✅ Case Study 1: Hospital Management System (XML-Based Configuration)

✅ Solution: 📄 **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>hospital-management-xml</artifactId>
  <version>1.0</version>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.33</version>
    </dependency>
  </dependencies>
</project>
```

📄 **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
    https://www.springframework.org/schema/beans/spring-beans.xsd">

  <bean id="patient" class="com.example.hospital.Patient"/>
  <bean id="appointment" class="com.example.hospital.Appointment"/>
  <bean id="billing" class="com.example.hospital.Billing"/>

  <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>
```

📄 **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 retrieved.");
  }
}
```

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 to patient email.");
    }
}
```

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();
        appointment.bookAppointment();
        billing.generateBill();
    }
}
```

✅ Case Study 2: E-Commerce Order Processing (Java-Based Configuration)

✅ Solution: 📄 **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId> ecommerce-java-config</artifactId>
  <version>1.0</version>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.33</version>
    </dependency>
  </dependencies>
</project>
```

📄 **Product.java**

```
package com.example.ecommerce;
public class Product {
  public void addProduct() {
    System.out.println("Product added.");
  }
  public void listProducts() {
    System.out.println("Listing products.");
  }
}
```

📄 **Order.java**

```
package com.example.ecommerce;
public class Order {
  public void createOrder() {
    System.out.println("Order created.");
  }
  public void cancelOrder() {
    System.out.println("Order cancelled.");
  }
}
```

📄 **Payment.java**

```
package com.example.ecommerce;
public class Payment {
  public void processPayment() {
    System.out.println("Payment processed.");
  }

  public void refundPayment() {
```

```
        System.out.println("Payment refunded.");
    }
}
```

EcommerceService.java

```
package com.example.ecommerce;
public class EcommerceService {
    private final Product product;
    private final Order order;
    private final Payment payment;

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

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

AppConfig.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());
    }
}
```

✅ Case Study 3: Library Management System (Annotation-Based Configuration)

✅ Solution: 📄 **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
         http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>library-annotation-config</artifactId>
    <version>1.0</version>

    <dependencies>
        <!-- Spring Core & Context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.33</version>
        </dependency>

        <!-- Optional: For Java 8+ compatibility -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>5.3.33</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.10.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
```

```
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

Book.java

```
package com.example.library;
import org.springframework.stereotype.Component;
@Component
public class Book {
  public void addBook() {
    System.out.println("Book added to library.");
  }

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

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 members.");
  }
}
```

Loan.java

```
package com.example.library;
import org.springframework.stereotype.Component;
@Component
public class Loan {
  public void issueBook() {
    System.out.println("Book issued to member.");
  }

  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.Component;
@Component
public class LibraryService {

    @Autowired
    private Book book;

    @Autowired
    private Member member;

    @Autowired
    private Loan loan;

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

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("com.example.library")
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext(MainApp.class);
        LibraryService libraryService = context.getBean(LibraryService.class);
        libraryService.manageLibrary();
    }
}
```