

ASP.NET Web API 1

Albert István



Automatizálási és
Alkalmazott
Informatikai Tanszék

Tartalom

ASP.NET Core Web API

- Webes technológiákra, szabványokra épülő API
 - > HTTP protokoll (autentikáció stb)
 - > HTTP-képes és webes kliensek kiszolgálása
 - > JSON (vagy XML) üzenetek



Szoftverfejlesztés .NET platformra

Alapelvek

CONVENTION
CONFIGURATION

- Egyszerű
 - > Egyszerűen használható legyen (lightweight)
 - > Konvenció konfiguráció helyett (Convention-over-Configuration)
- Kiterjeszthető
- Tesztelhető
- Hordozható(bb)
 - > Többfajta hosztolási konfiguráció
- Nyílt forráskód



Szoftverfejlesztés .NET platformra

Routing

- A bejövő HTTP kérést melyik módszer szolgálja ki?
- Routing tábla alapján
- Leképezés: kérés adatai => controller action
- Az action paramétereknek is értéket kell adni
 - `ApiController.ControllerContext.RouteData.Values`
 - `HttpConfiguration.Routes`
- Konvenció alapú vagy attribútumos megadás



Szoftverfejlesztés .NET platformra

Model binding

- Hogyan lesz egy POST kérés adataiból függvényparaméter?
- Ökölszabály
 - > egyszerű típus => URI-ből (route dictionary, query string)
 - > komplex típus => MediaTypeFormatter-rel a kérés tartalmi részéből
- Megváltoztatható attribútumokkal
 - > FromUri: komplex típus esetén mégis az URI-ből vegye
 - > FromBody: egyszerű típus esetén mégis a Bodyből vegye

```
[HttpPost]
public void CreateTodoItem([FromBody] TodoItem item) {
    if (!ModelState.IsValid)
    {
        Context.Response.StatusCode = 400;
    }
}
```



Szoftverfejlesztés .NET platformra

Http Válasz előállítás

- HttpResponseMessage megy vissza a hálózaton
- 1. Ha a módszer HttpResponseMessage-et ad vissza, akkor pontosan azt küldi a kliensnek
- 2. Ha nincs visszatérési érték, akkor egy 204-es (No Content) kód megy vissza
- 3. Egyéb esetben a média típus formázó sorosítja az üzenetet és azt küldi vissza a kliensnek



Szoftverfejlesztés .NET platformra

Kliensek

- Kliens lehet bármi, ami tud HTTP valamilyen kvázi szabványos formátummal (XML, JSON) dolgozni
- JavaScript (böngészőben) – JQuery Ajax
- .NET, WinRT, Java, Python, C/C++ (libCurl)
- Android, iOS, Windows Phone



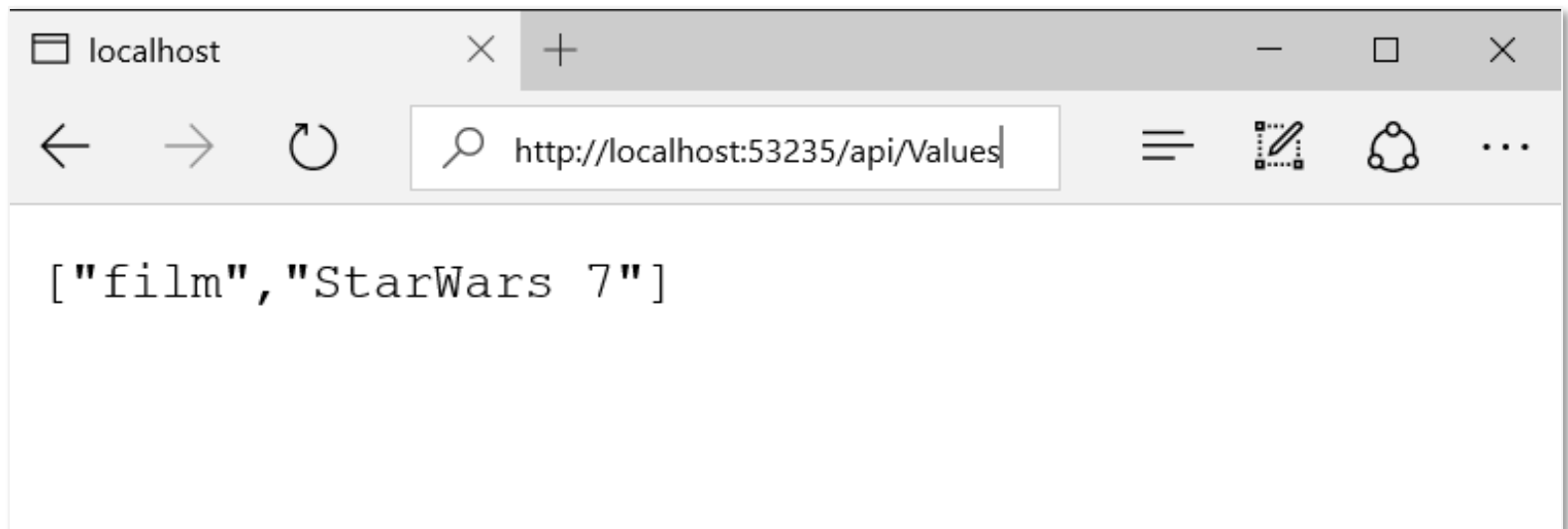
Szoftverfejlesztés .NET platformra

Kérdések?

Albert István
ialbert@aut.bme.hu

ASP.NET Core Web API

- Webes technológiákra, szabványokra épülő API
 - > HTTP protokoll (autentikáció stb)
 - > HTTP-képes és webes kliensek kiszolgálása
 - > JSON (vagy XML) üzenetek



Miért Web API?

- Különböző képességű kliensek kiszolgálása
 - > Kliensoldali követelmények
 - HTTP kliens
 - JSON parser
 - > Ma már a hűtőszekrény, villanykörte is tudja
 - > Mindenhol, ahol web van
 - > Ez a közös nevező!



REST

- REpresentational State Transfer
- Architektúrális stílus – nem szabvány!
 - > W3C csoport dolgozta ki
 - 2000, Roy Fielding disszertációja alapján
- Szabályok gyűjteménye elosztott architektúrákhoz
 - > adatközpontú, hypermedia szolgáltatások
- Szabályok betartva->RESTful
- Webes környezetre alkalmazható

REST és a Web

- Kliens-szerver szétválasztás (Client-server)
 - > Például: a kliens felé nyújtott interfész tárolási technológia független
- Állapotmentesség (Stateless)
 - > Web: a klasszikus webes architektúra
- Gyorsítótárazhatóság (Cacheable)
 - > Web: HTTP Cache-Control, Output Caching
- Köztes rétegek (Layered system)
 - > Web: proxy, load balancer
- Code on demand – opcionális
 - > Web: JavaScript küldése

REST és a web

- Egységes interfész elvek (Uniform Interface)
 - > Erőforrások azonosítása
 - Web: URI
 - > Erőforrások manipulálása
 - Web: HTTP igék
 - > Önleíró üzenetek
 - Művelet – Web: HTTP igék
 - Formátum – Web: Content negotiation
 - Gyorsítótárazhatóság – Web: Cache-Control
 - > HATEOAS (Hypermedia as the Engine of Application State)
 - a szerver válaszban a további akciókat is leírja
 - Web: linkek behelyezése a válaszba

REST vs. RPC

- RPC:
 - > Ezt a műveletet akarom meghívni: szobafoglalás
 - > Ezekkel a paraméterekkel:
 - Reservation{„Kiss Béla”, 2014-04-22, ...}
 - > HotelService.ReserveRoom(Reservation r)
- REST:
 - > Ezzel szeretnék csinálni valamit:
 - Reservation{„Kiss Béla”, 2014-04-22, ...}
 - > Ezt szeretném csinálni vele: *felvenni* <= általános művelet
 - > ReservationService.Post(Reservation r)
- Végeredményben technikai kérdés (is)
 - > HTTP+JSON+HATEOAS vs WS-* SOAP

RESTful Web vs RPC (WS-*) API

- RESTful Web:
 - ☺ egyszerű
 - ☺ kisebb overhead
 - ☺ jobban kihasználja a HTTP képességeit
 - ☺ URI-t tekintve strukturáltabb, logikusabb
 - ☺ több kliensplatform
- WS-^{*}
 - ☺ szabvány
 - ☺ típusosabb
 - ☺ gazdag middleware szolgáltatások
 - ☺ bármilyen művelet leírható - van amit nehéz HTTP igékkel leírni
 - ☺ nem csak HTTP

ASP.NET Web API vs WCF

- WCF: absztrakt modellt definiál, mely a kommunikációs technológiák széles skáláját lefedi
- Az EGY kommunikációs alrendszer a .NET-ben
 - > Tervezésekor a WS-* volt a középpontban
 - > Nem voltak elterjedtek a webes, RESTful API-k
- Kezdetben WCF Web API
- Ma már független a WCF-től
- Igazából az ASP.NET MVC-től is
 - > hasonló jellegű API

ASP.NET Web API vs WCF

- WCF:
 - > Ha WS-*^{de}-ot (is) használsz
 - szeretnél kliensproxy-t generáltatni (de: OData)
 - > Ha nem (csak) HTTP-t használsz
 - pl. duplex kommunikáció (de: SignalR)
 - > Ha régi (.NET 4.0 előtti) keretrendszert használsz
 - > Hatékony .NET - .NET kommunikáció
- Minden egyéb esetben ASP.NET Web API
- A Web API-t is lehet RPC jellegű interfésszel használni!

