ProFrame WAS
Studio 안내서

ProFrame WAS Version 4.0

Document info
Document name: “ProFrame WAS Studio 안내서”
Document date: 2007-03-01
Manual release version: 0.0.1
Software Version: ProFrame WAS 4.0

차례

그림 목차

매뉴얼에 대하여

매뉴얼의 대상

본 매뉴얼은 ProFrame Studio를 사용하여 시스템을 개발하고하는 개발자를 대상으로 한다.

매뉴얼의 전제 조건

본 매뉴얼은 ProFrame 시스템 개발자를 대상으로 개발작업을 시작하기 전 숙지해야 할 사항을 소개하기 위해 작성되었다. JEUS및 RDB, JAVA 언어에 익숙하고 유닉스 운영체제에 대한 기본적인 지식을 갖춘 개발자를 대상으로 작성되었다.

매뉴얼 구성

이 매뉴얼은 9장으로 구성되어있다.

1. **ProFrame Studio 설치 및 실행**
2. **ProBuilder Project**
3. **Meta 등록**
4. **ProMapper**
5. **DBIO**
6. **EMB**
7. **SO**
8. **BO**
9. **기타**

용어 설명

다음에 소개되는 용어는 본 문서 전체에 걸쳐서 사용되는 용어이다. 용어가 이해하기 어렵거나 명확하지 않을 때는 아래 정의를 참조하기 바란다.

용어	정의
WAS	Web Application Server의 약자로 EJB, Servlet, JSP, Web Service, JMS등의 서비스를 제공한다
PROMAPPER	입출력 전문을 정의하고 전문클래스를 생성하고 전문데이타를 받아서 객체로 변환하거나 객체를 전문데이타로 변환하는 등의 일을 수행한다.
DBIO	DataBase Input/Output의 약자로 데이터베이스를 쉽고 일관된 API로 접근할수 있도록 해주며 업무컴포넌트 개발시에 데이터베이스 벤더에 관계없이 코딩할수 있도록 해준다.
DTO	Data Transfer Object의 약자로 Java Object, DBIO 를 호출하기 위한 IN, OUT 객체로 사용된다.
BO	Business Object의 약자로 비즈니스 기능을 수행하는 재사용성의 단위로서 Service Object에서 레퍼런스 호출되며 정보 처리 및 계산 그리고 DBIO 호출 후 데이터 가공 등을 수행한다
SO	Service Object의 약자로 외부에 오퍼레이션을 노출할 수 있는 리소스 단위로서 단위 트랜잭션 수행의 주로 Business Object를 Orchestration하는 Flow 중심 어플리케이션이다.
EMB	Enterprise Module Bus의 약자로 서비스 모듈이 만들어지면, 그 모듈들을 조합하여 다양한 새로운 서비스를 만들 수 있는 서비스 기반의 아키텍처이다.

연락처

Korea
TmaxSoft Co., Ltd
18F Glass Tower, 946-1, Daechi-Dong, Kangnam-Gu, Seoul 135-708
South Korea
Tel: 82-2-6288-2114
Fax: 82-2-6288-2115
Email: info@tmax.co.kr
Web (Korean): http://www.tmax.co.kr

USA
TmaxSoft, Inc.
560 Sylvan Ave, Englewood Cliffs NJ 07632
USA
Tel: 1-201-567-8266
FAX: 1-201-567-7339
Email: info@tmaxsoft.com
Web (English): http://www.tmaxsoft.com

Japan
TmaxSoft Japan Co., Ltd.
6-7 Sanbancho, Chiyoda-ku, Tokyo 102-0075
Japan
Tel: 81-3-5210-9270
FAX: 81-3-5210-9277
Email: info@tmaxsoft.co.jp
Web (Japanese): http://www.tmaxsoft.co.jp

China
Beijing Silver Tower, RM 1507, 2# North Rd Dong San Huan,
Chaoyang District, Beijing, China, 100027
Tel: 86-10-6410-6148
Fax: 86-10-6410-6144
E-mail : info@tmaxchina.com.cn
Web (Chinese): http://www.tmaxchina.com.cn
ProFrame Studio 설치 및 실행

1.1 설치 전 준비사항

ProFrame Studio는 ProFrame 기반의 다양한 응용프로그램을 개발하는 통합개발 환경을 제공한다. ProFrame Studio의 시스템 요구사항은 아래와 같다.

ⓐ Windows2000 또는 Windows XP

ⓑ 500M 이상의 하드디스크 여유 공간

ⓒ 1024*768 이상 디스플레이 권장

ⓓ 최소 256M, 512M 이상 권장

ⓔ JDK 1.4.2_06 이상

1.2 설치 전 준비사항

ProFrame Studio설치 시 반드시JDK 1.4.2 06이상의 설치 후 JDK 의 PATH 가 환경변수에 제대로 설정이 되어있는지 반드시 확인하도록 한다. JDK 의 환경변수의 PATH 설정은 아래와 같다. 잘못 설정되어 있는 경우 아래와 같이 설치한 JDK의 bin 디렉토리 경로를 PATH의 맨 앞에 설정해 준다.

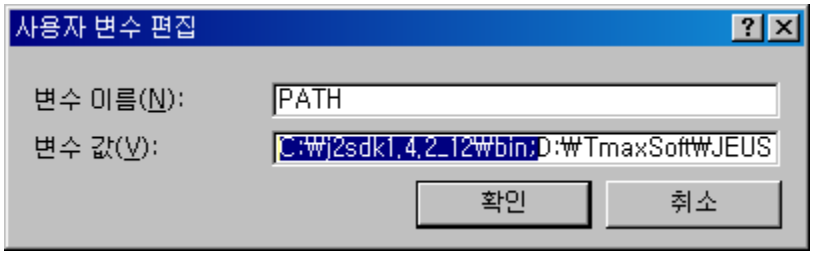


그림 *1 JDK PATH* 설정확인

1.3 ProFrame Studio 실행

ProFrame Studio 설치된 경로에서 proframe.exe 파일을 실행시킨다.

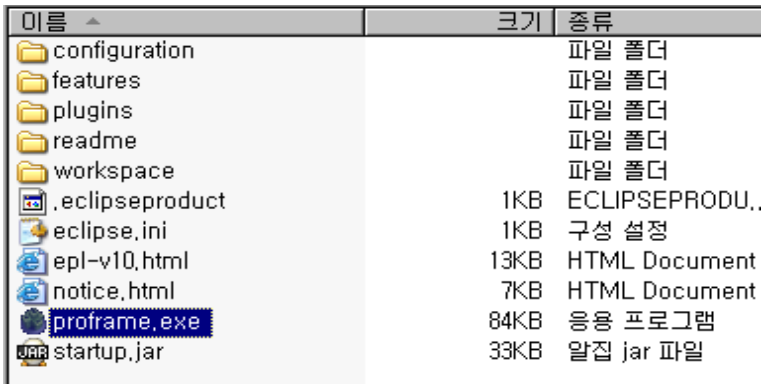


그림 *2 ProFrame.exe* 파일

1.4 ProFrame Studio 작업공간 설정

Proframe.exe를 실행시키면 로컬에 작업공간을 설정하라는 팝업창이 뜨면서 생성 된 Resource를 저장하는 위치를 지정한다.

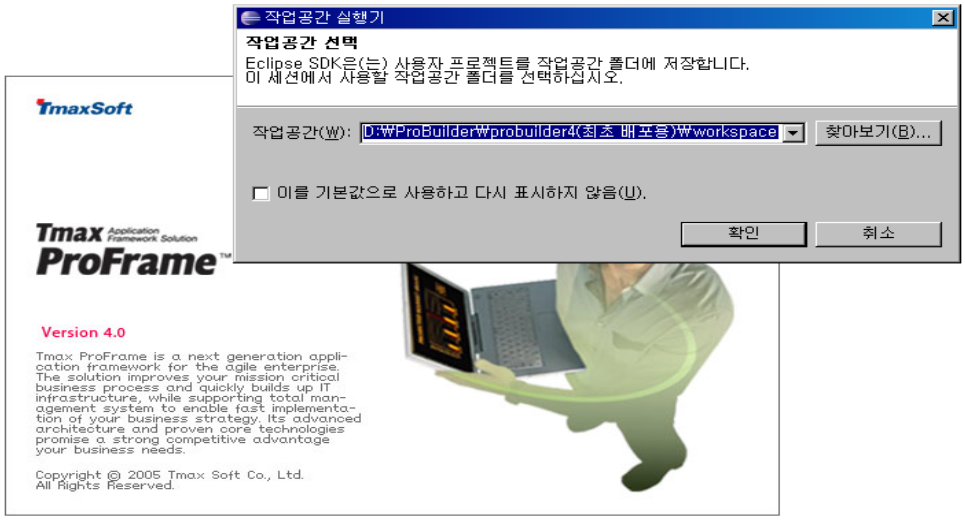


그림 *3* 작업공간 설정

1.5 ProFrame Studio Patch

Proframe version에 맞게 ProFrame Studio는 패치를 해야 하며 Patch를 위해서 매뉴에 도움말 > 소프트웨어 갱신 > 찾기 및 설치를 선택한다.

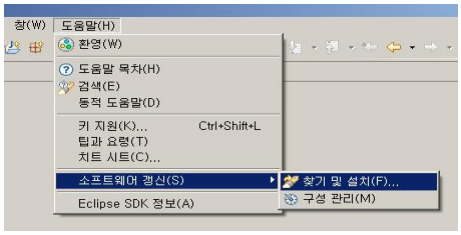


그림 *4* 소프트웨어 갱신

설치할 새 기능에 대해 검색을 선택한다.

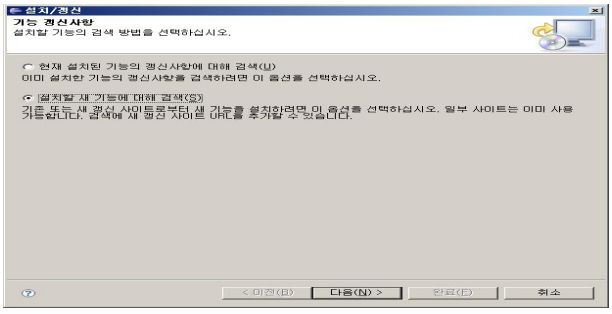


그림 *5* 새 기능 검색

새 아카이브된 사이트 선택한다.

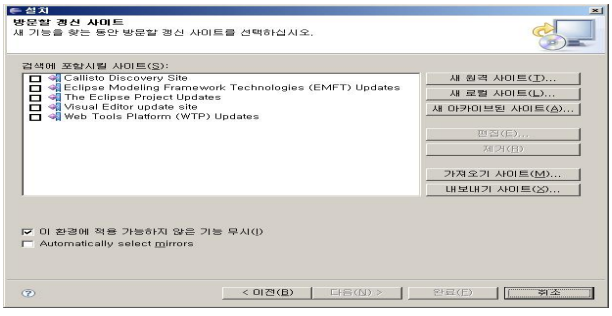


그림 *6* 새 아카이브된 사이트

패치 파일 선택한다.

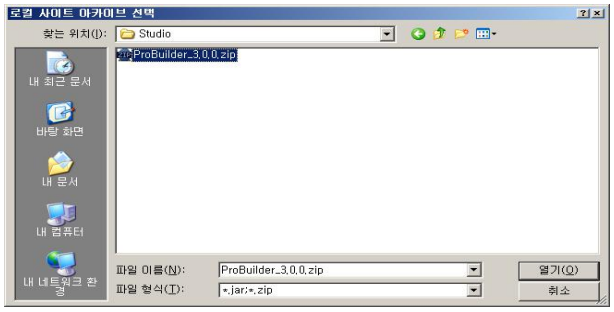


그림 *7* 패치파일 선택

작업 진행 후 ProFrame Studio 자동으로 재실행 된다.

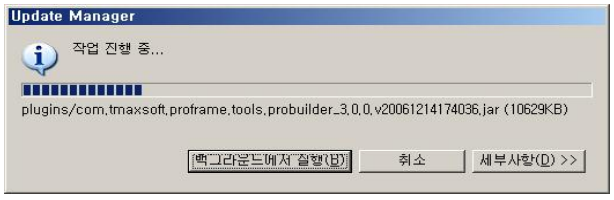


그림 *8 Patch* 진행

1.6 ProBuilder

ProFrame Studio에서 제공되는 ProBuilder는 ProFrame 기반의 개발환경을 제공하며 우측상단의 Perspective 열어 프로빌더 환경으로 변경한다.

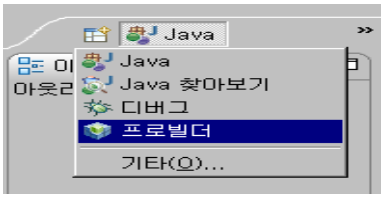


그림 *9* 프로빌더

ProBuilder의 매뉴에 창(W) > Perspective 사용자 정의(Z)를 선택한다.

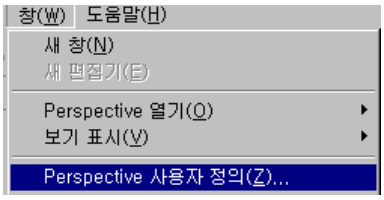


그림 *10 Perspective* 사용자 정의

Batch Object 부터 프로프레임 어프리케이션프로젝트까지 체크박스에 체크를 하여 ProBuilder에서 개발 시 오른쪽 마우스에 등록되어 보다 쉽게 작업을 진행한다.

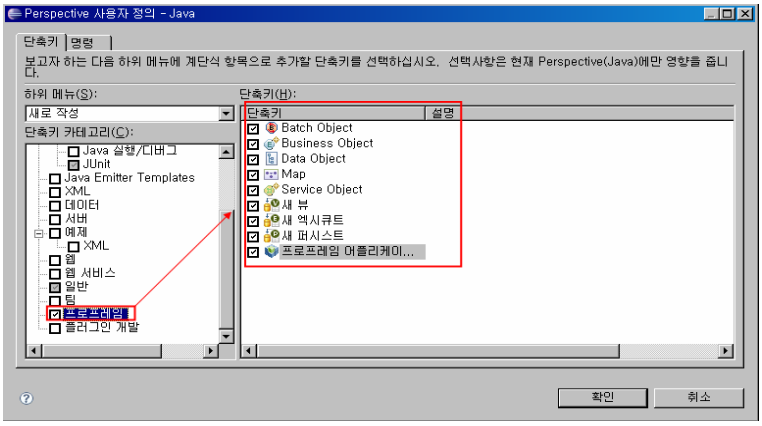


그림 *11* 프로빌더 단축키 등록

Business Object, Service Object에서 사용하는 Object Pool은 창(W) > 보기 표시(V) > 기타 선택한다.

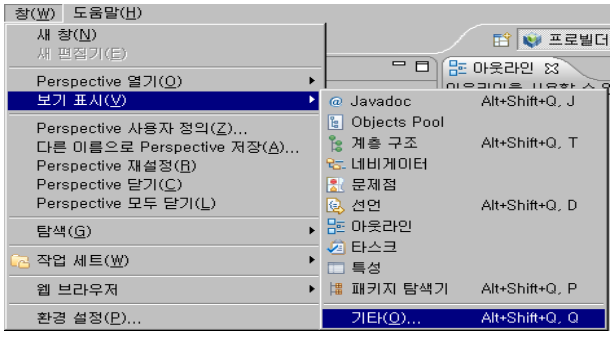


그림 *12 ProBuilder* 보기 표시 선택

프로프레임 > Object Pool을 선택한다.

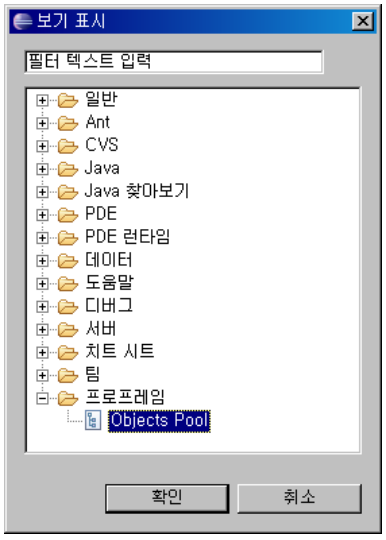


그림 *13 Object Pool* 선택

ProBuidler에서 Object Pool를 확인한다.

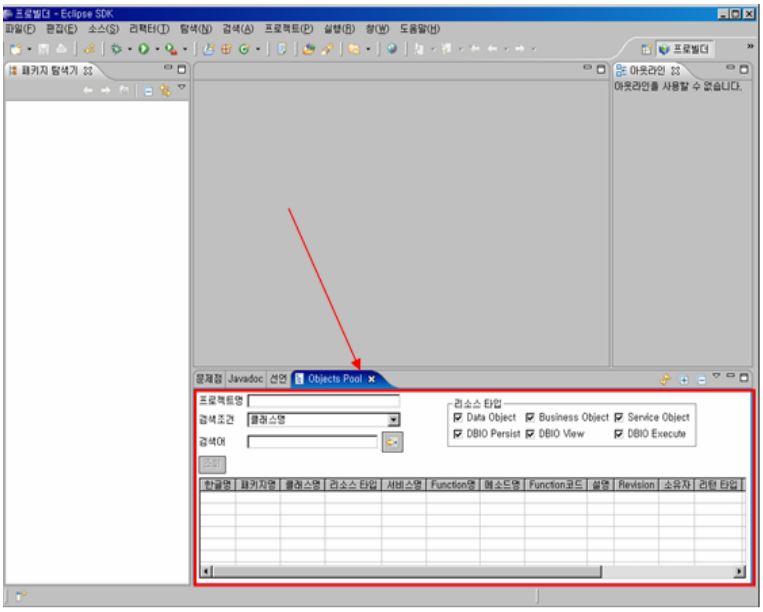


그림 *14 Object Pool* 위치 확인

2 ProBuilder Project

2.1 프로프레임 어플리케이션 프로젝트

ProBuidler에서 작업하는 모든 리소스는 프로젝트에 속해 있어야 한다. 따라서 프로그램을 개발할 때 가장 먼저해야 할 일은 해당 프로그램을 담아둘 프로젝트를 만드는 것이다. 프로젝트를 생성학 위해 파일(F) > 새로작성(N) > 프로프레임 어플리케이션 프로젝트 선택한다.

그림 *15* 프로프레임 어플리케이션 프로젝트

프로프렘임 어플리케이션 프로젝트의 환경설정을 한다.

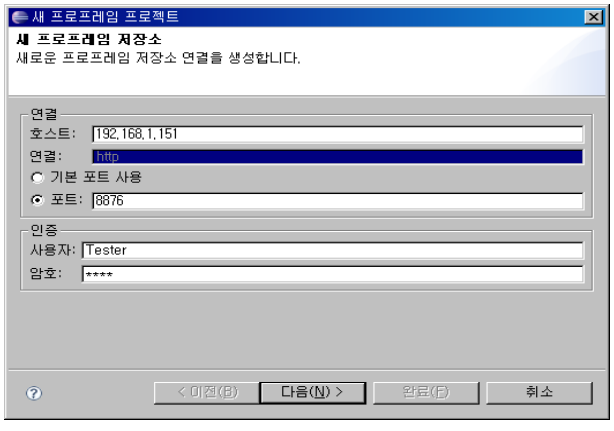


그림 *16* 프로젝트 환경설정

ⓐ 호스트: ProFrame WAS 4.0 Server Ip

ⓑ 포트: 환경파일 PfmDevSvr.xml <configField id="SERVER_PORT 확인

ⓒ 사용자: 환경파일 PfmDevSvr.xml <configField id="USER_ID 확인

ⓓ 암호: 환경파일 PfmDevSvr.xml <configField id="PASSWD 확인

프로프레임 어플리케이션을 기존어플리케이션을 선택하거나 새로 생성할 수 있으며 이 정보는 ProFrame DB Table의 PFM_RESOURCE Table에 정보가 등록된다. (RESOURCE_TYPE Column에 APPLICATION으로 등록)

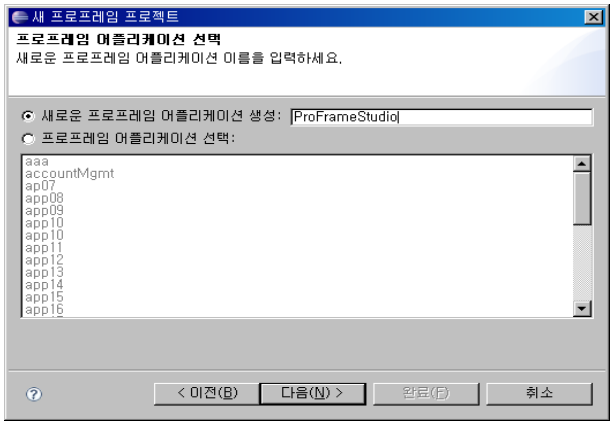


그림 *17* 프로젝트 생성

개발자 PC에서 사용하는 프로젝트 이름으로서 사용자가 임의로 생성 가능

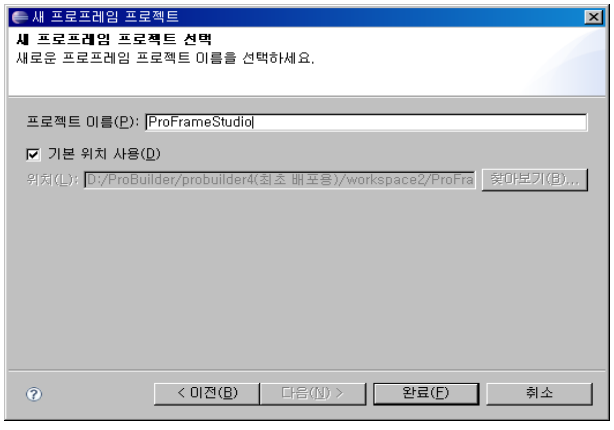


그림 *18* 프로젝트 생성

프로프레임 어플리케이션 생성시 패키지 탐색기에 아래와 같이 폴더가 생성된다.

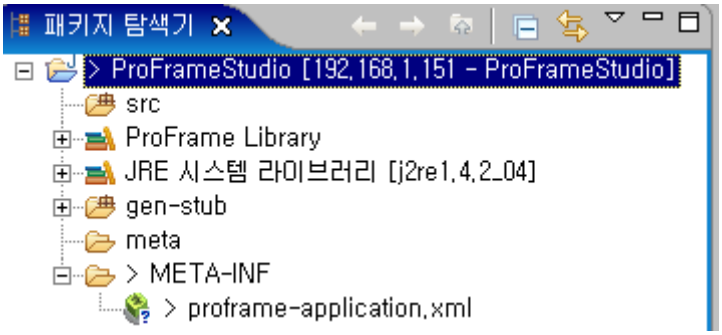


그림 *19* 패키지 탐색기

ⓐ src: Non ProFrame Java 프로그램 생성 시 폴더에 저장

ⓑ ProFrame Library

ⓒ JRE 시스템 라이브러리

ⓕ gen-stub: 웹서비스 작성 시 폴더가 생성되며 Default 값은 아님

ⓓ meta: ProFrame 서비스 작성 시 폴더에 저장

ⓔ META-INF: ProFrame 서비스작성 내역을 xml 파일로 저장

ProBuilder 내에서 제공하는 작성 파일 Type은 아래와 같다.

Type	Description	확장자
DTO	Data Transfer Object	dto
Map	BO2SO, SO2BO, BO2DBIO, DBIO2BO Mapping 제공	map
Message	서비스 입력 전문제공	msg
DBIO	DB 전문 생성	persist, view, execute
BO	Business Object	md
SO	Service Objcet	Sd

그림 *20 ProBuilder* 제공 작성 파일 *Type*

파일을 작성한 후에는 반드시 오른쪽 마우스에 팀(E) > 커밋(C)을 선택하여 컴파일한다. 그리고 파일의 확장자 뒤에 붙어있는 숫자는 Revision으로 커밋을 할 때 마다 올라간다. 또한 저장 안 된 파일은 파일명 앞에 > 표기되며 커밋 후에서야 > 표기가 없어진다.

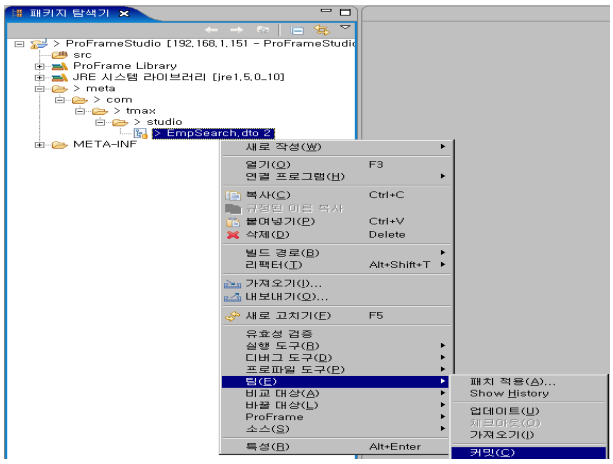


그림 *21* 커밋

커밋을 누르면 Probuilder에서 xml파일 생성하여ProFrame 통합서버에 전달하고
통합서버에서 컴파일하여 class파일을 생성한다.

Ex>PFM_HOME/pfm/temp_srcs/PFM_IO/xml/EmpSearch.xml

Ex>PFM_HOME/pfm/temp_srcs/PFM_IO/com/tmax/studio/com/tmax/studio/EmpSearch.java

Ex>PFM_HOME/pfm/classes/PFM_IO/com/tmax/studio/EmpSearch.class

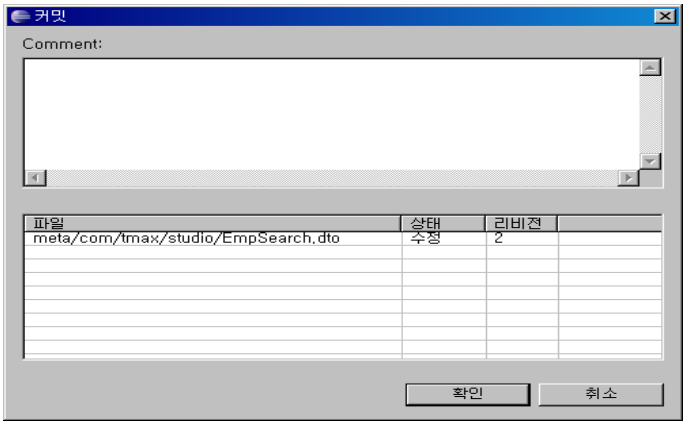


그림 *22* 커밋 확인

3 Meta 등록

ProBuidler의 ProMapper에서 사용 될 Meta 등록은 ProFrame WAS Admin 안내서를 참조한다.

4 ProMapper

ProMapper는 ProFrame에서 제공하는 I/O작성 및 매핑 툴이다. ProMapper는 DTO, Map, Message를 만들어 준다.

4.1 ProMapper의 Resource Type

ProMapper에서는 세가지로 나누어 리소스 타입을 정의하고 있다. 이 중에서 Message는 Structure를 기반으로 만들어진다. 내용은 아래와 같다.

Type	Description	확장자
DTO	Data Transfer Object 로 Java Object, DBIO 를 호출하기 위한 IN, OUT 객체로 사용	dto
Messag	서비스의 입출력에 사용되는 전문을 의미	Msg
Map	BO2SO, SO2BO, BO2DBIO, DBIO2BO Mapping 제공	map

그림 *23 ProMapper*의 *Type*

4.2 DTO

4.2.1 기본 DTO

Data Transfer Object 로 Java Object, DBIO 를 호출하기 위한 IN, OUT 객체로 사용된다. 오른쪽 마우스 새로작성(W) > Data Object 선택한다.

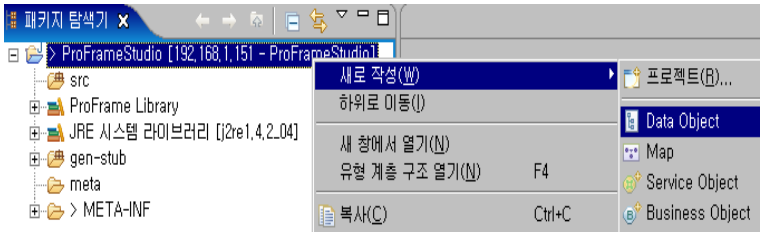


그림 *24 DTO* 생성

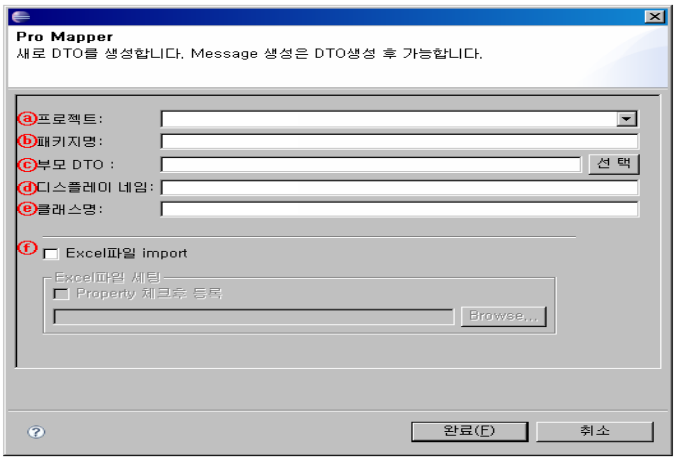


그림 *25 DTO* 생성

ⓐ 프로젝트: 서비스가 작성 될 프로젝트를 선택

ⓑ 패키지명: 패키지명을 등록

ⓒ 부모DTO: 상속 받을 DTO를 선택하여 부모 DTO 필드 사용 가능

ⓓ 디스플레이네임(논리명): 한글 영문 표기가 가능 패키지 내에서 유일

ⓔ 클래스명(물리명): 메타에 등록되는 이름으로 동일 패키지 내에서 유일

ⓕ Excel파일 Import: DTO구조가 정의 된 Excel 파일로 Import하여 DTO 구조를 정의

DTO 구조정의 화면은 아래와 같다.

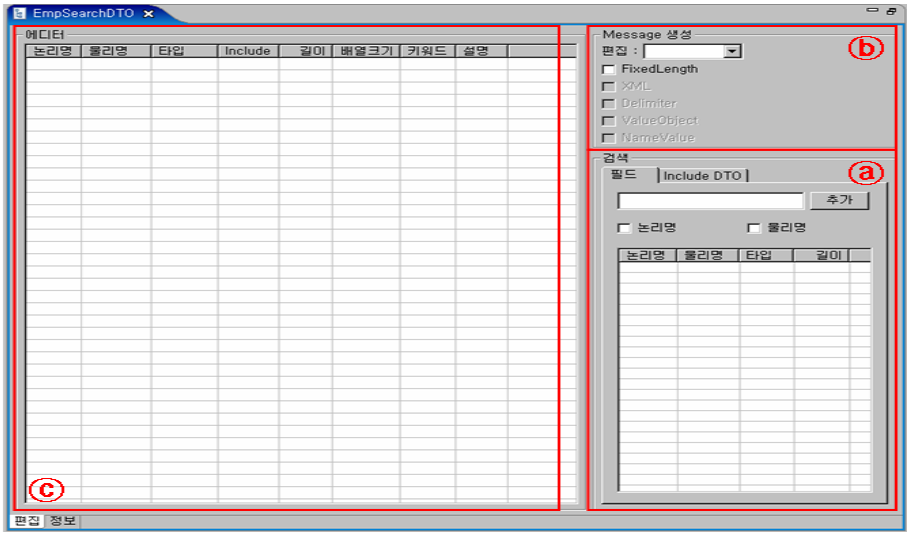


그림 *26 DTO* 구조정의 화면

ⓐ 메타 검색, DTO 검색으로 “%%%” 입력하면 등록된 모든 메타, DTO를 검색할 수 있다.

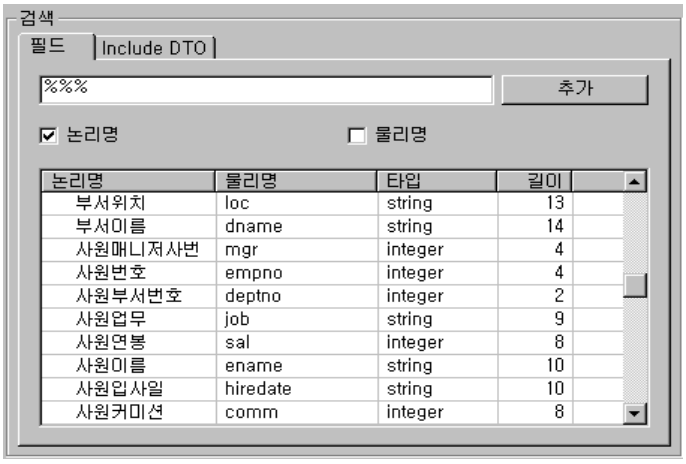


그림 *27 DTO* 구조정의 화면

ⓑ Message Type체크박스를 선택하면 후에 Message 자동생성 되며 패키지 탐색기에서 확인 할 수 있다.

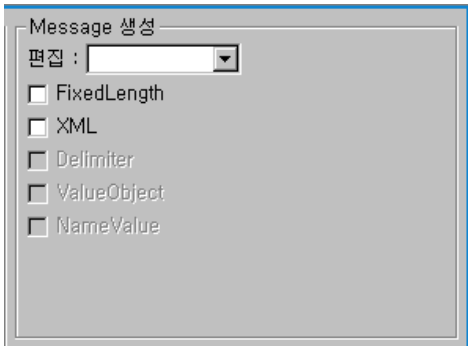


그림 *28 message type* 선택화면

ⓒ DTO 구조 정의를 한다.

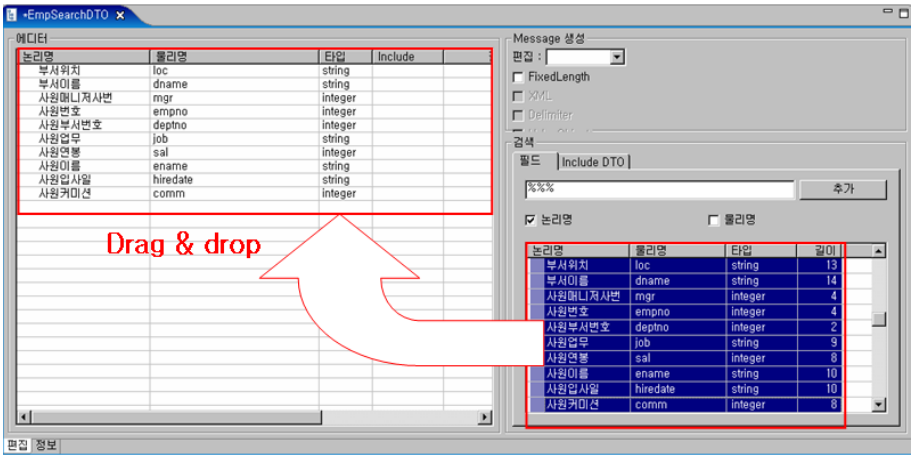


그림 *29 DTO* 구조정의 화면

DTO 구조를 정의한 커밋을 한 후 정보 창에서 생성 된 DTO정보를 확인할 수 있다.

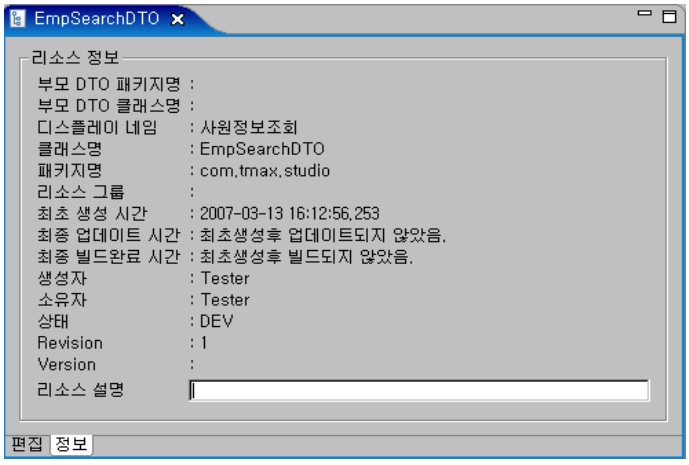


그림 *30 DTO* 구조정의 화면

4.2.2 DBIODTO

서비스 작성시 DBIODTO는 반드시 생성되어야 하며 목적은 DBIO 호출시 In, Out를 위함이며 생성 및 설정은 DTO와 같지만 DTO 구조정의 시 dbio 관련 메타를 만드시 포함해야한다.

논리명	물리명	타입
변경건수	dbio_affected_count_	integer
		integer

페이징건수	dbio_fetch_size_	
페이징시컨스	dbio_tetch_seq_	integer
전제건수	dbio_total_count_	integer

그림 *31 DBIO* 관련 메타 리스트

DBIO 입출력 관련 메타데이터를 “dbio”를 삽입하여 검색한다.

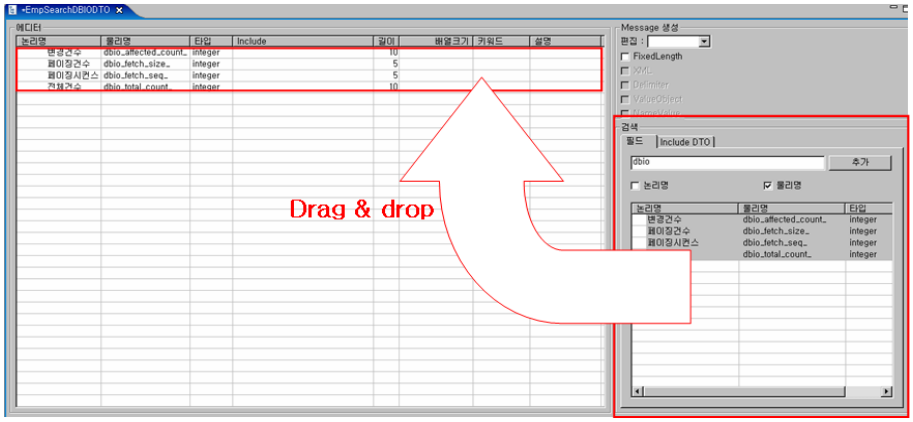


그림 *32 DBIODTO* 필수입력 메타

includeDTO를 검색하여 관련 DTO를가져온 후 물리명, 논리명을 입력한다. 단 반드시 배열크기는 “dbio_total_count_” 를 입력한다.

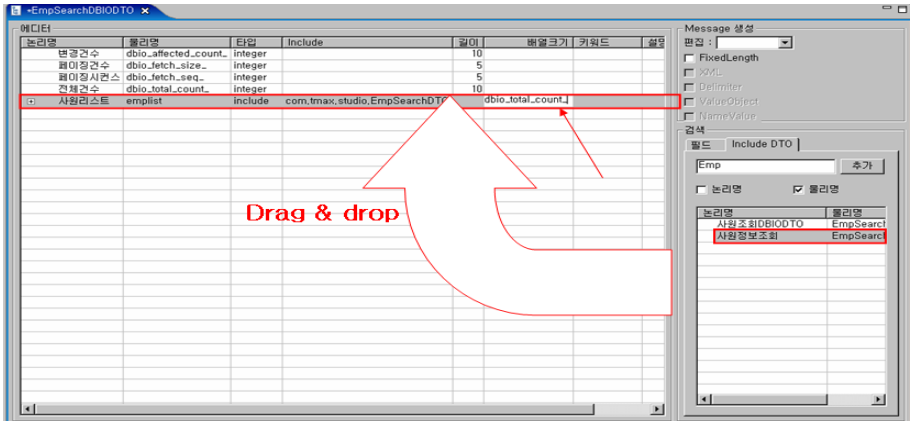


그림 *33 DBIODTO* 구조정의

기본적으로 DTO은 편집이 불가능하지만 편집시 필드의 타입과 길이를 수정할 수 있도록 환경설정으로 변경하면 된다. Probuilder 폴더의 plugins 폴더 아래의 com.tmax.proframe.promapper.was.jar 의env.properties의 파일에 APPLY_CAN_MODIFY_FIELD_TYPE, APPLY_CAN_MODIFY_FIELD_LENGTH의 값이 각각 true일 경우 타입과 길이가 변경이 가능하다.

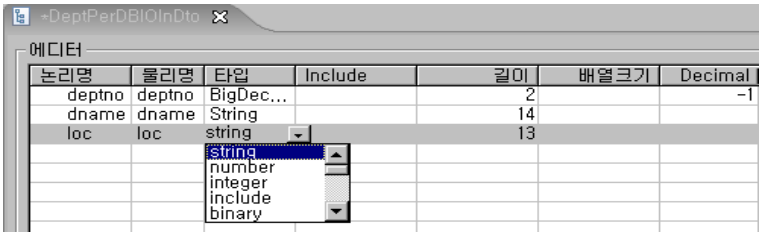


그림 *34 DTO* 구조변경 화면

변경된 DTO Property는 dev_property table에 sequence Number가 변경되어 저장되며 Admin에서는 Defaut 설정만이 검색이 된다.

4.3 Message

Message는 채널단 단말에서 서비스 입출력을 정의하는 것이다. DTO 생성 시 Message 생성부분에서 FixedLength, XML Type으로 Messgae를자동 생성할 수 있으나 EMB Desinger에서 호출, 편집이 불가능하다.

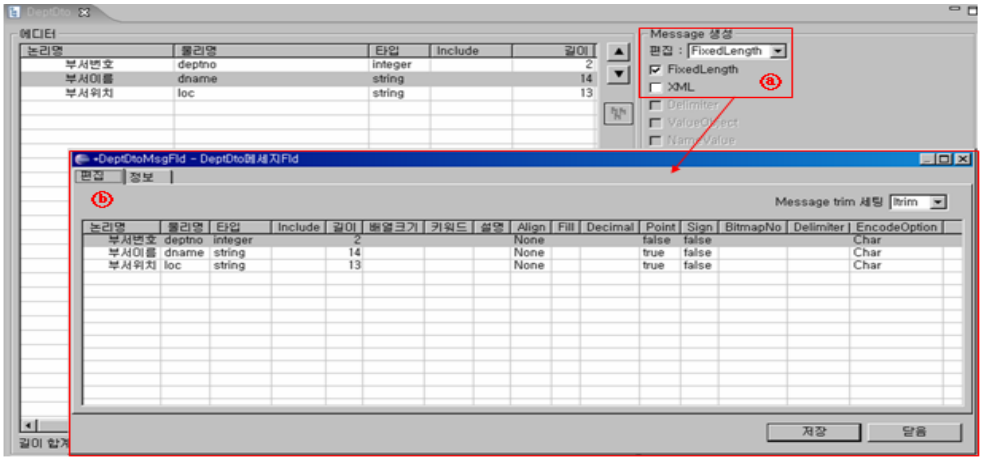


그림 *35 message* 편집

ⓐ Message 생성: Message 타입 선택 및 생성

ⓑ Message 편집창: 각각의 필드를 선택하여 편집 가능

4.4 MAP

Business Object 와 Service Object 사이의Mapping 혹은 Business Object와DBIO간의 Mapping 등을 2가지 타입의 Mapping으로 정의한다.

Map Type	설명	확장자
bypass	입출력이 동일한 경우	map
transformation	입출력이 상이한 경우	map

그림 *36 MAP Type*

Map 생성 방법은 오른쪽마우스 새로작성(W) > Map을 선택한다.

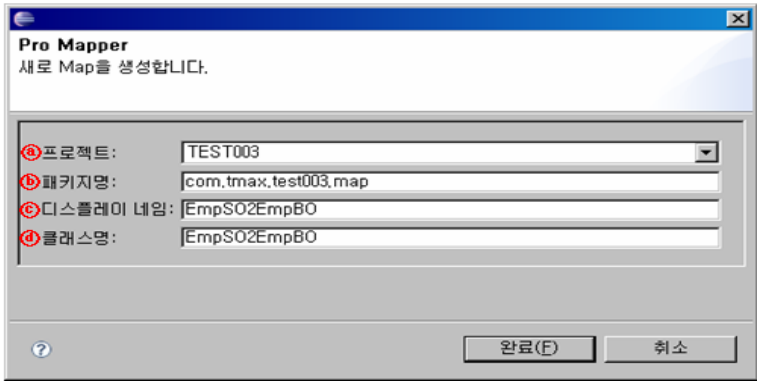


그림 *37 MAP* 생성

ⓐ 프로젝트: 서비스가 작성 될 프로젝트를 선택

ⓑ 패키지명: 패키지명을 등록

ⓒ 디스플레이네임(논리명): 한글 영문 표기가 가능 패키지 내에서 유일

ⓓ 클래스명(물리명): 메타에 등록되는 이름으로 동일 패키지 내에서 유일

Map 편집

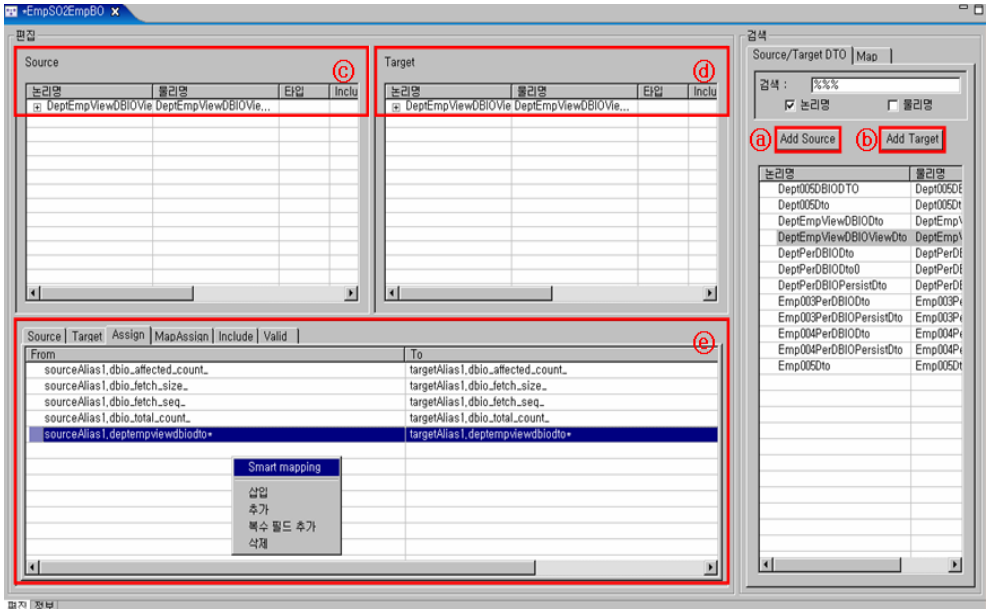


그림 *38 MAP* 편집

ⓐ Add Source: 버튼을 선택하여 ⓒ에 DTO 추가

ⓑ Add Target: 버튼을 선택하여 @에 DTO 추가

ⓒ Source

ⓓ Target

ⓔ Map 편집창

Source 정의

Target 정의

Assign: Smart Mapping 기능(필드명이 같은것끼리 자동맵핑) 제공, Drag & Drop

MapAssign

Include

Valid

5 DBIO

5.1 DBIO 개요

DBIO는 ProFrame에서 제공하는 데이터베이스 접근에 관한 표준적인 방법으로 데이터베이스의 데이터를 Control 하는 Class이다.

5.2 DBIO의 특징

DBIO의 특징은 아래와 같다.

ⓐ DB 접근 방식을 표준화한다.

ⓑ DB 접근 시 일괄된 에러 처리를 지원한다.

ⓒ DBMS 벤더에 대한 의존성을 배제할 수 있다.

ⓓ 중복된 데이터 베이스 프로그램을 최소화한다.

ⓔ 업무로직과 DB 접근 로직에 신경 쓰지 않고 업무로직에 전념할 수 있다.

ⓕ Tool을 이용하여 생산성 향상시키고 오류 가능성을 줄여준다.

5.3 DBIO Type

DBIO에서는 SQL의 사용 유형에 따라 크게 세가지로 나누어 Persist, View, Execute 라는 타입을 정의하고 있다. 각 타입의 특징은 다음과 같다.

ⓐ Persist: 단일 테이블에 대한 SELET/UPDATE/INSERT/DELETE의 기능을 수행한다.

ⓑView: Persist에서는 지원할 수 없는 여러 테이블에 대한 JOIN등의 SELECT를 수행한다.

ⓒExecute: Persist 에서 지원할 수 없는 INSERT/UPDATE/DELETE등의 기능을 수행한다.

5.3.1 Persist

단일 테이블에 대한 SELET / UPDATE / INSERT / DELETE의 기능을 수행하는 DBIO이다. 오른쪽마우스에 새로작성(W) > 새 퍼시스트를 선택한다.

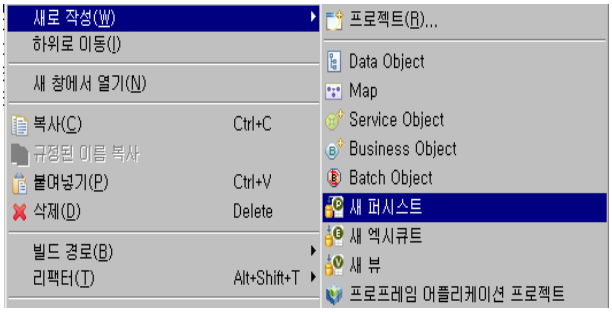


그림 *39 Persist* 생성

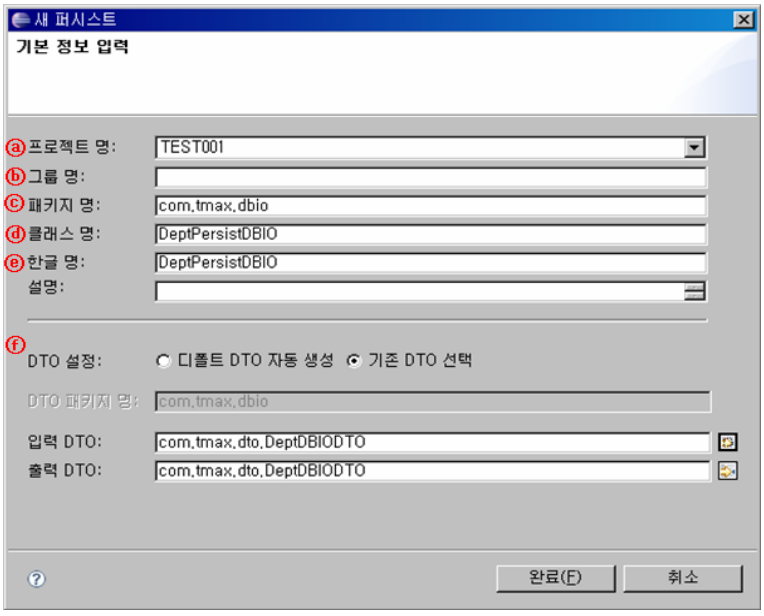


그림 *40 Persist* 기본정보 입력

ⓐ 프로젝트: 서비스가 작성 될 프로젝트를 선택

ⓑ 그룹명:

ⓒ 패키지명: 패키지명을 등록

ⓓ 클래스명(물리명): 메타에 등록되는 이름으로 동일 패키지 내에서 유일

ⓔ 한글명(논리명): 한글 영문 표기가 가능 패키지 내에서 유일

ⓕ DTO 설정: 기존 DTO 선택 →  버튼을 눌려 기작성된 DBIIO DTO를 선택 (디폴트DTO자동생성은 View에서 설명)

Persist 설정 Overview 화면은 아래와 같습니다.

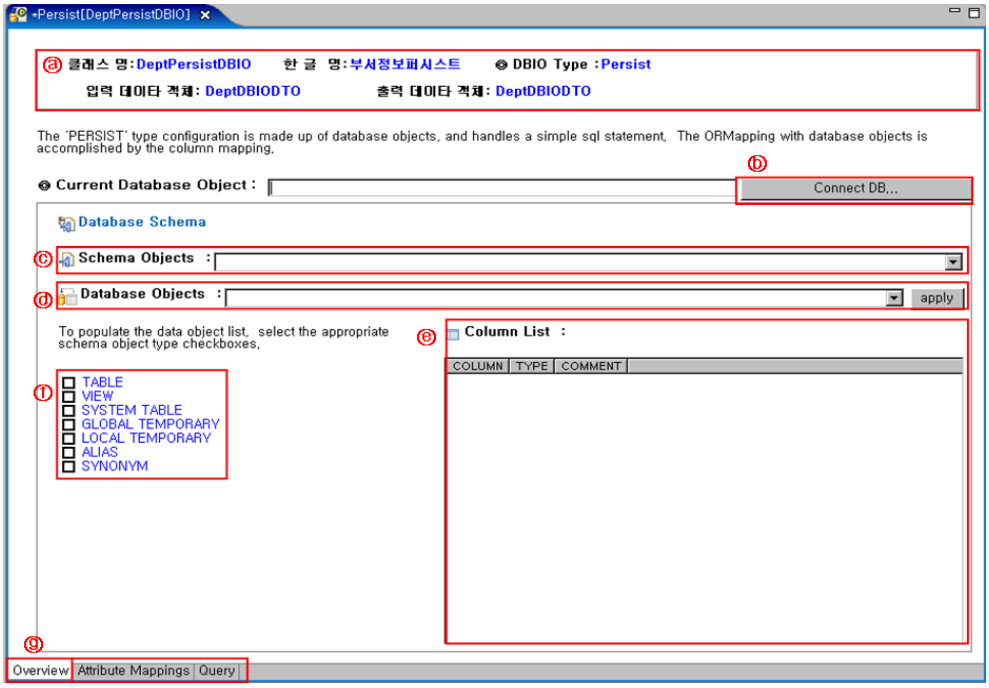


그림 *41 Persist Overview*

ⓐ Persist 설정정보 표시창

ⓑ 버튼을 선택하여 DB: 연결

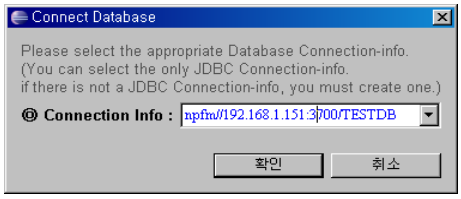


그림 *42* 디볼트 *DTO* 생성

ⓒ DB Schema 선택

ⓓ DB Table 선택 후 Apply 버튼을 누른다.

ⓔ 선택 테이블에 대한 Column 정보

ⓕ Schema Object Tpye 선택 (Default: Table)

ⓖ Nevigetor

Persist 설정 Attribute 화면은 아래와 같습니다.

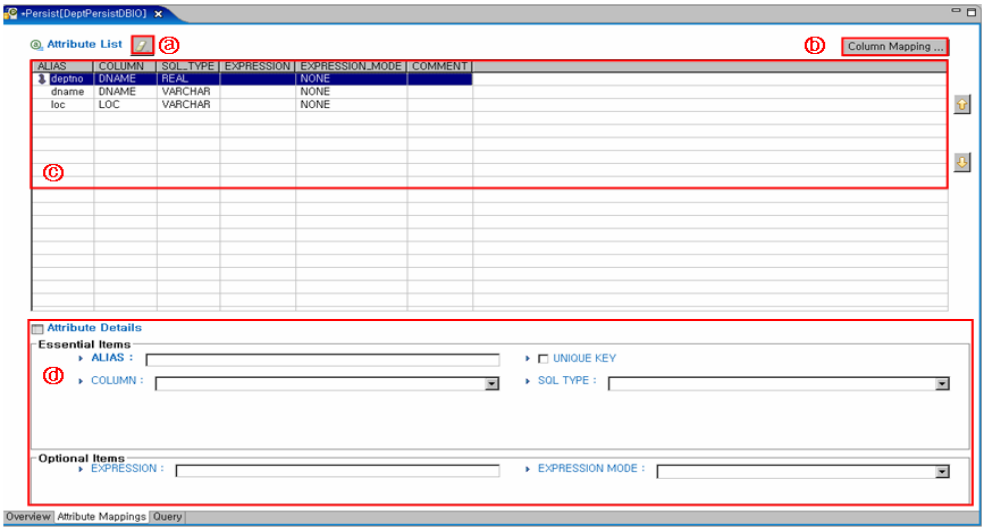


그림 *43 Attribute Mapping*

ⓐ Column에 부여된 속성을 삭제

ⓑ 버튼을 선택하여 사용할 Column을 선택 (Available → Selected)

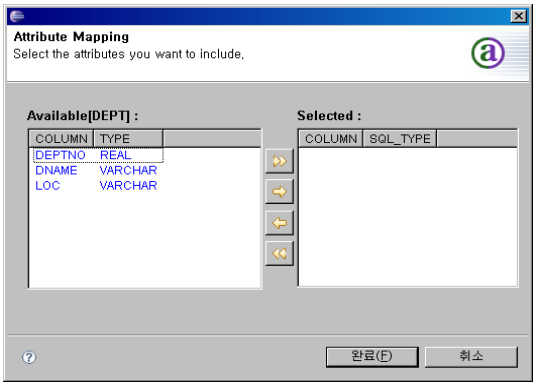


그림 *44 Column*선택

ⓒ Table Column 정보

ⓓ 각각의 Column에 대한 속성 부여

Persist 설정 Query 화면은 아래와 같습니다.

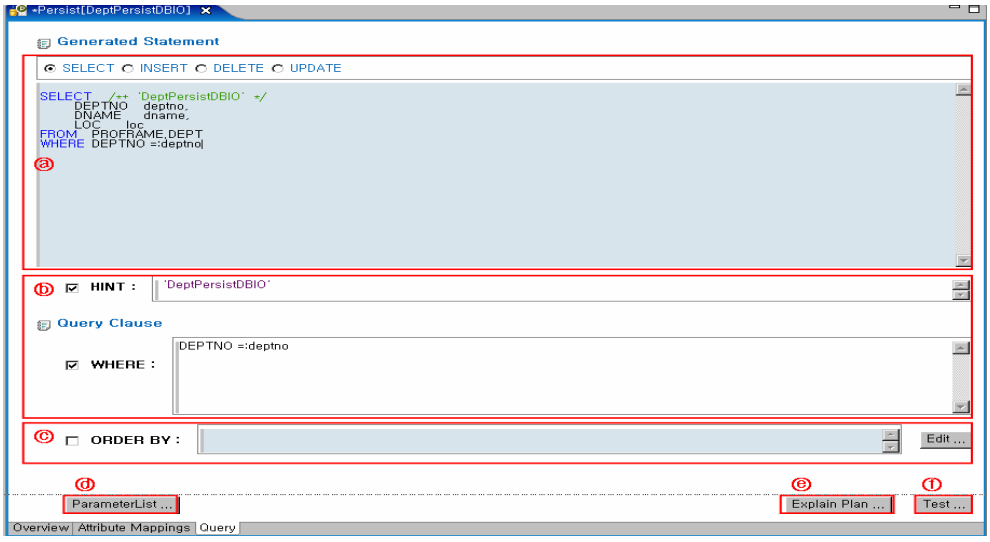


그림 *45 Query*

ⓐ 생성 된 Query list 목록 (콤보박스 선택에 따라 Query의 성격이 변경되지 않는다)

ⓑ HINT: 테이블 주석처리 / WHERE: Query 조건절 입력

ⓒ ORDER BY: Query 결과의 정렬타입을 Edit 버튼을 눌러 설정

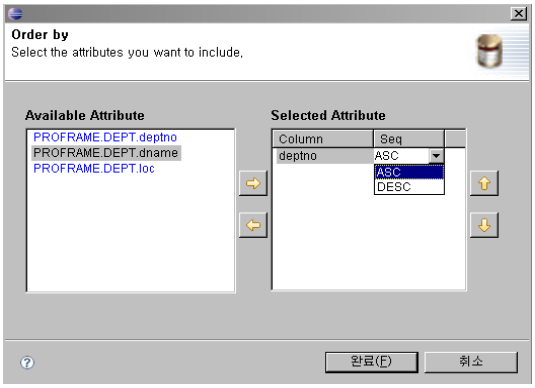


그림 *46 order by* 설정

ⓓ Parameter List를 선택하여 Query 데이터를 입력한다.

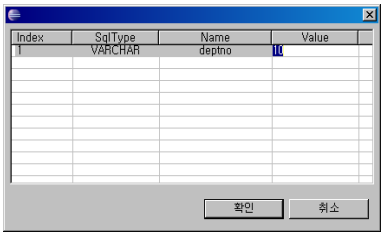


그림 *47 Parameter* 입력

ⓓ Explain Plan

ⓓ Test버튼을 눌러 입력값에 대한 결과값이 나온다.

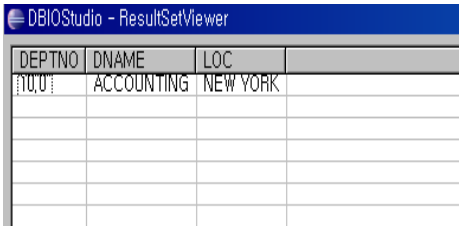


그림 *48* 결과

5.3.2 Viewer

Persist에서는 지원할 수 없는 여러 테이블에 대한 JOIN등의 SELECT를 수행하는 DBIO이다. 프로젝트에 오른쪽마우스 새로작성(W) > 새 뷰 선택한다.

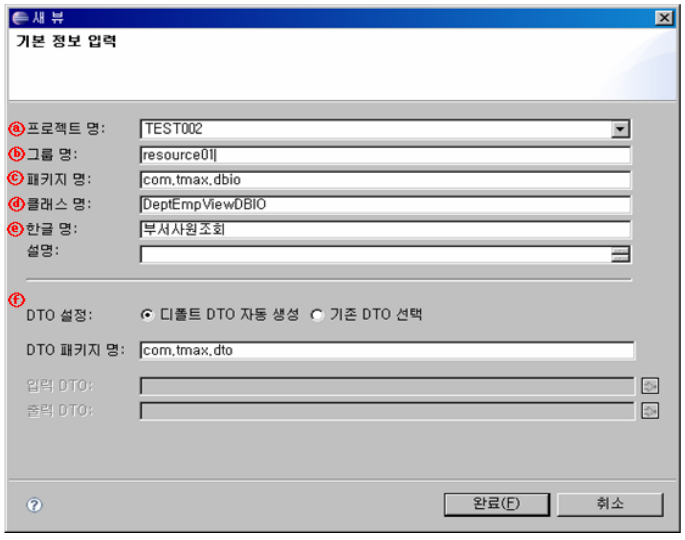


그림 *49 Viewer* 기본정보 입력

ⓐ 프로젝트: 서비스가 작성 될 프로젝트를 선택

ⓑ 그룹명:

ⓒ 패키지명: 패키지명을 등록

ⓓ 클래스명(물리명): 메타에 등록되는 이름으로 동일 패키지 내에서 유일

ⓔ 한글명(논리명): 한글 영문 표기가 가능 패키지 내에서 유일

ⓕ DTO 설정: 디폴트DTO 선택한 후 DTO 패키지명을 입력

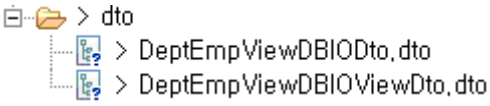


그림 *50 Viewer* 디폴트 *DTO* 자동생성 결과

View OverView 화면

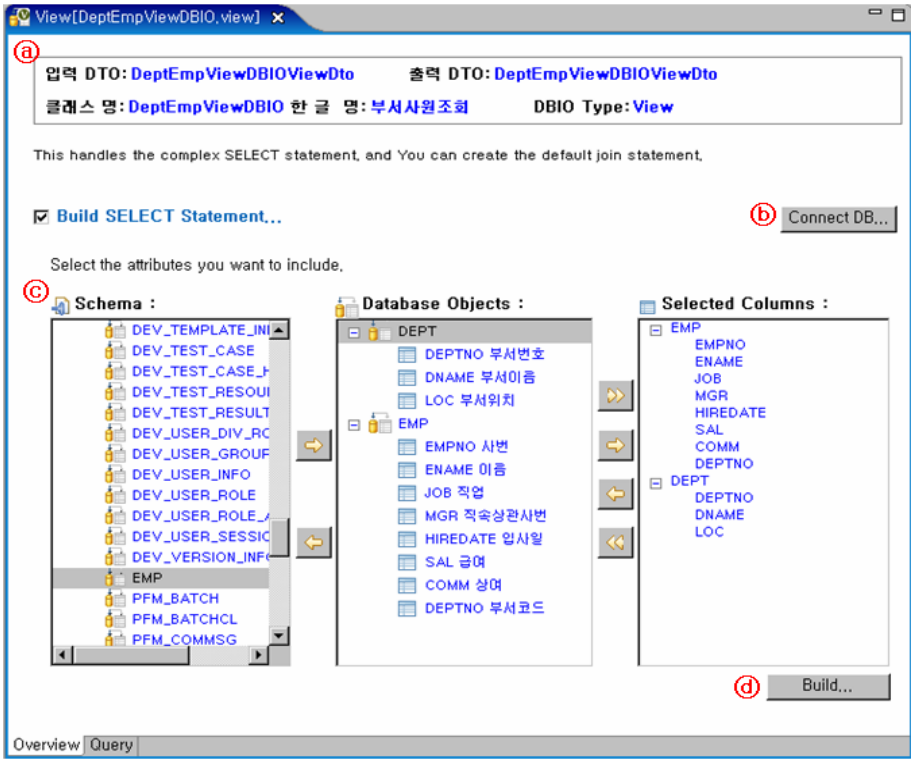


그림 *51 Viewer Overview* 화면

ⓐ View 설정정보 표시창

ⓑ 버튼을 선택하여 DB: 연결

Ⓒ DB Schema, Object, Columns를 차례대로선택

ⓓ 선택된 Columns의 대한 Query 생성 버튼

View Query 화면

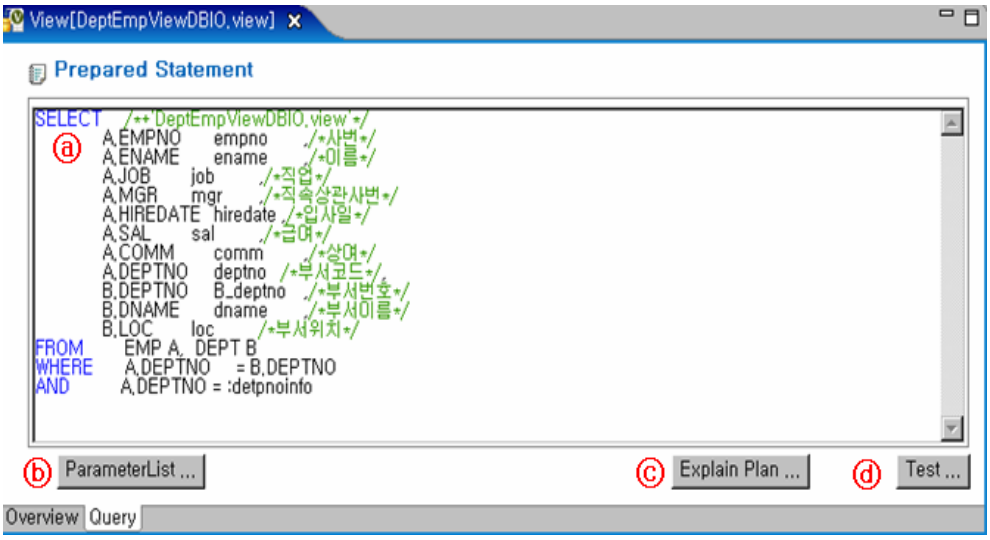


그림 *52 Viewer Query* 화면

ⓐ 자동 생성된 Query 및 Query 입력창

ⓑ Parameter List를 선택하여 Query 데이터를 입력한다.

