Web APIs su *Spring Web*

Code Academy

Programuok savo ateitį!

Code Academy
Programuok savo ateitį

Http protokolas

Http tai protokolas, leidžiantis pasiimti resursus, tokius kaip *HTML* dokumentai. Tai yra duomenų perdavimo pagrindas internete.

Tai yra *client-server* protokolas - tai reiškia, jog užklausos yra inicijuojamos kliento, pavyzdžiui interneto naršyklės, o jų apdorojimas vyksta serveryje.

Žinutės, kurios yra siunčiamos iš kliento vadinamos *request*, o serverio grąžinami atsakymai yra vadinami *response*.

Code Academy
Programuok savo ateitį

Request žinutės sandara

- Http metodas (GET, POST, …)
- Kelias iki resurso
- Http protokolo versija
- Antraščių (header) laukai
- Tuščia linija
- Žinutės korpusas (body)

```
POST /resource.html
HTTP/1.1
Content-Type: text/xml;
charset=utf-8
Content-Length: 38

<?xml version="1.0"
encoding="utf-8"?>
```

Code Academy
Programuok savo ateitį!

Response žinutės sandara

- Http protokolo versija
- Statuso kodas, identifikuojantis ar užklausa įvyko sėkmingai
- Statuso žinutė, trumpai apibūdinanti statuso kodą
- Antraščių (header) laukai
- Tuščia linija
- Žinutės korpusas (body)

```
HTTP/1.1 200 OK
Content-Length: 88
Content-Type: text/html

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Code Academy
Programuok savo ateitį

Pagrindiniai *Http* metodai

- ***GET*** - grąžina nurodytą resursą, be jokių kitų pašalinių efektų. Šis metodas neturi korpuso
- ***POST*** - išsaugo naują resursą nurodytame adrese, resursas aprašytas užklausos korpuse
- ***PUT*** - modifikuoja nurodytą resursą, pokyčiai aprašyti užklausos korpuse
- ***DELETE*** - ištrina nurodytą resursą. Šis metodas neturi korpuso

Code Academy
Programuok savo ateitį!

REST API

REST API - sistemos sąsaja, kuri naudoja *Http* metodus (*GET, PUT, POST, DELETE)* norint pasiekti ar modifikuoti duomenis.

Tokių sistemų grąžinami duomenys nėra apdoroti resursai. Pavyzdžiui, internetiniai puslapiai yra neapdoroti duomenys, atiduodami tokiais formatais kaip *JSON* arba *XML*, kurie yra apdirbami jau sąsajos klientų.

REST API dažniausiai naudojamas bendravimui tarp skirtingų internetinių sistemų.

Code Academy
Programuok savo ateitį

RESTful API

RESTful API - vadinamos sistemos, leidžiančios pasiekti jų resursus per ***REST API***, ir tam naudoja visus keturis *Http* metodus - *GET, POST, PUT, DELETE*

HATEOAS - yra *REST* sistemų architektūros dalis. Tai standartas pateikti *REST API* taip, kad klientas galėtų ja naudotis neturėdamas jokių žinių apie sistemą. Kiekvienos užklausos rezultatas grąžina meta informaciją apie kitas galimas užklausas bei jų adresus. Taip klientas gali naudotis sistema nenagrinėdamas sistemos dokumentacijos, o analizuodamas informaciją gautą iš siųstų užklausų.

Code Academy
Programuok savo ateitį

REST API su Spring

```
@RestController
@RequestMapping(path = "/api/hello")
public class HelloWorldController {
    @RequestMapping(path = "/{name}", method = GET)
    public String helloWorld(@PathVariable String name) {
        return "Hello " + name + "!";
    }
}
```

- *RestController* nusako, kad tai *Spring bean* komponentas, atsakingas už Http užklausų aptarnavimą
- *RequestMapping* leidžia nurodyti kelią iki klasės/metodo, bei aptarnaujamus metodus
- *PathVariable* nurodo, kad metodo parametras yra gaunamas iš užklausos path parametro

Code Academy
Programuok savo ateitį

REST API su Spring

