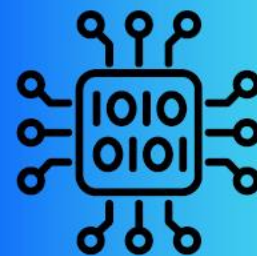


# TechEd

Největší česko-slovenská  
**konference o IT**

**20. - 22. 5. 2025 | PRAHA/ONLINE**

[www.teched.cz](http://www.teched.cz) | [www.techedsr.sk](http://www.techedsr.sk)



# Chybové stavy v REST API

Miroslav Holec | TechEd 2025

[mirek@miroslavholec.cz](mailto:mirek@miroslavholec.cz)

[miroslavholec.cz](https://miroslavholec.cz)

**TechEd**

 **GOPAS**





# Miroslav Holec

KONZULTANT PRO .NET

Firemní školení < REST API a Blazor v .NETu >

Návrh REST API a konzultace k API

Tvorba obsahu



## Videa na YouTube

[youtube.com/mirekholec](https://youtube.com/mirekholec)

## Premium videa

[www.miroslavholec.cz/premium](http://www.miroslavholec.cz/premium)

## Novinky pro vývojáře

[www.dotnetnews.cz](http://www.dotnetnews.cz)

## Průvodce designem REST API

[www.restapi.cz](http://www.restapi.cz)

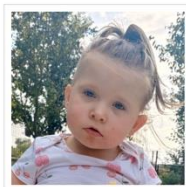


# ADNP ASOCIACE

Est. 2024

~ 5% dětí se narodí se vzácným onemocněním

~ 5% těchto onemocnění lze léčit



adnpasociace.cz



# Agenda

1. Logování a trasování chyb
2. Generické chybové struktury
3. Problem Details RFC
4. Stavové kódy
5. Podpora v ASP.NET Core
6. DEMO

Demo app bude dostupná na

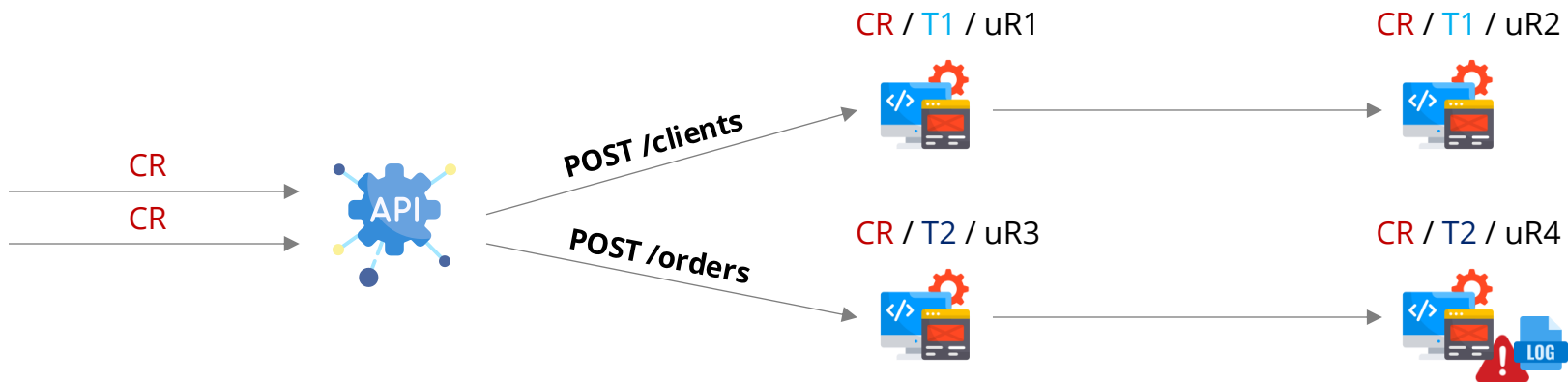
[www.odkaz.me/teched25](http://www.odkaz.me/teched25)



# Logování a trasování chyb

- každý požadavek má přidělen specifický a unikátní identifikátor
- při vzniku chyby dojde k zalogování včetně identifikátoru a vrácení identifikátoru na klienta
- RequestID si generuje server sám, ostatní může přijmout od klienta

<b>RequestId</b>	unikátní pro každý samostatný HTTP požadavek každé služby
<b>TraceId</b>	identifikuje řetěz (trasu) požadavků
<b>CorrelationId</b>	předává se mezi HTTP požadavky pro vysledování komunikace mezi službami



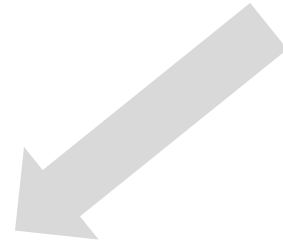
# Status Codes & Generic Error

- **4xx** – chyby způsobené nesprávným požadavkem klienta
- **5xx** – chyby na straně serveru
- **všechny chybové struktury mají stejné společné vlastnosti**
- tyto generické struktury lze dále dle scénáře rozšiřovat

```
public class Error
{
    public string Title { get; set; }
}
```

```
{
    "title": "Bad Request"
}
```

```
{
    "title": "Low balance",
    "credit": 100,
    "price": 5000,
}
```



# RFC 9457 – Problem Details / dříve 7807

- primárně navržen jako popis chyby v **JSON** a **XML**
- chybové situace se rozlišují dle **type**, což má jako hodnotu URI – typicky odkaz na vysvětlení

type	povinný	string	about:blank nebo URI (včetně TAG; RFC 4151)
status	nepovinný	int	musí se shodovat s HTTP status code
title	nepovinný	string	neměnný pro daný typ chyby
detail	nepovinný	string	neparsovatelný, detailní popis chyby
instance	nepovinný	string	URI s odkazem na specifický výskyt

- sktrukturu lze rozšířit o libovolné vlastnosti
- doporučuje se používat v URI absolutní cestu



# RFC 9457 – Problem Details

HTTP/1.1 404 Not Found

Content-Type: **application/problem+json**

```
{  
  "type": "https://example.com/probs/resource-not-found",  
  "title": "Product not found",  
  "status": 404,  
  "detail": "Product 883746 not found",  
  "instance": "/products/883746"  
}
```

# Content Negotiation

- princip vychází z **RFC 9110, section 12** a popisuje vyjednávání o obsahu
- hlavičky: **Accept** : application/xml, **Accept-Language** : cs, **Accept-Encoding** : gzip

HTTP/1.1 404 Not Found












Content-Type: **application/problem+xml**

```
<?xml version="1.0"?>
<problem xmlns="urn:ietf:rfc:9457">
  <type>https://example.com/probs/resource-not-found</type>
  <title>Produkt nebyl nalezen</title>
  <status>404</status>
  <detail>Produkt 883746 nebyl nalezen</detail>
  <instance>/products/883746</instance>
</problem>
```

# Stavové kódy

- 400 Bad Request univerzální chybový stav, validační chyby
- 401 Unauthorized chybí token, neplatný token, selhala autentizace
- 402 Payment Required v praxi se používá pro operace vyžadující platbu
- 403 Forbidden neautorizován, lze maskovat i stavem 404 Not Found
- 404 Not Found server nenašel daný resource (nerozlišuje permanentnost)
- 408 Request Timeout server doposud neobdržel kompletní požadavek od klienta (upload)
- 409 Conflict primárně navržen pro konkurenci například u PUT metody
- 410 Gone definitivně zrušená URL nebo resource (odstraněno)
- 405 Method Not Allowed manipulace s resource skrze HTTP metodu, která není povolena
- 406 Not Acceptable klient vyžaduje formát dat (JSON, XML, ...), který není podporován
- 415 Unsupported Media T. klient posílá formát dat (JSON, XML, ...), který není podporován
- 500 Internal Server Error chyba na straně serveru

# Stavové kódy v ASP.NET Core – implicitně

- 400 Bad Request  MVC data annotations
- 401 Unauthorized 
- 402 Payment Required 
- 403 Forbidden 
- 404 Not Found  jen při routování
- 408 Request Timeout 
- 409 Conflict 
- 410 Gone 
- 405 Method Not Allowed 
- 406 Not Acceptable  jen MVC pomocí Behavior Options
- 415 Unsupported Media T. 

# ASP.NET Core

- framework zavádí chybové struktury ~~ValidationProblemDetails~~ : **ProblemDetails**
- vrací se z MVC i Minimal APIs v podobě **Results.BadRequest()**, **Results.Problem()**, atd.
- middlewares **UseExceptionHandler()** & **UseStatusCodePages()**
- customizace struktur pomocí **AddProblemDetails()**
- dvě strategie použití
  1. vracet z endpointů **ProblemDetails** a ty pouze přizpůsobit v **AddProblemDetails**
    - *pro malá API a mikroslužby, kde je většina kódu přímo v endpointu (controlleru)*
    - *pro vývojáře, kterým nevyhovují výjimky a rádi si komplikují život*
  2. **vracet kdekoliv výjimky a ty překlopit na chybové stavy pomocí ExceptionHandlerů** ✓

# Fluent Validation

- univerzální validační mechanismus, prakticky **nezbytný u Minimal APIs**
- každý kontrakt může mít N validátorů, které se registrují do kontejneru jako **IValidator<T>**
- validátor ověřuje nejen primitivní typy a pravidla, ale může pracovat i s různými službami
- podpora lokalizace, různá rozšíření skrze další balíčky, adaptéry pro různé technologie (Blazor, ...)

```
public class CustomerValidator : AbstractValidator<Customer>
{
    public CustomerValidator()
    {
        RuleFor(x => x.Surname).NotEmpty();
        RuleFor(x => x.Forename).NotEmpty().WithMessage("Please specify a first name");
        RuleFor(x => x.Discount).NotEqual(0).When(x => x.HasDiscount);
        RuleFor(x => x.Address).Length(20, 250);
        RuleFor(x => x.Postcode).Must(BeAValidPostcode).WithMessage("Please specify a valid postcode");
    }

    private bool BeAValidPostcode(string postcode)
    {
        // custom postcode validating logic goes here
    }
}
```

{ ? >

# Závěr

- zapojit **middleware** nebo filtr pro generování RequestId (TraceId, CorrelationId)
- pro všechny chyby generovat stejnou strukturu, ideálně **Problem Details** (RFC 9457)
- odchyťávat chyby pomocí **ExceptionHandler** a zbytek **UseStatusCodePages**
- definovat si vlastní nastavení struktur pomocí **AddProblemDetails**( + CustomizeProblemDetails)
- ukončit zpracování požadavku **výjimkou** a tu globálně handlovat
- validace vyřešit pomocí **Fluent Validation**

Demo app bude dostupná na

[www.odkaz.me/teched25](http://www.odkaz.me/teched25)



# TechEd

Největší česko-slovenská  
**konference o IT**

Miroslav Holec | TechEd 2025

[mirek@miroslavholec.cz](mailto:mirek@miroslavholec.cz)

[miroslavholec.cz](https://miroslavholec.cz)

