**Resumen .NET 6.0**

Nombre de servidor

[DESKTOP-44MC03F]

**Clase** 7

1) ¿Cuál es el primer elemento de una matriz 0 o 1?

2) ¿Cual es la diferencia entre un acumulador y un contador?

**Clase** 8

1. Diferencia entre vector y matriz?

2. como se recorre un array de dos dimensiones?

**Clase** 9

1. Nombre de la **Clase** para obtener datos aleatorios

2. donde se coloca el tipo de dato para las listas

**Clase** 10

1. Que Significan el Count y la Capacity en una lista?

2. Que son las estructuras de dato LIFO y FIFO, y dar un ejemplo de cada una.

Pilas: Estructura donde el ultimo en entrar es el primero en salir. new Stack<int>();

Colas: Similar a pila, pero el primero en entrar es el primero en salir. new Queue<int>();

**Clase** 11

1. Cuales son las tres estrucuras logicas que plantea el teorema de la programacion estructurada? --> De Secuencia, Seccion e Iteracion

2. Cual es la diferencia entre instancia y clase.

Diferencia entre instancia y clase:

La clase es el molde del objeto que quiero crear.

La instancia es el objeto creado a partir de una clase.

Paradigma: Forma de encarar un tema.

Objeto, Modelo (UML => Modelo grafico para representar objetos)

POO: reutiliacion de codigo.

**Clase** 12:

1. Como invocamos o llamamos a una funcion o metodo?

GetInfo es un metodo o una funcion y lo otro son propiedades, para poder llamar un metodo o funcion si o si necesito usar parentesis.

2. Es posible tener dos funciones con el mismo nombre? Justifique.

DateTime:

fecha = new DateTime(1986, 11, 4) //Anio, Mes, Dia

fecha.ToString("yyyy MM dd ddd")

fecha.ToString("dddd dd-MMM-yy HH:mm")

**Clase** 13:

1. Es necesario inicializar todos los atributos de la clase, en el metodo constructor?

RTA: Solo aquellos a los que realmente quiero/voy a inicializar.

2. Que sucede si no defino un cuerpo para los metodos get y set para una propiedad?

**Clase** 14:

1. Que es el recolector de basura? Garbage collector

Se da cuenta que hay muchos objetos que quedaron sin usar, los destruye, llama al destructor de ese objeto y libera memoria. Funciona automaticamente y detecta cuando hay mucha necesidad de memoria, lo va destruyendo.

2. Que es la herencia?

La herencia es un mecanismo de la POO, por el cual las clases se pueden derivar entre si, una clase hija se deriva de otra clase padre, o clase abuelo.

Protected --> Solo los que son parte de la herencia pueden ver/usar ese metodo / funcion.

Static --> Te permite usar un metodo de una clase sin necesidad de instanciarlo. (Si se puede, se define como static)

Sobrecarga de metodos --> Definir un metodo con el mismo nombre, pero que esa funcion tenga un procesamiento distinto basado en sus parametros.

Un objeto se puede comportar de distintas maneras cuando tiene q realizar una funcion.

Constructor: Funcion que se ejecuta al momento de instanciarse la clase. Se llama exctamente igual que la clase.

**Clase** 15:

1. Si yo creo una referencia de una instancia de una clase y modifico el valor de un atributo de dicha referencia, que sucede con el valor del atributo de la instancia original?

Al modificar el valor de un atributo a traves de una referencia, el valor del mismo atributo va a ser modificado acordando con el valor modificado de la misma referencia. Minuto 46

2. Que sucede si intentamos castear una instancia de un objeto en una clase a la cual no pertenece o no hereda?

Al castear una instancia en una clase a la cual no pertenece, se realizara un error.

Las instancias de las clases son de tipo referencia. Minuto 30.

**Clase** 16:

1. Para que usamos el modificador de acceso private?

2. Para que sirven los modificadores de clase abstract y sealed?

El Set se ejecuta cuando se asigna un valor a la propiedad.

El Get cuando se lee la propiedad.

int.TryParse(value, out nombreVariable); //Intenta la conversion, si se puede, el metodo devuelve el resultado de la conversion, y tambien, a traves del out, avisa que el metodo va a emitir una salida.

TimeSpan: Desde el inicio hasta ahora, es la fecha actual, desde el inicio hasta cierta fecha, es la fecha de nacimiento. Si yo resto la fecha actual y la de nacimiento, queda el espacio de tiempo entre esas dos variables.

Casting: (int) atrapar un determinado dato como si fuera otro.

Abstract: Le da la caracteristica a una clase de que no puede ser instanciada. Es decir, llamada.

Sealed: Sellada, esta clase no va a poder ser heredada mas.

**Clase** 17:

1. Cuantas interfaces puede implementar una clase?

2. Puedo declarar una variable como una interfaz? Si, si se puede declarar una variable como una interfaz.

Se puede decir que una interfas es como una clase abstracta.

C# no soporta herencia multiple, pero si podes implementar multiples interfaces.

Siempre es mejor trabajar con interfaces y no clases.

**Clase** 18:

1. Las interfaces tienen la capacidad de tener herencia multiple, o no?

Las interfaces pueden heredad de multiples interfaces, es capaz de herencia multiple.

2. La herencia, sirve para reutilizar codigo?

Los metodos que tienen que implementarse x las interfaces, tienen que ser publicos.

Las interfaces si puede tener herencia multiple.

**Clase** 19:

1. En los diccionarios, se puede utilizar cualquier tipo de dato como clave?

Si, en los diccionarios se pueden utilizar todos los tipos de datos. Pero no te conviene utilizar ciertos tipos de datos como Objetos, Listas, etc.

2. El repositorio de persona, puede guardar datos de cualquier persona, sea de la clase que sea, siempre y cuando herede de la clase persona? Justificar.

LinQ --> trabajar con conjuntos de datos.

**Clase** 20:

1. Que es una clase parcial?

2. Que es un tipo anonimo?

Partial.

**Clase** 21:

1. Cual es la interfaz mas comun que devuelven los metodos LinQ cuando devuelven un resultado?

IEnumerable

2. Si utilizamos interfaces, es necesario que conozca como esta implementada una interfaz por dentro?

No, no es necesario que la clase conozca como esta implementada una interfaz por dentro.

IEnumerable = Se va a poder enumerar. Tiene los metodos necesarios para poder ir yo integrando desde principio a fin.

Con las clases definimos comportamientos. Habilidad / capacidad de comportarse de cierta forma.

LinQ = Language Integrated Query = Consultas integradas al lenguaje.

Es un conjunto de metodos de extencion. Parte de los ORM.

Principios solidos (?

**Clase** 22:

1. Es obligatorio usar restricciones? min: 56:25

No es obligatorio usar restricciones.

2. Es posible invocar a un metodo de una instancia recien creada y que esa instancia no sea guardada en una variable? 1:29:10

Si

**Clase** 23:

1. Una biblioteca, podria considerarse una base de datos? porque?

2. Si mi estructura de datos, no es uniforme, y puede estar consituida de cualquier forma, que me conviene utilizar, NoSQL o SQL?

**Clase** 24:

1. Cuales son los comandos de definicion mas comunes para microsoft SQL Server?

2. Cuales son los comandos mas comunes de manipulacion de datos en SQL Server?

**Clase** 25:

1. Cuando tengo multiples PK, la combinacion de esta, se puede repetir?

2. Mediante la utilizacion del Join, tengo una limitacion a la cantidad de tablas a relacionar?

**Clase** 26:

1. Se puede combinar mas de un operador en la misma consulta para filtrar resultados con where?

2. Para ejecutar una consulta con order by, es necesario que las columnas involucradas cuenten con indices creados?

**Clase** 27:

1. Para que sirve using en C#?

2. Cuales son las dos distintas formas que tiene entity framework de encarar el problema?

Code First y Database First

**Clase** 28:

1. Que es LinQ y que facilidades nos proporciona.

LINQ es un componente de .NET Framework que proporciona capacidad de consulta en colecciones en C# o VB.

2. Cual es la interface que implementa DBset para realizar consultas con LinQ

Any() //Cualquiera que cumpla con la condicion

Select() //Selecciona partes definidas. Como un select comun

FirstOrDefault() //Selecciona el primero o unico de los registros que aparecen

**Clase** 29:

1. En el aproach code first de EF, debo tener una BD creada para poder comenzar a desarrollar mi app?

No. Podes usar Migration.

2. Que paquetes NuGet debo instalar para poder trabajar con EF utilizando Aproach code first, SQL, y Migrations.

Migraciones: Forma de hacer que el modelo de datos se genere o construya dentro de la BD.

Como?

1 - NuGet package manager console

2 - add migration FirstMigration

3 - Genera automaticamente la Migracion

**Clase** 30:

1. Cuales son las dos dependencias q tenemos q instalar para utilizar dapper con sqlserver?

System.Data.SqlClient

Dapper

2. Cual es el nombre del metodo que utiliza dapper para obtener el primer registro de una consulta o en su defecto un nulleable?

Dapper: ORM - Escribis la consulta.

**Clase** 31:

1. Que devuelve el metodo excecute de dapper?

Devuelven un numero: La cantidad de consultas que fueron afectadas.

**Clase** 32: A partir de ahora Todo Web

1. Que significa CSHTML?

​Que dentro de c-sharp esta envevido html.

Razor: Motor de procesamiento de texto.

2. Que significa MVC?

Model View Controler

[dotnet new mvc] en consola --> Proyecto web medio hecho

ASP.NET -> Model View Controler (MVC) -->

App Web con servidor y cliente: Angular o React aparte de C# solamente. Frameworks Web de JS

Empezamos con el mas simple: ASP.NET Core Web App

En el layout defino todo el esqueleto de la app, si pongo algo ahi, el mismo se aplica en todas las paginas.

**Clase** 33:

1. Cuales son los componentes principales del patron MVC?

Modelo - Vista - Controladores

2. Cual es la responsabilidad del componente controler?

Encargado de gestionar toda la informacion y logica. Devolver el modelo si fuese necesario. Elegir que vista mostrar.

**Clase** 34:

1. Para que me sirve el archivo viewstart y todo lo que se defina dentro de el?

El archivo viewstart sirve para comenzar / hacer init a la vista / layout. Asi el mismo puede inicializarce facilmente sin necesidad de colocar cierto codigo en cada archivo cshtml.

Layout: Todo dentro de eso, va aser reutilizado / repetido x todas las paginas. Ej: Header / Footer. Sin necesidad de escribir y repetir el mismo codigo todo el tiempo.

Con Razor, el layout es procesado x el servidor para hacer esta utilidad.

El @ sirve para empezar a escribir codigo C# dentro de html o css, etc

Dentro de viewstart y viewimport se encuentra el init del layout para todas las paginas.

ViewData[""] en el index o en cualquier pagina cshtml es lo que indica que se lea el init del view general con su layout predeterminado.

Cada vista puede tener un solo modelo. @model ... crea una variable Model, que se puede usar x ejemplo como Model.Vehiculos

**Clase** 35:

1. Que tipo de dato es un view data?

View Data es un diccionario.

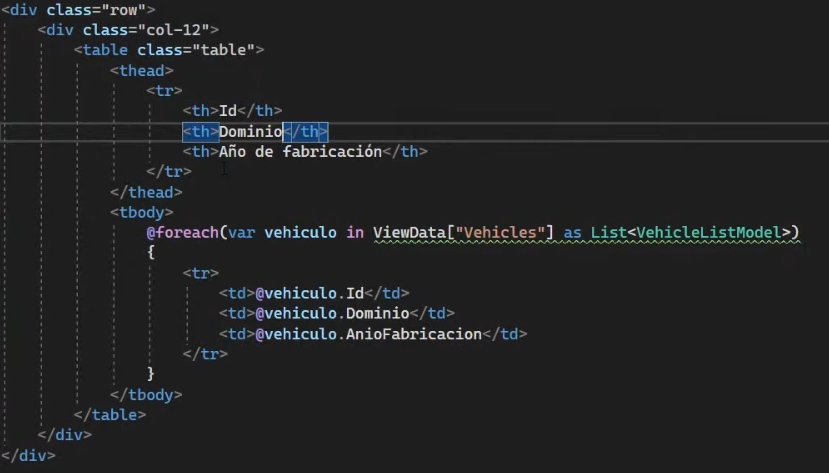
1. Cual de los 2 metodos trabaja mas rapido, viewBag o viewData?

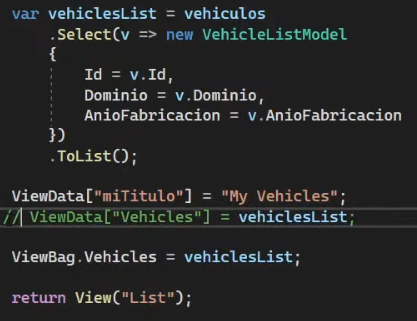
viewData es mas rapido que viewBag.

Cambiar coneccion: AppSettings - source = [DESKTOP-44MC03F]

Data/Applicationdvcontext

Herramientas -> Admin de paquetes NuGet -> Consola -> [update-database]



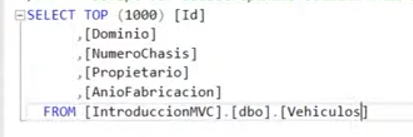


Cdn links --> links para los servidores.

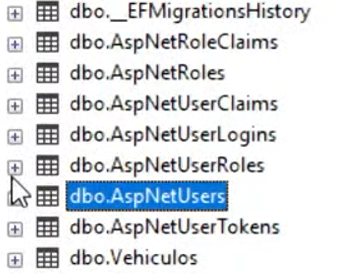
**Clase** 36:

1. \_
2. \_ Cual es la etiqueta de html para llamar al controlador, el atributo para llamar al controlador dentro del formulario?

Asp-action



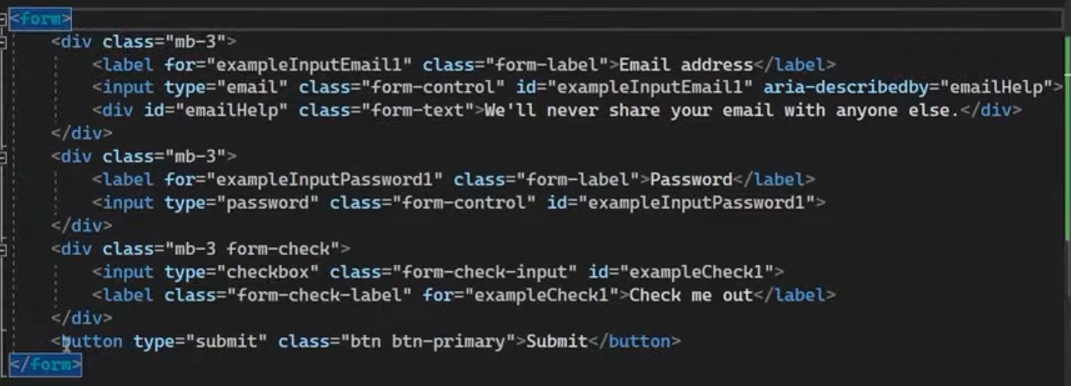
Generadas automaticamentes por MVC.



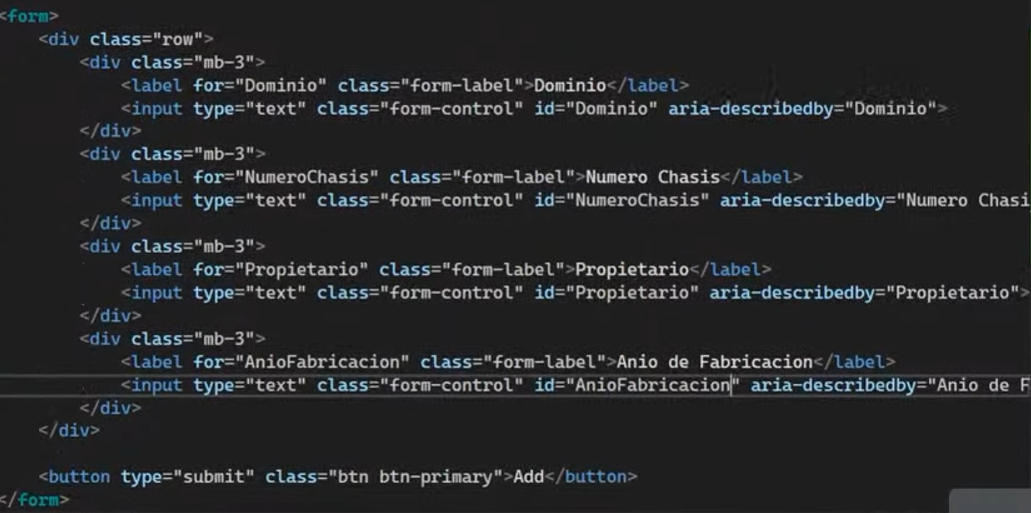
Click derecho -> edit top -> editar la tabla o agregar cosas.

