Razvoj Web Aplikacija

Predavanje 2

Ponavljanje

- Web API i klijent/server paradigma (*high-level way of doing things*)
- REST – Representational State Transfer
- Zahtjev/odgovor
  - Zahtjev traži da se dohvati ili utječe na resurs
  - Osobine zahtjeva: URL, HTTP metoda/verb, zaglavlja, tijelo (npr. JSON)
  - Odgovor opisuje što se dogodilo s resursom ili stanjem resursa
  - Osobine odgovora: status kod, zaglavlja, tijelo (npr. JSON)
- HTTP kodovi odgovora
  - 100-199 – info
  - 200-299 – uspjeh
  - 300-399 – preusmjeravanje
  - 400-499 – klijentska greška
  - 500-599 – poslužiteljska greška

Web API: anatomija akcije

- HTTP metode
- Usmjeravanje (engl. Routing)
- Modeli
- Rezultati akcija (engl. Action results)

HTTP metode

- Ponekad se nazivaju i HTTP "verbs" (glagoli)
- Prve korištene HTTP metode: GET (dohvati podatke) and POST (ažuriranje podataka)
- Neke dodatne HTTP metode:
  - **HEAD** – vraća samo zaglavlja
  - **CONNECT** – tuneliranje sigurnog povezivanja - nije povezano s HTTPS-om!
  - **OPTIONS** – slično kao HEAD, vraća komunikacijske opcije, podržane HTTP metode, CORS opcije i slično
  - **TRACE** – u svrhu debugiranja, ponavlja tijelo zahtjeva u odgovoru
- Dolaskom REST-a proširen je na metode koje se tiču s resursa
  - **PUT** – stvara ili ažurira stanje resursa, isto kao i POST; glavna razlika je u tome što PUT mora biti idempotentan, a POST ne mora
  - **DELETE** – uklanja stanje resursa
  - **PATCH** – mijenja stanje resursa prema djelomičnom ažuriranju poslanom u zahtjevu

HTTP methods

- **Idempotentnost** i **sigurnost** – svojstva HTTP metode
- HTTP metoda je **sigurna** ako ni na koji način ne utječe na resurs
- HTTP metoda je **idempotentna** ako uzastopni isti HTTP zahtjevi koji koriste tu metodu proizvode isti učinak na resurs kao jedan jedini HTTP zahtjev
  - GET /product/123 – više zahtjeva ne samo da daje isti rezultat, već je stanje resursa isto
  - PUT /product/123 – također
  - DELETE /product/123 – statusni kod različit (npr. proizvodi 200/204, a zatim 404) ali učinak je isti – resursa više nema
  - POST /product/ – jedan poziv stvara jedan resurs, više poziva stvara više resursa - **nije idempotentno**

HTTP metode

	Tijelo zahtjeva	Tijelo odgovora	Idempotentan	Siguran
GET	Ne	Da	Da	Da
HEAD	Ne	Ne	Da	Da
POST	Da	Da	Ne	Ne
PUT	Da	Da	Da	Ne
DELETE	Ne	Da	Da	Ne
CONNECT	Ne	Da	Ne	Ne
OPTIONS	Ne	Da	Da	Da
TRACE	Ne	Da	Da	Da
PATCH	Da	Da	Ne	Ne

Primjeri HTTP metode sa status kodovima

- Zahtjev: **GET** /product/123
- Slučaj bez pogreške: **status kod** je **200 OK**, **tijelo** je stanje traženog resursa u npr. JSON formatu (može biti i **XML**, ovisno o implementaciji)

- Zahtjev: **POST** /product (pazi ➔ nema identifikatora)
- Slučaj bez pogreške: **status kod** je **200 OK** (ili **201 Created**), **tijelo** je stanje <u>novostvorenog</u> resursa s njegovim ID-om

- Zahtjev: **PUT** /product/123
- Slučaj bez pogreške: **status kod** je **200 OK**
  - **201 Created ➔ tijelo** je stanje <u>novostvorenog</u> resursa
  - **204 No Content ➔ nema tijela**, što znači da je resurs stvoren kako je zatraženo

