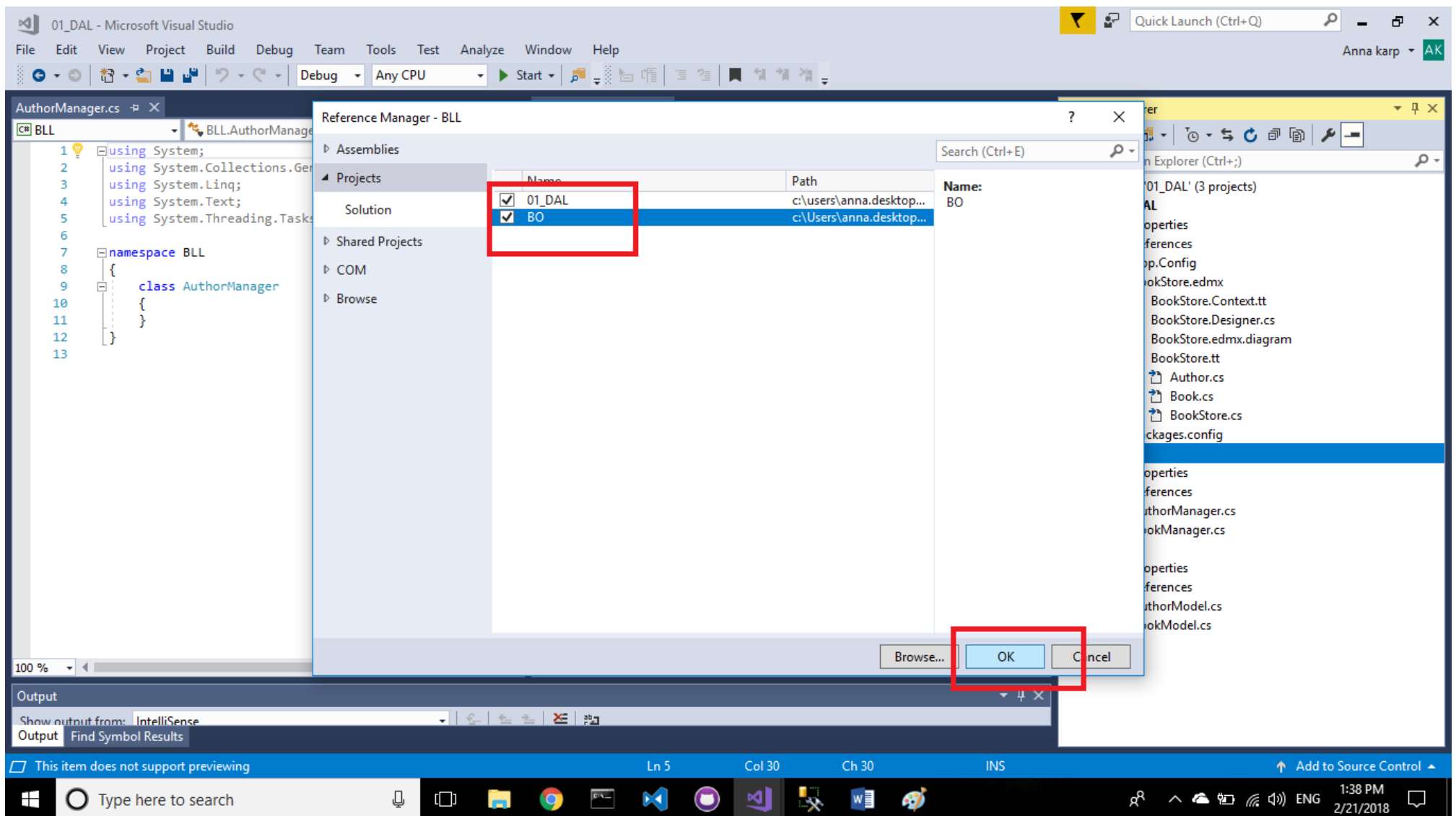


## Step 2– add to this project references





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



---

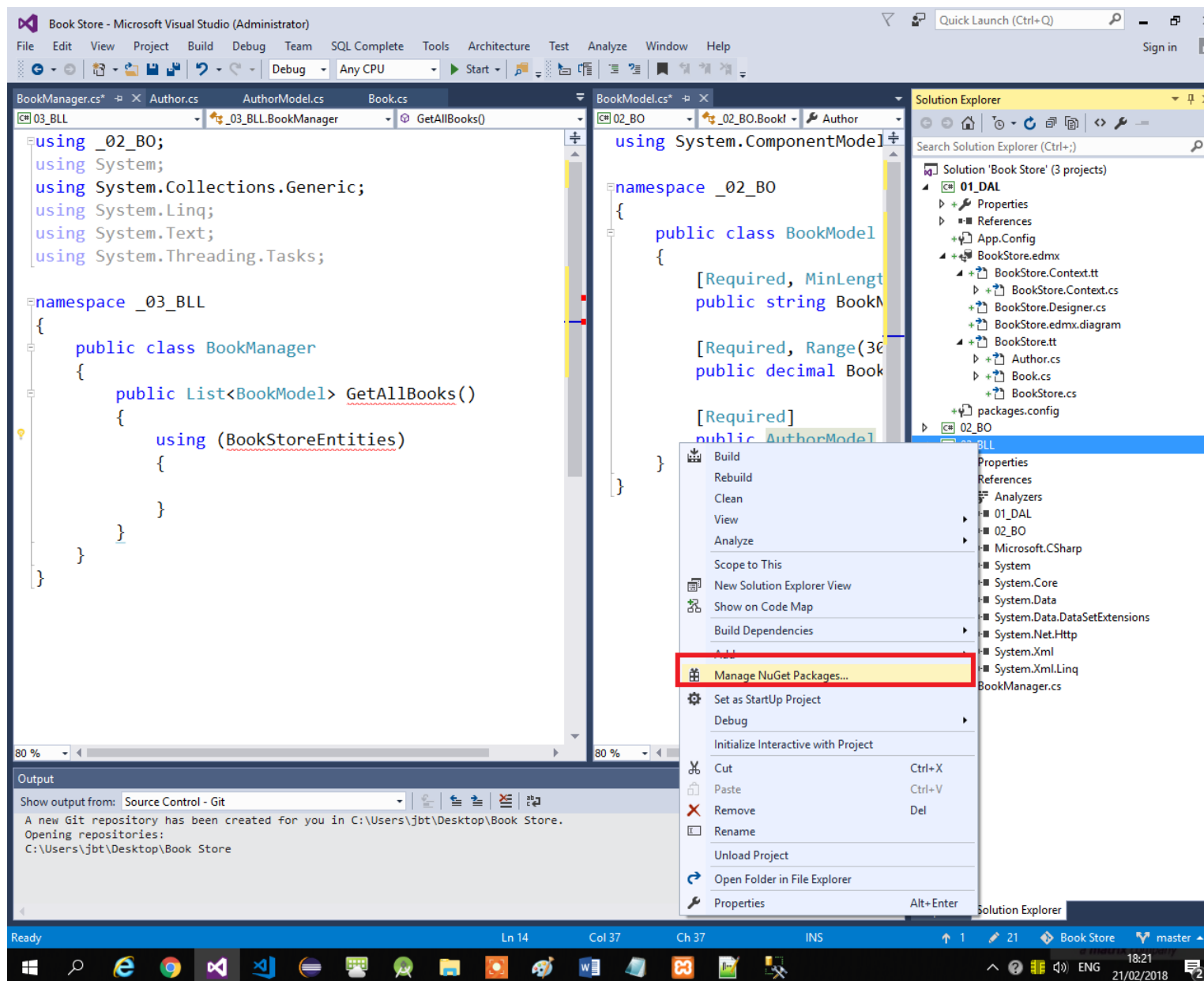
*Step 4– create a BookManager class – note: in the next step you will add ef and fix the error*

---

```
1 using _01_DAL;
2 using B0;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace BLL
10 {
11     public class BookManager
12     {
13
14         // Get all books:
15         public List<BookModel> GetAllOrders()
16         {
17             using(BookStoreEntities db= new BookStoreEntities())
18             {
19                 ret
20                 new
21             }
22         }
23     }
24 }
```

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

The screenshot displays the Microsoft Visual Studio interface with the NuGet Package Manager open. The 'Browse' tab is active, and the search term 'entity' is entered in the search box. The 'EntityFramework' package by Microsoft is selected, and the 'Install' button is highlighted. The Solution Explorer on the right shows the project structure for '01\_DAL'.

**NuGet Package Manager: BLL**

Package source: nuget.org

**EntityFramework**

Version: Latest stable 6.2.0 **Install**

**Description**

Entity Framework is Microsoft's recommended data access technology for new applications.

**Options**

**MySQL.Data.Entity** by Oracle, 630K downloads v6.10.6  
MySQL.Data.Entity.EF6

**Oracle.ManagedDataAccess.EntityFramework** by Oracle v12.2.1100  
The ODP.NET, Managed Driver Entity Framework package for EF 6 applications.

**EntityFramework** by Microsoft, 39.8M downloads v6.2.0  
Entity Framework is Microsoft's recommended data access technology for new applications.

**Microsoft.AspNet.Identity.EntityFramework** by Microsoft, 7 v2.2.1  
ASP.NET Identity providers that use Entity Framework.

**EntityFramework.SqlServerCompact** by Microsoft, 849K down v6.2.0  
Allows SQL Server Compact 4.0 to be used with Entity Framework.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

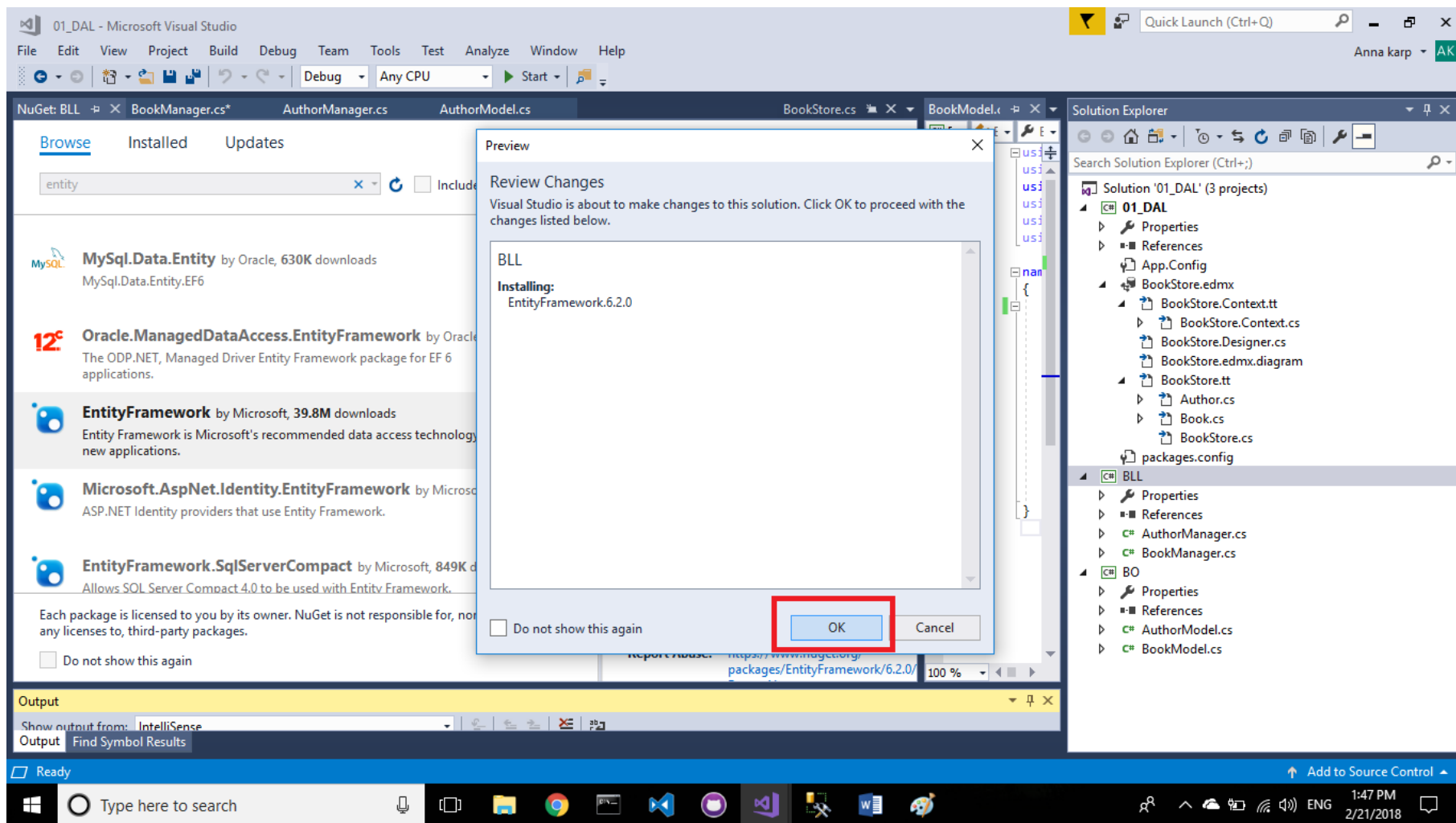
**Solution Explorer**

Search Solution Explorer (Ctrl+;):

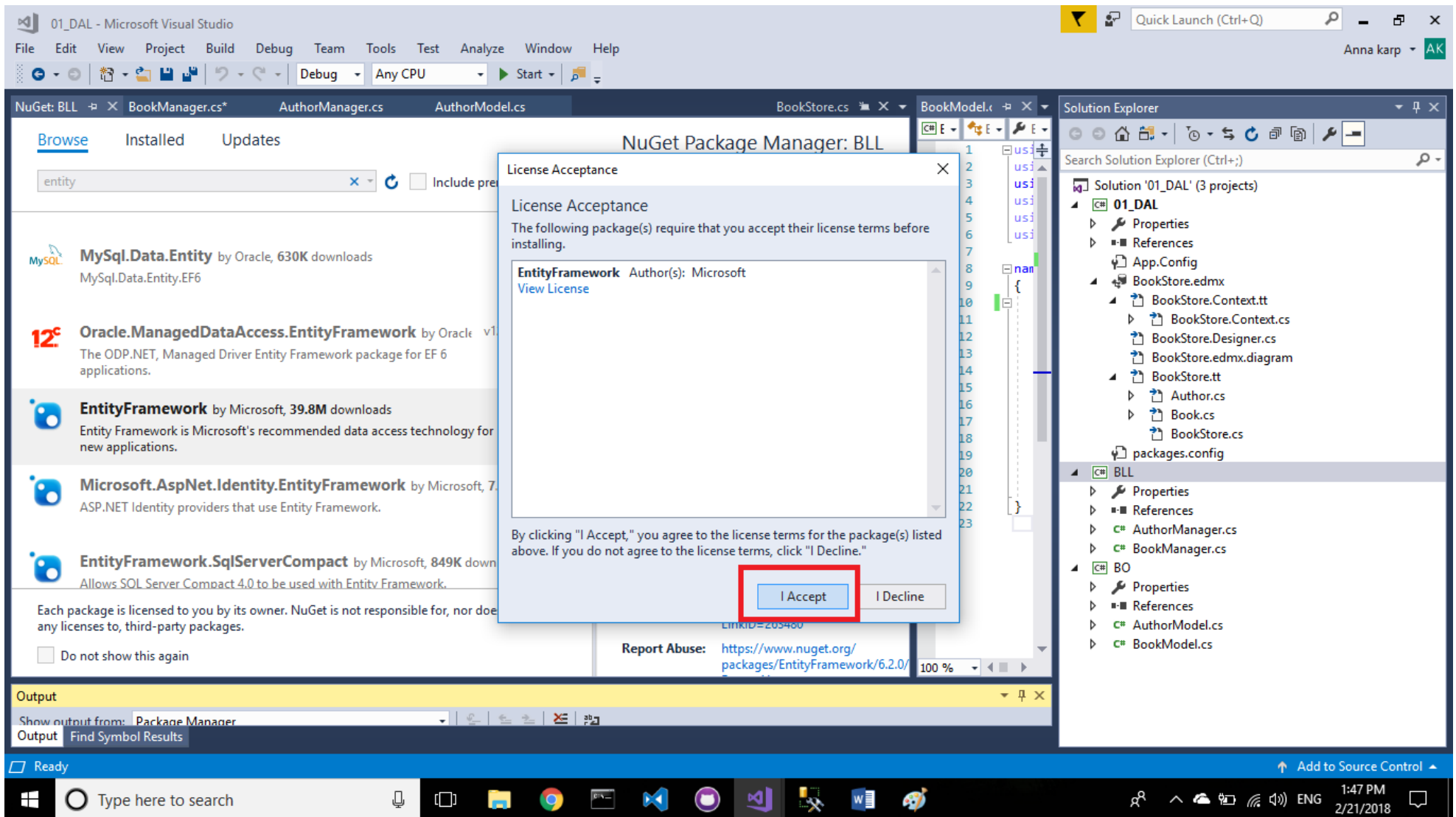
Solution '01\_DAL' (3 projects)

- 01\_DAL
  - Properties
  - References
  - App.Config
  - BookStore.edmx
    - BookStore.Context.tt
      - BookStore.Context.cs
    - BookStore.Designer.cs
    - BookStore.edmx.diagram
  - BookStore.tt
    - Author.cs
    - Book.cs
    - BookStore.cs
  - packages.config
- BLL
  - Properties
  - References
  - AuthorManager.cs
  - BookManager.cs
- BO
  - Properties
  - References
  - AuthorModel.cs
  - BookModel.cs

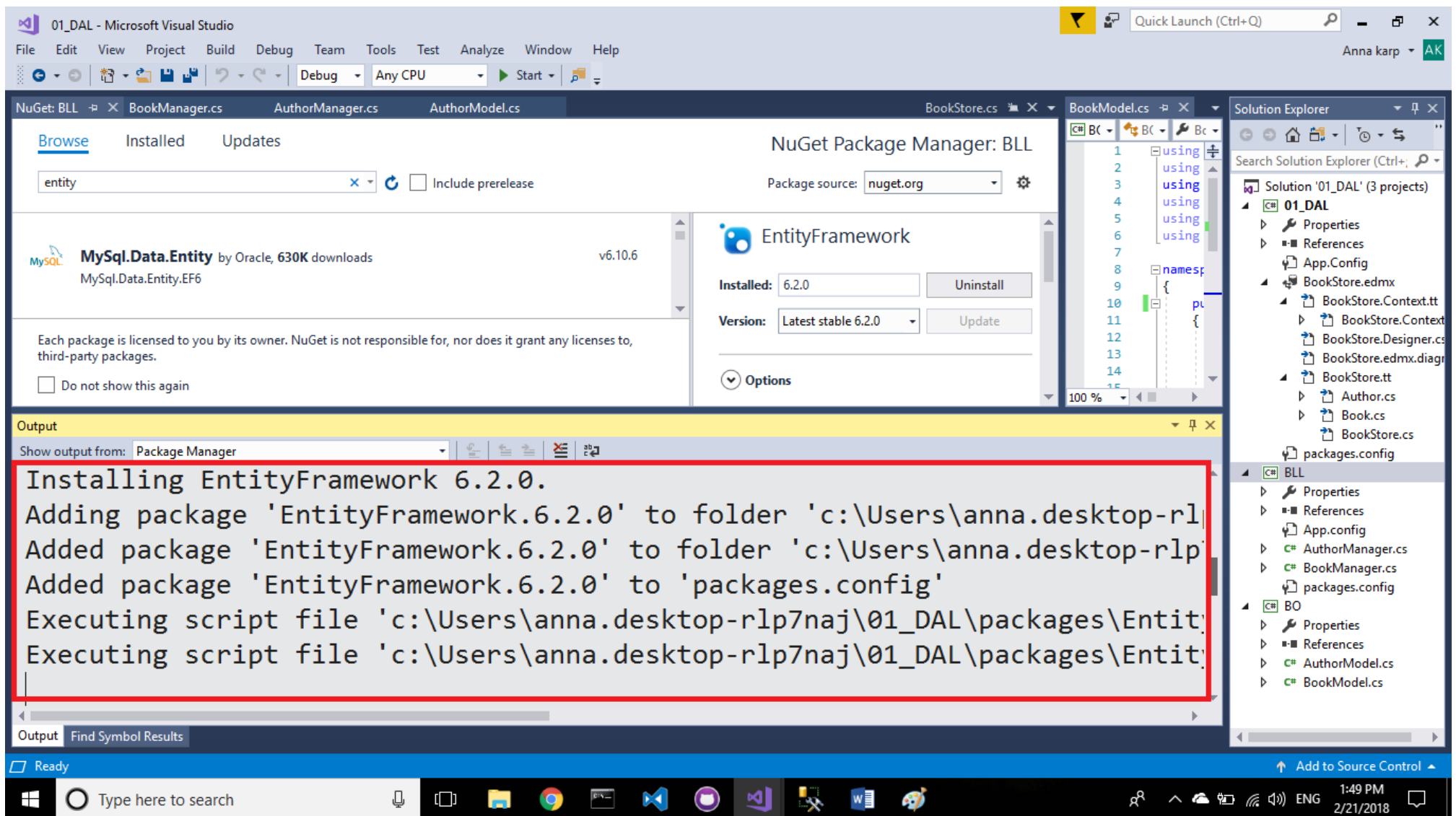
## Step 7– start adding the ef package



## Step 8—add the ef package



*Step 9– the ef package has added successfully*



### Step 10– add the relevant logic to the BookManager and AauthorManager classes

The screenshot displays the Microsoft Visual Studio IDE with two code files open: `BookManager.cs` and `AuthorManager.cs`. Both files are in the `BLL` namespace and implement methods to retrieve data from a database.

