Validiranje REST API poziva

Validiranje

- Validacija REST API poziva temelji se na provjeri podataka koji su poslani od strane klijenta, npr. Postman aplikacije
- S obzirom da je REST API pozive moguće obavljati bez korisničkog sučelja koje bi moglo uključivati zasebne validacije, nužno je provjeravati podatke na strani poslužitelja
- Za tu namjenu moguće je koristiti „Command” objekte koji služe za prijenos podataka od klijenta do poslužitelja te koji sadržavaju validacijske anotacije za provjeru ispravnosti podataka

Primjer „Command” objekta

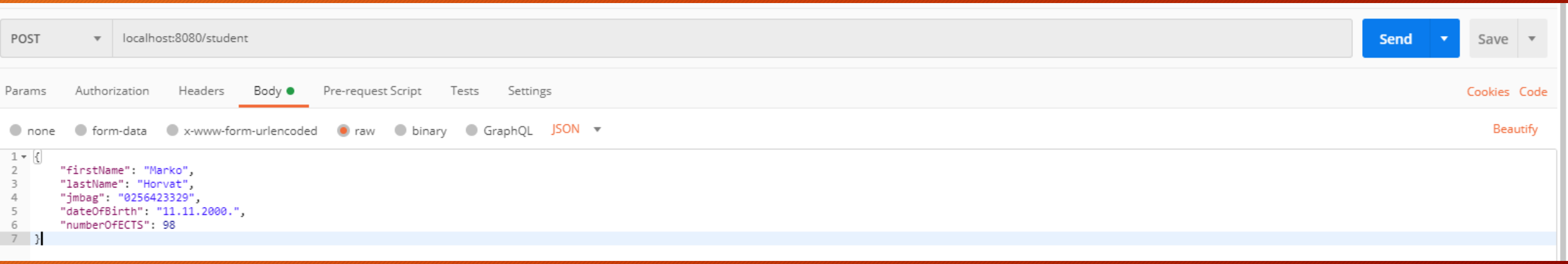
```java
public class StudentCommand {

    @NotBlank(message = "First name must not be empty")
    private String firstName;

    @NotBlank(message = "Last name must not be empty")
    private String lastName;

    @NotBlank(message = "JMBAG must not be empty")
    @Pattern(message = "JMBAG must have 10 digits", regexp="[\\d]{10}")
    private String jmbag;

    @JsonFormat(pattern = "dd.MM.yyyy.")
    @NotNull(message = "Date of birth must be entered")
    @Past(message = "Date of birth must be in the past")
    private LocalDate dateOfBirth;

    @NotNull(message = "Number of ECTS points must be entered")
    @PositiveOrZero(message = "Number of ECTS must be entered as a positive integer")
    @Max(message = "Number of ECTS can not be higher than 480", value = 480)
    private Integer numberOfECTS;
```

Validacijske anotacije

- Validacijske anotacije mogu biti sljedeće:
  - @NotBlank – onemogućava da su podaci nepopunjeni
  - @Pattern – omogućava provjeru usklađenosti vrijednost s regularnim izrazom
  - @JsonFormat – omogućava provjeru je li podatak u JSON format u skladu sa zadanim formatom
  - @NotNull – onemogućava da su vrijednosti podataka postavljeni na „null” vrijednost
  - @PositiveOrZero – onemogućava brojčanim vrijednostima da budu negativne

Spremanje podataka pomoću REST API metode

- U slučaju potrebe spremanja podataka putem POST metode iz REST API sučelja, potrebno je postaviti „Body” vrijednost u JSON obliku putem npr. Postman klijenta na sljedeći način:



Anotacija @Valid

- Pomoću @Valid anotacije se aktiviraju provjere definirane na „Command” objektu
- Potencijalne validacijske poruke moguće je komunicirati prema klijentu postavljanjem HTTP statusa, npr. „CREATED”, „CONFLICT” itd.
- Također, za automatsko pretvaranje JSON oblika u objektni oblik potrebno je koristiti anotaciju @RequestBody

Primjer korištenja anotacije @Valid i @RequestBody

```
@PostMapping
public ResponseEntity<StudentDTO> save(@Valid @RequestBody final StudentCommand command){
    return studentService.save(command)
            .map(
                    studentDTO -> ResponseEntity
                            .status(HttpStatus.CREATED)
                            .body(studentDTO)
            )
            .orElseGet(
                    () -> ResponseEntity
                            .status(HttpStatus.CONFLICT)
                            .build()
            );
}
```

ResponseEntity klasa

- Kako bi se kod vraćanja odgovora od REST API metode klijentu definirao HTTP status i podaci koji se šalju, koristi se klasa ResponseEntity
- Moguće je koristiti „builder pattern” kod kreiranja samog objekta:

```
ResponseEntity
        .status(HttpStatus.CONFLICT)
        .build()
```

Primjer PUT metode

```
@PutMapping("/{JMBAG}")
public ResponseEntity<StudentDTO> update(@PathVariable String JMBAG, @Valid @RequestBody
final StudentCommand updateStudentCommand){
    return studentService.update(JMBAG, updateStudentCommand)
            .map(ResponseEntity::ok)
            .orElseGet(
                    () -> ResponseEntity.notFound().build()
            );
}
```

Primjer DELETE metode

```
@ResponseStatus(HttpStatus.NO_CONTENT)
@DeleteMapping("/{JMBAG}")
public void delete(@PathVariable String JMBAG){
    studentService.deleteByJMBAG(JMBAG);
}
```

Pitanja?