

Practical No 06

1) Create “HelloWorld” application using a Spring boot.

DemoApplication.java

```
package com.example.demo;

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

@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
        ApplicationContext context=SpringApplication.run(DemoApplication.class, args);
        System.out.println("Hello world using spring...");
    }
}
```

Output:

```
Hello world using spring...
```

2) Create a class to demonstrate the database connectivity using spring boot. Use the problem statement and question 3 of the practical number 5.

CustomerOrderController.java

```
package com.example.demo.controller;
import com.example.demo.model.CustomerOrder;
import com.example.demo.service.CustomerOrderService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;
@RestController
@RequestMapping("/customers")
public class CustomerOrderController {
    @Autowired
    private CustomerOrderService customerOrderService;
```

```

    @GetMapping("/orders")
    public List<CustomerOrder> getAllCustomersWithOrders() {
        return customerOrderService.displayCustomersWithOrders();
    }
}

```

CustomerOrderResultSetExtractor.java

```

package com.example.demo.extractor;

import com.example.demo.model.CustomerOrder;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.ResultSetExtractor;

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

public class CustomerOrderResultSetExtractor implements ResultSetExtractor<List<CustomerOrder>> {
    @Override
    public List<CustomerOrder> extractData(ResultSet rs) throws SQLException, DataAccessException {
        Map<Integer, CustomerOrder> customerMap = new HashMap<>();

        while (rs.next()) {
            int customerId = rs.getInt("customer_id");
            CustomerOrder customer = customerMap.get(customerId);
            if (customer == null) {
                customer = new CustomerOrder();
                customer.setCustomerId(customerId);
                customer.setCustomerName(rs.getString("customer_name"));
                customer.setEmail(rs.getString("email"));
                customer.setOrders(new ArrayList<>());
                customerMap.put(customerId, customer);
            }
            int orderId = rs.getInt("order_id");
            if (orderId != 0) {
                CustomerOrder.Order order = new CustomerOrder.Order();
                order.setOrderId(orderId);
                order.setOrderDate(rs.getDate("order_date"));
                order.setOrderAmount(rs.getBigDecimal("order_amt"));
                customer.getOrders().add(order);
            }
        }
    }
}

```

```

    }
    return new ArrayList<>(customerMap.values());
}
}

```

CustomerOrder.java

```
package com.example.demo.model;
```

```
import java.math.BigDecimal;
```

```
import java.util.Date;
```

```
import java.util.List;
```

```
public class CustomerOrder {
```

```
    private int customerId;
```

```
    private String customerName;
```

```
    private String email;
```

```
    private List<Order> orders;
```

```
    public CustomerOrder() {}
```

```
    public CustomerOrder(int customerId, String customerName, String email, List<Order> orders) {
```

```
        this.customerId = customerId;
```

```
        this.customerName = customerName;
```

```
        this.email = email;
```

```
        this.orders = orders;
```

```
    }
```

```
    public int getCustomerId() { return customerId; }
```

```
    public void setCustomerId(int customerId) { this.customerId = customerId; }
```

```
    public String getCustomerName() { return customerName; }
```

```
    public void setCustomerName(String customerName) { this.customerName = customerName; }
```

```
    public String getEmail() { return email; }
```

```
    public void setEmail(String email) { this.email = email; }
```

```
    public List<Order> getOrders() { return orders; }
```

```
    public void setOrders(List<Order> orders) { this.orders = orders; }
```

```
    public static class Order {
```

```
        private int orderId;
```

```
        private Date orderDate;
```

```
        private BigDecimal orderAmount;
```

```
        public Order() {}
```

```
        public Order(int orderId, Date orderDate, BigDecimal orderAmount) {
```

```
            this.orderId = orderId;
```

```

        this.orderDate = orderDate;
        this.orderAmount = orderAmount;
    }
    public int getOrderId() { return orderId; }
    public void setOrderId(int orderId) { this.orderId = orderId; }

    public Date getOrderDate() { return orderDate; }
    public void setOrderDate(Date orderDate) { this.orderDate = orderDate; }

    public BigDecimal getOrderAmount() { return orderAmount; }
    public void setOrderAmount(BigDecimal orderAmount) { this.orderAmount = orderAmount; }
}
}

```

CustomerOrderRepository.java

```

package com.example.demo.repository;

import com.example.demo.extractor.CustomerOrderResultSetExtractor;
import com.example.demo.model.CustomerOrder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class CustomerOrderRepository {
    @Autowired
    private JdbcTemplate jdbcTemplate;

    public List<CustomerOrder> getAllCustomersWithOrders() {
        String sql = "SELECT c.customer_id, c.customer_name, c.email, " +
            "o.order_id, o.order_date, o.order_amt " +
            "FROM customer c " +
            "LEFT JOIN `order` o ON c.customer_id = o.customer_id";
        return jdbcTemplate.query(sql, new CustomerOrderResultSetExtractor());
    }
}

