**Exercício – Adicionar um controlador**

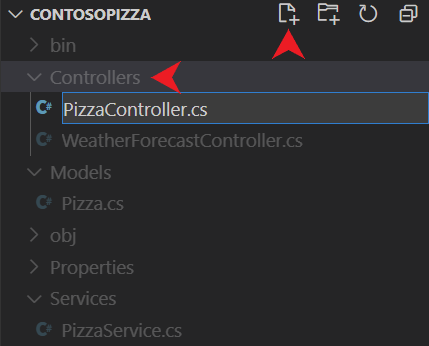
Concluído100 XP

* 5 minutos

Um *controlador* é uma classe pública com um ou mais métodos públicos conhecidos como *ações*. Por convenção, um controlador é colocado no diretório *Controllers* da raiz do projeto. As ações são expostas como pontos de extremidade HTTP no controlador da API Web.

**Criar um controlador**

1. Selecione a pasta *Controladores* no Visual Studio Code e adicione um novo arquivo chamado *PizzaController. cs*.



Um arquivo de classe vazio chamado *PizzaController.cs* é criado no diretório *Controladores*. O nome do diretório *Controladores* é uma convenção. O nome do diretório vem da arquitetura model-view-*controller* que a API Web usa.

**Observação**

Por convenção, os nomes de classe do controlador apresentam *Controlador* como sufixo.

1. Adicione o código a seguir a *Controllers/PizzaController.cs*. Salve suas alterações.

C#Copiar

using ContosoPizza.Models;

using ContosoPizza.Services;

using Microsoft.AspNetCore.Mvc;

namespace ContosoPizza.Controllers;

[ApiController]

[Route("[controller]")]

public class PizzaController : ControllerBase

{

public PizzaController()

{

}

// GET all action

// GET by Id action

// POST action

// PUT action

// DELETE action

}

Conforme você aprendeu anteriormente, essa classe deriva de ControllerBase, a classe base para trabalhar com solicitações HTTP no ASP.NET Core. Ela também inclui os dois atributos padrão que você aprendeu, [ApiController] e [Route]. Assim como antes, o atributo [Route] define um mapeamento para o token [controller]. Como essa classe de controlador é nomeada PizzaController , esse controlador manipula as solicitações para https://localhost:{PORT}/pizza .

**Obter todas as pizzas**

O primeiro verbo REST que precisamos implementar é GET, em que um cliente pode obter todas as pizzas da API. Você pode usar o atributo [HttpGet] interno para definir um método que retornará as pizzas de nosso serviço.

Substitua o comentário // GET all action em *Controllers/PizzaController.cs* pelo seguinte código:

C#Copiar

[HttpGet]

public ActionResult<List<Pizza>> GetAll() =>

PizzaService.GetAll();

A ação anterior:

* Responde apenas ao verbo HTTP GET, conforme indicado pelo atributo [HttpGet].
* Consulta o serviço para todas as pizzas e retorna automaticamente os dados com um valor Content-Type igual a application/json.

**Recuperar apenas uma pizza**

O cliente também pode querer solicitar informações sobre uma pizza específica em vez de toda a lista. Você pode implementar outra ação GET que requer um parâmetro id. Você pode usar o atributo [HttpGet("{id}")] interno para definir um método que retornará as pizzas de nosso serviço. A lógica de roteamento registra [HttpGet] (sem id ) e [HttpGet("{id}")] (com id ) como duas rotas diferentes. Você pode então criar uma ação separada para recuperar um único item.

Substitua o comentário // GET by Id action em *Controllers/PizzaController.cs* pelo seguinte código:

C#Copiar

[HttpGet("{id}")]

public ActionResult<Pizza> Get(int id)

{

var pizza = PizzaService.Get(id);

if(pizza == null)

return NotFound();

return pizza;

}

A ação anterior:

* Responde apenas ao verbo HTTP GET, conforme indicado pelo atributo [HttpGet].
* Requer que o valor do parâmetro id seja incluído no segmento da URL após pizza/. Lembre-se, o atributo [Route] do nível do controlador definiu o padrão /pizza.
* Consulta o banco de dados para uma pizza que corresponde ao parâmetro id fornecido.

