KOSA
Korea Software Industry Association

Spring Framework 과정

2021. 9. 1 ~ 3 (총3일, 21시간)

정 준 수 Ph.D

과정 목표

Spring Framework 구성 모듈을 기반으로 Maven 프로젝트로 구성하고 Spring의 핵심 기능인 의존성 주입(Dependency Injection), 제어의 역전(Inversion of Control) 및 웹(MVC) 애플리케이션과 RESTful 웹 서비스 프레임워크의 대한 이해와 실습을 통하여 프로그램 개발 역량을 습득

1. Spring Framework 개요 및 환경 설정
2. Spring 프로젝트 생성과 Maven 설정
3. DI(Dependency Injections)과 IoC(Inversion of Coversion)
4. 웹(MVC), JDBC
5. 실습예제

강의 진행 내용

1.첫째날: Netscape에서 Spring까지

2.둘째날: Software의 재사용(MVC)

3.세째날: RESTful API 구현과 JDBC 연동

(https://www.redhat.com/ko/topics/integration/whats-the-difference-between-soap-rest)

Web 의 동적 데이터 서비스 요구로부터 시작!

CGI Servlet/JSP EJB Spring

본격적으로 Spring 강의에 들어 가기에 앞서 전해 드릴 내용이 있습니다.

1. 사전 설문을 통하여 여러분들의 사전 Java 지식을 확인 하였으며, 이번 Spring 강의는 환경 설정 부분과 연결된 기술이 많아서 첫째날에는 가능한 개별 Framework을 가지고 설명 드리도록 하겠습니다.

2. 모르는 내용 또는 잠시 놓치신 내용이 있으면, Zoom 회의실을 만들어 놓았으니, 쉬는 시간을 이용하여 개별 설명을 전달할 수 있도록 하겠습니다.

프로그램 실습 환경 구축

1. JDK 설치
2. Tomcat 설치
3. Maven 설치
4. Spring Tools 4 for Eclipse 설치

1 일차

Netscape에서 Spring까지

Software란?

https://en.wikipedia.org/wiki/John_Tukey

World Wide Web

The **Web**, is an information system where documents and other web resources are identified by Uniform Resource Locators (URLs, such as `https://example.com/`), which may be interlinked by hyperlinks, and are accessible over the Internet.

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikimedia Shop

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

Print/export

Languages
Simple English
العربية
Bahasa Indonesia
Bahasa Melayu
Български

Welcome to Wikipedia,

the free encyclopedia that anyone can edit.

4,101,617 articles in English

- Arts
- Biography
- Geography
- History
- Mathematics
- Science
- Society
- Technology
- **All portals**

From today's featured article

Windsor Castle is a medieval castle and royal residence in Windsor in the English county of Berkshire, notable for its long association with the British royal family and for its architecture. The original motte-and-bailey castle, built after the Norman invasion by William the Conqueror, was designed to protect Norman dominance around the outskirts of London and to oversee a strategically important part of the River Thames. The castle's lavish early 19th-century State Apartments are architecturally significant, and the 15th-century St George's Chapel is an outstanding example of English Perpendicular Gothic design. Since the time of Henry I it has been used by a succession of monarchs and is the longest-occupied palace in Europe. A popular tourist attraction, it is used as a venue for hosting state visits, and is the Queen's preferred weekend home. It was used as a refuge for the royal family during the Second World War and survived a fire on 20 November 1992. More than five hundred people live and work in Windsor, making it the largest inhabited castle in the world. (**Full article...**)

Recently featured: HMS *New Zealand* – Luke P. Blackburn – *Metroid Prime*

Archive – By email – More featured articles...

Did you know...

In the news

- Israel launches a **major military operation** in the Gaza Strip, as hostilities with Hamas escalate.
- At least 50 schoolchildren are killed in **an accident** at a railway crossing near Manfalut, Egypt.
- Croatian generals **Ante Gotovina** and **Mladen Markač** are acquitted by the International Criminal Tribunal for the former Yugoslavia.
- **Xi Jinping** becomes General Secretary of the Chinese Communist Party and a new Politburo Standing Committee is inaugurated.
- A total solar eclipse **occurs** in parts of Australia and the South Pacific.
- **Justin Welby** is announced as the 105th Archbishop of Canterbury.
- An **earthquake** off the Pacific coast of Guatemala kills dozens of people.

Recent deaths: Bal Thackeray

Syrian civil war – Wikinews – More current events...

On this day...

The 1995 Blog

The Year the Future Began

Windows 95 This way to the INTERNET O.J. SET FREE

Home

FAQs about 1995

About the author

About the book

Read Chapter One

Order 1995, the book

Memorable moment in an exceptional year: The Netscape IPO of 1995

The year 1995 was filled with decisive turns and memorable developments; among them, the "Netscape Moment" of 25 years ago. It was an eye-popping moment in an exceptional year.

On August 9, 1995, Silicon Valley startup Netscape Communications made its stunning market debut and effectively illuminated the emergent World Wide Web for millions of people who, until then, were only vaguely familiar with its potential and promise.

Netscape's initial public offering of shares stimulated the dot.com boom of the late 1990s and enabled what has been called "the technological, social, and financial tone of the Internet age."

Even more flattering was Brian McCullough,

Information

This entry was posted on August 7, 2020 by W. Joseph Campbell in 1990s, Anniversaries, Early Web, Internet, Netscape, Popular culture, Technology, Watershed year, World Wide Web and tagged 1995, Browser war, dot.com boom, Early Web, Hinge moment, Microsoft, Netscape, Newsweek, Recent past, Watershed year.

Shortlink

https://wp.me/p4HZIe-Zd

Navigation

Previous post
Next post

https://1995blog.com/2020/08/07/memorable-moment-in-an-exceptional-year-the-netscape-ipo-of-1995/

CGI

웹서버와 외부 프로그램 사이에서 정보를 주고받는 방법이나 규약

CGI(Common Gateway Interface)

웹 서버는 정적 웹서버와 동적 웹서버로 나뉜다.

정적 웹 서버 :

- HTTP 서버가 있는 컴퓨터로 구성.
- 서버에 존재하는 이미 저장된 파일(HTML, 스크립트 등)을 브라우저에게 전송.
- 서버에 저장된 데이터가 변경되지 않는 한 고정된 웹 페이지를 보게 됨.

동적 웹 서버

- 정적 웹 서버와 어플리케이션 서버(AS)로 구성.
- AS는 웹 서버에서 처리하지 못하는 동적 데이터에 대응하기 위해 만들어진 어플리케이션 서버(데이터베이스 조회, 로직 처리 등)
- 어플리케이션 서버는 프로그램에게 응답을 전달 받아 웹 서버에 전달하게 됨.
- WAS는 웹 서버 + 어플리케이션 서버를 포함하는 개념.(동적 서버 콘텐츠를 수행하는 것으로 정적 콘텐츠를 전달하는 일반적인 웹 서버와는 구별 하는 것 같다.)
- 어플리케이션 서버는 하나의 프로토콜로서 CGI와 유사한 기능을 수행한다고 봐야할 것 같다.

프로세스 :

- WAS가 웹 서버로부터 처리 요청을 받으면, 프로그램의 실행결과를 웹 서버에 전달 → 웹 서버는 해당 결과를 웹 클라이언트에 전송

Tomcat 설치

Tomcat 8 Software Downloads

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

설치 확인:

C:₩Users₩user₩Desktop₩jspstudy₩apache-tomcat-8.5.70₩bin₩startup.bat

Web Application Server

Web Server

- Deliver static content.
- Content is delivered using the HTTP protocol only.
- Serves only web-based applications.
- No support for multi-threading.
- Facilitates web traffic that is not very resource intensive.

Application Server

- Delivers dynamic content.
- Provides business logic to application programs using several protocols (including HTTP).
- Can serve web and enterprise-based applications.
- Uses multi-threading to support multiple requests in parallel.
- Facilitates longer running processes that are very resource-intensive.

Servlet과 JSP

웹브라우저
1.JSP 요청
6. 응답
웹서버
2.전달
웹컨테이너
5. 전달
3. JSP File을
Servlet으로변환
4. 컴파일
JSP.class
JSP.java

Object-oriented programming

https://en.wikipedia.org/wiki/Object-oriented_programming

Java

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

https://en.wikipedia.org/wiki/Java_(programming_language)

Java 프로그램 실습

JDK 설치:

https://www.oracle.com/kr/java/technologies/javase-jdk11-downloads.html

JDK 설치 확인:

```
C:₩Users₩user>java -version
java version "11.0.12" 2021-07-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.12+8-LTS-237)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.12+8-LTS-237, mixed mode)
```

2. Spring 프로젝트 생성과 Maven 설정

Maven은 자바(Java) 기반 프로젝트의 Life Cycle을 관리를 위한 빌드 도구

- 메이븐은 자바용 프로젝트 관리도구
- 메이븐은 컴피일과 빌드를 동시에 수행
- 테스트를 병행하거나 서버 측 배포(Deploy) 자원을 관리할 수 있는 환경 제공
- Library 관리 기능도 내포하고 있음
- pom.xml 파일 관리를 통해 개발, 유지보수 면에서 프로젝트 관리가 용이

메이븐을 사용하는 이유

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

메이븐의 빌드 생명주기(LifeCycle)

Life Cycle을 plugin, phase, goal을 정리하여 이야기하면,

메이븐은 프로젝트 생성에 필요한 단계 (phase)들을

Build Life Cycle이라 정의하고 Build Life Cycle은

default, clean, site 세가지로 표준 정의합니다.

메이븐 프로젝트 생성 입력 내용

- **groupId** - 프로젝트 속하는 그룹 식별 값. 회사, 본부, 또는 단체를 의미하는 값이 오며, 패키지 형식으로 계층을 표현한다. 위에서는 net.madvirus를 groupId로 이용하였다.
- **artifactId** - 프로젝트 결과물의 식별 값. 프로젝트나 모듈을 의미하는 값이 온다. 위에서는 sample을 artifactId로 이용하였다.
- **version** - 결과물의 버전을 입력한다. 위에서는 기본 값인 1.0-SNAPSHOT을 사용하였다.
- **package** - 기본적으로 생성할 패키지를 입력한다. 별도로 입력하지 않을 경우 groupId와 동일한 구조의 패키지를 생성한다.

Maven 프로젝트의 기본 Directory 구성

application-core

- pom.xml
- src
  - main
    - java
      - com.package.dir
    - resources
  - test
    - java
      - com.package.dir
    - resources

기본적으로 생성되는 폴더를 포함한 Maven 프로젝트의 주요 폴더는 다음과 같다.

- src/main/java - 자바 소스 파일이 위치한다.
- src/main/resources - 프로퍼티나 XML 등 리소스 파일이 위치한다. 클래스패스에 포함된다.
- src/main/webapp - 웹 어플리케이션 관련 파일이 위치한다. (WEB-INF 폴더, JSP 파일 등)
- src/test/java - 테스트 자바 소스 파일이 위치한다.
- src/test/resources - 테스트 과정에서 사용되는 리소스 파일이 위치한다. 테스트 시에 사용되는 클래스패스에 포함된다.

기본적으로 생성되지 않은 폴더라 하더라도 직접 생성해주면 된다. 예를 들어 src/main 폴더에 resources 폴더를 생성해주면 Maven은 리소스 폴더로 인식한다.

Maven과 Library 관리 - 자바 버전 수정을 POM 파일 입력

pom.xml 파일을 열어서 아래 코드를 추가하여 이 설정은 컴파일 자바 버전을 1.8로 설정한다.

```
...생략

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>


</project>
```

POM 파일 기본 정의

Maven 프로젝트를 생성하면 pom.xml 파일이 프로젝트 루트 폴더에 생성된다. 이 pom.xml 파일은 Project Object Model 정보를 담고 있는 파일로서, 이 파일에서 다루는 주요 설정 정보는 다음과 같다.

- 프로젝트 정보 - 프로젝트의 이름, 개발자 목록, 라이센스 등의 정보를 기술
- 빌드 설정 - 소스, 리소스, 라이프 사이클 별 실행할 플러그인 등 빌드와 관련된 설정을 기술
- 빌드 환경 - 사용자 환경 별로 달라질 수 있는 프로파일 정보를 기술
- POM 연관 정보 - 의존 프로젝트(모듈), 상위 프로젝트, 포함하고 있는 하위 모듈 등을 기술

POM 파일에서 프로젝트 정보를 기술하는 태그는 다음과 같다.

- <name> - 프로젝트 이름
- <url> - 프로젝트 사이트 URL

POM 연관 정보는 프로젝트간 연관 정보를 기술하는데, 관련 태그는 다음과 같다.

- <groupId> - 프로젝트의 그룹 ID 설정
- <artifactId> - 프로젝트의 Artifact ID 설정
- <version> - 버전 설정
- <packaging> - 패키징 타입 설정. 위 코드의 경우 프로젝트의 결과 Artifact가 jar 파일로 생성됨을 의미한다. jar뿐만 아니라 웹 어플리케이션을 위한 war나 JEE를 위한 ear 등의 패키징 타입이 존재한다.
- <dependencies> - 이 프로젝트에서 의존하는 다른 프로젝트 정보를 기술한다.
  <dependency> - 의존하는 프로젝트 POM 정보를 기술
    - <groupId> - 의존하는 프로젝트의 그룹 ID
    - <artifactId> - 의존하는 프로젝트의 artifact ID
    - <version> - 의존하는 프로젝트의 버전
    - <scope> - 의존하는 범위를 설정

POM 파일의 <dependency> 부분 설정

Maven을 사용하지 않을 경우 개발자들은 코드에서 필요로 하는 라이브러리를 각각 다운로드 받아야 한다. 예를 들어 아파치 commons DBCP 라이브러리를 사용하기 위해서는 DBCP뿐만 아니라 common pool 라이브러리도 다운로드 받아야 한다. 물론 commons logging을 비롯한 라이브러리도 모두 추가로 다운로드 받아 설치해 주어야 한다. 즉 코드에서 필요로 하는 라이브러리뿐만 아니라 그 라이브러리가 필요로 하는 또 다른 라이브러리도 직접 찾아서 설치해 주어야 한다.

하지만, Maven을 사용할 경우에는 코드에서 직접적으로 사용하는 모듈에 대한 의존만 추가해주면 된다. 예를 들어 commons-dbcp 모듈을 사용하고 싶은 경우 다음과 같은 <dependency> 코드만 추가해주면 된다.

```
<dependency>
    <groupId>commons-dbcp</groupId>
    <artifactId>commons-dbcp</artifactId>
    <version>1.2.1</version>
</dependency>
```

그러면, Maven은 commons-dbcp 뿐만 아니라 commons-dbcp가 의존하는 라이브러리를 자동으로 처리해준다.

Maven 실습

https://maven.apache.org/download.cgi

http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

>java -cp target₩my-app-1.0-SNAPSHOT.jar com.mycompany.app.App

>mvn help:describe -Dcmd=compile //plugin:goal 확인

Java Projects with Maven 실습

https://spring.io/guides/gs/maven/

1. Spring Framework 개요 및 환경 설정

Spring Framework란?

