03강 의존 자동 주입

예제 준비

다음의 정보로 메이븐 프로젝트를 생성하자

- Group ID : com.green
- Artifact ID : ex03

기존의 프로젝트에서 POM.xml 을 복사해온 다음 메이븐 업데이트를 진행한다.

예제 준비

=> ex02프로젝트에서 다음 클래스를 복사해온다.

-spring.exception
- AlreadyExistingMemberException.java
- IdPasswordNotMatchingException.java

-spring.vo
- Member.java
- RegisterRequest.java

-spring.dao
- MemberDao.java

-spring.service
- MemberRegisterService.java

-spring.printer
- MemberPrinter.java
- MemberInfoPrinter.java

-spring.main

자동 주입의 개념

- 기존의 의존관계를 명시해서 의존 주입 기능을 사용할 수도 있지만 스프링은 보다 편리하게 자동 주입 기능을 제공해 준다.

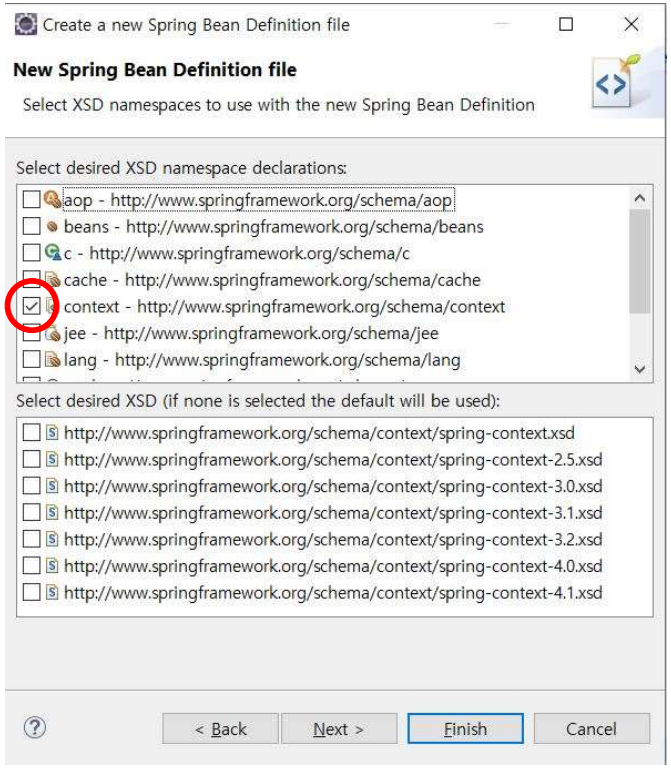
- 자동 주입 기능을 사용하기 위해서
  - 자동 주입 대상이 되는 클래스에 @Autowired 애노테이션을 사용
  - XML 설정에 <context:annotation-config /> 태그 추가

자동 주입의 개념

- 예제 추가해본다
  - appCtx1.xml
  - MemberRegisterService 추가
  - MemberInfoPrinter 추가
  - Main01

자동 주입의 개념

- appCtx1.xml
- 스프링 컨피그 파일 생성시 context를 체크한다



자동 주입의 개념

- appCtx1.xml

자동 주입 어노테이션을 사용하기 위해 필요

```
<context:annotation-config />

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

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

<bean id="memberprinter" class="spring.printer.MemberPrinter">
</bean>

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

자동 주입의 개념

- MemberRegisterService 추

1) xml에서 생성자 주입 제거

```
<bean id="memberRegSvc" class="spring.service.MemberRegisterService">
    <!-- <constructor-arg ref="memberDao" /> -->
</bean>
```

2) 자동 주입 어노테이션 추가 : @Autowired

```
@Autowired
public MemberRegisterService(MemberDao memberDao) {
    this.memberDao = memberDao;
}
```

자동 주입의 개념

- MemberInfoPrinter 추가

1) xml 수정

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

2) 자동 주입 어노테이션 추가 : @Autowired

```
    @Autowired
    private MemberDao memDao;

    private MemberPrinter printer;

