

올인원 스프링 프레임워크



Chapter 04

IoC 컨테이너



목차

1. 예제 프로젝트 준비: 학사 관리 시스템
2. 스프링 설정 파일
3. 메이븐 설정 파일
4. MainClass.java
5. 스프링 설정 파일 분리
6. 스프링 빈 범위

학습목표

- 빈을 생성하는 방법에 대해서 학습합니다.
- 멤버 필드를 초기화하는 방법에 대해서 학습합니다.
- 스프링 설정 파일을 분리하는 방법에 대해서 학습합니다.
- 빈의 범위를 알아봅니다.

Section 01

예제 프로젝트 준비:
학사 관리 시스템

1. 학사 관리 시스템의 개요

■ 학사 관리 시스템(EMS)

- 학생 관리(학생 등록, 조회, 수정, 삭제 등) 부분을 구현한 프로그램
- 프로젝트의 시나리오
 - ① 학생 5명의 샘플 데이터를 데이터베이스에 등록한다.
 - ② 전체 학생 정보를 출력한다.
 - ③ 새로운 학생 정보를 등록한다.
 - ④ 학생 정보를 수정한다.
 - ⑤ 학생 정보를 삭제한다.
 - ⑥ 시스템 정보를 출력한다.
- 프로젝트의 구조

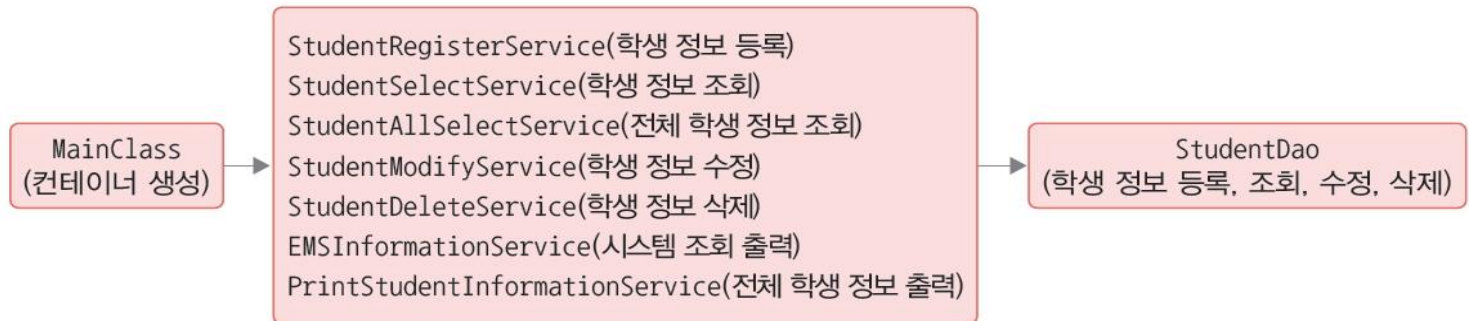


그림 4-1 학사 관리 시스템 프로그램의 흐름도

1. 학사 관리 시스템의 개요

■ 사용할 클래스와 IoC 컨테이너

MainClass 클래스	
MainClass	main() 메서드가 명시되어 있는 클래스
DB 관련 클래스	
DBConnectionInfo	데이터베이스 정보
Student	학생 정보
Service 클래스	
EMSInformationService	시스템 정보
PrintStudentInformationService	학생 정보 출력
StudentAllSelectService	전체 학생 정보 조회
StudentDeleteService	학생 정보 삭제
StudentModifyService	학생 정보 수정
StudentRegisterService	학생 정보 등록
StudentSelectService	학생 정보 조회
DAO 클래스	
StudentDao	DAO
Util 클래스	
InitSampleData	샘플 데이터 초기화
IoC 컨테이너	
applicationContext.xml	객체 생성과 조립

그림 4-2 학사 관리 시스템에서 사용될 클래스와 IoC 컨테이너 제작에 필요한 xml 파일

2. 학사 관리 시스템 만들기

■ 학사 관리 시스템 프로젝트 생성하기

- ch04_pjt_01 프로젝트를 생성하고, [src/main/java]에 패키지 3개를 만들고 각각의 클래스 생성하기

패키지	클래스
ch04_pjt_01.ems.member	DBConnectionInfo Student
ch04_pjt_01.ems.member.dao	StudentDao
ch04_pjt_01.ems.member.service	EMSInformationService PrintStudentInformationService StudentAllSelectService StudentDeleteService StudentModifyService StudentRegisterService StudentSelectService
ch04_pjt_01.ems.utils	InitSampleData

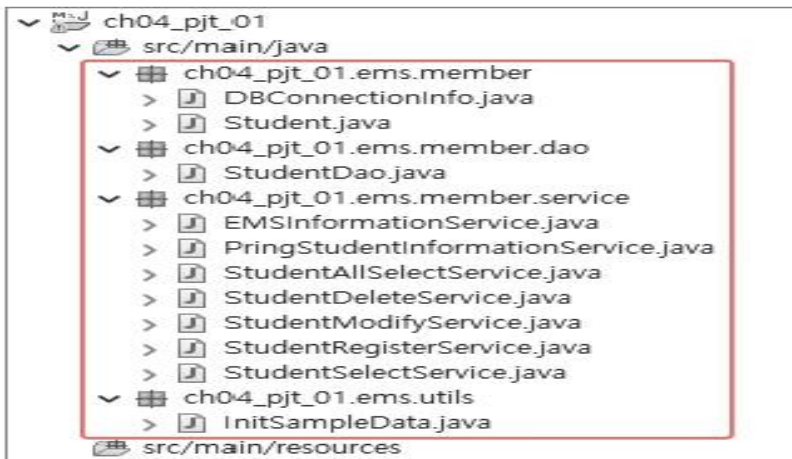


그림 4-3 생성된 패키지와 클래스의 구조

2. 학사 관리 시스템 만들기

■ Student 클래스

- 학생 정보를 담고 있음
- sNum 필드: 학생을 구분하는 학번
- 생성자에서 필드 초기화에 필요한 모든 값을 받음
- 모든 필드에 대한 getter와 setter 메서드를 가짐

코드 4-1

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\Student.java

```
01 package ch04_pjt_01.ems.member;
02
03 public class Student {
04
05     private String sNum;
06     private String sId;
07     private String sPw;
08     private String sName;
09     private int sAge;
10     private char sGender;
11     private String sMajor;
12
```

2. 학사 관리 시스템 만들기

```
13     public Student(String sNum, String sId, String sPw, String sName,
14         int sAge, char sGender, String sMajor) {
15         this.sNum = sNum;
16         this.sId = sId;
17         this.sPw = sPw;
18         this.sName = sName;
19         this.sAge = sAge;
20         this.sGender = sGender;
21         this.sMajor = sMajor;
22     }
23
24     public String getsNum() { return sNum; }
25     public void setsNum(String sNum) { this.sNum = sNum; }
26
27     public String getsId() { return sId; }
28     public void setsId(String sId) { this.sId = sId; }
29
30     public String getsPw() { return sPw; }
31     public void setsPw(String sPw) { this.sPw = sPw; }
32
33     public String getName() { return sName; }
34     public void setName(String sName) { this.sName = sName; }
35
36     public int getsAge() { return sAge; }
37     public void setsAge(int sAge) { this.sAge = sAge; }
38
39     public char getsGender() { return sGender; }
40     public void setsGender(char sGender) { this.sGender = sGender; }
41
42     public String getsMajor() { return sMajor; }
43     public void setsMajor(String sMajor) { this.sMajor = sMajor; }
44 }
```

2. 학사 관리 시스템 만들기

■ DBConnectionInfo 클래스

- 데이터베이스 연결에 필요한 정보(url, userId, userPw)를 가짐
- 모든 필드에 대한 getter와 setter 메서드를 가짐

코드 4-2

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\DBConnectionInfo.java

```
01 package ch04_pjt_01.ems.member;
02
03 public class DBConnectionInfo {
04
05     private String url;
06     private String userId;
07     private String userPw;
08
09     public String getUrl() { return url; }
10     public void setUrl(String url) { this.url = url; }
11
12     public String getUserId() { return userId; }
13     public void setUserId(String userId) { this.userId = userId; }
14
15     public String getUserPw() { return userPw; }
16     public void setUserPw(String userPw) { this.userPw = userPw; }
17
18 }
```

2. 학사 관리 시스템 만들기

■ StudentDao 클래스

- 일반적인 DAO(database access object) 클래스와 다르지 않음
- 데이터베이스에 접속하고, Service에 의해서 호출되며 데이터의 insert, search, update, delete 등의 기능을 수행함
 - ✓ 실제로는 mariaDB와 같은 데이터베이스를 사용해야 하지만, 데이터베이스를 아직 학습하지 않은 관계로 HashMap을 이용해서 데이터를 관리함
 - ✓ studentDB 필드에 학생 정보가 저장됨)
 - ✓ studentDB에서 key로 사용되는 값은 학생 정보(Student 클래스)에서 고유 값으로 부여되는 sNum(학번)을 이용함

2. 학사 관리 시스템 만들기

코드 4-3

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\dao\StudentDao.java

```
01 package ch04_pjt_01.ems.member.dao;
02
03 import java.util.HashMap;
04 import java.util.Map;
05 import ch04_pjt_01.ems.member.Student;
06
07 public class StudentDao {
08
09     private Map<String, Student> studentDB = new HashMap<String, Student>();
10
11     public void insert(Student student) {
12         studentDB.put(student.getNum(), student);
13     }
14
15     public Student select(String sNum) {
16         return studentDB.get(sNum);
17     }
18
19     public void update(Student student) {
20         studentDB.put(student.getNum(), student);
21     }
22
23     public void delete(String sNum) {
24         studentDB.remove(sNum);
25     }
26
27     public Map<String, Student> getStudentDB() {
28         return studentDB;
29     }
30
31 }
```

2. 학사 관리 시스템 만들기

■ StudentRegisterService 클래스

- 학생 정보를 데이터베이스(지금은 HashMap)에 저장하는 기능

코드 4-4

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\service\StudentRegisterService.java

```
01 package ch04_pjt_01.ems.member.service;
02
03 import ch04_pjt_01.ems.member.Student;
04 import ch04_pjt_01.ems.member.dao.StudentDao;
05
06 public class StudentRegisterService {
07
08     private StudentDao studentDao;
09
10     public StudentRegisterService(StudentDao studentDao) {
11         this.studentDao = studentDao;
12     }
13
14     public void register(Student student) {
15         if(verify(student.getsNum())) {
16             studentDao.insert(student);
17         } else {
18             System.out.println("The student has already registered.");
19         }
20     }
21
```

```
22     public boolean verify(String sNum) {
23         Student student = studentDao.select(sNum);
24         return student == null ? true : false;
25     }
26
27 }
```

2. 학사 관리 시스템 만들기

■ StudentSelectService 클래스

- 학생 한 명의 정보를 얻는 기능
- 다른 Service 클래스와 마찬가지로 생성자에서 전달받은 DAO 객체를 studentDao 필드에 저장함

코드 4-5

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\service\StudentSelectService.java

```
01 package ch04_pjt_01.ems.member.service;
02
03 import ch04_pjt_01.ems.member.Student;
04 import ch04_pjt_01.ems.member.dao.StudentDao;
05
06 public class StudentSelectService {
07
08     private StudentDao studentDao;
09
10     public StudentSelectService(StudentDao studentDao) {
11         this.studentDao = studentDao;
12     }
13
14     public Student select(String sNum) {
15         if (verify(sNum)) {
16             return studentDao.select(sNum);
17         }
18     }
19 }
```

2. 학사 관리 시스템 만들기

```
18         } else {
19             System.out.println("Student information is available.");
20         }
21
22         return null;
23     }
24
25     public boolean verify(String sNum) {
26         Student student = studentDao.select(sNum);
27         return student != null ? true : false;
28     }
29
30 }
```


2. 학사 관리 시스템 만들기

■ StudentAllSelectService 클래스

- 전체 학생 정보를 반환하는 allSelect()가 있음. 다른 Service 클래스와 마찬가지로 생성자를 통해서 studentDao를 초기화함

코드 4-6

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\service\StudentAllSelectService.java

```
01 package ch04_pjt_01.ems.member.service;
02
03 import java.util.Map;
04
05 import ch04_pjt_01.ems.member.Student;
06 import ch04_pjt_01.ems.member.dao.StudentDao;
07
08 public class StudentAllSelectService {
09
10     private StudentDao studentDao;
11
12     public StudentAllSelectService(StudentDao studentDao) {
13         this.studentDao = studentDao;
14     }
15
16     public Map<String, Student> allSelect() {
17         return studentDao.getStudentDB();
18     }
19
20 }
```

2. 학사 관리 시스템 만들기

■ StudentModifyService 클래스

- 생성자에서 studentDao를 초기화하고, modify()를 이용해서 학생 정보를 수정함

코드 4-7

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\service\StudentModifyService.java

```
01 package ch04_pjt_01.ems.member.service;
02
03 import ch04_pjt_01.ems.member.Student;
04 import ch04_pjt_01.ems.member.dao.StudentDao;
05
06 public class StudentModifyService {
07
08     private StudentDao studentDao;
09
10     public StudentModifyService(StudentDao studentDao) {
11         this.studentDao = studentDao;
12     }
13
14     public void modify(Student student) {
15         if (verify(student.getsNum())) {
16             studentDao.update(student);
17         } else {
18             System.out.println("Student information is available.");
19         }
20     }
21
22     public boolean verify(String sNum) {
23         Student student = studentDao.select(sNum);
24         return student != null ? true : false;
25     }
26
27 }
```

2. 학사 관리 시스템 만들기

■ StudentDeleteService 클래스

- 생성자에서 studentDao를 초기화하고 delete()를 이용해서 학생 정보를 삭제함

코드 4-8

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\service\StudentDeleteService.java

```
01 package ch04_pjt_01.ems.member.service;
02
03 import ch04_pjt_01.ems.member.Student;
04 import ch04_pjt_01.ems.member.dao.StudentDao;
05
06 public class StudentDeleteService {
07
08     private StudentDao studentDao;
09
10     public StudentDeleteService(StudentDao studentDao) {
11         this.studentDao = studentDao;
12     }
13
14     public void delete(String sNum) {
15         if (verify(sNum)) {
16             studentDao.delete(sNum);
17         } else {
18             System.out.println("Student information is available.");
19         }
20     }
21
22     public boolean verify(String sNum) {
23         Student student = studentDao.select(sNum);
24         return student != null ? true : false;
25     }
26
27 }
```

2. 학사 관리 시스템 만들기

■ PrintStudentInformationService 클래스

- 생성자에서 allSelectService를 초기화하고, printStudentsInfo()를 이용해서 전체 학생 정보를 출력

코드 4-9

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\service\PrintStudentInformationService.java

```
01 package ch04_pjt_01.ems.member.service;
02
03 import java.util.Iterator;
04 import java.util.Map;
05 import java.util.Set;
06
07 import ch04_pjt_01.ems.member.Student;
08
09 public class PrintStudentInformationService {
10
11     StudentAllSelectService allSelectService;
12
13     public PrintStudentInformationService(StudentAllSelectService
14     allSelectService) {
15         this.allSelectService = allSelectService;
16     }
17 }
```

2. 학사 관리 시스템 만들기

```
17     public void printStudentsInfo() {
18         Map<String, Student> allStudent = allSelectService.allSelect();
19         Set<String> keys = allStudent.keySet();
20         Iterator<String> iterator = keys.iterator();
21         System.out.println("STUDENT LIST START -----");
22
23         while (iterator.hasNext()) {
24             String key = iterator.next();
25             Student student = allStudent.get(key);
26             System.out.print("sNum:" + student.getNum() + "\t");
27             System.out.print("|sId:" + student.getId() + "\t");
28             System.out.print("|sPw:" + student.getPw() + "\t");
29             System.out.print("|sName:" + student.getName() + "\t");
30             System.out.print("|sAge:" + student.getAge() + "\t");
31             System.out.print("|sGender:" + student.getGender() + "\t");
32             System.out.println("|sMajor:" + student.getMajor() + "\t");
33
34         }
35         System.out.println("END -----");
36     }
37
38 }
```

2. 학사 관리 시스템 만들기

■ EMSInformationService 클래스

- EMS의 개발 연도, 개발자, 버전 등의 정보를 담고 있는 클래스
- 각각의 정보에 해당하는 필드와 getter와 setter 메서드가 있음

코드 4-10

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\member\service\EMSInformationService.java

```
01 package ch04_pjt_01.ems.member.service;
02
03 import java.util.Iterator;
04 import java.util.List;
05 import java.util.Map;
06 import java.util.Set;
07
08 import ch04_pjt_01.ems.member.DBConnectionInfo;
09
10 public class EMSInformationService {
11     private String info;
12     private String copyRight;
13     private String ver;
14     private int sYear;
15     private int sMonth;
16     private int sDay;
17     private int eYear;
18     private int eMonth;
19     private int eDay;
20     private List<String> developers;
21     private Map<String, String> administrators;
22     private Map<String, DBConnectionInfo> dbInfos;
23 }
```

2. 학사 관리 시스템 만들기

```
24     public void printEMSInformation() {
25         System.out.println("EMS INFORMATION START -----");
26         String devPeriod = sYear + "/" + sMonth + "/" + sDay + " ~ " + eYear
+ "/" + eMonth + "/" + eDay;
27         System.out.println(info + "(" + devPeriod + ")");
28         System.out.println(copyRight);
29         System.out.println(ver);
30         System.out.println("Developers: " + developers);
31         System.out.println("Administrator: " + administrators);
32         printDBInfo();
33         System.out.println("END -----");
34     }
35
36     private void printDBInfo() {
37         Set<String> keys = dbInfos.keySet();
38         Iterator<String> iterator = keys.iterator();
39
40         while (iterator.hasNext()) {
41             String key = iterator.next();
42             DBConnectionInfo info = dbInfos.get(key);
43             System.out.print "[" + key + " DB] ";
44             System.out.print("url: " + info.getUrl() + "\t");
45             System.out.print("userId: " + info.getUserId() + "\t");
46             System.out.print("userPw: " + info.getUserPw() + "\n");
47         }
48     }
49
50     public String getInfo() { return info; }
51     public void setInfo(String info) { this.info = info; }
52
```

2. 학사 관리 시스템 만들기

```
53     public String getCopyright() { return copyright; }
54     public void setCopyright(String copyright) { this.copyright = copyright; }
55
56     public String getVer() { return ver; }
57     public void setVer(String ver) { this.ver = ver; }
58
59     public int getsYear() { return sYear; }
60     public void setsYear(int sYear) { this.sYear = sYear; }
61
62     public int getsMonth() { return sMonth; }
63     public void setsMonth(int sMonth) { this.sMonth = sMonth; }
64
65     public int getsDay() { return sDay; }
66     public void setsDay(int sDay) { this.sDay = sDay; }
67
68     public int geteYear() { return eYear; }
69     public void seteYear(int eYear) { this.eYear = eYear; }
70
71     public int geteMonth() { return eMonth; }
72     public void seteMonth(int eMonth) { this.eMonth = eMonth; }
73
74     public int geteDay() { return eDay; }
75     public void seteDay(int eDay) { this.eDay = eDay; }
76
```


2. 학사 관리 시스템 만들기

```
77     public List<String> getDevelopers() { return developers; }
78     public void setDevelopers(List<String> developers) {
79         this.developers = developers;
80     }
81
82     public Map<String, String> getAdministrators() {
83         return administrators;
84     }
85
86     public void setAdministrators(Map<String, String> administrators) {
87         this.administrators = administrators;
88     }
89
90     public Map<String, DBConnectionInfo> getDbInfos() {
91         return dbInfos;
92     }
93
94     public void setDbInfos(Map<String, DBConnectionInfo> dbInfos) {
95         this.dbInfos = dbInfos;
96     }
97 }
```

