

Spring Boot REST Error Handling Default And Caught Management

springBoot08

Spring Boot REST Error Handling

- ❑ REST è un'architettura **stateless** in cui i client possono accedere e manipolare le risorse su un server.
- ❑ La gestione degli errori riveste un ruolo importante nella **qualità e affidabilità** dell'applicazione
- ❑ Gli errori lato server devono essere **intercettati** e **segnalati** al client in modo chiaro, preciso ed esaustivo
- ❑ Deve essere rispettato il **protocollo HTTP** sottostante, restituendo al client i più appropriati **codici HTTP**
- ❑ Spring fornisce diversi strumenti per **centralizzare** e **gestire** le condizioni di errore lato server

Spring Boot REST Error Handling HTTP Status Code

- ❑ Quando un client effettua una richiesta a un server HTTP e il server riceve correttamente la richiesta, il server deve notificare al client se la richiesta è stata gestita correttamente o meno
- ❑ In base al response code il client può dedurre il risultato di una certa richiesta
- ❑ HTTP realizza questo con cinque categorie di codici di stato:
 - **100-level** (**Informational**) – server acknowledges a request
 - **200-level** (**Success**) – server completed the request as expected
 - **300-level** (**Redirection**) – client needs to perform further actions to complete the request
 - **400-level** (**Client error**) – client sent an invalid request
 - **500-level** (**Server error**) – server failed to fulfill a valid request due to an error with server

Spring Boot REST Error Handling HTTP Status Code

❑ Il modo più semplice ed efficace di trattare gli errori è di **restituire uno status code appropriato**

❑ Alcuni comuni response code sono:

- **400** *Bad Request* – client sent an invalid request, such as lacking required request body or parameter
- **401** *Unauthorized* – client failed to authenticate with the server
- **403** *Forbidden* – client authenticated but does not have permission to access the requested resource
- **404** *Not Found* – the requested resource does not exist
- **412** *Precondition Failed* – one or more conditions in the request header fields evaluated to false
- **500** *Internal Server Error* – a generic error occurred on the server
- **503** *Service Unavailable* – the requested service is not available

Spring Boot REST Error Handling HTTP Status Code

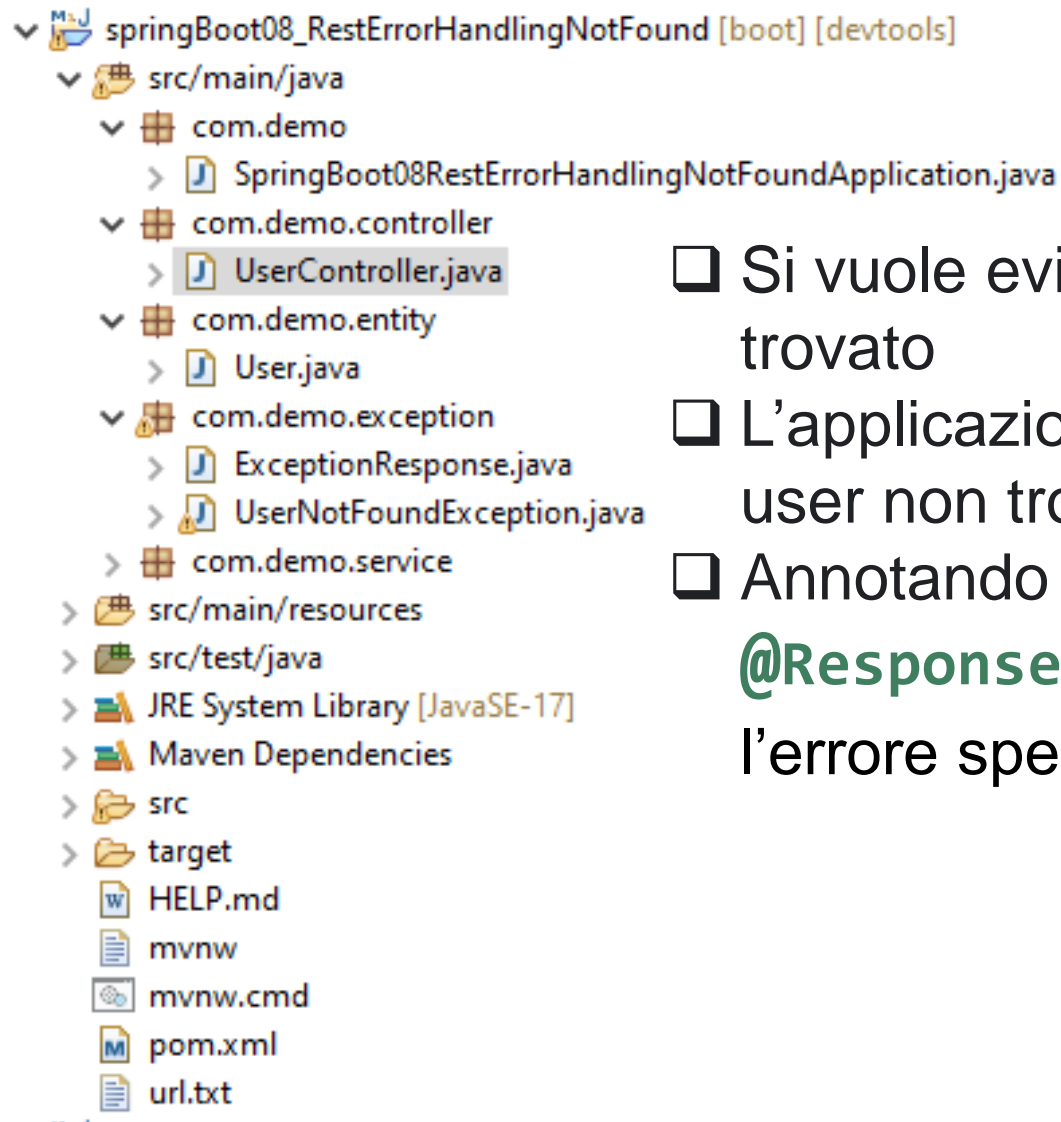
- ❑ Sebbene basilari, questi codici consentono al client di comprendere la natura dell'errore che si è verificato.
- ❑ In molti casi, tuttavia, è necessario fornire dettagli supplementari nel responso fornito al client
- ❑ **HTTP 500** segnala che qualche problema è avvenuto nel server nel trattare la richiesta, generalmente non dovuto a cause applicative o di business
- ❑ Pertanto, per minimizzare errori tipo 500, si dovrebbero intercettare e gestire gli errori interni e restituire l'appropriato codice HTTP, come **404** a fronte di risorsa richiesta non disponibile

Spring Boot REST Error Handling Response Body

- ❑ Uno standard di Response Handler in caso di exception potrebbe essere :
- ❑

```
{ "timestamp": "2019-09-16T22:14:45.624+0000",  
  "status": 403,  
  "error": "auth-0001",  
  "message": "Incorrect username and password",  
  "detail": "Ensure that the username and password are OK",  
  "path": "/api/book/1"  
}
```
- ❑ **error non dovrebbe** coincidere con il response code
- ❑ **message** è inteso presentabile in user interfaces e dovrebbe essere internazionalizzato
- ❑ **detail** è inteso per sviluppatori e la traduzione non è necessaria
- ❑ **path** è la URI della richiesta client

Spring Boot REST Error Handling Not Found



- ☐ Si vuole evitare un errore generico **500** a fronte di user non trovato
- ☐ L'applicazione lancia **UserNotFoundException** a fronte di user non trovato
- ☐ Annotando **UserNotFoundException** con **@ResponseStatus(HttpStatus.NOT_FOUND)** si ottiene l'errore specifico **404**

Spring Boot REST Error Handling Create

- ❑ Il response a fronte di **POST** ha uno status code appropriato (**201**) e contiene solo gli headers

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/users
- Body:**

```
{
  "id": 6,
  "name": "SPFORMAZIONE",
  "dob": "2023-04-24T11:02:18.454+00:00"
}
```
- Response Status:** 201 Created
- Response Time:** 32 ms
- Response Size:** 169 B
- Response Headers:**

KEY	Value
Location	http://localhost:8080/users/6
Content-Length	0
Date	Mon, 24 Apr 2023 12:13:10 GMT
Keep-Alive	timeout=60
Connection	keep-alive

Spring Boot REST Error Handling Not Found 2

UserController.java X UserDaoService.java UserNotFoundException.java ExceptionResponse

```
1 package com.demo.controller;
2 import java.net.URI;
16
17 @RestController
18 public class UserController {
19
20     @Autowired
21     private UserDaoService service;
22
23     @GetMapping("/users")
24     public List<User> retrieveAllUsers() {
25         return service.findAll();
26     }
27
28     //retrieves a specific user detail
29     @GetMapping("/users/{id}")
30     public User retrieveUser(@PathVariable int id) {
31         User user = service.findOne(id);
32         if (user == null)
33             //runtime exception
34             throw new UserNotFoundException("id: " + id);
35         return user;
36     }
37
```

```
38 //method that posts a new user detail and returns the status of the u
39 @PostMapping("/users")
40 public ResponseEntity<Object> createUser(@RequestBody User user) {
41     User savedUser = service.save(user);
42     URI location = ServletUriComponentsBuilder
43         .fromCurrentRequest()
44         .path("/{id}")
45         .buildAndExpand(savedUser.getId())
46         .toUri();
47     return ResponseEntity.created(location).build();
48 }
49 }
```

Spring Boot REST Error Handling Not Found 3

```

1 package com.demo.exception;
2 import org.springframework.http.HttpStatus;
3 import org.springframework.web.bind.annotation.ResponseStatus;
4
5 //@ResponseStatus(HttpStatus.NOT_FOUND)
6 public class UserNotFoundException
7     extends RuntimeException {
8     private static final long serialVersionUID = 1L;
9     public UserNotFoundException(String message) {
10         super(message);
11     }
12 }

```

Spring Boot REST Error Handling Not Found 4

- ❑ Exception **non** annotata **500** non è l'appropriato response code per *Resource Not Found* e **non c'è nessun controllo sul corpo del response**

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/users/55`. The response is a 500 Internal Server Error with a JSON body containing error details.

Params Authorization Headers (7) Body Pre-request Script Tests Settings

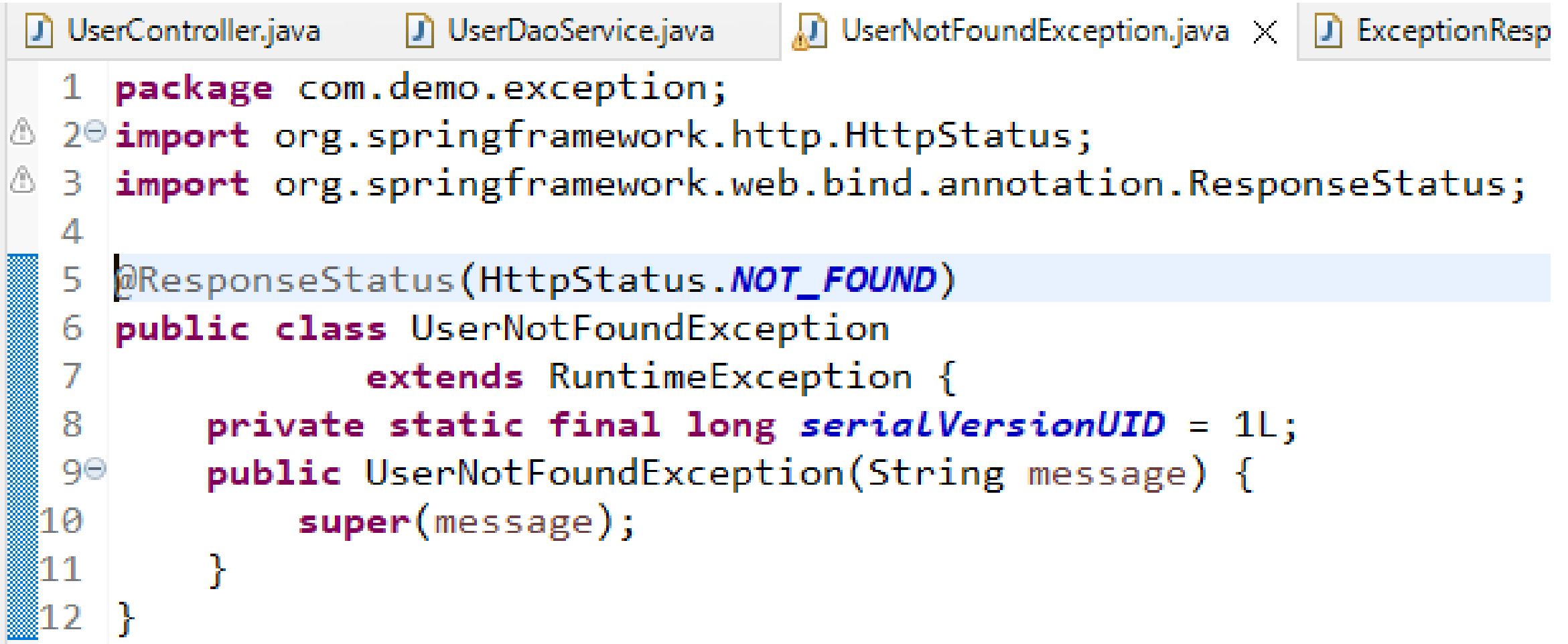
Body Cookies Headers (4) Test Results

Status: 500 Internal Server Error Time: 21.39 s Size: 5.22 KB Save

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2023-04-24T10:50:50.908+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "trace": "com.demo.exception.UserNotFoundException: id: 55\r\n\tat com.demo.controller.UserController.retrieveUser\r\n\t\t(UserController.java:34)\r\n\t\t\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\r\n\t\t\t\tjava.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\r\n\t\t\t\t\tat java.b",
6   "message": "id: 55",
7   "path": "/users/55"
8 }
```

Spring Boot REST Error Handling Not Found 5



```
UserController.java UserDaoService.java UserNotFoundException.java × ExceptionResp
1 package com.demo.exception;
2 import org.springframework.http.HttpStatus;
3 import org.springframework.web.bind.annotation.ResponseStatus;
4
5 @ResponseStatus(HttpStatus.NOT_FOUND)
6 public class UserNotFoundException
7     extends RuntimeException {
8     private static final long serialVersionUID = 1L;
9     public UserNotFoundException(String message) {
10         super(message);
11     }
12 }
```

Spring Boot REST Error Handling Not Found 6

- ❑ Exception **annotata**, **404** è l'appropriato response code per *Resource Not Found*, ma non c'è nessun controllo sul corpo del response

