

```

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>005-EmpCrudService</artifactId>
16 <version>0.0.1-SNAPSHOT</version>
17 <name>005-EmpCrudService</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-web</artifactId>
30 </dependency>
31 <dependency>
32 <groupId>mysql</groupId>
33 <artifactId>mysql-connector-java</artifactId>
34 <scope>runtime</scope>
35 </dependency>
36 <dependency>
37 <groupId>org.springframework.boot</groupId>
38 <artifactId>spring-boot-starter-test</artifactId>
39 <scope>test</scope>
40 <exclusions>
41 <exclusion>
42 <groupId>org.junit.vintage</groupId>
43 <artifactId>junit-vintage-engine</artifactId>
44 </exclusion>
45 </exclusions>
46 </dependency>
47 </dependencies>
48 <build>
49 <plugins>
50 <plugin>
51 <groupId>org.springframework.boot</groupId>
52 <artifactId>spring-boot-maven-plugin</artifactId>
53 </plugin>
54 </plugins>
55 </build>
56
57 </project>
58 =====application.properties=====
59 server.port=9001
60 logging.level.web=trace
61
62 spring.datasource.url=jdbc:mysql://localhost:3306/nineleapsdb
63 spring.datasource.username=root
64 spring.datasource.password=root
65 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
66 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
67 spring.jpa.show-sql=true
68 spring.jpa.hibernate.ddl-auto=update
69 =====Employee.java=====
70 package com.ameya.models;
71

```

```

72 import javax.persistence.Entity;
73 import javax.persistence.Id;
74 import javax.persistence.Table;
75
76 @Entity
77 @Table(name="ajemployee")
78 public class Employee {
79
80     @Id
81     private int id;
82     private String firstName;
83     private String lastName;
84     private double salary;
85     public Employee() {
86
87     }
88     public Employee(int id, String firstName, String lastName, double salary) {
89         super();
90         this.id = id;
91         this.firstName = firstName;
92         this.lastName = lastName;
93         this.salary = salary;
94     }
95     public int getId() {
96         return id;
97     }
98     public void setId(int id) {
99         this.id = id;
100     }
101     public String getFirstName() {
102         return firstName;
103     }
104     public void setFirstName(String firstName) {
105         this.firstName = firstName;
106     }
107     public String getLastName() {
108         return lastName;
109     }
110     public void setLastName(String lastName) {
111         this.lastName = lastName;
112     }
113     public double getSalary() {
114         return salary;
115     }
116     public void setSalary(double salary) {
117         this.salary = salary;
118     }
119     @Override
120     public String toString() {
121         return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" +
122             lastName + ", salary=" + salary
123             + "]";
124     }
125 }
126 =====EmployeeDAO.java=====
127 package com.ameya.daos;
128
129 import org.springframework.data.repository.CrudRepository;
130
131 import com.ameya.models.Employee;
132
133 public interface EmployeeDAO extends CrudRepository<Employee, Integer> {
134
135 }
136 =====EmployeeService.java=====
137 package com.ameya.services;
138
139 import java.util.List;
140
141 import com.ameya.models.Employee;
142
143 public interface EmployeeService {

```

```

144     List<Employee>getAllEmployees();
145     boolean addEmployee(Employee employee);
146     boolean updateEmployee(Integer empId,Employee employee);
147     Employee getEmployee(Integer empId);
148     boolean deleteEmployee(Integer empId);
149 }
150 =====EmployeeServiceImpl.java=====
151 package com.ameya.services.impl;
152
153 import java.util.ArrayList;
154 import java.util.List;
155 import java.util.Optional;
156
157 import org.springframework.beans.factory.annotation.Autowired;
158 import org.springframework.stereotype.Service;
159
160 import com.ameya.daos.EmployeeDAO;
161 import com.ameya.models.Employee;
162 import com.ameya.services.EmployeeService;
163
164 @Service
165 public class EmployeeServiceImpl implements EmployeeService {
166
167     @Autowired
168     EmployeeDAO employeeDao;
169
170     @Override
171     public List<Employee> getAllEmployees() {
172         List<Employee> employees=new ArrayList<>();
173         employeeDao.findAll().forEach(employees::add);
174         return employees;
175     }
176
177     @Override
178     public boolean addEmployee(Employee employee) {
179         return (employeeDao.save(employee)!=null?true:false);
180     }
181
182     @Override
183     public boolean updateEmployee(Integer empId, Employee employee) {
184         Optional<Employee> container=employeeDao.findById(empId);
185         if(!container.isPresent()) {
186             return false;
187         }
188         Employee empToUpdate=container.get();
189         empToUpdate.setFirstName(employee.getFirstName());
190         empToUpdate.setLastName(employee.getLastName());
191         empToUpdate.setSalary(employee.getSalary());
192
193         return (employeeDao.save(empToUpdate)!=null?true:false);
194     }
195
196     @Override
197     public Employee getEmployee(Integer empId) {
198         Optional<Employee> container=employeeDao.findById(empId);
199         Employee emp=null;
200         if(container.isPresent()) {
201             emp=container.get();
202         }
203         return emp;
204     }
205
206     @Override
207     public boolean deleteEmployee(Integer empId) {
208         Optional<Employee> container=employeeDao.findById(empId);
209         if(!container.isPresent()) {
210             return false;
211         }
212         employeeDao.deleteById(empId);
213         return true;
214     }
215
216 }

