Working with Minimal API in ASP.NET Core Web API

# 1. What is Minimal API?

- A lightweight alternative to MVC/Web API controllers.  
- Uses a fluent style to define routes and handlers directly in Program.cs.  
- Best for small services, microservices, and quick APIs.  
- Still supports DI (Dependency Injection), filters, middleware, authentication, etc.

# 2. Creating a Minimal API

Instead of creating controllers, define endpoints in Program.cs:

var builder = WebApplication.CreateBuilder(args);  
var app = builder.Build();  
  
app.MapGet("/", () => "Hello World!");  
  
app.MapGet("/greet/{name}", (string name) => $"Hello, {name}!");  
  
app.Run();

# 3. Supported HTTP Methods

app.MapGet("/products", () => "GET: All Products");  
app.MapPost("/products", () => "POST: Create Product");  
app.MapPut("/products/{id}", (int id) => $"PUT: Update Product {id}");  
app.MapDelete("/products/{id}", (int id) => $"DELETE: Remove Product {id}");

# 4. Returning JSON Response

app.MapGet("/student", () => new { Id = 1, Name = "Vraj Suratwala" });

# 5. Using Dependency Injection (DI)

builder.Services.AddSingleton<IStudentService, StudentService>();  
  
app.MapGet("/students", (IStudentService service) =>  
{  
 return service.GetAllStudents();  
});

# 6. Model Binding & Validation

app.MapPost("/register", (User user) =>  
{  
 return Results.Ok($"User {user.Name} Registered!");  
});  
  
public record User(string Name, string Email);

# 7. Action Results

app.MapGet("/status/{id}", (int id) =>  
{  
 if (id == 1)  
 return Results.Ok(new { Message = "Valid ID" });  
 else  
 return Results.NotFound(new { Message = "Invalid ID" });  
});

# 8. Route Parameters & Query Parameters

// Route parameter  
app.MapGet("/product/{id}", (int id) => $"Product ID = {id}");  
  
// Query parameter  
app.MapGet("/search", (string? q) => $"You searched for {q}");

# 9. Middleware with Minimal API

app.Use(async (context, next) =>  
{  
 Console.WriteLine($"Request: {context.Request.Path}");  
 await next();  
});

# 10. Example: CRUD Minimal API with In-Memory List

var products = new List<string> { "Book", "Pen", "Bag" };  
  
app.MapGet("/products", () => products);  
  
app.MapPost("/products", (string product) =>  
{  
 products.Add(product);  
 return Results.Ok(products);  
});  
  
app.MapDelete("/products/{name}", (string name) =>  
{  
 products.Remove(name);  
 return Results.Ok(products);  
});

# Summary: Minimal API vs Controller

|  |  |  |
| --- | --- | --- |
| Feature | Minimal API 🚀 | MVC/Web API Controller 🏛 |
| Boilerplate Code | Very low | More (Controllers, Actions) |
| Best Use Case | Small APIs, microservices | Large apps, complex apps |
| Flexibility | Simple routing | Advanced features |
| Performance | Faster (lightweight) | Slightly heavier |