2. 학사 관리 시스템 만들기

■ InitSampleData 클래스

- 프로그램 실행에 필요한 샘플 데이터로 5명의 학생 정보를 각각의 필드에 가짐
- getter, setter 메서드가 있음

코드 4-11

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\utils\InitSampleData.java

```
01 package ch04_pjt_01.ems.utils;
02
03 public class InitSampleData {
04
05     private String[] sNums = {"hbs001", "hbs002", "hbs003", "hbs004", "hbs005"};
06     private String[] sIds = { "rabbit", "hippo", "raccoon", "elephant", "lion" };
07     private String[] sPws = { "96539", "64875", "15284", "48765", "28661" };
08     private String[] sNames = { "agatha", "barbara", "chris", "doris", "elva" };
09     private int[] sAges = { 19, 22, 20, 27, 19 };
10     private char[] sGenders = { 'M', 'W', 'W', 'M', 'M' };
11     private String[] sMajors = { "English Literature", "Korean Language and
    Literature", "French Language and Literature", "Philosophy", "History", };
12
13     // getter, setter 메소드
14     public String[] getNums() { return sNums; }
15     public void setNums(String[] sNums) { this.sNums = sNums; }
16
17     public String[] getIds() { return sIds; }
18     public void setIds(String[] sIds) { this.sIds = sIds; }
```

5명의 학생 정보 샘플 데이터

2. 학사 관리 시스템 만들기

```
19
20     public String[] getPws() { return sPws; }
21     public void setsPws(String[] sPws) { this.sPws = sPws; }
22
23     public String[] getsNames() { return sNames; }
24     public void setsNames(String[] sNames) { this.sNames = sNames; }
25
26     public int[] getsAges() { return sAges; }
27     public void setsAges(int[] sAges) { this.sAges = sAges; }
28
29     public char[] getsGenders() { return sGenders; }
30     public void setsGenders(char[] sGenders) { this.sGenders = sGenders; }
31
32     public String[] getsMajors() { return sMajors; }
33     public void setsMajors(String[] sMajors) { this.sMajors = sMajors; }
34
35 }
```

Section 02

스프링 설정 파일

1. InitSampleData 빈

■ applicationContext.xml 생성

- [src/main/resources]에 applicationContext.xml 파일 생성



그림 4-4 생성된 applicationContext.xml 파일

■ 빈(Bean)의 생성과 초기화

- InitSampleData 빈을 생성하는 코드
 - ✓ InitSampleData는 학생 5명의 샘플 정보로 <bean>을 이용해서 빈을 생성할 수 있음

```
<bean id="initSampleData" class="ch04_pjt_01.ems.utils.InitSampleData" />
```

■ 빈(Bean)의 생성과 초기화

- <bean>은 id와 class 속성을 가짐
 - ✓ id: 빈을 가리키는 레퍼런스 변수 이름을 명시함
 - ✓ class: 클래스 이름을 명시함

기존의

객체 생성 방법 : InitSampleData initSampleData = new InitSampleData();

IoC 컨테이너에서

객체 생성 방법 : <bean id="initSampleData" class="ch04_pjt_01.ems.utils.InitSampleData"/>

패키지를 포함한 클래스 이름

그림 4-5 레퍼런스 변수와 클래스 이름을 명시하는 id와 class

1. InitSampleData 빈

■ 빈(Bean)의 생성과 초기화

- <bean>에는 클래스의 멤버 필드를 나타내는 하위 태그로 <property>가 있음
 - ✓ 이를 이용해서 필드를 초기화할 수 있음
 - ✓ <property>의 name 속성에는 필드 이름을 명시함
 - ✓ 데이터 타입이 배열인 경우 <array>와 <value>를 이용해서 값을 명시함

기존의

필드 초기화 방법 : `private String[] sNums = {"hbs001", "hbs002", "hbs003", "hbs004", "hbs005"}`

IoC 컨테이너에서

필드 초기화 방법 : `<bean id="initSampleData" class="ch04_pjt_01.ems.utils.InitSampleData"/>`

```
<property name="sNums">
  <array>
    <value>hbs001</value>
    <value>hbs002</value>
    <value>hbs003</value>
    <value>hbs004</value>
    <value>hbs005</value>
  </array>
</property>
</bean>
```

그림 4-6 <property>를 이용한 필드 초기화

■ 빈(Bean)의 생성과 초기화

- applicationContext.xml에서 sNums를 초기화했기 때문에 다시 InitSampleData에서 sNums를 초기화하지 않아도 됨
- 따라서 InitSampleData.java 코드([코드 4-11] 5행)를 다음과 같이 수정
- before

```
private String[] sNums = {"hbs001", "hbs002", "hbs003", "hbs004", "hbs005"};
```

- after

```
private String[] sNums;
```


1. InitSampleData 빈

■ 빈(Bean)의 생성과 초기화

- <bean>은 생성자를 호출하고, <property>의 <array> 값은 InitSampleData의 setsNums(String[] sNums) 메서드에 전달됨

```
<bean id="initSampleData"
      class="ch04_pjt_01.ems.utils.InitSampleData"/>
```

```
public InitSampleData() {
}
```

그림 4-7 <bean>은 생성자를 호출해서 빈을 생성함

```
<property name="sNums">
```

```
<array>
```

```
<value>hbs001</value>
```

```
<value>hbs002</value>
```

```
<value>hbs003</value>
```

```
<value>hbs004</value>
```

```
<value>hbs005</value>
```

```
</array>
```

```
</property>
```

```
public void setsNums(String[] sNums) {
    this.sNums = sNums;
}
```

그림 4-8 <property>가 필드를 초기화하기 위해서는 setter 메서드를 이용

1. InitSampleData 빈

■ 빈(Bean)의 생성과 초기화

- 자바에서는 개발자가 어떠한 생성자도 명시하지 않으면 컴파일러가 컴파일 단계에서 자동으로 디폴트 생성자(default constructor)를 생성

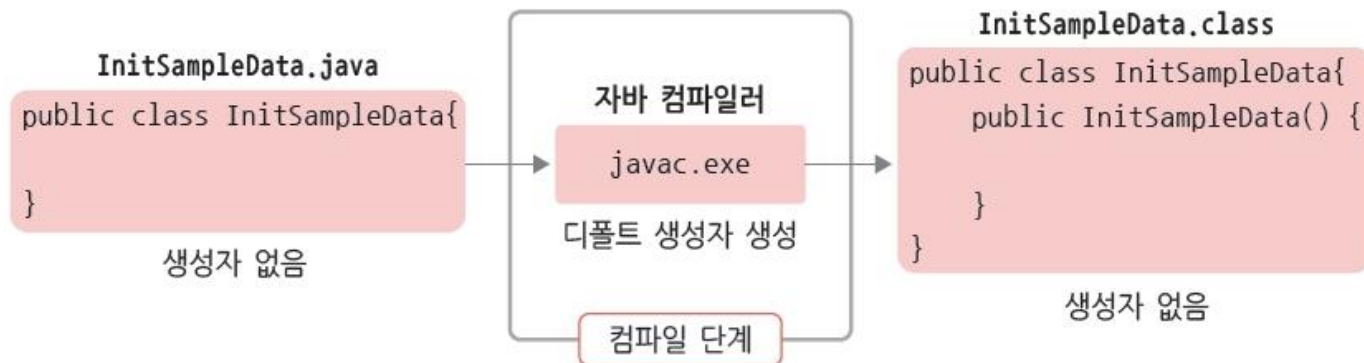


그림 4-9 생성자가 하나도 없는 경우 컴파일러가 디폴트 생성자를 생성

1. InitSampleData 빈

■ 빈(Bean)의 생성과 초기화

- 초기화 관련 에러 발생1) InitSampleData에 생성자가 있는 경우
 - ✓ InitSampleData에 생성자가 있다면 컴파일러는 컴파일 단계에서 디폴트 생성자를 만들지 않음
 - ✓ <bean>은 디폴트 생성자를 호출할 수 없게 되고 결과적으로 에러 발생

```
public InitSampleData(String[] sNums) {  
    this.sNums = sNums;  
}
```

실행 결과

Caused by: org.springframework.beans.BeanInstantiationException: Failed to instantiate [ch04_pjt_01.ems.utils.InitSampleData]: No default constructor found;

1. InitSampleData 빈

■ 빈(Bean)의 생성과 초기화

- 초기화 관련 예러 발생2) 초기화하려는 필드에 setter 메서드가 없는 경우
 - ✓ IoC 컨테이너에는 필드를 초기화할 수 없어 예러가 발생함
 - ✓ 예시) setter 메서드 주석 처리

```
//    public void setsNums(String[] sNums) {  
//        this.sNums = sNums;  
//    }
```

실행 결과

```
Caused by: org.springframework.beans.NotWritablePropertyException: Invalid property  
'sNums' of bean class [ch04_pjt_01.ems.utils.InitSampleData]: Bean property 'sNums'  
is not writable or has an invalid setter method. Did you mean 'sNames'?
```

1. InitSampleData 빈

■ 나머지 필드 초기화

- applicationContext.xml

```
<!-- sIds 필드 초기화 -->
<property name="sIds">
  <array>
    <value>rabbit</value>
    <value>hippo</value>
    <value>raccoon</value>
    <value>elephant</value>
    <value>lion</value>
  </array>
</property>
```

```
<!-- sPws 필드 초기화 -->
<property name="sPws">
  <array>
    <value>p0001</value>
    <value>p0002</value>
    <value>p0003</value>
    <value>p0004</value>
    <value>p0005</value>
  </array>
</property>
```

```
<!-- sNames 필드 초기화 -->
<property name="sNames">
  <array>
    <value>agatha</value>
    <value>barbara</value>
    <value>chris</value>
    <value>doris</value>
    <value>elva</value>
  </array>
</property>
```

```
<!-- sAges 필드 초기화 -->
<property name="sAges">
  <array>
    <value>19</value>
    <value>22</value>
    <value>20</value>
    <value>27</value>
    <value>19</value>
  </array>
</property>
```

```
<!-- sGenders 필드 초기화 -->
<property name="sGenders">
  <array>
    <value>M</value>
    <value>W</value>
    <value>W</value>
    <value>M</value>
    <value>M</value>
  </array>
</property>

<!-- sMajors 필드 초기화 -->
<property name="sMajors">
  <array>
    <value>English</value>
    <value>Korean</value>
    <value>French</value>
    <value>Philosophy</value>
    <value>History</value>
  </array>
</property>
```

1. InitSampleData 빈

■ 나머지 필드 초기화

- InitSampleData.java

```
private String[] sIds;  
private String[] sPws;  
private String[] sNames;  
private int[] sAges;  
private char[] sGenders;  
private String[] sMajors;
```

2. StudentDao 빈

■ StudentDao 빈 생성

- <bean>을 이용해서 빈을 생성함
- 이때 id와 class id 이름과 패키지를 포함한 클래스 이름을 명시함

```
<bean id="studentDao" class="ch04_pjt_01.ems.member.dao.StudentDao" />
```

■ Service 관련 빈 생성

- 모든 Service가 동일한 구조임
 - ✓ StudentRegister Service만 이해하면 모두 이해할 수 있음
- <bean>을 이용해서 StudentRegisterService 빈을 생성하는 코드
 - ✓ 앞에서 살펴본 다른 빈과 동일함

```
<bean id="studentRegisterService"  
      class="ch04_pjt_01.ems.member.service.StudentRegisterService" />
```

2. StudentDao 빈

■ Service 관련 빈 생성

- 객체가 생성될 때 생성자에서 studentDao 필드를 초기화함
 - ✓ 이를 위해서 StudentDao를 외부에서 주입받고 있음
 - ✓ 즉 디폴트 생성자가 아닌 StudentDao를 주입받는 생성자를 이용해서 객체 생성
- IoC 컨테이너에서 StudentRegisterService 빈을 생성할 때도 외부에서 StudentDao 객체를 주입해야 함
 - ✓ 이를 위해서 <constructor-arg> 태그를 이용함



그림 4-10 StudentRegisterService는 객체 생성 시 StudentDao를 외부에서 전달받음

2. StudentDao 빈

■ Service 관련 빈 생성

- Student RegisterService 빈 생성 코드에 <constructor-arg>를 추가한 코드

```
<bean id="studentDao" class="ch04_pjt_01.ems.member.dao.StudentDao" />  
  
<bean id="studentRegisterService"  
      class="ch04_pjt_01.ems.member.service.StudentRegisterService">  
  <constructor-arg ref="studentDao" />  
</bean>
```

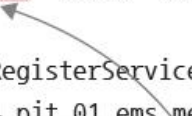


그림 4-11 StudentDao 빈을 가리키는 <constructor-arg>의 ref

- 디폴트 생성자가 없어서 발생하는 에러
 - ✓ StudentRegisterService 빈을 생성할 때 <constructor-arg>가 없는 경우
 - ✓ <constructor-arg> 주석 처리

```
<bean id="studentRegisterService"  
      class="ch04_pjt_01.ems.member.service.StudentRegisterService">  
  <!-- <constructor-arg ref="studentDao" /> -->  
</bean>
```

실행 결과

Caused by: org.springframework.beans.BeanInstantiationException: Failed to instantiate [ch04_pjt_01.ems.member.service.StudentRegisterService]: No default constructor found;

2. StudentDao 빈

■ 나머지 빈 생성하기

- 다른 Service 객체들도 IoC 컨테이너에서 빈을 생성함

```
<!-- StudentModifyService 빈 생성 -->
```

```
<bean id="studentModifyService"  
      class="ch04_pjt_01.ems.member.service.StudentModifyService">  
    <constructor-arg ref="studentDao" />  
</bean>
```

```
<!-- StudentDeleteService 빈 생성 -->
```

```
<bean id="studentDeleteService"  
      class="ch04_pjt_01.ems.member.service.StudentDeleteService">  
    <constructor-arg ref="studentDao" />  
</bean>
```

```
<!-- StudentSelectService 빈 생성 -->
```

```
<bean id="studentSelectService"  
      class="ch04_pjt_01.ems.member.service.StudentSelectService">  
    <constructor-arg ref="studentDao" />  
</bean>
```

```
<!-- StudentAllSelectService 빈 생성 -->
```

```
<bean id="studentAllSelectService"  
      class="ch04_pjt_01.ems.member.service.StudentAllSelectService">  
    <constructor-arg ref="studentDao" />  
</bean>
```

```
<!-- PrintStudentInformationService 빈 생성 -->
```

```
<bean id="printStudentInformationService "  
      class="ch04_pjt_01.ems.member.service.PrintStudentInformationService">  
    <constructor-arg ref="studentAllSelectService" />  
</bean>
```

3. DBConnectionInfo 빈

■ DBConnectionInfo 빈 생성

- 데이터베이스 연결에 필요한 url, userId, userPw 필드만 있음
 - ✓ <property>를 이용한 필드 초기화 코드만 작성하면 됨
- 주의) 일반적으로 프로젝트를 진행할 때 개발용 데이터베이스와 실제 서비스에 이용하는 데이터베이스를 구분함



그림 4-12 개발용 데이터베이스와 실제 서비스에 이용하는 데이터베이스

3. DBConnectionInfo 빈

■ DBConnectionInfo 빈 생성

- IoC 컨테이너에 DBConnectionInfo 빈을 생성할 때도 개발에 이용하는 데이터베이스 빈과 실제 서비스에 이용하는 데이터베이스 빈으로 구분

<!-- 개발용 데이터베이스 빈 생성 -->

```
<bean id="dev_DBConnectionInfoDev"
      class="ch04_pjt_01.ems.member.DBConnectionInfo">
  <property name="url" value="000.000.000.000" />
  <property name="userId" value="admin" />
  <property name="userPw" value="0000" />
</bean>
```

<!-- 실제 서비스에 이용하는 데이터베이스 빈 생성 -->

```
<bean id="real_DBConnectionInfo"
      class="ch04_pjt_01.ems.member.DBConnectionInfo">
  <property name="url" value="111.111.111.111" />
  <property name="userId" value="master" />
  <property name="userPw" value="1111" />
</bean>
```

4. EMSInformationService 빈

■ EMSInformationService 빈 생성

- 필드가 12개나 되고, 데이터 타입도 String, int, List, Map으로 다양함
- 이러한 데이터 타입을 어떻게 초기화하는지 살펴봄
 - ✓ 먼저 빈을 생성하고 데이터 타입이 String, int인 필드는 다음과 같이 코딩하기

```
<!-- EMSInformationService 빈 생성 -->
<bean id="eMSInformationService"
      class="ch04_pjt_01.ems.member.service.EMSInformationService">

    <!-- info 필드 초기화 -->
    <property name="info">
        <value>Education Management System program was developed in 2022.</value>
    </property>

    <!-- copyRight 필드 초기화 -->
    <property name="copyRight">
        <value>COPYRIGHT(C) 2022 EMS CO., LTD. ALL RIGHT RESERVED. CONTACT MASTER FOR
        MORE INFORMATION.</value>
    </property>
```

4. EMSInformationService 빈

```
<!-- ver 필드 초기화 -->  
<property name="ver">  
    <value>The version is 1.0</value>  
</property>
```

```
<!-- sYear 필드 초기화 -->  
<property name="sYear">  
    <value>2022</value>  
</property>
```

```
<!-- sMonth 필드 초기화 -->  
<property name="sMonth">  
    <value>3</value>  
</property>
```

```
<!-- sDay 필드 초기화 -->  
<property name="sDay">  
    <value>1</value>  
</property>
```

```
<!-- eYear 필드 초기화 -->  
<property name="eYear">  
    <value>2022</value>  
</property>
```

```
<!-- eMonth 필드 초기화 -->  
<property name="eMonth">  
    <value>4</value>  
</property>
```

```
<!-- eDay 필드 초기화 -->  
<property name="eDay">  
    <value>30</value>  
</property>
```

4. EMSInformationService 빈

■ EMSInformationService 빈 생성

- 코드를 좀 더 간결하게 변경하기

- ✓ 데이터가 한 개뿐이라면 <value> 태그를 사용하지 않고 <property> 태그의 value 속성을 이용할 수 있음

- before

```
<!-- info 필드 초기화 -->
<property name="info">
    <value>Education Management System program was developed in 2022.</value>
</property>
```

- after

```
<!-- info 필드 초기화 -->
<property name="info" value="Education Management System program was developed in 2022." />
```

- ✓ info 외의 다른 필드들도 value 속성을 이용하면 전체적인 코드가 간결해짐

4. EMSInformationService 빈

■ EMSInformationService 빈 생성

- 데이터 타입이 기초 데이터 타입이 아닌 참조 데이터 타입인 필드를 초기화하는 방법
- developers 초기화: List 데이터 타입

```
<property name="developers">
  <list>
    <value>Cheney.</value>
    <value>Eloy.</value>
    <value>Jasper.</value>
    <value>Dillon.</value>
    <value>Kian.</value>
  </list>
</property>
```


4. EMSInformationService 빈

■ EMSInformationService 빈 생성

- administrators 초기화: Map 데이터 타입

```
<!-- administrators 필드 초기화 -->
<property name="administrators">
  <map>
    <entry>
      <key>
        <value>Cheney</value>
      </key>
      <value>cheney@springPjt.org</value>
    </entry>
    <entry>
      <key>
        <value>Jasper</value>
      </key>
      <value>jasper@springPjt.org</value>
    </entry>
  </map>
</property>
```

4. EMSInformationService 빈

■ EMSInformationService 빈 생성

- 자바 파일에서 administrators 초기화할 경우

```
private Map<String, String> administrators = new HashMap<String, String>(){  
    put("Cheney", "cheney@springPjt.org");  
    put("Jasper", "jasper@springPjt.org");  
};
```

4. EMSInformationService 빈

■ EMSInformationService 빈 생성

- dbinfos 초기화

- ✓ DBConnectionInfo는 기초 데이터 타입이 아니기 때문에 <ref> 이용함

```
<!-- dbInfos 필드 초기화 -->
<property name="dbInfos">
  <map>
    <entry>
      <key>
        <value>dev</value>
      </key>
      <ref bean="dev_DBConnectionInfoDev" /> — 개발용 데이터베이스 지정
    </entry>
    <entry>
      <key>
        <value>real</value>
      </key>
      <ref bean="real_DBConnectionInfo" /> — 실제 서비스 데이터베이스 지정
    </entry>
  </map>
</property>
```

