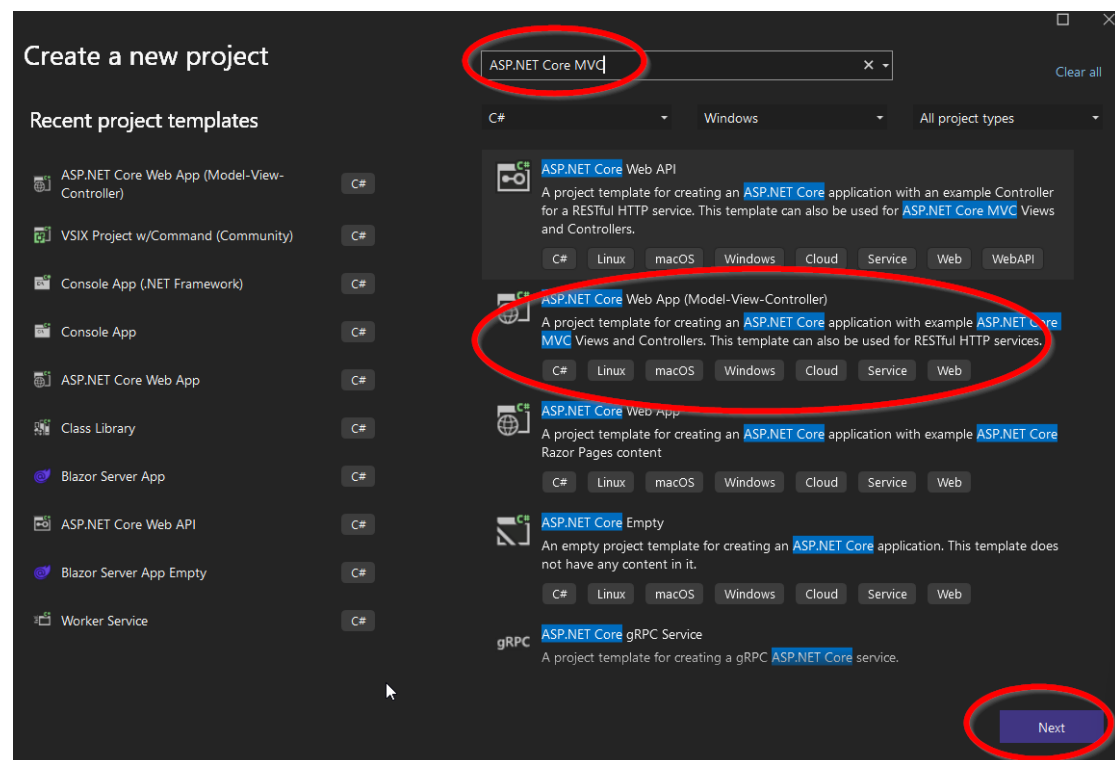# iVideos

In this document, I will be explaining step by step to create ASP.NET Core project with two views: first to upload video and save it in the server-local-disk and the other view is loading the video to the view and playing it.

