

X-UP 사용자 매뉴얼

1.3



1.3

X-UP 사용자 매뉴얼

TOBESOFT

차례

차례	iv
저작권 및 면책조항	xiii
일러두기	xiv
1. X-UP 소개	1
1.1 X-UP 이란?	1
X-UP의 정의	1
X-UP의 기능 및 특징	1
X-UP의 구성	2
2. X-UP 개요	3
2.1 X-UP 모델	3
Domain	4
2.2 X-UP Server	4
Bundle 처리 명세	4
Bundle 설치	5
Bundle 제거	5
Bundle Repository	5
X-UP Server 파일 구조	6
설치 및 업데이트	7
라이선스 등록	8
2.3 X-UP Builder	8
X-UP Builder 파일 구조	9
2.4 X-UP Administrator	11
3. X-UP Builder 기초	12
3.1 X-UP Builder 설치 및 삭제	12
X-UP Builder 설치 파일 종류	12
시스템 요구사항	12
설치	13
설치 정보 및 버전 확인	17

삭제	17
라이선스 적용	18
3.2 X-UP Builder 버전 패치 가이드	19
X-UP Builder 버전 패치(Update) 방법	19
About X-UP Bulder	26
4. X-UP Builder 구성	28
4.1 Menu and 개발도구 Bar	29
4.2 X-UP Explorer	31
X-UP Project	32
X-UP Filter	33
4.3 X-UP Wizard	35
New X-UP Project Wizard	36
New X-UP DataSource Wizard	37
New X-UP Model Wizard	39
New X-UP Automation Model Wizard	41
4.4 X-UP Editor	43
Domain Editor	43
Overview	43
Properties	44
DataSource	46
Transaction Model Editor	47
Input	47
Output	48
Generate Transaction Logic	49
Automation Model Editor	50
Palette	51
Drawing Pane	53
프로세스 커넥션	54
데이터 커넥션	55
이벤트 커넥션	55
조건 커넥션	56
Menu, 개발도구 Bar	57
Outline	58
Valiate and Error Mark	59
4.5 X-UP View	60
Result DataSet View	61
Outline View	61
Properties View	63
X-UP DataSource View	64

5. X-UP 설정	67
5.1 X-UP Server 설정	67
xup_config.xml 설정	68
log4j.xml 설정	68
auth_info.xml 설정	69
web.xml 설정	70
JCO libarary 설정	70
5.2 X-UP Builder Preference 설정	71
X-UP Page	71
DataSource page	72
X-UP Builder의 Preference에 데이터소스 정의하여 사용하기	74
compile 버전 변경	78
X-UP Builder의 Preference에서 Compile 버전 1.5로 변경하기	79
6. X-UP 모델	81
6.1 모델 로직 및 정책	81
Eager Loading	81
Singleton	82
6.2 Automation Model	82
SAP RFC Invoke	83
SAP RFC Invoke Dialog	84
InvokerInfo	85
Select Invoke	85
SQL입력	86
DBMS별 Data Type Mapping	87
CDATA 지원	88
First Row 기능	88
Dynamic SQL 지원	88
InvokerInfo	89
Event Handler	90
Modify Invoke	91
SQL 입력	92
InvokerInfo	93
SqlTypeBinding	93
Filtering	93
BeforeDataSource	94
AfterDataSource	95
Event Handler	95
Exception	97
CDATA 지원	97

Dynamic SQL 지원	97
Procedure Invoke	98
Procedure Invoke Dialog	99
First Row 기능	100
InvokeInfo	101
Input Binding / Output Binding	101
WebService Invoke	103
WebService Invoke Dialog	104
InvokeInfo	105
OpenApi Invoke	105
OpenApi Invoke Dialog	106
Sub Path와 Input Parameter	107
Result	108
InvokeInfo	108
X-UP Invoke	109
InvokeInfo	110
Model Invoke	110
InvokeInfo	111
Method Function	112
DataSetRowLoop Function	113
Merge Function	114
대상 데이터셋 설정	115
융합 로직 및 sql 쿼리 구현	116
새로운 데이터셋 생성	116
XML Parser Function	116
XML Parser Function Dialog	117
ParserInfo	118
ExtDataSet Function	118
ExtVariable Function	120
Decision	121
General	122
Connection Info	123
Condition Type	124
Condition Code	124
Repeat Task	126
Debug and Break Point	127
Parameter	128
File	128
Header	128

DataSet	129
Variable	131
6.3 Transaction Model	132
DB Binding	133
INSERT, UPDATE and DELETE	134
Transaction JDBC	134
DB Procedure Calling Code	136
X-UP Model Invoking Code	137
DataSet Creating Code	139
Error Code, Message 처리	140
Parameter를 이용한 ErrorCode, ErrorMsg 처리	140
예외발생 시 ErrorCode 변경	140
6.4 데이터 입-출력	140
6.5 데이터 포맷 및 파싱	141
XML 데이터 파싱	141
CSV 데이터 파싱	142
고정 길이 데이터 파싱	142
6.6 InvokingInfo	143
7. Data Source 정의	144
7.1 SAP RFC 데이터 소스	144
7.2 DB 데이터 소스	146
7.3 WebService 데이터 소스	147
7.4 OpenApi 데이터 소스	149
7.5 UDDI 데이터 소스	150
8. X-UP Automation 모델 개발하기	152
8.1 SAP RFC Invoke를 이용한 모델 개발하기	153
X-UP 프로젝트 생성하기	153
Automation 모델 생성 및 SAP RFC Invoke 생성하기	154
데이터소스 생성 및 Properties 설정하기	155
Invoke 테스트 하기	162
모델 테스트하기	164
8.2 Select Invoke를 이용한 모델 개발하기	164
X-UP 프로젝트 생성하기	165
Automation 모델 생성 및 Select Invoke 생성하기	166
입력 파라메터 설정 및 Properties 설정하기(SQL 변경)	170
Invoke 테스트 하기	174
Function을 이용한 추가 작업하기	175
모델 테스트하기	180
8.3 Modify Invoke를 이용한 모델 개발하기	181

X-UP 프로젝트 생성하기	181
Automation 모델 생성 및 Modify Invoke 생성하기	182
입력 파라미터 설정 및 Properties 설정하기	187
Invoke 테스트 하기	193
모델 테스트하기	195
8.4 Procedure Invoke를 이용한 모델 개발하기	197
X-UP 프로젝트 생성하기	197
Automation 모델 생성 및 Select Invoke 생성하기	198
Invoke 테스트 하기	204
모델 테스트하기	205
8.5 OpenApi Invoke를 이용한 모델 개발하기	206
X-UP 프로젝트 생성하기	206
Automation 모델 생성 및 OpenApi Invoke 생성하기	207
데이터소스 생성 및 Properties 설정하기	209
Invoke 테스트 하기	214
모델 테스트하기	215
8.6 WebService Invoke를 이용한 모델 개발하기	216
X-UP 프로젝트 생성하기	216
Automation 모델 생성 및 WebService Invoke 생성하기	217
데이터소스 생성 및 Properties 설정하기	218
Invoke 테스트 하기	224
모델 테스트하기	226
8.7 X-UP Invoke를 이용한 모델 개발하기	227
X-UP 프로젝트 생성하기	227
Automation 모델 생성 및 X-UP Invoke 생성하기	228
데이터소스 생성 및 Properties 설정하기	230
Invoke 테스트 하기	232
모델 테스트하기	235
8.8 DataSet Row Loop Function을 이용하여 데이터 가공하기	236
X-UP 프로젝트 생성하기	236
Automation 모델 생성하기	237
입력 파라미터를 위한 SAP RFC로직 구현하기	239
DataSetRowLoop 함수 생성하고 로직 설정하기	244
UserMethod Function을 이용하여 출력 파라미터 설정하기	245
모델 테스트하기	248
8.9 Merge Function을 이용하여 데이터 가공하기	250
X-UP 프로젝트 생성하기	250
Automation 모델 생성 및 Merge Function 생성하기	251
융합 할 데이터셋 생성하기	253

융합로직 설정하기	254
Function 테스트 하기	258
모델 테스트하기	259
8.10 XML Parser Function을 이용하여 데이터 가공하기	260
X-UP 프로젝트 생성하기	260
Automation 모델 생성 및 XML Parser Function 생성하기	261
입력 파라미터 설정 및 출력 인터페이스 생성하기	263
Function 테스트 하기	269
모델 테스트하기	272
8.11 UDDI를 이용하여 WebService Invoke 개발하기	273
X-UP 프로젝트 생성하기	273
Automation 모델 생성 및 UDDI 데이터소스 생성하기	274
UDDI를 이용하여 WebService 데이터소스 생성하기	277
WebService Invoke 개발하기	279
9. X-UP Transaction 모델 개발하기	280
9.1 DB Binding Code 마법사를 이용한 모델 개발하기	280
X-UP 프로젝트 생성하기	281
트랜잭션 모델 생성하기	281
입출력 파라미터 및 데이터셋 설정	282
위자드를 이용하여 모델 로직 클래스 수정하기	285
모델 테스트하기	290
9.2 JDBC Transaction Code 마법사를 이용한 모델 개발하기	293
X-UP 프로젝트 생성하기	293
트랜잭션 모델 생성하기	294
입출력 파라미터 및 데이터셋 설정	294
위자드를 이용하여 모델 로직 클래스 수정하기	298
모델 테스트하기	303
10. Deploy & Undeploy	306
10.1 모델을 X-UP서버에 Deploy하기	307
배치할 서버 설정하기	307
X-UP 모델 배치하기	308
웹브라우저로 배치된 모델 테스트 하기	312
10.2 X-UP 서버의 모델을Undeploy하기	314
Undeploy할 서버 설정하기	315
X-UP 모델 Undeploy하기	316
도메인 안에 있는 전체 모델에 대해 Undeploy 하기	320
11. XPLATFORM 어플리케이션에서 X-UP 모델 사용하기	325
11.1 TypeDefinition-Server 등록하기	325

11.2 모델 리스트와 인터페이스 가져오기	327
11.3 모델 호출 스크립트 작성하기	329
12. WebApplication(eGovFrame)에서 X-UP 사용하기	331
12.1 WebApplication 생성	331
12.2 WebApplication에 X-UP 적용	334
12.3 SAP RFC Invoke를 이용한 모델 개발 하기	339
12.4 WebApplication 서버에 개발 된 모델을 Export하기	339
12.5 WebApplication에서 모델 호출하기	341
12.6 테스트 하기	343
13. 기타 기능	345
13.1 Session Handling	345
X-UP 프로젝트 생성하기	345
Automation 모델 생성하기	346
UserMethod 생성하기	348
세션값 체크 Automation 모델 생성하기	348
user_id, user_pw 파라미터와 UserMethod 생성하기	349
세션해제 Automation 모델 생성하기	350
UserMethod 생성하기	351
테스트	352
13.2 Cookie Handling	354
X-UP 프로젝트 생성하기	355
Automation 모델 생성하기	356
Header 파라미터와 UserMethod 생성하기	357
UserMehod 로직 구현하고 output 파라미터 설정하기	358
테스트 화면만들기	359
13.3 File Handling	361
X-UP 프로젝트 생성하기	361
Automation 모델 생성하기	362
File 파라미터와 UserMethod 생성하기	364
UserMehod 로직 구현하고 테스트 파라미터 설정하기	365
Modify Invoke 및 Select Invoke 생성하고 출력 로직 설정하기	367
모델 테스트하기	369
13.4 User Library Handling	371
13.5 X-UP Server 관리	373
13.6 Reporting	373
13.7 Filtering	376
13.8 다국어 처리	377
13.9 Junit & Remote class 생성	377
JUnit Test Class 생성 방법	378

JUnit Test Class를 생성한 후 디버그 모드로 실행	379
Remote Test Souce 생성 방법	381
14. Exception 처리	382
15. Debugging	384
16. 오류 및 Error 확인 및 대응	386
17. 배포 안내	387
17.1 정책 및 일정 안내	387
배포 일정	387
배포 종류	388
17.2 배포 사이트 안내	388
부록 A. Dynamic Sql	389
부록 B. 예제 DB 테이블 생성 스크립트	394

이 문서에 잘못된 정보가 있을 수 있습니다. 투비소프트는 이 문서가 제공하는 정보의 정확성을 유지하기 위해 노력하고 특별한 언급 없이 이 문서를 지속적으로 변경하고 보완할 것입니다. 그러나 이 문서에 잘못된 정보가 포함되어 있지 않다는 것을 보증하지 않습니다. 이 문서에 기술된 정보로 인해 발생할 수 있는 직접적인 또는 간접적인 손해, 데이터, 프로그램, 기타 무형의 재산에 관한 손실, 사용 이익의 손실 등에 대해 비록 이와 같은 손해 가능성에 대해 사전에 알고 있었다고 해도 손해 배상 등 기타 책임을 지지 않습니다.

사용자는 본 문서를 구입하거나, 전자 문서로 내려 받거나, 사용을 시작함으로써, 여기에 명시된 내용을 이해하며, 이에 동의하는 것으로 간주합니다.

각 회사의 제품명을 포함한 각 상표는 각 개발사의 등록 상표이며 특허법과 저작권법 등에 의해 보호를 받고 있습니다. 따라서 본 문서에 포함된 기타 모든 제품들과 회사 이름은 각각 해당 소유주의 상표로서 참조용으로만 사용됩니다.

발행처 | (주)투비소프트
발행일 | 2014/07/03
주소 | (135-873) 서울시 강남구 봉은사로 617 인탑스빌딩 5층
전화 | 02-2140-7700
홈페이지 | www.tobesoft.co.kr

고객지원센터 | www.nexacro.co.kr
제품기술문의 | 1588-7895 (오전 10시부터 오후 5시까지)

일러두기

본 매뉴얼은 X-UP의 상세설명 및 X-UP Builder를 사용하여 Application을 개발하는 것에 대한 내용을 담고 있습니다.

책의 내용

본 매뉴얼은 다음과 같이 구성되어 있습니다.

- **1~2장. X-UP의 소개 및 개요**

X-UP 기술 및 연관기술에 대해 개괄적인 설명을 합니다.
X-UP의 구성 및 기본 개념에 대해 설명합니다.

- **3~4장. X-UP Builder의 소개 및 구성**

X-UP Builder 설치 및 메뉴 구성에 대해 설명합니다.

- **5장. X-UP Server와 Builder의 환경설정**

사용자가 개발 시 필요한 X-UP환경설정에 대해 설명합니다.

- **6장. X-UP 모델의 상세설명**

X-UP 모델의 개념에 대해 설명합니다.

Automation 모델의 기본 개념과 Invoke, Function 및 주요 기능에 대해 설명합니다.

Transaction 모델의 기본 개념과 주요 기능에 대하여 설명합니다.

X-UP 모델에서 제공되는 기타 서비스에 대하여 설명합니다.

- **7장. DataSource의 정의**

X-UP에서 사용되는 데이터소스에 대해 설명합니다.

- **8~9장. X-UP Builder를 이용하여 모델 개발하기**

X-UP Builder를 이용하여 예제를 통하여 실제로 모델을 개발합니다.

Automation 모델의 주요 기능을 이용하여 모델을 개발합니다.

Transaction 모델의 주요 기능을 이용하여 모델을 개발합니다.

- **10장. Deploy & Undeploy 설명**

Deploy & Undeploy 하는 방법에 대해 설명하고 X-UP Builder에서 개발한 모델을 Deploy 및 Undeploy합니다.

- **11장. XPLATFORM으로 X-UP 모델 호출**

X-UP Builder에서 개발한 모델을 XPLATFORM의 개발도구인 UX-Studio에서 사용하는 방법에 대해 설명합니다.

- **12~16장. X-UP의 기타기능 핸들링**

8~9장에서 설명된 기능 외X-UP에서 제공되는 주요 서비스에 대해 설명하고, 사용하는 방법에 대해 설명합니다.

- **17장. X-UP 배포안내**

X-UP 배포사항에 대해 설명합니다.

- **부록**

부록은 본문에서 자세히 설명하지 않고, 본 매뉴얼에서 필요한 스크립트 및 기본 설정에 대해서 간략히 제공합니다.

매뉴얼 표기법

본 매뉴얼은 독자의 이해도를 높이고자, 특별한 의미가 있는 단어나 문장은 별도의 표기법으로 표현했습니다. 다음은 그 표기법에 대한 설명입니다.

노트, 팁, 주의는 다음과 같이 제공됩니다.



노트는 본문에 간단하게 추가할 짧은 설명이나 참조, 논평을 제공하기 위해 사용합니다.



팁은 도움말 등의 팁을 제공하기 위해 사용합니다.



주의는 독자 또는 사용자의 주의를 환기하는 문장을 제공하기 위해 사용합니다.

1.

X-UP 소개

1.1 X-UP 이란?

X-UP의 정의

X-UP은 다양한 자원들(DataSource)을 대상으로 하여, 데이터를 수집 또는 융합하여 새로운 데이터셋으로 쉽게 생성하기 위한 웹어플리케이션 기반의 프레임워크입니다.

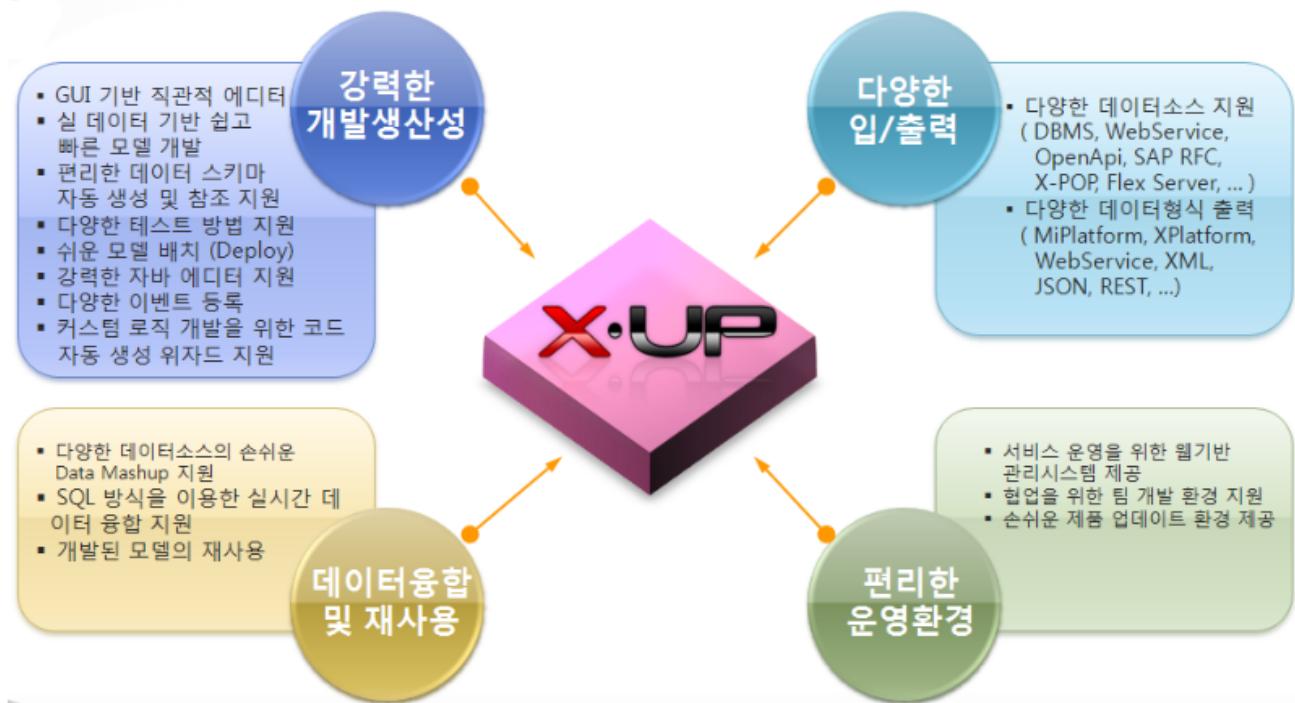
데이터를 수집, 가공, 저장하기 위한 논리 구조를 결정하고, 다양한 자원들로부터 데이터를 쉽게 수집 및 가공, 저장할 수 있는 컴포넌트들을 제공하며, 서비스를 개발하고 실행하기 위한 개발 도구와 API, 라이브러리 등을 제공합니다.

X-UP의 기능 및 특징

X-UP은 다양한 서비스를 혼합하여 새로운 서비스를 제공하는 Enterprise Mashup 기능을 제공합니다. X-UP은 다양한 데이터소스에서 다양한 포맷의 데이터를 수집하고 융합하여 다양한 방법으로 서비스할 수 있는 기능을 제공합니다.

X-UP의 개발은 X-UP Builder라는 개발도구를 사용하고, 다양한 데이터소스에서의 수집 로직과, 다양한 데이터 포맷의 파싱 로직을 개발합니다. 단순한 마우스 클릭, 드래그 앤 드랍 및 단순 값 입력만으로 로직 개발이 가능합니다.

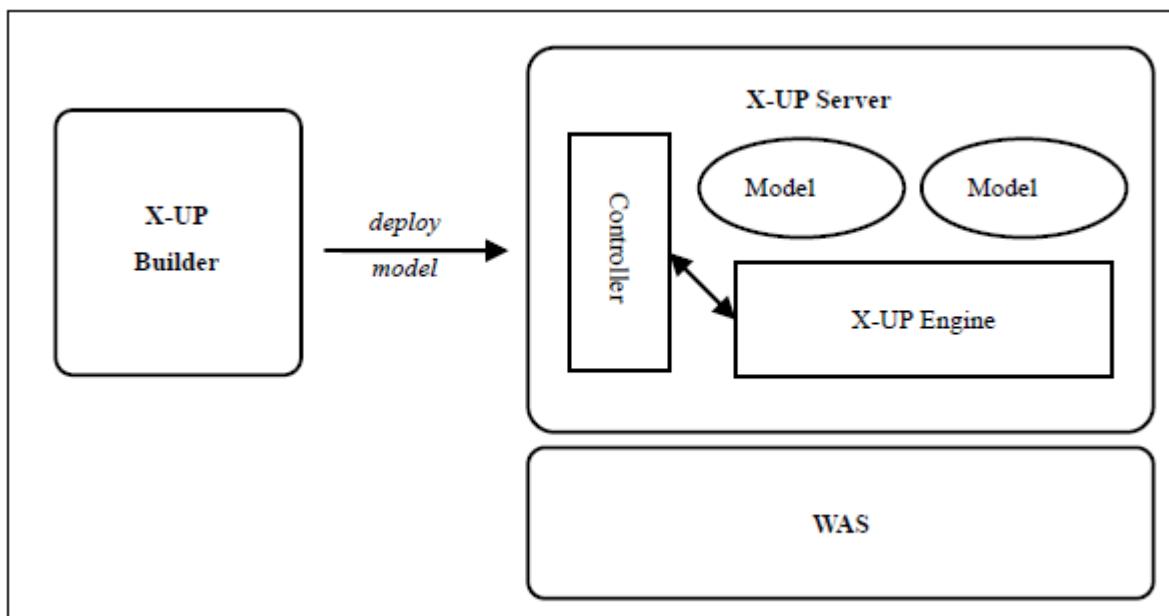
일반적인 매쉬업 애플리케이션들이 해당 자원으로부터 데이터를 가져오기만 하는 기능에 특화되어 있는것과 비교하여, X-UP은 다른 자원으로부터 데이터를 획득하고 가공하여 새로운 데이터를 생성하는 기능과, 해당 자원에 생성된 데이터를 업데이트할 수 있는 기능을 동시에 지원합니다.



X-UP의 구성

X-UP은 X-UP 모델, X-UP Builder, X-UP Server로 구성됩니다.

- X-UP 모델 : 수집/융합하는 로직 단위
- X-UP Builder : 수집/융합하기 위한 X-UP 모델을 개발하는 개발도구
- X-UP Server : X-UP 모델이 실제로 구동되는 서버



2.

X-UP 개요

X-UP은 다양한 데이터소스에서 데이터를 가공하여 새로운 데이터셋으로 쉽게 생성하기 위한 프레임워크입니다.

X-UP은 크게 X-UP 모델, X-UP Builder, X-UP Server의 세가지 기술로 분류합니다. 이 장에서는 X-UP의 세가지 기술과 관리자 서비스인 X-UP Administrator에 대해 설명합니다.

2.1 X-UP 모델

데이터셋을 수집하고 융합하기 위해서는 메타데이터라 칭하는 다양한 정보들이 설정되어야 하고, 더불어 각 값을 결정하기 위한 비즈니스 로직이 정의되어야 합니다. 하나의 데이터셋을 수집하고 융합하기 위한 메타데이터와 사용자 로직을 통칭하여 하나의 ‘X-UP 모델’이라고 칭합니다.

X-UP 모델은 실제 매쉬업을 위한 메타정보와 로직을 담고 있으며, X-UP Builder를 이용하여 개발합니다. X-UP Builder에서 개발된 X-UP모델은 X-UP Server에 배치되고, 외부에서 X-UP Server에 구동을 요청하면 X-UP Server는 해당 모델을 구동시킵니다.

X-UP에서의 모든 동작은 X-UP 모델 단위로 이루어집니다. X-UP Builder에서 X-UP 모델 단위로 개발하고 배치하며, 외부에서의 요청 또한 X-UP 모델 단위로 처리됩니다. 예를 들어 ‘Cat’이라는 Automation 모델을 개발하고 이를 X-UP Server에 배치한 후에는, 외부에서 ‘Cat’이라고 지칭하는 모델로부터 데이터를 수집하고 처리하여 결과를 받습니다.

X-UP 모델은 다양한 데이터소스를 사용할 수 있으며 데이터소스로부터 원시 데이터를 수집하는 방법을 정의할 수 있습니다. 그리고 수집한 원시 데이터를 수집하는 방법을 정의할 수 있습니다. 또 수집한 원시 데이터를 정형화된 데이터셋으로 변환하는 방법을 정의하거나 복수의 데이터셋을 하나의 데이터셋으로 융합할 수 있습니다. 필요한 경우, X-UP 사용자는 X-UP 모델에 데이터 수집 또는 융합에 필요한 임의의 로직을 추가할 수 있습니다.

Domain

X-UP에서 도메인이란 하나의 프로젝트를 칭합니다.

X-UP 모델은 각 도메인으로 구별합니다. 예를 들어 petStore도메인에 Order라는 모델이 있고, flowerStore도메인에 Order라는 모델이 있을 경우 두 모델은 별개의 것으로 처리됩니다. 두 모델의 이름은 같지만 두 모델은 서로 다른 도메인에 속한 것으로 ‘petStore의 Order’와 ‘flowerStore의 Order’로 구별됩니다. 따라서 외부 어플리케이션이 X-UP을 호출할 때는 반드시 도메인의 이름과 X-UP모델의 이름을 같이 명시하여야 합니다.

X-UP Builder에 의하여 X-UP Project가 생성되면 대상 도메인의 이름을 설정할 수 있습니다. X-UP Builder에 의하여 생성된 X-UP Project는 도메인에 해당합니다. 하나의 X-UP Project에는 복수의 도메인이 있을 수 없고, 하나의 도메인을 복수의 X-UP Project가 공유할 수 없습니다. 만약 서로 다른 X-UP Project 가 같은 도메인 이름을 사용하고 같은 X-UP Server에 배치된다면, 두 프로젝트의 X-UP 모델들이 전부 배치되는 것이 아니라 나중에 배치된 모델이 앞서 배치된 모델을 덮어쓰게 됩니다.

2.2 X-UP Server

X-UP Server는 배치된 X-UP 모델을 구동하는 서버이며, WAS위에서 웹 애플리케이션으로 동작합니다.

X-UP Builder에서 개발하는 것은 데이터의 수집, 융합, 가공을 위한 로직일 뿐 실제 기능은 X-UP Server에서 실행합니다.

X-UP Server는 X-UP에서 제공되는 각 서비스에 대해 Bundle형태로 구성되어 있습니다.

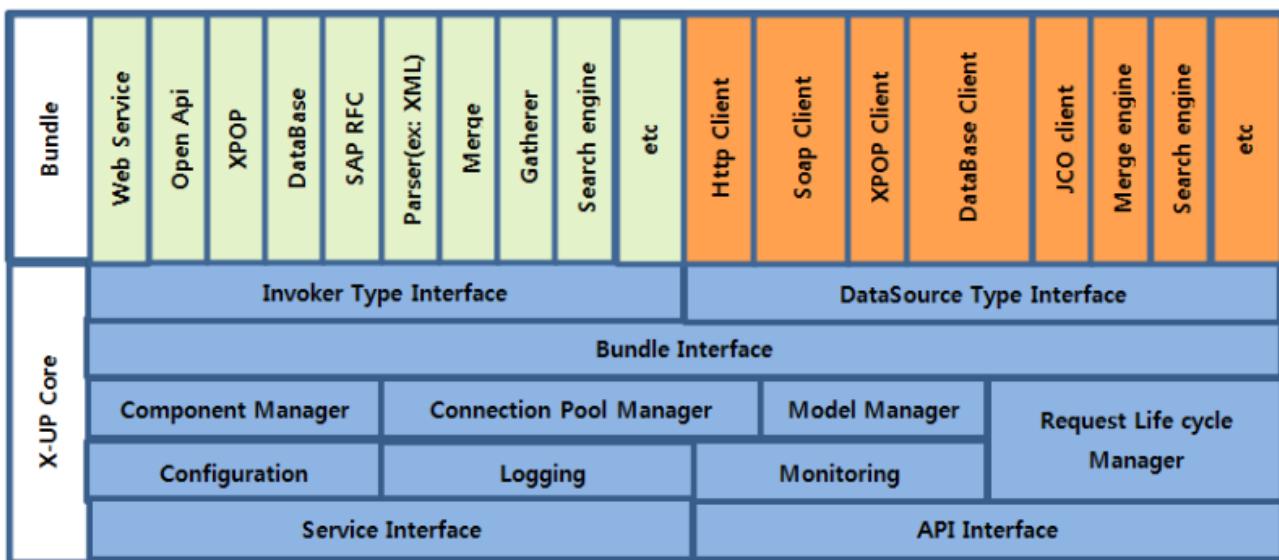
Bundle 처리 명세

Bundle이란 X-UP Core에서 동작할 수 있는 인보커 혹은 데이터소스와 같은 모듈 단위를 지칭합니다.

X-UP에서 제공하는 Bundle의 유형은 다음과 같습니다.

- **DataSource Type** : 데이터소스를 이용하여 외부와 Connection을 연결하고 원시데이터를 획득하는 주체
- **Invoker Type** : 데이터소스 탑입의 Bundle을 이용하여 획득 한 데이터를 가공하여 X-UP Builder 사용자가 지정한 흐름에 맞게 서비스를 실행하는 주체
 - Sub Type으로는 Gatherer와 Invoker 두가지 형태로 제공됩니다.

데이터소스 탑입의 Bundle에서 생성되는 Connection은 Pool을 통하여 일원화된 관리체계를 가지고 있습니다



Bundle 설치

X-UP 구동 시 Core에 의하여 자동적으로 Bundle이 설치가 됩니다.

Bundle 버전과 Engine(X-UP Server Engine) 버전을 이용하여 버전이 변경되지 않는 한 X-UP 구동 시 최초 한 번만 압축을 해제합니다.

Bundle 제거

Bundle의 Uninstall의 경우 Bundle 폴더와 압축파일을 제거하면 Uninstall이 수행되게 됩니다.

Bundle Repository

제공 된 Bundle 은 압축 해제 시 다음과 같은 폴더 구조를 가지게 됩니다.



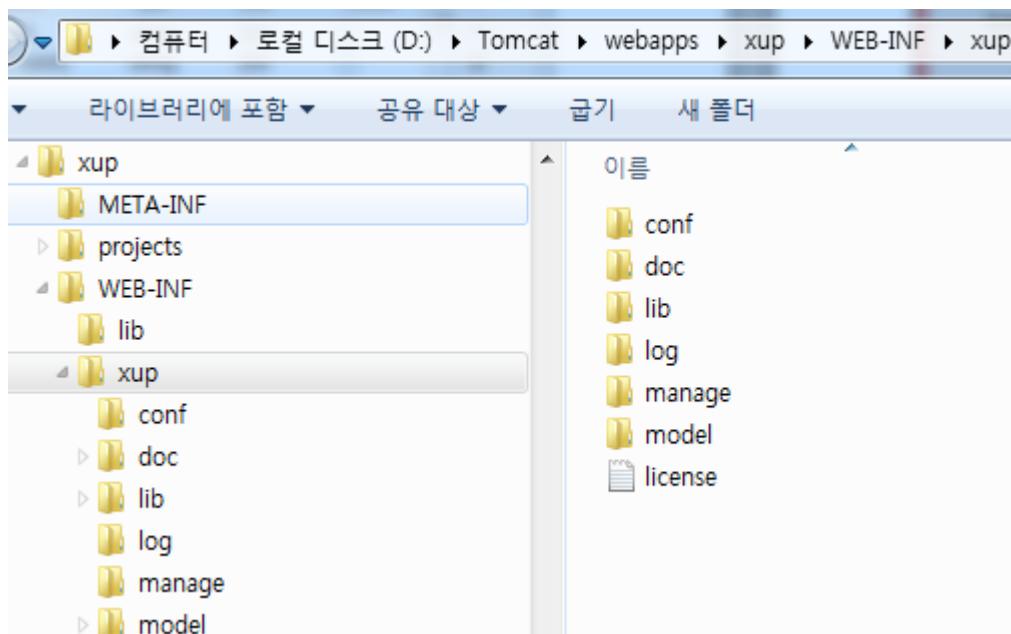
- **conf :** Bundle 이 구동 시 필요한 설정 정보를 가진 **bundle_config.xml** 이 위치합니다.
 - 추가 적으로 **xup_config.xml** 에 Bundle 의 설정 정보를 위치시켜도 설정이 적용됩니다.
 - 설정의 우선 순위는 **bundle_config.xml < xup_config.xml** 입니다.
- **lib :** Bundle 에 필요 한 Dependency Library 들이 위치 합니다. 각 Bundle 은 별도의 ClassLoader 로 로딩 되어 Library 사용의 독립성이 보장됩니다.

- META-INF : Bundle 의 메타파일 정보가 기록 된 MANIFEST.MF 파일이 위치합니다.

Name	Description
Bundle Adaptor	Bundle이 동작하기 위한 구현체 ex) Invoker, DataSource
bundle Id	Bundle의 명칭 (xup_cofing.xml에 기록된 Bundle의 id와 매칭됩니다.)
Bundle Core Jar Name	Bundle의 Core Jar 이름
Bundle Implementation Jar Name	외부에 노출된 클래스 파일입니다.
Bundle Version	Bundle을 관리하기 위한 버전
Engine Version	Bundle에 최적화된 Engine버전

X-UP Server 파일 구조

X-UP 서버의 파일 구조는 다음과 같습니다. 본 매뉴얼에서는 WAS로 Tomcat을 사용합니다. 아래 그림은 Tomcat WAS 아래에 webapps xup WEB-INF xup 하위의 파일 구조입니다.



Name	Description
conf	conf 폴더는 X-UP 서버의 환경설정 파일들을 가지고 있습니다. X-UP 서버 라이선스 파일이 필요하다면 conf 폴더안에 위치하도록 합니다.

Name	Description
doc	X-UP 가이드 문서와, 배포 내역, javadoc 등의 문서를 포함합니다.
lib	X-UP 서버의 Library입니다. 해당 폴더 아래에는 bundle 폴더와 core 폴더로 구분되어져 있습니다.
log	X-UP 서버에 대한 로그가 해당 폴더 아래에 생성됩니다.
manage	X-UP 관리자에 대한 데이터베이스 정보를 가집니다.
model	서버에 디플로이된 모델에 한해서 model 폴더에 프로젝트 단위로 생성됩니다.
license	라이선스 등의 문서입니다.

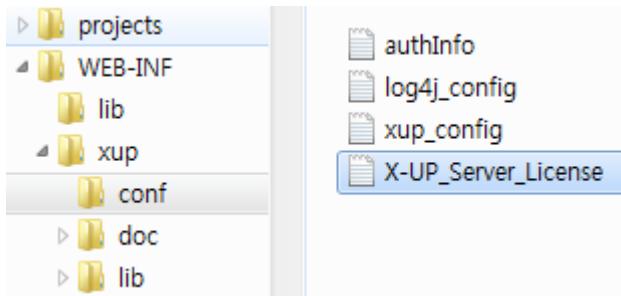
설치 및 업데이트

1. X-UP 서버를 만들기 위한 WAS를 설치합니다. 본 매뉴얼에서는 apache-tomcat-6.0.35를 사용합니다.
2. 다운로드 받은 apache-tomcat-6.0.35.zip을 C: 드라이브에 C:\Apache-tomcat-6.0.35로 압축해제 합니다.
3. xup.war 파일을 C:\Apache-tomcat-6.0.35\WEB-INF\webapps 디렉토리에 복사합니다.
4. Tomcat을 재기동 하면 webapps 폴더에 xup 폴더가 생성됩니다.
5. 서버 라이선스를 등록합니다. 방법은 아래 라이선스 등록을 참조하세요.

라이선스 등록

X-UP 기능을 정상적으로 사용하기 위해서는 라이선스를 등록해야 합니다. Tomcat, Weblogic등 Wasdnldp 설치된 X-UP Server의 라이선스 적용에 대해 설명합니다.

1. WAS에 X-UP을 설치합니다.
2. 제품구매 시 발급받은 라이선스 파일(X-UP_Server_License.xml)을 conf 폴더에 복사하여 갖다 놓습니다.



3. WAS를 재기동합니다.

2.3 X-UP Builder

X-UP Builder는 X-UP모델을 개발하는 개발도구입니다.

X-UP 사용자는 X-UP Builder를 사용하여 쉽고 빠르게 X-UP 모델을 개발하고 이를 서버에 배치시킬 수 있고, X-UP Builder를 사용하여 개발한 서비스는 X-UP Library를 호출하고, X-UP Library는 배치된 모델을 사용하여 데이터를 수집하고 융합하여 생성된 데이터셋을 반환합니다.

X-UP Builder는 드래그 앤 드롭과 직관적인 GUI Editor를 제공하여 Graphical한 방법으로 모델을 개발 및 테스트하며 서버에 배치하는 모든 작업을 개발 경험이 많지 않은 개발자라 할지라도 손쉽게 개발할 수 있도록 다양한 편의기능을 제공합니다.

X-UP Builder는 Eclipse Platform 기반위에 JDT, EMF, GEF 라이브러리들이 사용되어 개발되었습니다. X-UP Builder가 사용하는 JRE 및 Eclipse라이브러리 버전은 다음과 같습니다.

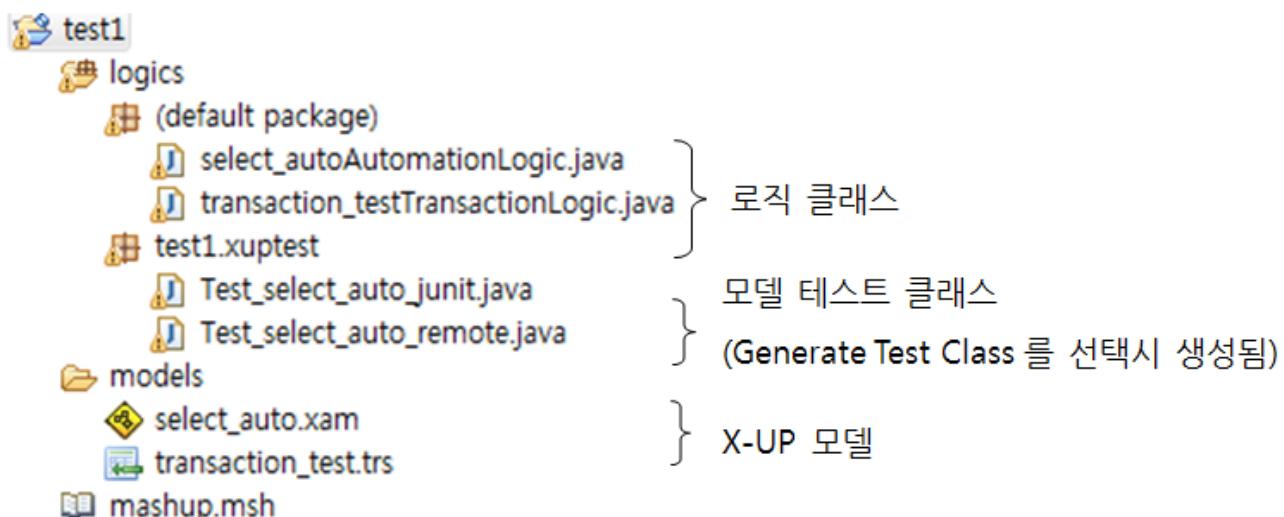
Name	Version	Description
JRE	1.6.0_31	Java Runtime Environment
Eclipse Platform	3.7.2	Eclipse Platform
JDT	3.7.2	Java Development 개발도구
EMF	2.7.0	Eclipse Modeling Framework
GEF	2.7.0	Graphical Modeling Framework

X-UP Builder에서 제공하는 기능은 다음과 같습니다.

- Create X-UP Project
- Define DataSource
- Find and View DataSource
- Model Test
- Debugging
- Deploy to server
- Validate
- Refactoring

X-UP Builder 파일 구조

X-UP Builder의 파일 구조는 다음과 같습니다.



Name	Type	Description
X-UP Project	project	X-UP 모델개발을 위한 프로젝트로서 프로젝트 이름이 곧 도메인이 됩니다. 개발이 완료된후 특정 모델을 호출할 때 프로젝트명은 중요한 키값이 됩니다.

Name	Type	Description
logics	src folder	로직정보를 담고있는 자바 소스들이 위치합니다. 로직 소스는 모델이 생성될 때 에디터에 의해 자동으로 생성되며 사용자에 의해 수정될 수 있습니다.
userLib	src folder	유저 라이브러리가 추가되면 자동생성됩니다.
logic class package	package folder	로직클래스는 패키지가 없이 생성되므로 가상의 패키지명인 default package 형태로 보입니다.
test class package	package folder	테스트 클래스가 생성될 때 자동으로 생성되며, 패키지명은 projectName + '.' + xuptest으로 자동 생성됩니다.
models	folder	에디터에서 작성된 모델들이 위치한다. 프로젝트안에서 생성된 모든 *.mdl, *.trs 모델들은 models 폴더 밑에 생성됩니다.
logic class	class	X-UP 모델과 같이 생성된 로직 클래스
test class	class	X-UP 모델에 의해 자동생성된 테스트 클래스
X-UP model	*.trs, *.xam	X-UP 모델로써 *.trs 는 Transaction Model, *.xam은 Automation Model 입니다.
mashup.msh	*.msh	X-UP Project에서 개발된 모델들에 대한 모든 중요한 정보를 가지고 있습니다. mashup.msh는 X-UP Project에 반드시 하나 존재해야 합니다.

logics 폴더에는 로직 클래스가 위치합니다. 기본적으로 생성되는 로직 클래스들은 패키지명이 없는 클래스로 생성되고 **logics** 폴더는 Class Path에 등록되어 있습니다.

X-UP Model에서 생성하는 로직 클래스에는 User logic class와 Base logic class로 나누어 집니다.

User logic class는 사용자가 모델 개발시 별도의 로직을 담을수 있도록 인터페이스 메소드들이 미리 정의된 클래스 파일로써, 모델이 처음 생성될 때 자동으로 생성되고 모델이 생성될때 속성값 중 logic 항목에 같이 생성한 User logic class이름을 자동으로 등록합니다.

주의) User logic class이름은 반드시 모델의 logic 항목에 정의된 이름과 같아야 합니다.

Base logic class 역시 모델 생성시 자동으로 생성됩니다. 그러나 이 로직 클래스는 기본적으로 사용자에겐 숨겨지고, 에디터가 Save 될때 마다 자동으로 내용을 갱신합니다. 즉 에디터가 자동 갱신하는 로직 클래스이므로 수정하더라도 저장시 다시 재작성됩니다. 모델 개발시 사용자는 Base logic class의 내용을 알 필요가 없으므로 디폴트로 숨겨져 있습니다.

userLib 폴더는 [project > import > X-UP > user library] 를 통해 사용자 라이브러리가 등록되면 자동으로 생성되고 해당 폴더밑에 모든 *.jar 들이 위치하게 됩니다.

모델 개발이 끝나고 서버에 배치(deploy)될 때 userLib 밑에 존재하는 모든 jar들도 logic class와 함께 추가됩니다.

models 폴더는 개발된 모든 X-UP 모델들이 위치하게 됩니다. 모든 *.trs, *.xam 파일들은 반드시 models 폴더밑에 있어야합니다.

mashup.msh 파일은 Deploy를 위한 Server URL이나 DataSource 정보와 같은 해당 X-UP 프로젝트에서 모델 개발시 필요한 중요한 정보를 포함합니다.

2.4 X-UP Administrator

X-UP Administrator는 X-UP 관리자 서비스입니다.

X-UP 관리자 서비스로 X-UP 시스템의 하드웨어 및 소프트웨어 사양을 브라우저에서 확인할 수 있으며 로그 및 deploy된 X-UP model에 대한 확인 및 제어가 가능한 서비스를 제공 합니다.

로그인 과정을 거쳐 deploy된 Domain에 대한 목록과 리소스, 이력등을 확인할 수 있으며 시스템에 대한 감시 및 알림 서비스를 제공하여 운영 및 관리에 편의를 제공합니다

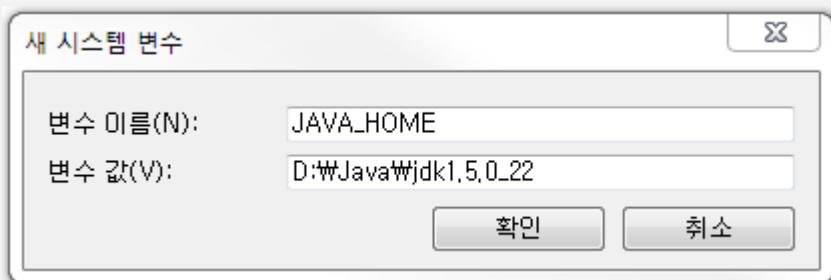


X-UP 관리자 시스템에 대한 자세한 가이드는 해당 실행 프로그램의 Helper를 이용하시길 바랍니다.

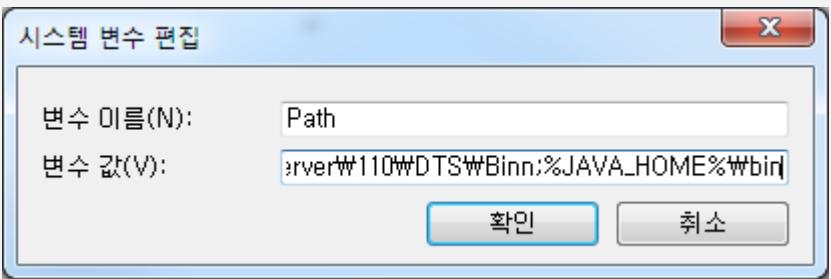


X-UP 관리자 시스템 사용시 필수사항 _ JDK(Java Development Kit) 설치 및 환경설정

1. <http://www.oracle.com>을 접속하여 메뉴에서 Downloads를 선택한 후 JDK를 다운받습니다.
2. 다운받은 exe 파일을 실행하여 설치합니다.
3. JDK 설치가 완료되면 환경변수 설정을 합니다.
4. 시스템 등록 정보(시스템 속성)에서 환경 변수 선택, 새로 만들기 선택
5. 시스템 변수를 다음과 같이 등록합니다.



6. [Path]를 선택하고 [편집]을 클릭하여 다음과 같이 추가합니다.



반드시 JAVA_HOME에 대하여 JDK 설치 경로를 입력하여야 합니다.(JRE 안됨)

3.

X-UP Builder 기초

이 장에서는 X-UP Builder 설치 및 삭제, 업데이트에 대하여 설명합니다.

3.1 X-UP Builder 설치 및 삭제

이 절에서는 X-UP Builder 설치 및 삭제 방법을 설명합니다. X-UP Builder는 이클립스 기반으로 Java 5.0 이상의 환경에서 실행되며 X-UP 모델을 개발하고 테스트할 수 있는 통합 개발 환경을 제공합니다.

X-UP Builder 설치 파일 종류

항목	Description
X-UP Builder Install(32bit)	32 bit용 X-UP Builder 설치 파일입니다.
X-UP Builder Install(64bit)	64 bit용 X-UP Builder 설치 파일입니다.

시스템 요구사항

X-UP Builder 설치 시 필요한 시스템 요구사항은 다음과 같습니다.

항목	요구사항 (Recommended)
운영체제(OS)	Windows 2000 Windows XP Windows Vista Windows 7 Windows 2000 Server Windows 2003 Server Linux(2.6)

항목	요구사항 (Recommended)
CPU	PentiumIV(2GHz)이상
메모리	1GB RAM 이상
디스크 공간	100MB 이상

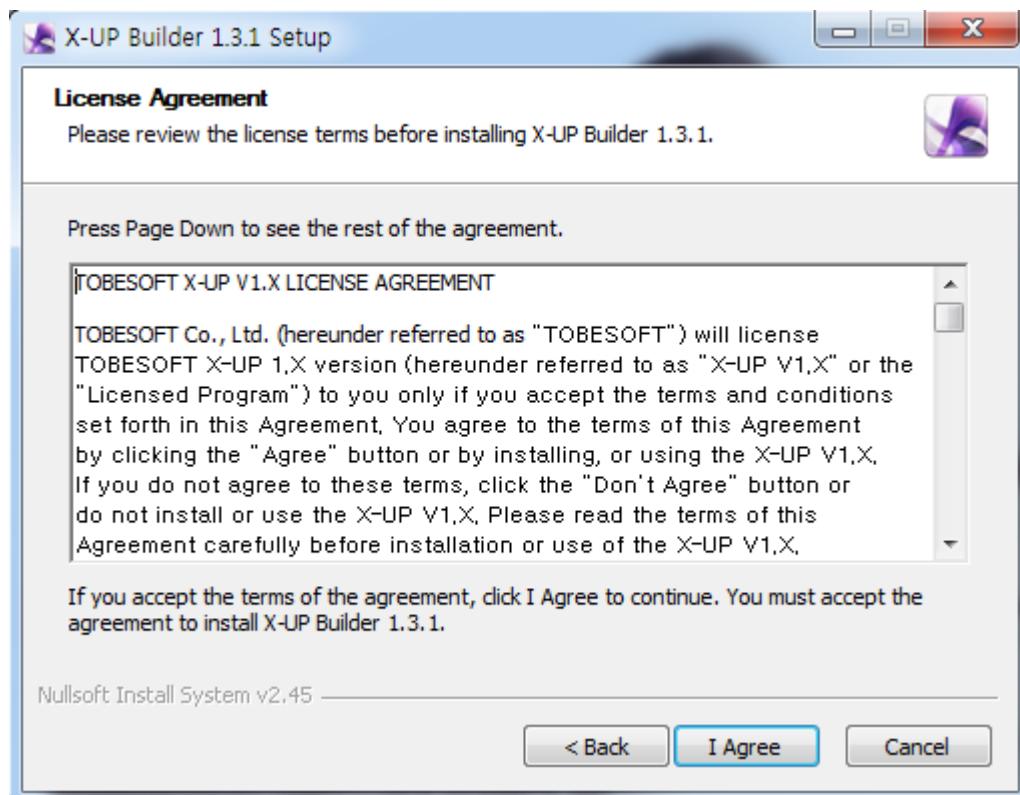
설치

X-UP 개발 개발도구인 X-UP Builder를 설치합니다. 본 매뉴얼에서는 Window 7 의 운영체제의 데스크탑에서 64비트용 X-UP Builder 설치 파일로 안내합니다.

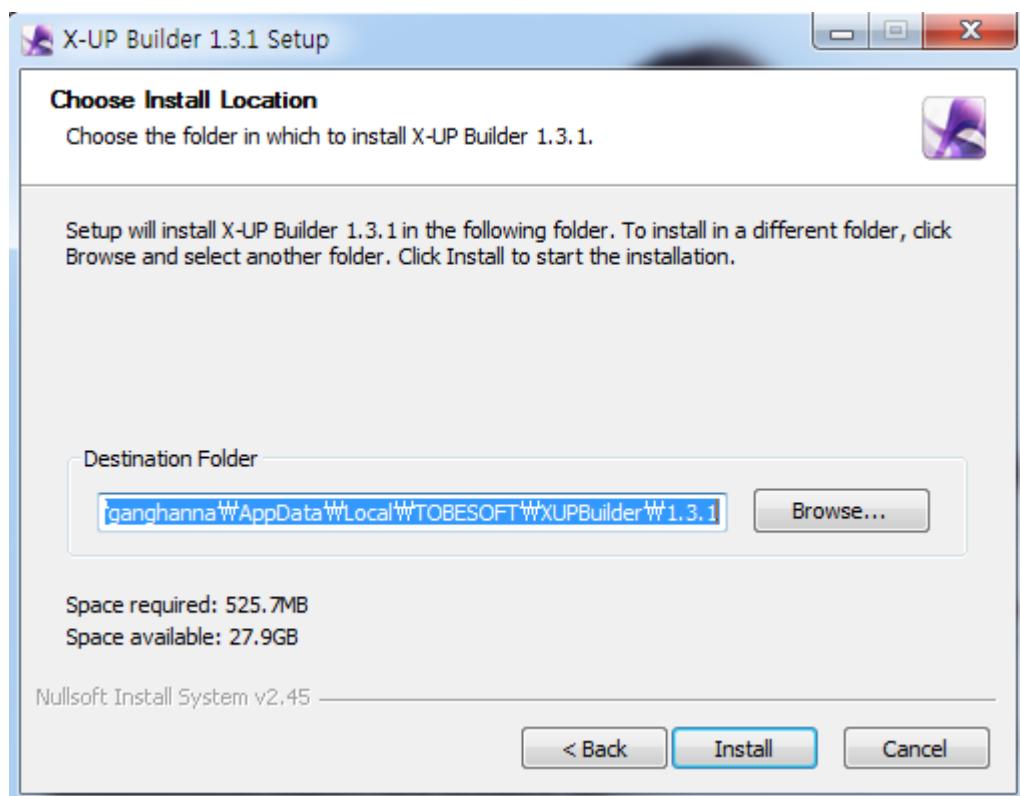
- 설치파일 X-UP_Builder_1.3.1_install.exe를 실행합니다.



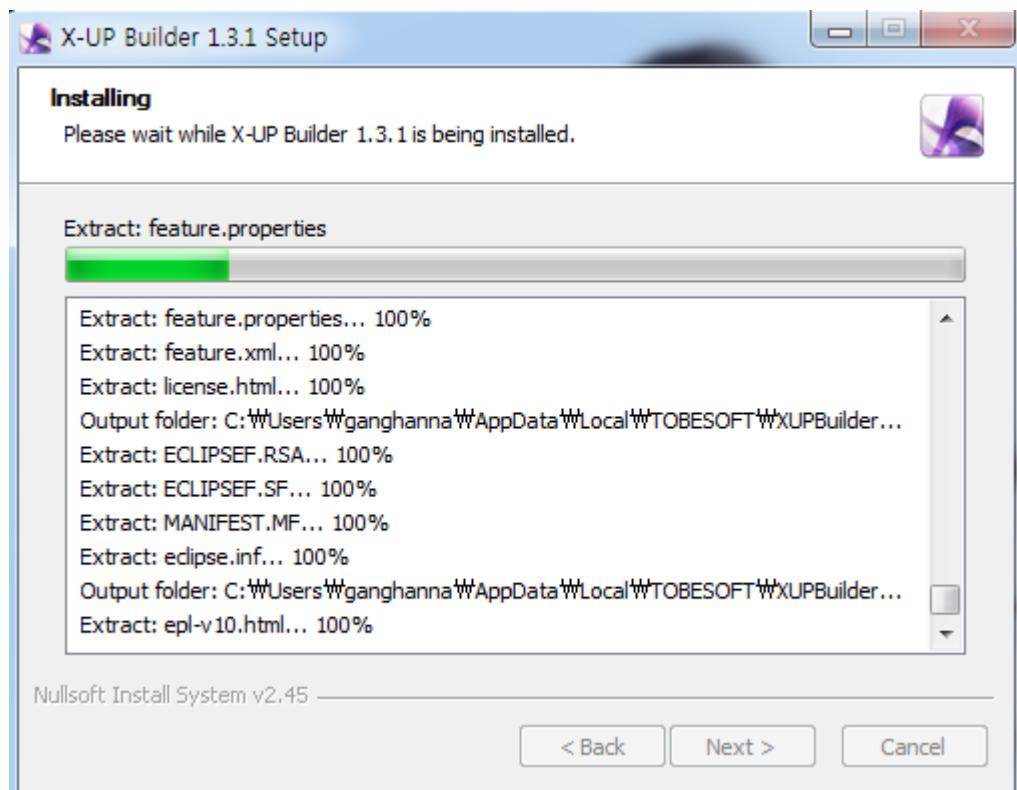
- X-UP Builder의 라이선스를 확인합니다. X-UP Builder 라이선스에 동의한다면 'I Agree'를 선택하여 다음 페이지로 진행합니다.



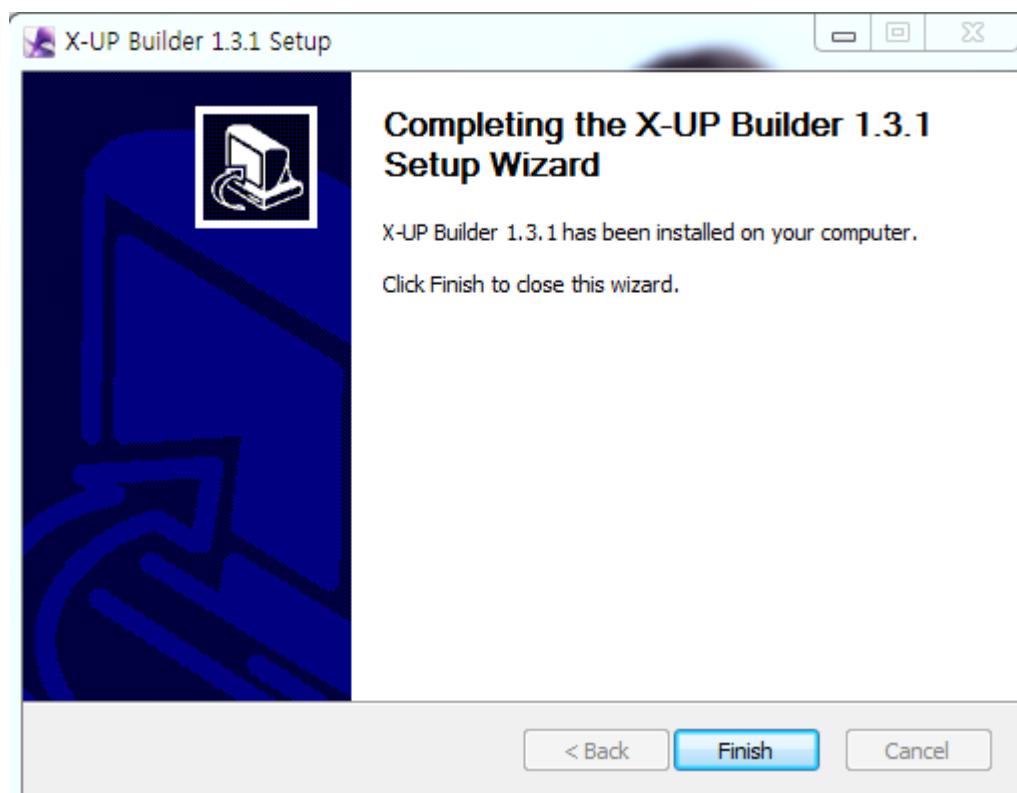
3. X-UP Builder가 설치될 폴더를 선택합니다. 변경을 원하는 경우에는 'Browse...' 버튼을 클릭하여 폴더를 변경합니다.



4. 지금까지 과정이 모두 정확하다면 'Install' 버튼을 클릭하여 설치를 진행합니다.



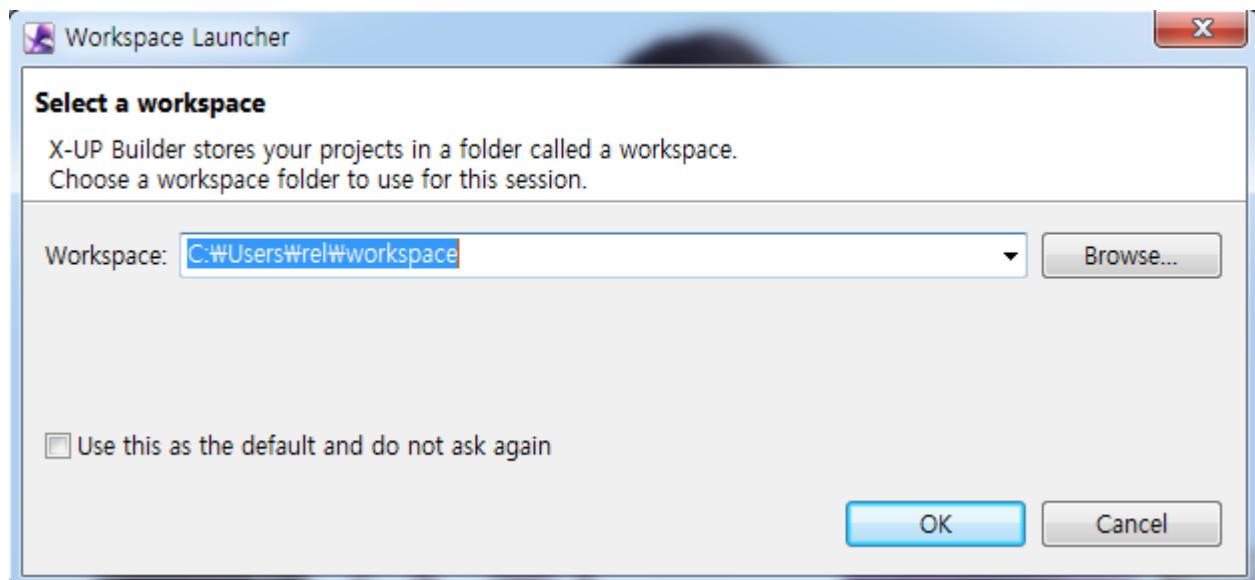
5. 설치가 정상적으로 완료되면 아래의 화면을 볼 수 있습니다. 'Finish' 버튼을 클릭하여 설치를 종료합니다.



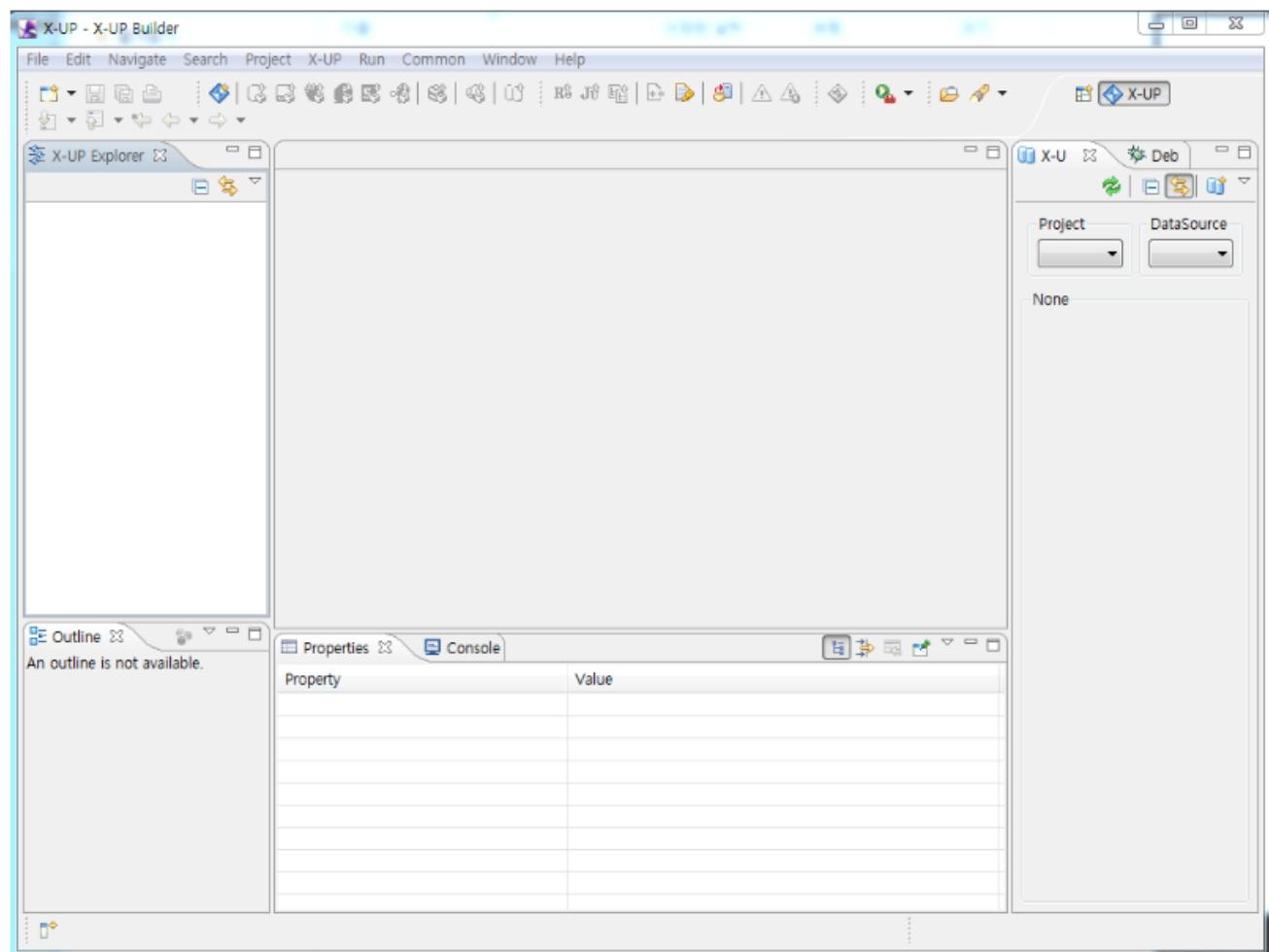
6. 설치완료 된 X-UP Builder를 실행합니다.

7. 작업공간이 될 워크스페이스(workspace)를 설정합니다. 워크스페이스 위치는 사용자가 원하는 위치로 정할 수

있습니다.



8. X-UP Builder의 실행화면입니다.

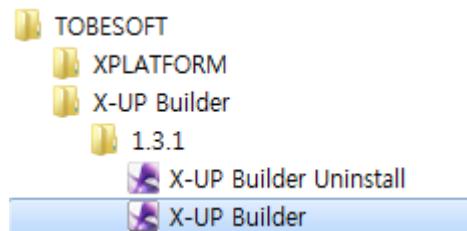


X-UP Builder라이선스를 등록합니다. 방법은 아래 라이선스 적용을 참조하세요.

설치 정보 및 버전 확인

[시작 > 모든 프로그램 > TOBESOFT > X-UP Builder > 1.3.1] 경로에서 X-UP Builder 및 X-UP Builder Uninstall 단축 아이콘을 확인할 수 있습니다.

1. 시작메뉴



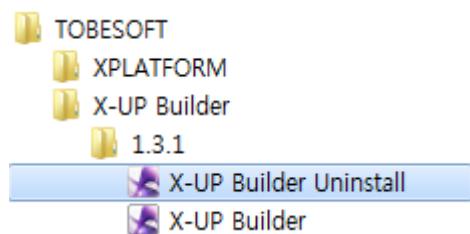
2. 단축아이콘

	메뉴	설명
	X-UP Builder Uninstall	X-UP Builder 삭제 단축 아이콘
	X-UP Builder	X-UP Builder 단축 아이콘

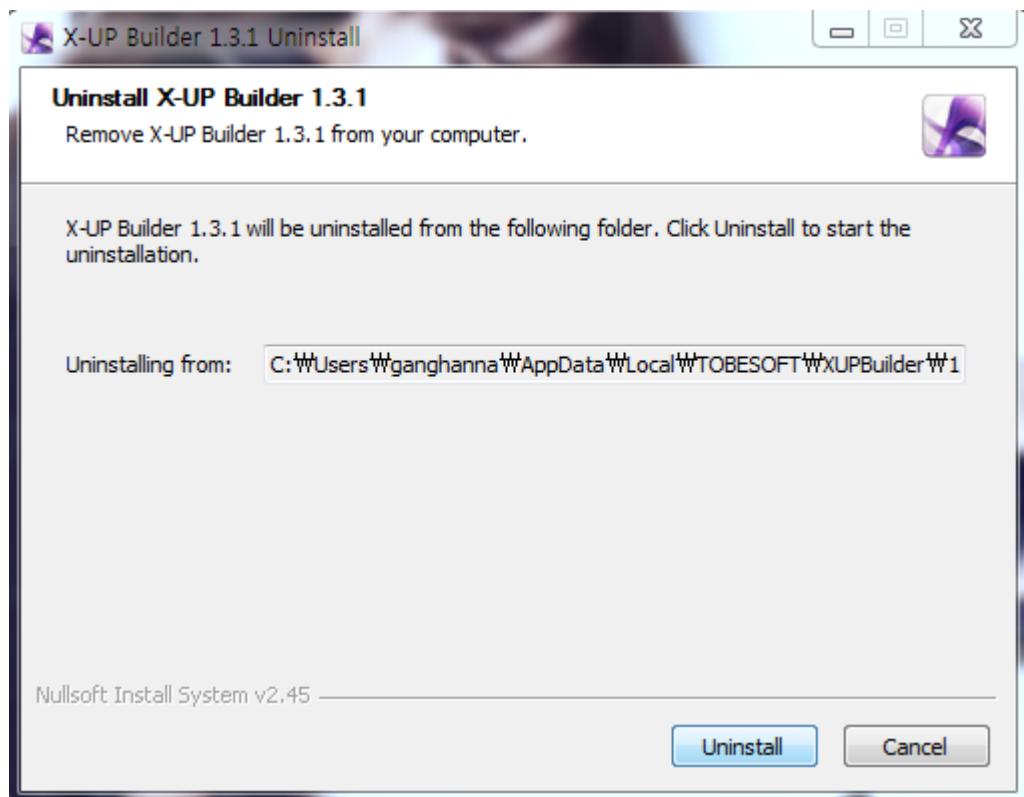
삭제

X-UP Builder를 삭제하는 방법에 대해 설명합니다.

1. 삭제파일 X-UP Builder Uninstall을 실행합니다.



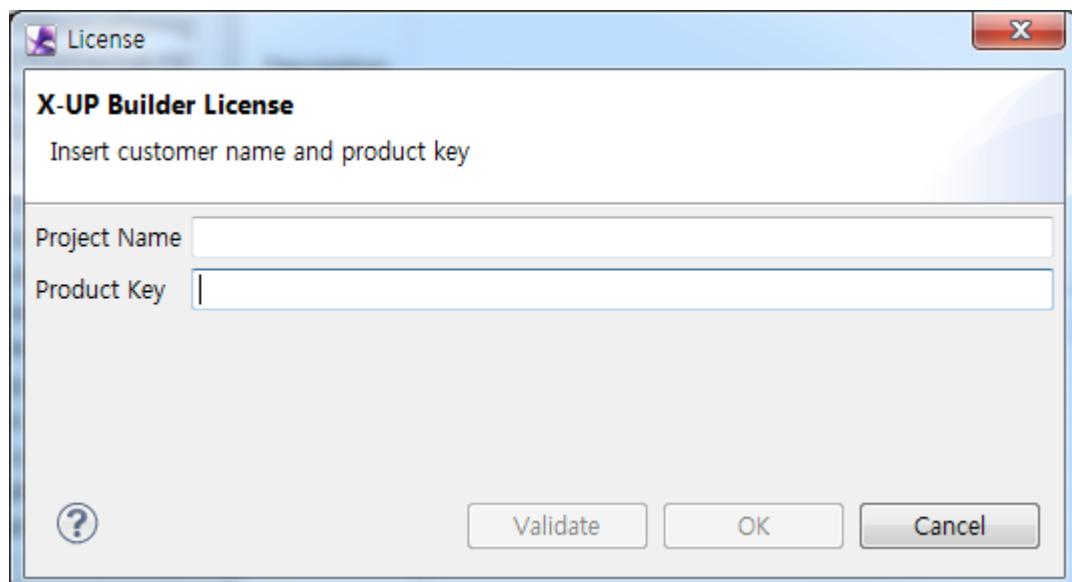
2. ‘Uninstall’ 버튼을 클릭하면 삭제가 시작됩니다.



3. 사용자의 컴퓨터에 설치된 프로그램 및 설치정보를 삭제 후 완료합니다.

라이선스 적용

X-UP Builder 라이선스 등록 과정은 다음과 같습니다.



License Manage Dialog는 2개의 정보를 입력해야 합니다.

- Project Name
- Product Key

X-UP Builder를 실행 한 후 License 입력ダイ얼로그에 Project Name과 Product Key를 입력하고 Validate버튼을 클릭하여 라이선스 정보를 검증합니다. 검증이 완료되면 OK버튼을 클릭합니다.

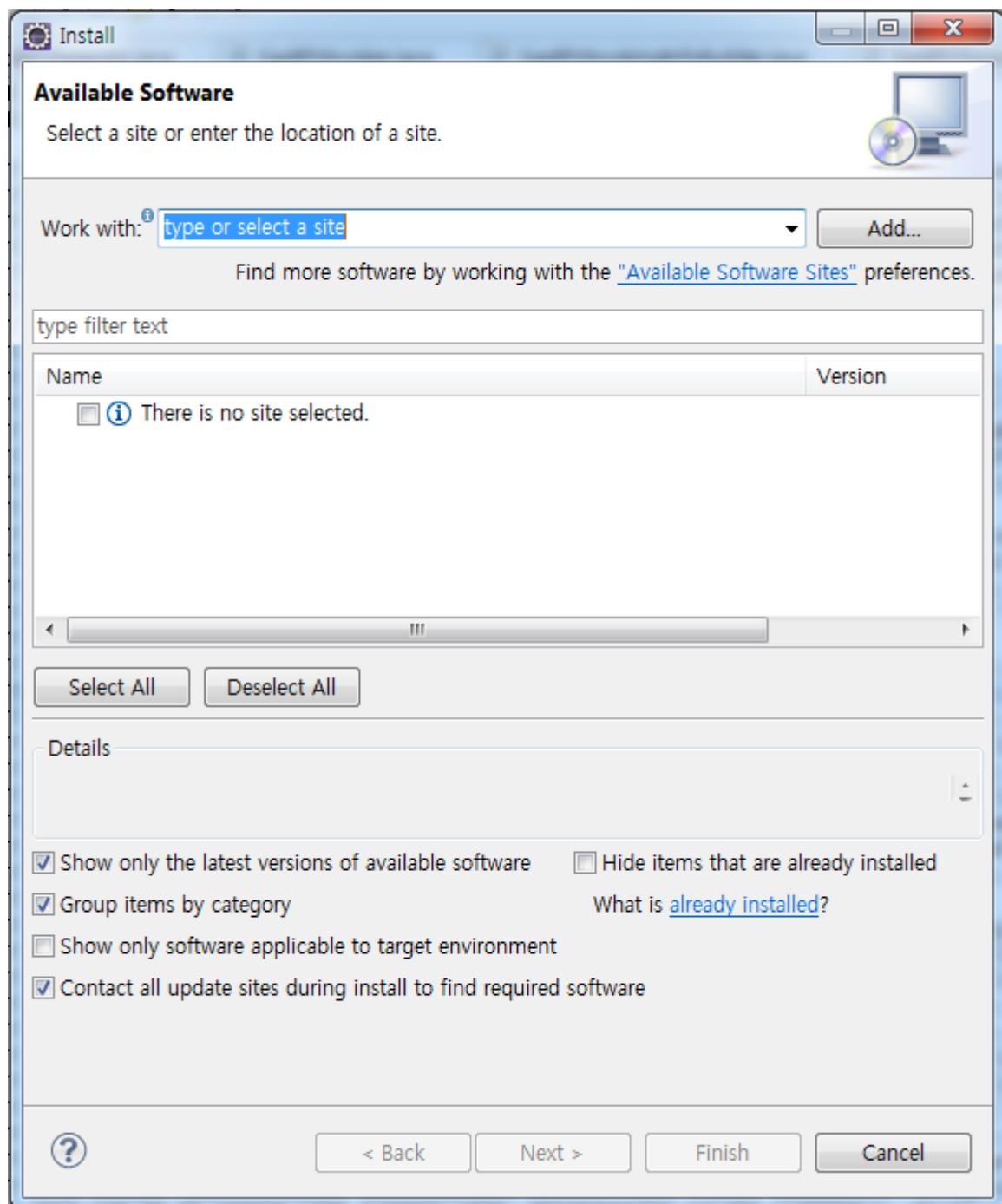


3.2 X-UP Builder 버전 패치 가이드

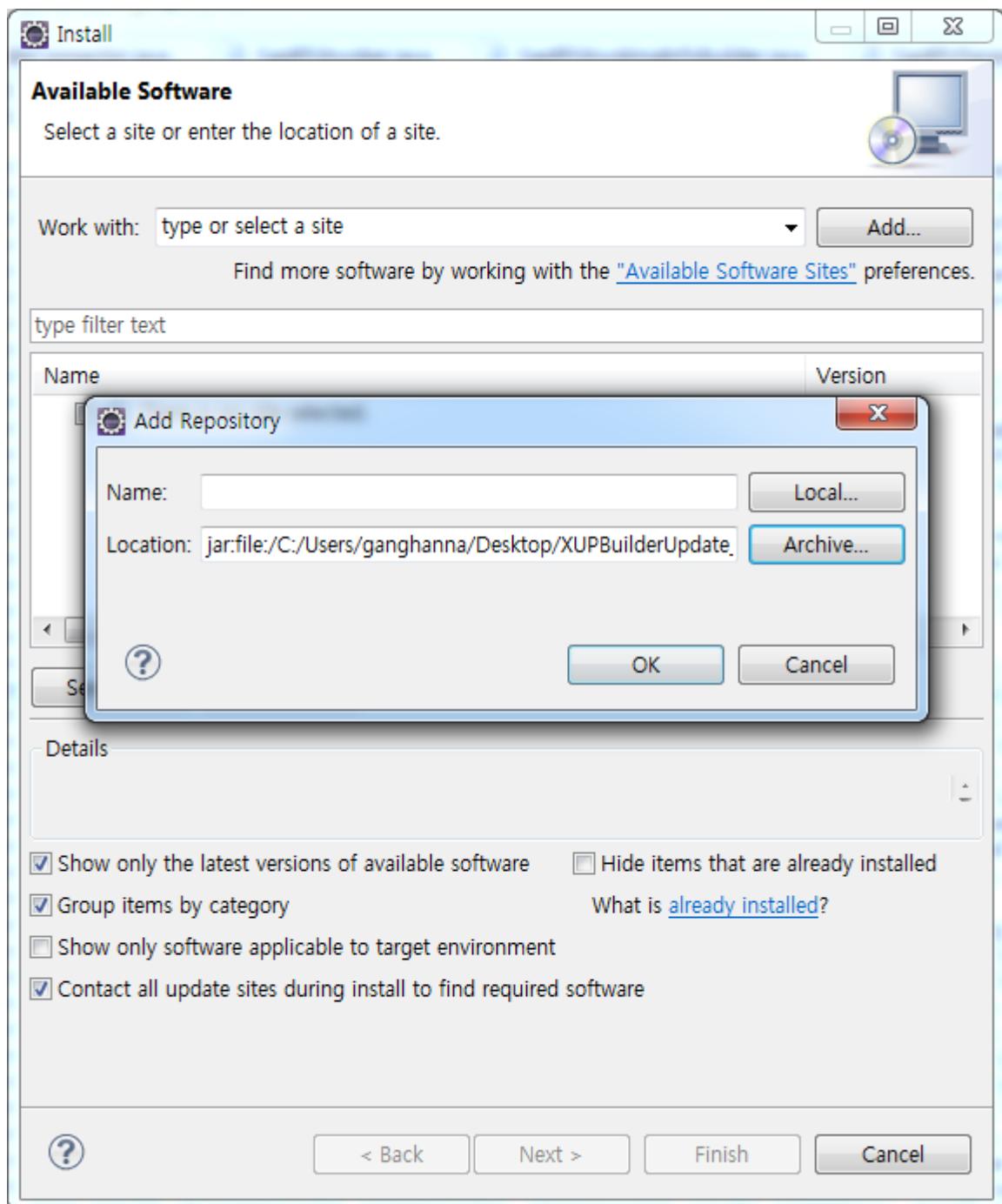
X-UP Builder 버전 패치(Update) 방법

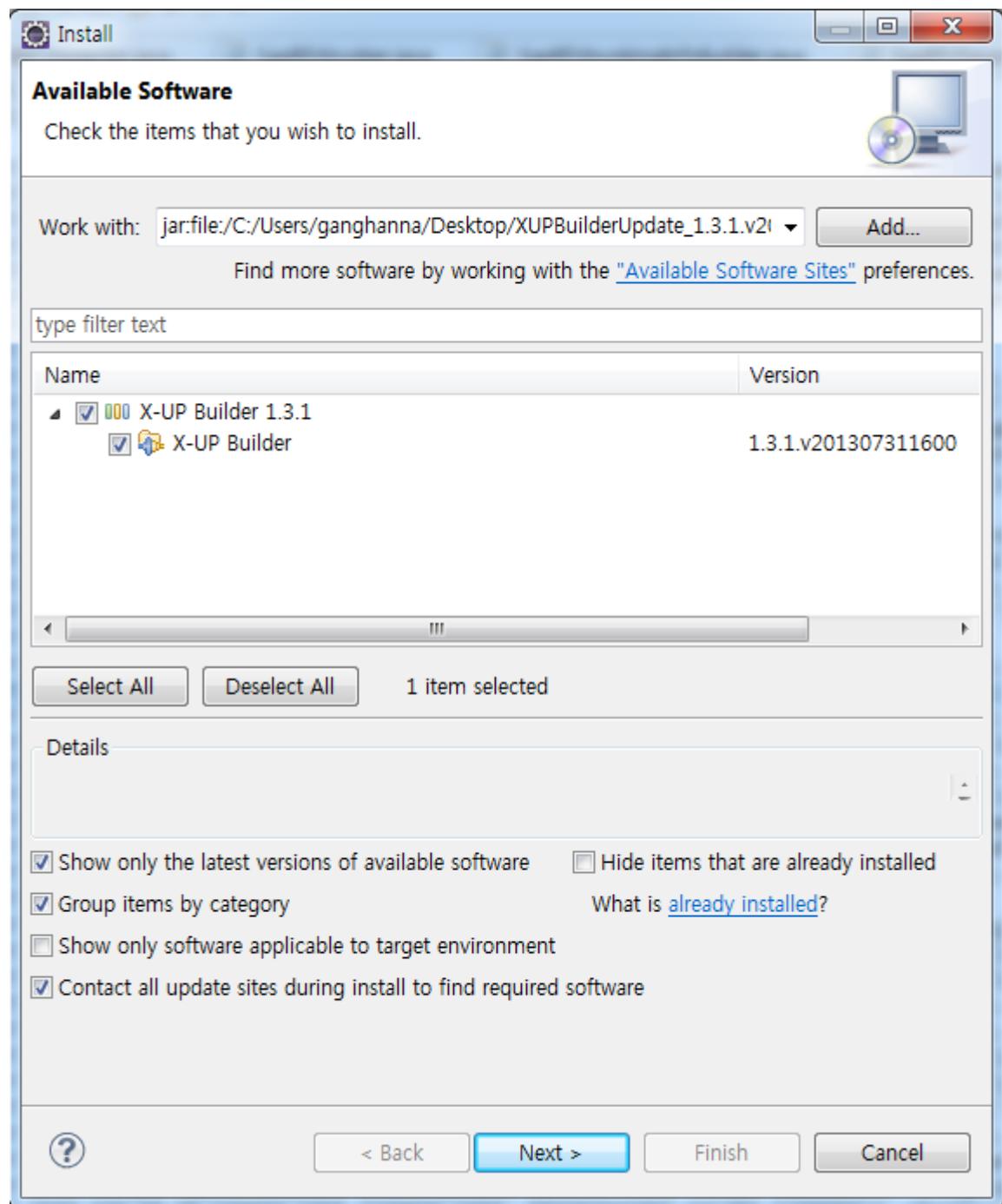
본 매뉴얼에서는 배포된 빌드파일이 Desktop에 저장되어 있다는 가정하에 기술합니다.

1. X-UP Builder를 실행합니다. 상단의 메뉴바에서 [Help > Install New Software..] 를 선택합니다.

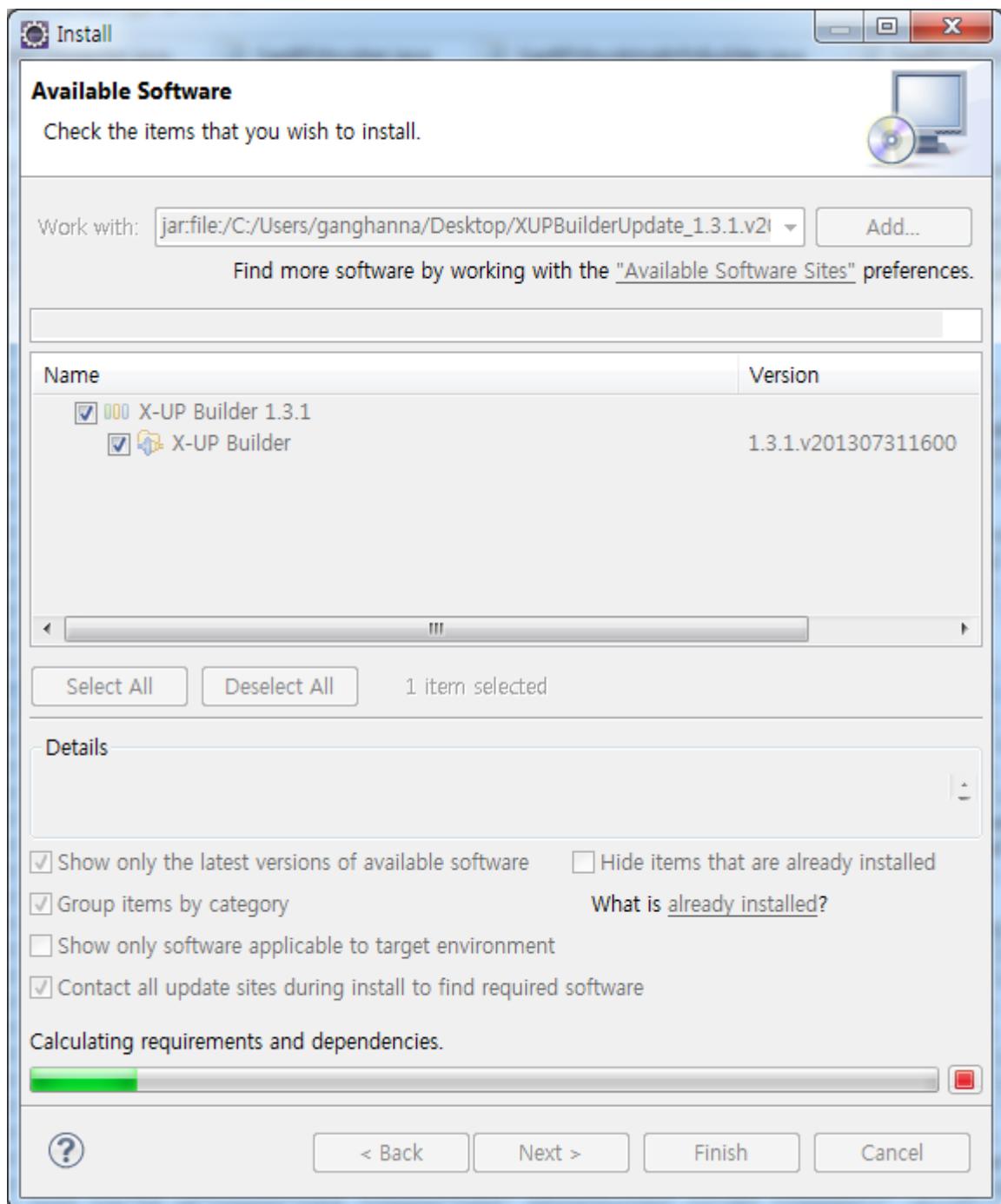


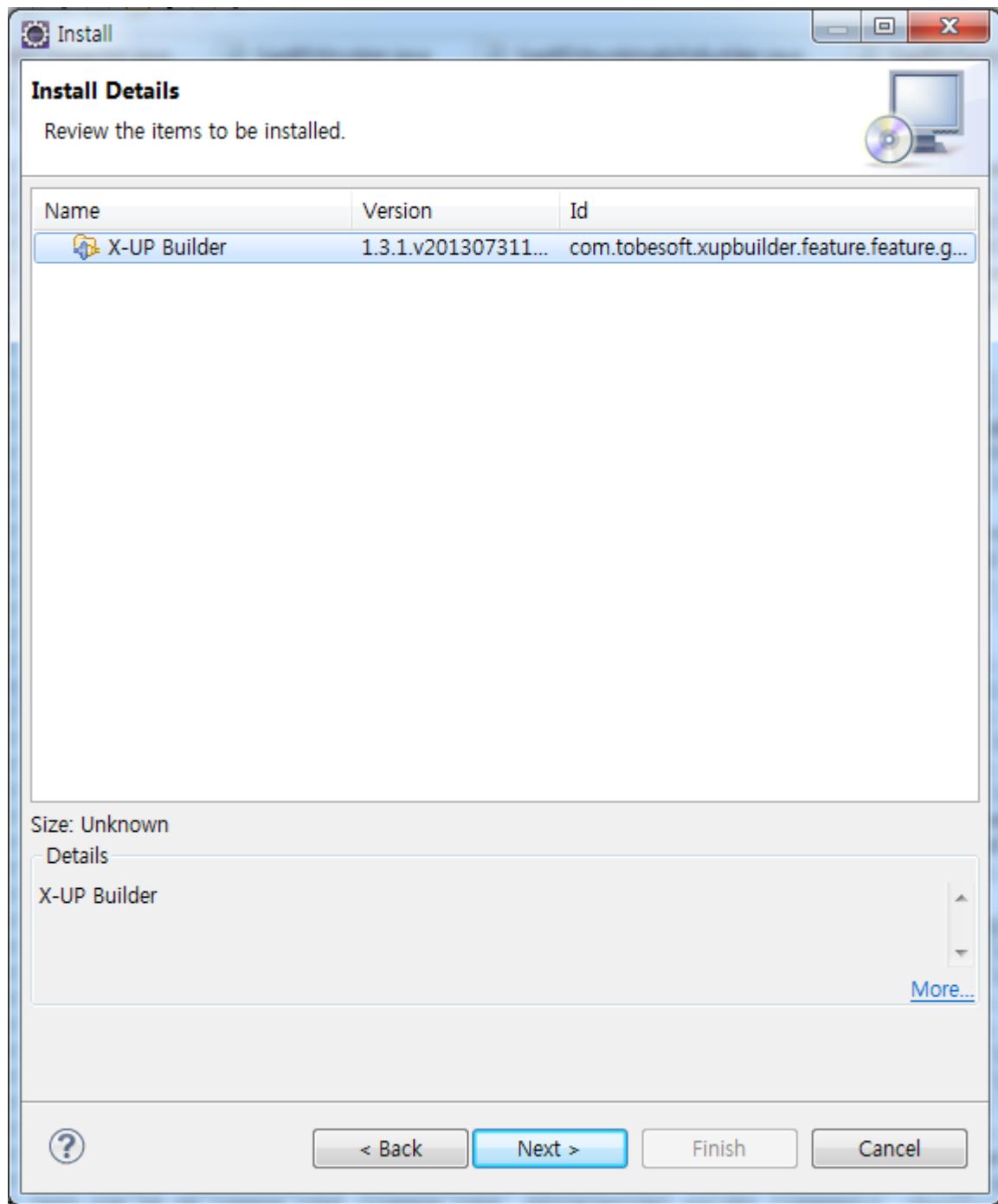
2. Install 창에서 [Add > Archive… > 파일 위치 > OK] 를 선택하여 배포 파일을 Add합니다.



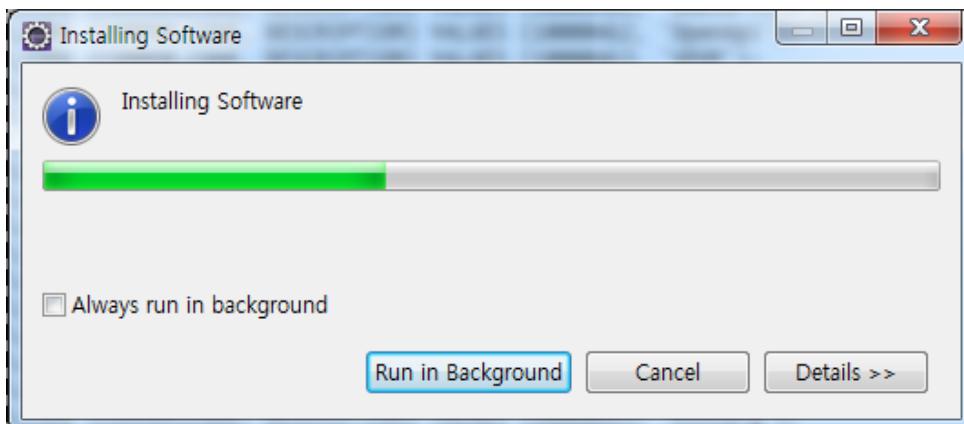
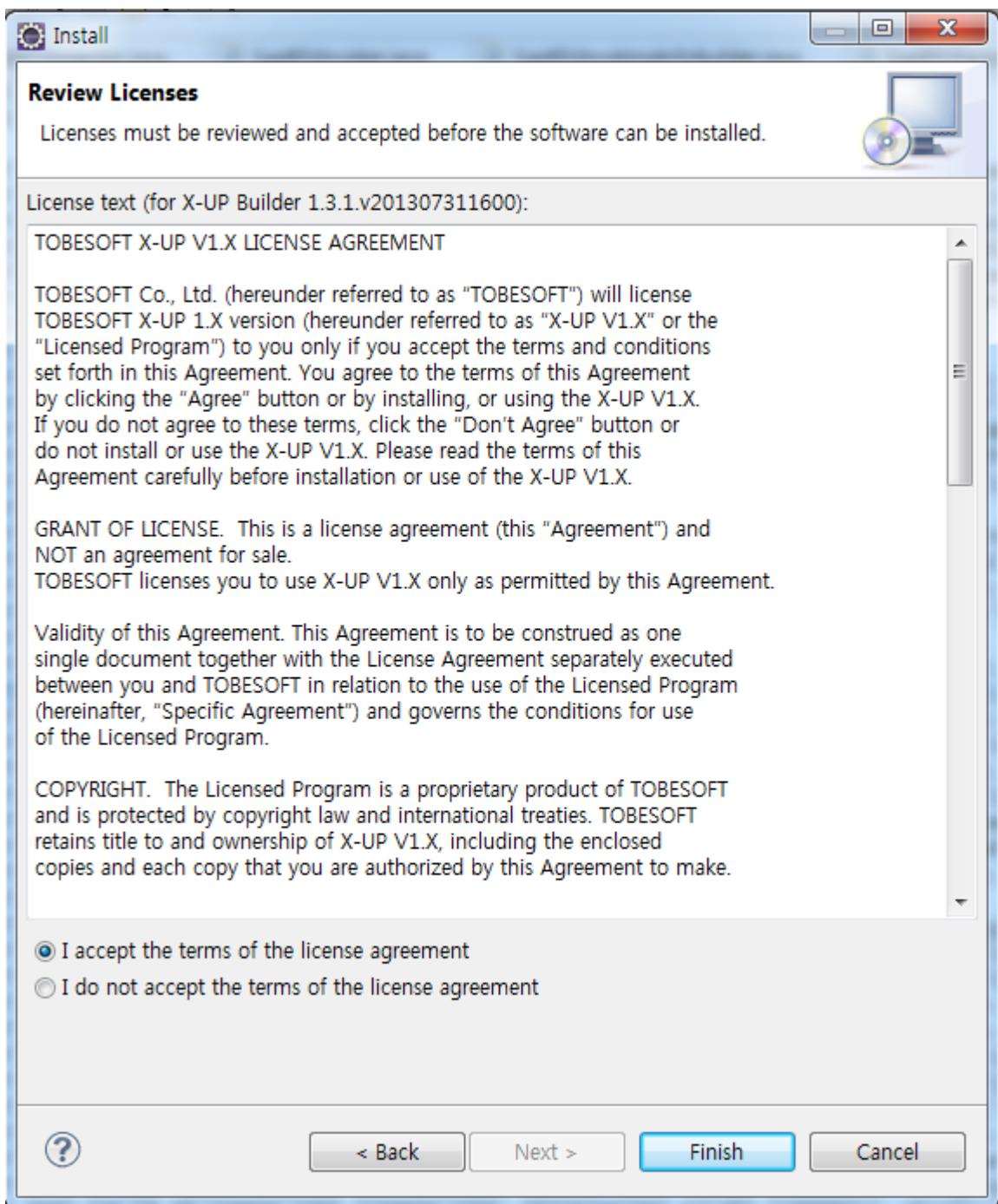


3. Next 버튼을 선택하여 인스톨을 시작합니다. 다시 한번 Next버튼을 선택하여 진행합니다.

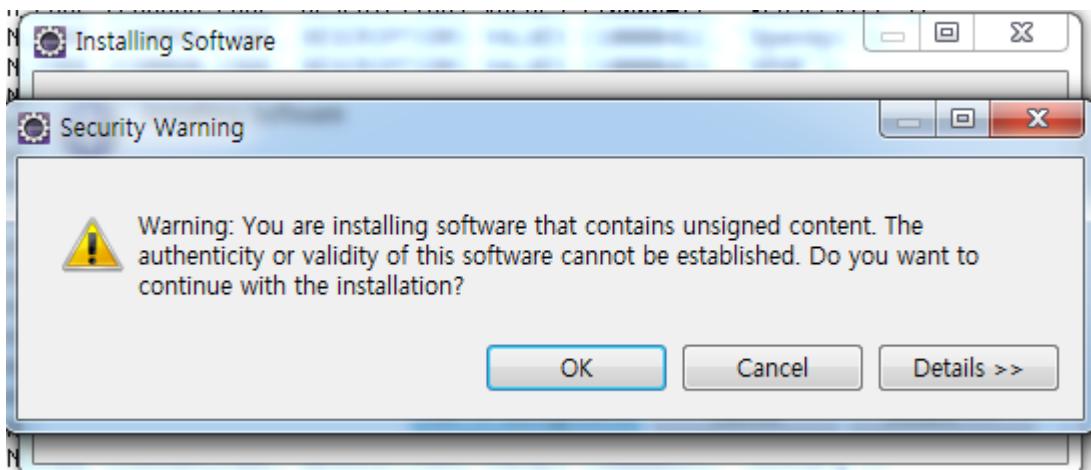




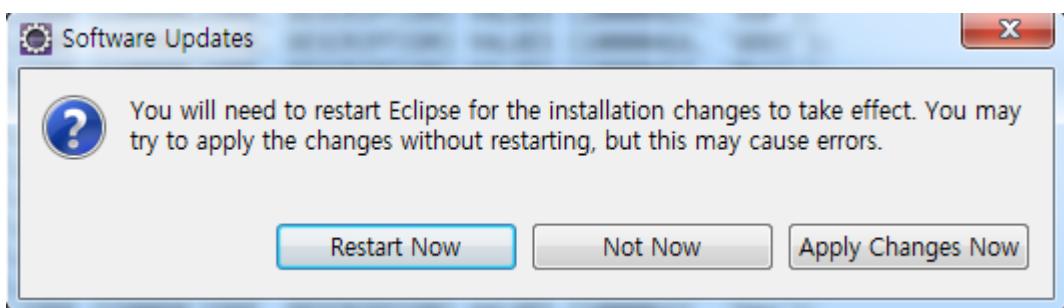
4. 'I accept the terms of the license agreement' 항목을 선택하고 Finish 버튼을 선택합니다.



5. Intalling Software에서 Security Warning 경고 발생시 'OK' 버튼을 선택합니다.

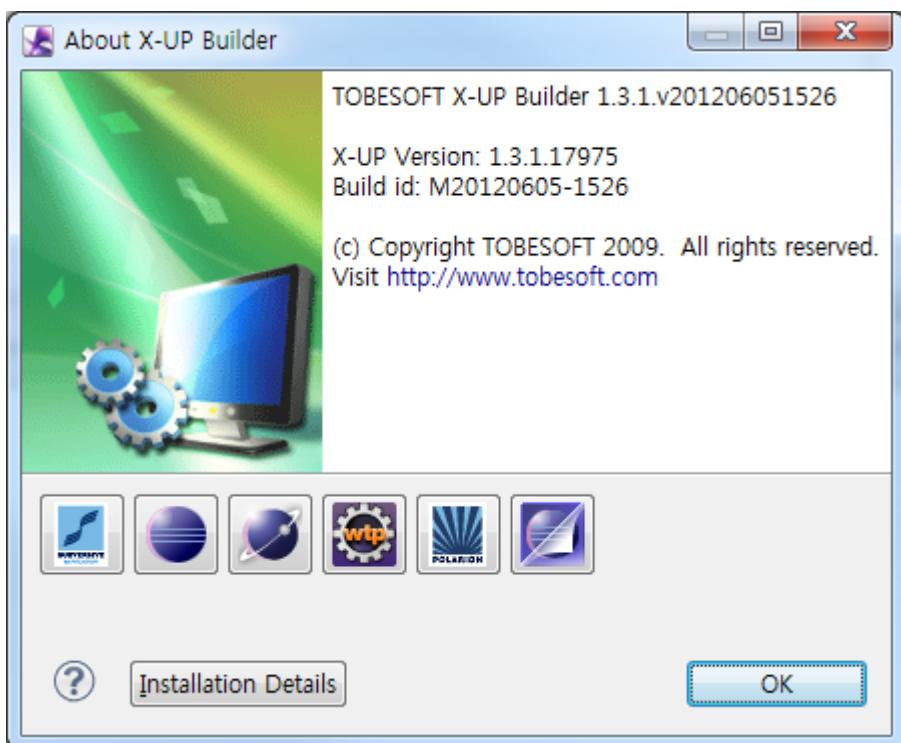


6. Update 후에 Restart Now 버튼을 선택하여 X-UP Builder를 재구동합니다. 기존에 있던 프로젝트 전체를 'Refresh'하여 업데이트를 최종적으로 완료합니다.



About X-UP Bulder

현재 X-UP Builder에 포함되어 사용되는 X-UP Library의 버전과 빌드된 날짜를 확인할 수 있습니다.



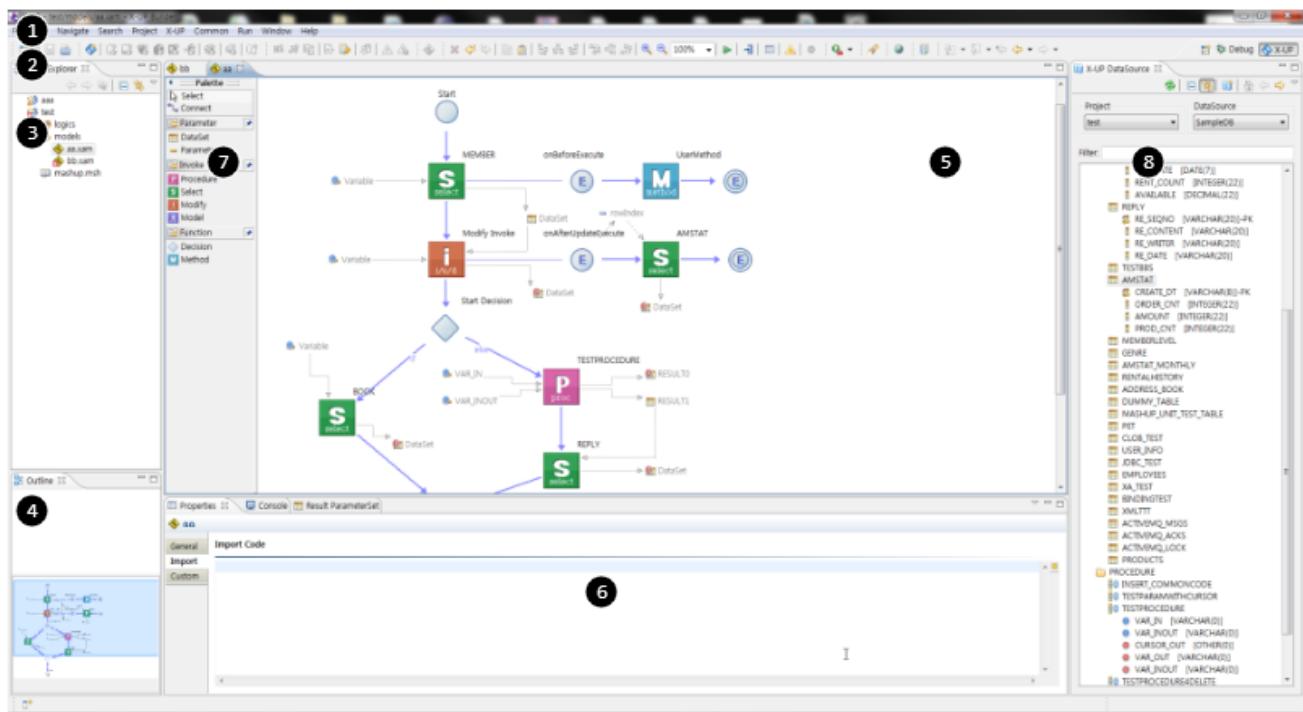
항목	Description
X-UP Version	X-UP Library의 버전
Build id	Build id는 빌드된 날짜 정보를 가집니다. 예를 들어 M20120605-1526 이면 2012년06월05일 15시26분 입니다.

4.

X-UP Builder 구성

이장에서는 X-UP Builder를 구성하는 메뉴 및 기능에 대하여 소개하고 설명합니다.

아래 그림은 X-UP Builder를 구성하는 Layout들에 대한 설명입니다.



영역	구성요소	설명
1	Menu Bar	Menu Bar는 이클립스 기본 메뉴 항목과 X-UP 모델 개발을 위한 메뉴 항목을 제공합니다.
2	개발도구 Bar	개발도구 Bar는 자주 사용되는 메뉴들을 아이콘 형식으로 모아놓은 곳이며, 이클립스 기본 항목과 X-UP 모델 개발을 위한 항목이 제공됩니다.
3	X-UP Explorer	X-UP Explorer는 활성화된 프로젝트의 내용들이 트리형식으로 나오고, X-UP 프로젝트만의 필터링 기능을 제공합니다.
4	Outline View	Outline View는 현재 활성화된 에디터에 연계되어 관련정보를 트리형태로 구성합니다. Domain Editor : 도메인 에디터의 각탭에서 정의된 모든 속성정보를 모아서 간략한 트리 형태로 구성합니다.

영역	구성요소	설명
		X-UP Editor : 모델 에디터에서 정의된 모든 노드들을 트리로 구성하고, 전체 화면의 축소한 Scrollable Thumbnail 화면을 출력합니다.
⑤	Content Pane	Content pane은 선택된 프로젝트 또는 활성화된 파일의 작업공간으로 주로 해당 파일과 연결된 에디터가 위치합니다. X-UP Builder에는 Domain Editor, Gathering Editor, Transaction Editor, Automation Editor 및 Java Editor 가 위치하게 됩니다.
⑥	Editor View	Editor Views는 X-UP Builder에서 제공되는 모든 뷰들이 위치합니다. View Layout은 Top, Bottom 2곳에서 나누어 위치하게 됩니다 모델 에디터와 연결된 뷰들이 위치하게 됩니다. Editor Info Views Function View Result DataSet View Properties View 기타 Eclipse Views
⑦	Editor Palette	Palette에는 모델 에디터에서 모델 개발시 필요한 컴포넌트들의 그룹입니다. 드로잉 기본 메뉴, Output DataSet 및 평선 컴포넌트들로 구성되어있습니다.
⑧	X-UP DataSource View	X-UP DataSource View는 프로젝트에 정의된 모든 데이터소스 정보를 조회하고 출력할 수 있습니다. 또한 모델 개발시 드래그 앤 드랍으로 쉽게 필요한 컴포넌트를 직관적으로 생성할 수 있습니다.

4.1 Menu and 개발도구 Bar

Menu Bar에는 이클립스 기본 메뉴에 X-UP Builder 메뉴가 추가된 형태로 구성되어 있습니다. 이클립스에 관련된 기본 정보는 이클립스 Web Site(<http://www.eclipse.org>)에서 찾아볼수 있습니다.

이클립스에서 자주 사용되는 공통 메뉴는 다음과 같습니다.

Name	Icon	Description
New		새로운 프로젝트 또는 파일을 생성합니다.
Open		기존 프로젝트나 파일을 오픈합니다.
Save		저장합니다.
Print		프린트합니다.
Eclipse Help		도움말을 보여줍니다.

다음 메뉴들은 기본기능외에 X-UP Builder만의 기능이 추가된 메뉴들입니다.

	Paste	Ctrl+V
	Rename	F2
	Delete	Delete
	Refresh	F5

Paste 는 기본 붙여넣기 기능외에 선택된 아이템중 xup model (*mdl or *.trs)파일들이 존재할 경우 다음 기능을 수행합니다.

- 같은 프로젝트에서 붙여넣기 할 경우 : 다른이름으로 모델이름을 변경하여 새로운 모델을 생성하도록 하고, 로직 클래스도 변경된 이름을 참조하여 같이 생성합니다.
- 다른 프로젝트로 붙여넣기 할 경우 : 모델파일 뿐만 아니라 해당 모델의 로직 클래스와 데이터소스 정보도 같이 붙여넣기를 시도합니다. 만약 같은 이름의 모델이 존재할 경우 compare input dialog 를 통해 로직클래스 정보를 보여주고 비교/검토할 수 있게하여 다른이름으로 변경할 수 있도록 합니다. 만약 데이터소스 이름도 같다면 데이터소스 역시 Compare Input Value Dialog를 통해 비교/검토하여 덮어쓸 것인지 무시할 것인지 결정하도록 합니다.

Rename은 기본 기능외에 선택된 아이템이 *.mdl or *.trs 파일일 경우 로직클래스의 이름도 변경될 모델이름을 참 고하여 새롭게 생성합니다. 만약 로직 클래스가 사용자에 의해 수정되었다면 수정된 로직클래스의 내용을 반영하여 생성합니다.(수정된 로직까지 복사합니다.)

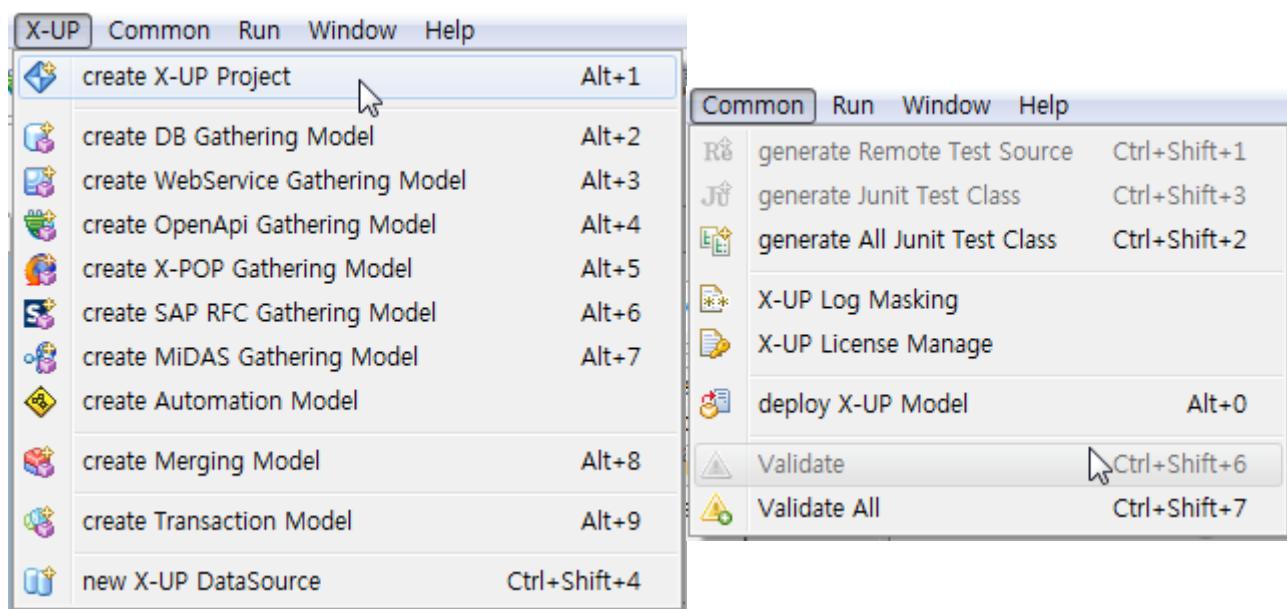
Delete는 기본 기능외에 선택된 아이템이 *.mdl or *.trs 파일일 경우 모델 파일 삭제외에 관련 로직 클래스도 같이 삭제합니다.

Refresh는 기본 기능외에 선택된 프로젝트가 xup project이면 소속된 모든 모델의 버전을 검사하여 동기화를 하고 모델 xml을 rebuild합니다.

X-UP Builder는 X-UP 모델 개발을 위한 다양한 메뉴들을 제공합니다.

X-UP Menu는 X-UP Builder 내 다양한 곳에 위치하며 주로 다음 몇군데에서 찾아볼 수 있습니다.

1. Menu bar에서 File New
2. Menu bar에서 X-UP, Common
3. 개발도구Bar에서 관련 아이콘들
4. X-UP Explorer > Mouse right click > Popup menu > New





Name	Shortcut Key	Description
create X-UP Project	Alt+1	새로운 X-UP 프로젝트를 생성합니다.
create Automation Model		새로운 Automation 모델을 생성 합니다.
create Transaction Model	Alt+9	새로운 Transaction 모델을 생성 합니다.
new X-UP DataSource	Ctrl+Shift+4	새로운 데이터소스 생성 합니다.
generate Remote Test Source	Ctrl+Shift+1	선택한 모델을 기반하여 Remote 테스트를 할 수 있는 테스트 클래스를 생성 합니다.
generate Junit Test Class	Ctrl+Shift+3	선택한 모델을 테스트할 수 있는 간단한 JUnit Test Case를 생성 합니다.
generate All Junit Test Class	Ctrl+Shift+2	선택한 프로젝트에 존재하는 모든 모델을 대상으로 JUnit Test Case를 생성 합니다.
X-UP Log Masking		선택된 프로젝트에 정의되어있는 Server URL들의 로그 암호화를 정의합니다.
X-UP License Manage		선택된 프로젝트에 정의되어있는 Server URL들의 라이센스를 관리합니다.
deploy X-UP Model	Alt+0	모델을 서버에 배치 합니다.
Validate	Ctrl+Shift+6	하나의 모델에 대한 모델인보크 Validate
Validate All	Ctrl+Shift+7	프로젝트에 존재하는 모든 모델에 대한 모델 인보크 Validate 합니다.

4.2 X-UP Explorer

X-UP Explorer는 X-UP Project 리스트를 트리형식으로 출력하고 필터링 기능을 제공합니다.

주요기능 :

- Project 정보 및 모델 리스트 출력
- 선택한 모델 상세 정보 출력 및 수정
- Filtering 제공
- 드래그 앤 드랍으로 모델 인보크 기능 제공
- 모델 더블 클릭시 에디터 오픈

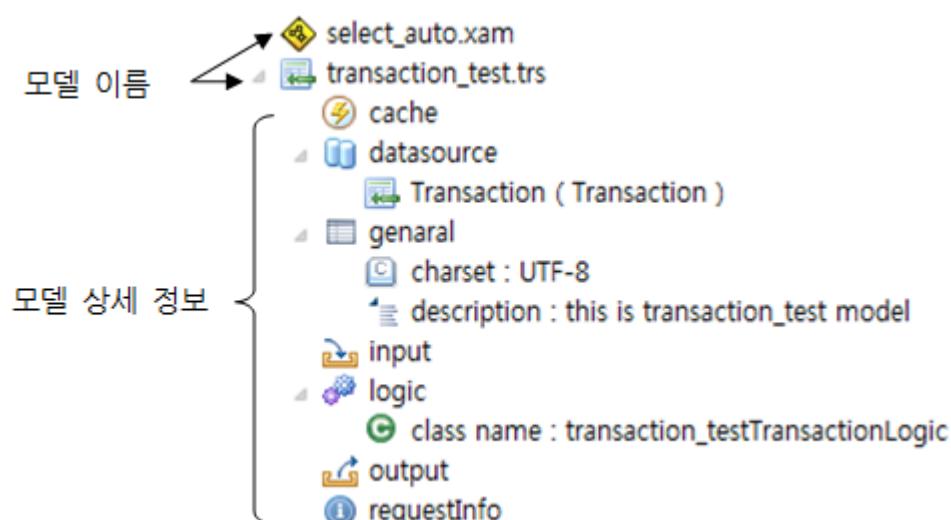
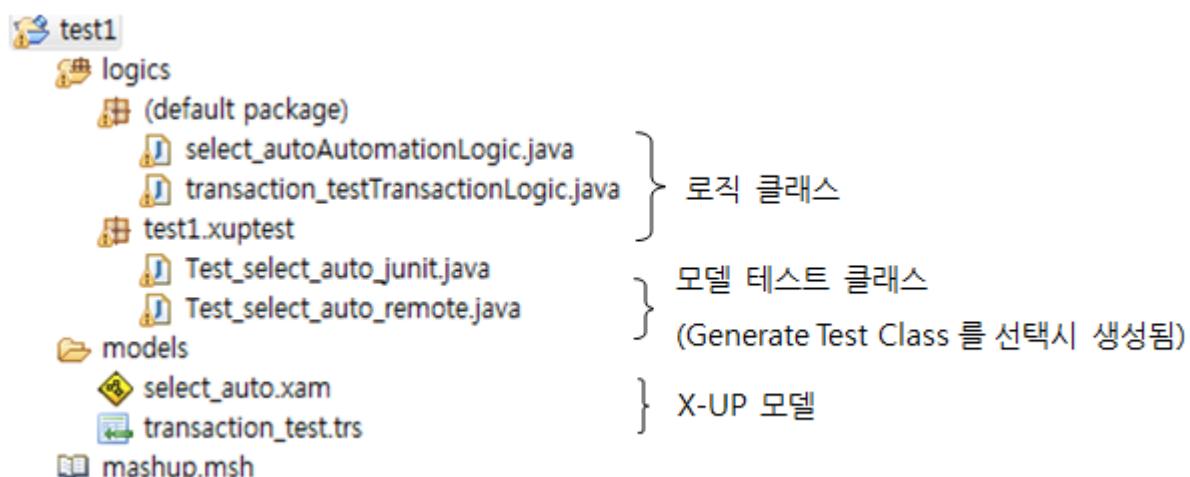
X-UP Project

X-UP Project는 기본 구조는 크게 세가지 형태로 되어있습니다.

- logics : 자바 파일을 위한 src 폴더
- userLib : 추가된 유저 라이브러리를 위한 폴더
- models : X-UP 모델을 위한 models 폴더
- mashup.msh : X-UP 도메인 파일



현재 프로젝트 생성시 존재하는 위의 4가지 폴더 및 파일 이름은 내부적으로 미리 약속된 이름으로 사용중
이므로 다른이름으로 변경하거나 삭제하면 안됩니다



Name	Sub Item	Description	double click action
cache	className	Cache 정보가 정의된 클래스명	Open Edit Cache Dialog

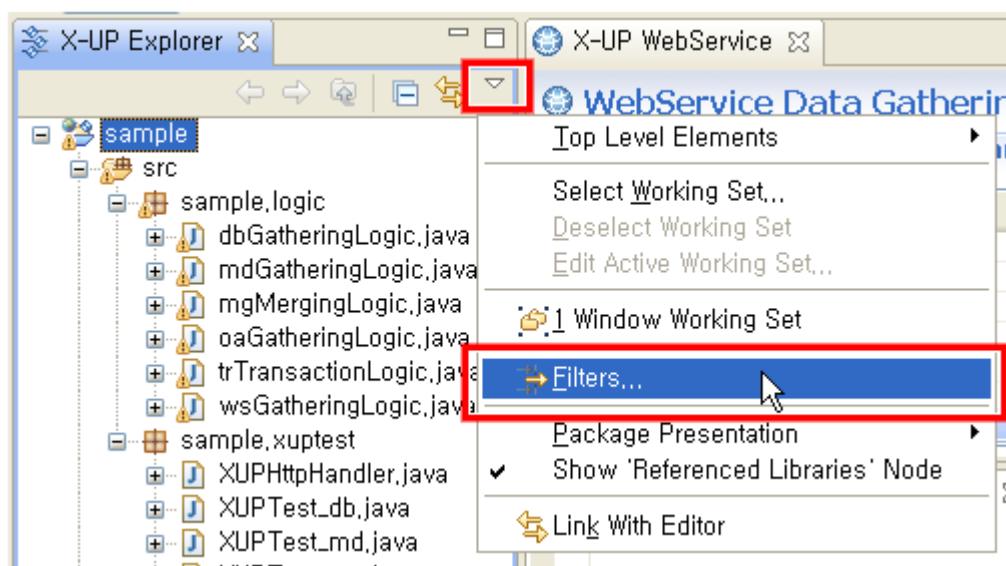
Name	Sub Item	Description	double click action
	isUsed	Cache 사용 유무	Open Edit Cache Dialog
datasource	datasourceName	데이터소스이름과 타입	Open Domain Editor and Open Edit DataSource Dialog
general	charset	문자셋	Open Edit Charset Dialog
	description	설명	Open Edit Description Dialog
input	parameter	input parameter	Open Model Editor
	dataset	input dataset (with column)	Open Model Editor
logic	className	로직 클래스명	Open Logic Class
output	dataset	output dataset (with column)	Open Model Editor
requestInfo	타입별 정보	타입별 정보	Open Model Editor

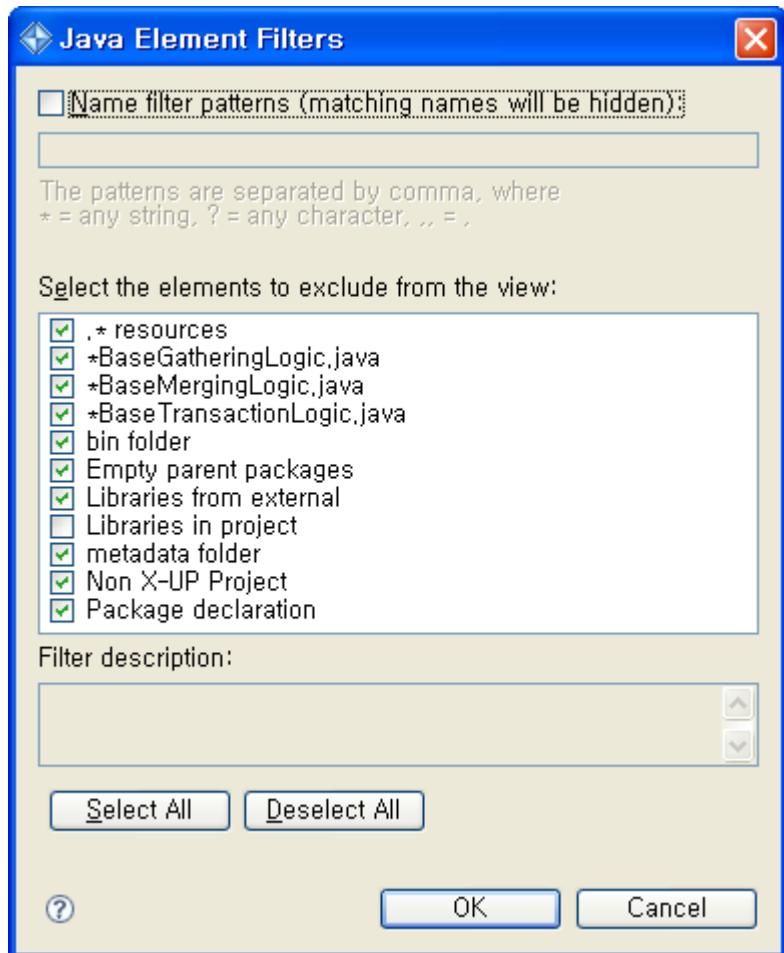


Automation 모델은 Transaction 모델과 달리 정형화되어있는 구조가 아니므로 Automation 모델에는 모델 상세 정보가 제공되지 않습니다.

X-UP Filter

X-UP Project는 일반 사용자를 위해 모델 개발시에는 직접적으로 관련없는 모든 구성요소를 보이지 않게 할 수 있도록 X-UP Explorer에는 필터링 기능이 제공합니다.

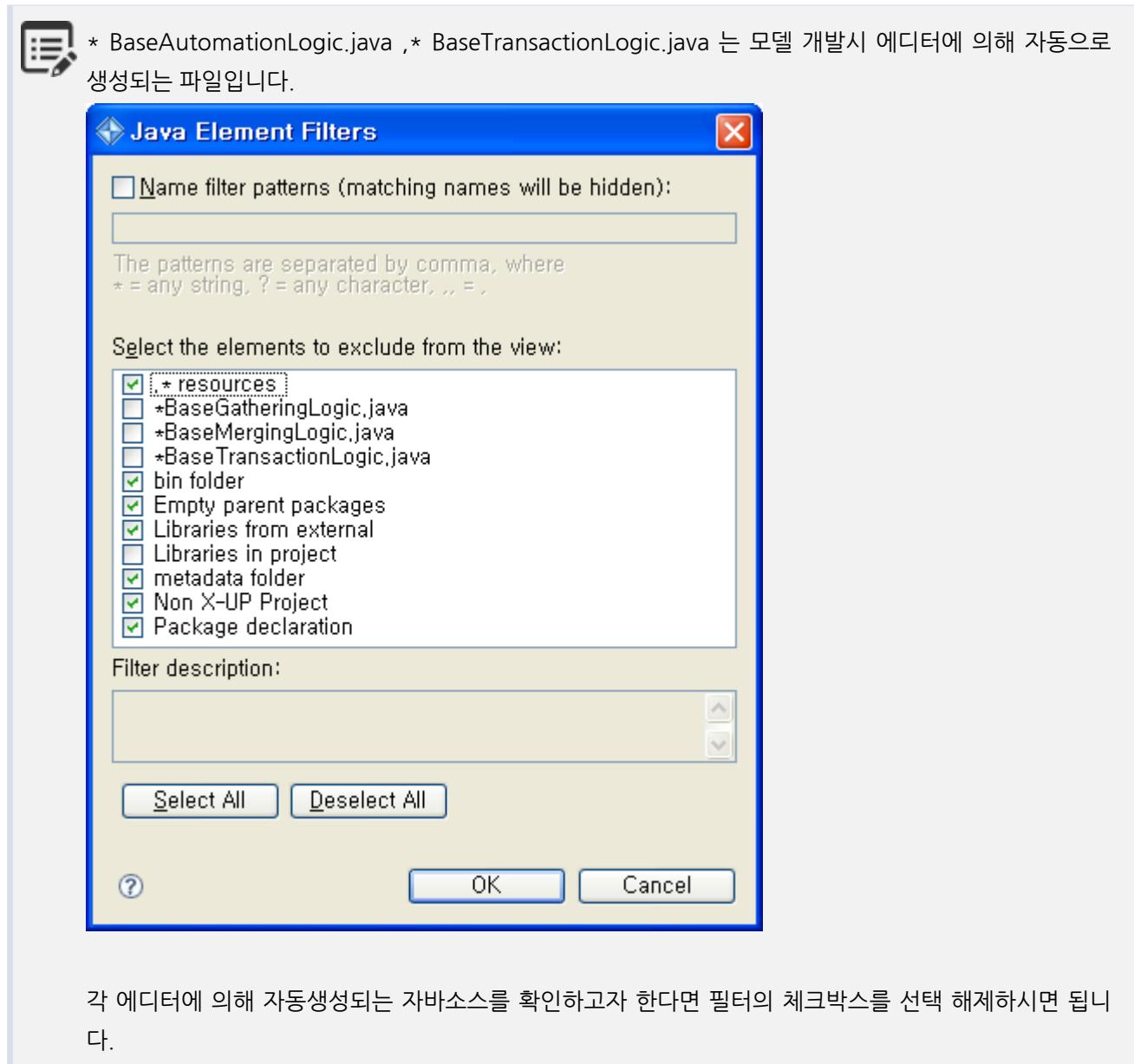




필터링 설정은 X-UP Explorer내에 존재하는 모든 프로젝트에 영향을 미칩니다.

필터링 리스트에서 특정 항목에 체크를 한다는 것은 체크된 항목을 숨기겠다는 의미가 됩니다.

Name	Description
.* resource	Hides resources with names that start with a '.'
*BaseAutomationLogic.java	Hide *BaseAutomationLogic.java
*BaseTransactionLogic.java	Hide *BaseTransactionLogic.java
bin folder	Hide bin folder
Empty parent packages	Hides empty packages which do not contain Java files but other sub-folders
Libraries from external	Hides libraries from external
Libraries in project	Hides local libraries i.e. those contained inside the project itself
metadata folder	Hide *.metadata folder
Non X-UP Project	Hides all projects without XSB nature
Package declaration	Hide package declaration



각 에디터에 의해 자동생성되는 자바소스를 확인하고자 한다면 필터의 체크박스를 선택 해제하시면 됩니다.

4.3 X-UP Wizard

X-UP Builder에서 제공하는 위자드는 크게 다음 4가지입니다.

- New X-UP Project Wizard
- New X-UP DataSource Wizard
- New X-UP Model Wizard
- New X-UP Automation Model Wizard

New X-UP Project Wizard는 X-UP 모델을 개발하고 테스트하기 위한 환경을 구성합니다. X-UP Builder에서 정의되는 Project의 단위는 X-UP에서 domain 단위에 해당합니다. X-UP Model을 개발하기 위해선 반드시 X-UP Project를 먼저 생성해야 합니다.

New X-UP DataSource Wizard는 X-UP Model에서 사용할 데이터소스를 생성합니다. 프로젝트내에 존재하는 데이터소스들은 여러 모델에서 공유될 수 있습니다. 생성된 데이터소스 정보는 mashup.msh에 등록되며 Domain Editor를 통해 추가 / 수정 / 삭제 / 복사등의 작업을 할 수 있습니다.

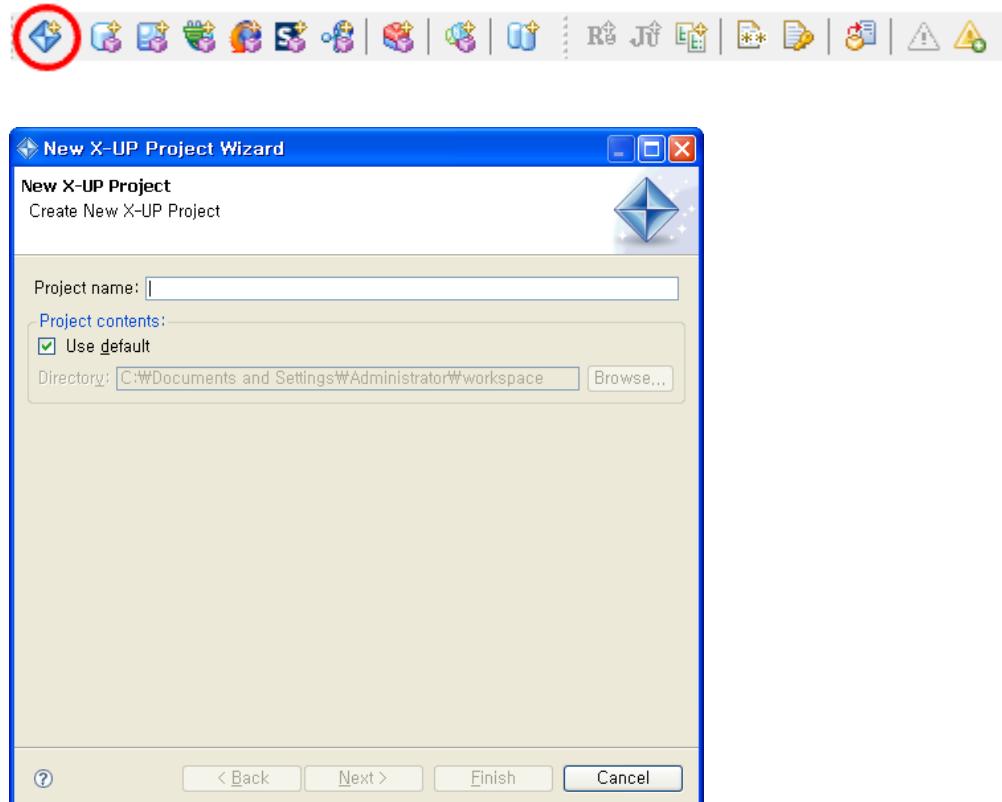
New X-UP Model Wizard는 실제 서비스할 X-UP 모델을 만들기 위한 다양한 소스 파일을 생성합니다. 소스파일은 다음 두가지 확장자로 나누어집니다.

1. *.trs (Transaction 모델을 만들기 위한 모델 소스 파일)
2. *.xam (Automation 모델을 만들기 위한 모델 소스 파일)

New X-UP Automation Model Wizard는 Automation Model을 만들기 위한 위자드입니다. Automation Model은 기존 모델의 단점을 보완하고 기능을 확장했으며 Graphical한 GUI Editor를 통해 Visual Language 형태의 서비스 개발을 가능하게 합니다.

New X-UP Project Wizard

X-UP Project Wizard는 다양한 곳에서 시작할 수 있으나 주로 개발도구Bar의 New X-UP Project 버튼()을 사용합니다.



X-UP Project 위자드는 프로젝트 이름만 입력하는 상당히 간단한 화면으로 구성되어 있습니다.

Finish 가 되면 해당 프로젝트 이름으로 프로젝트가 생성됩니다.



Project name은 다양한 곳에서 사용되기 때문에 숫자로 시작하거나 이름 사이에 띄어쓰기나 특수기호를 지원하지 않습니다. X-UP Builder의 X-UP Project 생성시 도메인 이름은 프로젝트 이름과 동일한 이름으로 자동 결정되고 rename 기능을 통해 변경 가능합니다.

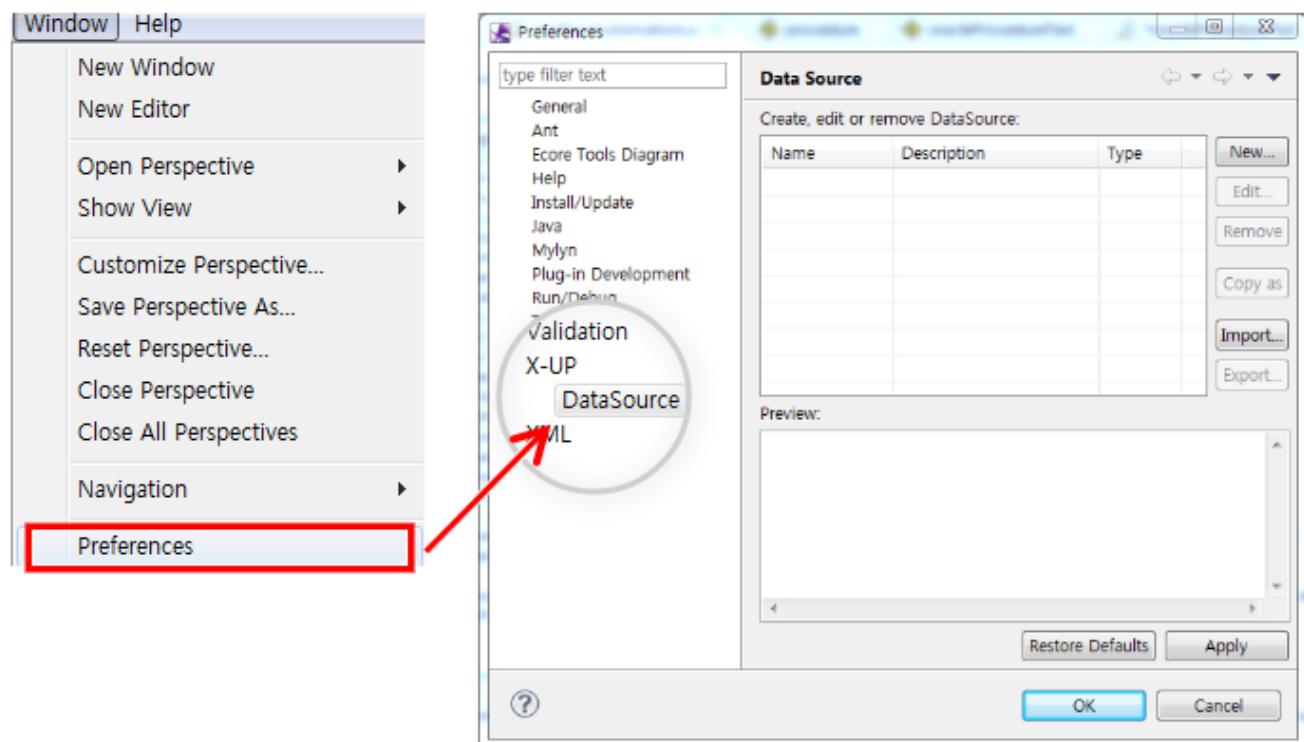
New X-UP DataSource Wizard

New X-UP DataSource Wizard는 선택된 X-UP Project에서 X-UP Model 개발 시 사용될 DataSource를 생성합니다.

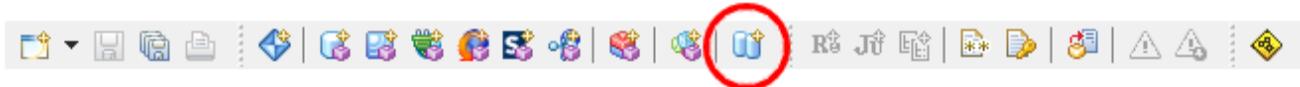


X-UP Builder에서 DataSource를 생성할 수 있는 곳은 다음과 같습니다.

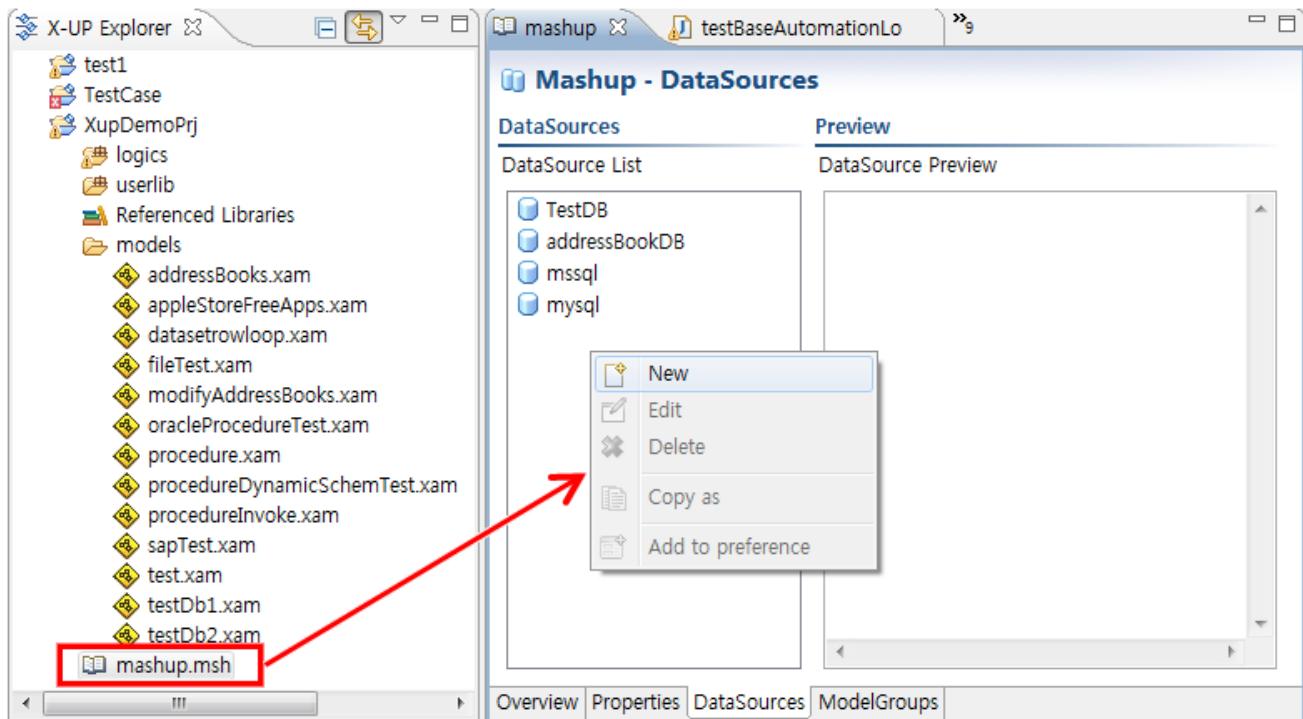
1. Window > Preferences… > X-UP > DataSource > New



2. 개발도구Bar > New DataSource

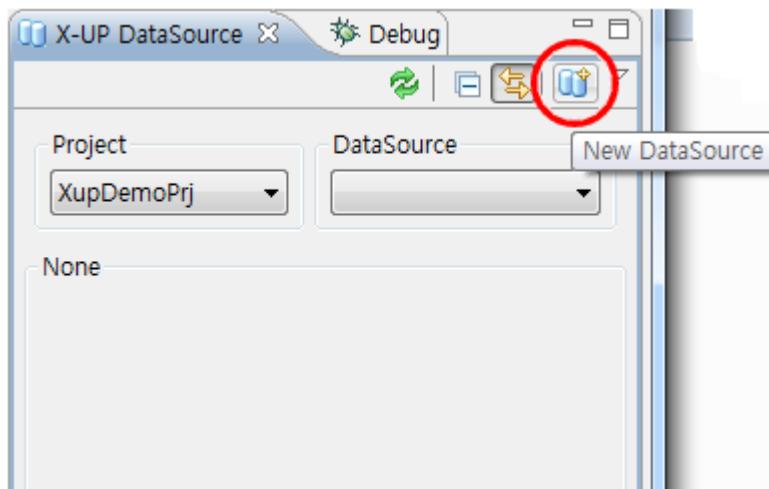


3. mashup.msh > Domain Editor > DataSource tab > DataSource List > New



4. New Model Editor Wizard Datasource Page

5. X-UP DataSource View New DataSource



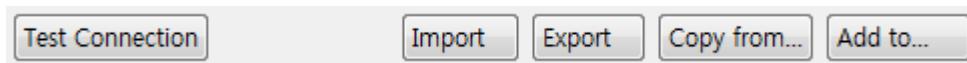
1번 preference 페이지의 데이터소스 리스트에 등록된 데이터소스들은 X-UP Builder 가 보관하게 됩니다. 이는 다음에 데이터소스 생성시 참조용으로 사용될 수 있습니다.

2번부터 5번은 모두 선택된(작업중인) X-UP Project 의 mashup.msh 에 등록되어 실제 X-UP 모델이 데이터를 가져올 때 사용됩니다.

정의할 수 있는 데이터 소스는 다음과 같습니다.

- WebService : 일반 Web Service 데이터소스
- OpenApi : Http Url로 접근하여 가져온 정형화된 데이터를 가공하고자 할 경우
- DB : DataBase 데이터소스
- UDDI : UDDI 데이터소스
- X-UP : (주)투비소프트 사의 X-UP 데이터소스
- SAP RFC : SAP RFC connection

X-UP Builder에서 X-UP 모델을 만들기 위해선 데이터소스 정보가 필요하나 모델을 만들 때마다 데이터소스를 정의해야 한다면 상당히 불편할 것 입니다. 그래서 불필요한 반복작업을 최대한 줄이고 재사용 가능한 형태로 데이터소스를 정의하고 보관할 필요가 있습니다.



X-UP Builder에서 제공하는 모든 DataSource 정의화면에는 다음 기능이 공통으로 있습니다.

- Test Connection : 테스트
- Import : 자신의 PC에 저장된 datasource xml file 가져오기
- Export : datasource 정보를 자신의 PC에 xml file 형태로 저장하기
- Copy from : X-UP Builder Preference에 등록된 datasource 정보 가져오기(복사)
- Add to : datasource 정보를 X-UP Builder Preference에 저장하기

예를 들어 'Copy from'의 경우, X-UP Builder는 환경설정 페이지인 [Window > Preferences... > X-UP > DataSouce]에서 데이터 소스를 정의했거나 또는 Create DataSource Dialog 화면에서 'Add to' 버튼으로 Preference에 등록한 데이터소스가 존재한다면 이후에 같은 데이터소스를 생성할때는 'Copy from' 기능을 이용하여 복사하여 빠르게 생성할 수 있습니다.



X-UP Preference에 정의된 데이터소스 정보들은 참조 및 재사용을 위한 것일뿐 모델 링크되는 개념은 아닙니다. 즉 특정 모델을 생성시 데이터소스 정의 화면에서 Preference에 존재하는 데이터소스를 선택했다 하더라도 Copy 해서 생성하는것이지 Link가 되는 것은 아닙니다.

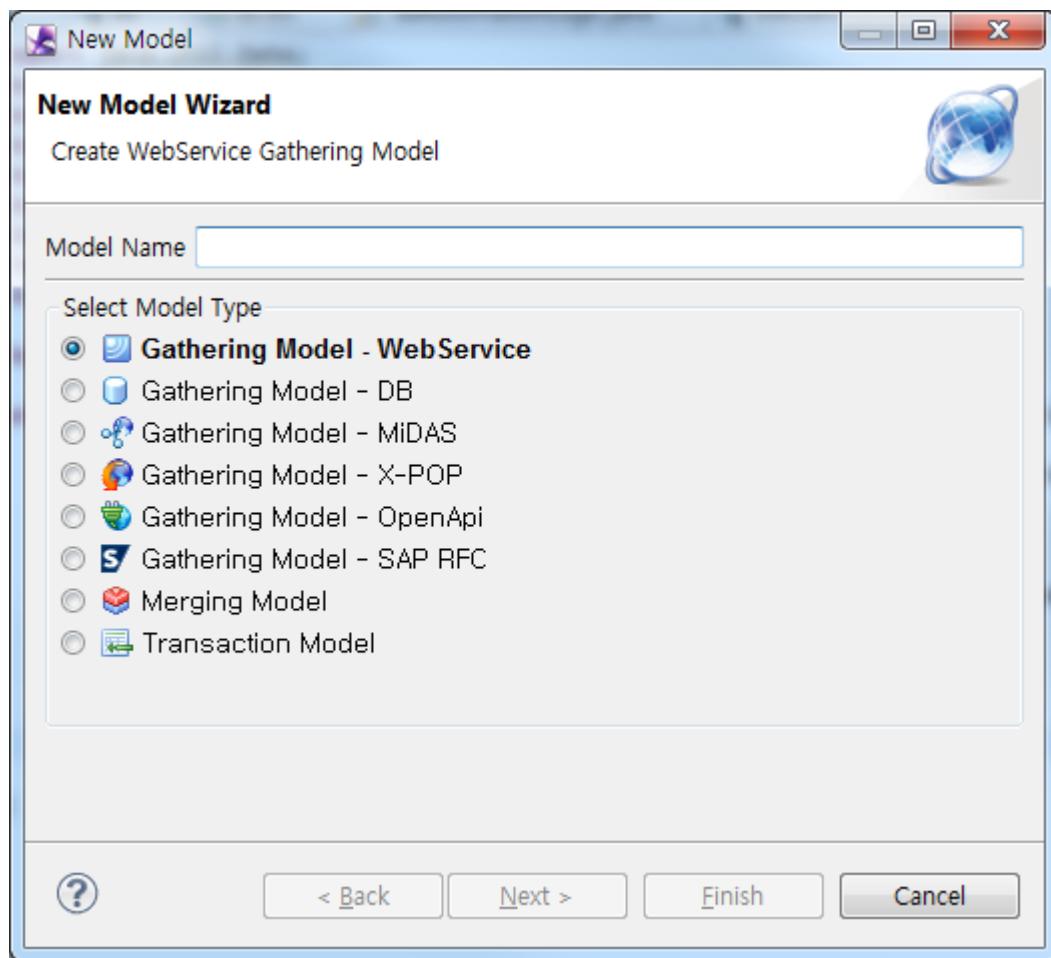
New X-UP Model Wizard

X-UP 모델을 만들기 위한 위자드로서 툴바에서 다양한 타입의 X-UP 모델 버튼을 통해 생성할 수 있습니다.



어떤 버튼을 선택하던지 동일한 모델생성 위자드가 나옵니다. 다만 툴바나 메뉴에서 선택한 모델의 타입이 미리 설정되어 나오게 됩니다. 그래서 만약 처음 생각했던 모델과 다른 모델을 만들고 싶다면 위자드 첫페이지의 모델타입을 변경하기만 하면 됩니다.

위자드의 첫 페이지에서는 모델이름과 모델타입을 결정합니다.



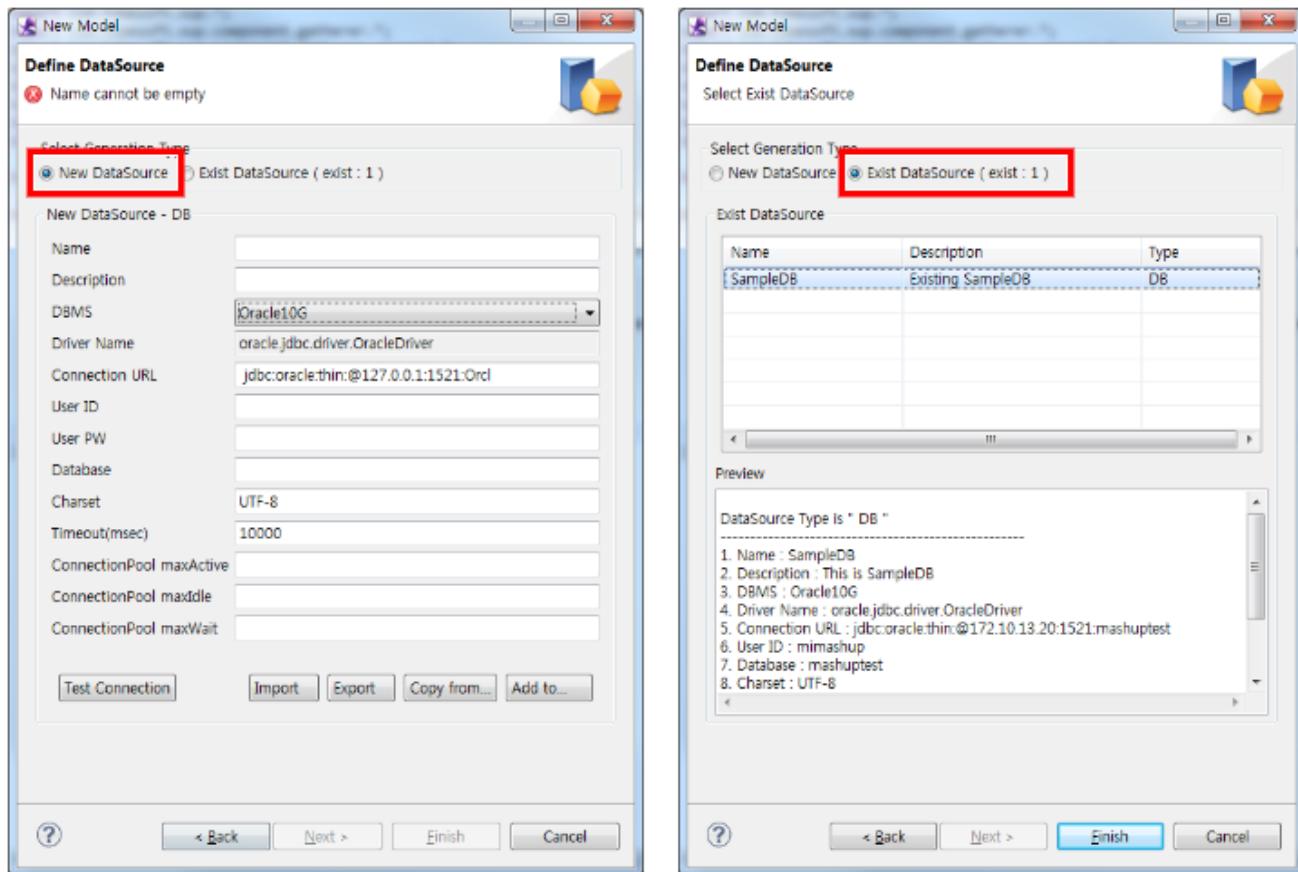
선택 가능한 모델 타입은 다음과 같습니다.

- Transaction Model



Transaction Model 일 경우는 에디터상에서 직접 데이터 소스를 정의합니다.

Define DataSource page는 해당 모델이 사용할 데이터소스를 정의합니다.



Select Generation Type 은 다음 두가지 타입이 있습니다.

- New DataSource : 새로운 데이터소스를 정의
- Exist DataSource : 기존에 정의된 데이터소스 중 하나를 선택

X-UP Builder는 프로젝트내에 동일한 타입의 데이터소스가 존재하지 않는다면 자동으로 New DataSource가 선택되고, 만약 같은 타입의 데이터소스가 사전에 등록된 것이 있다면 자동으로 Exist DataSource 타입이 선택됩니다.

이 자동 선택 기능은 편의성 기능일 뿐 타입의 설정은 사용자가 결정하는 것입니다.



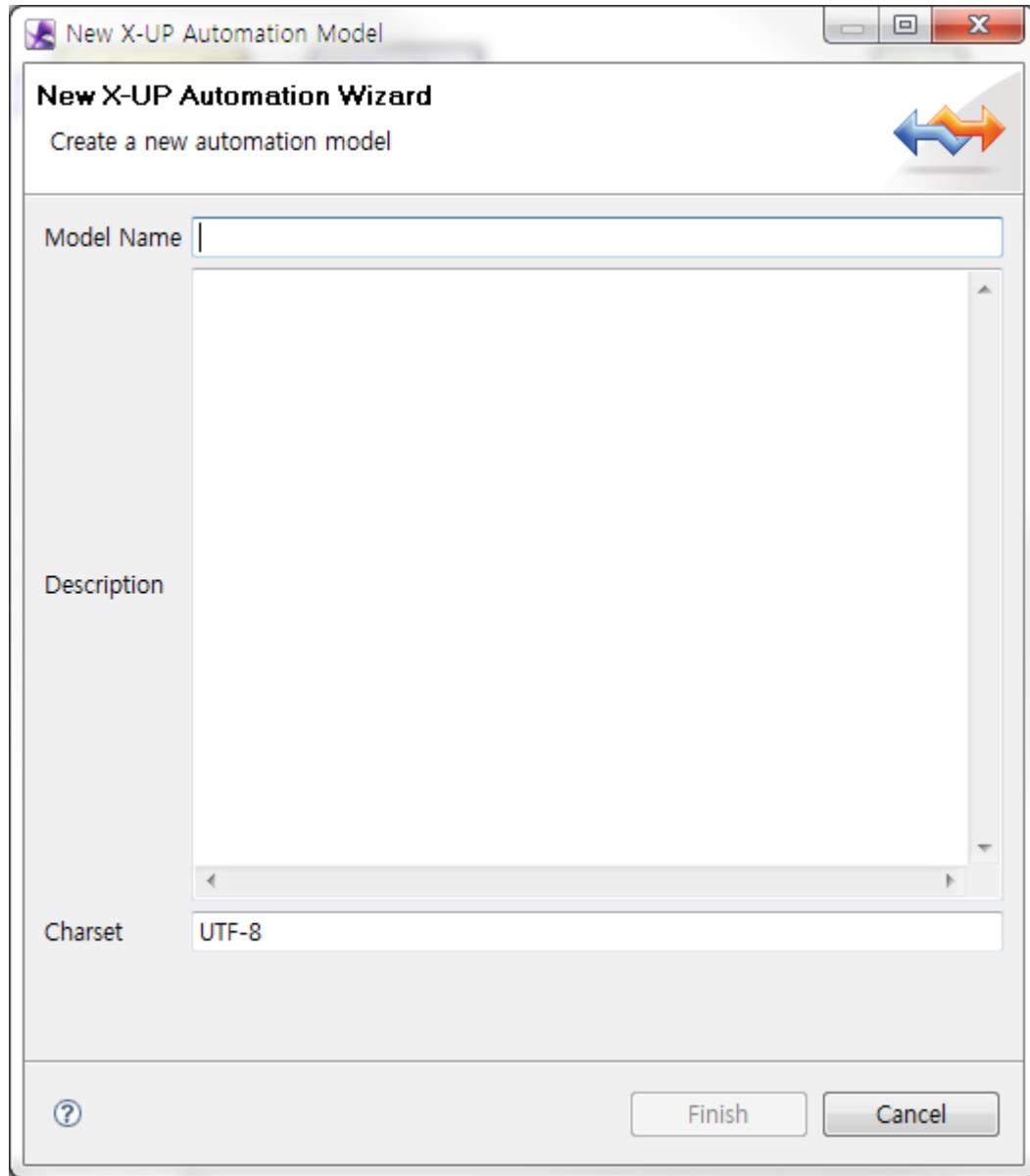
Exist DataSource 라디오 버튼 옆의 exist count 는 선택 모델 타입과 동일한 모델이 이미 하나이상 있을 경우 그 모델들이 사용한 데이터소스의 개수가 표시됩니다.

New X-UP Automation Model Wizard

New X-UP Automation Wizard는 Automation Model을 만들기 위해 새롭게 추가된 위자드입니다.



Automation Model은 기존의 합니다. Transaction Model에서 부족했던 부분을 보완하고 통합하였으며 Graphical 한 GUI 에디터를 제공하여 보다 Visual한 Language로 비즈니스 서비스를 개발할 수 있습니다.



Model Name 은 Automation 모델의 이름을 입력합니다. 프로젝트내에 중복되지 않는 이름을 등록하면 되며, 영어 이외의 다른 문자열은 허용되지 않습니다.

Description 은 모델의 간략한 설명을 입력합니다.

Charset은 문자열 인코딩 타입을 정의합니다.

4.4 X-UP Editor

Domain Editor

도메인 에디터는 **mashup.msh** 파일을 더블클릭하면 나오는 에디터로서 사용자명, 프로젝트 설명, 디플로이 경로, 데이터소스 정의, 모델그룹 정의 등을 합니다.

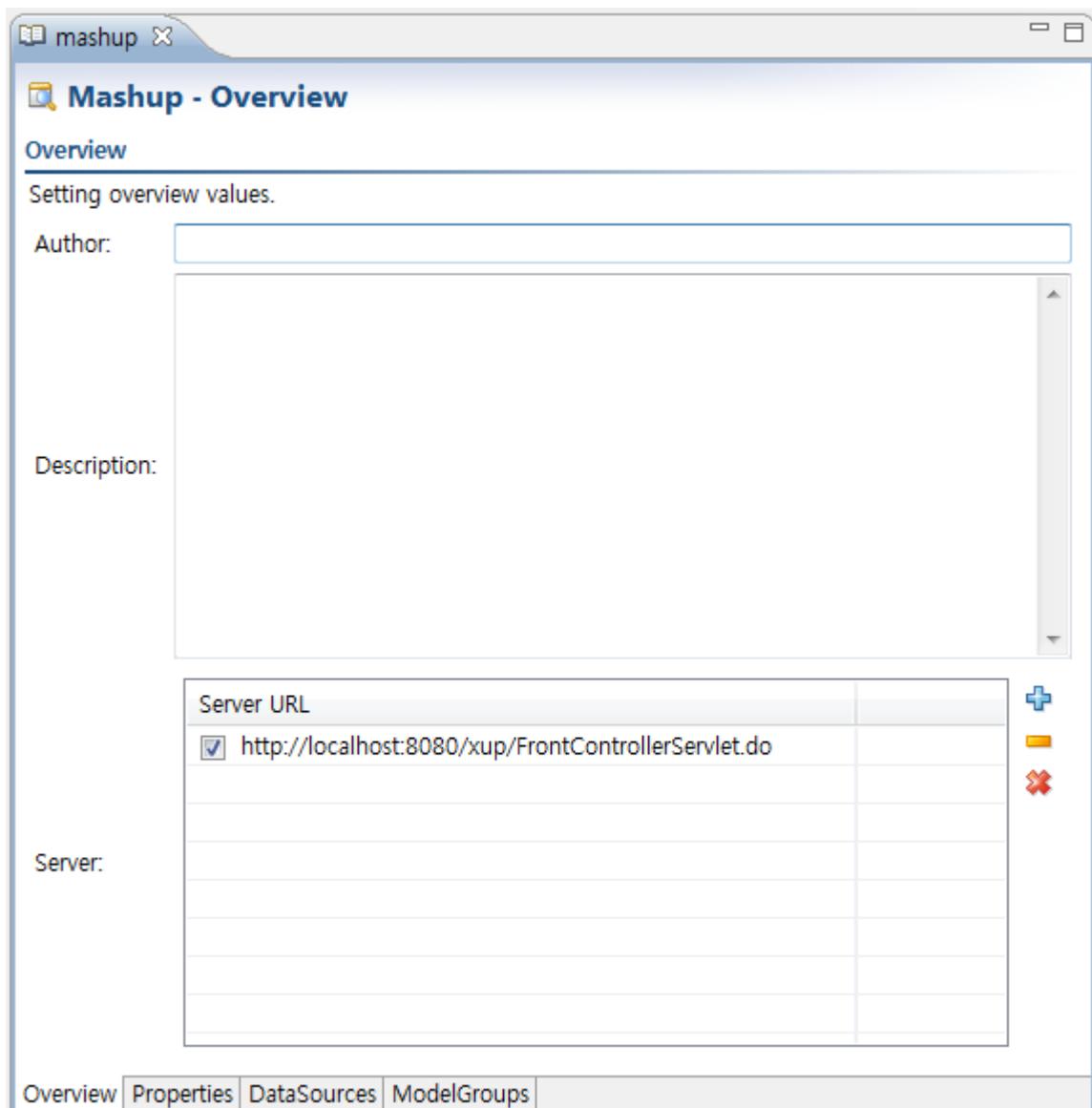
도메인 에디터는 4개의 탭으로 구성되어 있습니다.

- Overview
- Properties
- DataSource
- ModelGroups

Name	Description
Overview	Overview에서는 도메인의 간략한 설명과 디플로이 패스를 정의합니다. Author: 사용자 이름 Description : 설명 Server : 디플로이 패스 정보 하나이상의 서버에 동시에 디플로이가 가능하며 체크박스에 체크된것만 디플로이를 시도합니다.
Properties	프로퍼티들을 등록합니다.
DataSource	데이터소스들을 등록합니다. New : 새로운 데이터소스를 등록 Edit : 선택한 데이터소스를 수정 Delete : 선택한 데이터소스를 삭제 Copy as : 선택한 데이터소스를 복사 Add to preference : 선택한 데이터소스를 Preference 의 X-UP DataSource 에 등록
ModelGroups	모델그룹을 정의합니다.

Overview

Overview 페이지는 작성자명과 프로젝트 설명, 디플로이 패스를 등록합니다.

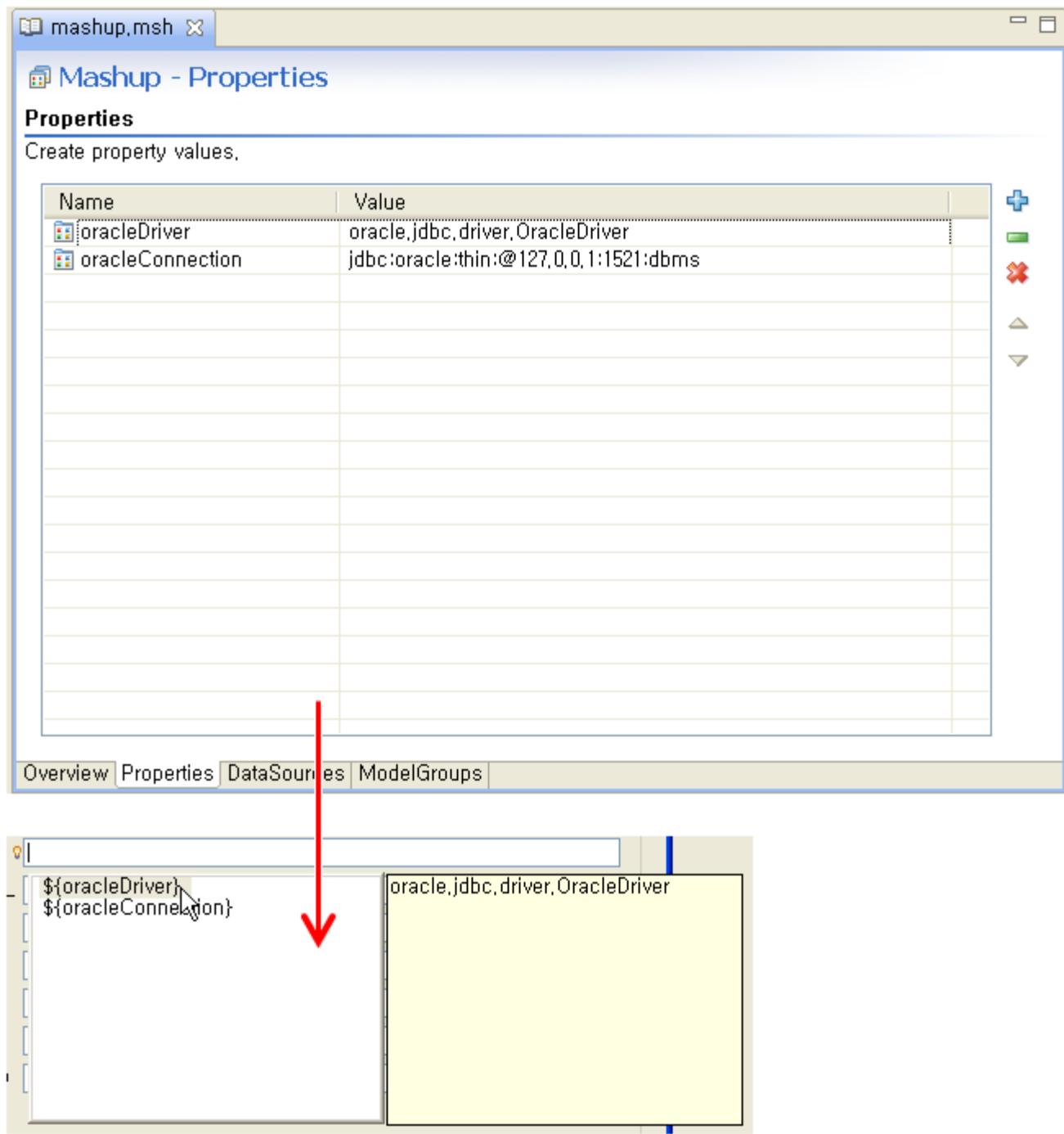


Name	Description
Author	사용자 이름 (선택)
Description	설명 (선택)
Server	<p>개발된 모델들을 서버에 디플로이 시킬 때 사용될 URL 리스트입니다.</p> <p>하나 이상의 URL에 한꺼번에 디플로이 시킬 수 있고, 리스트에는 존재하나 디플로이 되기를 원치 않는 URL은 체크박스를 해지 시켜 제외시킬 수 있습니다.</p>

Properties

X-UP 데이터소스를 생성할 때 자주 사용될 수 있는 정보를 프로퍼티에 등록하여 재사용할 수 있습니다. 등록된 값은 데이터소스 등록화면에서 Ctrl+Space 를 통해 Assist popup 형태로 지원됩니다.

등록된 프로퍼티들이 데이터소스의 입력란에 지원되는 모습은 다음과 같습니다.

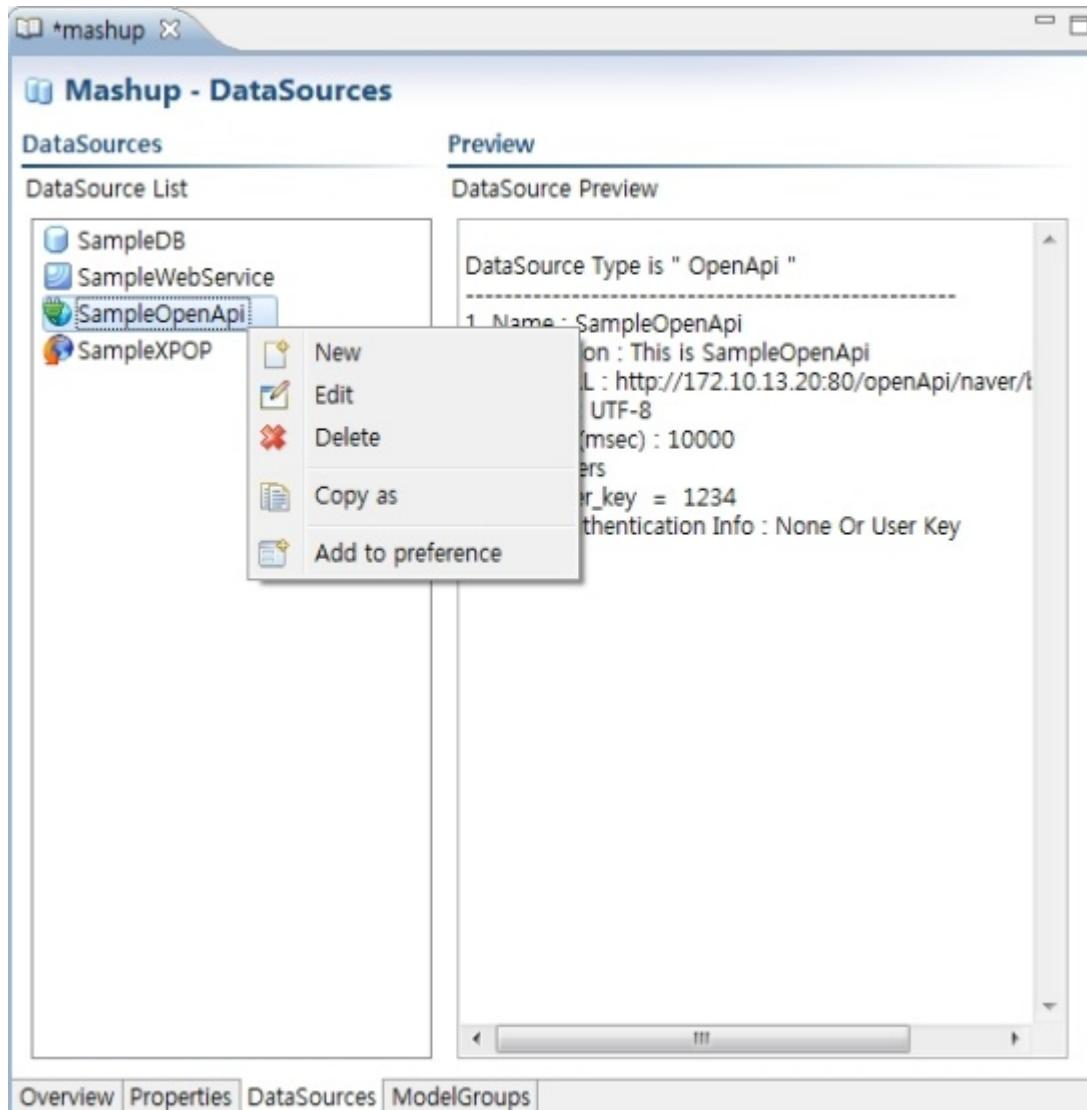


먼저 프로퍼티를 등록합니다.

데이터소스 입력란 앞에 아이콘()이 나타나는 필드는 CTRL+SPACE 를 누를 때 위의 Code Assist Pop-up 을 지원한다는 표시이고, 출력되는 리스트는 앞서 Properties 에 등록된 정보들이 나타나게 됩니다.

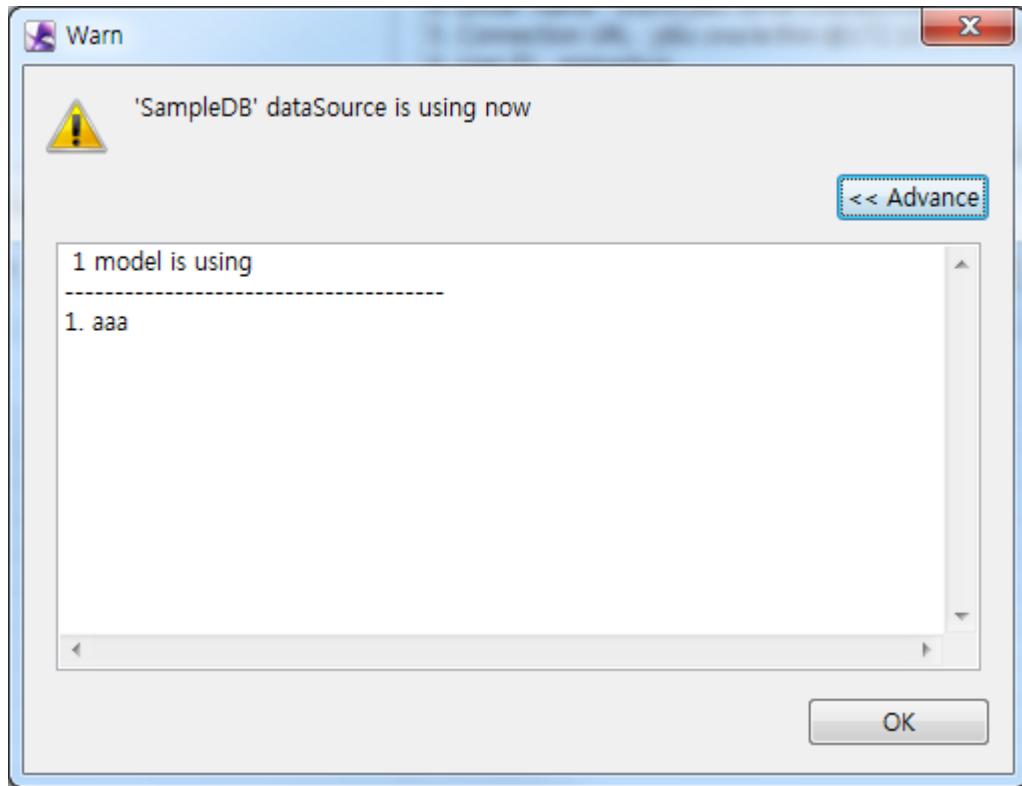
DataSource

DataSource 페이지에서는 X-UP 모델 개발시 사용되는 데이터 소스를 등록, 수정, 삭제, 복사등의 일을 할 수 있습니다.



Name	Description
New	새로운 데이터소스를 생성합니다.
Edit	선택한 데이터소스를 수정합니다. (이름은 수정불가)
Delete	선택한 데이터소스를 삭제합니다.
Copy as	선택한 데이터소스를 복사하여 다른 이름의 데이터 소스를 생성합니다.
Add to preference	선택한 데이터소스를 preference 의 X-UP DataSource 에 등록합니다.

사용중인 데이터소스를 삭제할려고 하면 다음 경고 메시지가 출력됩니다.



만약 모델에서 사용중인 데이터소스를 삭제하고자 한다면, 먼저 모델에서 다른 데이터소스로 변경한 후 삭제해야 합니다.

Transaction Model Editor

Transaction Model은 X-UP 모델 중 하나로서 데이터베이스와 관련된 트랜잭션 로직을 처리하기 위한 모델입니다. 트랜잭션 에디터에서는 해당 모델을 만들기 위해 필요한 기본적인 input / output paramter 정보를 쉽게 정의할 수 있도록 합니다.

- **Input** : 트랜잭션 모델이 호출될 때 필요한 파라미터 정보들로서 하나이상의 파라미터와 데이터셋 스키마들로 구성됩니다.
- **Output** : 트랜잭션 모델이 호출되면 결과값으로 리턴할 하나이상의 데이터셋 스키마 정보들로 구성됩니다.

Input

Input 탭은 모델의 input parameter 를 정의합니다.

The screenshot shows the 'Input Parameter' configuration screen in X-UP Builder. The 'Parameters' section displays a table of input parameters:

Name	Type	Size
ticket	dataset	
changeset	string	255
milestone	string	255
wiki	string	255
build	string	255
max	string	255
daysback	string	255
para7	string	255
ds	dataset	

A red circle highlights the 'dataset' entry in the 'Type' column for the 'ds' parameter. A red arrow points from this row down to the 'Columns' section.

The 'Columns' section displays a table of dataset schema columns:

Name	Type	Size	IsConst
S name0	string	255	<input type="checkbox"/>
S name1	string	255	<input type="checkbox"/>
S name2	string	255	<input type="checkbox"/>
S name3	string	255	<input type="checkbox"/>

At the bottom of the interface, there are tabs for 'Input' and 'Output', with 'Input' currently selected.

Input파라메터를 정의하는 과정은 일반적으로 다음과 같습니다.

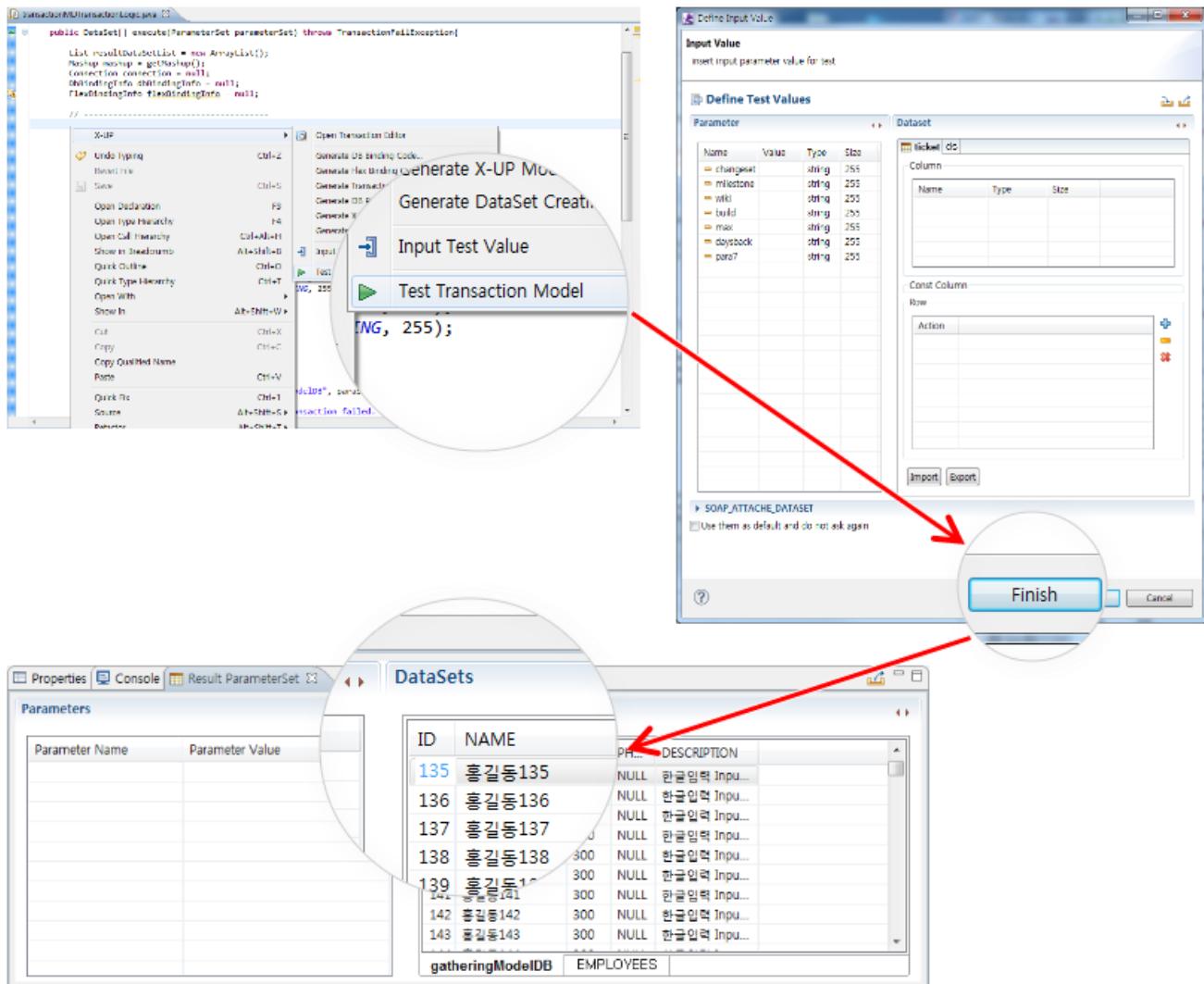
1. 새로운 파라메터를 추가합니다.
2. 해당 파라메터의 타입을 정의합니다. 파라메터의 타입은 XPLATFORM Data Type과 동일한 타입을 사용합니다.
3. 만약 파라메터 타입이 dataset 일 경우는 하위에 Columns 정보를 추가 정의합니다.

컬럼 정보는 직접 입력하거나 Import Schema Dialog 를 통해 입력합니다.

Output

output page는 트랜잭션 모델의 실행 결과인 output dataset schema를 정의합니다.

일반적으로 직접 정의하기 보단 트랜잭션 자바 파일에서 로직을 정의하고 Test Transaction Model 을 하여 나온 결과 데이터셋들을 자동으로 추가합니다.



Generate Transaction Logic

X-UP Code Generate Popup Wizard 는 트랜잭션 모델의 로직 클래스를 작성할 때 사용하며, 위자드 형태의 마법사를 통해 개발자가 직접 소스코드를 작성하지 않고도 쉽게 서비스 로직을 작성할 수 있도록 도와줍니다.

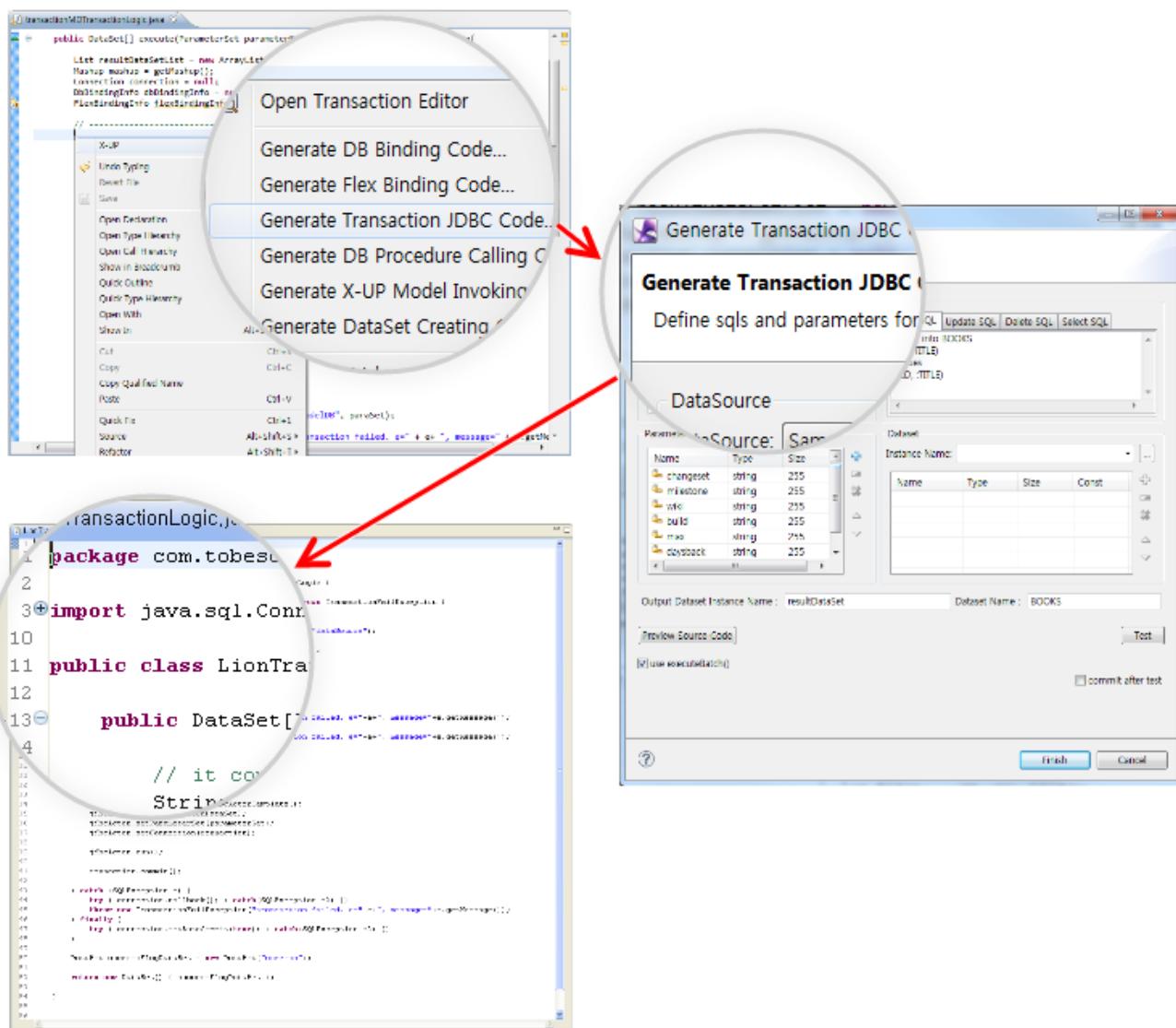
X-UP Code Generate 는 다음의 기능을 제공합니다.

- Generate DB Binding Code
- Generate Transaction JDBC Code
- Generate Procedure Calling Code
- Generate X-UP Model Invoking Code
- Generate DataSet Creating Code

X-UP Code Generate 메뉴는 X-UP 트랜잭션 로직 클래스에서 오른쪽 마우스 메뉴의 첫번째 항목에 나오며 생성하고자 하는 서브메뉴를 선택 후 위자드에 알맞은 항목을 입력하고 종료하면 자동으로 해당 소스코드를 위자드로 열었던 커서위치에 생성됩니다.

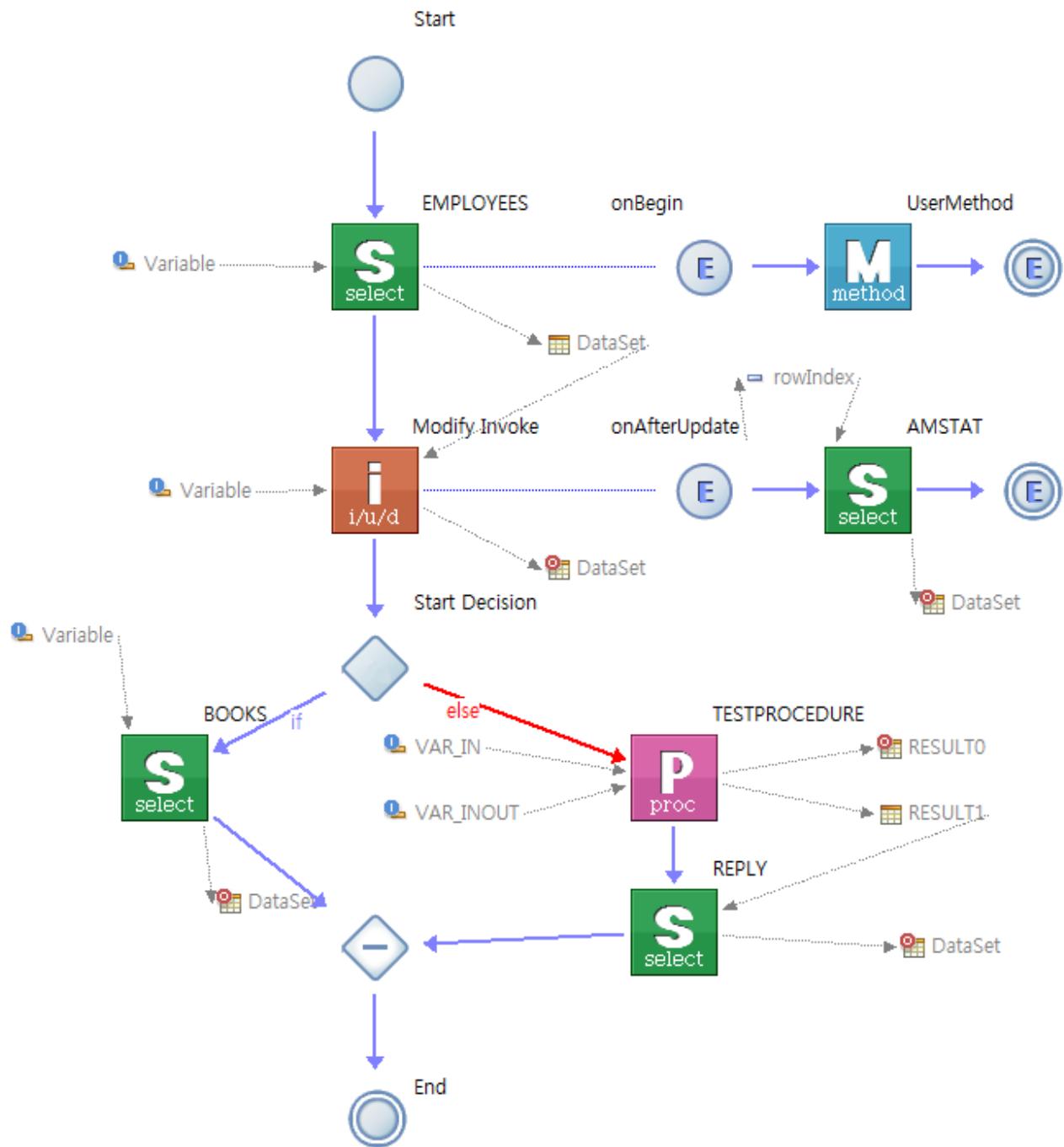
트랜잭션 로직 개발 흐름은 다음과 같습니다.

1. Generate Code Menu 선택
2. 위자드 실행 및 완료
3. 소스코드 자동생성



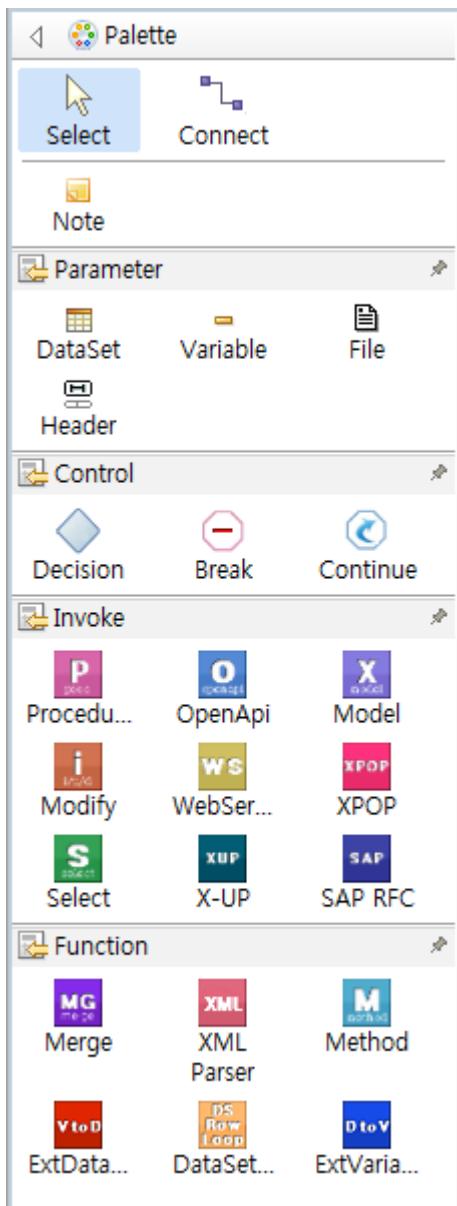
Automation Model Editor

X-UP Automation Model(XAM)은 Graphical한 비즈니스 서비스 개발이 가능한 X-UP 모델입니다. XAM은 Gathering Model과 Transaction Model의 한계를 보완하고 직관적이며 사용자 소스 코딩을 최소화 하고 드래그 앤 드롭만으로 쉽게 서비스 개발이 가능합니다.



Palette

Palette는 XAM Editor 에디터에서 사용가능한 기본적인 도구들을 가지고 있습니다.



현재 제공되는 항목은 다음과 같습니다.

- Common : Select / Connect / Note
- Parameter : DataSet / Variable / File / Header
- Control : Decision / Break / Continue
- Invoke : Procedure / OepnApi / Model / Modify / WebService / XPOP / Select / X-UP / SAP RFC
- Function : Merge / XML Parser / Method(User Method) / ExtDataSet / DataSetRowLoop / ExtVariable

Select는 디폴트로 선택되어지는 항목으로써 에디터의 노드들을 선택하거나 이동시킬 수 있습니다. Connect모드에서 Select모드로 변경하기위해선 직접 Palett에서 Select를 선택하거나 에디터에서 바탕화면을 클릭하면 자동으로 Select모드로 변경됩니다.

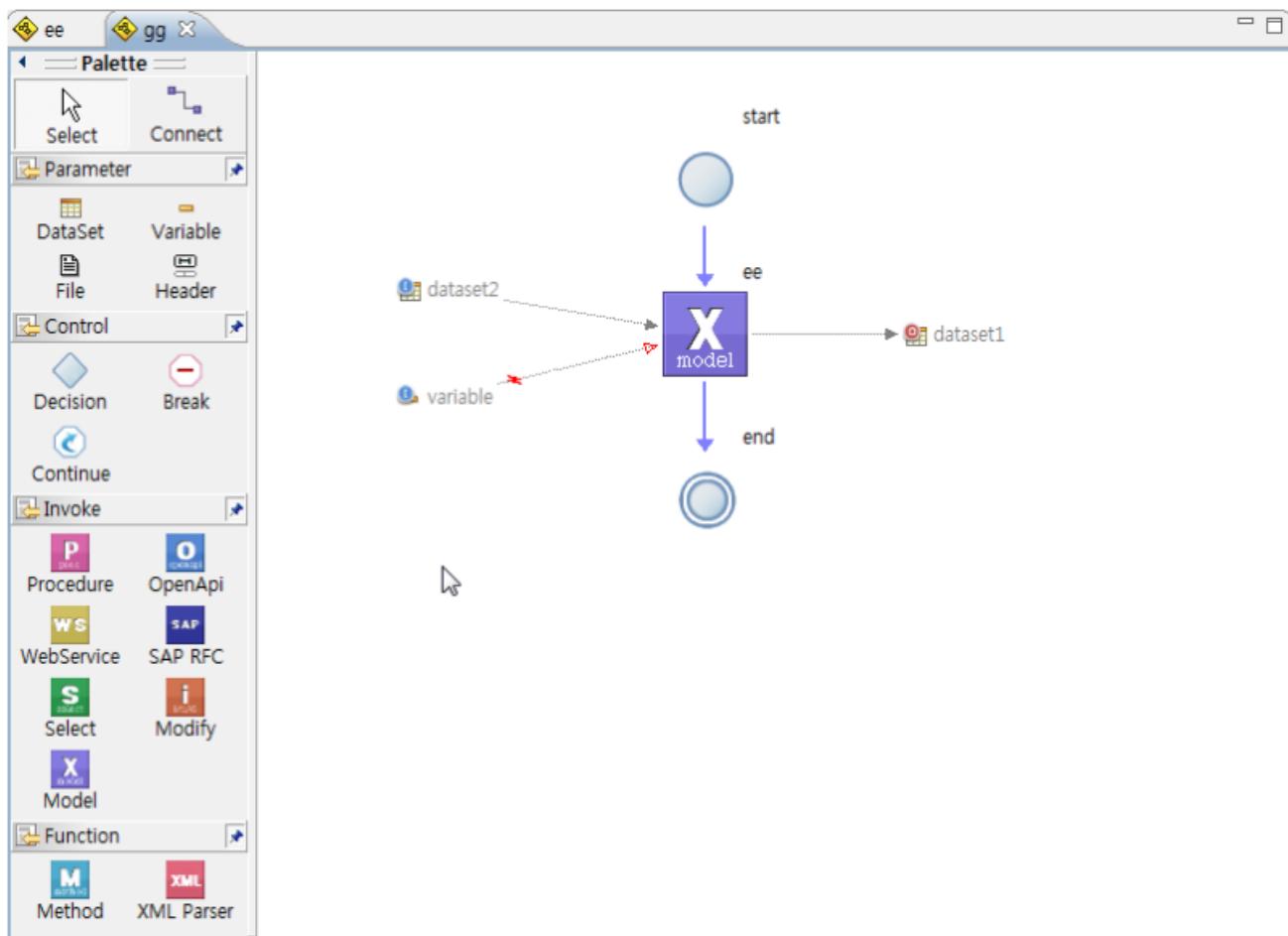
Connect는 노드들간에 커넥션을 생성할 수 있습니다. 커넥션이 선택되어지면 select모드로 변경되기 전까지 계속해서 커넥션을 생성할 수 있습니다. 커넥션 종류는 연결되는 노드의 타입에 따라 자동으로 결정됩니다.

기타 나머지 항목은 아래 상세 기능 정보에서 확인할 수 있습니다.

Drawing Pane

Drawing Pane은 실제 서비스를 디자인하기 위한 에디터입니다. 코딩을 최소화하고 드래그 앤 드랍 방식으로 Graphical하게 비즈니스 서비스를 구현할 수 있습니다.

처음 모델을 생성하면 Start 와 End노드만 존재합니다.



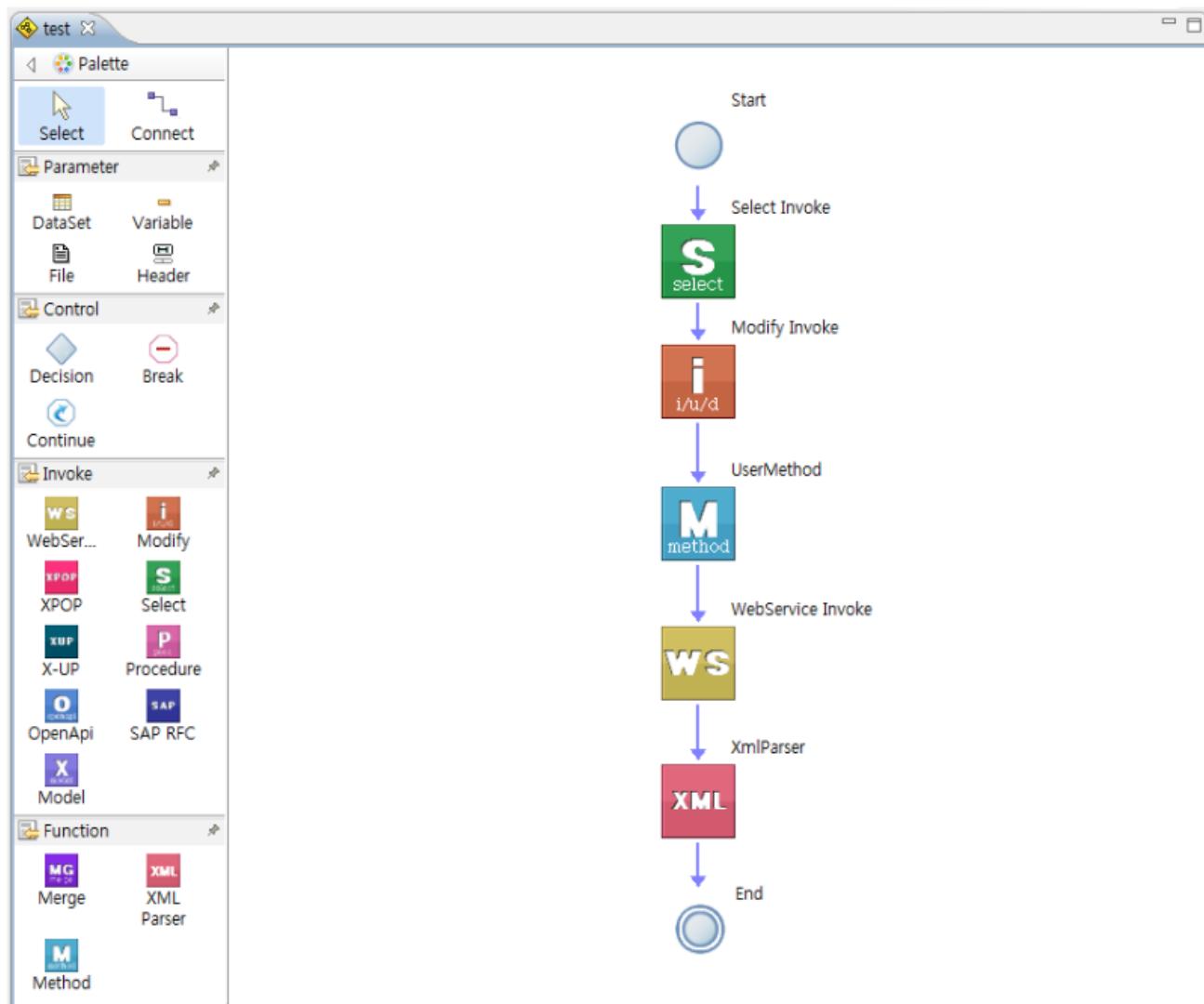
Start 와 End는 서비스의 시작과 종료를 의미하며 자유롭게 원하는 컴포넌트를 드래그 앤 드랍하여 비즈니스 서비스를 구성할 수 있습니다. XAM Editor는 기본적으로 컴포넌트와 커넥션의 조합으로 빠르고 쉽게 원하는 서비스를 구현할 수 있으며, 에디터에서 직접 사용자가 자바 코드를 등록하여 다른 컴포넌트와 함께 동작할 수도 있습니다.

XAM Editor에는 크게 4가지 타입의 커넥션이 존재합니다.

- 프로세스의 흐름을 정의하는 프로세스 커넥션
- 데이터의 흐름을 나타내는 데이터 커넥션
- 이벤트 핸들러와 연결된 이벤트 커넥션
- Decision 노드와 연결된 조건 커넥션

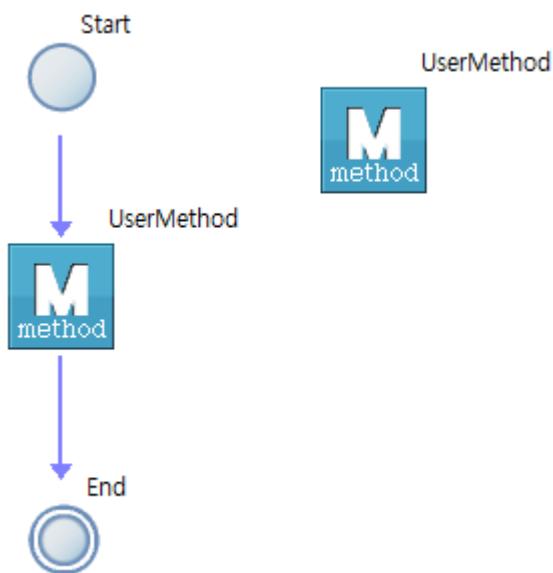
프로세스 커넥션

프로세스 커넥션은 하나의 서비스가 동작하기 위한 실행 순서를 나타냅니다.



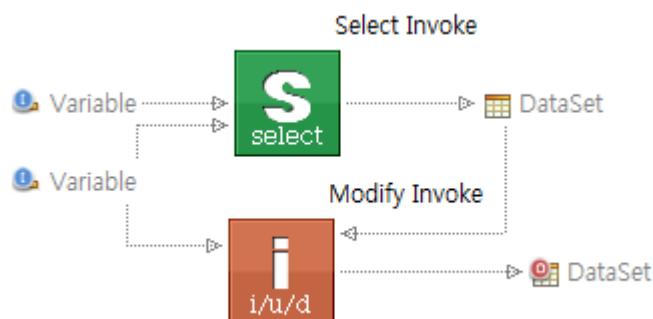
시작에서부터 종료에 이르기까지 각 컴포넌트의 실행 순서를 나타내므로 직관적인 서비스를 구현이 가능하고 가독성을 높입니다.

프로세스 커넥션이 연결되지 않은 컴포넌트들은 하나의 메서드로만 남겨져 실제 서비스에서 제외됩니다. 다만 사용자가 직접 소스 코드로 메소드를 호출하여 실행하는 것은 가능합니다.



데이터 커넥션

데이터 커넥션은 말그대로 컴포넌트를 기준으로한 데이터의 흐름을 나타냅니다.

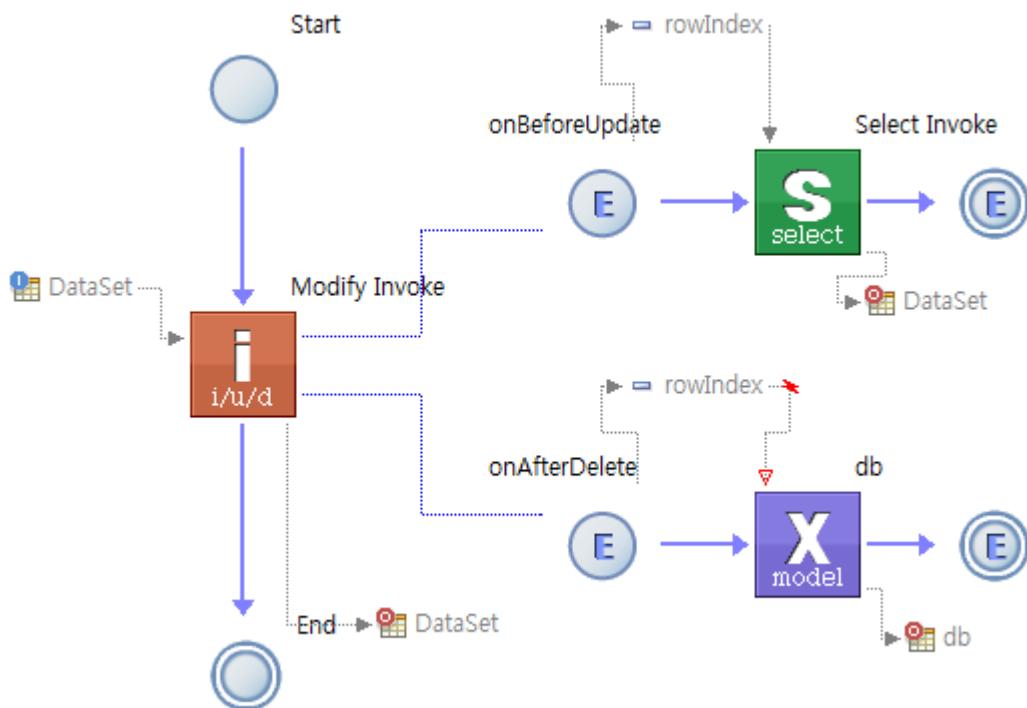


XAM에서의 모든 데이터는 파라미터로 관리되며 어떤 파라미터가 input 으로 사용되며, 컴포넌트 실행 결과로 어떤 파라미터가 output 으로 생성되는지 직관적으로 표현합니다.

파라미터는 하나이상의 컴포넌트에 input 으로 사용될 수 있으며 output 파라미터 역시 다른 컴포넌트의 input 으로 다시 사용될 수 있습니다.

이벤트 커넥션

이벤트 커넥션은 컴포넌트에 이벤트를 추가할 경우 자동으로 생성되며, 이벤트가 추가되었음을 나타냅니다.

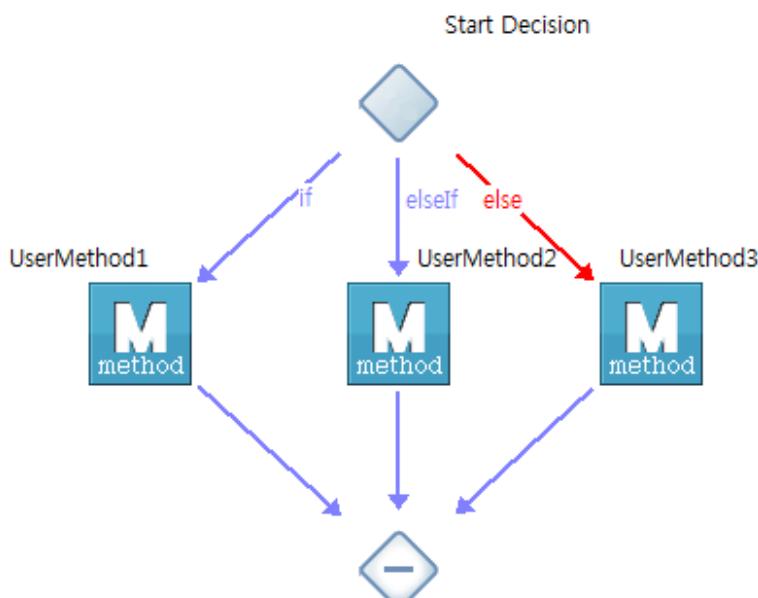


예를 들어 2개의 이벤트를 추가할 경우 이벤트 시작과 종료 노드가 생성되며 해당 컴포넌트에 이벤트 커넥션이 연결됩니다. 이벤트의 종류와 사용법은 컴포넌트에 따라 각각 다르며 이벤트 내부에서 역시 원하는 컴포넌트로 비즈니스 로직을 구성할 수 있습니다.

또한, 이벤트에서 호출되는 컴포넌트에서 또 다시 이벤트를 추가할 수도 있습니다.

조건 커넥션

조건 커넥션은 Decision Start 노드에 커넥션이 연결될 경우 자동으로 정의되며, 사용자의 조건코드를 직접 입력하여 이하 컴포넌트의 실행 여부를 결정할 수 있습니다.



if / else if / else 세가지 타입이 있으며 else 를 제외한 조건 커넥션은 사용자가 직접 조건 로직을 코드로 작성할 수 있습니다.

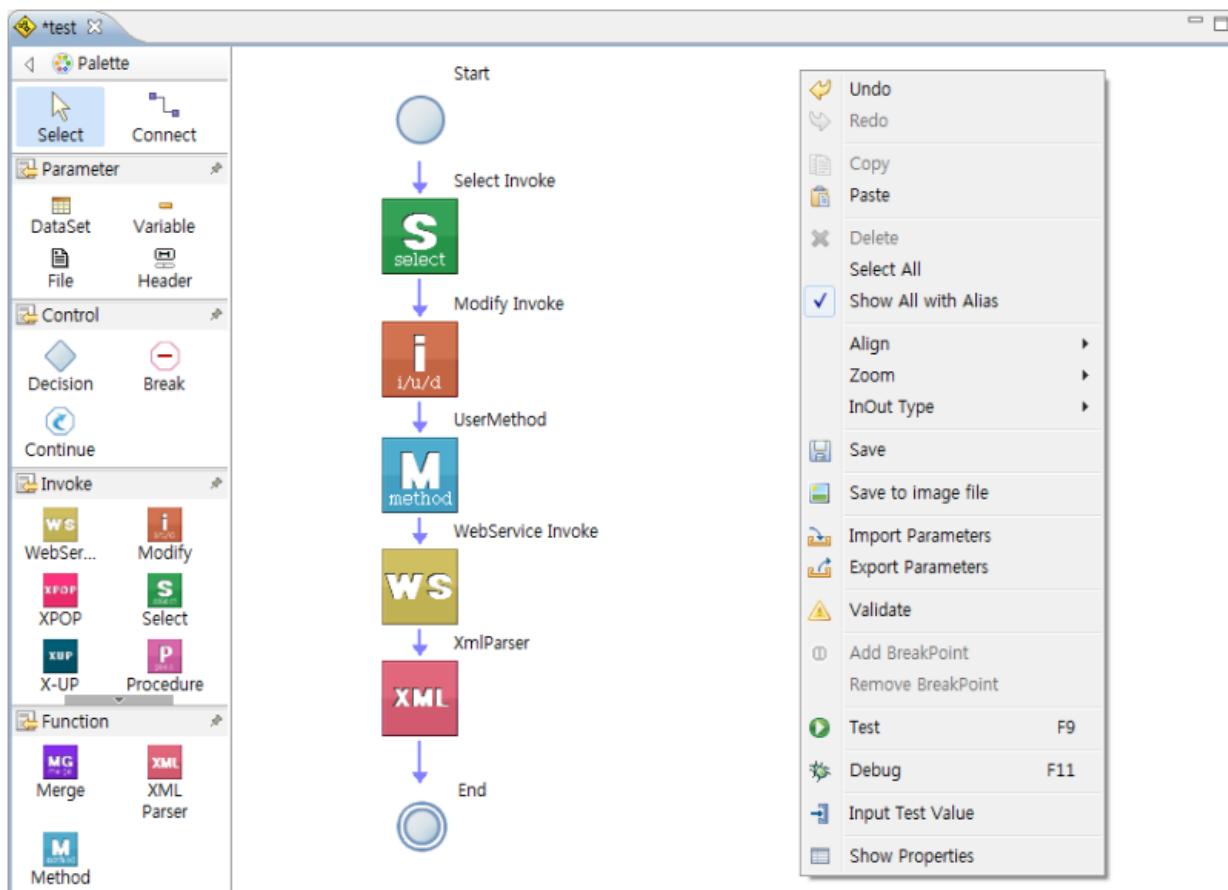
```
public boolean decisionconnection(ParameterSet globalParameterSet) throws
AutomationFailException {
    int currRowIndex = globalParameterSet.getIntValue("rIndex");
    DataSet ds = globalParameterSet.getDataSet("dataset1");

    if(currRowIndex == ds.getRowCount()) return false;
    else {
        return true;
    }
}
```

Menu, 개발도구 Bar

XAM Editor를 위한 메뉴와 툴바에는 자주 사용되는 에디터 메뉴들이 존재합니다.

에디터에서 오른쪽 마우스를 클릭했을 때 나오는 메뉴항목은 다음과 같습니다.



- Undo : 이전 작업으로 복원
- Redo : 복원했던 작업 다시 수행
- Copy : 에디터 노드를 클립보드에 복사
- Paste : 클립보드에 복사된 항목 붙여넣기
- Delete : 삭제
- Select All : 전체 선택
- Show All with Alias : 모든 Alias들을 보여주기
- Align : 정렬
 - Align Left / Center / Right / Top / Middle / Bottom
- Zoom : 확대
 - Zoom In / Zoom Out
- InOutType : In Out Type 정의
 - In / Out / Normal / In+Out
- Save : 저장
- Save to image file : 에디터화면을 이미지로 저장
- Import Parameters : 파라메터를 외부에서 가져오기(XML형식 파일)
- Export Parameters : 파라메터를 XML형식파일로 보내기
- Validate : 모델인보크된 항목에 대한 오류 검사
- Add BreakPoint : 브레이크 포인트 등록
- Remove BreakPoint : 브레이크 포인트 삭제
- Test : 모델 테스트 실행
- Debug : 디버그 모드로 실행
- Input Test Value : 모델 테스트시 필요한 Input parameter 등록창
- Show Properties : 프로퍼티뷰 활성화

Outline

Outline 은 Editor에서 작성된 정보들을 요약하여 다시 정렬하여 보여줍니다.

또한 Outline 과 Editor는 동기화되어 있어서 어느곳에서든 에디터 정보가 변경되면 동일하게 적용됩니다.

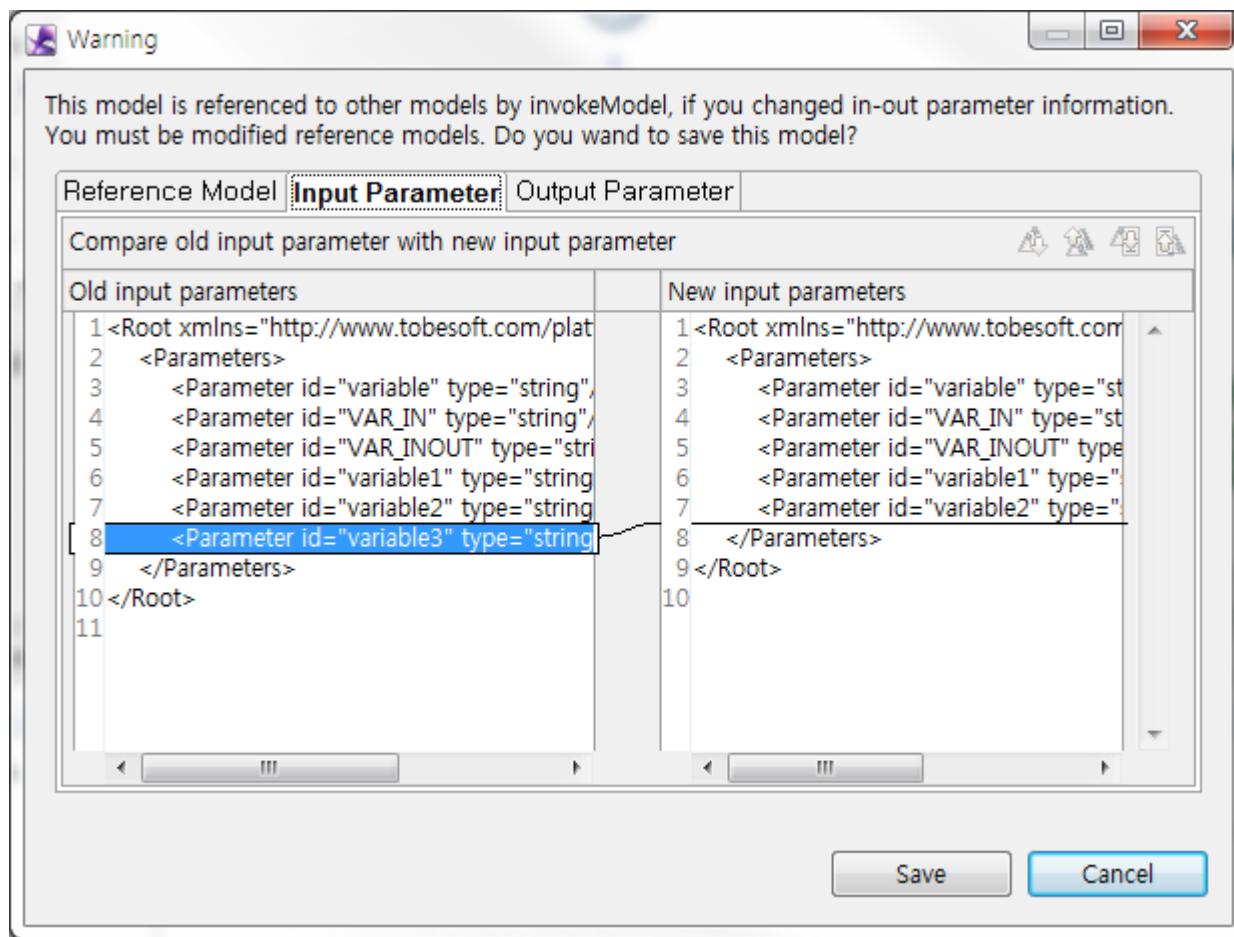
Outline 의 트리는 크게 다음 두가지로 구분되어 정렬됩니다.

- Parameters
- Processes

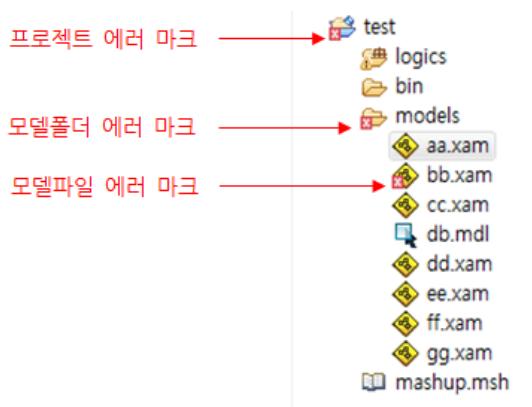
Validate and Error Mark

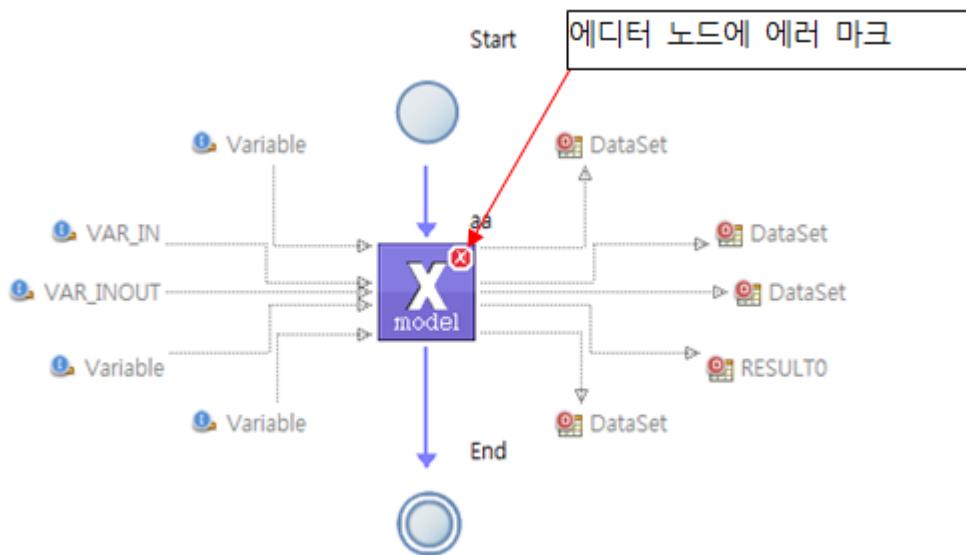
X-UP Builder는 Automation Model 개발시 발생할 수 있는 오류를 사전에 방지하기 위한 validate 기능을 가지고 있으며 오류 감지시 해당 노드에 Error Mark를 표시합니다.

자신을 참조중인 다른 모델이 존재한 가운데 자신의 주요정보가 변경될 경우 변경전과 변경후의 스키마 비교창을 출력합니다.



레퍼런스가 존재함에도 모델정보를 변경할 경우 참조한 모델파일과 에디터에서 해당 노드에 에러마크 표시됩니다.





validate 체크는 툴바 및 메뉴를 통해 수행할 수 있으며 모델 단위뿐만 아니라 프로젝트내 모든 모델을 한꺼번에 검사 할 수도 있습니다.

만일 에러의 원인이 없어지면 에러마크도 함께 없어집니다.



4.5 X-UP View

X-UP Builder에서는 다양한 뷰들이 존재합니다. X-UP Builder에서 제공하는 뷰에는 모델 개발시 공통적으로 사용 되는 뷰가 있습니다.

모델 개발시 공통적으로 사용되는 뷰

- Result DataSet View
- Outline View
- Properties View
- X-UP DataSource View

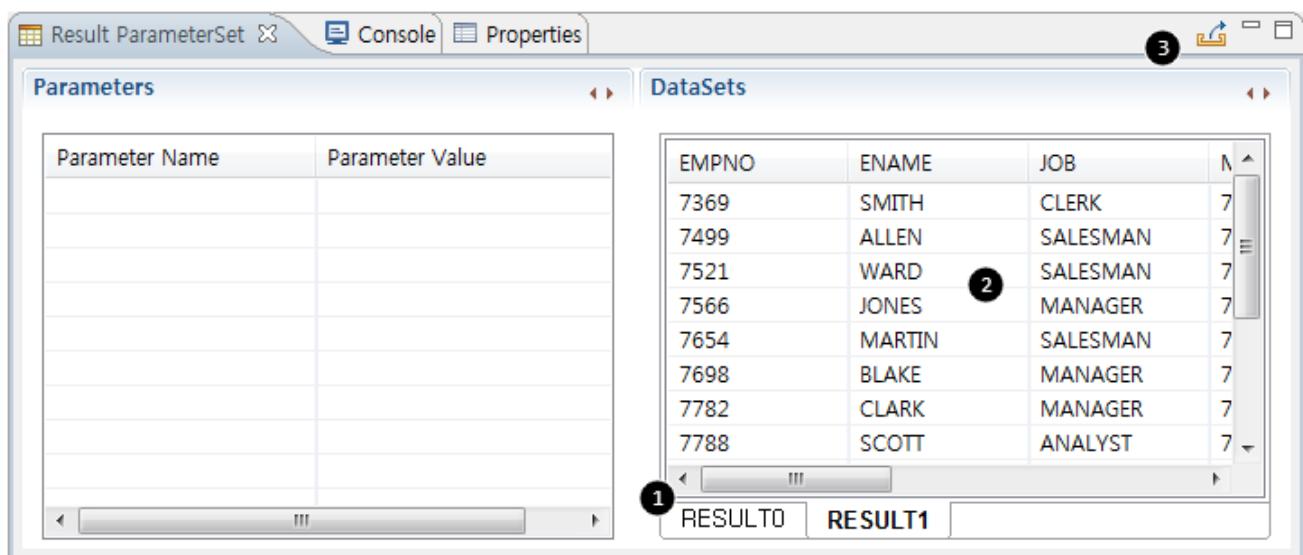
Result DataSet View

Result DataSet View 는 테스트하여 나온 결과 데이터셋의 정보를 테이블 형태로 출력합니다.

만약 데이터셋이 하나이상 존재할 경우 Tab 으로 구분하여 출력합니다.

Result DataSet View 는 다음과 같은 작업 후에 결과값을 보여주기 위해 나옵니다.

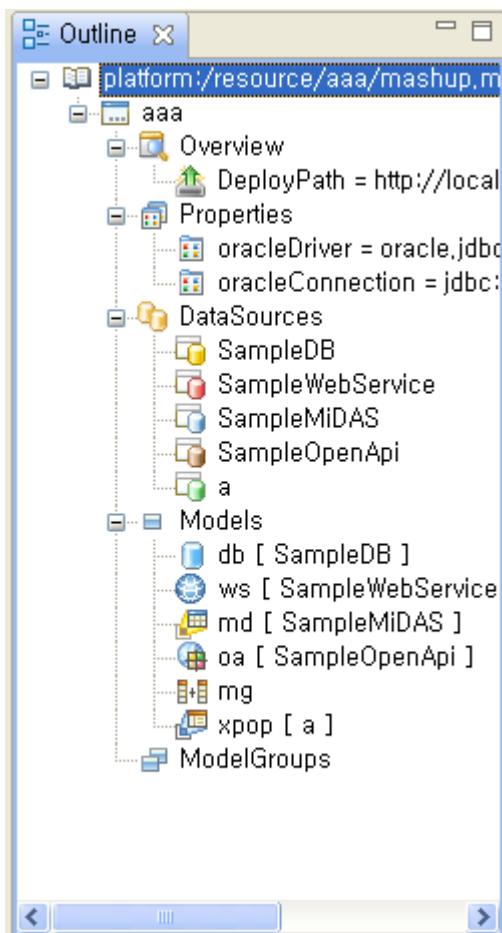
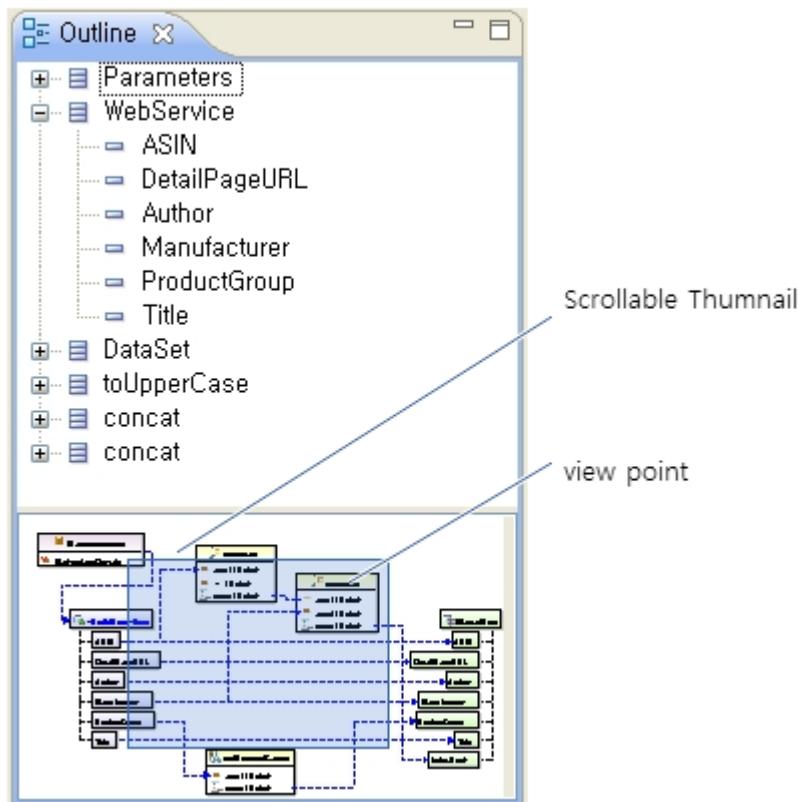
- 각 모델의 타입별 뷰에서 Preview 버튼을 클릭했을 경우
- 각 모델 에디터에서 모델 개발을 완료한 후 Test 를 했을 경우



	Name	Description
①	DataSet Tab	하나이상의 데이터셋이 존재할 경우 데이터셋 이름을 가진Tab으로 구분합니다.
②	DataSet Table	데이터셋 정보
③	Save	데이터셋 정보를 csv 형태의 파일로 저장

Outline View

현재 활성화된 에디터에 따른 간략한 정보를 트리 형태로 보여줍니다.





Scrollable Thumbnail 은 Model Editor's Outline view 하단에 출력되며 작업중인 에디터의 전체 그림을 화면에 맞게 축소시켜 보여줍니다.

또 에디터 화면에서 숨겨진 모습을 포함한 전체 그림을 보여주며, 뷰포인터를 마우스드래그를 하면 에디터 화면을 이동시킬 수 있습니다.

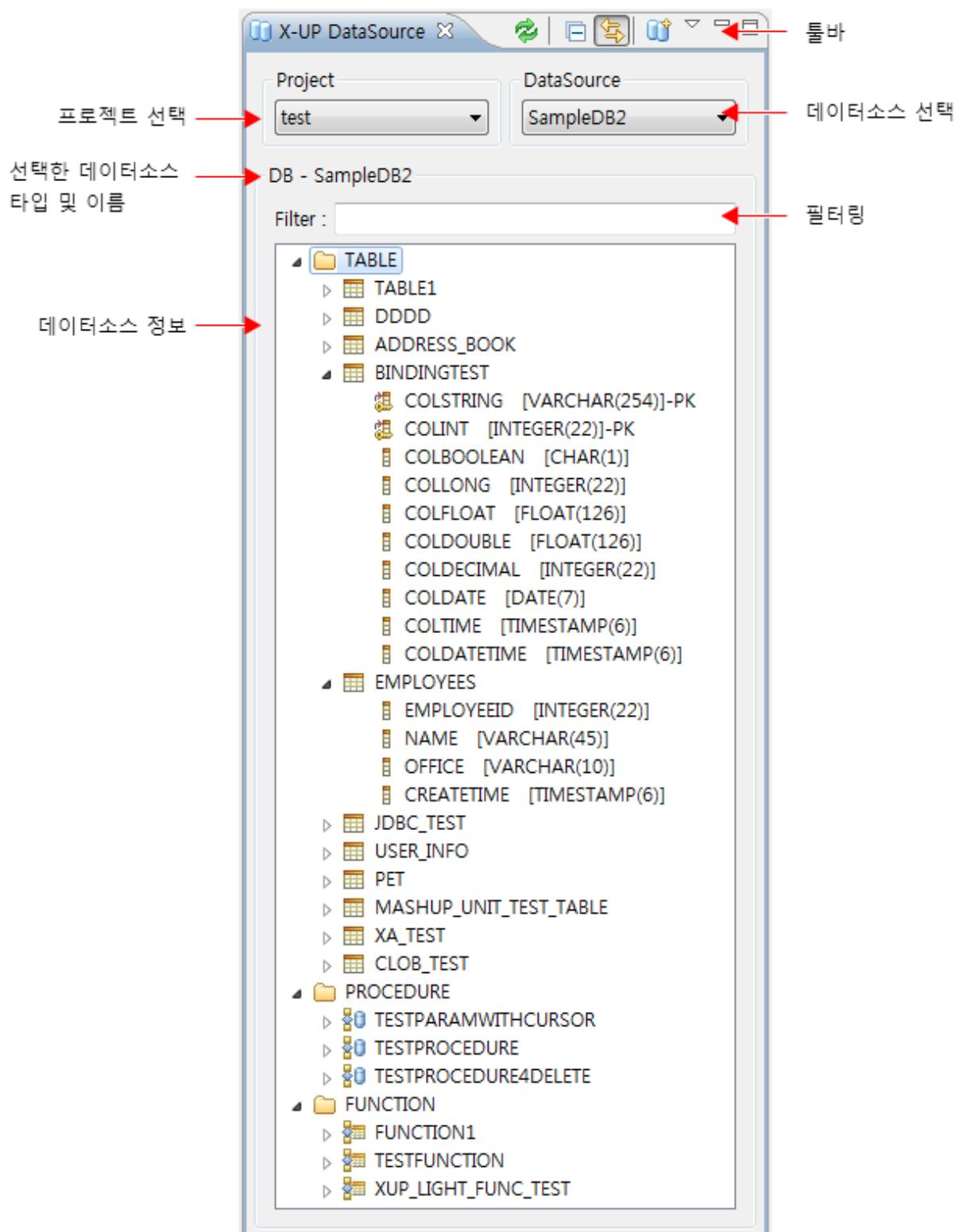
Properties View

이클립스에선 Properties View가 다양한곳에서 사용됩니다. X-UP Builder에서는 모델에디터에서 특정 노드의 속성 값을 설정하기 위해 주로 사용합니다.

Select	Properties	Description
Editor Pane	Cache Cache Polich Cla ss Charset Description Logic Name	Cache 사용 유무 Cache 클래스명 문자셋 정보 모델 설명 등록 로직클래스 이름 모델 이름
Parameter Table	Name	파라메터 테이블 이름
Parameter Column	Name Size Type Value	컬럼 이름 컬럼 사이즈 컬럼 타입 파라메터 컬럼값으로 직접 수정 가능
Tree Header	Name	트리 이름(raw data tree 일 경우는 데이터소스이름)
Tree Column	IsConst Name Size Type Value	Constant column 인지 설정 컬럼 이름 컬럼 사이즈 컬럼 타입 (string / int / float / . long / double / bool / decimal / date / time / datetime / blob) 타겟 트리의 컬럼만 값을 입력할수있다. (타겟 트리의 컬럼 value 를 입력하면 컬럼에 직접 값이 등록되어 해당 컬럼의 서비스값이 아니라 고정된 값이 리턴됩니다.)
Function Table	Name	평선 이름
Function Parameter Column	Name Type Value	평선 파라메터 이름 파라메터 타입 파라메터 값 (주의 : 값이 설정되면 고정된 값으로 리턴됩니다.)
Function Return Column	Name Type	리턴 노드 이름(수정불가) 리턴 타입(자바데이터 타입) (string / int / float / . long / double / boolean / char / Object)

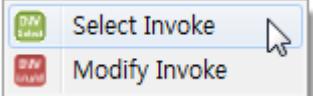
X-UP DataSource View

X-UP DataSource View는 프로젝트내 등록된 데이터소스 정보를 가지고 실제 접속하여 상세 리스트를 조회할 수 있으며 모델 개발시 드래그 앤 드랍을 통한 빠른 인보크 생성을 가능하게 합니다. 현재 DB, WebService, OpenApi, SAP RFC, UDDI, X-UP 데이터소스를 지원하고있으며 향후 계속 추가될 예정입니다.



X-UP DataSource View에서 제공하는 기능은 다음과 같습니다.

Icon	Name	Description
	Refresh	워크스페이스에 존재하는 모든 X-UP 프로젝트를 갱신합니다.
	CollapseAll	펼쳐진 모든 트리를 접습니다.
	Link with Editor	선택된 에디터의 모델이 위치한 프로젝트로 자동 선택됩니다.
	New DataSource	새로운 데이터소스를 생성합니다.
	Go Home	최상위 루트 트리로 이동합니다.
	Go Back	상위 트리 노드로 이동합니다.
	Go Into	선택된 노드의 하위 트리만 출력합니다.
	Select Project	프로젝트를 선택합니다.
	Select DataSource	데이터소스를 선택합니다.
	Filter	특정 문자열로 필터링합니다.
	Show Data	선택된 테이블의 모든 데이터를 조회합니다.
	Copy select sql to clipboard	선택된 테이블을 기준으로 기본 select sql을 생성하여 클립보드에 저장합니다.
	Copy insert sql to clipboard	선택된 테이블을 기준으로 기본 insert sql을 생성하여 클립보드에 저장합니다.
	Copy update sql to clipboard	선택된 테이블을 기준으로 기본 update sql을 생성하여 클립보드에 저장합니다.
	Copy delete sql to clipboard	선택된 테이블을 기

Icon	Name	Description
		준으로 기본 delete s ql를 생성하여 클립 보드에 저장합니다.
	Create Select Invoke	특정 테이블을 선택 한 후 에디터로 드래그 앤 드랍을 하여 Select Invoke를 빠르게 생성합니다.
	Create Modify Invoke	특정 테이블을 선택 한 후 에디터로 드래그 앤 드랍을 하여 Modify Invoke를 빠르게 생성합니다
	Create Procedure Invoke	특정 Procedure나 Function을 선택한 후 에디터로 드래그 앤 드랍을 하여 Procedure Invoke를 빠르게 생성합니다

5.

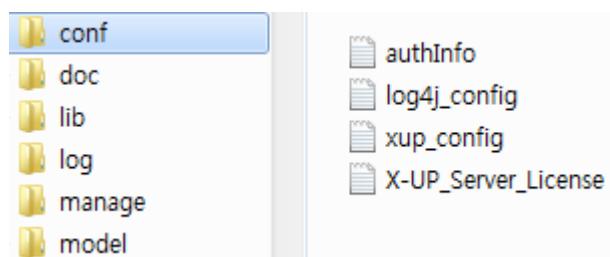
X-UP 설정

이 장에서는 사용자가 개발시에 필요한 X-UP Server와 X-UP Builder의 환경설정에 대하여 설명합니다.

5.1 X-UP Server 설정

X-UP 서버를 설정하기 위해서 여러가지 환경설정 파일을 제공합니다. X-UP 서버를 설치했다는 가정하에 환경설정 관련 파일에 대한 설명입니다.

WAS 설치 경로에서 [webapps > xup > WEB-INF > xup > conf] 경로로 가면 아래 그림과 같이 여러 환경 설정 파일이 존재합니다.



Name	Description
authInfo.xml	X-UP 서버에 관한 계정정보를 갖고있는 파일입니다.
log4j_config.xml	X-UP의 에러로그 설정에 관한 정보를 갖고있는 파일입니다.
xup_config.xml	X-UP의 환경 설정에 대한 정보를 갖고있는 파일입니다.
X-UP_Server_License.xml	X-UP 서버의 라이선스 파일입니다.

xup_config.xml 설정

xup_config.xml파일에는 X-UP Server에 대한 아래와 같은 환경설정정보를 가지고 있습니다.

	Name	Type	Description
properties	isBundleDevMode	Boolean	X-UP Server의 Bundle 개발 모듈 실행 여부입니다.
	isManageMode	Boolean	X-UP 관리자 실행 여부입니다.
	isDebugEnabled	Boolean	디버그에 대한 실행 여부입니다. isDebugEnabled가 true일 경우에만 로그를 남깁니다.
	exception.loglevel	Int	예외처리에 대한 로그 레벨 설정입니다.
	enableModelSingleton	Boolean	모델의 싱글톤 가능 여부입니다.
	logger.Api.enabled	Boolean	X-UP Server의 Api단에서 생기는 로그를 남길 지에 대한 여부입니다.
	logger.Parameter.enabled	Boolean	X-UP Server의 파라미터를 핸들링할 때 발생하는 로그를 남길 지에 대한 여부입니다.
	logger.DataSetBuilder.enabled	Boolean	Gathering 모델에서 원시데이터를 데이터셋으로 바꿔줄 때 발생하는 로그를 남길 지에 대한 여부입니다.
	logger.ParameterSetBuilder.enabled	Boolean	Invoke에서 발생하는 로그를 남길 지에 대한 여부입니다.



X-UP Server에서 사용하는 exception log level은 다음과 같습니다.

- 1 : allException 전체 예외에 대하여 전체 StackTrace와 메시지를 뿐입니다.
- 2 : allException shortly 전체 예외에 대하여 간단한 StackTrace와 메시지만 뿐입니다.
- 3 : userException 현재 예외에 대해 전체 StackTrace와 메시지를 뿐입니다.
- 4 : userException 현재 예외에 대해 간단한 StackTrace와 메시지를 뿐입니다. 4 레벨이 예외 로그가 가장 최소화되어 있습니다.



로그에 관하여 xup_config.xml 파일의 isDebugEnabled의 설정이 가장 우선 역시 됩니다. isDebugEnabled가 'False'로 설정되어있다면 log4j.xml 파일의 설정과 상관없이 로그를 남기지 않습니다.

log4j.xml 설정

X-UP은 자바 어플리케이션에서 빠르고 효과적으로 로깅할 수 있도록 도와주는 오픈 소스 프로젝트인 Apache사의 log4j를 지원합니다.



log4j에 대한 간략한 가이드

log4j의 구성

- Logger(Category) : 로깅 메세지를 Appender에 전달합니다.
- Appender : 전달된 로깅 메세지를 파일에다 기록할 것인지, 콘솔에 출력할 것인지 아니면 DB에 저장할 것인지 매개체 역할을 합니다.
- Layout : Appender가 어디에 출력할 것인지 결정했다면 어떤 형식으로 출력할 것인지 출력 layout을 결정합니다.

log4j의 로깅 레벨

- FATAL : 가장 크리티컬한 에러가 일어 났을 때 사용합니다.
- ERROR : 일반 에러가 일어 났을 때 사용합니다.
- WARN : 에러는 아니지만 주의할 필요가 있을 때 사용합니다.
- INFO : 일반 정보를 나타낼 때 사용합니다.
- DEBUG : 일반 정보를 상세히 나타낼 때 사용합니다.

auth_info.xml 설정

auth_info 파일은 X-UP에 대한 계정정보를 갖고있는 파일입니다.

기본 계정은 xup/xup이며 비밀번호 변경을 원한다면 <password></password> 값을 변경하면되고, 새로운 계정을 추가한다면 <userList></userList>안에 새로운 <user>에 대한 정보를 입력하고 저장하면 됩니다.

```
<?xml version='1.0' encoding='uft-8'?>
<authInfo>
  <userList>
    <user>
      <name>xup</name>
      <password>xup</password>
      <roles>admin</roles>
    </user>
    <user>
      <name>test</name>
      <password>test</password>
      <roles>admin</roles>
    </user>
  </userList>
</authInfo>
```



auth_info에 계정 및 비밀번호를 변경하게 되면 X-UP 관리자 계정 등에도 변경된 계정으로 적용됩니다.

web.xml 설정

X-UP Server의 웹 어플리케이션 루트 폴더 밑의 xup/WEB-INF에 web.xml파일이 위치합니다. web.xml파일은 서블릿의 설정과 서블릿 매팅, 필터, 인코딩 설정을 담당합니다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <servlet>
        <servlet-name>FrontControllerServlet</servlet-name>
        <servlet-class>com.tobesoft.xup.service.FrontControllerServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>FrontControllerServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
</web-app>
```

JCO libarary 설정

아래 라이브러리들은 SAP 가이드에서 시스템이 설정 된 경로에 해당 라이브러리를 위치 시켜 사용하도록 권장하고 있습니다.

JCO 2.0 Version에 필요한 library	JCO 3.0 Version에 필요한 library
librfc32.dll	sapjco3.dll
sapjcorfc.dll	sapjco3.jar
sapjco.jar	

X-UP에서는 X-UP Builder를 통한 개발 시 좀더 편리하게 사용할 수 있도록 해당 라이브러리를 배포하여 사용할 수 있도록 하고 있습니다.

X-UP에서는 JCO 2.0일 경우 librfc32를 로딩하여 jco 내부에서 sapjcorfc.dll을 로딩합니다. JCO 3.0일 경우는 jco

내부에서 sapjco3.dll을 로딩하기 때문에 X-UP 에서는 별도로 로딩하지 않습니다.

(설치 시 방안 (JCO 3.0 기준이며, JCO 3.0일 경우 librfc32.dll에 동일 적용됩니다.)

1. NetWeaver 사용

X-UP이 로딩 하지 않음

- I. xup_env.property com.tobesoft.useJcoStandalone= **false** 사용.
- II. SAP RFC DataSource 의 sapjco3.jar, sapjco3.dll 삭제

2. 개발 시와 운영 시

X-UP 로딩

- I. SAP RFC DataSource 에 위치 한 library 로딩
System Path에 위치한 library 로딩
- II. xup_env.property com.tobesoft.useStandAlone = **false** 사용.
Java ext 이용
- III. xup_env.property com.tobesoft.useStandAlone = **false** 사용.
- IV. SAP RFC DataSource 의 sapjco3.jar java ext 폴더 이동,
- V. SAP RFC DataSource 의 sapjco3.dll java bin 폴더 이동,

5.2 X-UP Builder Preference 설정

X-UP Preference 는 X-UP Builder의 전체 환경설정 정보를 설정할 수 있습니다.

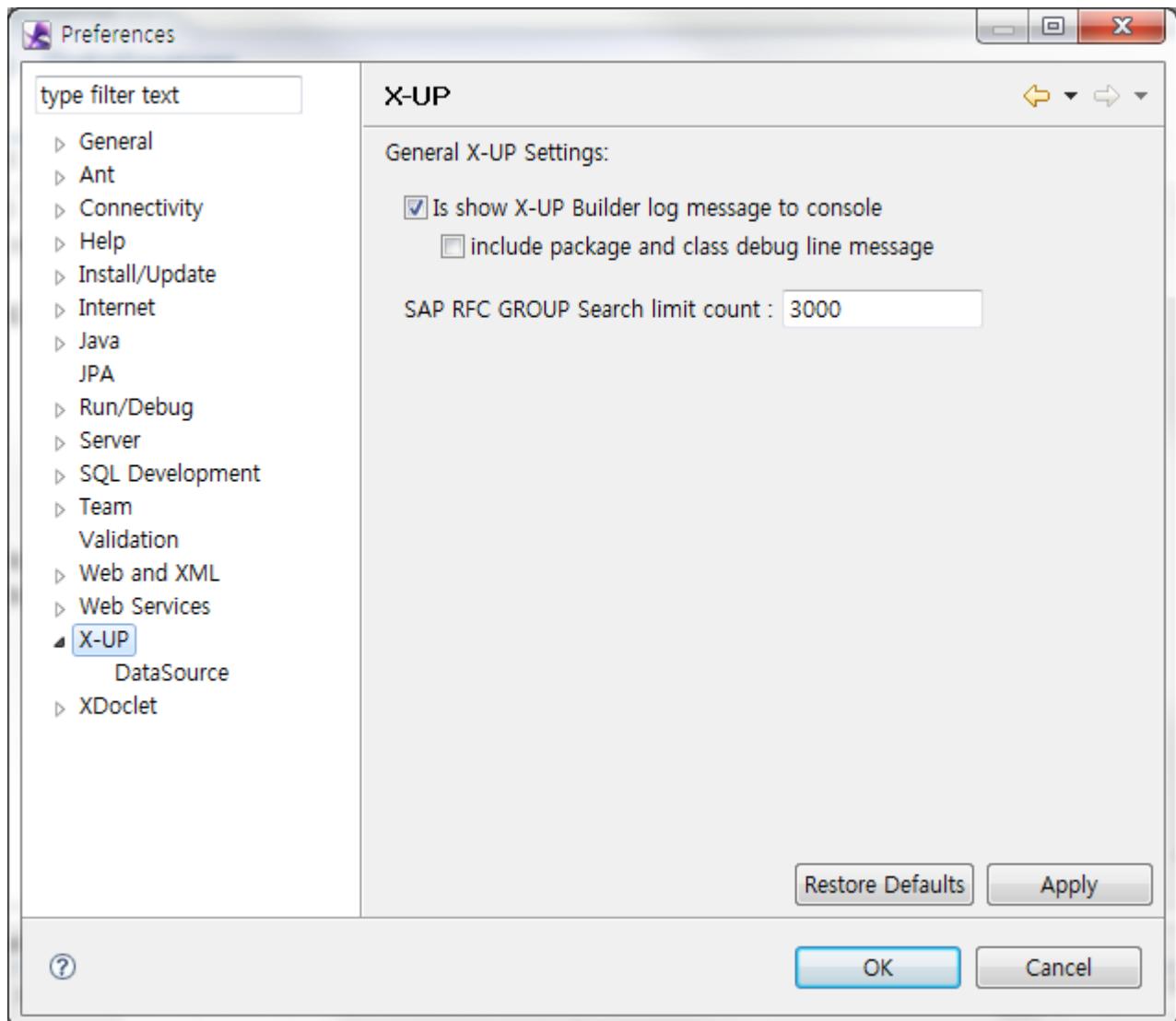
X-UP Preference 는 [Window > Preferences.. > X-UP] 으로 실행할 수 있습니다.

현재 X-UP Preference는 2개 페이지로 구성되어 있습니다.

- X-UP Page : General X-UP Builder Setting
- DataSource Page : 자주 사용하는 데이터 소스 관리

X-UP Page

X-UP Page에는 공통적인 환경설정 정보를 정의할 수 있습니다.



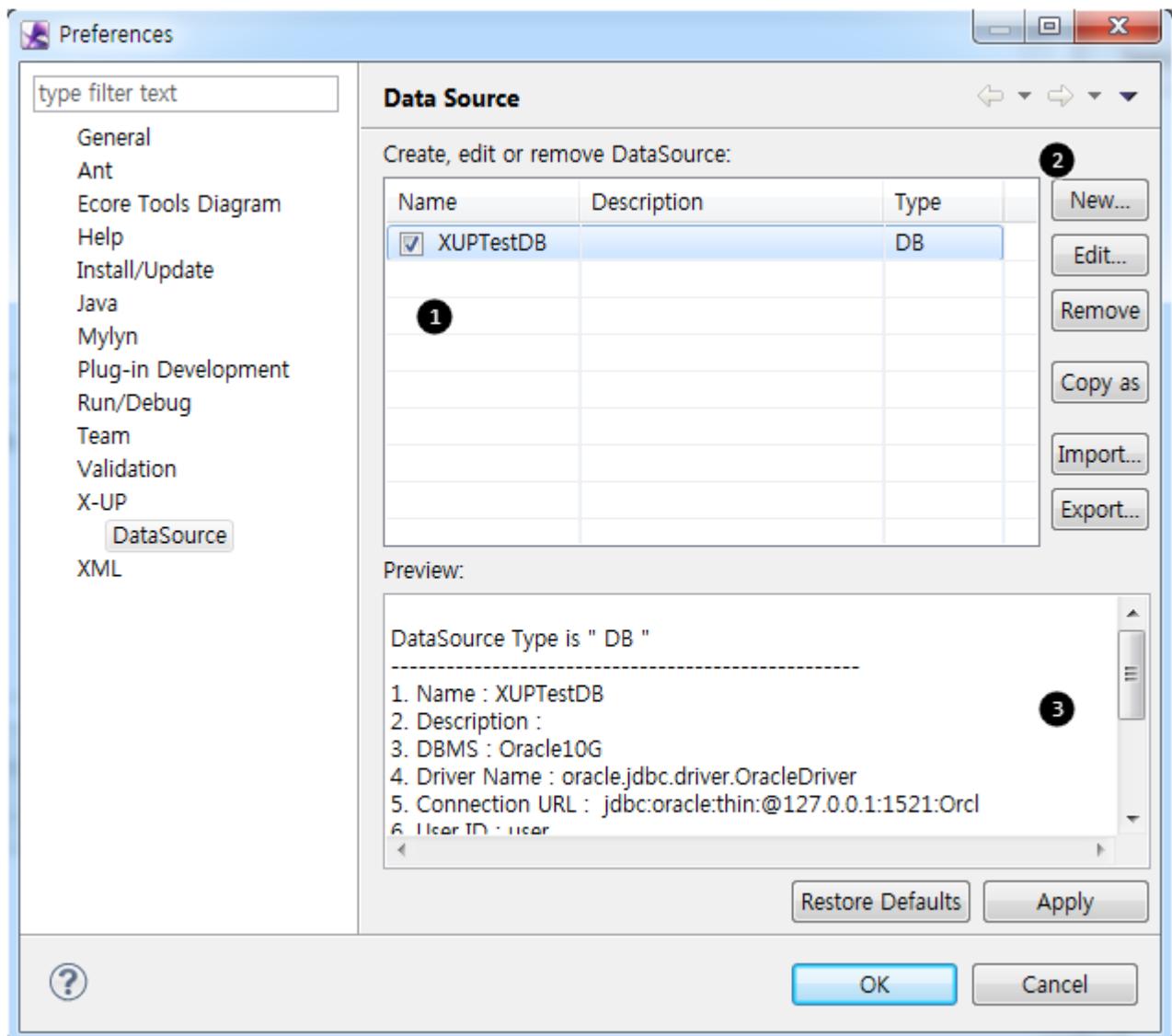
`Is show X-UP Builder log message to console`은 로그 출력을 console에 할 것인지 설정할 수 있습니다. 또한 `include package and class debug line message`를 통해 만약 로그 출력을 콘솔에서 한다면 package 와 class line message 도 포함시킬 것인지 정의할 수 있습니다.

`SAP RFC GROUP Search limit count`는 그룹 조회시 너무 많은 데이터가 나올 수 있으므로 정의된 숫자만큼만 출력되도록 제약을 줄 수 있습니다. 향후 다양한 환경설정 정보는 계속 추가될 예정입니다.

DataSource page

DataSource 페이지는 자주 사용되는 데이터 소스들을 X-UP Builder 에 미리 저장하여 X-UP 모델 생성시 참조 또는 재사용하기 위한 기능을 제공합니다.

등록된 데이터소스들은 프로젝트나 모델에 상관없이 어느 곳에서나 참조될 수 있습니다.

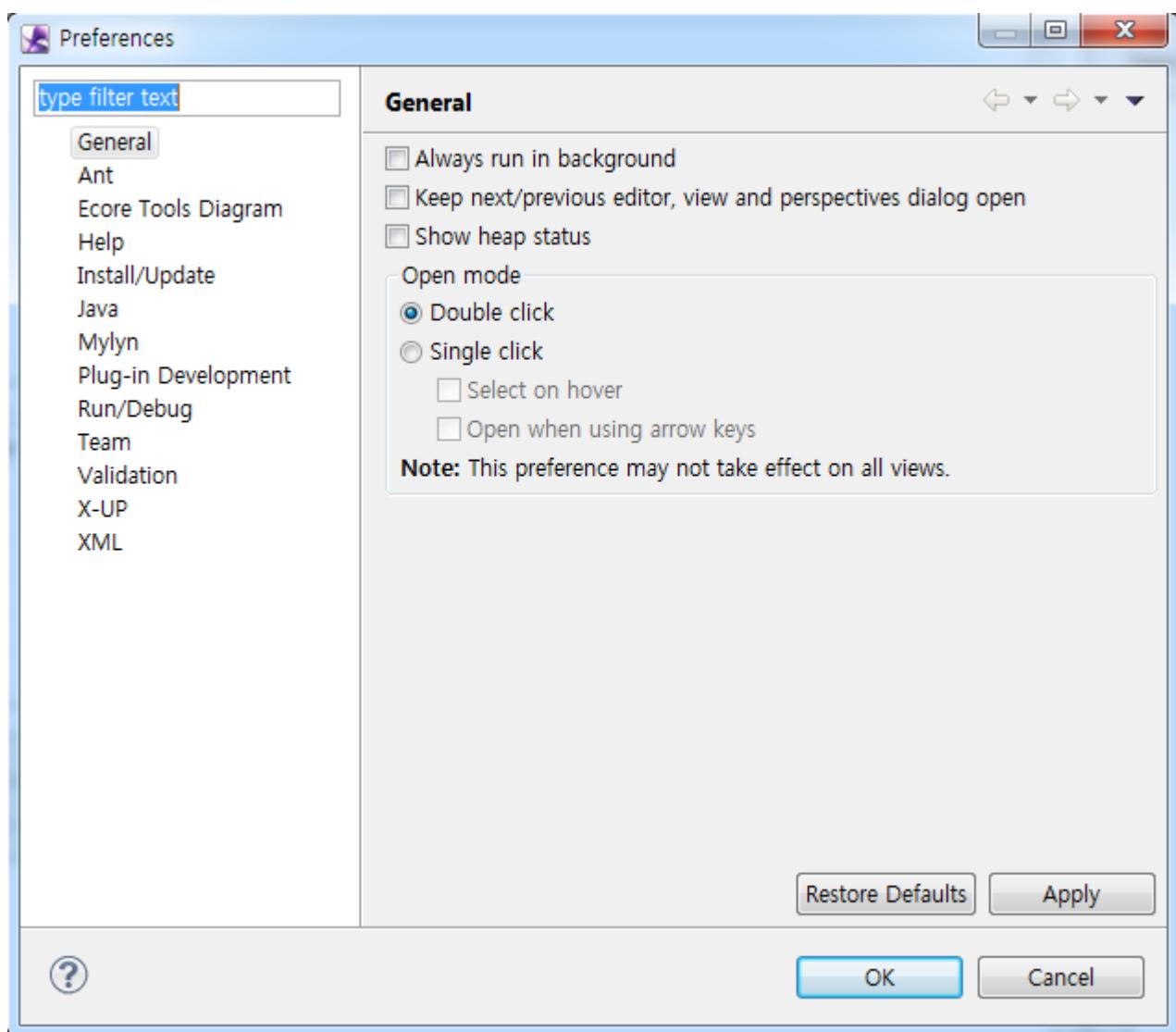


	Name	Description
1	DataSource Table	<p>데이터소스 리스트 테이블은 등록된 데이터소스의 리스트를 보여줍니다. Name : 데이터소스의 이름 CheckBox : 체크박스에 체크가 된 데이터소스만 모델 개발시 데이터소스 선택화면에서 나타나게 됩니다. Description : 데이터소스 작성시 등록된 설명 Service : 등록된 데이터소스의 타입을 출력합니다. 출력될 수 있는 데이터소스 타입은 다음과 같습니다.</p> <ul style="list-style-type: none"> - DB - WebService - UDDI - OpenApi - SapRfc - X-UP
2	Action Buttons	<p>New : 새로운 데이터소스 생성 Edit : 기존 데이터소스를 편집</p>

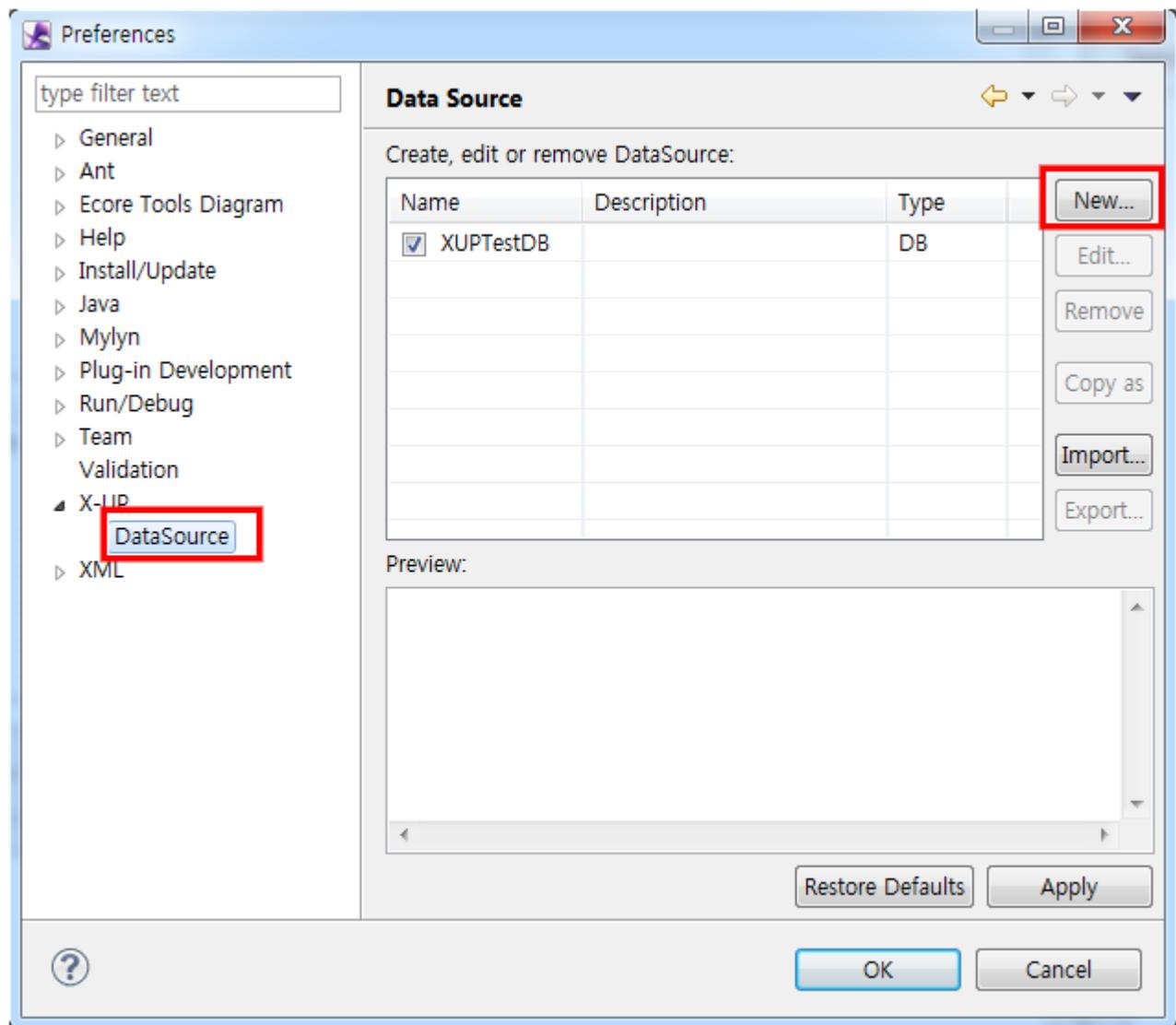
	Name	Description
		Remove : 선택한 데이터소스를 삭제 Copy as : 선택한 데이터소스를 복사 Import : 외부 데이터소스를 등록 Export : 등록된 모든 데이터소스를 xml 형태파일로 원하는 외부 디렉토리에 저장(백업).
③	Preview	선택된 데이터소스의 상세 정보를 출력합니다.

X-UP Builder의 Preference에 데이터소스 정의하여 사용하기

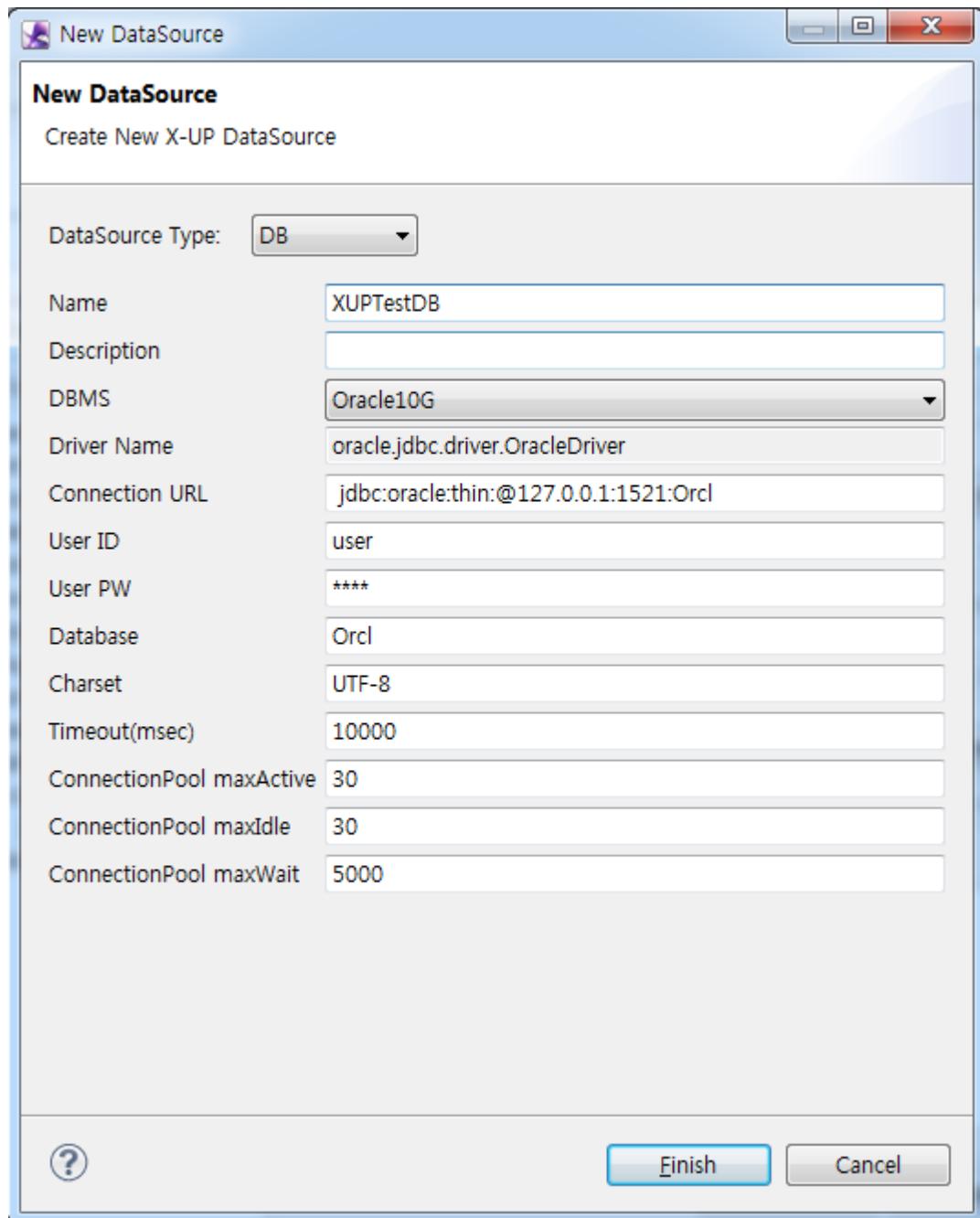
- [Window > Preferences..] 메뉴를 클릭하여 Preferences 위치드를 실행합니다.



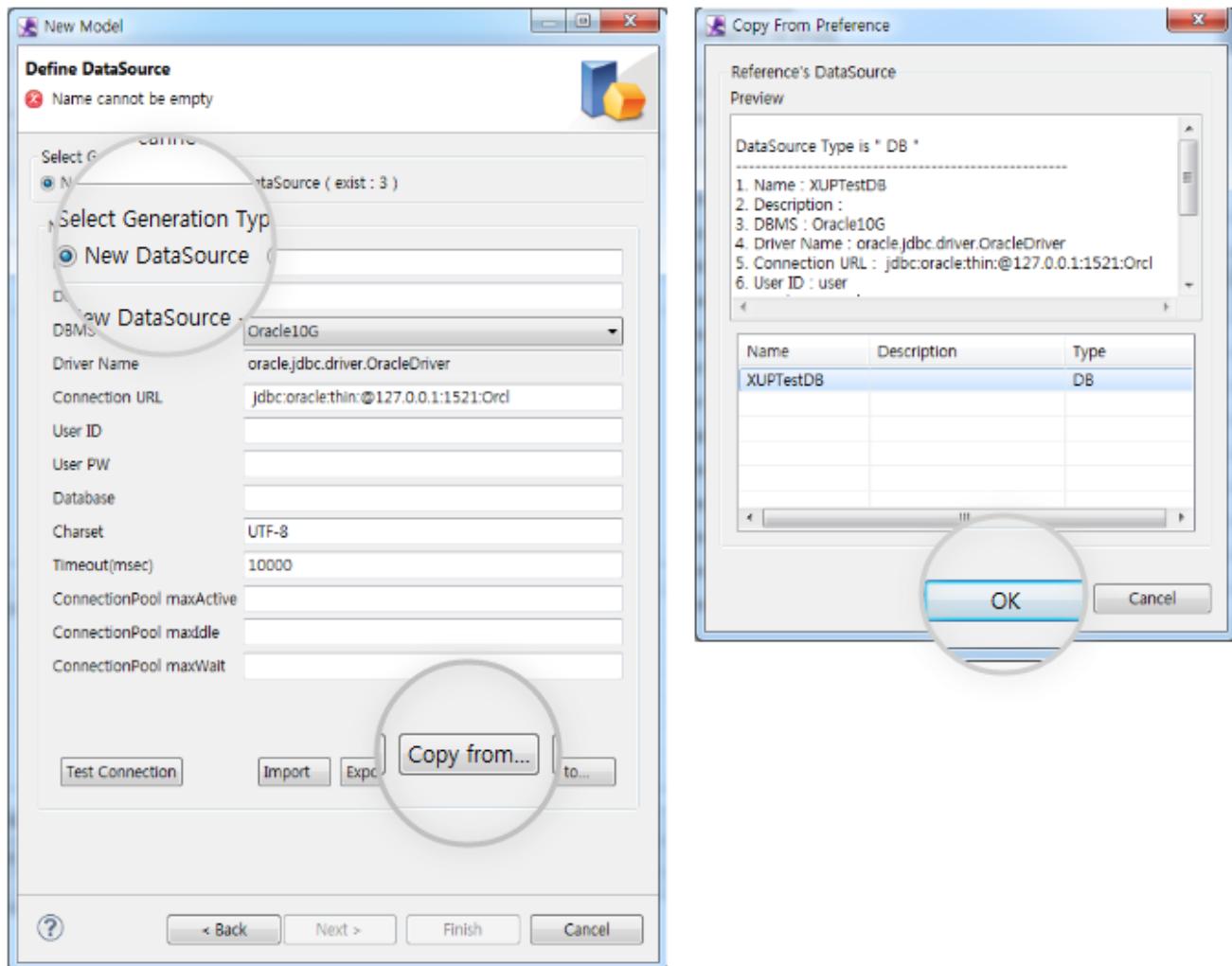
- X-UP DataSource 메뉴를 선택하면 데이터소스 설정화면이 나타납니다.

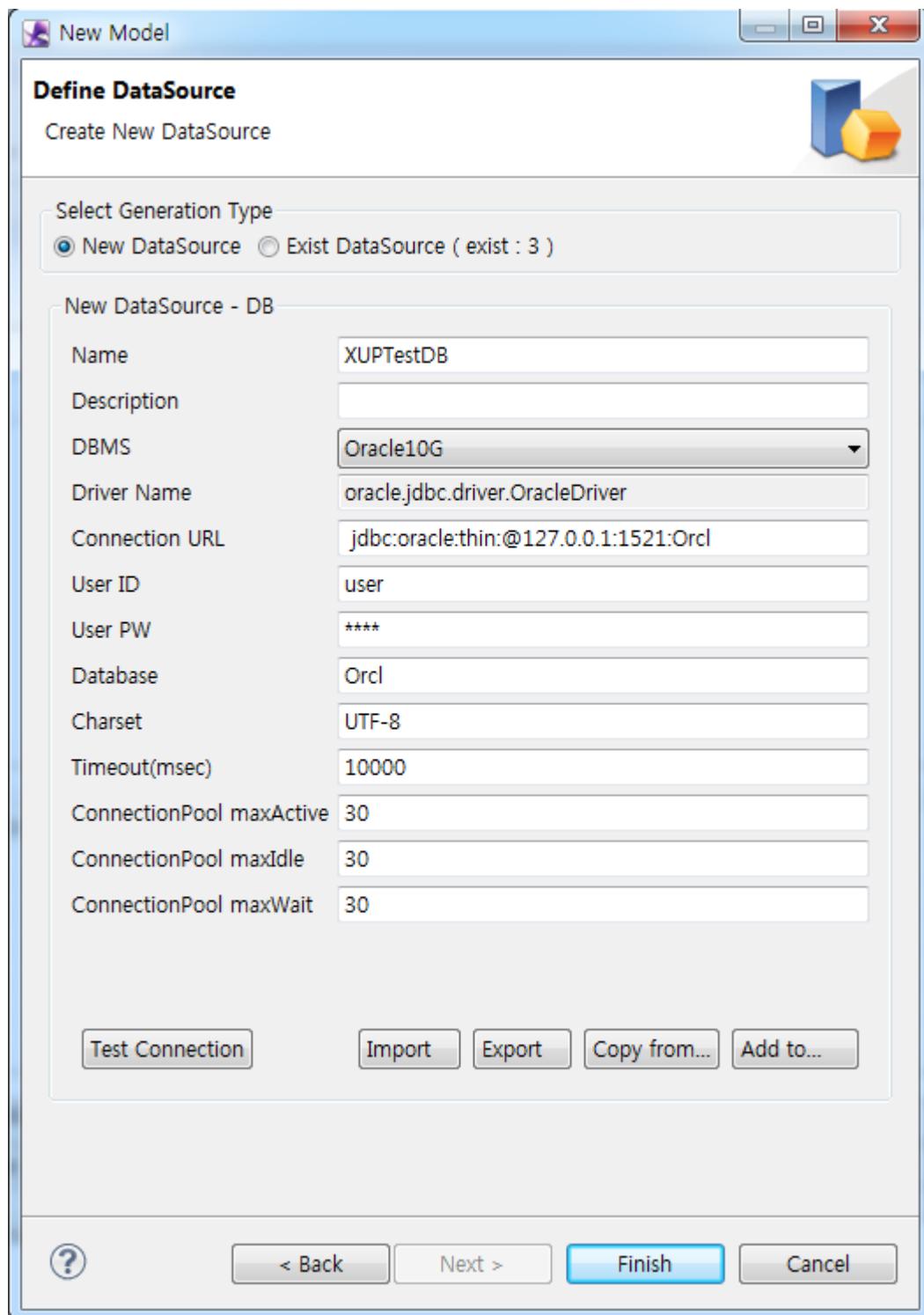


3. New 버튼을 클릭하여 새로운 데이터소스를 정의합니다.



4. 위와 같이 정의된 데이터소스는 X-UP 프로젝트에서 복사하여 사용할 수 있습니다. X-UP 모델 생성 위자드의 'Define DataSource' 창에서 Copy From... 버튼을 클릭하여 Preference에서 정의한 데이터소스를 복사합니다.



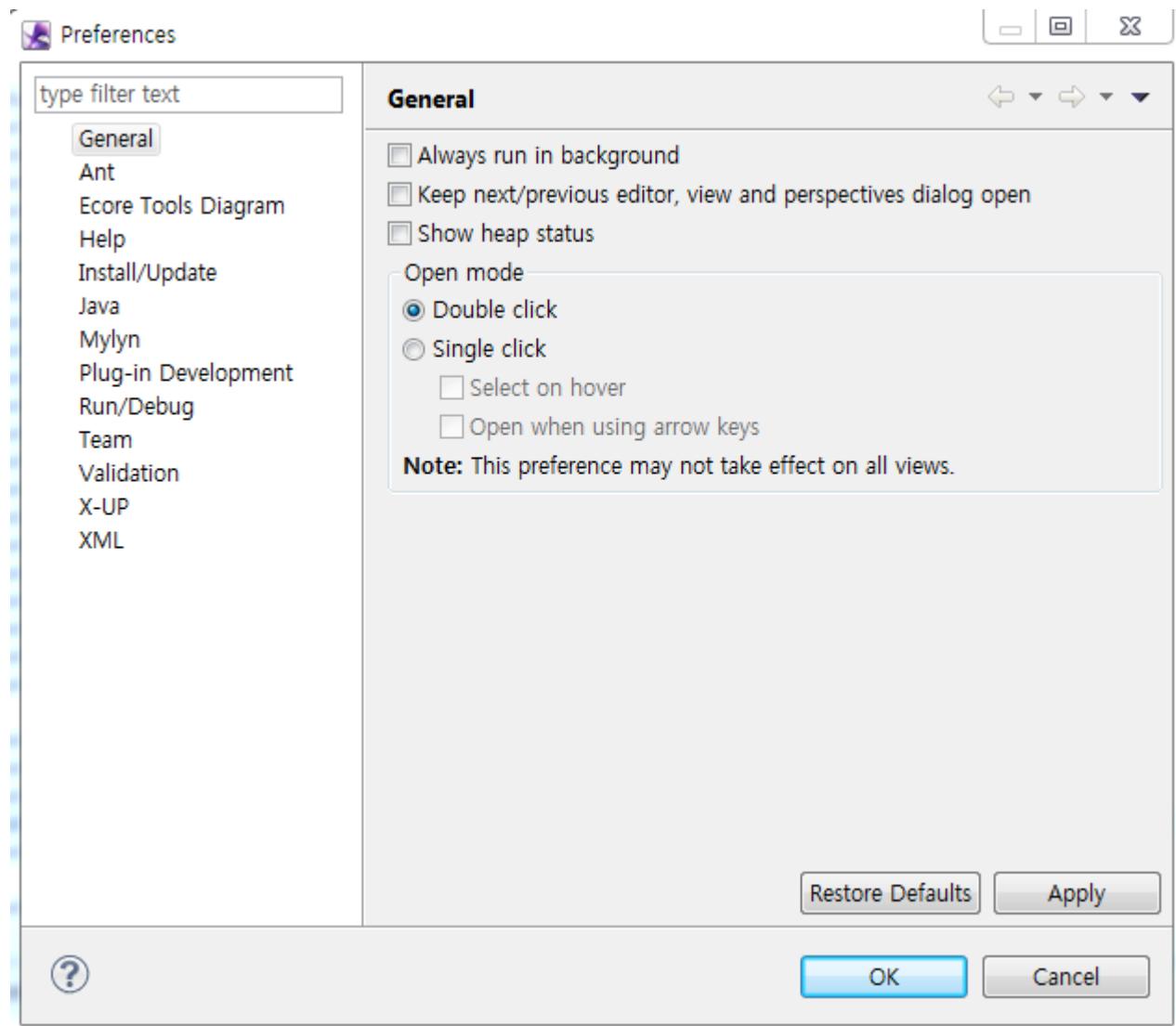


compile 버전 변경

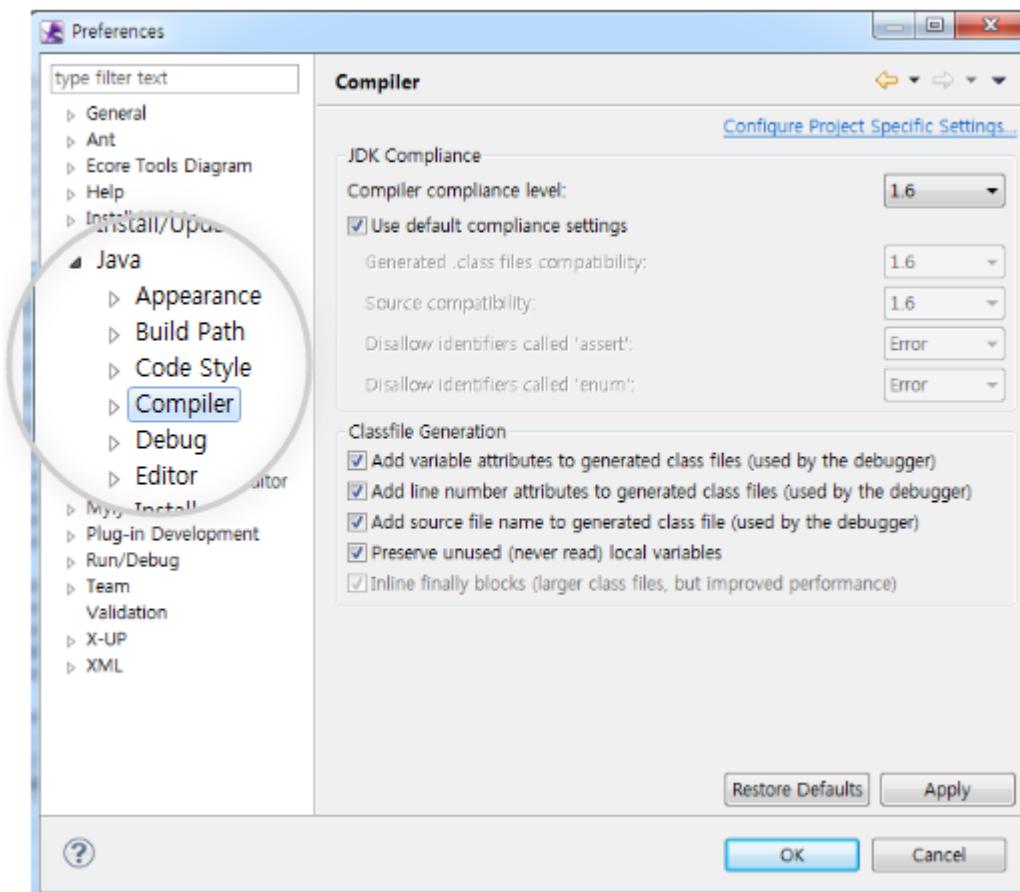
X-UP Builder가 사용하는 JRE 버전은 1.6입니다. 따라서 사용자가 아래와 같이 X-UP Builder에서 컴파일 버전을 변경할 수 있습니다.

X-UP Builder의 Preference에서 Compile 버전 1.5로 변경하기

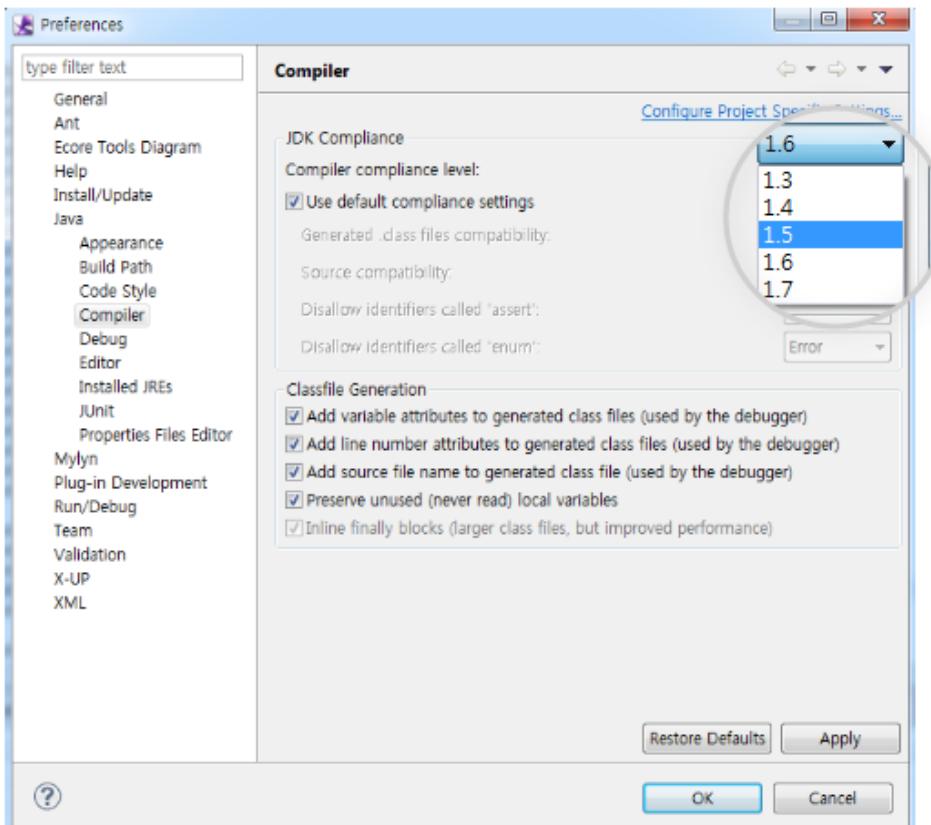
- [Window > Preferences] 메뉴를 클릭하여 Preferences 위자드를 실행합니다.



- Java Compiler 메뉴를 클릭합니다.



3. JDK Compliance의 Compiler compliance level을 1.5를 선택하고 Apply 버튼을 클릭합니다.



6.

X-UP 모델

이 장에서는 X-UP 모델 정책과 X-UP 모델에 정의된 데이터소스로부터 데이터를 수집하고 융합하는 여러 서비스에 대하여 설명합니다.

X-UP은 다음과 같은 X-UP 모델을 제공합니다.

- Automation Model
- Transaction Model

6.1 모델 로직 및 정책

X-UP 모델의 실제 동작은 로직을 담고 있는 java 클래스를 호출하여 이루어 집니다. X-UP Builder는 이러한 java 클래스 소스를 자동생성하며, 이 java 파일이 컴파일 되어 X-UP Server에 배치됩니다. 대부분의 경우 자동 생성된 java 클래스를 수정할 필요 없이 모델의 개발을 완료할 수 있습니다. 그러나 때에 따라서는 X-UP Builder의 기능만으로는 원하는 로직을 구현하지 못하는 경우도 있습니다. 이럴 때는 자동 생성된 java 파일을 직접 수정하여 구현하면 됩니다.

모델 위자드에 의하여 X-UP 모델을 생성하면 각 X-UP 모델의 타입에 따라 java 파일이 자동 생성됩니다.

Eager Loading

X-UP은 효율적인 메모리 관리를 위해 Lazy Loading 처리를 하였으나 디플로이된 모델에 대해 유효성 체크를 진행하고 속도향상 및 성능개선을 위해 Eager Loading으로 변경하였습니다.

Singleton

X-UP은 사용자가 개발한 모델의 실행 로직인 자바 클래스를 Singleton으로 설정할 수 있습니다. 해당 설정을 통하여 인스턴스를 재생성하지 않아 속도가 빠릅니다.

단, Singleton으로 설정되었기 때문에 자바 소스를 편집하여 사용하는 모델의 경우 클래스의 멤버 필드인 Variable 사용에 주의하여야 합니다. Thread간 멤버 필드인 Variable이 공유되어 사용되어지기 때문입니다.

Singleton 설정 방법은 X-UP 서버의 `xup_config.xml` 파일에서 ‘enableModelSingleton’ 프로퍼티 값으로 설정할 수 있습니다.

- true : Singleton 으로 클래스 생성
- false : Non Singleton으로 클래스 생성

6.2 Automation Model

Automation 모델은 GUI 에디터를 이용하여 하나의 서비스에 해당하는 로직을 다이어그램으로 표현합니다. 표현된 각각의 다이어그램은 Invoke라고 하며 임의의 데이터 소스로부터 데이터를 획득합니다. 데이터를 획득하기 위한 데이터 정보는 X-UP Builder의 에디터와 Properties View를 이용하여 작성하게 됩니다. 그리고 Automation 모델을 저장하게 되면 java 클래스 코드가 자동으로 생성이 되며 GUI에디터에서 모델을 수정하게 되면 자동으로 클래스 코드에 반영되게 됩니다.

Automation 모델에서 X-UP Builder는 다음 아홉가지의 Invoke를 제공합니다.

- SAP RFC Invoke : SAP RFC를 위한 모듈
- Select Invoke : DBMS select를 위한 모듈
- Modify Invoke : DBMS insert, update, delete를 위한 모듈
- Procedure Invoke : DBMS procedure를 위한 모듈
- OpenApi Invoke : Open Api를 위한 모듈
- WebService Invoke : WEB service를 위한 모듈
- UDDI Invoke : UDDI를 위한 모듈
- X-UP Invoke : Remote 에 존재하는 X-UP의 모델 invoking을 위한 모듈
- Model Invoke : X-UP의 모델 invoking을 위한 모듈

또한 다음 여섯가지의 function 기능을 제공합니다.

- Method : 에디터에서의 표현하지 못하는 자바소스 코드를 작성할 수 있습니다.
- Merge : 두개 이상의 데이터셋을 하나의 데이터셋으로 융합할 수 있습니다.
- XML Parser : 에디터에서의 특정 XML 을 쉽게 파싱 할 수 있습니다.

- ExtDataSet : 여러 개의 Variable을 합쳐 하나의 DataSet으로 만들수 있습니다.
- ExtVariable : DataSet의 컬럼을 분리하여 하나의 Variable로 추출할 수 있습니다.
- DataSetRowLoop : DataSet의 특정 Row값에 대하여 반복적 제어문 처리를 할 수 있습니다.

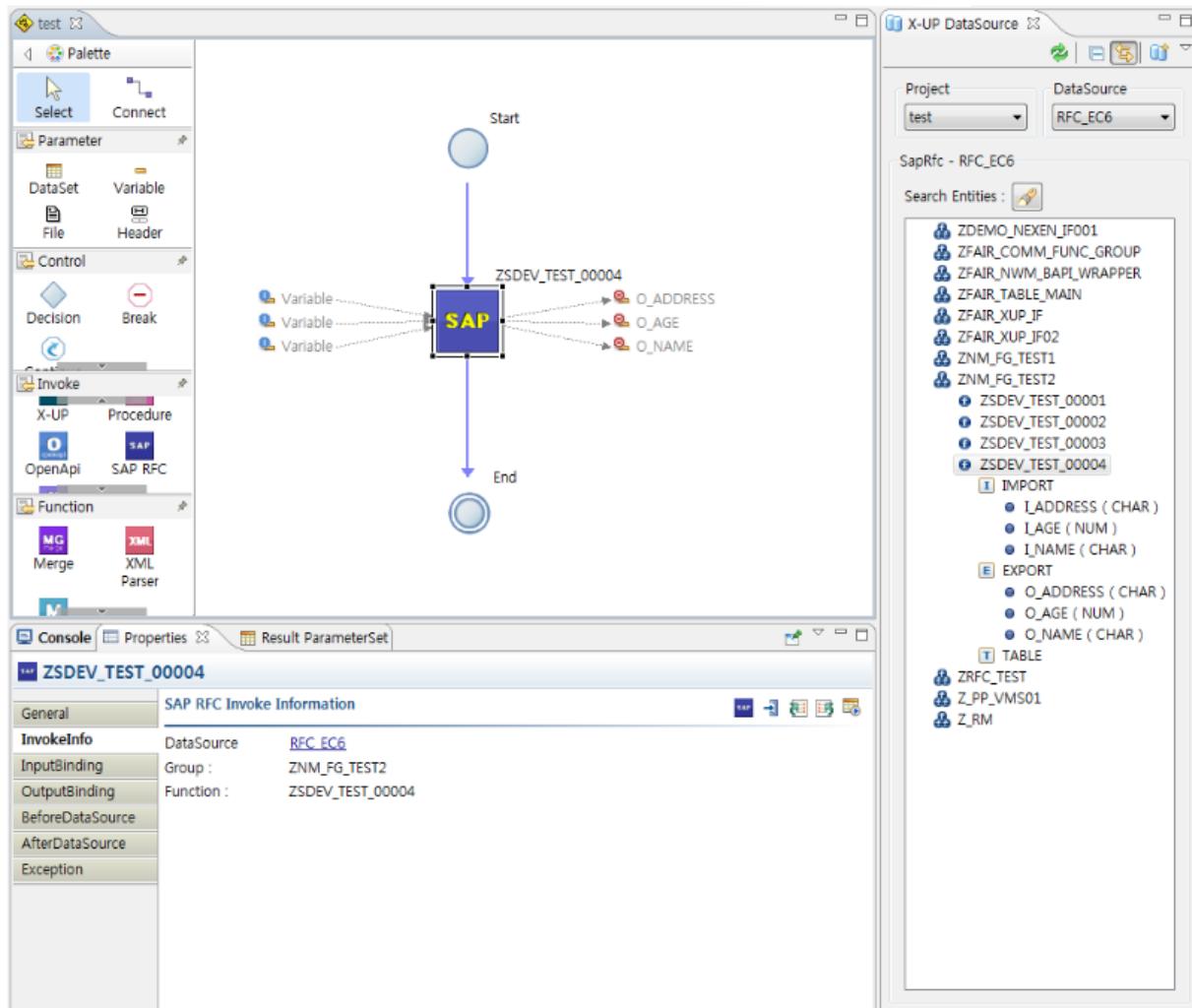
또한 다음 세가지의 분기 처리에 대한 Control 기능을 제공합니다.

- Decision : 에디터에서의 특정 조건에 따라 분기 처리를 할 수 있습니다.(if)
- Break : 에디터에서의 반복적인 작업을 중단 할 수 있습니다.(break)
- Continue : 에디터에서 특정 조건에서 반복적인 작업을 한번 건너뛸 수 있습니다.(for, continue)

SAP RFC Invoke

SAP RFC Invoke는 SAP RFC 정보를 파라메터형식으로 생성하기 위한 컴포넌트입니다. 호출 시 Variable or Database 형식으로 가져옵니다.

SAP RFC Invoke 생성은 Palette에서 SAP RFC 아이템을 선택하거나 X-UP DataSource View에서 SAP DataSource를 선택한 후 나오는 평선을 조회하여 특정 평선을 선택후 에디터로 드래그함으로써 빠르게 생성할 수 있습니다.

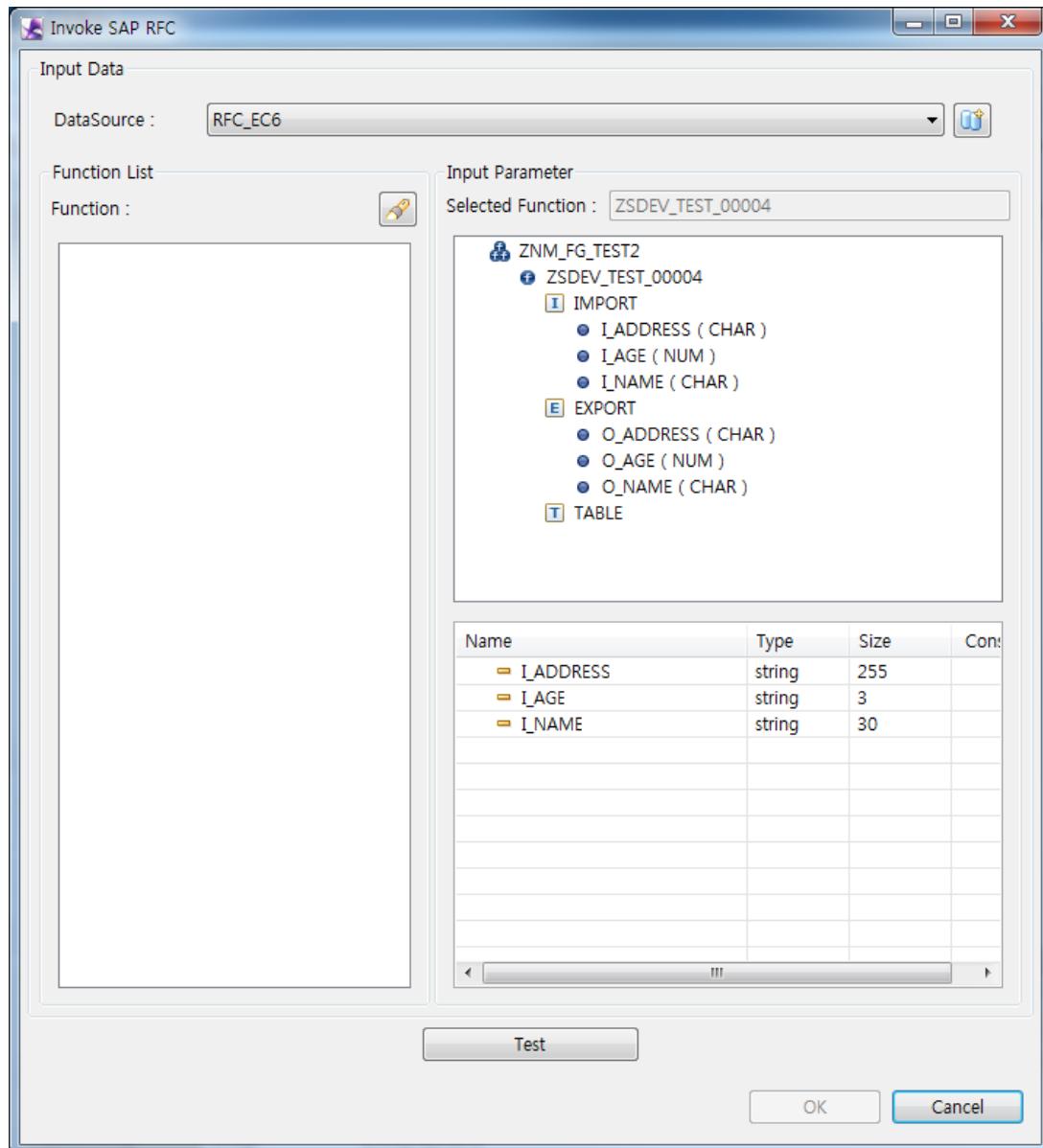


SAP RFC Invoke는 인보크를 생성하기 위한 SAP RFC Invoke Dialog 와 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General
- InvokeInfo
- InputBinding
- OutputBinding
- BeforeDataSource
- AfterDataSource
- Exception

SAP RFC Invoke Dialog

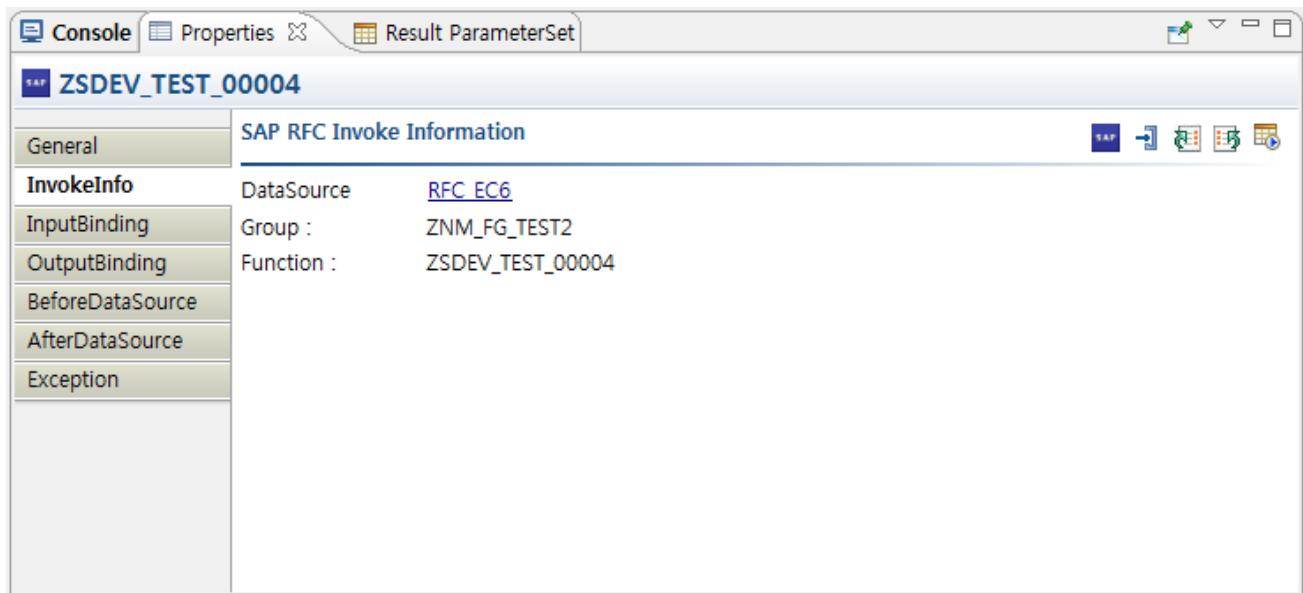
SAP RFC Invoke를 생성하고자 할 경우 다음과 같은 다이얼로그가 출력됩니다.



- DataSource : SAP RFC 데이터소스를 선택 (없다면 새로운 데이터소스 생성 가능)
- Function List : 선택한 데이터소스로부터 평면 리스트를 조회
- Input Parameter : 선택한 평면으로부터 입력 파라미터가 자동 생성됨
- Test :
 - 테스트를 수행한 결과를 출력
 - Output Variable 형태로 생성됩니다.

InvokeInfo

인보크를 수행하기 위한 필수 정보가 출력됩니다.



다른 Invoke처럼 인보크를 실행하기 위한 In-Out 정보가 이미 고정되어 있는 경우는 Information만 출력합니다.

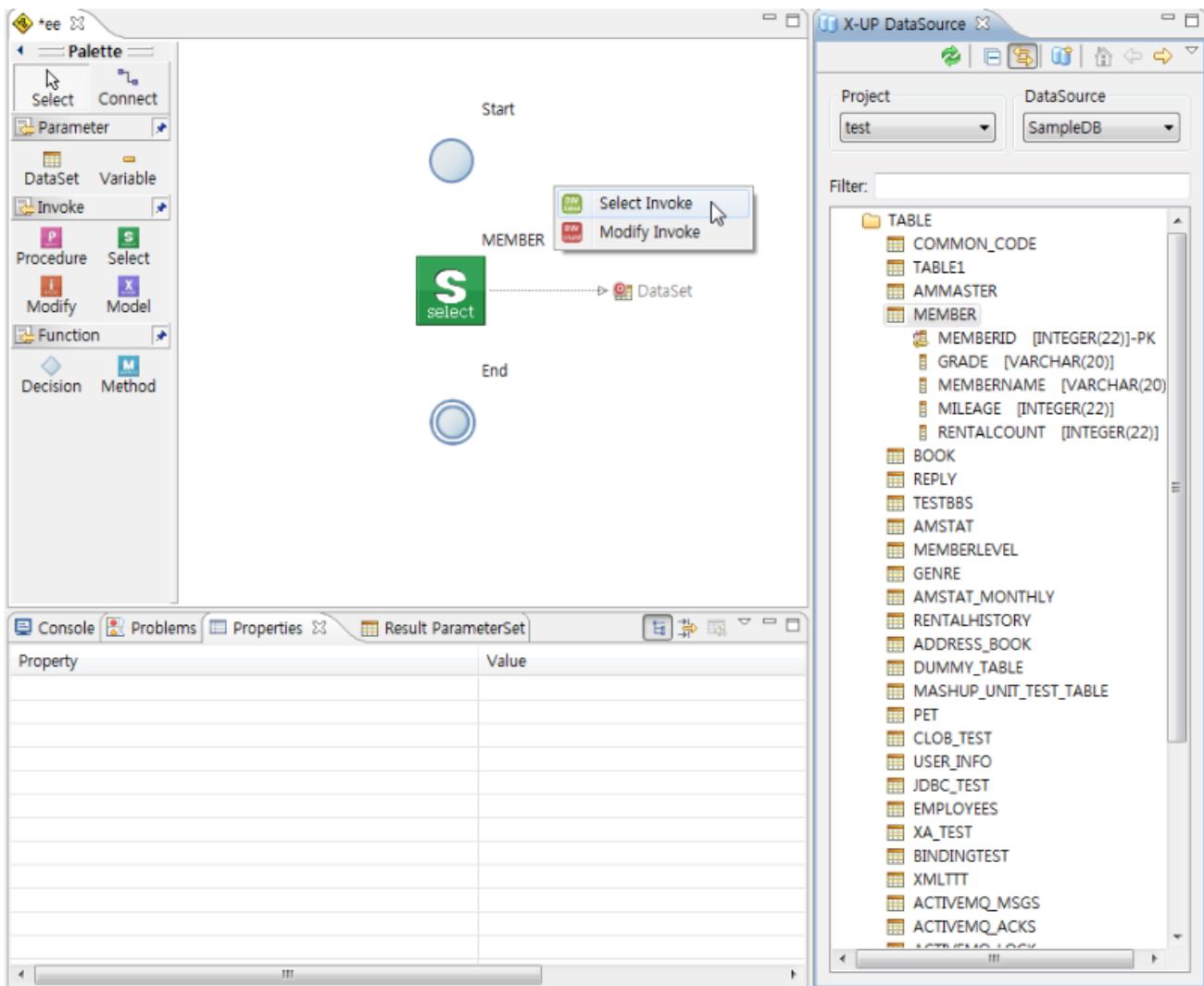
SAP RFC Invoke의 경우 InvokeInfo에서 다음 정보를 출력합니다.

- DataSource : 선택한 데이터소스 (클릭시 해당 데이터소스 편집창 오픈)
- Group : 정의된 SAP RFC 함수 그룹
- Function : 선택한 SAP RFC 함수

Select Invoke

Select Invoke는 DataBase의 Select 기능을 수행합니다.

X-UP Builder에서 Select Invoke 노드 생성은 Palette에서 Select 아이템을 선택하거나 X-UP DataSource View에서 테이블을 선택후 에디터로 드래그하여 Select Invoke를 선택함으로써 빠르게 생성할 수 있습니다.



Select Invoke의 프로퍼티에는 다음과 같은 속성탭을 가지고 있습니다.

- General : 일반적인 정보 정의
- InvokerInfo : 인보크 수행을 위한 필수 정보 정의
- BeforeDataSource : 데이터 소스 호출 전 수행 정의
- AfterDataSource : 데이터 소스 호출 후 수행 정의
- Exception : 예외처리

SQL입력

X-UP은 데이터를 조회하기 위한 SQL문을 자동으로 생성해 줍니다. 생성된 SQL문은 모든 컬럼을 where 조건 없이 조회하는 select sql 입니다. 필요에 따라 자동적으로 생성된 sql을 수정하여 임의의 데이터를 조회할 수 있습니다. SQL의 바인딩변수는 매개변수(Parameter) 이름 앞과 뒤에 '#'을 붙여서 표현합니다.

DBMS별 Data Type Mapping

Java	Platform	Oracle 10g	Mysql 5.0	Mssql 2008	DB2 9.0
String	STRING	CHAR VARCHAR VARCHAR2 NCHAR NVARCHAR2 XML	CHAR VARCHAR	char nchar varchar nvarchar	CHAR VARCHAR GRAPHIC VARGRAPHIC
Boolean	BOOLEAN		BIT BOOL BOOLEAN	bit	
BigDecimal	BIGDECIMAL	NUMBER(p) INTEGER INT SMALLINT DECIMAL DEC NUMERIC	DECIMAL	numeric decimal money samllmoney	DECIMAL NUMERIC
int	INTEGER		INTEGER MEDIUMINT TINYINT	tinyint int smallint	INTEGER SMALLINT
float	FLOAT	NUMBER(p,s) BINARY_FLOAT	FLOAT	real	REAL
double	DOUBLE	DOUBLE PRECISION FLOAT BINARY_DOUBLE	DOUBLE PRECISION DOUBLE	float	DOUBLE
long	LONG	LONG	BIGINT	bigint	BIGINT
Date	DATE	DATE	DATE YEAR	date	DATE
Time	TIME		TIME	time	TIME
Timestamp	DATETIME	TIMESTAMP	TIMESTAMP DATETIME	datetime samlldatetime datetime2 datetimeoffset	TIMESTAMP
Blob	BLOB	BLOB	BLOB TINYBLOB MEDIUMBLOB LONGBLOB	image	BLOB
Clob		CLOB NCLOB	TEXT LONGTEXT	text ntext	CLOB DBCLOB
byte[]				timestamp	
XML				XML	XML

CDATA 지원

SQL 문이 특수문자를 사용하고 있을 경우 ‘Enable Surround CDATA’를 선택하게 되면 SQL 문 전체를 CDATA로 감싸게 됩니다.



CDATA는 무엇인가?

XML이나 자바스크립트에서 사용하는 CDATA 섹션입니다. CDATA를 사용하는 목적은 XML코드를 작성한 후, XML파서에게 ‘CDATA섹션 안에 포함된 코드에는 태그가 없다. 그러니 무시하고 지나가라’라고 알려주는 역할을 합니다.

흔히 사용하는 "&",">","<" 같은 것들을 XML코드에 작성할 경우엔 XML파서가 태그로 인식하여 그대로 해석해 버리기 때문에 오류가 발생합니다. 이런 경우를 방지하기 위하여 CDATA섹션으로 특수문자가 포함된 해당 코드를 감싸주는 것입니다. 이렇게 하게 되면 XML파서가 태그가 아닌 String 자체로 인식하게 됩니다.

First Row 기능

만일 SQL에서 조회 된 데이터 양이 많아 클라이언트가 대기 해야 하는 경우 X-API에서 제공하고 있는 First Row 기능을 활용할 수 있습니다.

정의 된 RowCount 를 초과 할 경우 요청에 대한 HttpResponse 을 통해 응답을 보내기 전
com.tobesoft.xplatform.tx.HttpPartPlatformResponse 통하여 일부의 데이터를 미리 보낼 수 있습니다.



First Row는 무엇인가?

해당 First Row는 (주)투비소프트의 X-API 서버의 First Row 기능을 말합니다. X-API의 First Row란 대량의 데이터를 호출할 경우 성능 및 오류방지를 위하여 사용자가 지정한 First Row 수 만큼 분할하여 뿌려주는 기능입니다. 예를 들면 1000건의 데이터를 First Row를 이용하여 200개씩 나눠서 5회에 걸쳐 가져오거나, 500개씩 나눠서 2회에 걸쳐 가져옵니다.

사용자가 설정한 First Row 수 만큼 계속 뿌려주고 남은 나머지는 한꺼번에 마지막에 뿌려줍니다. 예를 들어 1000건의 데이터를 300개씩 First Row를 이용해서 가져오게 되면 3회에 걸쳐 300개씩 뿌려주고 나머지 100개는 마지막에 뿌려 총 4회에 걸쳐 1000건의 데이터를 가져옵니다.

Dynamic SQL 지원

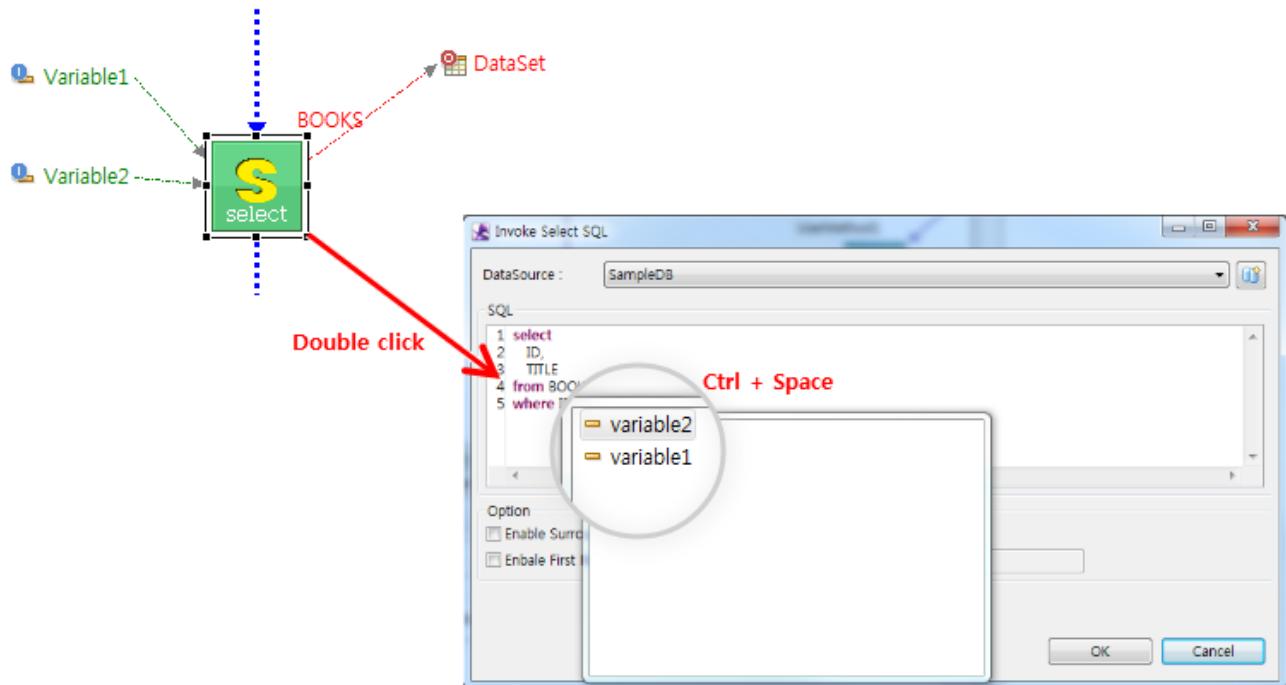
iBatis는 SQL의 재사용성과 유연성을 향상시키기 위해 매핑 구문 내에서 사용할 수 있는 동적 SQL (Dynamic SQL) 요소들을 제공합니다.

X-UP은 iBatis를 지원하고 있기 때문에 기존의 iBatis의 동적 SQL쿼리도 쿼리수정 없이 바로 사용할 수 있습니다.

다양한 동적 SQL 요소의 자세한 안내는 [부록 A. Dynamic Sql](#)을 참조합니다.

InvokeInfo

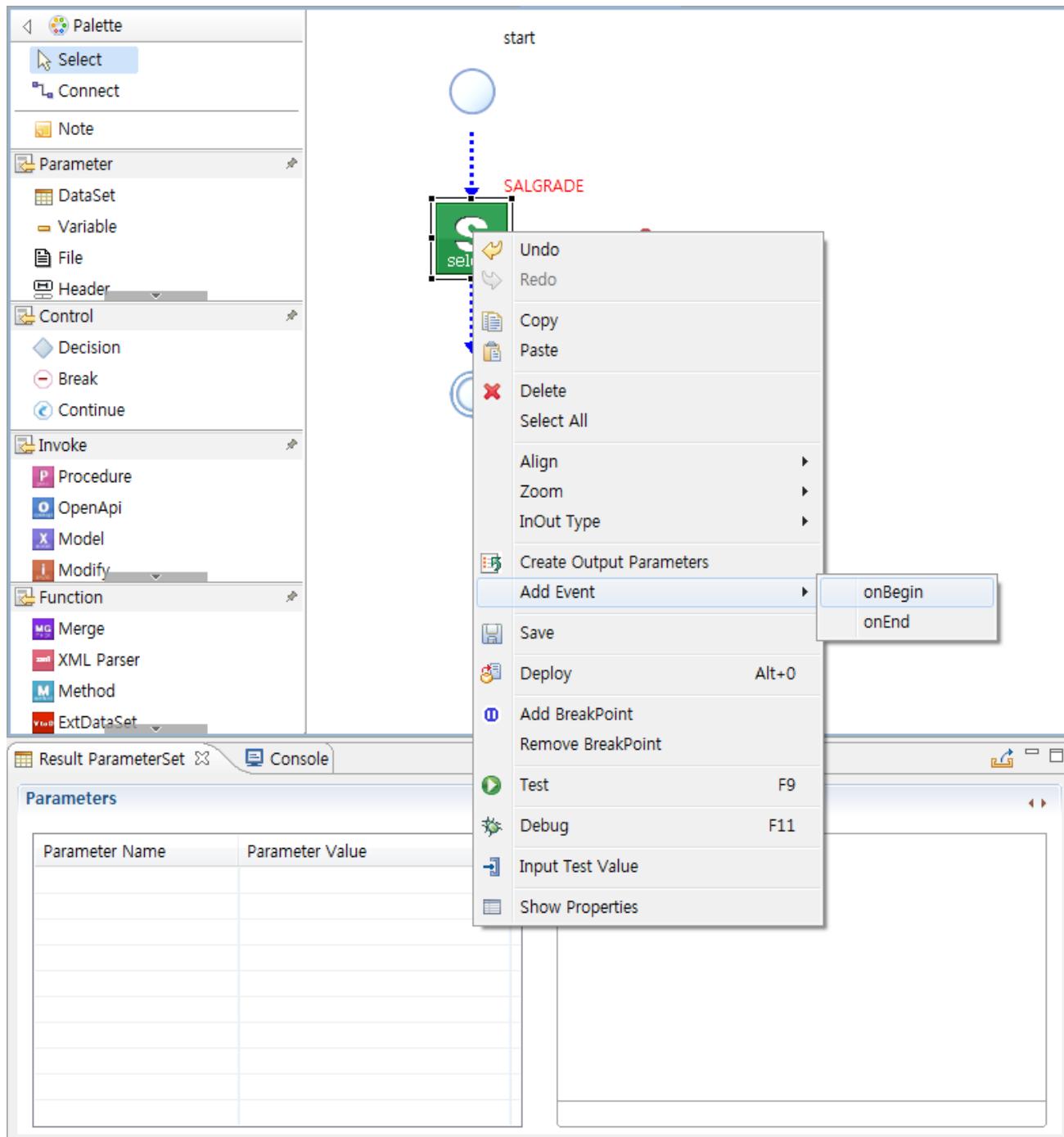
모든 인보크 노드는 반드시 InvokeInfo라는 프로퍼티탭을 가지고 있습니다. InvokeInfo는 인보크를 하기 위한 필수 정보를 입력합니다. Select Invoke에서는 데이터소스명과 sql 을 입력하게 됩니다.



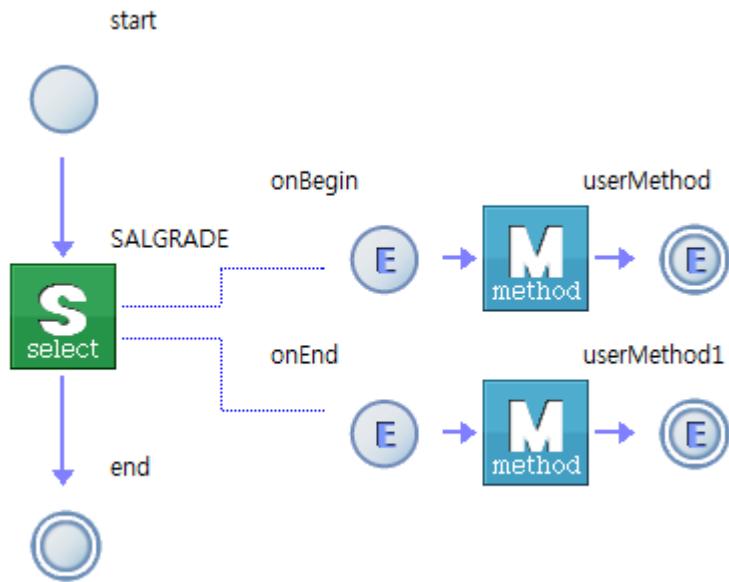
에디터에서 해당 인보크에 input parameter를 연결하면 sql 편집창에서 파라메터 어시스트를 사용할 수 있습니다.
파라메터 어시스트는 Ctrl+Space 또는 '#' 을 입력하면 자동으로 출력됩니다.

Event Handler

Event handler는 Select Invoke와 Modify Invoke에서 제공하며, 인보크 작업중 사용자가 원하는 추가 로직을 이벤트 내부에 정의할 수 있습니다.



이벤트 핸들러를 추가하고자 할 경우 해당 노드를 선택한 후 오른쪽 메뉴에서 Add Event 를 선택 후 추가하고자 하는 이벤트를 선택하면 해당 노드에 이벤트시작과 종료 노드가 추가됩니다.



Select Invoke의 이벤트 핸들러는 다음과 같습니다.

- **onBegin** : Invoke의 구동 바로 직전에 호출됩니다.
- **onEnd** : Invoke 구동 후 생성된 ParameterSet을 반환하기 직전에 호출됩니다.

Modify Invoke

Modify Invoke는 입력 파라미터를 가지고 insert, update, delete sql을 수행합니다.

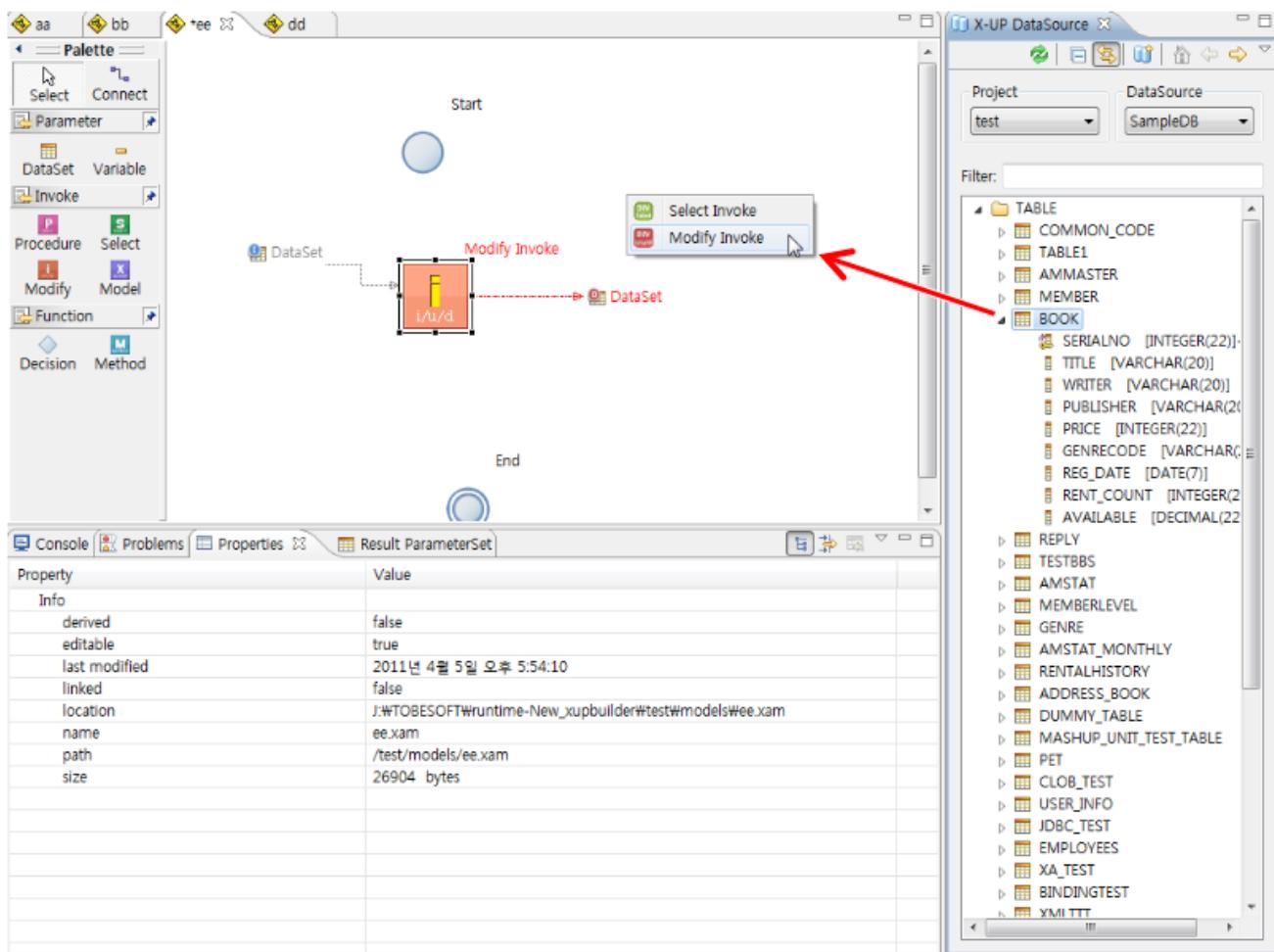
X-UP은 Modify Invoke 시 데이터셋의 Row의 Type을 보고 해당 Row를 insert, update, delete할지를 결정합니다. 데이터셋의 Row의 Type은 강제로 설정할 수도 있으나 일반적으로 데이터셋에 Row를 추가하거나 수정, 삭제할 때 자동적으로 Type이 결정됩니다.

X-UP이 행하는 Row에 대한 write의 판단 기준은 다음과 같습니다.

- DataSet에 존재하는 Row의 값을 테이블의 record에 그대로 반영한다.
- DataSet에 존재하지 않는 Row에 해당하는 테이블의 record는 신경 쓰지 않는다.
- DataSet에서 삭제한 Row에 해당하는 테이블의 record는 delete한다.

이러한 기준에 따라서 데이터셋의 특정 Row가 테이블에 존재하면 update를 하고, 존재하지 않으면 insert합니다. delete되는 대상은 데이터셋에서 removeRow()나 clearData()를 호출하여 삭제된 Row에 해당하는 record입니다.

X-UP Builder에서 Modify Invoke 노드 생성은 Palette에서 Modify 아이템을 선택하거나 X-UP DataSource View에서 테이블을 선택후 에디터로 드래그하여 Modify Invoke를 선택함으로써 빠르게 생성할 수 있습니다.



Modify Invoke의 프로퍼티에는 다음과 같은 속성탭을 가지고 있습니다.

- General
- InvokeInfo
- SqlTypeBinding
- Filtering
- Default Value
- BeforeDataSource
- AfterDataSource
- Exception

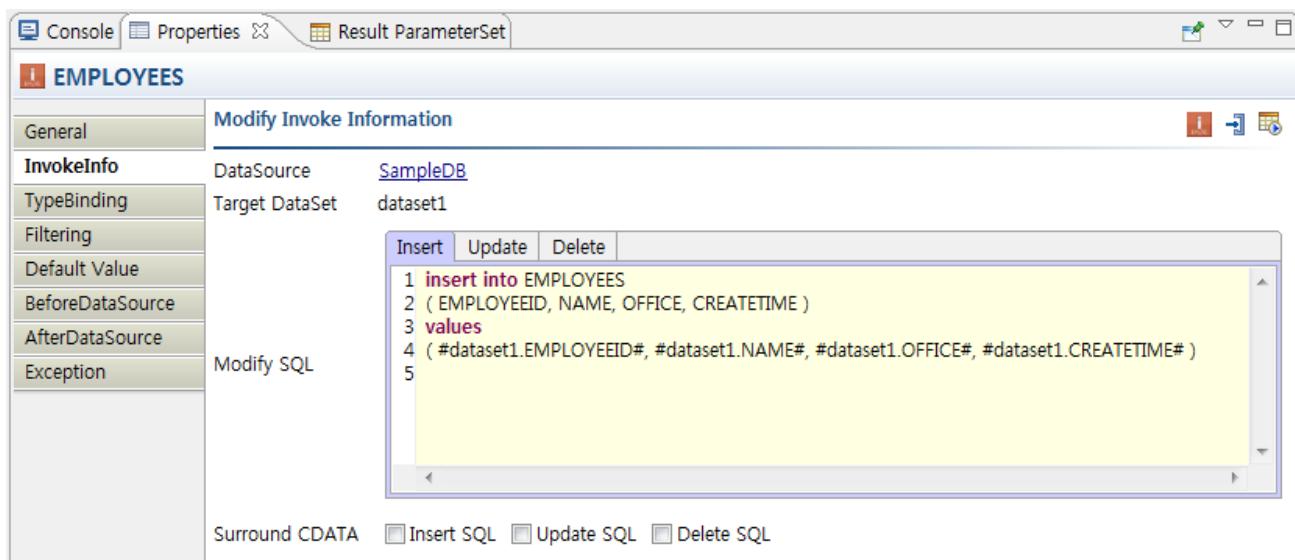
SQL 입력

Modify Invoke는 insert, update, delete 를 위한 SQL 문을 자동으로 생성해 줍니다.

- insert : DB 스키마에 대한 모든 컬럼에 대한 insert query를 생성합니다.
- update : DB 스키마에 PrimaryKey가 설정되어 있는 컬럼을 where 조건으로 update query를 생성합니다.
- delete : DB 스키마에 PrimaryKey가 설정되어 있는 컬럼을 where 조건으로 delete query를 생성합니다.

InvokeInfo

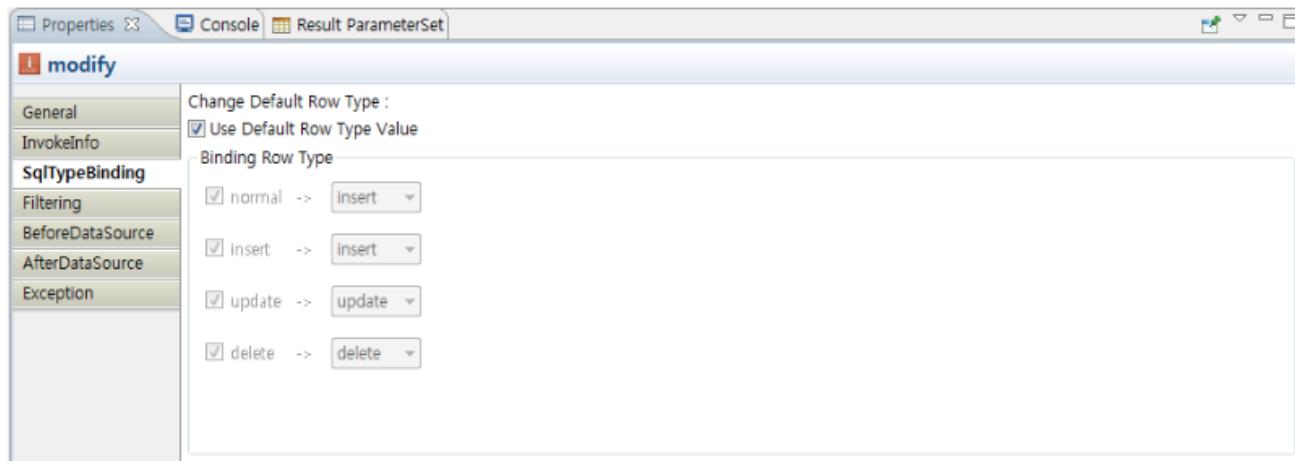
InvokeInfo는 Modify Invoke를 수행하기 위한 필수 정보를 입력합니다.



- DataSource : 데이터소스를 선택
- Target DataSet : sql실행의 타겟이 되는 데이터셋을 선택
- Modify SQL : insert / update / delete SQL 을 정의, SQL을 작성하는 방법은 Select Invoke와 동일합니다.

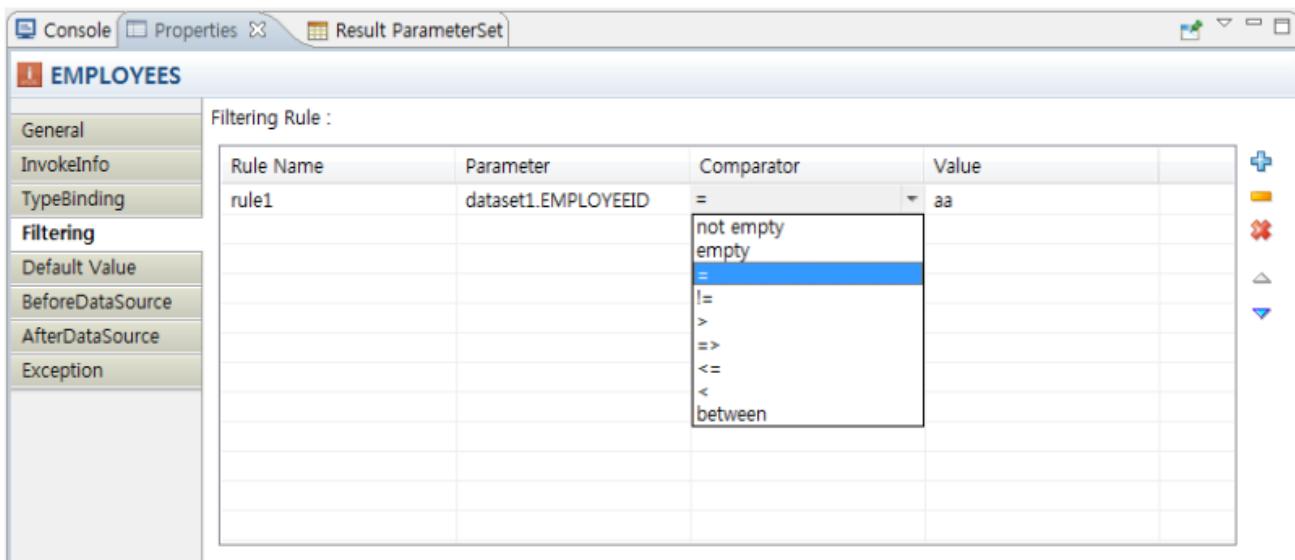
SqlTypeBinding

SqlTypeBinding은 데이터셋의 Row 타입을 바인딩시켜줍니다.



Filtering

Filtering은 입력 파라메터의 값이 특정 조건에 만족할 경우 Modify 액션에서 제외하고자 할 경우 정의합니다.



- Rule Name : Filtering rule 이름 정의
- Parameter : 입력 파라메터를 선택
- Comparator : 조건문 선택
- Value : 조건값 입력

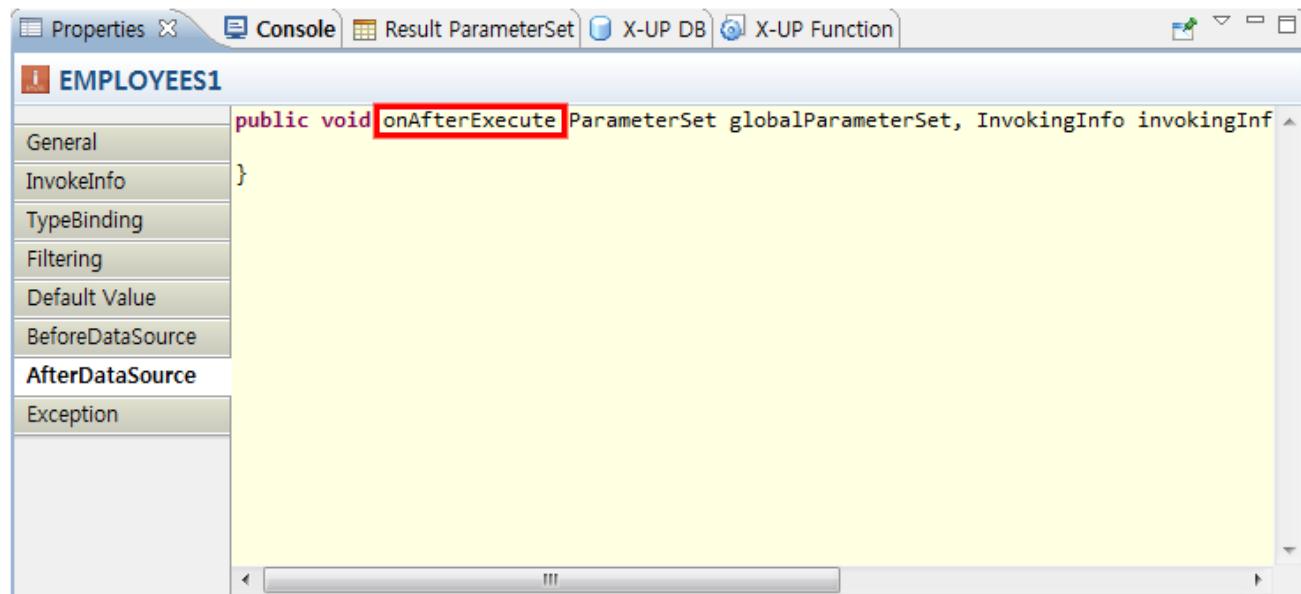
BeforeDataSource

BeforeDataSource탭은 해당 인보크 실행 바로 전에 소스를 수행하는 단계로 사용자가 원하는 로직을 자바 코드로 추가할 수 있습니다.



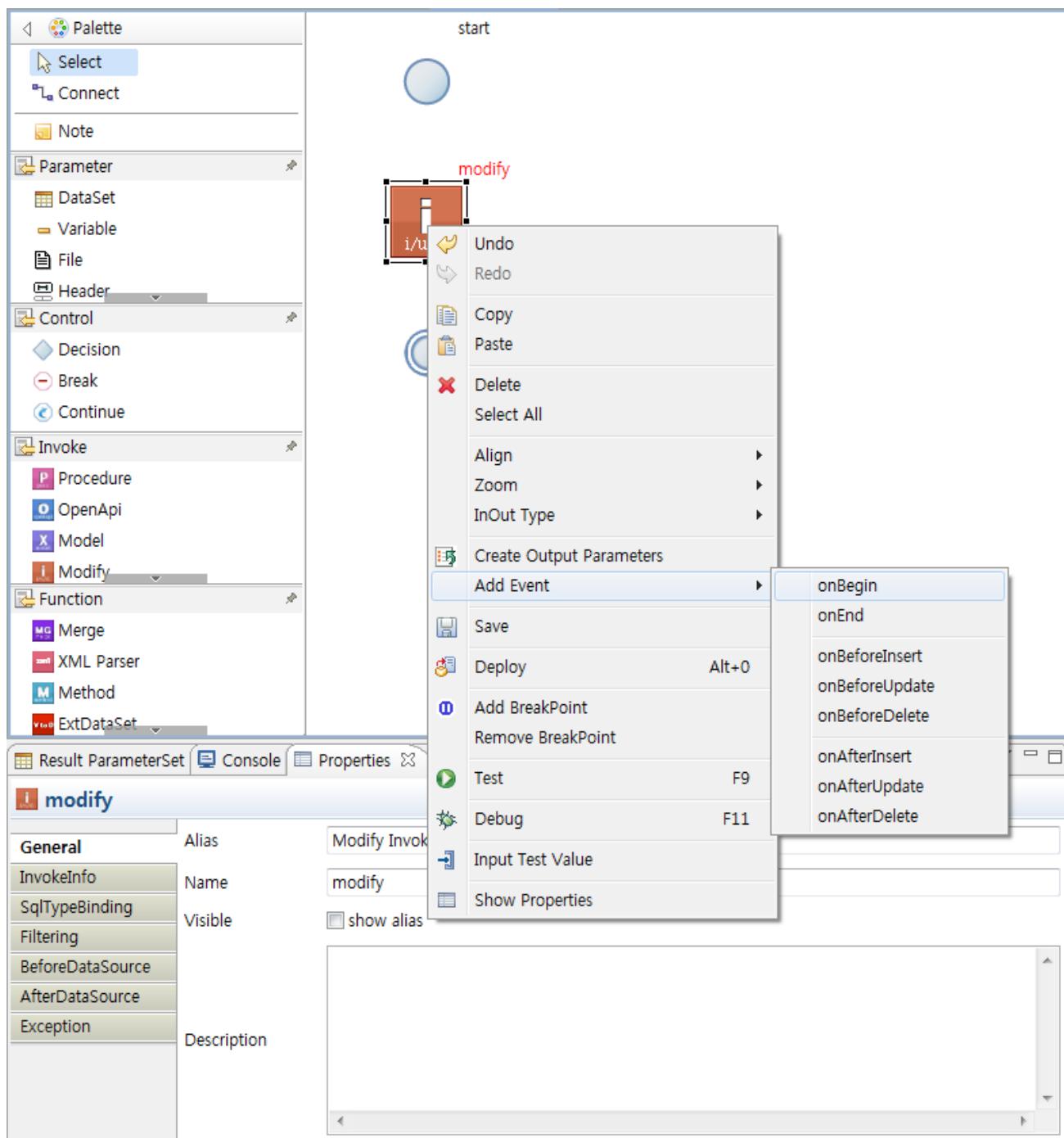
AfterDataSource

AfterDataSource탭은 해당 인보크 실행 바로 후에 소스를 수행하는 단계로 사용자가 원하는 로직을 자바 코드로 추가할 수 있습니다



Event Handler

- onBegin : Invoke의 구동 바로 직전에 호출됩니다.
- onEnd : Invoke 구동 후 생성된 ParameterSet을 반환하기 직전에 호출됩니다.
- onBeforeInsert: 데이터셋의 행의 RowType 이 DataSet.ROW_TYPE_INSERTED 일 경우 insert문을 수행하기 직전에 호출 됩니다.
- onAfterInsert: 데이터셋의 행의 RowType 이 DataSet.ROW_TYPE_INSERTED 일 경우 insert문을 수행한 직후에 호출 됩니다.
- onBeforeUpdate: 데이터셋의 행의 RowType 이 DataSet.ROW_TYPE_UPDATED 일 경우 update문을 수행하기 직전에 호출 됩니다.
- onAfterUpdate: 데이터셋의 행의 RowType 이 DataSet.ROW_TYPE_UPDATED 일 경우 update문을 수행한 직후에 호출 됩니다.
- onBeforeDelete: 데이터셋의 행의 RowType 이 DataSet.ROW_TYPE_DELETED 일 경우 delete문을 수행하기 직전에 호출 됩니다.
- onAfterDelete: 데이터셋의 행의 RowType 이 DataSet.ROW_TYPE_DELETED 일 경우 delete문을 수행한 직후에 호출 됩니다.



Exception

```

public void onExceptionOccured(ParameterSet globalParameterSet, InvokingInfo invokingInfo)
{
    throw new AutomationFailException(e.getMessage(), e);
}

```

CDATA 지원

SQL 문이 특수문자를 사용하고 있을 경우 'Enable Surround CDATA'를 선택하게 되면 SQL 문 전체를 CDATA로 감싸게 됩니다.



CDATA 는 무엇인가?

XML이나 자바스크립트에서 사용하는 CDATA 섹션입니다. CDATA를 사용하는 목적은 XML코드를 작성한 후, XML파서에게 'CDATA섹션 안에 포함된 코드에는 태그가 없다. 그러니 무시하고 지나가라' 라고 알려주는 역할을 합니다.

흔히 사용하는 "&",">","<" 같은 것들을 XML코드에 작성할 경우엔 XML파서가 태그로 인식하여 그대로 해석해 버리기 때문에 오류가 발생합니다. 이런 경우를 방지하기 위하여 CDATA섹션으로 특수문자가 포함된 해당 코드를 감싸주는 것입니다. 이렇게 하게 되면 XML파서가 태그가 아닌 String 자체로 인식하게 됩니다.

Dynamic SQL 지원

iBatis는 SQL의 재사용성과 유연성을 향상시키기 위해 매핑 구문 내에서 사용할 수 있는 동적 SQL (Dynamic SQL) 요소들을 제공합니다.

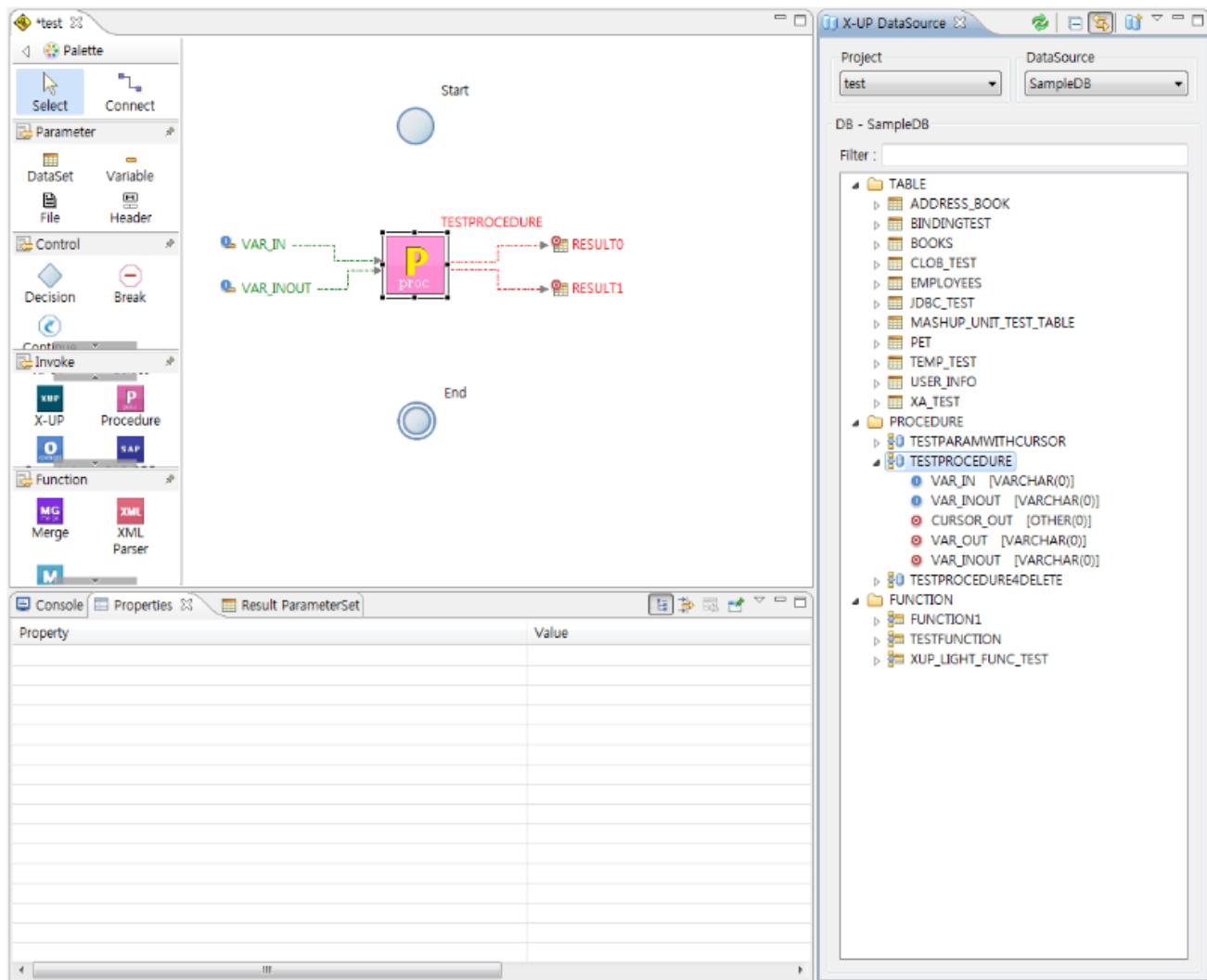
X-UP은 iBatis를 지원하고 있기 때문에 기존의 iBatis의 동적 SQL쿼리도 문법수정 없이 바로 사용할 수 있습니다.

다양한 동적 SQL 요소의 자세한 안내는 [부록 A. Dynamic Sql](#)을 참조합니다.

Procedure Invoke

Procedure Invoke는 데이터베이스의 procedure 및 function 을 호출합니다. 해당 Invoke는 DataSource와 Entity 정보를 입력하여 테스트를 수행하고 나면 실제 데이터베이스의 Procedure 및function 을 호출 한 결과를 바탕으로 자동으로 입력 파라메터와 출력 파라메터가 설정 됩니다.

X-UP Builder에서 Procedure Invoke 생성은 Palette에서 Procedure 아이템을 선택하거나 X-UP DataSource View에서 Procedure나 Function을 선택후 에디터로 드래그함으로써 빠르게 생성할 수 있습니다.



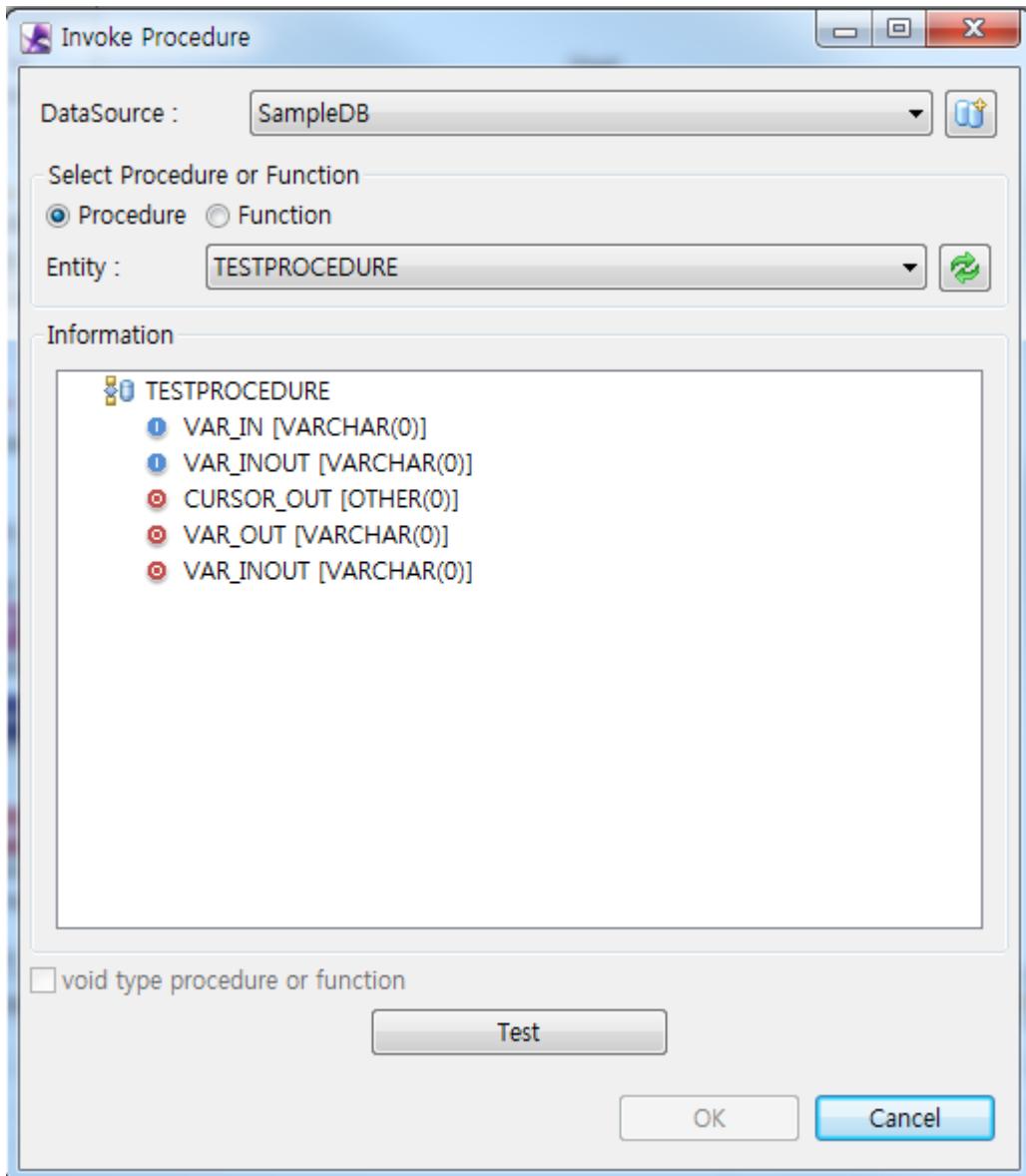
Procedure Invoke는 인보크를 생성하기 위한 Procedure Invoke Dialog 와 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General
- InvokeInfo
- InputBinding
- OutputBinding
- BeforeDataSource

- AfterDataSource
- Exception

Procedure Invoke Dialog

Procedure Invoke를 생성하고자 할 경우 다음과 같은 다이얼로그가 출력됩니다.



- DataSource
 - DataSource : 데이터소스를 선택 (없다면 새로운 데이터소스 생성 가능)
- Select Procedure or Function :
 - Procedure 또는 Function을 선택
 - Entity : 원하는 엔티티 선택
- Information :
 - 선택된 엔티티의 in-out 정보가 출력됩니다.

- 다만 output 정보는 실제와 다를 수 있으므로 반드시 Test 를 통해 실행 결과를 확인합니다.
- void type procedure or function
 - 만약 실행 결과가 없을 경우 이를 void 타입 인보크로 설정할 수 있습니다.
 - 이럴경우 이 인보크는 결과값을 리턴하지 않는 상태로 생성됩니다.
- Test :
 - 테스트를 수행한 결과를 출력합니다.
 - Value에 Test 값을 넣어주면 Test Result 가 나옵니다.

First Row 기능

만일 SQL에서 조회 된 데이터 양이 많아 클라이언트가 대기 해야 하는 경우 X-API 에서 제공 하고 있는 First Row 기능을 활용할 수 있습니다.

정의 된 RowCount 를 초과 할 경우 요청에 대한 HttpResponse 을 통해 응답을 보내기 전

com.tobesoft.xplatform.tx.HttpPartPlatformResponse 통하여 일부의 데이터를 미리 보낼 수 있습니다.



First Row 는 무엇인가?

해당 First Row는 (주)투비소프트의 X-API 서버의 First Row 기능을 말합니다. X-API의 First Row란 대량의 데이터를 호출할 경우 성능 및 오류방지를 위하여 사용자가 지정한 First Row 수 만큼 분할하여 뿌려주는 기능입니다. 예를 들면 1000건의 데이터를 First Row를 이용하여 200개씩 나눠서 5회에 걸쳐 가져오거나, 500개씩 나눠서 2회에 걸쳐 가져옵니다.

사용자가 설정한 First Row 수 만큼 계속 뿌려주고 남은 나머지는 한꺼번에 마지막에 뿌려줍니다. 예를 들어 1000건의 데이터를 300개씩 First Row를 이용해서 가져오게 되면 3회에 걸쳐 300개씩 뿌려주고 나머지 100개는 마지막에 뿌려 총 4회에 걸쳐 1000건의 데이터를 가져옵니다.



Procedure Invoke에서 First Row 사용 방법

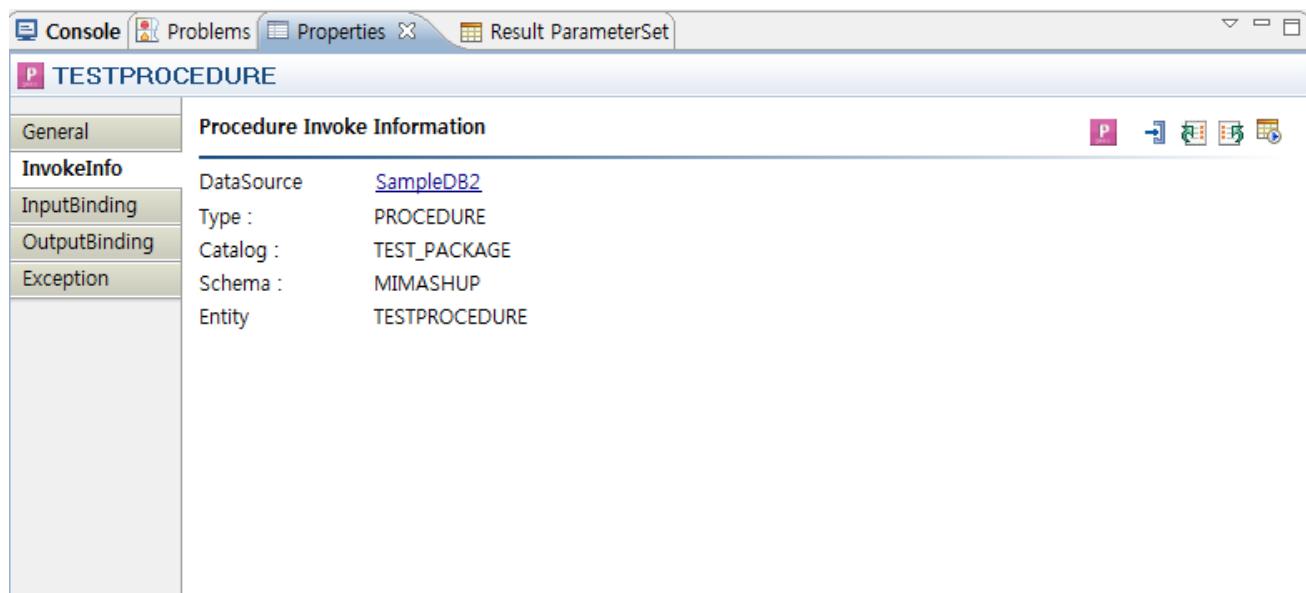
현재 X-UP Builder에서 개발 시에 Procedure Invoke에서 First Row를 설정할 수 있는 UI가 없습니다. 대신 logics 폴더의 (default package)아래에 모델 생성 시 자동 생성되는 BaseAutomationLocig.java 클래스의 DbProcedureInvokingInfo의 setFirstYN(), setFirstRowFireCount() 메서드로 설정할 수 있습니다.

```
DbProcedureInvokingInfo info = new DbProcedureInvokingInfo();
info.setDomainName("XupDemoPrj");
info.setDataSourceName("addressBookDB");
info.setCatalogName("PKGEMP");
info.setSchemaName("MIMASHUP");
info.setEntityName("SELECTEMP");
info.setEntityType("PROCEDURE");
info.setParameterSet(tmpParameterSet);
info.setResultNameSet(resultNameSet);
info.setProvisionalNameMap(provisionalNameMap);

