



Ingeniería de software

Laboratorio 3: Entendiendo MVC a profundidad

Resumen

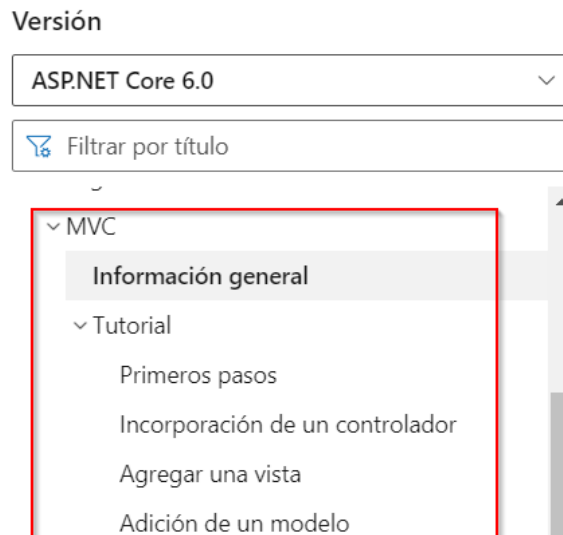
Existen muchos tipos de frameworks para el desarrollo web, en este curso nos vamos a enfocar en el uso de MVC utilizando ASP.NET Core.

Primera parte - Aprendiendo los conceptos básicos de MVC

El primer paso de este laboratorio es entender como funciona ASP.NET MVC Core, y el routing de una aplicación MVC. Para lograr esto se les recomienda revisar el tutorial de MVC de Microsoft:

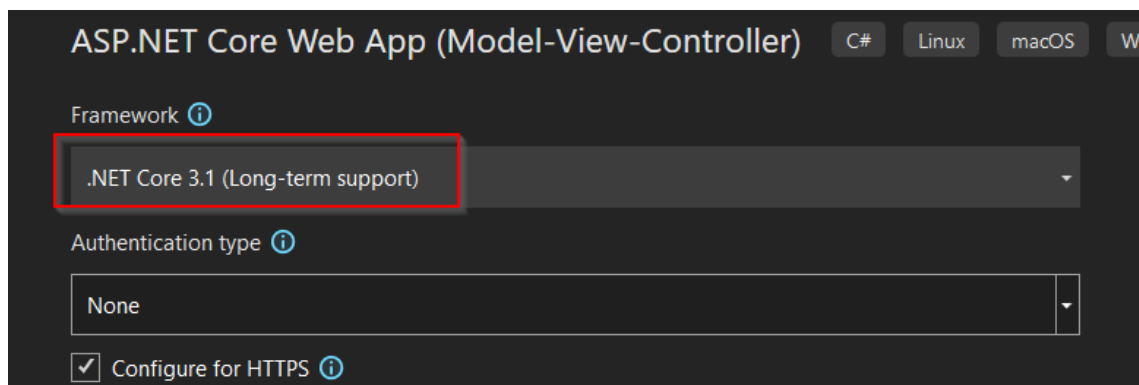
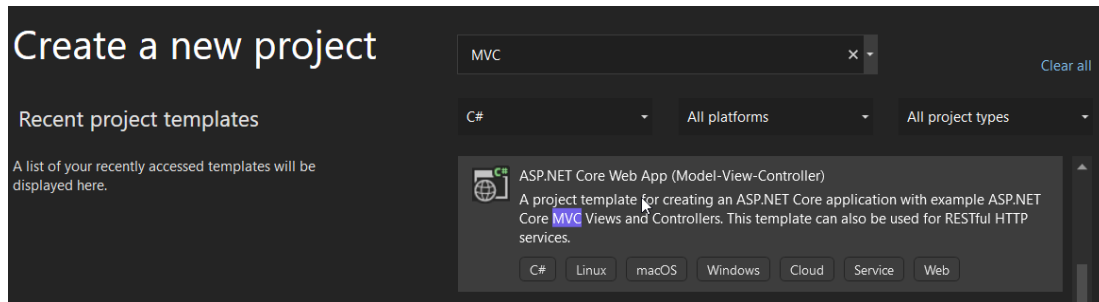
<https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-6.0>

Revise las primeras sesiones del tutorial:



Segunda parte - Creando mi primera pagina

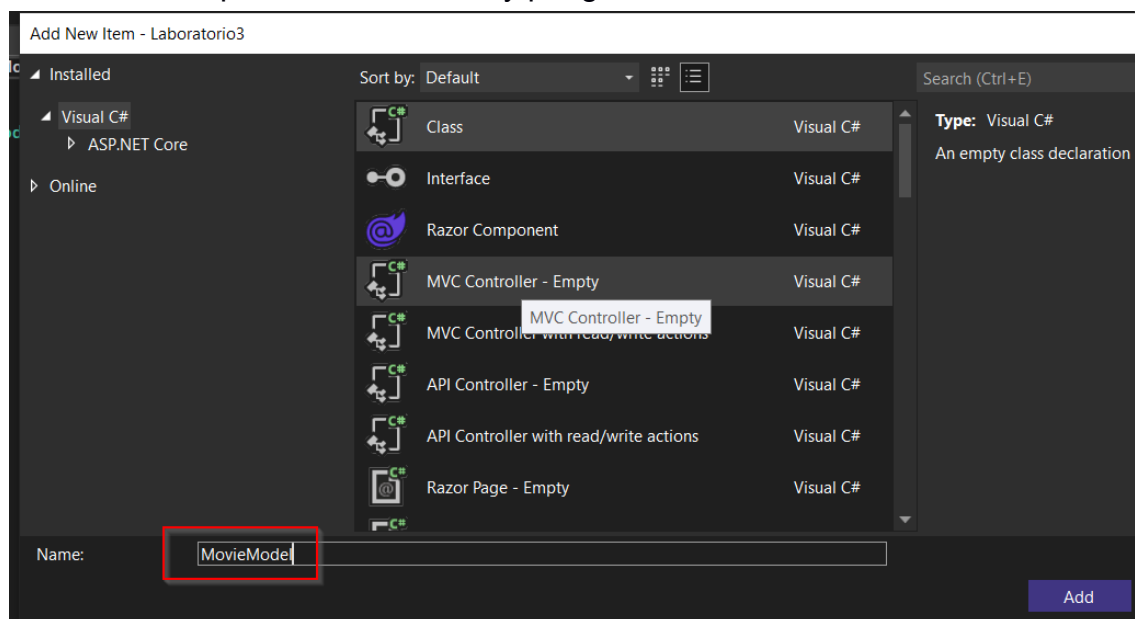
Primero, debe crear un nuevo proyecto del tipo ASP.NET MVC Core (este NO es el mismo tipo de proyecto utilizado en el lab 1) y nombre el proyecto como Laboratorio3, vincule este proyecto a nuevo repositorio en github.



Es hora de modificar el proyecto siguiendo los siguientes pasos:

1. Cree un modelo, para hacer esto dirijase al solution explorer y sobre la carpeta de *Models* y de click derecho y seleccione la opción *Add -> New item*.

Seleccione el tipo de archivo class y ponga el nombre *MovieModel*





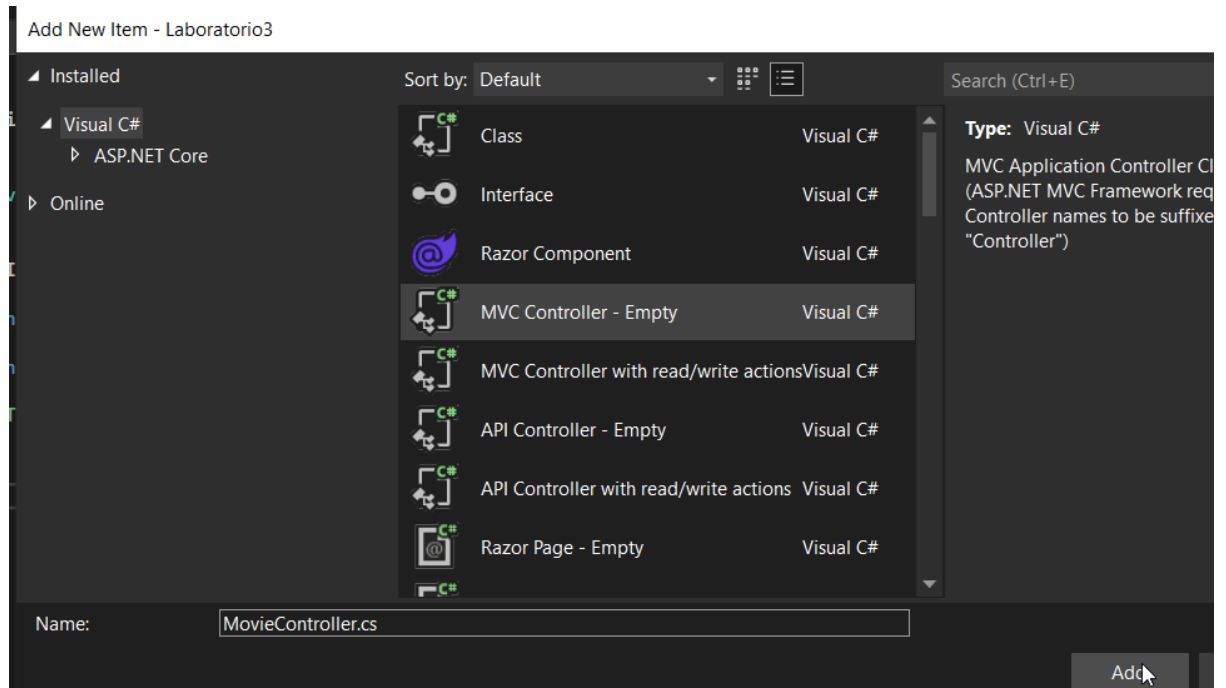
Esta acción va a crear una clase a la cual vamos a agregarle los siguientes atributos: id, nombre, género y año.

```
1 using System;
2
3 namespace Laboratorio3.Models
4 {
5     public class MovieModel
6     {
7         public int Id { get; set; }
8         public string Name { get; set; }
9         public string Genre { get; set; }
10        public DateTime ReleasedDate { get; set; }
11    }
12 }
13
14
```

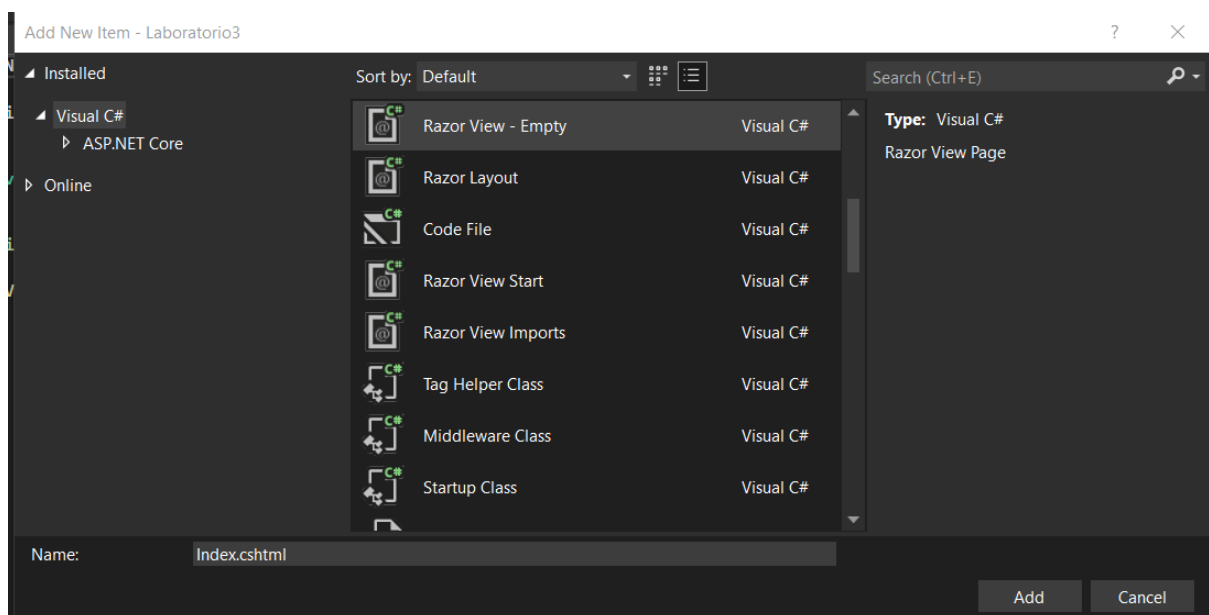
Notas importantes sobre el código:

- La palabra using se utiliza permite utilizar las funcionalidades y tipos de datos que provee la librería o clase que se llama, funciona similar a un import en otros lenguajes.
- Namespace es un conjunto clases, delegados, objetos relaciones. Estos ayudan a organizar el código. Para más información revisar el siguiente [artículo](https://platzi.com/blog/namespace-en-c-sharp/#:~:text=En%20C%23%20los%20Namespaces%20se,mantenemos%20bien%20limpio%20y%20estructurado.)
<https://platzi.com/blog/namespace-en-c-sharp/#:~:text=En%20C%23%20los%20Namespaces%20se,mantenemos%20bien%20limpio%20y%20estructurado.>

2. Cree un controlador, para lograr esto de click derecho sobre la carpeta *Controllers* en el *Solution Explorer*. Seleccione *Add -> Controller*. Ahí seleccione un controlador MVC vacío como en la siguiente imagen:



3. Dentro de la carpeta de views, agregaremos un nuevo folder llamado movie, en el cual estarán todas las vistas que el MovieController va a manejar. Luego de click derecho sobre el nuevo folder Movie, seleccione *Add -> View* en el explorador seleccione agregar razor view - empty como lo muestra la siguiente imagen. El nuevo view se debe llamar Index

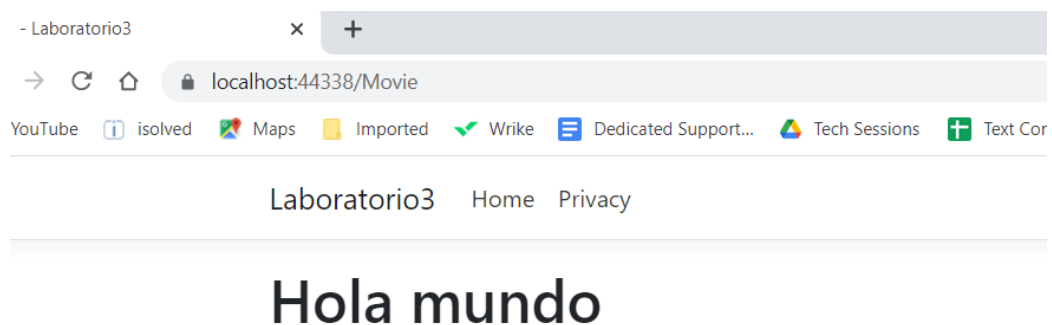




4. Vamos a agregar un pequeño título a nuestra nueva vista:

```
Index.cshtml  MovieController.cs  MovieModel.cs
1  @*
2  |  For more information on enabling MVC for empty projects,
3  |  *@
4  |  @{
5  |  }
6  |  <h1>Hola mundo</h1>
```

Ahora correremos la solución y una vez que podamos ver la página de inicio vamos a cambiar la URL a: <https://localhost:44338/Movie> (en su computadora su número de puerto puede variar localhost:44338). Así podremos ver un resultado como este:



Recordemos que como funciona el routing en proyectos MVC <https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/adding-controller?view=aspnetcore-6.0&tabs=visual-studio>

