Employee Management CRUD Project with Testing (Spring Boot + MySQL + JUnit 5 + Mockito)

1. Project Structure

---

employee-management/

■■■ src/main/java/

■ ■■■ com.example.employee/

■ ■■■ controller/

■ ■■■ service/

■ ■■■ repository/

■ ■■■ entity/

■ ■■■ exception/

■■■ src/test/java/

■ ■■■ com.example.employee/

■ ■■■ controller/

■ ■■■ service/

■ ■■■ repository/

■■■ pom.xml

2. Employee Entity

```
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Employee {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
private String name;
private String department;
private Double salary;
}
```

3. Repository Layer

```
public interface EmployeeRepository extends JpaRepository {}
```

4. Service Layer

```
interface EmployeeService {
Employee saveEmployee(Employee employee);
Employee getEmployee(Long id);
List getAllEmployees();
Employee updateEmployee(Long id, Employee employee);
void deleteEmployee(Long id);
}
```

```
@Service
@RequiredArgsConstructor
public class EmployeeServiceImpl implements EmployeeService {
private final EmployeeRepository repository;
public Employee saveEmployee(Employee employee) { return repository.save(employee); }
public Employee getEmployee(Long id) {
return repository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Employee Not Found : " + id));
}
public List getAllEmployees() { return repository.findAll(); }
public Employee updateEmployee(Long id, Employee updated) {
Employee emp = getEmployee(id);
emp.setName(updated.getName());
emp.setDepartment(updated.getDepartment());
emp.setSalary(updated.getSalary());
return repository.save(emp);
}
public void deleteEmployee(Long id) { repository.deleteById(id); }
}
```

5. Controller Layer

```
@RestController
@RequestMapping("/api/employees")
@RequiredArgsConstructor
public class EmployeeController {
private final EmployeeService service;
@PostMapping public Employee save(@RequestBody Employee employee) { return service.saveEmployee(employee); }
@GetMapping("/{id}") public Employee getById(@PathVariable Long id) { return service.getEmployee(id); }
```

@GetMapping public List getAll() { return service.getAllEmployees(); }

@PutMapping("/{id}") public Employee update(@PathVariable Long id, @RequestBody Employee employee) { return service.updateEmployee(id, employee); }

@DeleteMapping("/{id}") public String delete(@PathVariable Long id) { service.deleteEmployee(id); return "Employee deleted successfully!"; }

}

6. Custom Exception

---

public class ResourceNotFoundException extends RuntimeException {

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

}

7. JUnit 5 + Mockito Tests

- Service Layer Test
- Controller Layer Test using MockMvc
- Testing CRUD operations
- Mocking repository calls
- Verifying interactions

Practical Example:

@Test

void testSaveEmployee() {

Employee emp = new Employee(1L, "Arjun", "IT", 50000.0);

when(repository.save(emp)).thenReturn(emp);

Employee saved = service.saveEmployee(emp);

assertEquals("Arjun", saved.getName());

verify(repository, times(1)).save(emp);

}

8. Real-Time Features

- CRUD operations
- Exception Handling
- Unit Testing (JUnit + Mockito)
- Controller Testing (MockMvc)
- MySQL Integration