

**『SEMA』 차세대 정보시스템 구축**

개발표준정의서(Online)

[AA\_A\_10]

Ver. 1.00

2020.11.09



개 정 이 력

| **버전** | **작성일** | **변경 내용** | **작성자** | **승인자** |
| --- | --- | --- | --- | --- |
| 1.00 | 2020.11.09 | 최초 작성 | 홍기웅 | 윤화병 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[Table of Contents 3](#_Toc55320552)

[1. 문서 개요 7](#_Toc55320553)

[1.1. 개요 7](#_Toc55320554)

[1.1.1. 목적 7](#_Toc55320555)

[1.1.2. 관련 문서 7](#_Toc55320556)

[2. 환경 개발절차 8](#_Toc55320557)

[2.1. 개발 환경 구축 방법 8](#_Toc55320558)

[2.2. 디렉토리 / 표준구조 8](#_Toc55320559)

[2.3. 패키지 표준 8](#_Toc55320560)

[3. Online 개발 개요 10](#_Toc55320561)

[3.1. 서비스 개발의 기본 개념 10](#_Toc55320562)

[3.1.1. 용어정의(Terms) 10](#_Toc55320563)

[3.2. 온라인 서비스 개발 순서 11](#_Toc55320564)

[3.2.1. 개발 순서 상세 11](#_Toc55320565)

[4. Configuraion Portal 13](#_Toc55320566)

[4.1. 서비스 반영 13](#_Toc55320567)

[4.1.1. 용어정의(Terms) 13](#_Toc55320568)

[4.1.2. Class Extractor 14](#_Toc55320569)

[4.1.3. 서비스반영 화면 15](#_Toc55320570)

[4.1.4. 서비스반영 절차 15](#_Toc55320571)

[4.2. 서비스 설정 정보 변경 15](#_Toc55320572)

[4.2.1. 용어정의(Terms) 15](#_Toc55320573)

[4.2.2. 서비스 등록 16](#_Toc55320574)

[4.3. 화면등록 19](#_Toc55320575)

[4.3.1. 화면관리에서 화면추가 19](#_Toc55320576)

[4.3.2. 메뉴에 화면 등록 20](#_Toc55320577)

[4.4. 속성정보 등록 21](#_Toc55320578)

[4.4.1. 표준속성(Standard attribute) 21](#_Toc55320579)

[4.4.2. 확장속성(Extended attribute) 23](#_Toc55320580)

[4.5. 코드등록 – 공통 코드관리 25](#_Toc55320581)

[4.5.1. 공통코드의 유형 26](#_Toc55320582)

[4.5.2. 서브셋 코드(Subset Code) 26](#_Toc55320583)

[4.5.3. 확장코드(Extended Code) 26](#_Toc55320584)

[4.5.4. 외부코드(External Code) 27](#_Toc55320585)

[5. 서비스 개발 28](#_Toc55320586)

[5.1. 어노테이션(Annotation) 종류 28](#_Toc55320587)

[5.1.1. 공통 클래스(Common Class) 어노테이션 29](#_Toc55320588)

[5.1.2. 서비스 오퍼레이션(Service Operation) 어노테이션 31](#_Toc55320589)

[5.1.3. 메소드(Method) 어노테이션 32](#_Toc55320590)

[5.2. 상수(Constant) 32](#_Toc55320591)

[5.2.1. BXM에서 상수 선언 32](#_Toc55320592)

[5.2.2. 일반적인 상수 선언 32](#_Toc55320593)

[5.3. Big Decimal 33](#_Toc55320594)

[5.4. 데이터 컨테이너(Data Container) 34](#_Toc55320595)

[5.4.1. 베이스 컴포넌트(Base Component)별 데이터 컨테이너 34](#_Toc55320596)

[5.4.2. 확장된 데이터 컨테이너 35](#_Toc55320597)

[5.5. 에러메시지 다국어 35](#_Toc55320598)

[5.5.1. 용어정의(Terms) 36](#_Toc55320599)

[5.5.2. 설정작업 (Configuration Items) 36](#_Toc55320600)

[5.6. JSON 메시지 37](#_Toc55320601)

[5.6.1. 객체매퍼(ObjectMapper) 37](#_Toc55320602)

[5.6.2. 직렬화(Serialize) 38](#_Toc55320603)

[5.6.3. 역직렬화(Deserialize) 38](#_Toc55320604)

[5.6.4. 노드(Node) 38](#_Toc55320605)

[5.7. 다중에러메세지 39](#_Toc55320606)

[5.7.1. 예외(Exception) 발생이 명확할 때. 39](#_Toc55320607)

[5.7.2. 에러메시지 개수를 체크하여 예외(Exception) 발생. 39](#_Toc55320608)

[5.7.3. 에러 응답메시지 예 39](#_Toc55320609)

[5.8. 트랜잭션 분리(Transaction Require\_New) 40](#_Toc55320610)

[5.8.1. 트랜잭션 정의 40](#_Toc55320611)

[5.8.2. 사용 사례 40](#_Toc55320612)

[5.8.3. 코딩방법 및 샘플코드 41](#_Toc55320613)

[5.9. 내부연동거래 42](#_Toc55320614)

[5.9.1. 연동거래 42](#_Toc55320615)

[5.9.2. 예제 42](#_Toc55320616)

[5.10. 캐쉬(Cache) 43](#_Toc55320617)

[5.10.1. 캐쉬 종류 43](#_Toc55320618)

[5.10.2. 인스턴스캐쉬 43](#_Toc55320619)

[5.10.3. 트랜잭션캐쉬 48](#_Toc55320620)

[5.11. 서비스 검증 및 테스트 49](#_Toc55320621)

[5.11.1. 서비스입력검증 규칙 설정 49](#_Toc55320622)

[5.11.2. 속성검증확장규칙 51](#_Toc55320623)

[6. Java Code Convention 52](#_Toc55320624)

[7. SW보안약점진단가이드 53](#_Toc55320625)

[7.1. 설계단계 보안항목 53](#_Toc55320626)

[7.1.1. 입력데이터 검증 및 표현 53](#_Toc55320627)

[7.1.2. 보안기능 54](#_Toc55320628)

[7.1.3. 에러처리 54](#_Toc55320629)

[7.1.4. 세션통제 55](#_Toc55320630)

[7.2. 구현단계 SW보안약점 항목 55](#_Toc55320631)

[7.2.1. 입력데이터 검증 및 표현 55](#_Toc55320632)

[7.2.2. 보안기능 56](#_Toc55320633)

[7.2.3. 시간 및 상태 57](#_Toc55320634)

[7.2.4. 에러처리 57](#_Toc55320635)

[7.2.5. 코드오류 57](#_Toc55320636)

[7.2.6. 캡슐화 57](#_Toc55320637)

[7.2.7. API오용 58](#_Toc55320638)

[8. 샘플 프로그램 설명 59](#_Toc55320639)

[8.1. 단건 처리 59](#_Toc55320640)

[8.1.1. 화면 60](#_Toc55320641)

[8.1.2. 서비스 60](#_Toc55320642)

[8.1.3. 베이스 63](#_Toc55320643)

[8.1.4. DSO 66](#_Toc55320644)

[8.1.5. DAO 67](#_Toc55320645)

[8.1.6. 중요 전달 사항 67](#_Toc55320646)

[8.2. 다건 처리 70](#_Toc55320647)

[8.2.1. 화면 71](#_Toc55320648)

[8.2.2. 서비스 71](#_Toc55320649)

[8.2.3. 베이스 75](#_Toc55320650)

[8.2.4. DSO 77](#_Toc55320651)

[8.2.5. DAO 79](#_Toc55320652)

[8.2.6. 중요 전달 사항 80](#_Toc55320653)

[8.3. 다건 조회(페이징) 81](#_Toc55320654)

[8.3.1. 화면 82](#_Toc55320655)

[8.3.2. 서비스 82](#_Toc55320656)

[8.3.3. 베이스 85](#_Toc55320657)

[8.3.4. DSO 88](#_Toc55320658)

[8.3.5. DAO 89](#_Toc55320659)

[8.3.6. 중요 전달 사항 90](#_Toc55320660)

[8.4. 복합 처리 92](#_Toc55320661)

[8.4.1. 화면 92](#_Toc55320662)

[8.4.2. 서비스 93](#_Toc55320663)

[8.4.3. 베이스 99](#_Toc55320664)

[8.4.4. DSO 103](#_Toc55320665)

[8.4.5. DAO 104](#_Toc55320666)

[8.4.6. 중요 전달 사항 105](#_Toc55320667)

# 문서 개요

## 개요

### 목적

본 문서는 SEMA 차세대 정보시스템 구축 프로젝트(이하 ‘프로젝트’라 한다.)의 성공적인 구축을 위하여 SEMA 차세대 표준 프레임워크 기반의 소프트웨어 개발 과정에 참여하는 모든 Online 프로그램 개발자에게 필요한 Online 개발 표준을 전달할 수 있도록 합니다. 이를통해 Online 프로그램 개발자의 개발 효율을 향상시키고 표준화되고 패턴화된 산출물(프로그램 소스)를 통해 향후 운영 효율을 향상시키는 것에 목적이 있습니다.

### 관련 문서

|  |  |
| --- | --- |
| **안내서** | **설 명** |
| BXM Studio 설치 및 사용 가이드 | Online 개발을 위한 IDE 환경인 BXM Studio를 설치하고 사용하는 방법을 설명하고 있는 가이드이다. |
| BXM 온라인서비스 개발 가이드 | SEMA 차세대 표준 개발 프레임웍인 BXM Framework 기반 온라인 서비스를 개발하는 방법을 설명하고 있는 가이드이다. |
| 컨피규레이션 가이드 | 회원사업 영역 개발은 BX-CBP 패키지가 제시하는 개발방법론을 따르고 있다.  CBP 패키지에서 사용하는 각 영역별 환경설정 항목 및 설정 방법을 설명하고 있는 가이드이다. |
| JavaCode\_Conventions | SEMA 차세대 개발 표준인 Java Source를 작성함에 있어 필요한 명명규칙(Naming Rule), 주석규칙(Comment Rule), 문장규칙(Statement Rule)을 설명하고 있는 가이드이다. |
| 소프트웨어\_개발보안\_가이드 | 행정안전부와 한국인터넷진흥원에서 2019.11 발간한 전자정부 SW개발, 운영자를 위한 소프트웨어 개발보안 가이드이다. |

# 환경 개발절차

## 개발 환경 구축 방법

Online 개발을 위한 IDE 환경인 BXM Studio를 설치하고 사용하는 방법을 설명하고 있는 가이드이다. 별도의 파일에서 제공 하기로 한다.

## 디렉토리 / 표준구조

SEMA 차세대 정보시스템 구축 프로젝트의 표준 Package는 ‘kr.or.semis’ 패키지를 시작으로 L2시스템영문명을 사용하여 하위 패키지로 구성하도록 한다.

이때 본 프로젝트에서 신규 구축하는 시스템이 아닌, 기존 패키지를 기반으로 과학기술인공제회의 요건을 적용하는 업무시스템들은 해당 솔루션의 패키지 구성을 참조하여 아래와 같이 구성하도록 한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **번호** | 시스템명 | Package명 | 패키지 솔루션명 |
| **-** | **차세대표준** | **kr.or.semis.*{L2시스템영문명}*** | *In House 개발 차세대 표준* |
| 1 | 퇴직연금시스템 | bankware.corebanking.retirementpension | BX-CBP |
| 2 | 공제시스템 | bankware.corebanking.mutualaid |
| 3 | 회원복지시스템 | bankware.corebanking.memberwelfare |
| 4 | 통합회원관리시스템 | bankware.corebanking.*{업무팀 확인 중}* |
| 5 | 공통업무관리 | bankware.corebanking.*{업무팀 확인 중}* |
| 6 | 통계정보시스템 | bankware.corebanking.*{업무팀 확인 중}* |
| 7 | 투자자산관리시스템 | *{업무팀 확인 중}* | S-NOA |
| 8 | 리스크관리시스템 | *{업무팀 확인 중}* |
| 9 | 예산회계시스템 | com.erp.am | BX-ERP |
| 10 | 인사급여시스템 | com.erp.hr |

## 패키지 표준

SEMA 차세대 정보시스템 구축 프로젝트의 표준 Package는 ‘kr.or.semis’ 패키지를 시작으로 L2시스템영문명을 사용하여 하위 패키지로 구성하도록 한다.

|  |  |  |  |
| --- | --- | --- | --- |
| **번호** | Package Level | Package Name | 비고 |
| 1 | 1 | bankware | BX-CBP Pakcage 표준 (패키지 Root) |
| 2 | 2 | corebanking | BX-CBP Pakcage 표준 (계정계) |
| 3 | 2 | ***…*** | *L2 시스템* |
| 4 | 3 | ***{L2시스템영문 full name}*** | *L2 시스템* |
| 5 | 4 | bizproc | 공통 서비스(business proxy)를 관리함 |
| 6 | 5 | dto |  |
| 7 | 4 | common | L2 내부 공통 패키지 |
| 8 | 4 | constant | 상수 클래스를 관리 |
| 9 | 4 | enums | ENUM을 관리 |
| 10 | 4 | utility | static utility 클래스를 관리 |
| 11 | 4 | ***…*** | *L3 서브시스템* |
| 12 | 4 | ***{L3서브시스템영문 full name}*** | *L3 서브시스템* |
| 13 | 5 | batch | BATCH 프로그램 클래스를 관리 |
| 14 | 6 | dto |  |
| 15 | 5 | business | Interfaces 구현체와 같은 BO 클래스를 관리 |
| 16 | 6 | dto |  |
| 17 | 5 | dao | DBIO를 관리 |
| 18 | 6 | dto |  |
| 19 | 5 | daobatch | BATCH용 DBIO를 관리 |
| 20 | 6 | dto |  |
| 21 | 5 | dso | DBMS 종속성 및 Data 캐쉬 역할을 담당 |
| 22 | 5 | eai |  |
| 23 | 6 | dto | EAI/FEP용 OMM을 관리함 |
| 24 | 5 | Interfaces | Service BO 의 interface클래스를 관리 |
| 25 | 6 | dto |  |
| 26 | 5 | service | 서비스 클래스를 관리 |
| 27 | 6 | dto |  |

# Online 개발 개요

SEMA 차세대 표준 프레임워크 기반으로 Online 프로그램 개발을 할 때, BXM 프레임워크 및 CBP 설정포털에 대한 지식이 필요하다.

* BXM Framework
* Configuration Portal

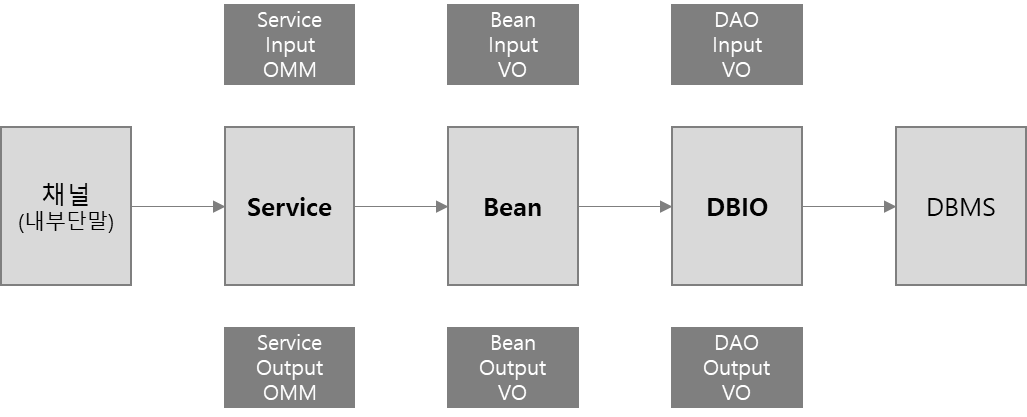
본 문서에서 서비스 개발 시과 관련된 정보를 모두 제공하는 것은 현실적으로 불가능하므로, 서비스 개발에 필요한 기본적인 내용을 설명하고 BXM Framework 및 Configuration Potal 관련된 상세한 사용법은 별도 문서/ URL로 제공하기로 한다.

## 서비스 개발의 기본 개념

### 용어정의(Terms)

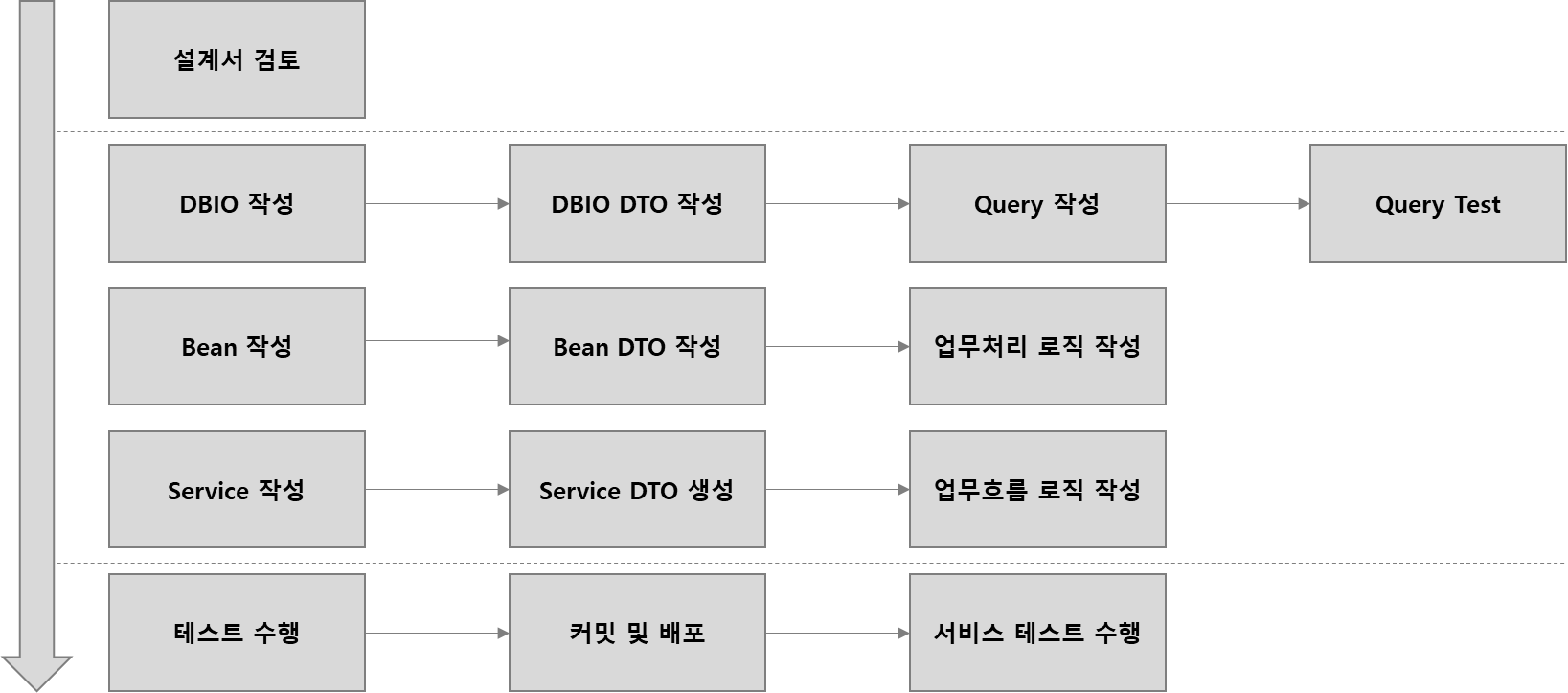
|  |  |  |
| --- | --- | --- |
| **번호** | 용어 | 설 명 |
| 1 | Service | 서비스는 화면/대외계 시스템 등에서 요청된 업무 처리를 수행하기 위한 메소드(오퍼레이션)들을 포함하고 있는 클래스 타입이다. |
| 2 | Bean | Bean은 비즈니스 로직을 구현하고 있는 메소드들을 포함하고 있는 클래스 타입이다. |
| 3 | DBIO | DBIO는 DBMS 데이터에 대한 처리를 수행하는 리소스로 메소드 선언들을 정의하고 있는 자바 인터페이스(.java)와 SQL, 파라미터 바인딩, 결과 맵핑 등에 대한 설정을 정의하고 있는 맵퍼(.dbio)파일로 구성된다 |
| 4 | DTO | DTO는 Data Transfer Object의 약자로, data를 저장 관리하며 전달하는 객체를 의미한다. 목적에 따라 OMM과 VO를 사용한다. |
| 4-1 | OMM | OMM(Object Message Mapping)은 BXM 프레임워크에서 사용하는 표준 데이터 전달 객체로, 서비스에서 사용하는 모든 DTO는 OMM으로 관리한다. |
| 4-2 | VO | VO는 Value Object의 약자로, Base 이하 내부에서 사용한다. |

위의 개념을 그림으로 표현하면 아래와 같다.



## 온라인 서비스 개발 순서

온라인 서비스는 DBIO > Bean > Service 순으로 작성하기를 권고한다.



### 개발 순서 상세

#### 테이블 생성

|  |  |  |
| --- | --- | --- |
| 번호 | 작 업 | 참조 문서 |
| 1 | 모델 생성 | 데이터 표준화 지침서 |
| 2 | 테이블 생성 |  |
| 3 | 메타 등록 | 신규 속성명 필요 시, 메타 사용자 매뉴얼 |
| 3 | 표준속성등록 | 신규 속성명 필요 시, 컨피규레이션 가이드 |
| 4 | 디폴트 DBIO 생성 | BXM 온라인 개발 가이드 |

#### 베이스(Base) 개발

|  |  |  |
| --- | --- | --- |
| 번호 | 작 업 | 참조 문서 |
| 1 | Java 인터페이스 개발 | BXM 온라인 개발 가이드 |
| 2 | BO 클래스 개발 |
| 3 | Manager 클래스 개발 |
| 4 | DBIO 개발 |
| 5 | Constant 정의 |  |
| 6 | Enum 정의 |  |
| 7 | DTO 개발 | BXM 온라인 개발 가이드 |
| 8 | 단위 테스트 |  |

#### 서비스(Service) 개발

##### 화면 개발

|  |  |  |
| --- | --- | --- |
| 번호 | 작 업 | 참조 문서 |
| 1 | 화면등록 | 컨피규레이션 가이드 |
| 2 | Nexacro UI 개발 | 개발표준정의서(UI) |
| 3 | 메뉴 등록 | 컨피규레이션 가이드 |
| 4 | 화면 테스트 |  |

##### 서비스 개발

|  |  |  |
| --- | --- | --- |
| 번호 | 작 업 | 참조 문서 |
| 1 | 화면등록 | 컨피규레이션 가이드 |
| 2 | 서비스 개발 | BXM 온라인 개발 가이드 |
| 3 | 서비스 등록 | 컨피규레이션 가이드 |
| 4 | 서비스 입력 항목 등록 |
| 5 | 서비스 테스트 |  |

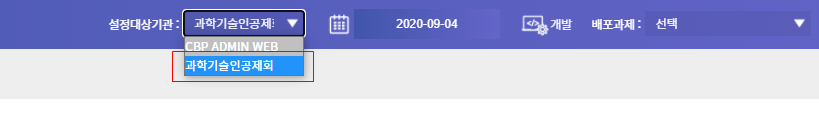
# Configuraion Portal

Onine 프로그램 개발을 위한 최소한의 컨피규레이션 포탈 작업은 다음과 같다.

|  |  |  |
| --- | --- | --- |
| 중분류 | 세분류 | 설 명 |
| 서비스 | 서비스 반영 | 개발한 서비스의 클래스 관련 정보를 CBP 패키지 기반 시스템에 등록하는 작업 |
| 서비스 관리 | 서비스의 클래스 이외의 설정 정보를 CBP 패키지 기반 시스템에 등록하는 작업 |
| 화면 | 화면 관리 | 개발한 화면을 등록하는 작업 |
| 속성 | 표준속성 관리 | 데이터 입출력항목, 테이블 컬럼 등으로 사용할 속성을 등록 |
| 확장속성 관리 | 기관별 다른 요건의 데이터 항목을 정의 |
| 메시지 | 에러메시지 관리 | 사용한 에러메시지를 등록 |
| 코드 | 공통코드 관리 | 사용한 공통코드를 등록 |

컨피규레이션 포탈(Configuration Portal)은 CBP 패키지 기반 시스템이 특정 기관에서 사용될 때 해당 기관의 환경에 맞게 다양한 설정을 하는 환경설정 페이지로 이해할 수 있다.

Configuration Portal에서 설정정보를 변경하려면, 아래와 같이 설정기관 정보를 ‘과학기술인공제회’로 변경해 놓은 상태에서 모든 설정정보를 변경해야 한다. (주의)



## 서비스 반영

서비스(Service)는 내/외부채널이 BX-CBP패키지에 요청하는 업무처리 단위이다. 서비스 제어를 위해서 수행되는 서비스의 기본적인 클래스 정보를 자동으로 반영하기 위하여 ‘서비스반영’ 작업을 수행한다.

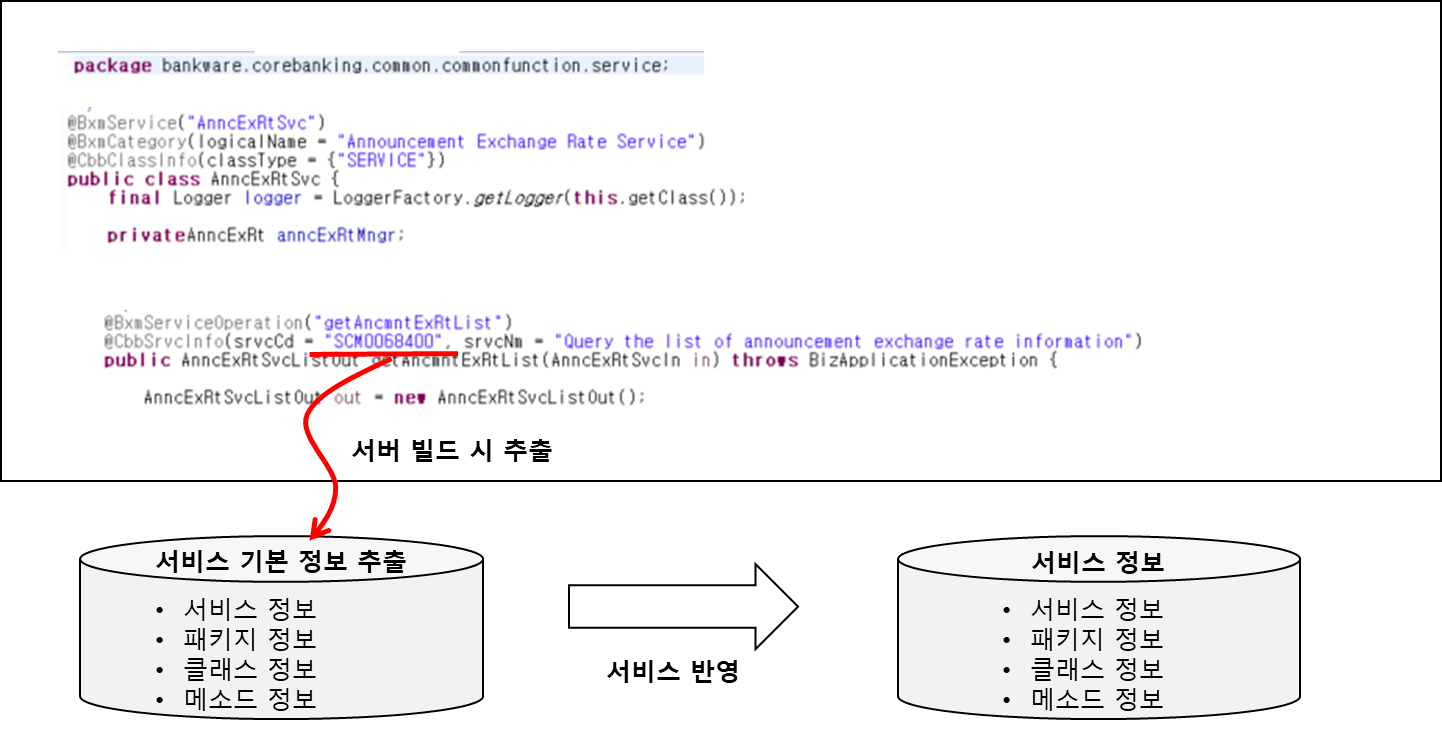
### 용어정의(Terms)

|  |  |
| --- | --- |
| 용어 | 설 명 |
| 현행정보 | 현행 BX-CBP에서 동작하고 있는 서비스의 클래스 관련 정보 |
| 최신정보 | 개발환경에서 Commit한 소스를 바탕으로 자동 추출한 서비스의 클래스 관련 정보 |
| 서비스 입/출력 항목 | 개발환경에서 Commit한 소스를 바탕으로 현재 동작하고 있는 모듈을 비교한 정보를 나타낸다.  속성명을 기준으로 상호 비교한 정보를 ‘변경’ 또는 ‘신규’로 표시한다. |

### Class Extractor

BXM Studio에서 소스 Commit을 하면 BXM프레임워크에서 수행되는 Class Extractor가 Commit 후 Build된 소스를 parsing하여 아래와 같은 정보를 자동 추출하게 된다.

