210824 아키텍처2 & 베이스 컴포넌트 & 공통

생성일	@2021년 8월 24일 오전 9:03
성함	나한주
속성	
수업 유형	이론
수정일	@2021년 8월 25일 오전 12:13
작성자	현 현동빈

아키텍처2

어플리케이션 계층 > 데이터 접근 레이어(DAO)

- 데이터접근하는 Object 의미

프로젝트(컴포넌트)간 SQL 조인 불가 → 자바코드로 불러와야 함

- DBIO가 프로그램 처리 속도의 대부분을 차지함 → Cost가 높다(CPU사용률, 메모리 사용률...리소스 비율)
- Join 수행시간 & 조회 + 조회 ⇒ Join 수행속도가 빠름

프로젝트(컴포넌트)간 SQL 조인 불가

- 프로젝트(컴포넌트) 간 조인을 통한 데이터의 공유는 제한된다. 타 프로젝트(컴포넌트)의 데이터가 필요하면, 해당 BO를 통해서 Java 소스 내에서 공유해야 한다. 조인이 성능차원에서는 이점이 있을 수 있으나, CBP에서는 컴포넌트의 독립성, 데이터 캡슐화 측면에서 SQL을 통한 조인을 금지하고 있다.
- 다만, 동일 컴포넌트 내에서의 조인은 허용한다.

DBMS 종속 쿼리 지양, 단순 쿼리를 지향

- ANSI SQL 문법이 있는 경우 ANSI 문법 준수
- SQL 문법을 자유롭게 Converting할 수 있어야 함
- 값의 연산은 SQL이 아닌 Java에서 처리하고 최종값만 DTO에 Set
- DBIO와 비즈니스 로직을 분리해서 처리해야 함(현재는 DBMS가 발달해서...예전에는 분리해서 처리했어야 함) Ex) PL-SQL(오라클)

MyBatis 동적태그 허용

온라인 기본 처리 흐름

하나의 서비스를 처리하기 위한 많은 기능들 중에 공통 기능들은 서비스 전/후처리를 이용하여 처리하고 있음

선처리

- 권한검사
- 입력 전문 매핑(DTO 생성) → JSON
- 입력값 검증 → 검증규칙
- Main Business Object 상태 검증

서비스

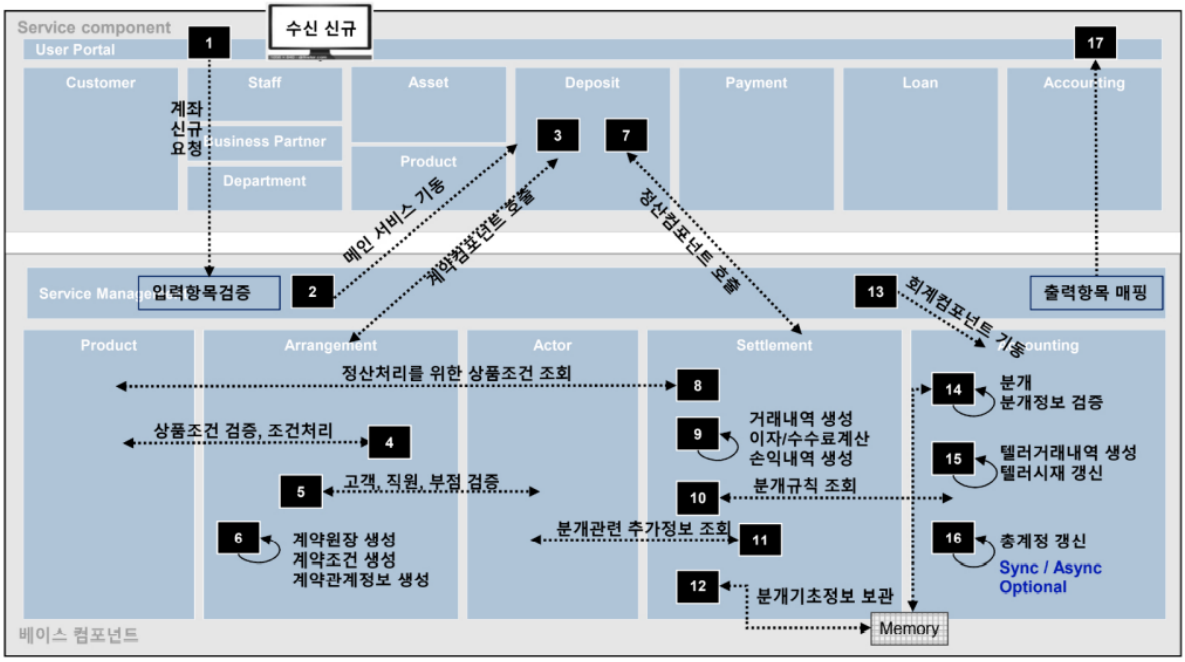
- 서비스제어 - 특정 상품에 따른 서비스 차단 등

후처리

- 분개처리를 위한 데이터 이동
- 출력 전문 매핑
- 다음 스탭 처리를 위한 출력 전문 보관(거래저널)
- 서비스 로그 기록

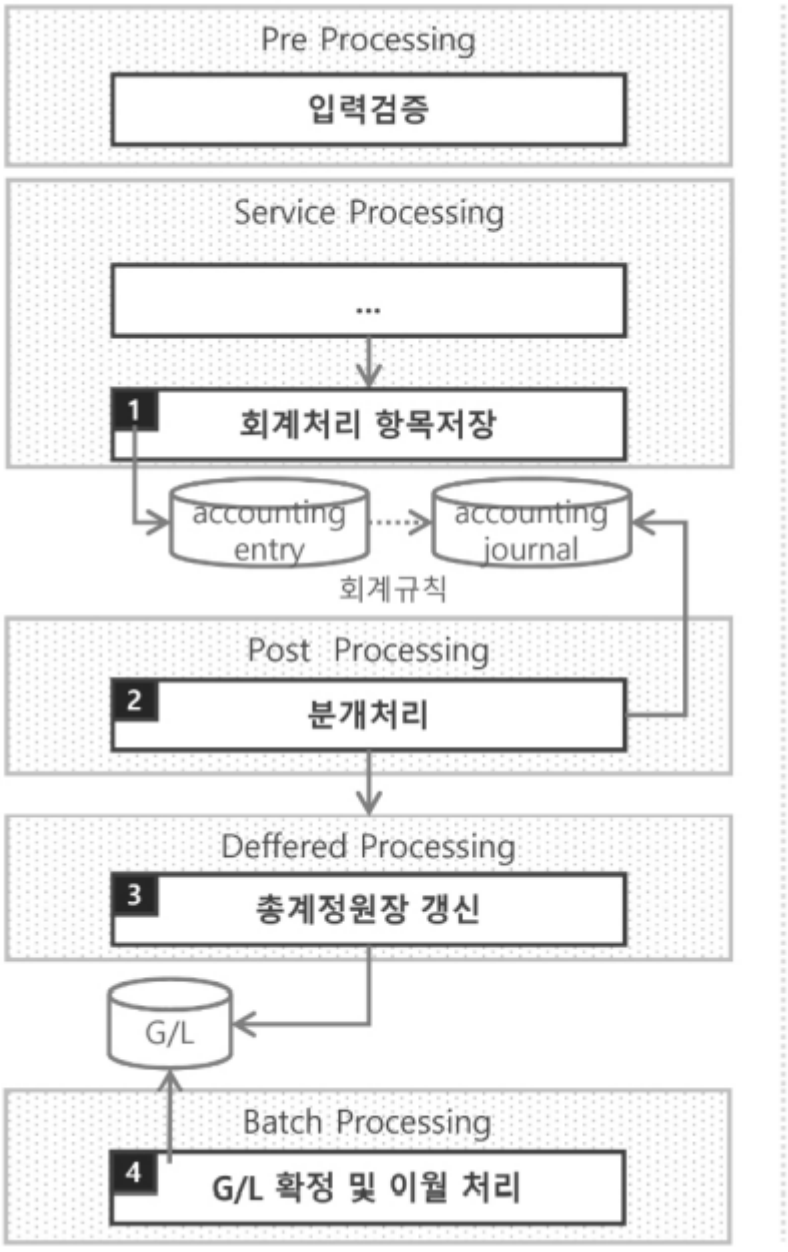
⇒ 서비스 선/후처리에서 일어나는 작업 이해하고 구분 할 수 있어야 함

온라인 기본 처리 흐름 > 상세



- BO는 극단적으로 분리되어 있음
- BO간의 호출을 최대한 배제(지양)

