Devops 10

Spring

Code Analysis

NAVER Cloud Camp

2조 | 엄지훈, 김태웅, 김수정, 홍화연, 신지현

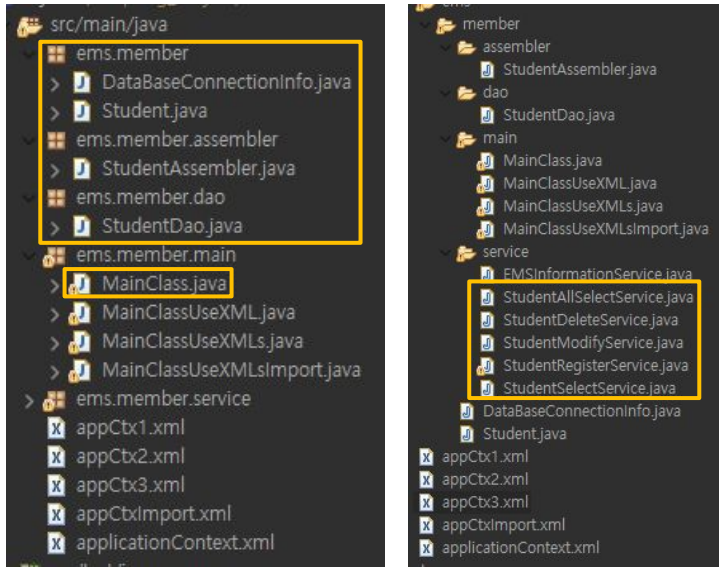
CHAPTER 1

CODE LIST

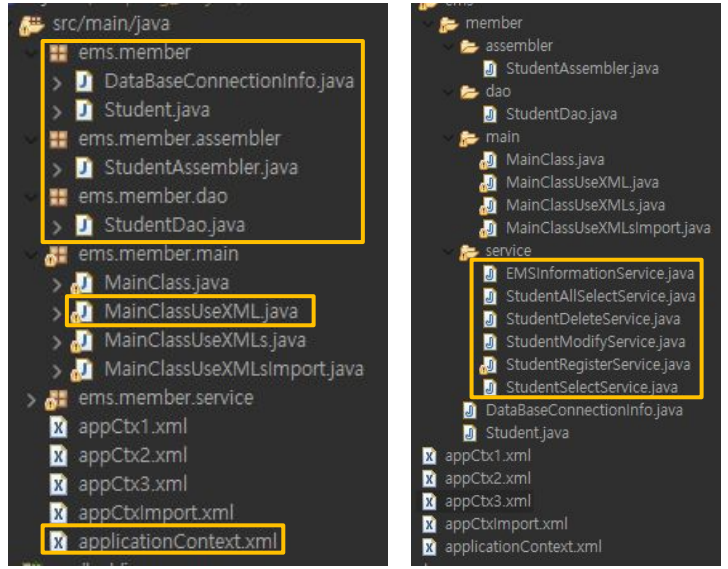
Code Analysis

파일 구성

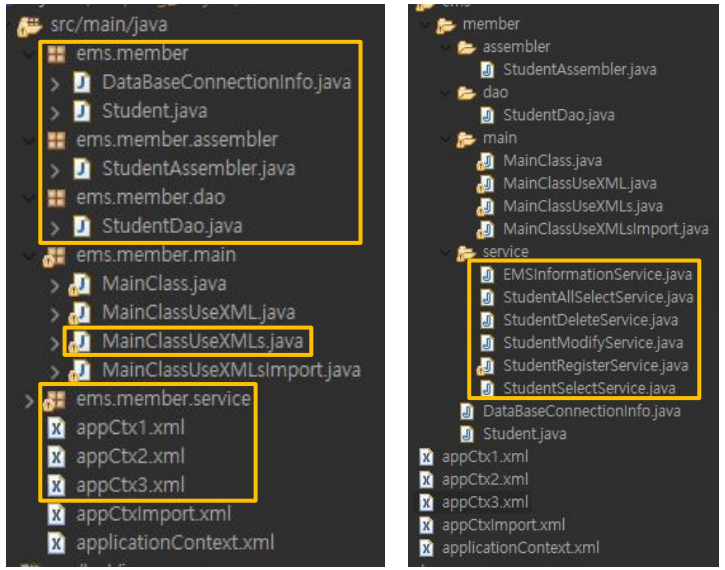
1. MainClass



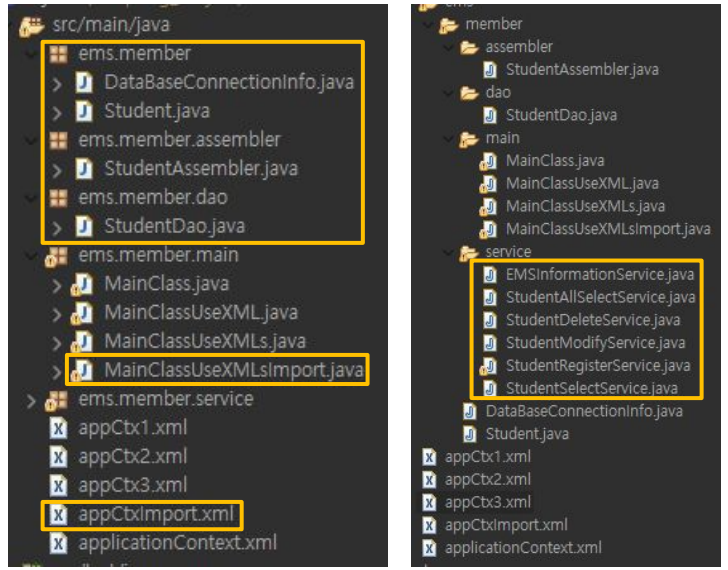
2. MainClassUseXML



3. MainClassUseXMLs

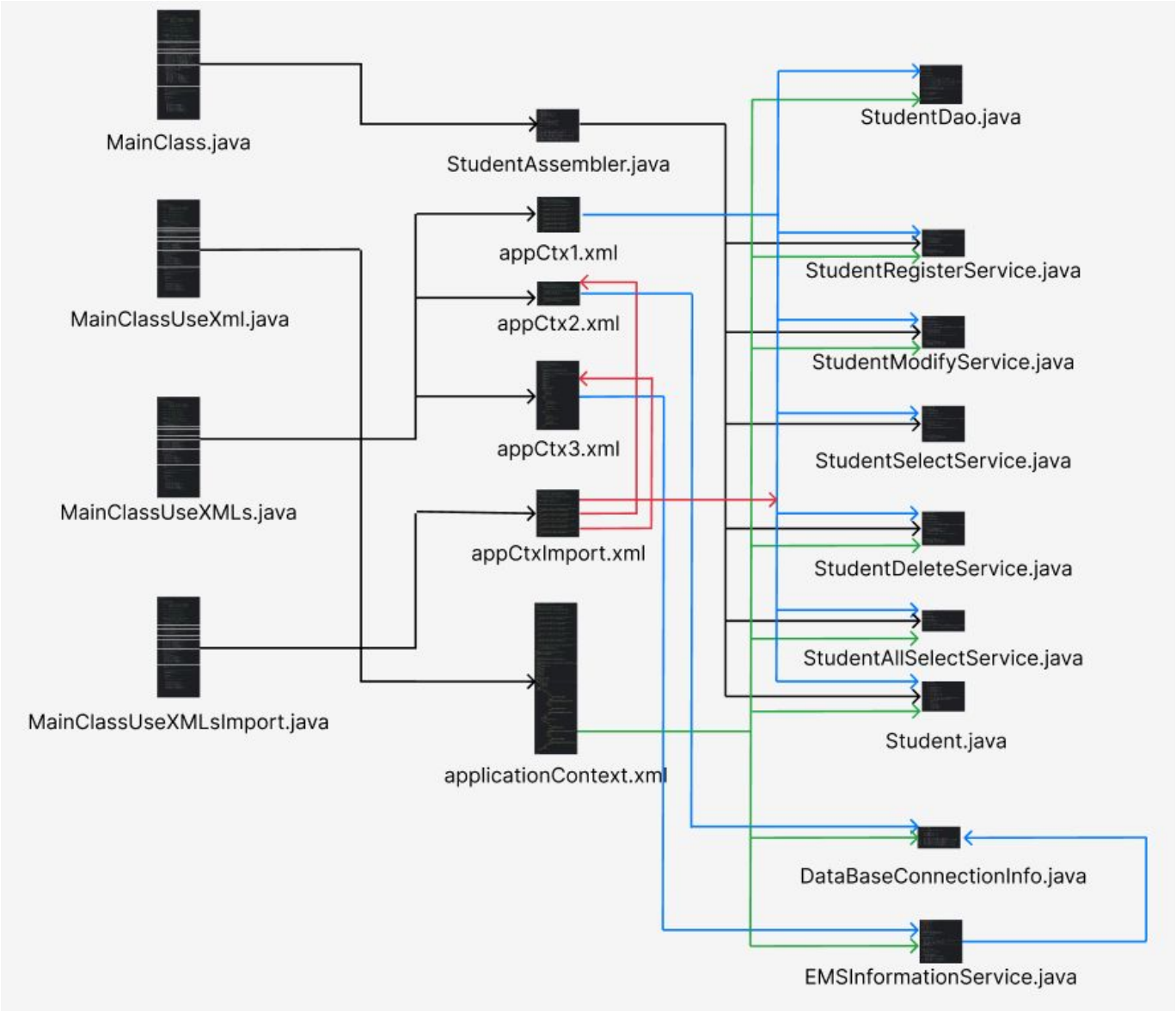


4. MainClassUseXMLsimport



도식도

https://www.figma.com/file/B3jwaF3jF4rNXoZXhXpy25/analyzation?type=design&node-id=29%3A2&mode=design&t=Rpq871vMzj3e9YDA-1



회원 정보 관리 process

1. 회원정보 등록

```java
public class MainClass {

    public static void main(String[] args) {

        String[] sNums = {"H39r8djakndfae32", "H39asdfaelu42o23", "H39iiemamca8w9h4",
                          "H39lkmn754fghia7", "H39plo865cuy8k92", "H39mnbviiaed89q1",
                          "H399omjjyv56t3d5", "H39lczaqwg644gj8", "H39ymbcsh74thgh2",
                          "H39lesvj7544vf89"};

        String[] sIds = {"rabbit", "hippo", "raccoon", "elephant", "lion",
                         "tiger", "pig", "horse", "bird", "deer"};

        String[] sPws = {"96539", "94875", "15284", "48765", "28661",
                         "60915", "30028", "29801", "28645", "28465"};

        String[] sNames = {"agatha", "barbara", "chris", "doris", "elva",
                           "fiona", "holly", "jasmin", "lena", "melissa"};

        int[] sAges = {19, 22, 20, 27, 19, 21, 19, 25, 22, 24};
        String[] sGenders = {"M", "W", "W", "M", "M", "M", "W", "M", "W", "W"};
        String[] sMajors = {"English Literature", "Korean Language and Literature",
                "French Language and Literature", "Philosophy", "History",
                "Law", "Statistics", "Computer", "Economics", "Public Administration"};
```

