

# ASP.NET Web API 2

Albert István



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Tartalom

## Szolgáltatás leírás

- Swagger 1.2 -> OpenAPI 2.0
  - > Specifikáció: [github.com/OAI/OpenAPI-Specification](https://github.com/OAI/OpenAPI-Specification)
- Json formátum (vagy bináris yaml)
- Szolgáltatás leírása
  - > Alap típusok: például dátum, idő, lebegőpontos
  - > Műveletek paraméterek, kényszerek
  - > Válasz objektumok típusai
  - > Biztonsági előírások
  - > ...



Szoftverfejlesztés .NET platformra

## Gyorsítótárazás (cache)

- A teljesítmény növelésének leghatékonyabb módja
- Az egyik legnehezebb feladat lehet jól csinálni



## Válasz gyorsítótárazása

- HTTP cache fejléc információk állítása
- Gyorsítótárazás
  - > Kliensen
  - > Proxy szerveren
- ResponseCache attribútum [Cache-control]
  - > Duration [max-age]
  - > Location [private / public / no-cache]
    - Public: kliensen, proxyon
    - Private: csak kliensen
  - > Profile



Szoftverfejlesztés .NET platformra

## Naplózás

- Tetszőleges naplózó komponens beinjektálható...

```
// This method gets called by the runtime. Use this method to configure
// references | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
    ILoggerFactory loggerFactory)
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();
}
```

- ... felhasználható

```
[HttpGet("GetLongText")]
// references | 0 exceptions
public string GetLongText()
{
    logger.LogInformation("**** GetLongText()");
    return "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."
}
```

```
public class ProductsController : Controller
{
    IMemoryCache cache;
    IDistributedCache dcache;
    ILogger logger;

    // references | 0 exceptions
    public ProductsController(IMemoryCache cache, IDistributedCache dcache, ILogger logger)
    {
        this.cache = cache;
        this.dcache = dcache;
        this.logger = logger;
    }
}
```

## Kivétel kezelés

- Kivételes esetekre
  - > Nem elérhető az adatbázis szerver
  - > Inkonzisztens input, pl rossz token
- Normál üzleti folyamatokra készülünk fel
  - > Van már ilyen nevű felhasználó az adatbázisban
  - > Rossz email cím formátum
- A szolgáltatás interfész tartalmazza ezeket az eseteket!



Szoftverfejlesztés .NET platformra

# Kérdések?

Albert István  
[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)

# Szolgáltatás leírás

- Swagger 1.2 -> OpenAPI 2.0
  - > Specifikáció: [github.com/OAI/OpenAPI-Specification](https://github.com/OAI/OpenAPI-Specification)
- Json formátum (vagy bináris yaml)
- Szolgáltatás leírása
  - > Alap típusok: például dátum, idő, lebegőpontos
  - > Műveletek paraméterek, kényszerek
  - > Válasz objektumok típusai
  - > Biztonsági előírások
  - > ...

# OpenAPI – automatikus dokumentáció

- NuGet csomag: `Install-Package Swashbuckle.AspNetCore -Pre`

```
public void ConfigureServices(IServiceCollection services)
{
    // bővítő metódus
    // a swagger doksi generáló szolgáltatás bejegyzése
    services.AddSwaggerGen(c => c.SwaggerDoc("v1",
        new Info { Title = "Web API Sample", Version = "v1" }));
}
```

```
public void Configure(IApplicationBuilder app, IHostingEnvironment
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();

    // bővítő metódus
    // routing a swagger.json fájl felé
    app.UseSwagger();

    // bővítő metódus
    // html, css UI a kérés manuális összeállításához
    app.UseSwaggerUI(c => c.SwaggerEndpoint(
        "/swagger/v1/swagger.json", "Web API Sample v1"));
}
```