**BookManager.cs** (Left Panel):

```
1 using _01_DAL;
2 using BO;
3 using System.Collections.Generic;
4 using System.Linq;
5
6 namespace BLL
7 {
8     public class BookManager
9     {
10         // Get all books:
11         public List<BookModel> GetAllBooks()
12         {
13             using (BookStoreEntities db = new BookStoreEntities())
14             {
15                 return db.Books.Select(b =>
16                     new BookModel
17                     {
18                         BookName = b.BookName,
19                         BookPrice = b.BookPrice,
20                         Author = new AuthorModel()
21                         {
22                             AuthorName = b.Author.AuthorName,
23                             AuthorAge = b.Author.AuthorAge,
24                             AuthorImage = b.Author.AuthorImage
25                         }
26                     }
27                 ).ToList();
28             }
29         }
30     }
31 }
32
33
34
```

**AuthorManager.cs** (Right Panel):

```
2 using BO;
3 using System.Collections.Generic;
4 using System.Linq;
5
6 namespace BLL
7 {
8     class AuthorManager
9     {
10         public List<AuthorModel> GetAllAuthors()
11         {
12             using (BookStoreEntities db = new BookStoreEntities())
13             {
14                 return db.Authors.Select(a =>
15                     new AuthorModel()
16                     {
17                         AuthorName = a.AuthorName,
18                         AuthorAge = a.AuthorAge,
19                         AuthorImage = a.AuthorImage
20                     }
21                 ).ToList();
22             }
23         }
24     }
25 }
26
27
28
29
```

The `Solution Explorer` on the right shows the project structure for `01_DAL`, including `BookStore.edmx`, `BookStore.tt`, `Author.cs`, `Book.cs`, `packages.config`, `BLL`, and `BO`. The `AuthorManager.cs` and `BookManager.cs` files are highlighted in the `BLL` folder.

The status bar at the bottom indicates the current position is Line 29, Column 1, Character 1, Insertion mode.

