

# Aprendiendo a Programar

## Tutorial del Curso

Autor:

Comunidad IT - Asociación Civil de Formación e Inserción Laboral en Tecnologías de la Información

# Contenido

Capítulo 2. Aplicaciones en la nube y cómo comenzar a programar .....	3
Capítulo 3. La interfaz de usuario .....	37
Capítulo 4. Inteligencia en la interfaz de usuario .....	82
Capítulo 5. El Servidor Web .....	136
Capítulo 6. El lenguaje C# .....	158
Capítulo 7. El Servidor Web. Conceptos Avanzados .....	189
Capítulo 8. Dónde y cómo se guardan los datos .....	211
Capítulo 9. Conectando los datos con la aplicación .....	242

Nota: Los capítulos 1, 10, 11 y 12 son teóricos, por lo que no hay tutorial incluido.

# Aplicaciones en la nube y cómo comenzar a programar

preparación del ambiente de desarrollo

Aprendiendo a programar. Capitulo 2. Tutorial

consideraciones iniciales

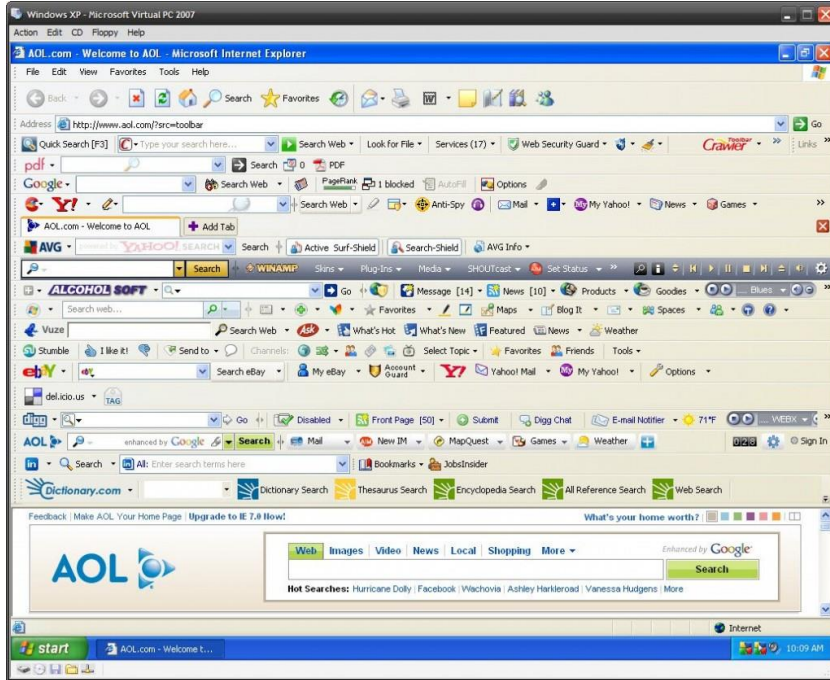
Para que puedas utilizar tu computadora para diseñar aplicaciones debes tener especial cuidado de los programas que tengas instalados

verificar primero

Tu computadora debe estar libre de virus, spyware y software innecesario. Debes desinstalar toda barra, widgets o programas que no sea esencial en tu computadora

esto incluye

Barras de  
navegación  
adicionales en el  
browser  
Y extensiones  
innecesarias



esto incluye

# Widgets y aplicaciones corriendo en segundo plano



considera

Cuidar el software que se instala en la computadora.

Si algo falla, es difícil detectar si se trata de un error en algo que estamos programando o generado por un programa secundario



si compartes la computadora  
con otras personas

Intenta crear y  
utilizar un perfil de  
usuario diferente  
exclusivamente  
para programar y  
mantenlo aislado



Cecilia Cornejo



Dave Linds



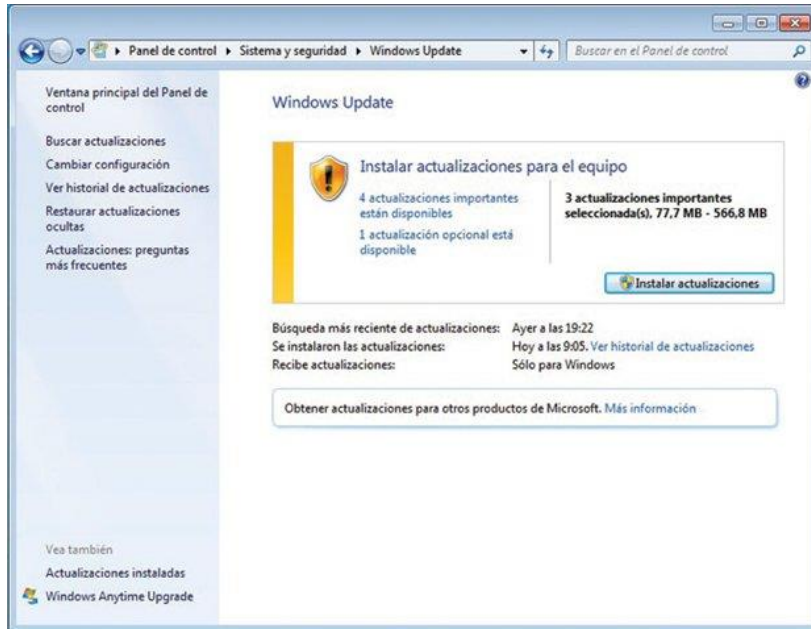
Tony Madigan



Other user



comencemos



Aunque sea una obviedad antes de instalar, asegúrate que tu Sistema Operativo este actualizado



última versión del  
navegador de internet

Descarga la ultima  
versión disponible  
que exista del  
browser de  
internet



no sólo de Microsoft, de todos  
los fabricantes

Debes asegurarte  
que tu aplicación  
funciona para  
todo tipo de  
cliente

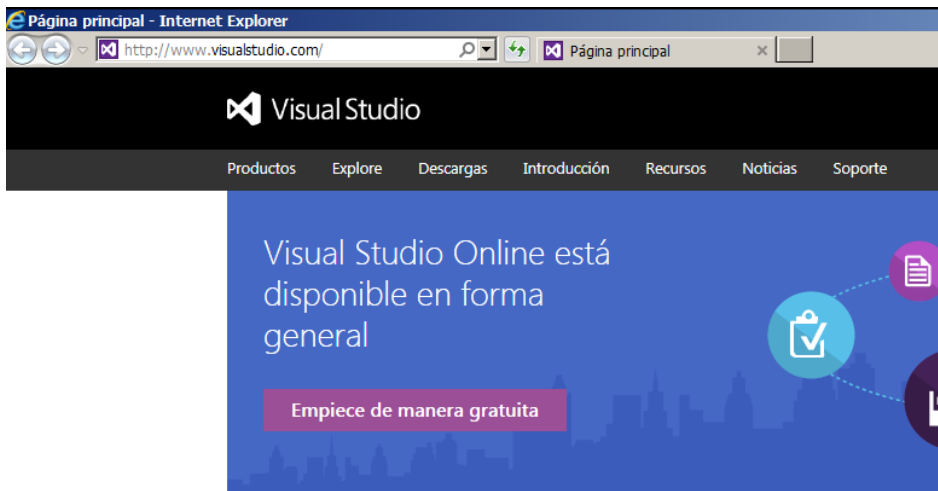
denominaciones

El entorno de desarrollo se llama  
Microsoft Visual Studio

Existen versiones Profesionales,  
Empresariales y Express

diferencias

Si bien las versiones Profesionales y Empresariales traen algunas características adicionales, con las versiones Express alcanzan (y sobran) para comenzar a programar



Cualquier aplicación, cualquier equipo

no descargues los programas de cualquier sitio

Utiliza siempre el sitio oficial para descargar.  
Las versiones Express no tienen costo

## Visual Studio Express 2013

Visual Studio Express 2013 para Web

Visual Studio Express 2013 para Windows

Visual Studio Express 2013 para Windows Desktop

Visual Studio Team Foundation Server Express 2013

Visual Studio Express 2012 para Windows Phone

Sin coste para los estudiantes:

Descárgalo hoy mismo en [DreamSpark.com](http://DreamSpark.com) →

no es necesario instalar todo

Para este curso  
precisas instalar

Visual Studio  
Express Edition  
Web



si tu PC tiene algunos años

Puedes instalar las versiones 2012 o 2010 si tu computadora tiene un Sistema Operativo de hace unos años También sirven para comenzar.

De todos modos recuerda, precisas una conexión permanente a Internet.

si tu conexión a internet no es muy rápida

Intenta descargar la versión Imagen ISO, es como una copia del DVD.

Luego quema un DVD con la imagen o bien utiliza Virtual CD-ROM (si no lo tienes, descárgalo) para abrir el la imagen ISO y ejecutar el instalador

Finalizada la instalación, la primera vez te solicitará

Iniciar sesión con un usuario de desarrollo.

Esto es para acceder a los servicios adicionales en la nube para desarrolladores.

Puede que sea tu cuenta habitual de hotmail/live u otra de productos MS

al momento de iniciar

Te solicitará conectarte a AZURE (opcional), puedes hacerlo en otro momento.

AZURE te permitirá acceder a servicios para tu aplicación, algunos son gratuitos, otros pagos.

Descubra las novedades de Express 2013 para Web

Puede encontrar información sobre las nuevas características y mejoras en Express 2013 para Web si revisa las secciones siguientes.

[Más información acerca de las nuevas características de Express 2013 para Web](#)

Conozca las novedades de .NET Framework 4.5.1

Descubra las novedades de Team Foundation Service

## Conectar a Azure

[Obtenga más información acerca de Azure](#)

Conectar ➔

[Reubicar la información sobre novedades](#)

## Novedades en desarrollo web

ASP.NET y Visual Studio Tools

ASP.NET 4.5 y Web

Microsoft Azure

## Crear aplicaciones para Web

[Más información acerca de la creación de sitios web](#)

[Más información acerca de la creación de API web](#)

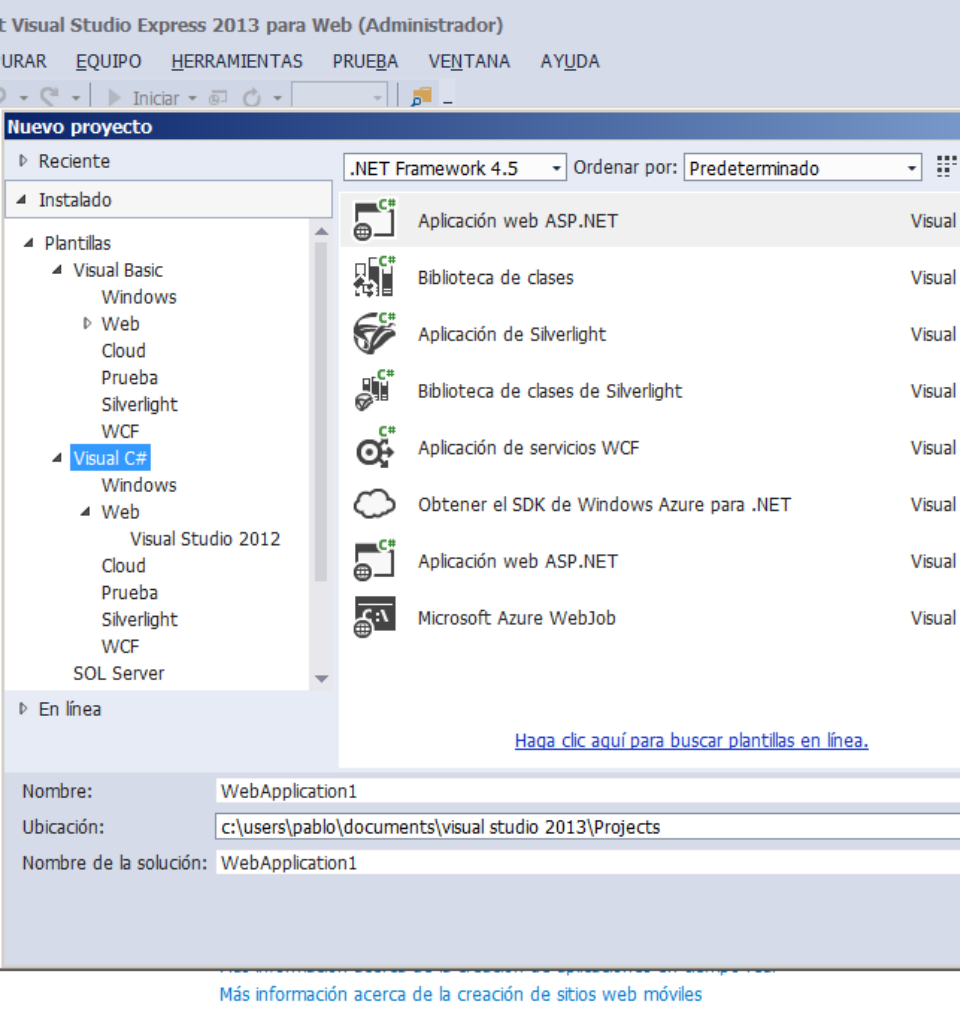
Más información acerca de la creación de aplicaciones en tiempo real

Más información acerca de la creación de sitios web móviles

si visualizaste la pantalla anterior

Haz completado la instalación con éxito.

Hagamos la primera prueba

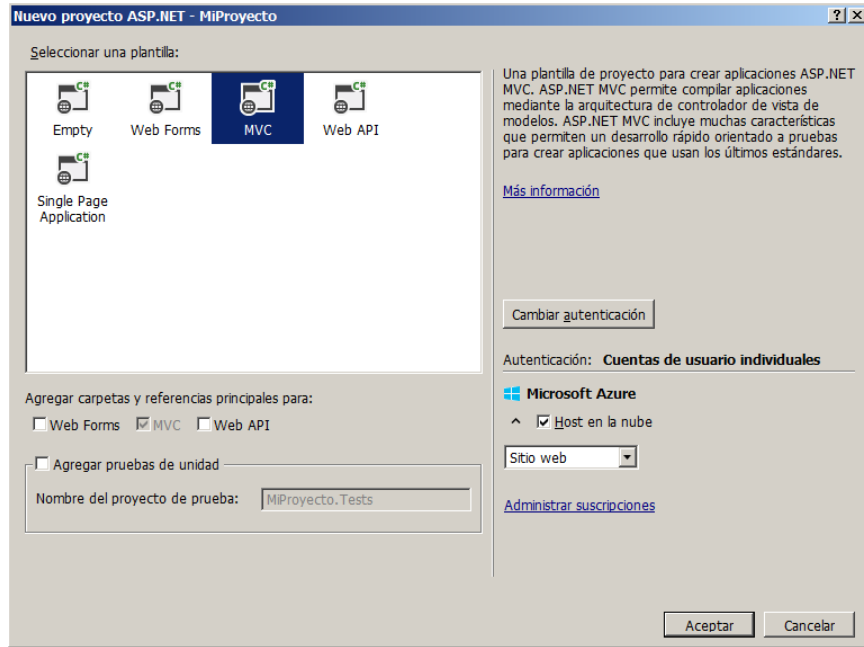


inicia un nuevo proyecto

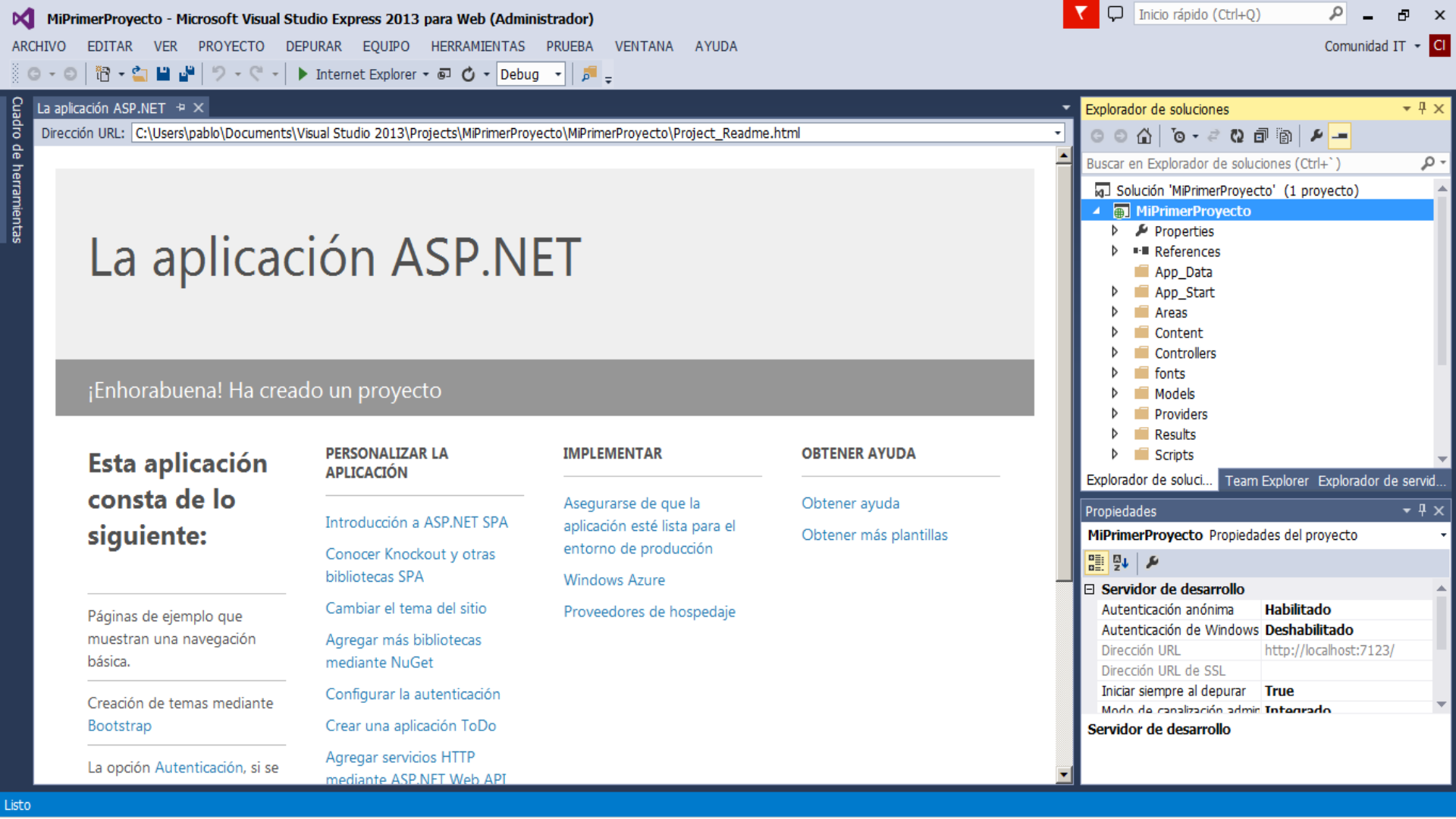
Visual Studio  
soporta diferentes  
lenguajes.  
Asegúrate de  
elegir Visual C# -  
Web – Aplicación  
Web ASP.NET

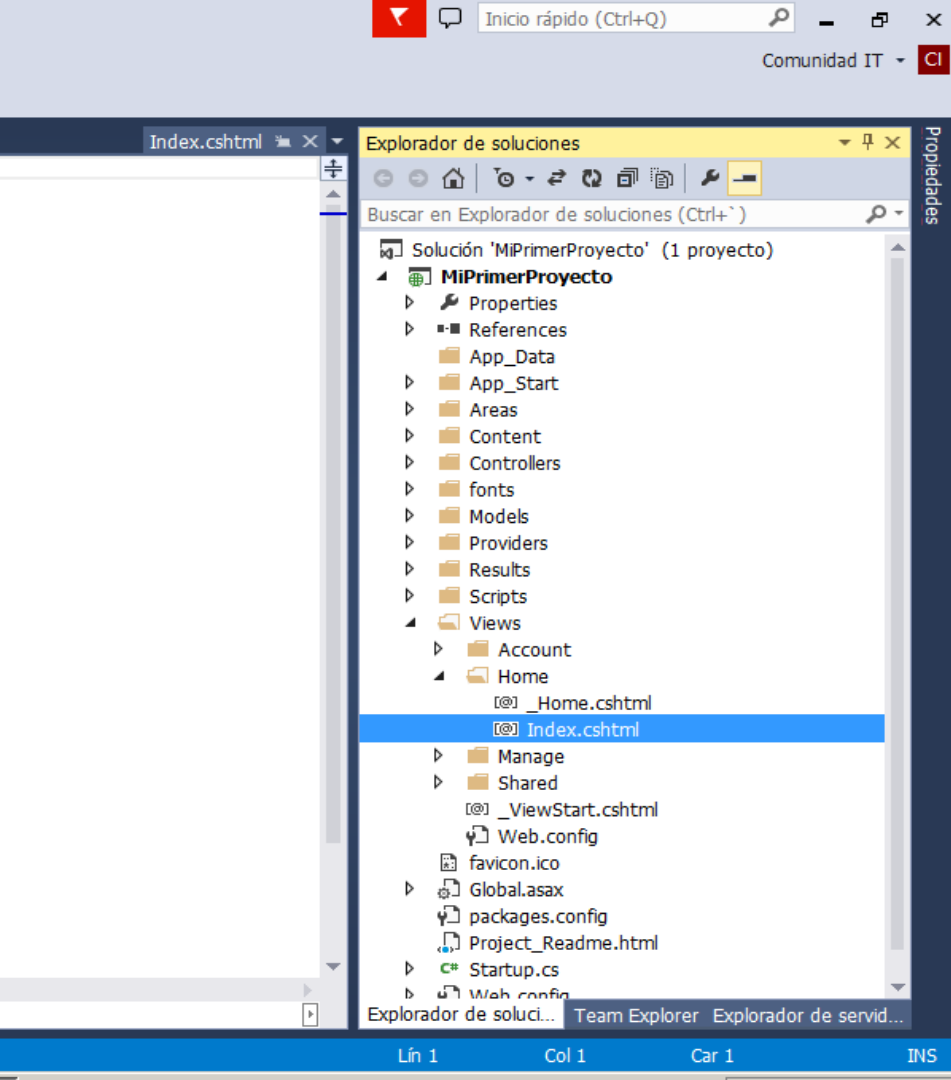
en la configuración inicial

Puedes utilizar la plantilla Single Page para entender la estructura.  
En este ejemplo utilizaremos la plantilla MVC



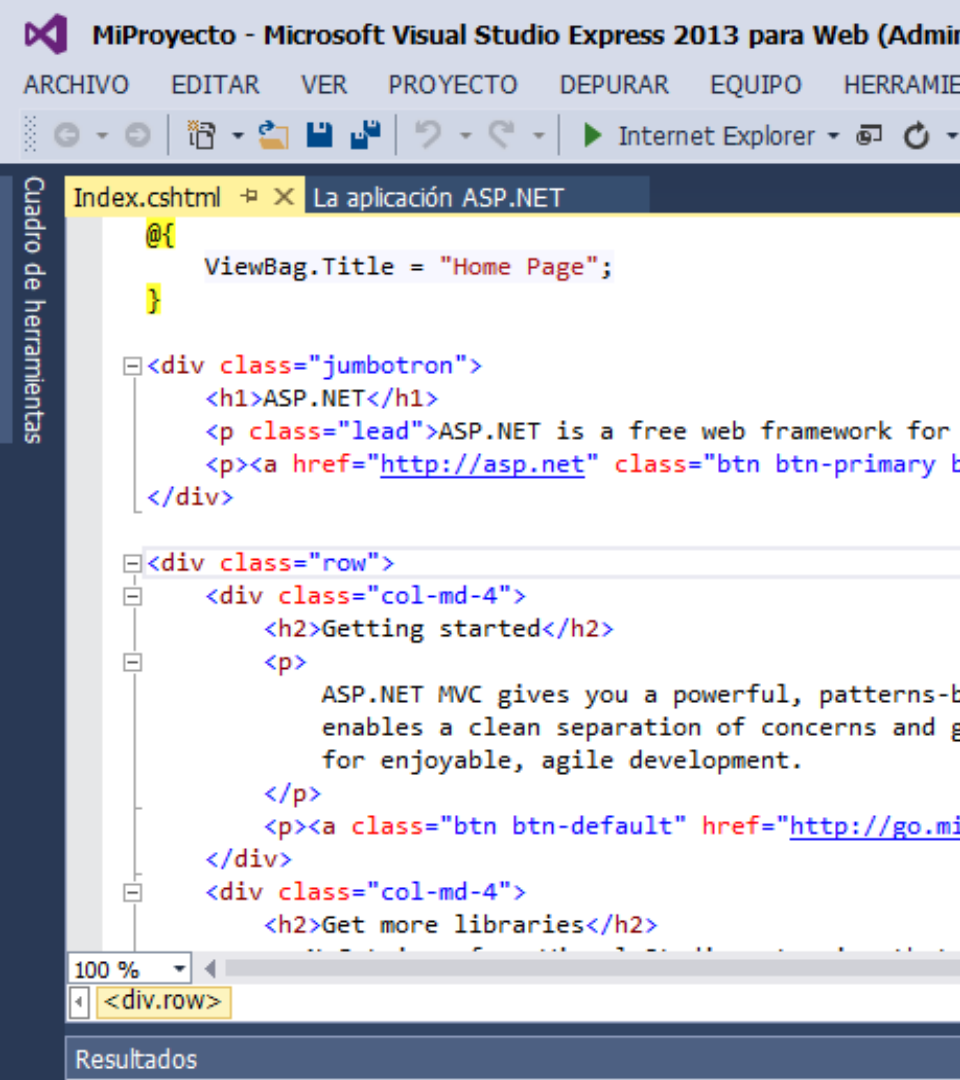






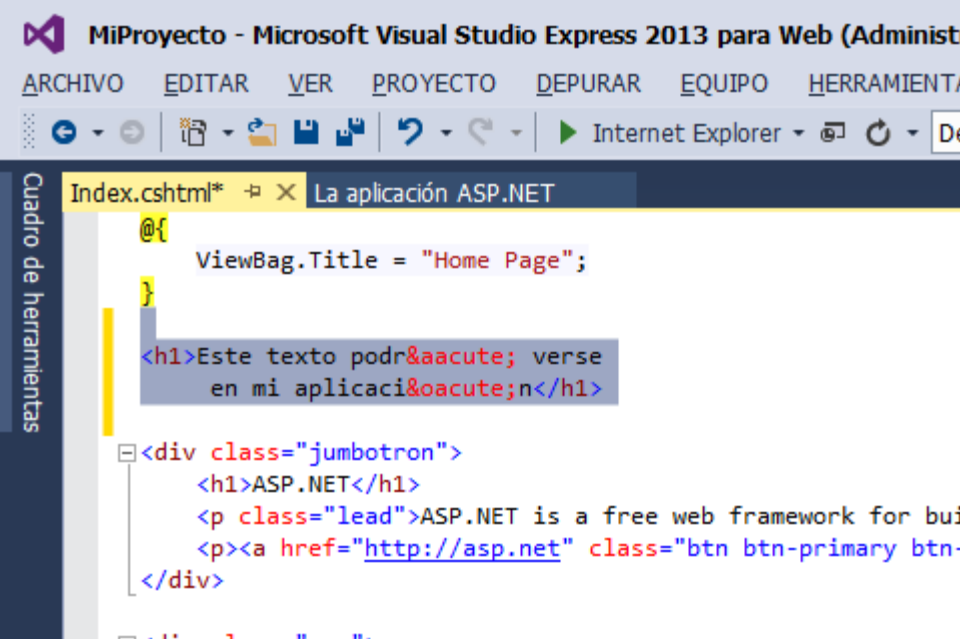
en la parte lateral de tu proyecto

Encontrarás el  
«Explorador de  
Soluciones»,  
desde aquí,  
podrás cada uno  
de los archivos de  
tu proyecto



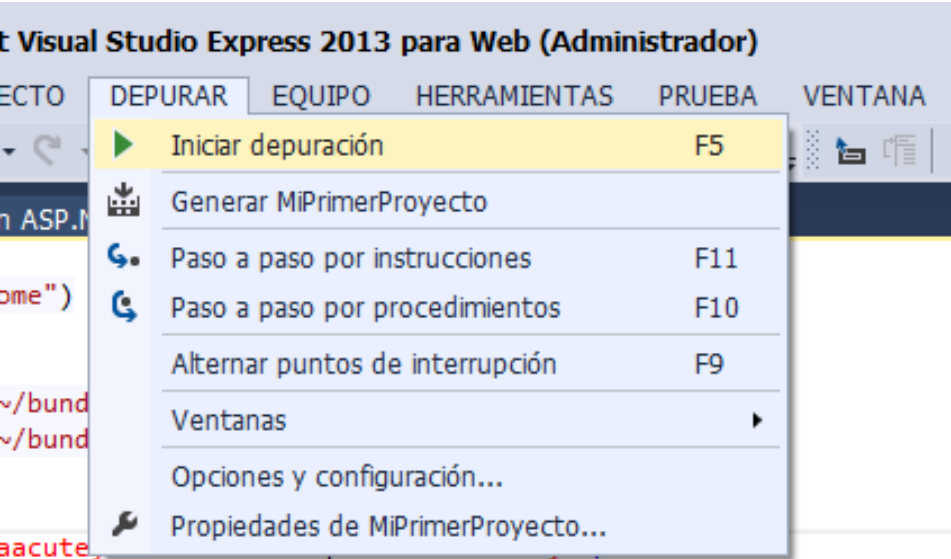
desde el explorador de soluciones

# Abre y edita index.cshtml



en el documento index.cshtml

Y escribe lo siguiente.  
Ten cuidado con los símbolos, forman parte del lenguaje HTML



presiona el botón Iniciar  
( el símbolo PLAY o presiona F5 )

Luego de unos  
segundos (lo que  
tarde en compilar),  
se abrirá tu  
aplicación en un  
browser de  
internet

[Nombre de aplicación](#)[Inicio](#)[Acerca de](#)[Contacto](#)[Registrarse](#)[Iniciar sesión](#)

# Este texto podrá verse en mi aplicación

# ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

[Learn more »](#)

## Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

[Learn more »](#)

## Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

[Learn more »](#)

## Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

[Learn more »](#)fH\_8Md-Pbioxomt:  
VInrAl5sdzR2jrCDn

/2NnnamQFrX-NbY

Felicitaciones!

Acabas de escribir tus primeras líneas de código y lograste que tu primera aplicación mostrara el resultado que deseabas.

entendiendo un poco lo que hicimos

Escribiste una porción de código en lenguaje HTML, el cual tiene una simbología en particular.