Primjeri HTTP status koda pogreške

- Status kod je **400** bez tijela, što znači da URL ili tijelo nisu točni prema onome što poslužitelj očekuje (neke informacije nedostaju?)
- Status kod je **401** (Unauthorized), znači da poslužitelj nema obvezne informacije o identitetu podnositelja zahtjeva
- Status kod je **403** (Forbidden), znači da tražitelj nema dovoljno dozvola za npr. stvaranje novog objekta
- Status kod je **404** (Not Found) bez tijela, što znači da npr. proizvod nije moguće pronaći
- Status kod je **410** (Gone) bez tijela, što znači da je npr. proizvod u nekom trenutku bio dostupan, ali više nije
- Status kod je **500** (Internal server error)
- …

Usmjeravanje

- Kada Web API primi zahtjev, on instancira kontroler i izvršava njegovu metodu
- Usmjeravanje temeljeno na atributima (Attribute based routing) vs. Konvencionalno usmjeravanje (Conventional routing)
- Zadano ponašanje – *attribute based routing*
  - Web-API pronalazi kontroler po nazivu
  - Metoda mora biti akcija (označena kao akcija s atributom HttpXxx)
  - Web-API ne traži koju akciju pokrenuti preko njenog naziva, već naziva HTTP metode (HttpXxx)
- Može se postaviti – *conventional routing*
  - Možemo postaviti uzorak za Web API za korištenje kontrolera / akcije

```
app.MapControllerRoute(name: "default", pattern: "{controller=Home}/{action=Index}/{id?}");

app.MapDefaultControllerRoute();
```

  - Web API ne bi trebao koristiti konvencionalno usmjeravanje ako je moguće

Usmjeravanje preko atributa

Route atribute – opisuje URL putanju koju treba uskladiti, bez obzira na HTTP metodu

- Primjer: [**Route**("api/[controller]")]

HTTP method atribute – opisuje URL putanju u kontekstu određene HTTP metode

- Primjer 1: [**HttpPut**]
- Primjer 2: [**HttpPut**("create_or_update")]
- Podržani: **Get**, **Post**, **Put**, **Patch**, **Delete**, **Head**, **Options**
- Moguće je dodati nepodržanu HTTP metodu (nasljeđivanje od HttpMethodAttribute)

Usmjeravanje preko atributa (cont.)

Zamjena tokena – zamjenjuje tokene ključnih riječi u predlošcima usmjeravanja, omotane s *[i]*

```
[Route("[controller]")]
[Route("api/[controller]")]
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return ControllerContext.MyDisplayRouteInfo();
    }
}
```

Parametri rute – mogu biti dio rute, omotani s *{ i }*

```
[HttpGet("{id}")]   // GET /api/test2/xyz
public IActionResult GetProduct(string id)
{
    return ControllerContext.MyDisplayRouteInfo(id);
}
```

Usmjeravanje preko atributa (cont.)

- **Dvosmislene akcije** – ako dvije ili više radnji imaju istu HTTP metodu i nijedno drugo svojstvo koje bi ih razlikovalo, smatraju se dvosmislenim (ili nerješivim) jer framework ne zna koju koristiti
  - Primjer: dvije akcije označene s [HttpGet]
- **Akcija najbolji-kandidat** – ako postoji akcija koja se može riješiti jer je "najbliža"
  - Bliskost - na temelju različitih pravila
  - Primjer 1: akcija s oznakom [HttpPost] najbolji je kandidat za POST zahtjev
  - Primjer 2: ako nema akcije označene s [HttpPost] i postoji neoznačena akcija, ta je akcija najbolji kandidat za POST zahtjev
  - Primjer 3: ako postoji više radnji označenih s [HttpPost], ali postoji jedna s određenom putanjom koja odgovara URL-u POST zahtjeva, ta je radnja najbolji kandidat za zahtjev
  - …

Model

- Model je apstrakcija za podatke
- Predstavlja enkapsulirane i tipizirane podatke
- U osnovi, model je samo klasa sa svojstvima

```
public class Pet
{
    public string Name { get; set; }
    public string Breed { get; set; }
}
```

