올인원
스프링 프레임워크

Chapter 06
@Configuration을 이용한 빈 생성

목차

1. XML 파일을 Java 파일로 변경하기
2. Java 파일 분리
3. @Import 애너테이션

학습목표

- @Configuration과 @Bean 애너테이션의 사용 방법을 학습합니다.
- 스프링 설정 파일을 분리하는 방법을 살펴봅니다.
- @Import 애너테이션에 대해서 살펴봅니다.

Section 01

XML 파일을
Java 파일로 변경하기

■ Java 파일을 이용한 스프링 설정 파일

- 작업 순서

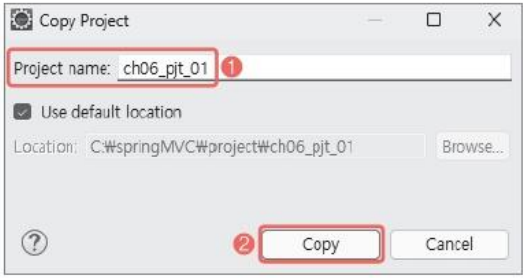
① 이클립스에서 ch04_pjt_01을 복사해서 ch06_pjt_01을 만든다.

② [src/main/resources]에 있는 applicationContext.xml을 Java 파일로 변경한다.

③ ❷번에서 제작한 Java 파일을 이용해서 스프링 컨테이너를 초기화한다.

■ Java 파일을 이용한 스프링 설정 파일

- 1. 이클립스의 Package Explorer에서 프로젝트 ch04_pjt_01을 복사, 붙여넣기
- 2. [Copy Project] 창의 Project name을 ch06_pjt_01로 변경하고 <Copy> 클릭



- 3. 프로젝트 생성 완료 후 pom.xml에서 <artifactId>를 ch06_pjt_01로 수정

```
<artifactId>ch06_pjt_01</artifactId>
```

- 4. [src/main/java] 내부의 패키지 이름을 ch06_pjt_01로 변경

■ Java 파일을 이용한 스프링 설정 파일

- 5. [src/main/resources] 내부의 applicationContext.xml에서 모든 패키지명을 ch06_pjt_01로 수정

```
<!-- InitSampleData 빈 -->
<bean id="initSampleData"
    class="ch04_pjt_01.ems.utils.InitSampleData">
    <property name="sNums">
        <array>
            <value>hbs001</value>
        ...생략...
```

```
<!-- InitSampleData 빈 -->
<bean id="initSampleData"
    class="ch06_pjt_01.ems.utils.InitSampleData">
    <property name="sNums">
        <array>
            <value>hbs001</value>
        ...생략...
```

2. XML 파일의 기능을 Java 파일로 변경하기

applicationContext.xml

- 하나의 파일에 모든 스프링 설정이 가능한 것을 확인할 수 있음
- DAO와 각종 Service, Bean 객체를 생성하는 코드

```
<bean id="initSampleData" class="ch06_pjt_01.ems.utils.InitSampleData">
<bean id="studentDao" class="ch06_pjt_01.ems.member.dao.StudentDao" />
<bean id="studentRegisterService"
      class="ch06_pjt_01.ems.member.service.StudentRegisterService">
    <constructor-arg ref="studentDao" />
</bean>
<bean id="studentModifyService"
      class="ch06_pjt_01.ems.member.service.StudentModifyService">
    <constructor-arg ref="studentDao" />
</bean>
<bean id="dev_DBConnectionInfoDev" class="ch06_pjt_01.ems.member.DBConnectionInfo">
    <property name="url" value="000.000.000.000" />
    <property name="userId" value="admin" />
    <property name="userPw" value="0000" />
</bean>
...
```

2. XML 파일의 기능을 Java 파일로 변경하기

■ applicationContext.xml을 자바 파일로 변경하기

- 패키지를 하나 추가하고 클래스를 만들어 빈 객체 생성
  - ✓ 패키지명: ch06_pjt_01.ems.configuration

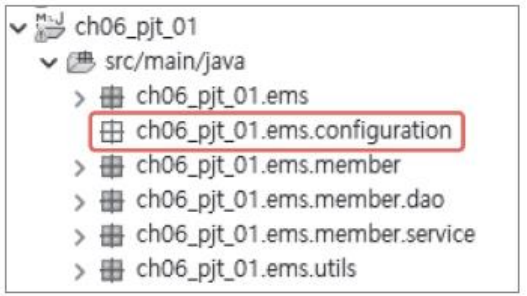


그림 6-1 ch06_pjt_01의 전체 패키지 구조

■ 자바를 이용한 스프링 설정 파일

- ch06_pjt_01.ems.configuration 패키지에 생성
- MemberConfig.java 파일을 만들고 애너테이션을 이용한 스프링 설정 파일 생성
- applicationContext.xml의 역할을 대신함

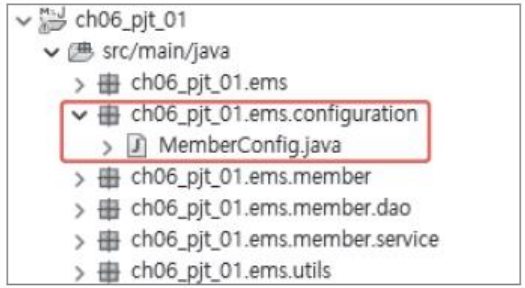


그림 6-2 MemberConfig.java 생성

2. XML 파일의 기능을 Java 파일로 변경하기

@Configuration

- XML을 이용하지 않고 애너테이션을 이용한 스프링 설정 파일을 만들기 위해 사용하는 애너테이션
- 자바 파일의 클래스 선언부에 명시함

코드 6-1 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 import org.springframework.context.annotation.Configuration;
04
05 @Configuration
06 public class MemberConfig {
07
08 }
```

@Bean

- MemberConfig.java를 이용해서 빈 객체를 생성하기 위한 애너테이션
- StudentDao 빈 객체를 생성하기 위한 코드

코드 6-2 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig.java

```
01 ...package, import문 생략...
02
03 @Configuration
04 public class MemberConfig {
05
06     @Bean
07     public StudentDao studentDao() {
08         return new StudentDao();
09     }
10 }
```

■ @Bean

- 빈을 생성하는 메서드의 문법 구조
  - ✓ 메서드 이름은 빈 객체의 id이고 반환되는 데이터 타입은 빈 객체의 타입명

applicationContext.xml

```
<bean id="stuentDao"
class="ch06_pjt_01.ems.member.dao.StudentDao"/)
```

MemberConfig.java

```
@Bean
public StudentDao studentDao() {
    return new StudentDao();
}
```

그림 6-3 XML 파일과 Java 파일에서 빈 객체를 생성하는 문법 비교

3. 생성자를 이용해 의존 객체 주입하기

StudentRegisterService를 이용한 주입

- StudentRegisterService 빈 객체를 생성하는 코드와 문법 구조

코드 6-3 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig.java

```
01 ...package, import문 생략...
02
03 @Configuration
04 public class MemberConfig {
05
06     @Bean
07     public StudentDao studentDao() {
08         return new StudentDao();
09     }
10
11     @Bean
12     public StudentRegisterService studentRegisterService() {
13         return new StudentRegisterService(studentDao());
14     }
15 }
```

3. 생성자를 이용해 의존 객체 주입하기

■ StudentRegisterService를 이용한 주입

- StudentRegisterService 빈 객체를 생성하는 코드와 문법 구조

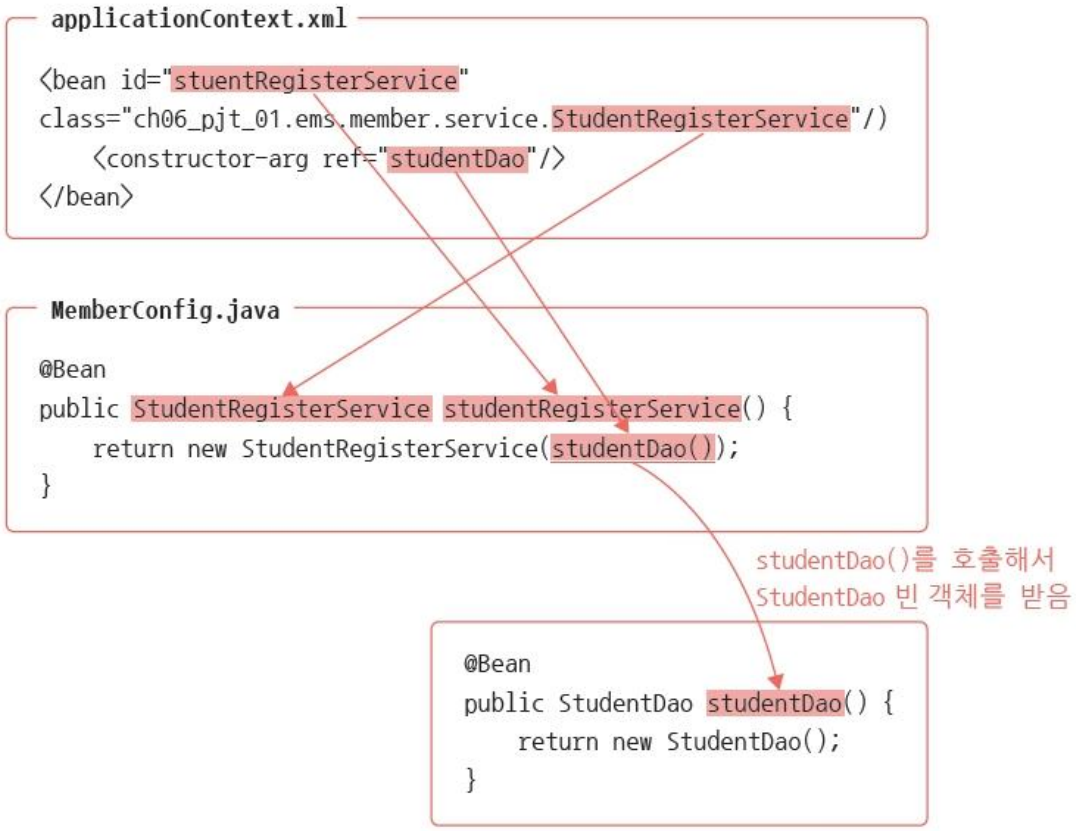


그림 6-4 StudentRegisterService 빈 객체가 생성되는 시점에 인자로 의존 객체(StudentDao) 주입

3. 생성자를 이용해 의존 객체 주입하기

■ setter 메서드를 이용한 주입

- DBConnectionInfo 빈 객체를 생성하는 코드와 문법 구조

코드 6-4 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig.java