info.setFirstYN(true);
info.setFirstRowFireCount(10);
```

InvokeInfo

인보크를 수행하기 위한 필수 정보가 출력됩니다.



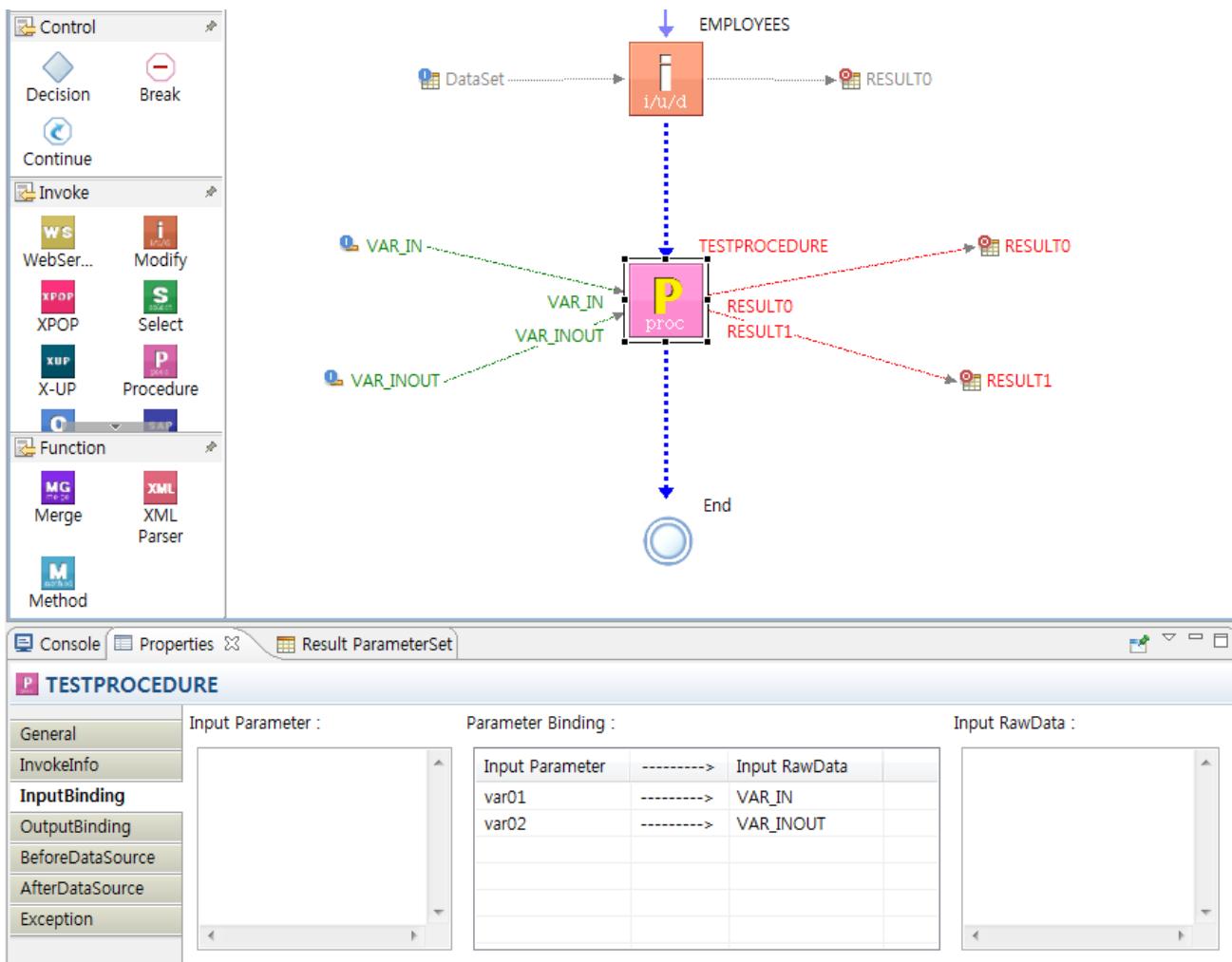
Procedure Invoke와 Model Invoke처럼 인보크를 실행하기 위한 In-Out 정보가 이미 고정되어 있는 경우는 Information 만 출력합니다.

Procedure Invoke의 경우 InvokeInfo에서 다음 정보를 출력합니다.

- DataSource : 선택한 데이터소스 (클릭시 해당 데이터소스 편집창 오픈)
- Type : PROCEDURE or FUNCTION
- Catalog : 선택한 카탈로그
- Schema : 선택한 스키마
- Entity : 선택한 엔티티

Input Binding / Output Binding

Input Binding / Output Binding은 in-out 파라메터의 이름이 고정된 인보크일 경우라도 사용자가 정의한 파라메터를 사용할 수 있도록 합니다.



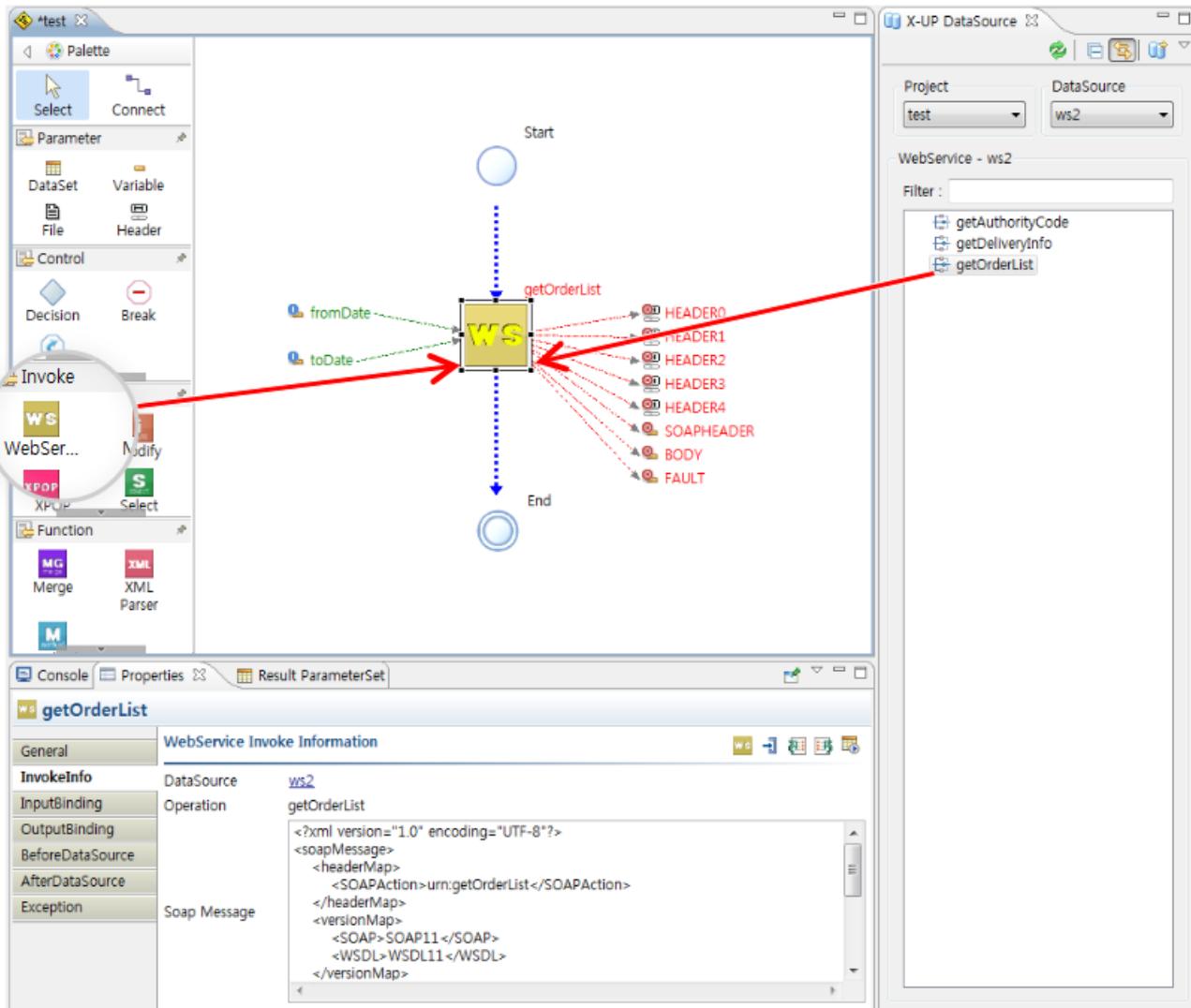
고정된 in-out 파라미터 이름과 상관없이 사용자가 원하는 파라미터 이름을 정의할 수 있습니다.

인보크와 함께 생성된 커넥션은 삭제할 수 없으며 만약 다른 파라미터와 연결하고자 한다면 reconnection을 통하여 변경할 수 있습니다.

WebService Invoke

WebService Invoke는 웹서비스 operation을 선택해 원하는 파라메터를 생성하기 위한 컴포넌트입니다. 호출시 webservice header, soap header, body, fault 정보를 파라메터 형식으로 가져옵니다.

X-UP Builder에서 WebService Invoke 생성은 Palette에서 WebService 아이템을 선택하거나 X-UP DataSource View에서 WebService DataSource 또는 UDDI DataSource를 선택후 특정 operation을 에디터로 드래그함으로써 빠르게 생성할 수 있습니다.



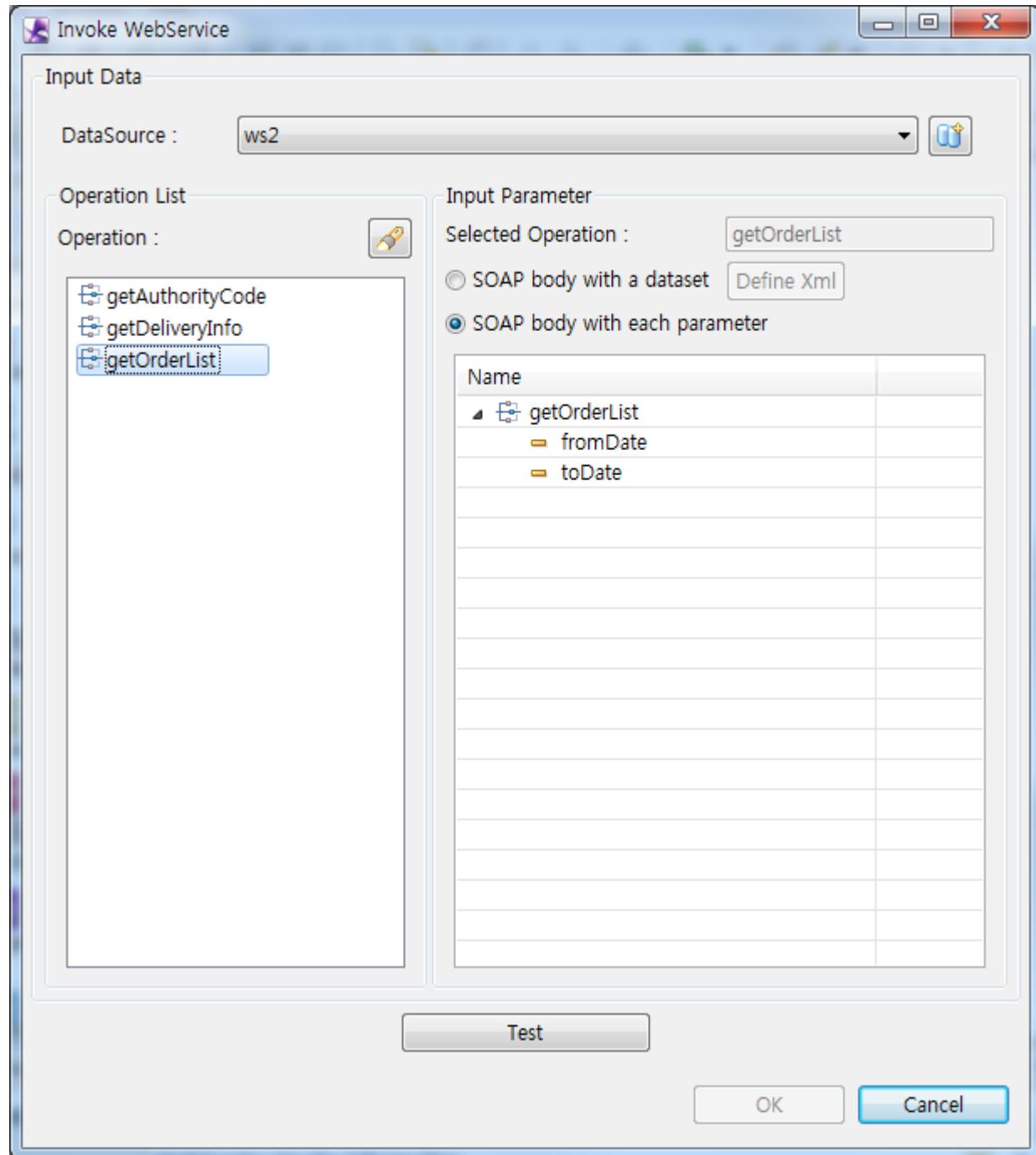
WebService Invoke는 인보크를 생성하기 위한 WebService Invoke Dialog 와 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General
- InvokeInfo
- InputBinding
- OutputBinding
- BeforeDataSource

- AfterDataSource
- Exception

WebService Invoke Dialog

WebService Invoke를 생성하고자 할 경우 다음과 같은ダイアログが 출력됩니다.

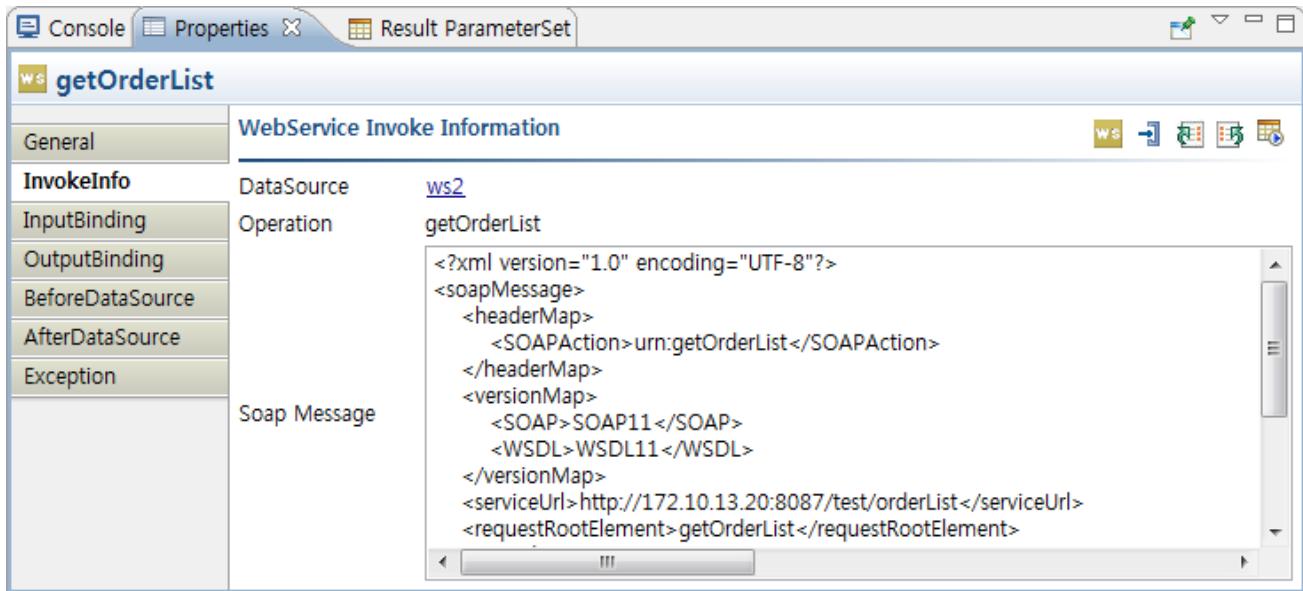


- DataSource : 웹서비스 데이터소스를 선택 (없다면 새로운 데이터소스 생성 가능)
- Operation : 오퍼레이션 리스트 조회
- Input Parameter:

- SOAP body with a dataset : input dataset으로 웹서비스 호출
- SOAP body with each parameter : input variable로 웹서비스 호출

InvokeInfo

인보크를 수행하기 위한 필수 정보가 출력됩니다.



다른 Invoke처럼 인보크를 실행하기 위한 In-Out 정보가 이미 고정되어 있는 경우는 Information 만 출력합니다.

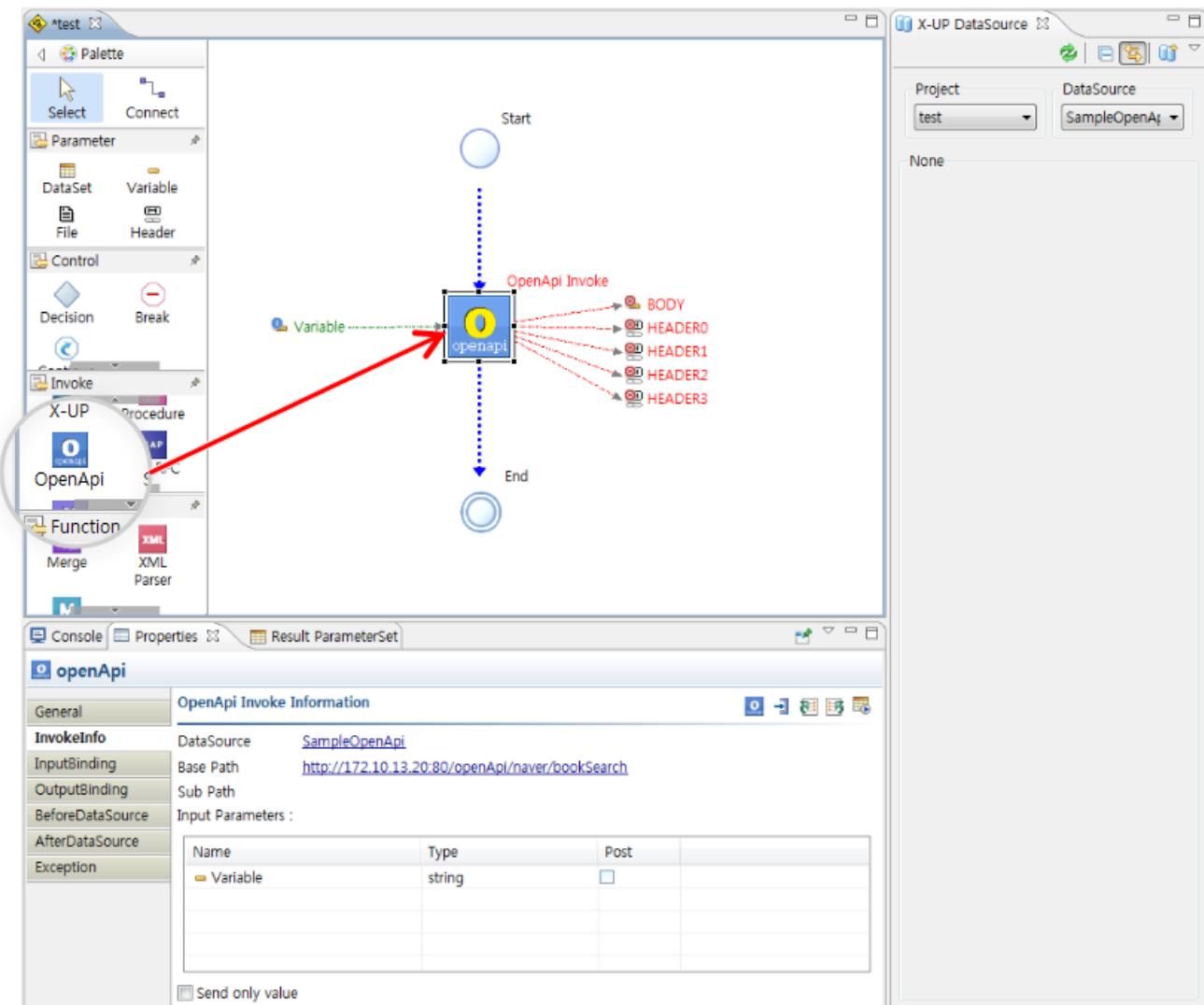
WebService Invoke의 경우 InvokeInfo에서 다음 정보를 출력합니다.

- DataSource : 선택한 데이터소스 (클릭시 해당 데이터소스 편집창 오픈)
- Operation : 선택한 오퍼레이션
- Soap Message : webservice soap message
- wsdl
- input parameter

OpenApi Invoke

OpenApi Invoke는 HTTP 정보를 파라메터형식으로 생성하기 위한 Invoke입니다. 호출 시 http header, html body 정보를 파라메터 형식으로 가져옵니다.

X-UP Builder에서 OpenApi Invoke 생성은 Palette에서 OpenApi 아이템을 선택후 에디터로 드래그함으로써 빠르게 생성할 수 있습니다.

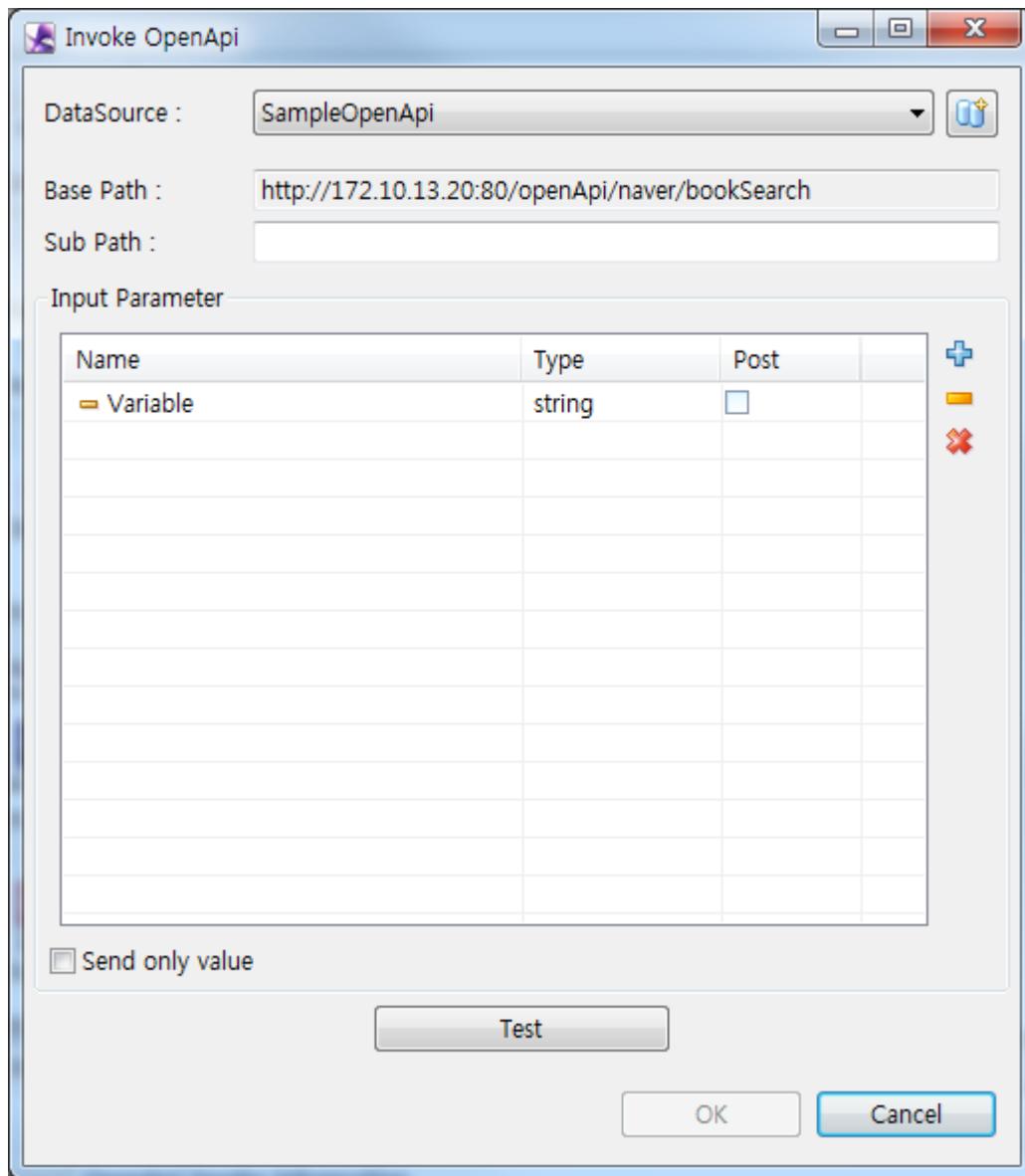


OpenApi Invoke는 인보크를 생성하기 위한 OpenApi Invoke Dialog 와 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General
- InvokeInfo
- InputBinding
- OutputBinding
- BeforeDataSource
- AfterDataSource
- Exception

OpenApi Invoke Dialog

OpenApi Invoke를 생성하고자 할 경우 다음과 같은 디아일로그가 출력됩니다.



- DataSource : OpenApi 데이터소스를 선택 (없다면 새로운 데이터소스 생성 가능)
- Base Path : 선택한 데이터소스에 정의된 URL
- Sub Path : Base path 뒤에 붙일 추가 URL 이 존재할 경우 입력
- Input Parameter : url 파라메터가 존재할 경우 추가
 - Post : 선택한 파라메터를 호출시 post방식으로 보내고자 할 경우 체크
- Send only value : 호출시 파라메터 value 전달

Sub Path와 Input Parameter

OpenApi로부터 데이터를 수집할 때 사용되는 전체 URL은 다음과 같이 구성됩니다.

<데이터소스 HTTP URL> + <SubPath> + <데이터소스 parameter> + <input parameter>

일반적으로 하나의 사이트에서 제공하는 OpenApi의 URL은 같은 base url을 사용하지만 제공하는 서비스에 따라 su

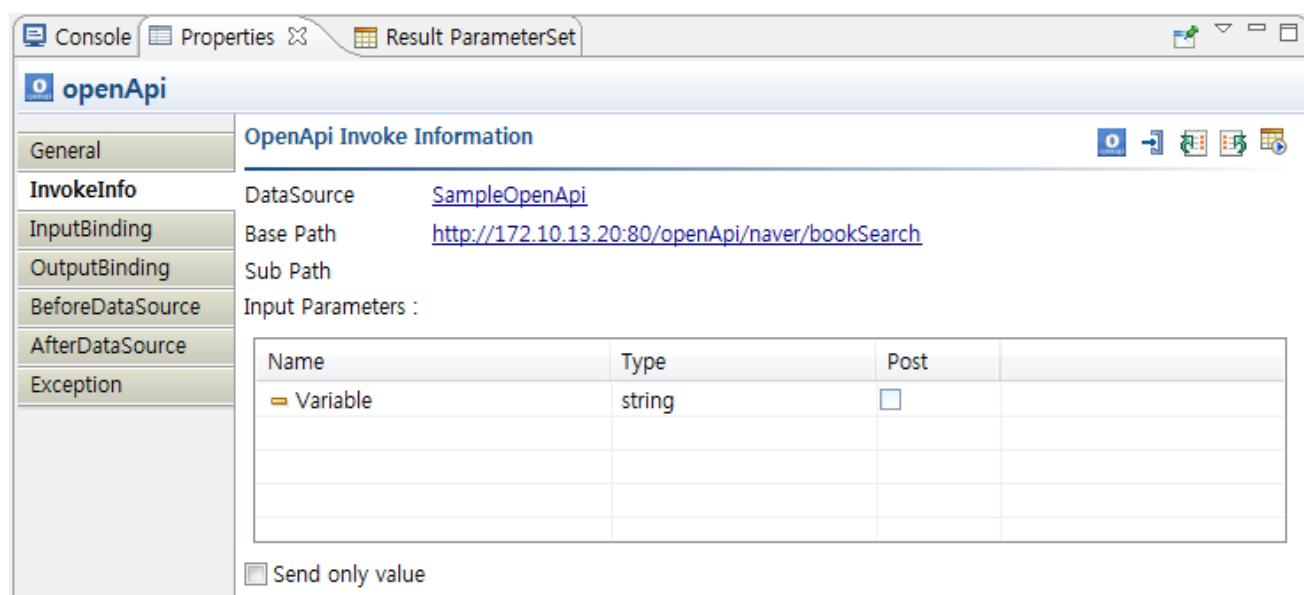
b path가 다를 수 있습니다. 항상 같은 base url을 데이터소스에 정의하고, 서비스 별로 다른 sub path는 각 Invoke에 정의합니다. 유사하게 인증 키와 같이 OpenApi를 호출할 때 항상 전달하여야 하지만 클라이언트가 입력할 필요가 없는 매개변수는 데이터소스에 정의하고, 실제 클라이언트에서 사용자가 입력할 매개변수는 Invoke에 정의합니다.

Result

OpenApi로부터 데이터를 수집하게 되면 수집된 결과가 'BODY'라는 파라매터를 생성합니다. 그리고 또한 데이터 수집을 통해 응답 받은 HTTP Header의 값들을 HEADER + INDEX로 파라매터로 생성합니다. (단, Header로 정의 된 데이터를 출력으로 지정 할 경우 HTTP Response Header에 정의 된 정보로 변경하여 응답합니다)

InvokeInfo

인보크를 수행하기 위한 필수 정보가 출력됩니다.



다른 Invoke처럼 인보크를 실행하기 위한 In-Out 정보가 이미 고정되어 있는 경우는 Information만 출력합니다.

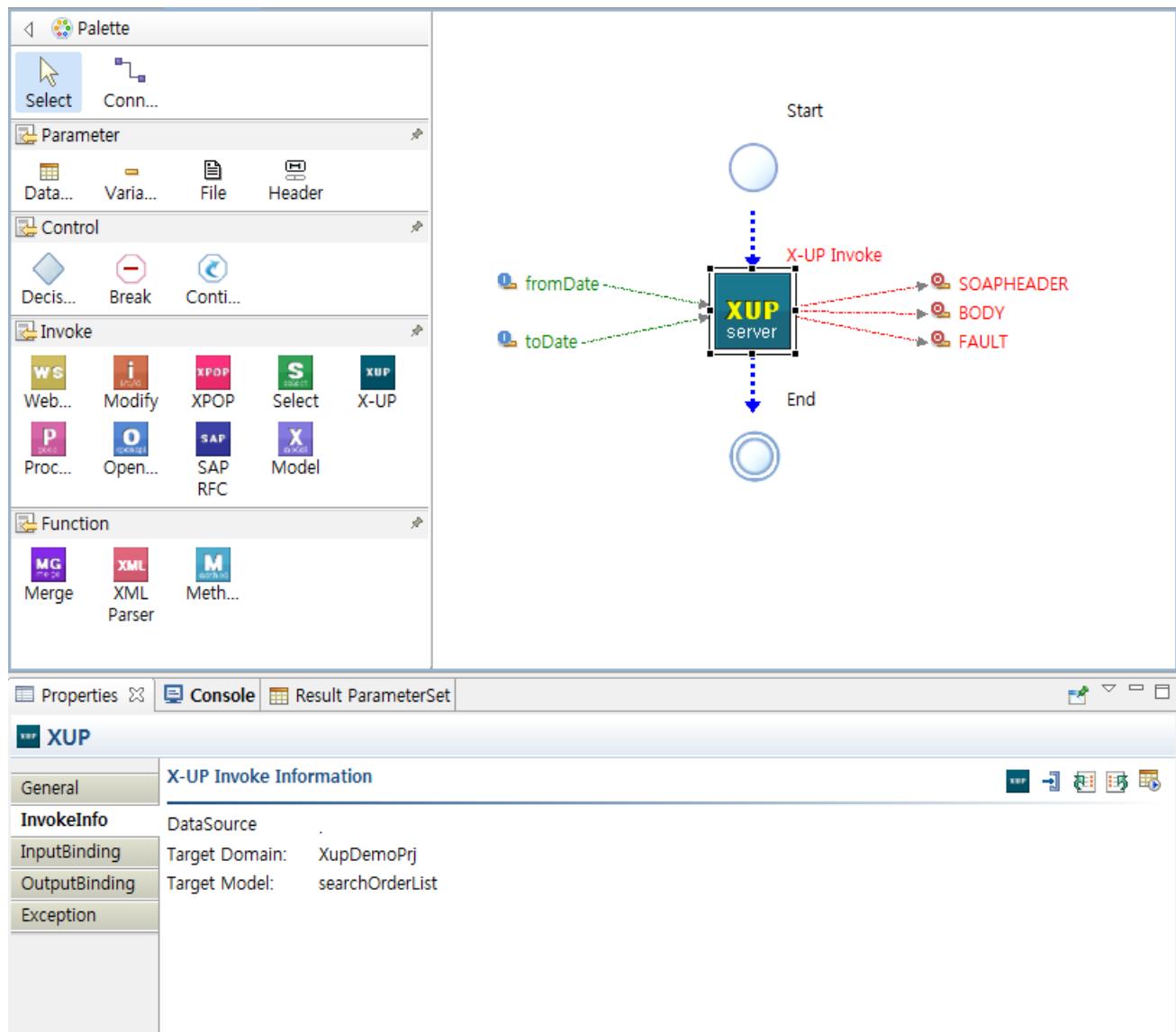
OpenApi Invoke의 경우 InvokeInfo에서 다음 정보를 출력합니다.

- DataSource : 선택한 데이터소스 (클릭시 해당 데이터소스 편집창 오픈)
- Base Path : 선택한 데이터소스에 정의된 base path url
- Sub Path : 정의된 sub path
- Input Parameters : 정의된 필수 input parameter
- Send only value : 정의된 send only value
- http header attribute
- sub path와 input parameter
- url

X-UP Invoke

X-UP Invoke는 다른 서버의 이미 만들어지고 Deploy된 X-UP 모델을 호출하기 위한 컴포넌트입니다.

X-UP Invoke는 Remote에 존재하는 X-UP 서버의 모델을 호출하여 데이터를 획득하고 새로운 파라미터를 생성합니다.

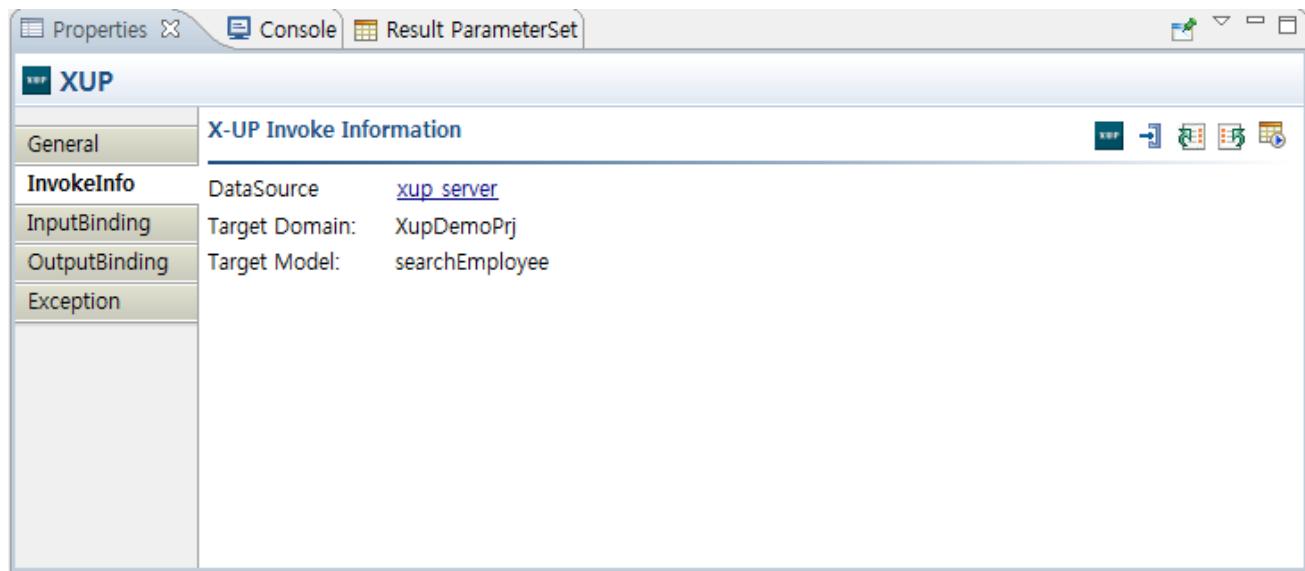


X-UP Invoke는 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General
- InvokeInfo
- InputBinding
- OutputBinding
- Exception

InvokeInfo

인보크를 수행하기 위한 필수 정보가 출력됩니다.



다른 Invoke처럼 인보크를 실행하기 위한 In-Out 정보가 이미 고정되어 있는 경우는 Information만 출력합니다.

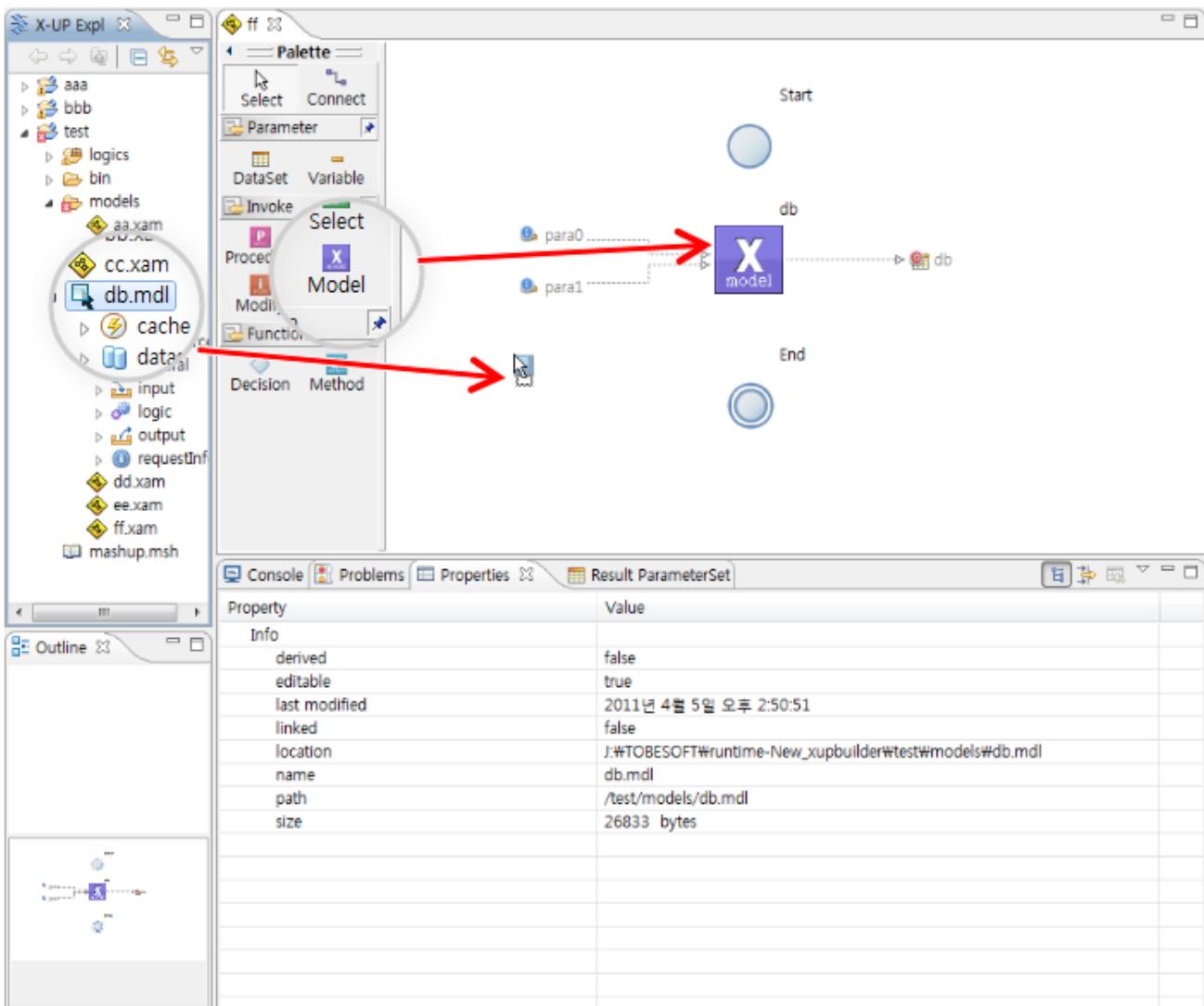
X-UP Invoke의 경우 InvokeInfo에서 다음 정보를 출력합니다.

- DataSource : 선택한 데이터소스 (클릭 시 해당 데이터소스 편집창 오픈)
- Target Domain : 선택한 서버의 타겟 도메인
- Target Model : 타겟 도메인안에 호출할 대상이 되는 타겟 모델

Model Invoke

Model Invoke는 같은 프로젝트 내의 이미 만들어져 있는 다른 X-UP 모델을 호출하기 위한 컴포넌트입니다.

Model Invoke 생성은 Palette에서 Model 아이템을 선택하거나 X-UP Explorer에서 직접 모델을 선택 후 에디터로 드래그함으로써 빠르게 생성할 수 있습니다.

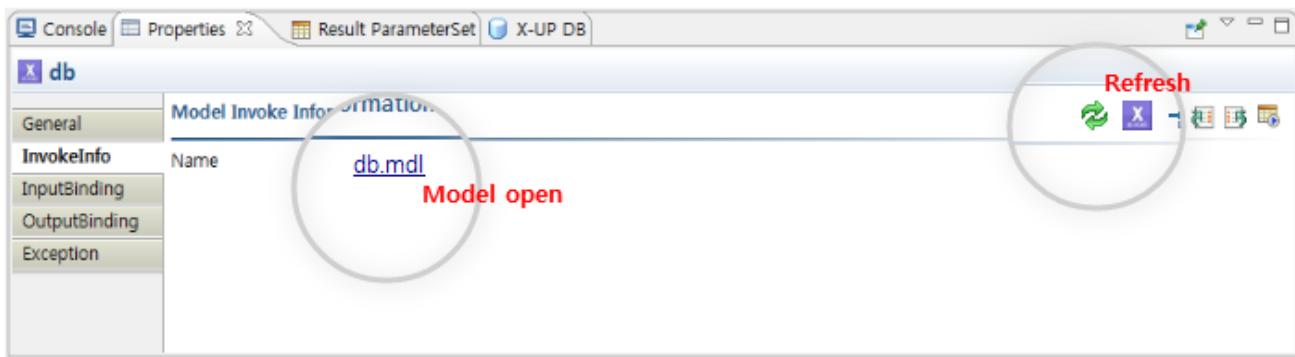


Model Invoke는 Palette에서 생성할 경우 나타나는 X-UP Model List Dialog와 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General
- InvokelInfo
- InputBinding
- OutputBinding
- Exception

InvokelInfo

Model Invoke를 수행하기 위한 필수 정보를 출력합니다.



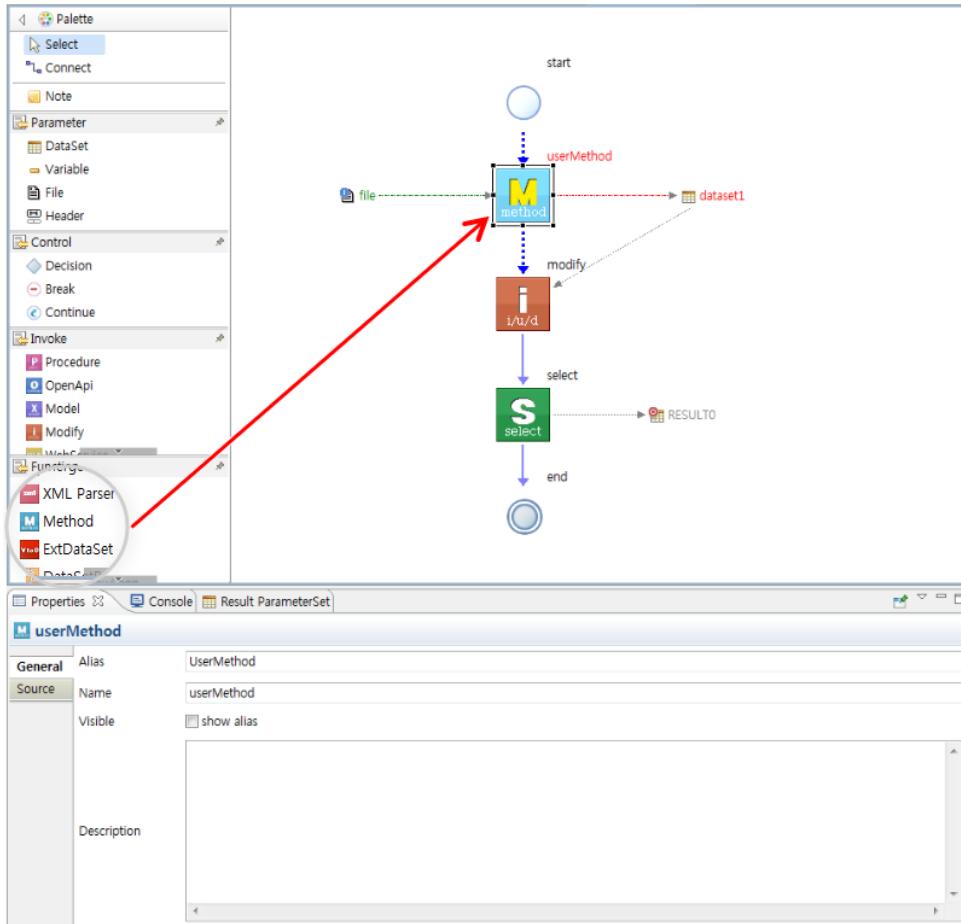
- Name : 인보크된 모델 이름을 출력합니다.

모델 이름을 클릭하면 해당 모델 에디터로 이동합니다.

우측 상단에 Refresh 메뉴를 통해 인보크된 모델의 정보가 변경되었을 경우 다시 갱신할 수 있도록 합니다.

Method Function

User Method는 모델 개발 시 사용자 정의 코드를 직접 작성할 수 있도록 합니다. 실제 자바 소스의 메소드로 생성되며 원하는 자바 코드를 자바 파일 없이 자유롭게 정의하여 모델에 추가할 수 있습니다.

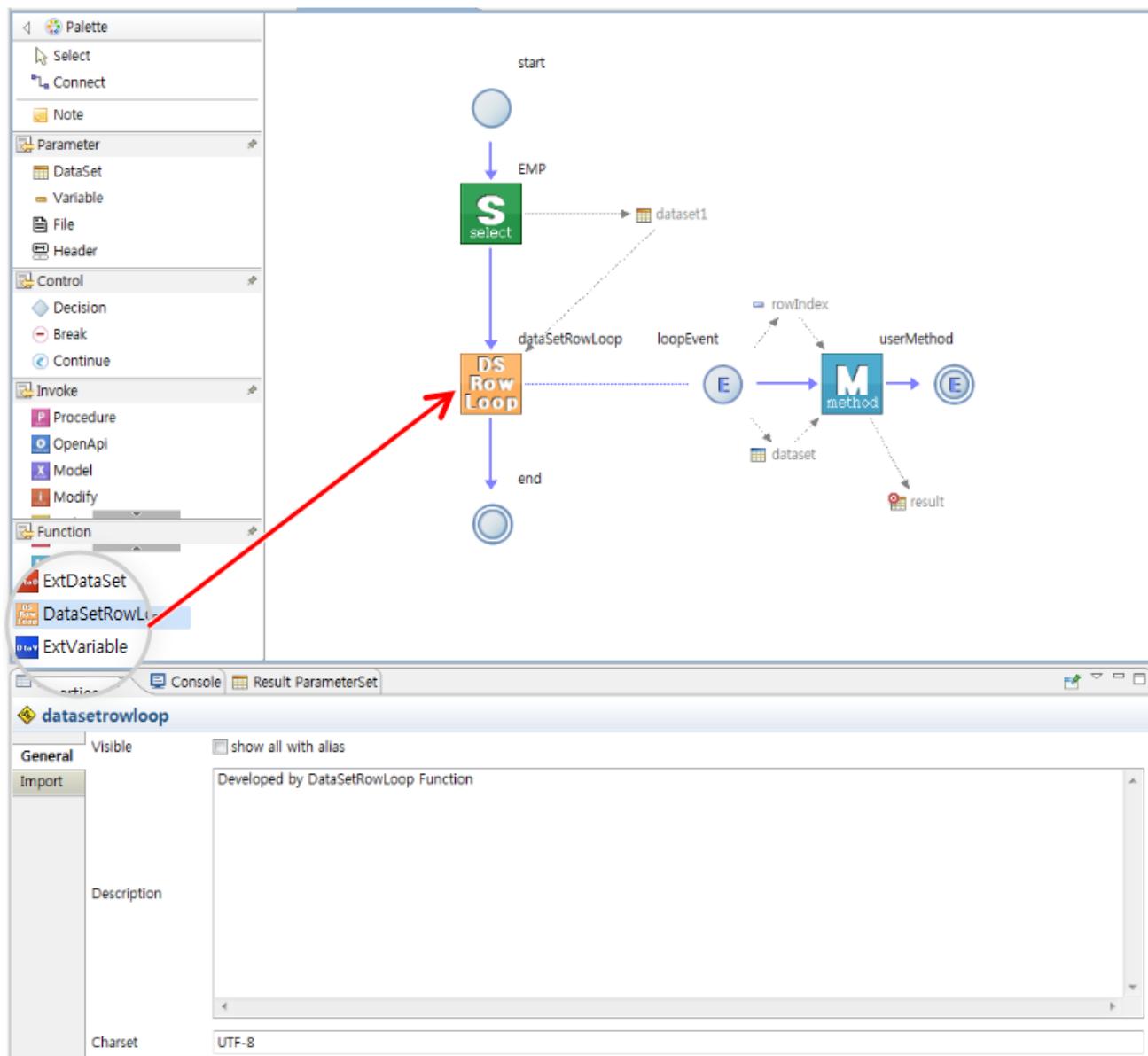


User Method에서 제공하는 프로퍼티 속성은 다음과 같습니다.

- General : 일반정보
- Source : 자바 코드 창

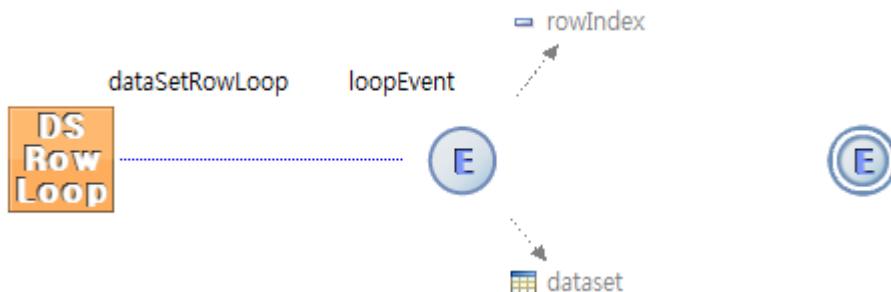
DataSetRowLoop Function

DataSetRowLoop Function은 X-UP Bulider Automation 모델 Editor내에서 DataSetRowLoop 함수에 한 개의 DataSet을 연결시키게 되면 DataSetRowLoop 함수에 연결된 데이터셋의 Row개수 만큼 반복하여 이벤트를 발생 시키는 함수입니다.



DataSet을 DataSetRowLoop 함수에 마우스 연결한 뒤 DataSetRowLoop 컴포넌트를 마우스 우클릭하여 AddEve

nt >loopEvent를 클릭하면 이벤트(Start, End)가 나오고 Start Event에 파라메터로 Variable 타입의 rowIndex와 DataSet 타입의 dataset이 추출됩니다.



DataSetRowLoop 함수에서 제공하는 프로퍼티 속성은 다음과 같습니다.

- General : 일반정보
- DataSet RowType : 데이터셋의 로우타입 설정 창

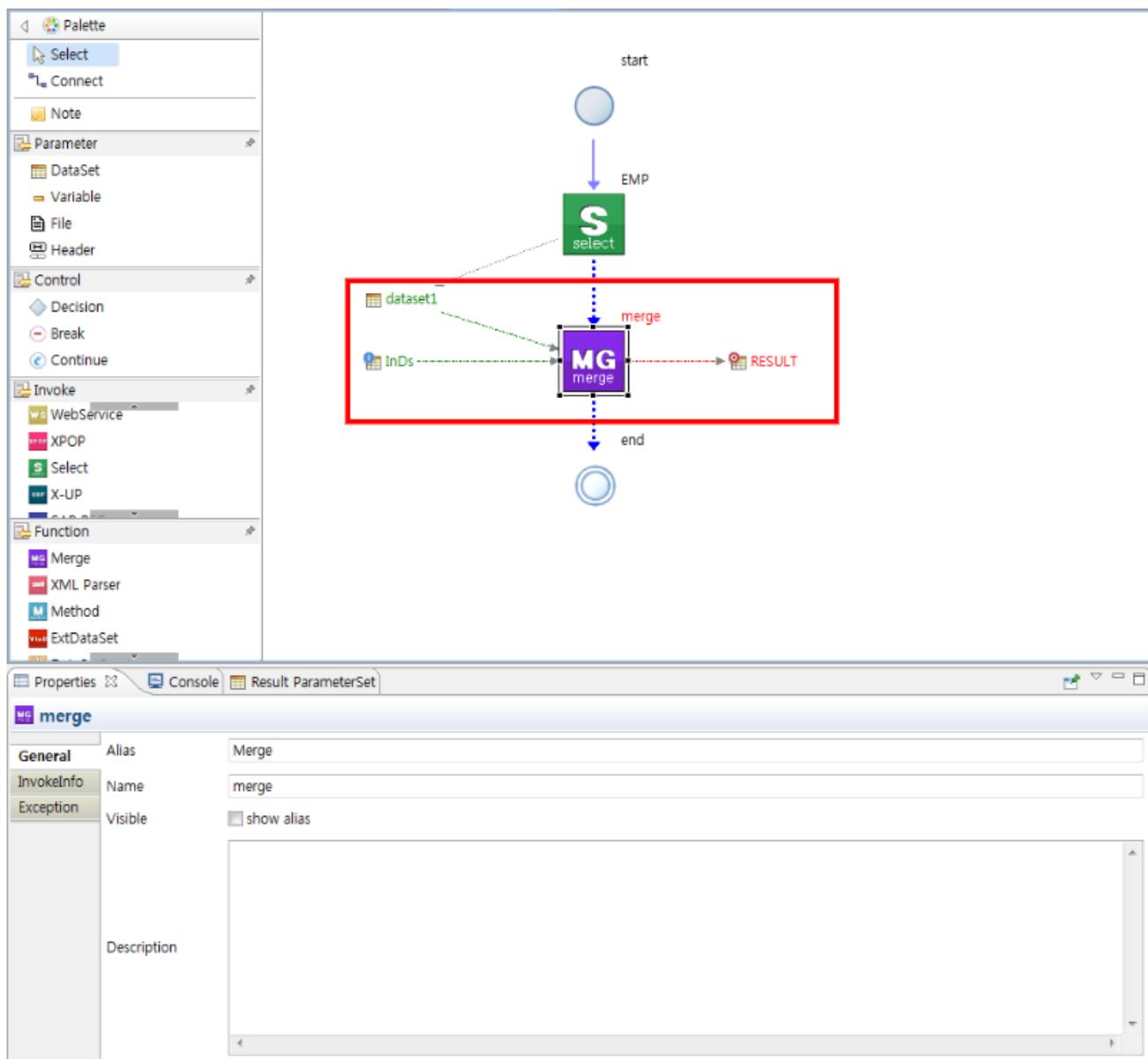
Merge Function

Merge Function은 복수의 데이터셋을 가지고 새로운 데이터셋을 생성하기 위한 컴포넌트입니다.

이러한 기능은 정형화된 복수의 데이터셋을 내장된 DBMS에 의해 sql 쿼리문으로 표현된 로직이 실행되어 이루어집니다.



X-UP은 내부적으로 메모리 데이터베이스 시스템을 사용하여 융합 모델을 구현하고 있습니다. 따라서 대량의 데이터를 출력하는 모델들을 융합할 경우 X-UP서버가 운영되는 WAS의 메모리 및 성능에 영향을 미칠 수 있습니다.



이러한 융합을 위한 융합 모델 개발은 다음과 같은 단계로 진행됩니다.

- 대상 데이터셋 설정
- 융합 로직 및 sql 쿼리 구현
- 새로운 데이터셋 생성

대상 데이터셋 설정

융합에 사용될 데이터셋을 설정합니다. X-UP모델에 의한 데이터셋 또는 사용자가 매개변수로 전달한 데이터셋이 설정될 수 있습니다.

융합 로직 및 sql 쿼리 구현

X-UP의 융합은 내장된 DBMS를 사용하며, 융합하기 위한 로직은 SQL의 Select 구문으로 구현합니다. X-UP은 개발자가 구현한 SQL과 설정된 대상 데이터셋을 내장 DBMS에 전달하고 SQL조회의 결과로 반환된 값을 처리하여 융합된 원시데이터를 생성합니다.

새로운 데이터셋 생성

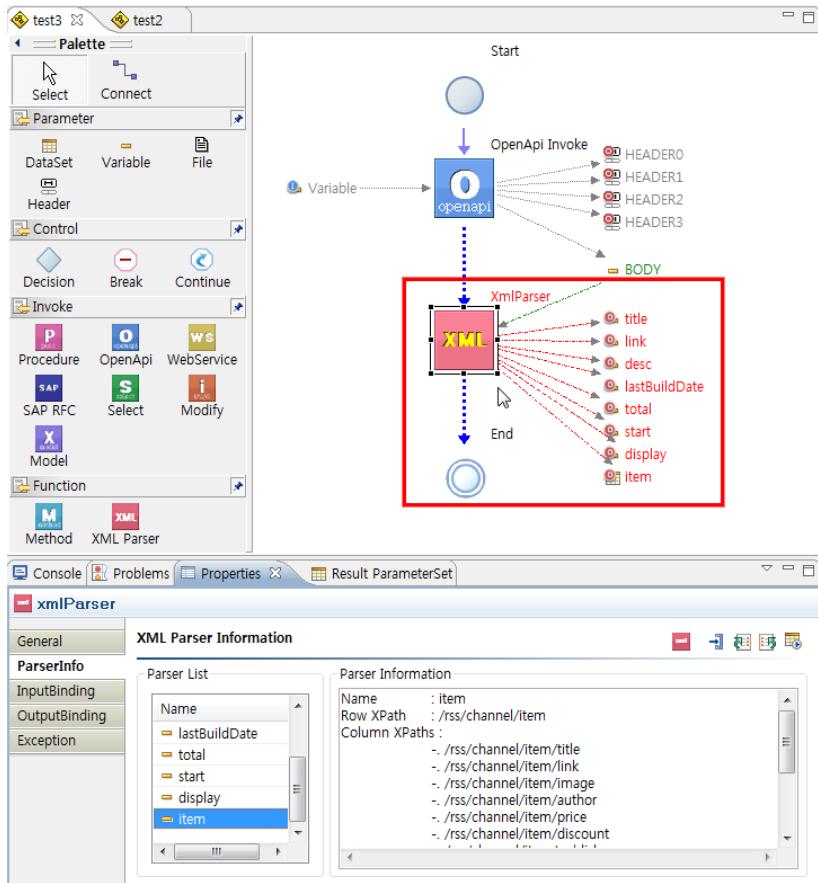
내장된 DBMS에 의한 융합된 원시데이터는 그대로 반환되는 것이 아니라 새로 생성된 데이터셋으로 변환됩니다.

Merge Function은 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General : 일반 정보
- InvokingInfo : Merge를 수행할 Sql 쿼리문에 대한 정보를 출력합니다.
- Exception

XML Parser Function

XML Parser Function은 XML 정보를 파싱하여 파라메터형식으로 생성하기 위한 컴포넌트입니다. 호출 시 Variable or Dataset 형식으로 가져옵니다.

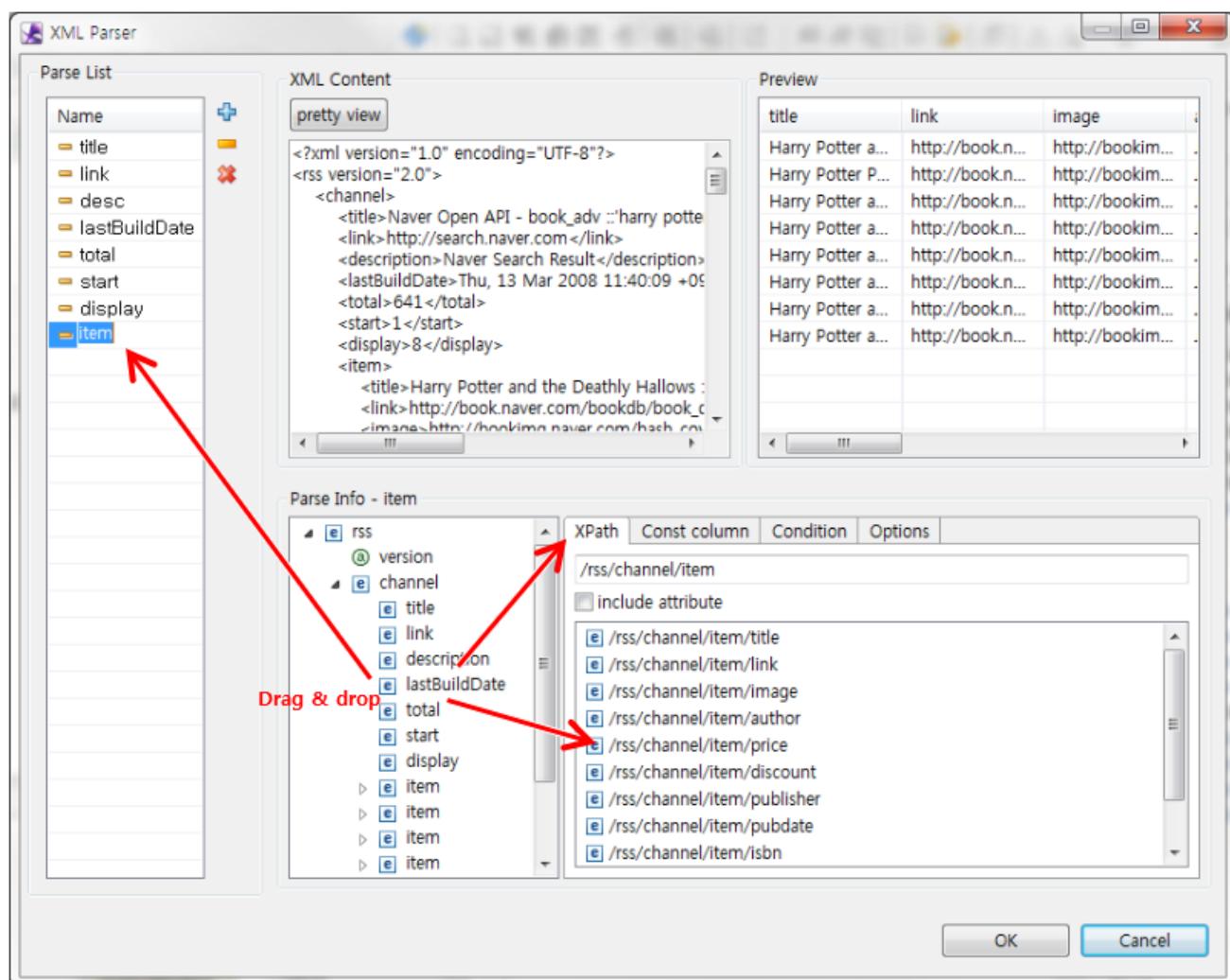


XML Parser Function은 인보크를 생성하기 위한 XML Parser Function Dialog 와 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General
- ParseInfo
- InputBinding
- OutputBinding
- Exception

XML Parser Function Dialog

XML Parser Function 을 생성하고자 할 경우 다음과 같은ダイ얼로그가 출력됩니다.

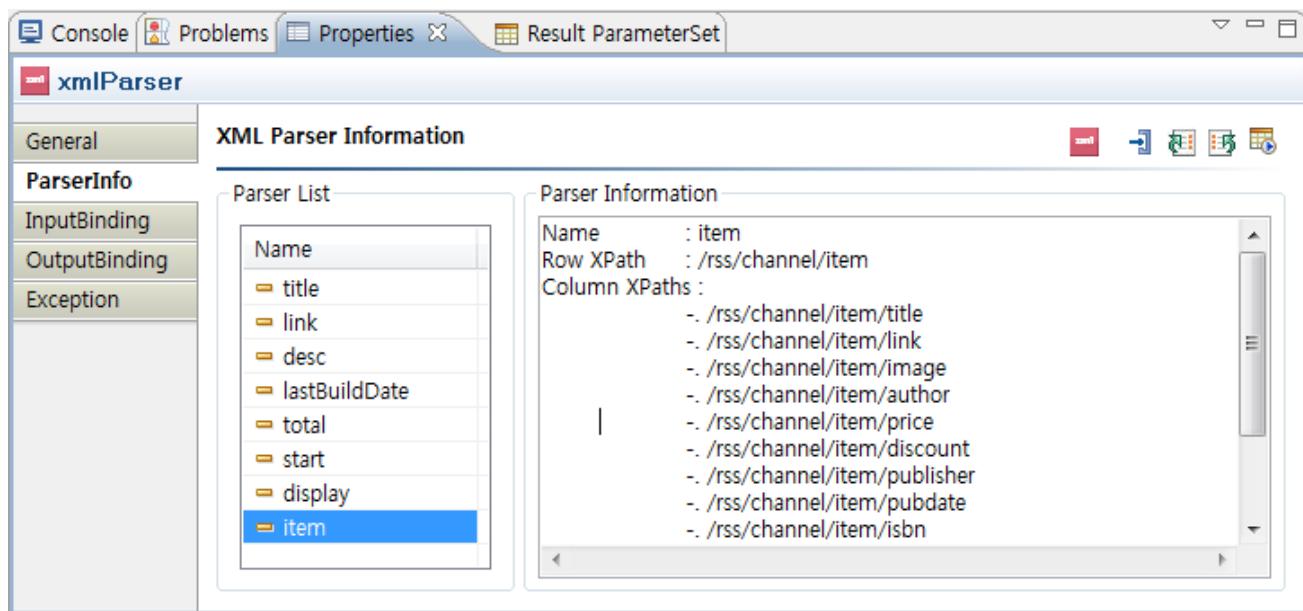


- Parse List: XML 파싱 리스트
- XML Content : 직접 입력 가능하며 파싱하고자하는 xml 정보를 직접 입력합니다.
- Preview : 파싱된 결과를 미리 보여줍니다.
- Parse Info : 선택한 파서의 파싱 정보를 정의합니다.

- 왼쪽 트리는 xml 정보를 트리 형태로 보여줍니다.
- XPath : 트리에서 선택한 아이템의 xml path 를 정의
- Const column : 데이터셋일 경우 const 컬럼을 정의
- Condition : 특정 조건에 만족하는것만 파싱하고자 할 경우 정의
- Options : 중복된 아이템이름이 존재할 경우 value처리시 구분자를 입력

ParserInfo

인보크를 수행하기 위한 필수 정보가 출력됩니다.



다른 Invoke 처럼 인보크를 실행하기 위한 In-Out 정보가 이미 고정되어 있는 경우는 Information 만 출력합니다.

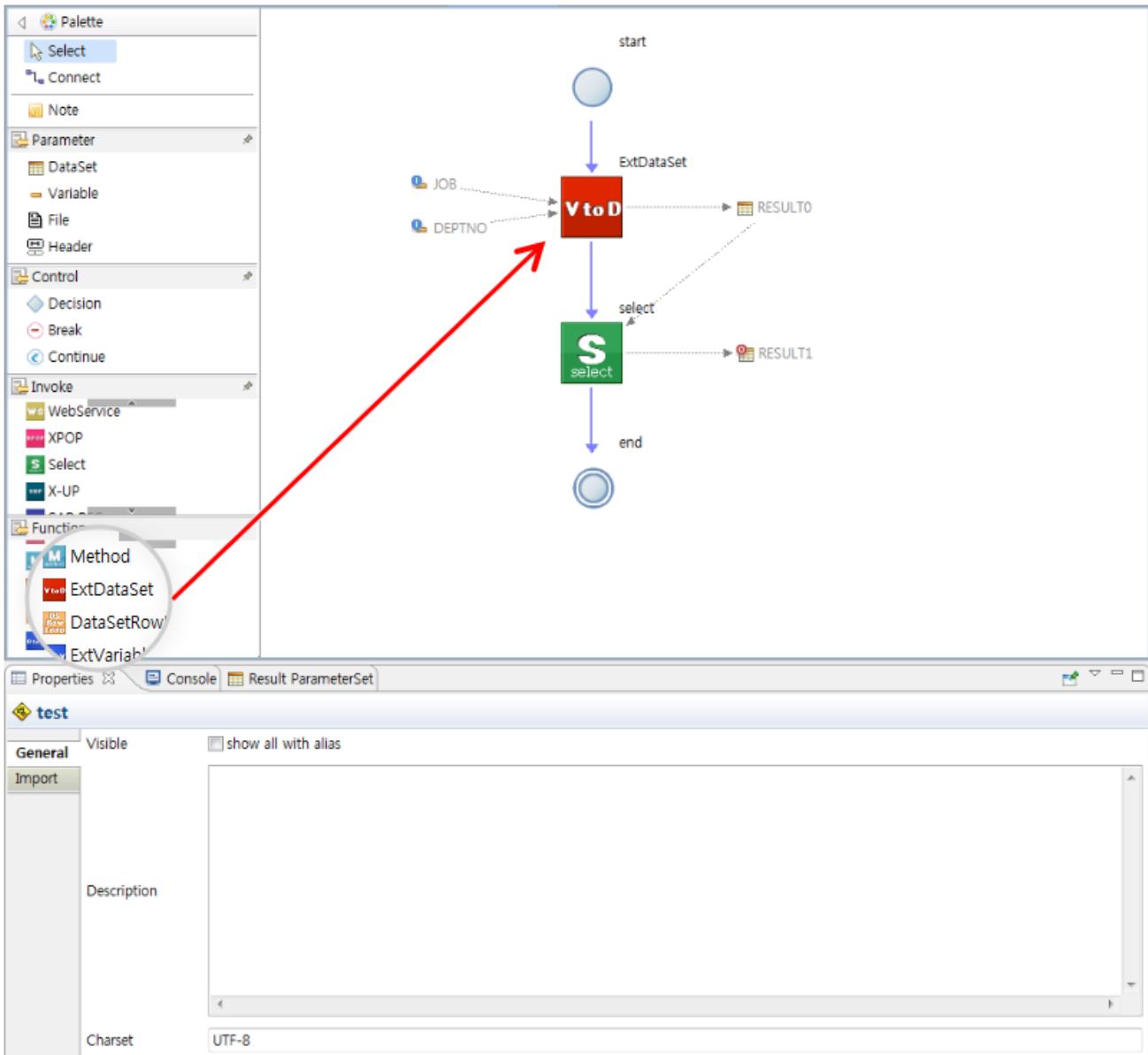
XML Parser Function의 경우 InvokelInfo에서 다음 정보를 출력합니다.

- Parser List: 정의된 파서 리스트 출력
- Parser Information : 선택한 파서의 파싱정보 출력

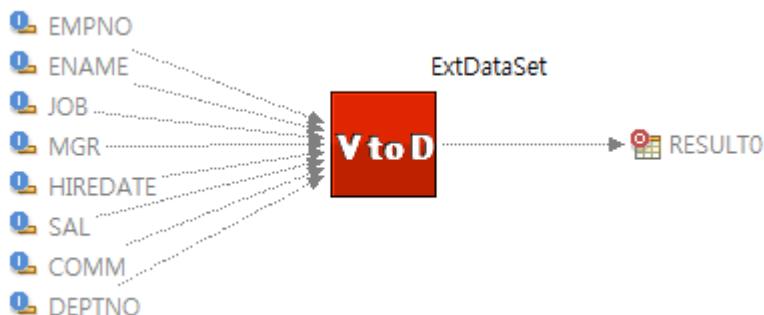
ExtDataSet Function

ExtDataSet Function은 여러 개의 Variable들을 ExtDataSet 함수에 입력 파라메터로 연결시켜 주게되면 자동적으로 각각의 Variable 값들이 하나의 데이터셋으로 만들어지는 함수입니다.

즉, Variable > DataSet으로 만드는 함수입니다.



ExtDataSet Function이 여러 개의 Variable과 연결되면 자동적으로 데이터셋이 Output값으로 추출합니다.



- ExtDataSet 함수에 연결된 Variables가 Update(갱신)되었을 경우 : Variables이 추가, 삭제 될 때 만들어져 있던 Output DataSet을 삭제한 뒤, 갱신된 데이터셋을 새로 만듭니다..
- ExtDataSet 함수에 연결된 Variable이 하나도 없을 경우 : Output 데이터셋을 삭제합니다.

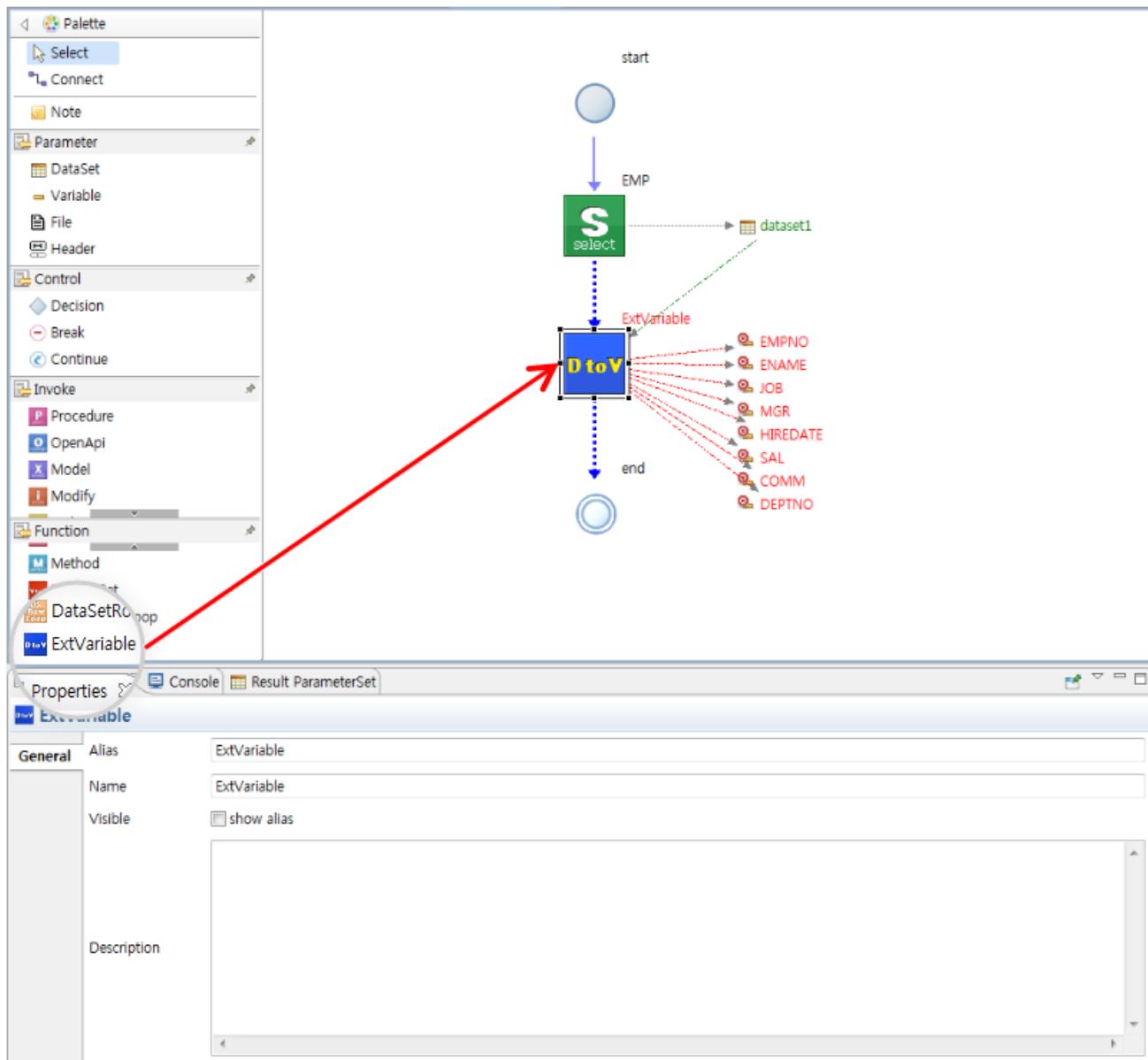
ExtVariable 함수는 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General

ExtVariable Function

ExtVariable Function은 데이터셋의 Column값들을 쉽게 ExtVariable 함수에서 Column Name, Column Value, Column Size가 각각의 Output 타입의 Variable 파라메터로 만들어지는 함수입니다.

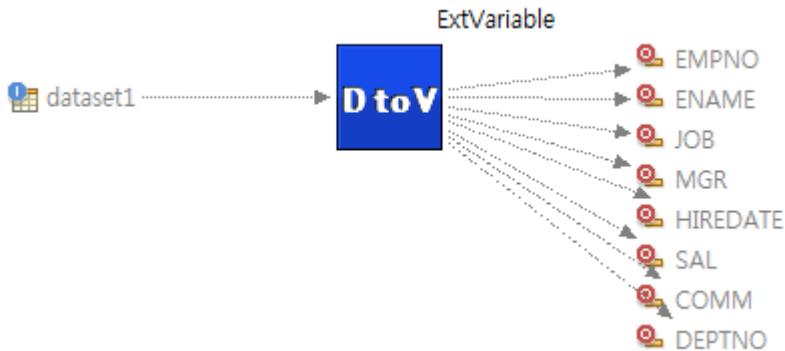
즉, DataSet > Variable 으로 만드는 함수입니다.



ExtVariable Function이 Dataset과 연결되면 Dataset내에 있는 Column 개수, 이름, 타입, 크기, 값에 맞춰 Variables을 Output 값으로 추출합니다.

예를 들어, Dataset의 column이 4개라면 4개의 Variables가 Output으로 출력됩니다. 각각의 Variable이름, 타입,

크기, 같은 Dataset 각각 Column 이름, 타입, 크기, 값으로 동일합니다.

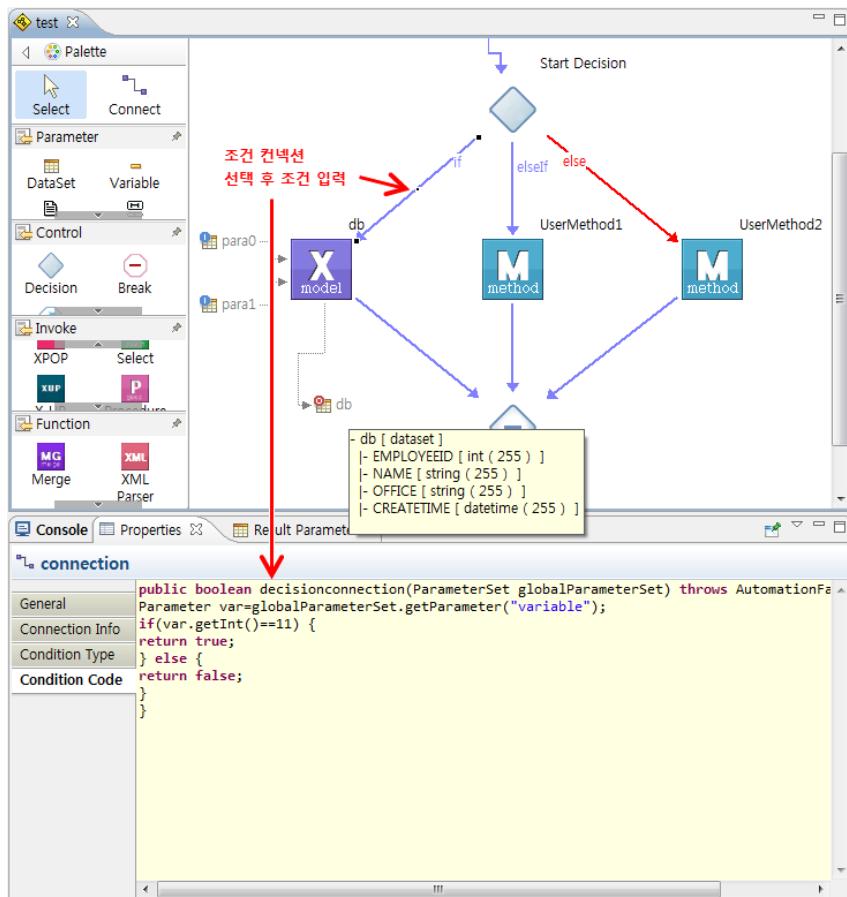


ExtVariable 함수는 다음과 같은 프로퍼티 속성탭을 가지고 있습니다.

- General

Decision

Decision은 조건에 따른 분기처리를 가능하게 합니다. 원하는 조건에 만족할 경우 특정 프로세스를 실행할 수 있습니다.



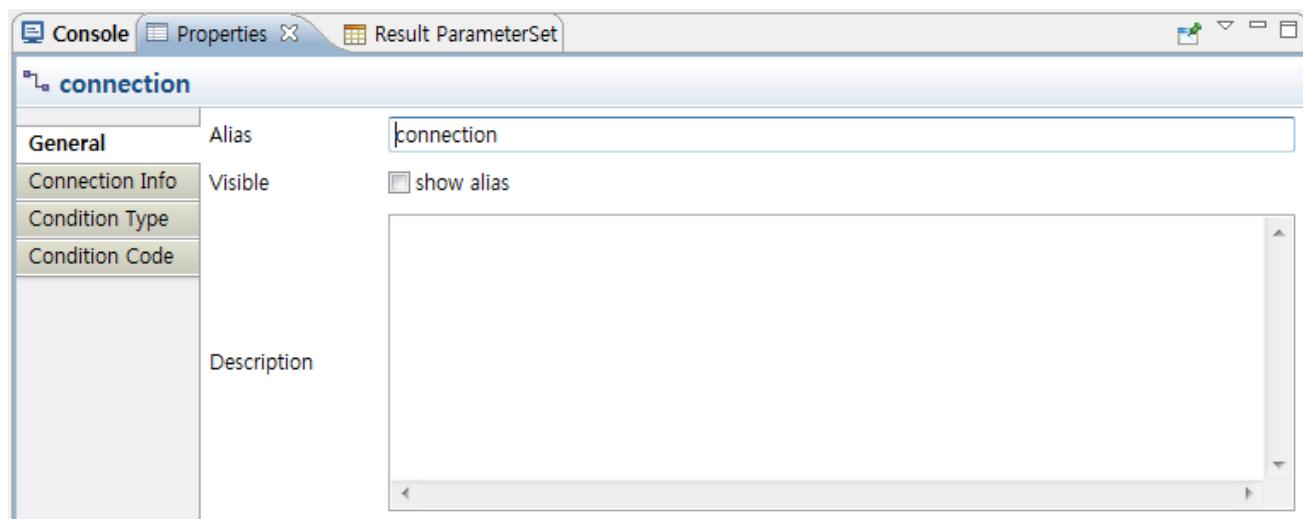
조건은 if / else if / else 세가지로 구분되며 해당 커넥션을 선택하여 원하는 조건타입으로 변경할 수 있습니다.

조건 커넥션의 프로퍼티 속성은 다음과 같습니다.

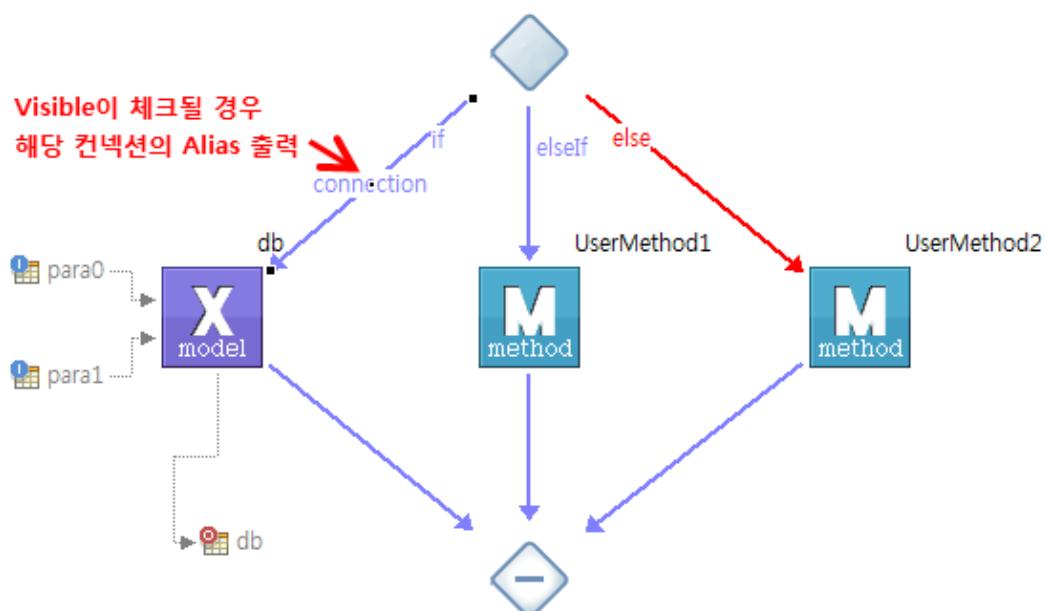
- General : 모든 컴포넌트 공통
- Connection Info : 모든 커넥션 노드 공통
- Condition Type : if / elseif / else 선택
- Condition Code : 조건 로직 입력

General

조건 커넥션의 General에는 Alias, Visible, Description 세가지 정보를 정의할 수 있습니다.

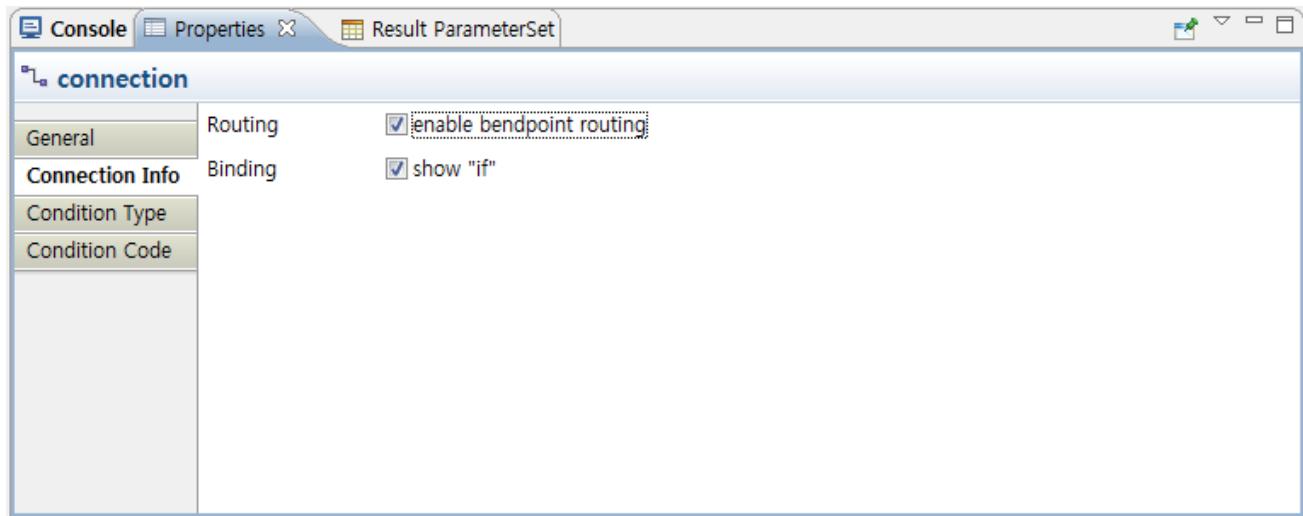


특히 Visible의 경우는 다른 컴포넌트의 visible과 다르게 Name/Alias의 정보가 없기 때문에 커넥션 중앙에 Alias를 보여줄지를 결정합니다.

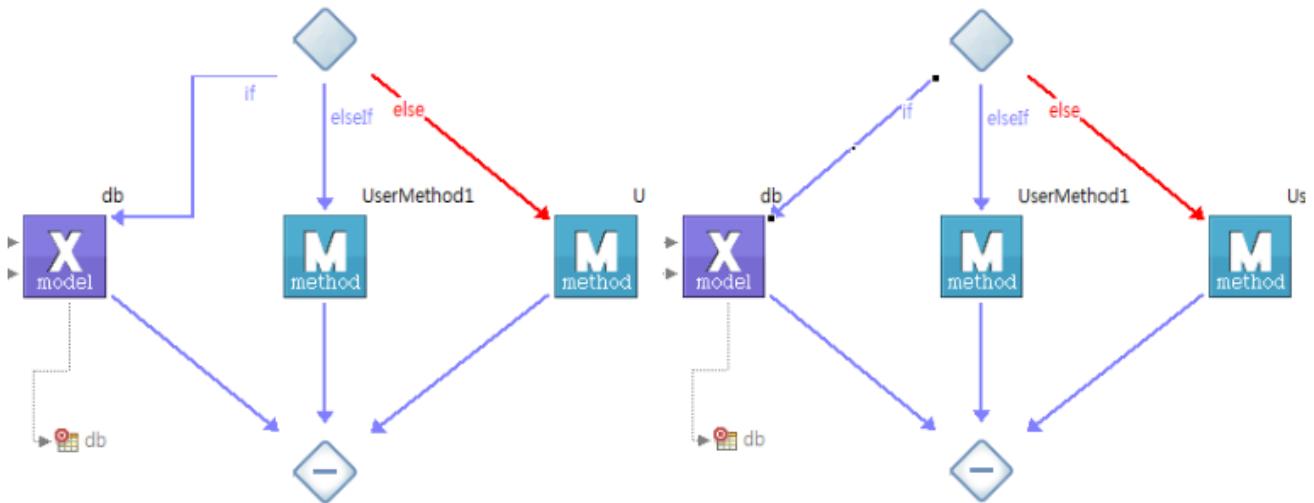


Connection Info

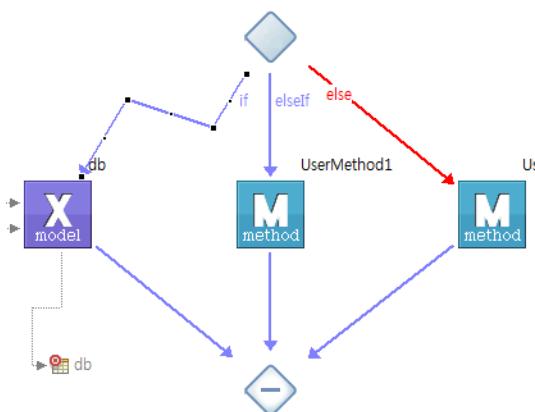
커넥션 정보 속성은 모든 커넥션의 공통 속성으로써 조건 커넥션 역시 동일합니다.



Routing : 커넥션 라우팅 타입을 결정합니다. (아래 그림의 오른쪽이 체크된 경우)



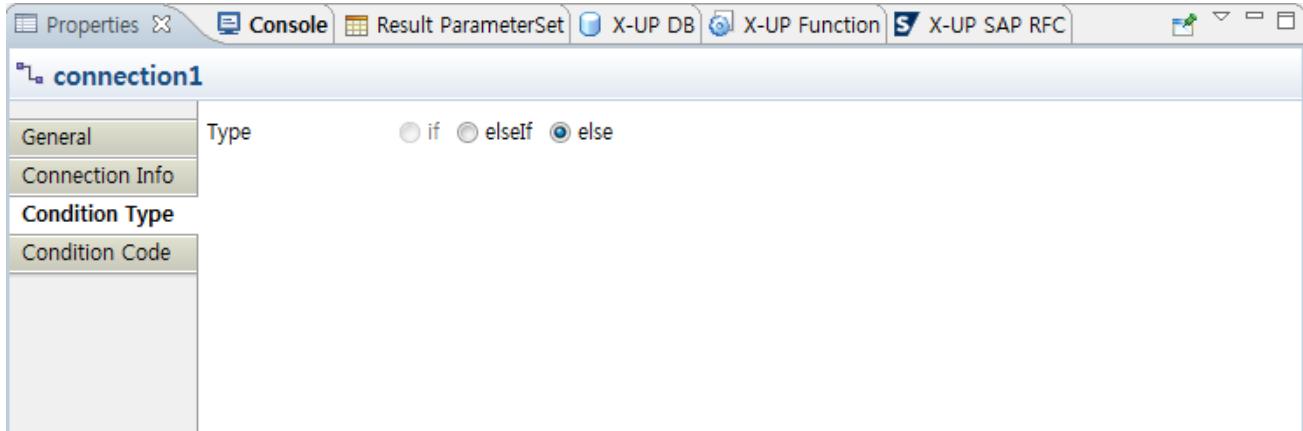
커넥션이 BendPoint 라우팅일 경우 사용자가 원하는 형태로 커넥션 모양을 정의할 수 있습니다.



Binding : 조건 커넥션일 경우 조건타입명을 보여줄지 여부를 결정합니다. 체크를 해제하면 커넥션에 보여지던 if/else / elseif 같은 문자열을 숨깁니다.

Condition Type

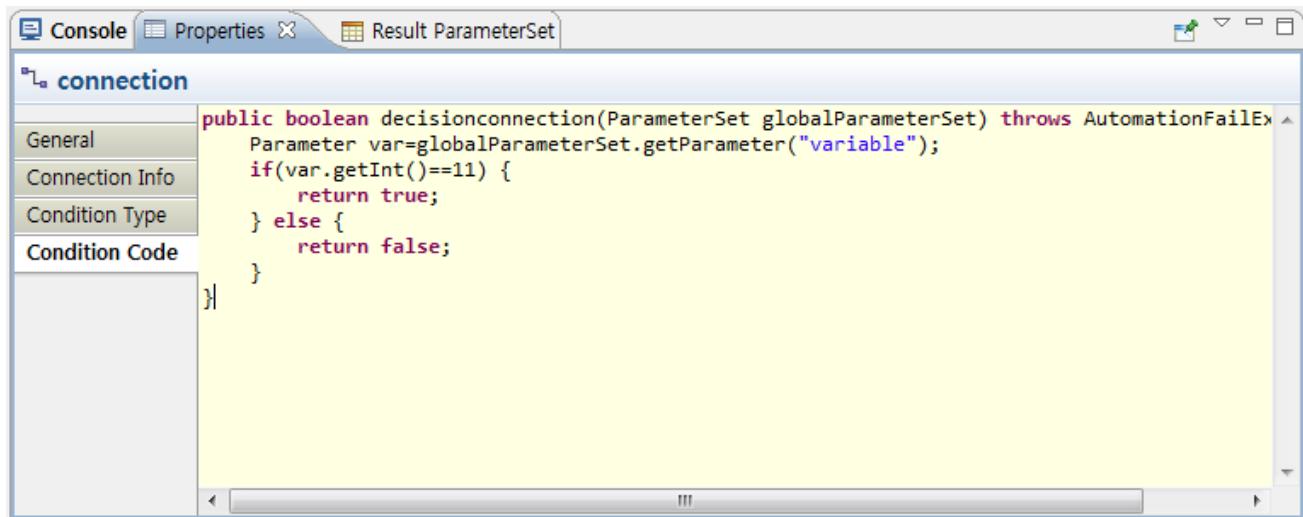
컨디션 타입은 사용자가 직접 컨디션 타입을 선택할 수 있습니다.



if/ elseif / else 선택할 수 있습니다.

Condition Code

컨디션 코드는 사용자가 직접 자바 코드로 원하는 조건 로직을 정의할 수 있습니다.



에디터상에 존재하는 모든 파라메터들은 globalParameterSet에 존재하므로 원하는 파라메터를 가져와 조건 로직을 정의할 수 있습니다.

반드시 로직의 결과값으로 true or false 를 리턴하도록 합니다.

프로퍼티내에 존재하는 모든 자바 코드 편집창은 공통적으로 자바 에디터에서 제공했던 동일한 어시스트 팝업 및 자

바운법 오류 검사 기능을 가지고 있습니다.

```

57 import com.tobesoft.xplatform.data.datatype.Datatype;
58 import com.tobesoft.xup.data.MashupFile;
59 import com.tobesoft.xup.bundle.invoker.openapi.OpenApiInvokingInfo;
60 import com.tobesoft.xup.bundle.invoker.model.ModelInvokingInfo;
61 import com.tobesoft.xup.bundle.invoker.FlowEvent;
62 import com.tobesoft.xplatform.data.datatype.PlatformDataType;
63 import com.tobesoft.xup.bundle.invoker.EventHandler;
64 import com.tobesoft.xup.util.data.XmlParsingInfo;
65
66 public class addressBooksAutomationLogic extends addressBooksBaseAutomationLogic {
67
68     public void start(ParameterSet globalParameterSet) throws AutomationFailException {
69     }
70     public void end(ParameterSet globalParameterSet, ParameterSet outputParameterSet) throws AutomationFailException {
71     }
72     public boolean decisionconnection1(ParameterSet globalParameterSet) throws AutomationFailException {
73
74         } globalParameterSet : ParameterSet
75     publ
76     77
78     79
79     80
81     82
83     84
85     86
86     publ
87
88     89
89     90
90     publ
91
92     93
93 }
```

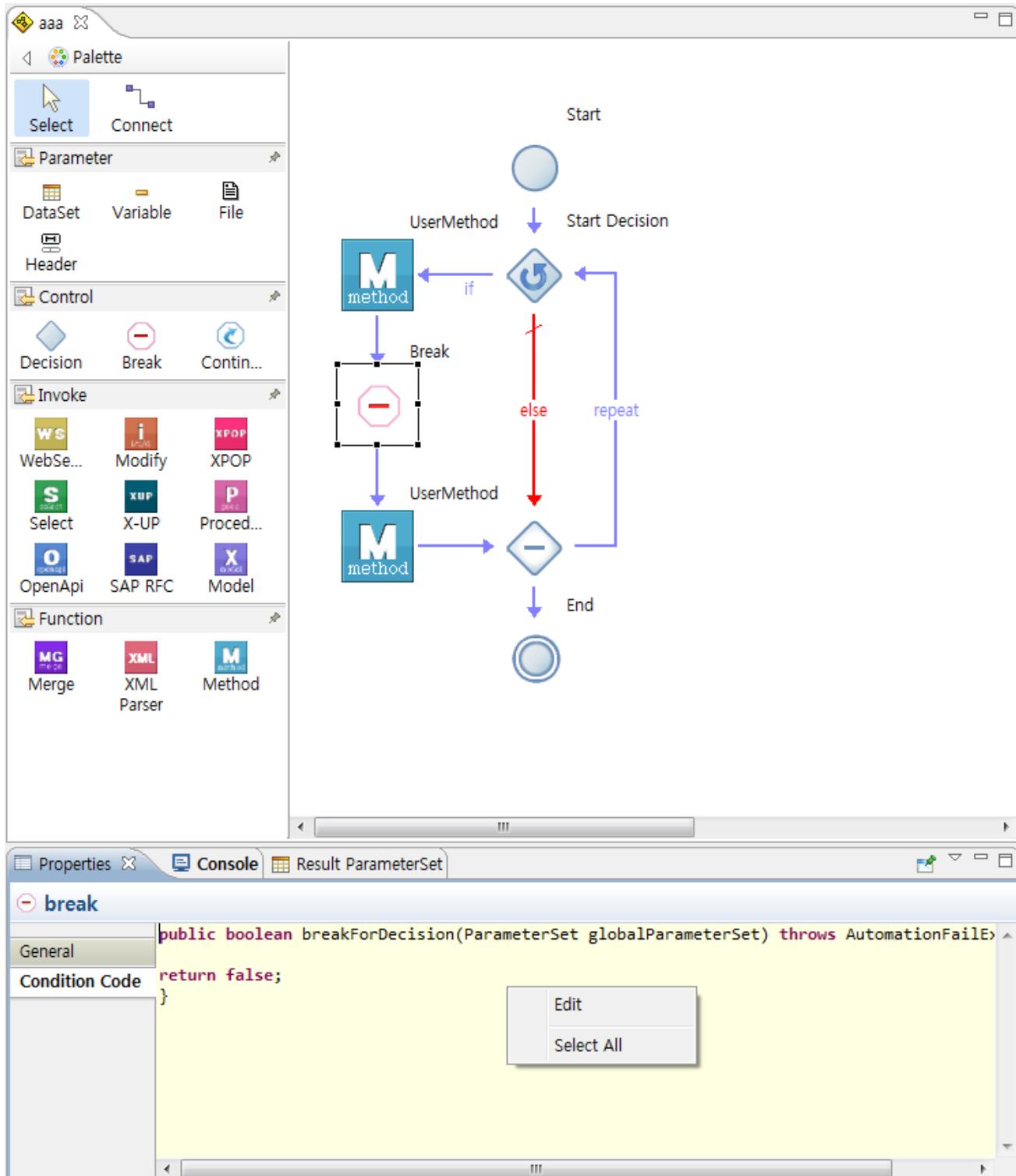
Ctrl + Space

```

57 import com.tobesoft.xplatform.data.datatype.Datatype;
58 import com.tobesoft.xup.data.MashupFile;
59 import com.tobesoft.xup.bundle.invoker.openapi.OpenApiInvokingInfo;
60 import com.tobesoft.xup.bundle.invoker.model.ModelInvokingInfo;
61 import com.tobesoft.xup.bundle.invoker.FlowEvent;
62 import com.tobesoft.xplatform.data.datatype.PlatformDataType;
63 import com.tobesoft.xup.bundle.invoker.EventHandler;
64 import com.tobesoft.xup.util.data.XmlParsingInfo;
65
66 public class addressBooksAutomationLogic extends addressBooksBaseAutomationLogic {
67
68     public void start(ParameterSet globalParameterSet) throws AutomationFailException {
69     }
70     public void end(ParameterSet globalParameterSet, ParameterSet outputParameterSet) throws AutomationFailException {
71     }
72     public boolean decisionconnection(ParameterSet globalParameterSet) throws AutomationFailException {
73         ParameterSet ds = globalParameterSet.get();
74         if(ds == null) {
75             return false;
76         }
77         String company = ds.getString(0, "COMPANY");
78         return "TOBESOFT".equals(company);
79     }
80
81     public void userMethod(ParameterSet globalParameterSet) throws AutomationFailException {
82         Parameter parameter = new Parameter("companyName", "TOBESOFT");
83         globalParameterSet.add(parameter);
84     }
85     public void userMethod1(ParameterSet globalParameterSet) throws AutomationFailException {
86         Parameter parameter = new Parameter("companyName", "Partner");
87         globalParameterSet.add(parameter);
88     }
89
90
91
92
93 }
```

Repeat Task

Repeat Task 기능은 반복문을 구현하기 위한 기능으로 Decision 노드를 사용하여 처리합니다.



Decision End노드에서 Decision Start노드로 커넥션을 연결하면 자동으로 repeat connection이 생성된다. 이때 반드시 1개의 조건만 있어야 가능하며 만약 1개이상의 조건커넥션이 존재한다면 repeat connection은 생성되지 않습

니다.

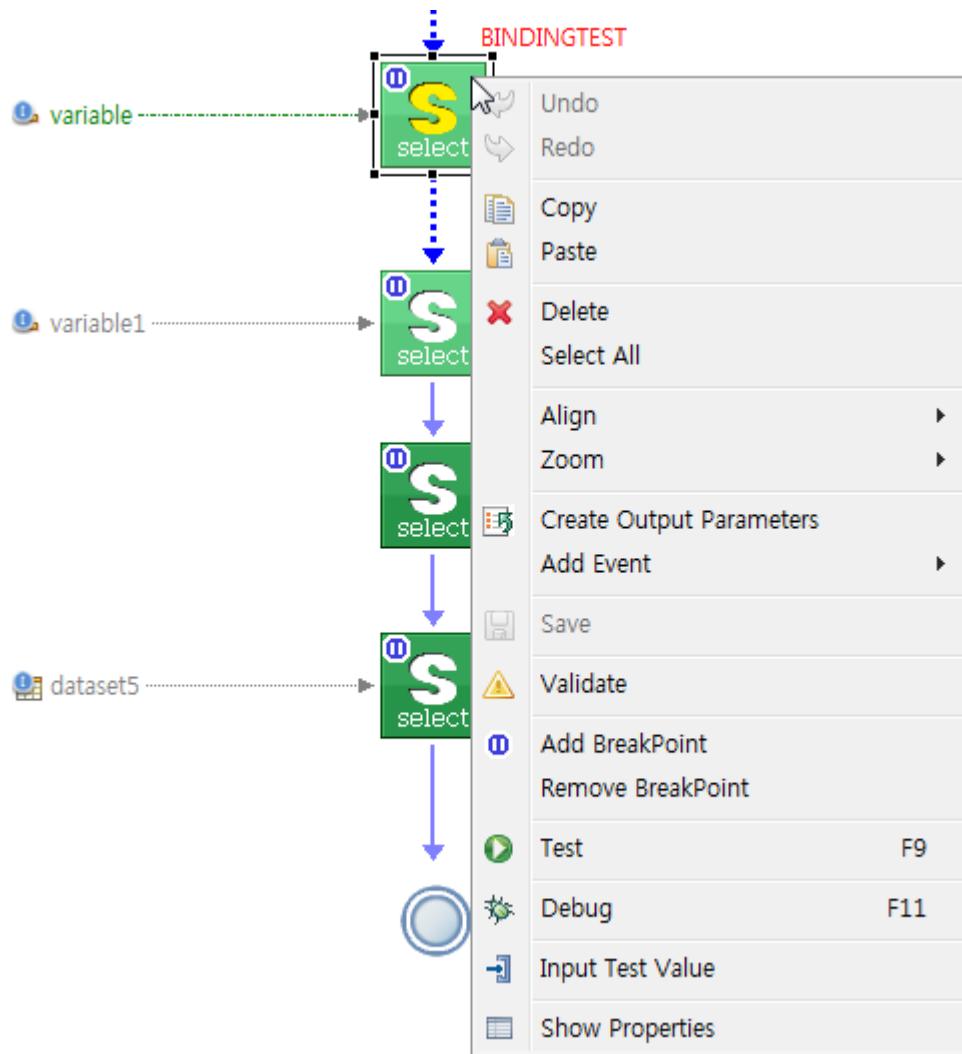
Break와 Continue 노드는 repeat 구문안에서만 사용가능하며 원하는 시점에서 반복을 중단하거나 계속하고자 할 경우 사용될 수 있습니다.

Break나 Continue노드를 선택하면 Properties Condition Code에 소스코드가 나옵니다. 소스코드에 마우스 오른쪽 버튼을 누르고 Edit를 클릭하면 Condition Code를 사용자가 직접 정의할 수 있습니다.

Debug and Break Point

Automation Model은 모델 디버깅을 위해 에디터상에서 노드에 직접 BreakPoint를 추가할 수 있습니다. BreakPoint가 노드에 추가되면 에디터가 소스 생성시 자동으로 해당 라인에 BreakPoint를 등록합니다.

BreakPoint는 각각 노드를 선택한 후 추가하거나 다중 선택후 한꺼번에 추가할 수도 있습니다.



추가된 BreakPoint는 Remove BreakPoint메뉴를 통해 삭제할 수 있습니다.

Automation Model의 모델 디버깅은 크게 두가지 방법으로 할 수 있습니다.

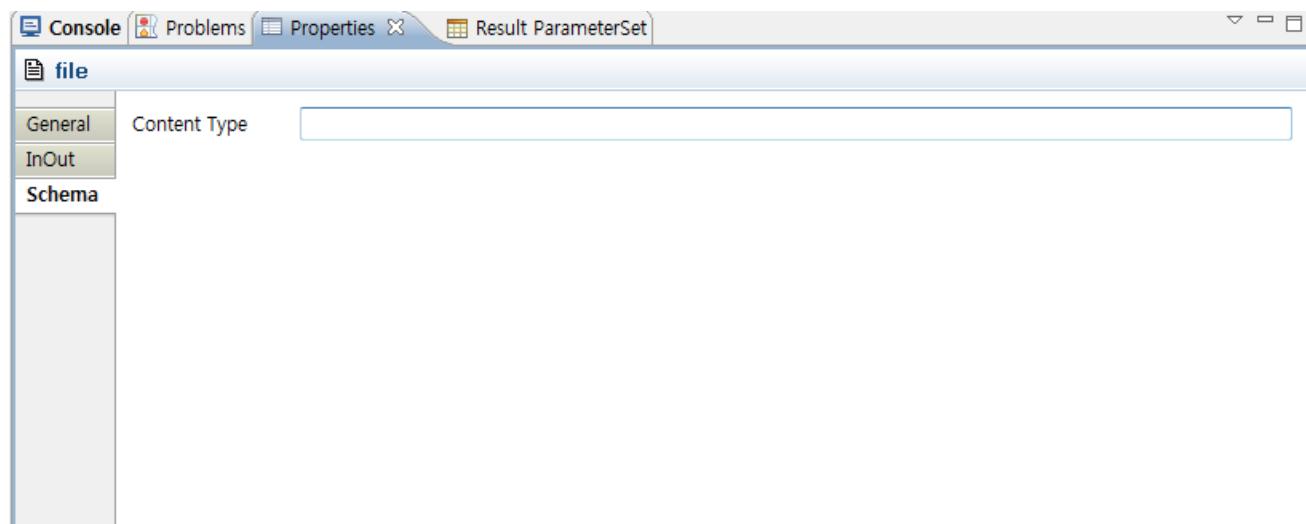
- 에디터의 Debug 메뉴를 통한 빠른 디버깅
- JUnit Test Class를 생성한 후 디버그 모드로 실행

Parameter

File

File Parameter는 파일 입출력을 정의하기 위한 파라미터입니다.

기본적으로 다른 파라미터와 동일한 프로퍼티를 가지고 있으며 Schema탭만 다르게 정의합니다



File의 Schema탭은 다음 항목을 설정합니다.

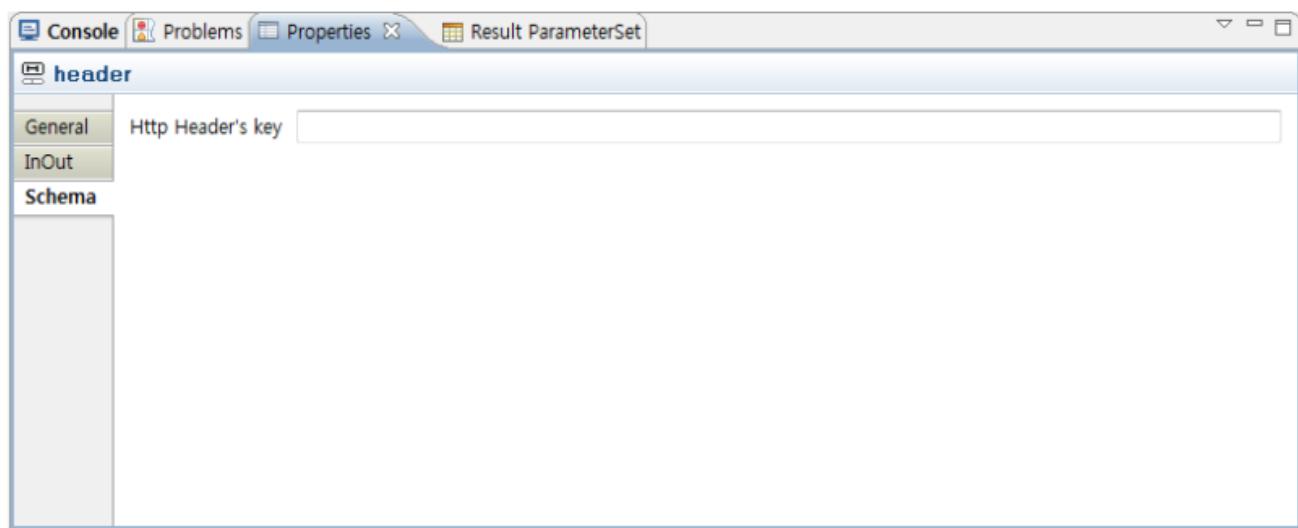
- Content Type : File content type을 정의합니다.
- 아래 표는 몇가지 양식을 보여줍니다.

MIME Type	File Extension
application/pdf	pdf
image/jpeg	jpeg
text/html	html
text/css	css

Header

Header Parameter는 HTTP Header 입출력 정보를 관리하기 위한 파라미터입니다.

기본적으로 다른 파라메터와 동일한 프로퍼티를 가지고 있으며 Schema탭만 다르게 정의합니다



Header의 Schema탭은 다음 항목을 설정합니다.

- Http Header's key : Http header key 값을 정의합니다.
- 아래 표는 몇가지 양식을 보여줍니다.

Field Name	Example
Accept	Accept: text/plain
Accept-Charset	Accept-Charset: utf-8
Content-Type	Content-Type: application/x-www-form-urlencoded
User-Agent	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox /21.0

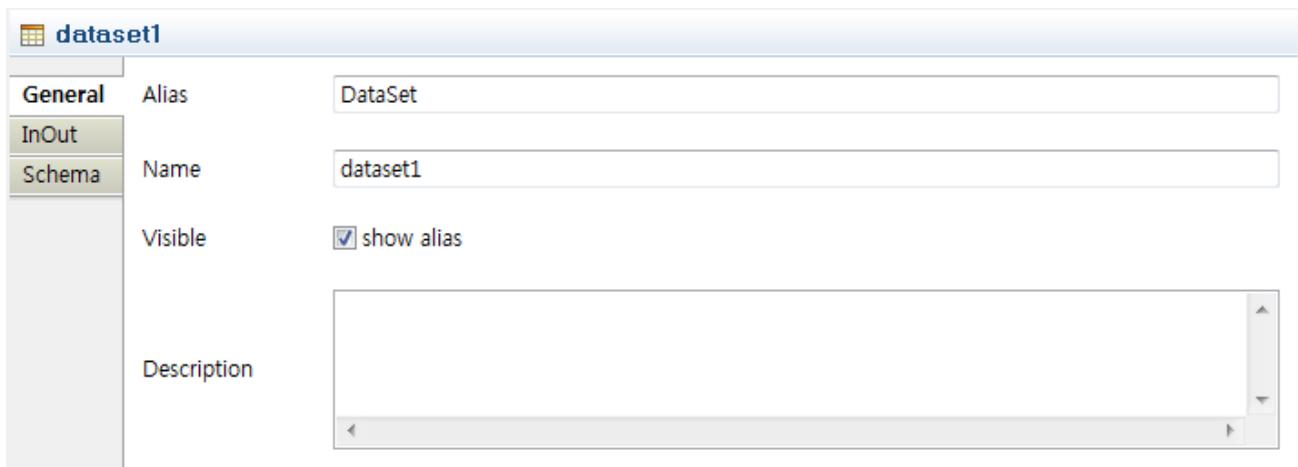
DataSet

DataSet Parameter는 X-API의 DataSet과 동일한 타입의 파라메터를 생성합니다.

데이터셋 프로퍼티는 다음 세가지 프로퍼티 탭을 가지고 있습니다.

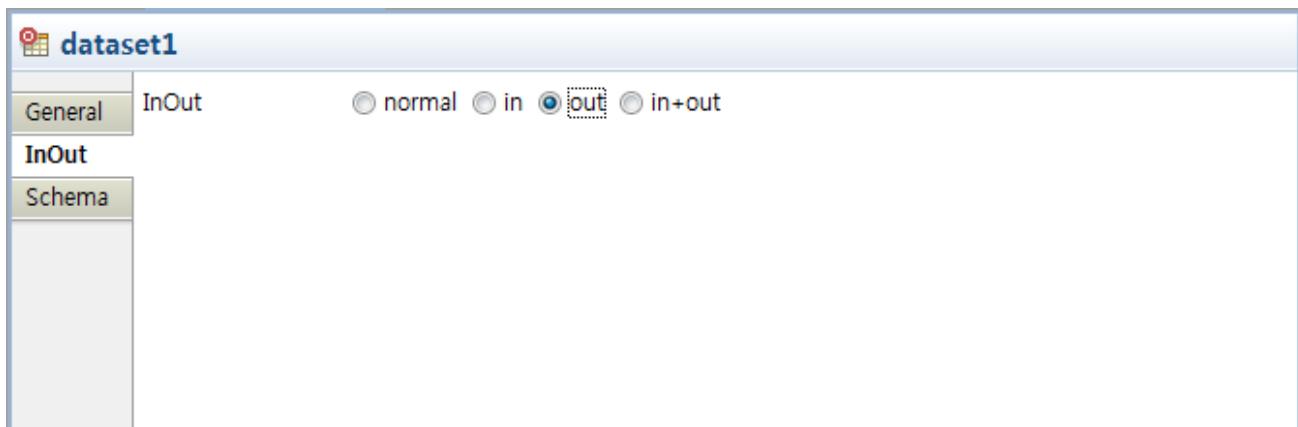
- General
- InOut
- Schema

General은 모든 노드들의 공통항목으로 name, alias, visible, description 을 정의합니다

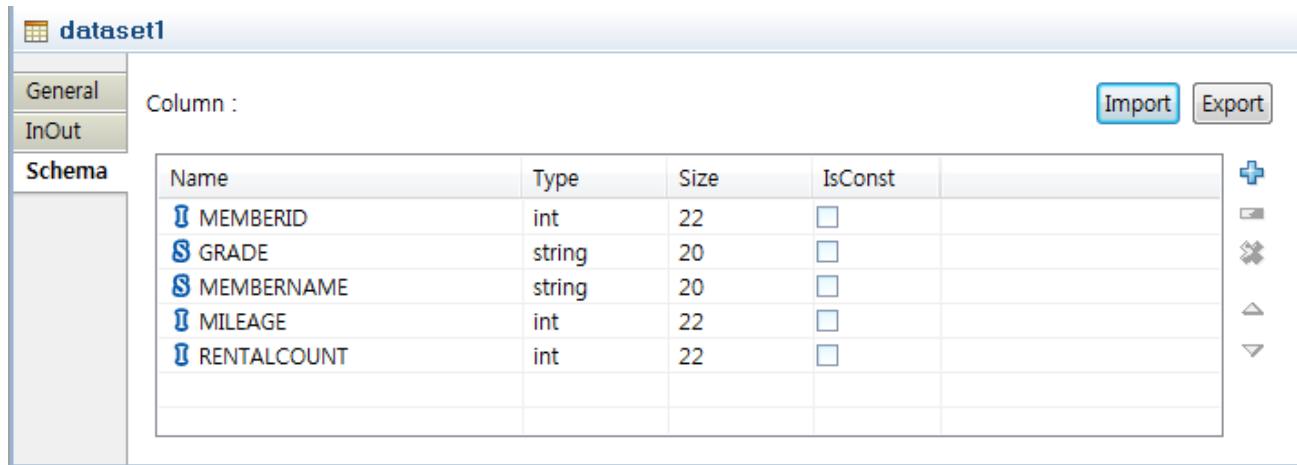


- Alias : 사용자가 원하는 별칭을 정의합니다. 한글입력도 가능합니다.
- Name : 데이터셋의 이름을 정의합니다. 에디터상에서 다른 파라메터와 중복되지 않는 이름으로 영어로만 정의합니다.
- Visible : 에디터상에서 Name을 보일지 Alias를 보일지를 선택합니다.
- Description : 설명을 입력합니다.

InOut은 모든 파라메터의 공통항목으로 파라메터의 타입을 선택합니다.

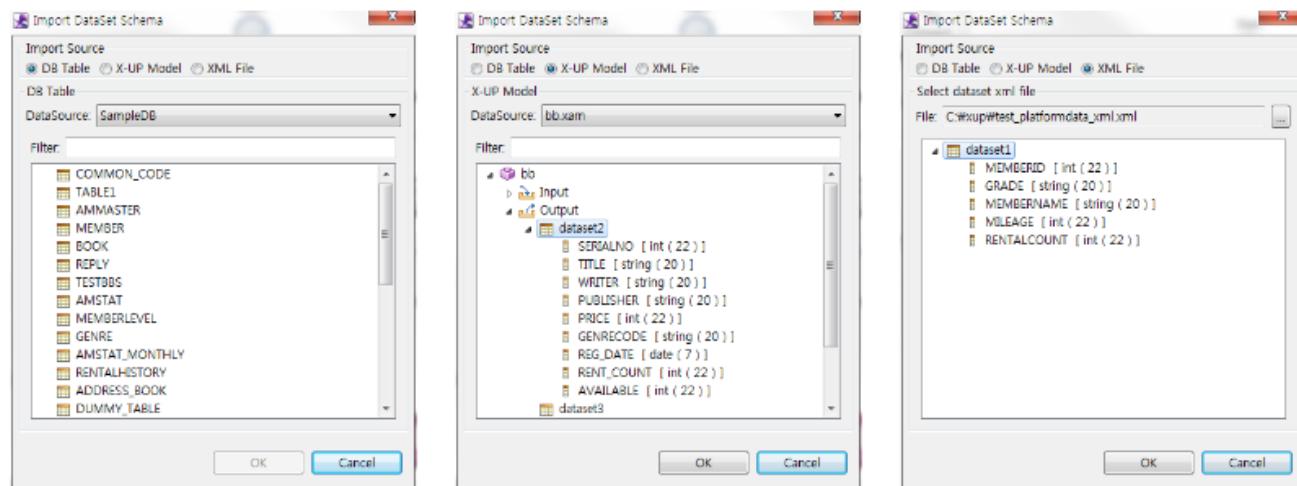


- normal : 모델내부에서만 사용되는 일반 파라메터
- in : 모델의 input parameter로 설정, 모델 메타데이터의 input 파라메터로 등록됩니다.
- out : 모델의 output parameter로 설정, 모델 메타데이터의 output 파라메터로 등록됩니다.
- in + out : 모델의 input parameter + output parameter로 설정, 모델 메타데이터의 input + output 파라메터로 등록됩니다.



Schema는 데이터셋의 컬럼 정보를 정의합니다. 사용자가 직접 컬럼 정보를 등록가능하며 Import / Export을 활용하여 외부로부터 가져오거나 내보낼 수 있습니다.

Import 버튼 클릭시 나오는 Import DataSet Schema Dialog는 다음 3곳에서 데이터셋 컬럼 정보를 가져올 수 있습니다



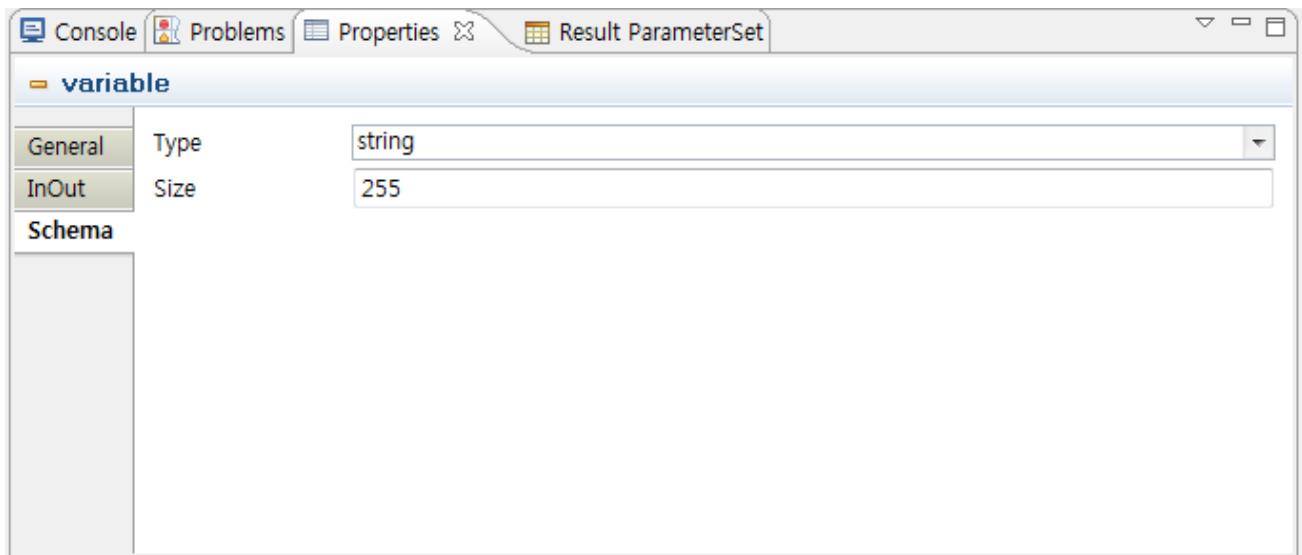
- DB Table : 등록된 데이터소스에 존재하는 데이터베이스 테이블로부터 컬럼 정보를 생성합니다.
- X-UP Model : 생성된 다른 모델로부터 데이터셋 컬럼 정보를 가져옵니다.
- XML File : 로컬 PC에 존재하는 PlatformData xml 파일로부터 데이터셋 컬럼 정보를 생성합니다.

Export는 등록된 데이터셋 컬럼 정보를 PlatformData 타입의 xml 파일로 로컬 PC에 저장하여 다양한 곳에서 재사용 가능하도록합니다.

Variable

Variable은 X-API의 Variable과 동일한 타입의 파라메터를 생성합니다.

기본적으로 다른 파라메터와 동일한 프로퍼티를 가지고 있으며 Schema탭만 다르게 정의합니다.



Variable의 Schema탭은 다음 두가지 항목을 설정합니다.

- Type : variable 파라미터의 타입을 정의합니다.
 - string / int / float / long / boolean / bigdecimal / date / time / datetime / blob
- Size : variable 파리미터의 사이즈를 정의합니다.

6.3 Transaction Model

트랜잭션 모델은 DBMS와의 연동을 목적으로 하는 모델이며, 다음과 같은 기능을 제공합니다.

- DB Binding: DB 바인딩을 위한 코드
- Transaction JDBC: DB와의 조회를 위한 JDBC 코드
- DB procedure Calling: DB의 프로시저를 호출하기 위한 코드
- X-UP Model Invoking: 기존 만들어진 X-UP 모델을 호출하기 위한 코드

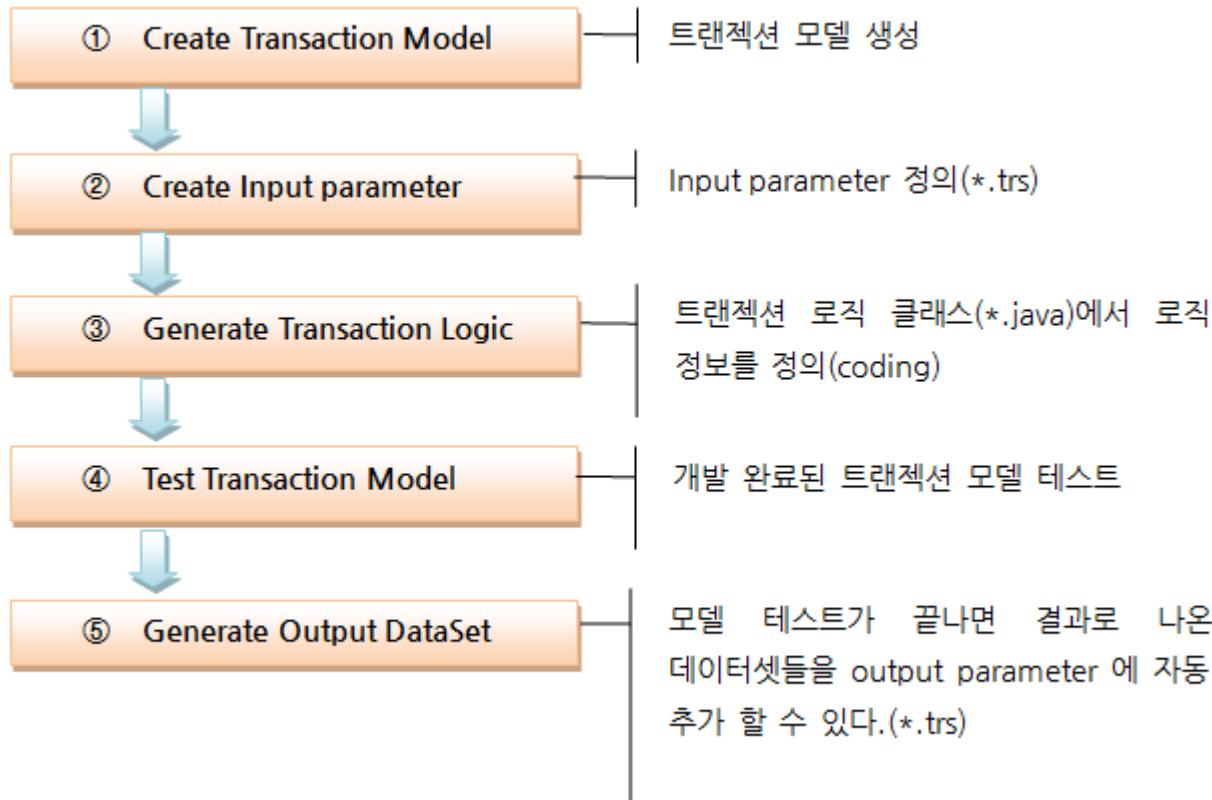
트랜잭션 모델은 DB와의 다양한 query를 지원하기 위한 모델입니다. 사용자는 트랜잭션 모델안에 임의의 로직을 Java로 구현할 수 있으며, 이러한 로직은 모델로 관리됩니다.

트랜잭션 모델은 DB 처리가 목적이지만, DB와 관계없는 사용자 임의의 로직을 구현하기 위한 용도로도 사용될 수 있습니다. 이러한 특성으로 인해 Automation 모델은 GUI 에디터에 의해서 개발되는데 반해 트랜잭션 모델은 기본적인 java 클래스 코드를 자동생성하고 java editor에 의하여 사용자가 수정하는 방식으로 개발합니다. 일단 클래스에 해당하는 코드가 자동 생성되고 난 후, 사용자는 DB와의 연동을 위한 코드를 위자드에 의하여 역시 자동 생성할 수 있습니다. 일반적인 트랜잭션모델의 개발은 다음과 같이 진행합니다.

- 입력과 출력을 정의
- 코드 생성 위자드에 의한 코드 생성

- 기타 코드 수정

트랜잭션 모델을 생성하는 과정은 일반적으로 다음과 같습니다.



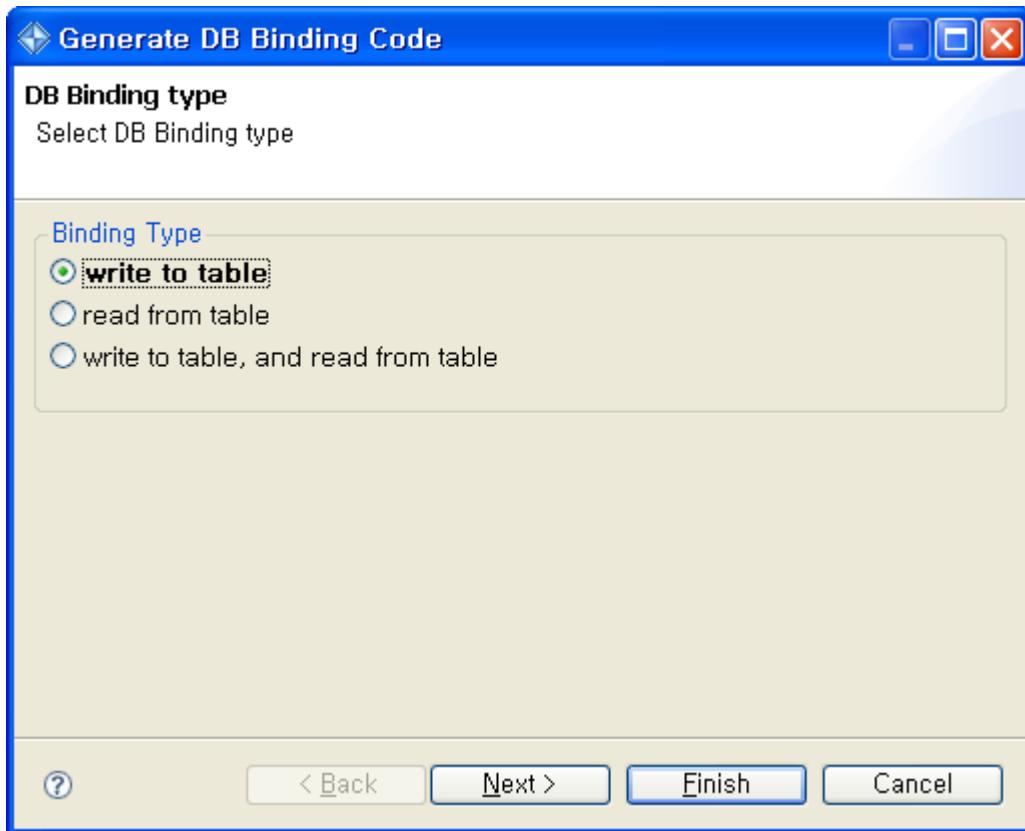
DB Binding

DB Binding은 데이터셋과 데이터베이스의 테이블의 데이터를 동기화하는 기능입니다. 구체적으로 데이터셋의 row를 테이블에 insert, update, delete하여 반영할 수 있으며, 테이블의 record를 select하여 데이터셋에 반영할 수 있습니다.

X-UP Builder의 Generate DB Binding Code는 DB 바인딩 소스 코드를 생성하기 위한 위자드입니다.

Generate DB Binding Code는 다음 세가지 타입으로 구성되어 있습니다.

- write to table
- read from table
- write to table, and read from table

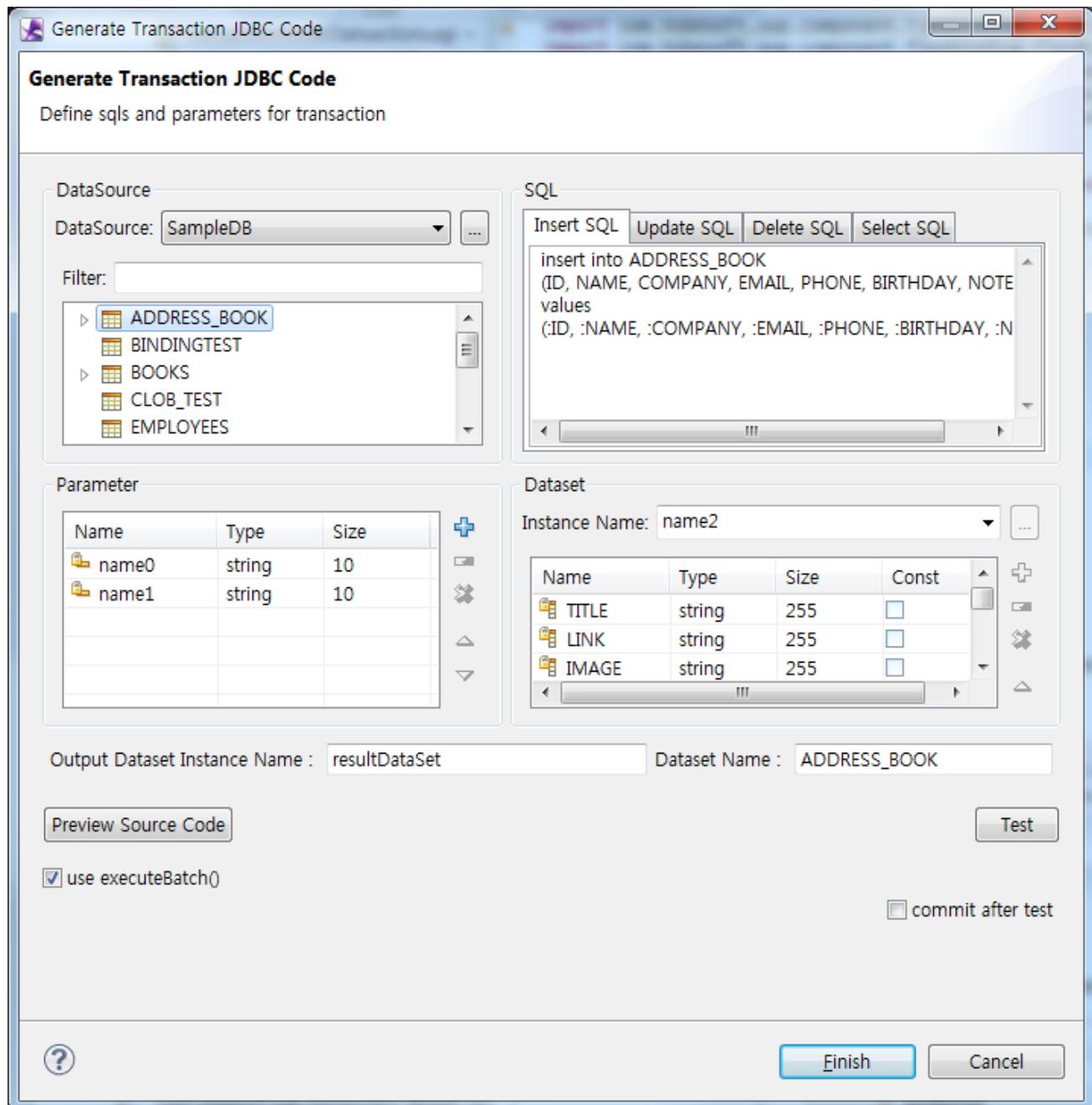


INSERT, UPDATE and DELETE

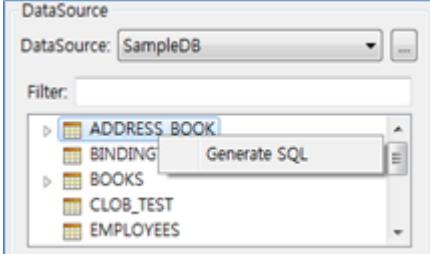
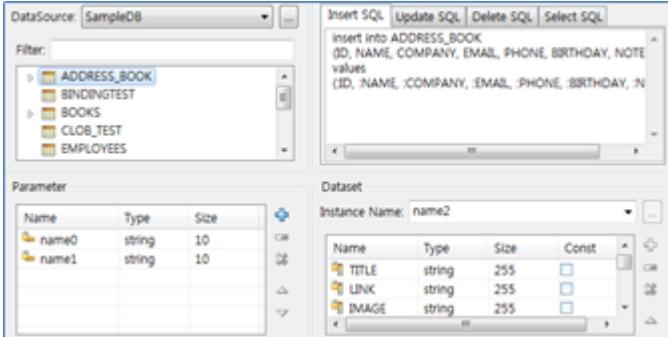
X-UP은 DB Binding의 write시에 데이터셋의 row의 type을 보고 해당 row를 insert, update, delete할지를 결정합니다. 데이터셋의 row의 type은 강제로 설정할 수도 있으나 일반적으로 데이터셋에 row를 추가하거나 수정, 삭제할 때 자동적으로 type이 결정됩니다.

Transaction JDBC

Generate Transaction JDBC Code는 사용자로 하여금 자유로운 insert / update / delete sql을 입력을 받고 일반 jdbc 소스 코드를 생성합니다.

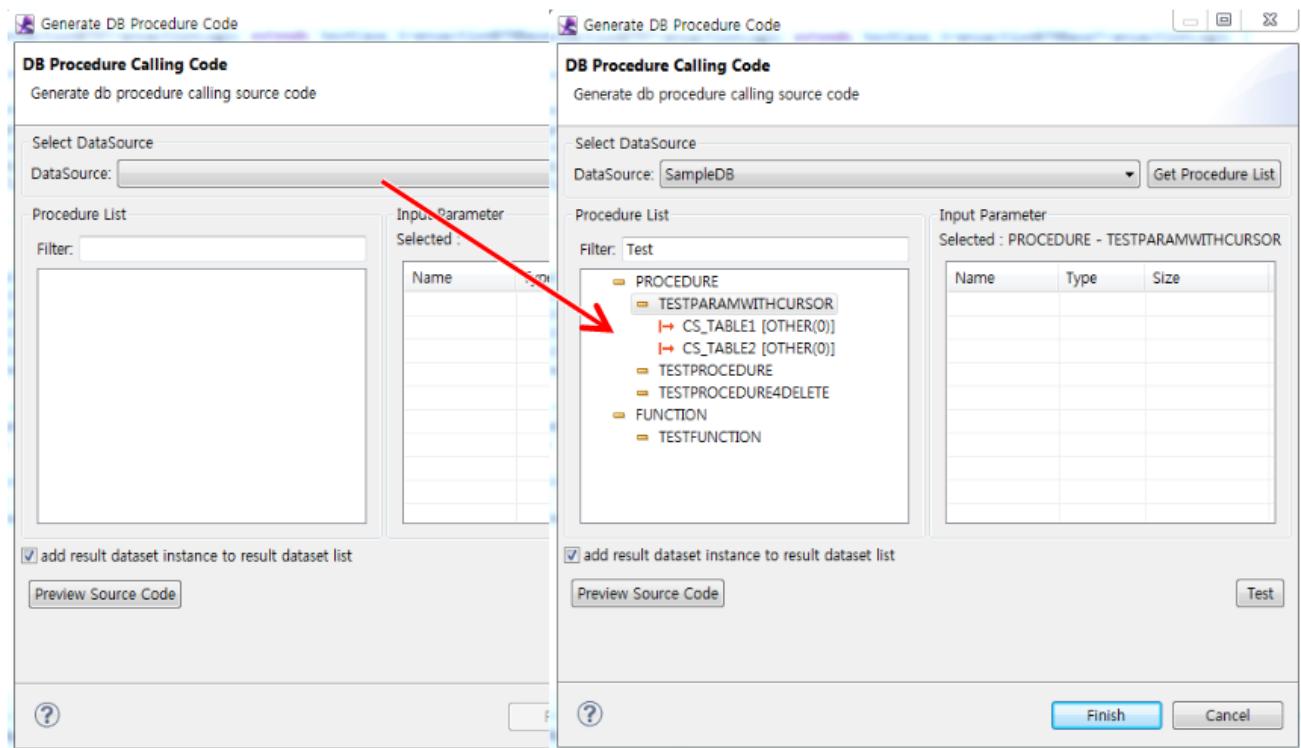


Name	Description
DataSource	데이터소스를 선택합니다.
DB Tables	질의문을 작성하고자 하는 데이터베이스 테이블을 선택합니다.
SQL	<p>Insert / Update / Delete 질의문을 작성합니다. 질의문은 사용자가 직접 작성하거나 DB Table에서 제공하는 기본 질의문 작성 메뉴를 통해 생성할 수도 있습니다. DB Table의 Generate SQL을 통해 생성된 질의문은 해당 테이블의 컬럼 정보를 통해 생성된 샘플 SQL이므로 사용자가 원하는 질의문인지 확인해야 합니다. 특히 update와 delete의 경우는 where 조건절 생성시 디폴트로 첫번째 컬럼이름으로 생성되므로 사용자가 직접 원하는 조건문으로 수정하여 사용하길 바랍니다.</p> <p>질의문에서 변수를 사용하는 예는 다음과 같습니다.</p>

Name	Description
	<p>예: update cat set color = :color, isOld = :isOld where age > :age 질의문의 변수명을 포함하기 위해서 반드시 문자열 앞에 “:” 를 추가한 다음 파라메터나 데이터셋 스키마의 이름과 동일한 이름을 질의문 변수로 사용합니다.</p>   <p>또는 Parameters 테이블의 특정 컬럼이나 데이터셋 테이블의 특정 컬럼을 선택한 후 질의문 작성란에 드래그 앤 드랍을 하면 자동으로 해당 컬럼이름을 변수명으로 삽입하여 줍니다.</p>
Parameter	Transaction Editor에 정의된 데이터셋을 제외한 Input 파라메터가 디플트로 나옵니다. 사용자가 직접 추가할 수도 있습니다.
DataSet Instance Name	Transaction Editor에 정의된 데이터셋 인스턴스 리스트 중 하나를 선택합니다. 없어도 사용자가 직접 인스턴스 이름을 입력할 수 있습니다.
DataSet Schema	데이터셋 스키마 정보를 출력합니다.
DataSet Rows	입력된 데이터셋 스키마정보를 토대로 row 정보가 자동 생성됩니다. 사용자는 각 row 에 해당하는 테스트값만 입력하면 됩니다.
Test	테스트를 합니다.
Test Result	테스트 결과를 출력합니다.

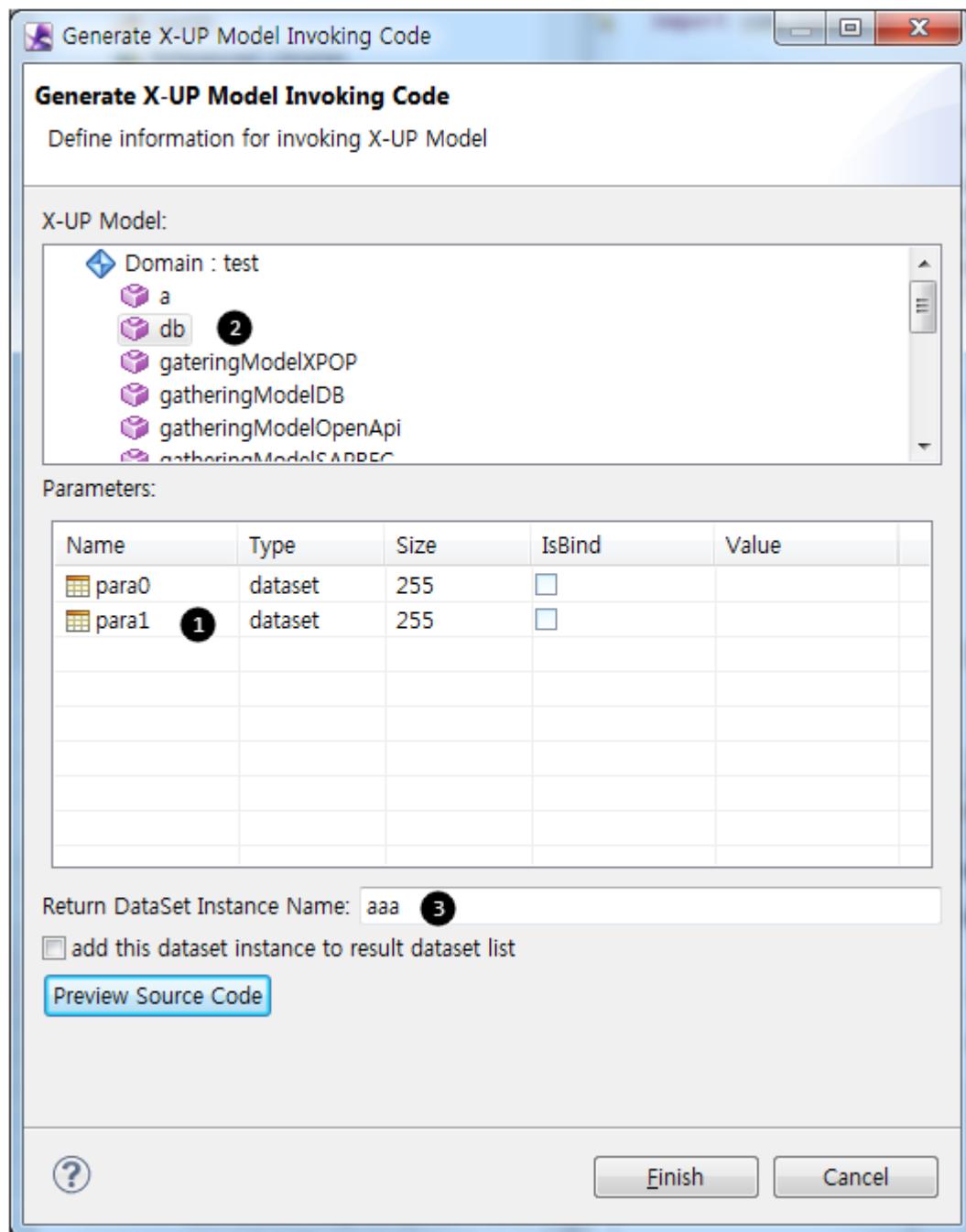
DB Procedure Calling Code

DB Procedure Calling Code는 데이터베이스의 프로시저의 정보를 가져오는 소스코드를 생성합니다. 호출할 프로시저에 대해 데이터소스를 선택하고 호출할 프로시저를 선택합니다.



X-UP Model Invoking Code

미리 정의된 X-UP Model 의 데이터셋 정보를 가져오는 소스코드를 생성합니다. 선택한 X-UP 모델을 Invoking 하기 위해 필요한 파라메터 정보와 Value 를 입력하고 리턴된 데이터셋 인스턴스 이름을 입력합니다.



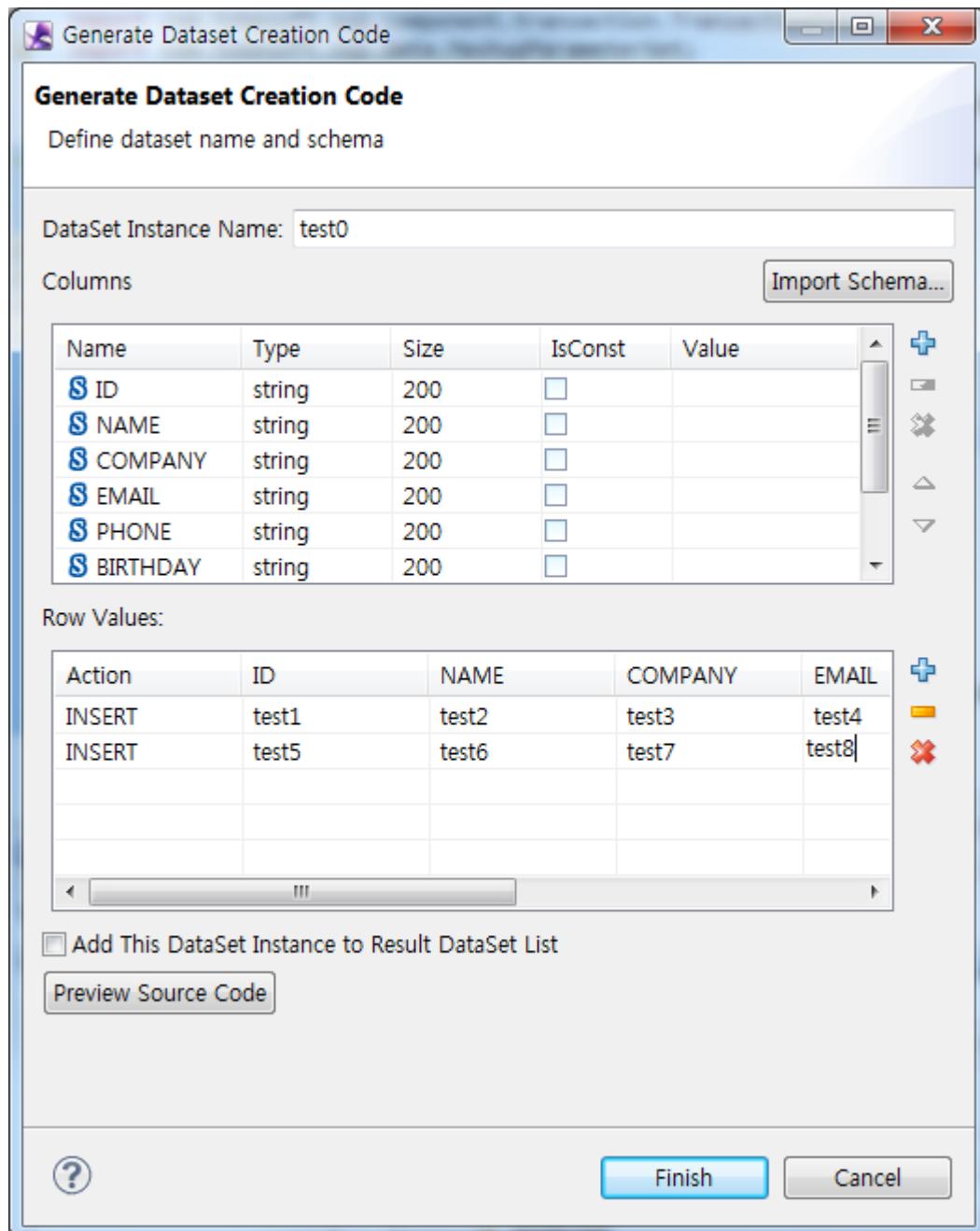
X-UP Model Invoking 과정은 다음과 같습니다.

1. X-UP Model 트리에서 원하는 모델을 선택합니다.
2. 선택된 모델을 호출하기 위해서 필요로 하는 파라미터의 값을 등록합니다.
3. 생성될 데이터셋 인스턴스 이름을 입력합니다.

DataSet Creating Code

데이터셋을 생성하는 소스코드를 자동 생성합니다.

사용자는 데이터셋 인스턴스 이름을 입력하고 Import Schema Dialog를 통하여지 아니면 직접 입력해서 데이터셋 스키마정보를 등록하고, 필요에 따라 샘플 row도 등록할 수 있습니다.



Error Code, Message 처리

X-UP뿐 아니라 (주)투비소프트의 MiPlatform, XPLATFORM 모두 트랜잭션 수행 시 서비스 호출의 성공 및 실패 여부를 CallBack 메서드의 ErrorCode, ErrorMsg(ErrorMessage)를 통해 확인할 수 있습니다.

기본 설정의 경우 예외가 발생하였을 때 ErrorCode(-1) ErrorMsg(에러메세지)를 전달합니다.

Parameter or Exception을 이용하여 별도의 ErrorCode, ErrorMsg를 전달할 수 있습니다.

Parameter를 이용한 ErrorCode, ErrorMsg 처리

에러 처리 할 모델의 출력 파라미터에 ‘ErrorCode’, ‘ErrorMsg’를 추가합니다. 단, 파라미터에 ‘ErrorCode’가 음수일 경우 에러로 판단하여 전체 트랜잭션에 rollback 처리를 수행함으로 주의하여야 합니다.

예외발생 시 ErrorCode 변경

X-UP에서 기본적으로 생성하는 ErrorCode ‘-1’ 값의 변경이 필요한 경우에 처리할 수 있습니다. 예외 발생 시 예외에 ErrorCode를 추가하여 처리합니다. 아래는 해당 설명에 대한 예시로 샘플 코드입니다.

```
AutomationFailException exception = new AutomationFailException( "데이터 조회에  
실패하였습니다. " );  
exception.setErrorCode( "-2381623" );
```

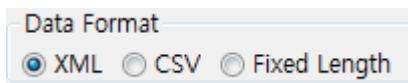
6.4 데이터 입-출력

X-UP은 HTTP 프로토콜 기반의 통신을 지원합니다. HTTP 프로토콜을 기반으로 사용하는 프로토콜인 WEB-Service 방식과 (주)투비소프트 자체 프로토콜인 DataSet 프로토콜, 마지막으로 일반적인 HTTP 프로토콜 방식인 GET/POST 방식을 지원합니다.

외부로부터 획득한 데이터는 X-UP 내부의 자료유형인 Parameter 객체로 변환되어 이용자에게 전달되며 File에 대한 처리는 모델 로직에서 사용자가 처리를 하게 됩니다.

6.5 데이터 포맷 및 파싱

원시데이터를 보고 어떤 형태의 데이터 포맷인지 결정합니다. X-UP에서는 다음 세가지 데이터 파싱 타입을 제공합니다.



XML 데이터 파싱

A screenshot of the 'Parsed Data - XML' interface. On the left is the 'XML Tree' panel, which shows the structure of an XML document with nodes like rss, version, channel, title, link, description, lastBuildDate, total, start, display, item, and item. A red arrow labeled 'Drag and drop' points from the tree to the 'Target XPath' panel on the right. The 'Target XPath' panel contains a text input field with '/rss/channel/item' and a checkbox for 'include attribute'. Below these are several suggested XPath expressions: /rss/channel/item/title, /rss/channel/item/link, /rss/channel/item/image, /rss/channel/item/author, /rss/channel/item/price, /rss/channel/item/discount, /rss/channel/item/publisher, /rss/channel/item/pubdate, and /rss/channel/item/chn. A second red arrow points from the 'Target XPath' panel back to the 'XML Tree' panel.

원시 데이터가 xml 타입일 경우 xml path 설정을 통해 원시 데이터셋을 생성합니다.

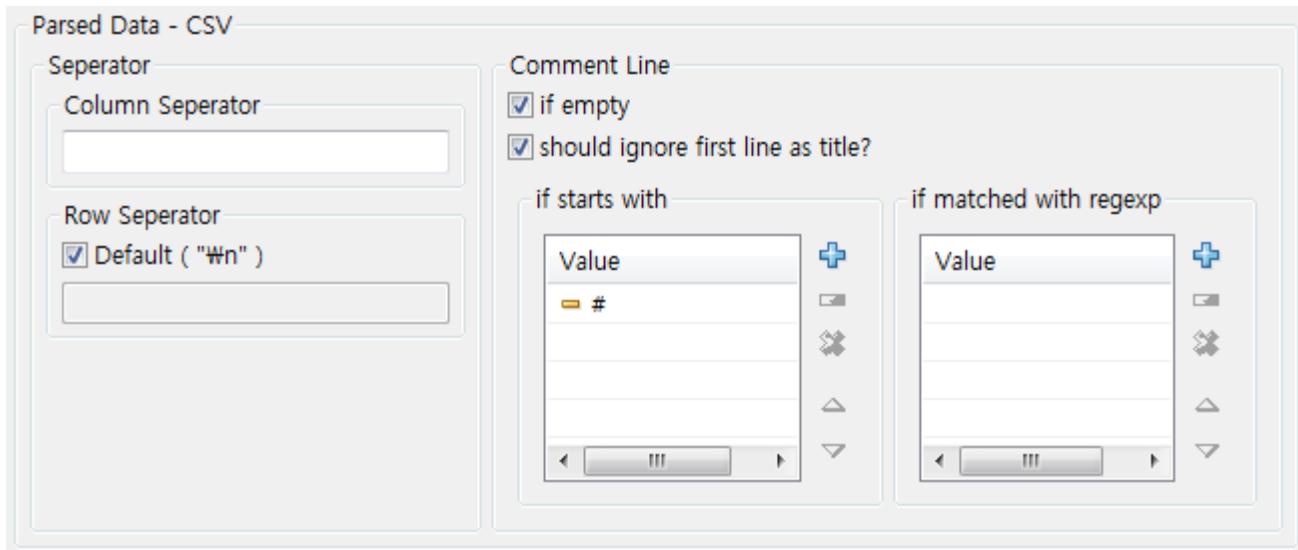
XML 데이터 포맷 정보 설정과정은 다음과 같습니다.

1. 오리지널 문자열(xml)이 분석되어 생성된 Tree 정보 확인
2. 출력된 XML Tree에서 특정 노드를 선택하여 Target XPath로 드래그 합니다.
3. 자동으로 생성된 각 컬럼별 xpath를 확인합니다.
4. Result Table에 원시데이터셋이 원하는 결과로 생성되었는지 확인합니다.

include attribute 가 체크된 이후에 row xpath를 드래그 앤 드랍 하게 되면 하위 모든 자식노드들을 자동으로 추가 할 때 attribute도 포함합니다.

CSV 데이터 파싱

원시 데이터가 CSV 타입일 경우 column separator, row separator, comment line 등의 설정을 통해 원시 데이터셋을 생성합니다.

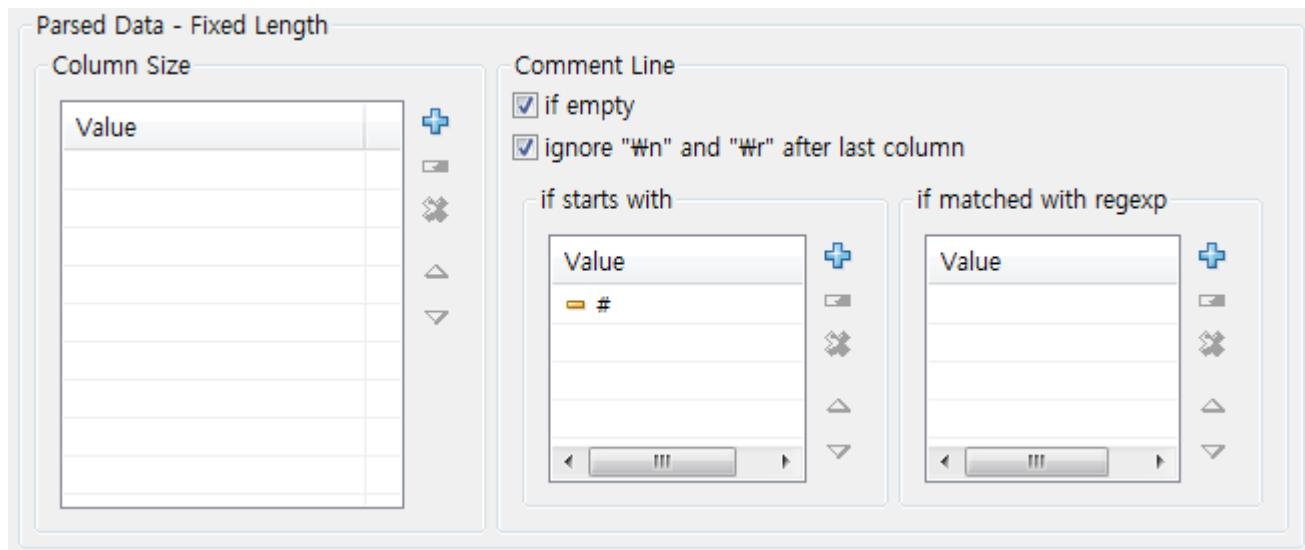


CSV 데이터 포맷 정보 설정과정은 다음과 같습니다.

1. Column Seperator (문자열 구분자)를 등록합니다.
2. Row Seperator (문단 구분자)를 등록합니다. (디폴트로 '\n' 이 설정됨)
3. 체크박스를 해제하고 직접 입력 가능합니다.
4. 특정 문자로 시작하거나 비어있는 라인을 파싱에서 제외하고자 한다면 다음 설정을 합니다.
 - I. Empty Line : 비어있는 라인이 있다면 파싱에서 제외합니다.
 - II. Start With : 등록된 문자로 시작하면 파싱에서 제외합니다.
 - III. Regular Expression : 등록된 regular express 문자열에 해당하는 패턴이 있다면 파싱에서 제외합니다.

고정 길이 데이터 파싱

원시 데이터가 Fixed Length 타입일 경우 column length 등의 설정을 통해 원시 데이터셋을 생성합니다.



Fixed Length 데이터 포맷 정보 설정과정은 다음과 같습니다.

1. 원시 Raw Data 를 보고 Column Length 를 순서대로 입력합니다.
2. 특정 문자로 시작하거나 비어있는 라인을 파싱에서 제외하고자 한다면 다음 설정을 합니다.
 - I. Empty Line : 비어있는 라인이 있다면 파싱에서 제외합니다.
 - II. Start With : 등록된 문자로 시작하면 파싱에서 제외합니다.
 - III. Regular Expression : 등록된 Regular express 문자열에 해당하는 패턴이 있다면 파싱에서 제외합니다.

6.6 InvokingInfo

X-UP Automation 모델에서는 Invoke에 필요한 특정 정보를 Properties View에서 입력을 받게 됩니다. Properties View에 입력 된 정보를 토대로 java코드를 자동으로 생성하여 Invoke를 수행하게 됩니다. 이때 Invoke마다 입력받는 properties는 약간의 차이가 있습니다.

7.

Data Source 정의

X-UP은 다양한 데이터소스로부터 다양한 형식의 데이터를 수집하고, 이를 가공하여 새로운 데이터로 생성합니다. 이 장에서는 X-UP에서 지원되는 데이터소스에 대해 설명합니다.

X-UP이 지원하는 데이터소스는 다음과 같습니다.

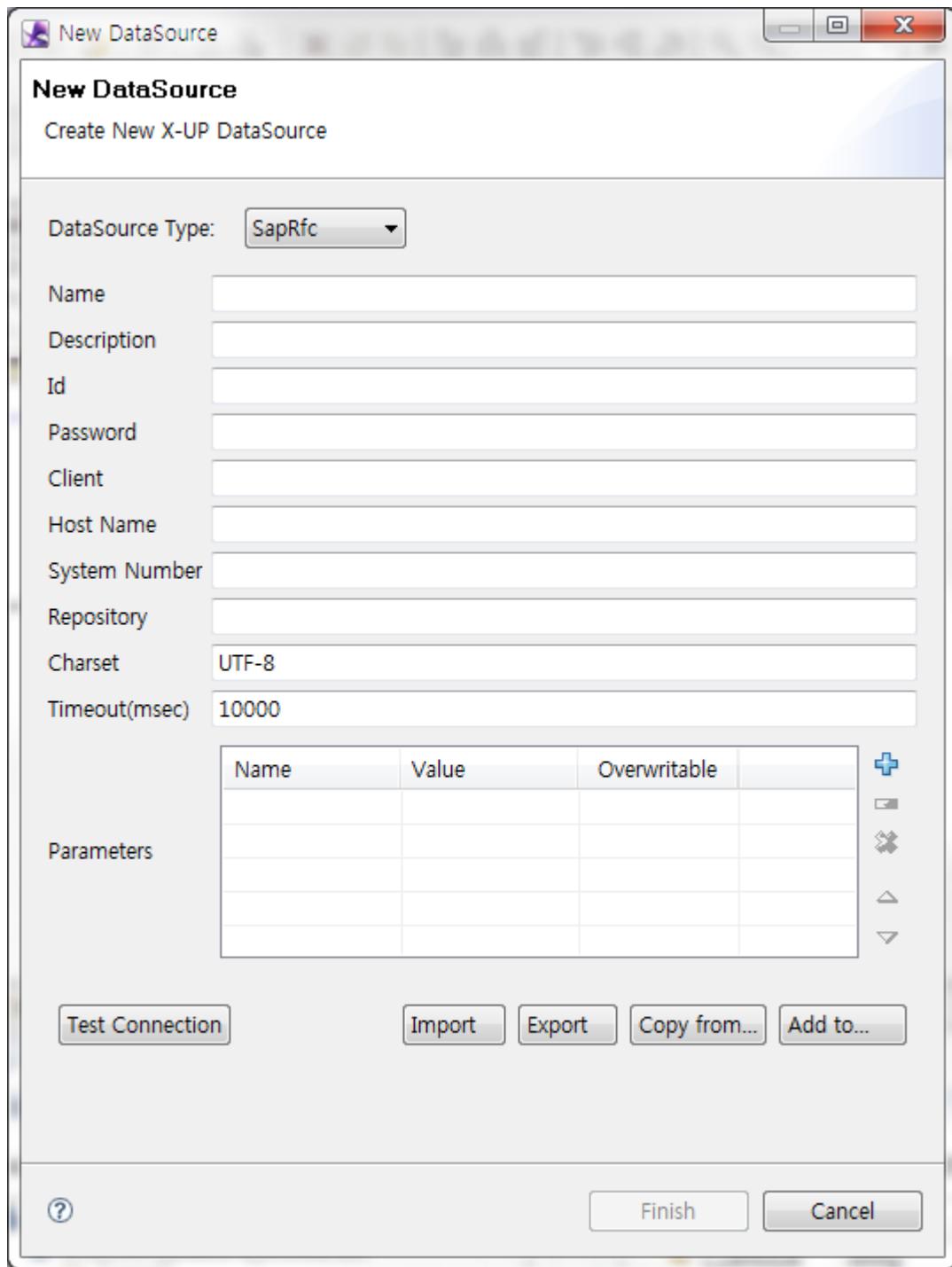
- SAP RFC
- DB (Data Base)
- OpenApi
- UDDI
- WebService
- X-UP

데이터소스는 데이터소스에 접속하기 위한 정보의 설정으로 이루어집니다. 접속하기 위한 정보는 데이터소스의 타입에 따라 다릅니다.

7.1 SAP RFC 데이터 소스

[Select X-UP Project > X-UP > new X-UP DataSource]를 선택하면 새로운 데이터소스를 생성할 수 있는 화면이 나옵니다.

SAP RFC 데이터소스를 작성하려면 DataSource Type 을 'SapRfc' 로 선택하면 해당 데이터소스 정보를 입력할 수 있습니다.

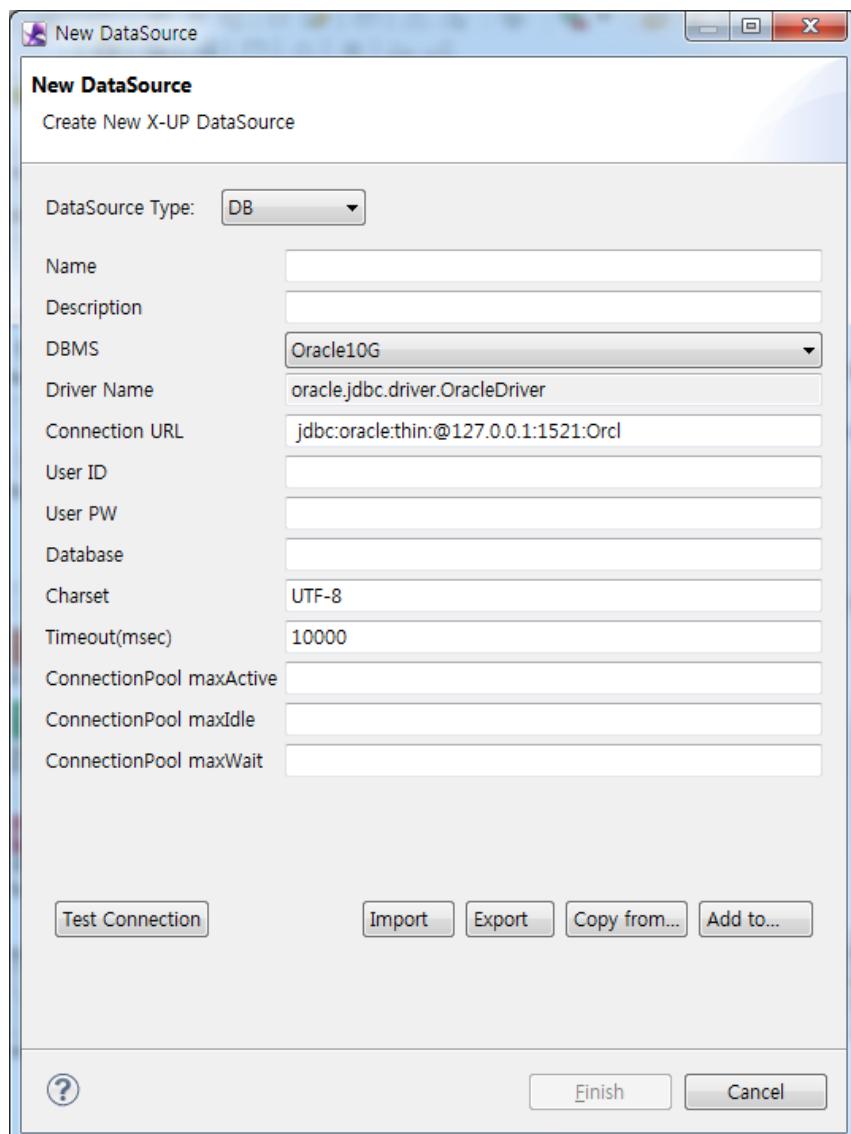


Name	Description
Name	데이터소스 이름
Description	설명
Id	아이디
Password	패스워드
Client	SAP RFC Client 정보
Host Name	SAP RFC Host Name 정보
System Number	SAP RFC System Number 정보

Name	Description
Repository	SAP RFC Repository 정보
Charset	인코딩 (default : UTF-8)
Timeout(msec)	장애시 대기시간 (1000 이면 1초)
Parameters	서비스 연결시 필요한 파라메터

7.2 DB 데이터 소스

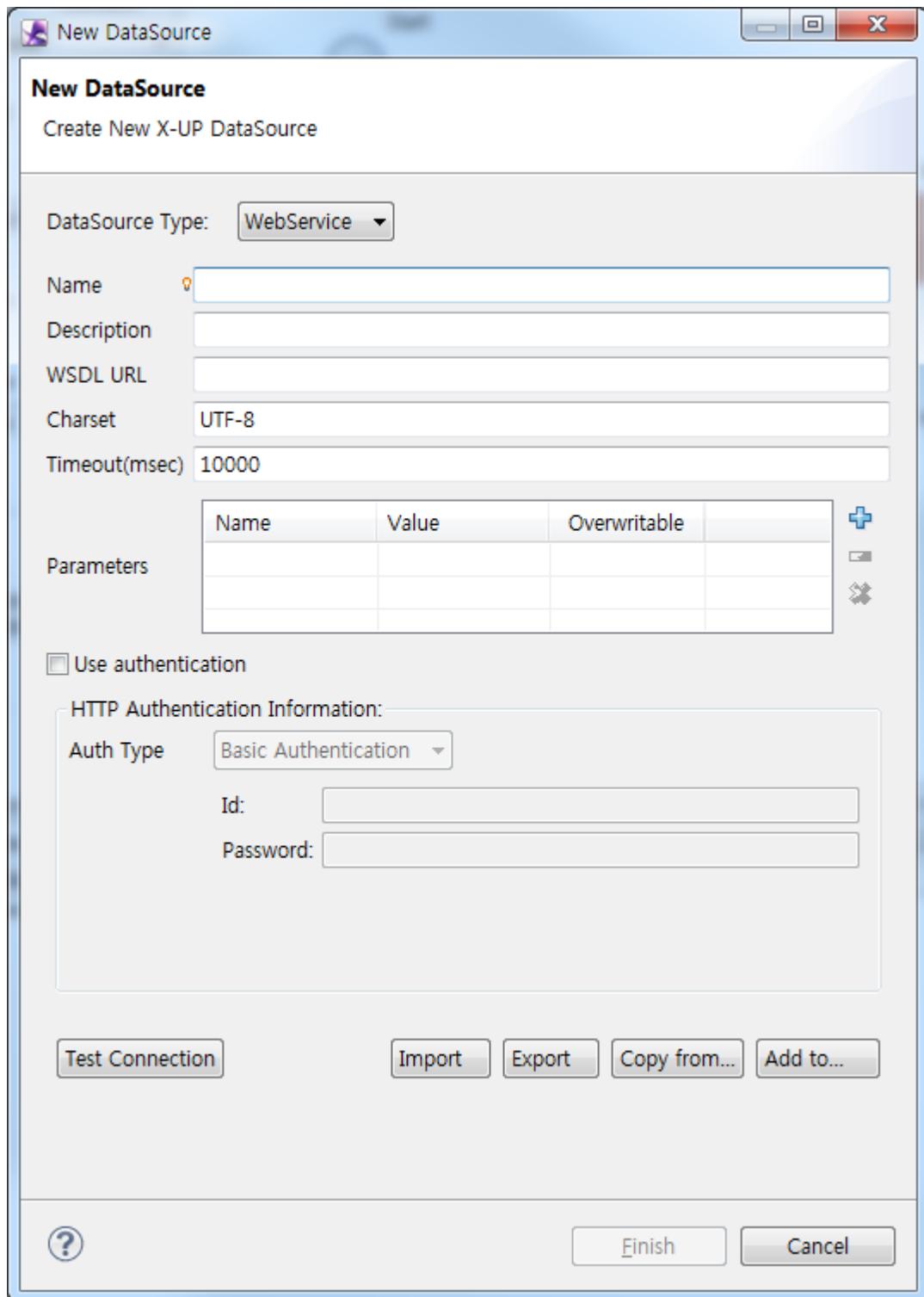
[Select X-UP Project > X-UP > new X-UP DataSource]를 선택하면 새로운 데이터소스를 생성할 수 있는 화면이 나옵니다. DB 데이터소스를 작성하려면 DataSource Type 을 DB 로 선택하면 해당 데이터소스 정보를 입력할 수 있습니다.



Name	Description
Name	데이터소스 이름
Description	설명
DBMS	Oracle, MSSQL, MySql, DB2 중 택1
Driver Name	드라이버 이름 (예: oracle.jdbc.driver.OracleDriver)
Connection URL	커넥션 URL (예: jdbc:oracle:thin:@127.0.0.1:1521:dbms)
User ID	아이디
User PW	패스워드
Database	데이터베이스 이름
Charset	캐릭터 셋(default : UTF-8)
Timeout(msec)	장애시 대기시간 (1000 이면 1초)
ConnectionPool maxActive	ConnectionPool 최대 Connection 수
ConnectionPool maxIdle	ConnectionPool 최소 Connection 수

7.3 WebService 데이터 소스

[Select X-UP Project > X-UP > new X-UP DataSource]를 선택하면 새로운 데이터소스를 생성할 수 있는 화면이 나옵니다. WebService 데이터소스를 작성하려면 DataSource Type 을 **WebService** 로 선택하면 해당 데이터소스 정보를 입력할 수 있습니다.

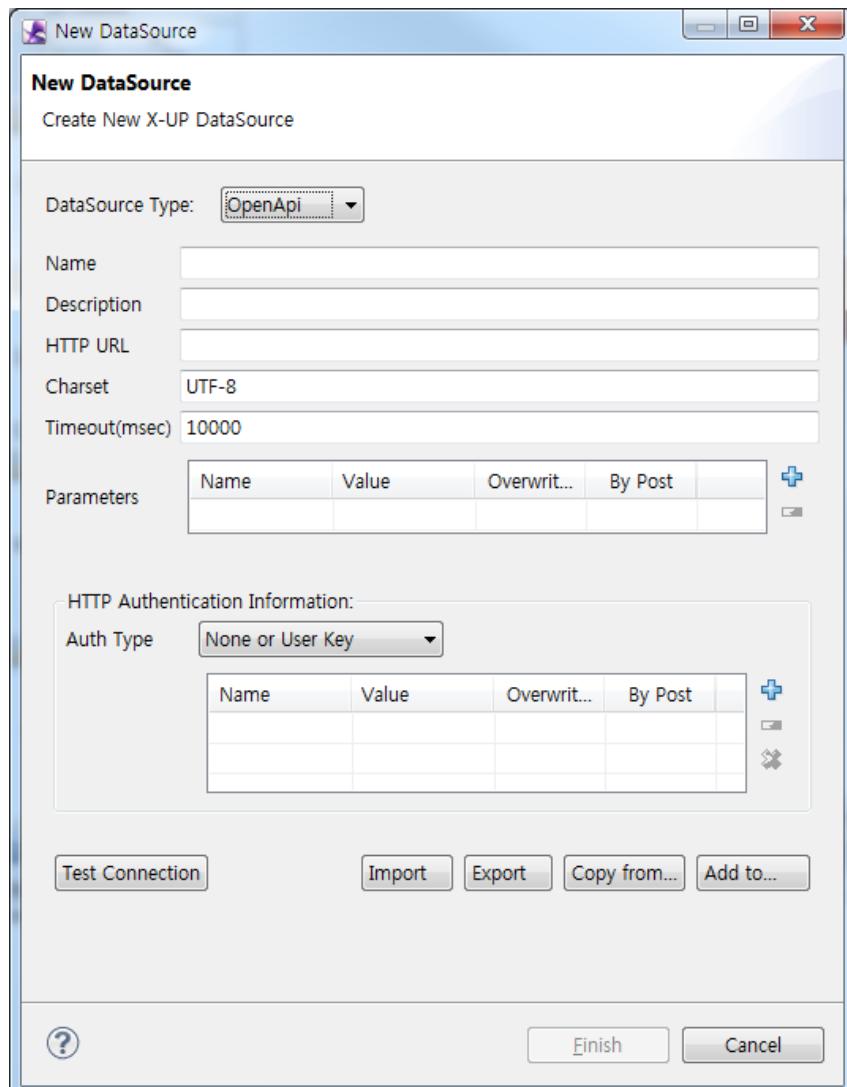


Name	Description
Name	데이터소스 이름
Description	설명
WSDL URL	웹서비스에 연결한 WSDL 주소
Charset	인코딩 (default : UTF-8)
Timeout(msec)	장애시 대기시간 (1000 이면 1초)

Name	Description
Parameters	서비스 연결시 필요한 파라메터
Use authentication	HTTP auth가 필요시 체크
Auth Type	Auth Type을 선택
User ID	HTTP auth를 위한 유저 아이디로써 만약 필요하다면 등록
User PW	HTTP auth를 위한 유저 패스워드로써 만약 필요하다면 등록

7.4 OpenApi 데이터 소스

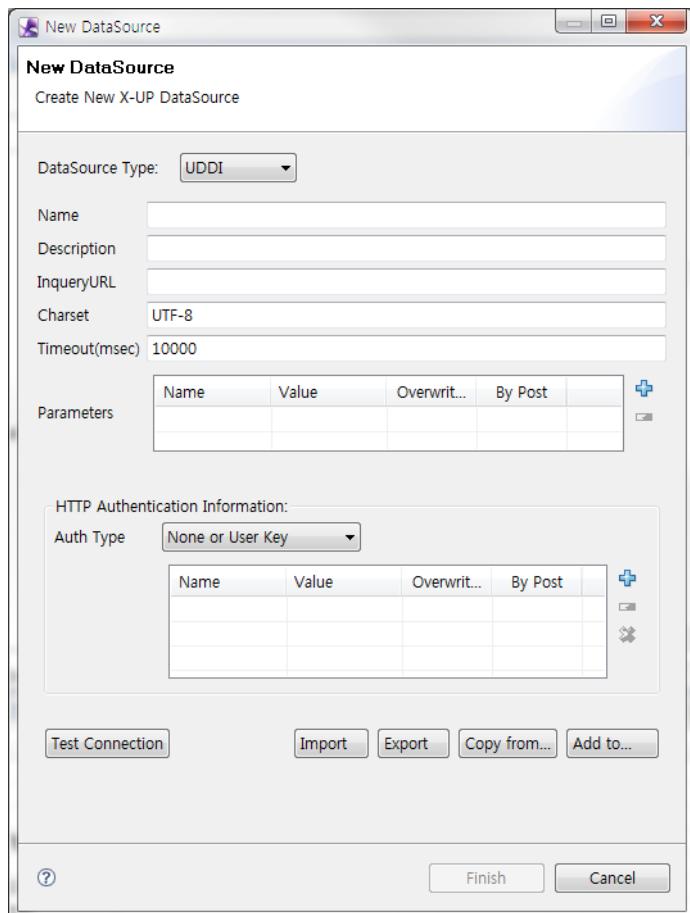
[Select X-UP Project > X-UP > new X-UP DataSource]를 선택하면 새로운 데이터소스를 생성할 수 있는 화면이 나옵니다. OpenApi 데이터소스를 작성하려면 DataSource Type 을 **OpenApi** 로 선택하면 해당 데이터소스 정보를 입력할 수 있습니다.



Name	Description
Name	데이터소스 이름
Description	설명
HTTP URL	연결하고자 하는 HTTP URL
Charset	인코딩 (default : UTF-8)
Timeout(msec)	장애시 대기시간 (1000 이면 1초)
Parameters	서비스 연결시 필요한 파라미터
Auth Type	Auth Type 선택 (None or User Key / Basic Authentication / User Define Login Page)
User ID	HTTP auth를 위한 유저 아이디로써 만약 필요하다면 등록

7.5 UDDI 데이터 소스

[Select X-UP Project > X-UP > new X-UP DataSource]를 선택하면 새로운 데이터소스를 생성할 수 있는 화면이 나옵니다. UDDI 데이터소스를 작성하려면 DataSource Type 을 UDDI 로 선택하면 해당 데이터소스 정보를 입력할 수 있습니다.



Name	Description
Name	데이터소스 이름
Description	설명
Inquery URL	연결하고자 하는 Inquery URL
Charset	인코딩 (default : UTF-8)
Timeout(msec)	장애시 대기시간 (1000 이면 1초)
Parameters	서비스 연결시 필요한 파라미터
Auth Type	Auth Type 선택 (None or User Key / Basic Authentication / User Define Login Page)

8.

X-UP Automation 모델 개발하기

이 장에서는 GUI 에디터를 이용하여 하나의 서비스에 해당하는 로직을 다이어그램으로 Graphicer 하게 표현하는 Automation 모델 개발 방법에 대해 설명하고, 실제로 데이터를 수집 및 가공하는 여러 서비스를 이용하여 개발합니다.

X-UP 모델 개발 단계는 다음과 같습니다.

1. X-UP Builder를 이용하여 모델을 개발한다.
2. X-UP Builder에서 개발된 모델을 테스트한다.
3. 개발된 모델을 X-UP Builder의 deploy 기능을 이용하여 X-UP 서버에 배치시킨다.
4. X-UP 테스트 웹 페이지에서 배치된 모델을 테스트한다.

Automation 모델에서 사용하는 Invoke를 통해 데이터 소스로부터 데이터를 획득합니다.

각각의 Invoke는 모듈 단위 처리가 가능하며 복수개의 Invoke를 통하여 모델을 구성할 수 있습니다.

이 장의 구성은 다음과 같습니다.

- SAP RFC Invoke를 이용한 모델 개발하기
- Select Invoke를 이용한 모델 개발하기
- Modify Invoke를 이용한 모델 개발하기
- Procedure Invoke를 이용한 모델 개발하기
- OpenApi Invoke를 이용한 모델 개발하기
- WebService Invoke를 이용한 모델 개발하기
- X-UP Invoke를 이용한 모델 개발하기
- DataSet Row Loop Function을 이용하여 데이터 가공하기
- Merge Function을 이용하여 데이터 가공하기
- XML Parser Function을 이용하여 데이터 가공하기
- UDDI를 이용하여 WebService 데이터 소스 생성하기

8.1 SAP RFC Invoke를 이용한 모델 개발하기

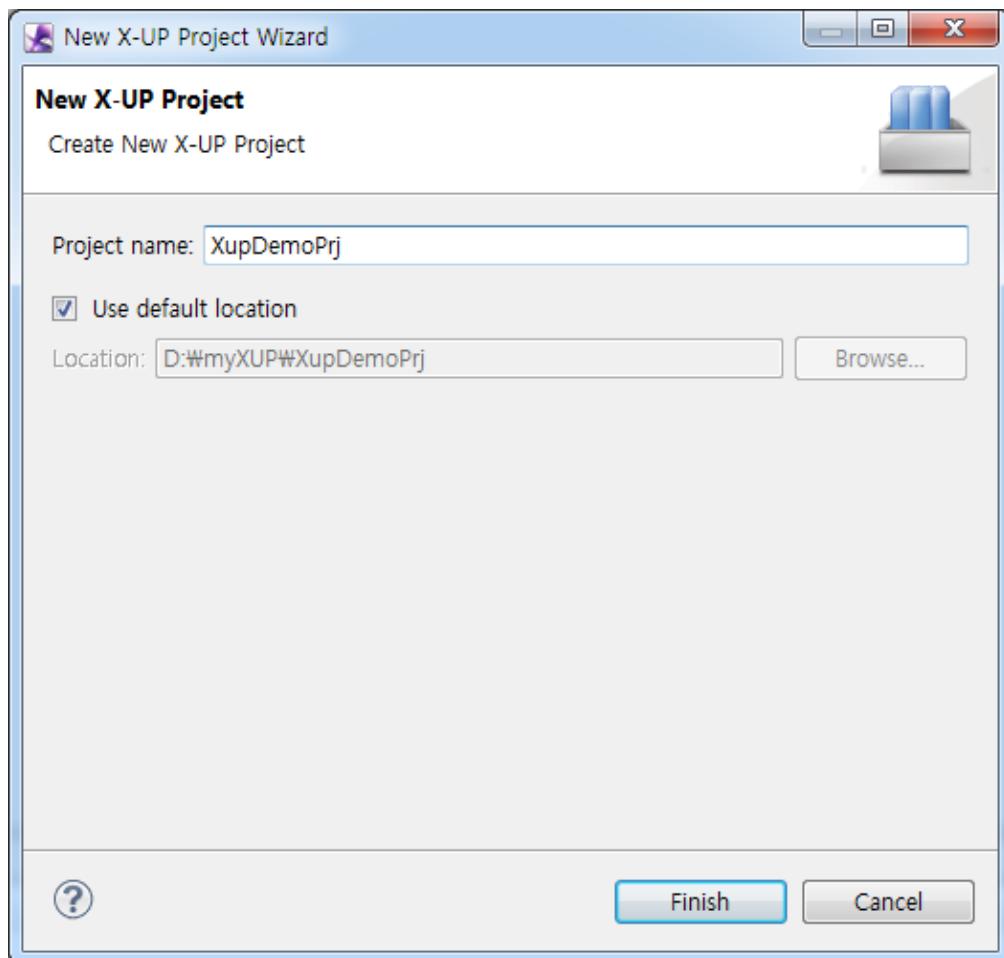
이 절에서는 SAP RFC Function을 이용한 모델 개발방법을 설명합니다.

이 절에서 설명하는 Automation 모델의 SAP RFC Invoke 개발단계는 다음과 같습니다.

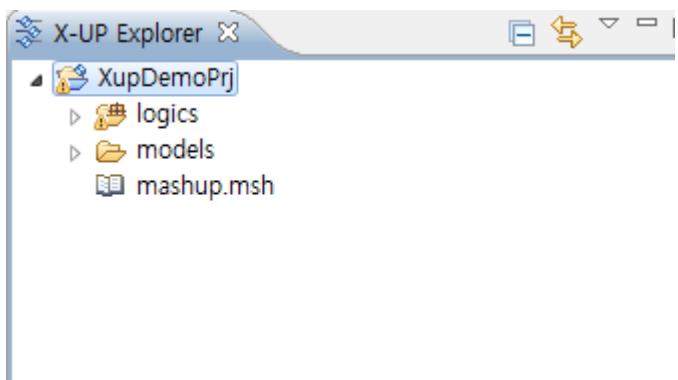
- X-UP 프로젝트 생성하기
- Automation 모델 생성 및 SAP RFC Invoke 생성 하기
- 데이터 소스 생성 및 Properties 설정 하기
- SAP RFC Invoke 테스트 하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭 합니다.

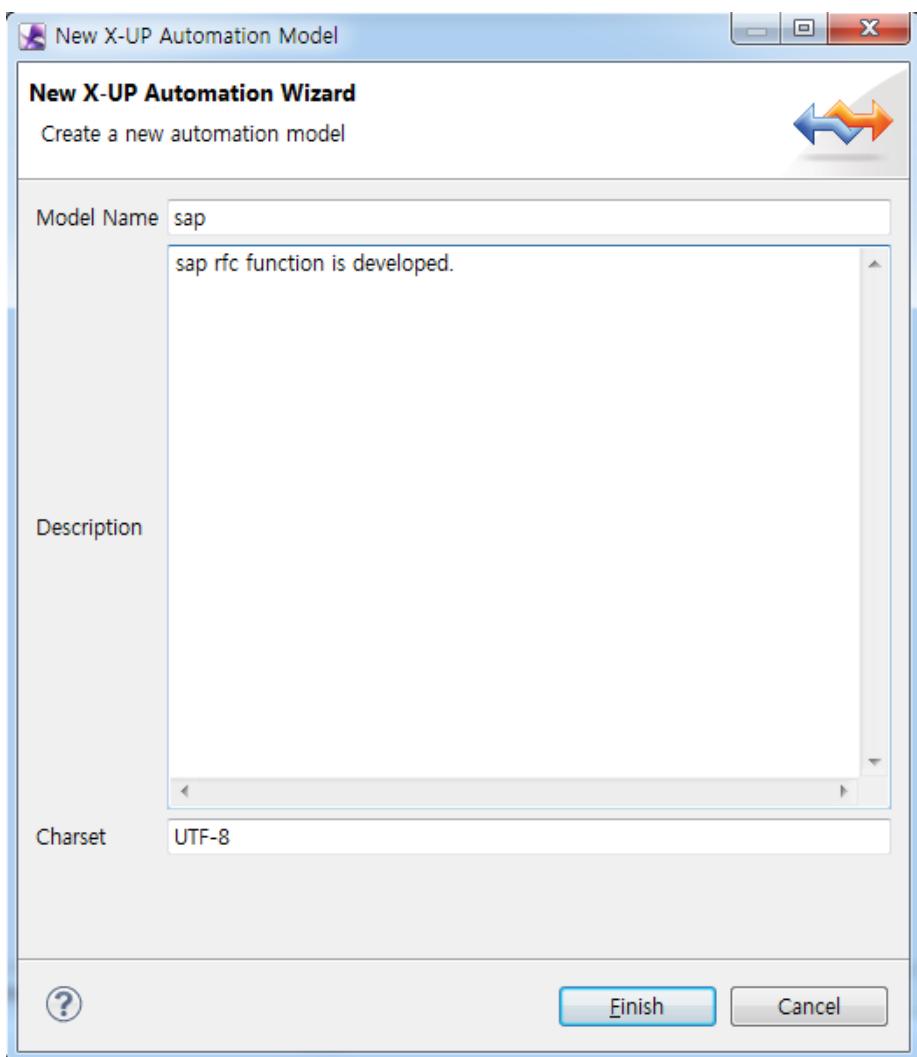


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

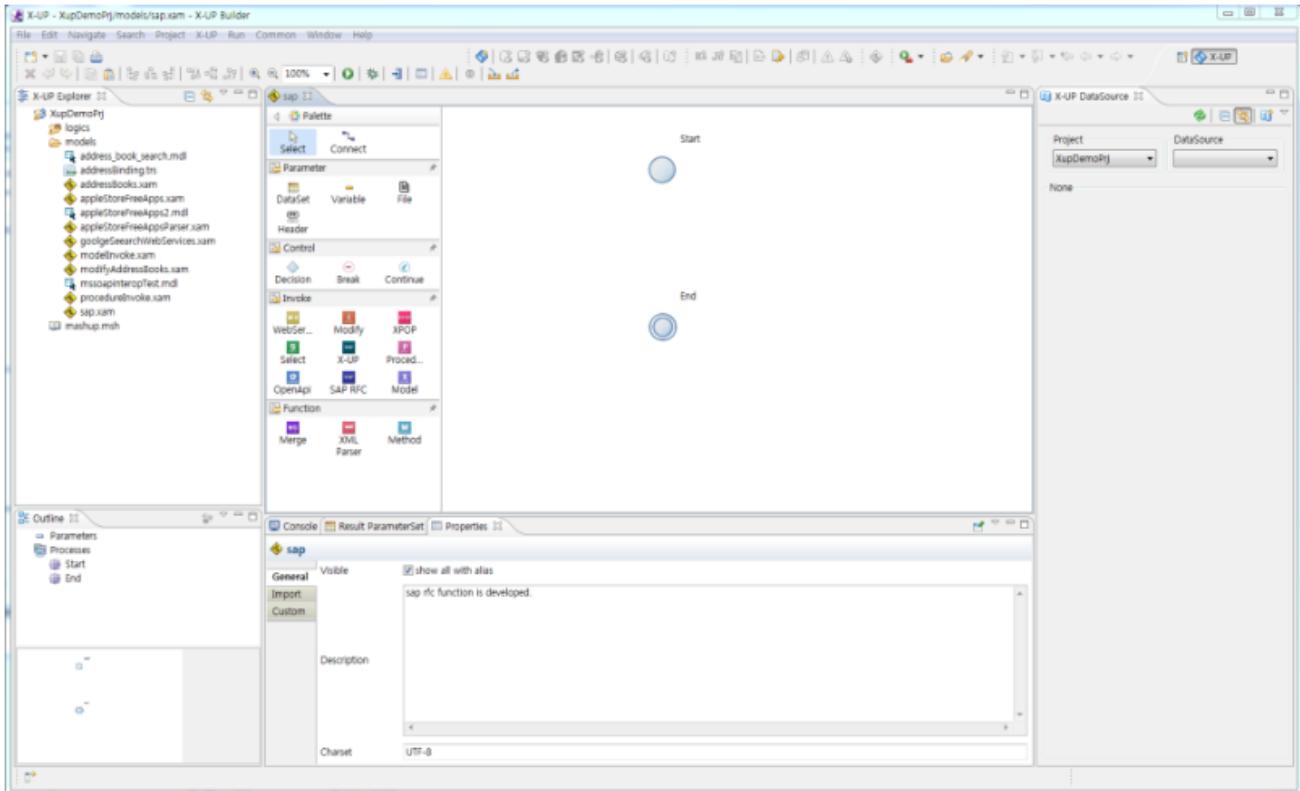


Automation 모델 생성 및 SAP RFC Invoke 생성하기

- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 원하는 모델 이름을 입력하고 OK 버튼을 클릭합니다.



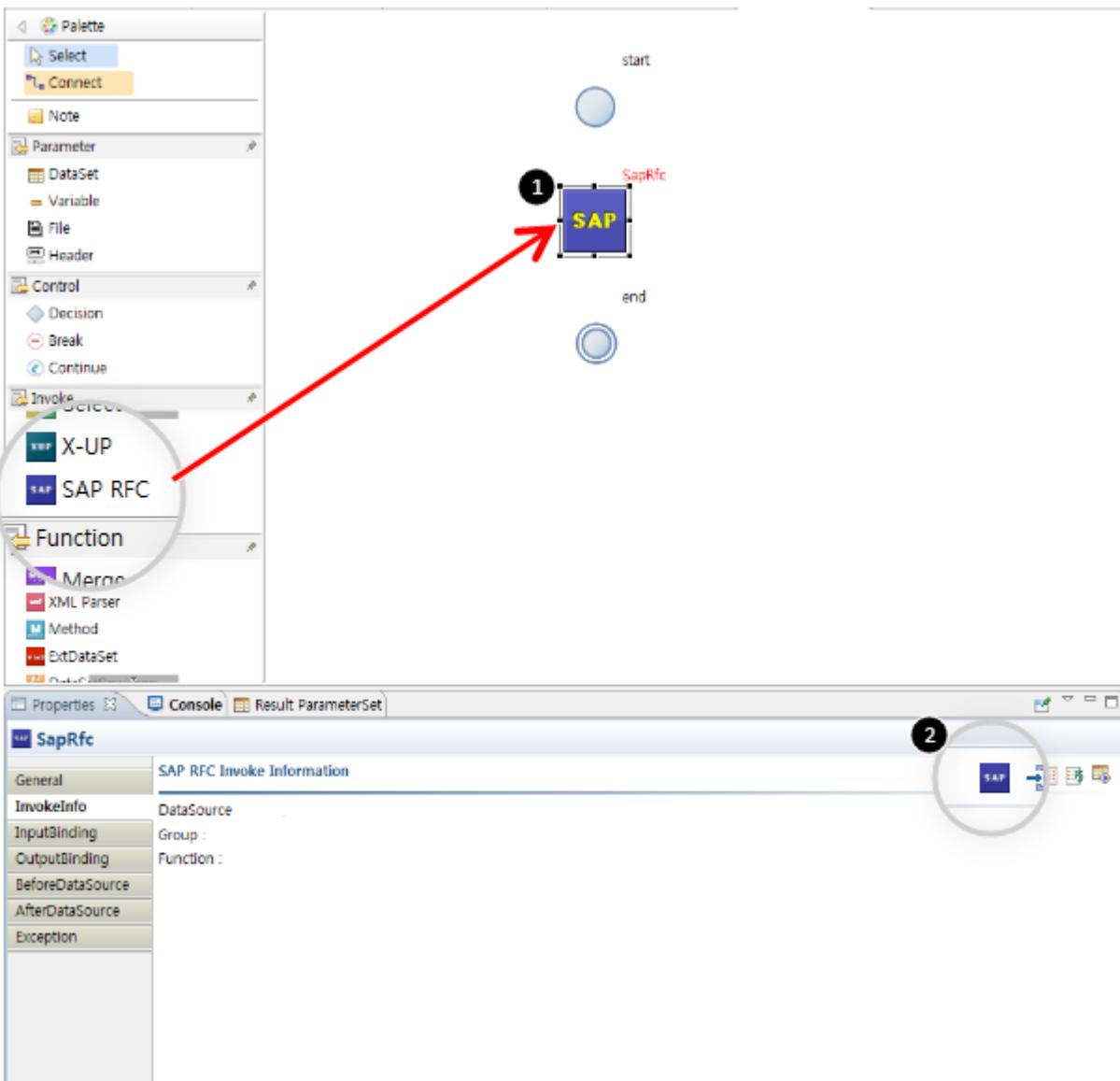
3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



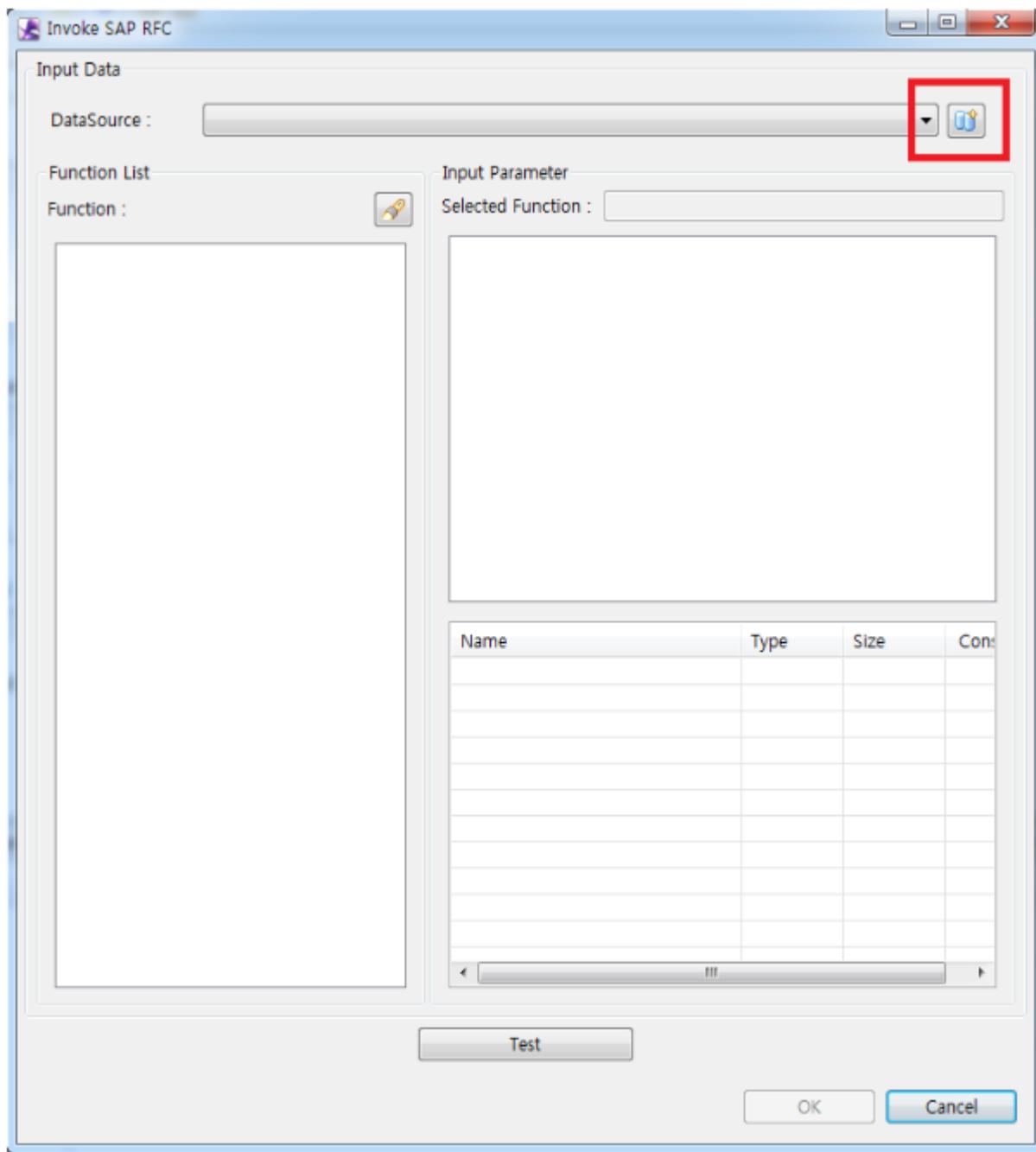
4. Palette 의 Invoke 중 'SAP RFC' 를 선택하고 에디터를 클릭 하게 되면 SAP RFC Invoke가 생성 됩니다.

데이터소스 생성 및 Properties 설정하기

1. 생성 된 SAP RFC Invoke를(❶) 더블클릭하거나 'Properties' 탭을 클릭하여 'SAP RFC icon'[❷] (❸) 을 클릭하면 위자드가 실행 됩니다.



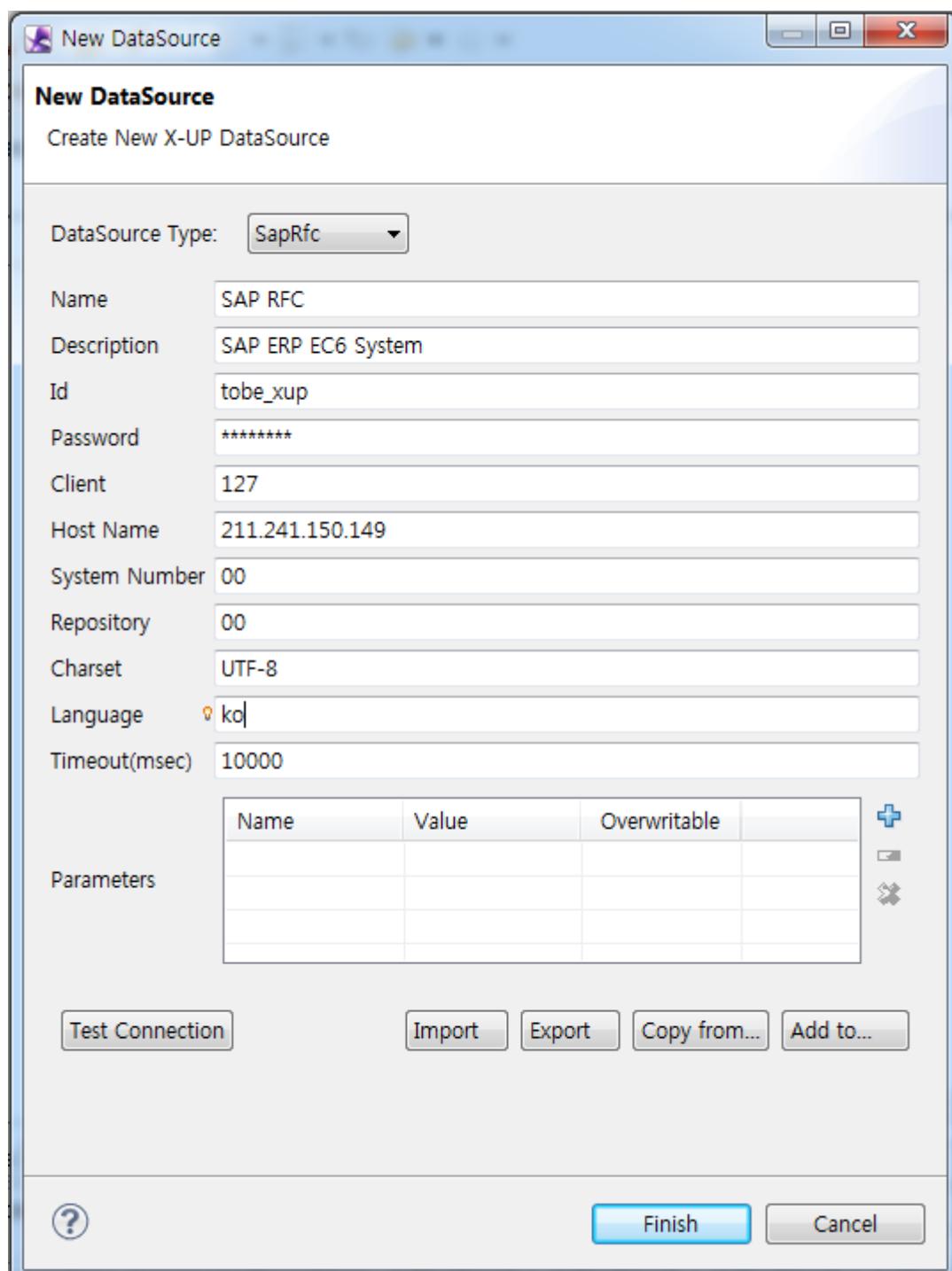
2. Invoke SAP RFC 위자드에서 **create new DataSource** [] 를 클릭합니다.



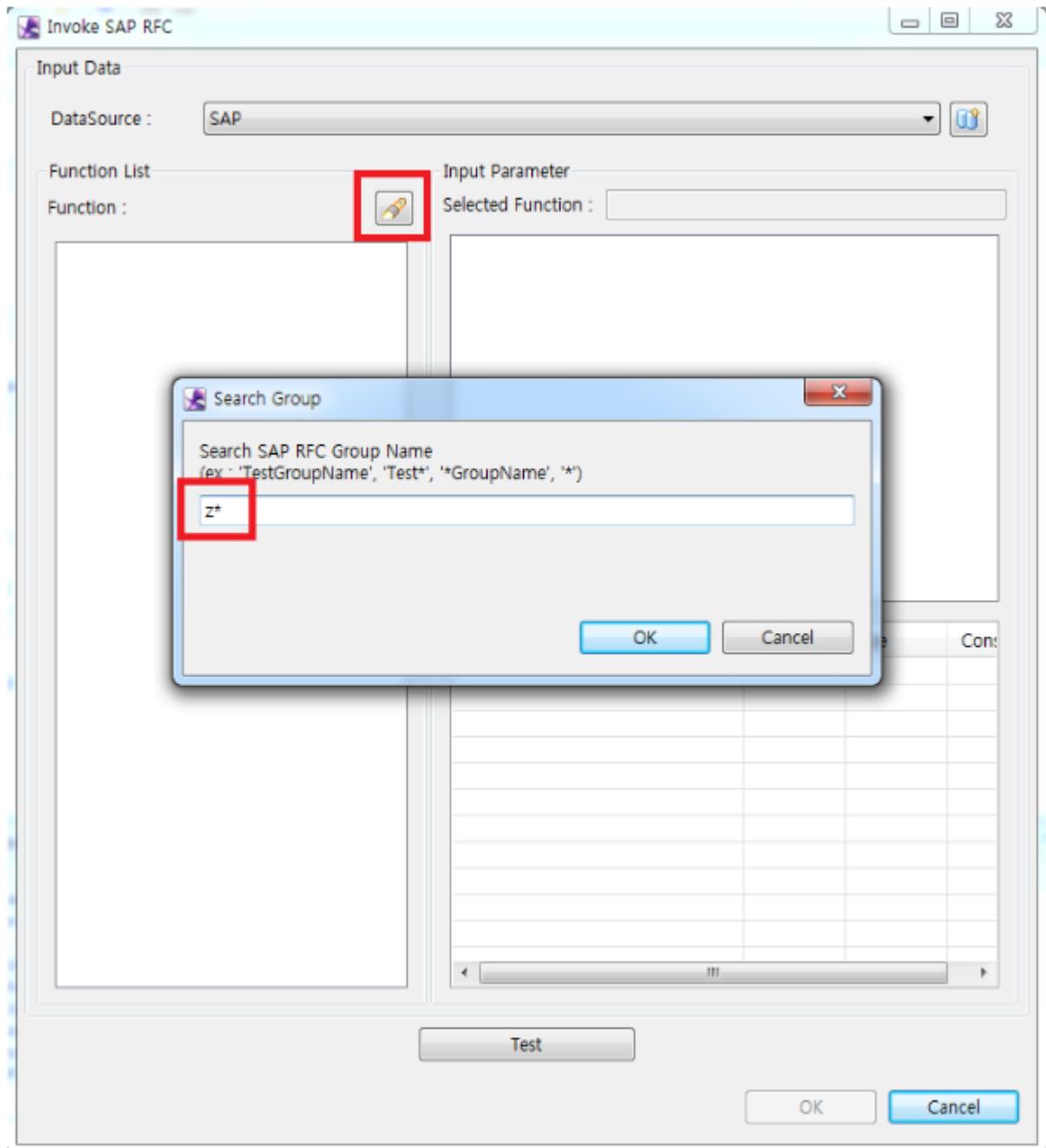
3. **New DataSource** 페이지에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
 4. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

Field Name	Field Value
Name	'SAP RFC' 데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
ID	SAP RFC 서버 접속 ID.

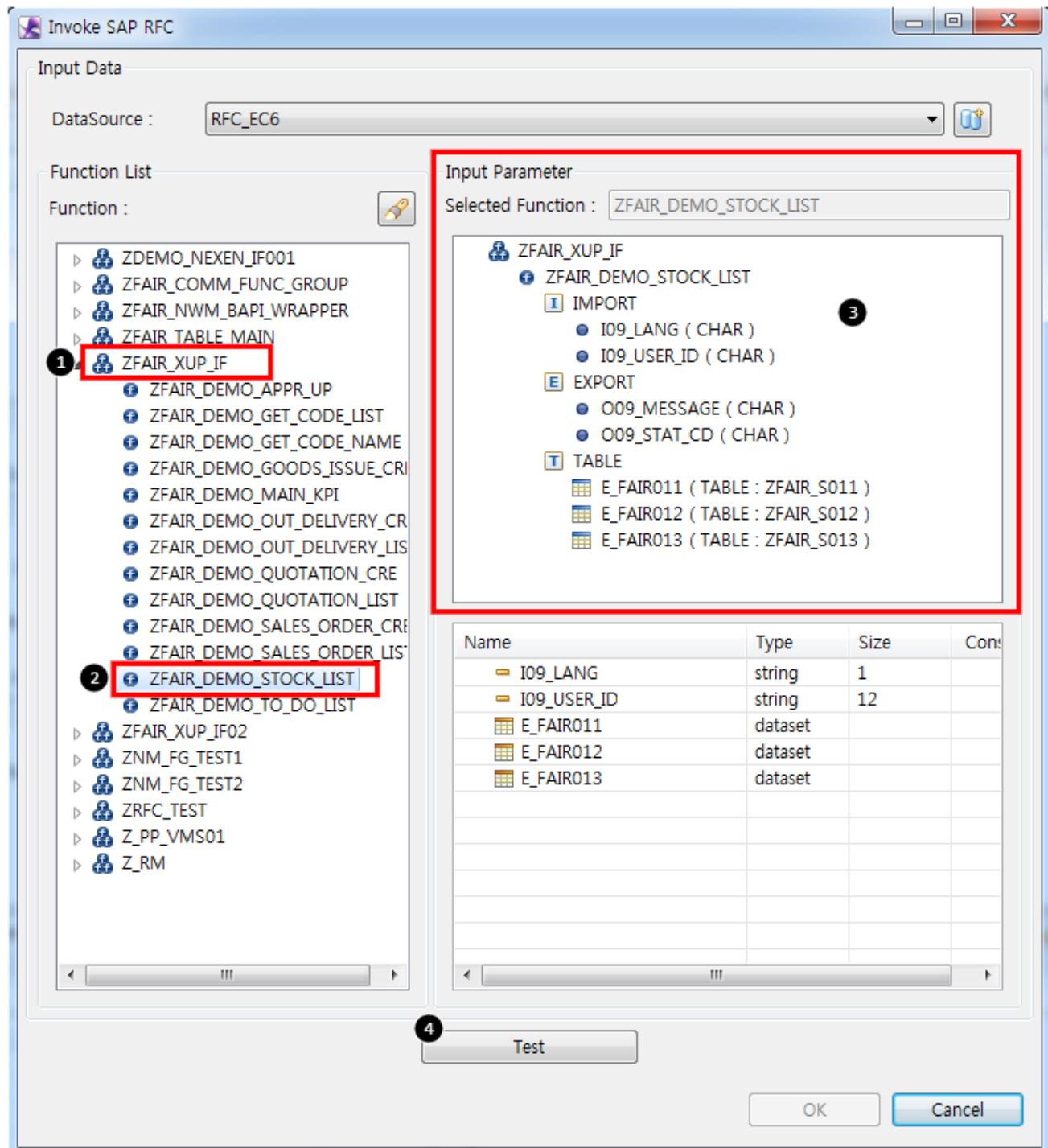
Field Name	Field Value
PASSWORD	SAP RFC 서버 접속 PASSWORD.
Client	로그인 계정에서 사용할 client 정보
Host Name	서버의 IP
System Number	로그인 계정에서 사용 할 System Number 정보
Repository	로그인 계정에서 사용할 Repository 정보
Charset	사용할 charset
Timeout	서버와의 접속을 유지할 최대 시간

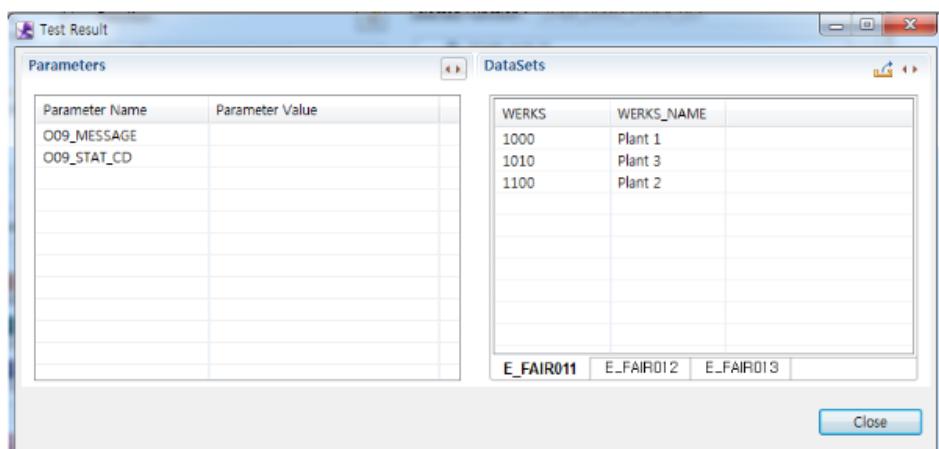
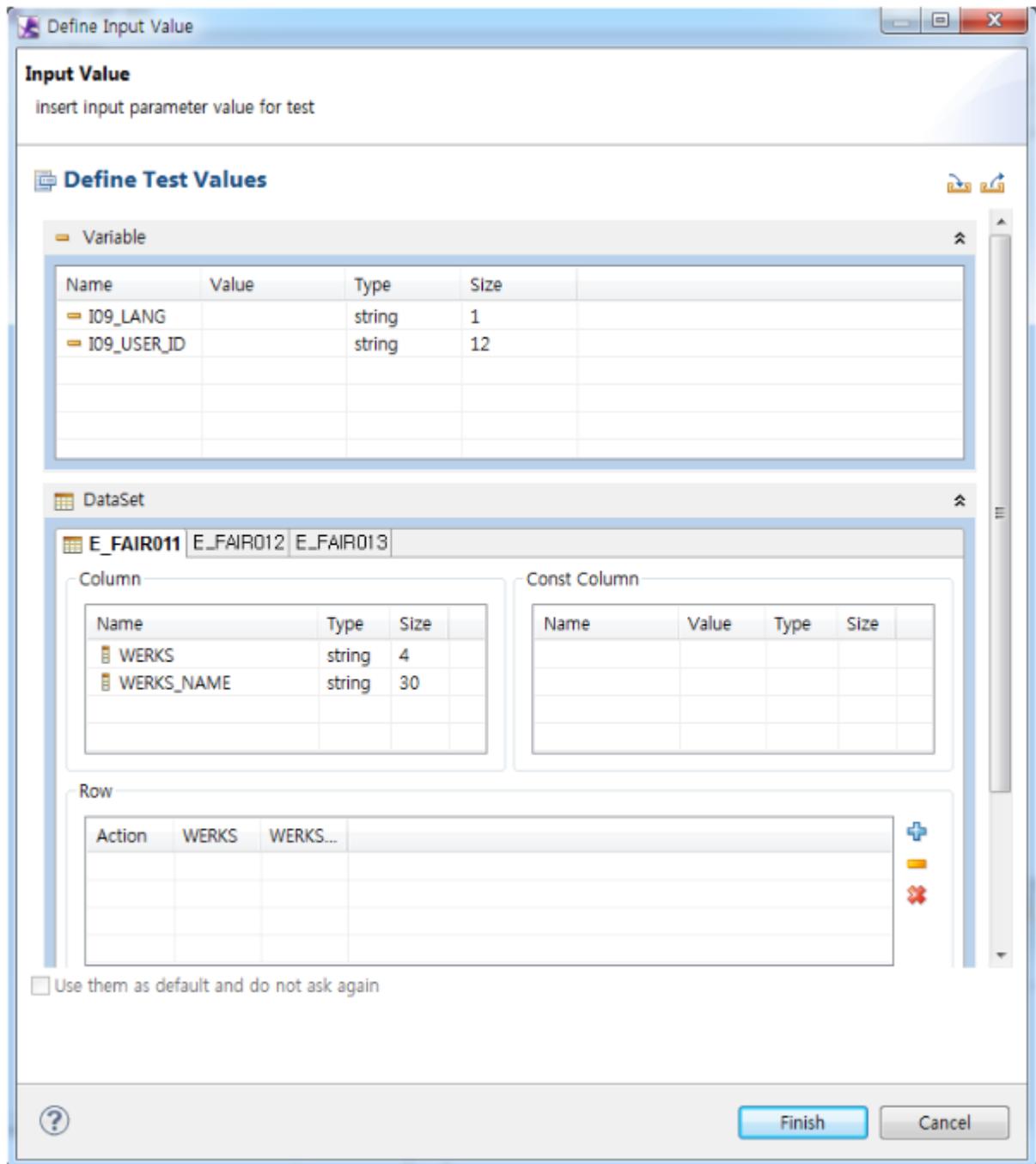


5. DataSource를 SAP RFC을 선택하고 get function list 버튼 [] 을 클릭하여 서버에 등록된 function 리스트를 호출합니다.
6. 그룹(function) 검색 시 * 검색이 가능합니다.

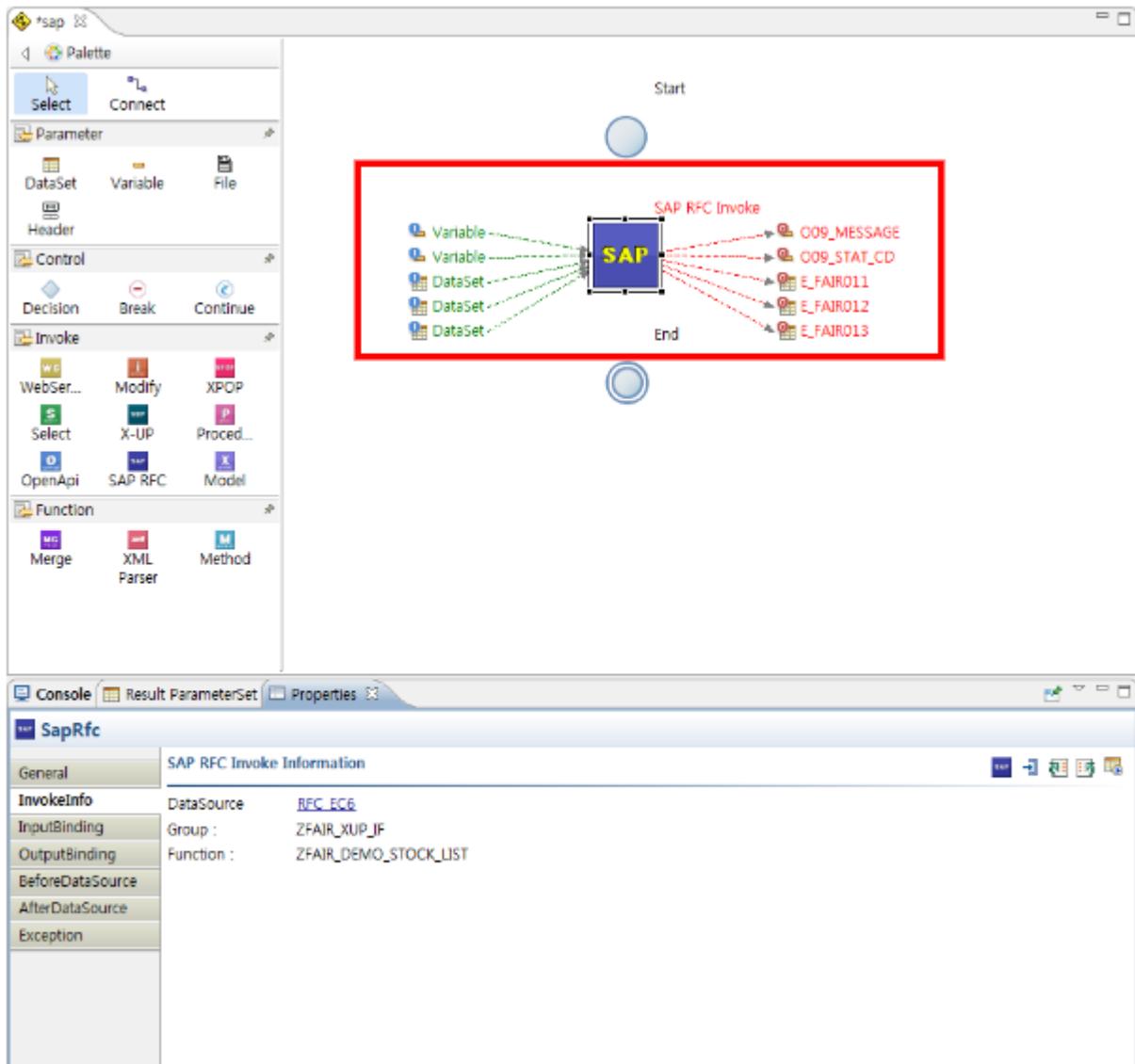


7. 검색 된 function 리스트 중 호출 하고자 하는 function 을 선택(❶) 하면 해당 function의 입력 파라메터 와 출력 파라메터의 정보를 확인(❷) 할 수 있습니다.
8. 동시에 SAP RFC 파라메터 형태를 X-UP Parameter 형태(❸)로 나타납니다.
9. Test 버튼(❹)을 클릭하여 생성 된 Define Input Value 위자드의 입력 파라메터의 값을 설정 하고 'Finish' 버튼을 클릭하여 정상적으로 데이터가 표시되는지 확인 합니다.



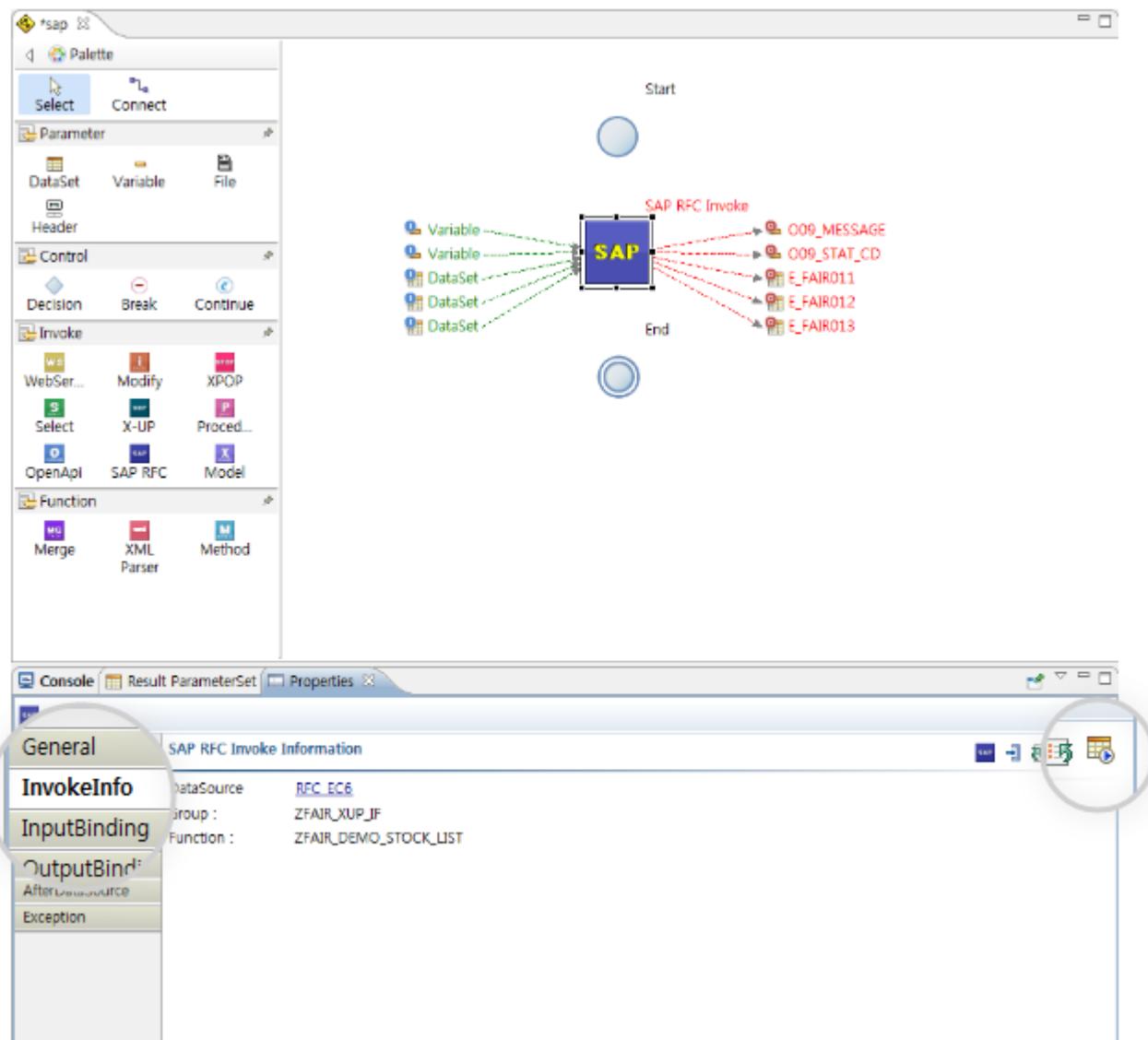


10. OK 버튼을 클릭 할 경우 입력 파라메터와 출력 파라메터가 생성이 된 것을 확인 할 수 있습니다.



Invoke 테스트 하기

1. 생성 된 SAP RFC Invoke입니다. Invoke의 테스트 결과를 바탕으로 입력 파라메터와 출력 파라메터가 생성됩니다.
2. SAP RFC Invoke의 Properties의 'InvokeInfo'를 선택하고 [] 를 클릭하여 테스트를 진행합니다.



3. Define Input Value 위자드가 생성 되면 Properties 설정 시 입력 한 테스트 값을 입력 하고 “Finish” 버튼을 클릭합니다.
4. 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인 할 수 있습니다.

The screenshot shows the Result ParameterSet view. It has two main sections: Parameters and DataSets.

Parameters table:

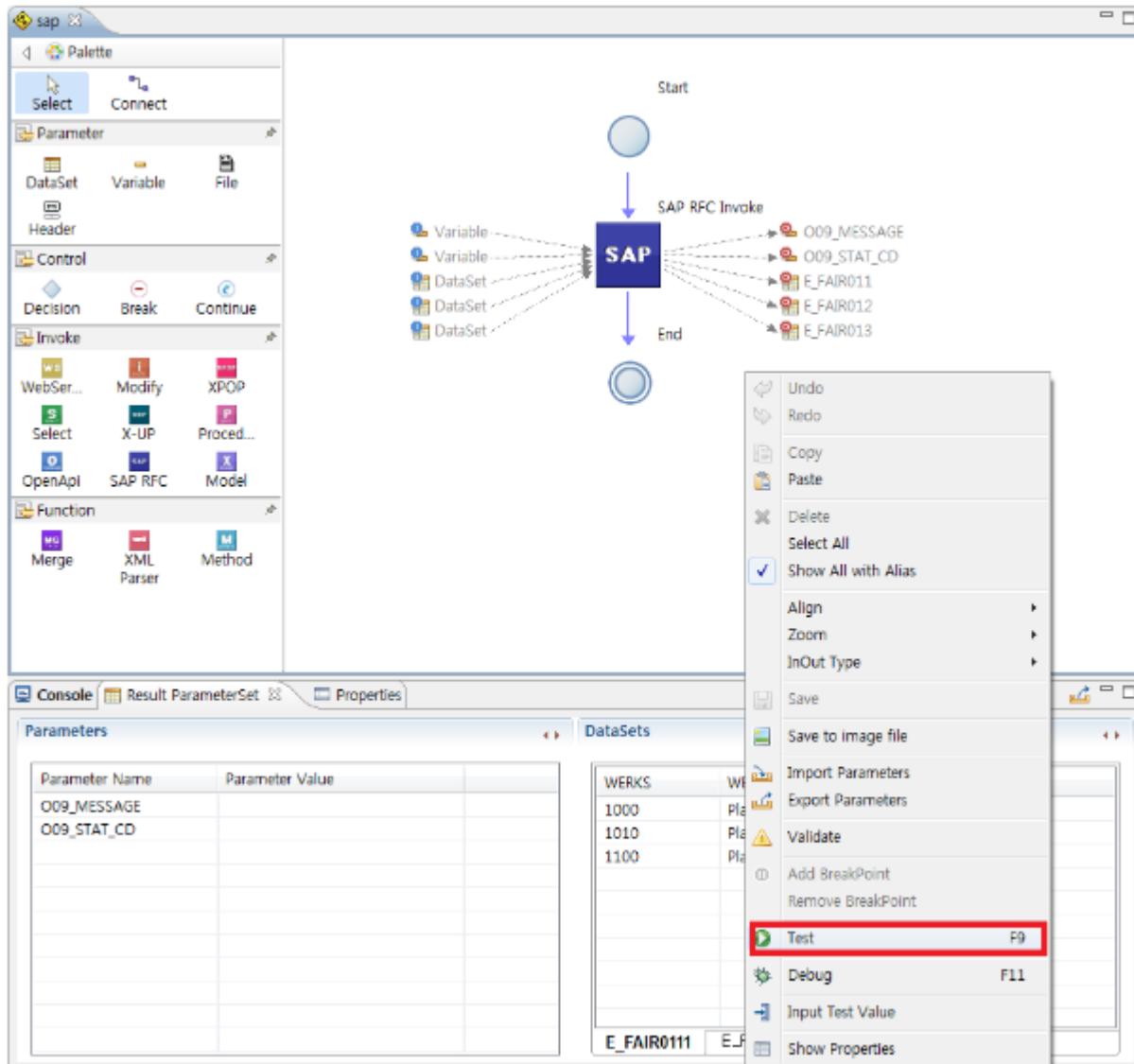
Parameter Name	Parameter Value
O09_MESSAGE	
O09_STAT_CD	

DataSets table:

WERKS	WERKS_NAME
1000	Plant 1
1010	Plant 3
1100	Plant 2

모델 테스트하기

- 모델 에디터에서 Palette의 Connect를 선택하여 Start 노드와 SAP RFC Invoke, end 노드를 연결 해 줍니다.
- 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.
- 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.



8.2 Select Invoke를 이용한 모델 개발하기

이 절에서는 Automation 모델을 이용하여 데이터베이스로부터 Select 쿼리로 데이터를 획득하는 개발방법에 대하여 설명합니다.



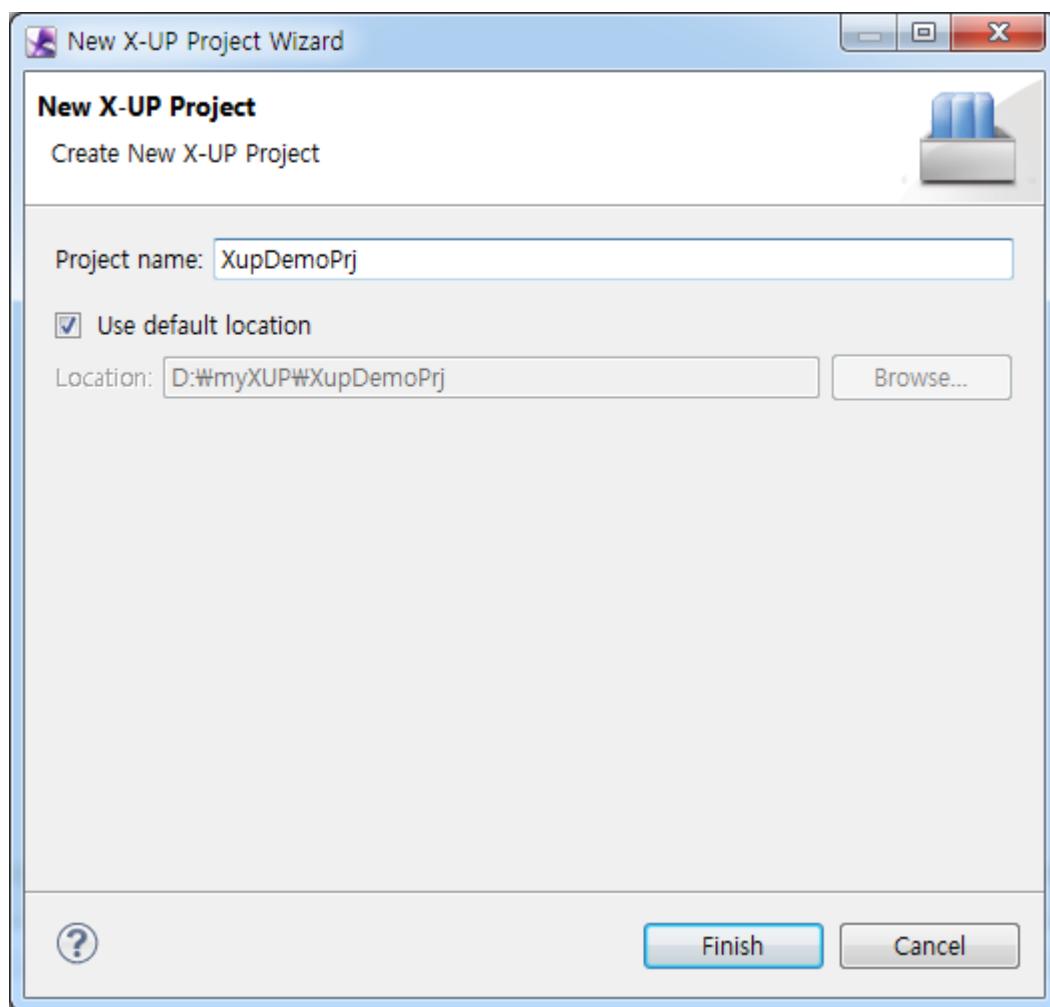
이 절에서 사용한 데이터베이스 테이블 정보는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 참조하십시오.

이 절에서 설명하는 Automation 모델의 Select Invoke 개발단계는 다음과 같습니다.

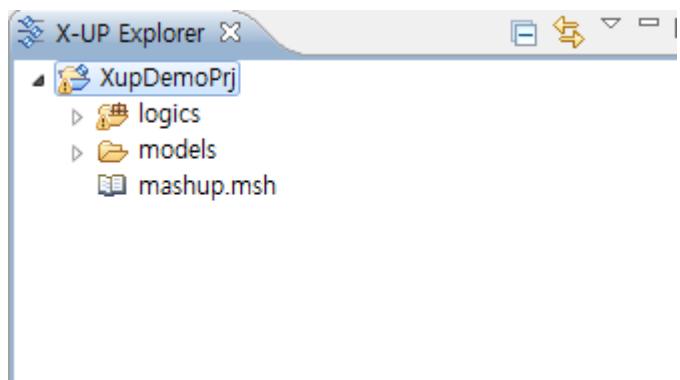
- X-UP 프로젝트 생성하기
- Automation 모델을 생성 및 Select Invoke 생성하기
- 입력 파라메터 설정 및 Properties 설정하기(SQL 변경)
- Invoke 테스트하기
- Function 을 이용한 추가 작업하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 Project name 필드에 프로젝트 이름을 입력하고 'Finish' 버튼을 클릭합니다.

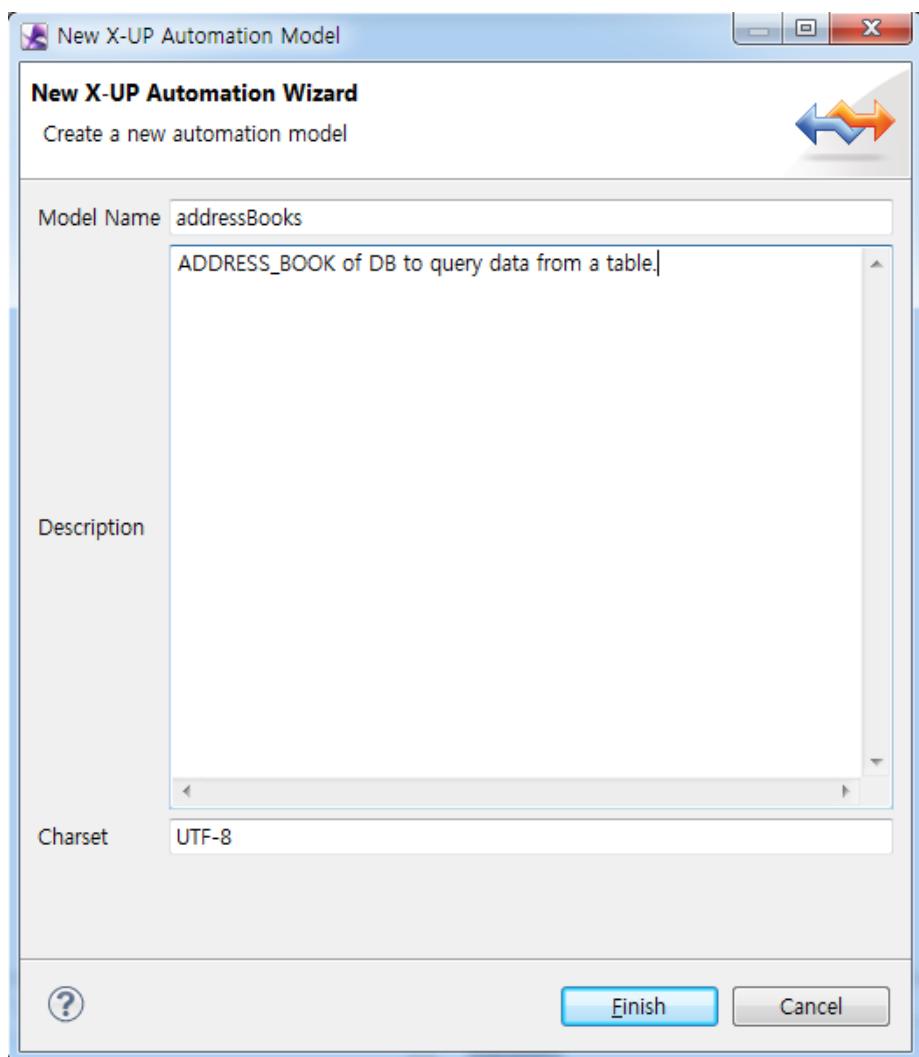


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

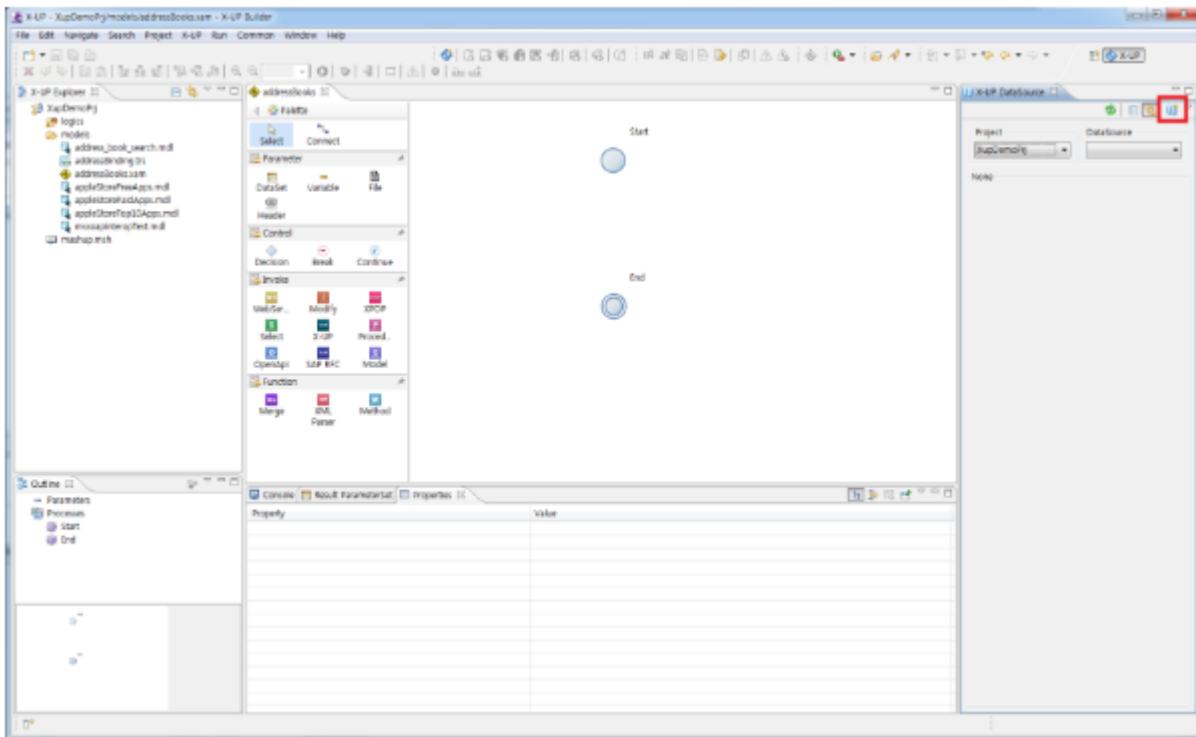


Automation 모델 생성 및 Select Invoke 생성하기

- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 OK 버튼을 클릭합니다.



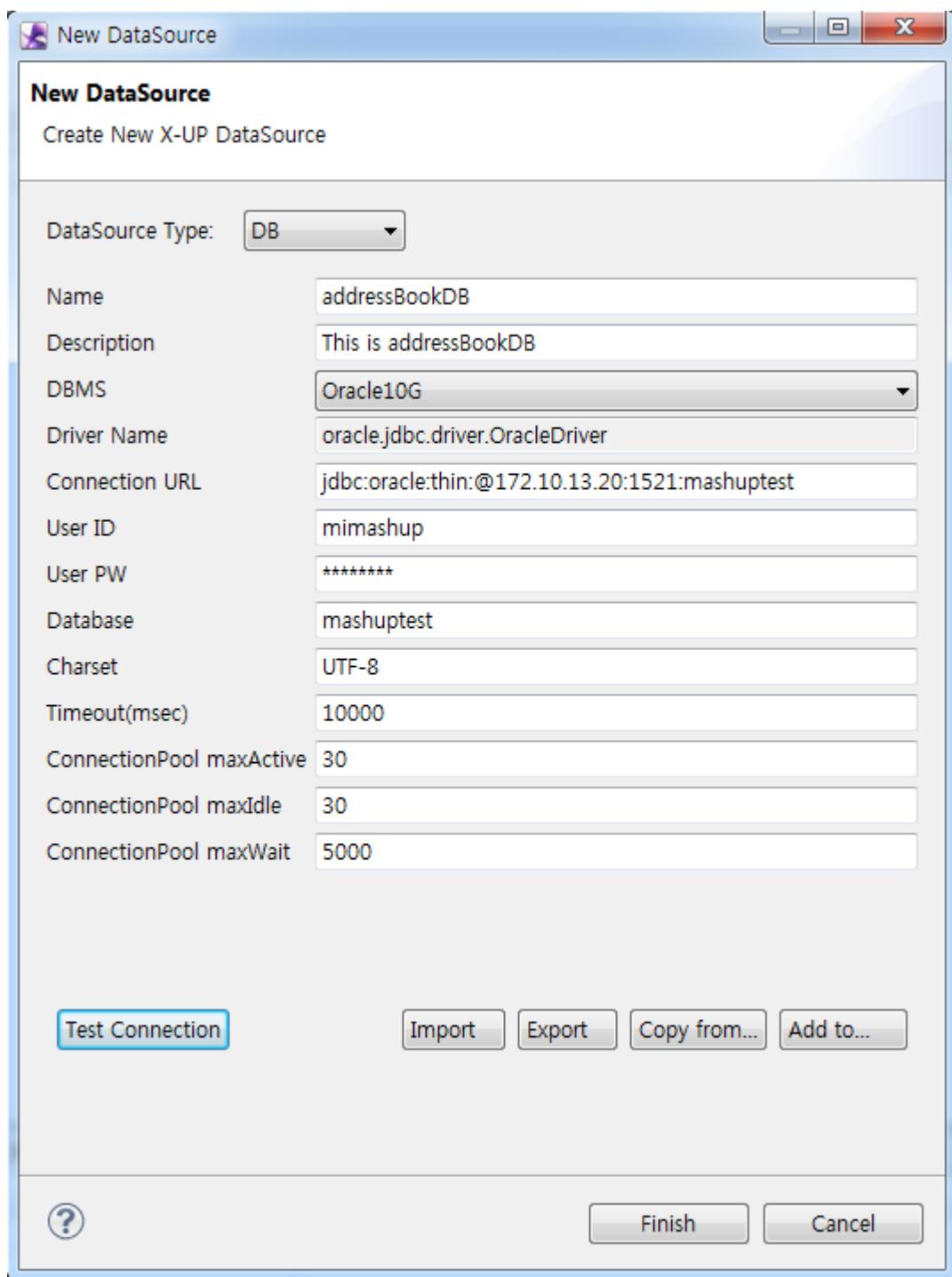
3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



4. 화면 우측의 X-UP DataSource 위자드에서 new DataSource를 선택해 Wizard를 실행합니다.
5. New DataSource 위자드에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
6. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

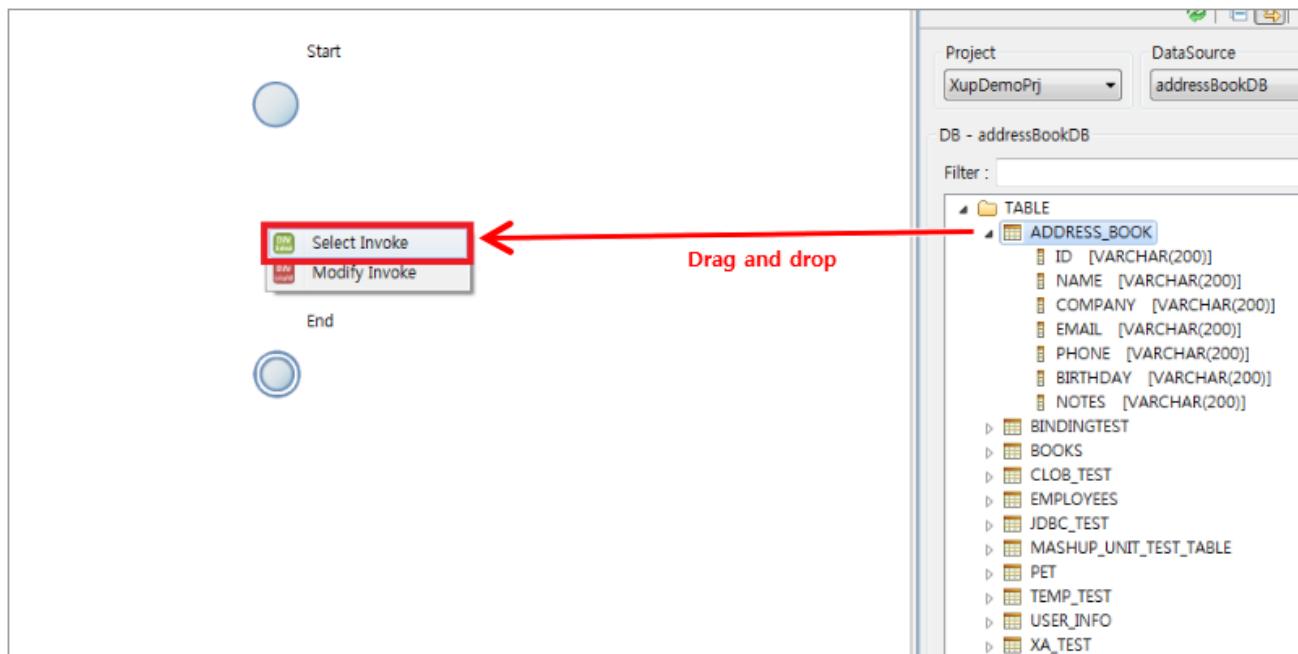
Field Name	Field Value
Name	'addressBookDB' 데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
DBMS	사용할 데이터베이스 시스템을 선택합니다. 이 예제는 'Oracle'을 사용합니다.
Connection URL	JDBC Connection String을 입력합니다.
User ID	데이터베이스 사용자 ID를 입력합니다.
User PW	데이터베이스 사용자 암호를 입력합니다.
Database	사용할 데이터베이스의 이름을 입력합니다.
Charset	'UTF-8' 문자 인코딩에 대한 값을 입력합니다.
Timeout(msec)	'10000' 서버와의 접속을 유지할 최대 시간
ConnectionPool MaxActive	'30' 사용하는 커넥션 최대수
ConnectionPool MaxIdle	'30' 사용하지 않는 커넥션의 최대수

Field Name	Field Value
ConnectionPool maxWait	'5000' 커넥션을 열기 위해 최대 기다리는 시간

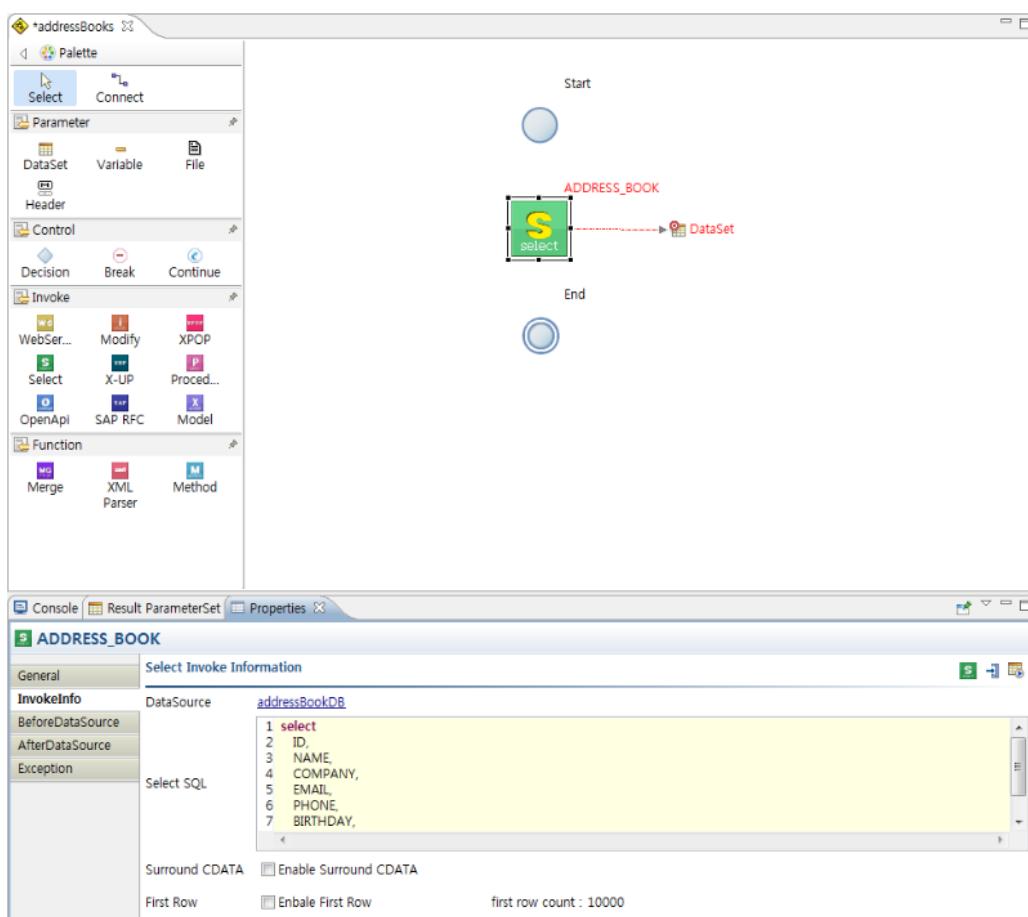


7. X-UP DataSource 뷰에서 DataSource []를 선택 해 테이블 목록을 가져옵니다.
8. 가져온 테이블 목록에서 'ADDRESS_BOOK' 테이블을 선택하여 에디터로 드래그 앤 드롭합니다.
9. Select Invoke와 Modify Invoke 중 Select Invoke를 하면 Select Invoke 가 생성됩니다.

10. 또는 Palette 의 Invoke 중 Select Invoke를 선택하고 에디터를 클릭 하게 되면 Select Invoke를 생성할 수 있습니다.



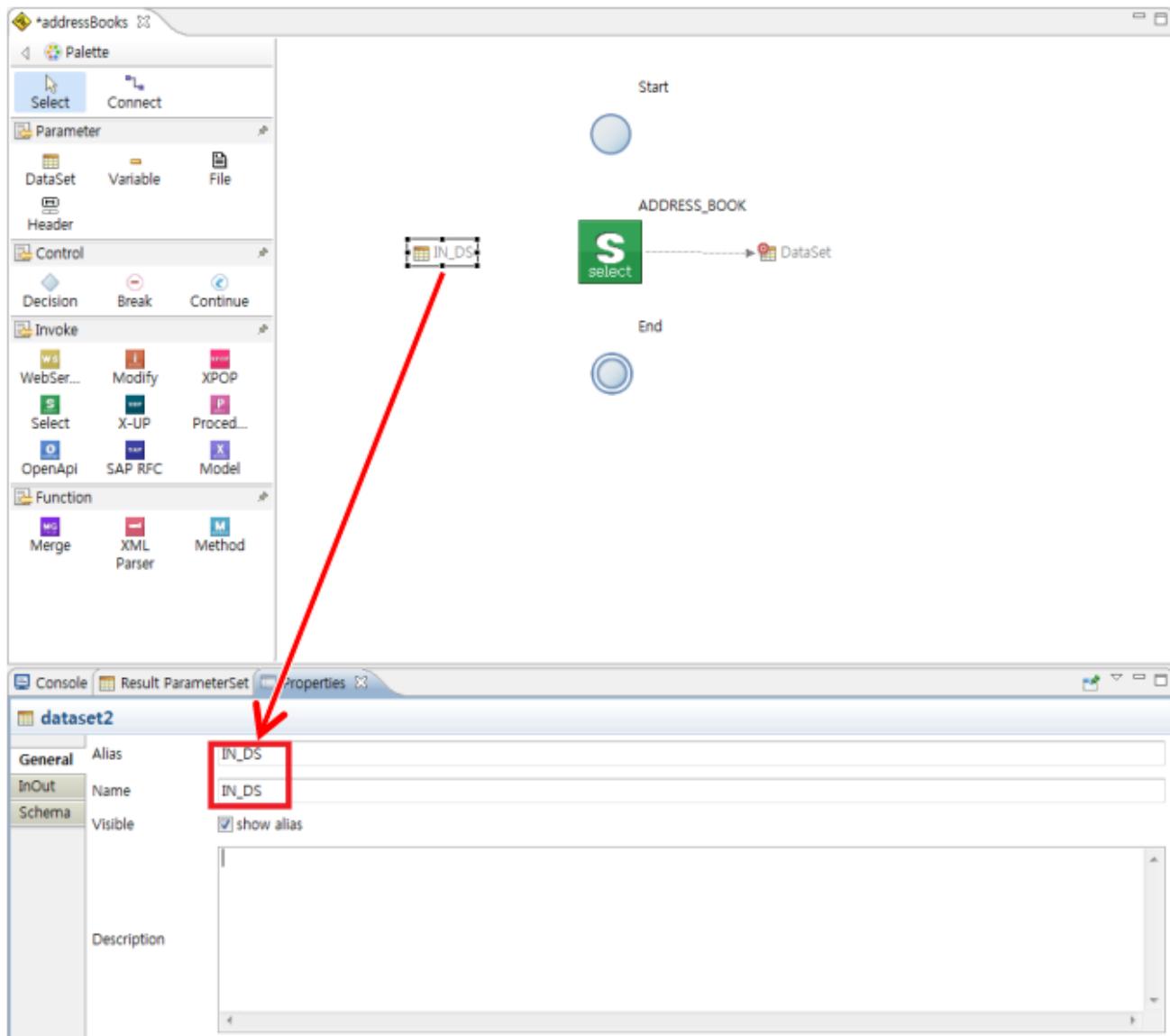
11. 생성 된 Select Invoke 기본 적으로 데이터셋을 출력 파라메터로 가지게 됩니다.
 12. 그리고 생성 된 SQL문은 모든 컬럼을 where 조건 없이 조회하는 Select sql 입니다.



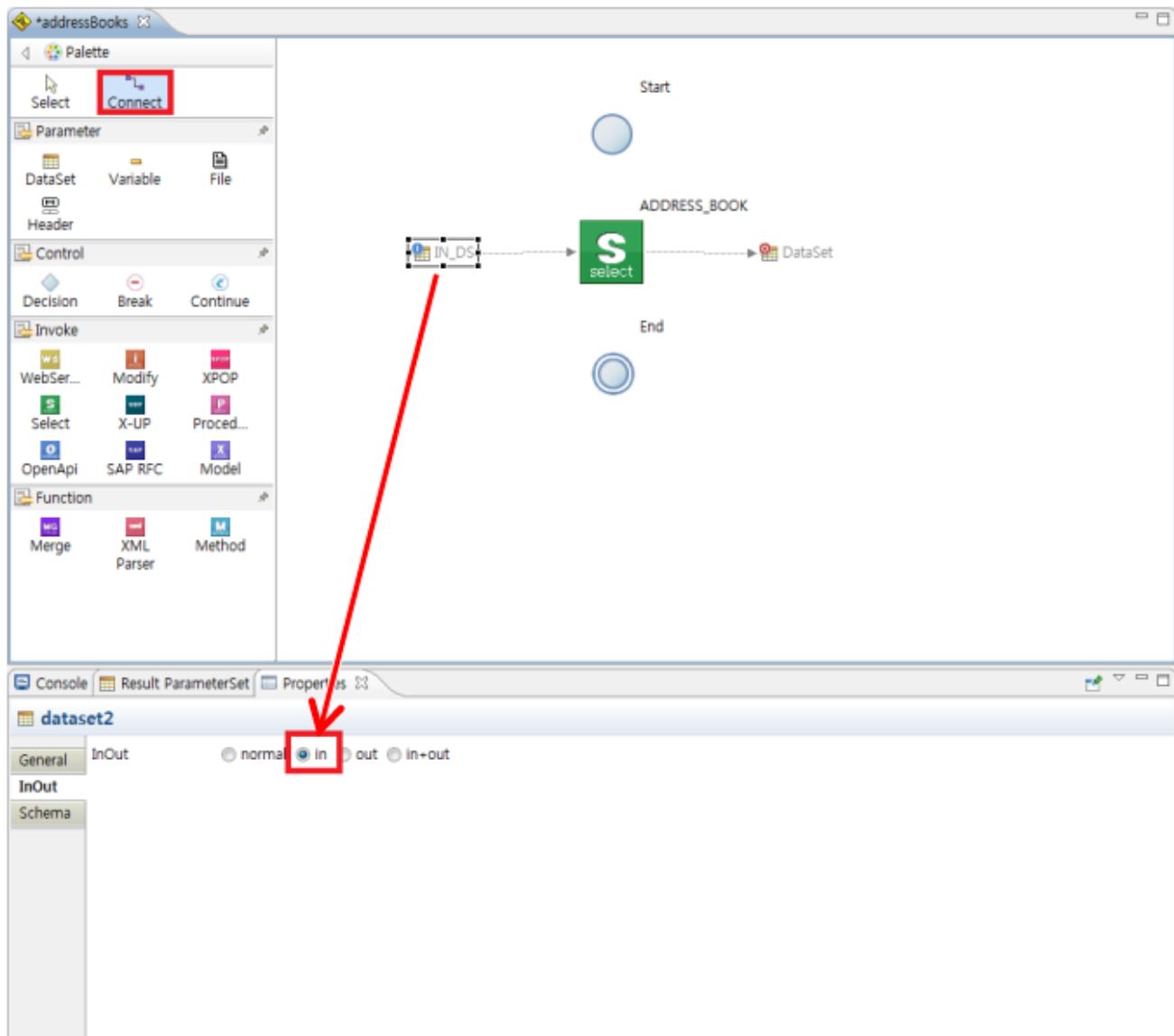
입력 파라미터 설정 및 Properties 설정하기(SQL 변경)

이번 단계에서는 입력 파라미터를 정의 하는 것과 데이터소스로부터 데이터를 획득하기 위해 필요한 정보를 입력하는 Properties를 설정 합니다.

1. Palette의 DataSet 를 선택 해 에디터를 클릭하면 데이터셋이 생성 됩니다.
2. 생성 된 데이터셋 의 기본 명칭은 'dataset + index'입니다. 이름과 Alias를 IN_DS로 변경합니다.



3. 변경 된 데이터셋의 Properties의 'InOut' 을 선택해 1번 'in' 을 선택 해야 만 X-UP의 입력 파라미터가 됩니다.
4. 그리고 데이터셋 과 Invoke를 연결 하여 select Invoke에 필요한 입력 파라미터로 설정 합니다.
5. Output 으로 나오는 데이터셋은 OUT_DS로 Alias와 이름을 바꾸고, 'out'으로 선택합니다.



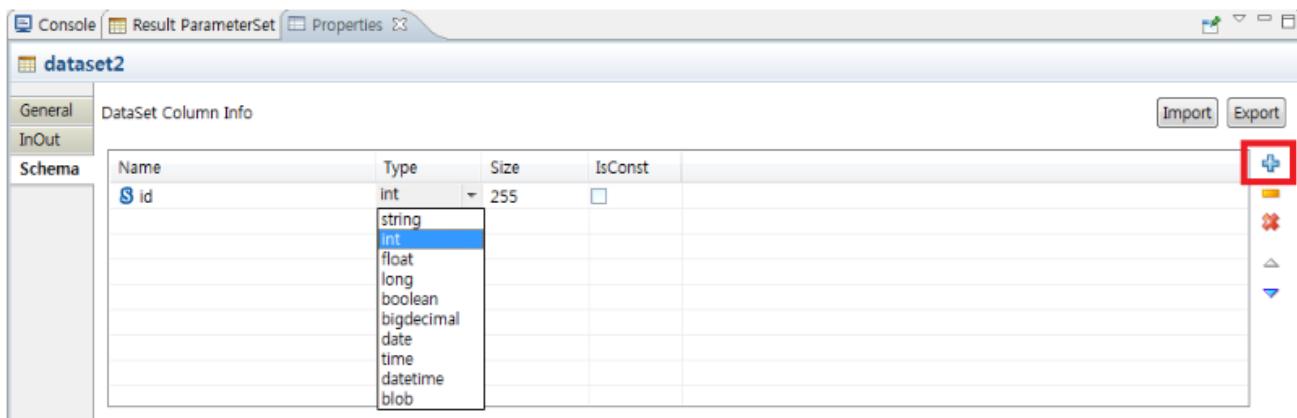
데이터셋 파라메터의 사용 시 Alias와 Name의 구분

Alias : 사용자가 원하는 별칭을 정의합니다. 한글입력도 가능합니다.

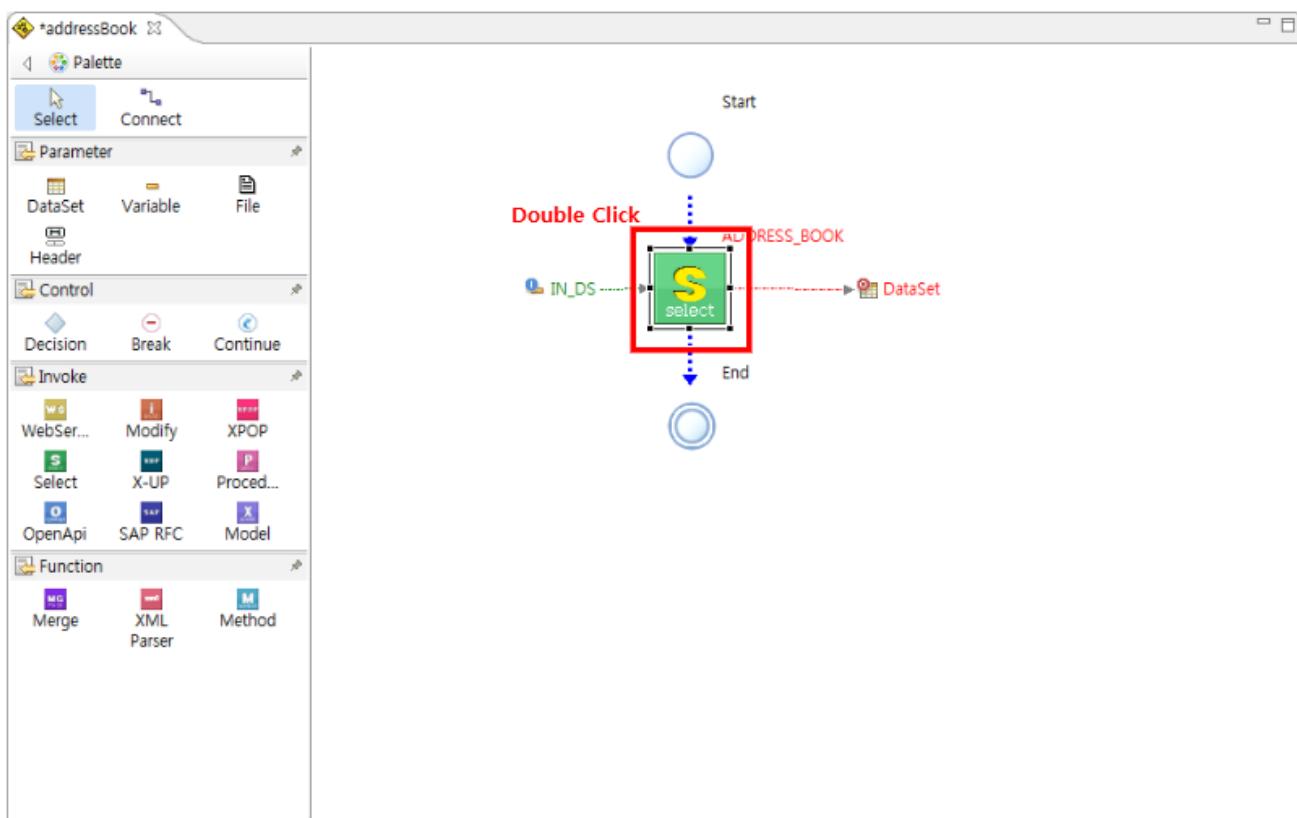
Name : 데이터셋의 이름을 정의합니다. 에디터상에서 다른 파라메터와 중복되지 않는 이름으로만 정의합니다.

개발 시 에디터 화면에서 보이는 것은 Alias이나, 자바코드 및 소스 제너레이션 시 실제로 구분되어지는 것은 Name입니다.

6. 그리고 Properties의 'Schema'를 선택하여 id를 추가하고 데이터의 Type을 지정합니다.



7. 아래 그림의 Select Invoke에서 Start, end와 Invoke를 Connect를 통해 연결을 했을 경우 모델 테스트 시 Invoke가 실행이 되게 됩니다.



8. Select Invoke를 더블클릭 하면 자동으로 생성 된 SQL문을 수정할 수 있습니다. 예제는 IN_DS의 id를 입력변수로 받아서 주소록 테이블(ADDRESS_BOOK)을 조회합니다. 조회 시 id가 Null이 아닐 경우 입력변수로 받은 ID만을 조회 합니다.

Code 8-1 수정 전

```
select
    ID,
    NAME,
    COMPANY,
```

```

EMAIL,
PHONE,
BIRTHDAY,
NOTES
from ADDRESS_BOOK

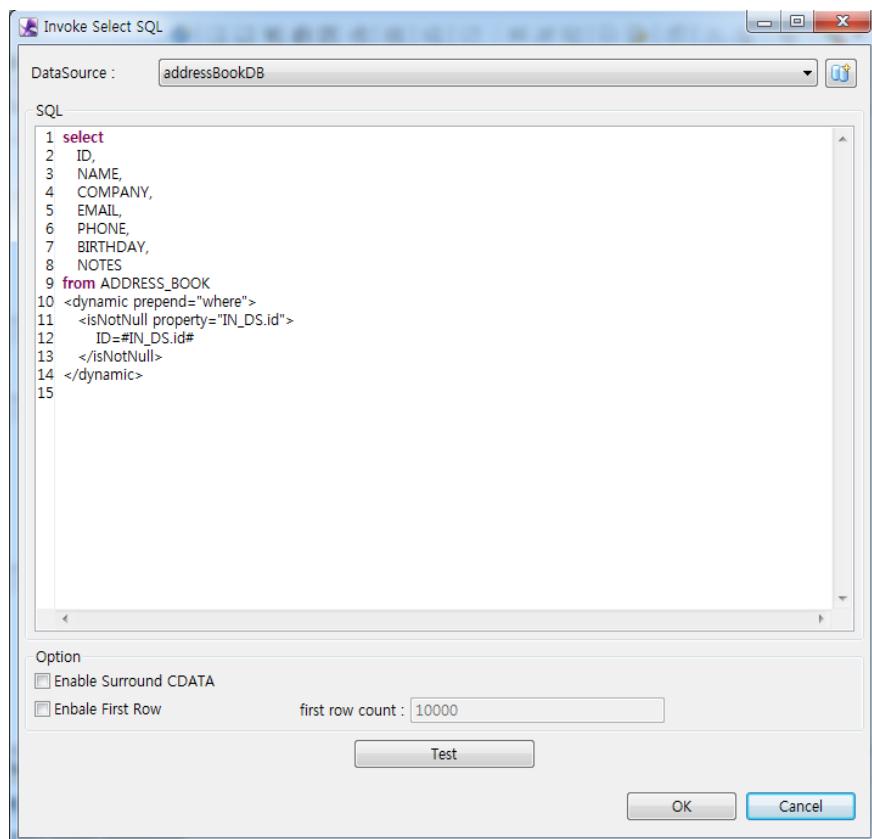
```

Code 8-2 수정 후

```

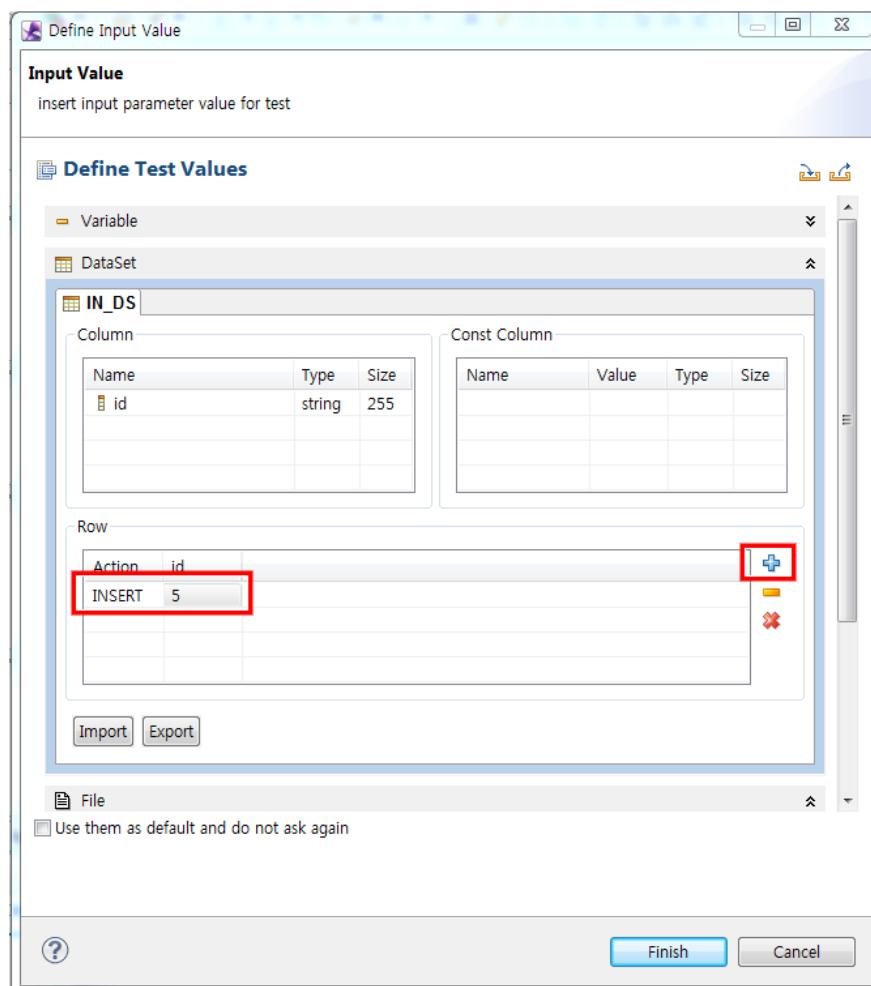
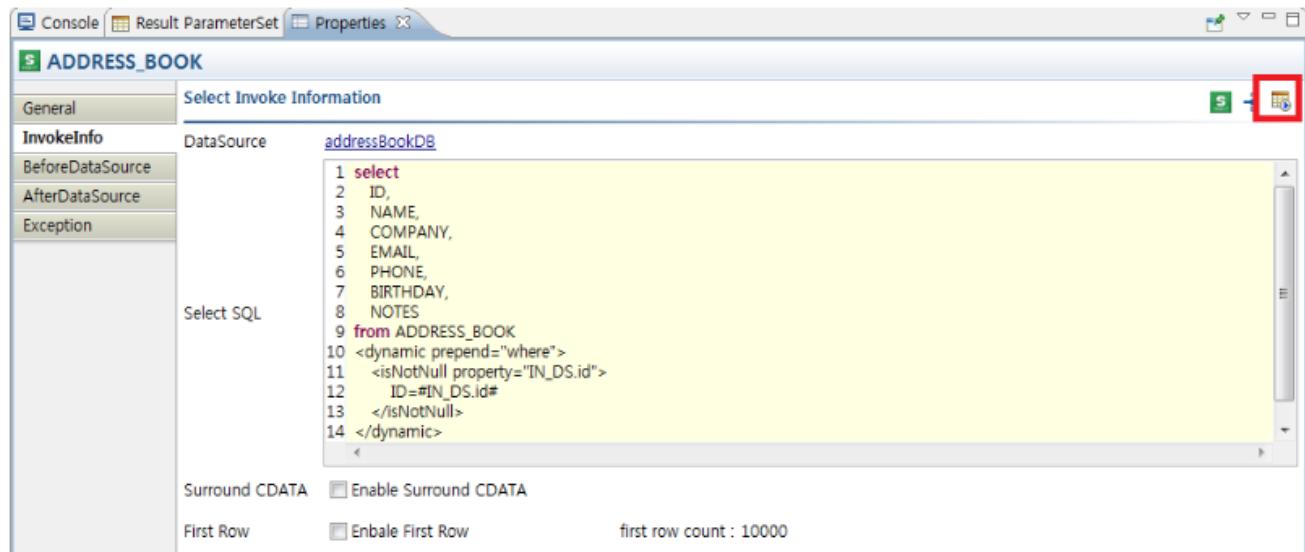
select
ID,
NAME,
COMPANY,
EMAIL,
PHONE,
BIRTHDAY,
NOTES
from ADDRESS_BOOK
<dynamic prepend="where">
<isNotNull property="N_DS.id">
    ID = # IN_DS.id #
</isNotNull>
</dynamic>

```

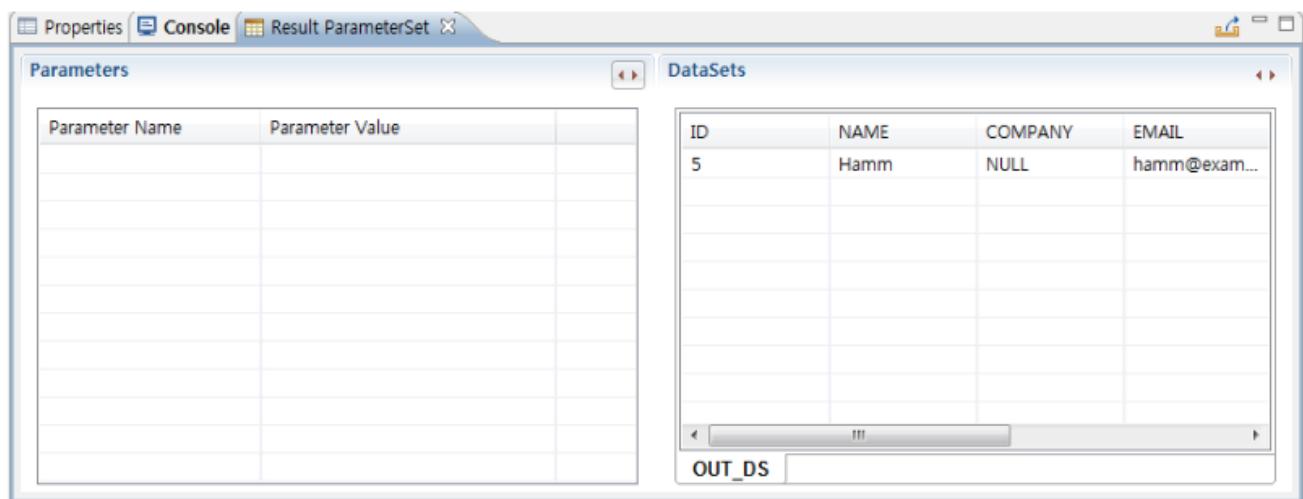


Invoke 테스트 하기

- Select Invoke의 Properties InvokeInfo에 보이는 SQL뷰의 우측 상단의 preview 버튼[]을 클릭합니다. Define Test Value 창에서 Row를 추가하여 '5'를 입력하고 'Finish' 버튼을 클릭합니다.



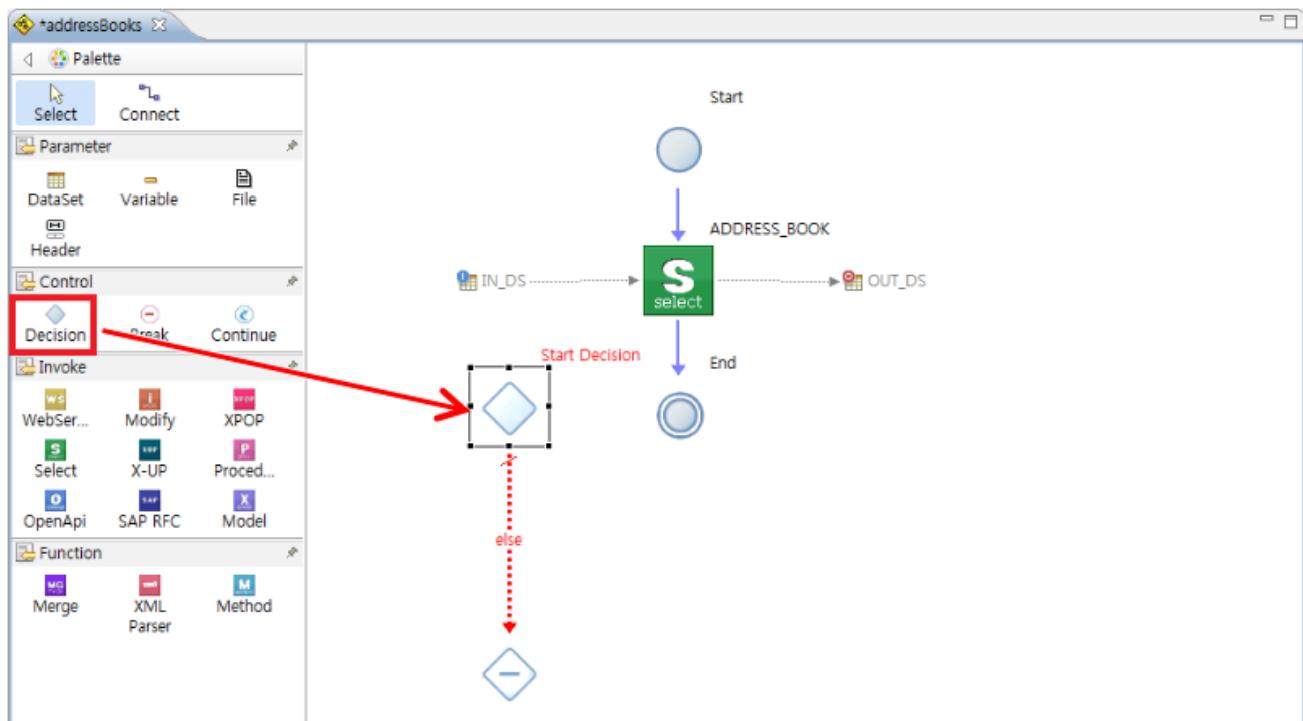
2. Result ParameterSet 뷰에 테스트 결과값이 표시됩니다



Function을 이용한 추가 작업하기

이번 단계에서는 **Select Invoke** 한 결과 출력 데이터셋의 column 'COMPANY' 가 TOBESOFT 이면 companyNameName(String)에 'TOBESOFT'를, 'COMPANY' 가 TOBESOFT 아니면 companyNameName(String)에 'Partners'를 담아서 출력 파라미터로 추가를 하는 방법을 설명합니다.

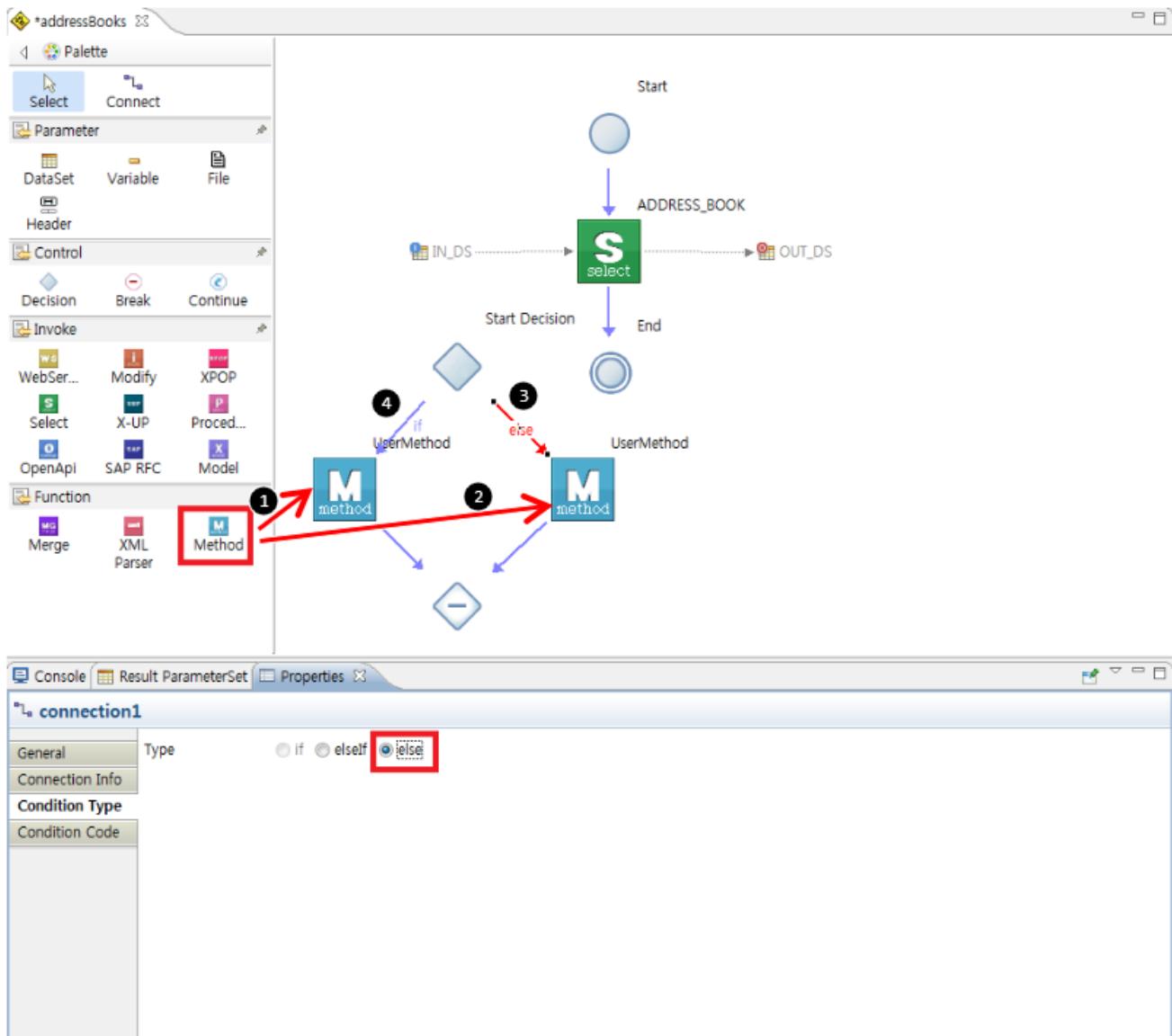
1. 먼저 에디터의 좌측 Palette의 Decision을 적당한 위치에 추가합니다.



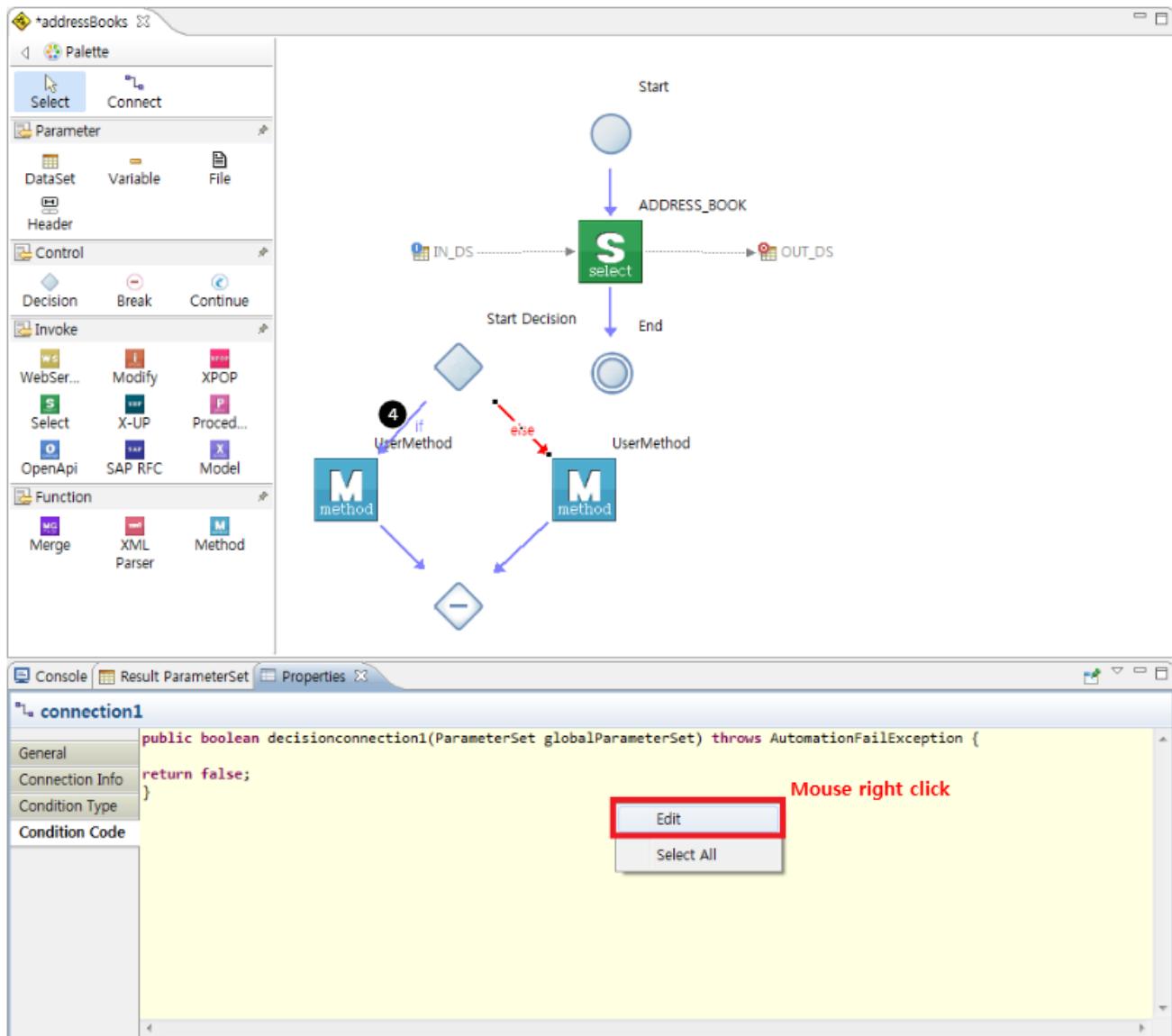
2. 아래 그림 중 1번 추가 작업을 할 수 있는 Method 한 개를 적당한 위치에 추가를 하고 2번 Method 하나 더 추

가 하고 Connect를 이용해 Decision과 연결합니다.

- 3번 if else를 선택하고 아래의 Properties의 'Condition Type'의 else를 선택합니다.

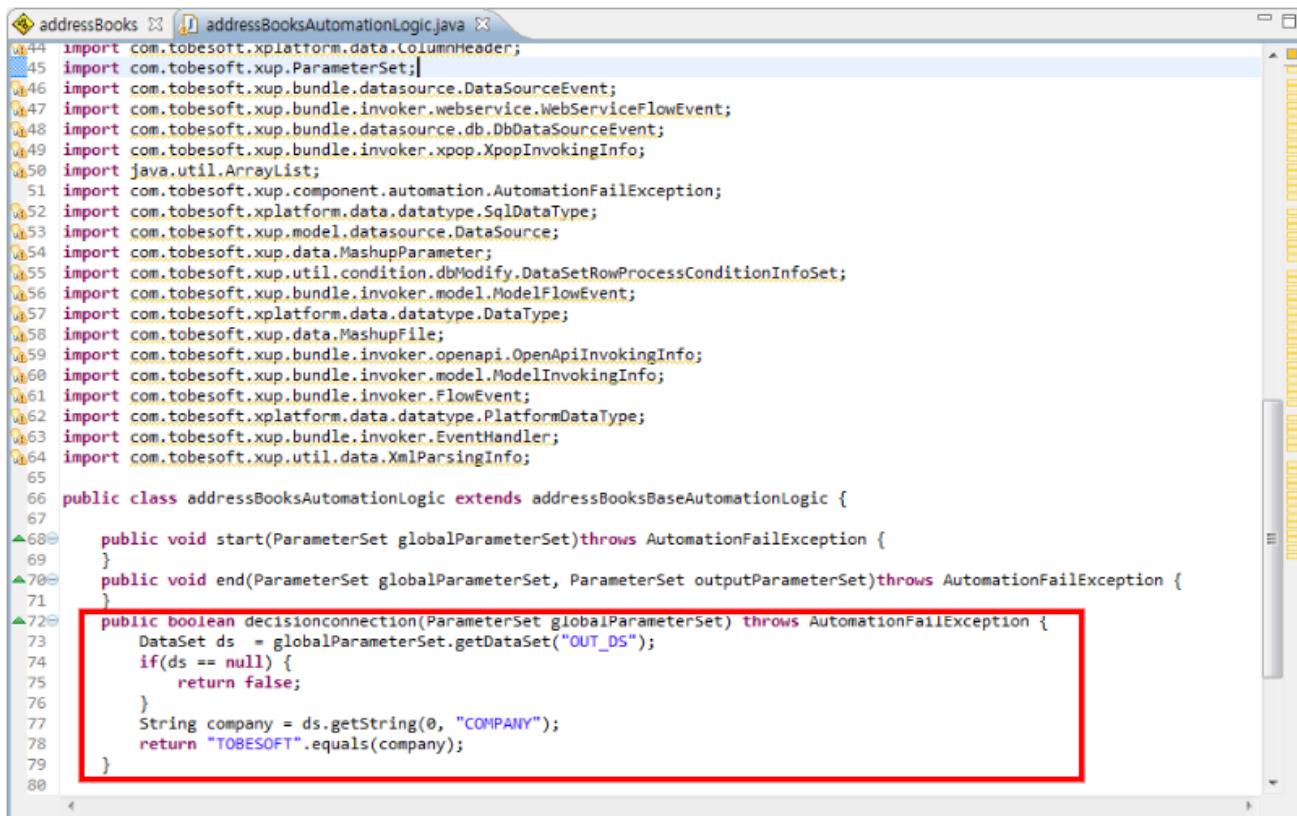


- 4번 if 를 선택하고 아래의 Properties 의 Source Mouse right click Edit을 선택합니다.



5. 자동으로 이동되는 AutomationLogic 클래스에 아래 소스코드를 작성하고 저장합니다.

