DEPARTMENT OF CSE
COURSE CODE: 23SDCS12A / 23SDCS12R
FULL STACK APPLICATION DEVELOPMENT

Date of the Session: __/__/_____ **Time of The Session:_____to_____**

LAB – 7 ➔ Spring Boot Web MVC Demo & Annotations

Prerequisites:

General Idea on Spring Boot MVC Architecture

Exercise:

1. Create a Spring Boot Web MVC application that demonstrates various request mappings and handling methods using a customer object. Implement the following demo operations
2. Create a method that maps to the URL "/demo1" and returns an integer.
3. Create a method that maps to the URL "/demo2" and returns a double value
4. Create a method that maps to the URL "/demo3" using @GetMapping and returns a formatted HTML string.
5. Create a method that maps to the URL "/demo4" using @GetMapping, initializes a string "KLEF", and returns a concatenated string "I Study at " with the initialized string.
6. Create a method that maps to the URL "/demo5/{id}" using @GetMapping, accepts a path variable "id", and returns its value.
7. Create a method that maps to the URL "/demo6/{a}/{b}" using @GetMapping, accepts two path variables "a" and "b", and returns their sum as a string.
8. Create a method that maps to the URL "/demo7" using @GetMapping, accepts a request parameter "id", and returns it as a string.
9. Create a method that maps to the URL "/demo8/{name}" using @GetMapping, accepts a path variable "name", and returns it as a string.
10. Create a method that maps to the URL "/addcustomer" using @PostMapping, accepts a Customer object in the request body, adds it to a list, and returns a confirmation message "Customer Added Successfully".
11. Create a method that maps to the URL "/viewcustomer" using @GetMapping and returns the list of all Customer objects added.

❖ Watch The Video And Do In Eclipse Workspace

7a https://youtu.be/yh-DnEb5qaY?si=lCoy24D-DmBttN1Z

7b https://youtu.be/6ZiregY_eIE?si=D-ROVFwTwI2aSGF1

application.properties

```
spring.application.name=Exp7
#http://localhost:8081
server.port=8081
```

Exp7Application.java

```
package com.example.demo;

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

@SpringBootApplication
public class Exp7Application {

	public static void main(String[] args) {
		SpringApplication.run(Exp7Application.class, args);
	}

}
```

AppController.java

```
package com.example.demo;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

import jakarta.websocket.server.PathParam;
```

```
@Controller
public class AppController {

   @GetMapping("/demo1")
   @ResponseBody
   public int fun1() {
      return 10;
   }

   @GetMapping("/demo2")
   @ResponseBody
   public Double fun2() {
      return 10.5;
   }

   @GetMapping("/demo3")
   public String fun3() {
      return "first.html";
   }

   @GetMapping("/demo4")
   @ResponseBody
   public String fun4() {
      String univ = "KLEF";
      return "I study at " + univ;
   }

   @GetMapping("/demo5/{id}")
   @ResponseBody
   public String fun5(@PathVariable int id) {
      return "" + id;
   }
```

```
    @GetMapping("/demo6/{a}/{b}")
    @ResponseBody
    public String fun6(@PathVariable int a, @PathVariable int b) {
        return "" + (a + b);
    }

    @GetMapping("/demo7")
    @ResponseBody
    public String fun7(@PathParam("id") int id) {
        return "" + id;
    }


    @GetMapping("/demo8/{name}")
    @ResponseBody
    public String fun8(@PathVariable String name) {
        return name;
    }

    List<User> l = new ArrayList<>();

    @PostMapping("/demo8")
    @ResponseBody
    public String fun9(@RequestBody User user ) {
            l.add(user);
        return "User Added";
    }

    @GetMapping("/demo8")
    @ResponseBody
    public List <User> fun10() {
        return l;
    }
}
```

first.html

```
<html>
<body>
This is a First HTML Page
</body>
</html>
```

User.java

```
package com.example.demo;

public class User {

	int id;
	String name;
	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;
	}
	@Override
	public String toString() {
		return "User [id=" + id + ", name=" + name + ", getId()=" + getId() + ",
getName()=" + getName()
				+ ", getClass()=" + getClass() + ", hashCode()=" + hashCode() + ",
toString()=" + super.toString()
				+ "]";
	}
}
```

VIVA QUESTIONS:

1. What are the key differences between Spring MVC and Spring Boot MVC?

- Spring MVC needs manual config; Spring Boot provides auto-config.
- Spring Boot has an embedded server; Spring MVC requires an external server.
- Spring Boot simplifies dependency management with starters.

2. How does Spring Boot simplify the development of Spring applications?

- Auto-configuration, embedded servers, starter dependencies, and CLI support.

3. Explain how to use @RequestBody and @ResponseBody annotations in Spring Boot.

- `@RequestBody` : Converts JSON to Java object.
- `@ResponseBody` : Converts Java object to JSON response.

4. Explain the role of the @Autowired annotation in Spring Boot and how it works in the context of Dependency Injection?

- Injects dependencies automatically (Constructor, Setter, or Field Injection).

5. What is the purpose of application.properties or application.yml in a Spring Boot application?

- Configures ports, database, logging, and security settings.

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured _____ out of 50 Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: