Book App – full steps

**אפיון הפרוייקט**

סעיף 1:

יש ליצור טבלאות מתאימות בdb עבור שמירת נתוני הauthors 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 הAuthor 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 לDAL (מסעיף 2) ולBO (מסעיף 3 ),

בתוך הBLL 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 consol-app המכיל קישור לBLL (מסעיף 4) ולBO (מסעיף 3),

בתוכו יש ליצור:

**the following actions:**

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

will call the AuthorManager in the BLL

The BLL will create an object of the EF from the DAL, and pass all the authors back

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

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 back

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

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.

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

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.

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

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:**

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

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

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

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

* Function -  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

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

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

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

Part 1- create the DB

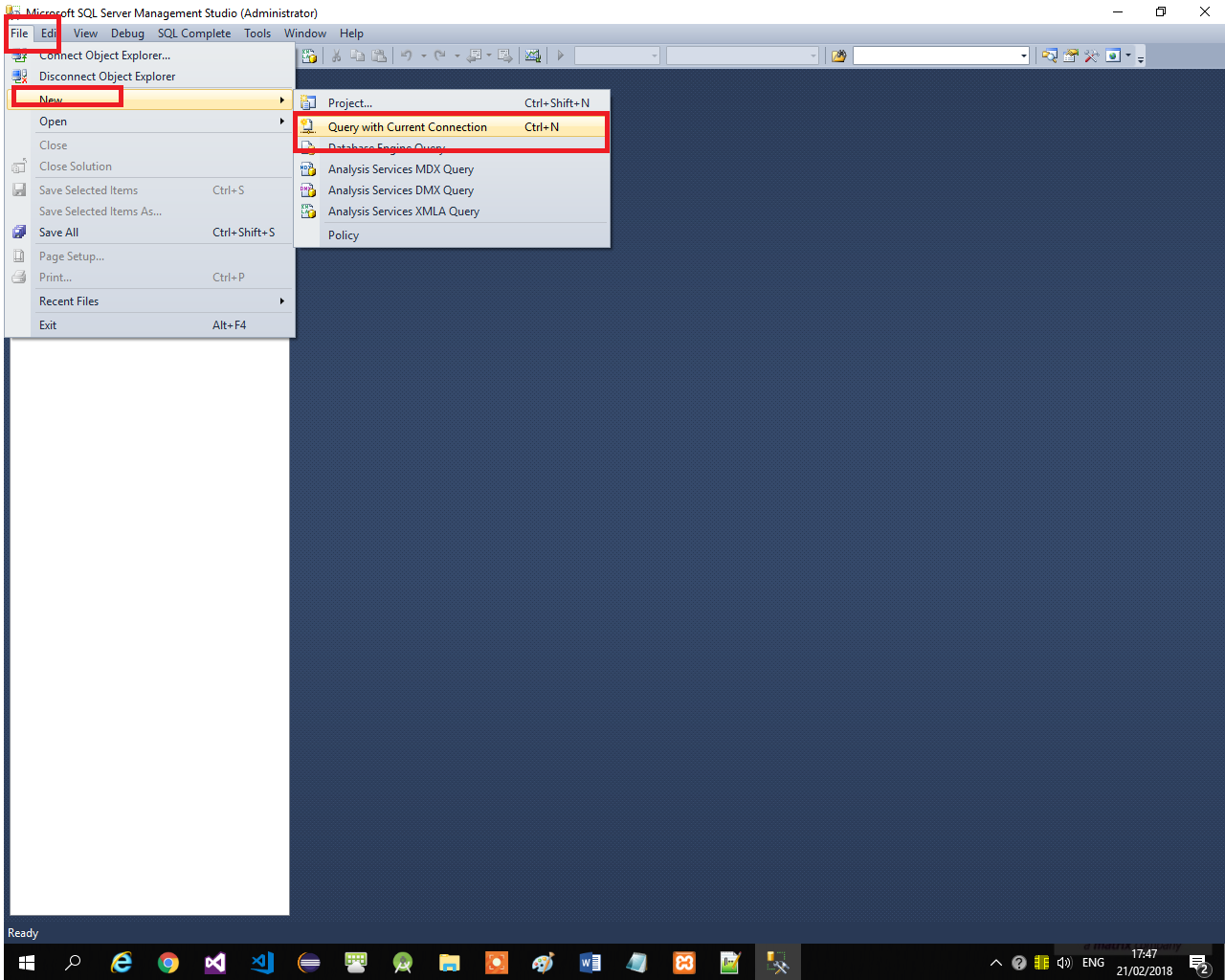
**Step 1 – create the script to add a new db to mssql**

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,

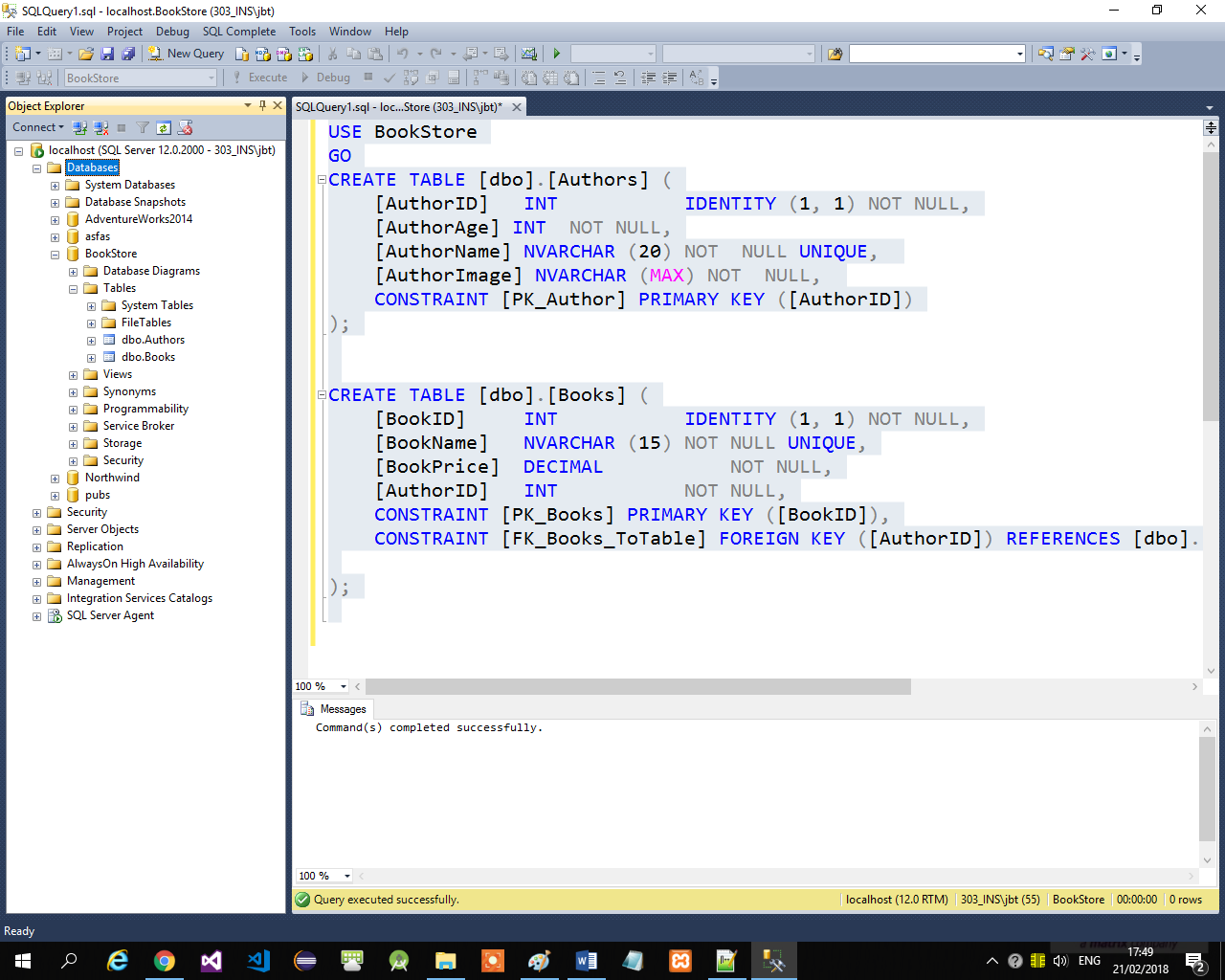
[AuthorID] INT NOT NULL,

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**



Add the folloeing function and SP

USE BookStore

GO

-- Create stored procedure to insert new record to the books table

create procedure InsertBook(@bookName nvarchar(15), @bookPrice decimal(18,0), @authorID int)

as

insert into dbo.Books(BookName, BookPrice, AuthorID)

values(@bookName, @bookPrice,@authorID)

go

-- Create function:

go

create function GetAuthorName(@firstName nvarchar(9),

@lastName nvarchar(10)) returns nvarchar(20)

as

begin

declare @fullName nvarchar(20)

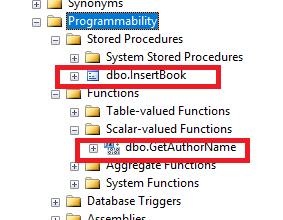
set @fullName = @firstName + ' ' + @lastName

return @fullName

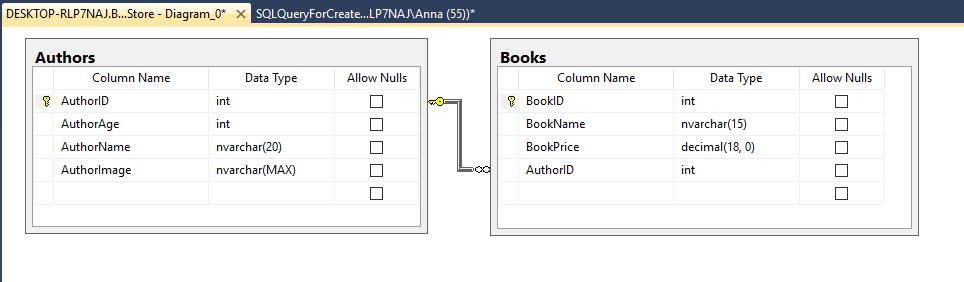
end

go

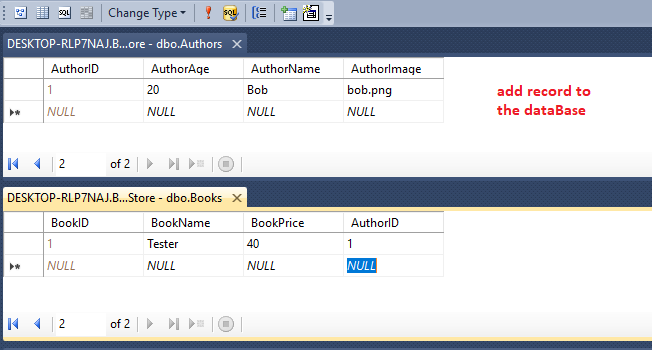
Run execute, and check that the command has completed successfully:



