Razor Pages

Razor Pages es un modelo de programación centrado en páginas en el lado servidor para crear interfaces de usuario web con ASP.NET Core. Dicha integración aporta las siguientes ventajas:

* Instalación fácil para aplicaciones web dinámicas con HTML, CSS y C#.
* Archivos organizados por característica para facilitar el mantenimiento.
* Combina el marcado con código de C# del lado servidor mediante la sintaxis Razor.

En Razor Pages se usa [*Razor*](https://learn.microsoft.com/es-es/aspnet/core/mvc/views/razor) para insertar código basado en servidor en las páginas web. La sintaxis de Razor combina HTML y [C#](https://dotnet.microsoft.com/languages/csharp) para definir la lógica de representación dinámica.

La clase PageModel es una [clase parcial](https://learn.microsoft.com/es-es/dotnet/csharp/programming-guide/classes-and-structs/partial-classes-and-methods) generada automáticamente por la plantilla de proyecto de ASP.NET Core. Aplica la separación de intereses con una clase PageModel de C#.

ASP.NET Core también admite el [**patrón de Controlador de vista de modelo (MVC)**](https://learn.microsoft.com/es-es/aspnet/core/mvc/overview) para compilar aplicaciones web. Use MVC cuando prefiera una separación clara entre el modelo, la vista y el controlador. Tanto Razor Pages como MVC pueden coexistir dentro de la misma aplicación. MVC está fuera del ámbito de este módulo.

***Copiar repositorio de github a la pc***

Abra un shell de comandos y clone el proyecto desde GitHub mediante la línea de comandos:

git clone <https://github.com/MicrosoftDocs/mslearn-create-razor-pages-aspnet-core>

También podría hacerse online la edición, CodeSpace pero eso es pago.

* Observe la combinación de código HTML, sintaxis de Razor y C# en el archivo.
  + La sintaxis de Razor se indica mediante caracteres @.
  + El código de C# se incluye en bloques @{ }. Tome nota de las directivas en la parte superior del archivo:
  + La directiva @page especifica que este archivo es una página de Razor.
  + La directiva @model especifica el tipo de modelo de la página (en este caso, IndexModel, que se define en *Pages/Index.cshtml.cs*).
* Revise el bloque de código de C#.
  + El código establece el valor del elemento Title dentro del diccionario ViewData en "Página principal".
  + El diccionario ViewData se usa para pasar datos entre la página de Razor y la clase IndexModel.
  + En tiempo de ejecución, el valor Title se usa para establecer el elemento <title> de la página.

**Creación de la página Lista de pizzas**

Para crear una nueva página de Razor, usará la CLI de .NET.

1. Como el comando bloquea el comando dotnet watch, abra otro comando haciendo clic con el botón derecho en la carpeta *ContosoPizza* en el **Explorador** y seleccione **Abrir en terminal integrado**.
2. Escriba el siguiente comando en la ventana del terminal:

CLI de .NETCopiar

dotnet new page --name PizzaList --namespace ContosoPizza.Pages --output Pages

El comando anterior:

* + Crea los archivos siguientes en el espacio de nombres ContosoPizza.Pages:
    - *PizzaList.cshtml*: la página de Razor
    - *PizzaList.cshtml.cs*: la clase PageModel que la acompaña
  + Almacena ambos archivos en el directorio *Pages* del proyecto.

Copié y me devolvió eso:

dotnet new page --name PizzaList --namespace ContosoPizza.Pages --output Pages

La plantilla "Página de Razor" se creó correctamente.

@page

@model ContosoPizza.Pages.PizzaListModel

@{

    ViewData["Title"] = "Pizza List 🍕";

   <h1>Pizza List 🍕</h1>

<!-- New Pizza form will go here -->

<!-- List of pizzas will go here -->

}

**Adición de la página Lista de pizzas al menú de navegación**

Este sería un buen momento para probar la página, pero aún no puede hacerlo porque la página no está vinculada al menú de navegación. Lo haremos ahora después.

1. Abra *Pages/Shared/\_Layout.cshtml*.
2. En el elemento <ul> con la clase navbar-nav (comienza en la línea 21), observe los elementos <li> que contienen los vínculos a las páginas *Inicio* y *Privacidad*. Agregue el código siguiente al final de la lista, después del elemento <li> que contiene el vínculo *Privacidad*:

razorCopiar

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-page="/PizzaList">Pizza List 🍕</a>

</li>

Esto agrega un vínculo a la página *Lista de pizzas* al menú de navegación.

**Registro de la clase PizzaService con el contenedor de inserción de dependencias**

La página Lista de pizzas depende del objeto PizzaService para recuperar la lista de pizzas. Usará la inserción de dependencias para proporcionar el objeto PizzaService a la página. Para que esto suceda, debe registrar la clase PizzaService con el contenedor.

1. Abra *Program.cs*.
2. En la sección que agrega servicios al contenedor, agregue el código siguiente:

C#Copiar

builder.Services.AddScoped<PizzaService>();

Este código registra la clase PizzaService con el contenedor de inserción de dependencias. El método AddScoped indica que se debe crear un nuevo objeto PizzaService para cada solicitud HTTP. Ahora PizzaService se puede insertar en cualquier página de Razor.

Dentro del bloque de espacio de nombres ContosoPizza.Pages, reemplace toda la clase PizzaListModel por el código siguiente:

C#Copiar

public class PizzaListModel : PageModel

{

private readonly PizzaService \_service;

public IList<Pizza> PizzaList { get;set; } = default!;

public PizzaListModel(PizzaService service)

{

\_service = service;

}

public void OnGet()

{

PizzaList = \_service.GetPizzas();

}

}

En el código anterior:

* Se crea un objeto PizzaService privado de solo lectura llamado \_service. Esta variable contendrá una referencia a un objeto PizzaService.
  + La palabra clave readonly indica que el valor de la variable \_service no se puede cambiar después de establecerlo en el constructor.
* Define una propiedad PizzaList que contendrá la lista de pizzas.
  + El tipo IList<Pizza> indica que la propiedad PizzaList contendrá una lista de objetos Pizza.
  + PizzaList se inicializa en default! para indicar al compilador que se inicializará más adelante, por lo que no se requieren comprobaciones de seguridad nulas.
* El constructor acepta un objeto PizzaService.
  + La inserción de dependencias proporciona el objeto PizzaService.
* Define un método OnGet que recupera la lista de pizzas del objeto PizzaService y la almacena en la propiedad PizzaList.