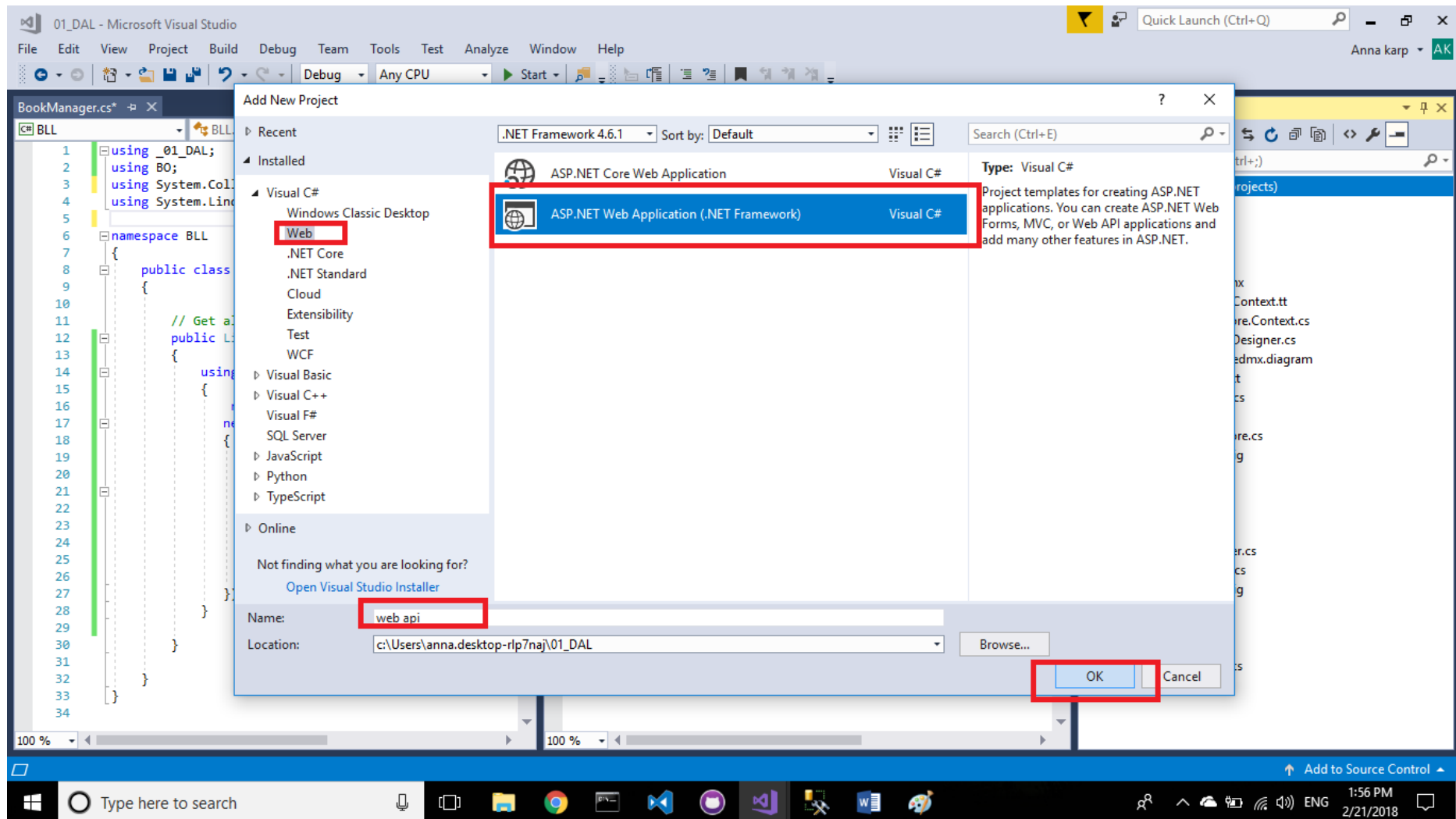


---

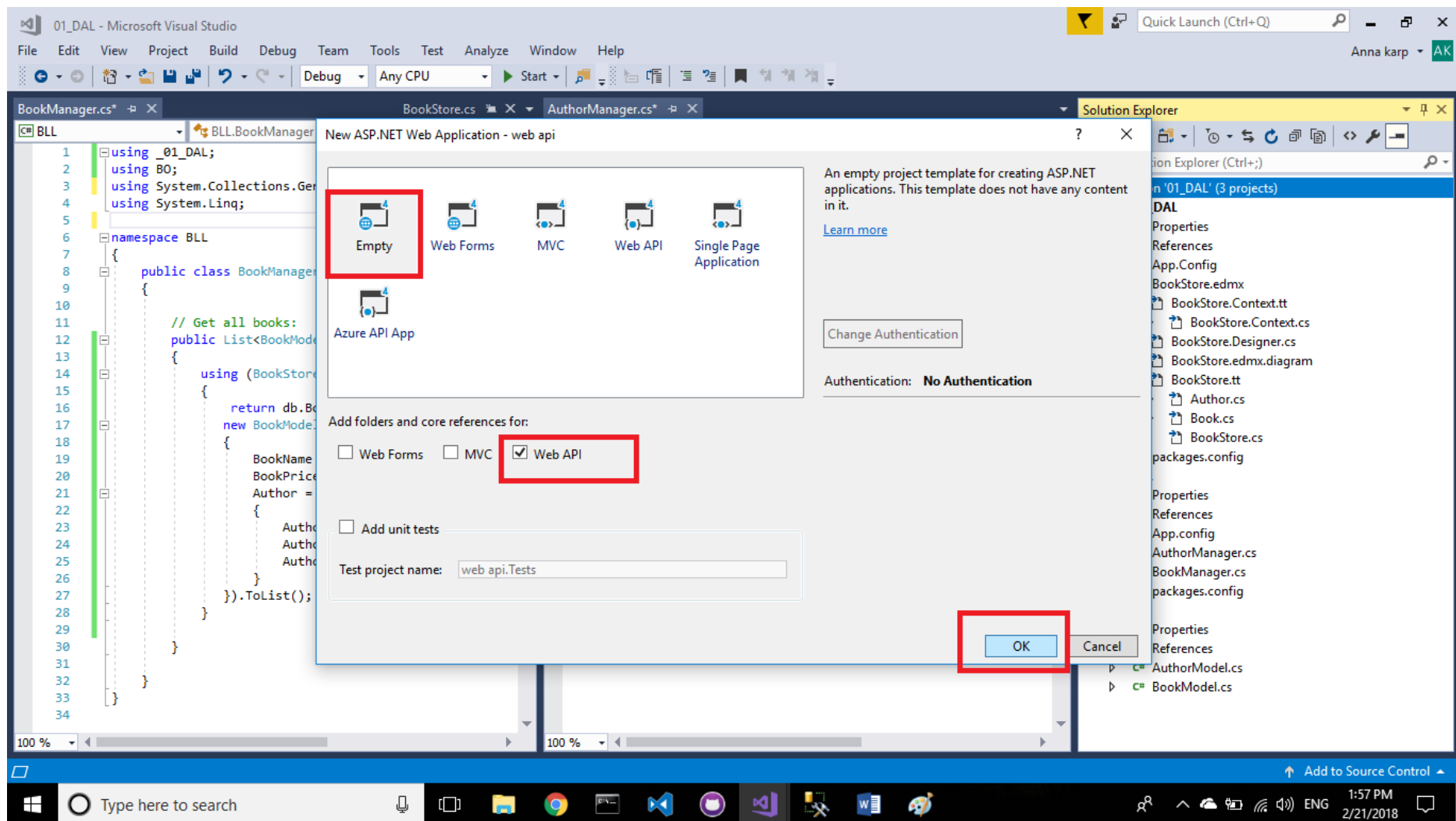
# *Part 5- create the Web- Api*

---

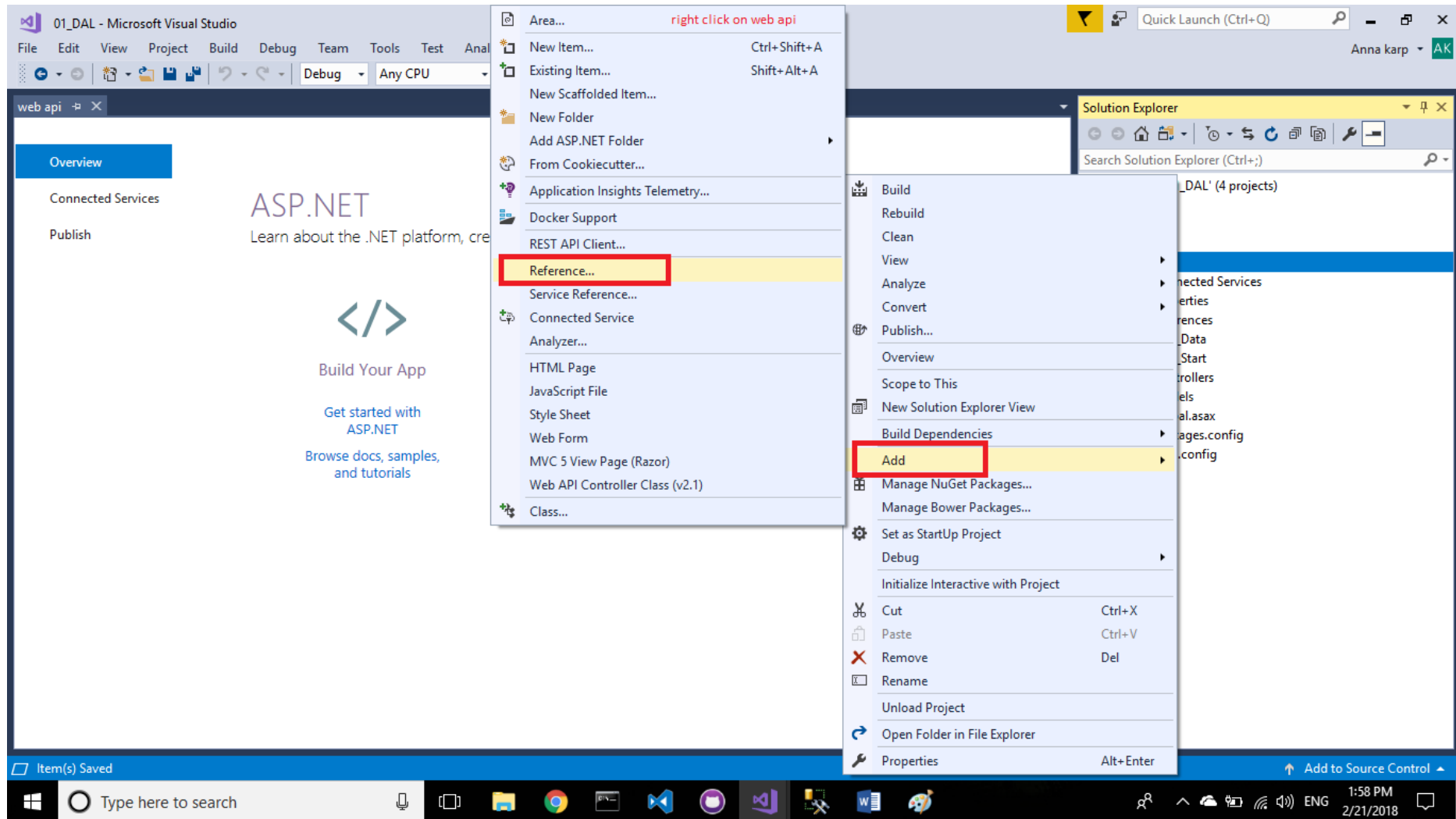
### Step 1– add to the current solution an ASP.NET project



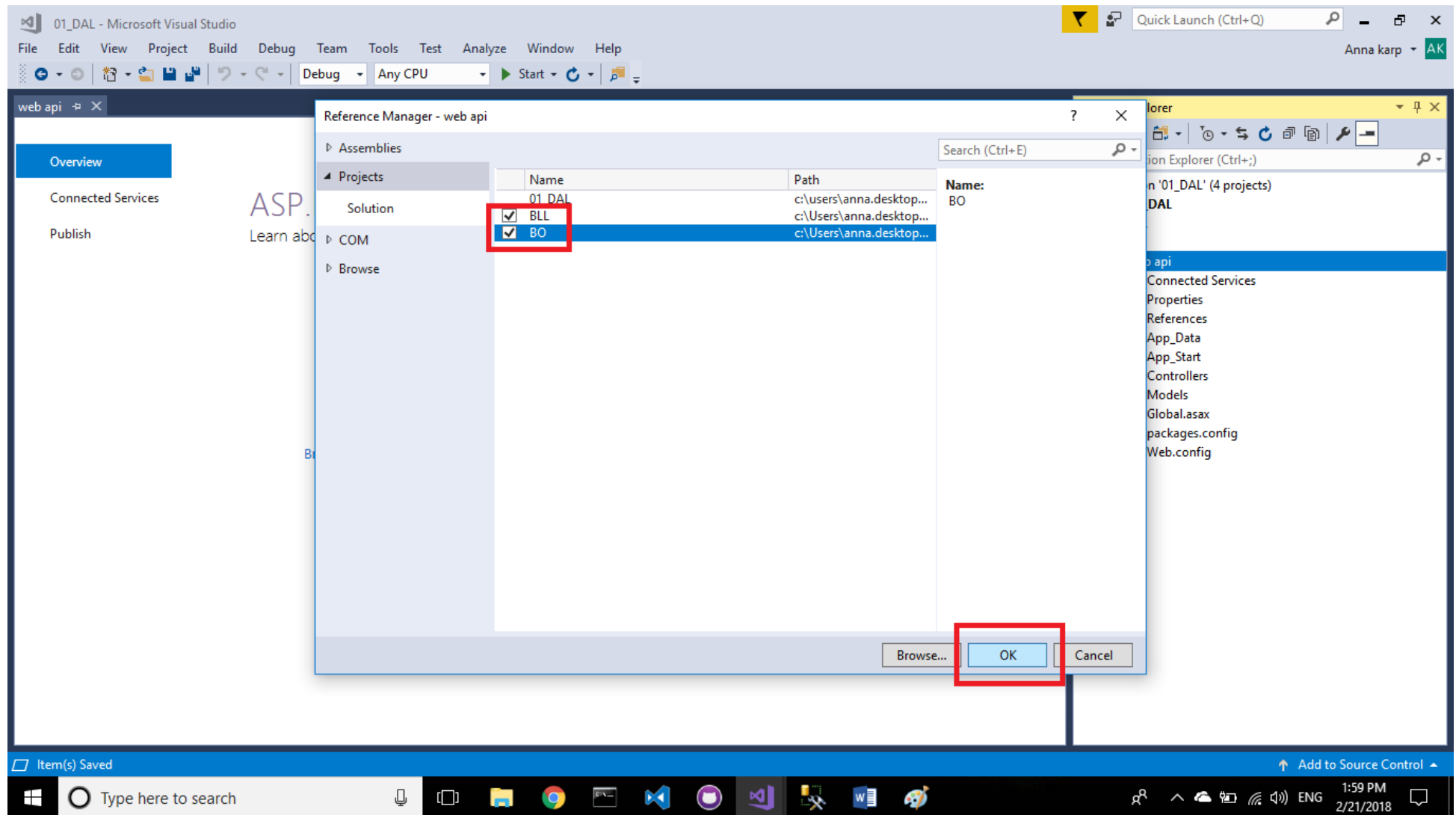
## Step 2– select an Empty web api project



### Step 3—add references



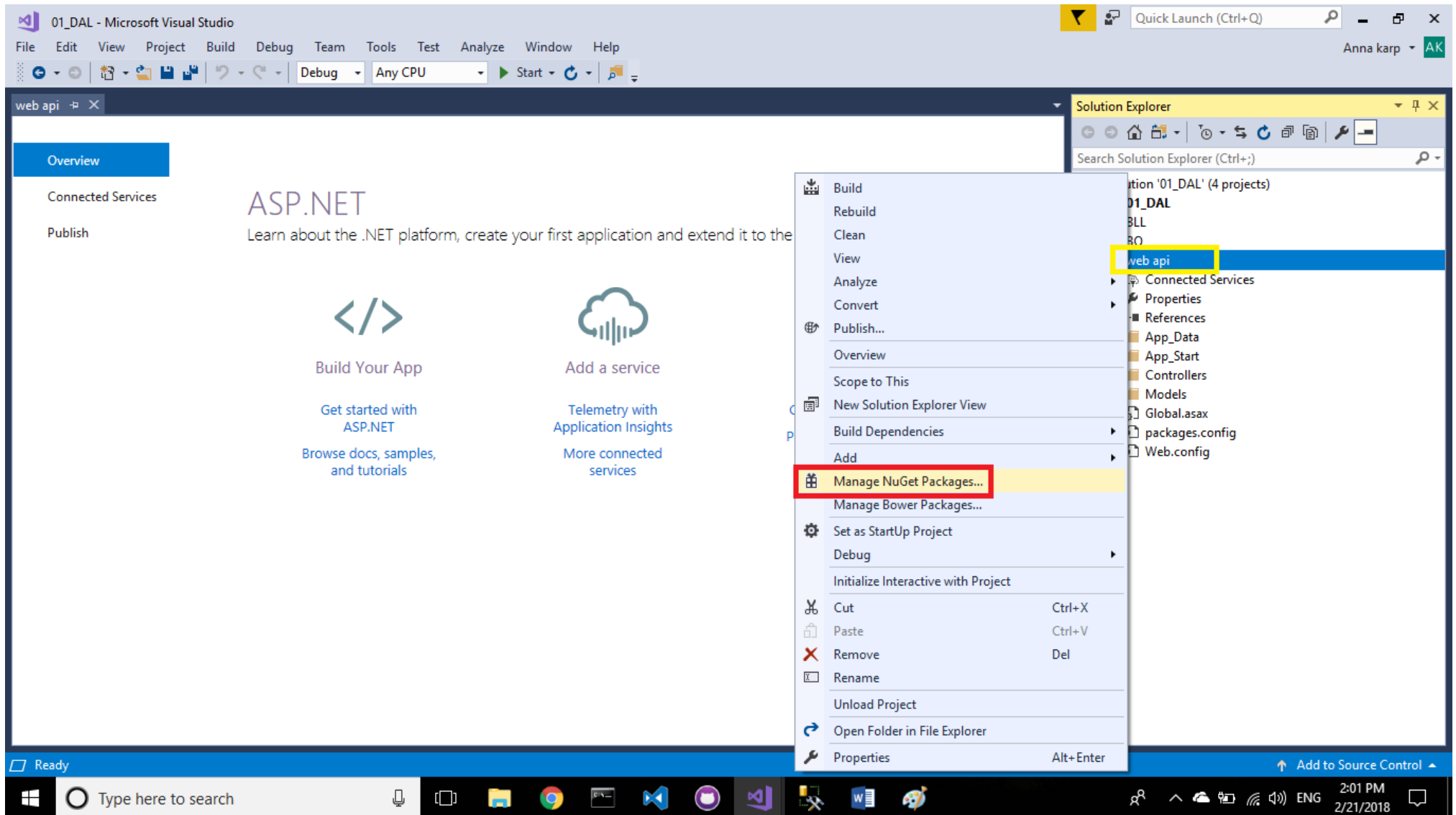
#### Step 4— add references to the BOL and BLL