- DB 등록 대신 main에 회원 정보 등록

2. Student.java로 정보 받기

```java
package ems.member;

public class Student {

    private String sNum;
    private String sId;
    private String sPw;
    private String sName;
    private int sAge;
    private String sGender;
    private String sMajor;

    public Student(String sNum, String sId, String sPw, String sName,
            int sAge, String sGender, String sMajor) {
        this.sNum = sNum;
        this.sId = sId;
        this.sPw = sPw;
        this.sName = sName;
        this.sAge = sAge;
        this.sGender = sGender;
        this.sMajor = sMajor;

    }

    public String getsNum() {
        return sNum;
    }

    public void setsNum(String sNum) {
        this.sNum = sNum;
    }
```

- Main에 등록된 회원 정보를 받는 class 생성

3. 회원정보 검색 (4가지 실행방법)

3-1. [MainClass]
- 메소드 호출

참조: StudentAssembler.java, StudentRegisterService.java, StudentModifyService.java, StudentSelectService.java, StudentAllSelectService.java

3-2. [MainClassUseXML]
- 스프링 **bean** 사용(하나의 **xml**에 **beans** 생성)

참조: applicationContext.xml

3-3. [MainClassUseXMLs]
- **beans** 생성 **xml** 분리

참조: classpath:appCtx1.xml", "classpath:appCtx2.xml", "classpath:appCtx3.xml

3-4. [MainClassUseXMLsimport]
- **3-3**에서 분리한 **xml**을 **import**하여 사용

참조: classpath:appCtxImport.xml

4. 정보 입력 받기(scanner)

```java
while(true) {
    Scanner scanner = new Scanner(System.in);
    String str = "";

    System.out.println("\n=========================================================="
            + "==========================================");
    System.out.println("Select number.");
    System.out.println("1. Check student information");
    System.out.println("2. Exit");

    str = scanner.next();
    if(str.equals("2")) {
        System.out.println("Bye~~");
        break;
    } else {
        System.out.println("Please input your class number.");
```

CHAPTER 2

PAGE DETAILS

Code Analysis

MainClass

1. MainClass 회원정보 등록

2. 서비스를 불러오는 객체 생성

- StudentAssembler로 서비스 객체 생성, 연결
- 객체 생성시 해당 객체의 메소드를 불러옴

```
package ems.member.assembler;

import ems.member.dao.StudentDao;

public class StudentAssembler {

    private StudentDao studentDao;
    private StudentRegisterService registerService;
    private StudentModifyService modifyService;
    private StudentDeleteService deleteService;
    private StudentSelectService selectService;
    private StudentAllSelectService allSelectService;
```

3-1. 학생 정보 저장

- Main에 등록된 학생 정보를 Student에 저장

MainClass

```
StudentRegisterService registerService = assembler.getRegisterService();
for (int j = 0; j < sNums.length; j++) {
    Student student = new Student(sNums[j], sIds[j], sPws[j], sNames[j],
            sAges[j], sGenders[j], sMajors[j]);
    registerService.register(student);
}
```

Student

```
package ems.member;

public class Student {

    private String sNum;
    private String sId;
    private String sPw;
    private String sName;
    private int sAge;
    private String sGender;
    private String sMajor;
```

입력정보	타입	객체이름
번호	String	sNum
아이디	String	sId
비밀번호	String	sPw
이름	int	sName
나이	String	sAge
성별	String	sGender
과목	String	sMajor

3-2. 입력받은 학생 번호 검색

- Main에서 입력한 학생 정보 Student에 저장

StudentRegisterService - register(Student student)

```
public StudentRegisterService(StudentDao studentDao) {
    this.studentDao = studentDao;
}

public void register(Student student) {
    String sNum = student.getsNum();
    if(verify(student.getsNum())) {
        studentDao.insert(student);
    } else {
        System.out.println("The student has already registered.");
    }
}
```

StudentDao – insert(Student student)

```
public void insert(Student student) {
    studentDB.put(student.getsNum(), student);
}
```

4. 회원정보 수정

- 매개변수로 입력한 회원정보를 StudentModifyService에서 검증하여 존재하는 번호인 경우 Student에 저장된 정보 수정

StudentModifyService - register(Student student)

```
StudentModifyService modifyService = assembler.getModifyService();
modifyService.modify(new Student("H39lesvj7544vf89", "deer", "00000", "melissa",
        26, "W", "Vocal Music"));
```

Student - public Student(String sNum, String sId, String sPw, String sName, int sAge, String sGender, String sMajor)

```
public Student(String sNum, String sId, String sPw, String sName,
        int sAge, String sGender, String sMajor) {
    this.sNum = sNum;
    this.sId = sId;
    this.sPw = sPw;
    this.sName = sName;
    this.sAge = sAge;
    this.sGender = sGender;
    this.sMajor = sMajor;
}
```

5. 입력받은 학생 정보 출력

- StudentSelectService에서 존재하는 번호인지 검증 후 Map에 번호 및 Student 객체 저장
- modifiedStudent에서 번호에 맞는 정보 출력

Main – 번호저장 및 print

```
StudentSelectService selectService = assembler.getSelectService();
Student modifiedStudent = selectService.select("H39lesvj7544vf89");
System.out.print("sNum:" + modifiedStudent.getsNum() + "\t");
System.out.print("|sId:" + modifiedStudent.getsId() + "\t");
System.out.print("|sPw:" + modifiedStudent.getsPw() + "\t");
System.out.print("|sName:" + modifiedStudent.getsName() + "\t");
System.out.print("|sAge:" + modifiedStudent.getsAge() + "\t");
System.out.print("|sGender:" + modifiedStudent.getsGender() + "\t");
System.out.print("|sMajor:" + modifiedStudent.getsMajor() + "\n\n");
```

StudentSelectService - select(String sNum)

```
public Student select(String sNum) {
    if(verify(sNum)) {
        return studentDao.select(sNum);
    } else {
        System.out.println("Student information is not available.");
    }

    return null;
}
```

StudentDao - select(String sNum), Map<String, Student>

```
private Map<String, Student> studentDB = new HashMap<String, Student>();

public Student select(String sNum) {
    return studentDB.get(sNum);
}
```

6. 전체 학생 정보 출력

- 전체 학생 번호를 받아 Map에 저장하여 iterator를 통해 정보 출력

Main – 전체 정보 출력

```
StudentAllSelectService allSelectService = assembler.getAllSelectService();
Map<String, Student> allStudent = allSelectService.allSelect();
Set<String> keys = allStudent.keySet();
Iterator<String> iterator = keys.iterator();

while (iterator.hasNext()) {
    String key = iterator.next();
    Student student = allStudent.get(key);
    System.out.print("sNum:" + student.getsNum() + "\t");
    System.out.print("|sId:" + student.getsId() + "\t");
    System.out.print("|sPw:" + student.getsPw() + "\t");
    System.out.print("|sName:" + student.getsName() + "\t");
    System.out.print("|sAge:" + student.getsAge() + "\t");
    System.out.print("|sGender:" + student.getsGender() + "\t");
    System.out.println("|sMajor:" + student.getsMajor() + "\t");
```

StudentAllSelectService

```
public Map<String, Student> allSelect() {
    return studentDao.getStudentDB();
}
```

StudentDAO

```
public Map<String, Student> getStudentDB() {
    return studentDB;
}
```

EMSInformationService

1. 변수 생성

- 프로그램에 대한 정보를 담고 있는 변수
- 학교에 대한 정보를 담고 있는 변수
- 데이터베이스에 대한 정보를 담고있는 변수

```
private String info;  3 usages
private String copyRight;  3 usages
private String ver;  3 usages
private int sYear;  3 usages
private int sMonth;  3 usages
private int sDay;  3 usages
private int eYear;  3 usages
private int eMonth;  3 usages
private int eDay;  3 usages
private List<String> developers;  3 usages
private Map<String, String> administrators;  3 usages
private Map<String, DataBaseConnectionInfo> dbInfos;  4 usages
```

2. 학교와 프로그램 정보 출력

- 프로그램과 학교에 대한 정보를 출력한다

```
public void outputEMSInformation(){  3 usages
    System.out.print("\n\n");
    String devPeriod = sYear + "/" + sMonth + "/" + sDay + " ~ " + eYear + "/" + eMonth + "/" + eDay;
    System.out.println(info + "(" + devPeriod + ")" + "\n" + copyRight + "\n" + ver);
    System.out.println("Developers : " + developers);
    System.out.println("Administrator : " + administrators);
    outputDataBaseInfo();
    System.out.print("\n\n");
}
```

3. DB 정보 출력

- Map에 들어있는 DB의 URL과 Id, Pwd를 출력한다

```
private void outputDataBaseInfo() {  1 usage
    Set<String> keys = dbInfos.keySet();
    Iterator<String> iterator = keys.iterator();

    while (iterator.hasNext()) {
        String key = iterator.next();
        DataBaseConnectionInfo info = dbInfos.get(key);
        System.out.println("[" + key + "]");
        System.out.print("jdbcUrl:" + info.getJdbcUrl() + "\t");
        System.out.print("userId:" + info.getUserId() + "\t");
        System.out.print("userPw:" + info.getUserPw() + "\n");
    }
}
```

StudentAllSelectService

1. StudentDao 객체 생성

- StudentDao 객체를 생성 후 기본 생성자에서 초기화함으로써 해당 클래스의 메소드들이 사용할 수 있게 하고있다

```
private StudentDao studentDao;  3 usages

public StudentModifyService(StudentDao studentDao) { this.studentDao = studentDao; }
```

2. 학생 정보 전체 목록 제공

- StudentDao의 getStudentDB메소드를 호출하여 학생 정보의 전체 목록을 제공한다.

```
public Map<String, Student> allSelect() { return studentDao.getStudentDB(); }
```

StudentDeleteService

1. StudentDao 객체 생성

- StudentDao 객체를 생성 후 기본 생성자에서 초기화함으로써 해당 클래스의 메소드들이 사용할 수 있게 하고있다

```
private StudentDao studentDao;  3 usages

public StudentModifyService(StudentDao studentDao) { this.studentDao = studentDao; }
```

2. 존재하는 학생인지 검색

- 매개변수로 입력받은 학생 번호가 학생 목록에 존재하는지 studentDao 객체의 select 메소드를 이용하여 검색한다
- 검색 결과를 true, false 형식으로 반환한다

```
    public boolean verify(String sNum){  1 usage
        Student student = studentDao.select(sNum);
        return student != null ? true : false;
    }
}
```

3. 학생 정보 삭제

- 매개변수로 입력받은 학생 객체의 학생 번호를 매개변수로 verify 메소드를 호출한다
- 호출 결과가 true일 경우 해당 학생의 정보를 삭제한다.
- 호출 결과가 false일 경우 메시지 출력 후 학생의 정보를 삭제하지 않는다

StudentModifyService

1. StudentDao 객체 생성

- StudentDao 객체를 생성 후 기본 생성자에서 초기화함으로써 해당 클래스의 메소드들이 사용할 수 있게 하고있다

```
private StudentDao studentDao;  3 usages

public StudentModifyService(StudentDao studentDao) { this.studentDao = studentDao; }
```

2. 존재하는 학생인지 검색

- 매개변수로 입력받은 학생 번호가 학생 목록에 존재하는지 studentDao 객체의 select 메소드를 이용하여 검색한다
- 검색 결과를 true, false 형식으로 반환한다

```
    public boolean verify(String sNum){  1 usage
        Student student = studentDao.select(sNum);
        return student != null ? true : false;
    }
}
```