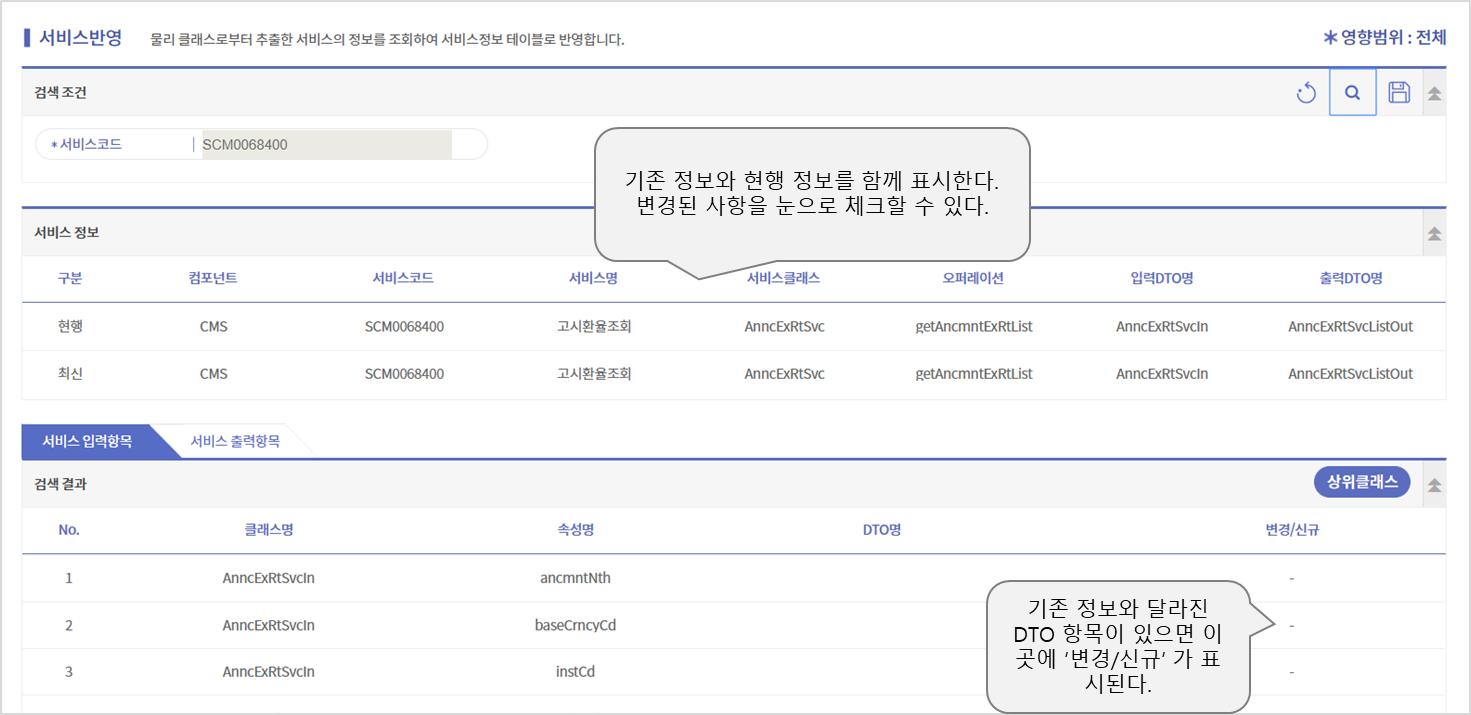
* 컴포넌트코드, 서비스코드, 서비스명, 서비스클래스, 오퍼레이션, 입력DTO명, 출력 DTO명
* 입력 OMM내부의 상세 항목명(속성명), 출력 OMM내부의 상세 항목명(속성명)



추출된 정보는 Configuration Portal의 아래 위치로 자동 저장되며, ‘서비스 반영’을 통해서 BX-CBP 공통영역에서 관리하는 ‘서비스 관리’ 영역으로 정보를 이관하게 된다. 만일 Commit을 수행하여도 아래 테이블에 관련 클래스의 정보가 생성되지 않았다면, Build가 정상적으로 수행되었는지를 점검해야 한다.

|  |  |  |
| --- | --- | --- |
| 순서 | Class Extractor 관련 테이블명 | 테이블 설명 |
| 1 | class\_xtrctn | BX-CBP에서 관리하는 클래스의 기본 정보를 관리 |
| 2 | class\_xtrctn\_mthd | BX-CBP에서 관리하는 클래스에서 구현된 메소드 목록을 관리 |
| 3 | class\_xtrctn\_dto | 서비스클래스에서 사용하는 In/Out OMM의 상세 항목을 관리 |
| 4 | class\_xtrctn\_srvc | 서비스클래스의 서비스 정보를 관리 |

### 서비스반영 화면



### 서비스반영 절차

개발된 새로운 서비스 클래스는 아래와 같은 절차를 이용하여 반영한다.

|  |  |  |
| --- | --- | --- |
| 순서 | 작업 내용 | 비고 |
| 1 | 정상 Build 확인 | Compile 오류없이 Build가 정상적으로 수행되었는지를 확인 |
| 2 | 서비스 클래스 Commit | BXM Studio에서 서비스 클래스를 Commit |
| 3 | CP::서비스 반영 화면 OPEN | 기관을 ‘과학기술인공제회’로 설정 후 진행 |
| 4 | 서비스코드 조회 | 개발한 서비스코드를 조회 |
| 5 | 서비스 정보 변경 사항 확인 |  |
| 6 | 서비스 입출력 항목 변경 사항 확인 | 입/출력 항목들의 ‘신규’, ‘변경’ 여부를 확인한다. |
| 7 | 서비스 반영 수행 |  |

## 서비스 설정 정보 변경

서비스 반영에서는 해당 서비스에서 클래스 관련된 정보만을 등록한다. 그 외에 서비스 제어를 위한 다양한 정보들을 서비스 관리 화면에서 직접 수정한다.

### 용어정의(Terms)

|  |  |
| --- | --- |
| 용어 | 설 명 |
| 서비스 속성 | 채널/거래실행/재무/회계 등 개별서비스가 가지는 속성 |
| 서비스 프로파일 | 개별 서비스에 적용되는 서비스 속성값들의 집합. 표준 프로파일이 개별서비스에 적용되며, 기관별로 프로파일 변경 가능 |
| 서비스 입력값 검증 | 서비스 입력 항목에 대한 입력값 검증 규칙을 설정 |

서비스 호출은 창구단말 호출 거래, 인터넷뱅킹 서비스, 외부기관 인터페이스(개설), 웹서비스(Web Service), 코어뱅킹 외부에 제공하는 원격객체호출 메소드 등이 해당 된다. 관련 테이블은 서비스 기본(sv\_srvc\_m)과, 기관별 서비스 프로파일명세(sv\_inst\_srvc\_prfl\_d)가 있다.

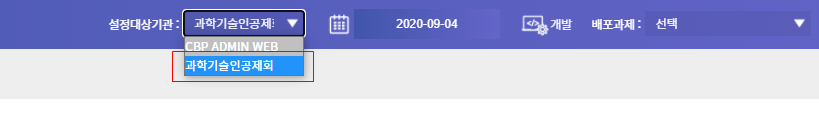
### 서비스 등록

개발된 새로운 서비스는 아래와 같은 절차를 이용하여 등록한다.

|  |  |  |
| --- | --- | --- |
| 순서 | 작업 내용 | 비고 |
| 1 | 서비스 반영 |  |
| 2 | CP::서비스 반영 화면 OPEN | 기관을 ‘과학기술인공제회’로 설정 후 진행 |
| 3 | 서비스 코드 존재여부 확인 | 서비스 관리화면에서 서비스 코드가 정상적으로 등록되었는지 확인 |
| 4 | 서비스 상태 변경 | 신청에서 활동으로 변경 |
| 5 | 기관에 서비스 할당 | 기관에서 ‘서비스 사용여부’를 체크 |
| 6 | 서비스명 다국어 등록 | 서비스를 위한 다국어 등록 |
| 7 | 서비스 프로파일 속성 관리 |  |
| 8 | 기관별 서비스프로파일 관리 |  |
| 9 | 서비스 입력값 검증 규칙 관리 |  |

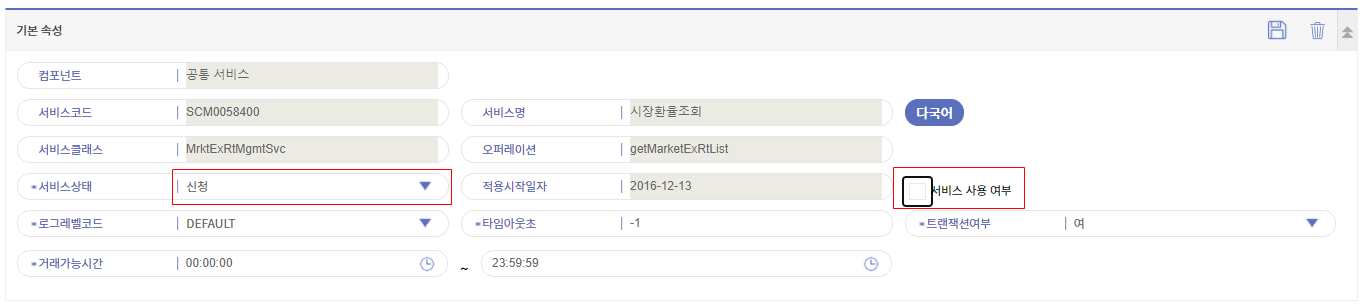
#### 기관코드 변경

Configuration Portal의 설정기관을 ‘과학기술인공제회’로 선택한다. (아래 나오는 모든 설정 정보에



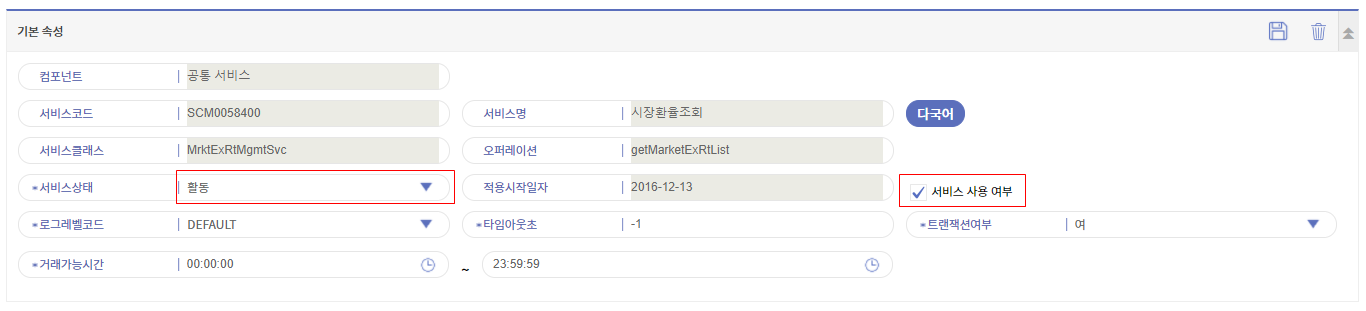
#### 서비스 코드 존재여부 확인

서비스 반영을 한 후 원하는 코드가 정상적으로 DB에 저장되었는지 아래와 같이 서비스 관리화면에서 확인할 수 있다. 서비스 반영을 한 신규 서비스의 “서비스상태코드” 값은 “신청”상태이다.



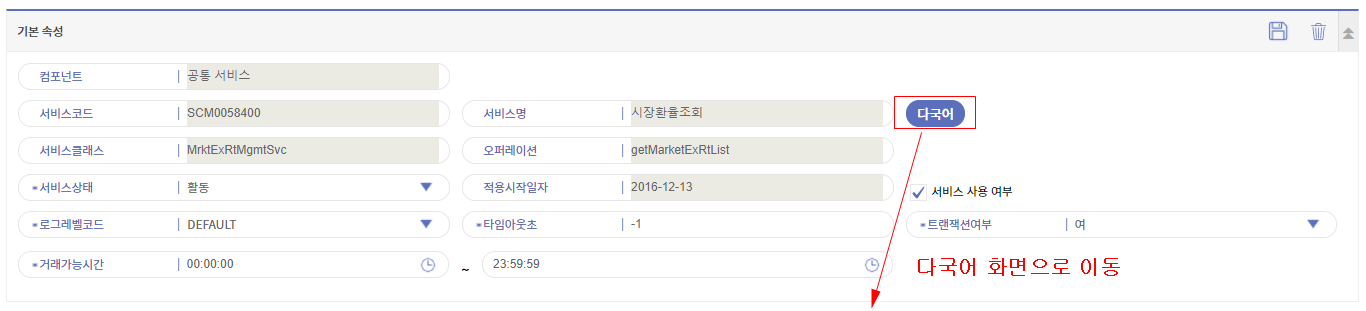
#### ‘서비스 상태’ 및 ‘사용 여부’ 변경

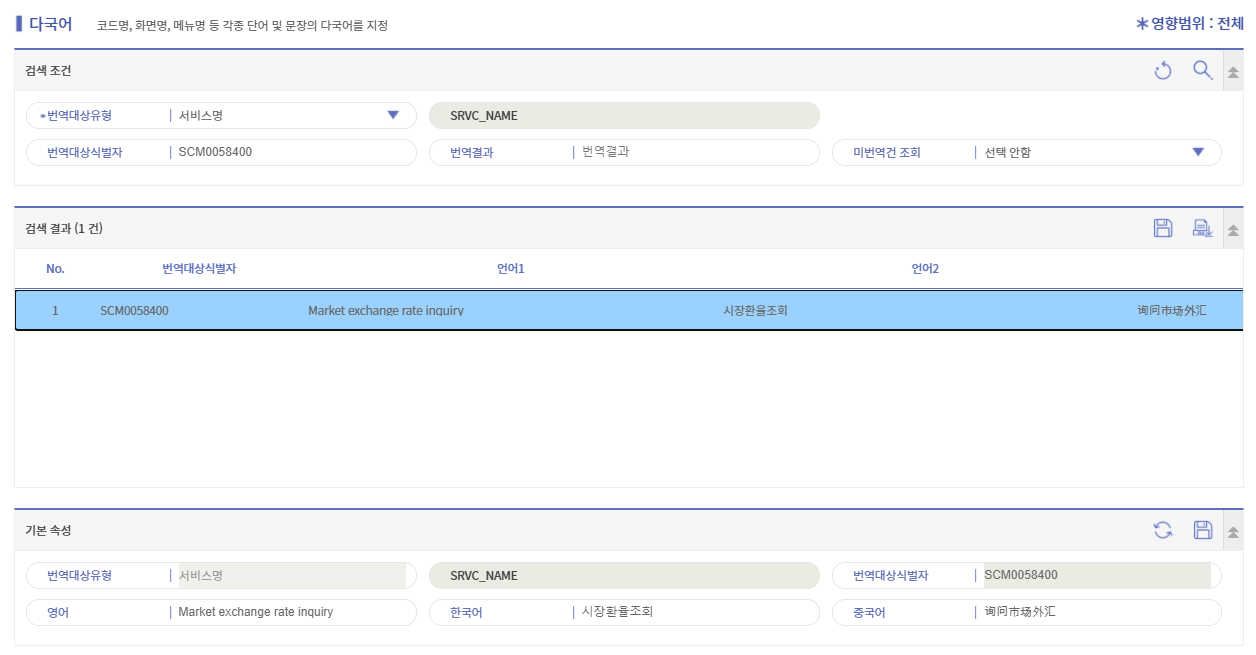
해당 서비스의 상태코드를 “활동”으로 변경한다. 서비스 사용여부를 V 체크 한다.



#### 서비스명 다국어 등록

서비스를 사용하기 위해서는 서비스 명의 다국어를 등록해야 한다.





#### 서비스 프로파일 속성관리

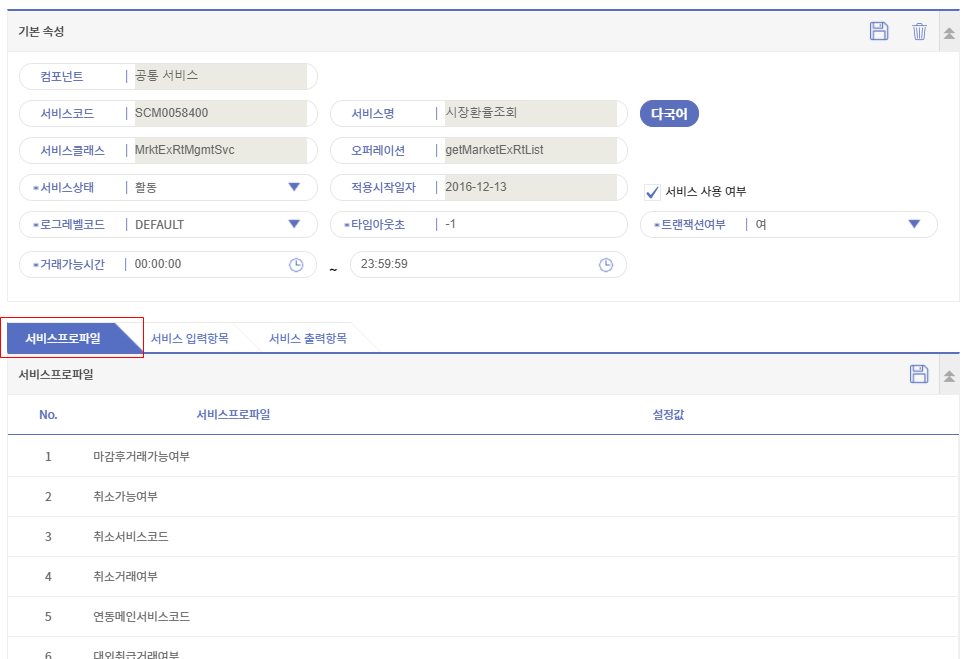
##### 서비스속성

현재 CBP에 구현된 서비스속성은 다음과 같다.

|  |  |  |
| --- | --- | --- |
| 프로파일속성 | | 설명 |
| 마감후거래가능여부 | clsgAfTxAblYn | 당일자 업무마감이후 당일자로 거래가 가능한지 여부임 |
| 재무거래여부 | fnclTxYn | 회계저널이 발생할 수 있는 거래인지 여부임. |
| 휴일거래가능여부 | hldyTxAblYn | 영업일이 아닌 휴일(공휴일 포함) 해당 거래될 수 있는지 여부임 |
| 거래가능종료시간 | txAblEndHms | 거래가능종료시간, HH:MM:SS형태로 지정됨 |
| 거래가능시작시간 | txAblStartHms | 거래가능시작시간, HH:MM:SS형태로 지정됨 |
| 대외취급거래여부 | extrnlAsyncSndTxYn | 해당 서비스에서 CBP 외부로 비동기식 시스템인터페이스가 발생하는지 여부임. 해당 시 ‘Y’로 지정됨. |
| 대외개설거래여부 | extrnlRcvdTxYn | 해당 서비스가 대외채널로 부터 실행되는 개설거래인지 여부임. 해당 시 ‘Y’로 지정됨. |
| 타임아웃거래여부 | timeoutTxYn | 해당 서비스코드가 타임아웃처리용 서비스이면 ‘Y’로 지정함 |
| 취소거래여부 | cnclTxYn | 해당 서비스코드가 취소정정 처리서비스이면 ‘Y’로 지정함 |
| 거래저널생성여부 | txJrnlCrtnYn | 취소정정 또는 대외취급거래의 Timeout 처리, 오류응답으로 인한 취소처리 등을 위해서 필요한 거래저널을 생성할 지의 여부임. 해당 시 ‘Y’로 지정됨. |
| 취소가능여부 | cnclAblYn | 해당 서비스가 취소대상인지 여부임. 해당 시 ‘Y’로 지정됨 |
| 취소서비스코드 | cnclSrvcCd | 해당 거래가 취소되어야 할 때, 해당 거래를 취소할 취소서비스코드 |

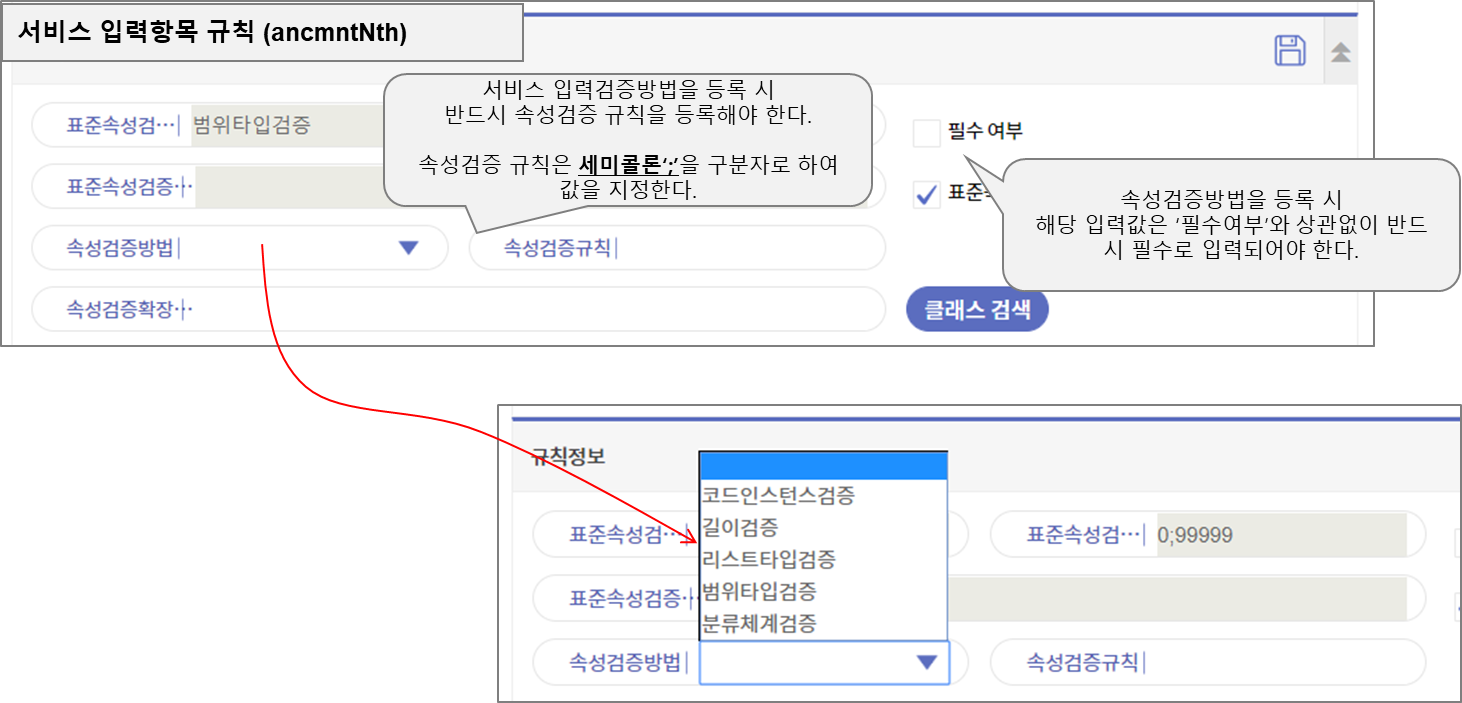
##### 서비스별 프로파일속성

|  |  |
| --- | --- |
| 항 목 | 설 명 |
| 프로파일속성명 | 서비스에 대한 관리속성 컬럼명 |
| 표준프로파일속성값 | 해당 서비스의 해당 프로파일속성에 대한 표준 값 |
| 기관프로파일속성값 | 표준과 동일하면 기술하지 않음  값을 다르게 해야 하는 경우, As-Is의 서비스프로파일 수집정보를 분석하여, CBP 속성값 체계에 맞게 변환하여 작성한다. |



#### 서비스 입력값 검증 규칙 설정

서비스 입력값을 패키지에서 기본으로 점검하기 위한 규칙을 설정한다.



|  |  |
| --- | --- |
| 용어 | 설 명 |
| 표준속성검증여부 사용여부 | 표준속성에서 정의한 검증 규칙을 사용할 것인지를 체크 |
| 표준속성검증방법 | 표준속성에서 등록한 검증방법을 보여줌 |
| 표준속성검증규칙 | 표준속성에서 등록한 검증규칙을 보여줌 |
| 표준속성검증확장규칙 | 표준속성에서 등록한 검증확장규칙을 보여줌 |
| 필수여부 | 서비스 호출 시, 필수값 체크를 해야 할 대상을 지정함 |
| 속성검증방법 | 코드인스턴스, 길이, 리스트, 범위, 분류체계 중에 하나를 선택 |
| 속성검증규칙 | 세미콜론(‘;’)을 구분자로 규칙을 등록 |
| 속성검증확장규칙 | 검증을 수행할 클래스를 지정 |

## 화면등록

### 화면관리에서 화면추가

화면을 추가하기 위해서는 사용자인터페이스 > 화면 > 화면을 이용한다.

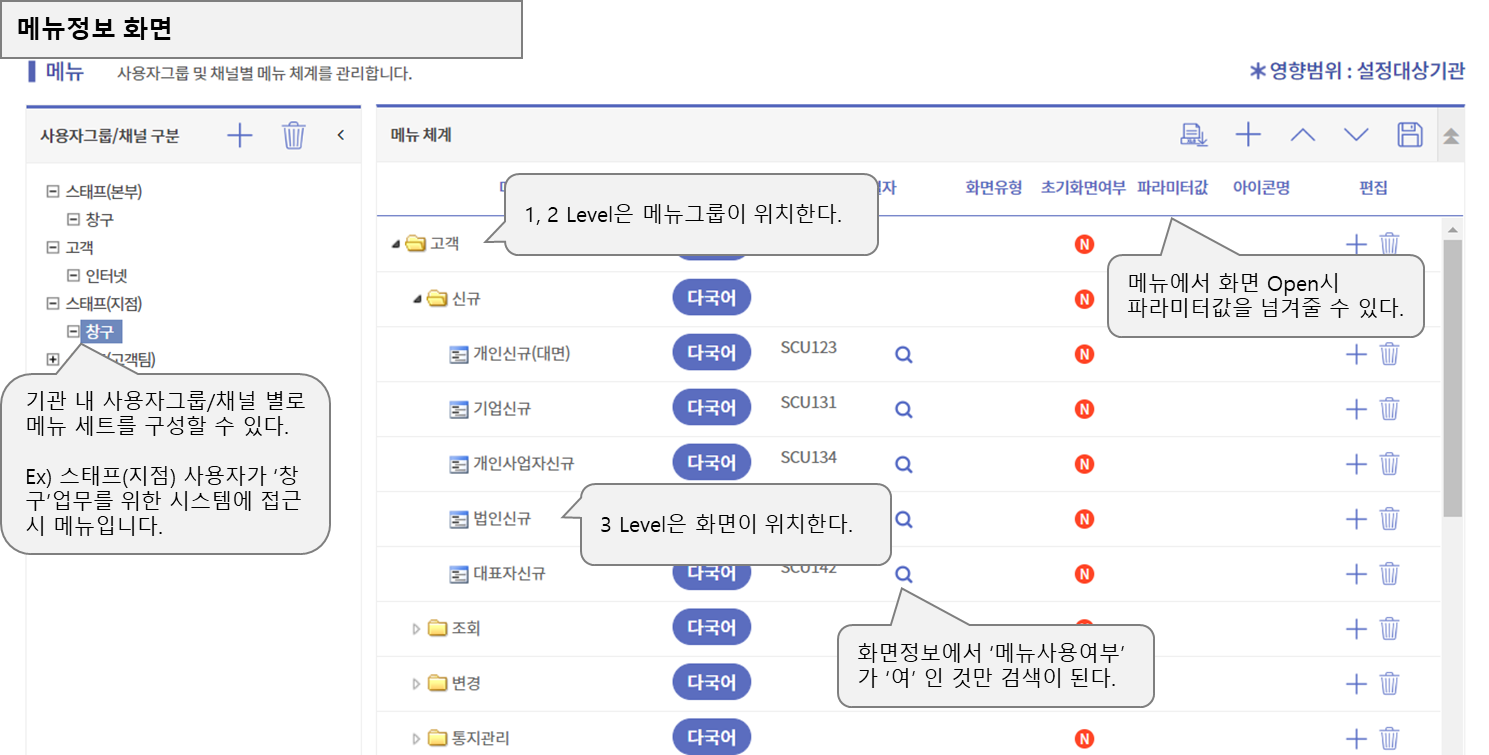


예를 들어, 다음과 같은 정보를 입력한 후, 저장버튼을 선택한다.

|  |  |
| --- | --- |
| 항목 | 값 |
| 컴포넌트 코드 | 화면의 업무 영역을 구분하기 위한 컴포넌트 구분 |
| 화면식별자 | 화면을 관리하기 위한 유니크한 식별자 |
| 화면번호 | 별도 화면번호를 관리하고자 할 때 등록 |
| 화면명 | 화면의 논리적인 명칭 |
| 메뉴대상 화면여부 | 메뉴로 등록되는 화면인지를 구분 |
| 화면 사용 여부 | 화면을 현재 사용하고 있는지를 설정. 사용하는 화면은 반드시 V 체크한다. |
| 화면 URL주소 | UI에서 화면을 로딩하기 위해 필요한 정보 |
| 배포대상화면 | Configuration Portal에서 사용하는 화면 중 배포와 관련된 화면일 때만 체크한다. |
| 팝업화면 여부 | 팝업으로 열리기 위한 화면인지를 구분 |

### 메뉴에 화면 등록

기관 메뉴에 화면을 등록하기 위해서는 사용자인터페이스 > 메뉴> 메뉴를 이용한다.



## 속성정보 등록

### 표준속성(Standard attribute)

표준속성(Attribute)이란 데이터입출력 또는 테이블컬럼으로 이용되는 데이터항목을 의미한다. BX-CBP패키지는 각 표준속성의 속성타입, 길이, 검증방법을 정의하여 입력검증을 수행한다.

|  |  |
| --- | --- |
| 용어 | 설 명 |
| 표준속성 | 데이터입출력항목, 테이블컬럼처럼 특정 데이터도메인(길이, 타입)을 갖는 테이터항목 |
| 테이블확장속성 | 계약/거래/정산/고객 등과 같이 핵심 엔티티이면서, 기관별 데이터항목에 대한 요건이 다를 경우, 이 항목들의 집합을 별도 확장테이블로 정의할 때, 확장테이블의 속성 |
| 표준속성검증규칙 | 표준속성의 값이 유효성을 검증하는 규칙 |
| 데이터입출력항목 | 개별 서비스의 입출력메시지를 구성하는 데이터항목 |

#### 관련 테이블

|  |  |
| --- | --- |
| 용어 | 설 명 |
| 표준속성 | cm\_std\_atr\_m |
| 테이블 속성 | cm\_tbl\_atrbt\_m |
| 테이블확장속성 | cm\_tbl\_xtn\_atr\_m |
| 표준속성검증규칙 | cm\_std\_atr\_vld\_vrtn\_m |

#### 표준속성 추가



“초기화” 버튼으로 화면을 초기화 후, 영문속성명을 작성하고, “속성명생성” 버튼으로 속성명(표준영문단어 약어의 조합)을 생성한다.

영문속성명은 영문단어(Full Name)를 띄어쓰기 형식으로 작성한다. 예)Open Date. 이때 영문단어(Full Name)은 CBP 표준단어사전에 사전에 등록되어 있는 단어이어야 한다. 미등록된 단어는 사용할 수 없다. 속성정보 및 검증규칙 항목을 작성 후 저장버튼을 선택하여 표준속성을 등록한다.