```

CustomerOrderRepository.java

```

package com.example.demo.repository;

import com.example.demo.extractor.CustomerOrderResultSetExtractor;
import com.example.demo.model.CustomerOrder;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class CustomerOrderRepository {
    @Autowired
    private JdbcTemplate jdbcTemplate;

    public List<CustomerOrder> getAllCustomersWithOrders() {
        String sql = "SELECT c.customer_id, c.customer_name, c.email, " +
            "o.order_id, o.order_date, o.order_amt " +
            "FROM customer c " +
            "LEFT JOIN `order` o ON c.customer_id = o.customer_id";

        return jdbcTemplate.query(sql, new CustomerOrderResultSetExtractor());
    }
}

```

Application.properties

```

spring.application.name=demo
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/company
spring.datasource.username=root

```

Output:

```

1  [
2      {
3          "customerId": 0,
4          "customerName": "Gaurav Bhuravane",
5          "email": "bhuravane@gmail.com",
6          "orders": []
7      },
8      {
9          "customerId": 1,
10         "customerName": "Pritesh Bhuravane",
11         "email": "pritesh@gmail.com",
12         "orders": [
13             {
14                 "orderId": 3,
15                 "orderDate": "2024-11-30",
16                 "orderAmount": 500.75
17             }
18         ]
19     }
20 ]

```

3 Create a maven-based project to demonstrate RESTful Web Services (without database) with spring boot. Consider the main class of the practical number 5.

Person.java

```
package com.example.PersonCRUD;
```

```
public class Person {
    int id;
    String name;
    int age;
    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 int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "Person [id=" + id + ", name=" + name + ", age=" + age + "]";
    }
    public Person(int id, String name, int age) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
    }
}
```

```
}
```

PersonController.java

```

package com.example.PersonCRUD;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PersonController {
    @Autowired
    PersonDAO service;

    @GetMapping("/persons")
    public List<Person> displayAllPerson()
    {
        return service.displayAll();
    }
    @PostMapping("/person")
    public String createPerson(@RequestBody Person per)
    {
        Person newPerson=service.savePerson(per);
        return "New Person Data added with id= "+newPerson.getId();
    }
    @GetMapping("/persons/{pid}")
    public ResponseEntity<Object> displayAPersonInfo(@PathVariable int pid)
    {
        Person per=service.findPerson(pid);
        if (per!=null)
        {
            return ResponseEntity.ok("Person FOund Succesfully"+per);
        }
        else {

```

```

        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Person not
saved");
    }
}
@PostMapping("/updateperson")
public String updatePerson(@RequestBody Person per)
{
    Person updatePerson=service.updatePerson(per);
    return "Person Data Update with id= "+updatePerson.getId();
}
@DeleteMapping("/person/{pid}")
public ResponseEntity<String> deletePersonById(@PathVariable int pid) {
    Person deletedPerson = service.deletePersonById(pid);
    if (deletedPerson != null) {
        return ResponseEntity.ok("Person with ID " + pid + " deleted successfully.");
    } else {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Person with ID " + pid + "
not found.");
    }
}
}

```

PersonDAO.java

```

package com.example.PersonCRUD;

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

import org.springframework.stereotype.Component;
@Component
public class PersonDAO {
    private static List<Person> persons=new ArrayList<Person>();
    static {
        persons.add(new Person(1, "Pritesh", 22));
        persons.add(new Person(2, "Bhupesh", 25));
        persons.add(new Person(3, "Sahil ", 24));
        persons.add(new Person(4, "Suchit", 28));
    }
    public List<Person> displayAll()
    {
        return persons;
    }
    public Person savePerson(Person personParam)

```



```

        {
            persons.add(personParam);
            return personParam;
        }
    public Person findPerson(int pid)
    {
        for(Person per:persons)
        {
            if (per.getId()==pid) {
                return per;
            }
        }
        return null;
    }
    public Person updatePerson(Person personParam)
    {
        int pid=personParam.getId();
        int index=0;
        for(Person per:persons)
        {
            if (per.getId()==pid)
                return persons.set(index, personParam);
            index++;
        }
        return null;
    }
    public Person deletePersonById(int pid) {
        for (Person per : persons) {
            if (per.getId() == pid) {
                persons.remove(per);
                return per;
            }
        }
        return null;
    }
}

```

PersonCrudApplication.java

```

package com.example.PersonCRUD;

import org.springframework.boot.SpringApplication;

```

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

@SpringBootApplication
public class PersonCrudApplication {

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

Output:



