Parte 3. Adición de una vista a una aplicación de ASP.NET Core MVC

17/09/2021Tiempo de lectura: 9 minutos  

En este artículo

Por Rick Anderson

En esta sección, modificará la clase `HelloWorldController` para usar archivos de vista de Razor. Esto encapsula correctamente el proceso de generar respuestas HTML a un cliente.

Las plantillas de vista se crean mediante Razor. Las plantillas de vista basadas en Razor:

- Tienen una extensión de archivo *.cshtml*.
- Ofrecen una forma elegante de crear un resultado HTML con C#.

Actualmente, el método `Index` devuelve una cadena con un mensaje en la clase de controlador. En la clase `HelloWorldController`, reemplace el método `Index` por el siguiente código:

C# Copiar

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

El código anterior:

- Llama al método View del controlador.
- Usa una plantilla de vista para generar una respuesta HTML.

Los métodos de controlador:

- Reciben el nombre de *métodos de acción*. Por ejemplo, el método de acción `Index` del código anterior.
- Normalmente devuelven un elemento IActionResult o una clase derivada de ActionResult, no un tipo como `string`.

Agregar una vista

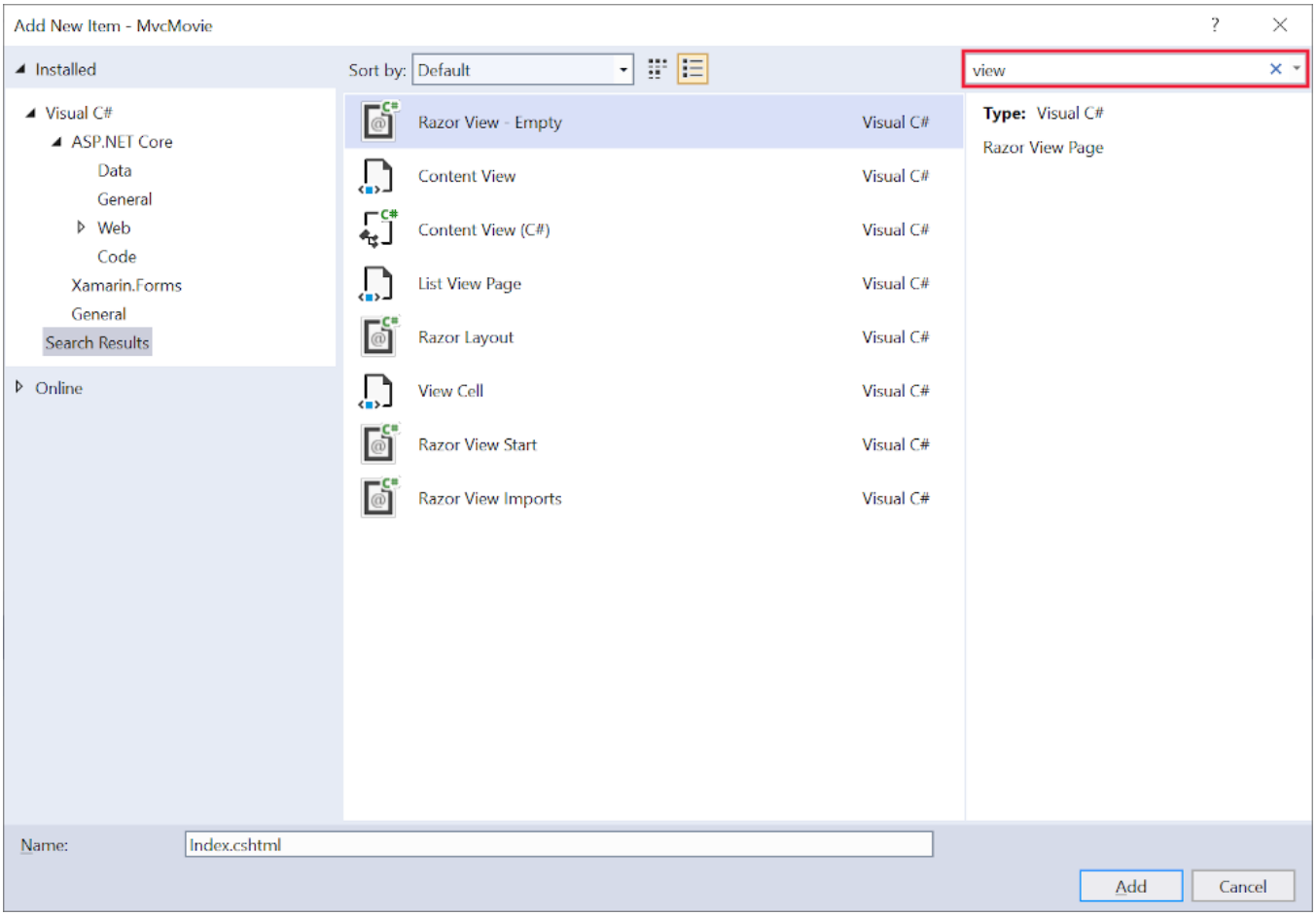
Visual Studio | Visual Studio Code | Visual Studio para Mac

Haga clic con el botón derecho en la carpeta *Views*, haga clic en **Agregar** > **Nueva carpeta** y asigne a la carpeta el nombre *HelloWorld*.

Haga clic con el botón derecho en la carpeta *Views/HelloWorld* y, luego, haga clic en **Agregar** > **Nuevo elemento**.

En el cuadro de diálogo **Add New Item - MvcMovie** (Agregar nuevo elemento: MvcMovie):

- En el cuadro de búsqueda situado en la esquina superior derecha, escriba *Vista*.
- Seleccione **Razor Vista - Vacía**.
- Conserve el valor del cuadro **Nombre**, *Index.cshtml*.
- Seleccione **Agregar**.



Reemplace el contenido del archivo de vista de Razor *Views/HelloWorld/Index.cshtml*por el siguiente:

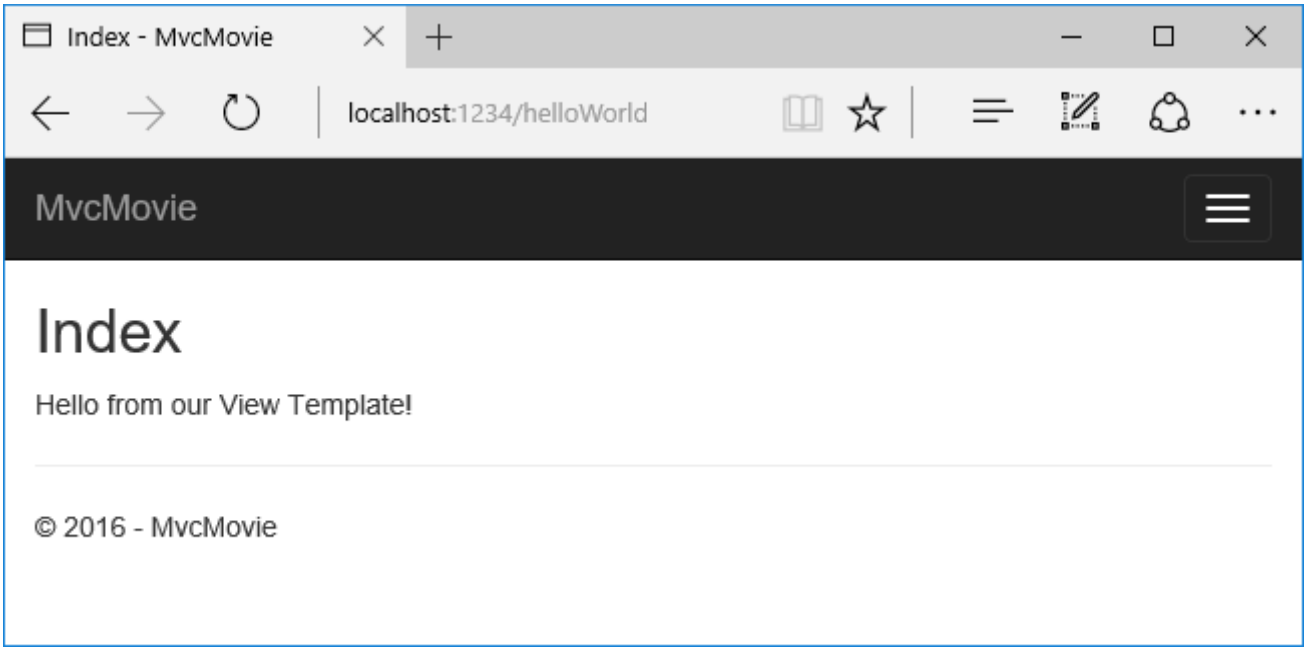
CSHTML

Copiar

```
@{
    ViewData["Title"] = "Index";
}

<h2>Index</h2>

<p>Hello from our View Template!</p>
```