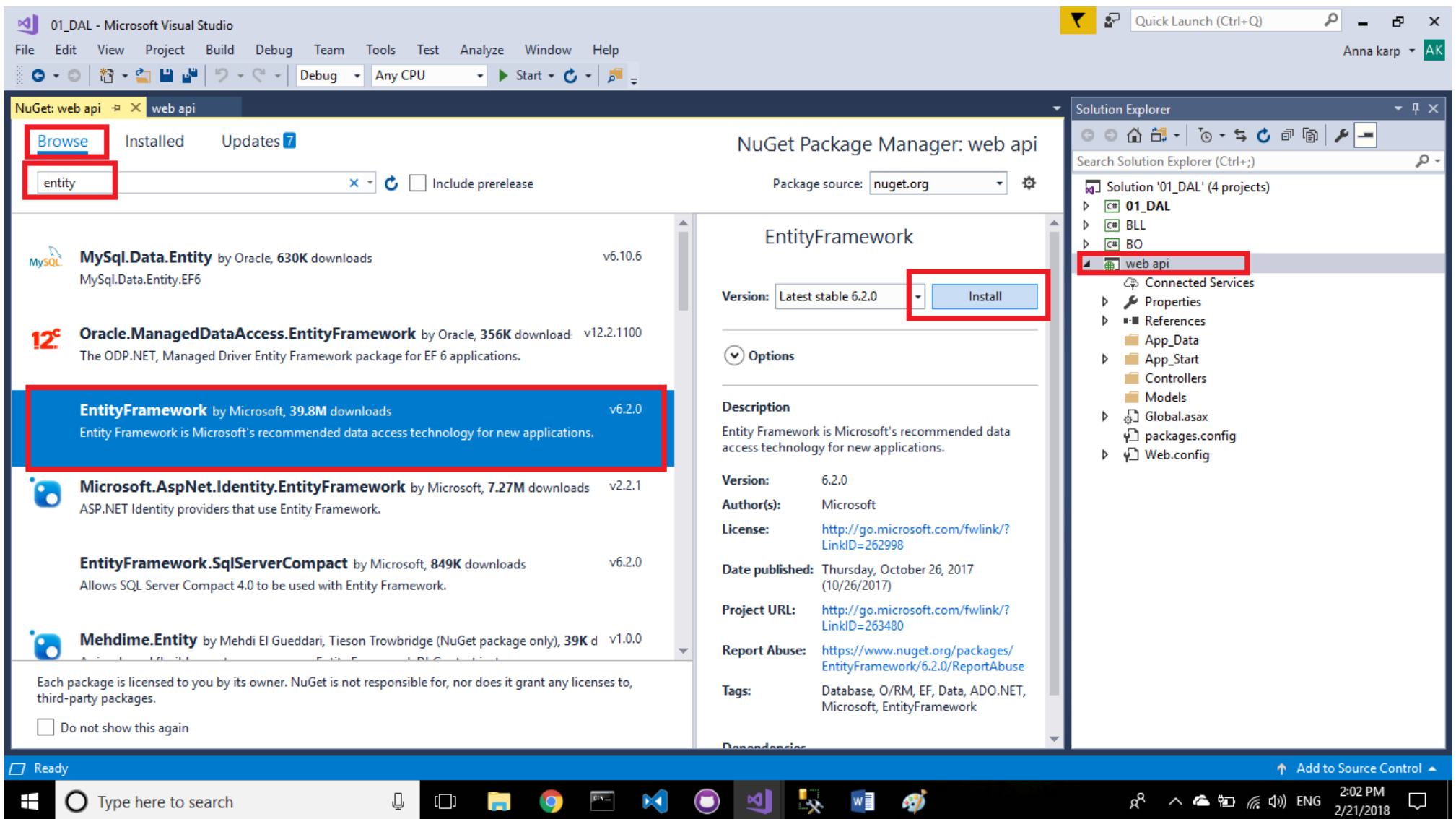
---

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

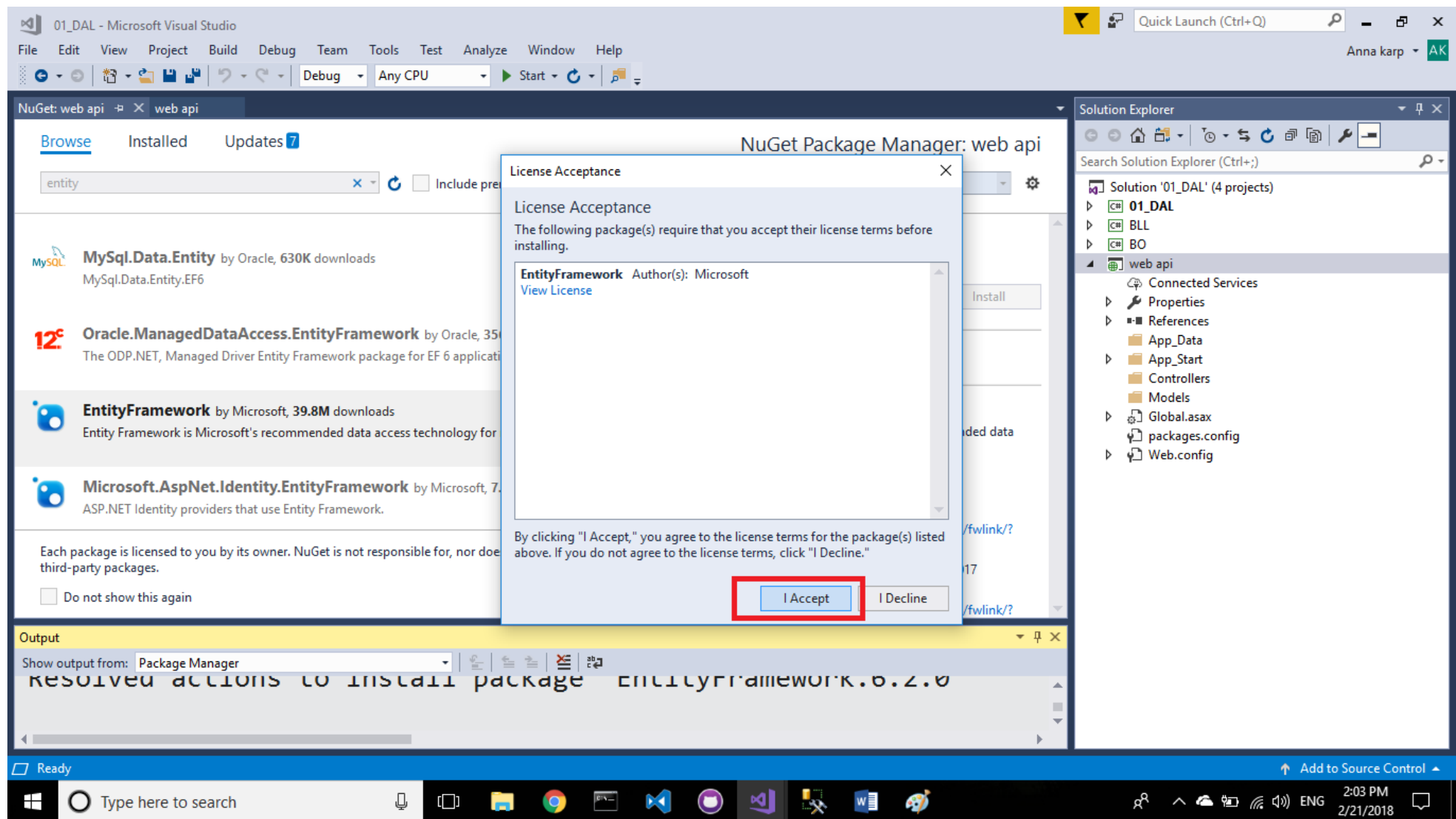
---



## 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

The image shows two instances of Microsoft Visual Studio. The top instance displays the `App.Config` file for the '01\_DAL' project. A red rectangle highlights the following XML snippet:

```
<connectionStrings>
  <add name="BookStoreEntities" connectionString="metadata=res://*/BookStore.csdl|res://*/BookStore.ssdl|res://*/BookStore.msl;provider=System.Data.SqlClient;providerAssembly=System.Data.SqlClient;providerType=System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
</connectionStrings>
```

The bottom instance displays the `Web.config` file for the 'web api' project. A green rectangle highlights the same XML snippet being added to the `<connectionStrings>` section. A large black arrow points from the red box in the top window to the green box in the bottom window, indicating the copy-paste action. A text box with the instruction "copy connection string from DAL to Web-api" is positioned near the arrow. The Solution Explorer on the right of the bottom window shows the project structure, with 'web api' and 'Web.config' highlighted.