- Također je temeljna struktura koja se koristi za **model binding**

Napomena: počevši od C# 9.0, možemo koristiti record tip

```
public record PetRecord(string Name, string Breed);
```

Model binding

- Model binding je u osnovi je način slanja ulaznih parametara akciji
- Slanje parametara može se obaviti na više načina
  - Iz URL-a
  - Preko query stringa
  - Iz unosa obrazaca i widgeta (uključujući skrivena polja)
  - Iz tijela zahtjeva, u npr. JSON formatu
  - Iz zaglavlja, npr. zaglavlja autentifikacije ili nekog novo implementiranog zaglavlja
- Binding radi s objektima (e.g. **Pet**) i primitivnim tipovima (**int**)

Model binding - primjer

```
[HttpGet("{id}")]
public Pet GetById(int id, bool dogsOnly) { … }
```

Pozivamo metodu pomoću GET zahtjeva sa sljedećim URL-om:

`https://contoso.com/api/pets/2?DogsOnly=true`

Model binding prolazi kroz sljedeće korake:

- Pronalazi prvi parametar metode **GetById**, cijeli broj pod nazivom **ID**.
- Pregledava dostupne izvore u HTTP zahtjevu i pronalazi **id = "2"** u podacima rute.
- Pretvara tekst **"2"** u cijeli broj **2**.
- Pronalazi sljedeći parametar metode **GetById**, boolean po imenu **dogsOnly**.
- Pregledava izvore i pronalazi **"DogsOnly=true"** u query stringu. Kod naziva se ne razlikuju velika i mala slova.
- Pretvara tekst **"true"** u boolean **true**.

Okvir zatim poziva metodu **GetById**, prosljeđujući **2** u **id**, te **true** u **dogsOnly**.

Model binding - pravila

Model binding dobiva podatke u obliku ključ-vrijednost parova iz sljedećih izvora u HTTP zahtjevu, sljedećim redoslijedom:

1. **Polja obrasca**
2. **Tijelo zahtjeva**
   - kontroleri moraju imati **[ApiController]** atribut
3. **URL t.j. podaci o ruti ("route data")**
   - samo za jednostavne tipove
4. **Query string**
   - samo za jednostavne tipove
5. **Prenesene datoteke**
   - samo za **IFormFile** ili **IEnumerable<IFormFile>**

Model binding - izvori

Ako zadano ponašanje ne odgovara, možemo izričito navesti kako vezati izvor.

Model binding atribut izvora:

- [FromQuery] - Dohvaća vrijednosti iz query stringa.
- [FromRoute] - Dohvaća vrijednosti iz podataka o ruti.
- [FromForm] - Dohvaća vrijednosti iz polja obrasca.
- [FromBody] - Dohvaća vrijednosti iz tijela zahtjeva.
- [FromHeader] - Dohvaća vrijednosti iz HTTP zaglavlja.

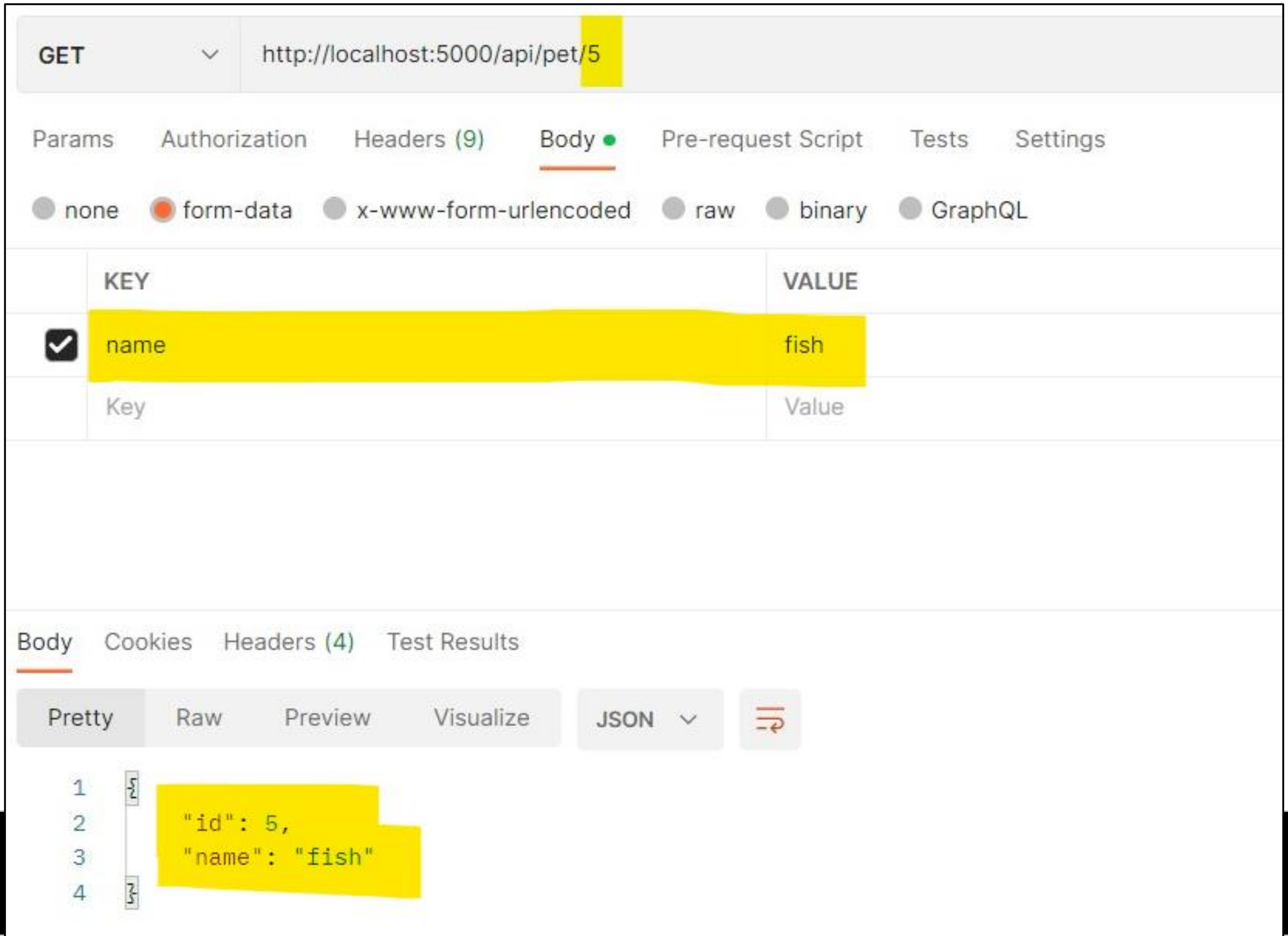
Atribut izvora možete primijeniti na npr. ulazni parametar kao što je ovaj:

```
public ActionResult<Pet> Create([FromBody] Pet pet)
```

Postoji mnogo drugih opcija model bindinga u ASP.NET Web API-ju!