Alapelvek

CONVENTION
CONFIGURATION

- Egyszerű
 - > Egyszerűen használható legyen (lightweight)
 - > Konvenció konfiguráció helyett (Convention-over-Configuration)
- Kiterjeszthető
- Tesztelhető
- Hordozható(bb)
 - > Többfajta hosztolási konfiguráció
- Nyílt forráskód


Történet


- 2007. – WCF webhttpbinding a .NET 3.5-ben
- 2008. – WCF REST Starter Kit
- 2010. – WCF Web API
- 2012. február – WCF Web API -> ASP.NET Web API
- 2012. augusztus – ASP.NET Web API v1 (MVC 4)
- 2013. október – ASP.NET Web API v2 (MVC5)
- 2014. január – ASP.NET Web API v2.1 (MVC5.1)
> <http://www.asp.net/web-api>
- 2015. ... - ASP.NET 5, egységes modell (MVC 6)
- 2017. ASP.NET **Core** Web API


ASP.NET: egységes webes modell


New ASP.NET Core Web Application - WebApiSample3


.NET Core ASP.NET Core 2.0 [Learn more](#)



Empty



Web API


Web Application


Web Application (Model-View-Controller)


Angular


React.js


React.js and Redux

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

[Learn more](#)

Change Authentication

Authentication **No Authentication**

☐ Enable Docker Support

OS: Windows

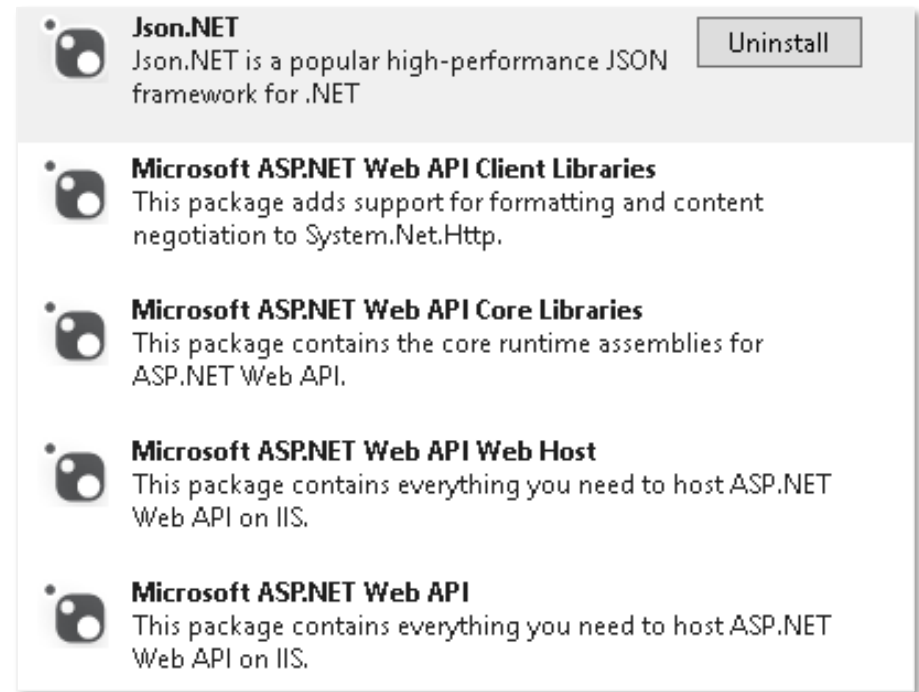
Requires [Docker for Windows](#)

Docker support can also be enabled later [Learn more](#)

OK Cancel

Telepítési környezet

- NuGet csomagok
- Az ASP.NET stack része
- Az MVC-hez igazított kiadási ütemezés



- Open-source:

<http://aspnetwebstack.codeplex.com/>

> (MVC, Web Pages)

Hello World

- Routing konfigurálása
 - > Használhatjuk az alapértelmezett beállításokat
- Adatelérési réteg megírása
- Controller megírása

Hello World - adatelérés

```
interface IProductRepository
{
    IEnumerable<Product> ListProducts();
}

public class InMemoryProductRepository : IProductRepository
{
    public IEnumerable<Product> ListProducts()
    {
        return new List<Product>
        {
            new Product{ ProductID=1, ProductName="Sör", UnitPrice=110},
            new Product{ ProductID=2, ProductName="Bor", UnitPrice=220}
        };
    }
}
```

Hello World - Controller

```
public class ProductsController : ApiController
{
    IProductRepository repo = new InMemoryProductRepository();

    public IEnumerable<Product> Get()
    {
        return repo.ListProducts();
    }
}
```

Hello World

localhost:16953/api/Products

Ehhez az XML-fájlhoz nem tartozik stíluslap-információ. A dokumentumfa az alábbi.

```
<ArrayOfProduct>
  <Product>
    <ProductID>1</ProductID>
    <ProductName>Sör</ProductName>
    <UnitPrice>110</UnitPrice>
  </Product>
  <Product>
    <ProductID>1</ProductID>
    <ProductName>Bor</ProductName>
    <UnitPrice>220</UnitPrice>
  </Product>
</ArrayOfProduct>
```

Konzol Vizsgáló Hibakereső Stíluszerkesztő Profilozó Hálózati

Módszer	Fájl	Tartomány	Fejlécek	Sütik	Paraméterek	Válasz	Időzítések
GET	Products	localhost:16953	Kérés URL: http://localhost:16953/api/Products Kérésmetódus: GET Állapotkód: 200 OK Szerkesztés és újraküldés				

Fejlécek szűrése

Válasz fejlécei (0,366 KB)

- Cache-Control "no-cache"
- Content-Length "348"
- Content-Type "application/xml; charset=utf-8"
- Date "Tue, 15 Apr 2014 19:41:56 GMT"
- Expires "-1"
- Pragma "no-cache"
- Server "Microsoft-IIS/8.0"
- X-AspNet-Version "4.0.30319"
- X-Powered-By "ASP.NET"
- X-SourceFiles "=?UTF-8?B?QzpcRHJvcGJveFwPa...SWVhXGFwaVxQcm9kdWN0cw==?="

Kérés fejlécei (0,684 KB)

- Host "localhost:16953"

Hello World - routing

- Hogyan jutott el a kérés a Get függvényhez? (action)
- WebApi.config:

```
public static void Register(HttpConfiguration config)
{
    config.MapHttpAttributeRoutes();

    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
    );
}
```



Routing

- A bejövő HTTP kérést melyik metódus szolgálja ki?
- Routing tábla alapján
- Leképezés: kérés adatai => controller action
- Az action paramétereknek is értéket kell adni
- *ApiController.ControllerContext.RouteData.Values*
- *HttpConfiguration.Routes*
- Konvenció alapú vagy attribútumos megadás

Routing – konvenció alapján

- Konvenció alapú megadás routing leképezéssel
- Egy routing elemei:
 - > név
 - > sablon
 - > alapértelmezett értékek (opcionális)
 - > kényszerek (opcionális)

`http://localhost:50884/api/products`

`api/{controller}`

```
public class ProductsController : ApiController {  
    public IEnumerable<Product> Get()  
}
```

Sablon felépítése

- Sablon: a formatstringhez hasonló
 - > placeholdereket helyezhetünk el
 - > speciális placeholderek: {controller}, {action}
 - > wildcard placeholder {*tags}
- A kérés és a sablon alapján routing dictionaryt építünk
 - > Pl. **api/{controller}/{category}/{name}**
 - > GET /api/Products/Drinks/Bor
 - Controller osztály neve: **„Products”**
 - Category paraméter: **„Drinks”**
 - Name paraméter: **„Bor”**

Alapértelmezett értékek

- Anonim osztállyal
- Megadható, hogy opcionális-e egy kulcs
- Olyan kulcs is kitölthető, amihez nincs is URL szegmens

Kényszerek

- Anonim osztállyal
- string = regex
- Saját kényszerek (IHttpRouteConstraint)

```
config.Routes.MapHttpRoute(  
    name: "DefaultApi",  
    routeTemplate: "api/{controller}/{category}/{id}",  
    defaults: new { id=RouteParameter.Optional },  
    constraints: new { category="Cars|Drinks|Chicks" }  
);
```