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

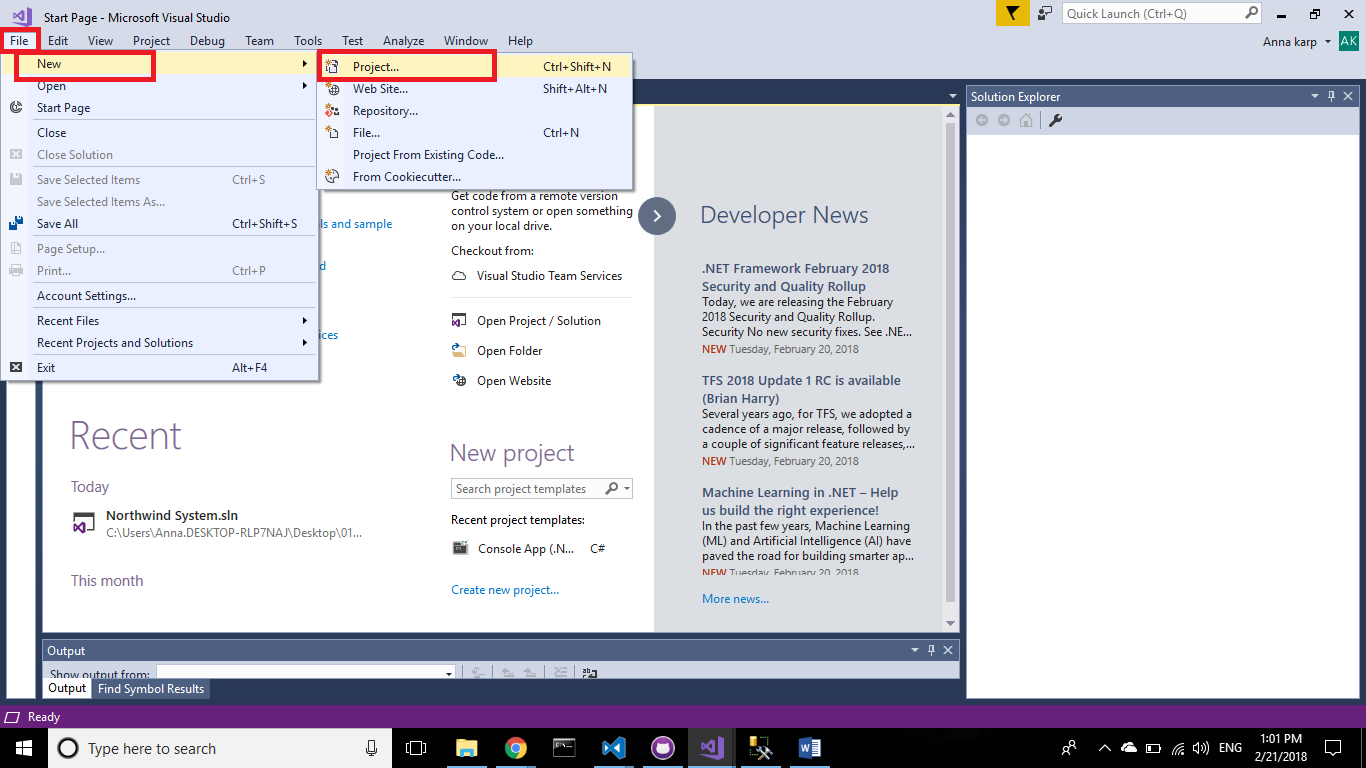


**Step 6– add records to the DB**

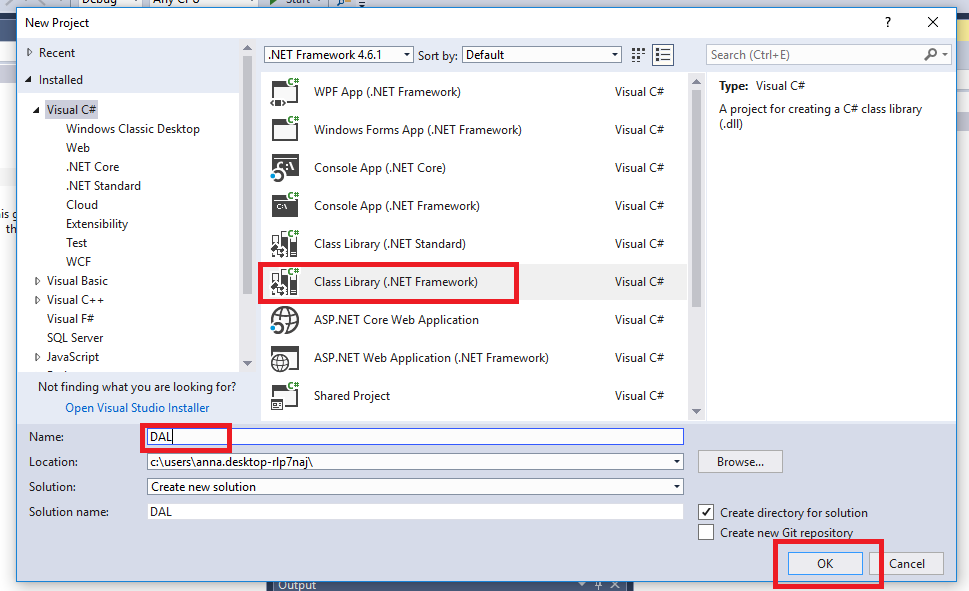


Part 2- the DAL

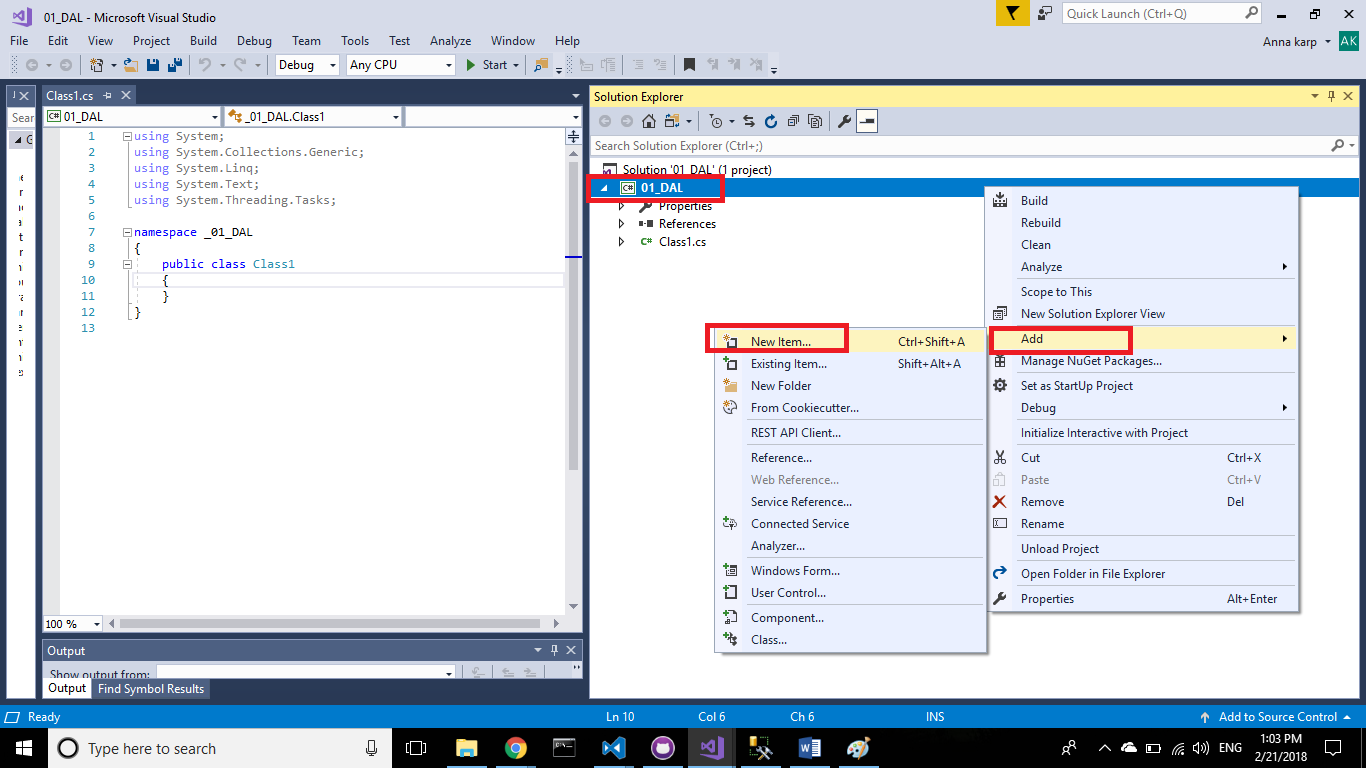
**Step 1 – open a new project in visual studio**



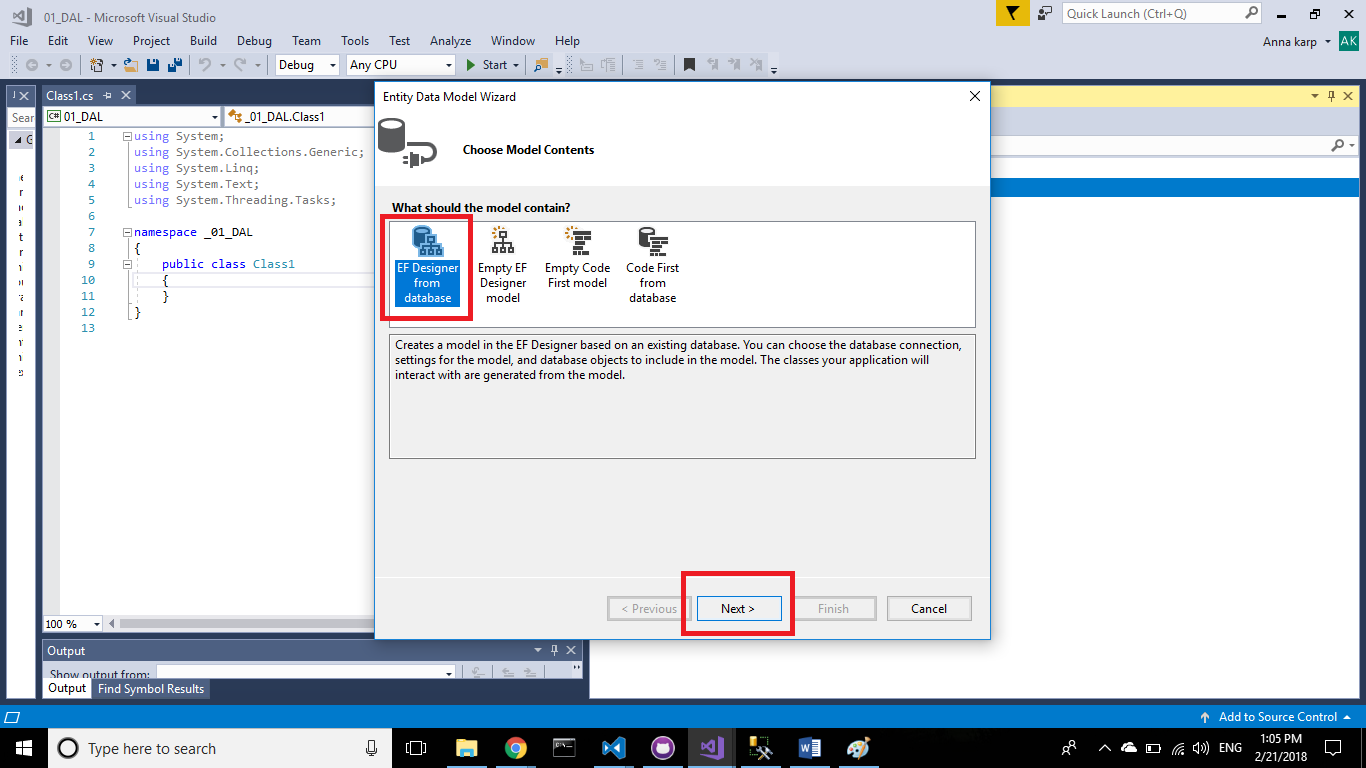
**Step 2– open a new Class library project**

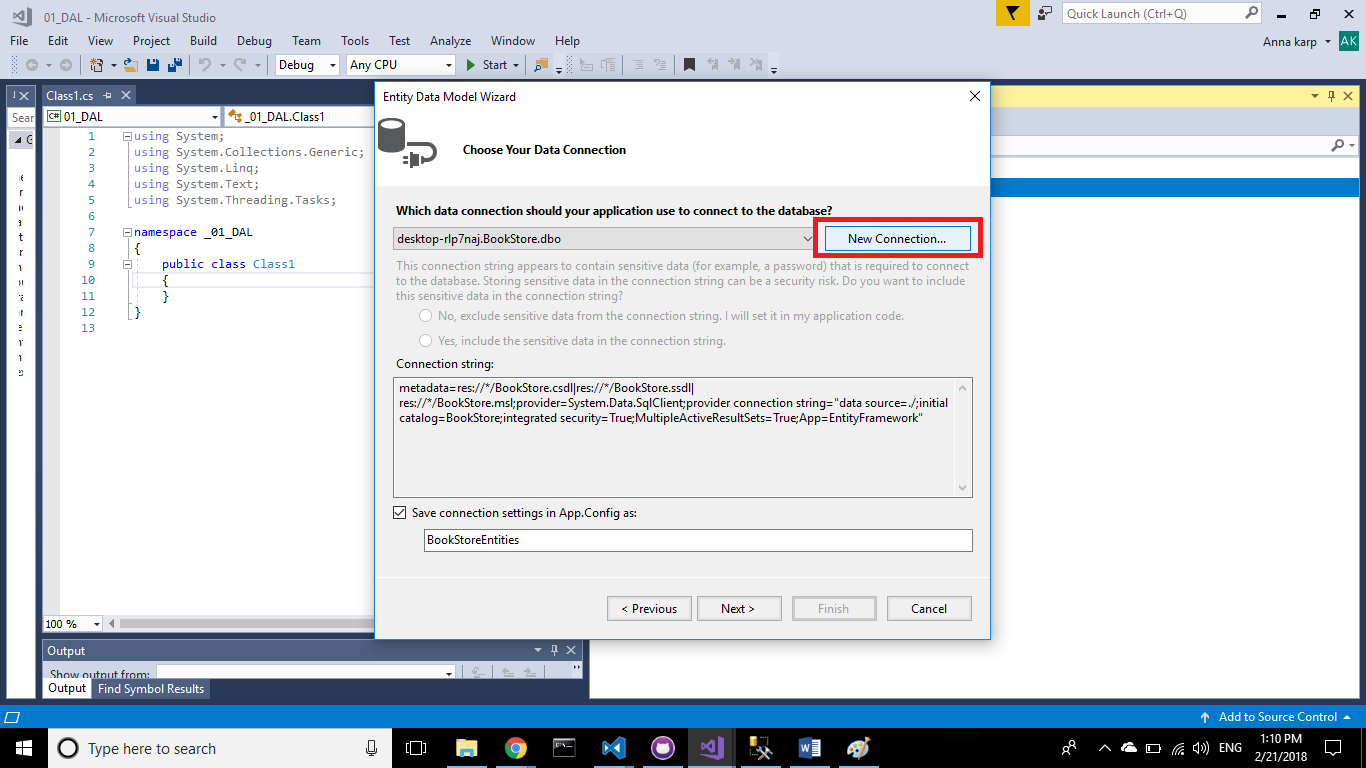


**Step 3 – start adding the EF**

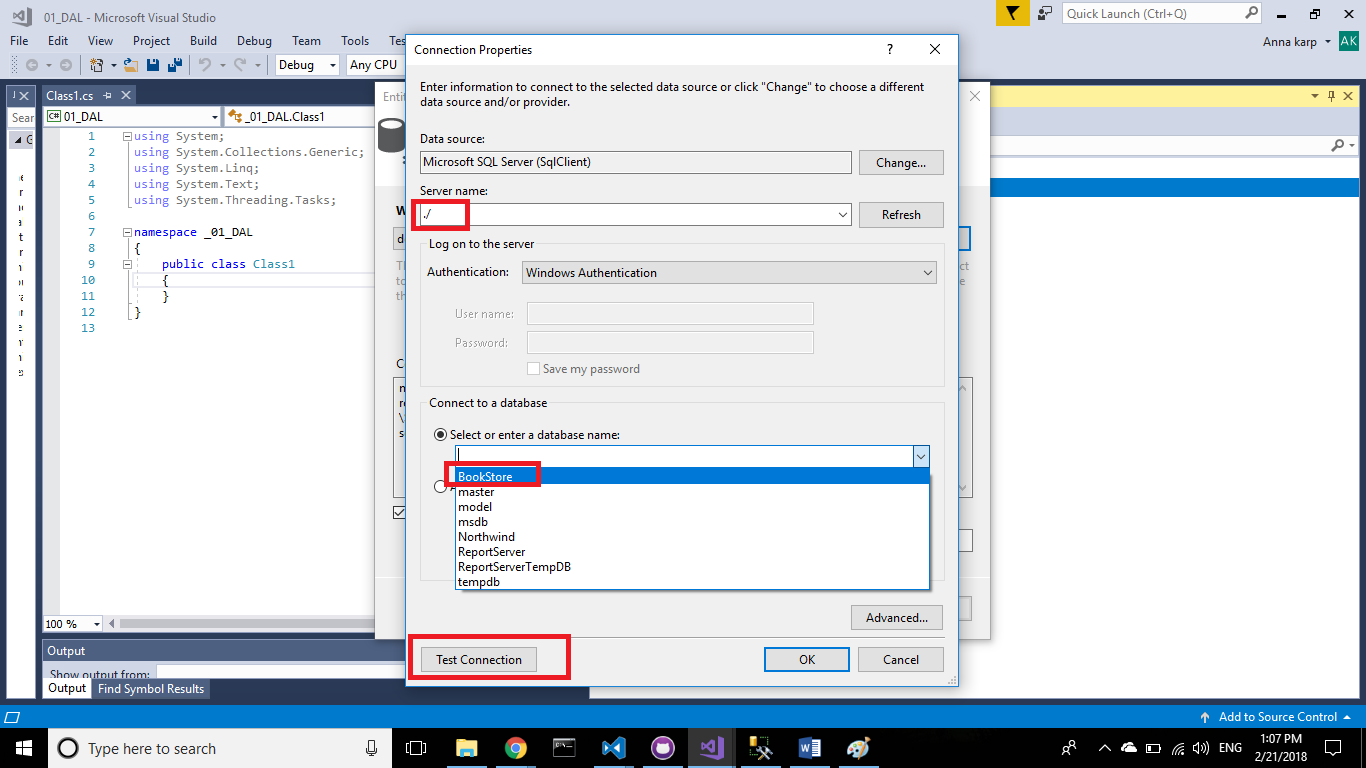


**Step 4– select ef from db**

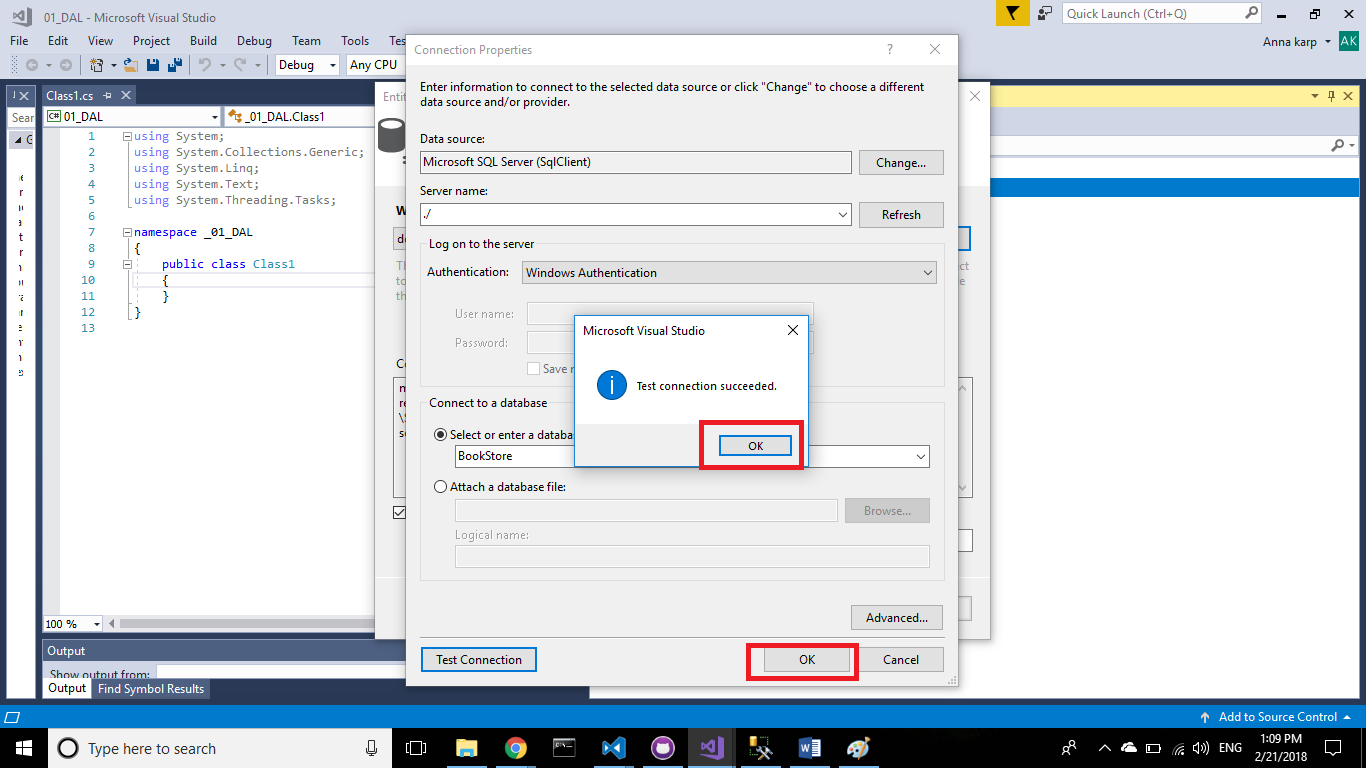


**Step 5- add a new sql connection**

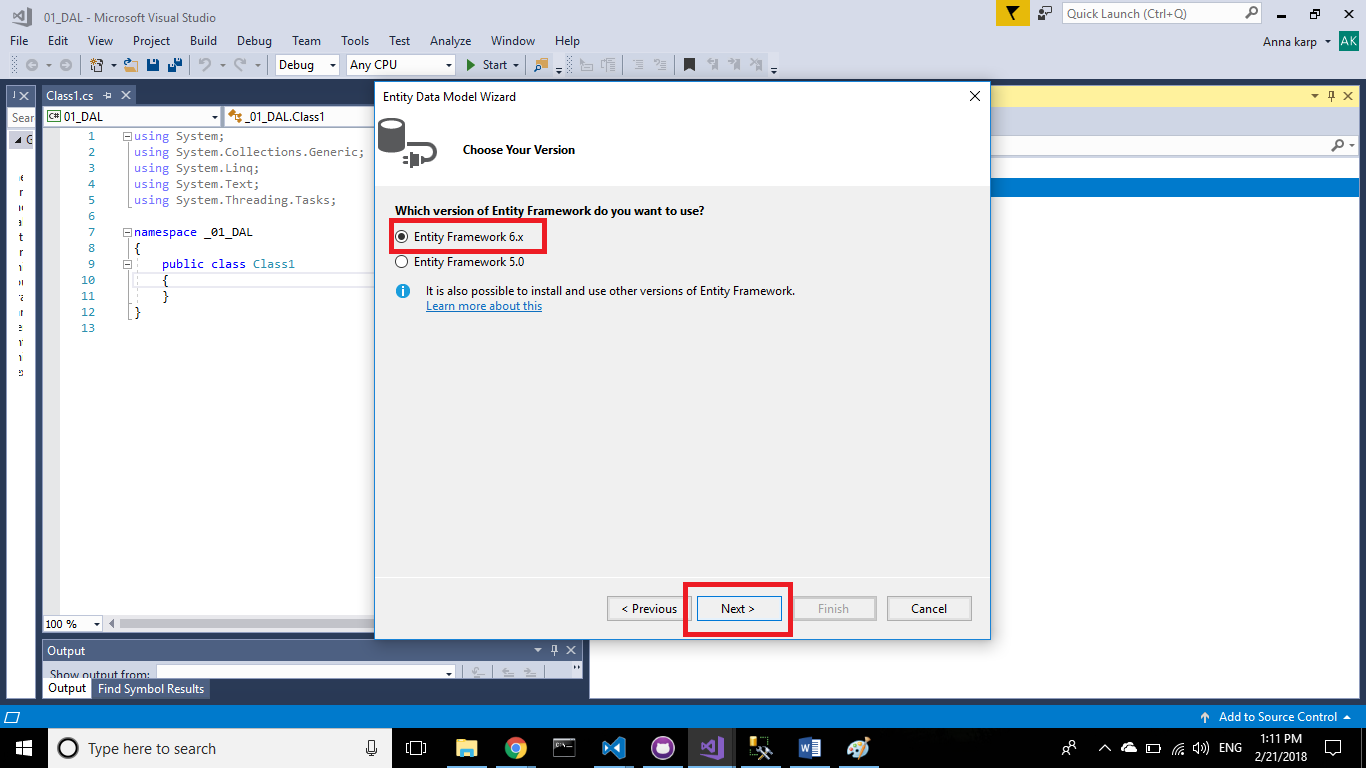
**Step 6– choose your server and then select the “BookStore” DB**