회계 처리 흐름

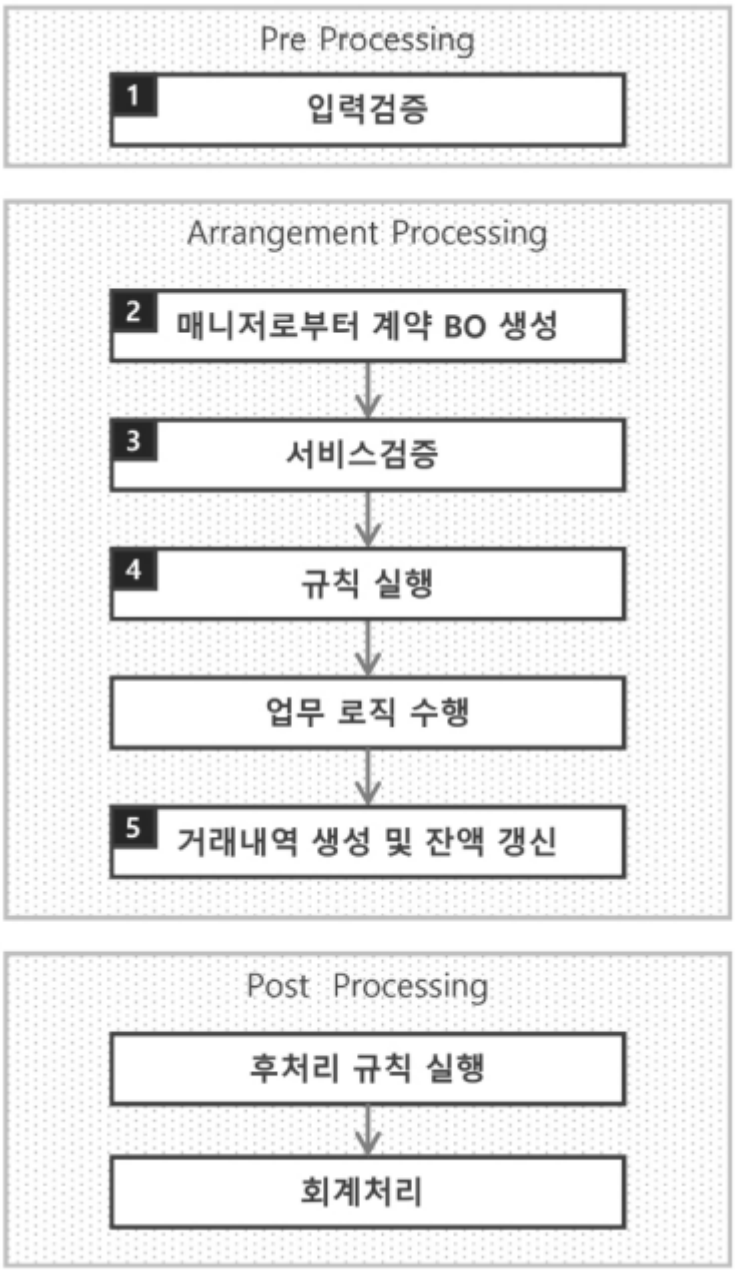


- 회계처리 항목저장 : 거래저널 ⇒ 엔트리 조립
- 총계정원장 : 회계는 서비스와는 독립적으로 운영(돌고 있음) ⇒ 회계를 위해 거래저널이 이용되기 때문에 거래저널이 문제없이 쌓여야 함
- 후처리에서 분개처리(회계호출)

기타 처리 흐름 > 계약성 거래 기본 흐름

- 선처리 → 입력검증(등록된 표준 속성 검증 규칙에 따라 자동적으로 입력값 검증)

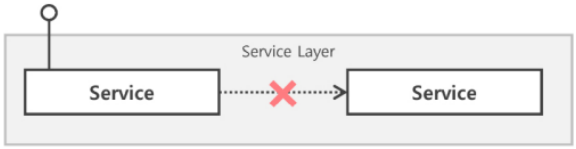
- Main Service
  - 매니저(Manager)로부터 계약BO 생성 (매니저 클래스로 대장BO(Domain) 호출)
  - 계약BO는 식별성 요건이 있는 BO(Domain BO)
  - 서비스검증 → 계약서비스 Validation(계약과 관련된 공통 검증 수행, 계약서비스의 등록된 규칙에 따라 다양한 고객 검증과 사고 여부 등을 검증)
  - 규칙 실행
  - Create transaction & Update balance
    - 거래내역을 생성하고 입출내역(엔트리) 등록
    - 계약에서 관리하는 잔액을 갱신하고, 회계처리를 위한 일계정보를 조립