# A nyers OpenAPI json

```
{
  "swagger": "2.0",
  "info": {
    "version": "v1",
    "title": "Web API Sample"
  },
  "basePath": "/",
  "paths": {
    "/api/Products/ByPrice/{minprice}/{maxprice}": {
      "get": {
        "tags": [
          "Products"
        ],
        "operationId": "ApiProductsByPriceByMinpriceByMaxpriceGet",
        "consumes": [

        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "minprice",
            "in": "path",
            "required": true,
            "type": "integer",
```

# Swagger UI - debuggoláshoz

The screenshot displays the Swagger UI interface for a 'Web API Sample'. At the top, a green header bar contains the Swagger logo and the URL 'http://localhost:50884/swagger/v1/swagger.json'. Below this, the title 'Web API Sample' is shown on the left, and a browser address bar on the right displays 'localhost:50884/swagger/'.

The interface lists two main API groups: 'Products' and 'Values'. Each group has a 'Show/Hide' and 'List' link. The 'Products' group contains two endpoints:

- GET /api/Products/ByPrice/{minprice}/{maxprice}
- GET /api/Products/ByName/{name}

The 'Values' group contains five endpoints:

- GET /api/Values
- POST /api/Values
- DELETE /api/Values/{id}
- GET /api/Values/{id}
- PUT /api/Values/{id}

At the bottom, a status bar indicates '[ BASE URL: / , API VERSION: v1 ]'.

# Swagger UI – API elérése

Response Content Type

## Parameters

Parameter	Value	Description	Parameter Type	Data Type
<b>minprice</b>	<input type="text" value="1"/>		path	integer
<b>maxprice</b>	<input type="text" value="30"/>		path	integer

Try it out!

[Hide Response](#)

## Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:50884/api/Products/ByPrice/1/30'
```

## Request URL

```
http://localhost:50884/api/Products/ByPrice/1/30
```

## Response Body

```
[
  {
    "unitPrice": 12,
    "name": null
  }
]
```

## Response Code



# Swagger – XML megjegyzésekkel

```
/// <summary>  
/// Visszaadja a termékeket ár szerint szűrve.  
/// </summary>  
/// <param name="minprice">Minimális ár, legalább 1, legfeljebb 100.</param>  
/// <param name="maxprice">Maximális ár.</param>  
/// <returns>Ár szerint szűrt termék lista.</returns>  
[HttpGet("ByPrice/{minprice:int:max(100):min(1)}/{maxprice}")]  
0 references  
public IEnumerable<Product> GetProduct(int minprice, int maxprice)
```

```
{  
    Title = "Web API Sample",  
    Version = "v1",  
    Description = "Egyszerű minta API a .NET tárgyhoz",  
    TermsOfService = "N/A",  
    Contact = new Contact  
    {  
        Name = "Albert István",  
        Email = "ialbert@aut.bme.hu",  
        Url = "http://www.aut.bme.hu"  
    }  
});  
services.ConfigureSwaggerGen(o =>  
    o.IncludeXmlComments(  
        $"{environment.ContentRootPath}\\bin\\Debug\\netcoreapp1.1\\WebApiSample2.xml"
```

Output

Output path:

bin\\Debug\\netcoreapp1.1\\

☒ XML documentation file:

bin\\Debug\\netcoreapp1.1\\WebApiSa

# Swagger – XML megjegyzésekkel

## Products

[Show/Hide](#)[List Operations](#)[Expand Operations](#)

GET

</api/Products/ByPrice/{minprice}/{maxprice}>[Visszaadja a termékeket ár szerint szűrve.](#)

### Response Class (Status 200)

Success

[Model](#) [Example Value](#)

#### Inline Model [

Inline Model 1

]

#### Inline Model 1 {

`unitPrice (integer, optional),``name (string, optional)`

}

Response Content Type 

### Parameters

Parameter	Value	Description	Parameter Type	Data Type
minprice	<input type="text" value="(required)"/>	Minimális ár, legalább 1, legfeljebb 100.	path	integer
maxprice	<input type="text" value="(required)"/>	Maximális ár.	path	integer

# Kód generálás

- Az OpenAPI specifikáció alapján kliens kód generálása
  - > C#, TypeScript, Java, ...
- Több eszköz
  - > AutoRest, NSwag, ...
- Több módszer
  - > Manuális, command line
  - > MSBuild task – a build folyamat része
  - > T4 kódgenerátor

# NSwagStudio - kódgenerálás

## Input: Swagger Specification

Swagger Specification Web API Assembly JSON Schema .NET Assembly

### Swagger Specification URL:

`http://petstore.swagger.io/v2/swagger.json`

Create local Copy

### Swagger Specification JSON (if specified)

