Documentation:

The application holds **events**, which are the main app **entities**.

The functionality of the application supports the following CRUD operations:

creating, **reading, editing, updating** and **deleting** events.

The application stores the data into a **database.**

Your application is built of the **following technology**:

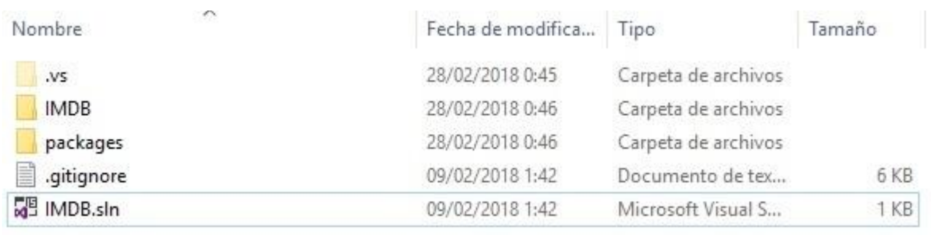
The C# project will automatically resolve its **NuGet dependencies** (described in packages.config) using the NuGet package restore when the project is built.

C#

- **.NET** framework (**.NET MVC** + **Entity Framework**)
- **Razor** view engine
- **Entity Framework** ORM
- **MS SQL Server** database

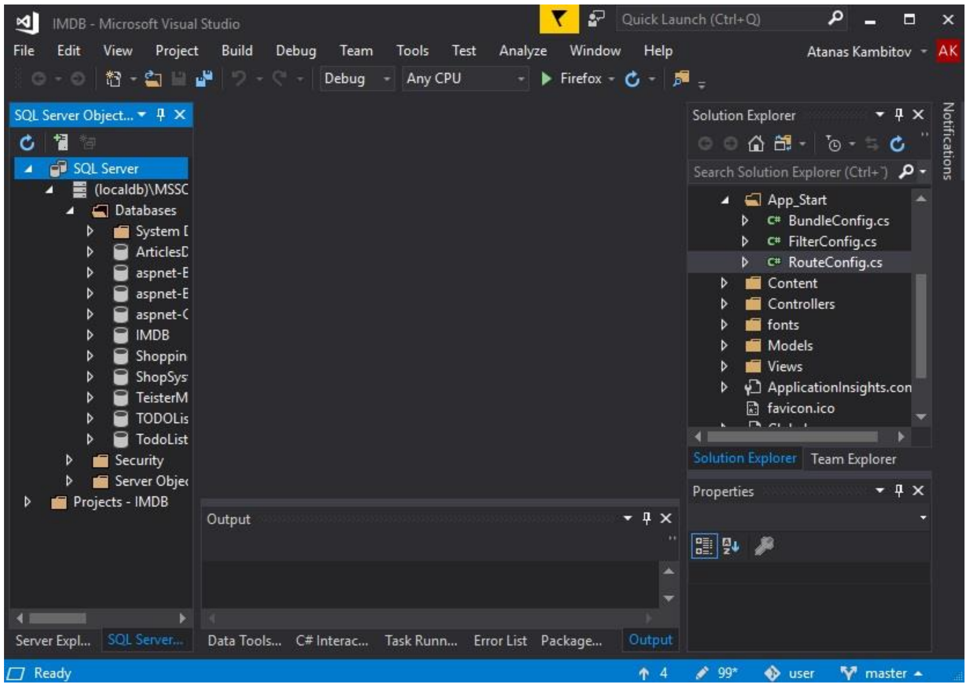
How to open and run the project?

01.Just unzip and open the solution file in the folder with Visual Studio 2017.



Nombre	Fecha de modifica...	Tipo	Tamaño
.vs	28/02/2018 0:45	Carpeta de archivos	
IMDB	28/02/2018 0:46	Carpeta de archivos	
packages	28/02/2018 0:46	Carpeta de archivos	
.gitignore	09/02/2018 1:42	Documento de tex...	6 KB
IMDB.sln	09/02/2018 1:42	Microsoft Visual S...	1 KB