3. 학생 정보 수정

- 매개변수로 입력받은 학생 객체의 학생 번호를 매개변수로 verify 메소드를 호출한다
  - 호출 결과가 true일 경우 해당 학생의 정보를 갱신한다
  - 호출 결과가 false일 경우 메시지 출력 후 학생의 정보를 갱신하지 않는다

```
public void modify(Student student) {  4 usages
    if(verify(student.getsNum())) {
        studentDao.update(student);
    } else {
        System.out.println("Student information is not available.");
    }
}
```

StudentRegisterService

1. StudentDao 객체 생성

- StudentDao 객체를 생성 후 기본 생성자에서 초기화함으로써 해당 클래스의 메소드들이 사용할 수 있게 하고있다

```
private StudentDao studentDao;  3 usages

public StudentModifyService(StudentDao studentDao) { this.studentDao = studentDao; }
```

2. 존재하는 학생인지 검색

- 매개변수로 입력받은 학생 번호가 학생 목록에 존재하는지 studentDao 객체의 select 메소드를 이용하여 검색한다
- 검색 결과를 true, false 형식으로 반환한다

```
    public boolean verify(String sNum){  1 usage
        Student student = studentDao.select(sNum);
        return student != null ? true : false;
    }
}
```

3. 학생 정보 등록

- 매개변수로 입력받은 학생 객체의 학생 번호를 매개변수로 verify 메소드를 호출한다
  - 호출 결과가 true일 경우 해당 학생의 정보를 등록한다
  - 호출 결과가 false일 경우 메시지 출력 후 학생의 정보를 등록하지 않는다

```
public void modify(Student student) {  4 usages
    if(verify(student.getsNum())) {
        studentDao.update(student);
    } else {
        System.out.println("Student information is not available.");
    }
}
```

StudentSelectService

1. StudentDao 객체 생성

- StudentDao 객체를 생성 후 기본 생성자에서 초기화함으로써 해당 클래스의 메소드들이 사용할 수 있게 하고있다

```
private StudentDao studentDao;  3 usages

public StudentModifyService(StudentDao studentDao) { this.studentDao = studentDao; }
```

2. 존재하는 학생인지 검색

- 매개변수로 입력받은 학생 번호가 학생 목록에 존재하는지 studentDao 객체의 select 메소드를 이용하여 검색한다
- 검색 결과를 true, false 형식으로 반환한다

```
    public boolean verify(String sNum){  1 usage
        Student student = studentDao.select(sNum);
        return student != null ? true : false;
    }
}
```

3. 학생 정보 검색

- 매개변수로 입력받은 학생 번호 변수를 매개변수로 verify 메소드를 호출한다
- 호출 결과가 true일 경우 해당 학생의 정보를 검색한다
- 호출 결과가 false일 경우 메시지 출력 후 학생의 정보를 검색하지 않는다

```
public void modify(Student student) {  4 usages
    if(verify(student.getsNum())) {
        studentDao.update(student);
    } else {
        System.out.println("Student information is not available.");
    }
}
```

DatabaseConnectionInfo

1. DB연결 설정하기 위한 정보를 저장

```java
package ems.member;

Explain | Test | Document | Fix | Ask
public class DataBaseConnectionInfo {  9 usages

    private String jdbcUrl;  2 usages
    private String userId;  2 usages
    private String userPw;  2 usages

    public String getJdbcUrl() { return jdbcUrl; }
    public void setJdbcUrl(String jdbcUrl) { this.jdbcUrl = jdbcUrl; }
    public String getUserId() { return userId; }
    public void setUserId(String userId) { this.userId = userId; }
    public String getUserPw() { return userPw; }
    public void setUserPw(String userPw) { this.userPw = userPw; }

}
```

- 해당 클래스를 통해 DB 연결을 관리하는 서비스나
  유틸리티 클래스에서 이 정보를 활용하여 실제 DB 연결을 수립할 수 있다.

Student

1. 학생의 정보를 담은 변수 선언

```
private String sNum;  3 usages
private String sId;  3 usages
private String sPw;  3 usages
private String sName;  3 usages
private int sAge;  3 usages
private String sGender;  3 usages
private String sMajor;  3 usages
```

2. 생성자를 통해 변수 초기화

```
public Student(String sNum, String sId, String sPw, String sName,
        int sAge, String sGender, String sMajor) {
    this.sNum = sNum;
    this.sId = sId;
    this.sPw = sPw;
    this.sName = sName;
    this.sAge = sAge;
    this.sGender = sGender;
    this.sMajor = sMajor;

}
```

3. getter와 setter 사용

```
public String getsNum() { return sNum; }

public void setsNum(String sNum) { this.sNum = sNum; }

public String getsId() { return sId; }

public void setsId(String sId) { this.sId = sId; }

public String getsPw() { return sPw; }

public void setsPw(String sPw) { this.sPw = sPw; }

public String getsName() { return sName; }

public void setsName(String sName) { this.sName = sName; }

public int getsAge() { return sAge; }

public void setsAge(int sAge) { this.sAge = sAge; }

public String getsGender() { return sGender; }

public void setsGender(String sGender) { this.sGender = sGender; }

public String getsMajor() { return sMajor; }

public void setsMajor(String sMajor) { this.sMajor = sMajor; }
```

MainClassUseXML

1. MainClass 회원정보 등록

2. beans 객체 불러오기

- applicationContext의 beans 객체 불러오기

3. applicationContext beans 생성

- bean 객체 불러오기

Class Name	
StudentDao	StudentSelectService
StudentRegisterService	StudentAllSelectService
StudentModifyService	DataBaseConnectionInfo
StudentDeleteService	EMSInformationService

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

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

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


    <bean id="registerService" class="ems.member.service.StudentRegisterService">
        <constructor-arg ref="studentDao" ></constructor-arg>
    </bean>

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

MainClassUseXMLs

1. MainClass 회원정보 등록

2. beans 객체 불러오기

- appCtx1, 2, 3으로 나뉘어진 beans 객체 불러오기

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

EMSInformationService informationService = ctx.getBean("informationService", EMSInformationService.class);
informationService.outputEMSInformation();
```

3-1. appCtx1 beans 생성

- bean 객체 불러오기

Class Name	
StudentDao	StudentDeleteService
StudentRegisterService	StudentSelectService
StudentModifyService	StudentAllSelectService

appCtx1

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

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

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


    <bean id="registerService" class="ems.member.service.StudentRegisterService">
        <constructor-arg ref="studentDao" ></constructor-arg>
    </bean>

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

    <bean id="deleteService" class="ems.member.service.StudentDeleteService">
```

3-2. appCtx2 beans 생성

- bean 객체 불러오기

Class Name
DataBaseConnectionInfo

- jdbc 유저 정보 주입

appCtx2

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

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="dataBaseConnectionInfoDev" class="ems.member.DataBaseConnectionInfo">
        <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe" />
        <property name="userId" value="scott" />
        <property name="userPw" value="tiger" />
    </bean>

    <bean id="dataBaseConnectionInfoReal" class="ems.member.DataBaseConnectionInfo">
        <property name="jdbcUrl" value="jdbc:oracle:thin:@192.168.0.1:1521:xe" />
        <property name="userId" value="masterid" />
```

3-3. appCtx3 beans 생성

- bean 객체 불러오기

Class Name
EMSInformationService

- 정보 주입

```xml
<bean id="informationService" class="ems.member.service.EMSInformationService">
    <property name="info">
        <value>Education Management System program was developed in 2015.</value>
    </property>
    <property name="copyRight">
        <value>COPYRIGHT(C) 2015 EMS CO., LTD. ALL RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.</value>
    </property>
    <property name="ver">
        <value>The version is 1.0</value>
    </property>
    <property name="sYear">
        <value>2015</value>
    </property>
    <property name="sMonth">
        <value>1</value>
    </property>
```

MainClass와 동일

4. 학생 정보 저장

5. 회원정보 수정

6. 입력받은 학생 정보 출력

MainClassUseXML

※ 자바 코드와 서비스 클래스들은 MainClass 와 동일하게 사용했기 때문에 상세설명은 생략합니다.

[MainClass] 와의 차이점 : 순수 자바코드로 객체를 생성하여 사용하던 방식에서 스프링 Bean 을 사용하여 객체를 생성하고, 의존성을 주입했다.

```
//      StudentAssembler assembler = new StudentAssembler();
        // 이번에는 순수자바코드 대신 스프링 bean 으로 객체를 생성해보자 ..!!
        ClassPathXmlApplicationContext ctx =
                new ClassPathXmlApplicationContext("classpath:applicationContext.xml");
```

1. **[applicationContext.xml] 파일 로딩 (이하 'appContext파일')**

   이 단계에서 appContext 파일에 등록된 Bean 이 생성된다. (lazy-init, prototype 설정이 없기 때문입니다)

applicationContext.xml

특징 : **DAO(및 service), DB, EMSinformation 기능을 가진 클래스들의 Bean 을 생성하고, 주입한다.**

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

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!-- StudentDao 클래스 객체(빈) 등록 -->
    <bean id="studentDao" class="ems.member.dao.StudentDao" ></bean>

    <!-- register, modify, delete, select, allselect 클래스에 위에서 생성한 studentDAO 빈을 생성자로 주입 -->
    <bean id="registerService" class="ems.member.service.StudentRegisterService">
        <constructor-arg ref="studentDao" ></constructor-arg>
    </bean>

    <bean id="modifyService" class="ems.member.service.StudentModifySe
        <constructor-arg ref="studentDao" ></constructor-arg>
    </bean>

    <bean id="deleteService" class="ems.member.service.StudentDeleteSe
        <constructor-arg ref="studentDao" ></constructor-arg>
    </bean>

    <bean id="selectService" class="ems.member.service.StudentSelectSe
        <constructor-arg ref="studentDao" ></constructor-arg>
    </bean>

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

```java
public class StudentRegisterService {

    private StudentDao studentDao;

    public StudentRegisterService(StudentDao studentDao) {
        this.studentDao = studentDao;
    }
```

2. DAO 및 Service 기능의 클래스들 생성 및 주입

2-1. StudentDao 클래스 및 service 패키지의 클래스들의 Bean 을 등록한다.

2-1. service 클래스들에 studentDao 클래스를 생성자로 주입한다.

applicationContext.xml

```xml
<!-- DB 연결 정보 (url, id, pw) 를 setter 방식으로 주입 -->
<bean id="dataBaseConnectionInfoDev" class="ems.member.DataBaseConnectionInfo">
    <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe" />
    <property name="userId" value="scott" />
    <property name="userPw" value="tiger" />
</bean>

<bean id="dataBaseConnectionInfoReal" class="ems.member.DataBaseConnectionInfo">
    <property name="jdbcUrl" value="jdbc:oracle:thin:@192.168.0.1:1521:xe" />
    <property name="userId" value="masterid" />
    <property name="userPw" value="masterpw" />
</bean>
```

3. DB connection 생성, 연결 정보 주입

3-1. setter 방식으로 정보를 주입하고 있다. (name값이 필드명과 동일해야한다)

3-2. 디비 연결에 두 개를 등록한 이유는 'EMSInformationService' 클래스에서 디비 정보가 Map 자료형으로 되어있어서 여러 개를 등록한 것으로 보인다.

(해당 프로젝트에서는 실제로 DB연결을 하지 않기에 정확한 확인이 불가)