```
1: "type": "integer"}, "name": {"t
```

## Outputs

☐ TypeScript Client ☒ CSharp Client ☐ CSharp Web API Controller (experimental)

Swagger Specification CSharp Client

Settings

### CSharp Client Settings

#### Namespace

`SampleService`

☐ Generate contracts output

#### Additional Namespace Usages (comma separated)

Output

## Outputs

☐ TypeScript Client ☒ CSharp Client

Swagger Specification CSharp Client

Settings

```
34  
35  
36 · ApiProductsByPriceByMinpriceByMaxpriceGetAsync(int minprice, int maxprice)  
37  
38 Cancellation.Token.None);  
39
```

# Gyorsítótárazás (cache)

- A teljesítmény növelésének leghatékonyabb módja
- Az egyik legnehezebb feladat lehet jól csinálni

# Típusai

- Nem elosztott – egy gépen van
  - > Egyszerű, ha egyetlen kiszolgáló van felskálázva
  - > Webfarm esetén kliens affinitás
    - Alkalmazás szintű információk alapján, pl munkamenet
- Elosztott
  - > Szinkronizáció, konzisztencia
  - > Külső processben fut
  - > Csak byte[]-t kezel, a sorosítás az alkalmazás feladata

# IMemoryCache interfész

- **CreateEntry**: új entitás vagy felülírja a meglévőt
  - > Egy CacheEntry objektummal tér vissza
- **Remove**: törli az elemet
- **TryGetValue**: kiolvassa az elemet

```
[HttpGet("GetTime1")]
0 references | 🟢 22 requests | 0 exceptions
public string GetTime1()
{
    DateTime time;
    ...
    if (!cache.TryGetValue("time", out time))
    {
        time = DateTime.Now;
        using (var e = cache.CreateEntry("time"))
        {
            e.Value = time;
            e.AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(5);
        }
    }
    return time.ToString();
}
```

# Bekapcsolása

- Szolgáltatás => Dependency Injection

```
// This method gets called by the runtime. Use this method  
0 references | 0 exceptions  
public void ConfigureServices(IServiceCollection services)  
{  
    services.AddMemoryCache();  
}
```

- Constructor injection

```
IMemoryCache cache;  
  
0 references | 0 exceptions  
public ProductsController(IMemoryCache cache)  
{  
    this.cache = cache;  
}
```



# Bővítő metódusok

- A nyers interfész felett az egyszerűbb használathoz
- Típusos
  - Get<T>
  - GetOrCreate <T>
  - Set <T>
  - TryGetValue<T>

# Cache használata 1

2 references

```
enum CacheKeys
```

```
{  
    Time,  
}
```

```
[HttpGet("GetTime2")]
```

0 references | 0 requests | 0 exceptions

```
public string GetTime2()
```

```
{  
    DateTime time;  
    if (!cache.TryGetValue(CacheKeys.Time, out time))  
    {  
        time = DateTime.Now;  
        cache.Set(CacheKeys.Time, time, TimeSpan.FromSeconds(5));  
    }  
    return time.ToString();  
}
```

# Cache használata 2

- Egyszerre több szál is inicializálhatja az objektumot!

