Como Criar um ProdutosController em um Ambiente Windows Usando ASP.NET Core

No desenvolvimento de aplicações web, especialmente em ambientes Windows, o ASP.NET Core é uma das tecnologias mais robustas e populares. Um dos componentes fundamentais de uma aplicação ASP.NET Core é o Controller, que gerencia a lógica de entrada e saída de dados. Neste artigo, vamos explorar como criar um ProdutosController para gerenciar operações CRUD (Create, Read, Update, Delete) em uma aplicação ASP.NET Core.

Configuração do Ambiente

Antes de começarmos, certifique-se de ter o .NET SDK instalado em seu sistema Windows. Você pode baixar a versão mais recente do SDK no site oficial da Microsoft.

1. **Verifique a instalação do .NET SDK:** Abra o Prompt de Comando (CMD) e execute:

   ```
   dotnet --version
   ```

2. **Crie um novo projeto ASP.NET Core:** No CMD, navegue até o diretório onde deseja criar o projeto e execute:

   ```
   dotnet new webapi -n ProdutosApi
   ```

3. **Navegue até o diretório do projeto:**

   ```
   cd ProdutosApi
   ```

Criando o ProdutosController

1. **Adicione um modelo de Produto:** No diretório Models, crie um arquivo chamado Produto.cs com o seguinte conteúdo:

   ```
   namespace ProdutosApi.Models
   {
      public class Produto
      {
          public int Id { get; set; }
          public string Nome { get; set; }
          public decimal Preco { get; set; }
      }
   }
   ```

2. **Adicione um contexto de dados:** No diretório Data, crie um arquivo chamado ProdutoContext.cs:

```
using Microsoft.EntityFrameworkCore;
using ProdutosApi.Models;

namespace ProdutosApi.Data
{
   public class ProdutoContext : DbContext
   {
       public ProdutoContext(DbContextOptions<ProdutoContext> options
) : base(options)
       {
       }

       public DbSet<Produto> Produtos { get; set; }
   }
}
```

3. **Configure o contexto no Startup.cs:** No arquivo Startup.cs, adicione as seguintes linhas no método ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)
{
   services.AddDbContext<ProdutoContext>(opt =>
      opt.UseInMemoryDatabase("ProdutosList"));
   services.AddControllers();
}
```

4. **Crie o ProdutosController:** No diretório Controllers, crie um arquivo chamado ProdutosController.cs:

```
using Microsoft.AspNetCore.Mvc;
using ProdutosApi.Models;
using ProdutosApi.Data;
using System.Collections.Generic;
using System.Linq;

namespace ProdutosApi.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class ProdutosController : ControllerBase
   {
       private readonly ProdutoContext _context;

       public ProdutosController(ProdutoContext context)
       {
           _context = context;

           if (_context.Produtos.Count() == 0)
           {
               // Create a new Produto if collection is empty,
```

```
                // which means you can't delete all Produtos.
                _context.Produtos.Add(new Produto { Nome = "Item1", Pr
eco = 10.00M });
                _context.SaveChanges();
            }
        }

        [HttpGet]
        public ActionResult<List<Produto>> GetAll() =>
            _context.Produtos.ToList();

        [HttpGet("{id}", Name = "GetProduto")]
        public ActionResult<Produto> GetById(int id)
        {
            var produto = _context.Produtos.Find(id);

            if (produto == null)
            {
                return NotFound();
            }

            return produto;
        }

        [HttpPost]
        public IActionResult Create(Produto produto)
        {
            _context.Produtos.Add(produto);
            _context.SaveChanges();

            return CreatedAtRoute("GetProduto", new { id = produto.Id
}, produto);
        }

        [HttpPut("{id}")]
        public IActionResult Update(int id, Produto produto)
        {
            var existingProduto = _context.Produtos.Find(id);

            if (existingProduto == null)
            {
                return NotFound();
            }

            existingProduto.Nome = produto.Nome;
            existingProduto.Preco = produto.Preco;

            _context.Produtos.Update(existingProduto);
            _context.SaveChanges();
```

```
            return NoContent();
        }

        [HttpDelete("{id}")]
        public IActionResult Delete(int id)
        {
            var produto = _context.Produtos.Find(id);

            if (produto == null)
            {
                return NotFound();
            }

            _context.Produtos.Remove(produto);
            _context.SaveChanges();

            return NoContent();
        }
    }
}
```

Executando a Aplicação

1. **Execute o projeto:** No CMD, dentro do diretório do projeto, execute:

   ```
   dotnet run
   ```

2. **Acesse a API:** Abra seu navegador e navegue até http://localhost:5000/api/produtos para ver a lista de produtos.

Conclusão

Neste artigo, criamos um ProdutosController básico para gerenciar operações CRUD em uma aplicação ASP.NET Core no ambiente Windows. Este é um ponto de partida para criar aplicações web robustas e escaláveis utilizando as ferramentas oferecidas pela plataforma .NET.