Cada instância de ActionResult usada na ação anterior é mapeada para o código de status HTTP correspondente na tabela a seguir:

| **ASP.NET Core resultado da ação** | **Código de status HTTP** | **Descrição** |
| --- | --- | --- |
| Ok está implícito | 200 | Um produto que corresponde ao parâmetro id fornecido existe no cache na memória. O produto está incluído no corpo da resposta no tipo de mídia, conforme definido no cabeçalho da solicitação HTTP accept (JSON por padrão). |
| NotFound | 404 | Um produto que corresponde ao parâmetro id fornecido não existe no cache na memória. |

**Compilar e testar o controlador**

1. Compile e inicie a API Web executando o seguinte comando:

CLI do .NETCopiar

dotnet run

1. Abra o terminal httprepl existente ou um novo terminal integrado do Visual Studio Code selecionando **Terminal**>**Novo Terminal** no menu principal.
2. Conecte-se à API Web executando o seguinte comando:

CLI do .NETCopiar

httprepl https://localhost:{PORT}

Outra opção é executar o seguinte comando a qualquer momento enquanto HttpRepl estiver em execução:

CLI do .NETCopiar

(Disconnected)> connect https://localhost:{PORT}

1. Para ver o ponto de extremidade de Pizza recém-disponível, execute o seguinte comando:

CLI do .NETCopiar

ls

O comando anterior detecta todas as APIs disponíveis no ponto de extremidade conectado. Ele deve exibir o seguinte código:

CLI do .NETCopiar

https://localhost:{PORT}/> ls

. []

Pizza [GET]

WeatherForecast [GET]

1. Vá até o ponto de extremidade Pizza executando o seguinte comando:

CLI do .NETCopiar

cd Pizza

O comando anterior mostra uma saída das APIs disponíveis para o ponto de extremidade Pizza:

CLI do .NETCopiar

https://localhost:{PORT}/> cd Pizza

/Pizza [GET]

1. Faça uma solicitação GET no HttpRepl usando o seguinte comando:

CLI do .NETCopiar

get

O comando anterior retorna uma lista de todas as pizzas em json:

CLI do .NETCopiar

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Date: Fri, 02 Apr 2021 21:55:53 GMT

Server: Kestrel

Transfer-Encoding: chunked

[

{

"id": 1,

"name": "Classic Italian",

"isGlutenFree": false

},

{

"id": 2,

"name": "Veggie",

"isGlutenFree": true

}

]

1. Para consultar apenas uma pizza, você pode fazer outra solicitação GET, mas passar um parâmetro id usando o seguinte comando:

CLI do .NETCopiar

get 1

O comando anterior retorna Classic Italian com a seguinte saída:

CLI do .NETCopiar

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Date: Fri, 02 Apr 2021 21:57:57 GMT

Server: Kestrel

Transfer-Encoding: chunked

{

"id": 1,

"name": "Classic Italian",

"isGlutenFree": false

}

1. Nossa API também lida com situações em que o item não existe. Chame a API novamente, mas passe um parâmetro pizza id inválido usando o seguinte comando:

CLI do .NETCopiar

get 5

O comando anterior retorna um erro 404 Not Found com a seguinte saída:

CLI do .NETCopiar

HTTP/1.1 404 Not Found

Content-Type: application/problem+json; charset=utf-8

Date: Fri, 02 Apr 2021 22:03:06 GMT

Server: Kestrel

Transfer-Encoding: chunked

{

"type": "https://tools.ietf.org/html/rfc7231#section-6.5.4",

"title": "Not Found",

"status": 404,

"traceId": "00-ec263e401ec554b6a2f3e216a1d1fac5-4b40b8023d56762c-00"

}

1. Retorne ao terminal dotnet na lista suspensa no Visual Studio Code e desligue a API Web selecionando Ctrl+C no teclado.

Agora você terminou de implementar os verbos GET. Na próxima unidade, você pode adicionar mais ações a PizzaController para dar suporte a operações CRUD em dados de pizza.