

Spring boot 2-2019

Spring Boot 2 and Swagger



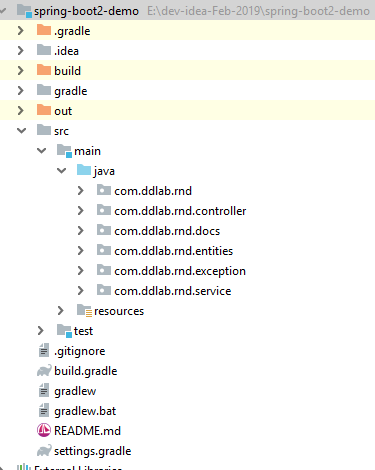
March 17, 2019

DDLAB INC

Debadatta Mishra

Spring Boot 2.1.3 Version with Spring Fox as Swagger

# Project Structure



# Build Structure (build.gradle)

plugins {  
 id **'org.springframework.boot'** version **'2.1.3.RELEASE'** id **'java'**}  
  
apply **plugin**: **'io.spring.dependency-management'**group = **'com.ddlab.rnd'**version = **'0.0.1-SNAPSHOT'**sourceCompatibility = **'1.8'**repositories {  
 mavenCentral()  
}  
  
dependencies {  
 implementation **'org.springframework.boot:spring-boot-starter'** implementation **'org.springframework.boot:spring-boot-starter-web'** implementation **'io.springfox:springfox-swagger2:2.9.2'** implementation **'io.springfox:springfox-swagger-ui:2.9.2'** implementation **'org.projectlombok:lombok:1.18.4'** annotationProcessor **'org.projectlombok:lombok:1.18.4'** implementation **'org.apache.commons:commons-io:1.3.2'** testImplementation **'org.springframework.boot:spring-boot-starter-test'**}

# Spring Boot Application (DDLABSpringBootApplication.java)

**package** com.ddlab.rnd;  
  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
**import** org.springframework.context.annotation.ComponentScan;  
  
@SpringBootApplication  
@ComponentScan(basePackages = {**"com.ddlab.rnd.\*"**})  
**public class** DDLABSpringBootApplication {  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(DDLABSpringBootApplication.**class**, args);  
 }  
}

# Swagger Configuration (SwaggerConfig.java)

**package** com.ddlab.rnd.docs;  
**import** org.springframework.context.annotation.Bean;  
**import** org.springframework.context.annotation.Configuration;  
**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 {  
@Bean  
 **public** Docket api() {  
 **return new** Docket(DocumentationType.***SWAGGER\_2***)  
 .select()  
 .apis(RequestHandlerSelectors.*basePackage*(**"com.ddlab.rnd"**))  
 .paths(PathSelectors.*any*())  
 .build()  
 .apiInfo(apiInfo());  
 }  
  
 */\*\*  
 \* Provides the basic api meta data information.  
 \*  
 \** ***@return*** *the api info  
 \*/* **private** ApiInfo apiInfo() {  
 **return new** ApiInfoBuilder()  
 .title(**"REST API for Spring Boot 2.1.3 Version"**)  
 .description(**"Demo Application for Spring Boot"**)  
 .termsOfServiceUrl(**"http://springfox.io"**)  
 .contact(**new** Contact(**"DDLab INC"**, **"http://www.github.com/debjava"**, **"deba.java@gmail.com"**))  
 .license(**"Spring Boot 2.1.3 Version"**)  
 .licenseUrl(**"http://www.github.com/debjava"**)  
 .version(**"0.0.1"**)  
 .build();  
 }  
}

# Exception Handler with @ControllerAdvice(ResponseExceptionHandler.java)

**package** com.ddlab.rnd.exception;  
  
**import** com.fasterxml.jackson.databind.ObjectMapper;  
**import** org.springframework.http.HttpHeaders;  
**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** org.springframework.web.context.request.WebRequest;  
**import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;  
  
