

Spring Framework

1. Framework 개념

CONTENTS

- 1 SW 재사용 방안들
- 2 디자인 패턴과 프레임워크의 관련성
- 3 프레임워크의 구성요소와 종류

학습 목표

- SW 재사용성을 높일 수 있는 방안에 대해 이해할 수 있습니다
- 디자인패턴과 프레임워크의 관련성에 대해 이해할 수 있습니다
- 프레임워크 구성요소와 종류에 대해 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

1. SW 재사용 방안들

| 복사(Copy) & 붙이기(Paste)

초보적인 재사용 방식으로 비슷한 예제를 다른 Source에서 복사해서 사용함.

```
GregorianCalendar date = (GregorianCalendar)Calendar.getInstance();
SimpleDateFormat df = new SimpleDateFormat("yyyyMMdd");
String date = df.format(date);
```

- 예를 들어, A라는 클래스에서 Date 타입을 String 타입으로 변환하는 코딩을 하고, 클래스 B에서 동일한 로직이 필요하여 복사했다고 가정한 경우

JDK 버전이 바뀌어 동일한 기능을 제공하는 향상된 인터페이스가 나오면 위의 코드를 사용한 A, B 클래스를 모두 변경해야 한다.

| 메서드 호출

자주 사용되고, 유사한 기능들을 모아 메서드로 정의하여 재사용함.

```
public class DateUtility {  
    public static String toStringToday(String format) {  
        GregorianCalendar date = (GregorianCalendar)Calendar.getInstance();  
        SimpleDateFormat df = new SimpleDateFormat("yyyyMMdd");  
        String date = df.format(date);  
    }  
}
```

```
String sdate = DateUtility.toStringToday("yyyMMdd");
```

| 메서드 호출

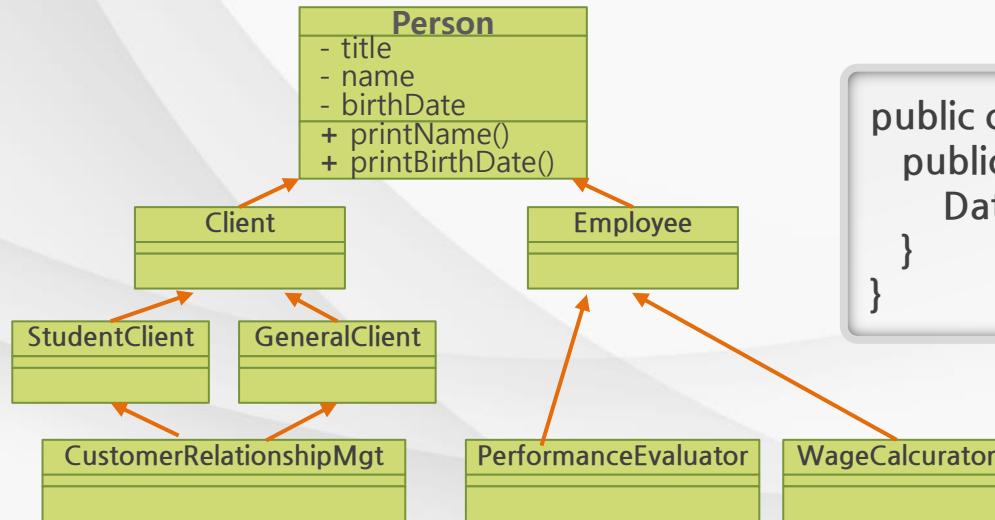
- ◎ JDK 버전이 바뀌거나 메서드의 내용이 수정되더라도 해당 클래스를 모두 수정할 필요 없이 `toStringToday()` 메서드의 내용만 수정하면 된다.

`toStringToday()` 메서드의 Signature를 변경하면 이 메서드를 사용하는 모든 클래스에 영향을 준다.

메서드 재사용 방법은 ‘복사 & 붙이기’보다는 진보된 방식이지만, 작업 영역간의 결합도(Coupling) 문제는 여전히 존재한다.

| 클래스 재사용 (상속)

자주 사용되고, 유사한 기능들을 모아 메서드로 정의하여 재사용함.

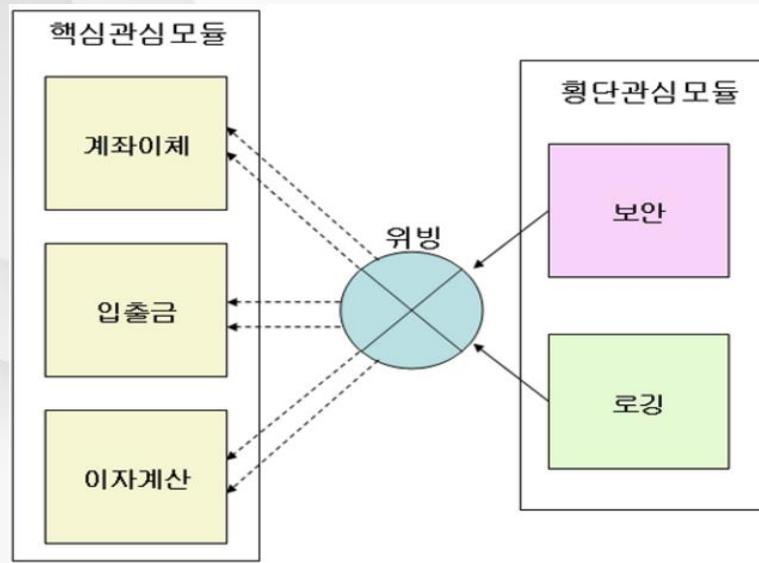


```
public class Person {
    public String printBirthDate(String format) {
        DateUtility.toStringToday(birthDate, format);
    }
}
```

- Person을 상속받은 모든 클래스들은 자동적으로 변경된 printBirthDate() 메서드를 사용하게 된다.
- DateUtility 클래스의 메서드가 변경되더라도 printBirthDate() 메서드의 인터페이스가 변하지 않으면 나머지 클래스들은 영향을 받지 않는다.

I AOP(Aspect Oriented Programming)

❖ 관심의 분리 (Separation of Concerns)



AOP가 핵심관심모듈의 코드를 직접 건드리지 않고 필요한 기능이 동작하도록 하는 데는 위빙(Weaving)이라고 하는 특수한 작업이 필요하다.

즉, AOP에서 위빙 작업을 통해 핵심모듈 사이 사이에 필요한 횡단 관심 코드가 동작하도록 엮어지게 만든다.

- AOP는 OOP를 더욱 OOP 답게 만들어 줄 수 있다.
- AOP는 OOP 뿐만 아니라 기존의 절차적 프로그래밍에도 적용될 수 있다.

A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and filled with out-of-focus, glowing circular lights in shades of yellow, orange, and blue, creating a bokeh effect.

2. 디자인 패턴과 프레임워크의 관련성

| 디자인패턴의 정의

프로그램 개발에서 자주 나타나는 과제를 해결하기 위한 방법 중 하나로, 소프트웨어 개발과정에서 발견된 Know-How를 축적하여 이름을 붙여 이후에 재사용하기 좋은 형태로 특정 규약을 묶어서 정리한 것.

- 이 용어를 소프트웨어 개발 영역에서 구체적으로 처음 제시한 곳은, GoF(Gang of Four)라 불리는 네 명의 컴퓨터 과학 연구자들이 쓴 서적 'Design Patterns: Elements of Reusable Object-Oriented Software'(재사용 가능한 객체지향 소프트웨어의 요소 - 디자인 패턴)이다.

| 디자인패턴의 정의

❖ 디자인 패턴을 사용하는 이유

- 요구사항은 수시로 변경-> 요구사항 변경에 대한 Source Code 변경을 최소화
- 여러 사람이 같이 하는 팀 프로젝트 진행 -> 범용적인 코딩 스타일을 적용
- 상황에 따라 인수 인계하는 경우도 발생 -> 직관적인 코드를 사용

| 프레임워크의 정의

비기능적(Non-Functional) 요구사항(성능, 보안, 확장성, 안정성 등)을 만족하는 구조와 구현된 기능을 안정적으로 실행하도록 제어 해주는 잘 만들어진 구조의 라이브러리의 덩어리

- 프레임워크는 애플리케이션들의 최소한의 공통점을 찾아 하부 구조를 제공함으로써 개발자들로 하여금 시스템의 하부 구조를 구현하는데 들어가는 노력을 절감하게 해줌

| 프레임워크의 정의

❖ 프레임워크를 사용하는 이유

- 비기능적인 요소들을 초기 개발 단계마다 구현해야 하는 불합리함을 극복해준다.
- 기능적인(Functional) 요구사항에 집중할 수 있도록 해준다.
- 디자인 패턴과 마찬가지로 반복적으로 발견되는 문제를 해결하기 위한 특화된 Solution을 제공한다.

디자인패턴과 프레임워크의 관련성

디자인 패턴은 프레임워크의 핵심적인 특징이고, 프레임워크를 사용하는 애플리케이션에 그 패턴이 적용된다는 특징을 가지고 있다. 하지만 프레임워크는 디자인 패턴이 아니다.

- 디자인 패턴은 애플리케이션을 설계할 때 필요한 구조적인 가이드라인이 되어 줄 수는 있지만 구체적으로 구현된 기반코드를 제공하지 않는다.
- 프레임워크는 디자인 패턴과 함께 패턴이 적용 된 기반 클래스 라이브러리를 제공해서 프레임워크를 사용하는 구조적인 틀과 구현코드를 함께 제공한다.

| 디자인패턴과 프레임워크의 관련성



개발자는 프레임워크의 기반코드를 확장하여



사용하면서 자연스럽게 그 프레임워크에서

사용된 패턴을 적용할 수 있게 된다.

A photograph of a person's hands holding a smartphone at night. The phone screen is illuminated, and the background is filled with blurred, colorful lights in shades of yellow, orange, and blue.

3. 프레임워크의 구성요소와 종류

■ IoC (Inversion of Control)

IoC란 “제어의 역전” 즉, 인스턴스 생성부터 소멸까지의 **인스턴스 생명주기 관리를 개발자가 아닌 컨테이너가 대신 해준다**는 뜻임.
즉, 컨테이너 역할을 해주는 프레임워크에게 제어하는 권한을 넘겨서 개발자의 코드가 신경 써야 할 것을 줄이는 전략이다.

■ IoC (Inversion of Control)



- 프레임워크의 동작원리를 제어흐름이 일반적인 프로그램 흐름과 반대로 동작하므로 IoC 라고 설명함.
- Spring 컨테이너는 IoC를 지원하며, 메타데이터(XML설정)를 통해 beans를 관리하고 어플리케이션의 중요부분을 형성함.
- Spring 컨테이너는 관리되는 bean들을 의존성주입(Dependency Injection)을 통해 IoC를 지원함.

■ 클래스 라이브러리 (Class Library)

프레임워크는 특정 부분의 기술적인 구현을 라이브러리 형태로 제공한다.

Class Library라는 구성요소는 프레임워크의 정의 중 하나인 "Semi Complete(반제품)" 이다. 라고 해석하게 만들었다.

특징	프레임워크	라이브러리
유저코드의 작성	프레임워크 클래스를 서브 클래싱 해서 작성	독립적으로 작성
호출흐름	프레임워크코드가 유저코드를 호출	유저코드가 라이브러리를 호출
실행흐름	프레임워크가 제어	유저코드가 제어
객체의 연동	구조프레임워크가 정의	독자적으로 정의

| 클래스 라이브러리 (Class Library)

❖ 라이브러리와 프레임워크의 차이점

- 프레임워크와 라이브러리를 구분하는 방법은 실행제어가 어디서 일어나는가에 달려있다.
- **라이브러리는** 개발자가 만든 클래스에서 직접 호출하여 사용하므로 실행의 흐름에 대한 제어를 개발자의 코드가 관리하고 있다.
- **프레임워크는** 반대로 프레임워크에서 개발자가 만든 클래스를 호출하여 실행의 흐름에 대한 제어를 담당한다.

디자인 패턴



디자인 패턴 + 라이브러리 = 프레임워크



프레임워크는 디자인 패턴과 그것이 적용된 기반 라이브러리의 결합으로 이해할 수 있다.

프레임워크의 라이브러리를 살펴볼 때도 적용된 패턴을 주목해서 살펴 본다면 그 구성을 이해하기 쉽다.

특히 프레임워크를 확장하거나 커스터마이징 할 때는 프레임워크에 적용된 패턴에 대한 이해가 꼭 필요하다.

프레임워크 종류

❖ 아키텍쳐 결정 = 사용하는 프레임워크의 종류 + 사용전략

기능	프레임워크 종류
웹(MVC)	Spring MVC, Struts2, Webwork, PlayFramework
OR(Object-Relational) 매팅	MyBatis, Hibernate, JPA, Spring JDBC
AOP(Aspect Oriented Programming)	Spring AOP, AspectJ, JBoss AOP
DI(Dependency Injection)	Spring DI, Google Guice
Build와 Library 관리	Ant + Ivy, Maven, Gradle
단위 테스트	jUnit, TestNG, Cactus
JavaScript	jQuery, AngularJS, Node.js

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Framework 개념]에 대해서 살펴보았습니다.

SW 재사용 방안들

Copy & Paste, Method, Inheritance, AOP

디자인 패턴과 프레임워크 관련성

개발자는 프레임워크의 기반코드를 확장하여 사용하면서 자연스럽게 프레임워크에서 사용된 패턴들을 적용할 수 있게 된다.

프레임워크의 구성요소와 종류

IoC(Inversion of Control), Design Pattern, Class Library

Spring Framework

2. 환경설정

CONTENTS

1

JDK 8 설치 및 API 문서

2

STS 3.7.3 설치 및 Spring API 문서

3

Tomcat 8 설치

4

Oracle 11g XE 설치

학습 목표

- JDK(Java Development Kit) 1.8 설치 및 API 문서를 찾을 수 있습니다.
- STS(Spring Tool Suite) 3.7.3 설치 및 Spring 3.2.17 API 문서를 찾을 수 있습니다.
- Tomcat 8.0 을 설치할 수 있습니다.
- Oracle 11g XE 를 설치할 수 있습니다.



1. JDK 8 설치 및 API 문서

I Java SE Development Kit 8 downloads

URL

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Google JDK1.8 download

전체 동영상 이미지 뉴스 지도 더보기 ▾ 검색 도구

검색결과 약 627,000개 (0.43초)

관련검색: jdk1 8 다운로드 jdk1 8.0 download jdk1 8.0 _25 download jdk1 8.0 _05 download jdk1 6 download

Java SE Development Kit 8 - Downloads - Oracle
www.oracle.com/.../downloads/jdk8-downloads-2133151.html ▾ 이 페이지 번역하기
Download JDK 8, a development environment for building applications and components using the Java programming language.
JDK 8 for ARM - Download · Oracle Binary Code License · Java Magazine

I Java SE Development Kit 8 downloads

URL

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

The screenshot shows the Oracle Java SE Development Kit 8 Downloads page. On the left, there is a sidebar with links to Java SE, Java EE, Java ME, Java SE Support, Java SE Advanced & Suite, Java Embedded, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, and Java Magazine. The main content area has a navigation bar with tabs: Overview, Downloads (which is highlighted with a red box), Documentation, Community, Technologies, and Training. Below the navigation bar, the title "Java SE Development Kit 8 Downloads" is displayed in bold. A thank you message follows, stating: "Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language." Another message below explains: "The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform." A "See also:" section lists several items:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

At the bottom, there are two links: "JDK 8u91 Checksum" and "JDK 8u92 Checksum".

I Java SE Development Kit 8 downloads

URL

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Java SE Development Kit 8u91

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.72 MB	jdk-8u91-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.69 MB	jdk-8u91-linux-arm64-vfp-hflt.tar.gz
Linux x86	154.74 MB	jdk-8u91-linux-i386.tar.gz
Linux x86	174.92 MB	jdk-8u91-linux-i586.tar.gz
Linux x64	152.74 MB	jdk-8u91-linux-x64.tar.gz
Linux x64	172.97 MB	jdk-8u91-linux-x64-v2.tar.gz
Mac OS X	227.29 MB	jdk-8u91-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.59 MB	jdk-8u91-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.95 MB	jdk-8u91-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.29 MB	jdk-8u91-solaris-x64.tar.Z
Solaris x64	96.78 MB	jdk-8u91-solaris-x64.tar.gz
Windows x86	182.29 MB	jdk-8u91-windows-i586.exe
Windows x64	187.4 MB	jdk-8u91-windows-x64.exe

Windows 64bit 버전 선택 : jdk-8u91-windows-x64.exe

I Java SE Development Kit 8 API 문서

URL

<https://docs.oracle.com/javase/8/docs/api/>

Google

JDK 8 api doc

전체 동영상 뉴스 지도 이미지 더보기 ▾ 검색 도구

검색 결과 약 511,000개 (0.51초)

관련검색: java api doc 한글 jdk documentation api java api doc java api doc download java 7 api doc

Java 8 API - Oracle
https://docs.oracle.com/javase/8/docs/api ▾ 이 페이지 번역하기
Provides reference-object classes, which support a limited degree of interaction A package of the Java Image I/O API dealing with synchronous notification of ...

Overview

Java.util - Java.lang - Classes -
Java.io - Prev Letter - Compact1

Platform Standard Edition 8

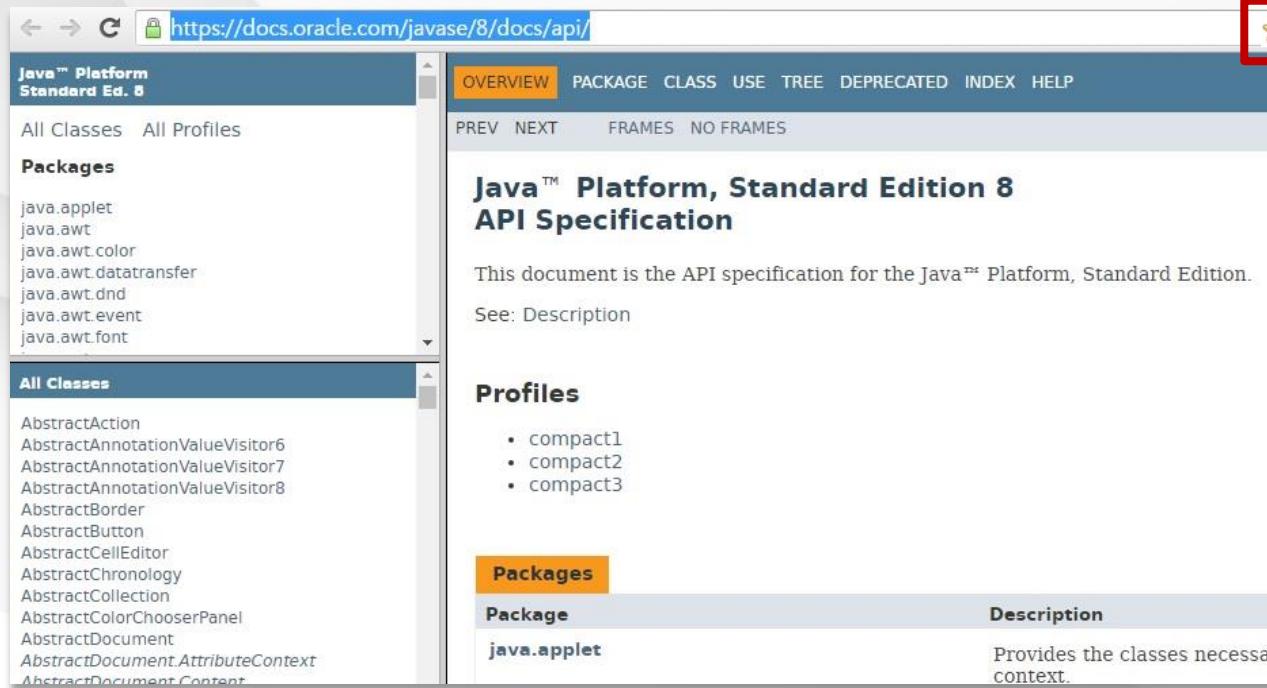
Java(tm) Platform, Standard Edition
8. API仕様 ... このパッケ ...

oracle.com 검색결과 더보기 »

I Java SE Development Kit 8 API 문서

URL

<https://docs.oracle.com/javase/8/docs/api/>



The screenshot shows the Java Platform, Standard Edition 8 API Specification page. The URL in the address bar is <https://docs.oracle.com/javase/8/docs/api/>. The page title is "Java™ Platform, Standard Edition 8 API Specification". The left sidebar lists "All Classes" and "Packages" (including java.applet, java.awt, etc.). The main content area includes sections for "Profiles" (compact1, compact2, compact3) and "Packages" (java.applet). A red box highlights the star icon in the top right corner of the browser window.

Browser의
즐겨찾기에 추가함

A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, showing blurred lights in various colors (yellow, orange, blue) that suggest a city at night or a well-lit indoor space.

2. STS 3.7.3 설치 및 Spring API 문서

I STS downloads

URL

<https://spring.io/tools/sts>

The screenshot shows a Google search results page for the query "sts download". The search bar at the top contains "sts download". Below the search bar, there are filters for "전체" (All), "동영상" (Videos), "이미지" (Images), "뉴스" (News), "지도" (Maps), and "더보기 ▾" (More). The search results indicate approximately 2,240,000 results found in 0.31 seconds. A red box highlights the first search result, which is the official Spring Tool Suite (STS) page.

검색결과 약 2,240,000개 (0.31초)

관련검색: sts download springsource sts download 64 bit spring sts download eclipse sts download sts 3.6 2 download

Spring Tool Suite™ (STS)

<https://spring.io/tools/sts> ▾ 이 페이지 번역하기

The Spring Tool Suite is an Eclipse-based development environment that is customized for developing Spring applications. It provides a ready-to-use ...

Versions

Spring Tool Suite™ Downloads. Use one of the links below to ...

Previous Spring Tool Suite ...

Previous Spring Tool Suite™ Downloads. Use one of the ...

I STS downloads

URL

<https://spring.io/tools/sts>

TOOLS

Spring Tool Suite™

The Spring Tool Suite is an Eclipse-based development environment customized for developing Spring applications. It provides a graphical user interface to implement, debug, run, and deploy Spring applications. It includes comprehensive tooling for Spring, including support for Spring Framework, Spring Boot, Spring Cloud, and Spring Integration. It also includes integrations for Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ, and many other tools. The Spring Tool Suite is built on top of the latest Eclipse releases.

Included with the Spring Tool Suite is the developer edition of Pivotal tc Server, the drop-in replacement for Apache Tomcat that's optimized for Spring. With its Spring Insight console, tc Server Developer Edition provides a graphical real-time view of application performance metrics that lets developers identify and diagnose problems from their desktops.

The Spring Tool suite supports application targeting to local, virtual and cloud-based

3.7.3 Release for Windows 선택 :
spring-tool-suite-3.7.3.RELEASE-e4.5.2-win32-x86_64.zip



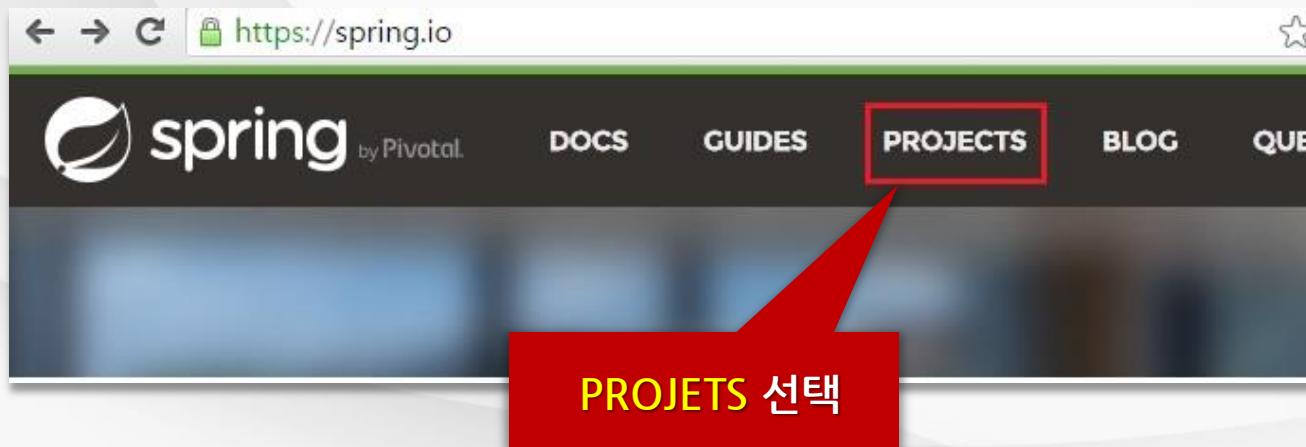
**DOWNLOAD STS
(3.7.3.RELEASE for Windows)**

[See All Versions](#)

| SpringFramework 3.2.17 release API 문서

URL

<http://spring.io>



| SpringFramework 3.2.17 release API 문서

URL

<http://spring.io>

The screenshot shows the Spring.io homepage with three main sections:

- SPRING IO PLATFORM**: Provides a cohesive, versioned platform for building modern applications. It is a modular, enterprise-grade distribution that delivers a curated set of dependencies.
- SPRING BOOT**: Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.
- SPRING FRAMEWORK**: Provides core support for dependency injection, transaction management, web apps, data access, messaging and more.

A large red callout box at the bottom points to the **SPRING FRAMEWORK** section, with the text "SPRING FRAMEWORK 선택" (Select Spring Framework) written in yellow.

| SpringFramework 3.2.17 release API 문서

URL

<http://docs.spring.io/spring/docs/3.2.17.RELEASE/javadoc-api/>

projects.spring.io/spring-framework/

Introduction

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Features

- Dependency Injection
- Aspect-Oriented Programming including Spring's declarative transaction management
- Spring MVC web application and RESTful web service framework
- Foundational support for JDBC, JPA, JMS
- Much more...

Spring Framework	
RELEASE	DOCUMENTATION
4.3.1 SNAPSHOT	Reference API
4.3.0 CURRENT	Reference API
4.2.7 SNAPSHOT	Reference API
4.2.6	Reference API
4.1.9	Reference API

3.2.17 API 선택

SpringFramework 3.2.17 release API 문서

URL

<http://docs.spring.io/spring/docs/3.2.17.RELEASE/javadoc-api/>

The screenshot shows the Spring Framework 3.2.17.RELEASE API documentation. The URL in the browser address bar is <http://docs.spring.io/spring/docs/3.2.17.RELEASE/javadoc-api/>. The page title is "Spring Framework 3.2.17.RELEASE API". The left sidebar has sections for "Spring Framework", "All Classes", and "Packages". The main content area displays the API documentation for the Spring AOP package. A red callout box with the text "Browser의 즐겨찾기에 추가함" (Add to bookmarks) points to the yellow star icon in the top right corner of the browser window.

Spring Framework 3.2.17.RELEASE API

This is the public API documentation for the Spring Framework.

See: Description

Packages	Package	Description
org.springframework.aop	org.springframework.aop	Core Spring AOP interfaces, built on AOP Alliance AOP interopera
org.springframework.aop.aspectj	org.springframework.aop.aspectj	AspectJ integration package.
org.springframework.aop.aspectj.annotation	org.springframework.aop.aspectj.annotation	Classes enabling AspectJ 5 @Annotated classes to be used in Sp
org.springframework.aop.aspectj.autoproxy	org.springframework.aop.aspectj.autoproxy	Base classes enabling auto-proxying based on AspectJ.
org.springframework.aop.config	org.springframework.aop.config	Support package for declarative AOP configuration, with XML sche

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

3. Tomcat 8 설치

| Tomcat8.0 downloads

Site

<https://tomcat.apache.org/download-80.cgi>

The screenshot shows a Google search results page for the query "tomcat 8.0 download". The search bar at the top contains the query. Below it, there are tabs for "전체" (All), "동영상" (Videos), "이미지" (Images), "뉴스" (News), "더보기 ▾" (More), and "검색 도구" (Search tools). The search results section displays the number of results: "검색결과 약 273,000개 (0.41초)". Below this, a list of related searches is shown: "관련검색: tomcat 8.0 설치", "tomcat 8.0 설정", "tomcat 8.0 eclipse plugin", "tomcat 8.0 eclipse", and "apache tomcat 8 download". A red rectangular box highlights the first result in the list:

Apache Tomcat® - Apache Tomcat 8 Software Downloads
<https://tomcat.apache.org/download-80.cgi> ▾ 이 페이지 번역하기
Welcome to the Apache Tomcat® 8.x software download page. This page provides download links for obtaining the latest versions of Tomcat 8.x software, ...

| Tomcat8.0 downloads

Site

<https://tomcat.apache.org/download-80.cgi>

The screenshot shows the Apache Tomcat download page. At the top left is the Tomcat logo (a yellow cat). To its right is the text "Apache Tomcat®". Below the logo is a navigation menu with links: Home, Taglibs, Maven Plugin, and a "Download" section containing links for Tomcat 9, 8, 7, 6, Connectors, Native, and Taglibs. In the center, there's a large red callout box with the text "8.0.36 Zip 버전 선택 : apache-tomcat-8.0.36.zip". A red arrow points from this box to the "8.0.36" link in the "Quick Navigation" menu below. The "Quick Navigation" menu also includes links for KEYS, 8.5.3, Browse, and Archives. At the bottom, there's a "Release Integrity" section with text about verifying file integrity using OpenPGP.

Apache Tomcat

Home
Taglibs
Maven Plugin

Download

Which version?
Tomcat 9
Tomcat 8
Tomcat 7
Tomcat 6
Tomcat Connectors
Tomcat Native
Taglibs

Apache Tomcat®

Tomcat 8 Software Download

Welcome to the Apache Tomcat® software download page. This page provides

Quick Navigation

KEYS | **8.0.36** | 8.5.3 | Browse | Archives

Release Integrity

You **must** [verify](#) the integrity of the downloaded files. We provide OpenPGP

Tomcat's Release Managers. We also provide MD5 and SHA-1 checksums for

ours.

| Tomcat8.0 downloads

Site

<https://tomcat.apache.org/download-80.cgi>

Tomcat 8.0

- Tomcat Connectors
- Tomcat Native
- Wiki
- Migration Guide
- Presentations

Problems?

- Security Reports
- Find help
- FAQ
- Mailing Lists
- Bug Database
- IRC

8.0.36

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
 - [zip \(pgp, md5, sha1\)](#)
 - [tar.gz \(pgp, md5, sha1\)](#)
 - [32-bit Windows zip \(pgp, md5, sha1\)](#)
 - [64-bit Windows zip \(pgp, md5, sha1\)](#)
 - [32-bit/64-bit Windows Service Installer \(pgp, md5, sha1\)](#)

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

4. Oracle 11g XE 설치

4. Oracle 11g XE(eXpress Edition) 설치

Tacademy

I Oracle 11g XE downloads

Site

<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

The screenshot shows a Google search results page. The search query is "oracle express edition download". The results include a link to the Oracle Database Express Edition 11g Release 2 Downloads page, which is highlighted with a red border.

Google search results for "oracle express edition download":

- 관련검색: 오라클 express edition 오라클 11g express edition 오라클 11g express edition 설치 오라클 10g express edition oracle 10g xe 다운로드
- Oracle Database Express Edition 11g Release 2 Downloads**
www.oracle.com/technetwork/.../[express-edition/downloads/](#) 이 페이지 번역하기
Unzip the download and run the DISK1/setup.exe. Download. Oracle Database Express Edition 11g Release 2 for Linux x64 -Unzip the download and the RPM ...
- Oracle Database 11g Express ...**
Oracle Database 11g Express Edition Documentation. Oracle ...

4. Oracle 11g XE(eXpress Edition) 설치

Tacademy

Oracle 11g XE downloads

Site

<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

The screenshot shows the Oracle Database Technology Index page. At the top, there is a navigation bar with links for Sign In/Register, Help, Country, Communities, I am a..., I want to..., Search, Products, Solutions, Documentation, Training, and Partner. Below the navigation bar, the breadcrumb path is Oracle Technology Network > Database > Database Technology Index. On the left, there is a sidebar with links for Database 12c, Database In-Memory, Multitenant, Options, Application Development, Big Data Appliance, Data Warehousing & Big Data, Database Appliance, Database Cloud, Exadata Database Machine, High Availability, Manageability, Migrations, and Security. The main content area has tabs for Overview, Downloads, Documentation, Community, and Learn More. The 'Downloads' tab is highlighted with a red box and a yellow callout bubble containing the Korean text 'Downloads 탭 선택'. Below the tabs, the section title is 'Oracle Database 11g Express Edition' and the subtitle is 'Free to develop, deploy, and distribute'. A brief description follows: 'Oracle Database 11g Express Edition (Oracle Database XE) is an entry-level, small-footprint database based on the Oracle Database 11g Release 2 code base. It's free to develop, deploy, and distribute; fast to download; and simple to administer.' At the bottom, there is a button labeled 'Get started with Oracle Database Express Edition 11g Release 2' and links for Downloads, Demonstrations, Online Documentation, and Express Edition Forum.

I Oracle 11g XE downloads

Site

<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

The screenshot shows the Oracle Technology Network (OTN) website. The top navigation bar includes links for Sign In/Register, Help, Country, Communities, I am a..., I want to..., Search, Products, Solutions, Downloads, Store, Support, Training, and Partner. The breadcrumb path is Oracle Technology Network > Database > Database Technology Index > Database Express Edition > Downloads. Below this, there are tabs for Overview, Downloads (which is selected), Documentation, Community, and Learn More. The main content area features a heading for "Oracle Database Express Edition 11g Release 2" dated June 4, 2014. A note states: "You must accept the OTN License Agreement for Oracle Database Express Edition 11g Release 2 to download this software." Two radio buttons are shown: "Accept License Agreement" (selected) and "Decline License Agreement". Below this, there are three download links for different operating systems: Windows x64, Windows x32, and Linux x64.

Sign In/Register Help Country Communities I am a... I want to... Search

Products Solutions Downloads Store Support Training Partner

Oracle Technology Network > Database > Database Technology Index > Database Express Edition > Downloads

Database 12c
Database In-Memory
Multitenant
Options
Application Development
Big Data Appliance
Data Warehousing & Big Data
Database Appliance
Database Cloud
Exadata Database Machine
High Availability
Manageability
Migrations

Overview Downloads Documentation Community Learn More

Oracle Database Express Edition 11g Release 2

June 4, 2014

You must accept the OTN License Agreement for Oracle Database Express Edition 11g Release 2 to download this software.

Accept License Agreement Decline License Agreement

 Oracle Database Express Edition 11g Release 2 for Windows x64
- Unzip the download and run the DISK1/setup.exe

 Oracle Database Express Edition 11g Release 2 for Windows x32
- Unzip the download and run the DISK1/setup.exe

 Oracle Database Express Edition 11g Release 2 for Linux x64
- Unzip the download and the RPM file can be installed as normal

I Oracle 11g XE downloads

Site

<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

The screenshot shows the Oracle Technology Network website. The navigation bar includes links for Sign In/Register, Help, Country, Communities, I am a..., I want to..., Search, Products, Solutions, Downloads, Store, Support, Training, and Partner. The breadcrumb path is Oracle Technology Network > Database > Database Technology Index > Database Express Edition > Downloads. On the left, there's a sidebar with links for Database 12c, Database In-Memory, Multitenant, Options, Application Development, Big Data Appliance, Data Warehousing & Big Data, Database Appliance, Database Cloud, Exadata Database Machine, High Availability, Manageability, and Migrations. The main content area displays the Oracle Database Express Edition page for June 4, 2014. It features a large red callout box with the text "11g Release Windows 64bit 버전 선택 : OracleXE112_Win64.zip". Below this, a note says "Thank you for accepting the License Agreement; you may now download this software." followed by a list of download links for Oracle Database Express Edition 11g Release 2 for Windows x64, x32, and Linux x64.

Sign In/Register Help Country ▾ Communities ▾ I am a... ▾ I want to... ▾ Search

Products Solutions Downloads Store Support Training Partner

Oracle Technology Network > Database > Database Technology Index > Database Express Edition > Downloads

Database 12c
Database In-Memory
Multitenant
Options
Application Development
Big Data Appliance
Data Warehousing & Big Data
Database Appliance
Database Cloud
Exadata Database Machine
High Availability
Manageability
Migrations

Overview Downloads Documentation C

Oracle Database Express Edition

June 4, 2014

11g Release Windows 64bit
버전 선택
: OracleXE112_Win64.zip

Thank you for accepting the License Agreement; you may now download this software.

Oracle Database Express Edition 11g Release 2 for Windows x64

- Unzip the download and run the DISK1/setup.exe

Oracle Database Express Edition 11g Release 2 for Windows x32

- Unzip the download and run the DISK1/setup.exe

Oracle Database Express Edition 11g Release 2 for Linux x64

- Unzip the download and the RPM file can be installed as normal

» Oracle Site
계정이 있어야
다운로드 가능

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [환경설정]에 대해서 살펴보았습니다.

JDK 8 설치 및 API 문서

- JDK(Java Development Kit) 1.8 다운로드 및 설치
- Java SE API 문서 찾기

STS 3.7.3 설치 및 Spring API 문서

- STS(Spring Tool Suite) 3.7.3 다운로드 및 설치
- SpringFramework 3.2.17 Release API 문서 찾기

Tomcat 8 설치

Tomcat 8.0 다운로드 및 설치

Oracle 11g XE 설치

Oracle 11g XE 버전 다운로드 및 설치

Spring Framework

3. Spring Framework 개요

CONTENTS

1

Spring Framework 개요

2

Spring Framework 특징

3

Spring Framework 기능요소

학습 목표

- Spring Framework의 정의와 전략에 대해 이해할 수 있습니다.
- Spring Framework의 특징에 대해 이해할 수 있습니다.
- Spring Framework의 기능요소에 대해 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

1. Spring Framework 개요

| Spring Framework란?

Java 엔터프라이즈 개발을 편하게 해주는 오픈소스 **경량급 애플리케이션 프레임워크**이다.

애플리케이션 프레임워크

- 특정 계층이나 기술, 업무 분야에 국한되지 않고 애플리케이션의 전 영역을 포괄하는 범용적인 프레임워크를 말한다.

경량급 프레임워크

- 단순한 웹컨테이너에서도 엔터프라이즈 개발의 고급기술을 대부분 사용할 수 있다.

| Spring Framework란?

Java 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크이다.

엔터프라이즈 개발 용이

- 개발자가 복잡하고 실수하기 쉬운 Low Level에 많이 신경 쓰지 않으면서 Business Logic 개발에 전념할 수 있도록 해준다.

오픈소스

- Spring은 OpenSource의 장점을 충분히 취하면서 동시에 OpenSource 제품의 단점과 한계를 잘 극복한다.

I Spring Framework 전략

❖ Spring 삼각형

엔터프라이즈 개발의 복잡함을 상대하는 Spring의 전략

→ Portable Service Abstraction, DI, AOP, POJO



I Spring Framework 전략

❖ Spring 삼각형

엔터프라이즈 개발의 복잡함을 상대하는 Spring의 전략

→ Portable Service Abstraction, DI, AOP, POJO



◎ Portable Service Abstraction(서비스 추상화)

트랜잭션 추상화, OXM 추상화, 데이터 액세스의 Exception
변환기능 등 기술적인 복잡함은 추상화를 통해 Low Level의 기술
구현 부분과 기술을 사용하는 인터페이스로 분리한다.

I Spring Framework 전략

❖ Spring 삼각형

엔터프라이즈 개발의 복잡함을 상대하는 Spring의 전략

→ Portable Service Abstraction, DI, AOP, POJO



◎ 객체지향과 DI(Dependency Injection)

Spring은 객체지향에 충실한 설계가 가능하도록 단순한 객체 형태로 개발할 수 있고, DI는 유연하게 확장 가능한 객체를 만들어 두고 그 관계는 외부에서 다이내믹하게 설정해준다.

I Spring Framework 전략

❖ Spring 삼각형

엔터프라이즈 개발의 복잡함을 상대하는 Spring의 전략

→ Portable Service Abstraction, DI, AOP, POJO



◎ AOP(Aspect Oriented Programming)

AOP는 애플리케이션 로직을 담당하는 코드에 남아 있는 기술
관련 코드를 분리해서 별도의 모듈로 관리하게 해주는 강력한
기술이다.

I Spring Framework 전략

❖ Spring 삼각형

엔터프라이즈 개발의 복잡함을 상대하는 Spring의 전략

→ Portable Service Abstraction, DI, AOP, POJO



◉ POJO(Plain Old Java Object)

POJO는 객체지향 원리에 충실하면서, 특정 환경이나 규약에 종속되지 않고 필요에 따라 재활용될 수 있는 방식으로 설계된 객체이다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

2. Spring Framework 특징

I Spring Framework 특징

01 컨테이너 역할

- Spring 컨테이너는 Java 객체의 LifeCycle을 관리하며, Spring 컨테이너로 부터 필요한 객체를 가져와 사용할 수 있다.

02 DI(Dependency Injection) 지원

- Spring은 설정 파일이나 어노테이션을 통해서 객체 간의 의존관계를 설정할 수 있도록 하고 있다.

I Spring Framework 특징

03 AOP(Aspect Oriented Programming) 지원

- Spring은 트랜잭션이나 로깅, 보안과 같이 공통적으로 필요로 하는 모듈들을 실제 핵심 모듈에서 분리해서 적용할 수 있다.

04 POJO(Plain Old Java Object) 지원

- Spring 컨테이너에 저장되는 Java객체는 특정한 인터페이스를 구현하거나, 특정 클래스를 상속받지 않아도 된다.

I Spring Framework 특징

05 트랜잭션 처리를 위한 일관된 방법을 지원

- JDBC, JTA 등 어떤 트랜잭션을 사용하던 설정을 통해 정보를 관리하므로 트랜잭션 구현에 상관없이 동일한 코드 사용 가능

06 영속성(Persistence)과 관련된 다양한 API 지원

- Spring은 MyBatis, Hibernate 등 데이터베이스 처리를 위한 ORM(Object Relational Mapping) 프레임워크들과의 연동 지원

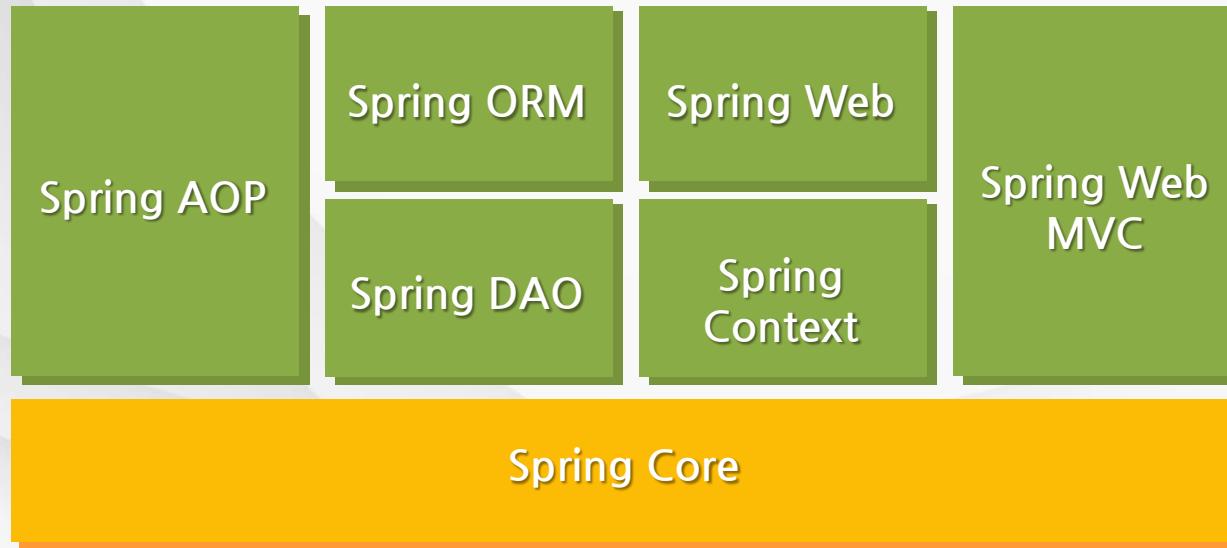
A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, showing blurred lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

3. Spring Framework 기능요소

| Spring 프레임워크를 구성하는 기능 요소



| Spring 프레임워크를 구성하는 기능 요소



Core
컨테이너

- Spring프레임워크의 기본기능을 제공한다.
- 이 모듈에 있는 BeanFactory는 Spring의 기본 컨테이너이면서 스프링 DI의 기반이다.

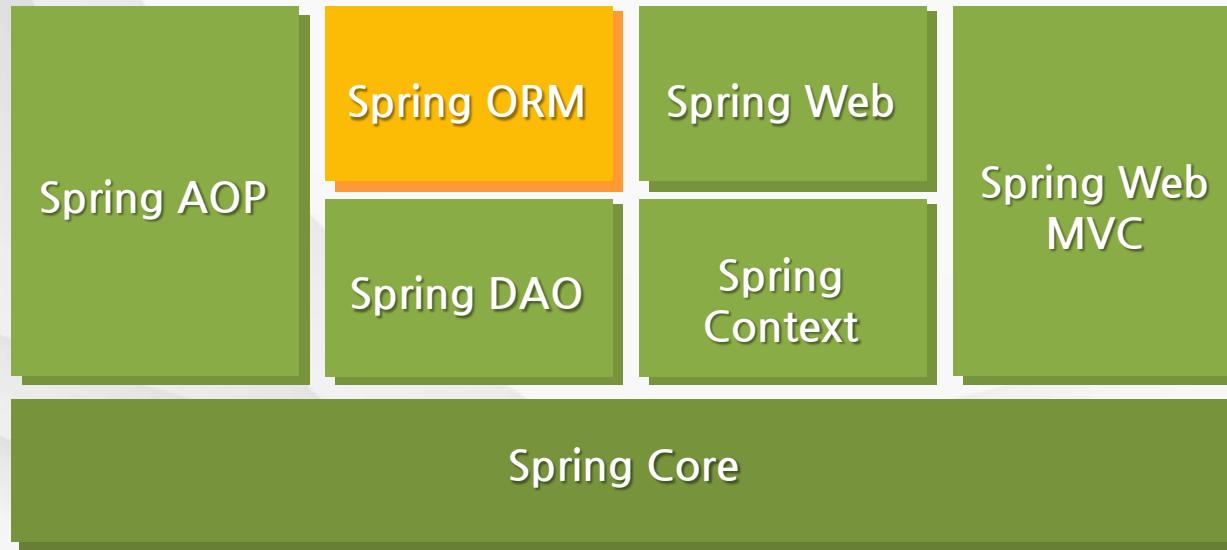
| Spring 프레임워크를 구성하는 기능 요소



AOP

- AOP 모듈을 통해 Aspect 지향 프로그래밍을 지원한다.
- AOP모듈은 스프링 애플리케이션에서 Aspect를 개발할 수 있는 기반을 지원한다.

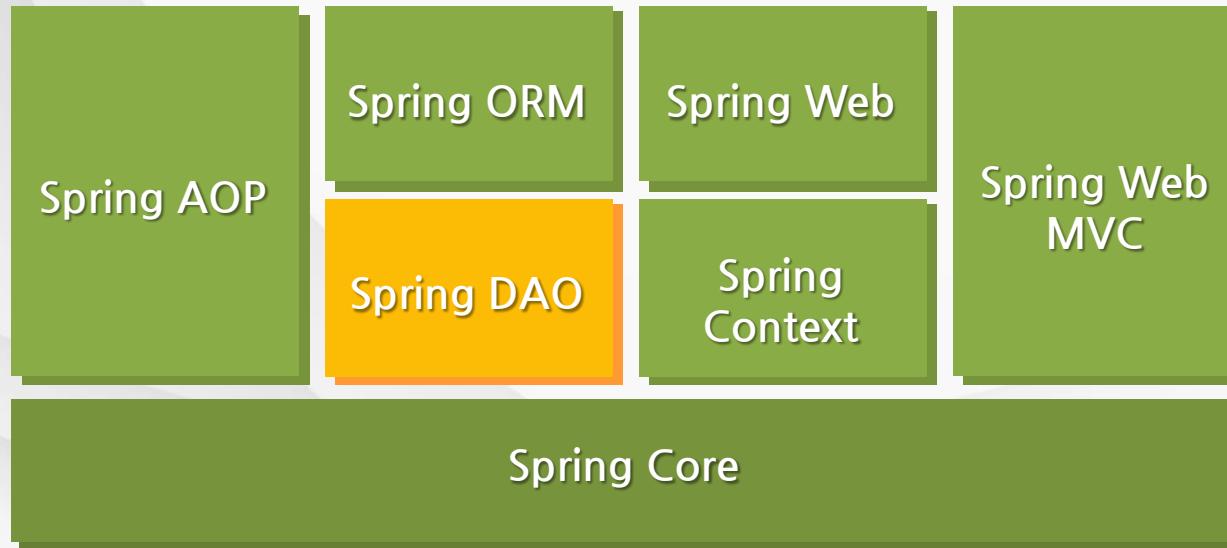
| Spring 프레임워크를 구성하는 기능 요소



ORM

- MyBatis, Hibernate, JPA 등 널리 사용되는 ORM 프레임워크와의 연결고리를 제공한다.
- ORM 제품들을 Spring의 기능과 조합해서 사용할 수 있도록 해준다.

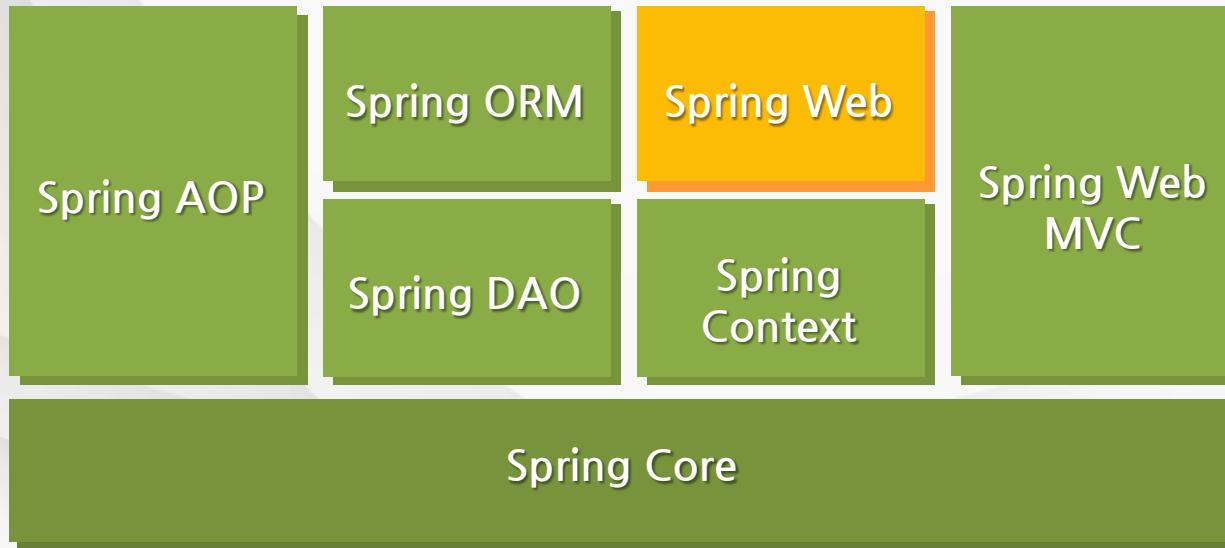
| Spring 프레임워크를 구성하는 기능 요소



DAO

- JDBC에 대한 추상화 계층으로 JDBC 코딩이나 예외처리 하는 부분을 간편화 시켰으며, AOP 모듈을 이용해 트랜잭션 관리 서비스도 제공한다.

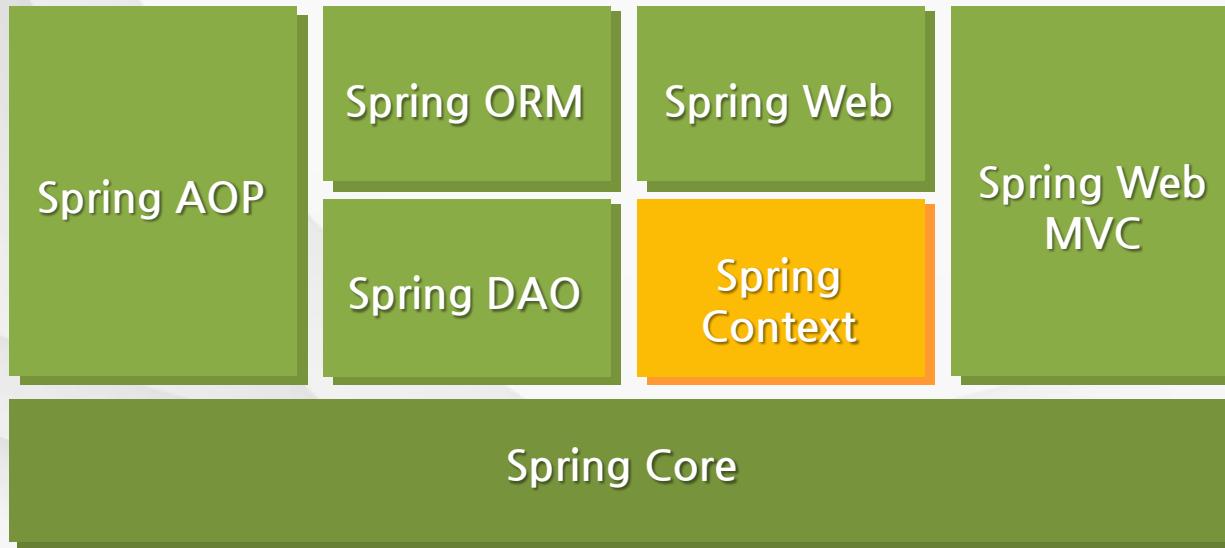
| Spring 프레임워크를 구성하는 기능 요소



Web

- 일반적인 웹애플리케이션 개발에 필요한 기본기능을 제공한다.
- Webwork나 Struts와 같은 다른 웹애플리케이션 프레임워크와의 통합을 지원한다.

| Spring 프레임워크를 구성하는 기능 요소



Context

- Context 모듈은 BeanFactory의 개념을 확장한 것으로 국제화(I18N) 메시지, 애플리케이션 생명주기 이벤트, 유효성 검증 등을 지원한다.

| Spring 프레임워크를 구성하는 기능 요소



**WebMVC
(Model/View
/Controller)**

- 사용자 인터페이스가 애플리케이션 로직과 분리되는 웹 애플리케이션을 만드는 경우에 일반적으로 사용되는 패러다임이다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring Framework 개요]에 대해서 살펴보았습니다.

Spring Framework 정의

Java 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크이다.

Spring Framework 전략

Portable Service Abstraction, DI, AOP, POJO

Spring Framework 특징

컨테이너, DI와 AOP, POJO 지원, 일관된 트랜잭션 처리방법, 다양한 ORM과의 연동

Spring Framework 기능요소

Core 컨테이너, Context, DAO, ORM, AOP, Web, WebMVC

Spring Framework

4. Spring Project 시작하기

CONTENTS

1

STS 소개 및 제공하는 기능

2

Maven과 Library 관리

3

Spring Project 작성하기

학습 목표

- STS 소개 및 제공하는 기능을 이해할 수 있습니다.
- Maven과 Library 관리에 대하여 이해할 수 있습니다.
- Spring Project 작성방법을 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights from streetlights or traffic.

1. STS 소개 및 제공하는 기능

I STS (SpringSource Tool Suite) 소개

Spring 개발업체인 SpringSource가 직접 만들어 제공하는
이클립스의 확장판으로 최신 이클립스를 기반으로 주요한 Spring
지원 플러그인과 관련된 도구를 모아서 Spring 개발에 최적화
되도록 만들어진 IDE이다.

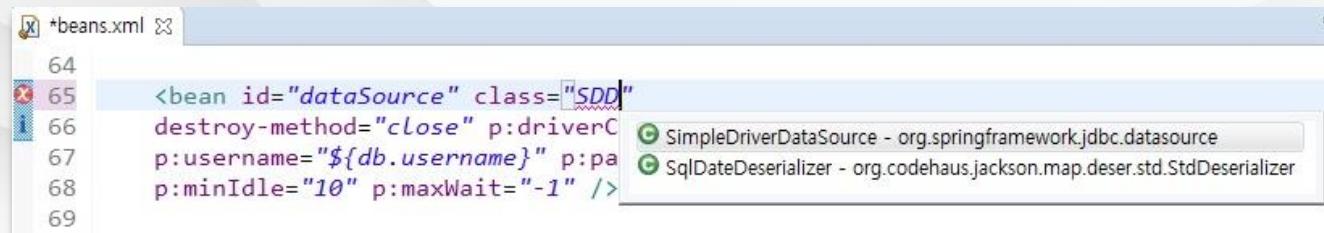


I STS가 제공하는 기능

01

Bean 클래스 이름 자동완성

- 현재 프로젝트의 모든 Source와 라이브러리, JDK안의 모든 클래스 중에서 첫 글자가 SDD로 시작하는 클래스를 자동으로 보여줌



The screenshot shows the Eclipse IDE interface with a code editor window titled "*beans.xml". The code is an XML configuration for a Spring bean named "dataSource". The line of interest is:

```
<bean id="dataSource" class="SDD"  
       destroy-method="close" p:driverC  
       p:username="${db.username}" p:pa  
       p:minIdle="10" p:maxWait="-1" />
```

The cursor is positioned at the start of the class attribute value "SDD". A tooltip appears, listing two suggestions:

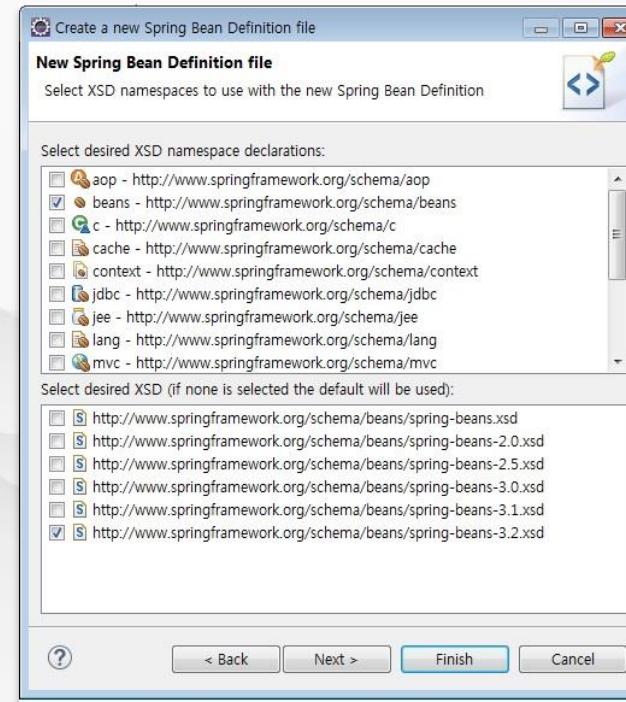
- SimpleDriverDataSource - org.springframework.jdbc.datasource
- SqlDateDeserializer - org.codehaus.jackson.map.deser.std.StdDeserializer

I STS가 제공하는 기능

02

설정파일 생성 위저드

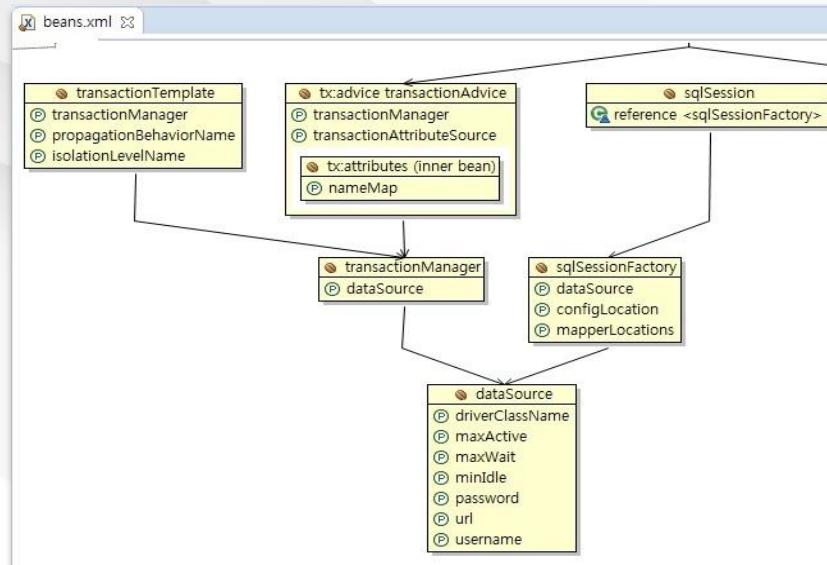
- Bean 설정파일 생성 위저드
중 사용할 Namespace와
Schema 버전을 선택하는
화면



I STS가 제공하는 기능

03 Bean 의존관계 그래프

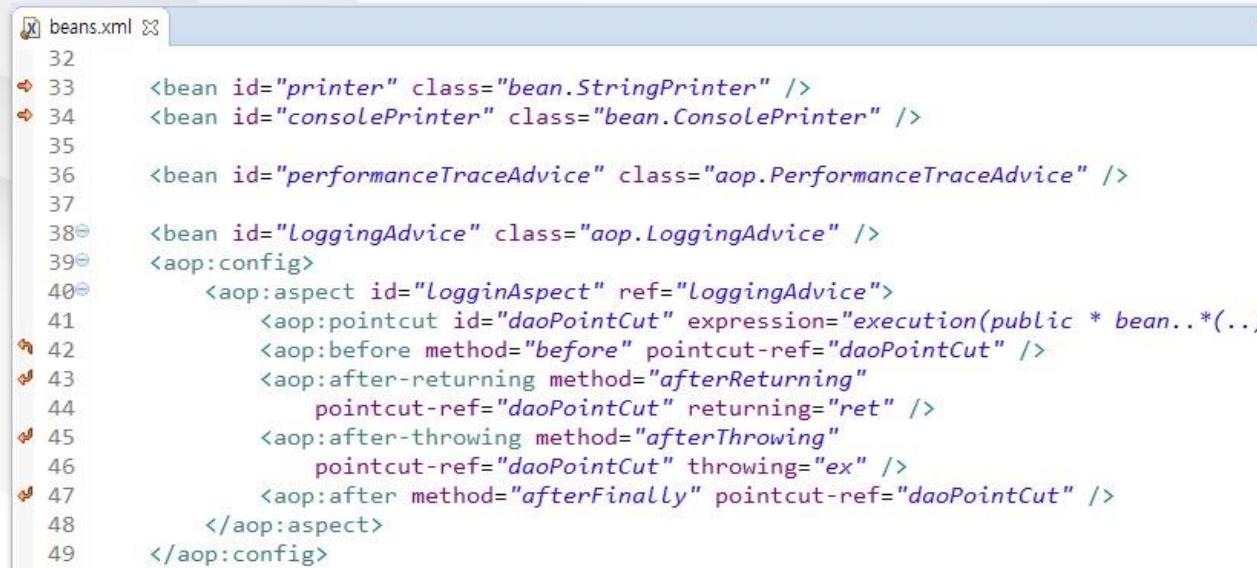
- Spring IDE는 XML 설정파일을 읽어서 자동으로 그래프 그려줌
- 각 Bean이 어떻게 참조되고, 어떤 Property를 갖는지 알 수 있음



I STS가 제공하는 기능

04 AOP 적용 대상 표시

- Spring IDE의 XML 설정파일 편집기를 이용하면 AOP의 적용 대상을 손쉽게 확인할 수 있다.



The screenshot shows the Eclipse IDE interface with the 'beans.xml' file open in the editor. The code is annotated with various icons, likely from the Spring IDE plugin, which highlight specific AOP-related elements. The annotations include:

- Blue arrows pointing to bean definitions (lines 32-37).
- Yellow arrows pointing to the `<aop:config>` element (line 38).
- Green arrows pointing to the `<aop:aspect>` element (line 40) and its nested `<bean id="logginAspect" ref="loggingAdvice">`.
- Purple arrows pointing to the `<ao:pointcut>` (line 41), `<ao:before>` (line 42), `<ao:after-returning>` (line 43), `<ao:after-throwing>` (line 45), and `<ao:after-finally>` (line 47) annotations.
- Red arrows pointing to the `expression`, `method`, `pointcut-ref`, `returning`, and `throwing` attributes throughout the code.

```
beans.xml
32
33     <bean id="printer" class="bean.StringPrinter" />
34     <bean id="consolePrinter" class="bean.ConsolePrinter" />
35
36     <bean id="performanceTraceAdvice" class="aop.PerformanceTraceAdvice" />
37
38     <bean id="LoggingAdvice" class="aop.LoggingAdvice" />
39     <aop:config>
40         <aop:aspect id="logginAspect" ref="loggingAdvice">
41             <ao:pointcut id="daoPointCut" expression="execution(public * bean..*(..))"/>
42             <ao:before method="before" pointcut-ref="daoPointCut" />
43             <ao:after-returning method="afterReturning"
44                 pointcut-ref="daoPointCut" returning="ret" />
45             <ao:after-throwing method="afterThrowing"
46                 pointcut-ref="daoPointCut" throwing="ex" />
47             <ao:after method="afterFinally" pointcut-ref="daoPointCut" />
48         </aop:aspect>
49     </aop:config>
```

A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, showing several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

2. Maven과 Library 관리

I Maven이란?