```java
package ems.member;

public class DataBaseConnectionInfo {

    private String jdbcUrl;
    private String userId;
    private String userPw;

    public String getJdbcUrl() {
        return jdbcUrl;
    }
    public void setJdbcUrl(String jdbcUrl) {
        this.jdbcUrl = jdbcUrl;
    }
    public String getUserId() {
        return userId;
    }
    public void setUserId(String userId) {
        this.userId = userId;
    }
    public String getUserPw() {
        return userPw;
    }
    public void setUserPw(String userPw) {
        this.userPw = userPw;
    }

}
```

applicationContext.xml

```xml
<!-- EMSInformationService 에 setter 방식으로 주입 -->
<bean id="informationService" class="ems.member.service.EMSInformationService">
    <property name="info">
        <value>Education Management System program was developed in 2015.</value>
    </property>
    <property name="copyRight">
        <value>COPYRIGHT(C) 2015 EMS CO., LTD. ALL RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.</value>
    </property>
    <property name="ver">
        <value>The version is 1.0</value>
    </property>
    <property name="sYear">
        <value>2015</value>
    </property>
    <property name="sMonth">
        <value>1</value>
    </property>
    <property name="sDay">
        <value>1</value>
    </property>
    <property name="eYear" value="2015" />
    <property name="eMonth" value="2" />
    <property name="eDay" value="28" />
    <property name="developers">
        <list>
            <value>Cheney.</value>
            <value>Eloy.</value>
            <value>Jasper.</value>
            <value>Dillon.</value>
            <value>Kian.</value>
        </list>
    </property>
    <property name="administrators">
        <map>
            <entry>
```

(우측 사진으로 이어짐)

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

</beans>
```

4. EMSInformationService 클래스 생성 및 주입

4-1. setter 방식으로 정보를 주입하고 있다.

4-2. <list> 타입, <map> 타입은 각 자료형의 속성에 맞게 key, value 값을 주입한다

appCtx1.xml

특징 : DAO및 service 기능을 가진 클래스들의 Bean 을 생성하고, 주입한다.

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


    <bean id="registerService" class="ems.member.service.StudentRegisterService">
        <constructor-arg ref="studentDao" ></constructor-arg>
    </bean>

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

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

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

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

</beans>
```

1. DAO 및 Service 기능의 클래스들 생성 및 주입

1-1. StudentDao 클래스의 Bean을 등록한다

1-2. service 패키지에 있는 클래스 중 EMSInformationService 를 제외한 다른 클래스들의 Bean을 등록한다

1-3. service 패키지의 Bean의 경우 생성자를 통해 정보를 주입한다.

appCtx2.xml

특징 : DataBaseConnectionInfo 클래스의 Bean을 생성하고 정보를 주입한다

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="dataBaseConnectionInfoDev" class="ems.member.DataBaseConnectionInfo">
        <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe" />
        <property name="userId" value="scott" />
        <property name="userPw" value="tiger" />
    </bean>

    <bean id="dataBaseConnectionInfoReal" class="ems.member.DataBaseConnectionInfo">
        <property name="jdbcUrl" value="jdbc:oracle:thin:@192.168.0.1:1521:xe" />
        <property name="userId" value="masterid" />
        <property name="userPw" value="masterpw" />
    </bean>

</beans>
```

1. DB에 관련된 정보를 저장하는 Bean 등록 및 정보 주입

1-1. DataBaseConnectionInfo 클래스의 Bean 두개를 등록한다

1-2. 각각의 Bean에는 setter를 이용하여 정보를 주입한다

1-3. dataBaseConnectionInfoDev의 경우 개발용 데이터베이스의 정보를 담고 있다

1-4. dataBaseConnectionInfoReal의 경우 실제 서비스 중인 프로그램의 데이터베이스의 정보를 담고 있다

appCtx3.xml

특징 : EMSInformationService 클래스의 Bean을 생성하고 정보를 주입한다

```xml
<bean id="informationService" class="ems.member.service.EMSInformationService">
    <property name="info">
        <value>Education Management System program was developed in 2015.</value>
    </property>
    <property name="copyRight">
        <value>COPYRIGHT(C) 2015 EMS CO., LTD. ALL RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.</value>
    </property>
    <property name="ver">
        <value>The version is 1.0</value>
    </property>
    <property name="sYear">
        <value>2015</value>
    </property>
    <property name="sMonth">
        <value>1</value>
    </property>
    <property name="sDay">
        <value>1</value>
    </property>
    <property name="eYear" value="2015" />
    <property name="eMonth" value="2" />
    <property name="eDay" value="28" />
```

1. Bean을 생성한 뒤 프로그램의 정보를 주입한다

1-1. EMSInformationService 클래스의 Bean을 등록한다

1-2. List와 Map 변수인 developers, administrators, dbInfos를 제외하고 다른 변수들의 정보를 setter를 이용하여 주입하고 있다

appCtx3.xml

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

2. developers와 administrators의 정보를 주입한다

2-1. developers의 경우 List 변수이기 때문에 <list> 태그와 setter를 이용하여 정보를 주입한다

2-2. administrators의 경우 Map 변수이기 때문에 <map> 태그와 setter를 이용하여 정보를 주입한다

appCtx3.xml

```xml
<import resource="appCtx2.xml"/>
```

```xml
<property name="dbInfos">
    <map>
        <entry>
            <key>
                <value>dev</value>
            </key>
            <ref bean="dataBaseConnectionInfoDev"/>
        </entry>
        <entry>
            <key>
                <value>real</value>
            </key>
            <ref bean="dataBaseConnectionInfoReal"/>
        </entry>
    </map>
</property>
```

3. dbInfos의 정보를 주입한다

3-1. dbInfos의 경우 Map 변수이기 때문에 <map> 태그와 setter를 이용하여 정보를 주입한다

3-2. 정보를 주입할 때 ref 태그를 이용하여 appCtx2.xml에서 생성한 Bean을 참조한다

3-3. 이 때 appCtx2.xml에서 생성한 Bean을 참조하기 위해 appCtx2.xml을 import해야 한다

MainClassUseXMLsimport

1. MainClass 회원정보 등록

2. appCtxImport beans 객체 생성

- appCtxImport.xml 파일에서 beans 객체 불러오기

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

EMSInformationService informationService = ctx.getBean( name: "informationService", EMSInformationService.class);
informationService.outputEMSInformation();
```

3-1. 학생 정보 저장

- Main에 등록된 학생 정보를 Student에 저장

MainClass

```
StudentRegisterService registerService = assembler.getRegisterService();
for (int j = 0; j < sNums.length; j++) {
    Student student = new Student(sNums[j], sIds[j], sPws[j], sNames[j],
            sAges[j], sGenders[j], sMajors[j]);
    registerService.register(student);
}
```

Student

```
package ems.member;

public class Student {

    private String sNum;
    private String sId;
    private String sPw;
    private String sName;
    private int sAge;
    private String sGender;
    private String sMajor;
```

입력정보	타입	객체이름
번호	String	sNum
아이디	String	sId
비밀번호	String	sPw
이름	int	sName
나이	String	sAge
성별	String	sGender
과목	String	sMajor

3-2. 입력받은 학생 번호 검색

- Main에서 입력한 학생 정보 Student에 저장

StudentRegisterService - register(Student student)

```
public StudentRegisterService(StudentDao studentDao) {
    this.studentDao = studentDao;
}

public void register(Student student) {
    String sNum = student.getsNum();
    if(verify(student.getsNum())) {
        studentDao.insert(student);
    } else {
        System.out.println("The student has already registered.");
    }
}
```

StudentDao – insert(Student student)

```
public void insert(Student student) {
    studentDB.put(student.getsNum(), student);
}
```

4. 회원정보 수정

- 매개변수로 입력한 회원정보를 StudentModifyService에서 검증하여 존재하는 번호인 경우 Student에 저장된 정보 수정

StudentModifyService - register(Student student)

```
StudentModifyService modifyService = assembler.getModifyService();
modifyService.modify(new Student("H39lesvj7544vf89", "deer", "00000", "melissa",
        26, "W", "Vocal Music"));
```

Student - public Student(String sNum, String sId, String sPw, String sName, int sAge, String sGender, String sMajor)

```
public Student(String sNum, String sId, String sPw, String sName,
        int sAge, String sGender, String sMajor) {
    this.sNum = sNum;
    this.sId = sId;
    this.sPw = sPw;
    this.sName = sName;
    this.sAge = sAge;
    this.sGender = sGender;
    this.sMajor = sMajor;
}
```

5. 입력받은 학생 정보 출력

- StudentSelectService에서 존재하는 번호인지 검증 후 Map에 번호 및 Student 객체 저장
- modifiedStudent에서 번호에 맞는 정보 출력

Main – 번호저장 및 print

```
StudentSelectService selectService = assembler.getSelectService();
Student modifiedStudent = selectService.select("H39lesvj7544vf89");
System.out.print("sNum:" + modifiedStudent.getsNum() + "\t");
System.out.print("|sId:" + modifiedStudent.getsId() + "\t");
System.out.print("|sPw:" + modifiedStudent.getsPw() + "\t");
System.out.print("|sName:" + modifiedStudent.getsName() + "\t");
System.out.print("|sAge:" + modifiedStudent.getsAge() + "\t");
System.out.print("|sGender:" + modifiedStudent.getsGender() + "\t");
System.out.print("|sMajor:" + modifiedStudent.getsMajor() + "\n\n");
```

StudentSelectService - select(String sNum)

```
public Student select(String sNum) {
    if(verify(sNum)) {
        return studentDao.select(sNum);
    } else {
        System.out.println("Student information is not available.");
    }

    return null;
}
```

StudentDao - select(String sNum), Map<String, Student>

```
private Map<String, Student> studentDB = new HashMap<String, Student>();

public Student select(String sNum) {
    return studentDB.get(sNum);
}
```

6. 전체 학생 정보 출력

- 전체 학생 번호를 받아 Map에 저장하여 iterator를 통해 정보 출력

Main – 전체 정보 출력

```
StudentAllSelectService allSelectService = assembler.getAllSelectService();
Map<String, Student> allStudent = allSelectService.allSelect();
Set<String> keys = allStudent.keySet();
Iterator<String> iterator = keys.iterator();

while (iterator.hasNext()) {
    String key = iterator.next();
    Student student = allStudent.get(key);
    System.out.print("sNum:" + student.getsNum() + "\t");
    System.out.print("|sId:" + student.getsId() + "\t");
    System.out.print("|sPw:" + student.getsPw() + "\t");
    System.out.print("|sName:" + student.getsName() + "\t");
    System.out.print("|sAge:" + student.getsAge() + "\t");
    System.out.print("|sGender:" + student.getsGender() + "\t");
    System.out.println("|sMajor:" + student.getsMajor() + "\t");
```

StudentAllSelectService

```
public Map<String, Student> allSelect() {
    return studentDao.getStudentDB();
}
```

StudentDAO

```
public Map<String, Student> getStudentDB() {
    return studentDB;
}
```

appCtxImport.xml

특징 : appCtx2.xml과 appCtx3.xml 파일을 import해서 각각의 xml 파일에서 생성한 Bean들을 사용한다

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

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


<bean id="registerService" class="ems.member.service.StudentRegisterService">
    <constructor-arg ref="studentDao" ></constructor-arg>
</bean>

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

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

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

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

2. DAO 및 Service 기능의 클래스들 생성 및 주입

2-1. StudentDao 클래스의 Bean을 등록한다

2-2. service 패키지에 있는 클래스 중 EMSInformationService 를 제외한 다른 클래스들의 Bean을 등록한다

2-3. service 패키지의 Bean의 경우 생성자를 통해 정보를 주입한다.