New Project

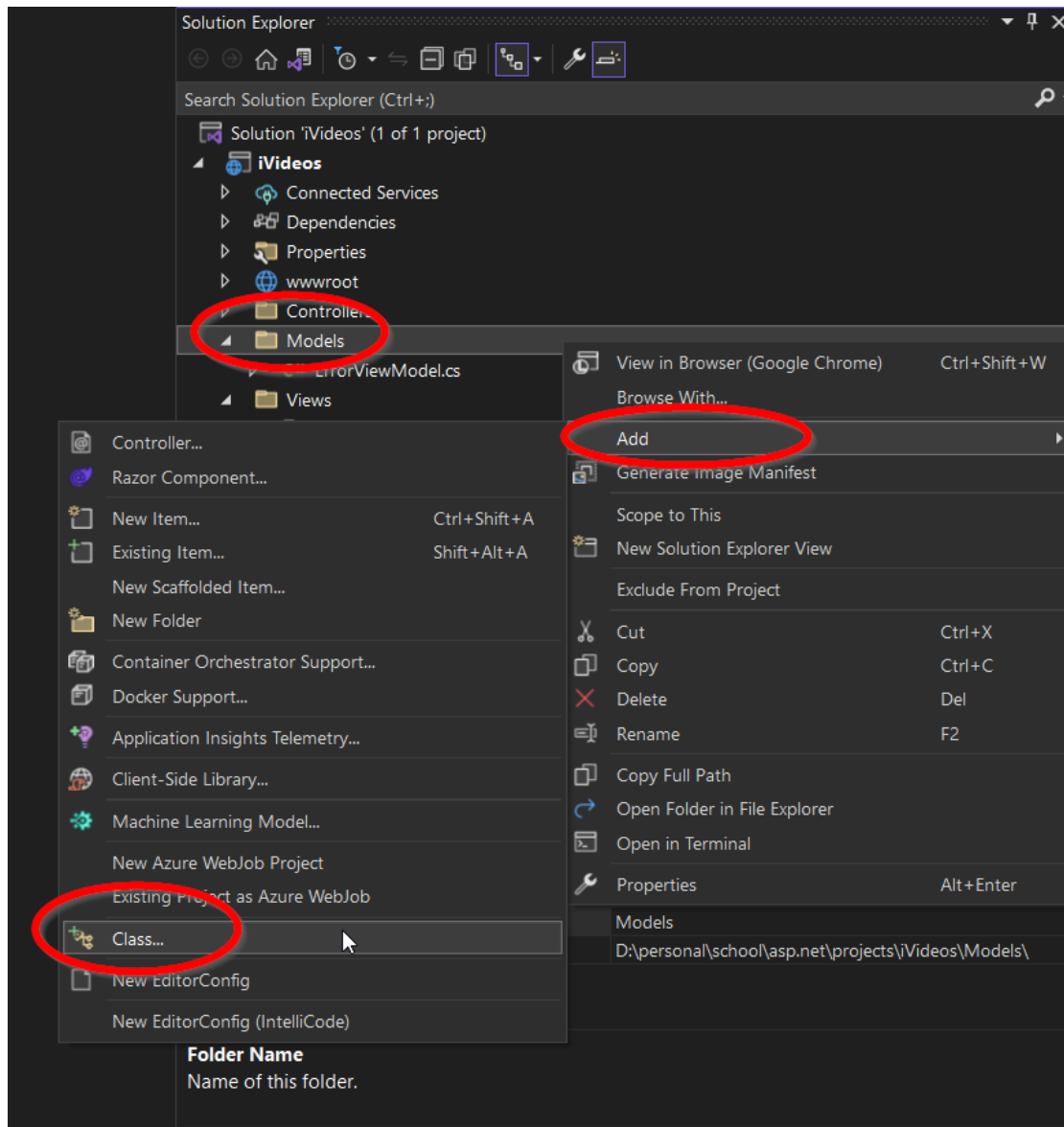Create new ASP.NET Core MVC project in VS 2022

Next & Next

Model Classes Preparation

We are going to mimic DB, but not really (if you want to save the video information in DB then you need to connect the classes using Entity Framework – you can refer to other project I have created in same repo)

Video Models

Right click on Models folder:

name the new class: Video.cs

```
public class Video
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public string Path { get; set; }
        public int OwnerId { get; set; }
    }
```

Create another class in same way, call it: VideoViewModel

VideoViewModel is meant to be used for the View (cshtml) so the user will fill the information.

the end user, will not provide the video path but the video file itself. so we need different class

```
public class VideoViewModel
    {
        [Required]
        public string Title { get; set; }
        [DataType(DataType.MultilineText)]
        public string Description { get; set; }
        [Required]
        public string Subject { get; set; }
        [Required]
        [DataType(DataType.Upload)]
        public IFormFile Video { get; set; }
    }
```

Dummy DB

We wont deal with DB, we will create a singleton class that save/load data from memory…i.e. next run everything will be empty…We will call it: *VideosDB*
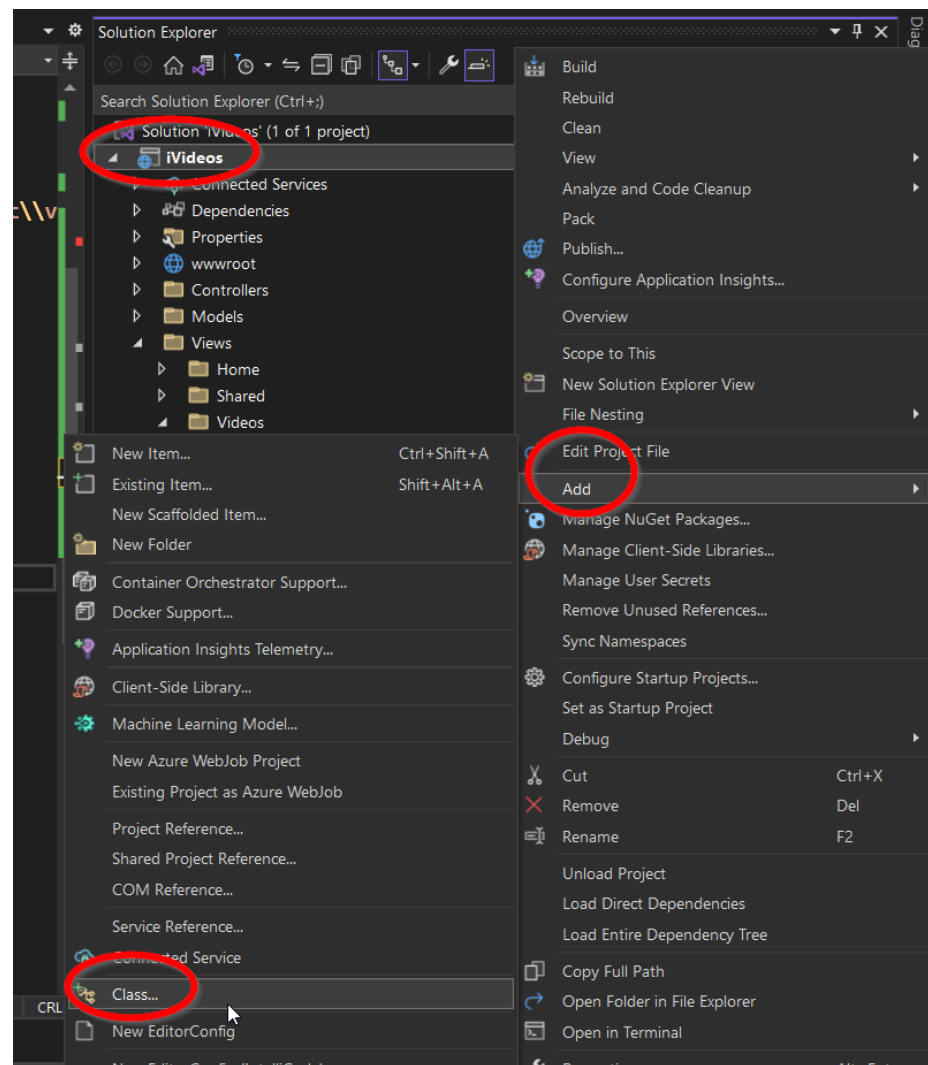
copy the implementation to your project

```csharp
public class VideosDB
{
    private static VideosDB the_;
    public List<Video> Videos { get; private set; }
    private VideosDB()
    {
        Videos = new List<Video>();
    }
    public static VideosDB Instance
    {
        get
        {
            the_ ??= new VideosDB();
            return the_;
        }
    }
    public void Add(Video v)
    {
        Videos.Add(v);
    }
    public void Remove(Video v)
    {
        Videos.Remove(v);
    }
    public void Clear()
    {
        Videos.Clear();
    }
}
```

Controller

Create new Controller class under *Controllers* folder, name the class: *VideosController.cs*
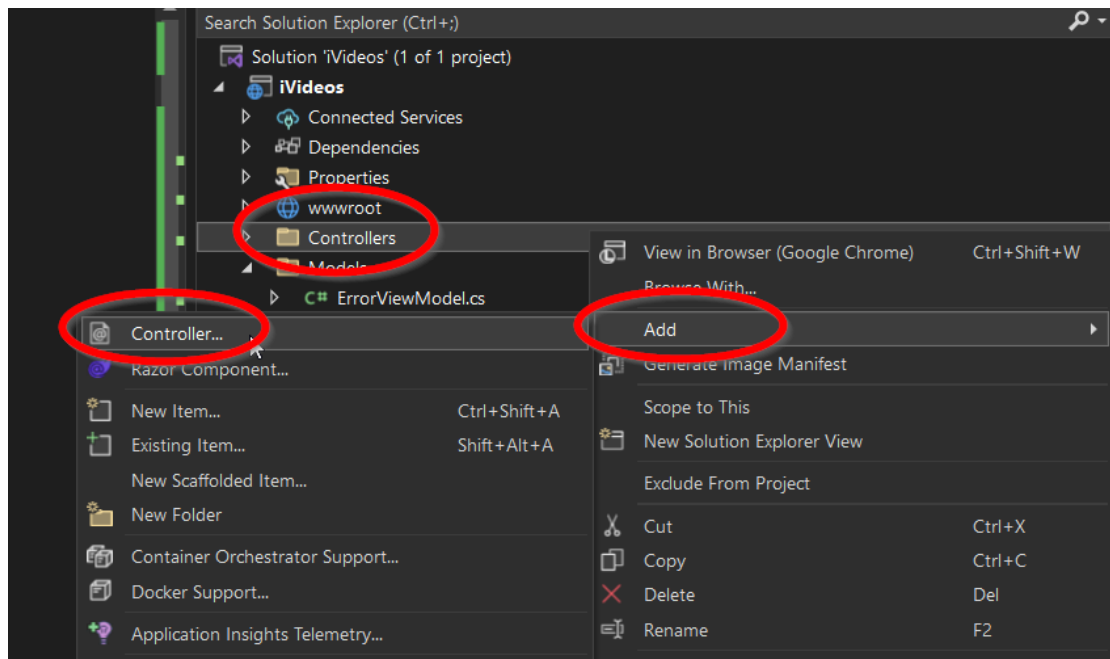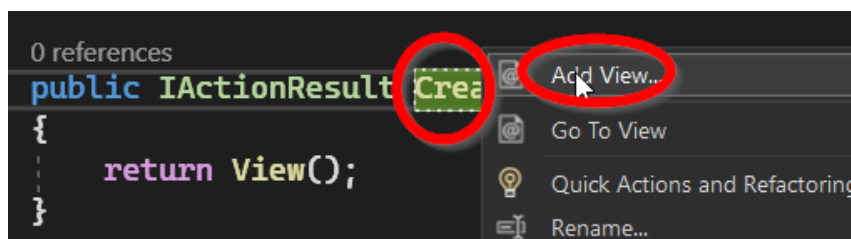

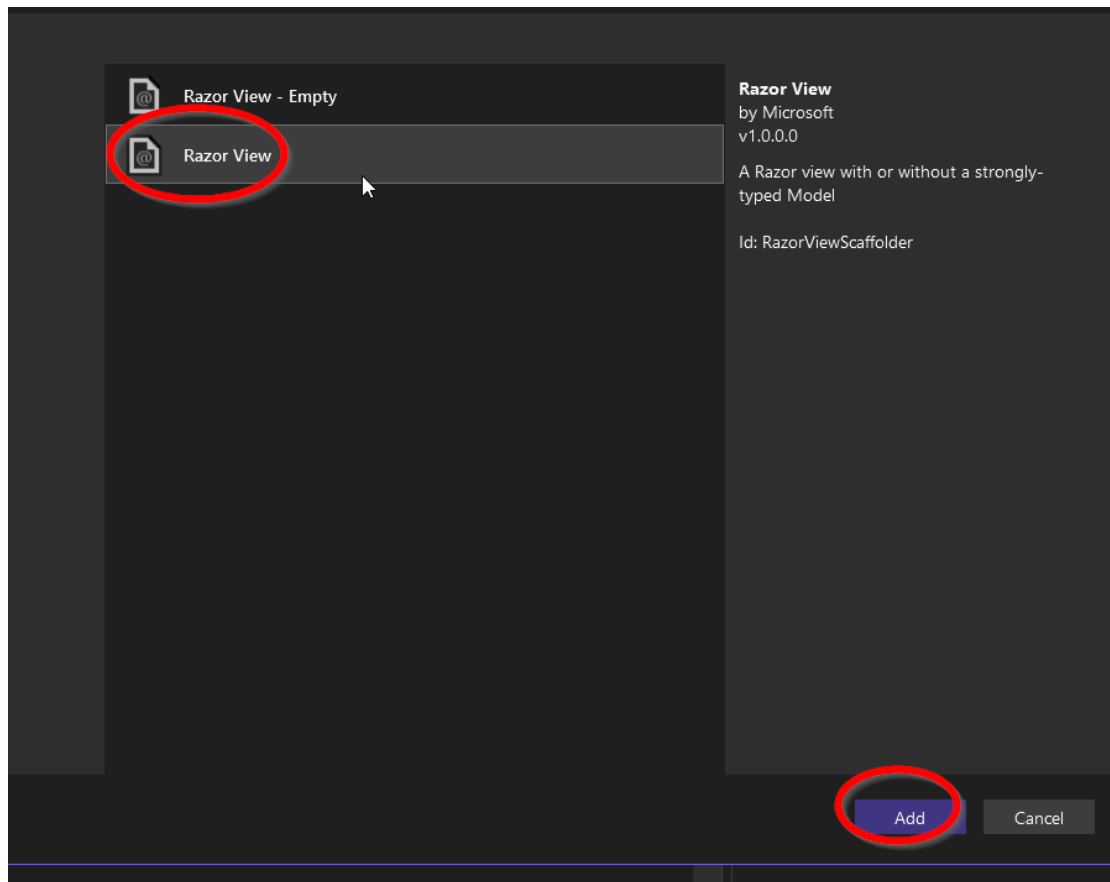Then select "MVC Controller – Empty" from the list

Create new method in the class:

```
public IActionResult Create()
        {
            return View();
        }
```

