

Programación Web .NET Core

Módulo 3



Introducción a ASP.NET MVC Framework

Desarrollo de un sitio web con ASP.NET MVC Framework

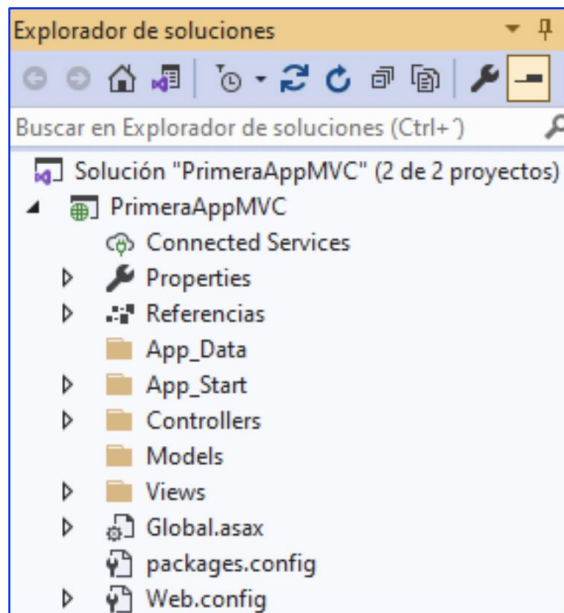
El Framework ASP.NET MVC implementa el **patrón Modelo-Vista-Controlador**. El patrón Modelo-Vista-Controlador busca separar la lógica de negocios de nuestra aplicación (Modelo), la visualización de los datos (Vista) y la comunicación entre éstos (Controlador).

Cuando se crea un proyecto de este tipo en Visual Studio .Net se pueden identificar **tres carpetas: *Models*, *Views* y *Controllers***.



¿Cuál será el contenido de estas carpetas?

- **Models:** aquí se implementan las clases y estructuras de datos que representen las entidades y la lógica de negocios de la aplicación.
- **Views:** en ella se establecerá cómo se mostrará la interfaz a los usuarios de la aplicación.
- **Controllers:** en esta carpeta se implementará el conjunto de clases que administrarán la comunicación entre las solicitudes que haga el cliente (Navegador), el modelo que procesará los datos de las solicitudes que lleguen del cliente. Comunicará a las vistas para que muestren los datos solicitados por el cliente.



Una representación gráfica del flujo de comunicación entre el navegador que hace la solicitud y la aplicación, podría ser la que vemos a continuación:



Controlador

Controlador

Para comprender mejor esta metodología de desarrollo de un sitio web usando ASP.NET MVC, se comenzará planteando escenarios simples para luego agregarles un poco más de complejidad.

En el primer escenario se resolverá toda la lógica de la aplicación dentro del **Controlador**, o sea, no codificaremos **Vistas** ni **Modelos**. Desde ya, esta metodología no es nada recomendable para utilizar en un escenario complejo, pero nos servirá para saber dónde desarrollar nuestro código.

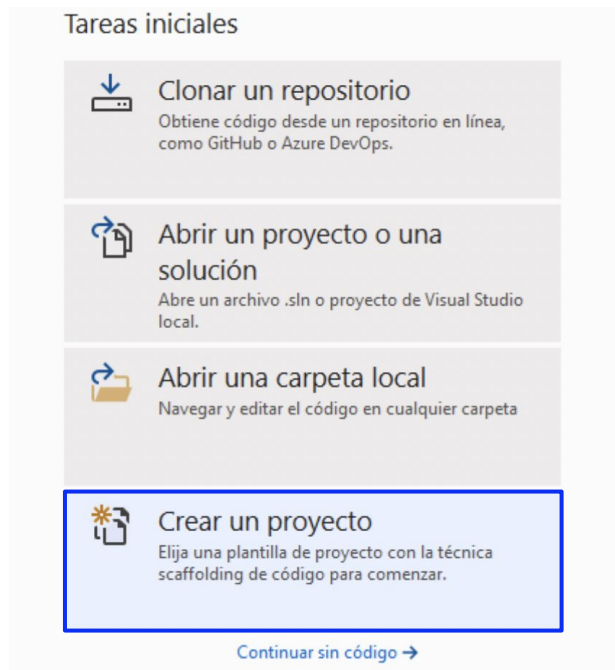


Escenario

Crear un sitio web que muestre en la página principal el nombre de una organización y una breve presentación, y en otra página los integrantes de la organización.

1. Comenzaremos con la creación de un nuevo proyecto. Seleccionar: **Crear un Proyecto**.

Veamos la secuencia de pasos, en las imágenes de las siguientes pantallas.



Crear un proyecto

Buscar plantillas (Alt+S)



Borrar todo

Plantillas de proyecto recientes

C#

Windows

Web



Aplicación web ASP.NET (.NET Framework)

Plantillas de proyecto para crear aplicaciones ASP.NET. Puede crear aplicaciones ASP.NET Web Forms, MVC o Web API y agregar muchas otras características en ASP.NET.

C#

Windows

Nube

Web



Prueba de controladores web para Edge (.NET Core)

Proyecto que contiene pruebas unitarias que pueden automatizar las pruebas de IU de los sitios web en el explorador Edge (con Microsoft WebDriver).

C#

Windows

Web

Prueba



Prueba de controladores web para Edge (.NET Framework)

Proyecto que contiene pruebas unitarias que pueden automatizar las pruebas de IU de los sitios web en el explorador Edge.

Activar Windows

Atrás

Siguiente

Ve a Configuración para activar Windows.

Configure su nuevo proyecto

Aplicación web ASP.NET (.NET Framework) C# Windows Nube Web

Nombre del proyecto

PrimeraAppMVC

Ubicación

C:\Users\amerc\source\repos

Nombre de la solución ⓘ

PrimeraAppMVC

☐ Colocar la solución y el proyecto en el mismo directorio

Framework

.NET Framework 4.7.2

Crear una aplicación web ASP.NET

**Vacio**

Una plantilla de proyecto vacía para crear aplicaciones ASP.NET. Esta plantilla no tiene contenido.

**Web Forms**

Una plantilla de proyecto para crear aplicaciones de ASP.NET Web Forms. ASP.NET Web Forms le permite crear sitios web dinámicos con un modelo familiar controlado por eventos para arrastrar y colocar. Una superficie de diseño y cientos de controles y componentes le permiten crear rápidamente sofisticados y eficaces sitios controlados por la interfaz de usuario y con acceso a datos.

**MVC**

Una plantilla de proyecto para crear aplicaciones ASP.NET MVC. ASP.NET MVC permite compilar aplicaciones mediante la arquitectura de controlador de vista de modelos. ASP.NET MVC incluye muchas características que permiten un desarrollo rápido orientado a pruebas para crear aplicaciones que usan los últimos estándares.

**API web**

Plantilla de proyecto para crear servicios HTTP REST que pueden llegar a una amplia gama de clientes, como, por ejemplo, exploradores y dispositivos móviles.

**Aplicación de página única**

Una plantilla de proyectos para crear aplicaciones HTML5 atractivas controladas por JavaScript del lado cliente mediante ASP.NET Web API. Las aplicaciones de una sola página proporcionan una experiencia de usuario atractiva que incluye interacciones del lado cliente mediante HTML5, CSS3 y JavaScript.

Autenticación

Sin autenticación

[Cambiar](#)

Agregar carpetas y referencias principales

☐ Formularios Web Forms

☒ MVC

☐ API web

Avanzado

☐ Configurar para HTTPS

☐ Compatibilidad con Docker
(Requiere [Docker Desktop](#))

