

Entity Framework ASP.NET Web Application

Preface

This document was intended for .NET framework and not for .NET core.

Further documentation can be found here:

<https://docs.microsoft.com/en-us/ef/ef6/get-started>

Step 1

Create a new ASP.NET Web Application (.NET Framework)

Step 2

Create a new database project

Step 3

Install Entity Framework

Step 4

Create Tables & Foreign Keys

Step 5

Create Entity Data Model

Step 6

Create a Controller & Test Controller using Postman

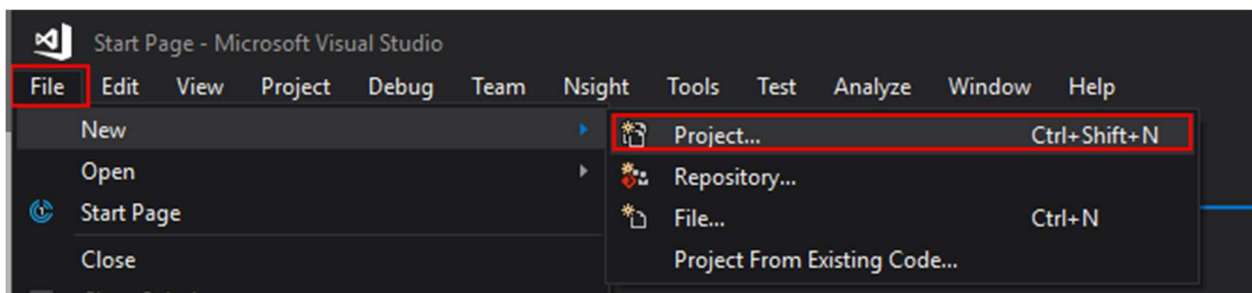
Notes

See end of document for additional notes

Step 1

Create a new ASP.NET Web Application (.NET Framework)

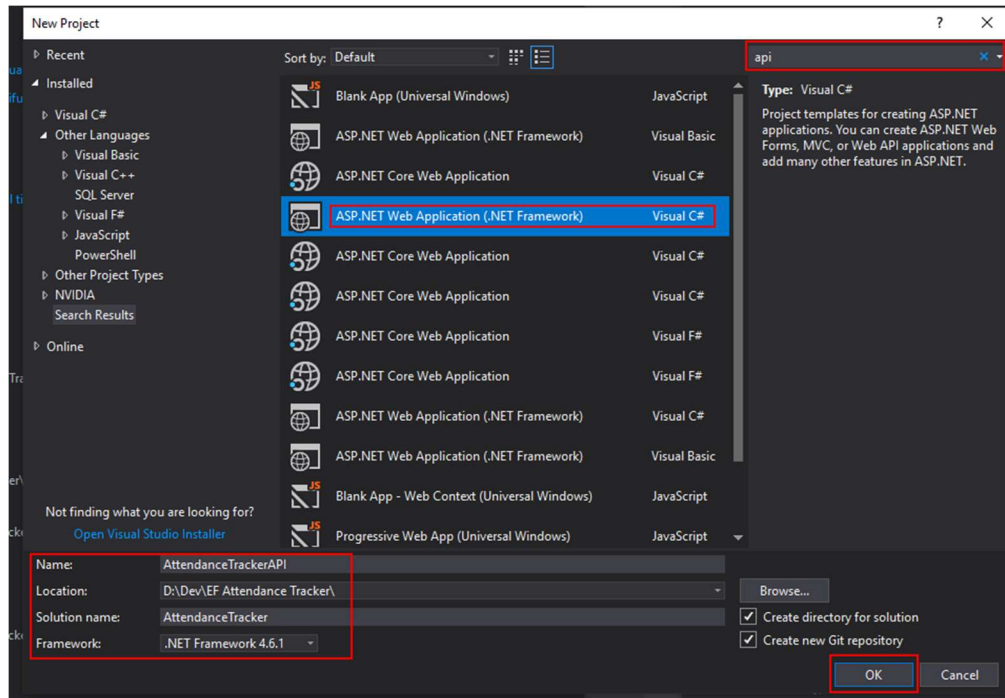
1. Create a new solution
 - a. **File > New > Project**



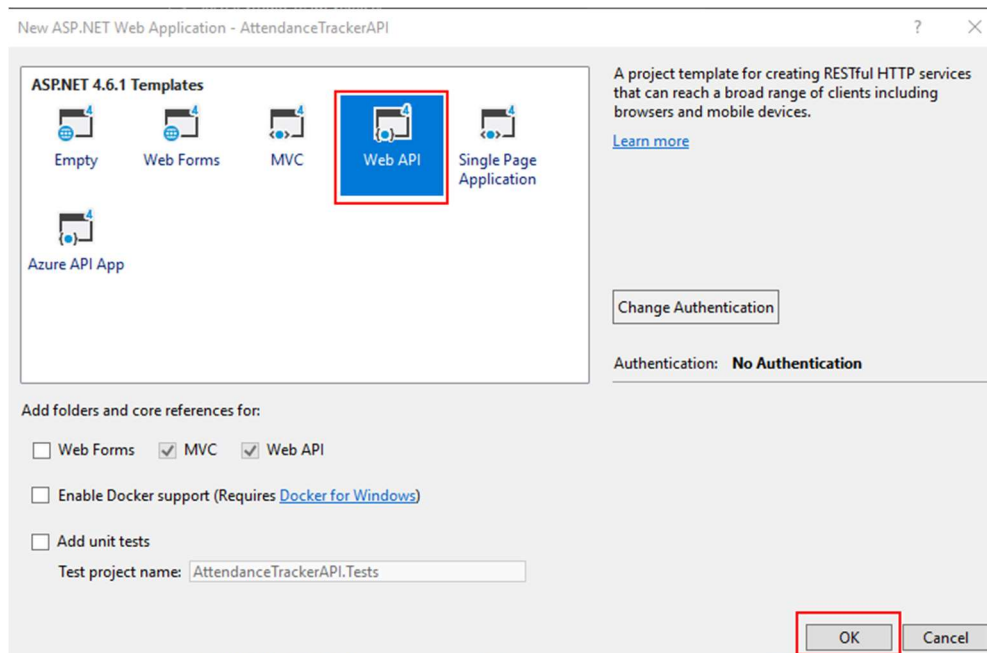
Entity Framework ASP.NET Web Application

2. Create a ASP.NET Web Application (.NET Framework)

- b. Search *API*
- c. Click on ASP.NET web app for C#
- d. Name the Web Application
- e. Select location
- f. Name your solution
- g. Press OK



3. Select Web API template and press OK

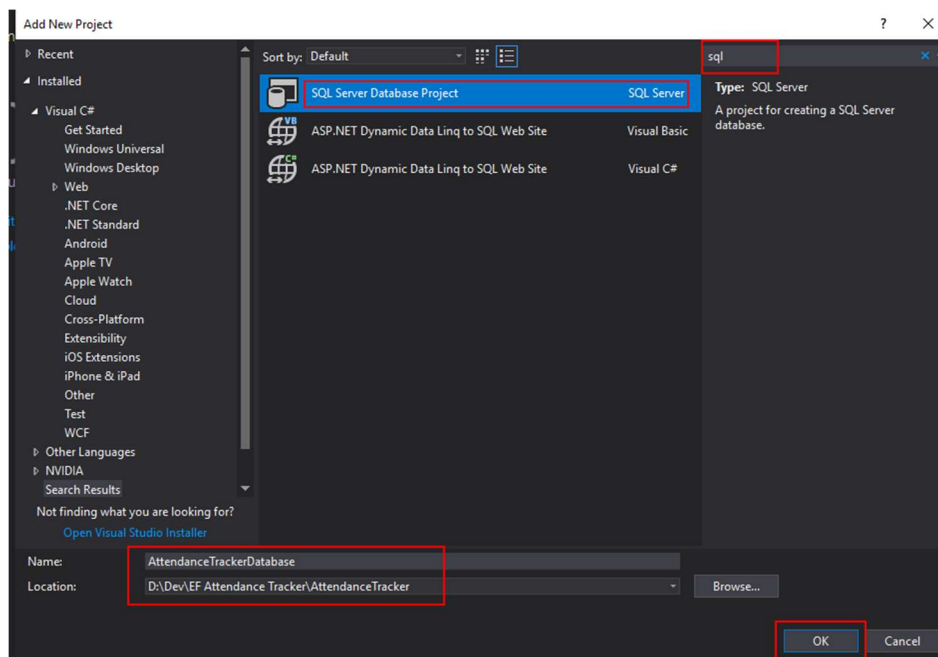
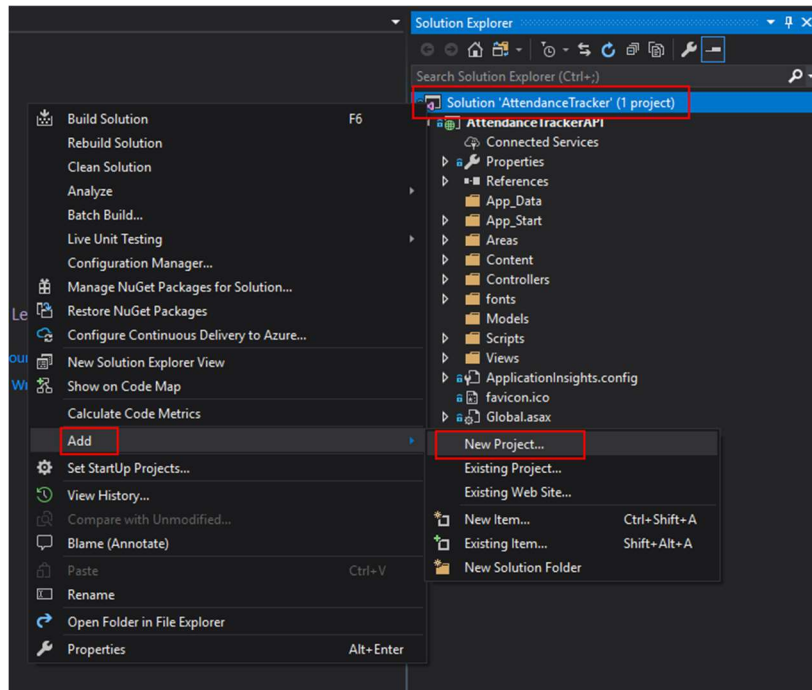


Entity Framework ASP.NET Web Application

Step 2

Create a new database project

1. Add a new database project
 - a. Right click on the **Solution** > **Add** > **New Project**
 - b. Search for **SQL**
 - c. Select **SQL Server Database Project**
 - d. Name the Database
 - e. Press OK

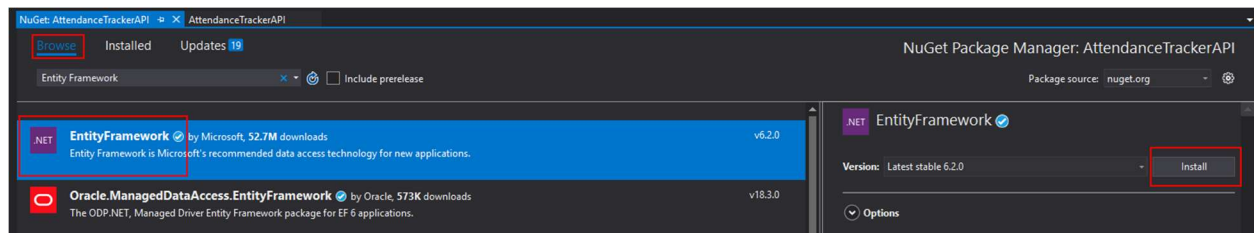
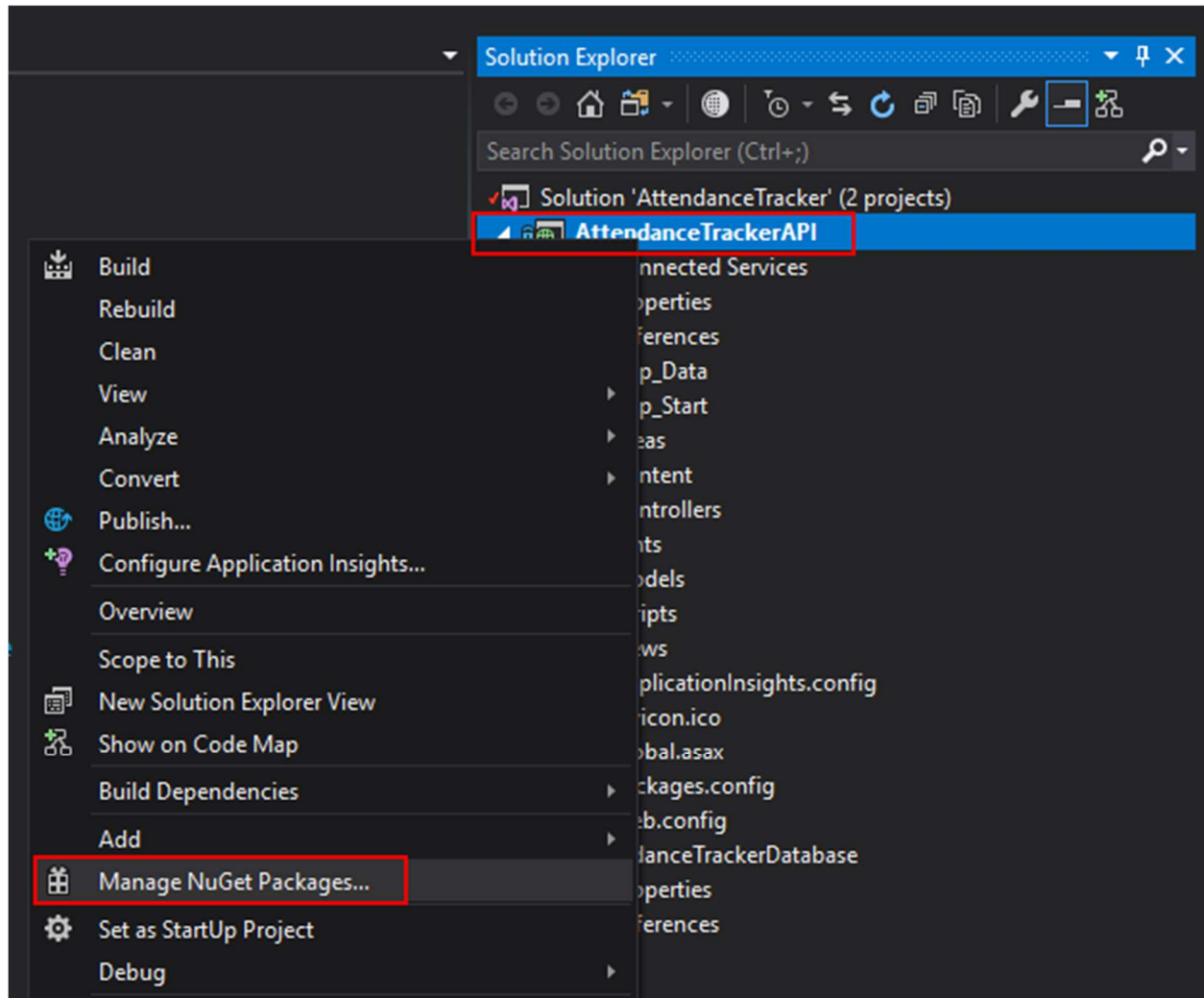


Entity Framework ASP.NET Web Application

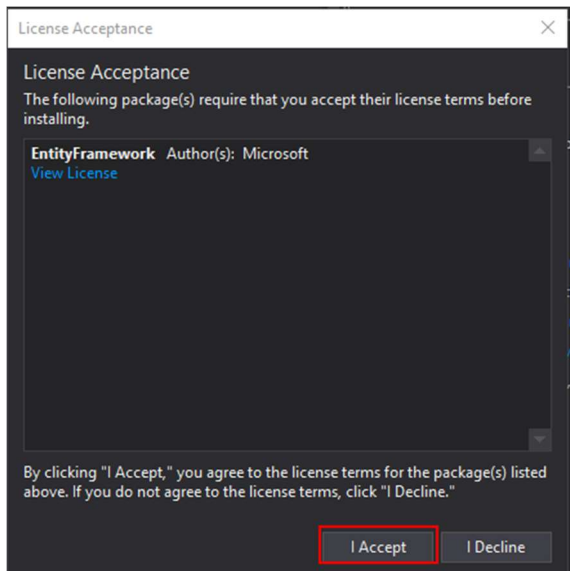
Step 3

Install Entity Framework

1. Add Entity Framework to your ASP.NET Web Application
 - a. Right click on the **Project > Manage NuGet Packages**
 - b. Select browse
 - c. Search for *Entity Framework*
 - d. Select Entity Framework and Install
 - e. Accept the License



