



# Spring

[eygou@induk.ac.kr](mailto:eygou@induk.ac.kr)

[lms.induk.ac.kr](https://lms.induk.ac.kr)

# 학습 개요

## ■ 등장 배경

- 개발자들에게 개발 시간은 단축시키고, 애플리케이션의 복잡도는 낮추며, 성능은 향상시킬 수 있는 강력한 API 집합을 제공에 대한 수요가 증대하고 있다.

## ■ 학습 목표

- Spring Framework의 등장 배경, 특징 및 장단점에 대하여 학습한다.

# 계속

## ■ 학습 목차

- 웹 애플리케이션 개발 기술의 발전
- Spring 등장 배경
- Spring 특징
- Spring Framework 발전

# 웹 애플리케이션 개발 기술의 발전

## ■ MVC Model 또는 MVP 패턴

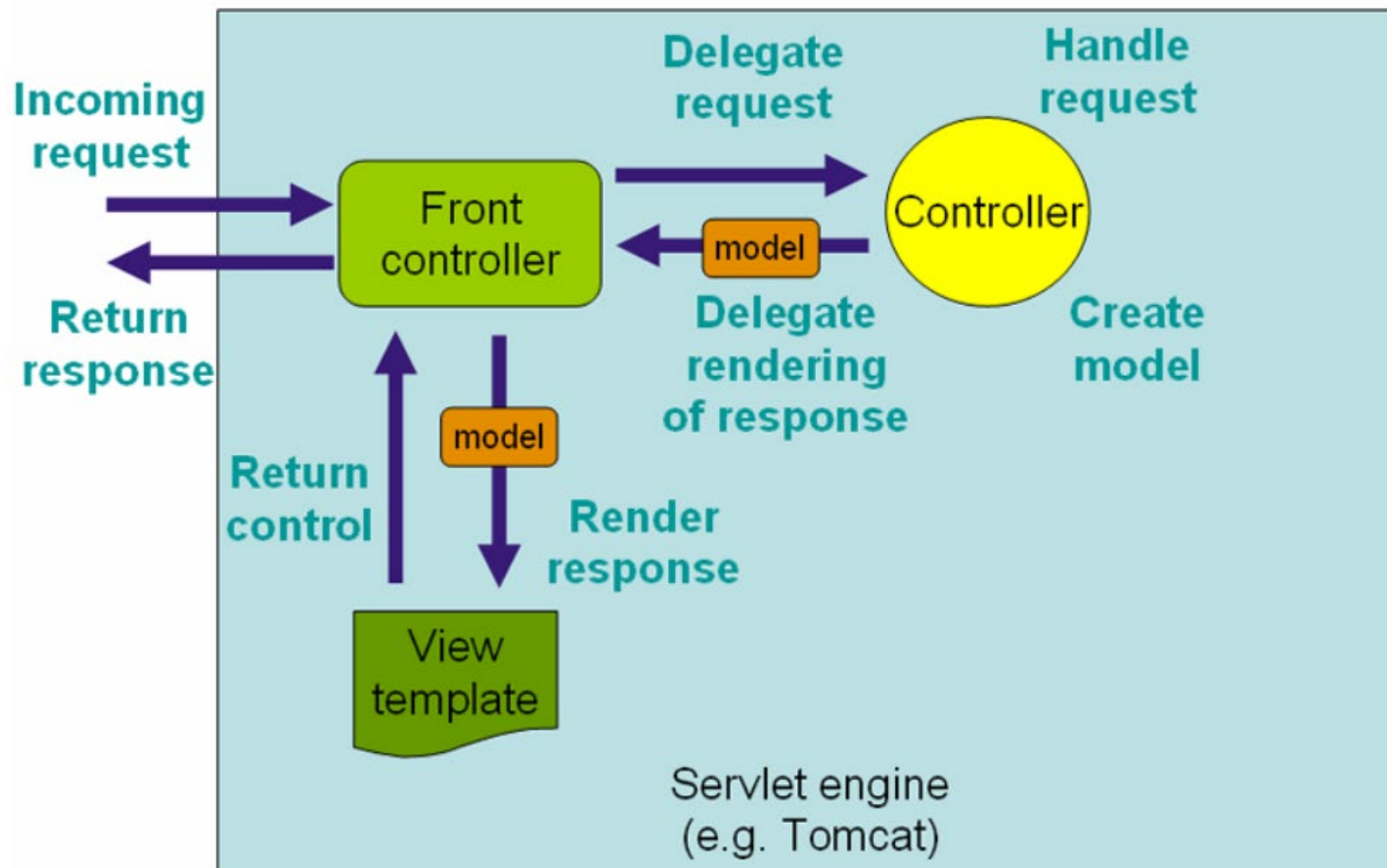
- MVC Model 1
  - 컨트롤러와 뷰가 일체형, 모델은 Beans 으로 구현
- MVC Model 2
  - 컨트롤러와 뷰가 분리, 모델은 Beans 으로 구현

## ■ Front Controller 모델

- MVC Model 2에서 Controller에 중복 코드 문제를 해결하고자 프론트 컨트롤러를 두고 모든 흐름을 제어하도록 함. 프론트 컨트롤러가 컨트롤러를 호출
- 각 컨트롤러는 결과값 반환을 프론트 컨트롤러에게 위임함.

# 계속

Figure 22.1. The request processing workflow in Spring Web MVC (high level)



# 계속

## ■ Java Framework 기반 개발

- J2EE 또는 JavaEE
  - EJB(Enterprise Java Beans)
- Spring
  - POJO(Plain Old Java Object)

## ■ Java Framework 종류

- **Spring** vs eGovFrame vs. JakartaEE
- Hibernate vs. MyBatis
- JSF, GWT, Vaddin ...

# 계속

- 일반적인 Framework 정의
  - 응용 소프트웨어의 표준 구조를 구현하기 위해 소프트웨어 개발자가 사용하는 소프트웨어 프레임워크들로 구성된다.
  - 재사용할 수 있는 수 많은 코드들을 통합하여 응용 소프트웨어를 위한 표준 코드를 작성하여 개발을 용이하게 한다.
  - 개발자의 수준 차이로 인하여 발생하는 품질 문제를 일정 수준으로 제공할 수 있다.
  - 자바에서는 클래스와 인터페이스들의 집합체이다.

# MVC 모델

## ■ 발전사

- Smalltalk-80 사용자 인터페이스의 중심 개념으로 MVC 패러다임이라고 하였음
- 1990년대초 마이크로소프트사는 모델-뷰-프리젠퍼 패턴을 사용하여 개발하였음
- 2002년 W3C에서 WWW를 위한 아키텍처로 채택하였음