Aplicaste los cambios en la aplicación, la cual se compila y te permite ver el resultado.



explora un poco más

Puedes editar todos los archivos en de extensión cshtml del proyecto y escribir trozos de código HTML. Comienza explorando en el buscador web con las claves «basic html elements»

debes detener la aplicación

Para editar y probar los nuevos cambios, siempre debes detener la aplicación



ten en cuenta

En Visual Studio, los archivos del tipo html se denominan cshtml (ya que pueden hacer cosas adicionales).  
Lo que encuentres para html, funciona para los cshtml

tu entorno está listo para desarrollar en un ambiente local

Esta primera prueba sirvió para verificar la correcta instalación del entorno de desarrollo. Considera que tu aplicación sólo es visible en tu computadora. Luego podrás instalarla en un servidor WEB para acceder a ella desde cualquier parte del mundo

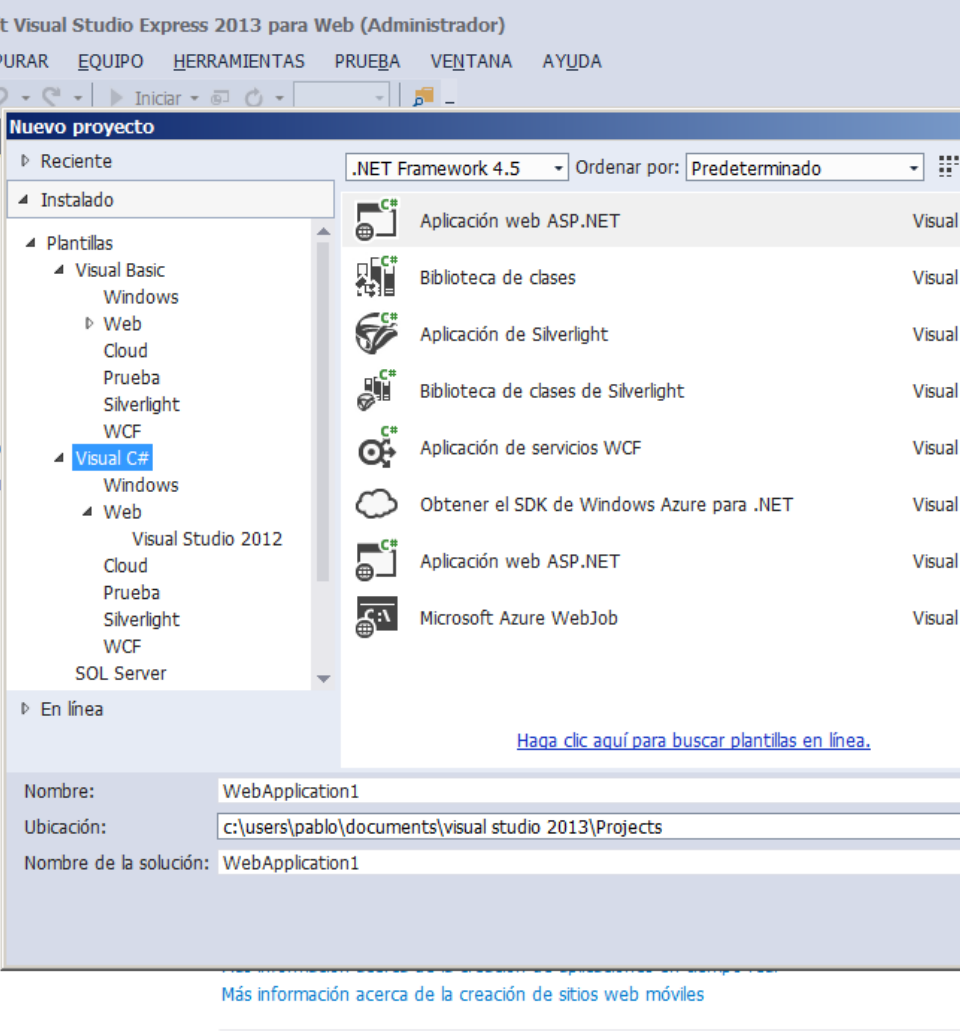
# La interfaz de usuario

Crear y editar HTML, uso de herramientas del desarrollador

Aprendiendo a programar. Capítulo 3. Tutorial

## objetivo

Crearemos una Aplicación Web de cero, modificaremos las interfaces de usuario en HTML, su estilo en CSS e identificaremos como se invocan las diferentes pantallas de nuestra aplicación



comencemos

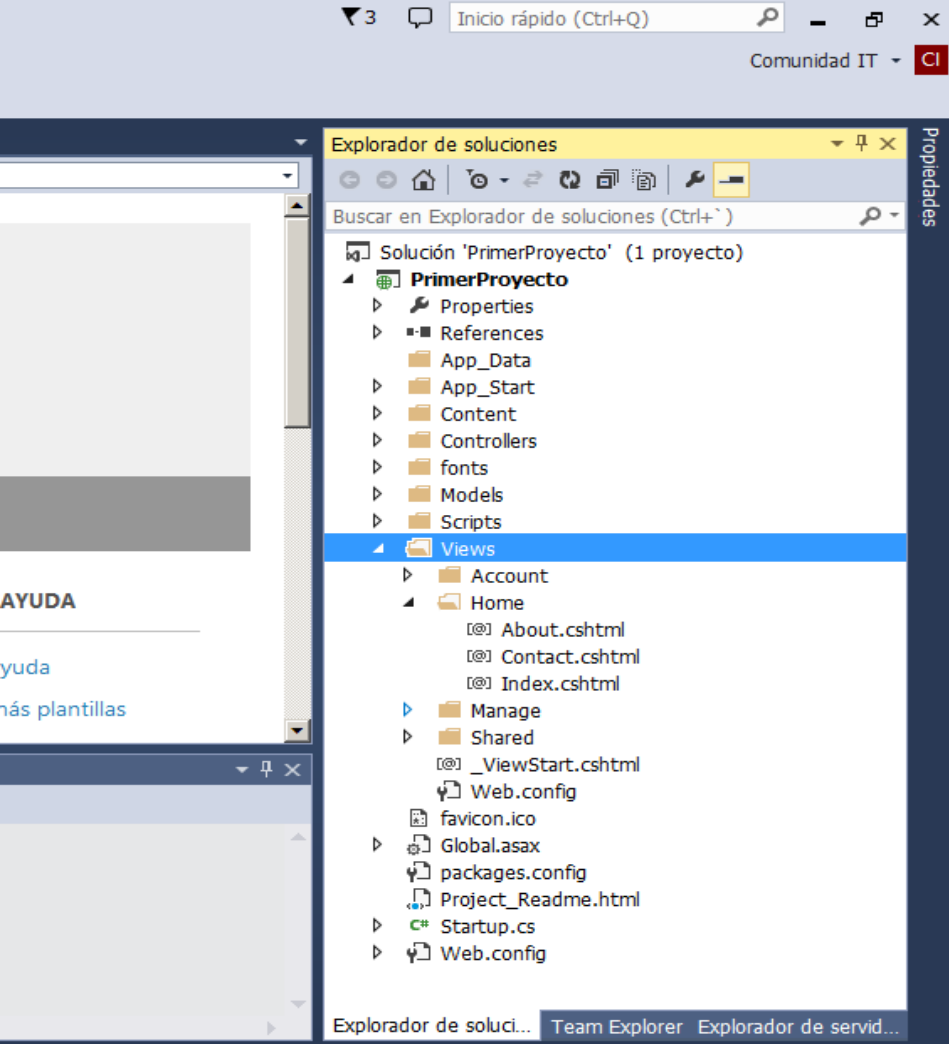
Crea un nuevo  
proyecto de tipo  
Visual C# -  
Aplicación web  
ASP.NET y utiliza la  
plantilla simple  
MVC

vamos a crear una nueva página y agregarla al proyecto

Recuerda que en programación, cuando creas un nuevo archivo se deben evitar los nombres que contengan espacios, acentos, números o símbolos.

Reemplaza los espacios con el guión bajo \_ o bien omítelos.





en el explorador de soluciones

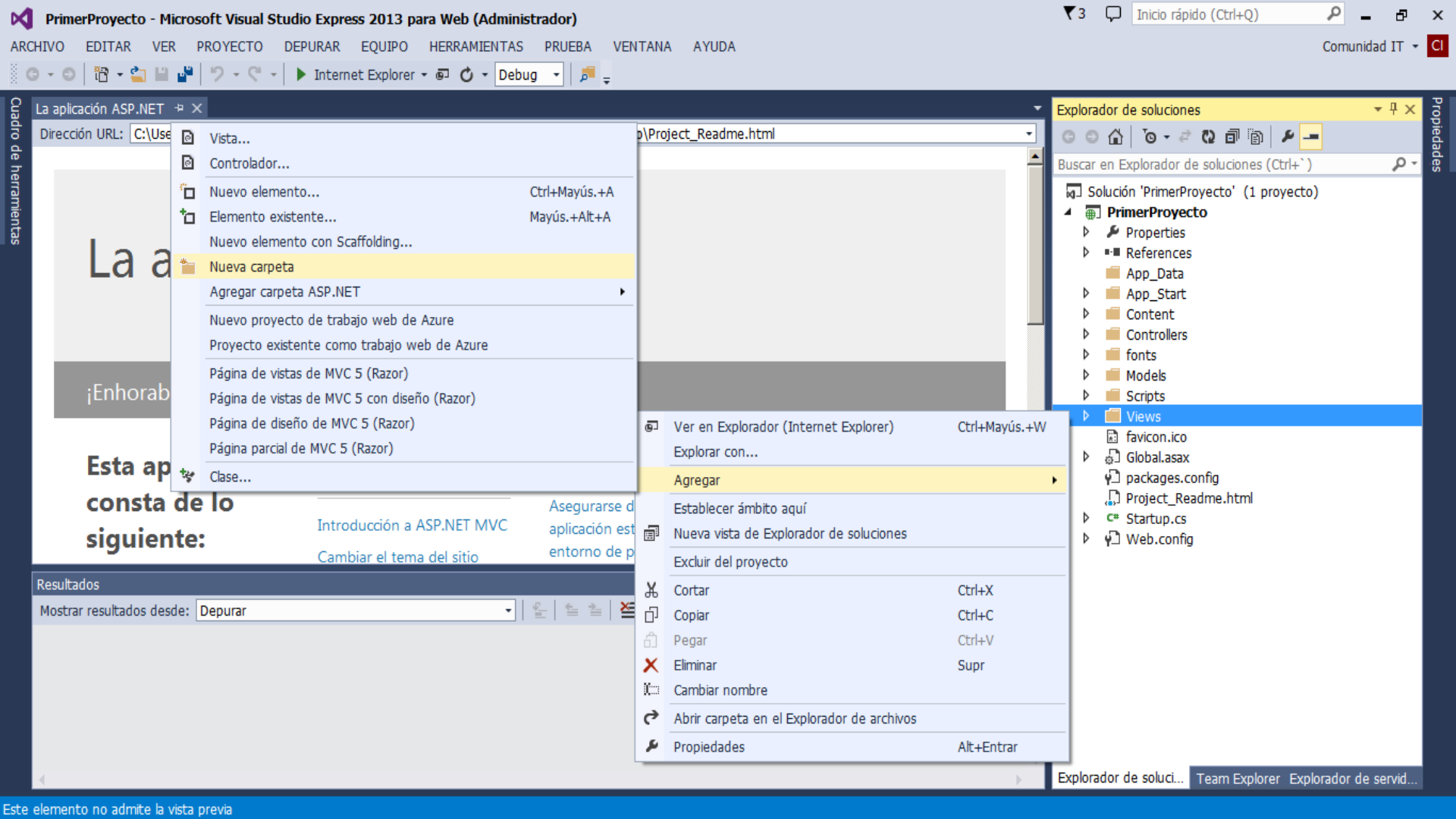
Las pantallas (vistas) HTML siempre están en la carpeta VIEWS. Las carpetas sirven para ordenar los archivos.

crea una nueva carpeta sobre VIEWS

Posiciónate sobre Views, click derecho.

Agregar – Nueva Carpeta

Coloca el nombre «Custom»



agrega una nueva vista (archivo cshtml), en la carpeta custom, recién creada

Posiciónate sobre Custom , click derecho.

Agregar – Nueva Vista

Coloca el nombre «Principal»

Agregar vista

Nombre de vista:

Plantilla:

Clase de modelo:

Clase de contexto de datos:

Opciones:

☐ Crear como vista parcial

☒ Bibliotecas de script de referencia

☒ Usar página de diseño:

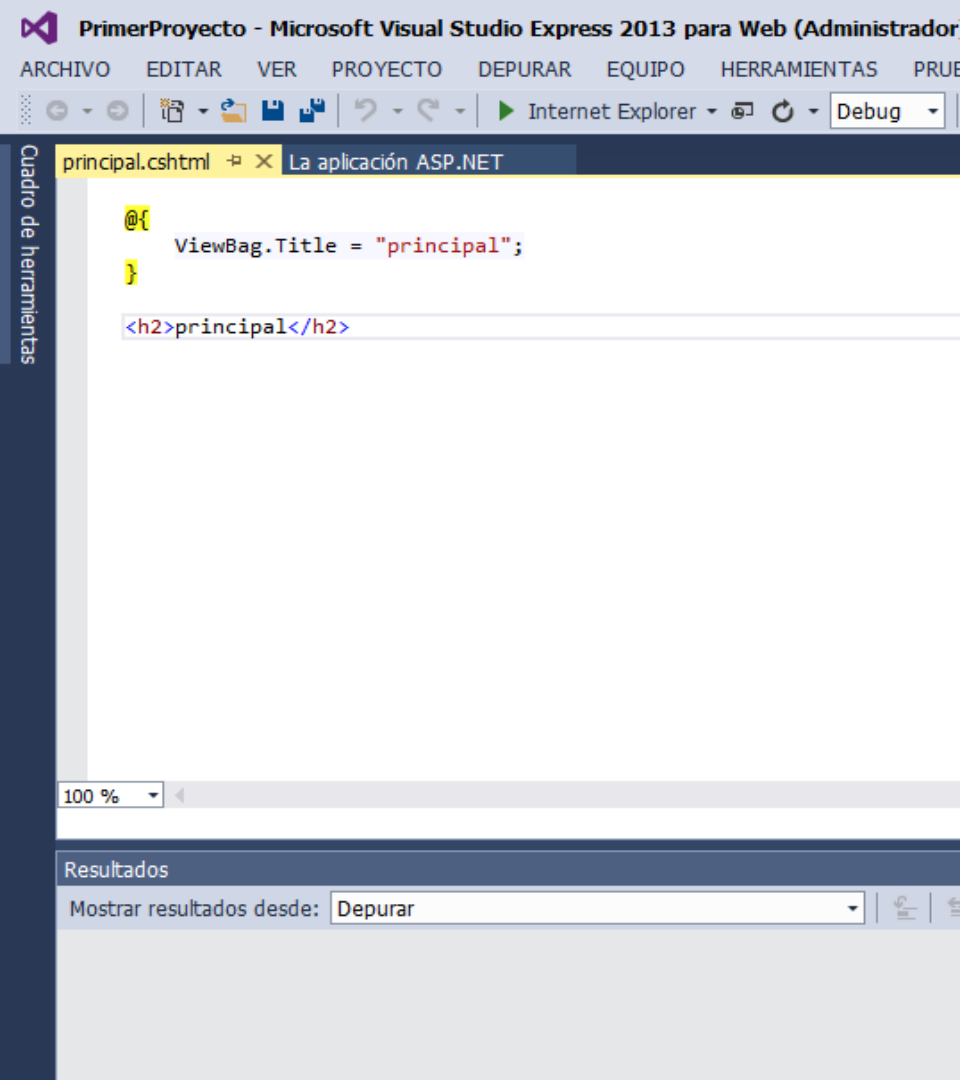
...

(Dejar en blanco si se define en un archivo \_viewstart de Razor)

Agregar Cancelar

al momento de crear la vista  
te preguntará si quieres usar  
una plantilla

Elige la plantilla vacía  
(empty)



este debería ser tu resultado

Tu vista está lista  
para ser editada

antes de continuar, debemos entender algunos principios

El modelo de desarrollo MVC tiene  
un comportamiento que necesitamos  
entender para poder continuar  
Veamos cuales son

las pantallas (vistas) son el resultado de una acción del usuario

Para poder ver una pantalla de nuestra aplicación, el usuario debe hacer algo, ya sea un click en un link o bien escribir la ruta completa en la barra de direcciones del navegador web.



nunca se solicita una pantalla por el nombre del archivo

En la barra de direcciones, nunca escribimos el nombre del archivo para poder acceder (en este caso /Custom/Principal.cshtml)  
Sino que accedemos en forma INDIRECTA, a través de los archivos llamados CONTROLLERS

para que se pide la pagina en forma indirecta

El CONTROLADOR, es el intermediario entre lo que pide el usuario y lo que deseamos mostrarle.

Un CONTROLADOR podría decidir mostrar un página u otra, por ejemplo /Custom/Principal.cshtml o bien /Custom/Principal\_Version\_Ingles.chnml

porqué se pide en forma indirecta una página?

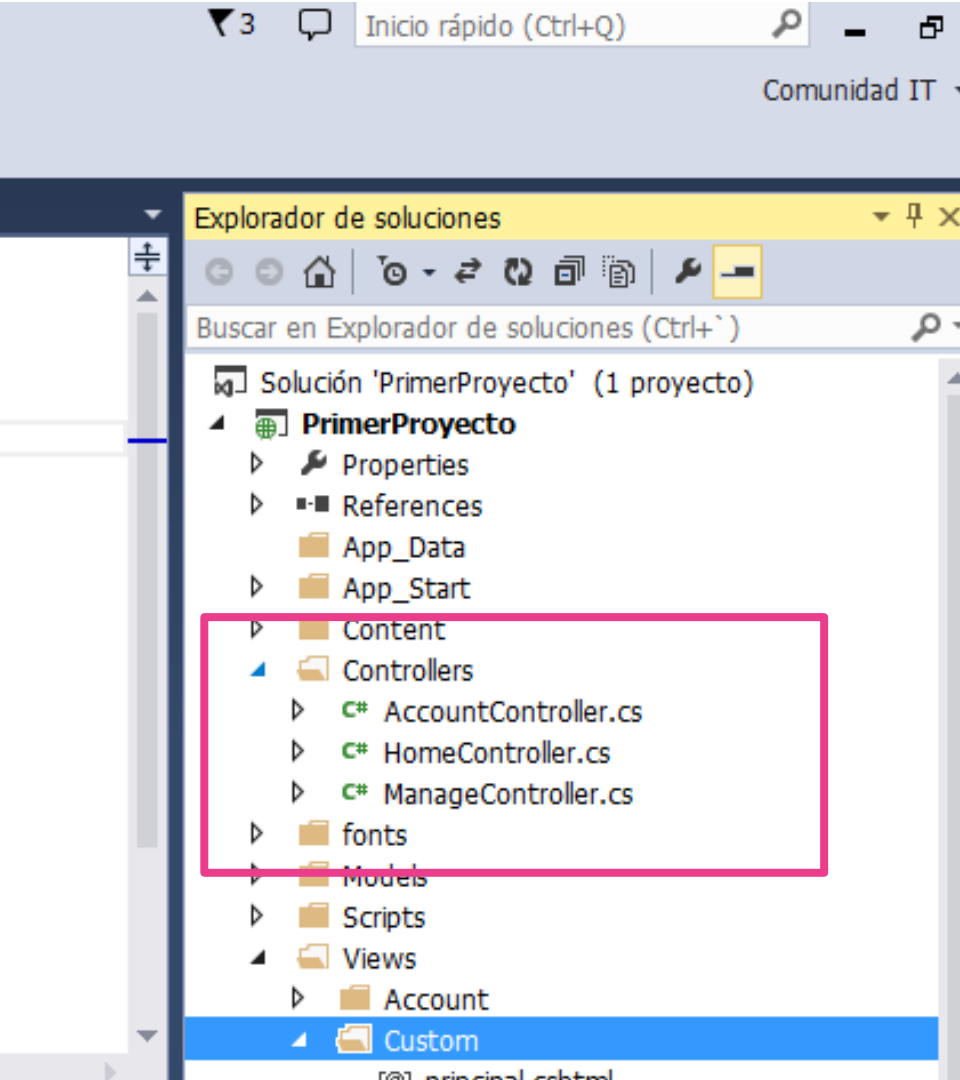
El CONTROLADOR permite tomar decisiones en nuestra aplicación, y en base a esas decisiones, mostrar uno u otro resultado

para el usuario de nuestra aplicación esto es INVISIBLE

Cuando solicitemos la página, lo haremos por el punto de entrada que definamos en el CONTROLADOR.

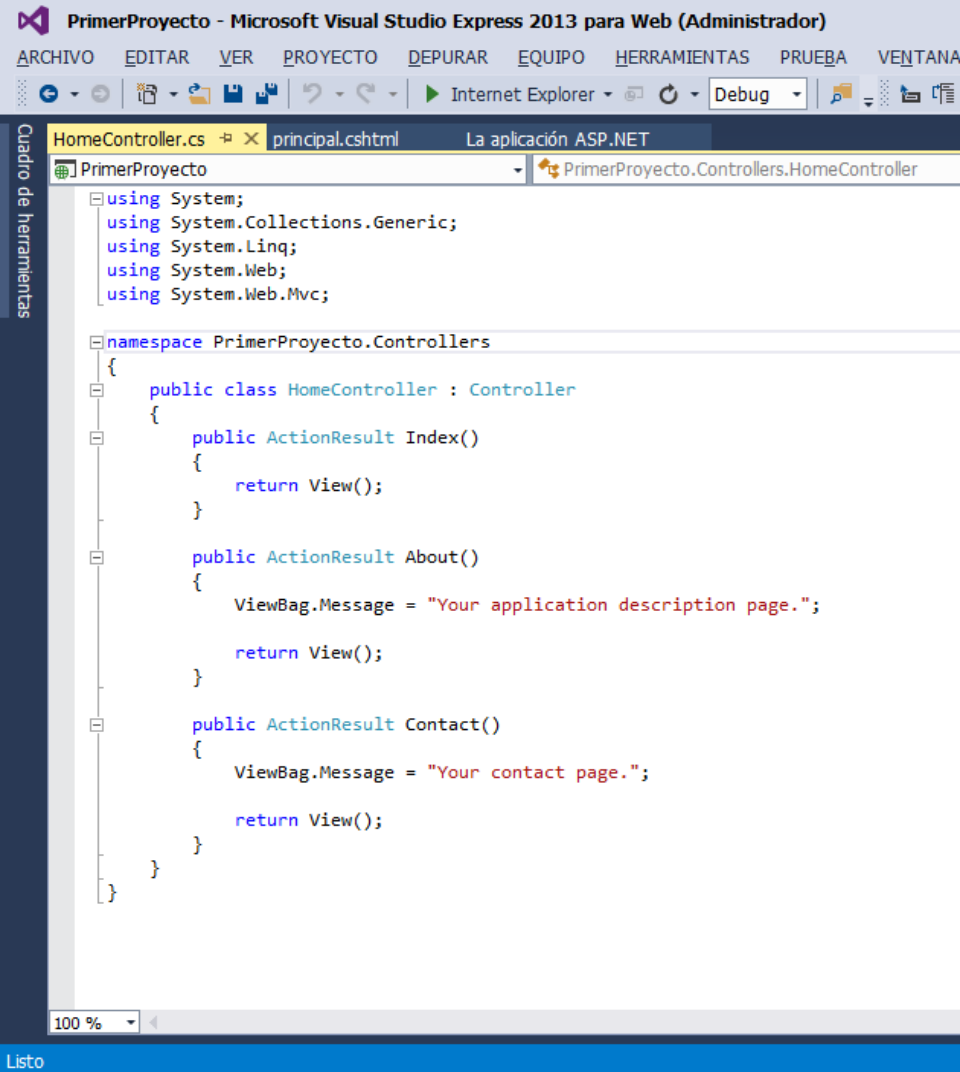
Este punto de entrada puede tener un nombre DISTINTO a la página.

El usuario no sabe que si le mostramos la pantalla A o B, solo sabe que ingresó a la ruta que le provee el CONTROLADOR



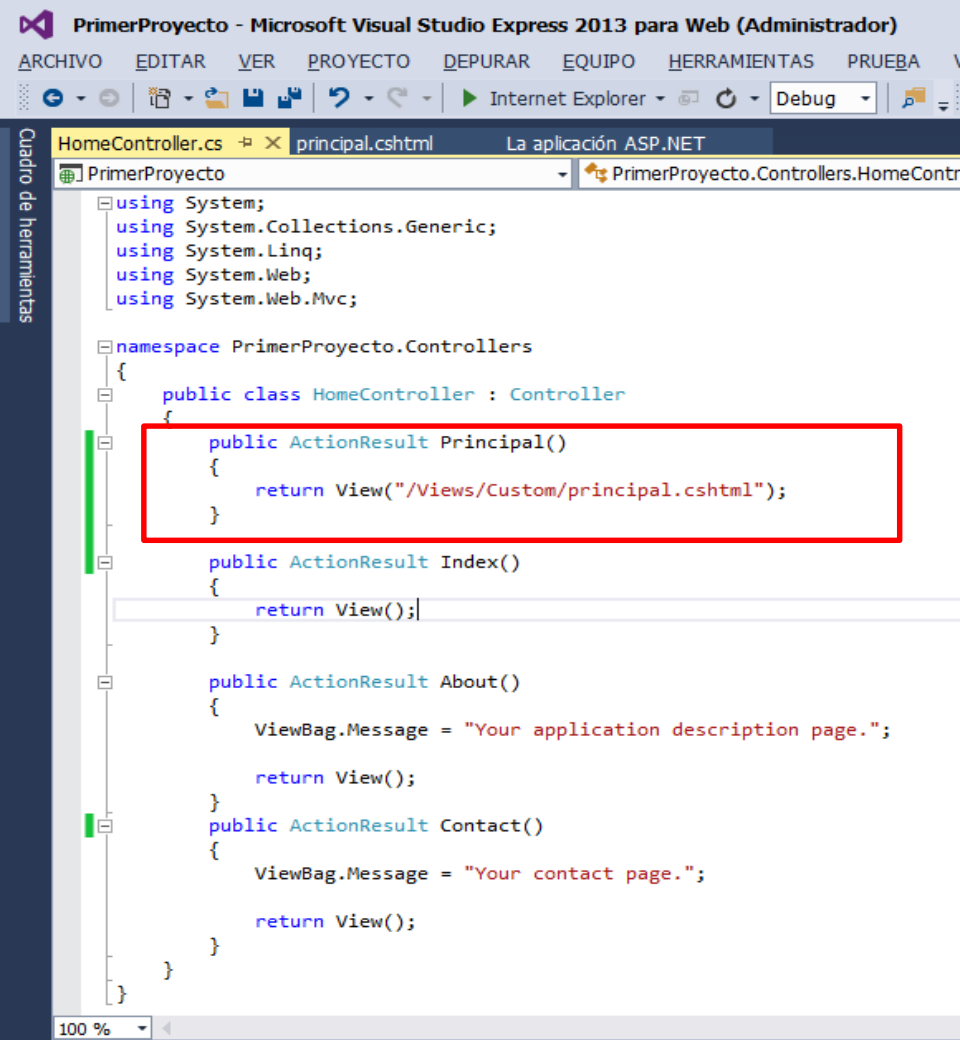
identificando los controladores

Los archivos  
Controladores  
están ubicados en  
la carpeta  
**CONTROLLER.**  
En este caso  
existen 3



abre y edita HomeController.cs

El lenguaje que ves  
escrito es C#.  
Veremos más detalles  
en el siguiente  
capítulo.  
Necesitamos crear un  
punto de acceso a  
nuestra vista



agrega el punto de entrada en el CONTROLADOR

Ten mucho cuidado de agregar la porción de código señalada, con sus símbolos especiales

observa lo siguiente

El punto de entrada le indica la ubicación del archivo, en nuestro caso la carpeta CUSTOM y la vista principal.cshtml

```
return View("/Views/Custom/principal.cshtml");
```



que significa?

Que nuestra vista, ahora es accesible a través del browser de internet a través del punto de entrada

**/Home/Principal**

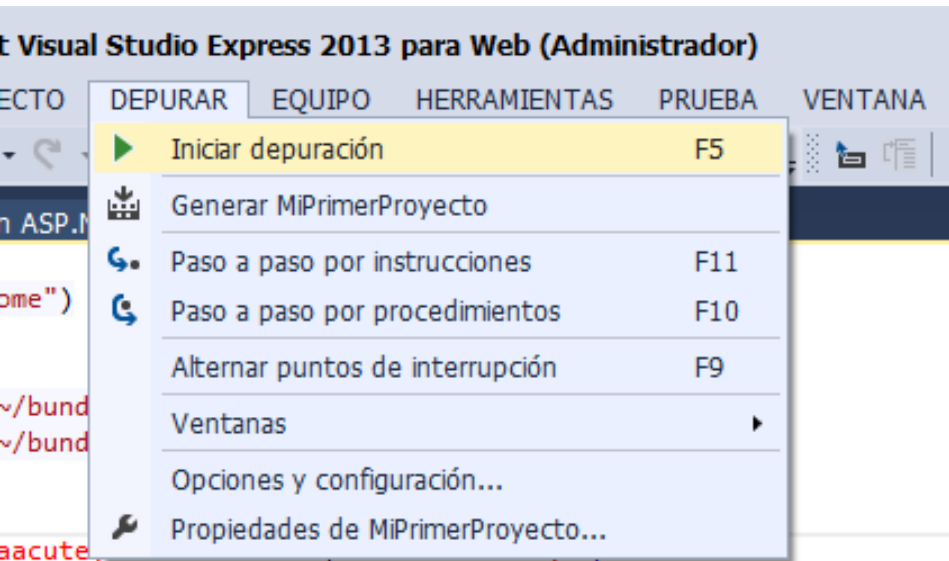
Home, es el nombre del controlador

**HomeController.cs**

Principal es el nombre de la función

**public ActionResult Principal()**

probemos la pagina

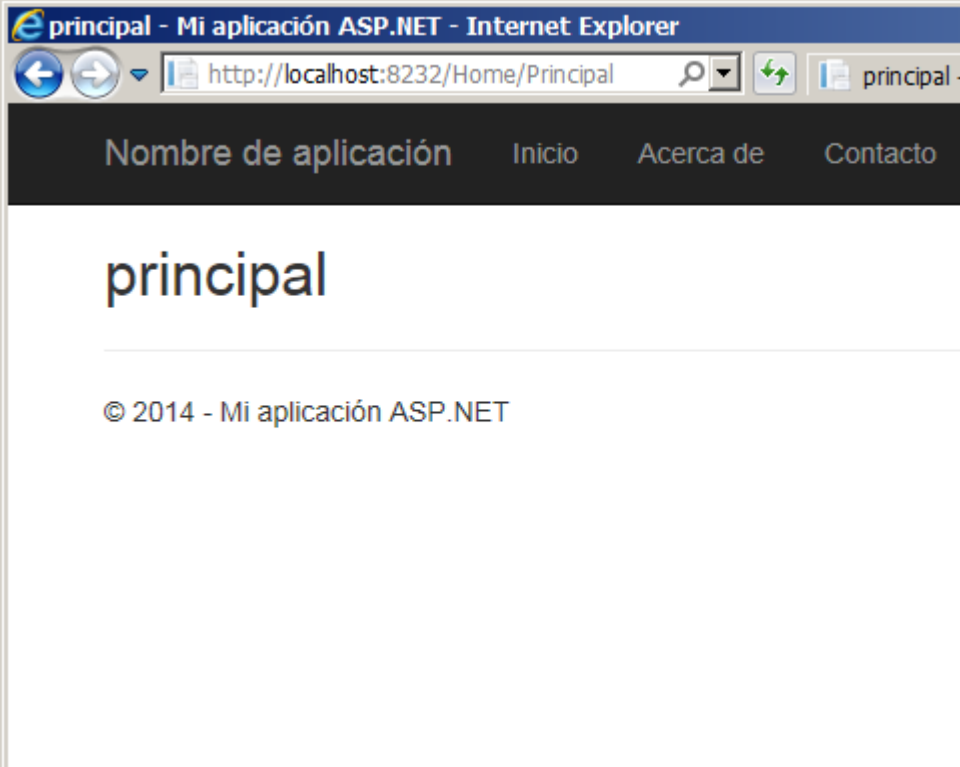


Ve al menú  
depurar y elige  
INICIAR  
DEPURACION o  
presiona F5

una vez que haya iniciado

Escribe (a mano)  
en la barra de  
direcciones, el  
punto de entrada  
/Home/Principal y  
presiona enter





el controlador tomará el CONTROL

El CONTROLADOR  
recibe la orden y deriva  
a la VISTA que  
indicaste en

```
return  
View("/Views/Custom/principal.cshtml");
```

haz creado un nuevo punto de entrada para tu vista

De ahora en más puedes editar la vista `principal.cshtml`, tu punto de entrada ya existe y siempre será el mismo.

Asimismo, puedes editar el punto de entrada y por el mismo camino `/Home/Principal` decidir presentar otra página

ejercita el concepto

Detén la aplicación.

Crea una nueva vista en la carpeta  
CUSTOM y nombrala Secundaria

Edita el archivo HomeController.chtml

ejercita el concepto

Modifica el punto de entrada  
reemplaza

```
return View("/Views/Custom/principal.cshtml");
```

**Por**

```
return View("/Views/Custom/secundaria.cshtml");
```

ejercita el concepto

Inicia nuevamente la aplicación (F5) y  
coloca en el browser la dirección  
/Home/Principal

Si todo es correcto, habrás visto que EL MISMO PUNTO DE  
ENTRADA ahora devolvió (RENDERIZÓ) la vista Secundaria



continua ejercitando

Crea una nueva carpeta llamada PRUEBAS.

Crea una vista con el nombre PRUEBA1.

