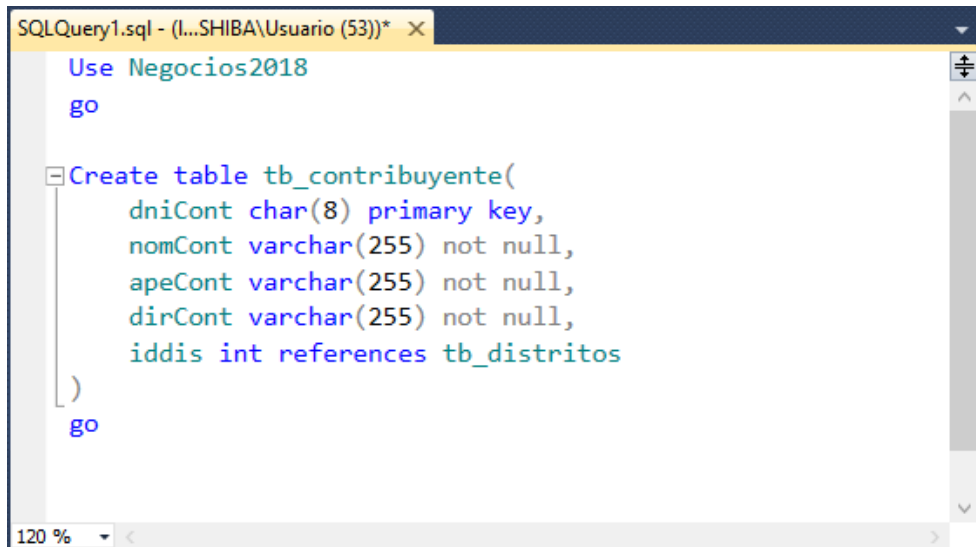


Sesión 01: Manipulación de Datos - Transacciones

Implemente un proyecto ASP.NET MVC, donde permita INSERTAR, CONSULTAR y ACTUALIZAR los datos de la tabla tb_contribuyente.

Creando la tabla tb_contribuyente

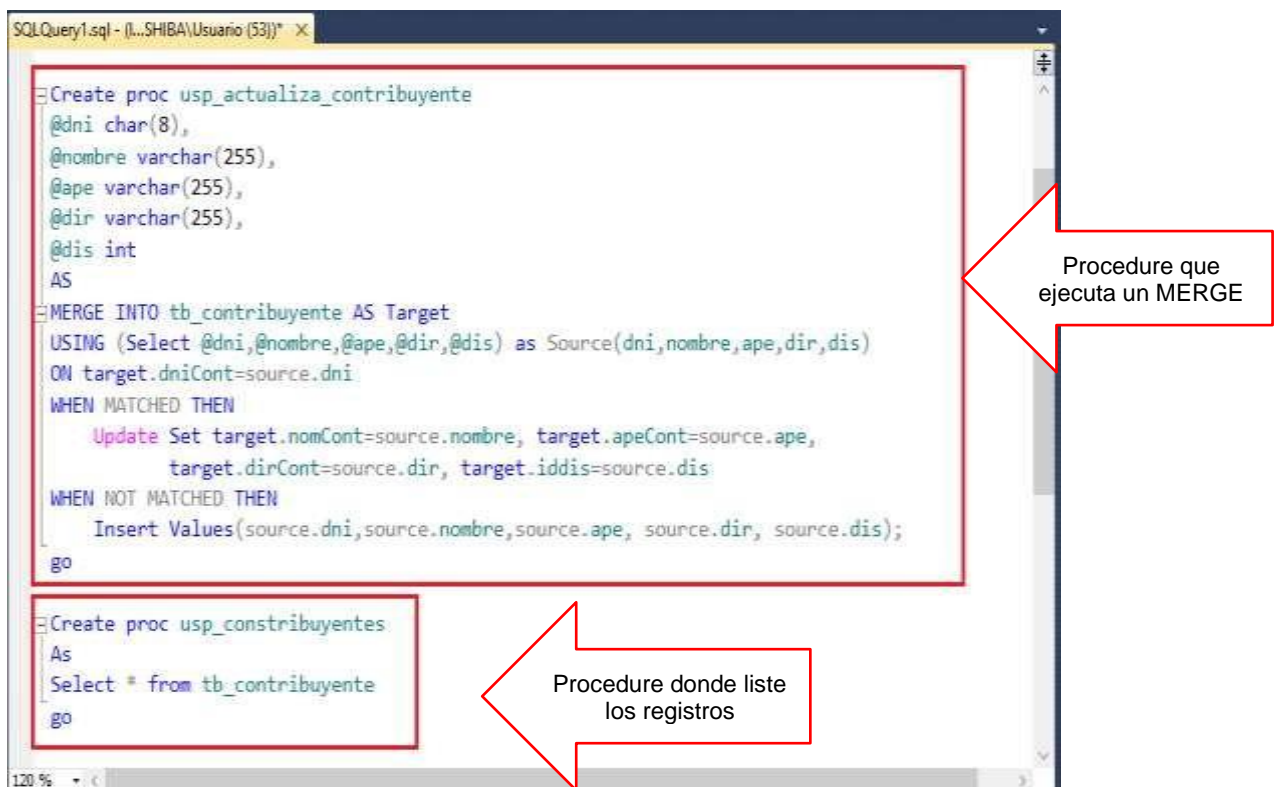
En el administrador del SQL Server, defina la tabla tb_contribuyente, tal como se muestra.



```
SQLQuery1.sql - (\\...SHIBA\\Usuario (53))* X
Use Negocios2018
go

Create table tb_contribuyente(
    dniCont char(8) primary key,
    nomCont varchar(255) not null,
    apeCont varchar(255) not null,
    dirCont varchar(255) not null,
    iddis int references tb_distritos
)
go
```

Crea los procedimientos almacenados: usp_actualiza_contribuyente, donde inserta o actualiza un registro (uso del MERGE), usp_contribuyentes, lista los registros



```
SQLQuery1.sql - (\\...SHIBA\\Usuario (53))* X

Create proc usp_actualiza_contribuyente
    @dni char(8),
    @nombre varchar(255),
    @ape varchar(255),
    @dir varchar(255),
    @dis int
AS
MERGE INTO tb_contribuyente AS Target
USING (Select @dni,@nombre,@ape,@dir,@dis) as Source(dni,nombre,ape,dir,dis)
ON target.dniCont=source.dni
WHEN MATCHED THEN
    Update Set target.nomCont=source.nombre, target.apeCont=source.ape,
            target.dirCont=source.dir, target.iddis=source.dis
WHEN NOT MATCHED THEN
    Insert Values(source.dni,source.nombre,source.ape, source.dir, source.dis);
go

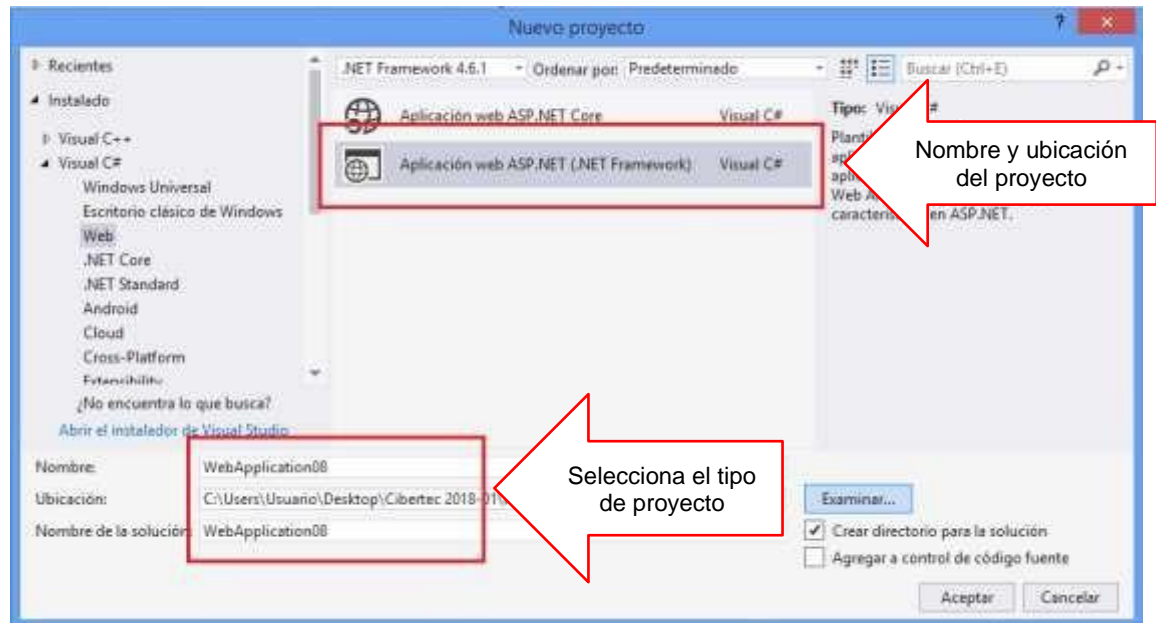
Create proc usp_contribuyentes
As
Select * from tb_contribuyente
go
```

Procedure que ejecuta un MERGE

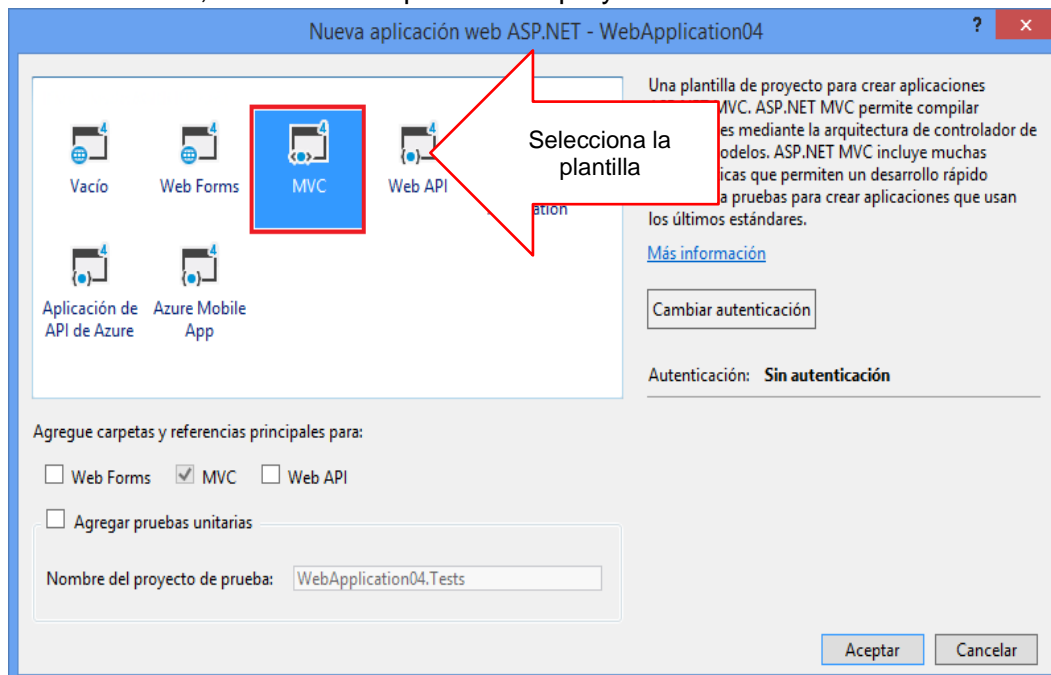
Procedure donde liste los registros

Creando el proyecto

Iniciamos Visual Studio; seleccionar el proyecto Web; selecciona la plantilla Aplicación web ASP.NET (.NET Framework), asignar el nombre y ubicación del proyecto; y presionar el botón ACEPTAR



A continuación, seleccionar la plantilla del proyecto MVC.



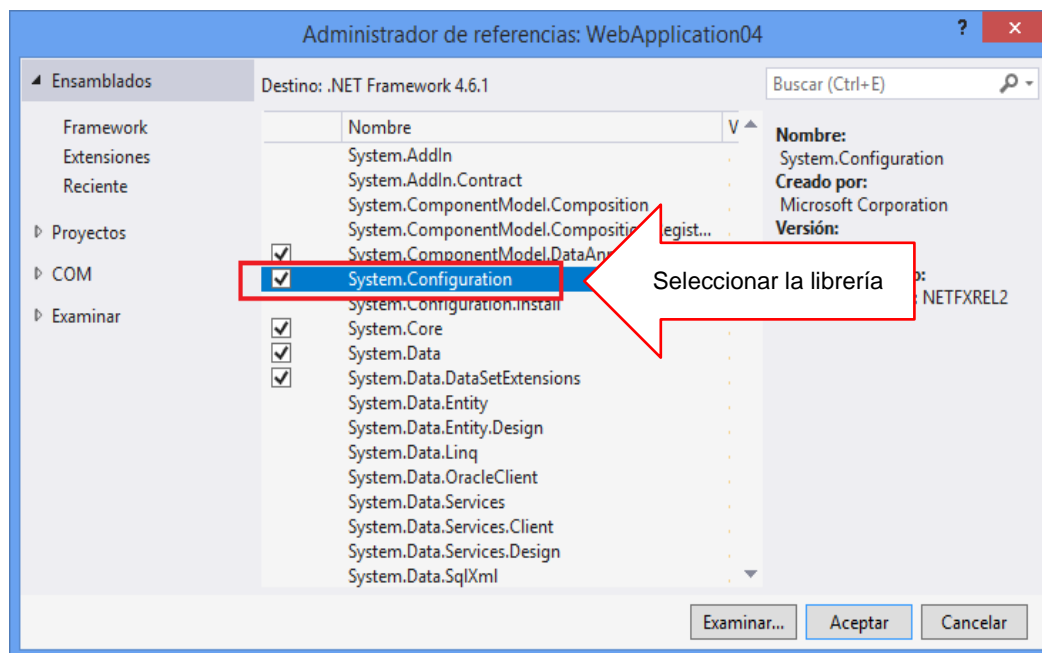
Publicando la cadena de conexión

Abrir el archivo Web.config, defina la etiqueta <connectionStrings> para agregar una cadena de conexión a la base de datos Negocios2018, tal como se muestra.



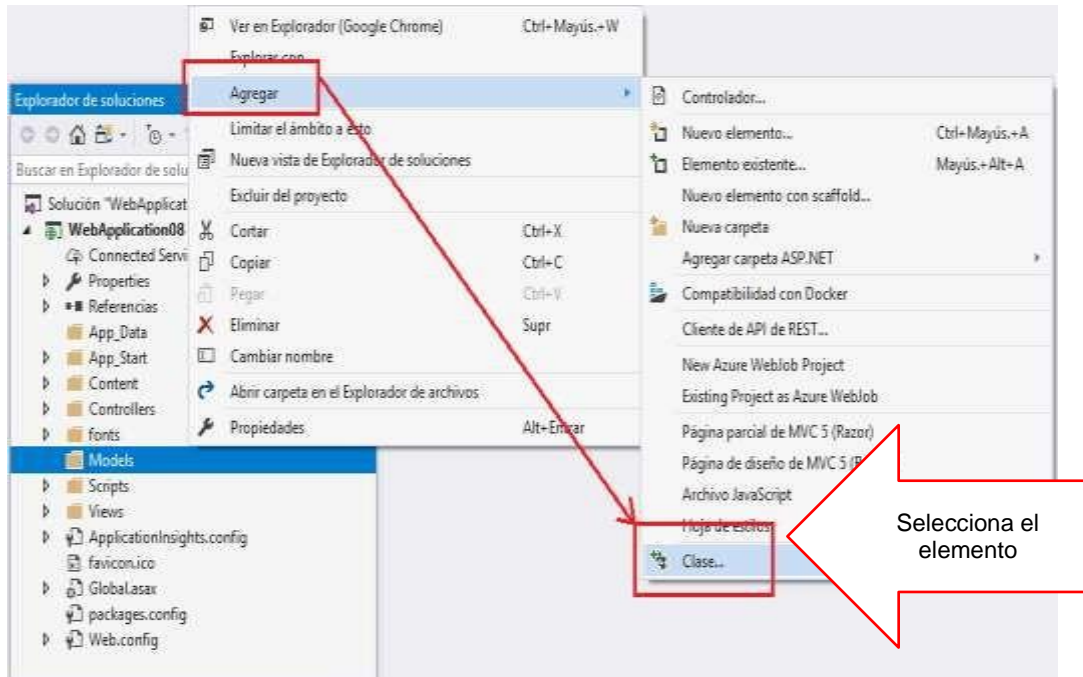
A continuación agregar la referencia al proyecto:

Selecciona desde la opción **Proyectos** → **Agregar Referencia**, selecciona la librería System.Configuration, tal como se muestra. Presiona el botón ACEPTAR.

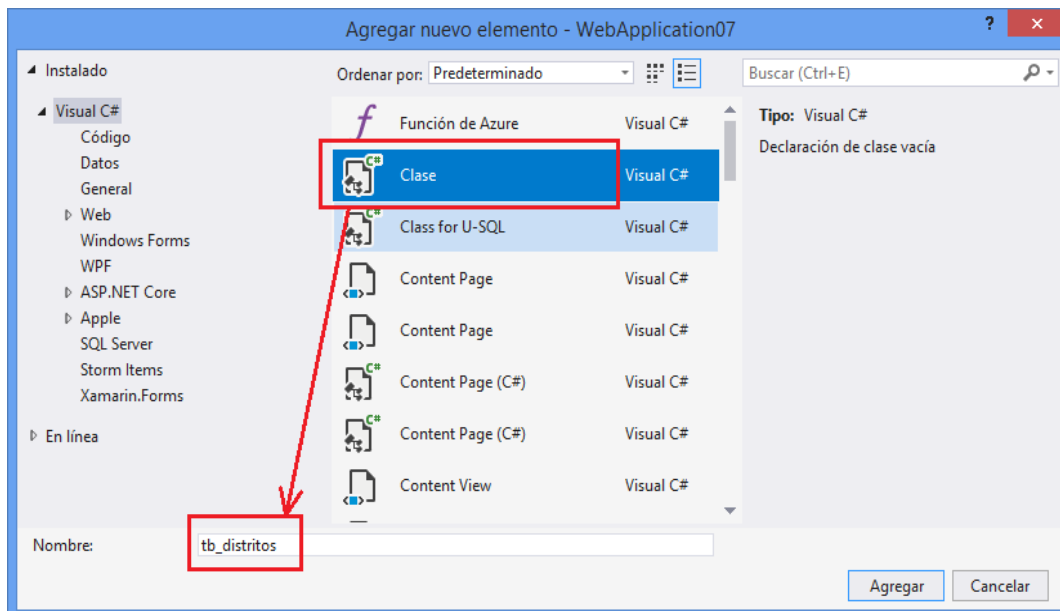


Creando las Clases del Modelo

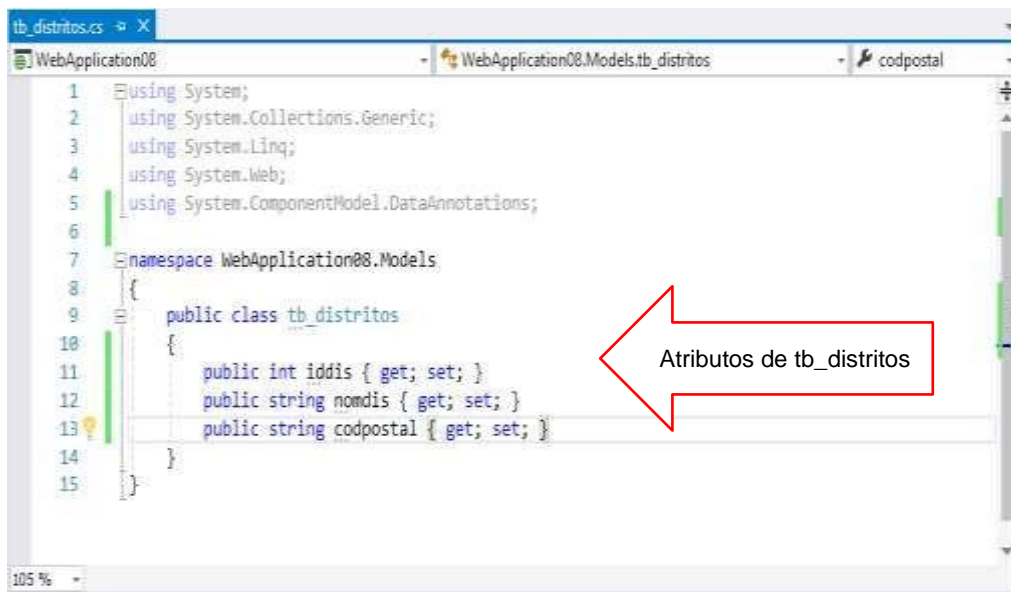
Para realizar las operaciones a la tabla tb_contribuyente y tb_distritos, agregamos, en la carpeta Models, las clases, tal como se muestra.



Asigne el nombre de la clase, la cual se llamará tb_distritos, tal como se muestra. Presiona el botón AGREGAR



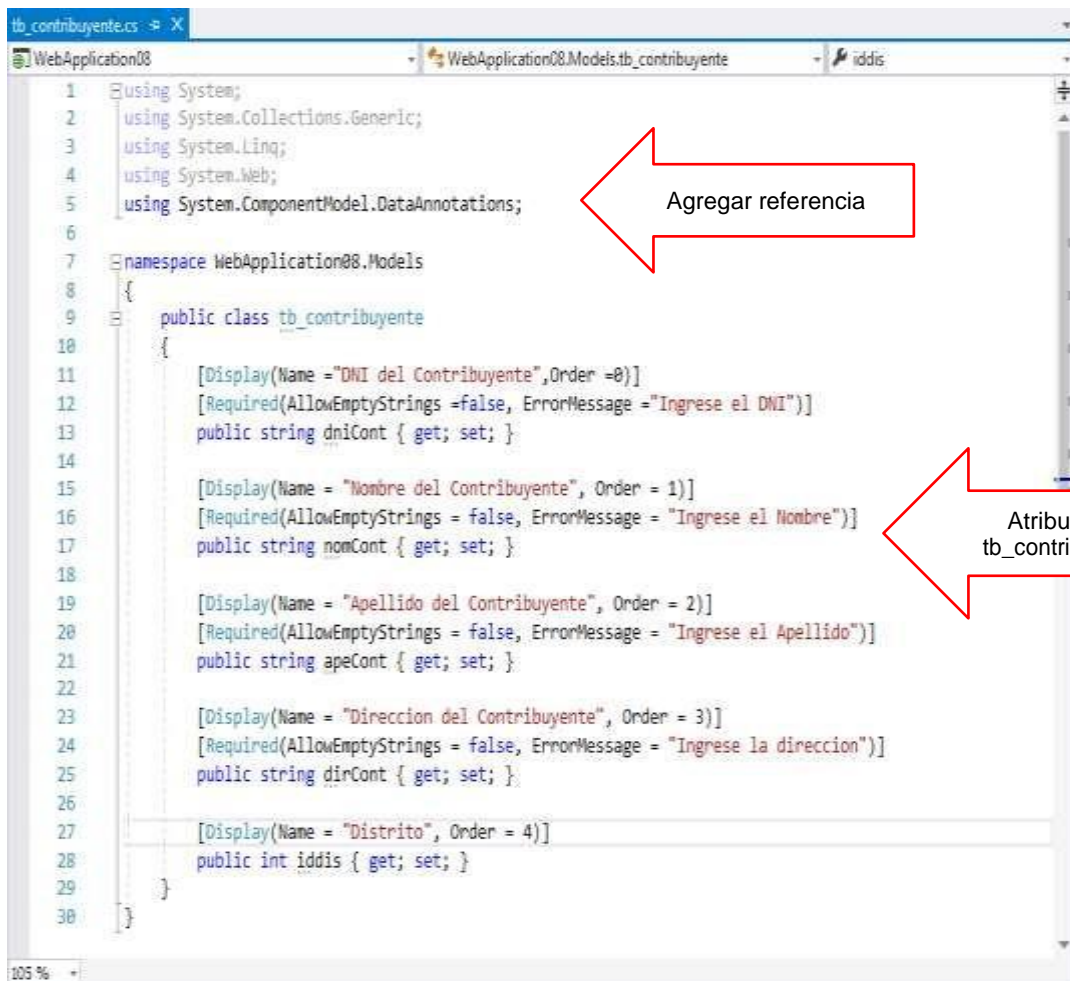
Defina los atributos de la clase tb_distritos, tal como se muestra.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.ComponentModel.DataAnnotations;
6
7 namespace WebApplication08.Models
8 {
9     public class tb_distritos
10     {
11         public int iddis { get; set; }
12         public string nomdis { get; set; }
13         public string codpostal { get; set; }
14     }
15 }
```

Atributos de tb_distritos

Defina los atributos de la clase tb_contribuyente, así como los atributos [Display] y [Required] para cada uno de los atributos



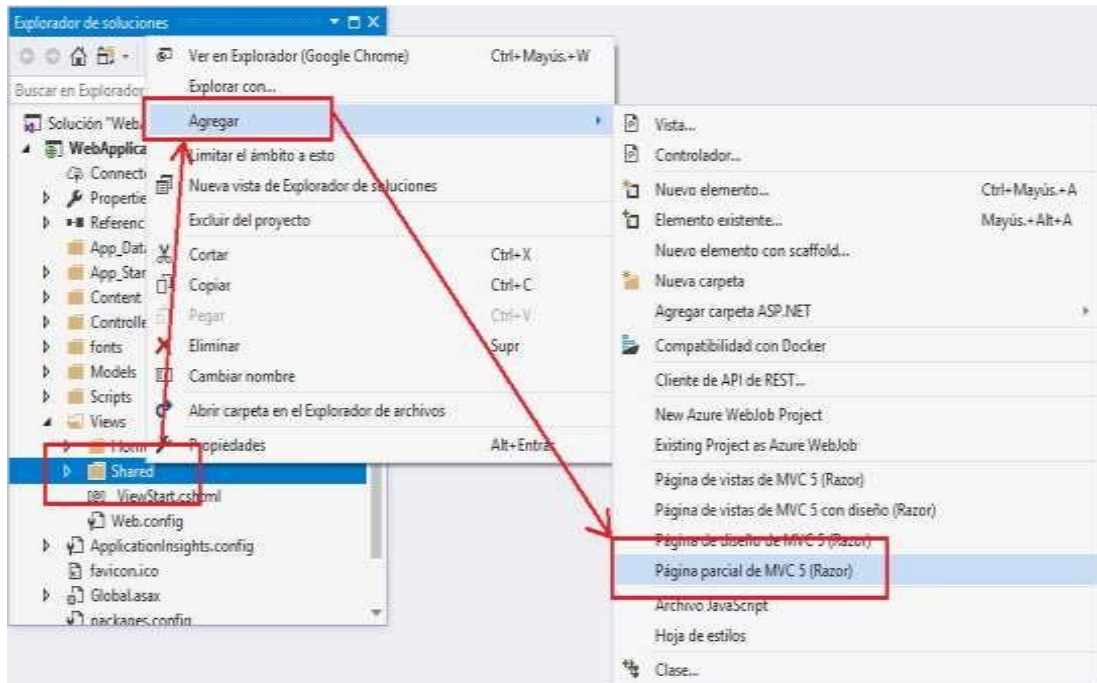
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.ComponentModel.DataAnnotations;
6
7 namespace WebApplication08.Models
8 {
9     public class tb_contribuyente
10     {
11         [Display(Name = "DNI del Contribuyente", Order = 0)]
12         [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el DNI")]
13         public string dniCont { get; set; }
14
15         [Display(Name = "Nombre del Contribuyente", Order = 1)]
16         [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el Nombre")]
17         public string nomCont { get; set; }
18
19         [Display(Name = "Apellido del Contribuyente", Order = 2)]
20         [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el Apellido")]
21         public string apeCont { get; set; }
22
23         [Display(Name = "Direccion del Contribuyente", Order = 3)]
24         [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la direccion")]
25         public string dirCont { get; set; }
26
27         [Display(Name = "Distrito", Order = 4)]
28         public int iddis { get; set; }
29     }
30 }
```

Agregar referencia

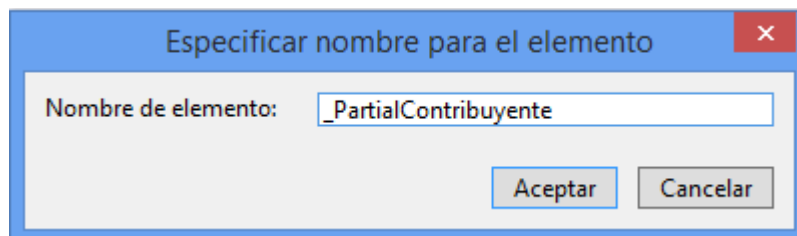
Atributos de tb_contribuyente

Agregando una Vista Parcial

Defina una Vista Parcial para listar de los registros de tb_distritos. En la carpeta **Shared**, selecciona la opción **AGREGAR** → **Página parcial de MVC 5 (Razor)**, tal como se muestra.

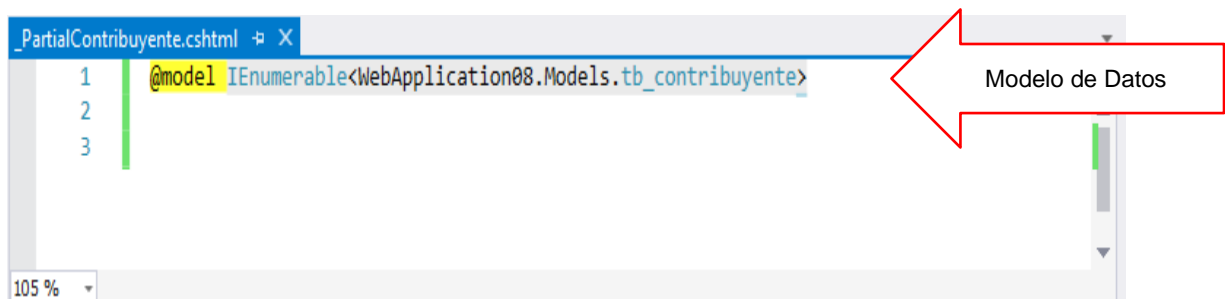


Asigne el nombre: **_PartialContribuyente**, tal como se muestra



A continuación, codifique la vista parcial:

1. Defina la estructura de datos que recibirá la vista (IEnumerable de tb_contribuyente)



2. A continuación, definimos una etiqueta **<table>**, donde visualizamos la cabecera y los registros de la consulta. En esta primera parte, dentro del **<table>** definimos la cabecera, utilizando la etiqueta **<th>**, tal como se muestra



```
1 @model IEnumerable<WebApplication08.Models.tb_contribuyente>
2
3 <table>
4 <tr>
5 <th>DNI</th>
6 <th>Nombre del Contribuyente</th>
7 <th>Apellido del Contribuyente</th>
8 <th>Direccion del Contribuyente</th>
9 <th>Distrito</th>
10 <th></th>
11 </tr>
12 </table>
13
```

Cabecera

3. Definida la cabecera, vamos a definir la estructura para listar los registros. Dentro de un **foreach**, vamos a listar cada uno de los campos del registro de la consulta, tal como se muestra. Agrega un **ActionLink**, para ejecutar el proceso Actualizar (Edit) un registro de contribuyente por su campo dniCont.



```
1 @model IEnumerable<WebApplication08.Models.tb_contribuyente>
2
3 <table>
4 <tr>
5 <th>DNI</th>
6 <th>Nombre del Contribuyente</th>
7 <th>Apellido del Contribuyente</th>
8 <th>Direccion del Contribuyente</th>
9 <th>Distrito</th>
10 <th></th>
11 </tr>
12
13 @foreach (var item in Model)
14 {
15 <tr>
16 <td>@Html.DisplayFor(modelItem => item.dniCont)</td>
17 <td>@Html.DisplayFor(modelItem => item.nomCont)</td>
18 <td>@Html.DisplayFor(modelItem => item.apeCont)</td>
19 <td>@Html.DisplayFor(modelItem => item.dirCont)</td>
20 <td>@Html.DisplayFor(modelItem => item.iddis)</td>
21 <td>@Html.ActionLink("Actualizar", "Edit", new { id=item.dniCont }, new { @class="btn btn-default" })</td>
22 </tr>
23 }
24 </table>
25
```

Lista de registros

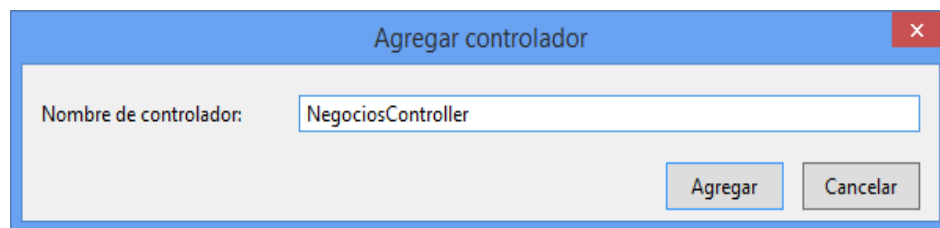
Creando un Controlador

A continuación, creamos un controlador: Desde la carpeta **Controllers**, selecciona la opción **Agregar, Controlador**.

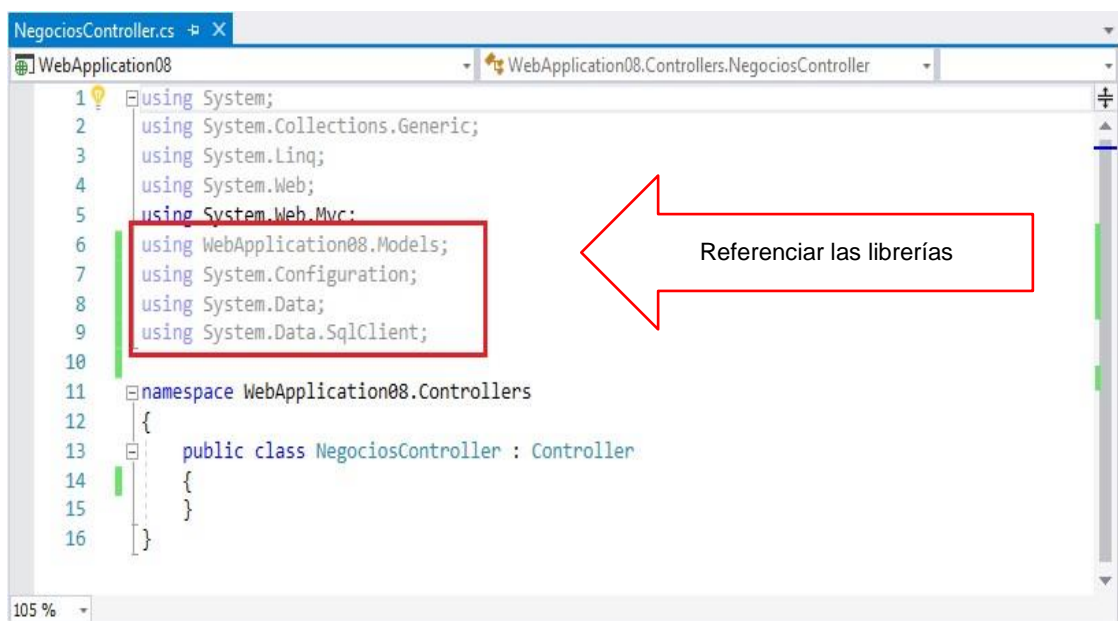
Selecciona el scaffold, Controlador de MVC5: en blanco, presiona el botón AGREGAR



Asigne el nombre del controlador: NegociosController, tal como se muestra



Referenciar la carpeta **Models**, la librería **Data.SqlClient** y el **System.Configuration** tal como se muestra.

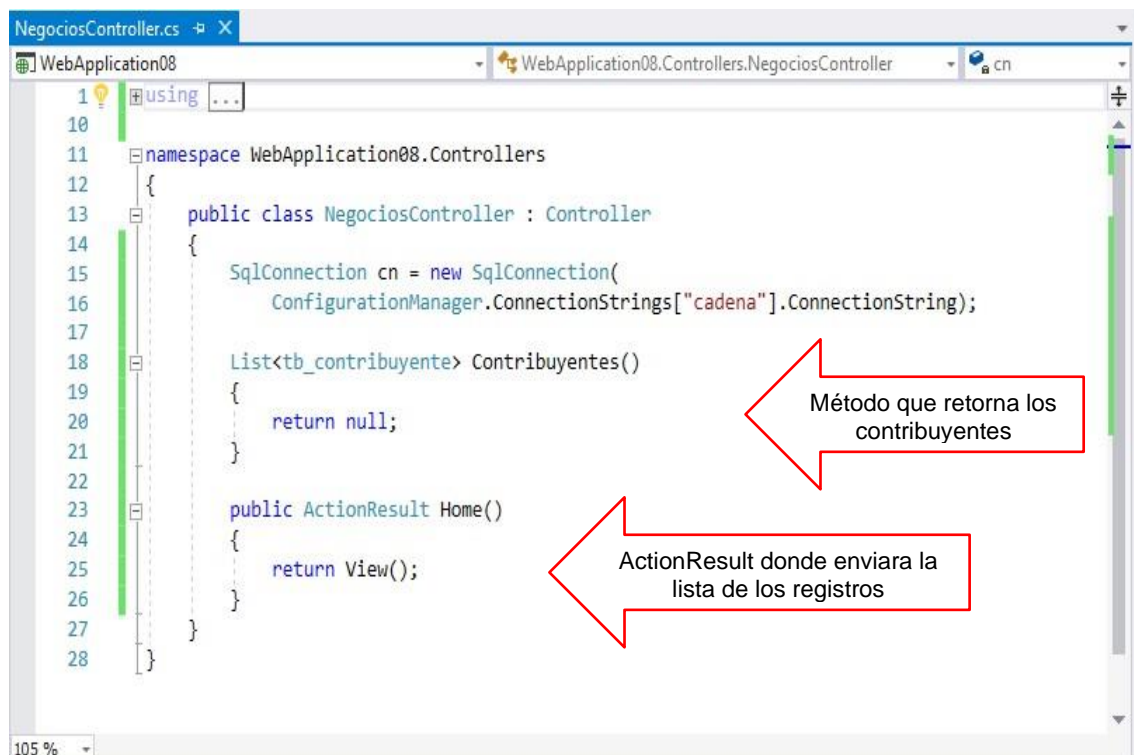


Dentro del controlador, defina la conexión a la base de datos Negocios2018 instanciando el **SqlConnection**, utilice el **ConfigurationManager**.



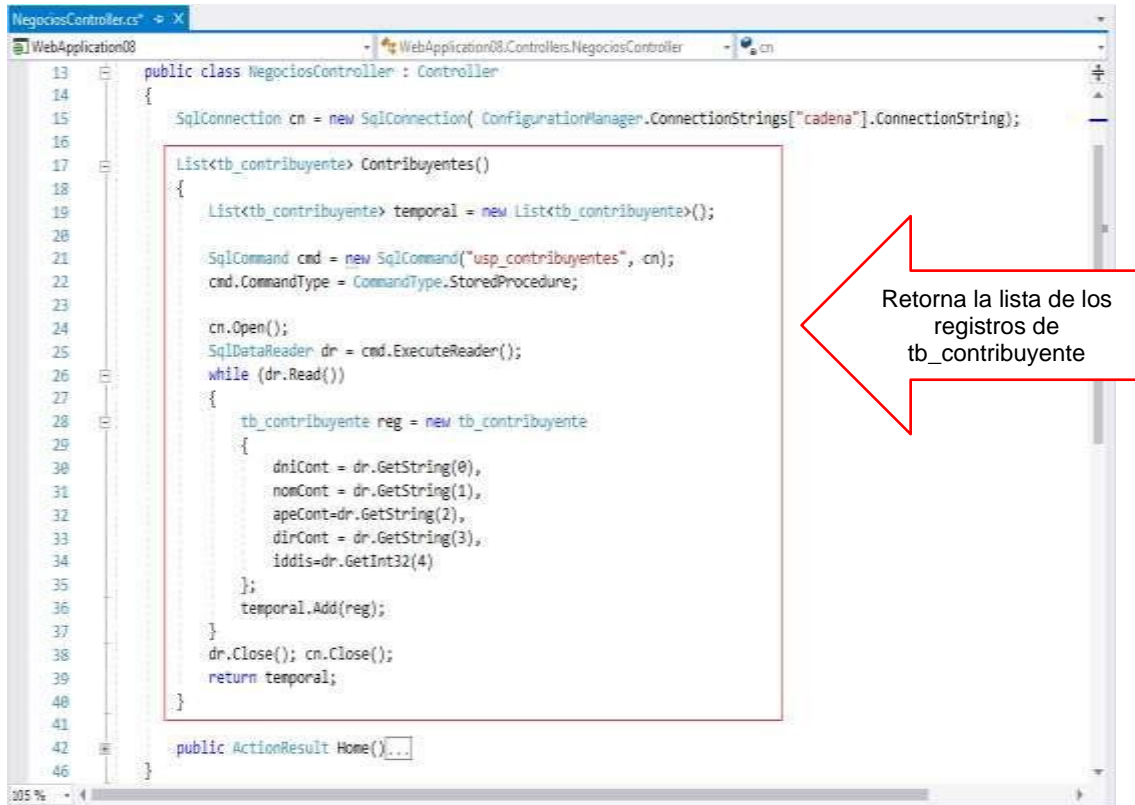
Trabajando con el ActionResult Home

1. Defina el método Contribuyentes(), el cual ejecutará el procedimiento almacenado usp_contribuyentes, retornando los registros de tb_contribuyentes.
2. Defina el ActionResult Home(), el cual enviará a la vista la lista de los registros de tb_contribuyentes.



Codifique el método llamado `Contribuyentes()`:

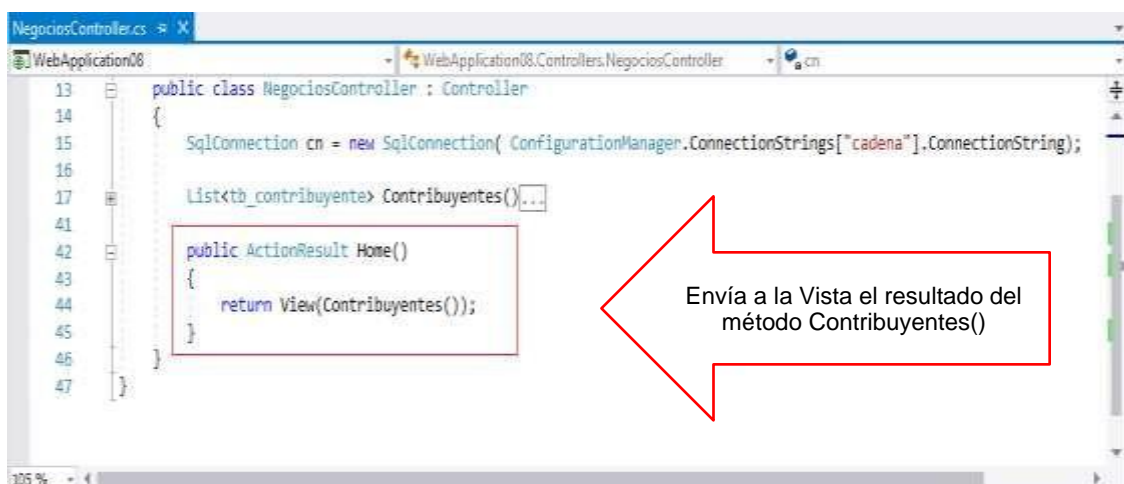
1. Ejecuta el procedure que liste los registros de `tb_contribuyente` a través del `SqlCommand`. El resultado se almacena en el `SqlDataReader`.
2. Para guardar los resultados en el temporal se realiza a través de un bucle (mientras se pueda leer: `dr.Read()`), vamos almacenando cada registro en el temporal.
3. Cerrar los objetos y enviar el objeto lista llamada temporal.



```
13 public class NegociosController : Controller
14 {
15     SqlConnection cn = new SqlConnection( ConfigurationManager.ConnectionStrings["cadena"].ConnectionString);
16
17     List<tb_contribuyente> Contribuyentes()
18     {
19         List<tb_contribuyente> temporal = new List<tb_contribuyente>();
20
21         SqlCommand cmd = new SqlCommand("usp_contribuyentes", cn);
22         cmd.CommandType = CommandType.StoredProcedure;
23
24         cn.Open();
25         SqlDataReader dr = cmd.ExecuteReader();
26         while (dr.Read())
27         {
28             tb_contribuyente reg = new tb_contribuyente
29             {
30                 dniCont = dr.GetString(0),
31                 nomCont = dr.GetString(1),
32                 apeCont = dr.GetString(2),
33                 dirCont = dr.GetString(3),
34                 iddis = dr.GetInt32(4)
35             };
36             temporal.Add(reg);
37         }
38         dr.Close(); cn.Close();
39         return temporal;
40     }
41
42     public ActionResult Home()...
```

Retorna la lista de los registros de `tb_contribuyente`

Defina el `ActionResult Home()`, el cual enviará a la Vista el resultado del método `Contribuyentes()`, tal como se muestra.



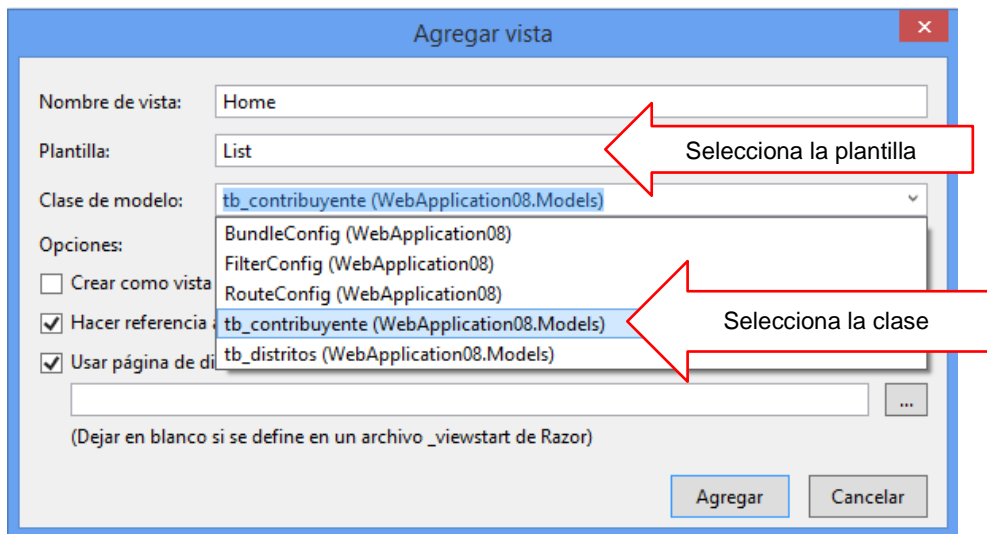
```
13 public class NegociosController : Controller
14 {
15     SqlConnection cn = new SqlConnection( ConfigurationManager.ConnectionStrings["cadena"].ConnectionString);
16
17     List<tb_contribuyente> Contribuyentes()...
41
42     public ActionResult Home()
43     {
44         return View(Contribuyentes());
45     }
46
47 }
```

Envía a la Vista el resultado del método `Contribuyentes()`

Agregando la Vista.

En el ActionResult, agrega la vista.

En dicha ventana, selecciona la **plantilla**, la cual será **List**; y la **clase de modelo** la cual es **tb_contribuyente**, tal como se muestra



Modifica la Vista:

1. Agregar un ActionLink el cual ejecutará el ActionResult Create.
2. Agrega la vista parcial llamado `_partialContribuyente`

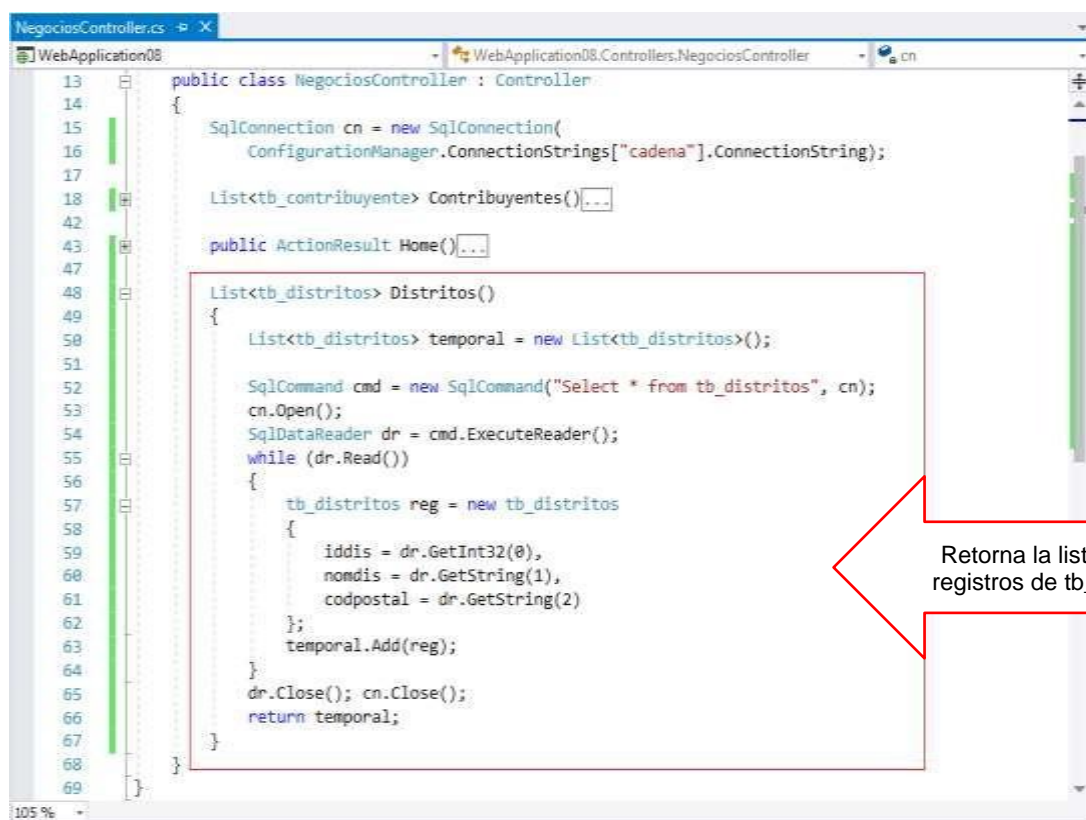


Ejecutamos la Vista, visualizamos los registros de la tabla tb_contribuyentes.



Trabajando con el ActionResult Create

Como primer paso, definimos un método que retorna la lista de los registros de tb_distritos, el cual permitirá seleccionar el distrito donde reside el contribuyente.



En el controlador Negocios, defina el ActionResult **Nuevo()**, de tipo Get y Post, tal como se muestra.

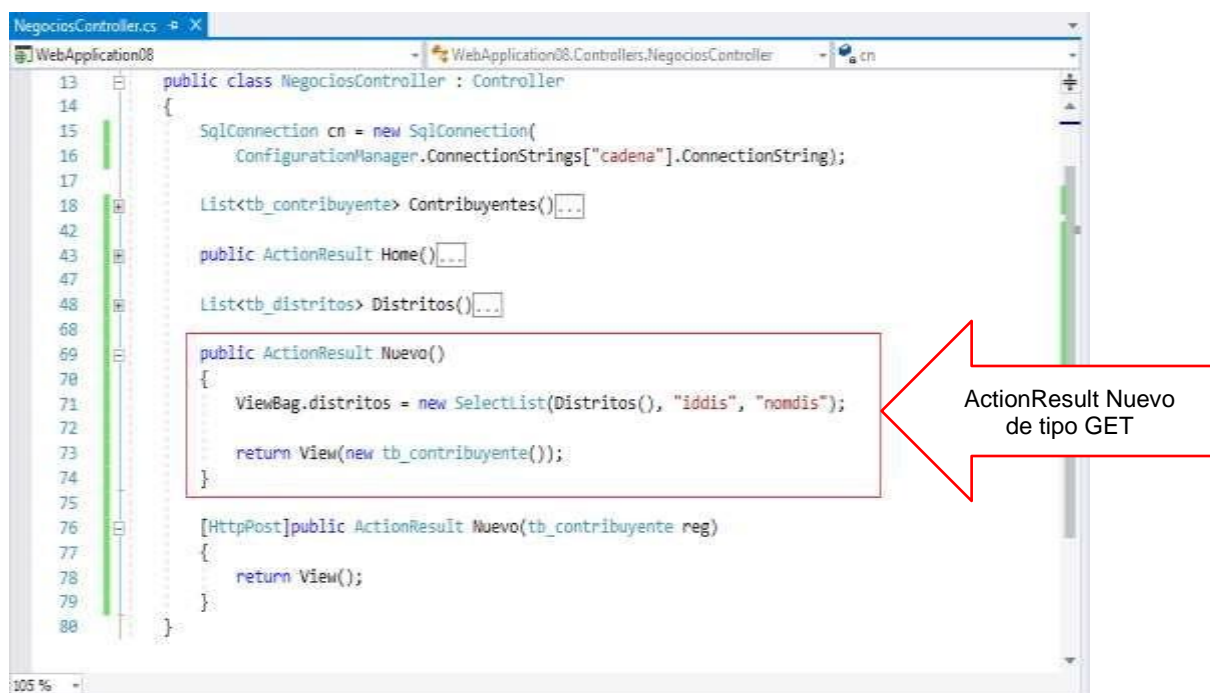


```
13 public class NegociosController : Controller
14 {
15     SqlConnection cn = new SqlConnection(
16         ConfigurationManager.ConnectionStrings["cadena"].ConnectionString);
17
18     List<tb_contribuyente> Contribuyentes() {...}
42
43     public ActionResult Home() {...}
47
48     List<tb_distritos> Distritos() {...}
68
69     public ActionResult Nuevo()
70     {
71         return View();
72     }
73
74     [HttpPost] public ActionResult Nuevo(tb_contribuyente reg)
75     {
76         return View();
77     }
78 }
```

Nuevo de tipo Get, envía los datos a la Vista

Nuevo tipo Post, recibe los datos y ejecuta un proceso

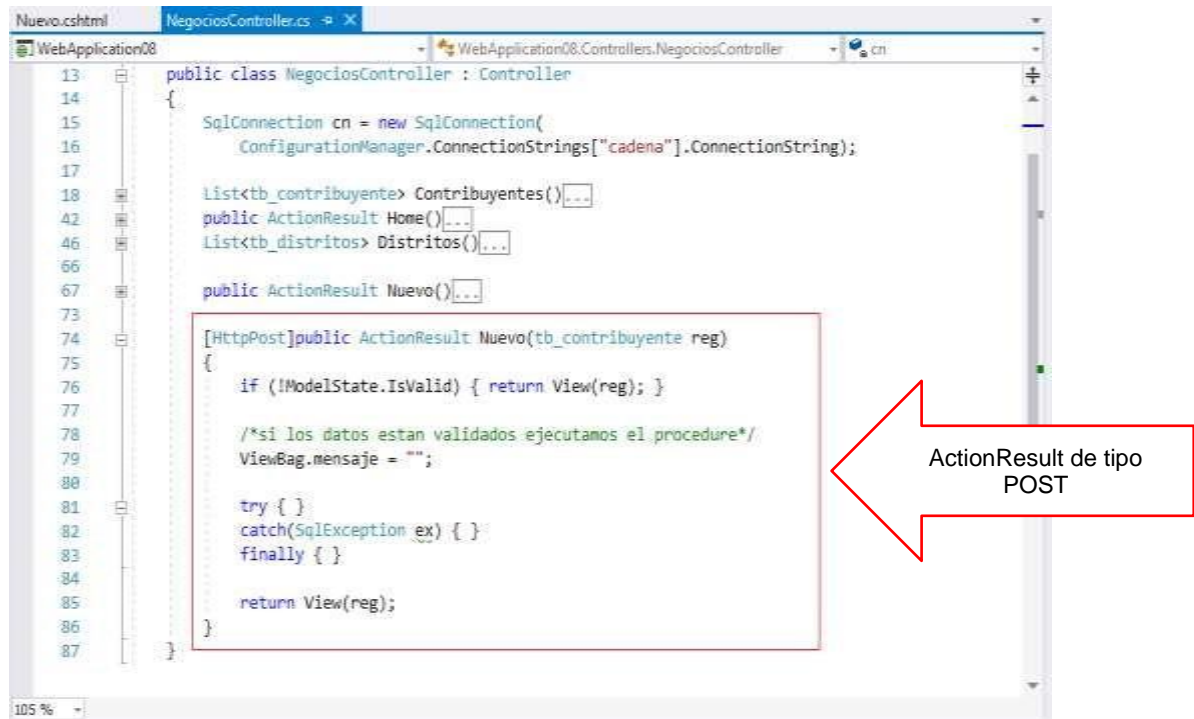
El **ActionResult Nuevo** de tipo GET, almaceno los distritos en el ViewBag.distritos; envía a la vista un nuevo registro de tb_contribuyente, tal como se muestra.



```
13 public class NegociosController : Controller
14 {
15     SqlConnection cn = new SqlConnection(
16         ConfigurationManager.ConnectionStrings["cadena"].ConnectionString);
17
18     List<tb_contribuyente> Contribuyentes() {...}
42
43     public ActionResult Home() {...}
47
48     List<tb_distritos> Distritos() {...}
68
69     public ActionResult Nuevo()
70     {
71         ViewBag.distritos = new SelectList(Distritos(), "iddis", "nomdis");
72         return View(new tb_contribuyente());
73     }
74
75     [HttpPost] public ActionResult Nuevo(tb_contribuyente reg)
76     {
77         return View();
78     }
79
80 }
```

ActionResult Nuevo de tipo GET

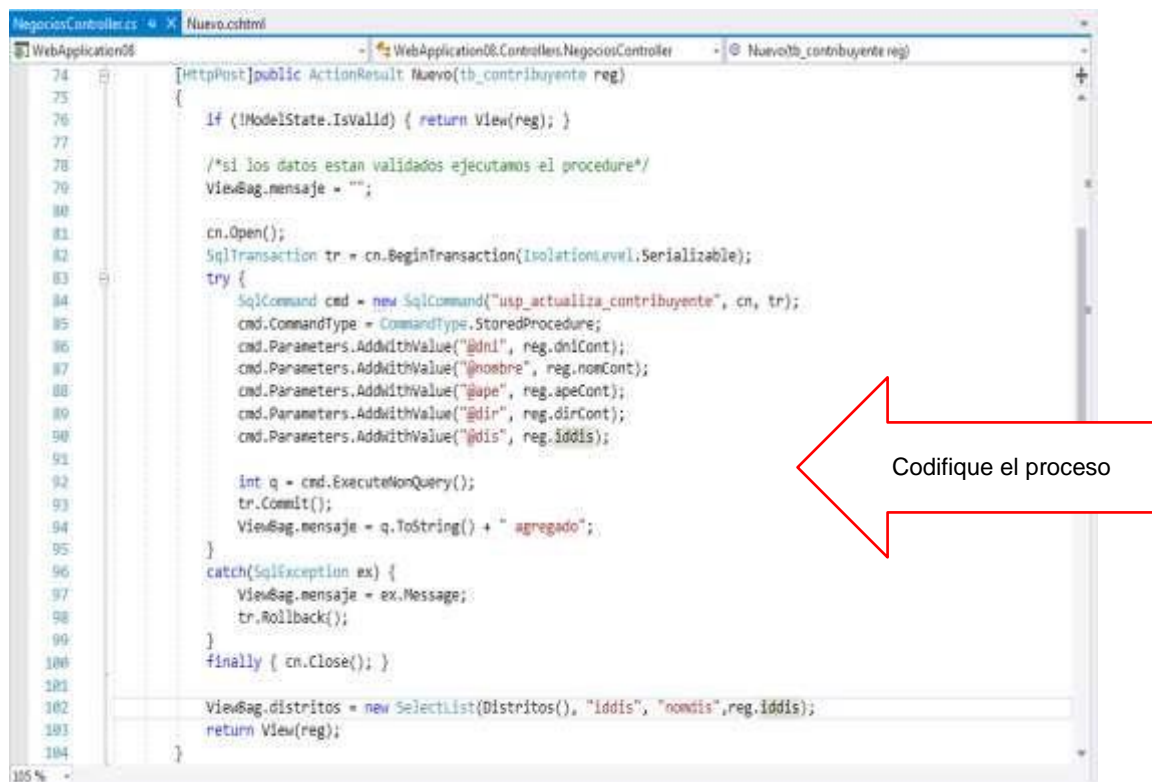
El **ActionResult Nuevo** de tipo POST, recibe los datos de la vista, los valida; si está OK, procederá a ejecutar el procedure, retornando un mensaje y los datos del registro agregado, tal como se muestra



```
13 public class NegociosController : Controller
14 {
15     SqlConnection cn = new SqlConnection(
16         ConfigurationManager.ConnectionStrings["cadena"].ConnectionString);
17
18     List<tb_contribuyente> Contribuyentes()...
19
20     public ActionResult Home()...
21
22     List<tb_distritos> Distritos()...
23
24     public ActionResult Nuevo()...
25
26     [HttpPost] public ActionResult Nuevo(tb_contribuyente reg)
27     {
28         if (!ModelState.IsValid) { return View(reg); }
29
30         /*si los datos estan validados ejecutamos el procedure*/
31         ViewBag.mensaje = "";
32
33         try { }
34         catch (SqlException ex) { }
35         finally { }
36
37         return View(reg);
38     }
39 }
```

ActionResult de tipo POST

Proceso del **ActionResult Nuevo** tipo POST, ejecutando un procedimiento almacenando y controlando el proceso por un **Transaction**, tal como se muestra



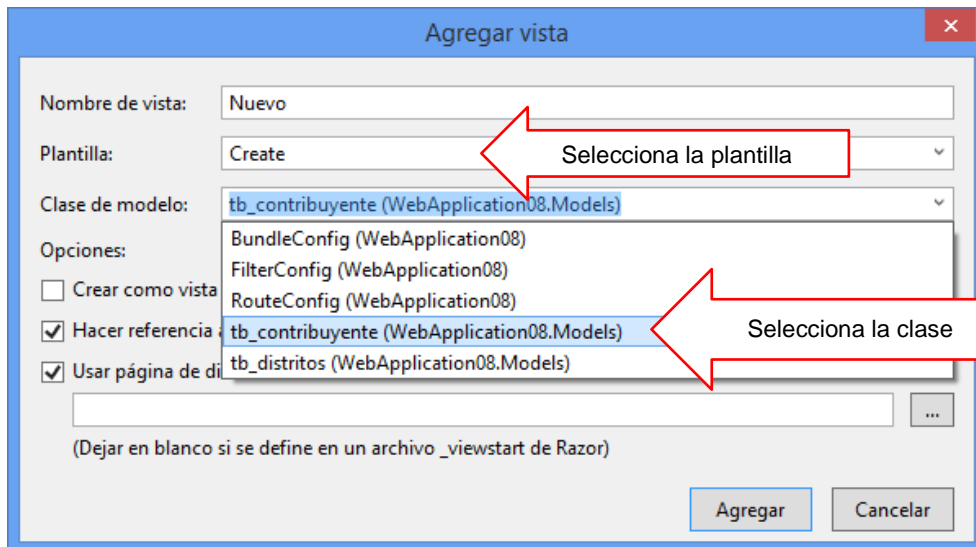
```
74 [HttpPost] public ActionResult Nuevo(tb_contribuyente reg)
75 {
76     if (!ModelState.IsValid) { return View(reg); }
77
78     /*si los datos estan validados ejecutamos el procedure*/
79     ViewBag.mensaje = "";
80
81     cn.Open();
82     SqlTransaction tr = cn.BeginTransaction(IsolationLevel.Serializable);
83     try {
84         SqlCommand cmd = new SqlCommand("usp_actualiza_contribuyente", cn, tr);
85         cmd.CommandType = CommandType.StoredProcedure;
86         cmd.Parameters.AddWithValue("@dni", reg.dniCont);
87         cmd.Parameters.AddWithValue("@nombre", reg.nomCont);
88         cmd.Parameters.AddWithValue("@ape", reg.apeCont);
89         cmd.Parameters.AddWithValue("@dir", reg.dirCont);
90         cmd.Parameters.AddWithValue("@dis", reg.iddis);
91
92         int q = cmd.ExecuteNonQuery();
93         tr.Commit();
94         ViewBag.mensaje = q.ToString() + " agregado";
95     }
96     catch (SqlException ex) {
97         ViewBag.mensaje = ex.Message;
98         tr.Rollback();
99     }
100     finally { cn.Close(); }
101
102     ViewBag.distritos = new SelectList(Distritos(), "iddis", "nomdis", reg.iddis);
103     return View(reg);
104 }
```

Codifique el proceso

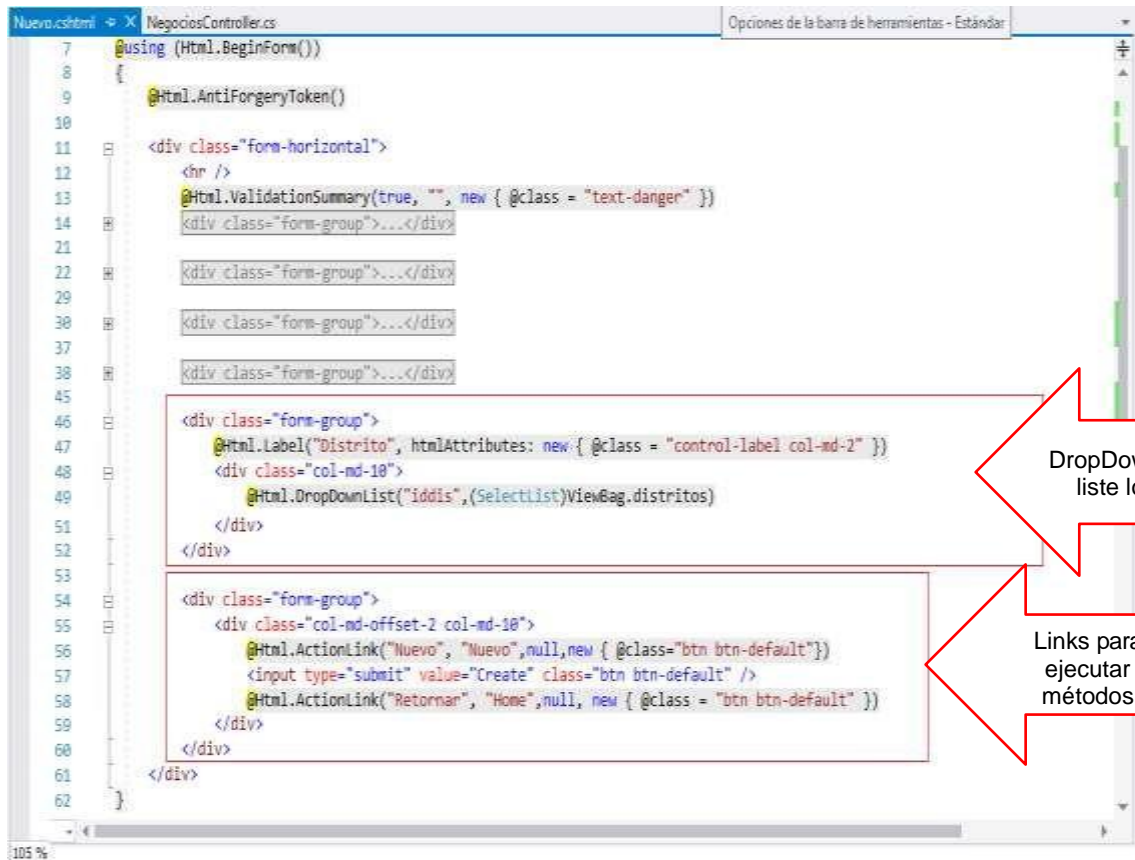
Agregando la Vista.

En el ActionResult, agrega la vista.

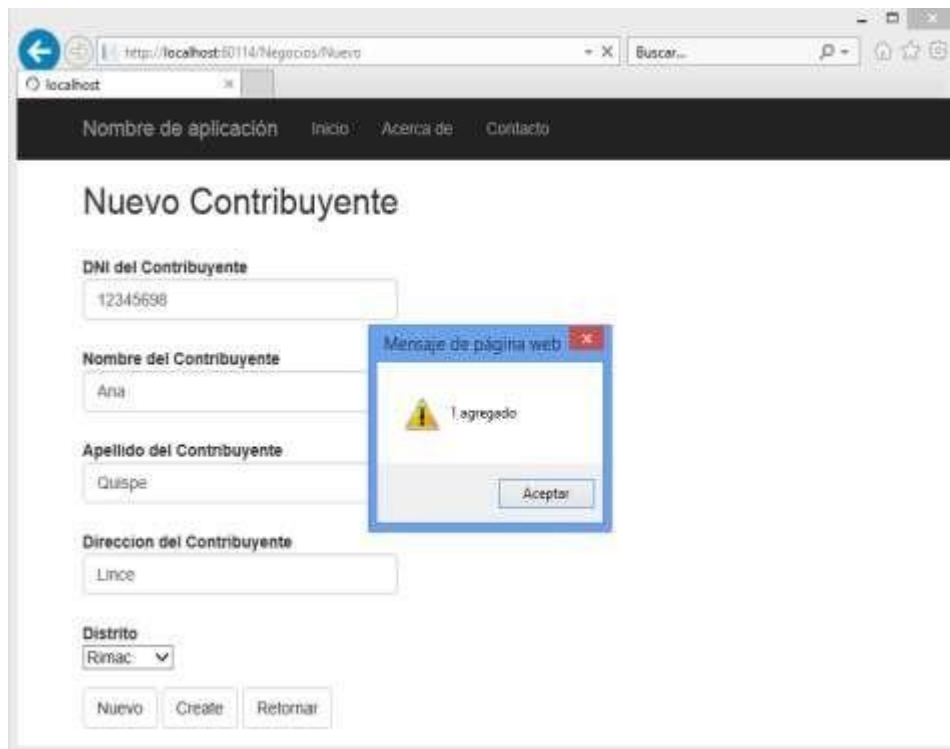
En dicha ventana, selecciona la **plantilla**, la cual será **Create**; y la **clase de modelo** la cual es **tb_contribuyente**, tal como se muestra



Modifique el diseño de la vista: en el campo iddis, cambiar por DropDownList, para listar los distritos; agrupe y diseñe los Links de los procesos: Nuevo, Create, Retornar, tal como se muestra



Ejecuta la Vista, ingresa los datos, al presionar el botón CREATE, agregamos un registro y visualizamos un mensaje.
Para retornar, presione el botón RETORNAR



The screenshot shows a web browser window at the URL `http://localhost:50114/Negocios/Nuevo`. The page has a navigation bar with links: 'Nombre de aplicación', 'Inicio', 'Acerca de', and 'Contacto'. The main heading is 'Nuevo Contribuyente'. Below it are several input fields: 'DNI del Contribuyente' (containing '12345698'), 'Nombre del Contribuyente' (containing 'Ana'), 'Apellido del Contribuyente' (containing 'Quispe'), 'Direccion del Contribuyente' (containing 'Lince'), and a 'Distrito' dropdown menu (set to 'Romac'). At the bottom are three buttons: 'Nuevo', 'Create', and 'Retornar'. A modal dialog box titled 'Mensaje de página web' is overlaid on the form, displaying a yellow warning icon and the text '¡agregado', with an 'Aceptar' button.

Trabajando con el ActionResult Edit

En el controlador Negocios, defina el ActionResult **Edit** GET y POST, tal como se muestra



```
13 public class NegociosController : Controller
14 {
15     SqlConnection cn = new SqlConnection(
16         ConfigurationManager.ConnectionStrings["cadena"].ConnectionString);
17
18     List<tb_contribuyente> Contribuyentes()...
42     public ActionResult Home()...
46     List<tb_distritos> Distritos()...
66
67     public ActionResult Nuevo()...
73     [HttpPost]public ActionResult Nuevo(tb_contribuyente reg)...
```

Annotations in the code editor:

- A red box highlights the `public ActionResult Edit(int id)` method (lines 105-108). A red arrow points to it from a label 'Action método GET'.
- A red box highlights the `[HttpPost]public ActionResult Edit(tb_contribuyente reg)` method (lines 110-113). A red arrow points to it from a label 'Action método POST'.

Codifique el **ActionResult Edit** (int id) método GET, donde filtra y envía el registro de la tabla tb_contribuyente filtrado por su campo dniCont; además enviamos a la vista la lista de distritos a través del ViewBag.distritos, tal como se muestra

```
104 public ActionResult Edit(string id)
105 {
106     tb_contribuyente reg = Contribuyentes().Where(c => c.dniCont == id).FirstOrDefault();
107     ViewBag.distritos = new SelectList(Distritos(), "iddis", "nomdis", reg.iddis);
108     return View(reg);
109 }
110
111 [HttpPost]public ActionResult Edit(tb_contribuyente reg)...
```

Parámetro id (enviado desde el ActionLink del Actualizar)

El **ActionResult Edit** de tipo POST, recibe los datos de la vista, los valida; si está OK, procederá a ejecutar el comando UPDATE, retornando un mensaje y los datos del registro agregado, tal como se muestra

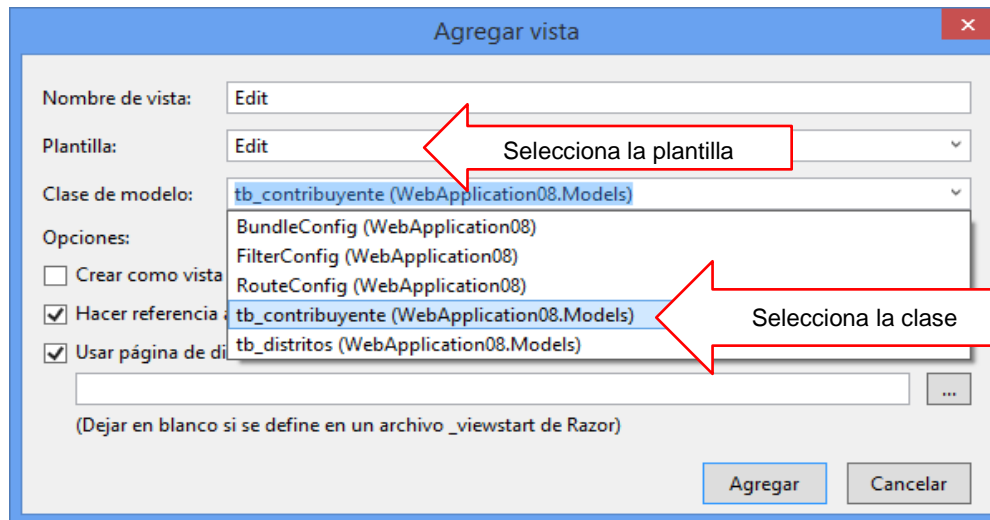
```
113 [HttpPost]public ActionResult Edit(tb_contribuyente reg)
114 {
115     if (!ModelState.IsValid) { return View(reg); }
116
117     /*si los datos estan validados ejecutamos el procedure*/
118     ViewBag.mensaje = "";
119
120     cn.Open();
121     SqlTransaction tr = cn.BeginTransaction(IsolationLevel.Serializable);
122     try{
123         SqlCommand cmd = new SqlCommand("usp_actualiza_contribuyente", cn, tr);
124         cmd.CommandType = CommandType.StoredProcedure;
125         cmd.Parameters.AddWithValue("@dni", reg.dniCont);
126         cmd.Parameters.AddWithValue("@nonbre", reg.nonCont);
127         cmd.Parameters.AddWithValue("@ape", reg.apeCont);
128         cmd.Parameters.AddWithValue("@dir", reg.dirCont);
129         cmd.Parameters.AddWithValue("@dis", reg.iddis);
130
131         int q = cmd.ExecuteNonQuery();
132         tr.Commit();
133         ViewBag.mensaje = q.ToString() + " actualizado";
134     }
135     catch (SqlException ex){
136         ViewBag.mensaje = ex.Message;
137         tr.Rollback();
138     }
139     finally { cn.Close(); }
140     ViewBag.distritos = new SelectList(Distritos(), "iddis", "nomdis", reg.iddis);
141     return View(reg);
142 }
143
```

Método POST, recupero el registro y actualizo los datos

Agregando la Vista.

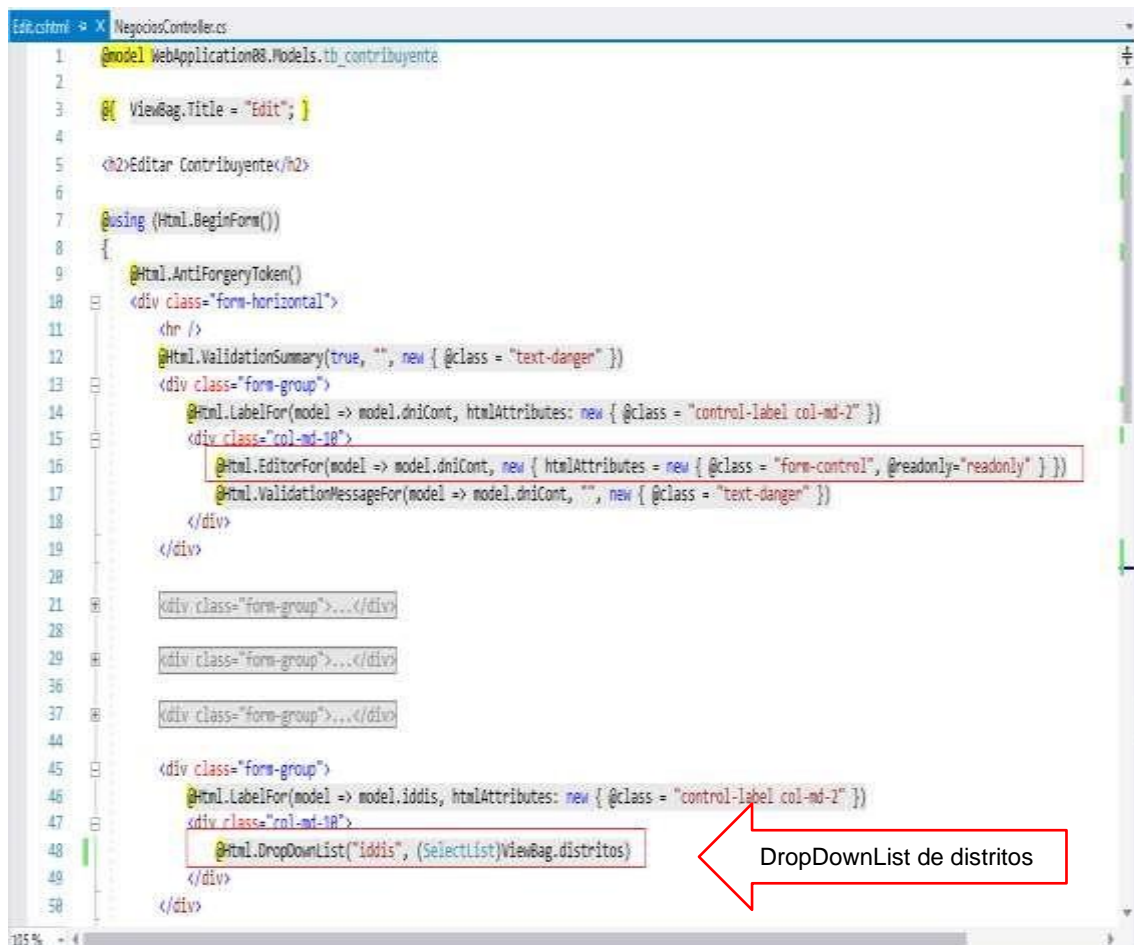
En el ActionResult, agrega la vista.

En dicha ventana, selecciona la **plantilla**, la cual será **Edit**; y la **clase de modelo** la cual es **tb_contribuyente**, tal como se muestra



Modifique el diseño de la Vista

Para que el control `EditorFor` del `dniCont` sea solo lectura, agregar la propiedad `@readonly="readonly"`. Modifique el campo `iddis` por un `DropDownList` para listar los registros de `tb_distritos`, tal como se muestra



Modifique el diseño del ActionLink Retornar, tal como se muestra

```

1  Edit:cshtml  X  NegociosControler.cs
2
3  7  using (Html.BeginForm())
4  {
5  8  {
6  9      @Html.AntiForgeryToken()
7  10     <div class="form-horizontal">
8  11         <hr />
9  12         @Html.ValidationSummary(true, "", new { @class = "text-danger" })
10 13         <div class="form-group">...</div>
11 20
12 21         <div class="form-group">...</div>
13 25
14 26         <div class="form-group">...</div>
15 36
16 37         <div class="form-group">...</div>
17 44
18 45         <div class="form-group">...</div>
19 51
20 52         <div class="form-group">
21 53             <div class="col-md-offset-2 col-md-10">
22 54                 <input type="submit" value="Actualizar" class="btn btn-default" />
23 55                 @Html.ActionLink("Retornar", "Home", null, new { @class = "btn btn-default" })
24 56             </div>
25 57         </div>
26 58     </div>
27 59 }
28 60
29 61 @section Scripts { @Scripts.Render("~/bundles/jqueryval") }
30 62
31 63 <script>
32 64     if (@ViewBag.mensaje != "") alert('@ViewBag.mensaje')
33 65 </script>

```


ActionLINK a modificar


<script> para visualizar el mensaje

Ejecuta la vista, selecciona el Contribuyente, donde visualizamos los datos del registro en la vista Edit.

Modifique los datos, al presionar el botón SAVE, se visualiza un mensaje indicando que se ha actualizado el registro.

