



IManufacturerToTraineeApplicationTests.java

```
1 package com.infy;
2 import java.time.LocalDate;
16 @SpringBootTest
17 public class IManufacturerToTraineeApplicationTests {
18     @Mock
19     private EmployeeRepository employeeRepository;
20     @InjectMocks
21     private EmployeeServiceImpl employeeServiceImpl = new EmployeeServiceImpl();
22     @Test
23     public void addEmployeeInvalidEmailId() throws Exception {
24         EmployeeDTO empdto=new EmployeeDTO();
25         empdto.setDateOfBirth(LocalDate.of(1997, 9, 29));
26         empdto.setEmailId("sibaprasadpanda100@gmail.com");
27         empdto.setEmployeeId(1234);
28         empdto.setManufacturingUnit(ManufacturingUnit.U001);
29         empdto.setName("siba");
30         EmployeeDTO empdto2=new EmployeeDTO();
31         empdto2.setDateOfBirth(LocalDate.of(1997, 9, 29));
32         empdto2.setEmailId("sibaprasadpanda100@gmail.com");
33         empdto2.setEmployeeId(1234);
34         empdto2.setManufacturingUnit(ManufacturingUnit.U001);
35         empdto2.setName("siba");
36         Mockito.when(employeeRepository.addEmployee(empdto)).thenReturn(empdto.getEmployeeId());
37         Exception exception = Assertions.assertThrows(Exception.class, () -> employeeServiceImpl.addEmployee(empdto2));
38         Assertions.assertEquals("Validator.INVALID_EMAILID", exception.getMessage());
39     }
40     @Test
41     public void getEmployeeInvalidEmployeeId() throws Exception {
42         EmployeeDTO empdto=new EmployeeDTO();
43         empdto.setDateOfBirth(LocalDate.of(1997, 9, 29));
44         empdto.setEmailId("sibaprasadpanda100@gmail.com");
45         empdto.setEmployeeId(1234);
46         empdto.setManufacturingUnit(ManufacturingUnit.U001);
47         empdto.setName("siba");
48         EmployeeDTO empdto2=new EmployeeDTO();
49         empdto2.setDateOfBirth(LocalDate.of(1997, 9, 29));
50         empdto2.setEmailId("sibaprasadpanda100@gmail.com");
51         empdto2.setEmployeeId(12345);
52         empdto2.setManufacturingUnit(ManufacturingUnit.U001);
53         empdto2.setName("siba");
54         Mockito.when(employeeRepository.addEmployee(empdto)).thenReturn(empdto.getEmployeeId());
55         Mockito.when(employeeRepository.getEmployeeDetails(empdto.getEmployeeId())).thenReturn(empdto);
56         Exception exception = Assertions.assertThrows(Exception.class, () -> employeeServiceImpl.getEmployeeDetails(empdto2.getEmployeeId()));
57         Assertions.assertEquals("Service.EMPLOYEE_NOT_FOUND", exception.getMessage());
58     }
59     @Test
60     public void updateEmployeeInvalidEmployeeId() throws Exception {
61         EmployeeDTO empdto=new EmployeeDTO();
62         empdto.setDateOfBirth(LocalDate.of(1997, 9, 29));
63         empdto.setEmailId("sibaprasadpanda100@gmail.com");
64         empdto.setEmployeeId(1234);
65         empdto.setManufacturingUnit(ManufacturingUnit.U001);
66         empdto.setName("siba");
67         EmployeeDTO empdto2=new EmployeeDTO();
68         empdto2.setDateOfBirth(LocalDate.of(1997, 9, 29));
69         empdto2.setEmailId("sibaprasadpanda100@gmail.com");
70         empdto2.setEmployeeId(12345);
71         empdto2.setManufacturingUnit(ManufacturingUnit.U001);
72         empdto2.setName("siba");
```



```
IManufacturerToTraineeApplicationTests.java
38     Assertions.assertEquals("Validator.INVALID_EMAILID", exception.getMessage());
39 }
40 @Test
41 public void getEmployeeInvalidEmployeeId() throws Exception {
42     EmployeeDTO empdto=new EmployeeDTO();
43     empdto.setDateOfBirth(LocalDate.of(1997, 9, 29));
44     empdto.setEmailId("sibaprasadpanda100@gmail.com");
45     empdto.setEmployeeId(1234);
46     empdto.setManufacturingUnit(ManufacturingUnit.U001);
47     empdto.setName("siba");
48     EmployeeDTO empdto2=new EmployeeDTO();
49     empdto2.setDateOfBirth(LocalDate.of(1997, 9, 29));
50     empdto2.setEmailId("sibaprasadpanda100@gmail.com");
51     empdto2.setEmployeeId(12345);
52     empdto2.setManufacturingUnit(ManufacturingUnit.U001);
53     empdto2.setName("siba");
54     Mockito.when(employeeRepository.addEmployee(empdto)).thenReturn(empdto.getEmployeeId());
55     Mockito.when(employeeRepository.getEmployeeDetails(empdto.getEmployeeId())).thenReturn(empdto);
56     Exception exception = Assertions.assertThrows(Exception.class, () -> employeeServiceImpl.getEmployeeDetails(empdto2.getEmployeeId()));
57     Assertions.assertEquals("Service.EMPLOYEE_NOT_FOUND", exception.getMessage());
58 }
59 @Test
60 public void updateEmployeeInvalidEmployeeId() throws Exception {
61     EmployeeDTO empdto=new EmployeeDTO();
62     empdto.setDateOfBirth(LocalDate.of(1997, 9, 29));
63     empdto.setEmailId("sibaprasadpanda100@gmail.com");
64     empdto.setEmployeeId(1234);
65     empdto.setManufacturingUnit(ManufacturingUnit.U001);
66     empdto.setName("siba");
67     EmployeeDTO empdto2=new EmployeeDTO();
68     empdto2.setDateOfBirth(LocalDate.of(1997, 9, 29));
69     empdto2.setEmailId("sibaprasadpanda100@gmail.com");
70     empdto2.setEmployeeId(12345);
71     empdto2.setManufacturingUnit(ManufacturingUnit.U001);
72     empdto2.setName("siba");
73     Mockito.when(employeeRepository.addEmployee(empdto)).thenReturn(empdto.getEmployeeId());
74     Exception exception = Assertions.assertThrows(Exception.class, () -> employeeServiceImpl.updateEmployee(empdto2.getEmployeeId(),empdto2.getEmailId()));
75     Assertions.assertEquals("Service.EMPLOYEE_NOT_FOUND", exception.getMessage());
76 }
77 @Test
78 public void deleteEmployeeInvalidEmployeeId() throws Exception {
79     EmployeeDTO empdto=new EmployeeDTO();
80     empdto.setDateOfBirth(LocalDate.of(1997, 9, 29));
81     empdto.setEmailId("sibaprasadpanda100@gmail.com");
82     empdto.setEmployeeId(1234);
83     empdto.setManufacturingUnit(ManufacturingUnit.U001);
84     empdto.setName("siba");
85     EmployeeDTO empdto2=new EmployeeDTO();
86     empdto2.setDateOfBirth(LocalDate.of(1997, 9, 29));
87     empdto2.setEmailId("sibaprasadpanda100@gmail.com");
88     empdto2.setEmployeeId(12345);
89     empdto2.setManufacturingUnit(ManufacturingUnit.U001);
90     empdto2.setName("siba");
91     Mockito.when(employeeRepository.addEmployee(empdto)).thenReturn(empdto.getEmployeeId());
92     Exception exception = Assertions.assertThrows(Exception.class, () -> employeeServiceImpl.deleteEmployee(empdto2.getEmployeeId()));
93     Assertions.assertEquals("Service.EMPLOYEE_NOT_FOUND", exception.getMessage());
94 }
95 }
96 }
```


eclipse-workspace - IManufacturer_ToTrainee/src/main/java/com/infy/utility/LoggingAspect.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- IManufacturer_ToTrainee
 - src/main/java
 - com.infy
 - IManufacturerToTraineeApplication.java
 - com.infy.dto
 - EmployeeDTO.java
 - ManufacturingUnit.java
 - com.infy.entity
 - Employee.java
 - com.infy.exception
 - InfyEmployeeException.java
 - com.infy.repository
 - EmployeeRepository.java
 - EmployeeRepositoryImpl.java
 - com.infy.service
 - EmployeeService.java
 - EmployeeServiceImpl.java
 - com.infy.utility
 - LoggingAspect.java
 - com.infy.validator
 - Validator.java
 - src/main/resources
 - application.properties
 - log4j2.properties
 - TableScript.sql
 - src/test/java
 - com.infy
 - IManufacturerToTraineeApplicationTests.java
 - JRE System Library [JavaSE-11]

Violations Outline

P	Line	created	Rule
---	------	---------	------

Console JUnit

Finished after 6.438 seconds

Runs: 4/4 Errors: 0 Failures: 0

IManufacturerToTraineeApplicationTests [Runner: JUnit 5] (0.149 s)

- deleteEmployeeInvalidEmployeeId() (0.118 s)
- updateEmployeeInvalidEmployeeId() (0.015 s)
- addEmployeeInvalidEmailId() (0.000 s)
- getEmployeeInvalidEmployeeId() (0.016 s)

Failure Trace

```
1 package com.infy.utility;
2 import org.apache.logging.log4j.LogManager;
3
4 @Component
5 @Aspect
6
7 public class LoggingAspect {
8
9     private Logger logger=LogManager.getLogger(this.getClass());
10
11     @AfterThrowing(pointcut = "execution(* com.infy.repository.*Impl.*(..))", throwing = "exception")
12     public void logDaoException(Exception exception){
13         logger.error(exception.getMessage(), exception);
14     }
15
16     @AfterThrowing(pointcut = "execution(* com.infy.service.*Impl.*(..))", throwing = "exception")
17     public void logServiceException(Exception exception){
18         logger.error(exception.getMessage(), exception);
19     }
20 }
21
22
23
24
25
26
27
```

Writable Smart Insert 1:1:0

00:00 02-03-2021



```
1 package com.infy.service;
2 import org.springframework.beans.factory.annotation.Autowired;
9 @Service(value = "employeeService")
10 @Transactional
11 public class EmployeeServiceImpl implements EmployeeService {
12     @Autowired
13     private EmployeeRepository employeeRepository;
14     @Override
15     public Integer addEmployee(EmployeeDTO employee) throws InfyEmployeeException {
16         Validator.validate(employee);
17         EmployeeDTO empdto=new EmployeeDTO();
18         empdto=employeeRepository.getEmployeeDetails(employee.getEmployeeId());
19         if(empdto==null)
20         {
21             return employeeRepository.addEmployee(employee);
22         }
23         else
24         {
25             throw new InfyEmployeeException("Service.EMPLOYEE_ALREADY_PRESENT");
26         }
27     }
28     @Override
29     public EmployeeDTO getEmployeeDetails(Integer employeeId) throws InfyEmployeeException {
30         EmployeeDTO empdto = employeeRepository.getEmployeeDetails(employeeId);
31         if (empdto == null) {
32             throw new InfyEmployeeException("Service.EMPLOYEE_NOT_FOUND");
33         }
34         return empdto;
35     }
36     @Override
37     public void updateEmployee(Integer employeeId, String emailId) throws InfyEmployeeException {
38         EmployeeDTO emp=employeeRepository.getEmployeeDetails(employeeId);
39         if(emp==null)
40         {
41             throw new InfyEmployeeException("Service.EMPLOYEE_NOT_FOUND");
42         }
43         else
44         {
45             employeeRepository.updateEmployee(employeeId, emailId);
46         }
47     }
48     @Override
49     public void deleteEmployee(Integer employeeId) throws InfyEmployeeException {
50         EmployeeDTO empdto = employeeRepository.getEmployeeDetails(employeeId);
51         if (empdto == null) {
52             throw new InfyEmployeeException("Service.EMPLOYEE_NOT_FOUND");
53         }
54         else
55         {
56             employeeRepository.deleteEmployee(employeeId);
57         }
58     }
59 }
60
```

```
1 package com.infy.repository;
2
3 import javax.persistence.EntityManager;
4 @Repository(value = "employeeRepository")
5 public class EmployeeRepositoryImpl implements EmployeeRepository {
6
7     @PersistenceContext
8     private EntityManager entityManager;
9
10
11     @Override
12     public Integer addEmployee(EmployeeDTO employee) {
13         Employee emp = new Employee();
14         emp.setEmployeeId(employee.getEmployeeId());
15         emp.setDateOfBirth(employee.getDateOfBirth());
16         emp.setEmailId(employee.getEmailId());
17         emp.setName(employee.getName());
18         emp.setManufacturingUnit(employee.getManufacturingUnit());
19         entityManager.persist(emp);
20         return emp.getEmployeeId();
21     }
22
23     @Override
24     public EmployeeDTO getEmployeeDetails(Integer employeeId) {
25         EmployeeDTO empdto = null;
26         Employee emp = entityManager.find(Employee.class, employeeId);
27         if (emp != null) {
28             empdto = new EmployeeDTO();
29             empdto.setEmployeeId(emp.getEmployeeId());
30             empdto.setDateOfBirth(emp.getDateOfBirth());
31             empdto.setEmailId(emp.getEmailId());
32             empdto.setName(emp.getName());
33             empdto.setManufacturingUnit(emp.getManufacturingUnit());
34         }
35         return empdto;
36     }
37
38     @Override
39     public void updateEmployee(Integer employeeId, String emailId) {
40         Employee emp = entityManager.find(Employee.class, employeeId);
41         emp.setEmailId(emailId);
42     }
43
44     @Override
45     public void deleteEmployee(Integer employeeId) {
46         Employee emp = entityManager.find(Employee.class, employeeId);
47         entityManager.remove(emp);
48     }
49 }
50
51
52
53
54
55
56 }
```

```
1 package com.infy.entity;
2 import java.time.LocalDate;
3
4 @Entity
5 public class Employee {
6     @Id
7     private Integer employeeId;
8     private String emailId;
9     @Column(name="employee_name")
10    private String name;
11    @Column(name="dob")
12    private LocalDate dateOfBirth;
13    @Enumerated(value=EnumType.STRING)
14    private ManufacturingUnit manufacturingUnit;
15    public Integer getEmployeeId() {
16        return employeeId;
17    }
18    public void setEmployeeId(Integer employeeId) {
19        this.employeeId = employeeId;
20    }
21    public String getEmailId() {
22        return emailId;
23    }
24    public void setEmailId(String emailId) {
25        this.emailId = emailId;
26    }
27    public String getName() {
28        return name;
29    }
30    public void setName(String name) {
31        this.name = name;
32    }
33    public LocalDate getDateOfBirth() {
34        return dateOfBirth;
35    }
36    public void setDateOfBirth(LocalDate dateOfBirth) {
37        this.dateOfBirth = dateOfBirth;
38    }
39    public ManufacturingUnit getManufacturingUnit() {
40        return manufacturingUnit;
41    }
42    public void setManufacturingUnit(ManufacturingUnit manufacturingUnit) {
43        this.manufacturingUnit = manufacturingUnit;
44    }
45    @Override
46    public String toString() {
47        return "Employee [employeeId=" + employeeId + ", emailId=" + emailId + ", name=" + name + ", dateOfBirth="
48            + dateOfBirth + ", manufacturingUnit=" + manufacturingUnit + "];"
49    }
50    @Override
51    public int hashCode() {
52        final int prime = 31;
53        int result = 1;
54        result = prime * result + ((employeeId == null) ? 0 : employeeId.hashCode());
55        return result;
56    }
57    @Override
58    public boolean equals(Object obj) {
59        if (this == obj)
60            return true;
61        if (obj == null)
62            return false;
63        if (getClass() != obj.getClass())
64            return false;
65        Employee other = (Employee) obj;
66        if (employeeId == null) {
67            if (other.employeeId != null)
68                return false;
69        } else if (!employeeId.equals(other.employeeId))
70            return false;
71        if (emailId == null) {
72            if (other.emailId != null)
73                return false;
74        } else if (!emailId.equals(other.emailId))
75            return false;
76        if (name == null) {
77            if (other.name != null)
78                return false;
79        } else if (!name.equals(other.name))
80            return false;
81        if (dateOfBirth == null) {
82            if (other.dateOfBirth != null)
83                return false;
84        } else if (!dateOfBirth.equals(other.dateOfBirth))
85            return false;
86        if (manufacturingUnit == null) {
87            if (other.manufacturingUnit != null)
88                return false;
89        } else if (!manufacturingUnit.equals(other.manufacturingUnit))
90            return false;
91        return true;
92    }
93 }
```

```
22     }
23     public void setEmployeeId(Integer employeeId) {
24         this.employeeId = employeeId;
25     }
26     public String getEmailId() {
27         return emailId;
28     }
29     public void setEmailId(String emailId) {
30         this.emailId = emailId;
31     }
32     public String getName() {
33         return name;
34     }
35     public void setName(String name) {
36         this.name = name;
37     }
38     public LocalDate getDateOfBirth() {
39         return dateOfBirth;
40     }
41     public void setDateOfBirth(LocalDate dateOfBirth) {
42         this.dateOfBirth = dateOfBirth;
43     }
44     public ManufacturingUnit getManufacturingUnit() {
45         return manufacturingUnit;
46     }
47     public void setManufacturingUnit(ManufacturingUnit manufacturingUnit) {
48         this.manufacturingUnit = manufacturingUnit;
49     }
50     @Override
51     public String toString() {
52         return "Employee [employeeId=" + employeeId + ", emailId=" + emailId + ", name=" + name + ", dateOfBirth=" +
53             + dateOfBirth + ", manufacturingUnit=" + manufacturingUnit + "]";
54     }
55     @Override
56     public int hashCode() {
57         final int prime = 31;
58         int result = 1;
59         result = prime * result + ((employeeId == null) ? 0 : employeeId.hashCode());
60         return result;
61     }
62     @Override
63     public boolean equals(Object obj) {
64         if (this == obj)
65             return true;
66         if (obj == null)
67             return false;
68         if (getClass() != obj.getClass())
69             return false;
70         Employee other = (Employee) obj;
71         if (employeeId == null) {
72             if (other.employeeId != null)
73                 return false;
74         }
75         else if (!employeeId.equals(other.employeeId))
76             return false;
77         return true;
78     }
79 }
80
```