☐ Crear también un proyecto para pruebas unitarias

PrimeraAppMVC.Tests

Atrás

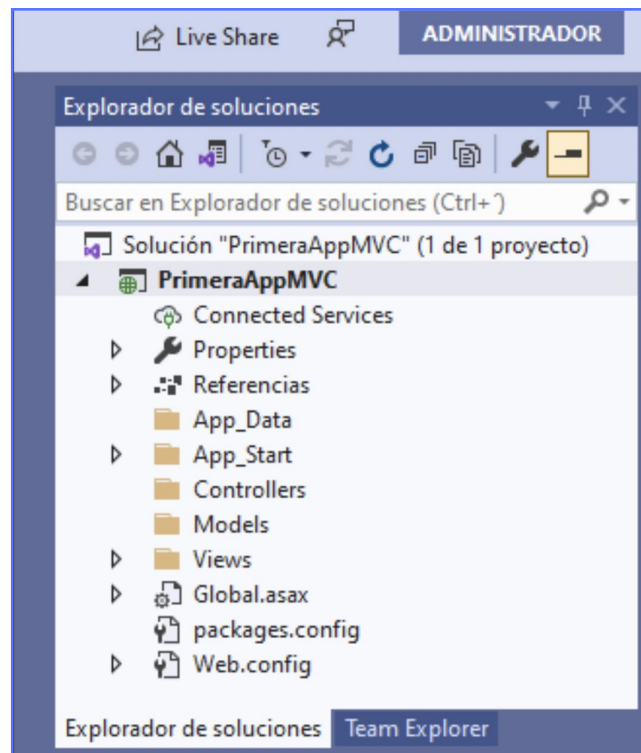
Crear

Activar Windows
Ve a Configuración para

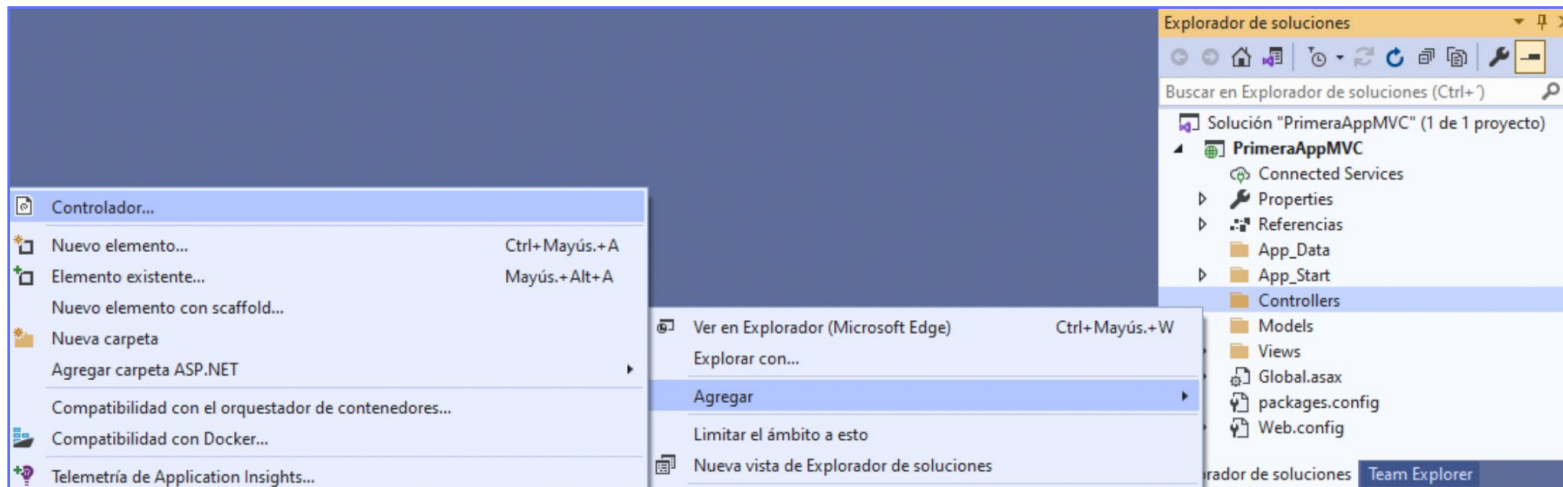
Como se puede observar, la estructura inicial de la aplicación no es funcional, eso quiere decir que no veremos nada si ejecutamos la aplicación.

Si observamos en la ventana del **Explorador de Soluciones** la carpeta **Controllers** está vacía. Como mencionamos antes, por lo menos debería haber un controlador para que nuestra aplicación pueda responder peticiones de un navegador.

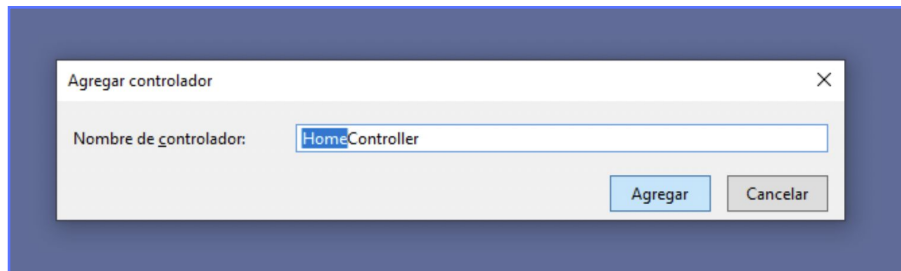
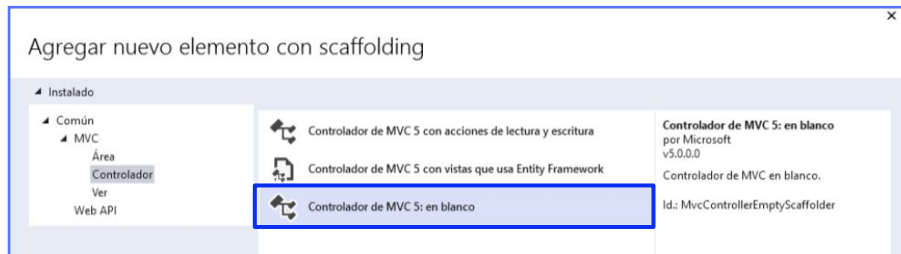
Por convención, en ASP.NET MVC el controlador "Home" es el que se ejecuta cuando se accede a un sitio web sólo con el nombre de dominio (por ej.: www.educacionit.com).



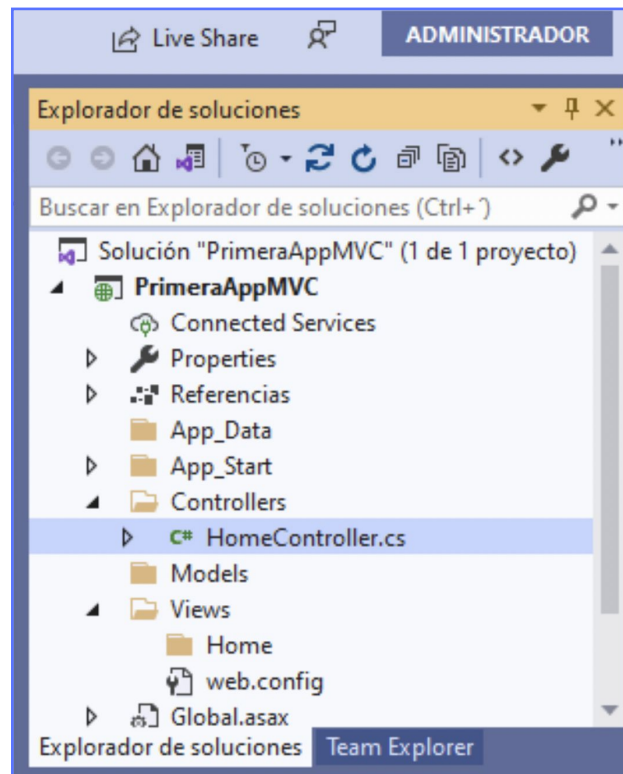
2. Por esa razón, nuestra primera tarea será crear el controlador "Home" haciendo clic con el botón derecho sobre la carpeta **Controllers** y seleccionando **Agregar > Controlador**.