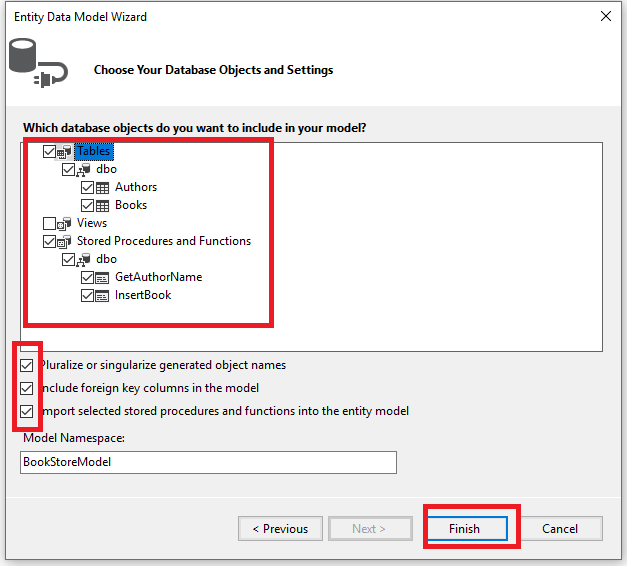
**Step 7– test the connection**



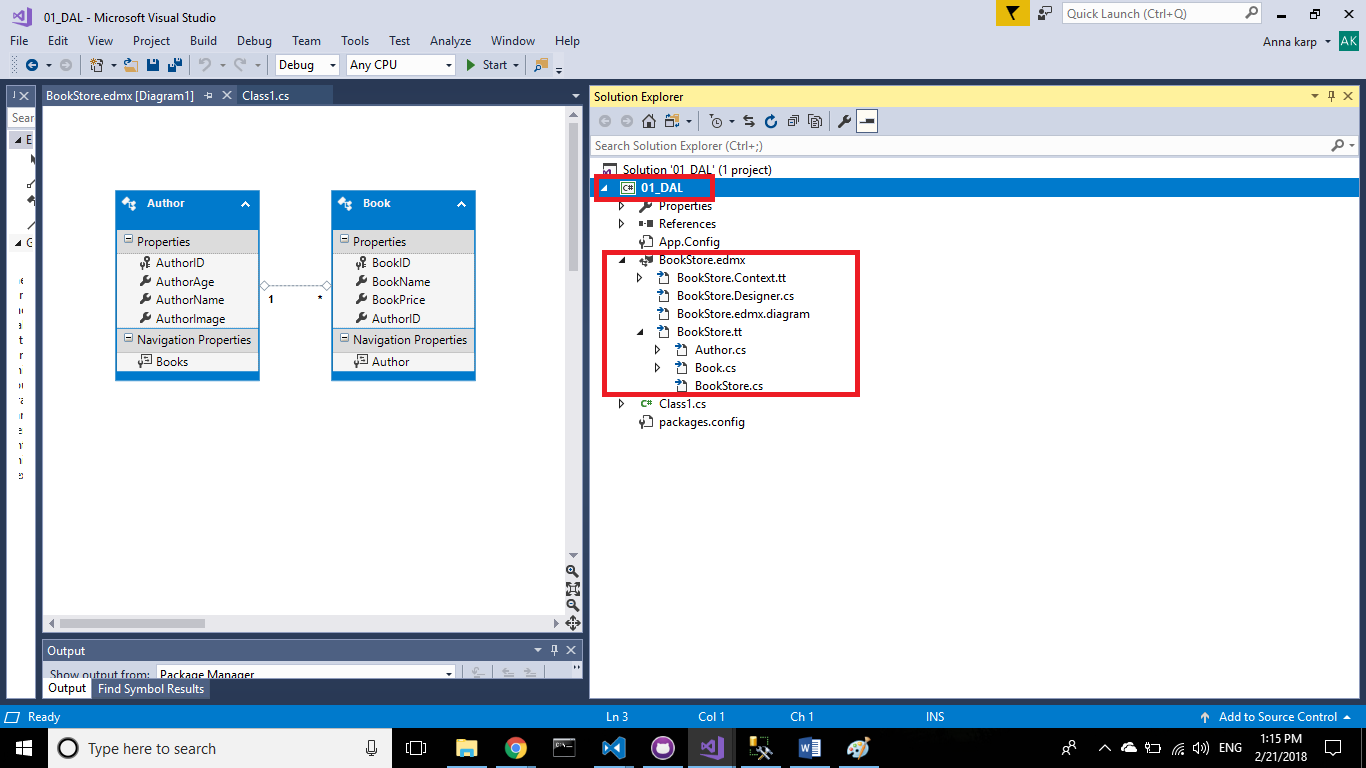
**Step 8– select the ef version**



**Step 9– select the relevant tables**

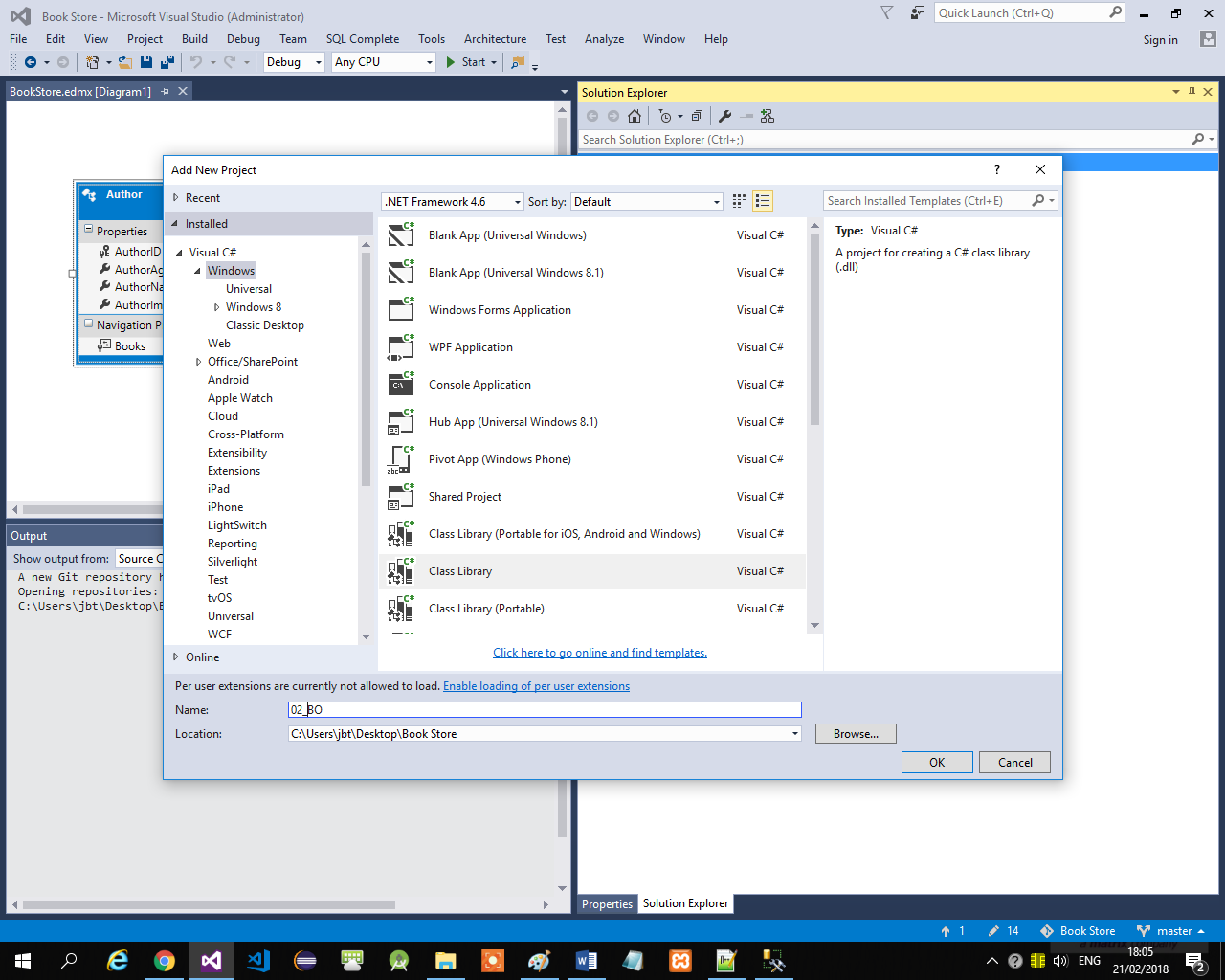


**Step 10– you added the ef successfully**

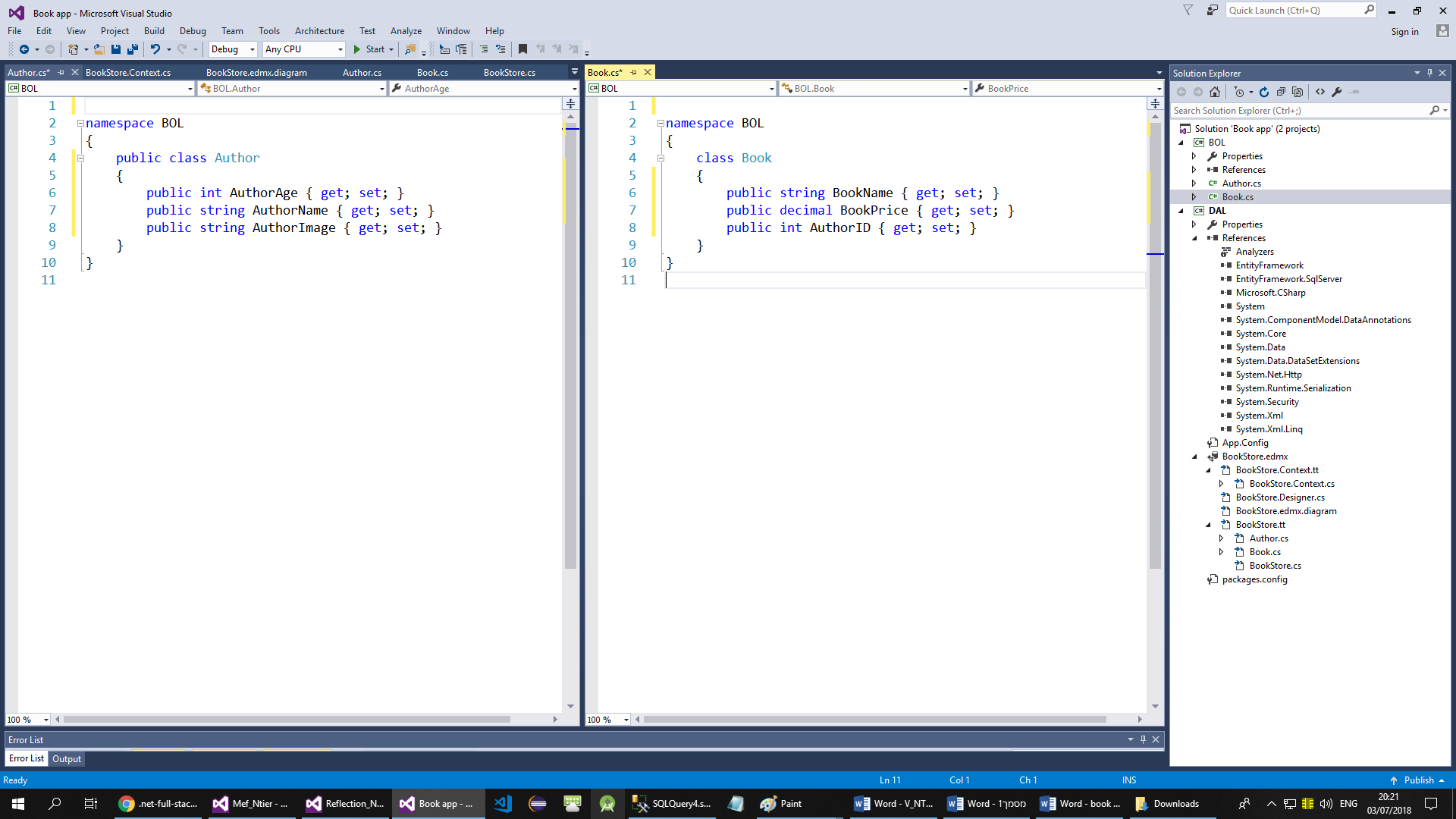


Part 3- create the BOL

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

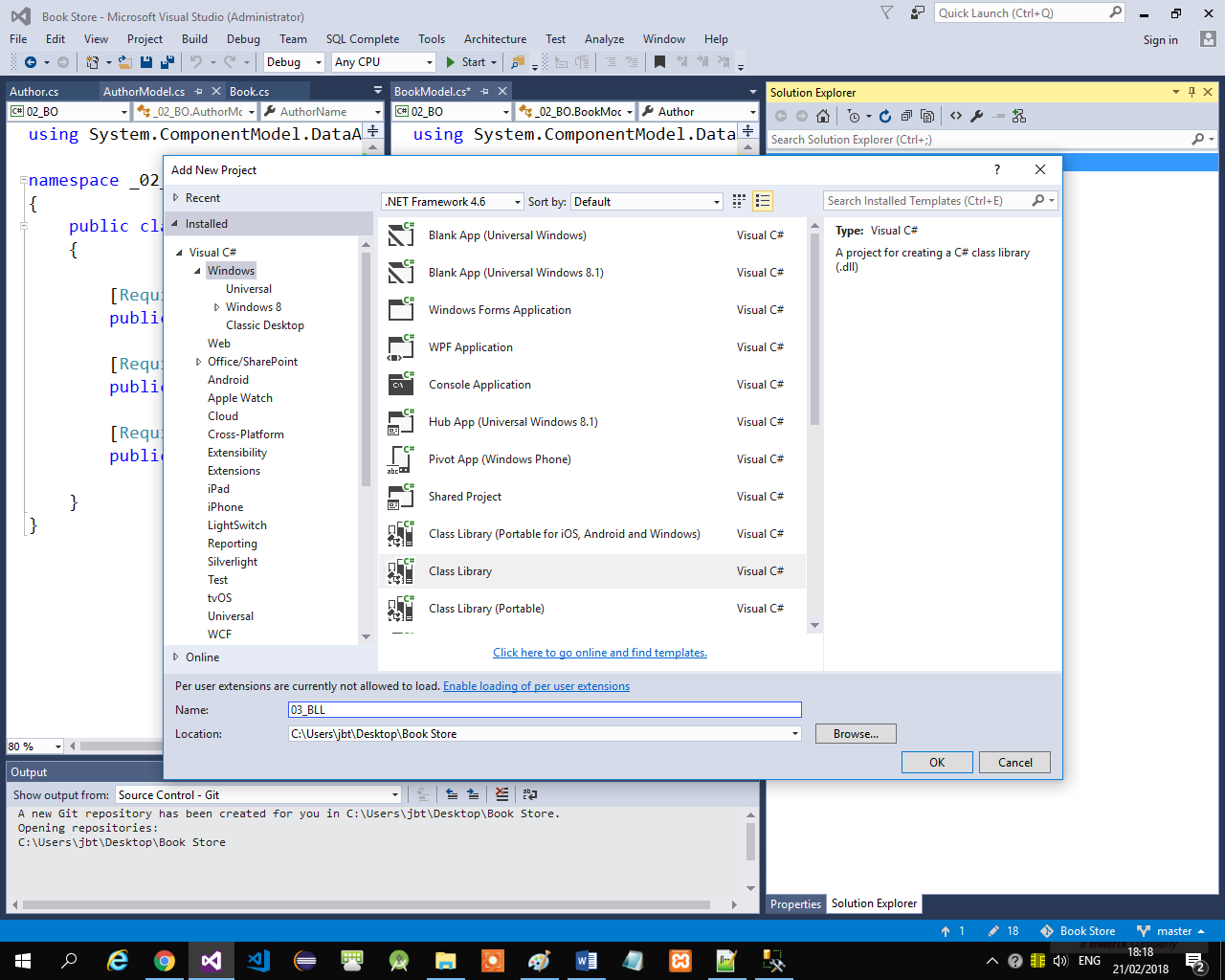


