TRAILHEAD
TECHNOLOGY PARTNERS

Rebuild your APIs Better, Stronger, Faster

With Minimal APIs

Jonathan “J.” Tower

Overview

A Brief History of RPC Tech in .NET

Quick intro to Minimal APIs

Comparing Controllers and Minimal APIs

Migrating Controllers to Minimal A

TRAILHEAD
TECHNOLOGY PARTNERS

Overview

A Brief History of RPC Tech in .NET

Quick intro to Minimal APIs

Comparing Controllers and Minimal APIs

Migrating Controllers to Minimal A

TRAILHEAD
TECHNOLOGY PARTNERS

Overview

A Brief History of RPC Tech in .NET

Quick intro to Minimal APIs

Comparing Controllers and Minimal APIs

Migrating Controllers to Minimal A

TRAILHEAD
TECHNOLOGY PARTNERS

Overview

A Brief History of RPC Tech in .NET

Quick intro to Minimal APIs

Comparing Controllers and Minimal APIs

Migrating Controllers to Minimal A

TRAILHEAD
TECHNOLOGY PARTNERS

Overview

A Brief History of RPC Tech in .NET

Quick intro to Minimal APIs

Comparing Controllers and Minimal APIs

Migrating Controllers to Minimal A

TRAILHEAD
TECHNOLOGY PARTNERS

Jonathan "J." Tower

Principal Consultant & Partner

Microsoft MVP in .NET

jtower@trailheadtechnology.com

trailheadtechnology.com/blog

jtowermi

jtower

Free

Consultation

bit.ly/th-offer

github.com/trailheadtechnology/minimal-apis

How We Got Here

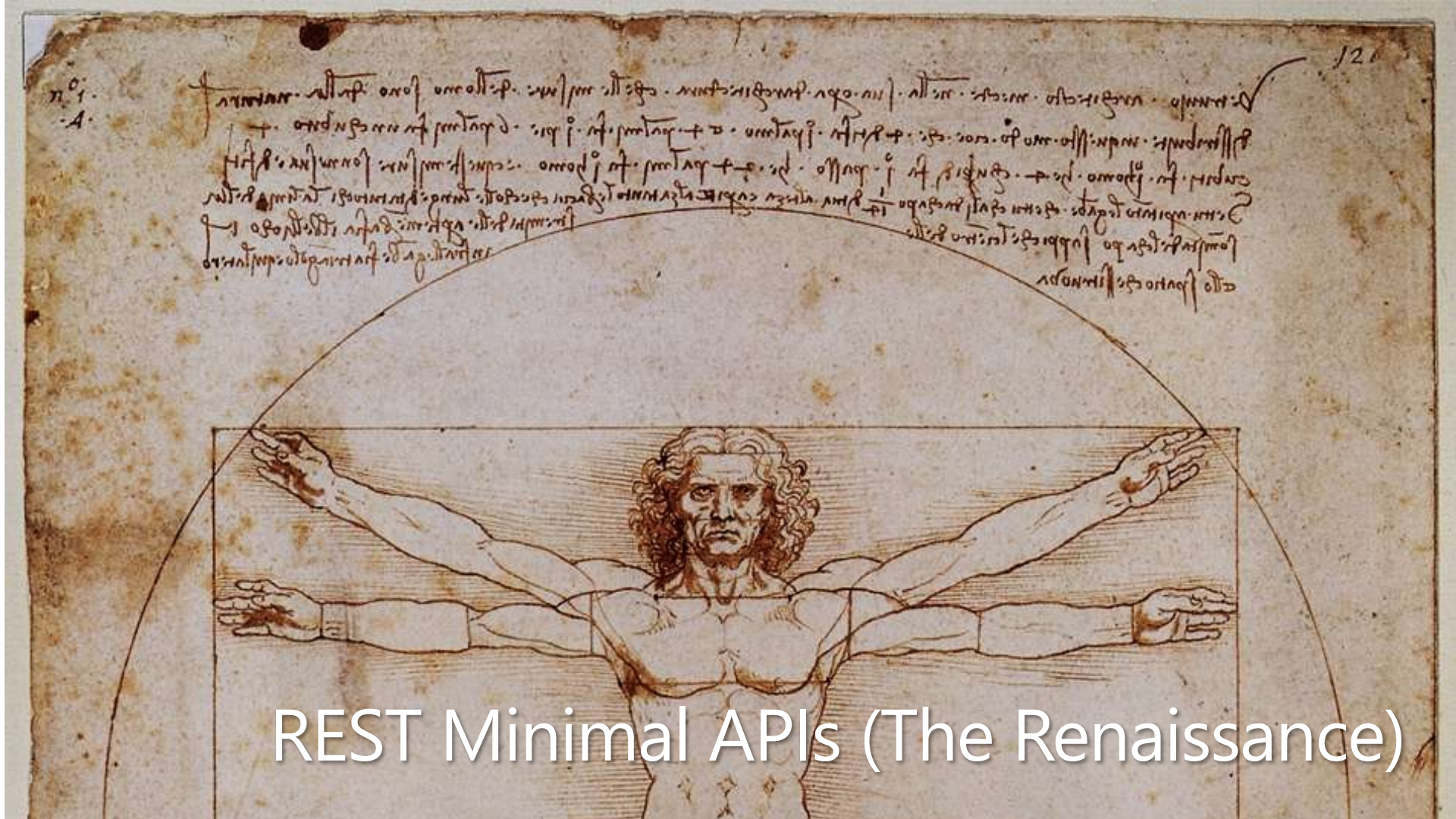
A brief *history* of RPC Technology for .NET Developers

TRAILHEAD
TECHNOLOGY PARTNERS

DCOM (Prehistoric Times)

SOAP & WCF (The Dark Ages)

REST Controllers (The Classical Period)



REST Minimal APIs (The Renaissance)

What's Available Today

Controller-Based APIs

Minimal APIs

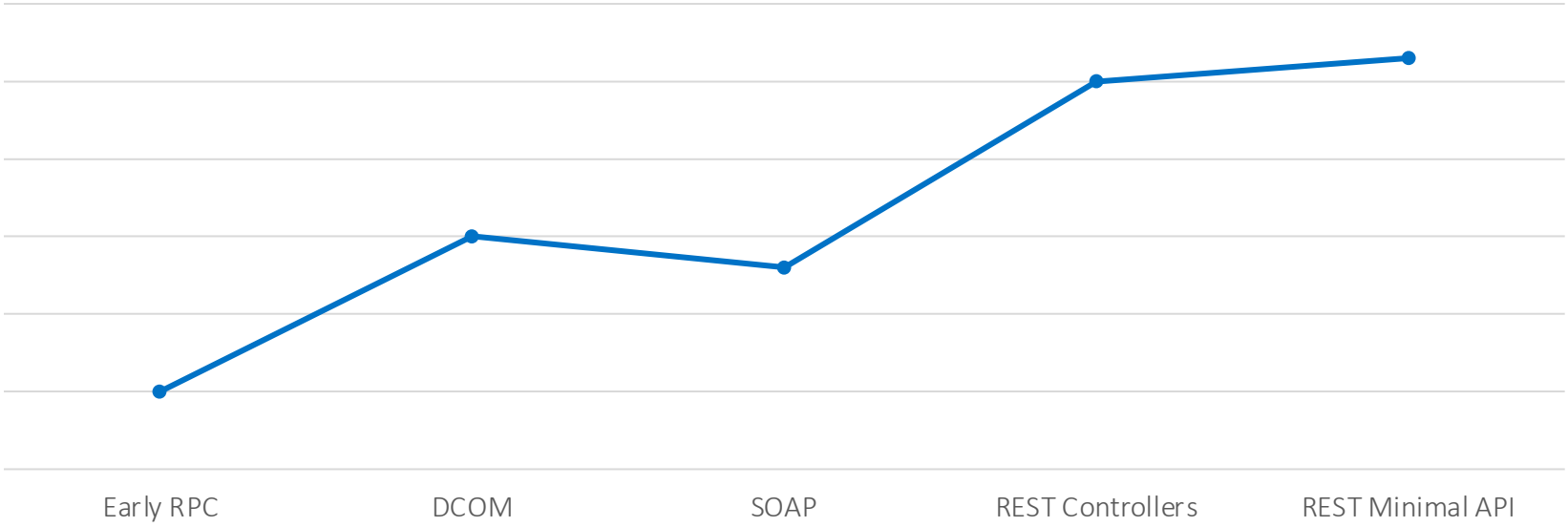
TRAILHEAD
TECHNOLOGY PARTNERS

An Evolutionary Improvement

- Transport and Payload Format Doesn’t Change
- Two Options For Doing The Same Thing
- New Programming Model
- Less Automatic Features
- Simpler, Faster, More Controller

TRAILHEAD
TECHNOLOGY PARTNERS

An Improvement; Not A Revolution



TRAILHEAD
TECHNOLOGY PARTNERS

A Quick Intro

Minimal APIs

Basic Controller API

Starup.cs

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllers();
    }

    public void Configure(IApplicationBuilder app, IHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseRouting();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```

HelloController.cs

```
[ApiController]
[Route("[controller]")]
public class HelloController : ControllerBase
{
    [HttpGet]
    public IActionResult Get()
    {
        return Ok("Hello, API!");
    }
}
```

Basic Minimal API

Progam.cs

```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/", () => "Hello World!");

app.Run();
```

Controllers Vs Minimal APIs

When to Use Each

Controllers and Minimal APIs: Compared

	Minimal APIs	Controllers
Organization	Must organize yourself ⊖	Organized in Controller file, Action Methods ⊕
Control	Granular Control ⊕	Magic ⊖
Bindings	Manual binding ⊖	Convention and manual bindings ⊕
Requirements	Just ASP.NET Core ⊕	MVC / Web API components ⊖
Speed	Fastest ⊕	Fast ⊖
Startup	Repeated startup (Microservices, Serverless) ⊕	Slow, infrequent startup ⊖
Memory	Small memory footprint ⊕	Bigger memory footprint ⊕
Learning	Easier to learn ⊕	Longer learning curve ⊖
Validation	Must validate yourself ⊖	Validation built-in ⊕

Controllers and Minimal APIs: Compared

	Minimal APIs	Controllers
Organization	Must organize yourself −	Organized in Controller file, Action Methods +
Control	Granular Control +	Magic −
Bindings	Manual binding −	Convention and manual bindings +
Requirements	Just ASP.NET Core +	MVC / Web API components −
Speed	Fastest +	Fast −
Startup	Repeated startup (Microservices, Serverless) +	Slow, infrequent startup −
Memory	Small memory footprint +	Bigger memory footprint +
Learning	Easier to learn +	Longer learning curve −
Validation	Must validate yourself −	Validation built-in +

Controllers and Minimal APIs: Compared

	Minimal APIs	Controllers
Organization	Must organize yourself ?	Organized in Controller file, Action Methods ?
Control	Granular Control +	Magic −
Bindings	Manual binding ?	Convention and manual bindings ?
Requirements	Just ASP.NET Core +	MVC / Web API components −
Speed	Fastest +	Fast −
Startup	Repeated startup (Microservices, Serverless) +	Slow, infrequent startup −
Memory	Small memory footprint +	Bigger memory footprint +
Learning	Easier to learn +	Longer learning curve −
Validation	Must validate yourself ?	Validation built-in ?

Organization

“It Depends” #1

Default Organization

Progam.cs

```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/endpoint1", () => ...);
app.MapGet("/endpoint2", () => ...);
app.MapGet("/endpoint3", () => ...);
app.MapGet("/endpoint4", () => ...);
app.MapGet("/endpoint5", () => ...);
app.MapGet("/endpoint6", () => ...);
app.MapGet("/endpoint7", () => ...);
app.MapGet("/endpoint8", () => ...);
app.MapGet("/endpoint9", () => ...);
app.MapGet("/endpoint10", () => ...);
...

app.Run();
```

Better Organization Options

Extension Methods

Your Own Modules

Carter or SmartModules

Extension Methods

MinimalApiExtensions.cs

```
public static class MinimalApiExtensions
{
    public static void MapUserEndpoints(this IEndpointRouteBuilder endpoints)
    {
        endpoints.MapGet("/users", () => "Get all users");
        endpoints.MapGet("/users/{id}", (int id) => $"Get user with ID {id}");
        endpoints.MapPost("/users", () => "Create a new user");
        endpoints.MapPut("/users/{id}", (int id) => $"Update user with ID {id}");
        endpoints.MapDelete("/users/{id}", (int id) => $"Delete user with ID {id}");
    }
}
```

Extension Methods

MinimalApiExtensions.cs

New in .NET 7

```
public static class MinimalApiExtensions
{
    public static void MapUserEndpoints(this IEndpointRouteBuilder endpoints)
    {
        endpoints.MapGroup("/users", builder =>
        {
            builder.MapGet("", () => "Get all users");
            builder.MapGet("/{id}", (int id) => $"Get user with ID {id}");
            builder.MapPost("", () => "Create a new user");
            builder.MapPut("/{id}", (int id) => $"Update user with ID {id}");
            builder.MapDelete("/{id}", (int id) => $"Delete user with ID {id}");
        });
    }
}
```

Extension Methods

Program.cs

```
var app = WebApplication.Create();

app.MapUserEndpoints();
app.MapOtherEndpoints();
app.MapStillMoreEndpoints();

app.Run();
```

Your Own Modules

IModule.cs

```
public interface IModule
{
    void MapEndpoints(IEndpointRouteBuilder endpoints);
}
```

Your Own Modules

UserEndpointsModule.cs

```
public class UserEndpointsModule : IModule
{
    public void MapEndpoints(IEndpointRouteBuilder endpoints)
    {
        endpoints.MapGroup("/users", builder =>
        {
            builder.MapGet("", () => "Get all users");
            builder.MapGet("/{id}", (int id) => $"Get user with ID {id}");
            builder.MapPost("", () => "Create a new user");
            builder.MapPut("/{id}", (int id) => $"Update user with ID {id}");
            builder.MapDelete("/{id}", (int id) => $"Delete user with ID {id}");
        });
    }
}
```

Your Own Modules

ModuleLoader.cs

```
public static class ModuleLoader
{
    public static void LoadAllModules(this IEndpointRouteBuilder endpoints)
    {
        var moduleTypes = Assembly.GetExecutingAssembly().GetTypes()
            .Where(t => typeof(IModule).IsAssignableFrom(t)
                        && !t.IsInterface && !t.IsAbstract);

        foreach (var moduleType in moduleTypes)
        {
            var module = (IModule)Activator.CreateInstance(moduleType);
            module.MapEndpoints(endpoints);
        }
    }
}
```

Your Own Modules

Program.cs

```
app.MapGroup("/api", endpoints =>
{
   endpoints.LoadAllModules();
});
```

Third-Party Modules

```
> dotnet add package carter
```

```
> dotnet add package Codewrinkles.MinimalApi.SmartModules
```

Third-Party Modules

UserModule.cs

```
public class UserModule : ICarterModule
{
    public void AddRoutes(IEndpointRouteBuilder app)
    {
       app.MapGroup("/users", builder =>
       {
          builder.MapGet("", () => "Get all users");
          builder.MapGet("/{id}", (int id) => $"Get user with ID {id}");
          builder.MapPost("", () => "Create a new user");
          builder.MapPut("/{id}", (int id) => $"Update user with ID {id}");
          builder.MapDelete("/{id}", (int id) => $"Delete user with ID {id}");
       });
    }
}
```

Third-Party Modules

Program.cs

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddCarter();

var app = builder.Build();
app.MapCarter();

app.Run();
```

Bindings

“It Depends” #2

Bindings

Minimal API

```
app.MapPost("/api/users", (string name, int age) =>
{
    // 'name' & 'age' bound from body or qs params
    // Minimal APIs inferr from the route and request data
});
```

Simpler
Uses param names, primarily

Controller

```
[HttpPost("/api/users")]
public IActionResult CreateUser([FromBody] Model data)
{
    // 'data' bound from request body using [FromBody]
    // Explicitly defined bindings using attributes
}
```

More Control
Convention first
Attributes second

Validation

“It Depends” #3

Attribute Validation

Minimal API

```
public class Model
{
    [Required(ErrorMessage = "Name is required")]
    [MaxLength(50, ErrorMessage = "Name cannot exceed 50 characters")]
    public string Name { get; set; }

    [Range(18, 100, ErrorMessage = "Age must be between 18 and 100")]
    public int Age { get; set; }
}
```

Fluent Validation

Model

```
public class Model
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

Validator

```
public class Validator  : AbstractValidator<UserCreationModel>
{
    public Validator()
    {
        RuleFor(x => x.Name)
            .NotEmpty().MaximumLength(50);

        RuleFor(x => x.Age).InclusiveBetween(18, 100);
    }
}
```

Validation

Minimal API

```
app.MapPost("/test", async (Model data, Validator val) =>
{
    var results = await val.ValidateAsync(data);

    if (!result.IsValid)
        return Results.BadRequest(result.Errors);

    return Results.Ok("User created successfully");
});
```

Inject a Fluent validator
Validation and persistence separate

Controller

```
public class UserController : ControllerBase
{
    [HttpPost("/test")]
    public IActionResult CreateUser(Model model)
    {
        if (!ModelState.IsValid)
            return BadRequest(ModelState);

        return Ok("User created successfully");
    }
}
```

ModelState built in
Validation and persistence mixed

When To Use Each

The Official Answer

What are Minimal APIs Good For?

Microservices

Speed

Small Footprint

TRAILHEAD
TECHNOLOGY PARTNERS

What are Controllers Good For?

Traditional Monoliths

Enterprise

Convention over Configuration

TRAILHEAD
TECHNOLOGY PARTNERS

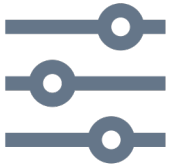
Maybe The Actual Answer?

Microservices

The More Nuanced Answer

Depends on your Needs

Pros/Cons



Your Preference

TRAILHEAD
TECHNOLOGY PARTNERS

Why J. Likes Minimal APIs

Full Control
(Less Magic)

