Spring Framework

Container And wiring

Overview

Topics Covered:

- Wiring in Spring
- Managing Dependencies in Code

Wiring in Spring

- Wiring refers to configuring object dependencies.
- XML configuration is often called “wiring”.

Example: Wiring a Service and DAO:

```
<bean id="invoiceDao" class="se.yrgo.dao.JdbcInvoiceDao"/>

<bean id="invoiceService" class="se.yrgo.service.InvoiceServiceImpl">
    <property name="invoiceDao" ref="invoiceDao"/>
</bean>
```

Here, InvoiceService depends on InvoiceDao.

Spring injects invoiceDao into invoiceService.

What is component

Component is a class that contains business logics.

A Service interface for example is a component.

Components are configured in Spring Container, making them accessible using getBean().

Managing Dependencies in Code

Spring helps manage dependencies only where required, not in every class.

Where NOT to Use Spring for Dependency Management:

- Domain Classes (e.g., Book & Author) → These relationships are dynamic and managed at runtime.

Where to Use Spring for Dependency Management:

- Service Layer
- Data Access Layer
- Database Connection Pools

Spring manages configurations that change due to architecture changes, not runtime behavior.

Why Centralized Configuration Matters

- Centralizing configuration ensures flexibility.
- Example: If we switch from JDBC to Hibernate, only one change is needed.

Configuration Before Using Spring:

```
InvoiceService service = new InvoiceServiceImpl();

service.setInvoiceDao(new JdbcInvoiceDao());
```

This requires manual modifications when changing implementations.

Why Centralized Configuration Matters

Configuration Using Spring:

```
ClassPathXmlApplicationContext container = new
ClassPathXmlApplicationContext("application.xml");

InvoiceService service = container.getBean("invoiceService",
InvoiceService.class);
```

With Spring, only the XML configuration changes, keeping the client code intact.

Summary

Feature	Description
Spring Container	Manages object creation and wiring
XML Configuration	Defines beans and their dependencies
Dependency Injection	Injects dependencies dynamically
Centralized Configuration	Allows easy implementation changes
Wiring in Spring	Connects services and DAOs automatically