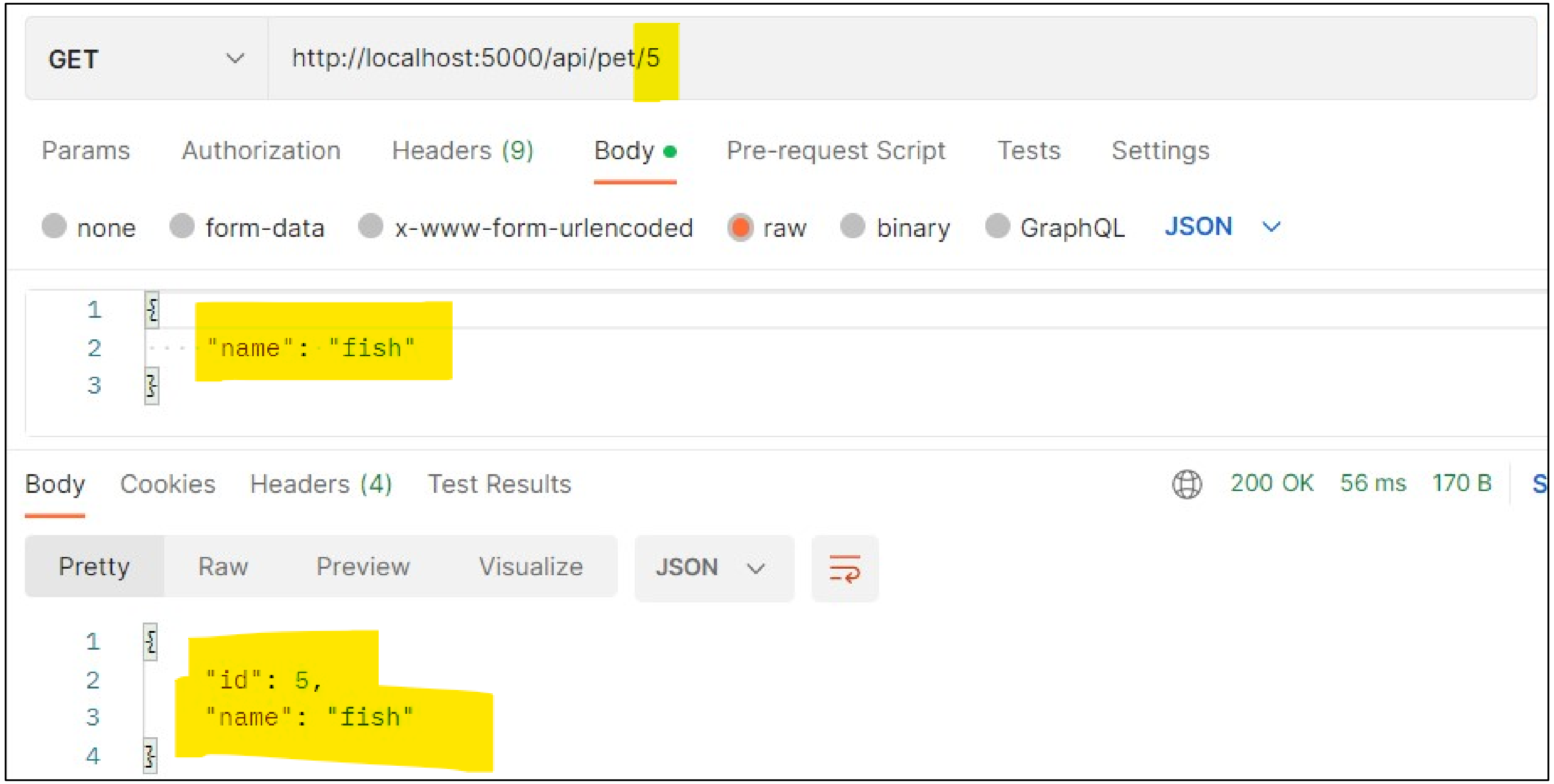
Model binding – FromForm primjer

```
[HttpGet("{id}")]
public PetRecord GetById(int id, [FromForm]string name) {
  return new PetRecord(id, name);
}
```