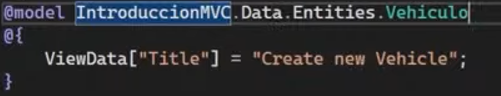
Boostrap

-> Form

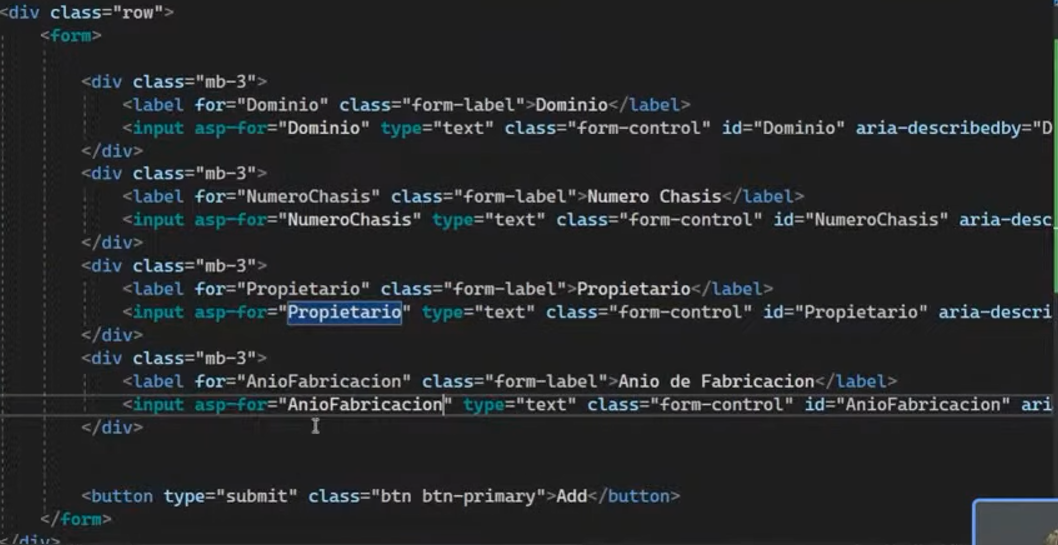


->

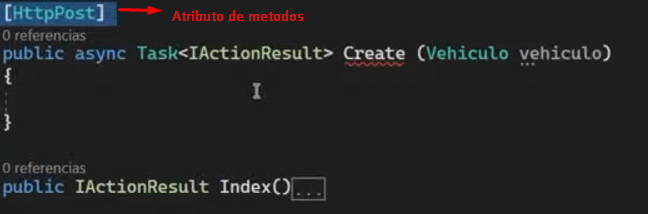




->



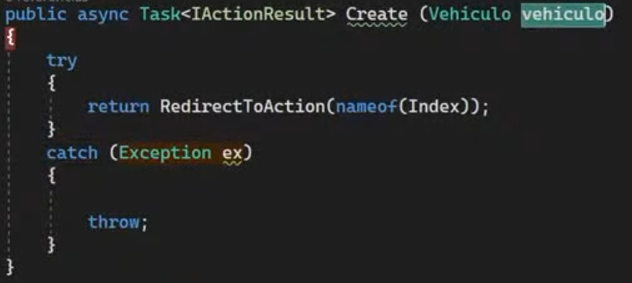
Asp-for -> Lo q va a guardar ese campo es lo q da(=”…”), es parte del framework de MVC. Se ejecuta del lado del servidor.



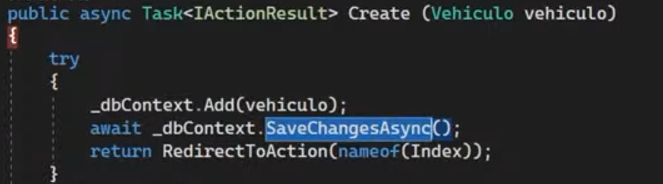
HttpPost -> esta entre corchetes. Va a atajar la accion post de html cuando venga en el url create.

[..] -> Significa que es un atributo, sirve para algo.

->



->



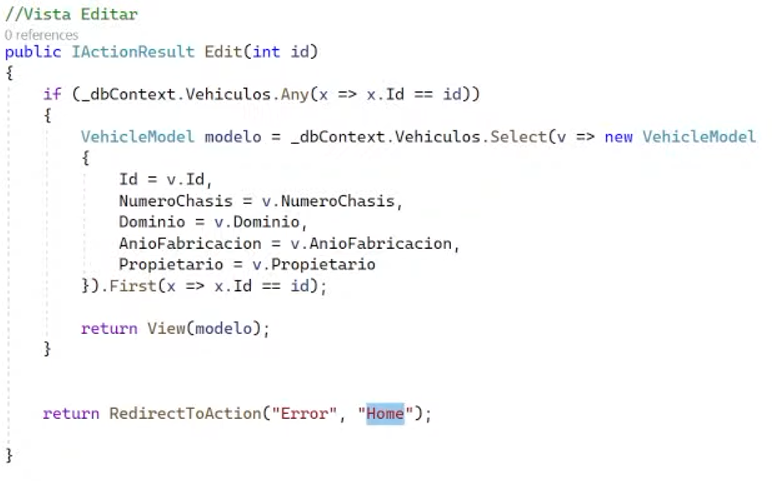
SaveChangesAsync() -> Hace el acceso a la BD y lo graba en otro hilo, en uno diferente al que contiene el resto de las cosas, de manera asincrono para que agilice el proceso en caso de muchas entradas.

**Clase** 37:

1. \_ Que es una vista parcial y para que sirve?

Es una vista comun donde vamos a colocar el codigo comun que nos va a servir en varias vistas, para evitar redundancias y no duplicar codigo.

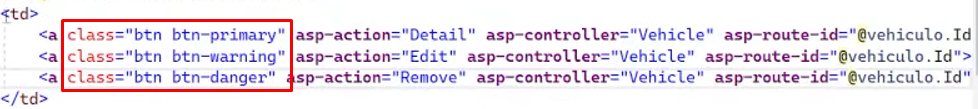
VehicleController ->



List ->



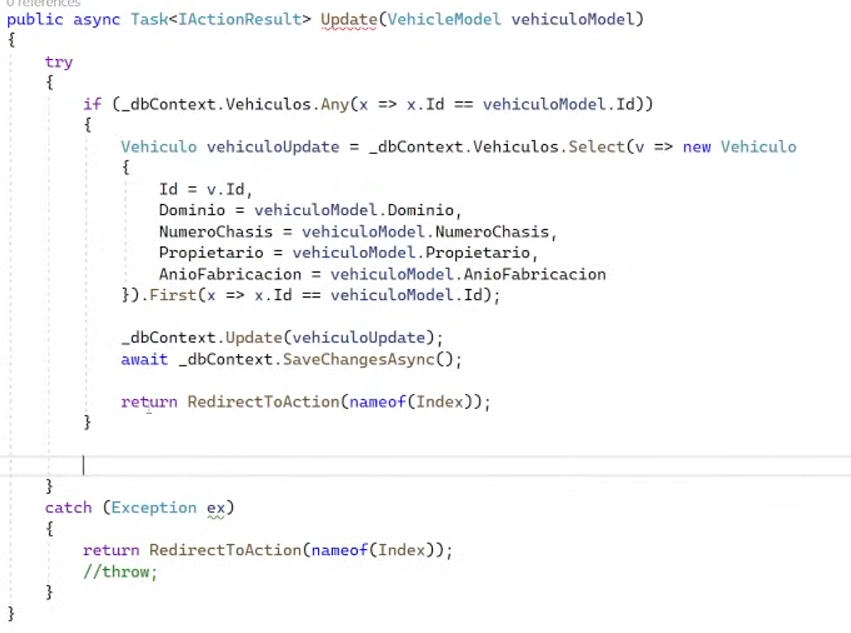




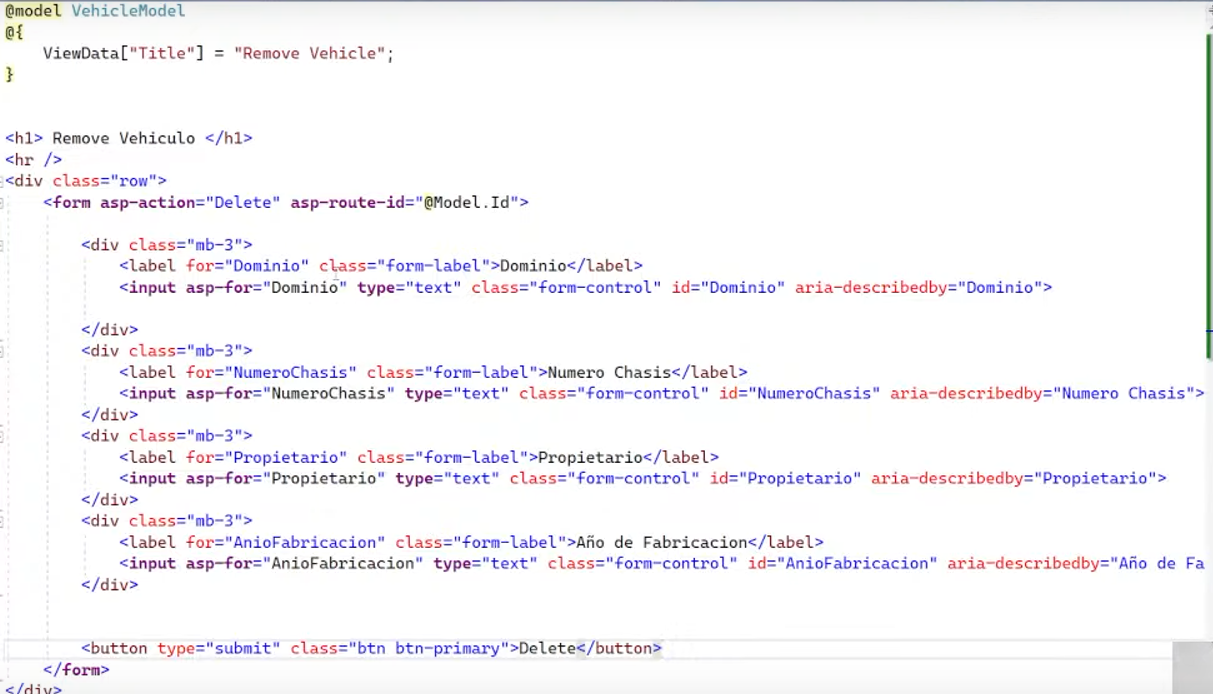
Edit.cshtml ->



VehicleController ->



Remove.cshtml ->

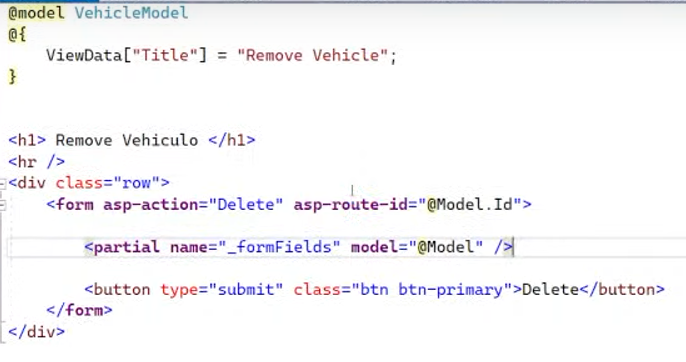


VehicleController ->

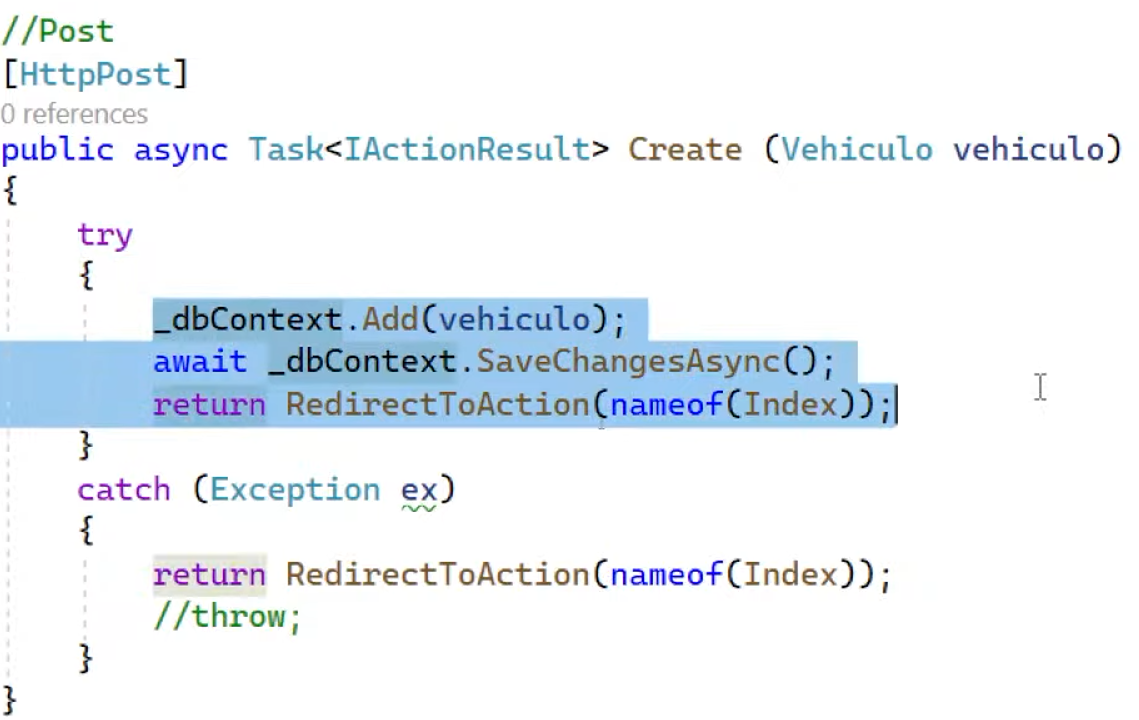


New.cshtml ->

Remove.cshtml ->



VehicleController ->

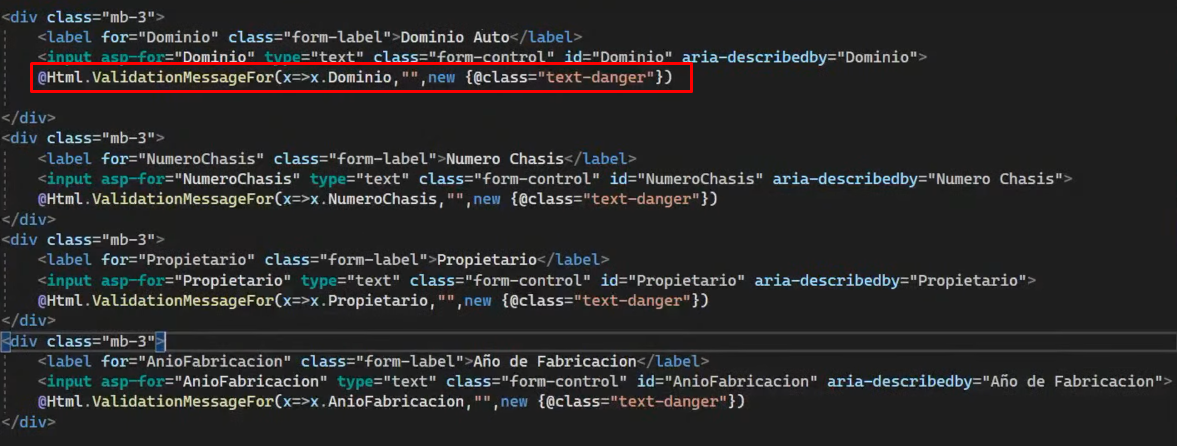


Clase 38:

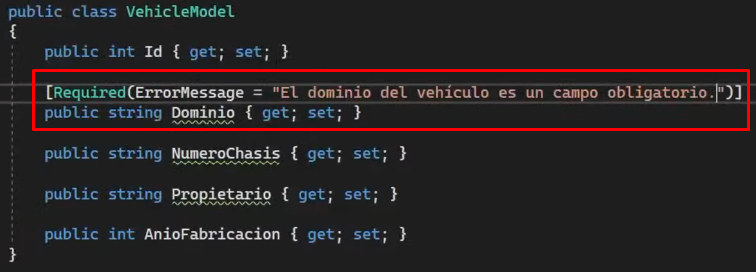
1. \_ Que using debemos incorporar en nuestro modelo para poder hacer validaciones de los campos del lado del servidor?

