# 01_Spring Legacy



## 1. 스프링 프레임워크란?

프레임워크 ( Framework )

- 사전적 의미는 뼈대 혹은 틀로서 소프트웨어 관점에서 접근하면 아키텍처에 해당하는 골격 코드이다.
  → 어플리케이션의 구조를 결정하는 코드를 프레임워크가 제공한다.
- 장점
  1) 빠른 구현 시간
  2) 쉬운 관리
  3) 개발자들의 역량 획일화

1. JDK 버전 관리
    - 프로젝트 우클릭 Properties → Project Facets, Java Compiler 확인
2. Maven의 주 용도(pom.xml)

- 프로젝트에 필요한 의존적인 라이브러리를 자동으로 관리해주는 용도

3. MVC 프로젝트 템플릿의 구조
   - src/main/java 개발되는 java 코드의 경로
   - src/main/resources 서버가 실행될 때 필요한 파일들의 경로
   - src/test/java 테스트 전용 경로
   - src/test/resources 테스트 시에만 사용되는 파일들 경로
   - src/main/webapp/WEB-INF/spring Spring 설정 파일의 경로
   - src/main/webapp/WEB-INF/views JSP 파일 경로
   - pom.xml Maven의 설정 파일
4. 스프링 프레임워크를 사용해야 하는 이유
   - 개발자의 구성에 따라 프로젝트의 결과 차이가 크기 때문에 그 상황을 극복하기 위해 기본 흐름, 구조를 맞춤 그에 따라 개발 시간을 단축
5. 다른 프레임워크와의 차별성

   1) 복잡함에 반기를 들어서 만든 프레임워크
      - 일반적인 Java 클래스, 인터페이스 이용으로 진입장벽 낮음
      - EJB에 비해 가볍기 때문에 빠른 시간에 엔터프라이즈급의 시스템 작성 가능

      > 💡 EJB
      >
      > 서버 측 컴포넌트 모델로, 동시 접속자가 10,000명 이상인 사이트 구축 시 사용하는 기술.
      > 동시 접속자가 많은 가운데 안정적인 트랜잭션이 필요한 경우에 구축

   2) 프로젝트의 전체 구조를 설계할 때 유용한 프레임워크

   3) 다른 프레임워크들의 포용
      - 다른 프레임워크들과의 통합을 지원하여 최소한의 수정 가능
      - 여러 종류의 프레임워크를 혼용해서 사용 가능

   4) 개발 생산성과 개발 도구의 지원
      - 유지보수에 있어서 xml의 설정 등을 사용

- STS, Eclipse 등의 플러그인 빠른 업데이트

## 2. 스프링의 주요 특징

1. POJO 기반의 구성 ( Plain Old Java Object )
    - 자바 객체의 라이프 사이클을 스프링 컨테이너가 직접 관리하며, 스프링 컨테이너로부터 필요한 객체를 얻어올 수 있다.
2. 의존성 주입 DI ( Dependency Injection ) 지원
    - 각 계층이나 서비스들 사이 또는 객체들 사이에 의존성이 존재할 경우 스프링 프레임워크가 서로를 연결시켜준다. 이는 클래스 사이에 약한 결합을 가능케 한다.
    - 생성자, set method, Annotation
3. AOP(Aspect Oriented Programming) 지원
    - 트랜잭션, 로깅, 보안 등 여러 모듈에서 공통적으로 지원하는 기능을 분리하여 사용 가능(반복 코드 x)
4. 확장성이 높다.

2. 개발 환경 및 설치

1. 개발 툴
    - Java Framework를 사용하기 툴은 크게 Eclipse, STS, IntelliJ 정도가 있고 각각의 특징은 아래와 같다.

        1) **Eclipse**

        Market Place에서 플러그인을 추가해서 사용해야 하며, Legacy프로젝트는 현재 버전 문제로 설치가 불가하다.

        2) **STS ( Spring Tool Suite )**

        스프링 프레임워크 전용 툴로써 레거시 방식과 부트 둘 다 사용 가능하나, 4버전부터는 레거시를 사용할 수 없다.

> 💡 이클립스를 기반으로 작성된 툴이기 때문에 UI 및 기능이 똑같다.
> ( 자바 및 웹 개발 가능 )

3) **intelliJ**

지원해주는 기능이 강력하나, 많은 기능들을 사용하려면 금액을 지불해야하는 유료 라는 점이 단점이다.

