WEEK-10

AIM: Implement basic CRUD (Create, Read, Update, Delete) operations on a MySQL database using JDBC in Java.

Explore Spring's stereotype annotations like @Component, @Service, @Repository, and @Controller to define beans with specific roles in the application.

Implement a Spring JDBC Template to interact with a MySQL relational database

CODE:

- **application.properties**

```
spring.application.name=spring-crud-jdbc

spring.datasource.url=jdbc:mysql://localhost:3306/studentdb

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.show-sql=true
```

- **JdbcConfig.java**

```
package com.example.demo.config;

import javax.sql.DataSource;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.core.JdbcTemplate;

@Configuration

public class JdbcConfig {

   @Bean

   public JdbcTemplate jdbcTemplate(DataSource dataSource) {

      return new JdbcTemplate(dataSource);

   }
```

}

- **StudentController.java**

```
package com.example.demo.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import com.example.demo.model.Student;
import com.example.demo.service.StudentService;
@RestController
@RequestMapping("/students")
public class StudentController {
   @Autowired
   private StudentService service;
   @PostMapping("/add")
   public String add(@RequestBody Student s) {
      service.save(s);
      return "Student added!";
   }
   @GetMapping("/get")
   public List<Student> list() {
      return service.listAll();
   }
   @GetMapping("/{id}")
   public Student get(@PathVariable int id) {
      return service.get(id);
   }
   @PutMapping("/update")
   public String update(@RequestBody Student s) {
      service.update(s);
```

```
        return "Student updated!";
    }
    @DeleteMapping("/delete/{id}")
    public String delete(@PathVariable int id) {
        service.delete(id);
        return "Student deleted!";
    }
}
```

Student.java

```
package com.example.demo.model;
public class Student {
    private int id;
    private String name;
    private String email;
    public Student() {}
    public Student(int id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }
    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 getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
}
```

- **StudentRepository.java**

```
package com.example.demo.repository;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
import com.example.demo.model.Student;
@Repository
public class StudentRepository {
   @Autowired
   private JdbcTemplate jdbcTemplate;
   public int save(Student s) {
      return jdbcTemplate.update(
         "INSERT INTO student(name, email) VALUES (?, ?)",
         s.getName(), s.getEmail());
   }
   public List<Student> listAll() {
      return jdbcTemplate.query("SELECT * FROM student",
         new BeanPropertyRowMapper<>(Student.class));
   }
   public Student getById(int id) {
      return jdbcTemplate.queryForObject(
         "SELECT * FROM student WHERE id = ?",
         new BeanPropertyRowMapper<>(Student.class),
         id
      );
   }
```

```
    public int update(Student s) {
        return jdbcTemplate.update(
            "UPDATE student SET name=?, email=? WHERE id=?",
            s.getName(), s.getEmail(), s.getId());
    }
    public int delete(int id) {
        return jdbcTemplate.update(
            "DELETE FROM student WHERE id=?", id);
    }
}
```

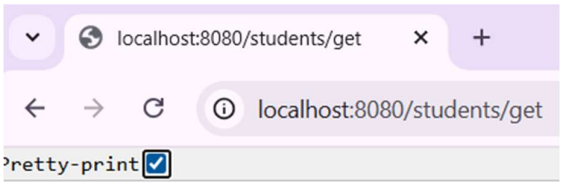
- **StudentService.java**

```
package com.example.demo.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.demo.model.Student;
import com.example.demo.repository.StudentRepository;
@Service
public class StudentService {
    @Autowired
    private StudentRepository repo;
    public int save(Student s) { return repo.save(s); }
    public List<Student> listAll() { return repo.listAll(); }
    public Student get(int id) { return repo.getById(id); }
    public int update(Student s) { return repo.update(s); }
    public int delete(int id) { return repo.delete(id); }
}
```

OUTPUT:

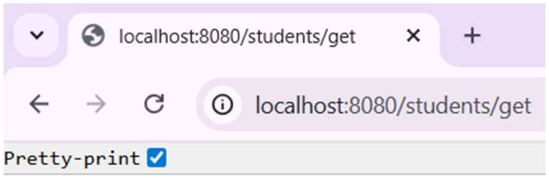
```
mysql> create database studentdb;
Query OK, 1 row affected (0.02 sec)

mysql> use studentdb;
Database changed
mysql> create table student(
    -> id int primary key auto_increment,
    -> name varchar(50),
    -> email varchar(50)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> INSERT INTO student(name, email) VALUES ("Manu", "manu@gmail.com");
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO student(name, email) VALUES ("Sai", "sai@gmail.com");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student(name, email) VALUES ("Koushi", "koushi@gmail.com");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student(name, email) VALUES ("Vamshi", "vamshi@gmail.com");
Query OK, 1 row affected (0.00 sec)
```



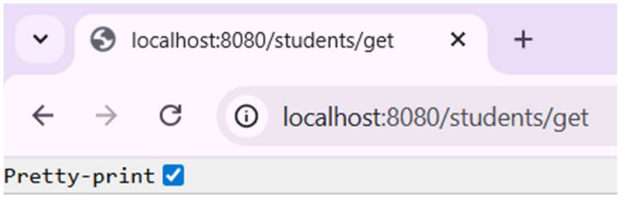

```
[
  {
    "id": 1,
    "name": "Manu",
    "email": "manu@gmail.com"
  },
  {
    "id": 2,
    "name": "Sai",
    "email": "sai@gmail.com"
  },
  {
    "id": 3,
    "name": "Koushi",
    "email": "koushi@gmail.com"
  },
  {
    "id": 4,
    "name": "Vamshi",
    "email": "vamshi@gmail.com"
  }
]
```

```
mysql> delete from student where id=4;
Query OK, 1 row affected (0.04 sec)
```




```
[
  {
    "id": 1,
    "name": "Manu",
    "email": "manu@gmail.com"
  },
  {
    "id": 2,
    "name": "Sai",
    "email": "sai@gmail.com"
  },
  {
    "id": 3,
    "name": "Koushi",
    "email": "koushi@gmail.com"
  }
]
```


```
mysql> UPDATE student SET name="Manogna Updated" WHERE id=1;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```




```
[
  {
    "id": 1,
    "name": "Manogna Updated",
    "email": "manu@gmail.com"
  },
  {
    "id": 2,
    "name": "Sai",
    "email": "sai@gmail.com"
  },
  {
    "id": 3,
    "name": "Koushi",
    "email": "koushi@gmail.com"
  }
]
```

RESULTS:

CRUD Operations are successfully done.

Spring's stereotype annotations @Component, @Service, @Repository, and @Controller are used.

MySQL database is used.