5. Ahora bien es hora de modificar el controlador para que la vista reciba una lista de MovieModels, para ser desplegados en el view cuando se modifique el view.



```
x.cshtml MovieController.cs MovieModel.cs
Laboratorio3 Laboratorio3.Controllers.MovieController Index()

1 using Laboratorio3.Models;
2 using Microsoft.AspNetCore.Mvc;
3 using System;
4 using System.Collections.Generic;
5
6 namespace Laboratorio3.Controllers
7 {
8     0 references
9     public class MovieController : Controller
10    {
11        0 references
12        public IActionResult Index()
13        {
14            var movies = GetListOfMovies();
15            ViewBag.MainTitle = "List of my favorite films";
16            return View(movies);
17        }
18
19        1 reference
20        private List<MovieModel> GetListOfMovies()
21        {
22            List<MovieModel> movies = new List<MovieModel>();
23            movies.Add(new MovieModel { Id = 1, Name = "Pulp Fiction", Genre = "Crime/Drama",
24                ReleasedDate = new DateTime(1994, 10, 14) });
25            movies.Add(new MovieModel { Id = 2, Name = "Toy Story", Genre = "Family/Comedy",
26                ReleasedDate = new DateTime(1995, 11, 22) });
27            movies.Add(new MovieModel { Id = 3, Name = "Mulan", Genre = "Family/Comedy",
28                ReleasedDate = new DateTime(1998, 06, 19) });
29
30            return movies;
31        }
32    }
33 }
```

Notas importantes sobre este código:

- El ViewBag se utiliza para poder transferir información del controlador a la vista que no es parte del modelo. Para ver mayor información de como funciona: <https://www.tutorialsteacher.com/mvc/viewbag-in-asp.net-mvc>
- El **IActionResult**, permite devolver diferentes acciones incluyendo redirigir a la vista deseada con modelos calculados, también puede devolver BadRequest entre otras acciones. Para ver otro tipos de acciones que se pueden devolver con el IActionResult revise este link: <https://docs.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0>
- Los list en C# al igual que en otros lenguajes son estructuras de datos que nos permiten tener un conjunto de elementos, las cuales se les pueden hacer una serie de operaciones. Para mayor información revise: <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-6.0>

6. El siguiente paso es modificar el View para desplegar todos los valores de la lista de MovieModels. Para ello es necesario utilizar el siguiente código.

```
Index.cshtml | MovieController.cs | MovieModel.cs
@model List<Laboratorio3.Models.MovieModel>;

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

<h1>@ViewBag.MainTitle</h1>

<div>
    <table class="table" >
        <thead>
            <tr>
                <th>Movie title</th>
                <th>Gender</th>
                <th>Released date</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
                <tr>
                    <td>@item.Name</td>
                    <td>@item.Genre</td>
                    <td>@item.ReleasedDate</td>
                </tr>
            }
        </tbody>
    </table>
</div>
<div>
    <button>I am a button</button>
</div>
```

A continuación se explicara un poco de qué significa este código, esta es una página razor, que nos permite trabajar con código HTML. En laboratorios futuros se cubrirá con mayor detenimiento las páginas razor.

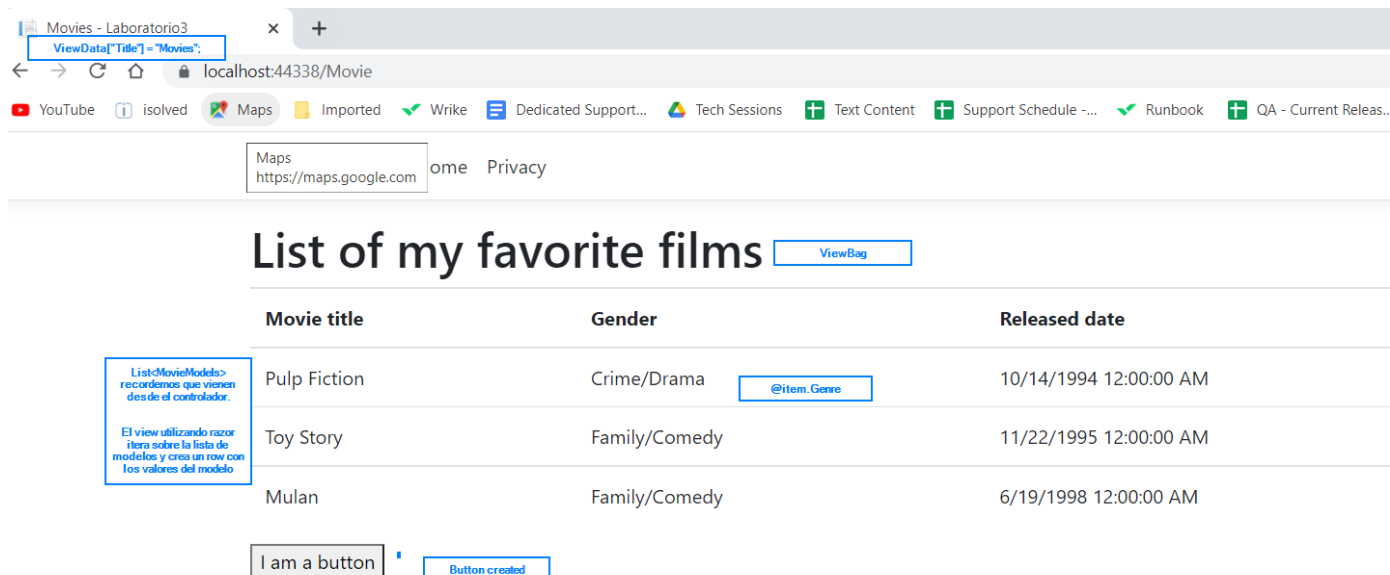
- Todos los tags que empiezan con <> y finalizan con </> son tags nativos de HTML (ejemplo: <div> </div>). Para ver más información de como funciona html: https://www.w3schools.com/html/html_intro.asp