//  public void setMemberDao(MemberDao memberDao) {
//      this.memDao = memberDao;
//  }

    @Autowired
    public void setPrinter(MemberPrinter printer) {
        this.printer = printer;
    }
```

자동 주입의 개념

- Main01

```
public class Main01 {

    public static void main(String[] args) {
        ApplicationContext ctx =
                new GenericXmlApplicationContext("classpath:appCtx01.xml");

        MemberRegisterService regSvc =
                ctx.getBean("memberRegSvc",MemberRegisterService.class);

        MemberInfoPrinter info =
                ctx.getBean("infoPrinter",MemberInfoPrinter.class);

        RegisterRequest regReq = new RegisterRequest();
        regReq.setEmail("hong@naver.com");
        regReq.setName("홍길동");
        regReq.setPassword("1234");
        regReq.setConfirmPassword("1234");

        regSvc.regist(regReq);

        info.printMemberInfo("hong@naver.com");

    }
}
```

자동 주입의 개념

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

<bean id="memberRegSvc"
class="spring.MemberRegisterService">
</bean>

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

<bean id="infoPrinter" class="spring.MemberInfoPrinter">
</bean>
```

```
@Autowired
public MemberRegisterService(MemberDao memberDao) {
        this.memberDao = memberDao;
}
```

```
@Autowired
private MemberDao memDao;

@Autowired
public void setPrinter(MemberPrinter printer) {
        System.out.println("setPrinter: " + printer);
        this.printer = printer;
}
```

빈 객체가 두개라면??

- @Autowired 애노테이션을 적용시 자동으로 주입할 객체를 선택해준다.

- 그러나 동일한 타입의 주입할 객체가 두개 이상이라면
에러를 발생시킨다.

예제2

- appCtx1.xml 수정

```
<!--    <bean id="memberPrinter" class="spring.printer.MemberPrinter">
    </bean> -->

    <bean id="printer1" class="spring.printer.MemberPrinter" />

    <bean id="printer2" class="spring.printer.MemberPrinter" />
```

- 에러 확인

빈 객체가 두개라면??

- 해결 방법
- @Qualifier 애노테이션을 사용한다.

 - 빈의 한정자를 지정한다.
 - @Autowired 로 주입된 대상에 @Qualifier의 값으로 한정자를 지정해준다.

빈 객체가 두개라면??

예제3

- appCtx1.xml 수정

```
<bean id="printer1" class="spring.printer.MemberPrinter" >
    <qualifier value="chk01"/>
</bean>
```

- MemberInfoPrinter 수정

```
@Autowired
@Qualifier("chk01")
public void setPrinter(MemberPrinter printer) {
    this.printer = printer;
}
```

@Autowired의 필수 여부 지정

- @Autowired 애노테이션을 적용하면 반드시 주입할 의존 객체가 있어야 한다

- 주입할 의존객체가 없는 경우의 예제3
  - appCtx1.xml 수정 => MemberDao가 없는 경우

```
<!--    <bean id="memberDao" class="spring.dao.MemberDao">
    </bean>
 -->
```

  - 에러 확인

@Autowired의 필수 여부 지정

- 해결 방법

⇒@Autowired뒤에 (required=false)를 붙인다.

=> 필수요건을 완화시킴

예제

- MemberInfoPrinter수정

```
public class MemberInfoPrinter {

    @Autowired(required = false)
    private MemberDao memDao;
```

- MemberRegisterService 수정

```
@Autowired(required = false)
public MemberRegisterService(MemberDao memberDao) {
    this.memberDao = memberDao;
}
```

=> required=false를 붙이면 의존 객체가 없는 경우 기본 객체를 붙인다.

@Autowired 애노테이션 적용 순서

- @Autowired 애노테이션 적용 순서

1) 타입이 같은 빈 객체를 검색
2) 타입이 같은 빈 객체가 두개 이상 존재 시 @Qualifier로 지정한 빈 객체를 검색
3) @Qualifier가 없는 경우 이름이 같은 빈 객체 검색

위 3가지 경우가 아닌 경우는 예외를 발생 시킨다.

@Autowired 애노테이션과 파라미터 개수

- @Autowired 애노테이션의 적용 대상 파라미터가 여러 개인 경우에도 사용이 가능하다.

```
private MemberDao memDao;
private MemberPrinter printer;

@Autowired
public void injectDependency(MemberDao memberDao, MemberPrinter printer) {
        this.memDao = memberDao;
        this.printer = printer;
}
```

- 여러 개의 파라미터 중 개별적으로 의존 객체를 지정해줄 수도 있다

```
private MemberDao memDao;
private MemberPrinter printer;

@Autowired
public void injectDependency(MemberDao memberDao, @Qualifier("sysout") MemberPrinter printer) {
        this.memDao = memberDao;
        this.printer = printer;
}
```

@Resource 애노테이션을 이용한 자동 주입

- @Autowired 애노테이션이 타입을 이용해서 주입할 객체를 검색한다면
- @Resource 애노테이션은 빈의 이름을 이용해서 주입할 객체를 검색한다.

- 사용법
  - 자동 주입 대상에 @Resource 애노테이션 사용
  => @Resource(name="의존 객체 이름")
  - XML 설정에 <context:annotation-config /> 적용

@Resource 애노테이션을 이용한 자동 주입

- 예제 4

  - MemberRegisterService 수정

```
public class MemberRegisterService {

    @Resource(name="memberDao")
    private MemberDao memberDao; // = new MemberDao();

//  @Autowired(required = false)
//  public MemberRegisterService(MemberDao memberDao) {
//      this.memberDao = memberDao;
//  }
```

@Resource 애노테이션을 이용한 자동 주입

- 예제 4
  - MemberInfoPrinter 수정

```
public class MemberInfoPrinter {

//  @Autowired(required = false)
    @Resource(name="memberDao")
    private MemberDao memDao;

    private MemberPrinter printer;

//  public void setMemberDao(MemberDao memberDao) {
//      this.memDao = memberDao;
//  }

//  @Autowired
//  @Qualifier("check01")
    @Resource(name="printer1")
    public void setPrinter(MemberPrinter printer) {
        this.printer = printer;
    }
```

@Resource 애노테이션을 이용한 자동 주입

- 다만 @Resource 애노테이션은 생성자에는 사용할 수 없다.
- 또한 @Resource 애노테이션이 올바르게 동작하려면 name 속성으로 지정한 빈 객체가 존재해야 한다.

@Resource 애노테이션의 적용 순서

1) name 속성에 지정한 빈 객체를 찾는다.
2) name 속성이 없는 경우 동일한 타입의 빈 객체를 찾는다.
3) name 속성이 없고 동일한 타입의 빈 객체가 두개 이상인 경우 같은 이름을 가진 객체를 찾는다.
4) name 속성이 없고 동일한 타입의 빈 객체가 두개 이상인 경우 같은 이름을 가진 객체가 없는 경우 @Qualifier를 이용해서 주입할 빈 객체를 찾는다.

위 4가지 경우가 아닌 경우는 예외를 발생 시킨다.

자동 주입과 명시적 주입간의 관계

- 자동 주입과 명시적 주입을 함께 사용하는 것이 가능하다.
- 다만 두개가 동시에 사용되는 경우 명시적 주입을 우선 사용한다.

- appCTX01.xml

```
<bean id="printer" class="spring.printer.MemberPrinter">
</bean>
<bean id="printer1" class="spring.printer.MemberPrinter">
</bean>

<bean id="infoPrinter" class="spring.printer.MemberInfoPrinter">
    <property name="printer" ref="printer1" />
</bean>
```

- MemberInfoPrinter

```
@Autowired
public void setPrinter(MemberPrinter printer) {
    this.printer = printer;
}
```

=> 자동 주입 원칙상 이름이 같은 printer이 주입되어야 하지만 명시적으로 printer1 을 주입하라고 하기 때문에 최종적으로 printer1 이 주입된다.