

Spring 시작하기

Spring이란?

Spring이란  
: 개념의 정리

자바 엔터프라이즈 개발을 편하게 해주는

오픈소스 경량급 애플리케이션 프레임워크

# Spring이란?

: 자바 엔터프라이즈 개발

- ▶ 기업 대상 애플리케이션 개발
  - ▶ 은행(금융), 포털, 물류/유통, 병원 등
  - ▶ + Future Business
- ▶ 환경과 조건
  - ▶ Client / Server (Network-Based)
  - ▶ Web 환경
  - ▶ Database
  - ▶ 분산환경(분산객체, 자원 관리, 컴포넌트)
- ▶ JEE (Java Enterprise Edition)
  - ▶ Servlet/JSP, JDBC, EJB, RMI, JNDI, JTA, JMS ...

# Spring이란?

## : Framework

### ▶ Framework의 정의

- ▶ 소프트웨어를 만드는데 기본이 되는 골격 코드
- ▶ 반제품. 완전한 애플리케이션 소프트웨어가 아님
- ▶ 문제 영역(도메인)을 해결하기 위해 잘 설계된 재사용 가능한 모듈
- ▶ 확장하여 비즈니스 요구사항에 맞는 완전한 애플리케이션으로 완성이 요구

### ▶ Framework의 종류(분류)

- ▶ 웹 애플리케이션 프레임워크
  - ▶ Struts, WebWork, Spring MVC 등
- ▶ 데이터베이스 애플리케이션 프레임워크
  - ▶ iBatis(MyBatis), Hibernate, Spring DAO 등
- ▶ 기타(지원) 프레임워크
  - ▶ Logging(Log4J), 빌드/배포(Ant, Maven), 단위 테스트(JUnit) 등

# Spring이란?

## : Application Framework

### ▶ Application Framework

- ▶ 특정 계층, 기술, 비즈니스에 국한되지 않고 애플리케이션 전 영역을 포괄
- ▶ 개발 전 과정을 빠르고 편리하며 효율적으로 진행하는 것을 목표로 함
- ▶ 자바 개발의 폭넓은 간소화
- ▶ EJB, Spring 등

# Spring이란?

## : Application Framework

### ▶ EJB (Enterprise Java Beans)

#### ▶ Java Bean이란?

- ▶ 컴포넌트 기반의 소프트웨어 모델, 스펙(1996년 12월 SUN)
- ▶ 자바 객체를 재사용 가능하게 하는(컴포넌트화 시킬 수 있는) 코딩 방침을 정의
- ▶ 자바 빈즈 스펙에 맞게 구현된 자바 코드를 웹에서 쉽게 이용하기 위한 JSP 표준 액션 태그 지원  
예) `<jsp:useBean>`, `<jsp:getProperty>`, `<jsp:setProperty>` 등
- ▶ 스펙의 일부
  - 디폴트 생성자 존재
  - 프로퍼티 변수는 `private`, `protected`로 정의
  - `public` 접근 지정자를 가지는 `setXXX()`, `getXXX()` 메서드 작성
- ▶ 엔터프라이즈 애플리케이션에서 필요한 보안, 트랜잭션, 분산 컴퓨팅 등의 서비스는 제공하지 않음

# Spring이란?

## : Application Framework

### ▶ EJB (Enterprise Java Beans)

#### ▶ Enterprise Java Beans란?

- ▶ 1998년 3월 Sun에서 발표한 엔터프라이즈급 애플리케이션 개발을 단순화 하기 위한 스펙
- ▶ 다수의 J2EE 서버 개발 벤더에서 EJB 스펙을 구현하여 WAS 제품을 출시
  - ▶ BEA의 WebLogic, IBM의 WebSphere, TMax의 JEUS 등
- ▶ 보안, 트랜잭션 지원, 분산 컴퓨팅 등 엔터프라이즈 애플리케이션 개발시 필요한 다양한 서비스를 컨테이너에서 제공하며 개발자는 비즈니스 로직에만 전념하도록 지원
- ▶ EJB의 문제
  - ▶ 컨테이너의 다양한 서비스를 제공받기 위해 지켜야만 하는 EJB 스펙 자체가 복잡
  - ▶ 작성된 코드는 EJB 컨테이너가 없을 경우 사용할 수 없음
  - ▶ EJB 컨테이너의 구현이 벤더에 따라 달라 컨테이너가 변경될 경우 호환이 어려웠음



# Spring이란?

## : Application Framework

### ▶ Spring Framework

- ▶ 2002년 Rod Johnson이 저술한 <Expert one-on-one: J2EE Design and Development>에서 소개된 코드를 기반으로 2003년 2월에 시작된 오픈소스 프로젝트
  - ▶ 복잡하고 무거운 J2EE에 대한 반발, EJB 없이 엔터프라이즈 애플리케이션을 개발하자고 주장
- ▶ POJO (Plain Old Java Object) : 특정 클래스를 상속하거나 인터페이스를 구현하지 않는 평범한 자바 클래스(느슨한 Java Bean, Spring Bean)를 이용하며 단순하지만 EJB에서 제공하는 고급 기술을 제공
- ▶ 진정한 의미에서 자바 개발의 폭넓은 간소화를 실현한 프레임워크
- ▶ 20여개가 넘는 다양한 모듈로 구성, 수십만 라인의 복잡하고 방대한 규모
- ▶ EJB와 비교하여 불필요하게 무겁지 않으며 고급 기능을 세련된 방식으로 적용
- ▶ 코드는 단순해지고 개발 과정은 편리해짐
- ▶ 군더더기 없이 깔끔한 기술을 가진 "경량급" 프레임워크
- ▶ 비슷한 기술 수준에서 훨씬 빠르고 간편하게 작성 가능

# Spring이란?

## : Application Framework

### ▶ Spring Framework

#### ▶ Container

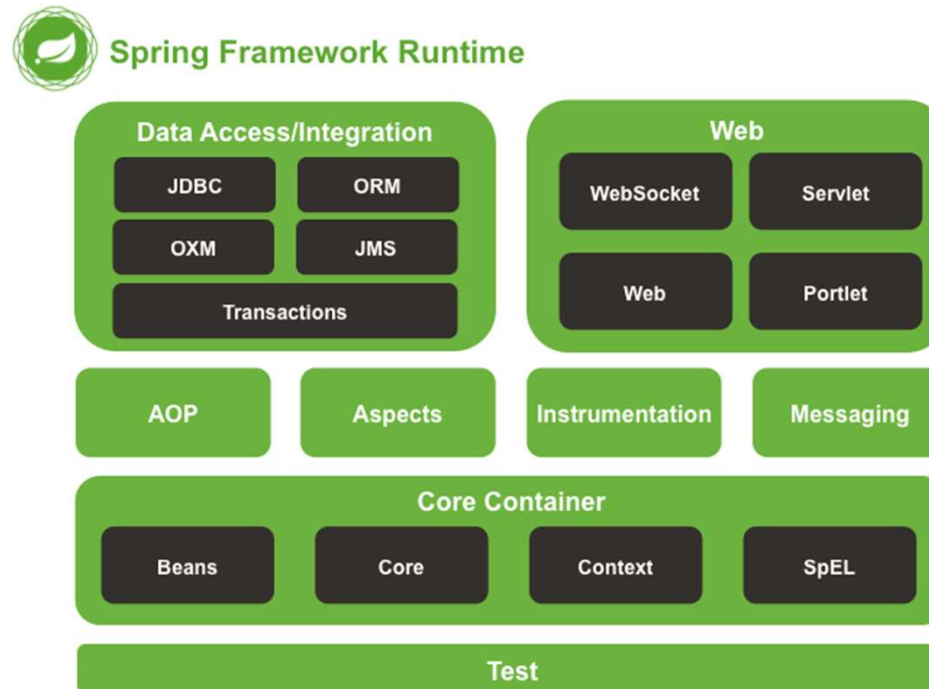
1. EJB의 비즈니스 서비스 컨테이너의 기능은 유지하되 복잡성을 제거한 컨테이너의 필요성
2. 객체들의 라이프사이클을 관리해주는 컨테이너의 기본적인 기능
3. 컨테이너에서 제공하는 **API**를 상속받거나 구현하여 코드를 작성하는 부분들을 제거
4. 컨테이너를 이루는 파일 자체의 용량이 매우 작으며 구동에 필요한 시간이 짧고 자체 부하는 무시할 수 있는 수준이며 컨테이너 내에 객체를 배치하는 복잡한 과정이 짧다
5. 컨테이너의 필요성
  - 컴포넌트, 객체의 자유로운 삽입이 가능하도록 하기 위한 호출의 독립성
  - 서비스를 설정하거나 찾기 위한 일관된 방법을 제시
  - 싱글톤이나 팩토리를 구현할 필요 없이 단일화된 객체에 대한 접근 방법을 제공
  - 비즈니스 객체에 부가적으로 필요한 각종 엔터프라이즈 서비스를 제공

# Spring이란?

: Application Framework

## ▶ Spring Framework

### ▶ 주요 모듈



# Spring이란?

## : Application Framework

### ▶ Spring Framework

#### ▶ 주요 전략

1. POJO를 이용한 가볍고(Lightweight) 비침투적(non-invasive) 개발
2. DI와 인터페이스 지향을 통한 느슨한 결합도 (Loosely Coupling)
3. Aspect와 공통 규약을 통한 선언적(Declarative) 프로그래밍
4. Aspect와 템플릿(Template)을 이용한 반복적이고 상투적인(Boilerplate) 코드 제거

# POJO 프로그래밍

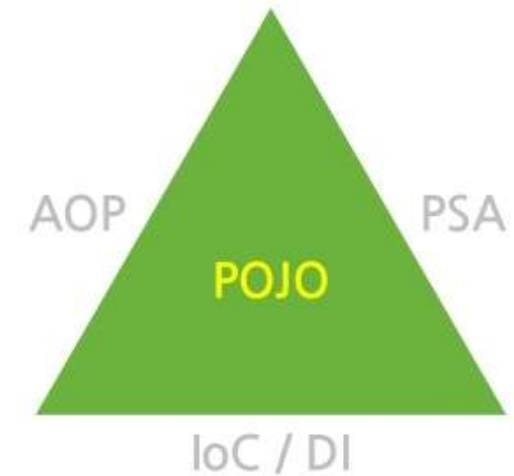
# POJO

## ▶ POJO의 개념

- ▶ Plain Old Java Object
- ▶ 자바 언어와 꼭 필요한 **API** 외에는 특정 규약에 종속되지 않는다
- ▶ 특정 환경에 종속되지 않는다 (기술과 비즈니스의 분리)
- ▶ 스프링에서는 스프링에 특화된 인터페이스 구현을 요구하지 않음
- ▶ 스프링 자체에 의존성이 높은 클래스 확장을 거의 요구하지 않음

## ▶ POJO 프로그래밍의 장점

- ▶ 스프링의 정수는 엔터프라이즈 개발에서 요구하는 모든 기술을 **POJO**를 통해 제공
- ▶ 비침투적 프로그래밍이 가능



# Spring 기술

IoC, DI 그리고 AOP

# IoC와 DI

## ▶ 용어의 설명

- ▶ IoC (Inversion of Control: 제어의 역전)
- ▶ DI (Dependency Injection: 의존성 주입)

## ▶ IoC와 DI

- ▶ 스프링의 가장 기본이 되는 기술이자 스프링 핵심 개발 원칙
- ▶ Bean : 스프링이 제어권을 가지고 직접 만들고 관계를 부여하는 객체
- ▶ Spring Bean : 스프링 컨테이너가 생성과 관계 설정 등을 제어
- ▶ IoC(DI) Container = Bean Factory = Application Context



# AOP

- ▶ AOP(Aspect Oriented Programming)
  - ▶ 관점 지향 프로그래밍
  - ▶ OOP를 대체하는 개념이 아니라 OOP를 더 OOP답게 해주는 기술
  - ▶ 관심의 분리 (Separation of Concern)
  - ▶ 횡단 관심(Crosscutting Concern)과 핵심 관심(Core Concern)
  - ▶ 핵심 관심 모듈과 횡단 관심 모듈이 긴밀하게 결합
  - ▶ OOP의 문제점 : 중복 코드, 지저분한 코드, 생산성 저하, 재활용성의 문제점
  - ▶ 필요한 시점에 횡단 관심 모듈을 삽입하여 동작하게 하는 기술
  - ▶ EJB AOP, JDK Dynamic Proxy, AspectJ, Spring AOP 등

# AOP

## ▶ AOP 개념

### ▶ 관심의 산재

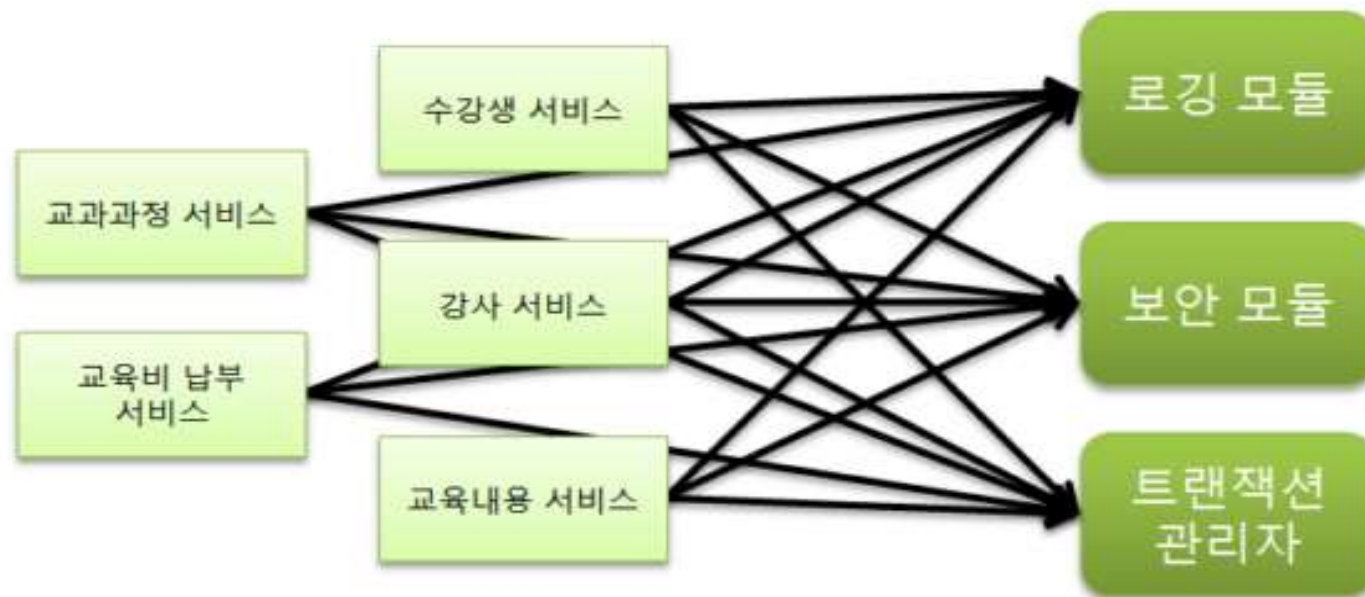
핵심 관심: Core Concern



# AOP

## ▶ AOP 개념

### ▶ 관심의 모듈화

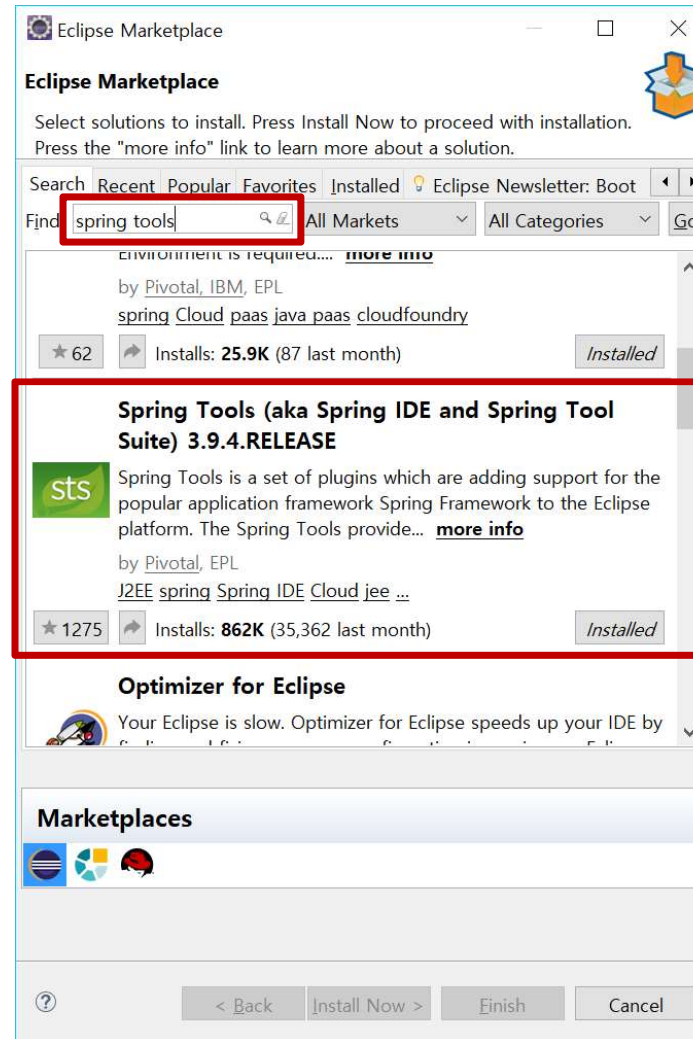


# Spring Web MVC 시작하기

# Spring Web MVC 시작하기

## : Eclipse 설정

- ▶ Help > Eclipse Marketplace
  - ▶ Spring Tools 검색 및 설치



# Spring Web MVC 시작하기

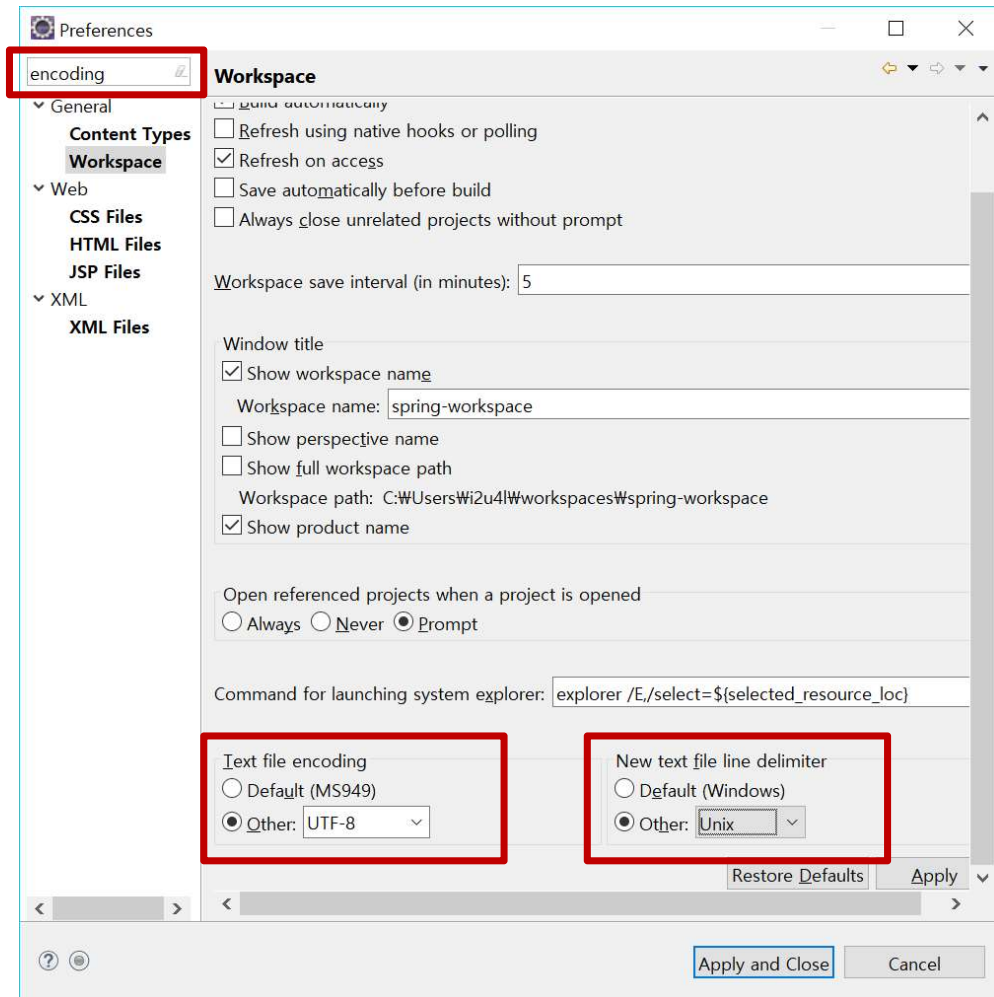
## : Eclipse 설정

### ▶ Workspace Encoding 설정

- 1) Window > Preference
- 2) 검색 창에서 encoding 검색
- 3) Workspace 항목의 encoding 변경

### ▶ Web File Encoding 설정

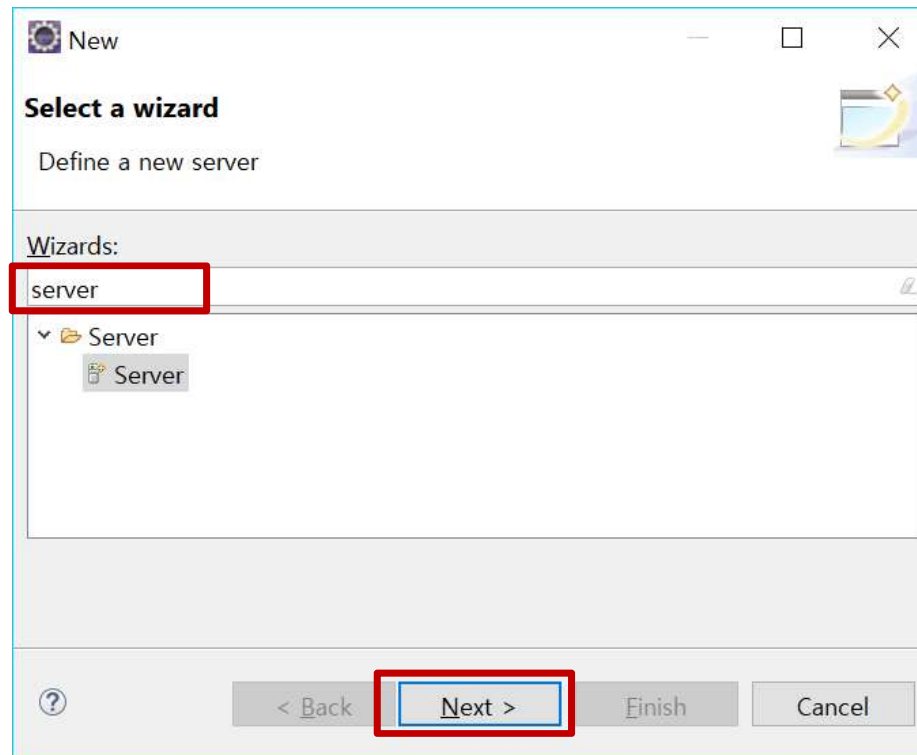
- ▶ 다음 파일들의 인코딩도 UTF-8로 변경
  - ▶ CSS Files, HTML Files, JSP Files



# Spring Web MVC 시작하기

## : Tomcat 설치 및 설정

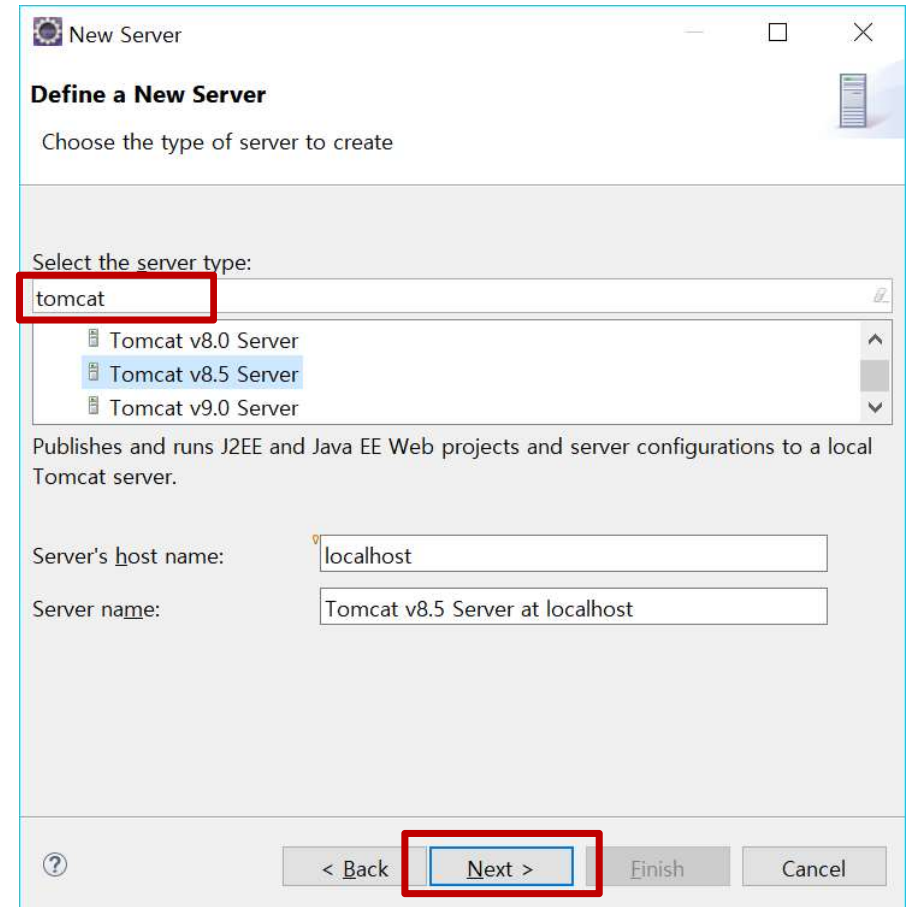
- ▶ 톰캣 다운로드 및 압축 해제
  - ▶ 개발 PC의 운영체제에 맞는 버전을 다운로드, 별도의 설치 작업은 필요 없음
  - ▶ <http://tomcat.apache.org/>
- ▶ Eclipse에 설정
  - ▶ File > New > Others
    - ▶ Server 검색



# Spring Web MVC 시작하기

## : Tomcat 설치 및 설정

- ▶ Eclipse에 설정
  - ▶ 다운로드 받은 Tomcat 버전과 동일한 서버 템플릿 선택
  - ▶ 필요하다면 host name과 Server name도 설정한다



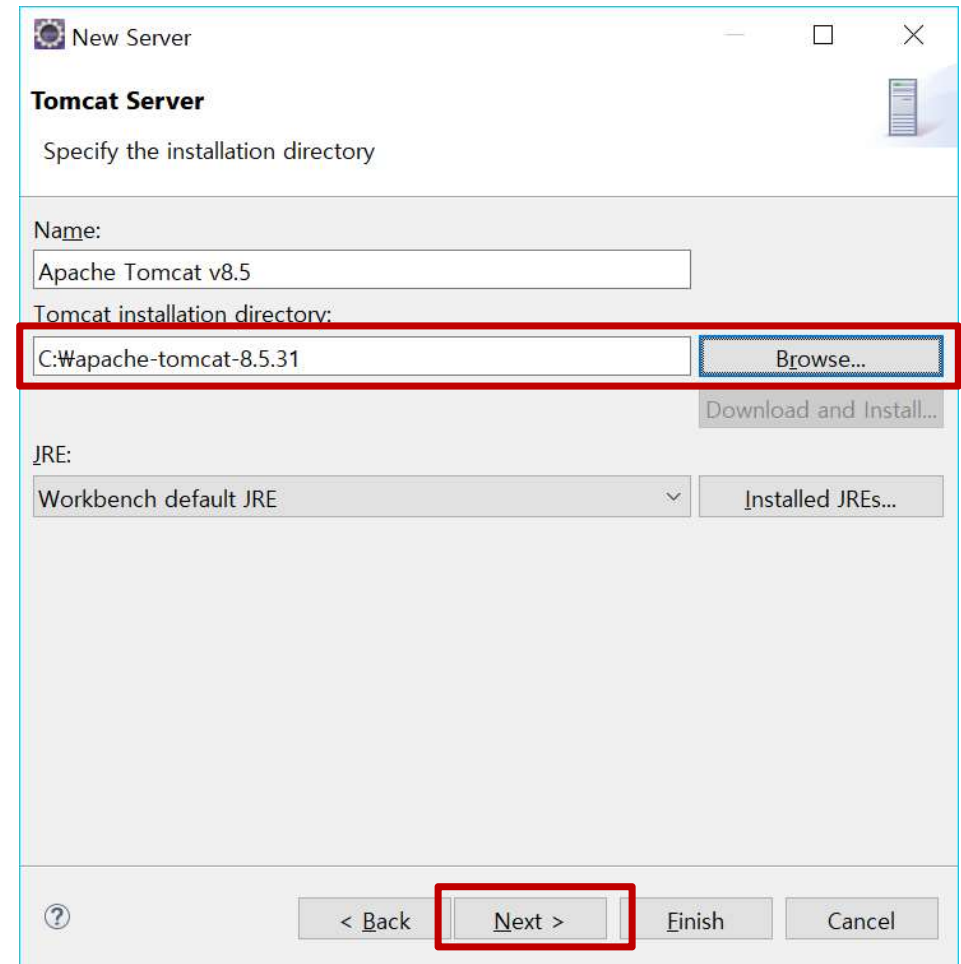


# Spring Web MVC 시작하기

## : Tomcat 설치 및 설정

### ▶ Eclipse에 설정

#### ▶ Tomcat이 설치되어 있는 디렉터리 선택

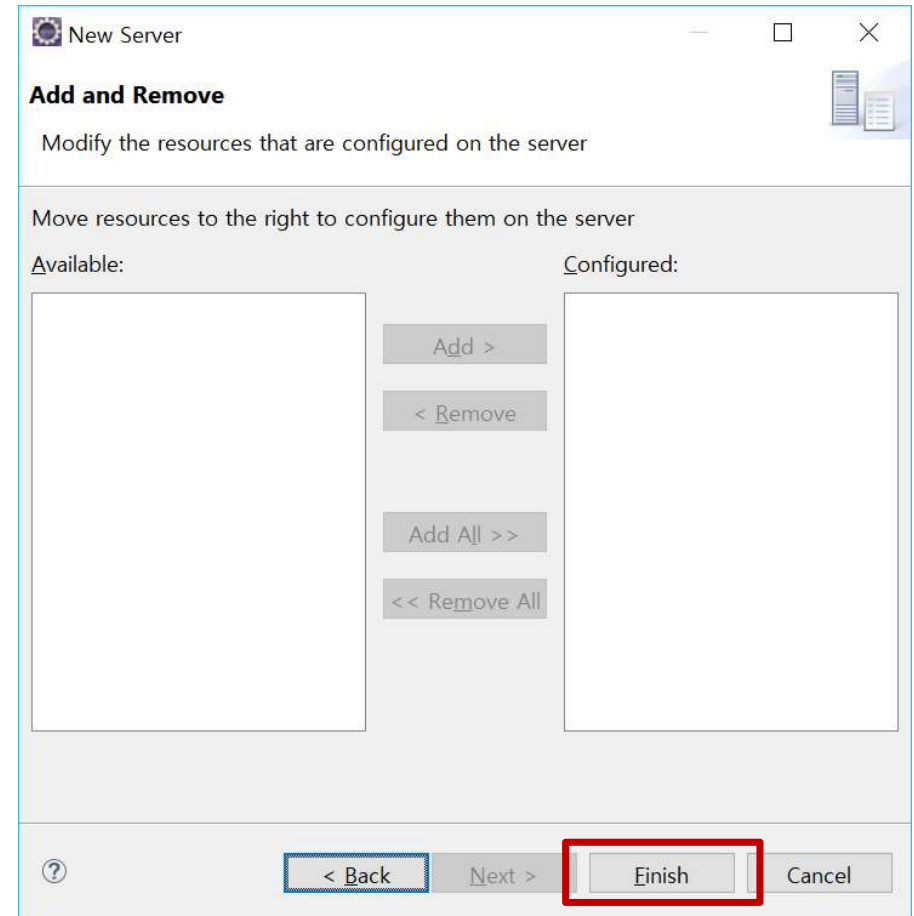


# Spring Web MVC 시작하기

## : Tomcat 설치 및 설정

### ▶ Eclipse에 설정

- ▶ (만약 있다면) 웹 리소스를 톰캣에 추가(혹은 삭제)

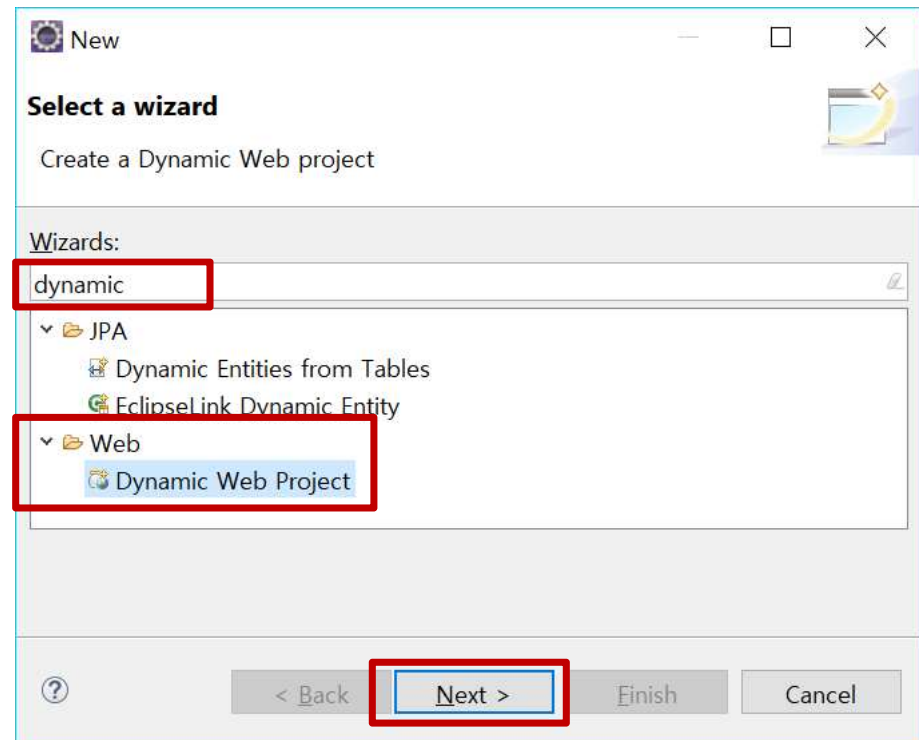


# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ Spring Web MVC 기반 애플리케이션을 작성해 봅니다  
IoC/DI를 지원하는 **Spring Container**에 대해 생각해 보고  
Spring MVC 기반의 웹 애플리케이션의 특징을 살펴봅니다.

- ▶ **Dynamic Web Project** 프로젝트 생성  
(보통 스프링 프로젝트는 **Maven** 빌드 기반의  
Maven 프로젝트로 작성)  
File > New > Other ...  
Dynamic 검색



# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ Project name 설정 : hellospring
- ▶ Target runtime에 Tomcat 서버 선택

**New 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 v8.5 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

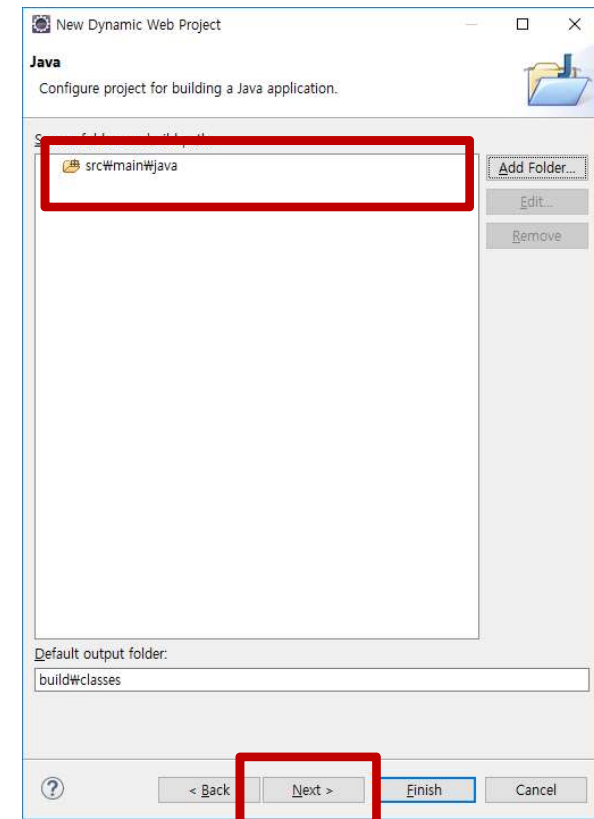
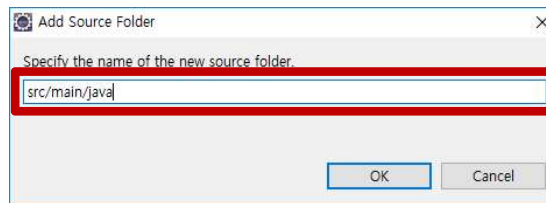
Working sets:

# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

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

- ▶ src/main/java



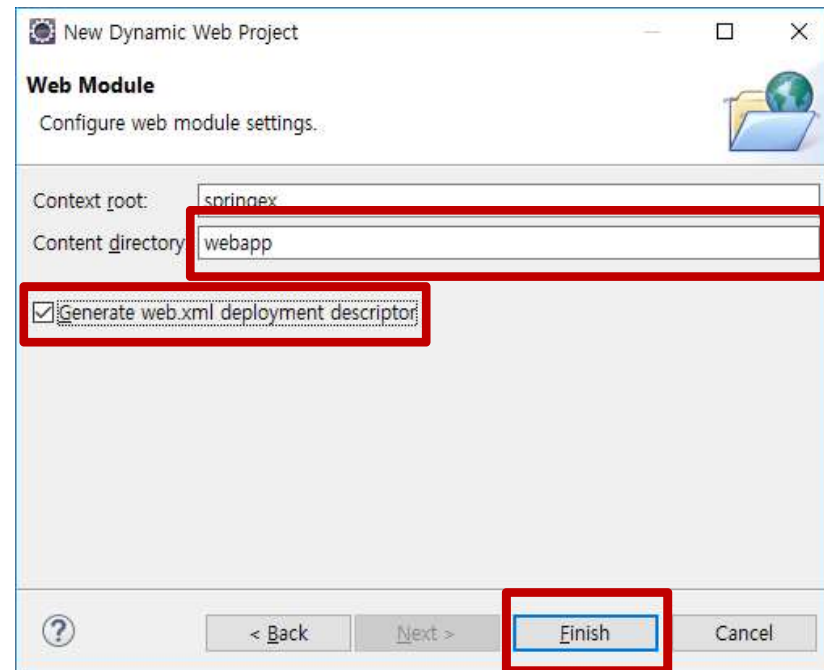
# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ maven 웹 애플리케이션 프로젝트는 웹 콘텐츠 폴더(WebContent)도 Dynamic Web Project와 다르다. 지우고 새로 만든다.

- ▶ webapp

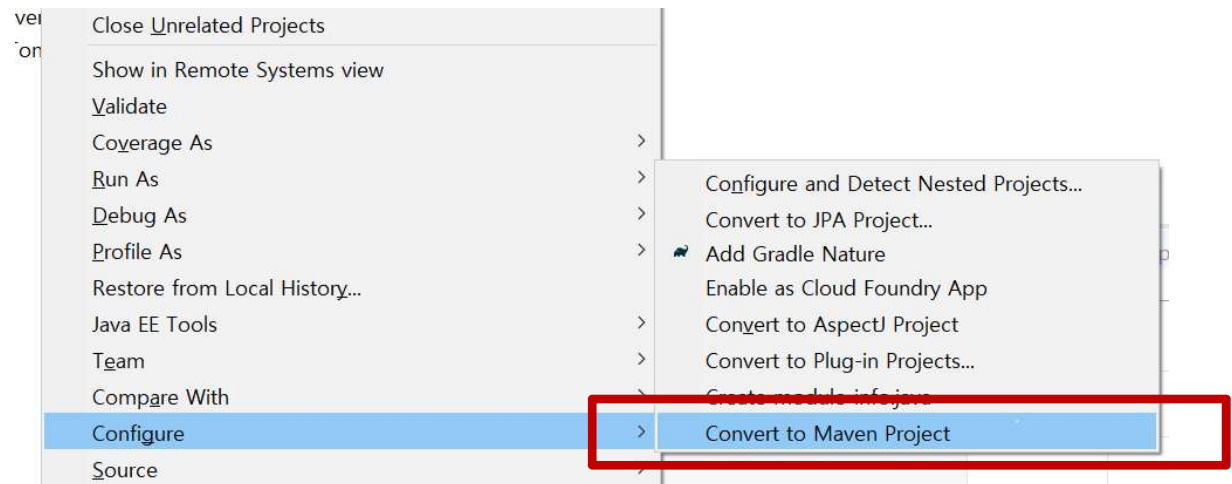
- ▶ Generate web.xml deployment descriptor 체크



# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ maven 프로젝트로 변경 (Convert)
  - ▶ 프로젝트 우클릭
  - ▶ Configure > Convert to Maven Project



# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ maven 프로젝트로 변경 (Convert)
  - ▶ 적절한 Group id와 Artifact id로 변경
- ▶ 여기서는 다음과 같이 설정
  - ▶ Group id : com.example
  - ▶ Artifact id : hellospring

Create new POM

**Maven POM**

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

Project: /hellospring

Artifact

Group Id: com.example

Artifact Id: hellospring

Version: 0.0.1-SNAPSHOT

Packaging: war

Name:

Description:

Finish Cancel



# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ 프로젝트 의존성(라이브러리) 추가 (in pom.xml)
  - ▶ Spring Core Library 추가

```
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-context</artifactId>  
  <version>4.2.3.RELEASE</version>  
</dependency>
```

- ▶ Spring Web Library 추가

```
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-web</artifactId>  
  <version>4.2.3.RELEASE</version>  
</dependency>
```

# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ 프로젝트 의존성(라이브러리) 추가 (in pom.xml)

- ▶ Spring MVC Library 추가

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>4.2.3.RELEASE</version>
</dependency>
```

- ▶ 라이브러리 버전을 프로퍼티로 관리하기

