Что нового в Minimal API на ASP.NET Core 8

Андрей Порожняков
.NET разработчик

О чём поговорим

- Что такое Minimal API, пример приложения
- Новые фичи Minimal API в ASP.NET Core 8.0
- Немного про Native AOT
- Выводы

Пример приложения на Minimal API

Содержимое файла Program.cs:

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

Работа с формами. Атрибут FromForm

- ✅ Работает для Content-Type **multipart/form-data** и **application/x-www-form-urlencoded**
- ✅ Добавляет **IAntiforgeryMetadata** к endpoint-ам

```
1.  var app = WebApplication.Create(args);
2.  app.MapPost("/handle-form", ([FromForm] MyObj obj)
3.    ⇒ Results.Ok(obj));
4.  app.Run();
```

Межсайтовая подделка запроса

Cross-Site Request Forgery, CSRF, или подделка межсайтовых запросов — это атака, при которой вредоносный сайт отправляет запрос на уязвимый сайт, где пользователь в данный момент вошёл в систему.

Для предотвращения CSRF-атаки сначала генерируется, а затем валидируется **anti-forgery token**. В ASP.NET Core валидация происходит на уровне **AntiforgeryMiddleware**.

Валидация токена выполняется, если в метадате endpoint-а есть **IAntiforgeryMetadata**.

→ работа с формами

AntiforgeryMiddleware

Чтобы подключить **AntiforgeryMiddleware**, используются методы **.AddAntiforgery()** и **.UseAntiforgery()**

Валидацию anti-forgery token можно отключить с помощью вызова **.DisableAntiforgery()** при объявлении endpoint-а

```
1.  var builder = WebApplication.CreateBuilder(args);
2.  builder.Services.AddAntiforgery();
3.  var app = builder.Build();
4.  app.UseAntiforgery();
5.  app.MapPost("/handle-form", ([FromForm] MyObj obj)
6.    => Results.Ok(obj)).DisableAntiforgery();
7.  app.Run();
```

→ работа с формами. Пример

```
1.  app.MapGet("/", (HttpContext ctxt, IAntiforgery antiforgery) ⇒
2.  {
3.      var token = antiforgery.GetAndStoreTokens(ctxt);
4.      var html = $"""<html><body>
5.  <form action="/handle-form" method="POST">
6.  <input name="{token.FormFieldName}" type="hidden"
7.         value="{token.RequestToken}" />
8.  <input type="text" name="comment" />
9.  <input type="submit" />
10. </form></body></html>""";
11.     return Results.Content(html, "text/html");
12. }
13. app.MapPost("/handle-form", ([FromForm] string comment)
14.     ⇒ Results.Ok(comment));
```

Атрибут FromKeyedServices

В DI добавлены методы **.AddKeyedSingleton(), .AddKeyedScoped()** и **.AddKeyedTransient()**

Атрибут **[FromKeyedServices("key")]** 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();
```

Short-circuit routing

✅ Позволяет проигнорировать все middleware, находящиеся после **EndpointRoutingMiddleware**.

✅ Снижает нагрузку на приложение и смежные системы: БД, логирование и прочие.

→ short-circuit routing

Метод .ShortCircuit()

- Вызывается у существующего endpoint-а
- Имеет необязательный параметр statusCode
- Параметр statusCode не учитывается, если endpoint возвращает код ответа
- Отображается в Swagger

```
1.  var app = WebApplication.Create(args);
2.  app.MapGet("/foo", () ⇒ Results.Ok()).ShortCircuit();
3.  app.MapGet("/bar", () ⇒
    Task.CompletedTask).ShortCircuit(400);
4.  app.MapGet("/baz", () ⇒ Results.Ok()).ShortCircuit(400);
5.  app.Run();
```

→ short-circuit routing

Метод .MapShortCircuit()

- Имеет обязательные параметры **statusCode** и **params[] routePrefixes**
- Создаёт новые endpoint-ы без внутренней логики при помощи .**Map()**
- Преобразует каждый маршрут в **routePrefix/{**catchall}**
- Не отображается в Swagger

```
1.  var app = WebApplication.Create(args);
2.  app.MapShortCircuit(400, "foo", "bar");
3.  app.Run();
```

Детальный обзор на short-circuit routing

- Зачем нужен short-circuit routing
- Обзор и сравнение методов **.ShortCircuit()** и **.MapShortCircuit()**
- Какие middleware будут пропущены, и как этим управлять
- Примеры использования

Native ahead-of-time (AOT)

Новый шаблон **ASP.NET Core Web API (native AOT)**. Короткое имя **webapiaot**

Изменения в .csproj

- <InvariantGlobalization>true</InvariantGlobalization>
- <PublishAot>true</PublishAot>

Изменения в Program.cs

- Использует .CreateSlimBuilder(args) вместо .CreateBuilder(args)
- Настраивает приложение для использования генератора исходного кода JSON
- Каждый сериализуемый объект должен быть явно прописан в JsonSerializerContext

→ Native ahead-of-time (AOT)

Ограничения CreateSlimBuilder

- ⛔ Не поддерживаются hosting startup assemblies и вызов **.UseStartup<Startup>()**
- ⛔ Не поддерживается загрузка статических ассетов из подключенных проектов и пакетов через вызов **.UseStaticWebAssets()**
- ⛔ Не поддерживается интеграция с IIS
- ⛔ Не поддерживается HTTPS и Quic (HTTP/3)
- ⛔ Не поддерживаются RegEx выражения в роутинге
- ⛔ Урезанные возможности логирования: **нет** провайдеров логирования Windows event log, Windows Event Tracing и провайдеров логирования в debugger console

→ Native ahead-of-time (AOT). Пример

```
1.  var builder = WebApplication.CreateSlimBuilder(args);
2.  builder.Services.ConfigureHttpJsonOptions(options ⇒
3.  {
4.      options.SerializerOptions.TypeInfoResolverChain.Insert(0,
    AppJsonSerializerContext.Default);
5.  });
6.  var app = builder.Build();
7.  app.MapGet("/", () ⇒ new MyObj("Hello!"))
8.  app.Run();
9.
10. [JsonSerializable(typeof(MyObj))]
11. internal partial class AppJsonSerializerContext :
    JsonSerializerContext { }
```

Выводы

- Minimal API активно развивается, появляются новые возможности
- Новые фичи не только догоняют контроллеры, но и опережают их
- Безальтернативность использования при публикации AOT-приложения

Ссылки на источники

- What's new in ASP.NET Core 8.0 | Microsoft Learn
- Series: Exploring the .NET 8 preview | Andrew Lock
- Тонкости работы short-circuit routing в ASP.NET Core 8.0 | Habr
- https://www.youtube.com/watch?v=UNViYZRpE_o
- https://github.com/dotnet/aspnetcore

Мои контакты

📩 aporozhniakov@gmail.com