Agrega en HomeController.cs un punto de entrada para la nueva vista. El punto de entrada debe tener un nombre distinto a los existentes.

hemos comprendido como se llaman las pantallas (VISTAS) a través de los Controladores (CONTROLLERS)

Es más, recuerdas que cuando iniciamos el proyecto elegimos la plantilla MVC ?, esto significa

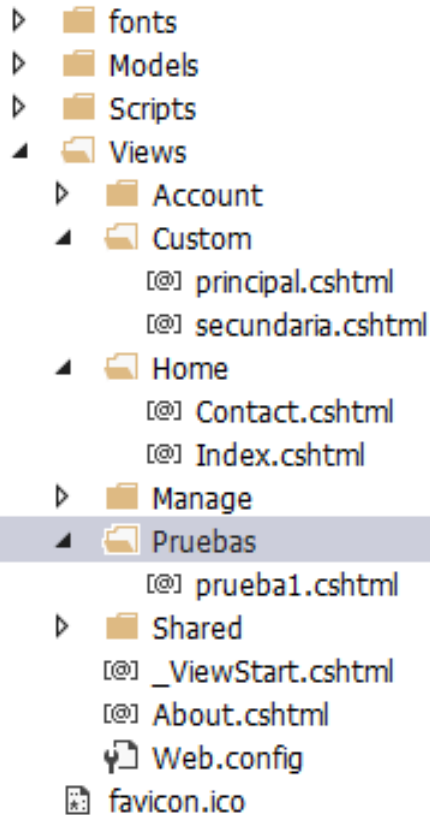
M = Modelo

V = Vistas

C = Controladores

Continuemos con HTML

El código HTML se edita en los archivos de tipo `chtml`, o sea, los ubicados en la carpeta **VIEWS**

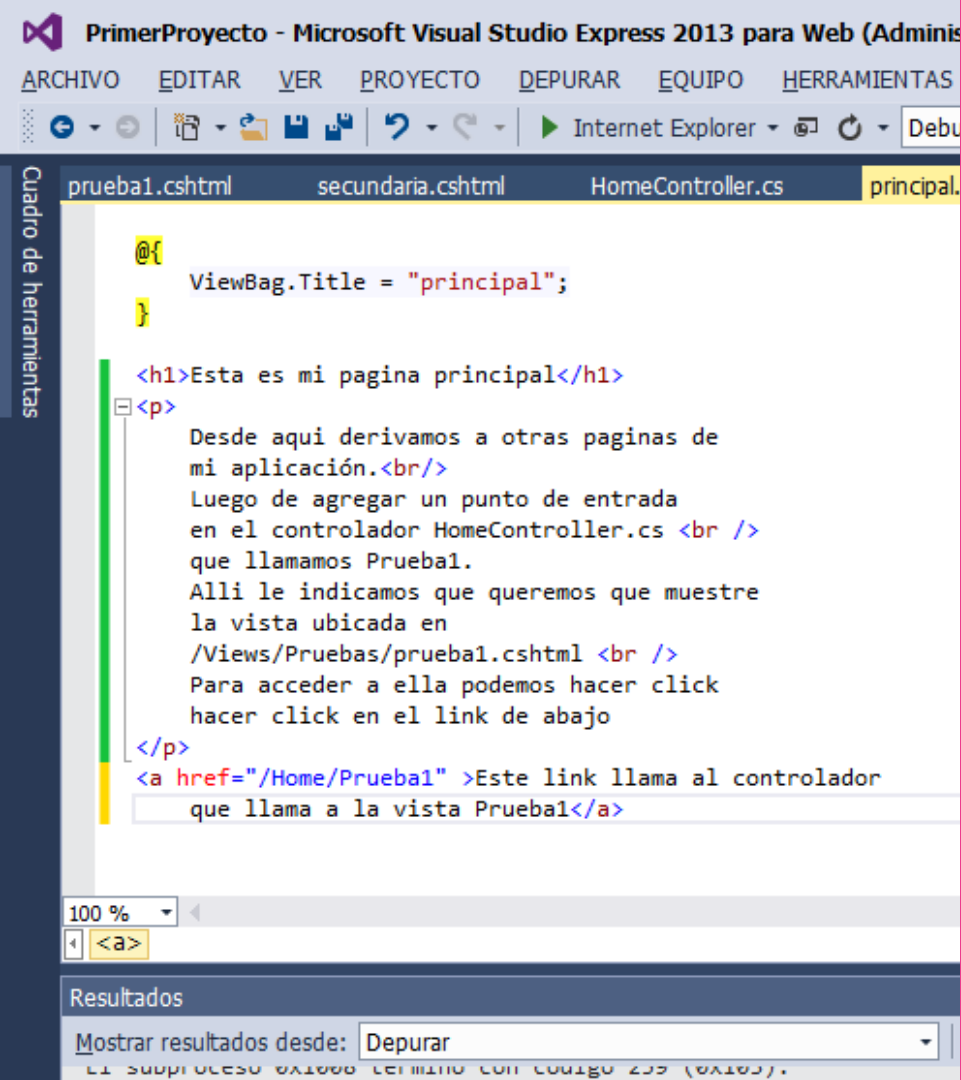


prueba los elementos HTML

En el curso, mencionamos los diferentes componentes HTML

Prueba cada uno de ellos en las diferentes páginas.

Para que ser más ordenado, utiliza diferentes vistas para que puedas probar los componentes por separado



edita principal.chnml

Agrega un título

<h1>

Un párrafo <p> y

un elemento link

<a href..>

a esta altura deberías haber logrado...

Agregar páginas (VISTAS)

Agregar puntos de entrada en el  
CONTROLADOR

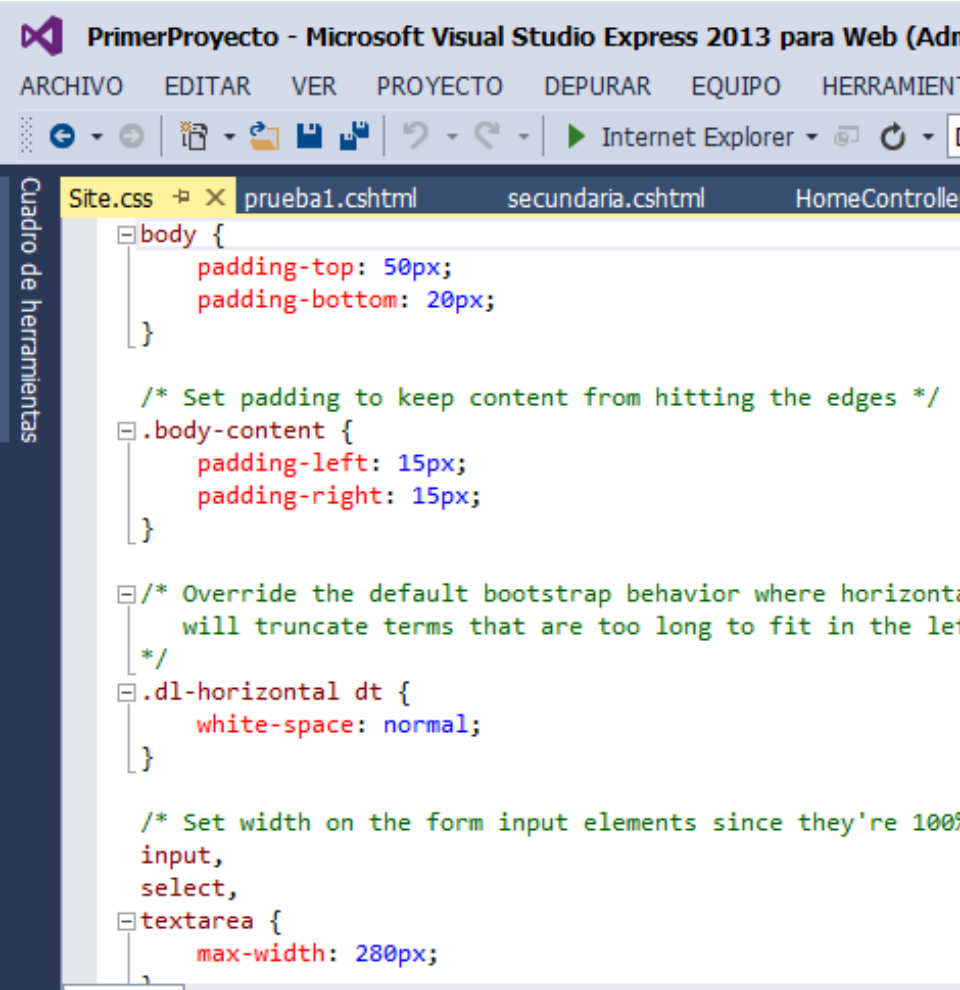
Editar las páginas en HTML y agregar  
elementos básicos

Controlar como se llaman las páginas  
usando elementos del tipo <a href..>

continua ejercitando

Intenta crear elementos del tipo:

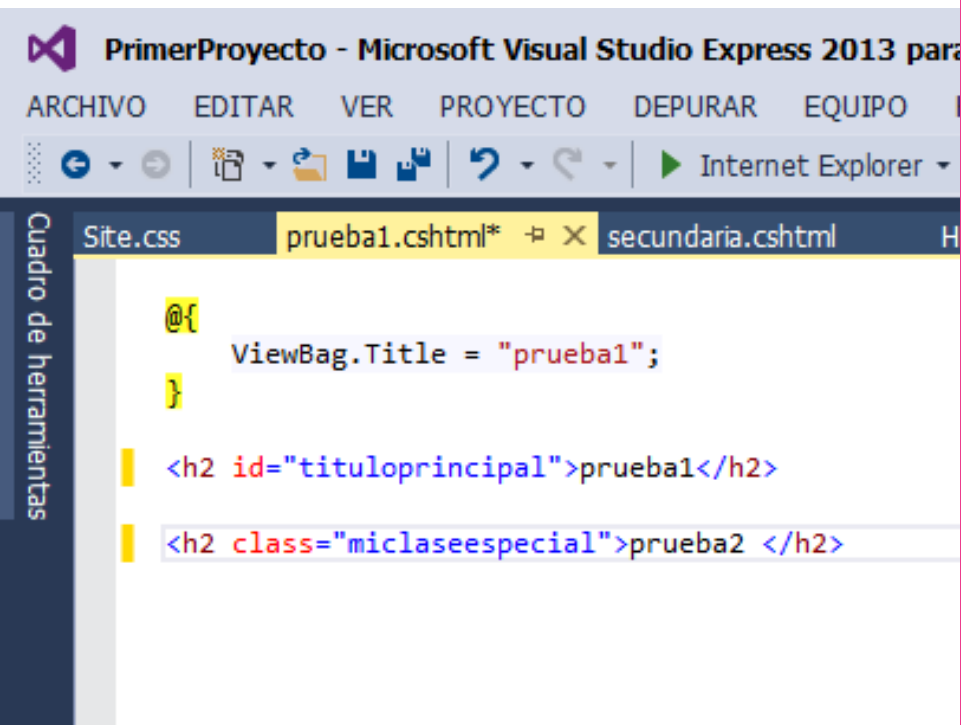
- SELECT (listas desplegables)
- INPUT TEXT (cajas de texto)
- TEXTAREA (cajas grandes de texto)
- A REF (links)
- UL (listas)



Editamos el estilo (colores y formas)

El estilo se define  
en el lenguaje CSS.  
El archivo por  
defecto se  
encuentra en la  
carpeta CONTENT  
y se llama Site.css

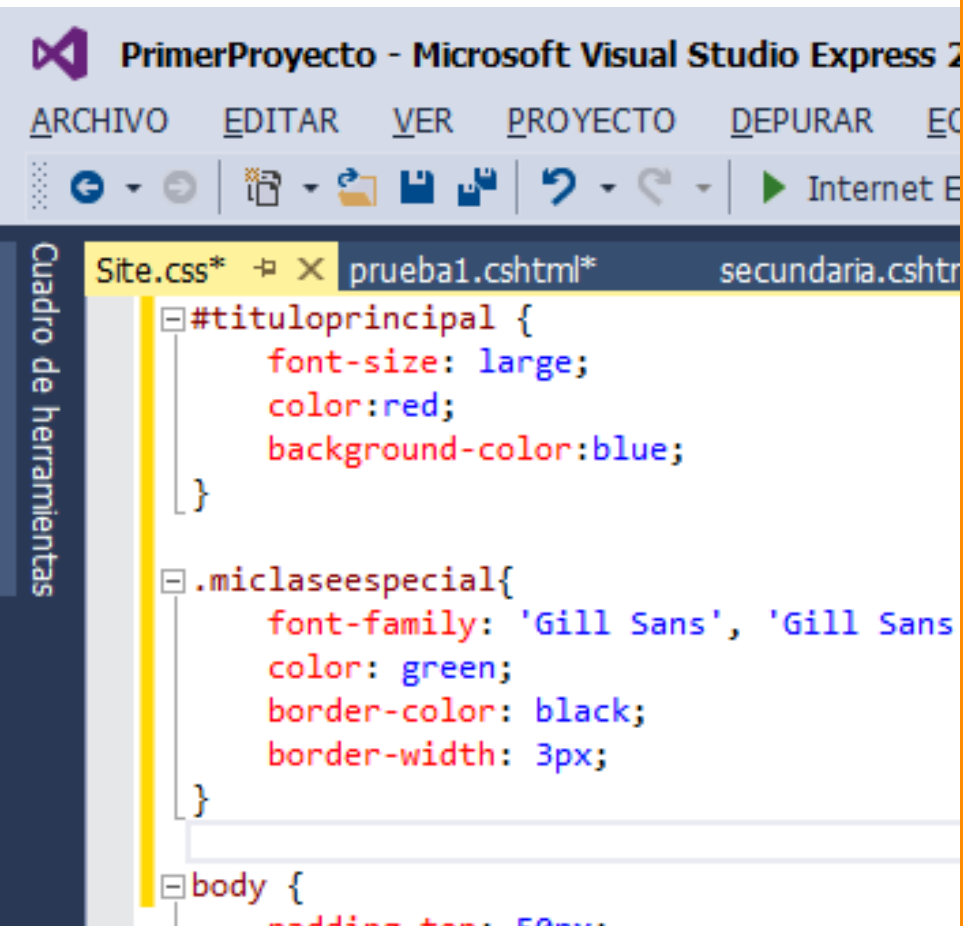




Recuerda el concepto que para editar un estilo tienes dos formas

Colocando un id al elemento HTML

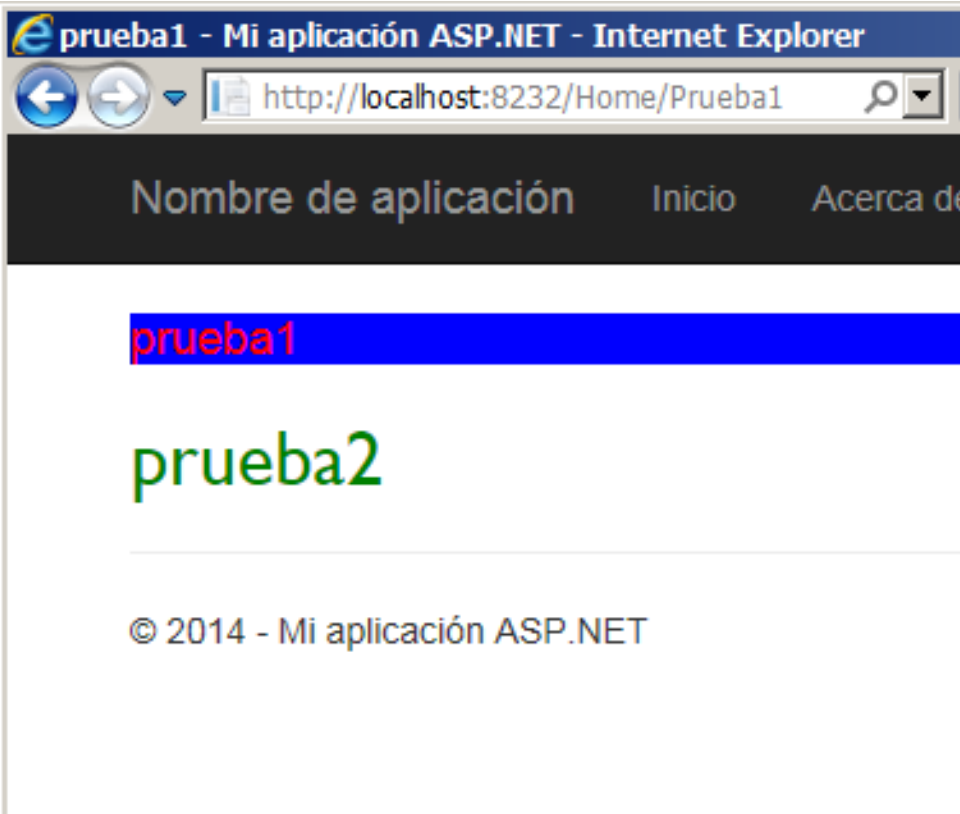
Colocando una CLASE al elemento html



luego, debes abrir el archivo de estilo Site.css

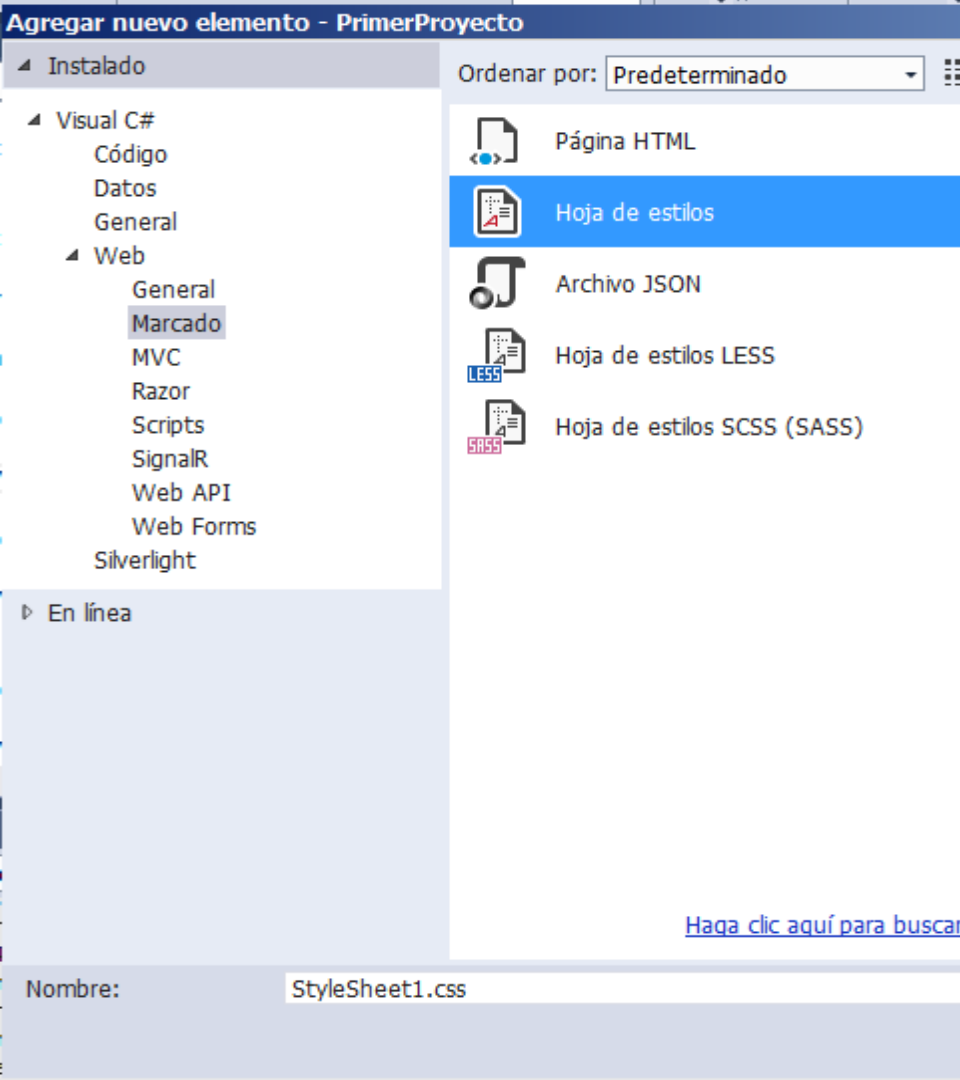
Coloca el estilo según el ID usando el prefijo #

Coloca el estilo según la CLASE usando el prefijo .



presiona la tecla Ejecutar  
( o presiona F5 )

Ejecuta la  
aplicación y  
visualiza los  
cambios en el  
estilo



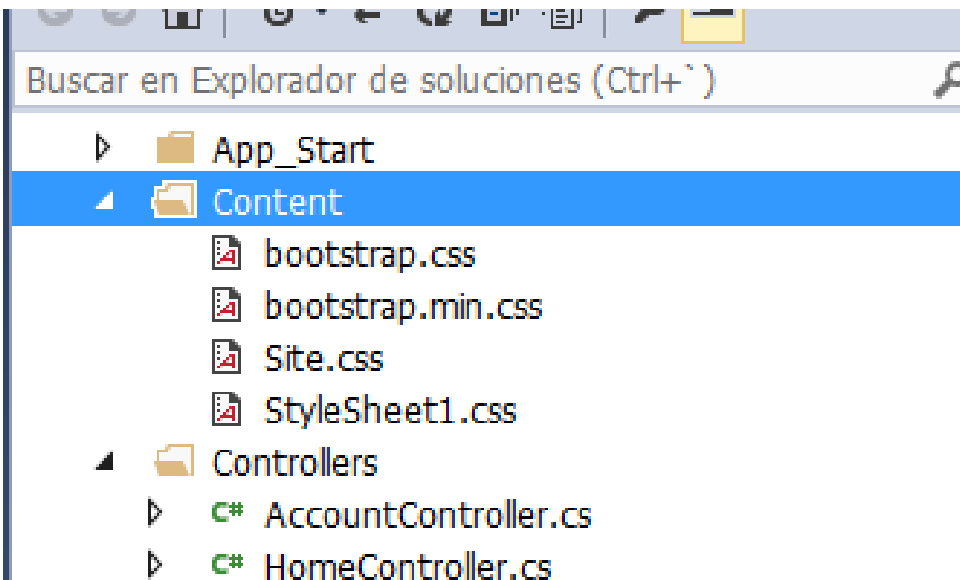
quieres tu propia hoja de estilo?

Puedes crear tu propia hoja de estilo, para esto, sobre la carpeta **CONTENT**, click derecho, agregar nuevo elemento

prueba diferentes estilos y formas para los elementos HTML

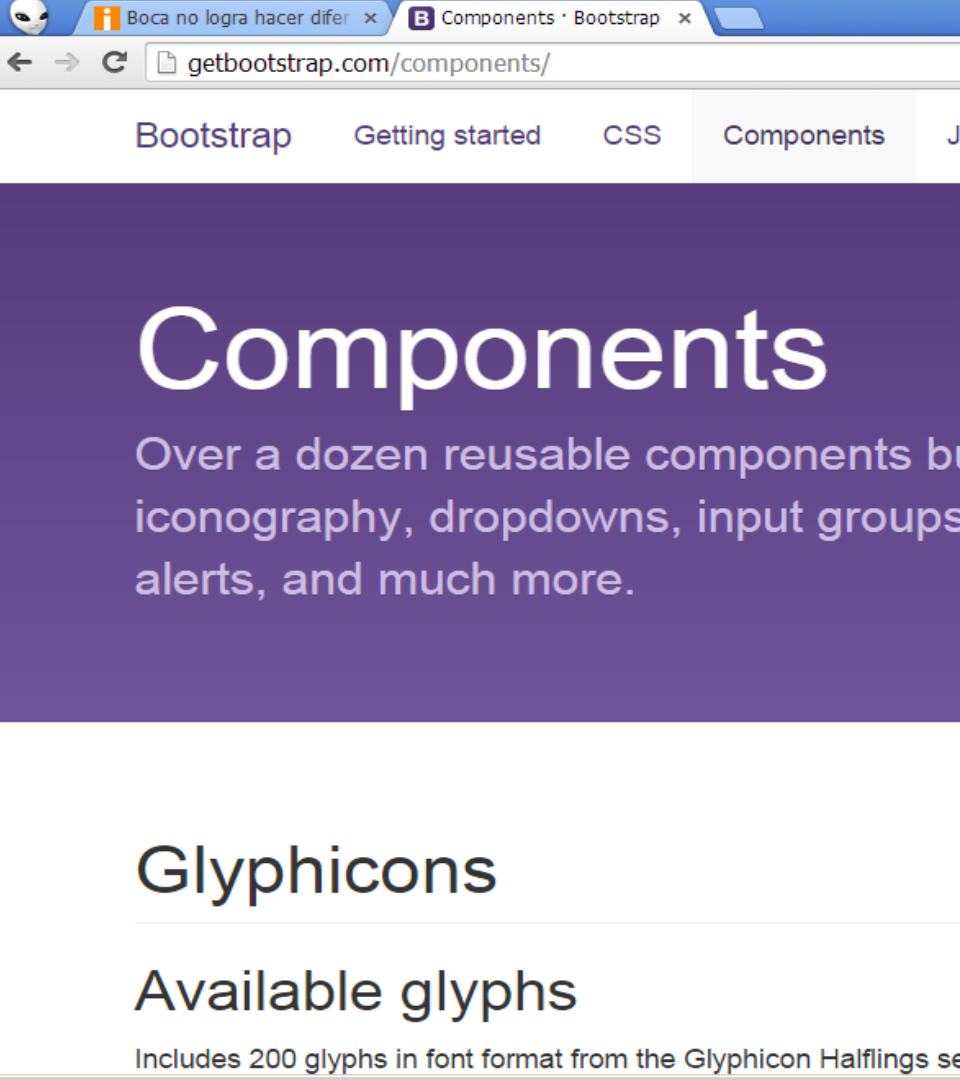
Coloca en el buscador web las  
siguientes claves  
«css basic styles»

Prueba cada uno, lo importante no es  
que quede bello, sino que sepas que  
se puede hacer con cada uno, para  
cuando debas usarlo



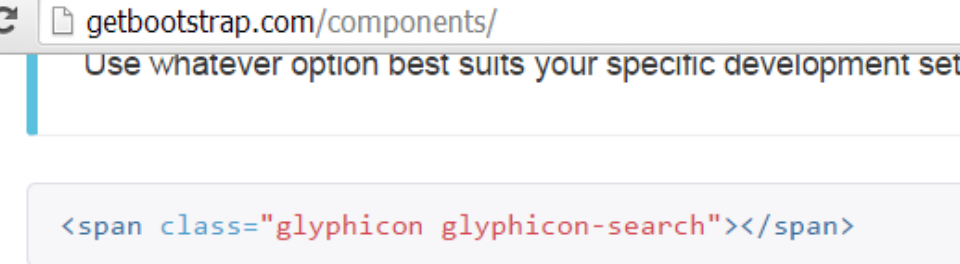
potencia tu aplicación

En Visual Studio 2013, tu aplicación trae incorporado BOOTSTRAP, que es una librería de estilos predefinidos



Busca la documentación en la web

Coloca en el  
buscador  
«Twitter Bootstrap  
v3.0  
Documentation»  
Empieza por  
components

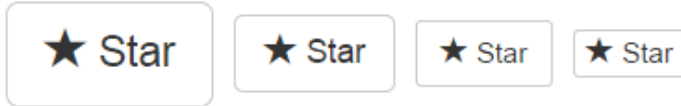


Bootstrap es muy simple de implementar

## Examples

Use them in buttons, button groups for a toolbar, navigation, or pr

### EXAMPLE



```
<button type="button" class="btn btn-default btn-lg">
  <span class="glyphicon glyphicon-star"></span> Star
</button>
```

Observa los ejemplos en HTML e inclúyelos en tus VISTAS.

Tu aplicación podrá adquirir un aspecto profesional



continúa ejercitando

Intenta comprender los diferentes elementos HTML y cómo aplicarles estilo

Utiliza Bootstrap para mejorar el aspecto

Define diferentes puntos de acceso y links entre las páginas para definir la estructura de tu sitio web

# Inteligencia en la interfaz de usuario

Javascript, uso de librerías externas

Aprendiendo a programar. Capítulo 4. Tutorial

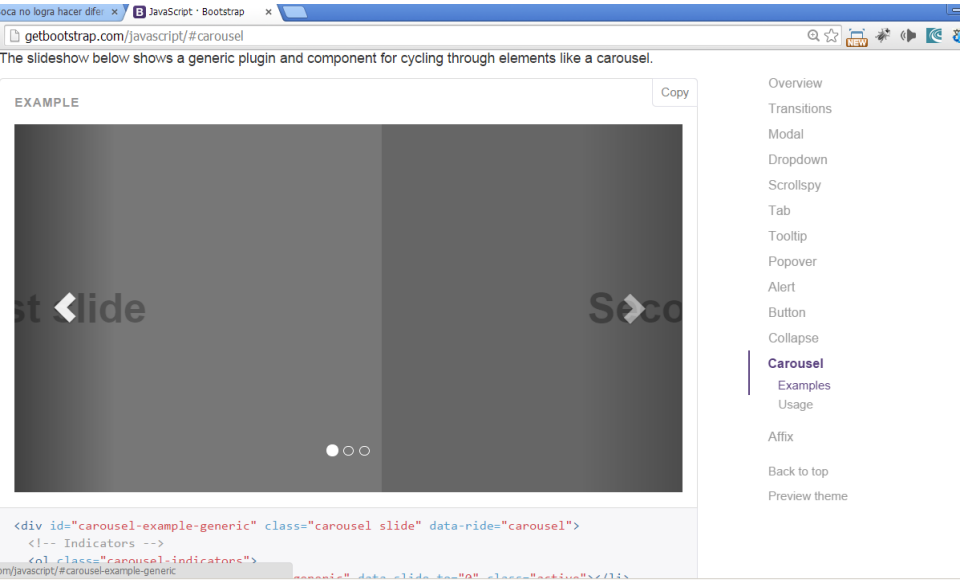
## objetivo

Crearemos una Aplicación Web de cero, utilizaremos Javascript para probar para agregar funciones a los componentes HTML  
Identificaremos librerías utiles externas de Javascript para incorporar a nuestro proyecto

conceptos previos

Javascript es el lenguaje de programación que te permite agregar funciones complejas al HTML.

Si te resulta dificultoso comprender que tipo de funciones, observa en la página de Bootstrap la sección JAVASCRIPT.



observa especialmente los  
siguientes controles de Bootstrap

Carousel  
Collapse  
Tab  
Modal  
DropDown

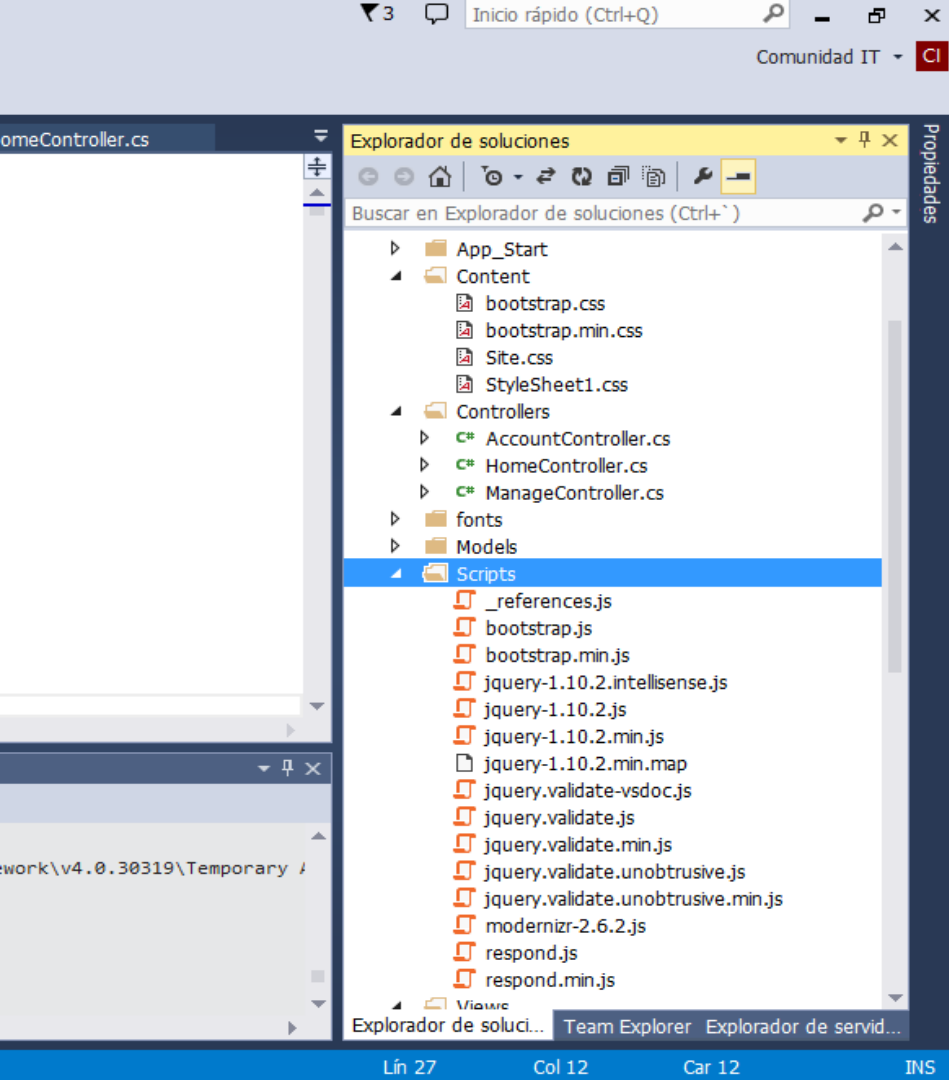
componentes predefinidos

Al crear un nuevo proyecto web en Visual Studio 2013, este incorpora automáticamente:

Javascript (no es necesario agregar nada)

JQUERY (Javascript Query)

BOOTSTRAP.JS (Sección Componentes Javascript)



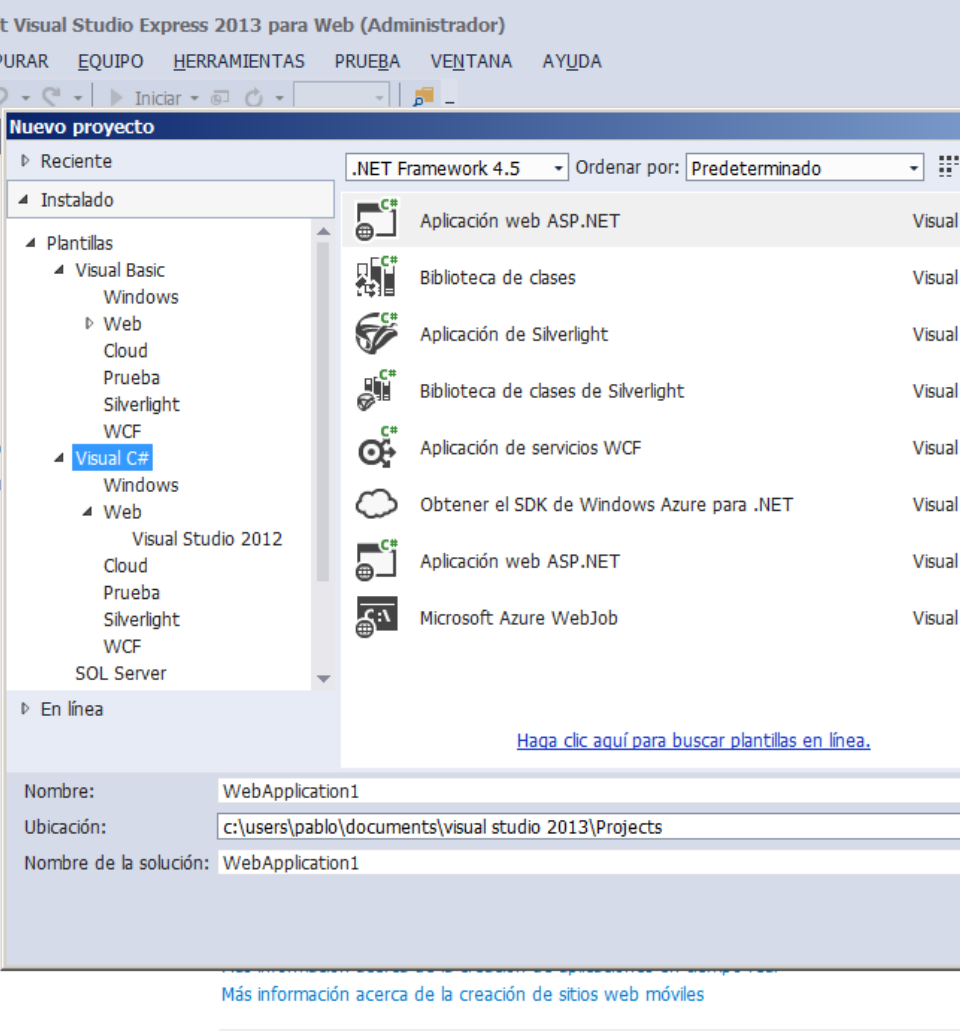
esto significa

Que podremos utilizar cualquiera de las características de estos controles, sin agregar nada. Porque ya vienen incluidos

comencemos con la ejercitación

Incorporaremos funciones  
JAVASCRIPT progresivamente  
Para esto, comencemos con un nuevo  
proyecto



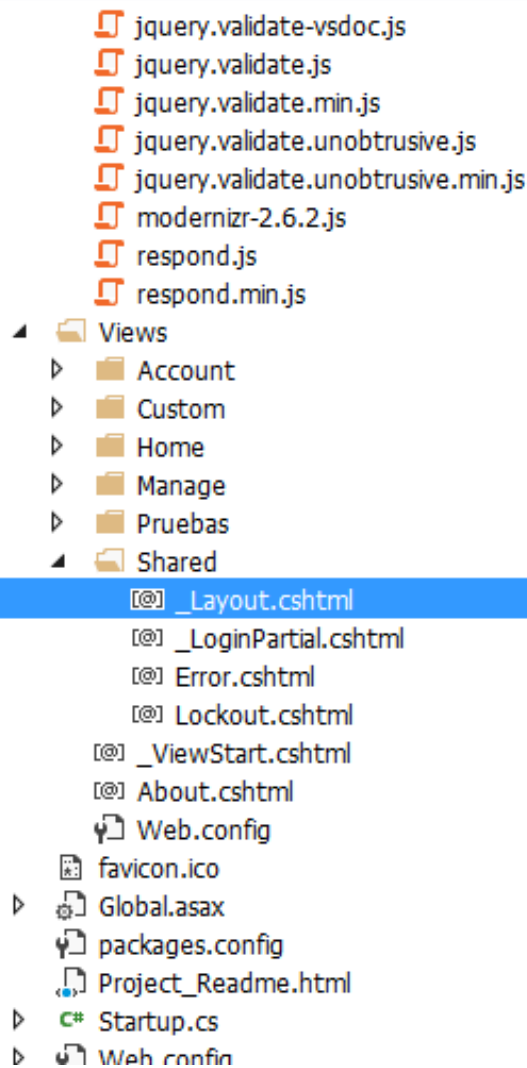


primero

Crea un nuevo  
proyecto de tipo  
Visual C# -  
Aplicación web  
ASP.NET y utiliza la  
plantilla simple  
MVC

antes de continuar precisamos un pequeño ajuste

Para simplificar el uso de Javascript y JQuery, precisamos que muevas a la cabecera la sección donde se incorpora JQuery a nuestra pagina.



ajusta las librerías JQuery

Ubica el archivo  
\_Layout.cshtml ubicado  
en Views / Shared.  
Edítalo y mueve la  
sección inferior hacia  
arriba como indica la  
imagen a continuación

bootstrap.css \_Layout.cshtml\* StyleSheet1.css Site.css prueba1.cshtml

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>@ViewBag.Title - Mi aplicación ASP.NET</title>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
</head>
<body>
<div class="navbar navbar-inverse navbar-fixed-top">
<div class="container">
<div class="navbar-header">
<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".nav
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
@Html.ActionLink("Nombre de aplicación", "Index", "Home", new { area = "" }, new {
</div>
<div class="navbar-collapse collapse">
<ul class="nav navbar-nav">
<li>@Html.ActionLink("Inicio", "Index", "Home")</li>
<li>@Html.ActionLink("Acerca de", "About", "Home")</li>
<li>@Html.ActionLink("Contacto", "Contact", "Home")</li>
</ul>
@Html.Partial("_LoginPartial")
</div>
</div>
<div class="container body-content">
@RenderBody()
<hr />
<footer>
</footer>
</div>
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</body>
</html>
```

80 %

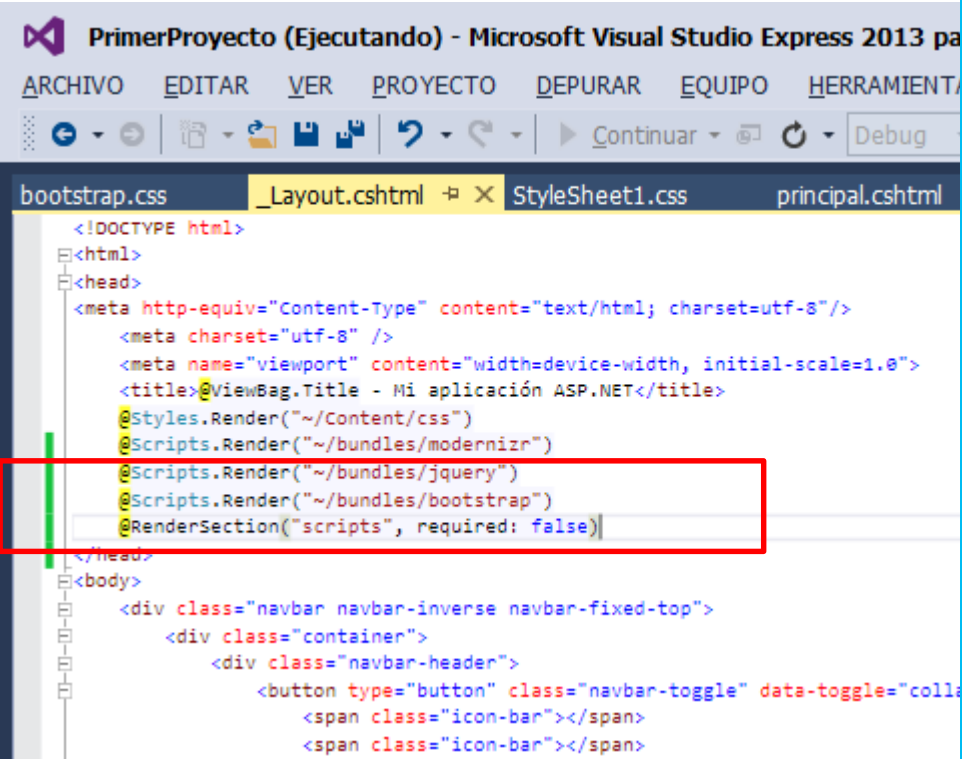
<html> <body> <div.container body-conte...>

## Explorador de soluciones

Buscar en Explorador de soluciones (Ctrl+')

- fonts
- Models
- Scripts
  - \_references.js
  - bootstrap.js
  - bootstrap.min.js
  - jquery-1.10.2.intellisense.js
  - jquery-1.10.2.js
  - jquery-1.10.2.min.js
  - jquery-1.10.2.min.map
  - jquery.validate-vsdoc.js
  - jquery.validate.js
  - jquery.validate.min.js
  - jquery.validate.unobtrusive.js
  - jquery.validate.unobtrusive.min.js
  - modernizr-2.6.2.js
  - respond.js
  - respond.min.js
- Views
  - Account
  - Custom
  - Home
  - Manage
  - Pruebas
  - Shared
    - \_Layout.cshtml
    - \_LoginPartial.cshtml
    - Error.cshtml
    - Lockout.cshtml

Explorador de soluciones Team Explorer Explorador de servidores



```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>@ViewBag.Title - Mi aplicación ASP.NET</title>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</head>
<body>
<div class="navbar navbar-inverse navbar-fixed-top">
<div class="container">
<div class="navbar-header">
<button type="button" class="navbar-toggle" data-toggle="coll...
<span class="icon-bar"></span>
<span class="icon-bar"></span>
```

traslada las líneas de JQuery y bootstrap

Así debería quedarte el header.

Asegúrate que estas líneas son las que estaban escritas en la parte inferior y que las has borrado de abajo

tu aplicación ya está lista para usar JQuery

Esta corrección de la ubicación de JQuery no es necesaria aplicarla en todos los proyectos.

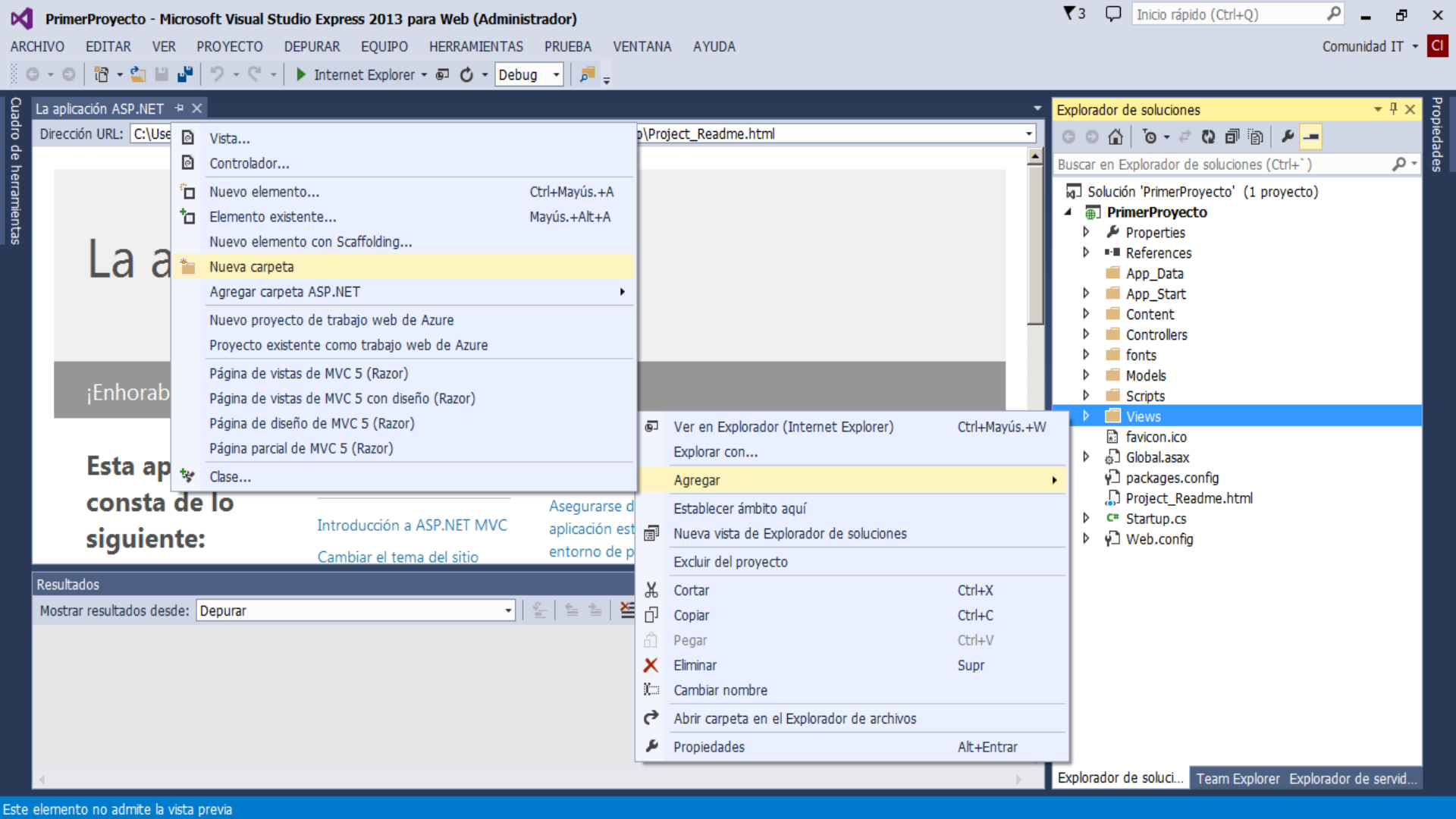
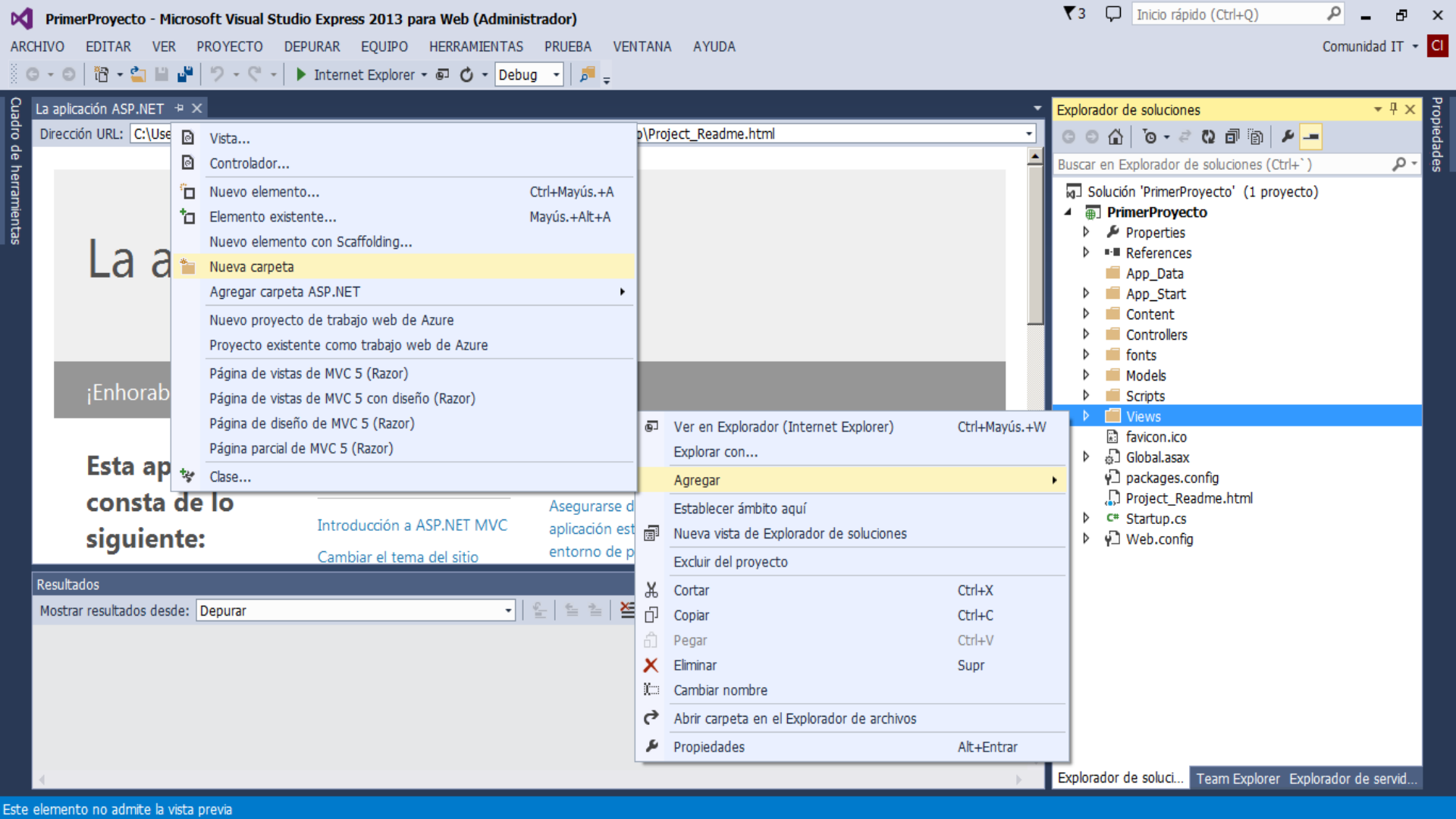
Lo hacemos en este tutorial para simplificarte las ejercitaciones que vamos a realizar.

crea una nueva carpeta sobre VIEWS

Posiciónate sobre Views, click derecho.

Agregar – Nueva Carpeta

Coloca el nombre «Custom»



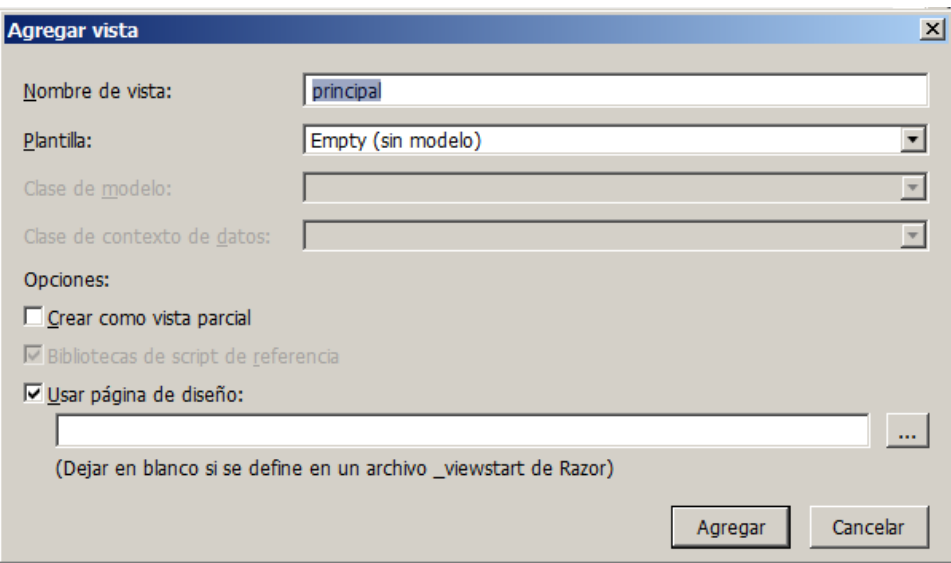


agrega una nueva vista (archivo cshtml), en la carpeta custom, recién creada

Posiciónate sobre Custom , click derecho.

Agregar – Nueva Vista

Coloca el nombre «Principal»



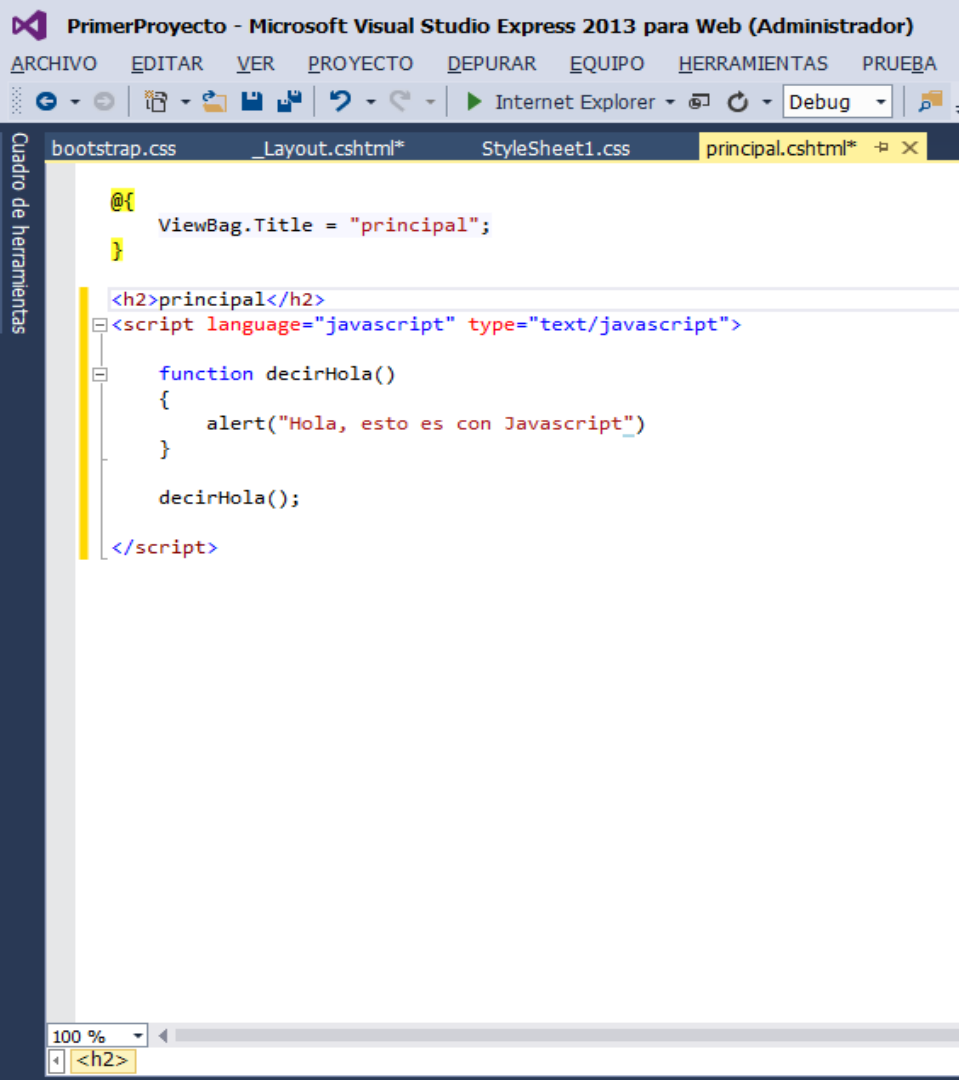
al momento de crear la vista  
te preguntará si quieres usar  
una plantilla

Elige la plantilla vacía  
(empty)

recuerda lo que vimos en el capítulo anterior

Para poder acceder a la vista principal, debemos crear un punto de entrada en el Controlador para la vista principal.

Repasa los conceptos desde la hoja 11 del anterior capítulo si no recuerdas como.

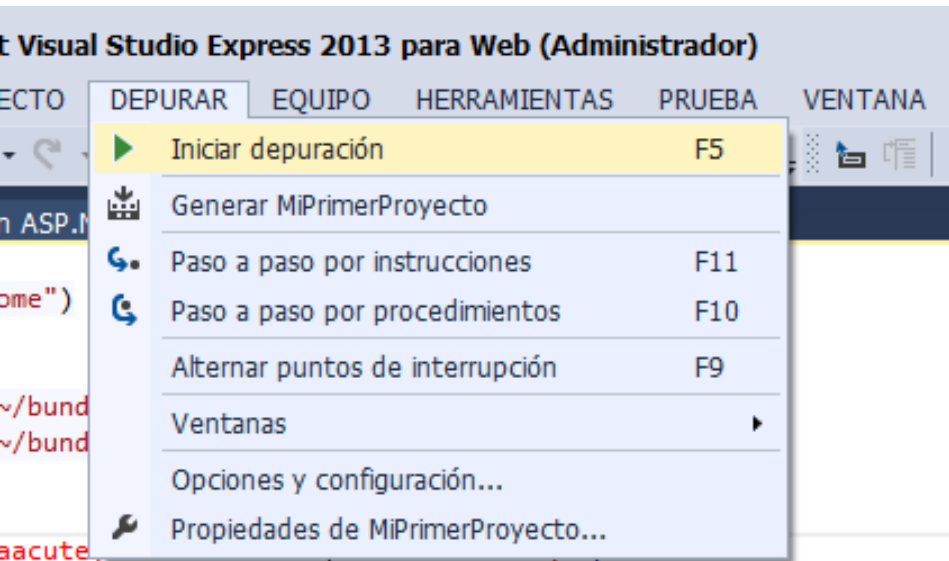


ubica la vista principal

Una vez ubicada  
incorpora el  
siguiente código



probemos la pagina

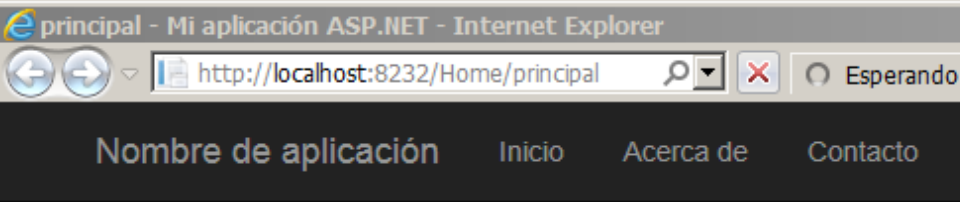


Ve al menú  
depurar y elige  
**INICIAR**  
**DEPURACION** o  
presiona F5

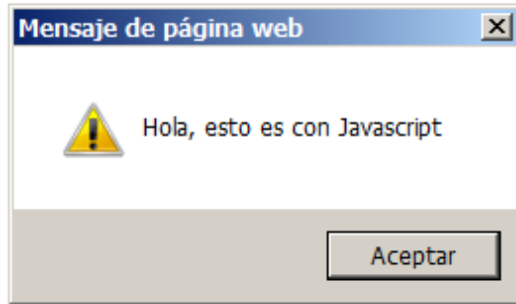
una vez que haya iniciado

Escribe (a mano)  
en la barra de  
direcciones, el  
punto de entrada  
/Home/Principal y  
presiona enter





principal



este debería ser tu resultado

Un alerta, en el  
centro de la  
pantalla, con el  
mensaje que  
habías escrito



bindear (ligar-enlazar) un evento javascript a un elemento HTML

Procedamos a disparar la misma alerta, esta vez, a través de un botón, cuando el usuario haga click.  
Vamos a usar Javascript / JQuery

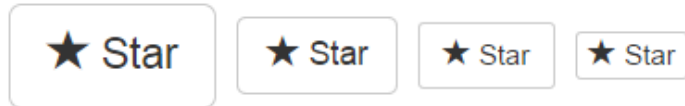
getbootstrap.com/components/  
Use whatever option best suits your specific development se

```
<span class="glyphicon glyphicon-search"></span>
```

## Examples

Use them in buttons, button groups for a toolbar, navigation, or p

### EXAMPLE

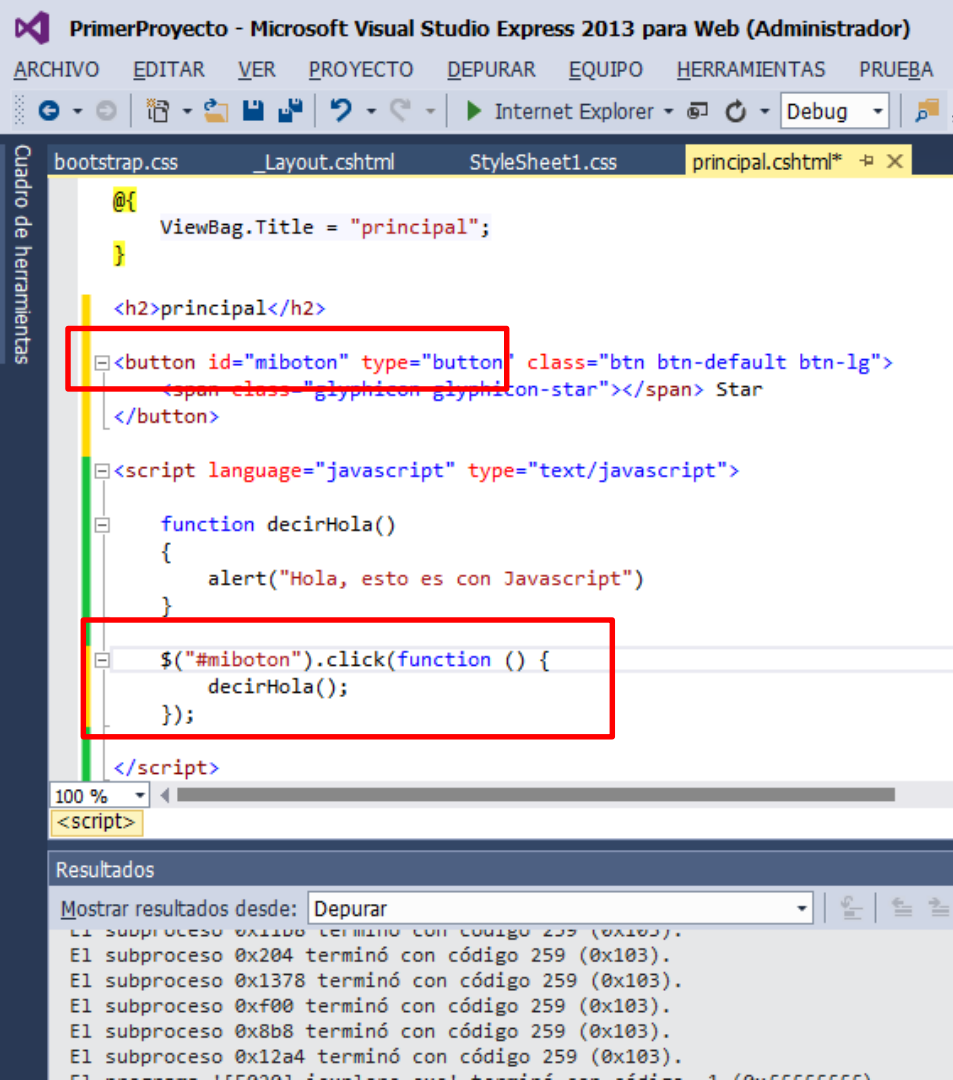


```
<button type="button" class="btn btn-default btn-lg">  
  <span class="glyphicon glyphicon-star"></span> Star  
</button>
```

Detén la aplicación, edita el archivo principal.cshtml

Incorpora un botón,  
para esto utilizamos  
el ejemplo HTML de  
botón de Bootstrap  
Si no recuerdas la  
forma, ve a la  
página, sección  
Componentes

```
@{  
    ViewBag.Title = "principal";  
}  
  
<h2>principal</h2>  
  
<button type="button" class="btn btn-default btn-lg">  
    <span class="glyphicon glyphicon-star"></span> Star  
</button>  
  
<script language="javascript" type="text/javascript">  
  
    function decirHola()  
    {  
        alert("Hola, esto es con Javascript")  
    }  
  
    decirHola();  
  
</script>
```

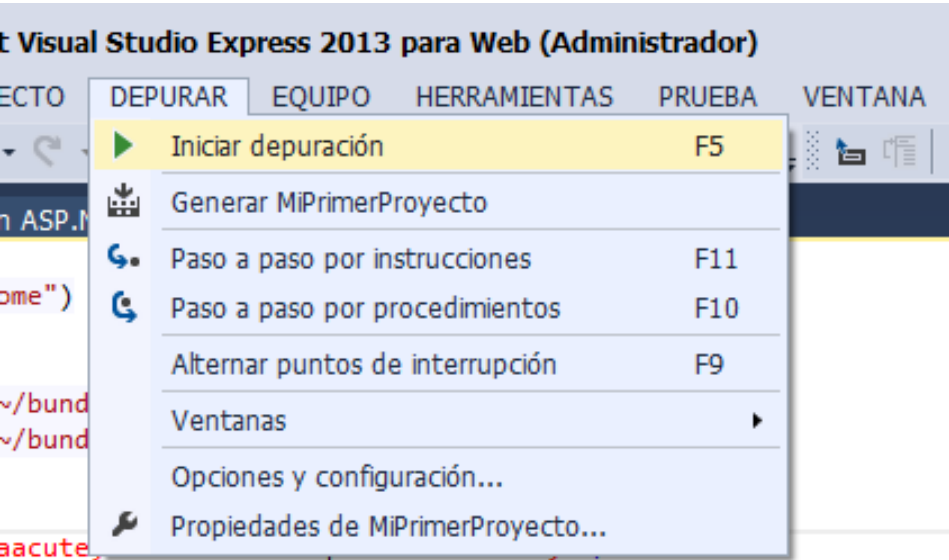


asocia el evento click a la función

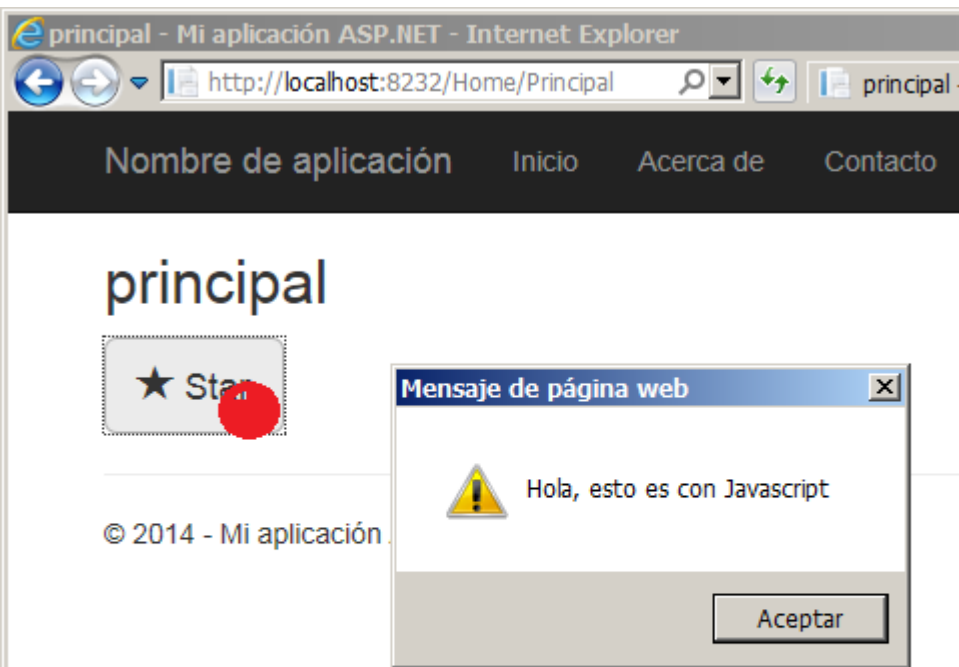
Asigna un id al  
elemento button  
HTML

Escribe en Javascript/  
Jquery que al hacer  
click, llame a la  
función

probemos la pagina



Inicia depuración.  
Esta vez la caja de  
alerta solo debería  
mostrarse cuando  
hagas click en el  
botón de la página



probemos la pagina

El alerta sólo  
aparece cuando  
haces click

continuemos ejercitando

Probaremos otros eventos JQuery para disparar el alerta.

La idea es comprender que acciones podemos atrapar para lanzar una función. Aunque lanzemos la misma alerta, presta especial atención a la acción que la dispare.

PrimerProyecto - Microsoft Visual Studio Express 2013 para Web (Administrador)

ARCHIVO EDITAR VER PROYECTO DEPURAR EQUIPO HERRAMIENTAS PRUEBA

bootstrap.css \_Layout.cshtml StyleSheet1.css principal.cshtml\*

```
@{
    ViewBag.Title = "principal";
}

<h2>principal</h2>
<button id="miboton" type="button" class="btn btn-default btn-lg">
    <span class="glyphicon glyphicon-star"></span> Star
</button>

<script language="javascript" type="text/javascript">
    function decirHola()
    {
        alert("Hola, esto es con Javascript")
    }

    $("#miboton").click(function () {
        decirHola();
    });
</script>
```

100 % <script>

Resultados

Mostrar resultados desde: Depurar

E1 subproceso 0x1106 terminó con código 259 (0x103).  
E1 subproceso 0x204 terminó con código 259 (0x103).  
E1 subproceso 0x1378 terminó con código 259 (0x103).  
E1 subproceso 0xf00 terminó con código 259 (0x103).  
E1 subproceso 0x8b8 terminó con código 259 (0x103).  
E1 subproceso 0x12a4 terminó con código 259 (0x103).  
E1 programa 1552201 terminó con código 1 (0x00000001).

asociemos otro evento

Detengamos la aplicación.  
Editemos principal.chtml  
Modifiquemos la palabra click por otros eventos



evalúa otros eventos

Si reemplazas click por siguiente

- `dblclick`
- `hover`
- `mousedown`
- `mouseup`

es fácil

Detengamos la  
aplicación.  
Editemos  
principal.chtml  
Modifiquemos la  
palabra click por otros  
eventos

```
<script language="javascript" type="text/javascript">  
  
    function decirHola() {  
        alert("Hola, esto es con Javascript")  
    };  
  
    $("#miboton").dblclick(function () {  
        decirHola();  
    });  
</script>
```

sigue explorando

Usa el buscador web con las claves  
« jquery events »

En la página oficial de JQuery,  
encontrarás detalladamente todos los  
eventos disponibles

componentes más complejos

Diseñemos un Carousel.

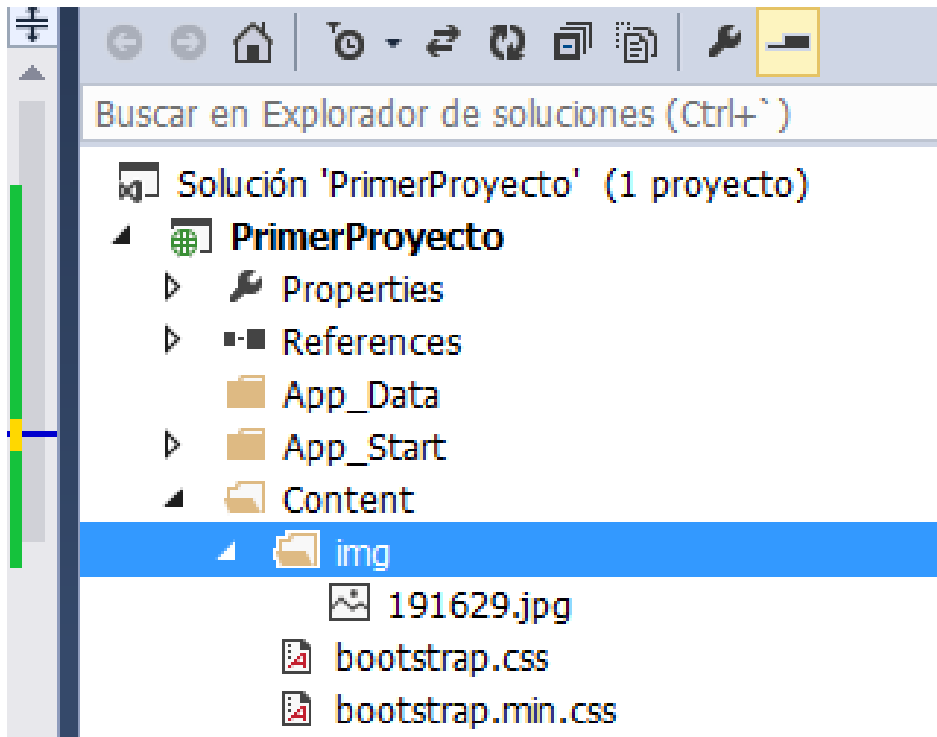
El Carousel es el componente típico para mostrar una serie de imágenes.

Para esto necesitamos que elijas 4 o 5 fotografías o imágenes que tengas en tu computadora. Haremos un Carousel de imagenes con ellas

Incorpora las imágenes al proyecto

En el explorador de soluciones, crea la carpeta img dentro de Content.

Elige una o varias fotos de tu compu y arrástralas hasta la carpeta



- ▶ Scripts
- ▲ Views
  - ▶ Account
  - ▶ Custom
  - ▶ Home
  - ▶ Manage
  - ▲ Pruebas

[@] prueba1.cshtml

- ▲ Shared
  - [@] \_Layout.cshtml
  - [@] \_LoginPartial.cshtml
  - [@] Error.cshtml
  - [@] Lockout.cshtml
- [@] \_ViewStart.cshtml
- [@] About.cshtml

Crea una vista adicional para probar

Crea una carpeta  
llamada Pruebas en  
Views, crea una vista  
llamada Prueba1

agrégle un punto de entrada

```
public class HomeController : Controller
{
    public ActionResult Prueba1()
    {
        return View("/Views/Pruebas/prueba1.cshtml");
    }
    public ActionResult Principal()
    {

```

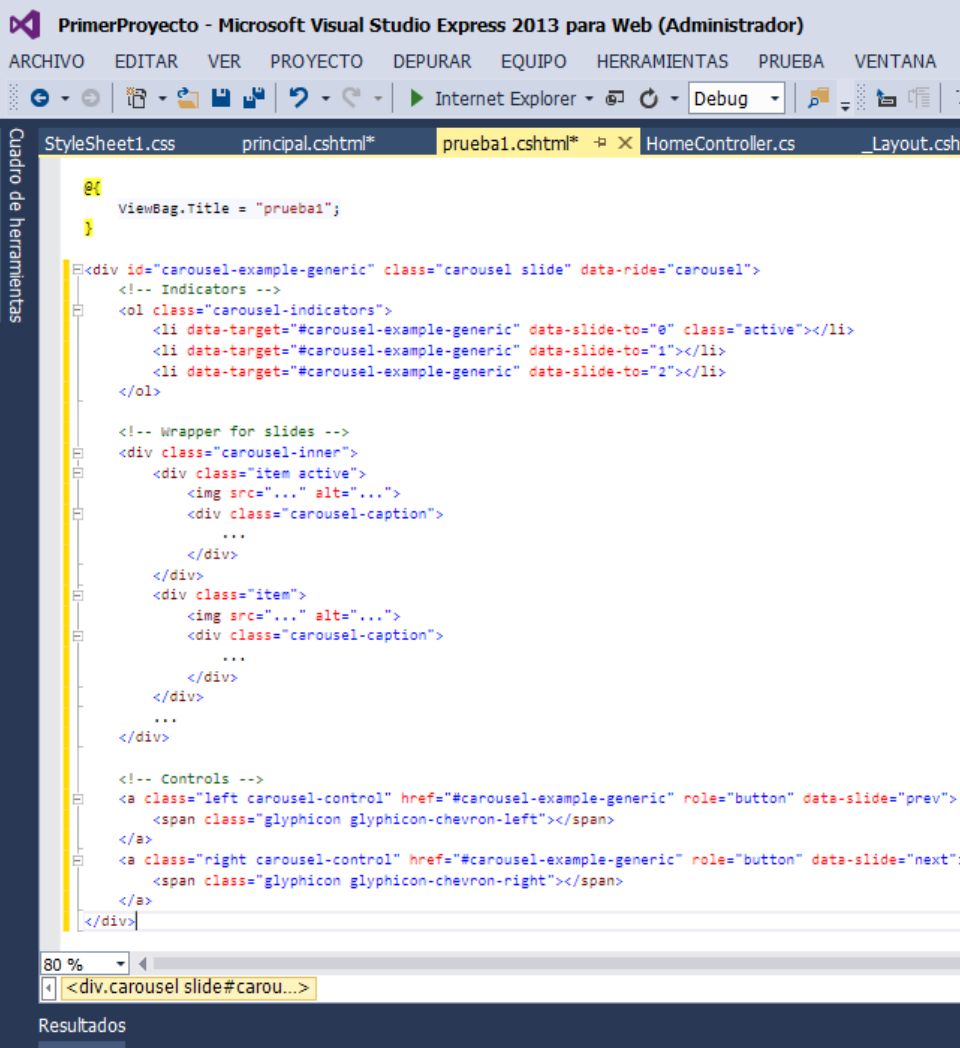
En el Controlador  
HomeController.cs  
Agrega el punto de  
entrada para la vista  
prueba1.  
Recuerda que para  
probar la ruta será  
/Home/Prueba1

consulta la documentación del Carousel

Dirígete a la página web de Bootstrap y consulta la documentación del Carousel, copia y pega el código HTML en tu página prueba1.cshtml

<http://getbootstrap.com/javascript/#carousel>





debería quedar algo así

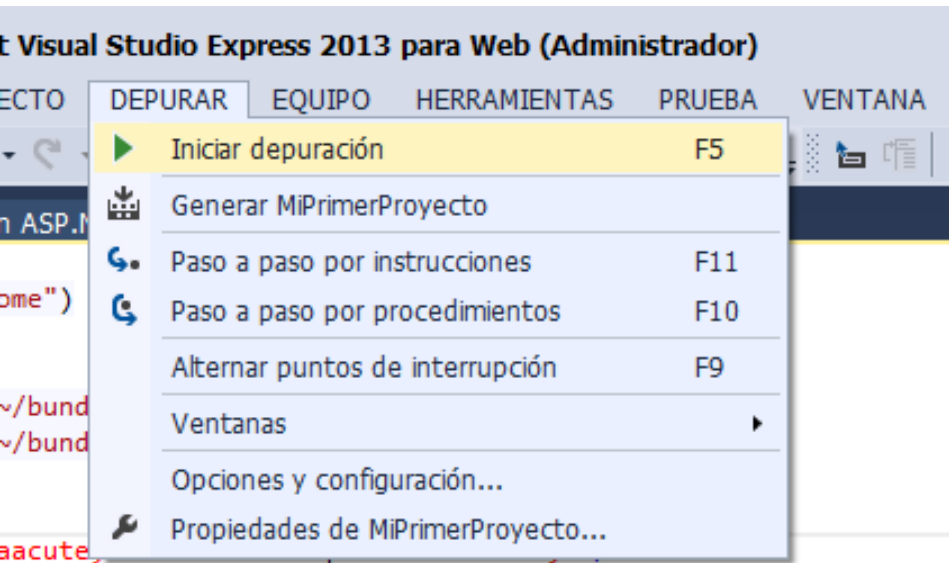
Dentro del código HTML ubica los elementos del tipo IMG  
Y completa la dirección completa de tus imágenes

```
<div class="carousel-inner">
  <div class="item active">
    
    <div class="carousel-caption">
      ...
    </div>
  </div>
  <div class="item">
    
    <div class="carousel-caption">
      ...
    </div>
  </div>
</div>
```

Indica las imágenes

Recuerda que las imágenes son las que guardaste en la carpeta Content/img  
Los nombres de las imágenes se corresponden con los de la carpeta

cruza los dedos y a probar



Inicia depuración.  
Si todo está bien,  
debería mostrar  
tus imágenes.  
Recuerda que el  
punto de entrada  
es Home/Prueba1



continúa explorando

Intenta implementar por ti mismo  
los controles

Tab, Modal, Dropdown y ToolTip

Dirígete a la documentación de la  
página y modifica lo que  
corresponda

validadores

Adicionalmente el template de proyectos Web de Visual Studio 2013, trae incorporada la librería

<http://jqueryvalidation.org>

Estos permiten verificar que los datos de un formulario cumplan determinadas condiciones y lance un alerta en caso de error

Microsoft

NGO DAY 2014  
LA TECNOLOGIA AL SERVICIO DE LAS CAUSAS SOCIALES

CONTACTANOS

Nombre  
Por favor complete este campo.

Organización  
Por favor complete este campo.

E-mail  
Por favor ingrese un correo electrónico válido.

Teléfono  
Por favor complete este campo.

Mensaje  
Por favor complete este campo.

ENVIAR

un ejemplo de como funcionan?

En este formulario, si no ingresas algun dato, lanza un mensaje de error al usuario.  
Todo esto con Javascript

ejercitemos

Crearemos un formulario HTML con varios campos del tipo input a los cuales les incorporamos validadores



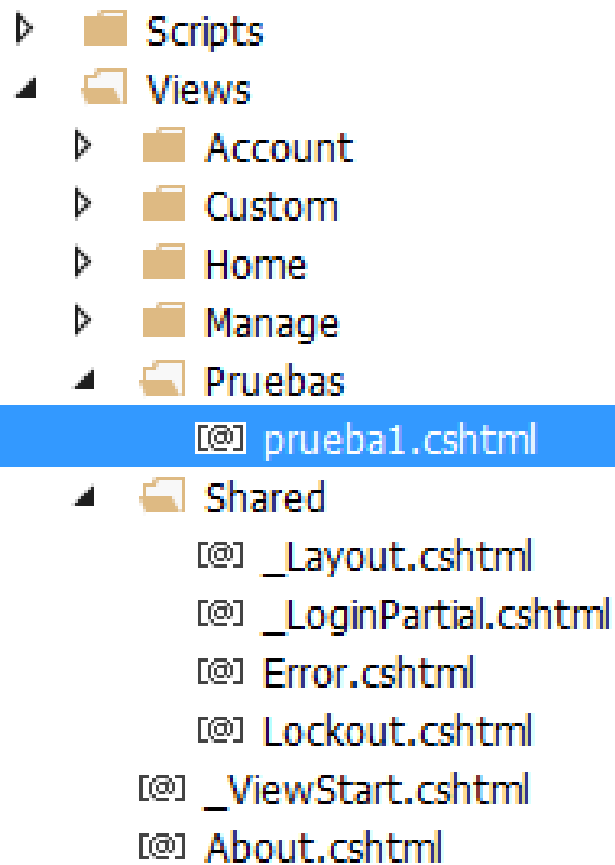
Precisamos agregar la librería

Edita el archivo  
\_layout.cshtml, para  
agregar a la sección  
HEAD, la librería de  
validadores

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>@ViewBag.Title - Mi aplicación ASP.NET</title>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
```

```
<script type="text/javascript" src="~/Scripts/jquery.validate.min.js"></script>
</head>
```

```
<body>
<div class="navbar navbar-inverse navbar-fixed-top">
<div class="container">
```



Crea una vista adicional para probar

Crea una carpeta  
llamada Pruebas en  
Views, crea una vista  
llamada prueba2 (al  
lado de prueba1)

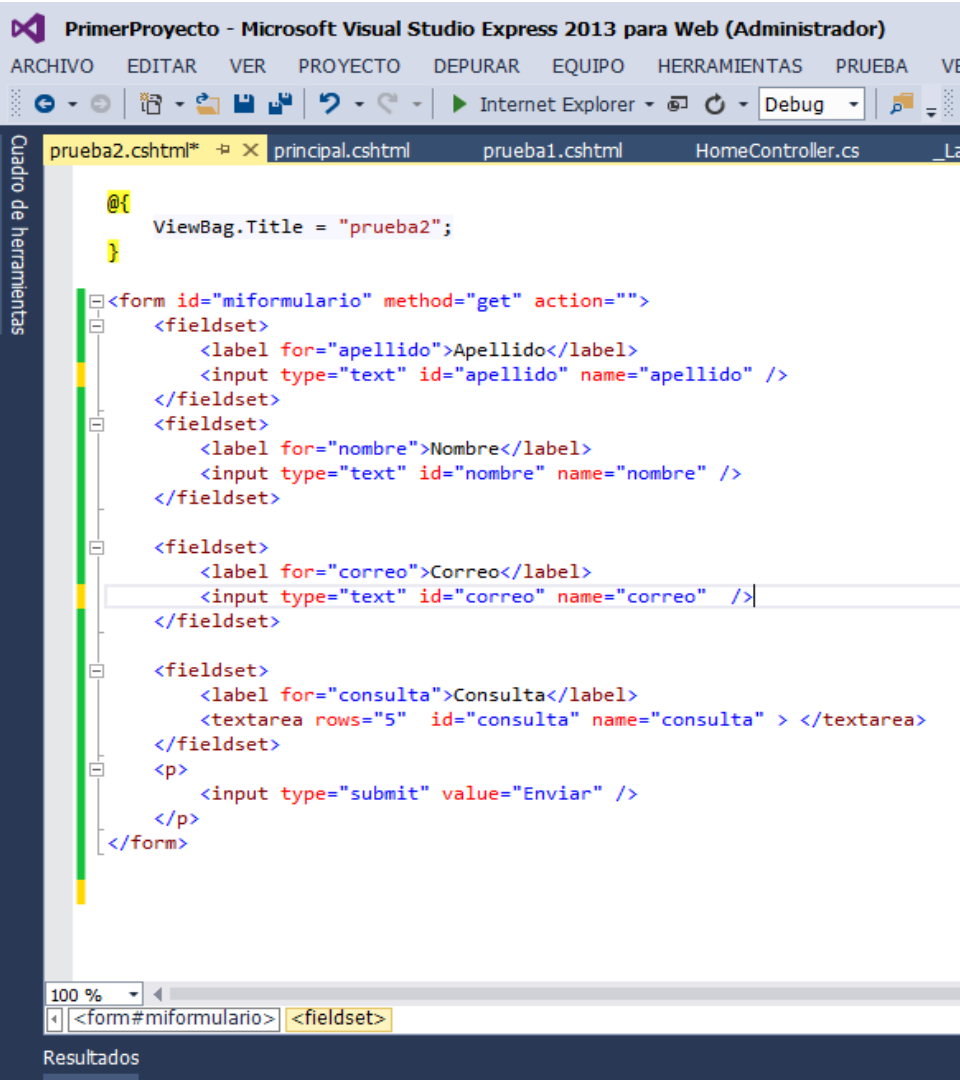
```
public class HomeController : Controller
{
    public ActionResult Prueba1()
    {
        return View("/Views/Pruebas/prueba1.cshtml");
    }
    public ActionResult Principal()
    {

```

agrégle un punto de entrada

En el Controlador  
HomeController.cs

Agrega el punto de  
entrada para la vista  
idéntico a prueba1  
pero para prueba2  
En este caso el punto  
de entrada será  
/Home/Prueba2



creamos un formulario dentro de prueba2

Ingresamos los campos indicados en la imagen

PrimerProyecto - Microsoft Visual Studio Express 2013 para Web (Administrador)

ARCHIVO EDITAR VER PROYECTO DEPURAR EQUIPO HERRAMIENTAS PRUEBA VEN

prueba2.cshtml principal.cshtml prueba1.cshtml HomeController.cs \_Layout

```
@{
    ViewBag.Title = "prueba2";
}

<form id="miformulario" method="get" action="">
    <fieldset>
        <label for="apellido">Apellido</label>
        <input type="text" id="apellido" name="apellido" required/>
    </fieldset>
    <fieldset>
        <label for="nombre">Nombre</label>
        <input type="text" id="nombre" name="nombre" />
    </fieldset>
    <fieldset>
        <label for="correo">Correo</label>
        <input type="text" id="correo" name="correo" required />
    </fieldset>
    <fieldset>
        <label for="consulta">Consulta</label>
        <textarea rows="5" id="consulta" name="consulta" > </textarea>
    </fieldset>
    <p>
        <input type="submit" value="Enviar" />
    </p>
</form>

<script type="text/javascript">
    $("#miformulario").validate();
</script>
```

100% <script>

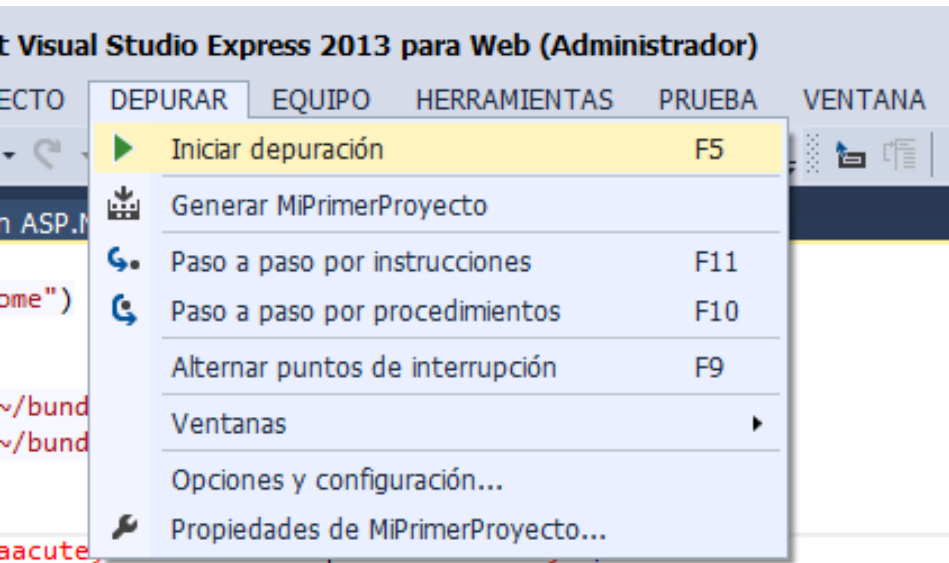
agregamos los validadores

Usando la documentación de la página

<http://jqueryvalidation.org/documentation>

Agregamos el validador required y el llamado a la función

ejecutamos la prueba



Inicia depuración.  
Recuerda que el  
punto de entrada  
es Home/Prueba2

continua ejercitando

Evalúa los validadores para email, teléfono, fecha, valor máximo y mínimo.

Los validadores son esenciales en cualquier formulario, procura conocer las capacidades de todos.

# El Servidor Web

El lenguaje C# y su integración con la interfaz

Aprendiendo a programar. Capítulo 5. Tutorial



objetivo

Utilizaremos otros operadores de C#  
Capturaremos información de la  
pantalla y la enviaremos al  
controlador para efectuar acciones.

conceptos previos

Es posible leer datos de una vista  
cshtml.

Esta lectura consiste en los siguientes  
pasos

- 1) El usuario tipea una dirección en el  
browser (o accede a través de un link)  
a un punto de entrada de un  
controlador

pasos para leer los datos de pantalla

2) El controlador recibe el pedido y retorna (con return view) la vista correspondiente

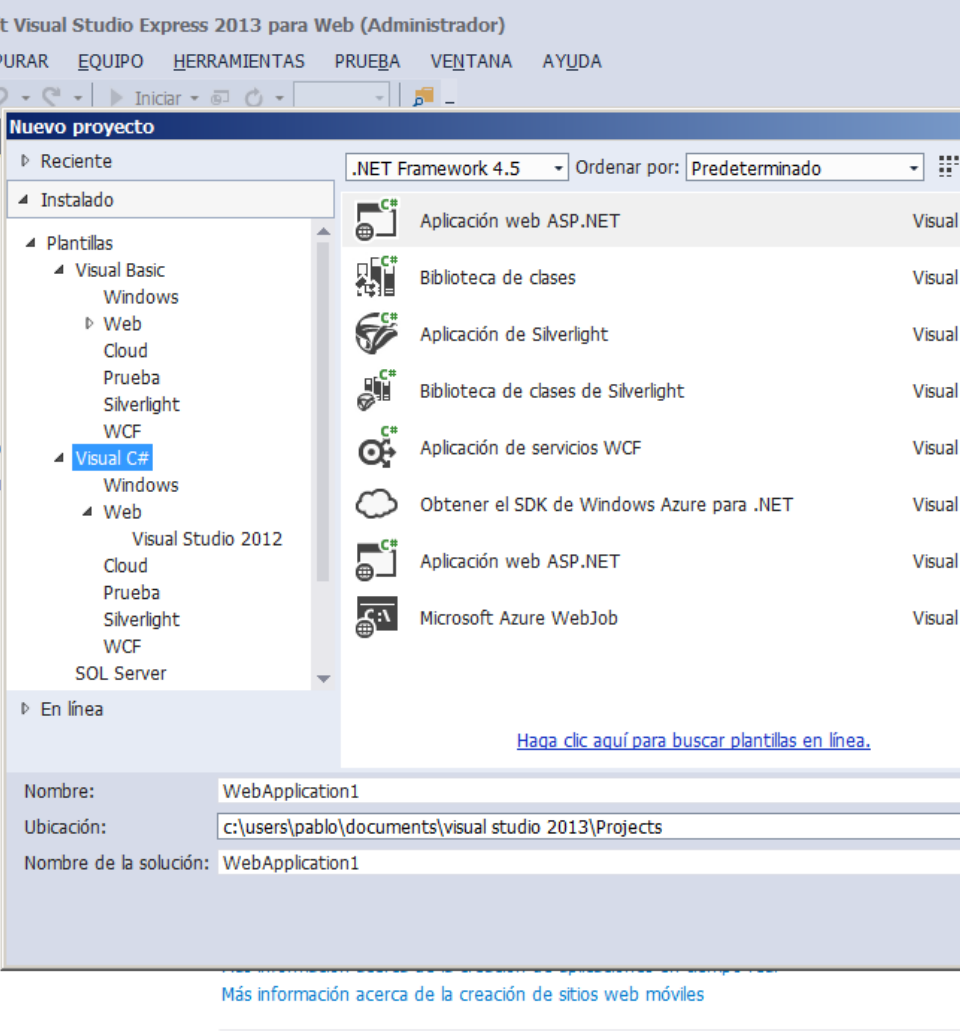
3) La vista posee dentro del HTML un formulario, en blanco o con datos precargados, adicionalmente tiene un botón del tipo INPUT TYPE SUBMIT

pasos para leer los datos de pantalla

4) El botón INPUT TYPE SUBMIT tiene escrito un punto de entrada a un controlador que es DIFERENTE al punto de entrada que se uso la primera vez. Este punto SOLO SE OCUPA DE LEER LOS DATOS. NADA MAS

pasos para leer los datos de pantalla

5) El controlador LEE Los datos, efectúa operaciones y devuelve una respuesta a través de un return view. Esta ultima vista solo contiene un mensaje de resultado.



comencemos

Crea un nuevo  
proyecto de tipo  
Visual C# -  
Aplicación web  
ASP.NET y utiliza la  
plantilla simple  
MVC

crea las siguientes secciones

Crea una carpeta dentro de views  
llamada Formularios

Crea una vista dentro de Formularios  
llamado Contacto

Edita Contacto.cshtml y crea elementos  
html como el formulario visto en el  
capítulo III, con nombre, apellido y  
dirección de email

continua con

Crea un controlador llamado Formas  
Dentro de el, crea un punto de entrada  
llamado Contacto y retorna la vista  
`/Views/Formularios/Contacto.cshtml`  
Prueba la aplicación y asegúrate que que  
muestre el formulario.  
Recuerda que el punto de entrada es  
`/Formas/Contacto`



```
<form id="miformulario" method="post" action="/Forma/GuardarContacto">
  <fieldset>
    <label for="apellido">Apellido</label>
    <input type="text" id="apellido" name="apellido" />
  </fieldset>
</fieldset>
```

Edita formulario .cshtml

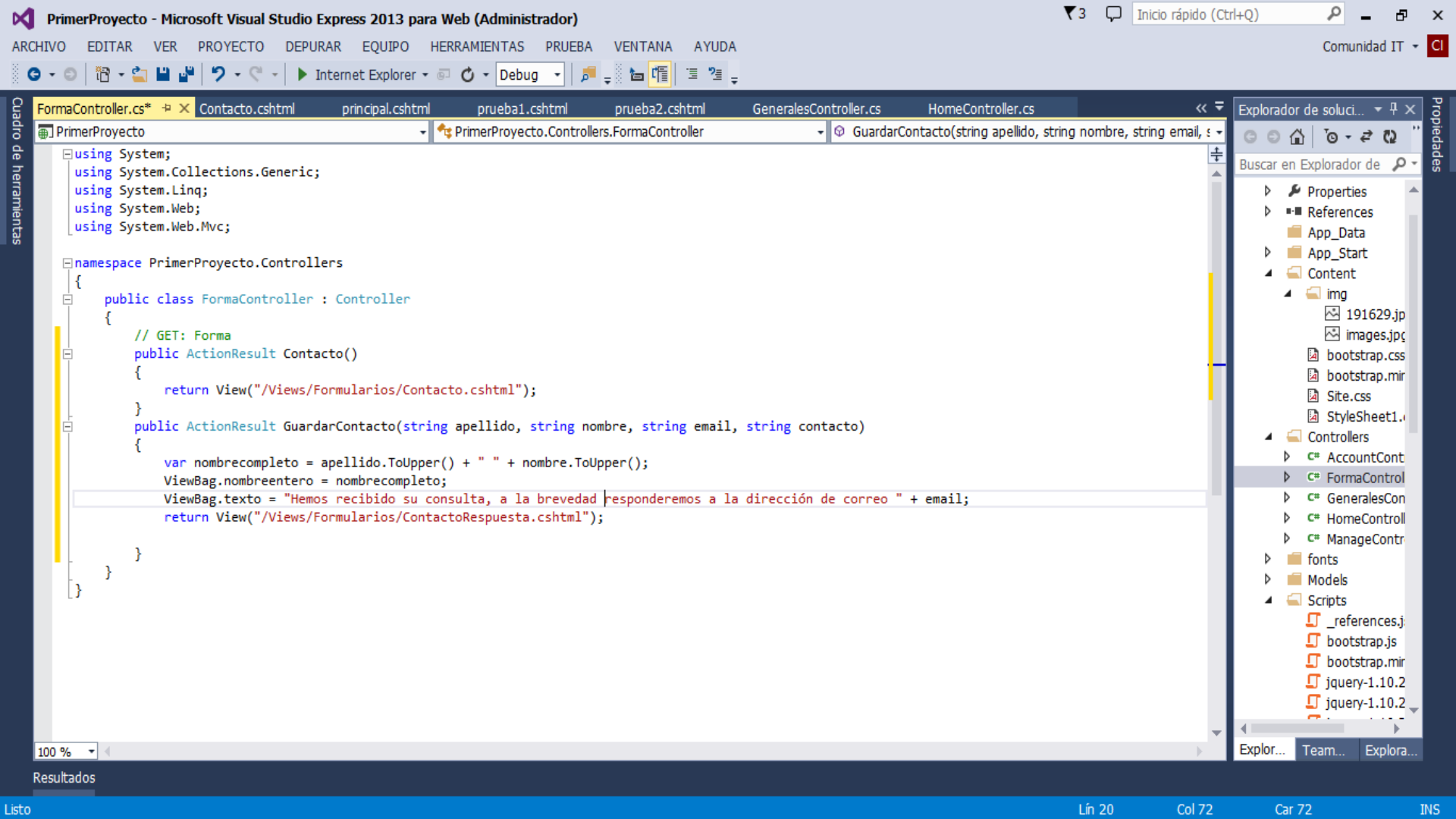
Edita el formulario  
llamado  
contacto.cshtml y  
escribe en la  
sección action lo  
siguiente

continua con

Vuelve al controlador denominado Formas y crea otro punto de entrada, esta vez, para recibir los datos del formulario.

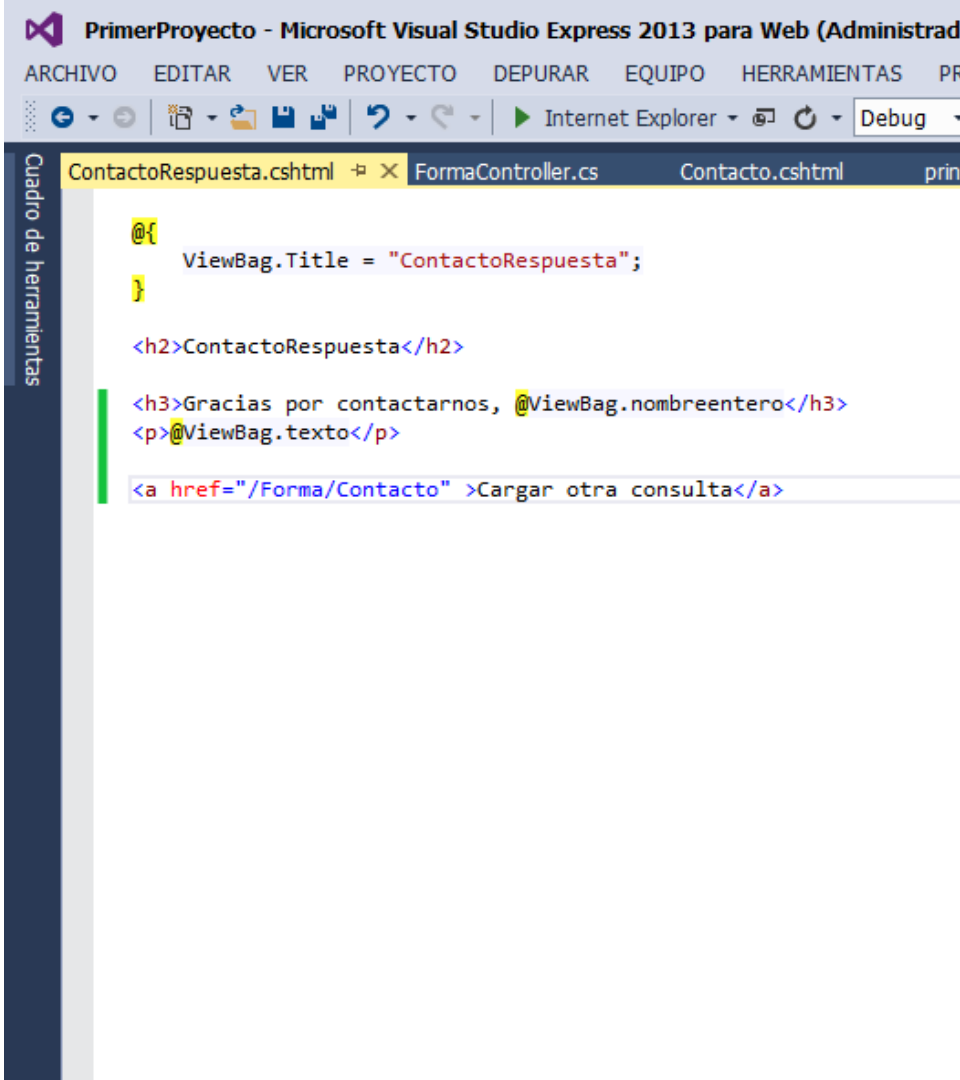
Denomínalo GuardarContacto.

El código del controlador debería quedar algo parecido a la siguiente imagen



por ultimo, y como dice el controlador

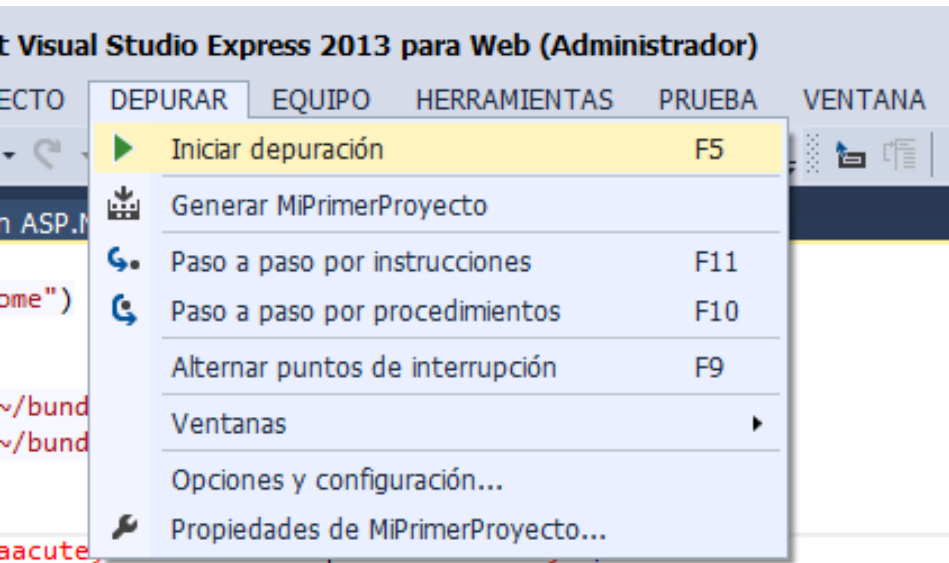
Crea una vista, en la carpeta Formularios  
Que se llame ContactoRespuesta  
La vista espera datos desde el  
controlador, que los envía a través de  
la ViewBag



Crea y edita el formulario  
ContactoRespuesta.cshtml

Debería quedar  
algo parecido a  
esto

ejecuta la aplicación



Prueba el circuito.  
Inicia desde  
/Forma/Contacto  
Y verifica como  
funciona cuando  
haces click en  
Enviar

si lo probaste con éxito, notarás una pequeña falla

El correo electrónico (email), estaba compuesto en un mensaje, sin embargo, no aparece.

Esto se debe en que hay una diferencia entre el nombre del parámetro en el controlador y lo que se envía del formulario.

Intenta corregirlo

continua ejercitando

Agrega validadores con JQuery Validator al formulario, para evitar que envíes datos vacíos.

Agrega una pagina de inicio a la aplicación (la que quieras) con una serie de botones que te envíen al formulario de contacto sin necesidad de tipear la ruta en la barra de direcciones



## Otro ejercicio

En el mismo proyecto, crea una vista en la carpeta formularios que reciba dos numeros.

La estructura es similar al formulario anterior, pero solo con dos campos.

Incluye el botón enviar

En el valor action del formulario coloca Formas/Sumar, este metodo calculará la suma

## Otro ejercicio

En el controlador Forma crea un punto de entrada que se llame Formulario, que devuelva la vista recién creada (con el nombre que hayas elegido)

Crea otro punto de entrada llamado Sumar, que reciba como parámetro ambos numeros.

## Otro ejercicio

Suma ambos números (truco, si te falla la suma, intenta que los parámetros en vez de ser de tipo string, que sean de tipo int)

Devuelve el resultado de los numeros en una vista que se llame `resultadodelasuma`

sigue explorando

En la medida que puedas explora con más detalles el lenguaje C# para ver que se puede hacer.

Operaciones más complejas, repeticiones y otras tantas operaciones.

sigue explorando

Utiliza el esquema de vistas y controladores para poder mostrar en pantalla el resultado o para solicitarle datos al usuario.

# El lenguaje C#

Aspectos básicos del lenguaje

Aprendiendo a programar. Capítulo 6. Tutorial

objetivo

Profundizaremos en el lenguaje C#  
Identificaremos los componentes y  
efectuaremos las primeras  
codificaciones

conceptos previos

C# es el lenguaje del servidor.

Este se usa para tomar decisiones complejas que involucren a todos los usuarios del sistema.

Es, por ejemplo, quien toma la decisión de qué página se presenta, y condiciona sus resultados.



```
public class HomeController : Controller
{
    public ActionResult Prueba1()
    {
        return View("/Views/Pruebas/prueba1.cshtml");
    }
    public ActionResult Principal()
    {

```

Ya estuviste trabajando con C#,  
sin darte cuenta

Cada vez que  
creabas un punto  
de entrada en el  
**CONTROLADOR**,  
estabas  
escribiendo  
código en C#

algunas consideraciones

En el servidor el código de nuestro programa se almacena en lo que se llaman Clases, que por lo general, están dentro de un archivo de extensión cs. Las clases son de diferente tipo, según el tipo, las cosas que puede hacer. Hasta ahora hemos conocido las clases de tipo Controller

PrimerProyecto - Microsoft Visual Studio Express 2013 para Web (Administrador)

ARCHIVO EDITAR VER PROYECTO DEPURAR EQUIPO HERRAMIENTAS PRUEBA VENTANA

HomeController.cs principal.cshhtml La aplicación ASP.NET

PrimerProyecto PrimerProyecto.Controllers.HomeController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PrimerProyecto.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

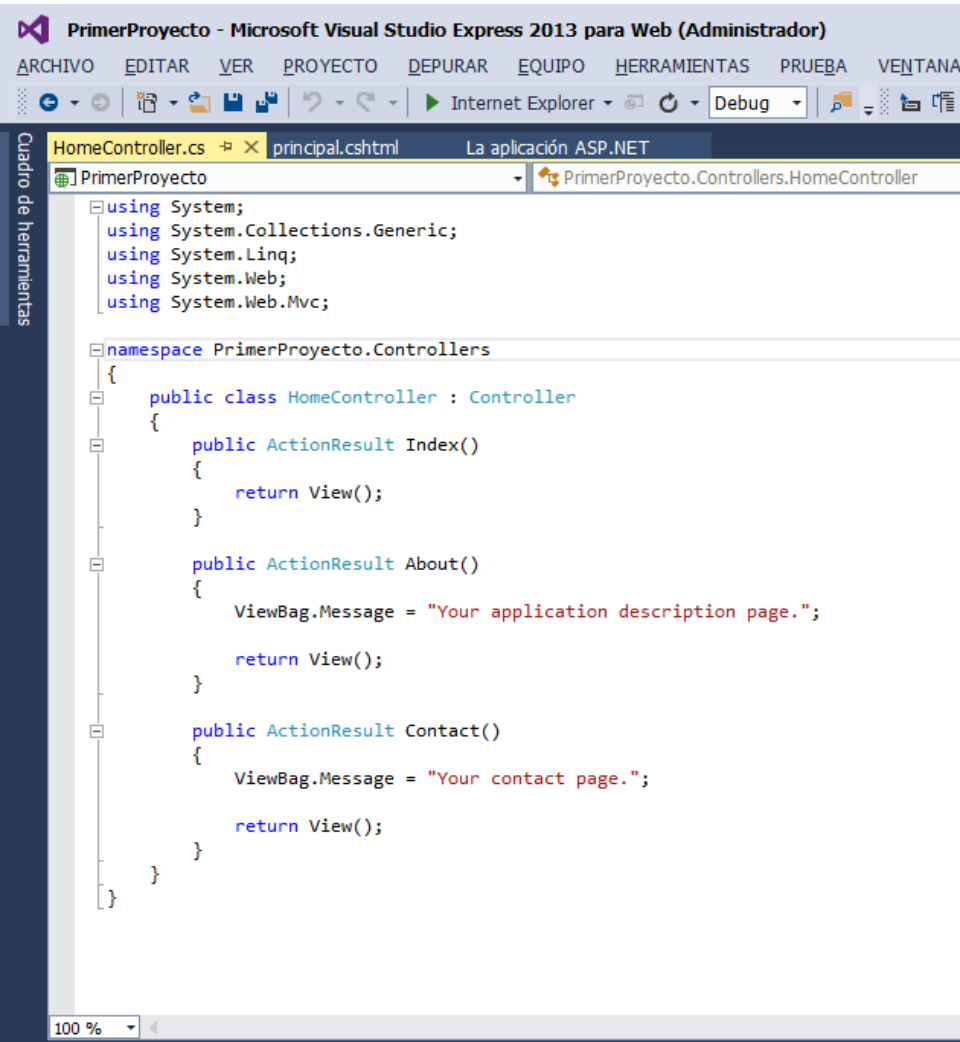
100 %

Listo

Entendiendo las partes

Los bloques de código se encierran entre { }

En este ejemplo tenemos la clase pública (public class) HomeController de tipo Controller



Entendiendo las partes

El controlador posee métodos (lo que para el controlador son los puntos de entrada) Los métodos tienen diferentes tipos, en este caso son de tipo ActionResult

```
public class HomeController : Controller
{
    public string OtroPuntoDeEntrada()
    {
        return "SoloDevuelvoTexto";
    }
    public ActionResult Prueba1()
    {
        return View("/Views/Pruebas/prueba1.cshtml");
    }
    public ActionResult Prueba2()
    {
        return View("/Views/Pruebas/prueba2.cshtml");
    }
    public ActionResult Principal()
    {
        return View("/Views/Custom/principal.cshtml");
    }
}
```

también puedes devolver otras cosas

En este caso hemos creado un punto de entrada que devuelve solo texto.

Para probarlo, ejecuta la aplicación y coloca en el browser /Home/OtroPuntoDeEntrada

comencemos con la ejercitación

Evaluaremos parámetros y  
tomaremos decisiones

```
public class HomeController : Controller
{

    public ActionResult Prueba1(string idioma,string color)
    {
        return View("/Views/Pruebas/prueba1.cshtml");
    }
    public ActionResult Prueba2()
```

Los puntos de entrada también pueden recibir parámetros

Es posible agregar valores al punto de entrada ( o a cualquier función de c#), estos valores se llaman parámetros

utilizando la instrucción IF (SI) para tomar una decisión

En C# podemos usar la instrucción IF para tomar una decisión.

La forma general es

```
if ( ....algo... ) {  
    ... si es que si  
} else {  
    .... si es que no  
}
```



```
using System.Web.Mvc;
```

```
namespace PrimerProyecto.Controllers
```

```
{  
    public class HomeController : Controller
```

```
{
```

```
    public ActionResult Prueba1(string idioma,string color)
```

```
{
```

```
        if (idioma == "ingles")
```

```
{
```

```
            return View("/Views/Pruebas/prueba1.cshtml");
```

```
        }
```

```
    else
```

```
{
```

```
        return View("/Views/Pruebas/prueba2.cshtml");
```

```
    }
```

```
}
```

```
    public ActionResult Prueba2()
```

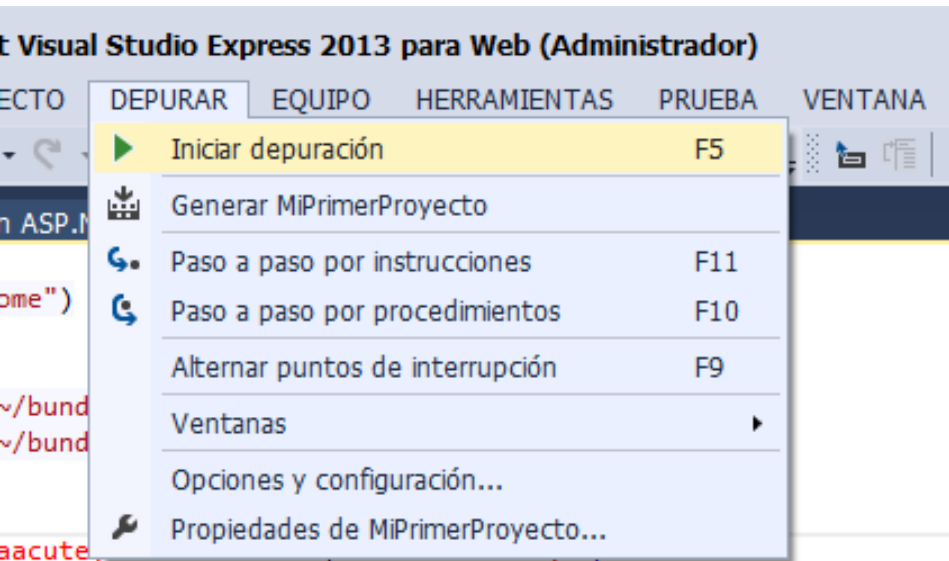
```
{
```

```
        return View("/Views/Pruebas/prueba2.cshtml");
```

así quedaría una evaluación

Recibimos un valor  
y en función de ese  
valor mostramos una  
u otra página

probemos la pagina



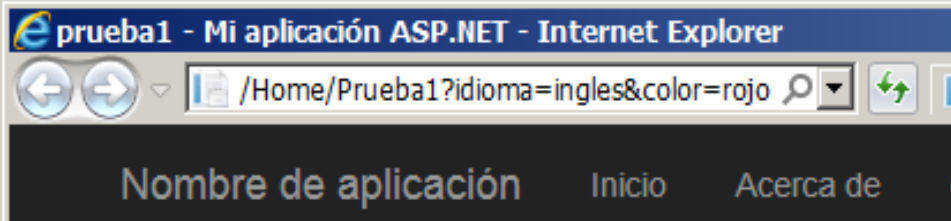
Ve al menú  
depurar y elige  
**INICIAR**  
**DEPURACION** o  
presiona F5

una vez que haya iniciado la página

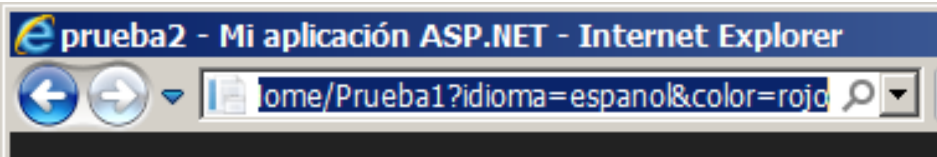
Escribimos la dirección de la página,  
pero esta vez le agregamos los  
parámetros

/Home/Prueba1?idioma=ingles&colo  
r=rojo

presta atención a los detalles



El primer  
parámetro va  
separado por ?  
Los siguientes  
parámetros  
siempre separados  
por &



modifica los parámetros y  
el resultado varía

Intercambia en  
donde dice  
idioma, entre el  
valor ingles y  
español, el  
resultado serán  
distintas páginas

ejercita

Crea una nueva aplicación de cero  
Del tipo Visual C# - Aplicación Web  
MVC

Crea una carpeta dentro de VIEWS  
llamada IDIOMAS y crea 2 vistas  
bienvenida\_espanol  
bienvenida\_ingles

ejercita

Edita cada una de las VISTAS y coloca un texto de bienvenida en el idioma correspondiente.

Crea un punto de entrada en el controlador HomeController.cs

Coloca un parámetro al punto de entrada llamado idioma

ejercita

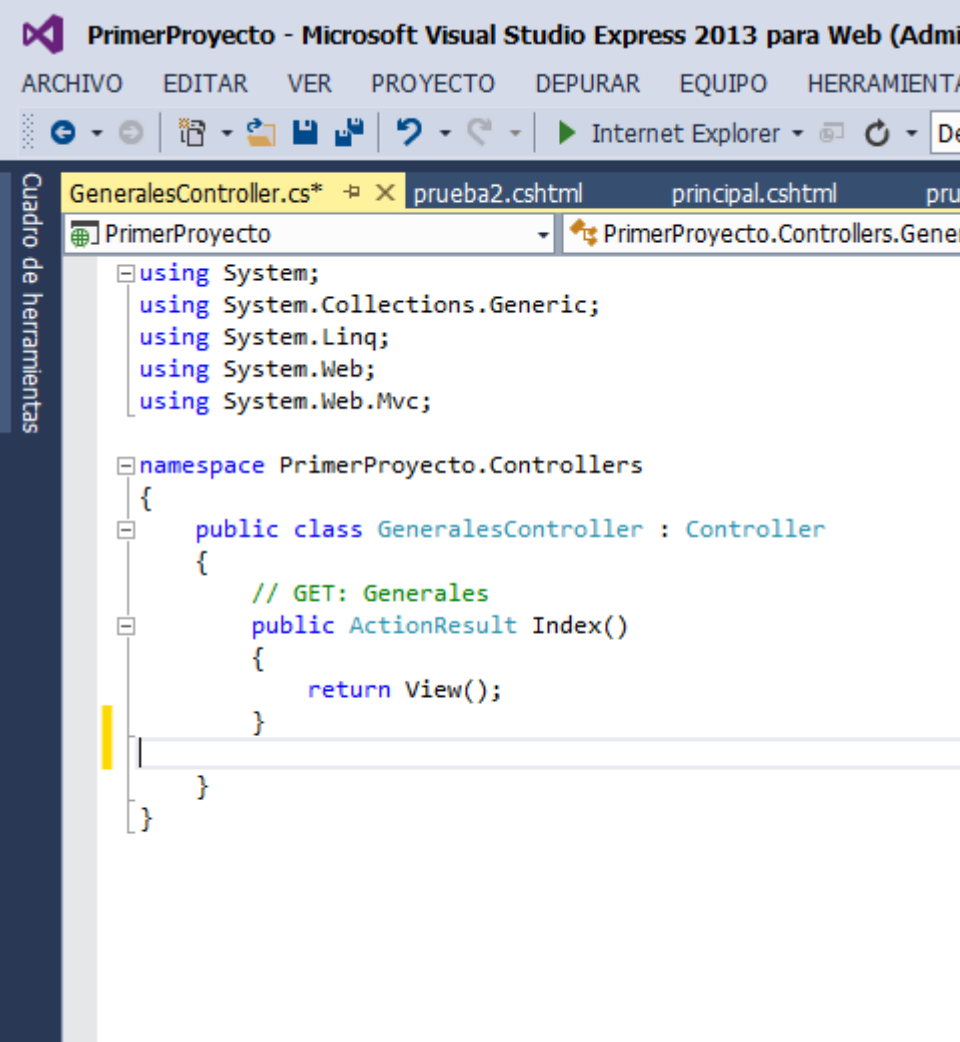
Según el idioma, coloca una  
sentencia de tipo IF para decidir si  
muestras la bienvenida en uno u otro  
idioma



crea tu propio controlador

Hasta ahora, hemos utilizado los controladores existentes en el proyecto.

Tu puedes crear un Controlador Propio para manejar el conjunto de VISTAS que necesites



sobre la carpeta Controllers

Haz click derecho,  
agregar nuevo  
controlador  
Elige controlador en  
blanco  
Coloca el nombre  
Generales (el subfijo  
controller lo agrega  
solo)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PrimerProyecto.Controllers
{
    public class GeneralesController : Controller
    {
        // GET: Generales
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult Principal()
        {
            return View("/Views/Custom/principal.cshtml");
        }
    }
}
```

crea un punto de entrada y pruébalo

# Crea un nuevo punto de entrada llamado Principal

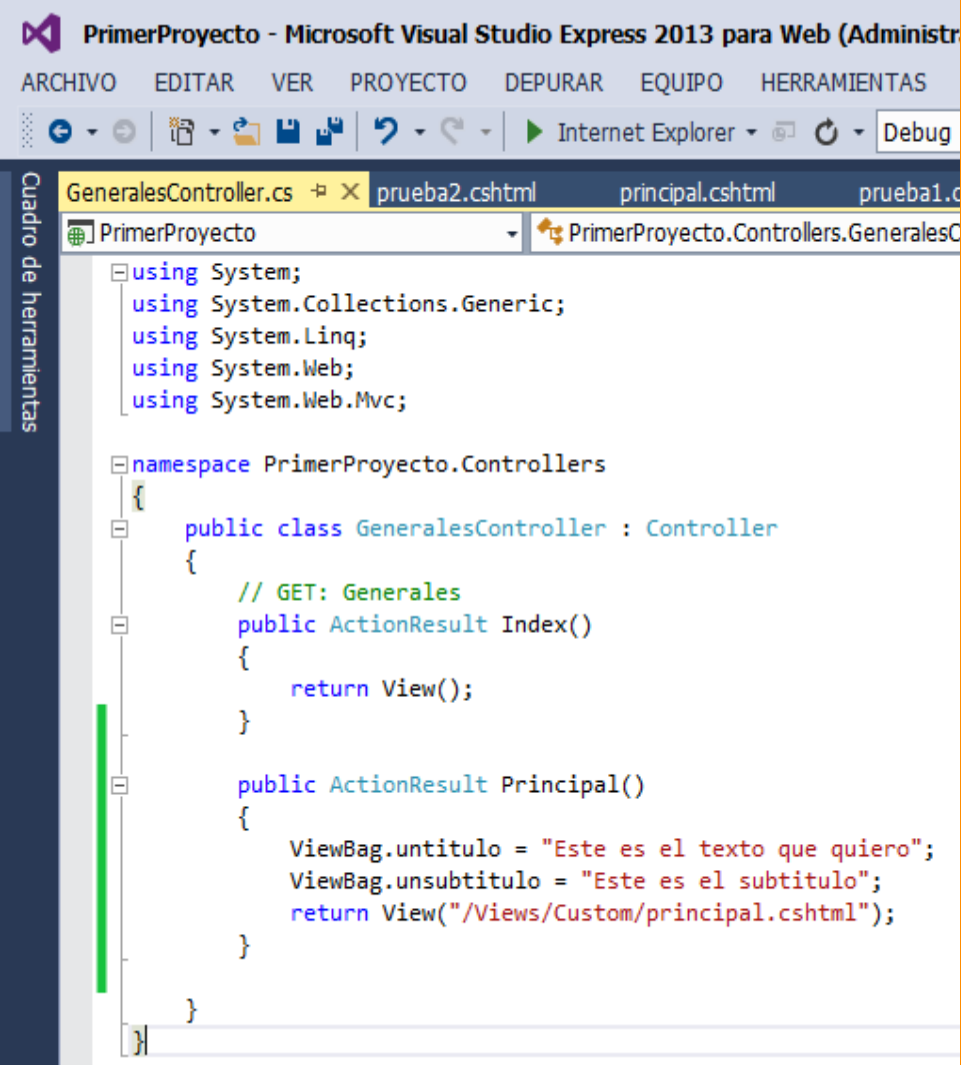
Haz el return view a alguna página de tu proyecto. Prueba en el browser con `/Generales/Principal`

usa tus propios controladores

De ahora en más, si el conjunto de páginas que creas son para algo en particular, crea tu propio controlador. Puedes ingresar a un grupo de páginas desde el mismo controlador usando diferentes puntos de entrada.

enviando parámetros a la vista

Otras de las funciones del controlador, es enviarle datos a la vista para que ciertos valores de las secciones sean dinámicas  
Para esto utiliza la bolsa de datos  
ViewPager



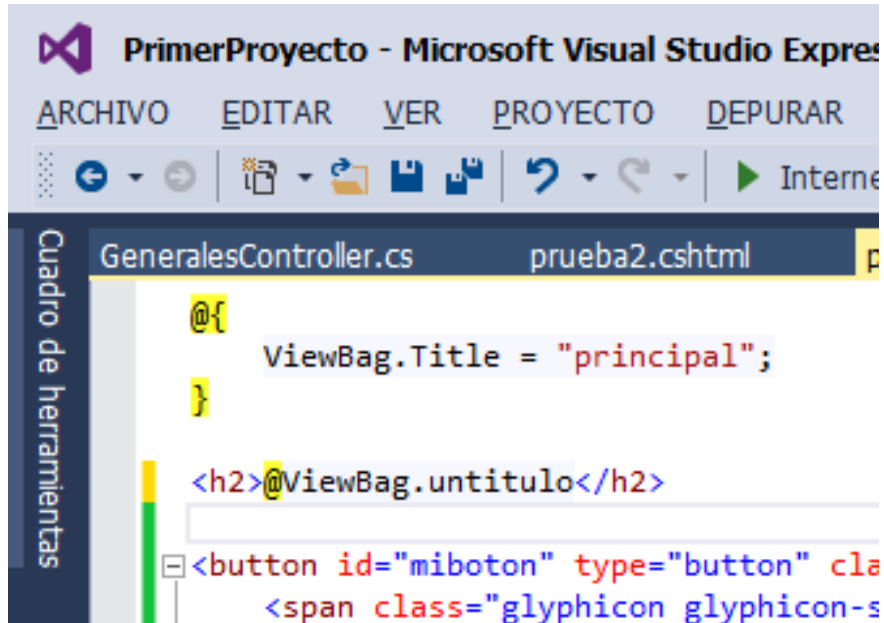
un ejemplo de envío de datos

Envía a la bolsa de  
datos un titulo y un  
subtitulo.

Escríbelo en el  
controlador como  
indica la imagen

edita la vista principal.chnml

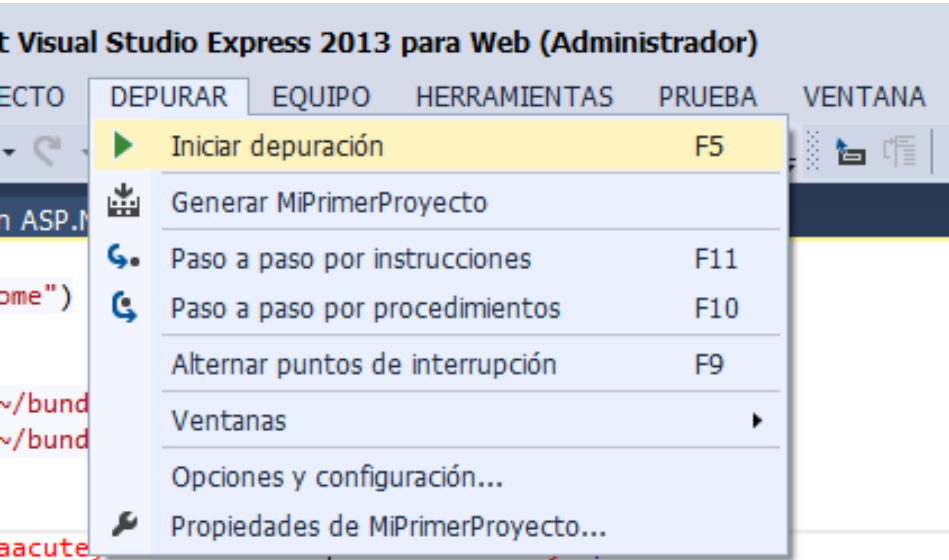
Edita la página y  
modifica el título  
contenido en h2 y  
reemplazalo por la  
forma  
`@ViewBag.<nombre>`



The screenshot shows the Microsoft Visual Studio Express interface. The title bar reads 'PrimerProyecto - Microsoft Visual Studio Express'. The menu bar includes 'ARCHIVO', 'EDITAR', 'VER', 'PROYECTO', and 'DEPURAR'. The toolbar contains icons for navigation and development. The 'Cuadro de herramientas' (Toolbox) is visible on the left. The active file is 'prueba2.cshtml'. The code in the editor is as follows:

```
@{  
    ViewBag.Title = "principal";  
}  
  
<h2>@ViewBag.untitulo</h2>  
  
<button id="miboton" type="button" cla  
    <span class="glyphicon glyphicon-s
```

ejecuta la aplicación



Deberías obtener  
como resultado  
que la página  
muestra en su  
titulo lo que le  
hayas indicado en  
el controlador



para que sirve enviar los datos desde controlador?

Podrías usar una sentencia if para escribir uno u otro texto de bienvenida, o para que una misma vista se llene con diferentes datos, dependiendo de la evaluación que se efectúe.

haz la siguiente prueba y evalúa el resultado

## Escribe en el controlador lo siguiente

```
if (idioma == "ingles") {  
    ViewBag.untitulo = "Welcome";  
} else {  
    ViewBag.untitulo = "Bienvenido";  
}
```

agrega parámetros al punto de entrada

Como hicimos anteriormente, agrega el parámetro idioma como uno de los parámetros del punto de entrada. Ejecuta la aplicación. Como resultado el título debería variar según el idioma

sigue explorando

Usa el buscador web con las claves  
« c# lenguaje sentencias básicas »  
Investiga más características del  
lenguaje C#, si deseas probarlas,  
puedes escribirlas en el controlador.

# El Servidor Web

## Conceptos Avanzados

Uso del Framework Microsoft .NET

Aprendiendo a Programar. Capítulo 7. Tutorial

objetivo

Utilizaremos funciones del framework de .NET para enviar correos  
Ejercitamos sobre diferentes características del mismo.

conceptos previos para enviar un correo

Antes que nada debemos aclarar una obviedad.

Podemos enviar un correo desde nuestra aplicación, siempre y cuando tengamos acceso a un Servidor De Correo.

Por lo general tenemos acceso a uno, el que usa nuestra cuenta de correo

conceptos previos para enviar un correo

Para identificar el servidor, debemos saber el nombre del SMTP

Coloca en el buscador web las siguientes claves

« SMTP Live Hotmail Configuration »

Si tienes otro correo que no sea Live Hotmail reemplaza el nombre que corresponda



Lo que estamos buscando

La configuración que estamos  
buscando es

SMTP address server

SMTP username

SMTP port

SMTP TLS/SSL:

Para live hotmail la configuracion es

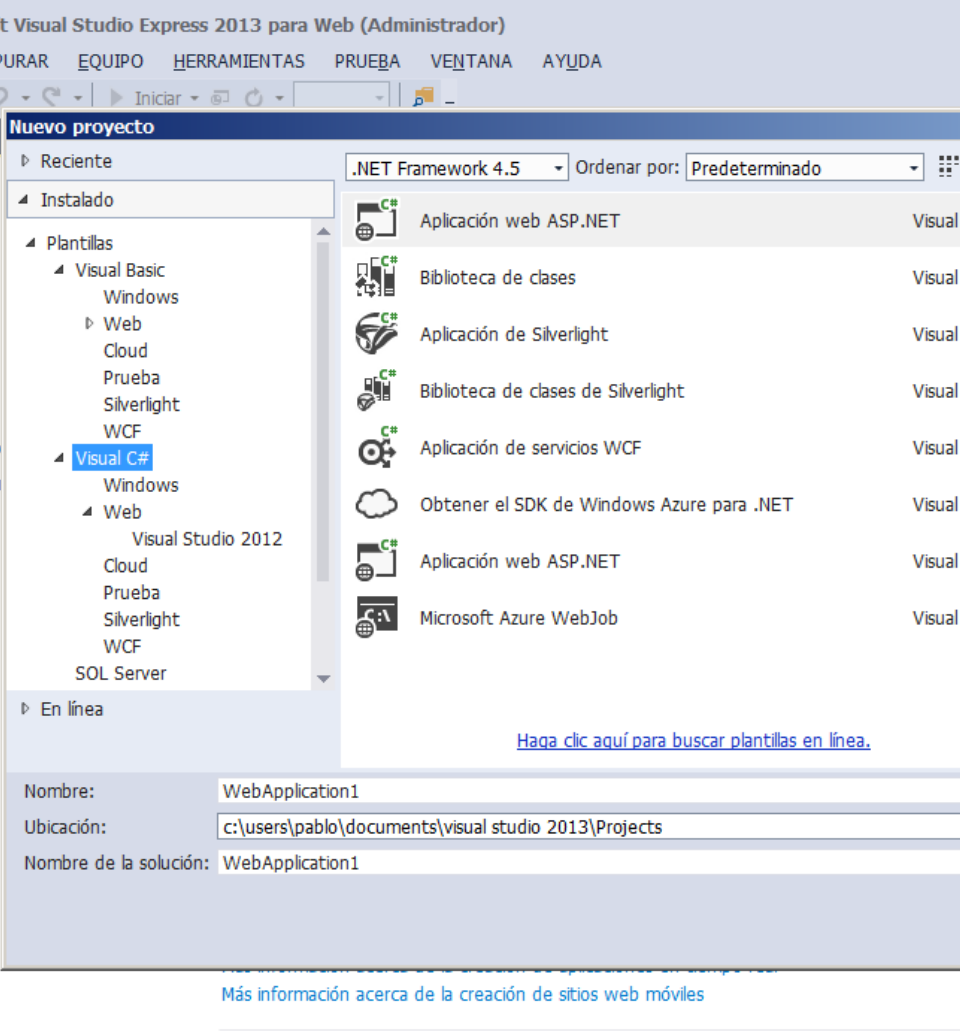
Si tienes una cuenta live, estos serían los valores

SMTP address server: smtp.live.com

SMTP username: tunombre@live.com

SMTP port: 587

SMTP TLS/SSL: si



comencemos

Crea un nuevo  
proyecto de tipo  
Visual C# -  
Aplicación web  
ASP.NET y utiliza la  
plantilla simple  
MVC

crea las siguientes secciones

Crea una carpeta dentro de views  
llamada Formularios

Crea una vista dentro de Formularios  
llamado Contacto

Edita Contacto.cshtml y crea elementos  
html como el formulario visto en el  
capítulo III, con nombre, apellido y  
dirección de email

continua con

Crea un controlador llamado Formas  
Dentro de el, crea un punto de entrada  
llamado Contacto y retorna la vista  
`/Views/Formularios/Contacto.cshtml`  
Prueba la aplicación y asegúrate que que  
muestre el formulario.  
Recuerda que el punto de entrada es  
`/Formas/Contacto`

```
<form id="miformulario" method="post" action="/Forma/GuardarContacto">
  <fieldset>
    <label for="apellido">Apellido</label>
    <input type="text" id="apellido" name="apellido" />
  </fieldset>
</fieldset>
```

Edita formulario .cshtml

Edita el formulario  
llamado  
contacto.cshtml y  
escribe en la  
sección action lo  
siguiente

continua con

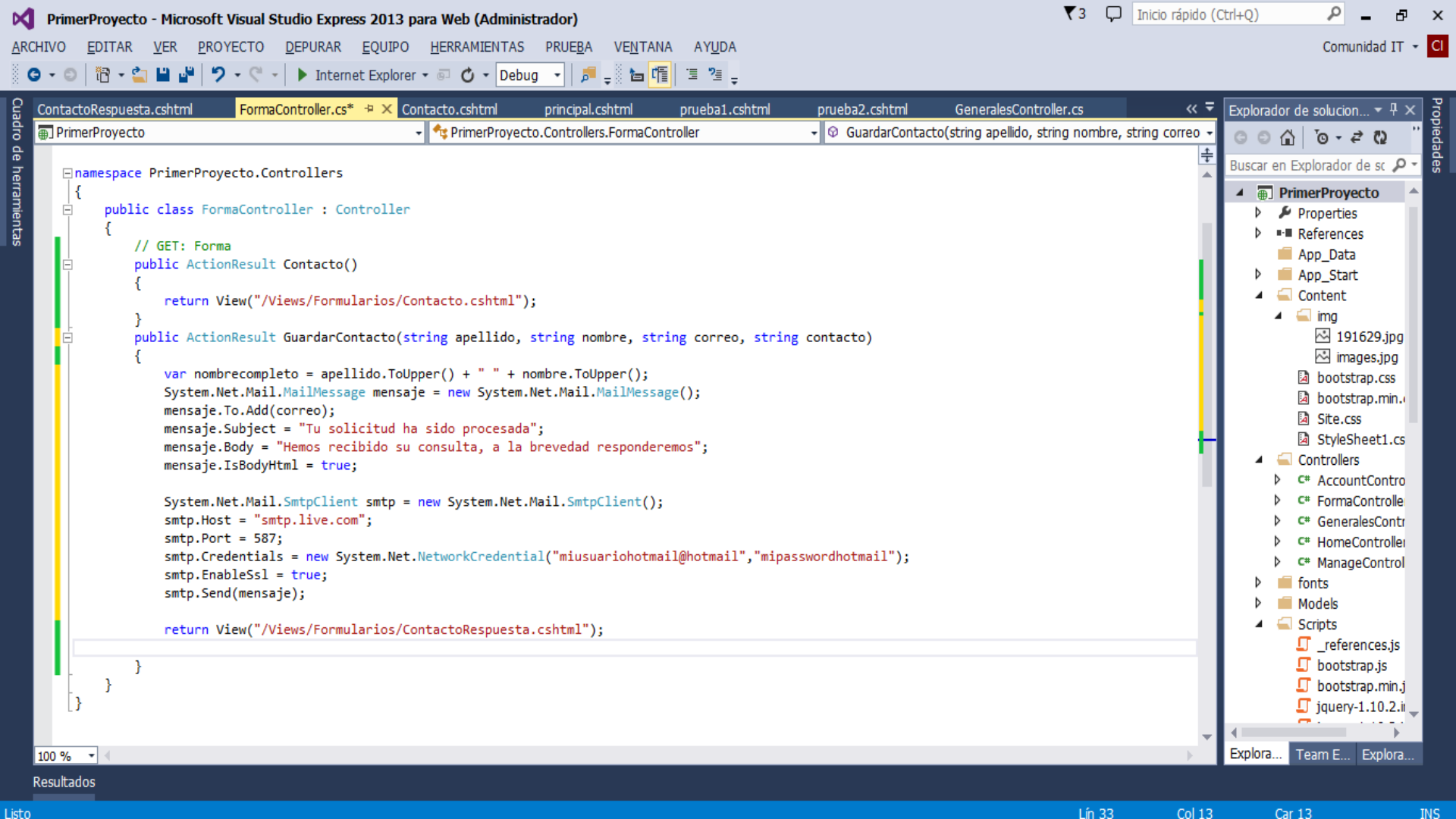
Vuelve al controlador denominado Formas y crea otro punto de entrada, esta vez, para recibir los datos del formulario.  
Denomínalo GuardarContacto.  
Hasta aquí el ejercicio es el mismo que en el capítulo anterior

Vamos a enviar un correo

Para esto, escribimos la función de envío de correo en el método GuardarContacto

Utilizamos la función de envío de mail provista en el framework de .NET, solo tenemos que tener presente los parámetros que buscamos con anterioridad

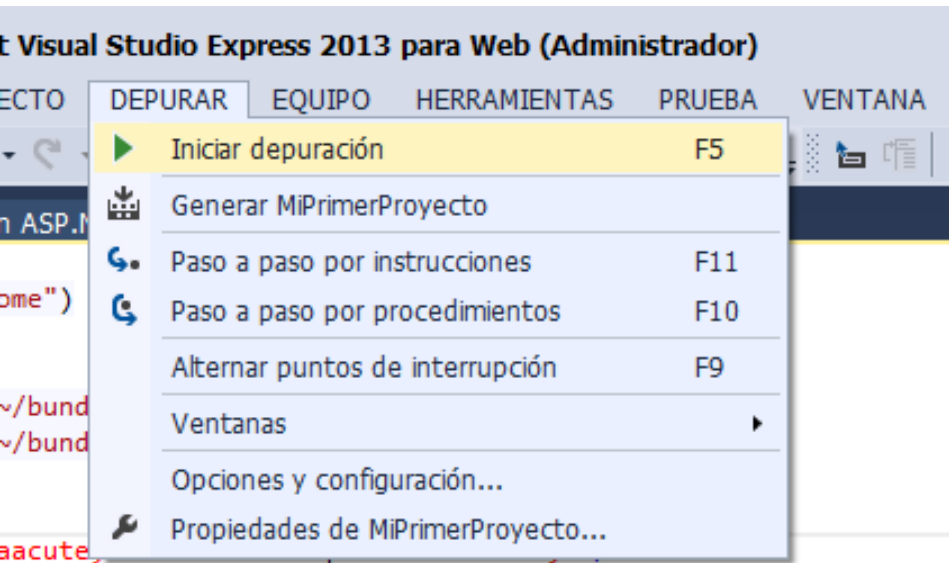




por ultimo, y como dice el controlador

Crea una vista, en la carpeta Formularios  
Que se llame ContactoRespuesta  
Informa en esta vista que haz enviado un  
correo.  
Podrías en el método mensaje.To.Add,  
agregar a más de un destinatario al  
mismo tiempo del mismo correo.

ejecuta la aplicación



Prueba el circuito.  
Inicia desde  
/Forma/Contacto  
Y verifica como  
funciona cuando  
haces click en  
Enviar

continua ejercitando

Agrega validadores con JQuery Validator al formulario, para evitar que envíes datos vacíos.

El más importante, la dirección de mail no debería estar vacía

## Otro ejercicio

En el mismo proyecto, crea una vista en la carpeta formularios que reciba dos números.

La estructura es similar al formulario anterior, pero solo con dos campos.

Incluye el botón enviar

En el valor action del formulario coloca Formas/Raices, este método calculará la raíz cuadrada de cada número

## Otro ejercicio

Para esto utilizaremos otra función del Framework .NET, la clase MATH, que posee un sinfín de operaciones matemáticas

En el caso de la raíz cuadrada el método es `Math.Sqrt( tu numero )`

Devuelve el resultado en una vista

sigue explorando

En la medida que puedas explora con más detalles el framework .NET  
La clase System.Math te permite efectuar calculos matemáticos avanzados de geometría (muy útil para 3D).

sigue explorando

La clase `System.Text` te permite manipular textos

La clase `System.Web.Sockets` te permite comunicaciones en dos vías, ideal para transferir datos entre dispositivos vía web, ideal para juegos en tiempo real.



sigue explorando

Si quieres usar los puertos serie o usb lo puedes hacer desde la clase `System.IO` (aunque no es un tipo de aplicación Web), con lo cual podrías controlar maquinaria y motores (robótica)

sigue explorando

Al punto que te encuentras, puedes utilizar el controlador para codificar lo que precises e ir descubriendo características del lenguaje.

Anímate a probar y no te preocupes por equivocarte, no hay riesgo que se rompa nada.

# Dónde y cómo se guardan los datos

Creación de tablas y consultas

Aprendiendo a Programar. Capítulo 8. Tutorial

objetivo

Exploraremos las herramientas  
de creación de tablas y edición de  
tablas y ejecutaremos consultas  
básicas

conceptos previos

En el capítulo teórico observamos que los datos se estructuran en tablas y campos.

Asimismo, una vez creadas y con datos, se pueden ejecutar consultas según diferentes criterios

conceptos previos

Para esto, es necesario tener instalado un SERVIDOR DE BASE DE DATOS.

Al momento de instalar Visual Studio, se nos provee en la instalación de un Servidor de Base de Datos (RDBM) llamado SQL Server Express

conceptos previos

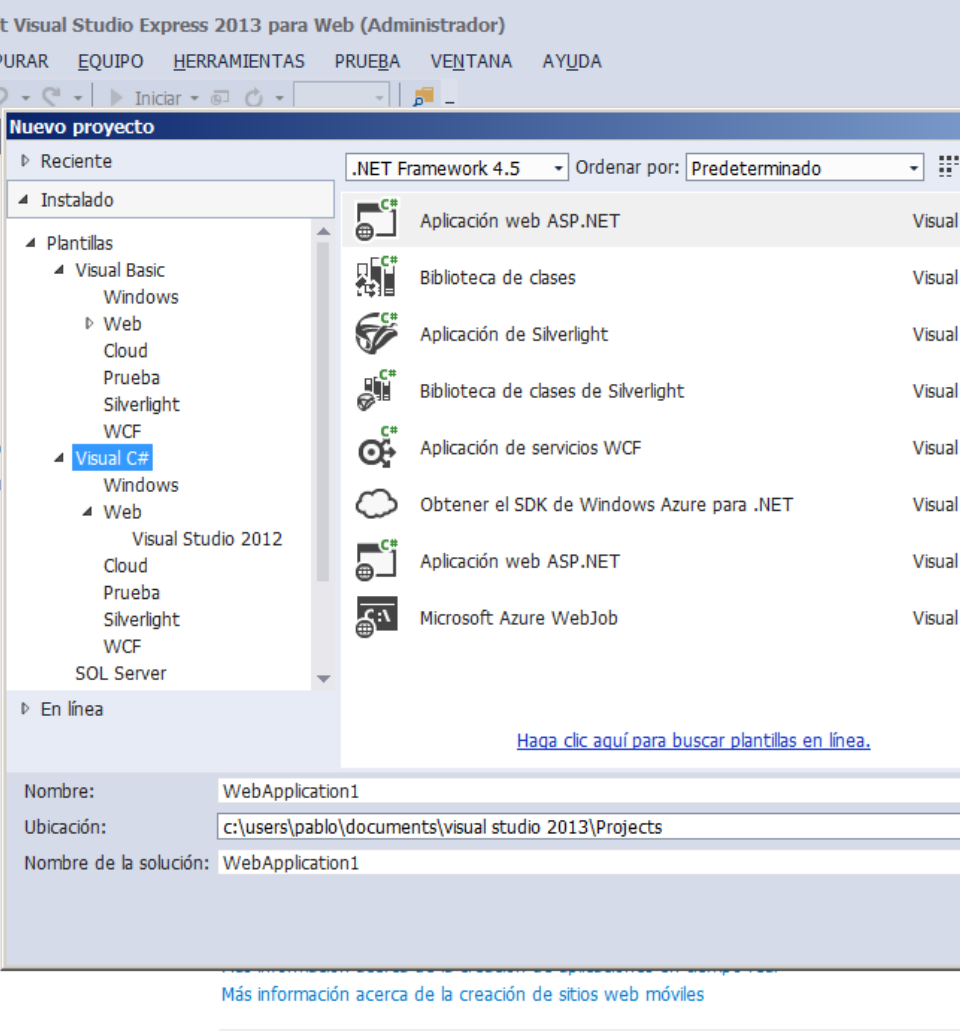
El Servidor de Base de Datos es un programa que se ejecuta en una máquina (por lo general encendida todo el tiempo) al cual todas las aplicaciones de una red (clientes) pueden enviarle consultas, insertar y modificar datos.

conceptos previos

En un entorno de trabajo real y en producción este Servidor por lo general se encuentra en una computadora separada

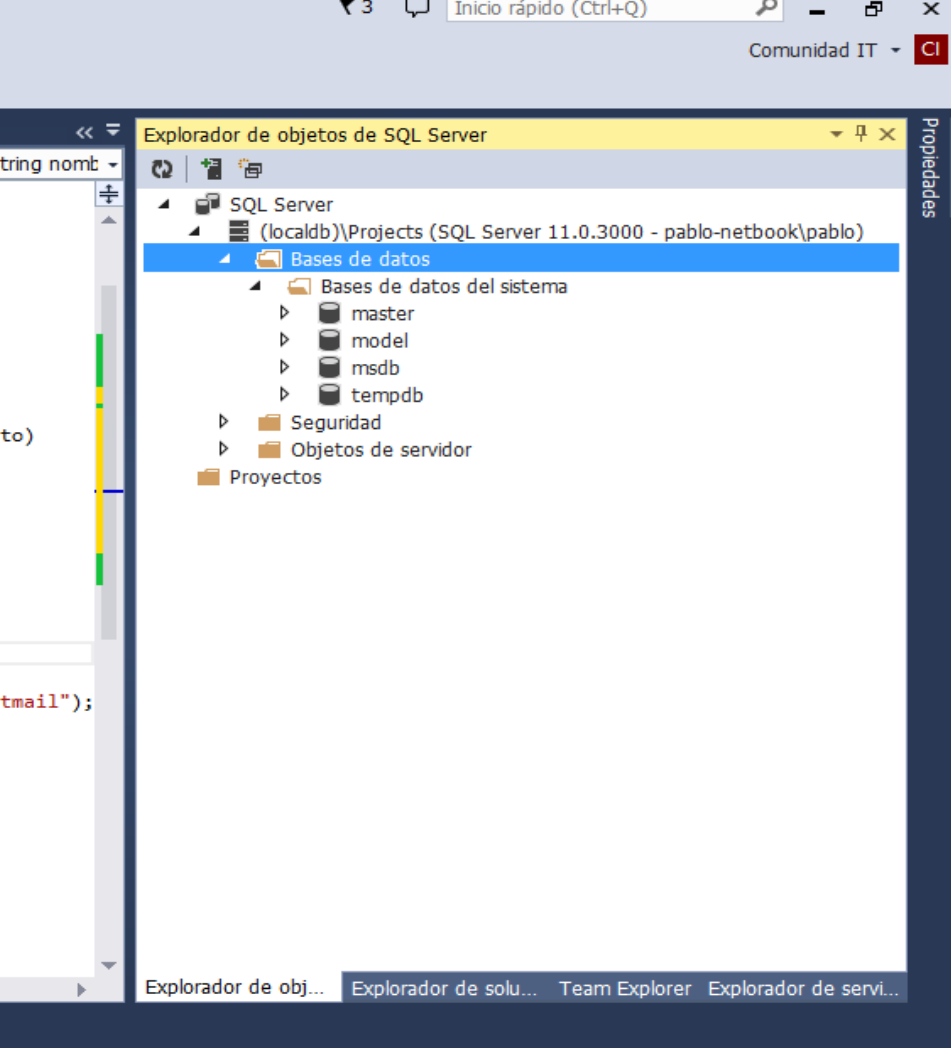
En nuestro ambiente de desarrollo, se encuentra instalada en nuestra computadora.





comencemos

Abre un proyecto  
existente o crea  
uno nuevo



ubica el explorador de objetos

Selecciona el  
menú ver,  
explorador de  
objetos de SQL  
Server.  
Expande el árbol

conceptos previos

Si por alguna razón no aparece el árbol indicado, es posible que no tengas correctamente instalado SQL Server Express

Intenta buscar las siguientes claves en el buscador de internet

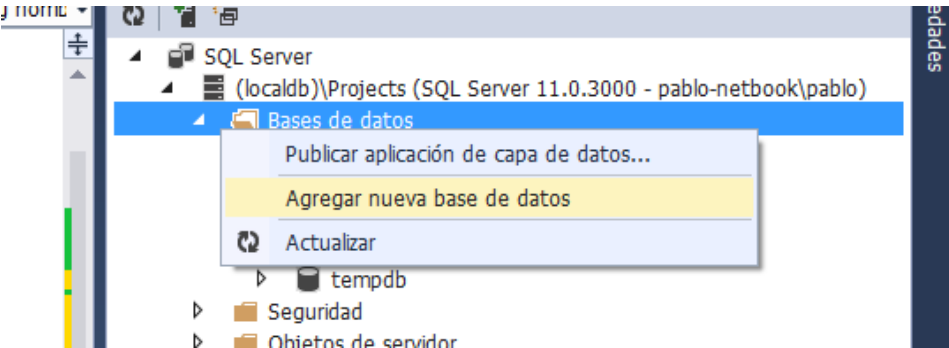
« Install SQL Server Express Edition Visual Studio»

crea una nueva base de datos

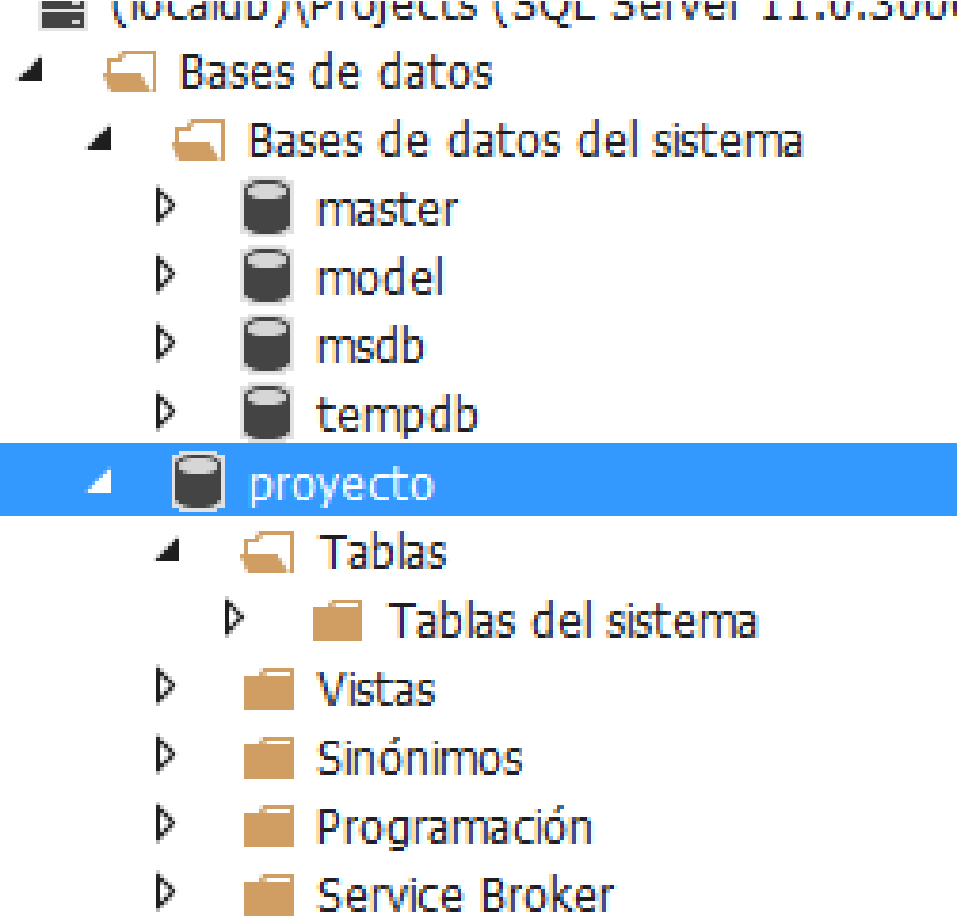
Existen algunas bases de datos previamente creadas de uso de interno del sistema.

Crearemos una nueva base de datos para este proyecto, la misma podrá ser utilizada en otras aplicaciones web que desarrollemos más adelante.

sobre la carpeta Base de Datos



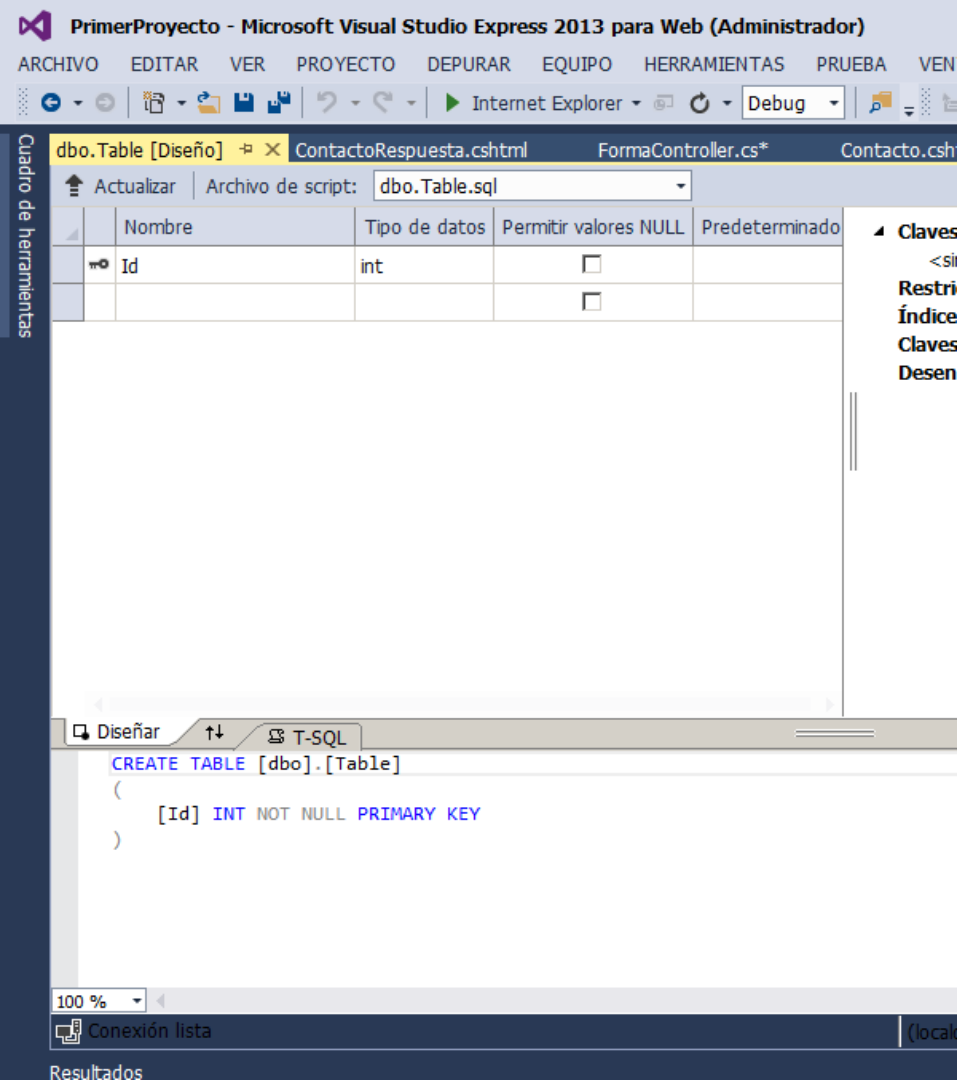
Click derecho,  
agrega una nueva  
base de datos.  
Colócale un  
nombre, por  
ejemplo  
«proyecto»



hemos creado la base de datos  
proyecto

Expande las carpetas  
de la base de datos  
proyecto  
recientemente  
creada.

Observa que aún no  
hay Tablas creadas



agrega una tabla

Haz click derecho sobre la carpeta Tablas, crear nueva Tabla.  
Aparecerá el siguiente formulario para que definas los campos

la base de datos y la nueva tabla (aún sin nombre)  
está lista para definirle los campos

Utilizaremos el formulario para definir  
cada uno de los campos.

Por cada campo se debe indicar el  
nombre, el tipo de dato, y si permite  
valores nulos (en blanco o no)



define la estructura de campos para la tabla Contactos

Para la tabla Contactos crearemos los siguientes campos

Id, tipo int, no permitir nulos

Apellido, tipo varchar, permitir nulos

Nombre, tipo varchar, permitir nulos

Correo, tipo varchar, permitir nulos

PrimerProyecto - Microsoft Visual Studio Express 2013 para Web (Administrador de servidores)

ARCHIVO EDITAR VER PROYECTO DEPURAR EQUIPO HERRAMIENTAS PRUEBAS

dbo.Table [Diseño]\* ContactoRespuesta.cshtml FormController.cs\*

Actualizar Archivo de script: dbo.Table.sql\*

	Nombre	Tipo de datos	Permitir valores NULL	Predeterminado
PK	Id	int	<input type="checkbox"/>	
	Apellido	varchar(100)	<input checked="" type="checkbox"/>	
	Nombre	varchar(100)	<input checked="" type="checkbox"/>	
	Correo	varchar(200)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

Diseñar T-SQL

```
CREATE TABLE [dbo].[Table]
(
    [Id] INT NOT NULL PRIMARY KEY,
    [Apellido] VARCHAR(100) NULL,
    [Nombre] VARCHAR(100) NULL,
    [Correo] VARCHAR(200) NULL
)
```

100 %

Conexión lista

debería quedar algo parecido a esto

Observa que los tipos de datos varchar poseen un número entre paréntesis. Es la cantidad máxima de caracteres para ese dato

define la clave primaria

Recuerda que todas las tablas deberían poseer una clave única de identificación Para hacer más eficientes las consultas. La Primary Key, por lo general es un tipo de dato int, llamado Id.

PrimerProyecto - Microsoft Visual Studio Express 2013 para Web (Administrador)

ARCHIVO EDITAR VER PROYECTO DEPURAR EQUIPO HERRAMIENTAS PRUE

dbo.Table [Diseño]\* ContactoRespuesta.cshtml FormController.cs\*

Actualizar Archivo de script: dbo.Table.sql\*

	Nombre	Tipo de datos	Permitir valores NULL	Predeter
	Id	int	<input type="checkbox"/>	
	Apellido	varchar(100)	<input checked="" type="checkbox"/>	
	Nombre	varchar(100)	<input checked="" type="checkbox"/>	
	Correo	varchar(200)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

Diseñar T-SQL

```
CREATE TABLE [dbo].[Table]
(
    [Id] INT NOT NULL PRIMARY KEY,
    [Apellido] VARCHAR(100) NULL,
    [Nombre] VARCHAR(100) NULL,
    [Correo] VARCHAR(200) NULL
)
```

100 %

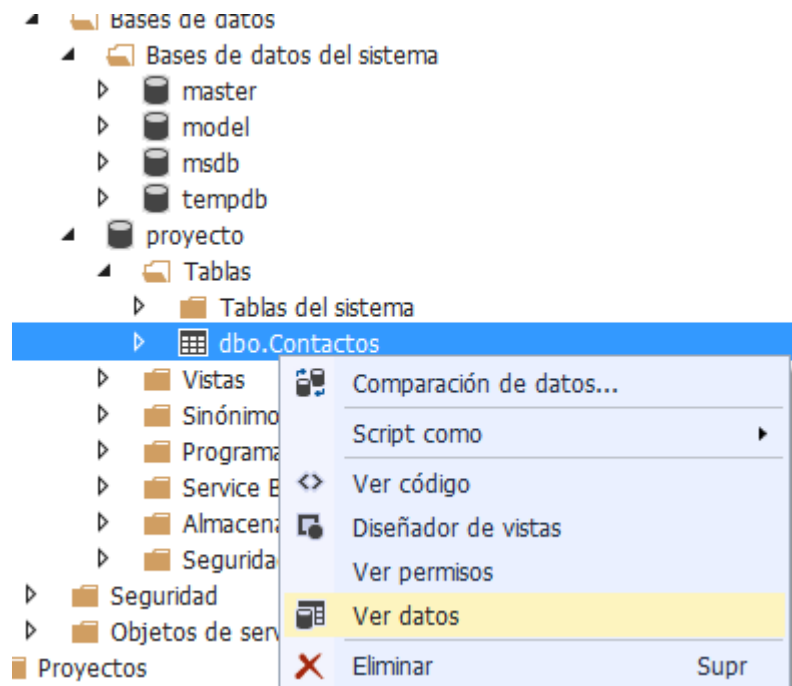
Conexión lista

define la clave primaria

Haz click derecho sobre el campo Id, y coloca la opción Establecer clave principal. Aparecerá una llave para indicarlo.

ya está casi lista

La tabla está casi lista para ser utilizada.  
Para esto, haz click en el botón guardar,  
te preguntará el nombre de la tabla  
Guárdala con el nombre «Contactos»  
Una vez creada y guardada, insertaremos  
un conjunto de datos de prueba



ingresando datos de prueba,  
opción ver datos

En el explorador de  
objetos de SQL, ubica la  
base proyectos, carpeta  
tablas, aparece nuestra  
tabla recién creada.  
Si no llegara a aparecer,  
intenta con la opción  
Actualizar

	Id	Apellido	Nombre	Correo
	1	Gottifredi	Pablo	pablo.gottifre...
	2	Listingart	Pablo	pablo@comuni...
	3	Acuña	Soledad	sacuña@comu...
►*	<u>NULL</u>	NULL	NULL	NULL

ingresando datos de prueba

En el formulario Ver Datos, ingresa datos de prueba

nuestra tabla está lista para efectuarle consultas

Hemos creado una base de datos y una nueva tabla. Asimismo hemos ingresado datos de prueba.

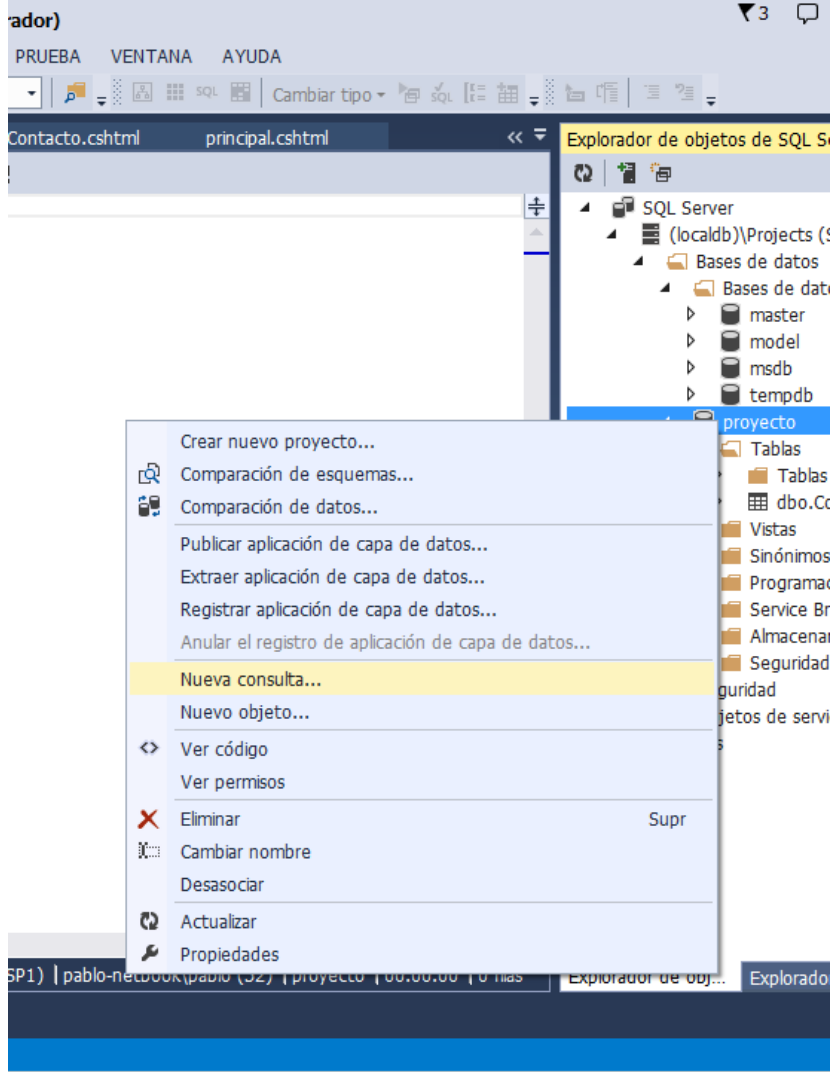
Estos datos quedan almacenados en forma permanente en nuestra Base de Datos.



sobre los datos creados efectuaremos consultas

Para efectuar consultas, es necesario abrir el editor de consultas.

Aquí insertaremos sentencias de lenguaje de consulta estructurado (Structured Query Language, o también conocido como SQL)



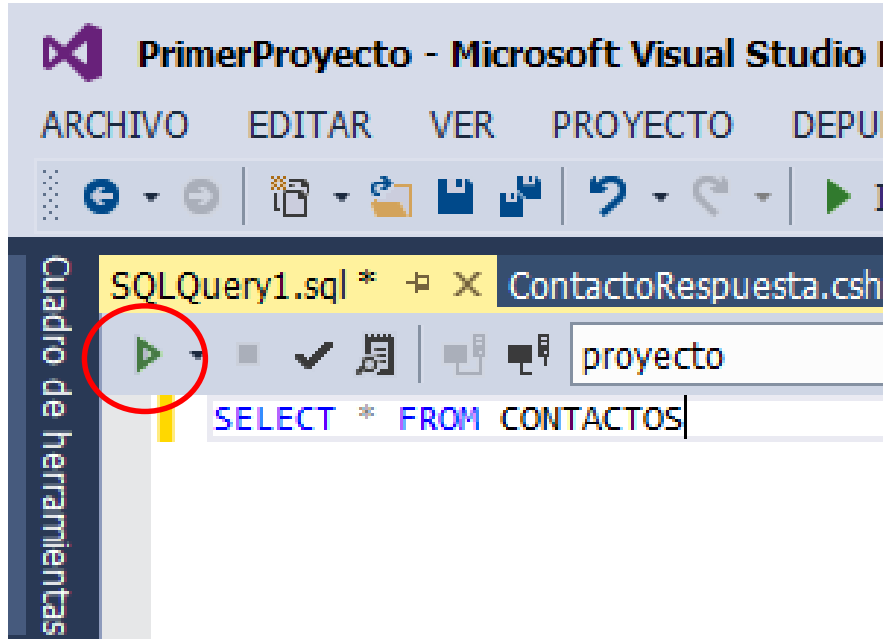
en el explorador de objetos de sql

Hacemos click derecho  
en nuestra base de  
datos proyecto y  
seleccionamos nueva  
consulta

escribimos nuestra primer consulta

```
SELECT * FROM  
CONTACTOS
```

Luego presionamos la  
tecla ejecutar



la primer consulta arroja un resultado

Al ejecutar la consulta `SELECT * ...`,  
obtenemos como resultado todos los  
registros de nuestra tabla.  
En esta interfaz podremos probar  
diferentes criterios de consulta sobre los  
datos, obteniendo diferentes resultados

ejecuta las siguientes consultas por separado y observa los resultados

a) `select * from Contactos`

b) `select * from Contactos order by Apellido`

c) `select * from Contactos where nombre = "Pablo"`

d) `insert into Contactos ("A","B","c@b")`

e) `select * from Contactos`

explora más sobre el lenguaje SQL

Explora algunas variantes de consultas  
sobre el lenguaje SQL desde esta interfaz

Ingresa en el buscador WEB  
« Sentencias básicas SQL »

continua ejercitando

Haz creado la tabla Contactos en la base de datos proyecto.

Puedes repetir el procedimiento desde el paso 12-13 para crear nuevas tablas en la misma base de datos.

continua ejercitando

Crea la tabla Localidades, con los siguientes campos

Id, int, no permitir nulos, clave primaria  
Descripcion, varchar(50), no nulos  
Estado, varchar(50) no nulos



continua ejercitando

Guarda la tabla Localidades, ingresa datos de prueba con ciudades/localidades de distintos estados/provincias

Intenta efectuar una consulta SQL que seleccione todas las localidades de un mismo estado

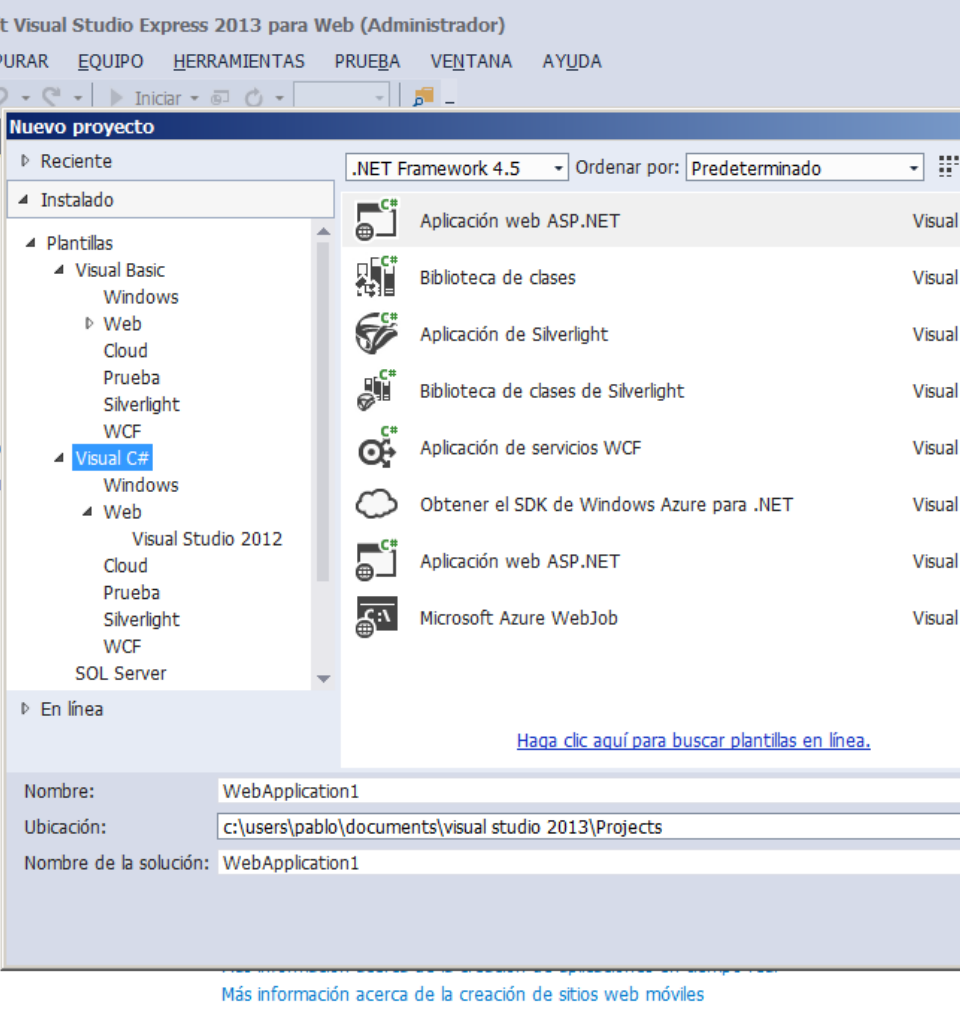
# Conectando los datos con la aplicación

Ejecutar consultas, recuperar y guardar información

Aprendiendo a programar. Capítulo 9. Tutorial

objetivo

Guardar y/o recuperar información almacenadas en una base de datos desde nuestra aplicación.



comencemos

Abra un proyecto existente, sino, cree uno nuevo del tipo Visual C# - Aplicación web ASP.NET y utiliza la plantilla simple MVC

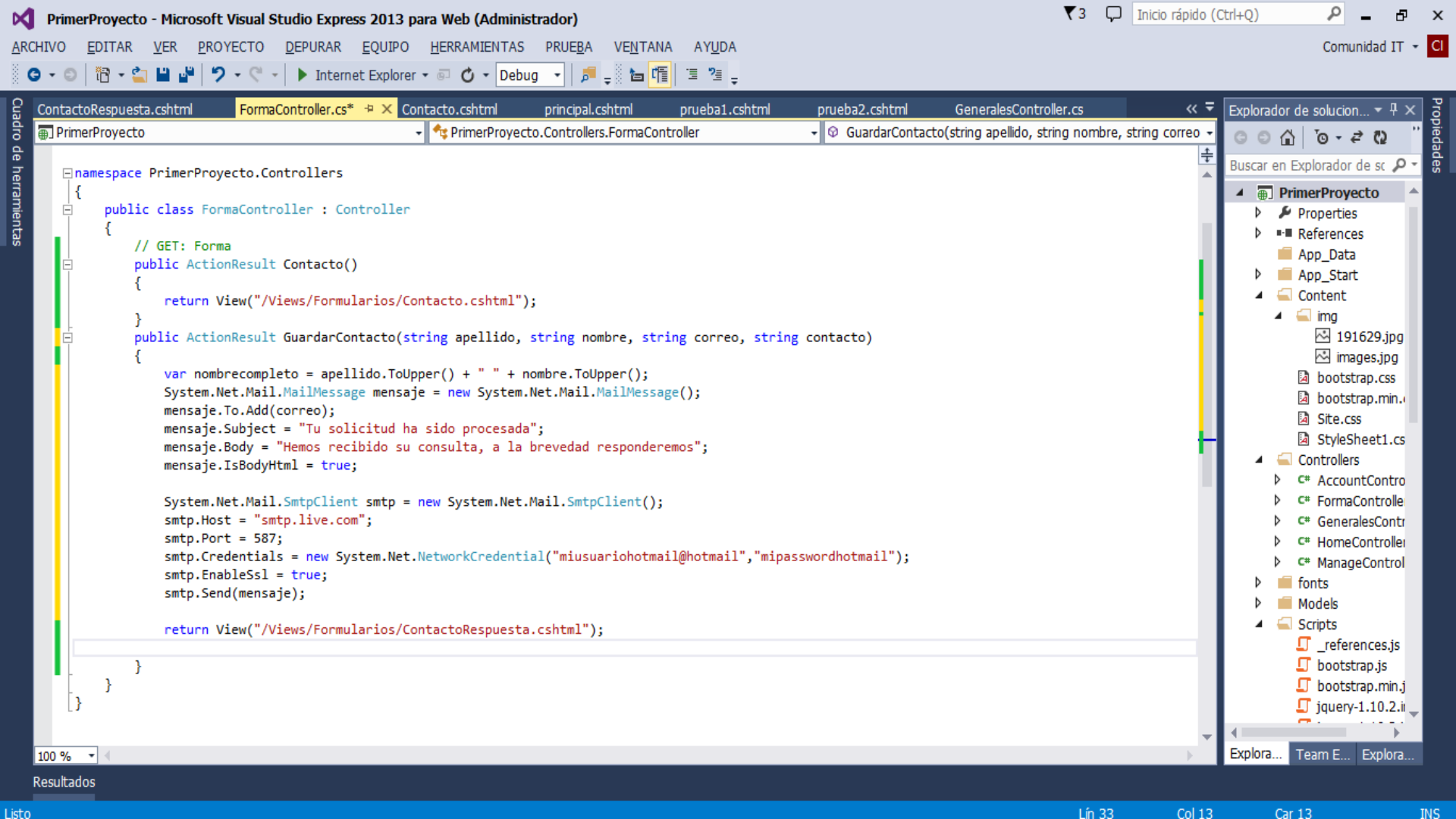
Crear o usar una base creada

La primera vez, debemos crear una estructura de base de datos, con sus tablas, sus campos y datos de pruebas.

Crear o usar una base creada

Siguiendo el ejercicio del capítulo anterior crear la tabla Contactos y Localidades, y cargue datos.

Si se trata del mismo proyecto, omita este paso. La base de datos ya está creada



Sobre el formulario de contacto creado en los ejercicios anteriores

Además de enviar un email, vamos a proceder a guardar (también puede guardarse sin enviar el correo, para esto habría que reemplazar el código)



Los parámetros básicos para interactuar con una base de datos son

Dirección IP del Servidor de Base de  
Datos

Mecanismo de Autenticación

Credenciales de Autenticación

Nombre de la Base de Datos

En nuestro caso los parámetros son

Dirección IP: localhost –significa que está instalado en nuestra máquina-

Mecanismo de autenticación:

Credenciales de autenticación: no es necesario

Nombre de la Base de Datos: en mi ejemplo, proyecto.

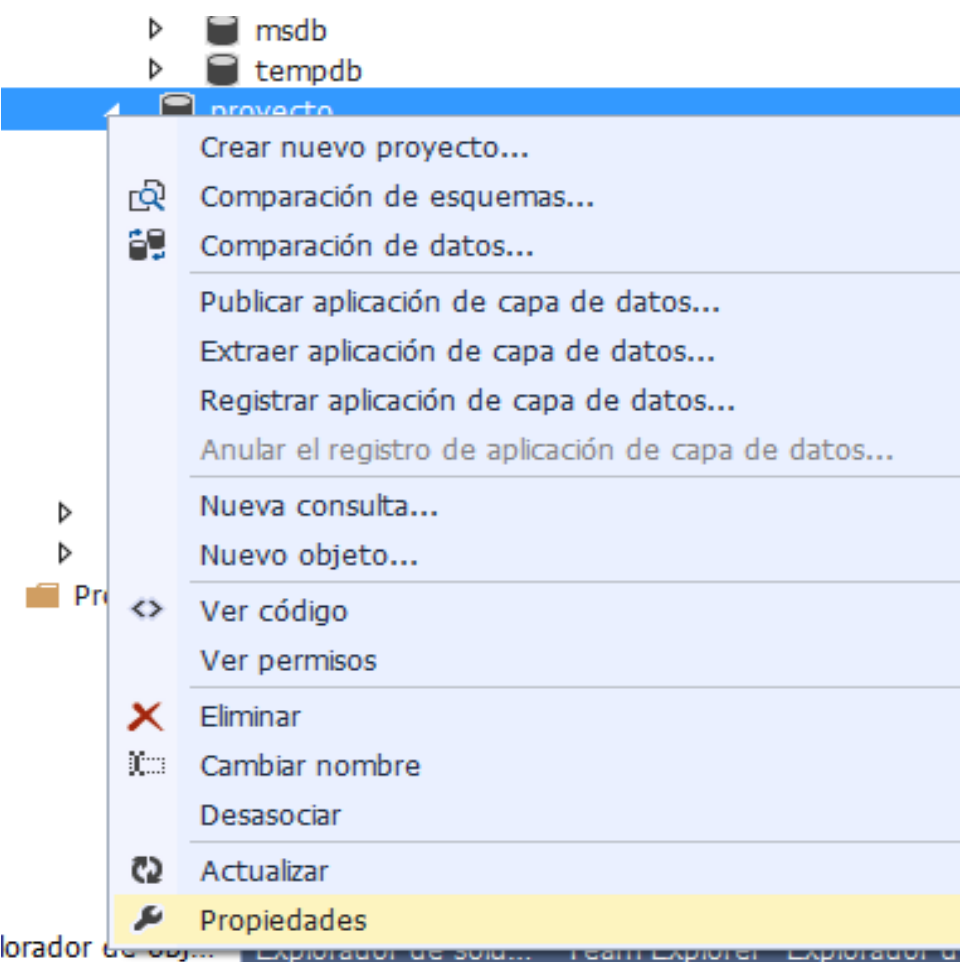
Para conectar los datos con nuestra aplicación bastan 6 pasos

- 1) Crear la conexión
- 2) Crear la sentencia
- 3) Asociar la sentencia a la conexión
- 4) Ejecutar la sentencia
- 5) Leer el resultado de la sentencia
- 6) Cerrar la conexión

## 1. Crear la conexión

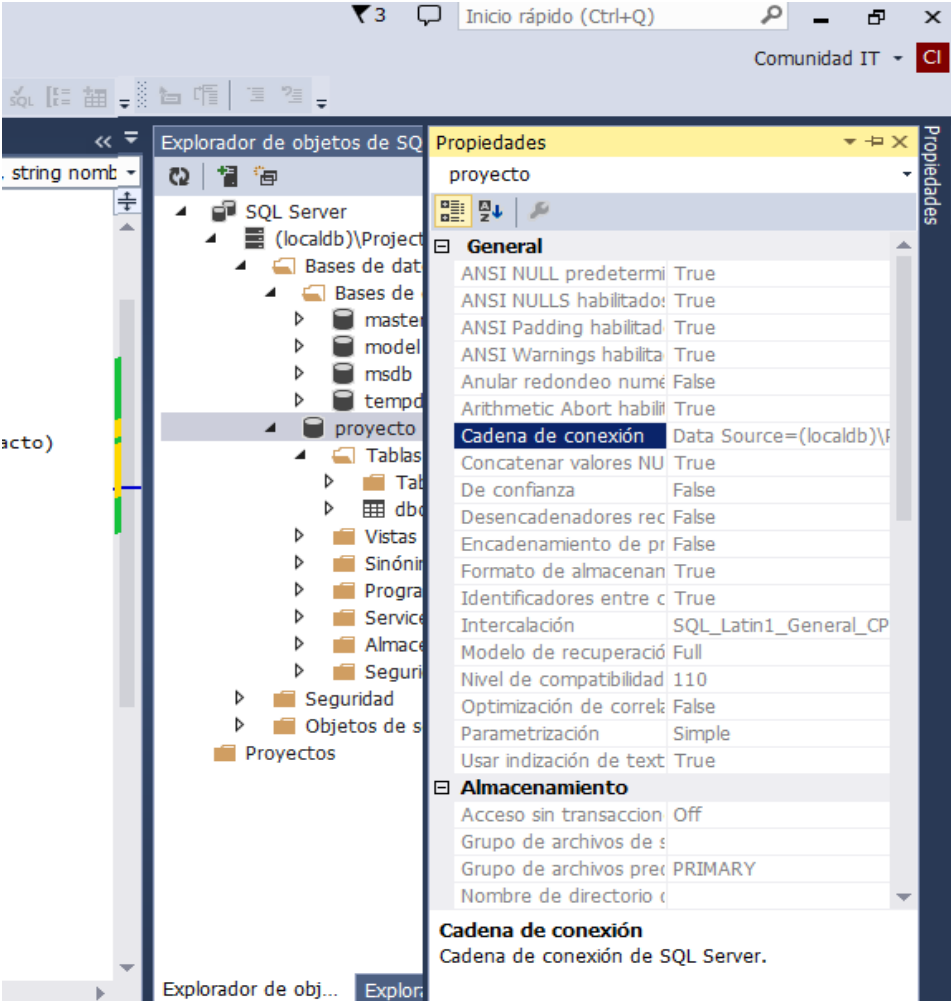
```
var connection = new System.Data.SqlClient.SqlConnection();  
connection.ConnectionString = "una conexión";  
connection.Open();
```

Donde «una conexión» se obtiene haciendo click derecho en el explorador de objetos de sql, a la derecha de la pantalla.



Para extraer la cadena de conexión

Haga click en el  
explorador de  
Objetos de SQL.  
Elija la Base de  
Datos. Click  
derecho,  
propiedades



Copie la cadena de conexión

Es un texto largo,  
algo parecido a esto  
Data

Source=(localdb)\Projects;Initial  
Catalog=proyecto;Integrated  
Security=True;Connect  
Timeout=30;Encrypt=False;Trust  
ServerCertificate=False

## 2. Crear la sentencia

```
var sentence = new System.Data.SqlClient.SqlCommand();  
sentence.CommandType = System.Data.CommandType.Text;  
sentence.CommandText = "Insert into Contacto (nombre,apellido,correo),  
values (@pnombre, @papellido, @pcorreo)";  
sentence.Parameters.Add( new  
System.Data.SqlClient.SqlParameter("pnombre", nombre ));  
sentence.Parameters.Add( new  
System.Data.SqlClient.SqlParameter("papellido", apellido ));  
sentence.Parameters.Add( new System.Data.SqlClient.SqlParameter("pcorreo",  
correo ));
```

### 3. Asociar sentencia a conexion

```
sentence.Connection = connection;
```

Es la sentencia menos compleja





5. Leer el resultado de la sentencia

```
var result = sentence.ExecuteNonQuery();
```

```
var mensaje = "";
```

```
if (result == 0)
```

```
    mensaje = "Exitoso!";
```

```
else
```

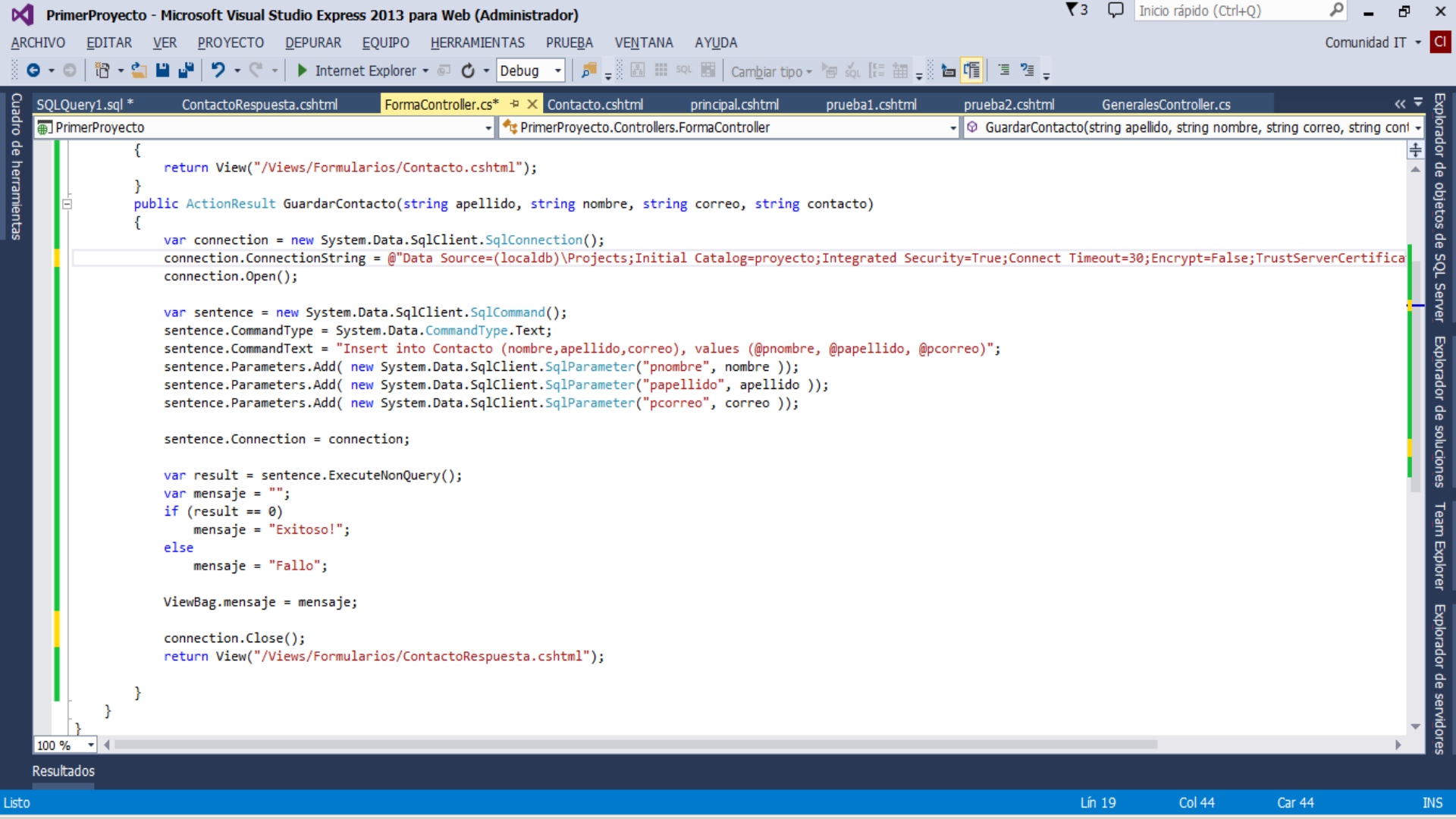
```
    mensaje = "Fallo";
```

Según la sentencia podría entregarse uno u otro mensaje

## 6. Cerrar la conexión

```
connection.Close();
```

Si no se cierra la conexión, se corre el riesgo de no poder conectarse más tarde desde nuestra aplicación



continúe ejercitando

Investiga las variantes para poder efectuar una consulta y mostrar los datos.

Ten en cuenta que una vez que obtengas los datos de la base de datos puedes enviarlos al controlador utilizando el ViewBag y luego imprimirlos en la vista

continúe ejercitando

Investiga las variantes para borrar un dato.  
Usa el buscador web para buscar ejemplos de escenarios CRUDL (Create Read Update Delete y List) bajo ASP.NET MVC, intenta implementarlos.

Coloca las claves «CRUDL Asp.NET»