Universidad Tecnológica de El Salvador

FACULTAD DE INFORMÁTICA Y CIENCIAS APLICADAS.
ESCUELA DE INFORMÁTICA.
CATEDRA DE PROGRAMACIÓN

PROGRAMACIÓN II

Proyecto de Investigación de Cátedra – Clase

CICLO: 02 – 2024

SAN SALVADOR, septiembre de 2024

Carrera	Ingeniería, Licenciatura, Técnicos		
Asignatura	Programación II	**Sección:**	02
Docente	Juan José Santos Escobar		
Fecha de entrega			
Unidad de aprendizaje 3: Diseño de aplicaciones web, utilizando base de datos y el modelo de Capas con MVC.			

Introducción.

ASP.NET ofrece tres modelos de programación:

- Páginas web (Web Pages).

Es el modelo de programación más simple para desarrollar páginas web ASP.NET. Proporciona una manera fácil de combinar HTML, CSS, JavaScript y código de servidor. Sus características principales son:

  - Fácil de aprender, entender y usar.
  - Construido alrededor de páginas web sencillas.
  - Similar a PHP y ASP clásico.
  - Permite escribir código de servidor con Visual Basic o C#.
  - Control total de HTML, CSS y JavaScript.

Las páginas web son fácilmente extensibles con ayudas web programables, incluyendo base de datos, video, gráficos, redes sociales y más.

- MVC.

Es un marco de trabajo para construir aplicaciones web utilizando el diseño MVC (Modelo Vista Controlador), el cual promueve la separación de áreas de interés por medio de:

  - El Modelo, que representa el núcleo de la aplicación.
  - La Vista, que muestra los datos.
  - El Controlador, que maneja la entrada.

El modelo MVC define aplicaciones web con tres capas lógicas:

  - La capa de negocios (lógica del Modelo).
  - La capa de visualización (lógica de la Vista).
  - El control de entrada (lógica del Controlador).

- Formularios web (Web Forms).

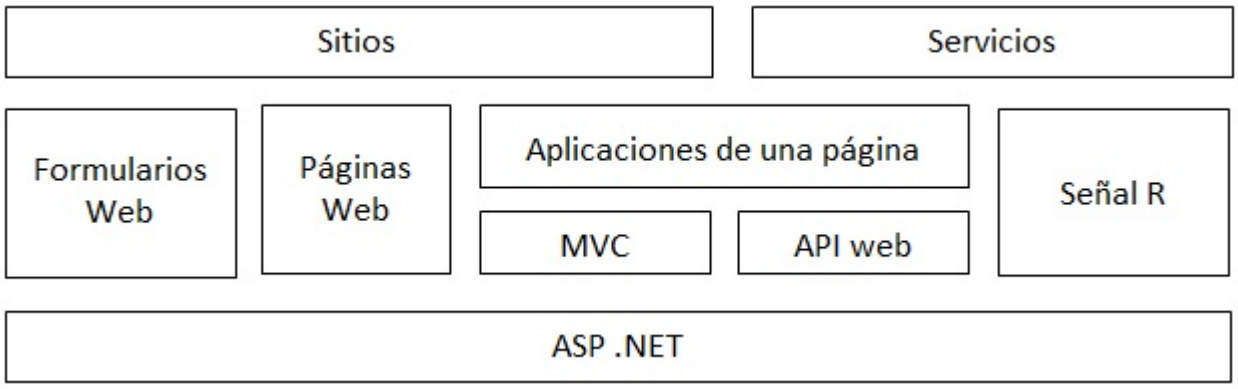
Es el modelo de programación ASP.NET más antiguo, con páginas web manejadas por evento como una combinación de HTML, controles de servidor y código de servidor.

Los formularios web son compilados y ejecutados en el servidor, el cual genera el HTML que muestra las páginas web.

Los formularios web vienen con cientos de controles web diferentes y con componentes web para construir sitios web controlados por el usuario con acceso a datos.

Panorama general de ASP.NET.

La figura siguiente muestra la arquitectura de ASP.NET distribuida en tres capas:



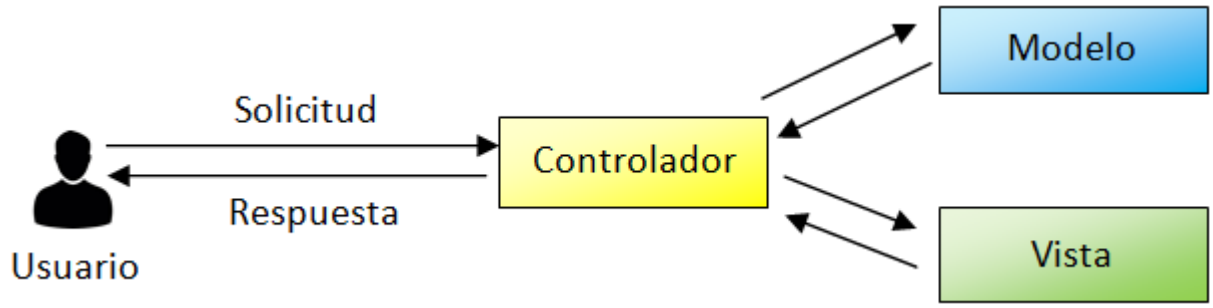
La capa inferior muestra a ASP.NET, el cual es un núcleo común sobre el cual se apoyan varios modelos.

