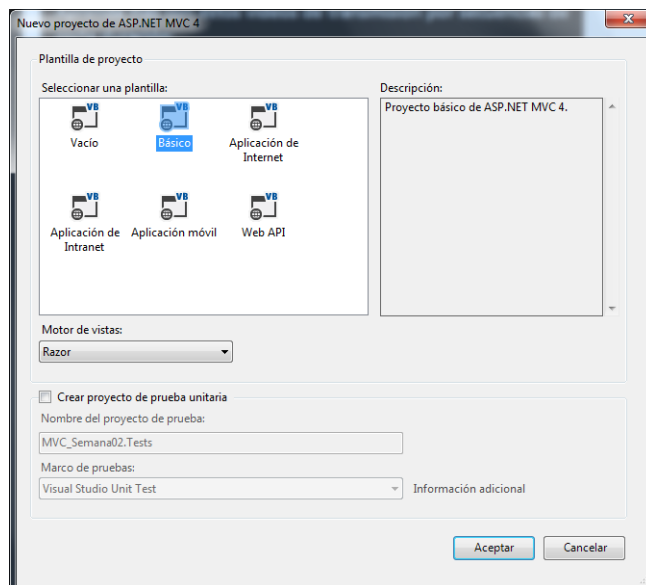
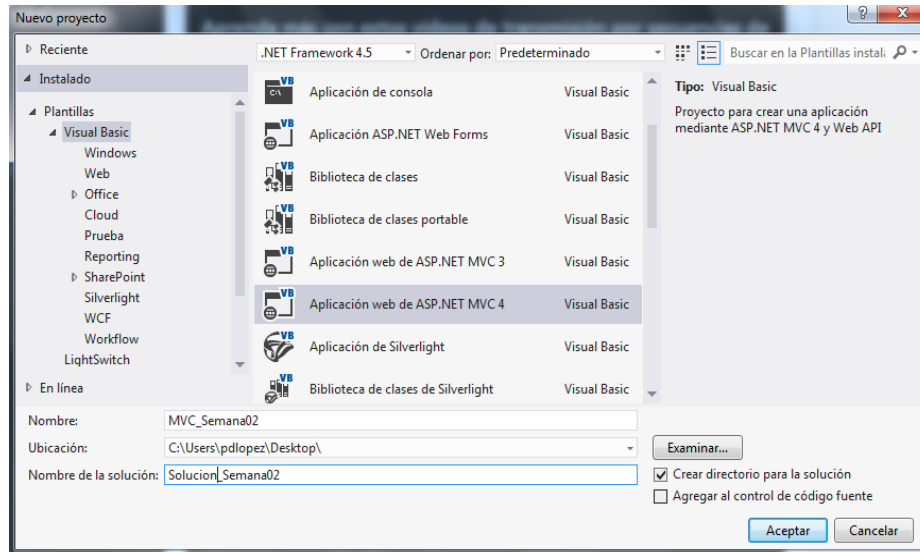


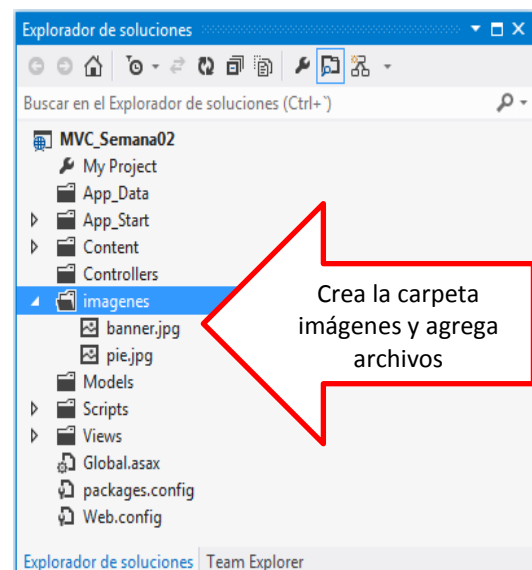
Trabajando con Anotaciones y Validaciones

Creamos el proyecto ASP.NET MVC 4 llamado MVC_Semana02, tal como se muestra

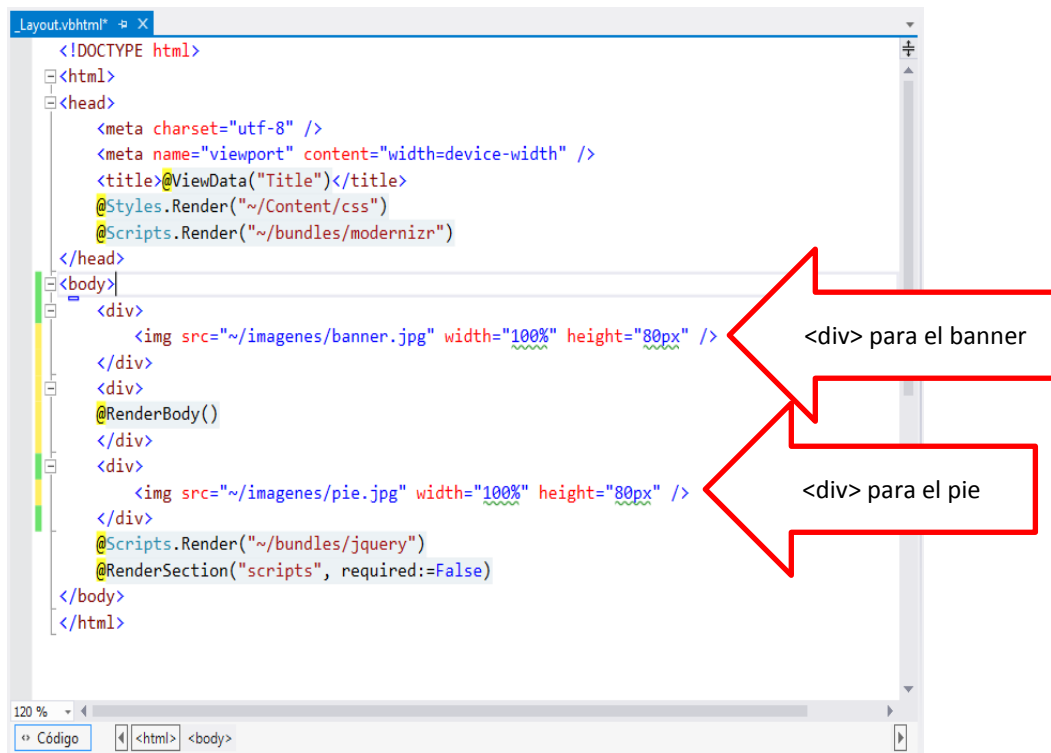


Seleccionar la plantilla BASICO y el motor de vista RAZOR y presiona el botón ACEPTAR

Crear en el proyecto una carpeta llamada IMÁGENES y agrega dos archivo: banner.jpg y pie.jpg, tal como se muestra

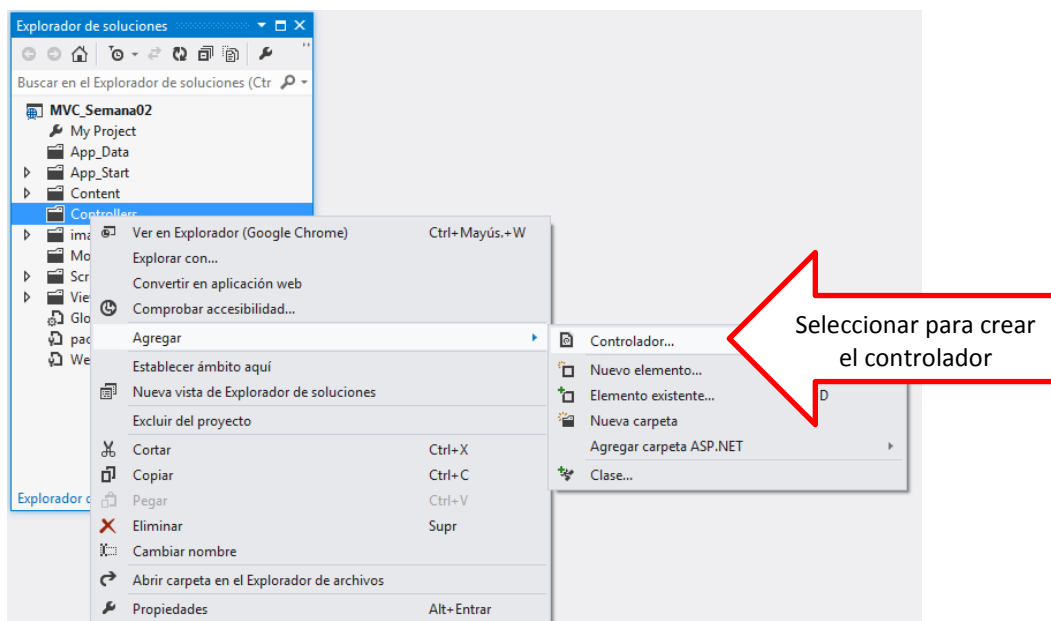


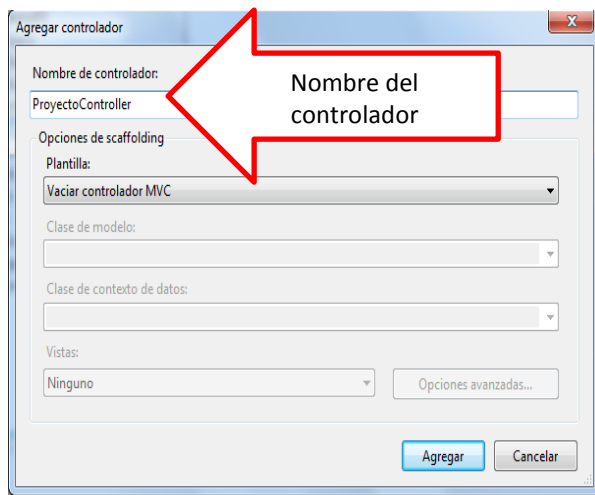
Defina la siguiente estructura en el _Layout, tal como se muestra



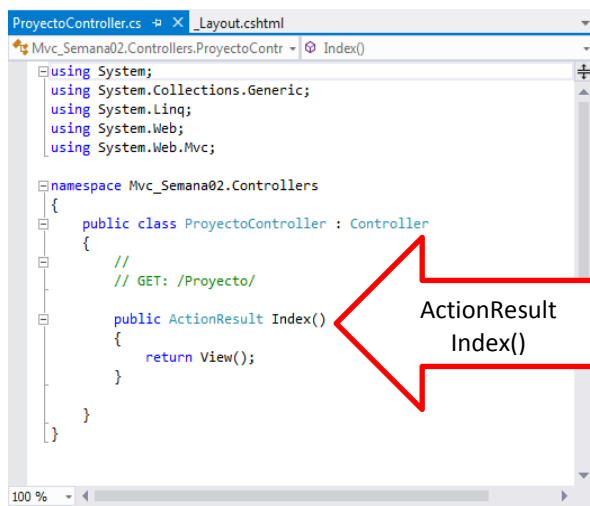
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />
  <title>@ViewData("Title")</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
<body>
  <div>
    
  </div>
  <div>
    @RenderBody()
  </div>
  <div>
    
  </div>
  @Scripts.Render("~/bundles/jquery")
  @RenderSection("scripts", required:=False)
</body>
</html>
```

A continuación creamos el Controlador del Proyecto, tal como se muestra

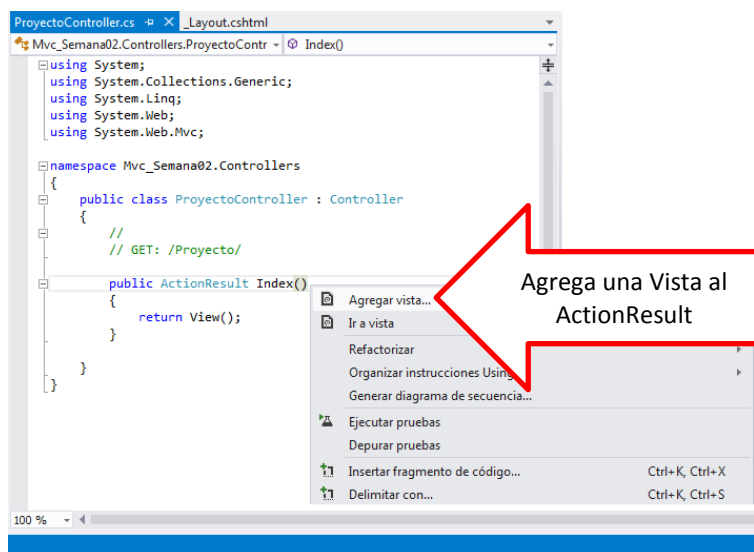




Al agregar el Controlador asigne el nombre ProyectoController, tal como se muestra en la figura



Creado el controlador se visualiza el ActionResult Index(). Agregar una Vista



Hacer click derecho al ActionResult, selecciona la opción Agregar vista..., tal como se muestra.

Agregar vista

Nombre de vista:

Motor de vistas:

☐ Crear una vista fuertemente tipada

Clase de modelo:

Plantilla para scaffold: ☒ Bibliotecas de script de referencia

☐ Crear como vista parcial

☒ Usar una página de diseño o maestra:

(Dejar vacío si se establece en un archivo _viewstart de Razor)

Id. de ContentPlaceHolder:

Conserve el nombre del ActionResult, presione el botón AGREGAR.

```

RouteConfig.cs | Index.cshtml | ProyectoController.cs | _Layout.cshtml
@{
    ViewBag.Title = "Proyecto";
}
<div>
    <table>
        <tr>
            <td><a href="#">Validaciones</a></td>
            <td><a href="#">Uso del DbContext</a></td>
        </tr>
    </table>
</div>
<div>
    <center><h1>Bienvenidos al Curso</h1></center>
    <center></center>
</div>

```

Tabla

Imagen

Diseña la página Index tal como se muestra.

```

RouteConfig.cs | Index.cshtml | ProyectoController.cs | _Layout.cshtml | FilterConfig.cs
Mvc_Semana02.RouteConfig
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Mvc_Semana02
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Proyecto", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}

```

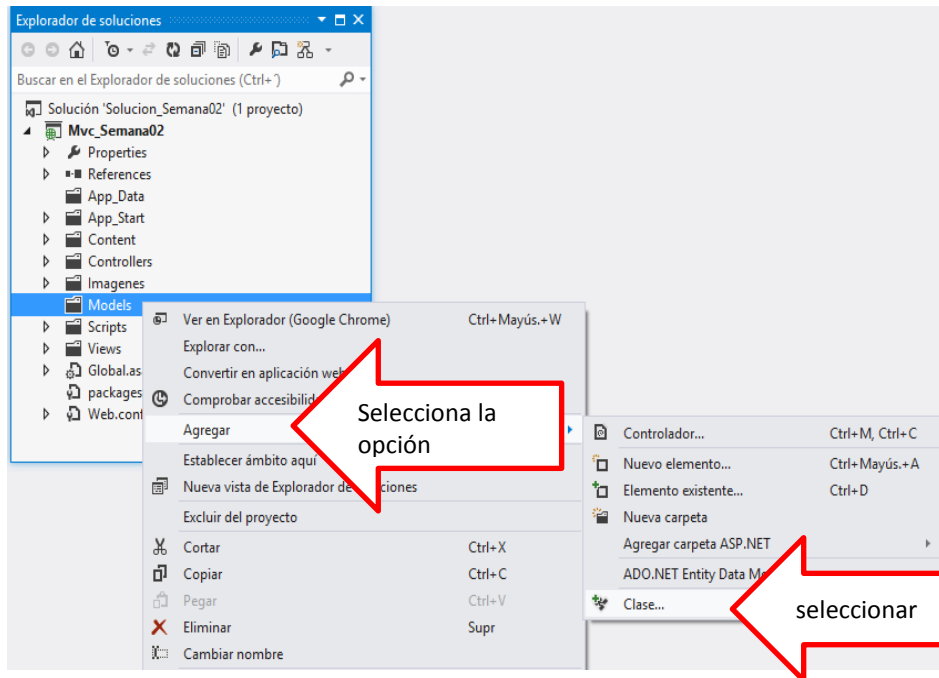
En el RouteConfig cambie el controller: Proyecto, tal como se muestra.

Al ejecutar el proyecto se visualiza la siguiente página:

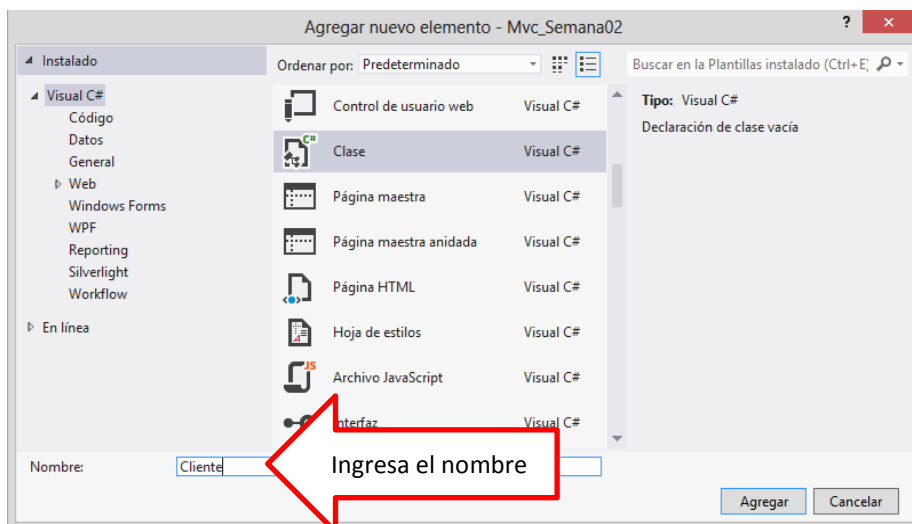


Manejo de Validaciones y Notaciones

1. Se desea implementar una página para el registro de los datos de un Cliente: código, nombre, apellido, dirección, teléfono; para ello debemos crear un Clase: desde la carpeta Models hacer click derecho y selecciona la opción **AGREGAR** y selecciona la opción **CLASE**.



Seleccionar el elemento **CLASE**, asignar un nombre: **Cliente** y presiona el botón AGREGAR.



```
Cliente.cs
Mvc_Semana02.Models.Ciente

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
//importar la librería
using System.ComponentModel.DataAnnotations;

namespace Mvc_Semana02.Models
{
    public class Ciente
    {
    }
}
```

Importar la librería de trabajo para notaciones.

Defina la estructura de la clase Cliente, tal como se muestra.