01\_DAL (Running) - Microsoft Visual Studio

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

Process: [2824] iisexpress.exe Lifecycle Events Thread: Stack Frame:

App.Config BookController.cs NuGet: web api web api

1 <?xml version="1.0" encoding="utf-8"?>  
2 <configuration>  
3 <configSections>  
4 <!-- For more information on Entity Framework configuration, visit <http://go.microsoft.com/fwlink/?LinkID=237468> -->  
5 <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" />  
6 </configSections>  
7 <connectionStrings>  
8 <add name="BookStoreEntities" connectionString="metadata=res://\*/BookStore.csdl|res://\*/BookStore.ssdl|res://\*/BookStore.msl;provider=System.Data.SqlClient;providerAssembly=System.Data.SqlClient;providerType=System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />  
9 </connectionStrings>  
10 <entityFramework>  
11 <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />  
12 <providers>  
13 <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />  
14 </providers>  
15 </entityFramework>  
16 </configuration>

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution '01\_DAL' (4 projects)

- Script Documents
- 01\_DAL
- Properties
- References
- App.Config
- BookStore.edmx
- BookStore.Context.tt

01\_DAL - Microsoft Visual Studio

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

Web.config\* App.Config BookController.cs NuGet: web api web api

49 </system.codedom>  
50 <entityFramework>  
51 <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />  
52 <providers>  
53 <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />  
54 </providers>  
55 </entityFramework>  
56  
57 <connectionStrings>  
58 <add name="BookStoreEntities" connectionString="metadata=res://\*/BookStore.csdl|res://\*/BookStore.ssdl|res://\*/BookStore.msl;provider=System.Data.SqlClient;providerAssembly=System.Data.SqlClient;providerType=System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />  
59 </connectionStrings>  
60  
61 </configuration>  
62

Solution Explorer

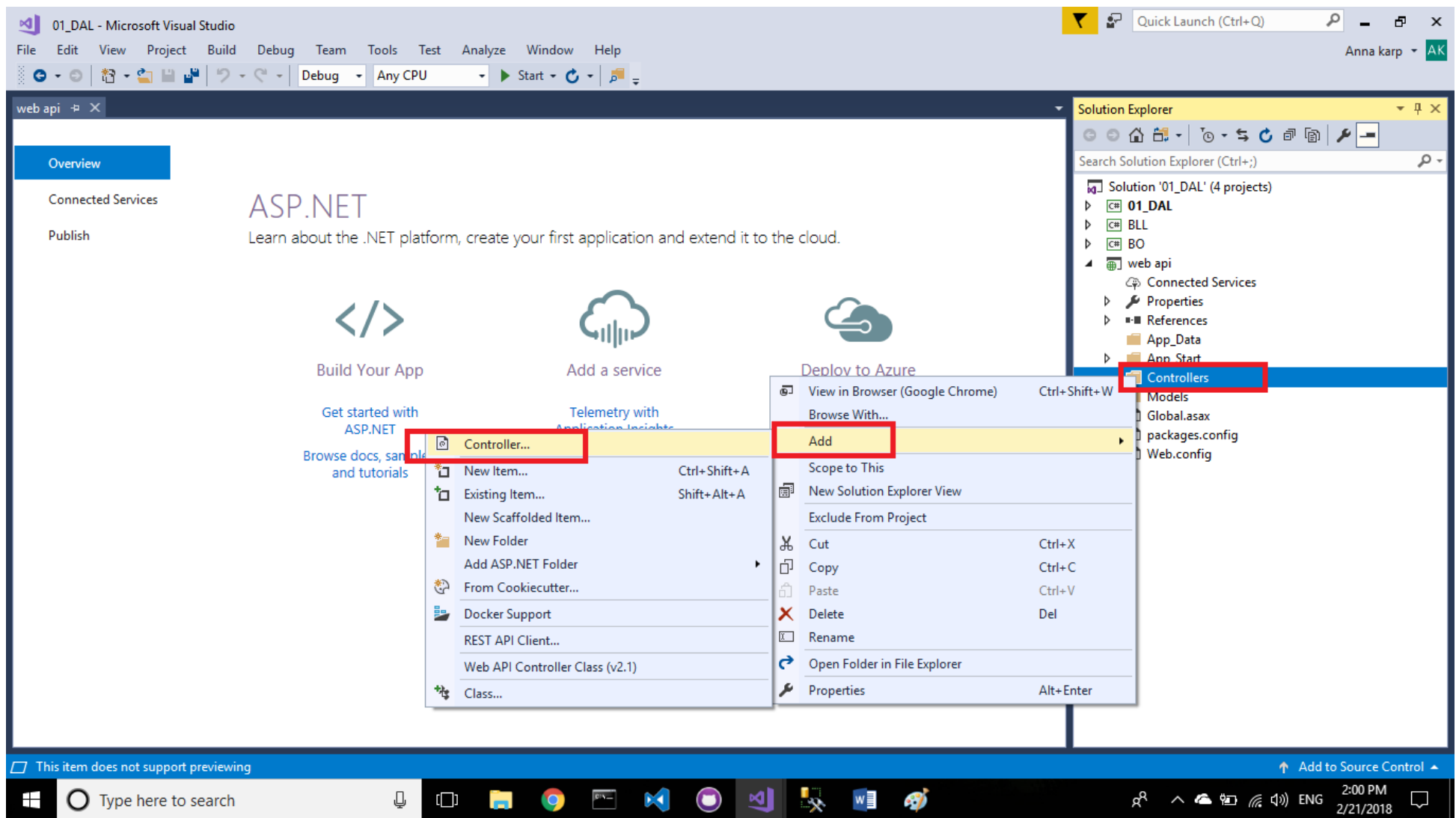
Search Solution Explorer (Ctrl+;)

Solution '01\_DAL' (4 projects)

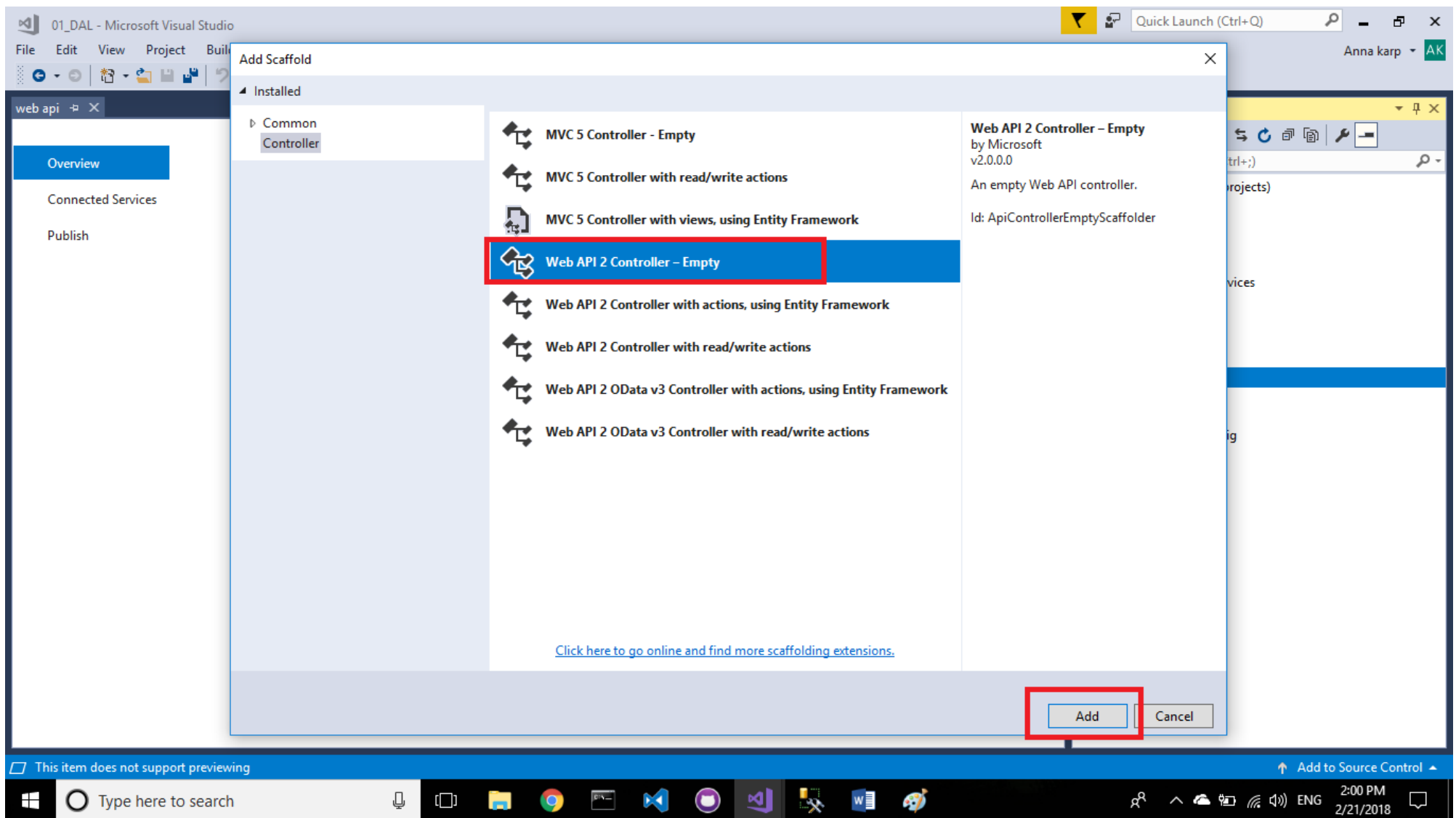
- 01\_DAL
- BLL
- BC
- web api
- Connected Services
- Properties
- References
- App\_Data
- App\_Start
- Controllers
- Models
- Global.asax
- packages.config
- Web.config