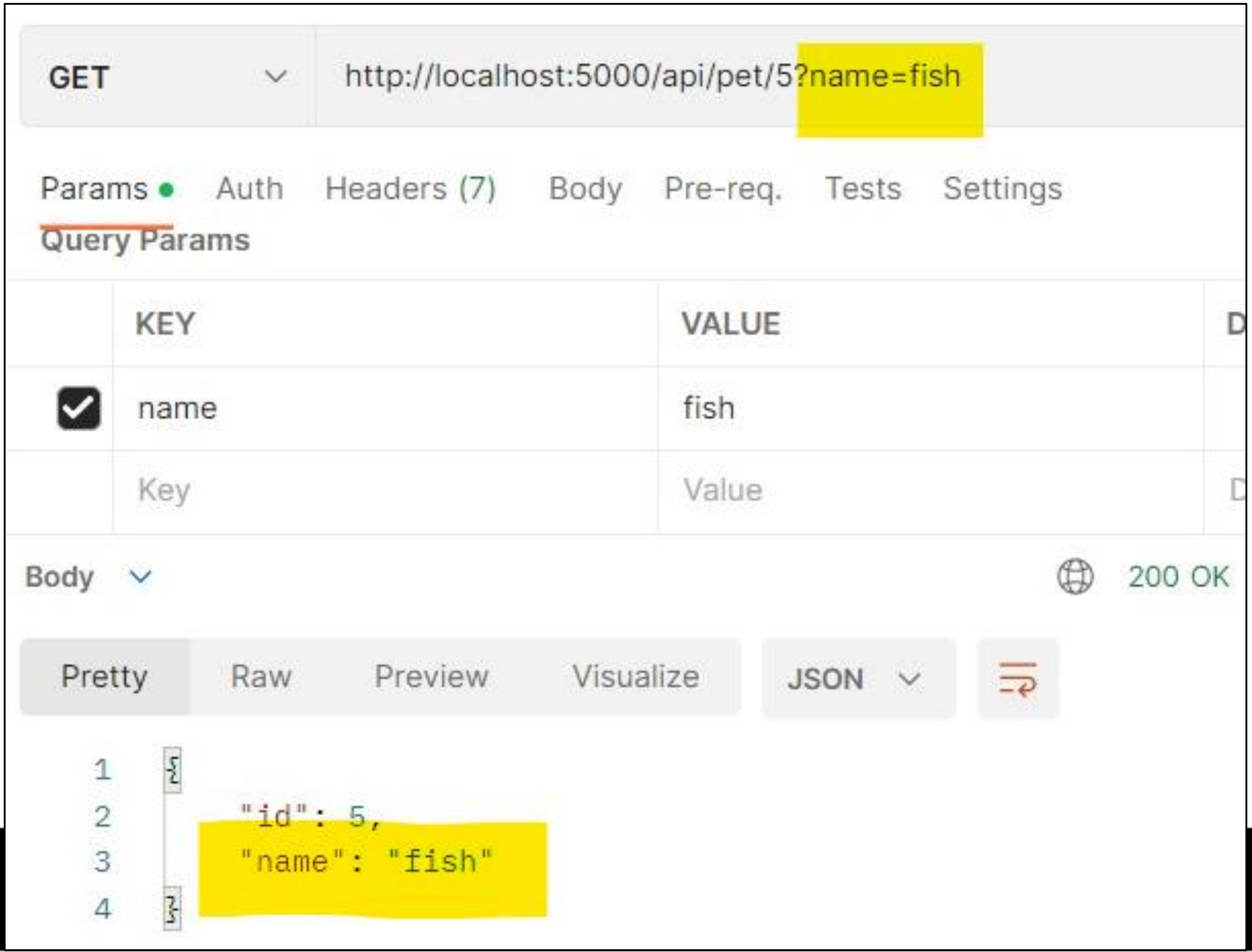
Model binding – FromBody primjer

```
[HttpGet("{id}")]
public PetRecord GetById(int id, [FromBody] PetRecord pet) {
  return new PetRecord(id, pet.Name);
}
```



Model binding – FromQuery primjer

```
[HttpGet("{id}")]
public PetRecord GetById(int id, [FromQuery] string name) {
  return new PetRecord(id, name);
}
```



