

```

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>io.springfox</groupId>
25 <artifactId>springfox-swagger-ui</artifactId>
26 <version>2.6.1</version>
27 <scope>compile</scope>
28 </dependency>
29 <dependency>
30 <groupId>io.springfox</groupId>
31 <artifactId>springfox-swagger2</artifactId>
32 <version>2.6.1</version>
33 <scope>compile</scope>
34 </dependency>
35 <dependency>
36 <groupId>org.springframework.boot</groupId>
37 <artifactId>spring-boot-starter-actuator</artifactId>
38 </dependency>
39 <dependency>
40 <groupId>org.springframework.boot</groupId>
41 <artifactId>spring-boot-starter-data-jpa</artifactId>
42 </dependency>
43 <dependency>
44 <groupId>org.springframework.boot</groupId>
45 <artifactId>spring-boot-starter-web</artifactId>
46 </dependency>
47 <dependency>
48 <groupId>mysql</groupId>
49 <artifactId>mysql-connector-java</artifactId>
50 <scope>runtime</scope>
51 </dependency>
52 <dependency>
53 <groupId>org.springframework.boot</groupId>
54 <artifactId>spring-boot-starter-test</artifactId>
55 <scope>test</scope>
56 <exclusions>
57 <exclusion>
58 <groupId>org.junit.vintage</groupId>
59 <artifactId>junit-vintage-engine</artifactId>
60 </exclusion>
61 </exclusions>
62 </dependency>
63 </dependencies>
64 <build>
65 <plugins>
66 <plugin>
67 <groupId>org.springframework.boot</groupId>
68 <artifactId>spring-boot-maven-plugin</artifactId>
69 </plugin>
70 </plugins>
71 </build>

```

```

72
73 </project>
74
75 =====application.properties=====
76
77 server.port=9001
78 logging.level.web=trace
79
80 spring.datasource.url=jdbc:mysql://localhost:3306/nineleapsdb
81 spring.datasource.username=root
82 spring.datasource.password=root
83 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
84 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
85 spring.jpa.show-sql=true
86 spring.jpa.hibernate.ddl-auto=update
87
88 management.endpoints.enabled-by-default=true
89 management.endpoints.web.exposure.include=*
90 management.endpoint.shutdown.enabled=true
91 management.endpoint.health.show-details=always
92
93 =====Employee.java=====
94 package com.ameya.models;
95
96 import javax.persistence.Entity;
97 import javax.persistence.Id;
98 import javax.persistence.Table;
99
100 import io.swagger.annotations.ApiModel;
101 import io.swagger.annotations.ApiModelProperty;
102
103 @Entity
104 @Table(name="ajemployee")
105 @ApiModel
106 public class Employee {
107
108     @Id
109     @ApiModelProperty(value="The Primary Key For Employee <b>empId</b>",required =
        true)
110     private int id;
111     @ApiModelProperty(value = "The Employee <b>firstName</b>",required = true)
112     private String firstName;
113     @ApiModelProperty(value = "The Employee <b>lastName</b>",required = true)
114     private String lastName;
115     @ApiModelProperty(value = "The Employee <b>salary</b>",required = true)
116     private double salary;
117     public Employee() {
118
119     }
120     public Employee(int id, String firstName, String lastName, double salary) {
121         super();
122         this.id = id;
123         this.firstName = firstName;
124         this.lastName = lastName;
125         this.salary = salary;
126     }
127     public int getId() {
128         return id;
129     }
130     public void setId(int id) {
131         this.id = id;
132     }
133     public String getFirstName() {
134         return firstName;
135     }
136     public void setFirstName(String firstName) {
137         this.firstName = firstName;
138     }
139     public String getLastName() {
140         return lastName;
141     }
142     public void setLastName(String lastName) {
143         this.lastName = lastName;

```

```

144     }
145     public double getSalary() {
146         return salary;
147     }
148     public void setSalary(double salary) {
149         this.salary = salary;
150     }
151     @Override
152     public String toString() {
153         return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" +
154             lastName + ", salary=" + salary
155             + "]";
156     }
157 }
158
159 =====EmployeeDAO.java=====
160 package com.ameya.daos;
161
162 import org.springframework.data.repository.CrudRepository;
163
164 import com.ameya.models.Employee;
165
166 public interface EmployeeDAO extends CrudRepository<Employee, Integer> {
167     public Employee getByFirstNameIgnoreCase(String firstName);
168 }
169 /*
170 List<Person> findByEmailAddressAndLastname(EmailAddress emailAddress, String
171 lastname);
172 // Enables the distinct flag for the query
173 List<Person> findDistinctPeopleByLastnameOrFirstname(String lastname, String
174 firstname);
175 List<Person> findPeopleDistinctByLastnameOrFirstname(String lastname, String
176 firstname);
177 // Enabling ignoring case for all suitable properties
178 List<Person> findByLastnameAndFirstnameAllIgnoreCase(String lastname, String
179 firstname);
180 // Enabling static ORDER BY for a query
181 List<Person> findByLastnameOrderByFirstnameAsc(String lastname);
182 List<Person> findByLastnameOrderByFirstnameDesc(String lastname);
183 @Query("SELECT t.title FROM Todo t where t.id = :id")
184 String findTitleById(@Param("id") Long id);
185 @Query("SELECT t.title FROM Todo t where t.id = :id")
186 Optional<String> findTitleById(@Param("id") Long id);
187 */
188 =====EmployeeService.java=====
189 package com.ameya.services;
190
191 import java.util.List;
192
193 import com.ameya.models.Employee;
194
195 public interface EmployeeService {
196     List<Employee> getAllEmployees();
197     boolean addEmployee(Employee employee);
198     boolean updateEmployee(Integer empId, Employee employee);
199     Employee getEmployee(Integer empId);
200     boolean deleteEmployee(Integer empId);
201     public Employee getByFirstNameIgnoreCase(String firstName);
202 }
203
204 =====EmployeeServiceImpl.java=====
205 package com.ameya.services.impl;
206
207 import java.util.ArrayList;
208 import java.util.List;
209 import java.util.Optional;
210
211 import org.springframework.beans.factory.annotation.Autowired;
212 import org.springframework.stereotype.Service;
213
214 import com.ameya.daos.EmployeeDAO;
215 import com.ameya.models.Employee;

```

```

212 import com.ameya.services.EmployeeService;
213
214 @Service
215 public class EmployeeServiceImpl implements EmployeeService {
216
217     @Autowired
218     EmployeeDAO employeeDao;
219
220     @Override
221     public List<Employee> getAllEmployees() {
222         List<Employee> employees=new ArrayList<>();
223         employeeDao.findAll().forEach(employees::add);
224         return employees;
225     }
226
227     @Override
228     public boolean addEmployee(Employee employee) {
229         return (employeeDao.save(employee)!=null?true:false);
230     }
231
232     @Override
233     public boolean updateEmployee(Integer empId, Employee employee) {
234         Optional<Employee> container=employeeDao.findById(empId);
235         if(!container.isPresent()) {
236             return false;
237         }
238         Employee empToUpdate=container.get();
239         empToUpdate.setFirstName(employee.getFirstName());
240         empToUpdate.setLastName(employee.getLastName());
241         empToUpdate.setSalary(employee.getSalary());
242
243         return (employeeDao.save(empToUpdate)!=null?true:false);
244     }
245
246     @Override
247     public Employee getEmployee(Integer empId) {
248         Optional<Employee> container=employeeDao.findById(empId);
249         Employee emp=null;
250         if(container.isPresent()) {
251             emp=container.get();
252         }
253         return emp;
254     }
255
256     @Override
257     public boolean deleteEmployee(Integer empId) {
258         Optional<Employee> container=employeeDao.findById(empId);
259         if(!container.isPresent()) {
260             return false;
261         }
262         employeeDao.deleteById(empId);
263         return true;
264     }
265
266     @Override
267     public Employee getByFirstNameIgnoreCase(String firstName) {
268         return employeeDao.getByFirstNameIgnoreCase(firstName);
269     }
270
271 }
272
273 =====ResourceNotFoundException.java=====
274
275 package com.ameya.exceptions;
276
277 public class ResourceNotFoundException extends RuntimeException {
278
279     private static final long serialVersionUID = 1L;
280
281     public ResourceNotFoundException(String message) {
282         super(message);
283     }

```

```

284     }
285     =====ResourceAlreadyExistsException.java=====
286     package com.ameya.exceptions;
287
288     public class ResourceAlreadyExistsException extends RuntimeException {
289
290         private static final long serialVersionUID = 1L;
291
292         public ResourceAlreadyExistsException(String message) {
293             super(message);
294         }
295     }
296
297     }
298     =====CannotAddException.java=====
299     package com.ameya.exceptions;
300
301     public class CannotAddException extends RuntimeException {
302
303         private static final long serialVersionUID = 1L;
304
305         public CannotAddException(String message) {
306             super(message);
307             // TODO Auto-generated constructor stub
308         }
309     }
310
311     }
312     =====CannotUpdateException.java=====
313     package com.ameya.exceptions;
314
315     public class CannotUpdateException extends RuntimeException {
316
317         private static final long serialVersionUID = 1L;
318
319         public CannotUpdateException(String message) {
320             super(message);
321             // TODO Auto-generated constructor stub
322         }
323     }
324     =====ExceptionDetails.java=====
325     package com.ameya.exceptions;
326
327     import java.util.Date;
328
329     import org.springframework.http.HttpStatus;
330
331     public class ExceptionDetails {
332         private HttpStatus status;
333         private Date timeStamp;
334         private String message;
335         public ExceptionDetails(HttpStatus status, Date timeStamp, String message) {
336             super();
337             this.status = status;
338             this.timeStamp = timeStamp;
339             this.message = message;
340         }
341         public HttpStatus getStatus() {
342             return status;
343         }
344         public void setStatus(HttpStatus status) {
345             this.status = status;
346         }
347         public Date getTimeStamp() {
348             return timeStamp;
349         }
350         public void setTimeStamp(Date timeStamp) {
351             this.timeStamp = timeStamp;
352         }
353         public String getMessage() {
354             return message;
355         }

```

```

356     public void setMessage(String message) {
357         this.message = message;
358     }
359
360
361 }
362 =====EmployeeExceptionsAdvice.java=====
363 package com.ameya.controllers.advices;
364
365 import java.util.Date;
366
367 import org.springframework.http.HttpStatus;
368 import org.springframework.http.ResponseEntity;
369 import org.springframework.web.bind.annotation.ControllerAdvice;
370 import org.springframework.web.bind.annotation.ExceptionHandler;
371
372 import com.ameya.exceptions.CannotAddException;
373 import com.ameya.exceptions.CannotUpdateException;
374 import com.ameya.exceptions.ExceptionDetails;
375 import com.ameya.exceptions.ResourceAlreadyExistsException;
376 import com.ameya.exceptions.ResourceNotFoundException;
377
378 @ControllerAdvice
379 public class EmployeeExceptionsAdvice {
380
381     @ExceptionHandler(value= {ResourceNotFoundException.class})
382     public ResponseEntity<Object> handleResourceNotFoundException
383     (ResourceNotFoundException ex){
384         ExceptionDetails errorDetails=new ExceptionDetails(
385             HttpStatus.NOT_FOUND,
386             new Date(),
387             ex.getMessage()
388         );
389         return new ResponseEntity<Object>(errorDetails,HttpStatus.NOT_FOUND);
390     }
391     @ExceptionHandler(value= {ResourceAlreadyExistsException.class})
392     public ResponseEntity<Object> handleResourceAlreadyExistsException
393     (ResourceAlreadyExistsException ex){
394         ExceptionDetails errorDetails=new ExceptionDetails(
395             HttpStatus.METHOD_NOT_ALLOWED,
396             new Date(),
397             ex.getMessage()
398         );
399         return new ResponseEntity<Object>(errorDetails,HttpStatus.METHOD_NOT_ALLOWED);
400     }
401     @ExceptionHandler(value= {CannotAddException.class})
402     public ResponseEntity<Object> handleCannotAddException
403     (CannotAddException ex){
404         ExceptionDetails errorDetails=new ExceptionDetails(
405             HttpStatus.INTERNAL_SERVER_ERROR,
406             new Date(),
407             ex.getMessage()
408         );
409         return new
410             ResponseEntity<Object>(errorDetails,HttpStatus.INTERNAL_SERVER_ERROR);
411     }
412     @ExceptionHandler(value= {CannotUpdateException.class})
413     public ResponseEntity<Object> handleCannotUpdateException
414     (CannotUpdateException ex){
415         ExceptionDetails errorDetails=new ExceptionDetails(
416             HttpStatus.INTERNAL_SERVER_ERROR,
417             new Date(),
418             ex.getMessage()
419         );
420         return new
421             ResponseEntity<Object>(errorDetails,HttpStatus.INTERNAL_SERVER_ERROR);
422     }
423 }
424 =====EmployeeController.java=====
425 package com.ameya.controllers;
426
427 import java.util.List;

```

```

427 import org.springframework.beans.factory.annotation.Autowired;
428 import org.springframework.http.HttpStatus;
429 import org.springframework.http.ResponseEntity;
430 import org.springframework.web.bind.annotation.DeleteMapping;
431 import org.springframework.web.bind.annotation.GetMapping;
432 import org.springframework.web.bind.annotation.PathVariable;
433 import org.springframework.web.bind.annotation.PostMapping;
434 import org.springframework.web.bind.annotation.PutMapping;
435 import org.springframework.web.bind.annotation.RequestBody;
436 import org.springframework.web.bind.annotation.RestController;
437
438 import com.ameya.exceptions.CannotAddException;
439 import com.ameya.exceptions.CannotUpdateException;
440 import com.ameya.exceptions.ResourceAlreadyExistsException;
441 import com.ameya.exceptions.ResourceNotFoundException;
442 import com.ameya.models.Employee;
443 import com.ameya.services.EmployeeService;
444
445 import io.swagger.annotations.Api;
446 import io.swagger.annotations.ApiOperation;
447 import io.swagger.annotations.ApiResponse;
448 import io.swagger.annotations.ApiResponses;
449
450 @RestController
451 @Api(value="EmployeeCRUDService",description = "Operations pertaining to Employee
CRUD Operations")
452 public class EmployeeController {
453
454     @Autowired
455     private EmployeeService employeeService;
456
457     @ApiOperation(value="View List Of Employees", response = Iterable.class)
458     @ApiResponses(
459         value= {
460             @ApiResponse(code=200,message="Successful Execution"),
461             @ApiResponse(code=404,message="Resource Not Found"),
462             @ApiResponse(code=401,message="Unauthorized")
463         }
464     )
465     @GetMapping(path="/employees")
466     public ResponseEntity<List<Employee>> getAllEmployees(){
467         List<Employee> employees=employeeService.getAllEmployees();
468         if(employees.size()==0) {
469             throw new ResourceNotFoundException("Resource Not Found");
470         }
471         return new ResponseEntity<List<Employee>>(employees,HttpStatus.OK);
472     }
473     @ApiResponses(
474         value= {
475             @ApiResponse(code=200,message="Successful Execution"),
476             @ApiResponse(code=404,message="Resource Not Found"),
477         }
478     )
479     @GetMapping(path="/employees/{empId}")
480     public ResponseEntity<Employee> getEmployee(@PathVariable("empId") int empId){
481         Employee employee=employeeService.getEmployee(empId);
482         if(employee==null) {
483             throw new ResourceNotFoundException("Resource Not Found");
484         }
485         return new ResponseEntity<Employee>(employee,HttpStatus.OK);
486     }
487     @PostMapping(path="/employees")
488     @ApiResponses(
489         value= {
490             @ApiResponse(code=200,message="Successful Execution"),
491             @ApiResponse(code=405,message="Already Exists"),
492         }
493     )
494     public ResponseEntity<String> addEmployee(@RequestBody Employee employee){
495         String retVal="Failed";
496         Employee emp=employeeService.getEmployee(employee.getId());
497         if(emp!=null) {
498             throw new ResourceAlreadyExistsException("Employee Already Exists");

