```
======================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>2.3.0.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.ameya</groupId>
    <artifactId>005-EmpCrudService</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>005-EmpCrudService</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.6.1</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.6.1</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <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>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

```
</project>

====================================application.properties=======================

server.port=9001
logging.level.web=trace

spring.datasource.url=jdbc:mysql://localhost:3306/nineleapsdb
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update

management.endpoints.enabled-by-default=true
management.endpoints.web.exposure.include=*
management.endpoint.shutdown.enabled=true
management.endpoint.health.show-details=always

====================================Employee.java=====================================
package com.ameya.models;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

@Entity
@Table(name="ajemployee")
@ApiModel
public class Employee {

    @Id
    @ApiModelProperty(value="The Primary Key For Employee <b>empId</b>",required =
    true)
    private int id;
    @ApiModelProperty(value = "The Employee <b>firstName</b>",required = true)
    private String firstName;
    @ApiModelProperty(value = "The Employee <b>lastName</b>",required = true)
    private String lastName;
    @ApiModelProperty(value = "The Employee <b>salary</b>",required = true)
    private double salary;
    public Employee() {

    }
    public Employee(int id, String firstName, String lastName, double salary) {
        super();
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.salary = salary;
    }
    public int getId() {
        return id;
    }
    public void setId(int 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;
```

```
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=" +
                lastName + ", salary=" + salary
154                     + "]";
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
        lastname);
171     // Enables the distinct flag for the query
172         List<Person> findDistinctPeopleByLastnameOrFirstname(String lastname, String
            firstname);
173         List<Person> findPeopleDistinctByLastnameOrFirstname(String lastname, String
            firstname);
174     // Enabling ignoring case for all suitable properties
175         List<Person> findByLastnameAndFirstnameAllIgnoreCase(String lastname, String
            firstname);
176     // Enabling static ORDER BY for a query
177         List<Person> findByLastnameOrderByFirstnameAsc(String lastname);
178         List<Person> findByLastnameOrderByFirstnameDesc(String lastname);
179         @Query("SELECT t.title FROM Todo t where t.id = :id")
180         String findTitleById(@Param("id") Long id);
181         @Query("SELECT t.title FROM Todo t where t.id = :id")
182         Optional<String> findTitleById(@Param("id") Long id);
183     */
184     ================================EmployeeService.java============================
185     package com.ameya.services;
186 
187     import java.util.List;
188 
189     import com.ameya.models.Employee;
190 
191     public interface EmployeeService {
192         List<Employee>getAllEmployees();
193         boolean addEmployee(Employee employee);
194         boolean updateEmployee(Integer empId,Employee employee);
195         Employee getEmployee(Integer empId);
196         boolean deleteEmployee(Integer empId);
197         public Employee getByFirstNameIgnoreCase(String firstName);
198     }
199 
200     ==============================EmployeeServiceImpl.java======================
201     package com.ameya.services.impl;
202 
203     import java.util.ArrayList;
204     import java.util.List;
205     import java.util.Optional;
206 
207     import org.springframework.beans.factory.annotation.Autowired;
208     import org.springframework.stereotype.Service;
209 
210     import com.ameya.daos.EmployeeDAO;
211     import com.ameya.models.Employee;
```

```
import com.ameya.services.EmployeeService;

@Service
public class EmployeeServiceImpl implements EmployeeService {

    @Autowired
    EmployeeDAO employeeDao;

    @Override
    public List<Employee> getAllEmployees() {
        List<Employee> employees=new ArrayList<>();
        employeeDao.findAll().forEach(employees::add);
        return employees;
    }

    @Override
    public boolean addEmployee(Employee employee) {
        return (employeeDao.save(employee)!=null?true:false);
    }

    @Override
    public boolean updateEmployee(Integer empId, Employee employee) {
        Optional<Employee> container=employeeDao.findById(empId);
        if(!container.isPresent()) {
            return false;
        }
        Employee empToUpdate=container.get();
        empToUpdate.setFirstName(employee.getFirstName());
        empToUpdate.setLastName(employee.getLastName());
        empToUpdate.setSalary(employee.getSalary());

        return(employeeDao.save(empToUpdate)!=null?true:false);
    }

    @Override
    public Employee getEmployee(Integer empId) {
        Optional<Employee> container=employeeDao.findById(empId);
        Employee emp=null;
        if(container.isPresent()) {
            emp=container.get();
        }
        return emp;
    }

    @Override
    public boolean deleteEmployee(Integer empId) {
        Optional<Employee> container=employeeDao.findById(empId);
        if(!container.isPresent()) {
            return false;
        }
        employeeDao.deleteById(empId);
        return true;
    }

    @Override
    public Employee getByFirstNameIgnoreCase(String firstName) {
        return employeeDao.getByFirstNameIgnoreCase(firstName);
    }

}

==========ResourceNotFoundException.java==========
package com.ameya.exceptions;

public class ResourceNotFoundException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public ResourceNotFoundException(String message) {
        super(message);
    }

```

```
}
==========ResourceAlreadyExistsException.java==========
package com.ameya.exceptions;

public class ResourceAlreadyExistsException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public ResourceAlreadyExistsException(String message) {
        super(message);

    }

}
==========CannotAddException.java==========
package com.ameya.exceptions;

public class CannotAddException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public CannotAddException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

}
==========CannotUpdateException.java==========
package com.ameya.exceptions;

public class CannotUpdateException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public CannotUpdateException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

}
==========ExceptionDetails.java==========
package com.ameya.exceptions;

import java.util.Date;

import org.springframework.http.HttpStatus;

public class ExceptionDetails {
    private HttpStatus status;
    private Date timeStamp;
    private String message;
    public ExceptionDetails(HttpStatus status, Date timeStamp, String message) {
        super();
        this.status = status;
        this.timeStamp = timeStamp;
        this.message = message;
    }
    public HttpStatus getStatus() {
        return status;
    }
    public void setStatus(HttpStatus status) {
        this.status = status;
    }
    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;
    }


}
==============================EmployeeExceptionsAdvice.java====================
package com.ameya.controllers.advices;

import java.util.Date;

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 com.ameya.exceptions.CannotAddException;
import com.ameya.exceptions.CannotUpdateException;
import com.ameya.exceptions.ExceptionDetails;
import com.ameya.exceptions.ResourceAlreadyExistsException;
import com.ameya.exceptions.ResourceNotFoundException;

@ControllerAdvice
public class EmployeeExceptionsAdvice {

    @ExceptionHandler(value= {ResourceNotFoundException.class})
    public ResponseEntity<Object> handleResourceNotFoundException
    (ResourceNotFoundException ex){
        ExceptionDetails errorDetails=new ExceptionDetails(
                HttpStatus.NOT_FOUND,
                new Date(),
                ex.getMessage()
                );
        return new ResponseEntity<Object>(errorDetails,HttpStatus.NOT_FOUND);
    }
    @ExceptionHandler(value= {ResourceAlreadyExistsException.class})
    public ResponseEntity<Object> handleResourceAlreadyExistsException
    (ResourceAlreadyExistsException ex){
        ExceptionDetails errorDetails=new ExceptionDetails(
                HttpStatus.METHOD_NOT_ALLOWED,
                new Date(),
                ex.getMessage()
                );
        return new ResponseEntity<Object>(errorDetails,HttpStatus.METHOD_NOT_ALLOWED);
    }
    @ExceptionHandler(value= {CannotAddException.class})
    public ResponseEntity<Object> handleCannotAddException
    (CannotAddException ex){
        ExceptionDetails errorDetails=new ExceptionDetails(
                HttpStatus.INTERNAL_SERVER_ERROR,
                new Date(),
                ex.getMessage()
                );
        return new
        ResponseEntity<Object>(errorDetails,HttpStatus.INTERNAL_SERVER_ERROR);
    }
    @ExceptionHandler(value= {CannotUpdateException.class})
    public ResponseEntity<Object> handleCannotUpdateException
    (CannotUpdateException ex){
        ExceptionDetails errorDetails=new ExceptionDetails(
                HttpStatus.INTERNAL_SERVER_ERROR,
                new Date(),
                ex.getMessage()
                );
        return new
        ResponseEntity<Object>(errorDetails,HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
================================EmployeeController.java=========================
package com.ameya.controllers;

import java.util.List;

```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.ameya.exceptions.CannotAddException;
import com.ameya.exceptions.CannotUpdateException;
import com.ameya.exceptions.ResourceAlreadyExistsException;
import com.ameya.exceptions.ResourceNotFoundException;
import com.ameya.models.Employee;
import com.ameya.services.EmployeeService;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;

@RestController
@Api(value="EmployeeCRUDService",description = "Operations pertaining to Employee
CRUD Operations")
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;

    @ApiOperation(value="View List Of Employees", response = Iterable.class)
    @ApiResponses(
            value= {
                    @ApiResponse(code=200,message="Successful Execution"),
                    @ApiResponse(code=404,message="Resource Not Found"),
                    @ApiResponse(code=401,message="UnAuthorized")
            }
            )
    @GetMapping(path="/employees")
    public ResponseEntity<List<Employee>> getAllEmployees(){
        List<Employee> employees=employeeService.getAllEmployees();
        if(employees.size()==0) {
            throw new ResourceNotFoundException("Resource Not Found");
        }
        return new ResponseEntity<List<Employee>>(employees,HttpStatus.OK);
    }
    @ApiResponses(
            value= {
                    @ApiResponse(code=200,message="Successful Execution"),
                    @ApiResponse(code=404,message="Resource Not Found"),
            }
            )
    @GetMapping(path="/employees/{empId}")
    public ResponseEntity<Employee> getEmployee(@PathVariable("empId") int empId){
        Employee employee=employeeService.getEmployee(empId);
        if(employee==null) {
            throw new ResourceNotFoundException("Resource Not Found");
        }
        return new ResponseEntity<Employee>(employee,HttpStatus.OK);
    }
    @PostMapping(path="/employees")
    @ApiResponses(
            value= {
                    @ApiResponse(code=200,message="Successful Execution"),
                    @ApiResponse(code=405,message="Already Exists"),
            }
            )
    public ResponseEntity<String> addEmployee(@RequestBody Employee employee){
        String retVal="Failed";
        Employee emp=employeeService.getEmployee(employee.getId());
        if(emp!=null) {
            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
```

```
public class SwaggerConfig {

    private ApiInfo metaData() {
        ApiInfo apiInfo=new ApiInfoBuilder()
                            .title("Spring Boot REST Api - EmpCrudService")
                            .description("Documentation For Employee Crud Service")
                            .license("LGPL Licensing")
                            .version("1.0")
                            .contact(new Contact("Ameya Joshi", "www.ameya.com",
                            "ameya@abc.com"))
                            .build();
        return apiInfo;
    }
    private Predicate<String> paths(){
        return Predicates.and(PathSelectors.regex("/employees.*"),
                Predicates.not(PathSelectors.regex("/error.*"))
                );
    }
    @Bean
    public Docket employeeCrudApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.ameya.controllers"))
                .paths(paths())
                .build().apiInfo(metaData());
    }
}
```

==========

http://localhost:9001/employees

```
{
    "id": 102,
    "firstName": "Ameya",
    "lastName": "Joshi",
    "salary": 45000
}
```

http://localhost:9001/employees/byname/AVANI

Actuator :
http://localhost:9001/actuator/beans
http://localhost:9001/actuator/env
http://localhost:9001/actuator/health

POST : http://localhost:9001/actuator/shutdown

Swagger :
http://localhost:9001/v2/api-docs