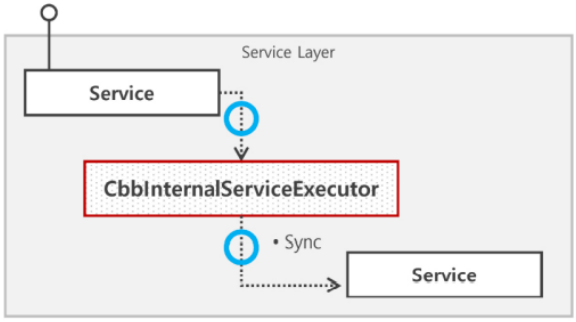
기타 처리 흐름 > 연동거래(거래 =서비스)

- 내부 서비스간 호출이 필요한 경우, 프레임워크 영역을 통해 호출할 수 있음. 이 때 호출되는 서비스는 외부에서 호출한 것과 같이 동작
- 서비스가 서비스를 직접 호출 X
- CbbInternalSeviceExecutor(서비스코드, Input Value)에 서비스코드 입력
- Ex) Postman(외부에서 호출) → getStaff → 입금 → getStaff

정의



✓ I/F 표준 규칙에 따라 서비스는 다른 서비스를 직접 호출 할 수 없다.



✓ I/F 표준 규칙에 따라 내부 서비스는 CbbInternalServiceExecutor 를 통해서 호출 한다.

설명

CbbInternalServiceExecutor class Usage

✓ **execute**(String, R) : for Synchronous Service

Example

```
DeptSrchSvcGetDeptListIn inputDto = new
DeptSrchSvcGetDeptListIn();

inputDto.setInstCd("000");

CbbInternalServiceExecutor.execute("SUP2008401", inputDto);
```

명명 표준 > 프로젝트

프로젝트종류	규칙	예시
서비스 프로젝트	L1컴포넌트 약어 + Svc를 사용한다.	회계(Accounting)서비스는 ACSvc
베이스 프로젝트	L1컴포넌트 약어를 사용한다.	회계(Accounting)는 AC
배치 프로젝트	L1컴포넌트 약어 + Bat를 사용한다.	수신(Deposit)배치는 DPBat
예외 프로젝트	컴포넌트로 정의 되지 않는 경우.	BizProxy, CORE-intrfc

명명 표준 > 패키지

프로젝트종류	주요 패키지	규칙	예시
서비스 프로젝트	서비스	CBP도메인명.L1명.L2명.service	bankware.corebanking.customer.open.service
	업무로직	CBP도메인명.L1명.L2명.bizproc	bankware.corebanking.customer.open.bizproc
베이스 프로젝트	Interface	CBP도메인명.L1명.L2명.interfaces	bankware.corebanking.actor.customer.interfaces
	Enum	CBP도메인명.L1명.enums	bankware.corebanking.actor.enums
배치 프로젝트	배치bean	CBP도메인명.L1명.L2명.배치작업식별자.batch	bankware.corebanking.deposit.transaction.dpstaccrual.batch.dto
	배치DAO	CBP도메인명.L1명,L2명.배치작업식별자.dao	bankware.corebanking.deposit.transaction.dpstaccrual.dao
예외 프로젝트	BizProxy	CBP도메인명.L1명.L2명.bizproc.interfaces	bankware.corebanking.customer.open.bizproc.interfaces

- Enum :~.L1.enums
- 나머지 : ~.L1.L2.~.

명명 표준 > 클래스

기본 규칙
- 파스칼표기법 사용. (첫문자 대문자)

프로젝트종류	주요 클래스	규칙	예시
서비스 프로젝트	서비스	표준약어+"Svc"	DpstDrwgSvc 수신입금서비스
	업무로직	표준약어+"BizProc"	DpstSrvcCustNoticeBizProc 수신서비스고객통지공통처리
베이스 프로젝트	Interface	표준약어	ArrCnd 계약조건, Cust 고객
	Enum	코드의 약어명("Cd" 제외) + "Enum"	ActorTpEnum(액터유형 Enum)
배치 프로젝트	배치bean	Job명("Job" 제외) + 표준약어+ [Reader, Writer, Processor, Tasklet]	대상고객추출ItemReader: CustCrdtInfoXtrctnTrgtReader
	배치DAO	테이블명 + 'Bat' + '_' + Job명("Job" 제외)	ArArrMBat_KtbLnOpnReport (계약기본 : ArArrM, Job명 : KtbLnOpnReportJob)
예외 프로젝트	BizProxy	표준약어	DpstSrvcCustNotice

- Enum(L1에서 모든 Enum은 여기서 정의)
- BIzProxy : Jar 파일을 만들기 위함
- 서비스, BizProc 알아두기

명명 표준 > 메소드

- 카멜표기법 사용

기본 규칙
- 카멜표기법 사용.
- 동사 + 명사(형용사)의 형태로 사용한다.
- 동사는 full name을 사용하고, 표준동사를 사용하는 것을 권고한다.
- 명사(형용사)는 표준단어(full name)을 사용한다.

[public/protected]
- 원칙 : 동사 + 표준단어조합(카멜표기법,full name)
- 작성예: createArrangement() 계약생성

[private]
- "_" + 표준동사 + 표준단어조합(카멜표기법, full name)
- 작성예: 입력검증: _validateInput()

명명 표준 > DB

- 데이터 모델 명명 규칙

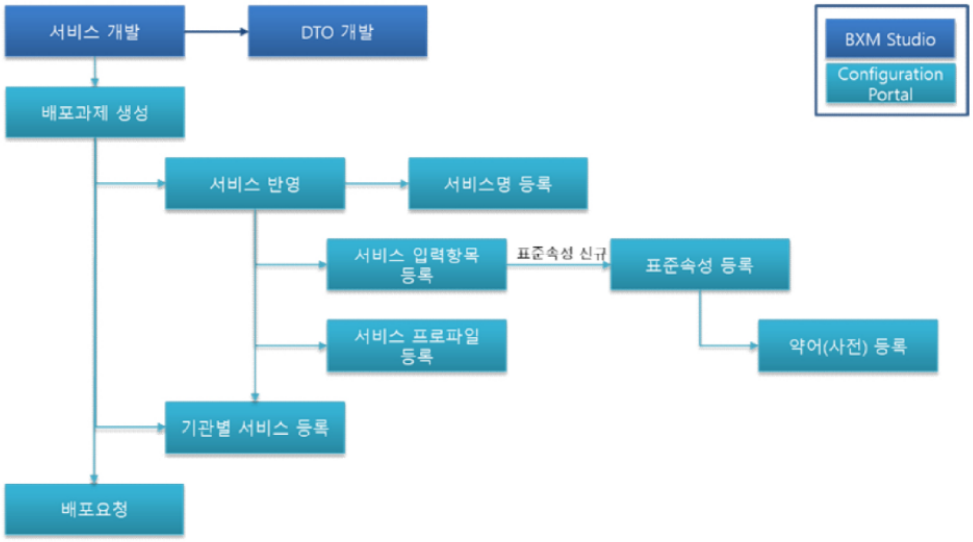
Object	길이	Rule
Schema	8 byte	[A-Z], [a-z],[0-9],_
Table	30 byte	베이스업무구분[2] + '_' + 테이블 물리명 + '_' + 테이블유형구분[1]
Column	30 byte	표준속성
View	30 byte	베이스업무구분[2] + '_' + 뷰 테이블 물리명 + '_' + 뷰테이블구분자[1]
Index	30 byte	베이스업무구분[2] + '_' + 데이터주제영역 + '_' + 테이블식별번호[3] + '_' + 인덱스식별번호[3]
PK	30 byte	'PK' + '_' + 테이블넘버

- PK
  - Unique함 (중복데이터 X) → PK로 조회를 하면 속도가 빠름
  - 제 1의 Index
  - PK라서 Indexing되어 있어서 속도가 빠름
  - Clusted Index
- Index를 추가함
  - Non Clusted Index