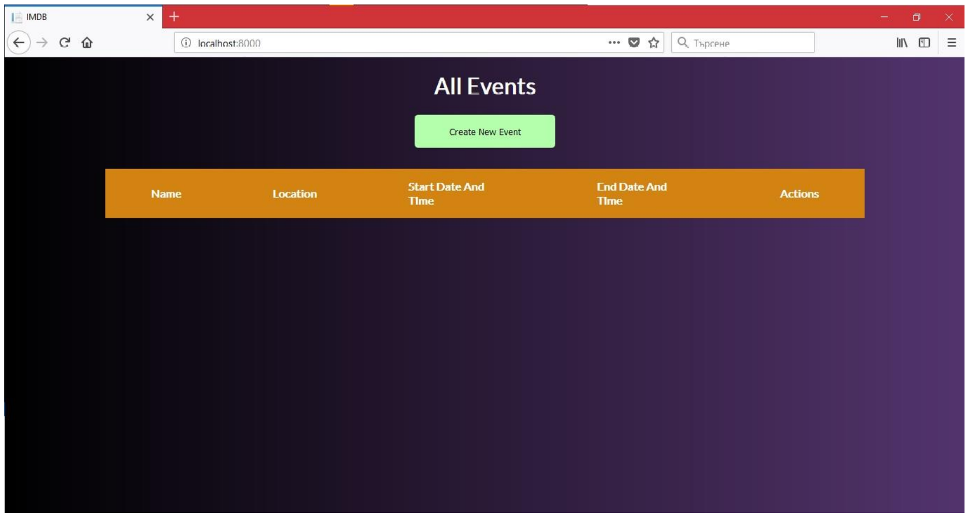
02.After the solution file is opened juts run it by clicking the green play button or press CTRL +F5 or just F5.



03. The default browser is Firefox, If you see the following website that means you have successfully opened the project and you are ready to test it.

Index Page:

Route: "/"



How does it work?

At this point you are at the index page which allows you to read all the events, unfortunately

you don’t have any events made yet so let's create one.

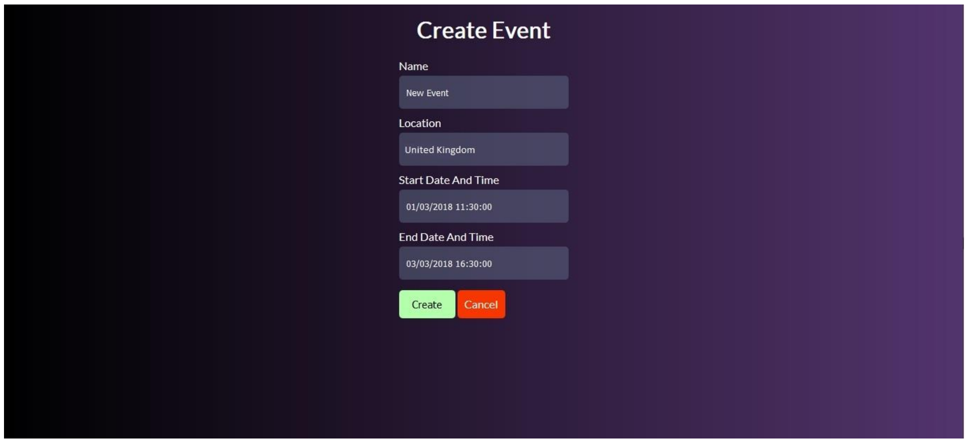
Creating Events:

01.Click the '**Create New Event**' button to create events.

02.You arrived at the create page, fill the form and press the 'Create' button or click cancel and go back to the index page.

Create Page:

Route: “/create”



Create Event

Name: New Event

Location: United Kingdom

Start Date And Time: 01/03/2018 11:30:00

End Date And Time: 03/03/2018 16:30:00

Keep in mind that to create a new event all the fields must be filled because they are required.

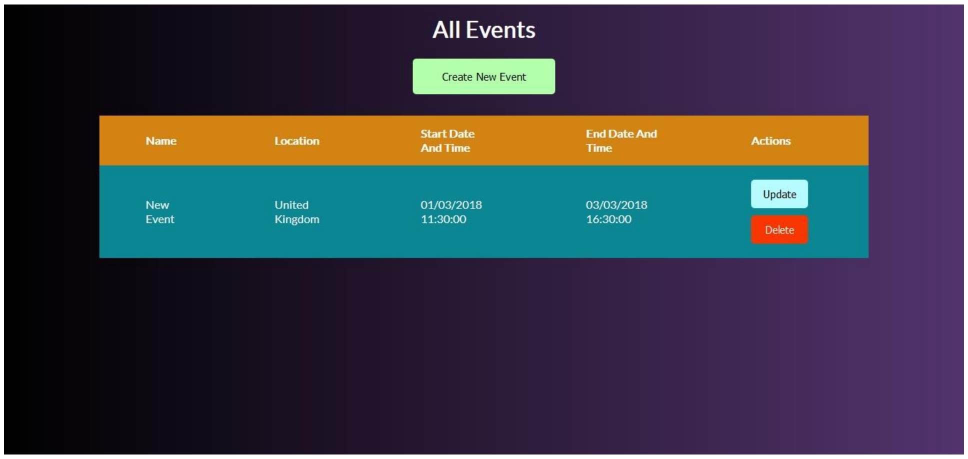
All the events are stored in a **Database**, we will look at it later.

All the fields are valid, the '**Start Date And Time'** and the '**End Date And Time'** fields expect

a date and time format but you can test them to see that there is a validation and it does not give you errors, instead it just lets you try again.

The application allows you to list many events, not just one.

After you have successfully created an event the App redirect you to the index page but this time you can clearly see your event.


All Events

Name	Location	Start Date And Time	End Date And Time	Actions
New Event	United Kingdom	01/03/2018 11:30:00	03/03/2018 16:30:00	Update Delete

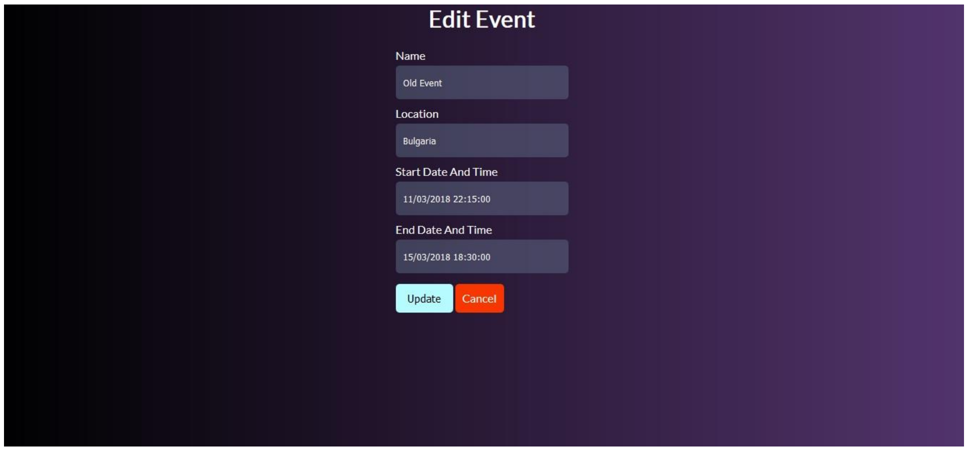
On the right you can see the option to **update** or **delete** an event.

Updating Events:

01.To update and event click the '**Update** 'button, you will see the following page:

Edit Page:

Route: “/edit/{id}”


Edit Event

Name: Old Event

Location: Bulgaria

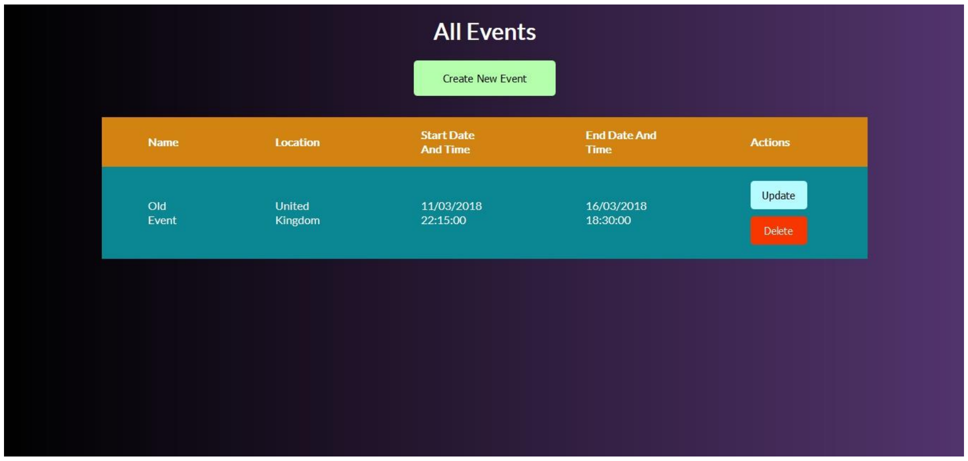
Start Date And Time: 11/03/2018 22:15:00

End Date And Time: 15/03/2018 18:30:00

02.Just implement the changes you want and click the '**Update**' button.

It redirects you to the index page where you can see the changes on the event.

For this example, we have made changes on all the fields.



Name	Location	Start Date And Time	End Date And Time	Actions
Old Event	United Kingdom	11/03/2018 22:15:00	16/03/2018 18:30:00	Update Delete

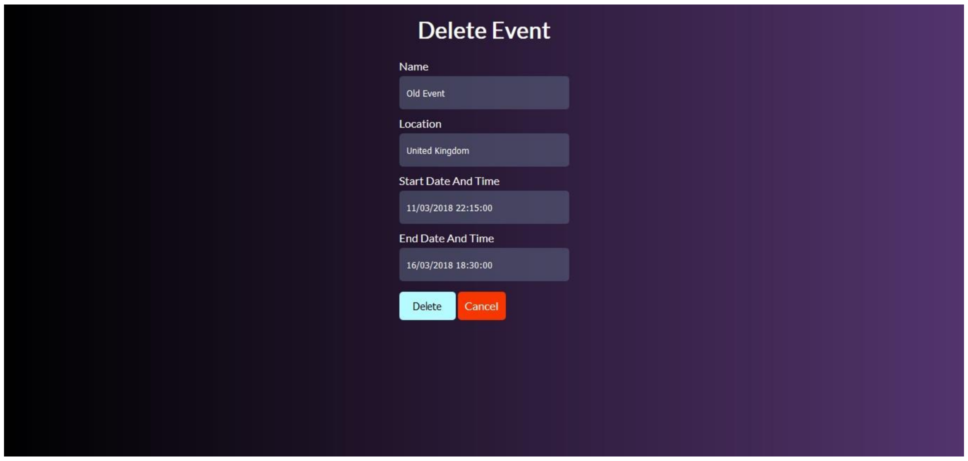
Deleting Events:

Deleting events works just the same way, just click on the delete button.

You will get to the delete page:

Delete Page:

Route: “/delete/{id}”

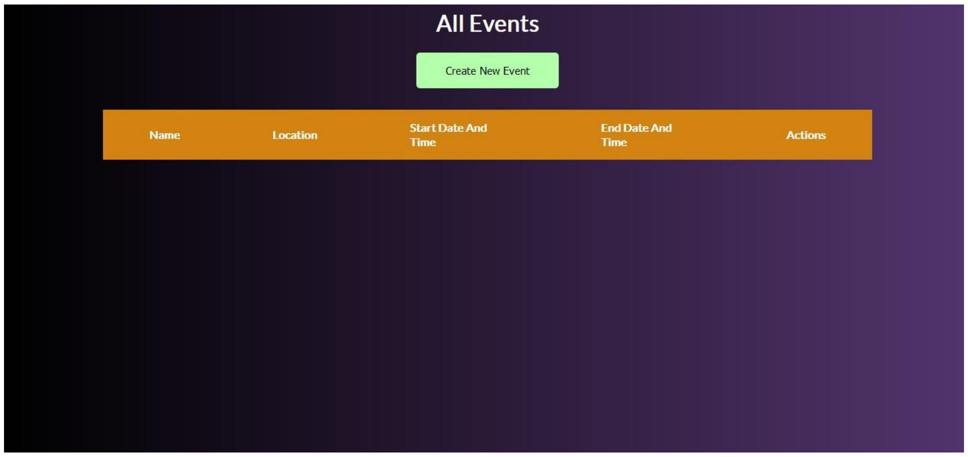


Name: Old Event
Location: United Kingdom
Start Date And Time: 11/03/2018 22:15:00
End Date And Time: 16/03/2018 18:30:00

In here you cannot change any fields because in the delete view they are set to **disabled**.

02.Click the '**Delete**' button again, it will redirect you to the index page and the event will disappear from the **Database**.

Since we had only one event added, now that we removed it the index page is empty, and it looks like in the beginning.



Name	Location	Start Date And Time	End Date And Time	Actions

The Code And The Project:

Now that we know how to use the application, lets' see how the **code** works.

Let's look at the **Folders** in our project.

This is an **MVC (Model View Controller)** project so the main thing to understand here is the functionality of the following:

01.Configurations

02.Controllers

03.Models

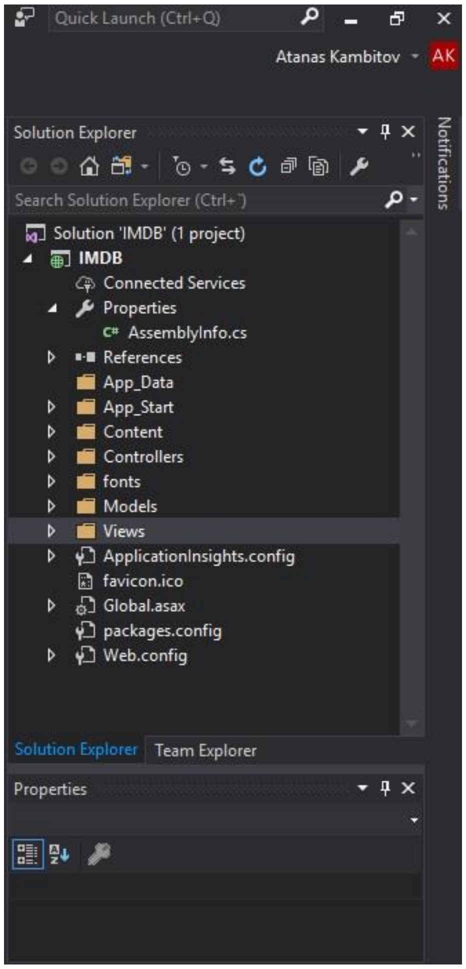
04.Views

05.The Database

Going back to **Visual Studio 2017**, on the right we can see the '**Solution Explorer**'.

If you cannot see it go to '**View**' and click on '**Solution Explorer**'.

It looks like this:



```
Solution 'IMDB' (1 project)
◢ IMDB
    Connected Services
  ◢ Properties
      AssemblyInfo.cs
  ▷ References
    App_Data
  ▷ App_Start
  ▷ Content
  ▷ Controllers
  ▷ fonts
  ▷ Models
  ▷ Views
  ▷ ApplicationInsights.config
    favicon.ico
  ▷ Global.asax
    packages.config
  ▷ Web.config
```

You can check the **AssemblyInfo.cs** file there is information insade.

The **App_Data** folder is empty.

When creating a new **.NET** application in Visual Studio, a couple of files and folders are created automatically and one of those folders is called **App_Data**.

App_Data folder can contain application data files like **LocalDB, .mdf files, xml files** and **other** data related files.

The **App_Start** Folder:

App_Start folder can contain class files which will be executed when the application starts. Typically, these would be config files like **AuthConfig.cs, BundleConfig.cs, FilterConfig.cs,**

RouteConfig.cs etc. MVC 5 includes **BundleConfig.cs, FilterConfig.cs** and **RouteConfig.cs** by default.

It contains three configuration files for this Application:

01.BundleConfig.cs

Bundling is a new feature in **.NET 4.5** that makes it easy to combine or bundle multiple files into a single file.

You can create CSS, JavaScript and other bundles.

Fewer files means fewer **HTTP requests** and that can improve first page load performance.

02.FilterConfig.cs

The FilterConfig.cs is a custom class in our code.

You can think of it as a little box that sits between the user's browser and our controller and filters out any invalid requests, or one that executes when a controller is done, and you need to postprocess the result to the user.

03.RouteConfig.cs

Routing enables you to use URLs that do not have to map to specific files in a Web site.

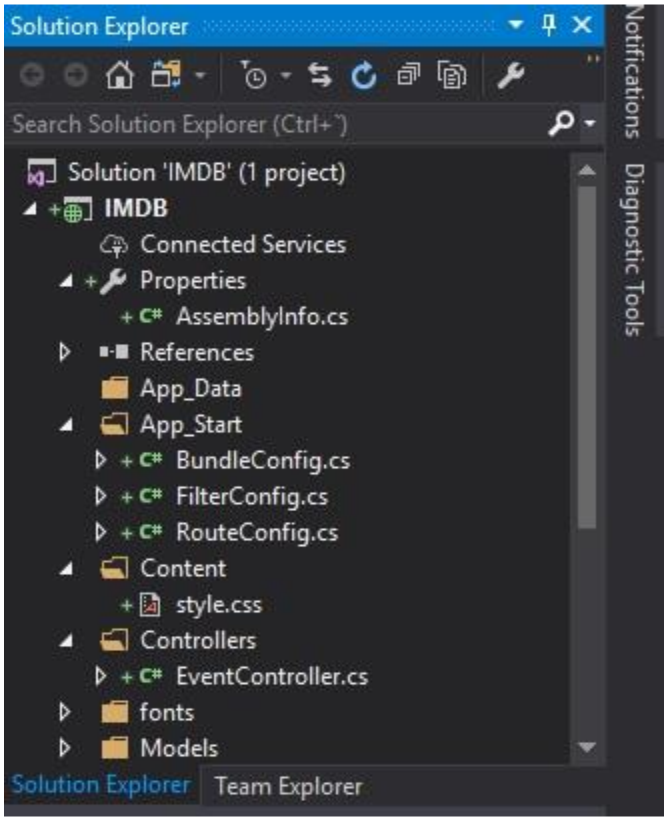
Because the URL does not have to map to a file, you can use URLs that are descriptive of the user's action and therefore are more easily understood by users.