## ■ 정의

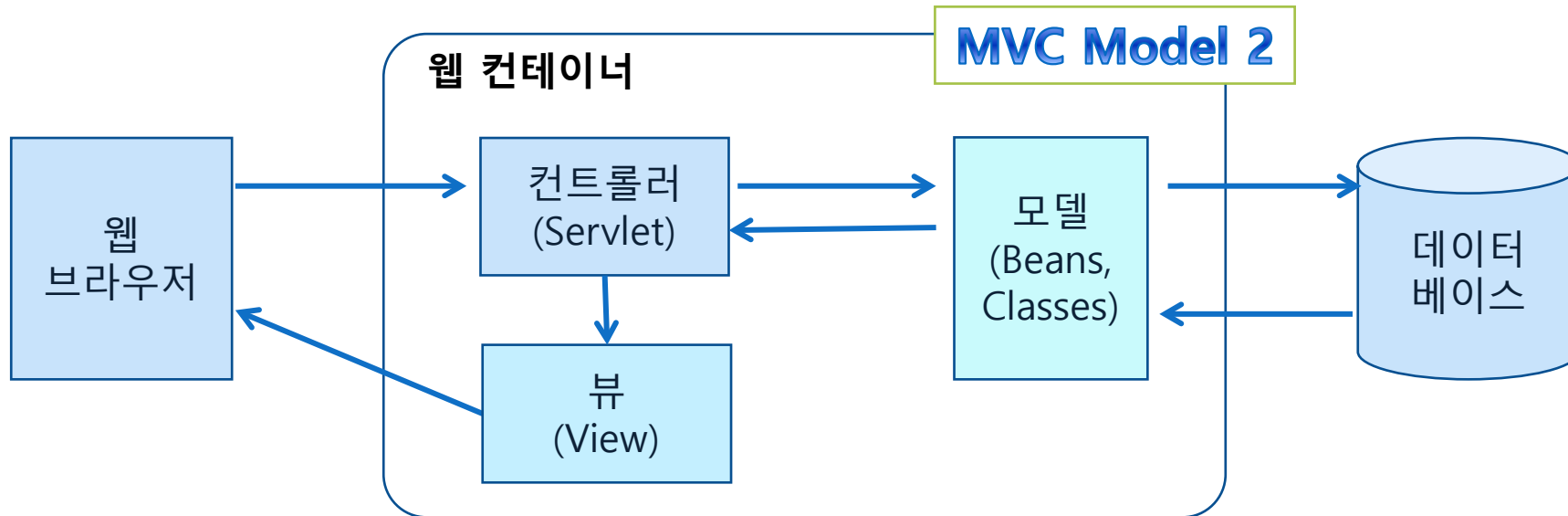
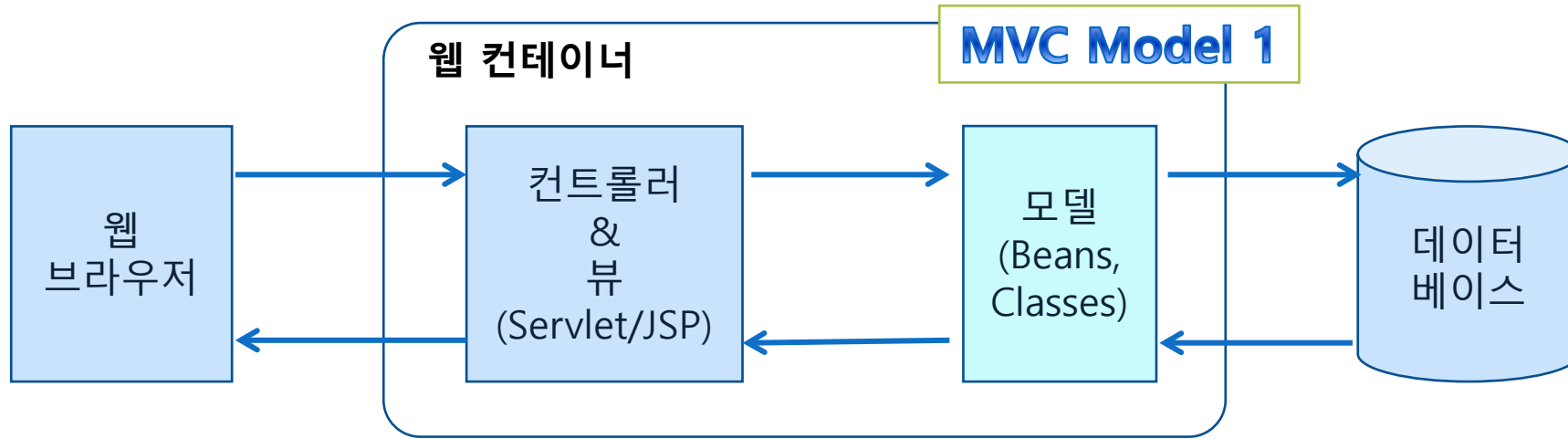
- 사용자 인터페이스로부터 비즈니스 로직을 분리하여 상호 영향 없이 수정이 가능하도록 하는 소프트웨어 아키텍처 패턴을 의미함

## ■ 구성

- MVC 모델을 구현하는 애플리케이션은 모델, 뷰, 컨트롤러 라는 모듈로 구성된다.



# 계속



# Java Enterprise Edition

- 자바 기반 엔터프라이즈 애플리케이션 개발 플랫폼
  - J2EE -> Java EE -> Jakarta EE
  - 5.0 버전 이전 : J2EE 1.2, 1.3, 1.4
  - 5.0 버전 ~ 8.0 버전 : Java EE 5, 6, 7, 8(August 31, 2017)
  - Jakarta EE 8 ~ 9
    - [https://en.wikipedia.org/wiki/Java\\_Platform,\\_Enterprise\\_Edition](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition)

# 계속

## ■ 발전사

- J2EE 1.2 (December 12, 1999)
- J2EE 1.3 (September 24, 2001)
- J2EE 1.4 (November 11, 2003)
- Java EE 5 (May 11, 2006)
- Java EE 6 (December 10, 2009)
- Java EE 7 (May 28, 2013, but April 5, 2013 according to spec document. June 12, 2013 was the planned kickoff date)
- Java EE 8 (August 31, 2017) Eclipse Foundation
- Jakarta EE 8 (September 10, 2019) - Java EE 8와 완전한 호환
- Jakarta EE 9 (November 22 2020)- javax.\* 에서 jakarta.\* namespace 변경

# Java EE와 EJB

## ■ EJB(Enterprise Java Beans)

- 분산 환경을 위한 컴포넌트
- EJB 컨테이너상에서 분산 처리와 분산 트랜잭션 처리를 담당하는 융합 컴포넌트
  - 분산된 EJB 컴포넌트를 동일한 기계에 존재하는 컴포넌트처럼 액세스할 수 있게 함
  - 분산된 데이터베이스의 트랜잭션을 하나의 데이터베이스를 사용한 것처럼 제어할 수 있게 함

Specification	Java EE 6	Java EE 7	Java EE 8
Servlet	3.0	3.1	4.0
JavaServer Pages ( <i>JSP</i> )	2.2	2.3	2.3
Unified Expression Language ( <i>EL</i> )	2.2	3.0	3.0
Debugging Support for Other Languages (JSR-45)	1.0	1.0	1.0
JavaServer Pages Standard Tag Library ( <i>JSTL</i> )	1.2	1.2	1.2
JavaServer Faces ( <i>JSF</i> )	2.0	2.2	2.3
Java API for RESTful Web Services ( <i>JAX-RS</i> )	1.1	2.0	2.1
Java API for WebSocket ( <i>WebSocket</i> )	n/a	1.0	1.1
Java API for JSON Processing ( <i>JSON-P</i> )	n/a	1.0	1.1
Common Annotations for the Java Platform (JSR-250)	1.1	1.2	1.3
Enterprise JavaBeans ( <i>EJB</i> )	3.1 Lite	3.2 Lite	3.2
Java Transaction API ( <i>JTA</i> )	1.1	1.2	1.2
Java Persistence API ( <i>JPA</i> )	2.0	2.1	2.2
Bean Validation	1.0	1.1	2.0
Managed Beans	1.0	1.0	1.0
Interceptors	1.1	1.2	1.2
Contexts and Dependency Injection for the Java EE Platform	1.0	1.1	2.0
Dependency Injection for Java	1.0	1.0	1.0

# 계속

## ■ EJB 컨테이너

- Servlet/JSP 컨테이너 기능에 EJB 실행 환경까지 제공하는 플랫폼을 의미한다.
- EJB(Enterprise Java Beans)는 비즈니스 로직을 구현하는 서버 측 컴포넌트를 의미한다.
- 분산 처리, 분산 트랜잭션을 지원, 보안을 지원, 재사용 가능한 컴포넌트 생성 지원 등

# 계속

## ■ J2EE 서버

- 높은 보안성과 신뢰성을 가져야 하는 기업용 애플리케이션을 개발하기 위해 필수적인 기술들을 집대성해놓은 플랫폼을 의미한다.
- 개발자들에게 개발 시간은 단축시키고, 애플리케이션의 복잡도는 낮추며, 성능은 향상시킬 수 있는 강력한 API 집합을 제공하는 것을 목표로 한다.
- Servlet, JSP, EJB 뿐 아니라 JMS, JNDI, JTA, Web services(XML, SOAP ...) 등 다양한 사양서(specification)의 내용을 지원한다.

# 계속

- EJB 단점
  - 단위 테스트가 어렵다.
  - 불필요한 메서드를 구현해야 한다.
  - 예외 처리가 번거롭다.
  - 배포가 불편하다.



# Spring

## ■ 등장 배경

- 자바 기반 애플리케이션 개발 과정에서 프레임워크 (자료구조, 알고리즘, 생명주기 관리 등)의 필요성이 증대
- 중량의 JavaEE 컨테이너를 대신할 경량 프레임워크에 대한 수요 증대


# 계속

## ■정의

- 자바 플랫폼 기반 오픈 소스 애플리케이션 프레임워크, IOC(Inversion of Control) 컨테이너
- 기업용(대형) 애플리케이션을 개발에 적합한 **경량**의 프레임워크
  - 웹 애플리케이션 개발에 종속되어 있지 않음
  - 오늘날 대규모 기업용 어플리케이션은 대부분이 웹 애플리케이션임

Spring Framework

spring.io/projects/spring-framework



Why Spring ▾Learn ▾Projects ▾TrainingSupportCommunity ▾⚙️

Spring Boot

Spring Framework

Spring Data >

Spring Cloud >

Spring Cloud Data Flow

Spring Security >

Spring for GraphQL

Spring Session >

Spring Integration

Spring HATEOAS

Spring REST Docs

Spring Batch

Spring AMQP

Spring CredHub

Spring Flo

Spring for Apache Kafka

Spring LDAP

Spring Shell



Spring Statemachine

Spring Vault

Spring Web Flow

# Spring Framework

5.3.16



OVERVIEWLEARNSUPPORT

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.

## Support Policy and Migration

For information about minimum requirements, guidance on upgrading from earlier versions and support policies, please check out [the official Spring Framework wiki page](#)

## Features

- [Core technologies](#): dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP.
- [Testing](#): mock objects, TestContext framework, Spring MVC Test, `WebTestClient`.
- [Data Access](#): transactions, DAO support, JDBC, ORM, Marshalling XML.
- [Spring MVC](#) and [Spring WebFlux](#) web frameworks.
- [Integration](#): remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.
- [Languages](#): Kotlin, Groovy, dynamic languages.

# 계속

## ■ 특징

- 프레임워크가 자바 객체의 생명 주기를 직접 관리
- **POJO 방식의 프레임워크**
  - Java EE의 EJB 를 사용하면서 해당 플랫폼에 종속되어 있는 무거운 객체들을 에 대응되는 개념으로 별도의 프레임워크 없이 Java EE를 사용할 때에 비해 특정 인터페이스를 직접 구현하거나 상속받을 필요가 없어 기존 라이브러리를 지원하기가 용이하고, 객체가 가볍다.
  - EJB의 경우 JavaEE 컨테이너에 종속되고 생성, 사용 및 테스트가 어려움
- DI : 프레임워크가 오브젝트를 호출하고, 의존성이 필요한 경우 프레임워크가 연결
- 핵심적인 비즈니스 로직과 관련이 없고, 여러 곳에서 공통적으로 사용되는 기능을 분리하여 실행 시 조합하는 방식인 AOP를 지원
- 영속성 관련된 다양한 서비스(Hibernate, JPA, MyBatis등) 지원하고 높은 확장성 지원

# Spring 주요 용어와 개념

- **IOC (Inversion of Control, 제어 역전)**
  - 제어권이 프레임워크에 있음
- **DI(Dependency Injection)**
  - 구성 요소간의 결합도를 낮추고, 프레임워크가 결합을 제어
- **AOP(Aspect Oriented Programming, 관점 지향 프로그래밍)**
  - 로깅, 트랜잭션, 보안 등 여러 모듈에서 공통적으로 사용하는 기능을 분리하여 관리

# 오브젝트 제어와 IOC

## ■ 기존 방법

- 사용하는 오브젝트가 사용할 오브젝트를 제어(생성 및 호출) 하는 구조였음

## ■ IOC (Inversion of Control)

- 사용하는 오브젝트가 사용할 오브젝트를 제어하지 않고, 제어 권한을 갖는 오브젝트에서 위임하는 구조
  - 제어 권한을 갖는 오브젝트 : 초기 특정 오브젝트가 담당하였으나 현재 컨테이너 또는 프레임워크가 이를 담당하므로 주로 컨테이너나 프레임워크를 의미함

# 계속

## ■ IOC 종류

- DL (Dependency Lookup, 의존성 검색)
  - 저장소에 저장되어 있는 Bean들에 접근하기 위하여 개발자가 컨테이너에서 제공하는 API를 이용하여 사용하고자 하는 Bean을 Lookup 하는 것
  - 컨테이너 종속성이 증가하는 단점이 있음

# 계속

- **DI (Dependency Injection, 의존성 주입)**
  - 각 계층 간, 각 클래스 간에 필요로 하는 의존 관계를 컨테이너가 자동으로 연결해주는 것
  - Setter Injection
    - 필요한 값 할당전까지 객체를 사용할 수 없음
  - Constructor Injection
    - 생성 순서를 지켜야 함



# AOP(Aspect Oriented Programming)

## ■배경

- 소프트웨어 개발할 때 마다 부딪치게 되는 공통적인 문제들이 존재함
  - 로깅, 보안, 인증, 트랜잭션, 리소스 풀링(resource pooling), 에러 검사, 정책 적용, 멀티쓰레드 안전 관리, 데이터 영속성(data persistency)
- 문제 영역(problem domain)을 구분하여 처리함
  - 핵심 관심(core concern) : OOP, Class, Component, Service
  - 횡단 관심(cross-cutting concern) : AOP, Aspect

# 계속

## ■ 정의

- 소프트웨어를 개발하는 과정에서 관심의 분리(separation of concern)를 통해 문제 영역을 핵심 관심과 횡단 관심의 독립적인 모듈로 분해하는 프로그래밍 패러다임, 관점지향 프로그래밍이라고 함
- 업무 기능과 시스템 공통 기능으로 분리하여 코드를 작성

## ■ 장점

- 업무 로직을 쉽게 이해할 수 있기 때문에 생산성이 향상됨
- 핵심 관심과 횡단 관심 간의 결합성이 낮아짐으로 업무 코드의 재사용성이 증대되고, 확장이 용이해짐

# Spring 버전 연혁

## ▪ 최초 버전(first version)

- 2002년 10월 "Expert One-on-One J2EE Design and Development" 책을 통해 소개된 Rod Johnson이 작성한 프레임워크

## ▪ 1.x.x (2003.06 )

- Spring 1.2.6 framework won a Jolt productivity award and a JAX (Java API for XML) Innovation Award

## ▪ 2.x.x (2006.10 )

- IOC 컨테이너 기능 향상, AOP 기능 향상, Java SE 5지원

# 계속

## ■ 3.x.x (2009.12 ~ 2016.12)

- Java SE 5/6/7 지원, SpEL 도입, Rest API 추가

## ■ 4.x.x (2013.12 ~ 2020.12)

- Java SE 6/7/8 지원, Java EE 6/7
- Core Container, Data Access, Caching, JMS, Web, WebSocket messaging, Testing 향상

# 계속

## ■ 5.x.x (2017.09 ~)

- Java SE 8과 Java EE 7 이상 필요
- Java SE 9 이상 지원, JUnit 5 지원
- Reactive Stream API를 지원하는 Reactor를 사용하는 non-blocking 웹 프레임워크인 Spring WebFlux 도입
- SpringOne = Spring 5.x 와 SpringBoot 2.x
- 현재 5.1.14GA 와 **5.2.4 GA**
  - GA(General Availability)는 테스트가 완료된 정식 릴리즈 버전, 안정적으로 운영되어야 하는 프로젝트에서 사용
- 참고
  - <https://github.com/spring-projects/spring-framework/wiki/What%27s-New-in-Spring-Framework-5.x>

# 계속

- 지원 계획
  - 5.2.x
    - 최신의 프로덕션 라인, GA : Sep 2019
  - 5.1.x
    - GA : September 2018, the end of 2020
  - 5.0.x
    - EOL phase : Q2 2019, active maintenance : January 2020
    - security patches : the end of 2020
  - 4.3.x
    - 4세대 마지막 브랜치
    - extended maintenance, security patches : January 2020
    - *official EOL (end-of-life) : December 31st, 2020*
  - 3.2.x *i*
    - *EOL (end-of-life) : December 31st, 2016*

# 계속

## ▪ JDK 지원 계획

- Spring Framework 5.3.x: JDK 8-17 (expected)
- Spring Framework 5.2.x: JDK 8-15 (expected)
- Spring Framework 5.1.x: JDK 8-12
- Spring Framework 5.0.x: JDK 8-10
- Spring Framework 4.3.x: JDK 6-8

[Spring Boot](#)
[Spring Framework](#)
[Spring Data](#)
[Spring Cloud](#)
[Spring Cloud Data Flow](#)
[Spring Security](#)
[Spring Session](#)
[Spring Integration](#)
[Spring HATEOAS](#)
[Spring REST Docs](#)
[Spring Batch](#)
[Spring AMQP](#)
[Spring for Android](#)
[Spring CredHub](#)
[Spring Flo](#)

# Spring Framework 5.2.4


[OVERVIEW](#)
[LEARN](#)

## Documentation

Each **Spring project** has its own; it explains in great details how you can use **project features** and what you can achieve with them.

5.2.4	CURRENT GA	<a href="#">Reference Doc.</a>	<a href="#">API Doc.</a>
5.2.5	SNAPSHOT	<a href="#">Reference Doc.</a>	<a href="#">API Doc.</a>
5.1.15	SNAPSHOT	<a href="#">Reference Doc.</a>	<a href="#">API Doc.</a>
5.1.14	GA	<a href="#">Reference Doc.</a>	<a href="#">API Doc.</a>
5.0.16	GA	<a href="#">Reference Doc.</a>	<a href="#">API Doc.</a>
4.3.26	GA	<a href="#">Reference Doc.</a>	<a href="#">API Doc.</a>



# Spring Framework 5.x.x

## ▪ Spring Framework 5.x.x 특징

- JavaSE 8과 Java EE 7이 베이스 라인
  - Servlet 3.1, JMS 2.0, JPA 2.1, JAX-RS 2.0,  $\pi$
- WebFlux, Reactive 프로그래밍 지원, 함수적 웹 프레임워크
- Kotlin 지원, JUnit 5 지원
- 제거됨 : Portlet, Velocity, JasperReports, XMLBeans, JDO, Guava

- **Spring Framework 5.x.x을 학습해야 하는 이유**
  - EJB 기반 개발보다 빠른 개발 및 높은 유지 보수성
  - 웹, 데이터베이스 등 특정 분야가 아닌 다양한 분야에서 프로젝트의 전체 구조를 설계할 때 유용
  - 현재 전자정부 표준프레임워크 3.9의 Spring Framework 4.3.22으로 업그레이드 되었고, 지속적으로 업그레이드될 예정임
    - <http://docs.spring.io/spring/docs/5.1.x.RELEASE/spring-framework-reference/htmlsingle/>

# 계속

## ▪ 표준프레임워크 4.x (2022.03) 실행환경

- Package 경로가 org.egovframe.rte로 변경 (groupId, artifactId 변경)
- **SpringBoot 2.4.5** 지원
- org.egovframe.rte.fdl.cmmn (spring framework 5.3.6 버전 변경)
- org.egovframe.rte.fdl.security (spring security 5.3.6 버전 변경)
- org.egovframe.rte.bat.core (spring batch 4.3.2 버전 변경)

# 계속

## ■ 전자정부 표준 프레임워크 3.10 (2021.03) 실행환경

- egovframework.rte.fdl.cmmn (spring framework 4.3.25 버전 변경)
- egovframework.rte.fdl.security (spring security 4.2.13 버전 변경)
- egovframework.rte.bat.core(spring batch 3.0.10 버전 변경)
- egovframework.rte.fdl.logging (log4j 2.13.2 버전 변경)
- egovframework.rte.psl.data.jpa (hibernate 5.4.10 버전 변경)
- egovframework.rte.psl.dataaccess (mybatis 3.5.3 버전 변경)

# 규모에 따른 서버 구성

## ■ 소규모 : 웹 애플리케이션 서버 단독

- 정적인 웹 페이지(HTML 5, CSS3, JavaScript, jQuery)와 동적인 웹 페이지(JSP, Servlet)를 처리함

## ■ 중규모 : 웹 서버와 애플리케이션 서버 분리

- 웹 서버 : 정적인 웹 페이지 처리
- 애플리케이션 서버 : JSP, Spring MVC, Spring Security, Spring Data JPA, MyBatis 등 처리

## ■ 대규모 : 웹 애플리케이션 서버와 애플리케이션 서버

- 웹 애플리케이션 서버 : 주로 동적인 웹 페이지, 프리젠테이션 로직 처리
- 애플리케이션 서버 : 업무 로직과 데이터 액세스 및 다른 조직의 애플리케이션 시스템과 연동 기능 제공

# 학습 후 기대효과

- Spring 등장 배경, 목표, 정의 및 특징을 설명할 수 있다.
- Spring 프로젝트를 생성할 수 있다.

# 계속

- POJO(Plain Old Java Object) 방식:
  - POJO는 Java EE의 EJB 를 사용하면서 해당 플랫폼에 종속되어 있는 무거운 객체들을 만드는 것에 반발하며 나타난 용어다. 별도의 프레임워크 없이 Java EE를 사용할 때에 비해 특정 인터페이스를 직접 구현하거나 상속받을 필요가 없어 기존 라이브러리를 지원하기가 용이하고, 객체가 가볍다.
- 관점 지향 프로그래밍(Asspect Oriented Programming, AOP):
  - 로깅, 트랜잭션, 보안 등 여러 모듈에서 공통적으로 사용하는 기능을 분리하여 관리할 수 있다. AspectJ를 포함하여 사용할 수 있고, 스프링에서 지원하는 실행에 조합하는 방식도 지원한다. 이 분리관리한다는게 개념이 처음에 이해해기가 어려운데, 추상/부모/클래스나 인터페이스로 관리된다는게 아니라 모듈을 관리해주는 모듈을 상하/인터페이스 관계없이 따로 마련한다는 개념에 가깝다. Spring에서는 반복할당을 줄이기 위해 포인터를 대신하여 스프링 어노테이션을 사용하는 것이라고 보면 된다.
- 의존성 주입(Dependency Injection, DI):
  - 프로그래밍에서 구성요소 간의 의존 관계가 소스코드 내부가 아닌 외부에서 설정을 통해 정의되는 방식이다. 코드 재사용을 높여 소스코드를 다양한 곳에 사용할 수 있으며 모듈간의 결합도도 낮출 수 있다. 계층, 서비스 간에 의존성이 존재하는 경우 스프링 프레임워크가 서로 연결시켜준다.
- 제어 역전(Inversion of Control, IoC):
  - 전통적인 프로그래밍에서는 개발자가 작성한 프로그램이 외부 라이브러리의 코드를 호출해서 이용했다. 제어 역전은 이와 반대로 외부 라이브러리 코드가 개발자의 코드를 호출하게 된다. 즉, 제어권이 프레임워크에 있어 필요에 따라 스프링 프레임워크가 사용자의 코드를 호출한다.
- 생명주기 관리
  - 스프링 프레임워크는 Java 객체의 생성, 소멸을 직접 관리하며 필요한 객체만 사용할 수 있다.