# Spring Framework

# 웹 어플리케이션 개발 01

강경미(carami@nate.com) & 김성박(urstory@gmail.com)

# 무엇이 개발자를 괴롭히는가?

- 수천라인의 코드
- 고구마 줄기처럼 파도파도 끝이 없는 코드들
- 같은 기능의 개발
- 유지보수

# 개발자의 고민

- 좀 더 나은 코드
- 좀 더 나은 성능

항상 프레임워크 기반으로 접근하라

로드 존슨

2003 년 *Expert One-on-One J2EE Design and Development* 출판
*J2EE 설계와 개발의 모든 영역에 대한 개발 전략을 다룬 책*
*이 책에 포함된 3만 라인의 샘플 애플리케이션 예제에 포함된 프레임워크가*
*스프링 프레임워크의 기원*

*엔터프라이즈 어플리케이션을 만들기 위한 기반이 되는 코드 -*
*스프링의 기원*

# Spring Core

# Spring 등장배경

Rod Johnson

*EJB* 개구려서 내가 이번에
*'EJB*안쓰고 개발하기 ' 책 냈으니
홍아들 한 번만 봐주삼~

오! 대박~ 이생퀴
잘만들었는데?
이거 오픈소스로
함 만들어보까?

Juergen Hoeller

Yann Caroff

ㅇㅇ~
*EJB*가 개겨울 같았으니까
이름을
'봄 ' 으로 하는게 어때?

# Spring Framework란?

- 엔터프라이즈 급 어플리케이션을 구축할 수 있는 가벼운 솔루션이자, 원스-스탑-숍(One-Stop-Shop)
- 원하는 부분만 가져다 사용할 수 있도록 모듈화가 잘 되어 있다.
- POJO를 이용한 가볍고 non-invasive(비 침투적) 개발
- IoC 컨테이너이다.
- DI와 인터페이스 지향을 통한 느슨한 결합도
- 선언적으로 트랜잭션을 관리할 수 있다.
- 완전한 기능을 갖춘 MVC Framework를 제공한다.
- AOP와 공통 규약을 통한 선언적 프로그래밍
- 템플릿을 통한 상투적인 코드 축소
- 스프링은 도메인 논리 코드와 쉽게 분리될 수 있는 구조를 가지고 있다.

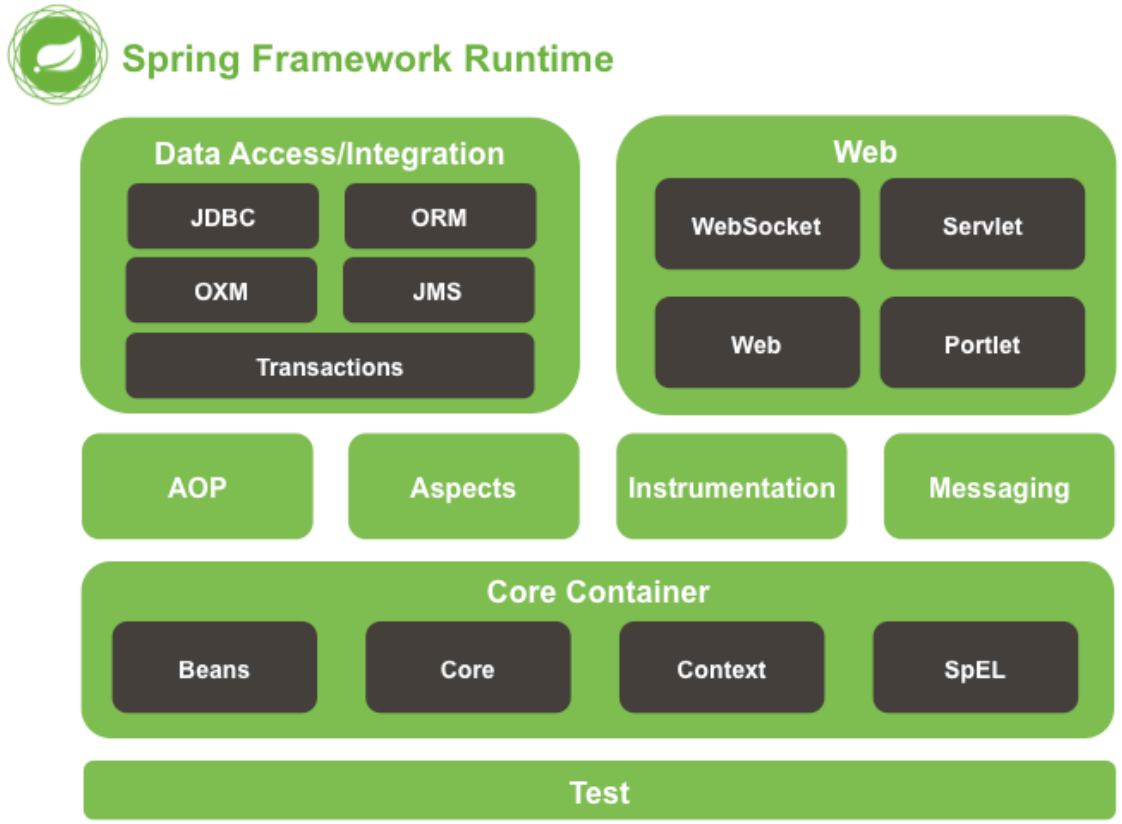
* 원-스탑-숍 (One-Stop-Shop) : 모든 과정을 한꺼번에 해결하는 상점.

