Towards the .NET Junior Developer

The extremely solid course

Lesson 10

Web applications development

Agenda

- REST methods
  - Idempotence
- ASP.NET Core
  - Design
  - Setup
  - Controllers
  - Middleware system
  - Swagger
  - DI
- Authentication and authorization
- Books of the day
- Links of the day
- Hometask

REST methods

REST methods

Method	Description
GET	Requires the existing resources from API
POST	Sends new resources to API
PUT	Replaces the resource by the new one
DELETE	Deletes the resource
PATCH	Updates the resource (only the necessary fields)
HEAD	Same as GET, but requires only headers
OPTIONS	Get the available methods for the chosen URI
TRACE	Traces the request that has been received by the web-server (for testing purposes)

Idempotence - GET

Id	NomenclatureNumber	Name	ProductTypeId	Description
344A15E8-2A71-4F0E-B246-087E7F4FFC31	0015-4326	Java Coffee	6B177375-279A-4267-86E3-0BE60C2D9D01	The best coffee (much better than programming la...

GET https://coffeestore.com/products/344A15E8-2A71-4F0E-B246-087E7F4FFC31

Status: 200 (OK)

Response content:

```
{
    "ID": "344A15E8-2A71-4F0E-B246-087E7F4FFC31",
    "NomenclatureNumber": "0015-4326",
    "Name": "Java Coffee",
    "ProductTypeId": "6B177375-279A-4267-86E3-0BE60C2D9D01",
    "Description": "The best coffee (much better than programming language)"
}
```

Server state hasn't change. ***GET request is idempotent***.

Idempotence - POST

Id	NomenclatureNumber	Name	ProductTypeId	Description
344A15E8-2A71-4F0E-B246-087E7F4FFC31	0015-4326	Java Coffee	6B177375-279A-4267-86E3-0BE60C2D9D01	The best coffee (much better than programming la...

POST https://coffeestore.com/products

```
{
    "ID": "88A11753-DC9D-4899-A7E9-BD0EA580F25A",
    "NomenclatureNumber": "0001-1200",
    "Name": "Sunshine Mexico",
    "ProductTypeId": "0C57276D-022F-4983-A175-550E19B17318",
    "Description": "Mexican Arabica with the fruit notes"
}
```

Status: 200 (OK)

Server state has changed (new product has been created). ***POST request is not idempotent***.

Idempotence - POST

Id	NomenclatureNumber	Name	ProductTypeId	Description
344A15E8-2A71-4F0E-B246-087E7F4FFC31	0015-4326	Java Coffee	6B177375-279A-4267-86E3-0BE60C2D9D01	The best coffee (much better than programming la...