**Step 2– create 2 classes with the relevant names and properties (according to the DB tables)**

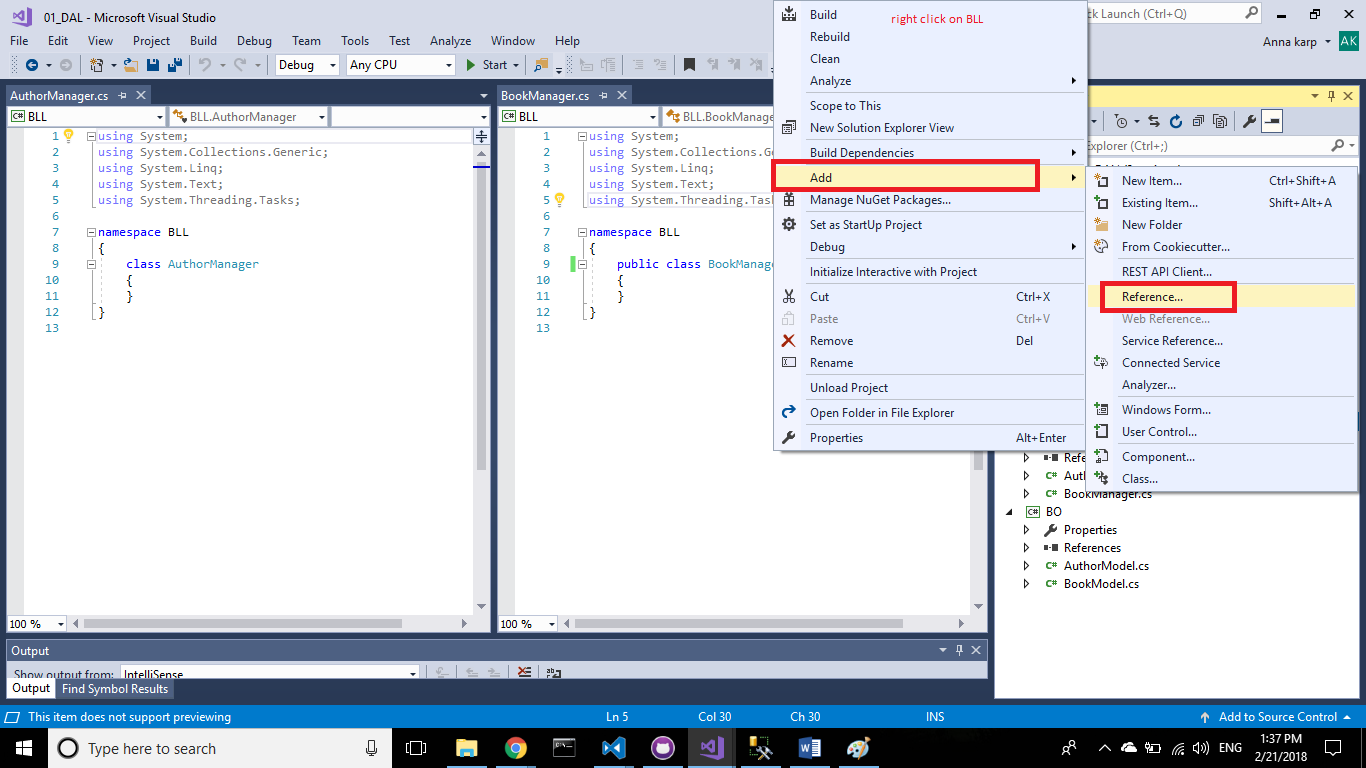


Part 4- create the BLL

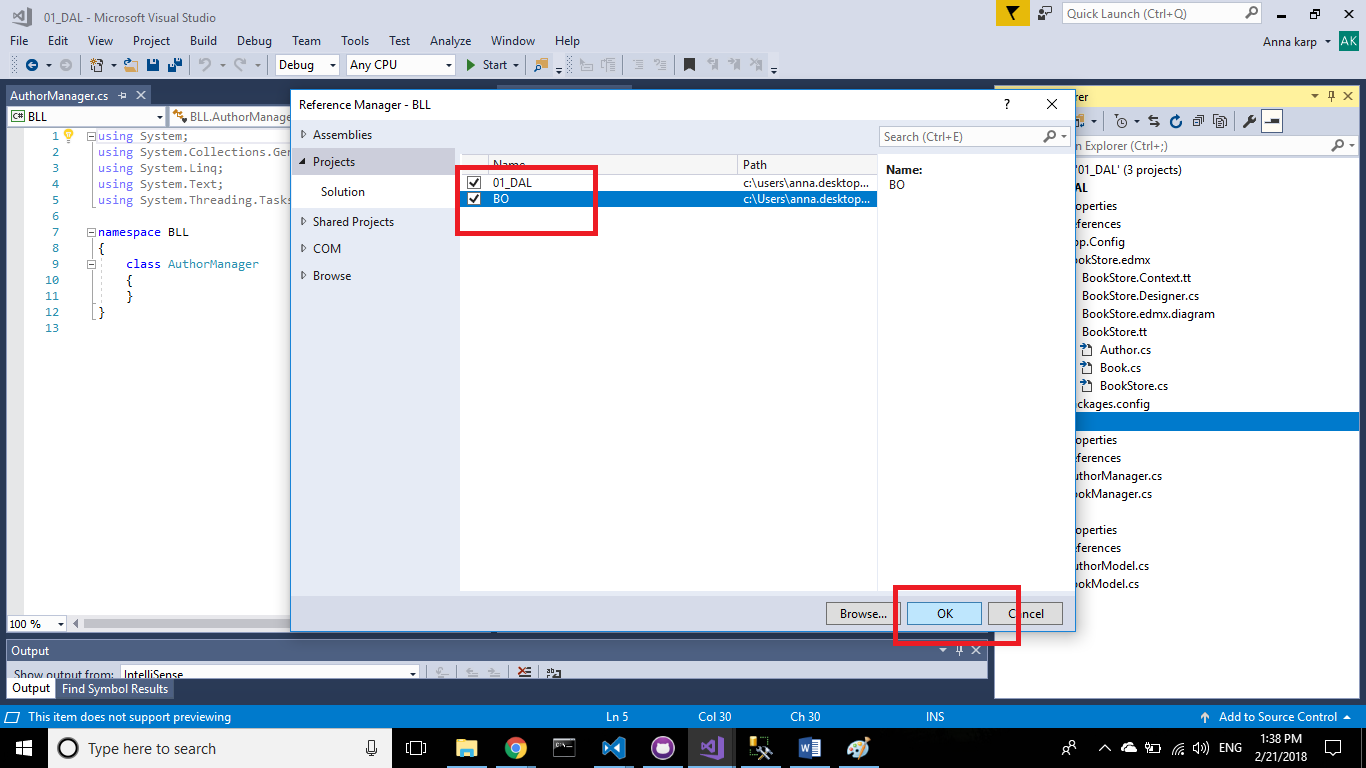
**Step 1– add to the current solution a new Class library project**



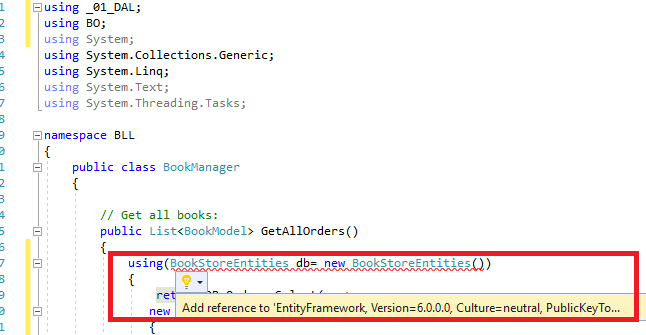
**Step 2– add to this project references**



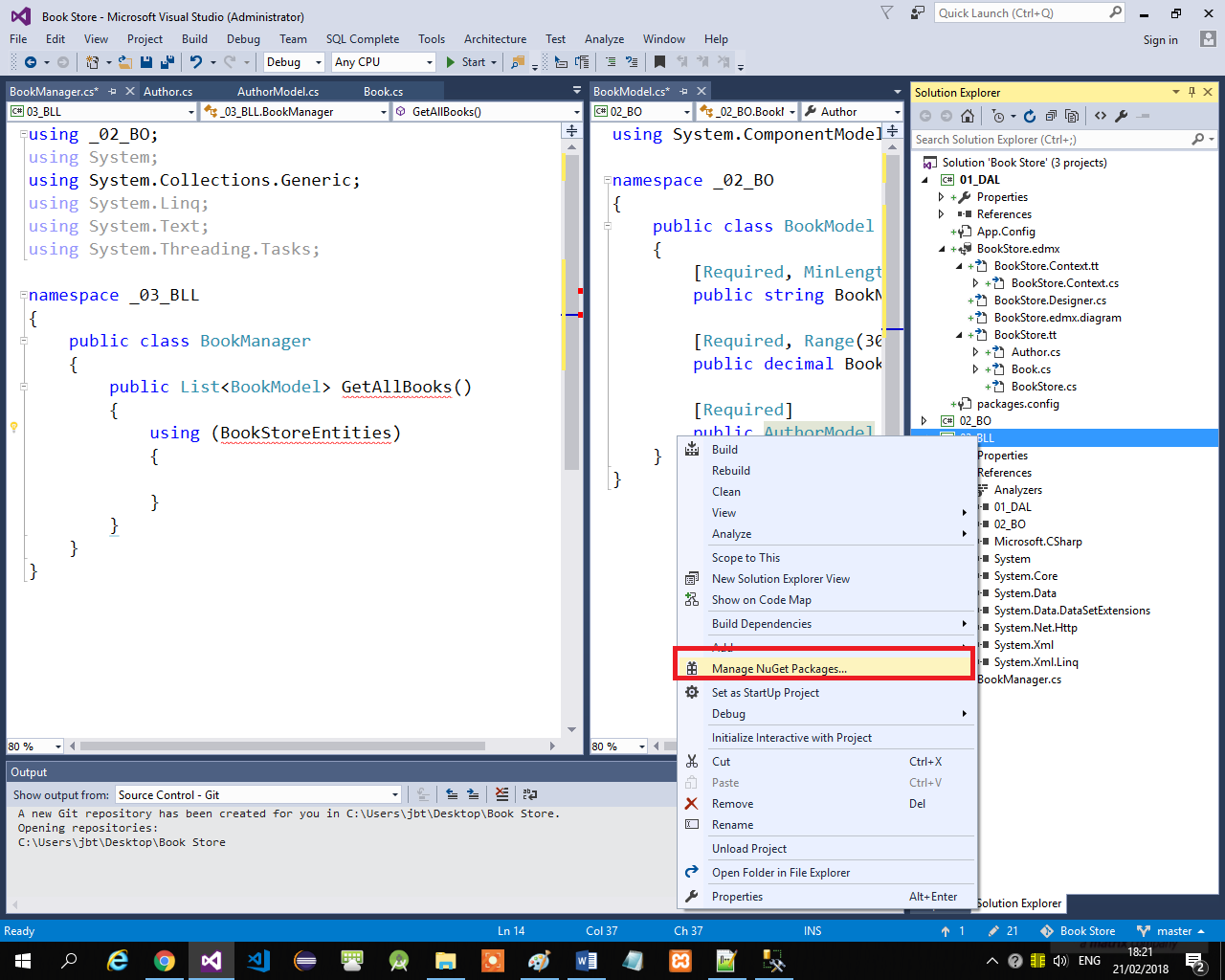
**Step 3– add to this project references to the DAL and BOL**