용어

번호	용어	설 명
1	Service	서비스는 화면/대외계 시스템 등에서 요청된 업무 처리를 수행하기 위한 메소드(오퍼레이션)들을 포함하고 있는 클래스 타입 @BxmService, @BxmServiceOperation
2	Bean	Bean은 비즈니스 로직을 구현하고 있는 메소드들을 포함하고 있는 클래스 타입 @BxmBean
3	DTO	DTO는 Data Transfer Object의 약자로, data를 저장 관리하며 전달하는 객체를 의미
4	OMM	OMM(Object Message Mapping)은 BXM 프레임워크에서 사용하는 표준 데이터 전달 객체로, 모든 DTO는 OMM으로 관리된다.
5	Base API	CBB에서 제공하는 API, 적절한 API의 조합으로 서비스를 작성한다.
6	연동거래 (내부연동)	서비스내에서 기존에 개발된 서비스를 호출하여 로직을 재사용 할 수 있다.
7	VO	VO 는 Value Object 의 약자로, DTO처럼 쉽게 Editor를 사용하여 데이터 객체를 생성 관리 한다. DTO와 다르게 OMM 관련 로직을 없고, 순수하게 getter, setter 를 가진 java 파일로 관리된다.

서비스 개발 절차



어노테이션 종류

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

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

- @BxmService는 클래스 이름과 같아야 함
- @BxmServiceOperation은 메서드의 이름과 같아야 함
- Transaction → 여러 SQL문을 동시에 실행시켰을 때 하나라도 오류가 발생하면 전부 Rollback시킴
- @CbbSrvcInfo안의 srvcCd에는 서비스코드가 들어가야함
- @BxmServiceOperation명, srvcAbrvtnNm명, 메서드 명은 동일하게 함

에러메시지

- 새로운 Exception을 던짐(throw)

- BXM에서는 throw new BizApplicationException(메시지코드, 메시지코드에 넘겨줄 String 값)

- → "저장 시 오류가 발생했습니다 @atrbt님"

트랜잭션 분리

- 특정 메소드 내의 데이터 처리를 별도의 트랜잭션으로 처리할 때 사용

- 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.

- 분리되어야 할 Transaction에 @TransactionalOperation (propagation= Propagation. REQUIRES_NEW)를 기술한다.

- 트랜잭션분리된 method내에서 Exception이 발생한 경우는, 분리된 트랜잭션에서 갱신한 데이터와 함께 상위 본 트랜잭션도 rollback처리된다.

- "모두"

내부연동거래

- 개발된 서비스를 재사용하기 위한 목적
- CbbInternalServiceExecutor.execute(~, ~)
- call depth에 따라 처리가 필요할 때, 첫 기동거래에서 callDepth는 0
- Ex) Postman → Svc call → Svc call, Svc call, Svc call
- ⇒ 0 1 2 3

베이스 컴포넌트

베이스 컴포넌트 제공 기능

- 주요 베이스 컴포넌트의 기능은 다음과 같음
- 액터는 고객,직원,파트너와 같은 업무관계자와 관련된 기능을 제공
- 계약은 계약사항과 관련된 기능을 제공
  - 은행과 고객과의 계약으로 생성된 계좌를 관리하거나
  - 가수/가지급등의 은행내의 내부 계좌를 관리하거나
  - 고객이 은행에 인터넷뱅킹등 회원가입과 관련된 계약을 관리하는 등 다양한 형태의 계약 사항을 관리할 수 있는 기능을 제공
- 정산은 거래이력을 관리하거나 계좌의 잔액을 관리하거나 수신의 이자계산 등을 처리하는 기능을 제공
- 상품은 상품정보를 조회하거나 검증하는 기능을 제공
- 공통은 기능개발에 필요한 각종 공통적인 기능을 제공
- 회계는 회계처리 및 업무 개시 마감등과 관련된 기능을 제공

베이스 컴포넌트 영문명

- 액터(AT), 계약(AR), 정산(ST), 상품(PD), 공통(CM)은 최소한 알고 있어야 함
- 액터(AT) 컴포넌트 ≠ 고객(CU) 컴포넌트
- 베이스 컴포넌트 & 서비스 컴포넌트 구분할 수 있어야함

공통

공통 영역 담당 프로젝트

- 서비스

- 베이스

- 서비스 컴포넌트(업무공통)
- 서비스 선/후처리
- 베이스 컴포넌트(단위기능)
- ⇒ 문제 가능 유형 : 베이스컴포넌트가 아닌 것 구분 할 수 있기

