Aplicación de Patrón de Diseño MVC con el uso de Scaffolding

1 APRENDIENDO A AUTOMATIZAR LA RECOSTRUCCIÓN POR CAMBIOS EN EL MODELO DE DATOS.

Como ha podido observar hasta ahora, siempre que es necesario hacer un cambio en el modelo de datos, ya sea por ampliarlo o para corregir algún error, hemos necesitado de borrar y recrea todo el escenario nuevamete para que coincidan con la base de datos.

Pero existe una forma de hacerlo de forma automatizada. Dentro del constructor de nuestro DataContext podemos usar diferentes funcionalidades para adaptar la base de datos a los cambios del modelo. Conoceremos el métdodo ***SetInitializer < TContext >*** "Establece el inicializador de base de datos para utilizar para el tipo de contexto dado. El inicializador de base de datos se llama cuando se utiliza el tipo DbContext dado para acceder a una base de datos por primera vez. La estrategia predeterminada para los contextos de Code First es una instancia de CreateDatabaseIfNotExists < TContext > ". (MSDN Microsoft, 17). Usando el ejercicio de la clase anterior, dirijase al constructor del contexto y realice:

```
public ScafMVCContext() : base("name=ScafMVCContext")
{
    Database.SetInitializer<ScafMVCContext>(
        new DropCreateDatabaseIfModelChanges<ScafMVCContext>()
        );
}
```

DropCreateDatabaseIfModelChanges<> : Ésta implementación lo que hace es que que eliminará, recreará y, opcionalmente, volverá a sembrar la base de datos sólo si el modelo ha cambiado desde que se creó la base de datos.

Nota: es importante saber que la estratégia indicada anteriormente solo es útil en el proceso de desarrollo, pues si la base de datos estubiera en producción implicaría que al hacer un cambio eliminariamos todos los datos de la base de datos.

Investigue la diferencia que existen con ***migratedatabasetolatestversion*** además de sus características y propiedades.

2 AGREGANDO RESTRICCIONES AL MODELO

Las restricciones son requerimientos o formas de presentación que podemos definir desde el modelo y que afectarán el modo en el que se verán los datos. Por ejemplo haremos que en el caso de la clase mdlCity me presente como un dato requerido (obligatorio) el nombre de la ciudad para poderlo guardar. Bastará con agregar la siguiente clausula

```
[Required]
public string Name { get; set; }
```

Nota: para esto deberá agregar la clase *System.ComponentModel.DataAnnotations*

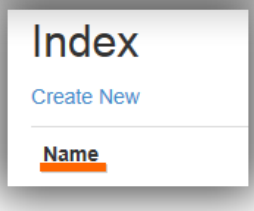
Otra de las cosas que podemos hacer desde el modelo mediante código es especificar el tamaño esperado de un campo e incluso mandar un error si se excede de esta restricción

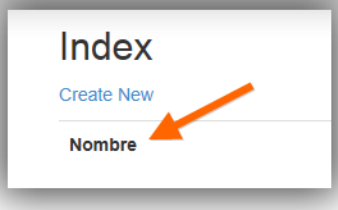
```
[Required]
[MaxLength(15, ErrorMessage ="Error: Ha excedido la cantidad de caracteres aceptados ")]
public string Name { get; set; }
```

Agregue tres restricciones más a su modelo.

Otras de las cosas que podemos hacer des de el modelo, es cambiar los nombre de las propiedades a como se presentan en el explorador. Ejemplo.

```
[Required]
[MaxLength(15, ErrorMessage ="Error: Ha excedido la cantidad de caracteres aceptados ")]
[Display(Name = "Nombre")]
public string Name { get; set; }
```





Observe el cambio…

Cambie todos los nombres de su modelo a una presentación en español.