<http://maven.apache.org> 라이브러리 관리 + 빌드 툴

❖ Maven을 사용하는 이유

- 편리한 Dependent Library 관리 - Dependency Management
- 여러 프로젝트에서 프로젝트 정보나 jar파일들을 공유하기 쉬움
- 모든 프로젝트의 빌드 프로세스를 일관되게 가져갈 수 있음

I Maven이전의 Library 관리 방법



I Maven의 Library 관리 방법



pom.xml

- ◎ Maven 프로젝트를 생성하면 pom.xml 파일이 생성된다.
- ◎ pom.xml 파일은 Project Object Model 정보를 담고 있다.

```
<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>SpringDI</groupId>
    <artifactId>SpringDI</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <build>
        <sourceDirectory>src</sourceDirectory>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <!-- http://mvnrepository.com/artifact/org.springframework/spring-context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>3.2.17.RELEASE</version>
        
```

| pom.xml 의존관계(dependency) 추가

❖ Spring 프레임워크 설치

- ◎ <http://mvnrepository.com> 접근한다.
- ◎ org.springframework로 검색한다.
- ◎ spring-jdbc 모듈과 spring-web 모듈을 추가한다.

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>3.2.3.RELEASE</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>3.2.3.RELEASE</version>
</dependency>
```

pom.xml 의존관계(dependency)

SpringDIWeb/pom.xml

Dependencies

Dependencies

- spring-context : 3.2.17.RELEASE
- spring-test : 3.2.17.RELEASE
- spring-aop : 3.2.17.RELEASE
- aspectjrt : 1.7.4
- aspectjweaver : 1.7.4
- spring-jdbc : 3.2.17.RELEASE
- ojdbc14 : 10.2.0.4.0
- commons-dbcp : 1.4
- mybatis : 3.3.1
- mybatis-spring : 1.2.5
- ojdbc6 : 11.1.0.7.0
- spring-webmvc : 3.2.17.RELEASE
- javax.servlet-api : 3.1.0
- jstl : 1.2
- jackson-mapper-asl : 1.9.13
- log4jdbc-log4j2-jdbc4 : 1.16

To manage your transitive dependency exclusions, please use the [Dependency Hierarchy](#) page.

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

SpringDIWeb/pom.xml

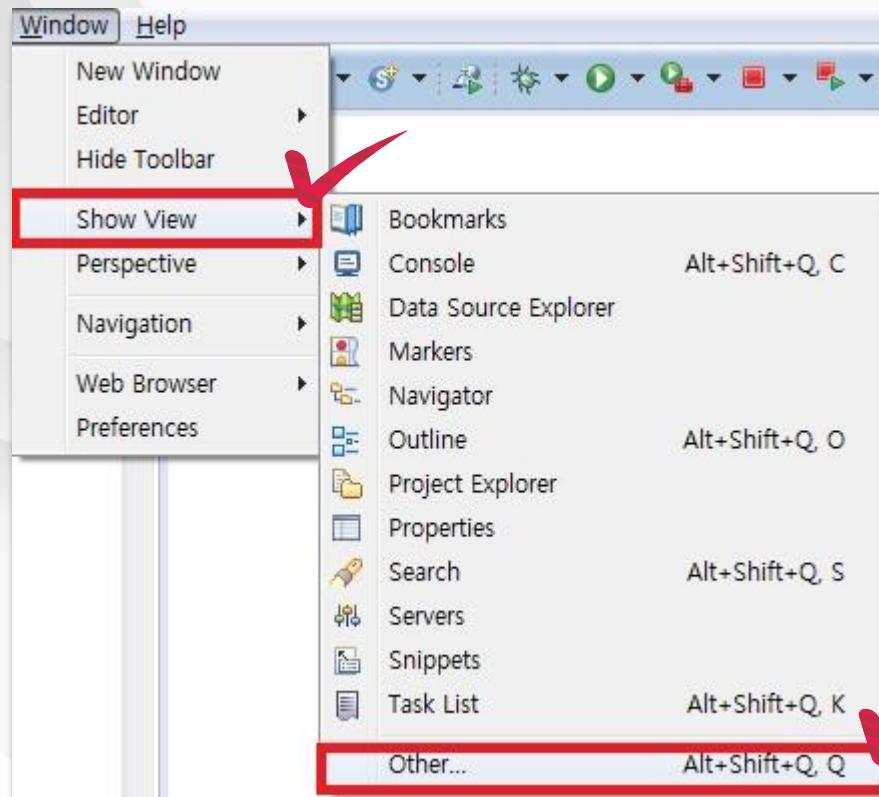
Dependency Hierarchy [test]

Dependency Hierarchy

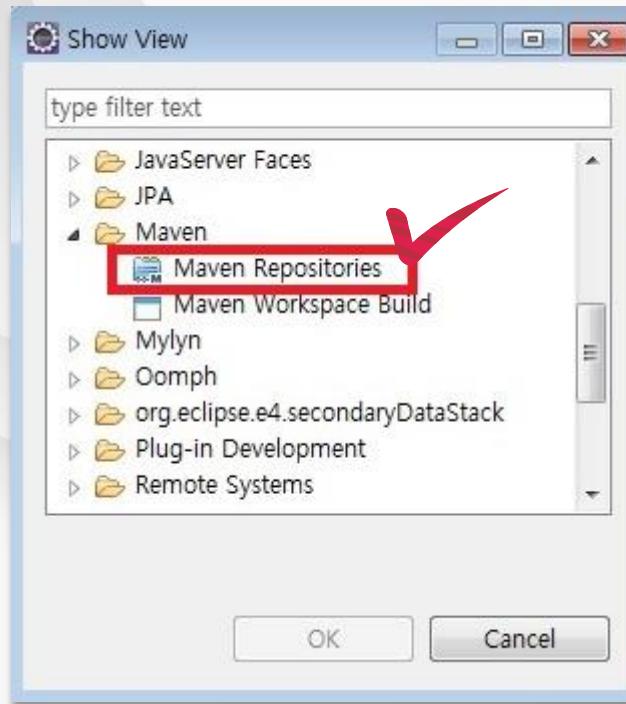
- spring-context : 3.2.17.RELEASE [compile]
 - spring-aop : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [
 - spring-beans : 3.2.17.RELEASE [compile]
 - spring-core : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [
 - spring-core : 3.2.17.RELEASE [compile]
 - commons-logging : 1.1.3 [compile]
 - spring-expression : 3.2.17.RELEASE [compile]
 - spring-core : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [
- spring-test : 3.2.17.RELEASE [compile]
 - spring-core : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [
- spring-aop : 3.2.17.RELEASE [compile]
 - aopalliance : 1.0 [compile]
 - spring-beans : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [
 - spring-core : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [
 - aspectjrt : 1.7.4 [compile]
 - aspectjweaver : 1.7.4 [compile]
- spring-jdbc : 3.2.17.RELEASE [compile]
 - spring-beans : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [
 - spring-core : 3.2.17.RELEASE (omitted for conflict with 3.2.17.RELEASE) [

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

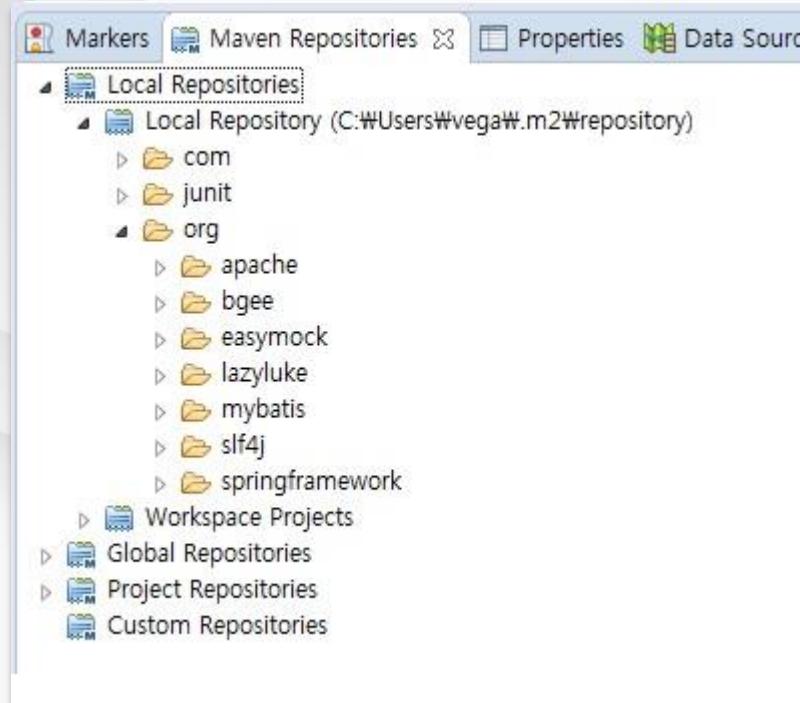
Eclipse 제공 : Maven Repositories View



I Eclipse 제공 : Maven Repositories View



Eclipse 제공 : Maven Repositories View

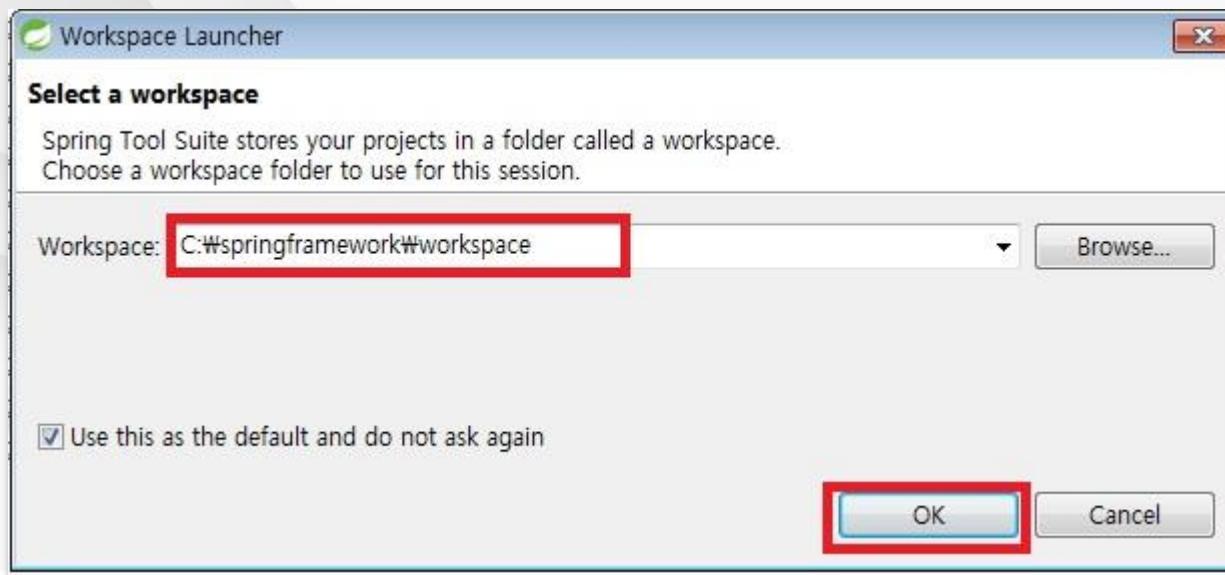


A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

3. Spring Project 작성하기

I STS 시작하기

- ① C:\springframework\sts-bundle\sts 3.7.3.RELEASE\STS.exe 실행



I Spring Project 생성 및 Spring Module 설치

- Java Project -> Convert to Maven Project -> Add Spring Project Nature
- pom.xml 파일에 dependency 추가 :
<https://mvnrepository.com>에서 spring context module 검색

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring Project 시작하기]에 대해서 살펴보았습니다.

STS 소개 및 제공하는 기능

Spring 개발에 최적화된 IDE, 클래스 자동완성기능, 설정파일 생성 위저드

Maven과 Library 관리

- ◎ 라이브러리 관리 + 빌드 툴, pom.xml
- ◎ 편리한 Dependent Library 관리 기능

Spring Project 시작하기

Java Project -> Convert to Maven Project -> Add Spring Project Nature

Spring Framework

5. IoC와 DI

CONTENTS

1 IoC(Inversion of Control)

2 DI(Dependency Injection)

3 Spring DI 컨테이너

학습 목표

- IoC(Inversion of Control)에 대하여 이해할 수 있습니다.
- DI(Dependency Injection)에 대하여 이해할 수 있습니다.
- Spring DI 컨테이너에 대하여 이해할 수 있습니다.

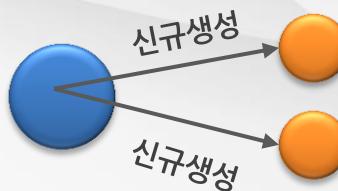
A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

1. IoC(Inversion of Control)

I IoC의 개념

IoC(제어권의 역전)이란, 객체의 생성, 생명주기의 관리까지 모든 객체에 대한 제어권이 바뀌었다는 것을 의미한다.

- 컴포넌트 의존관계 결정 (component dependency resolution), 설정(configuration) 및 생명주기(lifecycle)를 해결하기 위한 디자인 패턴(Design Pattern)



IoC가 아닌 경우



IoC인 경우

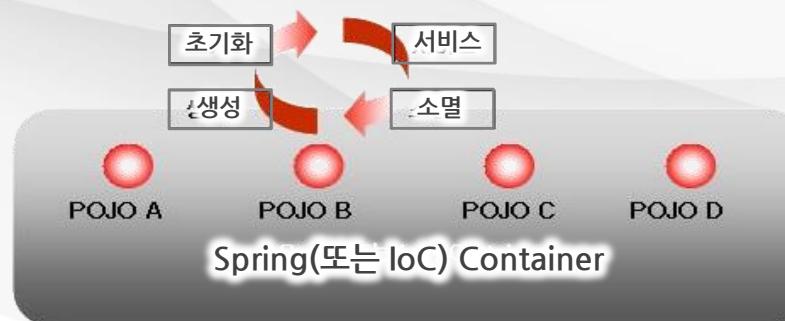
IoC 컨테이너

- 스프링 프레임워크도 **객체에 대한 생성 및 생명주기를 관리할 수 있는 기능을 제공하고 있음.** 즉, IoC 컨테이너 기능을 제공한다.

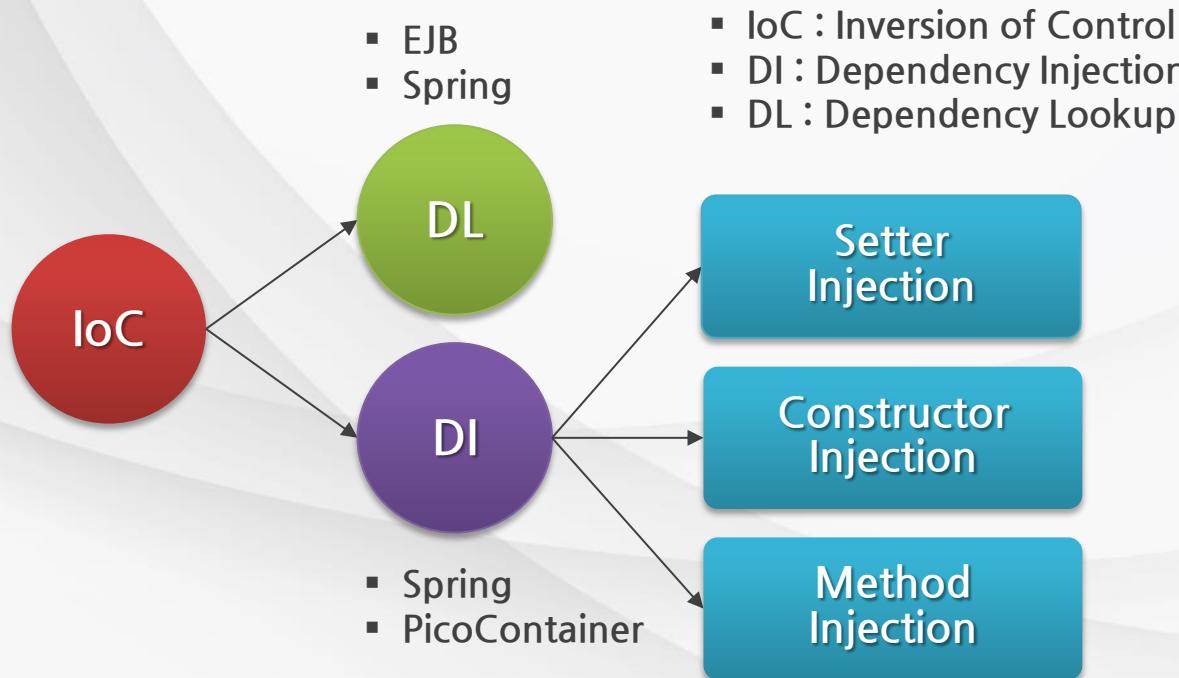
IoC 컨테이너는 객체의 생성을 책임지고, 의존성을 관리한다.

POJO의 생성, 초기화, 서비스, 소멸에 대한 권한을 가진다.

개발자들이 직접 POJO를 생성할 수 있지만 컨테이너에게 맡긴다.



| IoC의 분류



| DL(Dependency Lookup)과 DI(Dependency Injection)

DL (Dependency Lookup) 의존성 검색	저장소에 저장되어 있는 Bean에 접근하기 위해 컨테이너가 제공하는 API를 이용하여 Bean을 Lookup 하는 것
DI (Dependency Injection) 의존성 주입	각 클래스간의 의존관계를 빠 설정(Bean Definition) 정보를 바탕으로 컨테이너가 자동으로 연결해주는 것

- ◎ DL 사용시 컨테이너 종속성이 증가하여, 주로 DI를 사용함

Setter
Injection

Constructor
Injection

Method
Injection

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, showing blurred lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

2. DI(Dependency Injection)

I DI의 개념

각 클래스간의 의존관계를 빈 설정 (Bean Definition) 정보를 바탕으로 컨테이너가 자동으로 연결해주는 것을 말함

- 개발자들은 단지 빈 설정파일에서 의존관계가 필요하다는 정보를 추가하면 된다.
- 객체 레퍼런스를 컨테이너로부터 주입 받아서, 실행 시에 동적으로 의존관계가 생성된다.
- 컨테이너가 흐름의 주체가 되어 애플리케이션 코드에 의존관계를 주입해 주는 것이다.

DI
(Dependency Injection)
장점

- 코드가 단순해진다.
- 컴포넌트 간의 결합도가 제거된다.

I DI의 유형

Setter Injection

Setter 메서드를 이용한 의존성 삽입

- 의존성을 입력 받는 setter 메서드를 만들고 이를 통해 의존성을 주입한다.

Constructor Injection

생성자를 이용한 의존성 삽입

- 필요한 의존성을 포함하는 클래스의 생성자를 만들고 이를 통해 의존성을 주입한다.

Method Injection

일반 메서드를 이용한 의존성 삽입

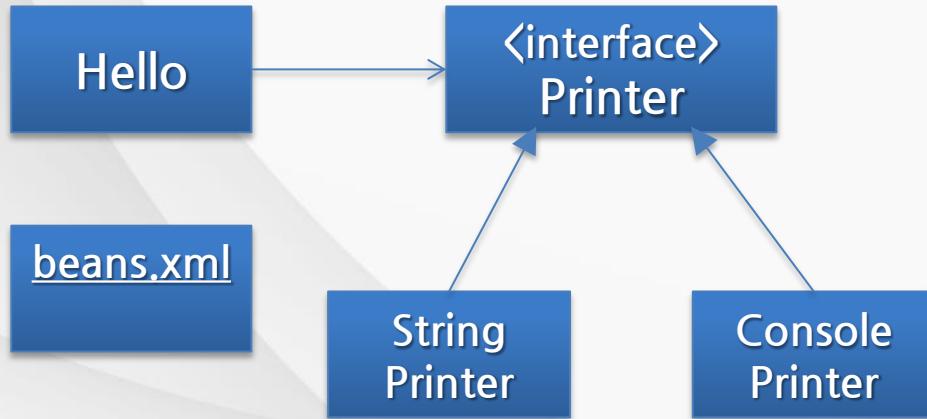
- 의존성을 입력 받는 일반 메서드를 만들고 이를 통해 의존성을 주입한다.

I DI를 이용한 클래스 호출방식



2. DI(Dependency Injection)

Setter Injection

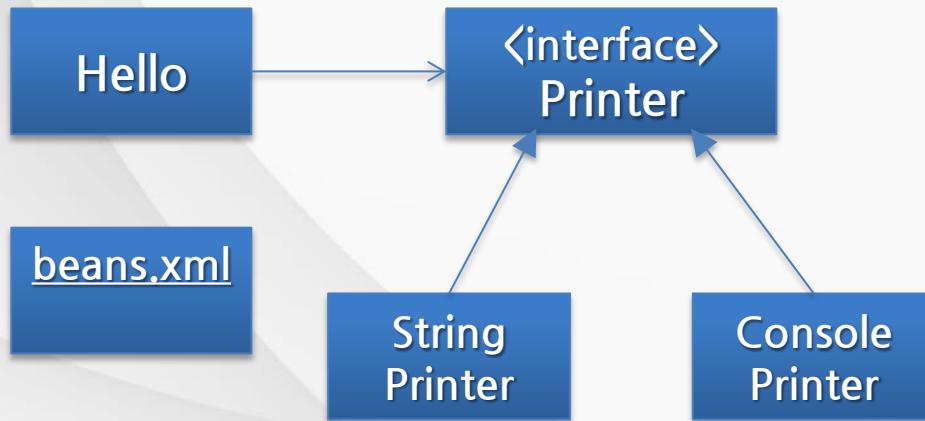


```
beans.xml
10
11<bean id="hello" class="bean.Hello">
12  <property name="name" value="Spring" />
13  <property name="printer" ref="printer" />
14</bean>
15
16<bean id="printer" class="bean.StringPrinter" />
17<bean id="consolePrinter" class="bean.ConsolePrinter" />
```

```
Hello.java
1 package bean;
2
3 import java.util.List;
4
5 public class Hello {
6     String name;
7     Printer printer;
8
9     public Hello() { }
10
11    public void setName(String name) {
12        this.name = name;
13    }
14
15    public void setPrinter(Printer printer) {
16        this.printer = printer;
17    }
}
```

2. DI(Dependency Injection)

Constructor Injection



```
beans.xml
```

```
1 package bean;
2
3 import java.util.List;
4
5 public class Hello {
6     String name;
7     Printer printer;
8
9     public Hello() { }
10
11    public Hello(String name, Printer printer) {
12        this.name = name;
13        this.printer = printer;
14    }
}
```

A code editor window showing the `Hello.java` file. The constructor definition is highlighted with a red border. The code defines a `Hello` class with a string attribute `name` and a `Printer` attribute. It has a no-argument constructor and a constructor that takes `String name` and `Printer printer` as parameters, setting them using `this` references.

```
beans.xml
```

```
11
12<bean id="hello" class="bean.Hello">
13    <constructor-arg index="0" value="Spring" />
14    <constructor-arg index="1" ref="printer" />
15</bean>
16
17<bean id="printer" class="bean.StringPrinter" />
18<bean id="consolePrinter" class="bean.ConsolePrinter" />
```

A code editor window showing the `beans.xml` configuration file. The second bean definition is highlighted with a red border. It defines a bean with `id="hello"` of type `bean.Hello`. It has two constructor arguments: the first is a string value "Spring", and the second is a reference to a bean with `id="printer"`. Below this, there are definitions for beans with `id="printer"` and `id="consolePrinter"`.

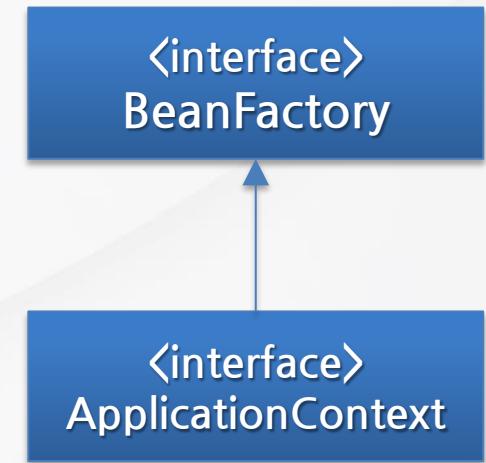
A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

3. Spring DI 컨테이너

| Spring DI 컨테이너의 개념

Spring DI 컨테이너가 관리하는 객체를 **빈(bean)**이라고 하고,
이 빈(bean)들을 관리한다는 의미로 컨테이너를
빈 팩토리 (BeanFactory)라고 부른다.

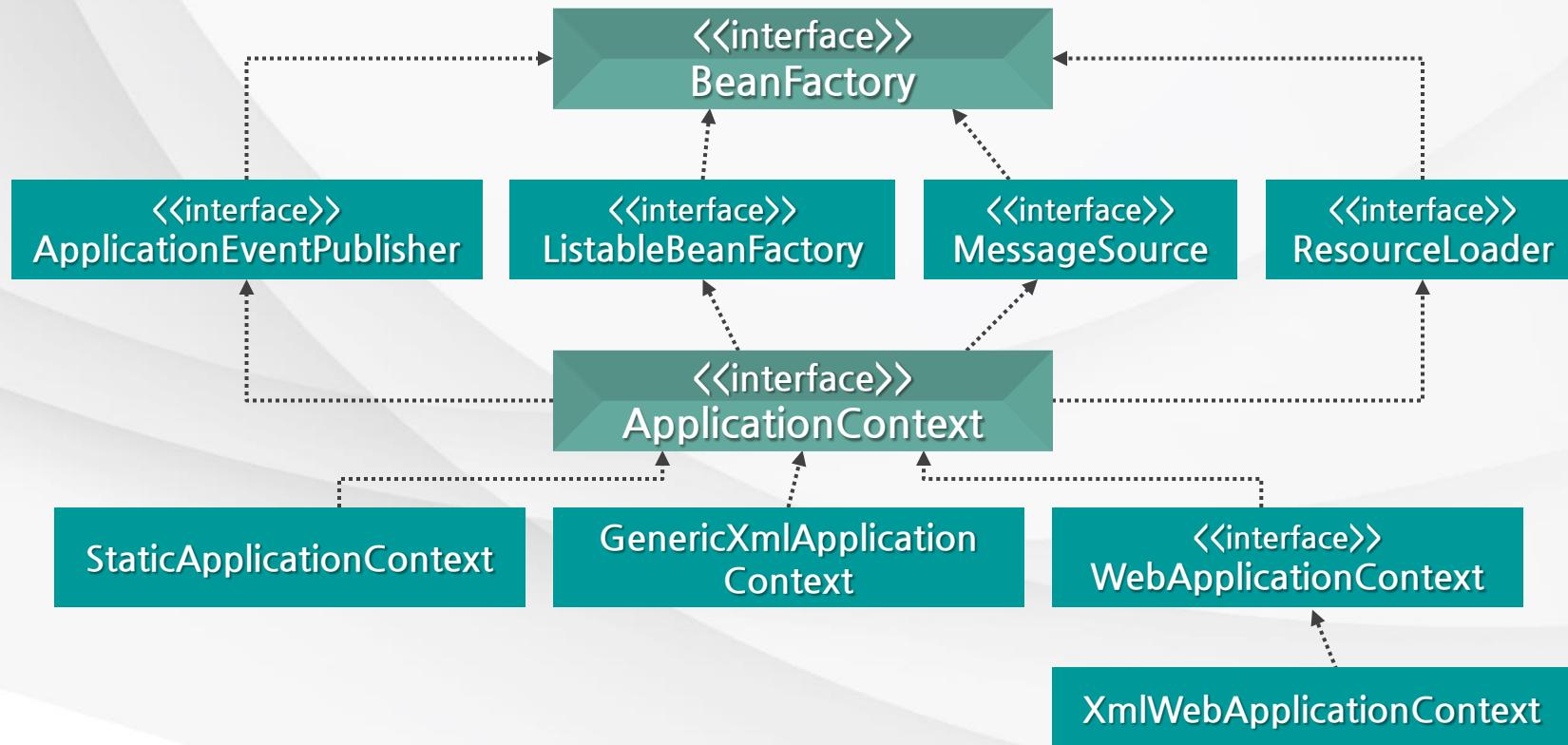
- 객체의 생성과 객체 사이의 런타임(run-time) 관계를 DI 관점에서
볼 때는 컨테이너를 **BeanFactory**라고 한다.
- Bean Factory에 여러 가지 컨테이너 기능을 추가하여
애플리케이션 컨텍스(ApplicationContext)라고 부름



I BeanFactory와 ApplicationContext

BeanFactory	<ul style="list-style-type: none">▪ Bean을 등록, 생성, 조회, 반환 관리함▪ 보통은 BeanFactory를 바로 사용하지 않고, 이를 확장한 ApplicationContext를 사용함▪ getBean() 메서드가 정의되어 있음
ApplicationContext	<ul style="list-style-type: none">▪ Bean을 등록, 생성, 조회, 반환 관리하는 기능은 BeanFactory와 같음▪ Spring의 각종 부가 서비스를 추가로 제공함▪ Spring이 제공하는 ApplicationContext 구현 클래스가 여러 가지 종류가 있음

I BeanFactory와 ApplicationContext



A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [IoC와 DI]에 대해서 살펴보았습니다.

IoC(Inversion of Control)

제어의 역전, IoC 컨테이너, DL, DI

DI(Dependency Injection)

- ◎ 클래스 간의 의존관계를 컨테이너가 주입
- ◎ Setter Injection, Constructor Injection

Spring DI 컨테이너

BeanFactory, ApplicationContext

Spring Framework

6. DI 애플리케이션 작성(1)

CONTENTS

1

Spring DI 용어

2

POJO 클래스 작성

3

설정 메타정보 XML 작성

4

DI 테스트 클래스 작성

학습 목표

- Spring DI 용어에 대하여 이해할 수 있습니다.
- POJO 클래스를 작성할 수 있습니다.
- 설정 메타정보 XML을 작성할 수 있습니다.
- DI 테스트 클래스를 작성할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

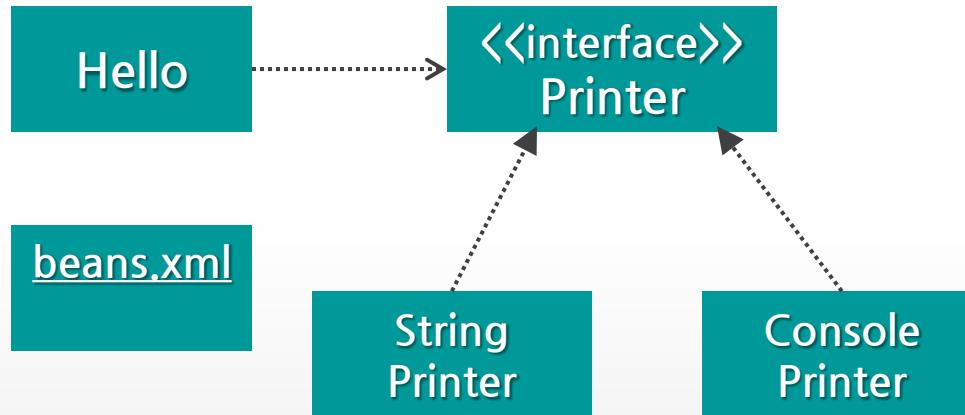
1. Spring DI 용어

빈 (Bean)	<ul style="list-style-type: none">▪ 스프링이 IoC 방식으로 관리하는 객체라는 뜻▪ 스프링이 직접 생성과 제어를 담당하는 객체를 Bean이라고 부름
빈 팩토리 (BeanFactory)	<ul style="list-style-type: none">▪ 스프링의 IoC를 담당하는 핵심 컨테이너를 가리킴▪ Bean을 등록, 생성, 조회, 반환하는 기능을 담당함▪ 이 BeanFactory를 바로 사용하지 않고 이를 확장한 ApplicationContext를 주로 이용함
애플리케이션 컨텍스트 (ApplicationContext)	<ul style="list-style-type: none">▪ BeanFactory를 확장한 IoC 컨테이너▪ Bean을 등록하고 관리하는 기능은 BeanFactory와 동일하지만 스프링이 제공하는 각종 부가 서비스를 추가로 제공함▪ 스프링에서는 ApplicationContext를 BeanFactory 보다 더 많이 사용함
설정 메타정보 (Configuration metadata)	<ul style="list-style-type: none">▪ ApplicationContext 또는 BeanFactory가 IoC를 적용하기 위해 사용하는 메타정보를 말함▪ 설정 메타정보는 IoC컨테이너에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용됨

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

2. POJO 클래스 작성

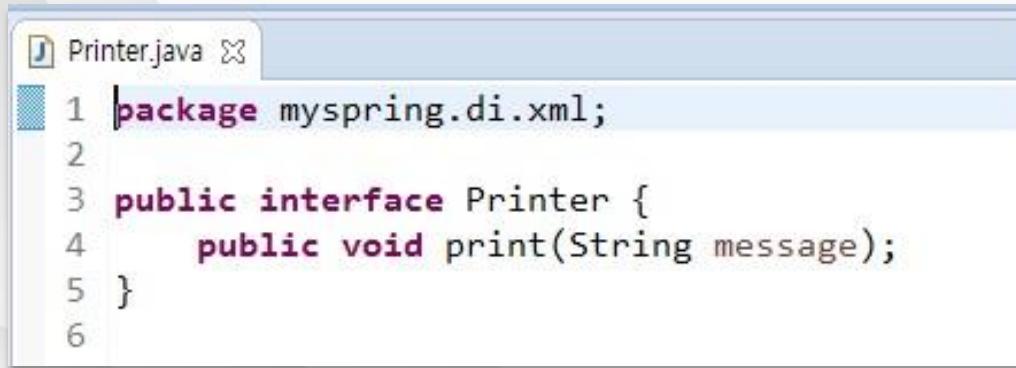
I POJO 클래스 다이어그램



Hello.java

```
1 package myspring.di.xml;
2
3 public class Hello {
4     String name;
5     Printer printer;
6
7     public Hello() {}
8
9     public void setName(String name) {
10         this.name = name;
11     }
12
13    public void setPrinter(Printer printer) {
14        this.printer = printer;
15    }
16
17    public String sayHello() {
18        return "Hello " + name;
19    }
20
21    public void print() {
22        this.printer.print(sayHello());
23    }
24 }
```

Printer.java



The image shows a screenshot of a Java code editor. The title bar says "Printer.java". The code in the editor is:

```
1 package myspring.di.xml;
2
3 public interface Printer {
4     public void print(String message);
5 }
6
```

| StringPrinter.java

```
1 package myspring.di.xml;
2
3 public class StringPrinter implements Printer {
4     private StringBuffer buffer = new StringBuffer();
5
6     public void print(String message) {
7         this.buffer.append(message);
8     }
9
10    public String toString() {
11        return this.buffer.toString();
12    }
13 }
14
```

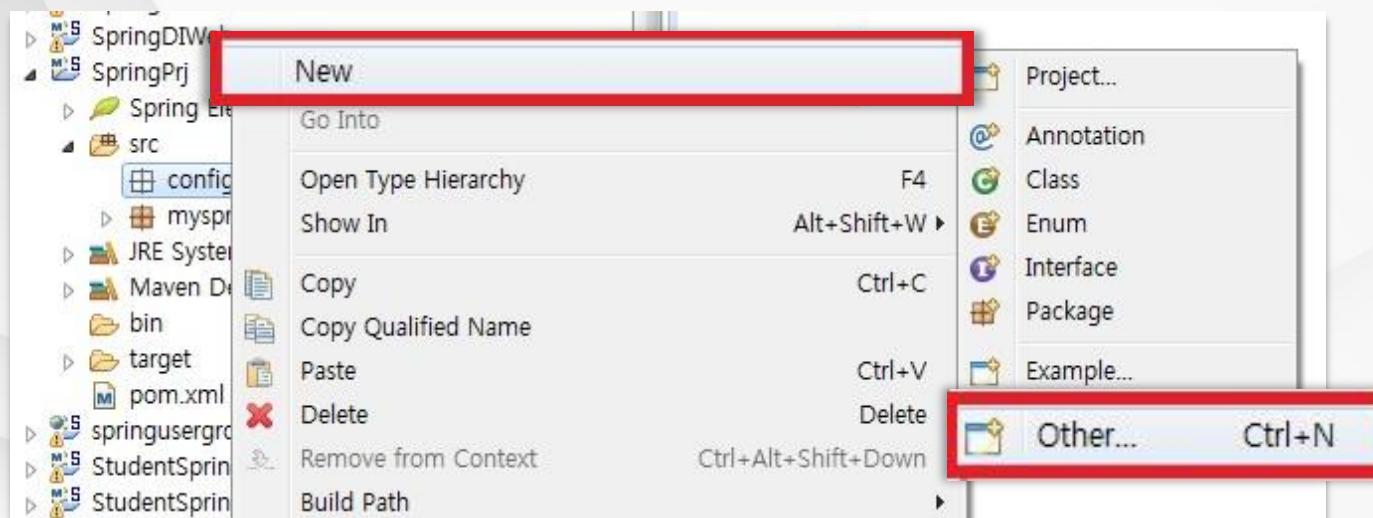
ConsolePrinter.java

```
1 package myspring.di.xml;
2
3 public class ConsolePrinter implements Printer {
4     public void print(String message) {
5         System.out.println(message);
6     }
7 }
8
```

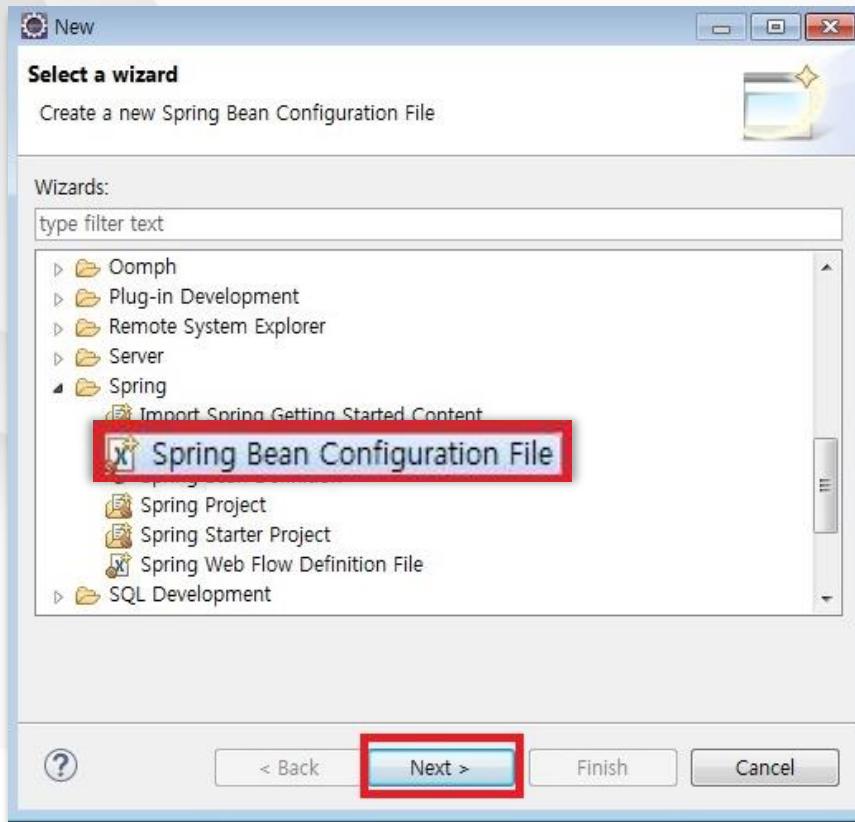
A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban environment.

3. 설정 메타정보 XML 작성

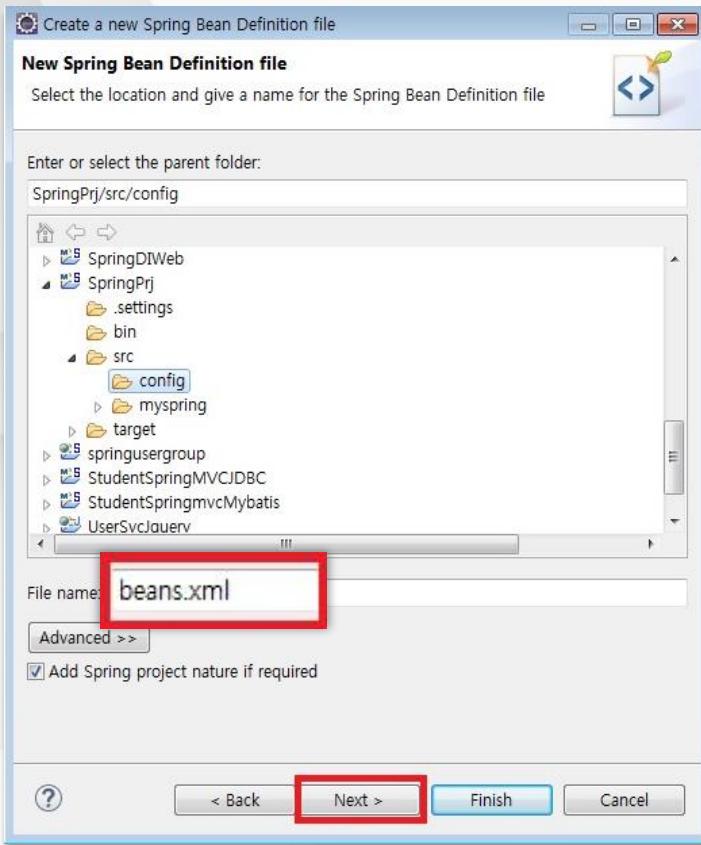
Bean Configuration(빈 설정) XML 작성 (1)



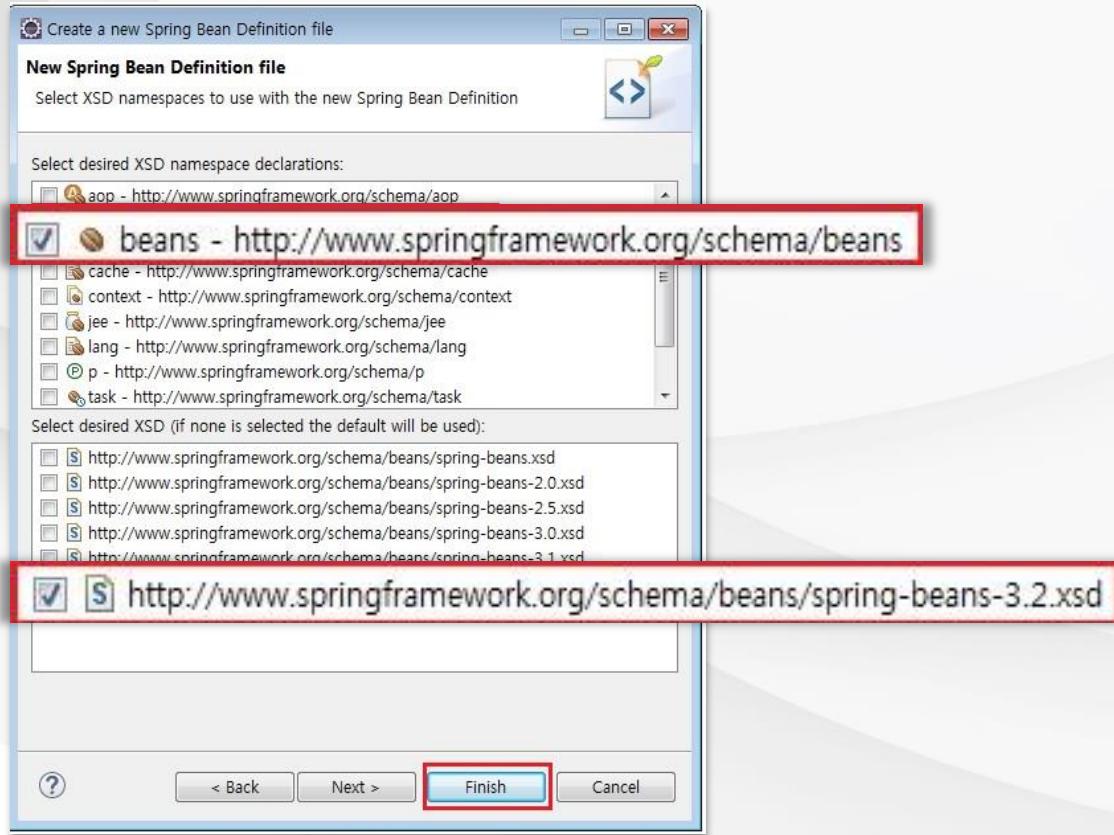
Bean Configuration(빈 설정) XML 작성 (2)



Bean Configuration(빈 설정) XML 작성 (3)



Bean Configuration(빈 설정) XML 작성 (4)



Bean Configuration(빈 설정) XML 작성 (5)



The screenshot shows a code editor window with a tab labeled "beans.xml". The content of the file is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.springframework.org/schema/beans http:
5
6     <bean id="hello" class="myspring.di.xml.Hello">
7       <property name="name" value="Spring" />
8       <property name="printer" ref="printer" />
9     </bean>
10
11    <bean id="printer" class="myspring.di.xml.StringPrinter" />
12    <bean id="consolePrinter" class="myspring.di.xml.ConsolePrinter" />
13
14 </beans>
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

4. DI 테스트 클래스 작성

HelloBeanTest.java

```
*HelloBeanTest.java ✘
1 package myspring.di.xml.test;
2
3+import org.springframework.context.ApplicationContext;□
4
5 public class HelloBeanTest {
6     public static void main(String[] args) {
7         ApplicationContext context =
8             new GenericXmlApplicationContext("config/beans.xml");
9         Hello hello = (Hello) context.getBean("hello");
10        System.out.println(hello.sayHello());
11        hello.print();
12        Printer printer = (Printer) context.getBean("printer");
13        System.out.println(printer.toString());
14
15        Hello hello2 = context.getBean("hello", Hello.class);
16        hello2.print();
17
18        System.out.println(hello == hello2);
19    }
20 }
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [DI 애플리케이션 작성(1)]에 대해서 살펴보았습니다.

Spring DI 용어

빈(Bean), 빈 팩토리(BeanFactory), 애플리케이션 컨텍스트(ApplicationContext),
설정 메타정보(Configuration Metadata)

POJO 클래스 작성

의존관계가 있는 Java 클래스 작성
Hello.java, Printer.java, StringPrinter.java, ConsolePrinter.java

설정 메타정보 XML 작성

빈 설정(Bean Configuration) XML 파일작성 - beans.xml

DI 테스트 클래스 작성

DI 컨테이너 (ApplicationContext)를 사용한 테스트 클래스 작성

Spring Framework

7. DI 애플리케이션 작성(2)

CONTENTS

- 1 jUnit의 개요와 특징
- 2 jUnit을 사용한 DI 테스트 클래스
- 3 Spring-Test를 사용한 DI 테스트 클래스

학습 목표

- jUnit의 개요와 특징에 대하여 이해할 수 있습니다.
- jUnit을 사용한 DI 테스트 클래스를 작성할 수 있습니다.
- Spring-Test를 사용한 DI 테스트 클래스를 작성할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

1. jUnit의 개요와 특징

| jUnit의 개요

Java에서 독립된 단위테스트(Unit Test)를 지원해주는
프레임워크이다.

❖ 단위테스트(Unit Test)란?

- 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차, 즉 모든 함수와 메소드에 대한 테스트 케이스(Test case)를 작성하는 절차를 말한다.
- jUnit은 보이지 않고 숨겨진 단위 테스트를 끌어내어 정형화시켜 단위테스트를 쉽게 해주는 테스트 지원 프레임워크다.

| jUnit의 특징

- ◎ TDD의 창시자인 Kent Beck과 디자인 패턴 책의 저자인 Erich Gamma가 작성했다.
- ◎ 단정(assert) 메서드로 테스트 케이스의 수행 결과를 판별한다.
예) assertEquals(예상 값, 실제 값)
- ◎ jUnit4부터는 테스트를 지원하는 어노테이션을 제공한다.
@Test @Before @After
- ◎ 각 @Test 메서드가 호출할 때마다 새로운 인스턴스를 생성하여 독립적인 테스트가 이루어지도록 한다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

2. jUnit을 사용한 DI 테스트 클래스

I jUnit 라이브러리 설치

<http://mvnrepository.com>에 접근한다.

junit으로 검색한다.

junit 4.12 버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
```

| jUnit에서 테스트를 지원하는 어노테이션(Annotation) (1)

@Test

- ◎ @Test가 선언된 메서드는 테스트를 수행하는 메소드가 된다.
- ◎ Junit은 각각의 테스트가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test마다 객체를 생성한다.

@Ignore

- ◎ @Ignore가 선언된 메서드는 테스트를 실행하지 않게 한다.

@Before

- ◎ @Before가 선언된 메서드는 @Test 메소드가 실행되기 전에 반드시 실행되어 진다.
- ◎ @Test 메소드에서 공통으로 사용하는 코드를 @Before 메소드에 선언하여 사용하면 된다.

| jUnit에서 테스트를 지원하는 어노테이션(Annotation) (2)

@After

- @After가 선언된 메서드는 @Test 메소드가 실행된 후 실행된다.

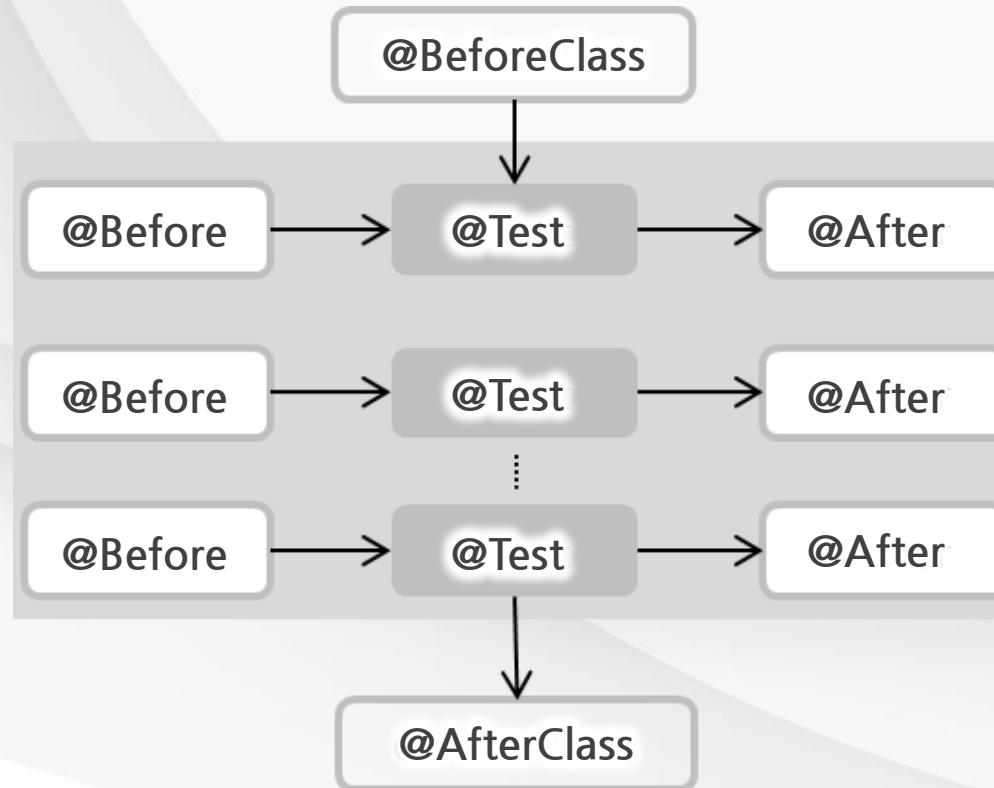
@BeforeClass

- @BeforeClass 어노테이션은 @Test 메소드 보다 먼저 한번만 수행되어야 할 경우에 사용하면 된다.

@AfterClass

- @AfterClass 어노테이션은 @Test 메소드 보다 나중에 한번만 수행되어야 할 경우에 사용하면 된다.

jUnit에서 테스트를 지원하는 어노테이션(Annotation) (3)



| 테스트 결과를 확인하는 단정(assert) 메서드

org.junit.Assert

- +assertArrayEquals(expected, actual)
- +assertEquals(expected, actual)
- +assertNotNull(object)
- +assertSame(expected, actual)
- +assertTrue(object)

I 테스트 결과를 확인하는 단정(assert) 메서드

assertEquals(a, b);

- 객체 A와 B가 일치함을 확인한다.

assertArrayEquals(a, b);

- 배열 A와 B가 일치함을 확인한다.

assertSame(a, b);

- 객체 A와 B가 같은 객체임을 확인한다.
- assertEquals 메서드는 두 객체의 값이 같은지 확인하고, assertSame메서드는 두 객체의 레퍼런스가 동일한가를 확인한다.
(== 연산자)

assertTrue(a);

- 조건 A가 참인가를 확인한다.

assertNotNull(a);

- 객체 A가 null이 아님을 확인한다.

- ◎ 이외에도 다양한 assert 메서드가 존재함.

- <http://junit.sourceforge.net/javadoc/org/junit/Assert.html>

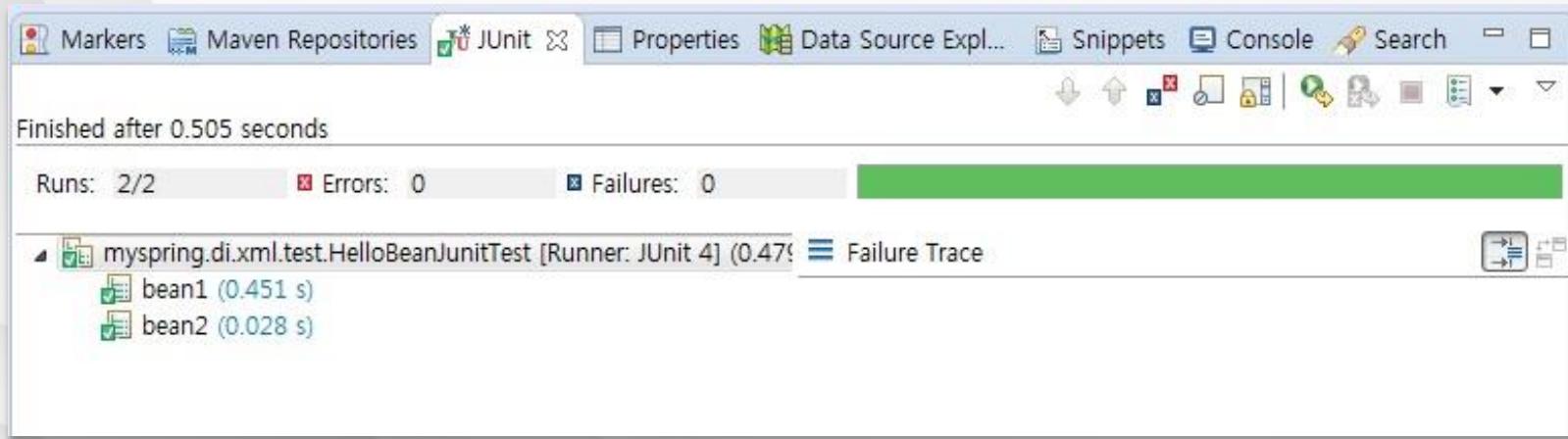
| jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 작성

```
1 HelloBeanJunitTest.java
2
3+ import static org.junit.Assert.*;
4
5
6 public class HelloBeanJunitTest {
7     private ApplicationContext context;
8
9
10    @Before
11    public void init() {
12        context = new GenericXmlApplicationContext("config/beans.xml");
13    }
14
15    @Test
16    public void bean1() {
17        Hello hello = (Hello) context.getBean("hello");
18        assertEquals("Hello Spring", hello.sayHello());
19        hello.print();
20
21
22        Printer printer = (Printer) context.getBean("printer");
23        assertEquals("Hello Spring", printer.toString());
24    }
25
26    @Test
27    public void bean2() {
28        Printer printer = (Printer) context.getBean("printer");
29        Printer printer2 = context.getBean("printer", Printer.class);
30
31        assertSame(printer, printer2);
32    }
33 }
```

jUnit을 사용한 DI 테스트 클래스(HelloBeanJunitTest.java) 실행



| jUnit을 사용한 DI 테스트 클래스(HelloBeanJUnitTest.java) 실행



A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

3. Spring-Test를 사용한 DI 테스트 클래스

| Spring-Test 라이브러리 설치

<http://mvnrepository.com>에 접근한다.

spring-test로 검색한다.

Spring-test 3.2.17 버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>3.2.17.RELEASE</version>
</dependency>
```

| Spring-Test에서 테스트를 지원하는 어노테이션(Annotation) (1)

@RunWith(SpringJUnit4ClassRunner.class)

- @RunWith는 jUnit 프레임워크의 테스트 실행방법을 확장할 때 사용하는 어노테이션이다.
- SpringJUnit4ClassRunner라는 클래스를 지정해주면 jUnit이 테스트를 진행하는 중에 ApplicationContext를 만들고 관리하는 작업을 진행해 준다.
- @RunWith 어노테이션은 각각의 테스트 별로 객체가 생성되더라도 싱글톤(Singleton)의 ApplicationContext를 보장한다.

| Spring-Test에서 테스트를 지원하는 어노테이션(Annotation) (2)

@ContextConfiguration

- 스프링 빈(Bean) 설정 파일의 위치를 지정할 때 사용되는 어노테이션이다.

@Autowired

- 스프링DI에서 사용되는 특별한 어노테이션이다.
- 해당 변수에 자동으로 빈(Bean)을 매핑 해준다.
- 스프링 빈(Bean) 설정 파일을 읽기 위해 굳이 GenericXmlApplicationContext를 사용할 필요가 없다.

Spring-Test를 사용한 DI 테스트 클래스(HelloBeanSpringTest.java)작성

```
*HelloBeanSpringTest.java
1 package myspring.di.xml.test;
2
3 import static org.junit.Assert.*;
4
5
6 @RunWith(SpringJUnit4ClassRunner.class)
7 @ContextConfiguration(locations = "classpath:config/beans.xml")
8 public class HelloBeanSpringTest {
9     @Autowired
10    private ApplicationContext context;
11
12    @Test
13    public void bean1() {
14        Hello hello = (Hello) context.getBean("hello");
15        assertEquals("Hello Spring", hello.sayHello());
16        hello.print();
17        assertEquals(context.getBean("printer").toString(), "Hello Spring");
18
19        Hello hello2 = context.getBean("hello", Hello.class);
20        hello2.print();
21        assertSame(hello,hello2);
22    }
23 }
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [DI 애플리케이션 작성(2)]에 대해서 살펴보았습니다.

jUnit의 개요와 특징

단위테스트를 지원하는 프레임워크, assert 메서드를 사용하여 테스트 결과 확인

jUnit을 사용한 DI 테스트 클래스

- jUnit 설치, asset 메서드를 사용하여 테스트 결과 확인
- @Test, @Before 어노테이션 사용

Spring-Test를 사용한 DI 테스트 클래스

Spring-Test 설치, @RunWith(SpringJUnit4ClassRunner.class),
@ContextConfiguration, @Autowired 어노테이션 사용

Spring Framework

8. DI 애플리케이션 작성(3)

CONTENTS

1

Bean 의존관계 설정 방법

2

프로퍼티(Property) 값 설정 방법

3

프로퍼티(Property) 파일을 이용한 값 설정 방법

학습 목표

- Bean 의존관계 설정 방법에 대하여 이해할 수 있습니다.
- 프로퍼티(Property)값 설정 방법에 대하여 이해할 수 있습니다.
- 프로퍼티(Property) 파일을 이용한 설정 방법에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

1. Bean 의존관계 설정 방법

| Setter Injection : <property> 태그

Setter 메서드를 통해 의존관계가 있는 Bean을 주입하려면 <property> 태그를 사용할 수 있다.

- ref 속성은 사용하면 Bean 이름을 이용해 주입할 Bean을 찾는다.
- value 속성은 단순 값 또는 Bean이 아닌 객체를 주입할 때 사용한다.

| Setter Injection : <property> 태그

```
public class Hello {  
    String name;  
    Printer printer;  
  
    public Hello() { }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public void setPrinter(Printer printer) {  
        this.printer = printer;  
    }  
}
```

```
<bean id="hello" class="myspring.di.xml.Hello">  
    <property name="name" value="Spring" />  
    <property name="printer" ref="printer" />  
</bean>  
  
<bean id="printer"  
      class="myspring.di.xml.StringPrinter" />
```

| Constructor Injection : <constructor-arg> 태그

Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> 태그를 사용할 수 있다.

Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.

```
public class Hello {  
    String name;  
    Printer printer;  
  
    public Hello() {}  
  
    public Hello(String name, Printer printer) {  
        this.name = name;  
        this.printer = printer;  
    }  
}
```

| Constructor Injection : <constructor-arg> 태그

Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> 태그를 사용할 수 있다.

Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.

❖ 생성자 주입을 위한 설정 : index 지정

```
<bean id="hello" class="myspring.di.xml.Hello">
    <constructor-arg index="0" value="Spring"/>
    <constructor-arg index="1" ref="printer"/>
</bean>
```

| Constructor Injection : <constructor-arg> 태그

Constructor를 통해 의존관계가 있는 Bean을 주입하려면 <constructor-arg> 태그를 사용할 수 있다.

Constructor 주입방식은 생성자의 파라미터를 이용하기 때문에 한번에 여러 개의 객체를 주입할 수 있다.

❖ 생성자 주입을 위한 설정 : 파라미터 이름 지정

```
<bean id="hello" class="myspring.di.xml.Hello">
    <constructor-arg name="name" value="Spring"/>
    <constructor-arg name="printer" ref="printer"/>
</bean>
```

I POJO 클래스 수정 및 Bean 설정 파일 수정

```
*Hello.java ✘
1 package myspring.di.xml;
2
3 public class Hello {
4     String name;
5     Printer printer;
6
7     public Hello() {}
8
9     public Hello(String name, Printer printer) {
10         this.name = name;
11         this.printer = printer;
12     }
}
```

```
beans.xml ✘
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/sche
5
6     <bean id="hello2" class="myspring.di.xml.Hello">
7         <constructor-arg index="0" value="Spring" />
8         <constructor-arg index="1" ref="printer" />
9     </bean>
10
11     <bean id="printer"
12         class="myspring.di.xml.StringPrinter" />
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

2. 프로퍼티(Property) 값 설정 방법

| 단순 값(문자열이나 숫자)의 주입(Injection)

Setter 메서드를 통해 Bean의 레퍼런스가 아니라 단순 값을 주입하려고 할 때는 <property> 태그의 value 속성을 사용한다.

```
public class Hello {  
    String name;  
    Printer printer;  
  
    public Hello() {}  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

```
<bean id="hello3" class="myspring.di.xml.Hello">  
    <property name="name" value="스프링" />  
</bean>
```

| 컬렉션(Collection) 타입의 값 주입(Injection) (1)

Spring은 List, Set, Map, Properties와 같은 컬렉션 타입을 XML로 작성해서 프로퍼티에 주입하는 방법을 제공한다.

- ❖ List와 Set 타입 : <list>와 <value> 태그를 이용
- 프로퍼티가 Set 타입 이면 <list> 대신에 <set>을 사용하면 된다.