Java 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 어플리케이션 프레임워크

- **자바(JAVA) 플랫폼을 위한 오픈소스(Open Source) 어플리케이션 프레임워크**
- **자바 엔터프라이즈 개발을 편하게 해주는 오픈 소스 경량급 애플리케이션 프레임워크**
- **자바 개발을 위한 프레임워크로 종속 객체를 생성해주고, 조립해주는 도구**
- **POJO(Plain Old Java Object) BEAN CONTAINNER**

Spring Framework의 특징

- 경량의 컨테이너로서 자바 객체를 직접 관리
- POJO(Plain Old Java Object) 방식의 프레임워크
  (POJO는 자바 모델이나, 기능, 프레임워크 등에 따르지 않고 홀로 독립적이며
  단순한 기능만을 가진 객체를 의미, 자바에서는 이러한 객체를 Bean이라 부른다)
- IoC(Inversion of Control) 지원
- DI(Dependency Injection) 지원
- AOP(Aspect-Oriented Programming) 지원
- iBATIS, myBATIS, Hibernate 등의 데이터베이스 라이브러리 지원

Spring Framework의 사용 장점

- 스프링 프레임워크를 적용하면 자바 코드 작성 부분을 줄일 수 있는 효과를 얻음
- 반면, 초기 프로젝트 셋팅이 다소 복잡하므로 충분한 연습이 필요함
- 반복되는 작업을 줄일 수 있어 기능 개발에 집중할 수 있음
- 프로젝트 관리가 용이함
- 다수의 개발자와 동시에 프로젝트 개발이 용이함

Spring 프레임워크를 구성하는 기능 요소

Maven Spring Boot Run

https://devwithus.com/mvn-spring-boot-run/

Spring Quickstart Guide

https://spring.io/quickstart

Spring Tools 4 for Eclipse

https://spring.io/tools

https://spring.io/guides/gs/sts/

spring

Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.

https://spring.io/projects

What is Spring Bean ?

https://devwithus.com/what-is-spring-bean/

프로그램 결합 강도

강한 결합

A객체가 B/C객체를 직접 생성한다.

약한 결합

B/C객체가 외부에 생성되어 A객체에 주입된다.

Dynamic web page

A server-side dynamic web page is a web page whose construction is controlled by an application server processing server-side scripts.

https://en.wikipedia.org/wiki/Dynamic_web_page

Dynamic web page 실습

2 일차

Software Resue
(SW의 재사용)

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.

AOP(Aspect Oriented Programming)

❖관심의 분리 (Seperatrion 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.

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 정리**한 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-Funtional) 요구사항(성능, 보안, 확장성, 안정성 등)을 만족하는 구조와 구현된 기능을 안정적으로 실행하도록 제어해주는 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.

Software design pattern

https://en.wikipedia.org/wiki/Software_design_pattern

디자인패턴과 프레임워크의 관련성

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

- **디자인 패턴**은 애플리케이션을 설계할 때 필요한 **구조적인 가이드라인**이 되어 줄 수는 있지만 구체적으로 **구현된 기반코드를 제공하지 않는다.**

- **프레임워크**는 디자인 패턴과 함께 패턴이 적용 된 **기반 클래스 라이브러리를 제공**해서 프레임워크를 사용하는 **구조적인 틀과 구현코드를 함께 제공한다.**

디자인패턴과 프레임워크의 관련성

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

IoC (Inversion of Control)

IoC란 "제어의 역전" 즉, 인스턴스 생성부터 소멸까지의 인스턴스 생명주기 관리를 개발자가 아닌 컨테이너가 대신 해준다는 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.

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

❖ 아키텍처 결정 = 사용하는 프레임워크의 종류 + 사용전략

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

Model–view–controller

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

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 logic과 Presentation logic을 분리**하기 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

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

MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER

4. 웹 MVC

- MVC는 기본적으로 Model View Controller의 약자로 프로그램 개발 시 세가지 역할로 구분하여 개발하는 방법론
- 이렇게 하는 이유는 완벽한 분업화를 통해 해당 역할 개발자가 자신의 역할에만 집중하여 개발 하기 위함

MVC(Model-View-Controller) 패턴의 개념

MVC 패턴을 사용하면, 사용자 인터페이스로부터 비즈니스 로직을 분리하여 어플리케이션의 시가적 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 어플리케이션 개발이 가능

- **Model** : 어플리케이션의 정보(데이터, Business Logic 포함)
- **View** : 사용자에게 제공할 화면(Presentation Logic)
- **Controller** : Model과 View 사이의 상호 작용을 관리

모델2 아키텍쳐 개념

1. 모델 1 아키텍쳐 : Controlloer 역할을 JSP가 담당함
2. 모델 2 아키텍쳐 : Controlloer 역할을 Servlet이 담당함

모델2 아키텍쳐 호출 순서

① 웹 브라우저가 웹 어플리케이션 실행 요청하면, 웹 서버가 드 요청을 받아서 서블릿 컨테이너 (ex: 톰캣서버)에 넘겨준다.
서블릿 컨테이너는 URL을 확인하고 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 인지 값으로 넘긴다.

모델2 아키텍쳐 호출 순서

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

모델2 아키텍쳐 호출 순서

④ 서블릿은 모델 객체로 부터 반환 받은 값을 JSP에 전달한다.

모델2 아키텍쳐 호출 순서

⑤ *JSP*는 서블릿으로 부터 전달받은 값 객체를 참조하여 웹 브라우저가 출력할 결과 화면을 만들고, 웹 브라우저에 출력함으로써 요청 처리를 완료한다.

모델2 아키텍쳐 호출 순서

⑥ 웹브라우저는 서버로부터 받은 응답 내용을 화면에 출력한다.

DispatcherServlet 클레스

① Front Controller 패턴 적용

② Web.xml에 설정

③ 클라이언트로부터
모든 요청을 전달 받음

④ Controller나 View와 같은
Spring MVC의 구성요소를
이용하여 클라이언트에게
서비스를 제공

Spring MVC의 주요 구성 요소

구성요소	설 명
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 매핑을 가질 수 있음
Controller	클라이언트의 요청을 처리한 뒤, Model를 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 처리 과정

Spring MVC의 주요 구성 요소의 처리 과정

① 클라이언트의 요청을 받은 DispatcherServlet를 web,xml 에 설정

② 클라이언트의 요청을 처리할 Controller를 작성

③ Spring Bean으로 Controller 를 등록

④ JSP를 이용한 View 영역의 코드를 작성

⑤ Browser 상에서 JSP를 실행

.

Spring MVC기반 웹 어플리케이션 작성 절차

① 클라이언트의 요청이 DispatcherServlet에게 전달한다.

② DispatcherServlet은 HandlerMapping을 사용하여 클라이언트의 요청을 처리할 Controller를 확득한다.

③ DispatcherServlet은 Controller 객체를 이용하여 클라이언트의 요청을 처리한다.

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

⑤ DispatcherServlet은 ViewResolver로부터 응답 결과를 생성할 View 객체를 구한다.

⑥ View는 클라이언트에게 전송할 응답을 생성한다.

.

•**JAVA** 파일 **:** 자바 파일들이 위치해 있는 디렉터리**.** 자바로 작성된 **Controller, Service, DAO** 파일들이 위치해있다**.**

•**webapp :** 웹과 관련된 파일들이 위치해있는 디렉터리**. html, css, js, jsp** 파일이 위치해있다**.** 또한 웹 어플리케이션 구동에 필요한 **XML** 설정파일들이 위치해있다**.**

•**webapp**의 **resources : JSP**파일을 제외한 **html, css, js** 파일이 위치해있다**.**

•**spring** 폴더 **:** 스프링 컨테이너를 생성하기 위한 스프링 설정파일이 위치해있다**.**

•**views** 폴더 **: View**로 사용될 **JSP**파일이 위치해있다**.**

•**pom.xml : Maven** 설정파일이다**. Maven**은 프로젝트 빌드 도구 및 관리 도구로서 **pom.xml**을 통해 설정할 수 있다**.**

프로젝트 구조는 **Model2**를 적용한 웹 어플리케이션을 만들 수 있도록 구성되있다

DispatcherServlet 설정

DispatcherServlet은 web.xml 설정 파일에서 관련 정보를 명시한 후 스프링이 구동될 때 구현화 된다. 프로젝트를 처음 생성할 때 web.xml에 명시된 DispatcherServlet 설정 부분은 다음과 같다.

web.xml

```
<servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
</servlet-mapping>
```

<servlet> 태그는 **DispatcherServlet**을 구현하기 위해 어떤 클래스를 이용해야 할지와 초기 파라미터 정보를 포함하고 있다.

•<servlet-name> : 해당 서블렛의 이름을 지정하면 이 지정된 이름을 가지고 다른 설정파일에서 해당 서블릿 정보를 참조한다.
•<servlet-class> : 어떤 클래스를 가지고 DispatcherServlet을 구현할 것인지를 명시하고 있다.
•<init-param> : 초기화 파라미터에 대한 정보. servlet에 대한 설정 정보가 여기에 들어간다. 만약 초기화 파라미터에 대한 정보를 기술하지 않을 경우 스프링이 자동적으로 appServlet-context.xml 을 이용하여 스프링 컨테이너를 생성한다.
•<load-on-startup> : 서블릿이 로딩될 때 로딩 순서를 결정하는 값. 톰캣이 구동되고 서블릿이 로딩되기 전 해당 서블릿에 요청이 들어오면 서블릿이 구동되기 전까지 기다려야한다. 이 중 우선순위가 높은 서블릿부터 구동할 때 쓰이는 값이다.
<servlet-mapping>에서는 서블렛이 **<url-pattern>**에서 지정한 패턴으로 클라이언트 요청이 들어오면 해당 **<servlet-name>**을 가진 **servlet**에게 이 요청을 토스하는 정보를 기술한다.

Spring MVC 실습

https://spring.io/guides/gs/serving-web-content/

- gs-serving-web-content

- https://devwithus.com/mvn-spring-boot-run/

Spring Bean & IoC

In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC *container* are called *beans*. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application. Beans, and the *dependencies* among them, are reflected in the *configuration metadata* used by a container.

Spring - Registering beans in component classes

1 Registering a new bean definition. The method will not be CGLIB proxied like @Configuration classes.

```
@Configuration
@ComponentScan("com.example")
public class MyConfig{
   .......
}
```

Scan

```
package com.example;

@Component
public class TestBean {

@Bean
public AnotherBean anotherBean(){
  return new AnotherBean();
}
.......
}
```

3 We can even use @Configuration instead of @Component because @Configuration is annotated with @Component itself. Any class annotated with @Configuration will be CGLIB proxied.

4 Calling this method directly will give a new instance every time. But if enclosing class is annotated with @Configuration, same instance will be returned every time because of CGLIB enhancement.

2 We can even use different package here because this bean is not scanned but a direct instance is created by TestBean.

```
package com.example
public class AnotherBean {
 ......
}
```

LogicBig.com

3. DI(Dependency Injections)과 IoC(Inversion of Coversion)

의존성 주입 (Dependency Injection)

객체 자체가 아니라 Framework에 의해 객체의 의존성이 주입되는 설계 패턴, 즉 각 클레스간의 의존관계를 Bean 설정 정보를 바탕으로 컨테이너가 자동으로 연결

- Framework에 의해 동적으로 주입되므로 여러 객체 간의 결합이 줄어든다.
- Dependency Injection은 Spring Framework에서 지원하는 IoC의 형태

프로그램 결합 강도

강한 결합

A객체가 B/C객체를 직접 생성한다.

약한 결합

B/C객체가 외부에 생성되어 A객체에 주입된다.

설정에 명시된대로 Container가

- bean 객체를 생성하고
- 종속성 주입을 수행한다

Dependency Injection(의존성 주입)과 Inversion Of Control(제어의 역전)은 같은 의미로 사용된다.

- IoC는 DI를 통해 달성된다.

IoC(제어의 역전): 프로그램 제어권을 framework가 가져가는 것

- 개발자가 모든 제어의 중심이지만 코드 전체에 대한 제어는 framework가 한다.
- 개발자가 설정(xml, annotation 등)만 하면 Container가 알아서 처리한다.
- 즉, 우리는 Framework 속에서 프로그래밍을 하는 것

의존성 주입의 세가지 유형

1. Contructor Injection – 생성자를 이용한 의존성 주입

    - 생성자를 통한 전달
    - <constructor-arg ref="cat"></constructor-arg>

2. Setter Injection – Setter 메서드를 이용한 의존성 주입

    - setter()을 통한 전달
    - <property name="myName" value="poodle"></property>

3. Method Injection – 일반 메서드를 이용한 의존성 주입

    - 멤버 변수를 통한 전달

Spring DI Container

Spring Framework의 핵심 컴포넌트

- Container는 DI를 사용하여 응용 프로그램을 구성하는 bean 객체를 관리한다.