**import** java.io.IOException;  
**import** java.nio.file.AccessDeniedException;  
  
**@ControllerAdvice**  
**public class** ResponseExceptionHandler **extends** ResponseEntityExceptionHandler {  
  
 **@ExceptionHandler({AccessDeniedException.class})** **public** ResponseEntity<Object> handleAccessDeniedException(Exception ex, WebRequest request) {  
 **return new** ResponseEntity<Object>(  
 *toJSON*(**"Access denied message here"**), **new** HttpHeaders(), HttpStatus.***FORBIDDEN***);  
 }  
  
 **@ExceptionHandler({NullPointerException.class})** **public** ResponseEntity<Object> handleNullPointerException(Exception ex, WebRequest request) {  
 **return new** ResponseEntity<Object>(  
 *toJSON*(ex.getMessage()), **new** HttpHeaders(), HttpStatus.***INTERNAL\_SERVER\_ERROR***);  
 }  
  
 **@ExceptionHandler({IllegalArgumentException.class})** **public** ResponseEntity<Object> handleIllegalArgumentException(Exception ex, WebRequest request) {  
 **return new** ResponseEntity<Object>(  
 *toJSON*(ex.getMessage()), **new** HttpHeaders(), HttpStatus.***BAD\_REQUEST***);  
 }

**private static** String toJSON(Object obj) {  
 ObjectMapper mapper = **new** ObjectMapper();  
 String toJson = **null**;  
 **try** {  
 toJson = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(obj);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return** toJson;  
 }  
}

# Service Layer(EmpService.java)

**package** com.ddlab.rnd.service;  
  
**import** com.ddlab.rnd.entities.Employee;  
  
**import** java.util.List;  
  
**public interface** EmpService {  
  
 List<Employee> getAllEmployees();  
  
 List<Employee> getAllEmpsException1();  
  
 Employee getEmployeeById(String empId);  
  
 Employee searchEmpByName(String firstName, String lastName);  
  
 **void** createEmployee(Employee emp);  
  
 **void** deleteEmployee(**int** id);  
  
 Employee updateEmployeeInfo(Employee emp);  
  
 Employee updateEmployeeLogin(Employee emp);  
}

# Service Implementation (EmpServiceImpl.java)

**package** com.ddlab.rnd.service;  
  
**import** com.ddlab.rnd.entities.Employee;  
**import** org.springframework.context.annotation.Primary;  
**import** org.springframework.stereotype.Service;  
  
**import** java.util.Arrays;  
**import** java.util.List;  
  
@Service  
@Primary  
**public class** EmpServiceImpl **implements** EmpService {  
  
 @Override  
 **public** List<Employee> getAllEmployees() {  
 **return** Arrays.*asList*(  
 **new** Employee(1, **"John"**, **"Abraham"**),  
 **new** Employee(2, **"Vidya"**, **"Balan"**),  
 **new** Employee(3, **"Debadatta"**, **"Mishra"**));  
 }  
  
 @Override  
 **public** Employee getEmployeeById(String empId) {  
 Employee emp = **null**;  
 **try** {  
 **int** id = Integer.*parseInt*(empId);  
 emp = **new** Employee(id, **"Deb"**, **"Mishra"**);  
 } **catch** (NumberFormatException nfe) {  
 **throw new** IllegalArgumentException(**"Invalid emp Id, unable to create an employee"**);  
 }  
 **return** emp;  
 }  
  
 @Override  
 **public** Employee searchEmpByName(String firstName, String lastName) {  
 **if** (firstName == **null** || lastName == **null**)  
 **throw new** IllegalArgumentException(**"Invalid search criteria"**);  
 **return new** Employee(13, firstName, lastName);  
 }  
  
 @Override  
 **public void** createEmployee(Employee emp) {  
 System.***out***.println(**"emp = "** + emp);  
 System.***out***.println(**"Employee created successfully..."**);  
 }  
  
 @Override  
 **public void** deleteEmployee(**int** id) {}  
  
 @Override  
 **public** Employee updateEmployeeInfo(Employee emp) {  
 **return** emp;  
 }  
  
 @Override  
 **public** List<Employee> getAllEmpsException1() {  
 **throw new** NullPointerException(**"Unable to retrieve the employee details, emp server is down"**);  
 }  
  
 @Override  
 **public** Employee updateEmployeeLogin(Employee emp) {  
 **return** emp;  
 }  
}

# Entity Layer(Employee.java)

**package** com.ddlab.rnd.entities;  
  
**import** io.swagger.annotations.ApiModelProperty;  
**import** lombok.AllArgsConstructor;  
**import** lombok.Data;  
**import** lombok.NoArgsConstructor;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
**public class** Employee {  
 @ApiModelProperty(required = **true**, example = **"123"**)  
 **private int empId**;  
  
 @ApiModelProperty(required = **true**, example = **"Deb"**)  
 **private** String **firstName**;  
  
 @ApiModelProperty(required = **true**, example = **"Mishra"**)  
 **private** String **lastName**;  
}

# GetController.java

**package** com.ddlab.rnd.controller;  
  
**import** com.ddlab.rnd.entities.Employee;  
**import** com.ddlab.rnd.service.EmpService;  
**import** io.swagger.annotations.Api;  
**import** io.swagger.annotations.ApiOperation;  
**import** io.swagger.annotations.ApiResponse;  
**import** io.swagger.annotations.ApiResponses;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.MediaType;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.\*;  
  
**import** java.util.List;  
  
**@Api(  
 value = "All Employees in an Organization",  
 description = "API for all Employees",  
 tags = {"All Employees"})  
@RequestMapping(value = "/get")  
@RestController  
public class** GetController {  
  
 **@Autowired private EmpService empService;**  
  
 @ApiOperation(  
 value = **"Provides list of Employees in an organization"**,  
 tags = {**"All Employees"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = Employee[].**class**, message = **"Retrieved"**),  
 @ApiResponse(code = 500, message = **"Unable to retrieve employees"**)  
 })

**@GetMapping(path = "/allEmps")** **public** ResponseEntity<?> getAllEmployees() {  
 List<Employee> allEmps = **empService**.getAllEmployees();  
 **return** ResponseEntity.*ok*(allEmps);  
 }

**@ApiOperation(  
 value = "Provides list of Employees with NullPointer Exception",  
 tags = {"All Employees"})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = Employee[].class, message = "Received"),  
 @ApiResponse(code = 500, message = "Unable to retrieve employees")  
 })** **@GetMapping(path = "/allEmpException1")** **public** ResponseEntity<?> getAllEmpsWithException1() {  
 List<Employee> allEmps = **empService**.getAllEmpsException1();  
 **return** ResponseEntity.*ok*(allEmps);  
 }  
  
 @ApiOperation(  
 value = **"Provides Employee by Id"**,  
 tags = {**"All Employees"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = Employee.**class**, message = **"Received"**),  
 @ApiResponse(code = 400, response = String.**class**, message = **"Bad request to get emp info"**),  
 @ApiResponse(code = 500, message = **"Unable to retrieve employees"**)  
 })  
 **@GetMapping(  
 path = "/emp/id/{id}",  
 produces = {MediaType.*APPLICATION\_JSON\_VALUE*})** **public** ResponseEntity<?> getEmployeePerId(  
 **@PathVariable(value = "id", required = true) String id**) {  
 Employee empById = **empService**.getEmployeeById(id);  
 **return** ResponseEntity.*ok*(empById);  
 }

@ApiOperation(  
 value = **"Search Employee by name"**,  
 tags = {**"All Employees"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = Employee.**class**, message = **"Received"**),  
 @ApiResponse(code = 400, response = String.**class**, message = **"Bad request to get emp info"**),  
 @ApiResponse(code = 500, message = **"Unable to retrieve employees"**)  
 })  
 **@GetMapping(  
 path = "/emp/name",  
 produces = {MediaType.*APPLICATION\_JSON\_VALUE*})**  
 **public** ResponseEntity<?> getEmployeeByName(  
 @RequestParam(value = **"firstName"**, required = **false**) String firstName,  
 @RequestParam(value = **"lastName"**, required = **false**) String lastName) {  
 Employee empById = **empService**.searchEmpByName(firstName, lastName);  
 **return** ResponseEntity.*ok*(empById);  
 }  
}

# PostController

**package** com.ddlab.rnd.controller;  
  
**import** com.ddlab.rnd.entities.Employee;  
**import** com.ddlab.rnd.service.EmpService;  
**import** io.swagger.annotations.Api;  
**import** io.swagger.annotations.ApiOperation;  
**import** io.swagger.annotations.ApiResponse;  
**import** io.swagger.annotations.ApiResponses;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.HttpHeaders;  
**import** org.springframework.http.HttpStatus;  
**import** org.springframework.http.MediaType;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.\*;  
**import** org.springframework.web.servlet.support.ServletUriComponentsBuilder;  
**import** org.springframework.web.util.UriComponentsBuilder;  
  
**import** java.net.URI;  
**import** java.util.Map;  
  
@Api(  
 value = **"Add Employee in Organization"**,  
 description = **"API for add employee in organisation"**,  
 tags = {**"Employee creation"**})  
@RequestMapping(value = **"/post"**)  
**@RestController**  
**public class** PostController {  
  
 **@Autowired private EmpService empService;**

**@ApiOperation(  
 value = "Create an employee in organization",  
 tags = {"Employee creation"})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 201, response = Void.class, message = "Created"),  
 @ApiResponse(code = 500, message = "Unable to create employees")  
 })  
 @PostMapping(  
 path = "/createEmp",  
 consumes = {MediaType.*APPLICATION\_JSON\_VALUE*})** **public** ResponseEntity<Void> createEmployee(@RequestBody(required = **true**) Employee emp) {  
 **empService**.createEmployee(emp);  
 *// URI location =  
 // ServletUriComponentsBuilder.fromCurrentRequest()  
 // .path("/{id}")  
 // .buildAndExpand(emp.getEmpId())  
 // .toUri();  
 // return ResponseEntity.created(location).build();* UriComponentsBuilder builder = UriComponentsBuilder.*fromPath*(**"/ddlab/get"**);  
 HttpHeaders headers = **new** HttpHeaders();  
 headers.setLocation(builder.path(**"/emp/id/{id}"**).buildAndExpand(emp.getEmpId()).toUri());  
 **return new ResponseEntity<Void>(headers, HttpStatus.*CREATED*);**  
 }

@ApiOperation(  
 value = **"Saves employee information in database"**,  
 tags = {**"Employee creation"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 201, response = Void.**class**, message = **"Created"**),  
 @ApiResponse(code = 500, message = **"Unable to create employees"**)  
 })  
 **@PostMapping(  
 path = "/saveEmp",  
 consumes = {MediaType.*APPLICATION\_FORM\_URLENCODED\_VALUE*})** **public** ResponseEntity<Void> saveEmployee(  
 @RequestParam(value = **"id"**, required = **false**) **int** id,  
 @RequestParam(value = **"firstName"**, required = **true**) String firstName,  
 @RequestParam(value = **"lastName"**, required = **true**) String lastName) {  
 Employee emp = **new** Employee(id, firstName, lastName);  
 *// It works like @FormParam in Jersey* System.***out***.println(**"emp = "** + emp);  
 **empService**.createEmployee(emp);  
 UriComponentsBuilder builder = UriComponentsBuilder.*fromPath*(**"/ddlab/get"**);  
 HttpHeaders headers = **new** HttpHeaders();  
 headers.setLocation(builder.path(**"/emp/id/{id}"**).buildAndExpand(emp.getEmpId()).toUri());  
 **return new** ResponseEntity<Void>(headers, HttpStatus.***CREATED***);  
 }

@ApiOperation(  
 value = **"Saves all employees information in database with an admin key"**,  
 tags = {**"Employee creation"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 201, response = Void.**class**, message = **"Created"**),  
 @ApiResponse(code = 500, message = **"Unable to create employees"**)  
 })  
 **@PostMapping(  
 path = "/createEmpKey",  
 consumes = {MediaType.*APPLICATION\_JSON\_VALUE*})** **public** ResponseEntity<Void> createEmployeeKey(  
 @RequestHeader(**"key"**) String Key, @RequestBody(required = **true**) Employee emp) {  
 System.***out***.println(**"Key = "** + Key);  
 UriComponentsBuilder builder = UriComponentsBuilder.*fromPath*(**"/ddlab/get"**);  
 HttpHeaders headers = **new** HttpHeaders();  
 headers.setLocation(builder.path(**"/emp/id/{id}"**).buildAndExpand(111).toUri());  
 **return new** ResponseEntity<Void>(headers, HttpStatus.***CREATED***);  
 }  
}

# PutController

**package** com.ddlab.rnd.controller;  
  
**import** com.ddlab.rnd.entities.Employee;  
**import** com.ddlab.rnd.service.EmpService;  
**import** io.swagger.annotations.Api;  
**import** io.swagger.annotations.ApiOperation;  
**import** io.swagger.annotations.ApiResponse;  
**import** io.swagger.annotations.ApiResponses;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.MediaType;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.PutMapping;  
**import** org.springframework.web.bind.annotation.RequestBody;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
  
@Api(  
 value = **"Update Employee in Organization"**,  
 description = **"API for employee updation in organisation"**,  
 tags = {**"Employee updation"**})  
@RequestMapping(value = **"/put"**)  
@RestController  
**public class** PutController {  
  
 @Autowired **private** EmpService **empService**;  
  
 @ApiOperation(  
 value = **"Create an employee in organization"**,  
 tags = {**"Employee updation"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = Employee.**class**, message = **"Ok"**),  
 @ApiResponse(code = 500, message = **"Unable to create employees"**)  
 })  
 **@PutMapping(  
 path = "/updateEmp",  
 consumes = {MediaType.*APPLICATION\_JSON\_VALUE*})** **public ResponseEntity<?> updateEmployee(@RequestBody(required = true) Employee emp)** {  
 Employee updatedEmp = **empService**.updateEmployeeInfo(emp);  
 **return** ResponseEntity.*ok*(updatedEmp);  
 }  
}

# PatchController

**package** com.ddlab.rnd.controller;  
**import** com.ddlab.rnd.entities.Employee;  
**import** com.ddlab.rnd.service.EmpService;  
**import** io.swagger.annotations.Api;  
**import** io.swagger.annotations.ApiOperation;  
**import** io.swagger.annotations.ApiResponse;  
**import** io.swagger.annotations.ApiResponses;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.MediaType;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.PatchMapping;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestParam;  
**import** org.springframework.web.bind.annotation.RestController;  
  
@Api(  
 value = **"Update Employee Partially in Organization"**,  
 description = **"API for employee updation partially in organisation"**,  
 tags = {**"Employee Patch updation"**})  
@RequestMapping(value = **"/patch"**)  
@RestController  
**public class** PatchController {  
  
 @Autowired **private** EmpService **empService**;  
  
 @ApiOperation(  
 value = **"Create an employee in organization"**,  
 tags = {**"Employee Patch updation"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = Employee.**class**, message = **"Ok"**),  
 @ApiResponse(code = 500, message = **"Unable to update employees"**)  
 })  
 **@PatchMapping(  
 path = "/updatePartialEmp",  
 produces = {MediaType.*APPLICATION\_JSON\_VALUE*})**  
 **public** ResponseEntity<?> createEmployee(  
 @RequestParam(value = **"id"**, required = **true**) String id,  
 @RequestParam(value = **"loginName"**, required = **true**) String loginName) {  
 *// First find the emp by id* Employee empById = **empService**.getEmployeeById(id);  
 empById.setFirstName(loginName);  
 Employee updatedEmp = **empService**.updateEmployeeLogin(empById);  
 System.***out***.println(**"updatedEmp = "** + updatedEmp);  
 **return** ResponseEntity.*ok*(updatedEmp);  
 }  
}

# DeleteController

**package** com.ddlab.rnd.controller;  
  
**import** com.ddlab.rnd.service.EmpService;  
**import** io.swagger.annotations.Api;  
**import** io.swagger.annotations.ApiOperation;  
**import** io.swagger.annotations.ApiResponse;  
**import** io.swagger.annotations.ApiResponses;  
**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.PathVariable;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
  
@Api(  
 value = **"Delete Employee Info in an Organization"**,  
 description = **"API for all Employees"**,  
 tags = {**"Employee deletion"**})  
@RequestMapping(value = **"/delete"**)  
@RestController  
**public class** DeleteController {  
  
 @Autowired **private** EmpService **empService**;  
  
 @ApiOperation(  
 value = **"Create an employee in organization"**,  
 tags = {**"Employee deletion"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 204, response = Void.**class**, message = **"Deleted"**),  
 @ApiResponse(code = 500, message = **"Unable to create employees"**)  
 })  
 **@DeleteMapping("emp/{id}")**  
 **public** ResponseEntity<Void> deleteArticle**(@PathVariable("id") Integer id**) {  
 **empService**.deleteEmployee(id);  
 **return new** ResponseEntity<Void>(HttpStatus.***NO\_CONTENT***);  
 }  
}

# FileUploadController

**package** com.ddlab.rnd.controller;  
**import** com.ddlab.rnd.entities.Employee;  
**import** com.ddlab.rnd.service.EmpService;  
**import** com.fasterxml.jackson.databind.ObjectMapper;  
**import** io.swagger.annotations.\*;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.\*;  
**import** org.springframework.web.bind.annotation.\*;  
**import** org.springframework.web.multipart.MultipartFile;  
**import** java.io.File;  
**import** java.io.IOException;  
  
@Api(  
 value = **"Upload Employee Resume in Organization"**,  
 description = **"API for employee resume upload in organisation"**,  
 tags = {**"Employee Resume upload"**})  
@RequestMapping(value = **"/upload"**)  
@RestController  
**public class** FileUploadController {  
  
 @Autowired **private** EmpService **empService**;  
  
 @ApiOperation(  
 value = **"Upload Employee Resume"**,  
 tags = {**"Employee Resume upload"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 201, response = String.**class**, message = **"Uploaded"**),  
 @ApiResponse(code = 500, message = **"Unable to upload employee's resume"**)  
 })  
 **@PostMapping(value = "/singleFile", consumes = MediaType.*MULTIPART\_FORM\_DATA\_VALUE*)** **public** ResponseEntity<?> handleFileUpload**(@RequestParam("file") MultipartFile multipartFile**) {  
 **long** fileSize = multipartFile.getSize();  
 System.***out***.println(**"fileSize = "** + fileSize);  
 System.***out***.println(**"File Name = "** + multipartFile.getOriginalFilename());  
 File file = **new** File(**"E:/sure-delete/"** + multipartFile.getOriginalFilename());  
 **try** {  
 **multipartFile.transferTo(file);** } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return** ResponseEntity.*ok*(**"File uploaded with size "** + fileSize + **" successfully"**);  
 }  
  
 @ApiOperation(  
 value = **"Upload Employee and Resume"**,  
 tags = {**"Employee Resume upload"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(  
 code = 201,  
 response = String.**class**,  
 message = **"Emp Created and Resume Uploaded"**),  
 @ApiResponse(code = 500, message = **"Unable to upload employee's resume"**)  
 })  
 @PostMapping(  
 value = **"/empIdAndResume"**,  
 consumes = {MediaType.***MULTIPART\_FORM\_DATA\_VALUE***},  
 produces = MediaType.***TEXT\_PLAIN\_VALUE***)  
 **public** ResponseEntity<?> handleEmployeAndResume(  
 @RequestPart(**"file"**) MultipartFile multipartFile,  
 @RequestPart(value = **"emp"**, required = **true**) String empJsonTxt) {  
 System.***out***.println(**"multipartFile = "** + multipartFile);  
 System.***out***.println(**"EMP as Json String = "** + empJsonTxt);  
 ObjectMapper objectMapper = **new** ObjectMapper();  
 **try** {  
 Employee emp = objectMapper.readValue(empJsonTxt, Employee.**class**);  
 System.***out***.println(**"Now emp = "** + emp);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **long** fileSize = multipartFile.getSize();  
 System.***out***.println(**"fileSize = "** + fileSize);  
 System.***out***.println(**"File Name = "** + multipartFile.getOriginalFilename());  
 File file = **new** File(**"E:/sure-delete/"** + multipartFile.getOriginalFilename());  
 **try** {  
 **multipartFile.transferTo(file);** } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return** ResponseEntity.*ok*(**"File uploaded with size "** + fileSize + **" successfully"**);  
 }