```
public class Hello {  
    List<String> names;  
  
    public void setNames(List<String> list) {  
        this.names = list;  
    }  
}
```

```
<bean id="hello" class="myspring.di.xml.Hello">  
    <property name="names">  
        <list>  
            <value>Spring</value>  
            <value>IoC</value>  
            <value>DI</value>  
        </list>  
    </property>  
</bean>
```

| 컬렉션(Collection) 타입의 값 주입(Injection) (2)

Spring은 List, Set, Map, Properties와 같은 컬렉션 타입을 XML로 작성해서 프로퍼티에 주입하는 방법을 제공한다.

❖ Map 타입 : <map>과 <entry> 태그를 이용

```
public class Hello {  
    Map<String, Integer> ages;  
  
    public void setAges(Map<String, Integer> ages) {  
        this.ages = ages;  
    }  
}
```

```
<bean id="hello" class="myspring.di.xml.Hello">  
    <property name="ages">  
        <map>  
            <entry key="Kim" value="30" />  
            <entry key="Lee" value="35" />  
            <entry key="Ahn" value="40" />  
        </map>  
    </property>  
</bean>
```

I POJO 클래스 수정 및 Bean 설정 파일 수정

```
Hello.java
1 import java.util.List;
2
3 public class Hello {
4     String name;
5     Printer printer;
6     List<String> names;
7
8     public Hello() {
9     }
10
11    public void setNames(List<String> list) {
12        this.names = list;
13    }
14}
```

```
*beans.xml
1
2
3
4
5
6 <bean id="hello" class="myspring.di.xml.Hello">
7     <property name="name" value="Spring" />
8     <property name="printer" ref="printer" />
9     <property name="names">
10        <list>
11            <value>Spring</value>
12            <value>IoC</value>
13            <value>DI</value>
14        </list>
15    </property>
16 </bean>
```

DI 테스트 클래스 수정

```
1 HelloBeanSpringTest.java ✘
2
3 16 @RunWith(SpringJUnit4ClassRunner.class)
4 17 @ContextConfiguration(locations = "classpath:config/beans.xml")
5 18 public class HelloBeanSpringTest {
6 19     @Autowired
7 20     private ApplicationContext context;
8
9 21
10 22     @Test
11     public void bean1() {
12         Hello hello = (Hello) context.getBean("hello");
13         assertEquals("Hello Spring", hello.sayHello());
14         hello.print();
15
16         assertEquals(3, hello.getNames().size());
17         List<String> list = hello.getNames();
18         for (String value : list) {
19             System.out.println(value);
20         }
21     }
22 }
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

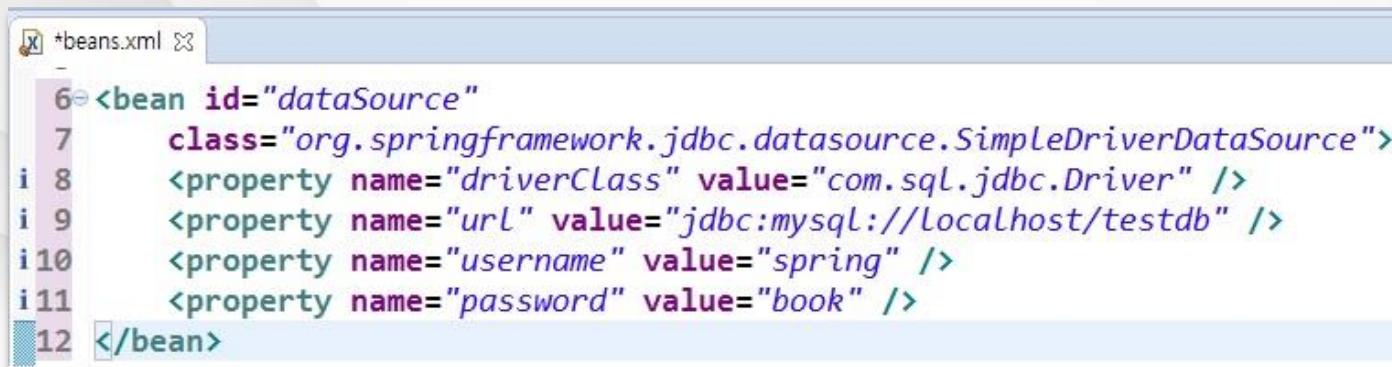
3. 프로퍼티(Property) 파일을 이용한 설정 방법

| 환경에 따라 자주 변경되는 내용의 분리

- XML의 Bean 설정 메타정보는 애플리케이션 구조가 바뀌지 않으면 자주 변경되지 않는다.
- 반면에 프로퍼티 값으로 제공되는 일부 설정정보 (예-DataSource Bean이 사용하는 DB 연결정보)는 애플리케이션이 동작하는 환경(개발, 테스트, 스테이징, 운영)에 따라서 자주 바뀔 수 있다.
- 변경되는 이유와 시점이 다르다면 분리하는 것이 객체지향 설계의 기본 원칙이다. 설정에도 동일한 원칙을 적용할 수 있다.
- 환경에 따라 자주 변경될 수 있는 내용은 properties 파일로 분리하는 것이 가장 깔끔하다 XML처럼 복잡한 구성이 필요 없고 키와 값의 쌍(key=value)으로 구성하면 된다.

| 환경에 따라 자주 변경되는 내용의 분리의 예시(1)

- value 속성에 설정된 값들은 환경에 따라 변경될 수 있는 내용이다.
- 자주 변경되는 값들은 properties 파일에 넣어 분리하는 것이 좋다.

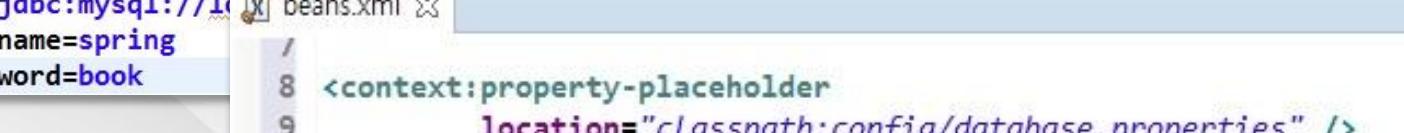


The screenshot shows a code editor window with a tab labeled "beans.xml". The XML code defines a bean named "dataSource" of class "org.springframework.jdbc.datasource.SimpleDriverDataSource". It contains four properties: "driverClass" with value "com.mysql.jdbc.Driver", "url" with value "jdbc:mysql://localhost/testdb", "username" with value "spring", and "password" with value "book". The code is numbered from 6 to 12 on the left.

```
6<bean id="dataSource"
7    class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
i 8        <property name="driverClass" value="com.mysql.jdbc.Driver" />
i 9        <property name="url" value="jdbc:mysql://localhost/testdb" />
i 10       <property name="username" value="spring" />
i 11       <property name="password" value="book" />
12    </bean>
```

■ 환경에 따라 자주 변경되는 내용의 분리의 예시(2)

- 프로퍼티 파일로 분리한 정보는 \${ } (프로퍼티 치환자)을 이용하여 설정한다.
 - \${ } 값을 치환해주는 기능은 <context:property-placeholder> 태그에 의해 자동으로 등록되는 PropertyPlaceHolderConfigurer Bean이 담당한다.



The screenshot shows two open files in an IDE: `database.properties` and `beans.xml`. The `database.properties` file contains the following configuration:

```
1 db.driverClass=com.mysql.jdbc.Driver
2 db.url=jdbc:mysql://localhost:3306/test
3 db.username=spring
4 db.password=book
```

The `beans.xml` file contains the following XML configuration:

```
8<context:property-placeholder
9    location="classpath:config/database.properties" />
10
11<bean id="dataSource"
12    class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
13    <property name="driverClass" value="${db.driverClass}" />
14    <property name="url" value="${db.url}" />
15    <property name="username" value="${db.username}" />
16    <property name="password" value="${db.password}" />
17</bean>
```

3. 프로퍼티(Property) 파일을 이용한 설정 방법

Bean 설정 파일 수정

The image shows a screenshot of an IDE with two code editors side-by-side.

value.properties (Left Editor):

```
1 myname=Spring
2 myprinter=printer
3 value1=JUnit
4 value2=AOP
5 value3=DI
```

beans.xml (Right Editor):

```
7
8 <context:property-placeholder
9     location="classpath:config/value.properties" />
10
11<bean id="hello" class="myspring.di.xml.Hello">
12     <property name="name" value="${myname}" />
13     <property name="printer" ref="${myprinter}" />
14     <property name="names">
15         <list>
16             <value>${value1}</value>
17             <value>${value2}</value>
18             <value>${value3}</value>
19         </list>
20     </property>
21 </bean>
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [DI 애플리케이션 작성(3)]에 대해서 살펴보았습니다.

Bean 의존관계 설정 방법

<property>, <constructor-arg> 태그

프로퍼티(property) 값 설정 방법

- <property> 태그의 value 속성
- <list>, <set>, <map> 태그

프로퍼티(property) 파일을 이용한 값 설정방법

Properties 파일 작성, \${ } 치환자 사용, <context:property-placeholder>

Spring Framework

9. DI 애플리케이션 작성(4)

CONTENTS

1

Bean 등록 메타정보 구성 전략

2

Bean 등록 및 의존관계 설정 Annotation

3

프로퍼티(Property) 파일을 이용한 값 설정 방법

학습 목표

- Bean 등록 메타정보 구성 전략에 대하여 이해할 수 있습니다.
- Bean 등록 및 의존관계 설정 Annotation에 대하여 이해할 수 있습니다.
- 프로퍼티(Property) 파일을 이용한 설정 방법에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, showing several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

1. Bean 등록 메타정보 구성 전략

| 전략(1) XML 단독 사용

- 모든 Bean을 명시적으로 XML에 등록하는 방법이다.
- 생성되는 모든 Bean을 XML에서 확인할 수 있다는 장점이 있으나 Bean의 개수가 많아지면 XML 파일을 관리하기 번거로울 수 있다.
- 여러 개발자가 같은 설정파일을 공유해서 개발하다 보면 설정파일을 동시에 수정하다가 충돌이 일어나는 경우도 적지 않다.
- DI에 필요한 적절한 setter 메서드 또는 constructor가 코드 내에 반드시 존재해야 한다.
- 개발 중에는 어노테이션 설정방법을 사용했지만, 운영 중에는 관리의 편의성을 위해 XML설정으로 변경하는 전략을 쓸 수도 있다.

| 전략(2) XML과 빈 스캐닝 (Bean Scanning)의 혼용

- Bean으로 사용될 클래스에 특별한 어노테이션(Annotation)을 부여해주면 이런 클래스를 자동으로 찾아서 Bean으로 등록한다.
- 특정 어노테이션이 붙은 클래스를 자동으로 찾아서 Bean으로 등록 해주는 방식을 **빈 스캐닝(Bean Scanning)**을 통한 자동인식 Bean 등록기능이라고 한다.
- 어노테이션을 부여하고 자동 스캔으로 빈을 등록하면 XML 문서 생성과 관리에 따른 수고를 덜어주고 개발 속도를 향상시킬 수 있다.
- 애플리케이션에 등록될 Bean이 어떤 것들이 있고, Bean들 간의 의존관계가 어떻게 되는지를 한눈에 파악할 수 없다는 단점이 있다.



2. Bean 등록 및 의존관계 설정 Annotation

I Bean 등록 Annotation

@Component	컴포넌트를 나타내는 일반적인 스테레오 타입으로 <bean> 태그와 동일한 역할을 함
@Repository	퍼시스턴스(persistence) 레이어, 영속성을 가지는 속성(파일, 데이터베이스)을 가진 클래스
@Service	서비스 레이어, 비즈니스 로직을 가진 클래스
@Controller	프리젠테이션 레이어, 웹 어플리케이션에서 웹 요청과 응답을 처리하는 클래스

- ◎ @Repository, @Service, @Controller는 더 특정한 유즈케이스에
대한 @Component의 구체화된 형태이다.

I Bean 의존관계 주입 Annotation (1)

@Autowired, @Resource 어노테이션은 **의존하는** 객체를
자동으로 주입해 주는 어노테이션이다.

@Autowired

- 정밀한 의존관계 주입 (Dependency Injection)이 필요한 경우에 유용하다.
- @Autowired는 프로퍼티, setter 메서드, 생성자, 일반메서드에 적용 가능하다.
- 의존하는 객체를 주입할 때 주로 **Type**을 이용하게 된다.
- @Autowired는 <property>, <constructor-arg> 태그와 동일한 역할을 한다.

I Bean 의존관계 주입 Annotation (2)

@Autowired, @Resource 어노테이션은 **의존하는 객체를 자동으로 주입해 주는 어노테이션**이다.

@Autowired는 타입으로, @Resource는 이름으로 연결한다는 점이 다르다.

@Resource

- 어플리케이션에서 필요로 하는 자원을 자동 연결할 때 사용된다.
- @Resource는 **프로퍼티**, setter 메서드에 적용 가능하다.
- 의존하는 객체를 주입할 때 주로 **Name**을 이용하게 된다.

I Bean 의존관계 주입 Annotation (3)

@Value

- 단순한 값을 주입할 때 사용되는 어노테이션이다.
- @Value("Spring")은 <property .. value="Spring" /> 와 동일한 역할을 한다.

@Qualifier

- @Qualifier는 @Autowired 어노테이션과 같이 사용되어 진다.
- @Autowired는 타입으로 찾아서 주입하므로, 동일한 타입의 Bean 객체가 여러 개 존재할 때 특정 Bean을 찾기 위해서는 @Qualifier를 같이 사용해야 한다.

| Component Scan을 지원하는 태그

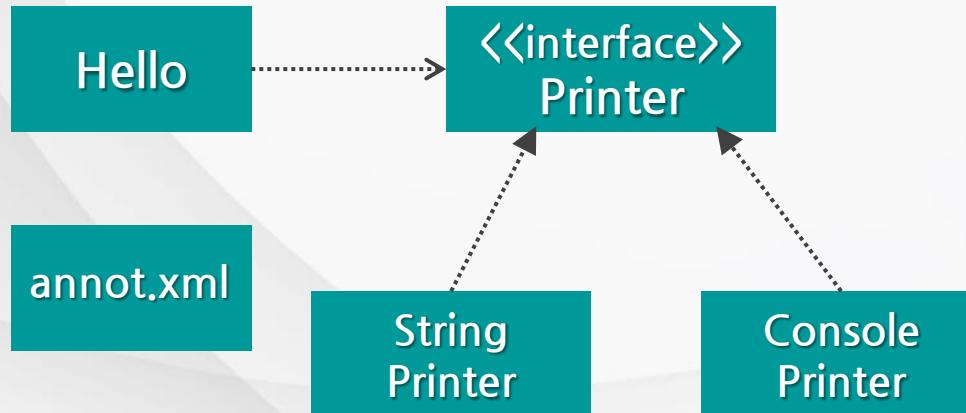
❖ <context:component-scan> 태그

- @Component를 통해 자동으로 Bean을 등록하고,
@Autowired로 의존관계를 주입받는 어노테이션을 클래스에서
선언하여 사용했을 경우에는 해당 클래스가 위치한 특정 패키지를
Scan하기 위한 설정을 XML에 해주어야 한다.

```
<context:component-scan base-package="myspring.di.annot" />
```

- <context:include-filter>태그와 <context:exclude-filter>태그를
같이 사용하면 자동 스캔 대상에 포함시킬 클래스와 포함시키지
않을 클래스를 구체적으로 명시할 수 있다.

| 어노테이션을 사용한 POJO 클래스 작성 (1)



| 어노테이션을 사용한 POJO 클래스 작성 (2)

❖ StringPrinter.java

```
StringPrinter.java ✘
1 package myspring.di.annot;
2
3 import org.springframework.stereotype.Component;
4
5 @Component("stringPrinter")
6 public class StringPrinter implements Printer {
7     private StringBuffer buffer = new StringBuffer();
8
9     public void print(String message) {
10         this.buffer.append(message);
11     }
12
13     public String toString() {
14         return this.buffer.toString();
15     }
16 }
17
```

| 어노테이션을 사용한 POJO 클래스 작성 (3)

❖ ConsolePrinter.java

```
1 package myspring.di.annot;
2
3 import org.springframework.stereotype.Component;
4
5 @Component("consolePrinter")
6 public class ConsolePrinter implements Printer {
7     public void print(String message) {
8         System.out.println(message);
9     }
10 }
```

| 어노테이션을 사용한 POJO 클래스 작성 (4)

❖ Hello.java

```
1 package myspring.di.annot;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6 @Component
7 public class Hello {
8     @Value("Spring")
9     String name;
10
11     @Autowired
12     @Qualifier("stringPrinter")
13     Printer printer;
14
15     public String sayHello() {
16         return "Hello " + name;
17     }
18
19     public void print() {
20         this.printer.print(sayHello());
21     }
22 }
23 }
```

I Bean Configuration (빈 설정) XML 작성

❖ annot.xml

```
annot.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:
4   xmlns:aop="http://www.springframework.org/schema/aop"
5   xsi:schemaLocation="http://www.springframework.org/schema/be
6       http://www.springframework.org/schema/context http://www.
7       http://www.springframework.org/schema/aop http://www.spr
8<
9   <!--어노테이션이 선언된 클래스들을 스캔하기 위한 설정 -->
10  <context:component-scan base-package="myspring.di.annot" />
11
12 </beans>
```

| DI 테스트 클라이언트 수정

❖ HelloBeanSpringTest.java

```
1 package myspring.di.annot.test;
2
3+ import static org.junit.Assert.assertEquals;
4
5 @RunWith(SpringJUnit4ClassRunner.class)
6 @ContextConfiguration(locations = "classpath:config/annot.xml")
7 public class HelloBeanSpringTest {
8     @Autowired
9     private ApplicationContext context;
10
11     @Test
12     public void bean1() {
13         Hello hello = (Hello) context.getBean("hello");
14         assertEquals("Hello Spring", hello.sayHello());
15         hello.print();
16
17         Printer printer = context.getBean("stringPrinter", Printer.class);
18         assertEquals("Hello Spring", printer.toString());
19     }
20 }
```



3. 프로퍼티(Property) 파일을 이용한 설정 방법

Properties 파일 및 Bean 설정파일 작성

```
value.properties
1 myname=Spring
2 myprinter=printer
3 value1=JUnit
4 value2=AOP
5 value3=DI
6 printer1=stringPrinter
7 printer2=consolePrinter|
```

```
annot.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
4   xsi:schemaLocation="http://www.springframework.org/schema/beans
5     http://www.springframework.org/schema/context http://www.
6     http://www.springframework.org/schema/aop http://www.spi
7
8
9   <!--어노테이션이 선언된 클래스들을 스캔하기 위한 설정 -->
10  <context:component-scan base-package="myspring.di.annot" />
11
12  <context:property-placeholder
13    location="classpath:config/value.properties" />
14
15 </beans>
```

| 어노테이션을 사용한 POJO 클래스 수정 (2)

The image shows a screenshot of an IDE with two open files:

- value.properties**: A properties file containing:

```
1 myname=Spring
2 myprinter=printer
3 value1=JUnit
4 value2=AOP
5 value3=DI
⇒ 6 printer1=stringPrinter
7 printer2=consolePrinter|
```
- Hello.java**: A Java class definition:

```
1 package myspring.di.annot;
2
3 import javax.annotation.Resource;□
4
5 @Component
6 public class Hello {
7     @Value("${myname}")
8     String name;
9
10    // @Autowired
11    // @Qualifier("stringPrinter")
12    // @Resource(name="${printer1}")
13    // Printer printer;
14
15    public String sayHello() {
16        return "Hello " + name;
17    }
18
19    public void print() {
20        this.printer.print(sayHello());
21    }
22
23 }
```

The code in the Java file uses annotations to map the properties from the properties file to class members. The line `6 printer1=stringPrinter` in the properties file is highlighted in blue, and the corresponding annotation `@Resource(name="${printer1}")` in the Java code is also highlighted in blue.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [DI 애플리케이션 작성(4)]에 대해서 살펴보았습니다.

Bean 등록 메타정보 구성 전략

XML 단독사용, XML과 빈 스캐닝의 혼용

Bean 등록 및 의존관계 설정 Annotation

- @Component, @Repository, @Service, @Controller
- @Autowired, @Qualifier, @Value, @Resource

프로퍼티(property) 파일을 이용한 값 설정방법

Properties 파일 작성, \${ } 치환자 사용, <context:property-placeholder>

Spring Framework

10. 사용자 관리 프로젝트

CONTENTS

1

사용자 관리 프로젝트 아키텍쳐

2

사용자 관리 프로젝트 클래스 설계

3

사용자 관리 프로젝트 클래스 Code

학습 목표

- 사용자 관리 프로젝트 아키텍쳐에 대하여 이해할 수 있습니다.
- 사용자 관리 프로젝트 클래스 설계에 대하여 이해할 수 있습니다.
- 사용자 관리 프로젝트 클래스 Code에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

1. 사용자 관리 프로젝트 아키텍쳐

| 아키텍쳐 개요

- 대부분의 중·대규모 웹 애플리케이션은 효율적인 개발 및 유지보수를 위하여 계층화(Layering)하여 개발하는 것이 일반적이다.
- 사용자관리 프로젝트 아키텍쳐에서 기본적으로 가지는 계층은 **프리젠테이션 계층(Presentation Layer)**, **서비스 계층(Service Layer)**, **데이터액세스 계층(Data Access Layer)** 3계층과 모든 계층에서 사용되는 **도메인 모델 클래스**로 구성되어 있다.
- 각각의 계층은 계층마다 **독립적으로 분리하여 구현**하는 것이 가능해야 하며, 각 계층에서 담당해야 할 기능들이 있다.

| 아키텍쳐 개요



- 위의 세 가지 계층은 독립적으로 분리할 수 있도록 구현해야 하며, 일반적으로 각 계층 사이에서는 인터페이스(Interface)를 이용하여 통신하는 것이 일반적이다.

| 프리젠테이션 계층

- 브라우저상의 웹클라이언트의 요청 및 응답을 처리
- 상위계층(서비스계층, 데이터 액세스계층)에서 발생하는 Exception에 대한 처리
- 최종 UI에서 표현해야 할 도메인 모델을 사용
- 최종 UI에서 입력한 데이터에 대한 유효성 검증(Validation) 기능을 제공
- 비즈니스 로직과 최종 UI를 분리하기 위한 컨트롤러 기능을 제공
- @Controller 어노테이션을 사용하여 작성된 Controller 클래스가 이 계층에 속함

| 서비스 계층

- 애플리케이션 **비즈니스 로직 처리**와 비즈니스와 관련된 도메인 모델의 적합성 검증
- **트랜잭션(Transaction)** 처리
- 프리젠테이션 계층과 데이터 액세스 계층 사이를 연결하는 역할로서 두 계층이 직접적으로 통신하지 않게 하여 **애플리케이션의 유연성을 증가**
- 다른 계층들과 통신하기 위한 **인터페이스를 제공**
- Service 인터페이스와 @Service 어노테이션을 사용하여 작성된 Service 구현 클래스가 이 계층에 속함

| 데이터 액세스 계층

- 영구 저장소(관계형 데이터베이스)의 데이터를 조작하는 **데이터 액세스 로직을 객체화**
- 영구 저장소의 **데이터를 조회, 등록, 수정, 삭제** 함
- **ORM(Object Relational Mapping) 프레임워크(MyBatis, Hibernate)**를 주로 사용하는 계층
- DAO 인터페이스와 @Repository 어노테이션을 사용하여 작성된 DAO 구현 클래스가 이 계층에 속함

| 도메인 모델 클래스

- 관계형 데이터 베이스의 엔티티와 비슷한 개념을 가지는 것으로
실제 VO(Value Object) 혹은 DTO(Data Transfer Object)
객체에 해당
- 도메인 모델 클래스는 3개의 계층 전체에 걸쳐 사용
- private으로 선언된 멤버변수가 있고, 그 변수에 대한 getter와
setter 메서드를 가진 클래스를 말함

| 테이블 설계

```
CREATE TABLE USERS
(
    userid      VARCHAR2(30) NOT NULL PRIMARY KEY,
    name        VARCHAR2(100) NOT NULL,
    gender      VARCHAR2(10),
    city        VARCHAR2(30)
);
```

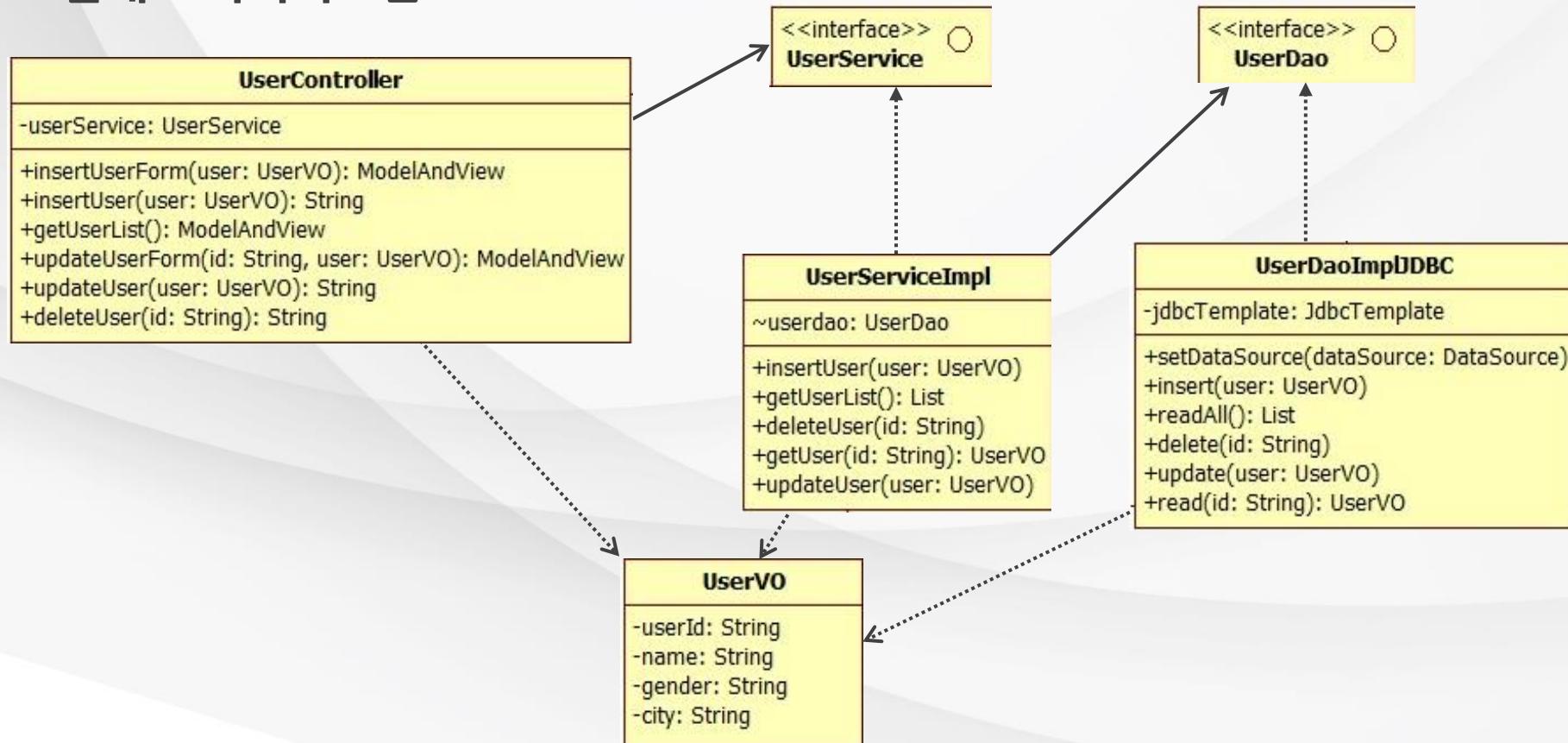
userid : 아이디, name : 이름, gender : 성별, city : 도시명



2. 사용자 관리 프로젝트 클래스 설계

2. 사용자 관리 프로젝트 클래스 설계

클래스 다이어그램



| 클래스의 역할

❖ 프리젠테이션 계층

UserController 클래스

- UI 계층과 서비스 계층을 연결하는 역할을 하는 클래스
- JSP에서 UserController를 통해서 서비스 계층의 UserService를 사용하게 된다.
- 서비스 계층의 UserService 인터페이스를 구현하나 객체를 IoC 컨테이너가 주입해준다.

| 클래스의 역할

❖ 서비스 계층

UserService 인터페이스

- 서비스 계층에 속한 상위 인터페이스

UserServiceImpl 클래스

- UserService 인터페이스를 구현한 클래스
- 복잡한 업무 로직이 있을 경우에는
이 클래스에서 업무 로직을 구현하면 된다.
- 데이터 액세스 계층의 UserDao 인터페이스를 구현한 객체를
IoC 컨테이너가 주입해준다.

| 클래스의 역할

❖ 데이터 액세스 계층

UserDao 인터페이스

- 데이터 액세스 계층에 속한 상위 인터페이스

UserDaoImplJDBC 클래스

- UserDao 인터페이스를 구현한 클래스로 이 클래스에서는 데이터 액세스 로직을 구현하면 된다.
- SpringJDBC를 사용하는 경우에는 DataSource를 IoC 컨테이너가 주입해준다.
- MyBatis를 사용하는 경우에는 SqlSession을 IoC 컨테이너가 주입해준다.



3. 사용자 관리 프로젝트 클래스 Code

UserVO.java

```
*UserVO.java ✎
1 package myspring.user.vo;
2
3 public class UserVO {
4
5     private String userId;
6     private String name;
7     private String gender;
8     private String city;
9
10    public UserVO() {}
11
12    public UserVO(String userId, String name, String gender, String city) {
13        this.userId = userId;
14        this.name = name;
15        this.gender = gender;
16        this.city = city;
17    }
18
19    public String getUserId() {
20        return userId;
21    }
22
23    public void setUserId(String userId) {
24        this.userId = userId;
25    }
```

UserService.java

```
UserService.java ✘  
1 package myspring.user.service;  
2  
3 import java.util.List;  
4  
5 public interface UserService {  
6  
7     public void insertUser(UserVO user);  
8  
9     public List<UserVO> getUserList();  
10  
11    public void deleteUser(String id);  
12  
13    public UserVO getUser(String id);  
14  
15    public void updateUser(UserVO user);  
16  
17 }  
18 }
```

UserServiceImpl.java

```
UserServiceImpl.java ✘
1 package myspring.user.service;
2
3 import java.util.List;
4
5 @Service("userService")
6 public class UserServiceImpl implements UserService {
7
8     @Autowired
9     UserDao userdao;
10
11     @Override
12     public void insertUser(UserVO user) {
13         userdao.insert(user);
14     }
15
16     public List<UserVO> getUserList() {
17         return userdao.readAll();
18     }
19 }
```

UserDao.java

```
 UserDao.java <h1>
1 package myspring.user.dao;
2
3+import java.util.List;
4
5
6
7 public interface UserDao {
8     public void insert(UserVO user);
9
10    public List<UserVO> readAll();
11
12    public void update(UserVO user);
13
14    public void delete(String id);
15
16    public UserVO read(String id);
17
18 }
```

UserDaoImplJDBC.java

```
1 package myspring.user.dao;
2
3 import java.sql.ResultSet;
4
5
6
7 @Repository("userDao")
8 public class UserDaoImplJDBC implements UserDao {
9     private JdbcTemplate jdbcTemplate;
10
11
12     @Autowired
13     public void setDataSource(DataSource dataSource) {
14         this.jdbcTemplate = new JdbcTemplate(dataSource);
15     }
16
17
18     class UserMapper implements RowMapper<UserVO> {
19         public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
20             UserVO user = new UserVO();
21             user.setUserId(rs.getString("userid"));
22             user.setName(rs.getString("name"));
23             user.setGender(rs.getString("gender"));
24             user.setCity(rs.getString("city"));
25
26         return user;
27     }
28
29
30
31
32
33 }
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [사용자 관리 프로젝트]에 대해서 살펴보았습니다.

사용자 관리 프로젝트 아키텍처

프리젠테이션 계층, 서비스 계층, 데이터액세스 계층, 도메인 클래스

사용자 관리 프로젝트 클래스 설계

클래스 다이어그램, 각 클래스들의 역할

사용자 관리 프로젝트 클래스 Code

각 클래스들의 Code 살펴보기

Spring Framework

11. Spring JDBC 개요

CONTENTS

1

데이터 액세스 공통 개념

2

Spring JDBC 개요

3

Spring JDBC의 JdbcTemplate 클래스

학습 목표

- 데이터 액세스 공통 개념에 대하여 이해할 수 있습니다.
- Spring JDBC 개념에 대하여 이해할 수 있습니다.
- Spring JDBC의 JdbcTemplate 클래스에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and filled with blurred, colorful circular lights, suggesting a night-time urban environment.

1. 데이터 액세스 공통 개념

I DAO(Data Access Object) 패턴

- 데이터 액세스 계층은 DAO 패턴을 적용하여 **비즈니스 로직과 데이터 액세스 로직을 분리**하는 것이 원칙이다.
- 비즈니스 로직이 없거나 단순하면 DAO와 서비스 계층을 통합 할 수도 있지만 의미 있는 비즈니스 로직을 가진 엔터프라이즈 애플리케이션이라면 데이터 액세스 계층을 DAO 패턴으로 분리해야 한다.
- DAO패턴은 서비스계층에 영향을 주지 않고 **데이터 액세스 기술을 변경할 수 있는 장점을 가지고 있다.**

| 컨넥션 풀링을 지원하는 DataSource

컨넥션 풀링은 미리 정해진 개수만큼의 DB 컨넥션을 풀(Pool)에 준비해두고, 애플리케이션이 요청할 때마다 Pool에서 꺼내서 하나씩 할당해주고 다시 돌려받아서 Pool에 넣는 식의 기법이다.

- 다중 사용자를 갖는 엔터프라이즈 시스템에서라면 반드시 DB 컨넥션 풀링 기능을 지원하는 DataSource를 사용해야 한다.
- Spring에서는 DataSource를 공유 가능한 Spring Bean으로 등록해 주어 사용할 수 있도록 해준다.

DataSource 구현 클래스 종류

❖ 테스트환경을 위한 DataSource

SimpleDriverDataSource

- Spring이 제공하는 가장 단순한 DataSource 구현 클래스이다.
- getConnection()을 호출할 때마다 매번 DB 커넥션을 새로 만들고
따로 풀(pool)을 관리하지 않으므로 단순한 테스트용으로만
사용해야 한다.

SingleConnectionDriverDataSource

- 순차적으로 진행되는 통합 테스트에서는 사용 가능하다.
- 매번 DB 커넥션을 생성하지 않기 때문에
SimpleDriverDataSource 보다 빠르게 동작한다.

DataSource 종류

❖ 오픈소스 DataSource

Apache Commons DBCP

- ◎ 가장 유명한 **오픈소스 DB 커넥션 풀(pool)** 라이브러리이다.
- ◎ Apache의 Commons 프로젝트(<http://commons.apache.org/dbcp/>)

c3p0 JDBC/DataSource Resource Pool

- ◎ c3p0는 JDBC 3.0 스펙을 준수하는 Connection과 Statement 풀(pool)을 제공하는 라이브러리이다.
 - ◎ c3p0 웹 사이트(<http://www.mchange.com/projects/c3p0/>)
- * 두 가지 모두 수정자(setter) 메서드를 제공하므로 Spring Bean으로 등록해서 사용하기 편리하다.

A photograph of a person's hands holding a black smartphone. The phone's screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

2. Spring JDBC 개요

I JDBC란?

JDBC는 모든 자바의 데이터 액세스 기술의 근간이 된다.
엔티티 클래스와 애노테이션을 이용하는 최신 ORM 기술도
내부적으로는 DB와의 연동을 위해 JDBC를 이용한다.

- 안정적이고 유연한 기술이지만, **로우 레벨 기술**로 인식되고 있다.
- 간단한 SQL을 실행하는 데도 **중복된 코드가 반복적으로 사용되며**, DB에 따라 일관성 없는 정보를 가진 채로 Checked Exception으로 처리한다.

장점

대부분의 개발자가 잘 알고 있는 친숙한 데이터 액세스 기술로 **별도의 학습 없이 개발이 가능하다.**

단점

Connection과 같은 공유 리소스를 제대로 릴리즈 해주지 않으면 **시스템의 자원이 바닥나는 버그를 발생시킨다.**

I Spring JDBC란?

JDBC의 장점과 단순성을 그대로 유지하면서도 기존 JDBC의 단점을 극복할 수 있게 해주고, 간결한 형태의 API 사용법을 제공하며, JDBC API에서 지원되지 않는 편리한 기능을 제공한다.

- Spring JDBC는 반복적으로 해야 하는 많은 작업들을 대신 해준다.
- Spring JDBC를 사용할 때는 실행할 SQL과 바인딩 할 파라미터를 넘겨 주거나, 쿼리의 실행 결과를 어떤 객체에 넘겨 받을지를 지정하는 것만 하면 된다.
- Spring JDBC를 사용하려면 먼저, DB 커넥션을 가져오는 **DataSource**를 Bean으로 등록해야 한다.

| Spring JDBC가 해주는 작업(1)

Connection 열기와 닫기

- Connection과 관련된 모든 작업을 Spring JDBC가 필요한 시점에서 알아서 진행한다.
- 진행 중에 예외가 발생했을 때도 열린 모든 Connection 객체를 닫아준다.

Statement 준비와 닫기

- SQL 정보가 담긴 Statement 또는 PreparedStatement를 생성하고 필요한 준비 작업을 해주는 것도 Spring JDBC가 한다.
- Statement도 Connection과 마찬가지로 사용이 끝나고 나면 Spring JDBC가 알아서 객체를 닫아준다.

| Spring JDBC가 해주는 작업(2)

Statement 실행

- SQL 담긴 Statement를 실행하는 것도 Spring JDBC가 해준다.
- Statement의 실행결과는 다양한 형태로 가져올 수 있다.

ResultSet Loop 처리

- ResultSet에 담긴 쿼리 실행 결과가 한 건 이상이면 ResultSet 루프를 만들어서 반복해주는 것도 Spring JDBC가 해주는 작업이다.

| Spring JDBC가 해주는 작업(3)

Exception 처리와 반환

- JDBC 작업 중 발생하는 모든 예외는 Spring JDBC 예외 변환기가 처리한다.
- 체크 예외(Checked Exception)인 SQLException을 런타임 예외(Runtime Exception)인 DataAccessException 타입으로 변환한다.

Transaction 처리

- Spring JDBC를 사용하면 transaction과 관련된 모든 작업에 대해서는 신경 쓰지 않아도 된다.



3. Spring JDBC의 JdbcTemplate 클래스

I JdbcTemplate 클래스

Spring JDBC가 제공하는 클래스 중 JdbcTemplate은 JDBC의 모든 기능을 최대한 활용할 수 있는 **유연성**을 제공하는 클래스이다.

- JdbcTemplate이 제공하는 기능은 **실행**, **조회**, **배치**의 세 가지 작업이다.

- **실행** : Insert나 Update같이 DB의 데이터에 변경이 일어나는 쿼리를 수행하는 작업
- **조회** : Select를 이용해 데이터를 조회하는 작업
- **배치** : 여러 개의 쿼리를 한 번에 수행해야 하는 작업

I JdbcTemplate 클래스 생성

- JdbcTemplate은 DataSource를 파라미터로 받아서 아래와 같이 생성할 수 있다.

```
JdbcTemplate template = new JdbcTemplate(dataSource);
```

- DataSource는 보통 Bean으로 등록해서 사용하므로 JdbcTemplate이 필요한 DAO 클래스에서 DataSource Bean을 DI(의존관계 주입) 받아서 JdbcTemplate을 생성할 때 인자로 넘겨주면 된다.
- JdbcTemplate은 멀티스레드 환경에서도 안전하게 공유해서 쓸 수 있기 때문에 DAO클래스의 인스턴스 변수에 저장해 두고 사용할 수 있다.

I JdbcTemplate 클래스 생성 Code

- 아래의 코드는 일반적으로 사용되는 DAO 클래스의 기본구조이다.
DataSource에 대한 수정자 메서드에서 직접 JdbcTemplate 객체를
생성해준다.

```
public class UserDAOJdbc {  
    JdbcTemplate jdbcTemplate;  
  
    @Autowired  
    public void setDataSource(DataSource dataSource) {  
        jdbcTemplate = new JdbcTemplate(dataSource);  
    }  
    .....  
}
```

I JdbcTemplate 클래스의 update() 메서드

- INSERT, UPDATE, DELETE와 같은 SQL을 실행할 때는 JdbcTemplate의 update() 메서드를 사용한다.

```
int update(String sql, [SQL 파라미터])
```

- update() 메서드를 호출할 때는 SQL과 함께 바인딩 할 파라미터는 Object 타입 가변인자 (Object ... args)를 사용할 수 있다.
- update() 메서드의 리턴되는 값은 SQL 실행으로 영향을 받은 레코드의 개수를 리턴한다.

I JdbcTemplate 클래스의 update() 메서드 Code

```
public int update(User user) {  
    StringBuffer updateQuery = new StringBuffer();  
    updateQuery.append("UPDATE USERS SET ");  
    updateQuery.append("password=?, name=? ");  
    updateQuery.append("WHERE id=? ");  
  
    int result = this.jdbcTemplate.update(updateQuery.toString(),  
        user.getName(),user.getPassword(),user.getId());  
  
    return result;  
}
```

I JdbcTemplate 클래스의 queryForObject() 메서드

- SELECT SQL을 실행하여 하나의 Row를 가져올 때는 JdbcTemplate의 queryForObject() 메서드를 사용한다.

```
<T> T queryForObject(String sql, [SQL 파라미터],  
RowMapper<T> rm)
```

- SQL 실행 결과는 여러 개의 칼럼(Column)을 가진 하나의 로우(Row)
- T는 VO 객체의 타입에 해당된다.
- SQL 실행 결과로 돌아온 여러 개의 column을 가진 한 개의 Row를 RowMapper 콜백을 이용해 VO 객체로 매팅 해준다.

JdbcTemplate 클래스의 queryForObject() 메서드 Code

```
public User findUser (String id) {+
    return this.jdbcTemplate.queryForObject("select * from users
    where id=?",new Object[] {id},+
    new RowMapper<User>() {+
        public User mapRow(ResultSet rs, int rowNum)+{
            throws SQLException{+
                User user = new User();+
                user.setId(rs.getString("id"));+
                user.setName(rs.getString("name"));+
                user.setPassword(rs.getString("password"));+
                return user;+
            }+
        }//RowMapper+
    );//queryForObject+
};//findUser+
```

I JdbcTemplate 클래스의 query() 메서드

- SELECT SQL을 실행하여 여러 개의 Row를 가져올 때는 JdbcTemplate의 query() 메서드를 사용한다.

```
<T> List<T> query(String sql, [SQL 파라미터],  
RowMapper<T> rm)
```

- SQL 실행 결과로 돌아온 여러 개의 column을 가진 여러 개의 Row를 RowMapper 콜백을 이용해 VO 객체로 매팅 해준다.
- 결과 값은 매팅 한 VO 객체를 포함하고 있는 List 형태로 받는다. List의 각 요소가 하나의 Row에 해당된다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring JDBC 개요]에 대해서 살펴보았습니다.

데이터 액세스 공통개념

DAO 패턴, 커넥션 풀링, DataSource

Spring JDBC 개요

- JDBC 개요 및 장단점
- Spring JDBC 개요 및 역할

Spring JDBC 의 JdbcTemplate 클래스

JdbcTemplate 클래스의 update(), queryForObject(), query() 메서드

Spring Framework

12. Spring JDBC 환경설정

CONTENTS

1

DB 설정 및 JDBC Driver 설치

2

Spring JDBC 설치 및 DataSource 설정

3

사용자관리 프로젝트 테스트

학습 목표

- DB 설정 및 JDBC Driver 설치에 대하여 이해할 수 있습니다.
- Spring JDBC 설치 및 DataSource 설정에 대하여 이해할 수 있습니다.
- 사용자관리 프로젝트 테스트에 대하여 이해할 수 있습니다.



1. DB 설정 및 JDBC Driver 설치

I DBA 권한으로 접속

SQL Command 실행



DBA권한으로 접속

A screenshot of the 'Run SQL Command Line' window. It shows the SQL*Plus welcome screen: 'SQL*Plus: Release 11.2.0.2.0 Production' and 'Copyright (c) 1982, 2010, Oracle. All rights reserved.' Below this, the SQL prompt 'SQL>' is followed by the command 'conn sys as sysdba;'. The response 'Enter password:' is shown, followed by 'Connected.' and another SQL prompt 'SQL>'.

Password는 설치할 때 지정한 oracle11

I DB scott 계정 생성

scott 계정 생성

```
SQL> create user scott identified by tiger  
  2 default tablespace users  
  3 temporary tablespace temp;
```

```
User created.
```

scott 계정에 권한 주기

```
SQL> grant connect, resource to scott;
```

```
Grant succeeded.
```

■ DB users 테이블 생성

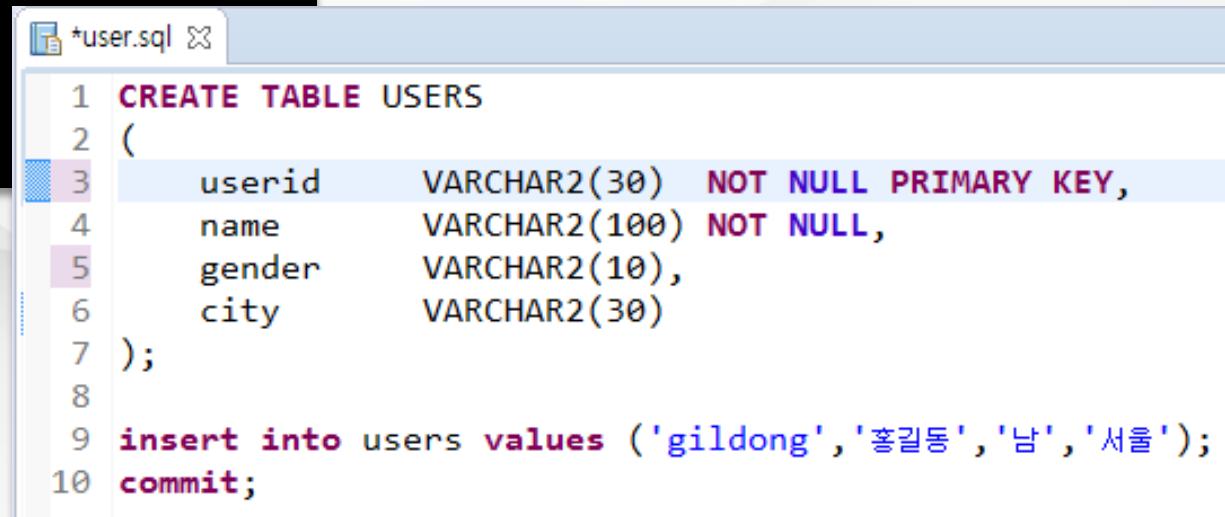
scott 계정으로 접속 후 users 테이블 생성

```
SQL> conn scott/tiger  
Connected.  
SQL> start c:\springframework\user.sql;
```

Table created.

1 row created.

Commit complete.



The screenshot shows a code editor window titled "user.sql". The code is as follows:

```
1 CREATE TABLE USERS  
2 (  
3     userid      VARCHAR2(30) NOT NULL PRIMARY KEY,  
4     name        VARCHAR2(100) NOT NULL,  
5     gender      VARCHAR2(10),  
6     city        VARCHAR2(30)  
7 );  
8  
9 insert into users values ('gildong', '홍길동', '남', '서울');  
10 commit;
```

| Oracle Jdbc Driver 라이브러리 검색 및 설치

<http://mvnrepository.com>에 접근한다.

oracle ojdbc6로 검색한다.

oracle jdbc 12.1.0.1 버전을 pom.xml에 추가한다.

Oracle Jdbc Driver 라이브러리 검색 및 설치

The screenshot shows the Maven Repository search interface. In the search bar, the query "oracleojdbc6" is entered. Below the search bar, the text "Found 195 results" is displayed. A specific result is highlighted with a red border: "2. Oracle JDBC Driver For Java Oracle.jdbc.driver.OracleDriver Ojdbc6". This result is associated with the group ID "com.hynnet" and the artifact ID "oracle-driver-ojdbc6". The full name of the dependency is "Oracle JDBC Driver For Java Oracle.jdbc.driver.OracleDriver Ojdbc6".

```
<!-- https://mvnrepository.com/artifact/com.hynnet/oracle-driver-ojdbc6 -->
<dependency>
    <groupId>com.hynnet</groupId>
    <artifactId>oracle-driver-ojdbc6</artifactId>
    <version>12.1.0.1</version>
</dependency>
```



2. Spring JDBC 설치 및 DataSource 설정

| Spring JDBC 설치

<http://mvnrepository.com>에 접근한다.

spring jdbc로 검색한다.

spring jdbc 3.2.17 버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>3.2.17.RELEASE</version>
</dependency>
```

DataSource 설정

- DataSource를 Spring Bean으로 등록하여 사용할 수 있다.

```
*beans.xml
8 <context:property-placeholder
9     location="classpath:config/value.properties" />
10
11<bean id="dataSource"
12     class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
13     <property name="driverClass" value="${db.driverClass}" />
14     <property name="url" value="${db.url}" />
15     <property name="username" value="${db.username}" />
16     <property name="password" value="${db.password}" />
17 </bean>
```

```
value.properties
1 db.driverClass=oracle.jdbc.OracleDriver
2 db.url=jdbc:oracle:thin:@127.0.0.1:1521:orcl
3 db.username=scott
4 db.password=tiger
```

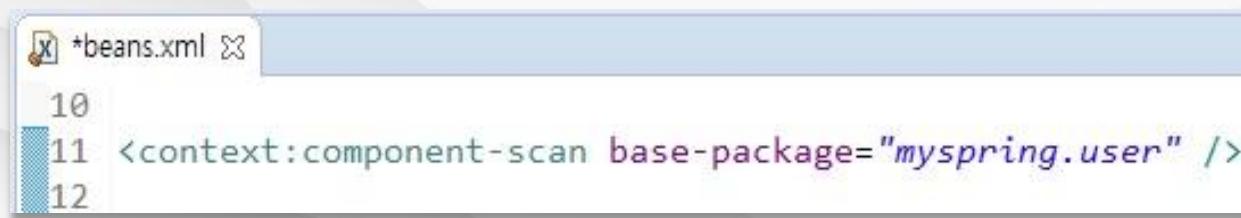
A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

3. 사용자관리 프로젝트 실행

| 사용자관리 프로젝트의 Bean 등록 및 의존관계 설정

❖ <context:component-scan> 태그 사용

- @Service, @Repository 어노테이션을 선언한 클래스들과
@Autowired 어노테이션을 선언하여 의존관계를 설정한 클래스들이
위치한 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.



The screenshot shows a code editor window with a file named "beans.xml". The code in the editor is as follows:

```
10
11 <context:component-scan base-package="myspring.user" />
12
```

DataSource 설정 테스트

```
beans.xml
26<bean id="dataSource"
27    class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
28    <property name="driverClass" value="${db.driverClass}" />
29    <property name="url" value="${db.url}" />
30    <property name="username" value="${db.username}" />
31    <property name="password" value="${db.password}" />
32 </bean>
```

```
public class UserClient {
    @Test
    public void dataSourceTest() {
        DataSource ds = (DataSource) context.getBean("dataSource");
        try {
            System.out.println(ds.getConnection());
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

실행 결과

oracle.jdbc.driver.T4CConnection@19d31f3

| 사용자 조회 테스트

```
35  
36 @Autowired  
37 UserService service;  
38  
39 @Test  
40 public void getUserTest() {  
41     UserVO user = service.getUser("gildong");  
42     System.out.println(user);  
43     assertEquals("홍길동", user.getName());  
44 }  
45 }
```

실행 결과

```
Markers Console  
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin\javaw.e  
User [userId=gildong, name=홍길동, gender=남, city=서울]
```

| 사용자 등록 및 목록조회 테스트

```
UserClient.java ✘
35
36    @Autowired
37    UserService service;
38
39    @Test
40    public void insertUserTest() {
41        service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
42
43        for (UserVO user : service.getUserList()) {
44            System.out.println(user);
45        }
46    }
```

실행 결과

The screenshot shows the Eclipse IDE interface with the JUnit perspective selected. The top bar includes tabs for Markers, Console, Maven Repositories, JUnit (which is selected), Properties, and Data. The console output window displays the results of the executed test:

```
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin\javaw.exe
등록된 Record UserId=dooly Name=둘리
User [userId=dooly, name=둘리, gender=남, city=경기]
User [userId=gildong, name=홍길동, gender=남, city=서울]
```

| 사용자 정보수정 테스트

```
UserClient.java
36 @Autowired
37 UserService service;
38
39 @Test
40 public void updateUserTest() {
41     service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
42
43     UserVO user = service.getUser("dooly");
44     System.out.println(user);
45 }
```

실행 결과

```
Markers Console Maven Repositories JUnit Properties
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin\
갱신된 Record with ID = dooly
User [userId=dooly, name=김둘리, gender=여, city=부산]
```

| 사용자 정보삭제 테스트

```
UserClient.java
36     @Autowired
37     UserService service;
38
39     @Test
40     public void deleteUserTest() {
41         service.deleteUser("dooly");
42
43         for (UserVO user : service.getUserList()) {
44             System.out.println(user);
45         }
46     }
```

실행 결과

```
Markers Console Maven Repositories JUnit Properties Data
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin\java
삭제된 Record with ID = dooly
User [userId=gildong, name=홍길동, gender=남, city=서울]
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring JDBC 환경설정]에 대해서 살펴보았습니다.

DB 설정 및 JDBC Driver 설치

- DB 접속 계정 생성 및 테이블 생성
- Oracle JDBC Driver 설치

Spring JDBC 설치 및 DataSource 설정

- Spring JDBC 라이브러리 설치
- DataSource 설정

사용자관리 프로젝트 테스트

- 사용자 관리 프로젝트 실행
- 사용자 조회, 등록, 목록조회, 수정, 삭제 테스트

Spring Framework

13. AOP 개요

CONTENTS

- 1 AOP의 개요와 용어
- 2 Spring AOP의 특징 및 구현방식
- 3 AspectJ와 Spring AOP 라이브러리 설치

학습 목표

- AOP의 개요와 용어에 대해 이해할 수 있습니다.
- Spring AOP의 특징 및 구현방식에 대해 이해할 수 있습니다.
- AspectJ 와 Spring AOP 설치에 대해 이해할 수 있습니다.

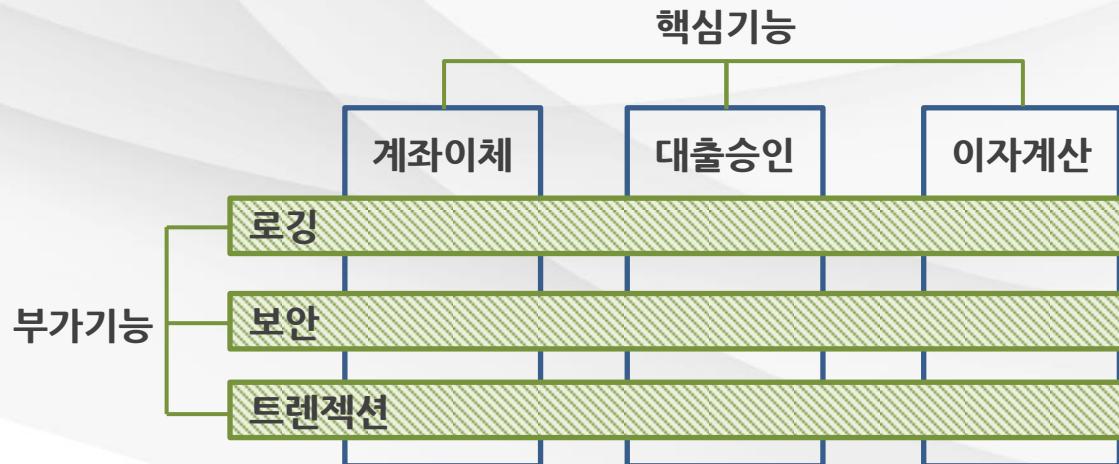
A photograph of a person's hands holding a smartphone at night. The phone screen is illuminated, and the background is filled with blurred, colorful lights in shades of yellow, orange, and blue, creating a bokeh effect.

AOP는

1. AOP의 개요와 용어

| 핵심기능과 부가기능

- 업무(Biz) 로직을 포함하는 기능을 **핵심 기능(Core Concerns)**
- 핵심기능을 도와주는 부가적인 기능(로깅, 보안 등)을 **부가기능 (Cross-cutting Concerns)** 이라고 부른다.
- 객체지향의 기본 원칙을 적용하여도 핵심기능에서 부가기능을 분리해서 모듈화하는 것은 매우 어렵다.



I AOP(Aspect Oriented Programming)의 개요

AOP는 애플리케이션에서의 **관심사의 분리(기능의 분리)** 즉, 핵심적인 기능에서 부가적인 기능을 분리한다. 분리한 부가기능을 **애패스트(Aspect)**라는 독특한 모듈형태로 만들어서 설계하고 개발하는 방법

- OOP를 적용하여도 핵심기능에서 부가기능을 **쉽게 분리된 모듈로 작성하기 어려운 문제점을** AOP가 해결해 준다고 볼 수 있다.
- AOP는 부가기능을 애패스트(Aspect)로 정의하여, **핵심기능에서 부가기능을 분리함으로써 핵심기능을 설계하고 구현할 때 객체지향적인 가치**를 지킬 수 있도록 도와주는 개념이다.

| 애스펙트(Aspect)

- 애스펙트는 부가기능을 정의한 코드인 **어드바이스(Advice)**와 어드바이스를 어디에 적용하지를 결정하는 **포인트컷(PointCut)**을 합친 개념이다.

Advice + PointCut = Aspect

- AOP 개념을 적용하면 핵심기능 코드 사이에 침투된 부가기능을 독립적인 애스펙트로 구분해 낼 수 있다.
- 구분된 부가기능 애스펙트를 런타임 시에 필요한 위치에 동적으로 참여하게 할 수 있다.

| AOP 용어

타겟(Target)

- ◎ 핵심기능을 담고 있는 모듈로, 타겟은 부가기능을 부여할 대상이 된다.

어드바이스(Advice)

- ◎ 어드바이스는 타겟에 제공할 부가기능을 담고 있는 모듈이다.

조인 포인트(Join Point)

- ◎ 어드바이스가 적용될 수 있는 위치를 말한다.
- ◎ 즉, 타겟 객체가 구현한 인터페이스의 모든 메서드는 조인 포인트가 된다.

| AOP 용어

포인트 컷(Pointcut)

- 어드바이스를 적용할 타겟의 메서드를 선별하는 정규표현식이다.
- 포인트컷 표현식은 execution으로 시작하고, 메서드의 Signature를 비교하는 방법을 주로 이용한다.

애플렉트(Aspect)

- 애플렉트는 AOP의 기본 모듈이다.
- 애플렉트 = 어드바이스 + 포인트컷
- 애플렉트는 싱글톤 형태의 객체로 존재한다.

| AOP 용어

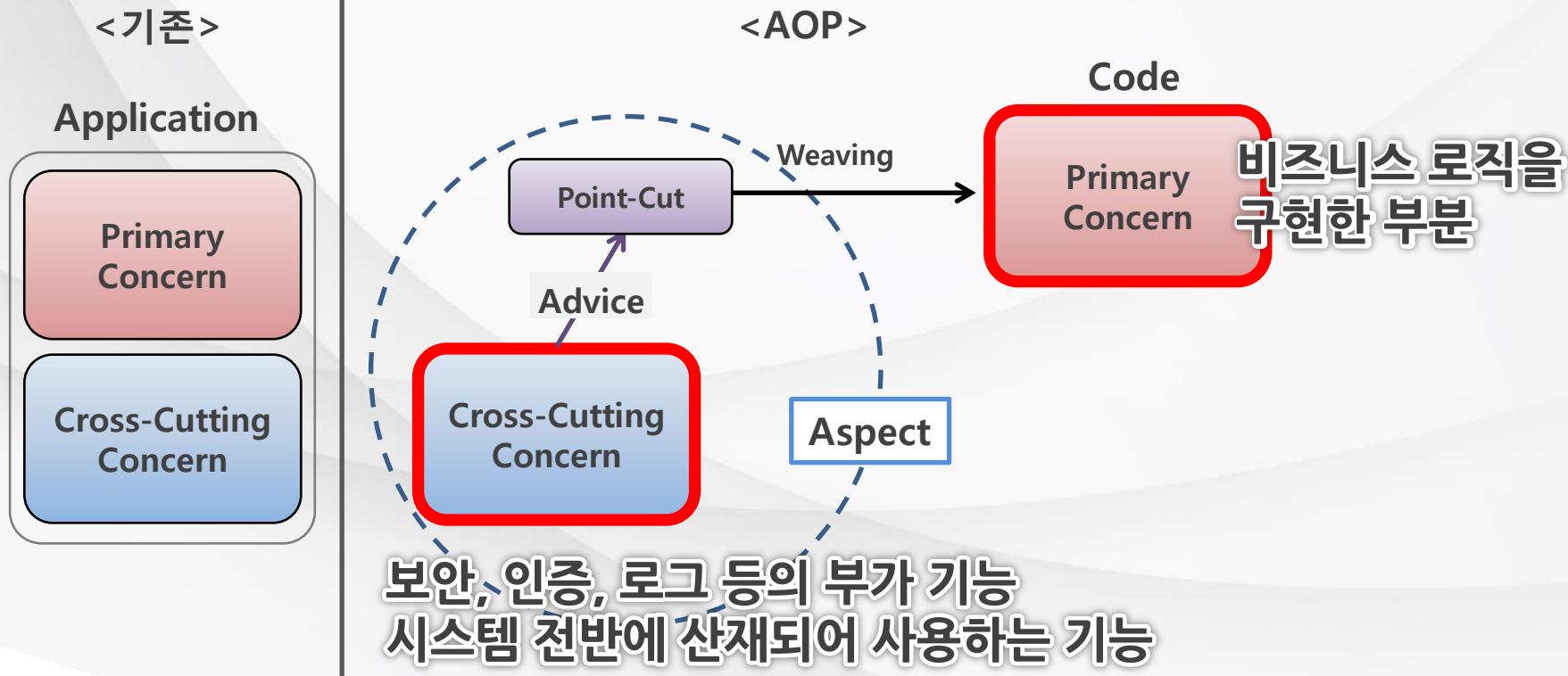
어드바이저(Advisor)

- 어드바이저 = 어드바이스 + 포인트컷
- 어드바이저는 Spring AOP에서만 사용되는 특별한 용어이다.

위빙(Weaving)

- 위빙은 포인트컷에 의해서 결정된 타겟의 조인 포인트에 부가기능(어드바이스)을 삽입하는 과정을 뜻한다.
- 위빙은 AOP가 핵심기능(타겟)의 코드에 영향을 주지 않으면서 필요한 부가기능(어드바이스)을 추가할 수 있도록 해주는 핵심적인 처리과정이다.

I AOP 용어정리



A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, showing blurred lights in various colors (yellow, orange, blue) that suggest a city at night or a well-lit indoor space.

2. Spring AOP의 특징 및 구현방식

| Spring AOP의 특징

❖ (1) Spring은 프록시(Proxy) 기반 AOP를 지원한다.

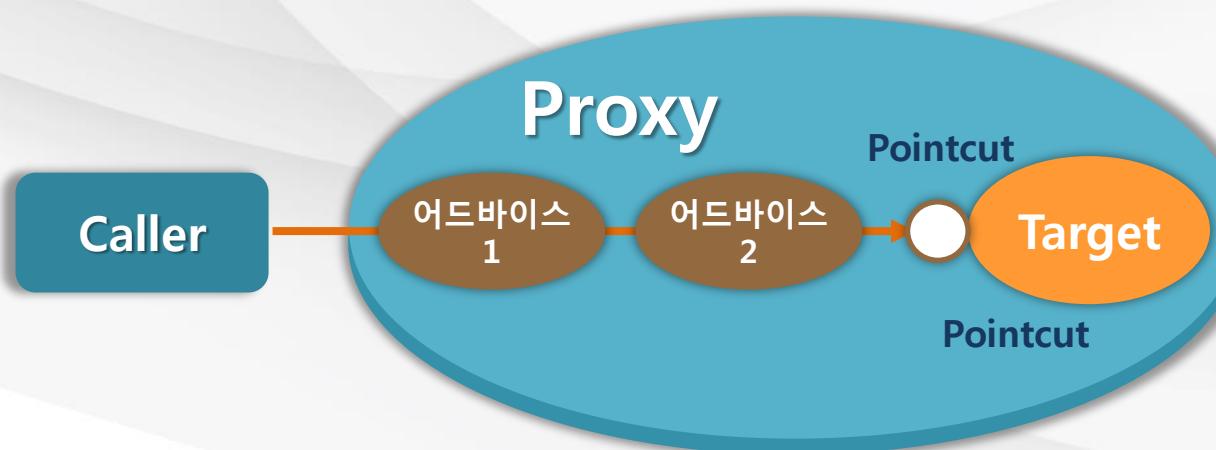
- Spring은 타겟(target) 객체에 대한 프록시를 만들어 제공한다.
- 타겟을 감싸는 프록시는 실행시간(Runtime)에 생성된다.
- 프록시는 어드바이스를 타겟 객체에 적용하면서 생성되는 객체이다.



| Spring AOP의 특징

❖ (2) 프록시(Proxy)가 호출을 가로챈다(Intercept).

- 프록시는 타겟 객체에 대한 호출을 가로챈 다음 어드바이스의 부가기능 로직을 수행하고 난 후에 타겟의 핵심기능 로직을 호출한다. (전처리 어드바이스)
- 또는 타겟의 핵심기능 로직 메서드를 호출한 후에 부가기능(어드바이스)을 수행하는 경우도 있다.(후처리 어드바이스)



| Spring AOP의 특징

❖ (3) Spring AOP는 메서드 조인 포인트만 지원한다.

- ◎ Spring은 동적 프록시를 기반으로 AOP를 구현하므로
메서드 조인 포인트만 지원한다. 즉, 핵심기능(타겟)의 메서드가
호출되는 런타임 시점에만 부가기능(어드바이스)을 적용할 수 있다.
- ◎ 반면에 AspectJ 같은 고급 AOP 프레임워크를 사용하면
객체의 생성, 필드값의 조회와 조작, static 메서드 호출 및
초기화 등의 다양한 작업에 부가기능을 적용할 수 있다.

| Spring AOP의 구현 방식

XML 기반의 POJO 클래스를 이용한 AOP 구현

- 부가기능을 제공하는 Advice 클래스를 작성한다.
- XML 설정 파일에 <aop:config>를 이용해서 애스펙트를 설정한다.
(즉, 어드바이스와 포인트컷을 설정함)

@Aspect 어노테이션을 이용한 AOP 구현

- @Aspect 어노테이션을 이용해서 부가기능을 제공하는 Aspect 클래스를 작성한다. 이때 Aspect 클래스는 어드바이스를 구현하는 메서드와 포인트컷을 포함한다.
- XML 설정 파일에 <aop:aspectj-autoproxy />를 설정한다.



3. AspectJ와 Spring AOP 라이브러리 설치

I AspectJ Runtime 라이브러리 설치

<http://mvnrepository.com>에 접근한다.

aspectj runtime로 검색한다.

aspectj runtime 1.7.4 버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>1.7.4</version>
</dependency>
```

I AspectJ Weaver 라이브러리 설치

<http://mvnrepository.com>에 접근한다.

aspectj weaver로 검색한다.

aspectj weaver 1.7.4 버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.7.4</version>
</dependency>
```

| Spring AOP 라이브러리 설치

<http://mvnrepository.com>에 접근한다.

spring aop로 검색한다.

spring aop 3.2.17 버전을 pom.xml에 추가한다.

```
<!-- http://mvnrepository.com/artifact/org.springframework/spring-aop -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>3.2.17.RELEASE</version>
</dependency>
```

I AspectJ Runtime API 문서

Google aspectj runtime api doc

전체 동영상 뉴스 이미지 지도 더보기 ▾ 검색 도구

[Overview \(AspectJ\(tm\) runtime API\) - Eclipse](#)
eclipse.org/aspectj/doc/next/runtime-api/ ▾ 이 페이지 번역하기
org.aspectj.lang, Provides several interfaces for obtaining reflective information about a join point, as well as several exceptions that can be thrown by AspectJ ...

JoinPoint public interface JoinPoint. Provides reflective access to ...	frames The Overview page is the front page of this API document and ...
JoinPoint.StaticPart Interface JoinPoint.StaticPart. All Known Subinterfaces: JoinPoint.	Signature public interface Signature. Represents the signature at a ...
ProceedingJoinPoint ProceedingJoinPoint exposes the	MethodSignature org.aspectj.lang.reflect. Interface

I AspectJ Runtime API 문서

URL

<http://www.eclipse.org/aspectj/doc/next/runtime-api/>

The screenshot shows a web browser displaying the AspectJ Runtime API documentation. The address bar contains the URL <http://www.eclipse.org/aspectj/doc/next/runtime-api/>. In the top right corner of the browser window, there is a yellow star icon inside a red square. A red callout box with the Korean text "Browser의 즐겨찾기에 추가함" (Add to Favorites) points to this star icon. The main content area shows navigation tabs for Overview, Package, Class, Tree, Deprecated, Index, PREV, NEXT, FRAMES, and NO FRAMES. Below these tabs, a section titled "Packages" lists three packages: org.aspectj.lang, org.aspectj.lang.reflect, and org.aspectj.runtime.reflect. Each package entry includes a brief description.

Package	Description
org.aspectj.lang	Provides several interfaces for aspect weaving, join point, as well as several utility classes.
org.aspectj.lang.reflect	Contains interfaces that enable aspects to query about each possible join point.
org.aspectj.runtime.reflect	

Browser의
즐겨찾기에 추가함

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [AOP 개요]에 대해서 살펴보았습니다.

AOP의 개요와 용어

- ◎ 핵심기능(타겟)과 부가기능(어드바이스)의 분리, 포인트컷
- ◎ 조인포인트, 위빙, 애스펙트, 어드바이저

Spring AOP의 특징 및 구현방식

- ◎ Spring은 프록시 기반의 AOP, 메서드 조인 포인트만 지원
- ◎ AOP 구현방식은 XML 기반, 어노테이션 기반

AspectJ와 Spring AOP 라이브러리 설치

- ◎ Aspectj runtime, Aspectj weaver 설치
- ◎ Spring AOP 설치, Aspectj runtime api 문서 찾기

Spring Framework

14. AOP 어플리케이션 작성(1)

CONTENTS

- 1 Advice 클래스 작성
- 2 AOP 설정 및 테스트
- 3 PointCut 표현식

학습 목표

- Advice 클래스 작성에 대하여 이해할 수 있습니다.
- AOP 설정 및 테스트에 대하여 이해할 수 있습니다.
- PointCut 표현식에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

1. Advice 클래스 작성

| Spring AOP의 구현 방식

01 XML 기반의 POJO 클래스를 이용한 AOP 구현

- 부가기능을 제공하는 Advice 클래스를 작성한다.
- XML 설정 파일에 <aop:config>를 이용해서 애스펙트를 설정한다.
(즉, 어드바이스와 포인트컷을 설정함)

02 @Aspect 어노테이션을 이용한 AOP 구현

- @Aspect 어노테이션을 이용해서 부가기능을 제공하는 Aspect 클래스를 작성한다. 이때 Aspect 클래스는 어드바이스를 구현하는 메서드와 포인트컷을 포함한다.
- XML 설정 파일에 <aop:aspectj-autoproxy />를 설정한다.

| Advice의 종류

Around 어드바이스	<ul style="list-style-type: none">▪ 타겟의 메서드가 호출되기 이전(before) 시점과 이후(after) 시점에 모두 처리해야 할 필요가 있는 부가기능을 정의한다. → Joinpoint 앞과 뒤에서 실행되는 Advice
Before 어드바이스	<ul style="list-style-type: none">▪ 타겟의 메서드가 실행되기 이전(before) 시점에 처리해야 할 필요가 있는 부가기능을 정의한다. → Joinpoint 앞에서 실행되는 Advice
After Returning 어드바이스	<ul style="list-style-type: none">▪ 타겟의 메서드가 정상적으로 실행된 이후(after) 시점에 처리해야 할 필요가 있는 부가기능을 정의한다. → Jointpoint 메서드 호출이 정상적으로 종료된 뒤에 실행되는 Advice
After Throwing 어드바이스	<ul style="list-style-type: none">▪ 타겟의 메서드가 예외를 발생된 이후(after) 시점에 처리해야 할 필요가 있는 부가기능을 정의한다. → 예외가 던져질 때 실행되는 Advice

| Advice 클래스 정보

- **클래스명** : PerformanceTraceAdvice.java
- **클래스 기능** : 이 어드바이스는 타겟 객체의 메서드 실행 시간을 계산해서 출력해주는 부가기능을 제공한다.
- **Advice 유형** : Around 어드바이스
(타겟 객체의 메서드 실행 전, 후의 시간을 측정하여 계산하면 타겟 객체의 메서드 실행 시간을 알 수 있음)
- **구현 메서드명** : trace(ProceedingJoinPoint joinPoint)

| JoinPoint 인터페이스

- ◎ JoinPoint는 Spring AOP 혹은 AspectJ에서 AOP가 적용되는 지점을 뜻한다.
- ◎ 해당 지점을 AspectJ에서 JoinPoint라는 인터페이스로 나타낸다.
- ◎ JoinPoint 인터페이스는 getArgs() (메서드 아규먼트를 반환한다),
getThis() (프록시 객체를 반환한다), getTarget() (대상 객체를 반환한다),
getSignature() (어드바이즈 되는 메서드의 설명(description)을 반환한다),
toString() (어드바이즈 되는 메서드의 설명을 출력한다)과 같은 다수의
유용한 메서드를 제공한다.
- ◎ 모든 어드바이스는 org.aspectj.lang.JoinPoint 타입의 파라미터를
어드바이스 메서드에 첫 번째 매개변수로 선언할 수 있다.
- ◎ Around 어드바이스는 JoinPoint의 하위 클래스인
ProceedingJoinPoint 타입의 파라미터를 필수적으로 선언해야 한다.

JoinPoint 인터페이스

org.aspectj.lang

Interface JoinPoint

All Known Subinterfaces:

[ProceedingJoinPoint](#)

public interface **JoinPoint**

Provides reflective access to both the state available at a join point and static information about it. This information is available from the body of advice using the special form `thisJoinPoint`. The primary use of this reflective information is for tracing and logging applications.

Method Summary

java.lang.Object[]	getArgs() Returns the arguments at this join point.
<u>Signature</u>	getSignature() Returns the signature at the join point.
java.lang.Object	getTarget() Returns the target object.
java.lang.String	toString()

I ProceedingJoinPoint 인터페이스

org.aspectj.lang

Interface ProceedingJoinPoint

All Superinterfaces:

[JoinPoint](#)

```
public interface ProceedingJoinPoint  
extends JoinPoint
```

ProceedingJoinPoint exposes the proceed(..) method in order to support around advice in @AJ aspects

Method Summary

java.lang.Object	proceed() Proceed with the next advice or target method invocation
java.lang.Object	proceed(java.lang.Object[] args) Proceed with the next advice or target method invocation The given args Object[] must be in the same order and size as the advice signature but without the actual joinpoint instance

PerformanceTraceAdvice.java

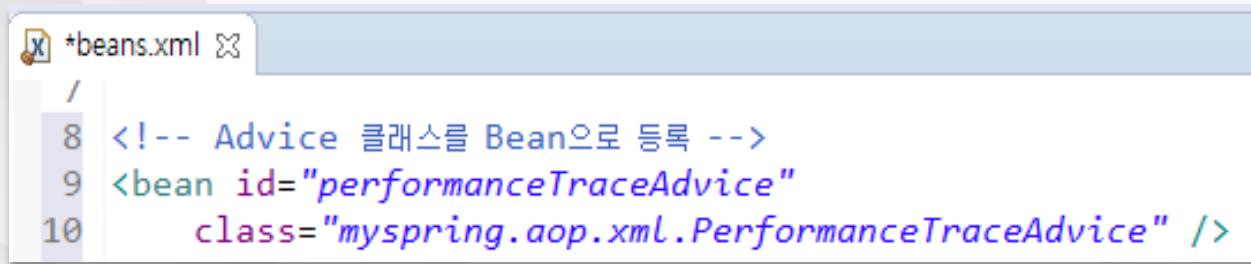
```
*PerformanceTraceAdvice.java ✘
1 package myspring.aop.xml;
2 import org.aspectj.lang.ProceedingJoinPoint;
3
4 public class PerformanceTraceAdvice {
5     public Object trace(ProceedingJoinPoint joinPoint) throws Throwable {
6         //타겟 메서드의 signature 정보
7         String signatureString = joinPoint.getSignature().toShortString();
8         System.out.println(signatureString + " 시작");
9         //타겟의 메서드가 호출되기 전의 시간
10        long start = System.currentTimeMillis();
11        try {
12            //타겟의 메서드 호출
13            Object result = joinPoint.proceed();
14            return result;
15        } finally {
16            //타겟의 메서드가 호출된 후의 시간
17            long finish = System.currentTimeMillis();
18            System.out.println(signatureString + " 종료");
19            System.out.println(signatureString + " 실행 시간 : " +
20                (finish - start) + " ms");
21        }
22    }
23 }
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

2. AOP 설정 및 테스트

| Advice 클래스를 Bean으로 등록

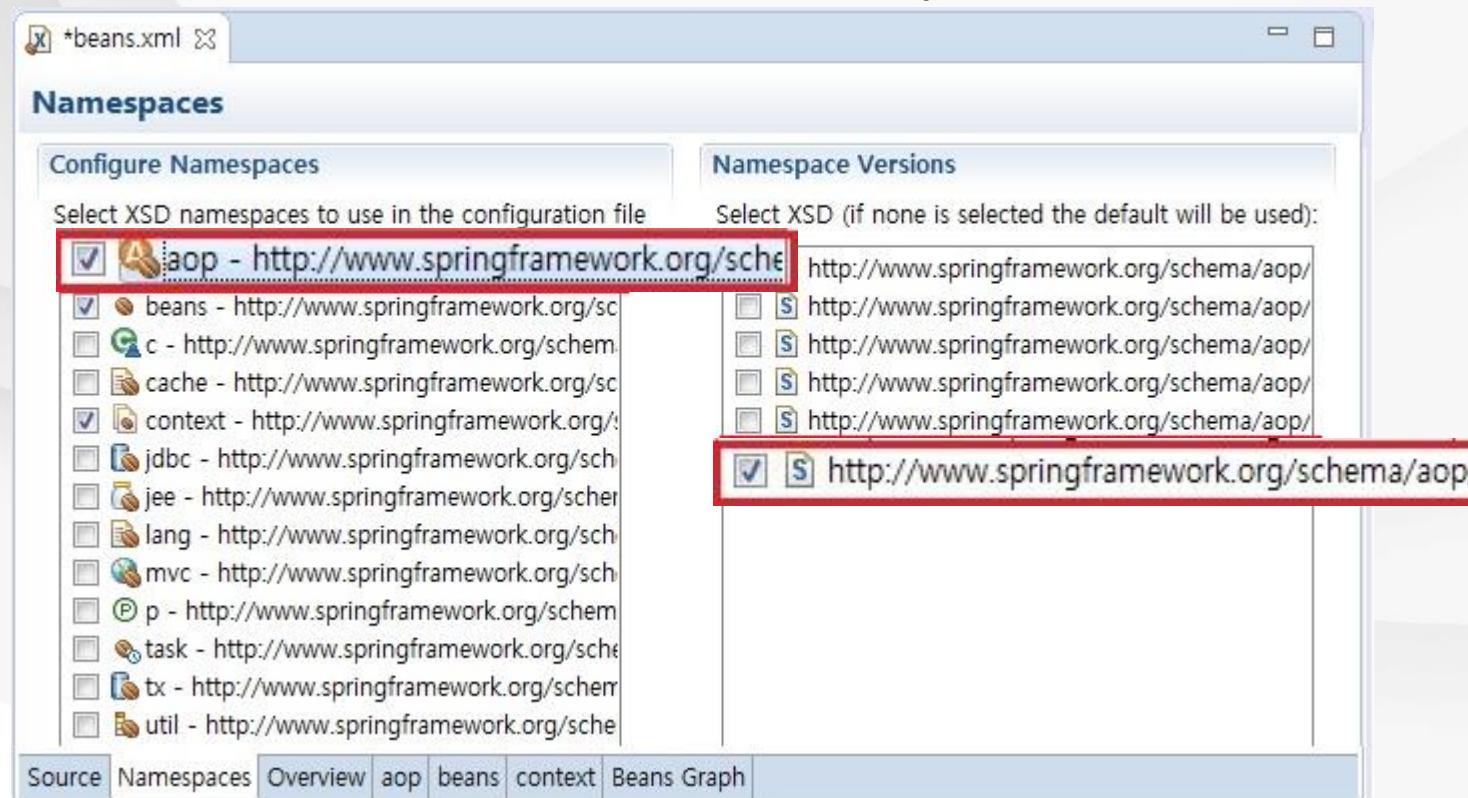
- 작성한 Advice 클래스를 XML 설정파일에 Bean으로 등록해야 한다.



```
*beans.xml
8 <!-- Advice 클래스를 Bean으로 등록 -->
9 <bean id="performanceTraceAdvice"
10   class="myspring.aop.xml.PerformanceTraceAdvice" />
```

I AOP 네임스페이스 추가

- 설정 파일에 AOP와 관련된 내용을 설정하려면 aop 네임스페이스를 추가해 주어야 한다.



I AOP 설정

- <aop:config> : AOP 설정 정보임을 나타낸다.
- <aop:aspect> : 애스펙트를 설정한다.
- <aop:around pointcut="execution()"> : Around 어드바이스와 포인트컷을 설정한다.



The screenshot shows a code editor window with the file name "beans.xml". The code is an XML configuration for AOP, specifically defining an aspect and its advice.

```
beans.xml
10<!-- AOP 설정 -->
11<aop:config>
12  <aop:aspect id="traceAspect" ref="performanceTraceAdvice">
13    <aop:around pointcut="execution(public * myspring.user.service..*(...))"
14                  method="trace" />
15  </aop:aspect>
16</aop:config>
17
18<!-- Advice 클래스를 Bean으로 등록 -->
19<bean id="performanceTraceAdvice"
20      class="myspring.aop.xml.PerformanceTraceAdvice" />
```

The code defines an aspect named "traceAspect" which contains an around advice. The advice is triggered on any public method in any class under the package "myspring.user.service". The advice itself is named "trace". Below this, a bean is defined with the id "performanceTraceAdvice" and the class "myspring.aop.xml.PerformanceTraceAdvice".

I AOP 설정에 대한 설명

- <aop:aspect> 태그의 ref 속성은 애스팩트로서 기능을 제공할 Bean을 설정할 때 사용함
- <aop:around> 태그의 pointcut 속성의 execution 지시자(designator)는 어드바이스를 적용할 패키지, 클래스, 메서드를 표현할 때 사용됨
- myspring.user.service 패키지 및 그 하위 패키지에 있는 모든 public 메서드를 포인트컷으로 설정하고 있음
- UserServiceImpl의 public 메서드가 호출될 때 PerformanceTraceAdvice Bean의 trace() 메서드가 호출 되도록 설정하고 있음

```
<!-- AOP 설정 -->
<aop:config>
    <aop:aspect id="traceAspect" ref="performanceTraceAdvice">
        <aop:around pointcut="execution(public * myspring.user.service..*(..))"
                     method="trace" />
    </aop:aspect>
</aop:config>

<!-- Advice 클래스를 Bean으로 등록 -->
<bean id="performanceTraceAdvice"
      class="myspring.aop.xml.PerformanceTraceAdvice" />
```

I Around Advice와 AOP 설정 테스트

- UserService Bean의 메서드를 호출하면, Around Advice가 적용된 것을 확인해 볼 수 있다.

The screenshot shows an IDE interface with two main panes. The top pane displays the `UserClient.java` file, which contains Java code for testing a `UserService` bean. The bottom pane shows the JUnit Console output, which includes the test results and the execution logs for the `UserService.getUser` method.

```
22 public class UserClient {
23     @Autowired
24     ApplicationContext context;
25
26     @Test
27     public void getUserTest() {
28         service = context.getBean(UserService.class);
29         UserVO user = service.getUser("gildong");
30         System.out.println(user);
31         assertEquals("홍길동", user.getName());
32     }
}
```

Markers Console Maven Repositories JUnit Properties Data

<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin\javaw

```
UserService.getUser(..) 시작
UserService.getUser(..) 종료
UserService.getUser(..) 실행 시간 : 1919 ms
User [userId=gildong, name=홍길동, gender=남, city=서울]
```

| Advice를 정의하는 태그

- 각 타입의 Advice를 정의하기 위해 아래와 같은 태그를 제공한다.

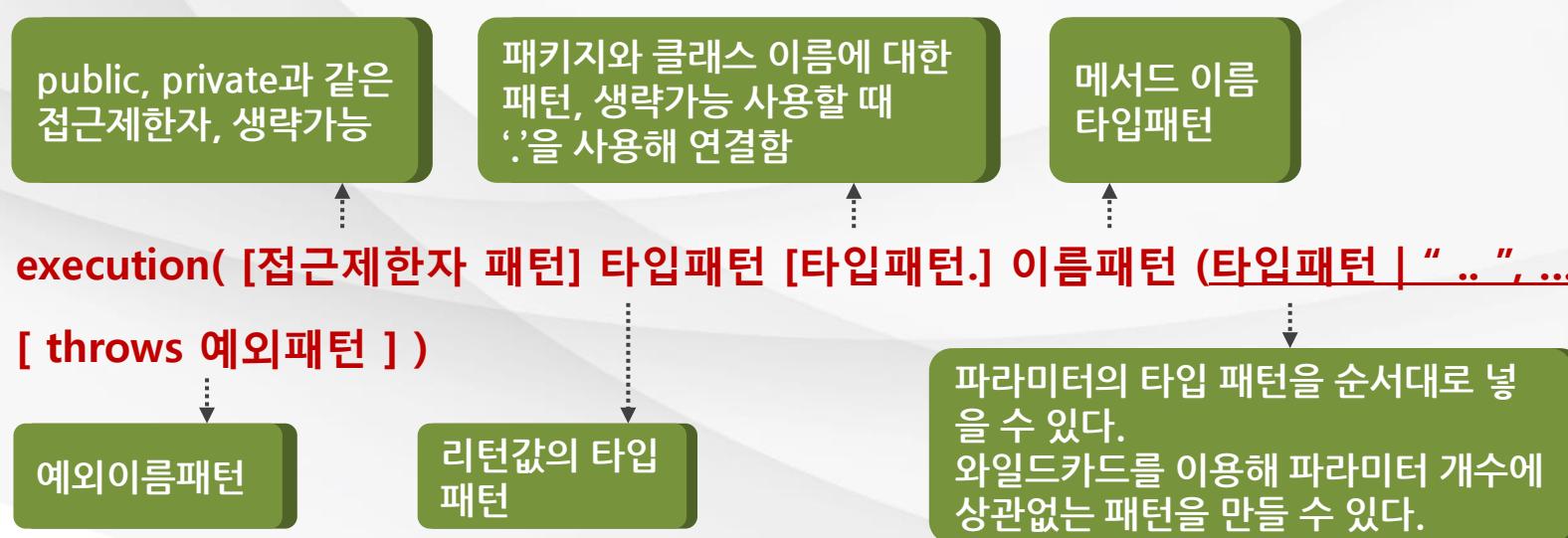
<aop:before>	<ul style="list-style-type: none">메서드 실행 전에 적용되는 어드바이스를 정의한다.
<aop:after-returning>	<ul style="list-style-type: none">메서드가 정상적으로 실행된 후에 적용되는 어드바이스를 정의한다.
<aop:after-throwing>	<ul style="list-style-type: none">메서드가 예외를 발생시킬 때 적용되는 어드바이스를 정의한다. try-catch 블록에서 catch 블록과 비슷하다.
<aop:after>	<ul style="list-style-type: none">메서드가 정상적으로 실행되는지 또는 예외를 발생시키는지 여부에 상관없이 어드바이스를 정의한다.try-catch-finally에서 finally 블록과 비슷하다.
<aop:around>	<ul style="list-style-type: none">메서드 호출 이전, 이후, 예외발생 등 모든 시점에 적용 가능한 어드바이스를 정의한다.

A photograph of a person's hands holding a black smartphone. The person is wearing a dark long-sleeved shirt. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

3. PointCut 표현식

PointCut 표현식 문법

- AspectJ 포인트컷 표현식은 포인트컷 지시자를 이용하여 작성한다.
- 포인트컷 지시자 중에서 가장 대표적으로 사용되는 것은 execution()이다.
- execution() 지시자를 사용한 포인트컷 표현식의 문법구조는 다음과 같다.



PointCut 표현식 예시

Any return type

package

class

method

Any type and number
of arguments

"execution(* aspects.trace.demo.*.*(..))"

01 execution(* hello(..))

- ◎ hello라는 이름을 가진 메서드를 선정하는 것이다.
파라미터는 모든 종류를 다 허용한다.

02 execution(* hello())

- ◎ 파라미터 패턴이 ()로 되어 있으니
hello 메서드 중에서 파라미터가 없는 것만 선택한다.

PointCut 표현식 예시

Any return type

package

class

method

Any type and number
of arguments

"execution(* aspects.trace.demo.*.*(..))"

03

execution(* myspring.user.service.UserServiceimpl.*(..))

- myspring.user.service.UserServiceimpl 클래스를 직접 지정하여 이 클래스가 가진 모든 메서드를 선택한다.

04

execution(* myspring.user.service.*.*(..))

- myspring.user.service 패키지의 모든 클래스에 적용된다.
하지만 서브패키지의 클래스는 포함되지 않는다.

PointCut 표현식 예시

Any return type

package

class

method

Any type and number
of arguments

"execution(* aspects.trace.demo.*.*(..))"

05 execution(* myspring.user.service..*.*(..))

- myspring.user.service 패키지의 모든 클래스에 적용된다.
그리고 '..'를 사용해서 서브패키지의 모든 클래스까지 포함한다.

06 execution(* *..Target.*(..))

- 패키지에 상관없이 Target이라는 이름의 모든 클래스에 적용된다.
다른 패키지에 같은 이름의 클래스가 있어도 적용이 된다는 점에
유의해야 함

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [AOP 어플리케이션 작성(1)]에 대해서 살펴보았습니다.

Advice 클래스 작성

Advice 종류, PointCut 인터페이스, PerformanceTraceAdvice.java

AOP설정 및 테스트

- Advice 클래스를 Bean으로 등록
- <aop:config> <aop:aspect> <aop:around> 설정

PointCut 표현식

- execution() 지시자
- execution(리턴타입패턴 패키지패턴 메서드패턴(파라미터패턴))

Spring Framework

15. AOP 어플리케이션 작성(2)

CONTENTS

1

Aspect 클래스 선언 및 설정

2

Aspect 클래스 구현

3

Aspect 클래스 테스트

학습 목표

- Aspect 클래스 선언 및 설정에 대하여 이해할 수 있습니다.
- Aspect 클래스 구현에 대하여 이해할 수 있습니다.
- Aspect 클래스 테스트에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

1. Aspect 클래스 선언 및 설정

| Spring AOP의 구현 방식

01 XML 기반의 POJO 클래스를 이용한 AOP 구현

- 부가기능을 제공하는 Advice 클래스를 작성한다.
- XML 설정 파일에 <aop:config>를 이용해서 애스펙트를 설정한다.
(즉, 어드바이스와 포인트컷을 설정함)

02 @Aspect 어노테이션을 이용한 AOP 구현

- @Aspect 어노테이션을 이용해서 부가기능을 제공하는 Aspect 클래스를 작성한다. 이때 Aspect 클래스는 어드바이스를 구현하는 메서드와 포인트컷을 포함한다.
- XML 설정 파일에 <aop:aspectj-autoproxy />를 설정한다.

| @Aspect 어노테이션

- Aspect 클래스 선언할 때 @Aspect 어노테이션을 사용한다.
- AspectJ 5버전에 새롭게 추가된 어노테이션이다.
- @Aspect 어노테이션을 이용할 경우 XML 설정 파일에 어드바이스와 포인트컷을 설정하는 것이 아니라 클래스 내부에 정의할 수 있다.
- <aop:aspectj-autoproxy> 태그를 설정파일에 추가하면
@Aspect 어노테이션이 적용된 Bean을 Aspect로 사용 가능하다.

| Aspect 클래스 정보

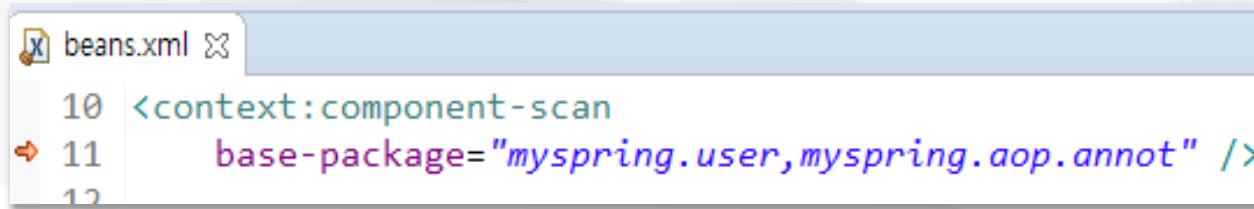
- ◉ **클래스명** : LoggingAspect.java
- ◉ **클래스 기능** : 이 Aspect 클래스는 4가지 유형의 어드바이스와 포인트컷을 설정하여 타겟 객체의 파라미터와 리턴값, 예외 발생 시 예외 메시지를 출력하는 기능을 제공
- ◉ **Advice 유형** : Before, AfterReturning, AfterThrowing, After
- ◉ **구현 메서드명** : before(JoinPoint joinPoint)
afterReturning(JoinPoint joinPoint, Object ret)
afterThrowing(JoinPoint joinPoint, Throwable ex)
afterFinally(JoinPoint joinPoint)

I Aspect 클래스 선언 및 설정

1. 클래스 선언부에 **@Aspect** 어노테이션을 정의한다.
2. 이 클래스를 애스펙트로 사용하려면 Bean으로 등록해야 하므로 **@Component** 어노테이션도 함께 정의한다.

```
package myspring.aop.annot;

import org.aspectj.lang.JoinPoint;
@Component
@Aspect
public class LoggingAspect {
```



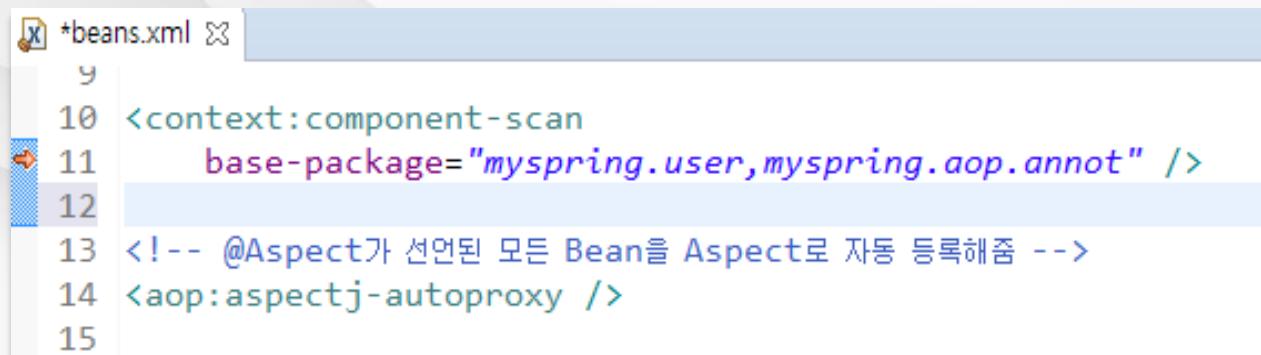
The screenshot shows a code editor window with a file named "beans.xml". The XML code defines a component scan with a base package:

```
<context:component-scan
    base-package="myspring.user,myspring.aop.annot" />
```

I Aspect 클래스 선언 및 설정

3. XML 설정파일에 <aop:aspectj-autoproxy /> 선언한다.

이 선언은 Bean으로 등록된 클래스 중에서 @Aspect가 선언된 클래스를 모두 애스펙트로 자동 등록 해주는 역할을 한다.



The screenshot shows a code editor window with the file name "*beans.xml". The code content is as follows:

```
9
10 <context:component-scan
11     base-package="myspring.user,myspring.aop.annot" />
12
13 <!-- @Aspect가 선언된 모든 Bean을 Aspect로 자동 등록해줌 -->
14 <aop:aspectj-autoproxy />
15
```

The line 14, which contains the XML tag for automatic proxying, is highlighted with a blue selection bar. A small orange arrow icon is positioned to the left of the line number 11, pointing towards the highlighted code.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

2. Aspect 클래스 구현

I Advice를 정의하는 어노테이션

- Advice를 정의하기 위하여 아래와 같은 어노테이션을 제공한다.

@Before("pointcut")

- 타겟 객체의 메서드가 실행되기 전에 호출되는 어드바이스
- JoinPoint를 통해 파라미터 정보를 참조할 수 있다.

@After("pointcut")

- 타겟 객체의 메서드가 정상 종료됐을 때와 예외가 발생했을 때 모두 호출되는 어드바이스
- 리턴값이나 예외를 직접 전달받을 수는 없다.

@Around("pointcut")

- 타겟 객체의 메서드가 호출되는 전 과정을 모두 담을 수 있는 가장 강력한 기능을 가진 어드바이스

I Advice를 정의하는 어노테이션

- Advice를 정의하기 위하여 아래와 같은 어노테이션을 제공한다.

`@AfterReturning(pointcut="",
returning="")`

- 타겟 객체의 메서드가 정상적으로 실행을 마친 후에 호출되는 어드바이스
- 리턴값을 참조할 때는 returning 속성에 리턴값을 저장할 변수 이름을 지정해야 한다.

`@AfterThrowing(pointcut="",
throwing="")`

- 타겟 객체의 메서드가 예외가 발생하면 호출되는 어드바이스
- 발생된 예외를 참조할 때는 throwing 속성에 발생한 예외를 저장할 변수 이름을 지정해야 한다.

I Before 어드바이스

- ◎ @Before 어드바이스를 이용해서 실행되는 타겟 객체의 메서드명과 파라미터를 출력하는 어드바이스이다.
- ◎ 아래의 before 메서드는 myspring 패키지 또는 그 하위 패키지에 있는 모든 public 메서드가 호출되기 이전에 호출된다.

```
15    @Before("execution(public * myspring..*(..))")
16    public void before(JoinPoint joinPoint) {
17        String signatureString = joinPoint.getSignature().getName();
18
19        System.out.println("@Before [ " + signatureString + " ] 메서드 실행 전처리 수행");
20
21        for (Object arg : joinPoint.getArgs()) {
22            System.out.println("@Before [ " + signatureString + " ] 아규먼트 " + arg);
23        }
24    }
```

I AfterReturning 어드바이스

- ◎ @AfterReturning 어드바이스를 이용해서 실행되는 타겟 객체의 메서드명과 리턴값을 출력하는 어드바이스이다.
- ◎ 아래의 afterReturning 메서드는 myspring.user.service 패키지 하위에 있는 모든 public 메서드가 정상 종료된 이후에 호출된다.
- ◎ 리턴값을 참조할 때는 returning 속성을 이용해서 리턴 값을 담을 변수 이름을 지정해야 한다.

```
26@ AfterReturning(pointcut="execution(public * myspring.user.service.*.*(..))", returning="ret")
27public void afterReturning(JoinPoint joinPoint, Object ret) {
28    String signatureString = joinPoint.getSignature().getName();
29    System.out.println("@AfterReturing [ " + signatureString + " ] 메서드 실행 후처리 수행");
30    System.out.println("@AfterReturing [ " + signatureString + " ] 리턴값= " + ret);
31}
32}
```

I AfterThrowing 어드바이스

- ◎ @AfterThrowing 어드바이스를 이용해서 실행되는 타겟 객체의 메서드명과 예외 메시지를 출력하는 어드바이스이다.
- ◎ 아래의 afterThrowing 메서드는 클래스명이 UserService로 시작되는 클래스에 속한 모든 메서드가 예외가 발생된 이후에 호출된다.
- ◎ 발생된 예외를 참조할 때는 throwing 속성을 이용해서 예외객체를 담을 변수 이름을 지정해야 한다.

```
28@  @AfterThrowing(pointcut="execution(* *..UserService.*(..))",
29                  throwing="ex")
30public void afterThrowing(JoinPoint joinPoint, Throwable ex) {
31    String signatureString = joinPoint.getSignature().getName();
32    System.out.println("@AfterThrowing [ " + signatureString + " ] 메서드 실행 중 예외 발생");
33    System.out.println("@AfterThrowing [ " + signatureString + " ] 예외= " + ex.getMessage());
34}
```

| After 어드바이스

- @After 어드바이스를 이용해서 실행되는 타겟 객체의 메서드명을 출력하는 어드바이스이다.
- 아래의 afterFinally 메서드는 메서드명이 User로 끝나는 메서드들이 정상 종료됐을 때와 예외가 발생했을 때 모두 호출된다.
- 반드시 반환해야 하는 리소스가 있거나 메서드 실행 결과를 항상 로그로 남겨야 하는 경우에 사용할 수 있다.
하지만 리턴 값이나 예외를 직접 전달받을 수는 없다.

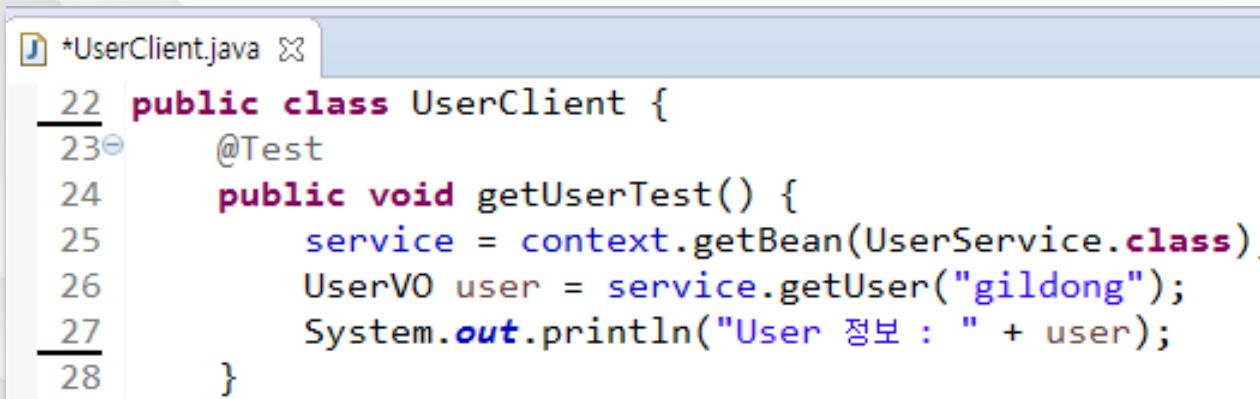
```
42     @After("execution(* *.*.*User(..))")  
43     public void afterFinally(JoinPoint joinPoint) {  
44         String signatureString = joinPoint.getSignature().getName();  
45         System.out.println("@After [ " + signatureString + " ] 메서드 실행 완료");  
46     }
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

3. Aspect 클래스 테스트

I Aspect 클래스 테스트(1)

- UserService Bean의 getUser 메서드를 호출하면, Advice가 적용된 것을 확인해 볼 수 있다.



```
*UserClient.java ✘
22 public class UserClient {
23     @Test
24     public void getUserTest() {
25         service = context.getBean(UserService.class);
26         UserVO user = service.getUser("gildong");
27         System.out.println("User 정보 : " + user);
28     }
}
```

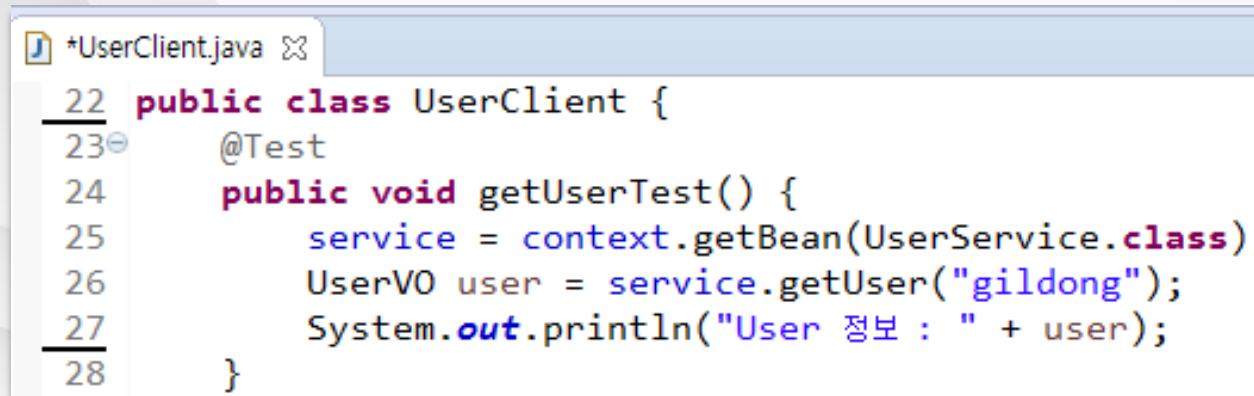
I Aspect 클래스 테스트(1)

- UserService Bean의 getUser 메서드를 호출하면, Advice가 적용된 것을 확인해 볼 수 있다.

```
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bin\javaw.exe (2016. 7. 28. 오후 12:10:42)
UserService.getUser(..) 시작
@Before [ getUser ] 메서드 실행 전처리 수행
@Before [ getUser ] 마크먼트 gildong
@Before [ read ] 메서드 실행 전처리 수행
@Before [ read ] 마크먼트 gildong
@After [ getUser ] 메서드 실행 완료
@AfterReturing [ getUser ] 메서드 실행 후처리 수행
@AfterReturing [ getUser ] 리턴값=User [userId=gildong, name=홍길동, gender=남, city=서울]
UserService.getUser(..) 종료
UserService.getUser(..) 실행 시간 : 2189 ms
User 정보 : User [userId=gildong, name=홍길동, gender=남, city=서울]
```

I Aspect 클래스 테스트(2)

- UserService Bean의 getUser 메서드가 예외 발생 시, Advice가 적용된 것을 확인해 볼 수 있다.



```
*UserClient.java
22 public class UserClient {
23     @Test
24     public void getUserTest() {
25         service = context.getBean(UserService.class);
26         UserVO user = service.getUser("gildong");
27         System.out.println("User 정보 : " + user);
28     }
}
```

I Aspect 클래스 테스트(2)

- UserService Bean의 getUser 메서드가 예외 발생 시, Advice가 적용된 것을 확인해 볼 수 있다.

```
UserService.getUser(..) 시작
```

```
@Before [ getUser ] 메서드 실행 전처리 수행  
@Before [ getUser ] 마규먼트 gildong  
@Before [ read ] 메서드 실행 전처리 수행  
@Before [ read ] 마규먼트 gildong  
@After [ getUser ] 메서드 실행 완료  
@AfterThrowing [ getUser ] 메서드 실행 중 예외 발생  
@AfterThrowing [ getUser ] 예외=PreparedStatementCallback; bad SQL grammar  
[select * from users where userid = ?;]; nested exception
```

```
UserService.getUser(..) 종료
```

```
UserService.getUser(..) 실행 시간 : 857 ms
```

| Aspect 클래스 테스트(3)

- UserService Bean의 updateUser 메서드를 호출하면, Advice가 적용된 것을 확인해 볼 수 있다.

```
@Test  
public void updateUserTest() {  
    service.updateUser(new UserVO("gildong", "홀길동2", "남2", "경기2"));  
    System.out.println(service.getUser("gildong"));  
}
```

Aspect 클래스 테스트(3)

UserService.updateUser(..) 시작

@Before [updateUser] 메서드 실행 전처리 수행

@Before [updateUser] 마크먼트 User [userId=gildong, name=홍길동2, gender=남2, city=경기2]

@Before [update] 메서드 실행 전처리 수행

@Before [update] 마크먼트 User [userId=gildong, name=홍길동2, gender=남2, city=경기2]

갱신된 Record with ID = gildong

@After [updateUser] 메서드 실행 완료

@AfterReturing [updateUser] 메서드 실행 후처리 수행

@AfterReturing [updateUser] 리턴값=null

UserService.updateUser(..) 종료

UserService.updateUser(..) 실행 시간 : 1034 ms

UserService.getUser(..) 시작

@Before [getUser] 메서드 실행 전처리 수행

@Before [getUser] 마크먼트 gildong

@Before [read] 메서드 실행 전처리 수행

@Before [read] 마크먼트 gildong

@After [getUser] 메서드 실행 완료

@AfterReturing [getUser] 메서드 실행 후처리 수행

@AfterReturing [getUser] 리턴값=User [userId=gildong, name=홍길동2, gender=남2, city=경기2]

UserService.getUser(..) 종료

UserService.getUser(..) 실행 시간 : 139 ms

User [userId=gildong, name=홍길동2, gender=남2, city=경기2]

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [AOP 어플리케이션 작성(2)]에 대해서 살펴보았습니다.

Aspect 클래스 선언 및 설정

@Aspect, @Component , <aop:aspectj-autoproxy />

Aspect 클래스 구현

@Before, @AfterReturning, @AfterThrowing, @After

Aspect 클래스 테스트

- ◎ execution() 지시자에 설정된 메서드를 호출
- ◎ LoggingAspect가 적용되어 출력된 로그를 확인

Spring Framework

16. MyBatis 개요

CONTENTS

- 1 MyBatis의 개요와 특징
- 2 MyBatis와 MyBatis-Spring의 주요 컴포넌트
- 3 MyBatis-Spring을 사용한 예제

학습 목표

- MyBatis 개요와 특징에 대하여 이해할 수 있습니다.
- MyBatis와 MyBatis-Spring의 주요 컴포넌트에 대하여 이해할 수 있습니다.
- MyBatis-Spring을 사용한 예제에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

1. MyBatis 개요와 특징

I MyBatis의 개요

MyBatis(<http://www.mybatis.org/mybatis-3/>)는
자바 오브젝트와 SQL문 사이의 자동 Mapping 기능을 지원하는
ORM 프레임워크이다.

- MyBatis는 SQL을 별도의 파일로 분리해서 관리하게 해주며,
객체-SQL 사이의 파라미터 Mapping 작업을 자동으로 해주기 때문에
많은 인기를 얻고 있는 기술이다.
- MyBatis는 Hibernate나 JPA(Java Persistence Api)처럼 새로운 DB
프로그래밍 패러다임을 익혀야 하는 부담이 없이, 개발자가 익숙한
SQL을 그대로 이용하면서 JDBC 코드 작성의 불편함도 제거해주고,
도메인 객체나 VO 객체를 중심으로 개발이 가능하다는 장점이 있다.

I MyBatis의 특징

01

쉬운 접근성과 코드의 간결함

- ◎ 가장 간단한 퍼시턴스 프레임워크
- ◎ XML 형태로 서술된 JDBC 코드라고 생각해도 될 만큼 JDBC의 모든 기능을 MyBatis가 대부분 제공한다.
- ◎ 복잡한 JDBC코드를 걷어내며 깔끔한 소스코드를 유지할 수 있다.
- ◎ 수동적인 파라미터 설정과 쿼리 결과에 대한 맵핑 구문을 제거할 수 있다.

I MyBatis의 특징

02 SQL문과 프로그래밍 코드의 분리

- ◎ SQL에 변경이 있을 때마다 자바 코드를 수정하거나 컴파일 하지 않아도 된다.
- ◎ SQL 작성과 관리 또는 검토를 DBA와 같은 개발자가 아닌 다른 사람에게 맡길 수도 있다.

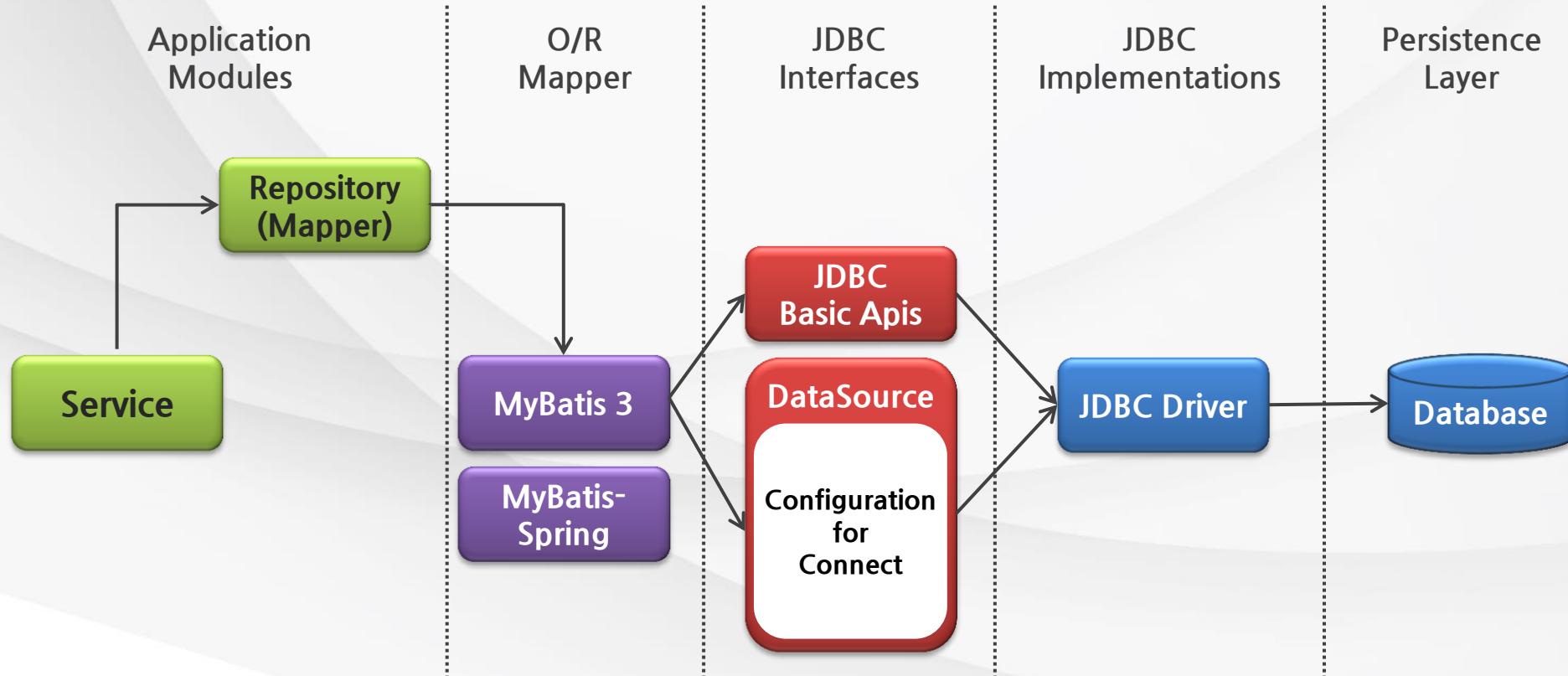
03 다양한 프로그래밍 언어로 구현가능

- ◎ Java, C#, .NET , Ruby

A photograph of a person's hands holding a black smartphone. The background is dark with blurred, colorful circular bokeh lights, suggesting a night scene or a city street.

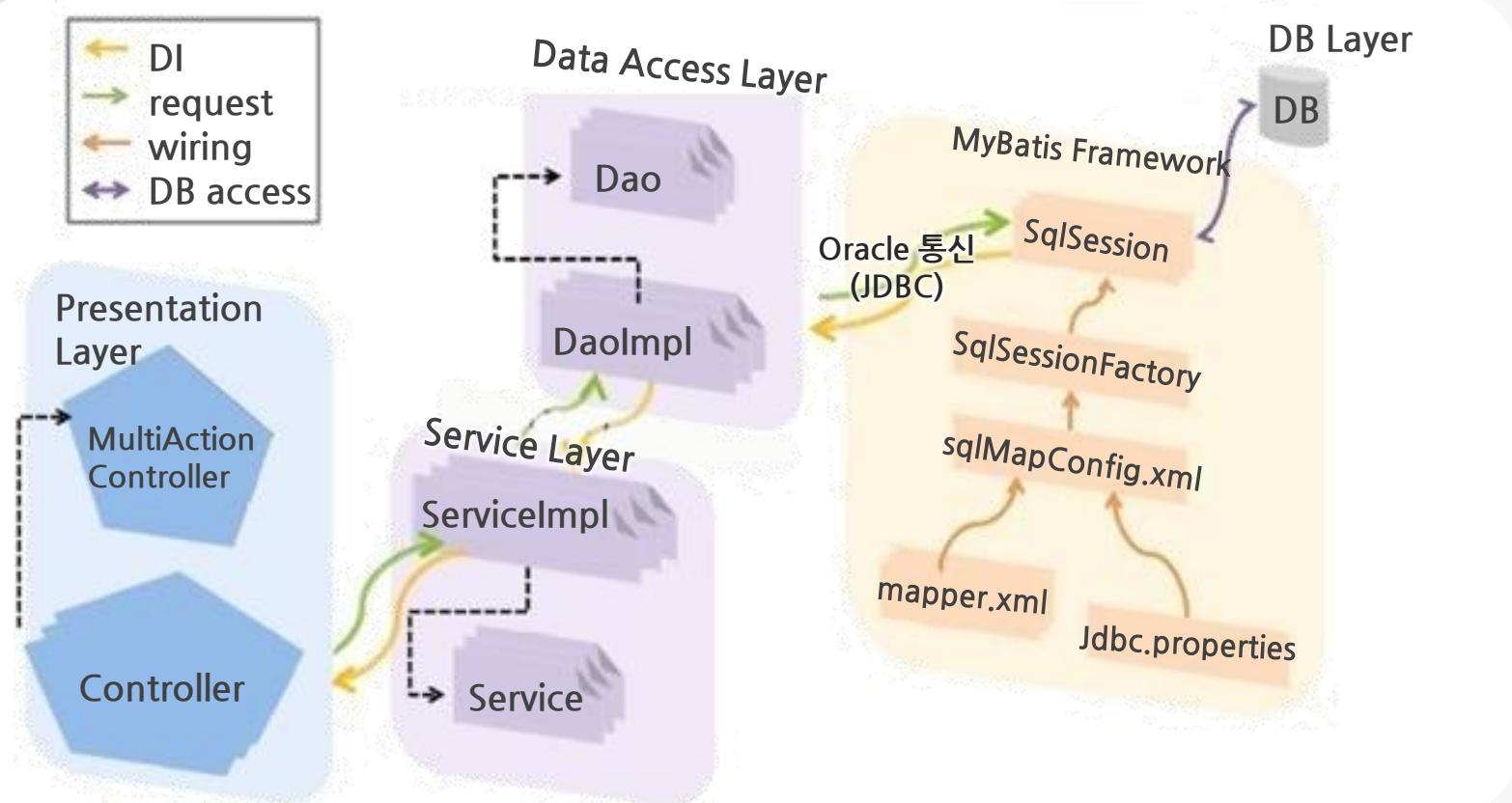
2. MyBatis3와 MyBatis-Spring의 주요 컴포넌트

■ MyBatis와 MyBatis-Spring을 사용한 DB 액세스 Architecture



2. MyBatis3과 MyBatis-Spring의 주요 컴포넌트

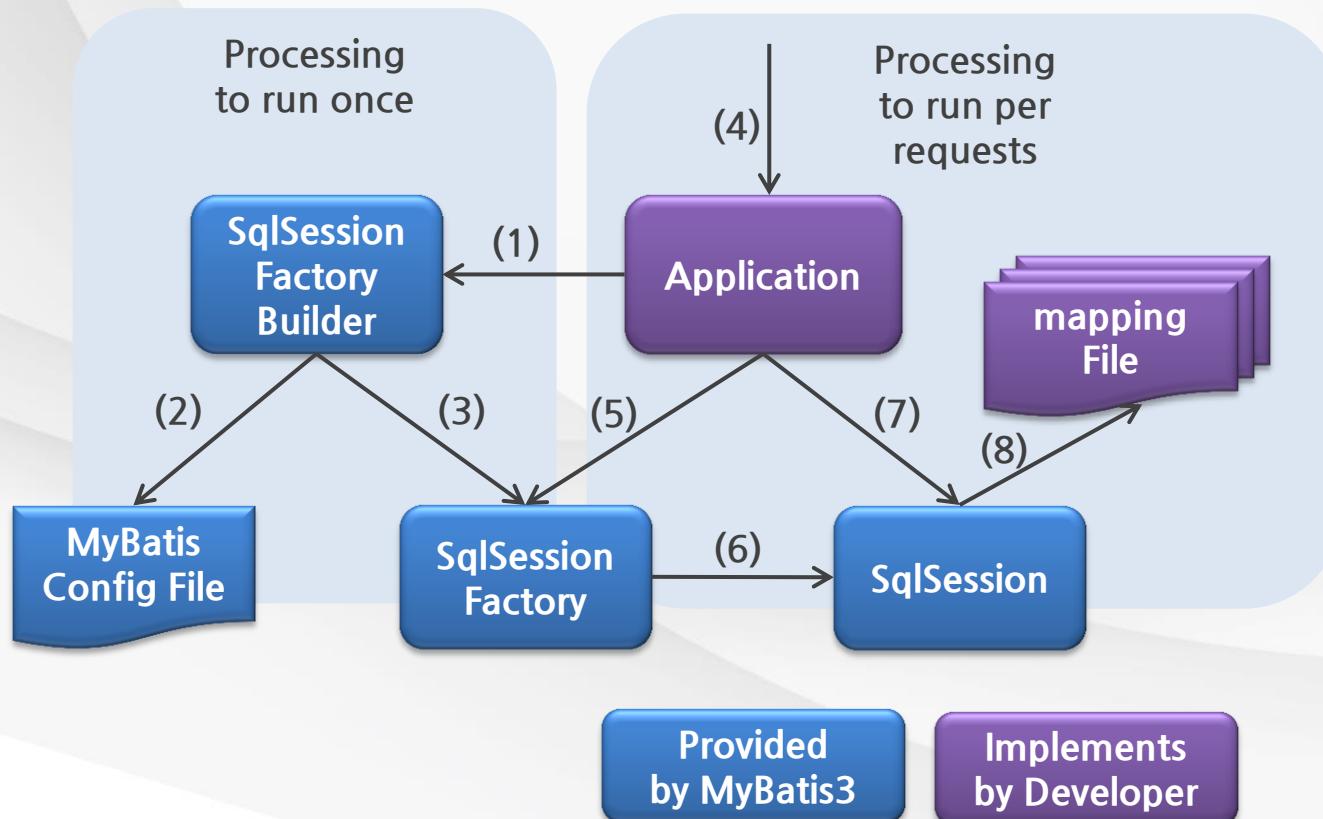
I MyBatis를 사용하는 데이터 액세스 계층



2. MyBatis3과 MyBatis-Spring의 주요 컴포넌트

Tacademy

■ MyBatis3의 주요 컴포넌트



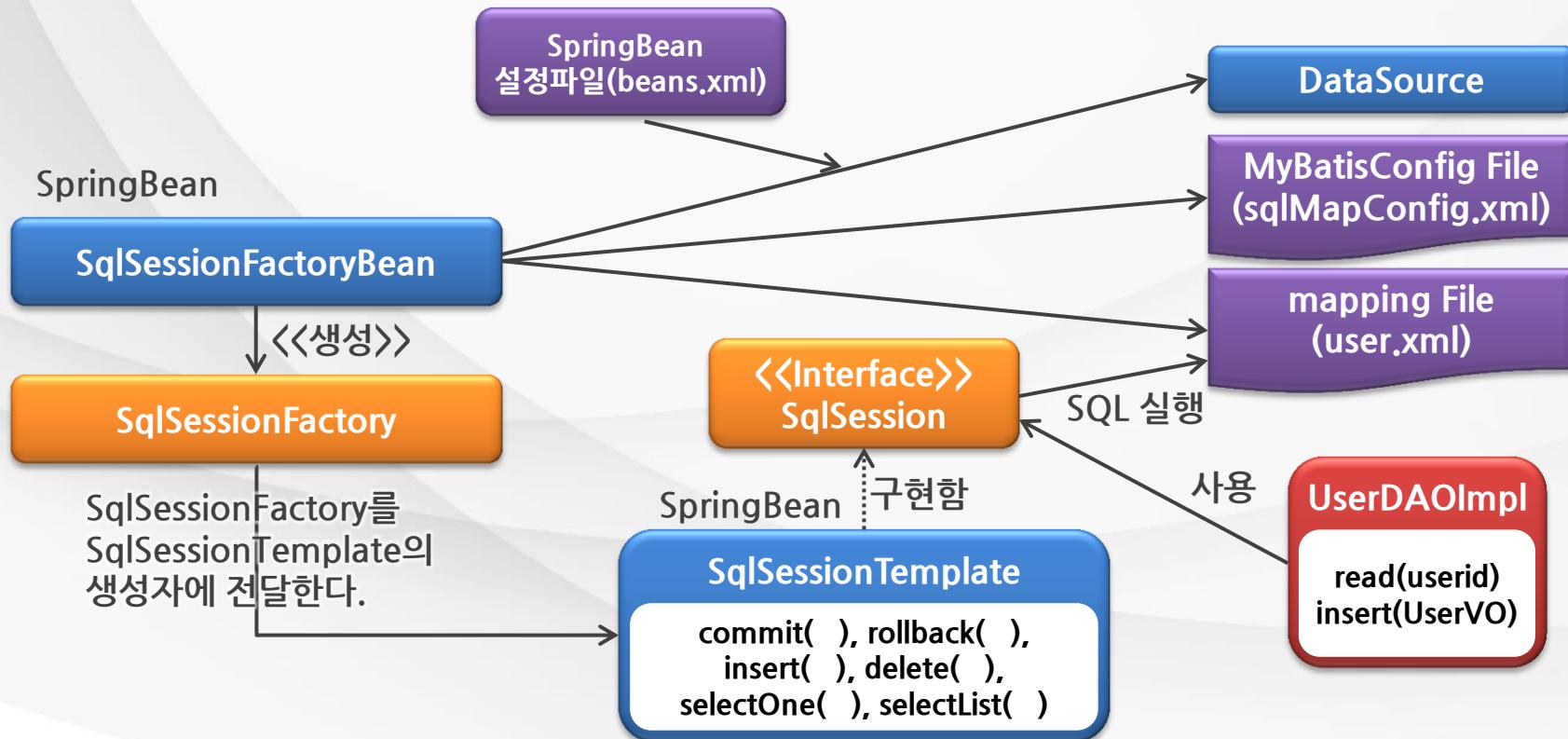
■ MyBatis3의 주요 컴포넌트의 역할

MyBatis 설정파일 (SqlMapConfig.xml)	<ul style="list-style-type: none">데이터베이스의 접속 주소 정보나 Mapping 파일의 경로 등의 고정된 환경정보를 설정한다.
SqlSession FactoryBuilder	<ul style="list-style-type: none">MyBatis 설정 파일을 바탕으로 SqlSessionFactory를 생성한다.
SqlSessionFactory	<ul style="list-style-type: none">SqlSession을 생성한다.
SqlSession	<ul style="list-style-type: none">핵심적인 역할을 하는 클래스로서 SQL 실행이나 트랜잭션 관리를 실행한다.SqlSession 오브젝트는 Thread-Safe 하지 않으므로 thread마다 필요에 따라 생성한다.
mapping 파일 (user.xml)	<ul style="list-style-type: none">SQL문과 OR Mapping을 설정한다.

2. MyBatis3과 MyBatis-Spring의 주요 컴포넌트

Tacademy

MyBatis-Spring의 주요 컴포넌트



■ MyBatis-Spring의 주요 컴포넌트의 역할

MyBatis 설정파일 (sqlMapConfig.xml)	<ul style="list-style-type: none">▪ VO 객체의 정보를 설정한다.
SqlSession FactoryBean	<ul style="list-style-type: none">▪ MyBatis 설정파일을 바탕으로 SqlSessionFactory를 생성한다.▪ Spring Bean으로 등록해야 한다.
SqlSessionTemplate	<ul style="list-style-type: none">▪ 핵심적인 역할을 하는 클래스로서 SQL 실행이나 트랜잭션 관리를 실행한다.▪ SqlSession 인터페이스를 구현하며, Thread-safe하다.▪ Spring Bean으로 등록해야 한다.

■ MyBatis-Spring 의 주요 컴포넌트의 역할

Mapping 파일 (user.xml)

- SQL문과 OR Mapping을 설정한다.

Spring Bean 설정파일(bean.xml)

- SqlSessionFactoryBean을 Bean 등록할 때
DataSource 정보와 MyBatis Config 파일정보,
Mapping 파일의 정보를 함께 설정한다.
- SqlSessionTemplate을 Bean으로 등록한다.

A photograph of a person's hands holding a black smartphone. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

3. MyBatis-Spring을 사용한 예제

■ Mapping 파일 (SQL문 포함)

```
*User.xml
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4<mapper namespace="userNS">
5
6<select id="selectUserById" parameterType="string" resultType="User">
7     select * from users where userid=#{value}
8 </select>
9<insert id="insertUser" parameterType="User">
10    insert into users values(#{userId},#{name},#{gender},#{city} )
11 </insert>
```

■ MyBatis Configuration 파일

```
SqlMapConfig.xml
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
3      "http://mybatis.org/dtd/mybatis-3-config.dtd">
4
5<configuration>
6    <typeAliases>
7        <typeAlias alias="User" type="my.spring.vo.UserVO" />
8        <typeAlias alias="Student" type="my.spring.vo.Student" />
9        <typeAlias alias="Dept" type="my.spring.vo.Dept" />
10    </typeAliases>
11 </configuration>
```

Spring Bean 설정 파일

```
beans.xml
126
127      <!-- MyBatis 설정 -->
128      <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
129          <property name="dataSource" ref="dataSource" />
130          <property name="configLocation" value="classpath:config/SqlMapConfig.xml" />
131          <property name="mapperLocations">
132              <list>
133                  <value>classpath:config/User.xml</value>
134                  <value>classpath:config/Student.xml</value>
135              </list>
136          </property>
137      </bean>
138      <!-- SqlSessionTemplate 설정 -->
139      <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
140          <constructor-arg ref="sqlSessionFactory" />
141      </bean>
```

데이터 액세스 계층의 DAO 구현 클래스

UserDaoImpl.java

```
11 @Repository("userDao")
12 public class UserDaoImpl implements UserDao {
13     @Autowired
14     private SqlSession session;
15
16     public void insert(UserVO user) {
17         session.update("userNS.insertUser", user);
18         System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" + user.getName());
19     }
20
21     @Override
22     public UserVO read(String id) {
23         UserVO user = session.selectOne("userNS.selectUserById", id);
24         return user;
25     }
```

```
<mapper namespace="userNS">
    <select id="selectUserById" parameterType="string" resultType="User">
        select * from users where userid=#{value}
    </select>
    <insert id="insertUser" parameterType="User">
        insert into users values(#{userId},#{name},#{gender},#{city} )
    </insert>
```

User.xml →

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [MyBatis 개요]에 대해서 살펴보았습니다.

MyBatis 개요와 특징

MyBatis는 ORM 프레임워크, 자바 코드와 SQL를 분리함

MyBatis 와 MyBatis-Spring 주요 컴포넌트

- SqlSessionFactoryBuilder, SqlSessionFactory, SqlSession
- SqlSessionFactoryBean, SqlSessionTemplate

MyBatis-Spring을 사용한 예제

MyBatis Config 파일, Mapping 파일 , Spring Bean 설정 파일
DAO 클래스에서 SqlSession의 사용

Spring Framework

17. MyBatis 어플리케이션 작성(1)

CONTENTS

1

MyBatis와 MyBatis-Spring 설치

2

Mapping 파일 작성 및 MyBatis 설정

3

DAO 클래스 작성 및 테스트

학습 목표

- MyBatis 및 MyBatis-Spring 설치에 대하여 이해할 수 있습니다.
- Mapping 파일 작성 및 MyBatis 설정에 대하여 이해할 수 있습니다.
- DAO 클래스 작성 및 테스트에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city street.

1. MyBatis와 MyBatis-Spring 설치

I MyBatis 라이브러리 검색 및 설치

<http://mvnrepository.com>에 접근한다.

mybatis로 검색한다.

mybatis 3.4.1 버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.1</version>
</dependency>
```

