```
==============================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-swagger2</artifactId>
            <version>2.9.2</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</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/sapientdb
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.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.health.show-details=always
management.endpoint.shutdown.enabled=true

=======================================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="aj_emp")
@ApiModel
public class Employee {

    @Id
    @ApiModelProperty(value="The Primary Key For Employee <b>id</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) {
        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;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    @Override
    public String toString() {
        return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" +
        lastName + ", salary=" + salary
                + "]";
    }

}

=======================================EmployeeDAO.java=================
==================
package com.ameya.daos;

import org.springframework.data.repository.CrudRepository;

import com.ameya.models.Employee;

public interface EmployeeDAO extends CrudRepository<Employee, Integer> {

    public Employee getByFirstNameIgnoreCase(String firstName);
}
/*
 // Enables the distinct flag for the query
    List<Person> findDistinctPeopleByLastnameOrFirstname(String lastname, String
    firstname);
    List<Person> findPeopleDistinctByLastnameOrFirstname(String lastname, String
    firstname);
    // Enabling ignoring case for all suitable properties
    List<Person> findByLastnameAndFirstnameAllIgnoreCase(String lastname, String
    firstname);
    // Enabling static ORDER BY for a query
    List<Person> findByLastnameOrderByFirstnameAsc(String lastname);
    List<Person> findByLastnameOrderByFirstnameDesc(String lastname);
    @Query("SELECT t.title FROM Todo t where t.id = :id")
    String findTitleById(@Param("id") Long id);
    @Query("SELECT t.title FROM Todo t where t.id = :id")
    Optional<String> findTitleById(@Param("id") Long id);
 */
```

```
==========EmployeeService.java==========
package com.ameya.services;

import java.util.List;

import com.ameya.models.Employee;

public interface EmployeeService {
    List<Employee>getAllEmployees();
    boolean addEmployee(Employee employee);
    boolean updateEmployee(Integer empId,Employee employee);
    Employee getEmployee(Integer empId);
    boolean deleteEmployee(Integer empId);
    public Employee getByFirstNameIgnoreCase(String firstName);
}
==========EmployeeServiceImpl.java==========
package com.ameya.services.impl;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.ameya.daos.EmployeeDAO;
import com.ameya.models.Employee;
import com.ameya.services.EmployeeService;

@Service
public class EmployeeServiceImpl implements EmployeeService {

    @Autowired
    private 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) {
        Employee employee=null;
        Optional<Employee> container=employeeDao.findById(empId);
        if(container.isPresent()) {
            employee=container.get();
        }
        return employee;
    }

    @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 {

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

```
    }

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

public class ResourceAlreadyExistsException extends RuntimeException {

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

}
==================================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) {
        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;
    }

}
====================================GlobalExceptionHandler.java==========
=================
package com.ameya.exceptions.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.ExceptionDetails;
import com.ameya.exceptions.ResourceAlreadyExistsException;
import com.ameya.exceptions.ResourceNotFoundException;

@ControllerAdvice
public class GlobalExceptionHandler {

    @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.BAD_REQUEST,
                new Date(),
                ex.getMessage()
                );
        return new ResponseEntity<Object>(errorDetails,HttpStatus.BAD_REQUEST);
    }
}
=======================================EmployeeController.java==========
=======================
package com.ameya.controllers;

import java.util.List;
```

```
368
369 import org.springframework.beans.factory.annotation.Autowired;
370 import org.springframework.http.HttpStatus;
371 import org.springframework.http.ResponseEntity;
372 import org.springframework.web.bind.annotation.DeleteMapping;
373 import org.springframework.web.bind.annotation.GetMapping;
374 import org.springframework.web.bind.annotation.PathVariable;
375 import org.springframework.web.bind.annotation.PostMapping;
376 import org.springframework.web.bind.annotation.PutMapping;
377 import org.springframework.web.bind.annotation.RequestBody;
378 import org.springframework.web.bind.annotation.RestController;
379
380 import com.ameya.exceptions.ResourceAlreadyExistsException;
381 import com.ameya.exceptions.ResourceNotFoundException;
382 import com.ameya.models.Employee;
383 import com.ameya.services.EmployeeService;
384
385 import io.swagger.annotations.Api;
386 import io.swagger.annotations.ApiOperation;
387 import io.swagger.annotations.ApiResponse;
388 import io.swagger.annotations.ApiResponses;
389
390 @RestController
391 @Api(value="EmployeeCrudService",description = "Operations pertaining to Employee
    CRUD Operations")
392 public class EmployeeController {
393
394     @Autowired
395     private EmployeeService employeeService;
396     @ApiOperation(value="View List Of Employees",response=Iterable.class)
397     @ApiResponses(
398                 value= {
399                         @ApiResponse(code=200,message="Successful Execution"),
400                         @ApiResponse(code=404,message="Resource Not Found"),
401                         @ApiResponse(code=401,message="UnAuthorized")
402                 }
403             )
404     @GetMapping("/employees")
405     public ResponseEntity<List<Employee>> getAllEmployees(){
406         List<Employee> employees=employeeService.getAllEmployees();
407         if(employees.size()==0) {
408             throw new ResourceNotFoundException("!! Resource Not Found !!");
409         }
410         return new ResponseEntity<List<Employee>>(employees,HttpStatus.OK);
411     }
412     @ApiResponses(
413             value= {
414                     @ApiResponse(code=200,message="Successful Execution"),
```

```
415                 @ApiResponse(code=404,message="Resource Not Found"),
416         }
417         )
418     @GetMapping(path="/employees/{empId}")
419     public ResponseEntity<Employee> getEmployee(@PathVariable("empId") int empId){
420         Employee employee=employeeService.getEmployee(empId);
421         if(employee==null) {
422             throw new ResourceNotFoundException("Resource Not Found");
423         }
424         return new ResponseEntity<Employee>(employee,HttpStatus.OK);
425     }
426     @PostMapping(path="/employees")
427     @ApiResponses(
428             value= {
429                     @ApiResponse(code=200,message="Successful Execution"),
430                     @ApiResponse(code=405,message="Already Exists"),
431             }
432             )
433     public ResponseEntity<String> addEmployee(@RequestBody Employee employee){
434         String retVal="Failed";
435         Employee emp=employeeService.getEmployee(employee.getId());
436         if(emp!=null) {
437             throw new ResourceAlreadyExistsException("Employee Already Exists");
438         }
439         boolean status=employeeService.addEmployee(employee);
440         if(status) {
441             retVal="Success";
442         }
443         return new ResponseEntity<String>(retVal,HttpStatus.OK);
444     }
445     @PutMapping(path="/employees/{empId}")
446     @ApiResponses(
447             value= {
448                     @ApiResponse(code=200,message="Successful Execution"),
449                     @ApiResponse(code=404,message="Resource Not Found"),
450             }
451             )
452     public ResponseEntity<String> update(@RequestBody Employee employee,
453             @PathVariable("empId") int empId){
454         String retVal="Failed";
455         Employee emp=employeeService.getEmployee(employee.getId());
456         if(emp==null) {
457             throw new ResourceNotFoundException("Resource Not Found");
458         }
459         boolean status=employeeService.updateEmployee(empId, employee);
460         if(status) {
461             retVal="Success";
462         }
```

```
        return new ResponseEntity<String>(retVal,HttpStatus.OK);
    }
    @DeleteMapping("/employees/{empId}")
    @ApiResponses(
            value= {
                    @ApiResponse(code=200,message="Successful Execution"),
                    @ApiResponse(code=404,message="Resource Not Found"),
            }
            )
    public ResponseEntity<String> delete(@PathVariable("empId") int empId){
        String retVal="Failed";
        Employee emp=employeeService.getEmployee(empId);
        if(emp==null) {
            throw new ResourceNotFoundException("Resource Not Found");
        }
        boolean status=employeeService.deleteEmployee(empId);
        if(status) {
            retVal="Success";
        }
        return new ResponseEntity<String>(retVal,HttpStatus.OK);
    }
    @GetMapping("/employees/getbyname/{firstName}")
    public Employee getByFirstNameIgnoreCase(@PathVariable("firstName") String
    firstName) {
        return employeeService.getByFirstNameIgnoreCase(firstName);
    }
}
============================SwaggerConfig.java=========================
package com.ameya.configs;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.google.common.base.Predicate;
import com.google.common.base.Predicates;

import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {
```

```
510
511     private ApiInfo metaData() {
512         ApiInfo apiInfo=new ApiInfoBuilder()
513                                 .title("Spring Boot REST Api - EmpCrudService")
514                                 .description("Documentation For Employee CRUD Service")
515                                 .license("LGPL License")
516                                 .version("1.0")
517                                 .contact(new Contact("Ameya
                                    Joshi","www.ameya.com","ameya@ameya.com"))
518                                 .build();
519         return apiInfo;
520     }
521     private Predicate<String> paths(){
522         return Predicates.and(PathSelectors.regex("/employees.*"),
523                 Predicates.not(PathSelectors.regex("/error.*"))
524                 );
525     }
526     @Bean
527     public Docket employeeCrudApi() {
528         return new Docket(DocumentationType.SWAGGER_2)
529                             .select()
530
                            .apis(RequestHandlerSelectors.basePackage("com.ameya.controll
                            ers"))
531                         .paths(paths())
532                         .build()
533                         .apiInfo(metaData());
534     }
535 }
536 ========================CustomHealthIndicator.java====================
537 package com.ameya.actuators;
538
539 import org.springframework.boot.actuate.health.Health;
540 import org.springframework.boot.actuate.health.HealthIndicator;
541 import org.springframework.stereotype.Component;
542 @Component
543 public class CustomHealthIndicator implements HealthIndicator {
544     private final String message_key = "Service A";
545     @Override
546     public Health health() {
547     if (!isRunningServiceA()) {
548     return Health.down().withDetail(message_key, "Not Available").build();
549     }
550     return Health.up().withDetail(message_key, "Available").build();
551     }
552     private Boolean isRunningServiceA() {
553     Boolean isRunning = true;
554     // Logic Skipped
```

```
    return isRunning;
    }
}

================================================================================
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
http://localhost:9001/swagger-ui.html


```