POST https://coffeestore.com/products

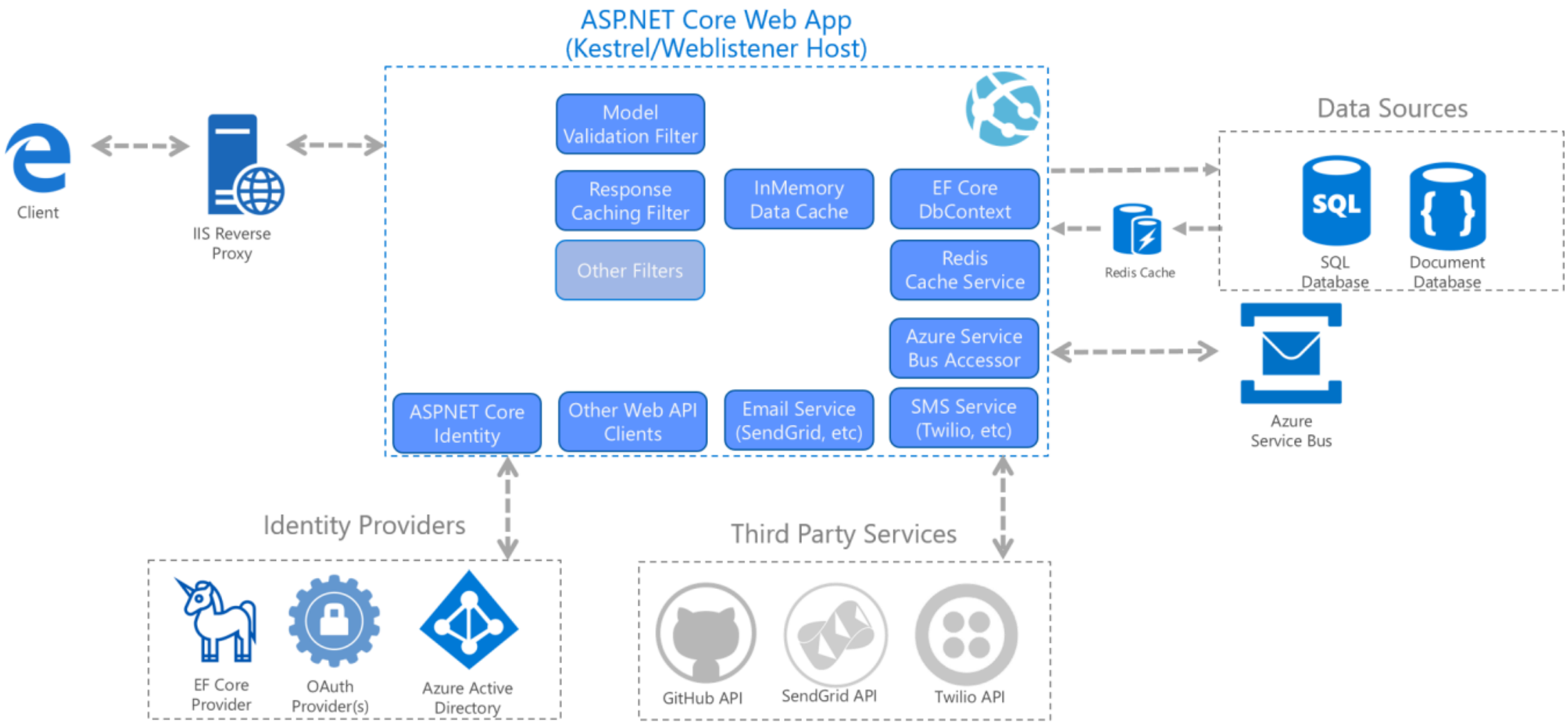
```
{
    "ID": "88A11753-DC9D-4899-A7E9-BD0EA580F25A",
    "NomenclatureNumber": "0001-1200",
    "Name": "Sunshine Mexico",
    "ProductTypeId": "0C57276D-022F-4983-A175-550E19B17318",
    "Description": "Mexican Arabica with the fruit notes"
}
```

Status: 200 (OK)

Server state has changed (new product has been created). ***POST request is not idempotent***.