```
Cliente.cs
Mvc_Semana02.Models.Ciente

using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Mvc_Semana02.Models
{
    public class Ciente
    {
        public string codigo { get; set; }
        public string nombre { get; set; }
        public string apellido { get; set; }
        public string direccion { get; set; }
        public string telefono { get; set; }
    }
}
```

Definir las propiedades de Cliente

A continuación validamos el código del cliente: campo requerido; para ello se utiliza la notación **Required**

```
Cliente.cs
Mvc_Semana02.Models.Ciente

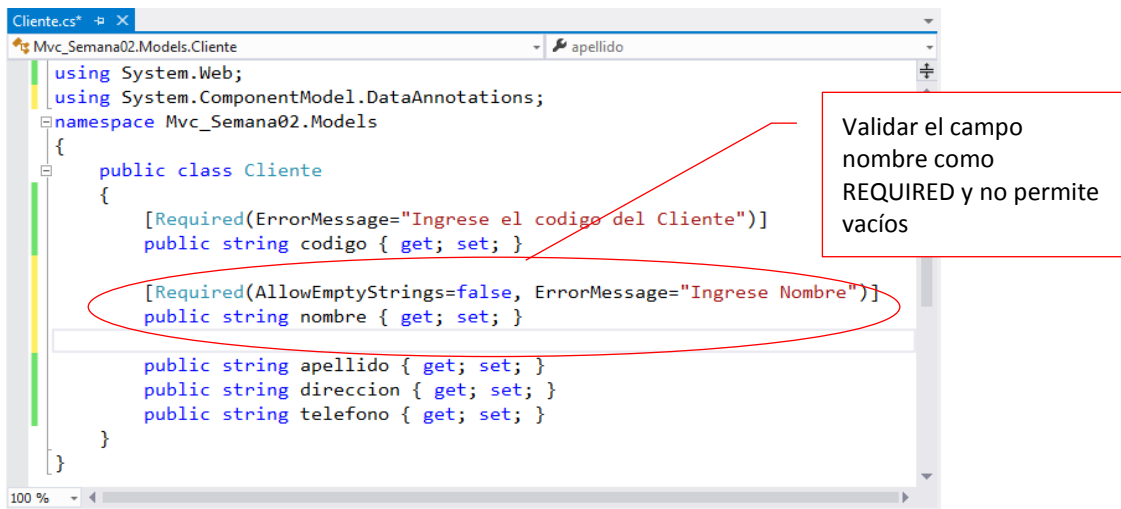
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Mvc_Semana02.Models
{
    public class Ciente
    {
        [Required(ErrorMessage="Ingrese el codigo del Cliente")]
        public string codigo { get; set; }

        public string nombre { get; set; }
        public string apellido { get; set; }
        public string direccion { get; set; }
        public string telefono { get; set; }
    }
}
```

Validar el campo código como REQUIRED

Validamos el nombre: campo requerido y no permite datos vacíos; uso de notación **Required**



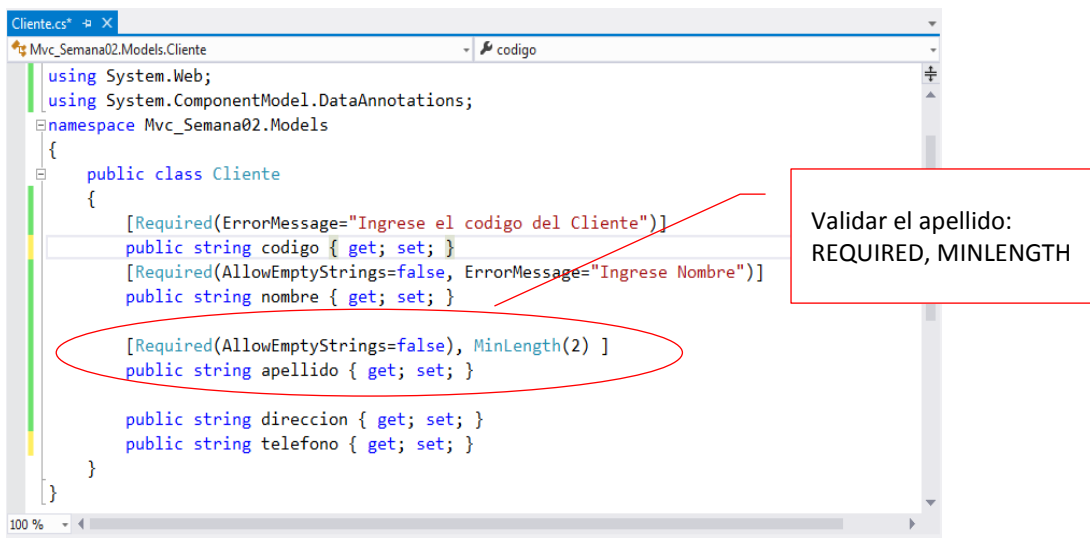
```
using System.Web;
using System.ComponentModel.DataAnnotations;
namespace Mvc_Semana02.Models
{
    public class Cliente
    {
        [Required(ErrorMessage="Ingrese el codigo del Cliente")]
        public string codigo { get; set; }

        [Required(AllowEmptyStrings=false, ErrorMessage="Ingrese Nombre")]
        public string nombre { get; set; }

        public string apellido { get; set; }
        public string direccion { get; set; }
        public string telefono { get; set; }
    }
}
```

Validar el campo nombre como REQUIRED y no permite vacíos

Validamos el campo apellido: requerido y longitud mínima 2: **Required** y **MinLength**



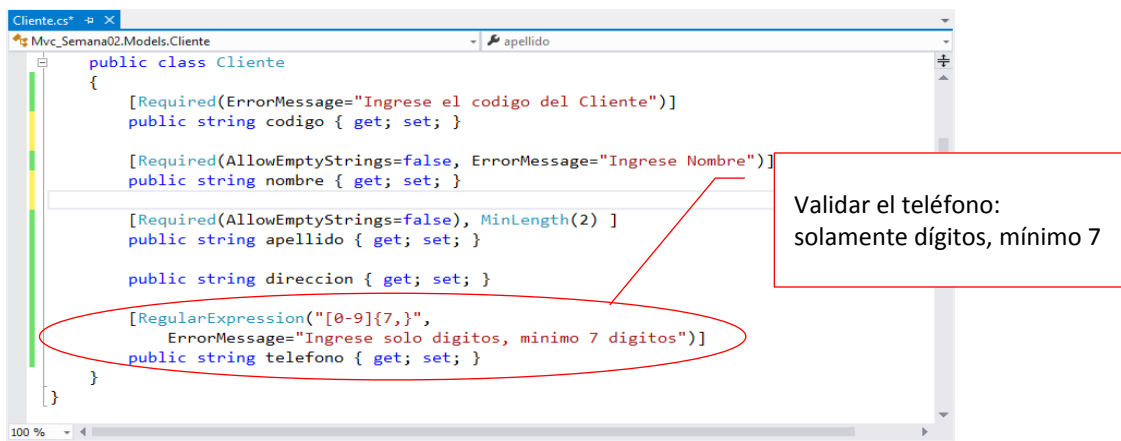
```
using System.Web;
using System.ComponentModel.DataAnnotations;
namespace Mvc_Semana02.Models
{
    public class Cliente
    {
        [Required(ErrorMessage="Ingrese el codigo del Cliente")]
        public string codigo { get; set; }
        [Required(AllowEmptyStrings=false, ErrorMessage="Ingrese Nombre")]
        public string nombre { get; set; }

