ASSIGNMENT-6

Question 1:

Problem Statement: Developing a Simple Spring Application to Manage Customers

Scenario

You have been hired as a software developer at a customer management company. Your task is

to develop a simple application to manage customer information. The application should be able

to add new customers, retrieve details of a customer, and list all customers in the system. The

goal is to use Spring Core concepts to build this application efficiently.

Objectives

1. Define and Configure Spring Beans:

o Create a Customer class to represent the customer entity.

o Create a CustomerService class to manage customer-related operations.

o Use Spring's XML configuration to define and manage beans.

2. Application Context:

o Use Spring's ApplicationContext to load bean definitions and manage beans.

o Configure the application context using an XML configuration file.

3. Dependency Injection:

o Use setter injection to manage dependencies between beans.

4. Bean Scope:

o Understand and apply the singleton scope to ensure that the CustomerService bean

is shared across the application.

Detailed Requirements

1. Customer Class:

o Fields: id, name, email, and phone.

o Methods: Getters and setters for each field.

2. CustomerService Class:

o Manage a collection of customers.

o Methods:

▪ addCustomer(Customer customer): Adds a new customer to the system.

▪ getCustomerById(int id): Retrieves a customer by their ID.

▪ getAllCustomers(): Returns a list of all customers in the system.

3. Spring Configuration:

o Create an XML configuration file (applicationContext.xml) to define the Customer

and CustomerService beans.

o Use singleton scope for the CustomerService bean.

4. Main Application:

o Load the Spring application context from the XML configuration file.

o Use the CustomerService bean to add, retrieve, and list customers.

Steps to Follow

1. Define the Customer and CustomerService classes with appropriate fields and methods.

2. Create the Spring XML configuration file (applicationContext.xml) with bean definitions.

3. Load the Spring application context in the main application using

ClassPathXmlApplicationContext.

4. Use the CustomerService bean to manage customer information by adding, retrieving, and listing

customers.

SOLUTION

Customer.java

```java
package com.example;

	public class Customer {
	    private int id;
	    private String name;
	    private String email;
	    private String phone;

	    public Customer(int id, String name, String email, String phone) {
	        this.id = id;
	        this.name = name;
	        this.email = email;
	        this.phone = phone;
	    }


	    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;
	    }

	    public String getPhone() {
	        return phone;
	    }

	    public void setPhone(String phone) {
	        this.phone = phone;
	    }
	}
```

CustomerService.java

```java
package com.example;
```

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

public class CustomerService {
    private List<Customer> customerList = new ArrayList<>();

    public void addCustomer(Customer customer) {
        customerList.add(customer);
    }

    public Customer getCustomerById(int id) {
        for (Customer customer : customerList) {
            if (customer.getId() == id) {
                return customer;
            }
        }
        return null;  // Return null if customer with given id is not found
    }

    public List<Customer> getAllCustomers() {
        return new ArrayList<>(customerList);
    }
}
```

CustomerApp.java

```java
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

public class CustomerApp {
    public static void main(String[] args) {
        // Load Spring application context
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Retrieve CustomerService bean
        CustomerService customerService = (CustomerService)
context.getBean("customerService");

        // Add new customers
        Customer customer1 = new Customer(1, "surya", "surya@example.com",
"1234567890");
        Customer customer2 = new Customer(2, "perumal",
"perumal@example.com", "9876543210");

        customerService.addCustomer(customer1);
        customerService.addCustomer(customer2);

        // Retrieve and print customer details
        Customer retrievedCustomer = customerService.getCustomerById(1);
        if (retrievedCustomer != null) {
```

```
                    System.out.println("Retrieved Customer: " +
retrievedCustomer.getName() + " (" +
                            retrievedCustomer.getEmail() + ")");
                } else {
                    System.out.println("Customer not found.");
                }

                // List all customers in the system
                List<Customer> allCustomers = customerService.getAllCustomers();
                System.out.println("All Customers:");
                for (Customer customer : allCustomers) {
                    System.out.println(customer.getName() + " - " +
customer.getEmail());
                }
            }
        }
```

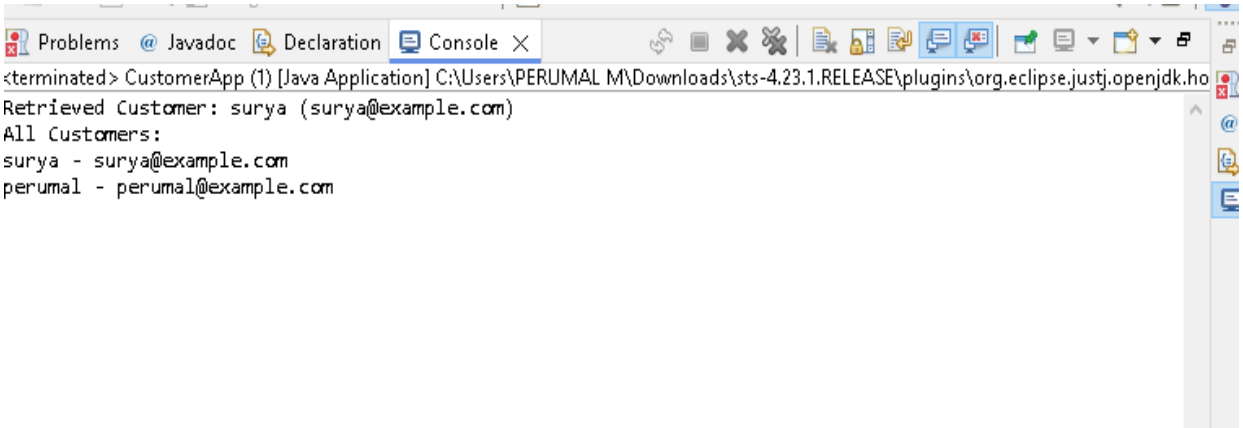
applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Bean definition for CustomerService -->
    <bean id="customerService" class="com.example.CustomerService"
scope="singleton">
        <!-- Inject dependencies (none in this case as it's a simple example) -->
    </bean>

</beans>
```