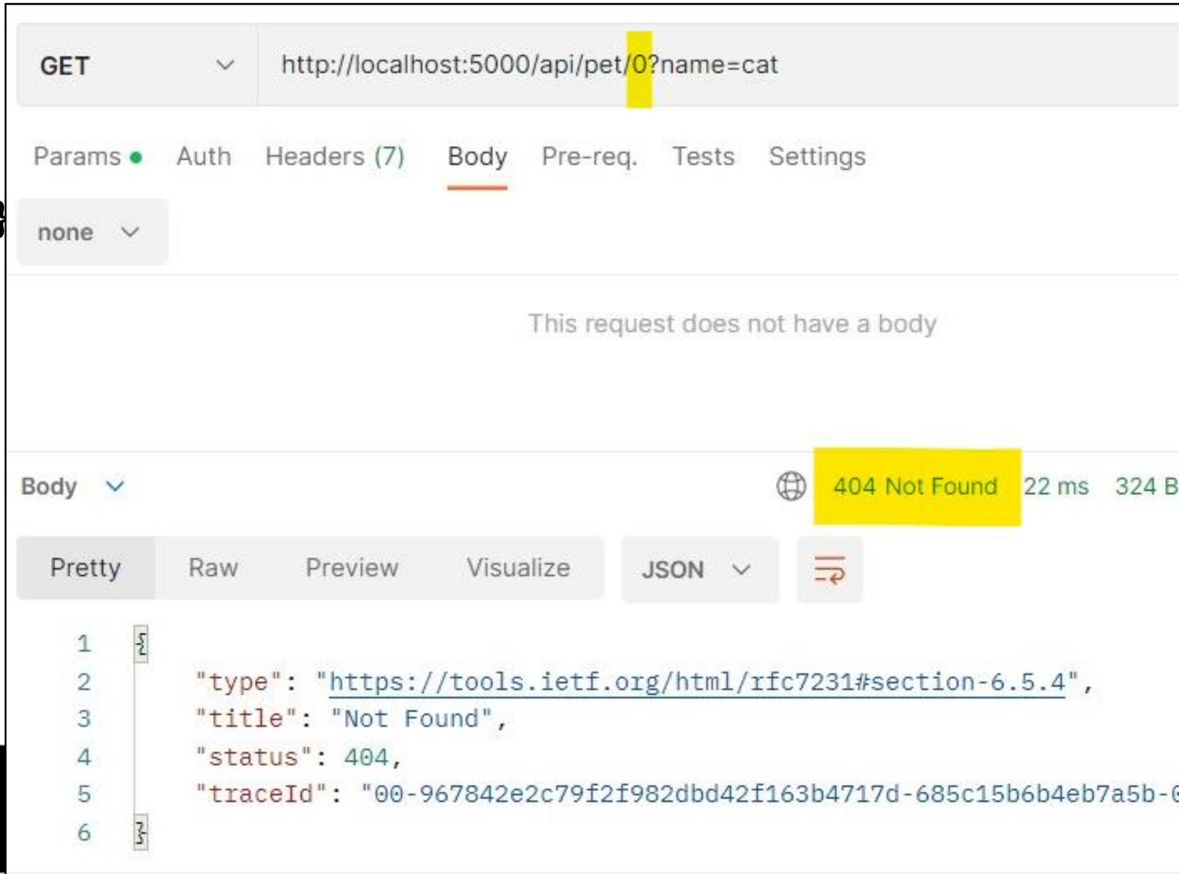
Tipovi rezultata akcije

- U naivnoj verziji, ono što je akcija vrati serijalizira se i vraća klijentu
- Kako onda vratiti pogreške ili preusmjeravanja?
- Rješenje 1 - automatske pogreške: kada provjera valjanosti ne uspije, pogreška **400** automatski se vraća
  - Može biti previše ograničavajuće
- Rješenje 2 – promjena tipa podatka koji se vraća u tip koji podržava vraćanje prilagođenih pogrešaka
  - **ActionResult<T>**
  - **IActionResult** (ili ActionResult)
  - Također: Result<T>, IResult (netipično, osim u Minimal API)

Tipovi rezultata akcije - primjer

```
[HttpGet("{id}")]
public ActionResult<Pet> GetById(int id, string name) {
  if(id==0)
    return NotFound(); // short for return new NotFoundResult();
  else if (string.IsNullOrEmpty(name))
      return BadRequest(); // also shortened

  return new Ok(new Pet { Id = id, Name = name }
}
```



Minimal API - DEMO

Brzo mapiranje HTTP akcije na određenu putanju (bez kontrolera!)

```
app.MapGet("/", () => "Hello World!");
```

Pokretanje servera na određenom portu

```
app.Run("http://localhost:3000");
```

Čitanje konfiguracije

```
var message = app.Configuration["HelloKey"] ?? "Hello";
app.MapGet("/", () => message);
```

Usmjeravanje

```
app.MapGet("/", () => "This is a GET");
app.MapPost("/", () => "This is a POST");
app.MapPut("/", () => "This is a PUT");
app.MapDelete("/", () => "This is a DELETE");
```

Hvala na pažnji!