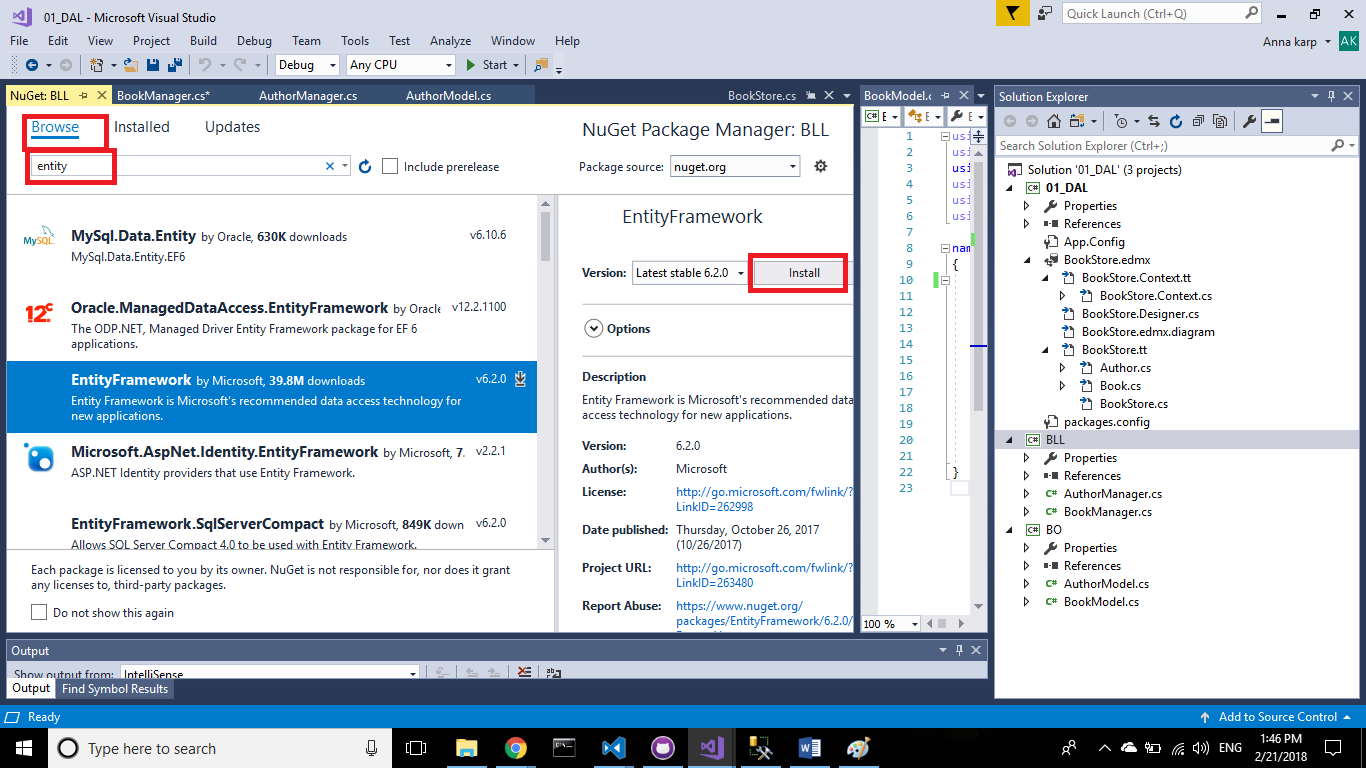
**Step 4– create a BookManager class – note: in the next step you will add ef and fix the error**



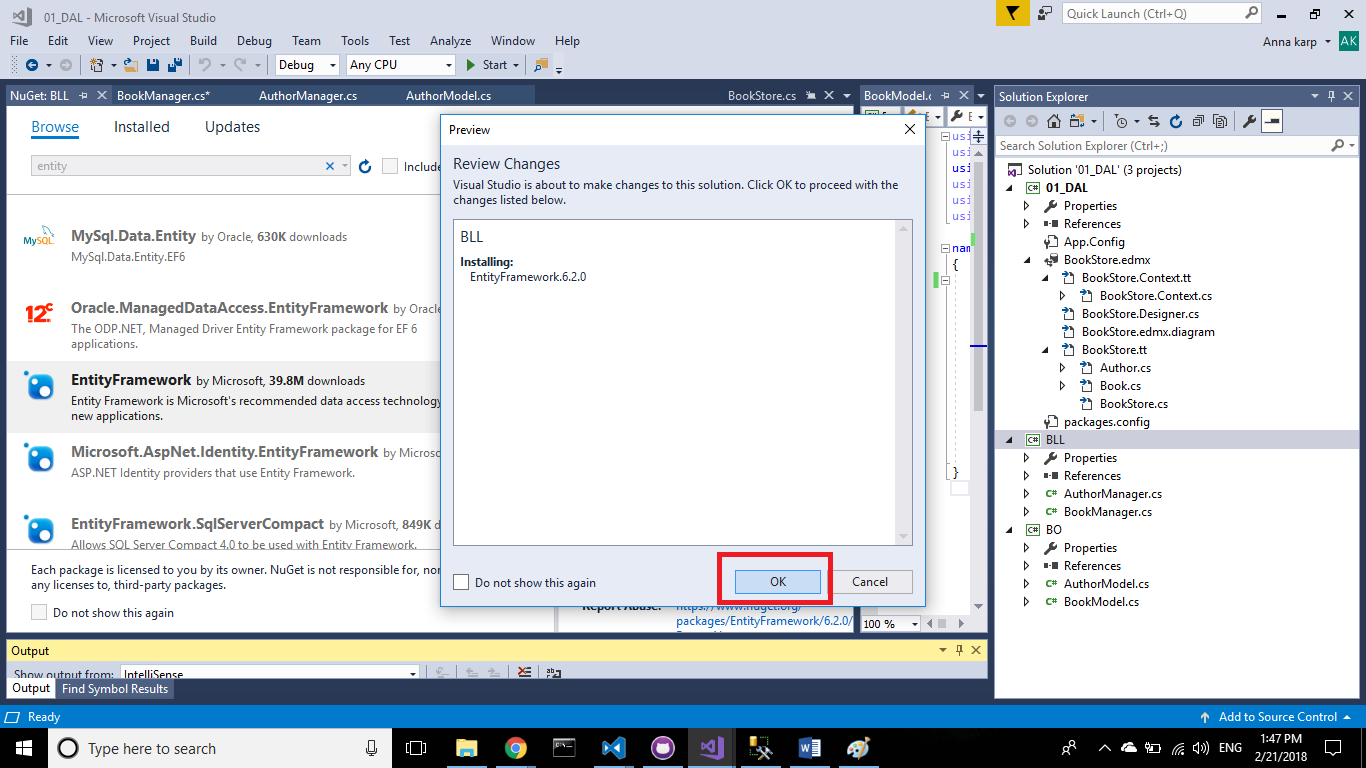
**Step 5– add ef to the BLL with the nuget packages manager**



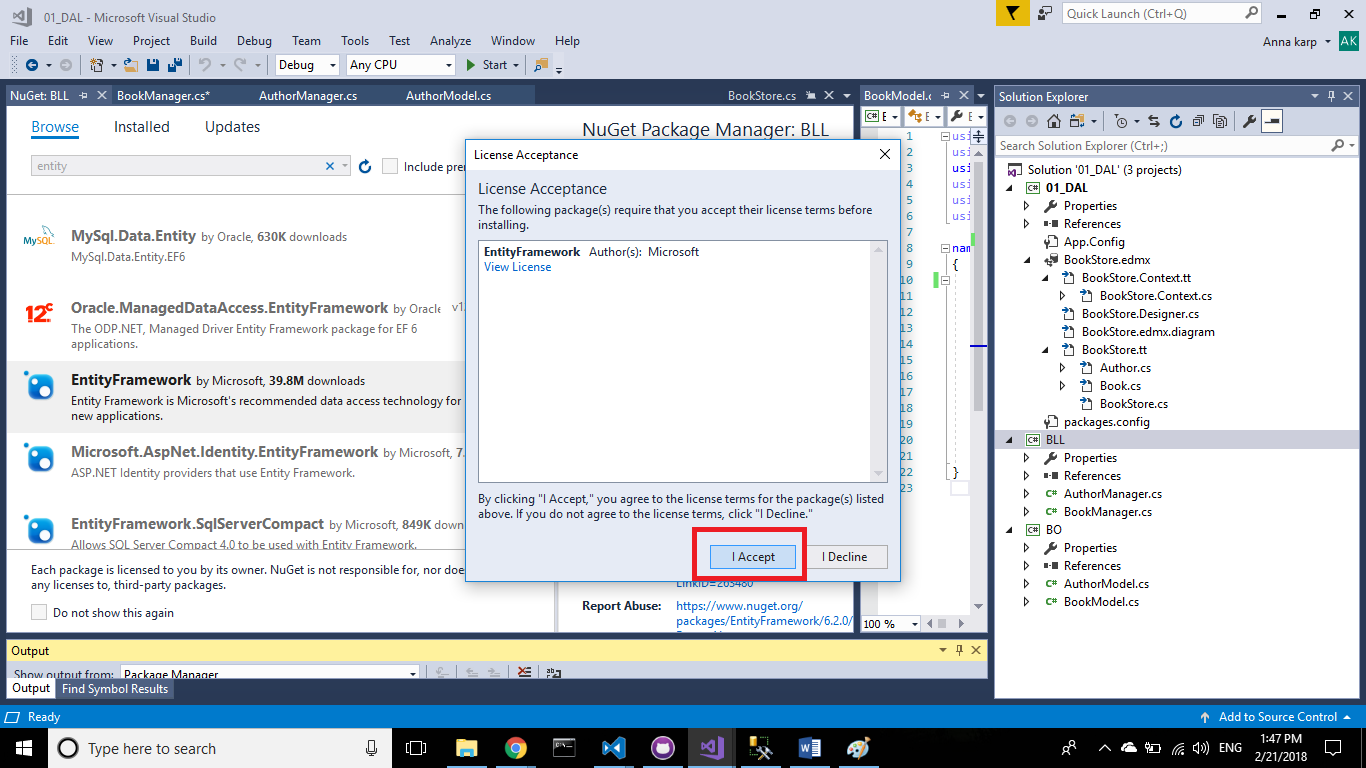
**Step 6– browse entity package**



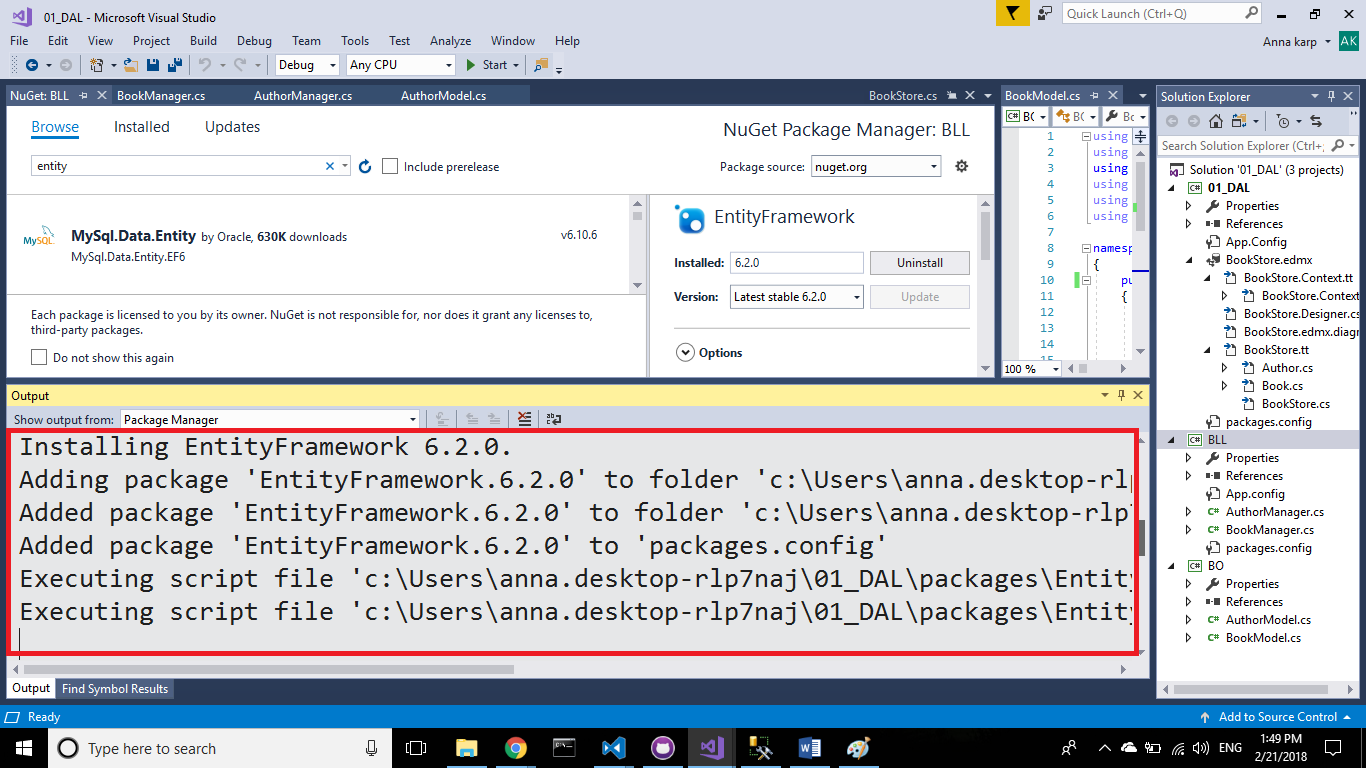
**Step 7– start adding the ef package**



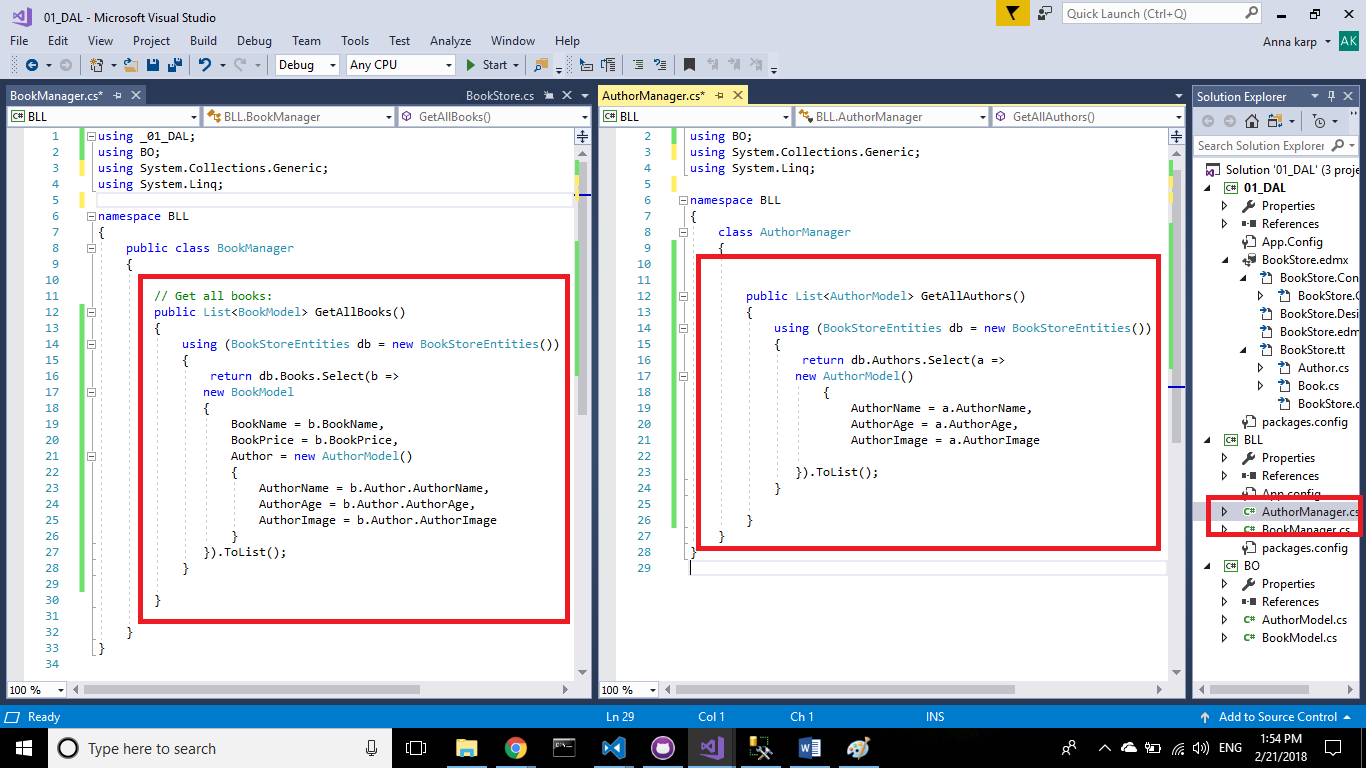
**Step 8– add the ef package**



**Step 9– the ef package has added successfully**



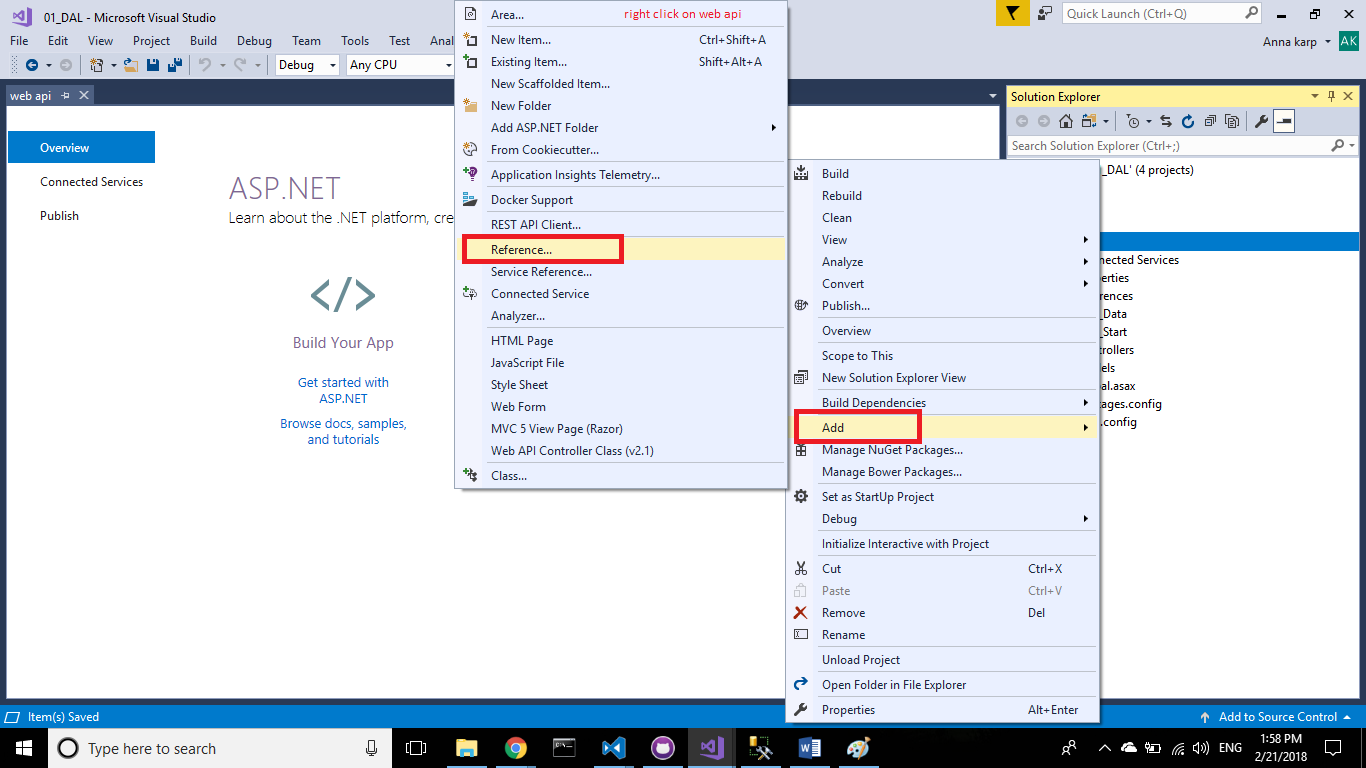
**Step 10– add the relevant logic to the BookManager and AouthorManager classes**



