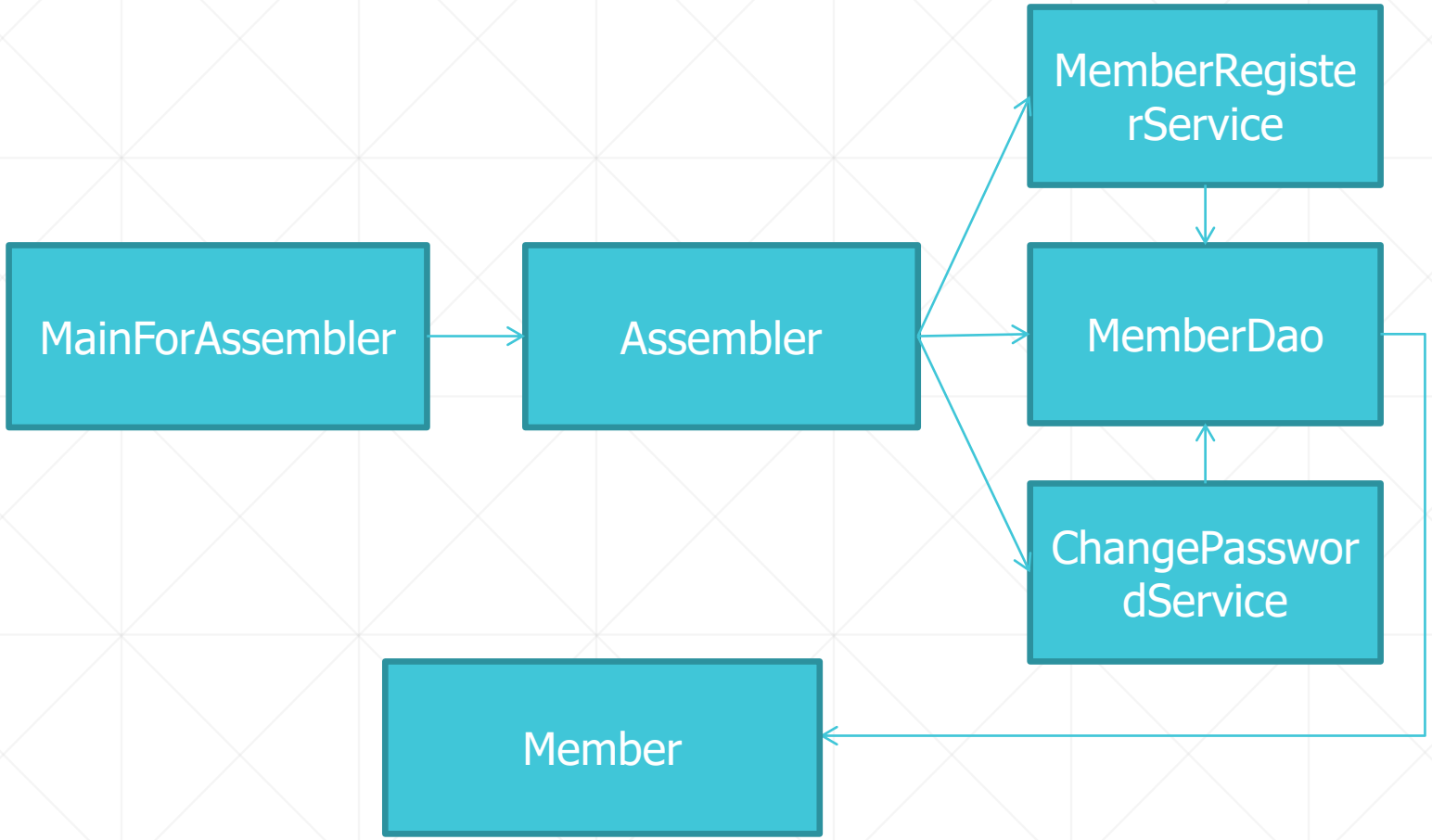
Spring Framework

자바 프로젝트 만들기

Normal Project

- Non-DI



Normal Project

- Non-DI
  - Member.java(1)

```java
import java.util.Date;

public class Member {
    private Long id;
    private String email;
    private String password;
    private String name;
    private Date registerDate;

    public Member(String email, String password,
            String name, Date registerDate) {
        this.email = email;
        this.password = password;
        this.name = name;
        this.registerDate = registerDate;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
```

Normal Project

- Non-DI
  - Member.java(2)

```
public void setRegisterDate(Date registerDate) {
    this.registerDate = registerDate;
}

public void changePassword(String oldPassword, String newPassword){
    if(!password.equals(oldPassword)){
        throw new IdPasswordNotMatchingException();
    }
    this.password = newPassword;
}
```

Normal Project

- Non-DI
  - MemberDao.java

```
public class MemberDao {
    private static long nextId=0;

    private Map<String,Member> map = new HashMap<>();

    public Member selectByEmail(String email){
        return map.get(email);
    }
    public void insert(Member member){
        member.setId(++nextId);
        map.put(member.getEmail(), member);
    }
    public void update(Member member){
        map.put(member.getEmail(), member);
    }
    public Collection<Member> selectAll() {
        return map.values();
    }
}
```

Normal Project

- Non-DI
  - MemberRegisterService.java

```
public class MemberRegisterService {
    private MemberDao memberDao;
    public MemberRegisterService() {
        // TODO Auto-generated constructor stub
    }
    public MemberRegisterService(MemberDao memberDao) {
        this.memberDao = memberDao;
    }
    public void regist(RegisterRequest req){
        Member member = memberDao.selectByEmail(req.getEmail());
        if(member != null){
            throw new AlreadyExistingMemberException("email "+req.getEmail());
        }
        Member newMember = new Member(
                req.getEmail(),
                req.getPassword(),
                req.getName(),
                new Date());
        memberDao.insert(newMember);
    }
}
```

Normal Project

- Non-DI
  - ChangePasswordService.java

```
public class ChangePasswordService {
    private MemberDao memberDao;
    public ChangePasswordService(MemberDao memberDao) {
        this.memberDao = memberDao;
    }
    public void changePassword(String email, String oldPassword, String newPassword){
        Member member = memberDao.selectByEmail(email);
        if(member == null){
            throw new MemberNotFoundException();
        }

        member.changePassword(oldPassword, newPassword);

        memberDao.update(member);
    }
}
```

Normal Project

- Non-DI
  - Assembler.java

```
public class Assembler {
    private MemberDao memberDao;
    private MemberRegisterService regSvc;
    private ChangePasswordService pwdSvc;

    public Assembler() {
        memberDao = new MemberDao();
        regSvc = new MemberRegisterService(memberDao);
        pwdSvc = new ChangePasswordService(memberDao);
    }
    public MemberDao getMemberDao(){
        return memberDao;
    }
    public MemberRegisterService getRegSvc() {
        return regSvc;
    }
    public ChangePasswordService getPwdSvc() {
        return pwdSvc;
    }
}
```

Normal Project

- Non-DI
  - MainForAssembler.java(1)

```
public class MainForAssembler {

    private static Assembler assembler = new Assembler();

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        while(true){
            System.out.println("명령어를 입력하세요");
            String command = scan.nextLine();
            if(command.equalsIgnoreCase("exit")){
                System.out.println("종료합니다");
                break;
            }
            if(command.startsWith("new")){
                processNewCommand(command.split(" "));
                continue;
            }else if(command.startsWith("change")){
                processChargeCommand(command.split(" "));
                continue;
            }
            printHelp();
        }
    }
```

Normal Project

- Non-DI
  - MainForAssembler.java(2)

```
private static void printHelp() {
    System.out.println();
    System.out.println("잘못된 명령입니다. 다시 명령어를 입력하세요.");
    System.out.println();
    System.out.println();
    System.out.println();
    System.out.println();
}

private static void processChargeCommand(String[] arg) {
    if(arg.length!=4){
        printHelp();
        return;
    }
    ChangePasswordService changePwdService = assembler.getPwdSvc();
    try {
        changePwdService.changePassword(arg[1], arg[2], arg[3]);
        System.out.println("암호를 변경했습니다.");
    } catch (MemberNotFoundException e1) {
        System.out.println("존재하지 않는 이메일입니다.\n");
    } catch (IdPasswordNotMatchingException e2) {
        System.out.println("이메일과 암호가 일치하지 않습니다.\n");
    }
```

Normal Project

- Non-DI
  - MainForAssembler.java(3)