In .NET routing, you can define URL patterns that map to request-handler files, but that do not necessarily include the names of those files in the URL.

In addition, you can include placeholders in a URL pattern so that variable data can be passed to the request handler without requiring a query string.

You can find more information about these configurations and how exactly they work on the internet.

When you open some of the first folders the project looks like in the following image:



The **Content** folder:

Content folder contains static files like css files, images and icons files.

MVC 5 application includes bootstrap.css, bootstrap.min.css and Site.css by default.

In our case it contains one **CSS** file which is used for the styling of the application.

It's not very big you can check it out.

The **Controller** Folder:

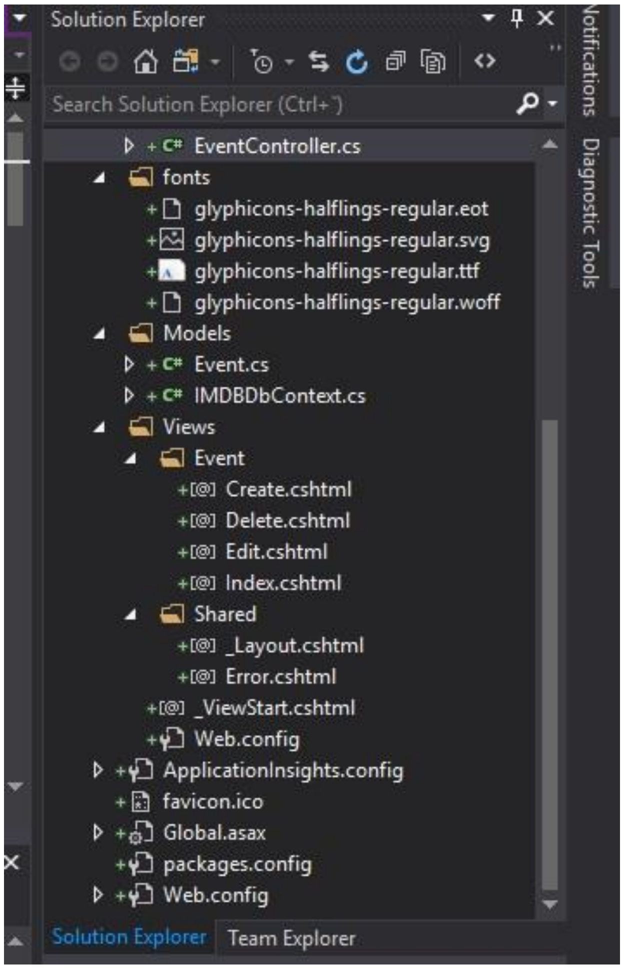
It holds all our controllers, in this case we have only one which is the **EventController.cs**

Controllers process incoming requests, handle user input and interactions, and execute appropriate application logic.

A controller class typically calls a separate view component to generate the HTML markup for the request.

ALL THE EXPLANATION ON HOW IT WORKS IS INSADE OPEN THE EventController.cs FILE .

Further on the project looks like this:



```
    EventController.cs
  fonts
    glyphicons-halflings-regular.eot
    glyphicons-halflings-regular.svg
    glyphicons-halflings-regular.ttf
    glyphicons-halflings-regular.woff
  Models
    Event.cs
    IMDBDbContext.cs
  Views
    Event
      Create.cshtml
      Delete.cshtml
      Edit.cshtml
      Index.cshtml
    Shared
      _Layout.cshtml
      Error.cshtml
    _ViewStart.cshtml
    Web.config
  ApplicationInsights.config
  favicon.ico
  Global.asax
  packages.config
  Web.config
```

Fonts folder contains custom font files for your application.