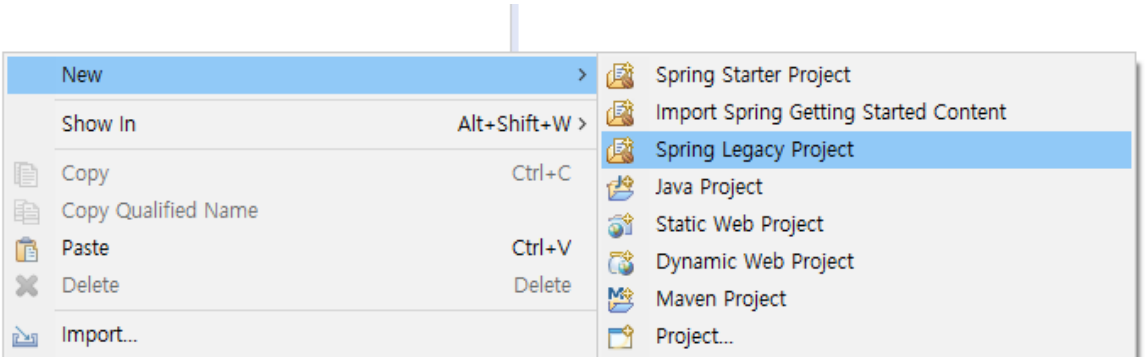
2. WAS
   - 아파치-톰캣을 사용할 것이나, 부트는 톰캣이 내장되어있어 추가적으로 설치 불필요

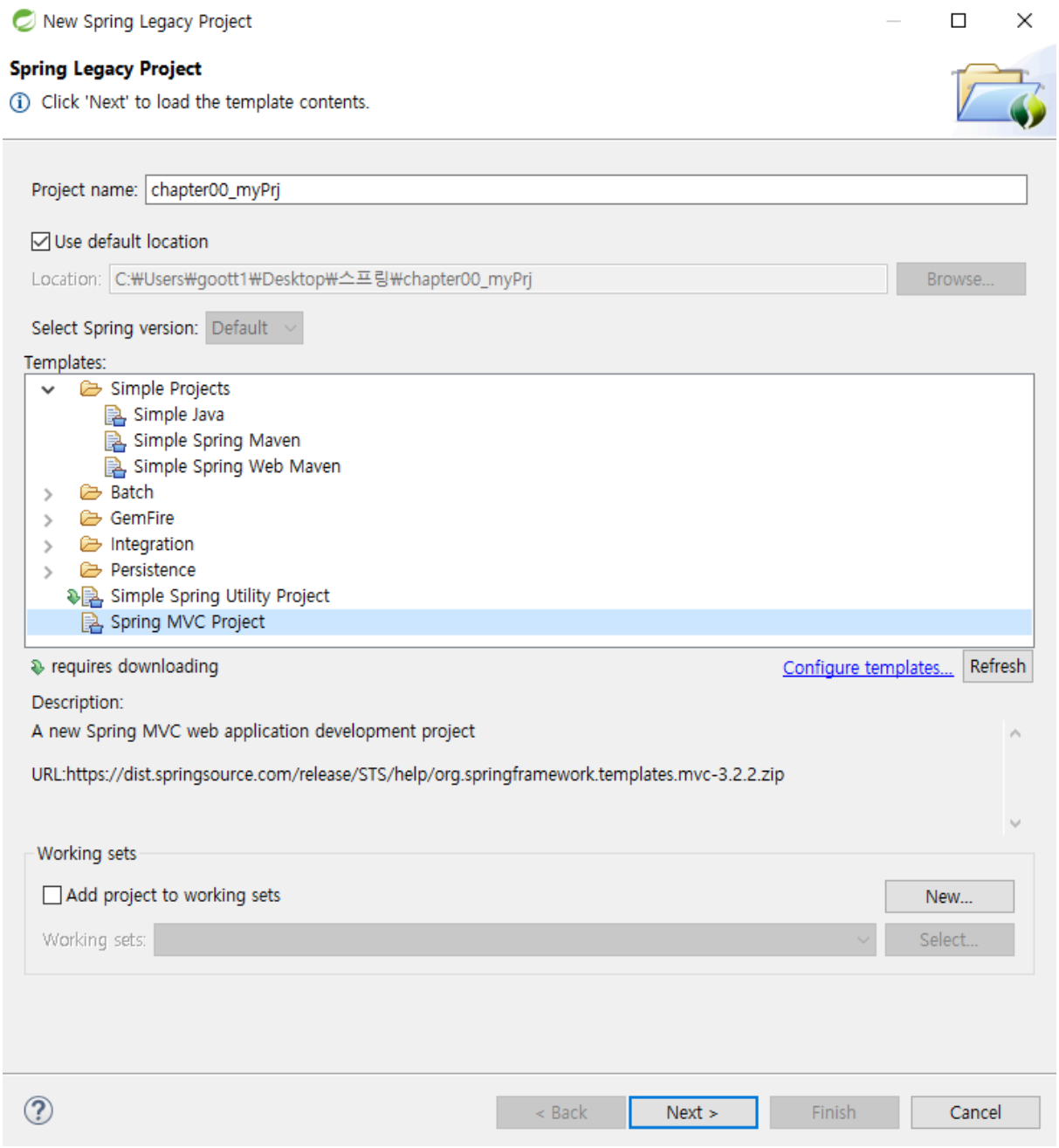
3. DB
   - Oracle, MySQL, MariaDB 등 RDB 위주로 개발진행

## 3. 프로젝트 ( Legacy ) 만들기

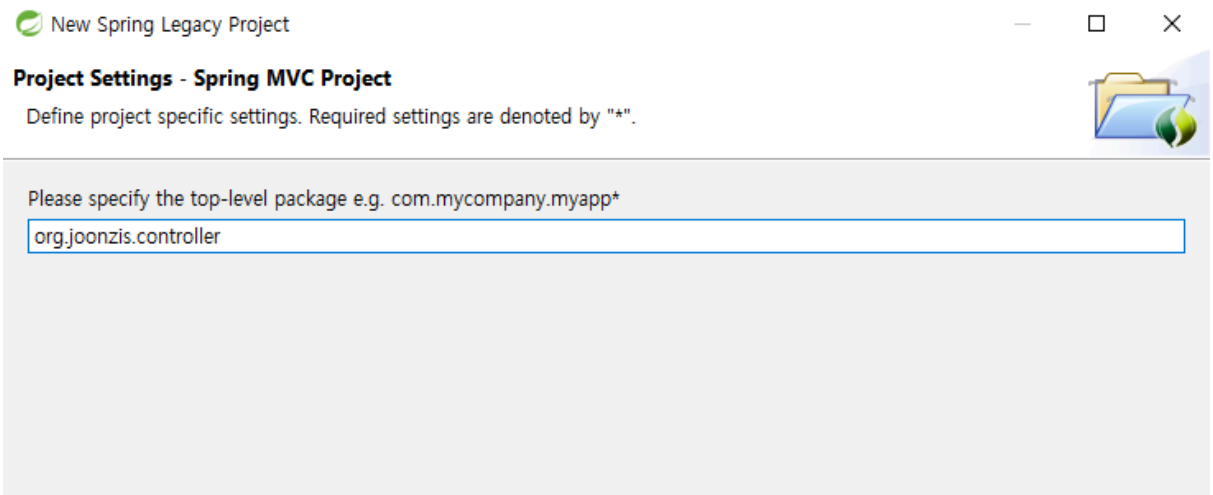
스프링은 크게 레거시 버전과 부트 버전으로 나뉜다.

이번 내용에서는 레거시 버전을 다룰 예정이다.






프로젝트명을 입력하고, 아래에 생성할 프로젝트의 구조(MVC)를 선택한 뒤 다음으로 넘어간다.




기본 패키지를 설정하는 부분으로 3단계 이상의 패키지를 작성하는 곳이다.

마지막 경로 ( controller ) 가 컨텍스트 패스가 되니 신경 써주도록 한다.