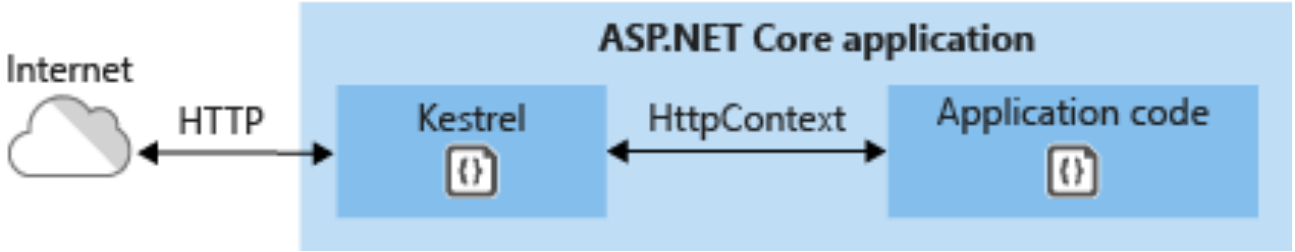
ASP.NET Core

ASP.NET Core Architecture



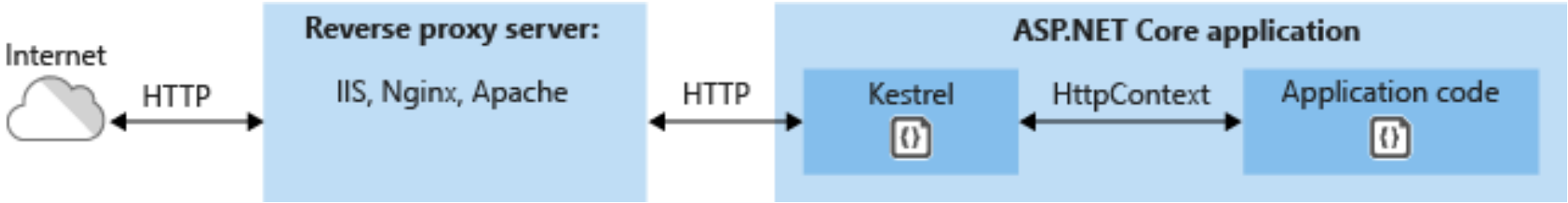
Design

Kestrel as ***edge*** server

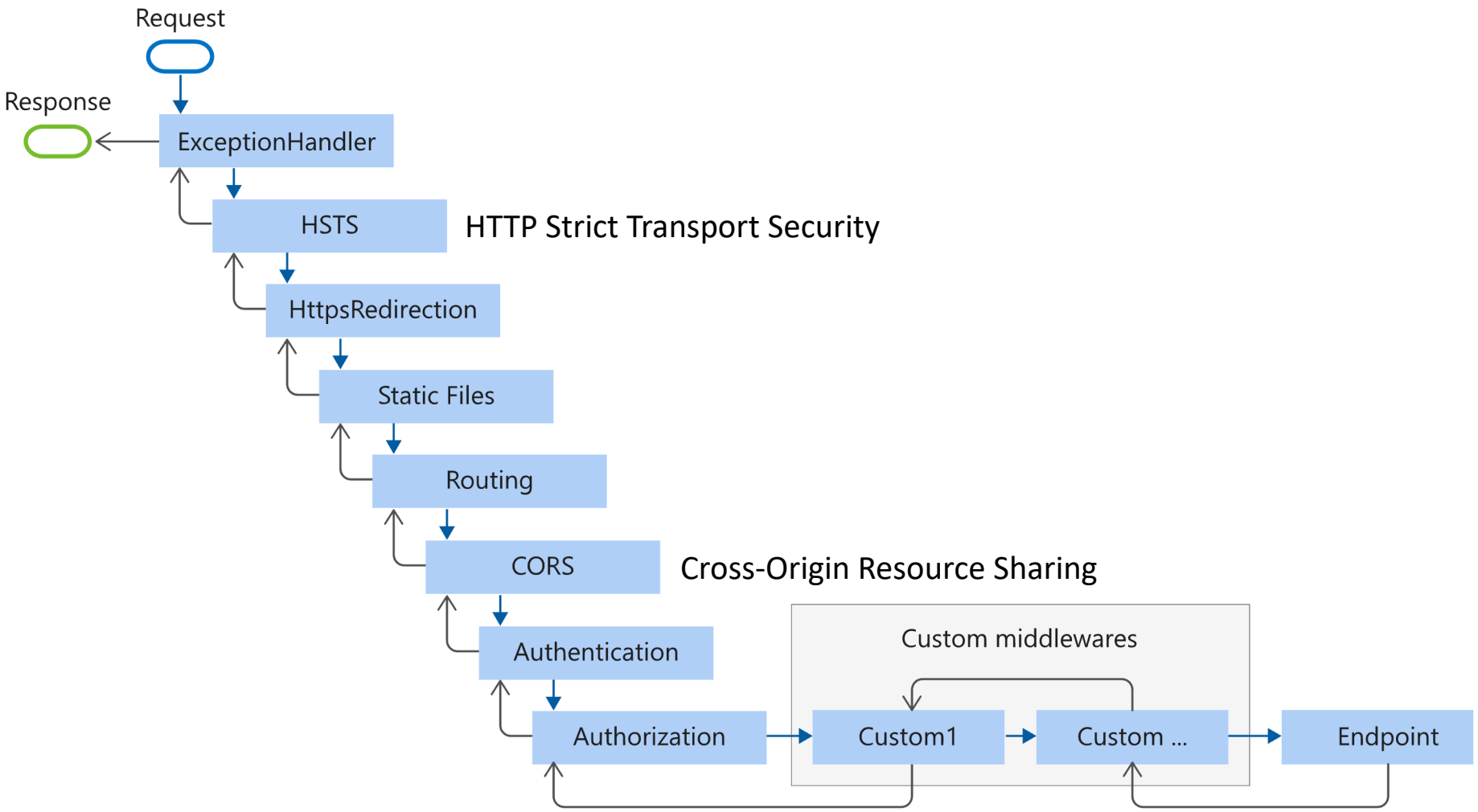


Design

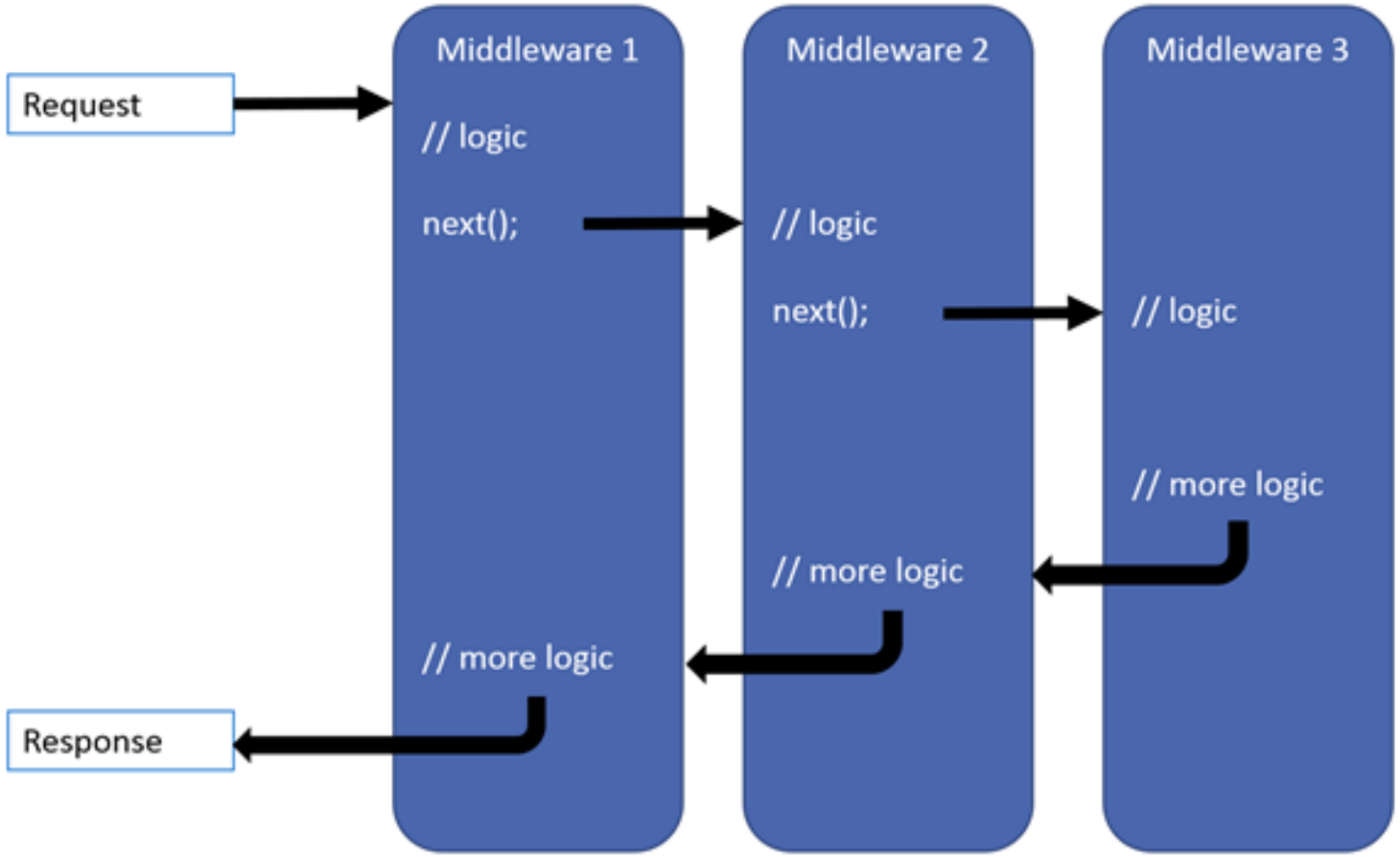
Kestrel with reverse-proxy server



Design


Request
Response
ExceptionHandler
HSTS
HTTP Strict Transport Security
HttpsRedirection
Static Files
Routing
CORS
Cross-Origin Resource Sharing
Authentication
Custom middlewares
Authorization
Custom1
Custom ...
Endpoint

Middleware system



Setup

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

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Controllers

```
namespace WebApplicationExample.Controllers
{
    [ApiController]
    [Route("[controller]")]
    3 references
    public class WeatherForecastController : ControllerBase
    {
        private static readonly string[] Summaries = new[]
        {
        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
    };

        private readonly ILogger<WeatherForecastController> _logger;