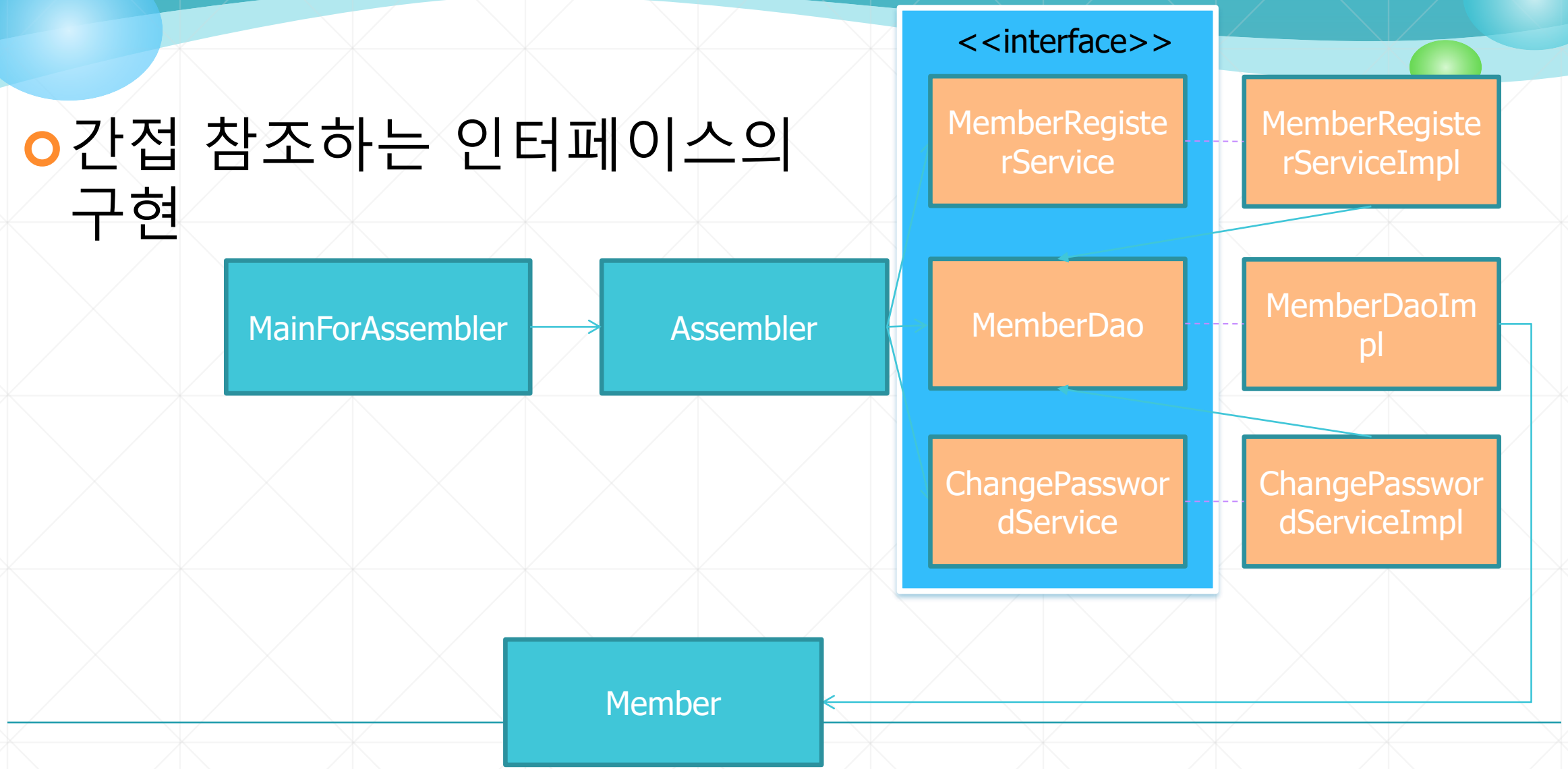
```
private static void processNewCommand(String[] arg) {
    if(arg.length!=5){
        printHelp();
        return;
    }
    MemberRegisterService regSvc = assembler.getRegSvc();
    RegisterRequest req = new RegisterRequest();
    req.setEmail(arg[1]);
    req.setName(arg[2]);
    req.setPassword(arg[3]);
    req.setConfirmPassword(arg[4]);
    if(!req.isPasswordEqualToConfirmPassword()){
        System.out.println("암호가 일치하지 않습니다.");
        return;
    }
    try {
        regSvc.regist(req);
        System.out.println("등록했습니다\n");
    } catch (Exception e) {
        System.out.println("이미 존재하는 이메일입니다\n");
    }
}
```

Normal Project

- Assembler에서 Dao들 간의 강한 연관관계로 인해 객체의 재사용성이 떨어짐
- 새로운 기능 추가로 인한 Assembler의 복잡성과 의존도가 증가.

Normal Project

- 간접 참조하는 인터페이스의 구현



Normal Project

- 간접 참조하는 인터페이스의 구현
  - MemberDao.java

```
public interface MemberDao {

    public Member selectByEmail(String email);
    public void insert(Member member);
    public void update(Member member);
    public Collection<Member> selectAll();
}
```

Normal Project

```
public class MemberDaoImpl implements MemberDao{
    private static long nextId=0;

    private Map<String,Member> map = new HashMap<>();

    public Member selectByEmail(String email){
        return map.get(email);
    }
    public void insert(Member member){
        member.setId(++nextId);
        map.put(member.getEmail(), member);
    }
    public void update(Member member){
        map.put(member.getEmail(), member);
    }
    public Collection<Member> selectAll() {
        return map.values();
    }
```

- 간접 참조하는 인터페이스의 구현
  - MemberDaoImpl.java

Normal Project

- 간접 참조하는 인터페이스의 구현
  - MemberRegisterService.java

```
public interface MemberRegisterService {
    public void regist(RegisterRequest req);
}
```

Normal Project

- 간접 참조하는 인터페이스의 구현
  - MemberRegisterServiceImpl.java

```
public class MemberRegisterServiceImpl implements MemberRegisterService{
    private MemberDao memberDao;
    public MemberRegisterServiceImpl() {
        // TODO Auto-generated constructor stub
    }
    public MemberRegisterServiceImpl(MemberDao memberDao) {
        this.memberDao = memberDao;
    }
    public void regist(RegisterRequest req){
        Member member = memberDao.selectByEmail(req.getEmail());
        if(member != null){
            throw new AlreadyExistingMemberException("email "+req.getEmail());
        }
        Member newMember = new Member(
                req.getEmail(),
                req.getPassword(),
                req.getName(),
                new Date());
        memberDao.insert(newMember);
    }
}
```

Normal Project

- 간접 참조하는 인터페이스의 구현
  - ChangePasswordService.java

```
public interface ChangePasswordService {
    public void changePassword(String email, String oldPassword, String newPassword);
}
```

Normal Project

- 간접 참조하는 인터페이스의 구현
  - ChangePasswordServiceImpl.java

```
public class ChangePasswordServiceImpl implements ChangePasswordService{
    private MemberDao memberDao;
    public ChangePasswordServiceImpl(MemberDao memberDao) {
        this.memberDao = memberDao;
    }
    public void changePassword(String email, String oldPassword, String newPassword){
        Member member = memberDao.selectByEmail(email);
        if(member == null){
            throw new MemberNotFoundException();
        }

        member.changePassword(oldPassword, newPassword);

        memberDao.update(member);
    }
}
```