        [Required(AllowEmptyStrings=false), MinLength(2)]
        public string apellido { get; set; }

        public string direccion { get; set; }
        public string telefono { get; set; }
    }
}
```

Validar el apellido: REQUIRED, MINLENGTH

Validar el teléfono: solo dígitos, su longitud mínima es 7



```
using System.Web;
using System.ComponentModel.DataAnnotations;
namespace Mvc_Semana02.Models
{
    public class Cliente
    {
        [Required(ErrorMessage="Ingrese el codigo del Cliente")]
        public string codigo { get; set; }
        [Required(AllowEmptyStrings=false, ErrorMessage="Ingrese Nombre")]
        public string nombre { get; set; }

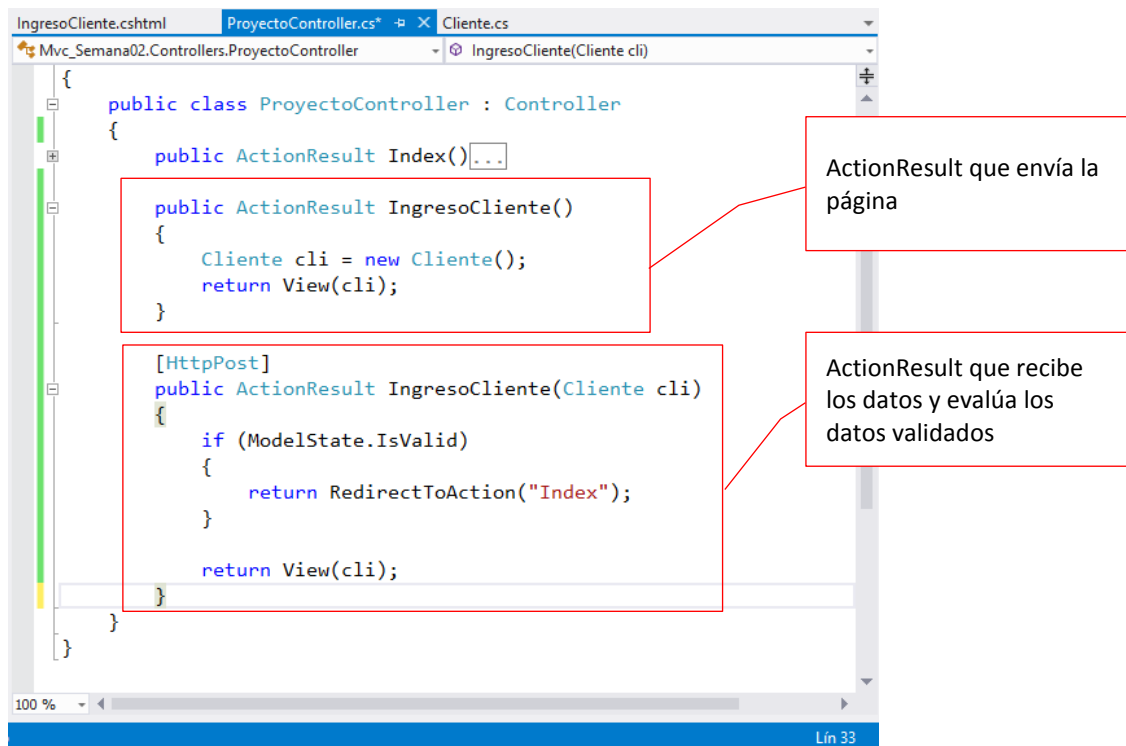
        [Required(AllowEmptyStrings=false), MinLength(2)]
        public string apellido { get; set; }

