**Objectives**

* Demonstrate integration of RESTful Web Service of type GET and test the service using postman.
  + REST Web Service architecture with Controller, Service and Dao, service methods.

A RESTful Web Service is an application program interface (API) that uses HTTP methods (GET, POST, PUT, DELETE) to perform CRUD operations on resources. It adheres to the principles of REST (Representational State Transfer) architecture.

**Layered Architecture :**

To maintain separation of concerns, the REST web service follows a layered structure:

**1. Controller Layer**

* Exposes API endpoints to the client.
* Maps HTTP methods (e.g., GET) using annotations like @GetMapping.
* Delegates logic to the Service layer.

**Example:** Handles GET /employees

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

**2. Service Layer**

* Contains business logic.
* Acts as a bridge between the Controller and DAO layers.
* Annotated with @Service.
* Methods can be marked as @Transactional when needed.

**Example:**

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

**3. DAO (Data Access Object) Layer**

* Handles direct access to data.
* In our case, it fetches static data (via Spring XML config).
* Typically interacts with the database or other data sources in real-world applications.

**Example:**

public ArrayList<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

**Data Flow for GET Request (/employees)**

1. Client (e.g., Postman or Angular) sends a GET request to /employees.
2. The Controller receives the request and invokes the service method.
3. The Service Layer contains business logic and calls the DAO layer.
4. The DAO fetches employee data (from XML or DB).
5. The data flows back to the client in JSON format.

**Testing the REST Service Using Postman**

**Steps:**

1. **Start Spring Boot application** (e.g., using mvn spring-boot:run).
2. **Open Postman**.
3. **Send GET Request**:
   * **Method**: GET
   * **URL**: http://localhost:8080/employees
4. **Click Send**.
5. **Response**:
   * A JSON array of employee objects will be displayed.

[

{

"id": 1,

"name": "Alice",

"department": "HR",

"skill": "Communication"

},

{

"id": 2,

"name": "Bob",

"department": "Engineering",

"skill": "Java"

}

// ...

]