```
01 ...package, import문 생략...
02
03 @Configuration
04 public class MemberConfig {
05     ...생략...
06
07     @Bean
08     public DBConnectionInfo dev_DBConnectionInfoDev() {
09         DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
10         dbConnectionInfo.setUrl("000.000.000.000");
11         dbConnectionInfo.setUserId("admin");
12         dbConnectionInfo.setUserPw("0000");
13
14         return dbConnectionInfo;
15     }
16
17     @Bean
18     public DBConnectionInfo real_DBConnectionInfoDev() {
19         DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
20         dbConnectionInfo.setUrl("111.111.111.111");
21         dbConnectionInfo.setUserId("master");
22         dbConnectionInfo.setUserPw("1111");
23
24         return dbConnectionInfo;
25     }
26
27 }
```

3. 생성자를 이용해 의존 객체 주입하기

■ setter 메서드를 이용한 주입

- setter 메서드에 주입되는 객체의 타입이 List와 Map 타입인 경우

applicationContext.xml

```
<bean id="dev_DBConnectionInfoDev" class="ch06_pjt_01.ems.member.DBConnectionInfo">
    <property name="url" value="000.000.000.000" />
    <property name="userId" value="admin" />
    <property name="userPw" value="0000" />
</bean>
```

MemberConfig.java

```
@Bean
public DBConnectionInfo dev_DBConnectionInfoDev() {
    DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
    dbConnectionInfo.setUrl("000.000.000.000");
    dbConnectionInfo.setUserId("admin");
    dbConnectionInfo.setUserPw("0000");

    return dbConnectionInfo;
}
```

applicationContext.xml

```
<bean id="real_DBConnectionInfo" class="ch06_pjt_01.ems.member.DBConnectionInfo">
    <property name="url" value="111.111.111.111" />
    <property name="userId" value="master" />
    <property name="userPw" value="1111" />
</bean>
```

MemberConfig.java

```
@Bean
public DBConnectionInfo real_DBConnectionInfoDev() {
    DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
    dbConnectionInfo.setUrl("111.111.111.111");
    dbConnectionInfo.setUserId("master");
    dbConnectionInfo.setUserPw("1111");

    return dbConnectionInfo;
}
```

그림 6-5 DBConnectionInfo의 setter를 이용해서 빈 객체의 프로퍼티를 초기화

3. 생성자를 이용해 의존 객체 주입하기

■ setter 메서드를 이용한 주입

- EMSInformationService 빈 객체를 생성하는 코드

코드 6-5 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig.java

```
01 ...package, import문 생략...
02
03 @Configuration
04 public class MemberConfig {
05     ...생략...
06
07     @Bean
08     public EMSInformationService eMSInformationService() {
09
10         EMSInformationService emsInformationService = new EMSInformationService();
11         emsInformationService.setInfo("Education Management System program was
   developed in 2022.");
12          emsInformationService.setCopyRight("COPYRIGHT(C) 2022 EMS CO., LTD.
   ALL RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.");
13          emsInformationService.setVer("The version is 1.0");
14          emsInformationService.setsYear(2022);
15          emsInformationService.setsMonth(3);
16          emsInformationService.setsDay(1);
17          emsInformationService.seteYear(2022);
18          emsInformationService.seteMonth(4);
19          emsInformationService.seteDay(30);
20
```

3. 생성자를 이용해 의존 객체 주입하기

■ setter 메서드를 이용한 주입

```
21          List<String> developers = new ArrayList<String>();
22          developers.add("Cheney.");
23          developers.add("Eloy.");
24          developers.add("Jasper.");
25          developers.add("Dillon.");
26          developers.add("Kian.");
27          emsInformationService.setDevelopers(developers);
28
29          Map<String, String> administrators = new HashMap<String, String>();
30          administrators.put("Cheney", "cheney@springPjt.org");
31          administrators.put("Jasper", "jasper@springPjt.org");
32          emsInformationService.setAdministrators(administrators);
33
34          Map<String, DBConnectionInfo> dbInfos = new HashMap<String,
    DBConnectionInfo>();
35          dbInfos.put("dev", dev_DBConnectionInfoDev());
36          dbInfos.put("real", real_DBConnectionInfoDev());
37          emsInformationService.setDbInfos(dbInfos);
38
39          return emsInformationService;
40
41      }
42  }
```

3. 생성자를 이용해 의존 객체 주입하기

■ setter 메서드를 이용한 주입

applicationContext.xml

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

MemberConfig.java

```
@Bean
 public EMSInformationService eMSInformationService() {

    ...생략...
    List<String> developers = new ArrayList<String>();
    developers.add("Cheney.");
    developers.add("Eloy.");
    developers.add("Jasper.");
    developers.add("Dillon.");
    developers.add("Kian.");
    emsInformationService.setDevelopers(developers);
    ...생략...
}
```

그림 6-6 주입되는 의존 객체의 타입이 List인 경우 List를 이용해서 빈 객체의 프로퍼티를 초기화

3. 생성자를 이용해 의존 객체 주입하기

■ setter 메서드를 이용한 주입

applicationContext.xml