        public string direccion { get; set; }

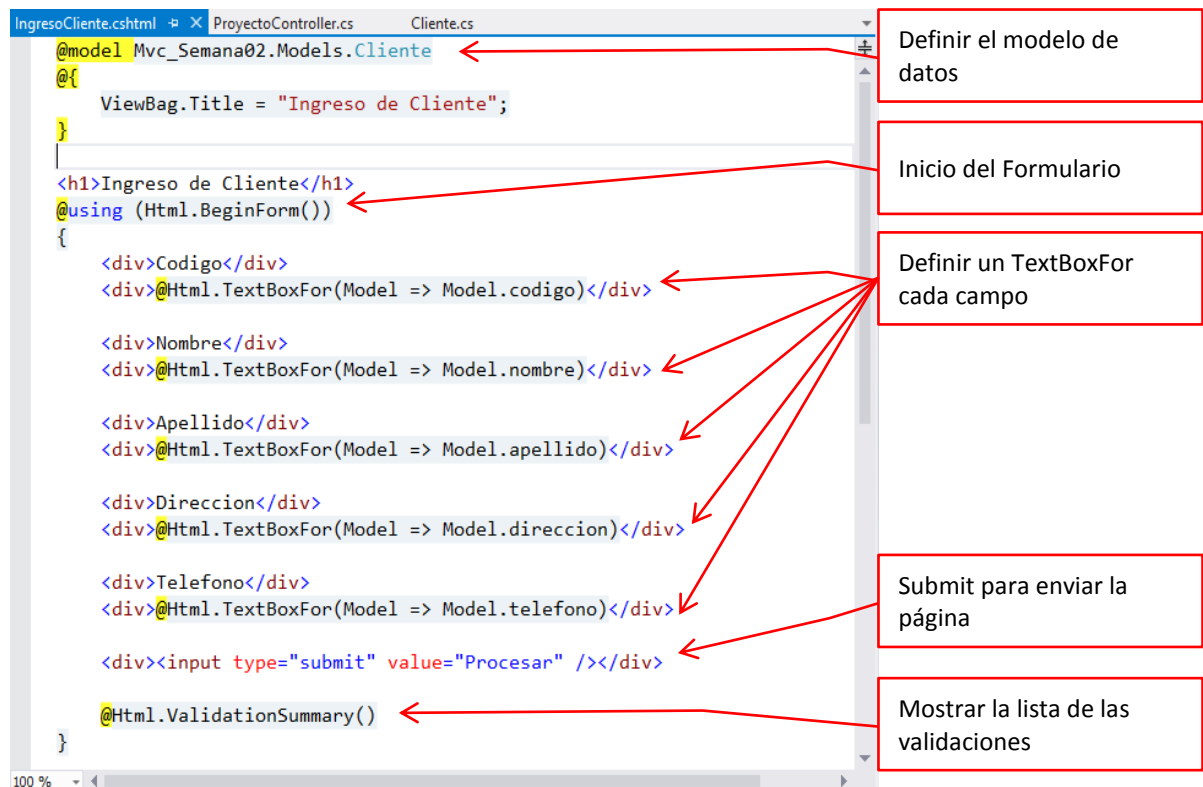
        [RegularExpression("[0-9]{7,}", ErrorMessage="Ingrese solo digitos, minimo 7 digitos")]
        public string telefono { get; set; }
    }
}
```

Validar el teléfono: solamente dígitos, mínimo 7

En el controlador Proyecto, defina el ActionResult **IngresoCliente**, donde el primero envía la estructura de Cliente y el segundo (HttpPost) recibe los datos y ejecuta la validación de su campos



A continuación agrega una Vista a IngresoCliente y defina su estructura tal como se muestra.



Ejecuta el proyecto y selecciona la Vista, observa si ejecuto el botón Procesar se muestra en el ValidationSummary() las validaciones de los campos

The screenshot shows a web browser window displaying a client login form titled "Ingreso de Cliente". The form includes input fields for "Codigo", "Nombre", "Apellido", "Direccion", and "Telefono". The "Telefono" field contains the value "w22222". A "Procesar" button is located below the form. Below the button, a red validation summary lists the following errors:

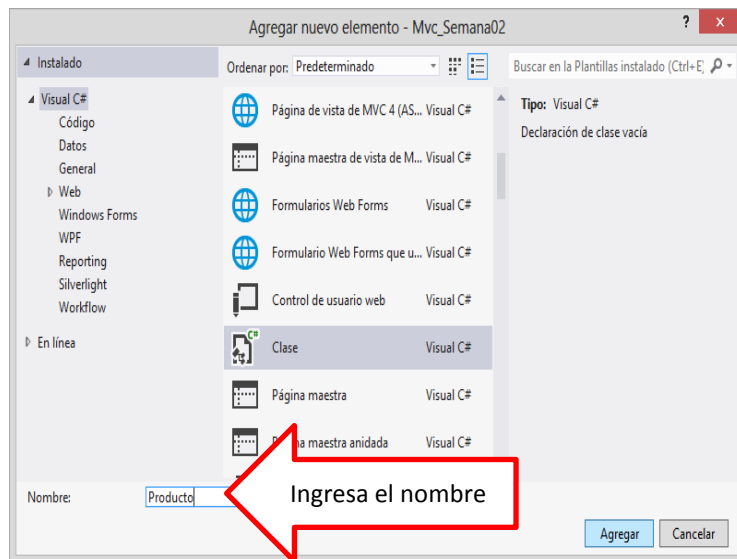
- Ingrese el codigo del Cliente
- Ingrese Nombre
- El campo apellido es obligatorio.
- Ingrese solo digitos, minimo 7 digitos

The browser's address bar shows the URL "localhost:9431/Proyecto/IngresoCliente". The background of the page features the CIBERTEC logo and the tagline "PORQUE LA TECNOLOGÍA, DEJÓ DE SER SOLO TECNOLOGÍA.".

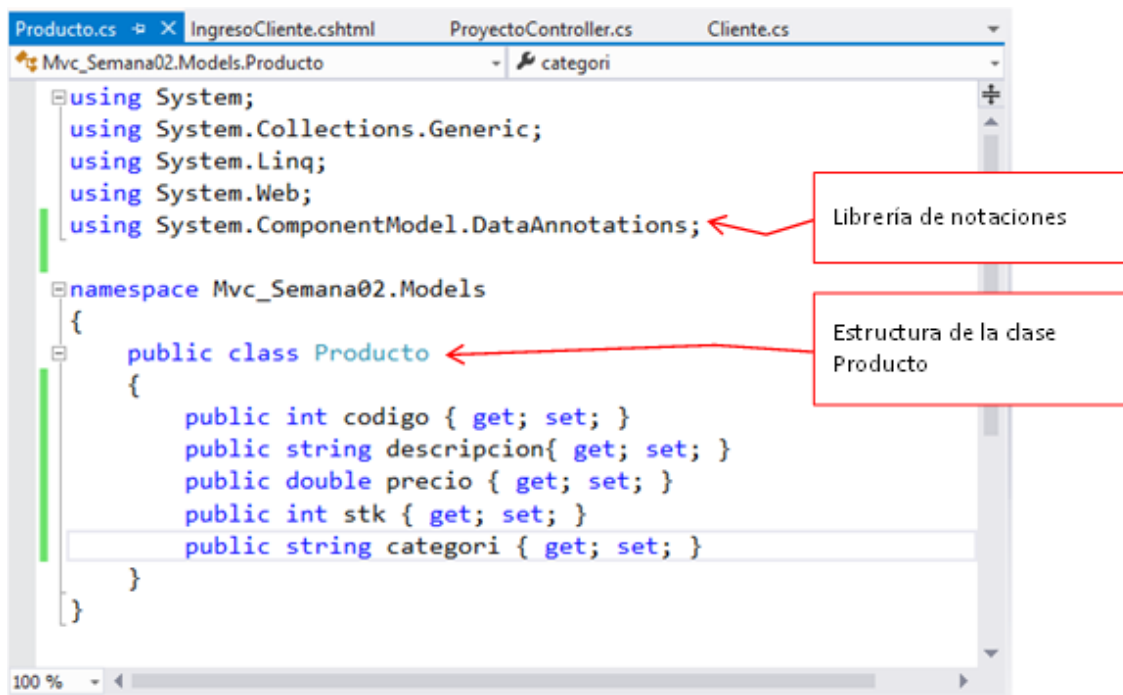
INFORMES E INSCRIPCIONES
Patricia Angus P.
Telefono: 4519-2500 Anexo: 4853
Email: patricia.angus@cibertec.edu.pe

Sede Norte: Av. Carlos Izaguirre 233, Independencia
www.cibertec.edu.pe/dca

2. Se desea implementar una página para el registro de los datos de un Producto: código, descripción, precio, stock, categoría; para ello debemos crear un Clase: desde la carpeta Models hacer click derecho y selecciona la opción **AGREGAR** y selecciona la opción **CLASE**.

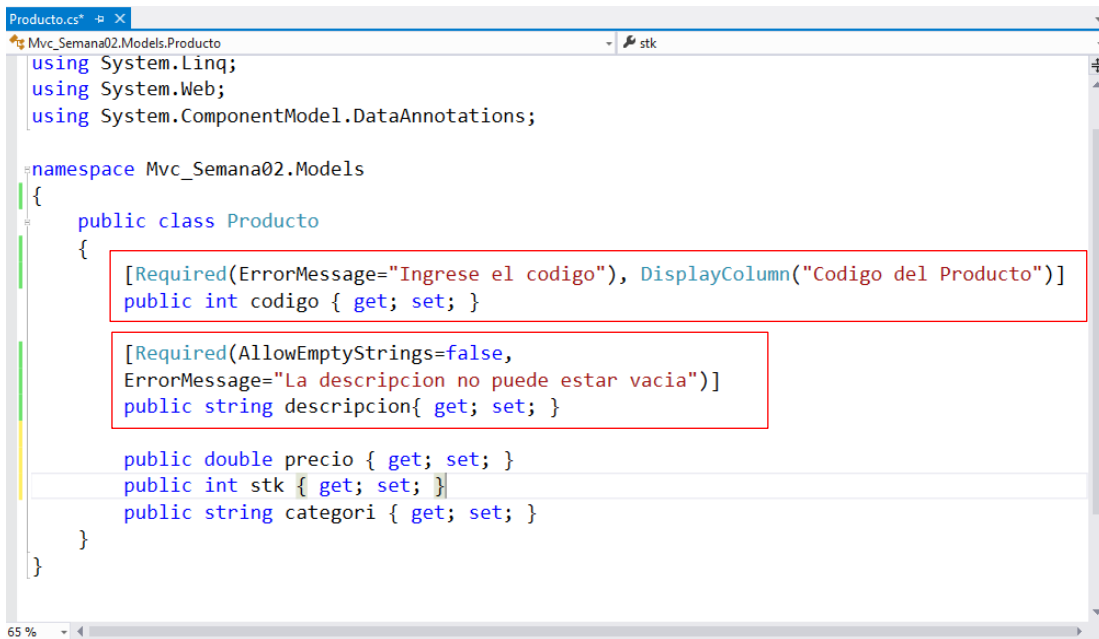


Defina la estructura de la clase Producto



Aplicando las notaciones

1. Campo código; requerido, título del campo: Código del producto
2. Campo descripción: requerido, no permite vacío.