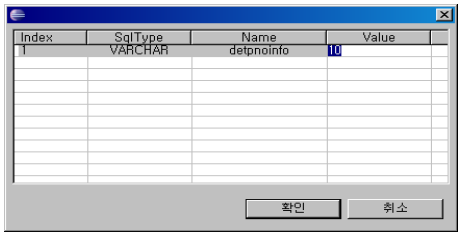
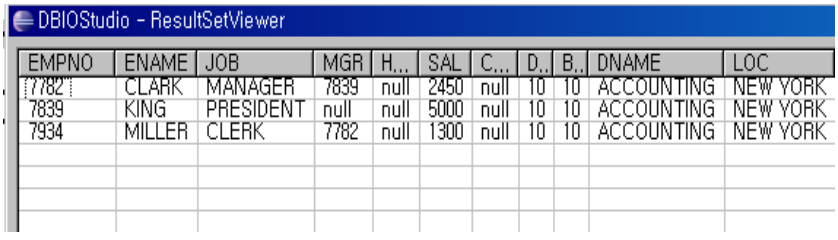


그림 *53 Parameter* 입력

ⓒ Explain Plan

ⓓ Test버튼을 눌러 입력값에 대한 결과값이 나온다.



EMPNO	ENAME	JOB	MGR	H...	SAL	C...	D..	B..	DNAME	LOC
7782	CLARK	MANAGER	7839	null	2450	null	10	10	ACCOUNTING	NEW YORK
7839	KING	PRESIDENT	null	null	5000	null	10	10	ACCOUNTING	NEW YORK
7934	MILLER	CLERK	7782	null	1300	null	10	10	ACCOUNTING	NEW YORK

그림 *54* 결과

5.3.3 Excute

Persist 에서 지원할 수 없는 INSERT / UPDATE / DELETE 등의 기능을 수행하는 DBIO이다. 오른쪽마우스 > 새로작성(W) > Execute를 선택한다.

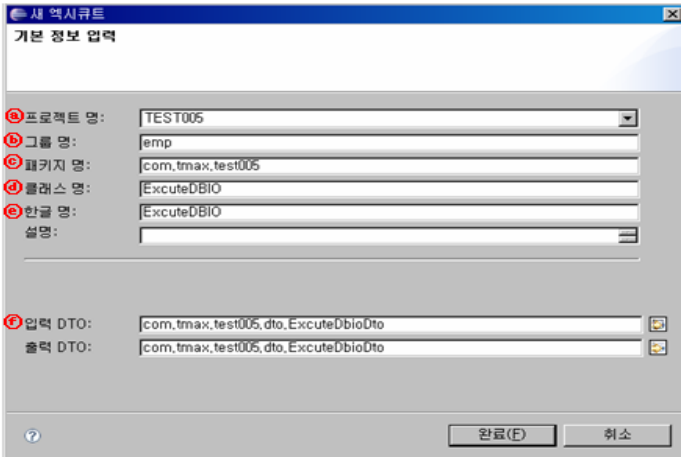


그림 *55 Excute* 기본정보 입력

ⓐ 프로젝트: 서비스가 작성 될 프로젝트를 선택

ⓑ 그룹명:

ⓒ 패키지명: 패키지명을 등록

ⓓ 클래스명(물리명): 메타에 등록되는 이름으로 동일 패키지 내에서 유일

ⓔ 한글명(논리명): 한글 영문 표기가 가능 패키지 내에서 유일

ⓕ DTO 설정

Excute OverViewer 화면

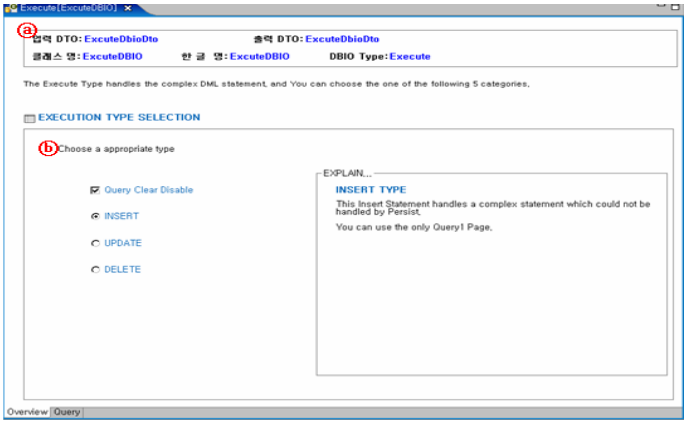


그림 *56 Excute* 기본정보 입력

ⓐ ExcuteDBIO 기본정보

ⓑ 각각의 Query에 대한 Explain

Excute Query 화면

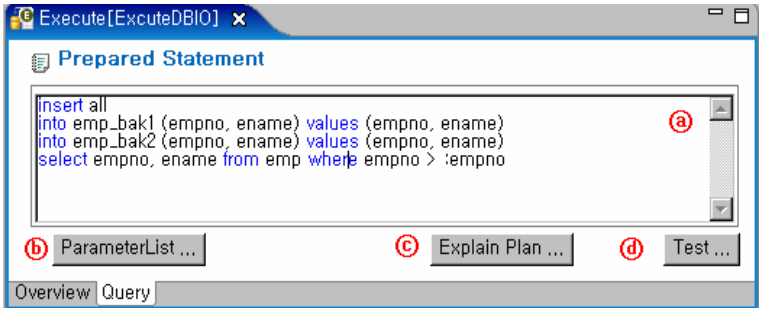


그림 *57 Excute Query*

ⓐ Query: 사용 할 Query를 직접 기입

ⓑ ParmeterList: 데이터를 입력

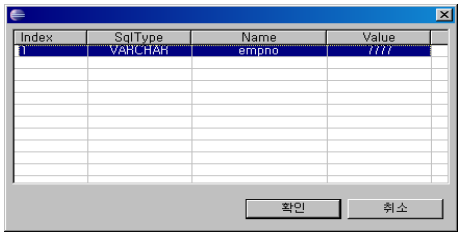


그림 *58 Excute Parmeter* 입력

ⓒ Explain Plan

ⓓ TEST: Query에 따라 임시로 반영된 결과 Row Count를 팝업으로 알림

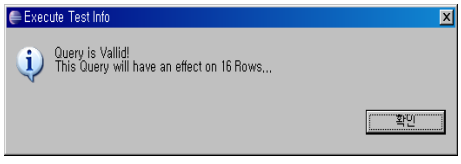


그림 *59 Excute* 결과

6 EMB

Enterprise Module Bus의 약자로 서비스 모듈이 만들어지면, 그 모듈들을 조합하여 다양한 새로운 서비스를 만들 수 있는 서비스 기반의 아키텍처이다.

6.1 EMB Designer

EMB Designer는 서비스 모듈 생성, 수정, 변경을 Graphic하게 제공하며 BO, SO 생성 시 사용한다.

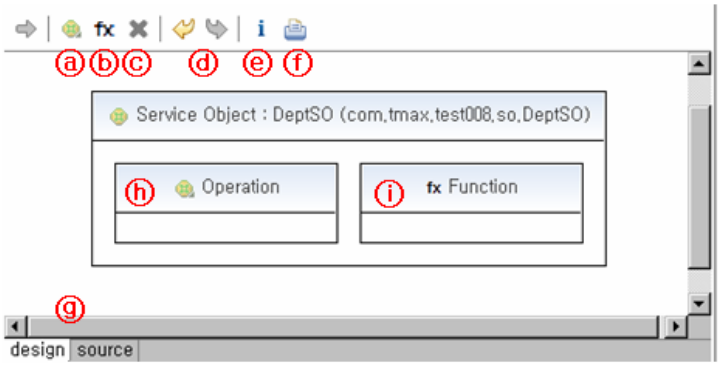


그림 *60 EMB Desginer*

ⓐ Add an Operation: Public Method 생성

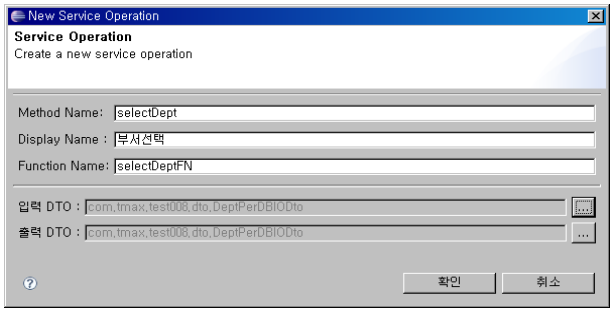


그림 *61 Public Method* 생성

Operation 생성 후 특성탭

특성	설명
General	Method Name, Display Name, Function Name(Admin에 서비스 등록시 Display) 설정
In/Out	Input/OutPut DTO 설정
Pre Processing	선처리 설정
Pro Processing	후처리 설정

ⓑ Add a Virtual function: Private Method 생성

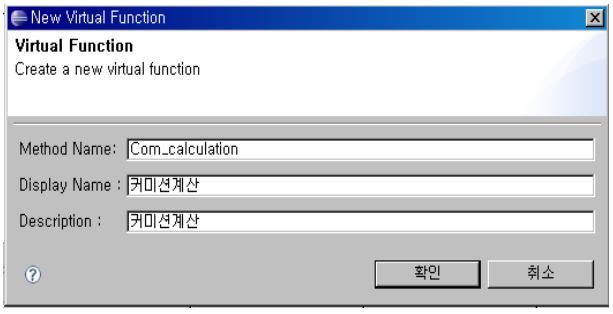


그림 *62 Private Method* 생성

Virtual function 생성 후 특성탭

특성	설명
General	Method Name, Display Name, Description 설정
Signature	Virtual Function의 Return type을 정의
Exception	예외처리

ⓒ Delete: 생성된 Method 삭제

ⓓ undo, redo: 뒤로가기, 앞으로가기

ⓕ 자바소스생성결과

ⓖ Design, Source 선택화면

ⓗ Public Method 생성 결과

ⓘ Private Method 생성 결과

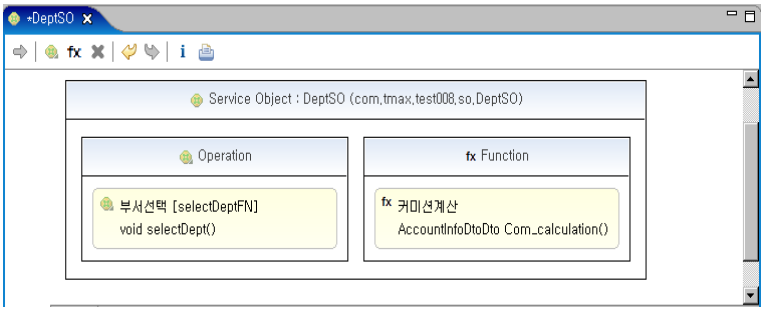


그림 *63 Method* 생성 결과

생성 된 Public Method를 더블 클릭하게 되면 EMB-Designer Edit창으로 넘어간다.

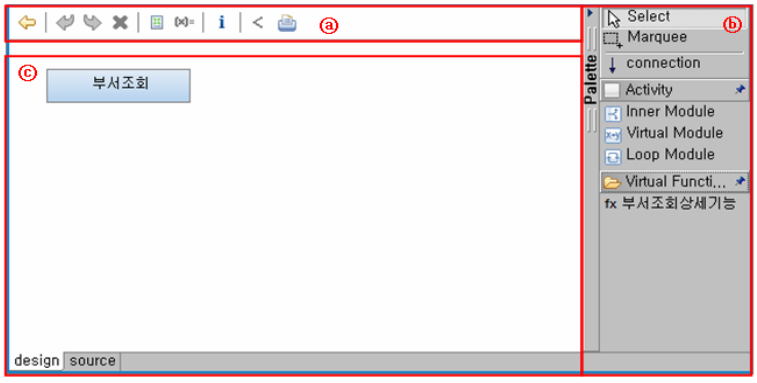


그림 *64 EMB-Design* 편집창

ⓐ EMB Design 편집도구

ⓑ EMB Design Tool

Tool		설명
Select		에디터 창에 떠 있는 리소스, 다이어그램 선택 시
Marquee		다수의 다이어그램을 선택 시
Connection		서비스 또는 Flow를 나타내기 위한 다이어그램의 연결 시
Activity	Inner Module	EMB Design에서 제공하는 기본 모듈들
	Virtual Module	
	Loop Module	
Virtual Function		Private Method 생성 시

그림 *65 EMB Design Tool*

ⓒ EMB Design Viewer: Object Pool에서 모듈을 검색하여 Drag & Drop으로 생성 편집이 가능하며 일반적으로 하드코딩을 허용하지 않으며 모듈에 따라 지정 된 일부 구간에서만 코딩을 허용한다.

Object Pool에서 모듈 Drag & Drop

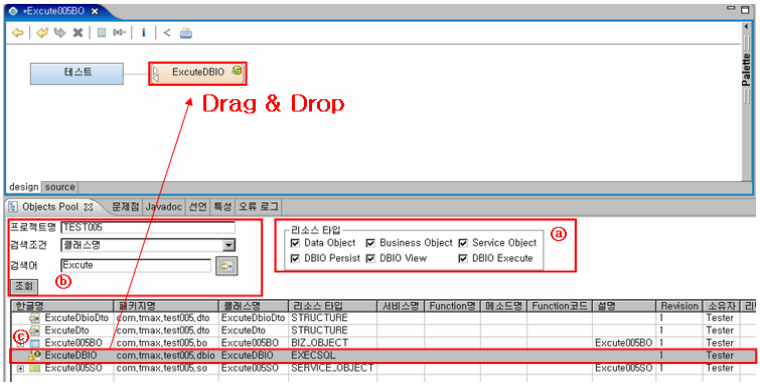


그림 *66 Object* 검색

ⓐ 리소스 타입 선택

ⓑ 프로젝트명, 검색조건, 검색에를 등록

ⓒ 검색결과 표시

6.2 EMB Designer Edit

EMB Designer 내에Palette에는 Edit를 위한 다양한 도구를 제공한다.

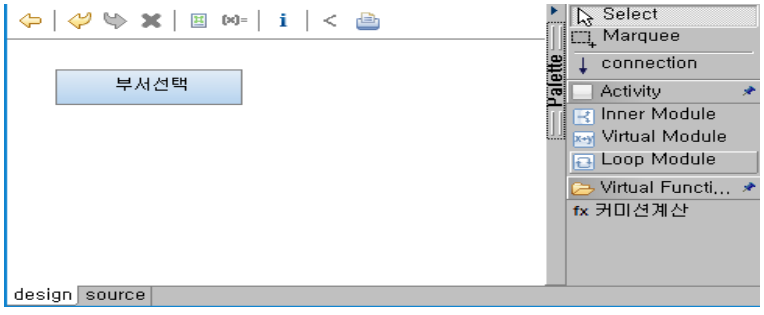


그림 *67 EMB Design Palette*

각각의 모듈들은 특성창을 이용하여 편집 수정을 제공한다.

Module	설명	특성	단축키
Inner Module	여러모듈들이 동시에 사용하는 변수를 지정하거나 모듈들의 집입조건, 예외처리등을 정하기 위해 사용	Gerneral, Condition, Exception	EMB Design 상에서 I
Virtual Module	입출력이 업이 단순한 코딩을 Flow 상에 삽입하고 싶을 때 사용됨	Gerneal, Condition	EMB Design 상에서 V
Loop Module	다수의 모듈이 반복적으로 사용될 때 사용됨	Gerneal, Condition, Pre/Post	EMB Design 상에서 L

특성	설명
Gerneral	모듈의 일반적인 이름을 설정하고 Inner 모듈에서만 Type 설정이 가능
Condition	모듈의 상태 정의 코딩 구간 제공
Exception	예외처리 코딩 구간 제공
Pre/Post	반복문 처리구간 제공

6.2.1 Inner-Module

EMB Designer에서 Inner Module 호출 시 General type를 function으로 지정하면 아래와 같은 소스 코드가 생성 된다. Condition 설정시 Line 91에서 boolean type으로 정의가 가능하며 child node는 Line 94 ~ 95 에 삽입된다.

```
89      //[BEGIN_NODE_BLOCK, 2, dept_con]
90      //[BEGIN_CONDITION_EXPRESSION, 2, dept_con] - You can insert your own code below this line
91      if ( true )
92      //[END_CONDITION_EXPRESSION, 2, dept_con] - You can insert your own code above this line
93      {
94          if( Logger.isDebugEnabled()) Logger.debug( "Node started. method: dept_con,  module: Inner Module(node id: 2)");
95          if( Logger.isDebugEnabled()) Logger.debug( "Node ended. method: dept_con,  module: Inner Module(node id: 2)");
96      }
97
98      if( Logger.isDebugEnabled()) Logger.debug("svcObjectType opeartion  dept_con is ended.");
99
100 }
```

그림 *68 Inner-Module block* 예제

EMB Designer에서 Inner Module 호출 시 General type를 function으로 지정하면 아래와 같은 소스 코드가 생성 된다. Condition 설정시 Line 91에서 boolean type으로 정의가 가능하며Line 96에서 child node를 호출한다.

```
89      //[BEGIN_NODE_BLOCK, 1, dept_con]
90      //[BEGIN_CONDITION_EXPRESSION, 1, dept_con] - You can insert your own code below this line
91      if ( true )
92      //[END_CONDITION_EXPRESSION, 1, dept_con] - You can insert your own code above this line
93      {
94         if( Logger.isDebugEnabled()) Logger.debug( "Node started. method: dept_con,  module: Inner Module(node id: 1)");
95         try {
96            dept_conInnerModule1(cb, inDto, outDto, contextMap);
97         } catch(Exception e) {
98            Logger.error( "dept_con Inner Module(node id: 1)" + ", " + e.getMessage(), e);
99            if(e instanceof ProFrameException)
100              throw (ProFrameException) e;
101           else
102              throw new ProFrameException(1003, e.getMessage(), e);
103        }
104        if( Logger.isDebugEnabled()) Logger.debug( "Node ended. method: dept_con,  module: Inner Module(node id: 1)");
105     }
106
107     if( Logger.isDebugEnabled()) Logger.debug("svcObjectType opeartion  dept_con is ended.");
108
109  }
```

그림 *69 Inner-Module block* 예제

6.2.2 Inner-Module 기능

EMB Designer내에서 Inner Module은 분기점의 역할뿐 아니라 변수 및 컨택스트를 정의하여 다양한 서비스를 작성할 수 있다.

Inner-Module 기능		설명
컨텍스트		선택한DTO를 hashMap으로 객체 생성하여 Mapping 사용
변수 정의	DTO 타입	DTO를 변수로 정의
	프리미티브 타입	String, number, integer, binary, short, int, long, float, double, boolean,byte,char
	사용자 정의 타입	Java api

그림 *70 Inner-Module* 컨텍스트 및 변수정의 부분

또한 Inner-Module은 자식노드를 가질 수 있다.

6.2.3 Virtual-Module

EMB Designer Virtaul-Module호출 시 아래와 같은 소스 코드가 생성 된다. Condition 설정시 Line 91에서 boolean type으로 정의가 가능하며Line 95에서 소스코딩이 허용된다.

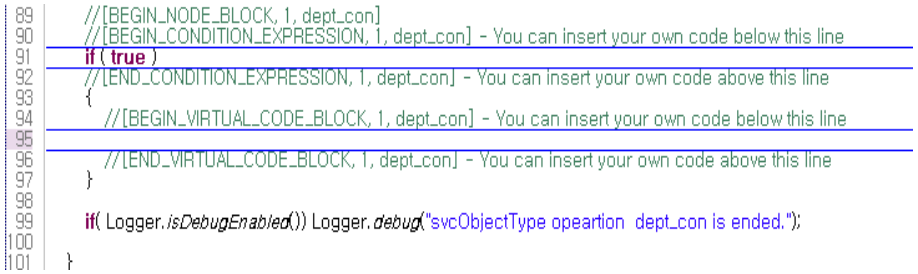


그림 *71 Virtual-Module* 예제

또한 Virtual-Module은 자식노드를 가질 수 없다.

6.2.4 Loop-Module

EMB Designer Loop-Module호출 시 아래와 같은 소스 코드가 생성 된다. Condition 설정시 Line 91에서 boolean type으로 정의가 가능하며Line 96, 99에서 소스코딩이 허용되며 Line 97,98 사이에 child Node가 들어가게 되며 for, while, do 와 같은 반복문을 사용하기 위한 모듈이다.

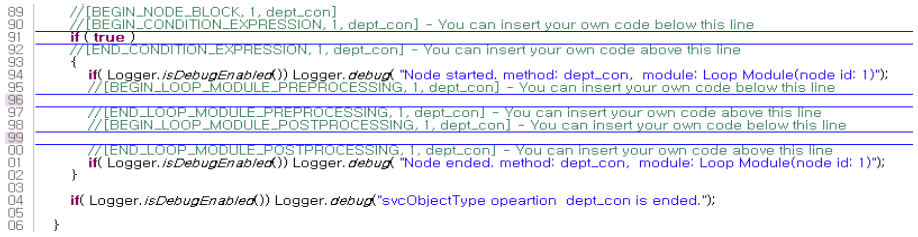


그림 *72 Loop-Module block* 예제

그러므로 Loop-Module은 자식노드를 가질 수 있다.

6.3 DBIO Call Method

ProFrameWAS 4.0은 DBIO Call Metheod가 EMB Designer내에 기본적으로 등록되어 있어 코딩없이도 간단하게 CRUD를 구현할 수 있다.

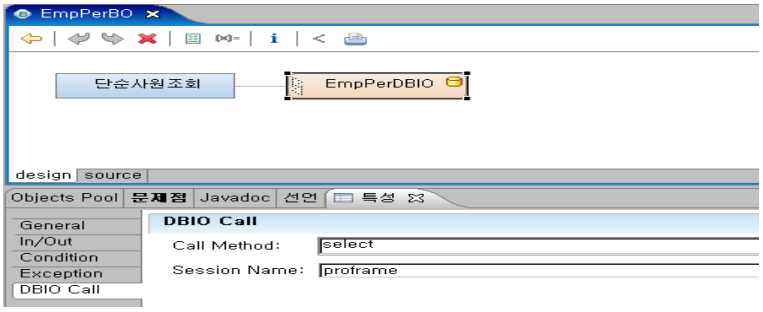


그림 *73 DBIO Call Method* 설정

/home/JEUS_HOME/lib/application 폴더 아래의 dbio_config.xml 파일에서 <datasource conn_name = 과 동일하게 Session Name을 작성한다.

DBIO 종류에 따라 아래와 같은 DBIO Call Method Type를 제공한다.

DBIO Type	Query Type	Call Method
Persist	select	select, selectPage, selectDynamic, selectDynamicPage, selectForupdate
	update	update, updateMulti, updateDynamic
	delete	delete,deleteMulti,deleteDynamic
	insert	insert, insertMulti
View	select	select, selectPage, selectDynamic, selectDynamicPage, selectForupdate, selectMultiDynamic, selectMultiDynamicPage,
Excute	insert	insert, insertMulti, insertDynamic, insertMultiDynamic

그림 *74 DBIO Call Method Type*

DBIO Call Method 기능.

Call Method	기능
select	Select Query
selectPage	Select Qurey 결과값의 페이지와 사이즈를 조절하는 Method로 Tester 시 dbio_fetch_seq_(결과값에 대한 페이지 수 지정)와 dbio_fetch_size_(한 화면에 결과값 호출 수 지정)를 입력
selectDynamic	Select Dynamic Sql 사용하며 Where Condition에 Dynamic Query를 입력
selectDynamicPage	Dynamic Sql + selectPage
selectForupdate	selectForupdate

selectMultiDynamic	Select다중 Query를 지원하며 Tester 시 Dbio_total_count_ 에서 설정한다 또한 Dynamic Sql을 지원
selectMultiDynamicPage	selectMultiDynamic + selectPage
update	Update Query
updateMulti	Update 다중 Query를 지원하며 Tester 시 Dbio_total_count_ 에서 설정
updateDynamic	Update Dynamic Sql 사용하며 Where Condition에 Dynamic Query를 입력
delete	Delete Query 지원 메소드
deleteMulti	Delete 대한 다중 Query를 지원하며 Tester 시 Dbio_total_count_ 에서 설정
deleteDynamic	Delete Dynamic Sql 사용하며 Where Condition에 Dynamic Query를 입력
insert	Insert Query
insertMulti	Insert다중 Query를 지원하며 Tester 시 Dbio_total_count_ 에서 설정
insertDynamic	Insert Dynamic Sql 사용하며 Where Condition에 Dynamic Query를 입력
insertMultiDynamic	Insert다중 Query를 지원하며 Tester 시 Dbio_total_count_ 에서 설정한다 또한 Dynamic Sql을 지원

그림 *75 DBIO Call Method* 기능

6.3.1 DBIO Call Method Example

DBIO Call Method 예제를 위해 Oracle에서 제공하는 DEPT/EMP Table을 사용했음을 알리는 바이다.

6.3.2 select

select 설정

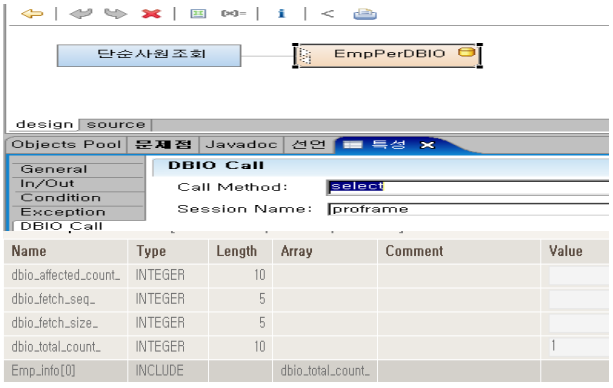


그림 *76 select* 설정 및 테스트 시 입력 필드

6.3.3 selectPage

selectPage설정

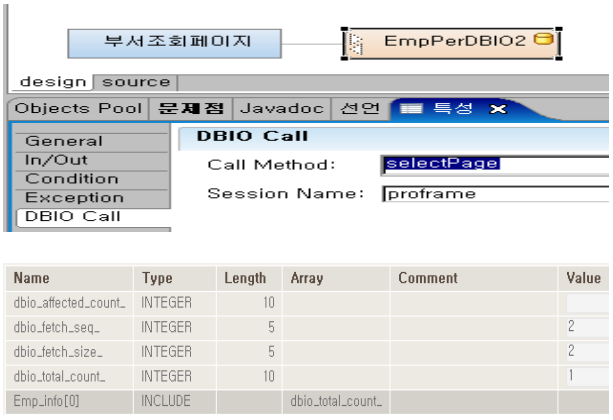


그림 *77 selectPage* 설정 및 테스트 시 입력 필드

selectPage Test 결과

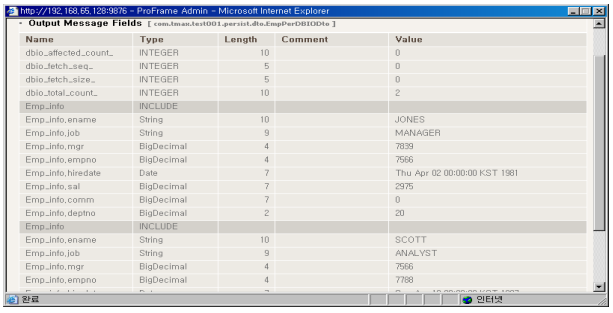


그림 *78 selectPage* 테스트 결과

6.3.4 selectDynamic

selectDynamic 설정 (Sql Query 에 /*DYNAMIC_KEYWORD*/ 삽입)

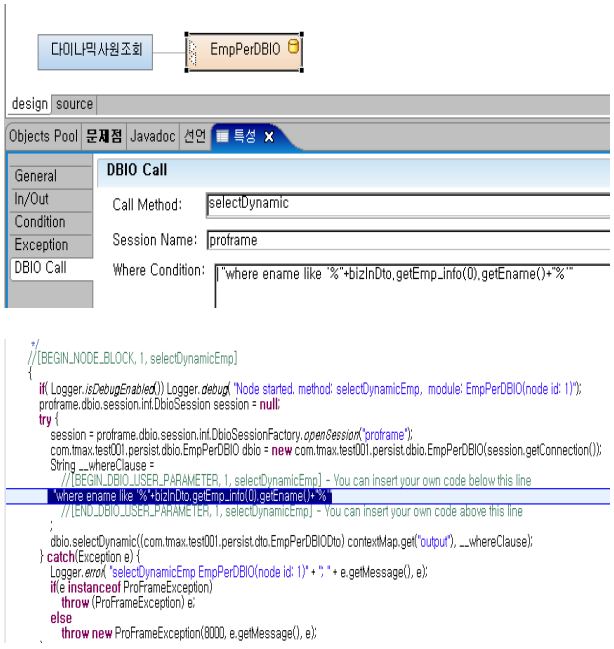


그림 *79 selectDynamic* 설정

6.3.5 selectDynamicPage

selectDynamicPage 설정 (Sql Query 에 /*DYNAMIC_KEYWORD*/ 삽입)

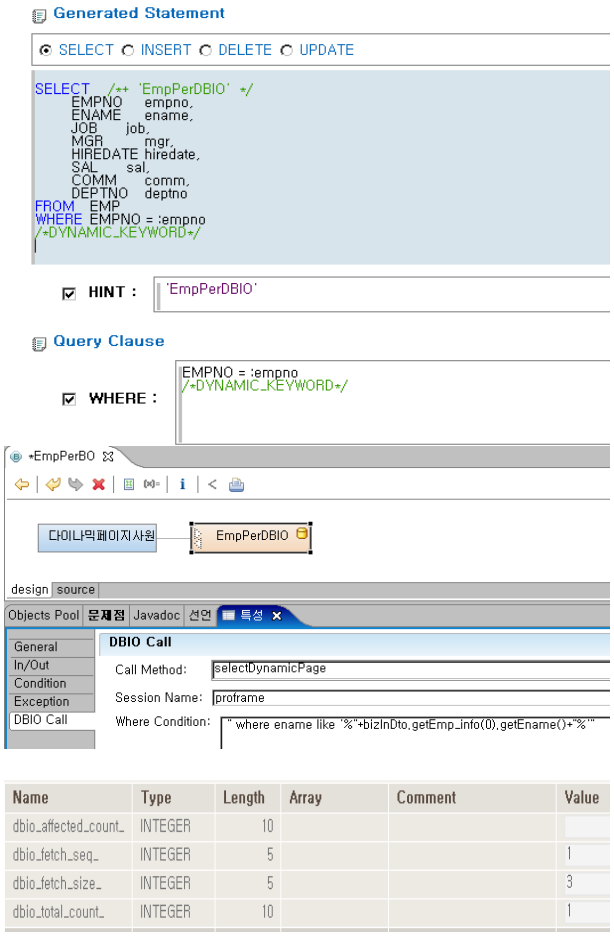


그림 *80 selectDynamicPage* 설정 및 테스트시 입력필드

6.3.6 selectForUpdate

selectForUpdate 설정 (준비 중)

그림 *81 selectForUpdate* 설정 및 테스트시 입력필드

6.3.7 selectMultiDynamic

selectMultiDynamic 설정 (테스트 시 + 버튼을 눌러 입력필드를 추가, Sql Query 에 /*DYNAMIC_KEYWORD*/ 삽입)

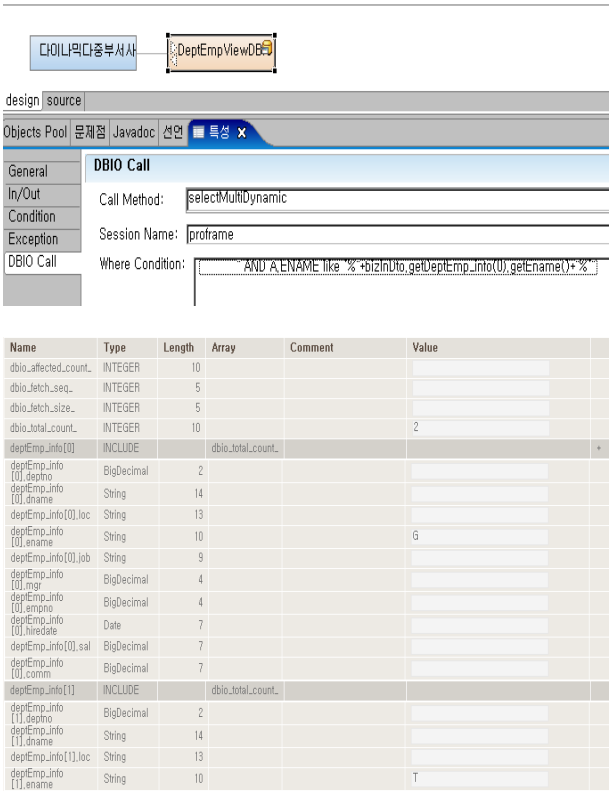
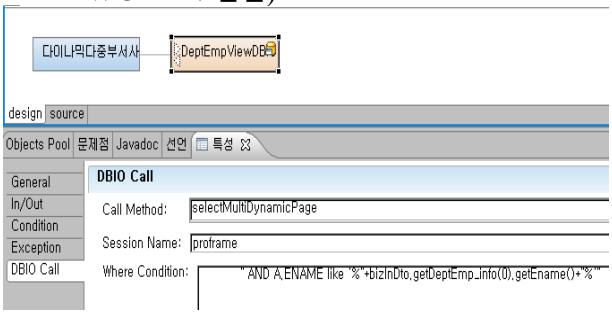


그림 *82 selectMultiDynamic* 설정 및 테스트시 입력필드

6.3.8 selectMultiDynamicPage

selectMultiDynamicPage 설정(테스트 시 + 버튼을 눌러 입력필드를 추가, Sql Query 에 /*DYNAMIC_KEYWORD*/ 삽입)

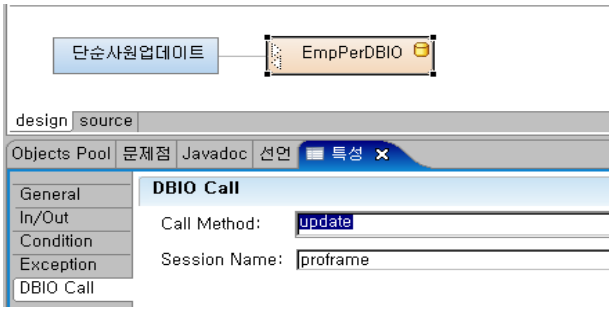


Name	Type	Length	Array	Comment	Value
dbio_affected_count_	INTEGER	10			
dbio_fetch_seq_	INTEGER	5			1
dbio_fetch_size_	INTEGER	5			1
dbio_total_count_	INTEGER	10			2
deptEmp_info[0]	INCLUDE		dbio_total_count_		
deptEmp_info [0].deptno	BigDecimal	2			
deptEmp_info [0].dname	String	14			
deptEmp_info[0].loc	String	13			
deptEmp_info [0].ename	String	10			G
deptEmp_info[0].job	String	9			
deptEmp_info [0].mgr	BigDecimal	4			
deptEmp_info [0].empno	BigDecimal	4			
deptEmp_info [0].hiredate	Date	7			
deptEmp_info[0].sal	BigDecimal	7			
deptEmp_info [0].comm	BigDecimal	7			
deptEmp_info[1]	INCLUDE		dbio_total_count_		
deptEmp_info [1].deptno	BigDecimal	2			
deptEmp_info [1].dname	String	14			
deptEmp_info[1].loc	String	13			
deptEmp_info [1].ename	String	10			T

그림 *83 selectMultiDynamicPage* 설정 및 테스트시 입력필드

6.3.9 update

update 설정



Name	Type	Length	Array	Comment	Value
dbio_affected_count_	INTEGER	10			
dbio_fetch_seq_	INTEGER	5			
dbio_fetch_size_	INTEGER	5			
dbio_total_count_	INTEGER	10			1
Emp_info[0]	INCLUDE		dbio_total_count_		
Emp_info[0].ename	String	10			PARK
Emp_info[0].job	String	9			TESTER
Emp_info[0].mgr	BigDecimal	4			7788
Emp_info[0].empno	BigDecimal	4			9999
Emp_info [0].hiredate	Date	7			
Emp_info[0].sal	BigDecimal	7			3000
Emp_info[0].comm	BigDecimal	7			3000
Emp_info[0].deptno	BigDecimal	2			30

그림 *84 update* 설정 및 테스트시 입력필드

Name	Type	Length	Comment	Value
dbio_affected_count_	INTEGER	10		1
dbio_fetch_seq_	INTEGER	5		0
dbio_fetch_size_	INTEGER	5		0
dbio_total_count_	INTEGER	10		0

그림 *85 update* 실행 결과 창

6.3.10 updateMulti

updateMulti 설정(테스트 시 + 버튼을 눌려 입력필드를 추가)

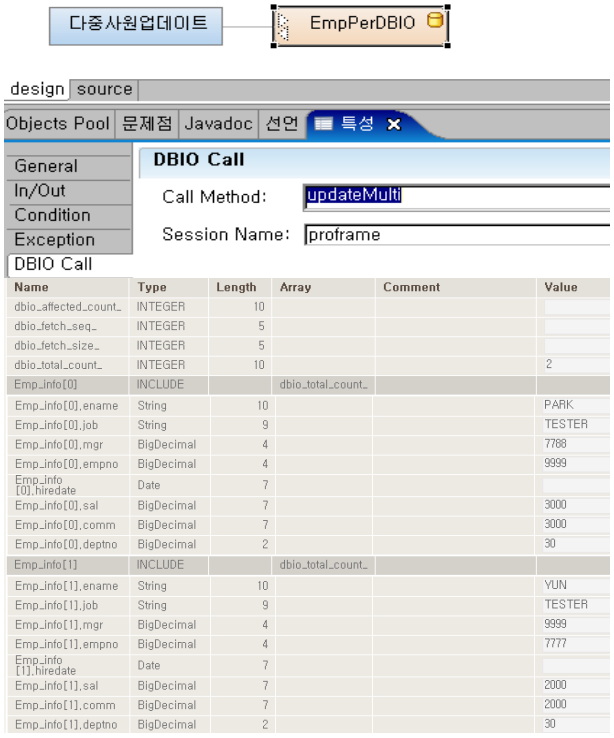


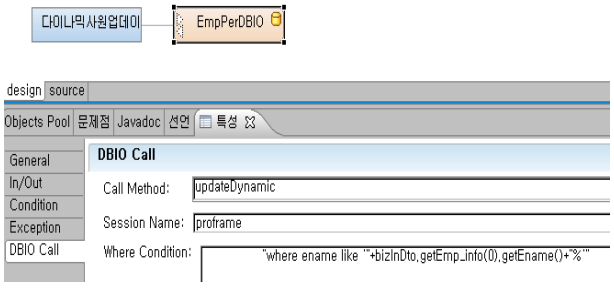
그림 *86 updateMulti* 설정 및 테스트시 입력필드

Name	Type	Length	Comment	Value
dbio_affected_count_	INTEGER	10		2
dbio_fetch_seq_	INTEGER	5		0
dbio_fetch_size_	INTEGER	5		0
dbio_total_count_	INTEGER	10		0

그림 *87 updateMulti* 실행 결과 창

6.3.11 updateDynamic

updateDynamic 설정(Sql Query 에 /*DYNAMIC_KEYWORD*/ 삽입)



Name	Type	Length	Array	Comment	Value
dbio_affected_count_	INTEGER	10			
dbio_fetch_seq_	INTEGER	5			
dbio_fetch_size_	INTEGER	5			
dbio_total_count_	INTEGER	10			1
Emp_info[0]	INCLUDE		dbio_total_count_		
Emp_info[0].ename	String	10			YU
Emp_info[0].job	String	9			sale
Emp_info[0].mgr	BigDecimal	4			7788
Emp_info[0].empno	BigDecimal	4			4444
Emp_info[0].hiredate	Date	7			
Emp_info[0].sal	BigDecimal	7			2000
Emp_info[0].comm	BigDecimal	7			2000
Emp_info[0].deptno	BigDecimal	2			10

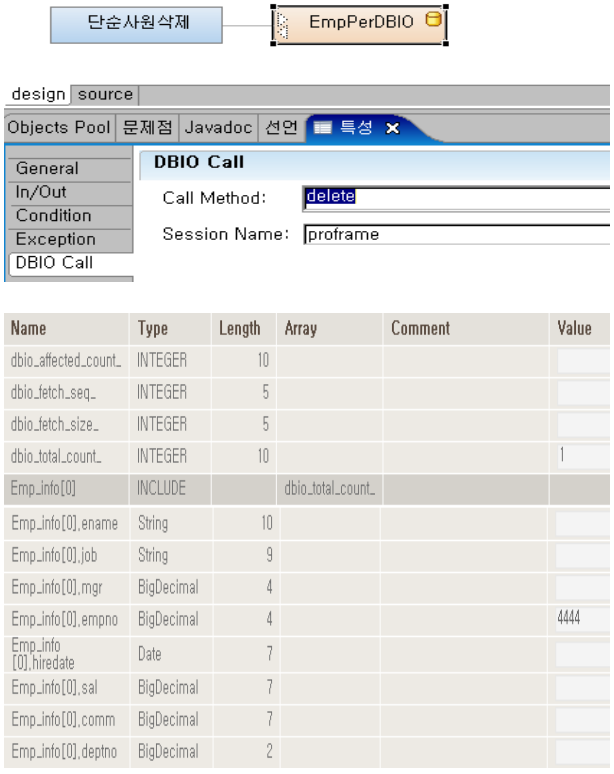
그림 *88 updateDynamic*설정 및 테스트시 입력필드

Name	Type	Length	Comment	Value
dbio_affected_count_	INTEGER	10		1
dbio_fetch_seq_	INTEGER	5		0
dbio_fetch_size_	INTEGER	5		0
dbio_total_count_	INTEGER	10		0

그림 *89 updateDynamic* 실행 결과 창

6.3.12 delete

delete 설정



Name	Type	Length	Array	Comment	Value
dbio_affected_count_	INTEGER	10			
dbio_fetch_seq_	INTEGER	5			
dbio_fetch_size_	INTEGER	5			
dbio_total_count_	INTEGER	10			1
Emp_info[0]	INCLUDE		dbio_total_count_		
Emp_info[0].ename	String	10			
Emp_info[0].job	String	9			
Emp_info[0].mgr	BigDecimal	4			
Emp_info[0].empno	BigDecimal	4			4444
Emp_info[0].hiredate	Date	7			
Emp_info[0].sal	BigDecimal	7			
Emp_info[0].comm	BigDecimal	7			
Emp_info[0].deptno	BigDecimal	2			

그림 *90 delete* 실행 설정 및 테스트 시 입력필드

Name	Type	Length	Comment	Value
dbio_affected_count_	INTEGER	10		1
dbio_fetch_seq_	INTEGER	5		0
dbio_fetch_size_	INTEGER	5		0
dbio_total_count_	INTEGER	10		0

그림 *91 delete* 실행 결과 창

6.3.13 deleteMulti

deleteMulti 설정

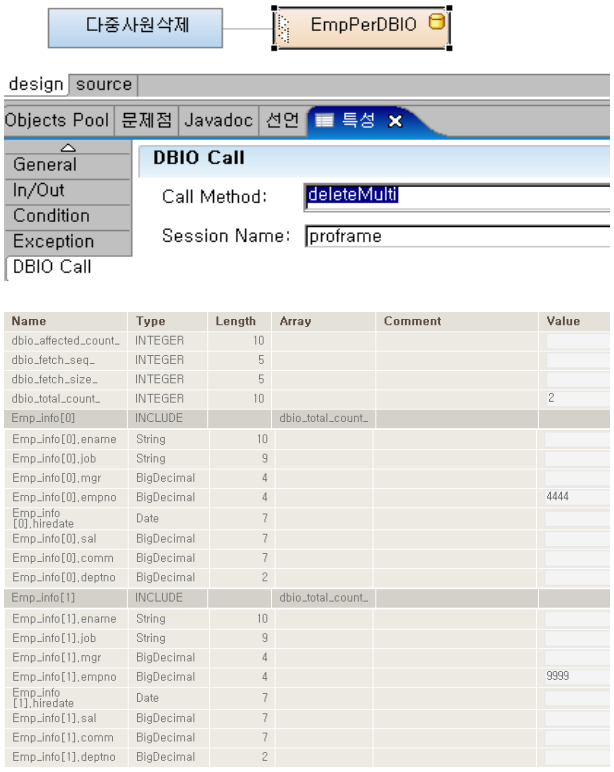


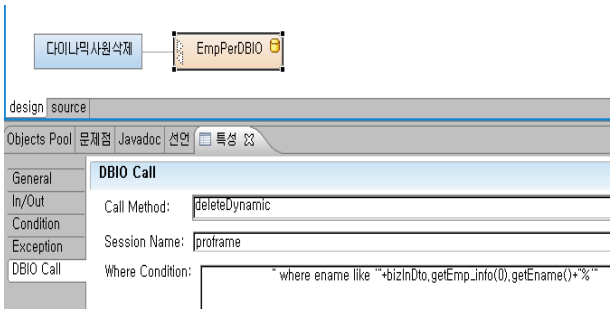
그림 *92 deleteMulti* 설정 및 테스트 시 입력필드

Name	Type	Length	Comment	Value
dbio_affected_count_	INTEGER	10		2
dbio_fetch_seq_	INTEGER	5		0
dbio_fetch_size_	INTEGER	5		0
dbio_total_count_	INTEGER	10		0

그림 *93 deleteMulti* 실행 결과 창

6.3.14 deleteDynamic

deleteDynamic 설정(Sql Query 에 /*DYNAMIC_KEYWORD*/ 삽입)



Name	Type	Length	Array	Comment	Value
dbio_affected_count_	INTEGER	10			
dbio_fetch_seq_	INTEGER	5			
dbio_fetch_size_	INTEGER	5			
dbio_total_count_	INTEGER	10			1
Emp_info[0]	INCLUDE		dbio_total_count_		
Emp_info[0].ename	String	10			P
Emp_info[0].job	String	9			
Emp_info[0].mgr	BigDecimal	4			
Emp_info[0].empno	BigDecimal	4			
Emp_info [0].hiredate	Date	7			
Emp_info[0].sal	BigDecimal	7			
Emp_info[0].comm	BigDecimal	7			
Emp_info[0].deptno	BigDecimal	2			

그림 *94 deleteDynamic* 설정 및 테스트 시 입력필드

Name	Type	Length	Comment	Value
dbio_affected_count_	INTEGER	10		1
dbio_fetch_seq_	INTEGER	5		0
dbio_fetch_size_	INTEGER	5		0
dbio_total_count_	INTEGER	10		0

그림 *95 deleteDynamic* 실행 결과 창

6.3.15 insert

insert 설정

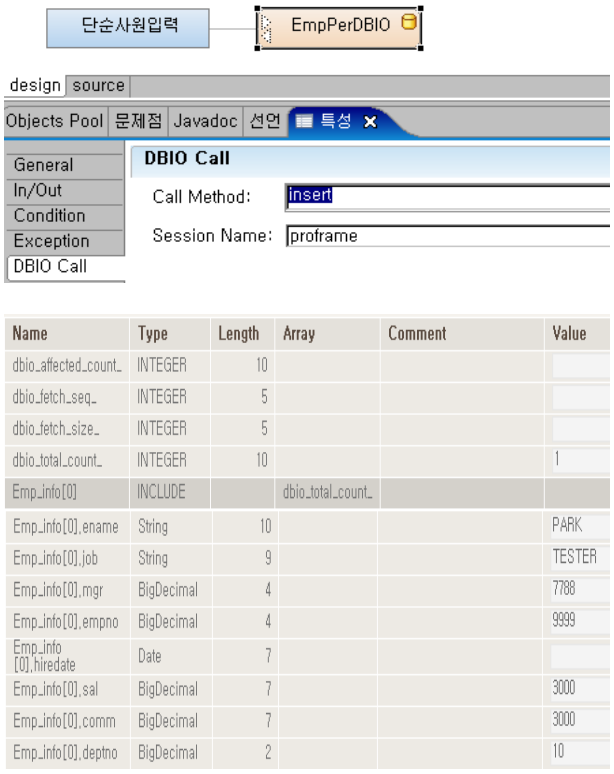


그림 *96 insert* 설정 및 테스트 시 입력필드

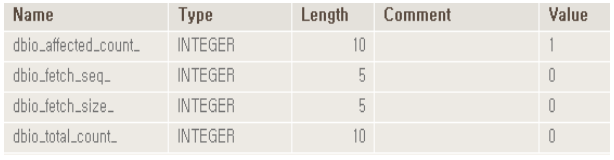
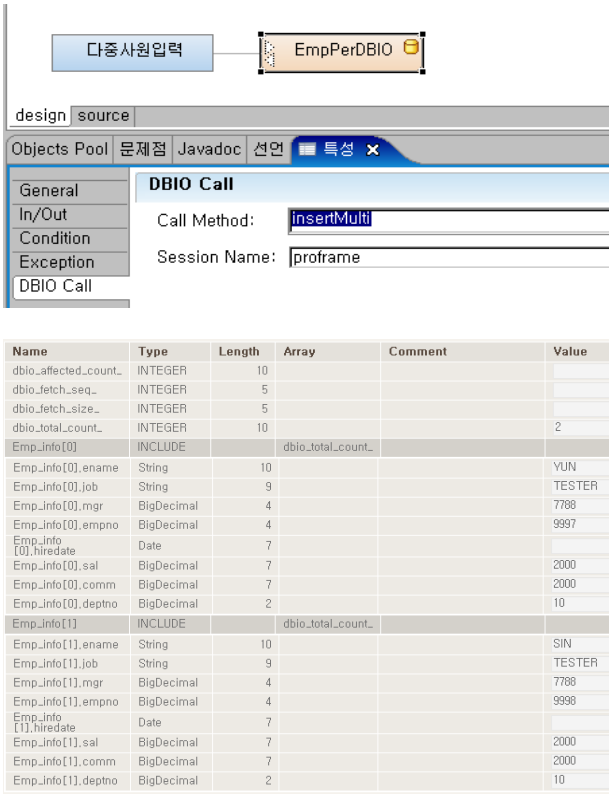


그림 *97 insert* 실행 결과 창

6.3.16 insertMulti

insertMulti 설정(테스트 시 + 버튼을 눌려 입력필드를 추가)



Name	Type	Length	Array	Comment	Value
dbio_affected_count_	INTEGER	10			
dbio_fetch_seq_	INTEGER	5			
dbio_fetch_size_	INTEGER	5			
dbio_total_count_	INTEGER	10			2
Emp_info[0]	INCLUDE		dbio_total_count_		
Emp_info[0].ename	String	10			YUN
Emp_info[0].job	String	9			TESTER
Emp_info[0].mgr	BigDecimal	4			7788
Emp_info[0].empno	BigDecimal	4			9997
Emp_info[0].hiredate	Date	7			
Emp_info[0].sal	BigDecimal	7			2000
Emp_info[0].comm	BigDecimal	7			2000
Emp_info[0].deptno	BigDecimal	2			10
Emp_info[1]	INCLUDE		dbio_total_count_		
Emp_info[1].ename	String	10			SIN
Emp_info[1].job	String	9			TESTER
Emp_info[1].mgr	BigDecimal	4			7788
Emp_info[1].empno	BigDecimal	4			9998
Emp_info[1].hiredate	Date	7			
Emp_info[1].sal	BigDecimal	7			2000
Emp_info[1].comm	BigDecimal	7			2000
Emp_info[1].deptno	BigDecimal	2			10

그림 *98 insertMulti* 설정 및 테스트 시 입력필드

Name	Type	Length	Comment	Value
dbio_affected_count_	INTEGER	10		2
dbio_fetch_seq_	INTEGER	5		0
dbio_fetch_size_	INTEGER	5		0
dbio_total_count_	INTEGER	10		0

그림 *99 insertMulti* 실행 결과 창

6.3.17 insertDynamic

updateDynamic 설정 참조

6.3.18 insertMultiDynamic

selectMultiDynamic 설정 참조

7 BO

Business Object의 약자로 비즈니스 기능을 수행하는 재사용성의 단위로서 Service Object에서 레퍼런스 호출 되며 정보 처리 및 계산 그리고 DBIO 호출 후 데이터 가공 등을 수행한다 BO의 작성방법은 오른쪽 마우스 새로작성(W) > Business Object를 선택한다.

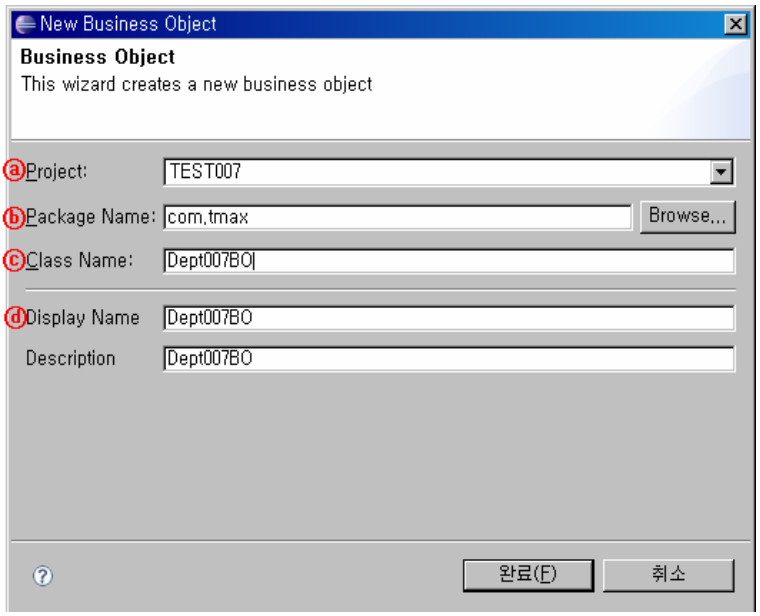


그림 *100 Business Object* 입력화면

ⓐ Project: 서비스가 작성 될 프로젝트를 선택

ⓑ Package Name: 패키지명을 등록

ⓒ Class Name: 메타에 등록되는 이름으로 동일 패키지 내에서 유일

ⓓ Display Name: 한글 영문 표기가 가능 패키지 내에서 유일

[BO, SO의 경우 ⓑ, ⓒ, ⓓ 일괄입력]

8 SO

Service Object의 약자로 외부에 오퍼레이션을 노출할 수 있는 리소스 단위로서 단위 트랜잭션 수행의 주로 Business Object를 Orchestration하는 Flow 중심 어플리케이션이다. SO의 작성방법은 오른쪽 마우스 새로작성(W) > Service Object를 선택한다.

ⓐ Project: 서비스가 작성 될 프로젝트를 선택

ⓑ Package Name: 패키지명을 등록

ⓒ Class Name: 메타에 등록되는 이름으로 동일 패키지 내에서 유일

ⓓ Display Name: 한글 영문 표기가 가능 패키지 내에서 유일

[BO, SO의 경우 ⓑ, ⓒ, ⓓ 일괄입력]

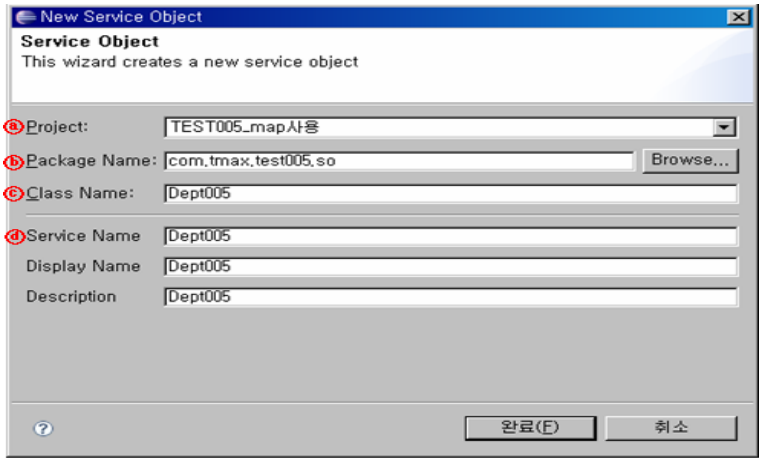


그림 *101 Business Object* 입력화면

서비스 작성 완료 후에는 반드시 Generate Application DD 클릭하여 Proframe.xml파일 생성한다.

그림 *102 ProFrame.xml* 파일 생성

8.1 ProFrame.xml

패키지 탐색기에서 ProFrame.xml 파일을 클릭하여 필요시 아래와 같이 설정을 변경한다.

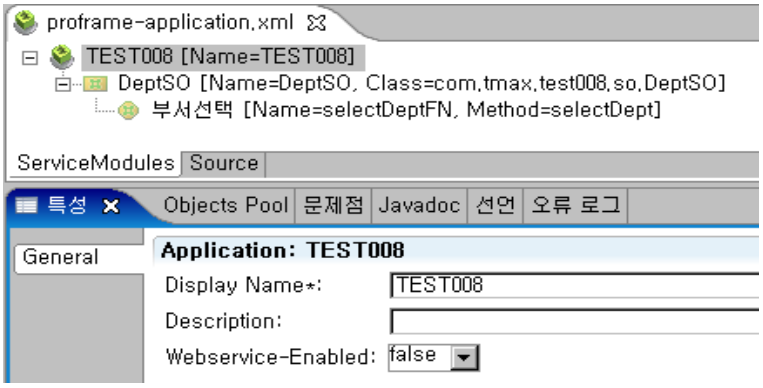


그림 *103 ProFrame.xml webService* 지원

Required: Transation을 하나로 묶어서 사용(default)

RequiresNew: Transaction을 개별적으로 사용

NotSupported

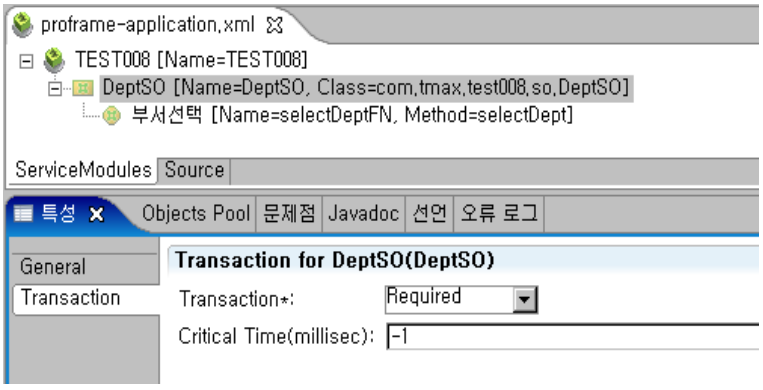


그림 *104 ProFrame.xml transctoion type*정의

등록된 서비스 정보

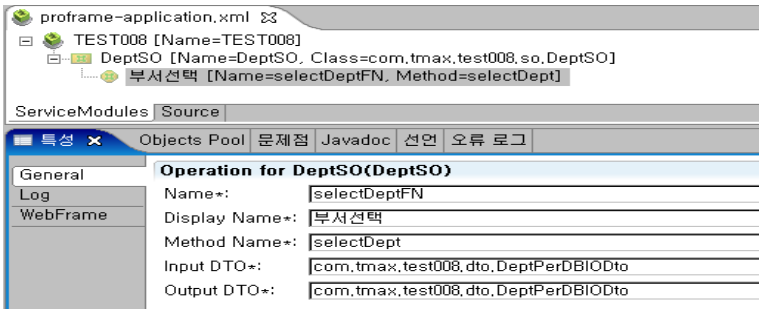


그림 *105* 등록된 서비스 정보확인

등록된 서비스 정보의 Log level을 설정한다.

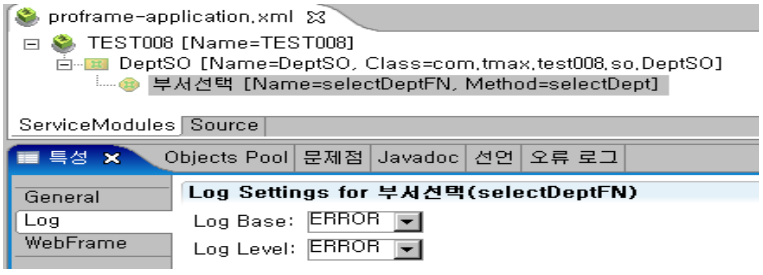


그림 *106 Log Level* 설정

Request Page: 거래요청 페이지 경로

Success Path: 거래성공시 경로

Failure Path: 거래실패시 경로

Inteceptor:

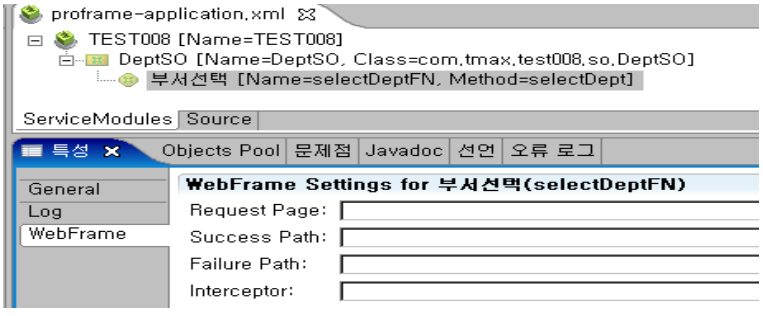


그림 *107 Web Application* 설정

9 기타

9.1 CVS 기능

9.1.1 Check-In, Out

ProBuilder 내에서 작성 된 파일에 대하여 Check In – Check out 기능을 제공한다. Check out을 하게 되면 해당 파일 앞에 검은색체크 박스와 동시에 수정 후 저장할 수 있다.

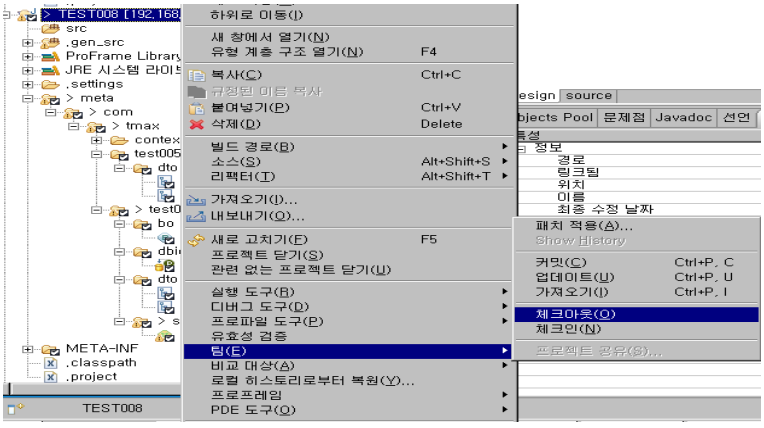


그림 *108 check-out* 화면

9.1.2 Update

마지막으로 commit 된 버전으로 변경된다. 오른쪽 마우스 팀(E) > 업데이트(U)를 선택한다.

9.1.3 Resource 가져오기

서버에 저장 된 서비스 모듈을 패키지명 또는 프로프레임리소스 명에 따라 다운로드 받을 수 있다. 오른쪽 마우스 팀(E) > 가져오기를 선택한다.

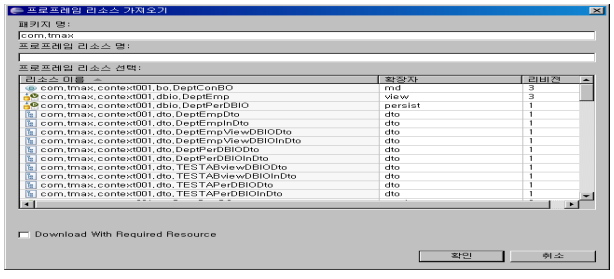


그림 *109 Resource* 가져오기

9.1.4 history

Commit 된 프로프레임 리소스를 버전별로 비교 변경이 된다. 오른쪽 마우스 팀(E) > Show History를 선택하며 수정을 하기 위해 우측 상단의 버튼을 사용한다.

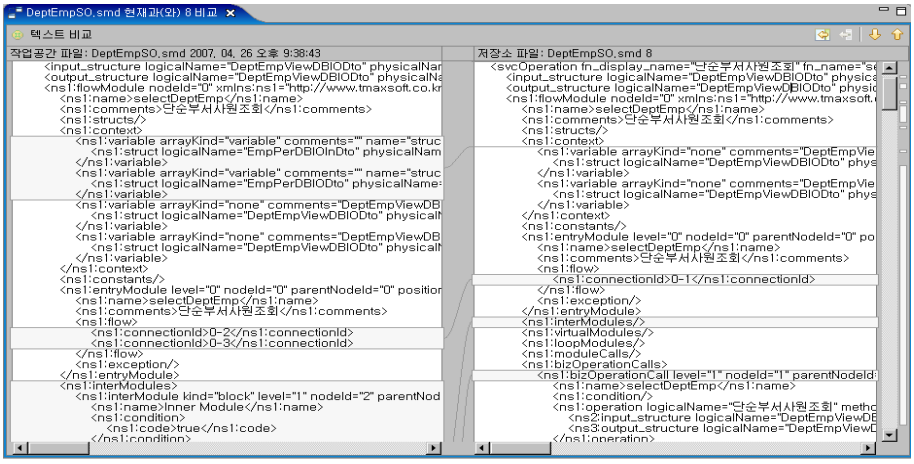


그림 *110 Resource history* 비교

9.1.5 Local history

날짜별로 프로프레임 리소스를 변경 된 사항을 확인할 수 있다. 오른쪽마우스 비교대상(A) > 로컬 히스토리(L)를 선택한다.

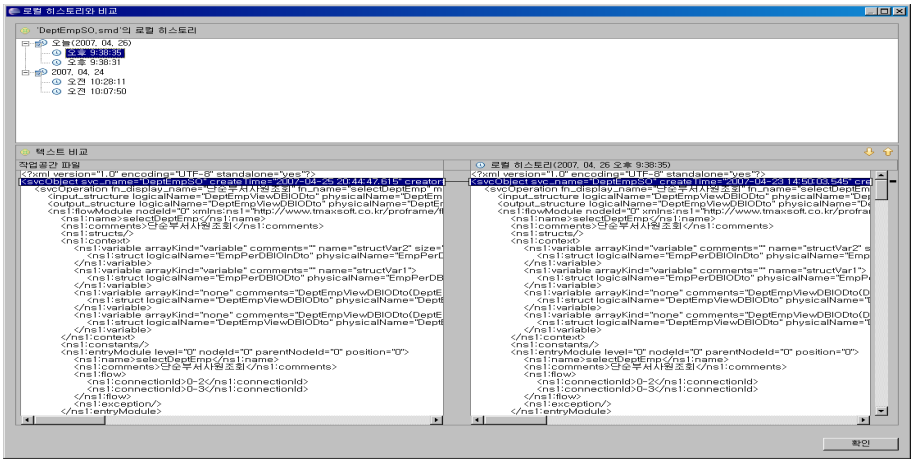


그림 *111* 로컬 히스토리