⬜ ⬜ Case Study 1: XML-Based Configuration

📘 **Case Study Title: Hospital Management System**

🧩 **Scenario:** A hospital wants a simple system to manage patient information, appointments, and billing. You need to implement these features using Spring's XML-based configuration.

🏥 **POJO Classes:**

1. Patient.java
   - registerPatient(): Register a new patient
   - getPatientDetails(): View details
2. Appointment.java
   - bookAppointment(): Book appointment
   - cancelAppointment(): Cancel it
3. Billing.java
   - generateBill(): Generate invoice
   - sendBill(): Email invoice

⬜ ⬜ Key Learning:

- Use of XML to wire beans.
- applicationContext.xml manages object creation and dependencies.
- Beans injected using and tags.

pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                    https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example.hospital</groupId>
  <artifactId>hospital-management-xml</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
    <!-- Spring Context for XML config -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.32</version>
    </dependency>
  </dependencies>

</project>
```

src/main/resource---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">

  <!-- Define Patient bean -->
  <bean id="patient" class="com.example.hospital.Patient" />

  <!-- Define Appointment bean -->
  <bean id="appointment" class="com.example.hospital.Appointment" />

  <!-- Define Billing bean -->
  <bean id="billing" class="com.example.hospital.Billing" />
```

```
  <!-- Define HospitalService bean and inject dependencies -->
  <bean id="hospitalService" class="com.example.hospital.HospitalService">
    <property name="patient" ref="patient"/>
    <property name="appointment" ref="appointment"/>
    <property name="billing" ref="billing"/>
  </bean>

</beans>
```

src/main/java:

Patient.java:

```
package com.example.hospital;

public class Patient {
  public void registerPatient() {
    System.out.println("Patient registered successfully.");
  }

  public void getPatientDetails() {
    System.out.println("Patient details.");
  }
}
```

Appointment.java:

```
package com.example.hospital;

public class Appointment {
  public void bookAppointment() {
    System.out.println("Appointment Booked.");
```

```
    }

    public void cancelAppointment() {
        System.out.println("Appointment Cancelled.");
    }
}
```

Billing.java:

```
package com.example.hospital;

public class Billing {

    public void generateBill() {
        System.out.println("Bill generated.");

    }

    public void sendBill() {
        System.out.println("Bill sent via email.");
        // TODO Auto-generated method stub

    }
}
```

HospitalService.java:

```
package com.example.hospital;

public class HospitalService {
    private Patient patient;
```

```
    private Appointment appointment;

    private Billing billing;



    public void setPatient(Patient patient) {

        this.patient = patient;

    }

    public void setAppointment(Appointment appointment) {

        this.appointment = appointment;

    }

    public void setBilling(Billing billing) {

        this.billing = billing;

    }

    public void manageHospital() {

        patient.registerPatient();

        patient.getPatientDetails();

        appointment.bookAppointment();

        billing.generateBill();

        billing.sendBill();

    }

}
```

MainApp.java:

```
package com.example.hospital;

import org.springframework.context.ApplicationContext;
```

```java
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
  public static void main(String[] args) {
    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
    HospitalService hospitalService = (HospitalService) context.getBean("hospitalService");
    hospitalService.manageHospital();
  }
}
```

Output:

Patient details.

Appointment Booked.

Bill generated.

Bill sent via email.

🔹 Case Study 2: Java-Based Configuration

📘 Case Study Title: E-Commerce Order Processing

🧩 **Scenario:** An e-commerce application handles product orders, payments, and inventory. We implement the service using Spring's Java configuration (@Configuration, @Bean

📦 POJO Classes:

1. Product.java
   - addProduct(), listProducts()
2. Order.java
   - createOrder(), cancelOrder()
3. Payment.java
   - processPayment(), refundPayment()

🧠 Key Learning:

- Uses @Configuration and @Bean to define dependencies.
- No need for XML.
- AnnotationConfigApplicationContext is used instead of ClassPathXmlApplicationContext.

pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                  https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.example.ecommerce</groupId>
 <artifactId>ecommerce-java-config</artifactId>
 <version>0.0.1-SNAPSHOT</version>

 <dependencies>
  <!-- Spring Context -->
  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>5.3.30</version>
  </dependency>
 </dependencies>
</project>
```

src/main/java:

Product.java:

```
package com.example.ecommerce;
```

```java
public class Product {
    public void addProduct() {
        System.out.println("Product added to inventory.");
    }

    public void listProducts() {
        System.out.println("Listing all products.");
    }
}
```

Order.java:

```java
package com.example.ecommerce;

public class Order {
    public void createOrder() {
        System.out.println("Order has been created.");
    }

    public void cancelOrder() {
        System.out.println("Order has been cancelled.");
    }
}
```

Payment.java:

```java
package com.example.ecommerce;

public class Payment {
    public void processPayment() {
```

```
        System.out.println("Payment processed successfully.");
    }

    public void refundPayment() {
        System.out.println("Payment has been refunded.");
    }
}
```

EcommerceService.java:

```
package com.example.ecommerce;

public class EcommerceService {
    private Product product;
    private Order order;
    private Payment payment;

    public EcommerceService(Product product, Order order, Payment payment) {
        this.product = product;
        this.order = order;
        this.payment = payment;
    }

    public void runEcommerceFlow() {
        product.addProduct();
        product.listProducts();
        order.createOrder();
        payment.processPayment();
    }
}
```

AppConfig.java

```java
package com.example.ecommerce;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

  @Bean

  public Product product() {

    return new Product();

  }

  @Bean

  public Order order() {

    return new Order();

  }

  @Bean

  public Payment payment() {

    return new Payment();

  }

  @Bean

  public EcommerceService ecommerceService() {

    return new EcommerceService(product(), order(), payment());

  }

}
```

MainApp.java:

```java
package com.example.ecommerce;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

  public static void main(String[] args) {

    ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

    EcommerceService service = context.getBean(EcommerceService.class);

    service.runEcommerceFlow();

  }

}
```

Output:

Product added to inventory.

Listing all products.

Order has been created.

Payment processed successfully.

🔹 Case Study 3: Annotation-Based Configuration

📘 Case Study Title: Library Management System

🧩 **Scenario:** A small community library wants a system to manage books, members, and loans. You implement this using annotation-based Spring (@Component, @Autowired).

📁 📚 **POJO Classes:**

1. Book.java

- addBook(), searchBook()

2. Member.java
- registerMember(), viewMembers()

3. Loan.java

- issueBook(), returnBook()

🧠 **Key Learning:**

• Use of annotations like @Component, @Autowired, @Service, @Repository.

• Spring automatically wires beans.

• Clean, decoupled structure without XML or manual bean declaration.

src/main/resources

pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"

        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example.library</groupId>

  <artifactId>library-annotation-config</artifactId>

  <version>0.0.1-SNAPSHOT</version>


  <dependencies>

    <dependency>

      <groupId>org.springframework</groupId>

      <artifactId>spring-context</artifactId>

      <version>5.3.36</version> <!-- Or latest stable 5.x -->

    </dependency>

  </dependencies>

</project>
```

src/main/java-

Book.java

```
package com.example.library;


import org.springframework.stereotype.Component;


@Component
public class Book {
    public void addBook() {
        System.out.println("Book added to catalog.");
    }


    public void searchBook() {
        System.out.println("Searching book in catalog.");
    }
}
```

Member.java

```
package com.example.library;


import org.springframework.stereotype.Component;


@Component
public class Member {
    public void registerMember() {
        System.out.println("Member registered.");
    }


    public void viewMembers() {
```

```
        System.out.println("Viewing all registered members.");
    }
}
```

Loan.java

```
package com.example.library;

import org.springframework.stereotype.Component;

@Component
public class Loan {
    public void issueBook() {
        System.out.println("Book issued.");
    }

    public void returnBook() {
        System.out.println("Book returned.");
    }
}
```

LibraryService.java:

```
package com.example.library;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class LibraryService {
```

```
    @Autowired
    private Book book;

    @Autowired
    private Member member;

    @Autowired
    private Loan loan;

    public void manageLibrary() {
        book.addBook();
        book.searchBook();
        member.registerMember();
        member.viewMembers();
        loan.issueBook();
        loan.returnBook();
    }
}
```

MainApp.java:

```
package com.example.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
```

```
@Configuration

@ComponentScan(basePackages = "com.example.library")

public class MainApp {

    public static void main(String[] args) {

        ApplicationContext context = new AnnotationConfigApplicationContext(MainApp.class);

        LibraryService service = context.getBean(LibraryService.class);

        service.manageLibrary();

    }

}
```

Output:

Member registered.

Viewing all registered members.

Book issued.

Book returned.