- Por otro lado los comandos que empiezan con @ son los comando razor que nos permiten hacer diferentes acciones como por ejemplo:
 - Definir con qué modelo vamos a trabajar en la vista:
`@model List<Laboratorio3.Models.MovieModel>;`
 - Definir qué tipo de título tendrá la página
`@{
 ViewData["Title"] = "Movies";
}`
 - Definir qué valores del modelo se va a utilizar
Ejemplo: `@item.Name`
 - Crear un ciclo para iterar la lista de modelos y generar el mismo código de la tabla una y otra vez

```
<tbody>
  @foreach (var item in Model)
  {
    <tr>
      <td>@item.Name</td>
      <td>@item.Genre</td>
      <td>@item.ReleasedDate</td>
    </tr>
  }
</tbody>
```

Código que se repite por cada modelo de la lista

7. Como resultado final se obtendrá los siguiente página:



ViewData["Title"] = "Movies";

localhost:44338/Movie

Maps
https://maps.google.com

List of my favorite films

ViewBag

Movie title	Gender	Released date
Pulp Fiction	Crime/Drama	10/14/1994 12:00:00 AM
Toy Story	Family/Comedy	11/22/1995 12:00:00 AM
Mulan	Family/Comedy	6/19/1998 12:00:00 AM

I am a button

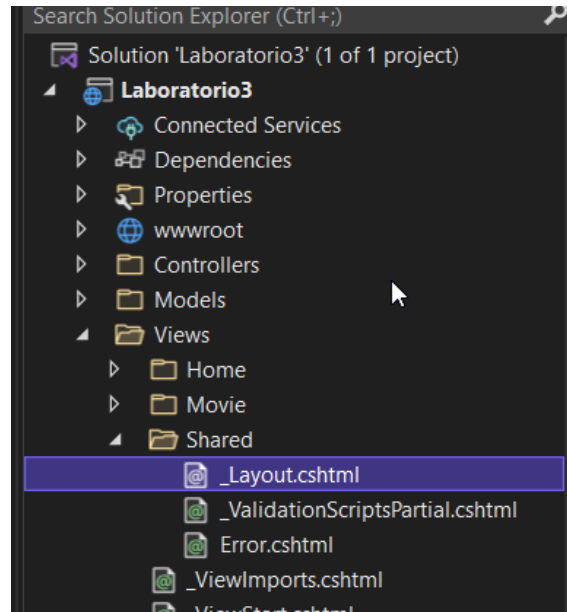
Button created

List<MovieModels> recordemos que vienen desde el controlador.

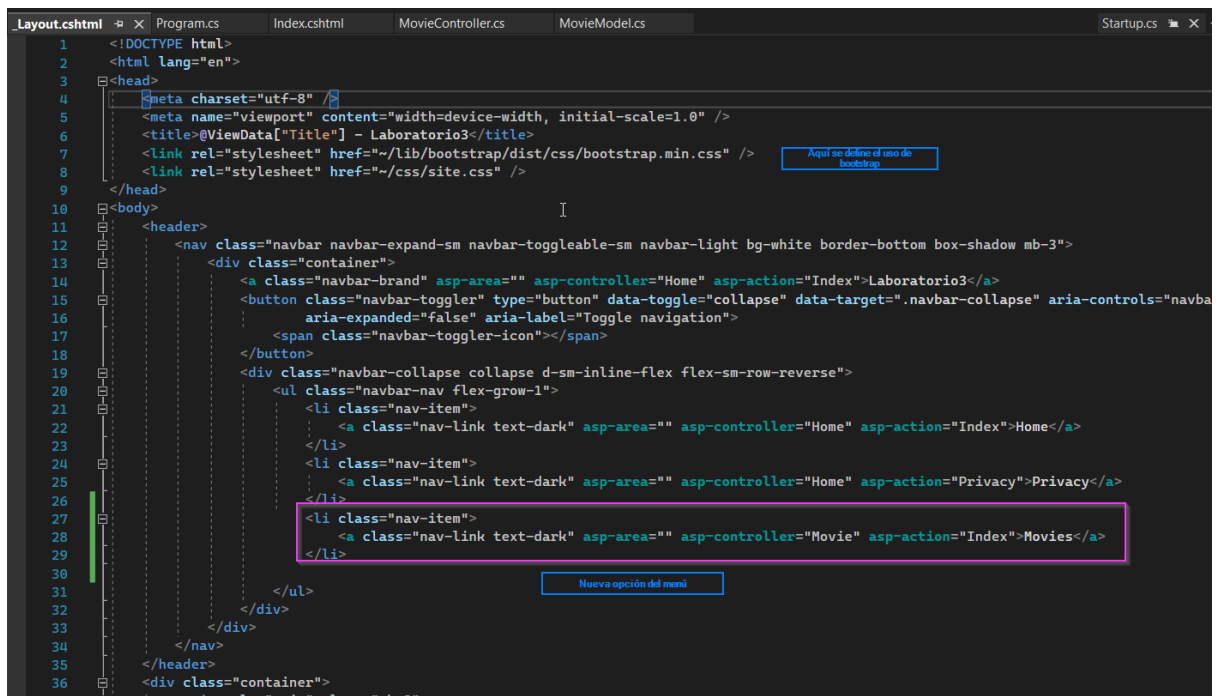
El view utilizando razor itera sobre la lista de modelos y crea un row con los valores del modelo

@item.Genre

8. Como último se agregaremos una opción en el menú del layout:



El layout como se vio en el tutorial es el que contiene la definición de la página html. También contiene la definición del menú de navegación, así que agregaremos una nueva opción que debe redirigir a la opción de Movies:



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>@ViewData["Title"] - Laboratorio3</title>
7   <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
8   <link rel="stylesheet" href="~/css/site.css" />
9 </head>
10 <body>
11   <div class="header">
12     <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
13       <div class="container">
14         <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Laboratorio3</a>
15         <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-expanded="false" aria-label="Toggle navigation">
16           <span class="navbar-toggler-icon"></span>
17         </button>
18         <div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
19           <ul class="navbar-nav flex-grow-1">
20             <li class="nav-item">
21               <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
22             </li>
23             <li class="nav-item">
24               <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
25             </li>
26             <li class="nav-item">
27               <a class="nav-link text-dark" asp-area="" asp-controller="Movie" asp-action="Index">Movies</a>
28             </li>
29           </ul>
30         </div>
31       </div>
32     </nav>
33   </div>
34   <div class="container">
35     <main role="main" class="pb-3">
```

9. Por último trate de crear un nuevo modelo, controlador y vista para un modelo de tipo canción, debe definir al menos 5 atributos en el modelo. Y el



controlador debe tener un IActionResult que devuelva una vista con un solo modelo, utilice la siguiente el siguiente mockup como ejemplo:

Laboratorio3 Home Privacy Movies Song

Mi canción preferida

Nombre:

Atributo2:

Atributo3:

Atributo4:

Atributo5:

Entregable

Para este laboratorio se debe compartir el repositorio usado con el asistente y la profesora a los siguientes usuarios de github:

- rebeca-ov
- ChristianRojasRios