The **Models** Folder:

It Contains all the models that we use, in this case we have two models.

Event Class Model:

It is basically a template for every event, in other words it says what properties should every event have.

ALL THE EXPLANATION IS INSADE OPEN THE Event.cs FILE .

The other file is very important class because is basically our connection to the database.

We use it to manipulate and perform operations with the database.

The **Views** folder:

Views folder contains html files for the application.

Typically view file is a **.cshtml** file where you write **html** and **C#** or **VB.NET** code.

Views folder includes separate folder for each controller.

For example, all the **.cshtml** files, which will be rendered by the **EventController** will be in View > **Event folder**.

For this project I have four views.

01.Create

02.Delete

03.Edit

04.Index

Shared folder under View folder contains all the views which will be s**hared among different controllers** e.g. layout files.

In this case they are **_Layout** and **Error**

Additionally, MVC project also includes following configuration files:

Global.asax:
Allows you to write code that runs in response to application level events, such as Application_BeginRequest, application_start, application_error, session_start, session_end etc.

Packages.config:
Packages.config file is managed by NuGet to keep track of what packages and versions are have installed in the application.

Web.config:
Web.config file contains application level configurations.

The Database:

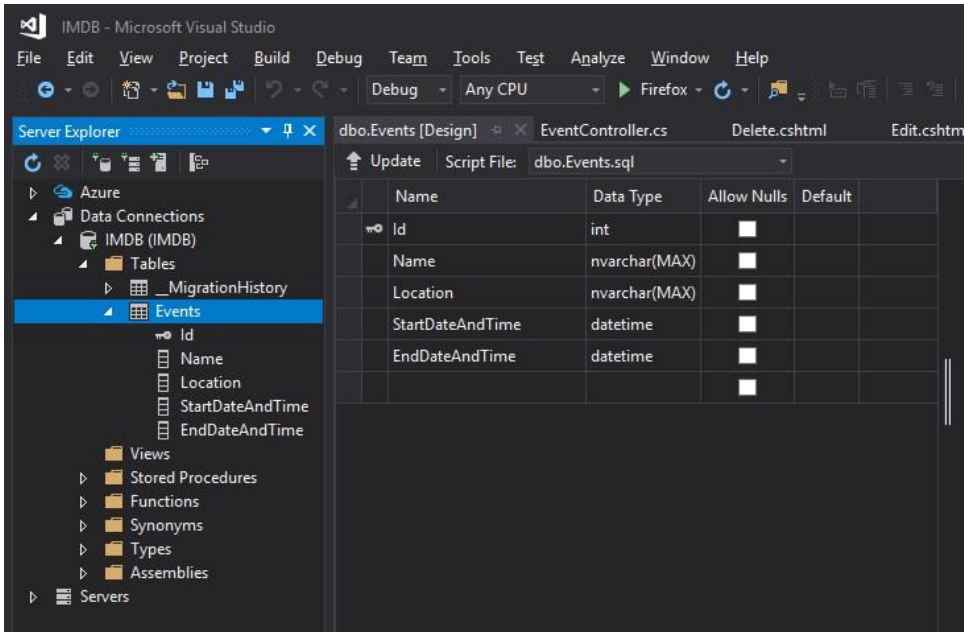
For this Application the Database is very simple, it contains one table **Events** because it works only with events.

To see the **Database**, we can use **MSSQL Server Management 2017** or the

Server Explorer in Visual Studio 2017.

If you don't see it on the left, go to view and click on Server Explorer.

To see the **Table Data Definition:** right click on the table and click on **'Open Table Definition'.**



Name	Data Type	Allow Nulls	Default
Id	int	☐	
Name	nvarchar(MAX)	☐	
Location	nvarchar(MAX)	☐	
StartDateAndTime	datetime	☐	
EndDateAndTime	datetime	☐	
		☐	

The table is like the Event Model and the form you filled earlier, it contains :