La capa central contiene los diferentes modelos que hacen uso de ASP.NET.

La capa superior muestra los dos posibles contextos a ser entregados para los navegadores web.

Funcionamiento de MVC en ASP.NET.

La figura siguiente muestra de una manera sencilla el funcionamiento del modelo MVC en ASP.NET.



Su funcionamiento es el siguiente:

- El usuario realiza una solicitud por medio de la llamada a un método de acción que se encuentra en una clase.
- El controlador recibe la solicitud.
- El controlador recupera el modelo, el cual puede contener una tabla, una cadena, código XML, datos en formato json, etc.
- El controlador envía el modelo a la vista.
- La vista es una plantilla que representa al modelo de forma visual, tomando los datos y convirtiéndolos en marcaciones HTML.

- La vista es devuelta al usuario, la cual normalmente contiene HTML, JavaScript y CSS.

Características del modelo MVC.

MVC adopta el hecho que HTTP no tiene estado y por lo tanto no proporciona abstracción de estado. Corresponde a los desarrolladores encargarse del manejo del estado.

La arquitectura MVC tiene tres componentes principales sobre los cuales se trabaja para hacer funcionar la aplicación web:

- El **Modelo**: es un conjunto de clases que contiene los datos con los cuales se está trabajando, así como las reglas del negocio respecto a cómo los datos pueden ser cambiados y manipulados.
- La **Vista**: define cómo la interfaz del usuario de la aplicación será mostrada.

- El **Controlador**: es un conjunto de clases que se encarga de la comunicación por parte del usuario, el flujo general de la aplicación y la lógica específica de la aplicación.

El modelo MVC aplicado a páginas web ASP.NET.

El modelo MVC es usado frecuentemente en programación web. En ASP.NET es traducido aproximadamente como:

- El **Modelo**: son clases que representan el dominio en el cual está interesado. Estos objetos de dominio a menudo encapsulan datos almacenados en una base de datos, así como código que manipula los datos y aplica la lógica del negocio específica del dominio. En ASP.NET MVC esto es muy probablemente una capa de acceso a datos de algún tipo, usando una herramienta como **Entity Framework** o **NHibernate** combinada con el código a la medida contiendo la lógica específica del negocio.
- La **Vista**: es una plantilla que dinámicamente genera HTML.
- El **Controlador**: es una clase especial que administra la relación entre la Vista y el Modelo. Responde a las entradas del usuario, habla con el Modelo y decide cuál vista proporcionar (si hay alguna). En ASP.NET MVC esta clase es convencionalmente denotada por el sufijo Controller.

Diferencias entre el modelo MVC y el modelo de Formularios Web.

A continuación, se muestra un cuadro comparativo con las principales diferencias.

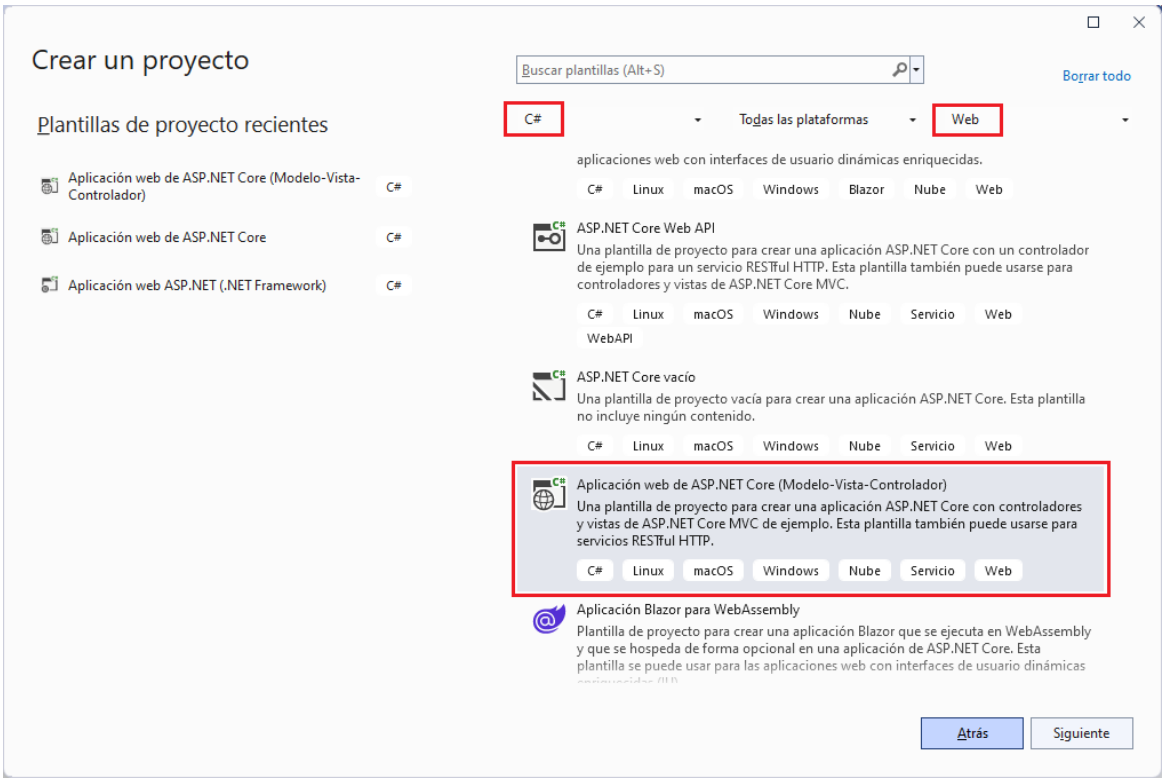
Modelo MVC	Modelo de Formularios Web
Toda aplicación web necesita una estructura y este modelo favorece la organización desde el principio hasta el final.	Arrastrar y soltar controles sobre un formulario no garantiza una estructura definida, pudiendo generar una estructura ambigua después de algunos meses.
Normalmente se genera menos código.	Normalmente contiene más código.
Ofrece una curva de aprendizaje suave.	La curva de aprendizaje puede ser pronunciada.
Proporciona un ciclo de vida único para todas las páginas del sitio.	Duplica el ciclo de vida por cada página del sitio.
No proporciona abstracción de estado porque adopta el hecho que Http no tiene estado.	Maneja estados, produciendo código complejo para recordar lo que está sucediendo en el sitio, aunque no exista ninguna actividad.
Fácilmente genera unidades de prueba que permiten ejecutar el nuevo código antes de su incorporación final.	No posee esta característica.