##### 속성타입 설명

|  |  |
| --- | --- |
| 속성타입 | 설 명 |
| 알파벳숫자 | A~Z, 0~9만 허용된 속성임. 특수문자 및 기호는 입력검증오류임 |
| 숫자 | 숫자 속성임. 소숫점 이하 자리수를 가질 수 있으며, Big Decimal 또는 integer에 해당함. |
| 8자리날짜텍스트 | 8자리 숫자인 String속성이며, 날짜형식인지 검증함 |
| 텍스트숫자 | 0~9만 허용된 String속성임. 소수점 이하 자리수를 지정할 수 없음 |
| 텍스트 | 일반 텍스트 속성임. A~Z, 0~9, 특수문자, 기호가 허용됨. |
| 여부 | Y, N만 허용됨 |

#### 속성검증방법 설명 및 검증규칙 작성방법

|  |  |  |  |
| --- | --- | --- | --- |
| 검증방법 | 설 명 | 검증규칙 작성법 | 대상 속성타입 |
| 코드인스턴스검증 | 입력값이 지정된 코드번호의 코드값에 해당하는지 검증함 | 코드번호를 지정함  예)11311 | 텍스트 |
| 길이검증 | 입력값의 길이가 최대값, 최소값 범위 내인지 검증함 | 최소길이;최대길이  예)1;10 | 알파벳숫자, 텍스트숫자, 텍스트 |
| 리스트검증 | 입력값이 가능한 값의 목록에 포함되어 있는지 검증함 | 예)AT;AR;CM | 알파벳숫자, 텍스트숫자, 텍스트, 8자리날짜텍스트 |
| 범위검증 | 입력값이 최소~최대값 범위내에 있는지 검증함 | 예)2014010;99991231 | 알파벳숫자, 텍스트숫자, 텍스트, 8자리날짜텍스트 |

|  |  |
| --- | --- |
| 항 목 | 설 명 |
| 속성길이 | 해당 속성이 갖는 길이정보. 속성타입이 알파벳숫자, 숫자, 텍스트숫자, 텍스트인 경우에만 길이를 작성하고, 그 외에는 작성하지 않는다.  해당 속성이 부모속성을 갖는 경우에는 부모속성의 정보에 따르기 때문에 작성하지 않는다. |
| 소수점 자리수 | 속성타입이 숫자인 경우에만 작성한다 |
| 속성검증 확장규칙 | 앞 표에서 언급된 검증규칙방식으로 지원되지 않는 특화된 검증규칙에 대해서는 검증용 클래스를 개발하고, 본 화면에서 그 클래스명을 지정하여, 런타임 시에 검증에 활용되게 된다.  작성방법: simple(full path qualified name)으로 형식으로 작성한다. 작성예) simple(bankware.corebanking.arrangement.arrangement.business.ArrArrIdExstncVdlnImpl)  참고: @ CbbClassInfo의 클래스타입이 “DTO\_SVLDTN”(Simple DTO Valiation) 또는 “DTO\_CVLDTN”(Complex DTO Validation) 인 클래스만 지정가능하다. |
| 서비스  입력검증  필수여부 | 해당 속성이 서비스 입력데이터항목으로 사용되는 경우, 서비스 전처리에서 조건없이 표준속성의 검증규칙을 적용할지 여부를 나타냄. Y인 경우, 조건없이 검증규칙을 적용함 |
| 부모속성명 | 해당 속성이 속성정보를 상속받아야 하는 부모속성을 의미함. 예)계약서비스제한금액통화코드 vs 통화코드 |

### 확장속성(Extended attribute)

CBP는 테이블의 속성이 기관별로 가변적일 때, 이를 원천테이블의 확장속성으로 관리하며, 확장속성의 추가/변경을 모델 변경없이 적용할 수 있다.

#### 테이블확장속성 추가

특정원천테이블에 대하여 테이블확장속성을 신규로 정의할 수 있다. 사전/메시지 > 속성 > 확장테이블속성 메뉴를 선택한다.

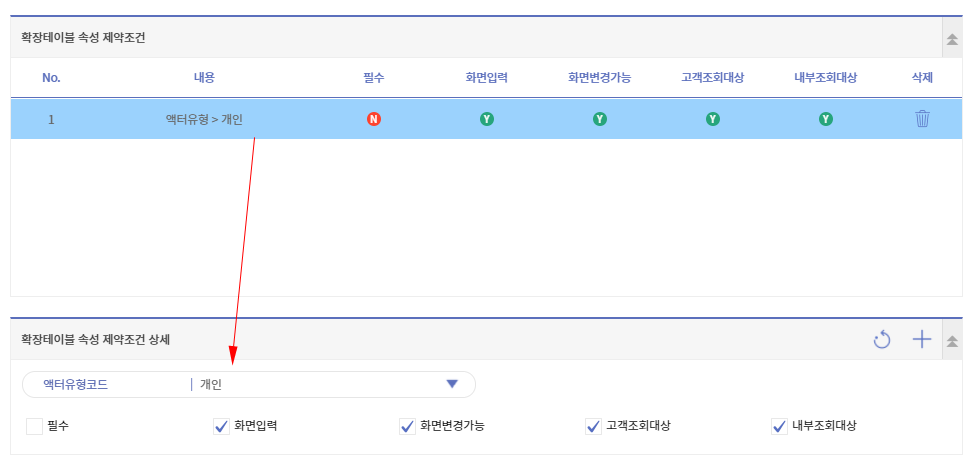


확장속성을 정의하고자 하는 테이블을 좌측 트리메뉴 영역에서 선택하고, “초기화” 버튼으로 초기화한 후, 기본정보 및 확장정보영역을 작성한다.

|  |  |
| --- | --- |
| 항 목 | 설 명 |
| 확장속성명 | 확장속성의 검색버튼으로 표준속성 팝업을 오픈하고, 표준속성을 선택한다. 즉 확장속성은 반드시 표준속성이어야 한다. |

|  |  |
| --- | --- |
| 항 목 | 설 명 |
| 표준속성검증 체크 | 해당 표준속성의 검증규칙을 적용할지 여부를 지정 |
| 확장속성검증 | 해당 테이블확장속성에 값을 지정할 때, 검증해야 할 검증규칙을 기술 |
| 속성검증방법 | 표준속성의 속성검증방법 준용 |
| 속성검증규칙 | 표준속성의 속성검증규칙 준용 |
| 속성검증확장규칙 | 표준속성의 속성검증확장규칙 준용 |

##### 확장정보



특정 테이블의 확장속성을 정의할 때, 추가정의특성의 값을 지정한다.

|  |  |
| --- | --- |
| 추가 정의 특성 | 설 명 |
| 필수여부 | 원천테이블의 Key생성 시, 확장속성이 필수적으로 지정되어야 하는지 여부. 액터테이블의 확장속성으로 생일이 정의되어 있고, 필수여부가 “Y”라면 이는 새로운 액터를 생성할 때, 반드시 생일 확장속성의 값이 지정되어야 함 |
| 화면입력여부 | 원천테이블의 Key에 대하여 화면에서 등록, 수정하는 화면이 있는 경우, 확장속성을 입력받을 지 여부를 의미 |
| 화면변경가능여부 | 원천테이블의 Key에 대하여 화면에서 변경하는 화면이 있는 경우, 확장속성을 변경할 수 있는지 여부를 의미 |
| 고객조회대상여부 | 원천테이블의 Key에 대하여 대고객 직접채널로 조회하는 화면이 있는 경우, 확장속성을 조회할지 여부를 의미 |
| 내부조회대상여부 | 원천테이블의 key에 대하여 내부 스태프용 조회하는 화면이 있는 경우, 확장속성을 조회할지 여부 |

## 코드등록 – 공통 코드관리

공통코드는 표준코드, 서브셋코드, 외부코드, 확장코드의 개념을 지원 한다.

|  |  |
| --- | --- |
| 용어 | 설 명 |
| 코드번호 | 고객유형코드, 계약상태코드처럼 특정한 유형의 열거형(enumeration)에 대한 부여한 고유한 Id |
| 코드 | 특정 코드번호에서 가능한 값의 나열 |

### 공통코드의 유형

|  |  |
| --- | --- |
| 유형 | 정 의 |
| 표준코드 | 일반적인 공통 코드를 지칭하는 유형으로서, 코드번호와 코드, 코드에 대한 텍스트만을 관리한다. |
| 표준서브셋코드 | 특정 표준코드의 부분집합으로서의 의미를 가지는 코드를 지칭한다. |
| 외부코드 | 멀티엔티티를 지원하기 위한 공통코드 유형이다. 하나의 코드번호에 대하여 기관별로 상이한 코드 값을 가지는 경우를 지원하기 위한 코드를 지칭한다. |
| 확장코드 | 은행코드, 금액유형코드, 통화코드처럼, 표준코드에서 관리하는 코드, 코드에 대한 텍스트 이외에 추가적인 정보항목이 있어서, 코드에 대한 별도 테이블이 존재하는 경우를 지원하기 위한 코드를 지칭한다. |

### 서브셋 코드(Subset Code)

서브셋코드(Subset Code)란 상위코드의 부분집합으로서 의미를 가지는 코드를 의미하므로, 서브셋코드는 상위코드의 코드목록을 벗어나는 코드를 가질 수 없다.

#### 표준 서브셋 코드

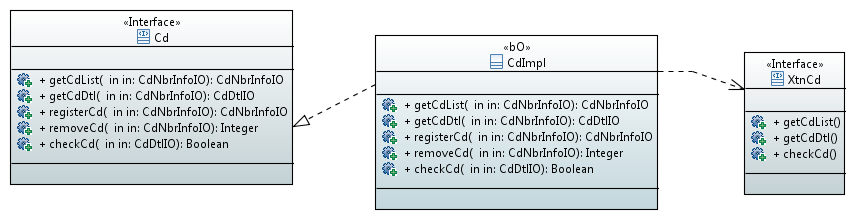
서브셋이 별도의 의미를 부여할 수 있는 집합인 경우, 표준서브셋코드로 생성한다. 표준서브셋코드는 별도의 의미를 지니므로, 모집합이 되는 상위코드번호와 다른 별도의 코드번호를 생성한다. 표준서브셋코드는 상위코드로서 표준코드만을 지원한다.

액터상세유형은 { 내국인, 해외거주자, 외국인, 상장기업, 비상장기업, 임의단체, 중앙정부, 지방자치단체 } 가 있을 수 있는데, 개인액터상세유형은 그 부분집합인 { 내국인, 해외거주자, 외국인 }으로 구성된다

### 확장코드(Extended Code)

은행코드, 금액유형코드, 통화코드처럼, 표준코드에서 관리하는 코드, 코드에 대한 텍스트 이외에 추가적인 정보항목이 있어서, 코드에 대한 별도 테이블이 존재하는 경우를 지원하기 위한 코드를 지칭한다.

* 확장코드를 지원하는 클래스는 XtnCd interface를 구현하여야 한다.
* 확장코드를 지원하는 클래스는 Singleton으로 구현되어야 하고, 만약 Prototype형 BO(Base Object)가 확장코드를 지원하는 경우, Manager에서 XtnCd를 구현한다.



* 확장코드의 등록은 코드등록화면이 아닌 해당 업무의 별도 등록화면을 구현한다.
* 확장코드도 기관별 코드가 달라지는 케이스가 있는데, 그러한 구현은 확장코드지원클래스에서 구현하여야 한다.
* 확장코드의 다국어지원 방식 - 확장코드의 다국어지원은 표준코드의 방식대로 코드번호+코드를 Key로 하여 다국어변환테이블을 생성한다.

### 외부코드(External Code)

외부(external)코드는 CBP에서 데이터항목 이상의 의미를 지니지 않아야 한다. 즉, 단순히 보관되는 데이터로서만 의미있는 코드이다.

고객유치 시 당행과 거래한 사유를 {접근성, 인지도, 지인의 소개, 좋은이미지, 친절한직원, 상품구성} 등으로 관리할 수 있는데, 이는 데이터항목으로서 의미를 지니지만 CBP에서 로직의 분기에 사용되지 않는다. 또한 성격상 기관마다 관리하고자 하는 항목이 다를 개연성이 매우 높다.

* 외부코드의 특징은 기관별로 요건이 매우 상이하여, 표준화하기가 불가능하다는 특징이 있다.
* 외부코드는 반드시 기관별로 정의한다.
* 외부코드는 CBP에서 분기의 조건으로 만들어서는 안된다.
* 외부코드를 enum으로 생성해서는 안된다.

# 서비스 개발

**서비스 개발 관련되어 작업분류**

|  |  |  |  |
| --- | --- | --- | --- |
| 대분류 | 중분류 | 세분류 | 설 명 |
| 개발 | 객체 | 어노테이션을 이용한 객체의 구분 | 어노테이션 종류 |
| 데이터 | 형태 | 상수 | CBP 패키지에서 상수처리 방법 |
| Big Decimal | Big Decimal 처리 방법 |
| 저장소 | 데이터 컨테이너 | 데이터를 객체로 저장하는 방법 |
| 언어 | 에러 메시지 다국어 | 다국어를 처리하는 방법 |
| 속성 | 확장속성 | 확장속성 | 확장속성 등록 및 처리 방법 |
| 메시지 | Json | Json 메시지 |  |
| 에러메시지 | 다중 에러메시지 |  |
| 거래 | 트랜잭션 | 트랜잭션 분리 |  |
| 연동거래 | 내부연동거래 |  |
| 취소정정 | 취소정정 |  |
| 성능 | 페이징 | 페이징 |  |
| 캐쉬 | 캐쉬 |  |
| 룰 | Biz Rule | 업무규칙 |  |
| 검증/테스트 | 검증 | 패러미터 값 검증 | 서비스 검증 및 테스트 방법 |

## 어노테이션(Annotation) 종류

기존의 객체지향 개념 및 언어에 영향을 주지 않으면서 객체/속성/메소드 등에 새로운 의미와 처리방법을 정의할 수 있다.

|  |  |  |
| --- | --- | --- |
| 구분 | 세분류 | 설 명 |
| 공통 클래스 | @BxmCategory | 클래스의 논리명 작성, 설명 |
| @CbbClassInfo | 클래스 종류 기술 |
| @BxmBean("XXXXXX") | 모든 빈 |
| @BxmService | 서비스 클래스 |
| 서비스 오퍼레이션 | @BxmServiceOperation | 빈(Bean) 서비스의 오퍼레이션 표시 |
| @TransactionalOperation | 오퍼레이션의 트랜잭션 처리 |
| @CbbSrvcInfo | 서비스 코드/명 기술, cbb 데이터베이스에 저장 |
| 메소드 | @Override | 인터페이스 메소드 구현시 반드시 기술 |
| @BxmCategory | 메소드의 논리명 작성, 설명 |

사용 예)

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 케이스 | 세분류 | 설 명 |
| 선언 | 프로토타입 클래스 선언 | @BxmCategory | 클래스의 논리명 작성, 설명 |
| @CbbClassInfo | 클래스 종류 기술 |
| @BxmBean("XXXXXX") |  |
| BO 클래스 | @BxmCategory | 클래스의 논리명 작성, 설명 |
| @CbbClassInfo | 클래스 종류 기술 |
| @BxmBean("XXXXXX") | 모든 빈 |
| 서비스 클래스 | @BxmCategory | 클래스의 논리명 작성, 설명 |
| @CbbClassInfo | 클래스 종류 기술 |
| @BxmService | 서비스 클래스 |
| 서비스 오퍼레이션 | @BxmServiceOperation | 빈서비스의 오퍼레이션 표시 |
| @TransactionalOperation | 오퍼레이션의 트랜잭션 처리 |
| @CbbSrvcInfo | 서비스 코드/명 기술, cbb 데이터베이스에 저장 |
| 메소드 | @Override | 인터페이스 메소드 구현시 반드시 기술 |
| @BxmCategory | 메소드의 논리명 작성, 설명 |

### 공통 클래스(Common Class) 어노테이션

**@BxmCategory**

logicalName은 클래스에 대한 설명을 명사 위주로 간략히 영어로 기술하고, type, description 항목은 작성하지 않는다.

@BxmCategory(logicalName = "Institution Parameter Manager" )

public class InstParmMngrImpl implements InstParmMngr {

**@CbbClassInfo**

CBB에서 사용하는 클래스의 종류를 기술한다. 가능한 Class Type 목록은 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Type** | **이름** | **설명** | **비고** |
| ARR\_CND | Arrangement Condition | 다양한 계약조건 구현을 위해 계약조건을 확장한 클래스 |  |
| ARR\_SRVC\_VLDTN | Arrangement Service Validation | 해당 계약에 대하여 유효한 서비스인지 검증기능을 구현한 클래스 |  |
| DTO\_SVLDTN | SimpleDTOValidation | 서비스입력DTO검증규칙에서 사용할 수 있는 Plug-in 클래스. Simple Type |  |
| DTO\_CVLDTN | ComplexDTOValidation | 서비스입력DTO검증규칙에서 사용할 수 있는 Plug-in 클래스. Complex Type |  |
| CONSTANT | CONSTANT | 상수정의 전용 클래스 |  |
| SERVICE\_BO | Service BO | 이자계산, 수수료계산, ~정보 관리와 같이 기능 제공을 목적으로 설계된 클래스로 클래스 내부에 별도의 데이터를 관리하지 않는 전통적인 방식의  클래스. 클래스의 객체 생성은 BXM의 getBean에 의해 객체를 획득함 |  |
| ENTITY\_BO | Entity BO | 계약, 고객, 부서와 같이 클래스 내부에서 데이터를 멤버변수로 관리하며, 해당 데이터를 기반으로 관련된 기능을 제공하는 형태의 도메인 클래스 예) 계약, 고객, 부서, 직원 등. |  |
| MNGR | Manager | 엔티티BO 방식으로 관리되는 클래스의 인스턴스(객체)를 생성하는 역할을 담당함 |  |
| NBRG | Numbering Extention | 공통채번규칙에서 사용할 수 있는 Plug-in 클래스 |  |
| VO | Value Object | OMM이외의 Value Object |  |
| XTN\_CODE | Extended Code | 확장코드를 구현한 클래스 |  |
| ITEM\_READER | Item Reader | BXM Batch의 Item Reader |  |
| ITEM\_WRITER | Item Writer | BXM Batch의 Item Writer |  |
| ITEM\_PROCESSOR | Item Processor | BXM Batch의 Item Processor |  |
| SERVICE | Service | Service | 미작성 시 자동설정 |
| BO | Business Object | BxmBean |
| DTO | DataTransferObject | OMM으로부터 생성된 입출력DTO |
| ENUM | ENUM | ENUM |

여러 개의 어노테이션을 동시에 사용가능하다(다중선택가능)

@CbbClassInfo(classType={"MNGR","XTN\_CODE"}) // Class type List

public class ClassMngrImpl implements ClassMngr {

**@BxmBean("XXXXXX")**

* 모든 Bean에 대하여 작성
* 클래스명과 동일한 경우 ("XXXXXXX")는 기술하지 않아도 됨.
* getBean 해서 사용하는 prototype 객체의 Bean명은 interface명으로 함. 즉 impl이 빠진 명칭(클래스명과 Bean명이 틀리는 유일한 경우). 단, interface를 구현한 ~impl이 여러 개인 경우, 클래스에서 impl을 제외한 이름으로 작성한다.
* 자바 interface에서는 기술하지 않아도 됨.

// 1. 선언

@BxmCategory(logicalName = "Standard Attribute BO")

@CbbClassInfo(classType={"CORE\_BO"})

@BxmBean

public class StdAtrbtMngrImpl implements StdAtrbtMngr {

**@BxmService**

서비스클래스에 대하여 반드시 기술해야 하며, 어노테이션내에 기술하는 이름은 서비스클래스명과 일치하게 작성해야 한다.

@BxmCategory(logicalName = "Aproval Demand Service")

@CbbClassInfo(classType={"SERVICE"})

@BxmService("AprvlDemandSvc")

public class AprvlDemandSvc {

### 서비스 오퍼레이션(Service Operation) 어노테이션

**@BxmServiceOperation**

서비스 Bean 생성용. 일반적으로 메소드명과 동일.

**@TransactionalOperation**

트랜잭선 보장을 위한 어노테이션. C,U,D가 있는 서비스만 기술하고, 조회서비스는 제거한다.

**@CbbSrvcInfo**

서비스코드, 서비스명 기술 ==> cbb서비스원장에 반영됨

/\*\*

\* staff login process, check whether login id is valid user and password is correct

\*

\* @param input

\* @return

\* @throws ApplicationException

\*/

@BxmServiceOperation("login")

@TransactionalOperation

@CbbSrvcInfo(srvcCd="SUP0010101",srvcNm="Staff Login Service",srvcAbrvtnNm="logIn")

public LogInSvcGetLoginUserOut login(LogInSvcGetLoginUserIn input)

throws BizApplicationException {

Staff staff = staffMgr.getStaff(input.getStaffId());

if (staff == null) {

### 메소드(Method) 어노테이션

**@Override**

Interface의 method를 구현한 경우, 반드시 기술한다.

**@BxmCategory**

메소드의 Logical Name 작성

@Override

@BxmCategory(logicalName="confirm of supervisor approval")

public void confirmSpvsrAprvl(AprvlCndCnfrmtnIn in, IOmmObject srvcOmm ) throws AprvlApplicationException, BizApplicationException

## 상수(Constant)

* private 상수는 일반적인 상수 선언으로 쓴다.
* public 상수는 컴파일 시 자동으로 상수값으로 대체된다(Inlining). 상수값이 변경되면 관련된 모든 클래스를 재컴파일해야 하므로, 즉시배포(hot deploy)가 수행되지 않는 문제점이 발생할 수 있다.

### BXM에서 상수 선언

원시형(primitive type)에 대해서만 상수선언 규칙을 따른다. 배열([]), 객체(List,Map...) 생성하는 경우는 상수선언규칙에 따를 필요가 없다.

public static final int DT[] = {384, 355, 354, 384, 354, 354};

public static final String CAL[] = {"1212122322121", "1212121221220", "1121121222120"};

public static final String TEST = value("test33333");

public static final Integer TEST2 = value(33333);

private static final <T> T value(T t){

return t;

}

### 일반적인 상수 선언

private인 경우, 아래와 같이 일반적인 상수를 선언한다. 이 경우 컴파일시 해당 값이 값으로 대체되어 바이트코드(bytecode)에 반영된다.

private static final String TEST ="test333";

## Big Decimal

* 금액 연산은 BigDecimal 을 사용하는 것을 권장한다.
* 초기화는 String 타입으로 하고, double이나 int 형으로 사용하지 않는다.

BigDecimal 사용하지 않은 경우의 문제점은 다음과 같다.

* 정확한 소수점 연산이 되지 않은 경우가 있으므로(아래 소스 참고), 부동소수점 유형을 사용하는 경우, equal연산이 의도한 바로 적용되지 않을 수 있음(ex. 단말 입력 값 0.5 != DB 조회 값 0.5)
* BigDecimal을 사용하지 않은 경우 나누기 연산 등을 마지막에 수행하는 등, 연산 순위를 고려해야 한다.

import java.math.BigDecimal;

public class Exam1 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// Double 연산의 문제

BigDecimal s1 = new BigDecimal("2.0");

BigDecimal s2 = new BigDecimal("1.1");

System.out.println(s1.subtract(s2)); // 결과 : 0.9

System.out.println(2.0 - 1.1); // 결과 : 0.8999999999999999

// BigDecimal의 초기화는 문자열로 해야됨.

//그렇지 않고 double 하게 되면 위의 문제가 그대로 발생.

BigDecimal x = new BigDecimal(2.0 - 1.1); // 잠재 오류 있음

System.out.println(x);

// 결과 : 0.899999999999999911182158029987476766109466552734375

// 무한소수 처리

BigDecimal a = new BigDecimal("3");

BigDecimal b = new BigDecimal("1");

// System.out.println(b.divide(a));

// Exception 발생 Non-terminating decimal expansion; no exact representable decimal result.

System.out.println(b.divide(a, 3, BigDecimal.ROUND\_DOWN));

// 결과 : 0.333

}

}

## 데이터 컨테이너(Data Container)

### 베이스 컴포넌트(Base Component)별 데이터 컨테이너

각 베이스 컴포넌트 별로 데이터 컨테이너 영역을 할당하고, 각 베이스 컴포넌트 프로젝트에서 키를 관리한다. key는 Integer로 0~99 까지 100개 할당.

#### CbbDataContainerType

* FRAMEWORK : 프레임워크 영역
* FRAMEWORK\_EXTENTION : 프레임워크 확장 영역
* APP\_CM
* APP\_SM
* APP\_SV
* APP\_AT
* APP\_AC
* APP\_AR
* APP\_PD
* APP\_ST

#### CbbApplicationContext 메소드

* getCbbDataContainerItem(CbbDataContainerType, Integer)
* setCbbDataContainerItem(CbbDataContainerType, Integer, Object)

#### 사용예제

* key 관리 constant class(interface).

package bankware.corebanking.applicationcommon.constant;

public class CmDataContainerDefinition {

public static final Integer CM\_COM\_DATA = 0;

}

* 맵 데이터 처리(Map data handling)

Map<String,Object> cmnData = (Map<String, Object>) CbbApplicationContext.getCbbDataContainerItem(CbbDataContainerType.APP\_CM,CmDataContainerDefinition.CM\_COM\_DATA);

if(cmnData == null){

cmnData = new HashMap<String, Object>();

CbbApplicationContext.setCbbDataContainerItem(CbbDataContainerType.APP\_CM,CmDataContainerDefinition.CM\_COM\_DATA, cmnData);

}

cmnData.put("key","value");

* 정수 데이터 처리(Integer data handling)

Integer cmnData = CbbApplicationContext.getCbbDataContainerItem(CbbDataContainerType.APP\_CM,CmDataContainerDefinition.CM\_COM\_DATA);

if (cmnData == null) {

cmnData = 1;

} else {

cmnData ++;

}

CbbApplicationContext.setCbbDataContainerItem(CbbDataContainerType.APP\_CM,CmDataContainerDefinition.CM\_COM\_DATA, cmnData);

* DataContainer에 객체를 set 한 후, 객체 내부의 값을 변경하는 경우는 DataContainer에 다시 set할 필요가 없지만 (Map,List,VO class...)
* 객체 자체가 변경되는 경우는 다시 DataContainer에 set을 해주어야 함. (String, Integer ...)

### 확장된 데이터 컨테이너

* 확장 가능하도록 key를 String 으로 사용함.

#### CbbApplicationContext 메소드

* getCbbExtendDataContainerItem(String)
* setCbbExtendDataContainerItem(String, Object)

## 에러메시지 다국어

Exception 발생시 arguments에 delimiter "@" 을 맨앞에 추가한다. delimiter "@" 으로 변환할 키값과 데이터를 구분하며, 다국어 테이블의 원천키값으로 조회하여 변환한다. 변환 종류 값과 상관없이 조회한다.

throw new BizApplicationException("BXME3000", new Object[]{"@atrbt",input.getNm())});

### 용어정의(Terms)

CBP 에러메시지는 크게 두가지로 구별된다.

* ~~화면에러메시지: 화면스크립트 상에서 출력되는 메시지~~
* 서버에러메시지: 서버 프로그램에서 서비스 처리 결과로 출력되는 메시지

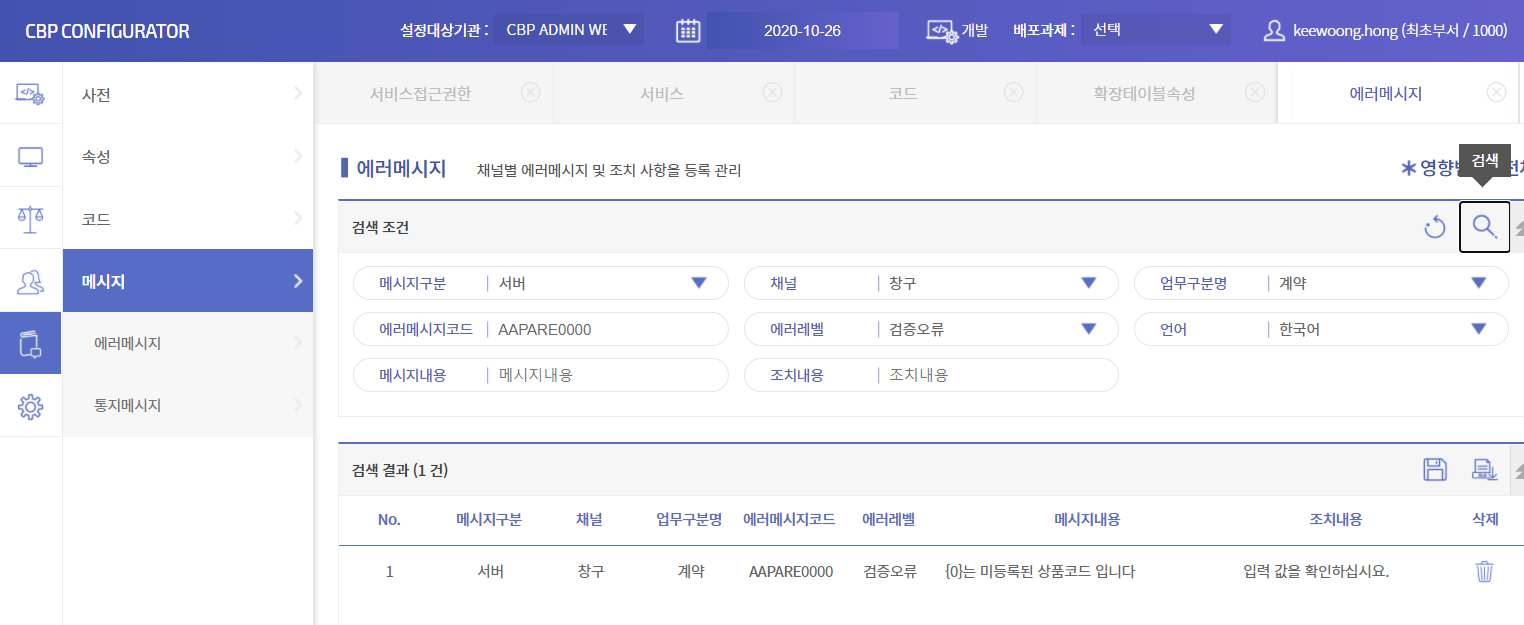
서비스별 에러메시지는 시스템의 화면, 서비스에서 출력되는 모든 에러메시지로, 공통 에러메시지 테이블, 서버어플리케이션의 하드코딩된 에러메시지, 화면스크립트의 하드코딩된 에러메시지를 대상으로 하고, 시스템솔루션(WAS, Middleware, Application F/W, DBMS 등)에서 산출되는 에러메시지는 제외한다.

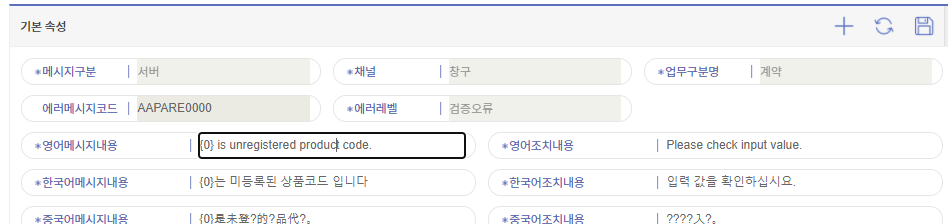
*※ UI에서 사용하는 메시지는 Nexacro Platfrom이 제공하는 globaldataset로 관리함에 따라 본 프로젝트에서는 ‘화면에러메시지’를 별도 관리하지 않음*

### 설정작업 (Configuration Items)

#### 에러메시지 신규 등록

CBP시스템에서 제공하지 않았던 신규 에러메시지를 등록한다. 신규 에러메시지는 반드시 CBP 소스에 반영되어야 한다. 사전/메시지 > 메시지 > 에러메시지 메뉴를 선택한다.