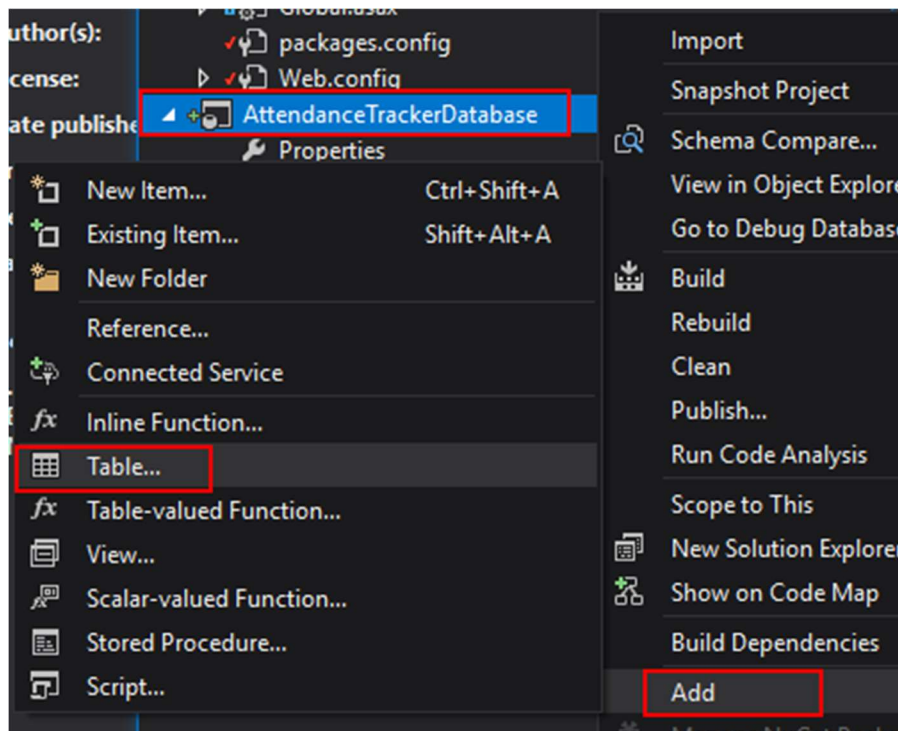
Entity Framework ASP.NET Web Application



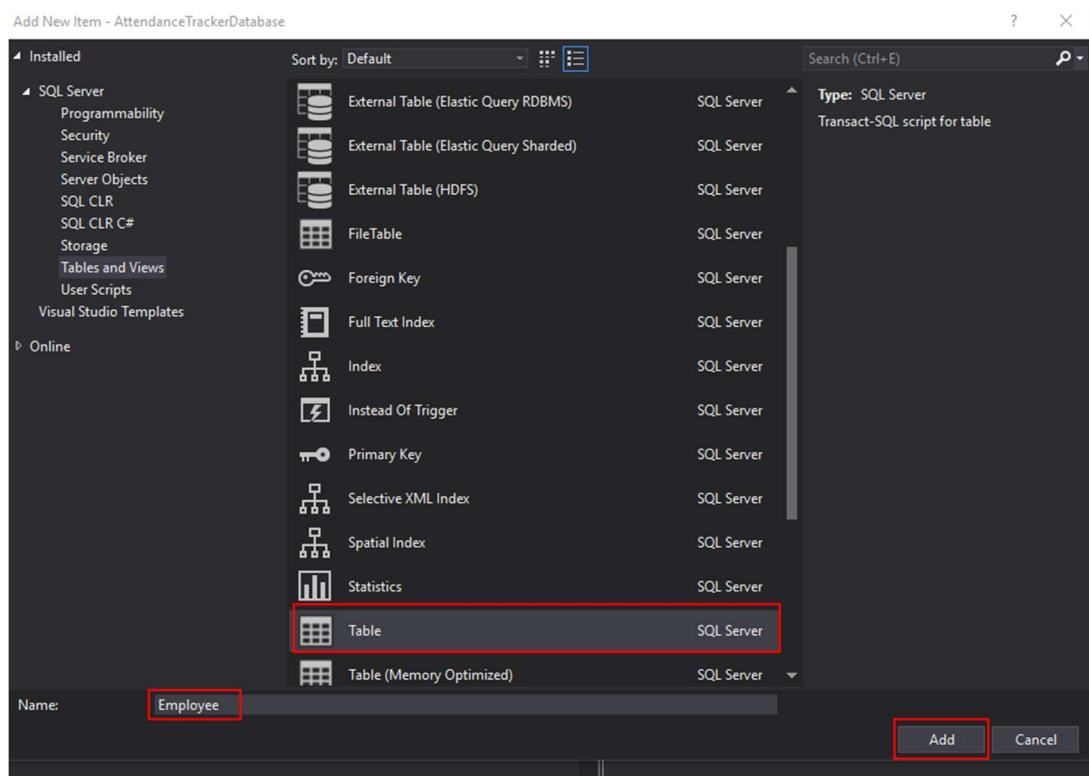
Step 4

Create Tables & Foreign Keys

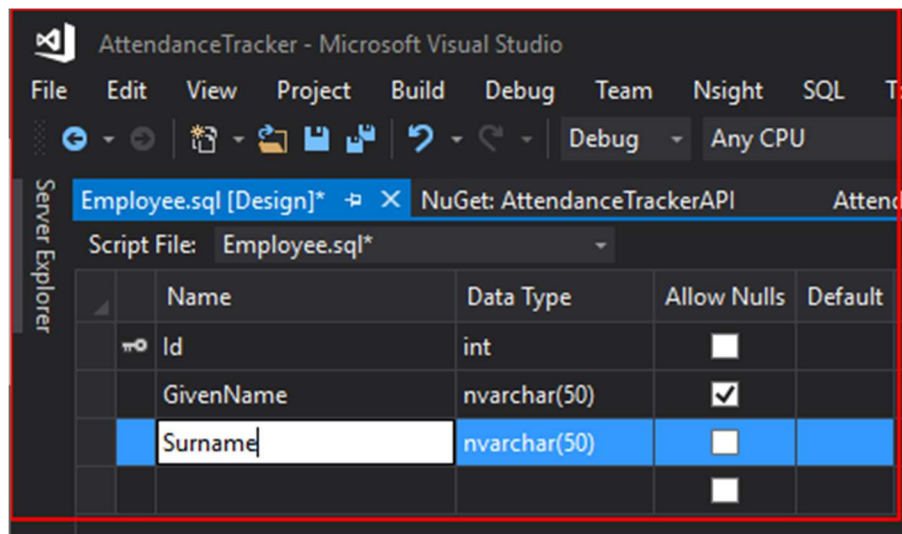
1. Create some Database Tables
 - a. Right click on the **Database Project** > **Add** > **Table**
 - b. Select Table
 - c. Name Table
 - d. Press Add



Entity Framework ASP.NET Web Application

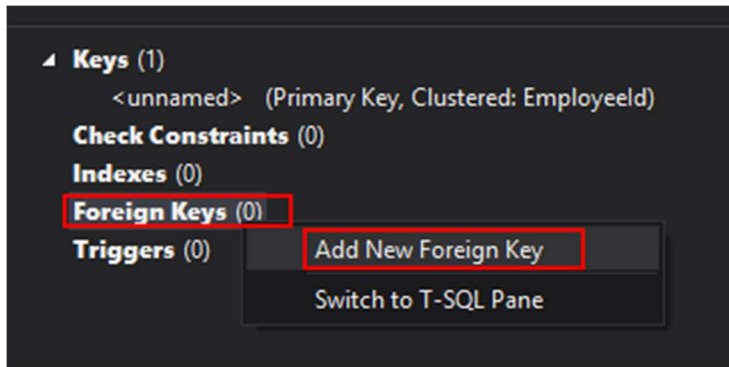


2. The table designer will open, this is where you can add columns into the table
 - a. Click on a blank field to add new column headers to the table
 - b. Select the appropriate data type
 - i. `nvarchar` will dynamically size
 - ii. `nchar` will add white space to the field value until 10 characters are in total.
For example:
Suppose `GivenName` has a value of "Bill" that's 4 characters, however, `nchar` adds 6 whitespace characters the result will be "Bill "
 - c. Make any other necessary changes



Entity Framework ASP.NET Web Application

3. Once you have 2 tables created you can create a foreign key constraint
 - a. Right click on **Foreign Keys** > **Add New Foreign Key**
 - b. Press Enter
 - c. Update the T-SQL
 - i. CONSTRAINT = Foreign key name
 - ii. FOREIGN KEY = Column name
 - iii. REFERENCES = Table and Primary Key