```
<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>
```

MemberConfig.java

```
@Bean
 public EMSInformationService eMSInformationService() {

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

그림 6-7 주입되는 의존 객체의 타입이 Map인 경우 Map을 이용해서 빈 객체의 프로퍼티 초기화

<여기서 잠깐!> EMSInformationService의 administrators Map의 value가 String이 아닌 참조형 객체 타입인 경우

방금 초기화한 EMSInformationService의 administrators는 Map의 value가 String 타입이지만, value가 String이 아닌 참조형 객체 타입이면 어떻게 할까요?

다음 슬라이드의 [그림 6-8]은 EMSInformationService의 dbInfos를 초기화하는 코드를 보여줍니다. dbInfos는 value로 DBConnectionInfo 객체를 이용하기 때문에 DBConnectionInfo 객체를 넣어줘야 합니다.

3. 생성자를 이용해 의존 객체 주입하기

applicationContext.xml

```
<property name="dbInfos">
    <map>
        <entry>
            <key>
                <value>dev</value>
            </key>
            <ref bean="dev_DBConnectionInfoDev" />
        </entry>
        <entry>
            <key>
                <value>real</value>
            </key>
            <ref bean="real_DBConnectionInfo" />
        </entry>
    </map>
</property>
```

MemberConfig.java

```
@Bean
 public EMSInformationService eMSInformationService() {

    ···생략···
    Map<String, DBConnectionInfo> dbInfos = new HashMap<String, DBConnectionInfo>();
    dbInfos.put("dev", dev_DBConnectionInfoDev());
    dbInfos.put("real", real_DBConnectionInfoDev());
    emsInformationService.setDbInfos(dbInfos);
    ···생략···
}
```

MemberConfig.java

```
@Bean
 public DBConnectionInfo dev_DBConnectionInfoDev() {
    ···생략···
    return dbConnectionInfo;
}
```

MemberConfig.java

```
@Bean
 public DBConnectionInfo real_DBConnectionInfoDev() {
    ···생략···
    return dbConnectionInfo;
}
```

그림 6-8 value가 객체인 경우 빈 객체를 주입해서 프로퍼티를 초기화

3. 생성자를 이용해 의존 객체 주입하기

■ InitSampleData 빈 객체를 포함한 전체 빈 객체를 생성하는 MemberConfig의 전체 코드

코드 6-6 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig.java

```
01 package ch06_pjt_01.ems.configuration;
02 ...package, import문 생략...
03
04 @Configuration
05 public class MemberConfig {
06
07     @Bean
08     public InitSampleData initSampleData() {
09
10         InitSampleData initSampleData = new InitSampleData();
11
12         String[] sNums = {"hbs001", "hbs002", "hbs003", "hbs004", "hbs005"};
13         initSampleData.setsNums(sNums);
14
15         String[] sIds = {"rabbit", "hippo", "raccoon", "elephant", "lion"};
16         initSampleData.setsIds(sIds);
17
18         String[] sPws = {"p0001", "p0002", "p0003", "p0004", "p0005"};
19         initSampleData.setsPws(sPws);
20
21         String[] sNames = {"agatha", "barbara", "chris", "doris", "elva"};
22         initSampleData.setsNames(sNames);
23
24         int[] sAges = {19, 22, 20, 27, 19};
25         initSampleData.setsAges(sAges);
26
```

3. 생성자를 이용해 의존 객체 주입하기

```
27          char[] sGenders = {'M', 'W', 'W', 'M', 'M'};
28          initSampleData.setsGenders(sGenders);
29
30          String[] sMajors = {"English", "Korean", "French", "Philosophy", "History"};
31          initSampleData.setsMajors(sMajors);
32
33          return initSampleData;
34      }
35
36      @Bean
37      public StudentDao studentDao() {
38          return new StudentDao();
39      }
40
41      @Bean
42      public StudentRegisterService studentRegisterService() {
43          return new StudentRegisterService(studentDao());
44      }
45
46      @Bean
47      public StudentModifyService studentModifyService() {
48
49          return new StudentModifyService(studentDao());
50
51      }
52
53      @Bean
54      public StudentDeleteService studentDeleteService() {
55          return new StudentDeleteService(studentDao());
56      }
57
```

3. 생성자를 이용해 의존 객체 주입하기

```
58      @Bean
59      public StudentSelectService studentSelectService() {
60          return new StudentSelectService(studentDao());
61      }
62
63      @Bean
64      public StudentAllSelectService studentAllSelectService() {
65          return new StudentAllSelectService(studentDao());
66      }
67
68      @Bean
69      public PrintStudentInformationService printStudentInformationService() {
70          return new PrintStudentInformationService(studentAllSelectService());
71      }
72
73      @Bean
74      public DBConnectionInfo dev_DBConnectionInfoDev() {
75          DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
76          dbConnectionInfo.setUrl("000.000.000.000");
77          dbConnectionInfo.setUserId("admin");
78          dbConnectionInfo.setUserPw("0000");
79
80          return dbConnectionInfo;
81      }
82
83      @Bean
84      public DBConnectionInfo real_DBConnectionInfoDev() {
85          DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
86          dbConnectionInfo.setUrl("111.111.111.111");
87          dbConnectionInfo.setUserId("master");
88          dbConnectionInfo.setUserPw("1111");
89
90          return dbConnectionInfo;
91      }
```

3. 생성자를 이용해 의존 객체 주입하기

```
92
93      @Bean
94      public EMSInformationService eMSInformationService() {
95          EMSInformationService emsInformationService = new EMSInformationService();
96          emsInformationService.setInfo("Education Management System program was
    developed in 2022.");
97          emsInformationService.setCopyRight("COPYRIGHT(C) 2022 EMS CO., LTD. ALL
    RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.");
98          emsInformationService.setVer("The version is 1.0");
99          emsInformationService.setsYear(2022);
100         emsInformationService.setsMonth(3);
101         emsInformationService.setsDay(1);
102         emsInformationService.seteYear(2022);
103         emsInformationService.seteMonth(4);
104         emsInformationService.seteDay(30);
105
106         List<String> developers = new ArrayList<String>();
107         developers.add("Cheney.");
108         developers.add("Eloy.");
109         developers.add("Jasper.");
110         developers.add("Dillon.");
111         developers.add("Kian.");
112         emsInformationService.setDevelopers(developers);
113
114         Map<String, String> administrators = new HashMap<String, String>();
115         administrators.put("Cheney", "cheney@springPjt.org");
116         administrators.put("Jasper", "jasper@springPjt.org");
117         emsInformationService.setAdministrators(administrators);
```

3. 생성자를 이용해 의존 객체 주입하기

```
118
119         Map<String, DBConnectionInfo> dbInfos = new HashMap<String,
    DBConnectionInfo>();
120         dbInfos.put("dev", dev_DBConnectionInfoDev());
121         dbInfos.put("real", real_DBConnectionInfoDev());
122         emsInformationService.setDbInfos(dbInfos);
123
124         return emsInformationService;
125     }
126 }
```

■ 스프링 컨테이너 초기화

- 1. ch06_pjt_01.ems 패키지의 MainClass.java 파일을 복사, 붙여넣기해 MainClassUseConfig.java 파일 생성
  - ✓ 스프링 설정 파일이 기존 XML 파일(applicationContext.xml)에서 애너테이션을 이용한 Java 파일(MemberConfig.java)로 변경됨
  - ✓ **스프링 컨테이너가 초기화되는 단계의 코드만 수정하면 됨**
- 2. MainClassUseConfig.java에서 스프링 컨테이너를 초기화하는 부분 수정

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

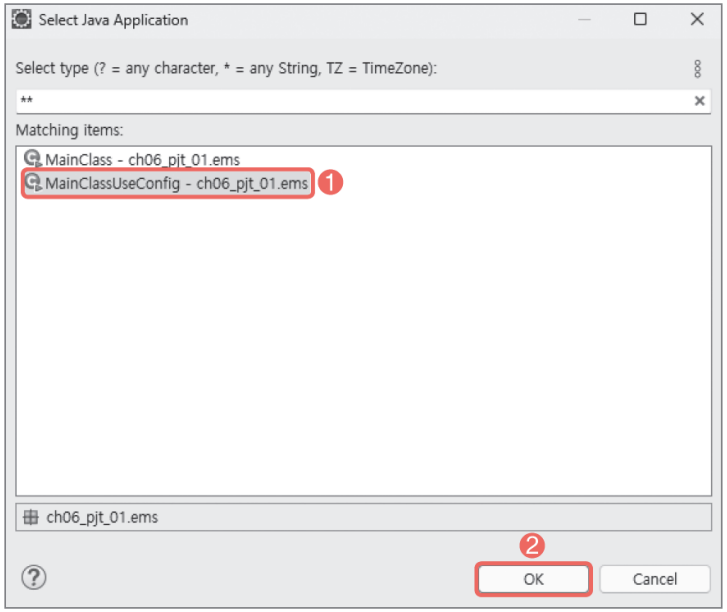
```
AnnotationConfigApplicationContext ctx =
                new AnnotationConfigApplicationContext(MemberConfig.class);
```

■ 스프링 컨테이너 초기화

- 3. 프로그램 실행

✓ ch06_pjt_01이 ch04_pjt_01의 복사본으로 별도의 기능 추가 없이 단지 스프링 설정 파일만 변경했기 때문에 실행 결과는 동일함

✓ 프로젝트를 실행하면 나오는 [Select] 창에서 MainClassUseConfig.java를 선택해 프로그램 실행



실행 결과

```
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:M|sMajor:Philosophy
sNum:hbs001|sId:rabbit|sPw:p0001|sName:agatha|sAge:19|sGender:M|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:M|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:M|sMajor:Philosophy
sNum:hbs001|sId:rabbit|sPw:p0001|sName:agatha|sAge:19|sGender:M|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:M|sMajor:History
sNum:hbs006|sId:deer|sPw:p0006|sName:melissa|sAge:26|sGender:w|sMajor:Music
END ----------------------------
...생략...
```

Section 02

Java 파일 분리

1. 분리할 파일 생성하기

■ 파일을 분리하는 이유

- 기능 단위로 스프링 설정 파일을 분리하면 개발의 편의성을 높여줌
- Java 파일을 이용해서 스프링 설정 파일을 제작할 때도 파일을 분리할 수 있음
- MemberConfig.java 파일을 3개의 파일로 분리하고 스프링 컨테이너 초기화에 이용해보기

■ 작업 순서

① MemberConfig1.java, MemberConfig2.java, MemberConfig3.java 파일을 만든다.

② MemberConfig.java의 코드를 ❶번에서 만든 각각의 파일들에 분리해서 나눈다.

■ 분리할 파일 생성

- ch06_pjt_01.ems.configuration 패키지에 MemberConfig1.java, MemberConfig2.java, MemberConfig3.java 파일 생성
  - ✓ 스프링 컨테이너에 빈 객체를 생성하는 역할을 해야 하기 때문에 @Configuration를 명시해야 함

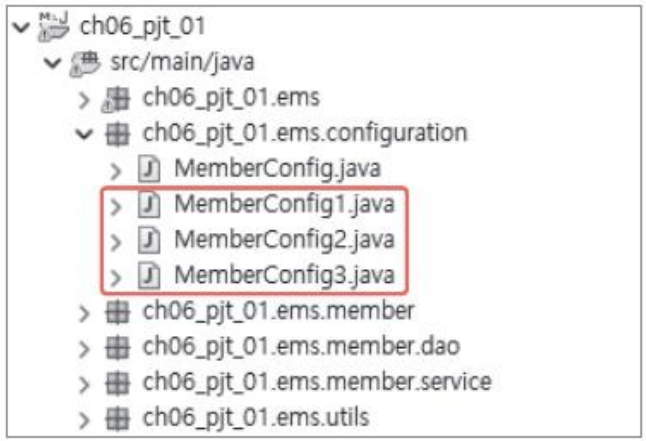


그림 6-9 MemberConfig1.java~MemberConfig3.java 생성

1. 분리할 파일 생성하기

■ 분리할 파일 생성

- @Configuration 적용

코드 6-7 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig1.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 import org.springframework.context.annotation.Configuration
04
05 @Configuration
06 public class MemberConfig1 {
07
08 }
```

코드 6-8 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig2.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 import org.springframework.context.annotation.Configuration
04
05 @Configuration
06 public class MemberConfig2 {
07
08 }
```

코드 6-9 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig3.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 import org.springframework.context.annotation.Configuration
04
05 @Configuration
06 public class MemberConfig3 {
07
08 }
```

MemberConfig1.java

- 학생 정보를 추가, 수정, 검색, 삭제하는 Service 및 DAO 빈 객체를 스프링 컨테이너에 생성하는 코드로 구성됨
  - ✓ 새롭게 코딩하는 것이 아니라 MeberConfig.java의 코드 중 InitSampleData부터 PrintStudentInformationService까지 복사하여 사용

코드 6-10 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig1.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 ...import문 생략...
04
05 @Configuration
06 public class MemberConfig1 {
07
08     @Bean
09     public InitSampleData initSampleData() {
10
11         InitSampleData initSampleData = new InitSampleData();
12
13         String[] sNums = {"hbs001", "hbs002", "hbs003", "hbs004", "hbs005"};
14         initSampleData.setsNums(sNums);
15
16         String[] sIds = {"rabbit", "hippo", "raccoon", "elephant", "lion"};
17         initSampleData.setsIds(sIds);
18
19         String[] sPws = {"p0001", "p0002", "p0003", "p0004", "p0005"};
20         initSampleData.setsPws(sPws);
```

2. Java 코드 분리하기

```
21
22          String[] sNames = {"agatha", "barbara", "chris", "doris", "elva"};
23          initSampleData.setsNames(sNames);
24
25          int[] sAges = {19, 22, 20, 27, 19};
26          initSampleData.setsAges(sAges);
27
28          char[] sGenders = {'M', 'W', 'W', 'M', 'M'};
29          initSampleData.setsGenders(sGenders);
30
31          String[] sMajors = {"English", "Korean", "French", "Philosophy", "History"};
32          initSampleData.setsMajors(sMajors);
33
34          return initSampleData;
35      }
36
37      @Bean
38      public StudentDao studentDao() {
39          return new StudentDao();
40      }
41
42      @Bean
43      public StudentRegisterService studentRegisterService() {
44          return new StudentRegisterService(studentDao());
45      }
46
47      @Bean
48      public StudentModifyService studentModifyService() {
49
50          return new StudentModifyService(studentDao());
51
52      }
53
```

2. Java 코드 분리하기

```
54      @Bean
55      public StudentDeleteService studentDeleteService() {
56          return new StudentDeleteService(studentDao());
57      }
58
59      @Bean
60      public StudentSelectService studentSelectService() {
61          return new StudentSelectService(studentDao());
62      }
63
64      @Bean
65      public StudentAllSelectService studentAllSelectService() {
66          return new StudentAllSelectService(studentDao());
67      }
68
69      @Bean
70      public PrintStudentInformationService printStudentInformationService() {
71          return new PrintStudentInformationService(studentAllSelectService());
72      }
73  }
```

MemberConfig2.java

- 데이터베이스와 관련한 빈 객체를 스프링 컨테이너에 생성하는 코드로 구성됨

코드 6-11 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig2.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 ...import문 생략...
04
05 @Configuration
06 public class MemberConfig2 {
07
08     @Bean
09     public DBConnectionInfo dev_DBConnectionInfoDev() {
10         DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
11         dbConnectionInfo.setUrl("000.000.000.000");
12         dbConnectionInfo.setUserId("admin");
13         dbConnectionInfo.setUserPw("0000");
14
15         return dbConnectionInfo;
16     }
17
18     @Bean
19     public DBConnectionInfo real_DBConnectionInfoDev() {
20         DBConnectionInfo dbConnectionInfo = new DBConnectionInfo();
21         dbConnectionInfo.setUrl("111.111.111.111");
22         dbConnectionInfo.setUserId("master");
23         dbConnectionInfo.setUserPw("1111");
24
25         return dbConnectionInfo;
26     }
27 }
```

MemberConfig3.java

- 시스템 정보와 관련한 빈 객체를 생성하는 코드로 구성됨

코드 6-12 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig3.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 ...import문 생략...
04
05 @Configuration
06 public class MemberConfig3 {
07
08     @Bean
09     public EMSInformationService eMSInformationService() {
10         EMSInformationService emsInformationService = new EMSInformationService();
11         emsInformationService.setInfo("Education Management System program was
   developed in 2022.");
12         emsInformationService.setCopyRight("COPYRIGHT(C) 2022 EMS CO., LTD. ALL
   RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.");
13         emsInformationService.setVer("The version is 1.0");
14         emsInformationService.setsYear(2022);
15         emsInformationService.setsMonth(3);
16         emsInformationService.setsDay(1);
17         emsInformationService.seteYear(2022);
18         emsInformationService.seteMonth(4);
19         emsInformationService.seteDay(30);
20
```

2. Java 코드 분리하기

```
21          List<String> developers = new ArrayList<String>();
22          developers.add("Cheney.");
23          developers.add("Eloy.");
24          developers.add("Jasper.");
25          developers.add("Dillon.");
26          developers.add("Kian.");
27          emsInformationService.setDevelopers(developers);
28
29          Map<String, String> administrators = new HashMap<String, String>();
30          administrators.put("Cheney", "cheney@springPjt.org");
31          administrators.put("Jasper", "jasper@springPjt.org");
32          emsInformationService.setAdministrators(administrators);
33
34          Map<String, DBConnectionInfo> dbInfos = new HashMap<String,
    DBConnectionInfo>();
35          dbInfos.put("dev", dev_DBConnectionInfoDev());
36          dbInfos.put("real", real_DBConnectionInfoDev());
37          emsInformationService.setDbInfos(dbInfos);
38
39          return emsInformationService;
40      }
41  }
```

MemberConfig3.java

- 에러 발생
  - ✓ 에러 내용: dev_DBConnectionInfoDev()와 real_DBConnectionInfoDev()가 해당 메서드를 찾을 수 없음을 의미함
  - ✓ MemberConfig.java 파일을 분리했기 때문에 MemberConfig3.java에서 MemberConfig2에 선언된 dev_DBConnectionInfoDev()와 real_DBConnectionInfoDev() 메서드를 찾을 수 없기 때문에 발생하는 에러

```
51          Map<String, DBConnectionInfo> dbInfos = new HashMap<String, DBConnectionInfo>();
52          dbInfos.put("dev", dev_DBConnectionInfoDev());
53          dbInfos.put("real", real_DBConnectionInfoDev());
54          emsInformationService.setDbInfos(dbInfos);
55
```

그림 6-10 dbInfos에서 에러 발생

- 해결
  - ✓ @Autowired를 이용해서 의존 객체를 자동으로 주입시키면 됨

■ @Autowired에 의한 의존 객체를 자동 주입 설정

- @Autowired를 이용해서 dev_DBConnectionInfoDev와 real_DBConnectionInfoDev를 초기화하고 이를 MemberConfig3.java가 이용함

코드 6-13 ch06_pjt_01\src\main\java\ch06_pjt_01\ems\configuration\MemberConfig3.java

```
01 package ch06_pjt_01.ems.configuration;
02
03 ...import문 생략...
04
05 @Configuration
06 public class MemberConfig3 {
07
08     @Autowired
09     DBConnectionInfo dev_DBConnectionInfoDev;
10
11     @Autowired
12     DBConnectionInfo real_DBConnectionInfoDev;
13
14     @Bean
15     public EMSInformationService eMSInformationService() {
16
17         EMSInformationService emsInformationService = new EMSInformationService();
18         emsInformationService.setInfo("Education Management System program was
   developed in 2022.");
19         emsInformationService.setCopyRight("COPYRIGHT(C) 2022 EMS CO.,
   LTD. ALL RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.");
```