역할

bean을 포함하고 관리하는 책임이 있다.

1. 객체(bean)를 생성하고
2. 객체들을 함께 묶고
3. 객체들을 구성하고
4. 객체들의 전체 수명주기(lifecycle)를 관리

Spring DI Container

설정 방법

- Spring Container metadata 설정 방법 (세 가지)

1. XML
    1. 빈 객체 정의 (Bean Definition)
    2. 의존성 주입 (Dependency Injection)
2. Java Annotations
3. Java Code

IoC Container(Spring DI Container)의 종류

BeanFactory

- 클레스를 통해 객체를 생성하고 이를 전달함
- 상속 등 객체 간의 관계를 형성하고 관리함
- Bean에 관련된 설정을 위한 xml 파일은 즉시 로딩하지만 객체는 개발자가 요구할 때 생성함
- XmlBeanFactory

ApplicationContext

- 클레스를 통해 객체를 생성하고 이를 전달함
- 상속 등 객체 간의 관계를 형성하고 관리함
- 국제화 지원 등 문자열에 관련된 다양한 기능을 제공
- 리스너로 등록되어 있는 Bean에 이벤트를 발생시킬 수 있음
- Bean에 관련된 서정을 위한 xml 파일은 즉시 로딩하면서 객체를 미리 생성해서 가지고 있음
- ClassPathXmlApplicationContext
- FileSystemXmlApplicationContext
- XmlWebApplicationContext
- AnnotationConfigApplicationContext

Spring DI 용어 정리

빈 (Bean)	▪ 스프링이 IoC 방식으로 관리하는 객체라는 뜻 ▪ 스프링이 직접 생성과 제어를 담당하는 객체를 Bean이라고 부름
빈 팩토리 (BeanFactory)	▪ 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 (Application Context)	▪ 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)	▪ ApplicationContext 또는 BeanFactory가 IoC를 적용하기 위해 사용하는 메타정보를 말함 ▪ 설정 메타정보는 IoC컨테이너에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용됨

스프링 어노테이션

Annotation	Package
@Service	import org.springframework.stereotype.Service;
@Repository	import org.springframework.stereotype.Repository;
@Component	import org.springframework.stereotype.Component;
@Autowired	import org.springframework.beans.factory.annotation.Autowired;
@Transactional	import org.springframework.transaction.annotation.Transactional;
@Scope	import org.springframework.context.annotation.Scope;

Spring MVC Annotations

@Controller	import org.springframework.stereotype.Controller;
@RequestMapping	import org.springframework.web.bind.annotation.RequestMapping;
@PathVariable	import org.springframework.web.bind.annotation.PathVariable;
@RequestParam	import org.springframework.web.bind.annotation.RequestParam;
@ModelAttribute	import org.springframework.web.bind.annotation.ModelAttribute;
@SessionAttributes	import org.springframework.web.bind.annotation.SessionAttributes;

Spring Security Annotations

@PreAuthorize	import org.springframework.security.access.prepost.PreAuthorize;

@Autowired
@Resource
@PostConstruc
@PreDestroy
@Qualifier
@Required
@Bean
@Component
@Service
@Repository

Dependency Injection 실습

강사 소개

정 준 수 / Ph.D (heinem@naver.com)

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 강의
- 現) 한국소프트웨어산업협회, AI, 머신러닝 강의
- 現) 서울디지털재단, AI 자문위원
- 現) 한성대학교 교수(겸)
- 전문분야: Computer Vision, 머신러닝(ML), RPA
- https://github.com/JSJeong-me/