Normal Project

- 간접 참조하는 인터페이스의 구현
  - Assembler.java

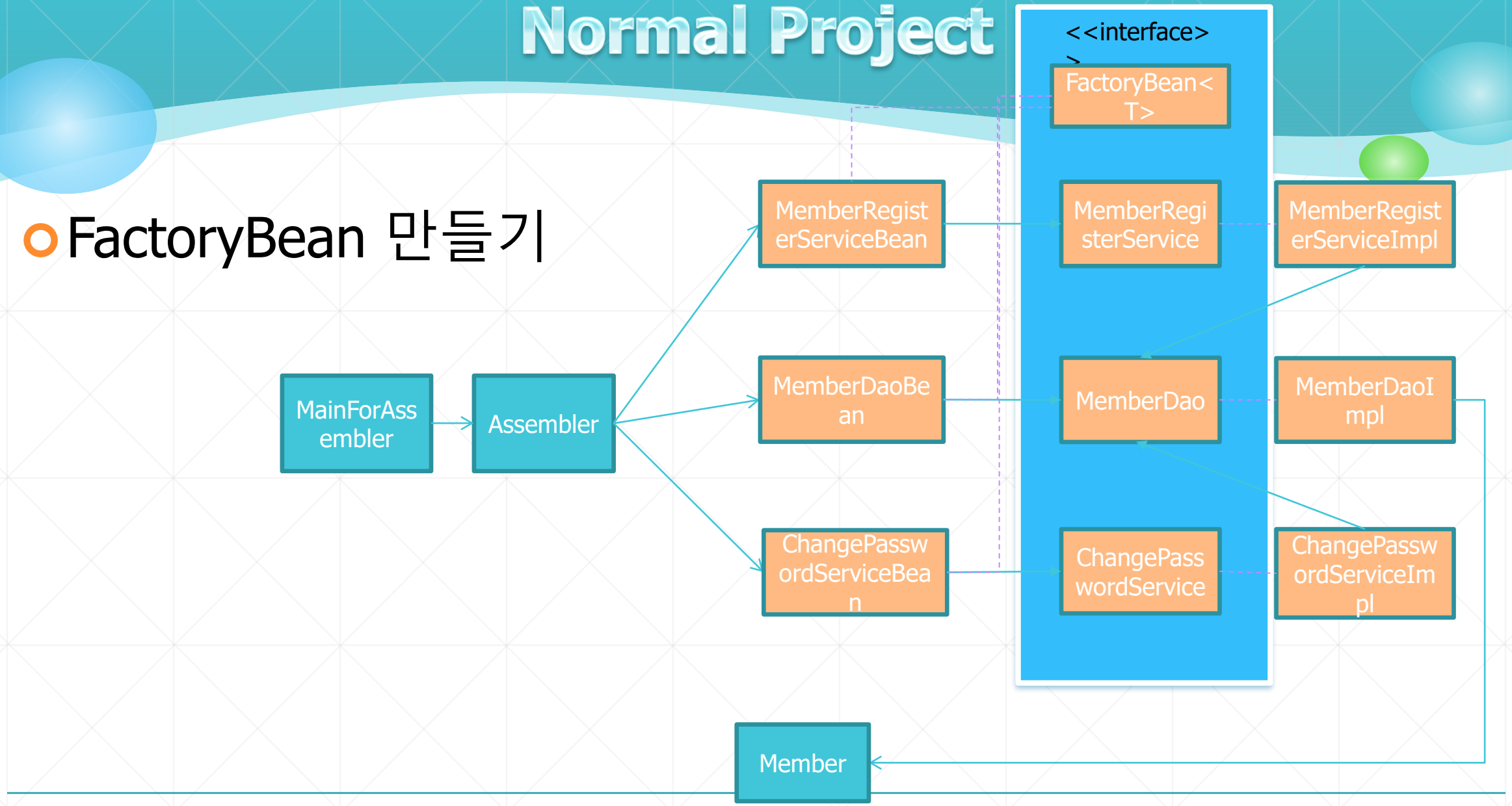
```
public class Assembler {
    private MemberDao memberDao;
    private MemberRegisterService regSvc;
    private ChangePasswordService pwdSvc;

    public Assembler() {
        memberDao = new MemberDaoImpl();
        regSvc = new MemberRegisterServiceImpl(memberDao);
        pwdSvc = new ChangePasswordServiceImpl(memberDao);
    }
    public MemberDao getMemberDao(){
        return memberDao;
    }
    public MemberRegisterService getRegSvc() {
        return regSvc;
    }
    public ChangePasswordService getPwdSvc() {
        return pwdSvc;
    }
}
```

Normal Project

- 간접 참조하는 인터페이스의 구현
  - 간접 참조하는 인터페이스를 구현 함으로써 간접적인 연결관계를 유지한다.
  - 객체 간의 연결관계가 느슨해 짐으로써 객체 내부의 내용이 바뀌거나 하더라도 큰 장애가 발생하지 않는다.
  - 하지만 아직까지 객체를 새로 만들고 종료하는 일이 빈번하게 발생하여 호출시의 오버헤드를 상승시키는 역할을 한다.

Normal Project

- FactoryBean 만들기



Normal Project

- FactoryBean 만들기
  - FactoryBean.java <interface>

```
public interface FactoryBean<T> {
    T getObject() throws Exception;

    Class<?> getObjectType();

    boolean isSingleton();
}
```

Normal Project

- FactoryBean 만들기
  - MemberDaoBean.java

```
public class MemberDaoBean
implements FactoryBean<MemberDao> {
    private MemberDao member;

    @Override
    public MemberDao getObject() throws Exception {
        if(this.member == null) {
            this.member = new MemberDaoImpl();
        }
        return this.member;
    }

    @Override
    public Class<?> getObjectType() {
        return MemberDao.class;
    }

    @Override
    public boolean isSingleton() {
        return true;
    }

}
```

Normal Project

- FactoryBean 만들기
  - MemberRegisterService Bean.java

```java
public class MemberRegisterServiceBean
implements FactoryBean<MemberRegisterService> {
    private MemberRegisterService mrs;
    private MemberDao memberDao;

    public MemberRegisterServiceBean(MemberDao memberDao) {
        this.memberDao = memberDao;
    }

    @Override
    public MemberRegisterService getObject() throws Exception {
        if(this.mrs == null) {
            this.mrs = new MemberRegisterServiceImpl(this.memberDao);
        }
        return this.mrs;
    }

    @Override
    public Class<?> getObjectType() {
        return MemberRegisterService.class;
    }

    @Override
    public boolean isSingleton() {
        return true;
    }

}
```

Normal Project

- FactoryBean 만들기
  - ChangePasswordService Bean.java

```java
public class ChangePasswordServiceBean
implements FactoryBean<ChangePasswordService> {
    private ChangePasswordService cps;
    private MemberDao memberDao;

    public ChangePasswordServiceBean(MemberDao memberDao) {
        this.memberDao = memberDao;
    }

    @Override
    public ChangePasswordService getObject() throws Exception {
        if(this.cps == null) {
            this.cps = new ChangePasswordServiceImpl(this.memberDao);
        }
        return this.cps;
    }

    @Override
    public Class<?> getObjectType() {
        return ChangePasswordService.class;
    }

    @Override
    public boolean isSingleton() {
        return true;
    }

}
```

Normal Project

- FactoryBean 만들기
  - Assembler.java

```
public class Assembler {
    private MemberDao memberDao;
    private MemberRegisterService regSvc;
    private ChangePasswordService pwdSvc;

    public Assembler() {
        try {
        memberDao = new MemberDaoBean().getObject();
        regSvc = new MemberRegisterServiceBean(memberDao).getObject();
        pwdSvc = new ChangePasswordServiceBean(memberDao).getObject();
        } catch(Exception e) {
            e.getMessage();
        }
    }
    public MemberDao getMemberDao(){
        return memberDao;
    }
    public MemberRegisterService getRegSvc() {
        return regSvc;
    }
    public ChangePasswordService getPwdSvc() {
        return pwdSvc;
    }
}
```