```

```

499     }
500     boolean status=employeeService.addEmployee(employee);
501     if(!status) {
502         throw new CannotAddException("Unable To Add Employee");
503     }
504     retVal="Success";
505     return new ResponseEntity<String>(retVal,HttpStatus.OK);
506 }
507 @PutMapping(path="/employees/{empId}")
508 @ApiResponses(
509     value= {
510         @ApiResponse(code=200,message="Successful Execution"),
511         @ApiResponse(code=404,message="Resource Not Found"),
512     }
513 )
514 public ResponseEntity<String> update(@RequestBody Employee employee,
515     @PathVariable("empId") int empId){
516     String retVal="Failed";
517     Employee emp=employeeService.getEmployee(employee.getId());
518     if(emp==null) {
519         throw new ResourceNotFoundException("Resource Not Found");
520     }
521     boolean status=employeeService.updateEmployee(empId, employee);
522     if(!status) {
523         throw new CannotUpdateException("Unable To Update");
524     }
525     retVal="Success";
526     return new ResponseEntity<String>(retVal,HttpStatus.OK);
527 }
528 @DeleteMapping("/employees/{empId}")
529 @ApiResponses(
530     value= {
531         @ApiResponse(code=200,message="Successful Execution"),
532         @ApiResponse(code=404,message="Resource Not Found"),
533     }
534 )
535 public ResponseEntity<String> delete(@PathVariable("empId") int empId){
536     String retVal="Failed";
537     Employee emp=employeeService.getEmployee(empId);
538     if(emp==null) {
539         throw new ResourceNotFoundException("Resource Not Found");
540     }
541     boolean status=employeeService.deleteEmployee(empId);
542     if(status) {
543         retVal="Success";
544     }
545     return new ResponseEntity<String>(retVal,HttpStatus.OK);
546 }
547 @GetMapping("/employees/byname/{firstName}")
548 public Employee getByFirstName(@PathVariable("firstName") String firstName) {
549     return employeeService.getByFirstNameIgnoreCase(firstName);
550 }
551 }
552 =====SwaggerConfig.java=====
553 package com.ameya.configs;
554
555 import org.springframework.context.annotation.Bean;
556 import org.springframework.context.annotation.Configuration;
557
558 import com.google.common.base.Predicate;
559 import com.google.common.base.Predicates;
560
561 import springfox.documentation.builders.ApiInfoBuilder;
562 import springfox.documentation.builders.PathSelectors;
563 import springfox.documentation.builders.RequestHandlerSelectors;
564 import springfox.documentation.service.ApiInfo;
565 import springfox.documentation.service.Contact;
566 import springfox.documentation.spi.DocumentationType;
567 import springfox.documentation.spring.web.plugins.Docket;
568 import springfox.documentation.swagger2.annotations.EnableSwagger2;
569
570 @Configuration
571 @EnableSwagger2

```



```

572 public class SwaggerConfig {
573
574     private ApiInfo metaData() {
575         ApiInfo apiInfo=new ApiInfoBuilder()
576             .title("Spring Boot REST Api - EmpCrudService")
577             .description("Documentation For Employee Crud Service")
578             .license("LGPL Licensing")
579             .version("1.0")
580             .contact(new Contact("Ameya Joshi", "www.ameya.com",
581                                 "ameya@abc.com"))
582             .build();
583     }
584     private Predicate<String> paths(){
585         return Predicates.and(PathSelectors.regex("/employees.*"),
586             Predicates.not(PathSelectors.regex("/error.*"))
587         );
588     }
589     @Bean
590     public Docket employeeCrudApi() {
591         return new Docket(DocumentationType.SWAGGER_2)
592             .select()
593             .apis(RequestHandlerSelectors.basePackage("com.ameya.controllers"))
594             .paths(paths())
595             .build().apiInfo(metaData());
596     }
597 }
598
599 =====
600
601 http://localhost:9001/employees
602 {
603     "id": 102,
604     "firstName": "Ameya",
605     "lastName": "Joshi",
606     "salary": 45000
607 }
608
609 http://localhost:9001/employees/byname/AVANI
610
611 Actuator :
612 http://localhost:9001/actuator/beans
613 http://localhost:9001/actuator/env
614 http://localhost:9001/actuator/health
615
616 POST : http://localhost:9001/actuator/shutdown
617
618 Swagger :
619 http://localhost:9001/v2/api-docs
620

```