# Release History

- 1.0 : 2004년 4월
- 2.0 : 2006년 6월
- 2.5 : 2007년 11월
- 3.0 : 2011년 12월
- 3.1.4 : 2013년 1월
- 4.0.1 : 2014년 1월
- 5.0 : 2017년 9월

- 2003년 6월 아파치 2.0 라이센스로 공개

# 프레임 워크 모듈




- 스프링 프레임워크는 약 20개의 모듈로 구성되어 있다.
- 필요한 모듈만 가져다 사용할 수 있다.

# AOP 와 인스트루멘테이션(Instrumentation)

- spring-AOP : AOP 얼라이언스(Aliance)와 호환되는 방법으로 AOP를 지원한다.
- spring-aspects : AspectJ와의 통합을 제공한다.
- spring-instrument : 인스트루멘테이션을 지원하는 클래스와 특정 WAS에서 사용하는 클래스로더 구현체를 제공한다. 참고로 BCI(Byte Code Instrumentations)은 런타임이나 로드(Load) 때 클래스의 바이트 코드에 변경을 가하는 방법을 말합니다.

# 메시징(Messaging)

- spring-messaging : 스프링 프레임워크 4는 메시지 기반 어플리케이션을 작성할 수 있는 Message, MessageChannel, MessageHandler 등을 제공한다. 또한, 해당 모듈에는 메소드에 메시지를 맵핑하기 위한 어노테이션도 포함되어 있으며, Spring MVC 어노테이션과 유사하다.

# 데이터 엑세스(Data Access) / 통합 (Integration)

- 데이터 엑세스/통합 계층은 JDBC, ORM, OXM, JMS 및 트랜잭션 모듈로 구성되어 있다.
- spring-jdbc : 자바 JDBC프로그래밍을 쉽게 할 수 있도록 기능을 제공한다.
- spring-tx : 선언적 트랜잭션 관리를 할 수 있는 기능을 제공한다.
- spring-orm : JPA, JDO및 Hibernate를 포함한 ORM API를 위한 통합 레이어를 제공한다.
- spring-oxm : JAXB, Castor, XMLBeans, JiBX 및 XStream과 같은 Object/XML 맵핑을 지원한다.
- spring-jms : 메시지 생성(producing) 및 사용(consuming)을 위한 기능을 제공. Spring Framework 4.1 부터 spring-messaging모듈과의 통합을 제공한다.

# 웹(Web)

- 웹 계층은 spring-web, spring-webmvc, spring-websocket, spring-webmvc-portlet 모듈로 구성된다.
- spring-web : 멀티 파트 파일 업로드, 서블릿 리스너 등 웹 지향 통합 기능을 제공한다. HTTP클라이언트와 Spring의 원격 지원을 위한 웹 관련 부분을 제공한다.
- spring-webmvc : Web-Servlet모듈이라고도 불리며, Spring MVC및 REST 웹 서비스 구현을 포함한다.
- spring-websocket : 웹 소켓을 지원한다.
- spring-webmvc-portlet : 포틀릿 환경에서 사용할 MVC구현을 제공한다.

# Spring Framework sub-projects

- Spring IO Platform - 스프링 프레임워크 의존성 관리
- Spring Boot
- Spring Framework
- Spring Cloud Data Flow - Big Data
- Spring Cloud - 클라우드 기반
- Spring Data - JPA, MongoDB, Redis, Elasticsearch ...
- Spring Integration - Enterprise Integration Pattern
- Spring Batch
- Spring Security
- Spring HATEOAS
- Spring Social
- Spring AMQP
- Spring Mobile

# 컨테이너(Container)란?

- 컨테이너는 인스턴스의 생명주기를 관리한다.
- 생성된 인스턴스들에게 추가적인 기능을 제공한다.

# IoC란?

- IoC란 Inversion of Control의 약어이다. inversion은 사전적 의미로는 '도치, 역전'이다. 보통 IoC를 제어의 역전이라고 번역한다.
- 개발자는 프로그램의 흐름을 제어하는 코드를 작성한다. 그런데, 이 흐름의 제어를 개발자가 하는 것이 아니라 다른 프로그램이 그 흐름을 제어하는 것을 IoC라고 말한다.

# POJO

- POJO = Plain Old Java Object
- https://www.martinfowler.com/bliki/POJO.html

# DI란?

- DI는 Dependency Injection의 약자로, 의존성 주입이란 뜻을 가지고 있다.
- DI는 클래스 사이의 의존 관계를 빈(Bean) 설정 정보를 바탕으로 컨테이너가 자동으로 연결해주는 것을 말한다.

# DI 예

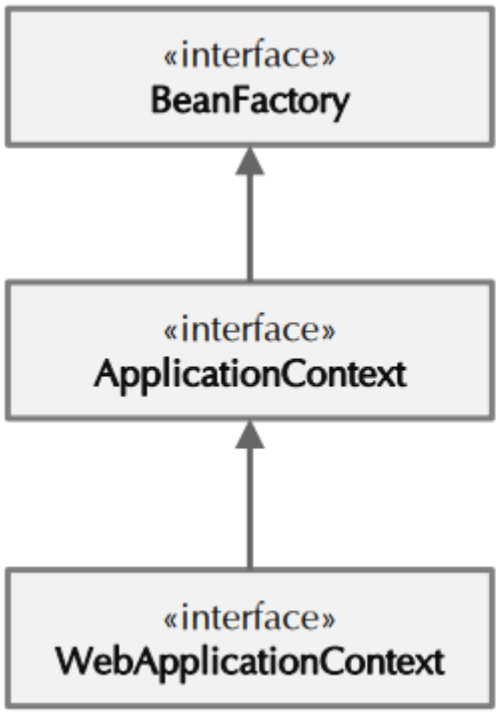
- DI가 적용 안 된 예.
- 개발자가 직접 인스턴스를 생성한다.

```
class 엔진 {

}


class 자동차 {

        엔진 v5 = new 엔진()

}
```

- Spring에서 DI가 적용된 예.
- 엔진 type의 v5변수에 아직 인스턴스가 할당되지 않았다.
- 컨테이너가 v5변수에 인스턴스를 할당해주게 된다.

```
@Component

class 엔진 {

}

@Component

class 자동차 {

    @Autowired

    엔진 v5;

}
```

# IoC 컨테이너




- 빈(bean) 객체에 대한 생성과 제공을 담당
- 단일 유형의 객체를 생성하는 것이 아니라, 여러 유형의 빈(bean)을 생성, 제공
- 객체 간의 연관 관계를 설정, 클라이언트의 요청 시 빈을 생성
- 빈의 라이프 사이클을 관리

- BeanFactory가 제공하는 모든 기능 제공
- 엔터프라이즈 애플리케이션을 개발하는데 필요한 여러 기능을 추가함
- I18N, 리소스 로딩, 이벤트 발생 및 통지
- 컨테이너 생성 시 모든 빈 정보를 메모리에 로딩함

- 웹 환경에서 사용할 때 필요한 기능이 추가된 애플리케이션 컨텍스트
- 가장 많이 사용하며, 특히 XmlWebApplicationContext를 가장 많이 사용

# ApplicationContext

- AnnotationConfigApplicationContext – 하나 이상의 Java Config 클래스에서 스프링 애플리케이션 컨텍스트를 로딩
- AnnotationConfigWebApplicationContext – 하나 이상의 Java Config 클래스에서 웹 애플리케이션 컨텍스트를 로딩
- ClassPathXmlApplicationContext – 클래스패스에 위치한 xml파일에서 컨텍스트를 로딩
- FileSystemXmlApplicationContext – 파일 시스템에서 지정된 xml파일에서 컨텍스트를 로딩
- XmlWebApplicationContext – 웹 애플리케이션에 포함된 xml파일에서 컨텍스트를 로딩

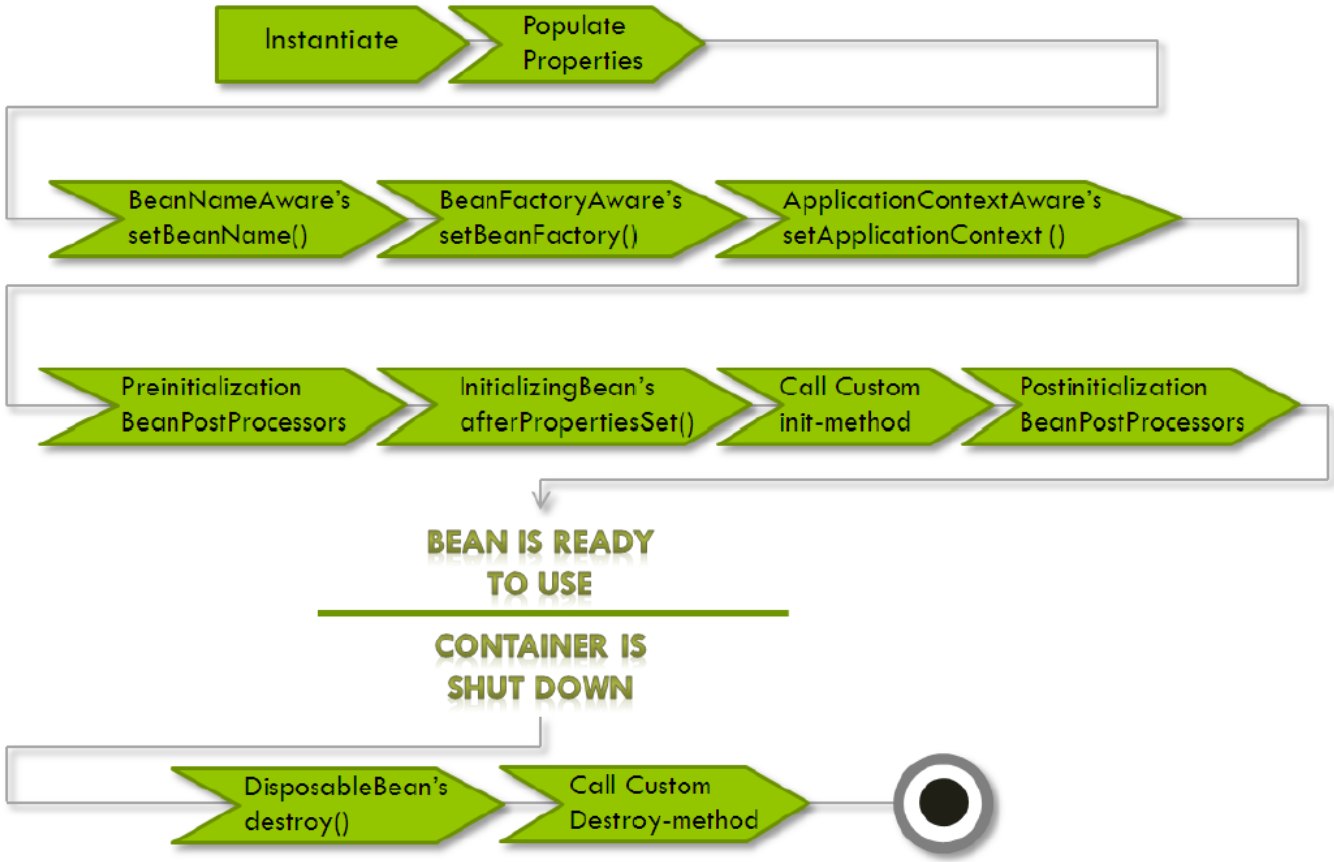
# 사용법

```
ApplicationContext context = new
FileSystemXmlApplicationContext("/spring/context.xml");
```

```
ApplicationContext context = new
ClassPathXmlApplicationContext("context.xml");
```

```
ApplicationContext context = new
AnnotationConfigApplicationContext(Config.class);
```

# Bean의 일생




1. 스프링이 빈을 인스턴스화 한다.
2. 스프링이 값과 빈의 레퍼런스를 빈의 프로퍼티로 주입한다.
3. 빈이 BeanNameAware를 구현하면 스프링이 빈의 ID를 setBeanName()메소드에 넘긴다.
4. 빈이 BeanFactoryAware를 구현하면 setBeanFactory()메소드를 호출하여 빈 팩토리 전체를 넘긴다.
5. 빈이 ApplicationContext를 구현하면 스프링이 setApplicationContext()메소드를 호출하고 둘러싼 애플리케이션 컨텍스트에 대한 참조를 넘긴다.
6. 빈이 Bean PostProcessor인터페이스를 구현하면 스프링은 postProcessBeforeInitialization()메소드를 호출한다.
7. 빈이 InitializingBean인터페이스를 구현하면 스프링은 afterPropertiesSet()메소드를 호출한다. 마찬가지로 빈이 init-method와 함께 선언됐으면 지정한 초기화 메소드가 호출된다.
8. 빈이 BeanPostProcessor를 구현하면 스프링은 postProcessAfterInitialization()메소드를 호출한다.
9. 빈은 애플리케이션에서 사용할 준비가 된 것이며, 애플리케이션 컨텍스트가 소멸될 때까지 애플리케이션 컨텍스트에 남아 있게 된다.
10. 빈이 DisposableBean인터페이스를 구현하면 스프링은 destroy()메소드를 호출한다. 마찬가지로 빈이 destory-method와 함께 선언됐으면 지정된 메소드가 호출된다.

# 생명주기 관련 애노테이션

- @PostConstruct -JSR-250 스펙으로 JSR-250을 구현하고 있는 다른 프레임워크에서도 사용가능하다. 인스턴스 생성후에 호출된다.
- @Bean(initMethod) - @Bean(initMethod="init") 과 같은 형태로 사용함으로써 초기화 메소드를 사용할 수 있다. 아래는 Java Config에서의 사용예이다.

```
@Bean(initMethod="init")
public MyBean mybean() {
        return new MyBean();
}
```

- @PreDestroy - JSR-250스펙에서 정의되어 있으며 종료될 때 사용할 메소드 위에 사용하면 된다.
- @Bean(destroyMethod) - @Bean(destroyMethod="destroy")와 같은 형태로 사용함으로써 종료될때 호출되도록 할 수 있다.

# 스프링 버전별 새로운 기능

- https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/new-in-3.0.html
- https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/new-in-3.1.html
- https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/new-in-3.2.html

# 스프링 버전별 새로운 기능

스프링 프레임워크 4.0의 새로운 기능 및 향상된 기능

1. 향상된 시작경험

2. Deprecated된 패키지 및 메서드 제거

3. Java 8 지원

4. Java EE 6, 7 지원

5. Groovy 빈 정의 DSL

6. 코어 컨테이너 개선

7. 전반적인 웹 개선

8. 웹소켓, SockJS,  STOMP 메시징

9. 테스트 개선

# 스프링 버전별 새로운 기능

스프링 프레임워크 4.1의 새로운 기능 및 향상된 기능

1. JMS 개선
2. 캐싱 개선
3. 웹 개선
4. 웹소켓 메시징 개선
5. 테스트 개선

# 스프링 버전별 새로운 기능

스프링 프레임워크 4.2의 새로운 기능 및 향상된 기능

1. 코어 컨테이너 개선
2. 데이터 엑세스 개선
3. JMS 개선
4. 웹 개선
5. 웹소켓 메시징 개선
6. 테스트 개선

# 스프링 버전별 새로운 기능

스프링 프레임워크 4.3의 새로운 기능 및 향상된 기능

1. 코어 컨테이너 개선
2. 데이터 엑세스 개선
3. 캐싱 개선
4. JMS 개선
5. 웹 개선
6. 웹소켓 메시징 개선
7. 테스트 개선
8. 새로운 라이브러리 및 서버 세대 지원

# 스프링 버전별 새로운 기능

1. JDK 8+9와 Java EE 7 베이스라인
2. 패키지, 클래스, 메서드 제거
3. 코어 컨테이너 개선
4. 일반적인 웹 개선
5. 리액티브(Reactive) 프로그래밍 모델
6. 테스트 개선

# Maven

# Maven

- Maven은 지금까지 애플리케이션을 개발하기 위한 반복적으로 진행해 왔던 작업들을 지원하기 위하여 등장한 도구이다.
- Maven을 사용하면 빌드(Build), 패키징, 문서화, 테스트와 테스트 리포팅, git, 의존성관리, svn등과 같은 형상관리서버와 연동(SCMs), 배포 등의 작업을 손쉽게 할 수 있다.
- Maven을 이해하려면 CoC(Convention over Configuration)라는 단어를 먼저 이해해야 한다.CoC란 일종의 관습을 말하는데, 예를 들자면 프로그램의 소스파일은 어떤 위치에 있어야 하고, 소스가 컴파일된 파일들은 어떤 위치에 있어야 하고 등을 미리 정해놨다는 것이다.이 말은 관습에 이미 익숙한 사용자는 쉽게 Maven을 사용할 수 있는데, 관습에 익숙하지 않은 사용자는 이러한 제약사항에 대해서 심한 거부감을 느낄 수 있다.Maven을 사용한다는 것은 어쩌면 이러한 관습 즉 CoC에 대해서 알아나가는 것이라고도 말할 수 있다.

# Maven을 사용할 경우 얻게 되는 잇점

- Maven을 사용할 경우, 굉장히 편리한 점들이 많다.
- 많은 사람들이 손꼽는 장점중에는 편리한 의존성 라이브러리 관리가 있다. 앞에서 JSTL을 학습할 때, 몇가지 파일을 다운로드 하여 /WEB-INF/lib폴더에 복사하여 사용했었다.
- 관련된 라이브러리가 많아질 수록 이러한 방식은 상당히 불편해진다. Maven을 사용하면 설정파일에 몇줄 적어줌으로써 직접 다운로드 받거나 하는 것을 하지 않아도 라이브러리를 사용할 수 있 있다.
- 프로젝트에 참여하는 개발자가 많아지게 되면, 프로젝트를 빌드하는 방법에 대하여 가이드하는 것도 쉬운일이 아니다. Maven을 사용하게 되면 Maven에 설정한 대로 모든 개발자가 일관된 방식으로 빌드를 수행할 수 있게 된다.
- Maven은 또한 다양한 플러그인을 제공해줘서, 굉장히 많은 일들을 자동화 시킬 수 있다.

# pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>kr.co.sunnyvale</groupId>
    <artifactId>examples</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>mysample</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

project : pom.xml 파일의 최상위 루트 엘리먼트(Root Element)이다.
modelVersion : POM model의 버전이다.
groupId : 프로젝트를 생성하는 조직의 고유 아이디를 결정한다. 일반적으로 도메인이름을 거꾸로 적는다.
artifactId : 해당 프로젝트에 의하여 생성되는 artifact의 고유 아이디를 결정한다. Maven을 이용하여 pom.xml을 빌드할 경우 다음과 같은 규칙으로 artifact가 생성된다. artifactid-version.packaging. 위 예의 경우 빌드할 경우 mysamexamples-1.0-SNAPSHOT.jar 파일이 생성된다.
packaging : 해당 프로젝트를 어떤 형태로 packaging 할 것인지 결정한다. jar, war, ear 등이 해당된다.
version : 프로젝트의 현재 버전. 추후 살펴보겠지만 프로젝트가 개발 중일 때는 SNAPSHOT을 접미사로 사용한다. Maven의 버전 관리 기능은 라이브러리 관리를 편하게 한다.
name : 프로젝트의 이름이다.
url : 프로젝트 사이트가 있다면 사이트 URL을 등록하는 것이 가능하다.

Maven 을 이용할 경우 얻게 되는 큰 잇점 중의 하나는 Dependency Management 기능이다. 위 pom.xml 파일에서 〈dependencies/〉 엘리먼트가 Dependency Management 기능의 핵심이라고 할 수 있다. 해당 엘리먼트 안에 필요한 라이브러리를 지정하게 된다.

# Spring 실습

# Spring 설정

- xml 파일을 이용한 명시적 설정
- Java Config를 이용한 명시적 설정
- Component scan을 이용해서 빈을 찾아 자동으로 DI하기

# @ComponentScan

- package가 지정되지 않으면 Java Config파일이 있는 패키지부터 하위 패키지까지 검색을 한다.
- @Component , @Controller, @Service, @Repository 애노테이션이 붙은 객체를 찾고, 자동으로 DI한다.
- xml파일로 설정할 때는 〈context:componet-scan〉을 사용한다.

# Spring JDBC

# Spring에서 제공하는 IoC/DI 컨테이너

- BeanFactory : IoC/DI에 대한 기본 기능을 가지고 있다.
- ApplicationContext : BeanFactory의 모든 기능을 포함하며, 일반적으로 BeanFactory보다 추천된다. 트랜잭션처리, AOP등에 대한 처리를 할 수 있다. BeanPostProcessor, BeanFactoryPostProcessor등을 자동으로 등록하고, 국제화 처리, 어플리케이션 이벤트 등을 처리할 수 있다.

# Spring JDBC

- JDBC 프로그래밍을 보면 반복되는 개발 요소가 있다.
- 이러한 반복적인 요소는 개발자를 지루하게 만든다.
- 개발하기 지루한 JDBC의 모든 저수준 세부사항을 스프링 프레임워크가 처리해준다.
- 개발자는 필요한 부분만 개발하면 된다.

# Spring JDBC – 개발자가 해야 할 일은?

| 동작 | 스프링 | 어플리케이션 개발자 |
|---|---|---|
| 연결 파라미터 정의. | | O |
| 연결 오픈. | O | |
| SQL 문 지정. | | O |
| 파라미터 선언과 파라미터 값 제공 | | O |
| 스테이트먼트(statement) 준비와 실행. | O | |
| (존재한다면) 결과를 반복하는 루프 설정 | O | |
| 각 이터레이션에 대한 작업 수행. | | O |
| 모든 예외 처리. | O | |
| 트랜잭션 제어. | O | |
| 연결, 스테이트먼트(statement), 리절트 셋(resultset) 닫기. | O | |

# Spring JDBC 패키지

- org.springframework.jdbc.core
- org.springframework.jdbc.datasource
- org.springframework.jdbc.object
- org.springframework.jdbc.support

# JdbcTemplate

- org.springframework.jdbc.core에서 가장 중요한 클래스
- 리소스 생성, 해지를 처리해서 연결을 닫는 것을 잊어 발생하는 문제등을 피할 수 있도록 한다.
- 스테이먼트(Statement)의 생성과 실행을 처리한다.
- SQL조회, 업데이트, 저장 프로시저 호출, ResultSet 반복호출 등을 실행한다.
- JDBC예외가 발생할 경우 org.springframework.dao패키지에 정의되어 있는 일반적인 예외로 변환시킨다.

# JdbcTemplate select 예제 1

- 열의 수 구하기

```
int rowCount = this.jdbcTemplate.queryForInt("select count(*)
from t_actor");
```

# JdbcTemplate select 예제 2

- 변수 바인딩 사용하기

```
int countOfActorsNamedJoe =
this.jdbcTemplate.queryForInt("select count(*) from t_actor
where first_name = ?", "Joe");
```

# JdbcTemplate select 예제 3

- String값으로 결과 받기

```
String lastName = this.jdbcTemplate.queryForObject("select
last_name from t_actor where id = ?", new Object[] {1212L},
String.class);
```

# JdbcTemplate select 예제 4

- 한 건 조회하기

```
Actor actor = this.jdbcTemplate.queryForObject(
 "select first_name, last_name from t_actor where id = ?",
 new Object[] {1212L},
 new RowMapper<Actor>() {
  public Actor mapRow(ResultSet rs, int rowNum) throws SQLException {
   Actor actor = new Actor();
   actor.setFirstName(rs.getString("first_name"));
   actor.setLastName(rs.getString("last_name"));
   return actor;
  }
 });
```

# JdbcTemplate select 예제 5

- 여러 건 조회하기

```
List<Actor> actors = this.jdbcTemplate.query(
 "select first_name, last_name from t_actor",
 new RowMapper<Actor>() {
  public Actor mapRow(ResultSet rs, int rowNum) throws SQLException {
   Actor actor = new Actor();
   actor.setFirstName(rs.getString("first_name"));
   actor.setLastName(rs.getString("last_name"));
   return actor;
  }
 });
```

# JdbcTemplate select 예제 6

- **중복 코드 제거 ( 1건 구하기와 여러건 구하기가 같은 코드에 있을 경우 )**

```
public List<Actor> findAllActors() {
  return this.jdbcTemplate.query( "select first_name, last_name from t_actor", new ActorMapper());
}
private static final class ActorMapper implements RowMapper<Actor> {
  public Actor mapRow(ResultSet rs, int rowNum) throws SQLException {
    Actor actor = new Actor();
    actor.setFirstName(rs.getString("first_name"));
    actor.setLastName(rs.getString("last_name"));
    return actor;
  }
}
```

# JdbcTemplate insert 예제

- INSERT 하기

```
this.jdbcTemplate.update("insert into t_actor (first_name,
last_name) values (?, ?)", "Leonor", "Watling");
```

# JdbcTemplate update 예제

- UPDATE 하기

```
this.jdbcTemplate.update("update t_actor set = ?  where id = ?",
"Banjo", 5276L);
```

# JdbcTemplate delete 예제

- DELETE 하기

```
this.jdbcTemplate.update("delete from actor where id = ?",
Long.valueOf(actorId));
```

# JdbcTemplate외의 접근방법

- NamedParameterJdbcTemplate
- SimpleJdbcTemplate
- SimpleJdbcInsert