```
<properties>
  <org.springframework-version>4.2.3.RELEASE</org.springframework-version>
</properties>
```

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

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

# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ DispatcherServlet에 대한 서블릿 매핑을 추가 (in web.xml)

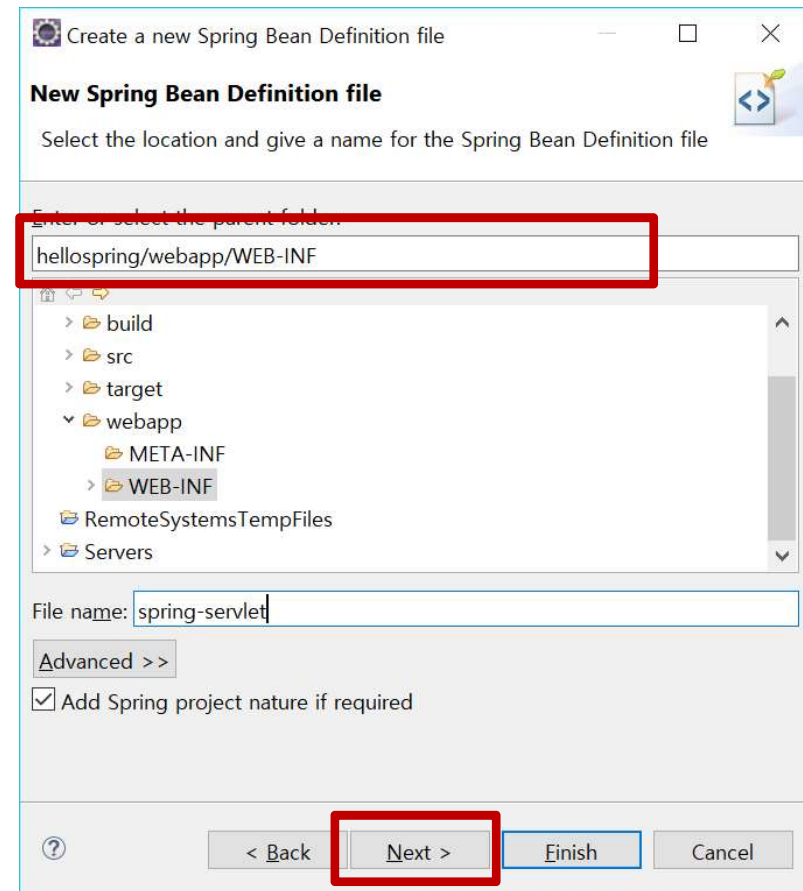
```
<servlet>
  <servlet-name>spring</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>

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

# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

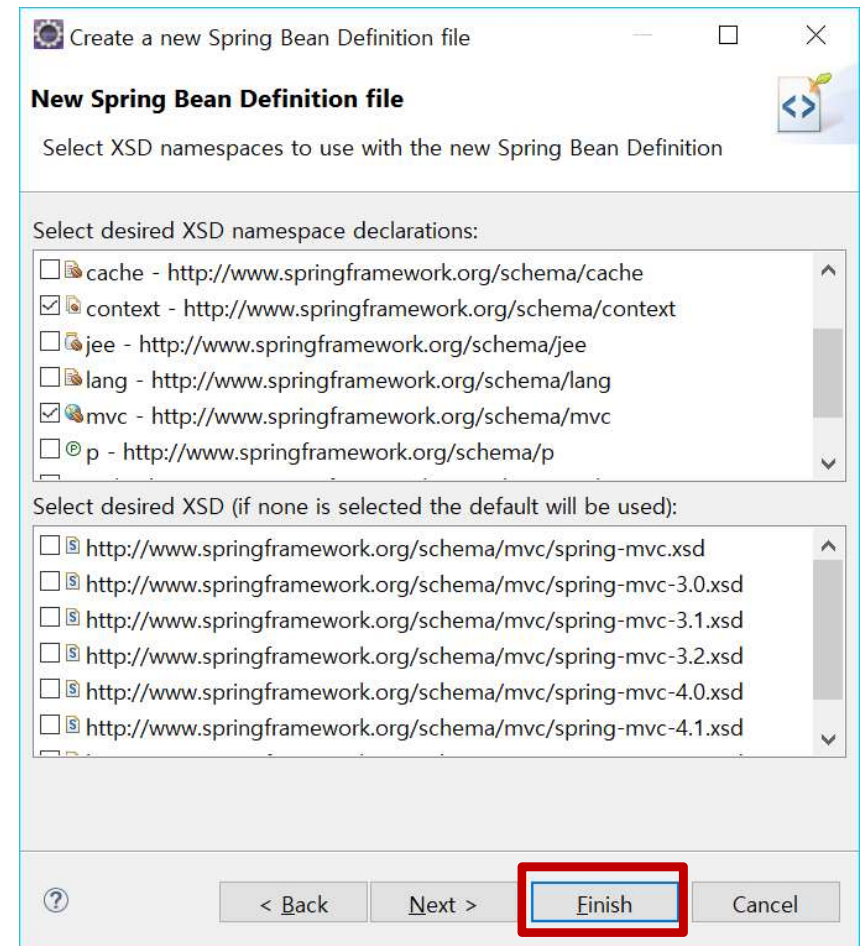
- ▶ 웹 애플리케이션 컨텍스트 설정 (in /WEB-INF/spring-servlet.xml)
  - ▶ webapp - WEB-INF 에서 우클릭 New > Other
    - ▶ Spring Bean Configuration File 선택



# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ 웹 애플리케이션 컨텍스트 설정  
(in /WEB-INF/spring-servlet.xml)
  - ▶ 애플리케이션에서 사용할 XSD 네임스페이스 임포트
  - ▶ 여기서는 다음 XSD들을 선택
    - ▶ beans
    - ▶ context
    - ▶ mvc



# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

- ▶ 웹 애플리케이션 컨텍스트 설정 (in /WEB-INF/spring-servlet.xml)
  - ▶ 컨텍스트 설정

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

# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

### ▶ Controller의 작성

- ▶ HelloController 클래스를 만들어 봅시다

```
package com.example.hellospring.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class HelloController {
    @RequestMapping("/hello")
    public ModelAndView hello(@RequestParam String name) {
        ModelAndView mav = new ModelAndView();

        mav.addObject( "message", "Hello " + name );
        mav.setViewName( "/WEB-INF/views/hello.jsp" );

        return mav;
    }
}
```

# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

### ▶ View jsp의 작성

#### ▶ webapp\WEB-INF\views\hello.jsp

```
...  
<body>  
  
<h1>${message }</h1>  
</body>  
...
```

### ▶ 실행

#### ▶ Run > Run As > Run on Server

#### ▶ 브라우저 주소창에 다음 주소를 입력해 보시다.

#### ▶ `http://localhost:8080/hellospring/hello?name=Spring`



# Spring Web MVC 시작하기

## : hellospring 웹 애플리케이션 작성

### ▶ 실행 및 생각해볼 것 들

#### ▶ 추가적으로 해 준 것들

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

#### ▶ 생략된 것들

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

# Spring Web MVC

## : 실습 과제

- ▶ Dynamic Web Project 기반으로 프로젝트를 생성
  - ▶ Project name : myportal
- ▶ 프로젝트를 Maven Project로 변경
  - ▶ Group Id : com.bitacademy
  - ▶ Artifact Id : myportal
- ▶ 다음의 Dependency를 추가하고 빌드해보기 : 버전은 4.2.3.RELEASE로 통일
  - ▶ spring-context
  - ▶ spring-web
  - ▶ spring-webmvc