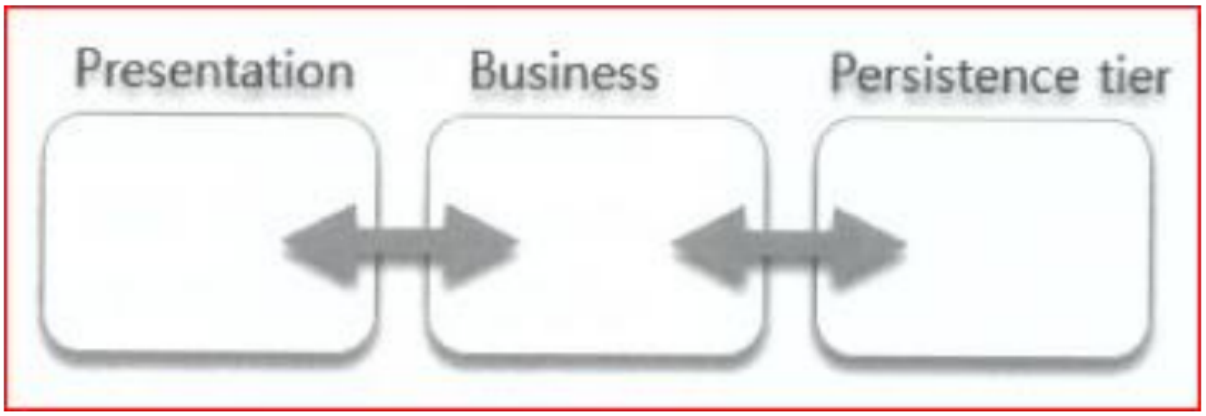
# 4. 스프링의 구조

## 중요하게 고려해야 하는 부분

1. 스프링 MVC를 이용하는 웹 프로젝트 전체 구조에 대한 이해.
2. 개발의 각 단계에 필요한 설정 및 테스트 환경
3. 기본적인 CRUD 구현 ( 리스트, 페이징, 검색 등등)

## 스프링 MVC의 기본 구성

일반적인 웹 프로젝트 구성 : 3-tier 방식




1. **Presentation ( 화면 계층 )**

   화면에 보여주는 기술을 사용하는 영역

   Servelt / JSP이나 스프링 MVC가 담당하는 영역

   프로젝트의 성격에 맞추어 앱으로 제작하거나 CS(Client-Server)로 구성될 수도 있음

   스프링 MVC와 JSP를 이용한 화면 구성이 이에 속함

2. **Business ( 비즈니스 계층 )**

   순수 비즈니스 로직 담당 영역

   주로 xxxService 파일이 담당

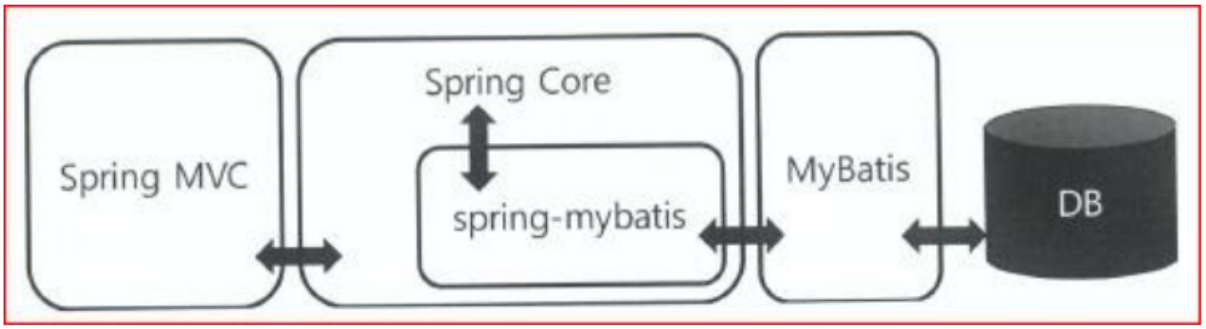
3. **Persistence ( 영속 or 데이터 계층 )**

   데이터를 어떤 방식으로 보관, 사용할지 설계가 들어가는 영역

   경우에 따라 네트워크나 원격 호출 기술이 접목될 수 있음

   Mybatis와 mybatis-spring 을 이용

## 스프링 MVC와 Mybatis 구조




1. **Spring MVC**

   Presentation Tier 구성

   root-context.xml, servlet-context.xml 등의 설정 파일이 해당 영역의 설정을 담당

2. **Spring Core**

   POJO ( Plain-Old-Java-Object ) 영역

   스프링의 의존성 주입을 이용해서 객체 간의 연관 구조를 완성하여 사용

3. **MyBatis**

   SQL 처리를 담당하는 구조

## 프로젝트 및 파일 네이밍 규칙

1. **Controller**
   - 스프링 MVC에서 동작하는 Controller 클래스
2. **Service, ServiceImpl**
   - 비즈니스 영역을 담당하는 인터페이스와 구현 클래스
3. **DAO or Repository**
   - DAO ( Data-Access-Object ) 나 Repository( 저장소 ) 이름으로 영역을 따로 구성하는 것이 보편적,
   - 별도의 DAO를 구성하는 대신 Mybatis의 Mapper 인터페이스를 활용
4. **VO or DTO**

- VO 나 DTO는 일반적으로 유사한 의미로 사용하며 데이터를 담는 객체를 의미

## 프로젝트 구조 및 요소들의 특징