3. En la ventana siguiente selecciona **Controlador de MVC 5: en blanco**.
4. En la próxima ventana, le asignaremos el nombre **HomeController** (por convención en ASP.NET MVC todos los controladores terminan con la palabra **Controller**):

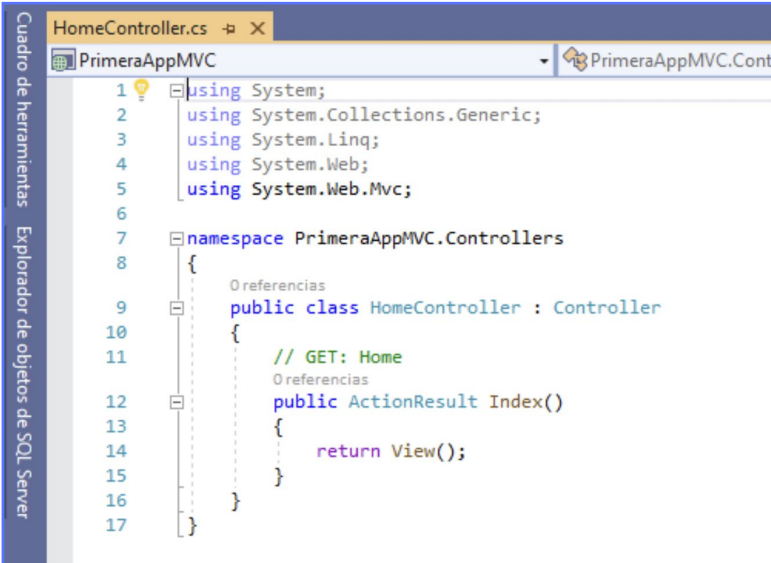


Si ahora observamos el **Explorador de soluciones** podremos comprobar que en la carpeta **Controllers** se agregó una clase llamada **HomeController.cs**:



También se creó una carpeta llamada **Home** en la carpeta **Views** pero, por ahora, no se analizará el contenido de esta carpeta creada ya que se resolverá toda la lógica dentro del Controlador.

Analicemos el código de la clase cuyo nombre es **HomeController.cs**:

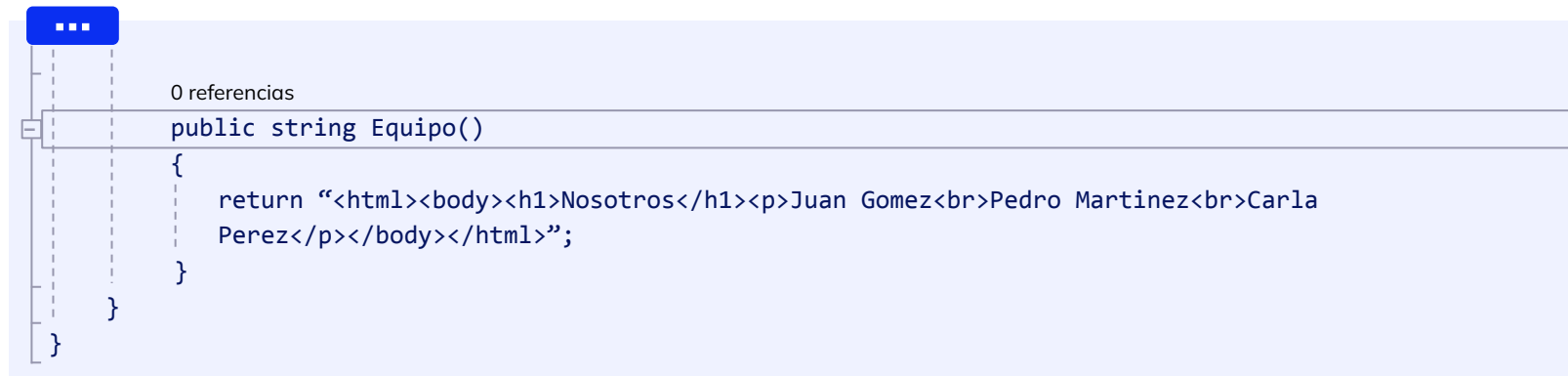


```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6
7 namespace PrimeraAppMVC.Controllers
8 {
9     // GET: Home
10     public class HomeController : Controller
11     {
12         // GET: Home
13         public ActionResult Index()
14         {
15             return View();
16         }
17     }
18 }
```



5. Se modificará el código generado automáticamente de esta clase con el siguiente contenido:

```
namespace PrimeraAppMVC.Controllers
{
    0 referencias
    public class HomeController : Controller
    {
        // GET: Home
        0 referencias
        public string Index()
        {
            return "<html><body><h1>ACME S.R.L.</h1><p>Una organización a su servicio, para resolver cualquier problema</p></body></html>";
        }
    }
}
```

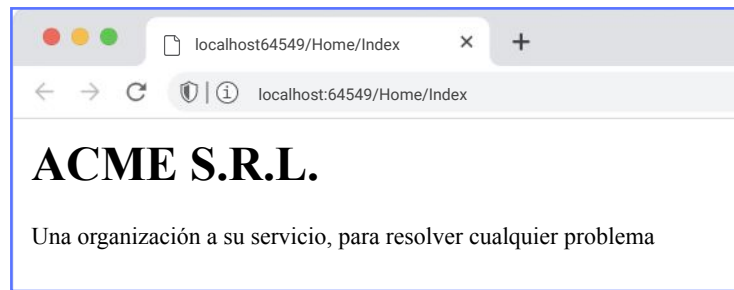


Antes de analizar su funcionamiento, vamos a ejecutar la página y veamos el resultado.

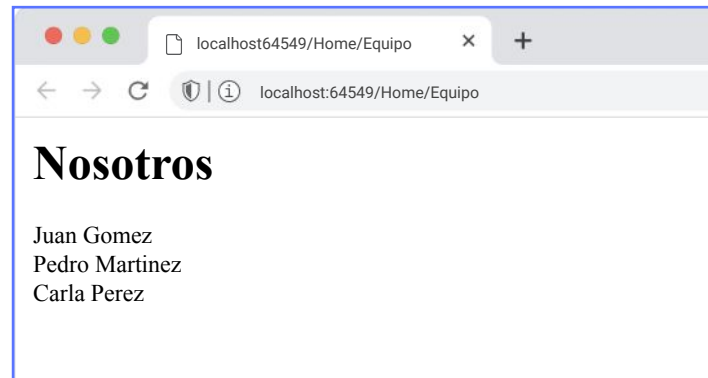
- a. Si se indica el nombre del dominio directamente: **http://localhost:puerto**



- b. Se obtendrá el mismo resultado si se realizan las peticiones que vemos a continuación:



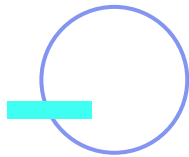
- c. Si, ahora, se realiza la petición siguiente:
`http://localhost:puerto/Home/Equipo`,
se obtiene como resultado lo que vemos:



Analizamos cómo ASP.NET MVC resuelve las peticiones HTTP hechas desde el navegador:

1. La clase **HomeController** hereda de la clase **Controller** y dentro de ella se definen una serie de métodos que responden a distintas URL's.
2. El método **Index** se ejecuta cuando hacemos la petición solamente con el nombre del dominio: **http://localhost:puerto**
3. Luego, el método **Equipo** se ejecuta en el momento en que realiza la solicitud: **http://localhost:puerto/Home/Equipo**

Como se mencionó, el controlador es una o varias clases que administran la comunicación entre las peticiones que hace el cliente, el modelo que procesa los datos que llegan del cliente y comunica a las vistas para que muestren los datos pedidos por el cliente. Para simplificar nuestro primer acercamiento a esta tecnología, resolvimos toda la lógica en el **Controlador**.



Básicamente el método **Index()** retorna un **string** con el contenido del HTML directamente:

```
public string Index()
{
    return "<html><body><h1>ACME S.R.L.</h1><p>Una organización a su servicio, para resolver cualquier problema</p></body></html>";
}
```

Claramente, ésta **no parece ser una solución eficiente si se tiene que retornar una gran cantidad de contenido**. Pero para comprender la comunicación entre el navegador y nuestro *Controlador*, queda claro **cómo deberíamos definir los nombres de los métodos y llamarlos desde el navegador**.



**¡Sigamos
trabajando!**