@ApiOperation(  
 value = **"Upload multiple resumes upload"**,  
 tags = {**"Employee Resume upload"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 201, response = String.**class**, message = **"Emp Resume Uploaded"**),  
 @ApiResponse(code = 500, message = **"Unable to upload employee's resume"**)  
 })  
 **@PostMapping(  
 value = "/multiUpload",  
 consumes = {MediaType.*MULTIPART\_FORM\_DATA\_VALUE*},  
 produces = MediaType.*TEXT\_PLAIN\_VALUE*)** **public** ResponseEntity<?> uploadingMultipleFiles(  
 @RequestParam(**"files"**) MultipartFile[] uploadingFiles) {  
 **for (MultipartFile uploadedFile : uploadingFiles) {  
 System.*out*.println("Uploaded File Name = " + uploadedFile.getOriginalFilename());  
 File file = new File("E:/sure-delete/" + uploadedFile.getOriginalFilename());  
 try {  
 uploadedFile.transferTo(file);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }** **return** ResponseEntity.*ok*(**"All Files uploaded successfully ..."**);  
 }  
}

**Note: Multiple files upload does not work with Swagger as per documentation, you can check in internet. You can use PostMan client to test. To test, select browse button and select multiple files at a time.**

**Besides, if you want to upload a photo along with json data like employee information, you have to give two parameters. One as MultiPartFile and another as RequestParam . In request param variable, receive the employee json information as String and then convert into Employee object. This is the only way to achieve it.**

# FileDownloadController

**package** com.ddlab.rnd.controller;  
**import** io.swagger.annotations.\*;  
**import** org.apache.commons.io.IOUtils;  
**import** org.springframework.http.HttpHeaders;  
**import** org.springframework.http.HttpStatus;  
**import** org.springframework.http.MediaType;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.\*;  
**import** java.io.\*;  
**import** java.nio.file.Files;  
**import** java.nio.file.Paths;  
  
@Api(  
 value = **"Download File"**,  
 description = **"API for file download"**,  
 tags = {**"File download"**})  
@RequestMapping(value = **"/download"**)  
@RestController  
**public class** FileDownloadController {  
  
 @ApiOperation(  
 value = **"Download file"**,  
 tags = {**"File download"**})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = String.**class**, message = **"Downloaded"**),  
 @ApiResponse(code = 500, message = **"Unable to download employee's resume"**)  
 })  
 **@PostMapping(value = "/file", produces = MediaType.*APPLICATION\_OCTET\_STREAM\_VALUE*)** **public** ResponseEntity<?> downloadFile(@RequestParam(**"fileName"**) String fileName) {  
 String dirPath = **"E:/sure-delete/"**;  
 **byte**[] fileBytes = **null**;  
 **try** {  
 fileBytes = Files.*readAllBytes*(Paths.*get*(dirPath + fileName));  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return** ResponseEntity.*ok*()  
 .contentType(MediaType.***APPLICATION\_OCTET\_STREAM***)  
 .header(HttpHeaders.***CONTENT\_DISPOSITION***, **"attachment; filename=\""** + fileName + **"\""**)  
 .body(fileBytes);  
 }  
  
 **@ApiOperation(  
 value = "Show file",  
 tags = {"File download"})  
 @ApiResponses(  
 value = {  
 @ApiResponse(code = 200, response = String.class, message = "Show"),  
 @ApiResponse(code = 500, message = "Unable to show employee's resume")  
 })**  
 **@GetMapping(value = "/showImage", produces = "image/jpg")**  
 **public** ResponseEntity<**byte**[]> getImageAsResponseEntity(  
 @RequestParam(**"fileName"**) String fileName) {  
 String dirPath = **"E:/sure-delete/"**;  
 **HttpHeaders headers = new HttpHeaders();  
 InputStream in = null;  
 byte[] media = new byte[0];  
 try {  
 in = new FileInputStream(dirPath + fileName);  
 media = IOUtils.*toByteArray*(in);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 ResponseEntity<byte[]> responseEntity = new ResponseEntity<>(media, headers, HttpStatus.*OK*);  
 return responseEntity;** }  
}

Here, you can directly hit the url in browser and can see the image. It also works in Swagger.

# Application.Properties

**server.servlet.context-path**=**/ddlab**