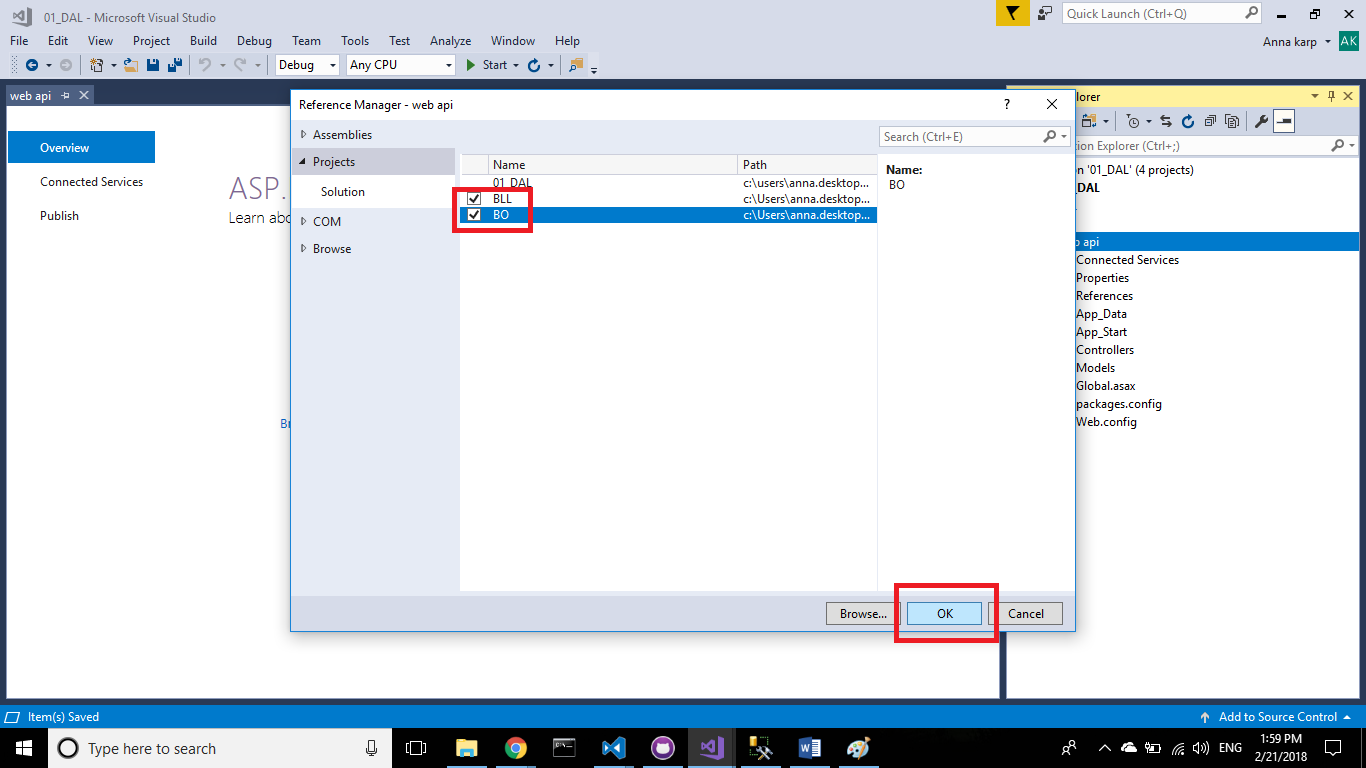
Part 5- create the console app

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

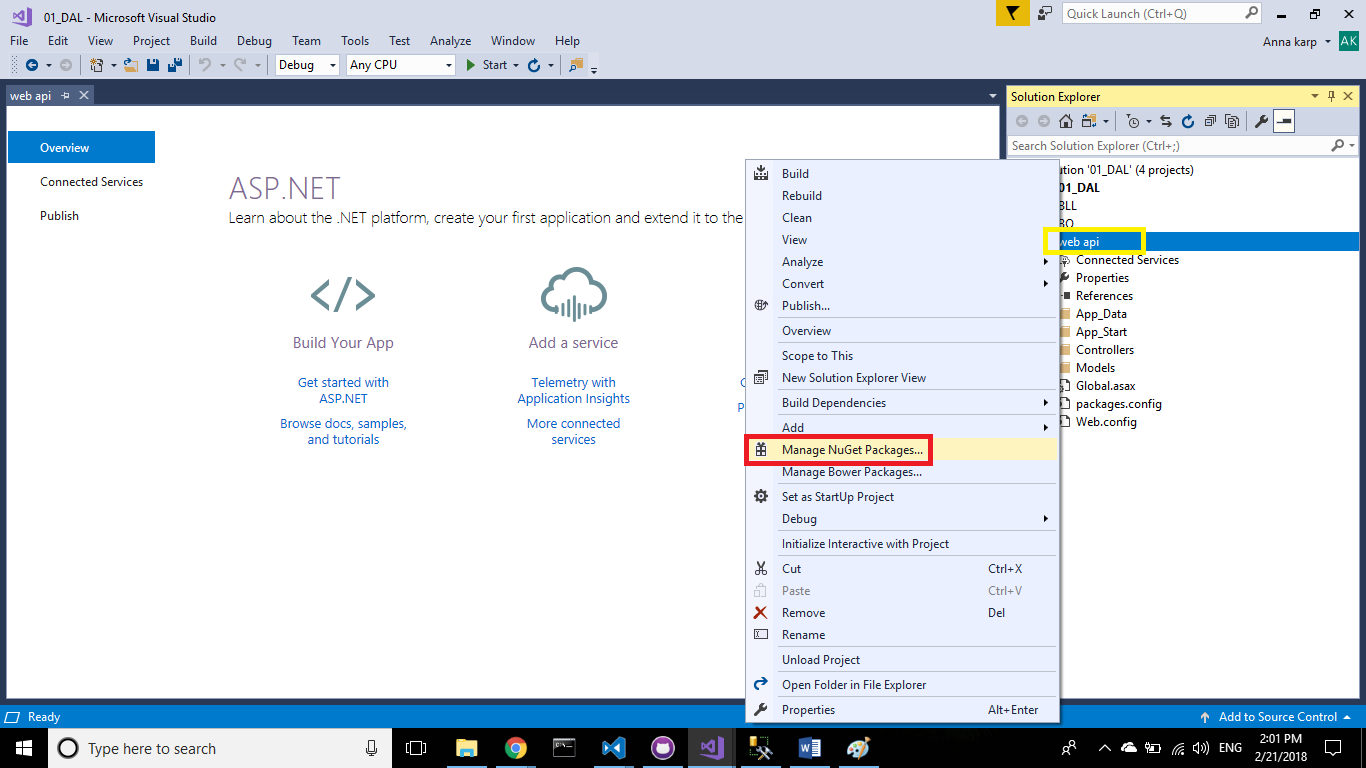
**Step 3– add references**



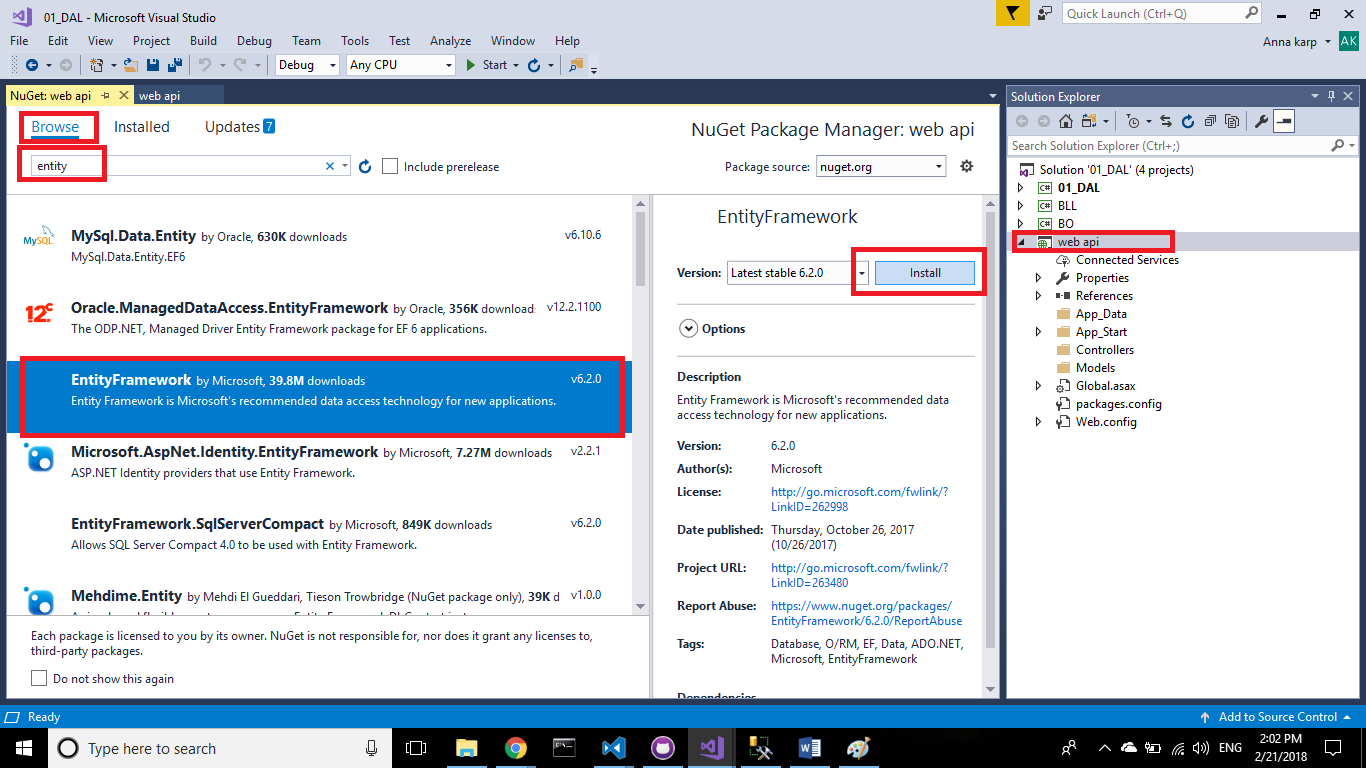
**Step 4– add references to the BOL and BLL**



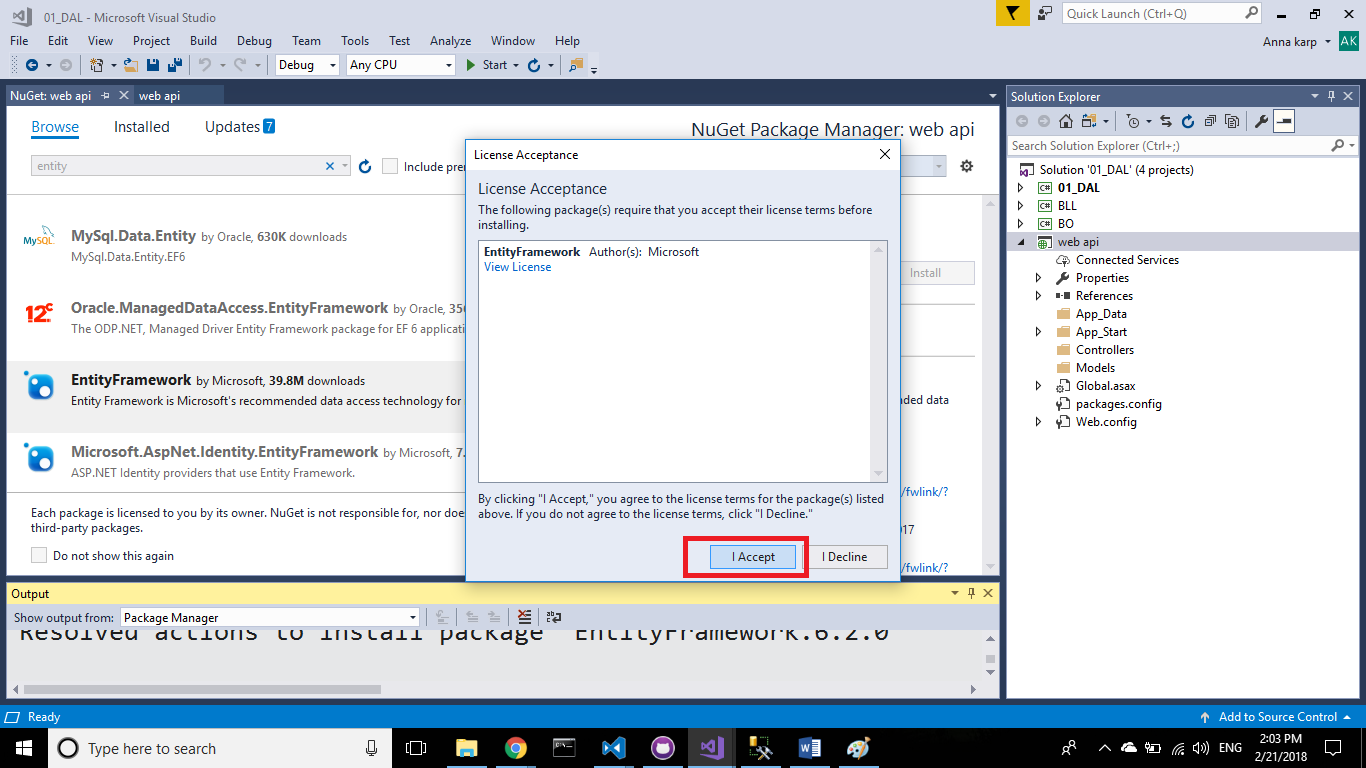
**Step 5– add a reference to the ef package with the nuget package manager**

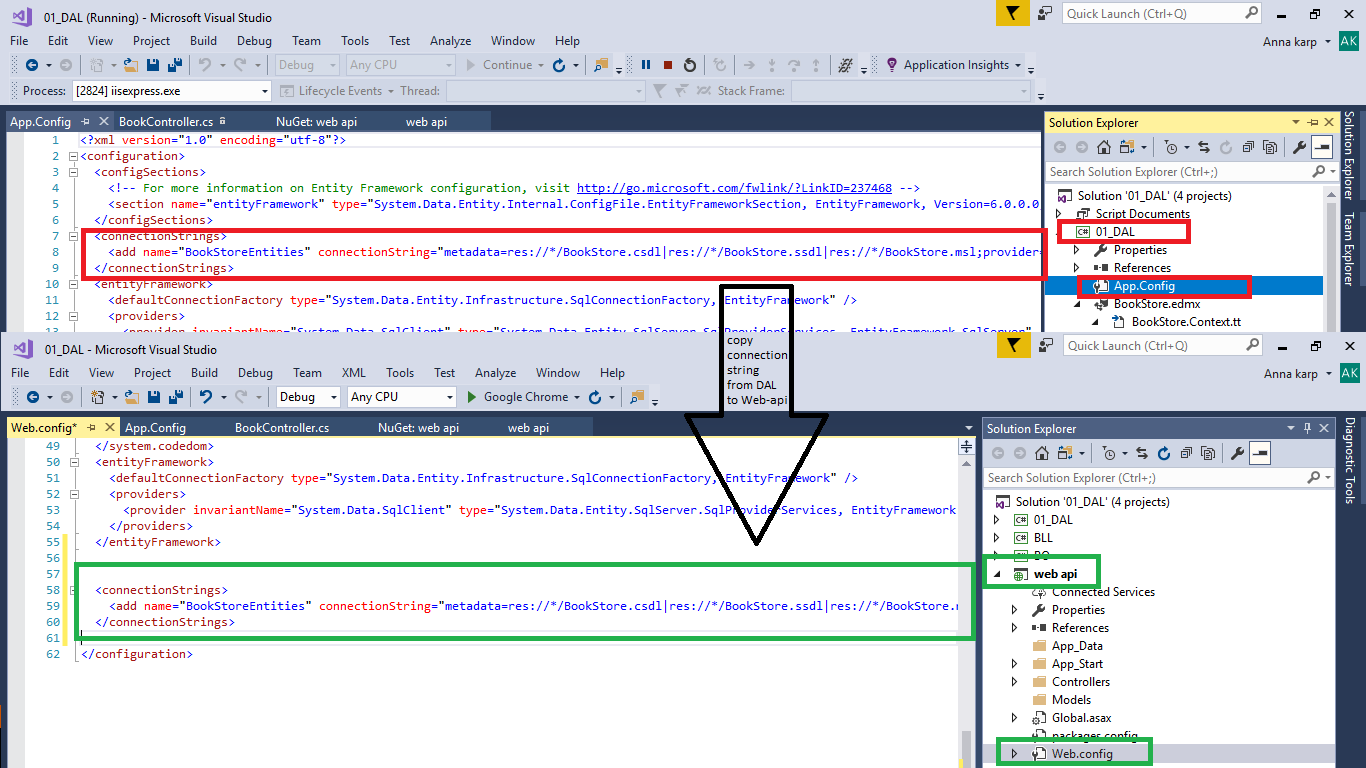


**Step 6– browse the entity option**



**Step 7– install the ef package**



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

GOOD LUCK!!!