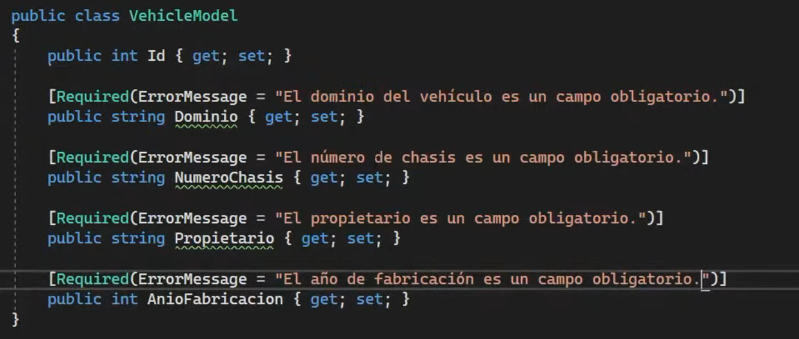
Using System.ComponentModel.DataAnotations;

formFields.cshtml ->



VehicleModels ->



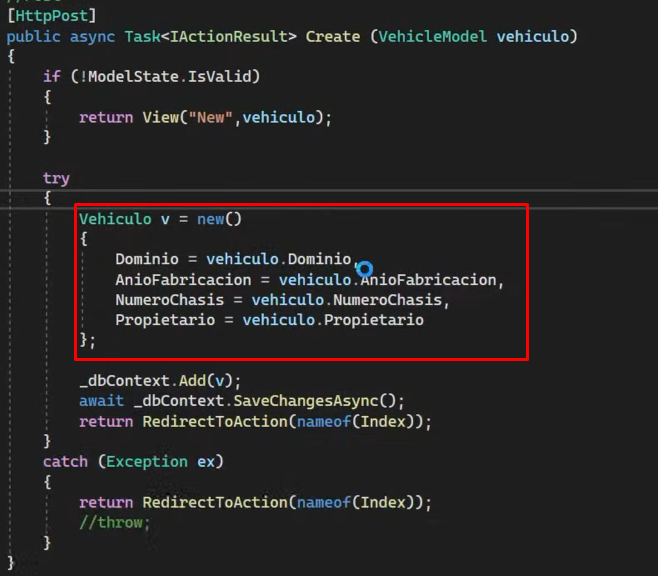


Largo maximo -> Importante para seguridad a veces, estando en el backend.





VehicleController ->



Clase 39:

1. \_
2. \_

Extra:

Crearemos una clase llamada Serie con las siguientes características:

Sus atributos son titulo, numero de temporadas, entregado, genero y creador.

Por defecto, el numero de temporadas es de 3 temporadas y entregado false.

El resto de atributos serán valores por defecto según el tipo del atributo.

Los constructores que se implementaran serán:

Un constructor por defecto.

Un constructor con el titulo y creador.

El resto por defecto.

Un constructor con todos los atributos, excepto de entregado.

Los métodos que se implementara serán:

Métodos get de todos los atributos, excepto de entregado.

Métodos set de todos los atributos, excepto de entregado.

Sobrescribe los métodos toString.

Crearemos una clase Videojuego con las siguientes características:

Sus atributos son titulo, horas estimadas, entregado, genero y compañia.

Por defecto, las horas estimadas serán de 10 horas y entregado false.

El resto de atributos serán valores por defecto según el tipo del atributo.

Los constructores que se implementaran serán:

Un constructor por defecto.

Un constructor con el titulo y horas estimadas.

El resto por defecto.

Un constructor con todos los atributos, excepto de entregado.

Los métodos que se implementara serán:

Métodos get de todos los atributos, excepto de entregado.

Métodos set de todos los atributos, excepto de entregado.

Sobrescribe los métodos toString.

Como vemos, en principio, las clases anteriores no son padre-hija, pero si tienen en común, por eso vamos a hacer una interfaz llamada Entregable con los siguientes métodos:

entregar(): cambia el atributo prestado a true.

devolver(): cambia el atributo prestado a false.

isEntregado(): devuelve el estado del atributo prestado.

Método compareTo (Object a), compara las horas estimadas en los videojuegos y en las series el numero de temporadas.

Como parámetro que tenga un objeto, no es necesario que implementes la interfaz Comparable.

Recuerda el uso de los casting de objetos.

Implementa los anteriores métodos en las clases Videojuego y Serie.

Ahora crea una aplicación ejecutable y realiza lo siguiente:

Crea dos arrays, uno de Series y otro de Videojuegos, de 5 posiciones cada uno.

Crea un objeto en cada posición del array, con los valores que desees, puedes usar distintos constructores.

Entrega algunos Videojuegos y Series con el método entregar().

Cuenta cuantos Series y Videojuegos hay entregados.

Al contarlos, devuélvelos.

Por último, indica el Videojuego tiene más horas estimadas y la serie con mas temporadas.

Muestralos en pantalla con toda su información (usa el método toString()).