Right click on the method name -> Add View…

Then

Then

View

review the Create.cshtml

Make sure it has an entry for Video..The file should looks like:

```
@model iVideos.Models.VideoViewModel

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>VideoViewModel</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create" enctype="multipart/form-data" >
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <textarea asp-for="Description" class="form-control"></textarea>
                <span asp-validation-for="Description" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Subject" class="control-label"></label>
```
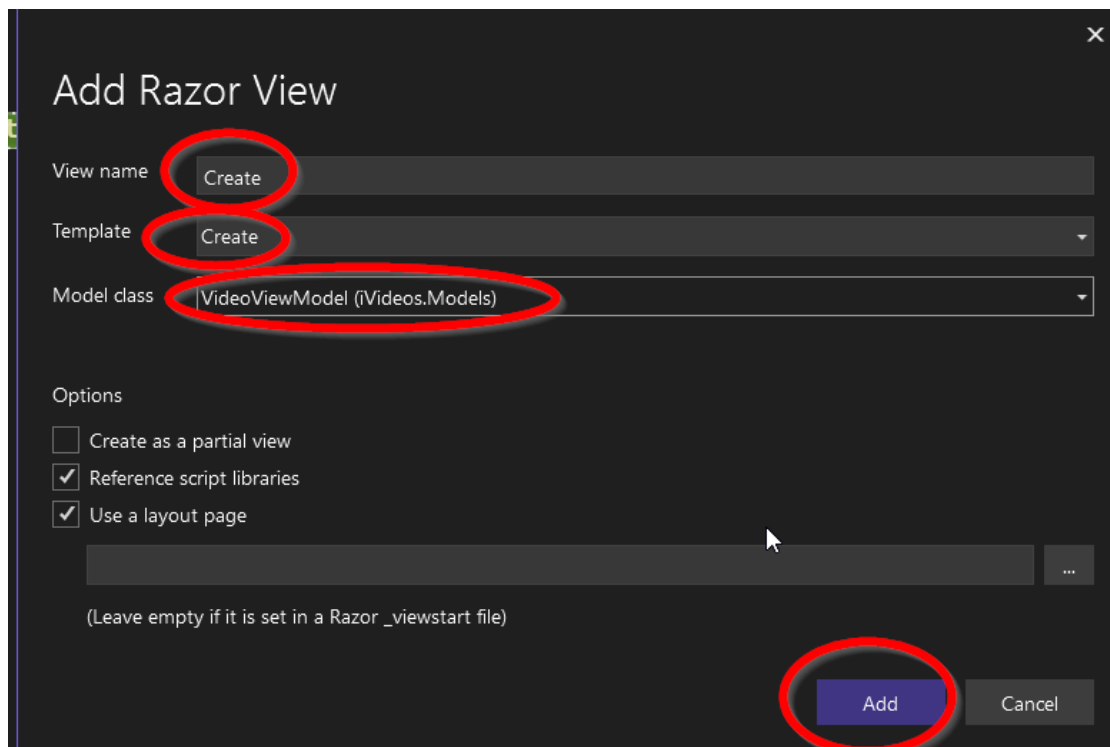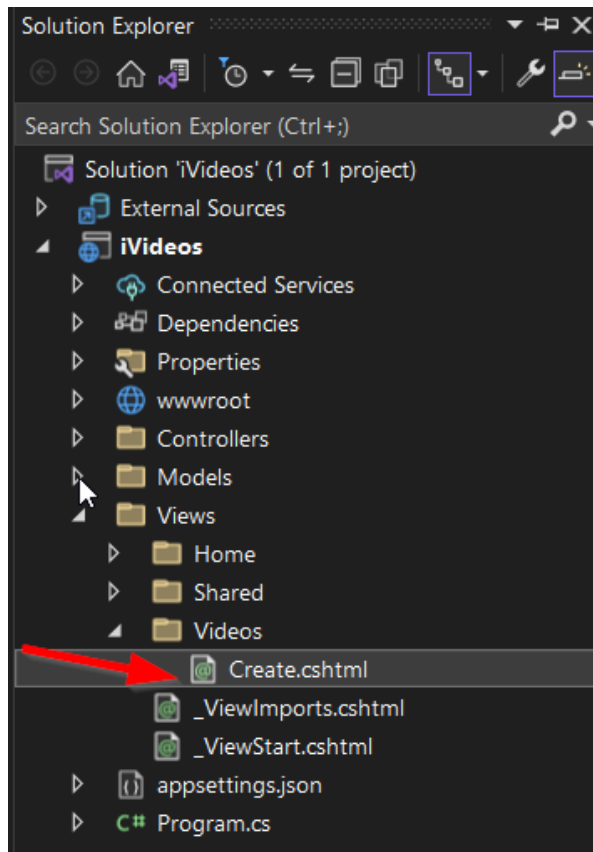
```
            <input asp-for="Subject" class="form-control" />
            <span asp-validation-for="Subject" class="text-
danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="Video" class="control-label"></label>
            <input asp-for="Video" class="form-control" />
            <span asp-validation-for="Video" class="text-
danger"></span>
        </div>
        <div class="form-group">
            <input type="submit" value="Create" class="btn btn-
primary" />
        </div>
    </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Review the app so far

run the app and change the URL to – port, in yellow may change from run to run…

https://localhost:7254/Videos/Create

As you may notice, click on Create does nothing, this because we didn't implement a Post method for create in the VideosController. Let's make it

In the <mark>VideosController</mark>, add new method as following:

```
[HttpPost]
        public IActionResult Create(VideoViewModel model)
        {
            return View();
        }
```

Now we will start fill the implementation

```
[HttpPost]
[DisableRequestSizeLimit] //Allow large files to be uploaded
        public IActionResult Create(VideoViewModel model)
        {
            //check if model is valid
            if (ModelState.IsValid)
            {
                //save video under wwwroor/videos
                string fname =
Path.Combine("videos",model.Video.FileName);
                string fullname =
Path.Combine(Environment.CurrentDirectory,"wwwroot", fname);
                MemoryStream stream = new MemoryStream();
                model.Video.CopyTo(stream);
                System.IO.File.WriteAllBytes(fullname, stream.ToArray());

                //save data into "dummy db"
                Video v = new Video();
                v.Subject = model.Subject;
                v.Title = model.Title;
                v.Description = model.Description;
                v.Id = Random.Shared.Next();
                v.Path = fname;

                VideosDB.Instance.Add(v);

                return RedirectToAction("Index");
            }
            else
            {
                return View();
            }
        }
```

Save and run the app again, click on create and notice that new file was created under the <mark>project\wwwroot\videos</mark>

Thumbnail

To save thumbnail as well of the video so you can serve these thumbnails to the client when it asks for list of videos follow the steps below:

1- From Nuget install MetdiaToolkit

2-Add new class, name it: *VideosUtils*

```csharp
using MediaToolkit.Model;
using MediaToolkit.Options;
using MediaToolkit;
public class VideosUtils
    {
        public static void SaveVideoThumbnail(string fname)
        {
            var video = new MediaFile
            {
                Filename = fname
            };
            var image = new MediaFile { Filename = fname + ".jpg" };

            using (var engine = new Engine())
            {
                engine.GetMetadata(video);
                engine.GetThumbnail(video, image,
                    new ConversionOptions { Seek =
TimeSpan.FromSeconds(video.Metadata.Duration.TotalSeconds / 2) });
            }
            //var thumbnail = File.ReadAllBytes(image.Filename);
            //File.Delete(video.Filename);
            //File.Delete(image.Filename);
            //return thumbnail;
        }
    }
```

3– Back to the *Videoscontroller,* update the method:

```csharp
public IActionResult Create(VideoViewModel model)
```

By adding the following line:

```csharp
VideosUtils.SaveVideoThumbnail(fullname);
```

Just right after:

```csharp
MemoryStream stream = new MemoryStream();
model.Video.CopyTo(stream);
System.IO.File.WriteAllBytes(fullname, stream.ToArray());
```

4– run the app and upload new video file and see that video and image are saved under the *wwwroot\videos* folder
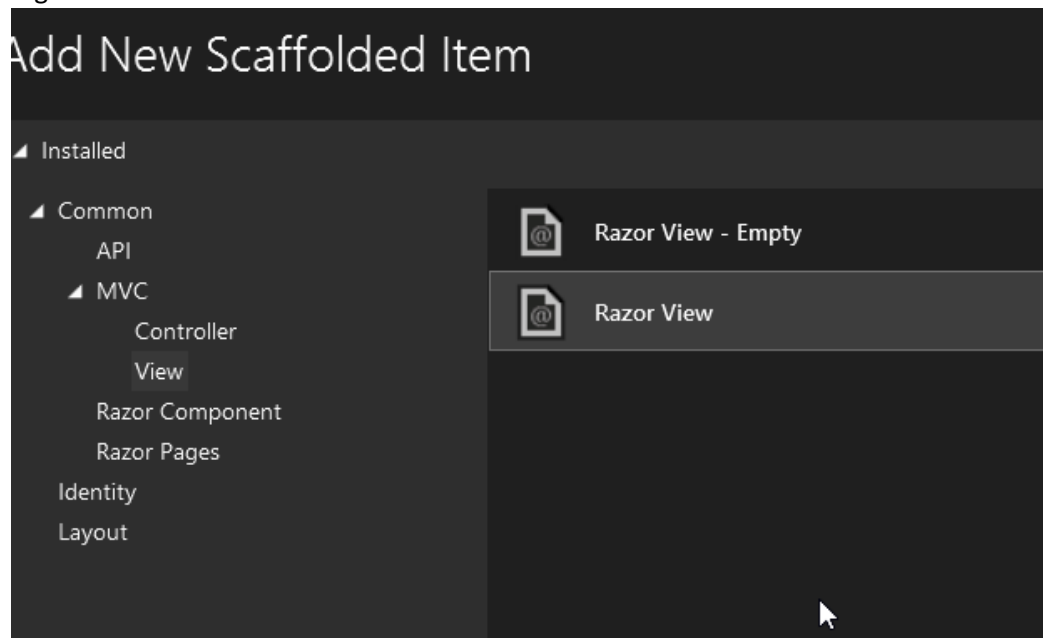
List Videos

1- Add new ViewModel for the Video list (it is also possible to utilize the existing one…)
We call it: VideoViewModel2

```csharp
public class VideoViewModel2
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public string Subject { get; set; }
        [DataType(DataType.ImageUrl)]
        public string ImagePath { get; set; }
    }
```
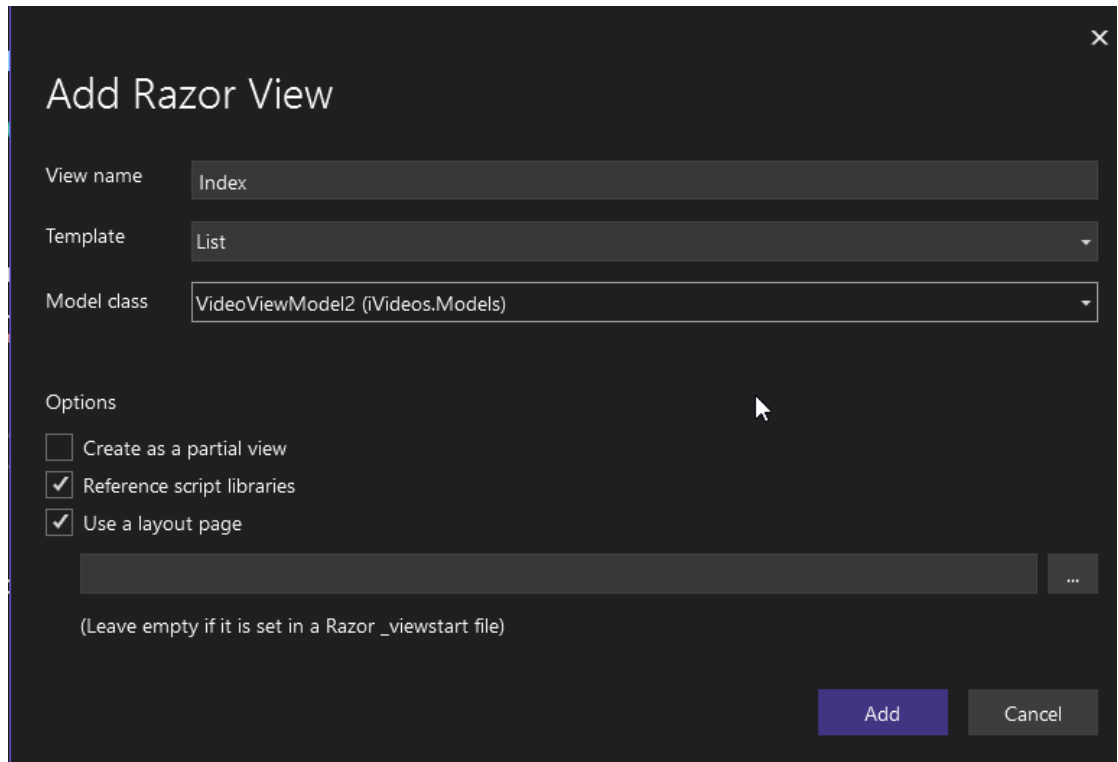
2- In the Video Controller add the following method:

```csharp
public IActionResult Index() {
        var viedoes = VideosDB.Instance.Videos;
        List<VideoViewModel2> vms = new List<VideoViewModel2>();
        foreach (var v in viedoes)
        {
            vms.Add(new VideoViewModel2()
            {
                Description = v.Description,
                Id = v.Id,
                Subject = v.Subject,
                Title = v.Title,
                ImagePath = @"videos\" + v.Path + ".jpg"
            });
        }
        return View(vms);
    }
```

3- Right click on the method name and add view…

4- Then

5- New view is created under: Views→Videos→Index.cshtml
here is the content (with modification)

```
@model IEnumerable<iVideos.Models.VideoViewModel2>

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Title)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Description)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Subject)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ImagePath)
            </th>
            <th></th>
        </tr>
    </thead>
```

```
<tbody>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Id)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Description)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Subject)
            </td>
            <td>
                <img src="@Url.Content(item.ImagePath)" alt="Image" width="100" height="100" />
                @*@Html.DisplayFor(modelItem => item.ImagePath)*@
            </td>
            <td>
                @Html.ActionLink("Go to Video", "GotoVideo", new { id=item.Id})
                @*@Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
                @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })*@
            </td>
        </tr>
    }
</tbody>
</table>
```

iVideos   Home   Privacy

Index

Create New

Id	Title	Description	Subject	ImagePath	
861315758	asdasd	asdasf sdf sdf	sdf		Go to Video

Visit Video Page

We are not going to focus on the style, only the functionality

in this page, we are going to display the video details and a video player to start/stop/pause

1- in the VideosController add new function as following:
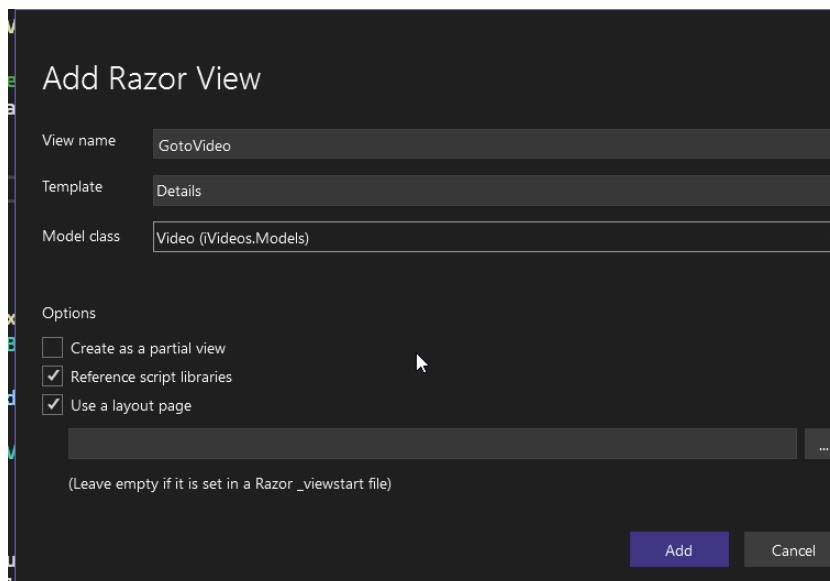
```
public IActionResult GotoVideo(int id)
    {
        //check if the video exists
        var v =
VideosDB.Instance.Videos.Where(u=>u.Id==id).FirstOrDefault();
        if(v!=null)
        {
            return View(v);
        }
        return View();
    }
```

2- Add View to the action
right click → Add View…
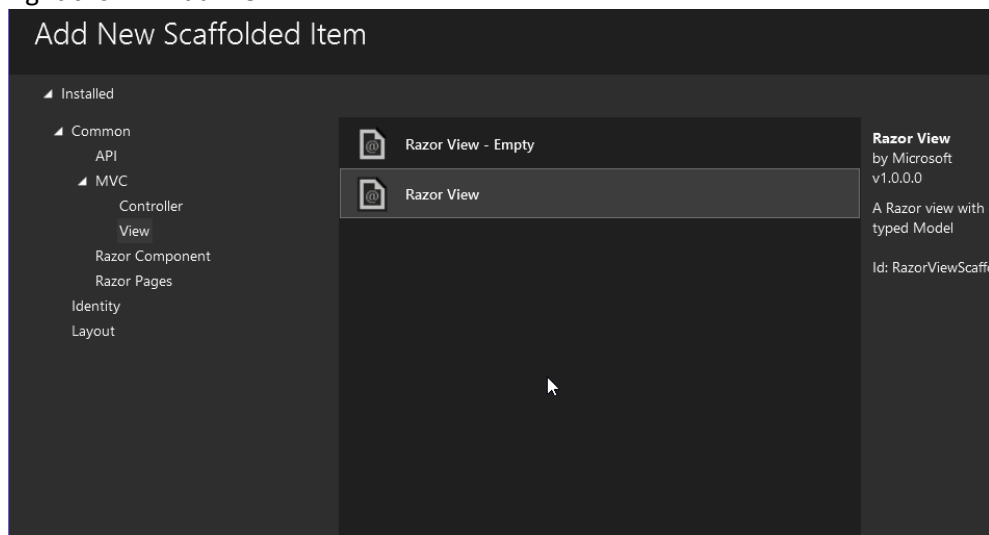
Then

a new cshtml is created – again, you may rewrite it as you wish to be more nice...
anyway, I am not going to use the default view, but this one:

```
@model iVideos.Models.Video

@{
    ViewData["Title"] = "GotoVideo";
}

<h1>@Model.Title</h1>

<div>
    <h4>Video – @Html.DisplayFor(model => model.Description)</h4>

    <hr />
    <dl class="row">

        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Title)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Title)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Description)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Description)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Subject)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Subject)
        </dd>

    </dl>
    <video id='vid' vid="@Model.Id" controls="controls" width="500"
height="400">
        <source src="~/@Model.Path" />
    </video>
</div>
```

Let JS video update C# back

In this section, we are going to let the JS to update the C# with the video current time to save the progress of the video

1- add @section scripts in the GotoVideo.cshtml at the end as following:

```
@section scripts
    {
    <script>
        let v = document.getElementById('vid')
        let currentTime = v.currentTime; //at this stage this is 0
        let duration = v.duration; //at this stage, this is undefined
        //update local variables
        v.ontimeupdate = (e) => {
            currentTime = Math.max(v.currentTime, currentTime);
            duration = v.duration;
        };
        //register for each 5 seconds to update server
        const updateInterval = setInterval(updateServer, 5000);
//update each 5 sec

        function updateServer() {
            //send data to the server! every 5 secs
            $.ajax({
                method: 'post',
                dataType: 'json',
                contentType: 'application/x-www-form-urlencoded',
                url: '/Videos/UpdateVidoeInfo',
                data: { vid: v.vid, duration: v.duration, currentTime:
currentTime },
                success: (d) => {
                    console.log(d);
                }
            });
        }

    </script>
}
```

2- Now add new method in the VideosController, name it: UpdateVidoeInfo'

```
[HttpPost]
        public void UpdateVidoeInfo(int vid, double duration, double
currentTime)
        {
            //save stuff to DB
            Console.WriteLine(vid);
        }
```

Final Code

You may find it under:

https://github.com/ahmad081177/asp.net.core/tree/main/iVideos