**Exception handling in spring boot**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.7.10</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.exception</groupId>  
 <artifactId>exceptionexample</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>exceptionexample</name>  
 <description>Demo project for Spring Boot</description>  
 <properties>  
 <java.version>1.8</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Package:model**

**User.java**

package com.exception.exceptionexample.model;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "user")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private long id;  
 @Column(name = "first\_name")  
 private String firstName;  
 @Column(name="last\_name")  
 private String lastName;  
 @Column(name="email")  
 private String email;  
 public User(){  
  
 }  
  
 public User(String firstName, String lastName, String email) {  
 super();  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.email = email;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
}

**package:repository**

**UserRepository(interface)**

package com.exception.exceptionexample.repository;  
  
import com.exception.exceptionexample.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface UserRepository extends JpaRepository<User,Long> {  
}

**package:controller**

**UserController.java**

package com.exception.exceptionexample.controller;  
  
import com.exception.exceptionexample.exception.ResourceNotFoundException;  
import com.exception.exceptionexample.model.User;  
import com.exception.exceptionexample.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping(value = "/api")  
public class UserController {  
 @Autowired  
 UserRepository userRepository;  
 //get all user  
 @GetMapping  
 public List<User> getAllUser(){  
  
 return this.userRepository.findAll();  
 }  
 //get user by id  
 @GetMapping("/{id}")  
 public User getUserById(@PathVariable(value = "id")long id){  
 return this.userRepository.findById(id).orElseThrow(()->new ResourceNotFoundException("User not found with id :" + id));  
  
 }  
  
 //create user  
 @PostMapping  
 public User createUser(@RequestBody User user){  
 return this.userRepository.save(user);  
 }  
  
 //update user  
 @PutMapping("/{id}")  
 public User updateUser(@RequestBody User user,@PathVariable("id") long id){  
 User existingUser=this.userRepository.findById(id).orElseThrow(()->new ResourceNotFoundException("User not found with id :" + id));  
 existingUser.setFirstName(user.getFirstName());  
 existingUser.setLastName(user.getLastName());  
 existingUser.setEmail(user.getEmail());  
 return this.userRepository.save(existingUser);  
 }  
 //delete user by id  
 @DeleteMapping("/{id}")  
 public ResponseEntity<User> deleteUser(@PathVariable("id")long id){  
 User existingUser=this.userRepository.findById(id).orElseThrow(()->new ResourceNotFoundException("User not found with id :" + id));  
 this.userRepository.delete(existingUser);  
 return ResponseEntity.*ok*().build();  
 }  
  
}

**package:exception**

**ErrorDetails.java**

package com.exception.exceptionexample.exception;  
  
import java.util.Date;  
  
public class ErrorDetails {  
 private Date timestamp;  
 private String message;  
 private String details;  
  
 public ErrorDetails(Date timestamp, String message, String details) {  
 super();  
 this.timestamp = timestamp;  
 this.message = message;  
 this.details = details;  
 }  
  
 public Date getTimestamp() {  
 return timestamp;  
 }  
  
 public void setTimestamp(Date timestamp) {  
 this.timestamp = timestamp;  
 }  
  
 public String getMessage() {  
 return message;  
 }  
  
 public void setMessage(String message) {  
 this.message = message;  
 }  
  
 public String getDetails() {  
 return details;  
 }  
  
 public void setDetails(String details) {  
 this.details = details;  
 }  
}

**GlobalExceptionHandler.java:**

package com.exception.exceptionexample.exception;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.context.request.WebRequest;  
  
import java.util.Date;  
  
@ControllerAdvice  
public class GlobalExceptionHandler {  
 // handling specific exception  
 @ExceptionHandler(ResourceNotFoundException.class)  
 public ResponseEntity<?> resourceNotFoundHandling(ResourceNotFoundException resourceNotFoundException, WebRequest request){  
 ErrorDetails errorDetails=new ErrorDetails(new Date(),resourceNotFoundException.getMessage(),request.getDescription(false));  
 return new ResponseEntity(errorDetails, HttpStatus.*NOT\_FOUND*);  
 }  
 @ExceptionHandler(Exception.class)  
 public ResponseEntity<?> globalExceptionHandling(Exception exception,WebRequest request){  
 ErrorDetails errorDetails=new ErrorDetails(new Date(),exception.getMessage(),request.getDescription(false));  
 return new ResponseEntity<>(errorDetails, HttpStatus.*INTERNAL\_SERVER\_ERROR*);  
 }  
}

**ResourceNotFoundException.java**

package com.exception.exceptionexample.exception;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.web.bind.annotation.ResponseStatus;  
  
@ResponseStatus(value= HttpStatus.*NOT\_FOUND*)  
public class ResourceNotFoundException extends RuntimeException{  
 private static final long *serialVersionUID*=1l;  
 public ResourceNotFoundException(String message){  
 super(message);  
 }  
}

**application.properties**

server.port=9901  
  
spring.datasource.url=jdbc:mysql://localhost:3306/usersdb  
spring.datasource.username=root  
#spring.datasource.password=ThePassword  
spring.jpa.hibernate.ddl-auto=update  
  
## Hibernate Properties  
# The SQL dialect makes Hibernate generate better SQL for the chosen database  
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect