```
1. SW 재사용 방안들
  1)Copy & Paste
    -초보적인 재사용 방식으로 비슷한 예제를 다른 source에서 복사해서 사용한다.
      GregorianCalendar date = (GregorianCalendar)Calendar.getInstance();
      SimpleDateFormat df = new SimpleDateFormat("yyyyMMdd");
      String date = df.format(date);
    -예를 들어 A라는 class에서 Date type을 String type으로 변환하는 coding을 하고, class B에서 동일한 logic이 필요하여 복사했다고 가정한 경우,
    -JDK version이 바뀌어 동일한 기능을 제공하는 향상된 interface가 나오면 위의 code를 사용한 A, B class를 모두 변경해야 한다.

  2)Method(or Function)
    -자주 사용되고, 유사한 기능들을 모아 method로 정의하여 재사용함.

      public class DateUtility{
        public static String toStringToday(String format){
          GregorianCalendar date = (GregorianCalendar)Calendar.getInstance();
          SimpleDateFormat df = new SimpleDateFormat(format);
          String date = df.format(date);
        }
      }
      String sdate = DateUtility.toStringToday("yyyyMMdd");

    -JDK version이 바뀌거나 method의 내용이 수정되더라도 해당 class를 모두 수정할 필요 없이 toStringToday() method의 내용만 수정하면 된다.
    -toStringToday() method의 signature를 변경하면 이 method를 사용하는 모든 class에 영향을 준다.
    -method 재사용방법은 '복사 & 붙이기'보다는 진보된 방식이지만, 작업 영역간의 결합도(Coupling) 문제는 여전히 존재한다.

  3)Class(Inheritance)
    public class Person{
      public String printBirthDate(String format){
        DateUtility.toStringToday(birthDate, format);
      }
    }

    -Person을 상속받은 모든 class들은 자동적으로 변경된 printBirthDate() method를 사용하게 된다.
    -DateUtility class의 method가 변경되더라도 printBirthDate() method의 interface가 변하지 않으면 나머지 class들은 영향을 받지 않는다.
    -부모 class가 바뀌면 자식 class를 변경해야 한다.

  4)AOP(Aspect Oriented Programming)
    -관심의 분리(Seperation of Concerns)
    -AOP는 OOP를 더욱 OOP 답게 만들어 줄 수 있다.
    -AOP는 OOP 뿐만 아니라 기존의 절차적 programming에도 적용될 수 있다.
    -AOP가 핵심관심 module의 code를 직접 건드리지 않고 필요한 기능이 동작하도록 하는데, 위빙(Weaving)이라고 하는 특수한 작업이 필요하다.
    -즉, AOP에서 weaving 작업을 통해 핵심 module 사이 사이에 필요한 횡단 관심 code가 동작하도록 엮어지게 만든다.


2. Design Pattern과 Framework의 관련성
  1)Design Pattern 이란?
    -program 개발에서 자주 나타나는 과제를 해결하기 위한 방법 중 하나로, software 개발과정에서 발견된 know-how
```

를 축적하여 이름을 붙여 이후에 재사용하기 좋은 형태로 특정 규약을 묶어서 정리한 것.
-이 용어를 software 개발 영역에서 구체적으로 처음 제시한 곳은, GoF(Gang of Four)라 불리는 네 명의 computer 과학 연구자들이 쓴 서적 'Design Patterns : Elements of Reusable Object-Oriented Software'(재사용 가능한 객체지향 software의 요소-Design Pattern)이다.

2)Design Pattern을 사용하는 이유
-요구사항은 수시로 변경 -> 요구사항 변경에 대한 source code 변경을 최소화
-여러 사람이 같이 하는 team project 진행 -> 범용적인 coding style을 적용
-상황에 따라 인수 인계하는 경우도 발생 -> 직관적인 code를 사용

3)Framework 란?
-비 기능적(Non-Functional) 요구사항(성능, 보안, 확장성, 안정성 등)을 만족하는 구조와 구현된 기능을 안정적으로 실행하도록 제어해 주는 잘 만들어진 구조의 library의 덩어리
-framework는 application들의 최소한의 공통점을 찾아 하부 구조를 제공함으로써 개발자들로 하여금 system의 하부 구조를 구현하는데 들어가는 노력을 절감하게 해 줌.

