AIM: Introduction to spring boot and restful services.

DESCRIPTION:

Spring Boot and Spring MVC are two popular frameworks for developing Java-based web applications.

Spring Boot is a framework for creating stand-alone, production-grade applications that run with minimal effort. It provides an easy way to set up and configure a web application using the Spring framework and its various components, such as Spring MVC.

Spring MVC is a module in the Spring framework that provides a Model-View-Controller (MVC) architecture for building web applications. It enables developers to build web applications with a clear separation of responsibilities between the model, view, and controller components.

Both Spring Boot and Spring MVC use annotations extensively to provide configuration and setup information to the framework. Some common annotations used in Spring MVC are:

- @Controller: Annotation used on a class to indicate that it is a controller class, responsible for handling incoming web requests.
- @RequestMapping: Annotation used on a method to specify the URL that the method should handle.
- @ResponseBody: Annotation used on a method to indicate that the method's return value should be used as the response body.
- @Autowired: Annotation used to automatically inject a bean into a field, method, or constructor.

Some common annotations used in Spring Boot are:

- @SpringBootApplication: Annotation used to enable configuration and setup of a Spring Boot application.
- @EnableAutoConfiguration: Annotation used to enable automatic configuration of a Spring Boot application
  based on the jars on the classpath and other factors.
- @Configuration: Annotation used to specify that a class is a configuration class, used to provide configuration
  information to the framework.
- @Bean: Annotation used to specify that a method should return a bean to be registered with the application context.
- @CrossOrigin is used to enable Cross-Origin Resource Sharing (CORS) in a Spring application. CORS is a
  security feature that restricts a web page from making requests to a different domain. By using the
  @CrossOrigin annotation, you can allow requests from specific domains to access your RESTful web
  services.
- @RestController is a specialized form of the @Controller annotation in Spring MVC. It is used to indicate that a class is a controller class responsible for handling RESTful web requests. By using the @RestController

> annotation, you can return plain data (such as JSON or XML) instead of a view, which is typical in a standard MVC setup.

These are just a few examples of the annotations used in Spring Boot and Spring MVC, and there are many more available to customize and extend the functionality of these frameworks.

- **context.xml**

```
<bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName"
                value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="2001" />
</bean>
<bean id="jdbcTemplate"
        class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource"></property>
</bean>
<bean id="edao4" class="com.flash.q4.EmployeeDao">
        <property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
```

- **pom.xml**

```
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jdbc</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>

        <dependency>
                <groupId>com.mysql</groupId>
                <artifactId>mysql-connector-j</artifactId>
                <scope>runtime</scope>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
</dependencies>
```

- **App.java**

```
package com.flash;

import org.springframework.boot.SpringApplication; import
org.springframework.boot.autoconfigure.SpringBootApplication; import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

@SpringBootApplication(exclude={DataSourceAutoConfiguration.class})
public class SpringJdbc1Application { public
	static void main(String[] args) {
		SpringApplication.run(SpringJdbc1Application.class, args);
	}
}
```

- **MyController.java**

```
package com.flash.controller;

import java.util.List;

import org.springframework.context.ApplicationContext; import
org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.CrossOrigin; import
org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.RestController;

import com.flash.entity.Employee;
import com.flash.q4.EmployeeDao;

@CrossOrigin
@RestController public
class MyController {

	@GetMapping("/getEmployee") public
	List<Employee> getEmployees(){
		ApplicationContext context = new
		ClassPathXmlApplicationContext("context.xml"); EmployeeDao dao =
		context.getBean("edao4", EmployeeDao.class); return dao.getEmployees();
	}
}
```

- **Employee.java**

```
package com.flash.entity;

public class Employee {
	private int id; private String name; private float
	salary; public Employee(int id, String name, float
	salary) {
		super(); this.id =
		id; this.name =
		name; this.salary =
		salary;
	}
```

```
	@Override
	public String toString() { return "Employee [id=" + id + ", name=" + name + ",
		salary=" + salary + "]";
	}
	public int getId() {
		return id;
	} public String
	getName() { return name;
	}
	public float getSalary() {
		return salary;
	}
	public void setId(int id) {
		this.id = id;
	} public void setName(String
	name) { this.name = name;
	}
	public void setSalary(float salary) {
		this.salary = salary;
	}
}
```

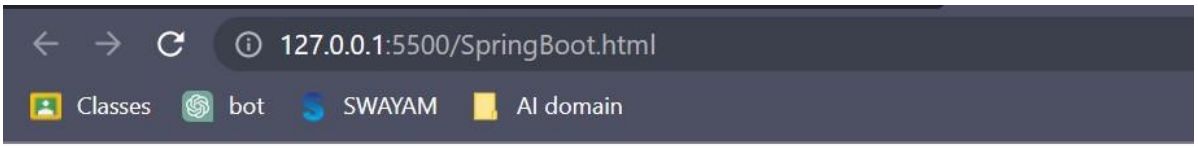
- **EmployeeDao.java**

```
package com.flash.q4;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import org.springframework.dao.DataAccessException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.ResultSetExtractor;

import com.flash.entity.Employee;

public class EmployeeDao { private JdbcTemplate jdbcTemplate;
	public EmployeeDao() {} public void
	setJdbcTemplate(JdbcTemplate jdbcTemplate) {
	this.jdbcTemplate = jdbcTemplate;
	} public List<Employee> getEmployees(){ return
jdbcTemplate.query("select * from employee1", new
ResultSetExtractor<List<Employee>>() { @Override
			public List<Employee> extractData(ResultSet rs) throws SQLException,
DataAccessException {
				List<Employee> list = new ArrayList<>(); while(rs.next()) list.add(new
				Employee(rs.getInt(1), rs.getString(2), rs.getFloat(3)));
				return list;
			}
		});
```

```
        }
}
```

- **SpringBoot.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
  <div class="container m-3">
    <div class="row bg-secondary text-center">
      <h4>Data is coming from spring-web</h4>
    </div>
    <table class="table">
      <thead>
        <tr>
          <th scope="col">ID</th>
          <th scope="col">Name</th>
          <th scope="col">Salary</th>
        </tr>
      </thead>
      <tbody id="tb"></tbody>
    </table>
  </div>
  <script>
    fetch('http://localhost:8080/getEmployee')
      .then((response) => response.json())
      .then(data => { let tbody = ''; for (const item of data)
        { tbody += `<tr><th scope=row'>${item.id}</th>
                <td>${item.name}</td>
                <td>${item.salary}</td></tr>`;
        } document.getElementById('tb').innerHTML =
        tbody;
      })
      .catch(error => console.error(error));
  </script>
</body>
</html>
```



Data is coming from spring-web

ID	Name	Salary
17	nayan	123500
19	shridhan	2200
45	sahil	2345400
54	nikhil	100000
34	kailas	552200

CONCLUSION:

From this practical, I have learned how to create and implement REST API in spring.