```
Producto.cs*
Mvc_Semana02.Models.Producto
stk

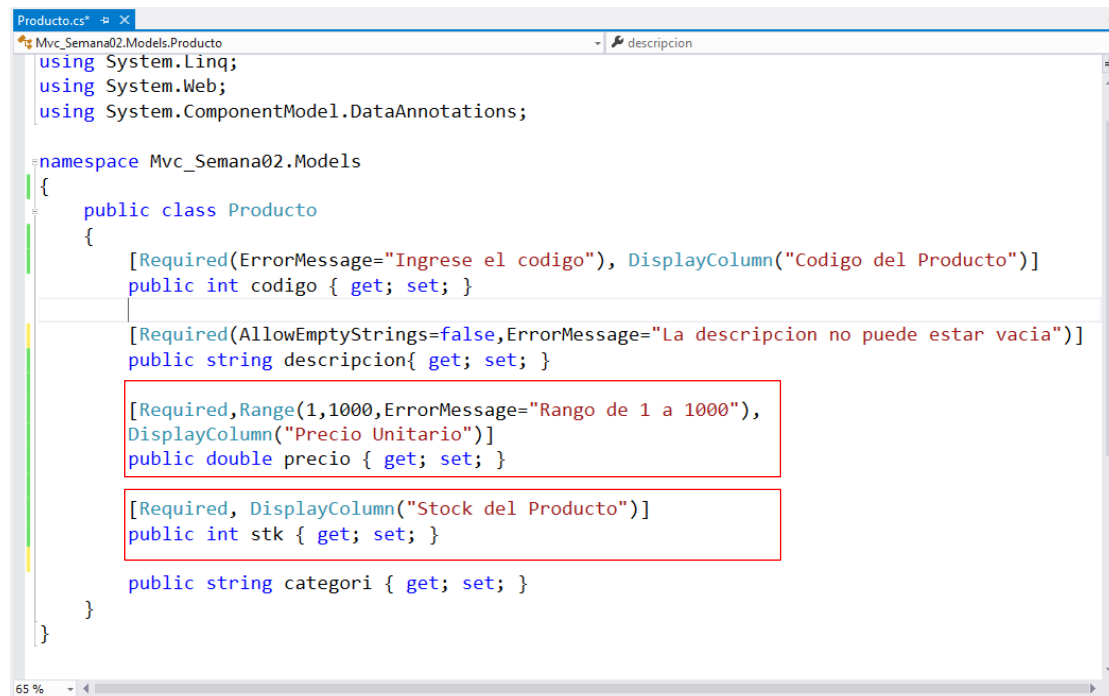
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Mvc_Semana02.Models
{
    public class Producto
    {
        [Required(ErrorMessage="Ingrese el codigo"), DisplayColumn("Codigo del Producto")]
        public int codigo { get; set; }

        [Required(AllowEmptyStrings=false, ErrorMessage="La descripcion no puede estar vacia")]
        public string descripcion { get; set; }

        public double precio { get; set; }
        public int stk { get; set; }
        public string categori { get; set; }
    }
}
```

3. Campo precio: requerido, rango entre 1 a 1000 y el título del campo: Precio Unitario
4. Campo stk: requerido y el título del campo: Stock del producto



```
Producto.cs*
Mvc_Semana02.Models.Producto
descripcion

using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Mvc_Semana02.Models
{
    public class Producto
    {
        [Required(ErrorMessage="Ingrese el codigo"), DisplayColumn("Codigo del Producto")]
        public int codigo { get; set; }

        [Required(AllowEmptyStrings=false, ErrorMessage="La descripcion no puede estar vacia")]
        public string descripcion { get; set; }

        [Required, Range(1,1000, ErrorMessage="Rango de 1 a 1000"), DisplayColumn("Precio Unitario")]
        public double precio { get; set; }

        [Required, DisplayColumn("Stock del Producto")]
        public int stk { get; set; }