Sablon választás

- Illeszkedésvizsgálat
 - > ami nem placeholder, annak egyeznie kell
 - > placeholderek: az URL elejéről indulva egyszerű hozzárendelés
 - > kényszerek és elhagyhatóság
- Több sablon is illeszkedhet
 - > az első illeszkedő nyer
- Ha nincs illeszkedő: 404-es hiba

Controller/action választás

- A felépített dictionary alapján
 - > Controller kiválasztása **konvenció** alapján
 - *controller* kulcshoz tartozó érték kikeresése (pl. Products)
 - *érték+Controller* (pl. ProductsController)
 - > Action kiválasztása a controller megfelelő függvényei közül
 - HTTP ige egyeztetése
 - ha van az *action* kulcshoz érték, akkor ezen érték egyeztetése
 - Paraméterek egyeztetése



Action illesztés

- Action minden publikus tagfüggvény
 - > Kivéve, ha rajta van a NonAction attribútum
- Minden action kiszolgálhat egy vagy több ígét
 - > attribútum (AcceptVerbs, HttpPost, stb.)
 - > névkonvencióval (prefix alapján, GetXXX)
 - > ha egyik se -> POST

Paraméter illesztés

- Ki kell elégítenünk a függvény paraméterlistáját
- Minden függvényparaméterhez értéket kell rendelni
 - > kivéve opcionális paraméterek
 - > kivéve komplex típusú paraméterek
- Forrás lehet
 - > route dictionary
 - > URL query string
- A legtöbb egyezés nyer

Routing – attribútumok alapján

- Eredetileg közösségi fejlesztés ☺
- Engedélyezés: IConfiguration példányban
 - > `config.MapHttpAttributeRoutes();`
- A routing sablont közvetlenül az actionre rakjuk
 - > a helyőrzők szerepe a paraméterek feltöltése

```
[Route("api/Products/{minprice}/{maxprice}")]
public IEnumerable<Product> GetProduct(int minprice, int maxprice)
{
    return repo.ListProducts()
        .Where(p => p.UnitPrice >= minprice
            && p.UnitPrice <= maxprice);
}
```

Controller szintű route-olás

- Route prefix / Route attribútum
 - > controller szintű
 - > ,~'-vel felülírható

```
[RoutePrefix("api/Products")] / [Route("api/[controller]")]  
public class ProductsController : ApiController  
{  
    IRepository repo = new InMemoryProductRepository();  
  
    [Route("{minprice:int}/{maxprice:int}")]  
    public IEnumerable<Product>  
        GetProduct (int minprice, int maxprice)  
    { ...
```


Szegmensek

- Kényszerek a szegmensekre
 - > gyakori típusokra
 - > szöveghossz (max, min)
 - > számoknál értéktartomány (max, min)
 - > regex
 - > saját: IHttpRouteConstraint
- Wildcard szegmens itt is van
- Opcionális értékek és default

```
[Route("filter/{minprice:int=100}/{maxprice:int?}")]
```

```
public IEnumerable<Product>
```

```
    GetProduct(int minprice, int maxprice=int.MaxValue)
```

Route választás

- Melyik route-ra illeszkedik az URL?
- RouteOrder propertyvel befolyásolható, hogy melyik nyerjen
- További szabályok a szegmensek alapján
- Végző esetben a sablon szövege dönt szövegrendezés alapján

Routing – attribútum vs konvenció

- Konvenció alapú
 - > egy helyre gyűjthetjük a route-okat
 - > sok esetet lefedhetünk egy sablonnal
- Attribútum alapú
 - > jobban finomhangolható
 - > vannak esetek, amik általános sablonnal nehezen megadhatók

```
[Route("order/{orderid:int}")]
```

```
public IEnumerable<Product> GetProductsByOrder(int id)
```

- A két módszer kombinálható!

Model binding

- Hogyan lesz egy POST kérés adataiból függvényparaméter?
- Ökölszabály
 - > egyszerű típus => URI-ból (route dictionary, query string)
 - > komplex típus => MediaTypeFormatter-rel a kérés tartalmi részéből
- Megváltoztatható attribútumokkal
 - > FromUri: komplex típus esetén mégis az URI-ból vegye
 - > FromBody: egyszerű típus esetén mégis a Bodyből vegye

[HttpPost]

```
public void CreateTodoItem([FromBody] TodoItem item) {  
    if (!ModelState.IsValid)  
        { Context.Response.StatusCode = 400; }  
}
```

...

Media Formatter

- Sorosítók, melyek meghatározott MIME típusokat képesek sorosítani
- MIME-típus pl.: application/**json**, application/**xml**
- Content-Type HTTP header
- Ősosztály: MediaTypeFormatter
- Beépített: XML, JSON, BSON, Form Url Encoded
- Sajátot is írhatunk leszármazással
 - > meg kell adnunk a támogatott MIME típust
 - > regisztráció: `HttpConfiguration . Formatters`

Model binding testreszabás

- Komplex típus kezelése egyszerű típusként:
TypeConverter
- Saját model binder
 - > IModelBinder, IValueProvider
 - > HttpParameterBinding
- Model binding folyamat testreszabása:
IActionValueBinder

TypeConverter példa

```
[TypeConverter(typeof(GeoPointConverter))]  
public class GeoPoint  
{  
    public double Latitude { get; set; }  
    public double Longitude { get; set; }  
  
    public static bool TryParse(string s, out GeoPoint result)  
    {
```

```
        class GeoPointConverter : TypeConverter  
        {  
            public override bool CanConvertFrom(ITypeDescriptorContext context,  
            {  
                if (sourceType == typeof(string))  
                {  
                    return true;  
                }  
                return base.CanConvertFrom(context, sourceType);  
            }  
  
            public override object ConvertFrom(ITypeDescriptorContext context,  
            CultureInfo culture, object value)  
            {  
                if (value is string)  
                {  
                    GeoPoint point;  
                    if (GeoPoint.TryParse((string)value, out point))  
                    {  
                        return point;  
                    }  
                }  
            }  
        }  
    }  
}
```

IValueProvider, IModelBinder

- IValueProvider

- > A nyers HTTP kérésből szótárat készít
- > <http://localhost/api/values/1?location=48,122>
 - id = "1"
 - location = "48,122,,
- > Ebből is készíthető saját...

- IModelBinder

- > A ValueProvider által készített szótárat használja
- > Paraméterre vagy típusra állítható be

- HttpParameterBinding: még általánosabb

Model Binding - összefoglalás

- Komplex típus
 - > Alapértelmezetten a HTTP kérés tartalmi részéből (body), Media Formatterrel (xml / json)
 - > Vagy FromUrllal, konverterrel
- Egyszerű típus
 - > URI + query string-ből
 - > Beépített az egyszerű típusokra, converter vagy ModelBinder
- Minden egyéb, például fejléc adatok: `HttpParameterBinding`

Content Negotiation

- Megegyezés a válasz formátumáról
- A kliens kér az Accept fejlécben (=> halmaz1)
 - > (IE) `Accept: text/html, application/xhtml+xml, */*`
 - > (FF) `Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8`
- A rendelkezésre álló és alkalmas media formatterek (=>halmaz2)
 - > *CanWriteType, SupportedMediaTypes*
- `HttpContext.Negotiate`

["film", "StarWars 7"]

F12 DOM Explorer Console 2 Debugger Network Performance Memory

Content type

Name / Path	
Values	
http://localhost:53235/api/	

Headers Body Parameters Cookies Timings

Request URL: **http://localhost:53235/api/Values**

Request Method: **GET**

Status Code: **200 / OK**

Request Headers

Accept: **text/html, application/xhtml+xml, image/jxr, */***

Accept-Encoding: **gzip, deflate**

```
<ArrayOfstring xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
  <string>film</string>
  <string>StarWars 7</string>
</ArrayOfstring>
```

Elements Console Sources Network Timeline Profiles Resources Security Audits Adblock Plus

View: ☐ Preserve log ☐ Disable cache | No throttling

Filter ☐ Hide data URLs **All** XHR JS CSS Img Media Font Doc WS Manifest Other

100 ms 200 ms 300 ms 400 ms 500 ms 600 ms

Name Headers Preview Response Timing

Values

data:image/svg+xml,...

Request Headers view source

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Accept-Encoding: gzip, deflate, sdch

Content Negotiation

- A két halmaz metszetét kell vennünk
 - > több egyezés esetén elsőként a media formatter által adott minőségi faktor dönt (MediaTypeMapping)
 - > másodsorban a kliens által adott minőségi faktor (q)
- Ha nincs döntés, akkor a kérés tartalmának formátuma
 - > Ha ekkor sincs döntés, akkor az első alkalmas formatter kapja meg a feladatot
- MediaTypeMapping
 - > egy media formatter megadhat request => quality leképezéseket

A biztonságról előzetes

- Authorize attribútum

- > Globálisan

- ```
config.Filters.Add(new AuthorizeAttribute());
```

- > Controller szinten

- ```
[Authorize(Roles="Admins")]
```

- ```
public class ProductsController : ApiController
```

- > Felhasználók, szerepek megadása

- AllowAnonymous attribútum

- > Action szinten engedélyezi a hívást

- Vagy saját attribútum saját logikával

# Http Válasz előállítás

- HttpResponseMessage megy vissza a hálózaton
  1. Ha a metódus HttpResponseMessage-et at vissza, akkor pontosan azt küldi a kliensnek
  2. Ha nincs visszatérési érték, akkor egy 204-es (No Content) kód megy vissza
  3. Egyéb esetben a média típus formázó sorosítja az üzenetet és azt küldi vissza a kliensnek

# HttpResponseMessage

- Az action-ben kézzel összerakhatjuk a választ

```
public HttpResponseMessage Post(Product product)
{
 repo.Add(product);
 HttpResponseMessage msg =
 new HttpResponseMessage(HttpStatusCode.Created);
 msg.Headers.Location =
 new Uri(Url.Link("DefaultApi",
 new {id=product.ProductID}));
 return msg;
}
```

# Action Filters

- Megadhatunk logikát, ami az action előtt, illetve után fut le
- Cross-cutting feladatokhoz
  - > autorizáció
  - > validáció
  - > naplózás
  - > hibakezelés, ...
- ActionFilter attribútum
  - > OnActionExecuting: előtte
  - > OnActionExecuted: utána
- A leszármaztatott attribútumot az actionre tesszük
- Globális: `HttpConfiguration.Filters`-hez hozzáadjuk



# Filter példa

```
private class SampleActionFilterImpl : IActionFilter
{
 private readonly ILogger _logger;
 public SampleActionFilterImpl(ILoggerFactory loggerFactory)
 {
 _logger = loggerFactory.CreateLogger<SampleActionFilterAttribute>()
 }

 public void OnActionExecuting(ActionExecutingContext context)
 {
 _logger.LogInformation("Business action starting...");
 // perform some business logic work
 }

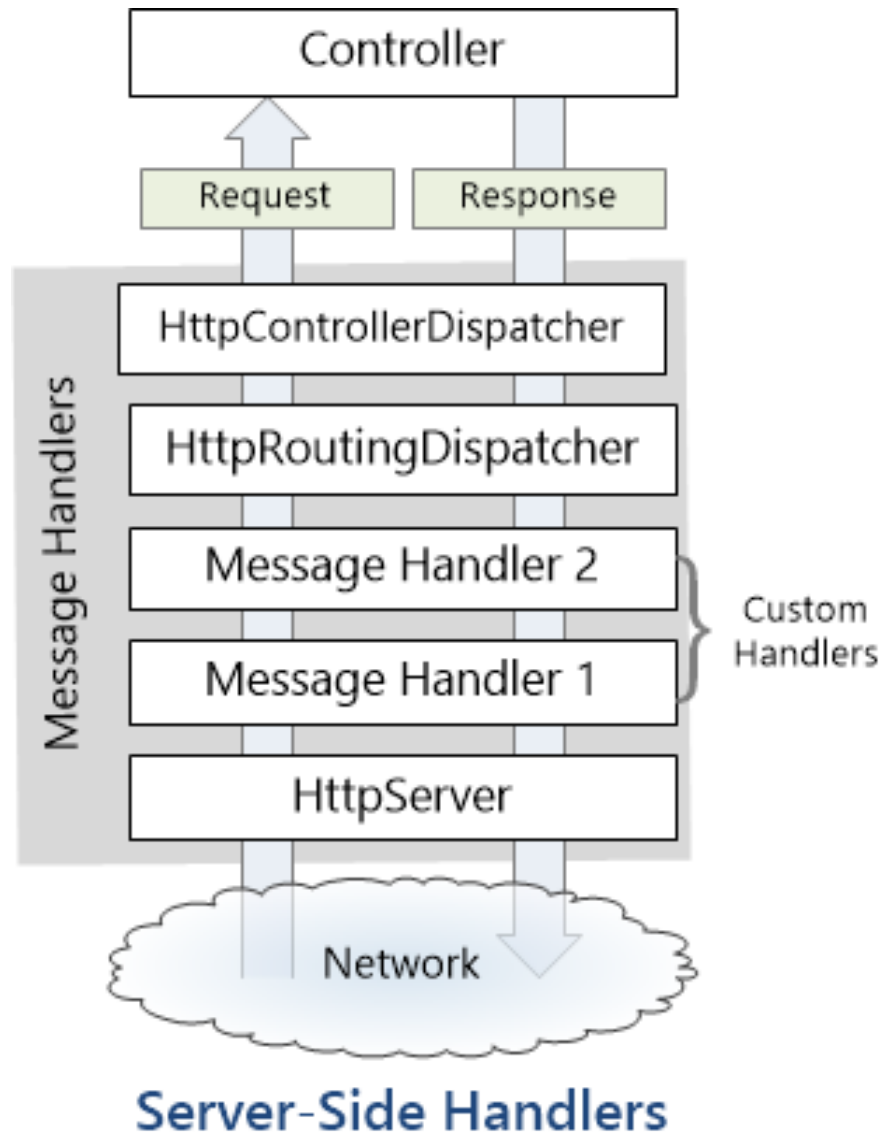
 public void OnActionExecuted(ActionExecutedContext context)
 {
 // perform some business logic work
 _logger.LogInformation("Business action completed.");
 }
}
```

# Message Handler

- HTTP kérésre HTTP választ képes előállítani
- Lényegében az egész szerver oldal erről szól (HttpServer)
- Saját üzenet kezelőhöz DelegatingHandler

```
Task<HttpResponseMessage> SendAsync(
 HttpRequestMessage request,
 CancellationToken cancellationToken
)
```
- base.SendAsync -> meghívjuk a lánc következő tagját (*innerHandler*)
- HttpConfiguration-ben regisztráljuk

# Message Handler



# Delegating Handler

```
public class ApiKeyHandler : DelegatingHandler{
 List<string> keys= new List<string> {"secretkey", "ultrasecretkey" };
 protected override Task<HttpResponseMessage> SendAsync(
 HttpRequestMessage request, CancellationToken cancellationToken)
 {
 if (!keys.Contains(request.RequestUri.ParseQueryString()["key"]))
 {
 HttpResponseMessage response =
 new HttpResponseMessage(HttpStatusCode.Forbidden);
 return Task<HttpResponseMessage>.FromResult(response);
 }
 return base.SendAsync(request, cancellationToken);
 }
}
```

# Globális regisztráció

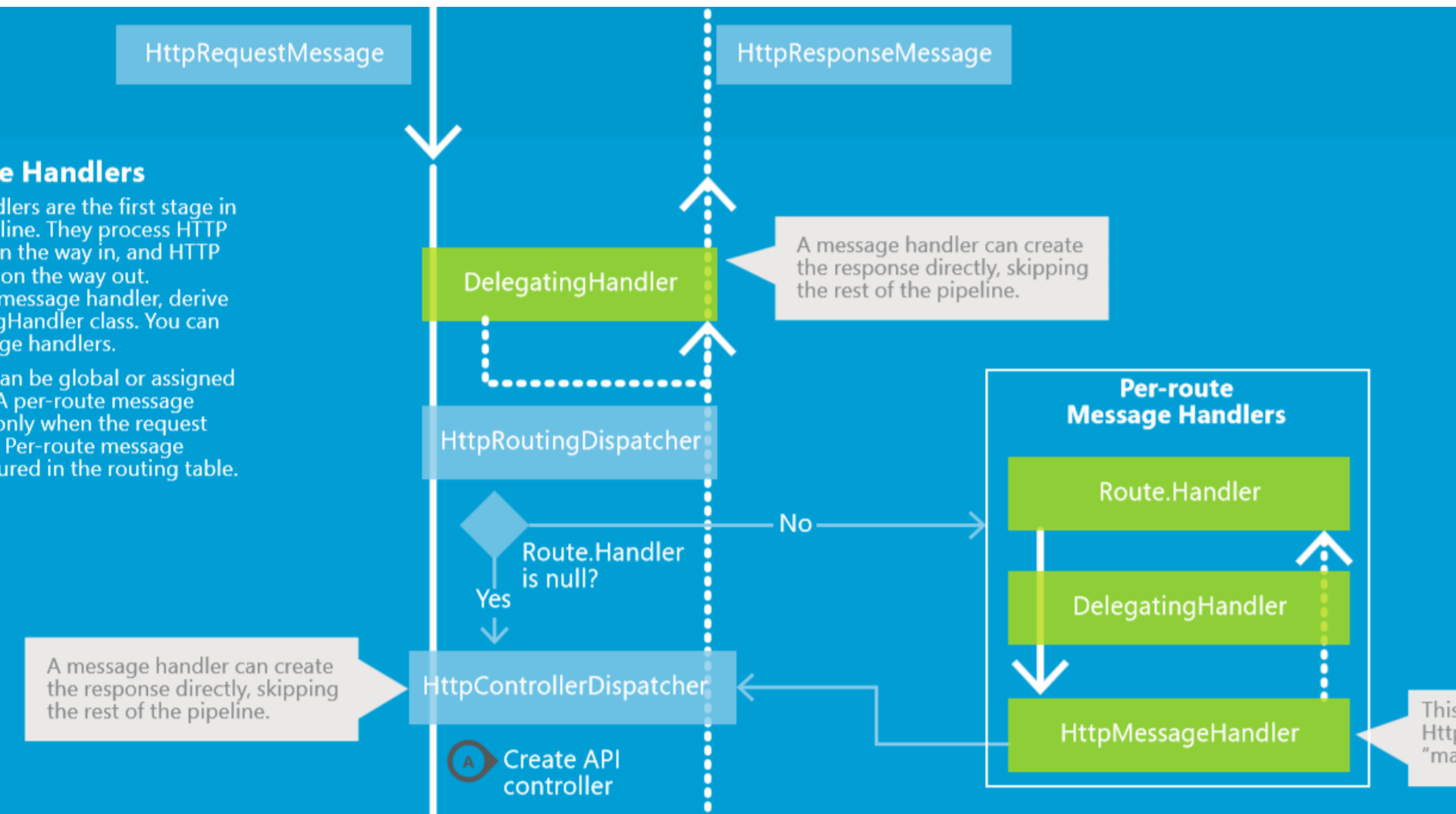
```
public static void Register(HttpConfiguration
config)
{
 config.MapHttpAttributeRoutes();
 config.Routes.MapHttpRoute(
 name: "DefaultApi",
 routeTemplate: "api/{controller}/{id}",
 defaults: new { id = RouteParameter.Optional }
);

 config.MessageHandlers.Add(new ApiKeyHandler());
}
```

# Route szintű regisztráció

```
public static void Register(HttpConfiguration config)
{
 config.MapHttpAttributeRoutes();
 config.Routes.MapHttpRoute(
 name: "DefaultApi",
 routeTemplate: "api/{controller}/{id}",
 defaults: new { id = RouteParameter.Optional }
 handler: new ApiKeyHandler(),
 constraints: null
);
}
```

# Üzenetkezelési folyamat – 1. rész



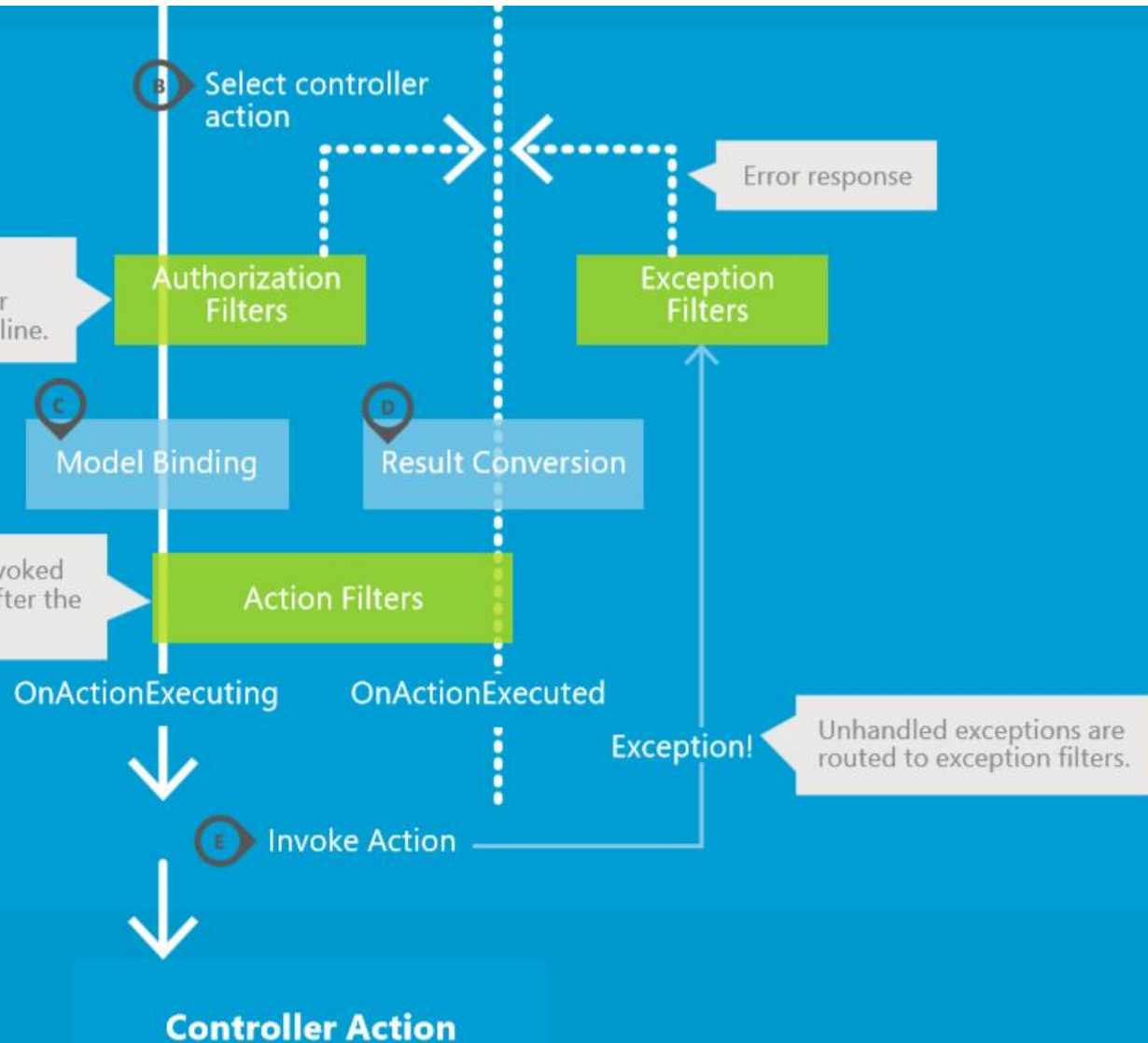
# Üzenetkezelési folyamat – 2. rész

## Controller

The controller is where you define the main logic for handling an HTTP request. Your controller derives from the ApiController class or implements the IHttpController interface.

If the request is not authorized, an authorization filter can create an error response and skip the rest of the pipeline.

Action filters are invoked twice, before and after the controller action.





# Hosting opciók

- Szükség van egy konfigurációra és egy *HttpMessageHandler* implementációra
- ASP.NET (*HttpServer*)
- Self-Host
  - > OWIN (*WebApp*)
  - > Klasszikus (*HttpSelfHostServer : HttpServer*) – WCF alapú, már nem javasolt
- Azure
- In-memory Hosting

# In-Memory Hosting

- A kliens szerver kommunikációt elejétől végéig lejátszhatjuk a memóriában
- Nincs hálózati kommunikáció!
- Integrációs teszteknel hasznos
- HttpClient( HttpResponseMessage handler )
  - > A HttpServer is egy HttpResponseMessage handler!
  - > OWIN test szerver Microsoft.Owin.Testing

# In-Memory Hosting

```
using (TestServer server=TestServer.Create<WebApiConfig>()) {
 // Create HttpClient and make a request to api/values
 HttpClient client = new HttpClient(server.Handler);

 var response = client.GetAsync(baseAddress+"api/Products").Result;

 Console.WriteLine(response);
 Console.WriteLine(response.Content.ReadAsStringAsync().Result);
 Console.ReadLine();
}
```

# Kliensek

- Kliens lehet bármi, ami tud HTTP valamilyen kvázi szabványos formátummal (XML, JSON) dolgozni
- JavaScript (böngészőben) – JQuery Ajax
- .NET, WinRT, Java, Python, C/C++ (libCurl)
- Android, iOS, Windows Phone

# .NET kliens

- 1.0 óta van HTTP kliens .NET-ben
  - > HttpWebRequest
  - > WebClient (.NET 2.0)
- 2012: HttpClient
  - > a .NET 4.5 része
  - > NuGet - Microsoft.Net.Http
- NuGet - Microsoft.AspNet.WebApi.Client
  - > JSON .NET
  - > Formatter-ek

# .NET kliens írása

1. NuGet telepítés (Microsoft.AspNet.WebApi.Client)
2. Szolgáltatás felfedezése
3. Modellosztályok „generálása”
4. HttpClient konfigurálása
5. Kérés összeállítása
6. Kérés elküldése
7. Válasz feldolgozása

# Szolgáltatások felfedezése

- Tudnunk kell a műveletekről és a paraméterben, visszatérési értékekben utazó entitásokról
- Valahogy meg kell jelenniük a modell/DTO osztályoknak a kliensben is
- Nincs WSDL de van OpenAPI (Swagger), ...
- Lehetőségek a műveletek felfedezésére, leírására
  - > dokumentáció, példa kérés-válasz
  - > WADL, **OpenAPI** (Swagger), ...
  - > entitások dokumentációja
  - > XML séma, forrásfájlok

# Dokumentációs eszközök 1

- Apriary.io (Oracle megveszi)
  - > külső szolgáltatás
  - > Pl.: <https://developers.themoviedb.org>
  - > Design first!
  - > API Blueprint: markdown-nal leírt API
  - > Automatikus szerver mock, kliens proxy, dokumentáció, példa generálás
  - > Debug szerver szolgáltatás



**GET**`/3/movie/{id}/alternative_titles`

Get the alternative titles for a specific movie id.

### Required Parameters

api\_key

### Optional Parameters

country ISO 3166-1 code.

append\_to\_response Comma separated, any movie method

[Example](#)[Show code sample](#)

### Request

```
1 | Accept: application/json
```

### Response

```
1 | 200 (OK)
2 | ETag: "a7664969471f59db2c272c672df92ae1"
3 | Content-Type: application/json
```

```
1 | {
2 | "id": 550,
3 | "titles": [
4 | {
5 | "iso_3166_1": "PL",
6 | "title": "Podziemny krag"
```

# Dokumentációs eszközök 2

- ASP.NET Web API Help Pages
  - > a Web API-s projektünkbe generálhatunk dokumentációs weboldalt
  - > ASP.NET MVC-re épül
  - > ApiExplorer
  - > XML kommentek
  - > teszt kliens is

# ASP.NET Web API Help Pages

The screenshot shows a web browser window with the address bar at `http://localhost:57785/Help` and the page title `GET api/Customers`. The page content includes:

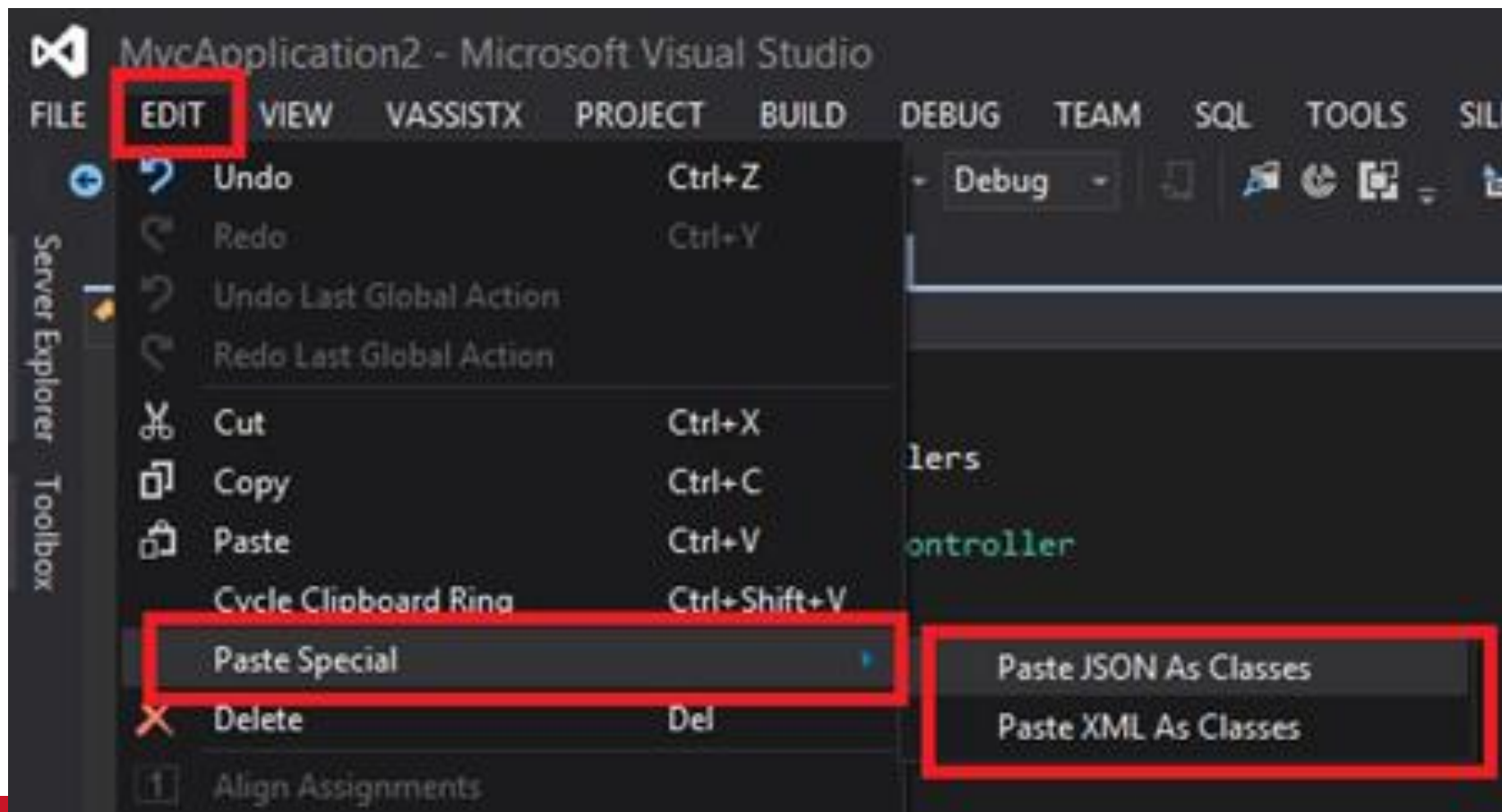
- GET api/Customers**: The main heading for the endpoint.
- Documentation for 'Get'.
- Response Information**: A section header.
- Response body formats**: A section header.
- application/json, text/json**: A tab for the JSON response format.
- Sample:** A JSON array of three customer objects:

```
[
 {
 "ID": 1,
 "FirstName": "sample string 2",
 "LastName": "sample string 3"
 },
 {
 "ID": 1,
 "FirstName": "sample string 2",
 "LastName": "sample string 3"
 },
 {
 "ID": 1,
 "FirstName": "sample string 2",
 "LastName": "sample string 3"
 }
]
```
- application/xml, text/xml**: A tab for the XML response format.
- Sample:** An XML array of customer objects:

```
<ArrayOfCustomer xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://schemas.datacontract.org/2004/07/MvcApplication1.Controllers">
 <Customer>
```

# Modellosztályok generálása

- Példa JSON/XML-ből gyárthatunk C# osztályokat



# Http kérés – .NET HttpClient, GET

- System.Net.Http.HttpClient osztály (.NET 4.5+, Core)
- HTTP GET, PUT, POST, DELETE kérések küldése
- **GetAsync**
  - > Letölti a megadott címen lévő tartalmat
  - > **HttpCompletionOption**: várja meg az egész tartalom letöltését vagy csak a header letöltését
  - > **CancellationToken** átadható
  - > Eredménye: **HttpResponseMessage**
    - StatusCode
    - **EnsureSuccessStatusCode()** – ha nem sikeres, akkor kivételt dob
    - Header: a válasz fejléce
    - Content (**HttpContent** - a válasz tartalma)
      - Kiolvasás: **ReadAsStreamAsync()**, **ReadAsStringAsync()**, **ReadAsByteArrayAsync()**
  - > Ha csak a tartalom kell: **GetStreamAsync**, **GetStringAsync**, **GetByteArrayAsync**

# .NET HttpClient GET példa: weboldal letöltése

```
HttpClient httpClient = null;
try
{
 httpClient = new HttpClient();

 // A teljes oldal letöltése
 HttpResponseMessage response =
 await httpClient.GetAsync("http://www.bing.com");

 // Az oldal tartalmának beolvasása
 var responseContent = await response.Content.ReadAsStringAsync();
}
catch (Exception) { /* Hibakezelés */ }
finally { httpClient.Dispose(); }
```

# REST-es bővítő metódusok

- Microsoft.AspNet.WebApi.Client csomag
- HttpClient-hez
  - > Post/PutAsJsonAsync
  - > Post/PutAsXmlAsync
    - Megadható a MediaTypeFormatter, naplózás stb.
- HttpContent-hez
  - > ReadAsStringAsync<T>: T típusú objektumot olvas
    - Megadható a MediaTypeFormatter, naplózás stb.
- HttpClientExtensions és HttpContentExtensions

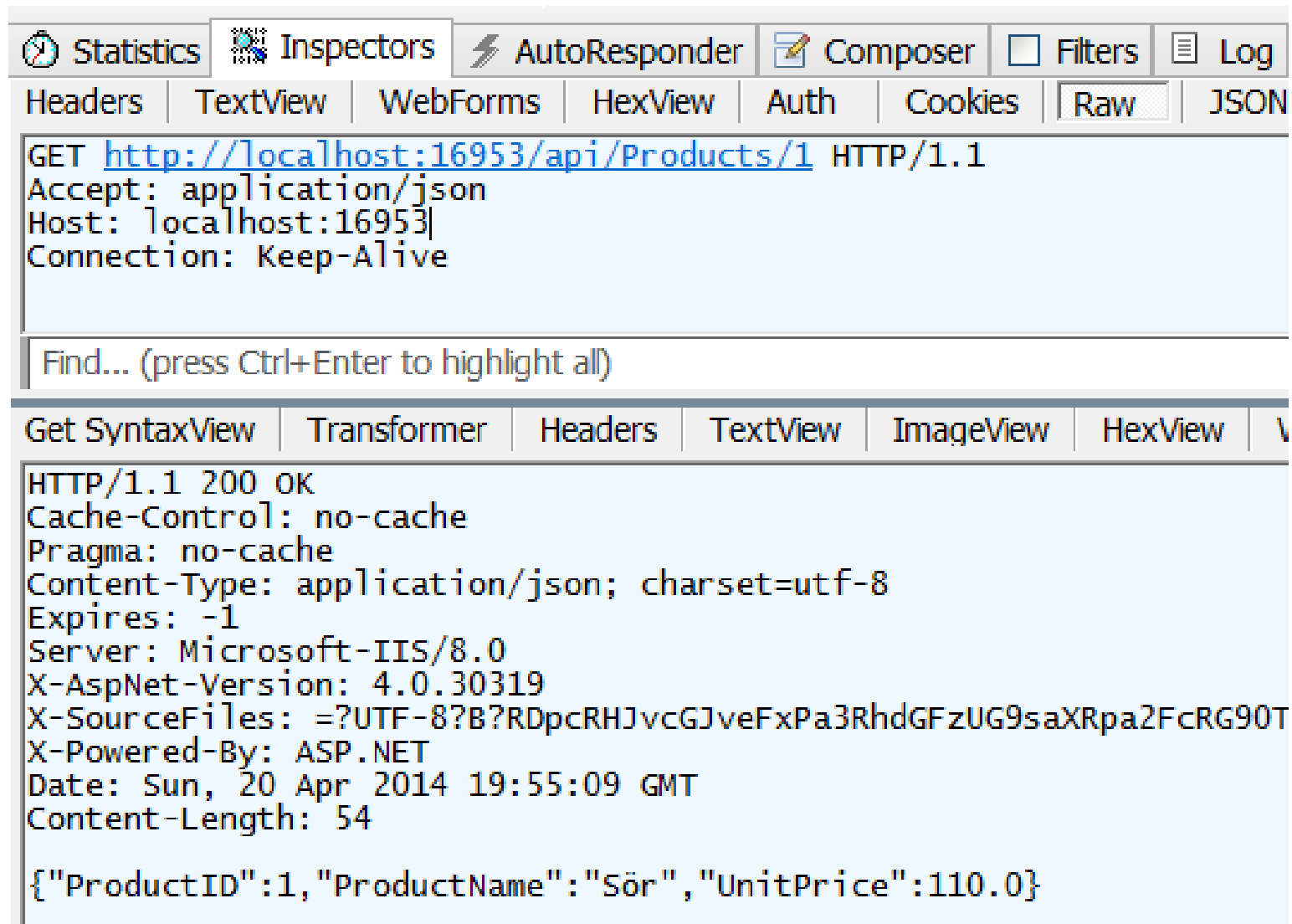
# Egyszerű kliens

```
static void Main(string[] args){
 DoWork().Wait(); Console.ReadLine(); }

static async Task DoWork(){
 using (HttpClient client = new HttpClient()) {
 client.BaseAddress = new Uri("http://localhost.fiddler:16953");
 client.DefaultRequestHeaders.Accept.Add(
 new MediaTypeWithQualityHeaderValue("application/json"));
 HttpResponseMessage response = await
 client.GetAsync("api/Products/1");
 if (response.IsSuccessStatusCode) {
 Product product = await response.Content.ReadAsAsync<Product>();
 Console.WriteLine("{0}\t{1}HUF", product.ProductName,
 product.UnitPrice);
 }
 }
}
```



# Kérés-válasz Fiddler-ben



The screenshot shows the Fiddler web proxy interface. The top toolbar includes buttons for Statistics, Inspectors, AutoResponder, Composer, Filters, and Log. Below the toolbar is a row of tabs: Headers, TextView, WebForms, HexView, Auth, Cookies, Raw, and JSON. The main pane displays a GET request to `http://localhost:16953/api/Products/1` with headers: `Accept: application/json`, `Host: localhost:16953`, and `Connection: Keep-Alive`. Below the request is a search bar labeled "Find... (press Ctrl+Enter to highlight all)". The bottom section shows the response headers: `HTTP/1.1 200 OK`, `Cache-Control: no-cache`, `Pragma: no-cache`, `Content-Type: application/json; charset=utf-8`, `Expires: -1`, `Server: Microsoft-IIS/8.0`, `X-AspNet-Version: 4.0.30319`, `X-SourceFiles: =?UTF-8?B?RDpcRHJvcGJveFxpPa3RhdGFzUG9saXRpa2FcRG90T`, `X-Powered-By: ASP.NET`, `Date: Sun, 20 Apr 2014 19:55:09 GMT`, and `Content-Length: 54`. The response body is a JSON object: `{"ProductID":1,"ProductName":"Sör","UnitPrice":110.0}`.

Statistics Inspectors AutoResponder Composer Filters Log

Headers TextView WebForms HexView Auth Cookies Raw JSON

GET <http://localhost:16953/api/Products/1> HTTP/1.1  
Accept: application/json  
Host: localhost:16953  
Connection: Keep-Alive

Find... (press Ctrl+Enter to highlight all)

Get SyntaxView Transformer Headers TextView ImageView HexView \

HTTP/1.1 200 OK  
Cache-Control: no-cache  
Pragma: no-cache  
Content-Type: application/json; charset=utf-8  
Expires: -1  
Server: Microsoft-IIS/8.0  
X-AspNet-Version: 4.0.30319  
X-SourceFiles: =?UTF-8?B?RDpcRHJvcGJveFxpPa3RhdGFzUG9saXRpa2FcRG90T  
X-Powered-By: ASP.NET  
Date: Sun, 20 Apr 2014 19:55:09 GMT  
Content-Length: 54

`{"ProductID":1,"ProductName":"Sör","UnitPrice":110.0}`

# Http kérés – .NET HttpClient, POST

- **PostAsync**: POST kérést küld a megadott címre
  - > Paraméter: **HttpContent**
    - StringContent, StreamContent, ByteArrayContent
    - MultipartContent / MultipartFormContent
  - > CancellationToken átadható
  - > Eredménye: **HttpResponseMessage**
- **PutAsync, DeleteAsync**: hasonlóan az előzőekhez
- **SendAsync( HttpRequestMessage )**
  - > Kérés összeállítása „kézzel”
- **DefaultRequestHeaders**: HTTP header beállítása
- **CancelPendingRequests()**: az összes függő kérés megszakítása
- **Dispose()**: erőforrások felszabadítása
- **HttpMessageHandler**
  - > Hívás testreszabása
  - > A konstruktorban átadható saját
  - > Pl. HttpClientHandler (Redirect, CookieContainer, Credentials, Proxy, stb.)