Configuration Center

- CP와 AP 나눠서 동작
- CP : 저장/삭제/수정 가능
- AP : 조회만 가능

동작 방식

- Configuration Data 배포
- 원복 기능
  - 배포 과정 중 오류 발생 시, 각 서버별 모든 Data를 일괄 원복 처리함(Rollback)
  - 각 서버별 배포결과를 Configuration Portal에서 일괄 관리
  - 특정 서버의 문제 발생 시, 해당 변경과제를 일괄 원복이 가능함

- 이 때, 기 배포가 완료되어 대상서버에서 Commit까지 된 자료까지 기존 Data로 원복을 시켜줌

기능

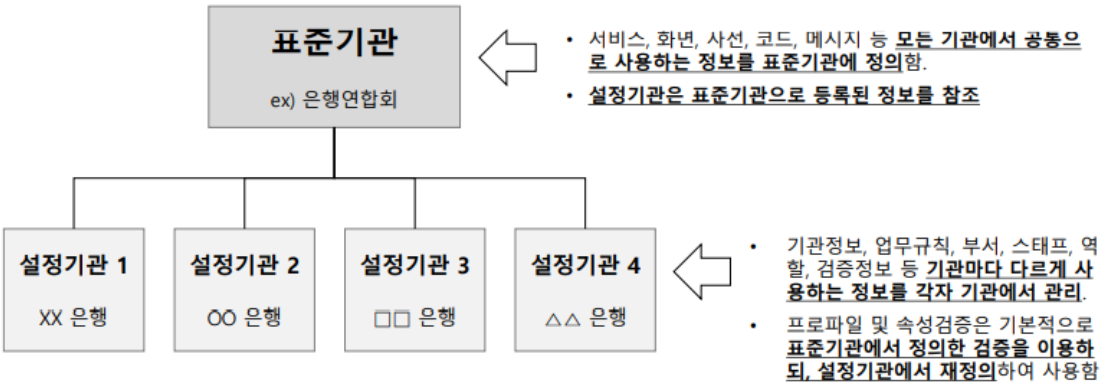
구분	내용
시스템운영	온라인서비스 및 클래스 정보등 시스템 전반적인 운영을 위한 정보를 관리한다. 또한 온라인서비스 기준정보를 Admin Portal로 배포하기 위한 배포관리 기능을 포함하고 있다.
사용자인터페이스	CBP의 화면속성 정보와 함께 사용자그룹 및 채널별 메뉴 체계를 관리한다.
업무규칙	CBP 업무를 수행하기 위한 업무 기준정보와 함께 해당 업무에서 사용하는 파라미터 정보 및 플러그인 클래스정보를 관리한다.
사용자	액터에서 관리하는 스태프, 부서의 기준정보를 관리한다. 또한 스태프의 역할과 함께 역할별 권한정보를 관리한다.
사전/메시지	단어 및 도메인 같은 사전정보와 함께 표준속성 및 속성별 검증유형을 을 관리한다. 또한 다국어, 공통코드, 각종 메시지에 대해서도 관리한다.
환경설정	시스템 전체에 영향을 주는 파라미터 정보와 함께 영업일 정보와 분류체계 정보를 관리한다.

로그인 및 화면 구성

- CP & AP에서는 기관설정을 하지 않고 로그인(들어가서 기관설정)
- 업무화면에서는 기관설정을 선택하고 로그인(들어가서는 기관설정 불가)

기관

CBP는 기관의 본래 의미를 따르면서 모든 Data의 기준이 되는 최상위 개념으로서 기관을 관리하고 있다. 또한 표준기관과 설정기관을 통해 멀티기관을 지원한다.



기관이란 법인 또는 기타 단체의 의사 결정이나 그 실행에 참여하는 지위에 있고, 그 행위가 법인이나 단체의 행위로 여겨지는 개인 또는 그 집단을 의미한다. **CBP에서는 기관의 본래 의미를 따르면서 모든 Data의 기준이 되는 최상위 개념으로서 관리하고 있다. CBP가 갖고 있는 모든 공통 베이스컴포넌트의 모델들은 일부 공통 기준정보를 제외하고, 기본적으로 최상위 개념으로 기관코드 정보를 보유하고 있다.**

기관 선택

- CP & AP : 로그인 시 기관설정 X
- 업무화면 : 로그인 시 기관설정 O
- Configuration Center 우측 상단에서 기관설정 시스템 일자 설정 가능
- Configuration Center 우측화면에서 영향범위 볼 수 있음