```
DataSet ds = globalParameterSet.getDataSet("OUT_DS");
if(ds == null) {
    return false;
}
String company = ds.getString(0, "COMPANY");
return "TOBESOFT".equals(company);
```

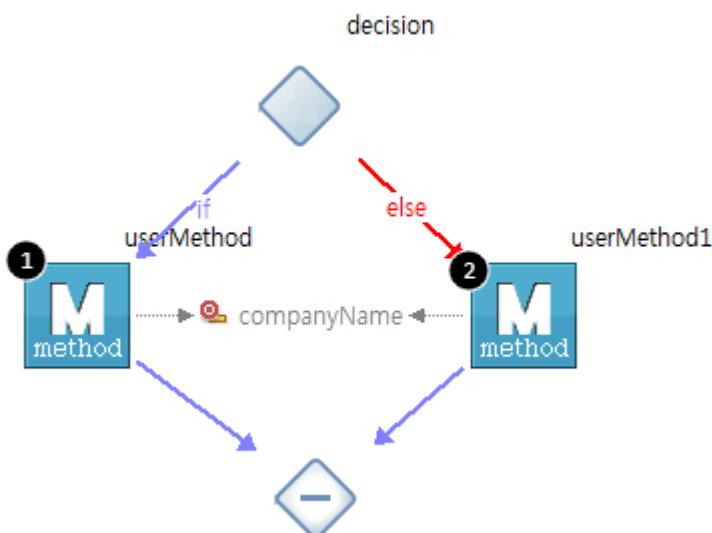


```

addressBooksAutomationLogic.java
1 import com.tobesoft.xplatform.data.ColumnHeader;
2 import com.tobesoft.xup.ParameterSet;
3 import com.tobesoft.xup.bundle.datasource.DataSourceEvent;
4 import com.tobesoft.xup.bundle.invoker.webservice.WebServiceFlowEvent;
5 import com.tobesoft.xup.bundle.datasource.db.DbDataSourceEvent;
6 import com.tobesoft.xup.bundle.invoker.xpop.XpopInvokingInfo;
7 import java.util.ArrayList;
8 import com.tobesoft.xup.component.automation.AutomationFailException;
9 import com.tobesoft.xplatform.datatype.SqlDataType;
10 import com.tobesoft.xup.model.datasource.DataSource;
11 import com.tobesoft.xup.data.MashupParameter;
12 import com.tobesoft.xup.util.condition.dbModify.DataSetRowProcessConditionInfoSet;
13 import com.tobesoft.xup.bundle.invoker.model.ModelFlowEvent;
14 import com.tobesoft.xplatform.data.datatype.DataType;
15 import com.tobesoft.xup.data.MashupFile;
16 import com.tobesoft.xup.bundle.invoker.openapi.OpenApiInvokingInfo;
17 import com.tobesoft.xup.bundle.invoker.model.ModelInvokingInfo;
18 import com.tobesoft.xup.bundle.invoker.FlowEvent;
19 import com.tobesoft.xplatform.data.datatype.PlatformDataType;
20 import com.tobesoft.xup.bundle.invoker.EventHandler;
21 import com.tobesoft.xup.util.data.XmlParsingInfo;
22
23 public class addressBooksAutomationLogic extends addressBooksBaseAutomationLogic {
24
25     public void start(ParameterSet globalParameterSet) throws AutomationFailException {
26     }
27
28     public void end(ParameterSet globalParameterSet, ParameterSet outputParameterSet) throws AutomationFailException {
29     }
30
31     public boolean decisionconnection(ParameterSet globalParameterSet) throws AutomationFailException {
32         DataSet ds = globalParameterSet.getDataSet("OUT_DS");
33         if(ds == null) {
34             return false;
35         }
36         String company = ds.getString(0, "COMPANY");
37         return "TOBESOFT".equals(company);
38     }
39
40 }

```

6. Virable을 하나 추가하고 이름을 ‘companyName’으로 변경합니다. 그 후 두 Method 함수의 Output 파라미터로 companyName을 Connect 합니다.



7. 1번 Method 함수를 더블클릭하여 Properties의 [Source > Mouse right click > Edit]을 선택합니다.
8. 자동 이동되는 AutomationLogic 클래스에 아래 소스코드를 작성하고 저장합니다.

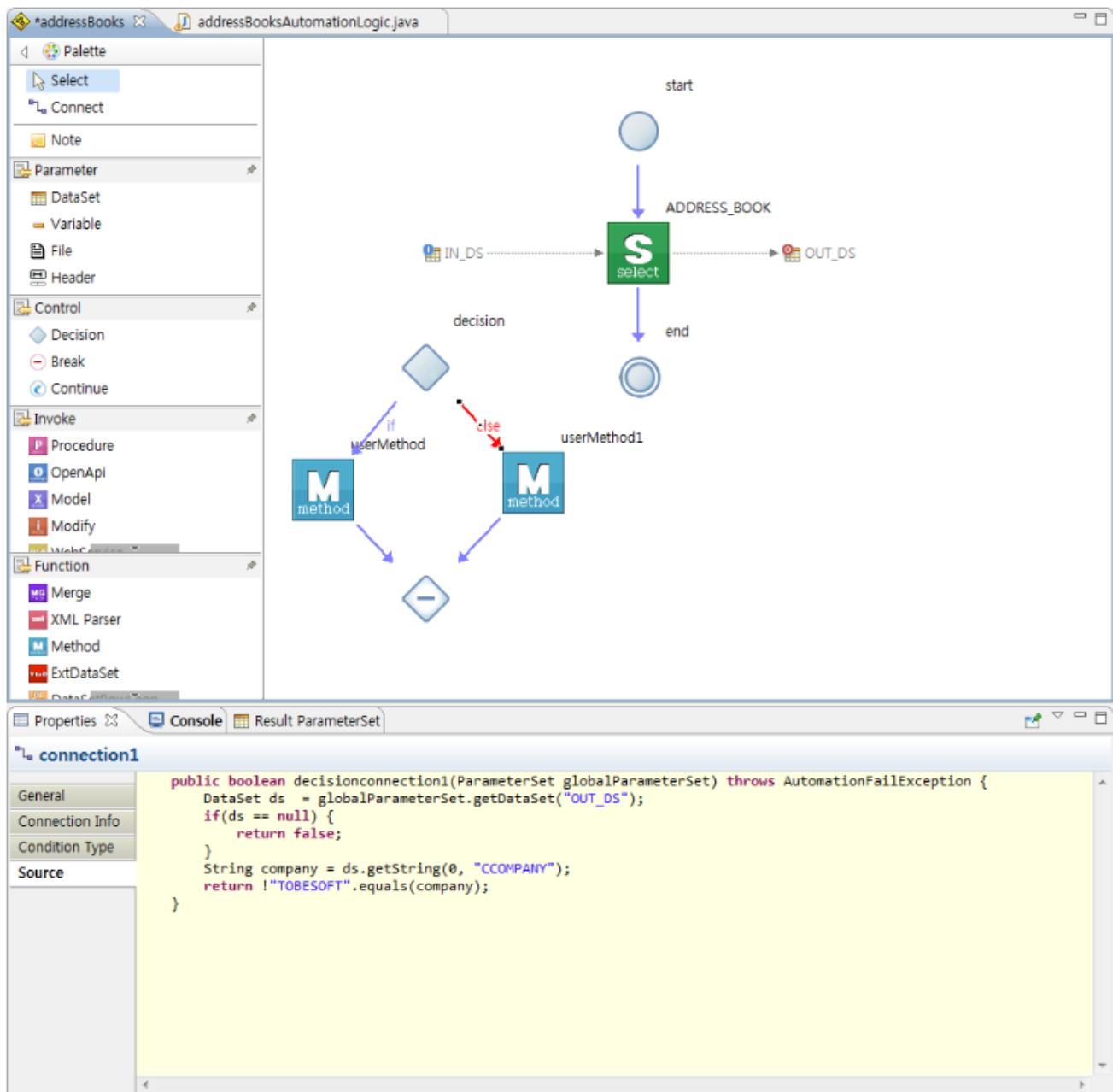
```

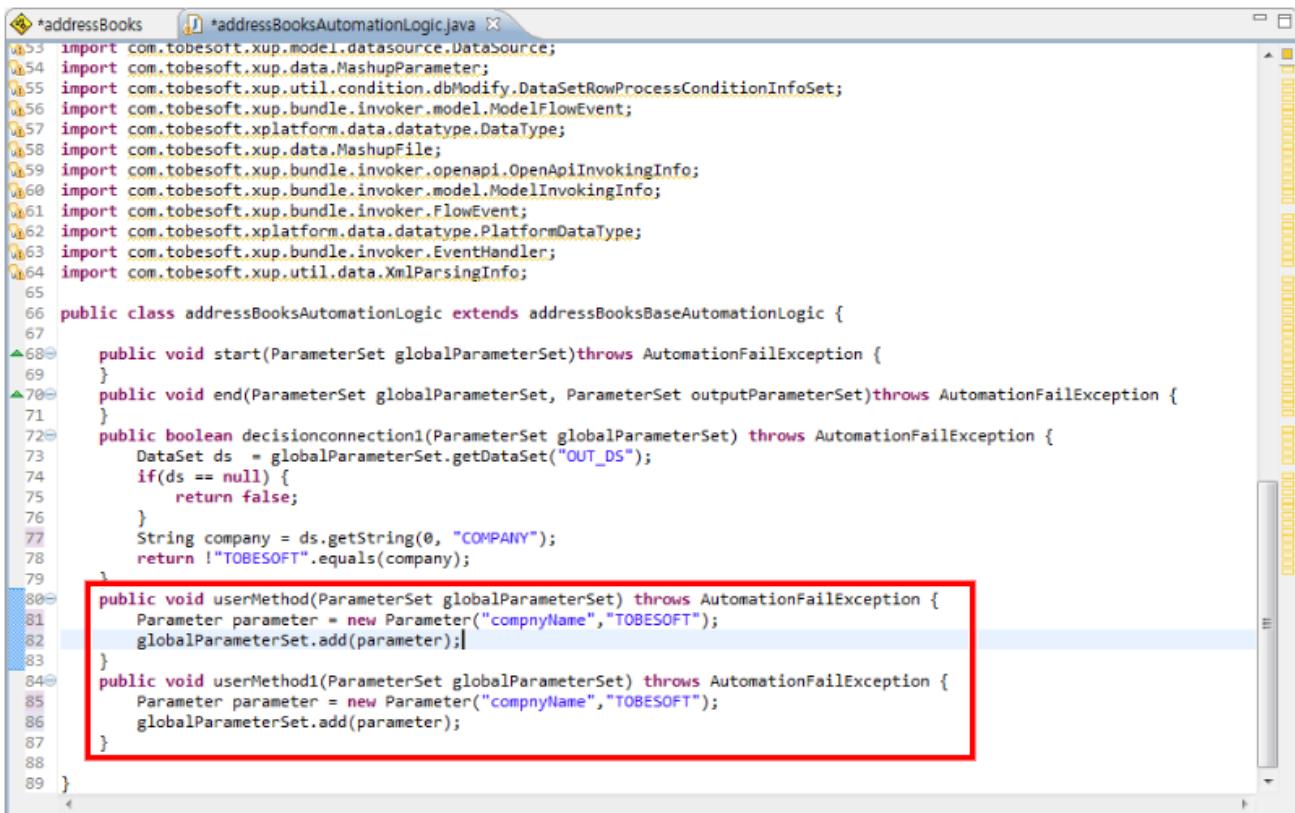
Parameter parameter = new Parameter("compyName", "TOBESOFT");
globalParameterSet.add(parameter);

```

9. 2번 Method 함수를 더블클릭하여 Properties의 [Source > Mouse right click > Edit]을 선택합니다. AutomationLogic 클래스에 아래 소스코드를 작성하고 저장합니다.

```
Parameter parameter = new Parameter("compnyName", "Partner");
globalParameterSet.add(parameter);
```





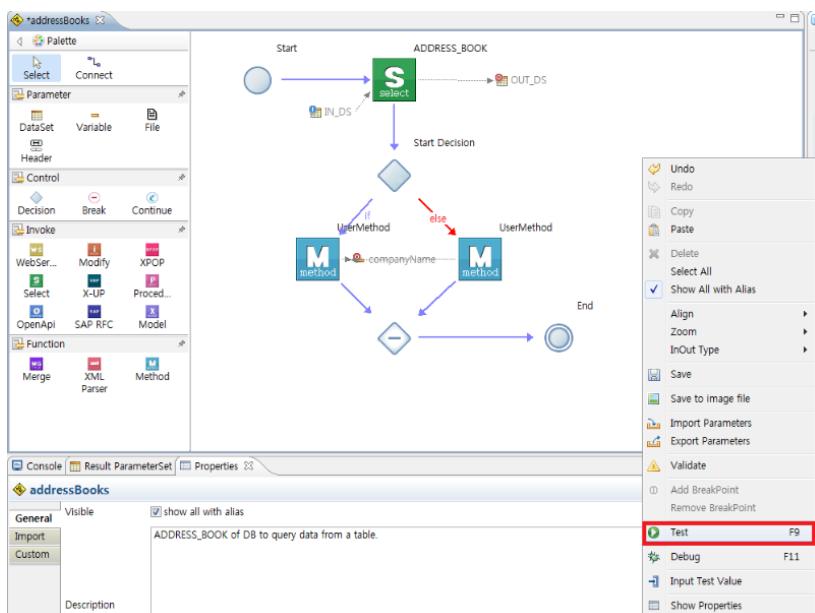
```

53 import com.tobesoft.xup.model.datasource.DataSource;
54 import com.tobesoft.xup.data.MashupParameter;
55 import com.tobesoft.xup.util.condition.dbModify.DataSetRowProcessConditionInfoSet;
56 import com.tobesoft.xup.bundle.invoker.model.ModelFlowEvent;
57 import com.tobesoft.xplatform.data.datatype.DataType;
58 import com.tobesoft.xup.data.MashupFile;
59 import com.tobesoft.xup.bundle.invoker.openapi.OpenApiInvokingInfo;
60 import com.tobesoft.xup.bundle.invoker.model.ModelInvokingInfo;
61 import com.tobesoft.xup.bundle.invoker.FlowEvent;
62 import com.tobesoft.xplatform.data.datatype.PlatformDataType;
63 import com.tobesoft.xup.bundle.invoker.EventHandler;
64 import com.tobesoft.xup.util.data.XmlParsingInfo;
65
66 public class addressBooksAutomationLogic extends addressBooksBaseAutomationLogic {
67
68     public void start(ParameterSet globalParameterSet) throws AutomationFailException {
69     }
70     public void end(ParameterSet globalParameterSet, ParameterSet outputParameterSet) throws AutomationFailException {
71     }
72     public boolean decisionconnection1(ParameterSet globalParameterSet) throws AutomationFailException {
73         DataSet ds = globalParameterSet.getDataSet("OUT_DS");
74         if(ds == null) {
75             return false;
76         }
77         String company = ds.getString(0, "COMPANY");
78         return !"TOBESOFT".equals(company);
79     }
80     public void userMethod(ParameterSet globalParameterSet) throws AutomationFailException {
81         Parameter parameter = new Parameter("compyName", "TOBESOFT");
82         globalParameterSet.add(parameter);
83     }
84     public void userMethod1(ParameterSet globalParameterSet) throws AutomationFailException {
85         Parameter parameter = new Parameter("compyName", "TOBESOFT");
86         globalParameterSet.add(parameter);
87     }
88 }

```

모델 테스트하기

1. Select Invoke와 Start Decision을 Connect를 사용해 연결하고 End Decision과 End를 Connect를 사용하여 연결해 줍니다.
2. 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.



3. 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.
4. Result ParameterSet 뷰의 하단 탭에는 **Parameters**와 **DataSets**의 탭이 있습니다. Parameters의 출력 파라미터로는 ‘companyName’이 있고, DataSets의 출력 파라메터로는 ‘OUT_DS’이라는 이름의 데이터셋이 존재하는 것을 확인 할 수 있습니다.

Parameter Name	Parameter Value
companyName	TOBESOFT

ID	NAME	COMPANY	EMAIL	PHONE
5	Hamm	TOBESOFT	hamm@example.com	NULL

8.3 Modify Invoke를 이용한 모델 개발하기

이 절에서는 Automation 모델을 이용하여 데이터베이스의 데이터를 변경하는 개발방법을 설명합니다.



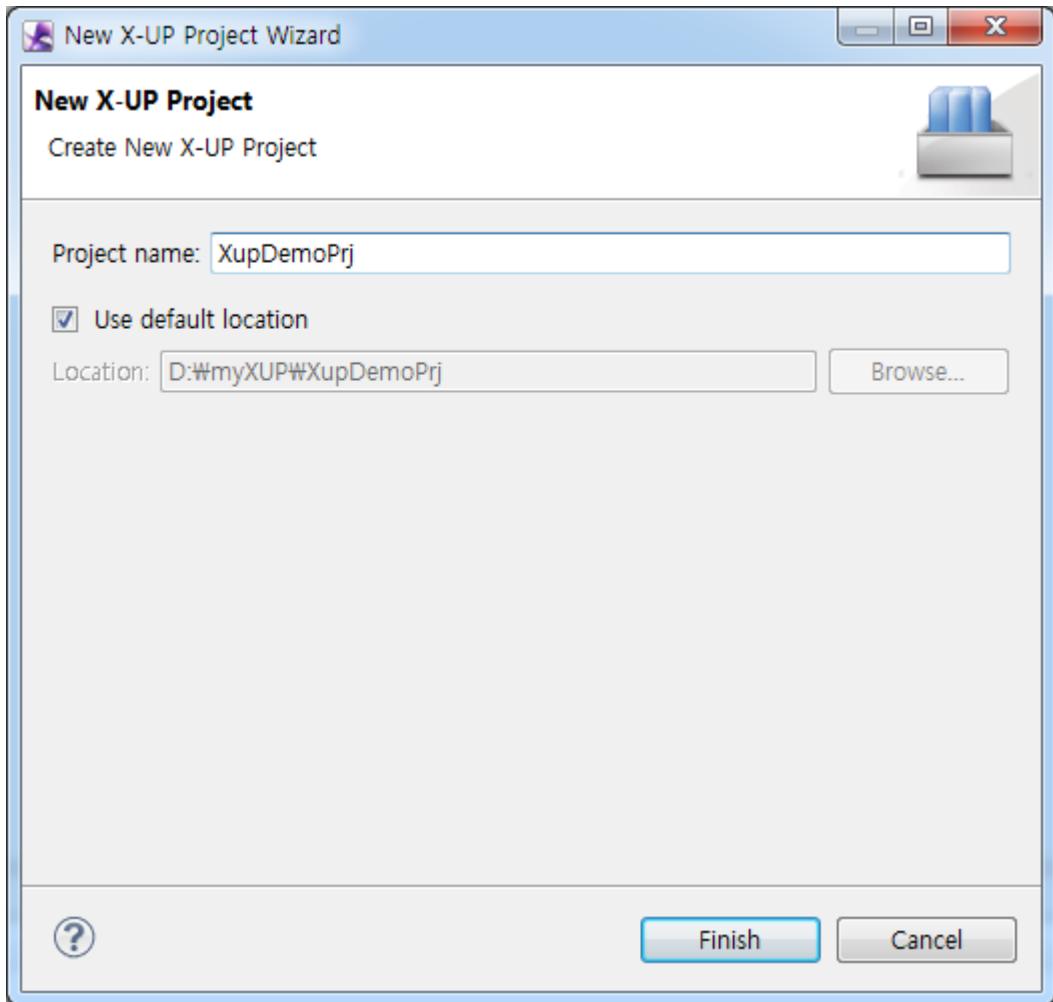
이 절에서 사용한 데이터베이스 테이블 정보는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 참조하십시오.

이 절에서 설명하는 Automation 모델의 Modify Invoke 개발단계는 다음과 같습니다.

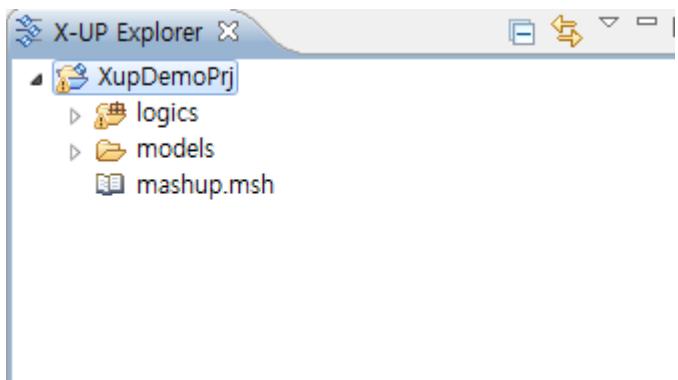
- X-UP 프로젝트 생성하기
- Automation 모델을 생성 및 select Invoke 생성하기
- 입력 파라메터 설정 및 Properties 설정하기
- Invoke 테스트하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭합니다.

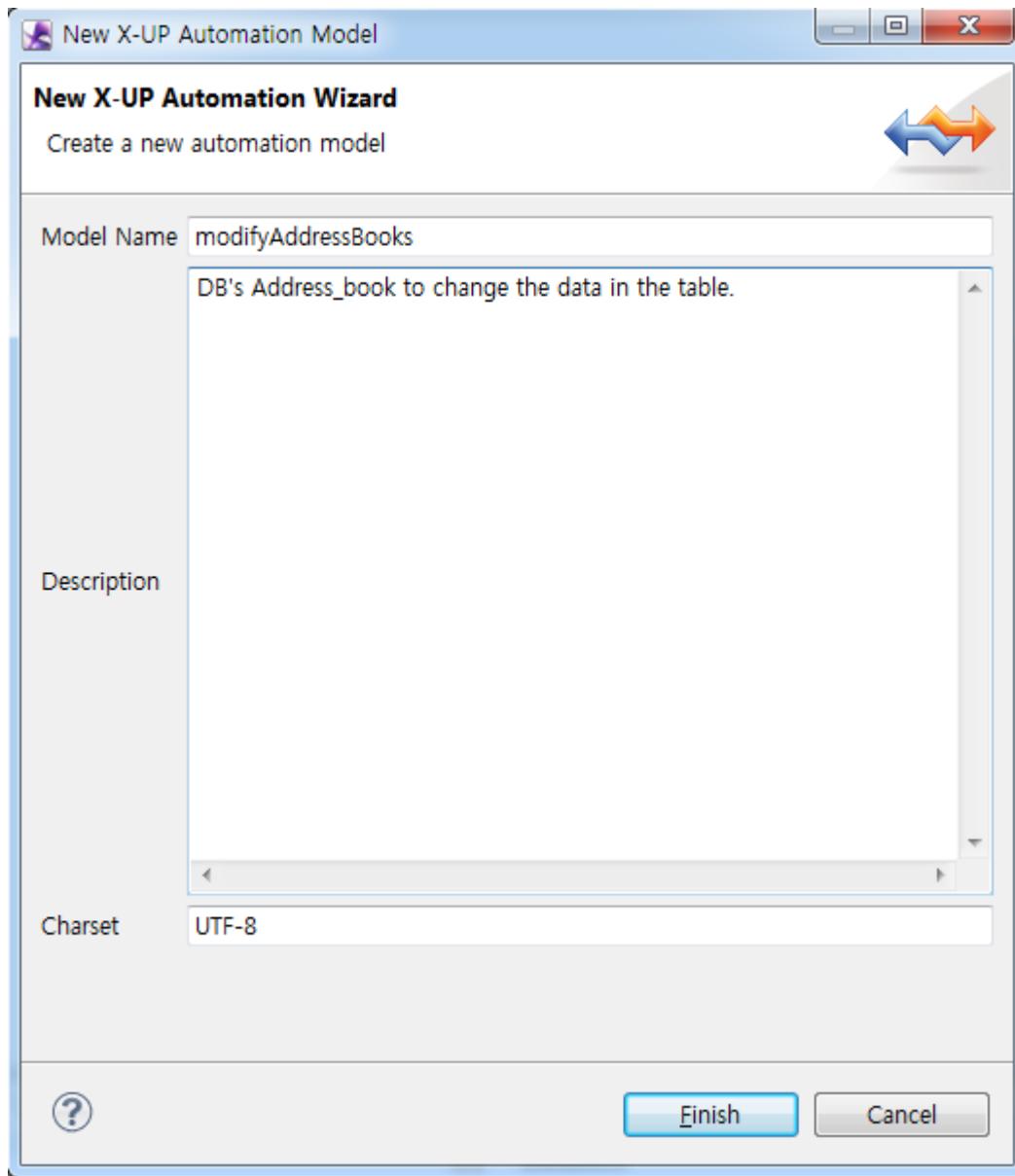


1. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

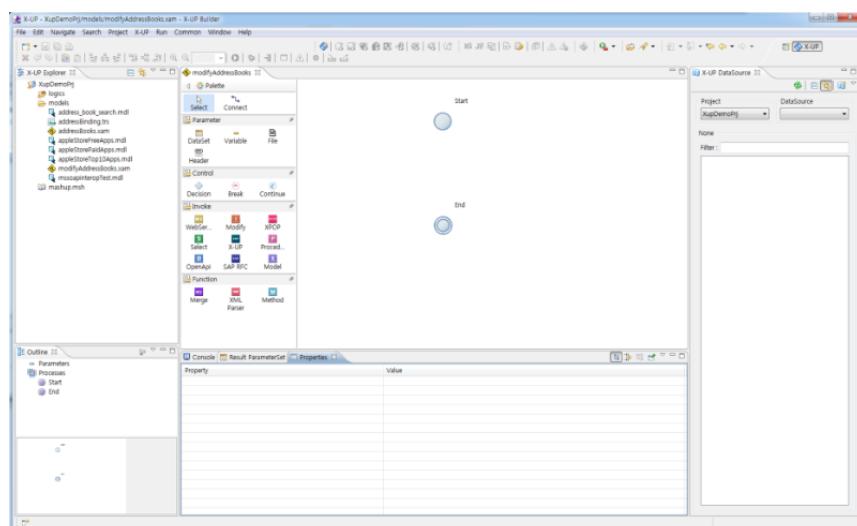


Automation 모델 생성 및 Modify Invoke 생성하기

1. [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 OK 버튼을 클릭합니다.

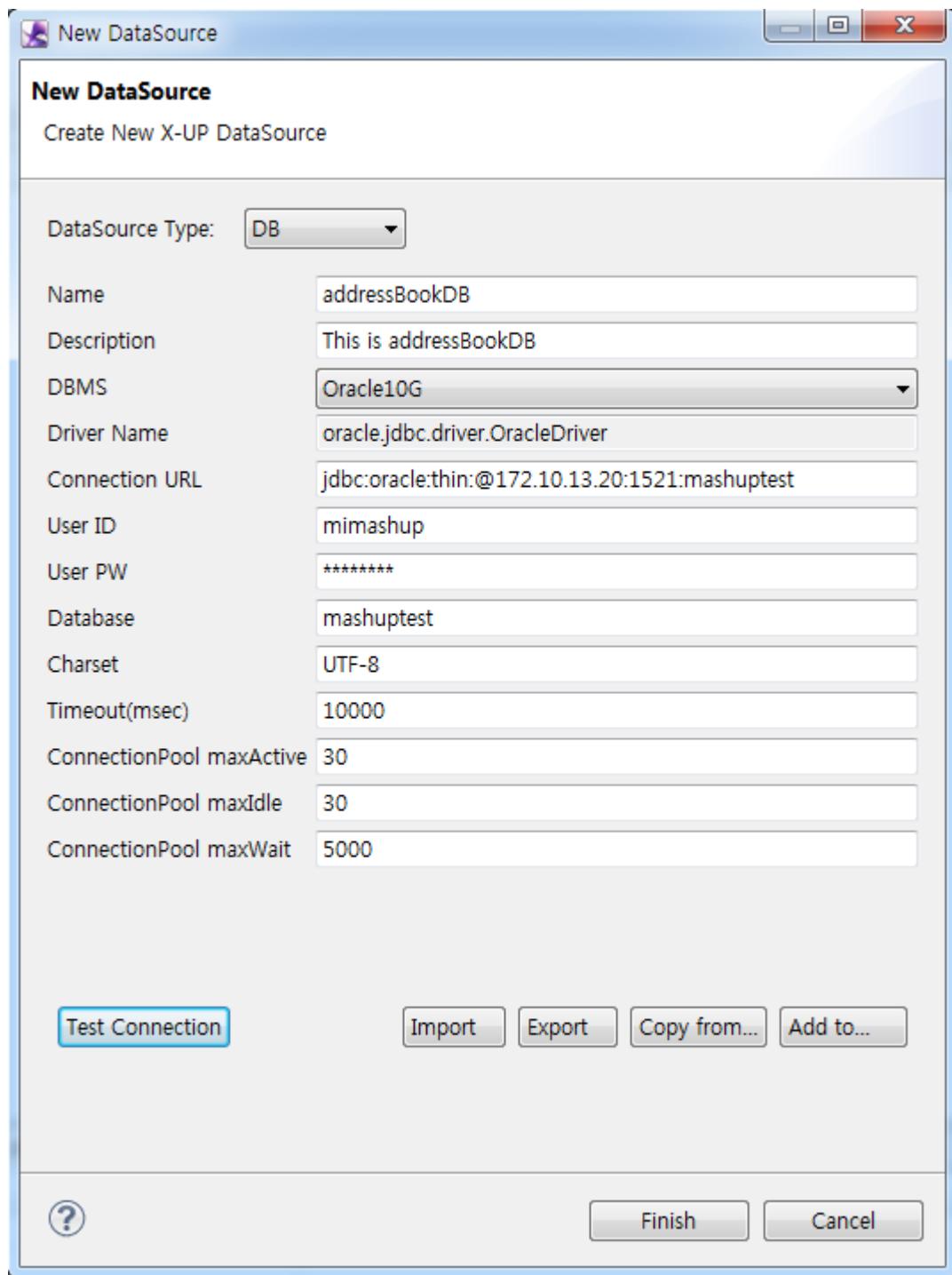


3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



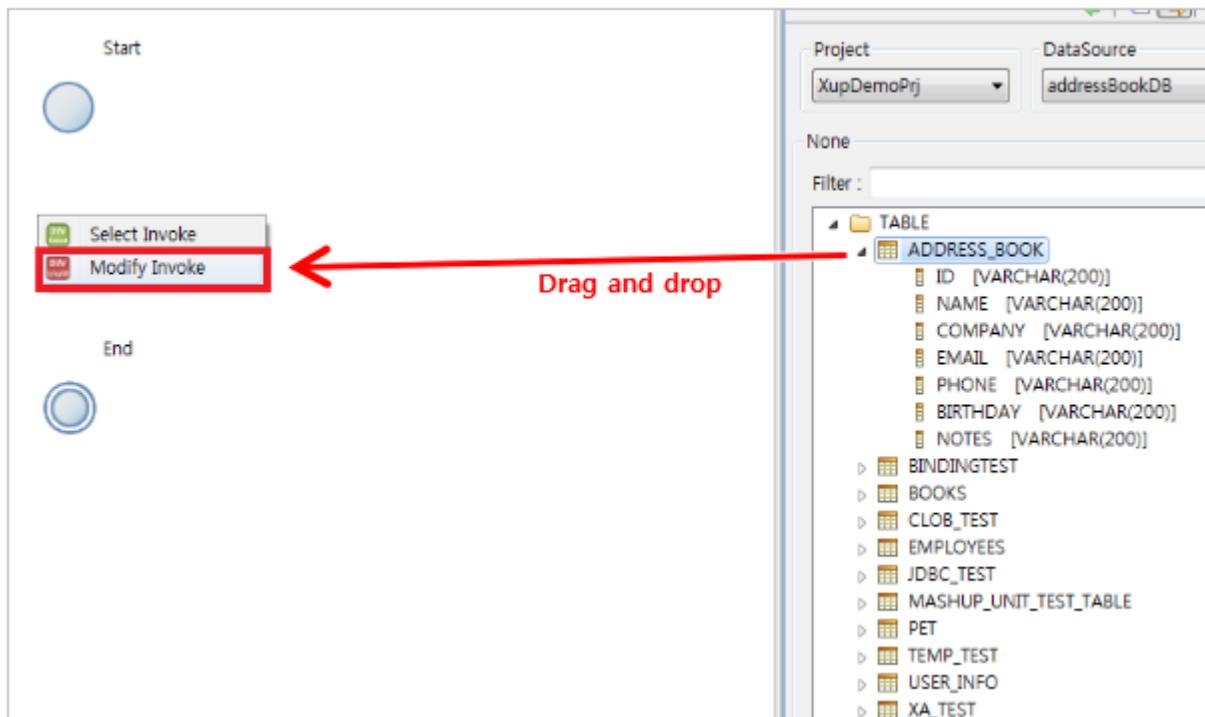
4. 화면 우측의 X-UP DataSource 위자드에서 new DataSource를 선택해 Wizard를 실행합니다.
5. New DataSource 위자드에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
6. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

Field Name	Field Value
Name	'addressBookDB' 데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
DBMS	사용할 데이터베이스 시스템을 선택합니다. 이 예제는 'Oracle'을 사용합니다.
Connection URL	JDBC Connection String을 입력합니다.
User ID	데이터베이스 사용자 ID를 입력합니다.
User PW	데이터베이스 사용자 암호를 입력합니다.
Database	사용할 데이터베이스의 이름을 입력합니다.
Charset	'UTF-8' 문자 인코딩에 대한 값을 입력합니다.
Timeout(msec)	'10000' 서버와의 접속을 유지할 최대 시간
ConnectionPool MaxActive	'30' 사용하는 커넥션 최대수
ConnectionPool MaxIdle	'30' 사용하지 않는 커넥션의 최대수
ConnectionPool maxWait	'5000' 커넥션을 열기 위해 최대 기다리는 시간

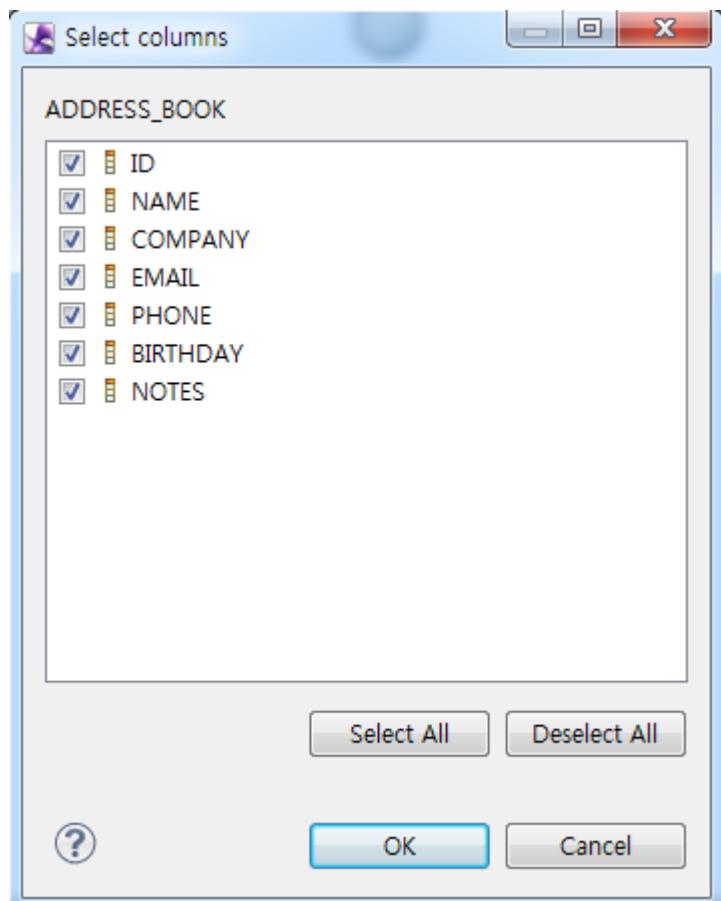


7. X-UP DataSource 뷰에서 DataSource[]를 선택 해 테이블 목록을 가져옵니다.
8. 가져온 테이블 목록에서 'ADDRESS_BOOK' 테이블을 선택하여 에디터로 드래그 앤 드롭합니다.
9. Select Invoke와 Modify Invoke 중 **Modify Invoke**를 선택하여 생성합니다.

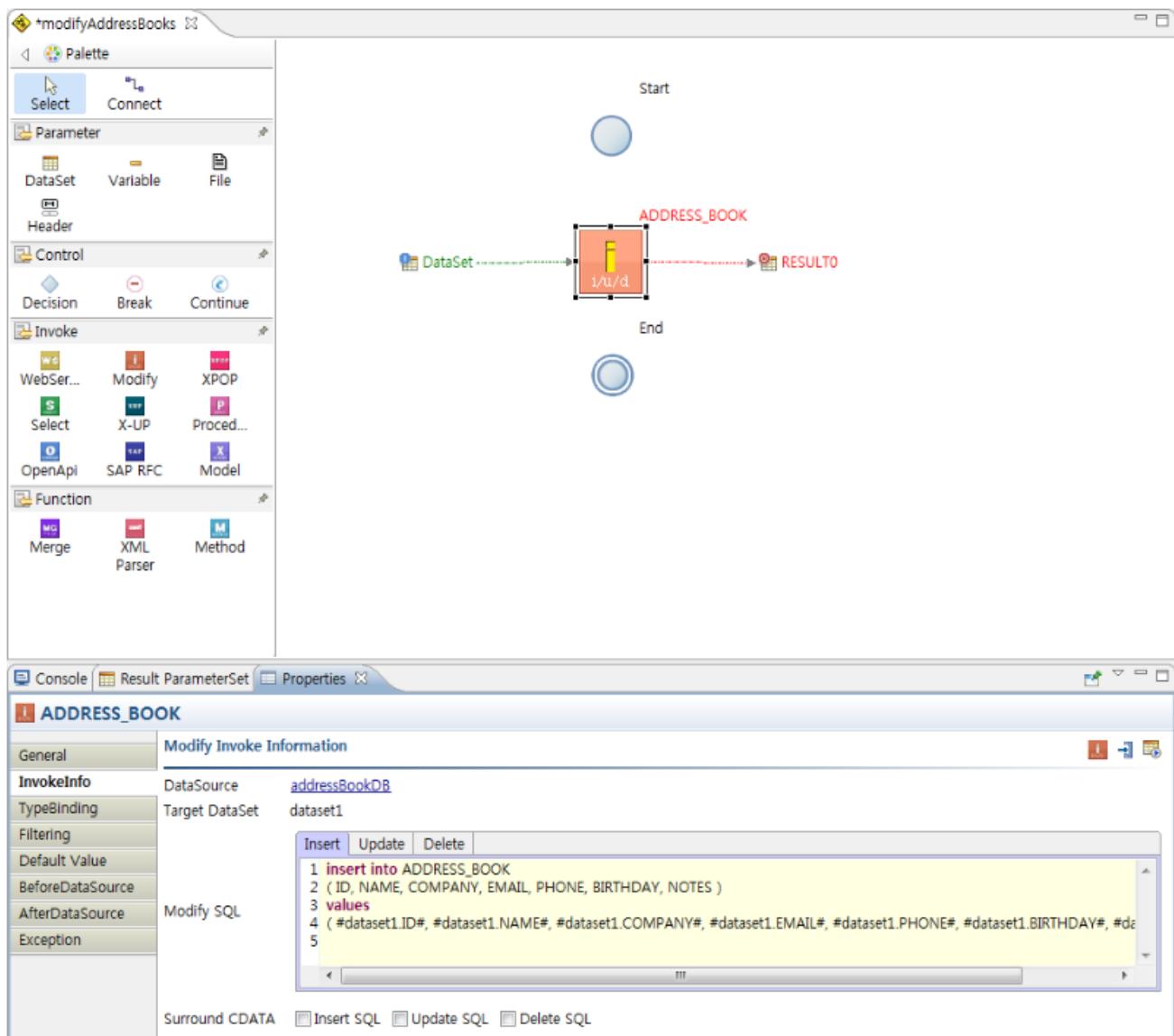
또는 Palette 의 Invoke 중 ‘Modify Invoke’를 선택하고 에디터를 클릭 하게 되면 Modify Invoke를 생성할 수 있습니다.



10. Modify Invoke 생성 시 최초 컬럼을 선택 할 수 있는 위자드가 실행 됩니다.

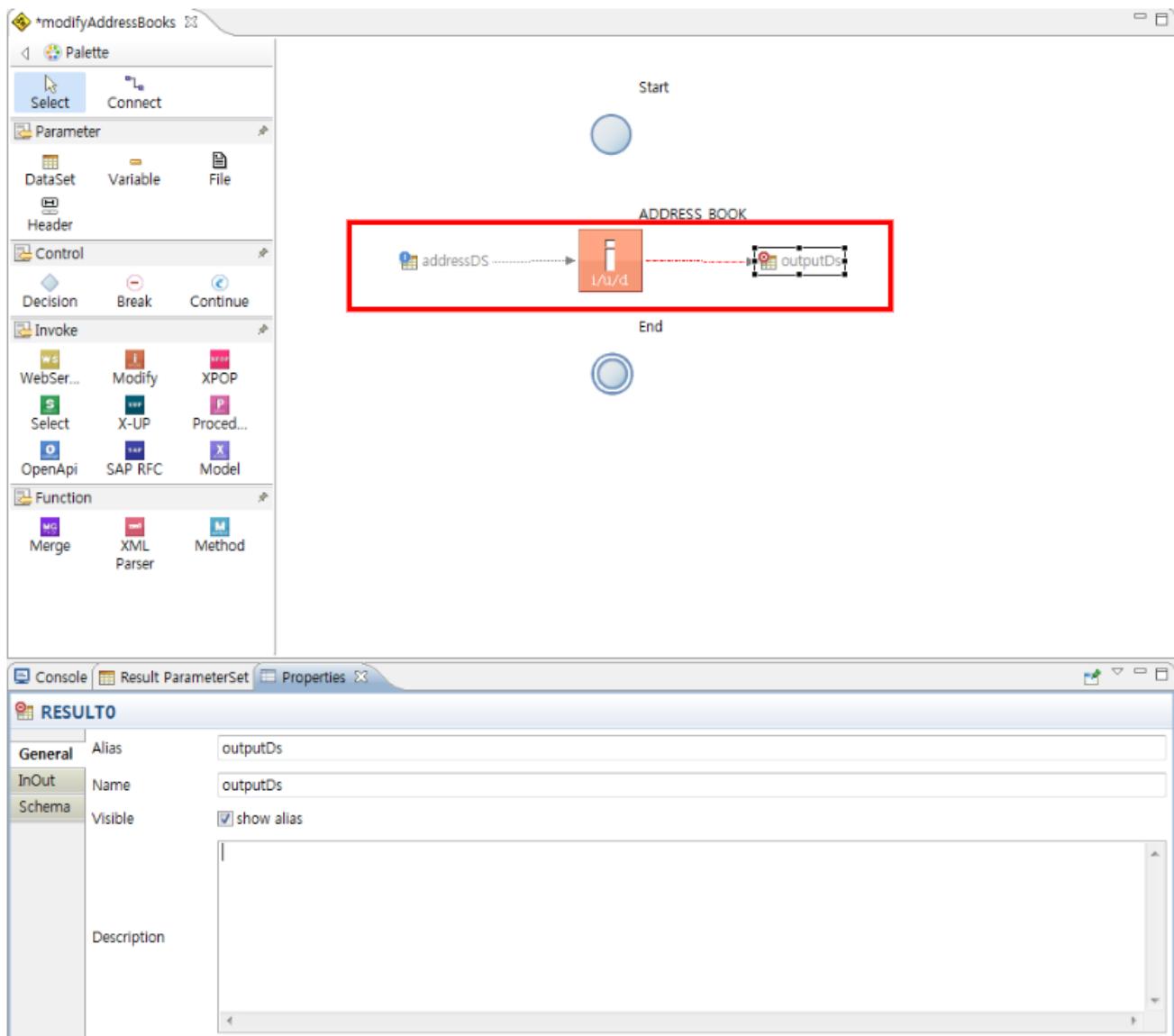


11. 'Select All' 버튼을 선택한 후 'OK'버튼을 선택합니다. 생성 된 Modify Invoke 기본 적으로 입력 데이터셋과 출력 데이터셋을 파라메터로 가지게 됩니다.

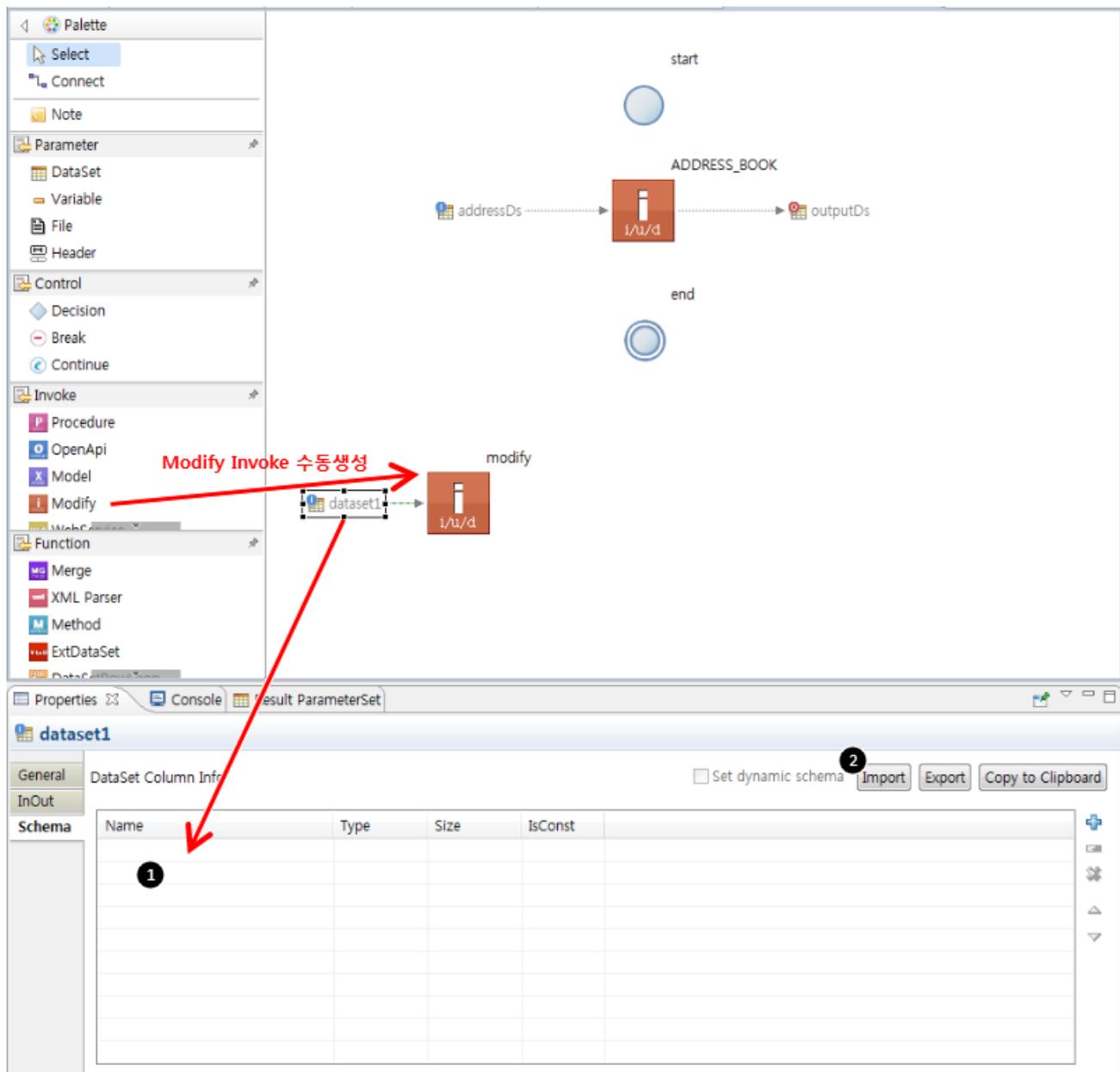


입력 파라메터 설정 및 Properties 설정하기

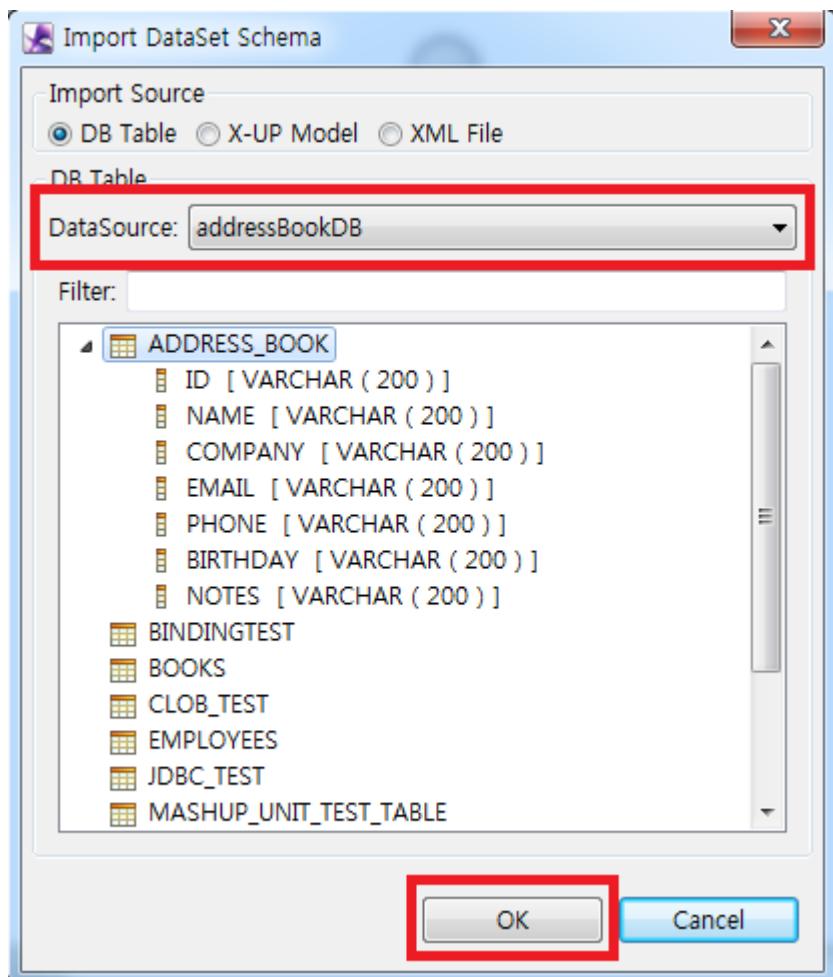
1. 입력 데이터셋을 클릭하여 Properties의 'General'을 선택하여 이름과 Alias를 'addressDs'로 변경합니다.
2. 출력 데이터셋을 클릭하여 Properties의 'General'을 선택하여 이름과 Alias를 클릭하여 'outputDs'로 변경합니다.



3. 좌측 Palette에서 Modify Invoke를 생성 하였을 경우 입력 파라미터를 수동으로 생성(아래그림 1번)하여 입력 파라미터 'addressDs'의 Schema를 지정해야 합니다. Import 버튼을 클릭(아래그림 2번)합니다.
4. 데이터셋의 컬럼 설정은 'Add' 버튼[+]을 클릭하여 사용자가 직접 입력하거나 'Import' 버튼을 클릭하여 데이터베이스 테이블 또는 다른 X-UP 모델, XML 파일에서 컬럼 정보를 가져올 수 있습니다.



5. Import 데이터셋 Schema 위치드에서 DB Table의 데이터소스 ‘addressBookDB’의 ‘ADDRESS_BOOK’ 테이블의 스키마를 선택합니다.

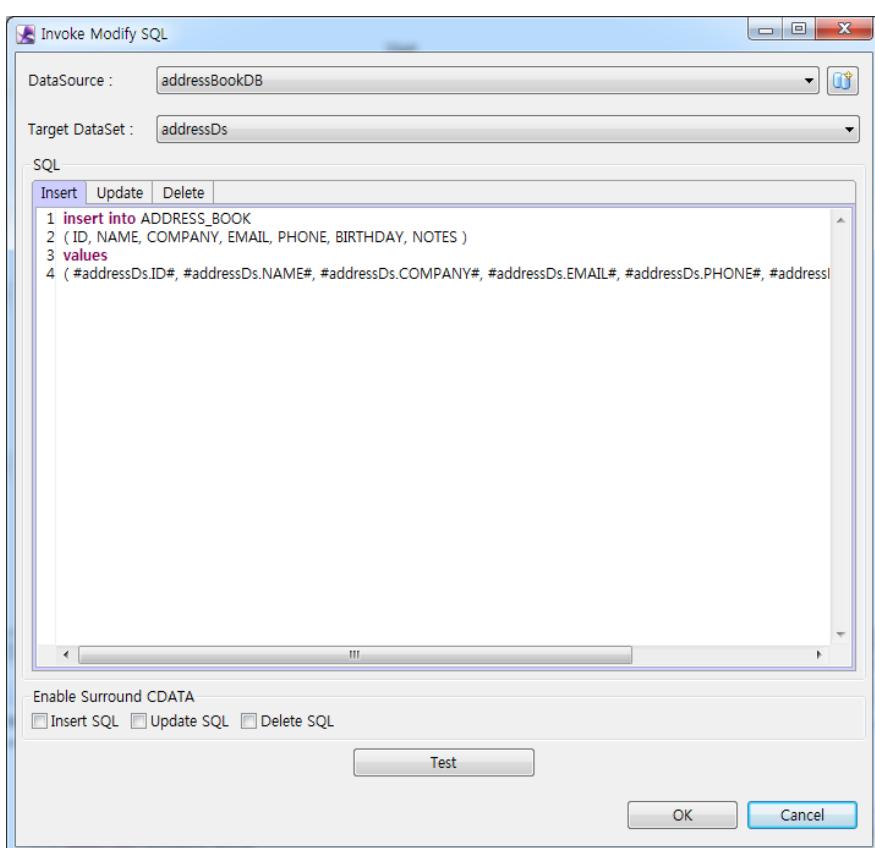
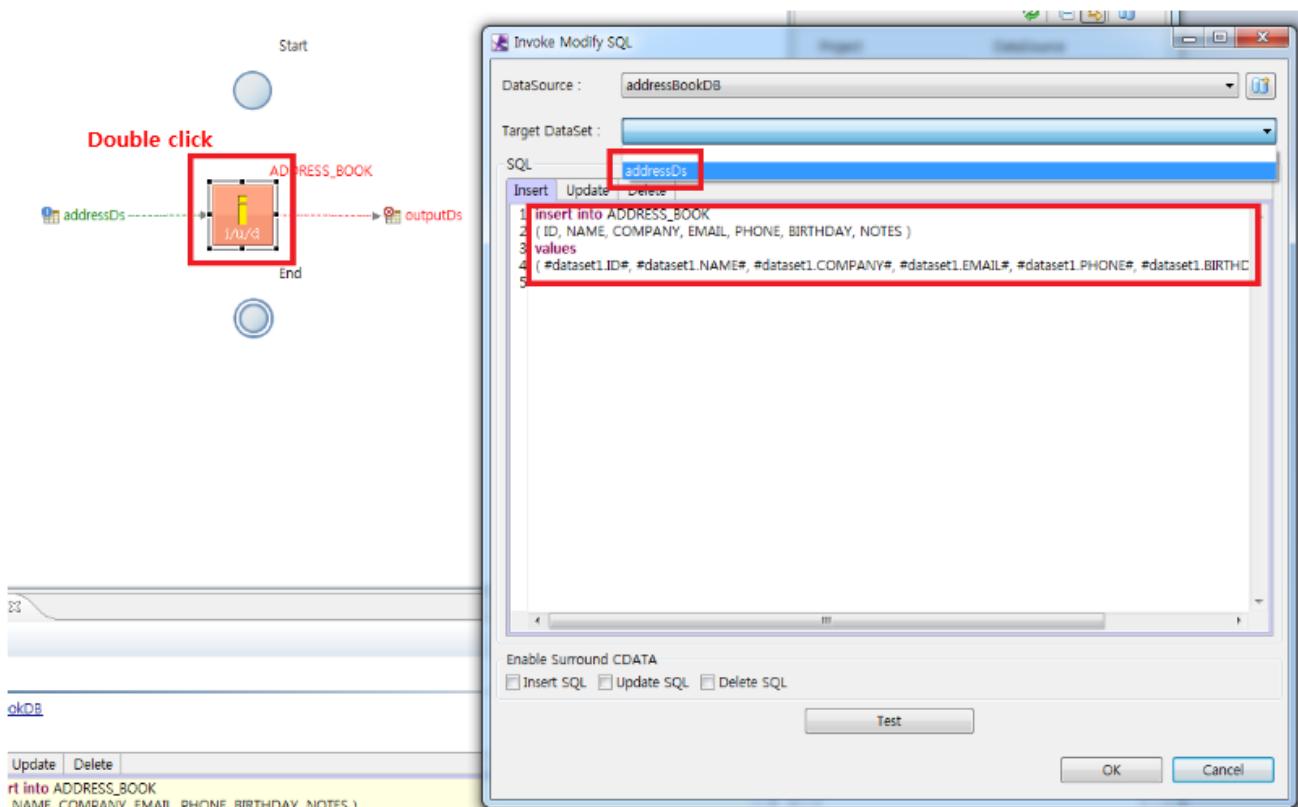


Import Source	설명
DB Table	데이터베이스의 테이블로부터 컬럼 정보를 가져옵니다. 현재 X-UP 프로젝트에 하나 이상의 DB 데이터소스가 설정되어 있어야 사용 가능합니다.
X-UP Model	현재 X-UP 프로젝트에 존재하는 다른 모델의 출력 데이터셋을 이용하여 컬럼 정보를 가져옵니다.
XML File	XML 파일로부터 데이터셋의 컬럼 정보를 가져옵니다. 해당 컬럼정보는 투비소프트에서 제공하는 데이터셋의 Schema 형식이어야 사용 가능합니다.

6. Import 데이터셋 Schema 위자드에서 설정한 컬럼 정보가 'Schema'에 아래와 같이 표시됩니다.

Name	Type	Size	IsConst
ID	string	200	<input type="checkbox"/>
NAME	string	200	<input type="checkbox"/>
COMPANY	string	200	<input type="checkbox"/>
EMAIL	string	200	<input type="checkbox"/>
PHONE	string	200	<input type="checkbox"/>
BIRTHDAY	string	200	<input type="checkbox"/>
NOTES	string	200	<input type="checkbox"/>

7. 변경은 Modify Invoke를 더블클릭하고 Invoke Modify SQL 팝업창의 Target DataSet 콤보박스가 나오는데 입력 파라메터 중 데이터셋의 종류를 보여줍니다. 여기서는 ‘addressDs’를 선택합니다. 자동으로 Insert / Update / Delete 쿼리가 입력된 것을 확인할 수 있습니다.





Enable Surround CDATA

Invoke Modify SQL 창에서 아래 옵션 메뉴인 Enable Surround CDATA는 체크항목에 대해서만 CDATA 처리를 수행합니다. 예를 들어, Insert SQL만 선택했다면 Insert 쿼리에만 적용이 되고 전부 선택했다면 모든 쿼리를 수행할 때마다 CDATA 처리를 수행합니다.

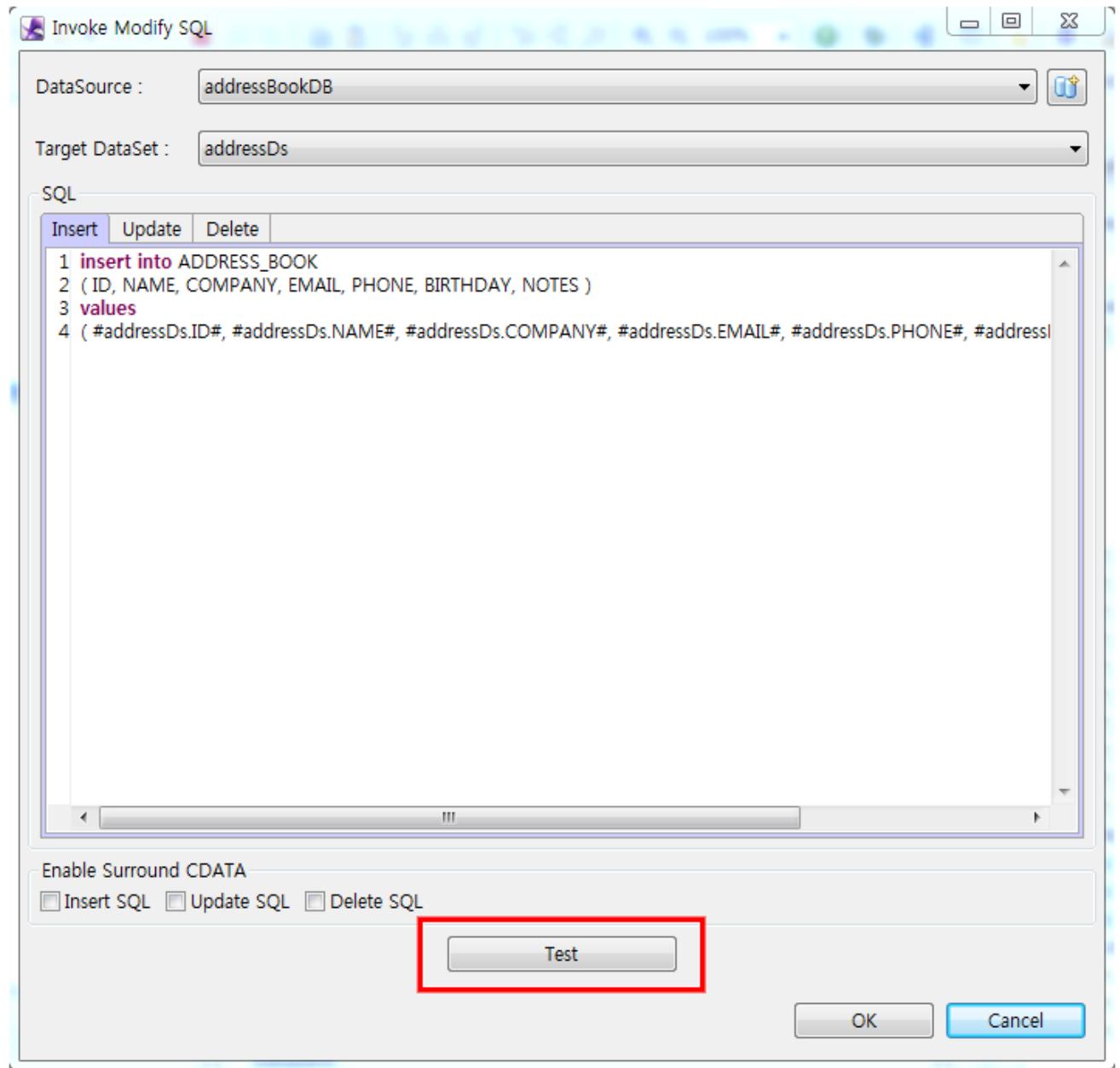
8. Modify Invoke의 **Filtering** 기능을 설정합니다. [Modify Invoke click > Properties > Filtering]
9. DB 테이블 'ADDRESS_BOOK'은 ID 컬럼이 primary key 이기 때문에 ID에 값이 없을 경우나, ID 값이 0일 경우 Filtering 기능을 사용하도록 합니다. 아래와 같이 Filtering Rule을 설정합니다.

Rule Name	Parameter	Comparator	Value
idRule1	addressDs.ID	empty	
idRule2	addressDs.ID	=	0

Rule Name	Parameter	Comparator	Value
idRule1	addressDs.ID	empty	
idRule2	addressDs.ID	=	0

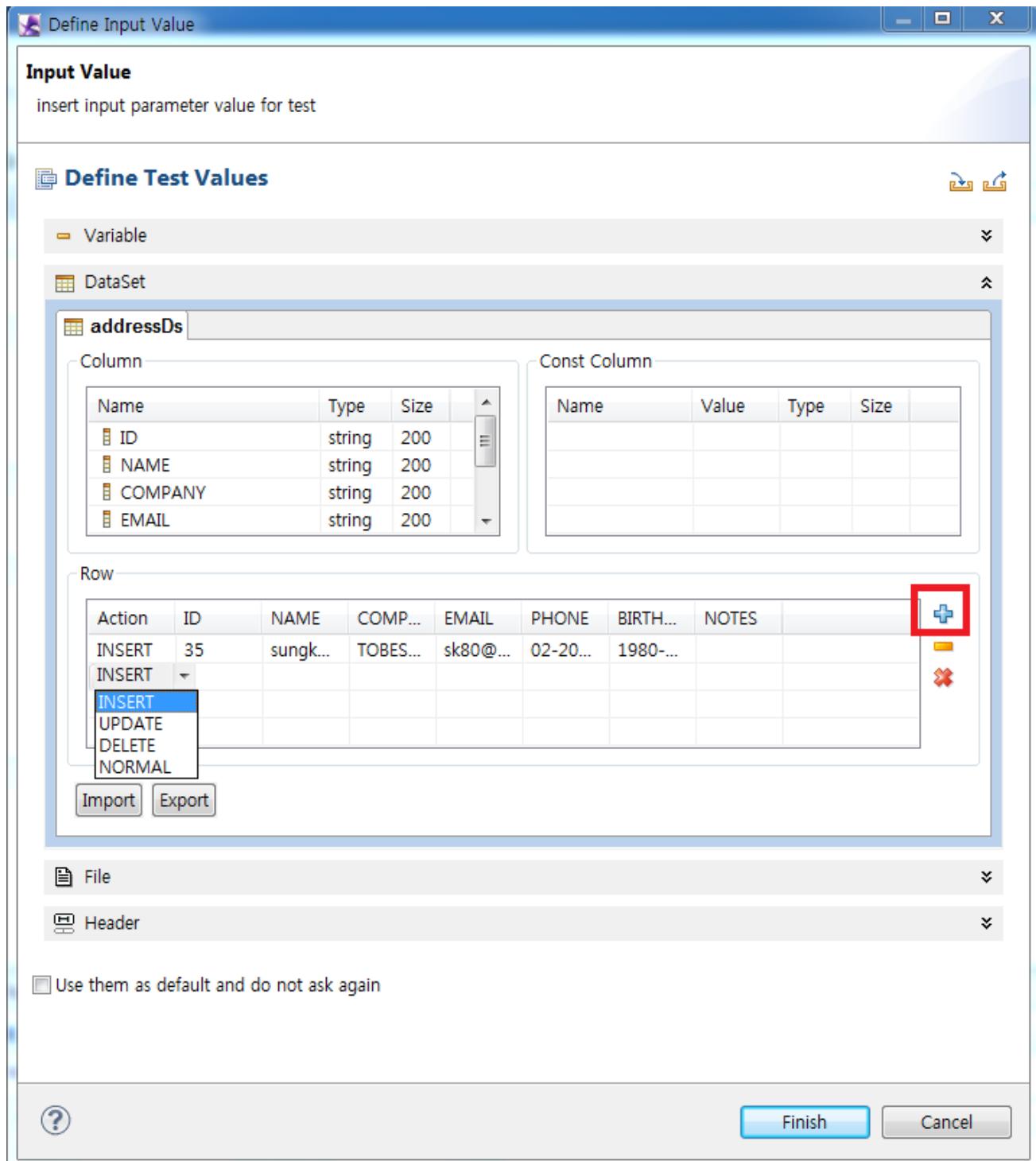
Invoke 테스트 하기

1. Start, End 와 Modify Invoke를 연결선을 이용하여 연결 합니다.
2. 테스트를 하기 위해 Modify Invoke를 더블 클릭하여 Test를 클릭합니다.



3. Modify Invoke 테스트에 사용할 데이터를 입력합니다. **Input Value** 위자드에서 1번 Add 버튼[+]을 클릭하여 데이터를 입력합니다.

데이터는 Filtering 기능을 확인 해야 하기 때문에 id값이 0인 데이터도 입력합니다.



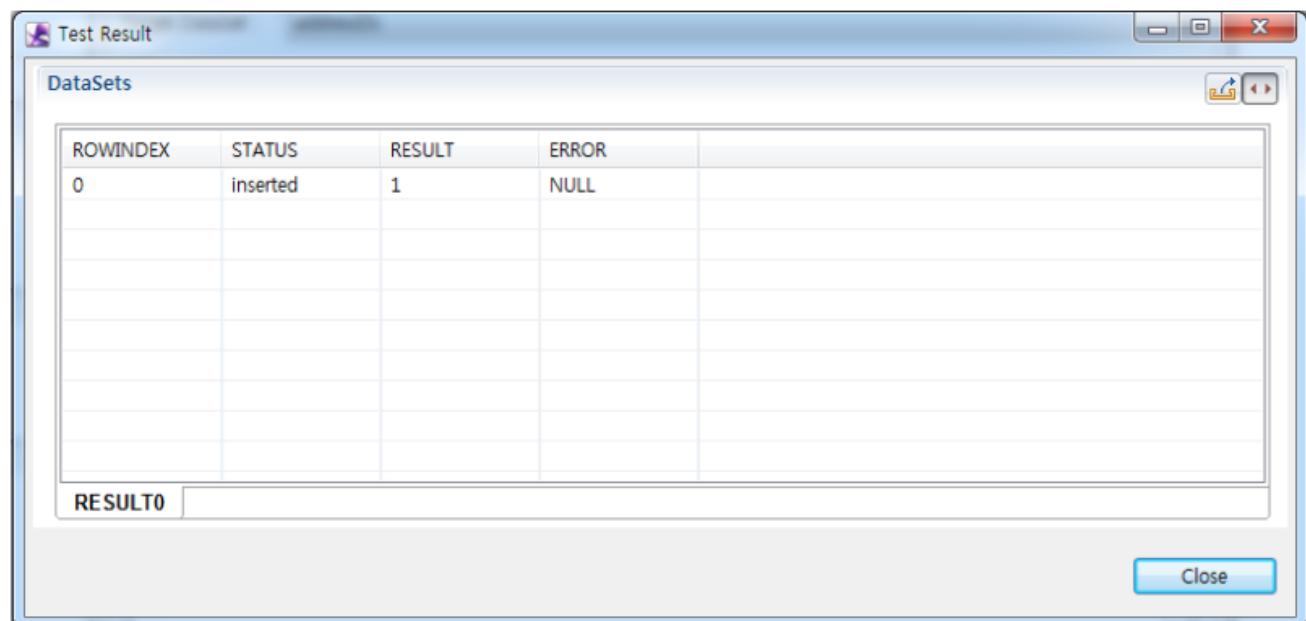
4. 데이터를 입력하고 난 후 'Finish' 버튼을 클릭합니다.

입력한 Row의 트랜잭션 처리 방법(Action) 필드에 대한 설명은 아래를 참조하십시오.

Action	Description
INSERT	'INSERT'가 설정된 Row는 DB 테이블에 추가됩니다.
UPDATE	'UPDATE'가 설정된 Row는 테이블에서 동일한 레코드를 찾아 업데이트를 수행합니다. 동일한 레코드는 기본 키(Primary Key) 컬럼의 값이 입력 데이터셋의 값과 동일한 레코

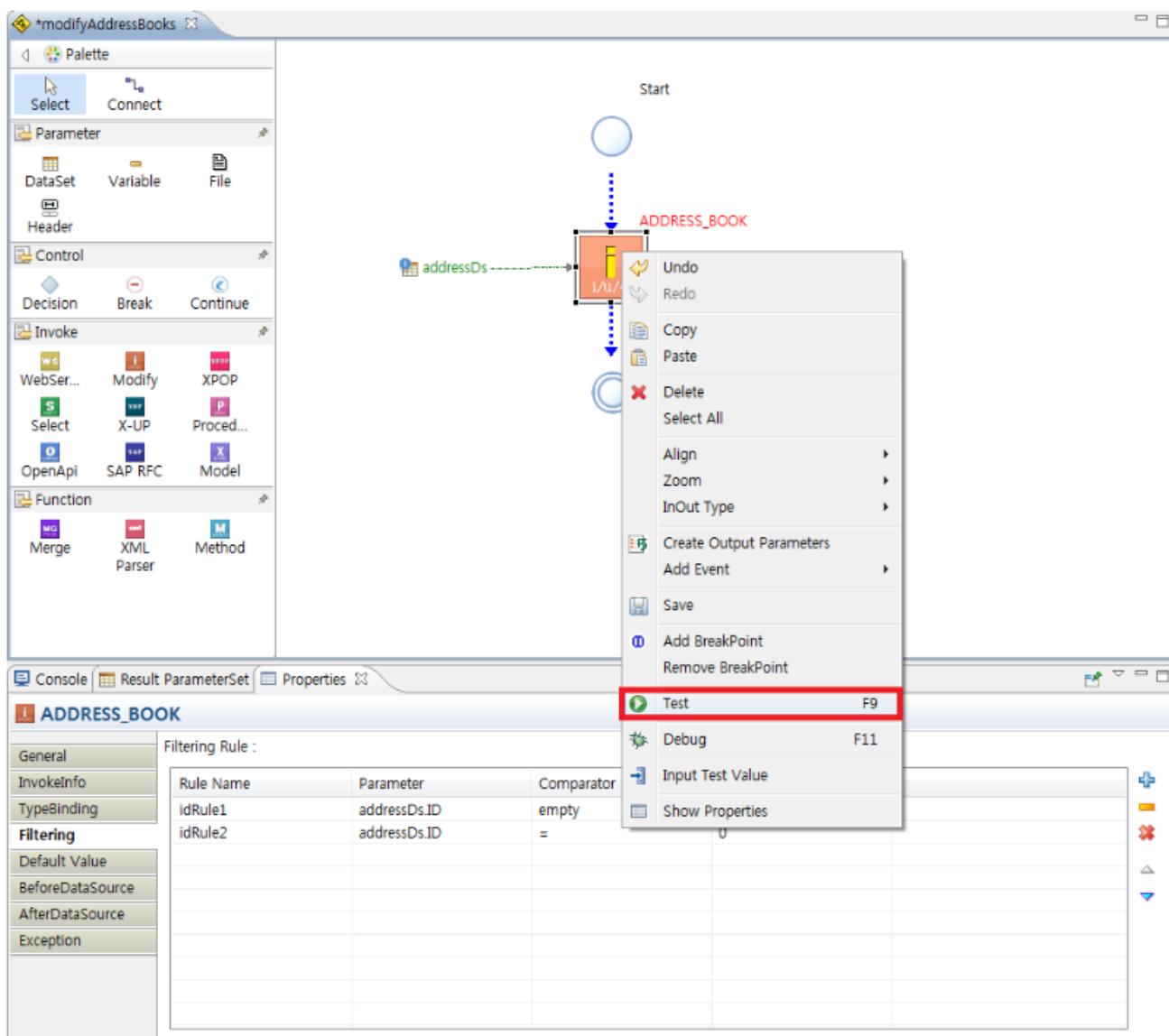
Action	Description
	드를 의미합니다. 그러므로 하나 이상의 기본 키 컬럼이 설정된 테이블에서만 ‘UPDATE’를 수행 할 수 있습니다.
DELETE	‘DELETE’가 설정된 Row는 테이블에서 동일한 레코드를 찾아 삭제합니다. 동일한 레코드는 기본 키(Primary Key) 컬럼의 값이 입력 데이터셋의 값과 동일 한 레코드를 의미합니다. 그러므로 하나 이상의 기본 키 컬럼이 설정된 테이블에서만 ‘DELETE’를 수행 할 수 있습니다.
NORMAL	입력 데이터로만 사용되며 DB 트랜잭션에는 관여하지 않습니다.

5. 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인할 수 있습니다.



모델 테스트하기

1. Start 노드와 Modify Invoke, End 노드를 Connect를 사용해 연결해줍니다.
2. 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.
3. 테스트 데이터는 ‘Invoke 테스트하기’ 와 동일한 테스트 데이터를 입력하여 테스트를 합니다.
4. 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.
5. Result ParameterSet 뷰의 하단 탭에는 Parameters와 DataSets의 탭이 있습니다. DataSets의 출력 파라미터로는 ‘outputDs’가 존재하는 것을 확인 할 수 있습니다.



Parameter Name	Parameter Value

RowIndex	Status	Result	Error
0	inserted	1	NULL

8.4 Procedure Invoke를 이용한 모델 개발하기

이 절에서는 Automation 모델을 이용하여 데이터베이스의 Procedure or function을 호출하는 개발방법을 설명합니다.



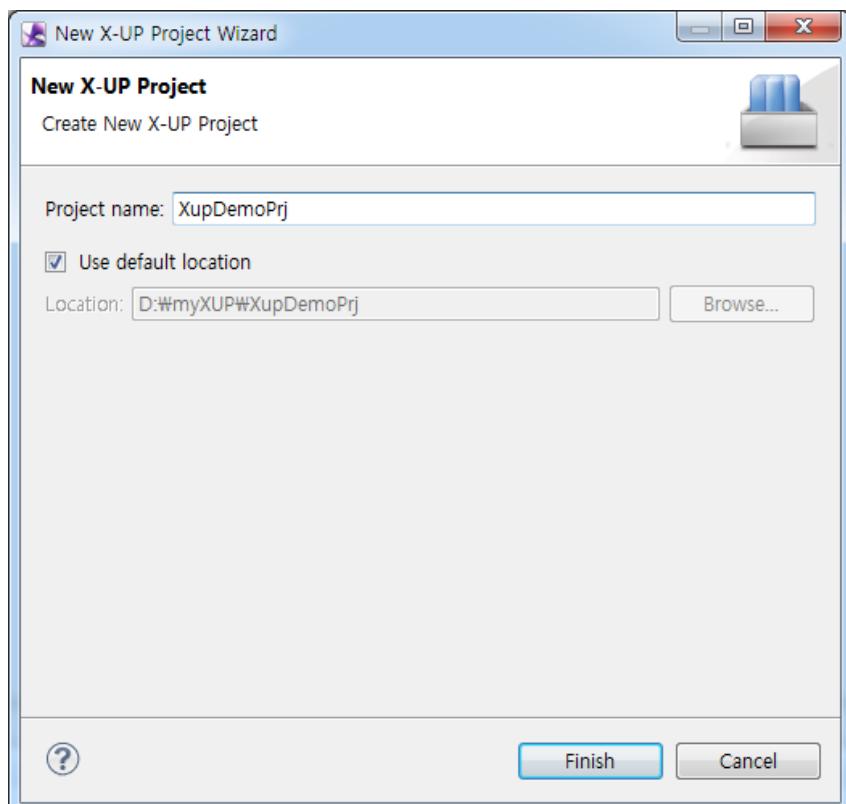
이 절에서 사용한 데이터베이스 테이블 정보는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 참조하십시오.

이 절에서 설명하는 Automation 모델의 Procedure Invoke 개발단계는 다음과 같습니다.

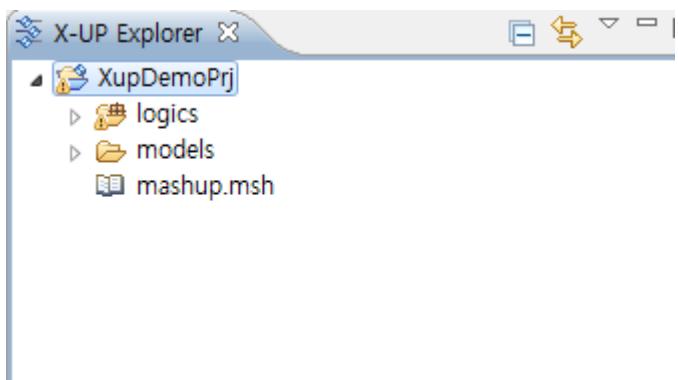
- X-UP 프로젝트 생성하기
- Automation 모델 생성 및 procedure Invoke 생성하기
- procedure Invoke 테스트하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭 합니다.

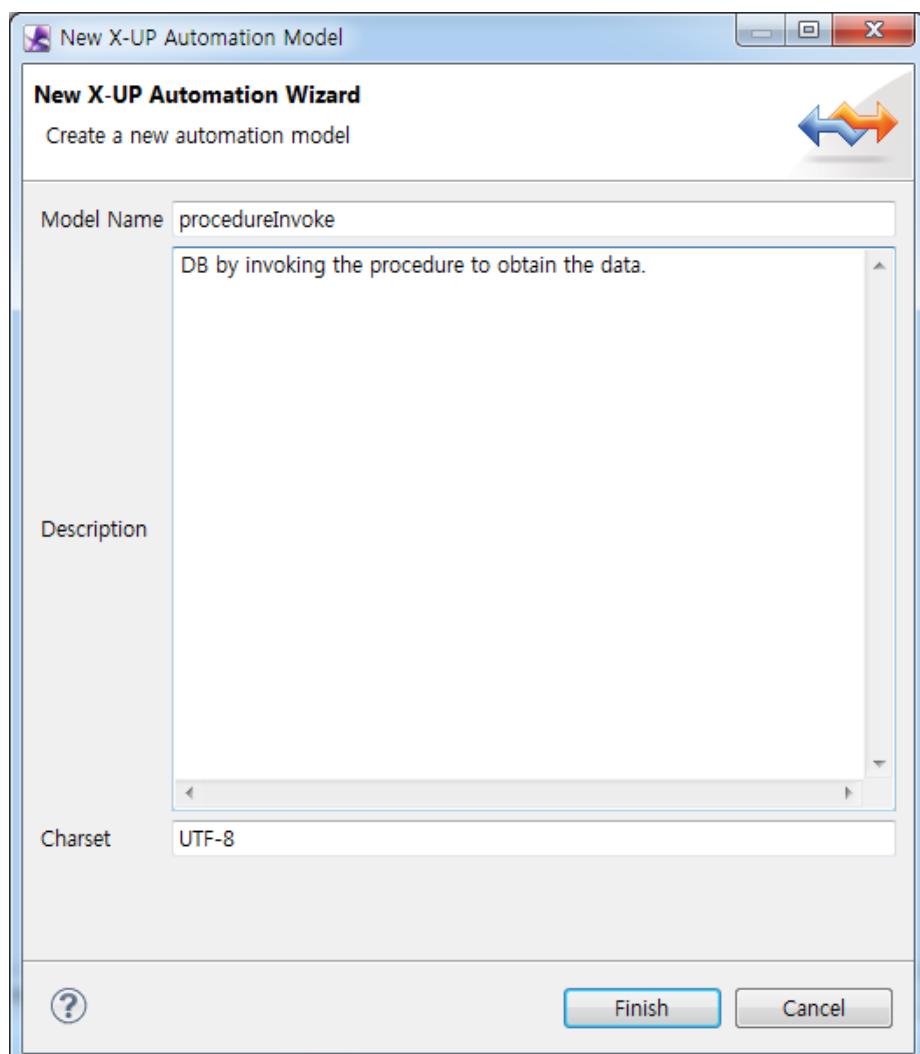


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

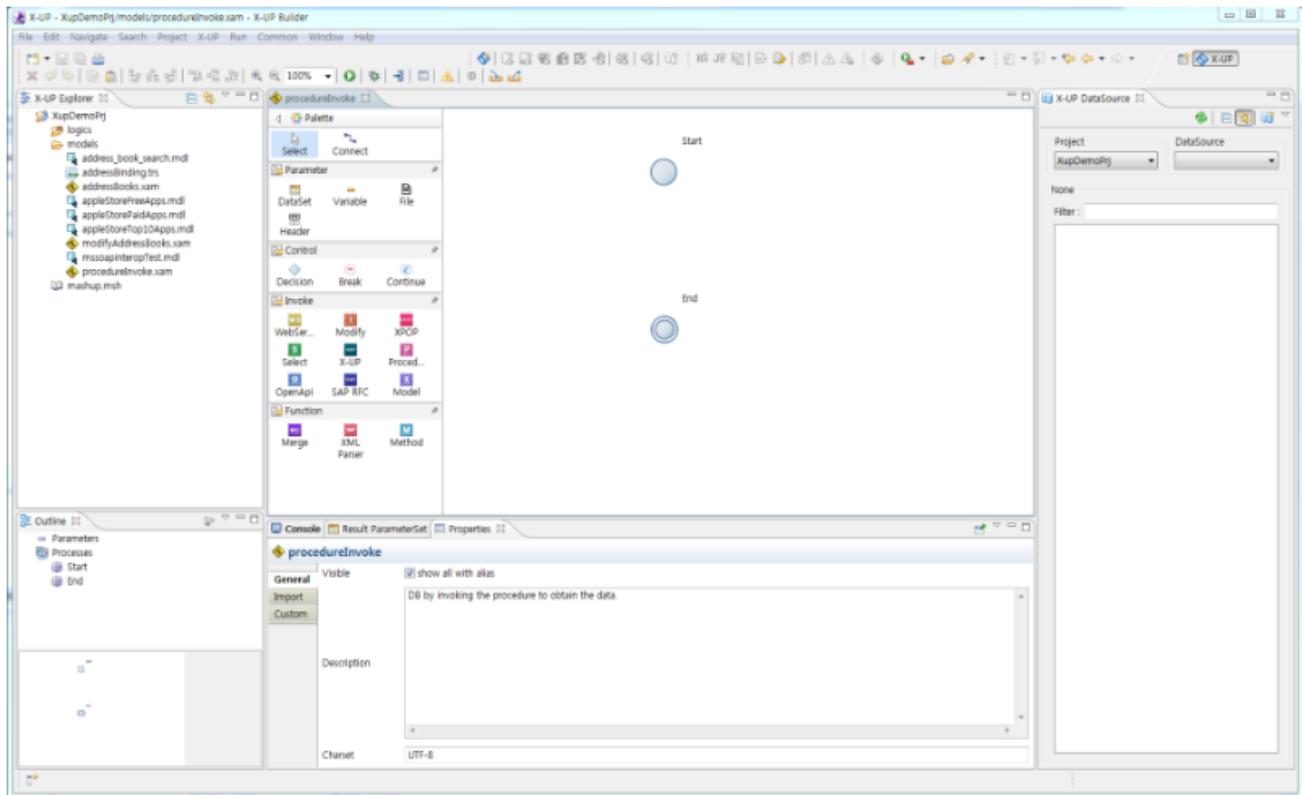


Automation 모델 생성 및 Select Invoke 생성하기

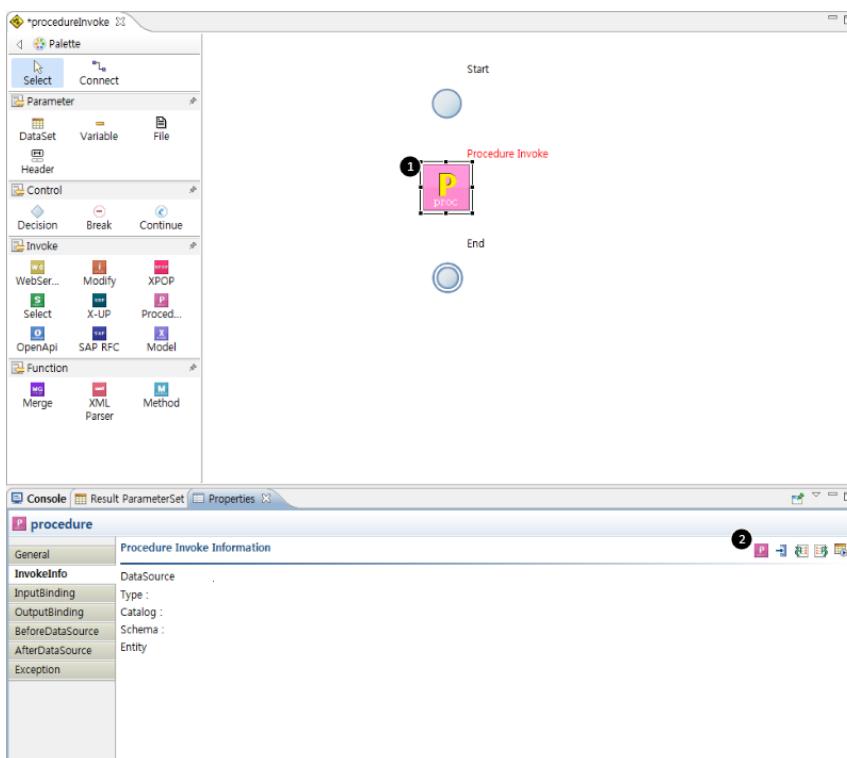
- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 원하는 모델 이름을 입력하고 OK 버튼을 클릭합니다.



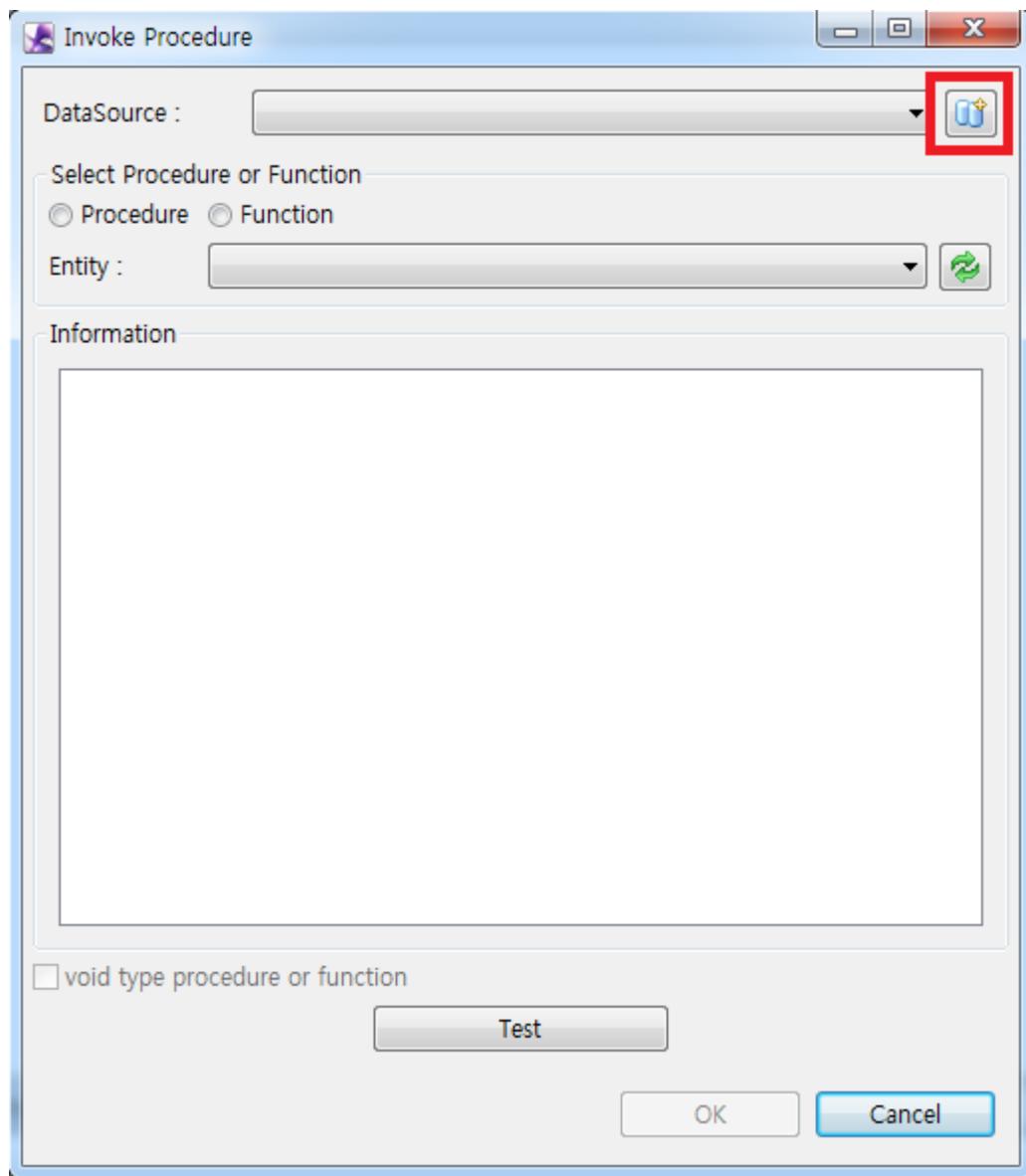
3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



4. Palette 의 Invoke 중 ‘Procedure’를 선택하고 에디터를 클릭 하게 되면 Procedure Invoke가 생성 됩니다.
5. 생성 된 Procedure Invoke를 (1) 더블클릭 하거나 ‘properties’ 탭을 클릭하여 ‘Procedure icon’[P](2) 을 클릭하면 위자드가 실행 됩니다.



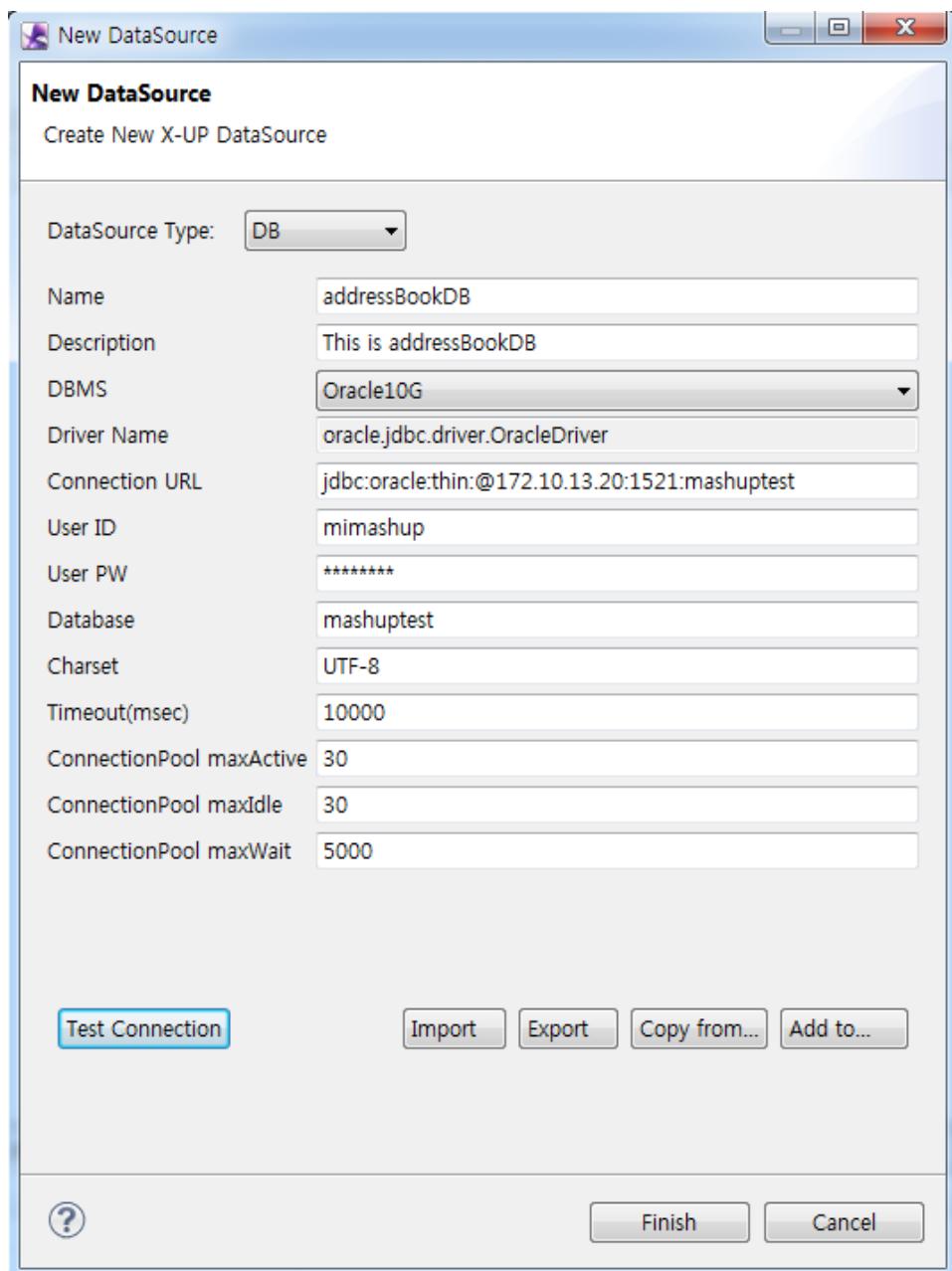
6. 위자드에서 **create new DataSource** [] 를 클릭합니다.



7. **New DataSource** 페이지에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
 8. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

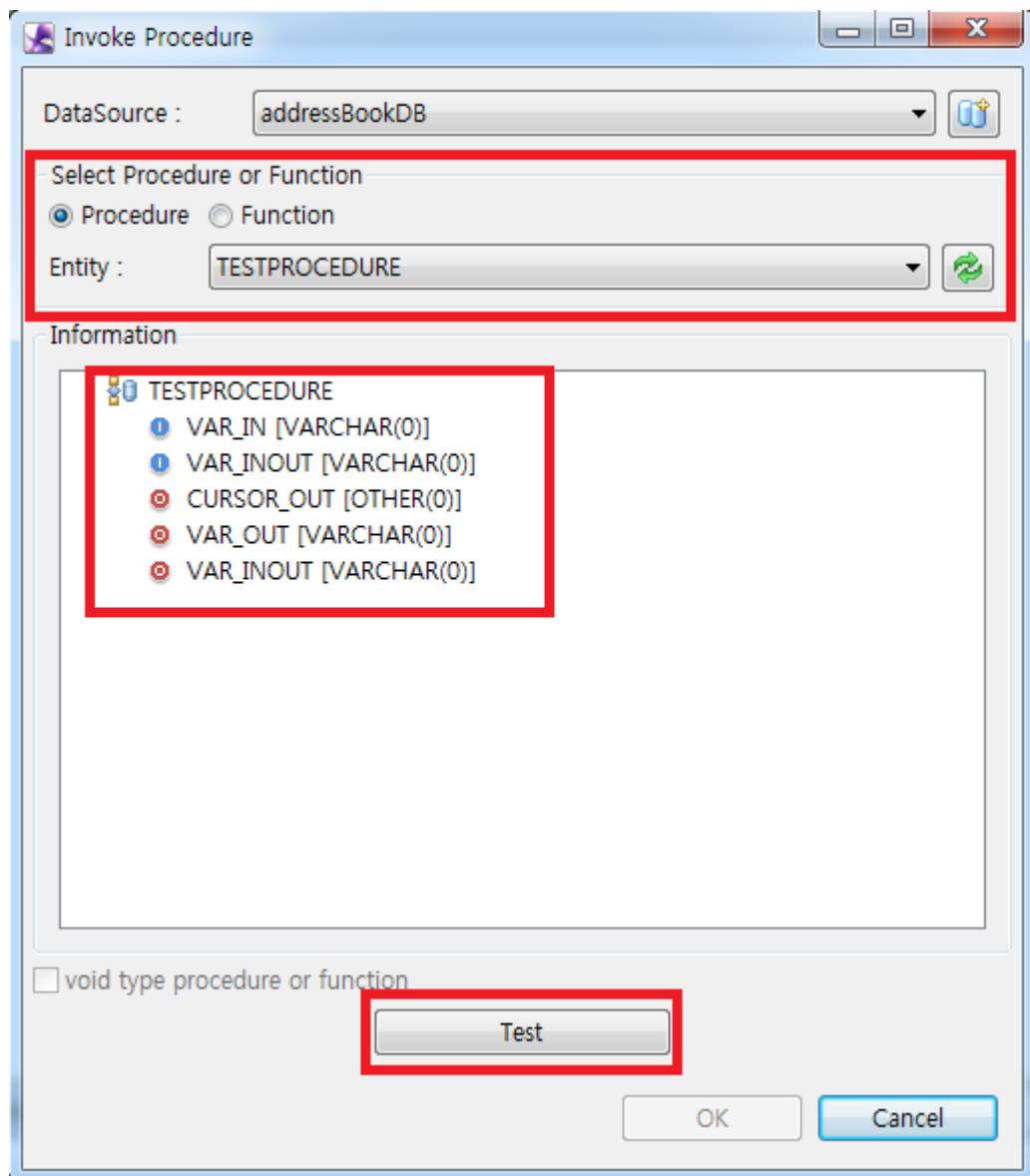
필드 명	필드 값
Name	'addressBookDB' 데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
DBMS	사용할 데이터베이스 시스템을 선택합니다. 이 예제는 'Oracle'를 사용합니다.
Connection URL	JDBC Connection String을 입력합니다.
User ID	데이터베이스 사용자 ID를 입력합니다.
User PW	데이터베이스 사용자 암호를 입력합니다.

필드 명	필드 값
Database	사용할 데이터베이스의 이름을 입력합니다.
Charset	'UTF-8' 문자 인코딩에 대한 값을 입력합니다.
Timeout(msec)	'10000' 서버와의 접속을 유지할 최대 시간
ConnectionPool MaxActive	'30' 사용하는 커넥션 최대수
ConnectionPool MaxIdle	'30' 사용 하지 않는 커넥션의 최대수
ConnectionPool maxWait	'5000' 커넥션을 열기 위해 최대 기다리는 시간

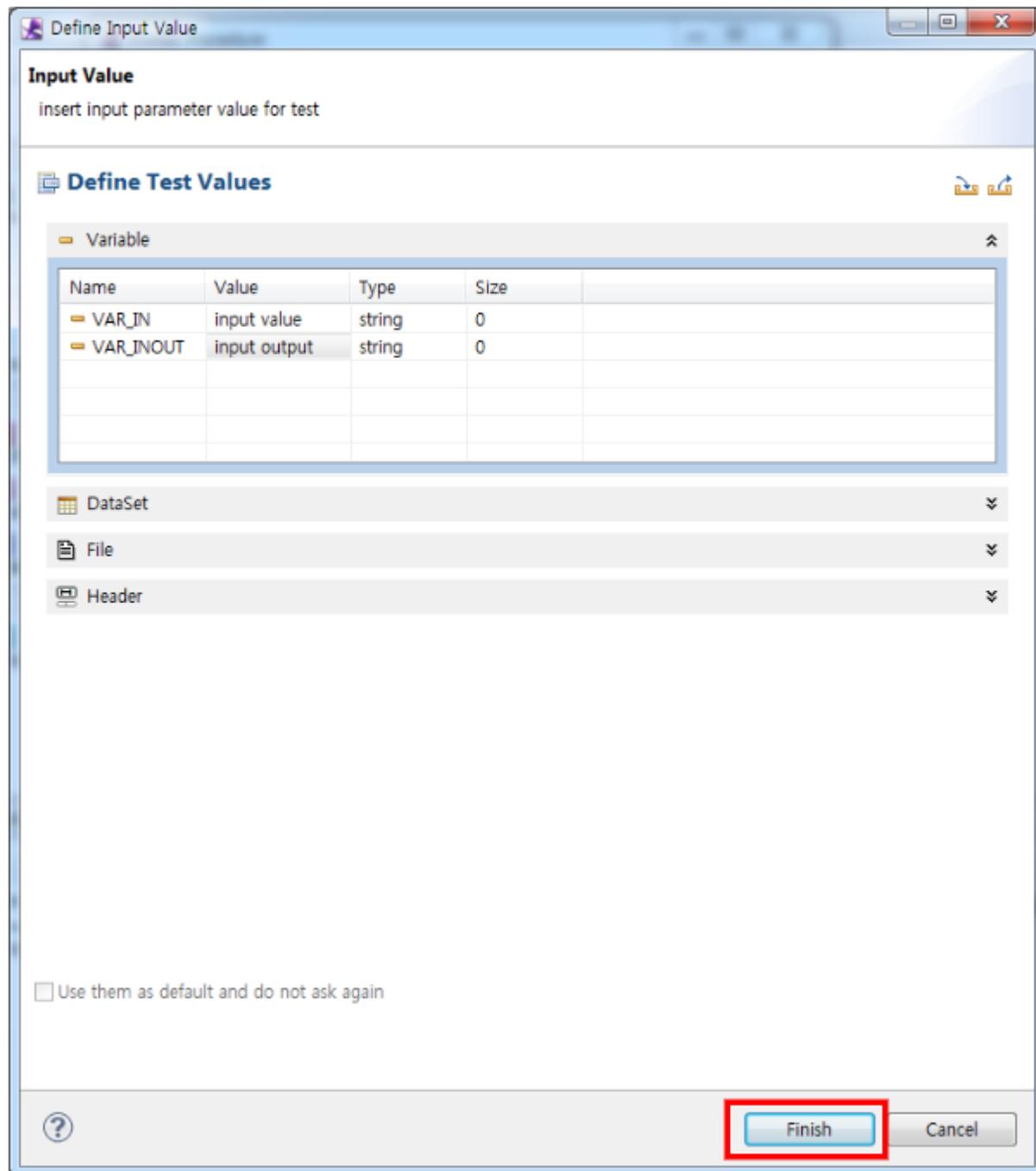


9. 정의 된 데이터 소스에서 Procedure와 Function의 리스트를 볼 수 있습니다.
10. DataSource와 Entity를 선택 합니다.
11. Information에 해당 procedure의 정의 된 입-출력 파라메터의 정보가 나타납니다. 파라메터는 입력 파라메터, 출력 파라메터 순으로 보여지게 되며 파라메터 명칭 앞 파란색 동그라미 이미지는 입력 파라메터를 나타내고, 빨간색 동그라미는 출력 파라메터를 나타냅니다.
12. 테스트 버튼을 클릭 합니다.

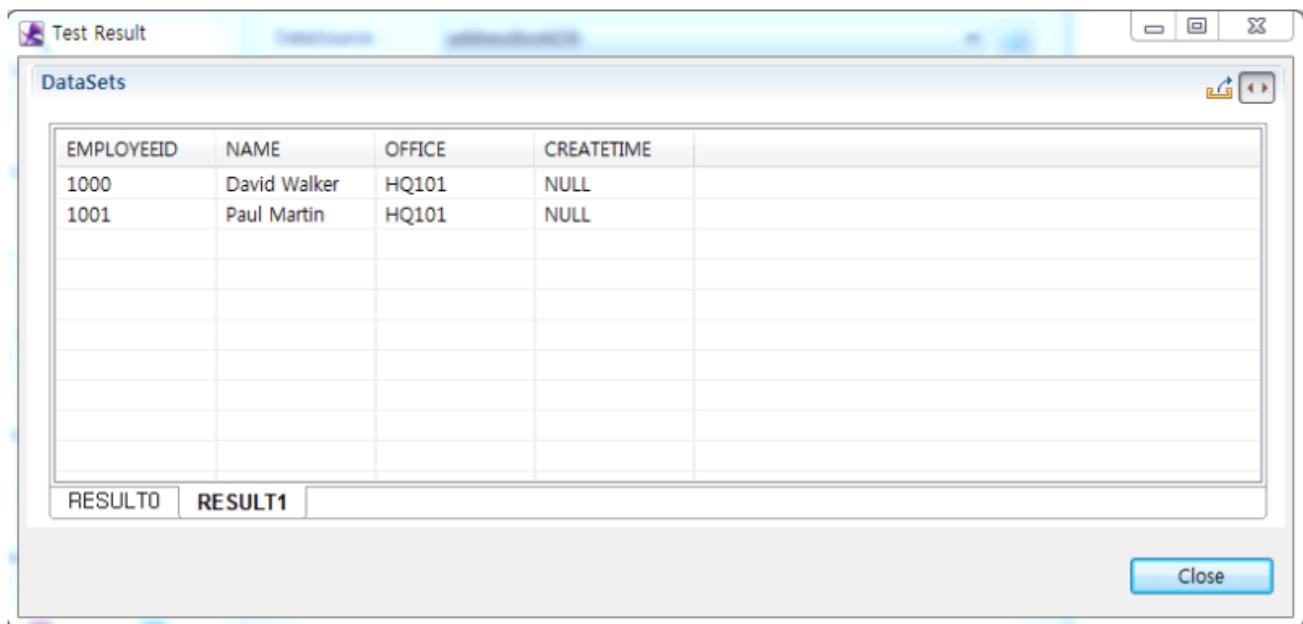
 Procedure 및 Function은 테스트를 수행하기 전 까지 확실한 출력의 값을 알 수 없습니다. (MSSQL, MySQL) 그래서 테스트를 수행하고난 뒤 출력결과를 확인 한 후에 Procedure Invoke가 생성됩니다.



13. Define Input Value에 값을 넣고 'Finish' 버튼을 클릭합니다.

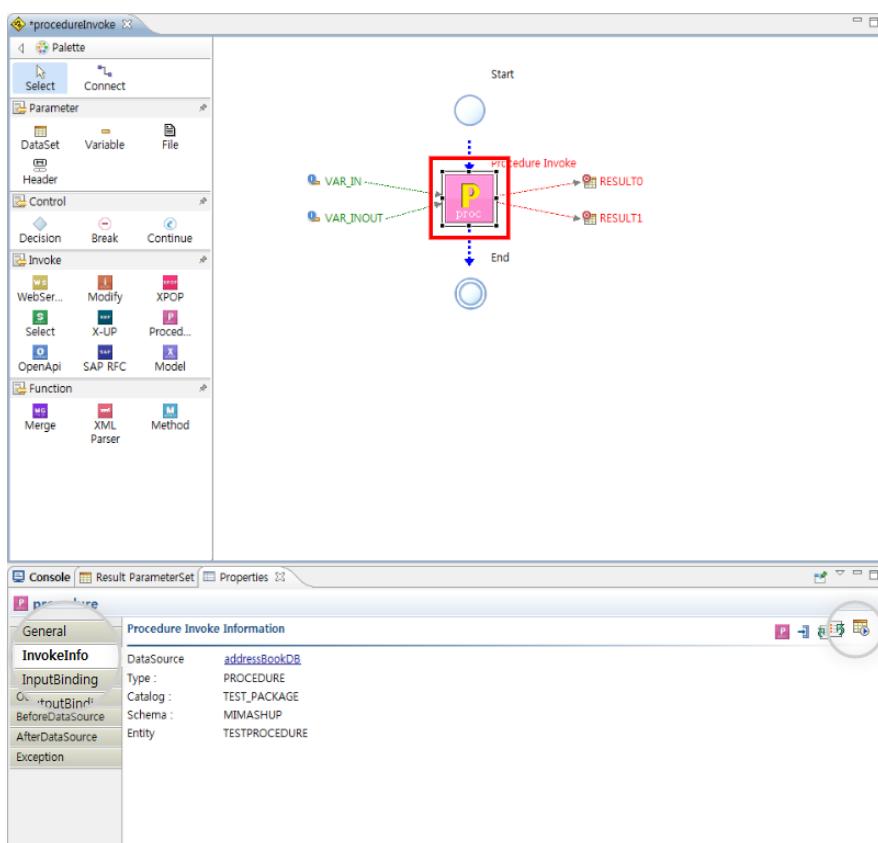


14. Procedure를 호출 했을 경우의 출력의 값을 확인 할 수 있습니다.
15. 'OK' 버튼을 클릭 하면 Procedure Invoke의 입력 파라메터와 출력 파라메터 설정을 완료 하게 됩니다.

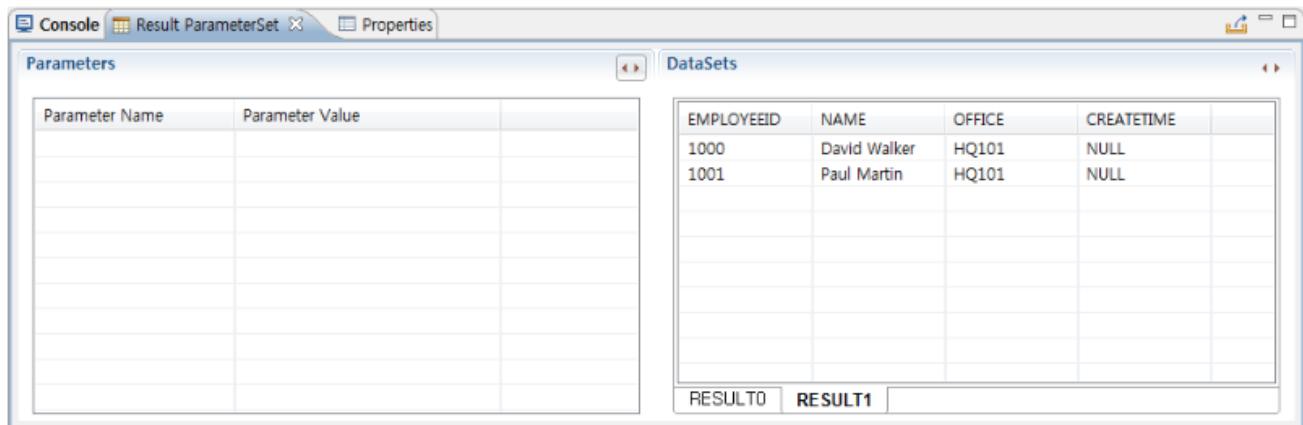


Invoke 테스트 하기

1. 생성 된 Procedure Invoke입니다. Invoke의 테스트 결과를 바탕으로 입력 파라미터와 출력 파라미터가 생성됩니다.
2. Procedure Invoke의 Properties의 'InvokeInfo'를 선택하고 [] 를 클릭하여 테스트를 진행합니다.

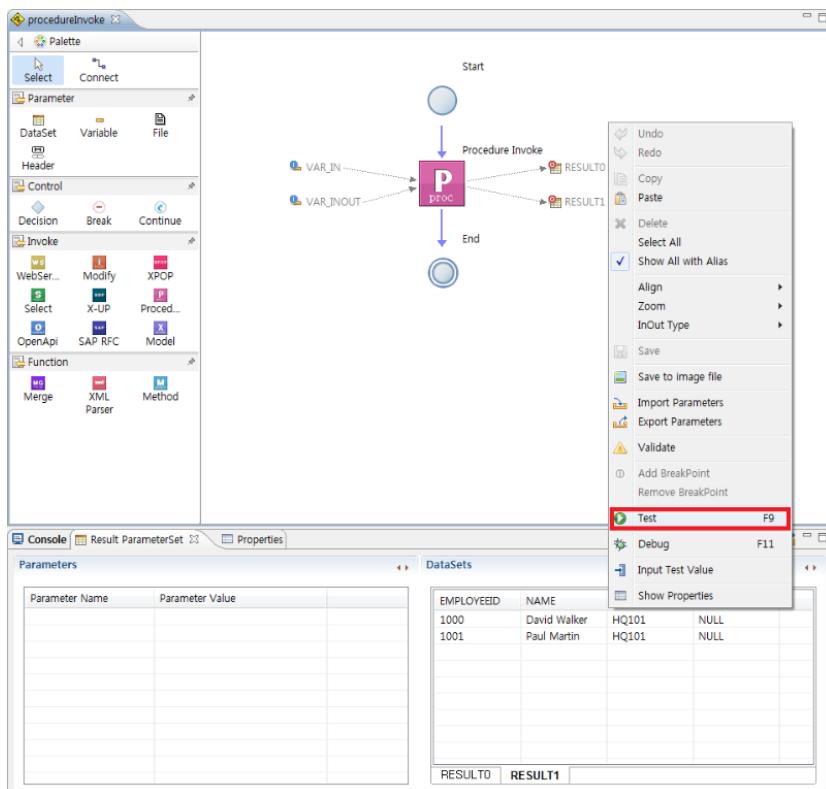


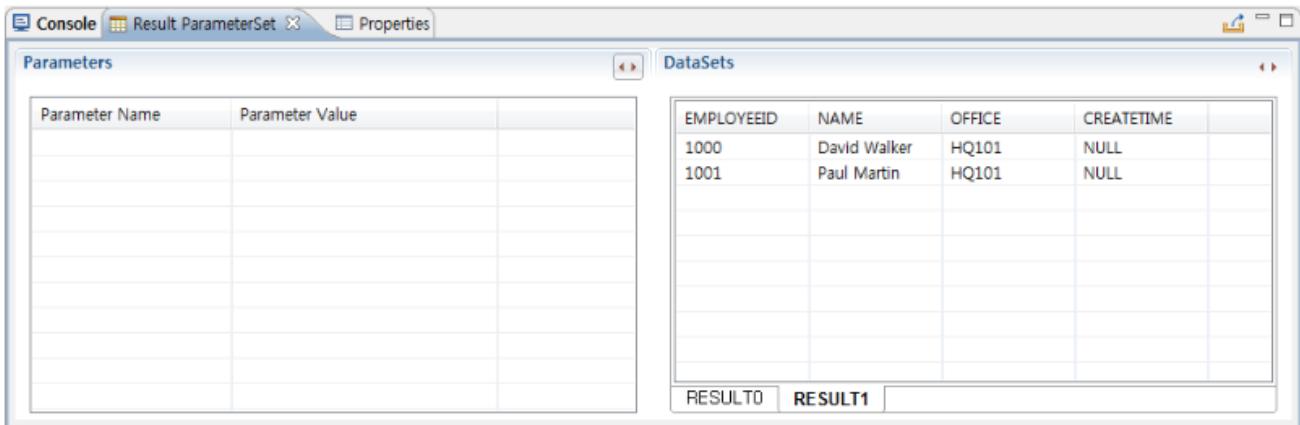
3. 'Automation 모델을 생성 및 Procedure Invoke 생성하기'에서 Procedure Invoke 생성 시 'Test' 버튼을 클릭 했을 때와 동일하게 값을 입력하고 테스트를 진행하게 되면 결과를 확인 할 수 있습니다.
4. 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인 할 수 있습니다.



모델 테스트하기

1. 모델 에디터에서 Palette의 Connect를 선택하여 Start 노드와 Procedure Invoke, End 노드를 연결 해 줍니다.
2. 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.
3. 테스트 데이터는 'Procedure Invoke 테스트하기' 와 동일한 테스트 데이터를 입력하여 테스트를 합니다.
4. 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.





8.5 OpenApi Invoke를 이용한 모델 개발하기

이 절에서는 Apple사의 Apple store 서비스를 이용한 OpenApi모델 개발방법을 설명합니다.

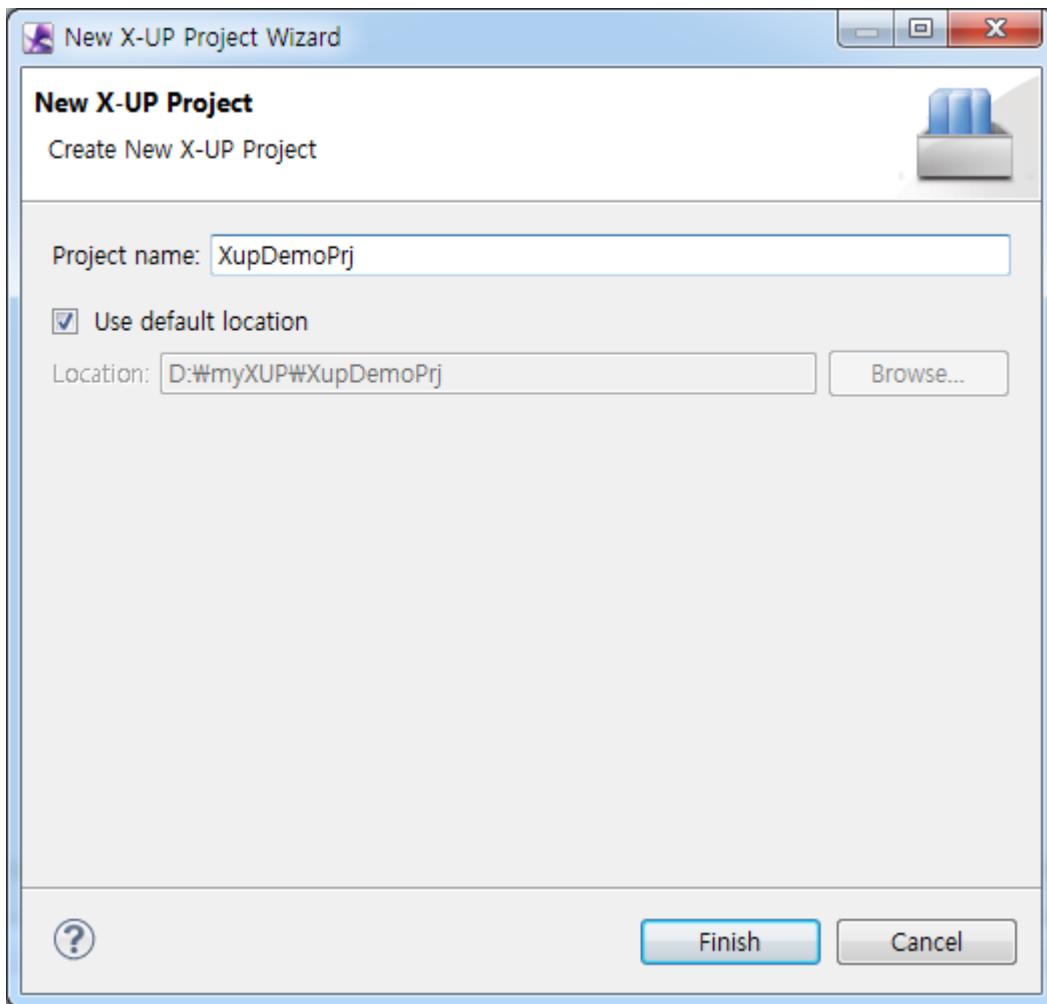
Apple store에는 각 카테고리 별 신작자료 및 인기순위 리스트를 제공합니다. 데이터는 XML기반의 RSS 형태로 제공됩니다. X-UP Builder로 XML형태의 데이터를 수집하여 정형화된 파라메터를 생성 합니다.

이 절에서 설명하는 Automation 모델의 OpenApi Invoke 개발단계는 다음과 같습니다.

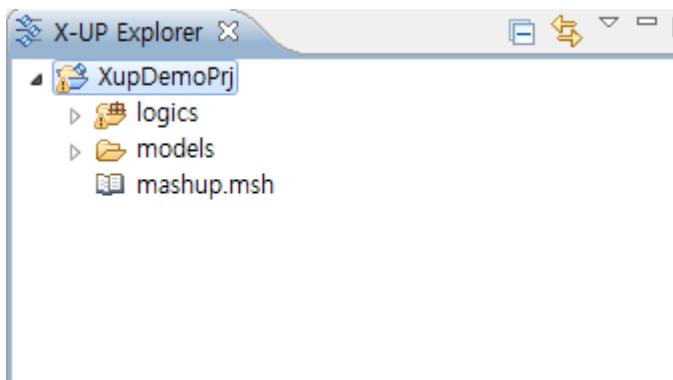
- X-UP 프로젝트 생성하기
- Automation 모델 생성 및 OpenApi Invoke 생성 하기
- 데이터 소스 생성 및 Properties 설정 하기
- OpenApi Invoke 테스트 하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭 합니다.

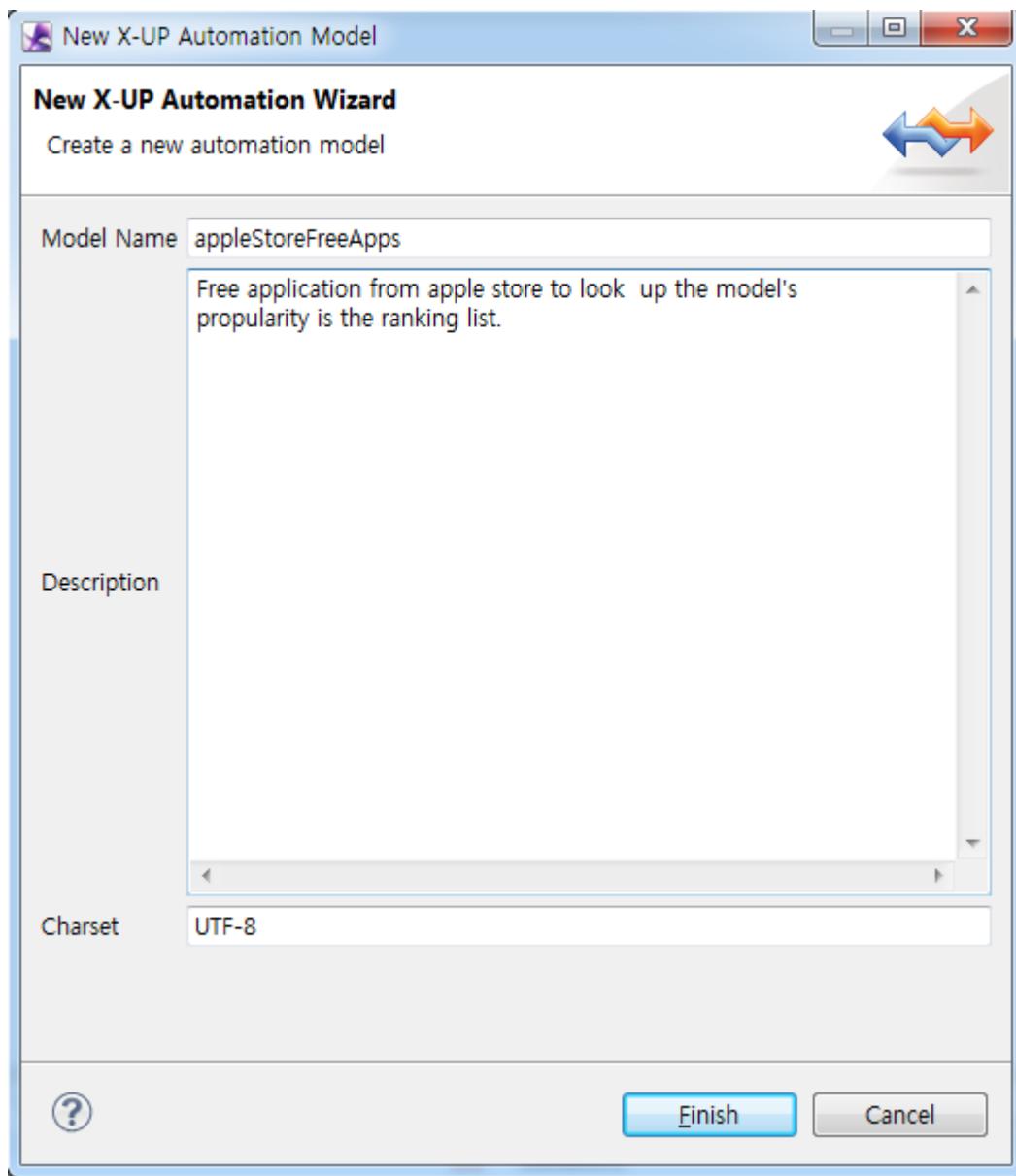


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

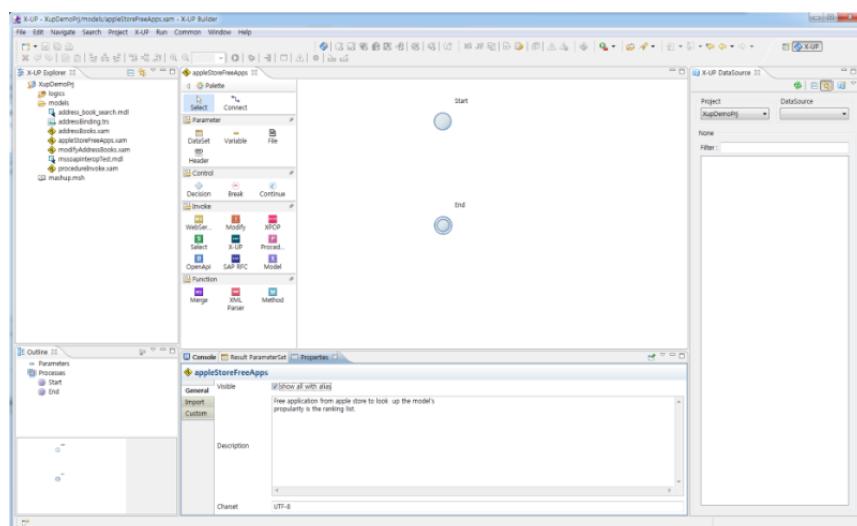


Automation 모델 생성 및 OpenAPI Invoke 생성하기

1. [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 원하는 모델 이름을 입력하고 OK 버튼을 클릭합니다.



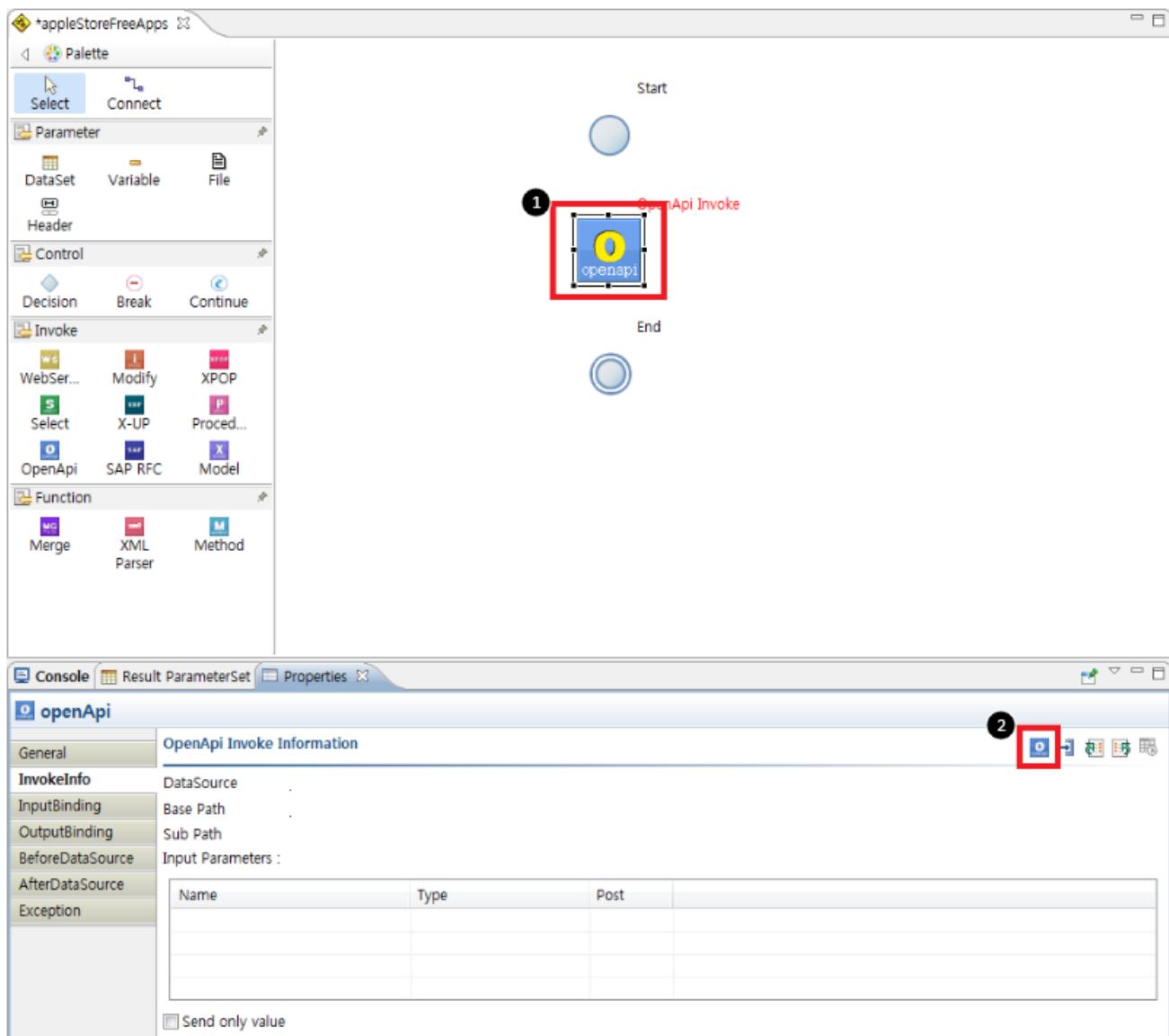
3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



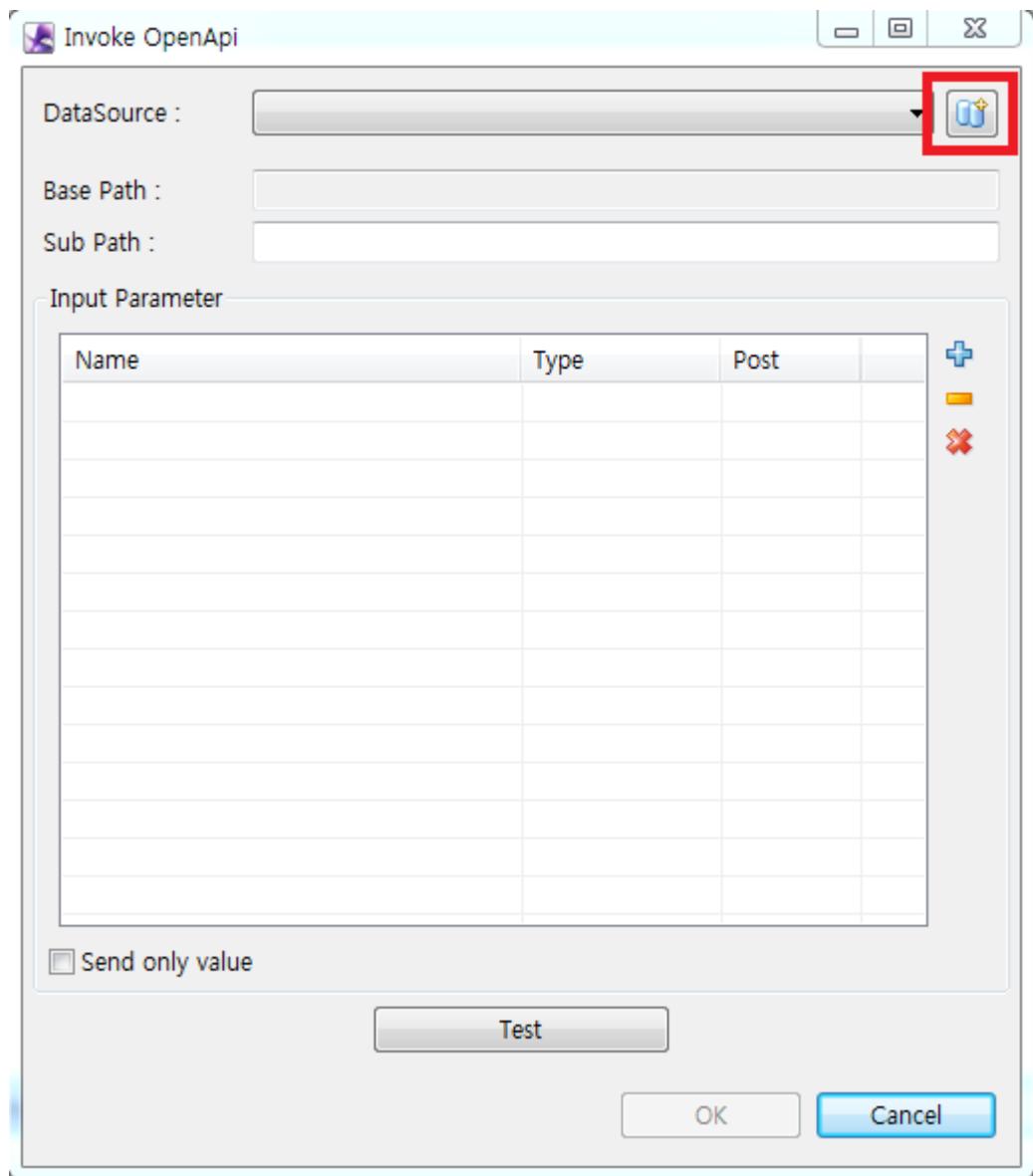
4. Palette 의 Invoke 중 ‘OpenApi’ 를 선택하고 에디터를 클릭 하게 되면 OpenApi Invoke가 생성 됩니다.

데이터소스 생성 및 Properties 설정하기

1. 생성 된 OpenApi Invoke를(1) 더블클릭하거나 ‘properties’ 탭을 클릭하여 ‘OpenApi icon’[](2) 을 클릭하면 위자드가 실행 됩니다.



2. 위자드에서 create new DataSource [] 를 클릭합니다.

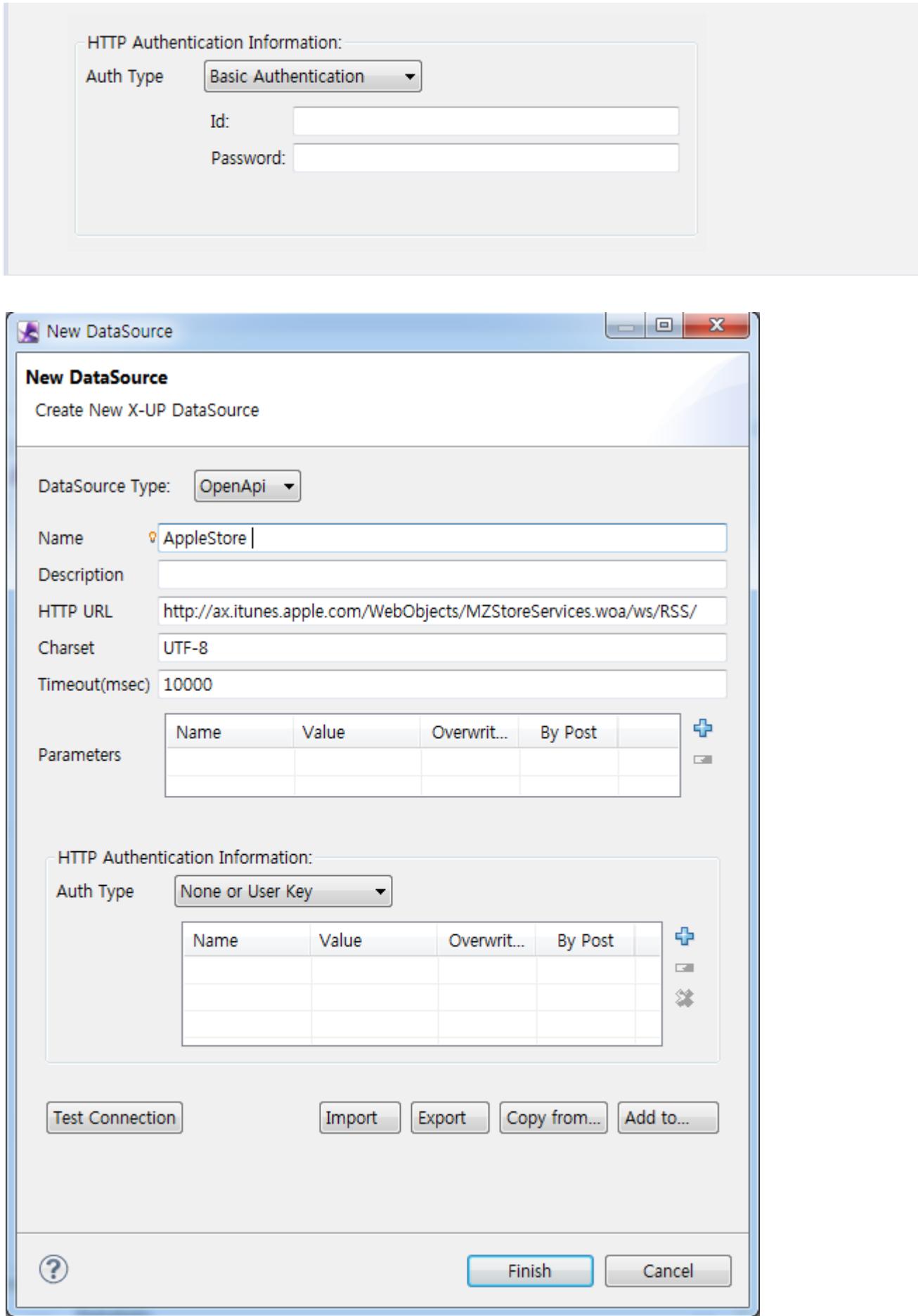


3. **New DataSource** 페이지에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
4. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

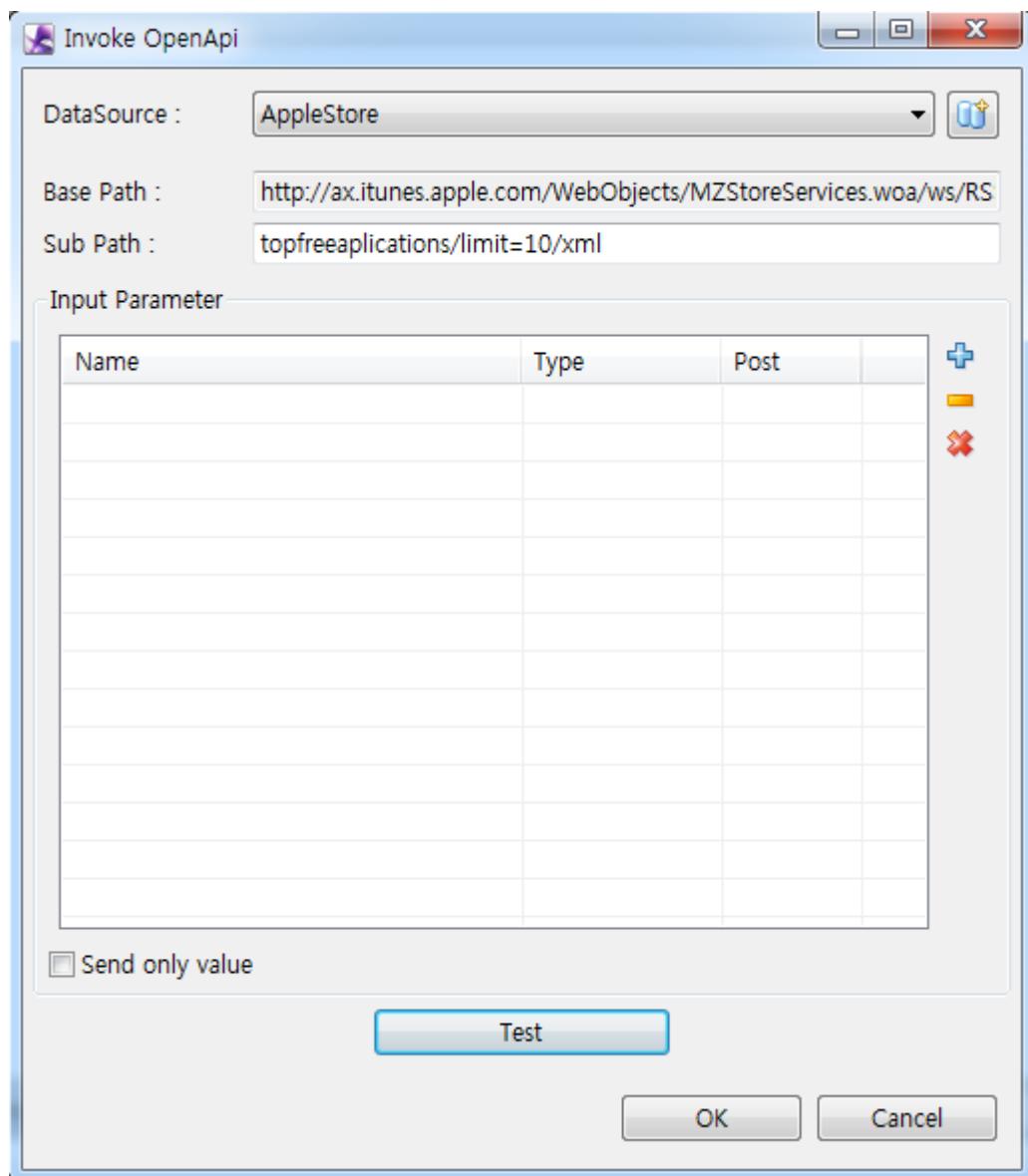
Field Name	Field Value
Name	'AppleStore' 데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
HTTP URL	'https://itunes.apple.com/us/rss/' OpenAPI 서비스에 대한 Base URL를 입력합니다.

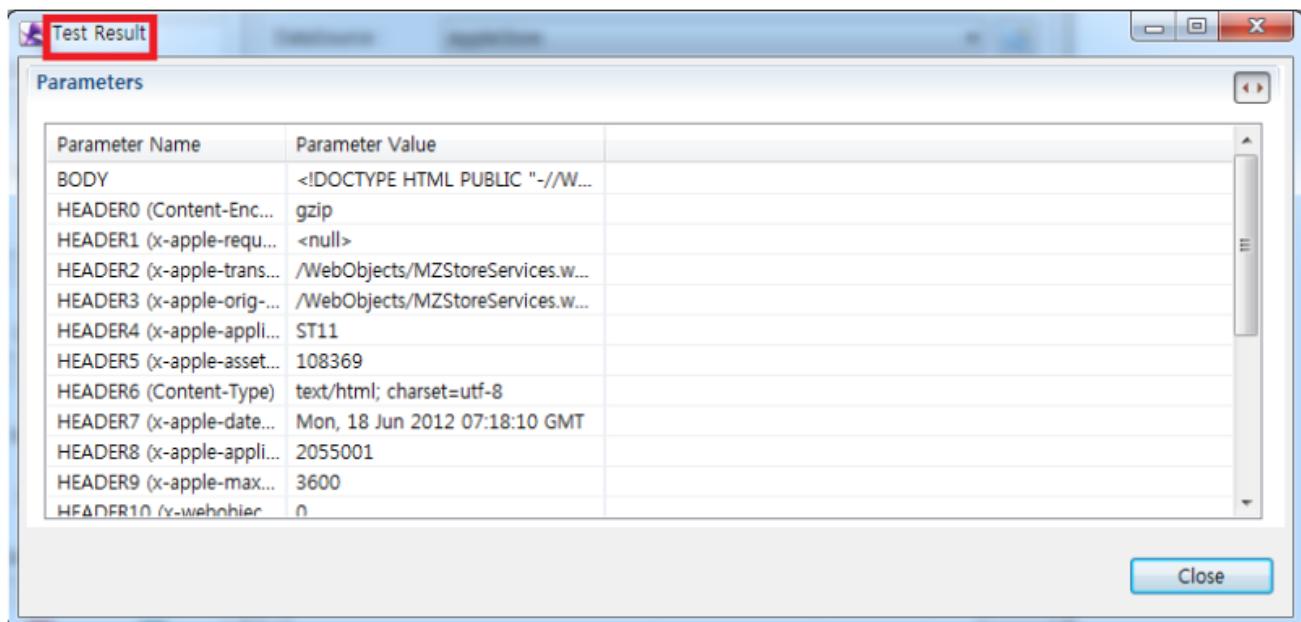


OpenApi를 제공하는 호스트가 HTTP Basic 인증을 요구하는 경우에는 HTTP auth Info의 'Id'와 'Password'에 유효한 값을 입력합니다.



5. 정의 된 데이터 소스에서 AppleStore를 선택 합니다.
6. Base Path에 적합한 Sub Path를 입력 하고 테스트 버튼을 클릭 합니다.
예제의 Sub Path는 'topfreeapplications/limit=10/xml' 입니다.
7. 서비스가 정상 적으로 호출 되어졌을 경우 아래와 같이 'BODY' 와 실제 서비스 요청 시의 HTTP Header 응답 값을 확인 할 수 있습니다.





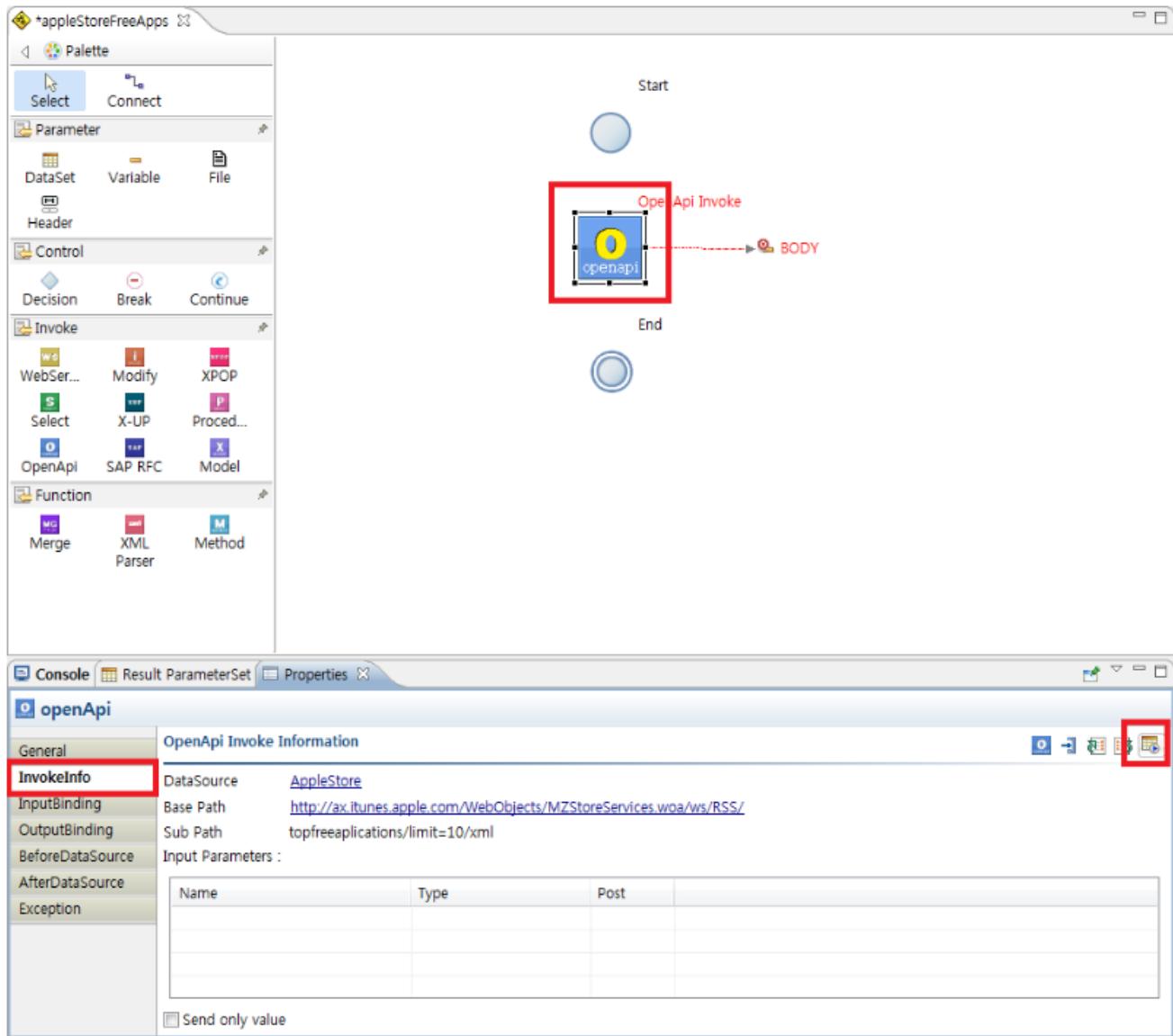
8. OK 버튼을 클릭 할 경우 테스트 시 확인 한 값이 출력 파라메터로 설정 된 것을 알 수 있습니다.

불 필요한 HTTP Header의 정보는 지워도 정상 작동 합니다

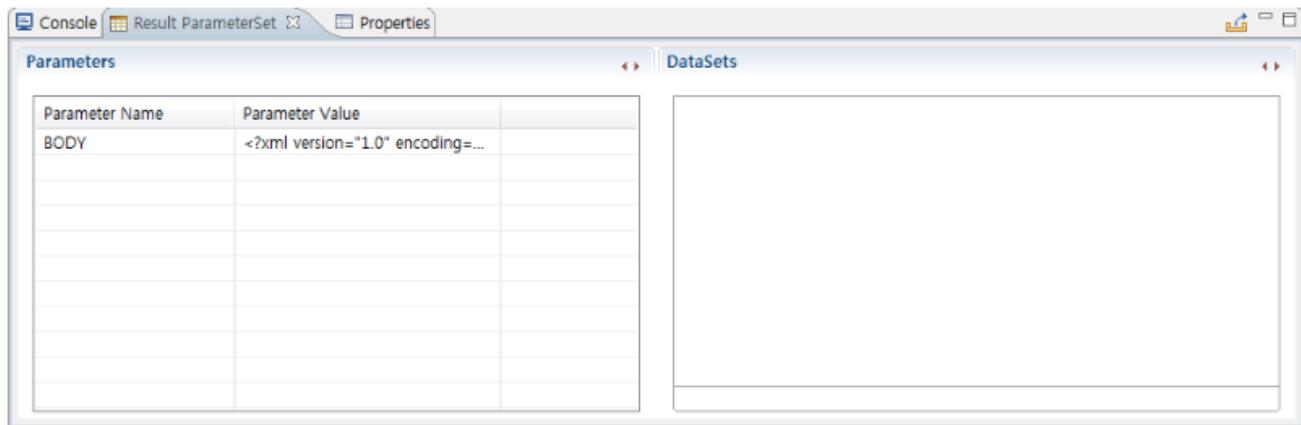
The screenshot shows the X-UP development environment with the project 'appleStoreFreeApps' open. On the left is the palette with various components like Select, Connect, Parameter, Header, Control, Invoke (including WebService, Modify, X-POP, Select, X-UP, Procedure, OpenApi, SAP RFC, Model), and Function (Merge, XML Parser, Method). In the center, there's a process diagram starting with a 'Start' node, followed by an 'OpenApi Invoke' node labeled 'openapi'. Dotted arrows point from this node to 16 'HEADER' nodes labeled HEADER0 through HEADER15. After these headers, there's an 'End' node. At the bottom, the 'Properties' tab is selected for the 'openApi' component, showing 'OpenApi Invoke Information' with fields: DataSource (AppleStore), Base Path (<http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/>), Sub Path (topfreeapplications/limit=10/xml), and Input Parameters (a table with one row: Name, Type, Post). A checkbox 'Send only value' is also present.

Invoke 테스트 하기

- 생성 된 OpenApi Invoke입니다. Invoke의 테스트 결과를 바탕으로 입력 파라미터와 출력 파라미터가 생성됩니다.
- 사용하지 않는 HTTP Header 응답 값을 지웁니다.
- OpenApi Invoke의 Properties의 'InvokeInfo'를 선택하고 [] 를 클릭하여 테스트를 진행합니다.

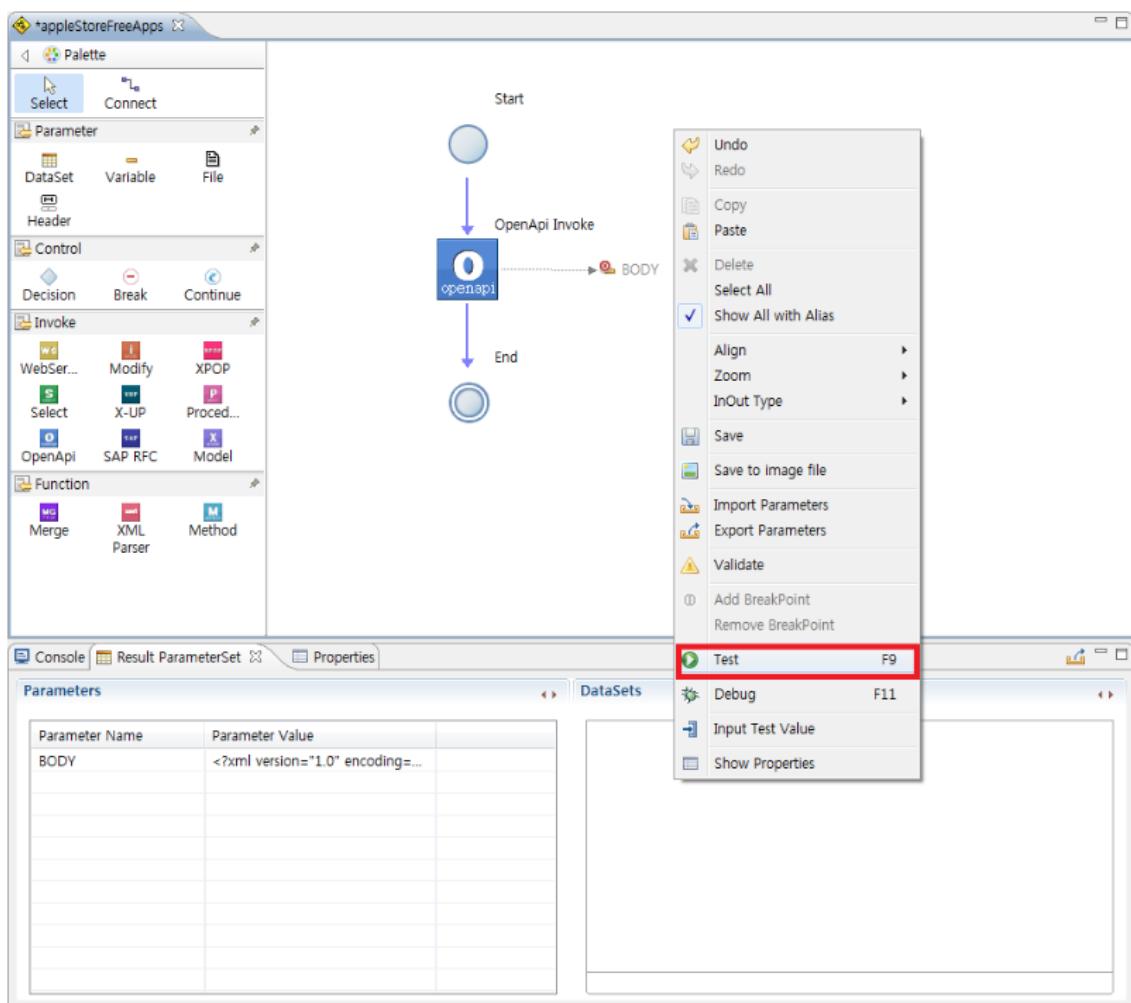


- 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인 할 수 있습니다. (본 예제와 같이 입력 파라미터가 없을 경우는 바로 출력 결과가 나타납니다.)



모델 테스트하기

- 모델 에디터에서 Palette의 Connect를 선택하여 Start 노드와 OpenApi Invoke, end 노드를 연결 해 줍니다.
- 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.
- 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.



8.6 WebService Invoke를 이용한 모델 개발하기

이 절에서는 Microsoft사의 web service testing system을 이용한 모델 개발방법을 설명합니다.

web service testing system는 사용자가 요청한 검색조건에 따라 SOAP XML형태의 검색목록을 반환합니다. X-UP 모델은 이 XML형태의 검색목록을 수집하여 정형화된 데이터셋 구조로 변환할 수 있습니다. 이 절에서 사용된 web service testing system의 WSDL 주소는 아래와 같습니다.

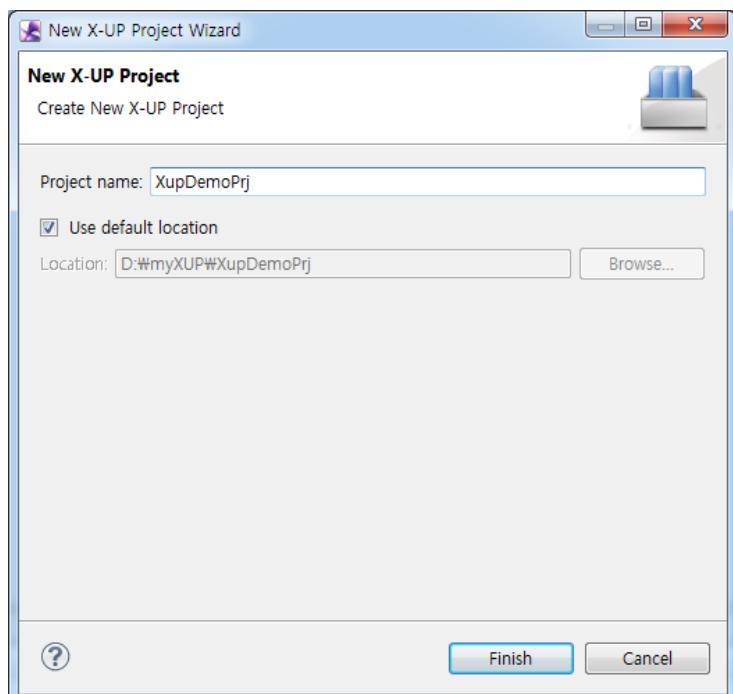
http://mssoapinterop.org/asmx/xsd/round4XSD.wsdl

이 절에서 설명하는 Automation 모델의 OpenApi Invoke 개발단계는 다음과 같습니다.

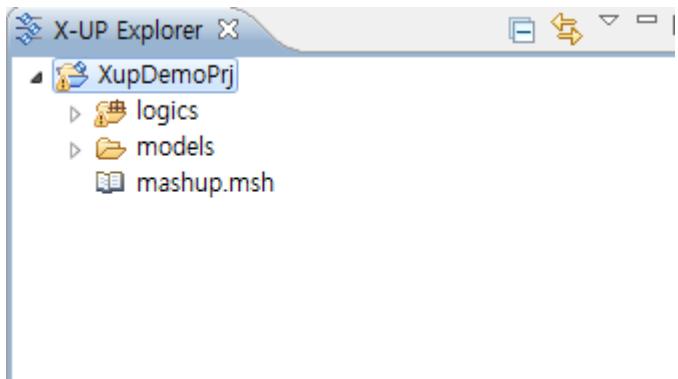
- X-UP 프로젝트 생성하기
- Automation 모델 생성 및 WebService Invoke 생성 하기
- 데이터 소스 생성 및 Properties 설정 하기
- WebService Invoke 테스트 하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭 합니다.

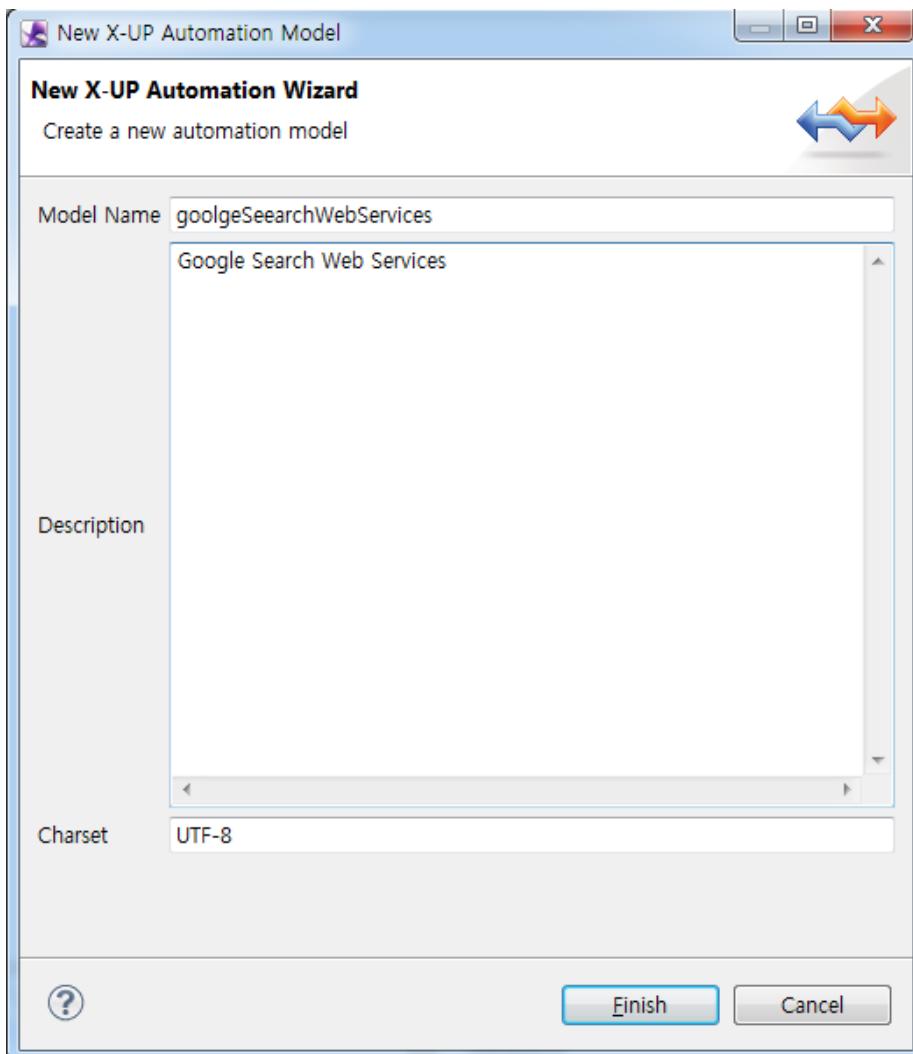


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

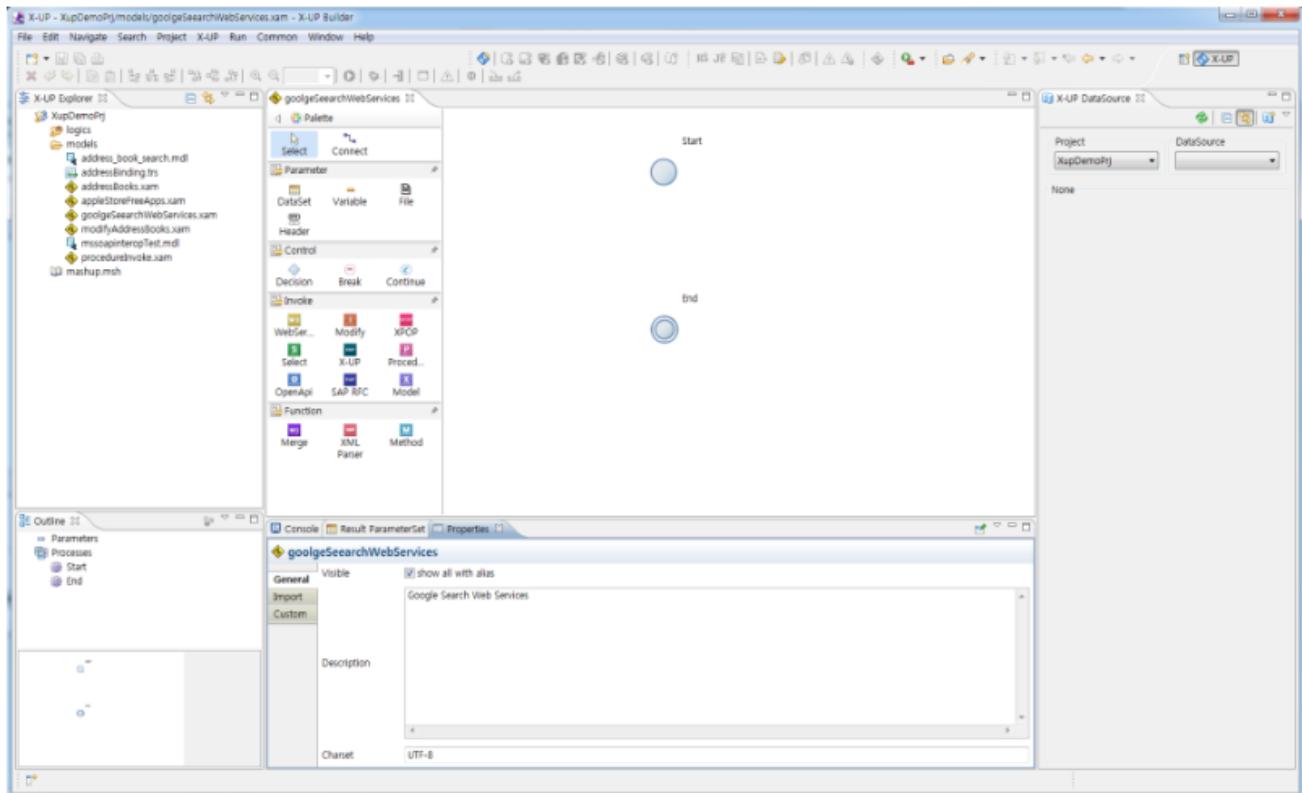


Automation 모델 생성 및 WebService Invoke 생성하기

1. [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 원하는 모델 이름을 입력하고 OK 버튼을 클릭합니다.



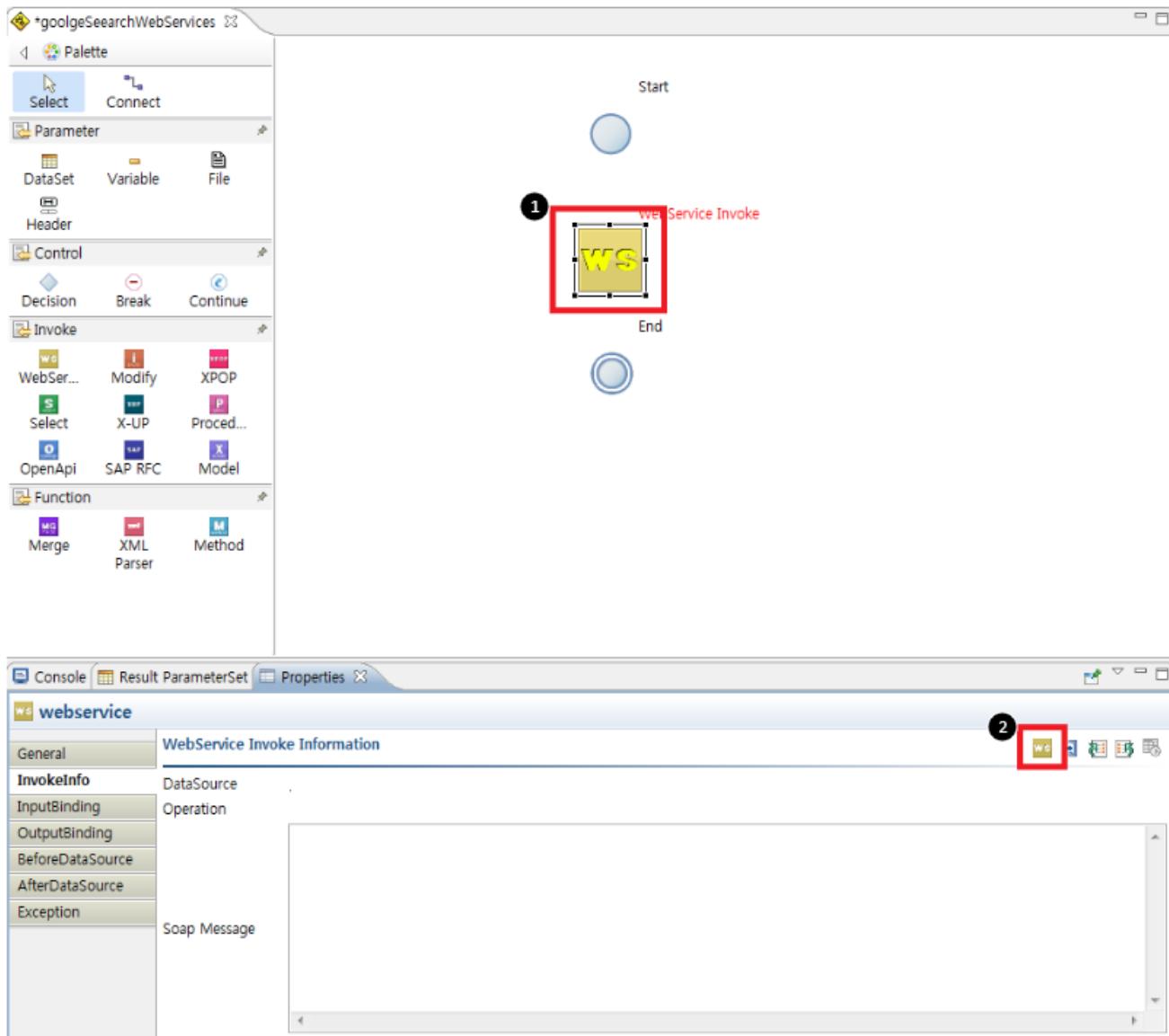
3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



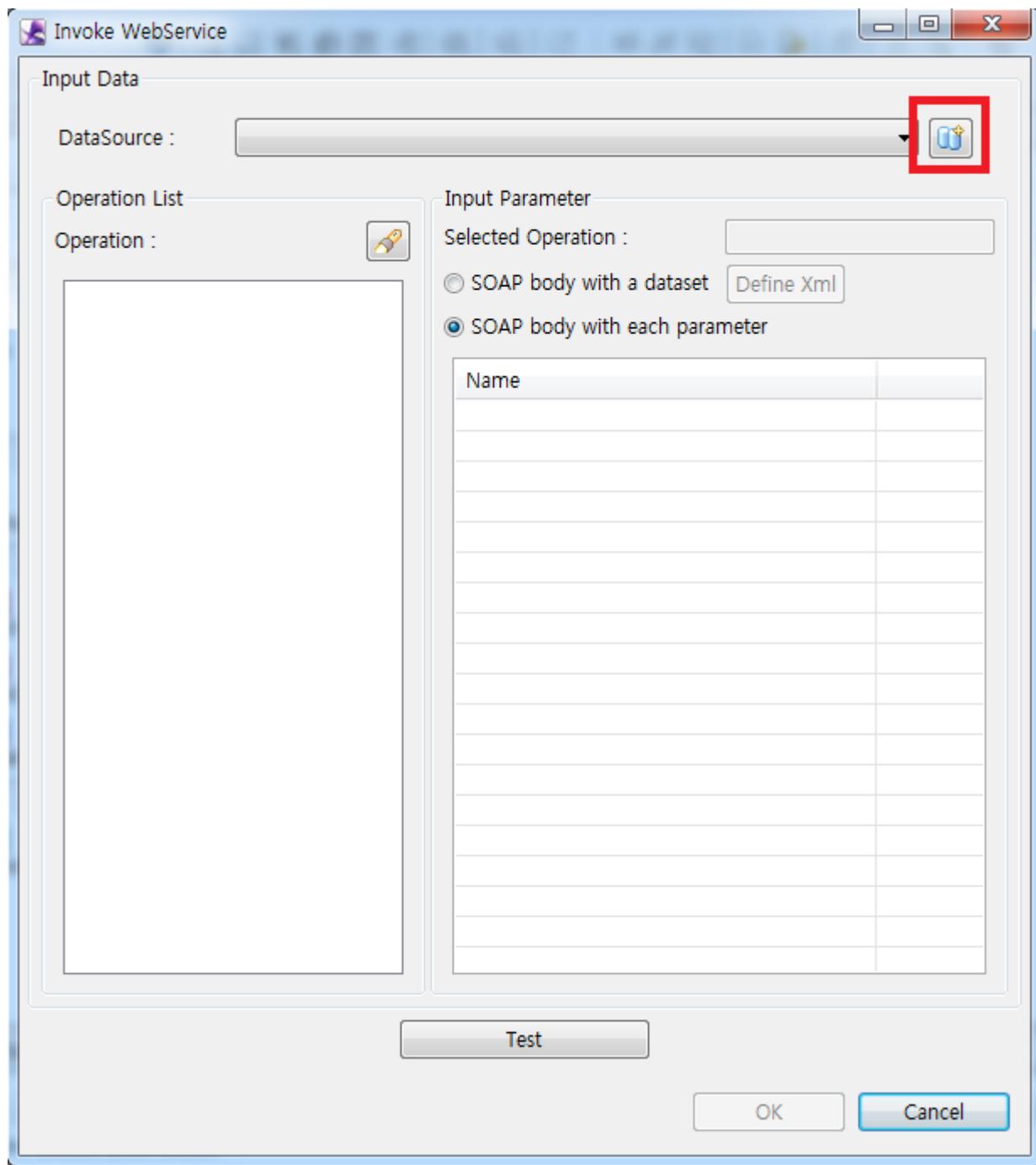
4. Palette의 Invoke 중 'WebService' 를 선택하고 에디터를 클릭 하게 되면 WebService Invoke가 생성 됩니다.

데이터소스 생성 및 Properties 설정하기

1. 생성 된 WebService Invoke를 (1) 더블클릭 하거나 'Properties' 탭을 클릭하여 'WebService icon'[] (2) 을 클릭하면 위자드가 실행 됩니다.

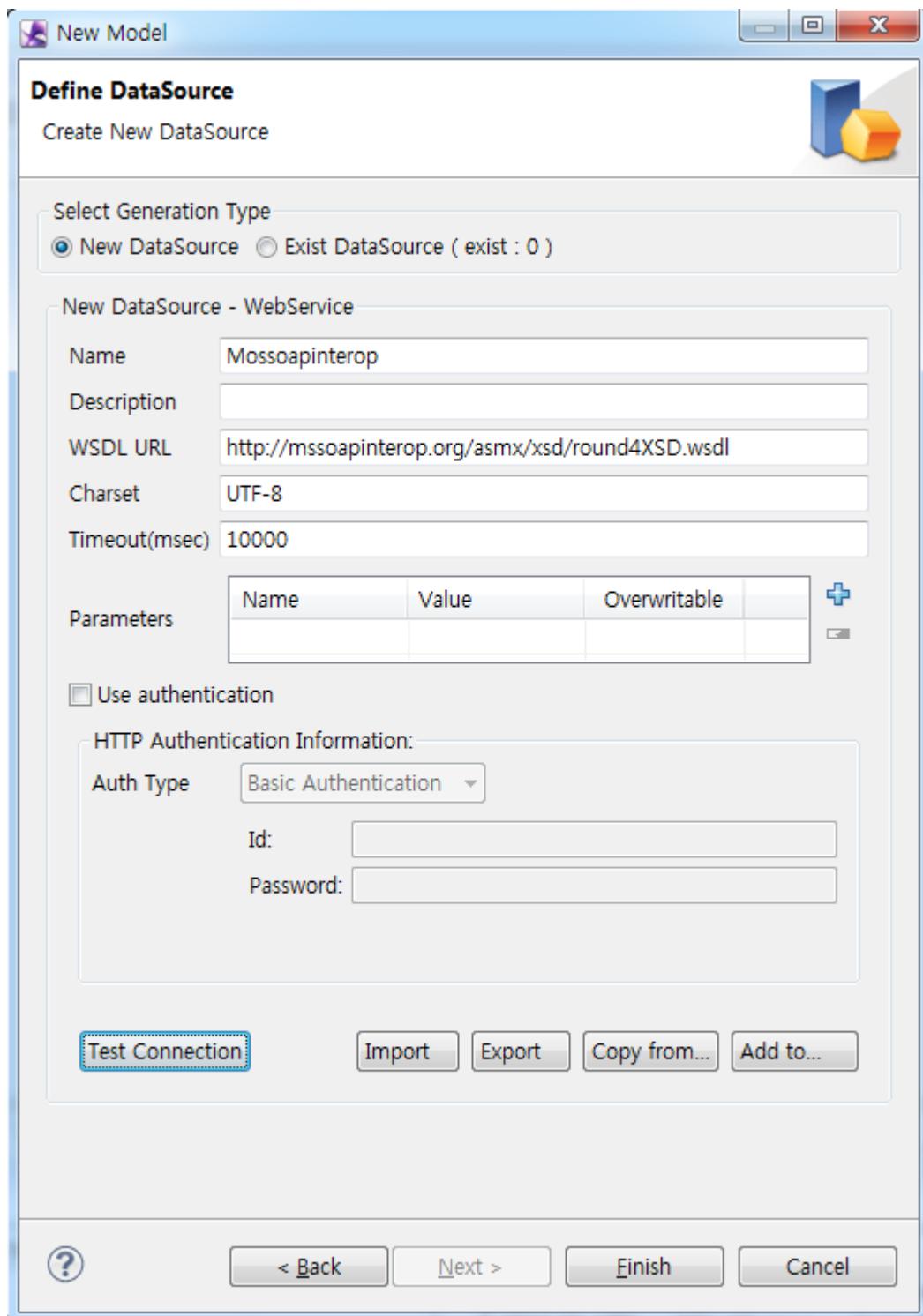


2. Invoke WebService 위자드에서 create new DataSource [] 를 클릭합니다.



3. **New DataSource** 페이지에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
4. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

Field Name	Field Value
Name	'Mssoapinterop' 데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
WSDL URL	'http://mssoapinterop.org/asmx/xsd/round4XSD.wsdl' 웹 서비스의 WSDL을 얻을 수 있는 URL를 입력합니다.

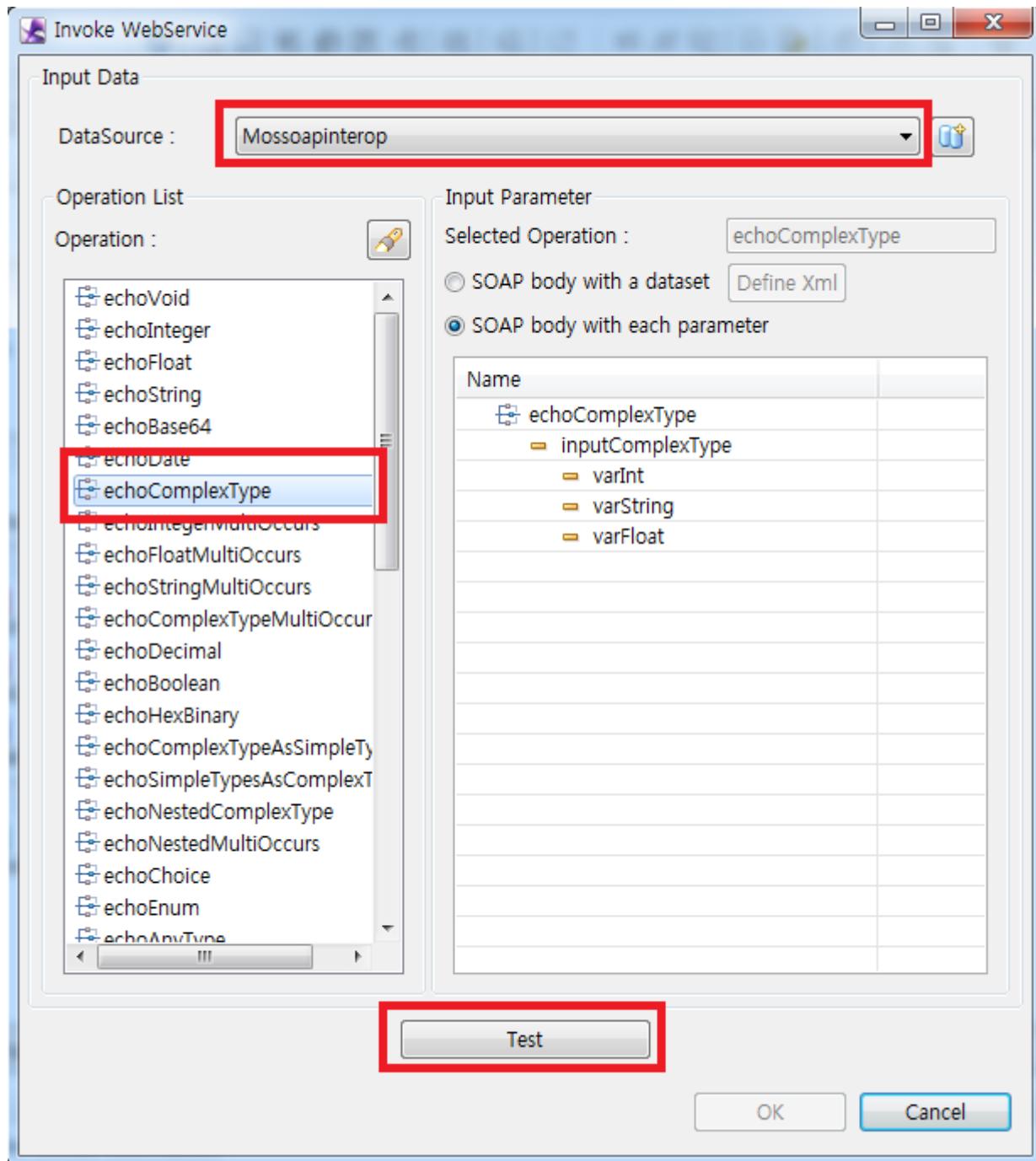


웹 서비스를 제공하는 호스트가 HTTP Basic인증을 요구하는 경우에는 HTTP Authentication Information의 'Id'와 'Password'에 유효한 값을 입력합니다.

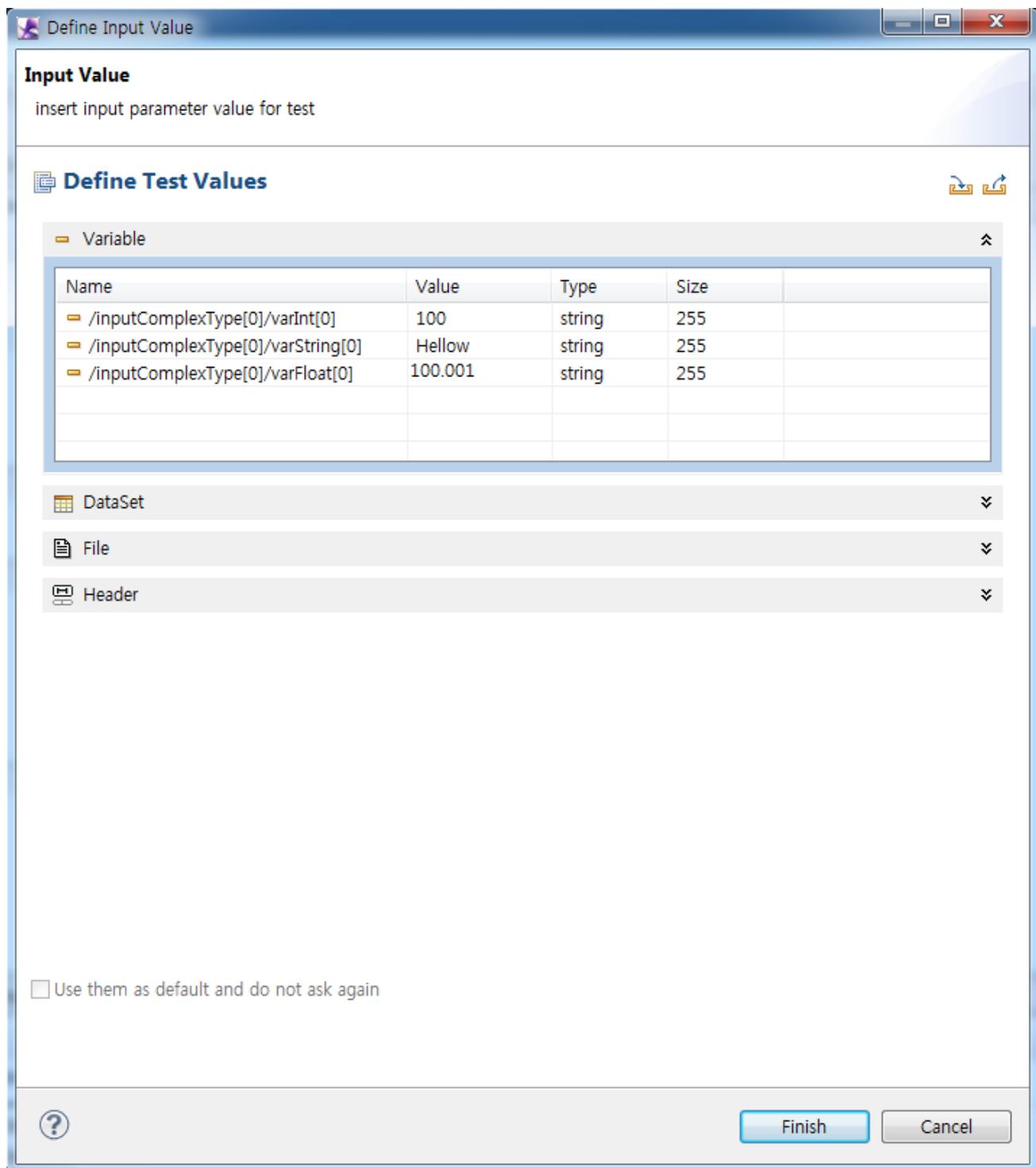
HTTP Authentication Information:

Auth Type	Basic Authentication
Id:	<input type="text"/>
Password:	<input type="password"/>

5. DataSource를 Mssoapinterop 을 선택 하면 자동으로 **오퍼레이션을 검색** 합니다. 또는 **get Operation List** 버튼 [] 을 클릭 하면 오퍼레이션을 검색 합니다.
6. 검색 된 오퍼레이션 중 'echoComplexType'을 선택 하면 입력 파라메터에 대한 정보를 확인 할 수 있습니다.
7. Test 버튼을 클릭 합니다.

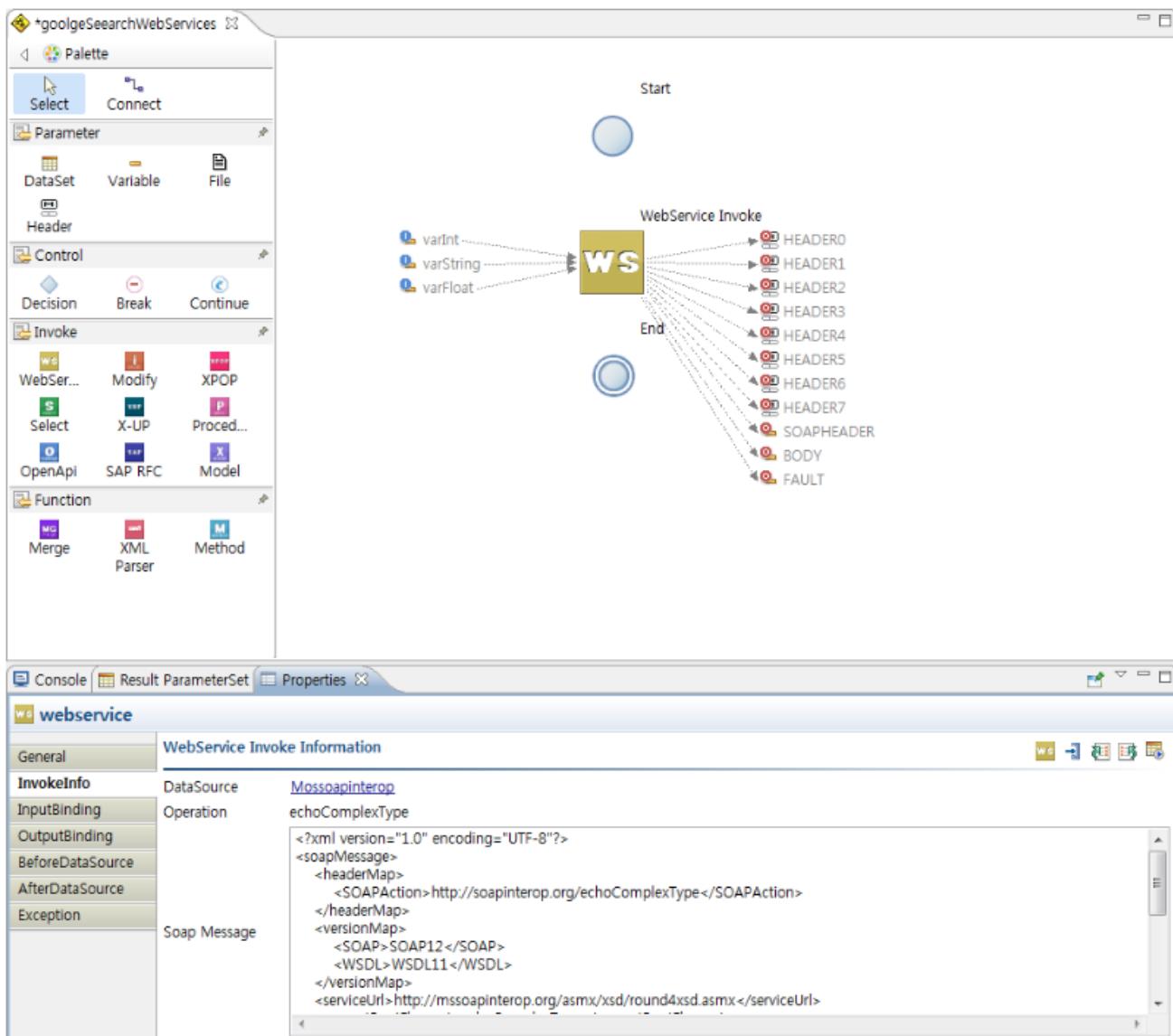


8. Define Input Value 위자드가 실행 되면 입력 파라메터 값을 아래와 같이 입력 한 후 'Finish' 버튼을 클릭 합니다.



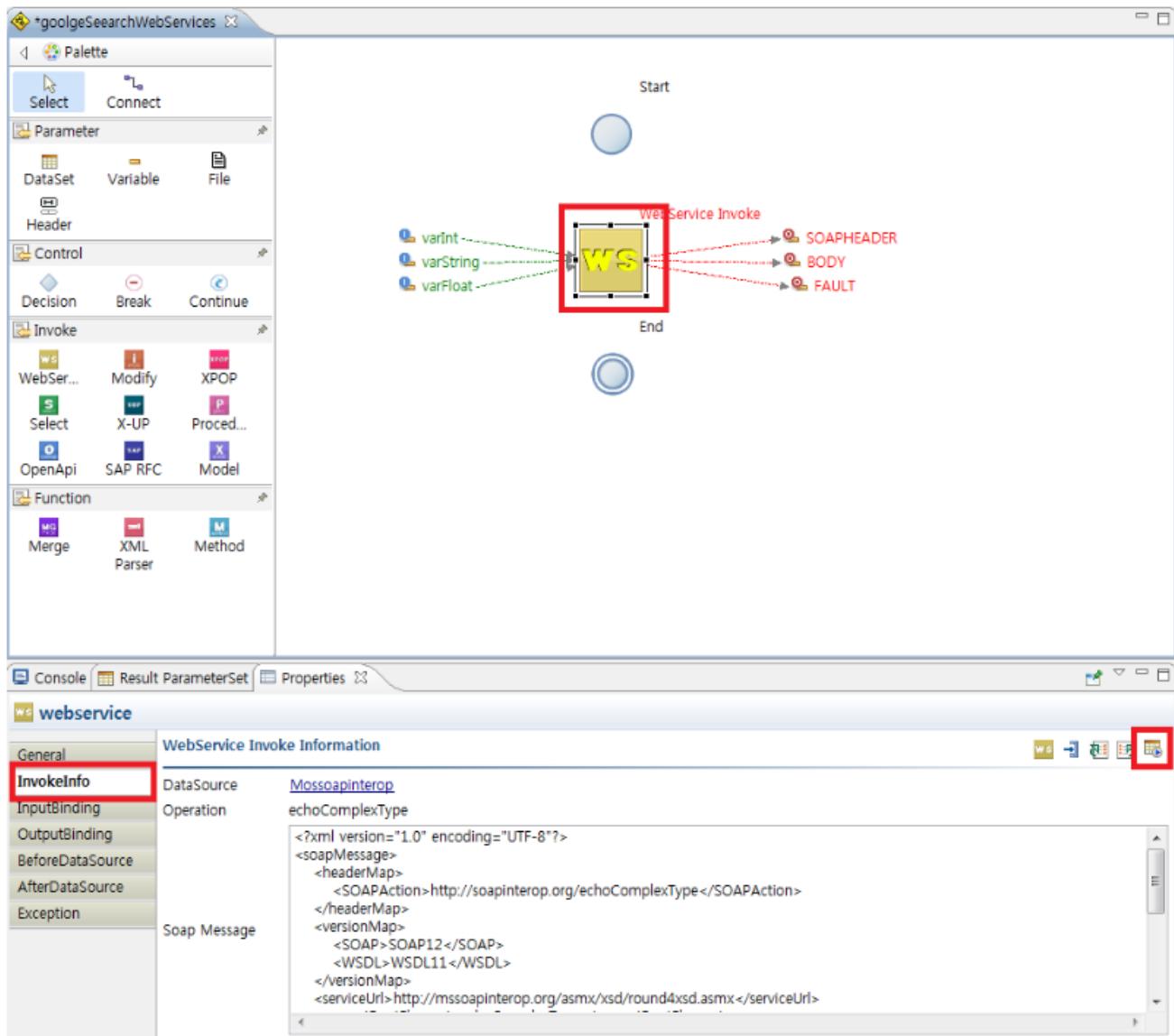
9. Invoke WebService 위자드의 OK 버튼을 클릭 할 경우 테스트 시 확인 한 값이 출력 파라메터로 설정 된 것을 알 수 있습니다.

 불 필요한 HTTP Header의 정보는 지워도 정상 작동 합니다.



Invoke 테스트 하기

1. 생성 된 WebService Invoke입니다. Invoke의 테스트 결과를 바탕으로 입력 파라미터와 출력 파라미터가 생성 됩니다.
2. 사용하지 않는 HTTP Header 응답 값을 지웁니다.
3. WebService Invoke의 Properties의 'InvokeInfo'를 선택하고 [] 를 클릭하여 테스트를 진행합니다.



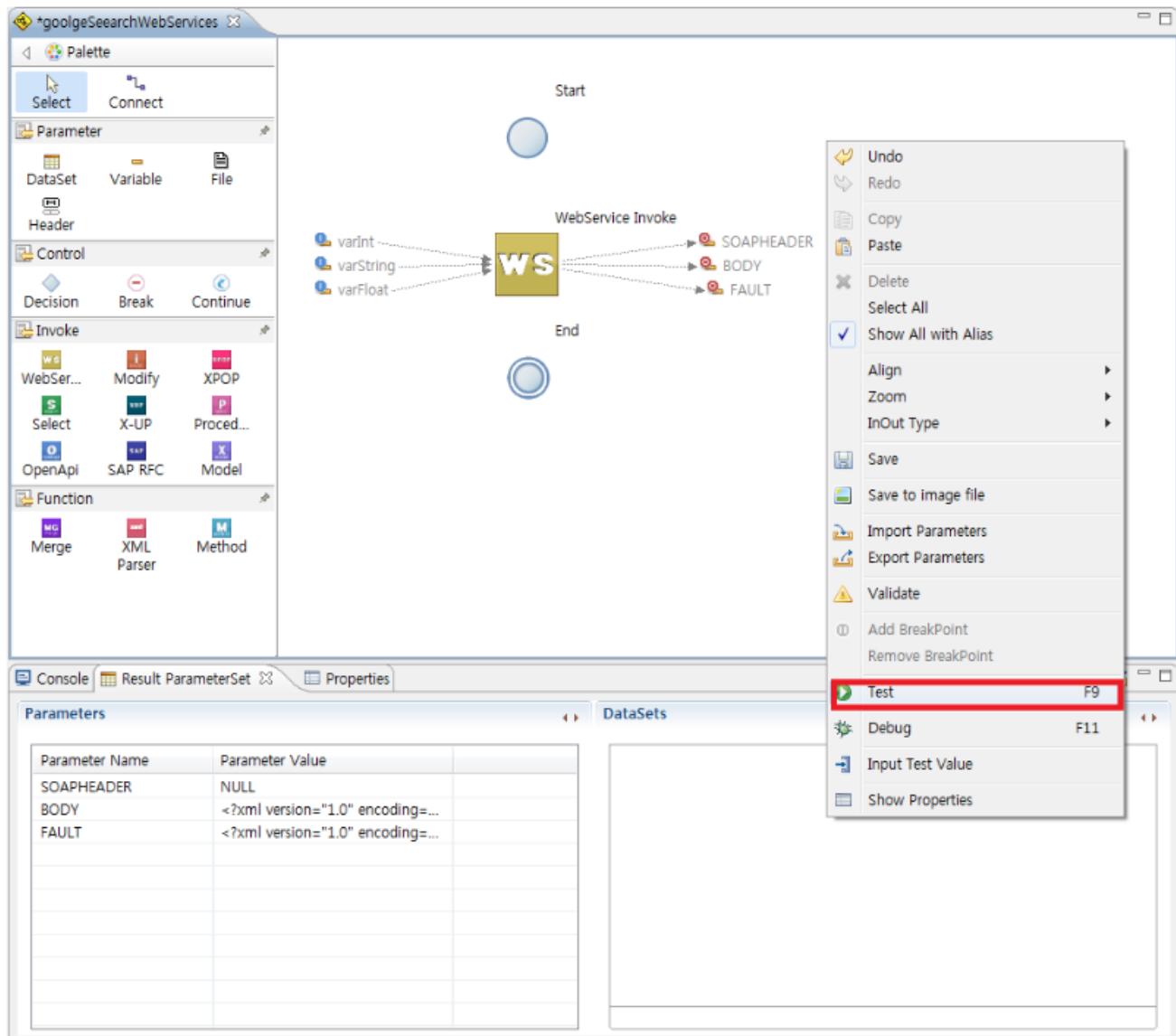
4. Define Input Value 위자드가 생성 되면 Properties 설정 시 입력 한 테스트 값을 입력 하고 'Finish' 버튼을 클릭합니다.
5. 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인할 수 있습니다.

The screenshot shows the X-UP tool interface with the "Result ParameterSet" tab selected. The "Parameters" section displays the following table:

Parameter Name	Parameter Value
SOAPHEADER	NULL
BODY	<?xml version="1.0" encoding="...>
FAULT	<?xml version="1.0" encoding="...>

모델 테스트하기

- 모델 에디터에서 Palette의 Connect를 선택하여 Start 노드와 WebService Invoke, end노드를 연결 해 줍니다.
- 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.
- 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.



8.7 X-UP Invoke를 이용한 모델 개발하기

이 절에서는 Automation 모델을 이용하여, 웹 브라우저에 deploy한 X-UP 모델을 불러와 데이터를 변경하는 개발 방법을 설명 합니다.



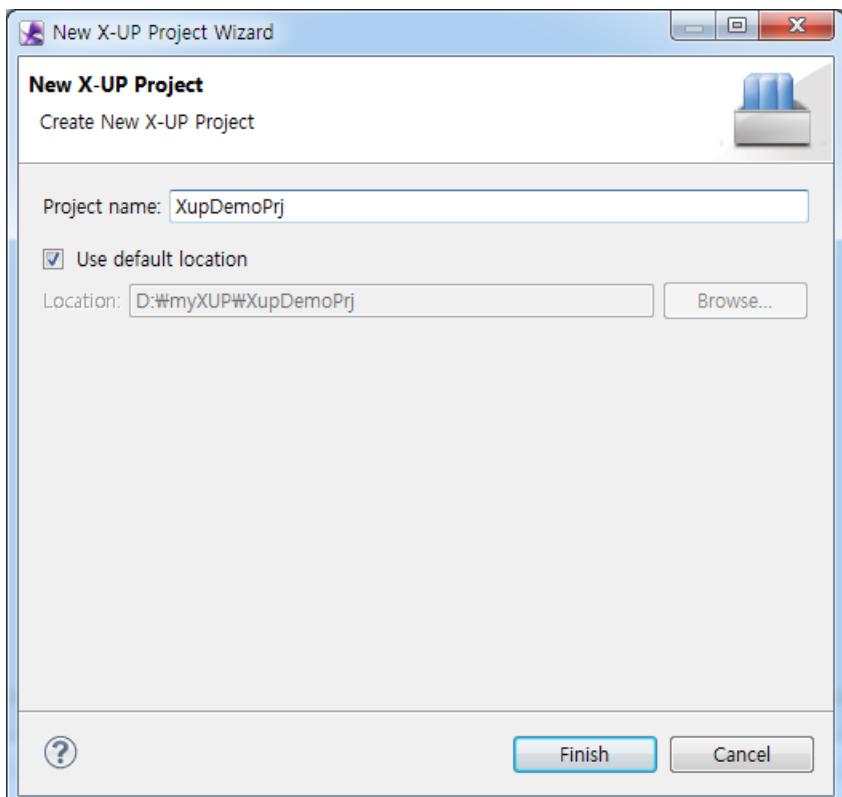
디플로이 방법은 [10.1 모델을 X-UP서버에 Deploy하기](#)를 참조하십시오.

이 절에서 설명하는 Automation 모델의 X-UP Invoke 개발 단계는 다음과 같습니다.

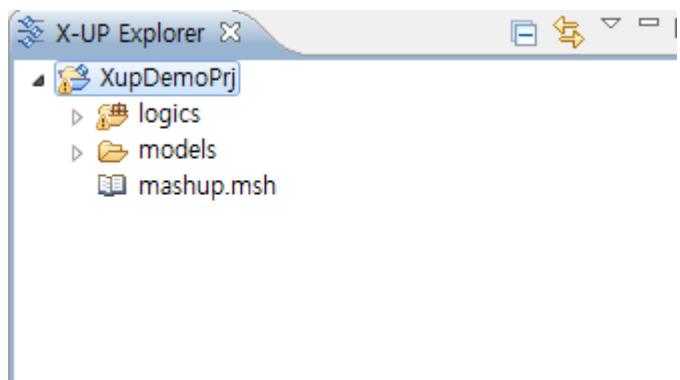
- X-UP 프로젝트 생성하기
- Automation 모델을 생성 및 X-UP Invoke 생성하기
- 데이터 소스 생성 및 Properties 설정하기
- Invoke 테스트하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 프로젝트 이름을 입력하고 "Finish" 버튼을 클릭합니다.

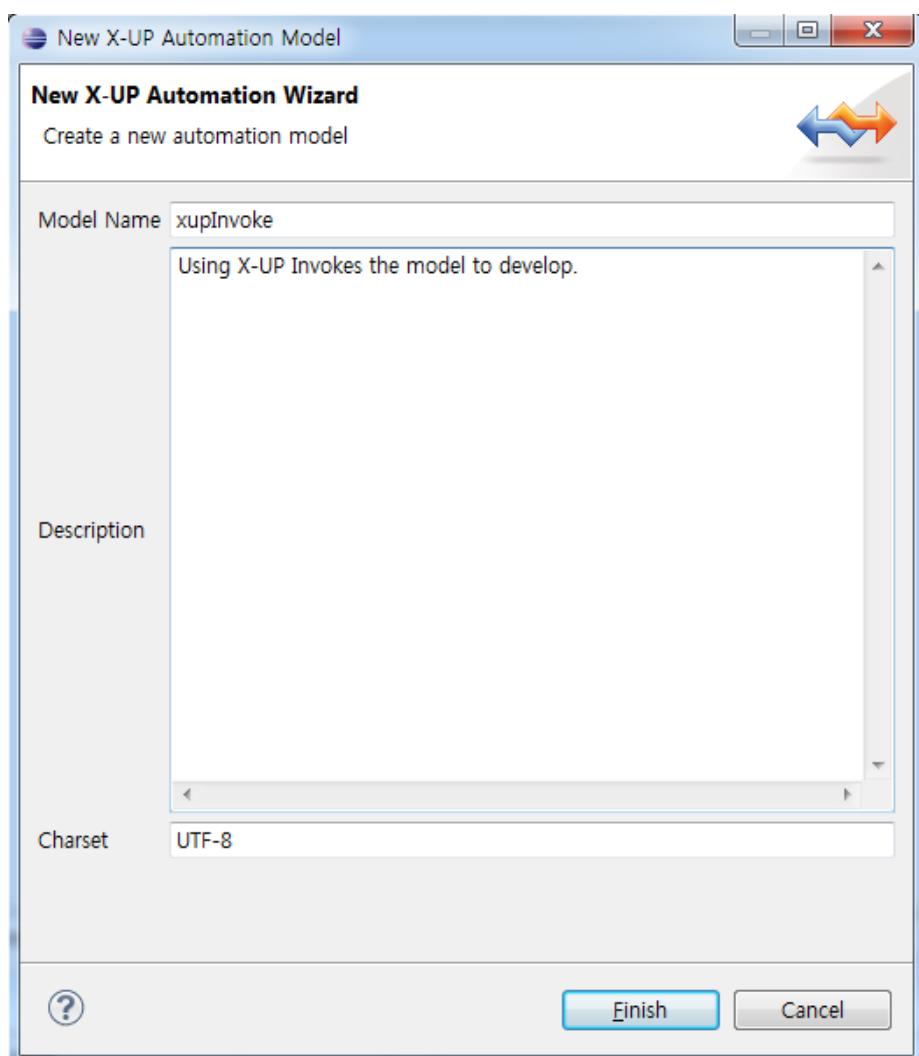


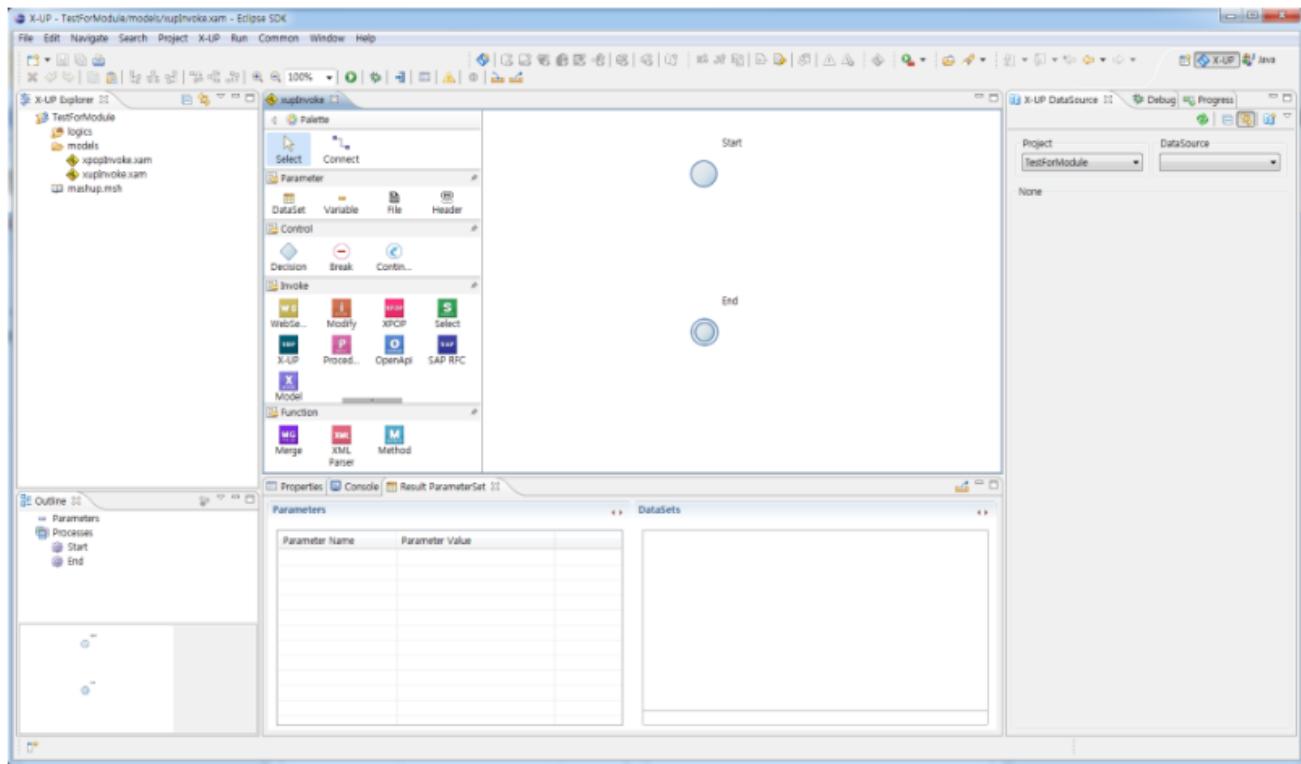
3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.



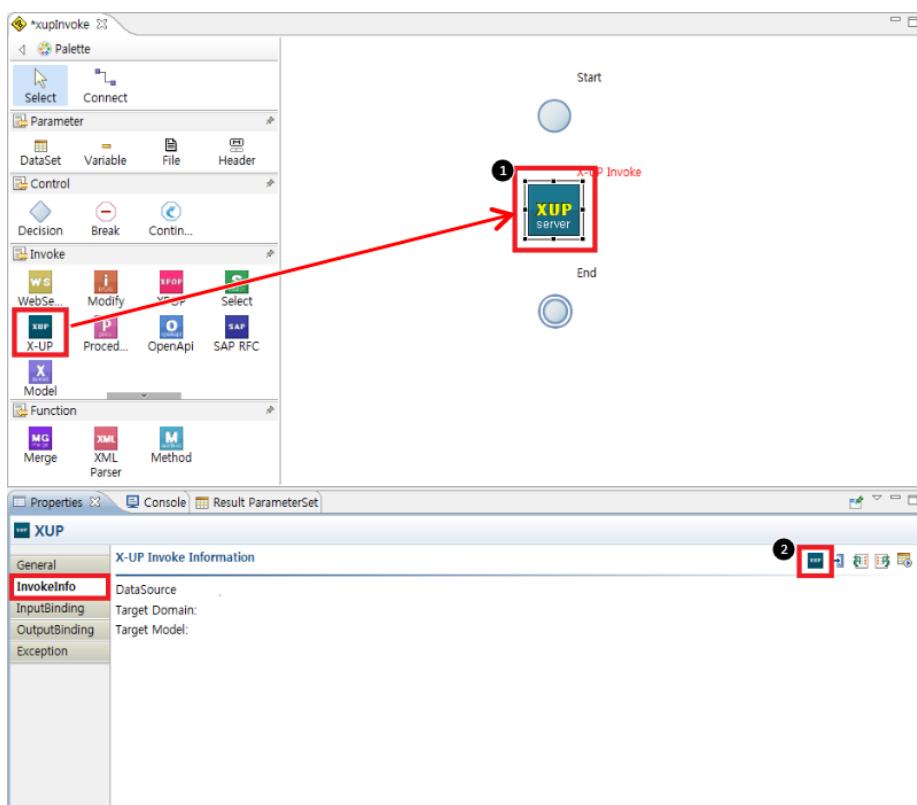
Automation 모델 생성 및 X-UP Invoke 생성하기

1. File New X-UP Automation Model 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 "Finish" 버튼을 클릭합니다.



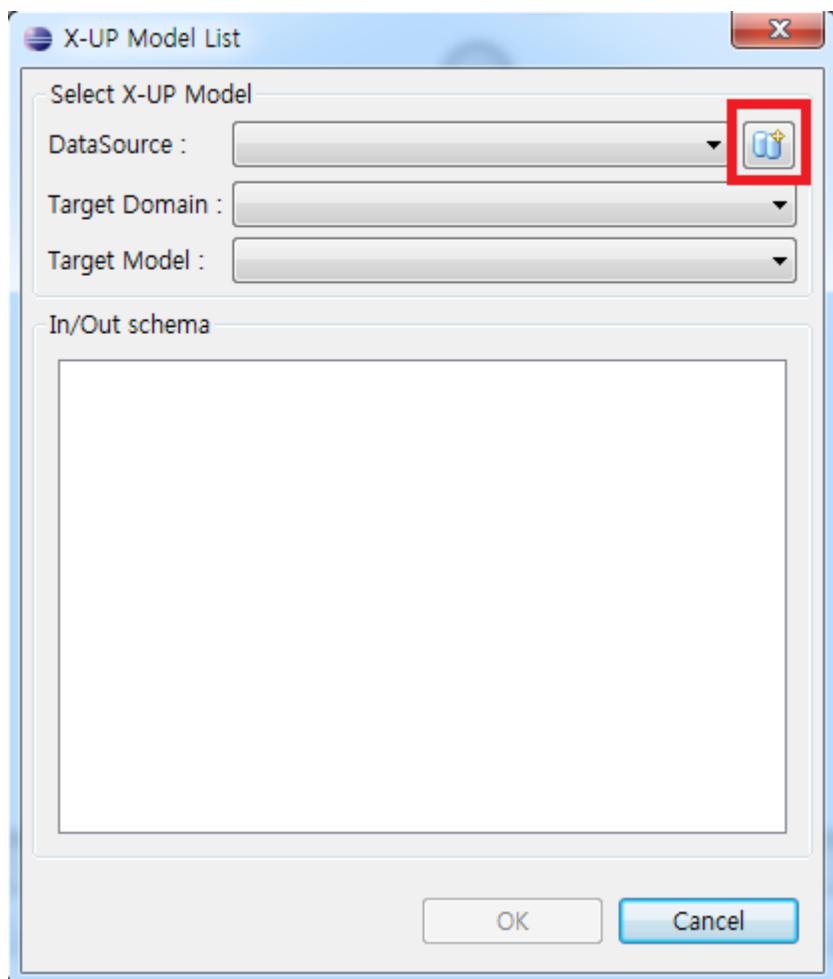


3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.
4. Palette 의 Invoke 중 'X-UP'를 선택하고 에디터를 클릭 하게 되면 X-UP Invoke가 생성 됩니다.
5. 생성된 X-UP Invoke를(❶) 더블클릭 하거나 'properties' 탭을 클릭하여 'X-UP icon'❷ 을 클릭하면 위자드가 실행 됩니다.



데이터소스 생성 및 Properties 설정하기

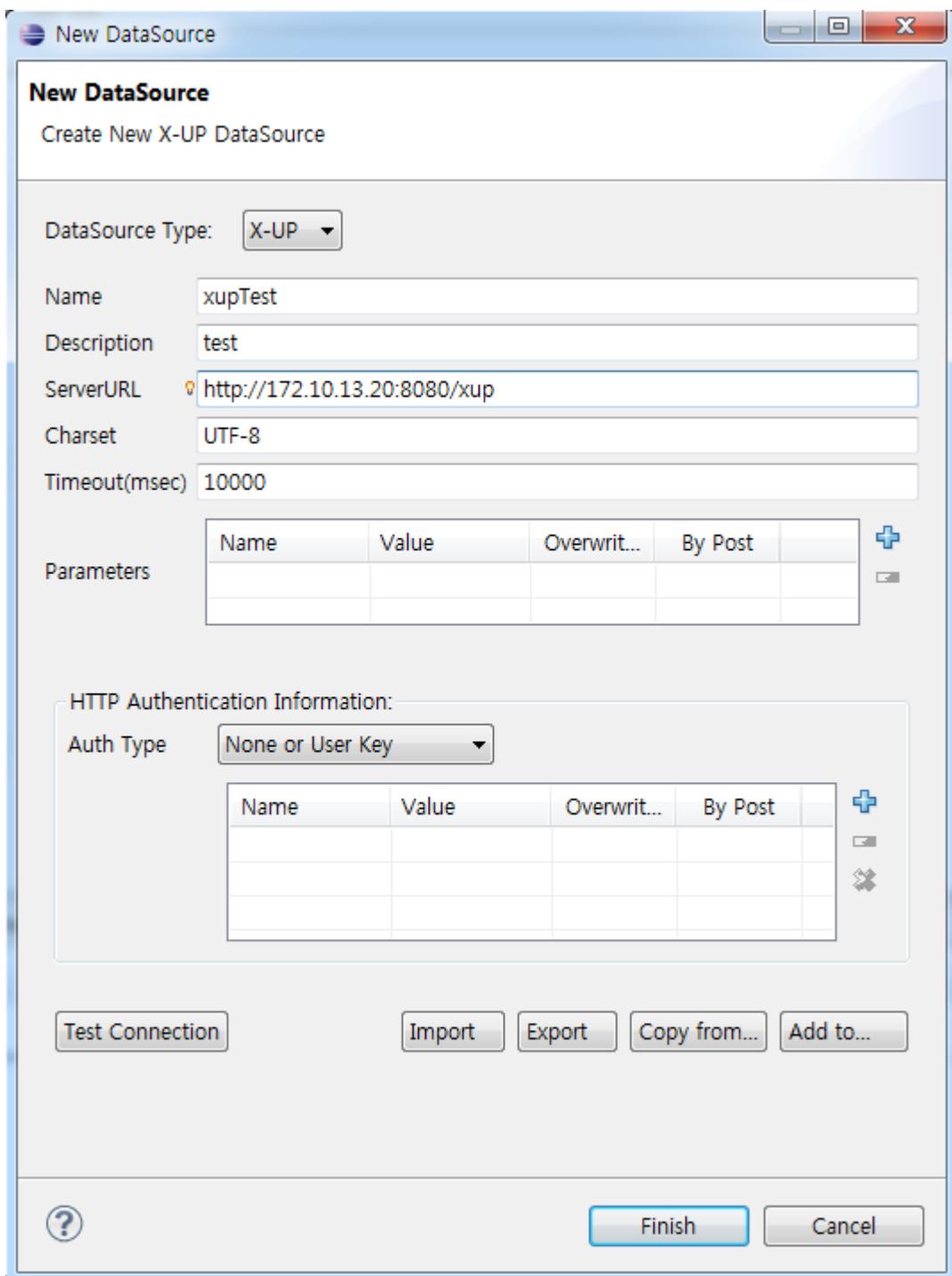
1. X-UP Model List 위자드에서 **create new DataSource** [] 를 클릭합니다.



2. **New DataSource** 위자드에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
 3. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

Field Name	Field Value
Name	'xupTest' 데이터 소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
Description	'test' 데이터 소스의 간략한 설명을 입력합니다.
ServerURL	'http://172.10.12.152:8080/xup/FrontControlServlet.do' 데이터 소스를 로딩 할 수 있는 서버의 URL을 입력합니다.
Charset	'UTF-8' 문자 인코딩을 나타냅니다.

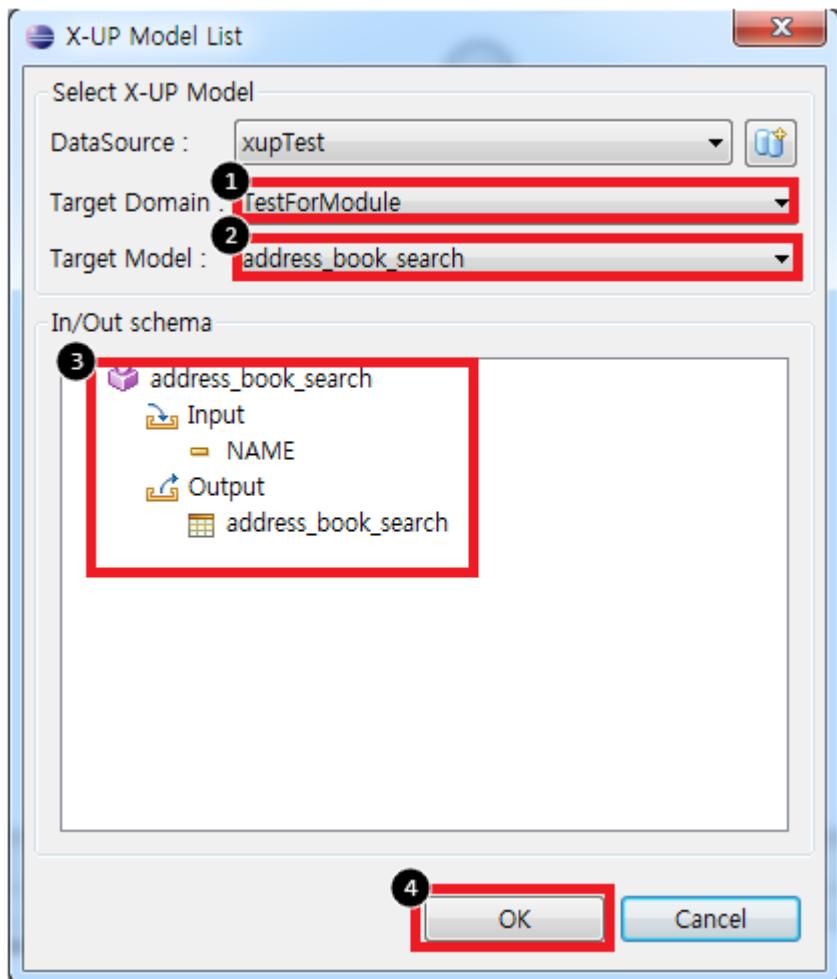
Field Name	Field Value
Timeout	'10000' 서버와의 접속을 유지할 최대 시간
Parameters	넘겨 줄 파라미터 값을 입력합니다.



데이터 소스가 정상적으로 'xupTest'로 지정이 되었는지 확인합니다.

- ① 가지고 온 모델이 위치하는 도메인을 선택합니다.

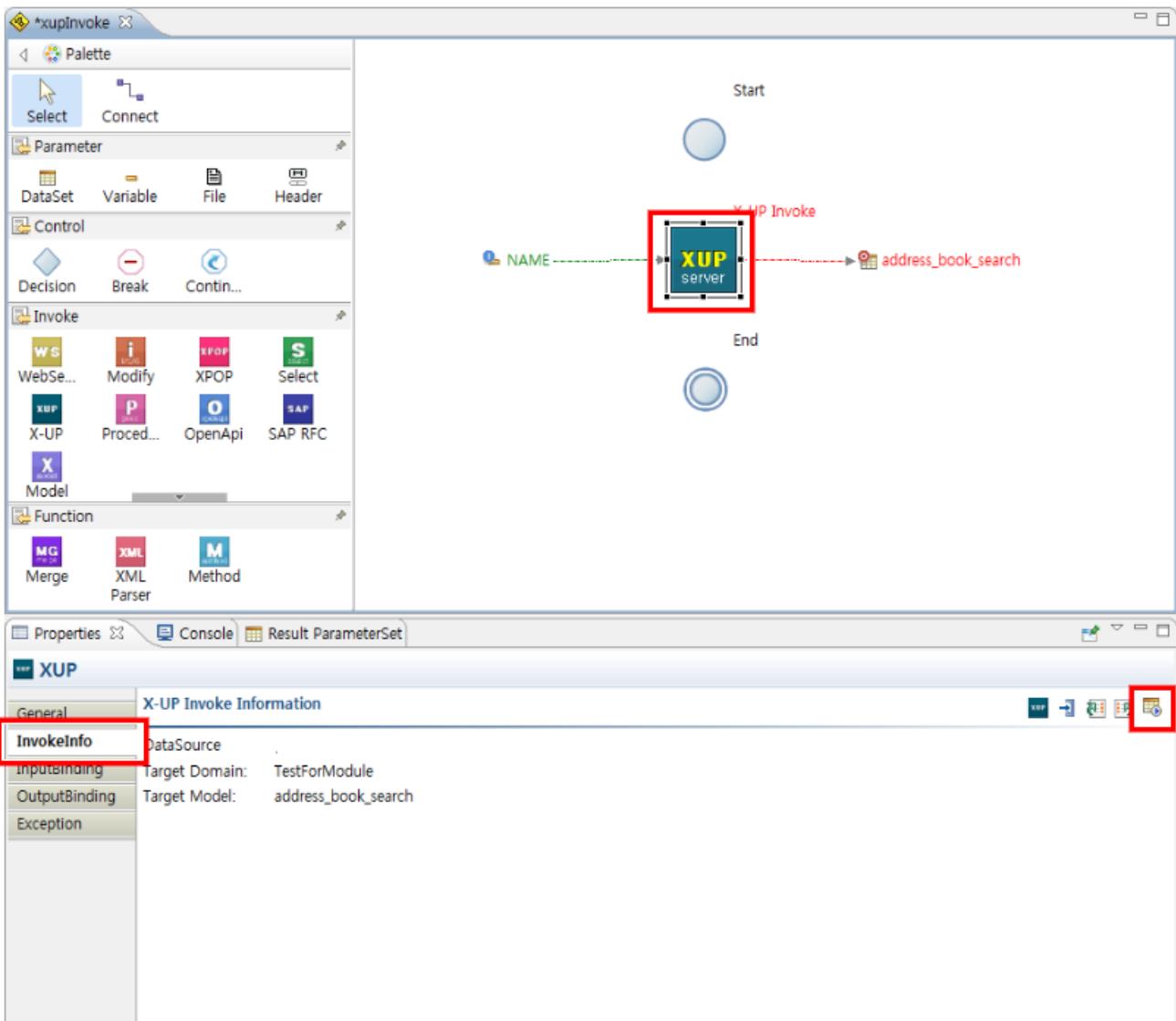
- ② 모델을 선택합니다.
- ③ 모델의 입력 파라미터와 출력 파라미터의 정보를 확인 할 수 있습니다.
- ④ 'OK'버튼을 클릭하여 위자드를 종료합니다.



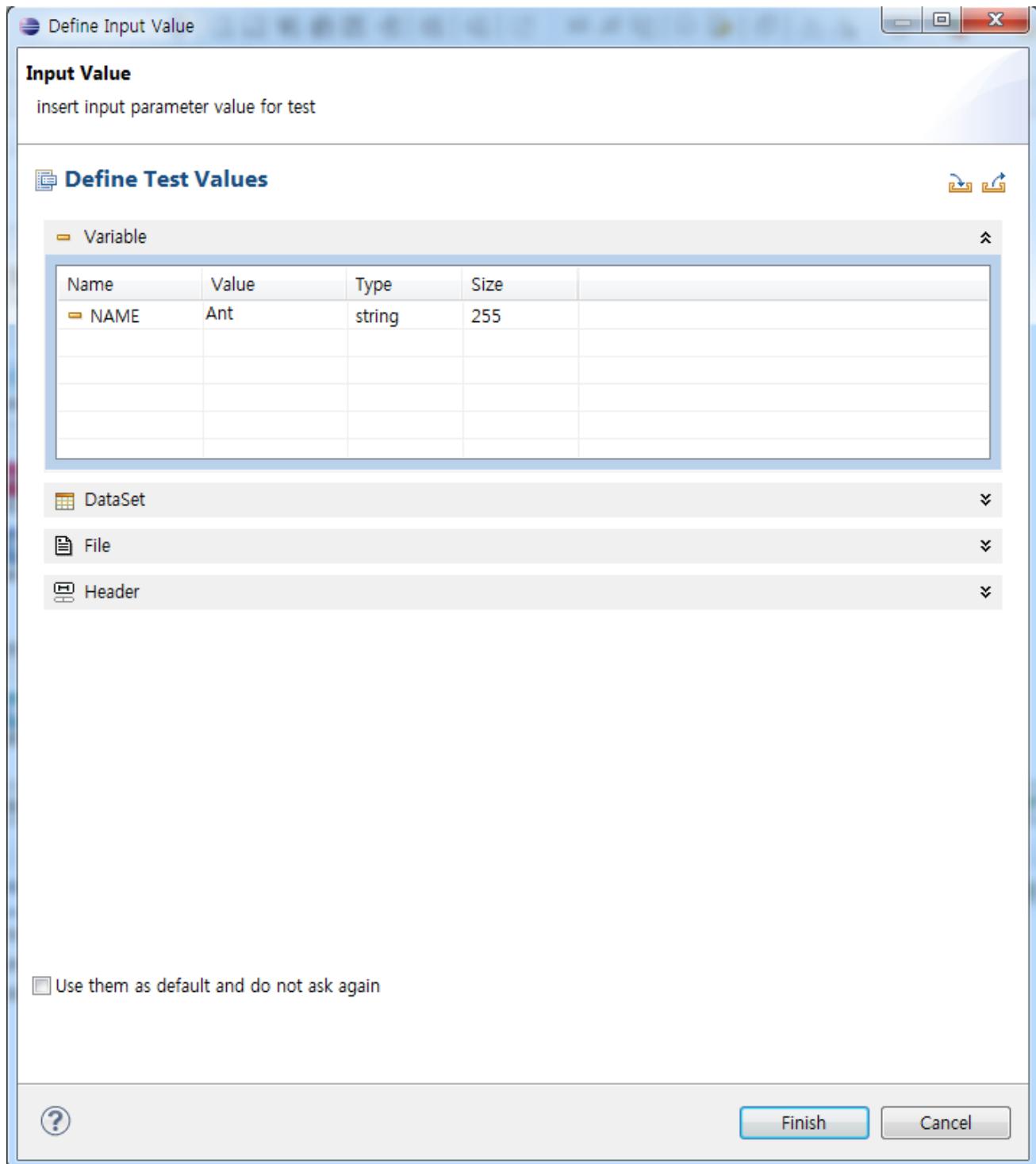
4. OK 버튼을 클릭 할 경우 입력 파라미터와 출력 파라미터가 생성이 된 것을 확인 할 수 있습니다.

Invoke 테스트 하기

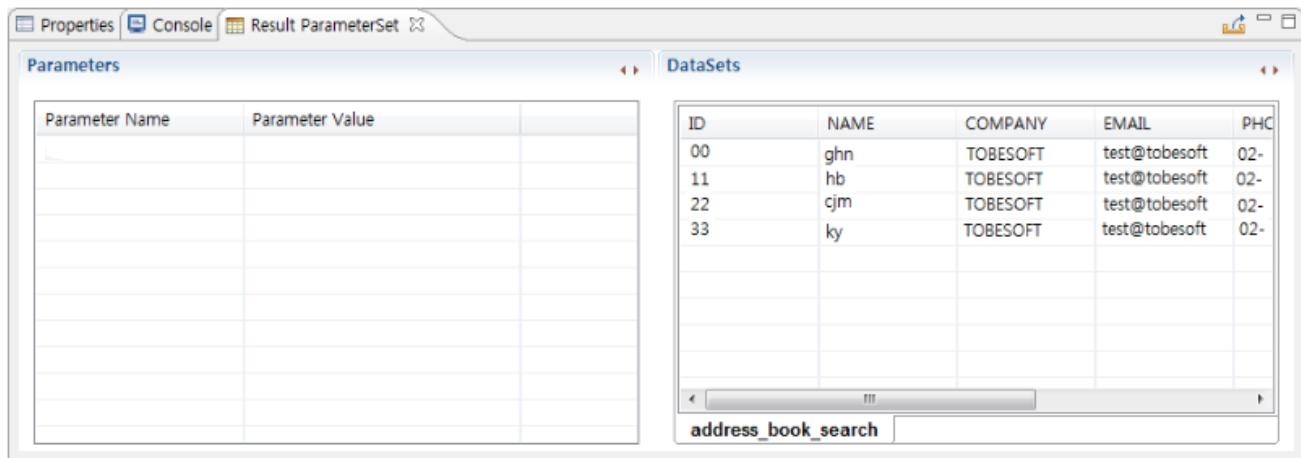
1. X-UP Invoke의 Properties의 'InvokeInfo'를 선택하고 [] 를 클릭하여 테스트를 진행합니다.



2. Define Input Value에서 값을 입력하고 'Finish' 버튼을 클릭합니다.

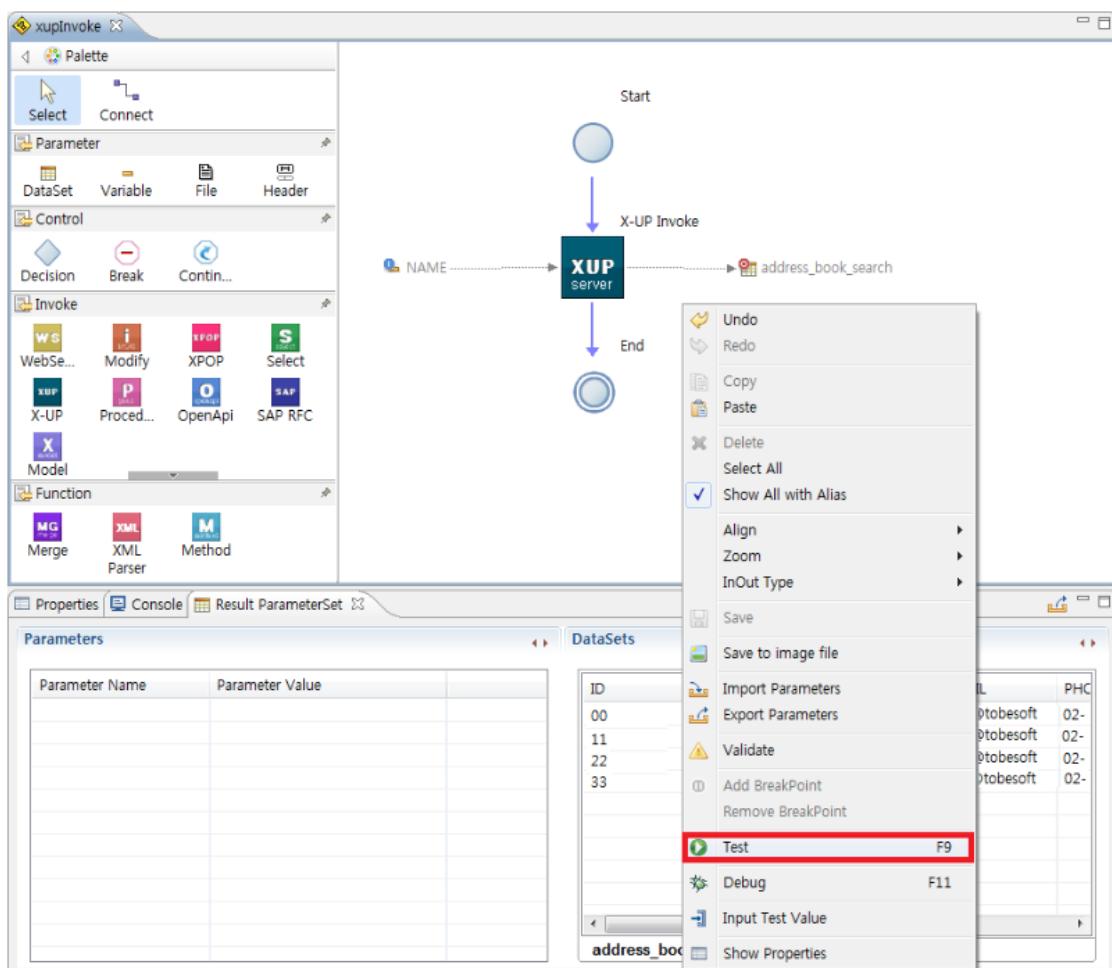


3. 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인 할 수 있습니다.



모델 테스트하기

- 모델 에디터에서 Palette의 Connect를 선택하여 Start 노드와 X-UP Invoke, End 노드를 연결해 줍니다.
 - 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.
- 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.



8.8 DataSet Row Loop Function을 이용하여 데이터 가공하기

이 절에서는 데이터셋의 한 행에 대하여 loop를 돌면서 반복문을 수행하여 특정 데이터만 모아 새로운 데이터셋을 만드는 DataSetRowLoop 함수에 대해 설명합니다.



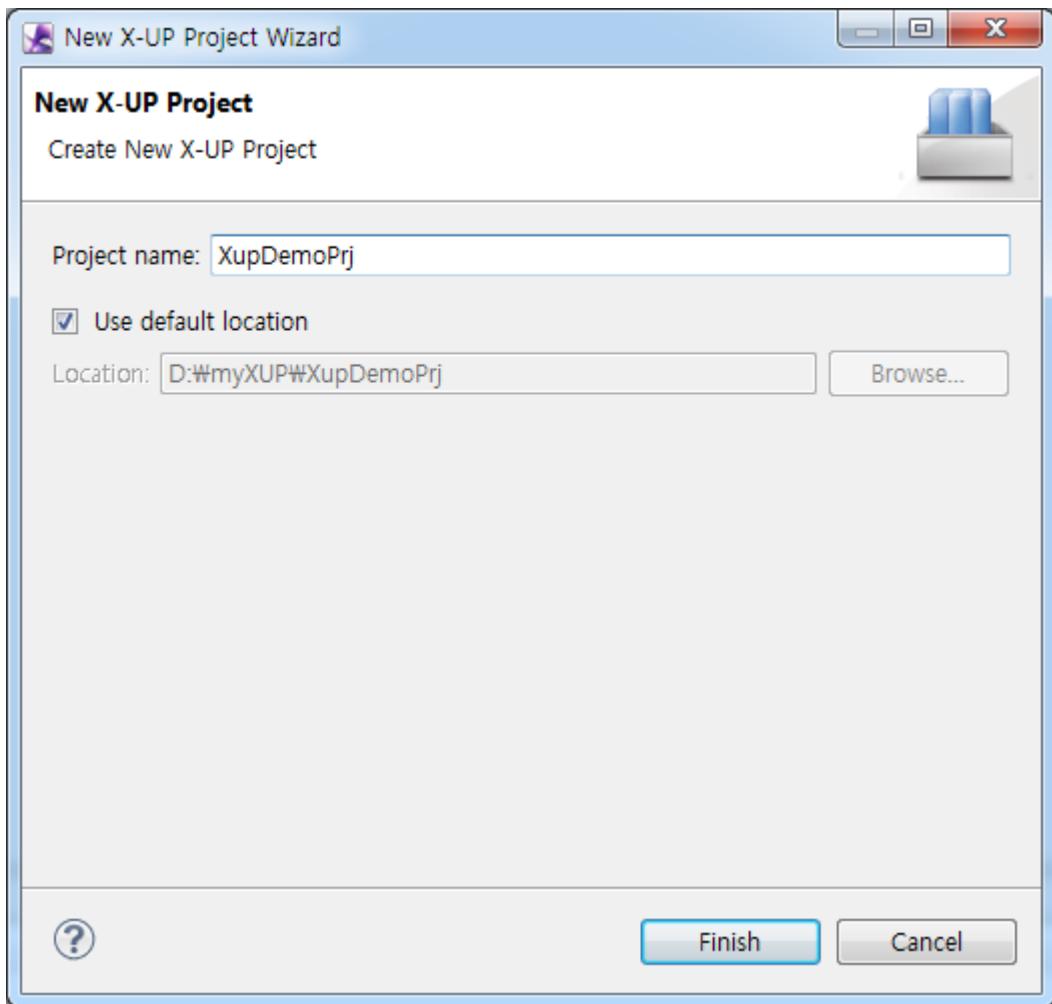
이 절에서 사용한 데이터베이스 테이블 정보는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 참조하십시오.

이 절에서 설명하는 DataSetRowLoop 함수를 이용한 모델 개발단계는 다음과 같습니다.

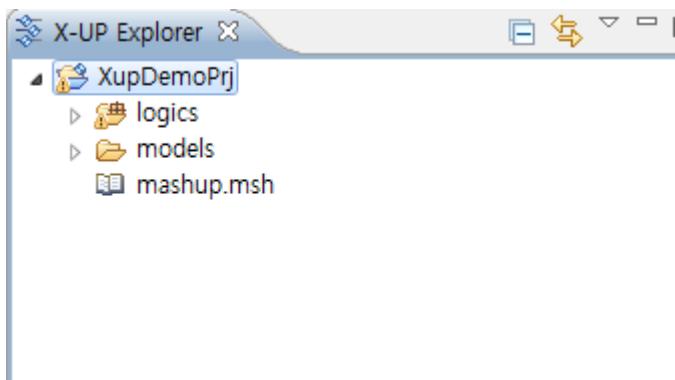
- X-UP 프로젝트 생성하기
- Automation 모델 생성하기
- 입력 파라메터를 위한 SAP RFC로직 구현하기
- DataSetRowLoop 함수 생성하고 로직 설정하기
- DataSetRowLoop 함수에 이벤트 추가하기
- UserMethod Function을 이용하여 출력 파라메터 설정하기
- Function 테스트하기
- 모델 테스트

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭 합니다.

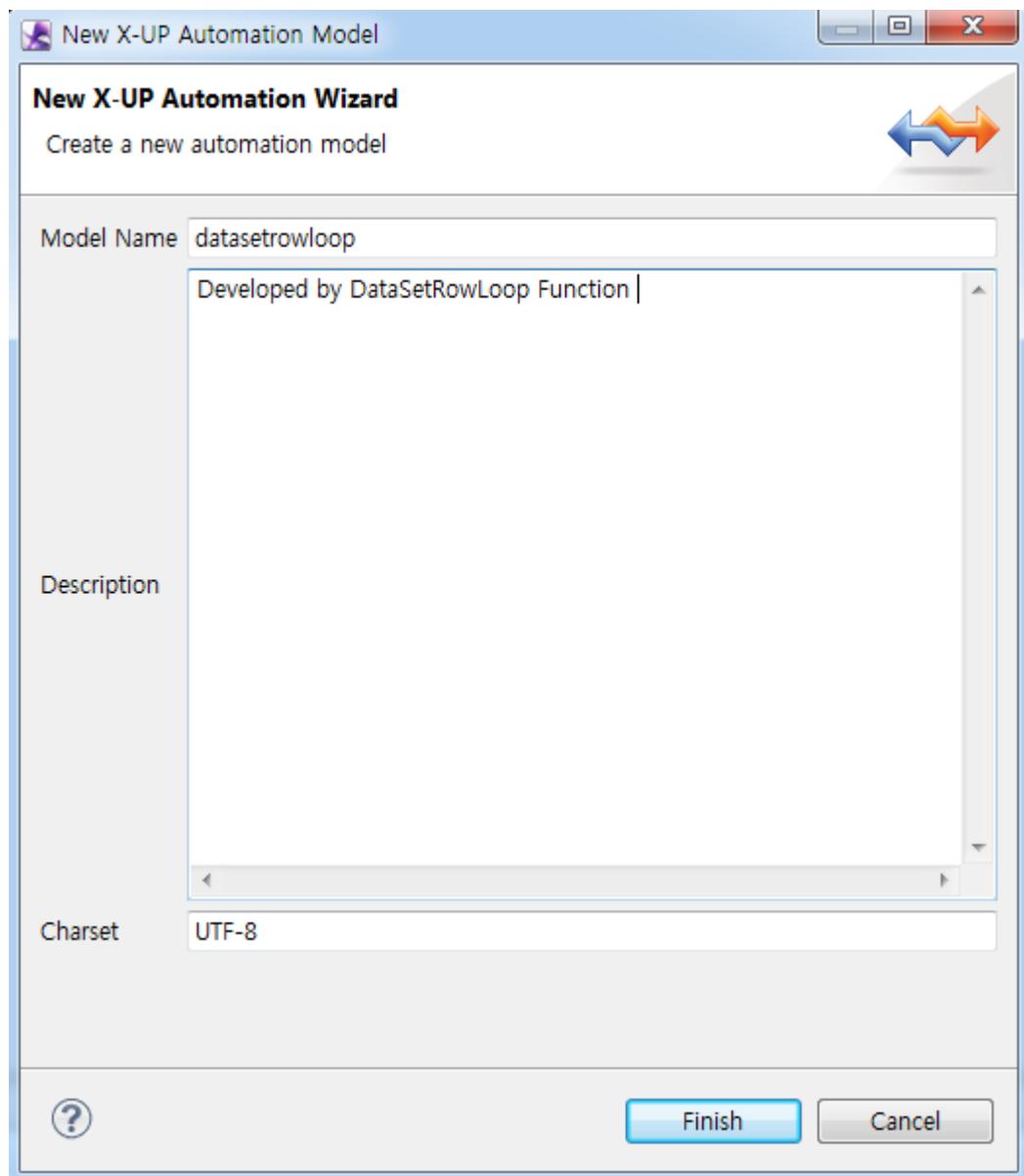


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

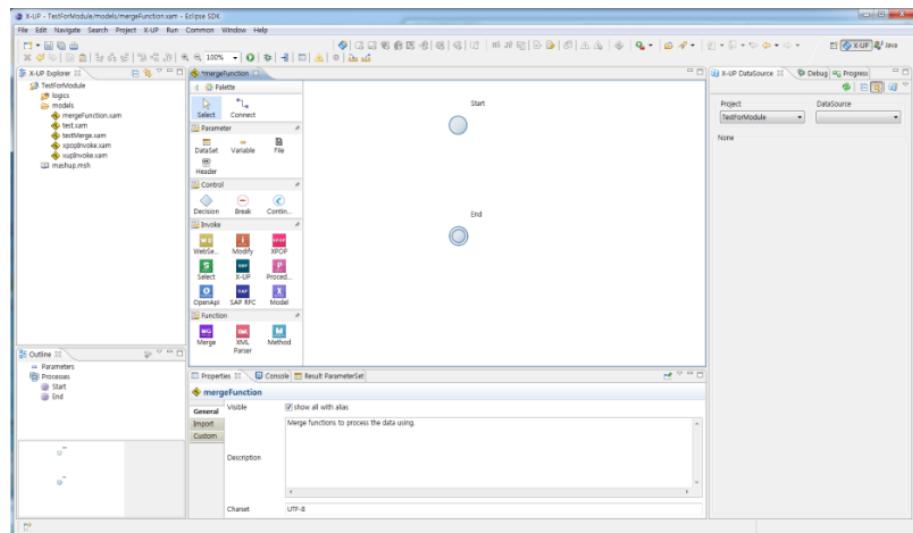


Automation 모델 생성하기

- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 원하는 모델 이름을 입력하고 OK 버튼을 클릭합니다.



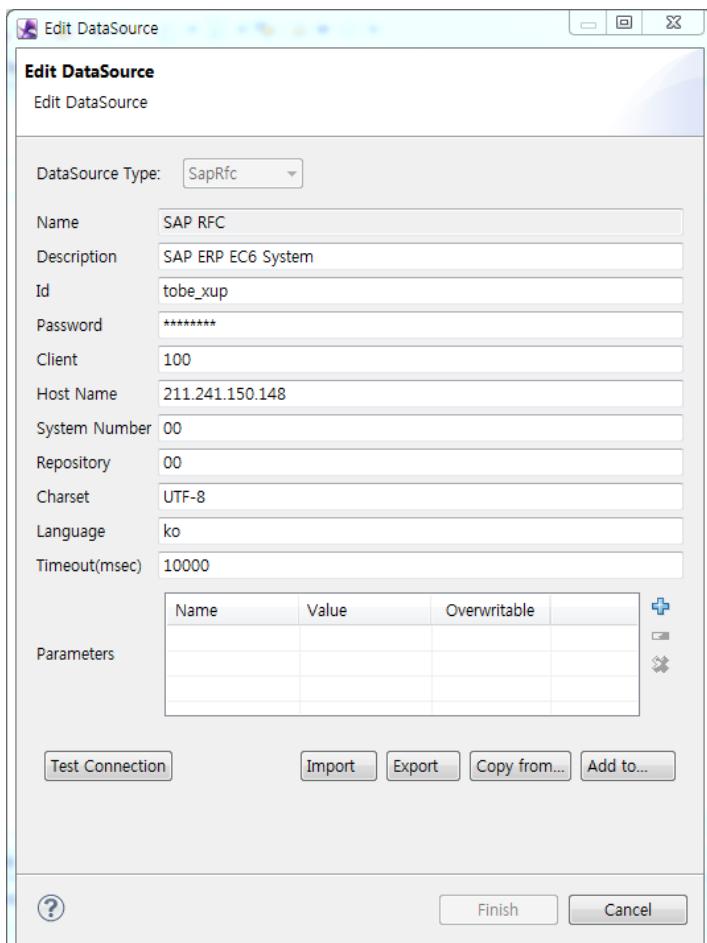
3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



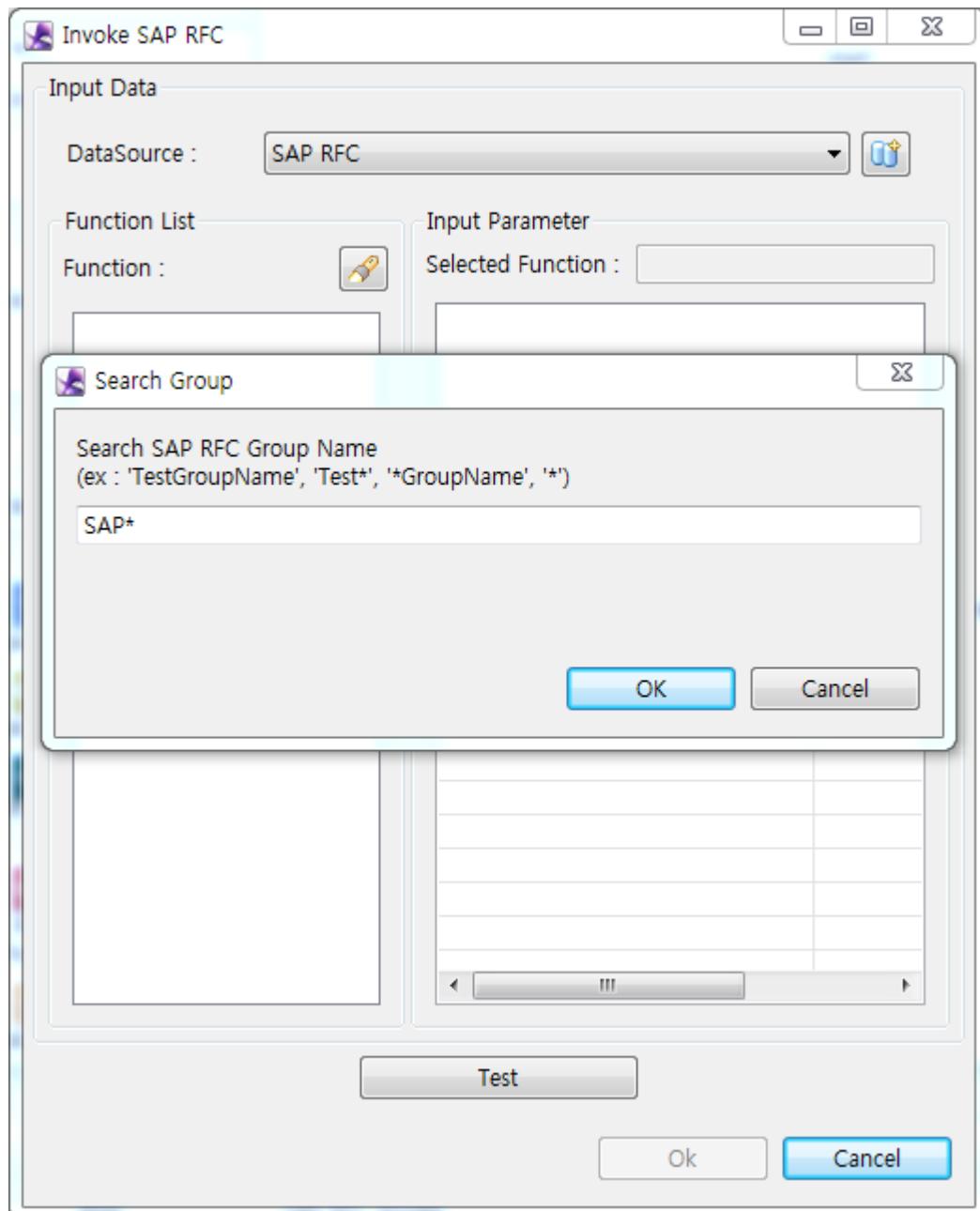
입력 파라메터를 위한 SAP RFC로직 구현하기

- 화면 우측의 X-UP DataSource 위자드에서 new DataSource[]를 선택해 Wizard를 실행합니다.
- New DataSource 위자드에서 새로운 데이터소스를 정의합니다. 아래와 같이 각 필드 값을 입력한 후 'Test Connection' 버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
- Connection 확인 후에 'Finish' 버튼을 클릭합니다.

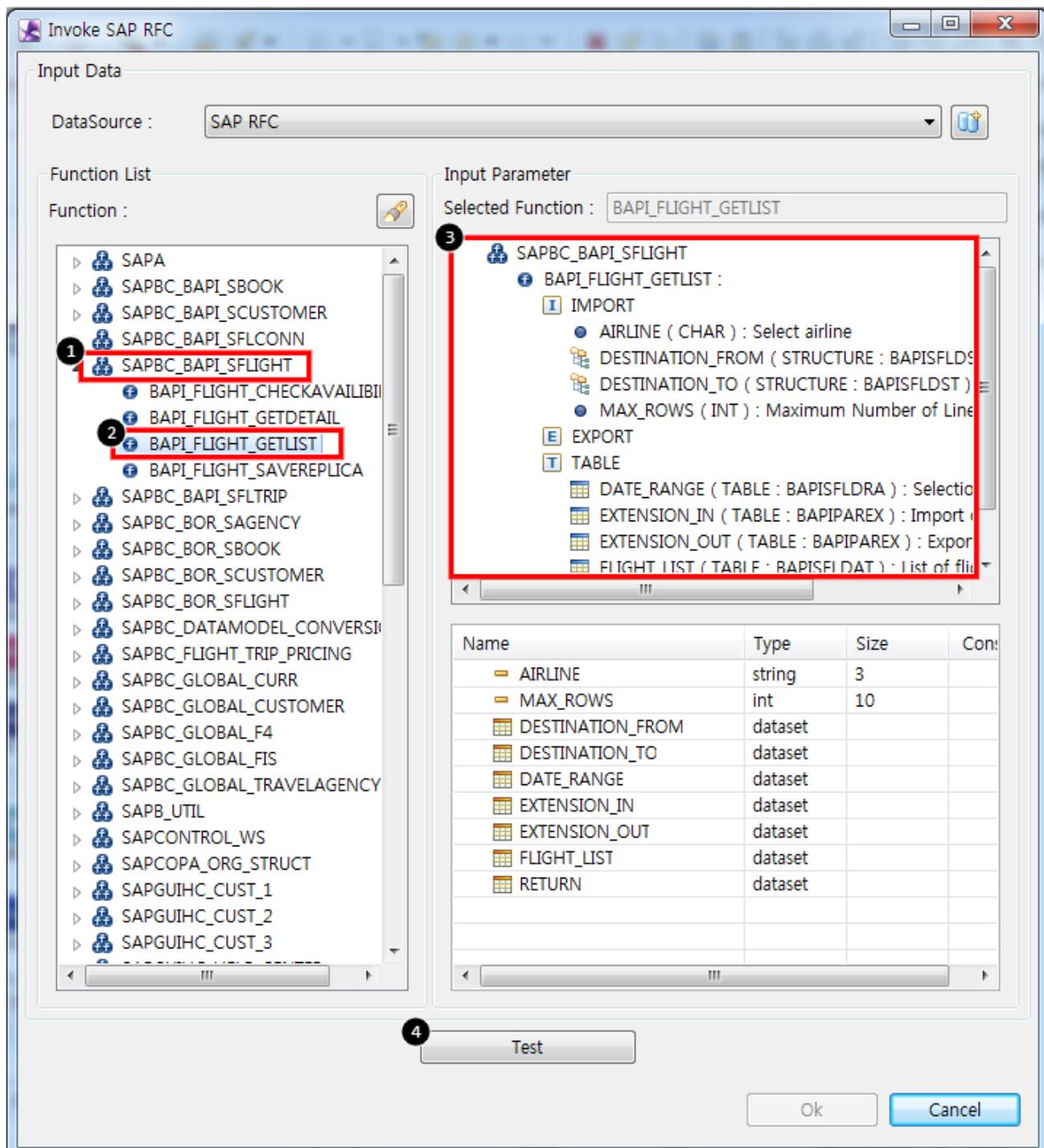
Field Name	Field Value
Name	'SAP RFC' 데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
ID	SAP RFC 서버 접속 ID.
PASSWORD	SAP RFC 서버 접속 PASSWORD.
Client	로그인 계정에서 사용할 client 정보
Host Name	서버의 IP
System Number	로그인 계정에서 사용 할 System Number 정보
Repository	로그인 계정에서 사용할 Repository 정보
Charset	사용할 charset
Timeout	서버와의 접속을 유지할 최대 시간

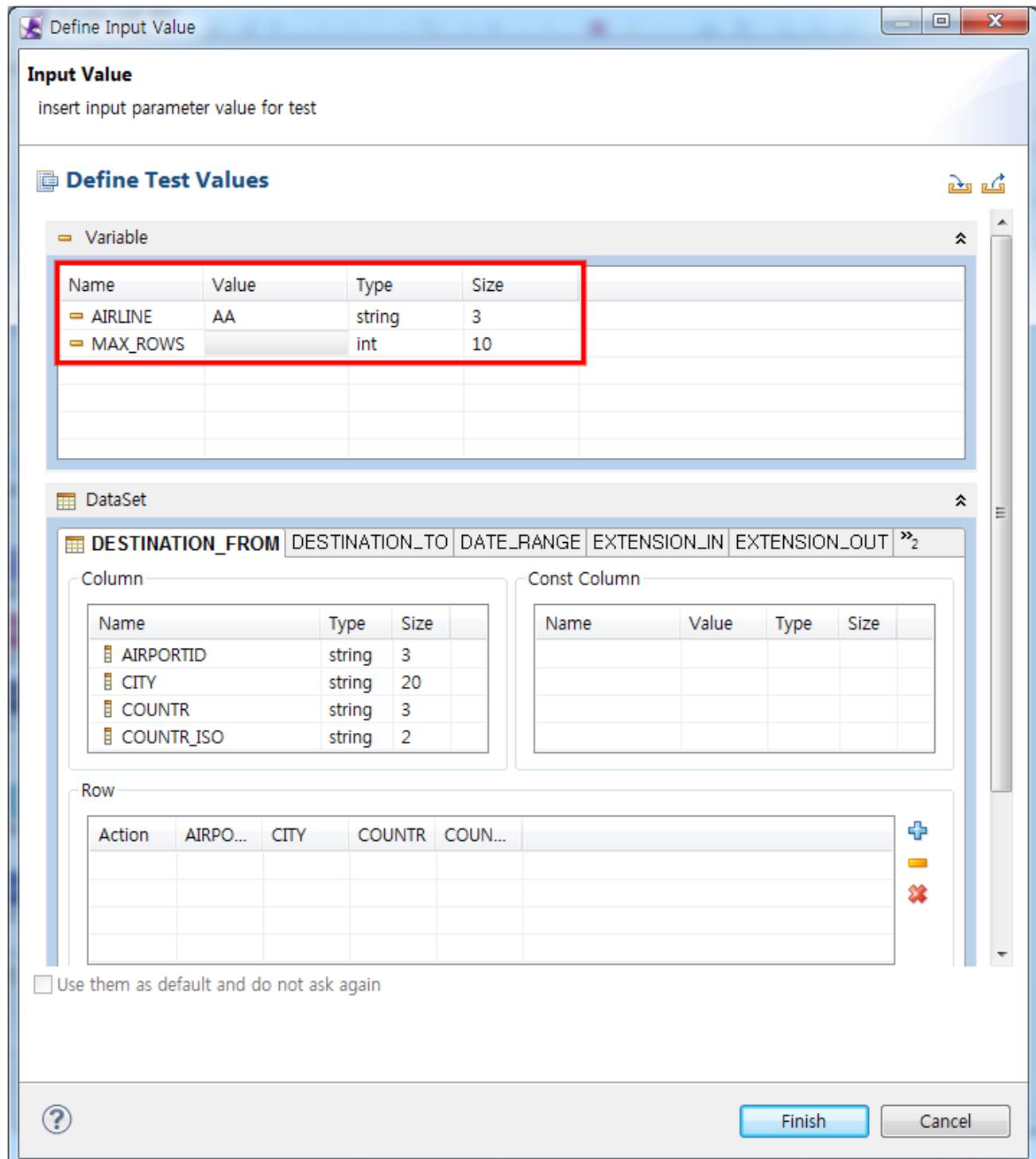


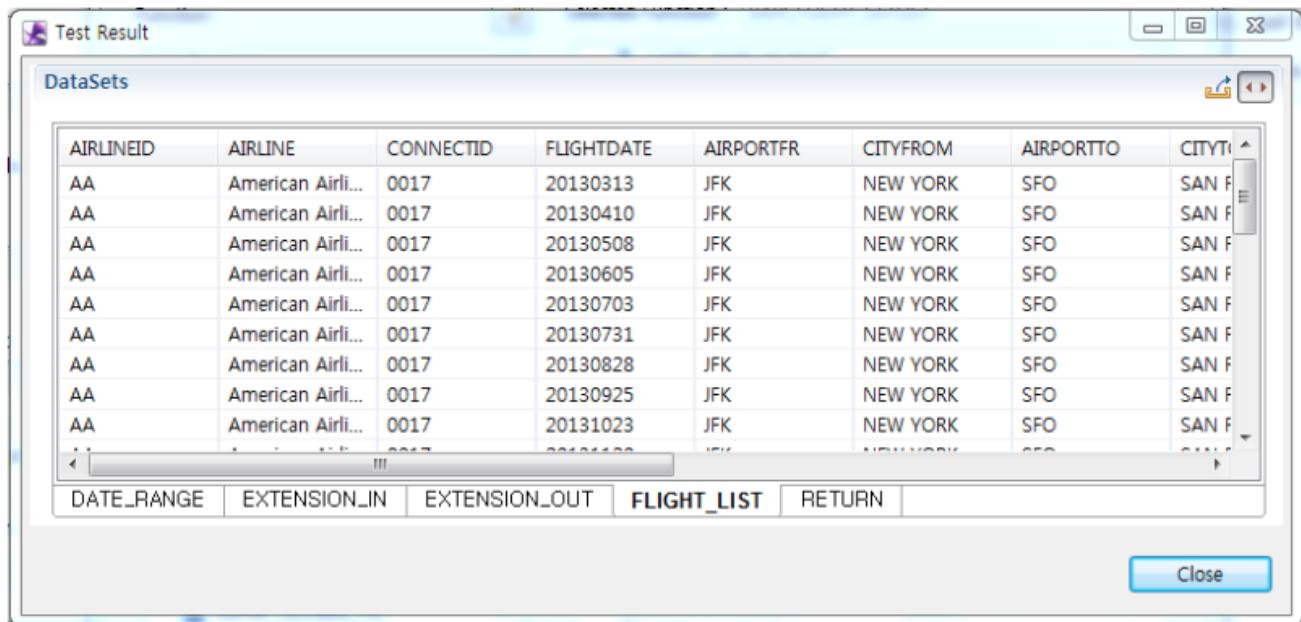
4. DataSource를 SAP RFC을 선택하고 get function list 버튼 [] 을 클릭하여 서버에 등록된 function 리스트를 호출합니다.
5. 그룹(function) 검색 시 * 검색이 가능합니다.



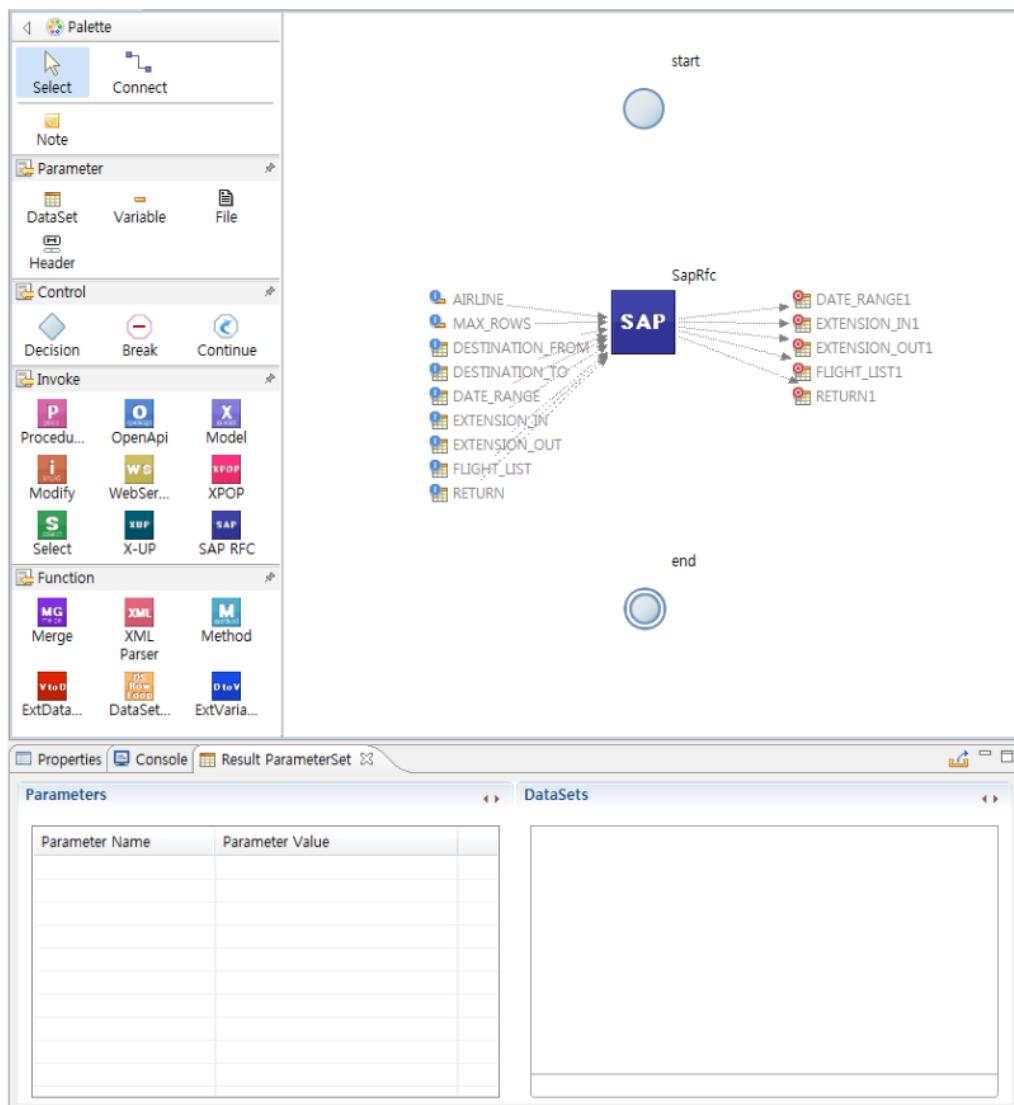
6. 검색 된 function 리스트 중 호출 하고자 하는 function 을 선택(1) 하면 해당 function의 입력 파라메터 와 출력 파라메터의 정보를 확인(2) 할 수 있습니다.
7. 동시에 SAP RFC 파라메터 형태를 X-UP Parameter 형태(3)로 나타납니다.
8. Test 버튼(4)을 클릭하여 생성 된 Define Input Value 위자드의 입력 파라메터의 값을 설정 하고 'Finish' 버튼을 클릭하여 정상적으로 데이터가 표시되는지 확인 합니다.





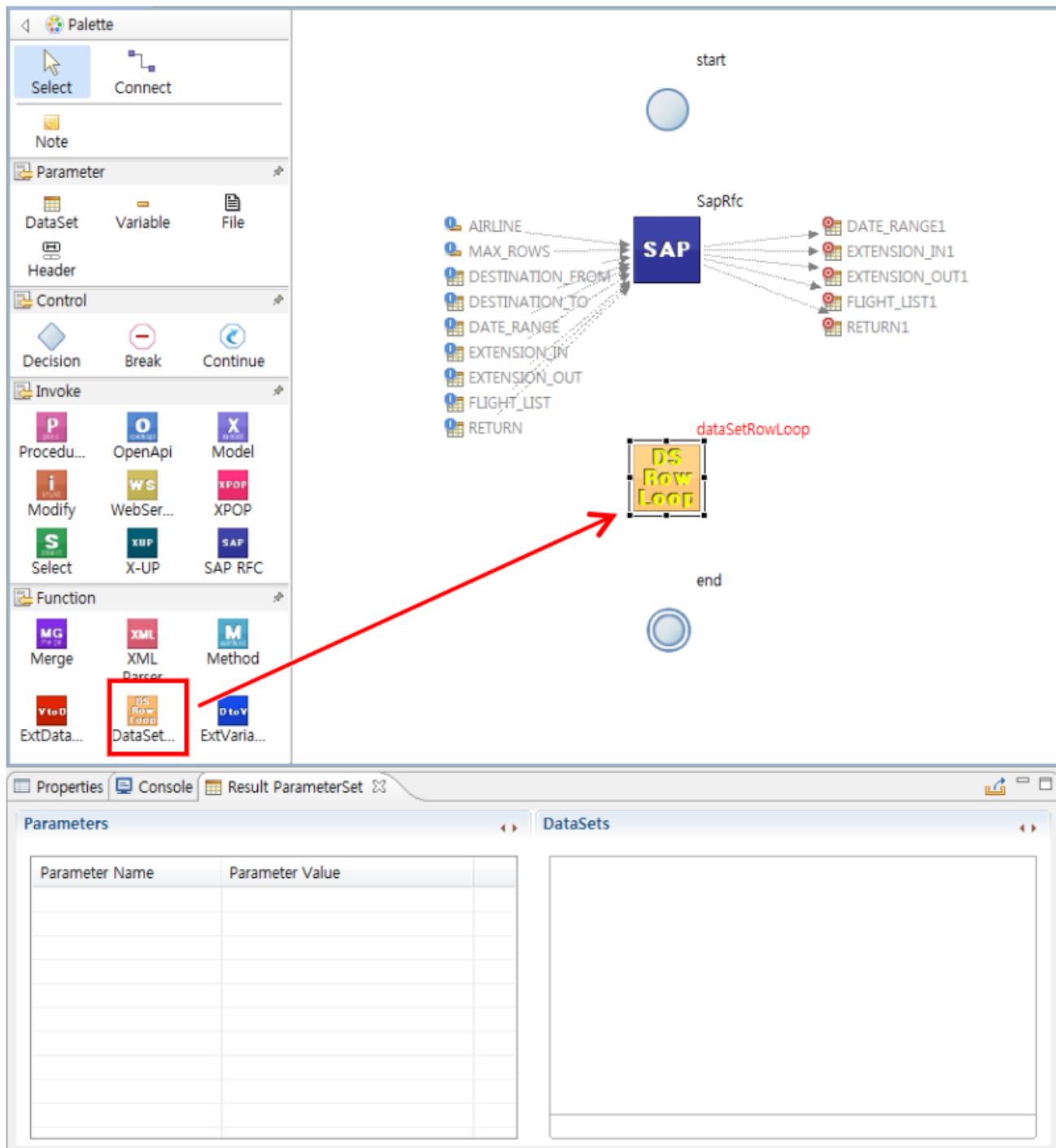


9. OK 버튼을 클릭 할 경우 입력 파라미터와 출력 파라미터가 생성이 된 것을 확인 할 수 있습니다.

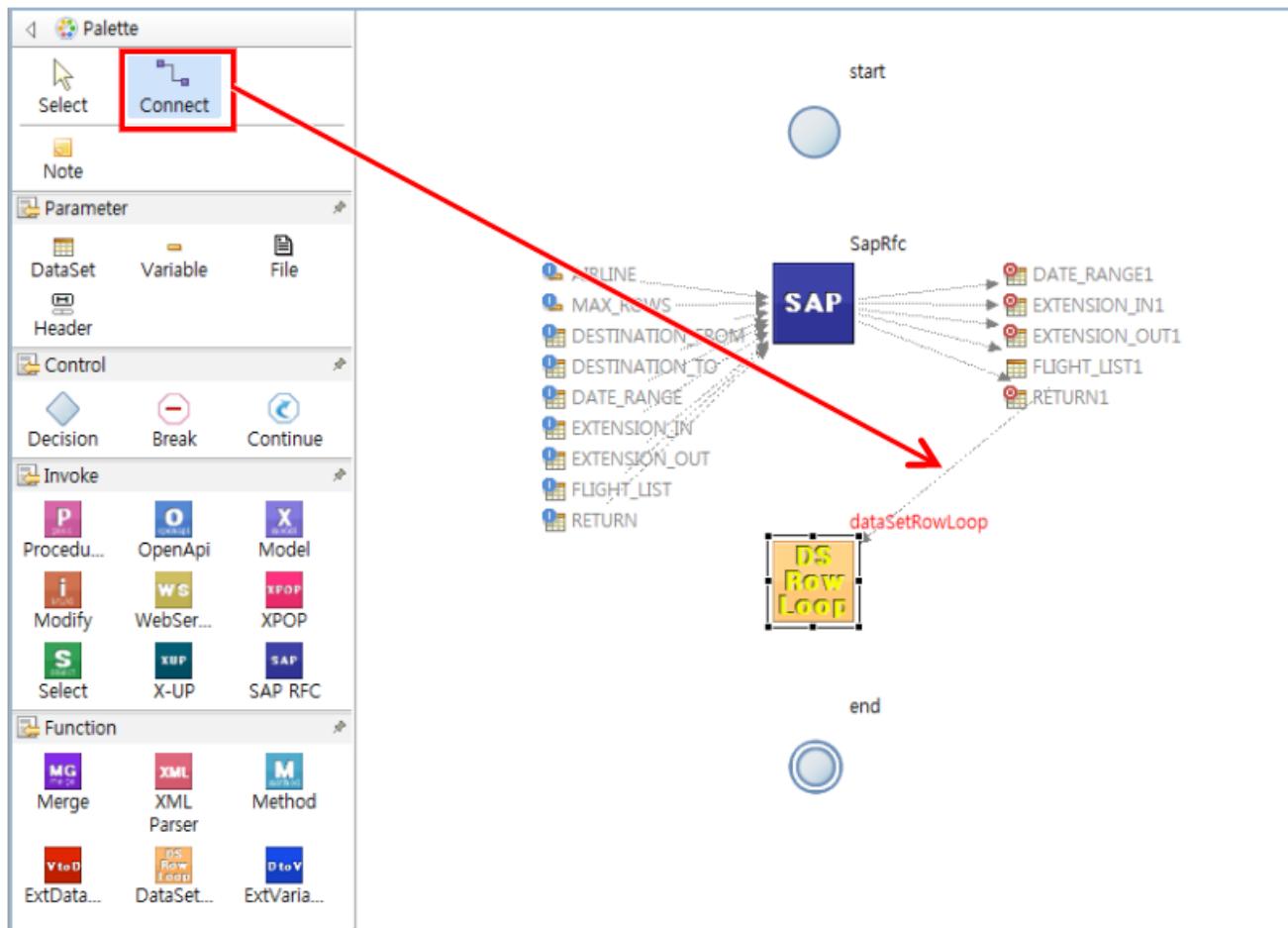


DataSetRowLoop 함수 생성하고 로직 설정하기

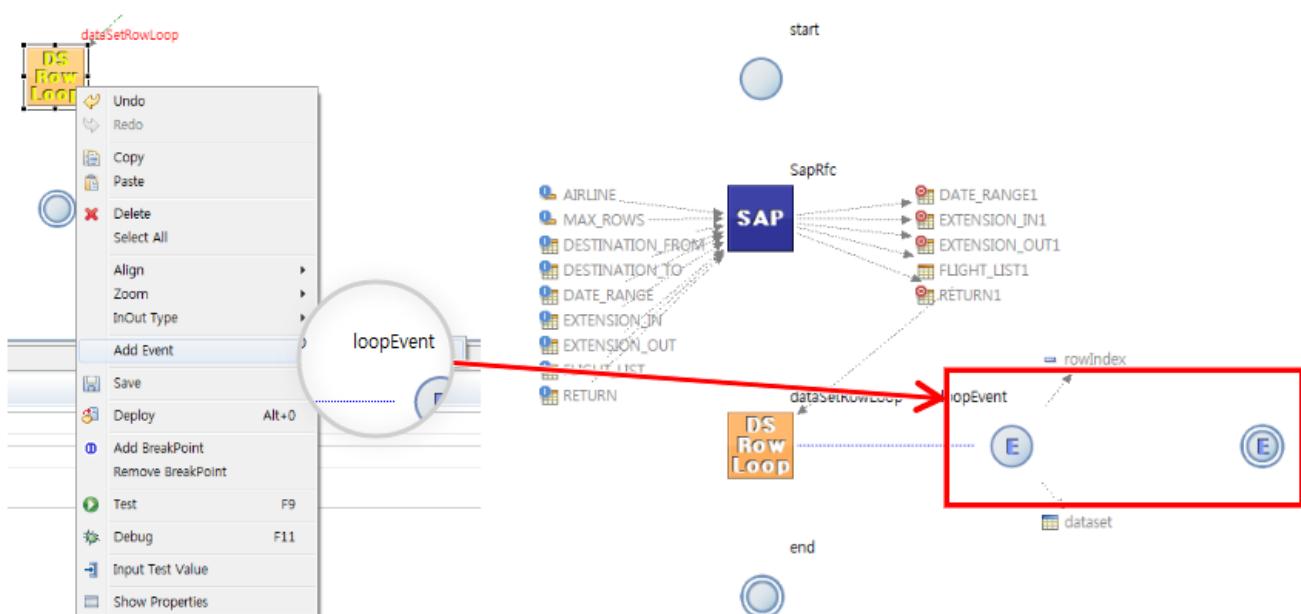
- Palette의 Function중 'DataSetRowLoop'를 선택하고 에디터를 클릭하게 되면 DataSetRowLoop 함수가 생성 됩니다.



- SAP RFC Invoke의 output 파라미터인 'FLIGHT_LIST1'을 Connect를 이용하여 DataSetRowLoop 함수의 in put 파라미터로 연결시켜줍니다.

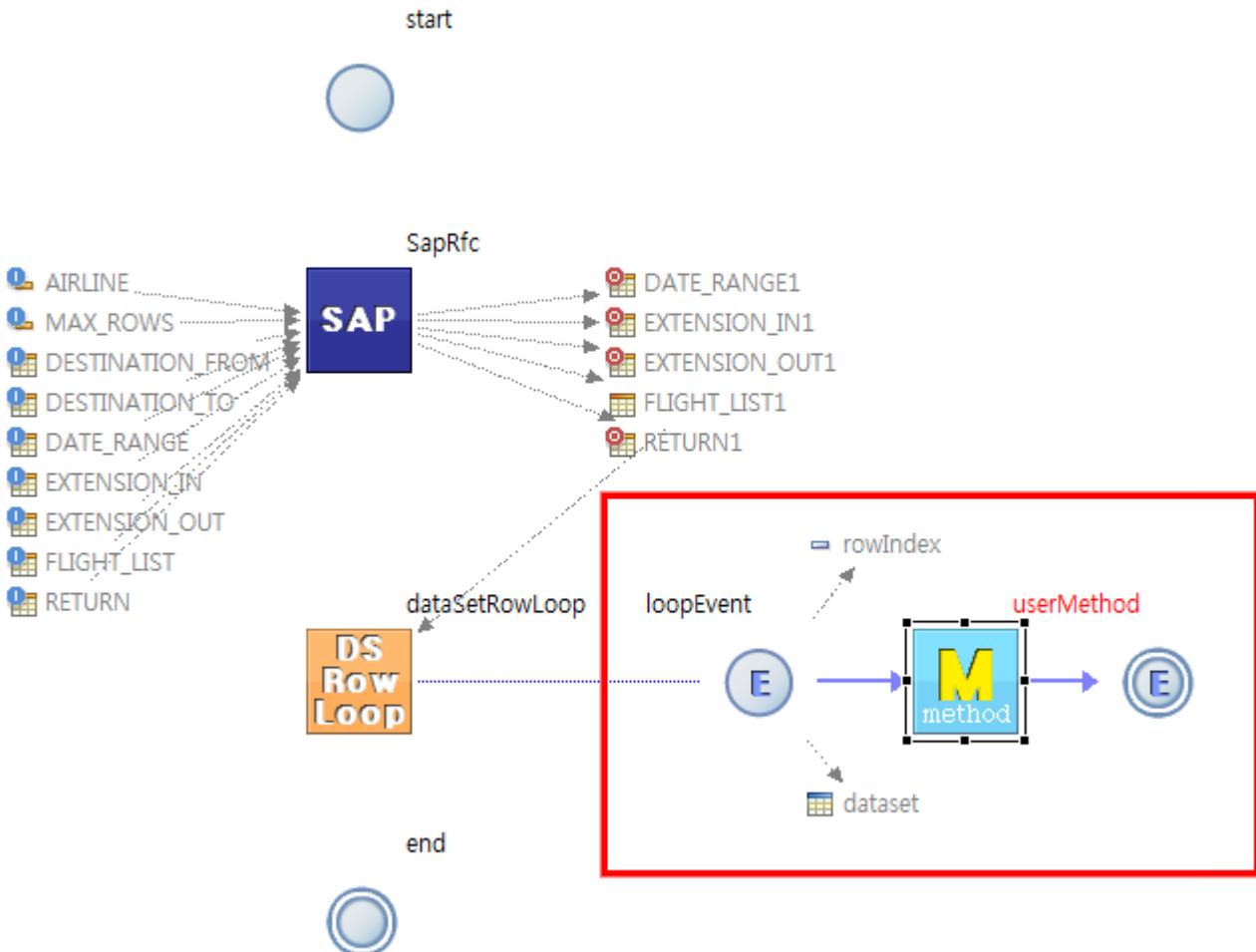


3. DataSetRowLoop 함수를 마우스 우클릭하여 Add Event loopEvent 선택하여 이벤트를 추가합니다.

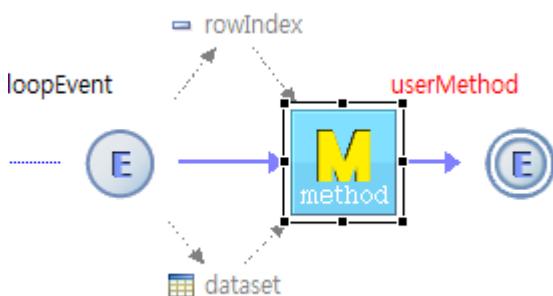


UserMethod Function을 이용하여 출력 파라메터 설정하기

- 이벤트 사이에 **UserMethod** 함수를 추가하고 Connect를 이용하여 이벤트와 UserMethod 함수를 연결시켜줍니다.



- 이벤트의 output으로 자동생성 된 **rowIndex**와 **dataset** 파라메터를 Connect를 이용하여 UserMethod 함수의 input 파라메터로 연결시켜줍니다.



- UserMethod 함수를 더블클릭하여 자동이동된 로직 클래스에서 다음과 같이 소스코드를 입력후 저장합니다.
(* DataTypes부분에서 에러가 난다면, com.tobesoft.xplatform.data.datatype.DataType 클래스를 import 해줍니다.)

