**Managing exceptions in a Spring Boot RESTful API**

Exceptions are inevitable in software code. Numerous factors could cause an exceptional scenario in your code. For instance, in the RESTful API we’ve designed, a user could attempt to access or delete a course with a non-existing course ID. They could also submit a malformed JSON request payload to create a new course through the POST endpoint. All these scenarios cause exceptions in the API.

If we do not handle exception then the default Spring Boot exception handling mechanism is in place. For instance, deleting a course that does not exist in the application presents the error message, as shown in the following listing.

Default exception handling

C:\sbip\repo>http DELETE :8080/resource/10

**HTTP/1.1 500**

{

"error": "Internal Server Error",

"message": "No class com.manning.sbip.ch07.model.Course entity with id

➥ 10 exists!",

"path": "/courses/10",

"status": 500,

"timestamp": "2021-06-23T16:38:20.105+00:00",

"trace": "org.springframework.dao.EmptyResultDataAccessException: No

➥ class com.manning.sbip.ch07.model.Course entity with id 10

➥ exists!\r\n\tat

➥ org.springframework.data.jpa.repository.support.SimpleJpaRepository.lam

➥ bda$deleteById$0(SimpleJpaRepository.java:166)\r\n\tat

➥ java.base/java.util.Optional.orElseThrow(Optional.java:401)\r\n\tat

➥ org.springframework.data.jpa.repository.support.SimpleJpaRepository.del

➥ eteById(SimpleJpaRepository.java:165)\r\n\tat

➥ java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native

➥ Method)\r\n\tat

➥ java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMe

➥ thodAccessorImpl.java:64)\r\n\tat

// Remaining section of the exception is omitted

As you may notice, the above error message is not a desired one and contains details that are not of much use to the API users. It also exposes to the caller information about the tech stack used for the implementation of the API, which is generally considered a security flaw. Further, the HTTP response code is also generic (500 Internal Server Error), which indicates that a server-side error has occurred.

The best practice to handle exception by following:

1. Exception should propagate through repository layer to controller layer.
2. We should define a class which is annotated with @ControllerAdvice
3. Add all the exception handler method for each type of exception.

By using this approach we can handle the exception in a centralized manner.