기관 영향범위

- 적용범위 기본적으로 설정기관 별 적용
- 확인해둘 것
  - 서비스사용현황
  - 서비스프로파일
  - DTO속성검증 규칙
  - 서비스제어, 서비스제한 정보

공통업무유형	적용범위	
	전체기관(표준기관)	설정기관
시스템운영	• 서비스기본정보 • *서비스프로파일* • *DTO속성검증규칙* • 클래스정보, 배포정보	• 서비스사용현황 • *서비스프로파일* • *DTO속성검증 규칙* • 서비스제어, 서비스제한 정보
사용자인터페이스	• 화면기본정보	• 화면사용현황, 메뉴정보
업무규칙		• 계약, 고객, 정산, 회계, 심사, 담보 업무규칙 • 채번, 문서, 승인, 중요증서 설정정보
사용자		• 부서, 사용자 정보 • 역할, 권한, 보안 정보
사전/메시지	• 단어, 복합단어, 도메인, 다국어 정보 • 표준속성, *속성검증 규칙*, 참조속성 정보 • 코드(표준, 확장, 서브셋) 정보 • 에러메시지	• *속성검증 규칙* • 테이블확장속성 정보 • 외부코드, 통화코드 • 통지메시지
환경설정	• 분류체계 정보	• 기관정보, 기관파라메터 • 영업일정보, 당일정보

설정기관별로 Configuration 정보를 별도 관리하여 사용한다. 그 중 일부 정보는 해당 기관에 설정정보가 등록되지 않은 경우 'Standard기관'의 정보를 이용한다.

예) 서비스프로파일

서비스코드	프로파일정보	기관	설정값
SAC6000100	txJrnlCrtnYn	Standard	Y
SAC6000100	txJrnlCrtnYn	901	N
SAC6000100	txJrnlCrtnYn	902	
SAC6000100	txJrnlCrtnYn	999	

예) 서비스프로파일 중 거래저널생성여부(txJrnlCtrnYn) 값이 있다.
이 값이 'Y'로 설정된 서비스는 거래저널을 생성하고, null 이거나 'N'인 서비스의 경우는 거래저널을 생성하지 않는다.

각 기관별 거래저널의 생성여부는 다음과 같다.

- **표준기관(Standard) : 거래저널을 생성함**
- **901 기관 : 거래저널을 생성하지 않음 (901 기관의 설정값 이용)**
- **902 기관 : 거래저널을 생성함. (Standard 기관의 설정값 이용)**
- **903 기관 : 거래저널을 생성함. (Standard 기관의 설정값 이용)**

설정기관이 관리하는 Data가 없을 경우, Standard 기관의 정보를 이용하는 정보

정보 유형
서비스 프로파일
DTO 속성검증 규칙
속성검증 규칙
테이블확장속성 검증규칙 정보

- 기본적으로 설정기관이 거래저널을 생성하는 지 여부를 체크한 뒤 그에 따라 표준기관 거래저널 생성여부 결정
- 902, 903 기관이 거래저널을 생성함 : 표준기관의 거래저널을 생성한다는 의미
- 설정기관이 관리하는 Data가 없을 경우, Standard기관의 정보를 이용하는 정보
  - 서비스 프로파일
  - DTO 속성검증 규칙

- 속성검증 규칙
- 테이블확장속성 검증규칙 정보

  ⇒ "프디속확"

기관 파라미터

- **CBP에서 기관 파라미터란?**
  - **기관별로 별도의 처리를 수행해야 하는 경우에 기관 파라미터 속성으로 정의하여 기관별로 값을 다르게 설정하여 처리하도록 함**
- **CBP 베이스에서의 기관별 별도 처리는 기관 파라미터를 이용함**
  - **CBP 주요 업무 로직 내에 기관별 별도 로직 처리가 필요한 부분은 기관 파라미터를 참조하여 별도의 처리가 되도록 처리하고 있음**
  - **기관별 요건이 추가됨에 따라 계속 늘어나고 있는 추세임**
- **기관 파라미터의 종류**
  - **기관, 보안, 상품, 고객, 정산, 부서, 서비스프로파일, 채번 등의 영역에서 약 150여개의 기관 파라미터가 정의되어 있으며, CBB에서 파라미터별 별도 처리가 수행되도록 구현되어 있음**

- "별도"