Handling validation exceptions in Spring Boot

Lecture notes

Built-in validation: Exceptions

- If constraint violations are detected during validation, exceptions are thrown
  - `MethodArgumentNotValidException` is thrown when using `@Valid`
  - `MethodArgumentNotValidException` is thrown when using other built-in validation annotations

Built-in validation: reporting errors

- In the server log, these exceptions result in the output of very long, almost unreadable messages reporting the errors
  - This is what it looks like when copied into Notepad and a few new lines inserted

```
Resolved [org.springframework.web.bind.MethodArgumentNotValidException:
Validation failed for argument [0] in public com.gdmatstaffs.validation_demo.dto.BookDTO
com.gdmatstaffs.validation_demo.book.BookRestController.createBook(com.gdmatstaffs.validation_demo.dto.BookDTO)
with 5 errors: [Field error in object 'bookDTO' on field 'id': rejected value [-1]; codes [Min.bookDTO.id,Min.id,
Min.int,Min]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [bookDTO.id,id];
arguments []; default message [id],1]; default message [must be greater than or equal to 1]] [Field error in object
'bookDTO' on field 'author': rejected value []; codes [NotBlank.bookDTO.author,NotBlank.author,NotBlank.java.lang.String,
NotBlank]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [bookDTO.author,author];
arguments []; default message [author]]; default message [must not be blank]] [Field error in object 'bookDTO' on field
'isbn': rejected value []; codes [NotBlank.bookDTO.isbn,NotBlank.isbn,NotBlank.java.lang.String,NotBlank]; arguments
[org.springframework.context.support.DefaultMessageSourceResolvable: codes [bookDTO.isbn,isbn]; arguments []; default
message [isbn]]; default message [must not be blank]] [Field error in object 'bookDTO' on field 'numberOfCopies':
rejected value [-1]; codes [Min.bookDTO.numberOfCopies,Min.numberOfCopies,Min.int,Min]; arguments [org.springframework.
context.support.DefaultMessageSourceResolvable: codes [bookDTO.numberOfCopies,numberOfCopies]; arguments []; default
message [numberOfCopies],0]; default message [must be greater than or equal to 0]] [Field error in object 'bookDTO' on
field 'title': rejected value []; codes [NotBlank.bookDTO.title,NotBlank.title,NotBlank.java.lang.String,NotBlank];
arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [bookDTO.title,title]; arguments [];
default message [title]]; default message [must not be blank]] ]
```

Built-in validation: reporting errors

- Setting the message attribute of the validation annotations helps…
  - … but not a lot

```
public class BookDTO
{
    @Min(value = 1, message = "Id must be greater than zero")
    private int id;

    @NotBlank(message = "Title cannot be blank")
    private String title;

    @NotBlank(message = "Author cannot be blank")
    private String author;

    @NotBlank(message = "ISBN must be present")
    private String isbn;

    @Min(value = 0, message = "Number of copies must be zero or more")
    private int numberOfCopies;

    private Collection<CopyDTO> copies;
}
```

```
Resolved [org.springframework.web.bind.MethodArgumentNotV
Validation failed for argument [0] in public com.gdmatsta
com.gdmatstaffs.validation_demo.book.BookRestController.c
with 5 errors: [Field error in object 'bookDTO' on field
codes [NotBlank.bookDTO.author,NotBlank.author,NotBlank.j
arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [bookDTO.author,author];
arguments []; default message [author]]; default message [Author cannot be blank]] [Field error in object 'bookDTO' on
field 'title': rejected value []; codes [NotBlank.bookDTO.title,NotBlank.title,NotBlank.java.lang.String,NotBlank];
arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [bookDTO.title,title]; arguments [];
default message [title]]; default message [Title cannot be blank]] [Field error in object 'bookDTO' on field 'isbn':
rejected value []; codes [NotBlank.bookDTO.isbn,NotBlank.isbn,NotBlank.java.lang.String,NotBlank]; arguments
[org.springframework.context.support.DefaultMessageSourceResolvable: codes [bookDTO.isbn,isbn]; arguments [];
default message [isbn]]; default message [ISBN must be present]] [Field error in object 'bookDTO' on field
'numberOfCopies': rejected value [-1]; codes [Min.bookDTO.numberOfCopies,Min.numberOfCopies,Min.int,Min];
arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes
[bookDTO.numberOfCopies,numberOfCopies]; arguments []; default message [numberOfCopies],0]; default message
[Number of copies must be zero or more]] [Field error in object 'bookDTO' on field 'id': rejected value [-1]; codes
[Min.bookDTO.id,Min.id,Min.int,Min]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes
[bookDTO.id,id]; arguments []; default message [id],1]; default message [Id must be greater than zero]] ]
```

Built-in validation: @ExceptionHandler

- Writing exception handler methods in the controller is a better way to report `MethodArgumentNotValidException`

```
@ResponseStatus(HttpStatus.BAD_REQUEST)
@ExceptionHandler(MethodArgumentNotValidException.class)
public Map<String, String> handleValidationExceptions(MethodArgumentNotValidException ex)
{
    Map<String, String> errors = new HashMap<>();

    ex.getBindingResult()
            .getAllErrors()
            .forEach(
                    (error) ->
                    {
                        String fieldName = ((FieldError) error).getField();
                        String errorMessage = error.getDefaultMessage();
                        errors.put(fieldName, errorMessage);
                    });
    return errors;
}
```

The @ExceptionHandler annotation marks a method as the handler for specified exceptions.
Spring Boot calls this method when the specified exception is thrown

Built-in validation: @ExceptionHandler

- The exception handler method returns a map containing the errors
  - The field names are the keys
  - The messages are the values
- The map is serialised as a JSON string and returned as the response to the HTTP PUT request

```
{
  "author": "Author cannot be blank",
  "isbn": "ISBN must be present",
  "id": "Id must be greater than zero",
  "title": "Title cannot be blank",
  "numberOfCopies": "Number of copies must be zero or more"
}
```

- Much better!

Built-in validation: @ExceptionHandler

- Another exception handler method is needed in the controller to report `ConstraintViolationException`

In `BookRestController`

```
@ExceptionHandler(ConstraintViolationException.class)
@ResponseStatus(HttpStatus.BAD_REQUEST)
public ResponseEntity<String> handleConstraintViolationExceptions(ConstraintViolationException ex)
{
    String violations =
            ex.getConstraintViolations() Set<ConstraintViolation<?>>
                    .stream() Stream<ConstraintViolation<?>>
                    .map(v -> v.getMessageTemplate()) Stream<String>
                    .collect(Collectors.joining( charSequence: "; "));

    return new ResponseEntity<>( body: "Validation error: " + violations, HttpStatus.BAD_REQUEST);
}
```

- The result…

```
GET http://localhost:8080/book/-3

HTTP/1.1 400
Content-Type: text/plain;charset=UTF-8
Content-Length: 46
Date: Wed, 19 Oct 2022 15:51:31 GMT
Connection: close

Validation error: Id must be greater than zero
```

Built-in validation: @ExceptionHandler

- Problem
  - Exception handler methods are needed in each controller that performs validation
- A solution
  - Write a global `ControllerAdvice` to handle all exceptions
  - (Background reading)

Built-in validation: testing for errors

```
@WebMvcTest(BookRestController.class)
@AutoConfigureMockMvc
class ValidationDemoApplicationTests
{
    @MockBean
    BookService bookService;

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void when_PostRequestToCreateBookHasInvalidBook_expect_ErrorMessages() throws Exception
    {
        String badBookDTO =
                "{\"id\": -1, \"title\": \"\", \"author\": \"\", \"isbn\": \"\", \"numberOfCopies\": -1}";
        mockMvc
            .perform(MockMvcRequestBuilders.post( urlTemplate: "/book/create")
            .content(badBookDTO)
            .contentType(MediaType.APPLICATION_JSON))

            .andExpect(MockMvcResultMatchers.status().isBadRequest())
            .andExpect(MockMvcResultMatchers.jsonPath(
                    expression: "$.id", Is.is( value: "Id must be greater than zero")))
            .andExpect(MockMvcResultMatchers.jsonPath(
                    expression: "$.title", Is.is( value: "Title cannot be blank")))
            .andExpect(MockMvcResultMatchers.jsonPath(
                    expression: "$.author", Is.is( value: "Author cannot be blank")))
            .andExpect(MockMvcResultMatchers.jsonPath(
                    expression: "$.isbn", Is.is( value: "ISBN must be present")))
            .andExpect(MockMvcResultMatchers.jsonPath(
                    expression: "$.numberOfCopies", Is.is( value: "Number of copies must be zero or more")))
            .andExpect(MockMvcResultMatchers.content().contentType(MediaType.APPLICATION_JSON));
    }
}
```

The map of errors can be tested in an integration test

Additional reading

- Do some background reading about the terms introduced in these notes
  - More about validation in Spring Boot:
    - https://reflectoring.io/bean-validation-with-spring-boot/
    - https://fullstackcode.dev/2022/05/10/complete-guide-to-spring-boot-validation/
    - Externalizing error messages
    - `ControllerAdvice`
  - General exception handling in Spring Boot
    - https://spring.io/blog/2013/11/01/exception-handling-in-spring-mvc
    - https://reflectoring.io/spring-boot-exception-handling/
    - https://www.baeldung.com/exception-handling-for-rest-with-spring