

.NET 7 Minimal APIs



Roberto Sanz Ciriano
[@rsciriano](https://twitter.com/rsciriano)

Backend developer



Co-coordinador



Ex.



What is minimal API?

Minimal APIs are a simplified approach for building fast HTTP APIs with ASP.NET Core. You can build fully functioning REST endpoints with minimal code and configuration.

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis/overview?view=aspnetcore-7.0>

Minimal APIs are architected to create HTTP APIs with minimal dependencies. They are ideal for microservices and apps that want to include only the minimum files, features, and dependencies in ASP.NET Core.

<https://learn.microsoft.com/en-us/aspnet/core/tutorials/min-web-api?view=aspnetcore-7.0&tabs=visual-studio>

What is minimal API?

```
Program.cs  [icon] X
Net7MinimalApi
1  var builder = WebApplication.CreateBuilder(args);
2  var app = builder.Build();
3  app.MapGet("/api/hello", (string name) => $"Hello {name ?? "world"}!");
4  app.Run();
```

```
Net7MinimalApi  [icon] X
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
</Project>
```

https://localhost:7062/api/hello

https://localhost:7062/api/hello

Hello world!

Classic API Hello world

```
Program.cs
Net7ClassicApi

1  var builder = WebApplication.CreateBuilder(args);
2  builder.Services.AddControllers();
3  var app = builder.Build();
4  app.MapControllers();
5  app.Run();
```

```
HelloController.cs
Net7ClassicApi

1  using Microsoft.AspNetCore.Mvc;
2
3  namespace Net7ClassicApi.Controllers;
4
5  [ApiController]
6  [Route("api/hello")]
7  public class HelloController : ControllerBase
8  {
9      [HttpGet]
10     public string Get(string name)
11     {
12         return $"Hello {name ?? "world"}!";
13     }
14 }
```

```
Net7ClassicApi
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
</Project>
```

```
https://localhost:7062/api/hello
https://localhost:7062/api/hello

Hello world!
```

The Awesome Hello Worlds API [WIP] 😊



<https://bit.ly/AwesomeHelloWorldsAPI>
(<https://github.com/rsciriano/HelloWorldsApi>)

Minimal CRUD v0.1

Lo sé, aquí hay unas cuantas cosas que mejorar

```
Program.cs  X
Net7MinimalApi

21 app.MapGet("api/worlds", (IWorldRepository repository) =>
22 {
23     return repository.GetAll();
24 });
25
26 app.MapGet("api/worlds/{id:int}", (int id, IWorldRepository repository) =>
27 {
28     return repository.GetById(id);
29 });
30
31 app.MapGet("api/worlds/{name:regex(^[a-zA-Z_-]+$})", (string name, IWorldRepository repository) =>
32 {
33     return repository.GetByName(name);
34 });
35
36 app.MapPost("api/worlds", (World world, IWorldRepository repository) =>
37 {
38     return repository.Create(world);
39 });
40
41 app.MapPut("api/worlds", (World world, IWorldRepository repository) =>
42 {
43     return repository.Update(world);
44 });
```

Classic CRUD v0.1

```
6 [Route("api/worlds")]
  1 reference
7 public class WorldsController : Controller
8 {
9     private readonly IWorldRepository _repository;
10
11     0 references
12     public WorldsController(IWorldRepository repository)
13     {
14         _repository = repository ?? throw new ArgumentNullException(nameof(repository));
15     }
16
17     [HttpGet]
18     0 references
19     public Task<IEnumerable<World>> GetAll()
20     {
21         return _repository.GetAll();
22     }
23
24     [HttpGet("{id:int}")]
25     0 references
26     public Task<World?> GetById(int id)
27     {
28         return _repository.GetById(id);
29     }
30
31     [HttpGet("{name:regex(^[[a-zA-Z_-]]+$)}")]
32     0 references
33     public Task<World?> GetByName(string name)
34     {
35         return _repository.GetByName(name);
36     }
37
38     [HttpPost]
39     0 references
40     public Task<World> Create([FromBody]World world)
41     {
42         return _repository.Create(world);
43     }
44 }
```

Aquí también se pueden
mejorar unas cuantas
cosas 😊

```
36 [HttpPut]
37 0 references
38 public Task<World> Update([FromBody] World world)
39 {
40     return _repository.Update(world);
41 }
42
43 [HttpDelete("{id:int}")]
44 0 references
45 public Task Delete(int id)
46 {
47     return _repository.Delete(id);
48 }
```

Classic Api v0.2 - ResponseCodes

```
[Route("api/worlds")]
```

1 reference

```
public class WorldsController : Controller  
{
```

```
    [HttpGet(Name = "GetAllWorlds")]
```

```
    [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(IEnumerable<World>))]
```

```
    [SwaggerOperation(Tags = new[] { "Worlds" })]
```

0 references

```
    public async Task<IActionResult> GetAll()
```

```
    {
```

```
        return Ok(await _repository.GetAll());
```

```
    }
```

```
    [HttpPost(Name = "CreateWorld")]
```

```
    [ProducesResponseType(StatusCodes.Status201Created, Type = typeof(World))]
```

```
    [SwaggerOperation(Tags = new[] { "Worlds" })]
```

0 references

```
    public async Task<IActionResult> Create([FromBody]World world)
```

```
    {
```

```
        return CreatedAtRoute(  
            "GetWorldById",  
            new { id = world.Id },  
            await _repository.Create(world));
```

```
    }
```

```
    [HttpGet("{id:int}", Name = "GetWorldById")]
```

```
    [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(World))]
```

```
    [ProducesResponseType(StatusCodes.Status404NotFound)]
```

```
    [SwaggerOperation(Tags = new[] { "Worlds" })]
```

0 references

```
    public async Task<IActionResult> GetById(int id)
```

```
    {
```

```
        var world = await _repository.GetById(id);
```

```
        ...
```

```
        if (world == null)
```

```
        {
```

```
            return NotFound();
```

```
        }
```

```
        else
```

```
        {
```

```
            return Ok(world);
```

```
        }
```

```
    }
```


Minimal Api v0.2 - ResponseCodes

```
app.MapPost("api/worlds", async (World world, IWorldRepository repository) =>
{
    await repository.Create(world);

    return Results.CreatedAtRoute("GetWorldById", new { id = world.Id }, world);
})
.Produces<World>(StatusCodes.Status201Created)
.WithName("CreateWorld")
.WithTags("Worlds");
```

```
app.MapGet("api/worlds/{id:int}", async (int id, IWorldRepository repository) =>
{
    var world = await repository.GetById(id);

    if (world is null)
    {
        return Results.NotFound();
    }

    return Results.Ok(world);
})
.Produces<World>(StatusCodes.Status200OK)
.Produces(StatusCodes.Status404NotFound)
.WithName("GetWorldById")
.WithTags("Worlds");
```

```
app.MapGet("api/worlds", async (IWorldRepository repository) =>
{
    return Results.Ok(await repository.GetAll());
})
.Produces<IEnumerable<World>>(StatusCodes.Status200OK)
.Produces(StatusCodes.Status404NotFound)
.WithName("GetAllWorlds")
.WithTags("Worlds");
```

Classic Api v0.3 - Model validation

```
[HttpPost(Name = "CreateWorld")]
[ProducesResponseType(StatusCodes.Status201Created, Type = typeof(WorldModel))]
[ProducesResponseType(StatusCodes.Status400BadRequest, Type = typeof(ValidationProblemDetails))]
[SwaggerOperation(Tags = new[] { "Worlds" })]
0 references
public async Task<IActionResult> Create([FromBody]WorldModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(new ValidationProblemDetails(ModelState));
    }

    await _repository.Create(model.Map());

    return CreatedAtRoute("GetWorldById", new { id = model.Id }, model);
}
```

```
5 references
public class WorldModel
{
    [Required]
    [Range(1, int.MaxValue)]
    3 references
    public int? Id { get; set; }

    [Required]
    [RegularExpression("^[a-zA-Z_-]+$")]
    2 references
    public string? Name { get; set; }
}
```

Minimal Api v0.3 - Model validation

```
app.MapPost("api/worlds", async (WorldModel model, IValidator<WorldModel> validator, IWorldRepository repository) =>
{
    var validationResult = validator.Validate(model);
    if (!validationResult.IsValid)
    {
        return Results.BadRequest(validationResult.Errors.AsValidationProblemDetails());
    }

    await repository.Create(model.MapToEntity());

    return Results.CreatedAtRoute("GetWorldById", new { id = model.Id }, model);
})
.Produces<World>(StatusCodes.Status201Created)
.Produces<HttpValidationProblemDetails>(StatusCodes.Status400BadRequest)
.WithName("CreateWorld")
.WithTags("Worlds");
```

```
7 references
public class WorldModel
{
    4 references
    public int? Id { get; set; }

    3 references
    public string? Name { get; set; }
}
```

```
2 references
public class WorldModelValidator: AbstractValidator<WorldModel>
{
    0 references
    public WorldModelValidator()
    {
        RuleFor(_ => _.Id)
            .NotNull()
            .GreaterThan(0)
            .WithMessage("The field Id must be between 1 and 2147483647.");

        RuleFor(_ => _.Name)
            .Cascade(CascadeMode.Stop)
            .NotEmpty()
            .WithMessage("The Name field is required.")
            .Matches("^[a-zA-Z_-]+$");
    }
}
```

Minimal Api v0.4 - EndpointFilters

```
69
70 - app.MapPost("api/worlds", async (WorldModel model, IValidator<WorldModel> validator, IWorldRepository repository) =>
71 {
72     var validationResult = validator.Validate(model);
73     if (!validationResult.IsValid)
74     {
75         return Results.BadRequest(validationResult.Errors.AsValidationProblemDetails());
76     }
77
78     await repository.Create(model.MapToEntity());
79
80     return Results.CreatedAtRoute("GetWorldById", new { id = model.Id }, model);
81 }
82 .Produces<World>(StatusCodes.Status201Created)
83 .Produces<HttpValidationProblemDetails>(StatusCodes.Status400BadRequest)
84 .WithName("CreateWorld")
85 - .WithTags("Worlds");
```

0 references

```
public static class ValidationEndpointFilterExtensions
{
    0 references
    public static RouteHandlerBuilder WithValidation(this RouteHandlerBuilder builder)
    {
        builder.AddEndpointFilter<ValidationEndpointFilter>();
        return builder;
    }

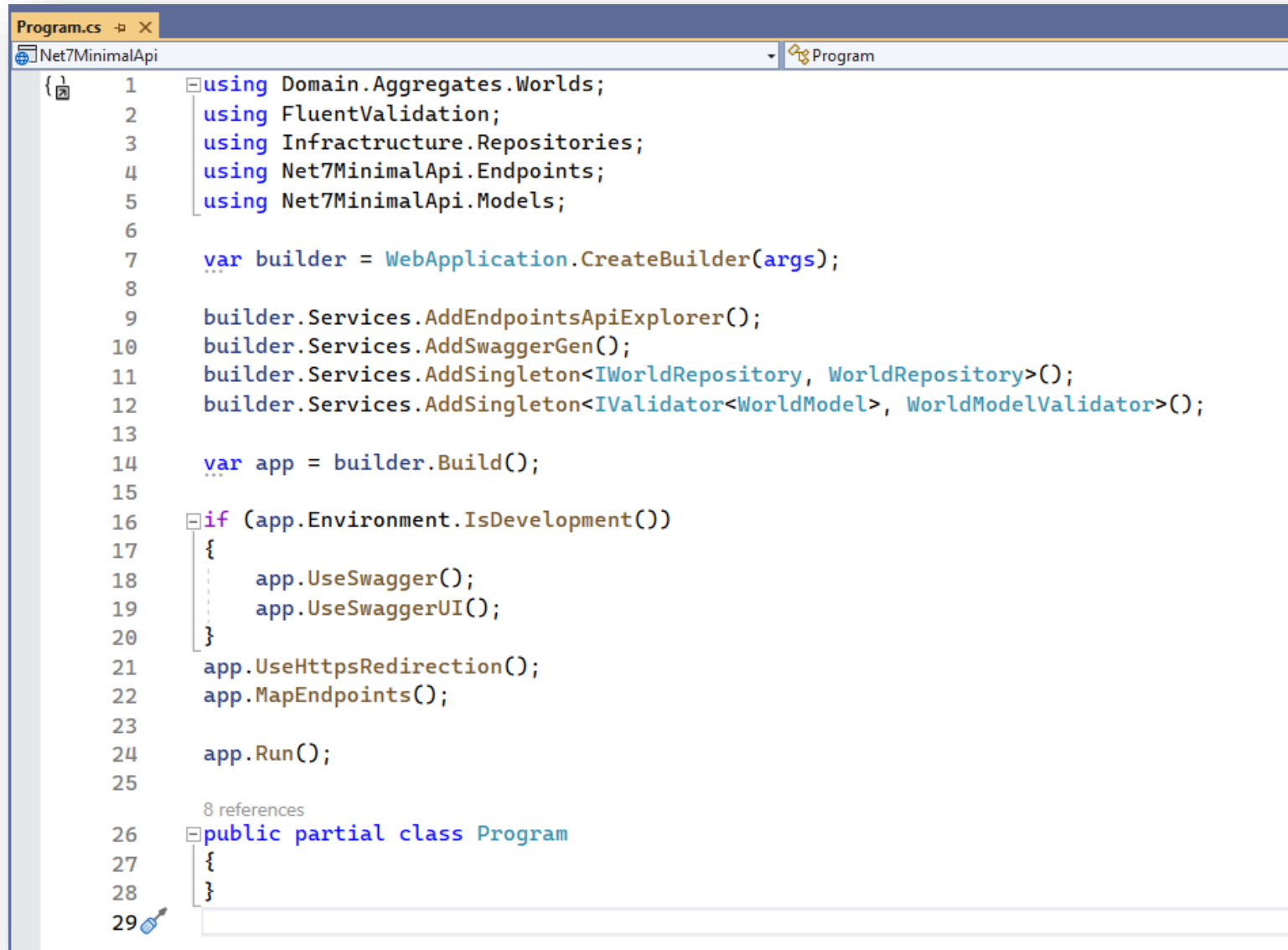
    1 reference
    public static RouteGroupBuilder WithValidation(this RouteGroupBuilder builder)
    {
        builder.AddEndpointFilter<ValidationEndpointFilter>();
        return builder;
    }
}
```

```
69
70 + app.MapPost("api/worlds", async (WorldModel model, IWorldRepository repository) =>
71 {
72
73
74
75 }
76 .Produces<World>(StatusCodes.Status201Created)
77 .Produces<HttpValidationProblemDetails>(StatusCodes.Status400BadRequest)
78 .WithName("CreateWorld")
79 + .WithTags("Worlds")
80 + .WithValidation();
```

Minimal Api v0.4 - EndpointFilters (bis)

```
7 public class ValidationEndpointFilter : IEndpointFilter
8 {
9     private readonly IServiceProvider _serviceProvider;
10
11     0 references
12     public ValidationEndpointFilter(IServiceProvider serviceProvider)
13     {
14         _serviceProvider = serviceProvider ?? throw new ArgumentNullException(nameof(serviceProvider));
15     }
16
17     0 references
18     public async ValueTask<object> InvokeAsync(EndpointFilterInvocationContext context, EndpointFilterDelegate next)
19     {
20         var genericValidatorType = typeof(IValidator<>);
21         foreach (var argument in context.Arguments)
22         {
23             if (argument is null) continue;
24             var argumentType = argument.GetType();
25             if (!argumentType.IsClass) continue;
26
27             var validatorType = genericValidatorType.MakeGenericType(argumentType);
28             var validator = (IValidator)_serviceProvider.GetService(validatorType);
29             if (validator is null) continue;
30
31             var validationTextType = typeof(ValidationContext<>).MakeGenericType(argumentType);
32             var validationContext = (IValidationContext)Activator.CreateInstance(validationTextType, argument);
33             var validationResult = await validator.ValidateAsync(validationContext);
34             if (!validationResult.IsValid)
35             {
36                 return Results.BadRequest(validationResult.Errors.AsValidationProblemDetails());
37             }
38             break;
39         }
40         return await next(context);
41     }
42 }
```

Minimal Api v0.4 - MapGroups (1 / 3)



```
Program.cs
Net7MinimalApi
Program

1 using Domain.Aggregates.Worlds;
2 using FluentValidation;
3 using Infrastructure.Repositories;
4 using Net7MinimalApi.Endpoints;
5 using Net7MinimalApi.Models;
6
7 var builder = WebApplication.CreateBuilder(args);
8
9 builder.Services.AddEndpointsApiExplorer();
10 builder.Services.AddSwaggerGen();
11 builder.Services.AddSingleton<IWorldRepository, WorldRepository>();
12 builder.Services.AddSingleton<IValidator<WorldModel>, WorldModelValidator>();
13
14 var app = builder.Build();
15
16 if (app.Environment.IsDevelopment())
17 {
18     app.UseSwagger();
19     app.UseSwaggerUI();
20 }
21 app.UseHttpsRedirection();
22 app.MapEndpoints();
23
24 app.Run();
25
26 8 references
27 public partial class Program
28 {
29 }
```