Small

Fast

Clean
Architecture

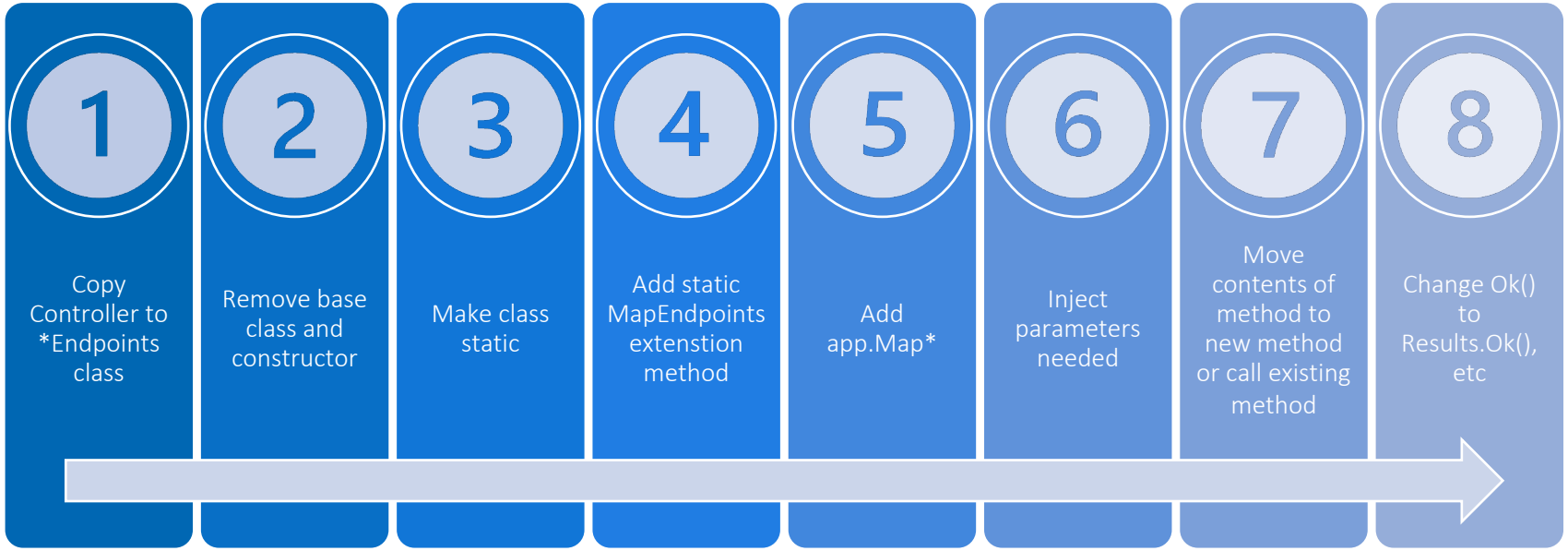
DEMO

Speed and Memory Footprint

Migrating Controllers To Minimal APIs

Step by Step

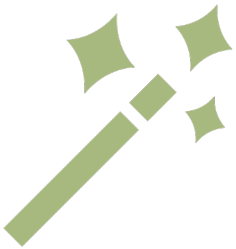
Migration Steps



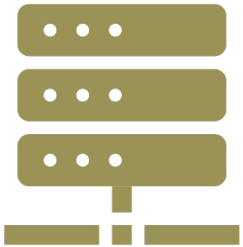
DEMO

Migrate a Controller

Other Helpful Tools

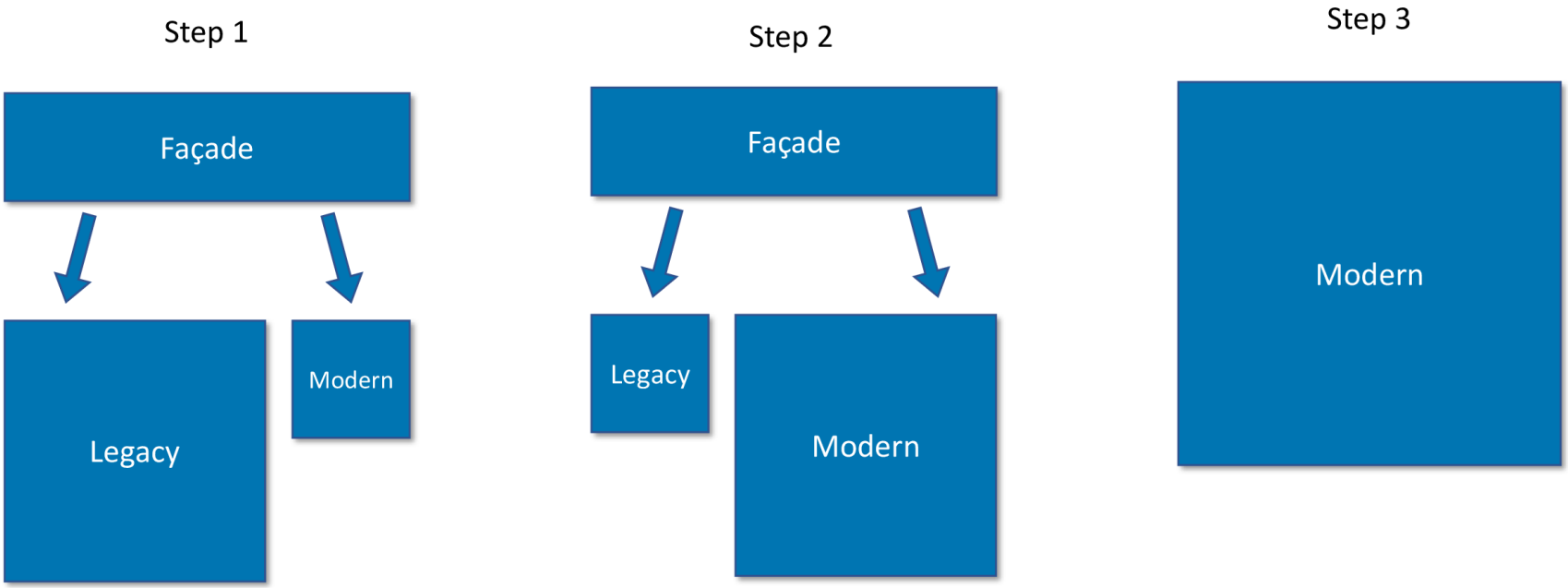


.NET Upgrade Assistant



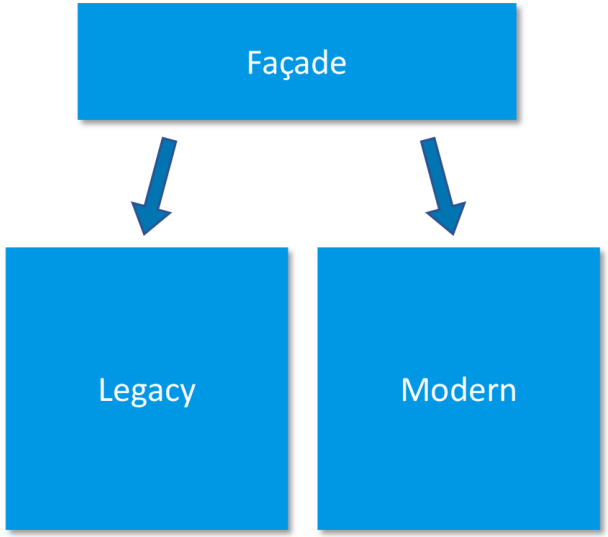
YARP

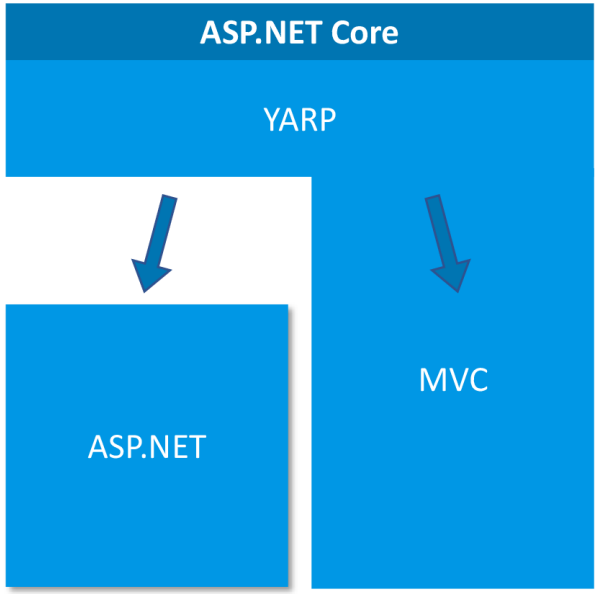
Strangler Fig Pattern



TRAILHEAD
TECHNOLOGY PARTNERS

Strangler Fig With YARP





TRAILHEAD
TECHNOLOGY PARTNERS

DEMO

YARP and .NET Upgrade Assistant

Recap

New Programming Model

Faster and Simpler

Better for Microservices

Same Transport and Payloads

Easy To Migrate To

TRAILHEAD
TECHNOLOGY PARTNERS

Recap

New Programming Model

Faster and Simpler

Better for Microservices

Same Transport and Payloads

Easy To Migrate To

TRAILHEAD
TECHNOLOGY PARTNERS

Recap

New Programming Model

Faster and Simpler

Better for Microservices

Same Transport and Payloads

Easy To Migrate To

TRAILHEAD
TECHNOLOGY PARTNERS

Recap

New Programming Model

Faster and Simpler

Better for Microservices

Same Transport and Payloads

Easy To Migrate To

TRAILHEAD
TECHNOLOGY PARTNERS

Recap

New Programming Model

Faster and Simpler

Better for Microservices

Same Transport and Payloads

Easy To Migrate To

TRAILHEAD
TECHNOLOGY PARTNERS

Recap

New Programming Model

Faster and Simpler

Better for Microservices

Same Transport and Payloads

Easy To Migrate To

TRAILHEAD
TECHNOLOGY PARTNERS

Thanks! Questions?

Jonathan "J." Tower

Microsoft MVP in .NET

jtower@trailheadtechnology.com

trailheadtechnology.com/blog

jtowermi

jtower

Free

Consultation

bit.ly/th-offer

github.com/trailheadtechnology/minimal-apis