MyBatis-Spring 라이브러리 검색 및 설치

<http://mvnrepository.com>에 접근한다.

mybatis spring로 검색한다.

mybatis spring 1.3.0 버전을 pom.xml에 추가한다.

```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.0</version>
</dependency>
```

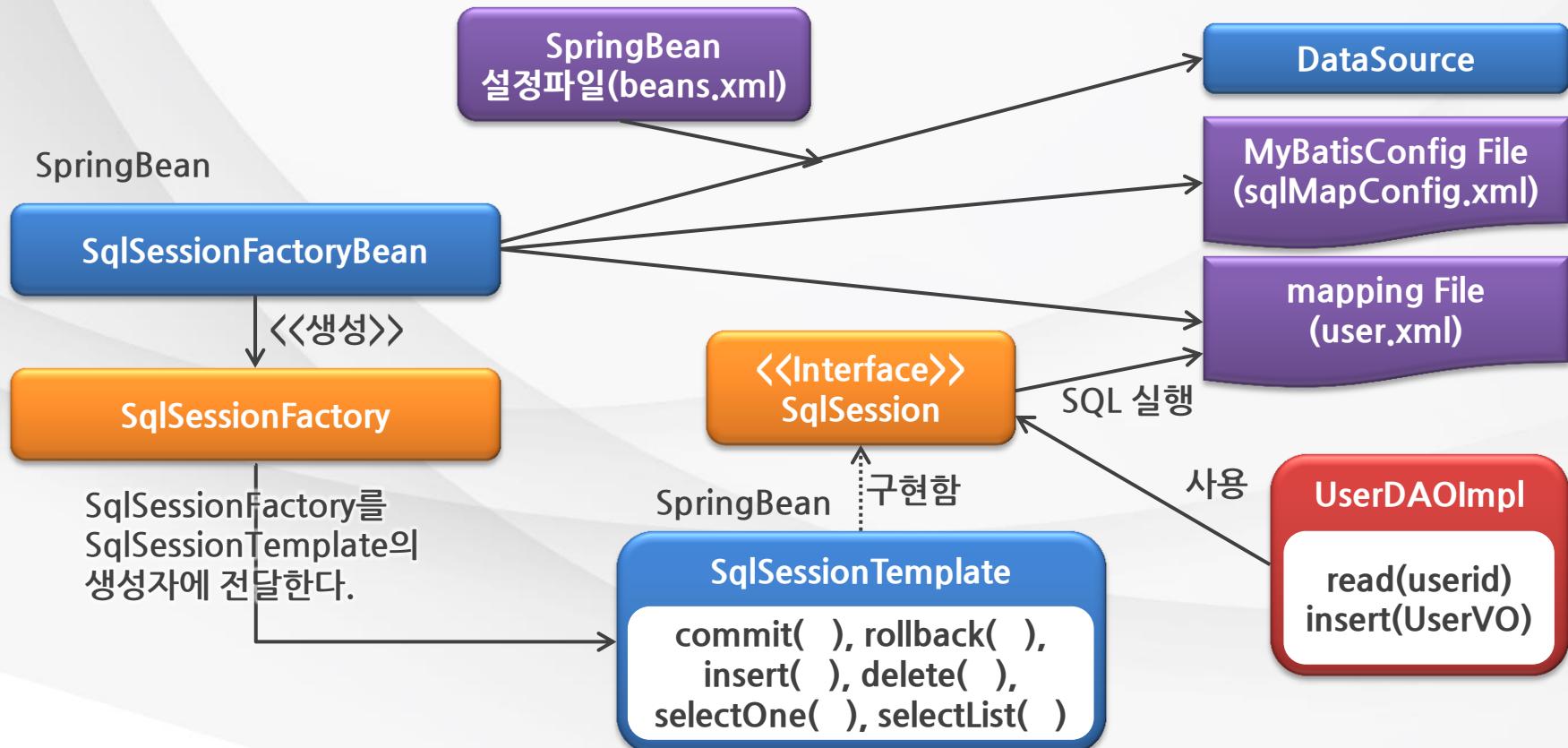


2. Mapping 파일 작성 및 MyBatis 설정

2. Mapping 파일 작성 및 MyBatis 설정

Tacademy

MyBatis-Spring의 주요 컴포넌트



MyBatis Config 파일 작성(SqlMapConfig.xml)

```
SqlMapConfig.xml
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
3      "http://mybatis.org/dtd/mybatis-3-config.dtd">
4
5<configuration>
6    <typeAliases>
7        <typeAlias alias="User" type="myspring.user.vo.UserVO" />
8    </typeAliases>
9 </configuration>
```

```
UserVO.java
1 package myspring.user.vo;
2
3 public class UserVO {
4
5     private String userId;
6     private String name;
7     private String gender;
8     private String city;
```

2. Mapping 파일 작성 및 MyBatis 설정

설정파일과 VO 객체와 테이블간의 관계

SqlMapConfig.xml

```
<typeAlias alias="User" type="myspring.user.vo.UserVO" />
```

User.xml

```
<select id="selectUserList" resultType="User">
    select * from users order by userid
</select>
<insert id="insertUser" parameterType="User">
    insert into users values(#{userId},#{name},#{gender},#{city} )
</insert>
```

UserVO.java

```
1 package myspring.user.vo;
2
3 public class UserVO {
4
5     private String userId;
6     private String name;
7     private String gender;
8     private String city;
```

CREATE TABLE USERS

```
(  
    userid      VARCHAR2(30) NOT NULL PRIMARY KEY,  
    name        VARCHAR2(100) NOT NULL,  
    gender      VARCHAR2(10),  
    city        VARCHAR2(30)  
);
```

I Mapping 파일 작성(1) : 조회 SQL 문

```
User.xml ✘
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4<mapper namespace="userNS">
5
6<select id="selectUserById" parameterType="string" resultType="User">
7   select * from users where userid=#{value}
8 </select>
9
10<select id="selectUserList" resultType="User">
11   select * from users order by userid
12 </select>
```

Mapping 파일 작성(2) : 등록, 수정, 삭제 SQL 문

```
*User.xml
14    <insert id="insertUser" parameterType="User">
15        insert into users values(#{userId},#{name},#{gender},#{city} )
16    </insert>
17
18    <update id="updateUser" parameterType="User">
19        update users set
20            name =
21            #{name},
22            gender = #{gender},
23            city = #{city}
24        where userid = #{userId}
25    </update>
26
27    <delete id="deleteUser" parameterType="string">
28        delete from users where
29            userid = #{value}
30    </delete>
31 </mapper>
```

I Spring Bean 설정 파일

```
beans.xml
10   <!-- SqlSessionFactoryBean 설정 -->
11   <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
12     <property name="dataSource" ref="dataSource" />
13     <property name="configLocation" value="classpath:config/SqlMapConfig.xml" />
14     <property name="mapperLocations">
15       <list>
16         <value>classpath:config/User.xml</value>
17       </list>
18     </property>
19   </bean>
20   <!-- SqlSessionTemplate 설정 -->
21   <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
22     <constructor-arg ref="sqlSessionFactory" />
23   </bean>
```

A photograph of a person's hands holding a smartphone. The phone screen is white. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

3. DAO 클래스 작성 및 테스트

SqlSessionTemplate을 사용한 DAO 클래스 작성(1)

The screenshot shows an IDE interface with two tabs: 'UserDaoImpl.java' and 'User.xml'. The 'UserDaoImpl.java' tab is active, displaying Java code for a DAO implementation. The 'User.xml' tab is visible but inactive.

```
1 package myspring.user.dao;
2
3+ import java.util.List;..
10 @Repository("userDao")
11 public class UserDaoImpl implements UserDao {
12@   @Autowired
13     private SqlSession session;
14
15@   @Override
16     public UserVO read(String id) {
17         UserVO user = session.selectOne("userNS.selectUserById", id);
18         return user;
19     }
20
21@   public List<UserVO> readAll() {
22     List<UserVO> userList = session.selectList("userNS.selectUserList");
23     return userList;
24 }
```

I SqlSessionTemplate을 사용한 DAO 클래스 작성(2)

The screenshot shows an IDE interface with two tabs: *UserDaoImpl.java and User.xml. The UserDaoImpl.java tab is active, displaying Java code for a DAO implementation. The User.xml tab is visible but contains no content.

```
26 public void insert(UserVO user) {  
27     session.update("userNS.insertUser", user);  
28     System.out.println("등록된 Record UserId=" + user.getUserId() +  
29                         " Name=" + user.getName());  
30 }  
31  
32 @Override  
33 public void update(UserVO user) {  
34     session.update("userNS.updateUser", user);  
35 }  
36  
37 @Override  
38 public void delete(String id) {  
39     session.delete("userNS.deleteUser", id);  
40     System.out.println("삭제된 Record with ID = " + id );  
41 }  
42 }
```

■ DAO 클래스 테스트 : 사용자 정보 조회

```
*UserClient.java ✘
21 public class UserClient {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void getUserTest() {
29         UserVO user = service.getUser("gildong");
30         System.out.println("User 정보 : " + user);
31         assertEquals("홍길동", user.getName());
32     }
}
```

실행 결과

[java.lang.IllegalStateException: Failed to load ApplicationContext](#)

Annotation-specified bean name 'userDao' for bean class [myspring.user.dao.UserDaoImplJDBC] conflicts with existing, non-compatible bean definition of same name and class [myspring.user.dao.UserDaoImpl]+

■ DAO 클래스 테스트 : 사용자 등록 및 목록 조회

```
UserClient.java ✘
21 public class UserClient {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void insertUserTest() {
29         service.insertUser(new UserVO("polar", "연아", "여", "경기"));
30
31         for (UserVO user : service.getUserList()) {
32             System.out.println(user);
33         }
34     }
}
```

실행 결과

UserService.insertUser(..) 실행 시간 : 1090
UserService.getUserList() 실행 시간 : 143 ms

■ DAO 클래스 테스트 : 사용자 정보 수정

```
UserClient.java X
21 public class UserClient {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void updateUserTest() {
29         service.updateUser(new UserVO("polar", "연아2", "여2", "경기2"));
30         System.out.println(service.getUser("polar"));
31     }
}
```

실행 결과

UserService.updateUser(..) 실행 시간 : 2031 ms
UserService.getUser(..) 실행 시간 : 189 ms

■ DAO 클래스 테스트 : 사용자 정보 삭제 및 목록조회

UserClient.java

```
21 public class UserClient {  
22     @Autowired  
23     ApplicationContext context;  
24     @Autowired  
25     UserService service;  
26  
27     @Test  
28     public void deleteUserTest() {  
29         service.deleteUser("polar");  
30  
31         for (UserVO user : service.getUserList()) {  
32             System.out.println(user);  
33         }  
34     }  
}
```

실행 결과

```
UserService.deleteUser(..) 실행 시간 : 2066 ms  
UserService.getUserList() 실행 시간 : 209 ms
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [MyBatis 어플리케이션 작성(1)]에 대해서 살펴보았습니다.

MyBatis 및 MyBatis-Spring 설치

- MyBatis 설치
- MyBatis-Spring 설치

Mapping 파일작성 및 MyBatis 설정

- Mapping 파일 작성
- MyBatis 설정

DAO 클래스 작성 및 테스트

- DAO 클래스 작성
- DAO 클래스 테스트

Spring Framework

18. MyBatis 어플리케이션 작성(2)

CONTENTS

1

Mapper 인터페이스 개념

2

Mapper 인터페이스 작성 및 설정

3

여러 개의 Mapper 인터페이스 설정

학습 목표

- Mapper 인터페이스 개념에 대하여 이해할 수 있습니다.
- Mapper 인터페이스 작성 및 설정에 대하여 이해할 수 있습니다.
- 여러 개의 Mapper 인터페이스 설정에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The phone's screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

1. Mapper 인터페이스 개념

■ Mapper 인터페이스

Mapper 인터페이스는 Mapping 파일에 기재된 SQL을
호출하기 위한 인터페이스

- ◎ Mapper 인터페이스는 SQL을 호출하는 프로그램을
Type Safe하게 기술하기 위해 MyBatis3.x부터 등장
- ◎ Mapping 파일에 있는 SQL을 자바 인터페이스를 통해
호출할 수 있도록 해줌

■ Mapper 인터페이스를 사용하지 않았을 때

```
session.selectOne("userNS.selectUserById", userid);
```

UserDAOImpl
read(userid)

```
<mapper namespace="userNS">  
  <select id="selectUserById" parameterType="String">
```

- Mapper 인터페이스를 사용하지 않으면,
SQL을 호출하는 프로그램은 SqlSession의 메서드의 아규먼트에 문자열로
네임스페이스+”.+SQL ID로 지정해야 함
- 문자열로 지정하기 때문에 오타에 의해 버그가 숨어있거나,
IDE에서 제공하는 code assist 를 사용할 수 없음

Mapper 인터페이스를 사용하였을 때

userMapper.selectUserById(userid);

```
<mapper namespace="myspring.user.dao.UserMapper">
  <select id="selectUserById" parameterType="String">
```

UserDAOImpl
read(userid)



- UserMapper 인터페이스는 개발자가 작성
- 패키지 이름+”.”+인터페이스 이름+”.”+메서드 이름이 네임스페이스+”.”+SQL ID가 되도록 네임스페이스와 SQL의 ID를 설정해야 함
- 네임스페이스 속성에는 패키지를 포함한 Mapper 인터페이스 이름
- SQL ID에는 매팅하는 메서드 이름을 지정하는 것



2. Mapper 인터페이스 작성 및 설정

■ Mapper 인터페이스 작성

```
UserMapper.java X
1 package myspring.user.dao;
2
3 import java.util.List;
4
5
6
7 public interface UserMapper {
8     UserVO selectUserById(String id);
9     List<UserVO> selectUserList();
10    void insertUser(UserVO userVO);
11    void updateUser(UserVO userVO);
12    void deleteUser(String id);
13 }
```

■ Mapping 파일 수정

기
준

```
<mapper namespace="userNS">
    <select id="selectUserById" parameterType="string" resultType="User">
        select * from users where userid=#{value}
    </select>
```

변
경

```
<mapper namespace="myspring.user.dao.UserMapper">
    <select id="selectUserById" parameterType="string" resultType="User">
        select * from users where userid=#{value}
    </select>
```



■ DAO 클래스 수정

기
준

```
@Repository("userDao")
public class UserDaoImpl implements UserDao {
    @Autowired
    private SqlSession session;
    @Override
    public UserVO read(String id) {
        UserVO user = session.selectOne("userNS.selectUserById", id);
        return user;
    }
}
```

변
경

```
@Repository("userDao")
public class UserDaoImpl implements UserDao {
    @Autowired
    private UserMapper userMapper;
    @Override
    public UserVO read(String id) {
        UserVO user = userMapper.selectUserById(id);
        return user;
    }
}
```



■ MapperFactoryBean의 설정

```
beans.xml ✘
10      <!-- Mapper 설정 -->
11      <bean id="userMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
i 12          <property name="mapperInterface" value="myspring.user.dao.UserMapper" />
i 13          <property name="sqlSessionTemplate" ref="sqlSession" />
14      </bean>
```

- MapperFactoryBean은 UserMapper를 구현하는 프록시 클래스를 생성하고, 그것을 어플리케이션에 injection 한다.
- 프록시는 런타임 시에 생성되므로, 지정된 Mapper는 실제 구현 클래스가 아닌, 인터페이스여야만 한다.
- MapperFactoryBean은 sqlSessionFactory나 sqlSessionTemplate를 필요로 한다.

Mapper 인터페이스의 사용 테스트

```
*UserClient.java ✘
21 public class UserClient {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void getUserTest() {
29         UserVO user = service.getUser("gildong");
30         System.out.println("User 정보 : " + user);
31         assertEquals("홍길동", user.getName());
32     }
}
```

실행 결과

```
Markers Console ✘ Maven Repositories JUnit Properties
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bi
UserDao.read(..) 실행 시간 : 1520 ms
UserService.getUser(..) 실행 시간 : 1521 ms
```

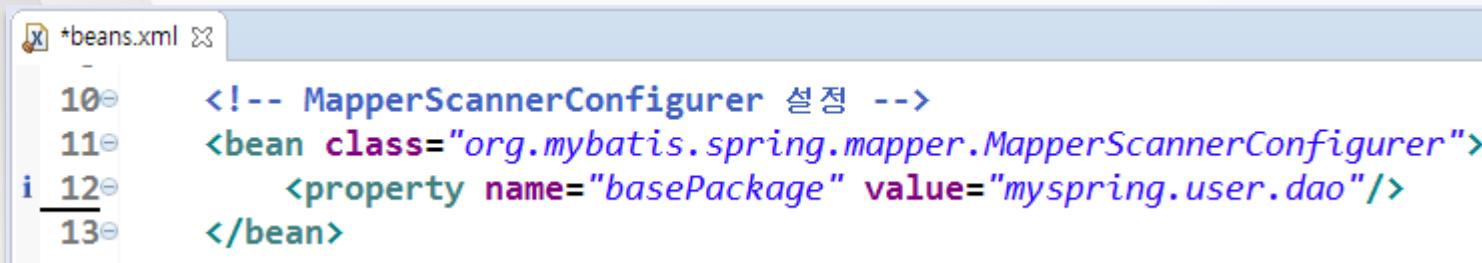


3. 여러 개의 Mapper 인터페이스 설정

I MapperScannerConfigurer의 사용

- MapperFactoryBean을 이용해 Mapper 인터페이스를 등록할 때
Mapper 인터페이스의 개수가 많아지게 되면 일일이 정의하는데
시간이 많이 걸림
- Mapper 인터페이스의 수가 많아지면
MapperScannerConfigurer를 이용하여 Mapper 인터페이스의
객체를 한 번에 등록하는 것이 편리함
- MapperScannerConfigurer를 이용하면 지정한 패키지 아래
모든 인터페이스가 Mapper 인터페이스로 간주되어
Mapper 인터페이스의 객체가 DI 컨테이너에 등록되는 것

I MapperScannerConfigurer의 설정



```
*beans.xml

10<!-- MapperScannerConfigurer 설정 -->
11<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
12    <property name="basePackage" value="myspring.user.dao"/>
13</bean>
```

- basePackage 속성에서 지정하는 것은
Mapper 인터페이스를 검색할 대상이 되는 Package
- myspring.user.dao 아래의 인터페이스들은 모두
Mapper 인터페이스에 대응하여 Mapper 객체가 생성된다는 점
- 예상하지 않은 다른 객체가 등록되어 오류가 발생할 수 있음

MapperScannerConfigurer의 설정 테스트

```
*UserClient.java ✘
21 public class UserClient {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void getUserTest() {
29         UserVO user = service.getUser("gildong");
30         System.out.println("User 정보 : " + user);
31         assertEquals("홍길동", user.getName());
32     }
}
```

실행 결과

Caused by:
org.springframework.context.annotation.ConflictingBeanDefinitionException: Annotation-specified bean name 'userDao' for
bean class [myspring.user.dao.UserDao] conflicts with
existing, non-compatible bean definition of same name and
class [myspring.user.dao.UserDaoImpl]

I Marker 인터페이스와 Marker 어노테이션의 사용

- 검색의 대상이 되는 Package 아래의 인터페이스들 중에서 Mapper로서 작성한 인터페이스로만 범위를 좁히려면 Marker 인터페이스와 Marker 어노테이션을 작성하여 MapperScannerConfigurer에 설정하면 됨

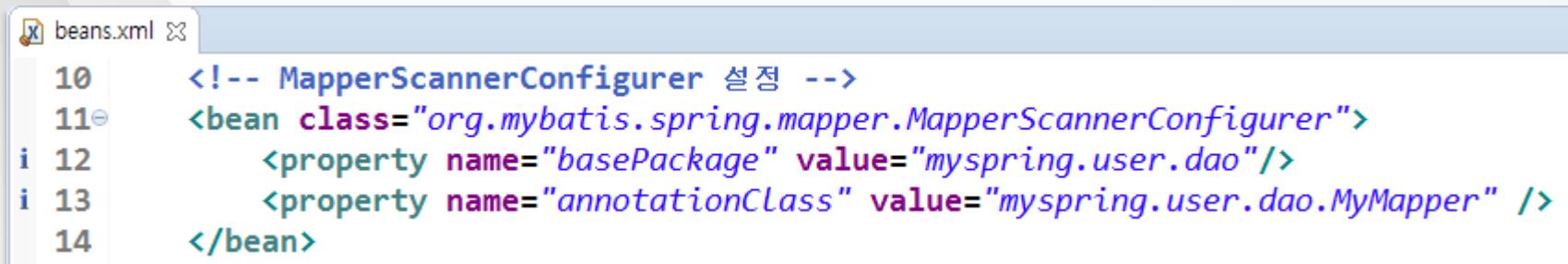
Marker 인터페이스

```
1 package myspring.user.dao;  
2  
3 public @interface MyMapper {  
4 }
```

Marker 어노테이션 부여

```
1 package myspring.user.dao;  
2+ import java.util.List;  
3  
4 @MyMapper  
5 public interface UserMapper {  
6     UserVO selectUserById(String id);  
7     List<UserVO> selectUserList();  
8     void insertUser(UserVO userVO);  
9     void updateUser(UserVO userVO);  
10    void deleteUser(String id);  
11 }
```

■ MapperScannerConfigurer에 Marker 어노테이션 지정



```
beans.xml
10  <!-- MapperScannerConfigurer 설정 -->
11  <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
12      <property name="basePackage" value="myspring.user.dao"/>
13      <property name="annotationClass" value="myspring.user.dao.MyMapper" />
14  </bean>
```

- org.mybatis.spring.mapper.MapperScannerConfigurer : 컴포넌트 스캔을 통하여 Mapper를 찾기 위한 설정
- basePackage : Mapper을 찾는 베이스 패키지
- annotationClass : Mapper를 지정하는 어노테이션 클래스

■ MapperScannerConfigurer의 설정 테스트

```
*UserClient.java
21 public class UserClient {
22     @Autowired
23     ApplicationContext context;
24     @Autowired
25     UserService service;
26
27     @Test
28     public void getUserTest() {
29         UserVO user = service.getUser("gildong");
30         System.out.println("User 정보 : " + user);
31         assertEquals("홍길동", user.getName());
32     }
}
```

실행 결과

Markers Console Maven Repositories JUnit Properties
<terminated> UserClient (2) [JUnit] C:\Program Files (x86)\Java\jre1.8.0_91\bi
UserDao.read(..) 실행 시간 : 1520 ms
UserService.getUser(..) 실행 시간 : 1521 ms

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [MyBatis 어플리케이션 작성(2)]에 대해서 살펴보았습니다.

Mapper 인터페이스 개념

- Mapping 파일의 SQL문 호출을 Type Safe하게 호출하기 위한 인터페이스

Mapper 인터페이스 작성 및 설정

- Mapper 인터페이스 작성
- MapperFactoryBean을 이용한 Mapper 등록

여러 개의 Mapper 인터페이스 설정

- 여러 개의 Mapper 인터페이스 작성
- MapperScannerConfigurer을 이용한 Mapper 등록

Spring Framework

19. Spring MVC 개요

CONTENTS

1

MVC 패턴의 개념과 모델2 아키텍쳐

2

Spring MVC 개념

3

Spring MVC 기반 웹어플리케이션 개발

학습 목표

- MVC 패턴의 개념과 모델2 아키텍쳐에 대하여 이해할 수 있습니다.
- Spring MVC 개념에 대하여 이해할 수 있습니다.
- Spring MVC 기반 웹 어플리케이션 개발에 대해 이해할 수 있습니다.

AOP는

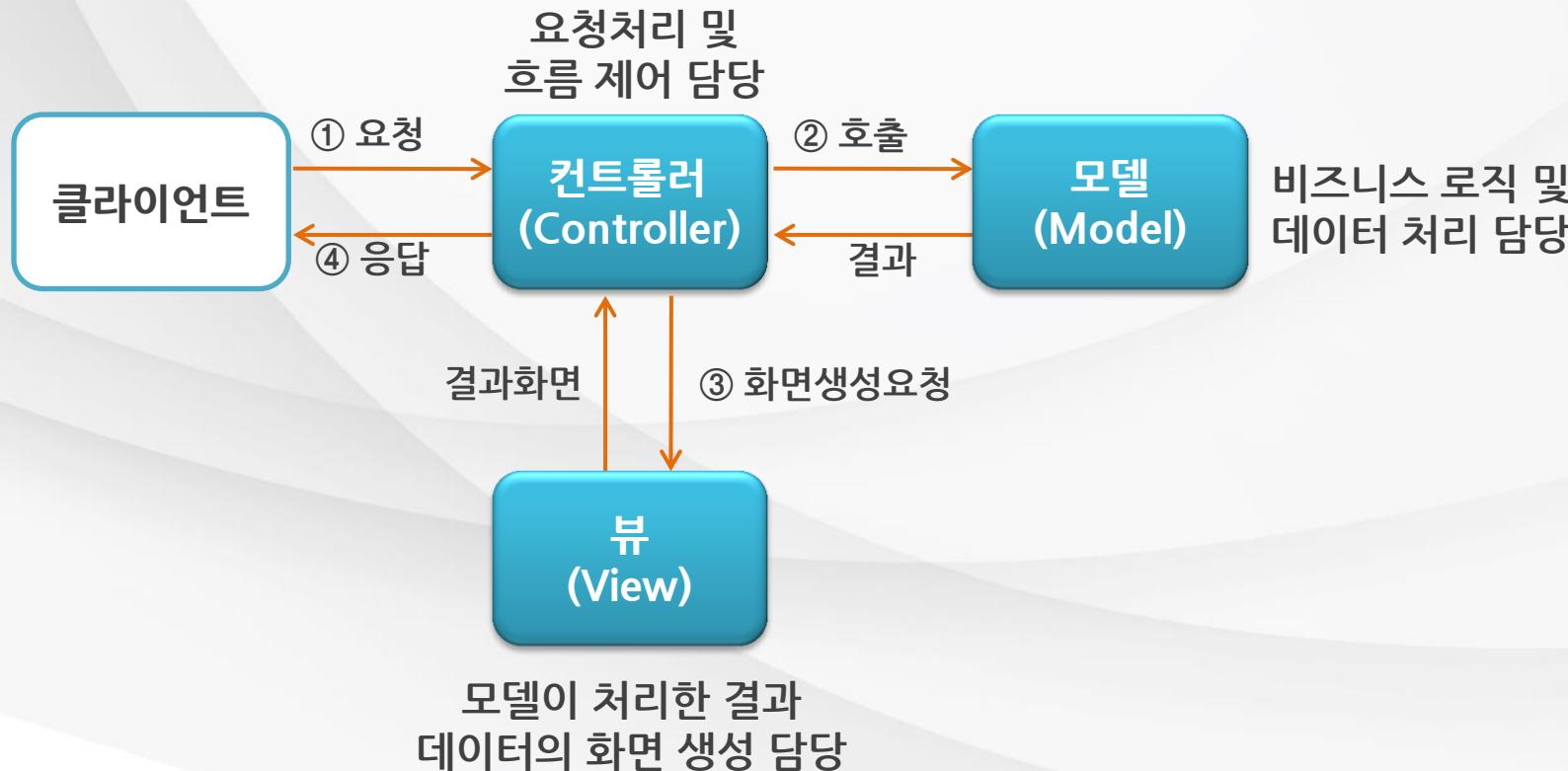
1. MVC 패턴의 개념과 모델2 아키텍쳐

I MVC(Model-View-Controller) 패턴의 개념

모델-뷰-컨트롤러(Model-View-Controller, MVC)는 소프트웨어 공학에서 사용되는 **아키텍쳐 패턴**으로 MVC 패턴의 주 목적은 **Business logic**과 **Presentation logic**을 분리하기 위함이다.

- MVC 패턴을 사용하면, 사용자 인터페이스로부터 비지니스 로직을 분리하여 애플리케이션의 시각적 요소나 그 이면에서 실행되는 비지니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 애플리케이션을 만들 수 있음
 - **Model** : 애플리케이션의 정보(데이터, Business Logic 포함)
 - **View** : 사용자에게 제공할 화면(Presentation Logic)
 - **Controller** : Model과 View 사이의 상호 작용을 관리

I MVC(Model-View-Controller) 패턴의 개념



| 각각의 MVC 컴포넌트의 역할

❖ 모델(Model) 컴포넌트

데이터 저장소(ex : 데이터베이스 등)와 연동하여 사용자가 입력한 데이터나 사용자에게 출력할 데이터를 다루는 일을 함

여러 개의 데이터 변경 작업(추가, 변경, 삭제)을 하나의 작업으로 묶는 트랜잭션을 다루는 일도 함

DAO 클래스, Service 클래스에 해당

| 각각의 MVC 컴포넌트의 역할

❖ 뷰(View) 컴포넌트

모델이 처리한 데이터나 그 작업 결과를 가지고 사용자에게 출력할 화면을 만드는 일을 함

생성된 화면은 웹 브라우저가 출력하고, 뷰 컴포넌트는 HTML과 CSS, Java Script를 사용하여 웹 브라우저가 출력할 UI를 만듦

Html과 JSP를 사용하여 작성할 수 있음

| 각각의 MVC 컴포넌트의 역할

❖ 컨트롤러(Controller) 컴포넌트

클라이언트의 요청을 받았을 때 그 요청에 대해 실제 업무를 수행하는 모델 컴포넌트를 호출하는 일을 함

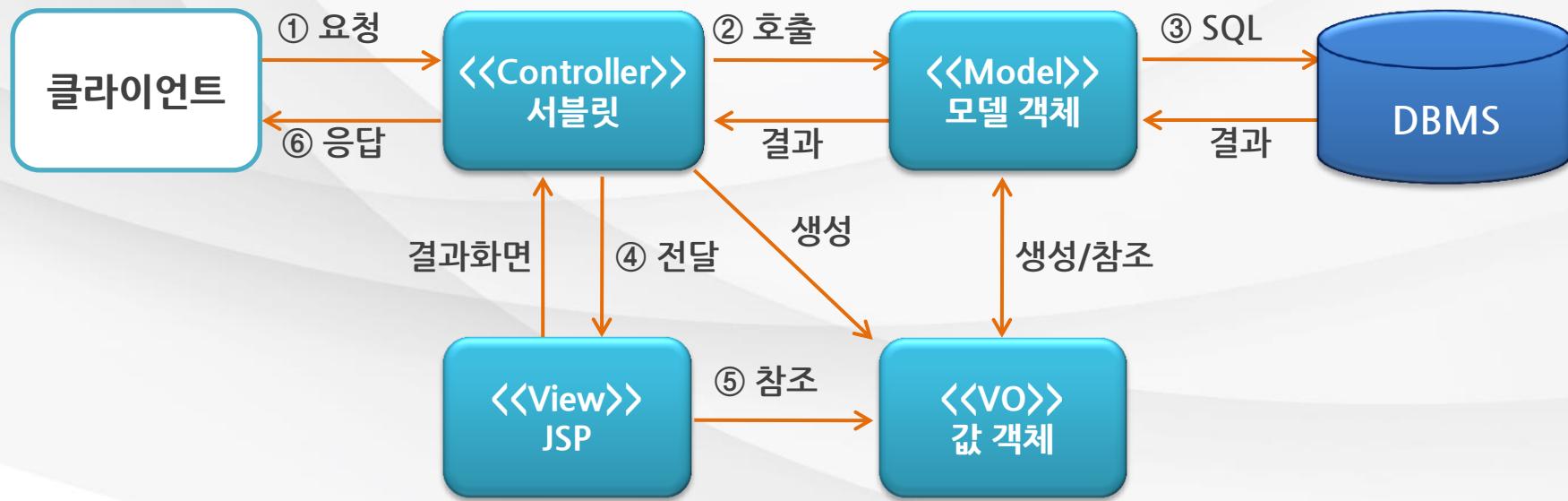
클라이언트가 보낸 데이터가 있다면, 모델을 호출할 때 전달하기 쉽게 데이터를 적절히 가공하는 일을 함

모델이 업무 수행을 완료하면, 그 결과를 가지고 화면을 생성하도록 뷰에게 전달(클라이언트 요청에 대해 모델과 뷰를 결정하여 전달)

Servlet과 JSP를 사용하여 작성할 수 있음

| 모델2 아키텍쳐 개념

- 모델 1 아키텍쳐 : Controller 역할을 JSP가 담당함
- 모델 2 아키텍쳐 : Controller 역할을 Servlet이 담당함



| 모델2 아키텍쳐 호출 순서



- ① 웹 브라우저가 웹 애플리케이션 실행을 요청하면, 웹 서버가 그 요청을 받아서 서블릿 컨테이너(ex : 톰캣서버)에 넘겨준다.
서블릿 컨테이너는 URL을 확인하여 그 요청을 처리할 서블릿을 찾아서 실행한다.

| 모델2 아키텍쳐 호출 순서



② 서블릿은 실제 업무를 처리하는 모델 자바 객체의 메서드를 호출한다.

만약 웹 브라우저가 보낸 데이터를 저장하거나 변경해야 한다면 그 데이터를 가공하여 VO 객체(Value Object)를 생성하고, 모델 객체의 메서드를 호출할 때 인자 값으로 넘긴다.

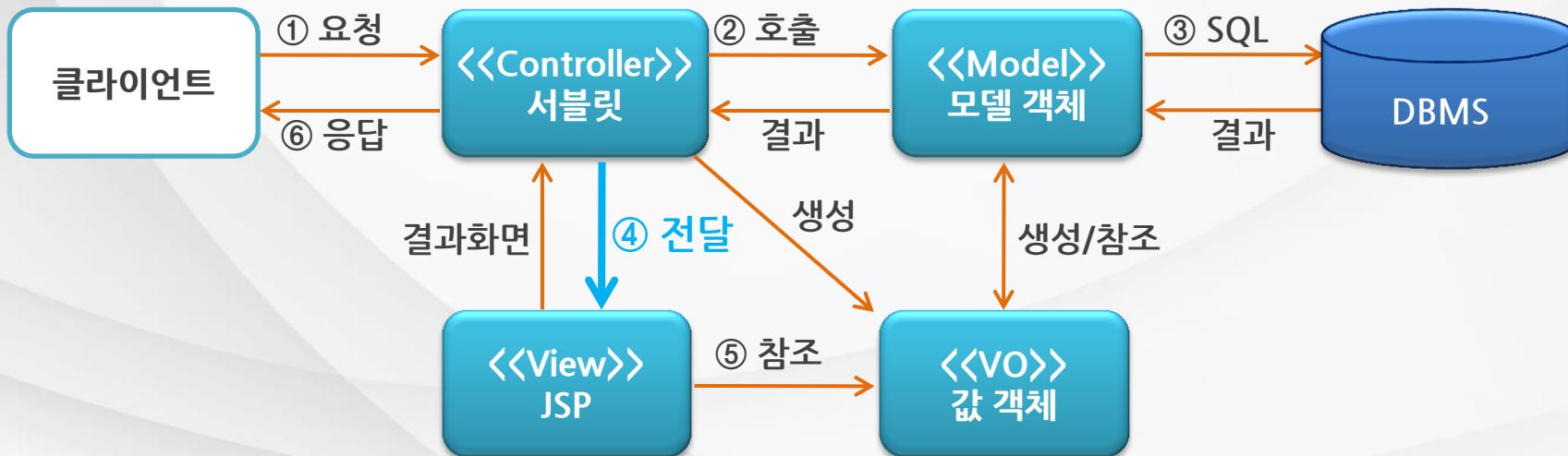
모델 객체는 엔터프라이즈 자바빈(EJB; Enterprise JavaBeans)일 수도 있고,
일반 자바 객체(POJO로 된 Service, DAO object) 일 수도 있다.

| 모델2 아키텍쳐 호출 순서



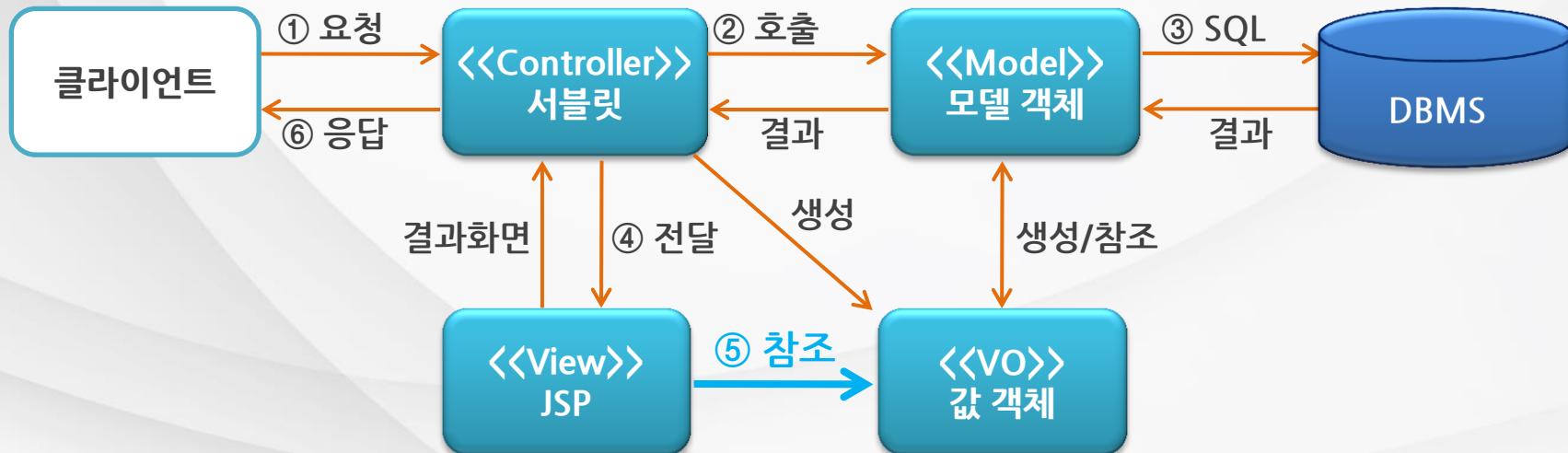
③ 모델 객체는 JDBC를 사용하여 매개변수로 넘어온 값 객체를 데이터베이스에 저장하거나, 데이터베이스로부터 질의 결과를 가져와서 VO 객체로 만들어 반환한다.

| 모델2 아키텍쳐 호출 순서



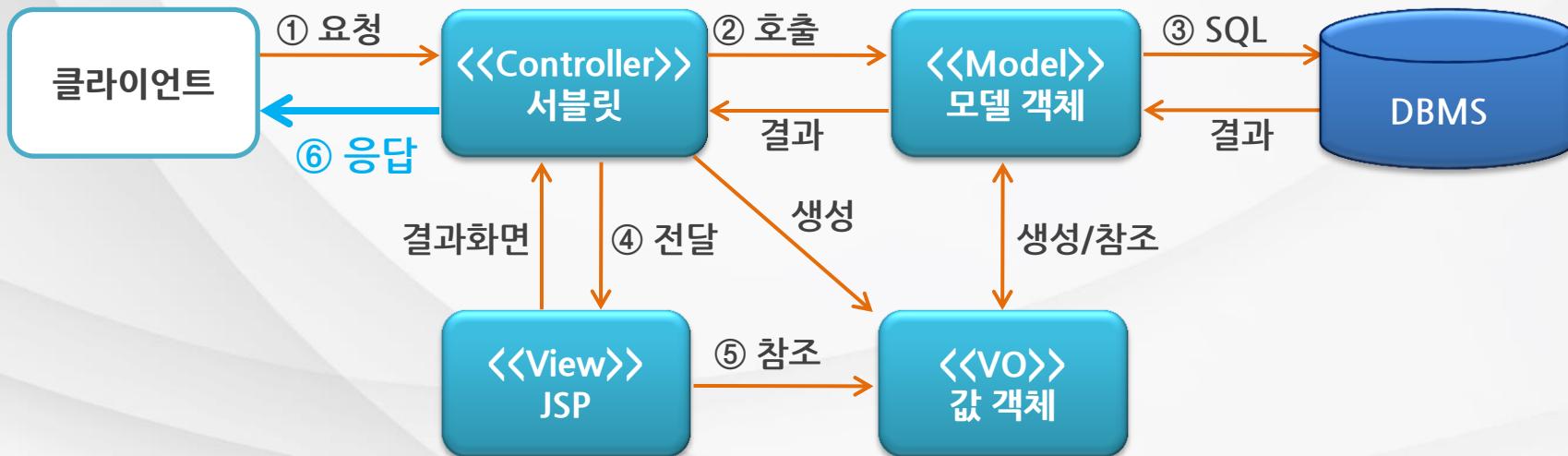
- ④ 서블릿은 모델 객체로부터 반환 받은 값을 JSP에 전달한다.

| 모델2 아키텍쳐 호출 순서



- ⑤ JSP는 서블릿으로부터 전달받은 값 객체를 참조하여 웹 브라우저가 출력할 결과 화면을 만들고, 웹 브라우저에 출력함으로써 요청 처리를 완료한다.

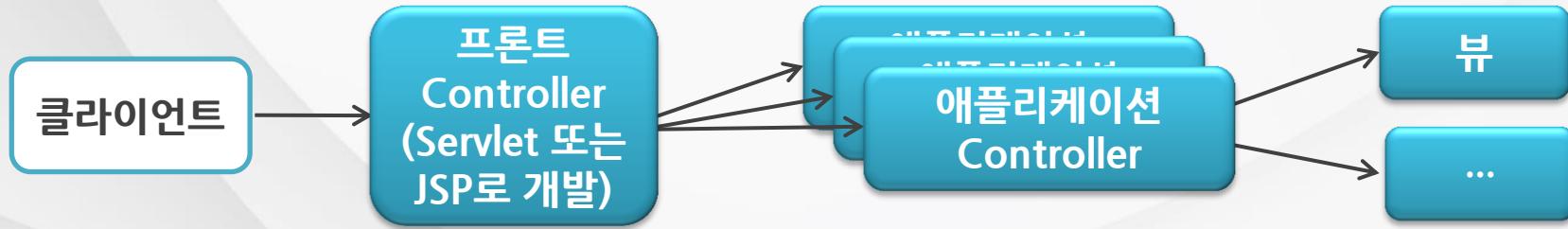
| 모델2 아키텍쳐 호출 순서



⑥ 웹 브라우저는 서버로부터 받은 응답 내용을 화면에 출력한다.

I Front Controller 패턴 아키텍쳐

❖ Front Controller 프로세스



- Front Controller는 클라이언트가 보낸 요청을 받아서 공통적인 작업을 먼저 수행
- Front Controller는 적절한 세부 Controller에게 작업을 위임
- 각각의 애플리케이션 Controller는 클라이언트에게 보낼 뷰를 선택해서 최종 결과를 생성하는 작업
- Front Controller 패턴은 인증이나 권한 체크처럼 모든 요청에 대하여 공통적으로 처리해야 하는 로직이 있을 경우 전체적으로 클라이언트의 요청을 중앙 집중적으로 관리하고자 할 경우에 사용

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular bokeh lights in shades of yellow, orange, and blue.

2. Spring MVC 개념

| Spring MVC의 특징

Spring은 DI 나 AOP 같은 기능뿐만 아니라,
서블릿 기반의 웹 개발을 위한 MVC 프레임워크를 제공

Spring MVC는 모델2 아키텍쳐와 Front Controller 패턴을
프레임워크 차원에서 제공

Spring MVC 프레임워크는 Spring을 기반으로 하고 있기 때문에
Spring이 제공하는 트랜잭션 처리나 DI 및 AOP 등을 손쉽게 사용

| Spring MVC와 Front Controller 패턴

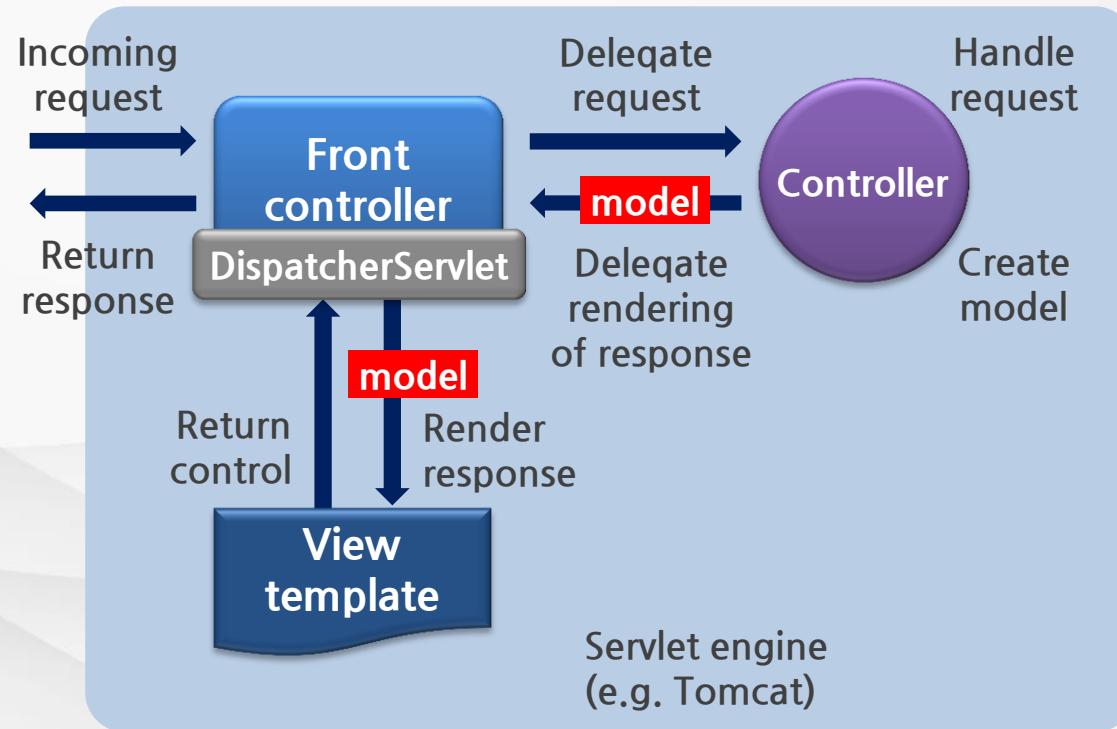
대부분의 MVC 프레임워크들은 Front Controller 패턴을 적용해서 구현

Spring MVC도 Front Controller 역할을 하는 DispatcherServlet이라는 클래스를 계층의 맨 앞단에 놓고, 서버로 들어오는 모든 요청을 받아서 처리하도록 구성

예외가 발생했을 때 일관된 방식으로 처리하는 것도
Front Controller의 역할

I DispatcherServlet 클래스

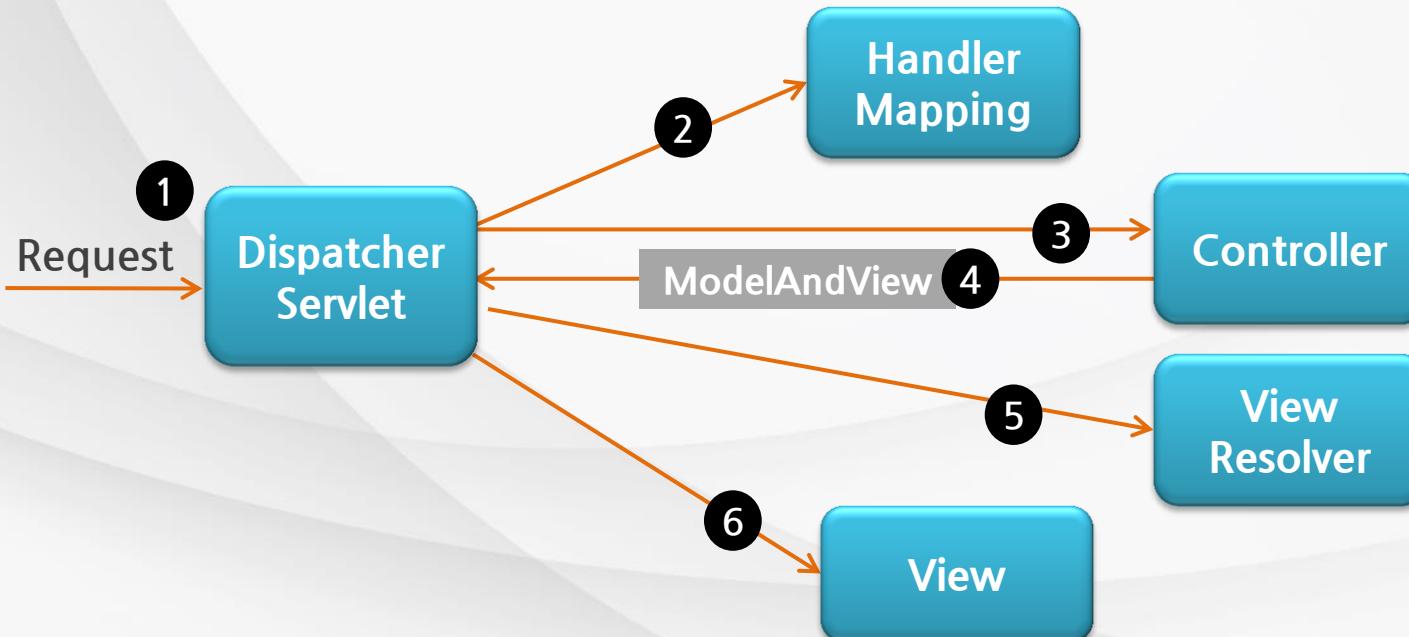
- Front Controller 패턴 적용
- web.xml에 설정
- 클라이언트로부터의 모든 요청을 전달 받음
- Controller나 View와 같은 Spring MVC의 구성요소를 이용하여 클라이언트에게 서비스를 제공



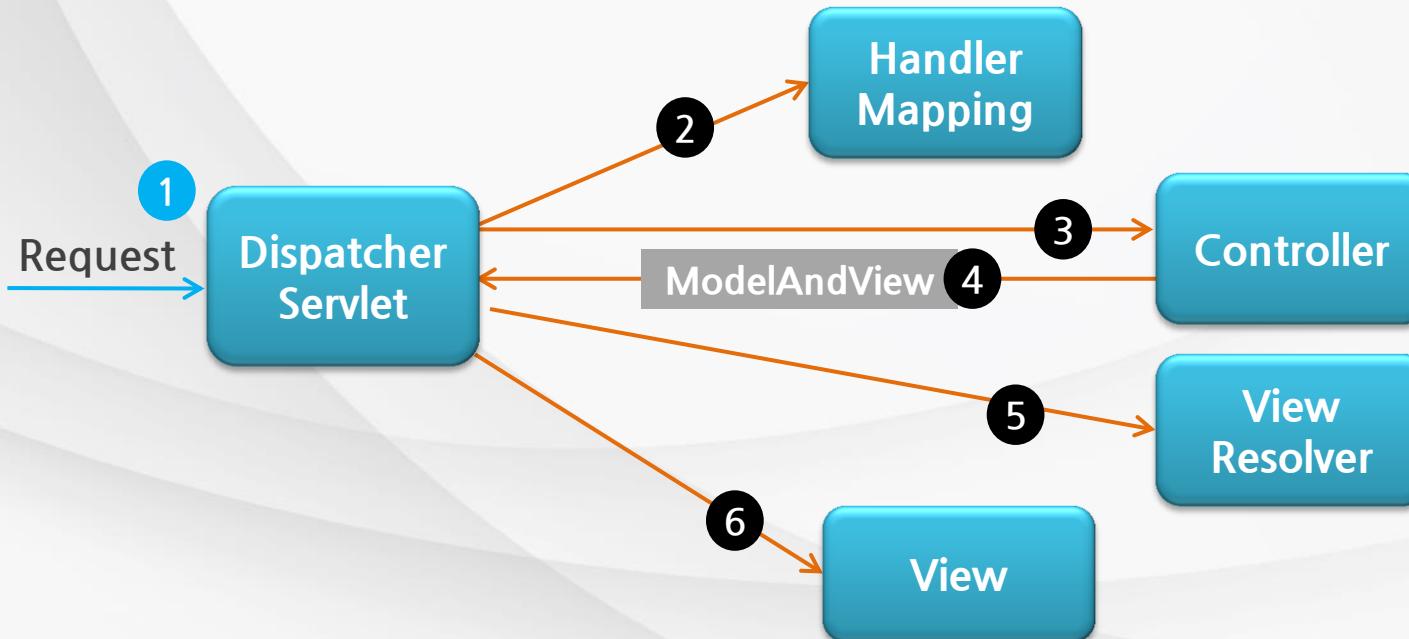
| Spring MVC의 주요 구성 요소

구성요소	설명
DispatcherServlet	클라이언트의 요청을 받아서 Controller에게 클라이언트의 요청을 전달하고, 리턴한 결과값을 View에게 전달하여 알맞은 응답을 생성
HandlerMapping	URL과 요청 정보를 기준으로 어떤 핸들러 객체를 사용할지 결정하는 객체이며, DispatcherServlet은 하나 이상의 핸들러 매팅을 가질 수 있음
Controller	클라이언트의 요청을 처리한 뒤, Model을 호출하고 그 결과를 DispatcherServlet에게 알려 줌
ModelAndView	Controller가 처리한 데이터 및 화면에 대한 정보를 보유한 객체
View	Controller의 처리 결과 화면에 대한 정보를 보유한 객체
ViewResolver	Controller가 리턴한 뷰 이름을 기반으로 Controller 처리 결과를 생성할 뷰를 결정

| Spring MVC의 주요 구성 요소의 요청 처리 과정

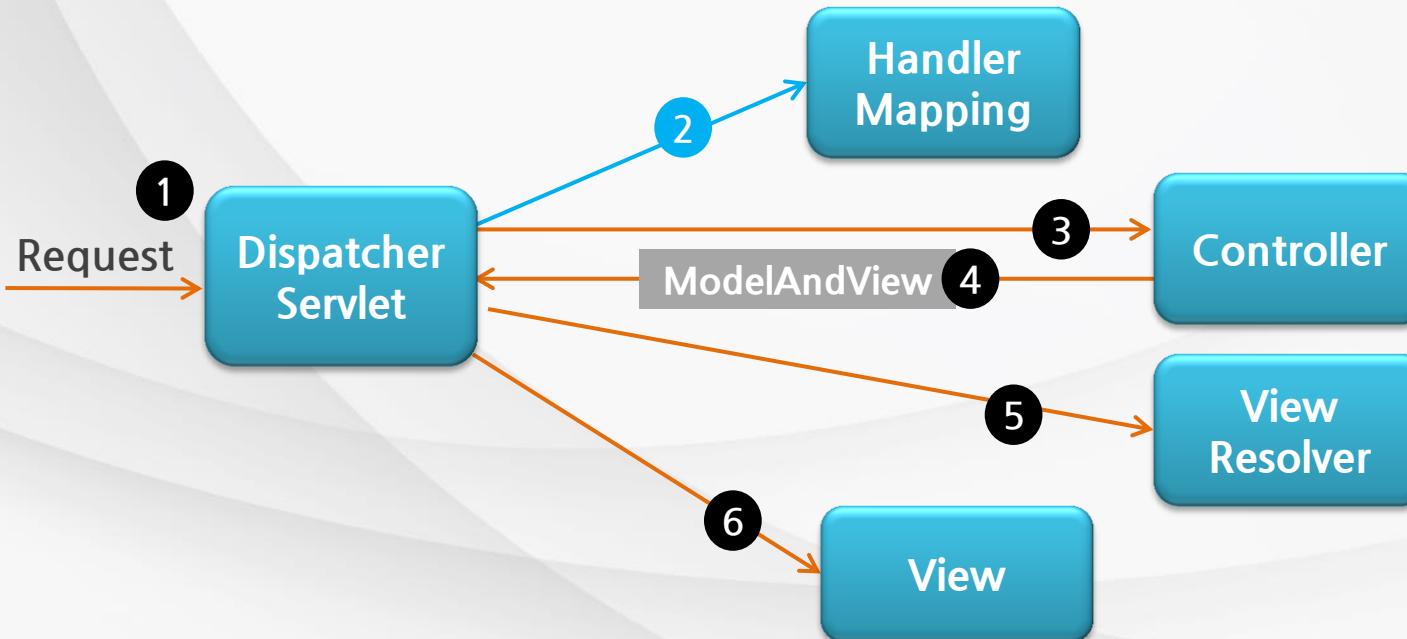


| Spring MVC의 주요 구성 요소의 요청 처리 과정



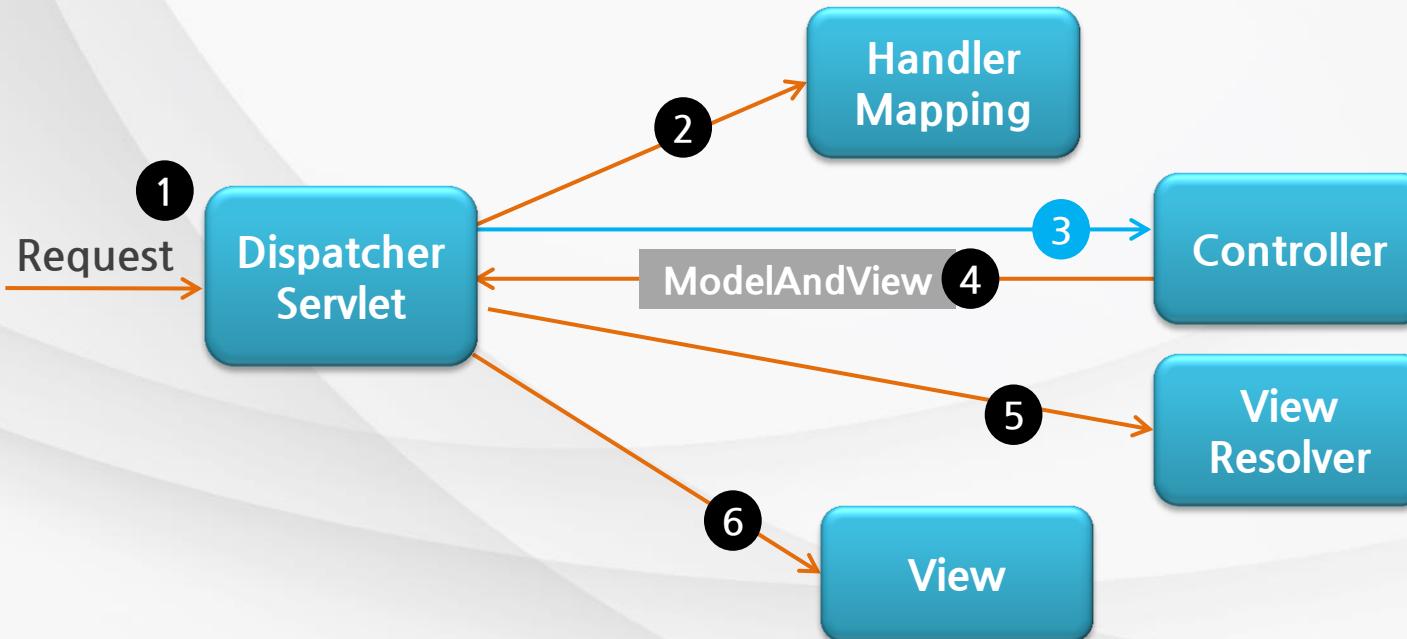
- ① 클라이언트의 요청이 DispatcherServlet에게 전달된다.

| Spring MVC의 주요 구성 요소의 요청 처리 과정(2)



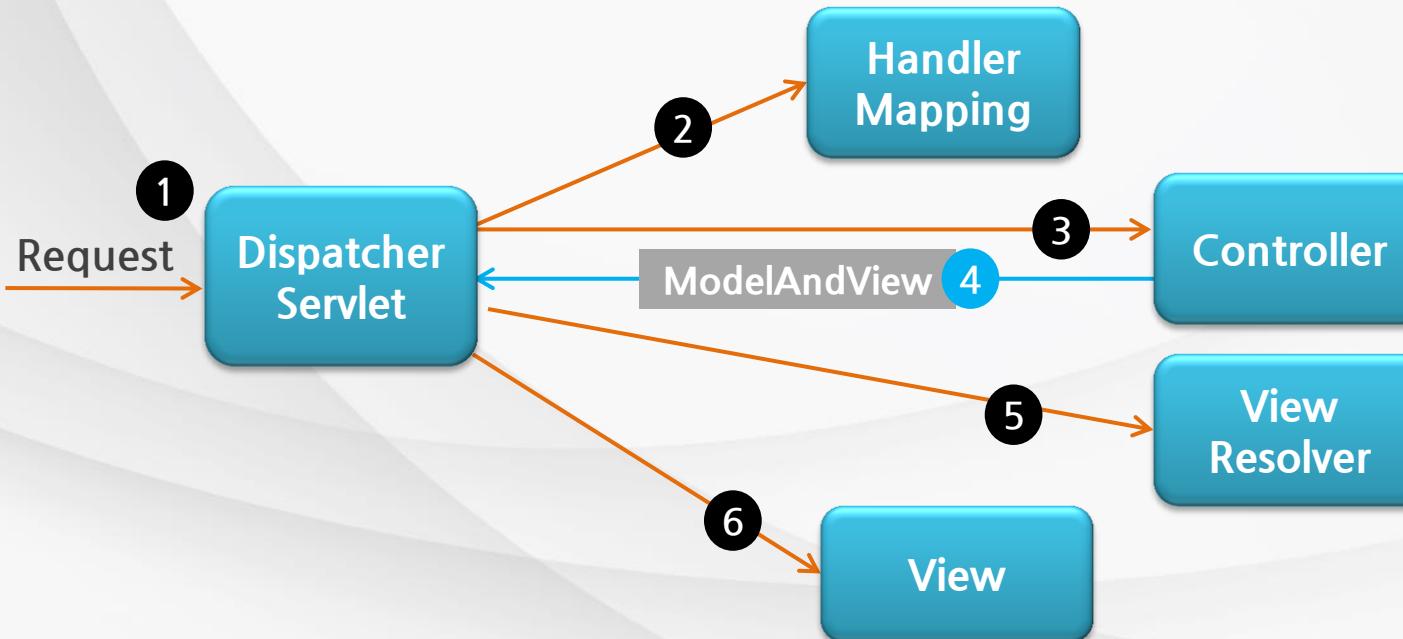
- ② DispatcherServlet은 HandlerMapping을 사용하여 클라이언트의 요청을 처리할 Controller를 획득한다.

| Spring MVC의 주요 구성 요소의 요청 처리 과정(2)



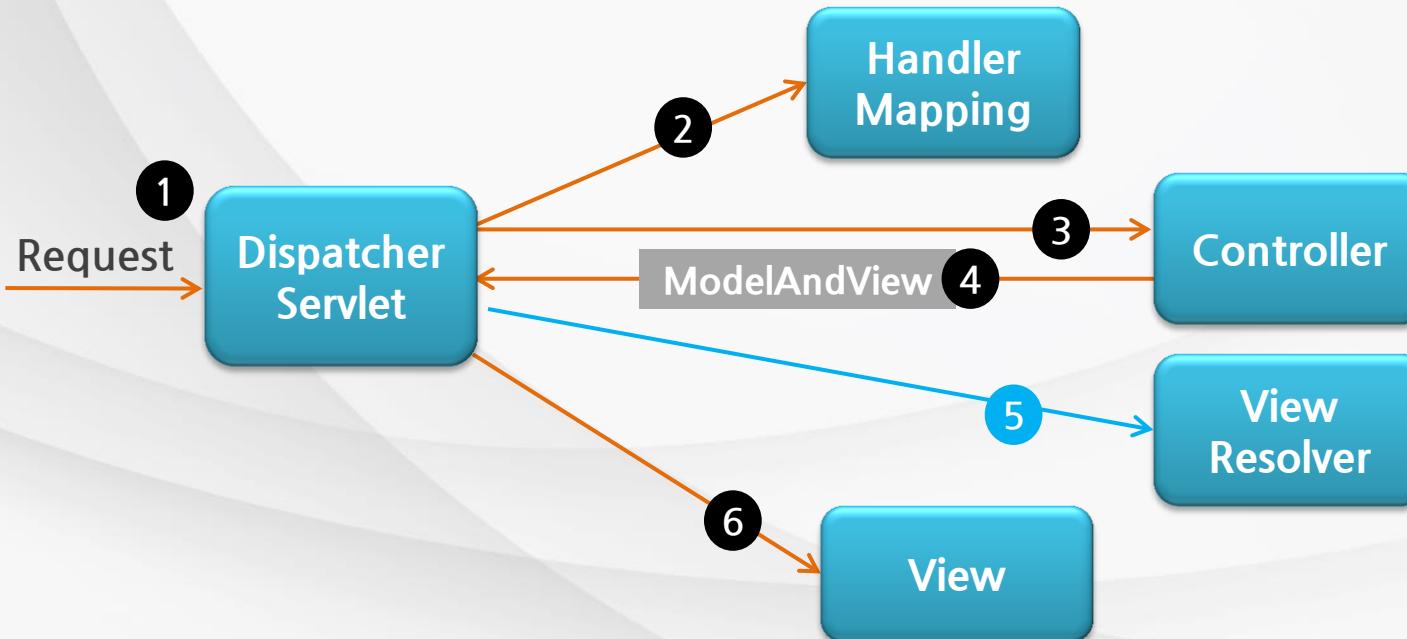
- ③ DispatcherServlet은 Controller 객체를 이용하여 클라이언트의 요청을 처리한다.

| Spring MVC의 주요 구성 요소의 요청 처리 과정(2)



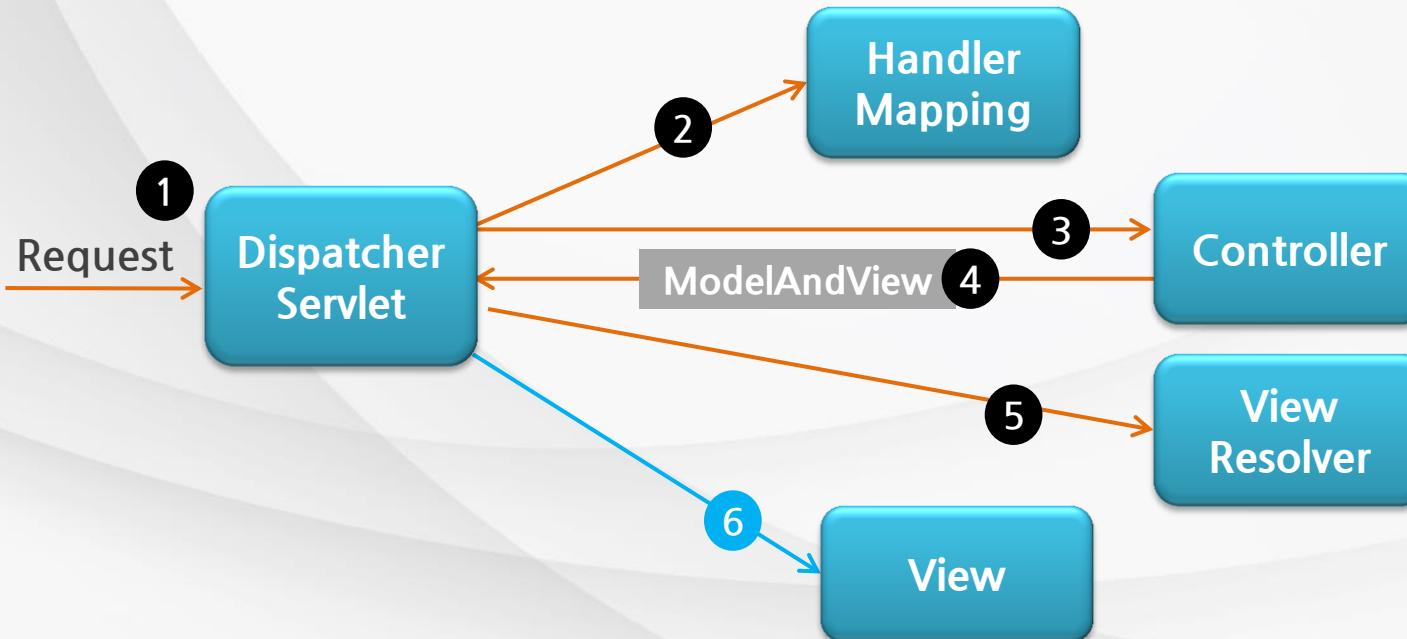
- ④ Controller는 클라이언트 요청 처리 결과와 View 페이지 정보를 담은 ModelAndView 객체를 반환한다.

| Spring MVC의 주요 구성 요소의 요청 처리 과정



- ⑤ DispatcherServlet은 ViewResolver로부터 응답 결과를 생성할 View 객체를 구한다.

| Spring MVC의 주요 구성 요소의 요청 처리 과정



⑥ View는 클라이언트에게 전송할 응답을 생성한다.



3. Spring MVC 기반 웹어플리케이션 개발

| Spring MVC기반 웹 어플리케이션 작성 절차

- ① 클라이언트의 요청을 받는 DispatcherServlet를 web.xml에 설정
- ② 클라이언트의 요청을 처리할 Controller를 작성
- ③ Spring Bean으로 Controller를 등록
- ④ JSP를 이용한 View 영역의 코드를 작성
- ⑤ Browser 상에서 JSP를 실행

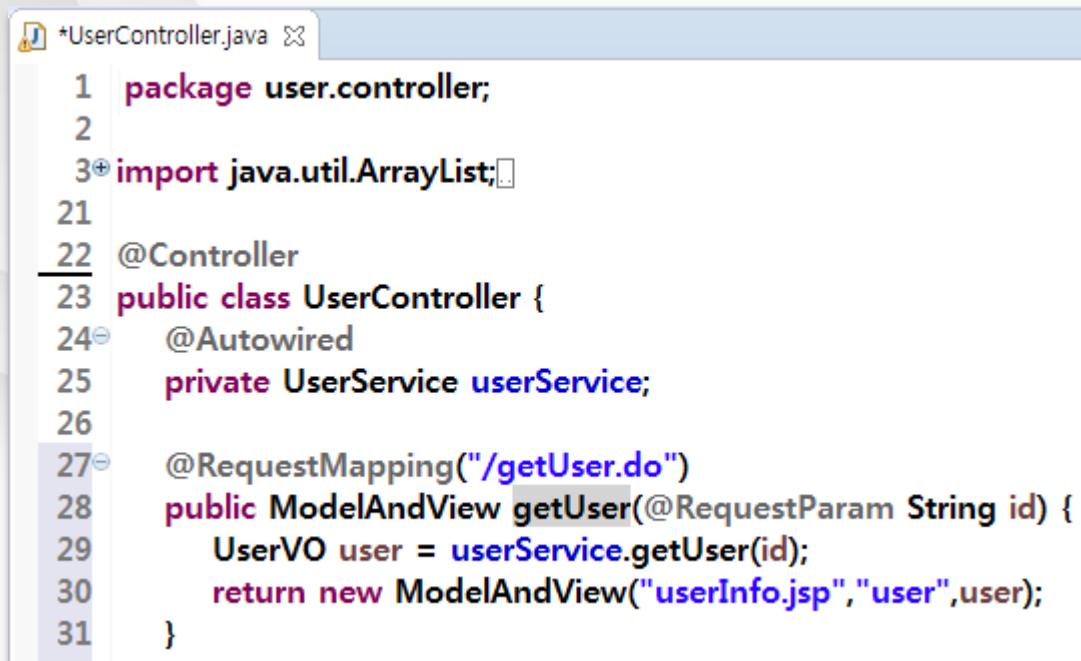
| Spring MVC기반 웹 어플리케이션 작성 절차(1)

① DispatcherServlet을 web.xml에 설정

```
*web.xml
35    <!-- The front controller of this Spring Web application, responsible for handling all
36    <servlet>
37        <servlet-name>springDispatcherServlet</servlet-name>
38        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
39    <init-param>
40        <param-name>contextConfigLocation</param-name>
41        <param-value>classpath:/config/beans*.xml</param-value>
42    </init-param>
43    <load-on-startup>1</load-on-startup>
44 </servlet>
45
46    <!-- Map all requests to the DispatcherServlet for handling -->
47    <servlet-mapping>
48        <servlet-name>springDispatcherServlet</servlet-name>
49        <url-pattern>*.do</url-pattern>
50    </servlet-mapping>
51 </web-app>
```

| Spring MVC기반 웹 어플리케이션 작성 절차(2)

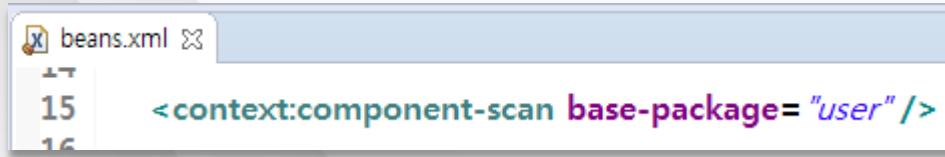
② 클라이언트의 요청을 처리할 Controller



```
*UserController.java
1 package user.controller;
2
3 import java.util.ArrayList;
4
5
6 @Controller
7 public class UserController {
8     @Autowired
9     private UserService userService;
10
11     @RequestMapping("/getUser.do")
12     public ModelAndView getUser(@RequestParam String id) {
13         UserVO user = userService.getUser(id);
14         return new ModelAndView("userInfo.jsp", "user", user);
15     }
16 }
```

| Spring MVC기반 웹 어플리케이션 작성 절차(3)

③ Spring Bean으로 Controller를 등록



A screenshot of a code editor window titled "beans.xml". The code in the editor is:

```
beans.xml
15   <context:component-scan base-package= "user" />
16
```

The XML declaration is highlighted in blue, and the attribute "base-package" is highlighted in purple.

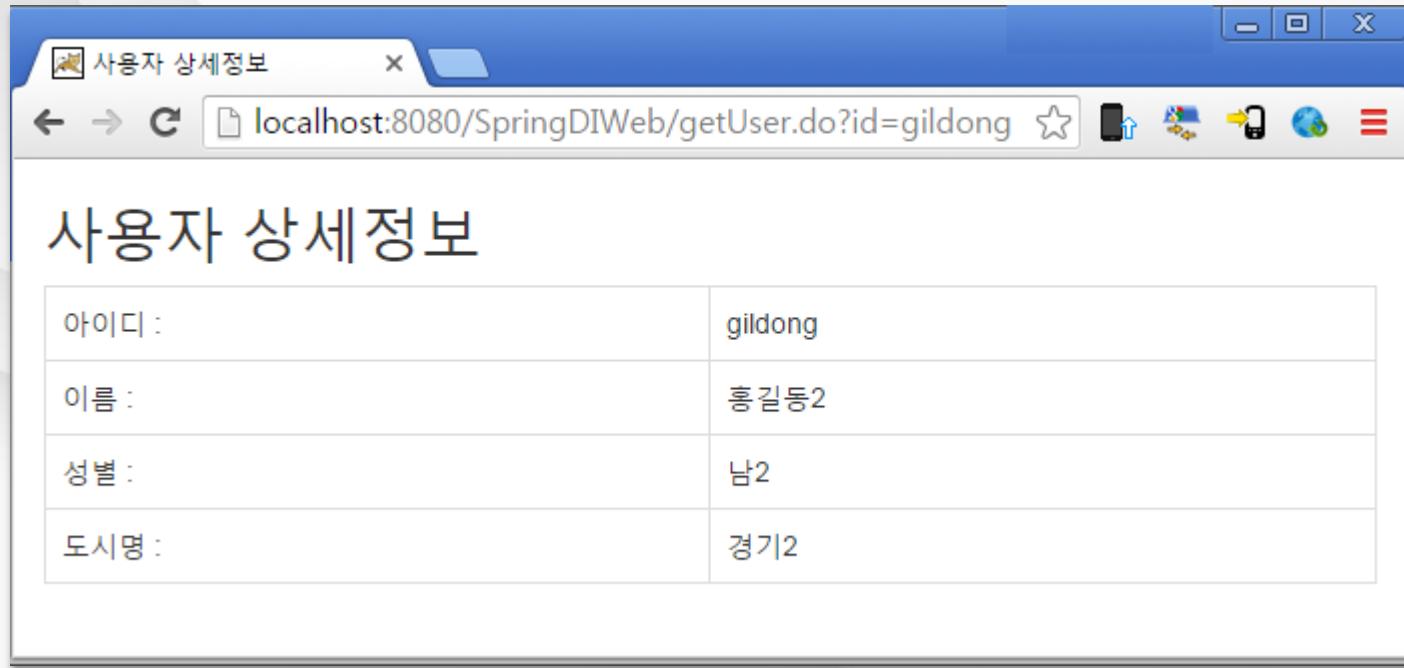
| Spring MVC기반 웹 어플리케이션 작성 절차(4)

④ JSP를 이용한 View 영역의 코드를 작성

```
*userInfo.jsp
3<html>
4<head>
5<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
6<title>사용자 상세정보</title>
7</head>
8<body>
9<div class="container">
10<h2>사용자 상세정보</h2>
11<table class="table table-bordered table-hover">
12<tr><td>아이디 :</td><td>${user.userId}</td></tr>
13<tr><td>이름 :</td><td>${user.name}</td></tr>
14<tr><td>성별 :</td><td>${user.gender}</td></tr>
15<tr><td>도시명 :</td><td>${user.city}</td></tr>
16</table>
17</div>
18</body>
19</html>
```

| Spring MVC기반 웹 어플리케이션 작성 절차(5)

- ⑤ Browser상에서 JSP를 실행한다.



A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring MVC 개요]에 대해서 살펴보았습니다.

MVC 패턴의 개념과 모델2 아키텍쳐

- MVC 패턴의 개념 및 각 컴포넌트의 역할
- 모델2 아키텍쳐의 개념

Spring MVC 개요

- Spring MVC의 특징
- Spring MVC의 주요 구성요소와 요청 처리 과정

Spring MVC 기반 웹어플리케이션 개발

- Spring MVC 기반 웹 어플리케이션 작성 절차

Spring Framework

20. Spring MVC 환경설정

CONTENTS

- 1 Server 등록 및 Browser 설정
- 2 Dynamic Web Project 작성
- 3 Spring MVC 설정

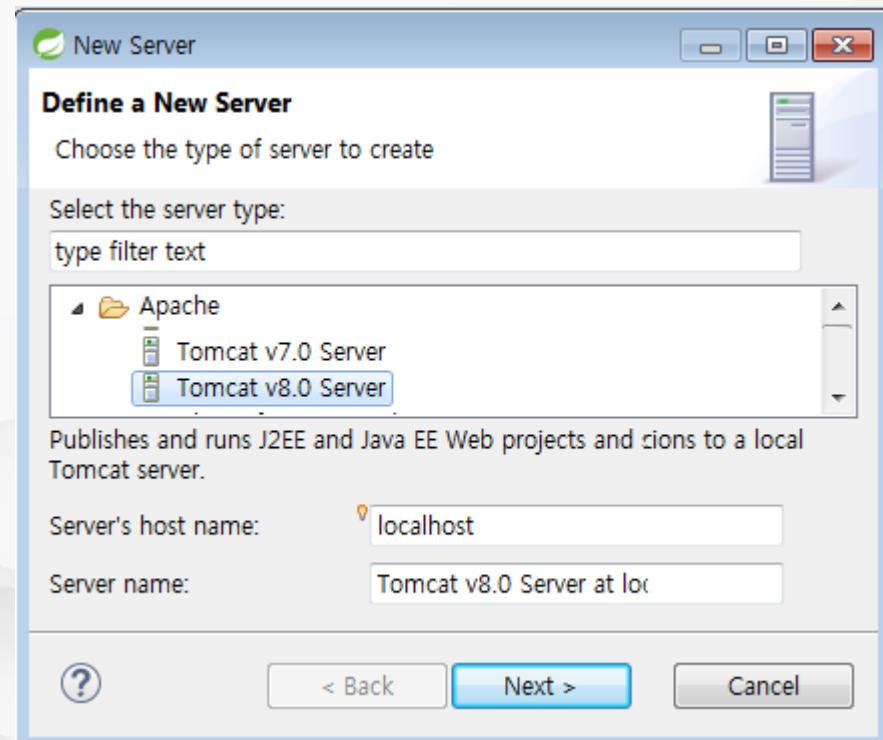
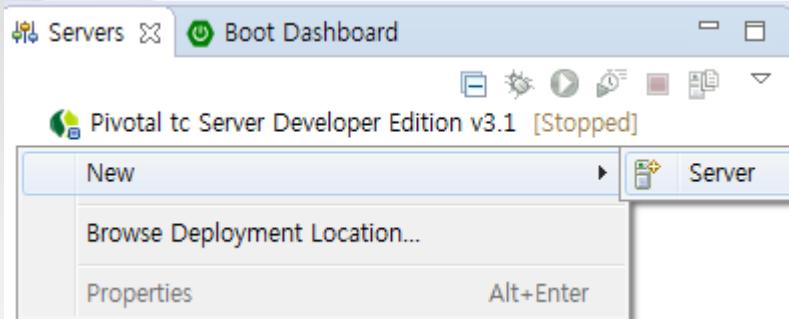
학습 목표

- Server 등록 및 Browser 설정에 대하여 이해할 수 있습니다.
- Dynamic Web Project 작성에 대하여 이해할 수 있습니다.
- Spring MVC 설정에 대하여 이해할 수 있습니다.

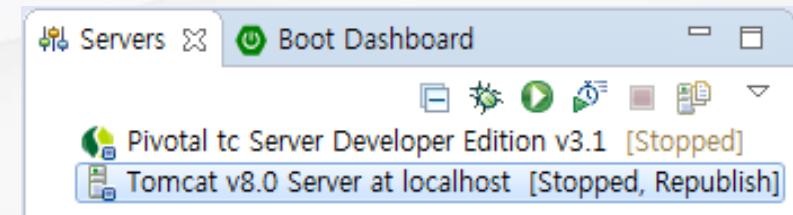
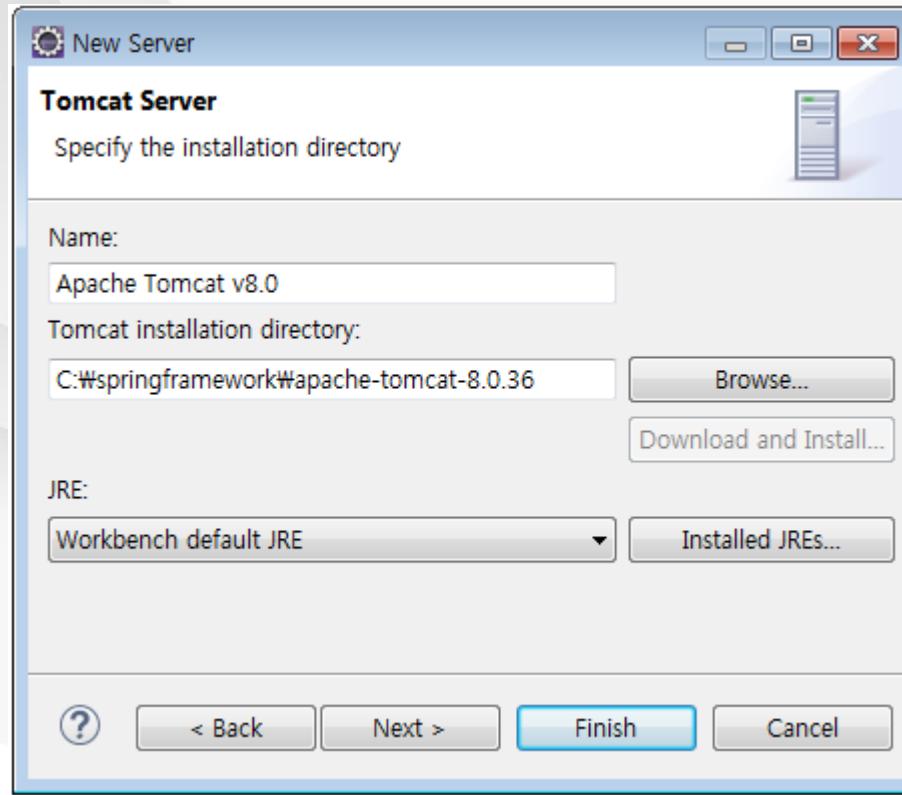
A photograph of a person's hands holding a black smartphone. The person is wearing a dark long-sleeved shirt. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

1. Server 등록 및 Browser 설정

Server 등록(1)

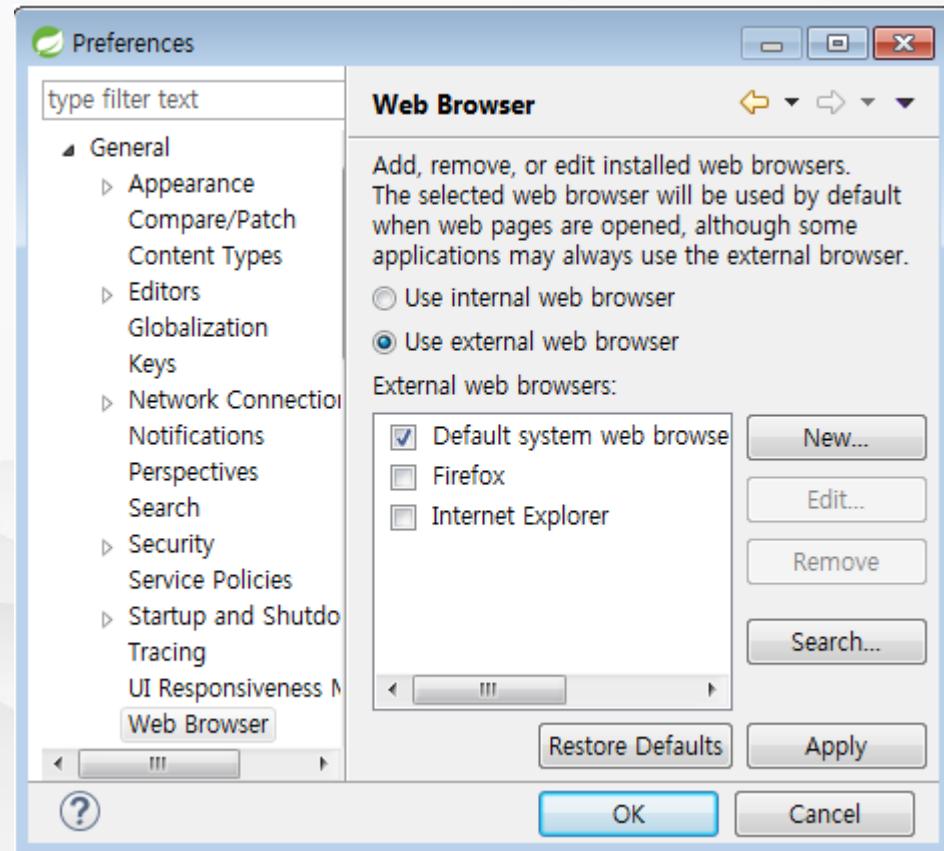


Server 등록(2)

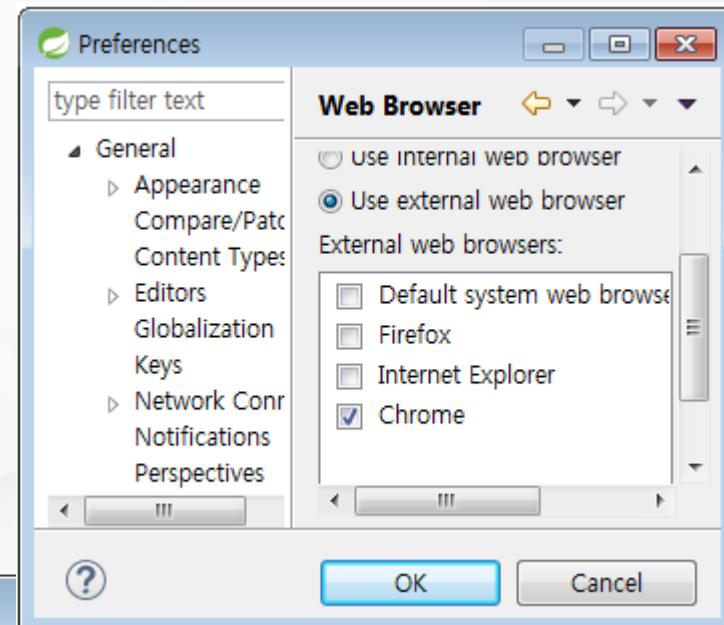
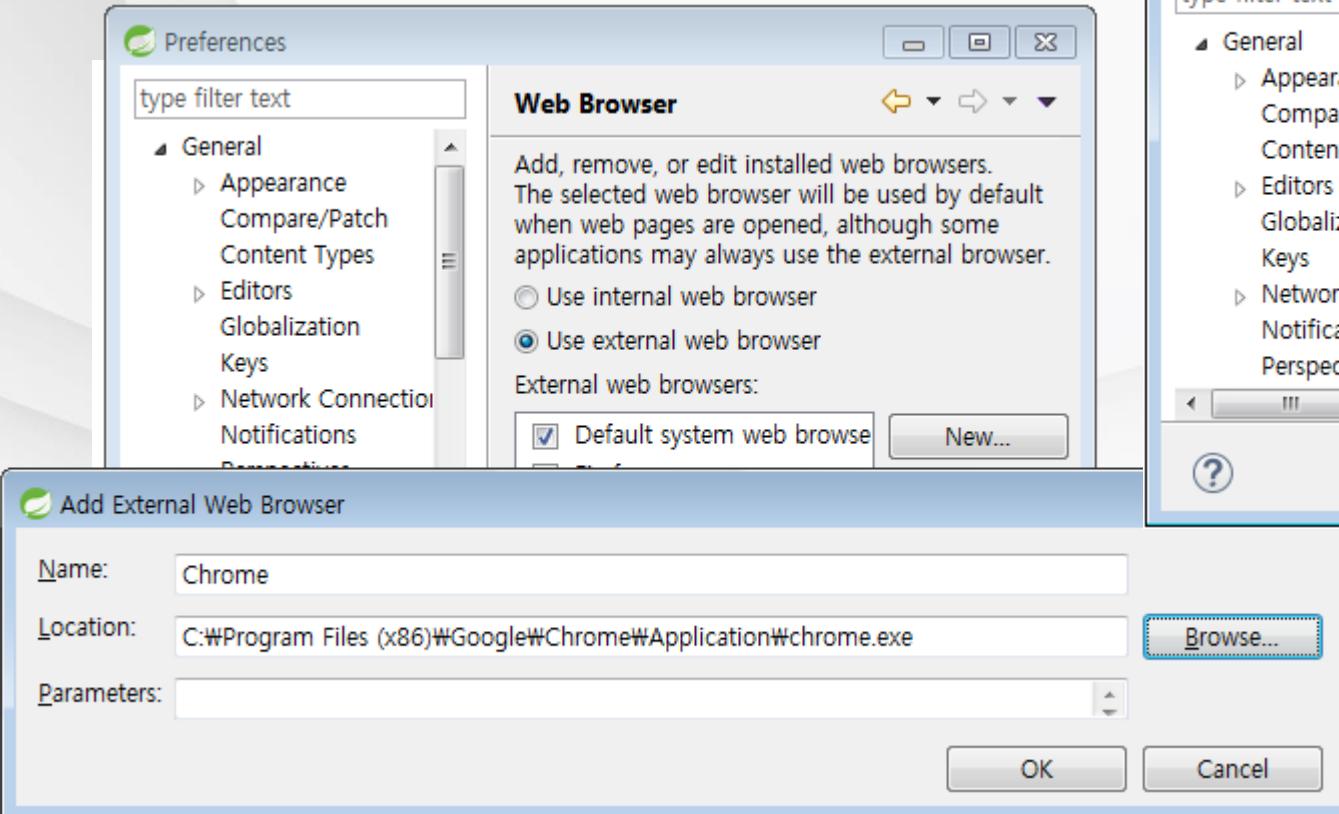


Chrome Browser 설정(1)

- Window -> Preferences -> General -> Web Browser 선택
- Use external web browser 선택



Chrome Browser 설정(2)

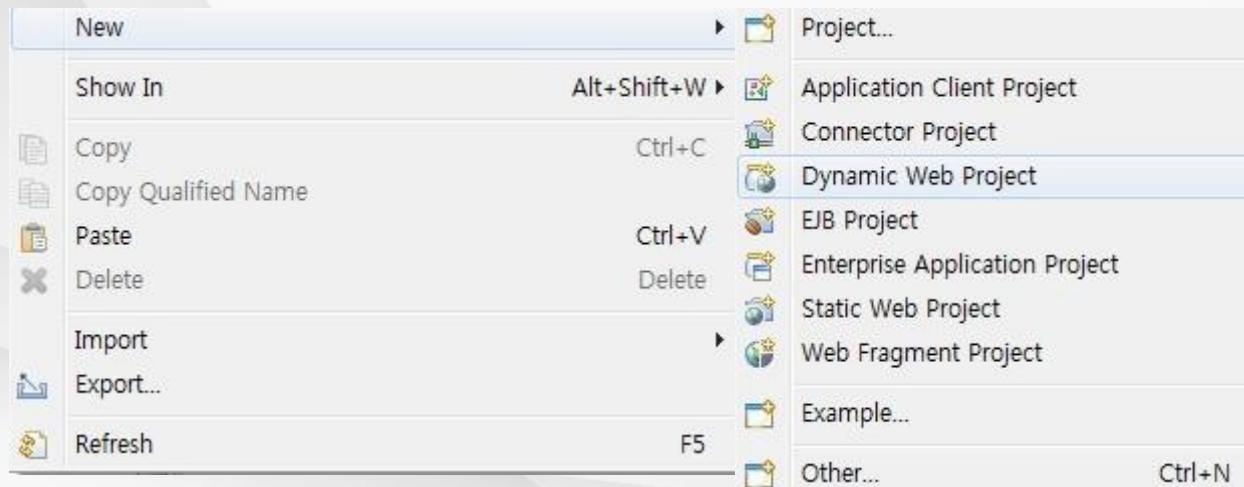


A photograph of a person's hands holding a black smartphone. The person is wearing a dark long-sleeved shirt. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

2. Dynamic Web Project

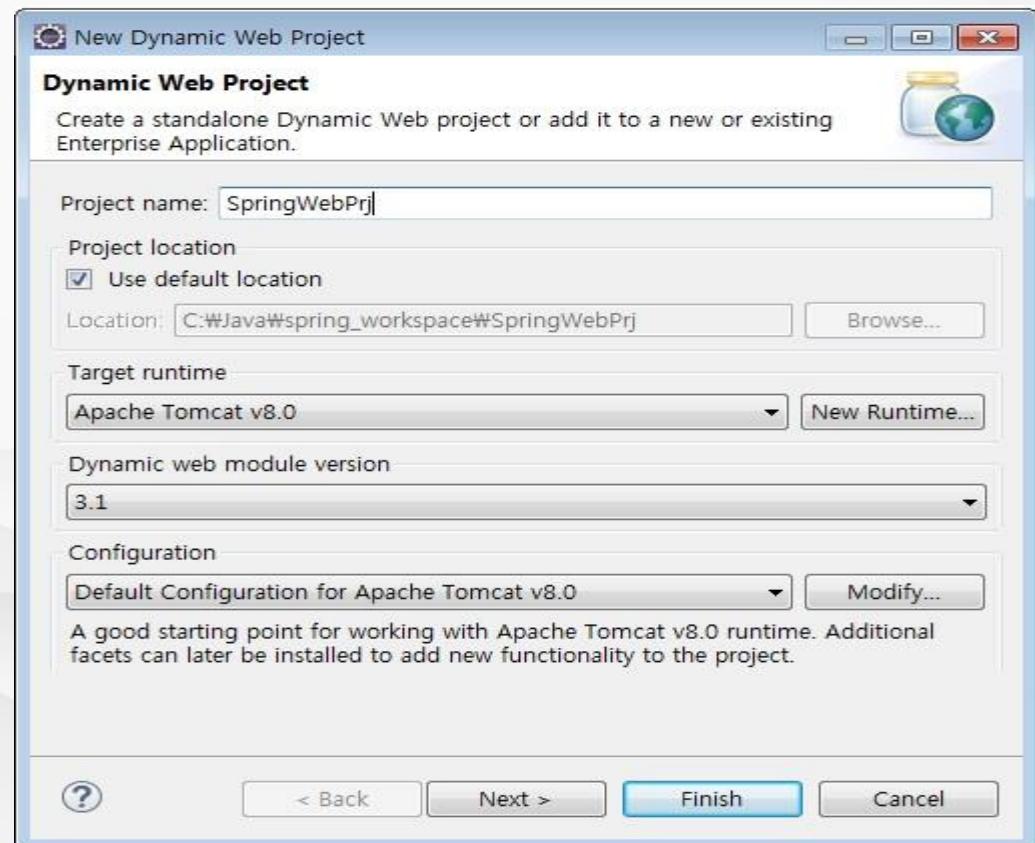
I Dynamic Web Project 작성(1)

- ◎ New -> Dynamic Web Project 선택



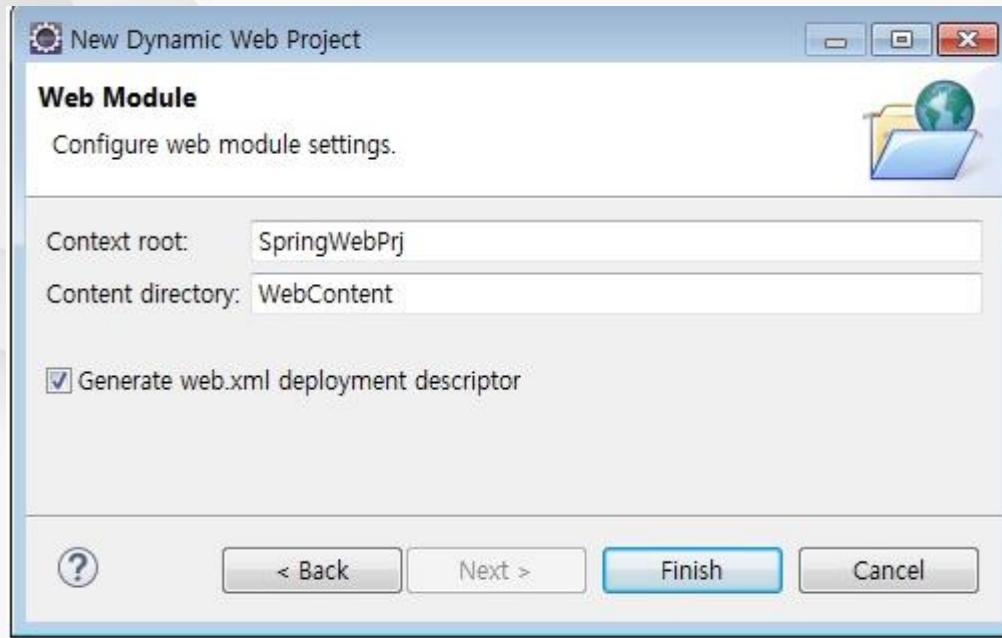
I Dynamic Web Project 작성(2)

- Project Name => SpringWebPrj
- Target runtime => Tomcat 8.0



I Dynamic Web Project 작성(3)

- ◎ web.xml 생성 되도록 선택

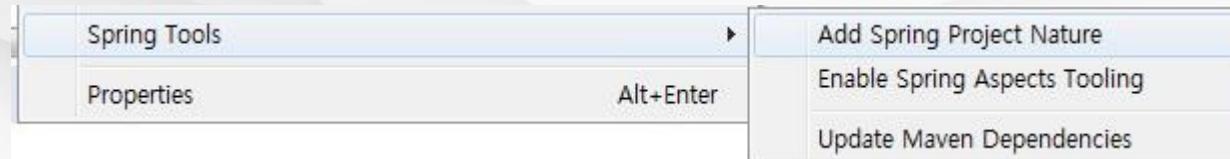


I Dynamic Web Project 작성(4)

- ◎ Maven Project로 변환



- ◎ Spring Project Nature 추가



I 기존 Project의 Source 와 설정파일 복사

- 기존에 작성한 Project의 pom.xml의 dependency 복사

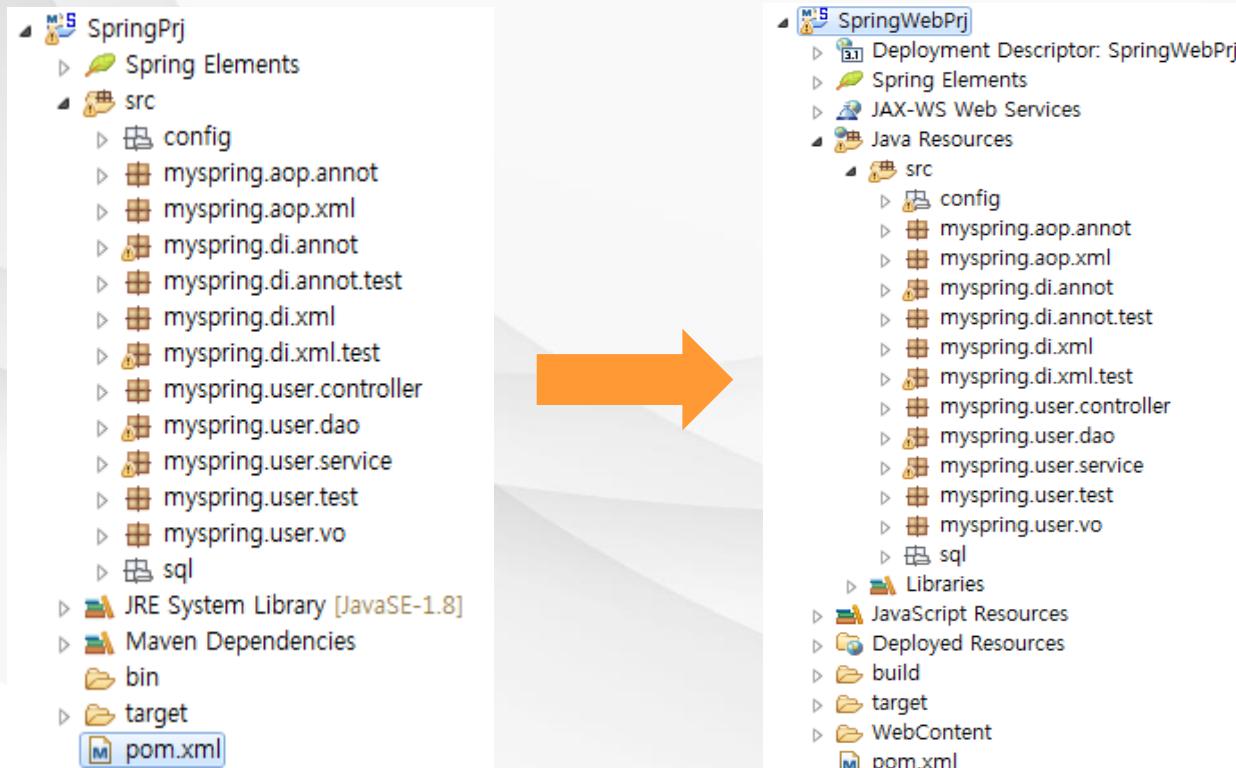
```
SpringPrj/pom.xml
20<dependencies>
21    <!-- http://mvnrepository.com -->
22    <dependency>
23        <groupId>org.springframework</groupId>
24        <artifactId>spring-webmvc</artifactId>
25        <version>3.2.17.RELEASE</version>
26    </dependency>
27    <!-- https://mvnrepository.com -->
28    <dependency>
29        <groupId>junit</groupId>
30        <artifactId>junit</artifactId>
31        <version>4.12</version>
32    </dependency>
```



```
SpringWebPrj/pom.xml
28<dependencies>
29    <!-- http://mvnrepository.com -->
30    <dependency>
31        <groupId>org.springframework</groupId>
32        <artifactId>spring-webmvc</artifactId>
33        <version>3.2.17.RELEASE</version>
34    </dependency>
35    <!-- https://mvnrepository.com -->
36    <dependency>
37        <groupId>junit</groupId>
38        <artifactId>junit</artifactId>
39        <version>4.12</version>
40    </dependency>
```

| 기존 Project의 Source 와 설정파일 복사

- ◎ 기존에 작성한 Project의 java package 디렉토리와 config, sql 디렉토리 복사



A photograph of a person's hands holding a black smartphone. The person is wearing a dark long-sleeved shirt. The background is dark with blurred, colorful circular lights (bokeh) in shades of yellow, orange, and blue.

3. Spring MVC 설정

| Spring mvc 라이브러리 검색 및 설치

http://mvnrepository.com에 접근

spring mvc로 검색

spring mvc 3.2.17 버전을 pom.xml에 추가

```
<!-- http://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>3.2.17.RELEASE</version>
</dependency>
```

■ web.xml : ContextLoaderListener 클래스 설정

The screenshot shows an IDE interface with two tabs open: one for the main `web.xml` file and another for a list of XML declarations.

The main `web.xml` tab displays the following configuration:

```
<!-- needed for ContextLoaderListener -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:/config/beans.xml</param-value>
</context-param>

<!-- Bootstraps the root web application context before servlets -->
<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
```

An orange arrow points from the bottom of the code editor towards the declaration of the `ContextLoaderListener`.

The second tab lists various XML declarations, with the entry for `ContextLoaderListener` highlighted:

- ↳ security-constraint
- ↳ security-role
- ↳ service-ref
- ↳ servlet
- ↳ servlet-mapping
- ↳ session-config
- ↳ welcome-file-list
- # comment - xml comment
- # contextloaderlistener - ContextLoaderListener
- # dispatcherservlet - DispatcherServlet declaration
- # XSL processing instruction - XSL processing instruction

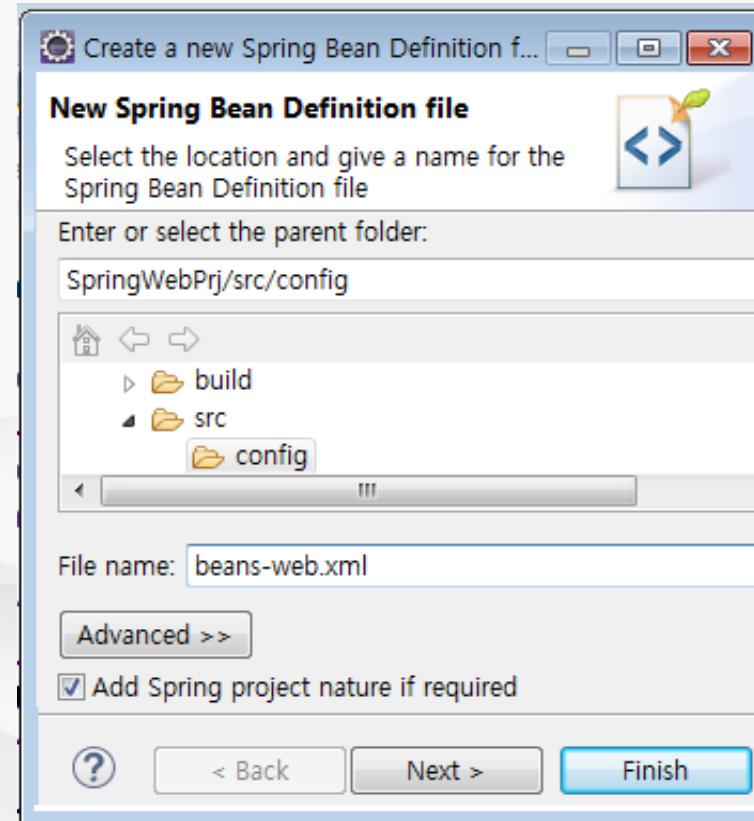
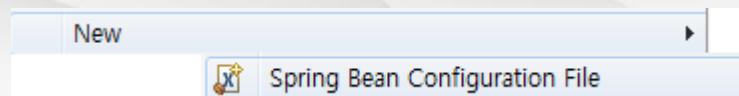
At the bottom of the interface, there is a message: "Press 'Ctrl+'".

■ Web 환경 설정을 위한 Spring Beans 파일 작성(1)(beans-web.xml)

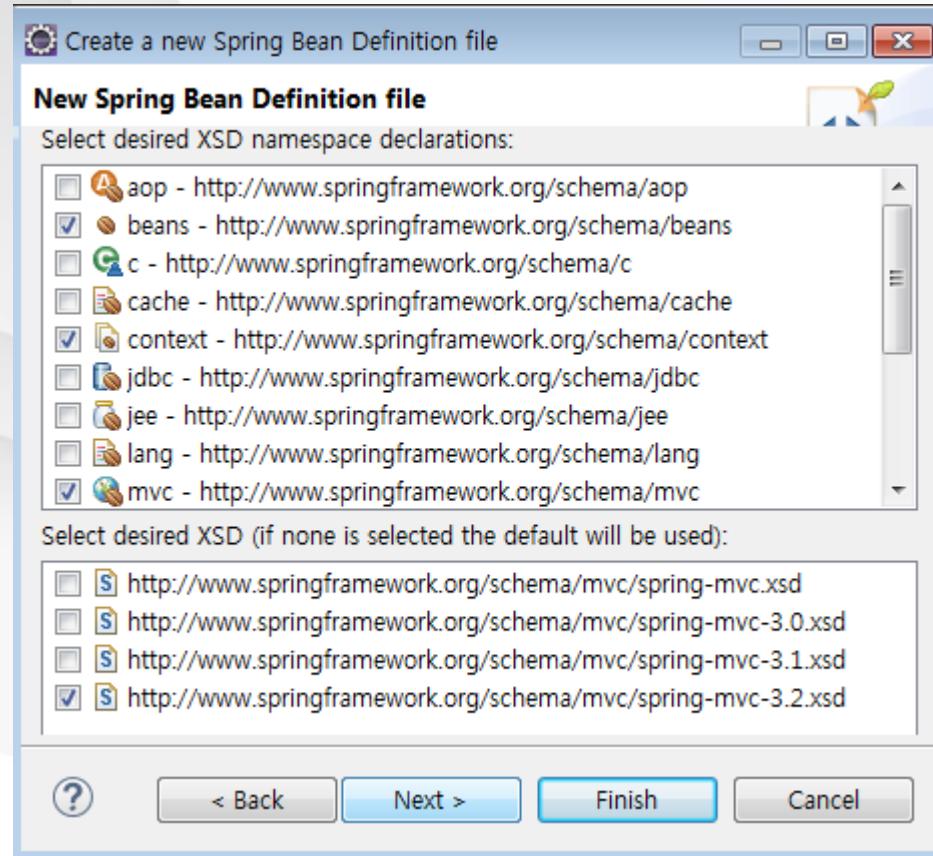
- ◎ Spring Perspective로 전환



- ◎ New → Spring Bean Configuration File



■ Web 환경 설정을 위한 Spring Beans 파일 작성(2)(beans-web.xml)



Namespace 선택

- beans
- context
- mvc

beans-web.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:context="http://www.springframework.org/schema/context"
5   xmlns:mvc="http://www.springframework.org/schema/mvc"
6   xsi:schemaLocation='
7     http://www.springframework.org/schema/beans
8     http://www.springframework.org/schema/context
9     http://www.springframework.org/schema/mvc
10    http://www.springframework.org/schema/context-3.2.xsd
11  </beans>
```

■ web.xml : DispatcherServlet 클래스 설정

```
*web.xml X
<> security-constraint
<> security-role
<> service-ref
<> servlet
<> servlet-mapping
<> session-config
<> welcome-file-list
# comment - xml comment
# contextloaderlistener - ContextLoaderListener
# dispatcherservlet - DispatcherServlet declaration
# XSL processing instruction - XSL processing instruction
Press 'Ctrl+Shift+F' to search for selected text.
```



```
*web.xml X
<servlet>
    <servlet-name>springDispatcherServlet</servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            classpath:/config/beans*.xml
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<!-- Map all requests to the DispatcherServlet for handling -->
<servlet-mapping>
    <servlet-name>springDispatcherServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring MVC 환경설정]에 대해서 살펴보았습니다.

Server 등록 및 Browser 설정

- Tomcat 8.0 Server 등록
- Chrome Browser 자동 실행 되도록 Eclipse 내에 설정

Dynamic Web Project 작성

- Dynamic Web Project 작성
- 기존 Project의 Source와 설정 파일 복사

Spring MVC 설정

- Spring MVC 설치
- web.xml에 Spring MVC 관련 내용 설정

Spring Framework

21. Spring MVC 어플리케이션 작성(1)

CONTENTS

1

EL과 JSTL

2

Hello 컨트롤러 작성

3

사용자 목록 조회 컨트롤러 작성

학습 목표

- EL과 JSTL에 대하여 이해할 수 있습니다.
- Hello 컨트롤러 작성에 대하여 이해할 수 있습니다.
- 사용자 목록 조회 컨트롤러 작성에 대해 이해할 수 있습니다.

AOP는

1. EL과 JSTL

I EL(Expression Language)의 개요

EL과 JSTL(Java Standard Tag Library)을 사용하면 <% %>와 같은 스크립팅 태그를 JSP에서 없앨 수 있다.

EL 표현식은 중괄호({})로 묶고 앞에 달러(\$)기호를 붙이며, 도트 연산자를 사용한다.

EL은 저장 객체의 출력을 단순화 하는 용도로 사용되므로, 저장 객체를 출력할 때도 스크립팅을 전혀 쓰지 않는다.

예를 들어, <%=request.getParameter("name")%>
대신에 **\${param.name}** 구문을 사용한다.

| EL(Expression Language)의 개요

EL은 기본적으로 4가지 Scope(page, request, session, application)의 객체에 접근하여 출력을 처리한다.

EL에서는 해당값이 null이거나 공백일 경우에는 아무 내용도 표시하지 않고 에러도 발생하지 않는다.

EL은 JSP에서 기본으로 지원하고, JSTL은 따로 설치해야 한다.

I EL(Expression Language)과 스크립팅 태그 비교

EL (Expression Language)	스크립팅 태그
<code> \${param.name}</code>	<code><%=request.getParameter("name")%></code>
<code> \${greet}</code>	<code><% String value = (String)request.getAttribute("greet"); out.println(value); %></code>
<code> \${user}</code>	<code><% UserVO user = (UserVO)request.getAttribute("user"); out.println(user); %></code>

I EL(Expression Language)과 스크립팅 태그 비교

EL (Expression Language)	스크립팅 태그
<code> \${user.name}</code>	<code><% UserVO user = (UserVO)request.getAttribute("user"); out.println(user.getName()); %></code>
<code> \${sessionScope.user.name}</code>	<code><% UserVO user = (UserVO)session.getAttribute("user"); out.println(user.getName()); %></code>

I JSTL(Java Standard Tag Library) 개요

JSTL이란 JSP 표준 라이브러리(JSP Standard Tag Library)

JSTL은 JSP에서 스크립팅을 사용하지 않으면서,
루프를 돌리거나 조건문을 실행할 수 있도록 해줌

JSP에서 자주 사용하는 기능(반복과 조건, 데이터 관리 포맷, XML
조작, 데이터베이스 액세스)을 구현해 놓은 Custom Tag Library 모음

JSTL은 EL(Expression Language)를 사용하여 표현

JSTL은 request, response, pageContext, application과 같은
JSP 내장 객체(Implicit Object)에 쉽게 접근할 수 있음

I JSTL(Java Standard Tag Library) 개요

- JSTL은 아래와 같이 5개의 Library가 있음

라이브러리	기능	URL 식별자	접두어
코어(core)	일반 프로그램 언어에서 제공하는 변수선언, 조건/제어/반복문 등의 기능을 제공한다.	http://java.sun.com/jsp/jstl/core	c
포맷팅 (formatting)	숫자, 날짜, 시간을 포맷팅 하는 기능과 국제화, 다국어 지원 기능을 제공한다.	http://java.sun.com/jsp/jstl/fmt	fmt
함수 (function)	문자열을 처리하는 함수를 제공한다	http://java.sun.com/jsp/jstl/functions	fn

I JSTL(Java Standard Tag Library) 개요

- JSTL은 아래와 같이 5개의 Library가 있음

라이브러리	기능	URL 식별자	접두어
데이터 베이스 (database)	데이터베이스의 데이터를 입력/수정/삭제/조회하는 기능을 제공한다.	http://java.sun.com/jsp/jstl/sql	sql
XML처리 (xml)	XML 문서를 처리할 때 필요한 기능을 제공한다.	http://java.sun.com/jsp/jstl/xml	x

I JSTL(Java Standard Tag Library) 개요

- ◎ JSTL의 사용

```
<!-- 선언부 -->
<%@ taglib prefix="c"    uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt"  uri="http://java.sun.com/jsp/jstl/fmt"%>
<!-- 사용-->
<c:set var="hello" value="Hello" />
${hello}
```

I JSTL(Java Standard Tag Library) Core 태그

- JSTL 태그 라이브러리 중에 가장 많이 사용하는 태그이다.
- <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>로 선언해야 함

라이브러리	태그	설명
변수지원	set	JSP에서 사용될 변수를 설정한다.
	remove	설정할 변수를 제거한다.
흐름제어	if	조건에 따라 내부 코드를 수행한다.
	choose	다중 조건을 처리할 때 사용한다.
	forEach	컬렉션이나 Map의 각 항목을 처리할 때 사용한다.
	forTokens	구분자로 분리된 각각의 토큰을 처리할 때 사용한다.

I JSTL(Java Standard Tag Library) Core 태그

- JSTL 태그 라이브러리 중에 가장 많이 사용하는 태그이다.
- <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>로 선언해야 함

라이브러리	태그	설명
URL 처리	import	URL을 사용하여 다른 자원의 결과를 삽입한다.
	redirect	지정한 경로로 리다이렉트 한다.
	url	URL을 재작성한다.
기타태크	catch	익셉션 처리에 사용된다.
	out	JspWriter에 내용을 알맞게 처리한 후 출력한다.

| JSTL 라이브러리 검색 및 설치

The screenshot shows the Maven Repository search interface. A search bar at the top contains the text "jstl". Below the search bar, the results for "Jstl" are displayed. The first result is for "javax.servlet > jstl" version 1.2, which has 713 usages. The result card includes a logo for the JavaServer Faces Technology Committee, the artifact name "Jstl", and license information "GPL | CDDL".

```
<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

AOP는

2. Hello 컨트롤러 작성

| Controller 와 JSP 구현 절차

1 클라이언트의 요청을 처리할 POJO 형태의 HelloController 클래스를 작성

2 Controller 클래스에 @Controller 어노테이션을 선언

3 요청을 처리할 메서드를 작성하고 @RequestMapping 어노테이션을 선언

4 JSP를 이용한 View 영역의 코드를 작성

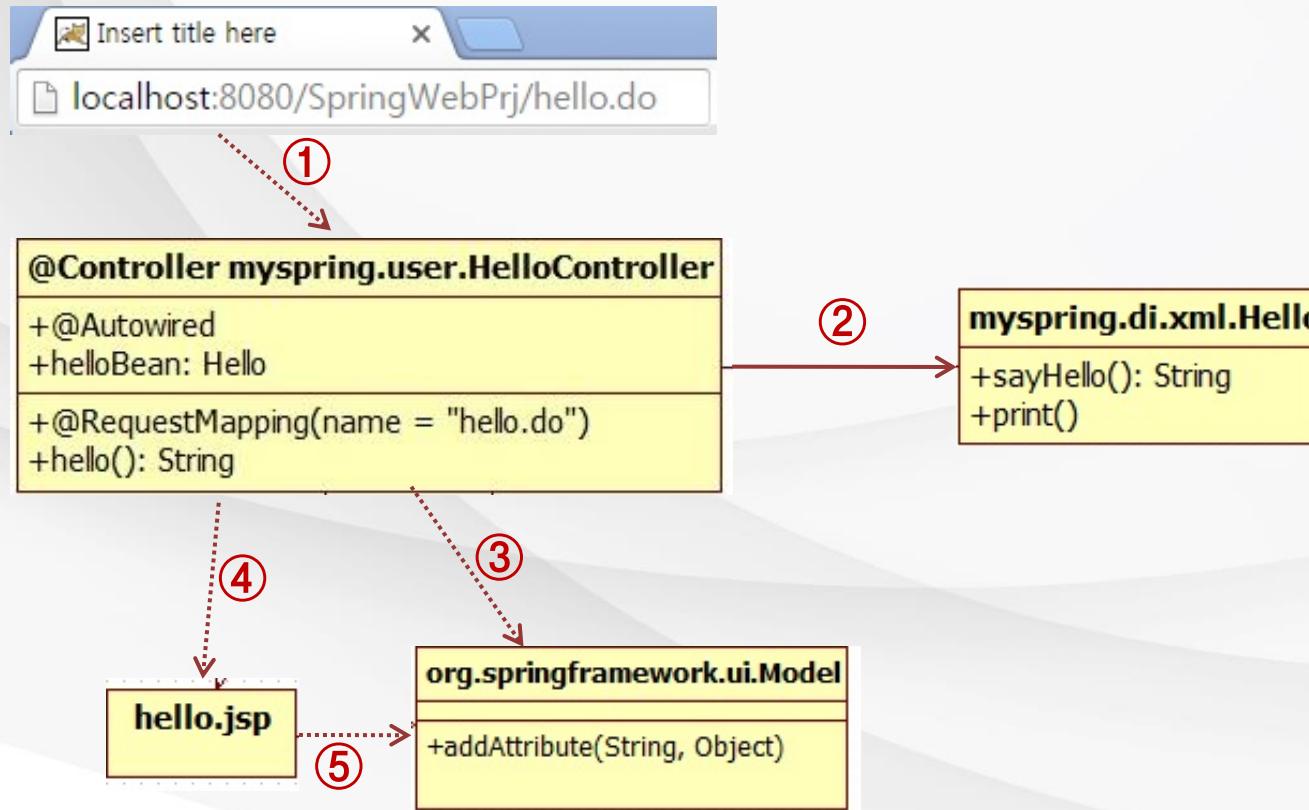
5 Browser 상에서 JSP를 실행

| Controller를 위한 핵심 어노테이션(Annotation)

구성요소	설명
@Controller	Controller 클래스 정의
@RequestMapping	HTTP 요청 URL을 처리할 Controller 메소드 정의

2. Hello 컨트롤러 작성

Controller와 JSP 호출 순서



| View에 데이터를 전달하는 Model 클래스

org.springframework.ui.Model

```
+addAttribute(name: String, value: Object): Model  
+addAttribute(value: Object): Model  
+addAllAttributes(Map<String, ?> attributes): Model
```

- Controller에서 Service를 호출한 결과를 받아서 view에게 전달하기 위해, 전달받은 결과를 Model 객체에 저장
- Model addAttribute(string name, Object value)
: value 객체를 name 이름으로 저장하고,
view 코드에서는 name으로 지정한 이름을 통해서 value를 사용

Controller 와 JSP 구현

```
*HelloController.java
```

```
package my.spring.controller;
@Controller
public class HelloController {
    @Autowired
    private Hello helloBean;

    @RequestMapping("/hello.do")
    public String hello(Model model) {
        String msg = helloBean.sayHello();
        model.addAttribute("greet", msg);
        return "hello.jsp";
    }
}
```

```
*Hello.java
```

```
package myspring.di.xml;
public class Hello {
    String name;
    Printer printer;

    public Hello() {}

    public String sayHello() {
        return "Hello " + name;
    }
}
```

Controller와 JSP 구현

```
hello.jsp
```

```
<%@ page language="java" co
  pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W
<html>
<head>
<meta http-equiv="Content-Ty
<title>Insert title here</title>
</head>
<body>
  ${greet}
</body>
</html>
```

```
beans.xml
```

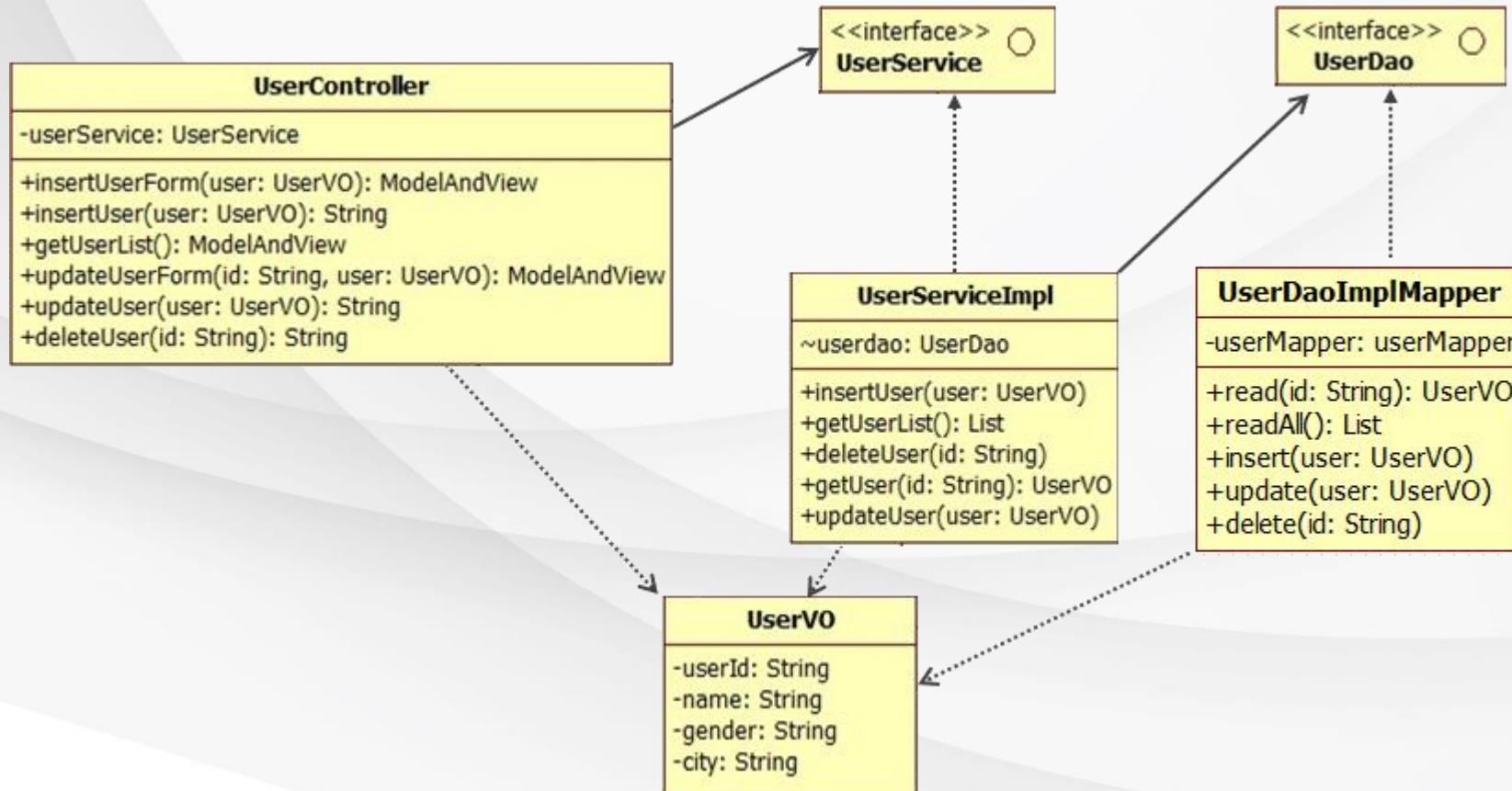
```
<context:component-scan
  base-package="myspring.user,myspring.aop.annot" />
```



A photograph showing a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred, colorful circular lights in shades of yellow, orange, and blue, suggesting an urban night setting.

3. 사용자 목록 조회 컨트롤러 작성

사용자 관리 프로젝트 클래스 설계



| 사용자 목록 조회 : Controller와 JSP 구현 절차

1 클라이언트의 요청을 처리할 POJO 형태의

UserController 클래스를 작성

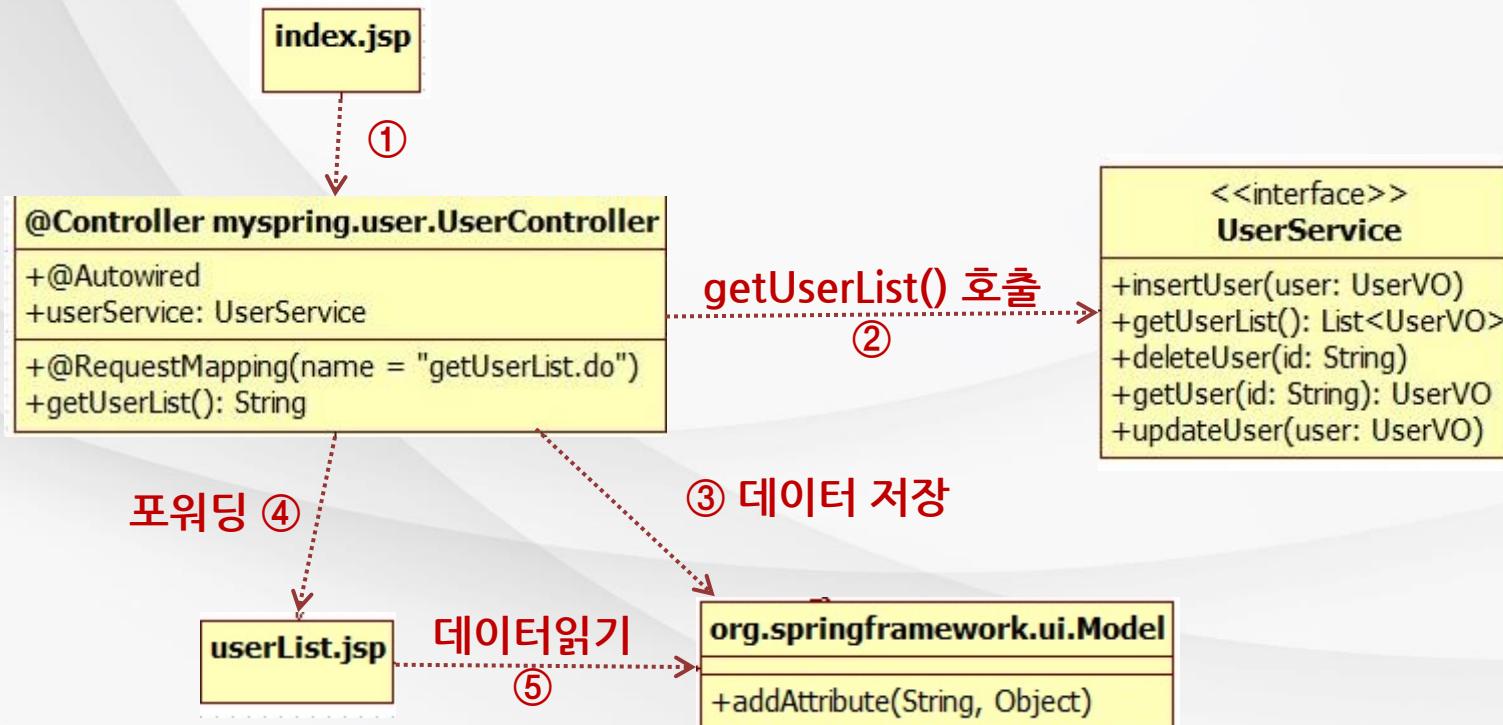
2 Controller 클래스에 @Controller 어노테이션을 선언

3 사용자 목록을 조회하는 getUserList() 메서드를 작성하고
@RequestMapping 어노테이션을 선언

4 userList.jsp 페이지에 View 영역의 코드를 작성

5 Browser 상에서 JSP를 실행

| 사용자 목록 조회 : Controller와 JSP 호출 순서



3. 사용자 목록 조회 컨트롤러 작성

| 사용자 목록조회 : Controller와 JSP 구현

1. index.jsp

```
*index.jsp
1 <%response.sendRedirect("getUserList.do");%>
```

3. beans.xml

```
beans.xml
<context:component-scan
    base-package= "myspring.user,myspring.aop.annot" />
```

2. UserController.java

```
*UserController.java
package myspring.user.controller;

@Controller
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping("/getUserList.do")
    public String getUserList(Model model) {
        List<UserVO> userList = userService.getUserList();
        model.addAttribute("userList", userList);
        return "userList.jsp";
    }
}
```

| 사용자 목록조회 : Controller와 JSP 구현

4. userList.jsp

```
userList.jsp ✎
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<tbody>
    <c:forEach var="user" items="${userList}">
        <tr>
            <td>${user.userId}</td>
            <td>${user.name}</td>
            <td>${user.gender}</td>
            <td>${user.city}</td>
            <td>수정</td>
            <td>삭제</td>
        </tr>
    </c:forEach>
</tbody>
```

3. 사용자 목록 조회 컨트롤러 작성

| 사용자 목록조회 : 결과 화면

The screenshot shows a web browser window with the following details:

- Address bar: localhost:8080/SpringWebPrj/getUserList.do
- Toolbar icons: back, forward, refresh, star, mobile, email, print, globe, and menu.
- Page title: 사용자 목록 (User List)
- Table data:

아이디	이름	성별	거주지		
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제
- Text link at the bottom left: 사용자 등록 (User Registration)

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring MVC 어플리케이션 작성(1)]에 대해서 살펴보았습니다.

EL과 JSTL

- EL(Expression Language)의 개요
- JSTL(Java Standard Tag Library)의 개요 및 설치

Hello 컨트롤러 작성

- 기존의 Hello Bean을 호출하는 HelloController 클래스 작성
- @Controller , @RequestMapping 어노테이션의 사용

사용자 목록 조회 컨트롤러 작성

- UserService Bean을 호출하여 사용자 목록을 조회하는 UserController 클래스 작성

Spring Framework

22. Spring MVC 어플리케이션 작성(2)

CONTENTS

1

특정 사용자 조회 기능 구현

2

사용자 등록화면 기능 구현

3

사용자 등록 기능 구현

학습 목표

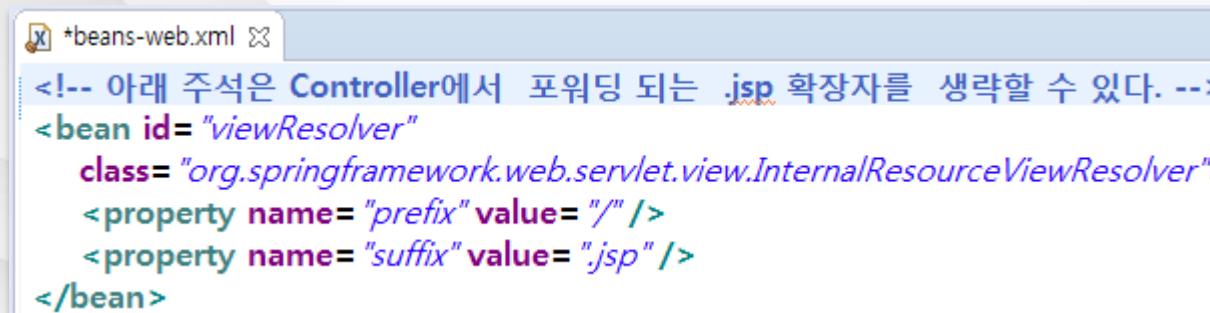
- 특정 사용자 조회 기능 구현에 대하여 이해할 수 있습니다.
- 사용자 등록화면 기능 구현에 대하여 이해할 수 있습니다.
- 사용자 등록 기능 구현에 대해 이해할 수 있습니다.



1. 특정 사용자 조회 기능 구현

| ViewResolver 설정

- ViewResolver는 Controller의 실행 결과를 어떤 view에서 보여줄 것인지 결정하는 기능을 제공
- InternalResourceViewResolver는 JSP를 사용하여 view를 생성



```
*beans-web.xml
<!-- 아래 주석은 Controller에서 포워딩 되는 .jsp 확장자를 생략할 수 있다. -->
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/" />
    <property name="suffix" value=".jsp" />
</bean>
```

- **prefix** - Controller가 리턴한 view 이름 앞에 붙을 접두어
- **suffix** - Controller가 리턴한 view 이름 뒤에 붙을 확장자
- **Controller가 처리 결과를 보여줄 view의 이름으로**
"hello"를 리턴 했다면 InternalResourceViewResolver에 의해 사용되는 view는 "/hello.jsp"가 됨

I 특정 사용자 조회 : Controller와 JSP 구현 절차

- 1 사용자 목록을 조회하는 `getUser(String id)` 메서드를 작성하고 `@RequestMapping`과 `@RequestParam` 어노테이션을 선언
- 2 `userList.jsp` 페이지를 수정
- 3 `userInfo.jsp` 페이지에 View 영역의 코드를 작성
- 4 Browser 상에서 JSP를 실행

Controller 를 위한 핵심 어노테이션(Annotation)

구성요소	설명
@RequestParam	HTTP 요청에 포함된 파라미터 참조 시 사용

```
userList.jsp ✘
<c:forEach var="user" items="${userList}">
    <tr>
        <td>
            <a href="getUser.do?id=${user.userId}">
                ${user.userId}
            </a>
        </td>
        <td>${user.name}</td>
        <td>${user.gender}</td>
        <td>${user.city}</td>
    </tr>
</c:forEach>
```

```
UserController.java ✘
package myspring.user.controller;

@Controller
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping("/getUser.do")
    public ModelAndView getUser(@RequestParam String id) {
        UserVO user = userService.getUser(id);
        return new ModelAndView("userInfo","user",user);
    }
}
```

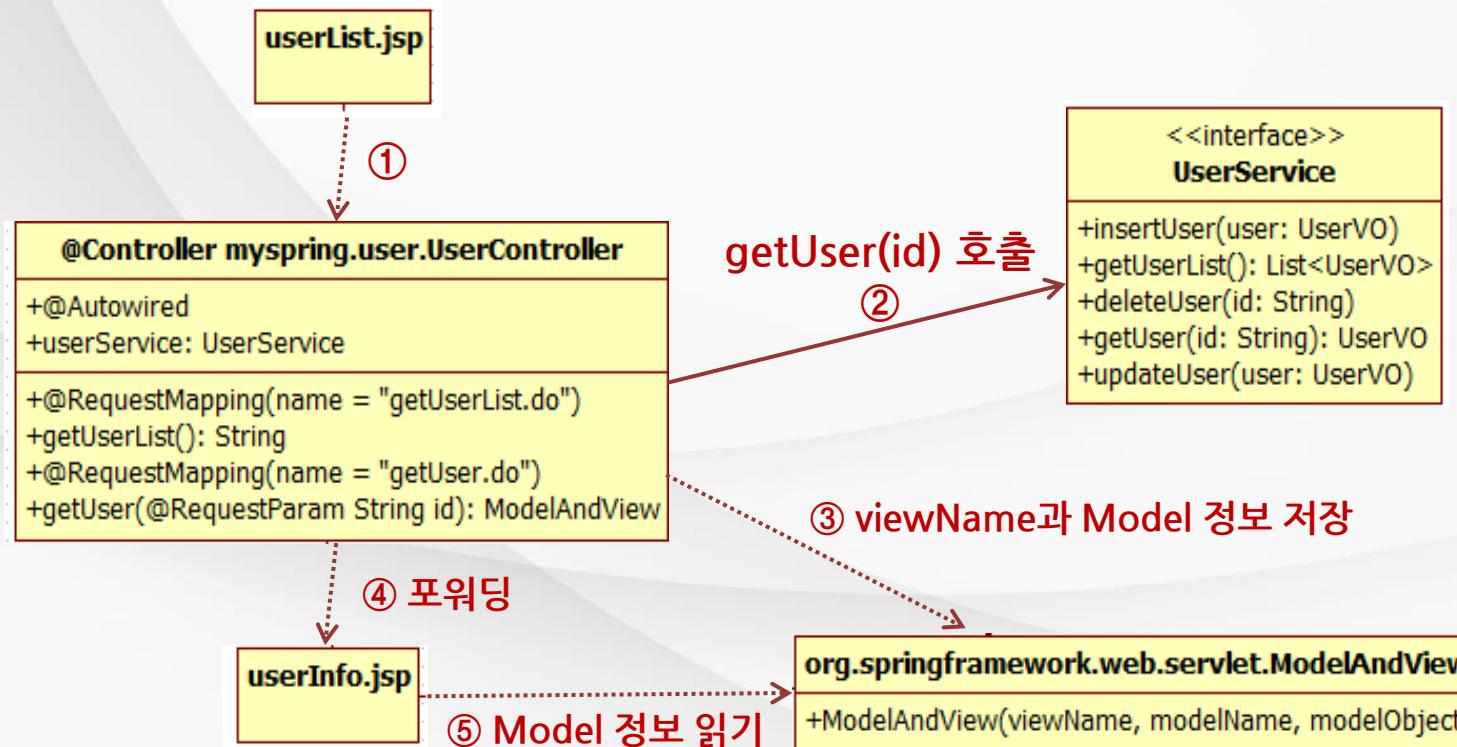
| View에 데이터와 화면정보를 전달하는 ModelAndView 클래스

org.springframework.web.servlet.ModelAndView

```
+ ModelAndView(viewName: String, modelName: String, modelObject: Object)
+ ModelAndView(viewName: String)
+ addObject(attrName: String, attrValue: Object)
+ getModel(): Map<String, Object>
+ getView(): View
+ setView(view: View)
+ setViewName(viewName: String)
```

- Controller에서 Service를 호출한 결과를 받아서 View에게 전달하기 위해, 전달받은 데이터와 화면정보를 ModelAndView 객체에 저장
- ModelAndView 클래스의 생성자나, setViewName() 메서드를 이용해서 View 이름을 지정할 수 있음
- addObject(String name, Object value) 메서드를 이용해 View에 전달할 데이터를 저장할 수 있음

| 특정 사용자 조회 : Controller 와 JSP 호출 순서



I 특정 사용자 조회 : Controller와 JSP 구현

1. userList.jsp

```
userList.jsp
<c:forEach var="user" items="${userList}">
    <tr>
        <td>
            <a href="getUser.do?id=${user.userId}">
                ${user.userId}
            </a>
        </td>
        <td>${user.name}</td>
        <td>${user.gender}</td>
        <td>${user.city}</td>
    </tr>
</c:forEach>
```

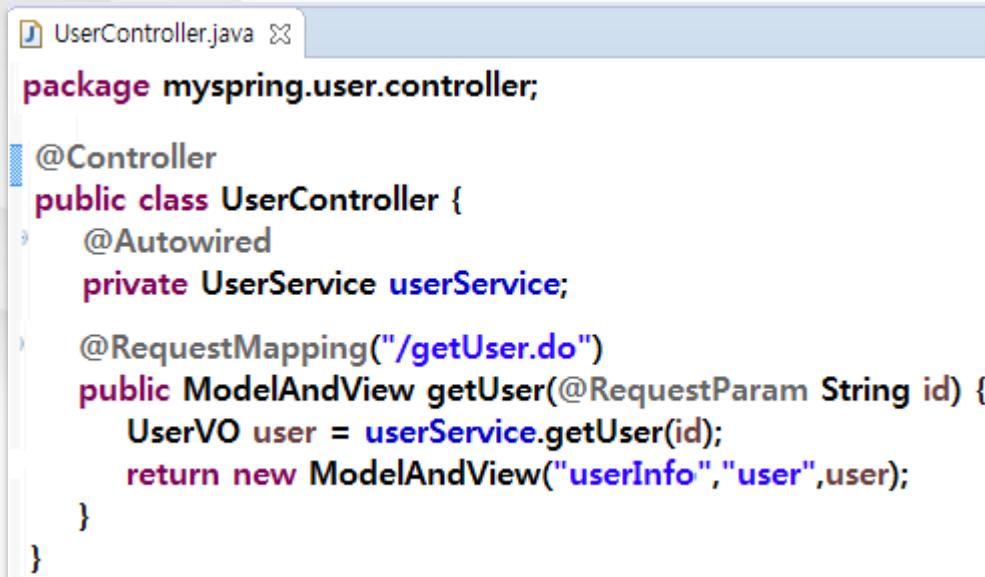
사용자 목록

아이디	이름	성별	거주지		
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제

사용자 등록

| 특정 사용자 조회 : Controller와 JSP 구현

2. UserController.java



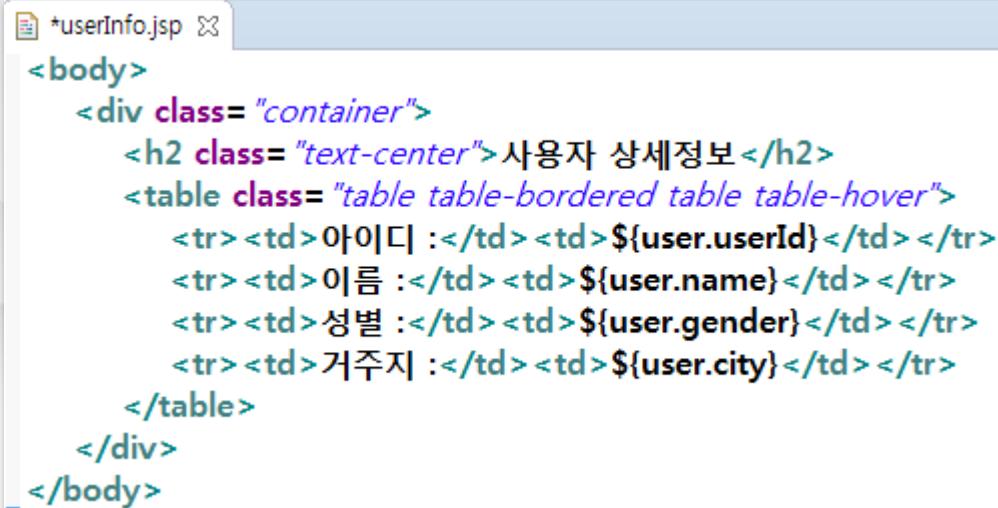
The screenshot shows a Java code editor window with the title bar "UserController.java". The code is annotated with Java annotations: `@Controller`, `@Autowired`, and `@RequestMapping`. The code defines a class `UserController` with a single method `getUser` that returns a `ModelAndView` object.

```
UserController.java
package myspring.user.controller;

@Controller
public class UserController {
    @Autowired
    private UserService userService;
    @RequestMapping("/getUser.do")
    public ModelAndView getUser(@RequestParam String id) {
        UserVO user = userService.getUser(id);
        return new ModelAndView("userInfo","user",user);
    }
}
```

| 특정 사용자 조회 : Controller와 JSP 구현

3. userInfo.jsp



```
*userInfo.jsp ✘
<body>
  <div class= "container">
    <h2 class= "text-center">사용자 상세정보</h2>
    <table class= "table table-bordered table-hover">
      <tr><td>아이디 :</td><td>${user.userId}</td></tr>
      <tr><td>이름 :</td><td>${user.name}</td></tr>
      <tr><td>성별 :</td><td>${user.gender}</td></tr>
      <tr><td>거주지 :</td><td>${user.city}</td></tr>
    </table>
  </div>
</body>
```

| 특정 사용자 조회 : 결과 화면

사용자 목록					
아이디	이름	성별	거주지		
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제

사용자 등록



사용자 상세정보

아이디 :	gildong
이름 :	홍길동2
성별 :	남2
거주지 :	경기2

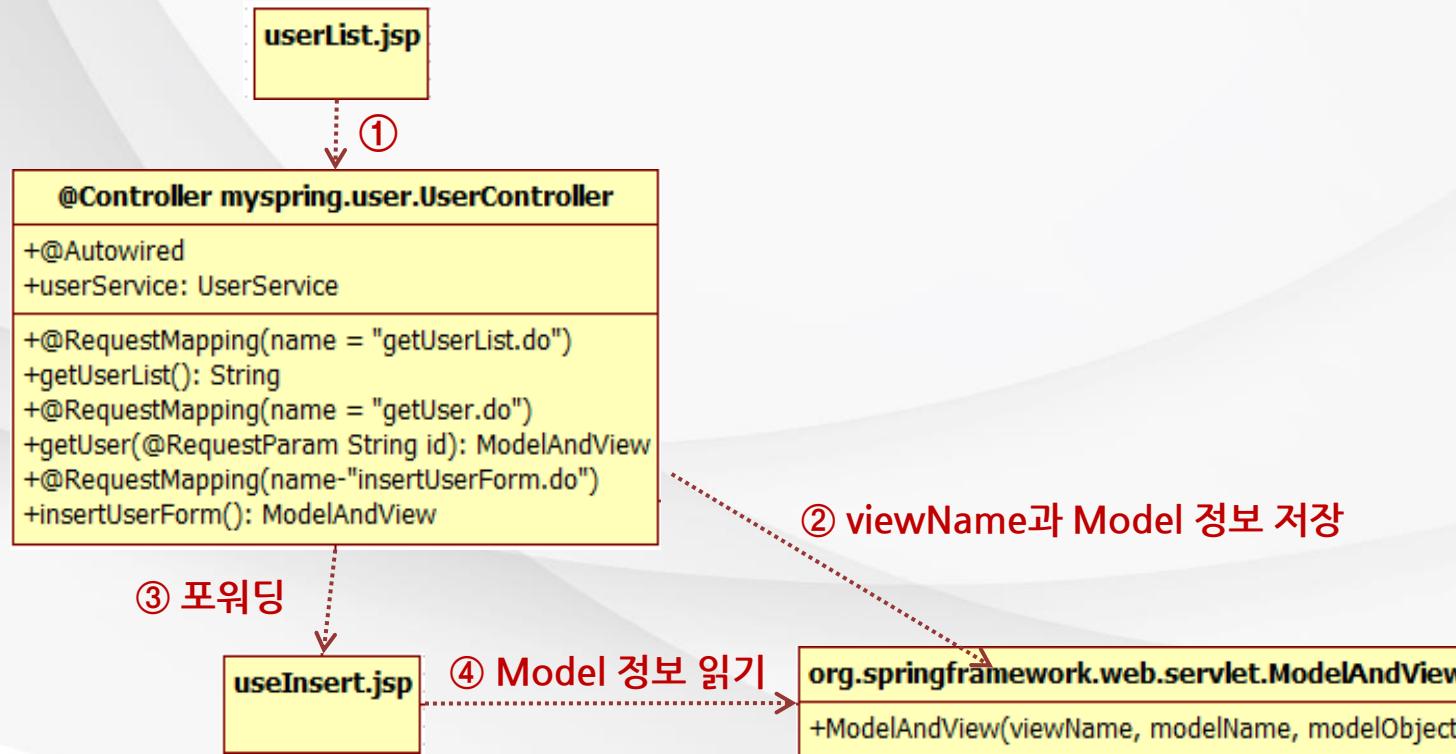
AOP는

2. 사용자 등록화면 기능 구현

| 사용자 정보 등록 화면 : Controller 와 JSP 구현 절차

- 1 사용자 정보를 등록하는 화면을 포워딩 해주는 insertUserForm()
메서드를 작성하고 @RequestMapping 어노테이션을 선언
- 2 userList.jsp 페이지를 수정
- 3 userInsert.jsp 페이지에 View 영역의 코드를 작성
- 4 Browser 상에서 JSP를 실행

| 사용자 정보 등록 화면 : Controller 와 JSP 호출 순서



| 사용자 정보 등록 화면 : Controller와 JSP 구현

1. userList.jsp

```
*userList.jsp ✎
<c:forEach var= "user" items= "${userList}" >
    .....
</c:forEach>
<tr>
    <td colspan= "7">
        <a href= "insertUserForm.do">사용자 등록 </a>
    </td>
</tr>
```

사용자 목록

아이디	이름	성별	거주지		
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제
사용자 등록					

| 사용자 정보 등록 화면 : Controller와 JSP 구현

2. UserController.java

```
UserController.java
@Controller
public class UserController {
    @RequestMapping("/insertUserForm.do")
    public ModelAndView insertUserForm() {
        List<String> genderList = new ArrayList<String>();
        genderList.add("남");
        genderList.add("여");
        List<String> cityList = new ArrayList<String>();
        cityList.add("서울");
        cityList.add("부산");
        cityList.add("대구");
        cityList.add("제주");
        Map<String, List<String>> map = new HashMap<>();
        map.put("genderList", genderList);
        map.put("cityList", cityList);
        return new ModelAndView("userInsert", "map", map);
    }
}
```

성별 :	<input type="radio"/> 남 <input type="radio"/> 여
거주지 :	<input type="button" value="서울 ▾"/> 서울 부산 대구 제주
<input type="button" value="등록"/> [사용자 목록보기]	

| 사용자 정보 등록 화면 : Controller와 JSP 구현

3. userInsert.jsp

The screenshot shows a Java IDE interface with two files open:

- UserController.java**: A Java class containing code to handle user registration. It creates a map of gender and city lists and returns a ModelAndView object.
- userInsert.jsp**: An JSP page template for user registration. It uses JSTL tags to display gender and city selection fields.

```
UserController.java
Map<String, List<String>> map = new HashMap<>();
map.put("genderList", genderList);
map.put("cityList", cityList);
return new ModelAndView("userInsert.jsp", "map", map);
```

```
userInsert.jsp
<div class="container">
    <h2 class="text-center">사용자 정보 등록</h2>
    .....
    <tr>
        <td>성별 :</td>
        <td><c:forEach var="genderName" items= '${map.genderList}'>
            <input type="radio" name="gender" value= '${genderName}'>${genderName}
        </c:forEach></td>
    </tr>
    <tr>
        <td>거주지 :</td>
        <td><select name="city">
            <c:forEach var="cityName" items= '${map.cityList}'>
                <option value= '${cityName}'>${cityName}</option>
            </c:forEach>
            </select></td>
    </tr>
```

2. 사용자 등록화면 기능 구현

| 사용자 정보 등록 화면 : 결과 화면

사용자 목록					
아이디	이름	성별	거주지		
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제
사용자 등록					



사용자 정보 등록

아이디 :	<input type="text"/>
이름 :	<input type="text"/>
성별 :	<input checked="" type="radio"/> 남 <input type="radio"/> 여
거주지 :	<input type="text" value="서울"/> ▼ 서울 부산 대구 제주
<input type="button" value="등록"/>	
사용자 목록보기	



A photograph showing a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred, colorful circular lights in shades of yellow, orange, and blue, suggesting an urban night setting.

3. 사용자 등록 기능 구현

| 사용자 정보 등록 : Controller 와 JSP 구현 절차

1 사용자 정보를 등록하는 insertUser(@ModelAttribute UserVO userVO) 메서드를 작성하고 @RequestMapping과 @ModelAttribute 어노테이션을 선언

: 등록 후에 목록 조회가 redirect 되도록 하여, 등록된 사용자 정보를 확인할 수 있도록 해야 함

2 userInsert.jsp 페이지에 View 영역의 코드를 작성

3 Browser 상에서 JSP를 실행

Controller를 위한 핵심 어노테이션(Annotation)

구성요소	설명
@ModelAttribute	HTTP 요청에 포함된 파라미터를 모델 객체로 바인딩

userInsert.jsp

```

아이디 :<input type="text" name="userId" />
이름 :<input type="text" name="name" />
성별 :<c:forEach var="genderName" items="${map.genderList}">
    <input type="radio" name="gender" value="${genderName}">${genderName}
</c:forEach>
거주지 :<select name="city">
    <c:forEach var="cityName" items="${map.cityList}">
        <option value="${cityName}">${cityName}</option>
    </c:forEach>
</select>

```

UserController.java

```

@RequestMapping("/insertUser.do")
public String insertUser(@ModelAttribute UserVO user) {

```

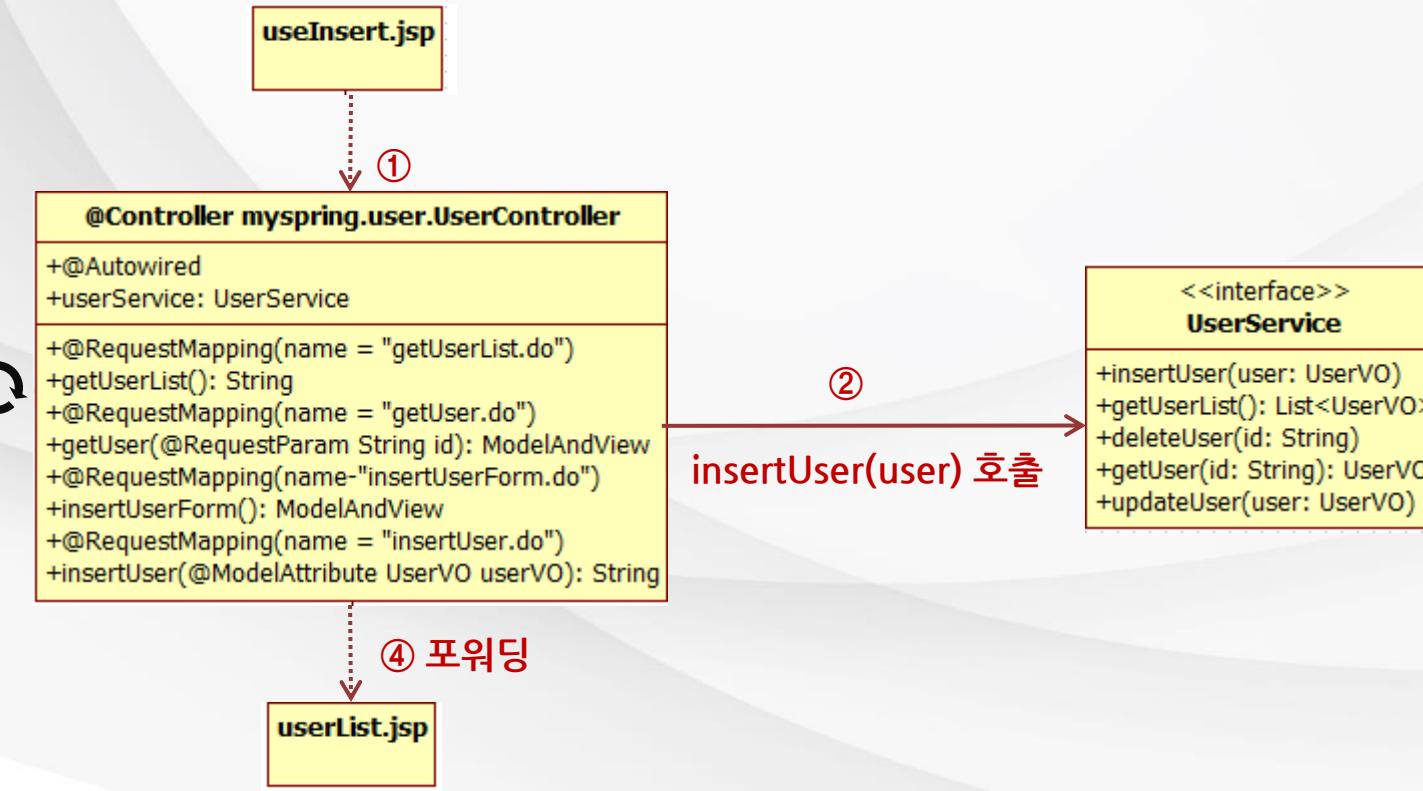
UserVO.java

```

public class UserVO {
    private String userId;
    private String name;
    private String gender;
    private String city;
}

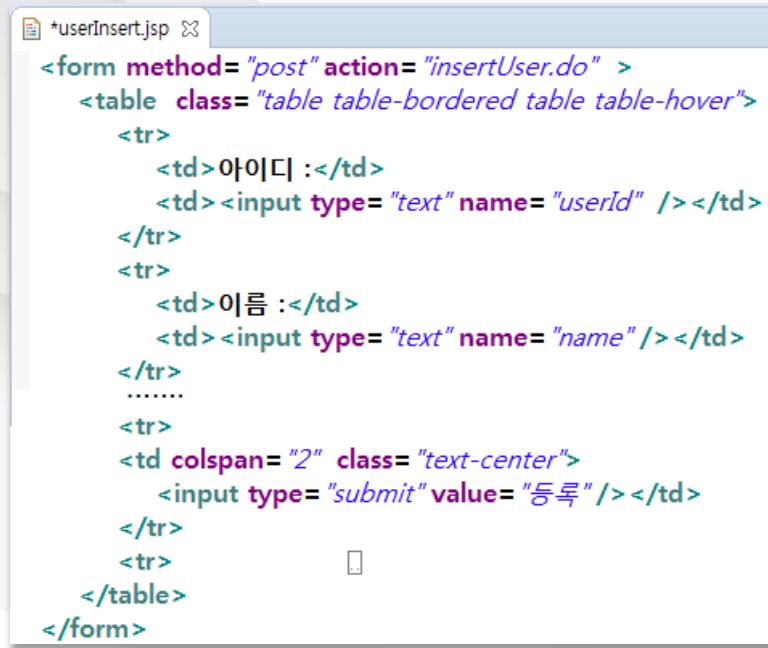
```

| 사용자 정보 등록 : Controller와 JSP 호출 순서



| 사용자 정보 등록 : Controller와 JSP 구현

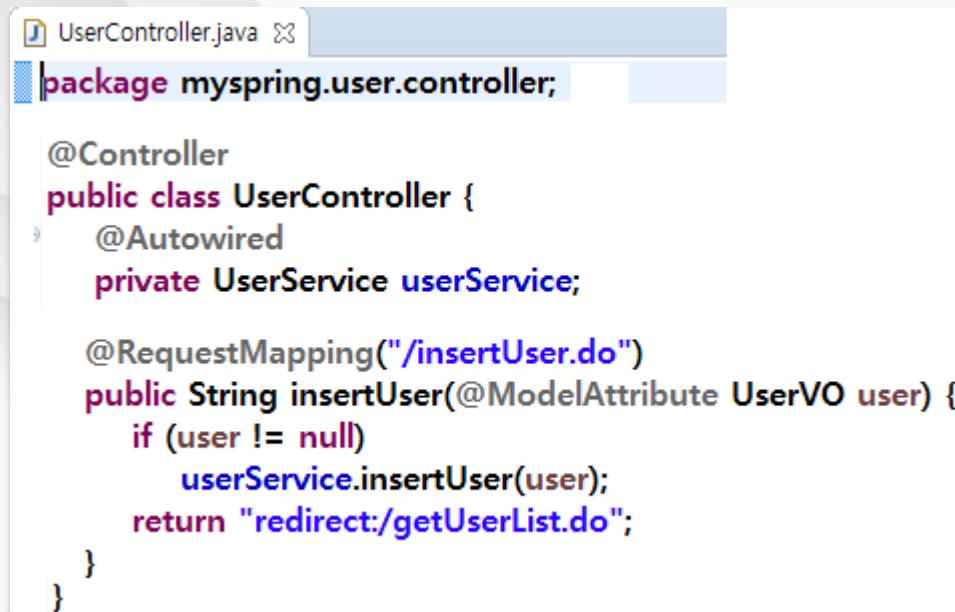
1. userInsert.jsp



```
*userInsert.jsp
<form method="post" action="insertUser.do" >
    <table class="table table-bordered table-hover">
        <tr>
            <td>아이디 :</td>
            <td><input type="text" name="userId" /></td>
        </tr>
        <tr>
            <td>이름 :</td>
            <td><input type="text" name="name" /></td>
        </tr>
        .....
        <tr>
            <td colspan="2" class="text-center">
                <input type="submit" value="등록" /></td>
            </tr>
            <tr>
                ...
            </tr>
    </table>
</form>
```

| 사용자 정보 등록 : Controller와 JSP 구현

2. UserController.java



The screenshot shows a Java code editor with a file named 'UserController.java' open. The code implements a controller for user registration. It includes annotations for the controller class, autowiring the service, and mapping the insertUser method. The method checks if the user is not null before calling the service's insertUser method and then returns a redirect to the userList page.

```
UserController.java
package myspring.user.controller;

@Controller
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping("/insertUser.do")
    public String insertUser(@ModelAttribute UserVO user) {
        if (user != null)
            userService.insertUser(user);
        return "redirect:/getUserList.do";
    }
}
```

■ web.xml : CharacterEncodingFilter 클래스 설정

- 요청(request) 데이터를 인코딩 해주는 Filter 클래스 설정

The screenshot shows a code editor window with a tab labeled "web.xml". The content of the file is as follows:

```
<!-- CharacterEncoding Filter -->
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>EUC-KR</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>*.do</url-pattern>
</filter-mapping>
```

| 사용자 정보 등록 : 결과 화면

사용자 정보 등록

아이디 :	vega2k
이름 :	박소율
성별 :	<input type="radio"/> 남 <input checked="" type="radio"/> 여
거주지 :	서울 ▼

등록

사용자 목록보기



사용자 목록

아이디	이름	성별	거주지	수정	삭제
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제
vega2k	박소율	여	서울	수정	삭제

사용자 등록

지금까지 [Spring MVC 어플리케이션 작성(2)]에 대해서 살펴보았습니다.

특정 사용자 조회 기능 구현

- UserService Bean을 호출하여 특정 사용자 정보를 조회하는 기능 구현
- @RequestParam 어노테이션, ViewResolver 설정

사용자 등록화면 기능 구현

- 사용자 등록하는 화면을 처리하는 기능 구현

사용자 등록 기능 구현

- UserService Bean을 호출하여 사용자 등록하는 기능 구현
- @ModelAttribute 어노테이션, CharacterEncodingFilter 설정

Spring Framework

23. Spring MVC 어플리케이션 작성(3)

CONTENTS

1

사용자 수정화면 기능 구현

2

사용자 수정 및 삭제 기능 구현

3

Spring MVC의 예외 처리

학습 목표

- 사용자 수정화면 기능 구현에 대하여 이해할 수 있습니다.
- 사용자 수정 및 삭제 기능 구현에 대하여 이해할 수 있습니다.
- Spring MVC의 예외 처리에 대해 이해할 수 있습니다.

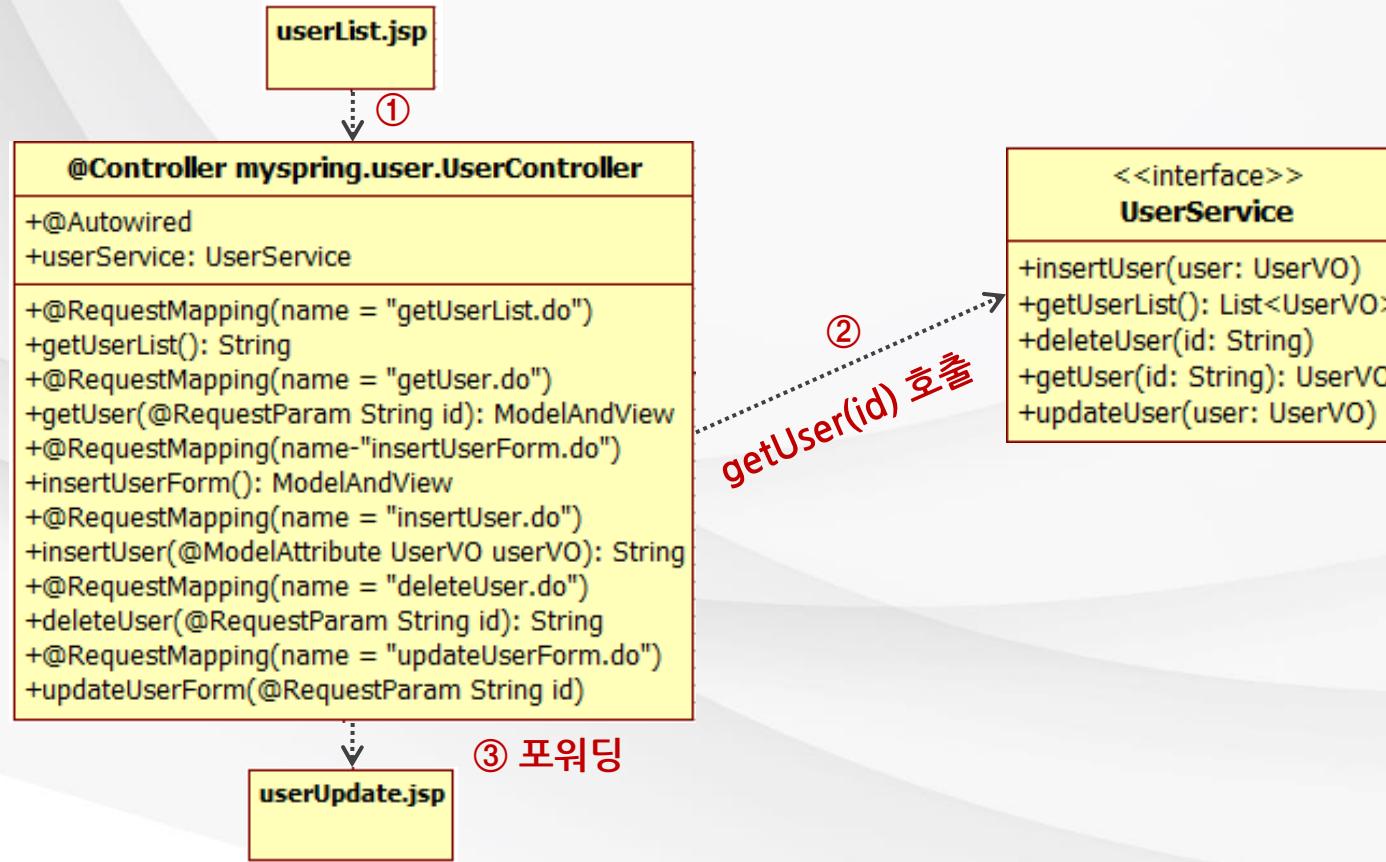


1. 사용자 수정화면 기능 구현

| 사용자 정보 수정화면 : Controller와 JSP 구현 절차

- 1 사용자 정보를 수정하는 화면을 포워딩 해주는 updateUserForm (@RequestParam String id) 메서드를 작성하고 @RequestMapping과 @RequestParam 어노테이션을 선언
- 2 userList.jsp 페이지를 수정
- 3 userUpdate.jsp 페이지에 View 영역의 코드를 작성
- 4 Browser 상에서 JSP를 실행

| 사용자 정보 수정화면 : Controller와 JSP 호출 순서



| 사용자 정보 수정화면 : Controller와 JSP 구현

1.userList.jsp

```
 userList.jsp ✘
<c:forEach var= "user" items= "${userList}">
  <tr>
    <td>
      <a href= "getUser.do?id=${user.userId}">${user.userId}</a>
    </td>
    <td>${user.name}</td>
    <td>${user.gender}</td>
    <td>${user.city}</td>
    <td><a href= "updateUserForm.do?id=${user.userId}">수정 </a></td>
    <td><a href= "deleteUser.do/${user.userId}">삭제 </a></td>
  </tr>
</c:forEach>
```

| 사용자 정보 수정화면 : Controller와 JSP 구현

2. UserController.java

The screenshot shows a Java code editor window with the file name "UserController.java" in the title bar. The code itself is a Java class definition for a controller. It includes annotations for RequestMapping and RequestMethod, and uses various Java collections like ArrayList and HashMap to manage lists of gender and city options, as well as a user object.

```
*UserController.java
@RequestMapping("/updateUserForm.do")
public ModelAndView updateUserForm(@RequestParam String id) {
    UserVO user = userService.getUser(id);
    List<String> genderList = new ArrayList<String>();
    genderList.add("남");
    genderList.add("여");
    List<String> cityList = new ArrayList<String>();
    cityList.add("서울");
    cityList.add("부산");
    cityList.add("대구");
    cityList.add("제주");
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("genderList", genderList);
    map.put("cityList", cityList);
    map.put("user", user);
    return new ModelAndView("userUpdate", "map", map);
}
```

| 사용자 정보 수정화면 : Controller와 JSP 구현

3. userUpdate.jsp

```
<h2 class="text-center">사용자 정보 수정</h2>
<tr>
    <td>거주지:</td>
    <td>
        <select name="city">
            <c:forEach items="${map.cityList}" var="cityName">
                <c:choose>
                    <c:when test="${cityName eq map.user.city}">
                        <option value="${cityName}" selected>${cityName}</option>
                    </c:when>
                    <c:otherwise>
                        <option value="${cityName}">${cityName}</option>
                    </c:otherwise>
                </c:choose>
            </c:forEach>
        </select>
    </td>
</tr>
```

| 사용자 정보 수정화면 : 결과 화면

사용자 목록					
아이디	이름	성별	거주지		
jay	박정우	남	부산	수정	삭제
polar	연아	여	부산	수정	삭제
vega2k	박소율	여	제주	수정	삭제

사용자 등록



사용자 정보 수정

아이디 :	vega2k
이름 :	박소율
성별 :	<input checked="" type="radio"/> 남 <input type="radio"/> 여
거주지 :	제주 ▾
<input type="button" value="수정"/>	

A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred, glowing circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

2. 사용자 수정 및 삭제 기능 구현

| 사용자 정보 수정 : Controller와 JSP 구현 절차

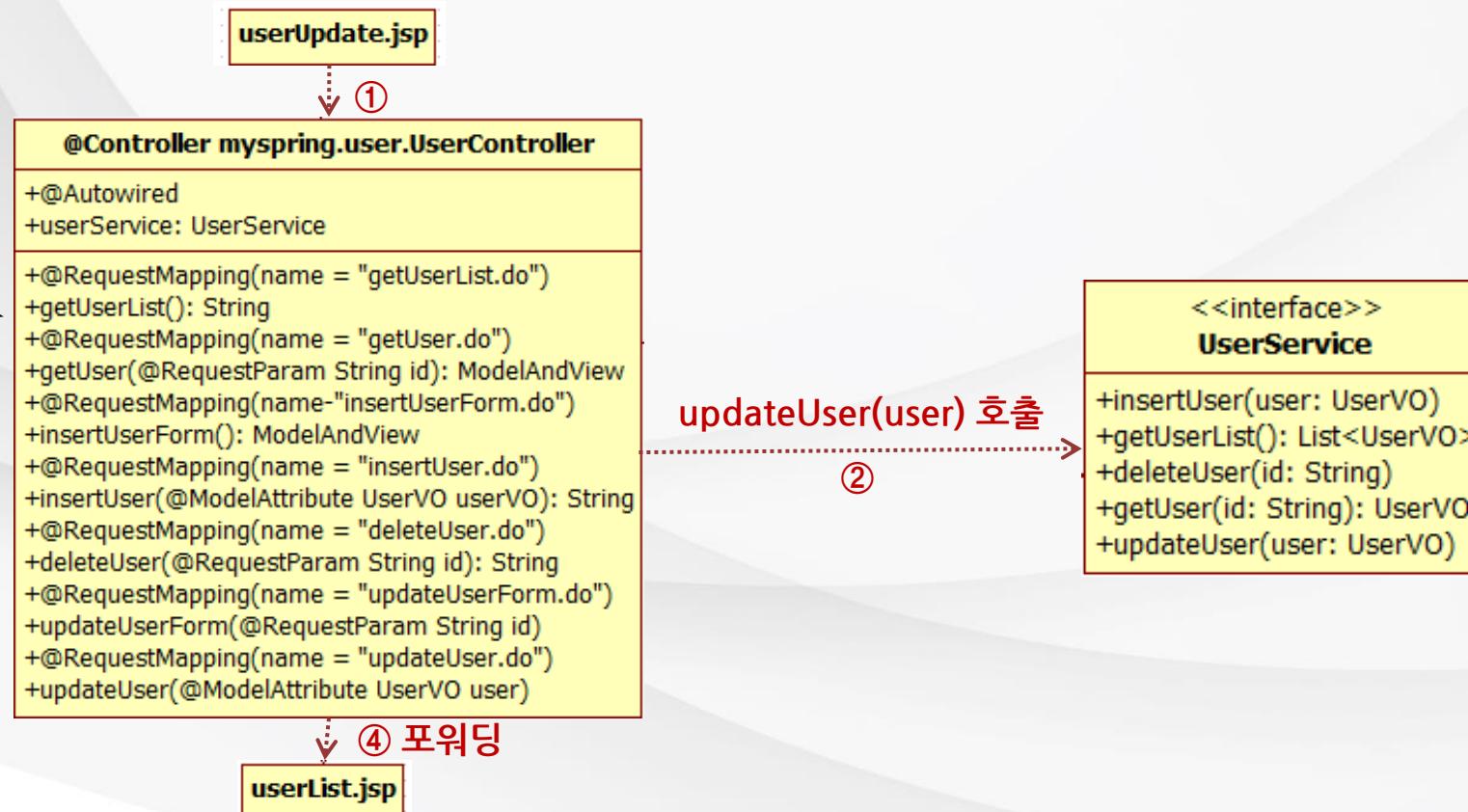
1 사용자 정보를 수정하는 화면을 포워딩 해주는 updateUser
(@ModelAttribute UserVO user) 메서드를 작성하고
@RequestMapping과 @ModelAttribute 어노테이션을 선언

: 수정 후에 목록 조회가 redirect 되도록 하여, 수정된 사용자
정보를 확인할 수 있도록 해야 함

2 userUpdate.jsp 페이지에 View 영역의 코드를 작성

3 Browser 상에서 JSP를 실행

| 사용자 정보 수정 : Controller와 JSP 호출 순서



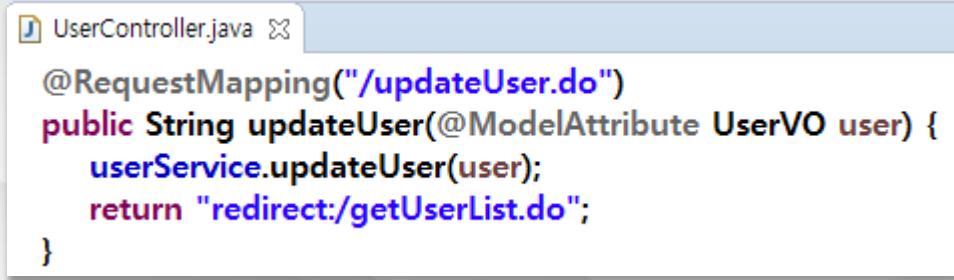
| 사용자 정보 수정 : Controller와 JSP 구현

1. userUpdate.jsp

```
userUpdate.jsp ✘
<h2 class= "text-center">사용자 정보 수정</h2>
<form method= "post" action= "updateUser.do">
  <input type= "hidden" name= "userId" value= "${map.user.userId}" />
  <table class= "table table-bordered table table-hover">
    <tr>
      <td>아이디 :</td>
      <td>${map.user.userId}</td>
    </tr>
    <tr>
      <td>이름 :</td>
      <td><input type= "text" name= "name" value= "${map.user.name}" />
      </td>
    </tr>
    .....
    <tr>
      <td colspan= "2" class= "text-center"><input type= "submit" value= "수정" /></td>
    </tr>
  </table>
</form>
```

| 사용자 정보 수정 : Controller와 JSP 구현

2. UserController.java



```
UserController.java
@RequestMapping("/updateUser.do")
public String updateUser(@ModelAttribute UserVO user) {
    userService.updateUser(user);
    return "redirect:/getUserList.do";
}
```

2. 사용자 수정 및 삭제 기능 구현

| 사용자 정보 수정 : 결과 화면

사용자 목록

아이디	이름	성별	거주지		
jay	박정우	남	부산	수정	삭제
polar	연아	여	부산		
vega2k	박소율	여	제주		

사용자 정보 수정

아이디 :	vega2k
이름 :	박소율New
성별 :	<input checked="" type="radio"/> 남 <input type="radio"/> 여
거주지 :	부산 ▾
수정	

사용자 목록

아이디	이름	성별	거주지		
jay	박정우	남	부산		
polar	연아	여	부산		
vega2k	박소율New	남	부산		

| 사용자 정보 삭제 : Controller와 JSP 구현 절차

1 사용자 정보를 삭제하는 deleteUser(@PathVariable String id)
메서드를 작성하고 @RequestMapping과 @PathVariable
어노테이션을 선언

: 삭제 후에 목록 조회가 redirect 되도록 하여,
삭제된 사용자를 확인할 수 있도록 해야 함

2 userList.jsp 페이지를 수정

3 Browser 상에서 JSP를 실행

| 사용자 정보 삭제 : Controller를 위한 핵심 어노테이션(Annotation)

구성요소	설명
@PathVariable	파라미터를 URL 형식으로 받을 수 있도록 해줌

```
userList.jsp
<c:forEach var="user" items="${userList}">
<tr>
<td>
    <a href="getUser.do?id=${user.userId}">${user.userId}</a>
</td>
<td>${user.name}</td>
<td>${user.gender}</td>
<td>${user.city}</td>
<td>
    <a href="updateUserForm.do?id=${user.userId}">수정</a>
</td>
<td><a href="deleteUser.do?id=${user.userId}">삭제</a></td>
</tr>
</c:forEach>
```

```
UserController.java
package myspring.user.controller;

@Controller
public class UserController {
    @Autowired
    private UserService userService;

    @RequestMapping(value="/deleteUser.do/{id}")
    public String deleteUser(@PathVariable String id) {
        userService.deleteUser(id);
        return "redirect:/getUserList.do";
    }
}
```

| @PathVariable 사용을 위한 DispatcherServlet의 url-pattern 변경

기
존

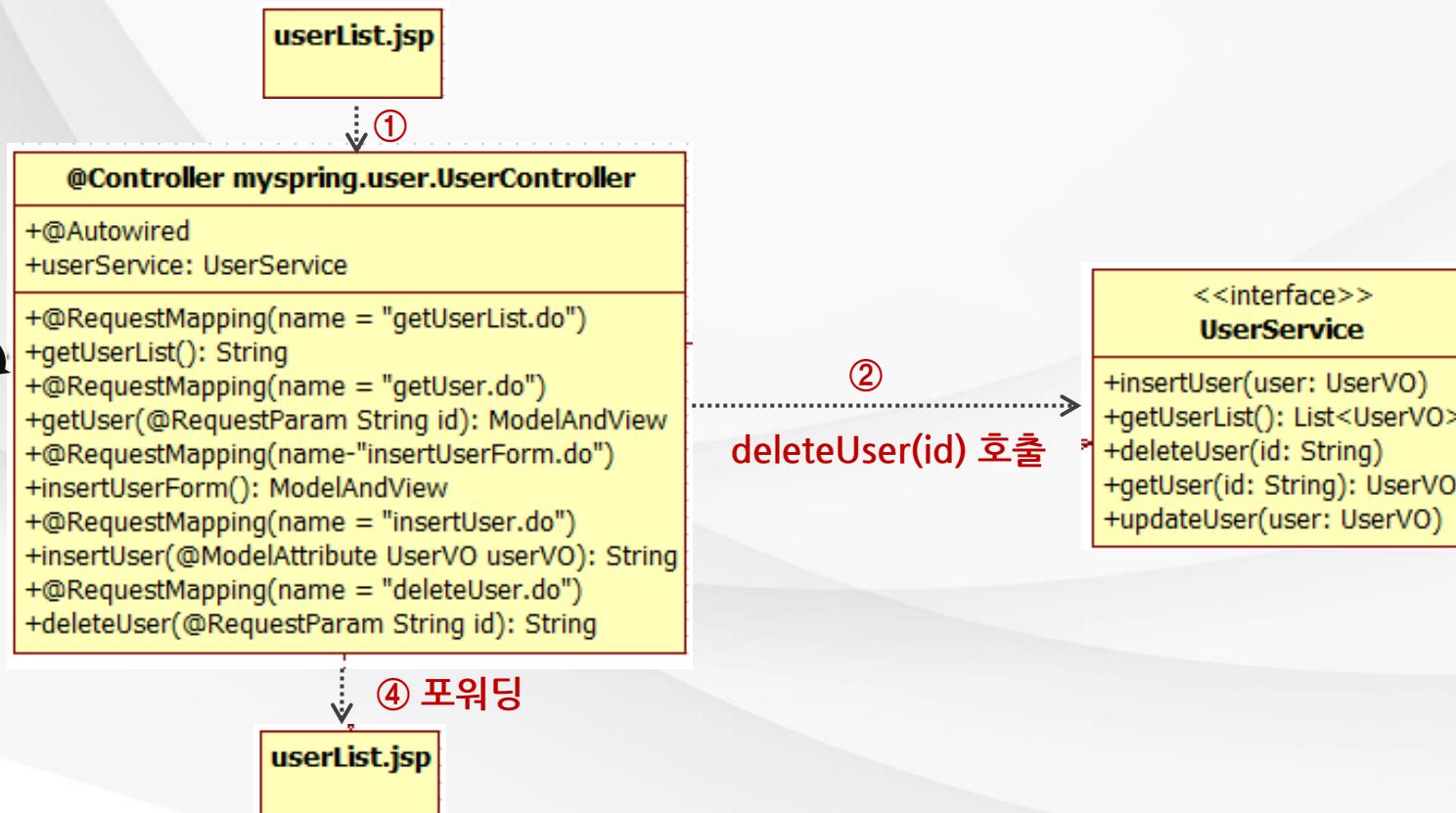
```
<servlet-mapping>
    <servlet-name>springDispatcherServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

변
경

```
<servlet-mapping>
    <servlet-name>springDispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```



| 사용자 정보 삭제 : Controller와 JSP 호출 순서



| 사용자 정보 삭제 : Controller와 JSP 구현

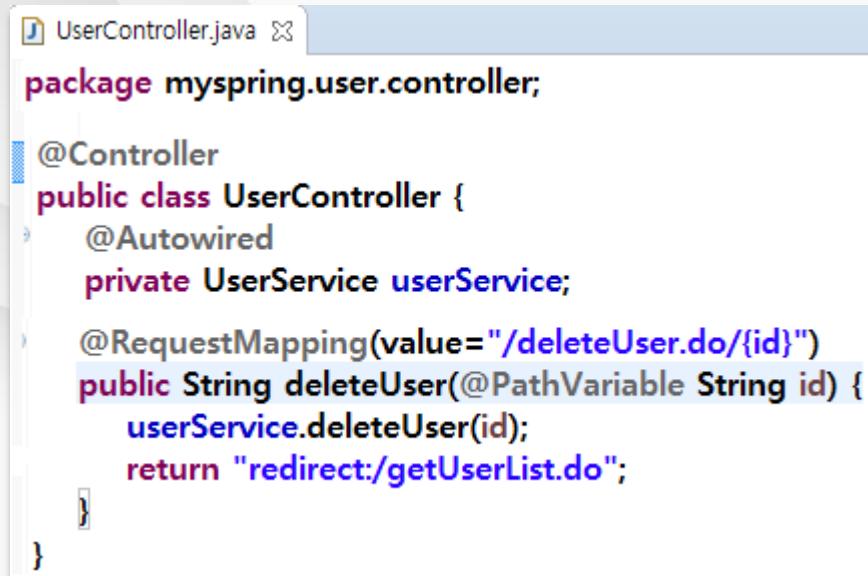
1. userList.jsp

```
userList.jsp ✎

<c:forEach var= "user" items= "${userList}" >
<tr>
  <td>
    <a href= "getUser.do?id=${user.userId}" >${user.userId}</a>
  </td>
  <td>${user.name}</td>
  <td>${user.gender}</td>
  <td>${user.city}</td>
  <td>
    <a href= "updateUserForm.do?id=${user.userId}" >수정</a>
  </td>
  <td><a href= "deleteUser.do?id=${user.userId}" >삭제</a></td>
  </tr>
</c:forEach>
```

| 사용자 정보 삭제 : Controller와 JSP 구현

2. UserController.java



```
UserController.java ✘
package myspring.user.controller;

@Controller
public class UserController {
    @Autowired
    private UserService userService;
    @RequestMapping(value="/deleteUser.do/{id}")
    public String deleteUser(@PathVariable String id) {
        userService.deleteUser(id);
        return "redirect:/getUserList.do";
    }
}
```

| 사용자 정보 삭제 : 결과 화면

사용자 목록

아이디	이름	성별	거주지	수정	삭제
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제
vega2k	박소율	여	서울	수정	<u>삭제</u>



사용자 목록

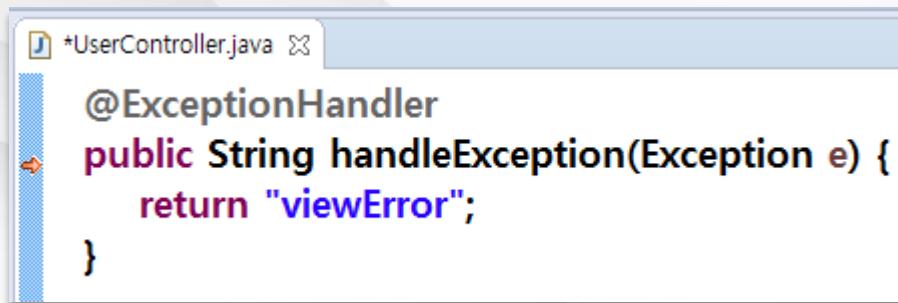
아이디	이름	성별	거주지	수정	삭제
gildong	홍길동2	남2	경기2	수정	삭제
polar	연아2	여2	경기2	수정	삭제

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

3. Spring MVC의 예외 처리

| @ExceptionHandler 어노테이션(Annotation)의 사용

- ◉ 컨트롤러의 메서드에 @ExceptionHandler 어노테이션을 설정하여 컨트롤러의 메서드에서 예외가 발생했을 때 예외 처리를 할 수 있음
- ◉ 예외가 발생했을 때, 예외 Type과 Message를 보여주는 JSP 페이지(viewError.jsp)를 작성해야 함



The screenshot shows a Java code editor window with a file named *UserController.java. The code contains the following:

```
@ExceptionHandler  
public String handleException(Exception e) {  
    return "viewError";  
}
```

Error Page 작성

```
viewError.jsp
```

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ page isErrorPage = "true" %>
<html>
<head><title>에러 발생</title></head>
<body>
요청 처리 과정에서 에러가 발생하였습니다.<br>
빠른 시간 내에 문제를 해결하도록 하겠습니다.
<p>
에러 타입: <%= exception.getClass().getName() %> <br>
에러 메시지: <b><%= exception.getMessage() %></b>
<hr>
<a href= "${pageContext.request.contextPath}">Home</a>
</body>
</html>
```

Exception 발생 시 Error Page가 보여짐

The screenshot shows a web browser window with the title "에러 발생". The address bar displays "localhost:8080/SpringWebPrj/getUser.do?id=v". The main content area of the browser shows the following error message and stack trace:

요청 처리 과정에서 에러가 발생하였습니다.
빠른 시간 내에 문제를 해결하도록 하겠습니다.

에러 타입: org.springframework.jdbc.BadSqlGrammarException
에러 메시지: ### Error querying database. Cause:
java.sql.SQLSyntaxErrorException: ORA-00904: "USERID1": 부적합한 식별자 ###
The error may exist in class path resource [config/User.xml] ### The error may
involve myspring.user.dao.UserMapper.selectUserById-Inline ### The error
occurred while setting parameters ### SQL: select * from users where userid1=?
Cause: java.sql.SQLSyntaxErrorException: ORA-00904: "USERID1": 부적합한
식별자 ; bad SQL grammar []; nested exception is
java.sql.SQLSyntaxErrorException: ORA-00904: "USERID1": 부적합한 식별자

[Home](#)

지금까지 [Spring MVC 어플리케이션 작성(3)]에 대해서 살펴보았습니다.

사용자 수정화면 기능 구현

- UserService Bean을 호출하여 사용자 정보를 수정하기 전에 조회하는 기능 구현

사용자 수정 및 삭제 기능 구현

- UserService Bean을 호출하여 사용자 정보를 수정 및 삭제하는 기능 구현
- @ModelAttribute, @PathVariable의 어노테이션 사용

Spring MVC 의 예외 처리

- @ExceptionHandler 어노테이션의 사용
- Error Page 작성

Spring Framework

24. Spring RESTful 웹서비스 개요 및 환경설정

CONTENTS

- 1 RESTful 웹서비스 개요
- 2 JSON과 XML
- 3 Spring MVC 기반 RESTful 웹서비스 환경설정

학습 목표

- RESTful 웹서비스 개요에 대하여 이해할 수 있습니다.
- JSON과 XML에 대하여 이해할 수 있습니다.
- Spring MVC 기반 RESTful 웹서비스 환경설정에 대해 이해할 수 있습니다.

AOP는

1. RESTful 웹서비스 개요

| Open API(Application Programming Interface) 이란?

Open API는 말 그대로 개방형 API이다.
API가 응용 프로그램을 개발할 때 사용하는 인터페이스라는
의미이므로, Open API는 **프로그래밍에서 사용할 수 있는
개방되어 있는 상태의 인터페이스**를 말한다.

- Daum, Naver 등의 포털 사이트나 통계청, 기상청 등과 같은
관공서에서도 가지고 있는 데이터를 외부 응용 프로그램에서
사용할 수 있도록 Open API를 제공하고 있다.
- Open API와 함께 자주 거론되는 기술이 REST이며, **대부분 Open
API는 REST 방식으로 지원되고 있다.**

■ REST(Representational Safe Transfer) 란?

HTTP URI + HTTP Method

HTTP URI를 통해 제어할 자원(Resource)을 명시하고,
HTTP Method (GET, POST, PUT, DELETE)를 통해
해당 자원(Resource)을 제어하는 명령을 내리는 방식의 아키텍쳐

- HTTP 프로토콜에 정의된 4개의 메서드들이 자원(Resource)에 대한
CRUD Operation을 정의

Http Method	CRUD
POST	Create(Insert)
GET	Read(Select)
PUT	Update or Create
DELETE	Delete

I REST(Representational Safe Transfer)ful API란?

RESTful API는 **HTTP와 URI 기반으로 자원에 접근할 수 있도록 제공하는 애플리케이션 개발 인터페이스** (REST의 원리를 따르는 시스템은 RESTful이란 용어로 지칭된다)

❖ 기존의 웹 접근 방식과 RESTful API 방식과의 차이점

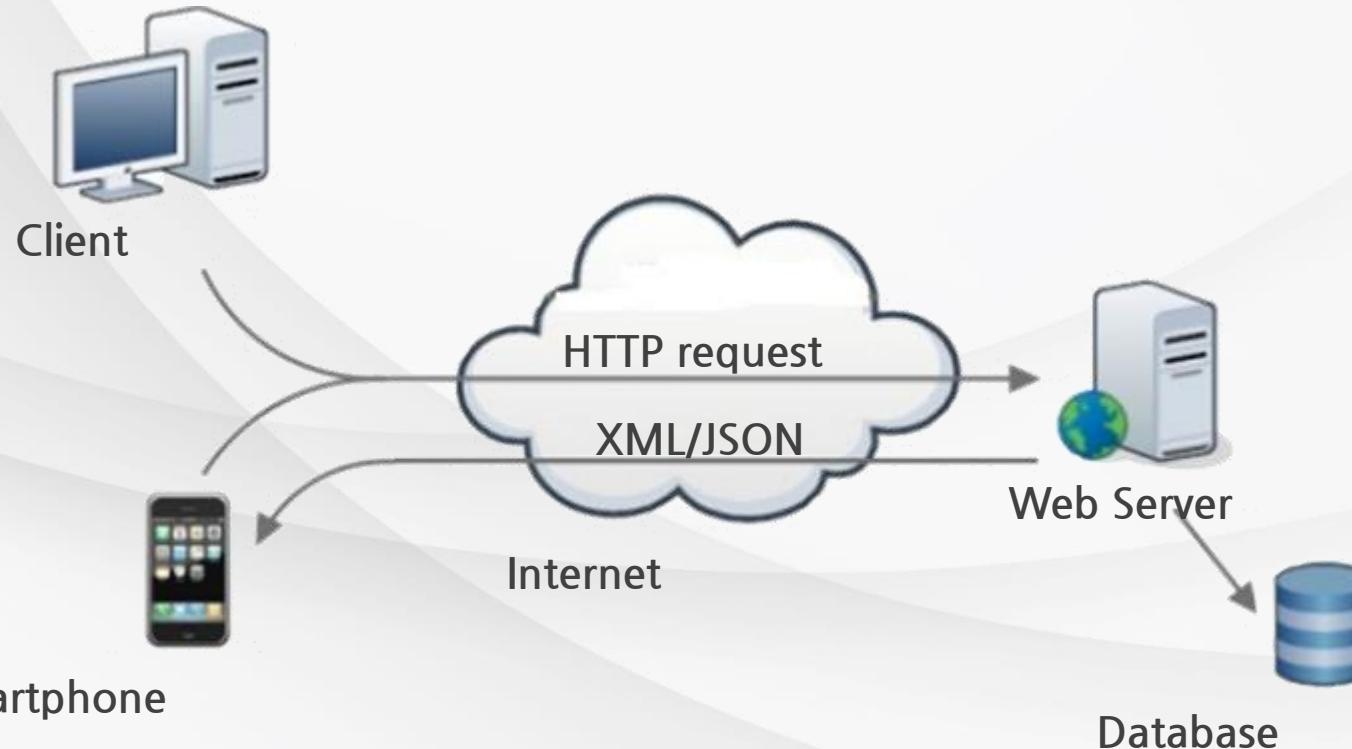
	기존 게시판	Restful API를 지원하는 게시판
글읽기	GET /list.do?no=510&name=java	GET /bbs/java/510
글등록	POST /insert.do	POST /bbs/java/510
글삭제	GET /delete.do?no=510&name=java	DELETE /bbs/java/510
글수정	POST /update.do	PUT /bbs/java/510

- 기존의 게시판은 GET과 POST만으로 자원에 대한 CRUD를 처리하며, URI는 액션을 나타낸다.
- RESTful 게시판은 4가지 메서드를 모두 사용하여 CRUD를 처리하며, URI는 제어하려는 자원을 나타낸다.

AOP는

2. JSON과 XML

■ RESTful 웹서비스와 JSON/XML



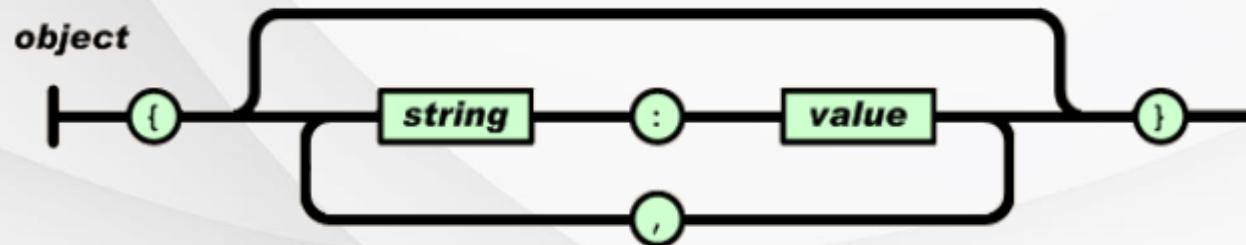
I JSON(JavaScript Object Notation)이란?

- ◎ <http://www.json.org>
- ◎ JSON은 경량(lightweight)의 DATA-교환 형식
- ◎ Javascript에서 객체를 만들 때 사용하는 표현식을 의미한다.
- ◎ JSON 표현식은 사람과 기계 모두 이해하기 쉬우며 용량이 작아서,
최근에는 JSON이 XML을 대체해서 데이터 전송 등에 많이 사용한다.
- ◎ 특정 언어에 종속되지 않으며, 대부분의 프로그래밍 언어에서
JSON 포맷의 데이터를 핸들링 할 수 있는 라이브러리를 제공하고 있다.

I JSON(JavaScript Object Notation) 형식

1. name-value 형식의 쌍(pair)

: 여러 가지 언어들에서 object, hashtable, struct로 실현되었다.



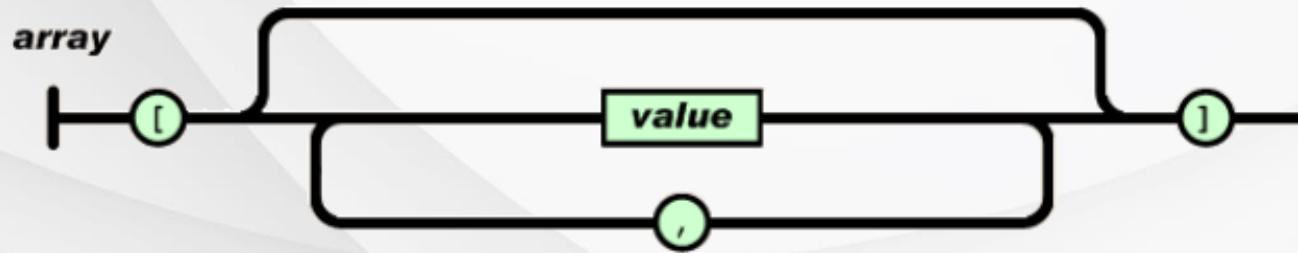
예시

```
{  
    "firstName": "Brett",  
    "lastName": "McLaughlin",  
    "email": "brett@newInstance.com"  
}
```

I JSON(JavaScript Object Notation) 형식

2. 값들의 순서화된 리스트 형식

: 여러 가지 언어들에서 array, list로 실현되었다.

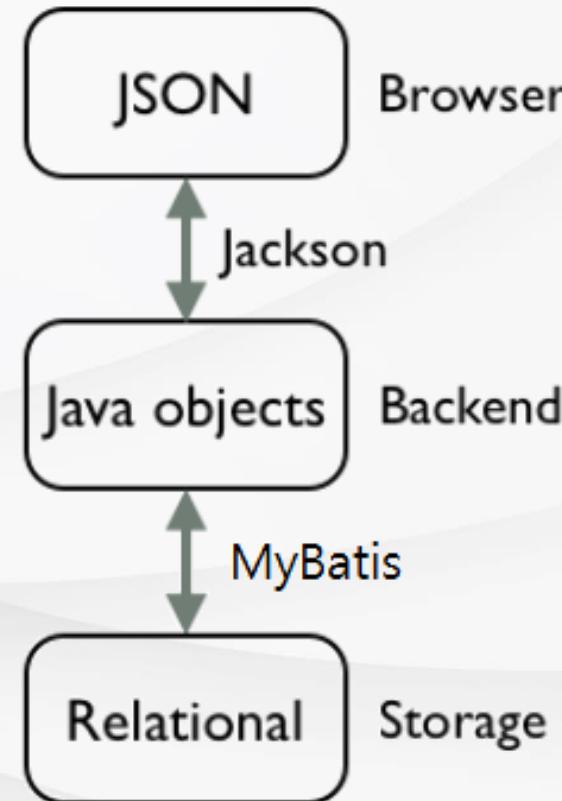


예시

```
{  
    "firstName": "Brett",  
    "lastName": "McLaughlin",  
    "email": brett@newInstance.com,  
    "hobby": ["puzzles", "swimming"]  
}
```

I JSON 라이브러리 - Jackson

- <http://jackson.codehaus.org>
- High-Performance JSON Processor!
- Jackson은 JSON 형태를 Java 객체로, Java 객체를 JSON 형태로 변환해주는 Java 용 JSON 라이브러리이다.
- 가장 많이 사용하는 JSON 라이브러리이다.



| XML(eXtensible Markup Language) 이란?

- ◎ XML은 Data를 저장하고 전달(교환) 하기 위한 언어이다.
- ◎ 인간/기계 모두에게 읽기 편한 언어이다.
- ◎ XML은 데이터의 구조와 의미를 설명합니다.

❖ XML과 HTML의 차이

XML	HTML
Data를 전달하는 것에 포커스를 맞춘 언어	Data를 표현하는 것에 포커스를 맞춘 언어
사용자가 마음대로 Tag를 정의할 수 있다	미리 정의된 Tag만 사용할 수 있다

| XML(eXtensible Markup Language) Tree 구조

- XML 문서는 “root”에서 시작해서 “leaves”로 뻗어가는 트리 구조
- XML 버전과 문자 인코딩을 정의하는 선언부(prolog)
- XML는 어떠한 데이터를 설명하기 위해 이름을 임의로 작성한 Tag로 데이터를 감싼다.

```
<? xml version="1.0" encoding= "UTF-8"?>  
<customer>  
  <name>홍길동</name>  
  <addr>서울</addr>  
  <phone>010-1234-5678</phone>  
</customer>
```

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

3. SpringMVC 기반 RESTful 서비스 환경설정

I Jackson Library 설치

http://mvnrepository.com에 접근

jackson mapper로 검색

jackson mapper 1.9.13 버전을 pom.xml에 추가

```
<!-- http://mvnrepository.com/artifact/org.codehaus.jackson/jackson-core-asl -->
<dependency>
    <groupId>org.codehaus.jackson</groupId>
    <artifactId>jackson-mapper-asl</artifactId>
    <version>1.9.13</version>
</dependency>
```

| web.xml의 DispatcherServlet url-pattern 변경

기
존

```
<servlet-mapping>
    <servlet-name>springDispatcherServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

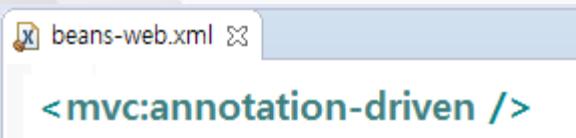
변
경

```
<servlet-mapping>
    <servlet-name>springDispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```



| Spring Bean Configuration 파일(bean-web.xml) 설정

- ◎ Spring MVC에 필요한 Bean들을 자동으로 등록해주는 태그

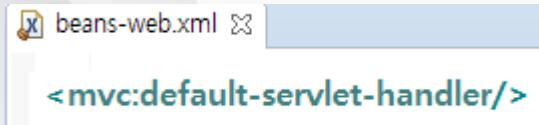


- ❖ annotation-driven 태그가 JSON과 관련하여 내부적으로 처리하는 설정

```
<bean id="jsonHttpMessageConverter"
      class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter" />
<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">
    <property name="messageConverters">
        <list>
            <ref bean="jsonHttpMessageConverter"/>
        </list>
    </property>
</bean>
```

| Spring Bean Configuration 파일(bean-web.xml) 설정

- DispatcherServlet의 변경된 url-pattern 때문에 필요한 태그 설정



❖ default-servlet-handler 태그를 설정하는 이유

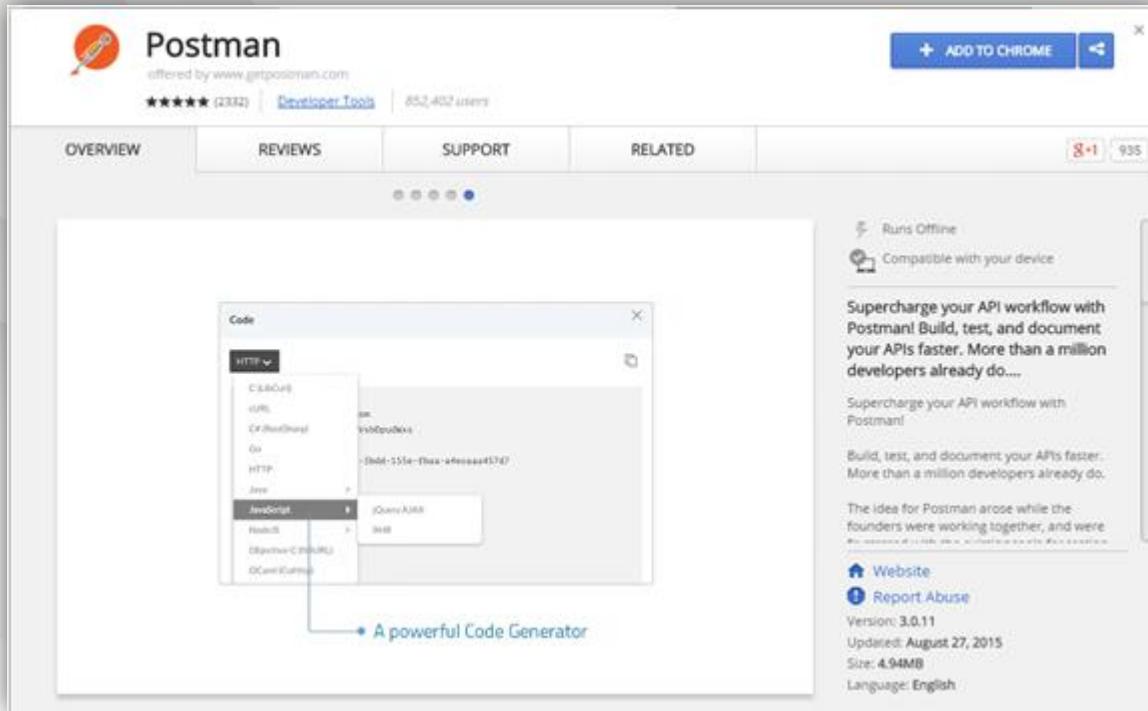
DispatcherServlet은 url-pattern을 "/" 와 같이 설정하게 되면서 tomcat의 server.xml에 정의되어 있는 url-pattern "/"을 무시하게 된다. 결국 DispatcherServlet url-pattern을 재정의하게 되어서 DefaultServlet은 더 이상 동작할 수 없게 된 것이다. Spring에서는 이를 해결하기 위해서 <mvc:default-servlet-handler /> 설정을 지원한다.

| Spring MVC기반 RESTful 웹서비스 구현 절차

- ① RESTful 웹서비스를 처리할 RestfulController 클래스 작성 및 Spring Bean으로 등록
- ② 요청을 처리할 메서드에 @RequestMapping
@RequestBody와 @ResponseBody 어노테이션 선언
- ③ REST Client Tool (Postman)을 사용하여 각각의 메서드 테스트
- ④ Ajax 통신을 하여 RESTful 웹서비스를 호출하는 HTML 페이지 작성

Postman: REST API 테스트하는 Chrome 확장 프로그램 설치

<https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop>



3. Spring MVC 기반 RESTful 웹서비스 환경설정

Tacademy

I Postman 실행

The screenshot shows the Postman application window. On the left, there is a sidebar titled "Chrome 앱" containing icons for "Advanced REST client" and "Postman". A large orange arrow points from the "Postman" icon towards the main Postman window. The main window has a dark header bar with tabs for "Runner", "Import", "Builder" (which is selected), "Team Library", and various status indicators. The URL bar shows "http://localhost:8080/Spring" with a "+" button. To the right of the URL bar is a dropdown for "No Environment" and a gear icon. The main content area is divided into sections: "GET" (selected), "http://localhost:8080/SpringWebPrj/users", "Params", "Send" (highlighted in blue), and "Save". Below these are tabs for "Authorization", "Headers", "Body", "Pre-request Script", and "Tests". The "Authorization" tab is selected, showing a dropdown menu with "Type" and "No Auth". On the far right of this section is a "Generate Code" button. To the left of the main content area, there is a "History" section titled "Today" with several recent requests listed:

- GET http://localhost:8080/SpringWebPrj/users
- DEL http://localhost:8080/SpringWebPrj/users/doolby2
- PUT http://localhost:8080/SpringWebPrj/users
- POST http://localhost:8080/SpringWebPrj/users
- POST http://localhost:8080/SpringWebPrj/users

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring RESTful 웹서비스 개요 및 환경설정]에 대해서 살펴보았습니다.

RESTful 웹서비스 개요

- Open API 와 REST
- RESTful API 개요

JSON과 XML

- JSON 개요, Jackson 라이브러리
- XML 개요

Spring MVC 기반 RESTful 웹서비스 환경설정

- Jackson 설치, Spring Bean Config 파일 설정, Postman 설치

Spring Framework

25. Spring RESTful 웹서비스 어플리케이션 작성(1)

CONTENTS

- 1 사용자 관리 RESTful 웹서비스 개요**
- 2 사용자 정보 조회 및 등록 기능 구현**
- 3 사용자 정보 수정 및 삭제 기능 구현**

학습 목표

- 사용자 관리 RESTful 웹서비스 개요에 대하여 이해할 수 있습니다.
- 사용자 정보 조회 및 등록 기능 구현에 대하여 이해할 수 있습니다.
- 사용자 정보 수정 및 삭제 기능 구현에 대해 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

1. 사용자 관리 RESTful 웹서비스 개요

| 사용자 관리 RESTful 웹서비스 URI와 Method

Action	Resource URI	HTTP Method
사용자 목록	/users	GET
사용자 보기	/users/{id}	GET
사용자 등록	/users	POST
사용자 수정	/users	PUT
사용자 삭제	/users/{id}	DELETE

■ RESTful Controller를 위한 핵심 어노테이션(Annotation)

- ◎ Spring MVC에서는 클라이언트에서 전송한 XML이나 JSON 데이터를 Controller에서 Java 객체로 변환해서 받을 수 있는 기능(수신)을 제공하고 있다.
- ◎ Java 객체를 XML이나 JSON으로 변환해서 전송할 수 있는 기능(송신)을 제공하고 있다.

어노테이션	설명
@RequestBody	HTTP Request Body(요청 몸체)를 Java 객체로 전달받을 수 있다.
@ResponseBody	Java 객체를 HTTP Response Body(응답 몸체)로 전송할 수 있다.

RESTful Controller를 위한 @ResponseBody가 있는 경우

```
@Controller  
@RequestMapping("/user")  
public class UserController {  
    @RequestMapping( value="/json/{id}", method = RequestMethod.GET)  
    @ResponseBody  
    public UserModel getByIdInJSON( @PathVariable String id){  
        UserModel user = new UserModel();  
        user.setId( id);  
        user.setName( "ellie");  
        return user;  
    }  
}
```

- @ResponseBody가 있는 getByIdInJSON 메서드의 경우

: MappingJacksonHttpMessageConverter가 리턴값인 UserModel 객체를 JSON으로 변환하는 작업을 처리한다.

| RESTful Controller를 위한 @ResponseBody가 없는 경우

```
@Controller  
@RequestMapping("/user")  
public class UserController {  
    @RequestMapping( value="/html/{id}", method = RequestMethod.GET)  
    public String getByIdInHTML( @PathVariable String id, ModelMap model){  
        UserModel user = new UserModel();  
        user.setId( id);  
        user.setName( "ellie");  
        model.addAttribute( "user", user);  
        return "user";  
    }  
}
```

- ◎ @ResponseBody가 없는 getByIdInHTML 메서드의 경우

: ViewResolver에 의해 선택된 /user.jsp가 포워드 되어지고,
user.jsp에서 UserModel 객체를 참조한다.



2. 사용자 정보 조회 및 등록 기능 구현

| 사용자 정보 조회 : 사용자 목록 조회 메서드 구현

```
RestfulUserController.java
```

```
@Controller
public class RestfulUserController {
    @Autowired
    private UserService userService;

    @RequestMapping(value="/users",method=RequestMethod.GET)
    @ResponseBody
    public Map getUserList() {
        List<UserVO> userList = userService.getUserList();
        Map result = new HashMap();
        result.put("result", Boolean.TRUE);
        result.put("data", userList);
        return result;
    }
}
```

2. 사용자 정보 조회 및 등록 기능 구현

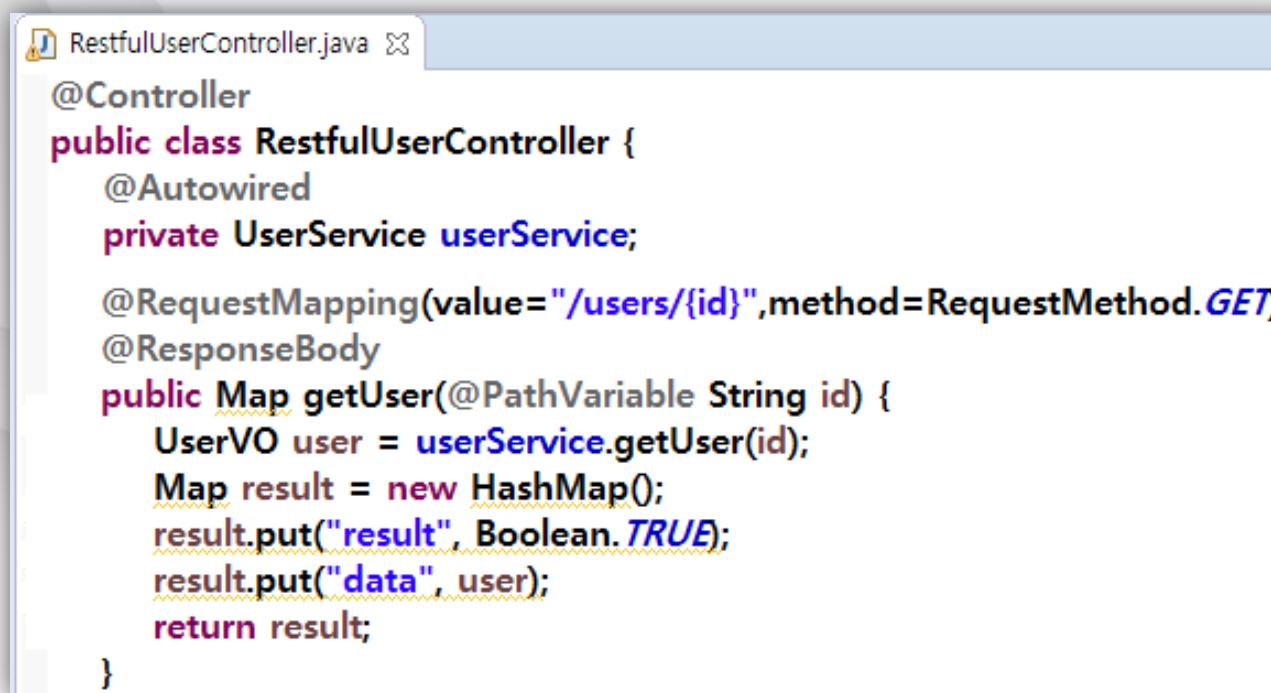
Tacademy

| 사용자 정보 조회 : 사용자 목록 조회 메서드 테스트

The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, the URL 'http://localhost:8080/SpringWebPrj/users', and buttons for 'Params', 'Send', 'Save', and 'Generate Code'. Below the header, there are tabs for 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Authorization' tab is currently active, showing 'Type' set to 'No Auth'. In the main body area, there are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Tests'. The 'Body' tab is active, showing a status of 'Status: 200 OK' and 'Time: 93 ms'. Below the tabs, there are buttons for 'Pretty', 'Raw', 'Preview', and 'JSON'. The JSON response is displayed in a code editor-like format with line numbers from 1 to 17. The response data is as follows:

```
1 {  
2   "result": true,  
3   "data": [  
4     {  
5       "userId": "doolby",  
6       "name": "둘리",  
7       "gender": "남",  
8       "city": "서울"  
9     },  
10    {  
11      "userId": "test",  
12      "name": "테스트",  
13      "gender": "여",  
14      "city": "경기"  
15    }  
16  ]  
17 }
```

| 사용자 정보 조회 : 특정 사용자 조회 메서드 구현



The screenshot shows a Java code editor window with a file named `RestfulUserController.java`. The code implements a REST controller for user information. It includes annotations for `@Controller`, `@Autowired`, `@RequestMapping`, and `@ResponseBody`. The `getUser` method takes a `String id` as a path variable and returns a `Map` containing a boolean key "result" and a `UserVO` object under the key "data".

```
RestfulUserController.java
@Controller
public class RestfulUserController {
    @Autowired
    private UserService userService;

    @RequestMapping(value="/users/{id}",method=RequestMethod.GET)
    @ResponseBody
    public Map getUser(@PathVariable String id) {
        UserVO user = userService.getUser(id);
        Map result = new HashMap();
        result.put("result", Boolean.TRUE);
        result.put("data", user);
        return result;
    }
}
```

2. 사용자 정보 조회 및 등록 기능 구현

Tacademy

| 사용자 정보 조회 : 특정 사용자 조회 메서드 테스트

GET <http://localhost:8080/SpringWebPrj/users/test> Params Send Save

Authorization Headers Body Pre-request Script Tests Generate Code

Type No Auth

Body Cookies Headers (4) Tests Status: 200 OK Time: 71 ms

Pretty Raw Preview JSON 🔍

```
1 | {  
2 |   "result": true,  
3 |   "data": {  
4 |     "userId": "test",  
5 |     "name": "테스트",  
6 |     "gender": "여",  
7 |     "city": "경기"  
8 |   }  
9 | }
```

| 사용자 정보 등록 : 사용자 등록 메서드 구현

```
RestfulUserController.java
```

```
@Controller
public class RestfulUserController {
    @Autowired
    private UserService userService;

    @RequestMapping(value="/users",
                   method=RequestMethod.POST,
                   headers = {"Content-type=application/json"})
    @ResponseBody
    public Map insertUser(@RequestBody UserVO user) {
        if (user != null)
            userService.insertUser(user);

        Map result = new HashMap();
        result.put("result", Boolean.TRUE);
        return result;
    }
}
```

2. 사용자 정보 조회 및 등록 기능 구현

Tacademy

| 사용자 정보 등록 : 사용자 등록 메서드 테스트

POST Params **Send** **Save**

Authorization Headers (1) Body Pre-request Script Tests **Generate Code**

Content-Type application/json; charset=UTF-8 **Bulk Edit** **Presets**

key value

Authorization Headers (1) Body Pre-request Script Tests **Generate Code**

form-data x-www-form-urlencoded raw binary **JSON (application/json)**

```
1 {  
2   "userId": "vega2k",  
3   "name": "박소율",  
4   "gender": "여",  
5   "city": "경기"  
6 }
```



3. 사용자 정보 수정 및 삭제 기능 구현

| 사용자 정보 수정 : 사용자 수정 메서드 구현

```
RestfulUserController.java
```

```
@Controller
public class RestfulUserController {
    @Autowired
    private UserService userService;
    @RequestMapping(value="/users",
                   method=RequestMethod.PUT,
                   headers = {"Content-type=application/json"})
    @ResponseBody
    public Map updateUser(@RequestBody UserVO user) {
        if(user != null)
            userService.updateUser(user);
        Map result = new HashMap();
        result.put("result", Boolean.TRUE);
        return result;
    }
}
```

| 사용자 정보 수정 : 사용자 수정 메서드 테스트

The screenshot shows the Postman application interface. At the top, there is a header bar with a 'PUT' method dropdown, a URL input field containing 'http://localhost:8080/SpringWebPrj/users', a 'Params' button, a 'Send' button, and a 'Save' button. Below the header, there are tabs for 'Authorization', 'Headers (1)', 'Body ●', 'Pre-request Script', and 'Tests'. The 'Headers (1)' tab is currently selected. A table below it shows one header entry: 'Content-Type' with a value of 'application/json; charset=UTF-8'. There are 'Bulk Edit' and 'Presets' buttons to the right of the table.

The screenshot shows the 'Body' tab of the Postman interface. It has tabs for 'Authorization', 'Headers (1)', 'Body ●', 'Pre-request Script', and 'Tests'. The 'Body ●' tab is selected. Below it, there are radio buttons for 'form-data', 'x-www-form-urlencoded', 'raw', and 'binary'. The 'raw' radio button is selected, and its dropdown menu shows 'JSON (application/json)'. The main area contains a code editor with the following JSON payload:

```
1 {  
2   "userId": "vega2k",  
3   "name": "박소율New",  
4   "gender": "남",  
5   "city": "부산"  
6 }
```

| 사용자 정보 삭제 : 사용자 삭제 메서드 구현

```
RestfulUserController.java ✘  
@Controller  
public class RestfulUserController {  
    @Autowired  
    private UserService userService;  
  
    @RequestMapping(value = "/users/{id}", method = RequestMethod.DELETE)  
    @ResponseBody  
    public Map deleteUser(@PathVariable String id) {  
        userService.deleteUser(id);  
  
        Map result = new HashMap();  
        result.put("result", Boolean.TRUE);  
        return result;  
    }  
}
```

3. 사용자 정보 수정 및 삭제 기능 구현

Tacademy

| 사용자 정보 삭제 : 사용자 삭제 메서드 테스트

DELETE http://localhost:8080/SpringWebPrj/users/dooly Params

Authorization Headers Body Pre-request Script Tests

Type

Body Cookies Headers (4) Tests Status: 200 OK Time: 55 ms

Pretty Raw Preview

```
1 [ {  
2   "result": true  
3 } ]
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring RESTful 웹서비스 어플리케이션 작성(1)]에 대해서 살펴보았습니다.

사용자 관리 RESTful 웹서비스 개요

- Resource URI와 Http Method 규칙
- @RequestBody, @ResponseBody 어노테이션

사용자 조회 및 등록 기능 구현

- RESTful 방식의 사용자 조회 및 등록하는 기능 구현
- Postman을 이용한 조회, 등록 테스트

사용자 수정 및 삭제 기능 구현

- RESTful 방식의 사용자 수정 및 삭제하는 기능 구현
- Postman을 이용한 수정, 삭제 테스트

Spring Framework

26. Spring RESTful 웹서비스 어플리케이션 작성(2)

CONTENTS

1 Ajax 개요

2 jQuery 개요

3 jQuery가 제공하는 기능

학습 목표

- Ajax 개요에 대하여 이해할 수 있습니다.
- jQuery 개요에 대하여 이해할 수 있습니다.
- jQuery가 제공하는 기능에 대해 이해할 수 있습니다.

A photograph of a person's hands holding a smartphone. The phone screen is white. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

AOP는

1. Ajax 개요

I Ajax(Asynchronous Javascript and Xml)란?

웹2.0을 실현하는 핵심기능인 Ajax는 웹 사용자들에게 좀 더 수준 높은 인터페이스를 제공할 수 있도록 도움을 주는 기술의 묶음이다.

- Ajax는 그 자체가 특정한 기술이 아니라, HTML과 CSS, Javascript, XML, XMLHttpRequest객체를 비롯한 기존의 여러 기술들을 조합해서 사용하는 “웹의 새로운 접근법”이라고 설명할 수 있다.
- “비동기적”이라는 의미는 서버로부터 데이터가 로드 되는 동안에도 계속해서 페이지를 사용할 수 있다는 뜻으로, 서버가 데이터를 전달해 주면 Ajax 이벤트가 발생하여 서버로부터 받은 데이터를 읽어 페이지의 일부를 수정하게 된다.

I Ajax의 활용 예시

01 라이브 검색

- ◎ 라이브 검색(혹은 자동완성)이라고 불리는 기능은 주로 Ajax를 이용, 검색Site의 대부분이 사용하는 기술이고, 검색어를 입력하는 동시에 검색 결과가 나타나도록 한다.

02 사용자 정보 표시

- ◎ 회원가입 시 중복 아이디일 경우, “이미 사용 중인 아이디입니다”를 아이디 입력과 동시에 중복 메시지를 표시해주는 기능에서도 활용되고 있음
- ◎ 온라인 쇼핑몰에서 장바구니에 원하는 물품을 추가했을 때 페이지 이동이나 전체 페이지의 새로 고침 없이도 물품 정보만 추가되는 기능을 구현하고자 할 때도 Ajax를 활용한다.

■ XMLHttpRequest(XHR) 객체 사용법

- ① XMLHttpRequest 객체 생성 : Request를 보낼 준비

```
var xhr = new XMLHttpRequest();
```

- ② Callback 함수를 만든다 : 서버에서 Response가 왔을 때 실행되는 함수

```
xhr.onreadystatechange = function(){  
    if(xhr.readyState == 4){  
        var myDiv = document.getElementById('myDiv');  
        myDiv.innerHTML= xhr.responseText;  
    }  
}
```

I XMLHttpRequest(XHR) 객체 사용법

- ③ Request를 Open 한다 : HTTP method와 호출할 Server의 url 정보를 전달

```
xhr.open("GET", "user.do");
```

- ④ Request를 Send 한다.

```
xhr.send();
```

| jQuery를 이용하여 Ajax 통신을 하는 경우

일반 Javascript로 Ajax 코드를 작성하게 되면 코딩량도 많아지고,
브라우저 별로 구현을 다르게 해주어야 하는 단점이 있다.

jQuery를 이용하면 더 적은 코딩량과 동일한 코드로 대부분의
브라우저에서 같은 동작을 할 수 있도록 해준다.
즉, 크로스 브라우징이 쉬워진다.

jQuery를 이용하면 100줄 정도의 Source 코드를
단 몇 줄 만으로 간단하게 서버와 데이터를 주고 받을 수 있다.

A photograph of a person's hands holding a black smartphone. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city street.

2. jQuery 개요

I jQuery(<http://www.jquery.com>)란?

Javascript를 좀 더 쉽게 사용하도록 만들어진
Javascript Library로서, jQuery는 가볍고 빠르며
코드의 간결함을 제공하는 오픈 소스이다.

jQuery의 특징

- 가벼움 (jquery-1.9.1.min.js의 경우 90.4KB)
- 크로스 브라우저 지원
- 강력한 CSS 셀렉터
- 메서드 체이닝
- Ajax 지원
- 풍부한 Plugin 지원

I jQuery 라이브러리 준비

1. CDN(Content Delivery Network) 호스트를 사용하는 방법

- 구글과 마이크로소프트에서 CDN 호스트를 지원함
- jQuery도 CDN 호스트 지원
→ <https://code.jquery.com/jquery-1.12.4.js>

2. http://www.jquery.com에서 직접 다운로드 하여 로컬에 저장 후에 사용하는 방법

I jQuery를 사용하여 코드의 단순화

jQuery 없는 DOM 스크립팅

```
var external_links =  
    document.getElementById('external_links');  
var links =  
    external_links.getElementsByTagName('a');  
for(var i=0; i < links.length; i++){  
    var link = links.item(i);  
    link.onClick = function(){  
        return confirm('You are going to visit: '  
            + this.href);  
    }  
} //for
```

jQuery를 사용한 DOM 스크립팅

```
$('#external_links a').click(function(){  
    return confirm('You are going to visit: '  
        + this.href);  
});
```

I jQuery가 제공하는 기능

1. HTML 엘리먼트 선택(Selector)
2. HTML 엘리먼트의 attribute 값 읽기와 쓰기
3. HTML 엘리먼트 동적으로 조작(Manipulation)
4. Loop 기능
5. CSS 조작
6. Event 처리
7. Ajax 처리

I jQuery 시작하기

- jQuery를 사용하는 모든 웹 페이지는 `$(document).ready()`로 시작해야 함.

```
<script>  
    $(document).ready(function(){  
        //페이지 로드시에 해야 할일  
    });  
    //또는 ready() 함수의 축약버전으로  
    $(function() {  
        //페이지 로드시에 해야 할일  
    });  
</script>
```

A photograph of a person's hands holding a black smartphone. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city street.

3. jQuery가 제공하는 기능

| Selector

```
//태그명 직접입력 선택  
$("div");  
  
// 클래스명으로 선택  
$(".class");  
  
//ID 명으로 선택  
$("#id");  
  
// 모든 Element 선택 하여 CSS적용  
$("*").css("border","1px");  
  
//다중 셀렉터 기준으로 해당 엘리먼트들 모두 선택  
$("div,span,p.myClass");  
  
//DIV요소중 3번째 이후 부터 ((index) 보다 큰(이후) 요소)  
$("div:gt(2)");  
  
// <input type="checkbox">  
$(" :checkbox");
```

I Attribute 값 읽기와 쓰기

- jQuery를 attr(attrName), attr(attrName, value), attr(object) 함수

: 선택된 element의 속성(attribute)의 값을 리턴, 설정한다.

적용 전

```
<img id="greatphoto">
```

attr()

```
$("#greatphoto").attr("src","brush-seller.jpg");
$("#greatphoto").attr("src");
$("#greatphoto").attr({
  alt: "Brush Seller",
  title: "photo by Kelly Clark"
});
```

적용 후

```

```

I Manipulation

◎ append() 함수

: 선택된 element의 content 맨 끝에 인자로 넘어온 내용을 추가한다.

◎ appendTo() 함수

: 선택된 element를 target에 해당하는 Element의 content의 끝에 추가한다.

```
$(".inner").append("<p>Test</p>");  
$("<p>Test</p>").appendTo(".inner");
```

◎ html(), html(htmlString) 함수

: 선택된 Element의 html을 리턴, 설정한다.

```
$("#div.demo-container").html();  
$("div").html("<h1>제목</h1>");
```

I Loop 기능

◎ `.each(function(index, element))` 함수

: jQuery 객체들을 위해 반복 로직을 처리하는 함수

```
<ul>
  <li>foo</li>
  <li>bar</li>
</ul>
```

```
$('.li').each(function(index,element) {
  alert(index + ': ' + $(element).text());
});
```

| CSS 조작

- ◎ **css(propertyName), css(propertyName, value), css(object) 함수**
: 선택된 element의 propertyName에 해당되는 CSS정보를 리턴, 설정한다.

```
$("span").css("width");
$("span").css("width", "100px");
$("span").css({"width":"100px",
  "height":"50px"});
```

- ◎ **addClass(className) 함수**
: 선택된 element에 해당되는 클래스 추가

```
$("p").addClass("myClass");
```

I Event 처리

◎ on() 함수

: 선택된 element에 이벤트 핸들러를 묶어준다.

```
$("p").on("click", function(){
    alert( $(this).text() );
});
```

```
$("div.test").on({
    mouseenter: function(){
        $(this).addClass("inside");
    },
    mouseleave: function(){
        $(this).removeClass("inside");
    }
});
```

I Ajax 처리

◎ \$.ajax() 함수

```
$('#btnSelect').on('click',function(){
    $.ajax({
        url:'users',
        type:'GET',
        contentType:'application/json; charset=utf-8',
        dataType:'json',
        error:function(xhr,status,msg){
            alert("상태값 :" + status + " Http에러메시지 :" + msg);
        },
        success:userSelectResult
    });
});
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring RESTful 웹서비스 어플리케이션 작성(2)]에 대해서 살펴보았습니다.

Ajax개요

- Ajax 개념 및 활용예시
- XMLHttpRequest (XHR) 객체

jQuery 개요

- jQuery 개념 및 특징
- jQuery 라이브러리 준비, `$(document).ready()`

jQuery가 제공하는 기능

- Selector, attribute 조작, Manipulation, CSS 조작, Event, Ajax

Spring Framework

27. Spring RESTful 웹서비스 어플리케이션 작성(3)

CONTENTS

1

사용자 관리 RESTful 웹서비스 클라이언트

2

사용자정보 조회 및 등록 기능 구현

3

사용자정보 수정 및 삭제 기능 구현

학습 목표

- 사용자 관리 RESTful 웹서비스 클라이언트에 대하여 이해할 수 있습니다.
- 사용자정보 조회 및 등록기능에 대하여 이해할 수 있습니다.
- 사용자정보 수정 및 삭제기능에 대해 이해할 수 있습니다.



1. 사용자 관리 RESTful 웹서비스 클라이언트

AOP는

| 클라이언트 프로그램 개요

① RESTful 웹서비스를 호출하기 위해 Ajax 통신방법을 이용함

② Ajax 통신을 하는 복잡한 코드를 간결하게 작성하기 위해
jQuery의 `$.ajax()` 함수를 사용함

③ 서버로 부터 받은 데이터를 이용하여 동적으로 테이블의 Row를
증가시킴

④ 사용자를 등록, 수정할 때 입력한 데이터를 JSON 포맷으로
변경하여 서버에게 전송함

⑤ 화면의 스타일과 관련된 코드를 좀 더 쉽게 작성하기 위해서
Bootstrap을 사용하였음

A photograph of a person's hands holding a smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

2. 사용자 정보 조회 및 등록 기능 구현

| 사용자 목록 조회(1)

사용자 목록

아이디	이름	성별	거주지	조회	삭제
dooly	둘리	남	제주	조회	삭제
jay	박정우	남	부산	조회	삭제
vega2k	박소을	여	경기	조회	삭제

| 사용자 목록 조회(2)

Html 코드

```
<div class= "container">
  <h2>사용자 목록 </h2>
  <table class= "table text-center">
    <thead>
      <tr >
        <th class= "text-center">아이디 </th>
        <th class= "text-center">이름 </th>
        <th class= "text-center">성별 </th>
        <th class= "text-center">거주지 </th>
      </tr>
    </thead>
    <tbody> </tbody>
  </table>
</div>
```

jQuery 코드

```
<script src= "https://code.jquery.com/jquery-1.12.4.min.js" ></script>
<script type= "text/javascript" >
  $(function(){
    userList();
  });
</script>
```

| 사용자 목록 조회(3)

Client : Ajax 요청

```
//사용자 목록 조회 요청
function userList() {
    $.ajax({
        url:'users',
        type:'GET',
        contentType:'application/json;charset=utf-8',
        dataType:'json',
        error:function(xhr,status,msg){
            alert("상태값 :" + status + " Http에러메시지 :" +msg);
        },
        success:userListResult
    });
//userList
```

Server : 컨트롤러

```
@RequestMapping(value="/users",
                 method=RequestMethod.GET)
@ResponseBody
public Map getUserList() {
    List<UserVO> userList = userService.getUserList();
    Map result = new HashMap();
    result.put("result", Boolean.TRUE);
    result.put("data", userList);
    return result;
}
```

| 사용자 목록 조회(4)

Client : Ajax 응답

```
//사용자 목록 조회 응답
function userListResult(xhr) {
    console.log(xhr.data);
    $("tbody").empty();
    $.each(xhr.data, function(idx, item){
        $('<tr>')
            .append($('<td>').html(item.userId))
            .append($('<td>').html(item.name))
            .append($('<td>').html(item.gender))
            .append($('<td>').html(item.city))
            .append($('<td>').html('<button id=WbtnSelectW>조회</button>'))
            .append($('<td>').html('<button id=WbtnDeleteW>삭제</button>'))
            .append($('<input type=WhiddenW id=Whidden_userIdW>').val(item.userId))
            .appendTo('tbody');
    });
}
//userListResult
```

Server : 컨트롤러

```
@RequestMapping(value="/users",
method=RequestMethod.GET)
@ResponseBody
public Map getUserList() {
    List<UserVO> userList = userService.getUserList();
    Map result = new HashMap();
    result.put("result", Boolean.TRUE);
    result.put("data", userList);
    return result;
}
```

2. 사용자 정보 조회 및 등록 기능 구현

| 특정 사용자 조회(1)

사용자 목록				
아이디	이름	성별	거주지	
dooly	둘리	남	제주	<button>조회</button> <button>삭제</button>
jay	박정우	남	부산	<button>조회</button> <button>삭제</button>
vega2k	박소율	여	경기	<button>조회</button> <button>삭제</button>

사용자 등록 및 수정

아이디:

이름:

성별:

남
 여

거주지:

등록 수정 초기화

I 특정 사용자 조회(2)

동적 Html 코드

```
//사용자 목록 조회 응답
function userListResult(xhr) {
    console.log(xhr.data);
    $("tbody").empty();
    $.each(xhr.data,function(idx,item){
        $('<tr>')
            .append($('<td>').html(item.userId))
            .append($('<input type='hidden' id='hidden_userId'').val(item.userId))
            .append($('<td>').html(item.name))
            .append($('<td>').html(item.gender))
            .append($('<td>').html(item.city))
            .append($('<td>').html('<button id='btnSelect'>조회</button>'))
            .append($('<td>').html('<button id='btnDelete'>삭제</button>'))
            .appendTo('tbody');
    });
}
//userListResult
```

jQuery 코드

```
<script type="text/javascript" >
$(function(){
    userList();
    userSelect();
});
</script>
```

I 특정 사용자 조회(3)

Client : Ajax 요청

```
//사용자 조회 요청
function userSelect() {
    //조회 버튼 클릭
    $('#body').on('click', '#btnSelect', function(){
        var userId = $(this).closest('tr').find('#hidden_userId').val();
        //특정 사용자 조회
        $.ajax({
            url:'users/' +userId,
            type:'GET',
            contentType:'application/json; charset=utf-8',
            dataType:'json',
            error:function(xhr,status,msg){
                alert("상태값 :" + status + " Http에러메시지 :" +msg);
            },
            success:userSelectResult
        });
    });
}); //조회 버튼 클릭
}//userSelect
```

Server : 컨트롤러

```
@RequestMapping(value= "/users/{id}",
                 method=RequestMethod.GET)
@ResponseBody
public Map getUser(@PathVariable String id) {
    UserVO user = userService.getUser(id);
    Map result = new HashMap();
    result.put("result", Boolean.TRUE);
    result.put("data", user);
    return result;
}
```

I 특정 사용자 조회(4)

Client : Ajax 응답

```
//사용자 조회 응답
function userSelectResult(xhr) {
    var user = xhr.data;
    $('input:text[name="userId"]').val(user.userId);
    $('input:text[name="name"]').val(user.name);
    $('input:radio[name="gender"]['+user.gender+']).prop('checked', true);
    $('select[name="city"]').val(user.city).attr("selected", "selected");
}
//userSelectResult
```

Server : 컨트롤러

```
@RequestMapping(value="/users/{id}",
    method=RequestMethod.GET)
@ResponseBody
public Map getUser(@PathVariable String id) {
    UserVO user = userService.getUser(id);
    Map result = new HashMap();
    result.put("result", Boolean.TRUE);
    result.put("data", user);
    return result;
}
```

2. 사용자 정보 조회 및 등록 기능 구현

Tacademy

I 특정 사용자 등록(1)

사용자 등록 및 수정

아이디:

이름:

성별:
 남
 여

거주지:

등록 수정 초기화

사용자 목록

아이디	이름	성별	거주지	조회	삭제
cloud	홍길동	남	경기	<button>조회</button>	<button>삭제</button>
dooly	둘리	남	제주	<button>조회</button>	<button>삭제</button>
jay	박정우	남	부산	<button>조회</button>	<button>삭제</button>
vega2k	박소율	여	경기	<button>조회</button>	<button>삭제</button>

2. 사용자 정보 조회 및 등록 기능 구현

I 특정 사용자 등록(2)

Html 코드

```
<form id= "form1" class= "form-horizontal">
  <h2>사용자 등록 및 수정</h2>
  <div class= "form-group">
    <label>아이디:</label>
    <input type= "text" class= "form-control" name= "userId" >
  </div>
  <div class= "form-group">
    <label>이름:</label>
    <input type= "text" class= "form-control" name= "name" >
  </div>
  .....
  <div class= "btn-group">
    <input type= "button" class= "btn btn-primary" value= "등록" id= "btnInsert" />
    <input type= "button" class= "btn btn-primary" value= "수정" id= "btnUpdate" />
    <input type= "button" class= "btn btn-primary" value= "초기화" id= "btnInit" />
  </div>
</form>
```

jQuery 코드

```
<script type= "text/javascript" >
  $(function(){
    userList();
    userSelect();
    userInsert();
  });
</script>
```

| 특정 사용자 등록(3)

Client : Ajax 요청

```

function userInsert(){
    //등록 버튼 클릭
    $('#btnInsert').on('click',function(){
        var userId = $('input:text[name="userId"]').val();
        var name = $('input:text[name="name"]').val();
        var gender = $('input:radio[name="gender"]:checked').val();
        var city = $('select[name="city"]').val();
        $.ajax({
            url: "users", type: 'POST', dataType: 'json',
            data: JSON.stringify({ userId: userId, name: name, gender: gender, city: city }),
            contentType: 'application/json', mimeType: 'application/json',
            success: function(response) {
                if(response.result == true) {
                    userList();
                }
            },
            error:function(xhr, status, message) {
                alert(" status: "+status+ " er:"+message);
            } });
    });//등록 버튼 클릭
}//userInsert

```

Server : 컨트롤러

```

@RequestMapping(value= "/users",
    method=RequestMethod.POST,
    headers = {"Content-type=application/json"})
@ResponseBody
public Map insertUser(@RequestBody UserVO user) {
    if (user != null)
        userService.insertUser(user);
    Map result = new HashMap();
    result.put("result", Boolean.TRUE);
    return result;
}

```



3. 사용자 정보 수정 및 삭제 기능 구현

3. 사용자 정보 수정 및 삭제 기능 구현

| 특정 사용자 수정(1)

사용자 등록 및 수정

아이디:
cloud

이름:
홍길동2

성별:

남
 여

거주지:
부산

[등록](#) [수정](#) [초기화](#)

사용자 목록

아이디	이름	성별	거주지	조회	삭제
cloud	홍길동2	여	부산	조회	삭제
dooly	둘리	남	제주	조회	삭제
jay	박정우	남	부산	조회	삭제
vega2k	박소율	여	경기	조회	삭제

3. 사용자 정보 수정 및 삭제 기능 구현

I 특정 사용자 수정(2)

Html 코드

```
<form id= "form1" class= "form-horizontal">
  <h2>사용자 등록 및 수정</h2>
  <div class= "form-group">
    <label>아이디:</label>
    <input type= "text" class= "form-control" name= "userId" >
  </div>
  <div class= "form-group">
    <label>이름:</label>
    <input type= "text" class= "form-control" name= "name" >
  </div>
  .....
  <div class= "btn-group">
    <input type= "button" class= "btn btn-primary" value= "등록" id= "btnInsert" />
    <input type= "button" class= "btn btn-primary" value= "수정" id= "btnUpdate" />
    <input type= "button" class= "btn btn-primary" value= "초기화" id= "btnInit" />
  </div>
</form>
```

jQuery 코드

```
<script type= "text/javascript" >
  $(function(){
    userList();
    userSelect();
    userInsert();
    userUpdate();
  });
</script>
```

3. 사용자 정보 수정 및 삭제 기능 구현

I 특정 사용자 수정(3)

Client : Ajax 요청

```
function userUpdate() {
    //수정 버튼 클릭
    $('#btnInsert').on('click', function() {
        var userId = $('input:text[name="userId"]').val();
        var name = $('input:text[name="name"]').val();
        var gender = $('input:radio[name="gender"]:checked').val();
        var city = $('select[name="city"]').val();
        $.ajax({
            url: "users", type: 'PUT', dataType: 'json',
            data: JSON.stringify({ userId: userId, name: name, gender: gender, city: city }),
            contentType: 'application/json', mimeType: 'application/json',
            success: function(response) {
                if(response.result == true) {
                    userList();
                }
            },
            error: function(xhr, status, message) {
                alert(" status: " + status + " er:" + message);
            }
        });
    });
}
```

Server : 컨트롤러

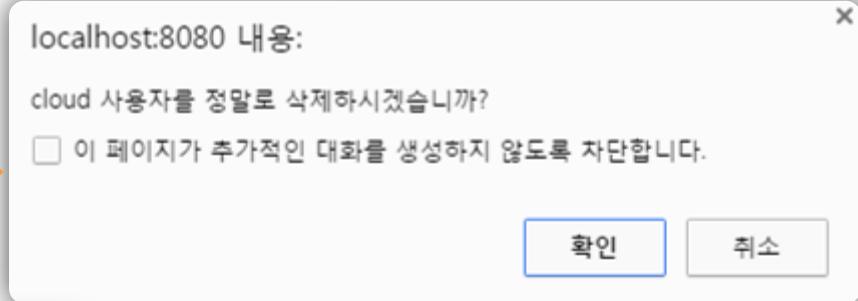
```
@RequestMapping(value = "/users",
    method=RequestMethod.PUT,
    headers = {"Content-type=application/json"})
@ResponseBody
public Map updateUser(@RequestBody UserVO user) {
    if(user != null)
        userService.updateUser(user);
    Map result = new HashMap();
    result.put("result", Boolean.TRUE);
    return result;
}
```

3. 사용자 정보 수정 및 삭제 기능 구현

| 특정 사용자 삭제(1)



사용자 목록				
아이디	이름	성별	거주지	
cloud	홍길동2	여	부산	<button>조회</button> <button>삭제</button>
dooly	둘리	남	제주	<button>조회</button> <button>삭제</button>
jay	박정우	남	부산	<button>조회</button> <button>삭제</button>
vega2k	박소율	여	경기	<button>조회</button> <button>삭제</button>



사용자 목록				
아이디	이름	성별	거주지	
dooly	둘리	남	제주	<button>조회</button> <button>삭제</button>
jay	박정우	남	부산	<button>조회</button> <button>삭제</button>
vega2k	박소율	여	경기	<button>조회</button> <button>삭제</button>

I 특정 사용자 삭제(2)

동적 Html 코드

```
//사용자 목록 조회 응답
function userListResult(xhr) {
    console.log(xhr.data);
    $("tbody").empty();
    $.each(xhr.data,function(idx,item){
        $('<tr>')
            .append($('<td>').html(item.userId))
            .append($('<input type='W'hiddenW' id='W'hidden_userIdW'>').val(item.userId))
            .append($('<td>').html(item.name))
            .append($('<td>').html(item.gender))
            .append($('<td>').html(item.city))
            .append($('<td>').html('<button id='W'btnSelectW'>조회 </button>'))
            .append($('<td>').html('<button id='W'btnDeleteW'>삭제 </button>'))
            .appendTo('tbody');
    });
}
//userListResult
```

jQuery 코드

```
<script type="text/javascript" >
$(function(){
    userList();
    userSelect();
    userInsert();
    userUpdate();
    userDelete();
});
</script>
```

I 특정 사용자 삭제(3)

Client : Ajax 요청

```
function userDelete() {
    //삭제 버튼 클릭
    $('body').on('click', '#btnDelete', function(){
        var userId = $(this).closest('tr').find('#hidden_userId').val();
        var result = confirm(userId + " 사용자를 정말로 삭제하시겠습니까?");
        if(result) {
            $.ajax({
                url:'users/' + userId, type:'DELETE',
                contentType:'application/json; charset=utf-8',
                dataType:'json',
                error:function(xhr,status,msg){
                    console.log("상태값 :" + status + " Http에러메시지 :" + msg);
                }, success:function(xhr) {
                    console.log(xhr.result);
                    userList();
                }
            });
        } //if
    });
} //삭제 버튼 클릭
//userDelete
```

Server : 컨트롤러

```
@RequestMapping(value="/users/{id}",
    method=RequestMethod.DELETE)
@ResponseBody
public Map deleteUser(@PathVariable String id) {
    userService.deleteUser(id);
    Map result = new HashMap();
    result.put("result", Boolean.TRUE);
    return result;
}
```

A photograph of a person's hands holding a black smartphone. The screen is white and blank. The background is dark with blurred, colorful circular lights, suggesting a night scene or a city at night.

학습정리



지금까지 [Spring RESTful 웹서비스 어플리케이션 작성(3)]에 대해서 살펴보았습니다.

사용자 관리 RESTful 웹서비스 클라이언트

- 클라이언트 프로그램 개요
- Ajax, jQuery, JSON, Bootstrap

사용자 정보 조회 및 등록 기능 구현

- 사용자 정보 조회
- 사용자 정보 등록

사용자 정보 수정 및 삭제 기능 구현

- 사용자 정보 수정
- 사용자 정보 삭제

Spring Framework

28. Spring RESTful 웹서비스 어플리케이션 작성(4)

CONTENTS

1

XML 응답을 주는 RESTful 웹서비스 개요

2

XML 응답을 주는 RESTful 웹서비스 구현

학습 목표

- XML 응답을 주는 RESTful 웹서비스
개요에 대하여 이해할 수 있습니다.

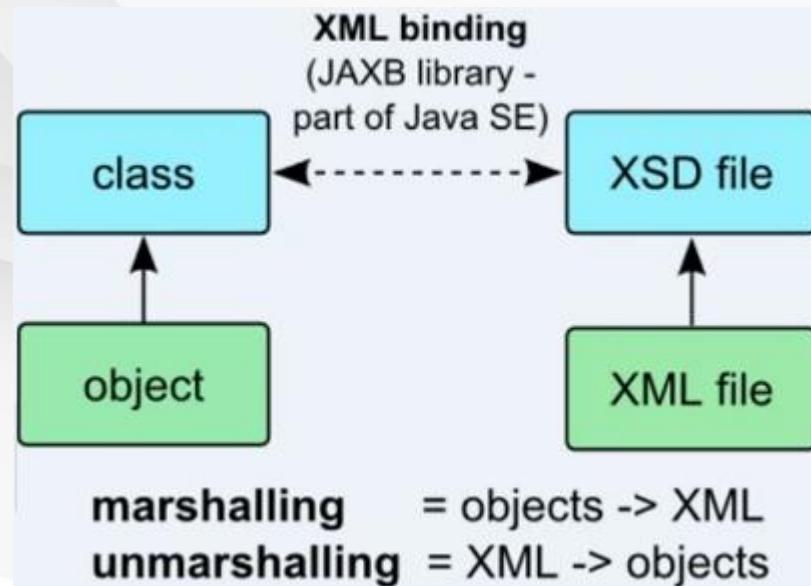
- XML 응답을 주는 RESTful 웹서비스
구현에 대하여 이해할 수 있습니다.

A photograph of a person's hands holding a black smartphone. The phone is held horizontally, with the screen facing the viewer. The background is dark and out of focus, featuring several blurred circular lights in shades of yellow, orange, and blue, suggesting a night-time urban setting.

1. XML 응답을 주는 RESTful 웹서비스 개요

I JAXB(Java Architecture for XML Binding)란?

JAXB는 Java 객체를 XML로 변환(직렬화, Marshalling) 해주고,
XML을 Java 객체로 변환(역직렬화, Unmarshalling) 해주는
작업을 좀 더 편리하게 할 수 있도록 해주는 API이다.



| 주요 JAXB 어노테이션(Annotation)

어노테이션	설명
@XmlRootElement	XML의 Root Element 이름을 정의한다. 클래스에서 사용하는 어노테이션으로 해당 클래스가 XML Root라는 것을 뜻합니다.
@XmlElement	XML의 Element 이름을 정의한다. 변수에 사용하는 어노테이션으로 해당 변수가 XML Element 라는 것을 뜻합니다.

I JAXB을 사용한 RESTful 웹서비스 구현 절차

- ① Java 객체를 XML로 변환하기 위해 @XMLRootElement와 @XMLElement 어노테이션을 선언한 UserVOXML 클래스를 작성
- ② RestfulController 클래스에 사용자 목록을 조회하는 getUserListXml() 메서드를 작성하고 @RequestMapping과 @ResponseBody 어노테이션을 선언
- ③ Postman 툴을 이용해 메서드를 테스트
- ④ jQuery기반 Ajax통신을 하는 userList_Xml.html를 작성



2. XML 응답을 주는 RESTful 웹서비스 구현

Java 객체를 XML로 변환 해주는 클래스 작성

UserVOXML.java

```
@XmlRootElement(name = "users")
public class UserVOXML {
    private String status;
    private List<UserVO> userList;

    public UserVOXML() {}

    public UserVOXML(String status, List<UserVO> userList) {
        this.status = status;
        this.userList = userList;
    }

    @XmlElement
    public void setStatus(String status) { this.status = status; }

    @XmlElement(name="user")
    public void setUserList(List<UserVO> userList) { this.userList = userList; }

    public String getStatus() { return status; }
    public List<UserVO> getUserList() { return userList; }
}
```

```
<users>
    <status>success</status>
    <user>
        <city>제주</city>
        <gender>남</gender>
        <name>둘리</name>
        <userId>dooly</userId>
    </user>
    <user>
        <city>부산</city>
        <gender>남</gender>
        <name>박정우</name>
        <userId>jay</userId>
    </user>
    <user>
        <city>경기</city>
        <gender>여</gender>
        <name>박소율</name>
        <userId>vega2k</userId>
    </user>
</users>
```

RESTful 웹서비스 컨트롤러 클래스

The screenshot shows a Java code editor window with a single file named "RestfulUserController.java". The code defines a controller class that handles XML requests for user lists.

```
*RestfulUserController.java
@Controller
public class RestfulUserController {
    @Autowired
    private UserService userService;

    @RequestMapping(value="/usersXml",
                    method=RequestMethod.GET)
    @ResponseBody
    public UserVOXML getUserListXml() {
        List<UserVO> list = userService.getUserList();
        UserVOXML xml = new UserVOXML("success", list);
        return xml;
    }
}
```

| Postman을 이용한 컨트롤러 클래스 테스트

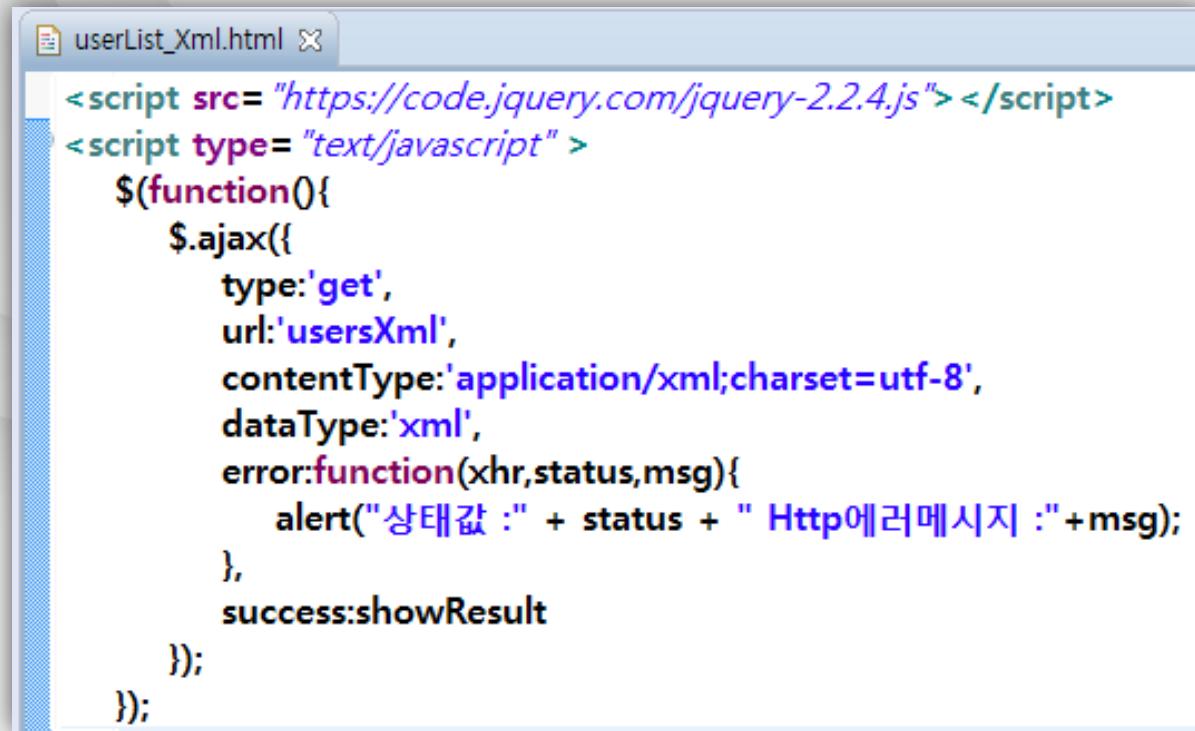
The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/SpringWebPrj/usersXml
- Status:** 200 OK
- Time:** 75 ms

The **Body** tab is selected, displaying the XML response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<users>
    <status>success</status>
    <user>
        <city>제주</city>
        <gender>남</gender>
        <name>둘리</name>
        <userId>dooly</userId>
    </user>
    <user>
        <city>부산</city>
        <gender>남</gender>
        <name>박정우</name>
        <userId>jay</userId>
    </user>
    <user>
        <city>경기</city>
        <gender>여</gender>
        <name>박소울</name>
        <userId>vega2k</userId>
    </user>
</users>
```

| RESTful 웹서비스 클라이언트 html : Ajax 요청



The screenshot shows a browser window with the title "userList_Xml.html". The content of the page is a block of JavaScript code using the jQuery library to make an AJAX request to a service named "usersXml". The code includes error handling and success callback functions.

```
<script src="https://code.jquery.com/jquery-2.2.4.js"></script>
<script type="text/javascript">
$(function(){
    $.ajax({
        type:'get',
        url:'usersXml',
        contentType:'application/xml;charset=utf-8',
        dataType:'xml',
        error:function(xhr,status,msg){
            alert("상태값 :" + status + " Http에러메시지 :" + msg);
        },
        success:showResult
    });
});
```

RESTful 웹서비스 클라이언트 html : Ajax 응답

```
userList_Xml.html
function showResult(xhr) {
    console.log(xhr);
    if($(xhr).find("status").text() == 'success') {
        $(xhr).find("user").each(function(idx,user){
            $('<tr>')
                .append($('<td>').html($(user).find("userId").text()))
                .append($('<td>').html($(user).find("name").text()))
                .append($('<td>').html($(user).find("gender").text()))
                .append($('<td>').html($(user).find("city").text()))
                .appendTo('tbody');
        });
    }
}
//showResult
</script>
```

```
<users>
    <status>success</status>
    <user>
        <city>제주</city>
        <gender>남</gender>
        <name>둘리</name>
        <userId>dooly</userId>
    </user>
    <user>
        <city>부산</city>
        <gender>남</gender>
        <name>박정무</name>
        <userId>jay</userId>
    </user>
    <user>
        <city>경기</city>
        <gender>여</gender>
        <name>박소율</name>
        <userId>vega2k</userId>
    </user>
</users>
```

지금까지 [Spring RESTful 웹서비스 어플리케이션 작성(4)]에 대해서 살펴보았습니다.

XML 응답을 주는 RESTful 웹서비스 개요

- JAXB 개요
- @XmlRootElement, @XmlElement

XML 응답을 주는 RESTful 웹서비스 구현

- 서버 컨트롤러 클래스 작성
- 클라이언트 HTML 작성