# Beszúrás példa

```
static async Task DoWork(){
 using (HttpClient client = new HttpClient()) {
 Product pAlinka = new Product {ProductName="Pálinka",
UnitPrice=440 };

 client.BaseAddress = new Uri("http://localhost.fiddler:16953");

 //Nem kell accept, a Content Type-ot pedig a PostXXX fv.
 beállítja

 HttpResponseMessage resp =
 await client.PostAsJsonAsync("api/Products",pAlinka);

 //Ha nem 200 => kivétel

 resp.EnsureSuccessStatusCode();

 Console.WriteLine("A {0} címe: {1}",
 pAlinka.ProductName,resp.Headers.Location);
 }
}
```

# Kérés-válasz Fiddler-ben

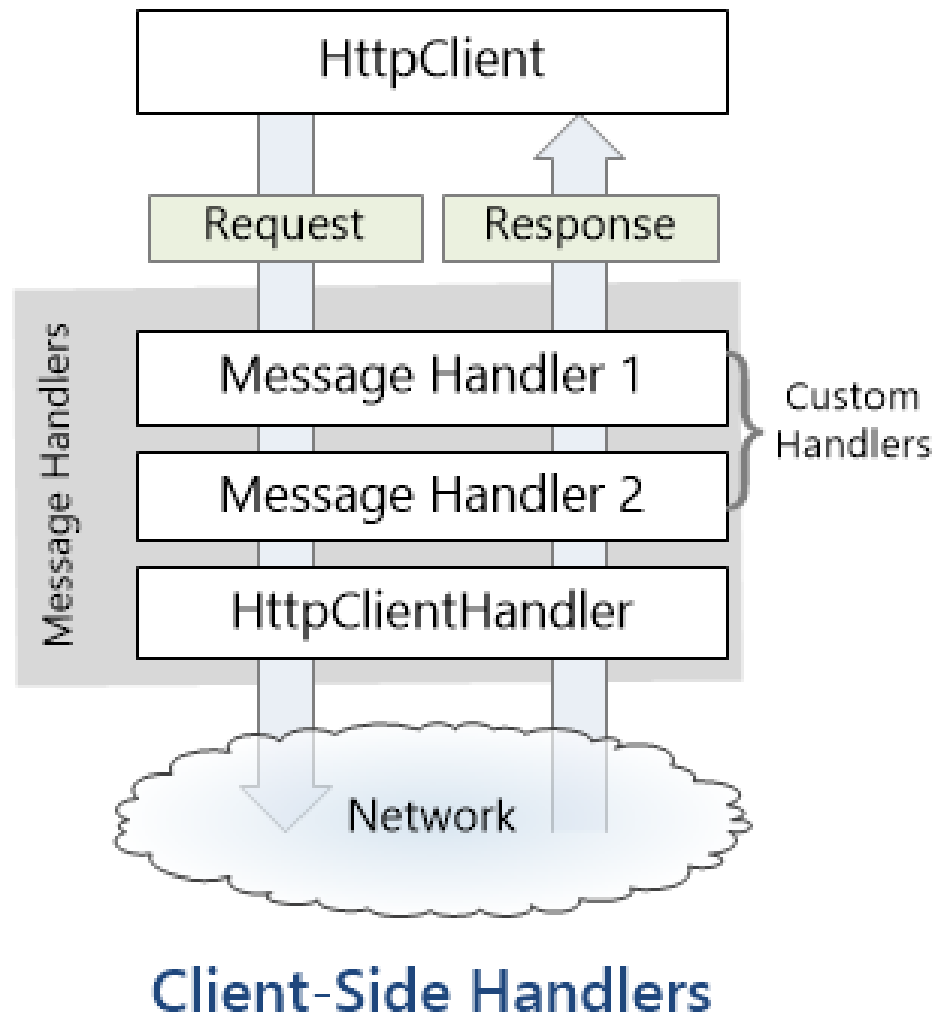
| Headers                                                                                                                                                                                                                                                                                                                           | TextView    | WebForms | HexView  | Auth      | Cookies | Raw |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|----------|----------|-----------|---------|-----|
| <pre>POST http://localhost:16953/api/Products HTTP/1.1 Content-Type: application/json; charset=utf-8 Host: localhost:16953 Content-Length: 58 Expect: 100-continue Connection: Keep-Alive  {"ProductID":0,"ProductName":"Pálinka","UnitPrice":440.0}</pre>                                                                        |             |          |          |           |         |     |
| Find... (press Ctrl+Enter to highlight all)                                                                                                                                                                                                                                                                                       |             |          |          |           |         |     |
| Get SyntaxView                                                                                                                                                                                                                                                                                                                    | Transformer | Headers  | TextView | ImageView | He      |     |
| <pre>HTTP/1.1 201 Created Cache-Control: no-cache Pragma: no-cache Expires: -1 Location: http://localhost:16953/api/Products/0 Server: Microsoft-IIS/8.0 X-AspNet-Version: 4.0.30319 X-SourceFiles: =?UTF-8?B?RDpcRHJvcGJveFwPa3RhdGFzUG9saXRpa X-Powered-By: ASP.NET Date: Sun, 20 Apr 2014 20:07:51 GMT Content-Length: 0</pre> |             |          |          |           |         |     |

# Http kérés - testreszabás

- Minden HttpClient példányhoz saját HTTP stack tartozik amit a HttpClientHandler reprezentál
- Ezt lehet testreszabni úgy, hogy saját handlert hozunk létre, módosítjuk a tulajdonságait és ezt adjuk át a HttpClient konstruktorának
  - > Cacheing
  - > Tömörítés
  - > Autentikáció
  - > Proxy
  - > ...

```
HttpClientHandler myHandler = new HttpClientHandler();
myHandler.AllowAutoRedirect = false;
HttpClient myClient = new HttpClient(myHandler);
```

# Kliensoldali Message Handler



# Http kérés – timeout beállítása

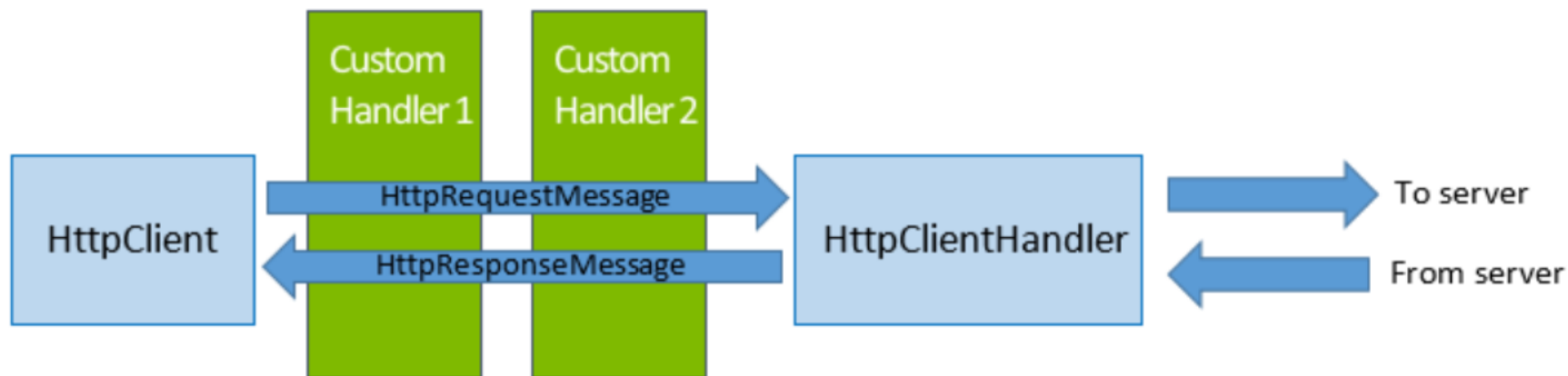
```
var cts = new CancellationTokSource();
cts.CancelAfter(TimeSpan.FromSeconds(30));

var httpClient = new HttpClient();
var resourceUri = new Uri("http://www.contoso.com");

try
{
 HttpResponseMessage response = await httpClient.GetAsync(resourceUri, cts);
}
catch (TaskCanceledException ex)
{
 // Handle request being canceled due to timeout.
}
catch (HttpRequestException ex)
{
 // Handle other possible exceptions.
}
```

# Http kérés – több handler

- Több handler is egymás után fűzhető
- A DelegatingHandler-ből kell származni, ezeket lehet egymás után fűzni, az utolsó a HttpClientHandler, ami küldi a kérést





# Http kérés – saját handler

- Egyszerű debug handler:

```
public class CustomHandler1 : DelegatingHandler
{
 // Constructors and other code here.
 protected async override Task<HttpResponseMessage> SendAsync(
 HttpRequestMessage request, CancellationToken cancellationToken)
 {
 // Process the HttpRequestMessage object here.
 Debug.WriteLine("Processing request in Custom Handler 1");

 // Once processing is done, call DelegatingHandler.SendAsync to pass
 // inner handler.
 HttpResponseMessage response = await base.SendAsync(request, cancella

 // Process the incoming HttpResponseMessage object here.
 Debug.WriteLine("Processing response in Custom Handler 1");

 return response;
 }
}
```

# Http kérés – saját handler befűzése

```
public class CustomHandler2 : DelegatingHandler
{
 // Similar code as CustomHandler1.
}
public class Foo
{
 public void CreateHttpClientWithChain()
 {
 HttpClientHandler systemHandler = new HttpClientHandler();
 CustomHandler1 myHandler1 = new CustomHandler1();
 CustomHandler2 myHandler2 = new CustomHandler2();

 // Chain the handlers together.
 myHandler1.InnerHandler = myHandler2;
 myHandler2.InnerHandler = systemHandler;

 // Create the client object with the topmost handler in the chain.
 HttpClient myClient = new HttpClient(myHandler1);
 }
}
```

# Kérdések?

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