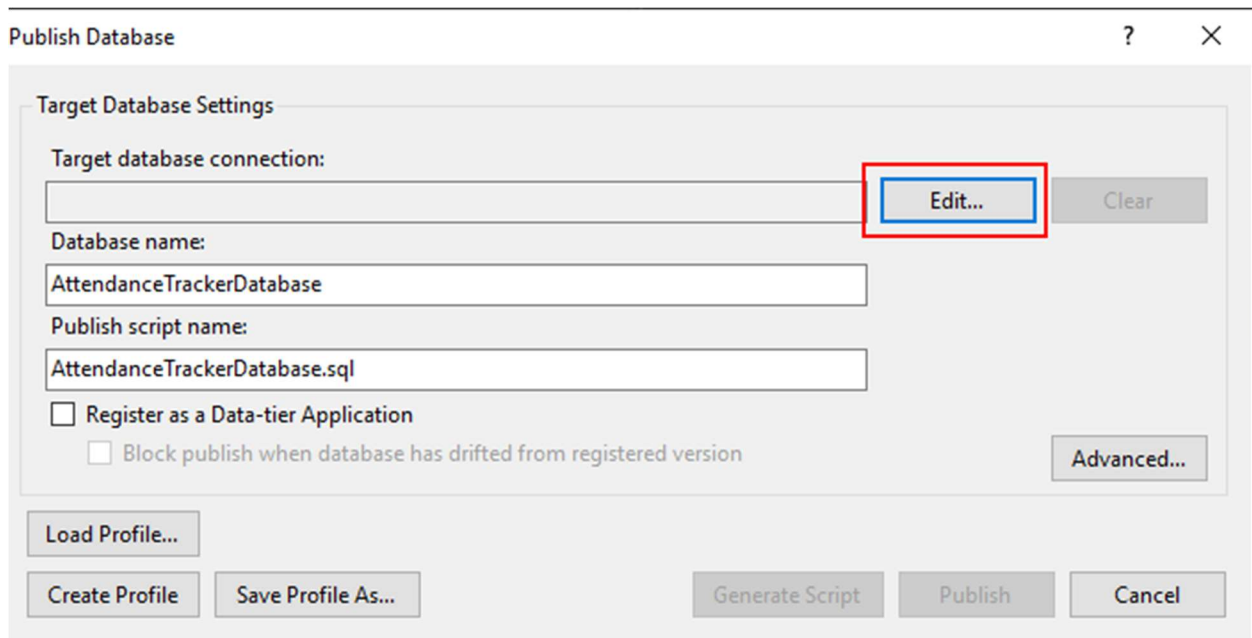
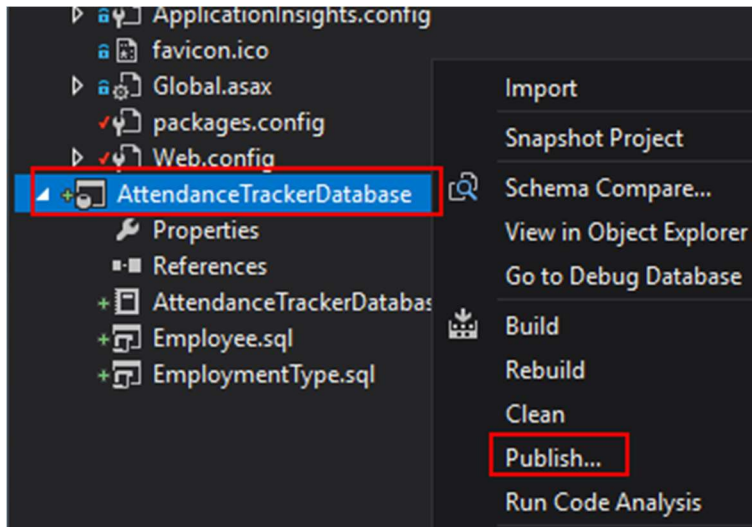
```
1 CREATE TABLE [dbo].[Employee]
2 (
3     [EmployeeId] INT NOT NULL PRIMARY KEY,
4     [GivenName] NVARCHAR(50) NULL,
5     [Surname] NVARCHAR(50) NOT NULL,
6     [TypeOfEmployment] NVARCHAR(50) NULL,
7     CONSTRAINT [FK_Employee_EmploymentType] FOREIGN KEY ([TypeOfEmployment]) REFERENCES [EmploymentType]([TypeOfEmployment])
8 )
9
```

Employee.sql [Design]				
Script File: Employee.sql				
	Name	Data Type	Allow Nulls	D
PK	EmployeeId	int	<input type="checkbox"/>	
	GivenName	nvarchar(50)	<input checked="" type="checkbox"/>	
	Surname	nvarchar(50)	<input type="checkbox"/>	
	TypeOfEmployment	nvarchar(50)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

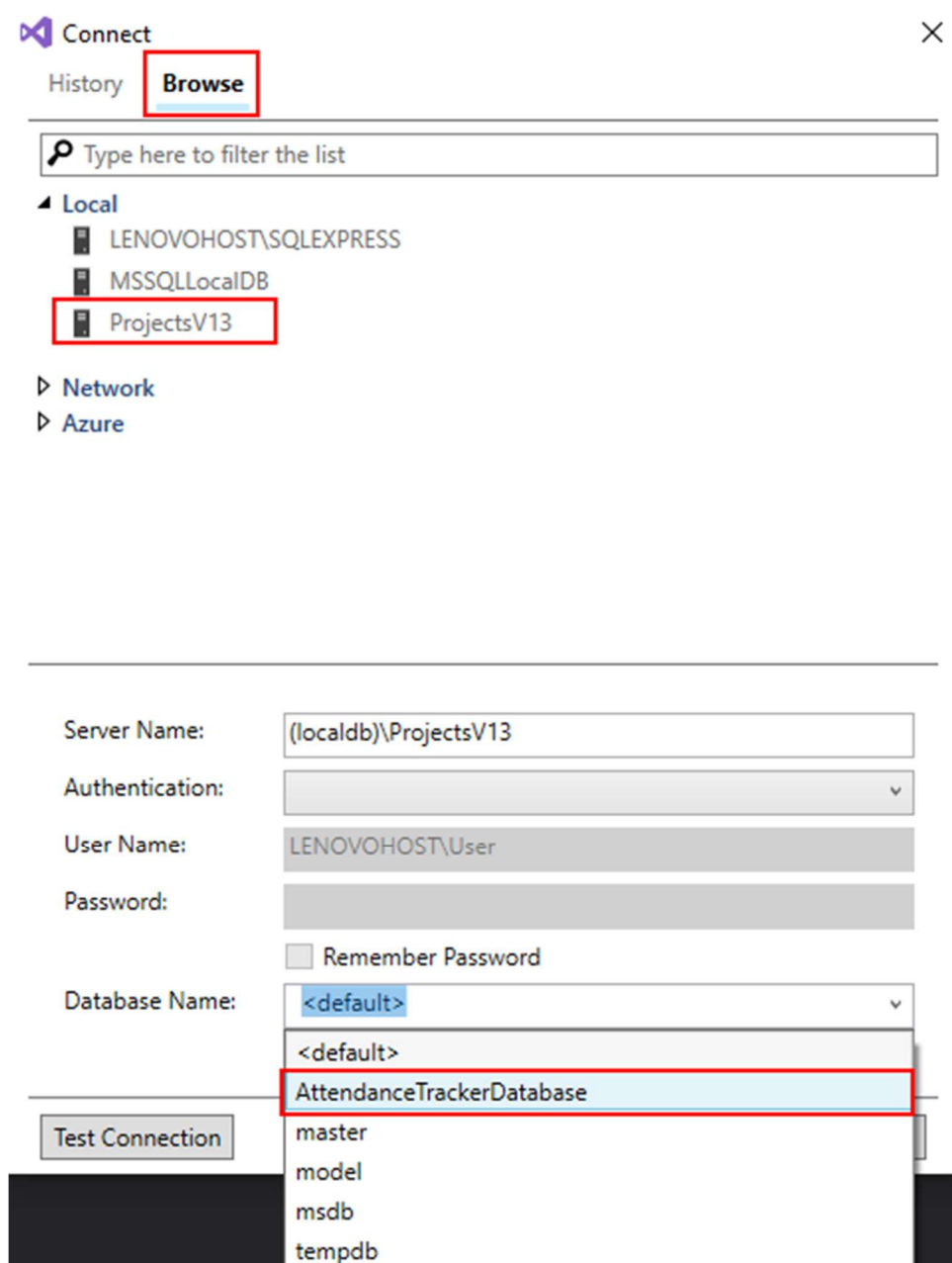
EmploymentType.sql [Design]			
Script File: EmploymentType.sql			
	Name	Data Type	Allow Nulls
PK	TypeOfEmployment	nvarchar(50)	<input type="checkbox"/>
			<input type="checkbox"/>

Entity Framework ASP.NET Web Application

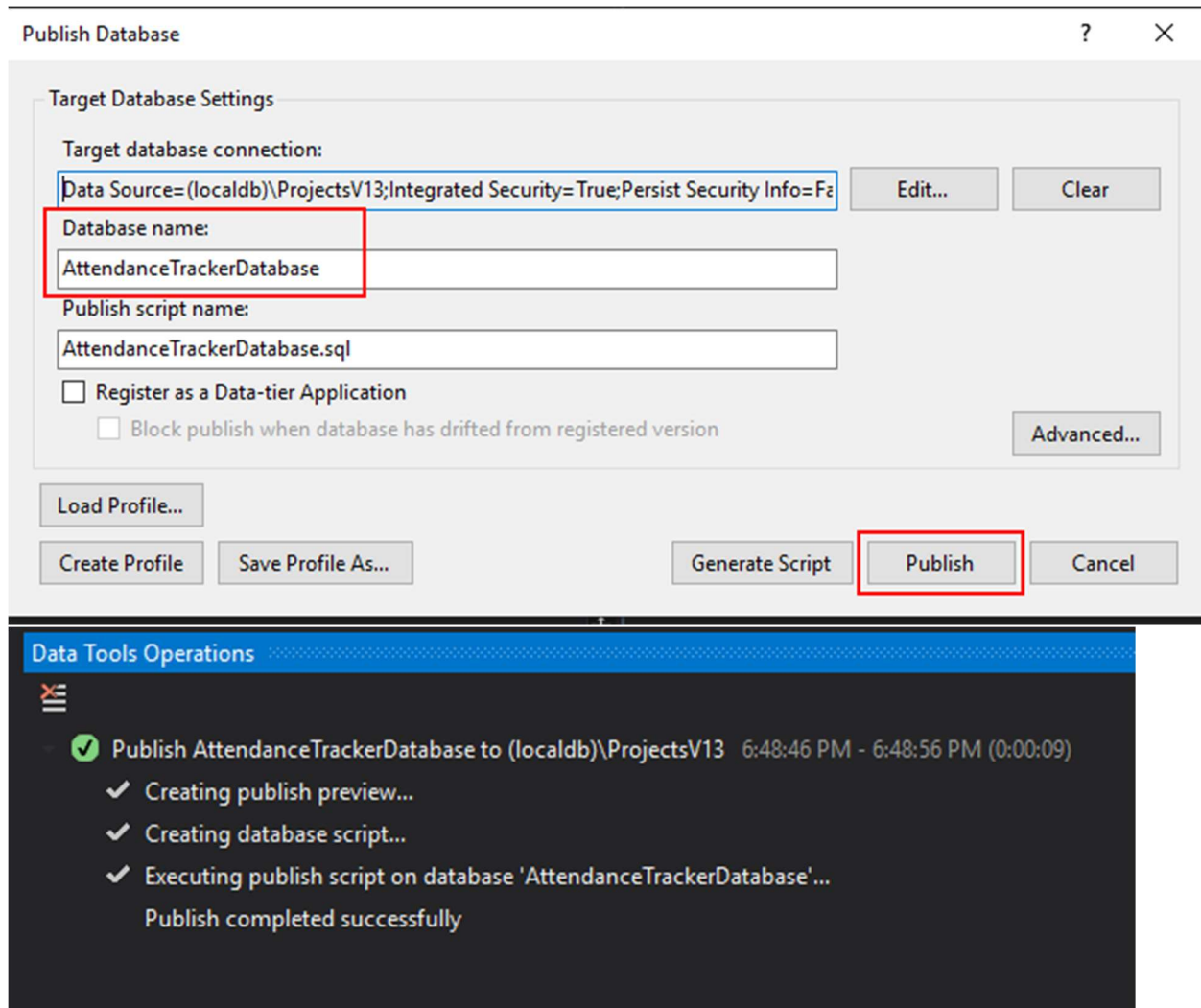
4. Publish the Database
 - a. Right click on the **Database Project** > **Publish**
 - b. Press Edit
 - c. Select **Browse** > **Local** > **ProjectsV13**
 - d. Select the Database name from the drop-down menu
 - e. Press OK
 - f. Confirm Database name is correct
 - g. Press Publish



Entity Framework ASP.NET Web Application



Entity Framework ASP.NET Web Application

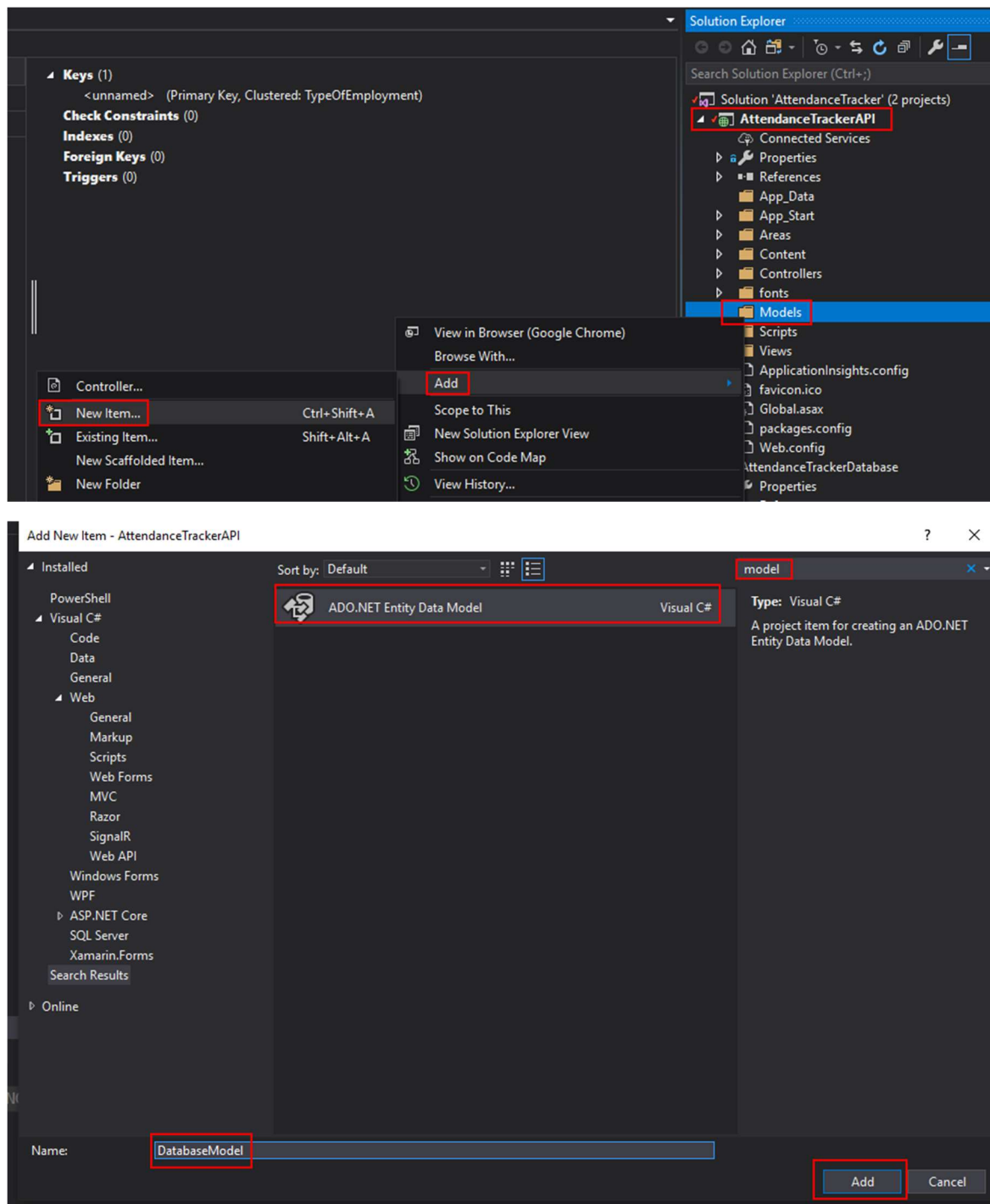


Step 5

Create Entity Data Model

1. Right click on the **Model** folder in the Web App Project > **Add > New Item**
2. Search *Model*
3. Select **ADO.NET Entity Data Model** > **name model** > **press Add**
4. Select EF Designer from database and press Next
5. From the drop-down menu select the Database and press Next
6. Confirm your tables are all visible
7. Press Finish

Entity Framework ASP.NET Web Application



Entity Framework ASP.NET Web Application

Entity Data Model Wizard

Choose Model Contents

What should the model contain?

EF Designer from database Empty EF Designer model Empty Code First model Code First from database

Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.

< Previous **Next >** Finish Cancel

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

lenovohost\localdb#8f6ba5d3.AttendanceTrackerDatabase.dbo New Connection...

lenovohost\localdb#55de7263.master.dbo

lenovohost\localdb#8f6ba5d3.AttendanceTrackerDatabase.dbo that is required to connect this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

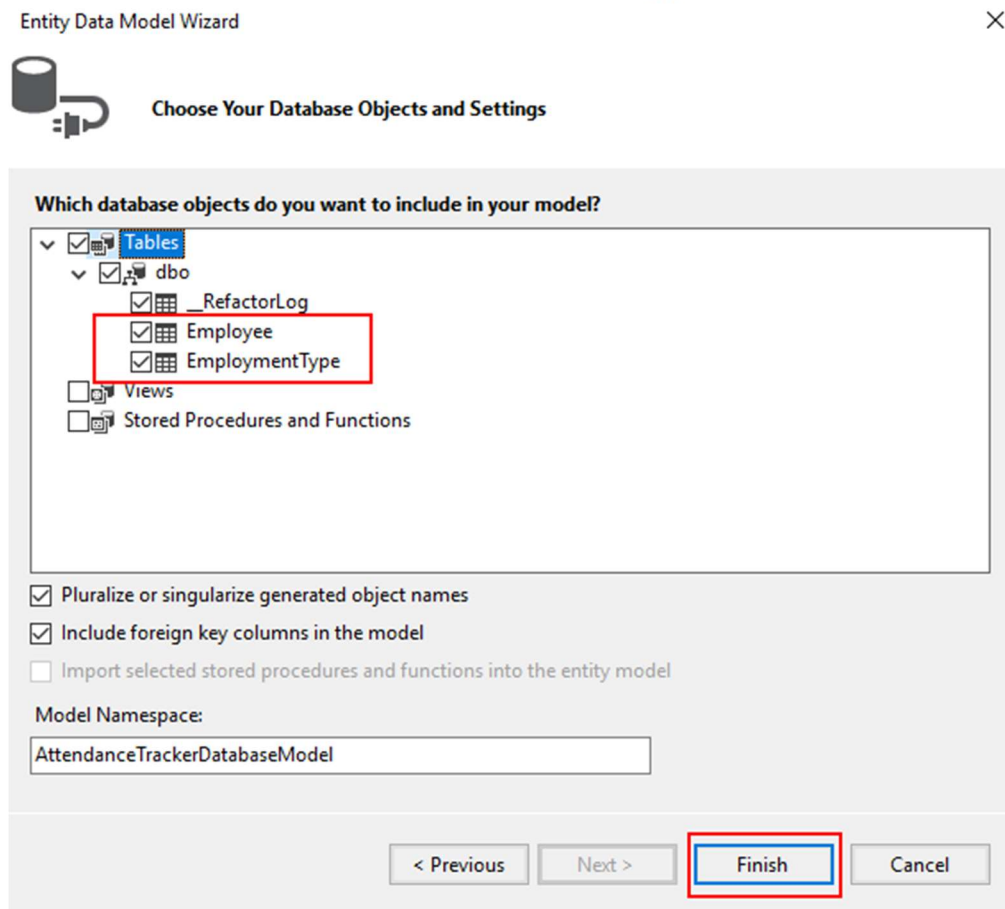
metadata=res://*/Models.DatabaseModel.csdl|res://*/Models.DatabaseModel.ssdl|res://*/Models.DatabaseModel.msl;provider=System.Data.SqlClient;provider connection string="data source=(localdb)\ProjectsV13;initial catalog=AttendanceTrackerDatabase;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"

☒ Save connection settings in Web.Config as:

AttendanceTrackerDatabaseEntities

< Previous **Next >** Finish Cancel

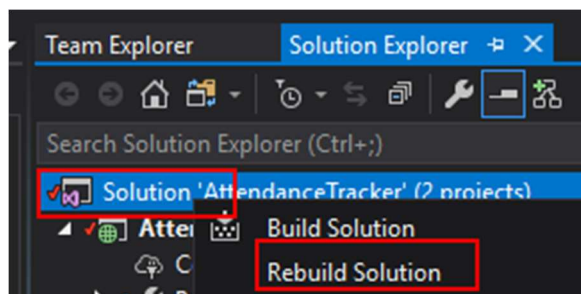
Entity Framework ASP.NET Web Application



Step 6

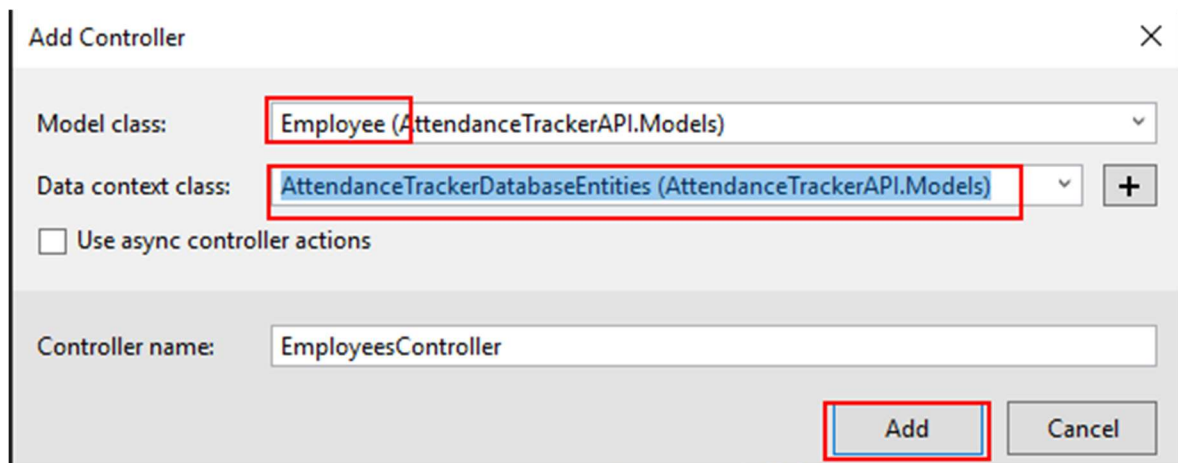
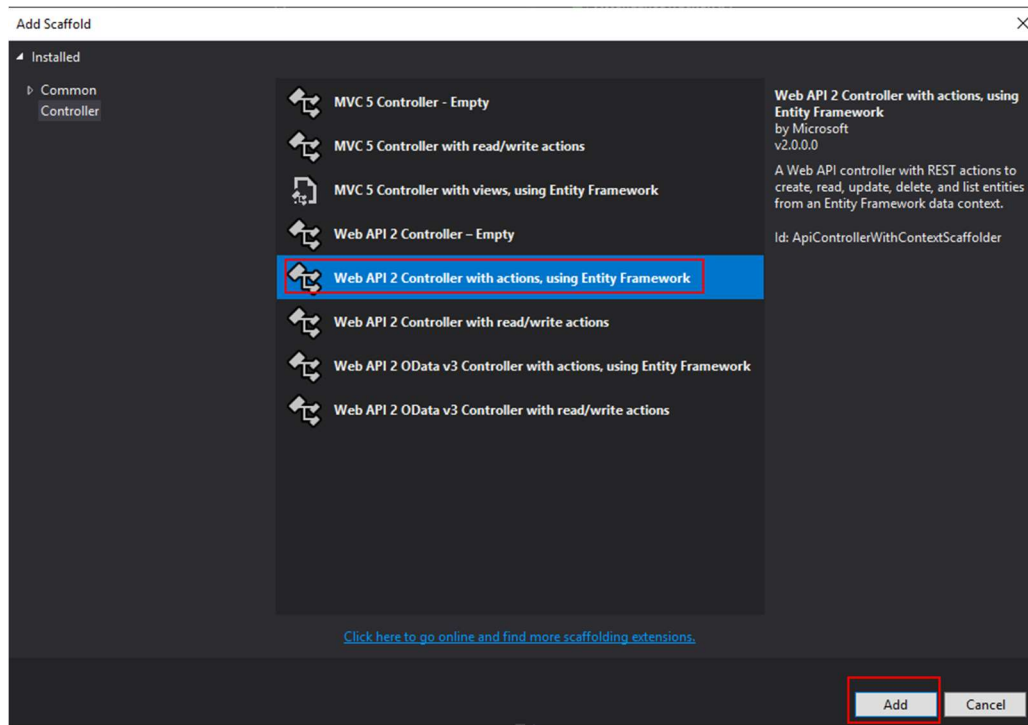
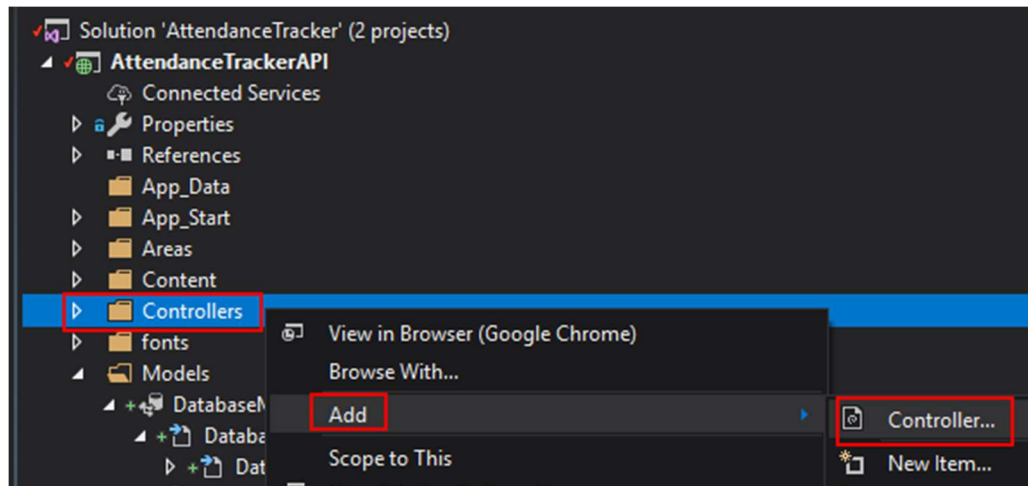
Create a Controller