4)Framework를 사용하는 이유
-비기능적인 요소들을 초기 개발 단계마다 구현해야 하는 불합리함을 극복해준다.
-기능적인 요구사항에 집중할 수 있도록 해준다.
-Design Pattern과 마찬가지로 반복적으로 발견되는 문제를 해결하기 위한 특화된 solution을 제공한다. 예)한글 encoding
-개발자는 각 개개인의 능력 차이가 큰 직종이고, 따라서 개발자의 구성에 따라 project의 결과 역시 차이가 크다.
-그래서 이런 상황을 극복하기 위한 code의 결과물이 framework이다.
-framework를 이용한다는 의미는 program의 기본 흐름이나 구조를 정하고, 모든 team원이 이 구조에 자신의 code를 추가하는 방식으로 개발하게 된다.
-framework의 최대 장점은 개발에 필요한 구조를 이미 code로 만들어 놓았기 때문에, 실력이 부족한 개발자라 하더라도 반쯤 완성한 상태에서 필요한 부분을 조립하는 형태의 개발이 가능하다는 점이다.
-회사의 입장에서 framework를 사용하면 일정한 품질이 보장되는 결과물을 얻을 수 있고, 개발자의 입장에서는 완성된 구조에 자신에 만든 code를 개발해서 넣어주는 형태이므로 개발시간을 단축시킬 수 있다.

5)Design Pattern과 Framework의 관련성
-Design Pattern을 Framework의 핵심적인 특징이고, Framework를 사용하는 Application에 그 패턴이 적용된다는 특징을 가지고 있다.
-하지만 Framework는 Design Pattern이 아니다.
-Design Pattern은 Application을 설계할 때 필요한 구조적인 guideline이 되어 줄 수는 있지만 구체적으로 구현된 base code를 제공하지 않는다.
-Framework는 Design Pattern과 함께 pattern이 적용된 기반 class library를 제공해서 Framework를 사용하는 구조적인 틀과 구현 code를 함께 제공한다.

6)결론
-개발자는 Framework의 base code를 확장하여 사용하면서 자연스럽게 그 Framework에서 사용된 pattern을 적용할 수 있게 된다.

3. Framework의 구성요소와 종류
1)IoC(Inversion of Control)
-'제어의 역전' 즉, instance 생성부터 소멸까지의 instance 생명주기 관리를 개발자가 아닌 container가 대신 해준다는 뜻임.
-container 역할을 해주는 framework에게 제어하는 권한을 넘겨서 개발자의 code가 신경 써야 할 것을 줄이는 전략이다.
-framework의 동작원리를 제어흐름이 일반적인 program 흐름과 반대로 동작하므로 IoC라고 설명함.
-Spring Container는 IoC를 지원하며, meta-data(XML설정)를 통해 beans를 관리하고 application의 중요부분을 형성함.
-Spring Container는 관리되는 bean들을 의존성 주입(Dependency Injection)을 통해 IoC를 지원함.

2)Class Library
-Framework는 특정 부분의 기술적인 구현을 library 형태로 제공한다.
-class library라는 구성요소는 Framework의 정의 중 하나인 'Semi Complete(반제품)'이다. 라고 해석하게 만들었다.

특징	Framework	Library
user code의 작성	Framework class를 sub-classing 해서 작성	독립적으로 작성
호출 흐름	Framework코드가 user code를 호출	user code가 library를 호출
실행흐름	Framework가 제어	user code가 제어
객체의 연동	구조 Framework가 정의	독자적으로 정의

3)결론
-Framework와 Library를 구분하는 방법은 실행제어가 어디서 일어나는가에 달려있다.
-Library는 개발자가 만든 class에서 직접 호출하여 사용하므로 실행의 흐름에 대한 제어를 개발자의 code가 관장하고 있다.
-Framework는 반대로 Framework에서 개발자가 만든 class를 호출하여 실행의 흐름에 대한 제어를 담당한다.

4)Design Pattern
-Design Pattern + Library = Framework
-Framework는 Design Pattern과 그것이 적용된 기반 library의 결합으로 이해할 수 있다.
-Framework의 Library를 살펴볼 때도 적용된 pattern을 주목해서 살펴 본다면 그 구성을 이해하기 쉽다.
-특히 Framework를 확장하거나 customizing 할 때는 Framework에 적용된 pattern에 대한 이해가 꼭 필요하다.

5)Framework 종류
-Architecture 결정 = 사용하는 Framework의 종류 + 사용전략
-Web(MVC) : Spring MVC, Struts2, Webwork, PlayFramework
-OR(Object-Relational) Mapping : MyBatis, Hibernate, JPA, Spring JDBC
-AOP(Aspect Oriented Programming) : Spring AOP, AspectJ, JBoss AOP
-DI(Dependency Injection) : Spring DI, Google Guice
-Build & Library Management : Ant + Ivy, Maven, Gradle
-Unit Testing : jUnit, TestNG, Cactus
-JavaScript : jQuery, AngularJS, Node.js

4. Spring의 주요 project

1)Spring Framework
-Spring을 이용해서 application을 개발할 때 기반이 되는 framework.
-Spring의 핵심 기능인 DI와 AOP 기능을 제공.
-Web Application을 개발할 때 사용하는 Spring MVC, Spring ORM 등의 기능도 포함

2)Spring Data
-data 연동을 위한 단일 API 제공
-JPA, MongoDB, Neo4j, Redis 등 RDBMS와 NOSQL 과의 연동을 적은 양의 code로 처리할 수 있도록 해준다.

3)Spring Security
-인증과 허가에 대한 기반 framework 및 관련 module을 제공
-web application을 위한 보안을 간단한 설정과 약간의 code구현으로 처리할 수 있다.

4)Spring Batch
-batch 처리를 위한 기반 framework를 제공
-data 처리, 흐름 제어, 실패 재처리 등 batch 처리 application이 필요로 하는 기능을 기본으로 제공

5)Spring Integration
-system 간의 연동을 위한 messaging framework를 제공

6)Spring Social
-Twitter, Facebook 등 social network 연동을 위한 기능을 제공

7)Groovy
-Java VM 상에서 동작하는 동적 언어

8)Grails
-Spring Framework나 Hibrernate등을 기반으로 해서 Groovy로 구현된 web application을 개발할 수 있는 Full Stack Framework

9)Spring Android
-Android 단말기와 접속을 간편하게 하고자 개발된 제품

10)Spring Tool Suite
-Eclipse 기반의 spring용 통합개발환경

5. Spring Framework 이란?
1)Enterprise application 에서 필요로 하는 기능을 제공하는 Framework 이다.
2)J2EE를 대체하는 Framework이다.
3)Java Enterprise 개발을 편하게 해주는 오픈소스 경량급 Application Framework이다.
4)Application Framework : 특정 계층이나 기술, 업무, 분야에 국한되지 않고 Application의 전 영역을 포괄하는 범용적인 Framework를 말한다.
5)경량급 Framework : 단순한 Web Container(예.tomcat)에서도 enterprise 개발의 고급기술을 대부분 사용할 수 있다.
6)Enterprise 개발 용이 : 개발자가 복잡하고 실수하기 쉬운 low level(보안, 인증, transaction)에 많이 신경쓰지 않으면서 Business logic 개발에 전념할 수 있도록 해준다.
7)Opensource : Spring은 opensource의 장점을 충분히 취하면서 동시에 opensource 제품의 단점과 한계를 잘 극복한다.
8)과거에 나왔던 다른 Framework들과 Spring과의 차별점
①복잡함에 반기를 들어서 만들어진 Framework
-Spring은 그 태생 자체가 enterprise급의 system이 실패하는 이유를 복잡성으로 보고, 복잡성을 해결하기 위해서 나온 경량화된 Framework이다.
-일반적인 Java의 class와 interface를 이용하는 구조를 사용하기 때문에 진입 장벽이 높지 않았고, EJB에 비해 가볍기 때문에 빠른 시간에 enterprise 급의 system을 작성할 수 있었다.
②Project의 전체 구조를 설계할 때 유용한 Framework
-다른 Framework들은 Web 영역이나 database 영역등의 전문적인 영역에 대해서만 지원하는 경우가 많았고, business logic을 처리하는 부분에 대한 설계는 개발자의 역량에 맡기는 경우가 많았다.
-반면 Spring은 어느 한 분야에 집중하지 않고, 전체를 설계하는 용도로 사용될 수 있었다.
③다른 Framework들의 포용
-Spring은 전체 구조에 집중했기 때문에 특정한 영역의 Framework와 공존하는 방식을 사용할 수 있었다.
-다른 Framework들은 특정 Framework를 채택하면 해당 영역 전체를 수정해야 하는 고질적인 문제를 가지고 있었지만, Spring은 다른 Framework들과의 통합을 지원했기 때문에 최소한의 수정이 가능했다.
-Spring의 최대 장점은 기본 뼈대를 흔들지 않고, 여러 종류의 Framework를 혼용해서 사용할 수 있다는 점이다.
④개발 생산성과 개발 도구의 지원
-Spring의 경우 이론적으로는 개발자가 제대로 이해애햐 하는 부분이 많지만, 결과적으로 code의 양은 확실히 줄어들 수 있었고, 유지보수에 있어서도 XML의 설정 등을 이용했기 때문에 환영받을 수 있었다.
-STS나 Eclipse, IntelliJ등의 Plug-in의 지원 역시 다른 Framework들에 비해서 빠른 update가 되었기 때문에 별도의 새로운 개발 도구에 대한 적응 없이도 개발이 가능했다.

6. Spring Framework의 역사
1)Spring이 처음 세상에 등장한 것은 2002년이다.
2)Rod Johnson(2012년 2월 SpringSource사와 VMWare사에서 물러나기로 했다)의 저서 'Expert One-on-One J2EE Design and Development(2002, Wrox)에서 MVC가 설명된 부분이 있다.
3)이 가운데 '이 framework는 sample이 아니며 공개되어 자유롭게 사용할 수 있다'는 취지의 문장에 이어서 Spring의 이름이 살짝 들어있던 것이 최초다.
4)Spring이 정식으로 등장한 것은 2004년 3월이다.
Refer to https://www.quora.com/What-is-the-history-of-The-Spring-Framework
Refer to https://www.slideshare.net/AliakseiZhynhiarousk/spring-framework-5-history-and-reactive-features

공개일	버전	
2004년 3월	1.0	DIxAOP container의 시작. Bean 정의 file 시대의 개막
2004년 9월	1.1	Bean 정의 file 간략화(평판이 좋지 않았다)
2005년 5월	1.2	Bean 정의 file의 거듭된 간략화
2006년 10월	2.0	Bean 정의 file이 DTD로부터 XML schema 형식으로 변경(독자 schema를 사용할 수 있게 됨) Annotation의 등장. JPA와 script 언어의 지원과 다기능화에 돌입
2007년 11월	2.5	Annotation 강화
2009년 12월	3.0	Annotation의 거듭된 강화. 하지만 대기업을 중심으로 Bean 정의 file이 여전히 선호
2011년 12월	3.1	Cloud 시대에 대응. Spring에 Cache 기능도 등장
2012년 12월	3.2	
2013년 12월	4.0	
2014년 9월	4.1	
2015년 7월	4.2	
2016년 6월	4.3	
2017년 2월	5.0	

5)2007년 Rod Johnson이 CEO를 맡아 Spring Framework 개발과 컨설팅을 하던 Interface21의 이름을 SpringSource로 변경
6)2009년 8월, VMWare가 SpringSource사를 인수
7)VMWare의 SpringSource의 인수는 VMWare의 VMWare vShhere와 VMWare vCenter를 기반으로 한 IaaS상에 Spring을 배치함으로써 Java application을 cloud에서 더 쉽게 이용할 수 있게 하려는 목적

7. Spring의 개요와 특징
1)EJB 기반으로 개발하지 않는다. POJO 기반으로 개발한다.
2)가볍다.
3)제어가 가능하다.
4)상호 관련이 적다.
5)AOP(Aspect Oriented Programming) 를 지원한다.
-transaction이나 logging, 보안과 같이 여러 module에서 공통으로 필요로 하지만 실제 module의 핵심은 아닌 기능들을 분리해서 각 module에 적용할 수 있다.
6)Container를 통한 lifecycle을 관리하고, Container로부터 필요한 객체를 가져와 사용할 수 있다.
7)XML 기반으로 component를 개발할 수 있도록 지원해 준다.
8)객체의 lifecycle을 관리하기 위하여 DI 를 사용하는 경량 Container이다.
9)DI(Dependency Injection) pattern을 지원한다.
-Spring은 설정 file을 통해서 객체 간의 의존 관계를 설정할 수 있도록 한다.
-따라서 객체는 직접 의존하고 있는 객체를 생성하거나 검색할 필요가 없다.
10)영속성과 관련된 다양한 API 를 지원한다.
-JDBC, iBatis, Hibernate, JPA ...

8. Spring Version Up
1)Spring 2.5 : Annotation을 활용하는 설정을 도입하면서 편리한 설정과 개발이 가능하도록 함.
2)Spring 3.0 : 별도의 설정 없이도 Java class만으로 설정 file을 대신할 수 있게 함.
3)Spring 4.0 : Mobile 환경과 Web 환경에서 많이 사용되는 REST 방식의 Controller 지원됨.

9. Spring Framework 전략
1)Spring 삼각형
2)Enterprise 개발의 복잡함을 상대하는 Spring의 전략
-The Spring Triangle.jpg 참조

10. Portable Serivce Abstraction
-Transaction 추상화, OXM 추상화, Data access의 Exception 변환기능 등 기술적인 복잡함은 추상화를 통해 low level의 기술 구현부분과 기술을 사용하는 interface로 분리한다.

11. 객체지향과 DI
-Spring은 객체지향에 충실한 설계가 가능하도록 단순한 객체 형태로 개발할 수 있고, DI는 유연하게 확장 가능한 객체를 만들어 두고 그 관계는 외부에서 dynamic하게 설정해 준다.

12. AOP
-Application logic을 담당하는 code에 남아있는 기술 관련 code를 분리해서 별도의 module로 관리하게 해 주는 강력한 기술이다.

13. POJO
-객체지향 원리에 충실하면서 특정 환경이나 규약에 종속되지 않고 필요에 따라 재활용될 수 있는 방식으로 설계된 객체이다.

14. Spring 구조
1)Spring Framework를 구성하는 기능 요소
-spring-framework-overview-ppt-12-728.jpg 참조
-Spring Core
--Spring Framework의 기본 기능을 제공
--이 module에 있는 BeanFactory는 Spring의 기본 Container이면서 Spring DI의 기반이다.
-Spring AOP
--이 module을 통해 Aspect 지향 programming을 지원
--이 module은 Spring Application에서 Aspect를 개발할 수 있도록 지원
-Spring ORM
--MyBatis, Hibernate, JPA 등 널리 사용되는 ORM Framework와의 연결고리를 지원
--ORM 제품들을 Spring의 기능과 조합해서 사용할 수 있도록 해준다.
-Spring DAO
--JDBC에 대한 추상화 계층으로 JDBC coding이나 예외처리 하는 부분을 간편화시켰으며, AOP 모듈을 이용해 transaction 관리 service도 제공
-Spring Web
--일반적인 WebApplication 개발에 필요한 기본기능을 제공한다.
--WebWork나 Struts와 같은 다른 Web Application Framework와의 통합을 지원
-Spring Context
--BeanFactory의 개념을 확장한 것으롤 국제화(i18N) 메시지, Application 생명주기 event, 유효성 검증 등을 지원
-Spring Web MVC

276 --사용자 interface가 Application logic과 분리되는 Web Application을 만드는 경우에 일반적으로 사용되는 paradime이다.
277
278 2)Spring Framework의 주요 module간 의존 관계
279 -모듈간 의존 관계.jpg 참조
280
281
282 15. Maven이란?
283 1)http://maven.apachde.org
284 2)Library 관리 + 빌드 툴
285 -Project의 전체적인 Lifecycle을 관리하는 도구이며, 많은 편리함과 이점이 있어 널리 사용되고 있다.
286 -기존에는 Ant가 많이 사용되었지만 Maven이 Ant를 넘어서 더 많은 개발자들이 사용하게 되었고 비교적 최근에는 Gradle이 새롭게 나와 사용되고 있다(대표적으로는 Android Studio).
287 -Project 구조와 내용을 기술하는 선언적 접근방식의 Opensouce Build Tool 이다.
288 -Compile과 동시에 build를 수행할 수 있으며 test를 병행하거나 server측 deploy 자원을 관리할 수 있는 환경을 제공한다.
289 -하지만 아무래도 개발자들에게 가장 큰 장점은 project의 종속 library들과 그 library에 영향을 미치는 dependency 자원까지 관리 할 수 있다는 점일 것 같다.
290 -즉, jar 파일을 download받아 project에 추가할 경우 그것과 연관된 다른 종속 library 또한 다 찾아야 하는 불편함을 Maven을 통해서 일관성 있는 library간의 의존관계 (의존성) 관리를 할 수 있다는 점이다.
291 -이는 단순히 library 뿐 아니라 project별 module의 의존성 또한 관리가 된다는 뜻이기도 하다.
292 -Maven은 project 전반의 resource 관리와 Configuration 파일, Doc 생성 및 이와 관련한 표준 directory 구조를 처음부터 일관된 형태로 구성하여 진행하기 때문에 project 관리 및 배포 역할을 하는 다른 tool 들과의 연계에서도 뛰어난 유연성을 보여준다.
293 -단점은, version별로 Eclipse에서 구동되는 방식이 약간 호환성이 떨어진다는 점인데, 사용된 Plug-in의 문제인지 Maven 자체의 하위 호환성 문제인지는 모르겠다.
294 -가끔 저장소 접근에 관한 문제도 발생한다고 알려져 있다.
295 -Maven을 학습하기 위해서는 Maven을 설치하고 POM (Project Object Model) 을 작성한 후 각종 build script 혹은 명령어를 통해 배워나가야 하지만 여기서는 기본적인 사용법과 더불어 STS를 통해 application을 작성 한 후 어떻게 Eclipse상에서 Maven을 활용하는지에 초점을 맞춰 진행하도록 한다.
296
297 3)Maven을 사용하는 이유
298 ①편리한 Dependent Library 관리 - Dependency Management
299 ②의존성(Dependency)
300 (1)Library download 자동화
301 -더 이상 필요한 (의존성 있는) library를 하나씩 download 받을 필요가 없다.
302 -필요하다고 선언만 하면 Maven이 자동으로 download 받아준다.
303 (2)Maven은 선언적 (명령식이 아니다)
304 -사용되는 jar 파일들을 어디서 다운로드 받고, 어느 release(version)인지 명시하면, coding을 하지 않아도 Maven이 알아서 관리한다. (re-download, 최신 version 설치 등)
305 (3)Maven이 관리한다.
306 -Library (lib) directory를 생성할 필요가 없다.
307 -Eclipse 내에서 library, classpath 환경 설정을 할 필요도 없다
308 --여러 project에서 project 정보나 jar file들을 공유하기 쉬움
309 --모든 project의 build process를 일관되게 가져갈 수 있음.
310
311 4)Maven 이전의 Library 관리 방법
312 -Library site 접속 -> Library download -> 압축 해제 -> Project에 Library 복사 -> classpath에 추가 -> 다시 Library site 접속
313
314 5)Maven의 Library 관리 방법
315 -pom.xml file 수정 -> build -> pom.xml file 수정
316
317 6)pom.xml

318 ①Maven project를 생성하면 pom.xml file이 생성
319 ②pom.xml file은 Project Object Model 정보를 담고 있다.
320 ③Project 정보 : project의 이름, 개발자 목록, license 등
321 -Build 설정 : source, resource, lifecycle별 실행한 plug-in(goal)등 build와 관련된 설정
322 -Build 환경 : 사용자 환경 별로 달라질 수 있는 profile 정보
323 -POM 연관 정보 : 의존 project(module), 상위 project, 포함하고 있는 하위 module 등
324 ④Project 당 하나의 pom.xml
325 -각각의 project는 pom.xml file을 하나씩 가진다.
326 -pom은 project 자체와 의존성에 대한 설정 및 정보를 포함한다.
327 -Maven은 pom.xml 을 읽어, project를 가공하는 방법을 이해한다.
328 ⑤3 가지 "coordinates"를 이용해 자원을 식별한다.
329 -Group ID
330 -Artifact ID
331 -Version
332 ⑥중요한 속성들
333 -<artifactId/>
334 --아티팩트의 명칭 (Artifact's name), groupId 범위 내에서 유일해야 한다.
335 -<groupId/>
336 --일반적으로 project의 package 명칭
337 -<version/>
338 --Artifact의 현재 version
339 -<name/>
340 --Application의 명칭
341 -<packaging/>
342 --Artifact packaging 유형
343 --POM, jar, WAR, EAR, EJB, bundle, ... 중에서 선택 가능
344 -<distributionManagement/>
345 --Artifact가 배포될 저장소 정보와 설정
346 -<parent/>
347 --Project의 계층 정보
348 -<scm/>
349 --Source code 관리 system 정보
350 -<dependencyManagement/>
351 --의존성 처리에 대한 기본 설정 영역
352 -<dependencies/>
353 --의존성 정의 및 설정 영역