



PROYECTO MVC

Instituto Tecnológico Superior del Sur de Guanajuato



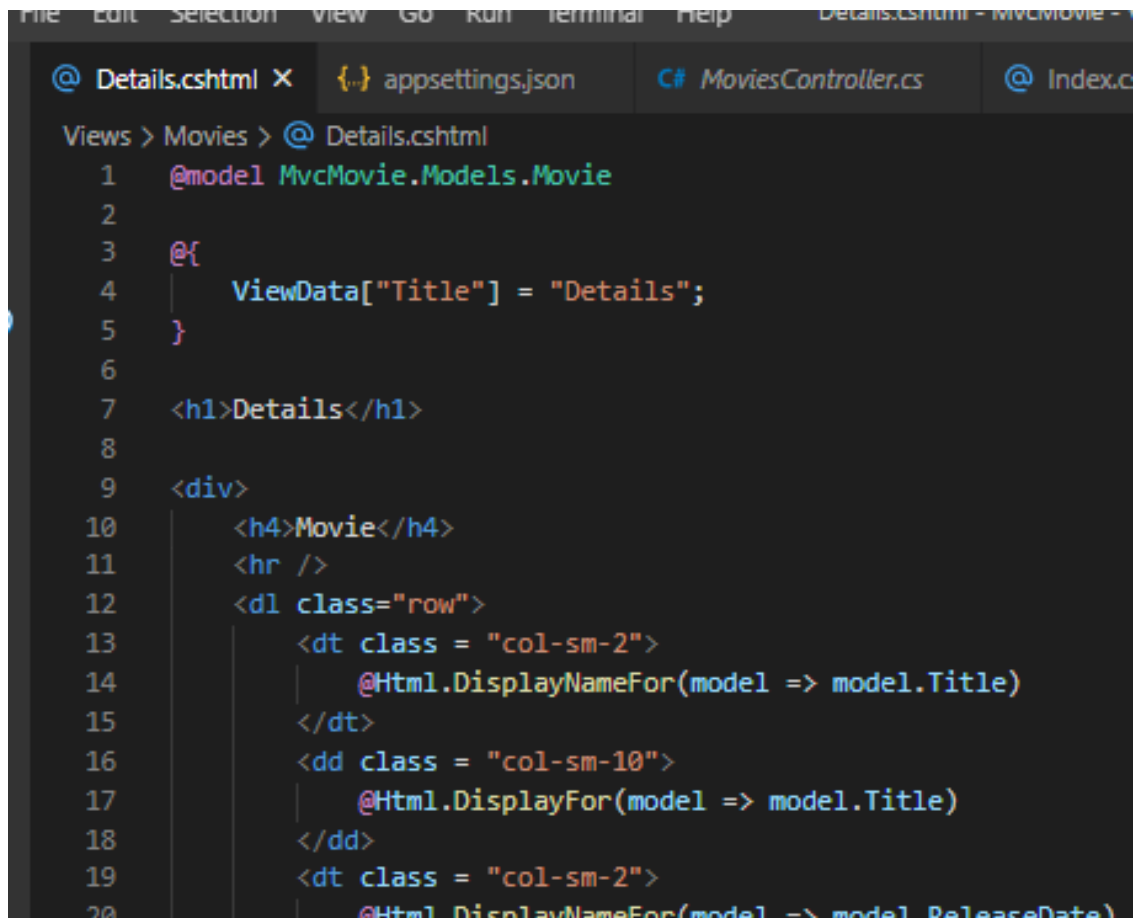
19 DE FEBRERO DE 2021

PROGRAMACION WEB 2

Alumno: Luis Alberto Ignacio Esteban

```
var movie = await _context.Movie  
    .FirstOrDefaultAsync(m => m.Id == id);
```

Se pasa una expresión lambda a FirstOrDefaultAsync para seleccionar entidades de película que coincidan con los datos de ruta o el valor de la cadena de consulta.



```
@model MvcMovie.Models.Movie  
  
@{  
    ViewData["Title"] = "Details";  
}  
  
<h1>Details</h1>  
  
<div>  
    <h4>Movie</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.ReleaseDate)
```

La declaración de @model en la parte superior del archivo de vista especifica el tipo de objeto que espera la vista. Cuando se creó el controlador de películas, @model se incluyó la siguiente declaración:

```
@model MvcMovie.Models.Movie
```

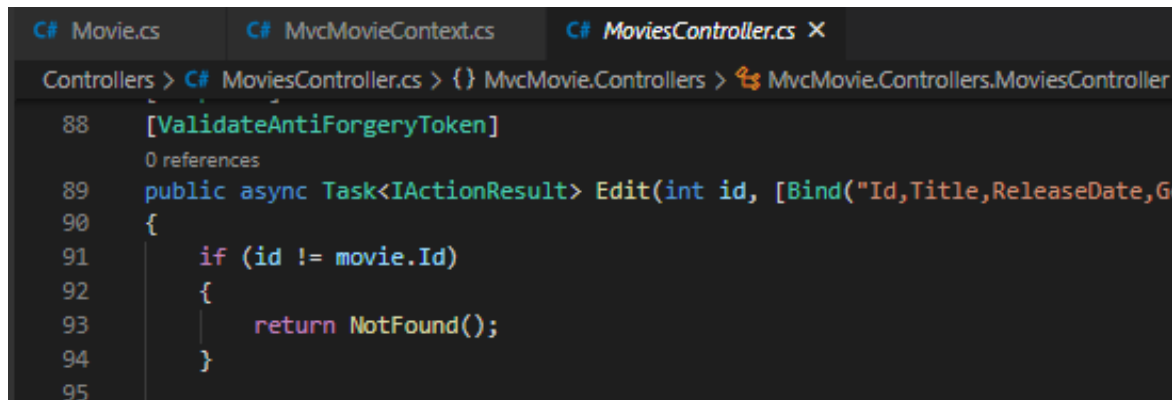
Esta @modeldirectiva permite el acceso a la película que el controlador pasó a la vista. El Modelobjeto está fuertemente tipado.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.AddDbContext<MvcMovieContext>(options =>
options.UseSqlite(Configuration.GetConnectionString("MvcMovieContext")));
}
```

El objeto `MvcMovieContext` controla la tarea de conexión a la base de datos y asignación de objetos `Movie` a los registros de la base de datos. El contexto de base de datos se registra con el contenedor de inserción de dependencias en el método `ConfigureServices` del archivo `Startup.cs`:



Los vínculos `Edit` (Editar), `Details` (Detalles) y `Delete` (Eliminar) se generan mediante el asistente de etiquetas de delimitador de MVC Core en el archivo `Views/Movies/Index.cshtml`.



El atributo `[Bind]` es una manera de proteger contra el exceso de publicación. Solo debe incluir propiedades en el atributo `[Bind]` que quiera cambiar.

```

84 // POST: Movies/Edit/5
85 // To protect from overposting attacks, enable the specific properties you w
86 // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
87 [HttpPost]
88 [ValidateAntiForgeryToken]
89 0 references
90 public async Task<IActionResult> Edit(int id, [Bind("Id,Title,ReleaseDate,Ge
91 {
92     if (id != movie.Id)
93     {
94         return NotFound();
95     }
96
97     if (ModelState.IsValid)
98     {
99         try
100         {
101             _context.Update(movie);

```

El atributo `HttpPost` especifica que este método `Edit` se puede invocar solamente para solicitudes POST. Podría aplicar el atributo `[HttpGet]` al primer método de edición, pero no es necesario hacerlo porque `[HttpGet]` es el valor predeterminado.

El atributo `ValidateAntiForgeryToken` se usa para impedir la falsificación de una solicitud y se empareja con un token antifalsificación generado en el archivo de vista de edición (`Views/Movies/Edit.cshtml`). El archivo de vista de edición genera el token antifalsificación con el asistente de etiquetas de formulario.

```

// GET: Movies
3 references
public async Task<IActionResult> Index(string searchString)
{
    var movies = from m in _context.Movie
                  select m;

    if (!String.IsNullOrEmpty(searchString))
    {
        movies = movies.Where(s => s.Title.Contains(searchString));
    }

    return View(await movies.ToListAsync());
}

```

El código `s => s.Title.Contains()` anterior es una expresión Lambda. Las lambdas se usan en consultas LINQ basadas en métodos como argumentos para métodos de operador de consulta estándar, tales como el método `Where` o `Contains` (usado en

el código anterior). Las consultas LINQ no se ejecutan cuando se definen ni cuando se modifican mediante una llamada a un método como Where, Contains u OrderBy. En su lugar, se aplaza la ejecución de la consulta. Esto significa que la evaluación de una expresión se aplaza hasta que su valor realizado se repita realmente o se llame al método ToListAsync.

```
namespace MvcMovie.Models
{
    public class MovieGenreViewModel
    {
        public List<Movie> Movies { get; set; }

        public SelectList Genres { get; set; }

        public string MovieGenre { get; set; }

        public string SearchString { get; set; }
    }
}
```

El modelo de vista de película y género contendrá:

Una lista de películas.

SelectList, que contiene la lista de géneros. Esto permite al usuario seleccionar un género de la lista.

MovieGenre, que contiene el género seleccionado.

SearchString, que contiene el texto que los usuarios escriben en el cuadro de texto de búsqueda.

Repositorio github

<https://github.com/LuisAlbertoIgnacioEsteban/MvcMovie>