Normal Project

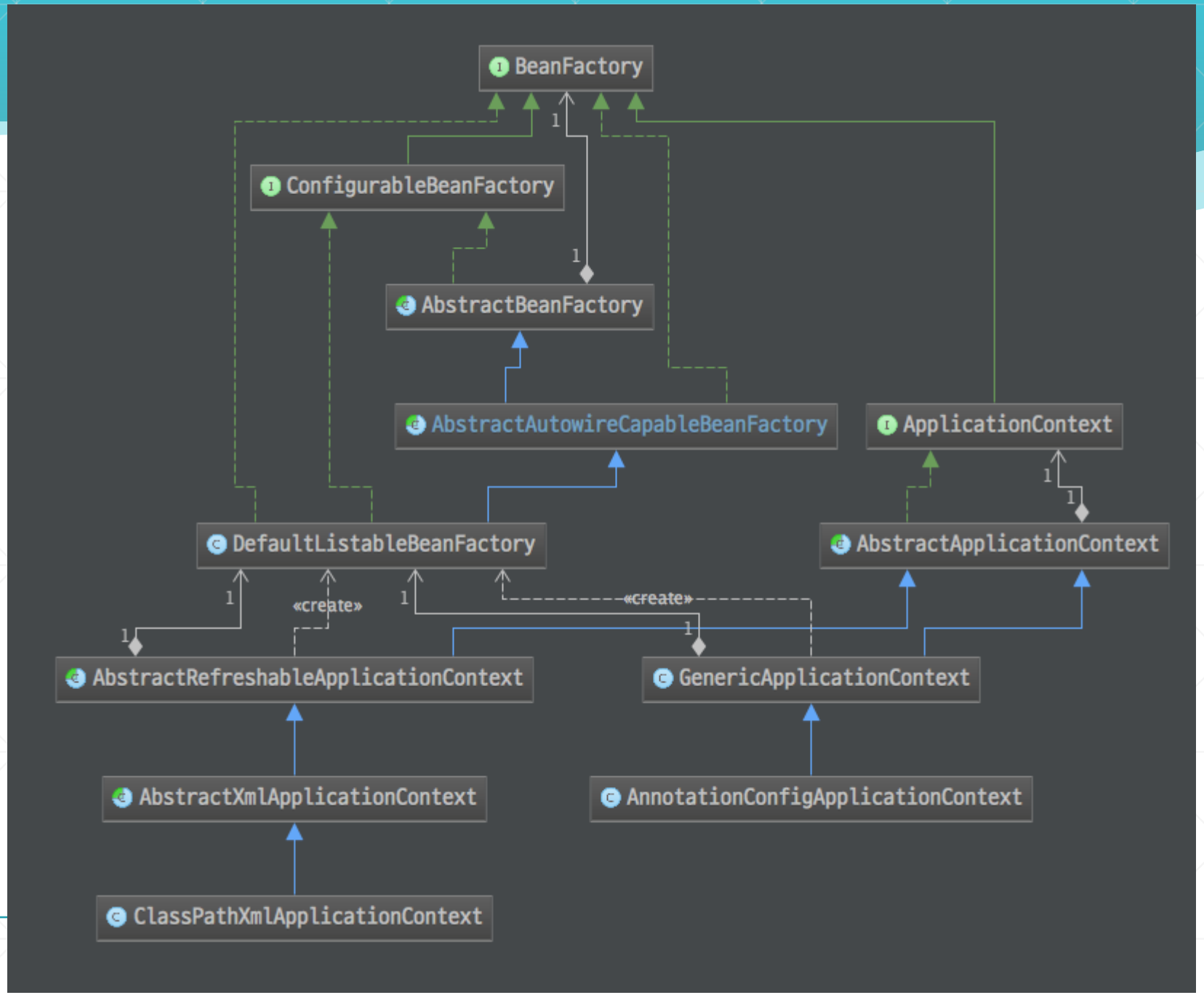
- FactoryBean 만들기
  - 팩토리 빈도 만들었지만 객체가 하나 더 추가될 경우 엄청나게 많은 객체를 만들어야 함은 물론 번거로움이 가중된다.
  - 구조적으로는 좋은 구조일 수 있으나 유지와 소스의 복잡성이 증가함

Spring

Spring 주요 요소

빈(bean, bean Object)	- 스프링이 IoC 방식으로 관리하는 오브젝트 - 스프링을 사용하는 application에서 생성되는 모든 오브젝트가 빈(bean)은 아니며, 그 중에서 스프링이 직접 생성과 제어를 담당하는 Object만을 지칭한다.
빈 팩토리 (bean factory)	- 스프링의 IoC를 담당하는 핵심 컨테이너 - 빈을 등록, 생성, 조회, 반납 등의 빈 관리 기능을 담당한다. - 보통은 Bean factory 를 바로 사용하지 않고, 이를 확장한 application context를 이용한다. - Bean factory가 구현하는 가장 기본적인 인터페이스 이름은 BeanFactory이며, 여기에 getBean과 같은 메소드가 정의되어 있다.
Application context	- Bean Factory 를 확장한 IoC 컨테이너 - Bean Factory 와 동일한 Bean 관리 기능 + 추가적인 부가 서비스 제공 - ApplicationContext는 BeanFactory를 상속한다.
설정정보 / 설정 메타정보 (configuration metadata)	- Bean Factory 혹은 Application Context가 IoC를 적용하기 위해 사용하는 메타정보 - Container의 어떤 기능을 세팅하고 조정하는 경우, 혹은 IoC 컨테이너에 의해 관리되는 애플리케이션 오브젝트를 생성하고 구성할 때 사용된다.
컨테이너 (IoC container)	- BeanFactory 혹은 ApplicationContext를 지칭하는 IoC 개념의 표현

- applicationContext 상속구조



Maven

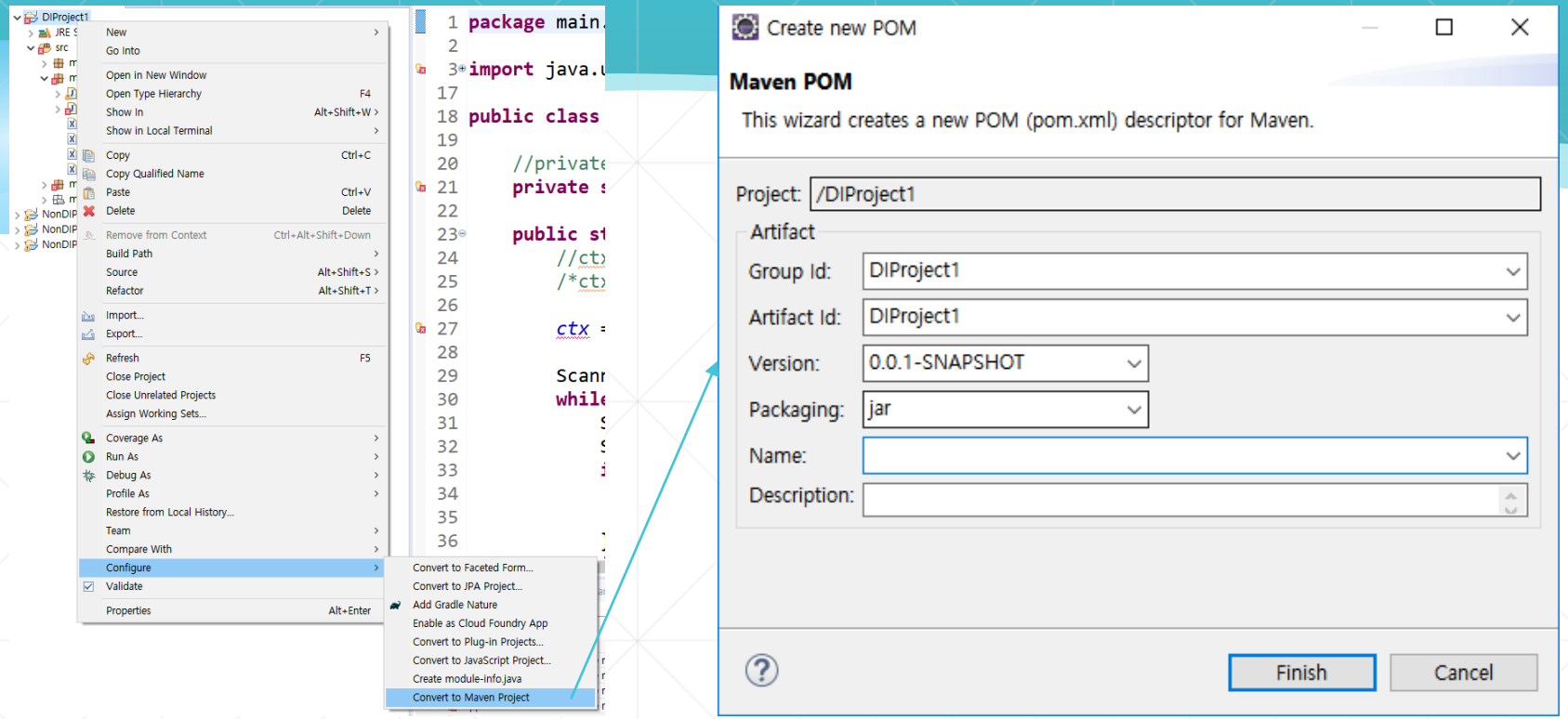
- Maven
  - 아파치 소프트웨어 재단에서 개발하는 Java 기반 프로젝트의 라이프사이클 관리를 위한 빌드 도구.
  - 이에 따라 컴파일과 빌드를 동시에 수행, 테스트를 병행하거나 서버 측 Deploy 자원을 관리할 수 있는 환경을 제공한다.
  - 또한 라이브러리 관리 기능도 내포하고 있다. Java로 개발하다 보면 다양한 라이브러리를 필요로 하게 되는데, pom.xml 파일에 필요한 라이브러리만 적으면 Maven이 알아서 다운받고 설치해주고 경로까지 지정해준다.