초기화 버튼을 누른 후, 메시지구분, 채널, 업무구분, 에러레벨과 함께 언어별 메시지내용, 조치내용을 작성한다. 메시지를 변수로 처리해야 하는 경우에는 {0}, {1}와 같이 작성한다. 채널코드 중 창구가 디폴트 채널이며 필수이다. 화면에 표시되는 3개의 언어는 모두 필수적으로 등록되어야 하며, 사용하지 않는 언어는 ‘ ‘ Space를 넣어서 등록하도록 한다.

#### 에러메시지 수정

본 화면에서 변경하고자 하는 에러메시지를 조회하여 조회 결과를 선택한다. 하단 상세 영역에서 에러메시지 내용을 변경 한 후 저장한다.

#### 에러메시지 채널 추가

기존 에러메시지코드에 대하여 채널을 추가하는 경우이다. 이는 특정 채널에 대한 에러메시지가 디폴트 채널인 창구와 다른 메시지가 산출되어야 하는 경우에 해당한다. 별도 채널의 경우, 변수항목이 다르거나, 메시지 내용이 다를 수 있으므로, 반드시 해당 업무영역 담당자와 협의하여 관련 소스에 반영여부를 검토해야 한다. 에러메시지 내용 수정과 동일하게 동작하며, 상세영역에서 [+] 버튼을 눌러서 채널항목을 활성화 한 후 채널값을 변경하여 기존 에러메시지코드에 대하여 채널을 추가한다.

## JSON 메시지

### 객체매퍼(ObjectMapper)

ObjectMapper는 여러 쓰레드가 동시에 접근해도 문제가 없으며(thread safe), 변환된 객체(object) 를 캐쉬에 저장하고 있다. 따라서, 객체를 매번 새로 만들기 보다는 애플리케이션 컨테이너(application container)가 가지고 있는 객체매퍼(ObjectMapper)를 사용하는 것이 바람직하다.

* 직접 사용하는 경우

ObjectMapper oMapper = CbbApplicationContext.getDataMapper();

* private method에서 사용하는 경우

ObjectMapper oMapper = \_getDataMapper();

private ObjectMapper \_getDataMapper() {

return CbbApplicationContext.getDataMapper();

}

### 직렬화(Serialize)

자바 객체를 JSON 메시지로 변환한다.

* 예제 1

ObjectMapper oMapper = \_getDataMapper();

oMapper.writeValueAsString(output);

* 예제 2

ObjectMapper objectMapper = \_getDataMapper();

StringWriter stringEmp = new StringWriter();

objectMapper.writeValue(stringEmp, emp1);

### 역직렬화(Deserialize)

JSON 메시지를 자바 객체로 변환한다.

// JSON message to Object

ObjectMapper oMapper = \_getDataMapper();

SqlExecuteOrder order = oMapper.readValue(in.getSvcObject(),SqlExecuteOrder.class);

### 노드(Node)

* sample1

ObjectNode node = oMapper.createObjectNode();

result.setResultObj(node.set("success", BooleanNode.FALSE).toString());

## 다중에러메세지

### 예외(Exception) 발생이 명확할 때.

import bankware.corebanking.context.CbbApplicationContext;

...

private void \_validationErrorProcess(List<BizApplicationException> validationErrorList) throws BizApplicationException {

for (BizApplicationException e:validationErrorList){

CbbApplicationContext.addErrorMessage(e); //에러 메시지 추가.

}

throw new BizApplicationException("AAAAAAAA", null); //대표 에러메시지, 첫번째에 들어감.

}

### 에러메시지 개수를 체크하여 예외(Exception) 발생.

if(CbbApplicationContext.getErrorMessageCount() > 0){

throw new BizApplicationException("AAAAAAAA", null);

}

### 에러 응답메시지 예

{

"header":{

"traceElements":[

],

"instCd":"001",

"userGrpCd":"01",

"custId":null,

"staffId":"00000001",

"cmpntCd":null,

"srvcCd":"PCM0038400",

"brnchCd":"0000",

"txDt":"20150203",

"txHms":"101822",

"lngCd":"kr",

"crncyCd":"",

"chnlDscd":"02",

"txTmstmp":1422926302000,

"tmZone":"",

"sysTmstmp":1429665502000,

"guid":"7779A0D0-E88D-11E4-B88F-239DD7C90000",

"application":"UPSvc",

"service":"LogInSvc",

"operation":"login",

"locale":"kr\_KR",

"returnCode":"1",

"data":null,

"systemId":null,

"remoteAddr":null,

"uuid":"7779A0D0-E88D-11E4-B88F-239DD7C90000",

"userId":null,

"errorMessages":[

{

"messageCode":"BXME9999",

"message":"BXME9999",

"traceMessage":"tat com.mysql.jdbc.MysqlIO.send(MysqlIO.java:3591)",

"detailMessages":"'autoReconnect=true' to avoid this problem."

},

{

"messageCode":"BXME9999",

"message":"BXME9999",

"traceMessage":"tat com.mysql.jdbc.MysqlIO.send(MysqlIO.java:3591)",

"detailMessages":"'autoReconnect=true' to avoid this problem."

}

]

}

}

## 트랜잭션 분리(Transaction Require\_New)

### 트랜잭션 정의

단일 서비스는 트랜잭션 기반으로 처리("Transactional") 되어야 한다. 즉, 해당 서비스에서 갱신한 데이터는 정상적으로 모두 반영(Commit)되거나 예외에 의하여 모두 무효(Rollback)처리되어야 한다.

트랜잭션 분리는 서비스의 정상종료(Commit)나 예외종료(Rollback)와 상관없이 서비스 내의 특정 데이터 처리를 별도의 트랜잭션으로 처리할 때 사용하는 방법이다. 트랜잭션분리된 메소드 내의 갱신데이터는 해당 메소드가 종료되는 시점에 Commit된다.

### 사용 사례

* 비밀번호가 틀릴 때, 서비스는 예외종료처리하지만, 비밀번호오류횟수를 +1 할 때,

### 코딩방법 및 샘플코드

분리되어야 할 Transaction에 @TransactionalOperation (propagation= Propagation. REQUIRES\_NEW)를 기술한다.

@TransactionalOperation(propagation=Propagation.REQUIRES\_NEW) //트랜잭션분리

private String \_createNewSpvsrAprvl(AprvlCndCnfrmtnIn in, IOmmObject srvcOmm,

List<String> matchedCnds) throws BizApplicationException,

AprvlApplicationException {

AprvlInfoIn aprvlInfoIn = new AprvlInfoIn();

aprvlInfoIn.setInstCd(cmnUtilMngr.getInstCd());// set [기관코드]

aprvlInfoIn.setAprvlTrgtSrvcCd(cmnUtilMngr.getSrvcCd());// set [승인대상서비스코드]

aprvlInfoIn.setWflowInstncId(null);// set [워크플로우인스턴스식별자]

aprvlInfoIn.setAprvlTrgtBizIdCntnt(in.getAprvlTrgtBizIdCntnt());// set [승인대상업무식별키내용]

aprvlInfoIn.setAprvlDemandStaffId(cmnUtilMngr.getStaffId());// set [승인요청스태프식별자]

aprvlInfoIn.setAprvlStsCd(CCM01.WFLOW\_APRVL\_STS\_CD\_BEF\_WFLOW);// set [승인상태코드] 09.workflow 요청 전

aprvlInfoIn.setAprvlAplctnDt(cmnUtilMngr.getTxDt());// set [승인신청년월일]

aprvlInfoIn.setAprvlAplctnCntnt(srvcOmm.toString());// set [승인신청내용]

aprvlInfoIn.setAprvlTpCd(CCM01.WFLOW\_TP\_CD\_APRVL);// set [승인유형코드] 11905 Workflow 유형코드 01 결재프로세스

aprvlInfoIn.setLastAprvlDt(null);// set [최종승인년월일]

aprvlInfoIn.setLastChngGuid(null);// set [최종변경GUID]

/\* 결재BO 호출 \*/

String aprvlId = aprvlMngr.createAprv(aprvlInfoIn);

/\* 결재발생조건 DBIO \*/

for ( String matchedcnd : matchedCnds ) {

CmAprvlIssueCndDIO indao = new CmAprvlIssueCndDIO();

indao.setAprvlId(aprvlId);// set [승인식별자]

indao.setAprvlCndNbr(matchedcnd);// set [결제조건번호]

cmnUtilMngr.setHdrClmn(indao); // Institution Code, National Code, GUID, Last Change TimeStamp

cmAprvlIssueCndD.insert(indao); //결재발생조건DBIO

}

return aprvlId;

}

트랜잭션분리된 method내에서 Exception이 발생한 경우는, 분리된 트랜잭션에서 갱신한 데이터도 rollback처리된다.

## 내부연동거래

### 연동거래

서비스에서 서비스를 호출하는 내부 연동거래는 “CbbInternalServiceExecutor” Class를 이용한다.

- 동기호출 : execute(String, R)

- 비동기호출 : executeAsync(String, R, long)

- 비동기호출(대기) : executeAsyncForWait(String, R, long)

서비스가 하나씩 호출 될 때마다 Call Depth가 0에서부터 +1씩 증가한다. 최초 인입서비스는 ‘0’이다. Call Depth의 확인은 CbbApplicationContext.getCurrentServiceCallDepth() 를 통하 가능하다. 내부연동거래 시에도 DataContainer의 공유가 가능하다.

### 예제

서비스 호출

DeptSrchSvcGetDeptListIn inputDto = new DeptSrchSvcGetDeptListIn();

inputDto.setInstCd("101");

CbbInternalServiceExecutor.execute("SxxYYY2008401", inputDto);

call depth에 따라 처리가 필요할 때. 첫 기동거래에서 callDepth는 0이다.

Integer callDepth = CbbApplicationContext.getCurrentServiceCallDepth();

if(callDepth == null || callDepth == 0 ){

//TODO do something

}

## 캐쉬(Cache)

### 캐쉬 종류

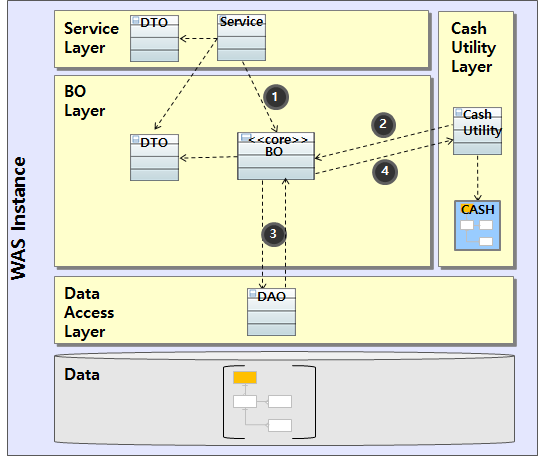
CBP에서 사용하는 캐쉬로는 크게 두 가지 종류가 있다.

- 인스턴스캐쉬 : 거래에 관계없이 모든 거래에서 공용으로 사용할 수 있는 캐쉬

- 트랜잭션캐쉬 : 중복을 제거할 목적으로 하나의 거래 내에서만 사용할 수 있는 캐쉬

### 인스턴스캐쉬

#### 처리 FLOW



* 서비스에서 BO 호출
* BO에서 인스턴스캐쉬에서 데이터가 조회
* 인스턴스캐쉬에 데이터가 없으면 DAO를 통하여 DB로부터 조회
* DB에서 조회한 데이터를 인스턴스캐쉬에 저장

#### 용도

* 인스턴스캐쉬는 변경이 거의 없는 데이터를 DB에서 매번 조회하면 성능이 떨어지기 때문에 이를 보완하기 위하여 사용하는 캐쉬이다.
* 인스턴스캐쉬에 저장되는 데이터는 되도록이면 변경사항이 적고 자주 사용하는 데이터 위주로 캐쉬해야 한다.

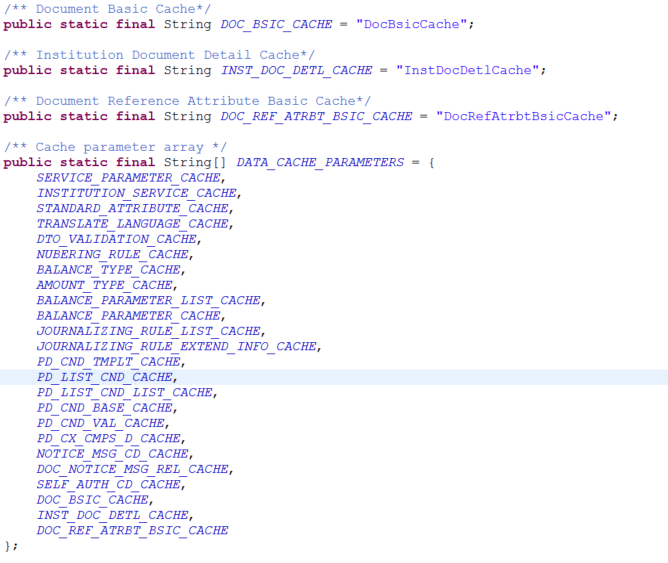
#### 제약사항

* 인스턴스캐쉬는 한번 저장되면 사용자가 제거하지 않는한 인스턴스를 다운할 때까지 제거되지 않기 때문에 아무 정보나 저장하면 안 된다.
* 특히 DTO가 아닌 POJO 클래스의 Object를 저장하면 해당 Object는 가비지콜렉션 대상이 않되기 때문에 주의해야 한다.
* 거래처리 과정에서 매번 생성하는 POJO 클래스 Object을 저장할 경우 메모리 릭이 발생할 수 있다.
* 위와 같은 문제가 발생할 수 있으므로 인스턴스캐쉬에 저장되는 정보는 DBIO의 DTO에 한해서만 저장해야 한다.
* 또한, 자주 변경되는 데이터는 인스턴스캐쉬에 저장하는 것은 지양해야 한다.
* 자주 변경되는 데이터가 인스턴스캐쉬에 저장되면 변경시마다 모든 WAS 인스턴스와 동기화를 해야하기 때문이다.

#### 새로운 캐쉬 생성 방법

* 캐쉬관리 클래스에 등록되지 않은 캐쉬를 사용하면 데이터는 캐쉬되지 않는다.
* 캐쉬관리 클래스에 새로운 캐쉬를 위한 캐쉬명을 등록한다.
* 새롭게 등록한 캐쉬명을 캐쉬관리 Array List에 캐쉬명을 등록한다.

- 캐쉬관리 클래스 : bankware.corebanking.applicationcommon.cache.interfaces.DataCacheCoreConfig



#### 캐쉬 클래스 개발

* 인스턴스캐쉬는 하나의 Base Component 내에서만 사용할 수 있으므로 Base Component 별로 캐쉬 클래스를 개발해야 한다.
* 새롭게 개발하는 캐쉬 클래스 파일은 반드시 두개의 메소드를 Override 해야 한다.
* init() 메소드에서 해당 캐쉬 클래스를 사용하여 인스턴스캐쉬에 저장하는 모든 캐쉬를 Clear 한다.
* saveCache() 메소드를 Override하여 캐쉬관리 클래스에 등록된 캐쉬인지 검증한다.
* 아래 소스는 공통에서 관리하는 캐쉬 클래스 파일이다.

package bankware.corebanking.applicationcommon.cache.business;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import bxm.common.annotaion.BxmCategory;

import bxm.container.annotation.BxmBean;

import bankware.corebanking.applicationcommon.cache.interfaces.DataCacheConfig;

import bankware.corebanking.applicationcommon.cache.interfaces.DataCacheManagerImpl;

/\*\*

\* Data cache manager

\*

\* @author hyungman.kang

\* @history

\* 2015.12.22 initial

\*/

@BxmBean

@BxmCategory(logicalName = "Data cache manager")

public class CmDataCacheManager extends DataCacheManager {

final static Logger logger = LoggerFactory.getLogger(CmDataCacheManager.class);

@PostConstruct

public void init() {

super.clearCache(DataCacheConfig.DTO\_VALIDATION\_CACHE);

super.clearCache(DataCacheConfig.STANDARD\_ATTRIBUTE\_CACHE);

super.clearCache(DataCacheConfig.TRANSLATE\_LANGUAGE\_CACHE);

super.clearCache(DataCacheConfig.NUBERING\_RULE\_CACHE);

}

/\*\*

\* Storing data in the cache

\*

\* @param cacheName (required) Cache name

\* @param cacheKey (required) Cache key

\* @param data (required) Object to cache

\* @return Object cached object

\*/

@Override

public Object saveCache(String cacheName, String cacheKey, Object data) {

if (!isValidCacheName(DataCacheConfig.DATA\_CACHE\_PARAMETERS, cacheName))

return null;

return super.saveCache(cacheName, cacheKey, data);

}

}

#### 데이터 변경, 삭제 시 인스턴스캐쉬 삭제 방법

* 인스턴스캐쉬는 WAS 인스턴스 별로 분리되어 있기 때문에 데이터 변경이나 삭제시 인스턴스캐쉬의 데이터를 삭제하면 해당 WAS 인스턴스 내에서만 삭제된다.
* 하나의 WAS 인스턴스에서 데이터 변경이나 삭제시 모든 WAS 인스턴스에 저장되어 있는 캐쉬를 동기화를 위해서는 DataCacheManager 클래스에서 제공하는 메소드를 호출해야한다.
* clearInstanceCache 메소드는 CBP가 실행되고 있는 모든 WAS 인스턴스에 명령어를 전달하여 캐쉬에 저장되어 있으면 삭제한다.

@Override

public StdAtrbt createStandardAttribute(StdAtrbtMngrRegisterIn in) throws BizApplicationException {

...

// clear cache

cmDataCacheManager.clearInstanceCache(DataCacheConfig.STANDARD\_ATTRIBUTE\_CACHE,

cmnSysHdrUtil.getLanguageCode().concat(in.getAtrbtNm()), null);

return stdAtrbt;

}

#### 인스턴스캐쉬 API

캐쉬에 데이터 저장

/\*\*

\* 입력된 데이터를 캐쉬에 저장

\*

\* @param cacheName (required) DataCacheConfig class에 등록한 캐쉬명

\* @param cacheKey (required) 입력된 데이터를 구별하기 위한 캐쉬키 값

\* @param data (required) 캐쉬에 저장할 데이터

\* @return 입력으로 받은 데이터를 반환한다.

\*/

public Object saveCache(String cacheName, String cacheKey, Object data);

캐쉬에 저장된 모든 데이터 삭제

/\*\*

\* 캐쉬에 저장된 모든 데이터를 삭제

\*

\* @param cacheName (required) DataCacheConfig class에 등록한 캐쉬명

\*/

public void clearCache(String cacheName);

특정 키에 해당하는 데이터 삭제

/\*\*

\* Clearing data from the cache

\*

\* @param cacheName (required) DataCacheConfig class에 등록한 캐쉬명

\* @param cacheKey (required) 삭제하고자 하는 데이터의 캐쉬키 값

\*/

public void clearCache(String cacheName, String cacheKey);

특정 키에 해당하는 데이터 중 원하는 데이터만 삭제

/\*\*

\* 저장된 데이터가 List 형식인 경우에 List에 속한 특정 데이터만 삭제

\*

\* @param cacheName (required) DataCacheConfig class에 등록한 캐쉬명

\* @param cacheKey (required) 삭제하고자 하는 데이터의 캐쉬키 값

\* @param dataKey (required) List 속한 데이터의 키 값

\*/

public void clearCache(String cacheName, String cacheKey, String dataKey);

모든 WAS 인스턴스에 저장된 데이터 동기화

/\*\*

\* 등록이나 변경시 모든 WAS 인스턴스에 저장된 데이터 동기화하고자 하는 경우

\*

\* @param cacheName (required) DataCacheConfig class에 등록한 캐쉬명

\* @param cacheKey (required) 삭제하고자 하는 데이터의 캐쉬키 값

\* @param dataKey (required) List 속한 데이터의 키 값

\*/

public void clearInstanceCache(String cacheName, String cacheKey, String dataKey);

캐쉬에 저장된 데이터 조회

/\*\*

\* Get data from the cache

\*

\* @param cacheName (required) DataCacheConfig class에 등록한 캐쉬명

\* @param key (required) 조회하고자 하는 데이터의 캐쉬키 값

\* @return 저장된 데이터를 변수 타입에 맞게 변환하여 반환

\*/

public <T> T getCacheData(String cacheName, String cacheKey);

테이터 크기

/\*\*

\* 캐쉬에 저장된 데이터 크기

\*

\* @param cacheName (required) DataCacheConfig class에 등록한 캐쉬명

\* @return int 캐쉬명에 해당하는 데이터의 크기 반환

\*/

public int getCacheSize(String cacheName);

### 트랜잭션캐쉬

#### 용도

* 트랜잭션캐쉬는 한 트랜잭션 내에서 동일한 데이터를 가지는 POJO 클래스 객체를 두개 이상 생성하지 않도록 하기위해 사용하는 캐쉬이다.
* 트랜잭션캐쉬는 트랜잭션이 종료되면 삭제되기 때문에 다음 트랜잭션에서는 사용할 수 없다.
* 트랜잭션캐쉬는 클래스 객체이든 DTO 객체이든 모든 데이터를 캐쉬할 수 있다.

#### 사용 방법

* 공통 베이스의 Utility인 ApplicationContextUtil를 사용하여 트랜잭션캐쉬에 저장한다.
* 트랜잭션캐쉬는 트랜잭션이 종료되면 자동 삭제되기 때문에 별도의 삭제 메소드가 없다.
* 트랜잭션캐쉬는 내부적으로 입력으로 받은 클래스명에 해당 객체를 자동으로 생성하기 때문에 외부에서 객체를 생성하지 않아야 한다.

@Override

public Tlr getTlr(String instCd,String deptId,String staffId) throws BizApplicationException {

TlrImpl tlr = (TlrImpl)ApplicationContextUtil.getUniqueBusinessBean(\_getUniqueKey(instCd,deptId,staffId),

"bankware.corebanking.accounting.cash.business.TlrImpl");

return tlr.getTlr(instCd,deptId,staffId) ? tlr : null;

}

위의 예제에서 TlrImpl 객체를 생성해서 반환한다.

#### 트랜잭션캐쉬 API

캐쉬에 객체 저장

/\*\*

\* 입력된 클래스명에 해당하는 객체를 생성하여 캐쉬에 저장

\*

\* @param key (required) 객체를 구별하기 위한 캐쉬키 값

\* @param className (required) 트랜잭션캐쉬에 저장할 클래스

\* @return 입력으로 받은 데이터를 반환한다.

\*/

public static Object getUniqueBusinessBean(String key, String className) throws BizApplicationException;

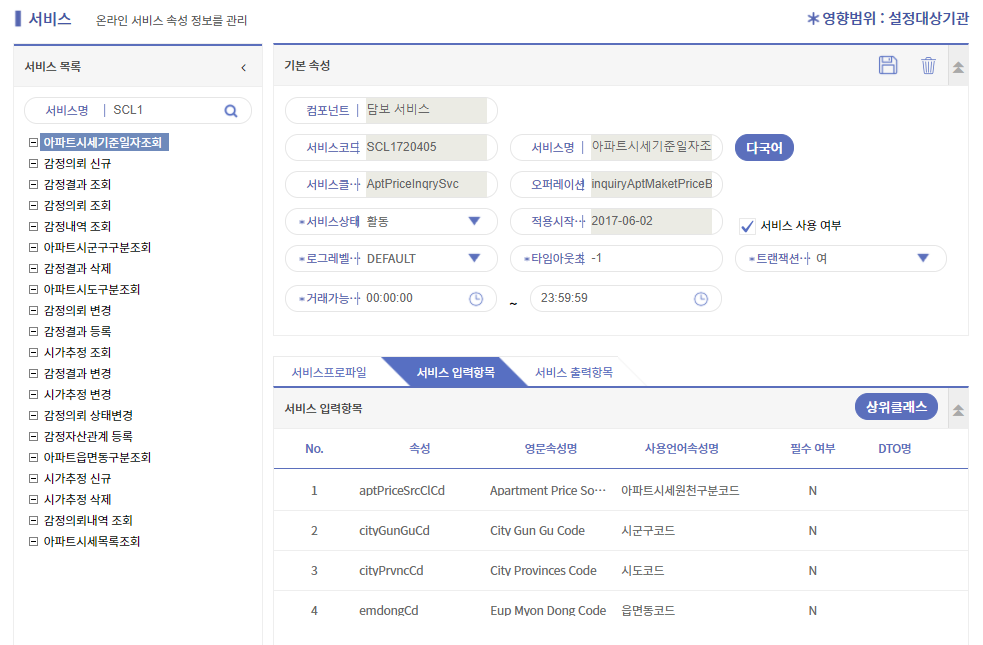
## 서비스 검증 및 테스트

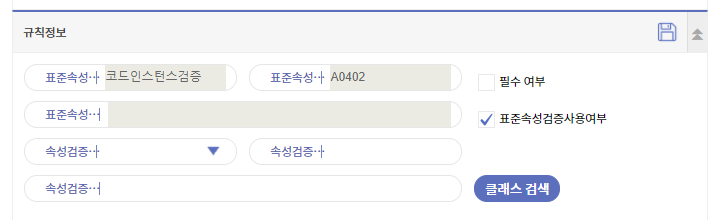
입력검증(Input Validation)이란 CBP서비스의 입력항목에 대하여 정당한 값이 들어왔는지 확인하는 처리절차를 의미한다.

CBP에서는 서비스 담당 프로그램의 개별 검증로직을 수행하지 않고, 서비스 입력DTO의 각 항목에 대한 검증규칙을 적용한다. 검증규칙을 사용하면, 재사용성 증가, 표준화, 검증규칙 확인 용이성을 확보할 수 있다. 기관별로 서로 다른 검증규칙을 설정할 수 있다.

### 서비스입력검증 규칙 설정

기본설정관리 > 서비스관리 > 서비스입력항목관리 > 서비스구성 에서 원하는 서비스를 선택하면 우측화면에 서비스명 및 입력 DTO명이 나타난다. (여기서는 SCL1720405이 서비스코드이고, AptMaketPriceBaseDtIn이 입력 DTO이다)





입력DTO의 속성 목록이 그리드에 나타나면, 특성 속성을 선택하여 검증규칙을 상세조회한다. 속성에 대한 검증규칙을 설정한 후 “저장” 버튼을 누른다. 검증규칙의 개별 항목에 대한 설명은 다음과 같다.

|  |  |
| --- | --- |
| 절차 | 설 명 |
| 표준속성검증사용여부 | * 서비스입력항목의 각 속성(Array와 같은 Collection 제외)은 표준속성이다 * 표준속성 자체의 검증규칙을 사용할지의 여부를 지정한다. Default값은 “여” * 표준속성검증사용여부를 “여”로 체크하면, 해당 표준속성의 속성검증규칙이 참고용으로 조회된다 |
| 필수여부 | * 해당 항목이 필수 입력항목인지 지정한다 |
| 속성검증방법코드 | * 속성검증방법을 코드화한 목록 중 하나를 선택. 2.3.1.2.2 “속성검증방법 설명 및 검증규칙 작성방법” 참조 |
| 속성검증규칙 | * 2.3.1.2.2 “속성검증방법 설명 및 검증규칙 작성방법” 참조 |

### 속성검증확장규칙

위의 화면에서 속성검증확장규칙 옆의 클래스 검색 버튼을 클릭하여 클래스레이어유형코드를 단순 DTO 검증, 복합 DTO 검증 중에서 선택한다.

|  |  |
| --- | --- |
| 절차 | 설 명 |
| 단순DTO검증클래스 | * 해당 속성에 대해서만 단순하게 검증하는 방식으로 아래와 같이 표기한다. * simple(ArrAcctNbrExstncVldtnImpl) |
| Complex 메소드 | * 해당 속성 뿐만 아니라 다양한 속성을 조합하여 검증하는 방식으로 아래와 같이 표기한다. 개별 복합DTO검증 클래스별로 필요한 파라미터가 다르므로 해당 복합DTO검증 클래스의 입력정보를 확인해야 한다 * complex(CntctPntCmpxVldtnImpl(cntctMthdTpCd,dtlCntctPntCntnt))   complex(CashTrnsfrVldtnImpl(cashTrnsfrDscd,prtlTrnsfrAmt,oprtnAcctNbr)) |

**참고사항**: 입력 항목의 추가 삭제는 서비스입력항목관리 화면에서 처리되지 않는다. 입력항목의 추가/삭제는 해당 서비스의 OMM(Object Message Mapping)클래스를 수정 또는 신규 개발하여 개발/운영 배포절차를 수행해야 한다.(해당 작업은 Jekins의 Run\_ClassInfoExtractor 작업을 통하여 수행된다) 신규추가되는 입력항목의 검증규칙은 default값이 적용된다.

**주의사항**: 서로 다른 서비스가 동일한 입력DTO(Data Transfrer Object)를 사용하는 경우, 검증규칙 변경은 해당 모든 서비스에 영향을 미치므로 주의해야 한다.

# Java Code Convention

Java Coding Convention은 Java를 이용하여 코딩을 하는 프로그래머 사이의 규칙이다.

소프트웨어를 개발하는 일련의 모든 과정에 들어가는 비용 중 80%가 유지보수에 사용되며, 소프트웨어를 직접 개발한 개발자가 그 소프트웨어의 유지보수를 담당하는 경우는 거의 보기 힘들다.

코딩 규칙을 지키면 다른 개발자가 그 소스 코드를 처음 보았을 때, 더 빠른 시간 안에 완벽하게 이해할 수 있도록 도와주기 때문에, 소프트웨어의 가독성이 높아진다.

*‘Java Coding Convention’ 관련되어 상세 내용은 개발표준정의서(Online) 문서와 함께 배포한 [AA\_AA\_A\_10\_개발표준정의서(JavaCode\_Conventions)\_Vx.xx\_yyyymmdd.docx] 문서를 참고한다.*

# SW보안약점진단가이드

'SW개발보안'은 SW개발과정에서 개발자의 실수, 논리적 오류 등으로 인해 발생될 수 있는 보안 취약점, 보안약점들을 최소화하여 사이버 보안 위협에 대응할 수 있는 안전한 SW를 개발하기 위한 일련의 보안 활동을 의미한다. 즉, SW개발 생명주기(SDLC, Software Development Life Cycle)의 각 단계별로 요구되는 보안활동을 수행함으로써 안전한 소프트웨어를 만들 수 있도록 한다.