01.Id - used to keep track ot all the Events,

02.Name of the Event,

03.Location,

04.Start Date And Time

05.End Date And Time

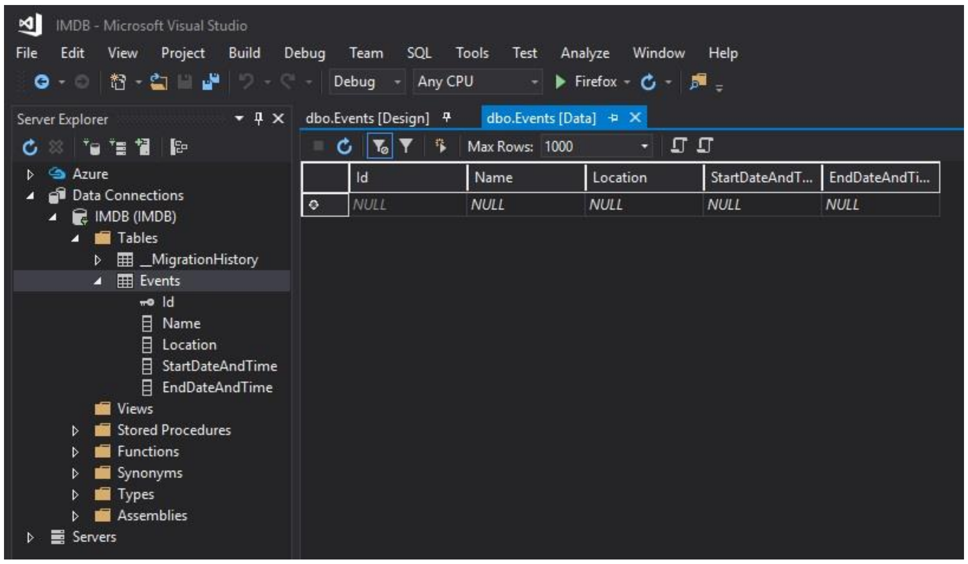
On the right you can see the **Data Types** of the columns and if they are **nulls**, this means that it the column cannot be **null** and it must be filled, it is **required**.

Int – rapresents a number, we need it for the Id of the event.

Nvarchar – represents a string or a text but it can contain UNICODE characters as well.

Datetime – is clear that is for dates

To see the **Table Data**: Right click on the Events table and click on **'Show Table Data'**.

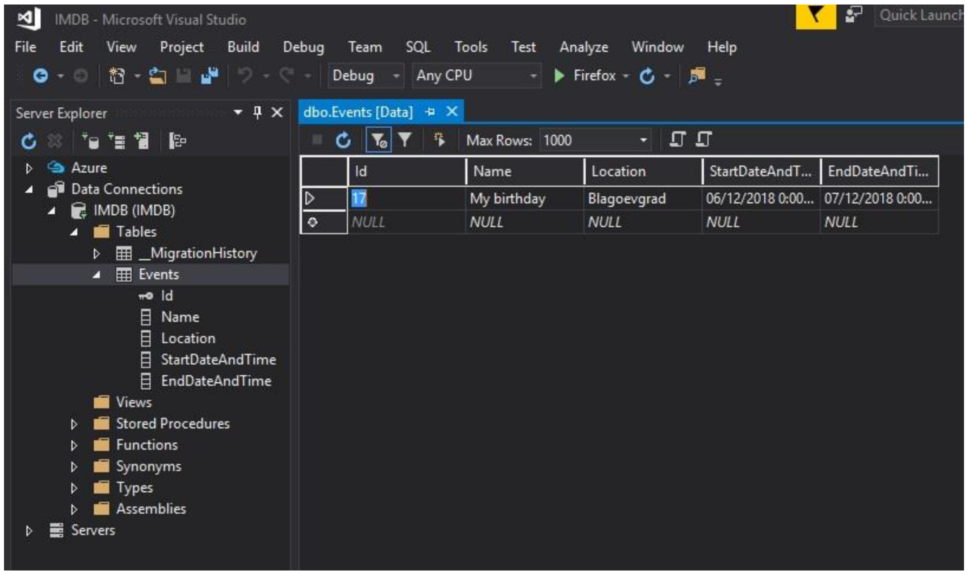


	Id	Name	Location	StartDateAndT...	EndDateAndTi...
	NULL	NULL	NULL	NULL	NULL

In here we can see all the information of the events registered in out database.

Right now is empty because there are no events.

To see that everything works correctly, I will register an event just to see its data in here.



	Id	Name	Location	StartDateAndT...	EndDateAndTi...
▷	17	My birthday	Blagoevgrad	06/12/2018 0:00...	07/12/2018 0:00...
	NULL	NULL	NULL	NULL	NULL