Error Handling

Inleiding
We gaan uit van de todo applicatie:
http://javawerkplaats.nl/todo-angular7-springboot2.10-h2.zip
Open de applicatie in IDEA (alleen het spring-boot gedeelte)

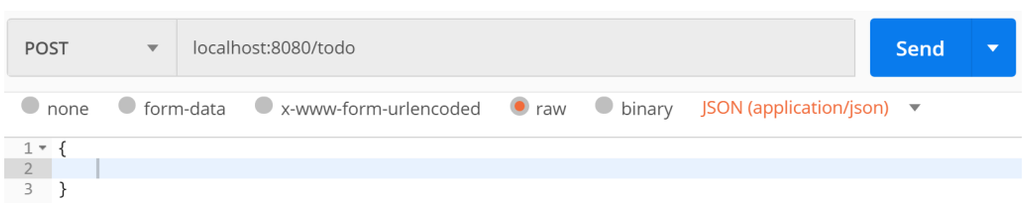
Run
command-line: mvn spring-boot:run
sidebar: Maven - demo - Plugins - spring-boot - spring-boot:run

Test de applicatie in Postman
POST /todo {“task”: “taak1”}
GET /todo

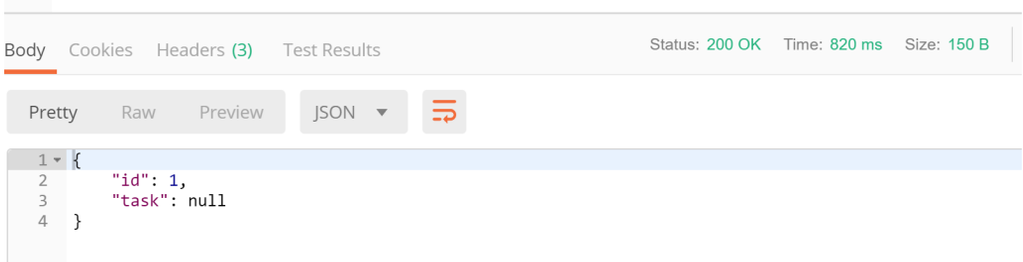
Een exception path
Tot nu toe hebben we altijd alleen op het “happy path” gewerkt.
Nu gaan we kijken wat er mis kan gaan.
En hoe we de gebruiker of andere systemen informeren dat het fout gegaan is.

Probleem 1

Stuur een lege json in een POST (zie onderstaande)




```
@PostMapping("/todo")
public Todo create(@RequestBody Todo todo) {
  return todoService.save(todo);
}
```

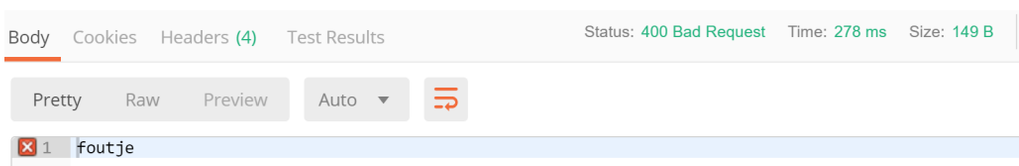


Error path

Als we geen data meegeven in todo dan wordt er **null** in de database geschreven.
De stappen:
1. We moeten controleren of data is ingevuld.
2. Zo ja, dan mag het in de database weggeschreven worden.
3. Zo nee, dan moet de schrijven geblokkeerd worden
   en de gebruiker moet een foutmelding krijgen

Oplossing 1: ResponseEntity

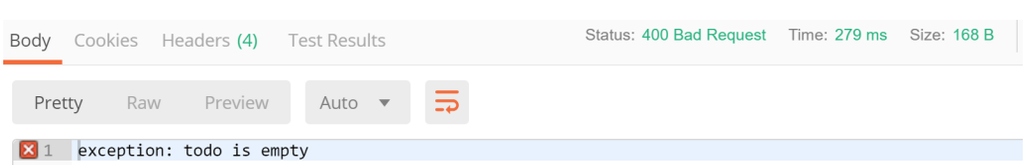
```
@PostMapping("/todo")
ResponseEntity<?> create(@RequestBody Todo todo) throws Exception {
  if (todo.getId() == 0 && todo.getTask() != null )
    return new ResponseEntity<Todo>(todoService.save(todo), HttpStatus.OK);
  else
    return new ResponseEntity<String>("foutje", HttpStatus.BAD_REQUEST);
}
```



Oplossing 2: @ExceptionHandler

```
@PostMapping("/todo")
Todo save(@RequestBody Todo todo) throws Exception {
  if (todo.getId() == 0 && todo.getTask() != null )
    return todoService.save(todo);
  else
    throw new Exception("todo is empty");
}

@ExceptionHandler(Exception.class)
public ResponseEntity<String> exceptionHandler(Exception ex) {
  return new ResponseEntity<>("exception: " + ex.getMessage(),
                              HttpStatus.BAD_REQUEST);
}
```



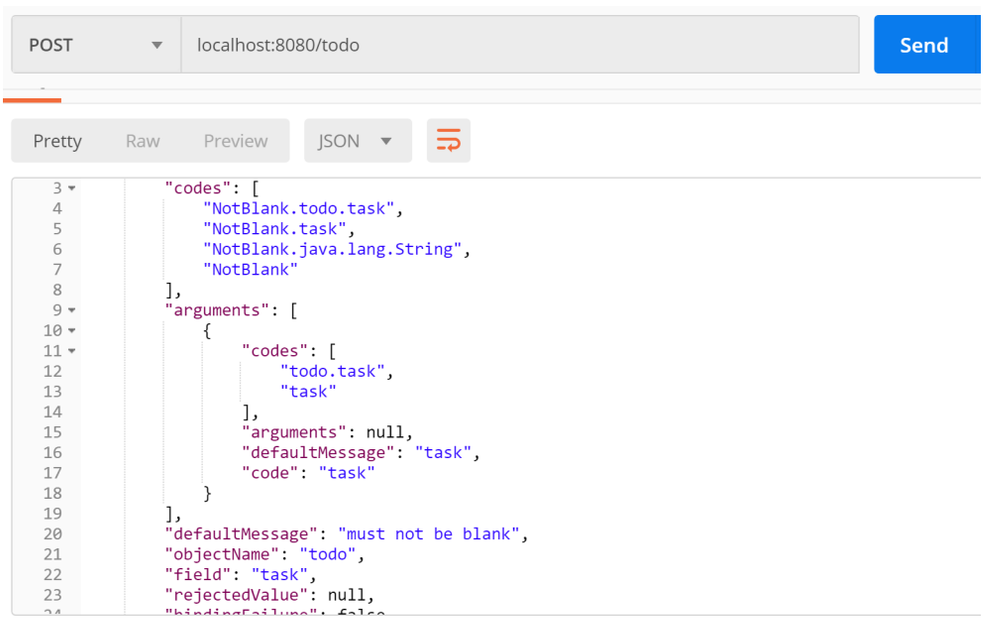
Oplossing 3: Java Validation

```java
@PostMapping("/todo")
Todo save(@Valid @RequestBody Todo todo) throws Exception {
  return todoService.save(todo);
}
@ResponseStatus(HttpStatus.BAD_REQUEST)
@ExceptionHandler(MethodArgumentNotValidException.class)
public List<FieldError>
exceptionHandler(MethodArgumentNotValidException ex) {
  BindingResult result = ex.getBindingResult();
  return result.getFieldErrors();
}
```

```java
// class Todo

@NotBlank
String task;


@NotNull
@Size(min=5)
@Max
@Pattern
@Email
```



Opdracht 1

Geef de create methode foutafhandeling. Gebruik drie bovenstaande oplossingen.
Kortom probeer de bovenstaande code zelf.