copy connection string from DAL to Web-api

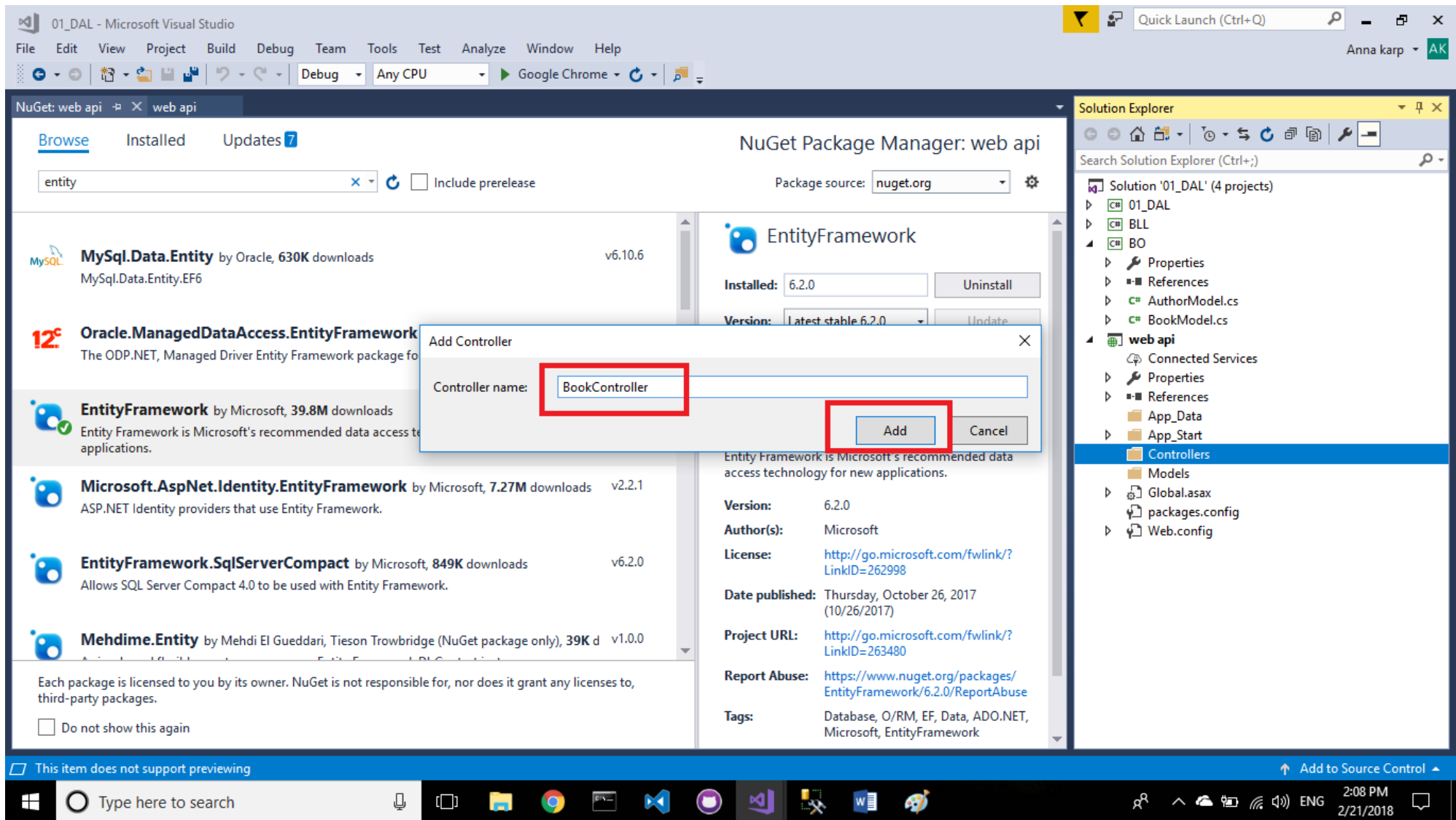
### Step 9—add a new controller



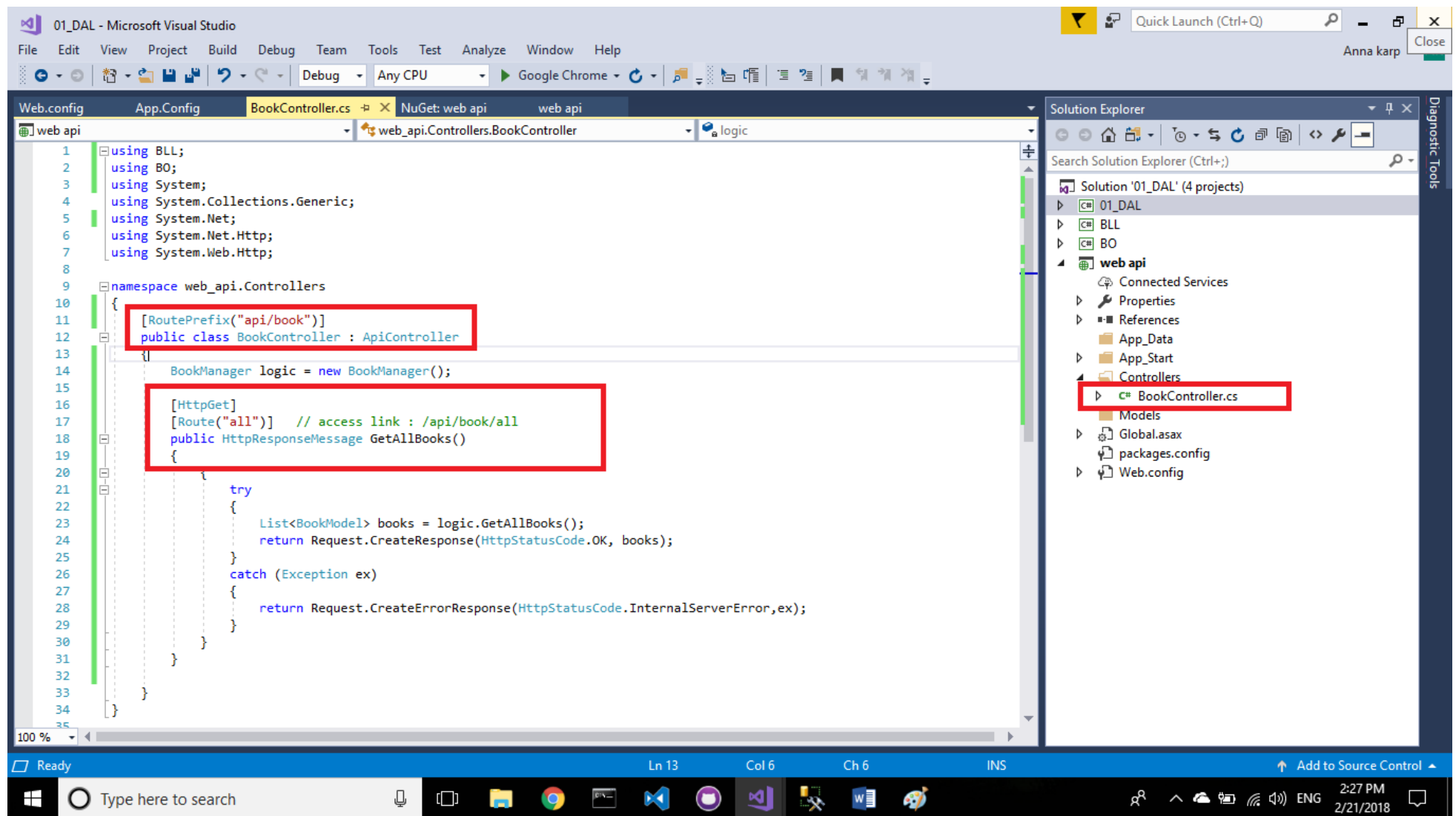
### Step 10– select a web api 2 controller



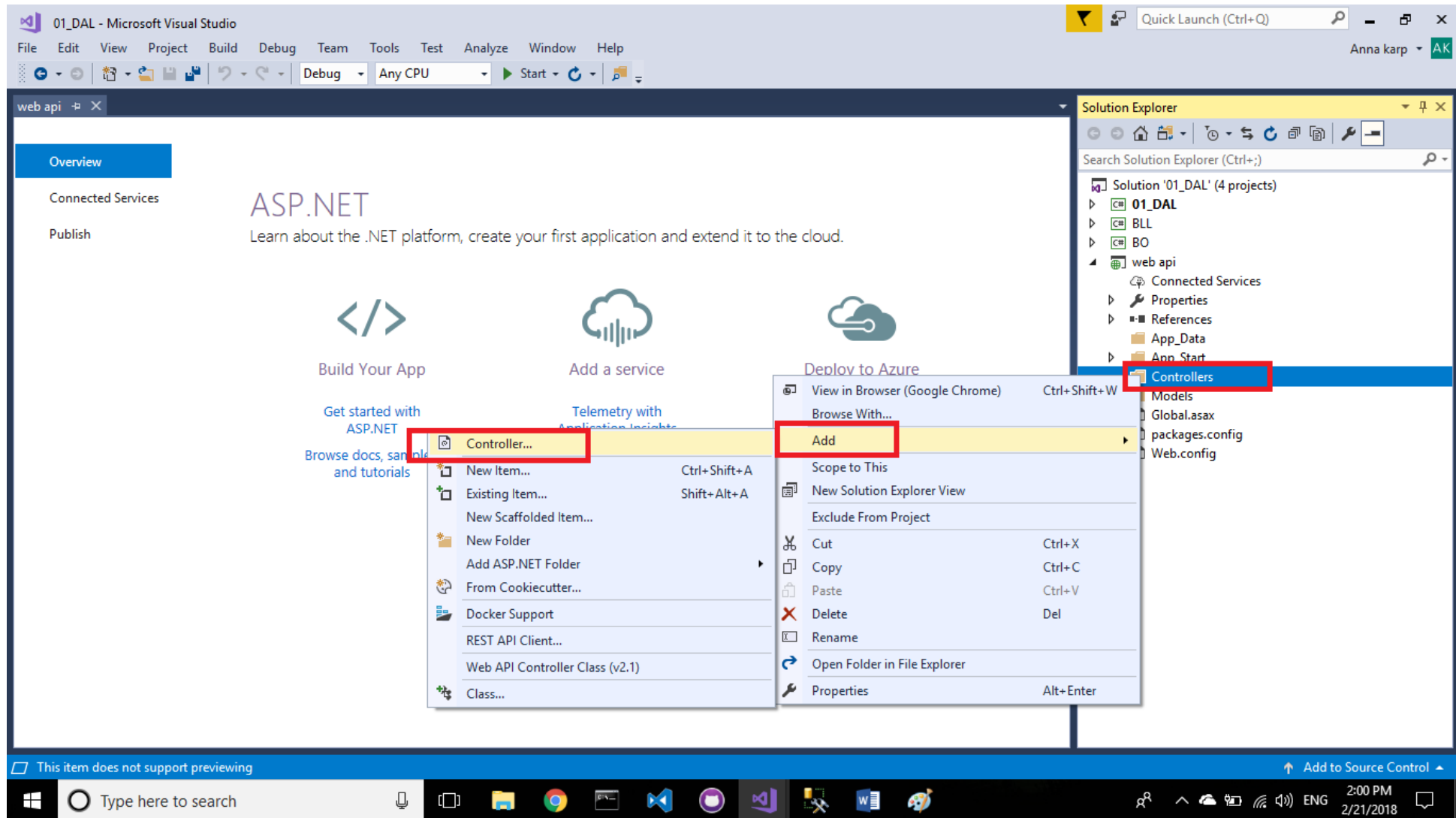
### Step 11– name the new controller



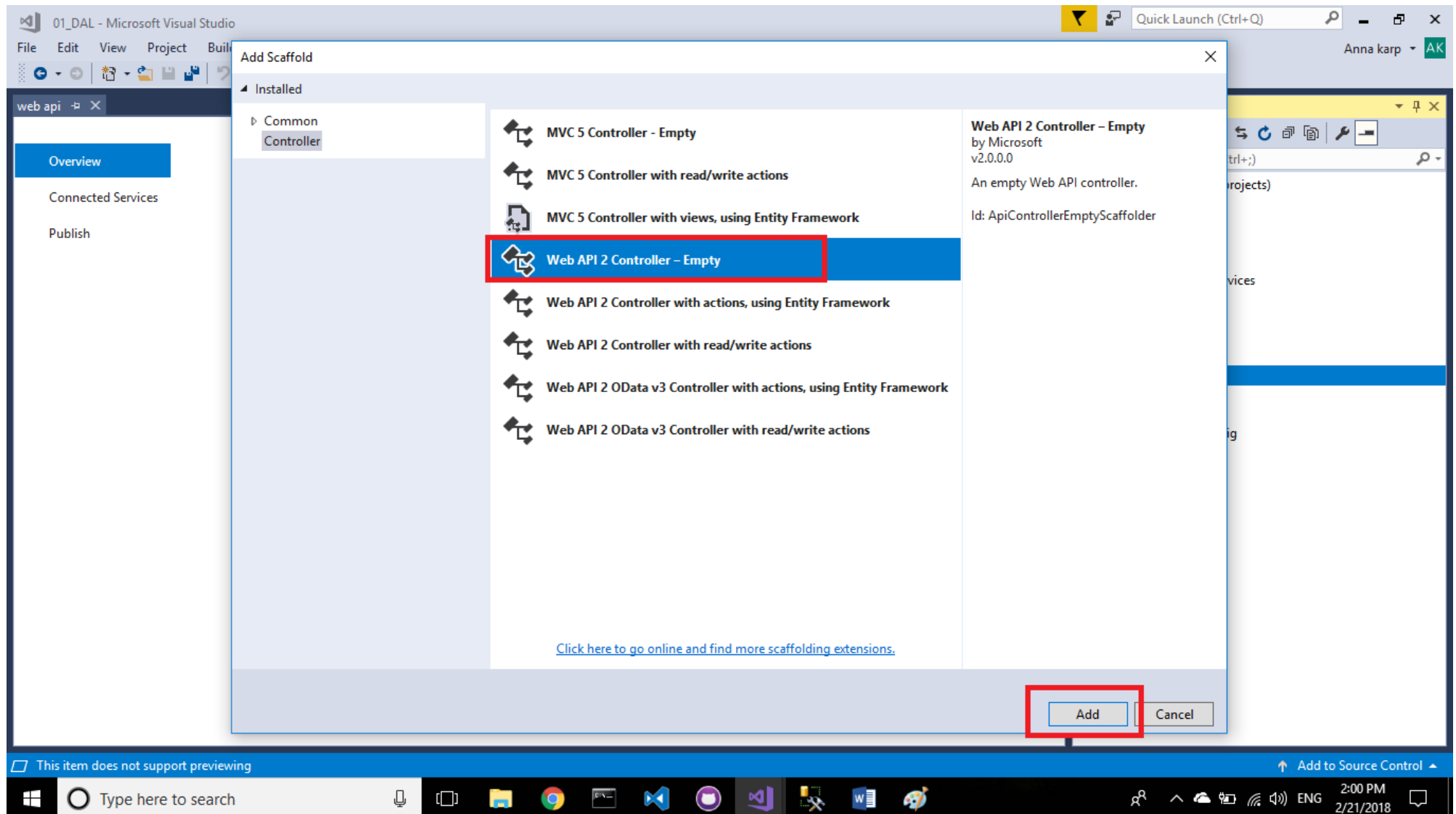
## Step 12– add the relevant content to the controller



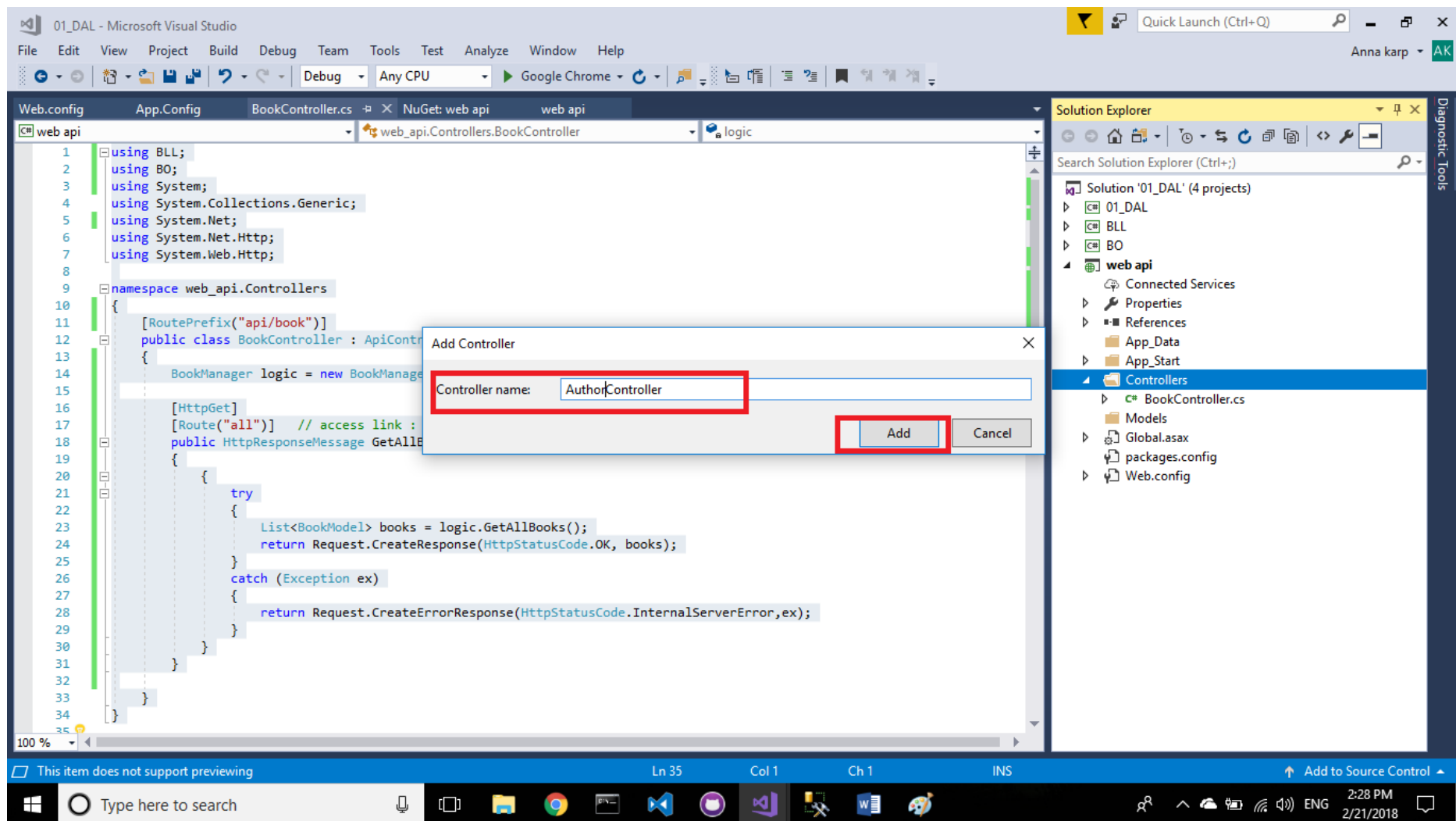
### Step 13– add another controller



*Step 14– select again the web api 2 controller*



### Step 15– name the controller





### Step 16– add the relevant content to the controller

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Code Editor:** Shows the `AuthorController.cs` file. The code includes the following content:

```
1 using BLL;
2 using BO;
3 using System;
4 using System.Collections.Generic;
5 using System.Net;
6 using System.Net.Http;
7 using System.Web.Http;
8
9 namespace web_api.Controllers
10 {
11     [RoutePrefix("api/author")]
12     public class AuthorController : ApiController
13     {
14         AuthorManager logic = new AuthorManager();
15
16         [HttpGet]
17         [Route("all")] // access link : /api/author/all
18         public HttpResponseMessage GetAllAuthors()
19         {
20             try
21             {
22                 List<AuthorModel> authors = logic.GetAllAuthors();
23                 return Request.CreateResponse(HttpStatusCode.OK, authors);
24             }
25             catch (Exception ex)
26             {
27                 return Request.CreateErrorResponse(HttpStatusCode.InternalServerError, ex);
28             }
29         }
30     }
31 }
32
33
34
35
```

Two red rectangles highlight the class definition and the `GetAllAuthors` method.
- Solution Explorer:** Located on the right, it shows the project structure for '01\_DAL'. The `web api` project is expanded, and `Controllers/AuthorController.cs` is highlighted with a red rectangle.
- Taskbar:** At the bottom, it shows the Windows taskbar with the search bar and various application icons.

---

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

---



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<ArrayOfBookModel xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/BO">
  <BookModel>
    <Author>
      <AuthorAge>20</AuthorAge>
      <AuthorImage>bob.png</AuthorImage>
      <AuthorName>Bob</AuthorName>
    </Author>
    <BookName>Tester</BookName>
    <BookPrice>40</BookPrice>
  </BookModel>
</ArrayOfBookModel>
```



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



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<ArrayOfAuthorModel xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/B0">
  <AuthorModel>
    <AuthorAge>20</AuthorAge>
    <AuthorImage>bob.png</AuthorImage>
    <AuthorName>Bob</AuthorName>
  </AuthorModel>
</ArrayOfAuthorModel>
```



---

*GOOD LUCK!!!*

---