```
[HttpGet("GetTime3")]
```

```
0 references | ✔ 23 requests | 0 exceptions
```

```
public string GetTime3()
```

```
{
```

```
    DateTime time = cache.GetOrCreate(CacheKeys.Time, e =>
```

```
{
```

```
    e.AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(5);
```

```
    return DateTime.Now;
```

```
});
```

```
    return time.ToString();
```

```
}
```

# Tárolási opciók

- Abszolút lejárati idő – lásd a példát
- Csúszó idő: az utolsó használat után eltelt idő
- Beállítható egyszerre mindkettő!
  - > 1 perc csúszó idő
  - > De egy óra múlva mindenképp frissítse be a szerver
- Explicit lejárati CancellationTokennel
- Prioritás
  - > Alacsony, közepes, magas, sose törölje
- Függőségek
- Callback törlés esetén

# Elosztott cache

- Előnyei
  - > Minden gépen ugyanazt látják a web szerverek
  - > A webszerver újraindítása nem befolyásolja a cache tartalmát így a farm teljesítményét
  - > Összességében kevesebb elérés a forrás adatbázis felé
- IDistributedCache interfész
  - > Get, Set, Remove, Refresh + Async, byte [] / string
- Beépített implementációk
  - > Redis
  - > SQL Server Distributed Cache

# Elosztott cache használata

- Csúszó és abszolút lejárati idők

```
[HttpGet("GetTime4")]
```

```
0 references | ✔ 23 requests | 0 exceptions
```

```
public string GetTime4()
```

```
{
```

```
    var time = dcache.GetString("time");
```

```
    if( time == null )
```

```
{
```

```
        time = DateTime.Now.ToString();
```

```
        dcache.SetString("time", time,
```

```
            new DistributedCacheEntryOptions
```

```
            { AbsoluteExpirationRelativeToNow
```

```
                = TimeSpan.FromSeconds(5) }));
```

```
    }
```

```
    return time;
```

```
}
```

# Válasz gyorsítótárázása

- HTTP cache fejléc információk állítása
- Gyorsítótárázás
  - > Kliensen
  - > Proxy szerveren
- ResponseCache attribútum [Cache-control]
  - > Duration [max-age]
  - > Location [private / public / no-cache]
    - Public: kliensen, proxyn
    - Private: csak kliensen
  - > Profile

# Kliens oldali cache-elés

```
[HttpGet("GetTime5")]  
[ResponseCache(Location = ResponseCacheLocation.Any, Duration = 5)]  
0 references | ✓ 11 requests | 0 exceptions  
public string GetTime5()  
{  
    return DateTime.Now.ToString();  
}
```

Status: 200 OK    Time: 57 ms    Size: 331 B

Body    Cookies    Headers (7)    Tests

Cache-Control → public,max-age=5

Content-Type → application/json; charset=utf-8



# Cache profilok

- Összevont cache beállítások akár controller szinten
- Profilonként más alapértelmezett értékek
  - > Egyesével helyben testreszabható

```
// Add framework services.  
services.AddMvc(o =>  
    o.CacheProfiles.Add("Default", new CacheProfile { Duration = 60 }));
```

```
[HttpGet("GetTime6")]  
[ResponseCache(CacheProfileName = "Default")]  
0 references | ✓ 1 request | 0 exceptions  
public string GetTime6()  
{  
    return DateTime.Now.ToString()  
}
```

Status: 200 OK   Time: 104 ms   Size: 332 B

Body   Cookies   Headers (7)   Tests

Cache-Control → public,max-age=60

# Mikor NE cache-eljünk kliens oldalon?

- Csak azonosított felhasználó számára szóló információ
- Azonosított felhasználótól függő tartalom esetén

# Szerver oldali cache-elés

- Ugyanúgy a ResponseCache attribútum
- nuGet Microsoft.AspNetCore.ResponseCaching
- Inicializálás

```
// This method gets called by the runtime. Use  
0 references | 0 exceptions  
public void ConfigureServices(IServiceCollection  
{  
    services.AddResponseCaching(o=>  
        o.MaximumBodySize = 100*1024  
    );  
}
```

```
// This method gets called by the runtime. Use  
0 references | 0 exceptions  
public void Configure(IApplicationBuilder app,  
{  
    app.UseResponseCaching();  
}
```

# Cache paraméterek

- Location: Any (NEM Client)

```
[HttpGet("GetTime6")]
```

```
[ResponseCache(Location = ResponseCacheLocation.Any, CacheProfileName = "Default")]
```

0 references | 0 requests | 0 exceptions

```
public string GetTime6()
```

```
{
```

```
    return DateTime.Now.ToString();
```

```
}
```

- Kérés fejléc: Cache-Control

GET ▾

http://localhost:50884/api/Products/gettime6

Auth Headers (1) Body Pre-req. Tests

Key

Value



Cache-Control

max-age=60

# Paraméterek figyelembe vétele

- VaryByQueryKeys: megkülönböztetés query kulcsok alapján

```
[HttpGet("GetTime6")]
[ResponseCache(Location = ResponseCacheLocation.Any,
    VaryByQueryKeys = new[] { "p" } )]
0 references | 3 requests | 0 exceptions
public string GetTime6()
{
    return DateTime.Now.ToString();
}
```

GET	http://localhost:50884/api/Products/gettime6?p=1	
Auth	Headers (1)	Body
	Key	Value
<input checked="" type="checkbox"/>	Cache-Control	max-age=60

# Tömörítés

- nuGet ...AspNetCore.ResponseCompression
- Opciók
  - > Tömörítés erőssége
  - > MIME típusok

```
// This method gets called by the runtime  
0 references | 0 exceptions  
public void ConfigureServices(IServiceCollection Services)  
{  
    services.AddResponseCompression();  
}
```

```
// This method gets called by the runtime  
0 references | 0 exceptions  
public void Configure(IApplicationBuilder app)  
{  
    app.UseResponseCompression();  
}
```

# Naplózás

- Tetszőleges naplózó komponens beinjektálható...

```
// This method gets called by the runtime. Use this method to configure
0 references | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
    ILoggerFactory loggerFactory)
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();
}
```

- ... felhasználható

```
[HttpGet("GetLongText")]
0 references | ✓ 1 request | 0 exceptions
public string GetLongText()
{
    logger.LogInformation("*** GetLongText");
    return "Lorem ipsum dolor sit amet,
}
```

```
public class ProductsController : Controller
{
    IMemoryCache cache;
    IDistributedCache dcache;
    ILogger logger;

    0 references | 0 exceptions
    public ProductsController(IMemoryCache cache, IDistributedCache dcache,
        ILogger<ProductsController> logger)
    {
        this.cache = cache;
        this.dcache = dcache;
        this.logger = logger;
    }
}
```

# Napló kategória

- Tipikusan az osztály neve
  - > Generikus típusparaméter
- Bejegyzés szintje: LogLevel
  - > Critical, Error, Warning, Information, Debug , Trace
  - > Attól függően, mennyire hosszú távú tartalma van az üzenetnek
- Azonosító: Event ID



# Szemantikus naplózás

- Struktúrált adatok
- A napló bejegyzés típusától függ a struktúra
- Jobban kereshető, indexelhető mint az egyszerű szöveg
- Például NLog

```
logger.ExtendedError("Could not contact customer", new { CustomerId = 1234, HttpStatusCode = 404 } );
```

---

```
{"TimeStamp":"2016-09-21T08:11:23.483Z","Level":"Info","LoggerName":"Acme.WebApp.OrderController",  
"Message":"Order resent to partner","CallSite":"Acme.WebApp.OrderController.ResendOrder",  
"OrderId":"1234","PartnerId":"4567",  
"NewState":"Sent","SendDate":"2016-09-21T08:11:23.456Z"}
```

# Szűrés

- Bejegyzés szintje alapján
  - > Például csak Warningokat és fontosabbakat
- Kategória alapján
- Providerenként vagy az összes provider szűrhető egyszerre
  - > LoggerFactory . WithFilter ...

# Szkóp

- A szkópon belül minden üzenet kiegészíthető további információval, például CorrelationID stb.

```
// This method gets called by the runtime. Use this method to configure
0 references | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
    ILoggerFactory loggerFactory)
{
    loggerFactory
        .WithFilter(new FilterLoggerSettings
        {
            { "Microsoft", LogLevel.Warning },
            { "System", LogLevel.Warning }
        })
        .AddConsole(includeScopes: true)
        .AddDebug();
}
```

```
[HttpGet("LongQuery")]
0 references | ✓ 1 request | 0 exceptions
public string LongQuery()
{
    using (logger.BeginScope("correlation id: 101"))
    {
        logger.LogInformation("*** GetLongText method is called")
        return "Lorem ipsum dolor sit amet, consectetur adipiscing elit."
    }
}
```

# Beépített providerek

- Console
- Debug - VS
- EventSource - ETW
- EventLog
- TraceSource
  - > TextWriterTraceListener
- AzureAppService

# Javaslatok

- Ügyelni kell az érzékeny információk naplózására!
- Fontos az üzemeltetőnek!
- Kivételek: minél több információt
  - > Ne csak a Message tulajdonságot!
  - > A hívás korábbi eseményeit is...

# Konfiguráció

- Hierarchikus kulcs-érték párok
  - > Kulcs: szöveg
  - > Érték: lehet sorosított objektum
- Forrás: provider alapú, több is lehet egyszerre
  - > Fájlok: json, xml, ini
  - > Parancssor argumentumok
  - > Környezeti változók
  - > Titkosított tár
  - > Azure Key Vault
  - > ...

# Json fájl

- Aktuális sablon – appsettings.json

0 references | 0 exceptions

```
public Startup(IHostingEnvironment env)
{
    this.environment = env;

    var builder = new ConfigurationBuilder()
        .SetBasePath(env.ContentRootPath)
        .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
        .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
        .AddEnvironmentVariables();
    Configuration = builder.Build();
}
```

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "MaxOrderQueue": 64
}
```

# Felhasználás

- Dependency injection: Configuration

2 references | 0 exceptions

```
public IConfigurationRoot Configuration { get; }
```

```
// This method gets called by the runtime. Use this method
```

0 references | 0 exceptions

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IConfiguration>(Configuration);
}
```

```
IConfiguration config;
```

0 references | 0 exceptions

```
public ProductsController(IMemoryCache cache, IDistributedCache
    ILogger<ProductsController> logger, IConfiguration config)
{
    this.cache = cache;
    this.dcache = dcache;
    this.logger = logger;
    this.config = config;
}
```

```
[HttpGet("GetConfigParam")]
```

0 references | ✓ 1 request | 0 exceptions

```
public string GetConfigParam()
{
    return config["MaxOrderQueue"];
}
```



# Típusos olvasás

- A config fájl tetszőleges elemét tudjuk olvasni
- Beépített konverzió, alapértelmezett érték
- Tömbök címkése: ":%tömbindex%:" szintaktikával

```
"OrderQueue": {  
    "MaxQueueSize": 32,  
    "RemoteQueueName": "RemoteDebugQueue12"  
}
```

```
[HttpGet("GetConfigParam3")]  
0 references | 0 requests | 0 exceptions  
public string GetConfigParam3()  
{  
    int p = config.GetValue<int>("OrderQueue:MaxQueueSize");  
    return p.ToString();  
}
```

# Típusos konfiguráció 1

- Saját konfigurációs osztály

```
public class OrderQueueConfig
{
    0 references | 0 exceptions
    public int MaxQueueSize { get; set; } = 1;
    1 reference | 0 exceptions
    public string RemoteQueueName { get; set; } = "DefaultQueue";
}
```

```
[HttpGet("GetConfigParam2")]
0 references | ✔ 1 request | 0 exceptions
public string GetConfigParam2()
{
    return queueConfig.RemoteQueueName;
}
```

# Típusos konfiguráció beállítása 1

- OrderQueue szekcióból olvas

```
// This method gets called by the runtime. Use this method
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddOptions();
    services.Configure<OrderQueueConfig>(
        Configuration.GetSection("OrderQueue"));
}
```

```
OrderQueueConfig queueConfig;

0 references | 0 exceptions
public ProductsController(IMemoryCache cache,
    ILogger<ProductsController> logger, ICon
    IOptions<OrderQueueConfig> queueConfig)
{
    this.queueConfig = queueConfig.Value;
    this.cache = cache;
    this.dcache = dcache;
    this.logger = logger;
    this.config = config;
}
```

```
"OrderQueue": {
    "MaxQueueSize": 32,
    "RemoteQueueName": "RemoteDebugQueue12"
}
```

# Típusos olvasás 2

- Egy szekció leképezése egy osztályra
- Get<T>, Bind metódusok

```
[HttpGet("GetConfigParam4")]  
0 references | 0 requests | 0 exceptions  
public string GetConfigParam4()  
{  
    return config.GetSection("OrderQueue")  
        .Get<OrderQueueConfig>().RemoteQueueName;  
}
```

```
"OrderQueue": {  
    "MaxQueueSize": 32,  
    "RemoteQueueName": "RemoteDebugQueue12"  
}
```

# Konfiguráció változás

- IOptions helyett IOptionsSnapshot objektumot kell injektálni, és reloadOnChange-t bekapcsolni

0 references | 0 exceptions

```
public ProductsController(IMemoryCache cache, IDistributedCache dcache,
    ILogger<ProductsController> logger, IConfiguration config,
    IOptionsSnapshot<OrderQueueConfig> queueConfig)
{
    this.queueConfig = queueConfig.Value;
    this.cache = cache;
    this.dcache = dcache;
    this.logger = logger;
    this.config = config;
}
```

0 references | 0 exceptions

```
public Startup(IHostingEnvironment env)
{
    this.environment = env;

    var builder = new ConfigurationBuilder()
        .SetBasePath(env.ContentRootPath)
        .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
```

# Kivétel kezelés

- Kivételes esetekre
  - > Nem elérhető az adatbázis szerver
  - > Inkonzisztens input, pl rossz token
- Normál üzleti folyamatokra készülünk fel
  - > Van már ilyen nevű felhasználó az adatbázisban
  - > Rossz email cím formátum
- A szolgáltatás interfész tartalmazza ezeket az eseteket!

# Globális kivételkezelés

- Több exception logger
- Egyetlen exception handler
- Mindkettő megkapja a kivétel kontextusát
  - > Kivétel
  - > Kérés, válasz
  - > Controller, Action
  - > ...

# Kivétel kezelése

- Alapértelmezett viselkedés: 500-as HTTP válasz
  - > Internal Server Error
- Speciális: `HttpResponseException`
  - > Megadható a státusz kód, például 404 stb.
  - > Saját szöveg.

```
public Product GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        var resp = new HttpResponseMessage(HttpStatusCode.NotFound)
        {
            Content = new StringContent(string.Format("No product with ID = {0}", id)),
            ReasonPhrase = "Product ID Not Found"
        }
        throw new HttpResponseException(resp);
    }
    return item;
}
```



# Kivételek a kliensen

- Státusz kódot meg kell vizsgálni
- A különböző státusz kódok és üzenetek részei a szolgáltatás interfésznek
- A kliens készülhet ezekre a státusz kódokra és kiírhat speciális üzenetet

# Kérdések?

Albert István  
[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)