Opdracht 2 (Probleem 2)

FindById kan fout gaan als er geen geldige id wordt meegegeven.
Geef de f ndById de benodigde fout afhandeling (NOT_FOUND (gebruik oplossing 1 en 2)

```java
if (todoService.findById(id).isPresent()) {   }
else {   }
```

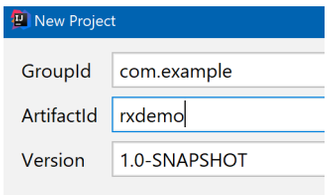
Opdracht (extra) (Probleem 3)

Maak een update methode. Alleen als er een Todo in de database staat.(oplossing 1 en 2)

```java
@PutMapping("/todo")
Todo update(@RequestBody Todo todo) throws Exception {
  // Als todo is aanwezig, anders return error
  return todoService.save(todo);
}
```

Streams en RxJava

Maak in IDEA een Maven project met RxJava




```
<dependencies>
    <dependency>
        <groupId>io.reactivex.rxjava2</groupId>
        <artifactId>rxjava</artifactId>
        <version>2.2.8</version>
    </dependency>
</dependencies>
```

Maak in src/main/java

New :: Java Class

```
public class StreamDemo {
    public static void main(String[] args) {
        List<Integer> lijst = Arrays.asList(1,2,3,4,5,6,7);

        lijst.stream().forEach(System.out::println);

        lijst.stream().filter(x -> x > 3).map(x -> x * 2)
        .forEach(System.out::println);

        int totaal = lijst.stream().reduce(0, (x, acc) -> acc + x );
        System.out.println(totaal);
    }
}
```

Opdracht 3

Maak een lijst van met de tekst
"the","quick","brown","fox","jumped","over","the","lazy","dog"

Gebruik streams zoals in StreamDemo
Print de woorden.
Print alleen de woorden langer dan 4 letters
Voeg de lijst samen tot één String met reduce

```
public class ObservableDemo {
    public static void main(String[] args) {
       List<Integer> lijst = Arrays.asList(1, 2, 3, 4, 5, 6, 7);

       Observable.fromIterable(lijst).subscribe(System.out::println);

       Observable.fromIterable(lijst).filter(x -> x > 3).map(x -> x * 2)
       .forEach(System.out::println);

       Observable.fromIterable(lijst).reduce(0, (x, acc) -> acc + x)
       .subscribe(System.out::println);
    }
}
```

Opdracht 4

Maak een lijst van met de tekst
"the","quick","brown","fox","jumped","over","the","lazy","dog"

Gebruik Observables zoals in ObservableDemo
Print de woorden.
Print alleen de woorden langer dan 4 letters
Voeg de lijst samen tot één String met reduce

Zip demo

```
private static void zipdemo() {
    var letters = Arrays.asList("a", "b", "c", "d", "e", "f", "g");
    var letter$ = Observable.fromIterable(letters);
    var teller$ = Observable.range(1, Integer.MAX_VALUE);
    var zip$ = letter$.zipWith(teller$,
            (letter, teller) -> String.format("%2d. %s", teller, letter));
    zip$.subscribe(System.out::println);
}
```

Opdracht 5

Maak een lijst van met de tekst
"the","quick","brown","fox","jumped","over","the","lazy","dog"

Indexeer de lijst zoals in bovenstaande zipdemo

RxJava Error Handling

```
Observable<Integer> myObserver = Observable.create(subscriber -> {
    subscriber.onNext(10);
    //subscriber.onComplete();
    subscriber.onError(new Exception("foutje"));

});

myObserver.subscribe(
        i -> System.out.println("next " + i),
        error -> System.out.println("error "  + error.getMessage()),
        () -> System.out.println("completed")
);
```

Angular RXJS Error Handling

Angular heeft RxJS, maar het werkt hetzelfde als RxJava.

Opdracht 6

Voeg fout afhandeling toe aan de subscribe.

Eerst: als console.log (zie onderstaande code)

Als tweede: geef een fout melding aan de gebruiker in de html.

```
public save() {
  this.todoService.save(this.todo).subscribe(
    next: () => this.todoList.reloadAll(),
    error: (e) => console.log("foutje " + e),
    complete: () => console.log("complete")
  );
}
```

Angular Validation

Angular heeft ReactiveForms. Hierin kun je default values en validators opnemen.

https://angular.io/guide/form-validation

https://medium.com/front-end-weekly/reactive-forms-and-form-validation-with-angular-fdcbcf98e1e8

Opdracht (extra)

Verander de todoForm in een reactive form.

Voeg de volgende regels toe.

Een task moet ingevuld worden en ten minste 5 letters bevatten.

Toon een fout melding als de validatie niet voldoet.