        0 references
        public WeatherForecastController(ILogger<WeatherForecastController> logger)
        {
            _logger = logger;
        }

        [HttpGet(Name = "GetWeatherForecast")]
        0 references
        public IEnumerable<WeatherForecast> Get()...
    }
}
```

Controllers

```
[HttpGet(Name = "GetWeatherForecast")]
0 references
public IEnumerable<WeatherForecast> Get()
{
    return Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = Summaries[Random.Shared.Next(Summaries.Length)]
    })
    .ToArray();
}
```

Swagger

Swagger
Supported by SMARTBEAR

Select a definition WebApplicationExample v1

WebApplicationExample 1.0 OAS3

https://localhost:7029/swagger/v1/swagger.json

WeatherForecast

GET /WeatherForecast

Parameters

Cancel

No parameters

Execute

Responses

Code	Description	Links
200	Success Media type text/plain Controls Accept header. Example Value | Schema	No links

```
[
  {
    "date": "2022-08-29T10:45:13.592Z",
    "temperatureC": 0,
    "temperatureF": 0,
    "summary": "string"
  }
]
```

DI

```
[HttpGet(Name = "GetWeatherForecast")]
0 references
public IEnumerable<WeatherForecast> Get()
{
    return Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = DateTime.Now.AddDays(index),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = Summaries[Random.Shared.Next(Summaries.Length)]
    })
    .ToArray();
}
```

DI

```
namespace WebApplicationExample
{

    public interface IWeatherForecastService
    {

        WeatherForecast[] GetForecasts();
    }
}

public class WeatherForecastService : IWeatherForecastService
{
    private static readonly string[] Summaries = new[]
    {
        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
    };


    public WeatherForecast[] GetForecasts()
    {
        return Enumerable.Range(1, 5).Select(index => new WeatherForecast
        {
            Date = DateTime.Now.AddDays(index),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = Summaries[Random.Shared.Next(Summaries.Length)]
        })
        .ToArray();
    }
}
```

DI

```
// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

// Dependency Injection
builder.Services.AddScoped<IWeatherForecastService, WeatherForecastService>();

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

DI

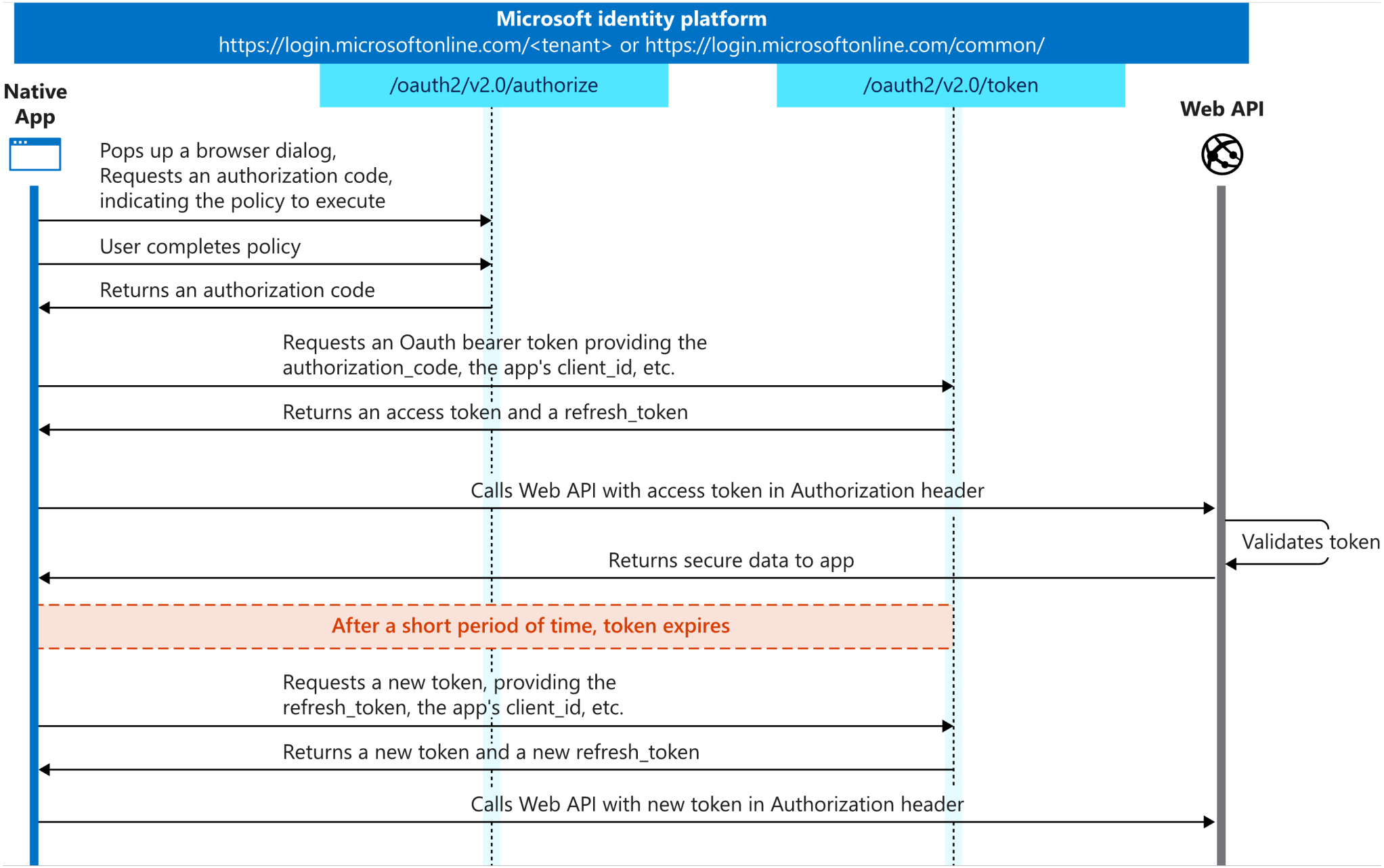
```
[ApiController]
[Route("[controller]")]
3 references
public class WeatherForecastController : ControllerBase
{
    private readonly ILogger<WeatherForecastController> _logger;
    private readonly IWeatherForecastService _weatherForecastService;

    0 references
    public WeatherForecastController(
        ILogger<WeatherForecastController> logger,
        IWeatherForecastService weatherForecastService)
    {
        _logger = logger;
        _weatherForecastService = weatherForecastService;
    }

    [HttpGet(Name = "GetWeatherForecast")]
    0 references
    public IEnumerable<WeatherForecast> Get()
    {
        return _weatherForecastService.GetForecasts();
    }
}
```

Authentication and authorization

Authentication vs Authorization

Authentication – ***who*** is this user?

Authorization – I ***know*** this user, but ***does it have the permissions*** for this operation?


Microsoft identity platform
https://login.microsoftonline.com/<tenant> or https://login.microsoftonline.com/common/
/oauth2/v2.0/authorize
/oauth2/v2.0/token
Native App
Web API
Pops up a browser dialog,
Requests an authorization code,
indicating the policy to execute
User completes policy
Returns an authorization code
Requests an Oauth bearer token providing the
authorization_code, the app's client_id, etc.
Returns an access token and a refresh_token
Calls Web API with access token in Authorization header
Validates token
Returns secure data to app
After a short period of time, token expires
Requests a new token, providing the
refresh_token, the app's client_id, etc.
Returns a new token and a new refresh_token
Calls Web API with new token in Authorization header

Encoded


```
eyJhbGciOiJIUzI1NiIsInR5c
CI6IkpXVCJ9.eyJzdWIiOiIxM
jM0NTY3ODkwIiwibmFtZSI6Ik
pvaG4gRG9lIiwiYWRtaW4iOnR
ydWV9.TJVA95OrM7E2cBab30R
MHrHDcEfxjoYZgeFONFh7HgQ
```


Decoded

```
{
  "alg": "HS256",
  "typ": "JWT"
}
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
)
```

Header

Payload

Signature

https://artsy.github.io/blog/2016/10/26/jwt-artsy-journey/

Authorization

- Role-based
- Resource-based
- Claims-based
- Policy-based
- View-based

Authorization – role-based

```
[Authorize(Roles = "HRManager,Finance")]
public class SalaryController : Controller
{
    public IActionResult Payslip() =>
                        Content("HRManager || Finance");
}
```

User has **HRManager** *or* **Finance** role

```
[Authorize(Roles = "PowerUser")]
[Authorize(Roles = "ControlPanelUser")]
public class ControlPanelController : Controller
{
    public IActionResult Index() =>
        Content("PowerUser && ControlPanelUser");
}
```

User has **PowerUser** *and* **ControlPanelUser** role

Authorization – policy-based

```
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("AtLeast21", policy =>
        policy.Requirements.Add(new MinimumAgeRequirement(21)));
});
```

Step 1. Register ***AtLeast21*** policy

```
public class MinimumAgeRequirement : IAuthorizationRequirement
{
    public MinimumAgeRequirement(int minimumAge) =>
        MinimumAge = minimumAge;

    public int MinimumAge { get; }
}
```

Step 2. Add **requirement** class

Authorization – policy-based

```
public class MinimumAgeHandler : AuthorizationHandler<MinimumAgeRequirement>
{
    protected override Task HandleRequirementAsync(
        AuthorizationHandlerContext context, MinimumAgeRequirement requirement)
    {
        var dateOfBirthClaim = context.User.FindFirst(
            c => c.Type == ClaimTypes.DateOfBirth && c.Issuer == "http://contoso.com");

        if (dateOfBirthClaim is null)
        {
            return Task.CompletedTask;
        }

        var dateOfBirth = Convert.ToDateTime(dateOfBirthClaim.Value);
        int calculatedAge = DateTime.Today.Year - dateOfBirth.Year;
        if (dateOfBirth > DateTime.Today.AddYears(-calculatedAge))
        {
            calculatedAge--;
        }

        if (calculatedAge >= requirement.MinimumAge)
        {
            context.Succeed(requirement);
        }

        return Task.CompletedTask;
    }
}
```

Step 3. Create **handler** for this type of requirement

Authorization – policy-based

```
builder.Services.AddSingleton<IAuthorizationHandler, MinimumAgeHandler>();
```

Step 4. **Register** handler in the middleware pipeline

Books of the day

Lock A. – ASP.NET Core in Action

Freeman A. – Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages

Links of the day

Authentication (MSDN)

Authorization (MSDN)

Authorization in ASP.NET Core MVC (Habr)

ASP.NET Core articles (C-SharpCorner)

That’s all for this time!