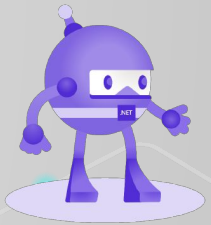
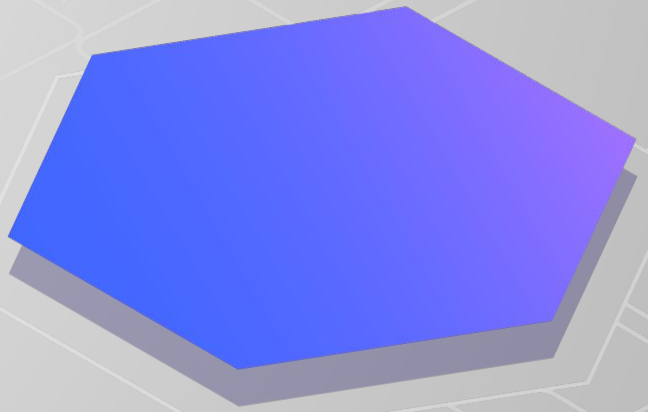


.NET

Minimal APIs con ASP.NET 6



.NET Conf Latam 2021

¡Hola!

Geovani de León

Senior Software Developer at PlusTI

Docente Universitario

Líder de la comunidad Microsoft Dev Group Cobán

Guatemalteco

https://github.com/yovafree

https://yovadeleon.dev

¿Qué es Minimal APIs?

Es la propuesta del equipo de Desarrollo de .NET 6, el cual permite crear APIs con una cantidad de código reducida.

Ideal para:

- Microservicios
- Aplicaciones que desean incluir solo los archivos, las características y las dependencias mínimas en ASP.NET Core.
- APIs ligeras o con funcionalidad pequeña.
- Abre la puerta para el uso de **Azure Functions**

.NET 6 Minimal API Framework

- En .NET 5, se introdujeron programas de nivel superior.
- Significa se puede abrir un archivo .cs, escribir algo de código y ejecutarlo sin espacios de nombres, clases y cualquier otro elemento que lo frene para ejecutarse.
- Las Minimal APIs de .NET 6 llevan el minimalismo a otro nivel.

Características de Lenguaje en Plantillas de .NET 6 Mininal API

- Declaraciones de nivel superior
- async Main
- Directivas de uso globales (a través de valores predeterminados impulsados por el SDK)
- Espacios de nombres con ámbito de archivo
- Expresiones de tipo de destino **new**
- Tipos de referencia que aceptan valores NULL

Demo 1

Comparativa .NET 5 WebAPI vs .NET 6 Minimal APIs

.NET 5

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace net5
{
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        1 reference
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

```
public class Startup
{
    0 references
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    1 reference
    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllers();
        services.AddSwaggerGen(c =>
        {
            c.SwaggerDoc("v1", new OpenApiInfo { Title = "net5", Version = "v1" });
        });
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    0 references
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
            app.UseSwagger();
            app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "net5 v1"));
        }
        app.UseHttpsRedirection();
        app.UseRouting();
        app.UseAuthorization();
        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```

.NET 6

```
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new() { Title = "net6", Version = "v1" });
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (builder.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
    app.UseSwagger();
    app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "net6 v1"));
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Directivas de Uso Globales

Disponibles en los SDKs:

Microsoft.NET.Sdk

Microsoft.NET.Sdk.Web

Microsoft.NET.Sdk.Worker

Microsoft.NET.Sdk.WindowsDesktop

SDK	Default namespaces
Microsoft.NET.Sdk	System System.Collections.Generic System.IO System.Linq System.Net.Http System.Threading System.Threading.Tasks
Microsoft.NET.Sdk.Web	System.Net.Http.Json Microsoft.AspNetCore.Builder Microsoft.AspNetCore.Hosting Microsoft.AspNetCore.Http Microsoft.AspNetCore.Routing Microsoft.Extensions.Configuration Microsoft.Extensions.DependencyInjection Microsoft.Extensions.Hosting Microsoft.Extensions.Logging
Microsoft.NET.Sdk.Worker	Microsoft.Extensions.Configuration Microsoft.Extensions.DependencyInjection Microsoft.Extensions.Hosting Microsoft.Extensions.Logging
Microsoft.NET.Sdk.WindowsDesktop (Windows Forms)	Microsoft.NET.Sdk namespaces System.Drawing System.Windows.Forms
Microsoft.NET.Sdk.WindowsDesktop (WPF)	Microsoft.NET.Sdk namespaces Removed System.IO Removed System.Net.Http

.NET 6 Minimal APIs

- WebApplication y WebApplicationBuilder
- Hot Reload en Minimal APIs
- Trabajo con puertos
- Lectura del entorno
- Logs
- Lectura de la configuración
- Proveedor de registro

.NET 6 Minimal APIs

- Gestión de solicitudes
- Controladores de ruta (Lambda, Función Local, Método de Instancia).
- Parámetros de ruta
- Restricciones de ruta
- Parámetros opcionales

Demo 2

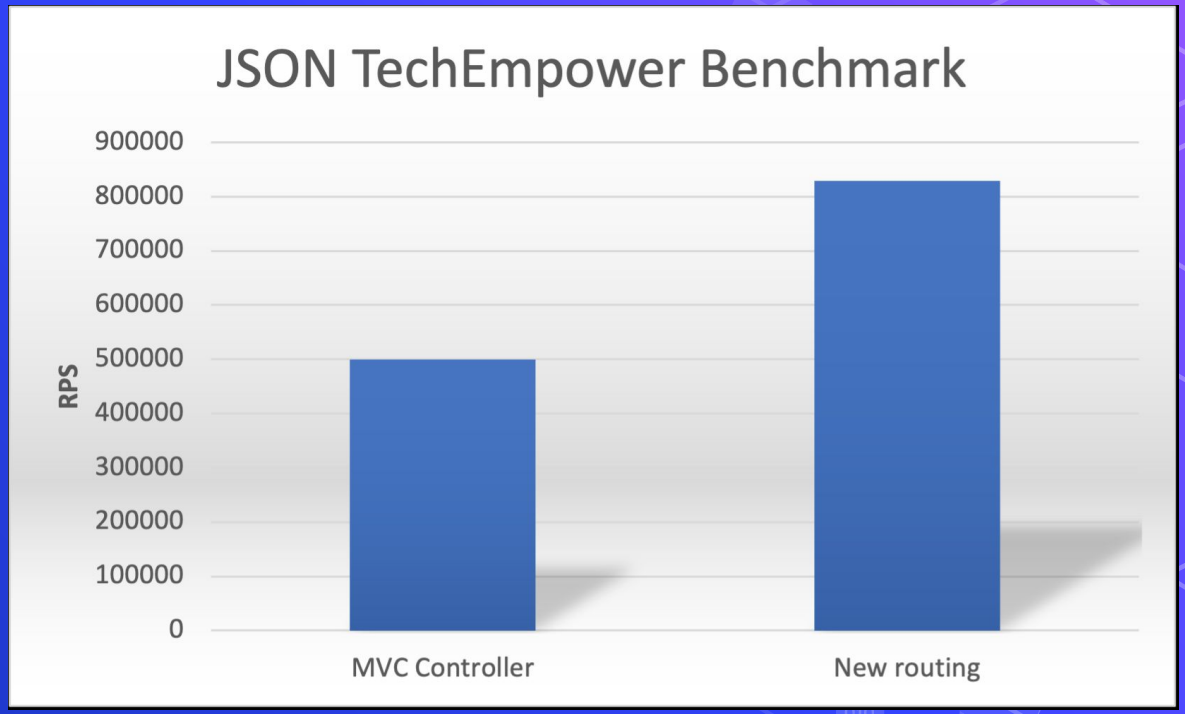
Swagger

Demo 3

MapControllers

Rendimiento

Estas nuevas API de enrutamiento tienen una sobrecarga mucho menor que las API basadas en controladores. Con las nuevas API de enrutamiento, ASP.NET Core puede alcanzar ~ 800k RPS en el benchmark TechEmpower JSON frente a ~ 500k RPS para MVC.



Fuente:
https://devblogs.microsoft.com/dotnet/asp-net-core-updates-in-net-6-preview-4/#performance

Diferencias entre las API mínimas y las API con controladores

- No se admiten filtros: por ejemplo, no se admiten IAsyncAuthorizationFilter, IAsyncActionFilter, IAsyncExceptionFilter, IAsyncResultFilter ni IAsyncResourceFilter.
- No se admite el enlace de modelos, es decir, IModelBinderProvider y IModelBinder. Se puede agregar compatibilidad con una corrección de compatibilidad de enlace personalizada.
- No se admite el enlace desde formularios. Esto incluye el enlace IFormFile. Tenemos previsto agregar compatibilidad con IFormFile en el futuro.
- No hay compatibilidad integrada con la validación, es decir, IModelValidator.
- No se admiten elementos de aplicación ni el modelo de aplicación. No hay ninguna manera de aplicar o crear sus propias convenciones.
- No se admite la representación de vistas integrada. Se recomienda usar Razor Pages para representar vistas.
- No se admite JsonPatch.
- No se admite OData.
- No se admite ApiVersioning. Vea esta incidencia para obtener más información.

Conclusiones y consideraciones

- Minimal != Simple
- El rendimiento es un beneficio pero no el beneficio
- Más opciones es algo bueno
- Una curva de aprendizaje de entrada más baja
- Para proyectos grandes considerar si es adecuado utilizar Minimal APIs

¡Gracias!

¿Preguntas?

Puedes comunicarte conmigo:

- @yovafree
- https://github.com/yovafree

https://github.com/yovafree/dotnet-conf-latam-2021