Controllers e Actions

Faculdade UNIDESC

Programação Para Web II

Profº Ruben Prado

Controllers e Actions

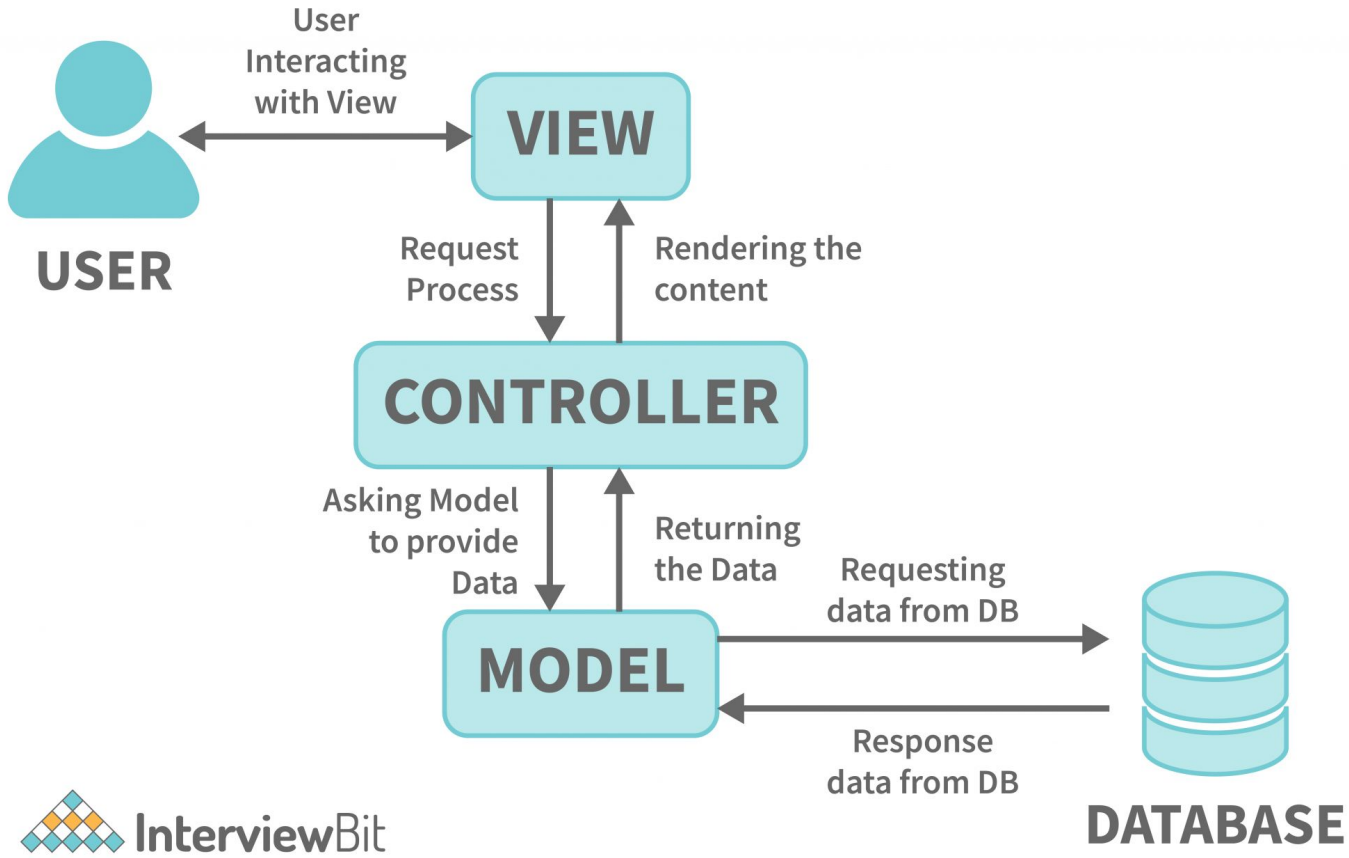
Controllers:

- São classes que recebem as requisições do usuário e contêm a lógica para processá-las.
- No padrão MVC, os controllers atuam como intermediários entre o Model (dados) e a View (interface), decidindo qual ação executar com base na requisição.

Actions:

- São métodos públicos dentro de um Controller que respondem a requisições específicas.
- Cada action retorna um resultado (geralmente um IActionResult), que pode ser uma view, redirecionamento ou dados em formato JSON, por exemplo.

MVC


User
Interacting
with View
VIEW
USER
Request
Process
Rendering the
content
CONTROLLER
Asking Model
to provide
Data
Returning
the Data
Requesting
data from DB
MODEL
Response
data from DB
DATABASE
InterviewBit

Integração no Padrão MVC

01

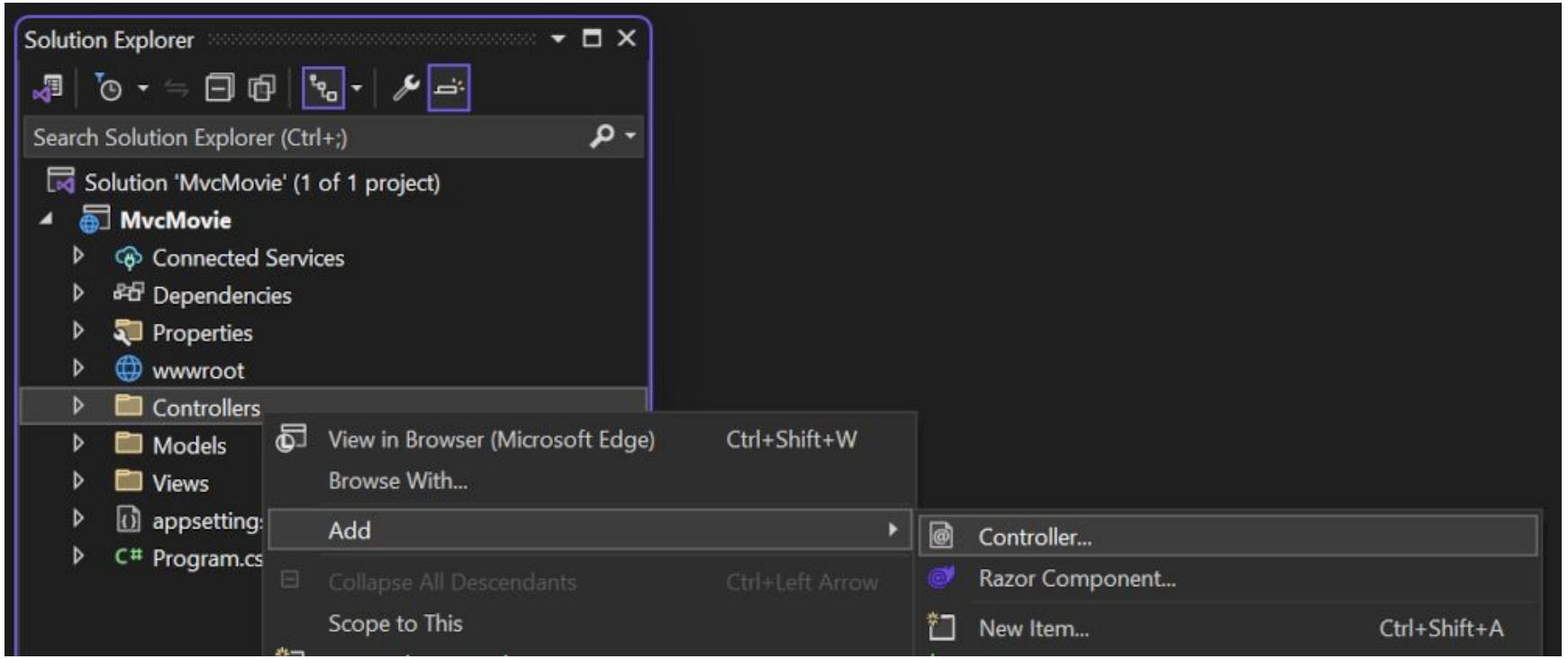
Model: Responsável pelos dados e regras de negócio.

02

View: Responsável pela apresentação.

