

```

1  =====pom.xml=====
2  <?xml version="1.0" encoding="UTF-8"?>
3  <project xmlns="http://maven.apache.org/POM/4.0.0"
4  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
6  https://maven.apache.org/xsd/maven-4.0.0.xsd">
7  <modelVersion>4.0.0</modelVersion>
8  <parent>
9  <groupId>org.springframework.boot</groupId>
10 <artifactId>spring-boot-starter-parent</artifactId>
11 <version>2.3.0.RELEASE</version>
12 <relativePath/> <!-- lookup parent from repository -->
13 </parent>
14 <groupId>com.ameya</groupId>
15 <artifactId>007-RequestValidation</artifactId>
16 <version>0.0.1-SNAPSHOT</version>
17 <name>007-RequestValidation</name>
18 <description>Demo project for Spring Boot</description>
19 <properties>
20 <java.version>1.8</java.version>
21 </properties>
22 <dependencies>
23 <dependency>
24 <groupId>org.springframework.boot</groupId>
25 <artifactId>spring-boot-starter-data-jpa</artifactId>
26 </dependency>
27 <dependency>
28 <groupId>org.springframework.boot</groupId>
29 <artifactId>spring-boot-starter-validation</artifactId>
30 </dependency>
31 <dependency>
32 <groupId>org.springframework.boot</groupId>
33 <artifactId>spring-boot-starter-web</artifactId>
34 </dependency>
35 <dependency>
36 <groupId>mysql</groupId>
37 <artifactId>mysql-connector-java</artifactId>
38 <scope>runtime</scope>
39 </dependency>
40 <dependency>
41 <groupId>org.springframework.boot</groupId>
42 <artifactId>spring-boot-starter-test</artifactId>
43 <scope>test</scope>
44 <exclusions>
45 <exclusion>
46 <groupId>org.junit.vintage</groupId>
47 <artifactId>junit-vintage-engine</artifactId>
48 </exclusion>
49 </exclusions>
50 </dependency>
51 </dependencies>
52 <build>
53 <plugins>
54 <plugin>
55 <groupId>org.springframework.boot</groupId>
56 <artifactId>spring-boot-maven-plugin</artifactId>
57 </plugin>
58 </plugins>
59 </build>
60
61 </project>
62 =====application.properties=====
63 server.port=9001
64 logging.level.web=trace
65
66 spring.datasource.url=jdbc:mysql://localhost:3306/nineleapsdb
67 spring.datasource.username=root
68 spring.datasource.password=root
69 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
70 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
71 spring.jpa.hibernate.ddl-auto=update

```

```

72  =====Person.java=====
73  package com.ameya.models;
74
75  import javax.persistence.Entity;
76  import javax.persistence.GeneratedValue;
77  import javax.persistence.GenerationType;
78  import javax.persistence.Id;
79  import javax.persistence.Table;
80  import javax.validation.constraints.NotBlank;
81  import javax.validation.constraints.NotNull;
82  import javax.validation.constraints.Size;
83
84  @Entity
85  @Table(name="aj_person")
86  public class Person {
87
88      @Id
89      @GeneratedValue(strategy=GenerationType.AUTO)
90      private long id;
91      @NotNull
92      @Size(min = 2, message="First Name Must be atleast 2 Characters")
93      private String firstName;
94      @NotNull
95      @Size(min = 2, message="Last Name Must be atleast 2 Characters")
96      private String lastName;
97      @NotNull
98      @NotBlank
99      private String emailId;
100     @NotNull
101     @Size(min = 2, message="Passport Number Must be atleast 2 Characters")
102     private String passportNumber;
103     public Person() {
104
105     }
106     public long getId() {
107         return id;
108     }
109     public void setId(long id) {
110         this.id = id;
111     }
112     public String getFirstName() {
113         return firstName;
114     }
115     public void setFirstName(String firstName) {
116         this.firstName = firstName;
117     }
118     public String getLastName() {
119         return lastName;
120     }
121     public void setLastName(String lastName) {
122         this.lastName = lastName;
123     }
124     public String getEmailId() {
125         return emailId;
126     }
127     public void setEmailId(String emailId) {
128         this.emailId = emailId;
129     }
130     public String getPassportNumber() {
131         return passportNumber;
132     }
133     public void setPassportNumber(String passportNumber) {
134         this.passportNumber = passportNumber;
135     }
136
137 }
138 =====PersonDAO=====
139 package com.ameya.daos;
140
141 import org.springframework.data.repository.CrudRepository;
142
143 import com.ameya.models.Person;
144

```

```

145 public interface PersonDAO extends CrudRepository<Person, Long> {
146
147 }
148 =====ResourceNotFoundException.java=====
149 package com.ameya.exceptions;
150
151 import org.springframework.http.HttpStatus;
152 import org.springframework.web.bind.annotation.ResponseStatus;
153
154 @ResponseStatus(value=HttpStatus.NOT_FOUND)
155 public class ResourceNotFoundException extends RuntimeException {
156
157     private static final long serialVersionUID = 1L;
158
159     public ResourceNotFoundException(String message) {
160         super(message);
161     }
162
163
164 }
165 =====ErrorDetails.java=====
166 package com.ameya.exceptions;
167
168 import java.util.Date;
169
170 public class ErrorDetails {
171
172     private Date timeStamp;
173     private String message;
174     private String details;
175     public ErrorDetails(Date timeStamp, String message, String details) {
176         super();
177         this.timeStamp = timeStamp;
178         this.message = message;
179         this.details = details;
180     }
181     public Date getTimeStamp() {
182         return timeStamp;
183     }
184     public String getMessage() {
185         return message;
186     }
187     public String getDetails() {
188         return details;
189     }
190
191 }
192 =====PersonController.java=====
193 package com.ameya.controllers;
194
195 import java.util.ArrayList;
196 import java.util.HashMap;
197 import java.util.List;
198 import java.util.Map;
199
200 import javax.validation.Valid;
201
202 import org.springframework.beans.factory.annotation.Autowired;
203 import org.springframework.http.ResponseEntity;
204 import org.springframework.web.bind.annotation.DeleteMapping;
205 import org.springframework.web.bind.annotation.GetMapping;
206 import org.springframework.web.bind.annotation.PathVariable;
207 import org.springframework.web.bind.annotation.PostMapping;
208 import org.springframework.web.bind.annotation.PutMapping;
209 import org.springframework.web.bind.annotation.RequestBody;
210 import org.springframework.web.bind.annotation.RequestMapping;
211 import org.springframework.web.bind.annotation.RestController;
212
213 import com.ameya.daos.PersonDAO;
214 import com.ameya.exceptions.ResourceNotFoundException;
215 import com.ameya.models.Person;
216
217 @RestController

```

```

218 @RequestMapping("/persons")
219 public class PersonController {
220     @Autowired
221     private PersonDAO personDao;
222
223     @GetMapping(path="/getAll")
224     public List<Person> getAllPersons(){
225         List<Person> persons=new ArrayList<>();
226         personDao.findAll().forEach(persons::add);
227         return persons;
228     }
229     @GetMapping(path="/{id}")
230     public ResponseEntity<Person> getPersonById(@PathVariable("id") long id){
231         Person person=personDao.findById(id).orElseThrow(()->
232             new ResourceNotFoundException("Person Not Found ["+id+"]")
233         );
234         return ResponseEntity.ok().body(person);
235     }
236     @PostMapping("/create")
237     public Person createPerson(@Valid @RequestBody Person person) {
238         return personDao.save(person);
239     }
240     @PutMapping("/update/{id}")
241     public ResponseEntity<Person> updatePerson(@PathVariable("id") long id,
242         @Valid @RequestBody Person person) {
243         Person per=personDao.findById(id).orElseThrow(()->
244             new ResourceNotFoundException("Person Not Found ["+id+"]")
245         );
246         per.setEmailId(person.getEmailId());
247         per.setFirstName(person.getFirstName());
248         per.setLastName(person.getLastName());
249         per.setPassportNumber(person.getPassportNumber());
250         Person p=personDao.save(per);
251         return ResponseEntity.ok(p);
252     }
253     @DeleteMapping(path="/delete/{id}")
254     public Map<String,Boolean> deletePerson(@PathVariable("id") long id){
255         Person per=personDao.findById(id).orElseThrow(()->
256             new ResourceNotFoundException("Person Not Found ["+id+"]")
257         );
258         personDao.delete(per);
259         Map<String,Boolean> response=new HashMap<>();
260         response.put("deleted successfully",Boolean.TRUE);
261         return response;
262     }
263 }
264 =====GlobalExceptionHandler.java=====
265
266 package com.ameya.controllers.advice;
267
268 import java.util.Date;
269
270 import org.springframework.http.HttpHeaders;
271 import org.springframework.http.HttpStatus;
272 import org.springframework.http.ResponseEntity;
273 import org.springframework.web.bind.MethodArgumentNotValidException;
274 import org.springframework.web.bind.annotation.ControllerAdvice;
275 import org.springframework.web.bind.annotation.ExceptionHandler;
276 import org.springframework.web.context.request.WebRequest;
277 import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
278
279 import com.ameya.exceptions.ErrorDetails;
280 import com.ameya.exceptions.ResourceNotFoundException;
281
282 @ControllerAdvice
283 public class GlobalExceptionHandler extends ResponseEntityExceptionHandler{
284
285     @ExceptionHandler(ResourceNotFoundException.class)
286     public ResponseEntity<?> resourceNotFoundException(ResourceNotFoundException ex,
287         WebRequest request){
288         ErrorDetails errorDetails=new ErrorDetails(new Date(),
289             ex.getMessage(), request.getDescription(false));

```

```
289         return new ResponseEntity<>(errorDetails,HttpStatus.NOT_FOUND);
290     }
291
292     @Override
293     protected ResponseEntity<Object>
294     handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
295         HttpHeaders headers, HttpStatus status, WebRequest request) {
296         ErrorDetails errorDetails=new ErrorDetails(new Date(),"Validation Failed"
297             , ex.getBindingResult().toString());
298         return new ResponseEntity<Object>(errorDetails,HttpStatus.BAD_REQUEST);
299     }
300 }
301 =====
302
303 http://localhost:9001/persons/create
304 {
305     "firstName":"A",
306     "lastName":"Joshi",
307     "emailId":"",
308     "passportNumber":"ABCD1234"
309 }
```