Section 03

메이븐 설정 파일

1. 메이븐 설정 파일

■ pom.xml에 spring-context 모듈 설정

코드 4-12

ch04_pjt_01\pom.xml

```
01 <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/xsd/maven-4.0.0.xsd">
02     <modelVersion>4.0.0</modelVersion>
03     <groupId>spring5</groupId>
04     <artifactId>ch03_pjt_02</artifactId>
05     <version>0.0.1-SNAPSHOT</version>
06
07     <!-- spring-context 모듈 -->
08     <dependencies>
09         <dependency>
10             <groupId>org.springframework</groupId>
11             <artifactId>spring-context</artifactId>
12             <version>5.2.9.RELEASE</version>
13         </dependency>
14     </dependencies>
15
16     <!-- 빌드 설정 -->
17     <build>
18         <plugins>
19             <plugin>
20                 <artifactId>maven-compiler-plugin</artifactId>
21                 <version>3.1</version>
```

1. 메이븐 설정 파일

```
22         <configuration>
23             <source>11</source>
24             <target>11</target>
25             <encoding>utf-8</encoding>
26         </configuration>
27     </plugin>
28 </plugins>
29 </build>
30
31 </project>
```

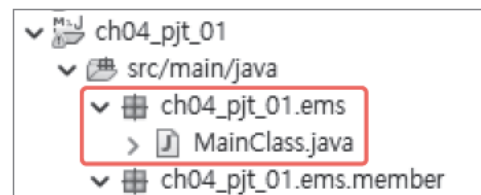
Section 04

MainClass.java

1. MainClass.java 코딩

■ main() 메서드 코딩

- GenericXmlApplicationContext와 IoC 컨테이너에 생성된 빈을 이용함
- 1. ch04_pjt_01 프로젝트의 [src/main/java]에 ch04_pjt_01.ems 패키지를 생성하고, 내부에 MainClass를 생성하기



- 2. 생성된 MainClass의 main()에 applicationContext.xml과 GenericXmlApplicationContext를 이용해서 IoC 컨테이너 생성

코드 4-13

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\MainClass.java

```
01 ...import문 생략...
02 public class MainClass {
03
04     public static void main(String[] args) {
05         // IoC 컨테이너 생성
06         GenericXmlApplicationContext ctx =
07             new GenericXmlApplicationContext("classpath:applicationContext.xml");
08         ...생략...
09     }
10 }
```


1. MainClass.java 코딩

■ main() 메서드 코딩

- 3. [코드 4-13]의 8행에 InitSampleData에 있는 샘플 데이터를 이용해서 데이터베이스에 학생 등록

코드 4-14

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\MainClass.java

```
01 // 샘플 데이터
02 InitSampleData initSampleData = ctx.getBean("initSampleData",
    InitSampleData.class);
03 String[] sNums = initSampleData.getsNums();
04 String[] sIds = initSampleData.getsIds();
05 String[] sPws = initSampleData.getsPws();
06 String[] sNames = initSampleData.getsNames();
07 int[] sAges = initSampleData.getsAges();
08 char[] sGenders = initSampleData.getsGenders();
09 String[] sMajors = initSampleData.getsMajors();
10
11 // 데이터베이스에 샘플 데이터 등록
12 StudentRegisterService registerService =
    ctx.getBean("studentRegisterService", StudentRegisterService.class);
13 for (int i = 0; i < sNums.length; i++) {
14     registerService.register(new Student(sNums[i], sIds[i], sPws[i],
        sNames[i], sAges[i], sGenders[i], sMajors[i]));
15
16 }
```

1. MainClass.java 코딩

■ main() 메서드 코딩

- 4. PrintStudentInformationService를 이용해 모든 학생 정보 출력
 - ✓ 3번의 registerService를 이용해 학번이 hbs006인 새로운 학생 정보 등록

코드 4-15

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\MainClass.java

```
01 // 학생 리스트
02 PrintStudentInformationService printStudentInformatinService =
    ctx.getBean("printStudentInformationService",
        PrintStudentInformationService.class);
03 printStudentInformatinService.printStudentsInfo(); // 학생 리스트
04
05 // 학생 등록
06 registerService = ctx.getBean("studentRegisterService",
    StudentRegisterService.class);
07 registerService.register(new Student("hbs006", "deer", "p00006", "melissa",
    26, 'w', "Music"));
08
09 printStudentInformatinService.printStudentsInfo(); // 학생 리스트
```

1. MainClass.java 코딩

■ main() 메서드 코딩

- 5. StudentSelectService를 이용해서 학번이 hbs006인 학생 정보 조회

코드 4-16

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\MainClass.java

```
01 // 학생 출력
02 StudentSelectService selectService =
    ctx.getBean("studentSelectService", StudentSelectService.class);
03 Student selectedstudent = selectService.select("hbs006");
04
05 System.out.println("STUDENT START -----");
06 System.out.print("sNum:" + selectedstudent.getNum() + "\t");
07 System.out.print("|sId:" + selectedstudent.getId() + "\t");
08 System.out.print("|sPw:" + selectedstudent.getPw() + "\t");
09 System.out.print("|sName:" + selectedstudent.getName() + "\t");
10 System.out.print("|sAge:" + selectedstudent.getAge() + "\t");
11 System.out.print("|sGender:" + selectedstudent.getGender() + "\t");
12 System.out.println("|sMajor:" + selectedstudent.getMajor());
13 System.out.println("END -----");
```

1. MainClass.java 코딩

■ main() 메서드 코딩

- 6. StudentModifyService를 이용해 학번이 hbs006인 학생 정보 수정

코드 4-17

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\MainClass.java

```
01 // 학생 수정
02 StudentModifyService modifyService =
    ctx.getBean("studentModifyService", StudentModifyService.class);
03 modifyService.modify(new Student("hbs006", "pig", "p0066", "melissa", 27,
    'w', "Computer"));
04
05 printStudentInformatinService.printStudentsInfo(); // 학생 리스트
```

- 7. StudentDeleteService를 이용해 학번이 hbs005인 학생 정보 삭제

코드 4-18

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\MainClass.java

```
01 // 학생 삭제
02 StudentDeleteService deleteService =
    ctx.getBean("studentDeleteService", StudentDeleteService.class);
03 deleteService.delete("hbs005");
04
05 printStudentInformatinService.printStudentsInfo(); // 학생 리스트
```

1. MainClass.java 코딩

■ main() 메서드 코딩

- 8. EMSInformationService를 이용해 시스템 정보 출력
 - ✓ 그리고 GenericXml ApplicationContext를 이용해 컨텍스트 닫음

코드 4-19

ch04_pjt_01\src\main\java\ch04_pjt_01\ems\MainClass.java

```
01 // 시스템 정보
02 EMSInformationService emsInformationService =
    ctx.getBean("eMSInformationService", EMSInformationService.class);
03 emsInformationService.printEMSInformation();
04
05 ctx.close();
```

1. MainClass.java 코딩

■ main() 메서드 코딩

- 9. Java Application을 이용해 프로그램 실행

✓ 모든 빈이 정상적으로 생성되고 필드 초기화가 올바르게 됐는지 확인하기

```
<terminated> MainClass [3] [Java Application] C:\Program Files\Java\jdk-11.0.14\bin\java.exe (2022.3.20. 오전 7:20:24 - 오전 7:20:25)

STUDENT LIST START
sNum:hbs003      sId:raccoon      sPw:p0003      sName:Chris      sAge:20      sGender:W      sMajor:French
sNum:hbs004      sId:elephant     sPw:p0004      sName:doris      sAge:27      sGender:W      sMajor:Philosophy
sNum:hbs001      sId:rabbit       sPw:p0001      sName:agatha     sAge:19      sGender:W      sMajor:English
sNum:hbs002      sId:hippo        sPw:p0002      sName:barbara    sAge:22      sGender:W      sMajor:Korean
sNum:hbs005      sId:lion         sPw:p0005      sName:elva       sAge:19      sGender:W      sMajor:History
END

STUDENT LIST START
sNum:hbs003      sId:raccoon      sPw:p0003      sName:Chris      sAge:20      sGender:W      sMajor:French
sNum:hbs004      sId:elephant     sPw:p0004      sName:doris      sAge:27      sGender:W      sMajor:Philosophy
sNum:hbs001      sId:rabbit       sPw:p0001      sName:agatha     sAge:19      sGender:W      sMajor:English
sNum:hbs002      sId:hippo        sPw:p0002      sName:barbara    sAge:22      sGender:W      sMajor:Korean
sNum:hbs005      sId:lion         sPw:p0005      sName:elva       sAge:19      sGender:W      sMajor:History
sNum:hbs006      sId:deer         sPw:p0006      sName:melissa    sAge:26      sGender:w      sMajor:Music
END

STUDENT START
sNum:hbs006      sId:deer         sPw:p0006      sName:melissa    sAge:26      sGender:w      sMajor:Music
END

STUDENT LIST START
sNum:hbs003      sId:raccoon      sPw:p0003      sName:Chris      sAge:20      sGender:W      sMajor:French
sNum:hbs004      sId:elephant     sPw:p0004      sName:doris      sAge:27      sGender:W      sMajor:Philosophy
sNum:hbs001      sId:rabbit       sPw:p0001      sName:agatha     sAge:19      sGender:W      sMajor:English
sNum:hbs002      sId:hippo        sPw:p0002      sName:barbara    sAge:22      sGender:W      sMajor:Korean
sNum:hbs005      sId:lion         sPw:p0005      sName:elva       sAge:19      sGender:W      sMajor:History
sNum:hbs006      sId:pig          sPw:p0006      sName:melissa    sAge:27      sGender:w      sMajor:Computer
END

STUDENT LIST START
sNum:hbs003      sId:raccoon      sPw:p0003      sName:Chris      sAge:20      sGender:W      sMajor:French
sNum:hbs004      sId:elephant     sPw:p0004      sName:doris      sAge:27      sGender:W      sMajor:Philosophy
sNum:hbs001      sId:rabbit       sPw:p0001      sName:agatha     sAge:19      sGender:W      sMajor:English
sNum:hbs002      sId:hippo        sPw:p0002      sName:barbara    sAge:22      sGender:W      sMajor:Korean
sNum:hbs006      sId:pig          sPw:p0006      sName:melissa    sAge:27      sGender:w      sMajor:Computer
END

EMS INFORMATION START
Education Management System program was developed in 2022. (2022/3/1 ~ 2022/4/30)
COPYRIGHT(C) 2022 EMS CO., LTD. ALL RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.
The version is 1.0
Developers: [Cheney., Eloy., Jasper., Dillon., Kian.]
Administrator: {Cheney=cheney@springPjt.org, Jasper=jasper@springPjt.org}
[dev DB] url: 000,000,000,000      userId: admin      userPw: 0000
[real DB] url: 111,111,111,111      userId: master      userPw: 1111
END
```

전체 학생

hbs006 학생 등록

hbs006 학생 조회

hbs006 학생 수정

hbs005 학생 삭제

프로그램, 관리자, 데이터베이스 정보 출력

그림 4-13 학사관리 시스템 실행 결과

1. MainClass.java 코딩

■ main() 메서드 코딩

- applicationContext.xml을 이용해서 빈을 생성하고, IoC 컨테이너의 빈이 사용이 사용되는 과정

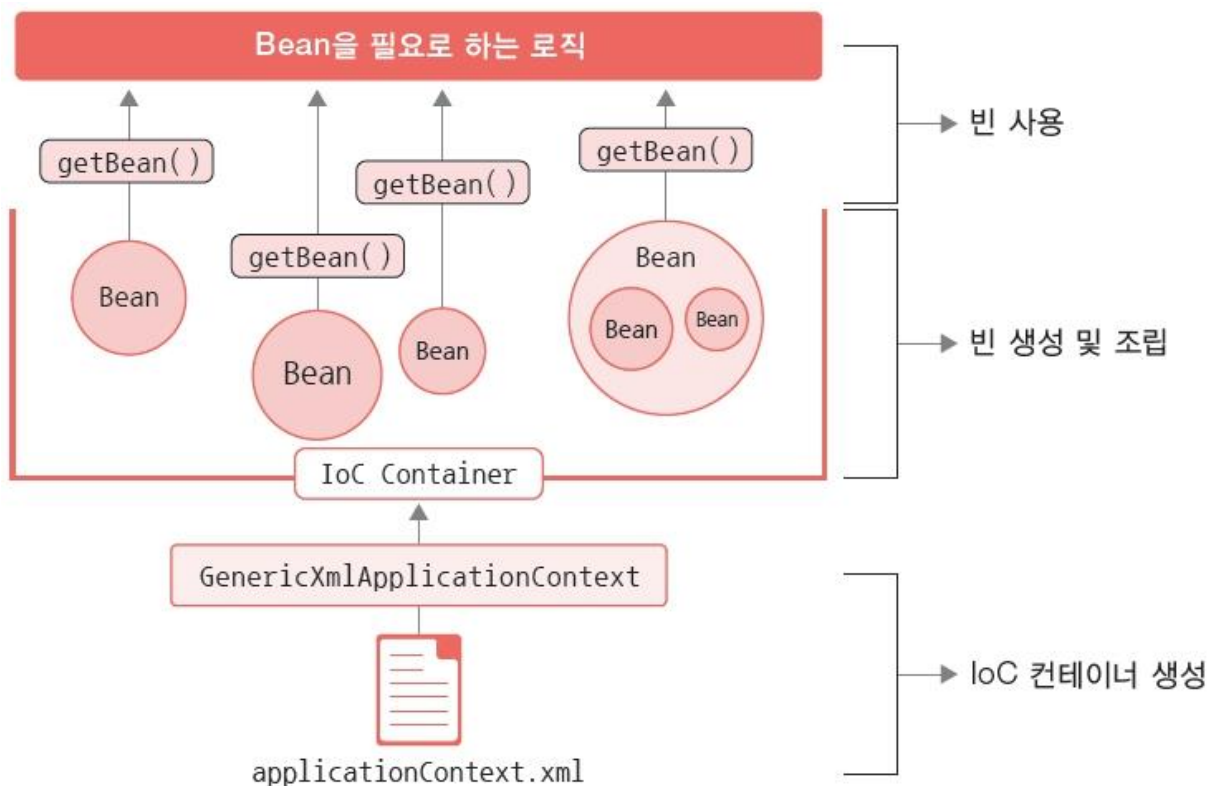


그림 4-14 IoC 컨테이너 생성 및 `getBean()`을 이용한 빈 사용

Section 05

스프링 설정 파일 분리

1. 파일 분리 방법 1: IoC 컨테이너에서 조합하기

- applicationContext.xml 파일을 3개 파일로 분리하기
 - 1. applicationContext.xml 파일을 복사하여 붙여넣음
 - 2. 파일 이름 변경: appCtx1.xml, appCtx2.xml, appCtx3.xml

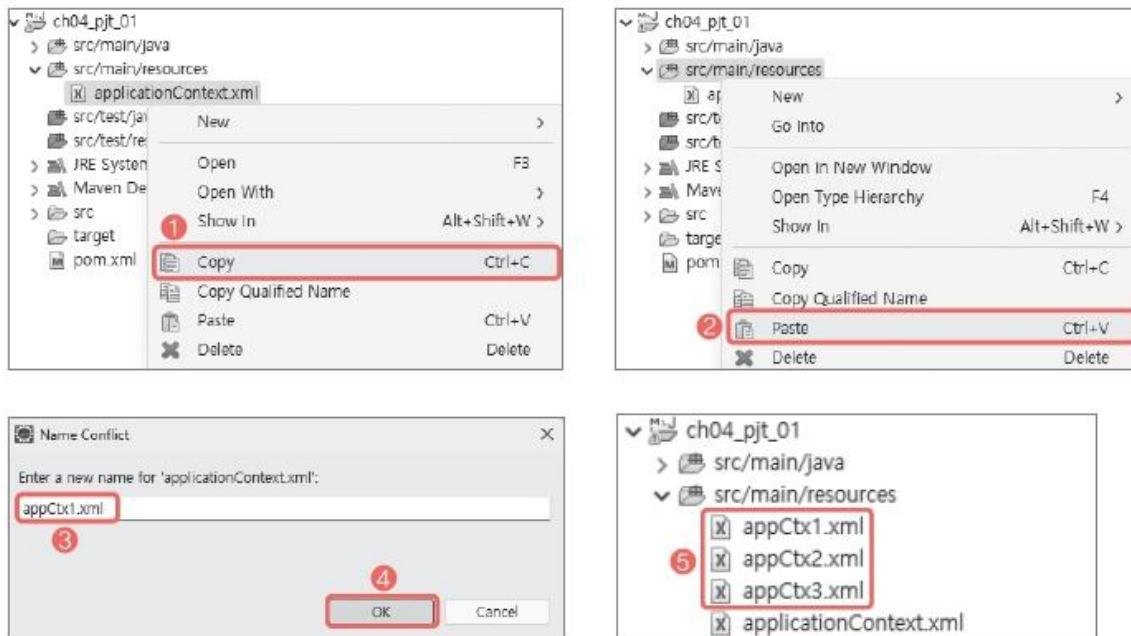


그림 4-15 applicationContext.xml을 복사하여 생성된 appCtx1.xml, appCtx2.xml, appCtx3.xml 파일

1. 파일 분리 방법 1: IoC 컨테이너에서 조합하기

■ appCtx1.xml

- InitSampleData, Dao, Service 빈을 생성하는 코드만 남기고 다른 코드는 삭제
 - ✓ 즉, DBConnectionInfo 빈과 EMSInformationService 빈을 생성하는 코드 삭제

코드 4-20

ch04_pjt_01\src\main\resources\appCtx1.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <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">
04
05     <!-- InitSampleData 빈 -->
06     <bean id="initSampleData"
07         class="ch04_pjt_01.ems.utils.InitSampleData">
08         <property name="sNums">
09             <array>
10                 <value>hbs001</value>
11                 <value>hbs002</value>
12                 <value>hbs003</value>
13                 <value>hbs004</value>
14                 <value>hbs005</value>
15             </array>
16             ...생략...
17     </bean>
18
```

```
19     <!-- StudentDao 빈 -->
20     <bean id="studentDao"
21         class="ch04_pjt_01.ems.member.dao.StudentDao" />
22
23     <bean id="studentRegisterService"
24         class="ch04_pjt_01.ems.member.service.StudentRegisterService">
25         <constructor-arg ref="studentDao" />
26     </bean>
27
28     ...생략...
29 </beans>
```

1. 파일 분리 방법 1: IoC 컨테이너에서 조합하기

■ appCtx2.xml

- DBConnectionInfo 빈을 생성하는 코드만 남겨놓고 다른 코드는 삭제

코드 4-21

ch04_pjt_01\src\main\resources\appCtx2.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <beans xmlns="http://www.springframework.org/schema/beans"
04       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05       xsi:schemaLocation="http://www.springframework.org/schema/beans
06       http://www.springframework.org/schema/beans/spring-beans.xsd">
07
08     <!-- DBConnectionInfo 빈 -->
09     <bean id="dev_DBConnectionInfoDev"
10           class="ch04_pjt_01.ems.member.DBConnectionInfo">
11       <property name="url"
12         value="000.000.000.000" />
13       <property name="userId" value="admin" />
14       <property name="userPw" value="0000" />
15     </bean>
16
17     <bean id="real_DBConnectionInfo"
18           class="ch04_pjt_01.ems.member.DBConnectionInfo">
19       <property name="url"
20         value="111.111.111.111" />
21       <property name="userId" value="master" />
22       <property name="userPw" value="1111" />
23     </bean>
24 </beans>
```