Vaya a `https://localhost:{PORT}/HelloWorld`:

- El método `Index` en `HelloWorldController` ejecutó la instrucción `return View();`, que especificaba que el método debe usar un archivo de plantilla de vista para representar una respuesta al explorador.
- No se especificó un nombre de archivo de plantilla de vista, por lo que MVC usa el archivo de vista predeterminado. Cuando no se especifica el nombre del archivo de vista, se devuelve la vista predeterminada. En este ejemplo, la vista predeterminada tiene el mismo nombre que el método de acción (`Index`). Se usa la plantilla de vista */Views/HelloWorld/Index.cshtml*.
- En la imagen siguiente se muestra la cadena "Hello from our View Template!" (Hola desde nuestra plantilla de vista) codificada de forma rígida en la vista:



Cambio de vistas y páginas de diseño

Seleccione los vínculos de menú **MvcMovie**, **Home** y **Privacy** . Cada página muestra el mismo diseño de menú. El diseño de menú se implementa en el archivo *Views/Shared/_Layout.cshtml*.

Abra el archivo *Views/Shared/_Layout.cshtml*.

Las plantillas de diseño permiten:

- Especificar el diseño del contenedor HTML de un sitio en un solo lugar
- Aplicar el diseño del contenedor HTML en varias páginas del sitio

Busque la línea `@RenderBody()`. `RenderBody` es un marcador de posición donde se mostrarán todas las páginas específicas de vista que cree, *encapsuladas* en la página de diseño. Por ejemplo, si selecciona el vínculo **Privacy** , la vista **Views/Home/Privacy.cshtml** se representa dentro del método `RenderBody`.

Cambio de los vínculos del título, el pie de página y el menú en el archivo de diseño

Reemplace el contenido del archivo *Views/Shared/_Layout.cshtml* por el marcado siguiente. Se resaltan los cambios:

CSHTML

Copiar

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Movie App</title>

    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white
border-bottom box-shadow mb-3">
            <div class="container">
                <a class="navbar-brand" asp-controller="Movies" asp-action="Index">Movie
App</a>

                <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-
between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Privacy">Privacy</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
```

```
            &copy; 2020 - Movie App - <a asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>

        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

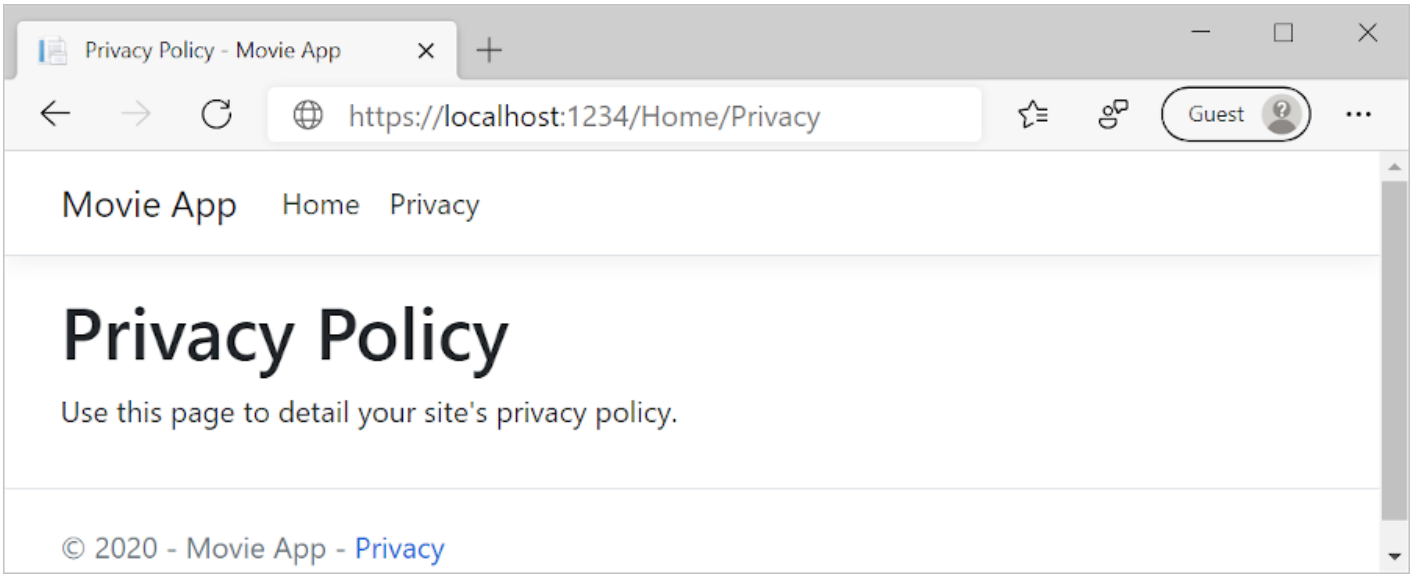
En el marcado anterior se realizan los cambios siguientes:

- Tres apariciones de `MvcMovie` a `Movie App`.
- El delimitador `<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">MvcMovie</a>` a `<a class="navbar-brand" asp-controller="Movies" asp-action="Index">Movie App</a>`.

En el marcado anterior, se omitieron el atributo del asistente de etiquetas delimitadorasy el valor de atributo de `asp-area=""` porque esta aplicación no usa Áreas.

Nota: El controlador `Movies` no se ha implementado. En este momento, el vínculo `Movie App` no es funcional.

Guarde los cambios y seleccione el vínculo **Privacy** . Observe cómo el título de la pestaña del explorador muestra **Privacy Policy - Movie App** en vez de **Privacy Policy - Mvc Movie**:



Seleccione el vínculo **Home**.

Observe que en el texto del título y del delimitador aparece **Movie App** (Aplicación de película). Los cambios se realizaron una vez en la plantilla de diseño, y todas las páginas del sitio reflejan el nuevo texto de vínculo y el nuevo título.

Examine el archivo *Views/_ViewStart.cshtml*:

CSHTML

Copiar

```
@{
    Layout = "_Layout";
}
```

El archivo *Views/_ViewStart.cshtml* trae el archivo *Views/Shared/_Layout.cshtml* a cada vista. Se puede usar la propiedad `Layout` para establecer una vista de diseño diferente o establecerla en `null` para que no se use ningún archivo de diseño.

Abra el archivo de vista *Views/HelloWorld/Index.cshtml*.

Cambie el título y el elemento `<h2>` como se indica a continuación:

CSHTML

Copiar

```
@{
    ViewData["Title"] = "Movie List";

}

<h2>My Movie List</h2>


<p>Hello from our View Template!</p>
```

El título y el elemento `<h2>` son algo diferentes para que quede claro qué parte del código cambia la presentación.

En el código anterior, `ViewData["Title"] = "Movie List";` establece la propiedad `Title` del diccionario `ViewData` en "Movie List" (Lista de películas). La propiedad `Title` se usa en el elemento HTML `<title>` en la página de diseño:

CSHTML

Copiar

```
<title>@ViewData["Title"] - Movie App</title>
```

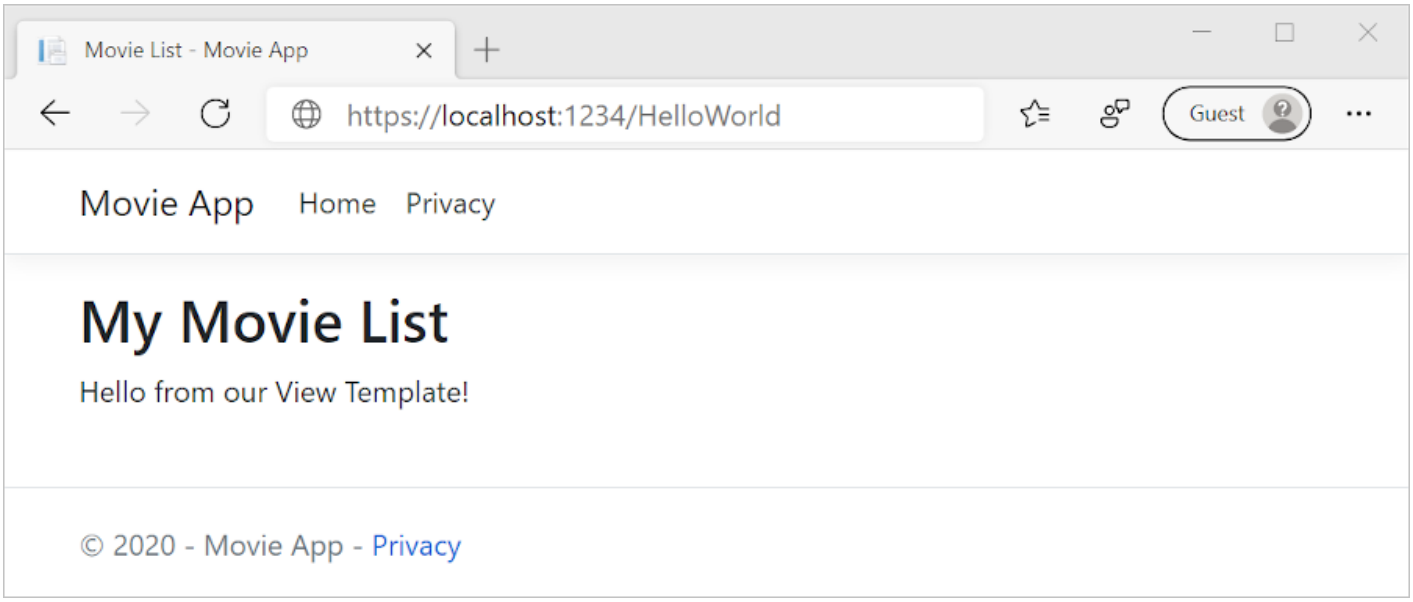
Guarde el cambio y navegue a `https://localhost:{PORT}/HelloWorld`.

Tenga en cuenta que los siguientes elementos han cambiado:

- Título del explorador
- Encabezado principal
- Encabezados secundarios

Si no hay ningún cambio en el explorador, podría estar viéndose contenido almacenado en caché. Presione CTRL+F5 en el explorador para forzar que se cargue la respuesta del servidor. El título del explorador se crea con `ViewData["Title"]`, que se definió en la plantilla de vista *Index.cshtml* y el texto "- Movie App" (-Aplicación de película) que se agregó en el archivo de diseño.

El contenido de la plantilla de vista *Index.cshtml* se combina con la plantilla de vista *Views/Shared/_Layout.cshtml*. Se envía una única respuesta HTML al explorador. Con las plantillas de diseño es fácil hacer cambios que se apliquen en todas las páginas de una aplicación. Para obtener más información, vea Diseño.



La pequeña cantidad de "datos", en este caso, el mensaje "Hello from our View Template!" (Hola desde nuestra plantilla de vista), está codificada de forma rígida. La aplicación de MVC tiene una "V" (vista) y una "C" (controlador), pero todavía no tiene una "M" (modelo).

Pasar datos del controlador a la vista

Las acciones del controlador se invocan en respuesta a una solicitud de dirección URL entrante. Una clase de controlador es donde se escribe el código que controla las solicitudes entrantes del explorador. El controlador recupera datos de un origen de datos y decide qué tipo de respuesta devolverá al explorador. Las plantillas de vista se pueden usar desde un controlador para generar y dar formato a una respuesta HTML al explorador.

Los controladores se encargan de proporcionar los datos necesarios para que una plantilla de vista represente una respuesta.

Las plantillas de vista **no** deben:

- Hacer lógica de negocios
- Interactuar con una base de datos directamente

Una plantilla de vista debe funcionar solo con los datos que le proporciona el controlador. Mantener esta "separación de intereses" ayuda a mantener el código:

- Limpio
- Fácil de probar
- Fácil de mantener

Actualmente, el método `Welcome` de la clase `HelloWorldController` toma un parámetro `name` y `ID`, y luego obtiene los valores directamente en el explorador.

En lugar de que el controlador represente esta respuesta como una cadena, cambie el controlador para que use una plantilla de vista. La plantilla de vista genera una respuesta dinámica, lo que significa que se deben pasar los datos adecuados del controlador a la vista para que se genere la respuesta. Para hacerlo, indique al controlador que coloque los datos dinámicos (parámetros) que necesita la plantilla de vista en un diccionario `ViewData`. Después, la plantilla de vista podrá acceder a los datos dinámicos.

En *HelloWorldController.cs*, cambie el método `Welcome` para agregar un valor `Message` y `NumTimes` al diccionario `ViewData`.

El diccionario `ViewData` es un objeto dinámico, lo que significa que se puede usar cualquier tipo. El objeto `ViewData` no tiene ninguna propiedad definida hasta que se agrega algo. El sistema de enlace de modelos de MVC asigna automáticamente los parámetros con nombre `name` y `numTimes` de la cadena de consulta a los parámetros del método. `HelloWorldController` completo:

C# Copiar

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

namespace MvcMovie.Controllers
{
    public class HelloWorldController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Welcome(string name, int numTimes = 1)
        {
            ViewData["Message"] = "Hello " + name;
            ViewData["NumTimes"] = numTimes;

            return View();
        }

    }
}
```

El objeto de diccionario `ViewData` contiene datos que se pasarán a la vista.

Cree una plantilla de vista principal denominada *Views/HelloWorld/Welcome.cshtml*.

Se creará un bucle en la vista *Welcome.cshtml* que muestra "Hello" (Hola) `NumTimes`. Reemplace el contenido de *Views/HelloWorld/Welcome.cshtml* con lo siguiente:

CSHTML Copiar

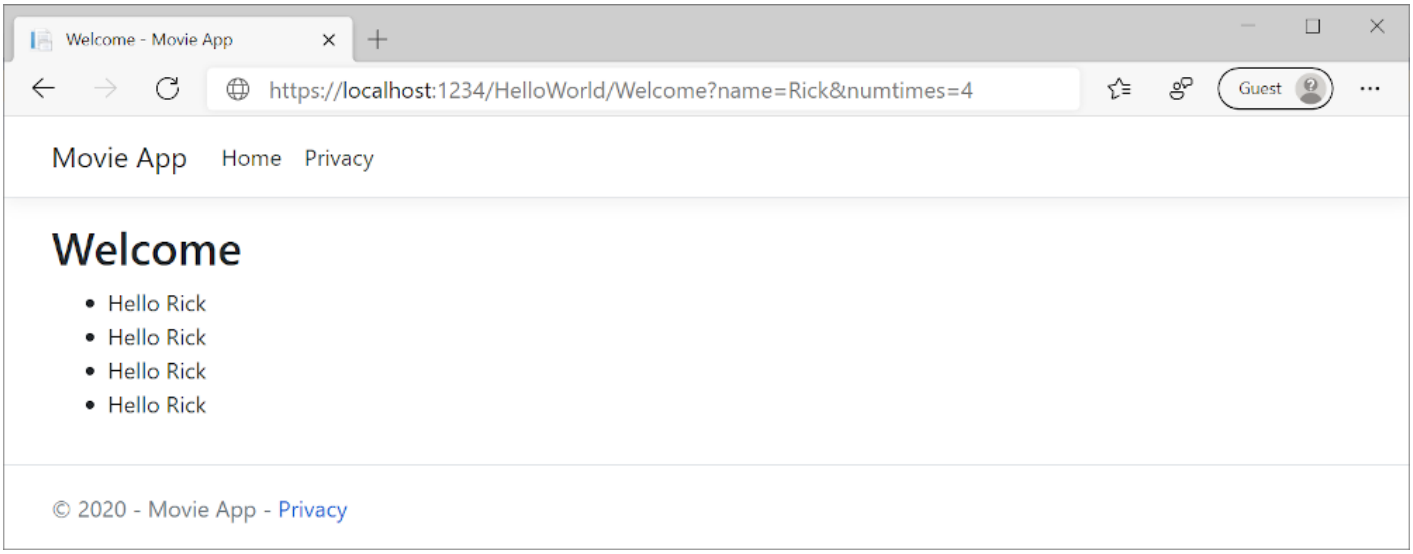
```
@{
    ViewData["Title"] = "Welcome";
```

```
}

<h2>Welcome</h2>

<ul>
    @for (int i = 0; i < (int)ViewData["NumTimes"]; i++)
    {
        <li>@ViewData["Message"]</li>
    }
</ul>
```

Guarde los cambios y vaya a esta dirección URL:

```
https://localhost:{PORT}/HelloWorld/Welcome?name=Rick&numtimes=4
```

Los datos se toman de la dirección URL y se pasan al controlador mediante el enlazador de modelos de MVC. El controlador empaqueta los datos en un diccionario `ViewData` y pasa ese objeto a la vista. Después, la vista representa los datos como HTML en el explorador.



En el ejemplo anterior, se usó el diccionario `ViewData` para pasar datos del controlador a una vista. Más adelante en el tutorial usaremos un modelo de vista para pasar datos de un controlador a una vista. El enfoque del modelo de vista que consiste en pasar datos es preferible al enfoque de diccionario `ViewData`.

En el tutorial siguiente crearemos una base de datos de películas.