Minimal Api v0.4 - MapGroups (2/3)

0 references

```
public static class EndpointsExtensions
```

```
{
```

1 reference

```
public static RouteGroupBuilder MapEndpoints(this IEndpointRouteBuilder endpoints)
```

```
{
```

```
    var group = endpoints
```

```
        .MapGroup("/")
```

```
        .WithValidation();
```

```
    group.MapHelloEndpoints();
```

```
    group.MapWorldEndpoints();
```

```
    return group;
```

```
}
```

```
}
```

```
public static class HelloEndpoints
```

```
{
```

1 reference

```
public static RouteGroupBuilder MapHelloEndpoints(this RouteGroupBuilder group)
```

```
{
```

```
    group.MapGet("api/hello", GetHello)
```

```
        .WithName("Hello")
```

```
        .WithTags("Hello");
```

```
    return group;
```

```
}
```

1 reference

```
private static string GetHello(string? name)
```

```
{
```

```
    return $"Hello {name ?? "world"}!";
```

```
}
```

```
}
```


Minimal Api v0.4 - MapGroups (3/3)

```
0 references
public static class WorldEndpoints
{
    1 reference
    public static RouteGroupBuilder MapWorldEndpoints(this RouteGroupBuilder group)
    {
        group.MapGet("api/worlds", GetAllWorlds)
            .Produces<IEnumerable<World>>(StatusCodes.Status200OK)
            .Produces(StatusCodes.Status404NotFound)
            .WithName("GetAllWorlds")
            .WithTags("Worlds");

        group.MapGet("api/worlds/{id:int}", GetWorldById)
            .Produces<World>(StatusCodes.Status200OK)
            .Produces(StatusCodes.Status404NotFound)
            .WithName("GetWorldById")
            .WithTags("Worlds");

        group.MapGet("api/worlds/{name:regex(\"^[a-zA-Z_-]+$\")}", GetWorldByName)
            .Produces<World>(StatusCodes.Status200OK)
            .Produces(StatusCodes.Status404NotFound)
            .WithName("GetWorldByName")
            .WithTags("Worlds");

        group.MapPost("api/worlds", CreateWorld)
            .Produces<World>(StatusCodes.Status201Created)
            .Produces<HttpValidationProblemDetails>(StatusCodes.Status400BadRequest)
            .WithName("CreateWorld")
            .WithTags("Worlds");

        group.MapPut("api/worlds", UpdateWorld)
            .Produces<World>(StatusCodes.Status200OK)
            .WithName("UpdateWorld")
            .WithTags("Worlds");

        group.MapDelete("api/worlds/{id:int}", DeleteWorld)
            .Produces(StatusCodes.Status204NoContent)
            .WithName("DeleteWorld")
            .WithTags("Worlds");

        return group;
    }
}
```

```
1 reference
private static async Task<IResult> GetAllWorlds(IWorldRepository repository)...
```

```
1 reference
private static async Task<IResult> GetWorldById(int id, IWorldRepository repository)...
```

```
1 reference
private static async Task<IResult> GetWorldByName(string name, IWorldRepository repository)...
```

```
1 reference
private static async Task<IResult> CreateWorld(WorldModel model, IWorldRepository repository)...
```

```
1 reference
private static async Task<IResult> UpdateWorld(World world, IWorldRepository repository)...
```

```
1 reference
private static async Task<IResult> DeleteWorld(int id, IWorldRepository repository)...
```


Next steps *(lo que se ha quedado en el tintero)*

- ▶ Unhandled errors
- ▶ Custom model binders
- ▶ Auth simplification
- ▶ Unit/Integration tests
- ▶ Vertical slices (<https://github.com/fernandoescolar/netcoreconf22/tree/main/barcelona/solution>)
- ▶ Performance
- ▶ ...

Thank you!!

Roberto Sanz Ciriano
[@rsciriano](https://twitter.com/rsciriano)