```

```

217 =====ResourceNotFoundException.java=====
218 ===
219 package com.ameya.exceptions;
220 public class ResourceNotFoundException extends RuntimeException {
221     private static final long serialVersionUID = 1L;
222     public ResourceNotFoundException(String message) {
223         super(message);
224     }
225 }
226
227
228
229 =====ResourceAlreadyExistsException.java=====
230 ===
231 package com.ameya.exceptions;
232 public class ResourceAlreadyExistsException extends RuntimeException {
233     private static final long serialVersionUID = 1L;
234     public ResourceAlreadyExistsException(String message) {
235         super(message);
236     }
237 }
238
239
240
241
242 =====CannotAddException.java=====
243 package com.ameya.exceptions;
244 public class CannotAddException extends RuntimeException {
245     private static final long serialVersionUID = 1L;
246     public CannotAddException(String message) {
247         super(message);
248         // TODO Auto-generated constructor stub
249     }
250 }
251
252
253
254
255 =====CannotUpdateException.java=====
256 package com.ameya.exceptions;
257 public class CannotUpdateException extends RuntimeException {
258     private static final long serialVersionUID = 1L;
259     public CannotUpdateException(String message) {
260         super(message);
261         // TODO Auto-generated constructor stub
262     }
263 }
264
265
266
267
268 =====ExceptionDetails.java=====
269 package com.ameya.exceptions;
270 import java.util.Date;
271 import org.springframework.http.HttpStatus;
272
273 public class ExceptionDetails {
274     private HttpStatus status;
275     private Date timeStamp;
276     private String message;
277     public ExceptionDetails(HttpStatus status, Date timeStamp, String message) {
278         super();
279         this.status = status;
280         this.timeStamp = timeStamp;
281         this.message = message;
282     }
283     public HttpStatus getStatus() {
284         return status;
285     }
286 }

```

```

288     public void setStatus(HttpStatus status) {
289         this.status = status;
290     }
291     public Date getTimeStamp() {
292         return timeStamp;
293     }
294     public void setTimeStamp(Date timeStamp) {
295         this.timeStamp = timeStamp;
296     }
297     public String getMessage() {
298         return message;
299     }
300     public void setMessage(String message) {
301         this.message = message;
302     }
303
304
305 }
306 =====EmployeeExceptionsAdvice.java=====
307 package com.ameya.controllers.advices;
308
309 import java.util.Date;
310
311 import org.springframework.http.HttpStatus;
312 import org.springframework.http.ResponseEntity;
313 import org.springframework.web.bind.annotation.ControllerAdvice;
314 import org.springframework.web.bind.annotation.ExceptionHandler;
315
316 import com.ameya.exceptions.CannotAddException;
317 import com.ameya.exceptions.CannotUpdateException;
318 import com.ameya.exceptions.ExceptionDetails;
319 import com.ameya.exceptions.ResourceAlreadyExistsException;
320 import com.ameya.exceptions.ResourceNotFoundException;
321
322 @ControllerAdvice
323 public class EmployeeExceptionsAdvice {
324
325     @ExceptionHandler(value= {ResourceNotFoundException.class})
326     public ResponseEntity<Object> handleResourceNotFoundException
327     (ResourceNotFoundException ex){
328         ExceptionDetails errorDetails=new ExceptionDetails(
329             HttpStatus.NOT_FOUND,
330             new Date(),
331             ex.getMessage()
332         );
333         return new ResponseEntity<Object>(errorDetails,HttpStatus.NOT_FOUND);
334     }
335     @ExceptionHandler(value= {ResourceAlreadyExistsException.class})
336     public ResponseEntity<Object> handleResourceAlreadyExistsException
337     (ResourceAlreadyExistsException ex){
338         ExceptionDetails errorDetails=new ExceptionDetails(
339             HttpStatus.METHOD_NOT_ALLOWED,
340             new Date(),
341             ex.getMessage()
342         );
343         return new ResponseEntity<Object>(errorDetails,HttpStatus.METHOD_NOT_ALLOWED);
344     }
345     @ExceptionHandler(value= {CannotAddException.class})
346     public ResponseEntity<Object> handleCannotAddException
347     (CannotAddException ex){
348         ExceptionDetails errorDetails=new ExceptionDetails(
349             HttpStatus.INTERNAL_SERVER_ERROR,
350             new Date(),
351             ex.getMessage()
352         );
353         return new
354             ResponseEntity<Object>(errorDetails,HttpStatus.INTERNAL_SERVER_ERROR);
355     }
356     @ExceptionHandler(value= {CannotUpdateException.class})
357     public ResponseEntity<Object> handleCannotUpdateException
358     (CannotUpdateException ex){
359         ExceptionDetails errorDetails=new ExceptionDetails(
360             HttpStatus.INTERNAL_SERVER_ERROR,

```

```

360         new Date(),
361         ex.getMessage()
362     );
363     return new
        ResponseEntity<Object>(errorDetails, HttpStatus.INTERNAL_SERVER_ERROR);
364     }
365 }
366 =====EmployeeController.java=====
367 package com.ameya.controllers;
368
369 import java.util.List;
370
371 import org.springframework.beans.factory.annotation.Autowired;
372 import org.springframework.http.HttpStatus;
373 import org.springframework.http.ResponseEntity;
374 import org.springframework.web.bind.annotation.DeleteMapping;
375 import org.springframework.web.bind.annotation.GetMapping;
376 import org.springframework.web.bind.annotation.PathVariable;
377 import org.springframework.web.bind.annotation.PostMapping;
378 import org.springframework.web.bind.annotation.PutMapping;
379 import org.springframework.web.bind.annotation.RequestBody;
380 import org.springframework.web.bind.annotation.RestController;
381
382 import com.ameya.exceptions.CannotAddException;
383 import com.ameya.exceptions.CannotUpdateException;
384 import com.ameya.exceptions.ResourceAlreadyExistsException;
385 import com.ameya.exceptions.ResourceNotFoundException;
386 import com.ameya.models.Employee;
387 import com.ameya.services.EmployeeService;
388
389 @RestController
390 public class EmployeeController {
391
392     @Autowired
393     private EmployeeService employeeService;
394
395     @GetMapping(path="/employees")
396     public ResponseEntity<List<Employee>> getAllEmployees(){
397         List<Employee> employees=employeeService.getAllEmployees();
398         if(employees.size()==0) {
399             throw new ResourceNotFoundException("Resource Not Found");
400         }
401         return new ResponseEntity<List<Employee>>(employees, HttpStatus.OK);
402     }
403     @GetMapping(path="/employees/{empId}")
404     public ResponseEntity<Employee> getEmployee(@PathVariable("empId") int empId){
405         Employee employee=employeeService.getEmployee(empId);
406         if(employee==null) {
407             throw new ResourceNotFoundException("Resource Not Found");
408         }
409         return new ResponseEntity<Employee>(employee, HttpStatus.OK);
410     }
411     @PostMapping(path="/employees")
412     public ResponseEntity<String> addEmployee(@RequestBody Employee employee){
413         String retVal="Failed";
414         Employee emp=employeeService.getEmployee(employee.getId());
415         if(emp!=null) {
416             throw new ResourceAlreadyExistsException("Employee Already Exists");
417         }
418         boolean status=employeeService.addEmployee(employee);
419         if(!status) {
420             throw new CannotAddException("Unable To Add Employee");
421         }
422         retVal="Success";
423         return new ResponseEntity<String>(retVal, HttpStatus.OK);
424     }
425     @PutMapping(path="/employees/{empId}")
426     public ResponseEntity<String> update(@RequestBody Employee employee,
427         @PathVariable("empId") int empId){
428         String retVal="Failed";
429         Employee emp=employeeService.getEmployee(employee.getId());
430         if(emp==null) {
431             throw new ResourceNotFoundException("Resource Not Found");

```

```

432     }
433     boolean status=employeeService.updateEmployee(empId, employee);
434     if(!status) {
435         throw new CannotUpdateException("Unable To Update");
436     }
437     retVal="Success";
438     return new ResponseEntity<String>(retVal,HttpStatus.OK);
439 }
440 @DeleteMapping("/employees/{empId}")
441 public ResponseEntity<String> delete(@PathVariable("empId") int empId){
442     String retVal="Failed";
443     Employee emp=employeeService.getEmployee(empId);
444     if(emp==null) {
445         throw new ResourceNotFoundException("Resource Not Found");
446     }
447     boolean status=employeeService.deleteEmployee(empId);
448     if(status) {
449         retVal="Success";
450     }
451     return new ResponseEntity<String>(retVal,HttpStatus.OK);
452 }
453 }
454 =====
455
456 http://localhost:9001/employees
457 {
458     "id": 102,
459     "firstName": "Ameya",
460     "lastName": "Joshi",
461     "salary": 45000
462 }
463

```