En resumen, el modelo de programación MVC es una alternativa más ligera que los formularios web tradicionales, ya que es un marco ligero y altamente comprobable, integrado con todas las características de ASP.NET existentes, como páginas maestras, seguridad y autenticación.

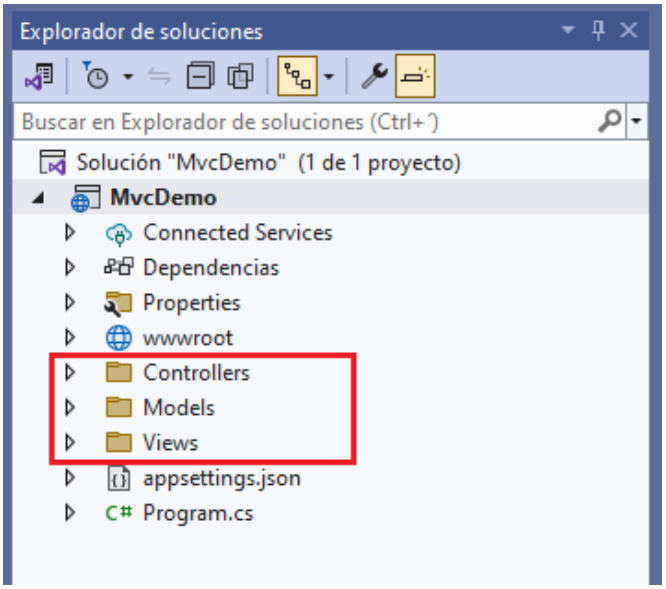
Generación de un proyecto MVC.

Para generar un proyecto MVC, realice lo siguiente:

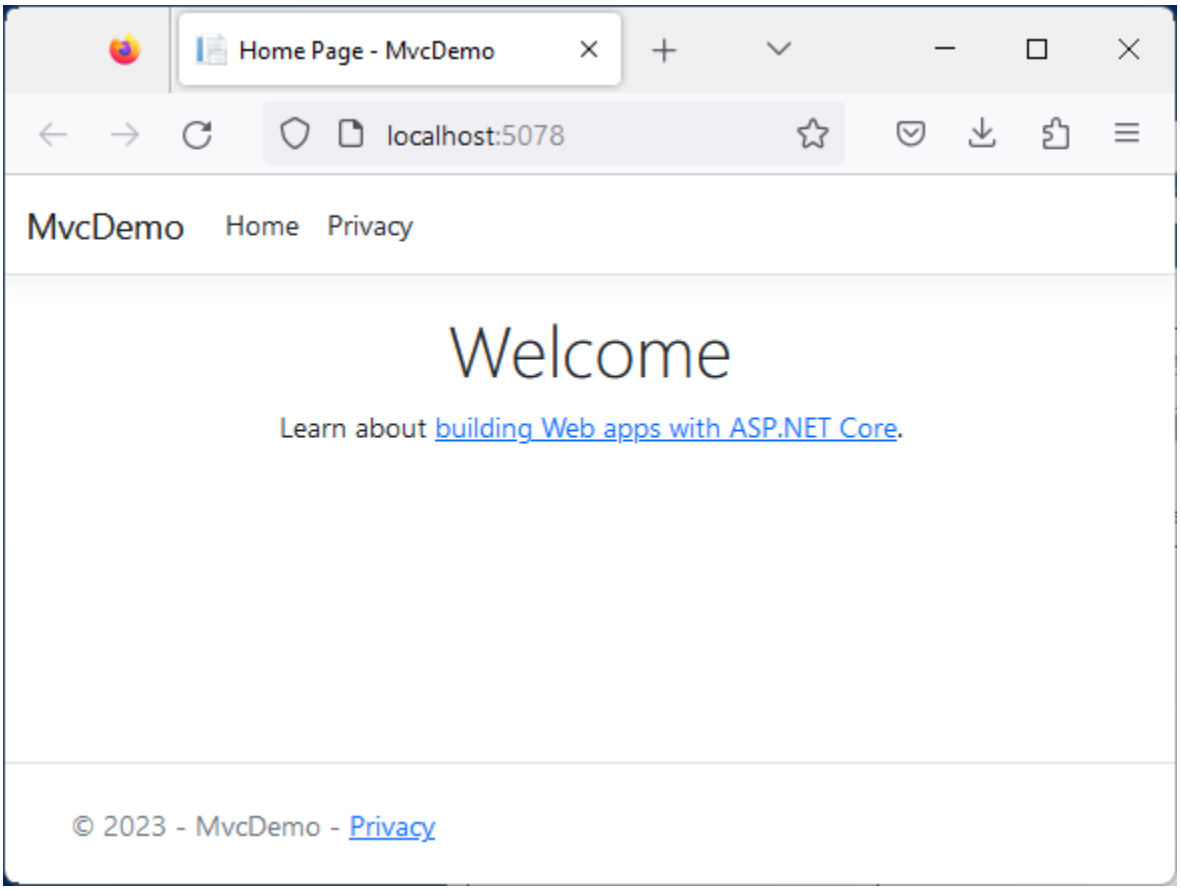
- Construya un nuevo proyecto definiendo las siguientes selecciones en el orden mostrado:
  - Lenguaje de programación: **C#**.
  - Tipo de proyecto: **Web**.
  - Plantilla: **Aplicación web de ASP.NET Core (Modelo-Vista-Controlador) Una plantilla de proyecto para crear una aplicación ASP.NET Core con controladores y vistas de ASP.NET Core MVC de ejemplo. Esta plantilla también puede usarse para servicios RESTful HTTP**.



- Haga clic sobre el botón *Siguiente*.
- Coloque el nombre *Ejemplo1MVC*.
- Presione el botón *Siguiente*.
- Desmarque todas las casillas de verificación.
- Observe que en la ventana **Explorador de soluciones** aparecen las carpetas **Models**, **Views** y **Controllers**, tal como se muestra en la figura siguiente:



- Ejecute el proyecto presionando **CRTL + F5**. Debería obtener una pantalla similar a la mostrada:



- Abra la carpeta **Controllers** y seleccione el archivo **HomeController.cs**. Revise el fragmento de código siguiente:

```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
    ...
```

El método de acción **Index()** contiene el código para devolver una vista. Este método es invocado cuando se carga la página.

- Abra la subcarpeta *Home* que se encuentra dentro de la carpeta **Views** y seleccione el archivo **Index.cshtml**. Este es el código que es invocado desde el método de acción mostrado en el numeral anterior.

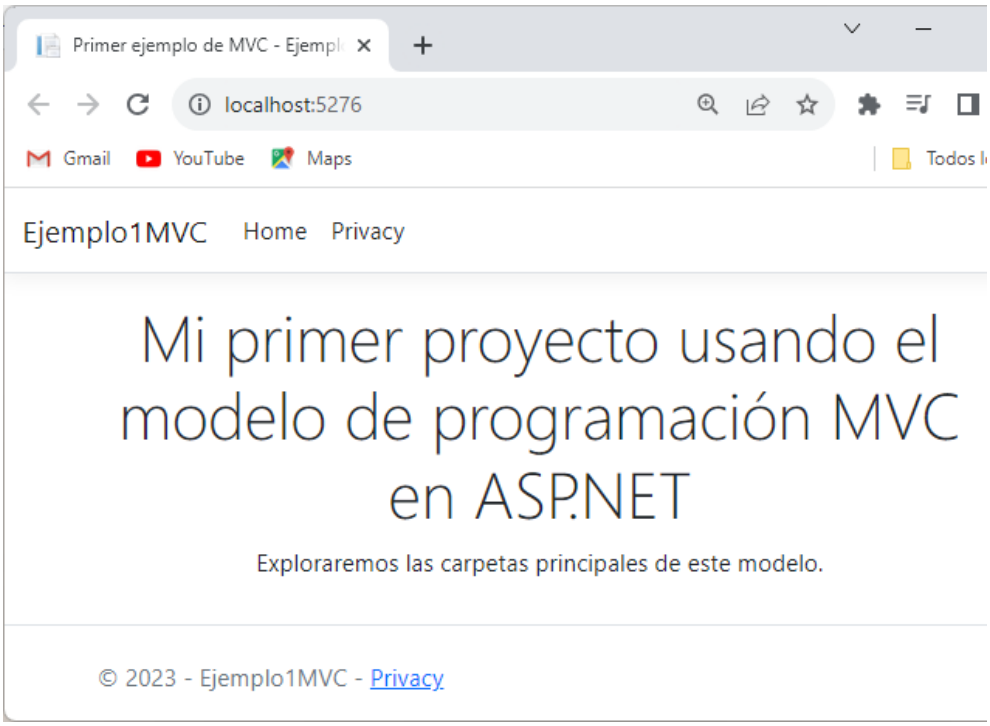
Revise el fragmento de código siguiente:

```
@{
   ViewData["Title"] = "Home Page";
}

<div class="text-center">
   <h1 class="display-4">Welcome</h1>
   <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web
apps with ASP.NET Core</a>.</p>
</div>
```

