61일차 강의

* ( 숫자만 입력 ) 랜덤추천학생(버튼) 전체학생추가(버튼)

(화면출력결과)

* 글자 입력 시 – 전체 학생 수만 표시
* 숫자 입력 시 – 번호: 7 / 이름: 윤승환 (주변에 테두리)

남은 학생수: 8명

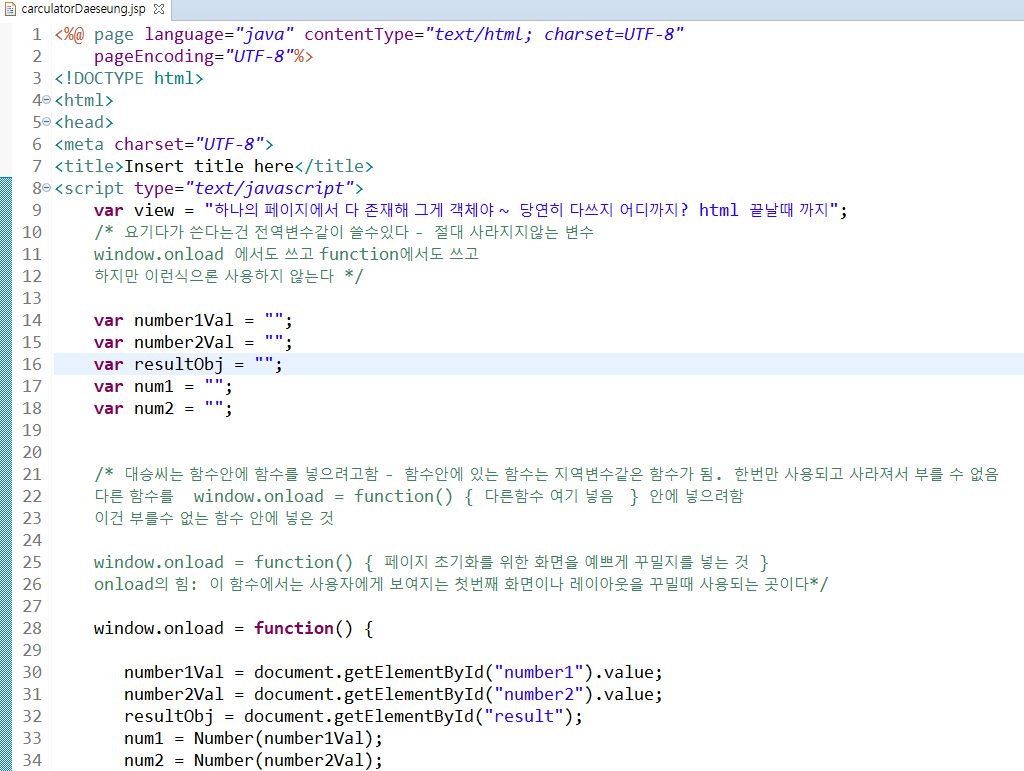
* 등록은- 전체 학생 추가 버튼을 통해서
* 랜덤추천학생을 1명 사용했다면 , 그 이후 다시 랜덤추천학생 버튼을 클릭 시 이미 당첨된 학생은 제외한 상태에서 랜덤추천이 이루어지고 남은 학생수도 표현된다
* 분리해야한다
* 사용자가 입력하는 영역
* 사용자가 입력한 사항에 대해 결과를 보는 영역
* 사용자들은 태그 안의 내용을 보는 것
* 출력에 관련된 태그를 만들어야 한다
* 사람을 이름을 입력하고 추가버튼을 누르면 추가되도록
* 계산기(+,-,\*,/)를 화면단 만 써서 만들자
* Window 라는 건 브라우저도 그 자체????(뭔말이야)

Window.onload 를 걸었다는 것은

새로고침하면 깜박깜박 거리는 그 이벤트가 onload 이벤트이다

모드 내용을 읽고 사용자에게 보여지기 전에 딱 한번 수행되는 함수가 window.onload

* 처음이자 마지막으로 딱 한번 수행됨
* 마치 생성자 같음
* 이미 다른 내용을 읽은 상태이므로 데이터베이스(EL태그)든 뭐든 불러서 사용가능 하다



* Spring 폴더 구조
  + Webapp
* webcontent 같은 역할
* jsp에 관련된 내용 관리는 여기서한다
  + Target
* Maven에 있는 build tool을 관리한다
* 자동적으로 관리가 되게해줌 (뭘?)
* Pom.xml을 다르면 maven이 알아서 작동이 됨.
* 프로젝트명 : hello-web (영상 4시 13분쯤 시작)
* Hello-web 이란 이름으로 다이나믹 프로젝트 – new
* 원래 존재하는 src -> remove
* src/main/java

src/main/java

src/main/java

src/main/java

4개를 추가한다

* next 누르고

cpontent directory 에 WebContent 를 지운다

cpontent directory 에 src/main/webapp 를 넣어준다

* 구조를 확인하고 스프링으로 변환해준다

프로젝트명 우클릭

Configure -> convert to Maven project

* Mave POM
  + Group id

: 팀 아이디, 프로젝트 관련 아이디

절대 겹치지 않는 자기만의 주소를 가져야한다

최소 3레벨 , 4레벨을 권장

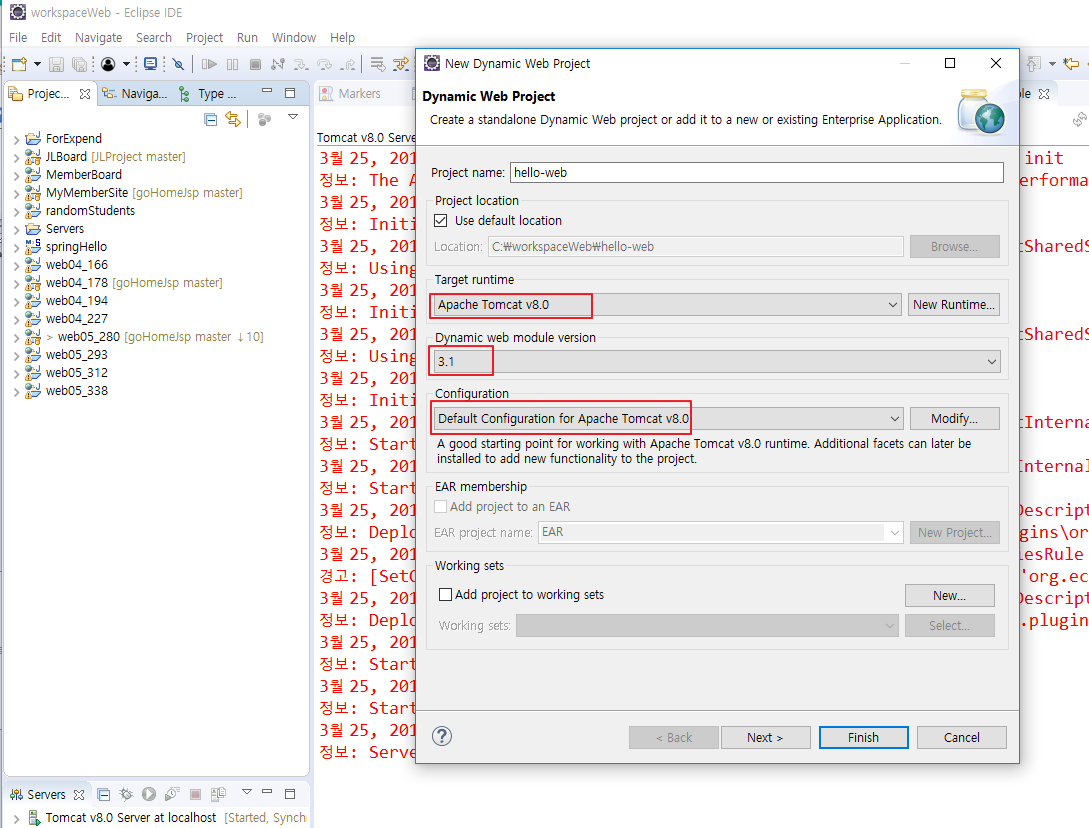
Ex) com.tg.hello

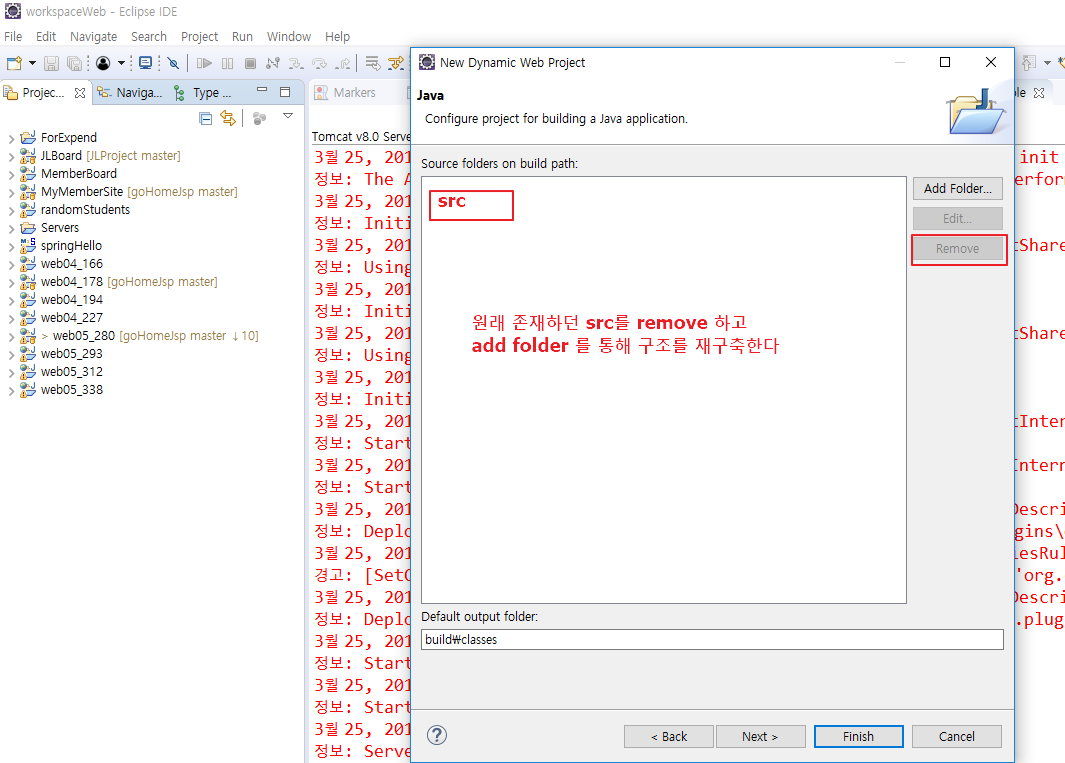
* + Artifact id:

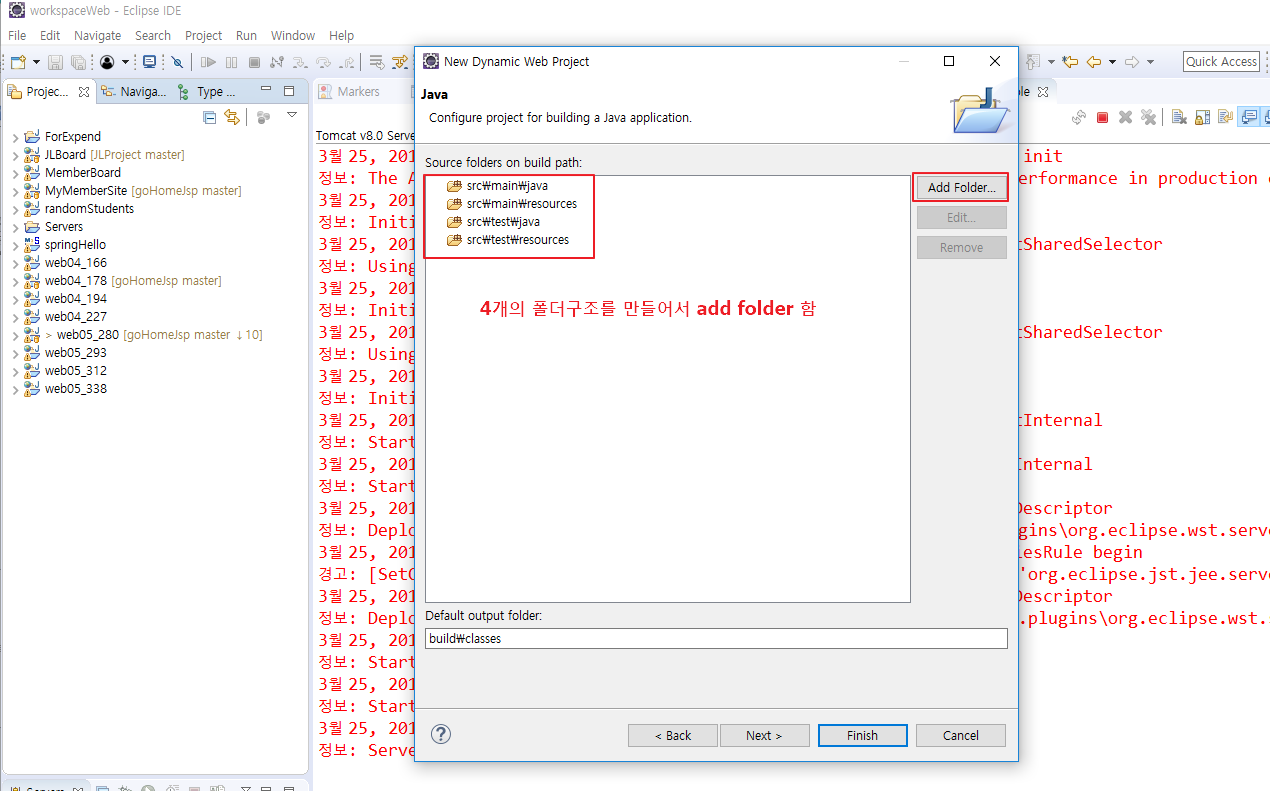
: 프로젝트명을 사용한다 달라도 무방

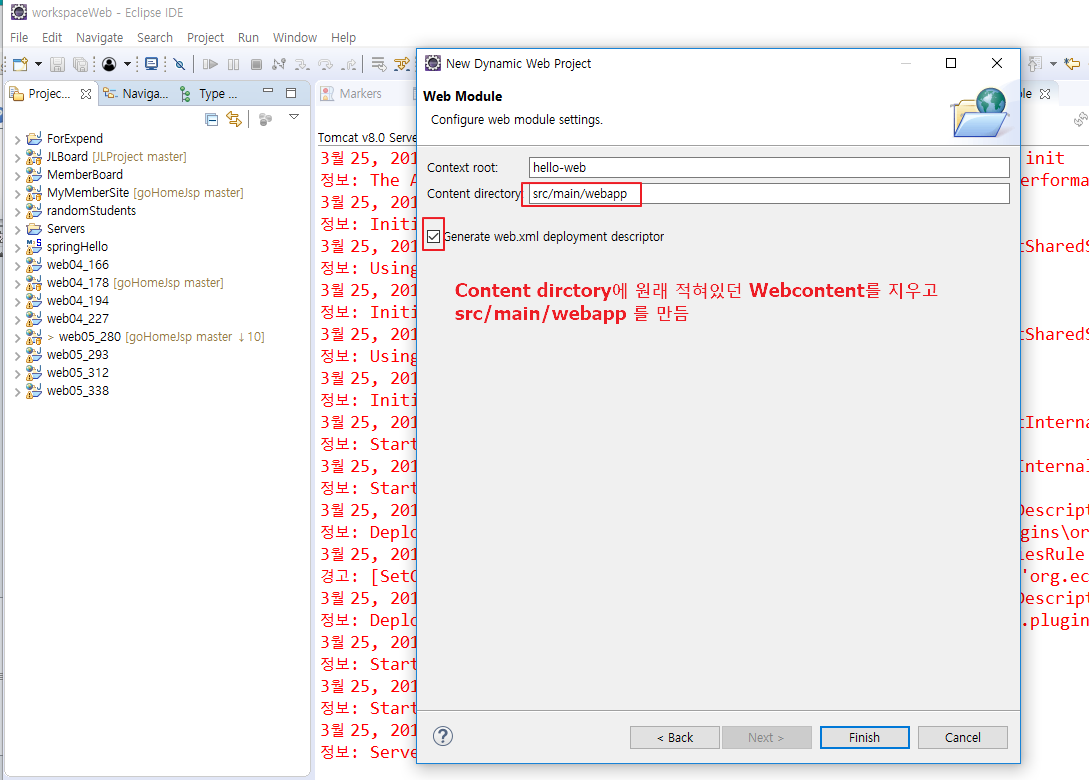
* + Packaging : war

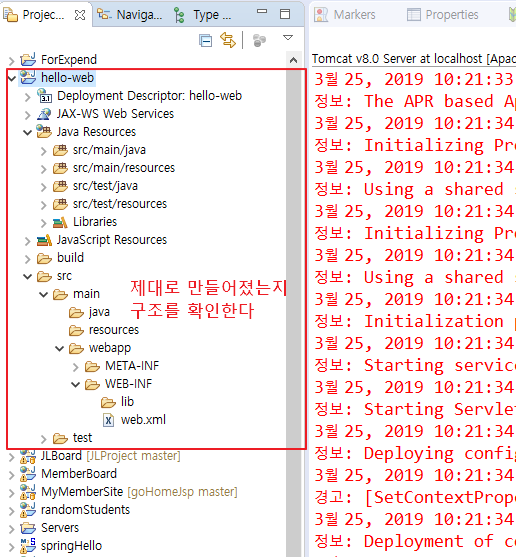
: web을 만드니 war 로 해준다

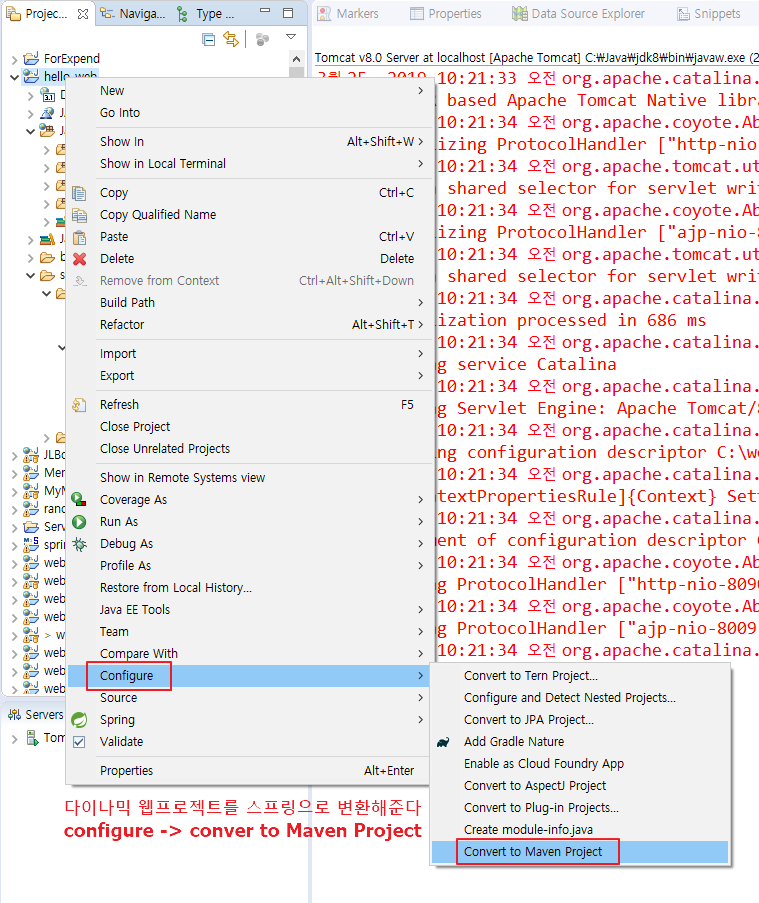


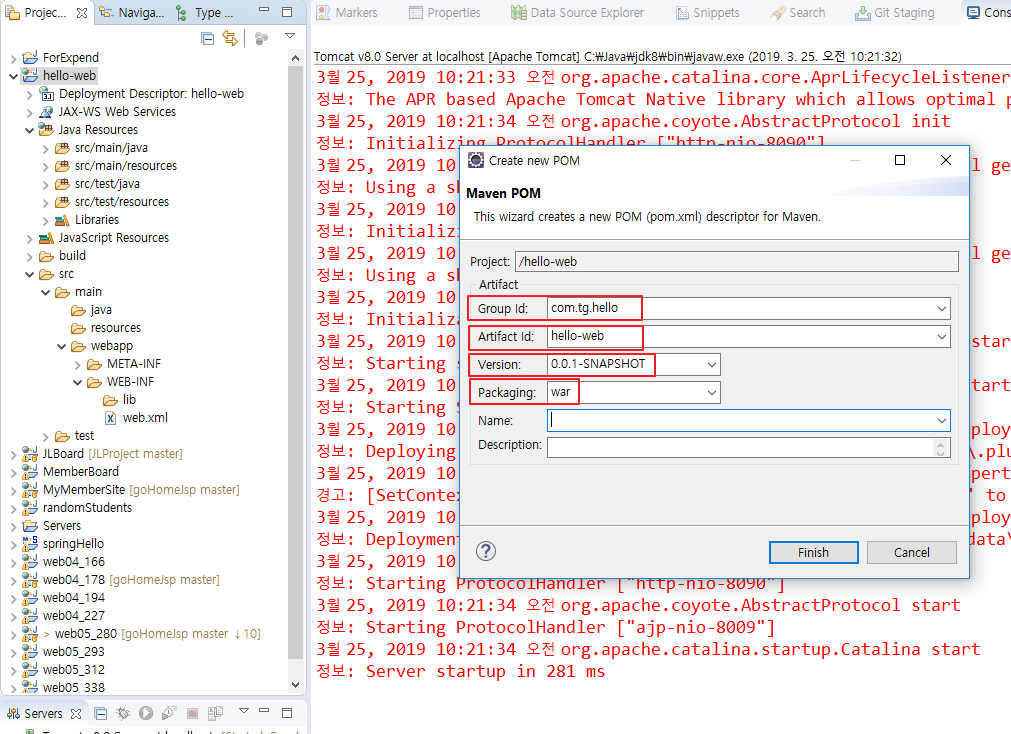


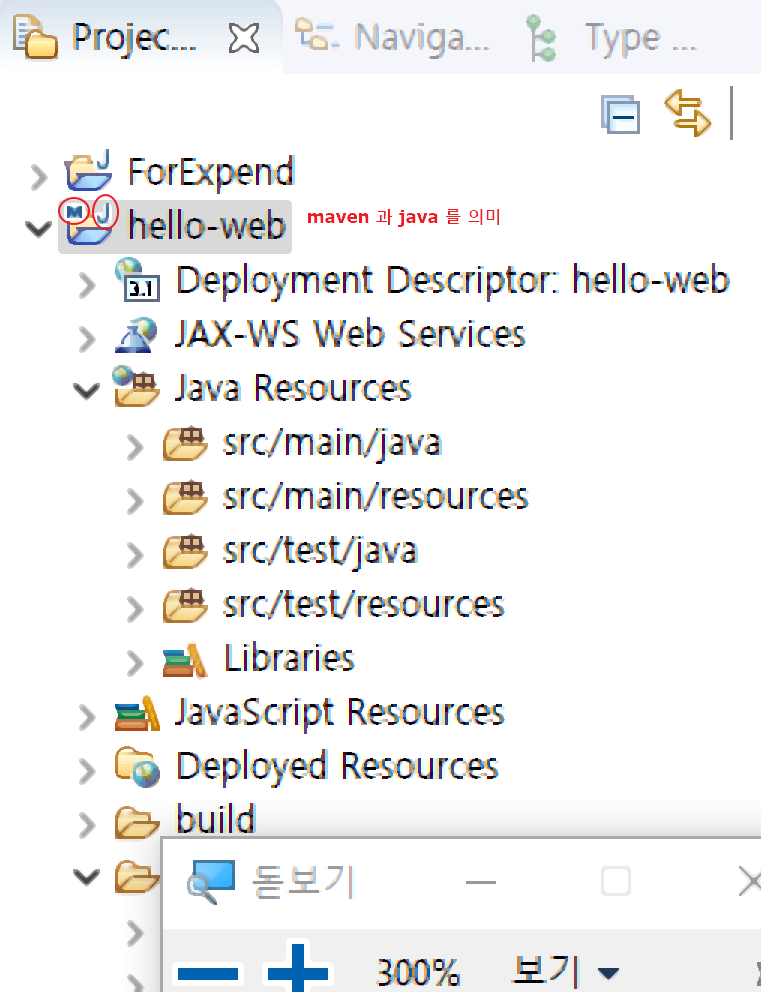












* Maven 을 pom.xml 을 통해서 다룬다.
* 메일, 메시지, 파일전송 문자처리(fmt), 등

이미 존재하는 기능들을 다양하게 메이븐을 통해서 들고온다(오픈소스이기에 마음대로 다운받아 사용가능 한 것)

* Pom.xml 안의

</build>

<dependencies>

이 사이에서 사용할 것을 추가해 준다

</dependencies>

</project>

<https://mvnrepository.com//> : MVN repository

위의 사이트로 들어가서

검색 : spring context 을 한다

Spring context 클릭

Dependency Injection : 을 통해

4.3.9 release 선택

* Dependency injection : 의존 주입

객체 주입

* Class 그자체를 Bean 이라고 부른다

특히 객체를 다루는 그자체를 클래스 대신 빈이라고 부름

* 가장 좋은 사례를 모아서 만든게
* spring-context.xml (설정에 관련된건 .xml)
* beans 를 통해 bean 을 관리한다

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

여기서 bean 을 관리한다

</beans>

* 아파치 **메이븐**(Apache **Maven**)은 자바용 프로젝트 관리 툴이다
* **[ Spring 스프링? ]**
  + EJB를 주 프레임워크로 사용할 때 불편했던 점들을 해소

(EJB : Enterprise Java Been 이전에 존재했던 프레임워크 – 불편한 점이 존재: 하나의 사항을 처리하기위해 무조건 상속을 받아야해서 무거움, 한줄을 처리해도 5~6줄을 가져와야 했음)

* + Apache2.0 License로 공개됨
  + 로드 존슨이 출간한 도서의 코드가 Spring의 근간이 됨
* **스프링 프레임워크(Spring framework)란?**
  + Java 엔터프라이즈 개발을 편하게 해주는 오픈 소스

경량급 애플리케이션 프레임워크

* + 애플리케이션 프레임워크

특정 계층이나 기술, 업무 분야에 국한되지 않고 애플리케이션의 전 영역을 포괄하는 범용적인 프레임워크를 말함

* + 경량급 프레임워크

단순한 웹 컨테이너에서도 엔터프라이즈 개발의 고급기술을 대부분 사용할 수 있음

* + 장점
* 엔터프라이즈 개발 용이

: 개발자가 복잡하고 실수하기 쉬운 Low Level에 많이 신경 쓰지 않으면서 Business Logic 개발에 전념할 수 있도록 해줌

* 오픈소스

: Spring은 OpenSource의 장점을 충분히 취하면서 동시에 OpenSource 제품의 단점과 한계를 잘 극복함.

**[ Spring Franwork의 특징 ]**

1. **컨테이너 역할**

: Spring 컨테이너는 Java 객체의 Life Cycle을 관리하며,

Spring 컨테이너로부터 필요한 객체를 가져와 사용할 수 있다.

1. **DI(Dependency Injection) 지원 (의존 주입 지원)**

: Spring은 설정 파일이나 어노테이션을 통해서 객체 간의 의존관계를 설정할 수 있도록 하고 있다.

(상속은 무겁고 쓸모없는것도 받게되고 확장된다는 것만이 좋은것이 아니다.

대신 DI를 통한 의존관계를 설정하면 (has), 필요한 것만 가져와서 손쉽게 객체를 사용이 가능하다

결합도는 낮추고 응집도는 높혀라

기능의 분리를 지원한다 – 객체지향을 알아서 해줌)

1. **AOP (Aspect Oriented Programming) 지원**

: Spring은 트랜잭션이나 로깅, 보안과 같이 공통적으로 필요로 하는 모듈들을 실제 핵심 모듈에서 분리해서 적용할 수 있다.

1. **POJO (Plain Old Java Object) 지원**

: Spring 컨테이너에 저장되는 Java 객체는 특정한 인터페이스를 구현하거나, 특정 클래스를 상속받지 않아도 된다.

( 경량화의 핵심,

앞으로 자바 코드 짜듯이 짜면 pojo에서 알아서 상속을 관리해준다)

1. **트랜잭션 처리를 위한 일관된 방법을 지원**

: JDBC, JPA 등 어떤 트랜잭션을 사용하던 설정을 통해 정보를 관리하므로 트랜잭션 구현에 상관없이 동일한 코드 사용 가능

( 이전엔 다양한 라이브러리에 따라 코드가 다 다르게 수정되어져야했다면, 이제는 라이브러리만 바꿔주면되지 코드를 수정할필요는 없게된다 – 확인필요?

일관된 코드를 작성하면 끝남)

1. **영속성(Persistence)과 관련된 다양한 API 지원**

: Spring은 Mybatis, Hibernate 등 데이터베이스 처리를 위한 ORM(Object Relational Mapping) 프레임워크들과의 연동 지원

( 영속성은 데이터베이스 말고는 존재할 수 없다.

JDBC 대신 Mybais를 사용해볼 계획이다.

Hibernate – ORM(객체관계맵핑) : 자바의 객체와 데이터베이스의 테이블을 연결 -> 자바의 객체를 지우면 테이터베이스의 테이블이 삭제되고, 테이블을 삭제하면 객체가 사라진다.

맵핑되어 있으므로 Select 이나 insert 할 필요가 없어진다는 의미)

* **IoC ( Inversion of Control)의 개념**

: **제어의 역전**이란, 객체의 생성, 생명주기의 관리까지 모든 객체에 대한 제어권이 바뀌었다는 것을 의미

( 이전에는 개발자의 의지에 의해 new 해서 만들고, null 값 넣고, destroy 했는데, 개발자가 만들어서 시스템에 적용시켰다면, 이제는 뒤바뀌어 시스템이 만들어놓고 관리하므로, 개발자는 이제 가져다가 사용만 하게 된다

– 스프링 컨테이너가 이것을 관리해준다)

* IoC 컨테이너

: 스프링 프레임워크도 객체에 대한 생성 및 생명주기를 관리할 수 있는 기능을 제공하고 있음.

즉, IoC 컨테이너 기능을 제공한다.

* + 특징

: IoC 컨테이너는 객체의 생성을 책임지고, 의존성을 관리한다.

POJO의 생성, 초기화, 서비스, 소멸에 대한 권한을 가진다.

개발자들이 직접 POJO를 생성할 수 있지만 컨테이너에게 맡긴다.

* 스프링 컨테이너가 IoC컨테이너의 기능을 제공해준다

: 이것이 컨테이너의 힘!!!!

(이전에도 다른 IoC컨테이너의 기능을 제공해주는 것들은 존재했음. 단지 스프링 컨테이너가 더 좋은 것)

* <bean> 클래스 그 자체가 bean이 된다

빈에 등록하면

* <beans> 는 컨테이너로서(스프링컨테이너로 친다. 실제는 좀 다르지만 managed bean)

<bean>을 관리해준다.

* Spring-context..xml에 <bean>을 등록하면,

파일에 스프링을 알리는 S 가 붙고

getBean 하면 메인에서 클래스를 가져와서 쓸수있다

( 더 이상 직접 new해서 만들 필요가 없는 것)

의존주입 하였으므로

* DTO (Data Transfer Object) 를 주로 사용함

이제 VO 도 사용해볼것임