1. Rebuild the solution



2. Right click on **Controllers folder > Add > Controller**
3. Select Web API 2 Controller with actions, using Entity Framework, press Add
4. Select the Model class to base the Controller on, press Add

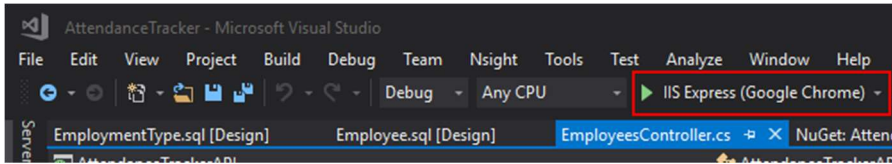
Entity Framework ASP.NET Web Application



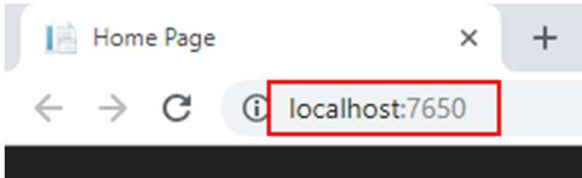
Entity Framework ASP.NET Web Application

Test controller from Postman

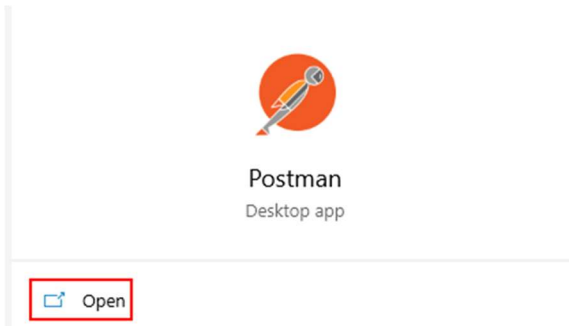
1. Start the IIS service



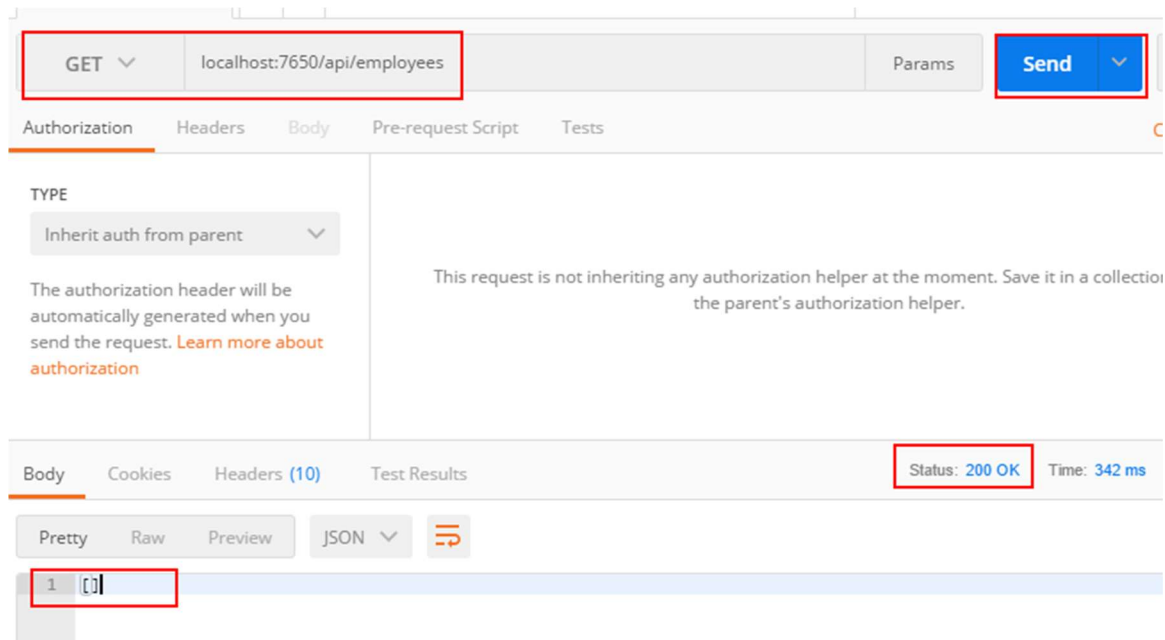
2. Gather the port number from the web browser