Maven

- Maven 장점
  - 컴파일과 빌드를 동시에 수행할 수 있다.
  - 서버의 Deploy 자원을 관리할 수 있는 환경을 제공한다.
  - pom.xml 파일을 통해 관리하므로 개발, 유지보수 측면에서 오픈소스 라이브러리, 프로젝트 등 관리가 용이하다.
  - IDE에 종속된 부분들을 제거할 수 있다.
  - Maven Profile 기능을 통해 배포 설정 파일을 관리하고 배포 파일을 생성할 수 있다.

Maven

- Maven 단점
  - Maven에서 기본적으로 지원하지 않는 빌드 과정을 추가해야 하는 경우 상당한 고생이 따른다.
  - 특정 플러그인이 설정이 약간만 달라도 해당 설정을 분리해서 중복 기술할 때가 발생한다. 불필요하게 설정이 길어지고 중복, 가독성 저하가 발생하여 유지보수성을 떨어뜨린다.
  - 이와 같은 단점을 해결하기 위해, Gradle이라는 새로운 빌드 툴이 등장하였다. Gradle은 안드로이드 애플리케이션의 기본 빌드 툴로 채택되었다.



Maven

일반 프로젝트 Maven으로 Build 하기

Maven

- pom.xml
  - Project Object Model을 설정하는 부분으로 프로젝트 내 빌드 옵션을 설정하는 부분

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <modelVersion>4.0.0</modelVersion>
  <groupId>DIProject1</groupId>
  <artifactId>DIProject1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <resources>
      <resource>
        <directory>src</directory>
        <excludes>
          <exclude>**/*.java</exclude>
        </excludes>
      </resource>
    </resources>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
          <release>10</release>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Maven

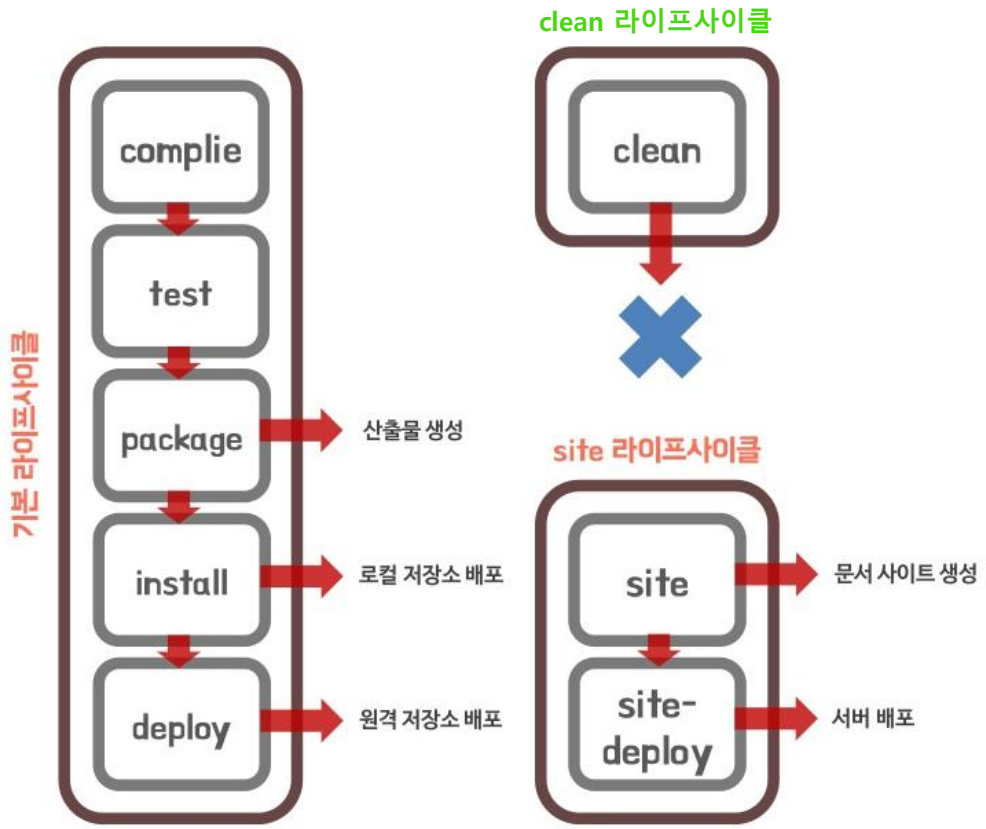
pom.xml project tags

<modelVersion>	maven의 pom.xml의 모델 버전
<groupId>	프로젝트를 생성한 조직 또는 그룹 명
<artifactId>	프로젝트에 할당한 고유 ID
<version>	애플리케이션의 버전. 접미사로 SNAPSHOT이 붙으면 아직 개발단계라는 의미
<packaging>	jar, war, ear, pom등 패키지 유형
<name>	프로젝트 명
<description>	프로젝트 설명
<url>	프로젝트를 찾을 수 있는 URL

Maven

pom.xml project tags

<properties>	pom.xml에서 중복해서 사용되는 설정(상수) 값들을 지정하는 부분. 다른 위치에서 ${...}로 표기해서 사용할 수 있다.
<groupId>	프로젝트를 생성한 조직 또는 그룹 명



Maven

Maven LifeCycle

Maven

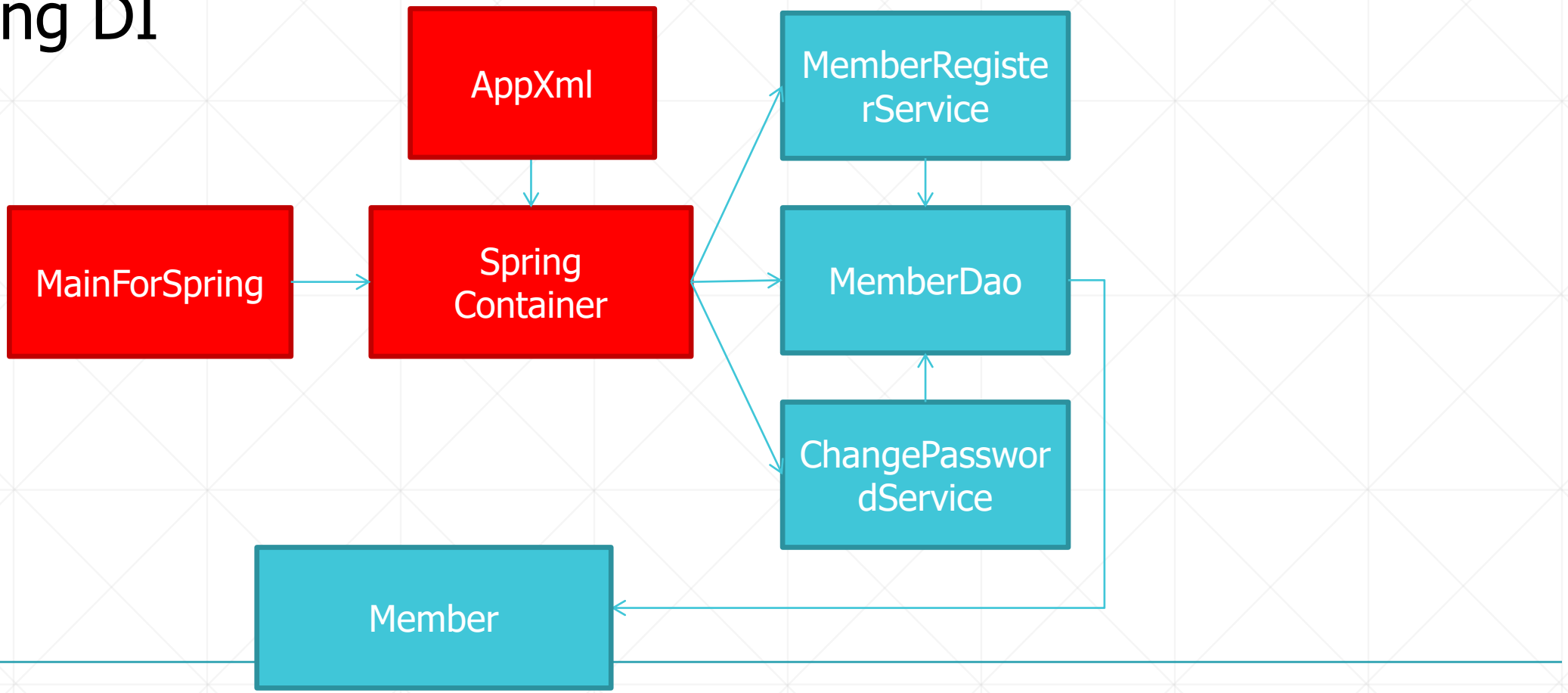
- Maven LifeCycle
  - 객체의 생명주기처럼 maven에는 라이프 사이클이 존재한다.
  - 크게 default, clean, site 라이프 사이클로 나누고 세부적으로 phase가 있다
  - mvn process-resources : resources:resources의 실행으로 resource 디렉토리에 있는 내용을 target/classes로 복사한다.
  - mvn compile : compiler:compile의 실행으로 src/java 밑의 모든 자바 소스를 컴파일해서 target/classes로 복사
  - mvn process-testResources, mvn test-compile : 이것은 위의 두 개가 src/java였다면 test/java의 내용을 target/test-classes로 복사. (참고로 test만 mvn test 명령을 내리면 라이프사이클상 원본 소스로 컴파일된다.)
  - mvn test : surefire:test의 실행으로 target/test-classes에 있는 테스트케이스의 단위테스트를 진행한다. 결과를 target/surefire-reports에 생성한다.

Maven

- Maven LifeCycle
  - mvn package : target디렉토리 하위에 jar, war, ear등 패키지파일을 생성하고 이름은 <build>의 <finalName>의 값을 사용한다 지정되지 않았을 때는 아까 설명한 "artifactId-version.extention" 이름으로 생성
  - mvn install : 로컬 저장소로 배포
  - mvn deploy : 원격 저장소로 배포
  - mvn clean : 빌드 과정에서 생긴 target 디렉토리 내용 삭제
  - mvn site : target/site에 문서 사이트 생성
  - mvn site-deploy : 문서 사이트를 서버로 배포

Spring DI

- Spring DI



Spring DI

- pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSc
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.co.sample</groupId>
  <artifactId>Prj01_4DIProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <!-- xml에서 사용할 속성들 -->
    <properties>
        <!-- 자바 버전 -->
        <java-version>16</java-version>
        <!-- 스프링 버전 -->
        <org.springframework-version>5.2.18.RELEASE</org.springframework-version>
        <!--<org.springframework-version>4.3.25.RELEASE</org.springframework-version> -->
        <org.slf4j-version>1.7.32</org.slf4j-version>
        <ch.qos.logback-version>1.2.6</ch.qos.logback-version>
    </properties>
```

```xml
    <!-- 프로젝트에서 사용할 라이브러리 정보 -->
    <dependencies>
        <!-- spring context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${org.springframework-version}</version>
        </dependency>
        <!-- slf4j -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${org.slf4j-version}</version>
        </dependency>
        <!-- logback -->
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>${ch.qos.logback-version}</version>
            <exclusions>
                <exclusion>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-api</artifactId>
                </exclusion>
            </exclusions>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
</project>
```

Spring DI

- Spring DI
  - AppCtx.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">

    <bean id="memberDao" class="main.java.spring.MemberDao">
    </bean>

    <bean id="memberRegSvc" class="main.java.spring.MemberRegisterService">
        <constructor-arg ref="memberDao"/>
    </bean>

    <bean id="changePwdSrc" class="main.java.spring.ChangePasswordService">
        <constructor-arg ref="memberDao"/>
    </bean>
```

Spring DI

- Spring DI
  - MainForSpring.java(1)

```
public class MainForSpring {

    //private static Assembler assembler = new Assembler();
    private static ClassPathXmlApplicationContext ctx = null; // 추가

    public static void main(String[] args) {
        ctx = new ClassPathXmlApplicationContext("main/java/main/appCtx.xml"); // 추가

        Scanner scan = new Scanner(System.in);
        while(true){
            System.out.println("명령어를 입력하세요");
            String command = scan.nextLine();
            if(command.equalsIgnoreCase("exit")){
                System.out.println("종료합니다");
                break;
```

Spring DI

- Spring DI
  - MainForSpring .java(2)

```
private static void processChargeCommand(String[] arg) {
    if(arg.length!=4){
        printHelp();
        return;
    }
    //ChangePasswordService changePwdService = assembler.getPwdSvc();
    ChangePasswordService changePwdService =
            ctx.getBean("changePwdSrc", ChangePasswordService.class);
    try {
        changePwdService.changePassword(arg[1], arg[2], arg[3]);
        System.out.println("암호를 변경했습니다.");
    } catch (MemberNotFoundException e1) {
        System.out.println("존재하지 않는 이메일입니다.\n");
    } catch (IdPasswordNotMatchingException e2) {
        System.out.println("이메일과 암호가 일치하지 않습니다.\n");
    }

}
```

Spring DI

- Spring DI
  - MainForSpring.java(3)

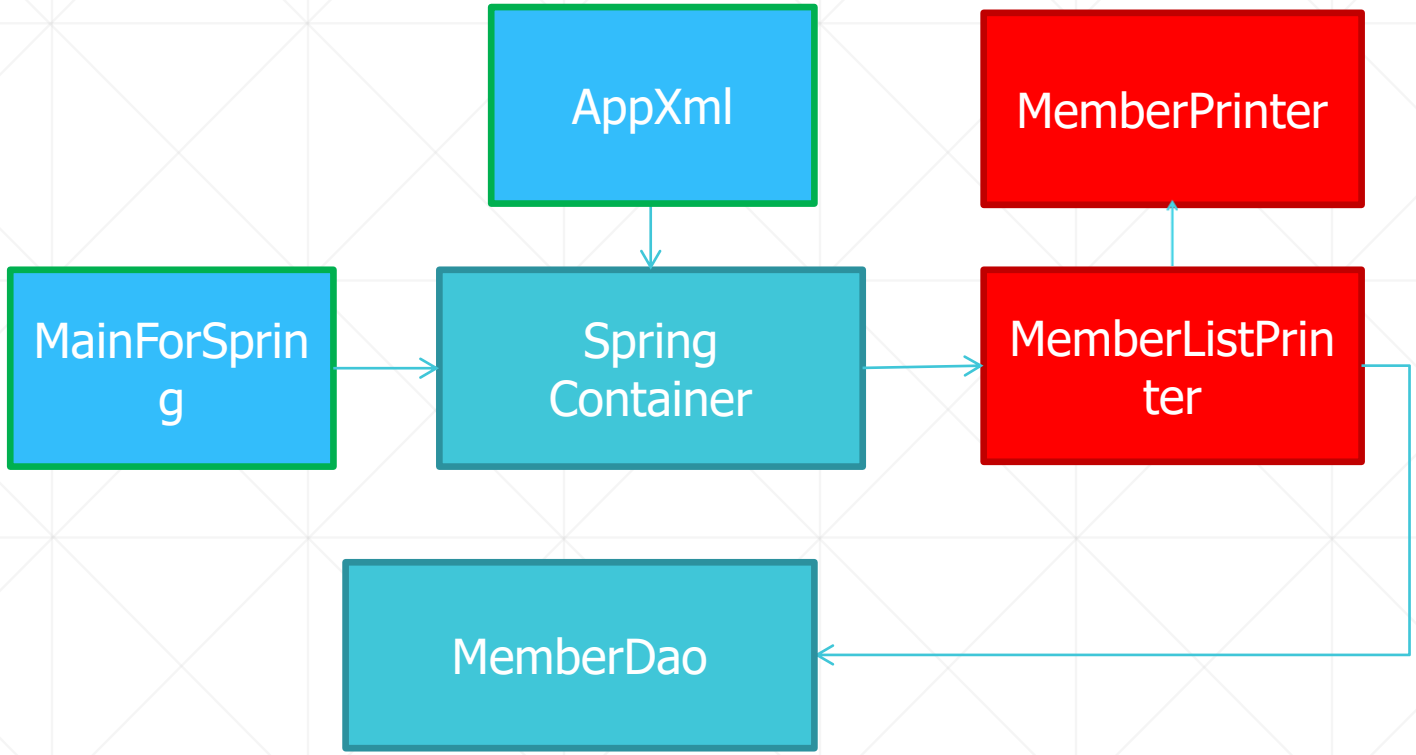
```
private static void processNewCommand(String[] arg) {
    if(arg.length!=5){
        printHelp();
        return;
    }
    //MemberRegisterService regSvc = assembler.getRegSvc();
    MemberRegisterService regSvc =
            ctx.getBean("memberRegSvc", MemberRegisterService.class);
    RegisterRequest req = new RegisterRequest();
    req.setEmail(arg[1]);
    req.setName(arg[2]);
    req.setPassword(arg[3]);
    req.setConfirmPassword(arg[4]);

    if(!req.isPasswordEqualToConfirmPassword()){
        System.out.println("암호가 일치하지 않습니다.");
        return;
    }
```

Spring DI Constructor

- DI 생성자 주입
  - DI 주입 시 생성자를 이용하여 Bean에 주입할 수 있다.
  - 생성자를 이용하는 방식은 xml bean태그들 아래에 <constructor-arg>태그를 통해 삽입할 수 있다.
  - 만약 두 개 이상의 생성자 메서드를 받을 경우 <constructor-arg>를 사용하여 여러 개를 받을 수 있다

Spring DI Constructor

- DI 생성자 주입



Spring DI Constructor

- DI 생성자 주입
  - MemberPrinter.java

```
public class MemberPrinter {
    public void print(Member member){
        System.out.println("회원정보 : 아이디 ="+member.getId()+
                ", 이메일="+member.getEmail()+", 이름="+member.getName()+
                ", 등록일="+member.getRegisterDate());
    }
}
```

Spring DI Constructor

- DI 생성자 주입
  - MemberListPrinter.java

```
public class MemberListPrinter {
    private MemberDao memberDao;
    private MemberPrinter printer;
    public MemberListPrinter(MemberDao memberDao,
                             MemberPrinter printer) {
        this.memberDao = memberDao;
        this.printer = printer;
    }

    public void printAll() {
        Collection<Member> members = memberDao.selectAll();
        for (Member m : members) {
            printer.print(m);
        }
    }
}
```

Spring DI Constructor

- DI 생성자 주입
  - appCtx.xml

```
<bean id="memberDao" class="main.java.spring.MemberDao">
</bean>
<!-- 추가 -->
<bean id="memberPrinter" class="main.java.spring.MemberPrinter">
</bean>

<bean id="memberRegSvc" class="main.java.spring.MemberRegisterService">
    <constructor-arg ref="memberDao"/>
</bean>

<bean id="changePwdSrc" class="main.java.spring.ChangePasswordService">
    <constructor-arg ref="memberDao"/>
</bean>
<!-- 추가 -->
<bean id="listPrinter" class="main.java.spring.MemberListPrinter">
    <constructor-arg ref="memberDao"/>
    <constructor-arg ref="memberPrinter"/>
</bean>
```

Spring DI Constructor

- DI 생성자 주입
  - MainForSpring.java(1)

```
            }
            if(command.startsWith("new")){
                processNewCommand(command.split(" "));
                continue;
            }else if(command.startsWith("change")){
                processChargeCommand(command.split(" "));
                continue;
            }else if(command.startsWith("list")){//추가된 부분
                processListCommand();
                continue;
            }
            printHelp();
        }
}
```

Spring DI Constructor

- DI 생성자 주입
  - MainForSpring.java(2)

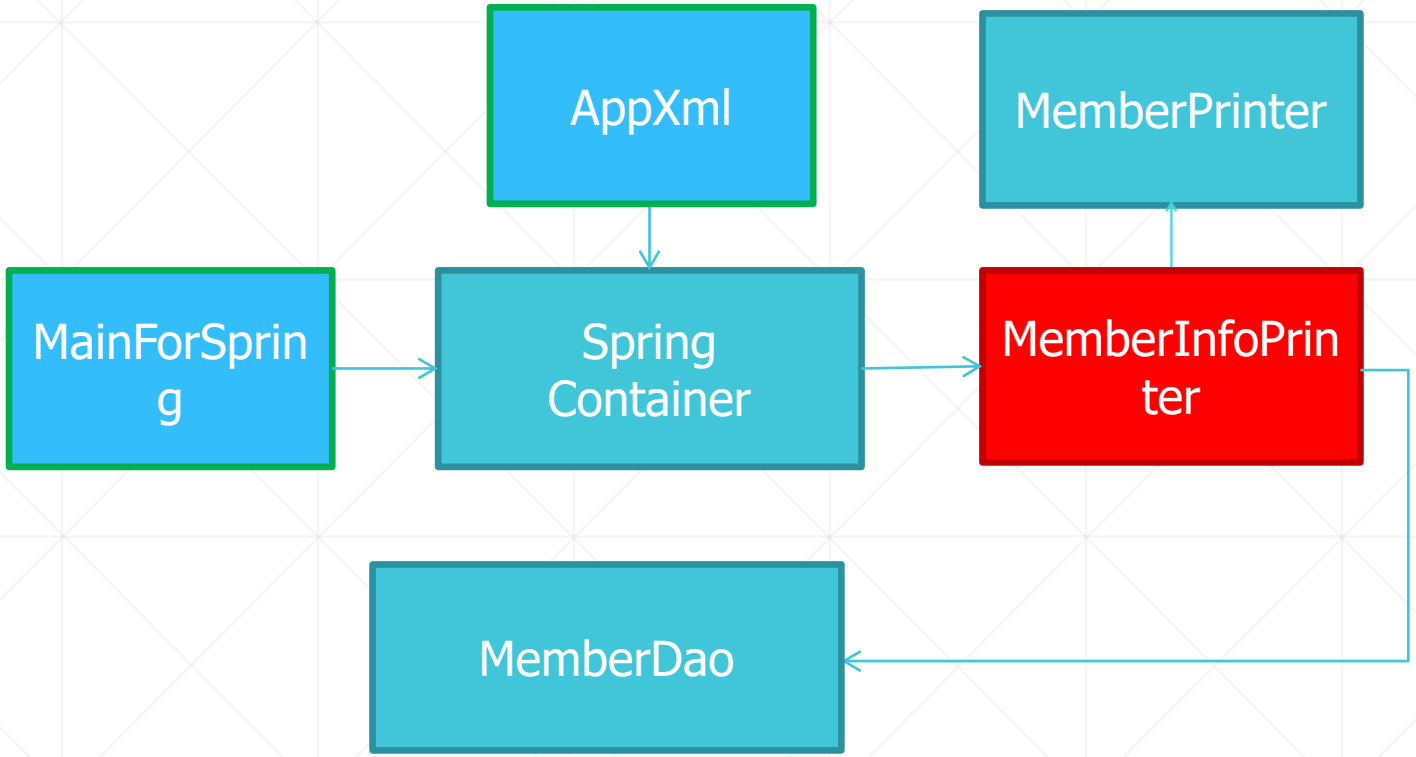
```
private static void processListCommand() {
    MemberListPrinter listPrinter=
            ctx.getBean("listPrinter",MemberListPrinter.class);
    listPrinter.printAll();
}
```

Spring DI Setter

- Setter 설정 방식
  - Setter를 메서드를 통해 객체를 삽입하는 방식이 존재한다
  - 사용하기 위해서는 반드시 클래스 안에 Setter메서드가 구현되어 있어야 한다.
  - 이 방식은 xml bean태그들 아래에 <property> 태그를 통해 삽입할 수 있다.

Spring DI Setter

- Setter 설정 방식



Spring DI Setter

- Setter 설정 방식
  - MemberInfoPrinter.java

```
public class MemberInfoPrinter {
    private MemberDao memDao;
    private MemberPrinter printer;
    public void setMemberDao(MemberDao memDao) {
        this.memDao = memDao;
    }
    public void setPrinter(MemberPrinter printer) {
        this.printer = printer;
    }

    public void printMemberInfo(String email){
        Member member = memDao.selectByEmail(email);
        if(member==null){
            System.out.println("데이터 없음\n");
            return;
        }
        printer.print(member);
        System.out.println();
    }
}
```

Spring DI Setter

- Setter 설정 방식
  - appCtx.xml

```
<bean id="infoPrinter" class="main.java.spring.MemberInfoPrinter">
    <property name="memberDao" ref="memberDao"/>
    <property name="printer" ref="memberPrinter"/>
</bean>
```

Spring DI Setter

- Setter 설정 방식
  - MainForSpring.java(1)

```
            processNewCommand(command.split(" "));
            continue;
        }else if(command.startsWith("change")){
            processChargeCommand(command.split(" "));
            continue;
        }else if(command.startsWith("list")){
            processListCommand();
            continue;
        }else if(command.startsWith("info")){
            processInfoCommand(command.split(" "));
            continue;
        }
```

Spring DI Setter

- Setter 설정 방식
  - MainForSpring.java(2)

```
private static void processInfoCommand(String[] arg) {
    if(arg.length!=2){
        printHelp();
        return;
    }
    MemberInfoPrinter infoPrinter =
            ctx.getBean("infoPrinter",MemberInfoPrinter.class);
    infoPrinter.printMemberInfo(arg[1]);
}
```

Spring DI Validation

- DI 기본 데이터 설정
  - 만약 기본 데이터를 넣고자 할 경우 ref 대신에 value를 이용해서 집어넣을 수 있다.
  - 해당 태그는 constructor 태그나 property안에 속성으로 넣을 수 있고 태그 아래 <value>태그를 넣어 삽입이 가능하다.
  - 값은 타입이 같지 않더라도 스프링 컨테이너에서 해당 타입에 맞게 변환시켜 삽입한다.