설계단계에서 개발 보안을 적용할 때 얻을 수 있는 효과는 아래와 같다.

* 개발보안을 일관성 있게 적용
* 구현단계에서 개발보안 항목을 명확히 알 수 있음

*‘SW개발보안’ 관련되어 상세 내용 및 개발 시 참고해야 할 샘플은 개발표준정의서(Online) 문서와 함께 배포한 [한국인터넷진흥원\_소프트웨어\_개발보안\_가이드(2019.11)\_V1.00.pdf] 문서를 참고한다.*

## 설계단계 보안항목

### 입력데이터 검증 및 표현

사용자, 프로그램 입력 데이터에 대한 유효성 검증체계를 갖추고, 실패시 처리할 수 있도록 설계한다.

|  |  |  |
| --- | --- | --- |
| 보안약점 항목 | 설명 | 설계반영 업무 |
| SR1-1. DBMS 조회 및 결과 검증 | DBMS 조회를 위한 질의문(SQL) 생성시 사용되는 입력 값과 조회결과에 대한 검증방법(필터링 등) 설계 및 유효하지 않은 값에 대한 처리방법 설계 | 모든 업무팀 |
| SR1-2. XML조회 및 결과 검증 | XML 조회를 위한 질의문(XPath, XQuery 등) 생성시 사용되는 입력값과 조회결과에 대한 검증방법 (필터링 등) 설계 및 유효하지 않은 값에 대한 처리방법 설계 | 해당없음 |
| SR1‐3 디렉터리 서비스 조회 및 결과 검증 | 디렉터리 서비스 조회(LDAP 등)시 사용되는 입력값과 조회결과에 대한 검증방법(필터링 등) 설계 및 유효하지 않은 값에 대한 처리방법 설계 | SSO솔루션 |
| SR1‐4. 시스템 자원 접근 및 명령어 수행 입력값 검증 | 시스템 자원접근 및 명령어 수행을 위해 사용되는 입력 값에 대한 유효성 검증방법과 유효하지 않은 값에 대한 처리방법 설계 | F/W |
| SR1‐5. 웹 서비스 요청 및 결과 검증 | 웹 서비스(게시판 등) 요청(스크립트 게시 등)과 응답결과(스크립트 포함 웹 페이지 등)에 대한 검증 방법과 적절하지 않은 데이터에 대한 처리방법 설계 | 해당없음 |
| SR1‐6. 웹 기반 중요기능 수행요청 유효성 검증 | 사용자 권한확인(인증 등)이 필요한 중요기능(결제 등)에 대한 웹 서비스 요청에 대한 유효성 검증 방법과 유효하지 않은 요청에 대한 처리방법 설계 | F/W |
| SR1‐7. HTTP 프로토콜 유효성 검증 | 비정상적인 HTTP 헤더, 자동연결 URL 링크 등 사용자가 원하지 않은 결과를 생성할 수 있는 HTTP 헤더 및 응답결과에 대한 유효성 검증방법과 유효하지 않은 값에 대한 처리방법 설계 | F/W |
| SR1‐8. 허용된 범위내 메모리 접근 | 허용된 범위의 메모리 버퍼에만 접근하여 저장 또는 읽기가 수행되어 버퍼오버플로우가 발생하지 않도록 처리방법 설계 | 해당없음 |
| SR1‐9. 보안기능 동작에 사용되는 입력값 검증 | 보안기능(인증, 인가, 권한부여 등) 동작을 위해 사용되는 입력값과 함수(또는 메서드)의 외부입력 값 및 수행결과에 대한 처리방법 설계 | 업무총괄, F/W |
| SR1‐10. 업로드·다운로드 파일 검증 | 업로드·다운로드 파일의 무결성, 실행권한 등에 관한 유효성 검사 방법을 설계하고, 검사 실패시 대응 방안 설계 | EDMS |

### 보안기능

인증, 접근통제, 권한관리, 비밀번호 등의 정책이 적절하게 반영될 수 있도록 설계한다.

|  |  |  |
| --- | --- | --- |
| 보안약점 항목 | 설명 | 설계반영 업무 |
| SR2-1. 인증 대상 및 방식 | 중요정보·기능과 인증방식을 정의하고, 정의된 중요정보 접근 및 중요기능 수행 허용을 위해 인증 기능이 우회되지 않고 수행될 수 있도록 설계 | F/W, 업무총괄 |
| SR2-2. 인증 수행 제한 | 인증 반복시도 제한 및 인증실패 등에 대한 인증제한 기능 설계 | 업무총괄 |
| SR2‐3 비밀번호 관리 | 안전한 비밀번호 조합규칙(비밀번호 길이, 허용문자 조합 등)을 설정하고, 안전한 저장 정책, 재설정 및 변경 정책, 패스워드 관리규칙(주기적 변경 등)이 적용되도록 설계 | 업무총괄 |
| SR2‐4. 중요자원 접근통제 | 중요자원(프로그램 설정, 민감한 사용자 데이터 등)을 정의하고, 정의된 중요자원에 대한 접근을 통제 하는 신뢰할 수 있는 방법(권한관리 포함) 및 접근통제 실패시 대응방안 설계 | 업무총괄 |
| SR2‐5. 암호키 관리 | 암호키 생성, 분배, 접근, 파기 등 안전하게 암호키 생명주기를 관리할 수 있는 방법 설계 | DB암호화솔루션 |
| SR2‐6. 암호연산 | 국제표준 또는 검증필 프로토콜로 등재된 안전한 암호 알고리즘을 선정하여 충분한 암호키 길이, 솔트, 충분한 난수값을 기반으로 암호연산 수행방법 설계 | DB암호화솔루션 |
| SR2‐7. 중요정보 저장 | 중요정보(비밀번호, 개인정보 등) 저장시 안전한 저장 및 관리방법 설계 | 업무총괄 |
| SR2‐8. 중요정보 전송 | 중요정보(비밀번호, 개인정보 등) 전송시 안전한 전송방법 설계 | 업무총괄 |

### 에러처리

에러 또는 오류상황을 처리하지 않거나 불충분하게 처리되어 중요정보 유출 등 보안약점이 발생하지 않도록 설계한다.

|  |  |  |
| --- | --- | --- |
| 보안약점 항목 | 설명 | 설계반영 업무 |
| SR3-1. 예외처리 | 오류메시지에 중요정보(개인정보, 시스템 정보, 민감 정보 등)가 포함되어 출력되거나, 에러 및 오류가 부적절하게 처리되어 의도치 않은 상황이 발생하는 것을 막기 위한 안전한 방안 설계 | F/W |

### 세션통제

HTTP를 이용하여 연결을 유지하는 경우 세션을 안전하게 할당하고 관리하여 세션정보노출이나 세션 하이재킹과 같은 침해사고가 발생하지 않도록 설계한다.

|  |  |  |
| --- | --- | --- |
| 보안약점 항목 | 설명 | 설계반영 업무 |
| SR4-1. 세션통제 | 다른 세션간 데이터 공유금지, 세션 ID 노출금지, (재)로그인시 세션ID 변경, 세션종료(비활성화, 유효기간 등) 처리 등 세션을 안전하게 관리할 수 있는 방안 설계 | F/W |

## 구현단계 SW보안약점 항목

### 입력데이터 검증 및 표현

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | SQL 삽입 | 검증되지 않은 외부입력값이 SQL 쿼리문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안약점 |
| 2 | 경로 조작 및 자원 삽입 | 검증되지 않은 외부입력값이 시스템 자원 접근경로 또는 자원제어에 사용되어 공격자가 입력값을 조작해 공격할 수 있는 보안약점 |
| 3 | 크로스사이트 스크립트 | 검증되지 않은 외부입력값에 의해 사용자 브라우저에서 악의적인 스크립트가 실행될 수 있는 보안약점 |
| 4 | 운영체제 명령어 삽입 | 검증되지 않은 외부입력값이 운영체제 명령문 생성에 사용되어 악의적인 명령어가 실행될 수 있는 보안약점 |
| 5 | 위험한 형식 파일 업로드 | 파일의 확장자 등 파일형식에 대한 검증 없이 업로드를 허용하여 발생할 수 있는 보안약점 |
| 6 | 신뢰되지 않는 URL 주소로 자동접속 연결 | 검증되지 않은 외부입력값이 URL 링크 생성에 사용되어 악의적인 사이트로 자동 접속될 수 있는 보안약점 |
| 7 | XQuery 삽입 | 검증되지 않은 외부입력값이 XQuery 쿼리문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안약점 |
| 8 | XPath 삽입 | 검증되지 않은 외부입력값이 XPath 쿼리문 생성에 사용 되어 악의적인 쿼리가 실행될 수 있는 보안약점 |
| 9 | LDAP 삽입 | 검증되지 않은 입력값이 LDAP 명령문 생성에 사용되어 악의적인 명령어가 실행될 수 있는 보안약점 |
| 10 | 크로스사이트 요청 위조 | 검증되지 않은 외부입력값에 의해 브라우저에서 악의적인 스크립트가 실행되어 공격자가 원하는 요청(Request)이 다른 사용자(관리자 등)의 권한으로 서버로 전송되는 보안약점 |
| 11 | HTTP 응답분할 | 검증되지 않은 외부입력값이 HTTP 응답헤더에 삽입되어 악의적인 코드가 실행될 수 있는 보안약점 |
| 12 | 정수형 오버플로우 | 정수를 사용한 연산의 결과가 정수값의 범위를 넘어서는 경우, 프로그램이 예기치 않게 동작될 수 있는 보안약점 |
| 13 | 보안기능 결정에 사용되는 부적절한 입력값 | 검증되지 않은 입력값이 보안결정(인증, 인가, 권한부여 등)에 사용되어 보안 메커니즘 우회 등을 야기할 수 있는 보안약점 |
| 14 | 메모리 버퍼 오버플로우 | 메모리 버퍼의 경계값을 넘어서 메모리값을 읽거나 저장하여 예기치 않은 결과를 발생시킬 수 있는 보안약점 |
| 15 | 포맷 스트링 삽입 | printf 등 외부입력값으로 포맷스트링을 제어할 수 있는 함수를 사용하여 발생 할 수 있는 보안약점 |

### 보안기능

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 적절한 인증 없는 중요 기능 허용 | 적절한 인증 없이 중요정보(금융정보, 개인정보, 인증정보 등)를 열람(또는 변경)할 수 있게 하는 보안약점 |
| 2 | 부적절한 인가 | 적절한 접근제어 없이 외부입력값을 포함한 문자열로 중요자원에 접근할 수 있는 보안약점 |
| 3 | 중요한 자원에 대한 잘못된 권한 설정 | 중요자원(프로그램 설정, 민감한 사용자 데이터 등)에 대한 적절한 접근권한을 부여하지 않아, 인가되지 않은 사용자 등에 의해 중요정보가 노출·수정되는 보안약점 |
| 4 | 취약한 암호화 알고리즘 사용 | 중요정보(금융정보, 개인정보, 인증정보 등)의 기밀성을 보장할 수 없는 취약한 암호화 알고리즘을 사용하여 정보가 노출될 수 있는 보안약점 |
| 5 | 중요정보 평문저장 | 중요정보(비밀번호, 개인정보 등)를 암호화하여 저장하지 않아 정보가 노출될 수 있는 보안약점 |
| 6 | 중요정보 평문전송 | 중요정보(비밀번호, 개인정보 등) 전송시 암호화하지 않거나 안전한 통신채널을 이용하지 않아 정보가 노출될 수 있는 보안약점 |
| 7 | 하드코드된 비밀번호 | 소스코드 내에 비밀번호가 하드코딩 되어 소스코드 유출시 노출 우려 및 주기적 변경 등 수정(관리자 변경 등)이 용이하지 않는 보안약점 |
| 8 | 충분하지 않은 키 길이 사용 | 데이터의 기밀성, 무결성 보장을 위해 사용되는 키의 길이가 충분하지 않아 기밀정보 누출, 무결성이 깨지는 보안약점 |
| 9 | 적절하지 않은 난수 값 사용 | 예측 가능한 난수사용으로 공격자로 하여금 다음 숫자 등을 예상하여 시스템 공격이 가능한 보안약점 |
| 10 | 하드코드된 암호화 키 | 소스코드 변경이 용이하지 내에 암호화키가 않는 보안약점 하드코딩 되어 소스코드 유출시 노출 우려 및 키 변경이 용이하지 않는 보안약점 |
| 11 | 취약한 비밀번호 허용 | 비밀번호 조합규칙(영문, 숫자, 특수문자 등) 미흡 및 길이가 충분하지 않아 노출될 수 있는 보안약점 |
| 12 | 사용자 하드디스크에 저장되는 쿠키를 통한 정보노출 | 쿠키(세션 ID, 사용자 권한정보 등 중요정보)를 사용자 하드디스크에 저장함으로써 개인정보 등 기밀정보가 노출될 수 있는 보안약점 |
| 13 | 주석문 안에 포함된 시스템 주요정보 | 소스코드내의 주석문에 인증정보 등 시스템 주요정보가 포함되어 소스코드 유출시 노출될 수 있는 보안약점 |
| 14 | 솔트 없이 일방향 해시 함수 사용 | 공격자가 솔트 없이 생성된 해시값을 얻게 된 경우, 미리 계산된 레인보우 테이블을 이용하여 원문을 찾을 수 있는 보안약점 |
| 15 | 무결성 검사 없는 코드 다운로드 | 원격으로부터 소스코드 또는 실행파일을 무결성 검사 없이 다운로드 받고 이를 실행하는 경우, 공격자가 악의적인 코드를 실행할 수 있는 보안약점 |
| 16 | 반복된 인증시도 제한 기능 부재 | 인증시도의 수를 제한하지 않아 공격자가 무작위 인증시도를 통해 계정접근 권한을 얻을 수 있는 보안약점 |

### 시간 및 상태

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 경쟁조건 : 검사 시점과 사용 시점(TOCTOU) | 멀티 프로세스 상에서 자원을 검사하는 시점과 사용하는 시점이 달라서 발생하는 보안약점 |
| 2 | 종료되지 않는 반복문 또는 재귀함수 | 종료조건 없는 제어문 사용으로 반복문 또는 재귀함수가 무한히 반복되어 발생할 수 있는 보안약점 |

### 에러처리

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 오류 메시지를 통한 정보 노출 | 개발자가 생성한 오류 메시지에 시스템 내부구조 등이 포함되어 민감한 정보가 노출될 수 있는 보안약점 |
| 2 | 오류 상황 대응 부재 | 시스템에서 발생하는 오류 상황을 처리하지 않아 프로그램 실행정지 등 의도하지 않은 상황이 발생할 수 있는 보안약점 |
| 3 | 부적절한 예외 처리 | 예외에 대한 부적절한 처리로 인해 의도하지 않은 상황이 발생될 수 있는 보안 약점 |

### 코드오류

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | Null Pointer 역참조 | Null로 설정된 변수의 주소값을 참조했을 때 발생하는 보안약점 |
| 2 | 부적절한 자원 해제 | 사용된 자원을 적절히 해제하지 않으면 자원 누수 등이 발생하고, 자원이 부족하여 새로운 입력을 처리할 수 없게 되는 보안약점 |
| 3 | 해제된 자원 사용 | 메모리 등 해제된 자원을 참조하여 예기치 않은 오류가 발생될 수 있는 보안약점 |
| 4 | 초기화되지 않은 변수 사용 | 변수를 초기화하지 않고 사용하여 예기치 않은 오류가 발생될 수 있는 보안약점 |

### 캡슐화

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | 잘못된 세션에 의한 데이터 정보노출 | 잘못된 세션에 의해 인가되지 않은 사용자에게 중요정보가 노출될 수 있는 보안약점 |
| 2 | 제거되지 않고 남은 디버그 코드 | 디버깅을 위해 작성된 코드를 통해 인가되지 않은 사용자에게 중요정보가 노출될 수 있는 보안약점 |
| 3 | 시스템 데이터 정보노출 | 사용자가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되는 보안약점 |
| 4 | Public 메서드로부터 반환된 Private 배열 | Private로 선언된 배열을 Public으로 선언된 메서드를 통해 반환(return)하면, 그 배열의 레퍼런스가 외부에 공개되어 외부에서 배열이 수정될 수 있 는 보안약점 |
| 5 | Private 배열에 Public 데이터 할당 | Public으로 선언된 데이터 또는 메서드의 파라미터가 Private로 선언된 배 열에 저장되면, Private 배열을 외부에서 접근할 수 있게 되는 보안약점 |

### API오용

|  |  |  |
| --- | --- | --- |
| No. | 보안약점 | 설명 |
| 1 | DNS lookup에 의존한 보안결정 | DNS는 공격자에 의해 DNS 스푸핑 공격 등이 가능하므로 보안 결정을 DNS 이름에 의존할 경우, 보안결정 등이 노출되는 보안약점 |
| 2 | 취약한 API 사용 | 취약하다고 알려진 함수를 사용함으로써 예기치 않은 보안위협에 노출될 수 있는 보안약점 |

## 개인정보의 안정성 확보조치

### 개요

SEMA 차세대 정보시스템 구축 시, ‘개인정보보호법’과 ‘개인정보의 안정성 확보조치 기준’에 따라 중요정보는 암호화하여 별도 관리한다.

* 중요정보 유형 및 관리 방식

|  |  |  |  |
| --- | --- | --- | --- |
| 유형 | 관리대상 | 암호화 방식 | 비고 |
| 비밀번호 | 비밀번호 | SHA-256 방식  일방향 암호화 수행 | ‘소프트웨어 개발보안 가이드’의 ‘3-2.3 비밀번호 관리’를 참고하여 안전하게 생성하도록 설계 |
| 개인식별정보 | 주민번호  계좌번호  법인카드아이디 | SHA-256방식  양방향 암호화 수행 | Meta 표준단어를 바탕으로 Framework에서 자동 암/복호화를 수행함. 개발자 별도 조치 필요 없음 |

모든 중요정보의 접근에 대해서는 별도 접근기록 보관을 해야 한다.

### 개인정보를 고려한 개발표준

* 서비스의 입력/출력으로 사용되는 항목은 Meta에 등록한 단어만을 사용한다.
* 표준 단어 중 개인정보와 관련된 단어는 별도 처리 유형을 BXM에서 관리한다.
* 서비스 개발 시, BXM Framework에서 Meta에 등록한 단어를 이용하여 자동으로 개인정보 안정성 확보 조치를 수행함에 따라 업무 개발자가 별도 처리는 하지 않아도 된다.

# 샘플 프로그램 설명

Online 소스의 표준 템플릿을 만들기 위해 자주 사용하는 기술적인 유형을 정하여 임의의 업무를 바탕으로 선도개발을 수행하였다. 선정된 유형은 다음과 같다.

- 단건처리

- 다건처리

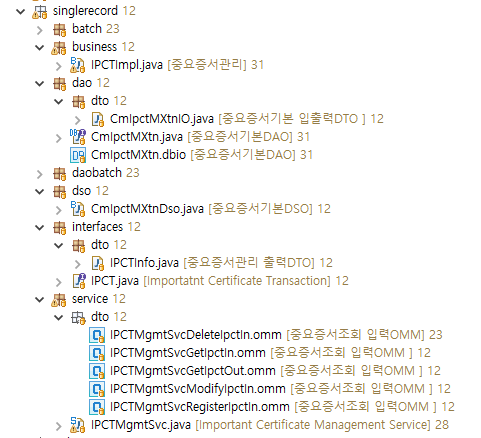
- 다건조회(페이징)

- 복합처리

해당 샘플 프로그램들 중에서 개발 시 주의해야 할 부분을 설명하였다.

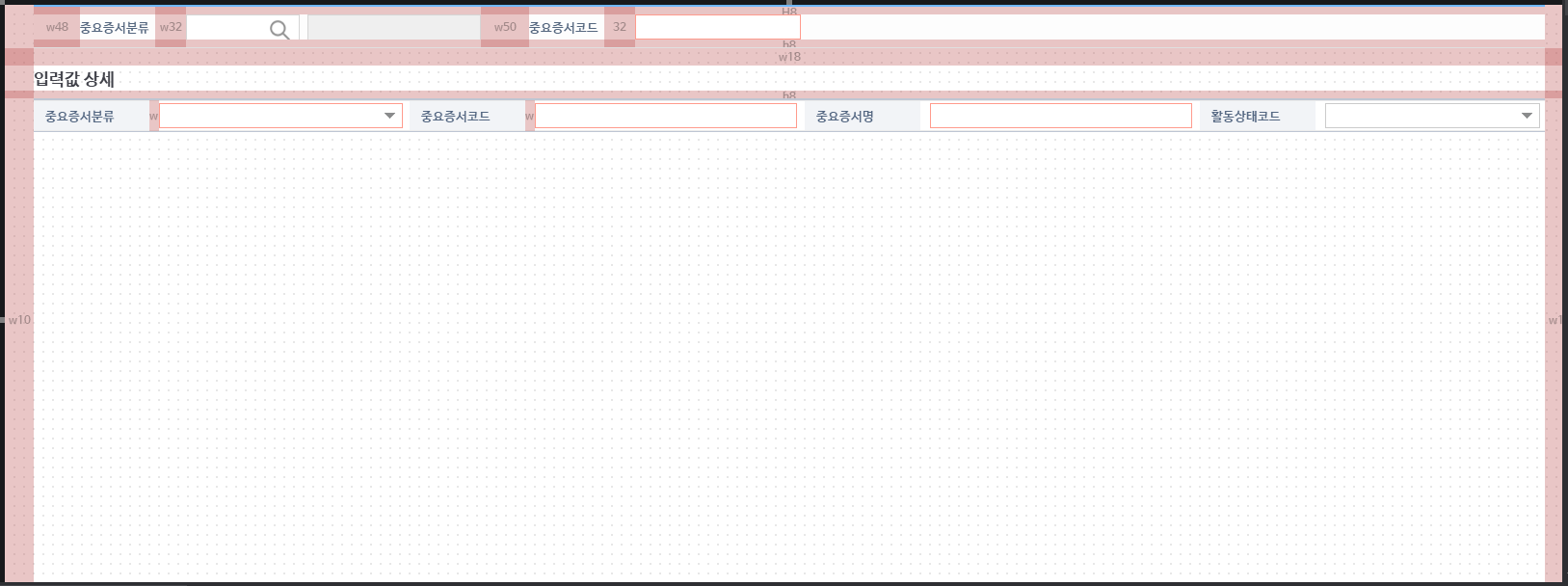
## 단건 처리

하나의 Record를 조회하고 등록/수정/삭제를 수행하는 유형이다. 임의로 ‘중요증서 관리’라는 업무를 통해 샘플을 구성하였다. ‘IPCTMgmtSvc’ 서비스 클래스에서 총 4개의 서비스를 제공하도록 설계하였다.



### 화면

화면은 아래와 같이 개발되었으며, 상단 ‘조회’, ‘저장’, ‘삭제’ 버튼으로 거래를 호출하도록 처리하였다. ‘저장’ 버튼 클릭 시, UI에서 ‘수정’ 또는 ‘신규’ 서비스를 판단하여 호출하도록 처리하였다.



### 서비스

서비스는 ‘IPCTMgmtSvc’ 클래스에서 ‘조회’, ‘신규’, ‘수정’, ‘삭제’ 서비스를 모두 처리하였다. 서비스 클래스의 구성은 아래와 같은 패턴으로 구성하였다.

// 1) Package 선언

package bankware.corebanking.leadingdevelopment.singlerecord.service;

// 2) Import 선언

import bankware.corebanking.leadingdevelopment.singlerecord.interfaces.IPCT;

…

// 3) 클래스 주석

/\*\*

\* <pre>

\*

\* 본 클래스는 중요증서와 관련된 처리를 담당한다.

\* - 중요증서 조회, 등록, 수정, 삭제

\* …

\* </pre>

\*/

// 4) 클래스 annotation 선언 및 클래스 정의

@BxmService("IPCTMgmtSvc")

@BxmCategory(logicalName = "Important Certificate Management Service")

public class IPCTMgmtSvc {

// 5) 멤버 객체 선언

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private CmnContext cmnContext;

private IPCT ipct; // Important Certificate Management

// 6) 서비스 메소드 annotation 선언 및 메소드 정의

@BxmServiceOperation("getIpct")

@CbbSrvcInfo(srvcCd="SLDTMP0001001",srvcNm="getIpct")

public IPCTMgmtSvcGetIpctOut getIpct(IPCTMgmtSvcGetIpctIn in) throws BizApplicationException {

// 6-1) Output 객체 선언

IPCTMgmtSvcGetIpctOut out = new IPCTMgmtSvcGetIpctOut();

// 6-2) Base API 호출 객체 조립

IPCTInfo ipctInfoIn = new IPCTInfo();

ipctInfoIn.setInstCd(\_getCmnContext().getInstCode());

…

// 6-3) Base API 호출

// Base API호출

IPCTInfo ipctInfoOut = new IPCTInfo();

ipctInfoOut = \_getIpct().getIpct(ipctInfoIn);

// 6-4) 결과가 null이 아니면 Output객체 조립

if (ipctInfoOut != null) {

out.setIpctCd(ipctInfoOut.getIpctCd());

…

}

// 6-5) Output 객체 리턴

return out;

}

// 7) 신규 등록 서비스 구현

@BxmServiceOperation("registerIpct")

@CbbSrvcInfo(srvcCd="SLDTMP0001002",srvcNm="registerIpct")

public DummyIO registerIpct(IPCTMgmtSvcRegisterIpctIn in) throws BizApplicationException {

// 7-1) Output 객체 선언 – 등록/수정/삭제는 DummyIO를 리턴

DummyIO out = new DummyIO();

IPCTInfo ipctInfo = new IPCTInfo();

ipctInfo.setInstCd(\_getCmnContext().getInstCode());

…

// 7-2) Base API 호출 – 등록/수정/삭제의 Base결과값은 Integer 형식임 (처리 건수)

// Base API호출

int cnt = \_getIpct().registerIpct(ipctInfo);

return out;

}

// 8) 수정 서비스 구현

@BxmServiceOperation("modifyIpct")

@CbbSrvcInfo(srvcCd="SLDTMP0001003",srvcNm="modifyIpct")

public DummyIO modifyIpct(IPCTMgmtSvcModifyIpctIn in) throws BizApplicationException {

DummyIO out = new DummyIO();

IPCTInfo ipctInfo = new IPCTInfo();

ipctInfo.setInstCd(\_getCmnContext().getInstCode());

…

// Base API호출

int cnt = \_getIpct().modifyIpct(ipctInfo);

return out;

}

// 9) 삭제 서비스 구현

@BxmServiceOperation("deleteIpct")

@CbbSrvcInfo(srvcCd="SLDTMP0001004",srvcNm="deleteIpct")

public DummyIO deleteIpct(IPCTMgmtSvcDeleteIpctIn in) throws BizApplicationException {

DummyIO out = new DummyIO();

IPCTInfo ipctInfo = new IPCTInfo();

ipctInfo.setInstCd(\_getCmnContext().getInstCode());

…

// Base API호출

int cnt = \_getIpct().deleteIpct(ipctInfo);

return out;

}

// 10) public 메소드를 먼저 구현한 후, private 메소드는 뒤에 작성한다.

// 11) Base API I/F 호출을 위한 getBean 구문을 서비스 제일 하단에 기술한다.

/\*\*

\* @return the cmnContext

\*/

private CmnContext \_getCmnContext() {

if (cmnContext == null) {

cmnContext = (CmnContext) CbbApplicationContext.getBean(CmnContext.class, cmnContext);

}

return cmnContext;

}

/\*\*

\* @return the Important Certificate Management

\*/

private IPCT \_getIpct() {

if (ipct == null) {

ipct = (IPCT) CbbApplicationContext.getBean(IPCT.class, ipct);

}

return ipct;

}

}

### 베이스

베이스는 ‘IPCT’인터페이스의 구현체인 IPCTImpl 클래스에서 ‘조회’, ‘신규’, ‘수정’, ‘삭제’ 기능을 제공하였다. 단순한 CRUD를 제공하는 베이스 클래스이기에 내부에 복잡한 로직이 포함되거나 하지는 않았다.

// 1) Package 선언

package bankware.corebanking.leadingdevelopment.singlerecord.business;

// 2) Import 선언

import bankware.corebanking.leadingdevelopment.singlerecord.dao.dto.CmIpctMXtnIO;

…

// 3) 클래스 주석

/\*\*

\* <pre>

\* This bean sample program provides common code management function.

…

\* </pre>

\*/

// 4) 클래스 annotation 선언 및 클래스 정의

@BxmBean

@BxmCategory(logicalName = "중요증서관리", description = "중요증서에 대한 관리를 한다.")

@CbbClassInfo(classType={"SERVICE\_BO"})

public class IPCTImpl implements IPCT {

// 5) 멤버 객체 선언

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private CmIpctMXtnDso cmIpctMXtnDso; // Important Certificate Business

private CmnContext cmnContext;

// 6) 메소드 annotation 선언 및 메소드 정의

@Override

public IPCTInfo getIpct(IPCTInfo in) throws BizApplicationException {

// 6-1) DSO 호출

CmIpctMXtnIO cmIpctMXtnIO = \_getCmIpctMXtnDso().select(\_setCmIpctMIO(in));

return \_setIPCTInfo(cmIpctMXtnIO);

}

// 7) 신규 등록 기능 구현

@Override

public int registerIpct(IPCTInfo in) throws BizApplicationException {

return \_getCmIpctMXtnDso().insert(\_setCmIpctMIO(in));

}

// 8) 수정 기능 구현

@Override

public int modifyIpct(IPCTInfo in) throws BizApplicationException {

return \_getCmIpctMXtnDso().update(\_setCmIpctMIO(in));

}

// 9) 삭제 기능 구현

@Override

public int deleteIpct(IPCTInfo in) throws BizApplicationException {

return \_getCmIpctMXtnDso().delete(\_setCmIpctMIO(in));

}

// 10) public 메소드를 먼저 구현한 후, private 메소드는 뒤에 작성한다.

private CmIpctMXtnIO \_setCmIpctMIO(IPCTInfo in) {

CmIpctMXtnIO out = new CmIpctMXtnIO();

if(in != null) {

out.setInstCd(in.getInstCd());

…

\_getCmnContext().setHeaderColumn(out);

}

return out;

}

private IPCTInfo \_setIPCTInfo(CmIpctMXtnIO in) {

IPCTInfo out = new IPCTInfo();

if(in != null) {

out.setInstCd(in.getInstCd());

…

}

return out;

}

// 11) Base API I/F 호출을 위한 getBean 구문을 서비스 제일 하단에 기술한다.

/\*\*

/\*\*

\* @return the ipctMgmtBiz

\*/

private CmIpctMXtnDso \_getCmIpctMXtnDso() {

if (cmIpctMXtnDso == null) {

cmIpctMXtnDso = (CmIpctMXtnDso) CbbApplicationContext.getBean(CmIpctMXtnDso.class, cmIpctMXtnDso);

}

return cmIpctMXtnDso;

}

/\*\*

\* @return the cmnContext

\*/

private CmnContext \_getCmnContext() {

if (cmnContext == null) {

cmnContext = (CmnContext) CbbApplicationContext.getBean(CmnContext.class, cmnContext);

}

return cmnContext;

}

}

### DSO

대부분의 DSO 클래스는 단순 DAO를 단순 매핑하는 수준에서 처리된다. 특정 Query의 활용도가 높으며 자주 변경되는 정보가 아닌 경우에는 캐쉬를 사용하여 성능을 높일 수 있다. 이때 Query에 대한 캐쉬 기능은 DSO에서 수행토록 한다.

package bankware.corebanking.leadingdevelopment.singlerecord.dso;

import bankware.corebanking.applicationcommon.importantcertificate.dao.dto.CmIpctMIO;

…

@BxmBean

@SuppressWarnings({ "all" })

@BxmCategory(logicalName = "중요증서기본DSO", description = "중요증서기본DSO")

public class CmIpctMXtnDso {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private CmIpctMXtn cmIpctMXtn;

@BxmCategory(description = "중요증서 단건 조회")

public CmIpctMXtnIO select(CmIpctMXtnIO cmIpctMIO) {

return \_getCmIpctM().select(cmIpctMIO);

}

@BxmCategory(description = "중요증서 신규 등록")

public int insert(CmIpctMXtnIO cmIpctMIO) {

return \_getCmIpctM().insert(cmIpctMIO);

}

@BxmCategory(description = "중요증서 수정")

public int update(CmIpctMXtnIO cmIpctMIO) {

return \_getCmIpctM().update(cmIpctMIO);

}

@BxmCategory(description = "중요증서 삭제")

public int delete(CmIpctMXtnIO cmIpctMIO) {

return \_getCmIpctM().delete(cmIpctMIO);

}

/\*\*

\* @return the cmIpctM

\*/

private CmIpctMXtn \_getCmIpctM() {

if (cmIpctMXtn == null) {

cmIpctMXtn = (CmIpctMXtn) CbbApplicationContext.getBean(CmIpctMXtn.class, cmIpctMXtn);

}

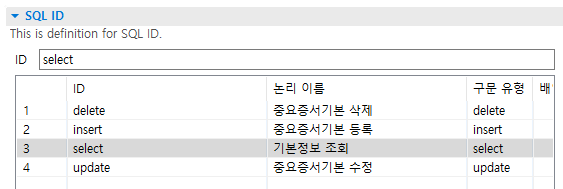
return cmIpctMXtn;

}

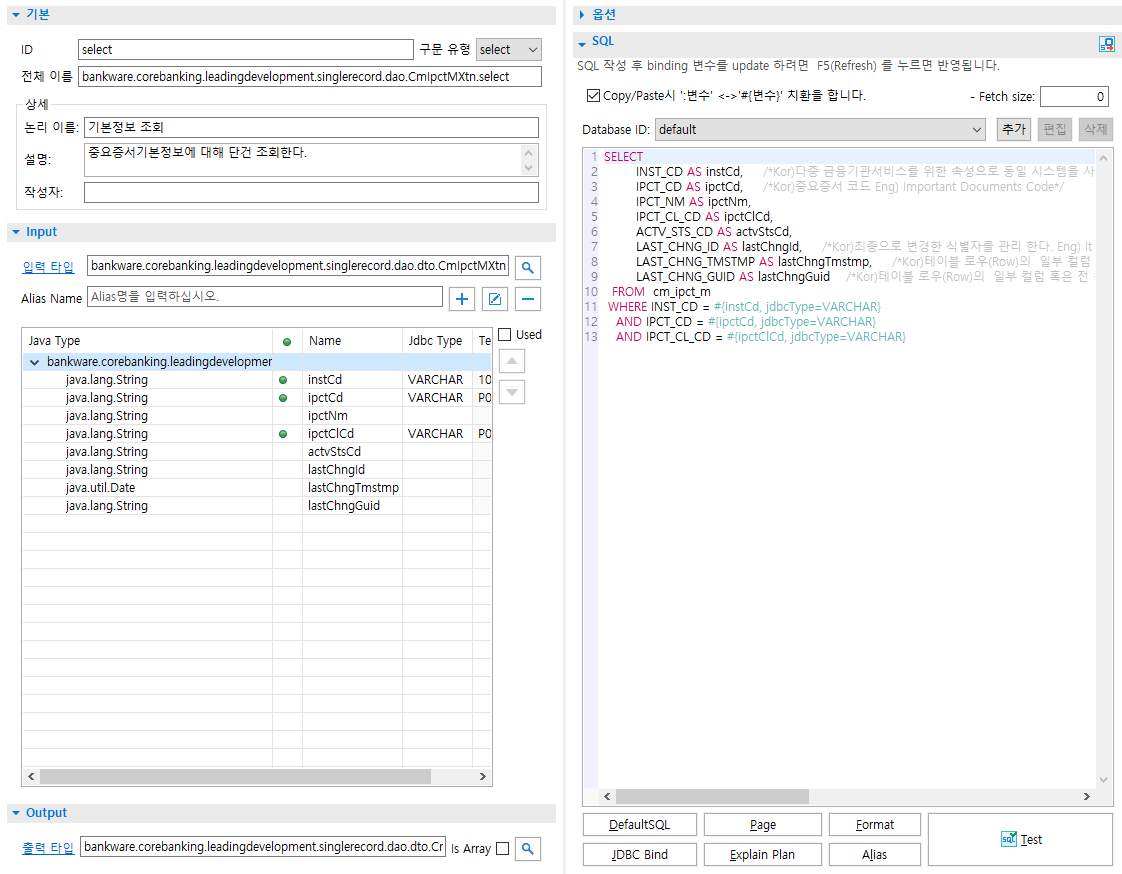
}

### DAO

단건처리를 위한 조회/등록/수정/삭제에 대한 Query를 작성하였다.



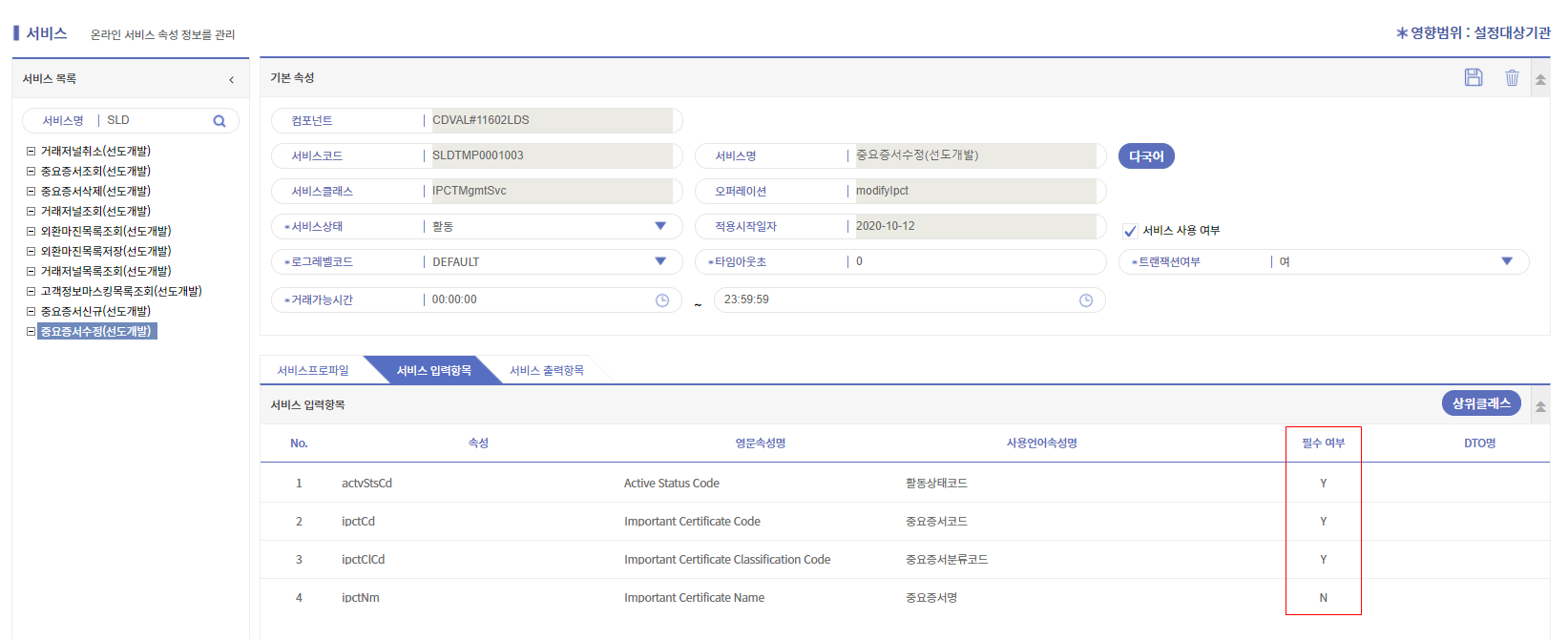
조회 select 예시)



### 중요 전달 사항

#### [서비스] 입력값의 필수 체크는 서비스 프로그램이 아닌 CP의 설정으로 처리

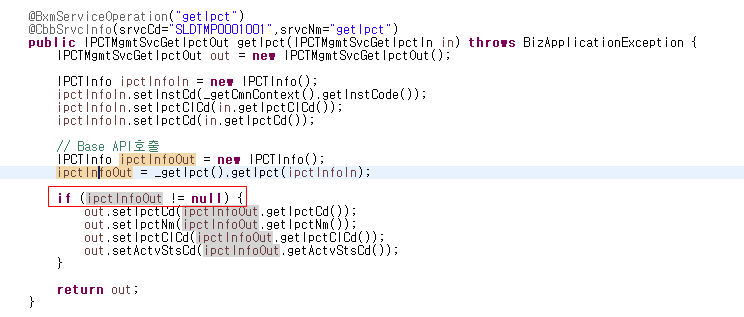
* 일반적으로 서비스의 최초 인입 시점에 Input 값에 대한 validation check를 수행함.
* BX-CBP 기반의 서비스를 구현할 때에는 서비스 입력값 체크는 소스가 아닌 CP의 서비스정보에서 수행하는 것을 표준으로 함.
* CP > 시스템운영 > 온라인서비스 > 서비스 화면 에서 해당 서비스를 선택. [서비스입력항목] Tab





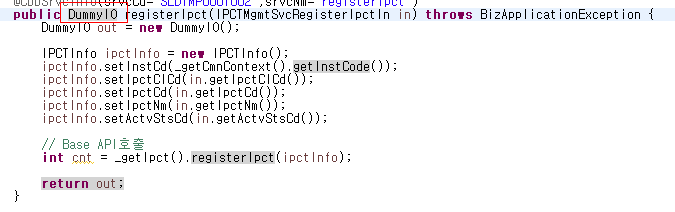
#### [서비스] Base 기능 호출 후 리턴 객체에 대한 null Check 필요

* 일반적으로 Base에서 특별한 처리를 하지 않는 상황에서 조회 결과가 없는 경우에는 리턴 객체가 null로 리턴된다. 이 부분을 간과하고 서비스에서 UI로 값을 get/set 처리 하면 오류가 발생할 수 있다.
* 리턴 객체를 처리하기 전에 null check를 추가하는 구문을 반드시 추가하도록 한다.



#### [서비스] 리턴값을 전달할 필요가 없는 등록/수정/삭제 서비스의 출력은 DummyIO를 이용

* 일반적으로 등록/수정/삭제인 경우 오류가 발생하지 않으면 단말로 리턴값을 던져줄 필요가 없다.

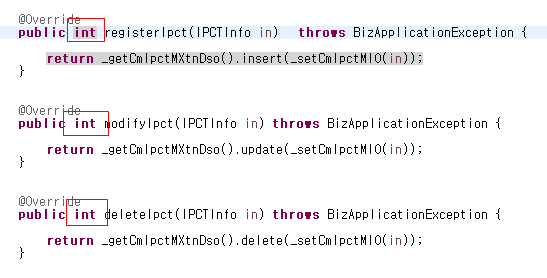


#### [베이스] 베이스의 API를 특정 서비스에서만 수행된다면 입력값 검증은 생략

* 베이스 API는 2가지 성격이 있을 수 있다.  
  1) 특정 서비스를 위해 생성한 베이스 기능 (대부분의 업무 베이스 클래스)  
  2) 다수의 서비스/베이스를 위해 생성한 베이스 기능 (대부분의 공통 클래스)
* 이때, 대부분의 업무 베이스인 (1)의 경우에는 베이스의 입력값 검증은 생략한다.  
  - 서비스의 입력값 검증을 믿고 베이스에서 추가로 검증하는 행위는 생략토록 한다.

#### [베이스] 등록/수정/삭제 베이스 API의 출력은 int 를 사용

* 등록/수정/삭제 기능을 제공하는 Base API의 출력은 반영 건수를 나타내는 int를 사용한다. 서비스에서 반영된 결과를 바탕으로 별도 처리 흐름을 제어할 수 있기 때문이다.

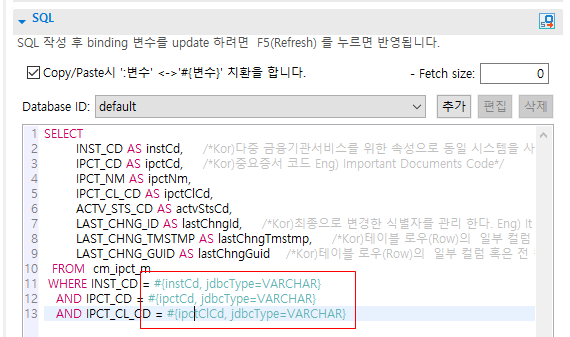


#### [DSO] Core에서 사용하는 테이블을 Site Base에서도 사용할 때는 “Xtn” 을 클래스명에 추가

* BX-CBP에서 사용하는 모든 클래스명은 중복을 허용하지 않고 있다. 중복이 발생할 수 있는 구간에 대해서는 Naming 명명표준으로 중복의 발생하지 않도록 처리하고 있다.
* DSO/DAO 클래스명은 테이블에 1:1 매핑 관계를 가진다.  
  예) CM\_IPCT\_M 🡺 CmIpctMDso, CmIpctM
* Core에서 이미 사용하고 있는 테이블을 Site Base에서도 사용하고자 할 때는 DSO/DAO 클래스를 아래와 같이 “Xtn” Keyword를 추가하여 명명한다. 이때, DAO에서 사용하는 VO들에 대해서도 “Xtn” Keyword를 추가하여 관리하도록 한다.  
  예) CM\_IPCT\_M 🡺 CmIpctM**Xtn**Dso, CmIpctM**Xtn**, CmIpctM**Xtn**IO

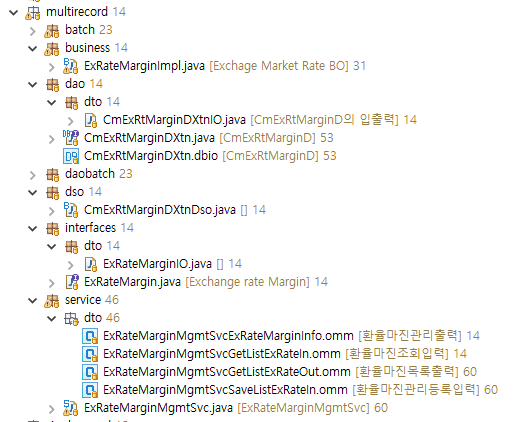
#### [DAO] Query의 변수에는 jdbcType을 지정

* jdbcType이 지정되지 않은 DAO 파라미터값에 null 값이 매핑되면 SQL에러가 발생한다. 시스템 에러가 발생하지 않도록 jdbcType을 항상 지정하도록 한다.



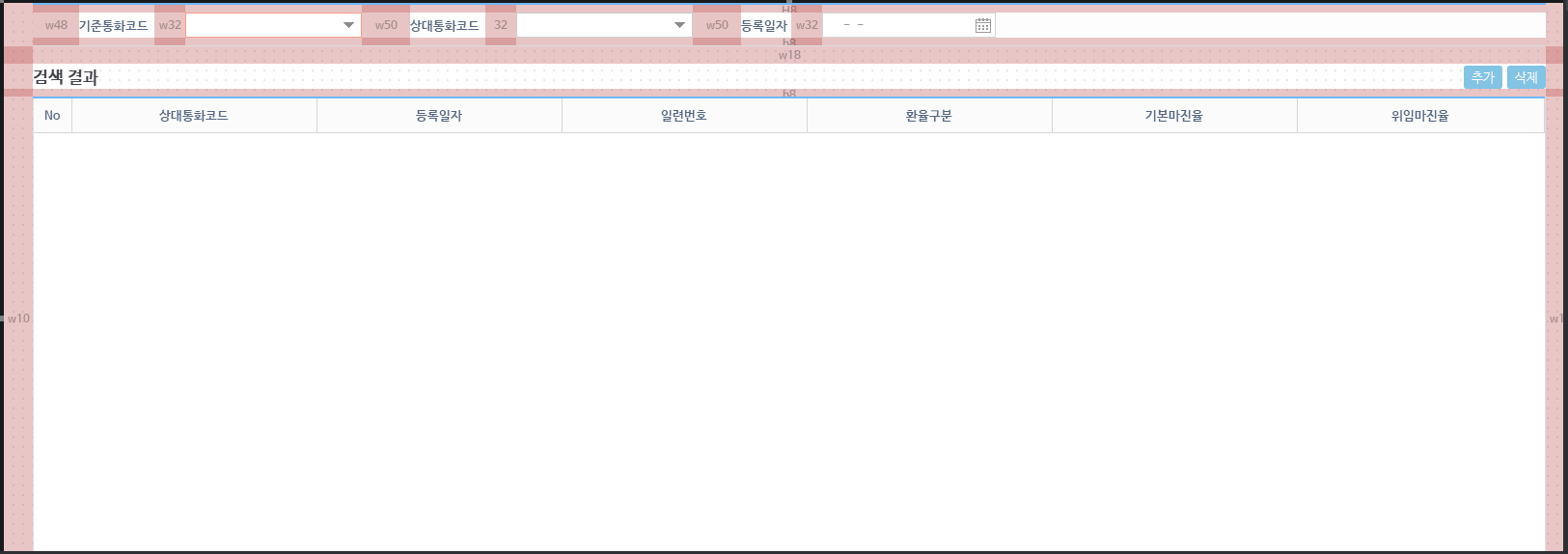
## 다건 처리

다수의 Record를 한번에 조회한 후, 화면의 그리드를 통해 일괄 등록/수정/삭제를 수행하는 유형이다. 임의로 ‘환율마진 관리’라는 업무를 통해 샘플을 구성하였다. ‘ExRateMarginMgmtSvc’ 서비스 클래스에서 총 2개의 서비스를 제공하도록 설계하였다.



### 화면

조회의 결과는 그리드로 표현된다. 이때, 그리드에서 ‘행추가’, ‘행삭제’를 통해 다수의 레코드를 한번에 저장할 수 있도록 처리하였다.



### 서비스

목록을 조회하는 것 이외의 앞서 설명한 단건 처리 서비스와 차별성은 없다. 목록 저장인 경우에는 등록/수정/삭제를 판별하여 분기처리하는 로직은 베이스 기능에서 처리하였다.

package bankware.corebanking.leadingdevelopment.multirecord.service;

import bankware.corebanking.leadingdevelopment.multirecord.interfaces.ExRateMargin;

…

/\*\*

\* <pre>

\* 본 클래스는 환율마진 관련 처리를 담당한다.

\* - 환율마진목록조회

\* - 환율마진정보조회

\*

\* 1) 주요 메소드

\* - {@link #getListExRate(ExRtMarginMgmtSvcIn)} : 환율마진목록조회

\* - {@link #saveListExRate(ExRtMarginMgmtSvcSaveIn)} : 환율마진정보등록

\*

\* 2)관련화면

\*

\* - LDTMP000020M :환율마진관리

\* </pre>

\*/

@BxmService("ExRateMarginMgmtSvc")

@BxmCategory(logicalName = "ExRateMarginMgmtSvc")

@CbbClassInfo(classType = {"SERVICE"})

public class ExRateMarginMgmtSvc {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private ExRateMargin exRateMargin;

private CmnContext cmnContext; // service restriction detail manager

@BxmServiceOperation("getListExRate")

@CbbSrvcInfo(srvcCd = "SLDTMP0002001", srvcNm = "getListExRate")

public ExRateMarginMgmtSvcGetListExRateOut getListExRate(ExRateMarginMgmtSvcGetListExRateIn in) throws BizApplicationException {

ExRateMarginMgmtSvcGetListExRateOut out = new ExRateMarginMgmtSvcGetListExRateOut();

List<ExRateMarginMgmtSvcExRateMarginInfo> outExRtMarginList= new ArrayList<ExRateMarginMgmtSvcExRateMarginInfo>();

// Base API 호출

List<ExRateMarginIO> boOuts = \_getExRateMargin().getListExRate(\_setExRateMarginIO(in));

if (boOuts != null && !boOuts.isEmpty()) {

for (ExRateMarginIO exRtMarginIO : boOuts ){

outExRtMarginList.add(\_setExRateMarginMgmtSvcOut(exRtMarginIO));

}

}

out.setExRateMarginList(outExRtMarginList);

return out;

}

@BxmServiceOperation("saveListExRate")

@TransactionalOperation

@CbbSrvcInfo(srvcCd = "SLDTMP0002002", srvcNm = "saveListExRate")

public DummyIO saveListExRate(ExRateMarginMgmtSvcSaveListExRateIn in) throws BizApplicationException {

List<ExRateMarginIO> boIn = new ArrayList<ExRateMarginIO>();

boIn = \_setExRtMarginIO(in);

// Base API 호출

int cnt = \_getExRateMargin().saveListExRate(boIn);

return new DummyIO();

}

private ExRateMarginIO \_setExRateMarginIO(ExRateMarginMgmtSvcGetListExRateIn in) {

ExRateMarginIO out = new ExRateMarginIO();

out.setInstCd(in.getInstCd()==null ? \_getCmnContext().getInstCode() : in.getInstCd());// set [기관코드]

out.setBaseCrncyCd(in.getBaseCrncyCd());// set [기준통화코드]

out.setRltdCrncyCd(in.getRltdCrncyCd());// set [상대통화코드]

out.setRgstrnDt(in.getRgstrnDt());// set [등록일자]

return out;

}

private ExRateMarginMgmtSvcExRateMarginInfo \_setExRateMarginMgmtSvcOut(ExRateMarginIO in) {

ExRateMarginMgmtSvcExRateMarginInfo out = new ExRateMarginMgmtSvcExRateMarginInfo();

out.setInstCd(in.getInstCd());// set [기관코드]

out.setBaseCrncyCd(in.getBaseCrncyCd());// set [기준통화코드]

out.setRltdCrncyCd(in.getRltdCrncyCd());// set [상대통화코드]

out.setRgstrnDt(in.getRgstrnDt());// set [등록년월일]

out.setSeqNbr(in.getSeqNbr());// set [일련번호]

out.setExRtDscd(in.getExRtDscd());// set [환율구분코드]

out.setBaseMarginRt(in.getBaseMarginRt());// set [기본마진율]

out.setDlgtdMarginRt(in.getDlgtdMarginRt());// set [위임마진율]

return out;

}

/\*\*

\*

\* @param in

\* @return

\*/

private List<ExRateMarginIO> \_setExRtMarginIO(ExRateMarginMgmtSvcSaveListExRateIn in) {

List<ExRateMarginIO> out = new ArrayList<ExRateMarginIO>();

for(ExRateMarginMgmtSvcExRateMarginInfo entry : in.getExRateMarginList() ){

ExRateMarginIO BoIn = new ExRateMarginIO();

BoIn.setInstCd(entry.getInstCd()==null ? \_getCmnContext().getInstCode() : entry.getInstCd());// set [기관코드]

BoIn.setBaseCrncyCd(entry.getBaseCrncyCd());// set [기준통화코드]

BoIn.setRltdCrncyCd(entry.getRltdCrncyCd());// set [상대통화코드]

BoIn.setRgstrnDt(entry.getRgstrnDt());// set [등록년월일]

BoIn.setSeqNbr(entry.getSeqNbr());// set [일련번호]

BoIn.setExRtDscd(entry.getExRtDscd());// set [환율구분코드]

BoIn.setBaseMarginRt(entry.getBaseMarginRt());// set [기본마진율]

BoIn.setDlgtdMarginRt(entry.getDlgtdMarginRt());// set [위임마진율]

BoIn.setSaveStsCd(entry.getSaveStsCd());// set [저장상태코드]

out.add(BoIn);

}

return out;

}

/\*\*

\* @return the exRt

\*/

private ExRateMargin \_getExRateMargin() {

if (exRateMargin == null) {

exRateMargin = (ExRateMargin) CbbApplicationContext.getBean(ExRateMargin.class, exRateMargin);

}

return exRateMargin;

}

private CmnContext \_getCmnContext() {

if (cmnContext == null) {

cmnContext = (CmnContext) CbbApplicationContext.getBean(

CmnContext.class, cmnContext);

}

return cmnContext;

}

}

### 베이스

목록을 저장하는 것 이외의 앞서 설명한 단건 처리 서비스와 차별성은 없다. 목록 저장인 경우에는 등록/수정/삭제를 판별하여 분기처리하는 로직은 베이스 기능에서 처리하였다.

package bankware.corebanking.leadingdevelopment.multirecord.business;

import bankware.corebanking.leadingdevelopment.multirecord.dao.dto.CmExRtMarginDXtnIO;

…

@BxmBean

@BxmCategory(logicalName = "Exchage Market Rate BO")

@CbbClassInfo(classType={"SERVICE\_BO"})

public class ExRateMarginImpl implements ExRateMargin {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private CmExRtMarginDXtnDso cmExRtMarginDXtnDso; // exchange rate marginDBIO

private CmnContext cmnContext; // Common system header utility

@Override

public List<ExRateMarginIO> getListExRate(ExRateMarginIO in) throws BizApplicationException {

List<ExRateMarginIO> out = new ArrayList<ExRateMarginIO>();

// Inquiry

List<CmExRtMarginDXtnIO> marginOutList = \_getCmExRtMarginDXtnDso().selectList(\_setCmExRateMarginXtnIO(in));

if(marginOutList != null && !marginOutList.isEmpty()) {

for (CmExRtMarginDXtnIO cmExRtMarginDIO : marginOutList) {

ExRateMarginIO exRtMarginIO = \_setExRateMarginIO(cmExRtMarginDIO);

out.add(exRtMarginIO);

} // end for

} // end if

return out;

}

@Override

public int saveListExRate(List<ExRateMarginIO> in)throws BizApplicationException {

int cnt = 0;

// Inquiry

if(in != null) {

for (ExRateMarginIO cmExRtMarginDIO : in) {

CmExRtMarginDXtnIO exRtMarginIO = \_setCmExRateMarginXtnIO(cmExRtMarginDIO);

if(SaveStsCdEnum.CREATE.getValue().equals(cmExRtMarginDIO.getSaveStsCd()) ) {

cnt = \_getCmExRtMarginDXtnDso().insert(exRtMarginIO);

} else if(SaveStsCdEnum.UPDATE.getValue().equals(cmExRtMarginDIO.getSaveStsCd()) ) {

cnt = \_getCmExRtMarginDXtnDso().update(exRtMarginIO);

} else if(SaveStsCdEnum.DELETE.getValue().equals(cmExRtMarginDIO.getSaveStsCd()) ) {

cnt = \_getCmExRtMarginDXtnDso().delete(exRtMarginIO);

} else {

throw new BizApplicationException("AAPASE0052", new Object[]{"@saveStsCd"});

}

} // end for

} // end if

return cnt;

}

private CmExRtMarginDXtnIO \_setCmExRateMarginXtnIO(ExRateMarginIO in) throws BizApplicationException {

CmExRtMarginDXtnIO inParm = new CmExRtMarginDXtnIO();

if(in != null) {

inParm.setInstCd(in.getInstCd());// set [기관코드]

inParm.setBaseCrncyCd(in.getBaseCrncyCd());// set [기준통화코드]

inParm.setRltdCrncyCd(in.getRltdCrncyCd());// set [상대통화코드]

inParm.setRgstrnDt(in.getRgstrnDt());// set [등록년월일]

inParm.setSeqNbr(in.getSeqNbr());// set [일련번호]

inParm.setExRtDscd(in.getExRtDscd());// set [환율구분코드]

inParm.setBaseMarginRt(in.getBaseMarginRt());// set [기본마진율]

inParm.setDlgtdMarginRt(in.getDlgtdMarginRt());// set [위임마진율]

// Set Default 기본 설정

\_getCmnContext().setHeaderColumn(inParm);

}

return inParm;

}

private ExRateMarginIO \_setExRateMarginIO(CmExRtMarginDXtnIO in) throws BizApplicationException {

ExRateMarginIO inParm = new ExRateMarginIO();

if(in != null) {

inParm.setInstCd(in.getInstCd());// set [기관코드]

inParm.setBaseCrncyCd(in.getBaseCrncyCd());// set [기준통화코드]

inParm.setRltdCrncyCd(in.getRltdCrncyCd());// set [상대통화코드]

inParm.setRgstrnDt(in.getRgstrnDt());// set [등록년월일]

inParm.setSeqNbr(in.getSeqNbr());// set [일련번호]

inParm.setExRtDscd(in.getExRtDscd());// set [환율구분코드]

inParm.setBaseMarginRt(in.getBaseMarginRt());// set [기본마진율]

inParm.setDlgtdMarginRt(in.getDlgtdMarginRt());// set [위임마진율]

}

return inParm;

}

/\*\*

\* @return the cmExRtDDso

\*/

private CmExRtMarginDXtnDso \_getCmExRtMarginDXtnDso() {

if (cmExRtMarginDXtnDso == null) {

cmExRtMarginDXtnDso = (CmExRtMarginDXtnDso) CbbApplicationContext.getBean(CmExRtMarginDXtnDso.class, cmExRtMarginDXtnDso);

}

return cmExRtMarginDXtnDso;

}

/\*\*

\* @return the cmnContext

\*/

private CmnContext \_getCmnContext() {

if (cmnContext == null) {

cmnContext = (CmnContext) CbbApplicationContext.getBean(CmnContext.class, cmnContext);

}

return cmnContext;

}

}

### DSO

단건 처리와 동일하게 DSO 클래스는 단순 DAO를 단순 매핑하는 수준에서 처리하였다.

package bankware.corebanking.leadingdevelopment.multirecord.dso;

import bankware.corebanking.leadingdevelopment.multirecord.dao.CmExRtMarginDXtn;

…

@BxmBean

@SuppressWarnings({ "all" })

@BxmCategory(logicalName = "", description = "")

public class CmExRtMarginDXtnDso {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private CmExRtMarginDXtn cmExRtMarginDXtn;

private CmnContext cmnContextCore;

@BxmCategory(description = "")

public CmExRtMarginDXtnIO select(CmExRtMarginDXtnIO cmExRtMarginDIO) {

return \_getCmExRtMarginDXtn().select(cmExRtMarginDIO);

}

@BxmCategory(description = "")

public List<CmExRtMarginDXtnIO> selectList(CmExRtMarginDXtnIO cmExRtMarginDIO) {

return \_getCmExRtMarginDXtn().selectList(cmExRtMarginDIO);

}

@BxmCategory(description = "")

public CmExRtMarginDXtnIO selectMaxNbr(String instCd,String baseCrncyCd, String rltdCrncyCd, String rgstrnDt ,String exRtDscd ) {

return \_getCmExRtMarginDXtn().selectMaxNbr(instCd , baseCrncyCd , rltdCrncyCd, rgstrnDt ,exRtDscd);

}

@BxmCategory(description = "")

public int insert(CmExRtMarginDXtnIO cmExRtMarginDIO) {

return \_getCmExRtMarginDXtn().insert(cmExRtMarginDIO);

}

@BxmCategory(description = "")

public int update(CmExRtMarginDXtnIO cmExRtMarginDIO) {

return \_getCmExRtMarginDXtn().update(cmExRtMarginDIO);

}

@BxmCategory(description = "")

public int delete(CmExRtMarginDXtnIO CmExRtMarginDIO) {

return \_getCmExRtMarginDXtn().delete(CmExRtMarginDIO);

}

/\*\*

\* @return the cmExRtD

\*/

private CmExRtMarginDXtn \_getCmExRtMarginDXtn() {

if (cmExRtMarginDXtn == null) {

cmExRtMarginDXtn = (CmExRtMarginDXtn) CbbApplicationContext.getBean(CmExRtMarginDXtn.class, cmExRtMarginDXtn);

}

return cmExRtMarginDXtn;

}

/\*\*

\* @return the cmnContext

\*/

private CmnContext \_getCmnContext() {

if (cmnContextCore == null) {

cmnContextCore = (CmnContext) CbbApplicationContext.getBean(

CmnContext.class, cmnContextCore);

}

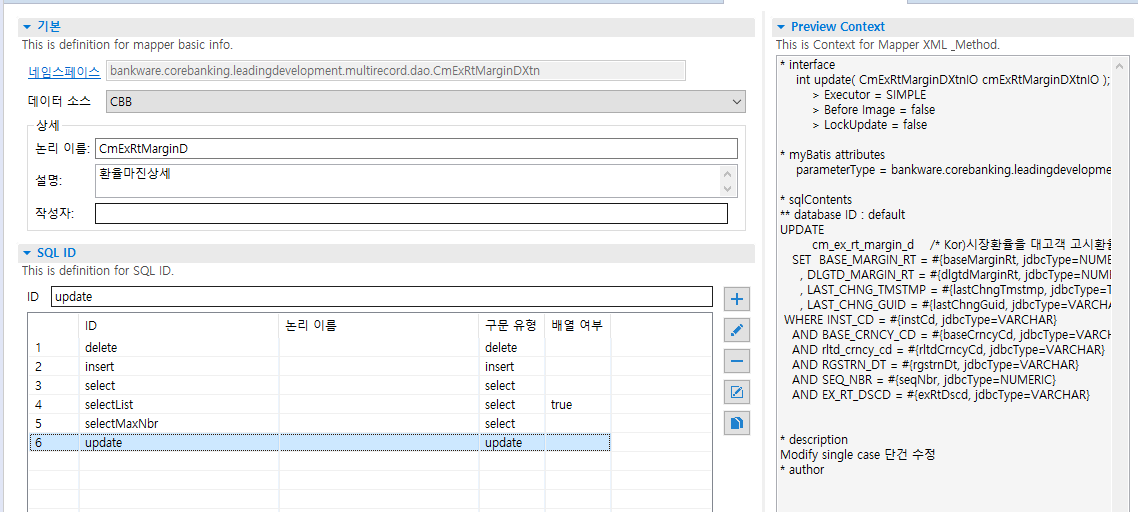
return cmnContextCore;

}

}

### DAO

기본 조회/등록/수정/삭제 거래 이외에 목록을 조회하는 “selectList” SQL\_ID와 단순한 채번을 위한 “selectMaxNbr” SQL\_ID를 작성하였다.



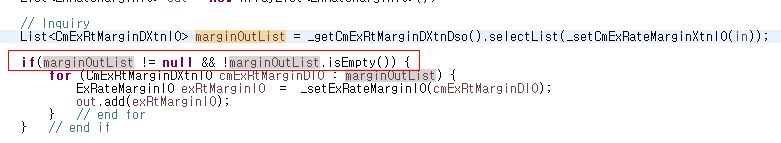
### 중요 전달 사항

#### [서비스/베이스] 서비스는 처리 흐름을 담당하고 베이스는 처리 기능을 담당

* 서비스와 베이스 클래스의 역할  
  - 서비스 : 업무의 흐름을 나타내는 Flow에 대한 구현을 담당  
  - 베이스 : 단위 업무 기능을 처리하기 위한 로직을 담당
* 위 기준으로 판단이 어려운 경우를 위한 보조 판단 기준
  1. 다수의 서비스 또는 베이스에서 호출할 것으로 예상되는 기능단위를 베이스로 구현
  2. 변화가 거의 되지 않는 로직은 베이스로 구현 (최소한의 CRUD기능 단위를 의미)
  3. 운영중에 변화가 예상되는 로직은 서비스로 구현

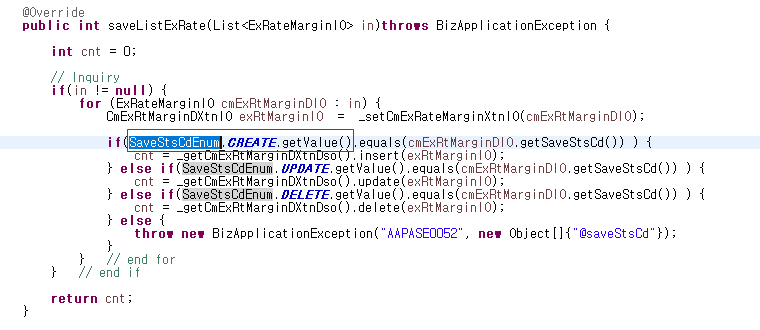
#### [서비스/베이스] 객체목록을 리턴하는 경우에는 null check와 함께 empty 여부를 함께 체크

* 단건의 객체에 대해서는 null 여부를 체크하도록 한다.
* 다건의 객체를 리턴하는 경우에는 혹시 발생할 수 있는 오류를 방지하기 위하여 null 체크와 함께 empty 여부까지 체크하도록 한다.



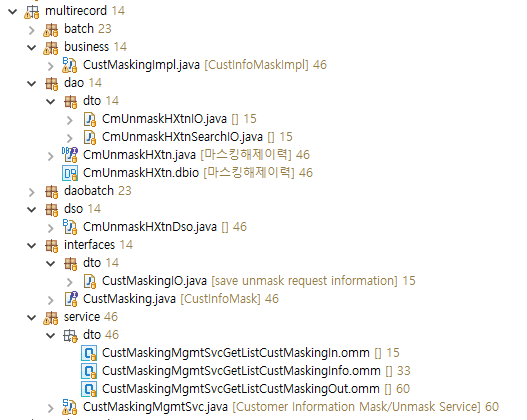
#### [서비스/베이스] String의 값 비교는 상수를 앞에 위치

* String의 값 비교를 위해서 equals()를 사용할 때는 의도치 않은 null 오류를 방지하기 위해서 상수를 앞에 위치한다.



## 다건 조회(페이징)

대량의 Record를 페이징으로 나눠서 조회하는 유형이다. 임의로 ‘고객정보 마스킹해제’라는 업무를 통해 샘플을 구성하였다. ‘CustMaskingMgmt’ 서비스 클래스에서 총 1개의 조회 서비스를 제공하도록 설계하였다.



### 화면

조회의 결과가 그리드로 표현된다. 이때, 페이지 스크롤이 내려갈 때 마다 페이징 처리를 통해 1000건씩 레코드를 추가하도록 처리하였다.



### 서비스

페이징 정보를 보내주어 목록을 조회하는 것 이외의 앞서 설명한 다건 조회 서비스와 차별성은 없다. 페이징 조회 이외에도 전체 건수를 함께 단말로 보내줘야 하기에 전체건수를 조회하는 처리도 함께 하였다.

package bankware.corebanking.leadingdevelopment.multirecord.service;

import bankware.corebanking.leadingdevelopment.multirecord.interfaces.CustMasking;

…

@BxmCategory(logicalName = "Customer Information Mask/Unmask Service")

@BxmService("CustMaskingMgmtSvc")

public class CustMaskingMgmtSvc {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private CustMasking custMasking; // CustMasking

private DeptMngr deptMngr; // Dept

private StaffMngr staffMngr; // Staff

private CustMngr custMngr; // Cust

private ScrnMngr scrnMngr; // Screen

@BxmServiceOperation("getListCustMasking")

@CbbSrvcInfo(srvcCd="SLDTMP0003001",srvcNm="getListCustMasking")

public CustMaskingMgmtSvcGetListCustMaskingOut getListCustMasking(CustMaskingMgmtSvcGetListCustMaskingIn in) throws BizApplicationException {

CustMaskingMgmtSvcGetListCustMaskingOut out = new CustMaskingMgmtSvcGetListCustMaskingOut();

// 1. 입력값 체크 (필수: 날짜 from~to)

// Service 검증규칙으로 처리

// 2. BO 호출을 위한 InDTO 생성

CustMaskingIO boIn = new CustMaskingIO();

boIn.setQryStartDt(DateUtils.getDate(in.getQryStartDt(), DateUtils.DASH\_DATE\_TYPE));

boIn.setQryEndDt(DateUtils.getDate(DateUtils.getDateDaysAfter(1, in.getQryEndDt()), DateUtils.DASH\_DATE\_TYPE));

boIn.setDeptId(in.getDeptId());

boIn.setStaffId(in.getStaffId());

boIn.setCustId(in.getCustId());

boIn.setScrnId(in.getScrnId());

boIn.setUnmaskRsnCd(in.getUnmaskRsnCd());

// 3-1. 전체 건수 BO 호출

int cnt = \_getCustMasking().getCustMaskingCnt(boIn);

out.setTotCnt(cnt);

// 3-2. 페이징 목록 BO 호출

List<CustMaskingIO> OutList = \_getCustMasking().getListCustMasking(boIn, in.getPgNbr(), in.getPgCnt());

// 4. 결과 out 생성 후 리턴

List<CustMaskingMgmtSvcGetListCustMaskingInfo> detailList = new ArrayList<CustMaskingMgmtSvcGetListCustMaskingInfo>();

if (OutList != null && !OutList.isEmpty()) {

for( CustMaskingIO entry : OutList ){

CustMaskingMgmtSvcGetListCustMaskingInfo inner = new CustMaskingMgmtSvcGetListCustMaskingInfo();

inner.setInstCd(entry.getInstCd());

inner.setUnmaskId(entry.getUnmaskId());

inner.setDeptId(entry.getDeptId());

inner.setStaffId(entry.getStaffId());

inner.setCustId(entry.getCustId());

inner.setScrnId(entry.getScrnId());

inner.setUnmaskRsnCd(entry.getUnmaskRsnCd());

inner.setUnmaskRsnCntnt(entry.getUnmaskRsnCntnt());

//inner.setUnmaskTmstmp(new SimpleDateFormat("yyyyMMddHHmmss").format(entry.getUnmaskTmstmp()));

inner.setUnmaskTmstmp(entry.getUnmaskTmstmp());

if(!StringUtils.isEmpty(entry.getDeptId())) {

inner.setDeptNm(\_getDeptMngr().getDept(entry.getInstCd(), entry.getDeptId()) == null ? "" : \_getDeptMngr().getDept(entry.getInstCd(), entry.getDeptId()).getDeptName());

}

if(!StringUtils.isEmpty(entry.getStaffId())) {

inner.setStaffNm(\_getStaffMngr().getStaff(entry.getInstCd(), entry.getStaffId()) == null ? "" : \_getStaffMngr().getStaff(entry.getInstCd(), entry.getStaffId()).getName());

}

if(!StringUtils.isEmpty(entry.getCustId())) {

inner.setCustNm(\_getCustMngr().getCust(entry.getInstCd(), entry.getCustId()) == null ? "" : \_getCustMngr().getCust(entry.getInstCd(), entry.getCustId()).getName());

}

if(!StringUtils.isEmpty(entry.getScrnId())) {

inner.setScrnNm(\_getScrnMngr().getScreen(entry.getScrnId()) == null ? "" : \_getScrnMngr().getScreen(entry.getScrnId()).getScreenName());

}

detailList.add(inner);

}

out.setCustMaskingList(detailList);

}

return out;

}

/\*\*

\* @return the custInfoMask

\*/

private CustMasking \_getCustMasking() {

if (custMasking == null) {

custMasking = (CustMasking) CbbApplicationContext.getBean(CustMasking.class, custMasking);

}

return custMasking;

}

/\*\*

\* @return the deptmngr

\*/

private DeptMngr \_getDeptMngr() {

if (deptMngr == null) {

deptMngr = (DeptMngr) CbbApplicationContext.getBean(DeptMngr.class, deptMngr);

}

return deptMngr;

}

/\*\*

\* @return the staffMngr

\*/

private StaffMngr \_getStaffMngr() {

if (staffMngr == null) {

staffMngr = (StaffMngr) CbbApplicationContext.getBean(StaffMngr.class, staffMngr);

}

return staffMngr;

}

/\*\*

\* @return the custMngr

\*/

private CustMngr \_getCustMngr() {

if (custMngr == null) {

custMngr = (CustMngr) CbbApplicationContext.getBean(CustMngr.class, custMngr);

}

return custMngr;

}

/\*\*

\* @return the scrnMngr

\*/

private ScrnMngr \_getScrnMngr() {

if (scrnMngr == null) {

scrnMngr = (ScrnMngr) CbbApplicationContext.getBean(ScrnMngr.class, scrnMngr);

}

return scrnMngr;

}

}

### 베이스

페이징 정보를 보내주어 목록을 조회하는 것 이외의 앞서 설명한 다건 조회 서비스와 차별성은 없다.

package bankware.corebanking.leadingdevelopment.multirecord.business;

import bankware.corebanking.leadingdevelopment.multirecord.dao.dto.CmUnmaskHXtnIO;

…

/\*\*

\* Manage Address

\*

\* Author keewoong.hong

\* History

\* 2017.09.18 initial version

\*/

@BxmBean

@BxmCategory(logicalName = "CustInfoMaskImpl")

@CbbClassInfo(classType={"SERVICE\_BO"})

public class CustMaskingImpl implements CustMasking {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private CmnContext cmnContext;

private CmUnmaskHXtnDso cmUnmaskHXtnDso;

/\* (non-Javadoc)

\* @see bankware.corebanking.applicationcommon.mask.interfaces.CustInfoMask#getListUnmaskReqInfo(bankware.corebanking.applicationcommon.mask.interfaces.dto.UnmaskReqInfoIn)

\*/

@BxmCategory(logicalName = "getListCustMasking")

@Override

public int getCustMaskingCnt(CustMaskingIO in) throws BizApplicationException {

// 1. BO 호출을 위한 InDTO 생성

CmUnmaskHXtnSearchIO boIn = new CmUnmaskHXtnSearchIO();

boIn.setInstCd(\_getCmnContext().getInstCode());

boIn.setQryStartDt(in.getQryStartDt());

boIn.setQryEndDt(in.getQryEndDt());

boIn.setDeptId(in.getDeptId());

boIn.setStaffId(in.getStaffId());

boIn.setCustId(in.getCustId());

boIn.setScrnId(in.getScrnId());

boIn.setUnmaskRsnCd(in.getUnmaskRsnCd());

// 2. 페이징 DAO 호출

int cnt = \_getCmUnmaskHXtnDso().selectListCount(boIn);

return cnt;

}

/\* (non-Javadoc)

\* @see bankware.corebanking.applicationcommon.mask.interfaces.CustInfoMask#getListUnmaskReqInfo(bankware.corebanking.applicationcommon.mask.interfaces.dto.UnmaskReqInfoIn)

\*/

@BxmCategory(logicalName = "getListCustMasking")

@Override

public List<CustMaskingIO> getListCustMasking(CustMaskingIO in, int pgNbr, int pgCnt) throws BizApplicationException {

List<CustMaskingIO> out = new ArrayList<CustMaskingIO>();

// 2. BO 호출을 위한 InDTO 생성

CmUnmaskHXtnSearchIO boIn = new CmUnmaskHXtnSearchIO();

boIn.setInstCd(\_getCmnContext().getInstCode());

boIn.setQryStartDt(in.getQryStartDt());

boIn.setQryEndDt(in.getQryEndDt());

boIn.setDeptId(in.getDeptId());

boIn.setStaffId(in.getStaffId());

boIn.setCustId(in.getCustId());

boIn.setScrnId(in.getScrnId());

boIn.setUnmaskRsnCd(in.getUnmaskRsnCd());

// 3. 페이징 DAO 호출

List<CmUnmaskHXtnIO> cmUnmaskHXtnIO = new ArrayList<CmUnmaskHXtnIO>();

cmUnmaskHXtnIO = \_getCmUnmaskHXtnDso().selectList(boIn, pgNbr, pgCnt);

// 4. 결과 out 생성 후 리턴

if(cmUnmaskHXtnIO != null && !cmUnmaskHXtnIO.isEmpty()) {

for( CmUnmaskHXtnIO entry : cmUnmaskHXtnIO ){

CustMaskingIO inner = new CustMaskingIO();

inner.setInstCd(entry.getInstCd());

inner.setUnmaskId(entry.getUnmaskId());

inner.setDeptId(entry.getDeptId());

inner.setStaffId(entry.getStaffId());

inner.setCustId(entry.getCustId());

inner.setScrnId(entry.getScrnId());

inner.setUnmaskRsnCd(entry.getUnmaskRsnCd());

inner.setUnmaskRsnCntnt(entry.getUnmaskRsnCntnt());

inner.setUnmaskTmstmp(entry.getUnmaskTmstmp());

out.add(inner);

}

}

return out;

}

/\*\*

\* @return the cmnContext

\*/

private CmnContext \_getCmnContext() {

if (cmnContext == null) {

cmnContext = (CmnContext) CbbApplicationContext.getBean(CmnContext.class, cmnContext);

}

return cmnContext;

}

/\*\*

\* @return the cmnContext

\*/

private CmUnmaskHXtnDso \_getCmUnmaskHXtnDso() {

if (cmUnmaskHXtnDso == null) {

cmUnmaskHXtnDso = (CmUnmaskHXtnDso) CbbApplicationContext.getBean(CmUnmaskHXtnDso.class, cmUnmaskHXtnDso);

}

return cmUnmaskHXtnDso;

}

}

**\***

### DSO

페이징 정보를 보내주어 목록을 조회하는 것 이외의 앞서 설명한 다건 조회 서비스와 차별성은 없다.

package bankware.corebanking.leadingdevelopment.multirecord.dso;

import bankware.corebanking.leadingdevelopment.multirecord.dao.CmUnmaskHXtn;

…

@BxmBean

@SuppressWarnings({ "all" })

@BxmCategory(logicalName = "", description = "")

public class CmUnmaskHXtnDso {

final Logger logger = LoggerFactory.getLogger(this.getClass());

private CmUnmaskHXtn cmUnmaskHXtn;

@BxmCategory(description = "")

public int selectListCount(CmUnmaskHXtnSearchIO in) throws BizApplicationException {

return \_getCmUnmaskH().selectListCount(in);

}

@BxmCategory(description = "")

public List<CmUnmaskHXtnIO> selectList(CmUnmaskHXtnSearchIO in, int pgNbr, int pgCnt) throws BizApplicationException {

return \_getCmUnmaskH().selectList(in, pgNbr, pgCnt);

}

/\*\*

\* @return the cmZipCdM

\*/

private CmUnmaskHXtn \_getCmUnmaskH() {

if (cmUnmaskHXtn == null) {

cmUnmaskHXtn = (CmUnmaskHXtn) CbbApplicationContext.getBean(CmUnmaskHXtn.class, cmUnmaskHXtn);

}

return cmUnmaskHXtn;

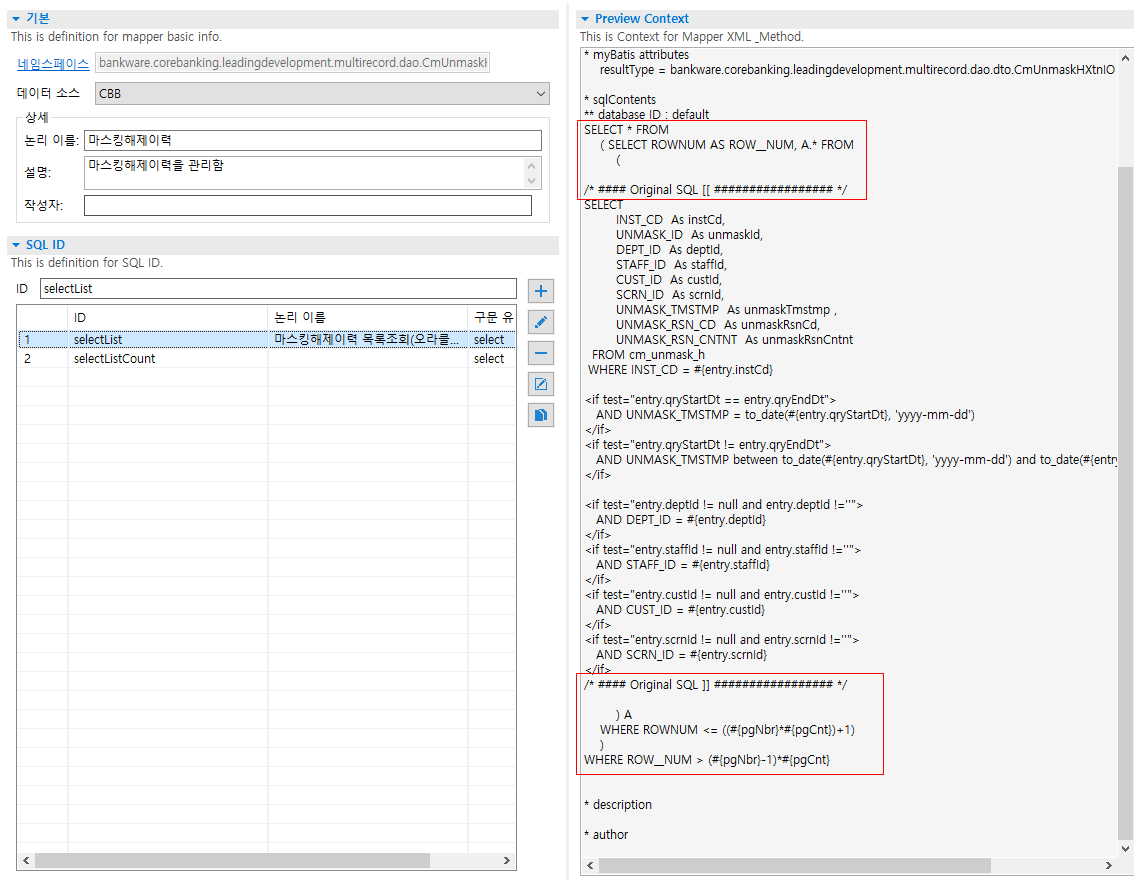
}

}

**\***

### DAO

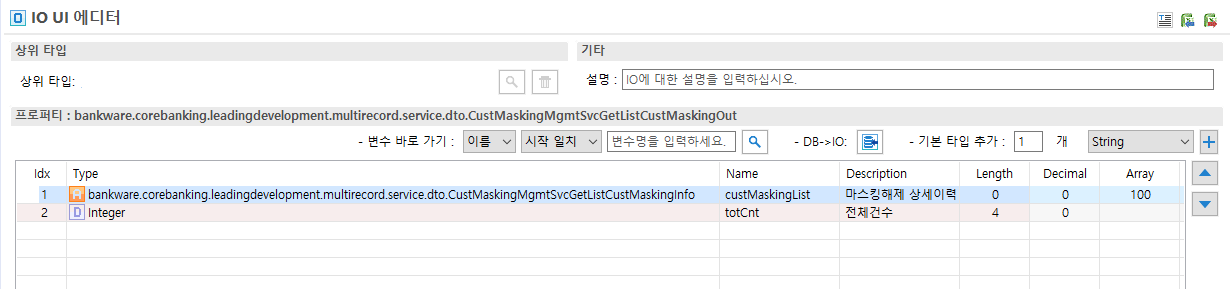
페이징으로 다건 조회하는 “selectList” SQL\_ID와 전체건수를 조회하는 “selectListCount” SQL\_ID를 작성하였다.



### 중요 전달 사항

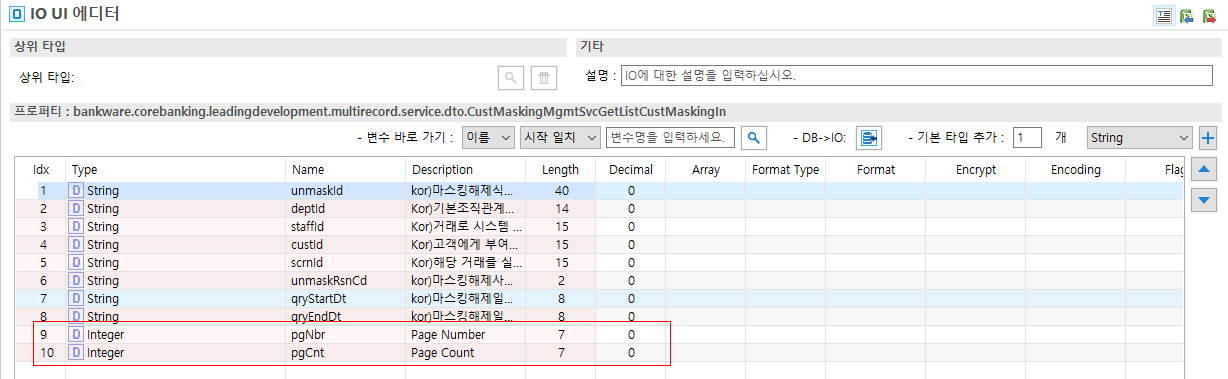
#### [서비스] 페이징 목록 조회와 함께 ‘전체건수’도 함께 단말로 전달 필요

* 페이징 처리 시, 단말 화면에서는 “현재건수 / 전체건수” 형태로 보여줌  
  - 조회 거래의 출력 항목에 전체 건수도 함께 전달 필요



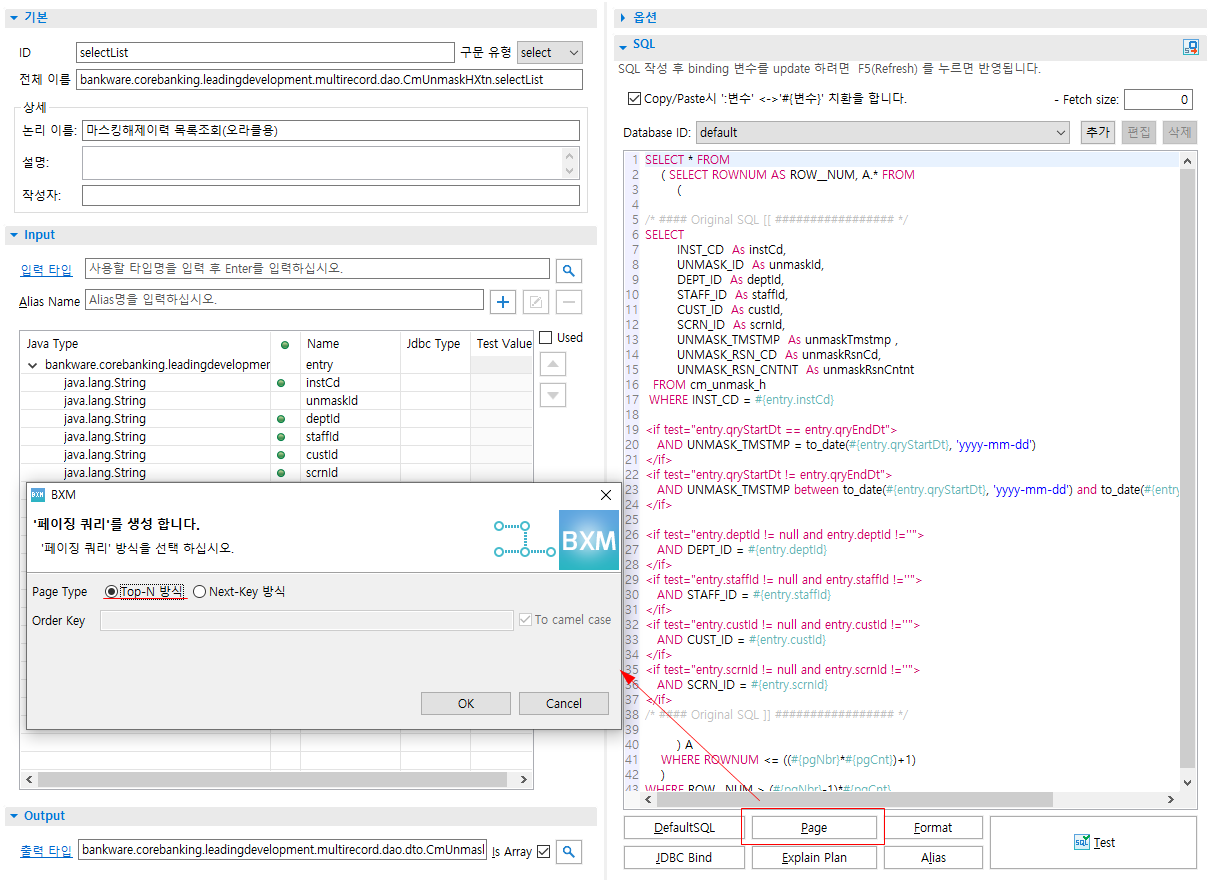
#### [서비스] 페이지 조회를 위해 ‘페이지 번호’, ‘페이징 건수’도 함께 서비스로 전달 필요

* 페이징 조회의 입력항목에 ‘페이지번호’와 ‘페이징건수’도 함께 서비스로 전달 필요



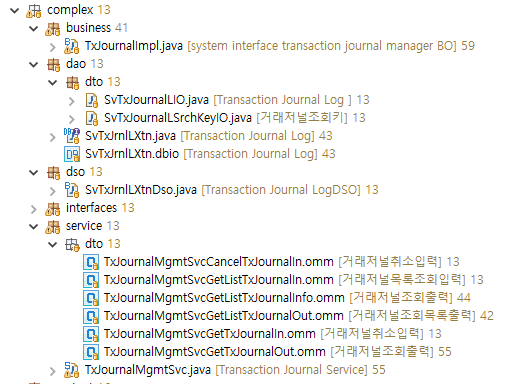
#### [DAO] 페이징 Query는 DBIO의 [Page] > Top-N방식 을 이용하여 작성

* 기본 Query를 작성 후에 DBIO 하단의 [Page] > Top-N 방식을 이용하여 작성하도록 한다.