1. 파일 분리 방법 1: IoC 컨테이너에서 조합하기

■ appCtx3.xml

- EMSInformationService 빈을 생성하는 코드만 남겨놓고 다른 코드는 삭제

코드 4-22

ch04_pjt_01\src\main\resources\appCtx3.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <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">
04
05     <!-- EMSInformationService 빈 -->
06     <bean id="eMSInformationService"
07         class="ch04_pjt_01.ems.member.service.EMSInformationService">
08         <property name="info" value="Education Management System program was
    developed in 2022." />
09         <property name="copyRight" value="COPYRIGHT(C) 2022 EMS CO., LTD. ALL
    RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION." />
10         <property name="ver" value="The version is 1.0" />
11         <property name="sYear" value="2022" />
12         <property name="sMonth" value="3" />
13         <property name="sDay" value="1" />
14         ...생략...
15     </bean>
16
17 </beans>
```

1. 파일 분리 방법 1: IoC 컨테이너에서 조합하기

■ 분리된 applicationContext.xml

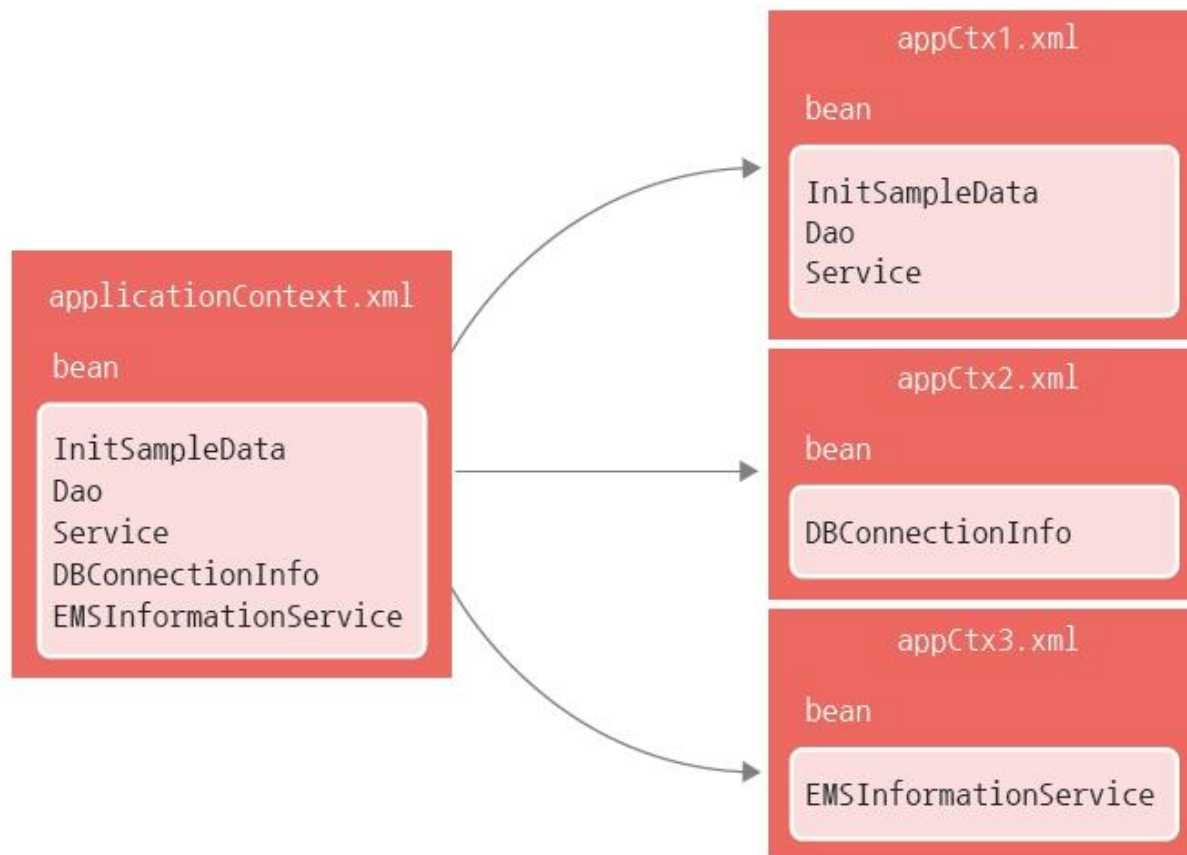


그림 4-16 appCtx1.xml, appCtx2.xml, appCtx3.xml로 분리된 applicationContext.xml

1. 파일 분리 방법 1: IoC 컨테이너에서 조합하기

■ 분리된 applicationContext.xml

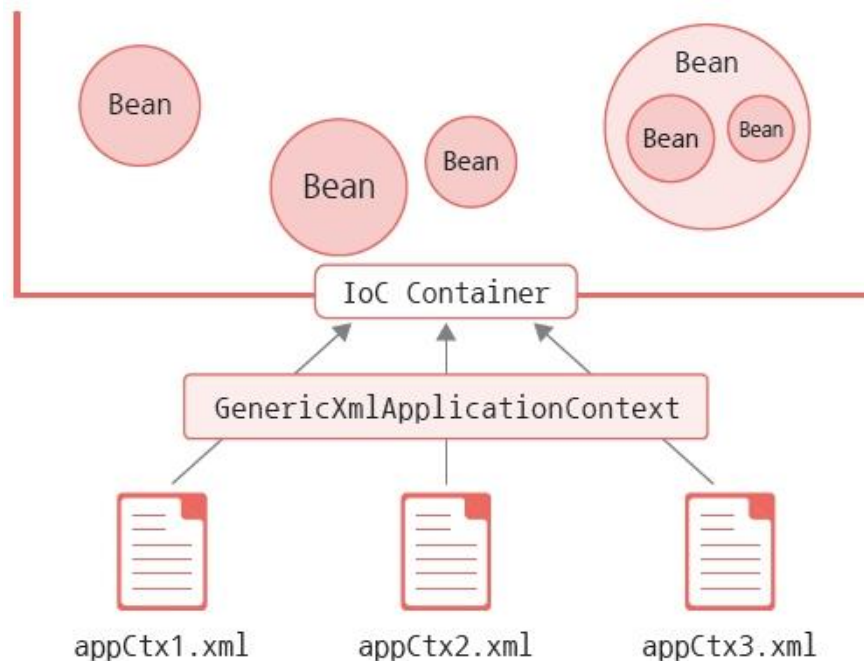


그림 4-17 동일한 스프링 컨테이너이므로 ref 속성으로 다른 스프링 설정 파일의 빈을 참조할 수 있음

1. 파일 분리 방법 1: IoC 컨테이너에서 조합하기

■ MainClass.java의 main() 메서드 수정

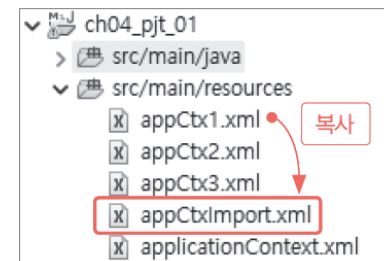
- [코드 4-13]의 6행에서 GenericXmlApplicationContext를 다음과 같이 수정

```
String[] appCtxs = {"classpath:appCtx1.xml", "classpath:appCtx2.xml",  
    "classpath:appCtx3.xml"};  
  
GenericXmlApplicationContext ctx =  
    new GenericXmlApplicationContext(appCtxs);
```

2. 파일 분리 방법 2: <import> 태그 사용하기

■ appCtxImport.xml 생성과 <import> 태그 추가

- 1. appCtx1.xml을 복사해서 appCtxImport.xml 생성



- 2. appCtxImport.xml에 appCtx2.xml, appCtx3.xml을 임포트하는 <import> 태그 추가

코드 4-23

ch04_pjt_01\src\main\resources\appCtxImport.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <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">
04
05     <import resource="classpath:appCtx2.xml"/>
06     <import resource="classpath:appCtx3.xml"/>
07
08     <!-- InitSampleData 빈 -->
09     <bean id="initSampleData"
10         class="ch04_pjt_01.ems.utils.InitSampleData">
11         ...생략...
12
13 </beans>
```


2. 파일 분리 방법 2: <import> 태그 사용하기

■ IoC 컨테이너를 생성하는 부분 수정

- 바로 앞에서 수정했던 main() 메서드 수정
- before

```
String[] appCtxs = {"classpath:appCtx1.xml", "classpath:appCtx2.xml",  
    "classpath:appCtx3.xml"};  
GenericXmlApplicationContext ctx =  
    new GenericXmlApplicationContext(appCtxs);
```

- after

```
GenericXmlApplicationContext ctx =  
    new GenericXmlApplicationContext("classpath:appCtxImport.xml");
```

2. 파일 분리 방법 2: <import> 태그 사용하기

■ <import> 태그를 이용한 분리

- Generic XmlApplicationContext를 생성할 때 설정 파일을 1개만 이용하면 됨

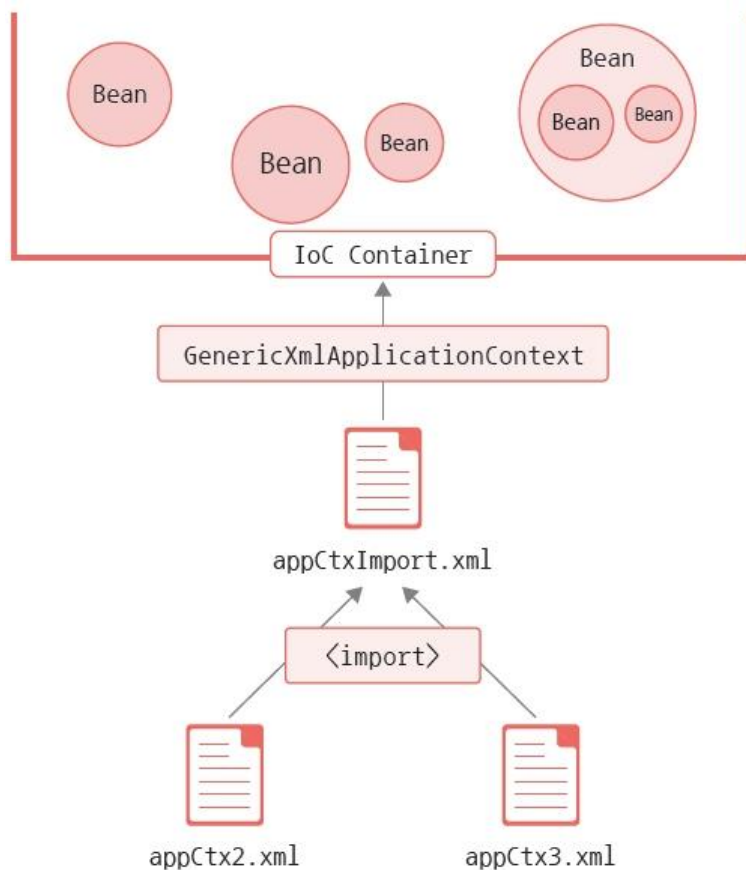


그림 4-18 <import>를 이용해서 n개의 설정 파일을 하나의 설정 파일처럼 사용한다.

