

# Chapter 01. 시작하기

---

**01.스프링이란?**

02.POJO 프로그래밍

03.스프링 기술

04.Spring @MVC 로 시작하기

# 자바 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크

## 1.2 자바 엔터프라이즈 개발

### 1.2.1 기업 대상 애플리케이션 개발

- 은행(금융), 네이버, 물류/유통 회사, 병원
- .... Future Business

### 1.2.2 환경과 조건

- C/S ( network )
- 웹 환경
- 데이터베이스
- 분산환경 ( 분산객체, 자원 관리, 컴포넌트 )

### 1.2.2 JEE( Java Enterprise Edition )

- Servlet/JSP, JDBC, EJB, RMI, JNDI, JTA, JMS ...

## 1.3 프레임워크( Framework )

### 1.3.1 정의

- 소프트웨어를 만드는 데 기본이 되는 골격 코드
- 반제품
- 완전한 애플리케이션 소프트웨어가 아니다.
- 문제 영역(도메인)을 해결하기 위한 잘 설계된 재사용 가능한 모듈
- 확장하여 비즈니스 요구사항에 맞는 완전한 애플리케이션으로 완성이 요구된다.

### 1.3.2 종류( 분류 )

- 웹 애플리케이션 프레임워크  
Struts, WebWork, Spring MVC
- 데이터베이스 애플리케이션 프레임워크  
iBatis( MyBatis ), Hibernate, Spring DAO
- 기타(지원) 프레임워크  
로깅(Log4J), 빌드/배포(Ant), 단위테스트(JUnit)

## 1.3 프레임워크( Framework )

### 1.3.3 애플리케이션 프레임워크

- 특정 계층, 기술, 특정 비즈니스에 국한되지 않은 애플리케이션 전 영역을 포괄
- 개발 전 과정을 빠르고 편리하며 효율적으로 진행하는 것을 목표
- 자바 개발의 폭넓은 간소화
- EJB, Spring

### 1.3.4 EJB( Enterprise Java Bean )

- Java Bean 이란?
  1. 컴포넌트 기반의 소프트웨어 모델 스펙(1996년 12월 SUN)
  2. 자바 객체를 재사용 가능하게 즉, 컴포넌트화 시킬 수 있는 코딩 방침을 정의
  3. 자바 빈즈 스펙에 맞게 구현된 자바코드를 웹에서 쉽게 이용하기 위해 JSP 표준액션 태그 지원  
예) <jsp:useBean>, <jsp:getProperty>, <jsp:setProperty> 지원
  4. 스펙의 일부
    - 디폴트 생성자 존재
    - 프로퍼티 변수는 private, protected로 정의
    - public 접근 지정자를 가지는 setXXX(), getXXX() 메소드 작성
  5. 엔터프라이즈 어플리케이션에서 필요한 보안, 트랜잭션, 분산 컴퓨팅 등의 서비스는 제공 않음

## 1.3 프레임워크( Framework )

### 1.3.4 EJB( Enterprise Java Bean )

– Enterprise Java Bean 이란?

1. 1998년 3월 Sun에서 엔터프라이즈급 어플리케이션 개발을 단순화하기 위해 발표한 스펙

2. 다수의 J2EE 서버 개발 벤더에서 EJB 스펙을 구현하여 WAS 제품 출시

예) BEA의 WebLogic, IBM의 WebSphere, TMax의 Jeus 등

3. 보안, 트랜잭션지원, 분산 컴퓨팅 등의 엔터프라이즈 어플리케이션 개발 시 필요한 다양한 서비스를 컨테이너에서 제공하며 개발자는 비즈니스 로직에 전념하도록 지원

4. 컨테이너의 다양한 서비스를 제공받기 위해서는 지켜야 하는 EJB 스펙 자체가 복잡 작성된 코드는 EJB 컨테이너가 없을 경우 사용할 수 없으며 EJB 컨테이너도 벤더에 따라 구현한 내용이 다르고 컨테이너가 변경될 경우 호환이 어렵다

## 1.3 프레임워크( Framework )

### 1.3.5 Spring

- 2002년 로드 존슨( Rod Johnson ) 이 쓴 「Expert one-on-one:J2EE Design and Development」에서 소개된 소스코드를 기반으로 2003년 2월에 시작된 오픈 소스 프로젝트
- POJO(Plain Old Java Object) 특정클래스를 상속하거나 인터페이스를 구현하지 않는 평범한 자바 클래스(느슨한 Java Bean, Spring Bean)를 이용하며 단순하지만 EJB에서 제공하는 고급 기술을 제공한다.
- 진정한 의미의 자바 개발의 폭 넓은 간소화 를 실현한 프레임워크
- 20여 개의 모듈과 수십만 라인의 복잡하고 방대한 규모
- 불필요하게 무겁지 않다. ( EJB와 비교 )
- 코드는 단순하고 개발과정은 편리
- 고급 기능을 세련된 방식으로 적용
- 군더더기 없이 깔끔한 기술을 가진 "경량급" 프레임워크
- 비슷한 기술 수준에서 훨씬 빠르고 간편하게 작성이 가능

### 1.3.5 Spring

#### - 컨테이너( Container )

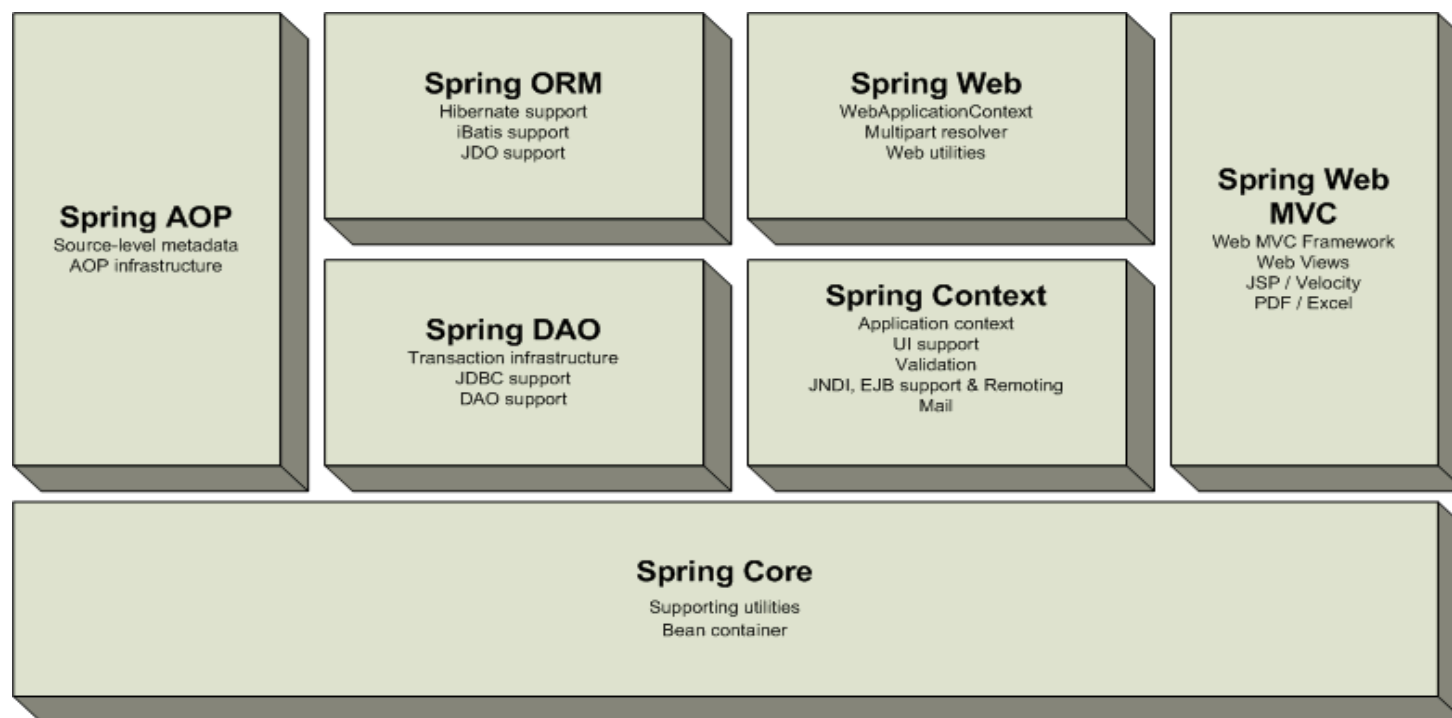
1. EJB의 비즈니스 서비스 컨테이너의 기능은 유지하되 복잡성을 제거한 컨테이너의 필요성
2. 객체들의 라이프사이클을 관리해주는 컨테이너의 기본적인 기능
3. 컨테이너에서 제공하는 API를 상속받거나 구현하여 코드를 작성하는 부분들을 제거
4. 컨테이너를 이루는 파일자체가 몇 메가 밖에 안 되는 작은 사이즈이며 구동에 필요한 시간이 짧고 자체 부하는 무시할 수준이고 컨테이너 내에 객체를 배치하는 복잡한 과정이 짧다.
5. 컨테이너의 필요성
  - 가) 컴포넌트, 객체의 자유로운 삽입이 가능하도록 하기 위한 호출의 독립성
  - 나) 서비스를 설정하거나 찾기 위한 일관된 방법을 제시
  - 다) 싱글톤이나 팩토리를 구현할 필요 없이 단일화된 객체에 대한 접근방법을 제공
  - 라) 비즈니스 객체에 부가적으로 필요한 각종 엔터프라이즈 서비스를 제공



## 1.3 프레임워크( Framework )

### 1.3.5 Spring

- 주요 모듈



### 1.3.5 Spring

#### - 주요 전략

1. POJO를 이용한 가볍고(lightweight) 비침투적(non-invasive) 개발
2. DI와 인터페이스 지향을 통한 느슨한 결합도(loose coupling)
3. Aspect와 공통 규약을 통한 선언적(declarative) 프로그래밍
4. Aspect와 템플릿(template)을 통한 반복적이고 상투적인(boilerplate) 코드 제거

# Chapter 01. 시작하기

---

01.스프링이란?

**02.POJO 프로그래밍**

03.스프링 기술

04.Spring @MVC 로 시작하기

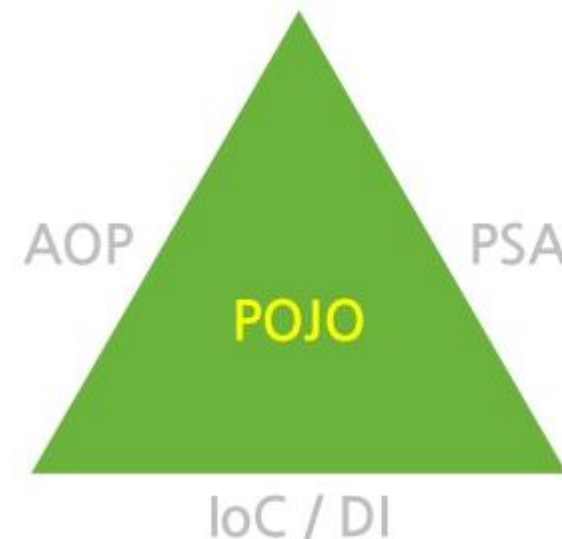
## 2.1 POJO 프로그래밍

### 2.1.1 POJO 란?

- (P)lain (O)ld (J)ava (O)bject
- 자바 언어와 꼭 필요한 API외에는 특정 규약에 종속되지 않는다.
- 특정 환경에 종속되지 않는다. (기술과 비즈니스 분리)
- 스프링에서는 스프링에 특화된 인터페이스 구현을 요구하지 않음
- 스프링 자체에 의존성이 높은 클래스 확장을 거의 요구 하지 않음

### 2.1.2 POJO 프로그래밍의 장점

- 스프링의 정수는 엔터프라이즈 개발에서 요구하는 모든 기술을 POJO를 통해 제공
- 비침투적 프로그램이 가능



# Chapter 01. 시작하기

---

01.스프링이란?

02.POJO 프로그래밍

**03.스프링 기술**

04.Spring @MVC 로 시작하기

### 3.1 IoC( 제어역전 ) 과 DI( 의존관계 주입 )

- 1) 스프링의 가장 기본이 되는 기술이자 스프링 핵심 개발 원칙
- 2) **Bean** : 스프링이 제어권을 가지고 직접 만들고 관계를 부여하는 오브젝트
- 3) 스프링 빈은 스프링 컨테이너가 생성과 관계 설정 등을 제어
- 4) IoC(DI) Container = Bean Factory = Application Context

### 3.2.1 AOP( Aspect Oriented Programming )

- 관점 지향 프로그래밍
- OOP를 더욱 더 OOP 답게 해 주는 ( 더욱 더 완벽하게 해 주는) 기술
- 관심의 분리 ( Separation of Concern )
- 횡단 관심( Crosscutting Concern )과 핵심관심( Core Concern )
- 핵심관심 모듈과 횡단 관심 모듈이 긴밀하게 결합 ( 핵심 모듈이 필요한 시점에..)
- OOP 문제점 : 중복코드, 지저분한 코드, 생산성 저하, 재활용성의 문제점
- 필요한 시점에 횡단 관심 모듈을 삽입하여 동작하게 하는 기술.
- EJB AOP, JDK Dynamic Proxy, **AspectJ**, **Spring AOP**

# 3.2 AOP

## 3.2.2 AOP 개념

1) 관심의 산재

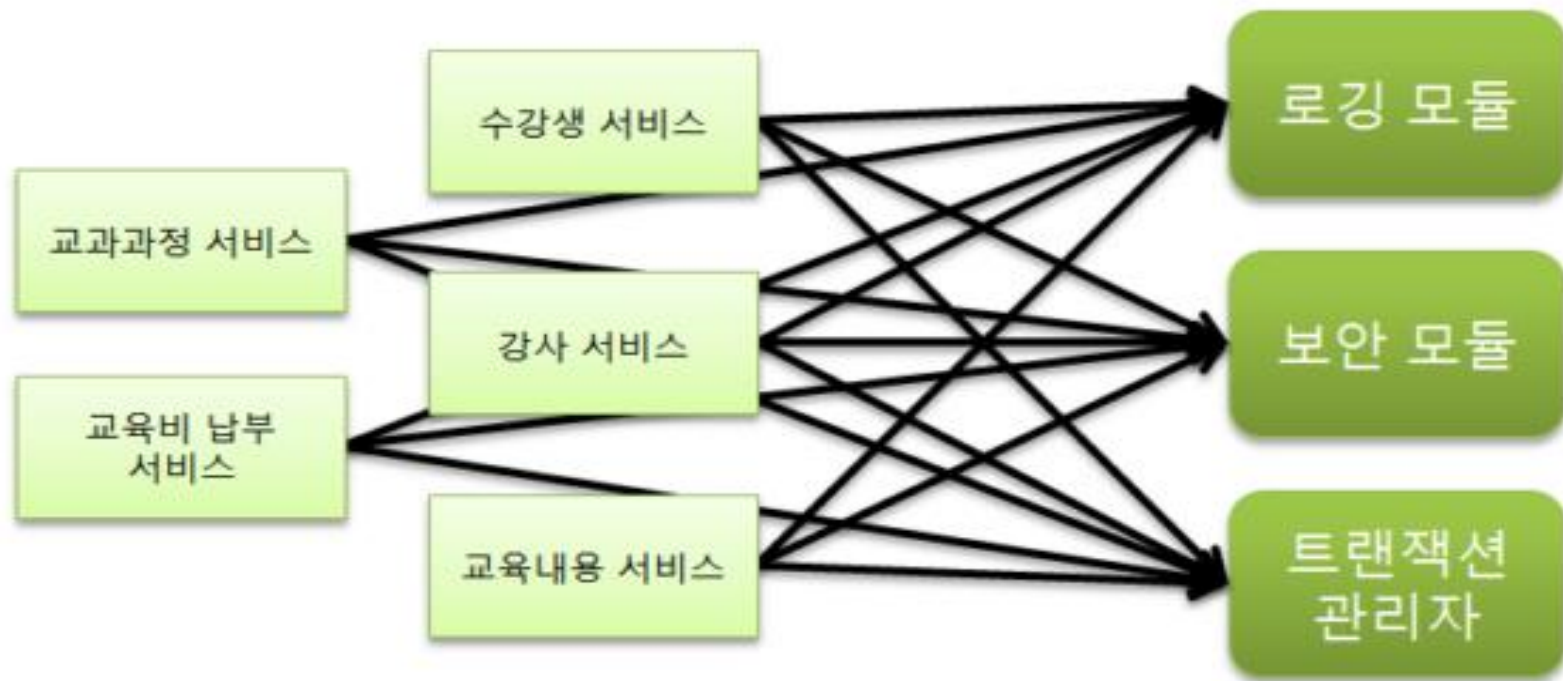
수강생 관리	강사 관리	교육 관리	...
보안...	보안...	보안...	
로깅...	로깅...	로깅...	
트랙잭션 시작...	트랙잭션 시작...	트랙잭션 시작...	
비즈니스 로직	비즈니스 로직	비즈니스 로직	
트랜잭션 끝...	트랜잭션 끝...	트랜잭션 끝...	...



## 3.2 AOP

### 3.2.2 AOP 개념

#### 2) 관심의 모듈화



# Chapter 01. 시작하기

---

01.스프링이란?

02.POJO 프로그래밍

03.스프링 기술

**04.Spring @MVC 로 시작하기**

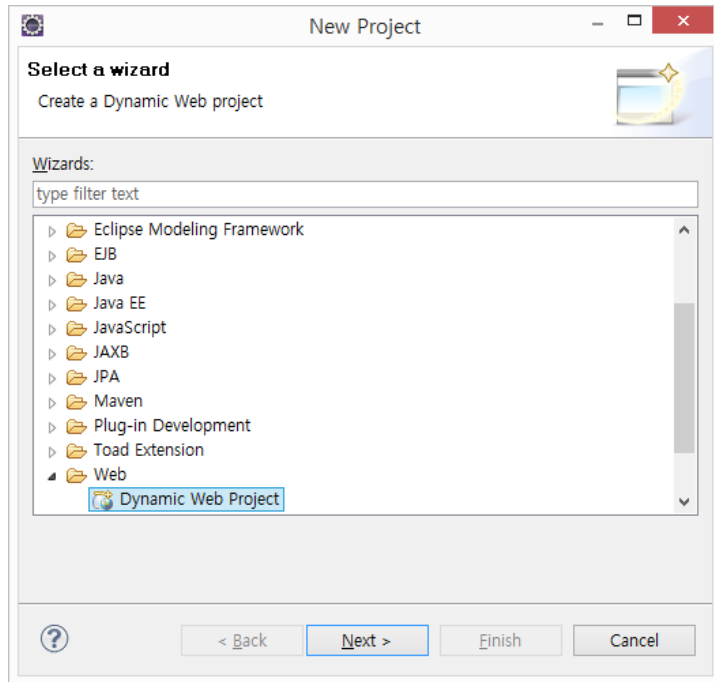
## 4.1 springex 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

Spring @MVC 기반의 springex을 같이 만들어 봅니다.

IoC/DI를 지원하는 Spring Container에 대해 생각해보고 Spring MVC기반의 웹 어플리케이션의 특징을 살펴 봅니다.

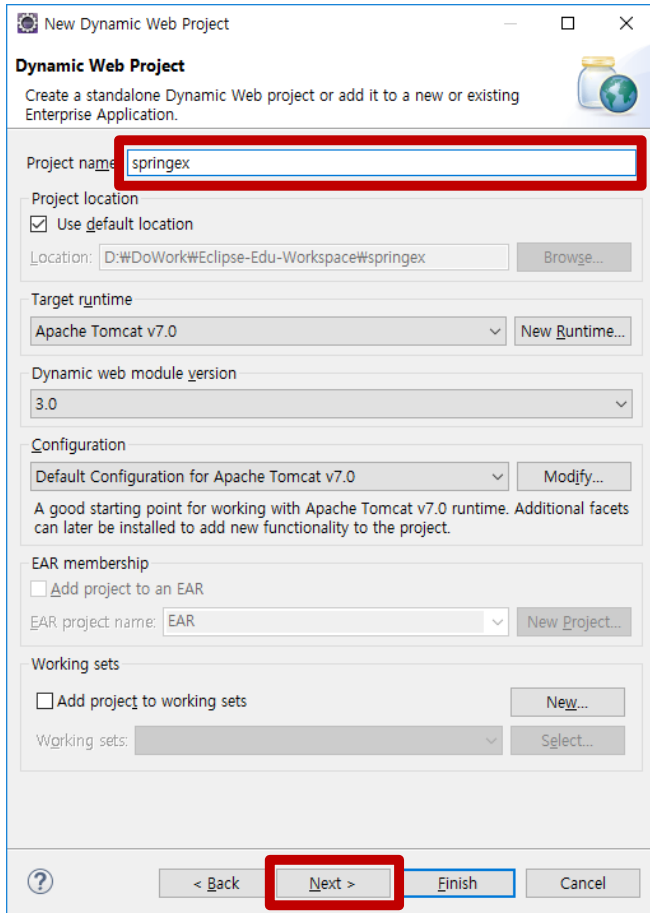
1. Dynamic Web Project 프로젝트 생성. ( 보통 스프링 프로젝트는 maven 빌드 기반의 maven 프로젝트로 생성)



## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

#### 2. project name 은 springex



New Dynamic Web Project

**Dynamic Web Project**  
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location  
☒ Use default location  
Location:

Target runtime

Dynamic web module version

Configuration  
   
A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

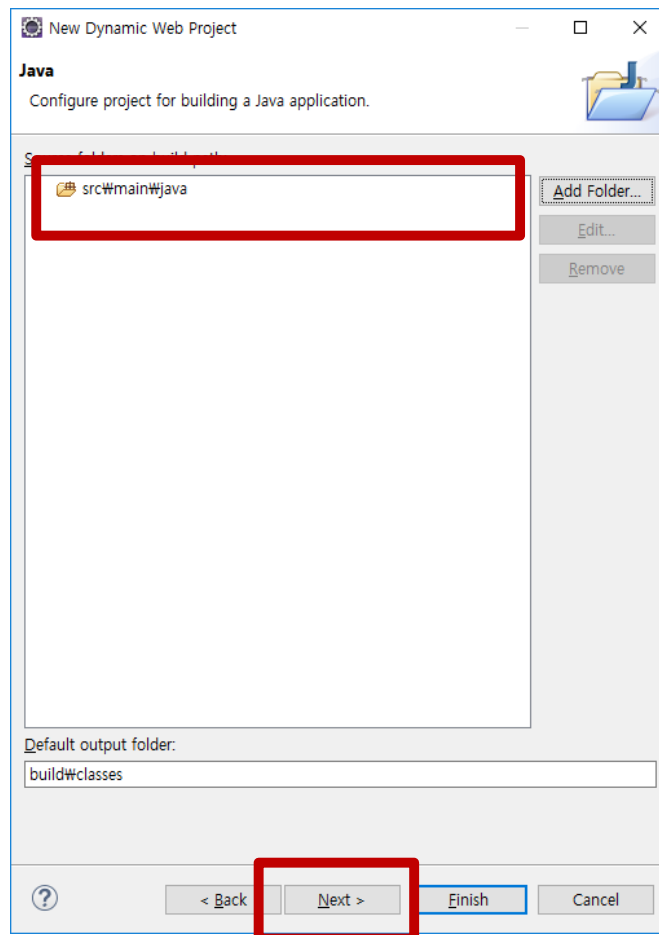
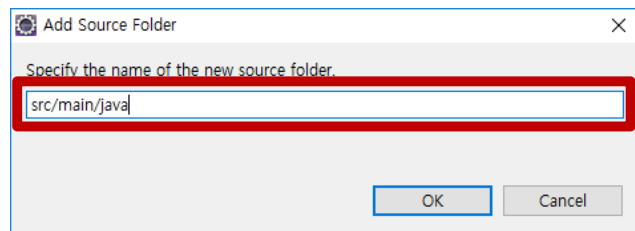
EAR membership  
☐ Add project to an EAR  
EAR project name:

Working sets  
☐ Add project to working sets   
Working sets:

## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

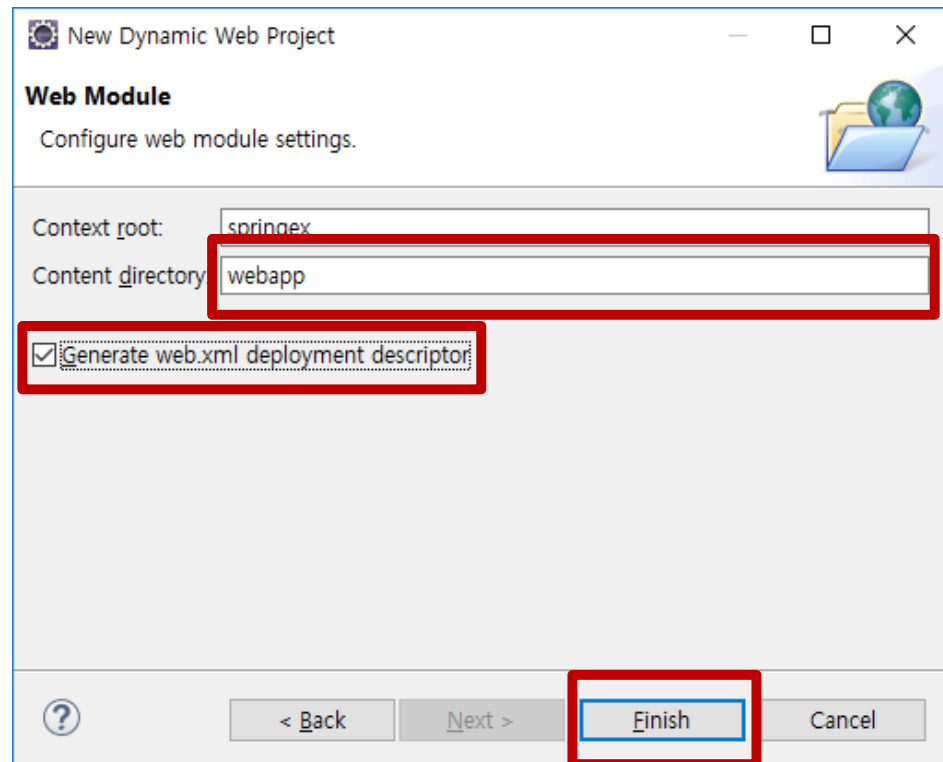
3. maven 웹 애플리케이션 프로젝트는 소스 폴더( src )가 Dynamic Web Project와 다르다. 따라서 지우고 새로 만든다.



## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

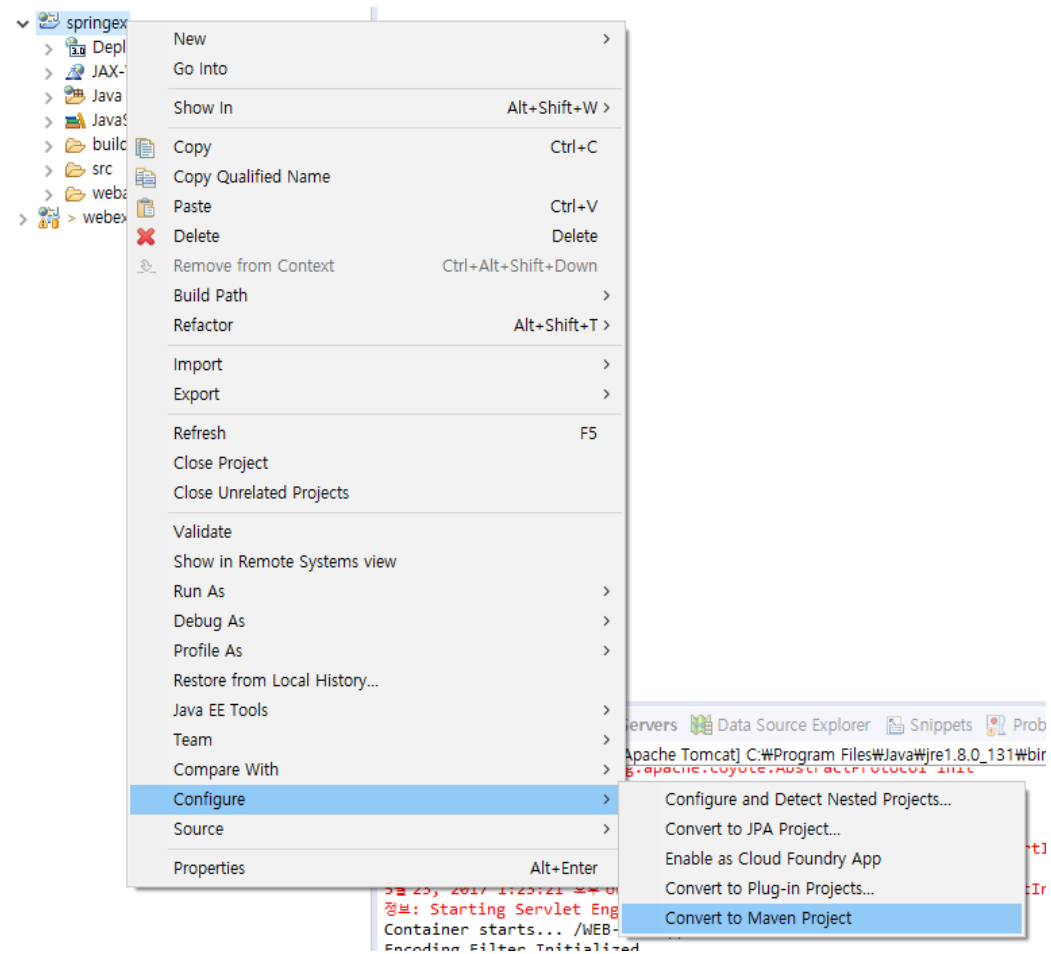
3. maven 웹 애플리케이션 프로젝트에서는 웹 콘텐츠 폴더( WebContent )도 Dynamic Web Project와 다르다. 따라서 지우고 새로 만든다.



# 4.1 hellospring 웹 애플리케이션 작성

## [실습예제] springex 웹 애플리케이션 작성하기

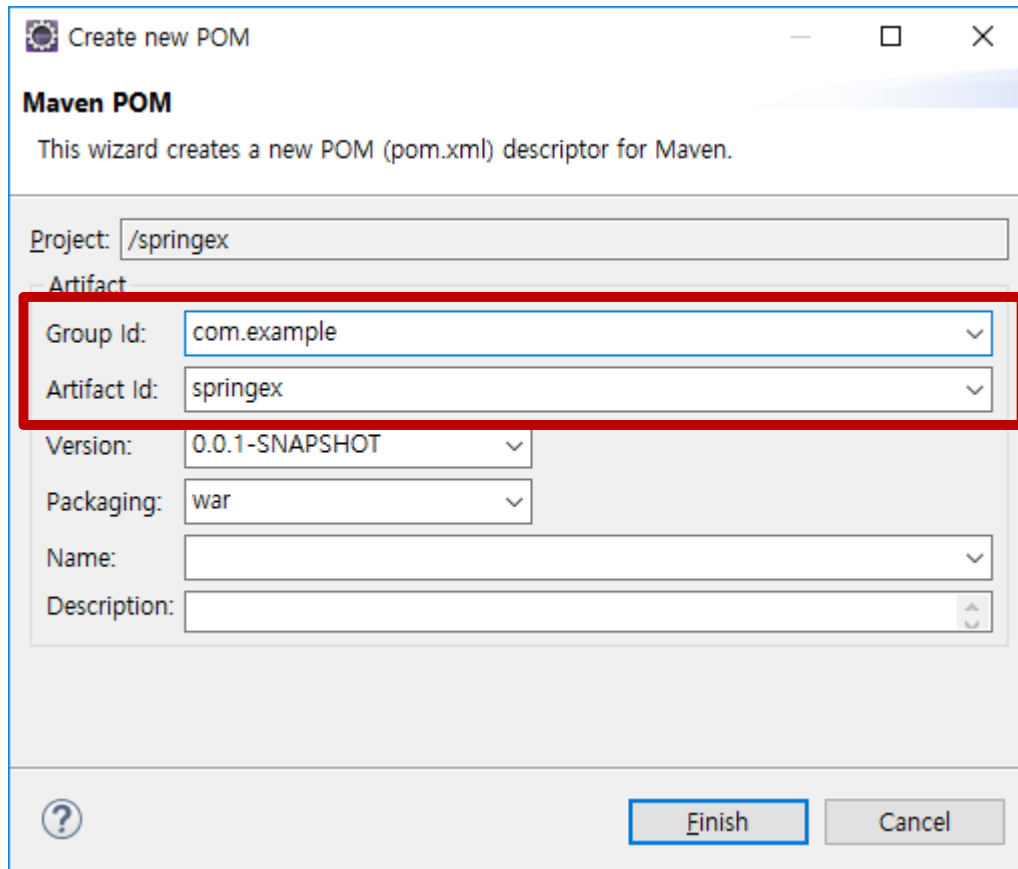
### 4. Maven 프로젝트로 바꾸기(Convert)



## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

#### 4. Maven 프로젝트로 바꾸기(Convert)



**Create new POM**

**Maven POM**

This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /springex

Artifact

Group Id: com.example

Artifact Id: springex

Version: 0.0.1-SNAPSHOT

Packaging: war

Name:

Description:

**Finish** Cancel



## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

#### 5. 프로젝트 의존성( 라이브러리 ) 추가 ( pom.xml )

##### 1) Spring Core Library 추가

```
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-context</artifactId>  
    <version>4.2.1.RELEASE</version>  
</dependency>
```

##### 2) Spring Web Library 추가

```
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-web</artifactId>  
    <version>4.2.1.RELEASE</version>  
</dependency>
```

## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

#### 5. 프로젝트 의존성( 라이브러리 ) 추가 ( pom.xml )

##### 3) Spring MVC Library 추가

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>4.2.1.RELEASE</version>
</dependency>
```

##### 4) 라이브러리 버전 프로퍼티로 관리 하기

```
<properties>
    <org.springframework-version>4.2.1.RELEASE</org.springframework-version>
</properties>
```

org.springframework-version 프로퍼티를 추가한 후, 각 각 dependency의 버전을 다음과 같이 수정한다.

```
<version>${org.springframework-version}</version>
```

## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

6. DispatcherServlet에 대한 서블릿 매핑 추가( web.xml )

```
<servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>

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

## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

#### 7. 웹 애플리케이션 컨텍스트 설정 ( /WEB-INF/spring-servlet.xml )

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
    http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <context:annotation-config />
    <context:component-scan base-package="com.example.springex.controller" />

</beans>
```

## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

#### 8. Controller 작성하기

```
@Controller
public class HelloController {

    @RequestMapping( "/hello" )
    public ModelAndView hello( @RequestParam String name ) {

        ModelAndView mav = new ModelAndView();
        mav.addObject( "hello", "Hello " + name );
        mav.setViewName( "/WEB-INF/views/index.jsp" );

        return mav;
    }
}
```

## 4.1 hellospring 웹 애플리케이션 작성

### [실습예제] springex 웹 애플리케이션 작성하기

#### 9. 실행 및 생각해 볼 것 들

##### - 추가적으로 해준 것들

- 1) pom.xml 구성
- 2) DispatcherServlet 등록( web.xml )
- 3) 서블릿 애플리케이션 컨텍스트 설정 (spring-servlet.xml)
- 4) Controller 작성

##### - 생략된 것들

- 1) 서블릿 작성
- 2) 파라미터 처리 request.getParameter()
- 3) forwarding