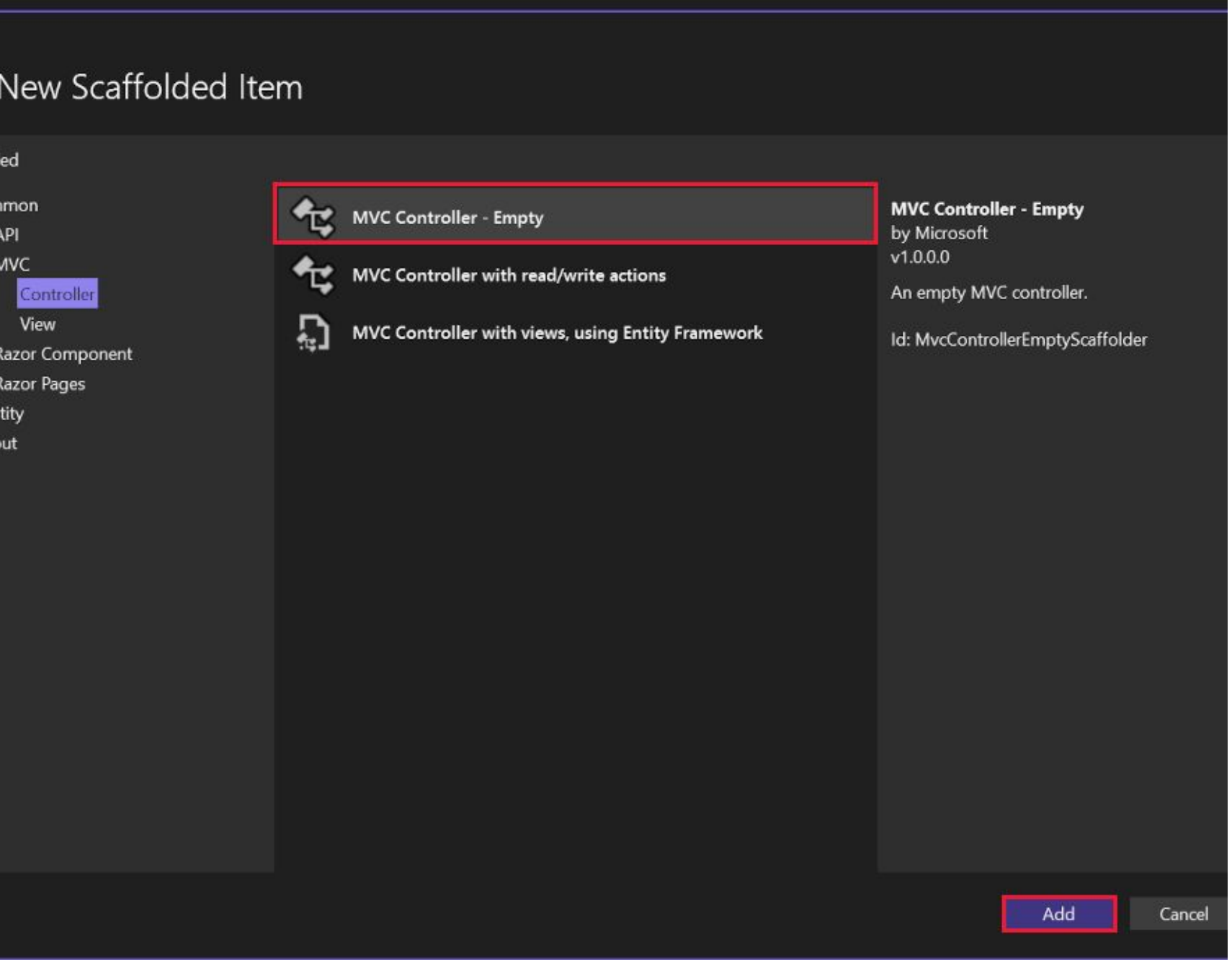
03

Controller: Coordena a interação entre o Model e a View, manipulando as requisições e definindo a resposta.

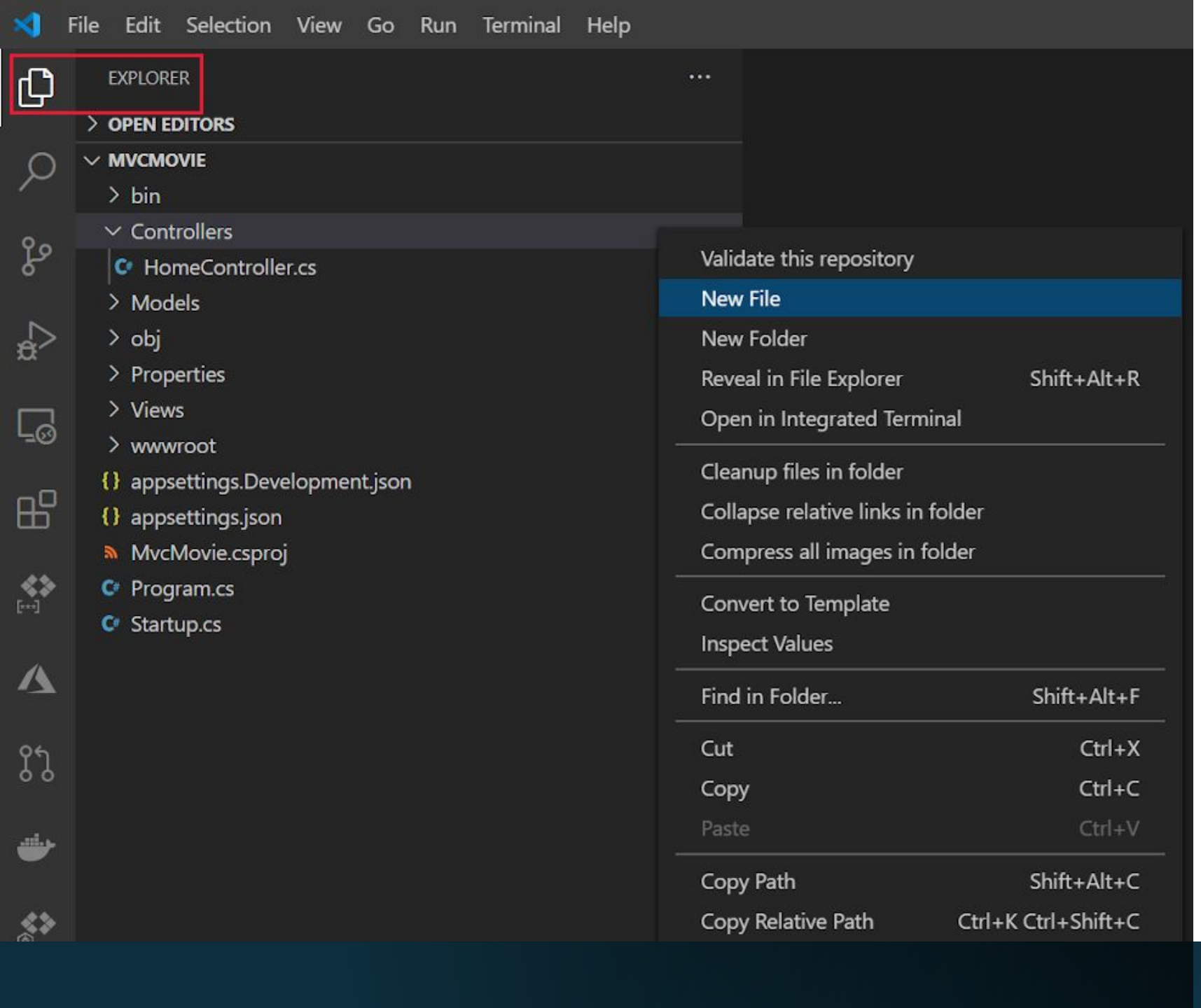


Como criar uma nova Controller

- No Gerenciador de Soluções, clique com o botão direito do mouse em Controladores > Adicionar > Controlador.



- Na caixa de diálogo Adicionar Novo Item com Scaffolding, selecione Controlador MVC – Vazio>Adicionar.
- Na caixa de diálogo Adicionar Novo Item – MvcMovie, insira HelloWorldController.cs e selecione Adicionar.



Como criar uma nova Controller

- Selecione o ícone EXPLORER e, em seguida, pressione Control (clique com o botão direito do mouse) Controladores > Novo Arquivo e nomeie o novo arquivo HelloWorldController.cs.

Como criar uma nova Controller

Cada método public em um controlador pode ser chamado como um ponto de extremidade HTTP. Na amostra acima, ambos os métodos retornam uma cadeia de caracteres. Observe os comentários que precedem cada método.

- Substitua o conteúdo de Controllers/HelloWorldController.cs pelo seguinte código:

```
using Microsoft.AspNetCore.Mvc;
using System.Text.Encodings.Web;
namespace MvcMovie.Controllers;
public class HelloWorldController : Controller
{
        // // GET: /HelloWorld/
        public string Index()
        {
            return "This is my default action...";
        }
        // // GET: /HelloWorld/Welcome/
        public string Welcome()
        {
            return "This is the Welcome action method...";
        }
}
```

- Cada método public em um controlador pode ser chamado como um ponto de extremidade HTTP. Na amostra acima, ambos os métodos retornam uma cadeia de caracteres.

Observe os comentários que precedem cada método.

Um ponto de extremidade HTTP:

- É uma URL direcionável no aplicativo Web, como https://localhost:5001/HelloWorld.
- Combina:
  - O protocolo usado: HTTPS.
  - O local de rede do servidor Web, incluindo a porta TCP: localhost:5001.
  - O URI de destino: HelloWorld.

- O primeiro comentário indica que este é um método HTTP GET invocado por meio do acréscimo de /HelloWorld/ à URL base.

- O primeiro comentário especifica um método HTTP GET invocado por meio do acréscimo de /HelloWorld/Welcome/ à URL base.

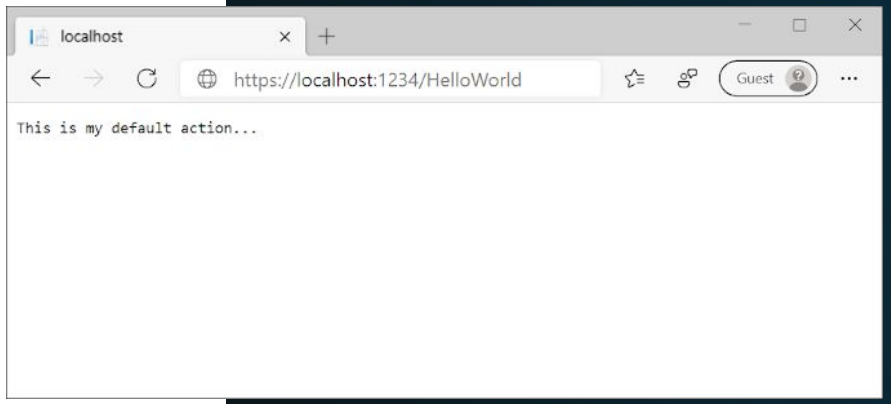
- Execute o aplicativo sem o depurador pressionando Ctrl+F5.

- Acrescente /HelloWorld ao caminho na barra de endereços. O método Index retorna uma cadeia de caracteres.

- O MVC invoca as classes do controlador e os métodos de ação dentro delas, dependendo da URL de entrada. A lógica de roteamento de URL padrão usada pelo MVC usa um formato como este para determinar o código a ser invocado:

- /[Controller]/[ActionName]/[Parameters]

- O formato de roteamento é definido no arquivo Program.cs .



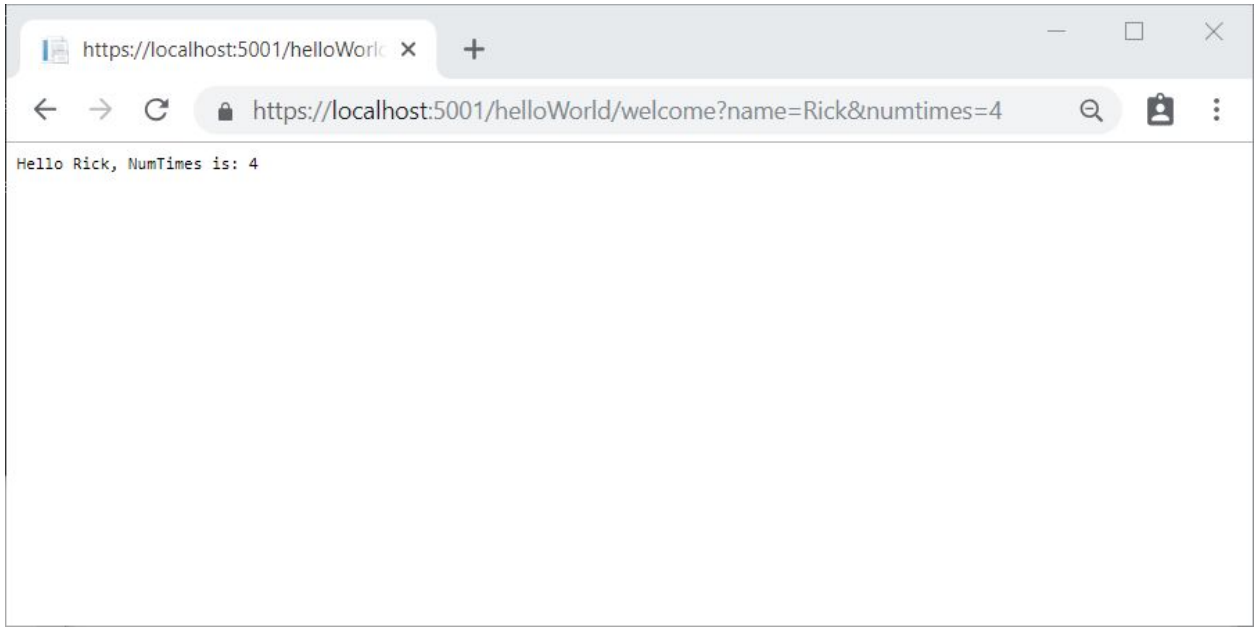
- Quando você acessa o aplicativo e não fornece nenhum segmento de URL, ele usa como padrão o controlador “Home” e o método “Index” especificado na linha do modelo realçada acima. Nos segmentos de URL anteriores:
- O primeiro segmento de URL determina a classe do controlador a ser executada. Portanto, o localhost:5001/HelloWorld mapeia para a classe HelloWorldController.
- A segunda parte do segmento de URL determina o método de ação na classe. Portanto, localhost:5001/HelloWorld/Index faz com que o método Index da classe HelloWorldController seja executado. Observe que você precisou apenas navegar para localhost:5001/HelloWorld e o método Index foi chamado por padrão. Index é o método padrão que será chamado em um controlador se não houver um nome de método explicitamente especificado.
- A terceira parte do segmento de URL (id) refere-se aos dados de rota.

- Modifique o código para passar algumas informações de parâmetro da URL para o controlador. Por exemplo, /HelloWorld/Welcome?name=Rick&numtimes=4.
- Altere o método Welcome para incluir dois parâmetros, conforme mostrado no código a seguir.

```
// GET: /HelloWorld/Welcome/
// Requires using System.Text.Encodings.Web;
public string Welcome(string name, int numTimes = 1)
{
    return HtmlEncoder.Default.Encode($"Hello {name}, NumTimes is: {numTimes}");
}
```

- O código anterior:
- Usa o recurso de parâmetro opcional do C# para indicar que o parâmetro numTimes usa 1 como padrão se nenhum valor é passado para esse parâmetro.
- Usa HtmlEncoder.Default.Encode para proteger o aplicativo contra entrada mal-intencionada, como através de JavaScript.
- Usa Cadeias de caracteres interpoladas em $"Hello {name}, NumTimes is: {numTimes}".

- Execute o aplicativo e navegue até: https://localhost:{PORT}/HelloWorld/Welcome?name=Rick&numtimes=4. Substitua {PORT} pelo número da porta.
- Experimente valores diferentes para name e numtimes na URL. O sistema de model binding do MVC mapeia automaticamente os parâmetros nomeados da cadeia de consulta para os parâmetros no método.



- O segmento de URL Parameters não é usado.
- Os parâmetros name e numTimes são passados na cadeia de caracteres de consulta.
- O ? (ponto de interrogação) na URL acima é um separador seguido pela cadeia de consulta.
- O caractere & separa os pares campo-valor.
- Substitua o método Welcome pelo seguinte código:

```
public string Welcome(string name, int ID = 1)
{
    return HtmlEncoder.Default.Encode($"Hello {name}, ID: {ID}");
}
```

- Execute o aplicativo e insira a seguinte URL: https://localhost:{PORT}/HelloWorld/Welcome?name=Rick

- Na URL anterior:

- O terceiro segmento de URL correspondeu ao parâmetro de rota id.
- O método Welcome contém um parâmetro id que correspondeu ao modelo de URL no método MapControllerRoute.
- O ? à direita inicia a cadeia de caracteres de consulta.

Como criar uma Action

- Os modelos de exibição são criados usando Razor. Modelos de exibição baseados em Razor:

- Tenha uma extensão de arquivo .cshtml.
- Forneça uma maneira elegante de criar a saída HTML com o C#.
- Atualmente, o método Index retorna uma cadeia de caracteres com uma mensagem na classe do controlador. Na classe HelloWorldController, substitua o método Index pelo seguinte código:

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

Como criar uma Action

- O código anterior:
  - Chama o método View do controlador.
  - Usa um modelo de exibição para gerar uma resposta HTML.

- Métodos do controlador:
- São chamados de métodos de ação. Por exemplo, o método de ação Index no código anterior.
- Geralmente, retorna IActionResult ou uma classe ou derivada de ActionResult, não um tipo como string.

No Visual Studio

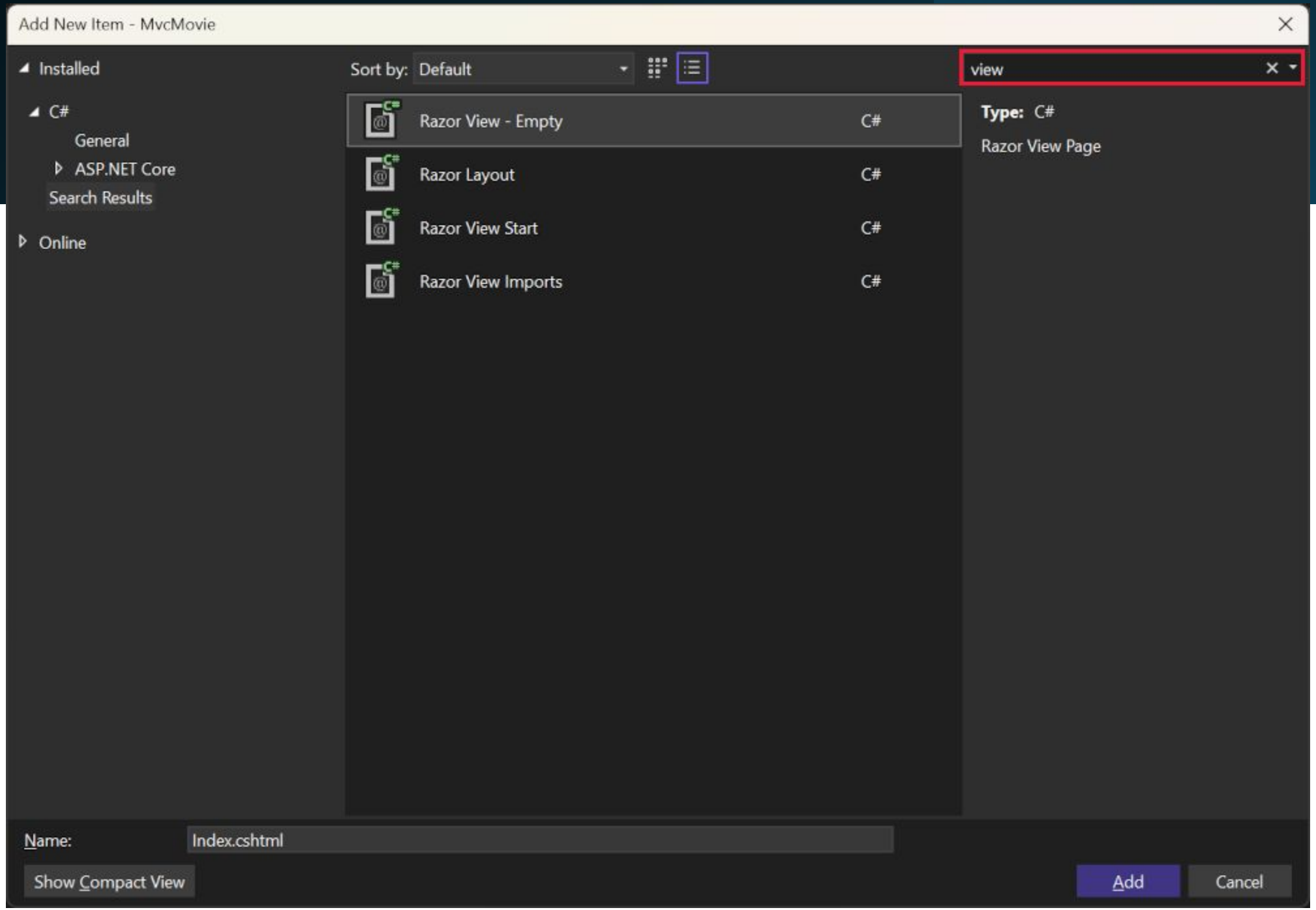
- Clique com o botão direito do mouse na pasta Exibições e, em seguida, Adicionar >Nova Pasta e nomeie a pasta HelloWorld.

- Clique com o botão direito do mouse na pasta Views/HelloWorld e, em seguida, clique em Adicionar > Novo Item.

- Na caixa de diálogo Adicionar Novo Item, selecione Mostrar Todos os Modelos.

- Na caixa de diálogo Adicionar Novo Item – NomeDoProjeto:

- Na caixa de pesquisa no canto superior direito, insira exibição
- Selecione Razor Exibição - Vazio
- Mantenha o valor da caixa Nome, Index.cshtml.
- Selecione Adicionar


Add New Item - MvcMovie
Installed
C#
General
ASP.NET Core
Search Results
Online
Sort by: Default
view
Razor View - Empty C#
Razor Layout C#
Razor View Start C#
Razor View Imports C#
Type: C#
Razor View Page
Name: Index.cshtml
Show Compact View
Add
Cancel

No Visual Studio Code

- Adicione uma exibição Index ao HelloWorldController:
- Adicione uma nova pasta chamada Views/HelloWorld.
- Adicione um novo arquivo à pasta Views/HelloWorld e dê a ele o nome Index.cshtml.

Como criar uma Action

- Substitua o conteúdo do arquivo de exibição Views/HelloWorld/Index.cshtmlRazor pelo seguinte:

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

<h2>Index</h2>

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

Como criar uma Action

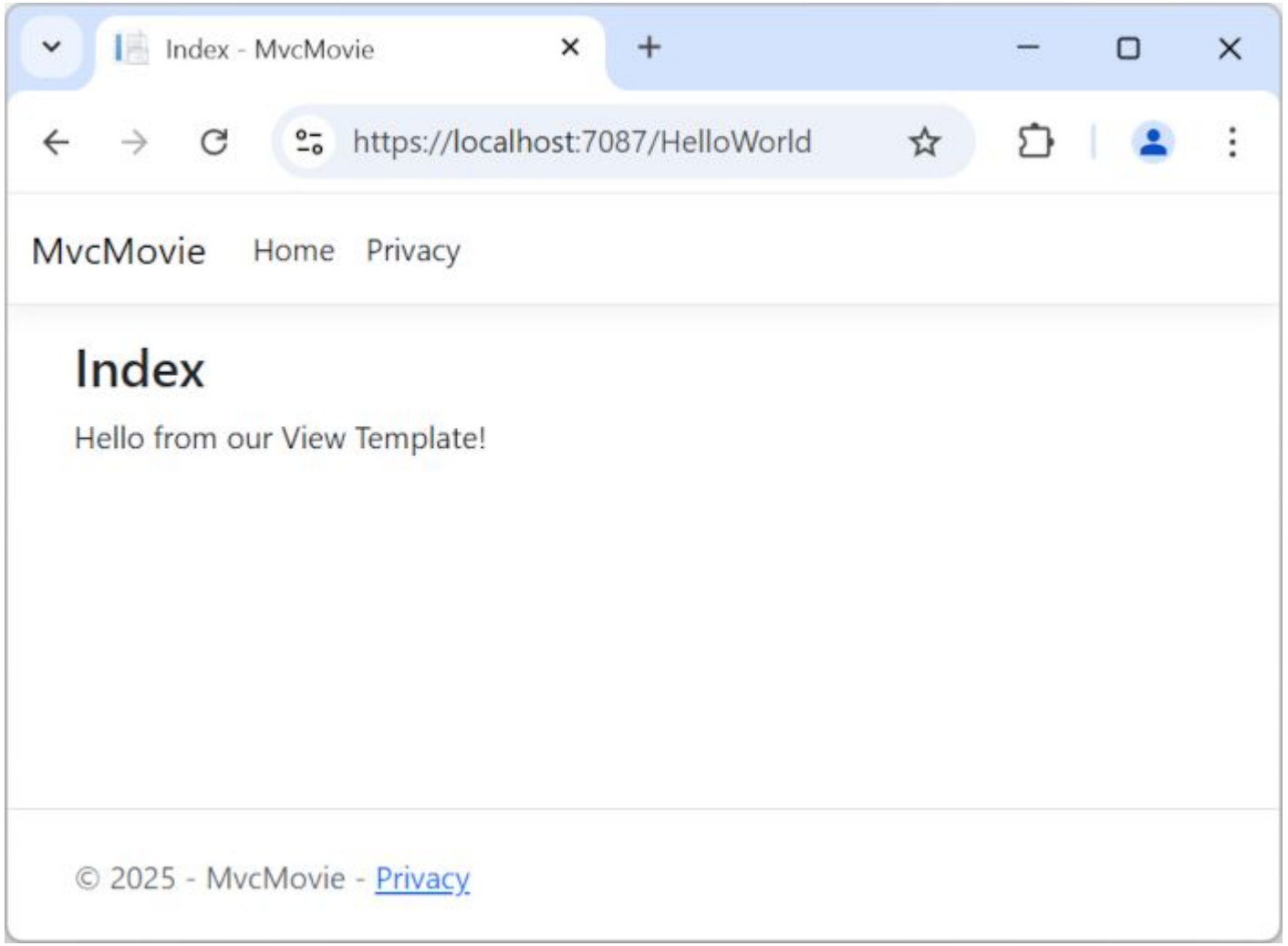
- Acesse o diretório https://localhost:{PORT}/HelloWorld:

- O método Index no HelloWorldController executou a instrução return View();, que especificou que o método deve usar um arquivo de modelo de exibição para renderizar uma resposta para o navegador.

- Como o nome do arquivo de modo de exibição não foi especificado, o MVC é padronizado para usar o arquivo de exibição padrão. Quando o nome do arquivo de exibição não é especificado, a exibição padrão é retornada. A exibição padrão tem o mesmo nome que o método de ação, Index neste exemplo. O modelo de exibição /Views/HelloWorld/Index.cshtml é usado.

Como criar uma Action

- A imagem a seguir mostra a cadeia de caracteres "Olá aqui do nosso Modelo de Exibição!" nativa do código do modo de exibição:



Alterar exibições e páginas de layout

- Selecione os links de menu do projeto padrão, Home e Privacy. Cada página mostra o mesmo layout de menu. O layout do menu é implementado no arquivo Views/Shared/_Layout.cshtml.

- Abra o arquivo Views/Shared/_Layout.cshtml .

- Os modelos de Layout permitem:

- Especificar o layout do contêiner HTML de um site em um só lugar.
- Aplicar o layout do contêiner HTML em várias páginas no site.
- Localize a linha @RenderBody(). RenderBody é um espaço reservado em que todas as páginas específicas à exibição criadas são mostradas, encapsuladas na página de layout. Por exemplo, se você selecionar o link Privacy, a exibição Views/Home/Privacy.cshtml será renderizada dentro do método RenderBody.

Alterar exibições e páginas de layout

- Alterar o título, o rodapé e o link de menu no arquivo de layout
- Substitua o conteúdo do arquivo Views/Shared/_Layout.cshtml pela seguinte marcação.
- Neste ponto é interessante realizar ajustes para melhor adequação do layout ao projeto que estamos realizando.
- A propriedade Layout pode ser usada para definir outra exibição de layout ou defina-a como null para que nenhum arquivo de layout seja usado.

- Abra o arquivo de exibição Views/HelloWorld/Index.cshtml.

- Altere o título e o elemento <h2> conforme realçado a seguir:

Alterar exibições e páginas de layout

- @{
- ViewData["Title"] = “Lista de Lanches";
- }
- <h2>Lanches Disponíveis</h2>

- View dos Lanches!

- ViewData["Title"] = “Lista de Lanches"; no código acima define a propriedade Title do dicionário ViewData como “Lista de Filmes”.
- A propriedade Title é usada no elemento HTML <title> na página de layout:

Passando dados do Controller para a View

- As ações do controlador são invocadas em resposta a uma solicitação de URL de entrada. Uma classe de controlador é o local em que o código é escrito e que manipula as solicitações recebidas do navegador. O controlador recupera dados de uma fonte de dados e decide qual tipo de resposta será enviada novamente para o navegador. Modelos de exibição podem ser usados em um controlador para gerar e formatar uma resposta HTML para o navegador.

- Os controladores são responsáveis por fornecer os dados necessários para que um modelo de exibição renderize uma resposta.

Passando dados do Controller para a View

- Os modelos de exibição não devem:
  - Processar lógicas de negócios
  - Interagir diretamente com algum banco de dados.
- O modelo de exibição deve funcionar somente com os dados fornecidos pelo controlador. Manter essa “separação de preocupações” ajuda a manter o código:
  - Limpo.
  - Testável.
  - Descomplicado em relação à manutenção.

Passando dados do Controller para a View

- Atualmente, o método Welcome na classe HelloWorldController usa um name e um parâmetro IDe, em seguida, gera os valores de saída diretamente no navegador.
- Em vez de fazer com que o controlador renderize a resposta como uma cadeia de caracteres, altere o controlador para que ele use um modelo de exibição. O modelo de exibição gera uma resposta dinâmica, o que significa que é necessário passar os dados apropriados do controlador para a exibição para gerar a resposta. Faça isso fazendo com que o controlador coloque os dados dinâmicos (parâmetros) que o modelo de exibição precisa em um dicionário ViewData. O modelo de exibição pode então acessar os dados dinâmicos.

Passando dados do Controller para a View

- Em HelloWorldController.cs, altere o método Welcome para adicionar um valor Message e NumTimes ao dicionário ViewData.
- O dicionário ViewData é um objeto dinâmico, o que significa que qualquer tipo pode ser usado. O objeto ViewData não tem propriedades definidas até que algo seja adicionado. O sistema de model binding do MVC mapeia automaticamente os parâmetros nomeados name e numTimes da cadeia de caracteres de consulta para os parâmetros do método.

Passando dados do Controller para a View

```
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();
    }
```

Passando dados do Controller para a View

- O objeto de dicionário ViewData contém dados que serão passados para a exibição.
- Criar um modelo de exibição de boas-vindas chamado Views/HelloWorld/Welcome.cshtml.
- Você criará um loop no modelo de exibição Welcome.cshtml que exibe “Olá” NumTimes. Substitua o conteúdo de Views/HelloWorld/Welcome.cshtml pelo fornecido a seguir:

Passando dados do Controller para a View

```
@{
    ViewData["Title"] = "Welcome";
}
<h2>Welcome</h2>
<ul>
    @for (int i = 0; i < (int)ViewData["NumTimes"]!; i++)
    {
        <li>@ViewData["Message"]</li>
    }
</ul>
```

Passando dados do Controller para a View

Salve as alterações e navegue para a seguinte URL:

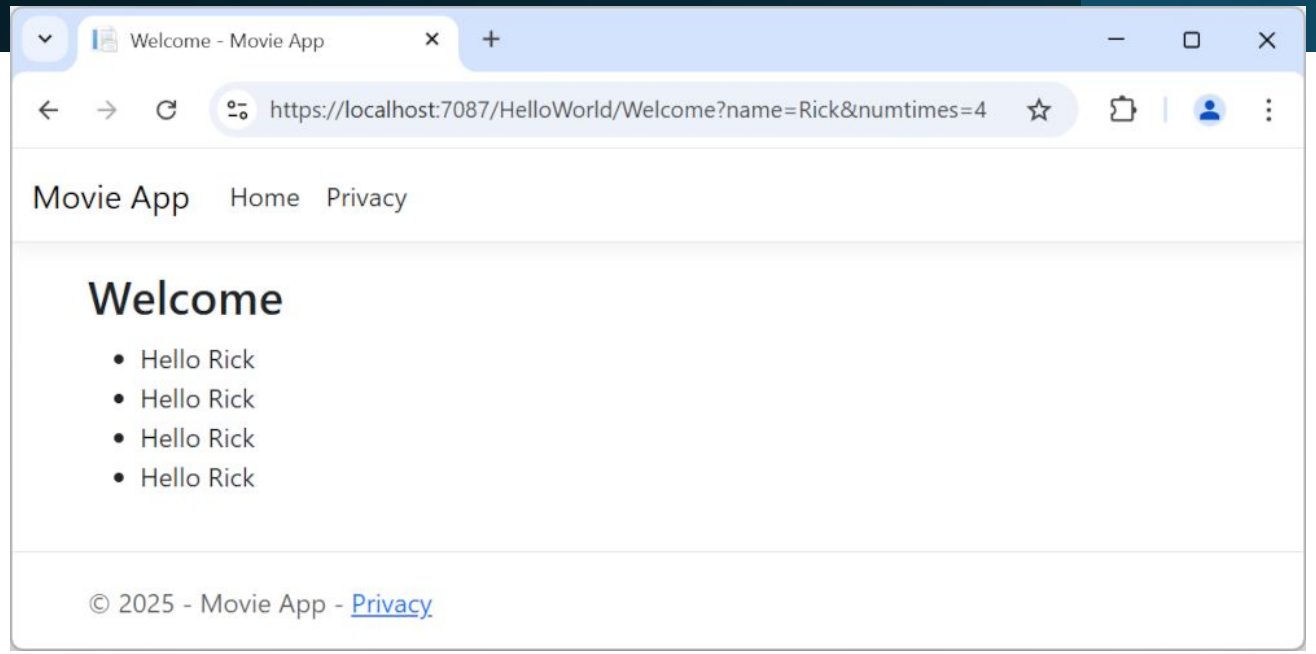
https://localhost:{PORT}/HelloWorld/Welcome?name=Rick&numtimes=4

Os dados são obtidos da URL e passados para o controlador usando o associador de modelo MVC.

O controlador empacota os dados em um dicionário ViewData e passa esse objeto para a exibição.

Em seguida, a exibição renderiza os dados como HTML para o navegador.

Passando dados do Controller para a View



No exemplo acima, o dicionário ViewData foi usado para passar dados do controlador para uma exibição. Mais adiante no tutorial, um modelo de exibição será usado para passar dados de um controlador para uma exibição. A abordagem do modelo de exibição para passar dados é geralmente a preferida em relação à abordagem do dicionário ViewData.