3. Open Postman Desktop App

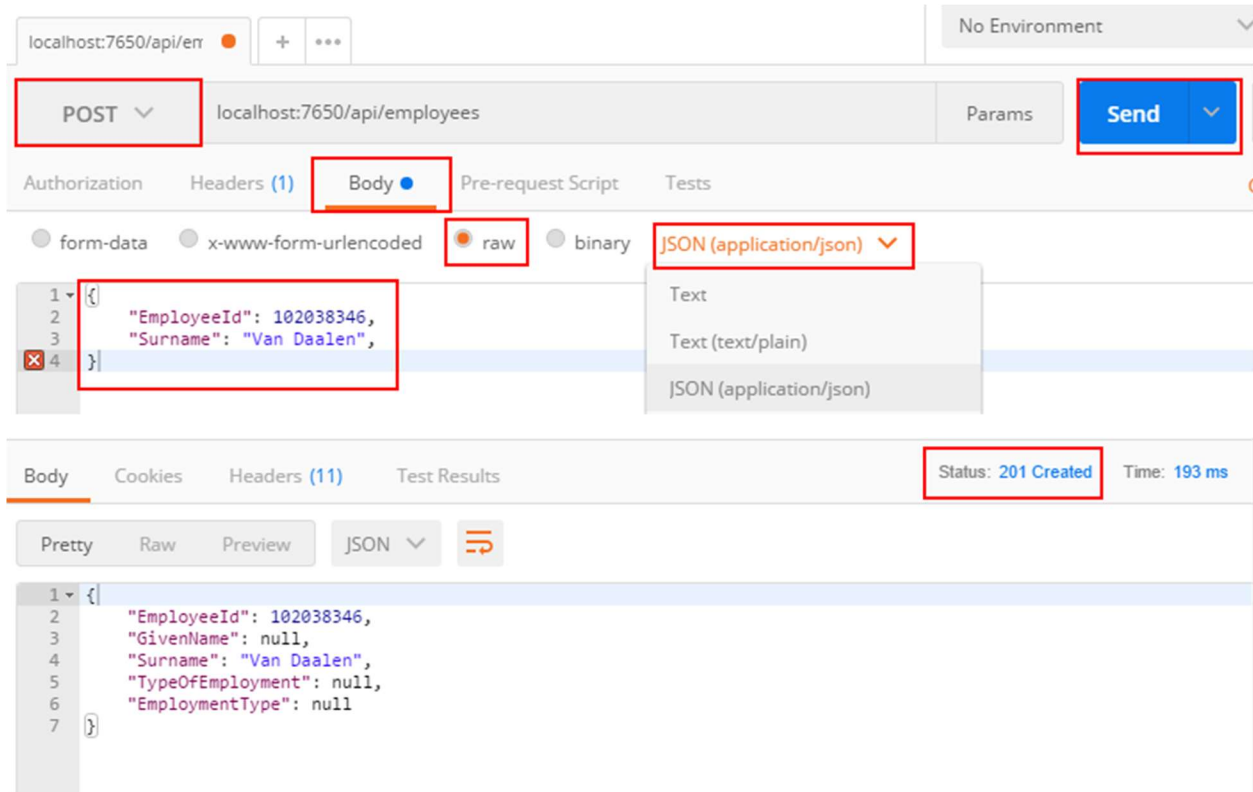


4. Send a GET request to the controller
 - a. Expected Status: 200 OK

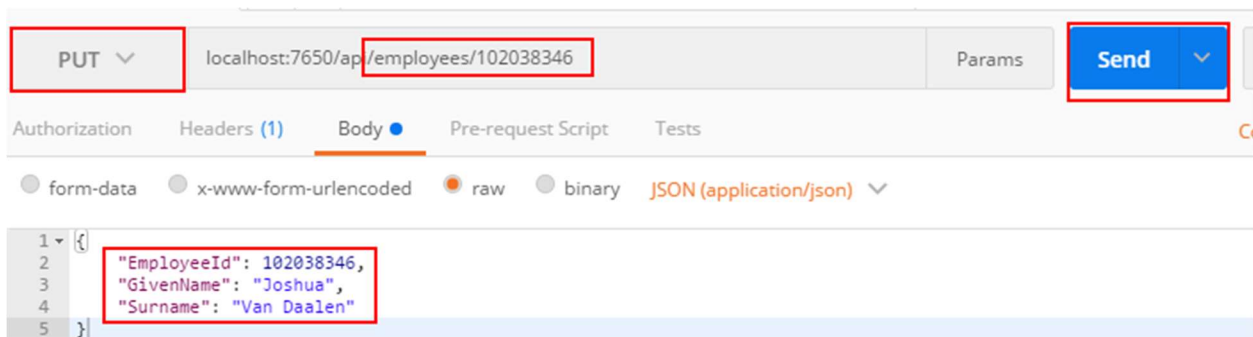


Entity Framework ASP.NET Web Application

5. Send a POST request to the controller
 - a. Set POST method
 - b. Click on Body
 - c. Click on RAW
 - d. Set JSON (application/json)
 - e. Create JSON Object
 - f. Press Send
 - g. Expected Status: 201 Created

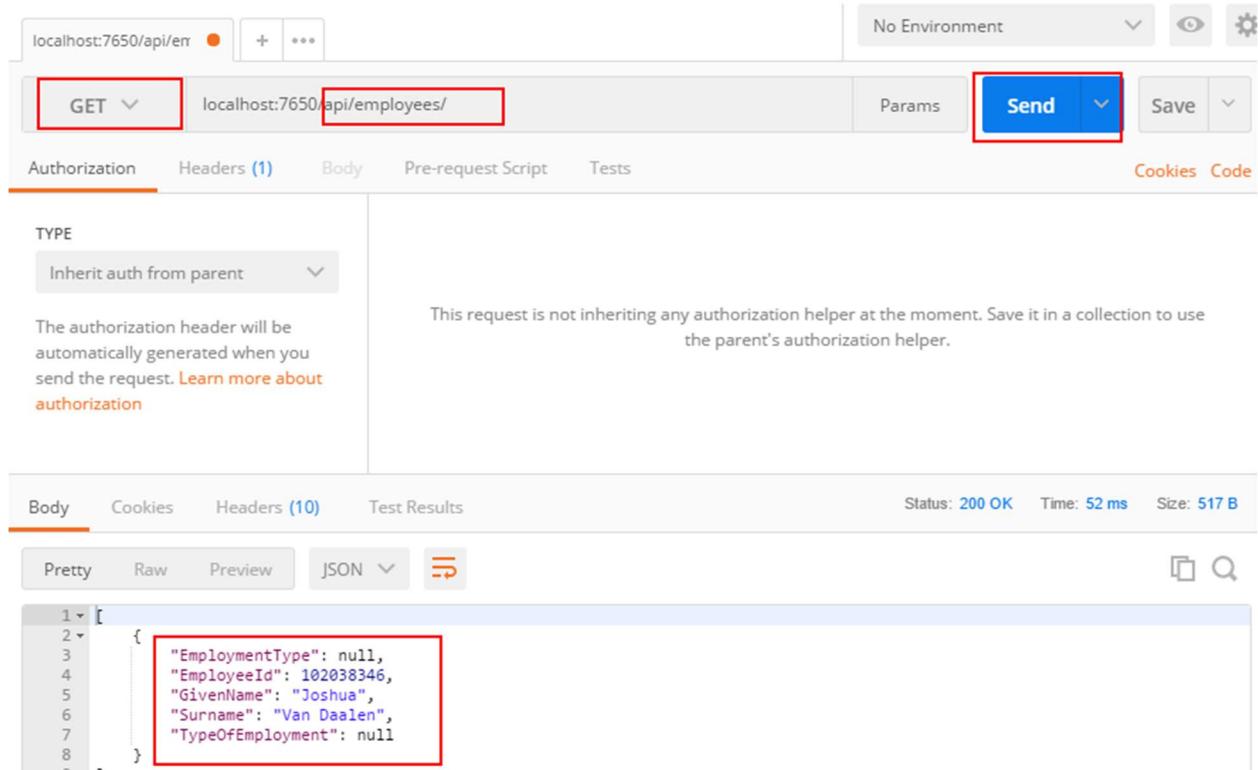


6. Send PUT request
 - a. Expected Status: 204 No Content



Entity Framework ASP.NET Web Application

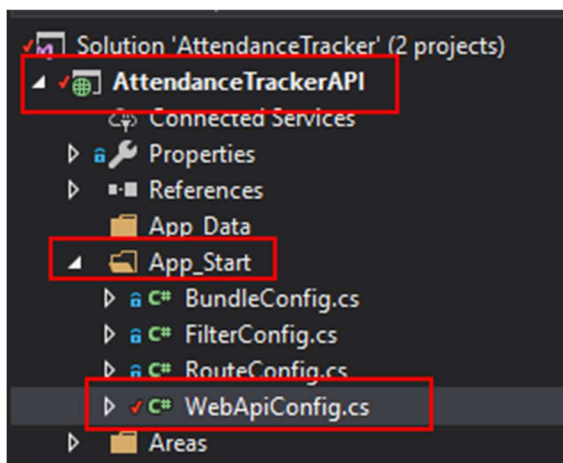
7. Send Get request again
 - a. Note the return body now shows the object we just Posted



Notes:

Additional code used to view JSON response from the web browser


1. Open the WebApiConfig.cs file



2. Add the following lines
 - a. jsonResponse = returns raw JSON to the web browser

Entity Framework ASP.NET Web Application

```
var jsonResponse = config.Formatters.JsonFormatter;  
jsonResponse.SerializerSettings.PreserveReferencesHandling =  
    Newtonsoft.Json.PreserveReferencesHandling.Objects;  
config.Formatters.Remove(config.Formatters.XmlFormatter);
```



localhost:7650/api/employees/ x +
← → ↻ ⓘ localhost:7650/api/employees/
[{"\$id":"1","EmploymentType":null,"EmployeeId":102038346,"GivenName":"Joshua","Surname":"Van Daalen","TypeOfEmployment":null}]

b. NoDollarSign = removed the \$id from the returning JSON object

```
var noDollarSign = config.Formatters.JsonFormatter;  
noDollarSign.SerializerSettings.PreserveReferencesHandling =  
    Newtonsoft.Json.PreserveReferencesHandling.None;
```



localhost:7650/api/employees/ x +
← → ↻ ⓘ localhost:7650/api/employees/
[{"EmploymentType":null,"EmployeeId":102038346,"GivenName":"Joshua","Surname":"Van Daalen","TypeOfEmployment":null}]