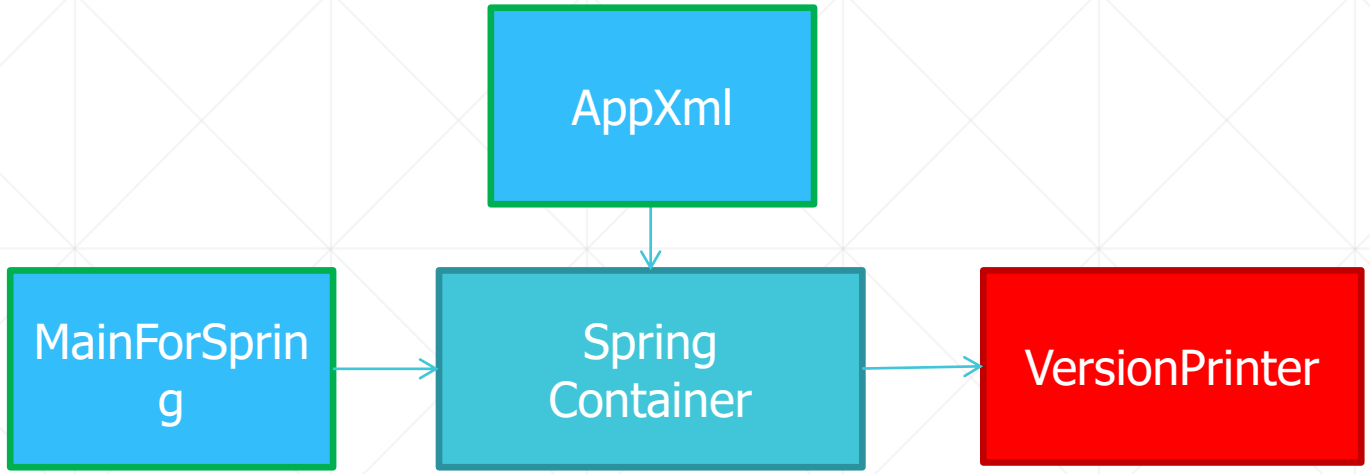
Spring DI Setter

- Setter 설정 방식
  - Setter 설정 하고자 하는 bean 객체에 autowire 라는 속성을 붙여 넣을 수 있다.
  - autowire 라는 속성에 “byName”을 붙여 넣게 되면 빈의 매개변수의 이름과 id가 같은 빈끼리 묶어준다.

```
<!-- <bean id="infoPrinter" class="main.java.spring.MemberInfoPrinter">
    <property name="memberDao" ref="memberDao"/>
    <property name="printer" ref="memberPrinter"/>
</bean> -->

<bean id="infoPrinter" class="main.java.spring.MemberInfoPrinter" autowire="byName"/>
```

Spring DI Validation

- DI 기본 데이터 설정



Spring DI Validation

- DI 기본 데이터 설정
  - VersionPrinter.java

```
public class VersionPrinter {
    private int majorVersion;
    private int minorVersion;
    public void setMajorVersion(int majorVersion) {
        this.majorVersion = majorVersion;
    }
    public void setMinorVersion(int minorVersion) {
        this.minorVersion = minorVersion;
    }

    public void print(){
        System.out.println(
                "이 프로그램의 버전은"+majorVersion+
                "."+minorVersion+"입니다.");
    }
}
```

Spring DI Validation

- DI 기본 데이터 설정
  - appCtx.xml

```
<bean id="versionPrinter" class="main.java.spring.VersionPrinter">
    <property name="majorVersion" value="5"/>
    <property name="minorVersion">
        <value>2</value>
    </property>
</bean>
```

Spring DI Validation

- DI 기본 데이터 설정
  - MainForSpring.java(1)

```
        }else if(command.startsWith("list")){
            processListCommand();
            continue;
        }else if(command.startsWith("info")){
            processInfoCommand(command.split(" "));
            continue;
        }else if(command.startsWith("version")){
            processVersionCommand();
            continue;
        }
        printHelp();
    }
```

Spring DI Validation

- DI 기본 데이터 설정
  - MainForSpring.java(2)

```
private static void processVersionCommand() {
    VersionPrinter versionPrinter =
            ctx.getBean("versionPrinter",VersionPrinter.class);
    versionPrinter.print();

}
```

Spring with singleton

- Singleton Object
  - 스프링은 별도 설정을 하지 않을 경우 한 개의 빈 객체만을 생성하며, 이들 빈 객체들이 "싱글톤(Singletion)" 범위를 갖는다.
  - 싱글톤은 단일 객체(single object)를 의미하는 단어로 스프링은 기본적으로 한 개의 <bean> 태그에 대해 한 개의 빈 객체를 생성한다.

```
private static void processVersionCommand() {
    VersionPrinter versionPrinter1 =
            ctx.getBean("versionPrinter",VersionPrinter.class);
    VersionPrinter versionPrinter2 =
            ctx.getBean("versionPrinter",VersionPrinter.class);
    System.out.println("versionPrinter1 == versionPrinter2 : "
            + (versionPrinter1 == versionPrinter2));
    versionPrinter1.print();
}
```

Spring with singleton

- Singleton Object

```
versionPrinter1 == versionPrinter2 : true
이 프로그램의 버전은5.2입니다.
명령어를 입력하세요
```

Spring DI conf.xml 분할 지정

- DI conf.xml 분할 지정
  - conf.xml을 두개를 통해서 설정을 분할 지정이 가능하다.
  - conf1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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">

    <bean id="memberDao" class="main.java.spring.MemberDao">
    </bean>

    <bean id="memberPrinter" class="main.java.spring.MemberPrinter">
    </bean>

</beans>
```

Spring DI conf.xml 분할 지정

- DI conf.xml 분할 지정
  - conf2.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">

    <bean id="memberRegSvc" class="main.java.spring.MemberRegisterService">
        <constructor-arg ref="memberDao"/>
    </bean>

    <bean id="changePwdSrc" class="main.java.spring.ChangePasswordService">
        <constructor-arg ref="memberDao"/>
    </bean>
    <bean id="listPrinter" class="main.java.spring.MemberListPrinter">
        <constructor-arg ref="memberDao"/>
        <constructor-arg ref="memberPrinter"/>
    </bean>
    <bean id="infoPrinter" class="main.java.spring.MemberInfoPrinter">
        <property name="memberDao" ref="memberDao"/>
        <property name="printer" ref="memberPrinter"/>
    </bean>
    <bean id="versionPrinter" class="main.java.spring.VersionPrinter">
        <property name="majorVersion" value="5"/>
        <property name="minorVersion">
            <value>2</value>
        </property>
    </bean>
</beans>
```

Spring DI conf.xml 분할 지정

- DI conf.xml 분할 지정
  - SpringForMain
    - GenericXmlApplicationContext에서 xml파일을 두개 이상 지정이 가능하다.

```
ctx = new ClassPathXmlApplicationContext("classpath:/main/java/main/conf1.xml",
        "classpath:/main/java/main/conf2.xml");
```

Spring DI import.xml

- DI xml import setting
  - xml 안에 import 설정을 넣음으로써 xml의 설정을 다른곳에서 받아 쓸 수가 있다.
  - conf1.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">

    <import resource="classpath:/main/java/main/conf2.xml"/>

    <bean id="memberDao" class="main.java.spring.MemberDao">
    </bean>

    <bean id="memberPrinter" class="main.java.spring.MemberPrinter">
    </bean>

</beans>
```

Spring DI import.xml

- DI xml import setting
  - SpringForMain

```
ctx = new ClassPathXmlApplicationContext("classpath:/main/java/main/conf1.xml");
```

학습정리

- 자바 프로젝트에서 클래스가 변경되면 객체 생성 부분과 참조 변수 선언 부분 모두를 수정해야 한다.