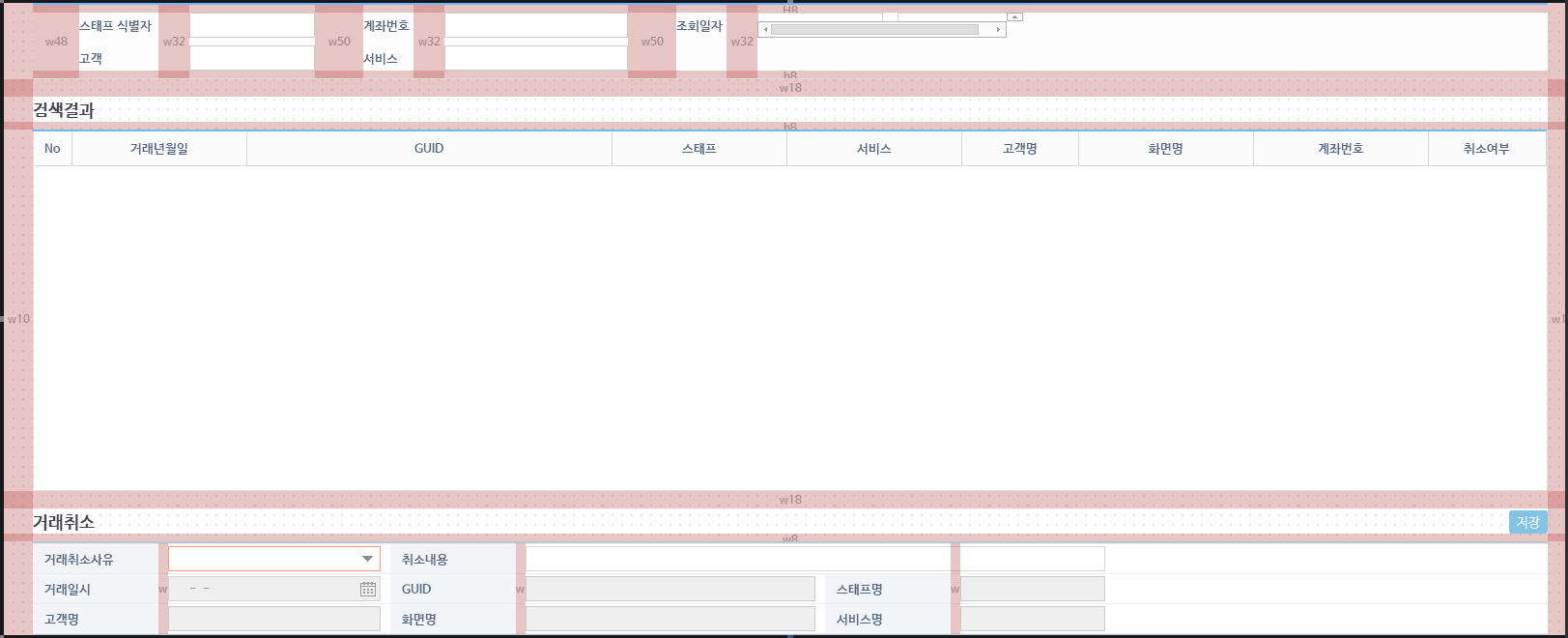
## 복합 처리

다수의 Record를 조회한 후 화면의 그리드에서 1건씩 상세 조회를 한 후 저장 거래를 일으키는 유형이다. 임의로 ‘저널정보 조회 및 취소’라는 업무를 통해 샘플을 구성하였다. ‘TxJournalMgmtSvc’ 서비스 클래스에서 총 3개의 조회 서비스를 제공하도록 설계하였다.



### 화면

조회의 결과가 그리드로 표현된다. 그리드의 레코드를 선택하면 상세정보 조회 거래를 통해 하단 상세 내역을 보여주고 있다. 하단 상세 영역에서 별도 저장 거래를 통해 ‘거래취소’를 호출하도록 처리하였다.



### 서비스

다건 조회 및 단건 상세조회, 단건 저장 처리를 수행하고 있다. 다양한 유형의 처리를 하나의 서비스에서 수행하는 구성을 가지고 있다.

package bankware.corebanking.leadingdevelopment.complex.service;

import bankware.corebanking.leadingdevelopment.complex.interfaces.TxJournal;

…

@BxmCategory(logicalName = "Transaction Journal Service")

@BxmService("TxJournalMgmtSvc")

public class TxJournalMgmtSvc {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private TxJournal txJournal; // Transaction Journal

private InstSrvcMngr instSrvcMngr; // institution service management

private TrnsfrLng trnsfrLng;

private StaffMngr staffMngr; // staff manager

private CustMngr custMngr;

private SrvcPrfl srvcPrfl; // service profile

private CmnContext cmnContext; // service restriction detail manager

@BxmServiceOperation("getTxJournal")

@CbbSrvcInfo(srvcCd="SLDTMP0004002",srvcNm="getTxJournal")

public TxJournalMgmtSvcGetTxJournalOut getTxJournal(TxJournalMgmtSvcGetTxJournalIn in) throws BizApplicationException {

TxJournalMgmtSvcGetTxJournalOut out = new TxJournalMgmtSvcGetTxJournalOut();

// Bo 입력값 셋팅

TxJournalIO boIn = new TxJournalIO();

boIn.setInstCd(in.getInstCd()==null ? \_getCmnContext().getInstCode() : in.getInstCd());

boIn.setTxDt(in.getTxDt());

boIn.setGuid(in.getGuid());

boIn.setSeqNbr(in.getSeqNbr()==null ? BigDecimal.ZERO : in.getSeqNbr());

// Base API 호출

TxJournalIO boOut = new TxJournalIO();

boOut = \_getTxJournal().getTxJournal(boIn);

// 서비스 출력값 셋팅

if (boOut != null) {

out.setInstCd(boOut.getInstCd());// set [기관코드]

out.setTxDt(boOut.getTxDt());// set [거래년월일]

out.setGuid(boOut.getGuid());// set [GUID]

out.setSeqNbr(boOut.getSeqNbr());// set [순번]

out.setStaffId(boOut.getStaffId());// set [스태프식별자]

out.setSrvcCd(boOut.getSrvcCd());// set [서비스코드]

out.setTxHms(boOut.getTxHms());// set [거래시각]

out.setScrnId(boOut.getScrnId());// set [화면식별자]

out.setAcctNbr(boOut.getAcctNbr());// set [계좌번호]

out.setCustId(boOut.getCustId());// set [고객식별자]

out.setCnclYn(boOut.getCnclYn());// set[취소여부]

out.setCnclDt(boOut.getCnclDt());// set[취소일자]

out.setTxCnclRsnCd(boOut.getTxCnclRsnCd());// set[거래취소사유코드]

out.setCnclCntnt(boOut.getCnclCntnt());// set[취소내용]

out.setStaffNm(!StringUtils.isEmpty(boOut.getStaffId()) ? \_getStaffMngr().getStaff(boOut.getStaffId()).getName() : boOut.getStaffId());// set [스태프명]

out.setSrvcNm(!StringUtils.isEmpty(boOut.getSrvcCd()) ? \_getTransferLanguage(CCM01.TRANS\_SRVC, boOut.getSrvcCd()) : boOut.getSrvcCd());// set [서비스명]

out.setScrnNm(!StringUtils.isEmpty(boOut.getScrnId()) ? \_getTransferLanguage(CCM01.TRANS\_SCRN, boOut.getScrnId()) : boOut.getScrnId());// set [화면명]

out.setCustNm(!StringUtils.isEmpty(boOut.getCustId()) ? \_getCustName(boOut.getInstCd(), boOut.getCustId()) : boOut.getCustId());// set [고객명]

}

return out;

}

@BxmServiceOperation("getListTxJournal")

@CbbSrvcInfo(srvcCd="SLDTMP0004001",srvcNm="getListTxJournal")

public TxJournalMgmtSvcGetListTxJournalOut getListTxJournal(TxJournalMgmtSvcGetListTxJournalIn in) throws BizApplicationException {

TxJournalMgmtSvcGetListTxJournalOut out = new TxJournalMgmtSvcGetListTxJournalOut();

List<TxJournalIO> txJrnlIOList = new ArrayList<TxJournalIO>();

List<TxJournalMgmtSvcGetListTxJournalInfo> tbl = new ArrayList<TxJournalMgmtSvcGetListTxJournalInfo>();

// Base API 호출

txJrnlIOList = \_getTxJournal().getListTxJournal(\_gettxJrnlSrchKeyIO(in));

// 서비스 출력값 셋팅

if(txJrnlIOList != null && !txJrnlIOList.isEmpty()) {

for(TxJournalIO txJrnlIO : txJrnlIOList) {

tbl.add(\_getTxJrnlSvcGetTxJrnlOut(txJrnlIO));

}

}

out.setTxJournalInfoList(tbl);

return out;

}

@BxmServiceOperation("cancelTxJournal")

@CbbSrvcInfo(srvcCd="SLDTMP0004003",srvcNm="cancelTxJournal")

public DummyIO cancelTxJournal(TxJournalMgmtSvcCancelTxJournalIn in) throws BizApplicationException {

DummyIO out = new DummyIO();

// Bo 입력값 셋팅

TxJournalIO boIn = new TxJournalIO();

boIn.setInstCd(in.getInstCd()==null ? \_getCmnContext().getInstCode() : in.getInstCd());

boIn.setTxDt(in.getTxDt());

boIn.setGuid(in.getGuid());

boIn.setTxCnclRsnCd(in.getTxCnclRsnCd());

boIn.setCnclCntnt(in.getCnclCntnt());

// Base API 호출

int cnt = \_getTxJournal().updateCancel(boIn);

return out;

}

/\*\*

\* \_gettxJrnlSrchKeyIO

\*

\* @param in

\* @return TxJrnlSrchKeyIO

\* @throws BizApplicationException

\*/

private TxJournalSrchKeyIO \_gettxJrnlSrchKeyIO(TxJournalMgmtSvcGetListTxJournalIn in) throws BizApplicationException {

TxJournalSrchKeyIO txJournalSrchKeyIO = new TxJournalSrchKeyIO();

txJournalSrchKeyIO.setInqryStartDt(in.getInqryStartDt());

txJournalSrchKeyIO.setInqryEndDt(in.getInqryEndDt());

txJournalSrchKeyIO.setStaffId(in.getStaffId());

txJournalSrchKeyIO.setSrvcCd(in.getSrvcCd());

txJournalSrchKeyIO.setAcctNbr(in.getAcctNbr());

txJournalSrchKeyIO.setCustId(in.getCustId());

return txJournalSrchKeyIO;

}

/\*\*

\* \_getTxJrnlSvcGetTxJrnlOut

\*

\* @param txJrnlIO

\* @return TxJrnlSvcGetTxJrnlOut

\* @throws BizApplicationException

\*/

private TxJournalMgmtSvcGetListTxJournalInfo \_getTxJrnlSvcGetTxJrnlOut(TxJournalIO txJrnlIO) throws BizApplicationException {

TxJournalMgmtSvcGetListTxJournalInfo out = new TxJournalMgmtSvcGetListTxJournalInfo();

out.setInstCd(txJrnlIO.getInstCd());

out.setTxDt(txJrnlIO.getTxDt());

out.setGuid(txJrnlIO.getGuid());

out.setStaffId(txJrnlIO.getStaffId());

out.setSrvcCd(txJrnlIO.getSrvcCd());

out.setSrvcNm(\_getTransferLanguage(CCM01.TRANS\_SRVC, txJrnlIO.getSrvcCd()));

out.setTxHms(txJrnlIO.getTxHms());

out.setScrnId(txJrnlIO.getScrnId());

out.setScrnNm(\_getTransferLanguage(CCM01.TRANS\_SCRN, txJrnlIO.getScrnId()));

out.setChnlCd(txJrnlIO.getChnlCd());

out.setAcctNbr(txJrnlIO.getAcctNbr());

out.setCustId(txJrnlIO.getCustId());

out.setCustNm(\_getCustName(txJrnlIO.getInstCd(), txJrnlIO.getCustId()));

out.setInpCntnt(txJrnlIO.getInpCntnt());

out.setCnclYn(txJrnlIO.getCnclYn());

out.setCnclDt(txJrnlIO.getCnclDt());

out.setTxCnclRsnCd(txJrnlIO.getTxCnclRsnCd());

out.setCnclCrctnTpCd(txJrnlIO.getCnclCrctnTpCd());

out.setOrgnlTxDt(txJrnlIO.getOrgnlTxDt());

out.setOrgnlTxGuid(txJrnlIO.getOrgnlTxGuid());

out.setCnclTxGuid(txJrnlIO.getCnclTxGuid());

out.setCnclSrvcCd(\_getCancelServiceCoce(txJrnlIO.getInstCd(), txJrnlIO.getSrvcCd()));

out.setCnclCntnt(txJrnlIO.getCnclCntnt());

out.setCupldTxCntnt(txJrnlIO.getCupldTxCntnt2());

out.setStaffNm(!StringUtils.isEmpty(txJrnlIO.getStaffId()) ? \_getStaffMngr().getStaff(txJrnlIO.getStaffId()).getName() : txJrnlIO.getStaffId());// set [스태프명]

return out;

}

/\*\*

\* \_getCancelServiceCoce

\*

\* @param instCd

\* @param srvcCd

\* @return String cacel service code

\* @throws BizApplicationException

\*/

private String \_getCancelServiceCoce(String instCd, String srvcCd) throws BizApplicationException {

// check valid service of institution

InstSrvcInfo instSrvc = \_getInstSrvcMngr().getInstService(instCd, srvcCd);

if( instSrvc == null ) {

throw new BizApplicationException("AAPSVE0016", new String[] {instCd + "/" + srvcCd});

}

// get service profile

List<SrvcPrflGetSrvcPrflIO> srvcPrflList = \_getSrvcPrfl().getListInstServiceProfileByJoin(instCd, srvcCd);

String srvcRstrctnRgstrnYn = null;

for(SrvcPrflGetSrvcPrflIO srvcPrfl : srvcPrflList) {

if(SrvcPrflAtrbtEnum.SERVICE\_RESTRICTION\_REGISTRATION\_YN.getValue().equals(srvcPrfl.getSrvcPrflAtrbtNm())) {

srvcRstrctnRgstrnYn = StringUtils.isEmpty(srvcPrfl.getInstSrvcPrflCntnt()) ? srvcPrfl.getSrvcPrflCntnt() : srvcPrfl.getInstSrvcPrflCntnt();

break;

}

}

String cancelServiceCode = "";

for(SrvcPrflGetSrvcPrflIO srvcPrfl : srvcPrflList) {

if(SrvcPrflAtrbtEnum.CANCEL\_SERVICE\_CODE.getValue().equals(srvcPrfl.getSrvcPrflAtrbtNm())) {

cancelServiceCode = StringUtils.isEmpty(srvcPrfl.getInstSrvcPrflCntnt()) ? srvcPrfl.getSrvcPrflCntnt() : srvcPrfl.getInstSrvcPrflCntnt();

break;

}

}

return cancelServiceCode;

}

private String \_getCustName(String instCd, String custId) throws BizApplicationException {

if(StringUtils.isEmpty(custId)) {

return "";

}

Cust cust = \_getCustMngr().getCust(instCd, custId);

return cust == null ? "" : cust.getName();

}

private String \_getTransferLanguage(String trnsfrKnd, String trnsfrKeyVal) throws BizApplicationException {

if(StringUtils.isEmpty(trnsfrKeyVal)) {

return "";

}

return \_getTrnsfrLng().getTransferLanguage(trnsfrKnd, trnsfrKeyVal);

}

/\*\*

\* @return the txJrnl

\*/

private TxJournal \_getTxJournal() {

if (txJournal == null) {

txJournal = (TxJournal) CbbApplicationContext.getBean(TxJournal.class, txJournal);

}

return txJournal;

}

/\*\*

\* @return the trnsfrLng

\*/

private TrnsfrLng \_getTrnsfrLng() {

if (trnsfrLng == null) {

trnsfrLng = (TrnsfrLng) CbbApplicationContext.getBean(TrnsfrLng.class, trnsfrLng);

}

return trnsfrLng;

}

/\*\*

\* @return the staffMngr

\*/

private StaffMngr \_getStaffMngr() {

if (staffMngr == null) {

staffMngr = (StaffMngr) CbbApplicationContext.getBean(StaffMngr.class, staffMngr);

}

return staffMngr;

}

/\*\*

\* @return the custMngr

\*/

private CustMngr \_getCustMngr() {

if (custMngr == null) {

custMngr = (CustMngr) CbbApplicationContext.getBean(CustMngr.class, custMngr);

}

return custMngr;

}

/\*\*

\* @return the instSrvcMgmt

\*/

private InstSrvcMngr \_getInstSrvcMngr() {

if (instSrvcMngr == null) {

instSrvcMngr = (InstSrvcMngr) CbbApplicationContext.getBean(InstSrvcMngr.class, instSrvcMngr);

}

return instSrvcMngr;

}

/\*\*

\* @return the srvcPrfl

\*/

private SrvcPrfl \_getSrvcPrfl() {

if (srvcPrfl == null) {

srvcPrfl = (SrvcPrfl) CbbApplicationContext.getBean(SrvcPrfl.class, srvcPrfl);

}

return srvcPrfl;

}

private CmnContext \_getCmnContext() {

if (cmnContext == null) {

cmnContext = (CmnContext) CbbApplicationContext.getBean(

CmnContext.class, cmnContext);

}

return cmnContext;

}

}

### 베이스

페이징 정보를 보내주어 목록을 조회하는 것 이외의 앞서 설명한 다건 조회 서비스와 차별성은 없다.

package bankware.corebanking.leadingdevelopment.complex.business;

import bankware.corebanking.leadingdevelopment.complex.dso.SvTxJrnlLXtnDso;

…

/\*\*

\* transaction journal management for system interface

\*

\* Author Yongsoon Park

\* History

\* 2015.10.08 initial version for 2.3

\*/

@BxmBean

@BxmCategory(logicalName = "system interface transaction journal manager BO")

@CbbClassInfo(classType={"SERVICE\_BO"})

public class TxJournalImpl implements TxJournal {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private SvTxJrnlLXtnDso svTxJrnlLXtn; // system transaction journal

private CmnContext cmnContext; //common system header utility

@BxmCategory(logicalName = "get transaction journalizing for canceling")

@Override

public TxJournalIO getTxJournal(TxJournalIO in) throws BizApplicationException {

TxJournalIO out = new TxJournalIO();

SvTxJournalLIO inDao = new SvTxJournalLIO();

inDao.setInstCd(in.getInstCd());

inDao.setTxDt(in.getTxDt());

inDao.setGuid(in.getGuid());

inDao.setSeqNbr(in.getSeqNbr());

SvTxJournalLIO dsoOut = new SvTxJournalLIO();

// Base API 호출

dsoOut = \_getSvTxJrnlLXtnDso().select(inDao);

if(dsoOut != null) {

out.setInstCd(dsoOut.getInstCd());

out.setTxDt(dsoOut.getTxDt());

out.setGuid(dsoOut.getGuid());

out.setSeqNbr(dsoOut.getSeqNbr());

out.setStaffId(dsoOut.getStaffId());

out.setSrvcCd(dsoOut.getSrvcCd());

out.setTxHms(dsoOut.getTxHms());

out.setScrnId(dsoOut.getScrnId());

out.setAcctNbr(dsoOut.getAcctNbr());

out.setCustId(dsoOut.getCustId());

out.setInpCntnt(dsoOut.getInpCntnt());

out.setCnclYn(dsoOut.getCnclYn());

out.setCnclDt(dsoOut.getCnclDt());

out.setTxCnclRsnCd(dsoOut.getTxCnclRsnCd());

out.setCnclCrctnTpCd(dsoOut.getCnclCrctnTpCd());

out.setCnclCntnt(dsoOut.getCnclCntnt());

out.setOrgnlTxDt(dsoOut.getOrgnlTxDt());

out.setOrgnlTxGuid(dsoOut.getOrgnlTxGuid());

out.setCnclTxGuid(dsoOut.getCnclTxGuid());

out.setLastChngTmstmp(dsoOut.getLastChngTmstmp());

out.setLastChngGuid(dsoOut.getLastChngGuid());

out.setCupldTxCntnt2(dsoOut.getCupldTxCntnt2());

}

return out;

}

@BxmCategory(logicalName = "get transaction journalizing list for canceling")

@Override

public List<TxJournalIO> getListTxJournal(TxJournalSrchKeyIO in) throws BizApplicationException {

List<TxJournalIO> out = new ArrayList<TxJournalIO>();

SvTxJournalLSrchKeyIO inDao = new SvTxJournalLSrchKeyIO();

inDao.setInstCd(\_getCmnContext().getInstCode());

inDao.setInqryStartDt(in.getInqryStartDt());

inDao.setInqryEndDt(in.getInqryEndDt());

inDao.setGuid(in.getGuid());

inDao.setStaffId(in.getStaffId());

inDao.setSrvcCd(in.getSrvcCd());

inDao.setScrnId(in.getScrnId());

inDao.setAcctNbr(in.getAcctNbr());

inDao.setCustId(in.getCustId());

List<SvTxJournalLIO> outDaoList = \_getSvTxJrnlLXtnDso().selectList(inDao);

if(outDaoList != null) {

for(SvTxJournalLIO outDao : outDaoList) {

TxJournalIO sub = new TxJournalIO();

sub.setInstCd(outDao.getInstCd());

sub.setTxDt(outDao.getTxDt());

sub.setGuid(outDao.getGuid());

sub.setSeqNbr(outDao.getSeqNbr());

sub.setStaffId(outDao.getStaffId());

sub.setSrvcCd(outDao.getSrvcCd());

sub.setTxHms(outDao.getTxHms());

sub.setScrnId(outDao.getScrnId());

sub.setAcctNbr(outDao.getAcctNbr());

sub.setCustId(outDao.getCustId());

sub.setInpCntnt(outDao.getInpCntnt());

sub.setCnclYn(outDao.getCnclYn());

sub.setCnclDt(outDao.getCnclDt());

sub.setTxCnclRsnCd(outDao.getTxCnclRsnCd());

sub.setCnclCrctnTpCd(outDao.getCnclCrctnTpCd());

sub.setCnclCntnt(outDao.getCnclCntnt());

sub.setOrgnlTxDt(outDao.getOrgnlTxDt());

sub.setOrgnlTxGuid(outDao.getOrgnlTxGuid());

sub.setCnclTxGuid(outDao.getCnclTxGuid());

sub.setCupldTxCntnt2(outDao.getCupldTxCntnt());

out.add(sub);

}

}

return out;

}

@BxmCategory(logicalName = "save service input")

@Override

public int updateCancel(TxJournalIO in) throws BizApplicationException {

SvTxJournalLIO inDao = new SvTxJournalLIO();

inDao.setInstCd(in.getInstCd());

inDao.setTxDt(in.getTxDt());

inDao.setGuid(in.getGuid());

inDao.setSeqNbr(BigDecimal.ZERO);

SvTxJournalLIO dsoOut = new SvTxJournalLIO();

// Base API 호출

dsoOut = \_getSvTxJrnlLXtnDso().select(inDao);

int cnt = 0;

if(dsoOut != null) {

dsoOut.setTxCnclRsnCd(in.getTxCnclRsnCd());

dsoOut.setCnclCntnt(in.getCnclCntnt());

dsoOut.setCnclTxGuid(\_getCmnContext().getGuid());

dsoOut.setCnclCrctnTpCd(CnclCrctnTpEnum.CORRECTION.getValue());

dsoOut.setCnclDt(\_getCmnContext().getTxDate()); // 취소년월일

dsoOut.setCnclYn(CCM01.YES);

\_getCmnContext().setHeaderColumn(dsoOut);

cnt = \_getSvTxJrnlLXtnDso().updateCancel(dsoOut);

}

if(cnt <= 0) {

throw new BizApplicationException("AAPCME0165", new Object[] { "@orgnlTxGuid" });

}

return cnt;

}

/\*\*

\* @return the svTxJrnlL

\*/

private SvTxJrnlLXtnDso \_getSvTxJrnlLXtnDso() {

if (svTxJrnlLXtn == null) {

svTxJrnlLXtn = (SvTxJrnlLXtnDso) CbbApplicationContext.getBean(SvTxJrnlLXtnDso.class, svTxJrnlLXtn);

}

return svTxJrnlLXtn;

}

/\*\*

\* @return the cmnContext

\*/

private CmnContext \_getCmnContext() {

if (cmnContext == null) {

cmnContext = (CmnContext) CbbApplicationContext.getBean(CmnContext.class, cmnContext);

}

return cmnContext;

}

}

### DSO

페이징 정보를 보내주어 목록을 조회하는 것 이외의 앞서 설명한 다건 조회 서비스와 차별성은 없다.

package bankware.corebanking.leadingdevelopment.complex.dso;

import bankware.corebanking.leadingdevelopment.complex.dao.SvTxJrnlLXtn;

...

@BxmBean

@SuppressWarnings({ "all" })

@BxmCategory(logicalName = "Transaction Journal LogDSO", description = "Transaction Journal LogDSO")

public class SvTxJrnlLXtnDso {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

private SvTxJrnlLXtn svTxJrnlLXtn;

private SysNonstopUtil sysNonstopUtil;

@BxmCategory(description = "거래저널 상세조회")

public SvTxJournalLIO select(SvTxJournalLIO svTxJrnlLIO) throws BizApplicationException {

return \_getSvTxJrnlLXtn().select(svTxJrnlLIO);

}

@BxmCategory(description = "거래저널 목록조회")

public List<SvTxJournalLIO> selectList(SvTxJournalLSrchKeyIO svTxJrnlLSrchKeyIO) throws BizApplicationException {

return \_getSvTxJrnlLXtn().selectList(svTxJrnlLSrchKeyIO);

}

@BxmCategory(description = "거래저널 취소")

public int updateCancel(SvTxJournalLIO svTxJrnlLIO) {

return \_getSvTxJrnlLXtn().updateCancel(svTxJrnlLIO);

}

/\*\*

\* @return the svTxJrnlL

\*/

public SvTxJrnlLXtn \_getSvTxJrnlLXtn() {

if (svTxJrnlLXtn == null) {

svTxJrnlLXtn = (SvTxJrnlLXtn) CbbApplicationContext.getBean(SvTxJrnlLXtn.class, svTxJrnlLXtn);

}

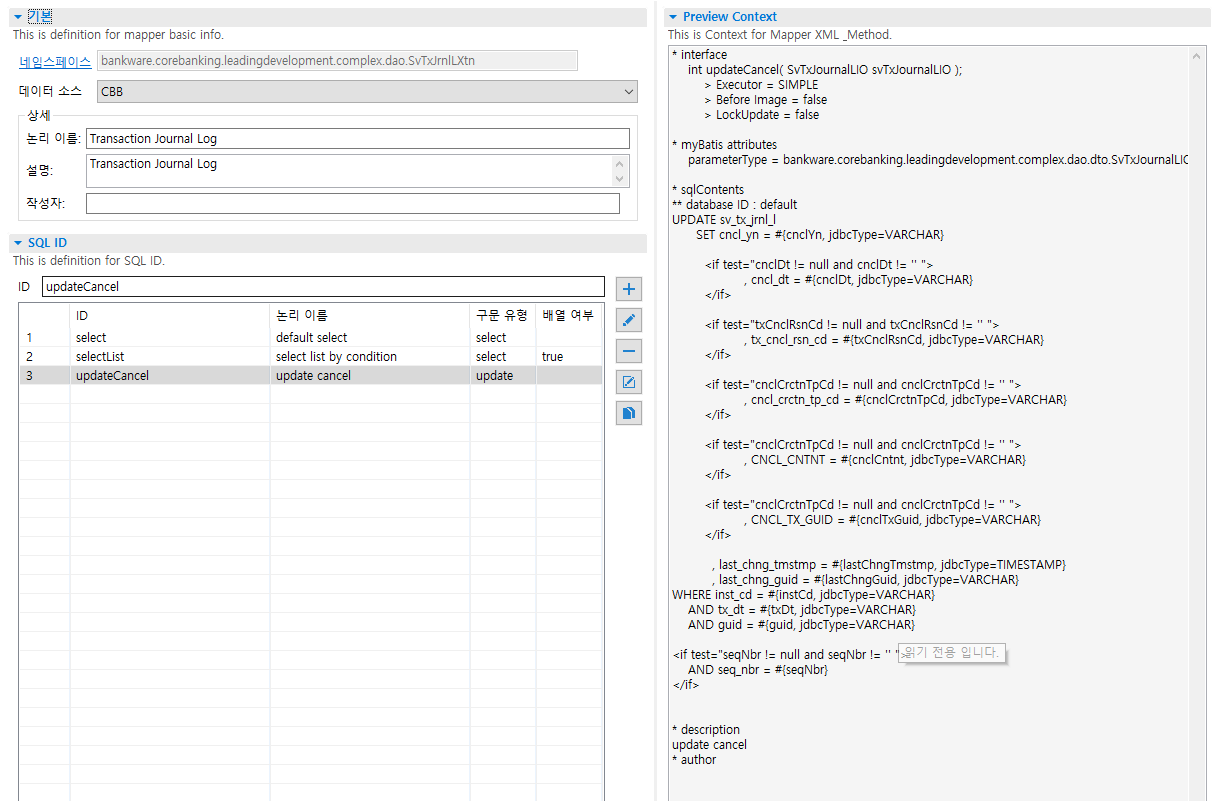
return svTxJrnlLXtn;

}

}

### DAO

다건 조회하는 “selectList” SQL\_ID, 단건 조회하는 “select”, 단건에 대한 취소정보를 업데이트하는 “updateCanel” SQL\_ID를 작성하였다.



### 중요 전달 사항

N/A