Output



Question 2:

Scenario

You have been hired as a software developer at a local library. Your task is to develop a simple

application to manage the library's book catalog. The application should be able to add new

books to the catalog, retrieve details of a book, and list all books available in the catalog. The

goal is to use Spring Core concepts to build this application efficiently.

Objectives

1. Define and Configure Spring Beans:

o Create a Book class to represent the book entity.

o Create a BookService class to manage book-related operations.

o Use Spring's XML configuration to define and manage beans.

2. Application Context:

o Use Spring's ApplicationContext to load bean definitions and manage beans.

o Configure the application context using an XML configuration file.

3. Dependency Injection:

o Use setter injection to manage dependencies between beans.

4. Bean Scope:

o Understand and apply the singleton scope to ensure that the BookService bean is

shared across the application.

Detailed Requirements

1. Book Class:

o Fields: title, author, and isbn.

o Methods: Getters and setters for each field.

2. BookService Class:

o Manage a collection of books.

o Methods:

▪ addBook(Book book): Adds a new book to the catalog.

▪ getBookByTitle(String title): Retrieves a book by its title.

▪ getAllBooks(): Returns a list of all books in the catalog.

3. Spring Configuration:

o Create an XML configuration file (applicationContext.xml) to define the Book and

BookService beans.

o Use singleton scope for the BookService bean.

4. Main Application:

o Load the Spring application context from the XML configuration file.

o Use the BookService bean to add, retrieve, and list books

Steps to Follow

1. Define the Book and BookService classes with appropriate fields and methods.

2. Create the Spring XML configuration file (applicationContext.xml) with bean definitions.

3. Load the Spring application context in the main application using

ClassPathXmlApplicationContext.

4. Use the BookService bean to manage the library catalog by adding, retrieving, and listing books.

Solution:

Book.java

```
package com.example;


public class Book {
    private String title;
    private String author;
    private String isbn;

    // Constructors, getters, and setters
    public Book() {}

    public Book(String title, String author, String isbn) {
        this.title = title;
        this.author = author;
        this.isbn = isbn;
    }

    // Getters and setters
    public String getTitle() {
```

```java
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }
}
```

BookService.java

```java
package com.example;


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

public class BookService {
    private List<Book> bookCatalog = new ArrayList<>();

    public void addBook(Book book) {
        bookCatalog.add(book);
    }

    public Book getBookByTitle(String title) {
        for (Book book : bookCatalog) {
            if (book.getTitle().equals(title)) {
                return book;
            }
        }
        return null;  // Return null if book with given title is not found
    }

    public List<Book> getAllBooks() {
        return new ArrayList<>(bookCatalog);
    }
}
```

LibraryApp.java

```java
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

public class LibraryApp {
    public static void main(String[] args) {
        // Load Spring application context
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        // Retrieve BookService bean
        BookService bookService = (BookService) context.getBean("bookService");

        // Add new books
        Book book1 = new Book("Thirukural", "Thiruvalluvar", "978-0134685991");
        Book book2 = new Book("Ramayanam", "kambar", "978-0132350884");

        bookService.addBook(book1);
        bookService.addBook(book2);

        // Retrieve and print book details
        Book retrievedBook = bookService.getBookByTitle("Ramayanam");
        if (retrievedBook != null) {
            System.out.println("Retrieved Book: " + retrievedBook.getTitle() + "
by " +
                    retrievedBook.getAuthor() + " (ISBN: " +
retrievedBook.getIsbn() + ")");
        } else {
            System.out.println("Book not found.");
        }

        // List all books in the catalog
        List<Book> allBooks = bookService.getAllBooks();
        System.out.println("All Books:");
        for (Book book : allBooks) {
            System.out.println(book.getTitle() + " by " + book.getAuthor() + "
(ISBN: " + book.getIsbn() + ")");
        }
    }
}
```

applicationContext.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Bean definition for BookService -->
    <bean id="bookService" class="com.example.BookService" scope="singleton">
```

```
        <!-- Inject dependencies (none in this case as it's a simple example) -->
    </bean>

</beans>
```

OUTPUT