- **Directiva Razor: @{ ViewData["Title"] = "Home Page"; }**
- Esta línea establece el título de la página utilizando ViewData, un diccionario que permite pasar datos entre el controlador y la vista.
- Modifique el código anterior como se muestra en el código resaltado:

```
@{
   ViewData["Title"] = "Primer ejemplo de MVC";
}

<div class="jumbotron">
<h1>Mi primer proyecto usando el modelo de programación MVC en
ASP.NET</h1>
<p class="lead">Exploraremos las carpetas principales de este modelo.</p>
</div>
...
```

- Vuelva a ejecutar el proyecto. Ahora debería obtener una pantalla similar a la siguiente:



- Modifique el tamaño de la ventana y observe cómo se reacomoda todo el contenido. Este es un efecto generado por **bootstrap**, una librería incluida por defecto dentro de la carpeta **wwwroot, lib, bootstrap**.

Diferencias entre el URL de una aplicación MVC y una aplicación Web Forms.

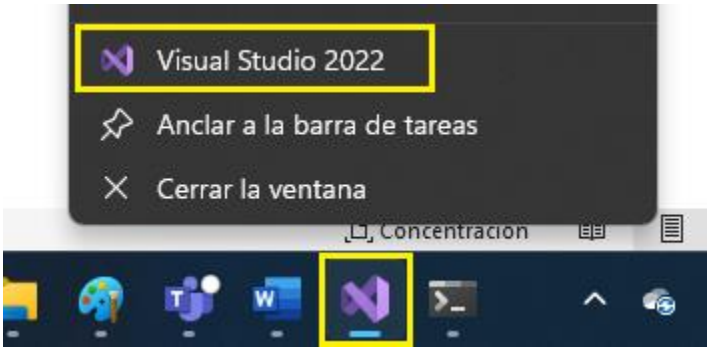
Respecto al URL entre ambos modelos, es importante tener en cuenta lo siguiente:

1. En el modelo **MVC**, la URL es mapeada hacia los métodos de acción de los controladores.
2. En el modelo **Web Forms**, la URL es mapeada hacia archivos físicos.

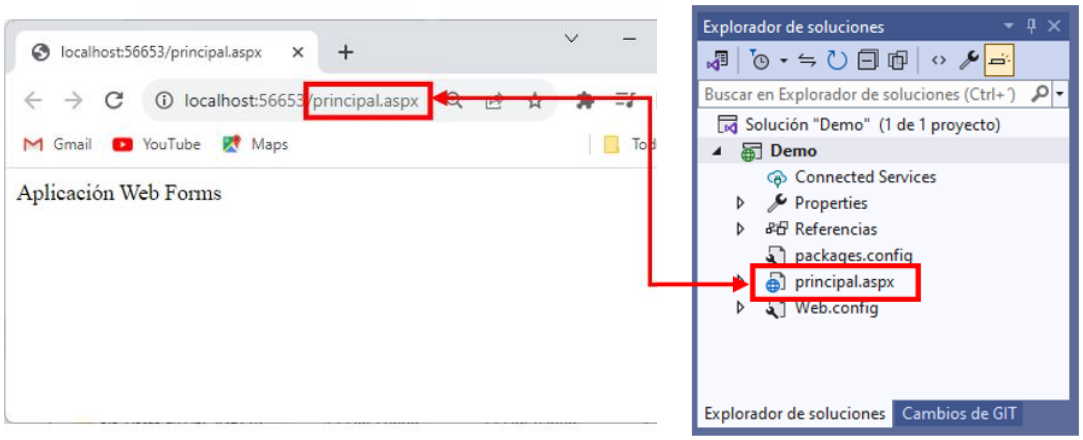
Evidentemente MVC presenta una ventaja de seguridad importante porque no expone el nombre de sus archivos.

Para demostrar lo anterior abra una segunda instancia de Visual Studio y construya un proyecto **Web Forms** de nombre **WebFormsDemo**, agregue una etiqueta y configure su propiedad **Text** con el valor "*Aplicación Web Forms*".

NOTA: Puede abrir una instancia adicional de Visual Studio haciendo clic derecho sobre el ícono que se encuentra en la barra de tareas de Windows y seleccionando la opción de Visual Studio que tiene instalada en su computadora. Recuerde que este material ha sido preparado con la versión 2022, por lo que las capturas de pantalla mostradas corresponden a esa versión.



Al ejecutar esta aplicación, su URL debería ser similar a la figura siguiente:



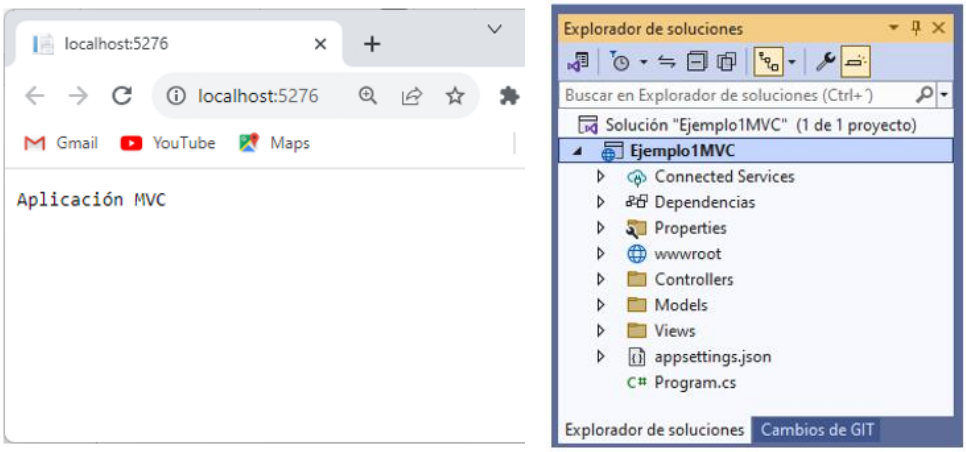
Observe que el nombre de la página mostrada en el navegador corresponde al nombre de un archivo físico mostrado en la ventana **Explorador de soluciones**.

Ahora cierre el proyecto *Ejemplo1MVC*, construya un proyecto **MVC** de nombre *MVCDemo* y modifique el método de acción **Index()** del archivo **HomeController.cs** como se muestra:

```
public class HomeController : Controller
{
    public string Index()
    {
        return "Aplicación MVC";
```

```
    }
...
```

Al ejecutar esta aplicación, su URL debería ser similar a la figura siguiente:



Observe que la información mostrada en el URL no hace referencia a ninguno de los elementos de la ventana **Explorador de soluciones**.

Controladores

Los controladores dentro del modelo MVC son responsables de manejar las entradas del usuario, a menudo haciendo cambios al modelo en respuesta a la entrada del usuario. De esta manera, los controladores tienen que ver con la esencia de la aplicación, trabajando con datos que llegan y proporcionando datos hacia la vista relevante.

En el modelo MVC, el URL le dice al mecanismo de ruteo cuál clase controladora instanciar y cuál método de acción llamar, y proporciona los argumentos requeridos a ese método. El método del controlador entonces decide cuál vista usar y esa vista entonces genera el HTML.

En lugar de tener una relación directa entre el URL y un archivo en el disco duro del servidor, existe una relación entre el URL y un método en una clase controladora. ASP.NET MVC implementa la variante del controlador frontal del modelo MVC y el controlador se ubica frente a todo excepto el subsistema de ruteo.

NOTA: MVC requiere que el nombre de todos los controladores finalice con el sufijo "**Controller**".

El controlador HomeController.

Este controlador es el responsable de la página de inicio en la raíz del sitio web, y contiene el código siguiente:

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

namespace Ejemplo1MVC.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }


        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
            NoStore = true)]
        public IActionResult Error()
```

```
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id
                ?? HttpContext.TraceIdentifier });
        }
    }
}
```

Observe que esta es una clase simple que hereda de la clase base **Controller**. El método **Index()** es responsable de decidir qué sucede cuando navega hacia la página de inicio del sitio web.

Sin embargo, persiste la pregunta "¿Cómo son mapeados los URL hacia los métodos de acción del controlador?" La respuesta es utilizando el ruteo de ASP.NET que se encuentra dentro del archivo **Program.cs**, cuyo código original es el siguiente:

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}
app.UseStaticFiles();
```

```
app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

El código resaltado define lo siguiente:

1. Un nombre por defecto (Default).
2. Una estructura para el URL (controlador/acción/parámetro).
3. Valores por defecto para el controlador (Home) y para el método de acción (Index). Adicionalmente define el parámetro como opcional.

Al ejecutar la aplicación, se invoca exitosamente el método de acción **Index()** contenido en el controlador **HomeController** aunque no sea especificado en el URL.

Ejecute nuevamente la aplicación y modifique el URL para pasar solamente el nombre del controlador (http://localhost:5078/home/). Debería obtener el mismo resultado.

Modifique nuevamente el URL para pasar el nombre del controlador y del método (http://localhost:5078/home/index/). De nuevo, debería obtener el mismo resultado.

Por último, para probar el parámetro agregue un valor al final del URL (http://localhost:5078/home/index/10). Invariablemente debería obtener el mismo resultado.

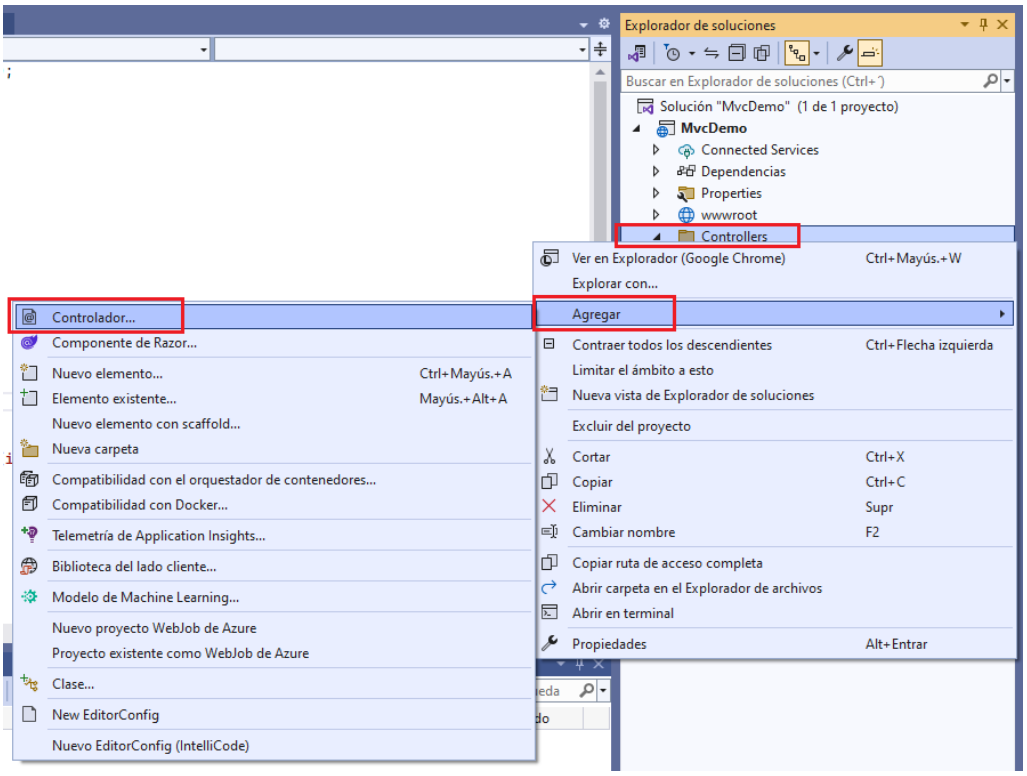
Si desea manejar el valor enviado en el URL cambie el código del método de acción **Index()** como se muestra:

```
namespace Ejemplo1MVC.Controllers
{
    public class HomeController : Controller
    {
        public string Index(string id)
        {
            return "id = " + id;
        }
        ...
```

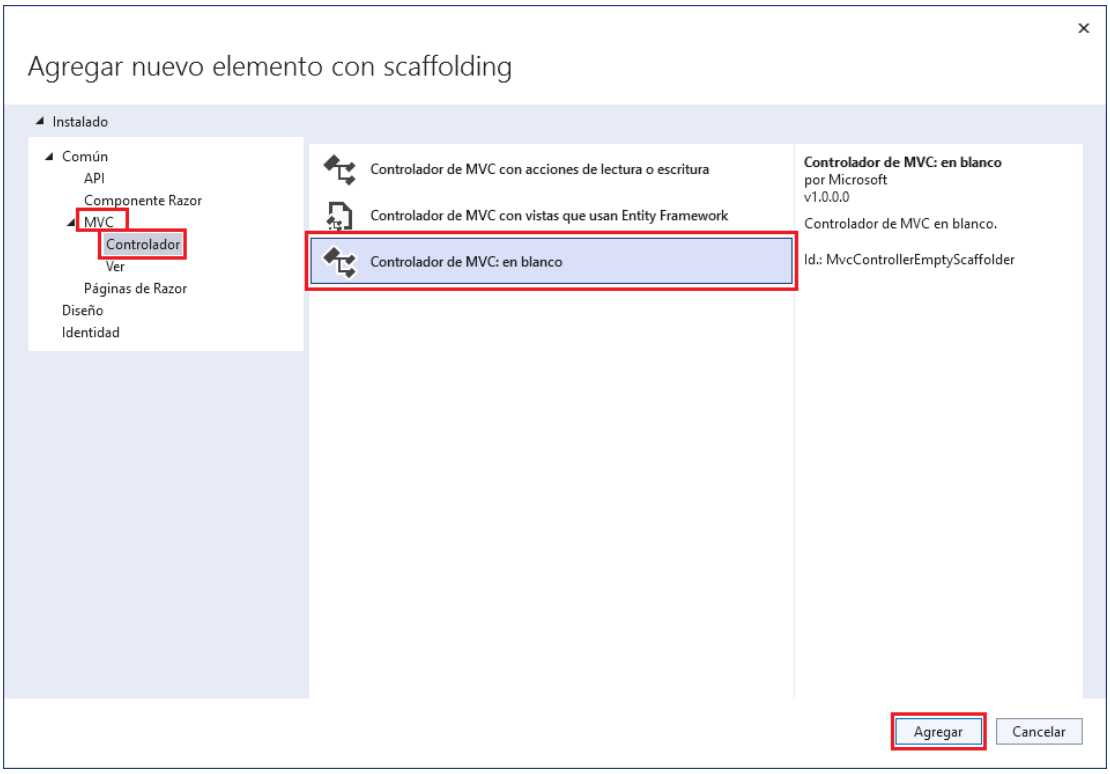
Ejecute nuevamente la aplicación utilizando todas las variantes propuestas para el URL.

Agregar un controlador.

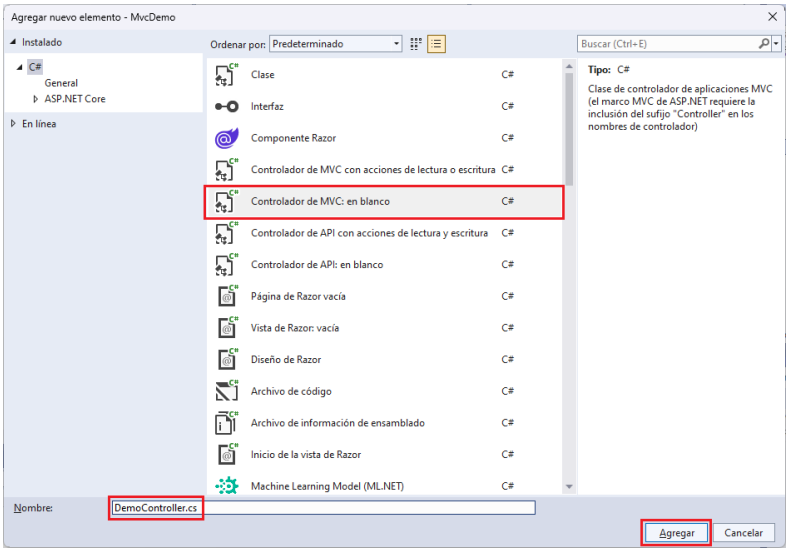
En la ventana **Explorador de soluciones** haga clic derecho sobre la carpeta **Controllers** y seleccione la opción **Agregar**, **Controlador**.



En el cuadro de diálogo **Agregar nuevo elemento con scaffolding**, seleccione la opción **MVC**, **Controlador**, **Controlador de MVC: en blanco** y presione el botón **Agregar**.



En el cuadro de diálogo **Agregar nuevo elemento**, seleccione la opción **Controlador de MVC: en blanco**, coloque el nombre *DemoController* y presione el botón **Agregar**.



Si ejecuta la aplicación obtendrá un mensaje de error debido a que el nuevo controlador no tiene ninguna vista asociada, que es precisamente lo que se invoca en su código. En su lugar, reemplace el contenido mostrado con el código siguiente:

```
using Microsoft.AspNetCore.Mvc;
using System.Text.Encodings.Web;

namespace MvcDemo.Controllers;

public class DemoController : Controller
{
    public string Index()
    {
        return "Este es el método de acción por defecto (\"Index()\")";
    }

    public string Bienvenida()
    {
        return "Este es el método de acción \"Bienvenida()\"";
    }
}
```

Cada método público en un controlador se puede llamar como un **punto final HTTP**. En el ejemplo anterior, ambos métodos devuelven una cadena.

Un **punto final HTTP**:

- Es una URL a la que se puede acceder en la aplicación web, como por ejemplo *https://localhost:5001/Home/Index*.
- Combina:
    - El protocolo utilizado: HTTPS.
    - La ubicación de red del servidor web, incluido el puerto TCP: localhost:5001.
    - El URI de destino: Bienvenida

Al iniciar la aplicación sin proporcionar ningún segmento de URL, el valor predeterminado es el controlador "*Home*" y el método "*Index*", donde:

- El primer segmento de URL determina la clase de controlador que se ejecutará. Entonces *localhost:5001/Home* se asigna a la clase Controlador *HomeController*.

- La segunda parte del segmento de URL determina el método de acción en la clase. Entonces *localhost:5001/Home/Index* hace que se ejecute el método *Index* de la clase *HomeController*.
- Se debe tener en cuenta que se puede navegar hasta *localhost:5001/Home* y se llamará al método *Index* de forma predeterminada si no se especifica explícitamente un nombre de método.

MVC invoca las clases controladoras, así como los métodos de acción dentro de ellas, dependiendo de la URL ingresada. La lógica de enrutamiento de URL por defecto de MVC usa un formato similar al siguiente:

/[Controlador]/[MétodoDeAcción]/[Parámetros]

Sin embargo, cuando no se proporciona ningún segmento de URL, el valor predeterminado es el controlador "*Home*" y el método "*Index*" especificado en la sección **app.MapControllerRoute** del archivo **Program.cs**.

Pruebe la aplicación con los valores URL siguientes:

- Sin especificar nada en la URL (cuando se ejecuta la aplicación).
- Usando *http://localhost:5078/Demo/Index*
- Usando *http://localhost:5078/Demo/Bienvenida*

Ahora reemplace el método de acción *Bienvenida* con el código siguiente:

```
public string Bienvenida(string nombre, int id = 1)
{
    return HtmlEncoder.Default.Encode($"Hola {nombre}, su id es: {id}");
}
```

Pruebe la aplicación con los valores siguientes:

- Usando *http://localhost:5078/Demo/Bienvenida*
- Usando *http://localhost:5078/Demo/Bienvenida?nombre=Juan*
- Usando *http://localhost:5078/Demo/Bienvenida?nombre=Juan&id=4*

En las cadenas anteriores:

- El signo de interrogación (?) es un separador entre la dirección URL y los parámetros.
- El signo & separa los pares clave-valor.

Finalmente, en el archivo **Program.cs** cambie el código de la ruta por defecto.

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Demo}/{action=Bienvenida}/{id?}");
```

Pruebe la aplicación con los valores siguientes:

- Sin especificar nada en la URL (cuando se ejecuta la aplicación).
- Usando *http://localhost:5078/Demo/Bienvenida*
- Usando *http://localhost:5078/Demo/Bienvenida?nombre=Juan*
- Usando *http://localhost:5078/Demo/Bienvenida?nombre=Juan&id=4*