```
@RestController
@RequestMapping(path = "/api/person")
public class PersonController {
    @RequestMapping(method = POST)
    public void save(@RequestBody Person person) {
        save(person);
    }
    public static class Person {
        String name;
    }
}
```

- *RequestBody* anotacija nurodo, kad parametro reikšmė yra *Http* užklausos korpusas.
  Jei korpuso turinys yra siunčiamas kaip *json,* jis automatiškai serializuojamas į nurodytą *java* objektą:
  korpusas {"name":"Jonas"} bus serializuojamas į naują *Person* objektą su lauko *name* reikšme lygia “Jonas”

Code Academy
Programuok savo ateitį

REST API su Spring

```
@RestController
@RequestMapping(path = "/api/person")
public class PersonController {
    @RequestMapping(method = POST)
    @ResponseStatus(code = HttpStatus.ACCEPTED)
    public void save(@RequestBody Person person) {
        save(person);
    }
    public static class Person {
        String name;
    }
}
```

- *ResponseStatus* anotacija leidžia nurodyti atsakymo *Http* kodą, šiuo atveju kodas bus 201 (ACCEPTED)

Code Academy
Programuok savo ateitį!

Klaidų valdymas su *Spring REST*

Kuriant internetinius puslapius yra įprasta “sugauti” įvykusias sistemos klaidas ir jas apdoroti taip, kad klientui būtų grąžinamas tik atsakymo *status* kodas bei suprantama klaidos žinutė.

Paprastai tokiam rezultatui pasiekti internetinėse sistemose yra naudojami *Exception mapper* arba *handler* objektai, kurių paskirtis sugauti įvykusias klaidas ir jas paversti į aiškias klaidos žinutes su *status* kodu.

Code Academy
Programuok savo ateitį

Klaidų valdymas su *Spring REST*

Vienas iš būdų naudoti *ExceptionHandler* anotaciją ant metodo, REST kontrolerio lygmenyje

```
@ExceptionHandler({ CustomException1.class, CustomException2.class })
    public void handleException() {
        //
    }
```

Šis metodas bus iškviestas įvykus *CustomException1* arba *CustomException2* klaidoms, **tačiau tik tada, kai klaidos įvyksta toje klasėje, kur yra panaudota anotacija.**

Code Academy
Programuok savo ateitį!

Klaidų valdymas su *Spring REST*

ExceptionHandler anotacija gali būti uždėta ant metodų su skirtingais parašais - metodų, priimančių klaidas kaip parametrus, metodų, nepriimančių jokių parametrų, metodų grąžinančių *Http status* kodus ir t.t.

Visus galimus metodo parašus galima rasti anotacijos faile, *Javadoc* aprašyme.

Code Academy
Programuok savo ateitį

Klaidų valdymas su *Spring REST*

Kitas būdas uždėti *ResponseStatus* anotaciją ant klaidos klasės

```
@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {

}
```

Spring karkasas pats suvaldys šią klaidą ir grąžins tokį kodą, koks yra nurodytas anotacijoje (šiuo atveju tai būtų 404).

Code Academy
Programuok savo ateitį!

Užduotis

Sukurkite REST kontrolerį, kuris įgyvendintų visus keturis *Http* metodus - *GET, POST, PUT, DELETE* manipuliuoti duomenimis gulinčiais *Map* objekte.

Panaudokite *PathVariable, ResponseStatus* ir *RequestBody* anotacijas, sukurkite metodų priimančių json reikšmes.

Įgyvendinkite tvarkingą klaidų valdymą - neradus arba nepavykus pakeisti reikšmių, turėtų būti grąžinami atitinkami atsakymai.

Code Academy
Programuok savo ateitį

Naudingos nuorodos

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview