Springboot

- Extension of Spring framework
- Rapid application development
- Ease managing dependency - Provides starter templates
- Auto Configuration
- Embedded server - JAR file (in place of WAR) - Run jar anywhere - Production Ready

Dependency Injection

- Pattern to achieve IOC
- Has Factory that creates bean of all classes
- Beans are stored in container

Ioc

- Giving control from developer to framework
- Spring will create objects for us

Spring Initializer

- Here create the barebone springboot project

- @Component - Create obj in container when application context starts
- @Controller - Component + Its Controller
- @RestController - Component + Its Controller + Returns ResponseBody always

---

- Run app : `mvn spring-boot : run`
- Change PORT - application.properties : `server.port = 8082`

- `@RequestMapping (value="/", method = RequestMethod.GET)`
- ~
- `@GetMappiing("/")`

Employee - Company Project

- Tools Used : Spring Boot + MySQL + Lombok + Validation + Mockito

1. application.properties

```
# Database Configuration
spring.datasource.name=empdb
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/empdb?serverTimeZone=UTC
#spring.datasource.url=jdbc:mysql://localhost:3306/empdb?
serverTimeZone=UTC&sessionVariables=sql_mode='NO_ENGINE_SUBSTITUTION'&jdbcComplian
tTruncation=false
spring.datasource.username=root
spring.datasource.password=password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.open-in-view=true
# For Image - Multipart Configuration
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=300MB
#spring.servlet.multipart.max-request-size=15MB
```

2. Employee.java

```
package com.ayushi.Task1.entity;

import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDate;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Employee {
    @Id
    //@GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long empId;

    @NotNull(message = "Name is required")
    @Size(min = 3, message = "Name should have at least 3 characters")
    private String empName;

    @PastOrPresent(message = "Joining date should be today or older")
    private LocalDate empJoiningDate;

    @ManyToOne
    @JoinColumn(name = "comp_id")
```

```
    @NotNull(message = "Company is required")
    private Company company;

}
```

3. EmployeeController.java

```
package com.ayushi.Task1.controller;

import com.ayushi.Task1.entity.Employee;
import com.ayushi.Task1.exception.EmployeeNotFoundException;
import com.ayushi.Task1.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class EmployeeController {
    @Autowired
    private EmployeeService employeeService;

    @PostMapping("/emp")
    public Employee addEmployee(@RequestBody Employee emp) {
        return employeeService.addEmployee(emp);
    }

    @GetMapping("/emps")
    public List<Employee> getEmployees() {
        return employeeService.getEmployees();
    }

    @GetMapping("/emp/{id}")
    public Employee getEmployeeById(@PathVariable("id") Long id) throws
EmployeeNotFoundException {
        return employeeService.getEmployeeById(id);
    }

    @PutMapping("/emp/{id}")
    public Employee updateEmployee(@PathVariable("id") Long id, @RequestBody
Employee emp) {
        return employeeService.updateEmployee(id, emp);
    }

    @DeleteMapping("/emp/{id}")
    public String deleteEmployee(@PathVariable("id") Long id) throws
EmployeeNotFoundException {
        employeeService.deleteEmployee(id);
        return "Record with id " + id + " is deleted!";
```

```
    }

}
```

4. EmployeeService.java

```
package com.ayushi.Task1.service;

import com.ayushi.Task1.entity.Employee;
import com.ayushi.Task1.exception.EmployeeNotFoundException;

import java.util.List;

public interface EmployeeService {
    public Employee addEmployee(Employee emp);

    public List<Employee> getEmployees();

    public Employee getEmployeeById(Long id) throws EmployeeNotFoundException;

    public Employee updateEmployee(Long id, Employee emp);

    public void deleteEmployee(Long id) throws EmployeeNotFoundException;

}
```

5. EmployeeServiceImpl.java

```
package com.ayushi.Task1.service;

import com.ayushi.Task1.entity.Employee;
import com.ayushi.Task1.exception.EmployeeNotFoundException;
import com.ayushi.Task1.repository.EmployeeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Objects;
import java.util.Optional;

@Service
public class EmployeeServiceImpl implements EmployeeService {
    @Autowired
    private EmployeeRepository employeeRepo;

    @Override
```

```
    public Employee addEmployee(Employee emp) {
        return employeeRepo.save(emp);
    }

    @Override
    public List<Employee> getEmployees() {
        return employeeRepo.findAll();
    }

    @Override
    public Employee getEmployeeById(Long id) throws EmployeeNotFoundException {
        Optional<Employee> emp = employeeRepo.findById(id);
        if (!emp.isPresent()) {
            throw new EmployeeNotFoundException("Employee of this Id " + id + " is
not found");
        }
        return emp.get();
    }

    @Override
    public Employee updateEmployee(Long id, Employee emp) {
        Employee empDB = employeeRepo.findById(id).get();
        if (Objects.nonNull(emp.getEmpName()) &&
!"".equalsIgnoreCase(emp.getEmpName())) {
            empDB.setEmpName(emp.getEmpName());
        }
        if (Objects.nonNull(emp.getEmpJoiningDate()) &&
!"".equalsIgnoreCase(String.valueOf(emp.getEmpJoiningDate()))) {
            empDB.setEmpJoiningDate(emp.getEmpJoiningDate());
        }
        return employeeRepo.save(empDB);
    }

    @Override
    public void deleteEmployee(Long id) throws EmployeeNotFoundException {
        Optional<Employee> emp = employeeRepo.findById(id);
        if (!emp.isPresent()) {
            throw new EmployeeNotFoundException("Employee of Id " + id + " is not
found");
        }
        employeeRepo.deleteById(id);
    }
}
```

6. EmployeeRepository.java

```
package com.ayushi.Task1.repository;

import com.ayushi.Task1.entity.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
}
```

EmployeeServiceImplTest

```
package com.ayushi.Task1.service;

import com.ayushi.Task1.entity.Company;
import com.ayushi.Task1.entity.Employee;
import com.ayushi.Task1.exception.EmployeeNotFoundException;
import com.ayushi.Task1.repository.EmployeeRepository;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;

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

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.times;

@SpringBootTest
class EmployeeServiceImplTest {

    @Autowired
    private EmployeeService employeeService;

    @MockBean
    private EmployeeRepository employeeRepo;

    Company comp = Company.builder().CompId(27L).CompName("XYZ Updated
Company").build();
    Employee emp1 = Employee.builder().empId(1L).empName("Emp
1").empJoiningDate(LocalDate.ofEpochDay(2023 - 06 - 11)).company(comp).build();
    Employee emp2 = Employee.builder().empId(2L).empName("Emp
2").empJoiningDate(LocalDate.ofEpochDay(2023 - 06 - 11)).company(comp).build();
    List<Employee> employees = new ArrayList<Employee>();

    @BeforeEach
    void setUp() {
        employees.add(emp1);
```

```
        employees.add(emp2);

        Mockito.when(employeeRepo.findById(1L)).thenReturn(Optional.of(emp1));
        Mockito.when(employeeRepo.findAll()).thenReturn(employees);
        Mockito.when(employeeRepo.save(emp1)).thenReturn(emp1);
        Mockito.doNothing().when(employeeRepo).deleteById(2L);
    }


    @Test
    void addEmployee() {
        assertEquals(emp1.getEmpId(),
employeeService.addEmployee(emp1).getEmpId());
        assertEquals(emp1.getEmpName(),
employeeService.addEmployee(emp1).getEmpName());
        assertEquals(emp1.getEmpJoiningDate(),
employeeService.addEmployee(emp1).getEmpJoiningDate());
        assertEquals(emp1.getCompany(),
employeeService.addEmployee(emp1).getCompany());
    }

    @Test
    void getEmployees() {
        assertEquals(2, employeeService.getEmployees().size());
    }

    @Test
    public void getEmployeeById() throws EmployeeNotFoundException {
        Long empId = 1L;
        String empName = "Emp 1";
        Employee e = employeeService.getEmployeeById(empId);
        assertEquals(empId, e.getEmpId());
        assertEquals(empName, e.getEmpName());
    }

    @Test
    void updateEmployee() {
        Employee emp1Updated = Employee.builder().empId(1L).empName("Emp 1
Updated").empJoiningDate(LocalDate.ofEpochDay(2023 - 06 -
11)).company(comp).build();
        assertEquals(emp1Updated.getEmpName(), employeeService.updateEmployee(1L,
emp1Updated).getEmpName());
        assertNotEquals(emp2.getEmpName(), employeeService.updateEmployee(1L,
emp1Updated).getEmpName());
        assertEquals("Emp 1 Updated", employeeService.updateEmployee(1L,
emp1Updated).getEmpName());
        assertNotEquals("Emp 1", employeeService.updateEmployee(1L,
emp1Updated).getEmpName());
    }

    @Test
//    @Disabled
    void deleteEmployee() throws EmployeeNotFoundException {
        Long id = 1L;
```

```
        employeeService.deleteEmployee(id);
        // Verify that deleteById was called on the repository with the correct id
        Mockito.verify(employeeRepo, Mockito.times(1)).deleteById(id);
    }


}
```

7. EmployeeNotFoundException.java

```
package com.ayushi.Task1.exception;

public class EmployeeNotFoundException extends Exception{
    public EmployeeNotFoundException() {
        super();
    }

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

    public EmployeeNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }

    public EmployeeNotFoundException(Throwable cause) {
        super(cause);
    }

    protected EmployeeNotFoundException(String message, Throwable cause, boolean
enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }
}
```

8. RestResponseEntityExceptionHandler.java

```
package com.ayushi.Task1.exception;

import com.ayushi.Task1.entity.ErrorMessage;
import jakarta.validation.ConstraintViolationException;
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.bind.annotation.ResponseStatus;
```

```
import org.springframework.web.context.request.WebRequest;
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandl
er;

import java.util.HashMap;
import java.util.Map;

@ControllerAdvice
@ResponseStatus
public class RestResponseEntityExceptionHandler extends
ResponseEntityExceptionHandler {
    @ExceptionHandler(EmployeeNotFoundException.class)
    public ResponseEntity<ErrorMessage>
empNotFoundException(EmployeeNotFoundException exception, WebRequest req) {
        ErrorMessage msg = new ErrorMessage(HttpStatus.NOT_FOUND,
exception.getMessage());
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(msg);
    }

    @ExceptionHandler(CompanyNotFoundException.class)
    public ResponseEntity<ErrorMessage>
compNotFoundException(CompanyNotFoundException exception, WebRequest req) {
        ErrorMessage msg = new ErrorMessage(HttpStatus.NOT_FOUND,
exception.getMessage());
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(msg);
    }

    @ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler(ConstraintViolationException.class)
    public ResponseEntity<Object>
handleInvalidArgument(ConstraintViolationException exception) {
        Map<String, String> errorMap = new HashMap<>();
        ErrorMessage errorMessage = null;
        exception.getConstraintViolations().forEach(violation -> {
            String field = violation.getPropertyPath().toString();
            String message = violation.getMessage();
            errorMap.put(field, message);
        });
        for (Map.Entry<String, String> entry : errorMap.entrySet()) {
            errorMessage = new ErrorMessage(HttpStatus.BAD_REQUEST,
entry.getValue());
        }
        return ResponseEntity.badRequest().body(errorMessage);
    }
}
```

9. ErrorMessage.java

```
package com.ayushi.Task1.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.http.HttpStatus;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ErrorMessage {
    private HttpStatus status;
    private String message;
}
```

10. Company.java

```
package com.ayushi.Task1.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Company {
    @Id
    //@GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long CompId;
    private String CompName;

    //@OneToMany(mappedBy = "company", cascade = CascadeType.ALL)
    //private List<Employee> employees;
}
```

11. CompanyController.java

```
package com.ayushi.Task1.controller;

import com.ayushi.Task1.entity.Company;
```

```
import com.ayushi.Task1.exception.CompanyNotFoundException;
import com.ayushi.Task1.service.CompanyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class CompanyController {
    @Autowired
    private CompanyService companyService;

    @PostMapping("/comp")
    public Company addCompany(@RequestBody Company comp) {
        return companyService.addCompany(comp);
    }

    @GetMapping("/comps")
    public List<Company> getCompanies() {
        return companyService.getCompanies();
    }

    @GetMapping("/comp/{id}")
    public Company getCompanyById(@PathVariable("id") Long id) throws
CompanyNotFoundException {
        return companyService.getCompanyById(id);
    }

    @PutMapping("/comp/{id}")
    public Company updateCompany(@PathVariable("id") Long id, @RequestBody Company
comp) {
        return companyService.updateCompany(id, comp);
    }

    @DeleteMapping("/comp/{id}")
    public String deleteCompany(@PathVariable("id") Long id) throws
CompanyNotFoundException {
        companyService.deleteCompany(id);
        return "Record with id " + id + " is deleted!";
    }

    @DeleteMapping("/comps")
    public String deleteAllCompany() {
        companyService.deleteAllCompany();
        return "Records deleted!";
    }
}
```

12. CompanyService.java

```
package com.ayushi.Task1.service;

import com.ayushi.Task1.entity.Company;
import com.ayushi.Task1.exception.CompanyNotFoundException;

import java.util.List;

public interface CompanyService {
    public Company addCompany(Company comp);

    public List<Company> getCompanies();

    public Company getCompanyById(Long id) throws CompanyNotFoundException;

    public Company updateCompany(Long id, Company comp);

    public void deleteCompany(Long id) throws CompanyNotFoundException;

    public void deleteAllCompany();
}
```

13. CompanyServiceImpl.java

```
package com.ayushi.Task1.service;

import com.ayushi.Task1.entity.Company;
import com.ayushi.Task1.exception.CompanyNotFoundException;
import com.ayushi.Task1.repository.CompanyRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Objects;
import java.util.Optional;

@Service
public class CompanyServiceImpl implements CompanyService {
    @Autowired
    private CompanyRepository companyRepo;

    @Override
    public Company addCompany(Company comp) {
        return companyRepo.save(comp);
    }

    @Override
    public List<Company> getCompanies() {
        return companyRepo.findAll();
    }
```

```
    @Override
    public Company getCompanyById(Long id) throws CompanyNotFoundException {
        Optional<Company> comp = companyRepo.findById(id);
        if (!comp.isPresent()) {
            throw new CompanyNotFoundException("Company with id " + id + " not
available");
        }
        return comp.get();
    }

    @Override
    public Company updateCompany(Long id, Company comp) {
        Company compDB = companyRepo.findById(id).get();
        if (Objects.nonNull(comp.getCompName()) &&
!"".equalsIgnoreCase(comp.getCompName())) {
            compDB.setCompName(comp.getCompName());
        }
        return companyRepo.save(compDB);
    }

    @Override
    public void deleteCompany(Long id) throws CompanyNotFoundException {
        Optional<Company> comp = companyRepo.findById(id);
        if (!comp.isPresent()) {
            throw new CompanyNotFoundException("Company with id " + id + " not
available to be deleted");
        }
        companyRepo.deleteById(id);
    }

    @Override
    public void deleteAllCompany() {
        companyRepo.deleteAll();
    }
}
```

14. CompanyRepository.java

```
package com.ayushi.Task1.repository;

import com.ayushi.Task1.entity.Company;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CompanyRepository extends JpaRepository<Company, Long> {
}
```

CompanyServiceImplTest

```
package com.ayushi.Task1.service;

import com.ayushi.Task1.entity.Company;
import com.ayushi.Task1.entity.Employee;
import com.ayushi.Task1.exception.CompanyNotFoundException;
import com.ayushi.Task1.repository.CompanyRepository;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;

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

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

@SpringBootTest
class CompanyServiceImplTest {
    @Autowired
    private CompanyService companyService;

    @MockBean
    private CompanyRepository companyRepo;
    Company comp1 = Company.builder().CompId(27L).CompName("XYZ Updated
Company").build();
    Company comp2 = Company.builder().CompId(28L).CompName("PQR Company").build();
    List<Company> companies = new ArrayList<>();
    Employee emp1 = Employee.builder().empId(1L).empName("Emp
1").empJoiningDate(LocalDate.ofEpochDay(2023 - 06 - 11)).company(comp1).build();

    @BeforeEach
    void setUp() {
        companies.add(comp1);
        companies.add(comp2);

        Mockito.when(companyRepo.findById(27L)).thenReturn(Optional.of(comp1));
        Mockito.when(companyRepo.findAll()).thenReturn(companies);
        Mockito.when(companyRepo.save(comp1)).thenReturn(comp1);
    }

    @Test
    void addCompany() {
        assertEquals(comp1.getCompId(),
companyService.addCompany(comp1).getCompId());
```

```
        assertEquals(comp1.getCompName(),
companyService.addCompany(comp1).getCompName());
    }

    @Test
    void getCompanies() {
        assertEquals(2, companyService.getCompanies().size());
    }

    @Test
    void getCompanyById() throws CompanyNotFoundException {
        Long compId = 27L;
        String compName = "XYZ Updated Company";
        Company comp = companyService.getCompanyById(compId);
        assertEquals(27L, comp.getCompId());
        assertEquals(compName, comp.getCompName());
    }

    @Test
    void updateCompany() {
        Company comp1Updated = Company.builder().CompId(27L).CompName("Comp
Updated").build();
        assertEquals(comp1Updated.getCompName(), companyService.updateCompany(27L,
comp1Updated).getCompName());
        assertEquals("Comp Updated", companyService.updateCompany(27L,
comp1Updated).getCompName());
    }

    @Test
//    @Disabled
    public void deleteCompany() throws CompanyNotFoundException {
        doNothing().when(companyRepo).deleteById(27L);
        companyService.deleteCompany(27L);
        verify(companyRepo,times(1)).deleteById(27L);

    }
    @Test
    void deleteAllCompany() {
        doNothing().when(companyRepo).deleteAll();
        companyService.deleteAllCompany();
        verify(companyRepo,times(1)).deleteAll();

    }
}
```