2. Java 코드 분리하기

```
20          emsInformationService.setVer("The version is 1.0");
21          emsInformationService.setsYear(2022);
22          emsInformationService.setsMonth(3);
23          emsInformationService.setsDay(1);
24          emsInformationService.seteYear(2022);
25          emsInformationService.seteMonth(4);
26          emsInformationService.seteDay(30);
27
28          List<String> developers = new ArrayList<String>();
29          developers.add("Cheney.");
30          developers.add("Eloy.");
31          developers.add("Jasper.");
32          developers.add("Dillon.");
33          developers.add("Kian.");
34          emsInformationService.setDevelopers(developers);
35
36          Map<String, String> administrators = new HashMap<String, String>();
37          administrators.put("Cheney", "cheney@springPjt.org");
38          administrators.put("Jasper", "jasper@springPjt.org");
39          emsInformationService.setAdministrators(administrators);
40
41          Map<String, DBConnectionInfo> dbInfos =
    new HashMap<String, DBConnectionInfo>();
42          dbInfos.put("dev", dev_DBConnectionInfoDev);
43          dbInfos.put("real", real_DBConnectionInfoDev);
44          emsInformationService.setDbInfos(dbInfos);
45
46          return emsInformationService;
47
48      }
49
50  }
```

MemberConfig3.java

- DBConnectionInfo 타입의 프로퍼티를 선언하고 @Autowired 애너테이션을 이용해서 의존 객체를 자동으로 주입함
  - ✓ DBConnectionInfo 타입의 객체를 필요로 하는 메서드에서 사용하기 위함
- MemberConfig3.java에는 DBConnectionInfo 타입의 빈 객체를 생성하는 코드가 없음
- 따라서 MemberConfig3에서 DBConnectionInfo 타입의 빈 객체를 사용하기 위해서는 의존 객체를 자동으로 주입받는 프로퍼티를 선언하고 사용해야 함
- 의존 객체 주입 코드

```
@Autowired
DBConnectionInfo dev_DBConnectionInfoDev;

@Autowired
DBConnectionInfo real_DBConnectionInfoDev;
```

MemberConfig3.java

MemberConfig2.java

```
@Bean
public DBConnectionInfo dev_DBConnectionInfoDev() {
    ...생략...
    return dbConnectionInfo;
}
```

```
@Bean
public DBConnectionInfo real_DBConnectionInfoDev() {
    ...생략...
    return dbConnectionInfo;
}
```

의존 객체 자동 주입

MemberConfig3.java

```
@Autowired
DBConnectionInfo dev_DBConnectionInfoDev;

@Autowired
DBConnectionInfo real_DBConnectionInfoDev;

...생략...

Map<String, DBConnectionInfo> dbInfos = new HashMap<String, DBConnectionInfo>();
dbInfos.put("dev", dev_DBConnectionInfoDev());
dbInfos.put("real", real_DBConnectionInfoDev());
emsInformationService.setDbInfos(dbInfos);
```

객체 사용

그림 6-11 @Autowired를 이용해서 필요한 객체 사용

■ 컨테이너 초기화 코드 수정

- 스프링 설정 파일을 분리했다면, 컨테이너를 초기화하는 코드 수정하기
  - ✓ ch06_pjt_01.ems 패키지에서 MainClassUseConfig.java 파일을 복사해서 MainClassUseConfigs.java를 만들어 수정함
- MemberConfig 대신 MemberConfig1, MemberConfig2, MemberConfig3을 사용하도록 수정

```
AnnotationConfigApplicationContext ctx =
            new AnnotationConfigApplicationContext(MemberConfig.class);
```

```
AnnotationConfigApplicationContext ctx =
            new AnnotationConfigApplicationContext(MemberConfig1.class, MemberConfig2.class,
MemberConfig3.class);
```

Section 03

@Import 애너테이션

1. @Import 애너테이션

■ <import> 태그를 이용한 외부 스프링 설정 파일 사용

```
<import resource="classpath:appCtx2.xml"/>
<import resource="classpath:appCtx3.xml"/>
```

■ @Import

- 스프링 설정 파일을 제작할 때 다른 Java 파일을 임포트하기 위한 애너테이션
- 1. MemberConfig1.java를 복사하여 MemberConfigImport.java를 생성한다.
- 2. MemberConfigImport.java에서 @Import 애너테이션을 추가한다.

```
@Configuration
public class MemberConfigImport {
    ...생략...
}
```

```
@Configuration
@Import({MemberConfig2.class, MemberConfig3.class})
public class MemberConfigImport {
    ...생략...
}
```

1. @Import 애너테이션

■ @Import 애너테이션 사용하기

- 만약 임포트할 클래스가 한 개라면 배열을 삭제하고, 간단하게 명시할 수 있음

```
@Import(MemberConfig2.class)
```

  - ✓ @Import 애너테이션에 의해서 하나의 파일로 결합됐으므로 스프링 컨테이너를 생성하는 코드도 간단해질 수 있음

- 3. MainClassUseConfigs.java를 복사해서 MainClassUseConfigsImport.java를 생성하고, MemberConfig1.class~MemberConfig3.class 대신 MemberConfigImport.class를 사용하도록 수정한다.

```
AnnotationConfigApplicationContext ctx =
            new AnnotationConfigApplicationContext(MemberConfig1.class, MemberConfig2.class,
MemberConfig3.class);
```

```
AnnotationConfigApplicationContext ctx =
            new AnnotationConfigApplicationContext(MemberConfigImport.class);
```