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

5. Campo categoría: requerido, expresión: primer carácter el resto números, título del campo: Categoría

```
Producto.cs* x
Mvc_Semana02.Models.Producto
- categori

using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Mvc_Semana02.Models
{
    public class Producto
    {
        [Required(ErrorMessage="Ingresa el codigo"), DisplayColumn("Codigo del Producto")]
        public int codigo { get; set; }

        [Required(AllowEmptyStrings=false, ErrorMessage="La descripcion no puede estar vacia")]
        public string descripcion { get; set; }

        [Required, Range(1,1000, ErrorMessage="Rango de 1 a 1000"), DisplayColumn("Precio Unitario")]
        public double precio { get; set; }

        [Required, DisplayColumn("Stock del Producto")]
        public int stk { get; set; }

        [Required, RegularExpression("[AZ][0-9]{3}",
            ErrorMessage="Caracter y 3 digitos"), DisplayColumn("Categoria")]
        public string categori { get; set; }
    }
}
```

En el controlador Proyecto, defina el ActionResult **RegistroProducto**, donde envía la estructura de Producto y el (HttpPost) recibe los datos y ejecuta la validación de su campos

```
ProyectoController.cs* x Producto.cs
Mvc_Semana02.Controllers.ProyectoController
- Index()

using System.Web.Mvc;
using Mvc_Semana02.Models;

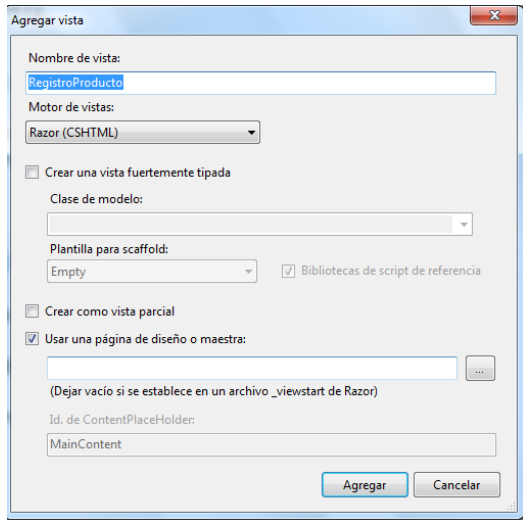
namespace Mvc_Semana02.Controllers
{
    public class ProyectoController : Controller
    {
        public ActionResult Index()...

        public ActionResult RegistroProducto()
        {
            Producto pro = new Producto();
            return View(pro);
        }

        [HttpPost]
        public ActionResult RegistroProducto(Producto pro)
        {
            if (ModelState.IsValid)
            {
                return RedirectToAction("Index");
            }
            return View(pro);
        }
    }
}
```

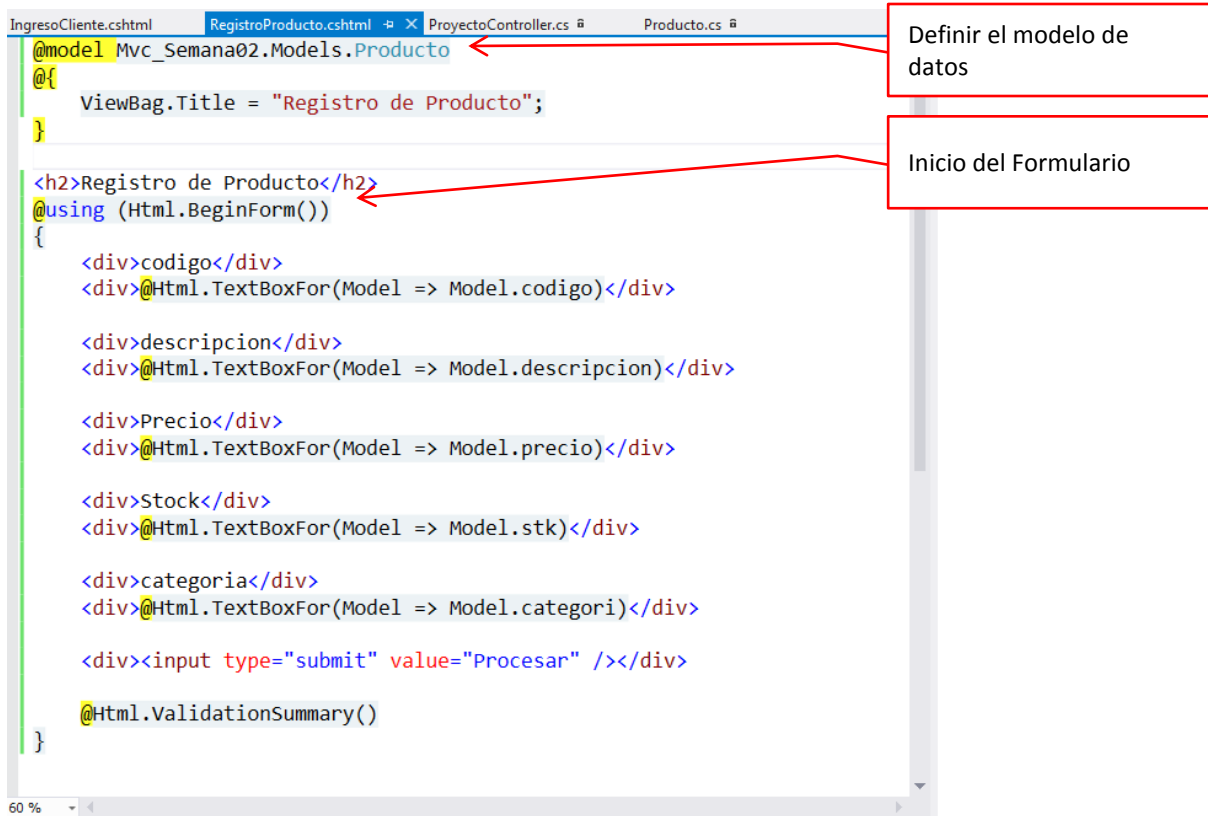
ActionResult que envía la página

ActionResult que recibe los datos y evalúa los datos validados



Agregar la vista RegistroProducto, tal como se muestra.

A continuación agrega una Vista a RegistroProducto y define su estructura tal como se muestra



Ejecute F5 y probar la página

localhost:9431/proyecto/RegistroProducto

CIBERTEC

PORQUE LA TECNOLOGÍA, DEJÓ DE SER SOLO TECNOLOGÍA.

Registro de Producto

codigo

descripcion

Precio

Stock

categoria

- La descripcion no puede estar vacia
- Rango de 1 a 1000
- Caracter y 3 digitos

INFORMES E INSCRIPCIONES
Patricia Angus P.
Telefono: 419-2900 Anexo: 4853
Email: patricia.angus@cibertec.edu.pe

Sede Norte: Av. Carlos Izaguirre 233, Independencia.
www.cibertec.edu.pe/dca

CIBERTEC