15. CompanyNotFoundException.java

```
package com.ayushi.Task1.exception;

public class CompanyNotFoundException extends Exception {
    public CompanyNotFoundException() {
```

```
        super();
    }

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

    public CompanyNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }

    public CompanyNotFoundException(Throwable cause) {
        super(cause);
    }

    protected CompanyNotFoundException(String message, Throwable cause, boolean
enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }
}
```

16. Image.java

```
package com.ayushi.Task1.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Image {
    @Id
    @GeneratedValue
    private int id;

    @Lob
    //@Column(name = "image", nullable = false, columnDefinition = "mediumblob")
    private byte[] image;

    private String data;


}
```

17. ImageUploadController.java

```
package com.ayushi.Task1.controller;

import com.ayushi.Task1.entity.Employee;
import com.ayushi.Task1.entity.Image;
import com.ayushi.Task1.repository.ImageRepository;
import com.ayushi.Task1.service.ImageService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
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.multipart.MultipartFile;

import java.io.IOException;
import java.net.MalformedURLException;

@RestController
public class ImageUploadController {
    @Autowired
    private ImageService imageService;

    @Autowired
    private ImageRepository imageRepo;

    @PostMapping("/image")
    public ResponseEntity<String> uploadImage(@RequestParam("keyy") MultipartFile
file) {
        try {
            if (file.isEmpty()) {
                return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Request must Contain
File");
            }
            if (!file.getContentType().equals("image/png")) {
                return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Only PNG Content
type allowed");
            }
            // File Upload In Directory Code
            boolean uploadStatus = imageService.uploadFile(file);
            if (uploadStatus) {
                // File Upload In DB Code
                Image img = new Image();
                img.setImage(file.getBytes());
                img.setData(file.getOriginalFilename());
                System.err.println(img);
                imageRepo.save(img);
                return ResponseEntity.ok("File Uploaded Successfully " +
```

```
file.getOriginalFilename());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Something went
wrong. Please try again!");
    }

    @RequestMapping(value = "/getImageFromDB/{id}", method = RequestMethod.GET,
produces = MediaType.IMAGE_PNG_VALUE)
    public ResponseEntity<byte[]> getImage(@PathVariable("id") int id) {
        byte[] imgDB = imageRepo.findById(id).get().getImage();
        return ResponseEntity.status(HttpStatus.OK)
                .contentType(MediaType.valueOf("image/png")).body(imgDB);
    }

    @GetMapping("/getImageFromDirectory/{fileName}")
    public ResponseEntity<Resource> func(@PathVariable String fileName) throws
MalformedURLException {
        Resource r = imageService.loadFileAsResource(fileName);
        return ResponseEntity.status(HttpStatus.OK)
                .contentType(MediaType.valueOf("image/png")).body(r);
    }

    @PutMapping("/image/{id}")
    public ResponseEntity<String> updateImage(@PathVariable Long id,
@RequestParam("keyy") MultipartFile image) throws IOException {
        Image imgDB = imageRepo.findById(Math.toIntExact(id)).get();
        imgDB.setImage(image.getBytes());
        imgDB.setData(image.getOriginalFilename());
        imageRepo.save(imgDB);
        return ResponseEntity.ok("Image Updated Successfully ");
    }

}
```

18. ImageService.java

```
package com.ayushi.Task1.service;

import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.stereotype.Component;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.net.MalformedURLException;
import java.nio.file.Files;
```

```
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;

@Component
public
class ImageService {
    public final String UPLOAD_DIR =
"C:\\Users\\Ayushi.Jain\\IdeaProjects\\Task1\\src\\main\\resources\\static\\image"
;

    public boolean uploadFile(MultipartFile file) {
        boolean f1 = false;
        try {
            Files.copy(file.getInputStream(), Paths.get(UPLOAD_DIR +
File.separator + file.getOriginalFilename()),
StandardCopyOption.REPLACE_EXISTING);
            f1 = true;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return f1;
    }

    public Resource loadFileAsResource(String fileName) throws
MalformedURLException {
        Path filePath = Paths.get(UPLOAD_DIR +
File.separator).resolve(fileName).normalize();
        Resource resource = new UrlResource(filePath.toUri());
        System.err.println(resource);
        return resource;
    }

}
```

19. ImageRepository.java

```
package com.ayushi.Task1.repository;

import com.ayushi.Task1.entity.Image;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface ImageRepository extends JpaRepository<Image, Integer> {
}
```

20. Main.java

```java
package com.ayushi.Task1;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Task1Application {

    public static void main(String[] args) {
        SpringApplication.run(Task1Application.class, args);
        System.err.println("Started....");
    }

}
```