04

Recuperando filmes por ID

Transcrição

Nesta aula, vamos aprimorar a busca por filmes no nosso sistema. De início, pensaremos em alguns casos peculiares que poderiam ser problemáticos na nossa API.

Digamos que um dos itens cadastrados é "Planeta dos Macacos", filme de 1968, do diretor Franklin J. Schaffner:

```
{
    "Titulo" : "Planeta dos Macacos",
    "Genero" : "Ação",
    "Duracao" : 112
}
```

COPIAR CÓDIGO

Além disso, também consta no nosso sistema o *remake* desse filme — "Planeta dos Macacos", de 2011, do diretor Tim Burton:

```
{
    "Titulo" : "Planeta dos Macacos",
    "Genero" : "Ação",
    "Duracao" : 119
}
```

COPIAR CÓDIGO

Ao buscar a listagem de filmes, teremos dois filmes com o mesmo nome, ainda que os tempos de durações sejam diferentes:

```
[
    {
        "titulo": "Avatar",
        "genero": "Ação",
        "duracao": 162
    },
    {
        "titulo": "Star Wars",
        "genero": "Aventura",
        "duracao": 100
    },
    {
        "titulo": "Planeta dos Macacos",
        "genero": "Ação",
        "duracao": 112
    },
    {
        "titulo": "Planeta dos Macacos",
        "genero": "Ação",
        "duracao": 119
    }
]
```

COPIAR CÓDIGO

Como podemos diferenciar esses filmes? Para esse caso em específico, uma opção seria adicionar um campo para data de lançamento ou nome dos diretores, mas eventualmente poderíamos nos deparar com outros cenários de conflitos semelhantes. O ideal, então, é ter um **identificador** para cada filme, um critério que garanta que todo filme seja **único** no sistema.

Uma maneira bem simples e tradicional é inserir o **atributo de identificador, o famoso ID**.

Adicionando o ID

No controlador, criaremos um campo do tipo `inteiro`, chamado `id`, cujo valor inicial é 0:

```
using FilmesApi.Models;
using Microsoft.AspNetCore.Mvc;

namespace FilmesApi.Controllers;

[ApiController]
[Route("[controller]")]
public class FilmeController : ControllerBase
{

    private static List<Filme> filmes = new List<Filme>();
    private static int id = 0;

    [HttpPost]
    public void AdicionaFilme([FromBody] Filme filme)
    {
        filmes.Add(filme);
        Console.WriteLine(filme.Titulo);
        Console.WriteLine(filme.Duracao);
    }

    [HttpGet]
    public IEnumerable<Filme> RecuperaFilmes()
    {
        return filmes;
    }
}
```

COPIAR CÓDIGO

Ao inserir um filme no sistema, vamos definir o ID do filme como `id++`. Assim, a cada novo filme, o ID será incrementado:

```
// ...

[HttpPost]
public void AdicionaFilme([FromBody] Filme filme)
{
        filme.Id = id++;
        filmes.Add(filme);
        Console.WriteLine(filme.Titulo);
        Console.WriteLine(filme.Duracao);
}

// ...
```

COPIAR CÓDIGO

Além disso, é preciso inserir a propriedade `Id` no nosso modelo. No método `AdicionaFilme()`, basta posicionar o cursor sobre `Id` (em `filme.Id`), pressionar "Alt + Enter" e selecionar "Gerar propriedade 'Id'".

Acessando o arquivo `Filme.cs`, a propriedade `Id` foi gerada abaixo da propriedade `Duracao`. Para facilitar a visualização desse campo, vamos colocá-lo antes da propriedade `Titulo`. Além disso, substituiremos o `internal set` por apenas `set`:

```
using System.ComponentModel.DataAnnotations;

namespace FilmesApi.Models;

public class Filme
{
    public int Id { get; set; }
    [Required(ErrorMessage = "O título do filme é obrigatório")]
    public string Titulo { get; set; }
    [Required(ErrorMessage = "O gênero do filme é obrigatório")]
    [MaxLength(50, ErrorMessage = "O tamanho do gênero não pode exce
```

```
    public string Genero { get; set; }
    [Required]
    [Range(70, 600, ErrorMessage = "A duração deve ter entre 70 e 60
    public int Duracao { get; set; }
}
```

COPIAR CÓDIGO

Não precisamos nos preocupar em colocar a anotação `[Required]`, pois é o nosso próprio sistema que insere esse ID, não um usuário.

Vamos salvar todas as alterações e reexecutar nossa aplicação. Ao reiniciá-la, perderemos a nossa listagem de filmes novamente, então vamos enviar alguns dados pelo Postman, mais uma vez:

```
{
    "Titulo" : "Planeta dos Macacos",
    "Genero" : "Ação",
    "Duracao" : 115
}
```

COPIAR CÓDIGO

```
{
    "Titulo" : "Planeta dos Macacos",
    "Genero" : "Ação",
    "Duracao" : 120
}
```

COPIAR CÓDIGO

```
{
    "Titulo" : "Star Wars",
    "Genero" : "Aventura",
```

```
    "Duracao" : 120
}
```

COPIAR CÓDIGO

Em seguida, vamos fazer uma requisição GET para verificar a listagem. Como resultado, notaremos que agora cada filme tem um ID:

```
[
    {
        "id": 0,
        "titulo": "Planeta dos Macacos",
        "genero": "Ação",
        "duracao": 115
    },
    {
        "id": 1,
        "titulo": "Planeta dos Macacos",
        "genero": "Ação",
        "duracao": 120
    },
    {
        "id": 2,
        "titulo": "Star Wars",
        "genero": "Aventura",
        "duracao": 120
    }
]
```

COPIAR CÓDIGO

Buscando por ID

Agora, conseguimos diferenciar nossos filmes pelo ID. O próximo passo é desenvolver uma maneira de **recuperar um elemento pelo ID**, independentemente do seu título, para ter um **retorno único**!

No controlador, já temos um POST para fazer a inserção de recursos e um GET para recuperar todos os filmes cadastrados no sistema. Que verbo utilizaremos agora para recuperar um filme só? Será que podemos repetir o GET?

Podemos, porém com algumas condições! Antes de nos aprofundar nessa questão, vamos desenvolver um **método de busca por ID**.

Ao final do controlador, vamos criar o método `RecuperaFilmePorId()`, inicialmente com retorno do tipo `void`. Assim como recebemos o filme por parâmetro no método `AdicionaFilmes()`, agora receberemos o ID:

```
// ...

public void RecuperaFilmePorId(int id)
{

}
```

COPIAR CÓDIGO

Em seguida, utilizaremos o LINQ para fazer uma solução bem elegante. Com o método `.FirstOfDefault()`, buscaremos o primeiro resultado cujo ID é igual ao recebido via parâmetro:

```
// ...

public void RecuperaFilmePorId(int id)
{
        return filmes.FirstOrDefault(filme => filme.Id == id);
}
```

COPIAR CÓDIGO

Assim, para cada elemento da lista `filmes`, verificaremos se seu ID é igual ao ID recebido por parâmetro. Se for igual, esse filme será retornado. Porém, se iterarmos

por toda a lista e não encontrarmos nenhum elemento que preencha esse requisito, retornaremos o valor *default* (padrão) — nesse caso, **nulo**.

Posicionando o cursor sobre a palavra `return` nesse método, pressionaremos "Alt + Enter" e selecionaremos "Corrigir tipo de retorno":

```
// ...

public Filme? RecuperaFilmePorId(int id)
{
        return filmes.FirstOrDefault(filme => filme.Id == id);
}
```

COPIAR CÓDIGO

Note que o Visual Studio indicou o tipo de retorno como `Filme?`, com um ponto de interrogação ao final, porque o `Filme` pode ser nulo. Sabemos que, caso não haja nenhum filme com o ID informado, o retorno será nulo, então vamos manter o ponto de interrogação. Se o removêssemos, estaríamos assumindo que o retorno nunca será nulo.

Por fim, faremos a indicação do verbo GET:

```
// ...

[HttpGet]
public Filme? RecuperaFilmePorId(int id)
{
        return filmes.FirstOrDefault(filme => filme.Id == id);
}
```

COPIAR CÓDIGO

Agora, tanto o método `RecuperaFilmes()` quando o método `RecuperaFilmePorId()` usam o verbo GET. Ao receber uma requisição GET em `/filmes`, como nosso sistema

saberá qual deles acionar? A diferença está no recebimento do parâmetro `id`!
Portanto, junto do verbo `HttpGet`, vamos informar que ele receberá o parâmetro `id`:

```
// ...

[HttpGet("{id}")]
public Filme? RecuperaFilmePorId(int id)
{
        return filmes.FirstOrDefault(filme => filme.Id == id);
}
```

COPIAR CÓDIGO

Quando passarmos um ID no GET, o sistema executará o `RecuperaFilmePorId()`. Do contrário, o método `RecuperaFilmes()` será acionado.

Sendo assim, será feito um *bind* automaticamente e o valor passado na requisição será passado como parâmetro ao método.

Testando

Vamos reiniciar nossa aplicação e popular nossa lista com dois filmes, usando o Postman para enviar requisições com os seguintes dados:

```
{
    "Titulo" : "Star Wars",
    "Genero" : "Aventura",
    "Duracao" : 120
}
```

COPIAR CÓDIGO

```
{
    "Titulo" : "Planeta dos Macacos",
    "Genero" : "Ação",
```

```
    "Duracao" : 120
}
```

COPIAR CÓDIGO

Em seguida, enviaremos uma requisição GET para verificar os itens cadastrados. Como resultado, temos a seguinte lista:

```
[
    {
        "id": 0,
        "titulo": "Star Wars",
        "genero": "Aventura",
        "duracao": 120
    },
    {
        "id": 1,
        "titulo": "Planeta dos Macacos",
        "genero": "Ação",
        "duracao": 120
    }
]
```

COPIAR CÓDIGO

Para recuperar o filme com ID igual a 1, basta realizar uma requisição GET com o parâmetro 1:

```
https://localhost:7106/filme/1
```

COPIAR CÓDIGO

Ao pressionar o botão "Send", o retorno no painel inferior do Postman será o filme "Planeta dos Macacos", cujo ID é 0:

```
{
    "id": 1,
    "titulo": "Planeta dos Macacos",
    "genero": "Ação",
    "duracao": 120
}
```

COPIAR CÓDIGO

Conseguimos adicionar um novo critério de busca! Agora, é possível pesquisar filmes a partir de seu ID e identificar os recursos do nosso sistema de maneira única!

No entanto, vamos assumir que eventualmente nossa lista pode ficar bastante extensa, por exemplo, com milhares de filmes. Não é interessante que o usuário fique restrito a recuperar apenas um filme pelo ID ou carregar todos os filmes de uma vez.