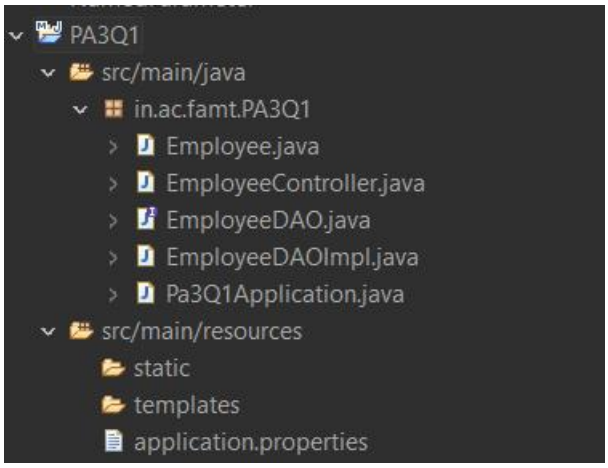
Programming Assignment No.3
Getting Started with Spring Boot

Q.1 Create a maven-based project to demonstrate the RESTful web service for an Employee database resource.

Implementation:



Code:

Employee.java

```
package in.ac.famt.PA3Q1;

public class Employee {
   private int id;
   private String name;
   private String address;

   // Constructors, getters, setters

   @Override
         public String toString() {
                  return "Employee [id=" + id + ", name=" + name + ", address=" +
address + "]";
         }

         public Employee() {
   }

   public Employee(int id, String name, String address) {
      this.id = id;
      this.name = name;
      this.address = address;
   }

   public int getId() {
      return id;
   }

   public void setId(int id) {
      this.id = id;
   }

   public String getName() {
      return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
}
```

EmployeeDAO.java

```
package in.ac.famt.PA3Q1;

import java.util.List;

public interface EmployeeDAO {
        int saveEmployee(Employee employee);

    List<Employee> getAllEmployees();

    Employee getEmployeeById(int employeeId);

    int updateEmployee(int id, Employee updatedEmployee);

    int deleteEmployee(int id);
}
```

EmployeeDAOImpl.java

```
package in.ac.famt.PA3Q1;

import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class EmployeeDAOImpl implements EmployeeDAO {

    private final JdbcTemplate jdbcTemplate;

    // Constructor injection
    public EmployeeDAOImpl(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    public int saveEmployee(Employee employee) {
        String sql = "INSERT INTO employee (id, name, address) VALUES (?, ?, ?)";
        return jdbcTemplate.update(sql, employee.getId(), employee.getName(),
employee.getAddress());
    }
```

```
    public List<Employee> getAllEmployees() {
        String sql = "SELECT * FROM employee";
        return jdbcTemplate.query(sql, new
BeanPropertyRowMapper<>(Employee.class));
    }

    @SuppressWarnings("deprecation")
            public Employee getEmployeeById(int id) {
        String sql = "SELECT * FROM employee WHERE id = ?";
        return jdbcTemplate.queryForObject(sql, new Object[]{id}, new
BeanPropertyRowMapper<>(Employee.class));
    }

    public int updateEmployee(int id, Employee updatedEmployee) {
        String sql = "UPDATE employee SET name = ?, address = ? WHERE id = ?";
        return jdbcTemplate.update(sql, updatedEmployee.getName(),
updatedEmployee.getAddress(), id);
    }

    public int deleteEmployee(int id) {
        String sql = "DELETE FROM employee WHERE id = ?";
        return jdbcTemplate.update(sql,id);
    }
}
```

EmployeeController.java

```
package in.ac.famt.PA3Q1;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

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

@RestController
@RequestMapping("/employees")
public class EmployeeController {
    private final List<Employee> employees = new ArrayList<>();

    @GetMapping
    public List<Employee> getAllEmployees() {
        return employees;
    }

    @GetMapping("/{id}")
    public ResponseEntity<Employee> getEmployeeById(@PathVariable int id) {
        Employee employee = findEmployeeById(id);
        if (employee != null) {
            return ResponseEntity.ok().body(employee);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @PostMapping
```

```
    public ResponseEntity<Employee> createEmployee(@RequestBody Employee
employee) {
        // Generate a unique ID for the new employee (for simplicity, you can use the list
size)
        int newEmployeeId = (int) (employees.size() + 1);
        employee.setId(newEmployeeId);

        employees.add(employee);

        return ResponseEntity.status(HttpStatus.CREATED).body(employee);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Employee> updateEmployee(@PathVariable int id,
@RequestBody Employee updatedEmployee) {
        Employee employee = findEmployeeById(id);

        if (employee != null) {
            // Update the existing employee with new data
            employee.setName(updatedEmployee.getName());
            employee.setAddress(updatedEmployee.getAddress());
            // Update other fields as needed

            return ResponseEntity.ok().body(employee);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteEmployee(@PathVariable int id) {
        Employee employee = findEmployeeById(id);

        if (employee != null) {
            employees.remove(employee);
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    private Employee findEmployeeById(int id) {
        return employees.stream()
                .filter(e -> e.getId() == id)
                .findFirst()
                .orElse(null);
    }
}
```

Pa3Q1Application.java

```
package in.ac.famt.PA3Q1;

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

@SpringBootApplication
```

```
public class Pa3Q1Application {

	public static void main(String[] args) {
		//SpringApplication.run(Pa3Q1Application.class, args);
		ApplicationContext context =
SpringApplication.run(Pa3Q1Application.class, args);
	EmployeeDAO employeeDAO = context.getBean(EmployeeDAO.class);

	// Example usage
	Employee employee = new Employee(4, "Vishakha", "123 Main St");
	int saveResult = employeeDAO.saveEmployee(employee);

	if (saveResult != 0) {
	   System.out.println("Employee data saved...");
	} else {
	   System.out.println("Employee data not saved....");
	}

	System.out.println("Following is the List of the Employees:");
	System.out.println(employeeDAO.getAllEmployees());

	// Example of getting employee by ID
	int employeeId = 4;
	Employee retrievedEmployee = employeeDAO.getEmployeeById(employeeId);
	System.out.println("Retrieved Employee: " + retrievedEmployee);

	// Example of updating employee
	Employee updatedEmployee = new Employee(4, "Vishakha Updated", "456
Second St");
	int updateResult = employeeDAO.updateEmployee(employeeId,
updatedEmployee);
	if (updateResult != 0) {
	   System.out.println("Employee data updated...");
	} else {
	   System.out.println("Employee data not updated....");
	}

	// Example of deleting employee
	int deleteResult = employeeDAO.deleteEmployee(employeeId);
	if (deleteResult != 0) {
	   System.out.println("Employee data deleted...");
	} else {
	   System.out.println("Employee data not deleted....");
	}
	   }

}
```

Output:

```
Employee data saved...
Following is the List of the Employees:
[Employee [id=1, name=Onkar, address=123 Main St], Employee [id=2, name=Aditya, address=456 Oak Ave], Employee [id=3, name=Dattu, ad
Retrieved Employee: Employee [id=4, name=Vishakha, address=123 Main St]
Employee data updated...
Employee data deleted...
```