Section 06

스프링 빈 범위

1. 스프링 빈 범위

■ 스프링 빈 범위(Spring Bean Scope)

- IoC 컨테이너에 생성된 빈을 `getBean()`으로 호출하면 항상 동일한 객체가 반환됨
- 스프링이 기본적으로 객체 범위를 싱글톤으로 관리하기 때문
- 싱글톤이란?
 - ✓ 객체를 호출할 때마다 새로 생성하지 않고 기존의 객체를 재사용함

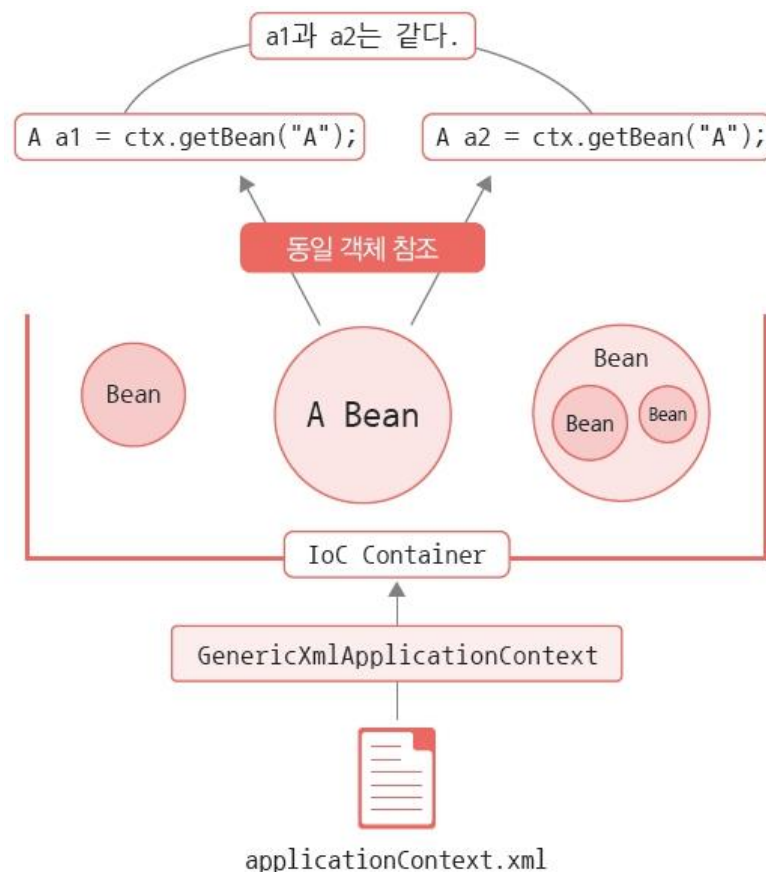


그림 4-19 a1과 a2는 동일 객체를 참조함

1. 스프링 빈 범위

■ ch04_pjt_02 예제

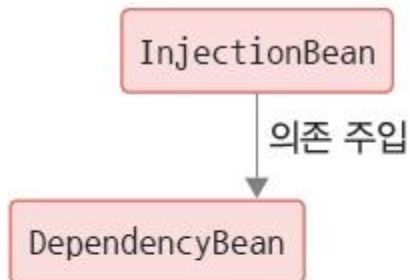
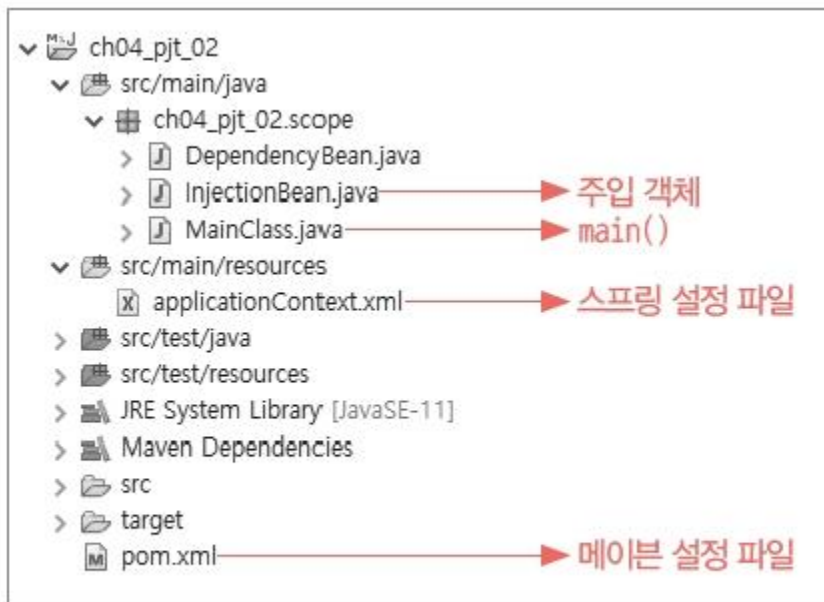


그림 4-20 프로젝트 구조와 의존 관계

1. 스프링 빈 범위

■ 1. 메이븐 설정 파일

- pom.xml에서 spring-context 모듈과 빌드 설정

코드 4-24

ch04_pjt_02\pom.xml

```
01 <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/xsd/maven-4.0.0.xsd">
02     <modelVersion>4.0.0</modelVersion>
03     <groupId>spring5</groupId>
04     <artifactId>ch04_pjt_02</artifactId>
05     <version>0.0.1-SNAPSHOT</version>
06
07     <!-- spring-context 모듈 -->
08     <dependencies>
09         <dependency>
10             <groupId>org.springframework</groupId>
11             <artifactId>spring-context</artifactId>
12             <version>5.2.9.RELEASE</version>
13         </dependency>
14     </dependencies>
15
16     <!-- 빌드 설정 -->
17     <build>
18         <plugins>
19             <plugin>
20                 <artifactId>maven-compiler-plugin</artifactId>
```

```
21                 <version>3.1</version>
22                 <configuration>
23                     <source>11</source>
24                     <target>11</target>
25                     <encoding>utf-8</encoding>
26                 </configuration>
27             </plugin>
28         </plugins>
29     </build>
30
31 </project>
```

1. 스프링 빈 범위

■ 2. DependencyBean 클래스

- 객체가 생성될 때 생성자를 통해 InjectionBean을 주입받음

코드 4-25

ch04_pjt_02\src\main\java\ch04_pjt_02\scope\DependencyBean.java

```
01 package ch04_pjt_02.scope;  
02  
03 public class DependencyBean {  
04  
05     InjectionBean injectionBean;  
06  
07     public DependencyBean(InjectionBean injectionBean) {  
08         this.injectionBean = injectionBean;  
09     }  
10 }
```

■ 3. InjectionBean 클래스

- 멤버가 없는 객체로 컴파일 단계에서 디폴트 생성자가 추가함

코드 4-26

ch04_pjt_02\src\main\java\ch04_pjt_02\scope\InjectionBean.java

```
01 package ch04_pjt_02.scope;  
02  
03 public class InjectionBean {  
04  
05 }
```

1. 스프링 빈 범위

■ 4. 스프링 설정 파일

- InjectionBean 빈을 생성해서 DependencyBean에 생성자를 통해 객체 주입

코드 4-27

ch04_pjt_02\src\main\resources\applicationContext.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <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">
04
05     <bean id="injectionBean" class="ch04_pjt_02.scope.InjectionBean"/>
06
07     <bean id="dependencyBean" class="ch04_pjt_02.scope.DependencyBean">
08         <constructor-arg ref="injectionBean"/>
09     </bean>
10
11 </beans>
```


1. 스프링 빈 범위

■ 5. MainClass 클래스

- main() 메서드에서 IoC 컨테이너를 생성하고 DependencyBean 빈을 가져옴

코드 4-28

ch04_pjt_02\src\main\java\ch04_pjt_02\scope\MainClass.java

```
01 package ch04_pjt_02.scope;
02
03 import org.springframework.context.support.GenericXmlApplicationContext;
04
05 public class MainClass {
06
07     public static void main(String[] args) {
08
09         GenericXmlApplicationContext ctx =
10             new GenericXmlApplicationContext("classpath:applicationContext.xml");
11
12         DependencyBean dependencyBean_01 = ctx.getBean("dependencyBean",
13             DependencyBean.class);
14         DependencyBean dependencyBean_02 = ctx.getBean("dependencyBean",
15             DependencyBean.class);
16
17         if(dependencyBean_01.equals(dependencyBean_02)) {
18             System.out.println("dependencyBean_01 == dependencyBean_02");
19         } else {
20             System.out.println("dependencyBean_01 != dependencyBean_02");
21         }
22
23         ctx.close();
24     }
25 }
```

1. 스프링 빈 범위

■ 5. MainClass 클래스

- 스프링은 싱글턴이기 때문에 main()에서 가져온 dependencyBean_01과 dependencyBean_02는 동일한 객체를 참조함

실행 결과

```
dependencyBean_01 == dependencyBean_02
```

- 싱글턴(singleton)
 - ✓ 스프링 컨테이너에서 생성된 빈을 getBean() 메서드로 이용해서 가져오면 가져올 때마다 새로운 빈이 생성되는 것이 아닌 동일한 빈이 계속해서 사용됨
 - ✓ 개발자가 특별하게 명시하지 않으면 기본적으로 모든 빈은 싱글턴 범위를 가짐
- 프로토타입(prototype)
 - ✓ 싱글턴 범위의 반대 개념
 - ✓ 프로토타입의 경우 개발자의 설정이 필요함
 - ✓ 스프링 설정 파일에서 빈을 정의할 때 scope 속성을 명시해주면 됨

1. 스프링 빈 범위

■ 6. 스프링 설정 파일

- applicationContext.xml에서 scope="prototype"을 명시하고, 프로그램 실행
- 다음은 [코드 4-27]의 7행을 수정한 코드

코드 4-29

ch04_pjt_02\src\main\resources\applicationContext.xml

```
01 <?xml version="1.0" encoding="UTF-8"
02
03 <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">
04
05     <bean id="injectionBean" class="ch04_pjt_02.scope.InjectionBean"/>
06
07     <bean id="dependencyBean" class="ch04_pjt_02.scope.DependencyBean"
    scope="prototype">
08         <constructor-arg ref="injectionBean"/>
09     </bean>
10
11 </beans>
```

실행 결과

dependencyBean_01 != dependencyBean_02