Что нового для Web API в ASP.NET Core 9

Андрей Порожняков
.NET разработчик в SKAI

О чём поговорим

- Web API в .NET
- Новые фичи в ASP.NET Core 9.0
- Гибридное кеширование
- Выводы

Эволюция Web API в .NET

2007

ASP.NET MVC

демонстрация первой версии

2012

Микросервисная архитектура

распространяется с середины 2010-х

2022

Minimal API

с выходом .NET 6

Пример контроллера

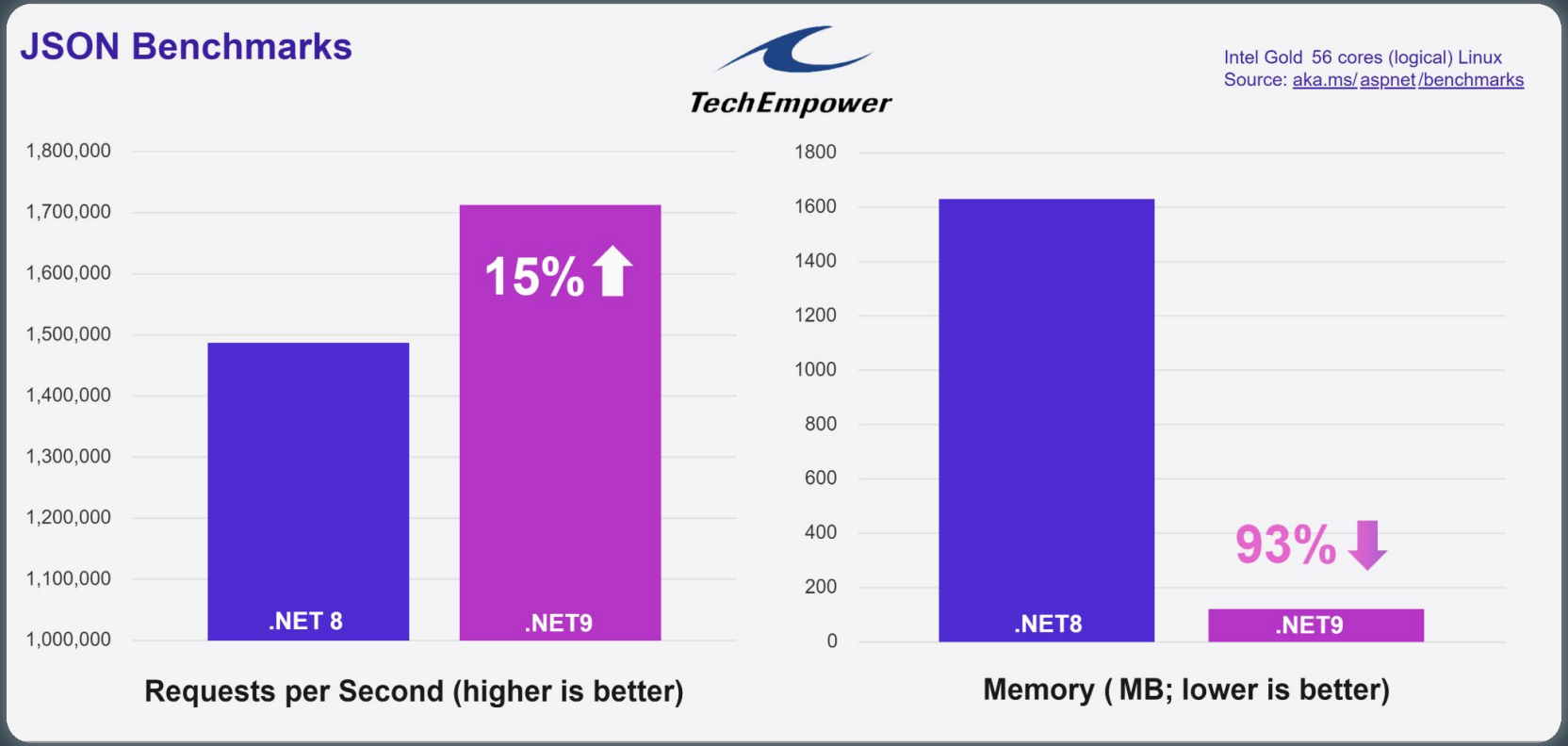
```
1.  [ApiController]
2.  [Route("")]
3.  public class CustomController : ControllerBase
4.  {
5.    [HttpGet]
6.    public string Get() ⇒ "Hello World!";
7.  }
```

Пример Minimal API

Содержимое файла Program.cs:

```
1.  var app = WebApplication.Create(args);
2.  app.MapGet("/", () ⇒ "Hello World!");
3.  app.Run();
```

Производительность Minimal API



OpenAPI

Шаблон ASP.NET Core Web API

ASP.NET Core 5-8

- Библиотека Swashbuckle.AspNetCore
- Services.AddSwaggerGen() для регистрации сервисов
- app.UseSwagger() и app.UseSwaggerUI() для подключения middleware

ASP.NET Core 9

- Библиотека Microsoft.AspNetCore.OpenApi
- Services.AddOpenApi() для регистрации сервисов
- app.MapOpenApi() для регистрации endpoint-а /openapi/v1.json

→ OpenAPI

Вернуть Swagger

Удаляем пакет
Microsoft.AspNetCore.OpenApi

→

Устанавливаем пакет
Swashbuckle.AspNetCore

```
1.  var builder = WebApplication.CreateBuilder(args);
2.  builder.Services.AddSwaggerGen();
3.  var app = builder.Build();
4.  app.UseSwagger();
5.  app.UseSwaggerUI();
6.  app.MapGet("/", ⇒ "Hello World!");
7.  app.Run();
```

UI доступен по localhost:<port>/swagger/index.html

Использовать OpenAPI + Swagger

Устанавливаем пакет **Swashbuckle.AspNetCore.SwaggerUI**

```
1.  var builder = WebApplication.CreateBuilder(args);
2.  builder.Services.AddOpenApi();
3.  var app = builder.Build();
4.  app.MapOpenApi();
5.  app.UseSwaggerUI(o ⇒
6.    o.SwaggerEndpoint("/openapi/v1.json", "v1"));
7.  app.MapGet("/", ⇒ "Hello World!");
8.  app.Run();
```

UI доступен по localhost:<port>/swagger/index.html

Использовать OpenAPI + Scalar

Устанавливаем пакет **Scalar.AspNetCore**

```
1.  var builder = WebApplication.CreateBuilder(args);
2.  builder.Services.AddOpenApi();
3.  var app = builder.Build();
4.  app.MapOpenApi();
5.  app.MapScalarApiReference();
6.  app.MapGet("/", ⇒ "Hello World!");
7.  app.Run();
```

UI доступен по localhost:<port>/scalar/v1

ProducesProblem для групп

```
1.  var group1 = app
2.    .MapGroup("/group1")
3.    .ProducesProblem(403, "application/problem+json");
4.  group1.MapGet("/one", () ⇒ "Hello World!");
5.  var group2 = app
6.    .MapGroup("/group2")
7.    .ProducesValidationProblem();
8.  group2.MapGet("/two", () ⇒ "Hello World!");
```

Keyed Services

Позволяет получить реализацию интерфейса по ключу

```
1.  var bldr = WebApplication.CreateBuilder(args);
2.  bldr.Services.AddKeyedSingleton<ISender, EmailSender>("email");
3.  bldr.Services.AddKeyedSingleton<ISender, SmsSender>("sms");
4.  var app = builder.Build();
5.  app.MapGet("/sms", ([FromKeyedServices("sms")] ISender sender)
6.    => sender.SendAsync("sms message text"));
7.  app.Run();
```

Использование в Middleware

```
1.  internal partial class CustomMiddleware
2.  {
3.    private readonly RequestDelegate _next;
4.
5.    public CustomMiddleware(RequestDelegate next,
6.      [FromKeyedServices("sms")] ISender sender)
7.    {
8.      _next = next;
9.    }
10.
11.   public Task Invoke(HttpContext cx,
12.     [FromKeyedServices("email")] ISender sender) => next(cx);
13. }
```

Выборочное отключение метрик

```
1.  var builder = WebApplication.CreateBuilder(args);
2.  builder.Services.AddOpenTelemetry();
3.  var app = builder.Build();
4.  app.MapGet("/", () ⇒ "endpoint").DisableHttpMetrics();
5.  var group = app.MapGroup("/group").DisableHttpMetrics();
6.  group.MapGet("/", () ⇒ "group");
7.  app.MapHealthChecks("/healthz").DisableHttpMetrics();
8.  app.Run();
```

Использование в контроллерах

```
1.  [ApiController]
2.  [Route("")]
3.  [DisableHttpMetrics]
4.  public class CustomController : ControllerBase
5.  {
6.    [DisableHttpMetrics]
7.    [HttpGet]
8.    public string Get() ⇒ "Hello World!";
9.  }
```

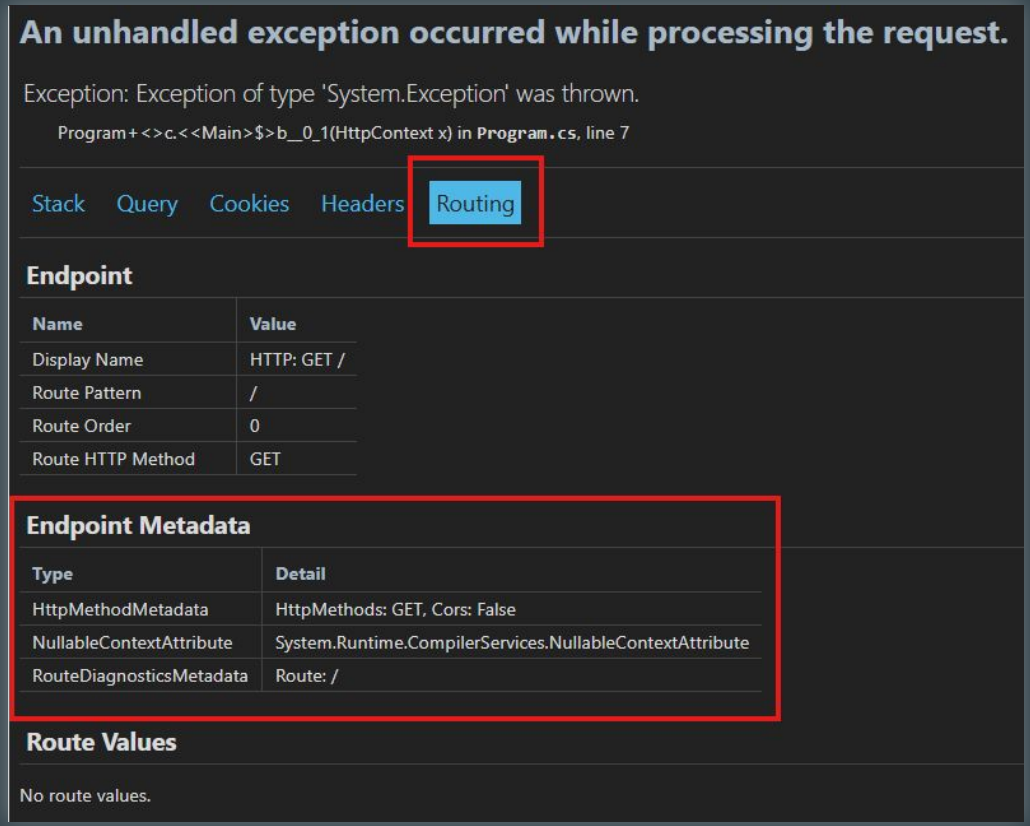
Использование в Middleware

```
1.  internal partial class CustomMiddleware(RequestDelegate next)
2.  {
3.    public Task Invoke(HttpContext ctx)
4.    {
5.      var mf = ctx.Features.Get<IHttpMetricsTagsFeature>();
6.      if (ctx.Request.Headers.ContainsKey("x-disable-metrics"))
7.      {
8.        mf.MetricsDisabled = true;
9.      }
10.
11.     return next(ctx);
12.   }
13. }
```

Глобальная обработка исключений

```
1.  app.UseExceptionHandler(new ExceptionHandlerOptions
2.  {
3.    StatusCodeSelector = ex ⇒ ex is TimeoutException
4.      ? StatusCodes.Status503ServiceUnavailable
5.      : StatusCodes.Status500InternalServerError
6.  });
```

Страница обработчика исключений



TypedResults может возвращать 500

Добавлены методы **InternalServerError** и **InternalServerError<TValue>**

```
1.  var app = WebApplication.Create(args);
2.  app.MapGet("/", () ⇒
3.    TypedResults.InternalServerError("Упс"));
4.  app.Run();
```

Кэширование в ASP.NET Core

Существующие реализации

Кэширование в памяти

- Интерфейс **IMemoryCache**
- Хранится в процессе
- Теряется при перезапуске

Распределённый кэш

- Интерфейс **IDistributedCache**
- Внешнее хранилище
- Сохраняется при перезапуске

Гибридное кэширование

Устанавливаем пакет

Microsoft.Extensions.Caching.Hybrid

```
1.  var builder = WebApplication.CreateBuilder(args);
2.  builder.Services.AddHybridCache();
3.  var app = builder.Build();
4.  app.MapGet("/{key}", async (string key,
5.    HybridCache cache, CancellationToken ct) =>
6.    await cache.GetOrCreateAsync(key,
7.      async token => await SomeFuncAsync(key, token),
8.      cancellationToken: ct));
9.  app.Run();
```

HybridCache в ASP.NET Core

- Как и зачем появилось
- Как подключить и настроить
- Ограничения и особенности
- Примеры использования
- Что происходит под капотом

Статья про гибридное кэширование

Запомнить

- .NET и ASP.NET Core становятся быстрее
- Убрали Swagger из шаблона Web API
- Сделано много небольших, но важных изменений
- Появилось гибридное кэширование

Ссылки на источники

- Анонс выхода .NET 9
- Новые возможности ASP.NET Core 9.0
- Использование OpenAPI
- Почему Swashbuckle.AspNetCore удалён в .NET 9
- Моя статья про гибридное кэширование

Мои контакты

📩 aporozhniakov@gmail.com