```

public void userMethod(ParameterSet globalParameterSet, EuserMethodParameters eps) throws
AutomationFailException {
    DataSet result = globalParameterSet.getDataSet("result");
    if(result == null) {
        result = new DataSet("result");
        result.addColumn("AIRLINE", DataTypes.STRING);
        result.addColumn("CONNECTIONID", DataTypes.INT);
        result.addColumn("FLIGHTDATE", DataTypes.DATE);
    }

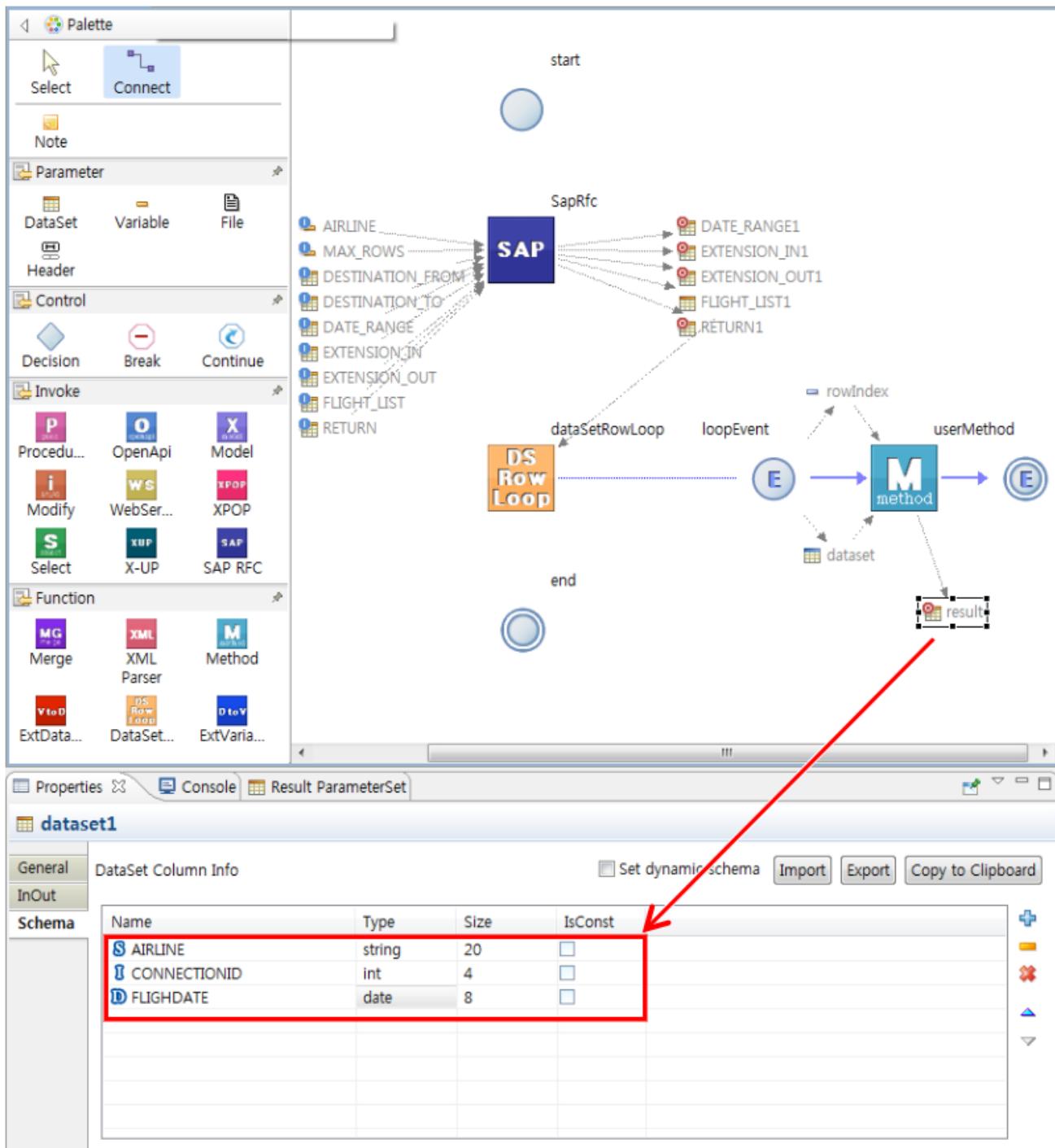
    DataSet dataset = eps.getDataSet();
    int month = (dataset.getDateTime(0, "FLIGHTDATE").getMonth());
    if(month == 4) {
        int row = result.newRow();
        result.set(row, "AIRLINE", dataset.getString(0, "AIRLINE"));
        result.set(row, "CONNECTIONID", dataset.getInt(0, "CONNECTID"));
        result.set(row, "FLIGHTDATE", dataset.getString(0, "FLIGHTDATE"));

        globalParameterSet.add("result", result);
    }
}

```

4. UserMethod함수의 output 파라메터로DataSet 파라메터 하나를 추가합니다. 이름을 result로 바꾸고 스키마를 아래와 같이 합니다. UserMethod함수와 Connect 해 줍니다.

Name	Type	Size
AIRLINE	string	20
CONNECTIONID	num	4
FLIGHTDATE	date	8



모델 테스트하기

- 모델 에디터에서 Palette의 **Connect**를 선택하여 Start 노드와 SAP RFC Invoke, DataSetRowLoop 함수, End 노드를 연결 해 줍니다.
- 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 **Test**를 선택하면 모델을 테스트 할 수 있습니다.

The screenshot shows the X-UP development environment interface.

Palette (Left):

- Select:** Active tool.
- Connect:** Tool for connecting components.
- Note:** Note-taking feature.
- Parameter:** Category for DataSet, Variable, and File.
- Header:** Header component.
- Control:** Category for Decision, Break, Continue, Invoke, and Function blocks.
- Invoke:** Sub-category for Procedure, OpenApi, Model, Modify, WebSer..., XPOP, Select, X-UP, and SAP RFC.
- Function:** Category for Merge, XML Parser, Method, VtoD, DataSet..., ExtVaria..., and ExtData... blocks.

Diagram (Center):

```

graph TD
    start((start)) --> SAP[SAP]
    SAP -- SapRfc --> DS[DS Row Loop]
    DS -- dataSetRowLoop --> E((E))
    E -- rowIndex --> M[M method]
    M -- result --> end((end))
    
```

The diagram illustrates a workflow starting from a 'start' node, connecting to a SAP component via 'SapRfc'. The SAP component then feeds into a 'DS Row Loop' (orange diamond). From the 'DS Row Loop', the flow goes to an 'E' event node, which then connects to a 'M method' (blue rectangle). Finally, the 'M method' leads to an 'end' node.

Context Menu (Top Right):

- Undo
- Redo
- Copy
- Paste
- Delete
- Select All
- Show All with Alias
- Align
- Zoom
- InOut Type
- Save
- Save to image file
- Deploy Alt+O
- Import Parameters
- Export Parameters
- Validate
- Add BreakPoint
- Remove BreakPoint
- Test F9** (highlighted with a red box)
- Debug F11
- Input Test Value
- Show Properties

Properties (Bottom Left):

Properties for 'sapTest' node:

- General: Visible, show all with alias checked.
- Import: Description field is empty.
- Charset: UTF-8.

Console (Bottom Middle):

AIRLINE	CONNECTID	FLIGHTDATE
American Airl...	17	20130410
American Airl...	17	20140409
American Airl...	64	20130412
American Airl...	64	20140411

EXTENSION_OUT1 RETURN1 result >>2

Result ParameterSet (Bottom Right):

Parameter Name	Parameter Value

8.9 Merge Function을 이용하여 데이터 가공하기

이 절에서는 복수의 모델로부터 얻어진 데이터셋들을 융합하여 새로운 데이터셋을 생성하는 Merge 함수에 대해 설명합니다.

이 절에서 설명하는 융합 모델 개발단계는 다음과 같습니다.

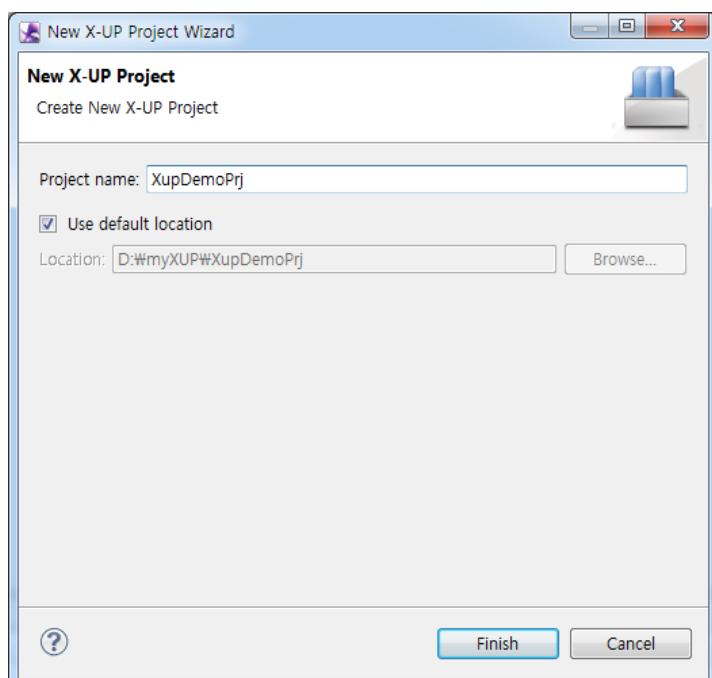
- X-UP 프로젝트 생성하기
- Automation 모델 생성 및 Merge 함수 생성 하기
- 융합 할 데이터셋 생성 하기
- 융합 로직 설정하기
- Merge 테스트 하기
- 모델 테스트하기



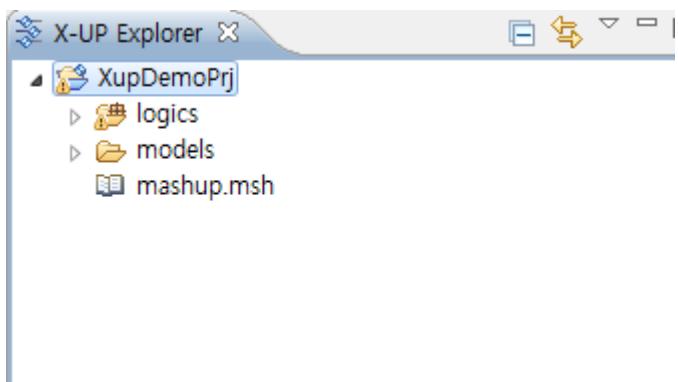
X-UP은 내부적으로 메모리 데이터베이스 시스템을 사용하여 융합 모델을 구현하고 있습니다. 따라서 대량의 데이터를 출력하는 모델들을 융합할 경우 X-UP서버가 운영되는 WAS의 메모리 및 성능에 영향을 미칠 수 있습니다.

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 선택합니다.

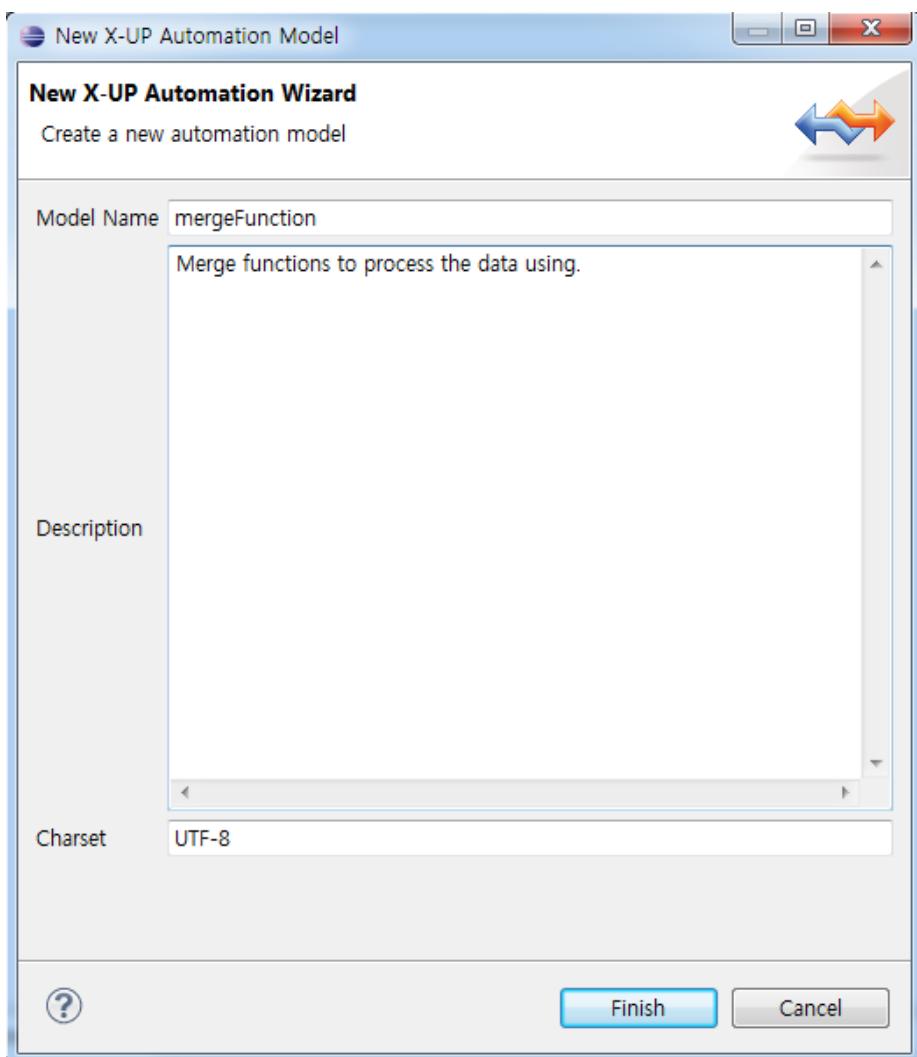


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

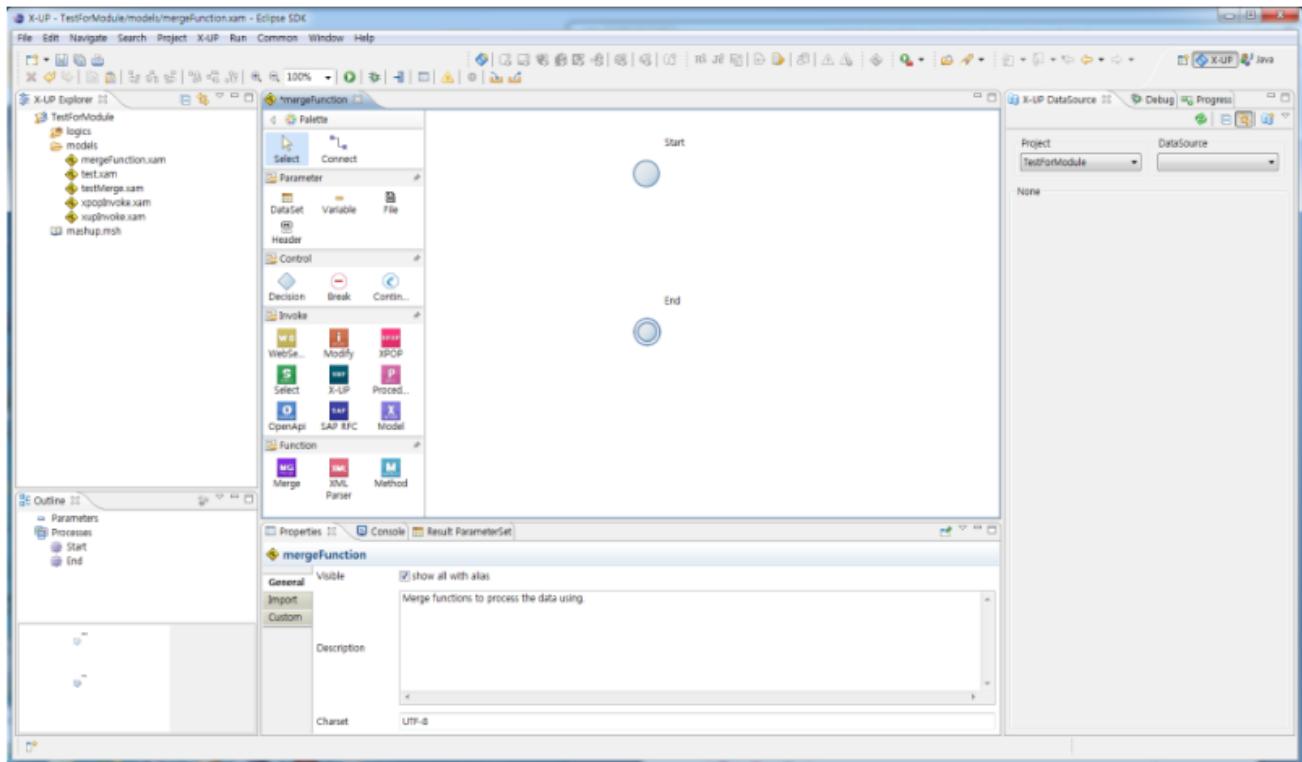


Automation 모델 생성 및 Merge Function 생성하기

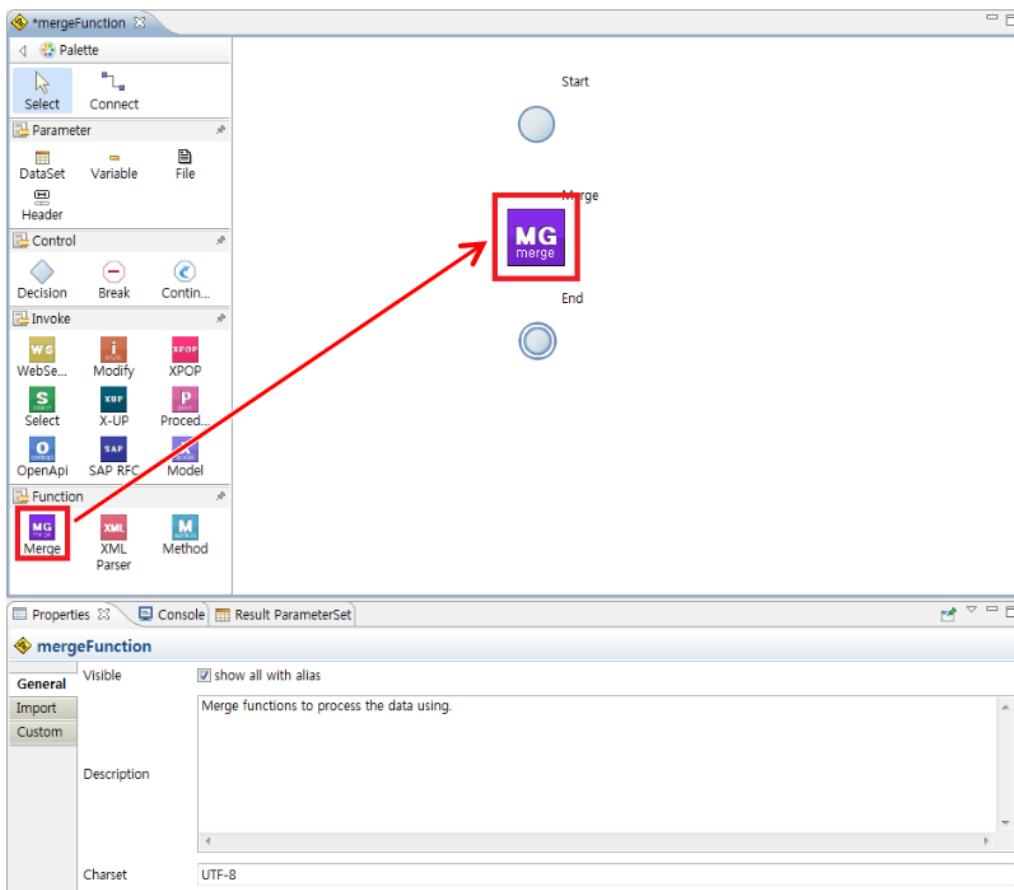
1. [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 'Finish' 버튼을 클릭합니다.



3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.

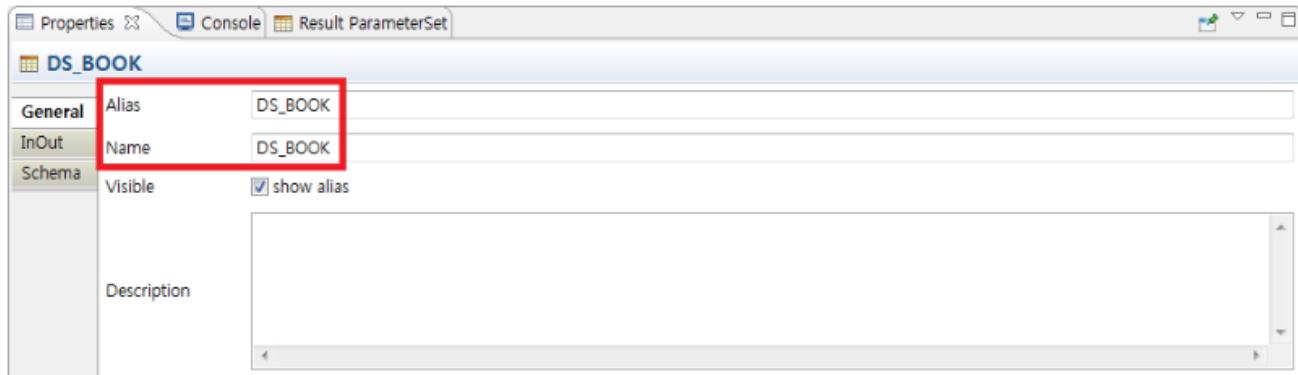


4. Palette의 Function 중 ‘Merge’를 선택하고 에디터를 클릭 하게 되면 Merge 함수가 생성 됩니다.



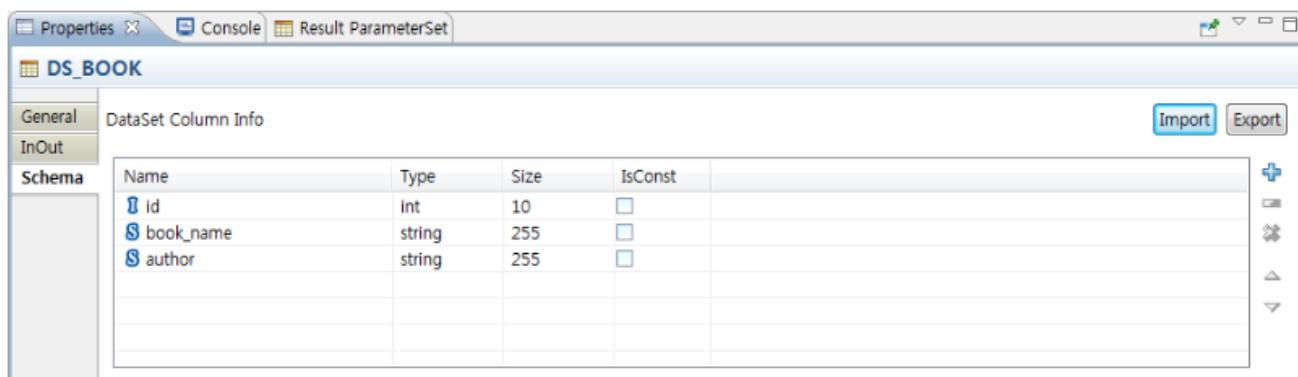
융합 할 데이터셋 생성하기

- 화면 Palette에서 'DataSet'을 선택 하여 에디터를 클릭 합니다.
- 생성된 데이터셋의 기본 명칭은 'dataset' 입니다. 이름과 Alias를 DS_BOOK 으로 변경합니다.



- Properties의 Schema에서 [+]를 클릭하여 Column 정보를 다음과 같이 추가 합니다.

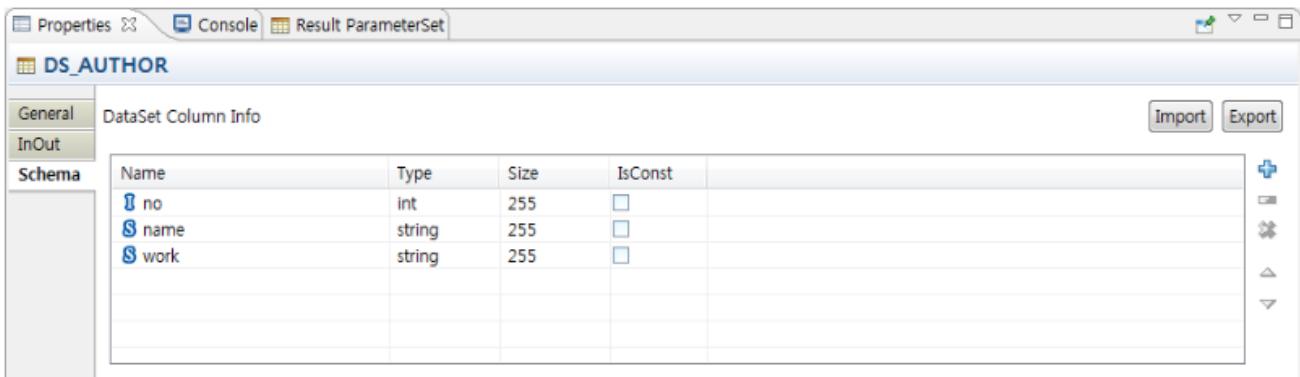
Dataset Name	DS_BOOK		
Column Info	Name	Type	Size
	id	int	10
	book_name	string	255
	author	string	255



- DS_BOOK과 Merge 함수를 Palette의 Connect를 이용하여 연결 합니다.
- 위와 같은 방법으로 DataSet 하나를 더 생성하여 이름과 Alias를 DS_AUTHOR로 변경합니다.
- Properties의 Schema에서 [+]를 클릭하여 Column 정보를 다음과 같이 추가 합니다.

Dataset Name	DS_AUTHOR		
Column Info	Name	Type	Size
	no	int	10
	name	string	255

	work	string	255
--	------	--------	-----



7. DS_AUTHOR와 Merge 함수를 Palette의 Connect를 이용하여 연결 합니다.

융합로직 설정하기

1. 에디터의 Merge 함수(1)를 더블클릭 하거나 ‘Properties’ 탭을 클릭하여 ‘Merge icon’[MG] (2)을 클릭하면 위자드가 실행 됩니다.

The screenshot shows the X-UP Automation editor interface. On the left is the Palette, which includes sections for Select, Connect, Parameter (DataSet, Variable, File), Control (Decision, Break, Continue...), Invoke (WebSe..., Modify, XPOP, Select, X-UP, Procedure..., OpenApi, SAP RFC, Model), and Function (Merge, XML Parser, Method). A 'Merge' icon from the Function section is highlighted with a red box and labeled with a circled '1'. In the main workspace, a process flow is shown with a Start node, a 'Merge' icon, and an End node. Two dashed arrows point from DS_BOOK and DS_AUTHOR datasets to the Merge icon. The 'Properties' window for the 'merge' object is open at the bottom. The 'InvokeInfo' tab is selected, showing a single entry '1'. The 'Merge SQL' field is empty. The 'Merge Information' tab is also visible. A circled '2' highlights the 'Merge icon' button in the Properties window's toolbar.

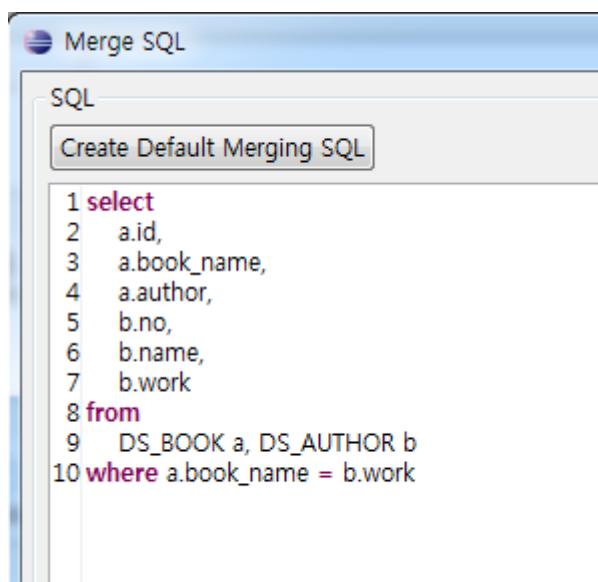
2. 'Create Default Merging SQL'을 클릭하면 기본적인 융합 로직이 SQL에디터에 생성됩니다.
3. SQL 에디터에 설정된 융합 로직을 아래와 같이 수정합니다.

Code 8-3 수정 전

```
select
    a.id,
    a.book_name,
    a.author,
    b.no,
    b.name,
    b.work
from
    DS_BOOK a, DS_AUTHOR b
```

Code 8-4 수정 후

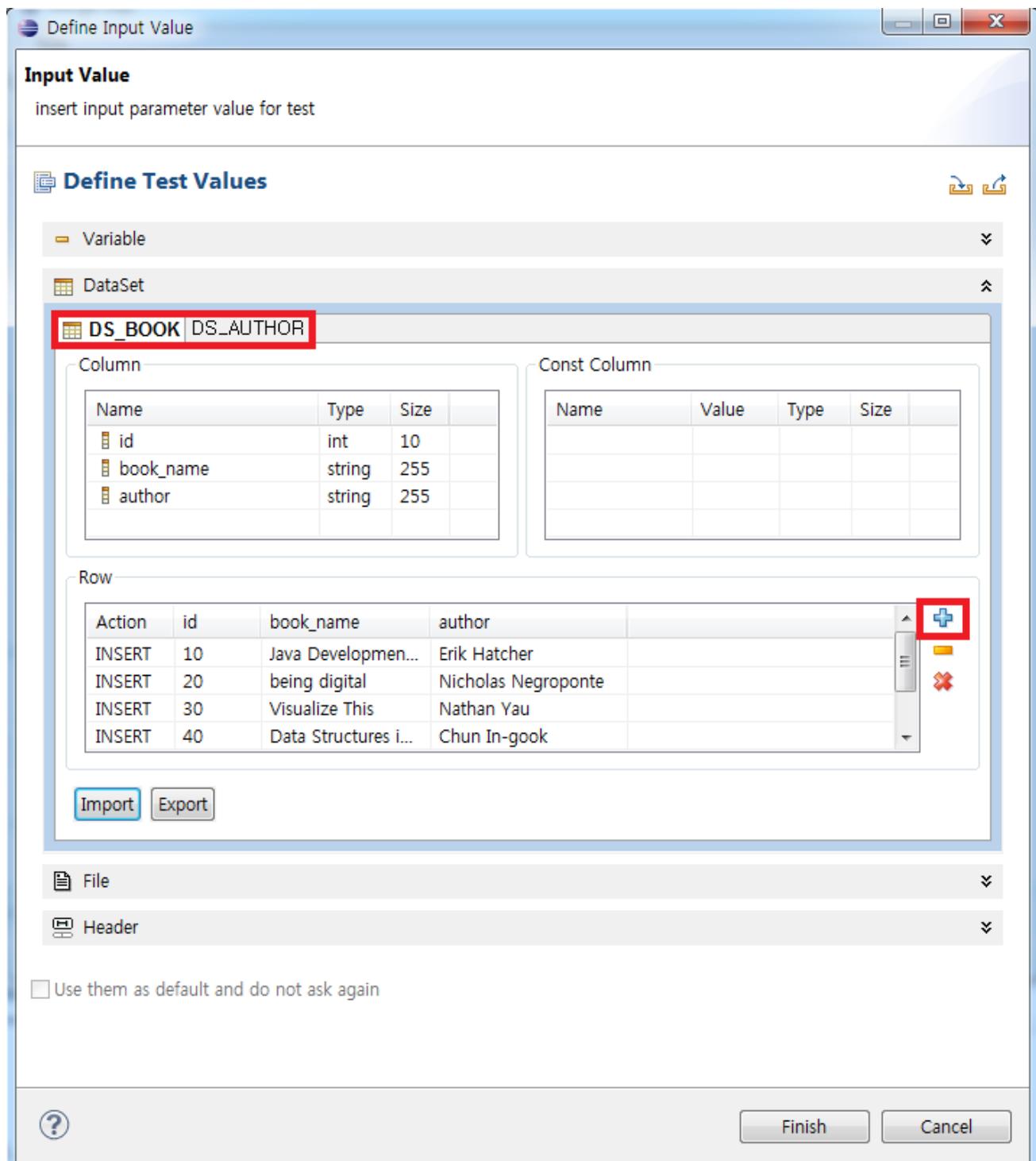
```
select
    a.id,
    a.book_name,
    a.author,
    b.no,
    b.name,
    b.work
from
    DS_BOOK a, DS_AUTHOR b
Where a.book_name = b.work
```

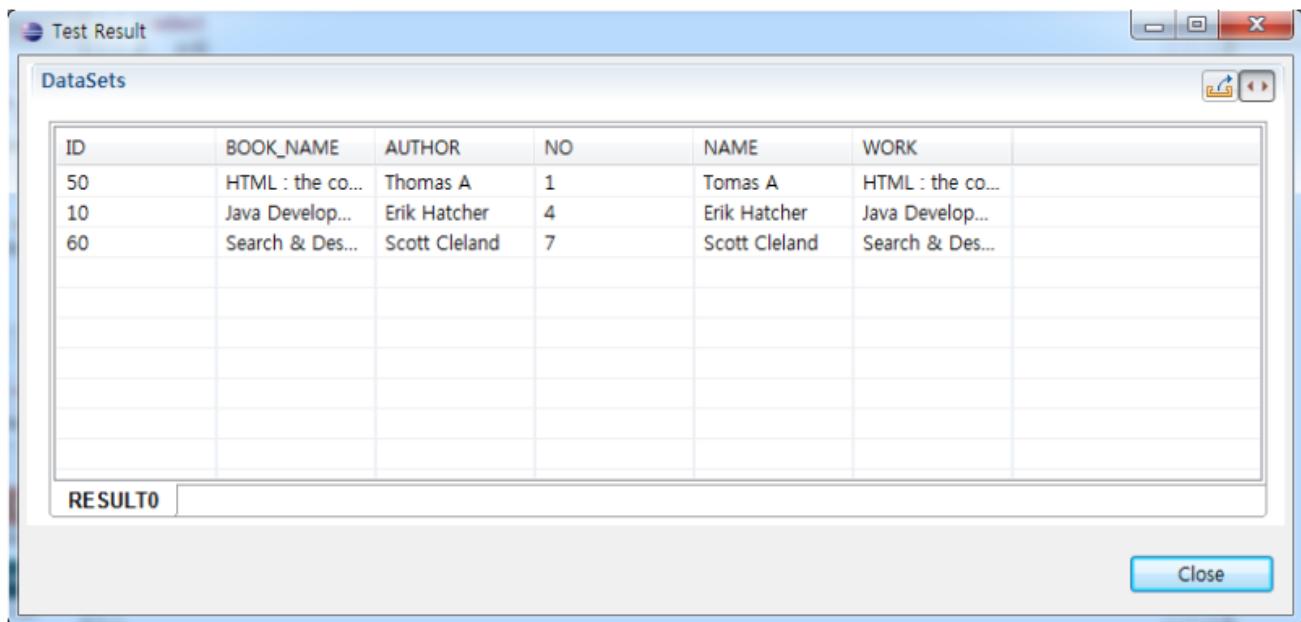


4. SQL 에디터의 Test 버튼을 클릭하여 융합 로직을 테스트 합니다.
5. Define Input Value에 다음의 값을 넣고 'Finish' 버튼을 클릭합니다.

DS_BOOK			
Action	id	book_name	author
INSERT	10	Java Development with Ant	Erik Hatcher
INSERT	20	being digital	Nicholas Negroponte
INSERT	30	Visualize This	Nathan Yau
INSERT	40	Data Structures in C	Chun In-gook
INSERT	50	HTML : the comlete reference	Thomas A
INSERT	60	Search & Destroy: Why You Can't Trust Google Inc	Scott Cleland

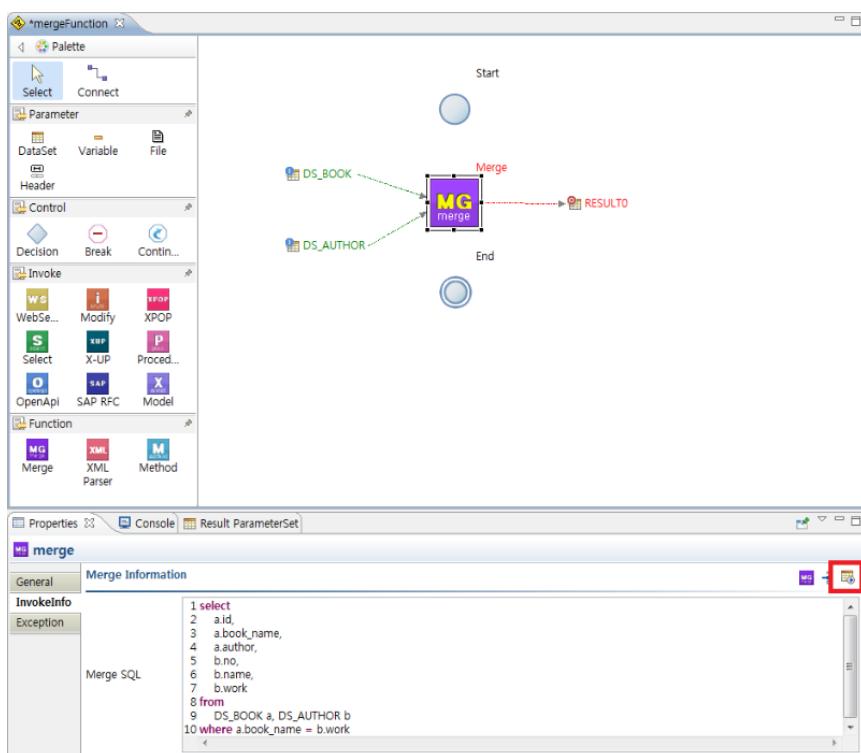
DS_AUTHOR			
Action	id	name	work
INSERT	1	Tomas A	HTML : the comlete reference
INSERT	2	Tomas A	Global Sourcing Logistics
INSERT	3	Tomas A	Elements Of Distribution Theory
INSERT	4	Erik Hatcher	Java Development wth Ant
INSERT	5	Erik Hatcher	Lucene in Action
INSERT	6	Susan M. Weinschenk	Neuro Web Design: What makes them click?)
INSERT	7	Scott Cleland	Search & Destroy: Why You Can't Trust Google Inc





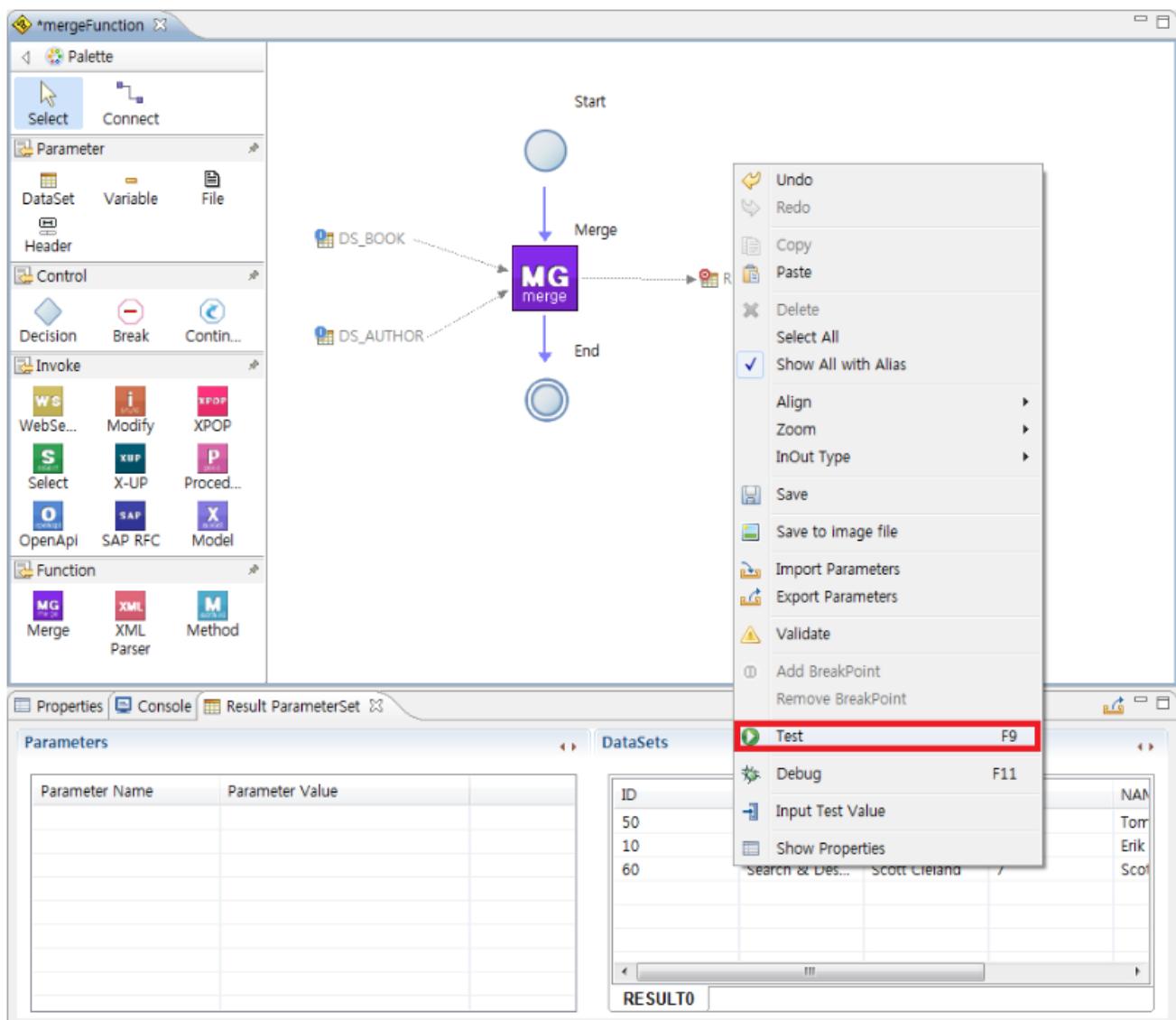
Function 테스트 하기

1. 테스트 결과를 바탕으로 출력 파라메터가 생성됩니다.
2. Merge 함수의 Properties의 'InvokeInfo'를 선택하고 [] 를 클릭하여 테스트를 진행합니다.
3. 위 '융합 로직 설정 하기'에서 SQL 에디터에서 Test 버튼을 선택했을 경우와 동일하게 값을 입력하고 테스트를 진행하게 되면 결과를 확인 할 수 있습니다.
4. 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인 할 수 있습니다.



모델 테스트하기

- 모델 에디터에서 Palette의 Connect를 선택하여 Start 노드와 Merge 함수, End 노드를 연결 해 줍니다.
- 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.
- 테스트 데이터는 ‘Function 테스트하기’ 와 동일한 테스트 데이터를 입력하여 테스트를 합니다.
- 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.



8.10 XML Parser Function을 이용하여 데이터 가공하기

XML 형태의 데이터를 (주)투비소프트에서 제공하는 데이터 타입인 데이터셋 형태나 Primitive 타입의 데이터로 손쉬운 가공 방법을 제공하고 있습니다.

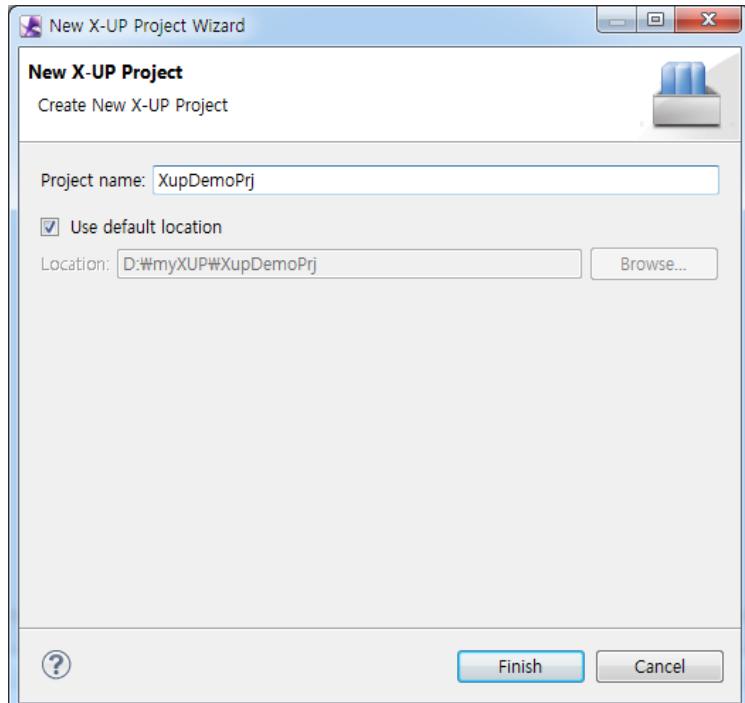
XML Parser를 이용하여 [8.5 OpenApi Invoke를 이용한 모델 개발하기](#)의 출력 결과를 데이터셋으로 가공하는 개발 방법을 설명 합니다.

이 절에서 설명하는 Automation 모델의 XML Parser를 이용한 개발단계는 다음과 같습니다.

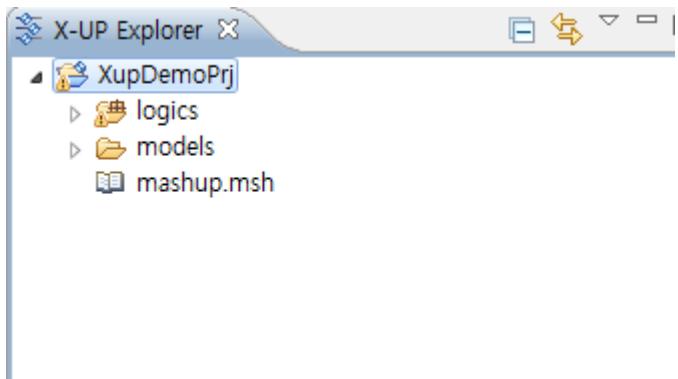
- X-UP 프로젝트 생성하기
- Automation 모델을 생성 및 XML Parser 생성 하기
- 입력 파라메터 설정 및 출력 인터페이스 생성 하기
- XML Parser 테스트 하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭합니다.

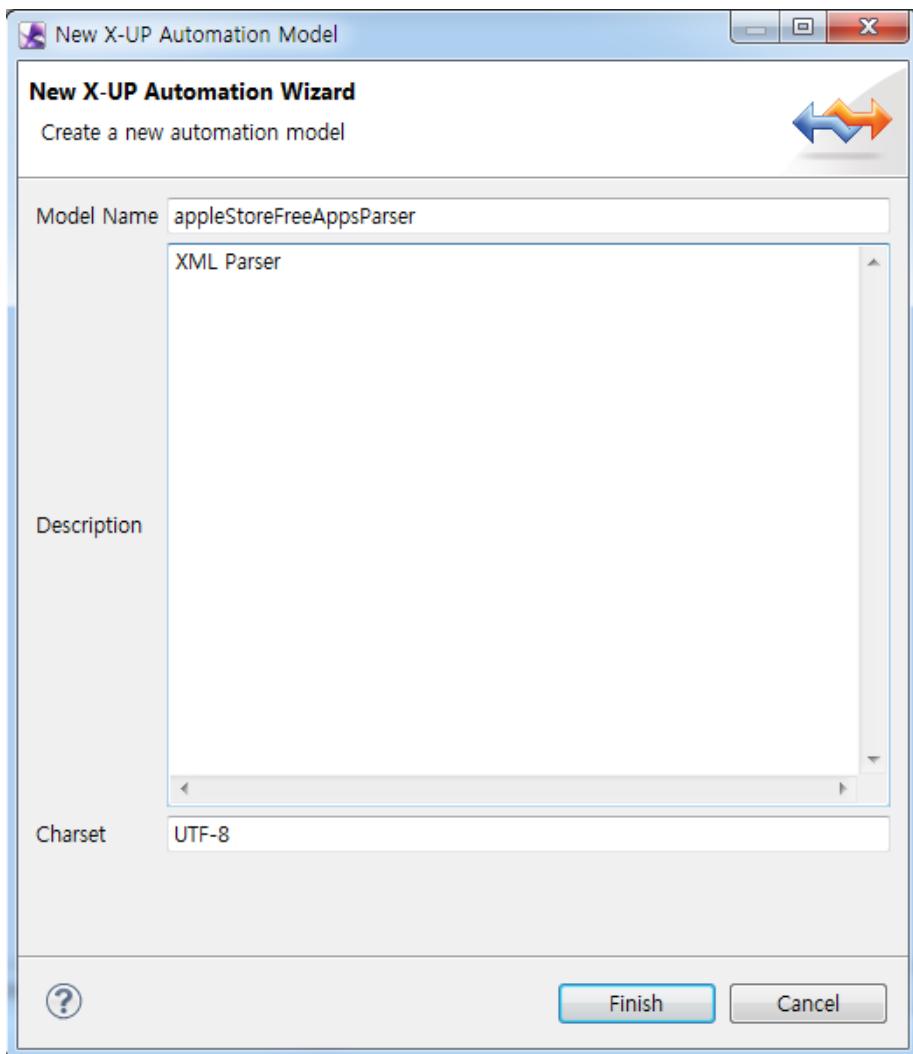


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

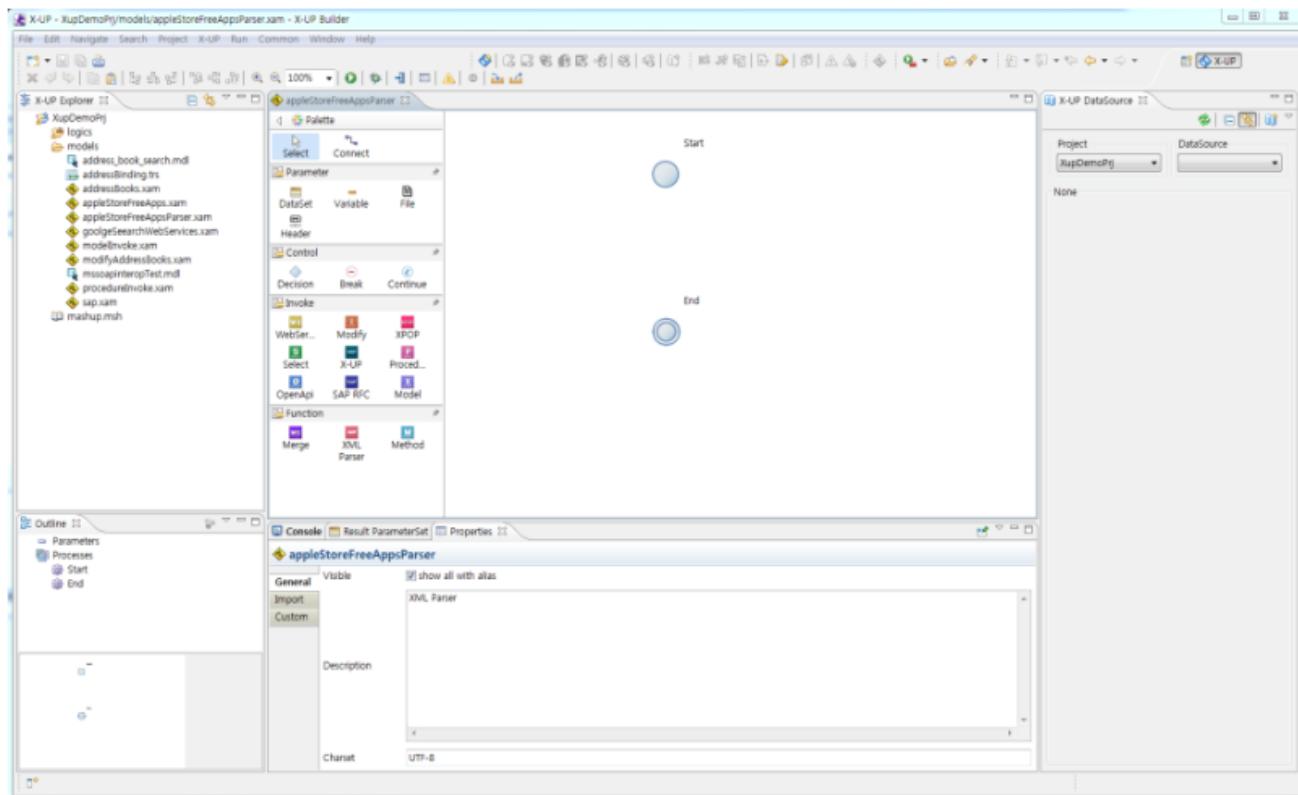


Automation 모델 생성 및 XML Parser Function 생성하기

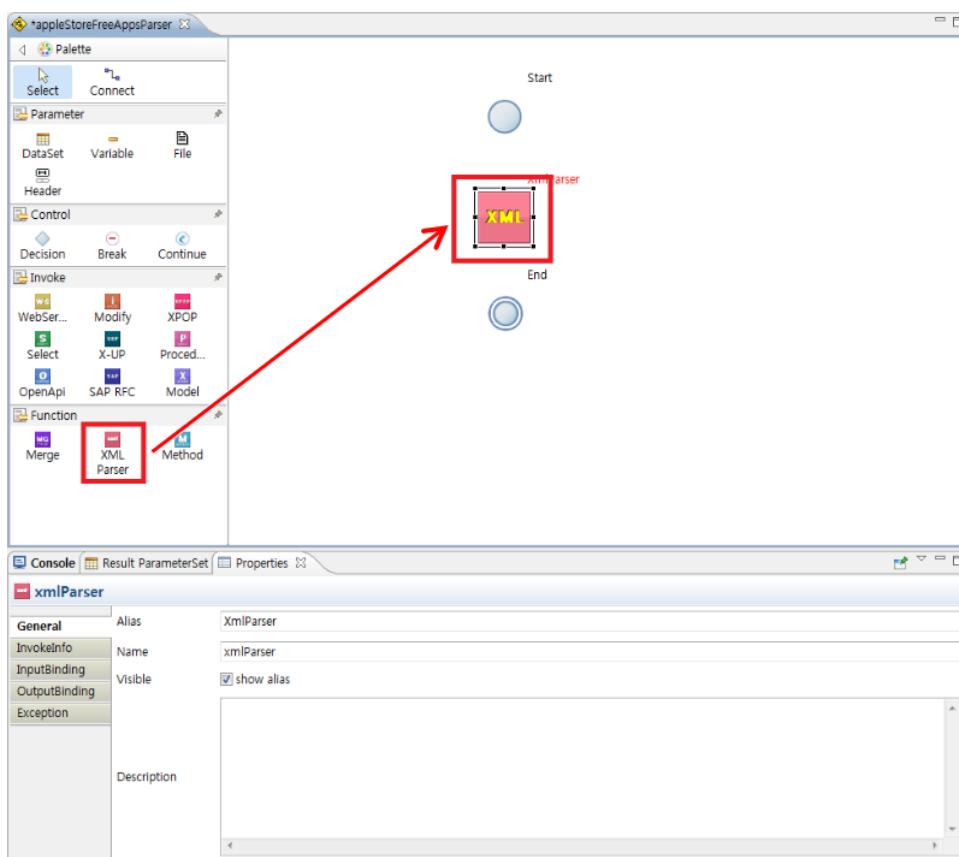
1. [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 원하는 모델 이름을 입력하고 OK 버튼을 클릭합니다.



3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.

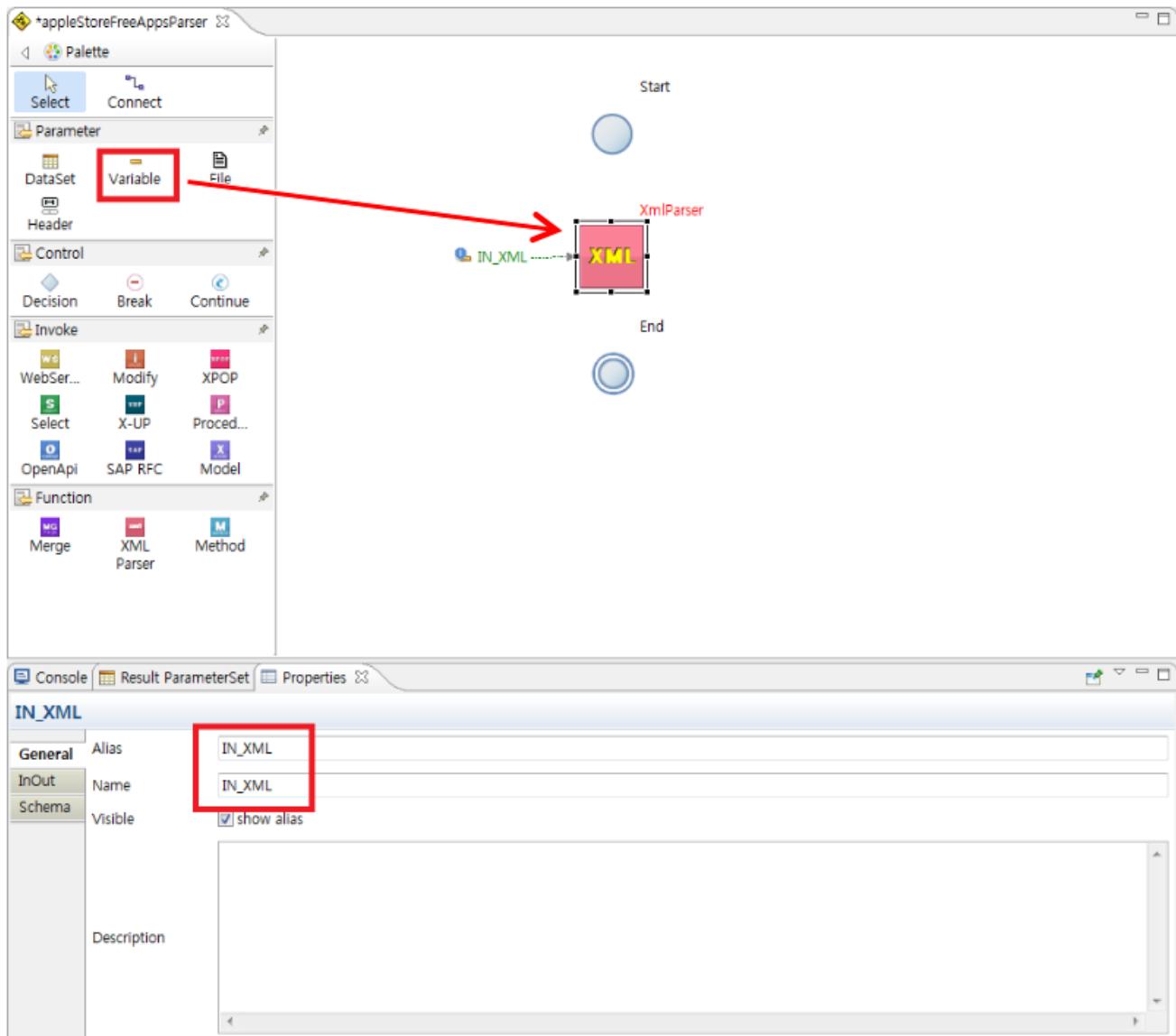


4. Palette 의 Function 중 'XML Parser' 를 선택하고 에디터를 클릭 하게 되면 XML Parser가 생성 됩니다.

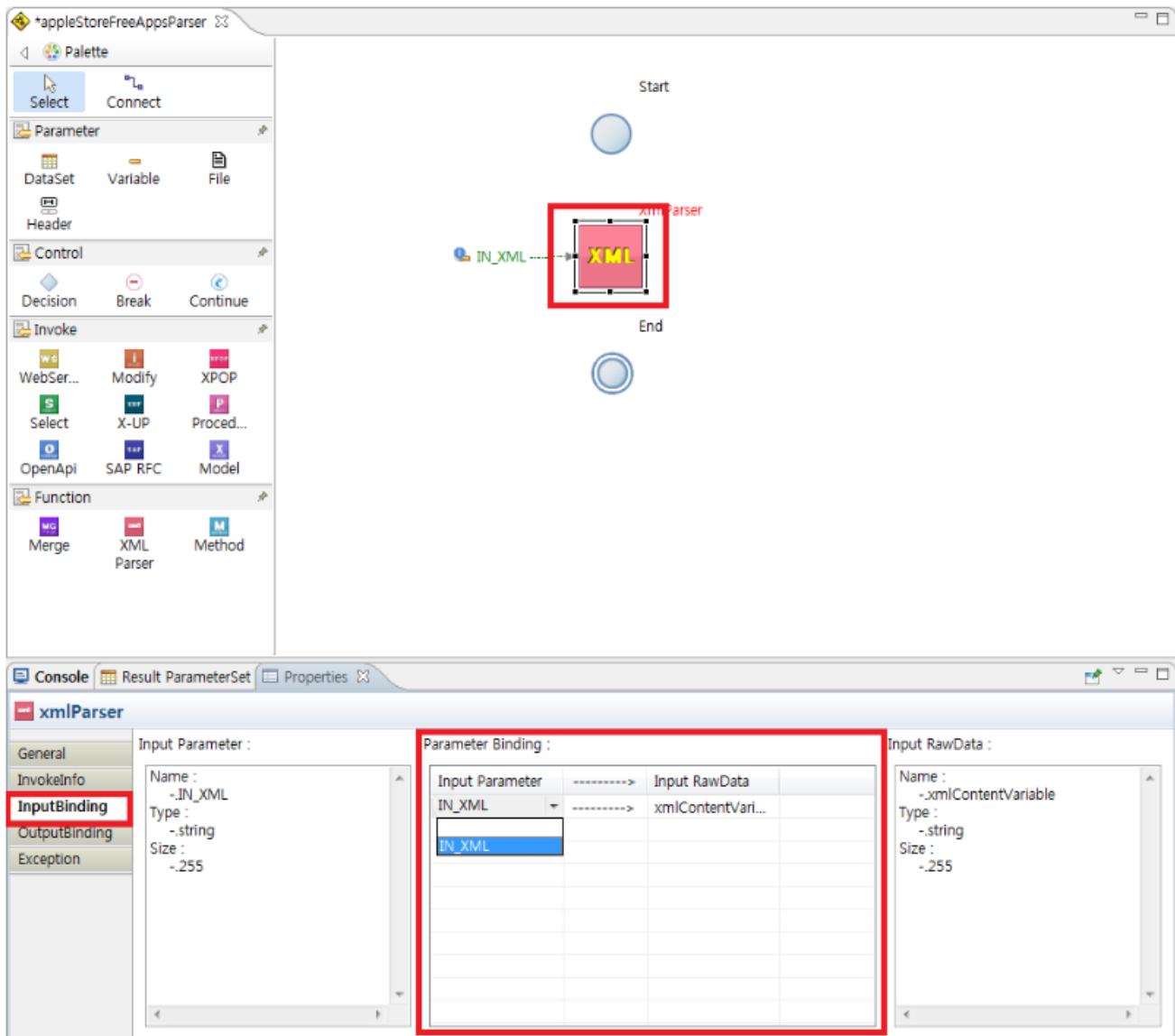


입력 파라미터 설정 및 출력 인터페이스 생성하기

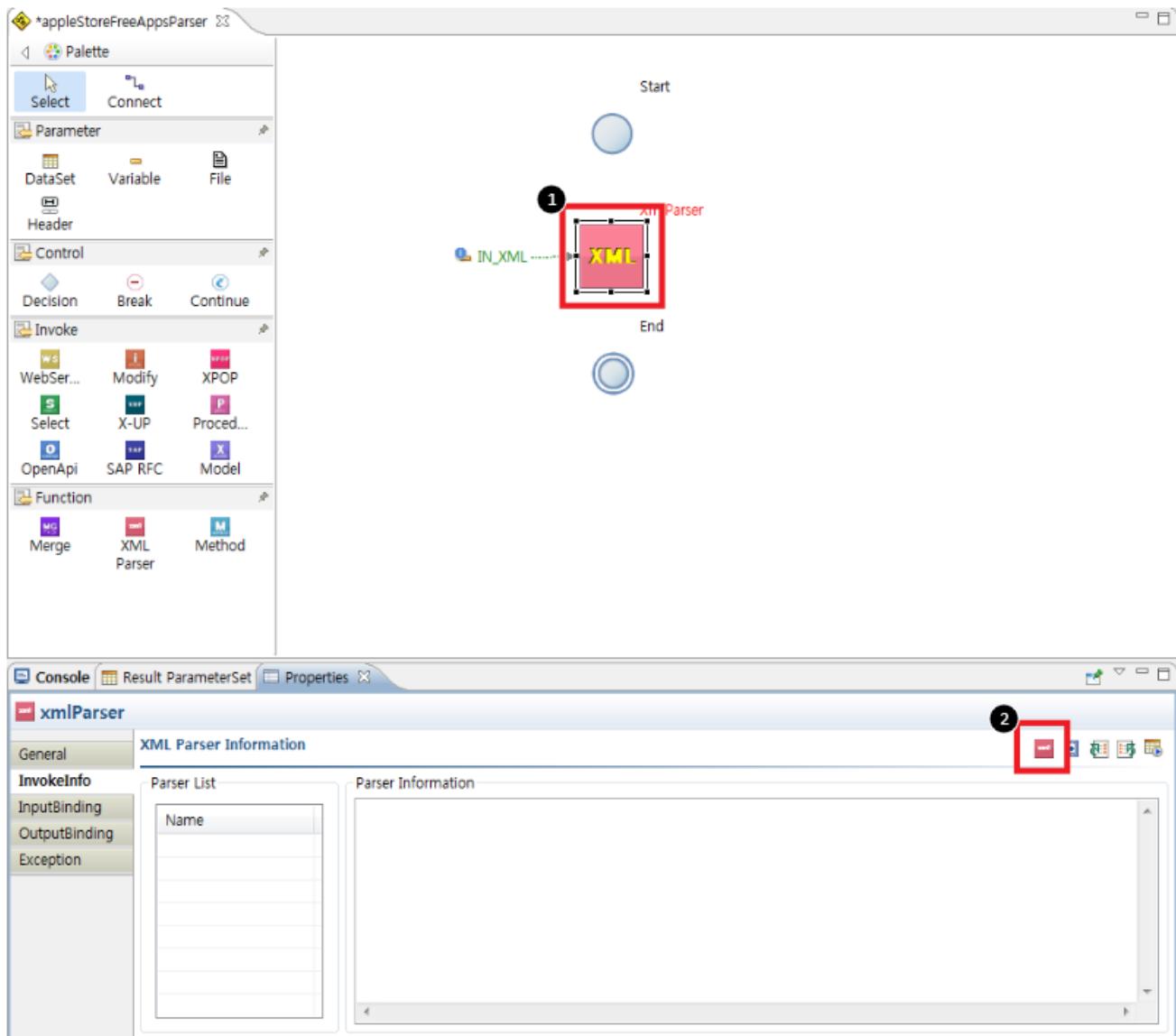
- 화면 Palette에서 'Variable'을 선택 하여 에디터를 클릭 합니다.
- 생성 된 Variable의 기본 명칭은 'variable' 입니다. 이름과 Alias를 IN_XML로 변경합니다.
- IN_XML 과 XML Parser를 연결선을 이용하여 연결 합니다.



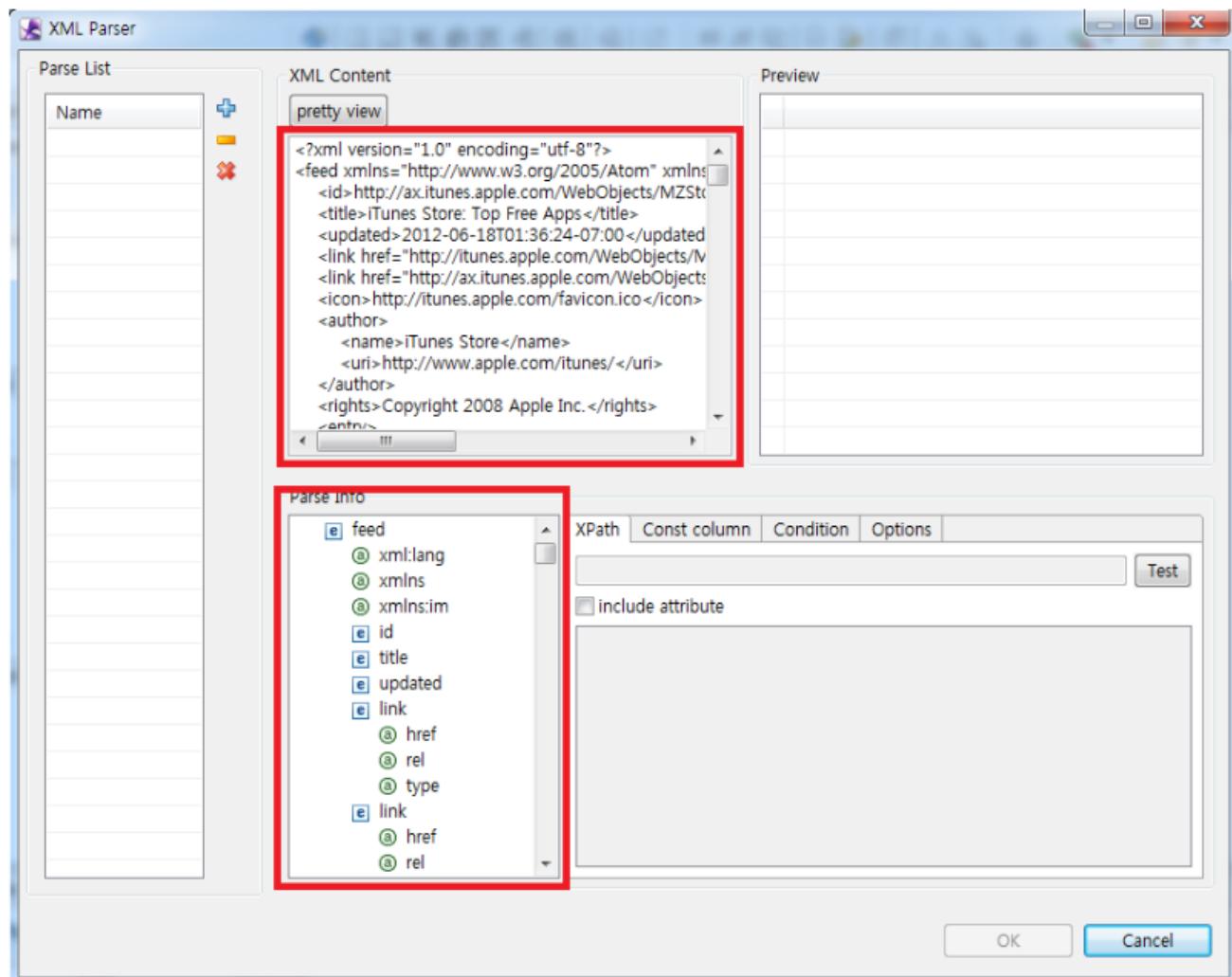
- 연결선을 연결 하면 다른 선과는 다르게 화살표의 색깔이 빨간색이며 BODY와 연결 된 부분 앞쪽에 빨간색 선이 그어져 있는 게 보입니다. (입력 파라미터의 Binding 처리가 되어 있지 않은 것입니다.)
- XML Parser의 'Properties'의 'InputBinding' 탭을 클릭하여 'Input Parameter'와 'Input RawData'의 Parameter Binding 을 시켜 줍니다.



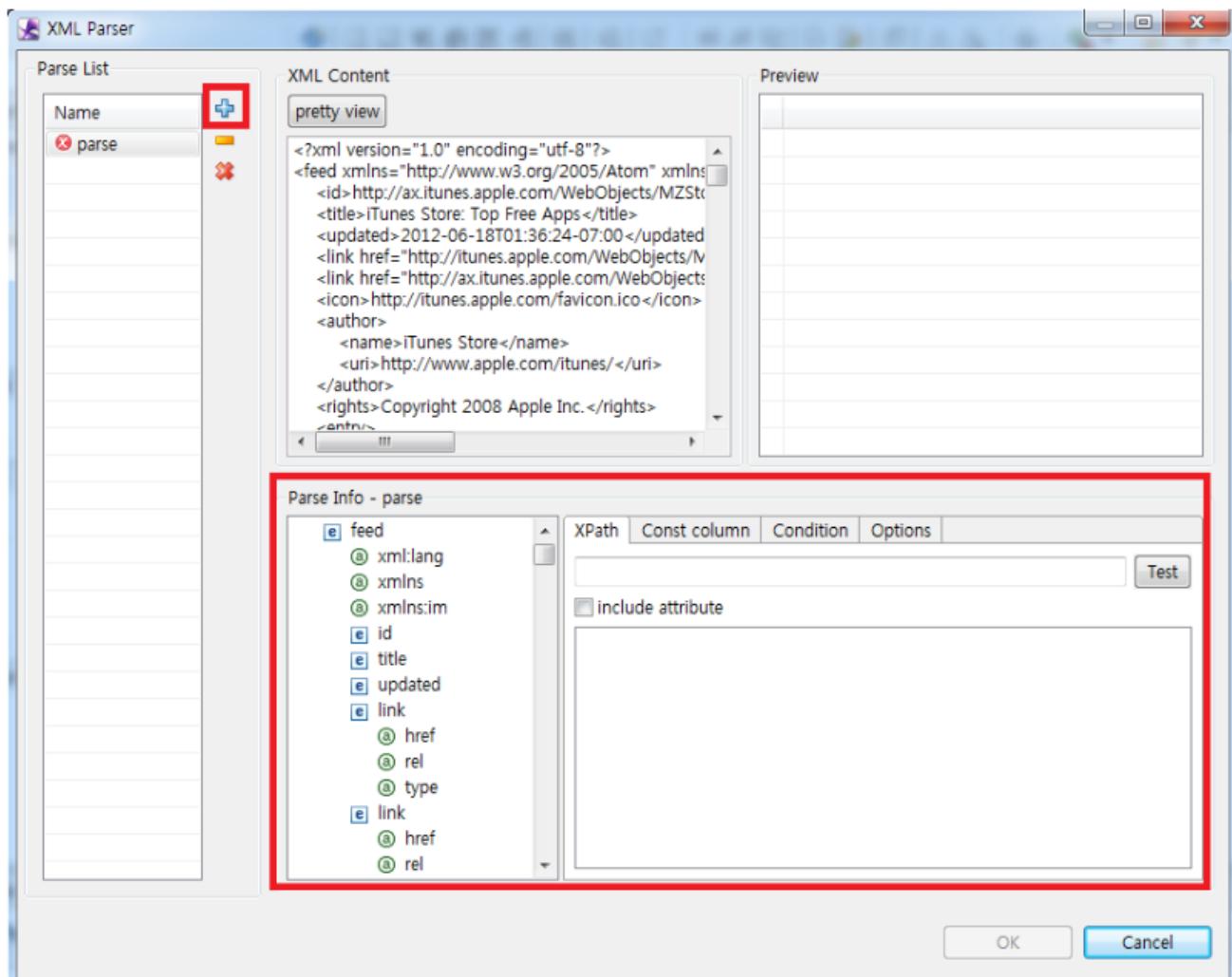
6. XML Parser를(1) 더블클릭 하거나 ‘Properties’ 탭의 ‘ParserInfo’를 클릭하여 ‘XML Parser icon’[](2)을 클릭하여 XML Parser 위자드를 실행 시킵니다.



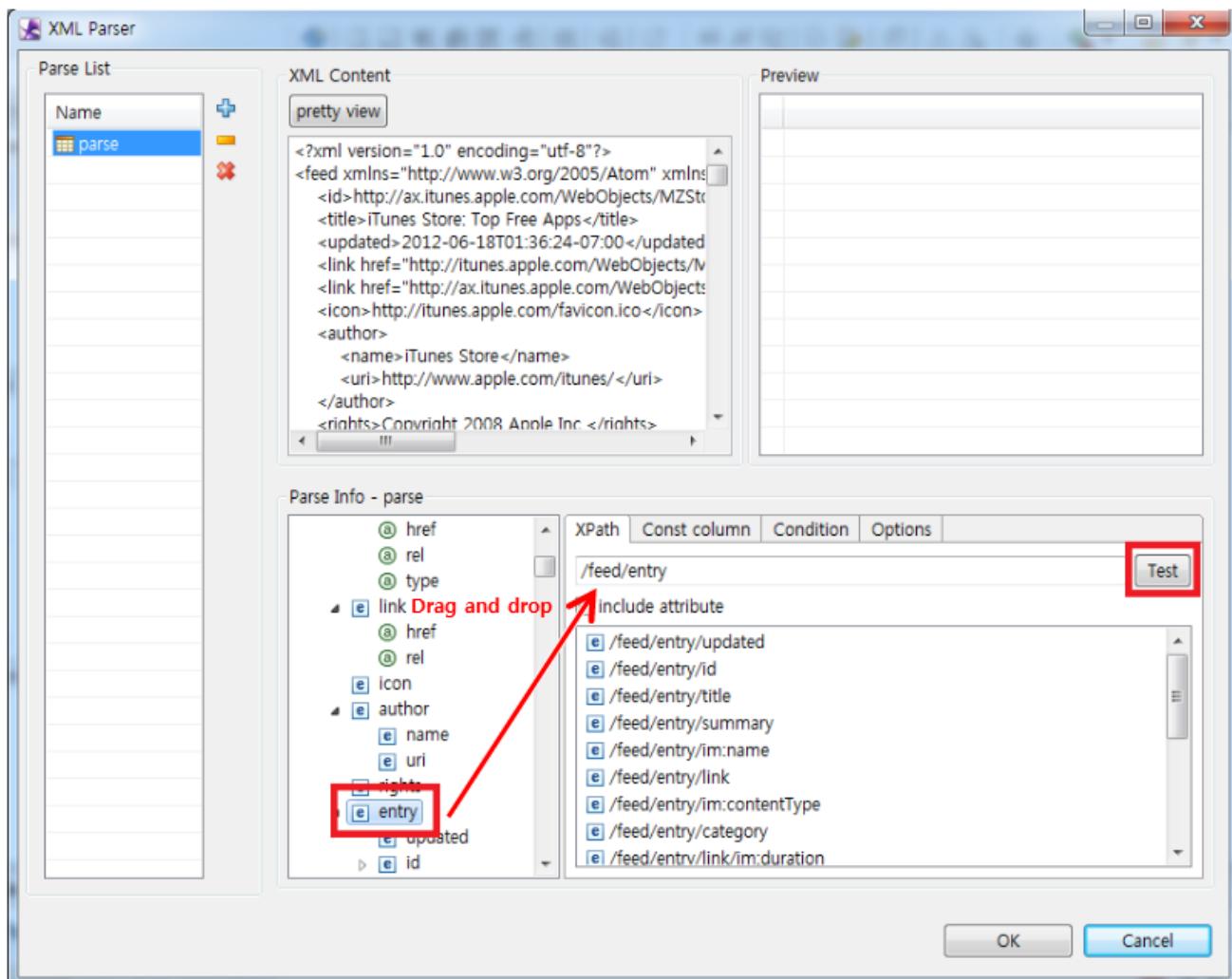
7. 8.5 OpenApi Invoke를 이용한 모델 개발하기를 테스트 하여 출력 파라메터 중 ‘BODY’ 파라메터의 값(XML 문자열)을 복사 합니다.
8. XML Parser 위자드의 ‘XML Content’ 영역에 XML 문자열 을 붙여넣기 합니다.
9. 다음 과 같이 XML Content에 xml 값이 보여지게 되고 pretty view를 클릭 하면 알아보기 쉽게 나타납니다.
10. 그리고 하단의 ‘Parser Info’ 부분에 XML이 Tree 형태로 보여지게 됩니다.



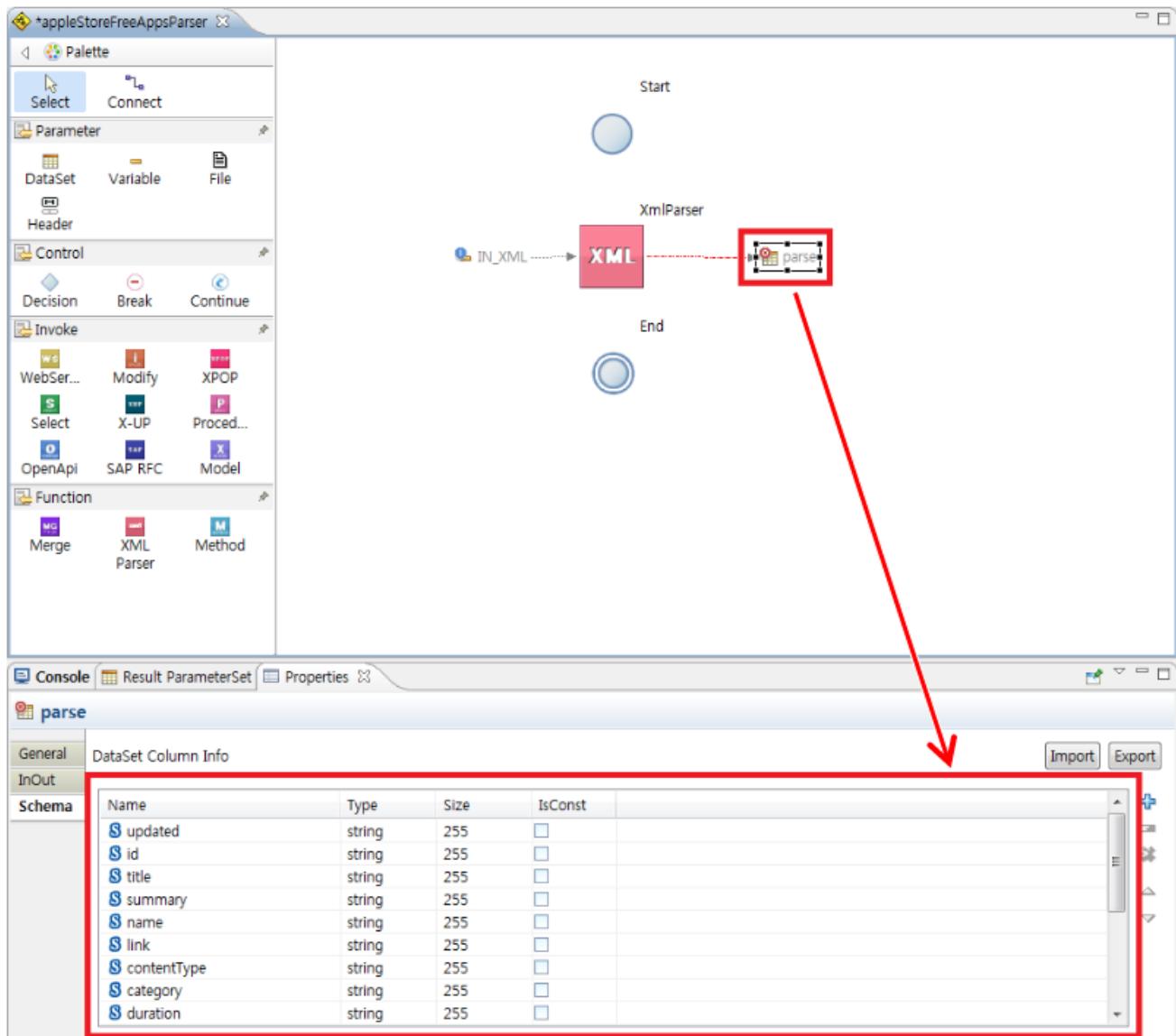
11. 좌측 상단의 ‘Parse List’의 Add 버튼[+]을 클릭하여 파싱을 할 구조를 설정 합니다.
12. Parse List 가 추가가 되면 ‘Parse Info’ 부분의 ‘XPath, Const column, Condition, Option’ 탭이 활성화 됩니다.



13. 'Parser Info'의 노드 중 반복되는 entry 노드를 **XPath**로 Drag and drop합니다.
14. 'XPath' 하단의 영역에 설정 된 **XPath** 하위의 자식 노드들의 정보가 생성 됩니다. 해당 정보가 데이터셋을 생성 할 경우 컬럼으로 생성이 됩니다.
15. 'Test' 버튼을 클릭하여 결과를 'Preview' 를 통해 확인 합니다.

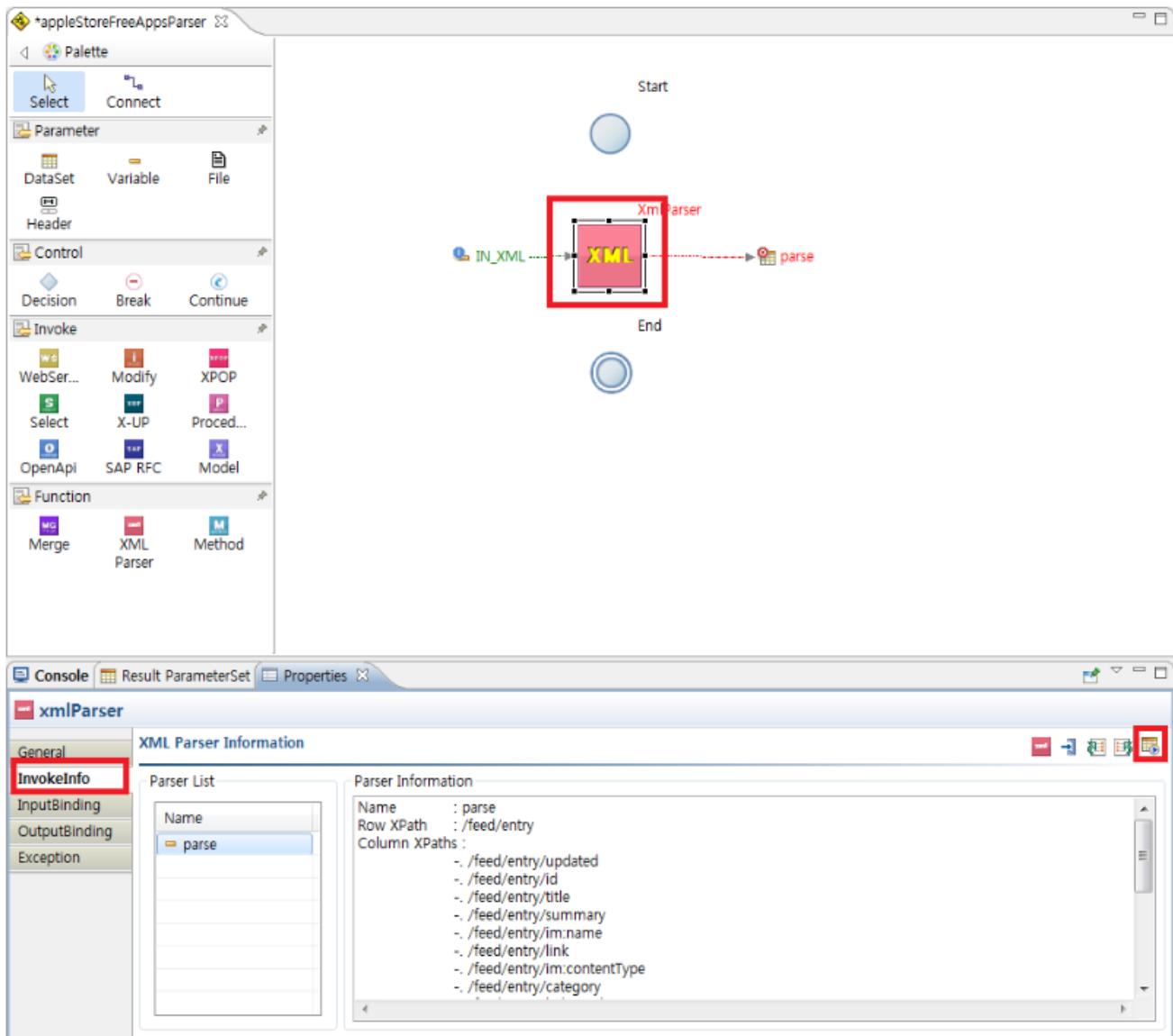


16. OK 버튼을 클릭하게 되면 출력 파라미터로 'parse'라는 이름의 데이터셋이 생성이 됩니다.
17. 데이터셋의 스키마는 XPath에 해당하는 하위 노드의 명칭으로 구성이 됩니다.

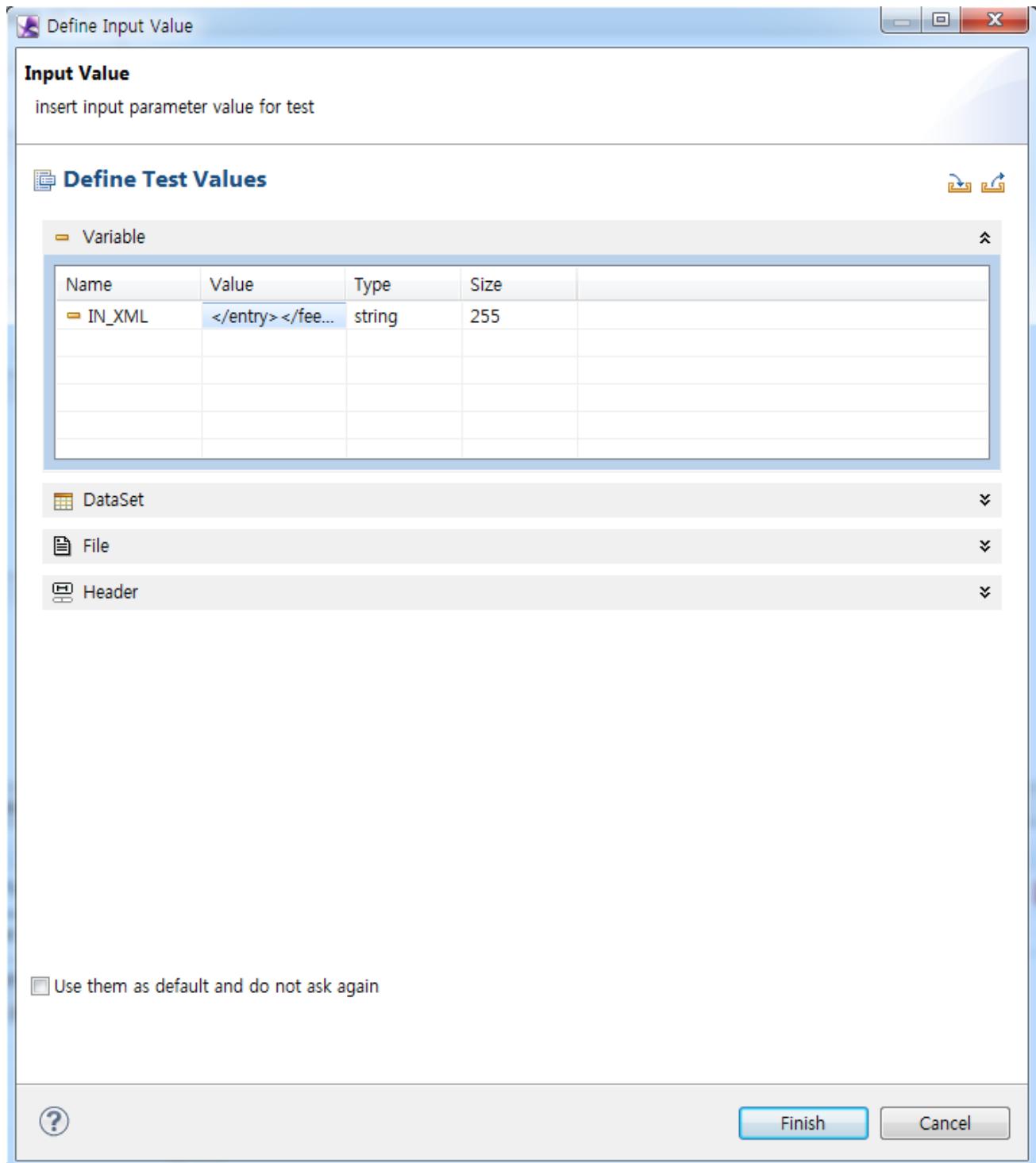


Function 테스트 하기

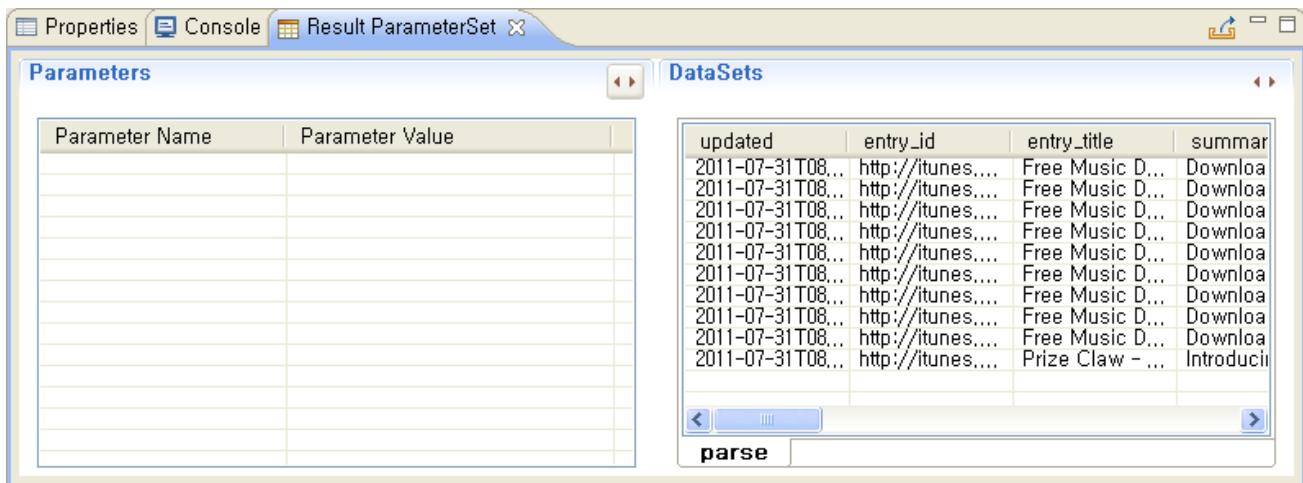
1. XML Parser의 Properties의 'ParserInfo'를 선택하고 [] 를 클릭하여 테스트를 진행합니다.



2. Define Input Value 위자드가 실행 되면 입력 파라메터인 XML을 입력 합니다.
3. 'Finish' 버튼을 클릭 합니다.



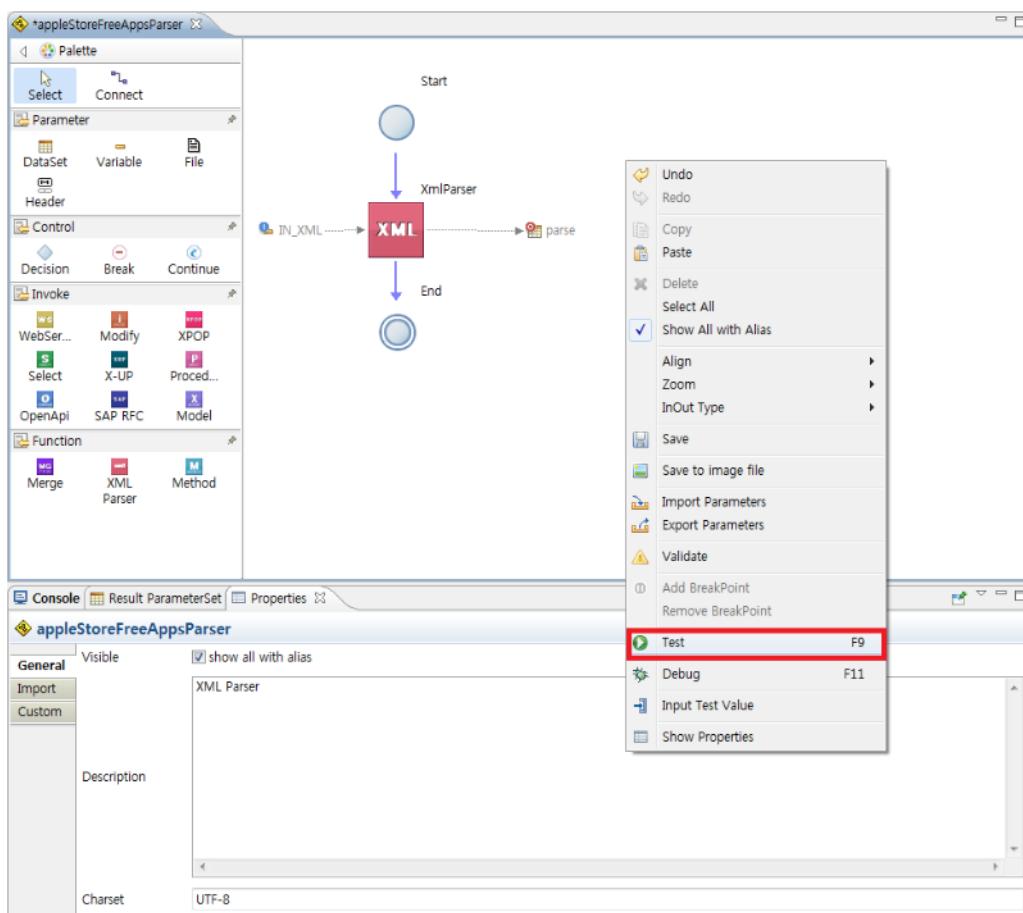
4. 테스트가 성공적으로 완료되면 Result ParameterSet 뷰에서 결과를 확인 할 수 있습니다.



모델 테스트하기

- 모델 에디터에서 Palette의 Connect를 선택하여 Start 노드와 XML Parser, End 노드를 연결 해 줍니다.
- 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.

테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.



8.11 UDDI를 이용하여 WebService Invoke 개발하기

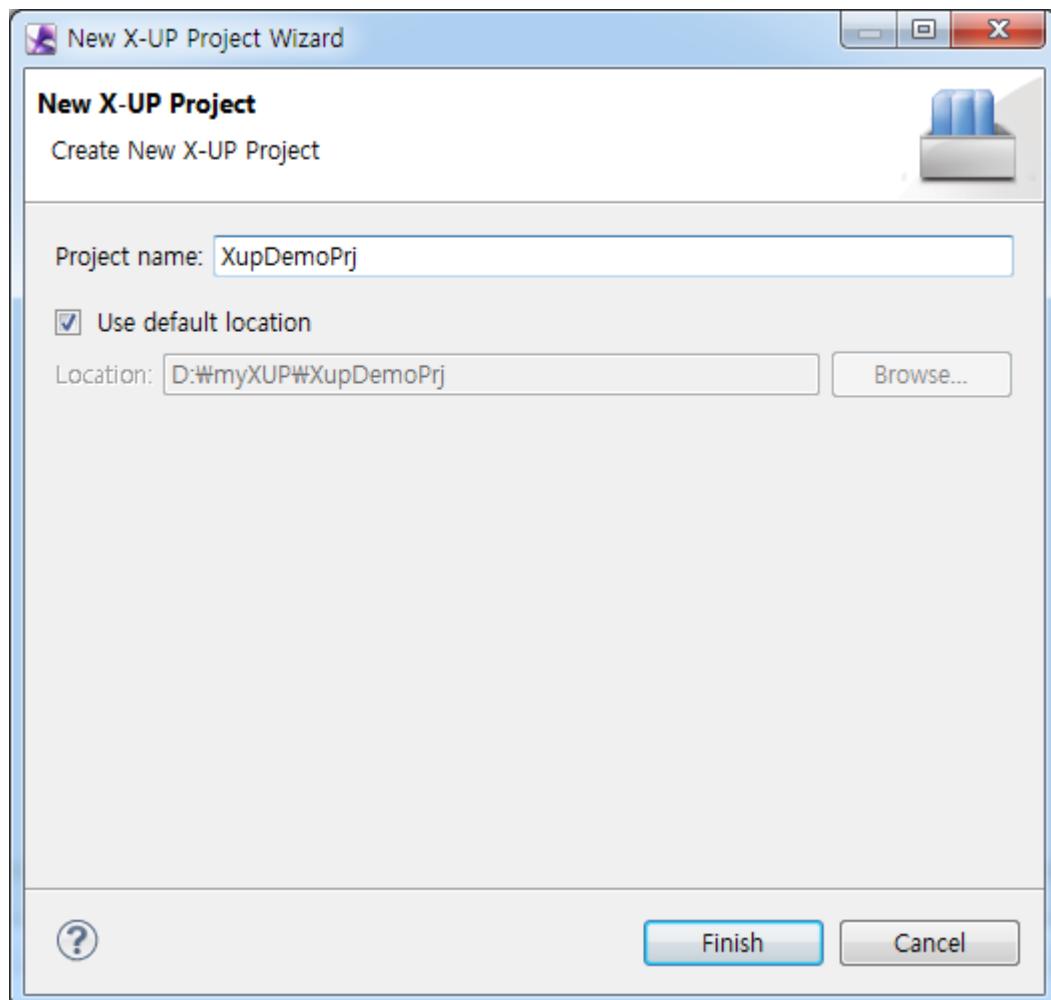
이 절에서는 UDDI에 등록된 서비스 정보를 획득하는 개발 방법을 설명합니다.

UDDI를 이용한 개발 단계는 다음과 같습니다.

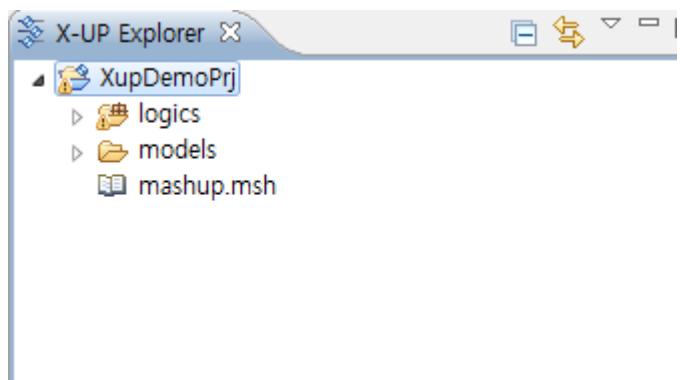
- X-UP 프로젝트 생성하기
- Automation 모델을 생성 및 UDDI 데이터 소스 생성하기
- UDDI를 이용하여 WebService 데이터 소스 생성하기.
- WebService Invoke 개발하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭합니다.

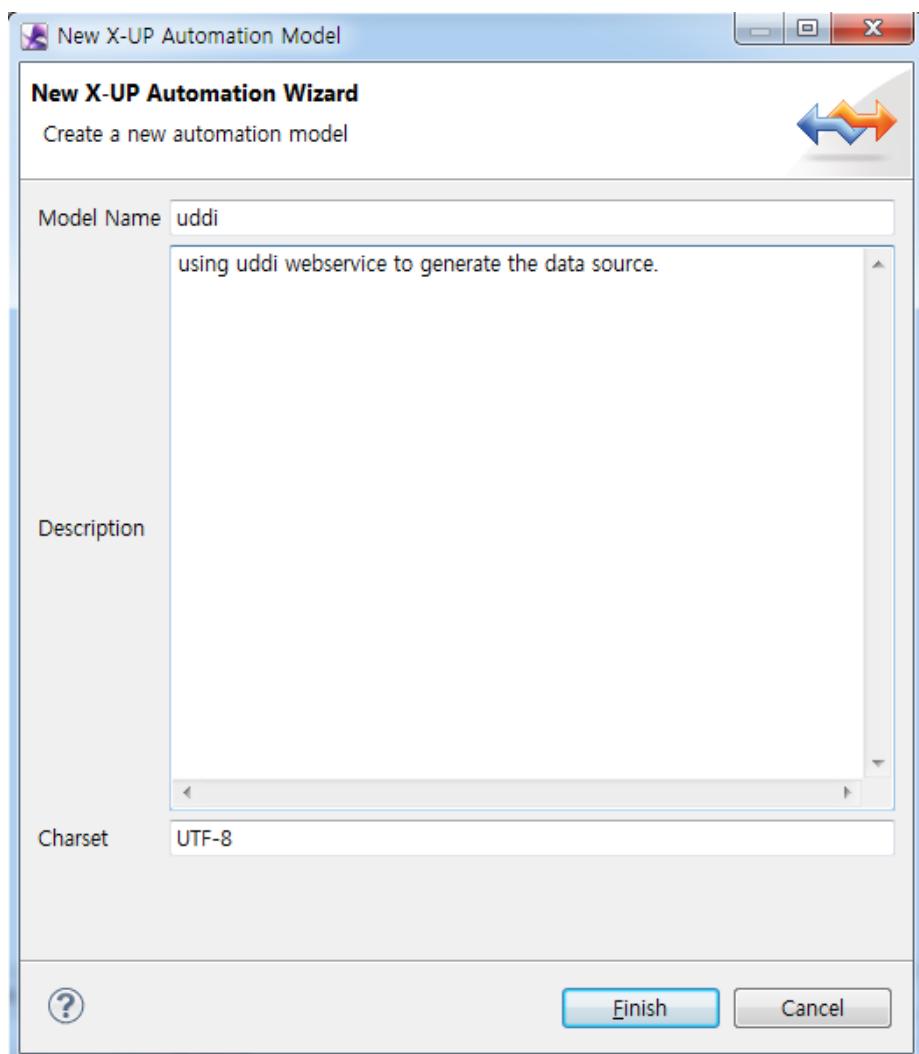


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

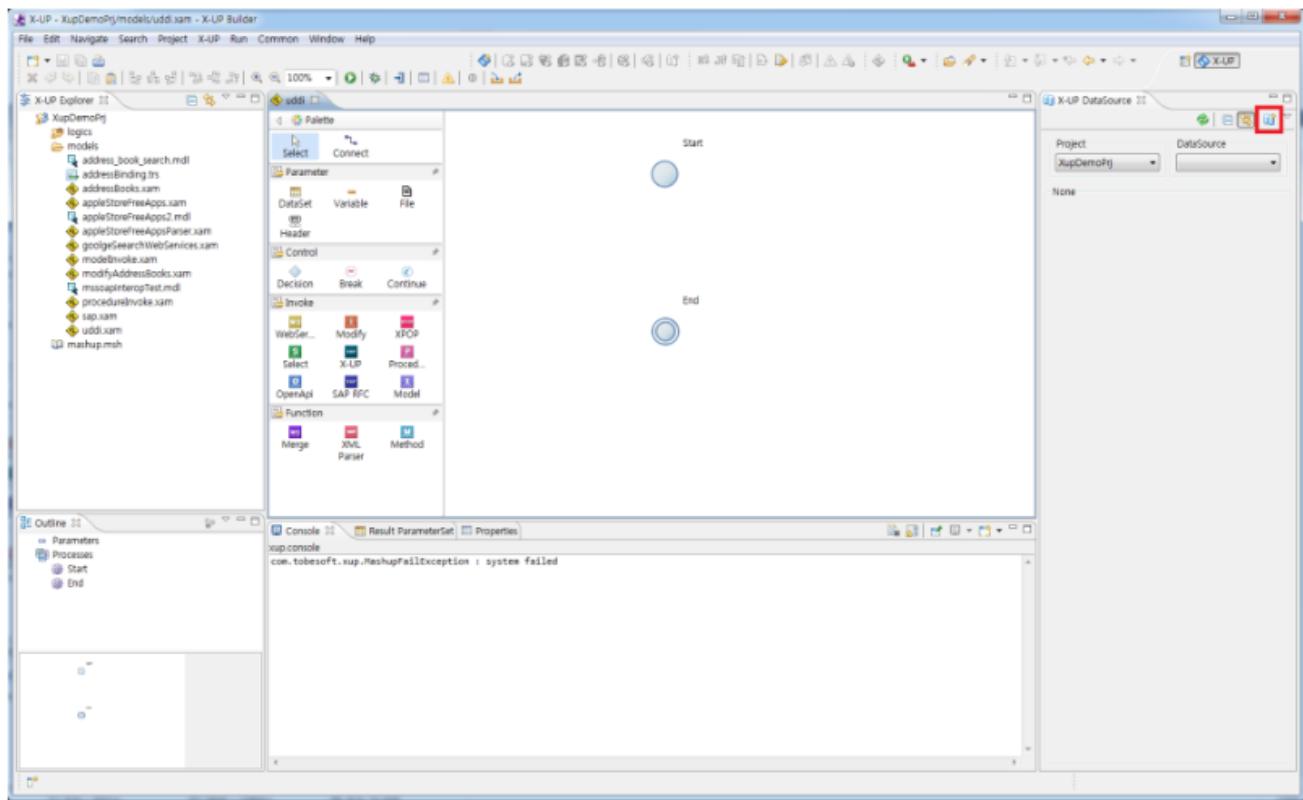


Automation 모델 생성 및 UDDI 데이터소스 생성하기

- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 원하는 모델 이름을 입력하고 OK 버튼을 클릭합니다.

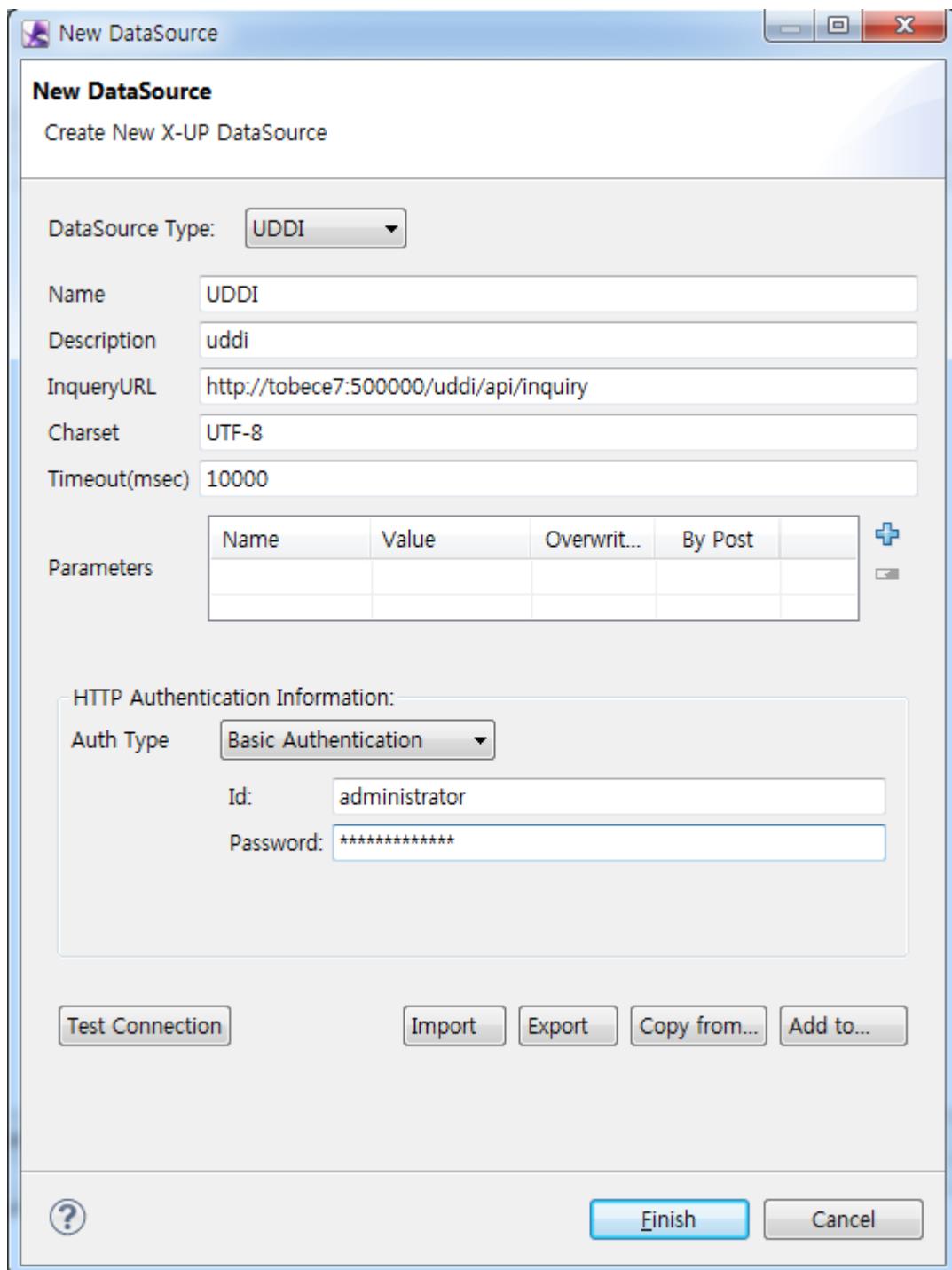


3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



4. 화면 우측의 X-UP DataSource 뷰에서 new DataSource를 선택해 Wizard를 실행합니다.
5. New DataSource 페이지에서 새로운 데이터소스를 정의합니다. 각 필드 값을 입력한 후 'Test Connection'버튼을 선택하여 DataSource와 연결이 정상적으로 맺어지는지 확인합니다.
6. Connection 확인 후에 'Finish' 버튼을 클릭합니다.

Field Name	Field Value
Name	데이터소스 이름으로 다른 데이터소스 이름과 중복되지 않는 값을 입력합니다.
Inquiry URL	UDDI에 접속 할 수 있는 URL를 입력합니다.



UDDI 를 제공하는 호스트가 HTTP Basic인증을 요구하는 경우에는 HTTP auth Info의 ‘Id’와 ‘Password’에 유효한 값을 입력합니다.

HTTP Authentication Information:

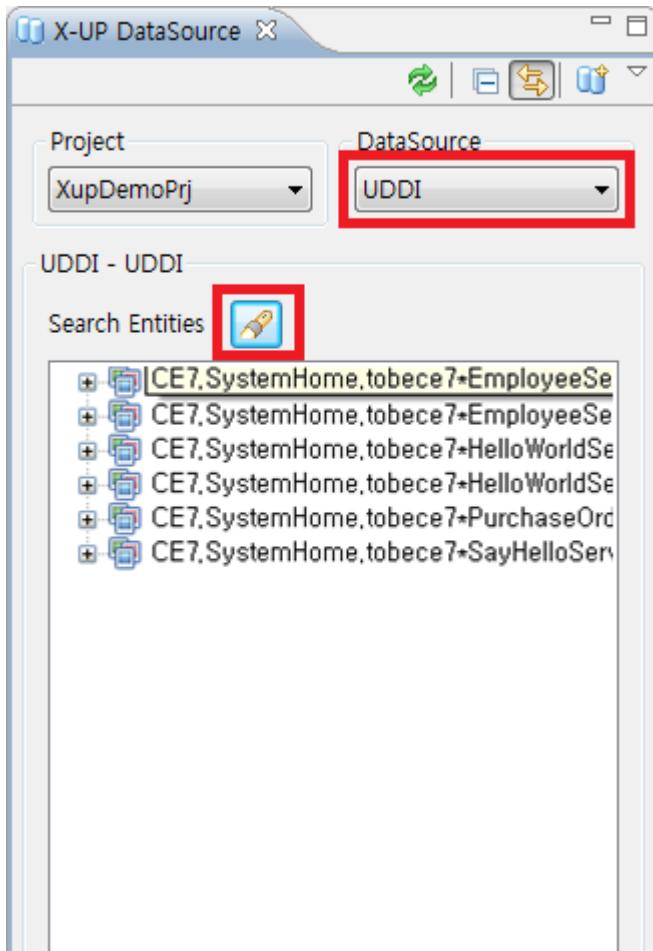
Auth Type: **Basic Authentication**

Id:

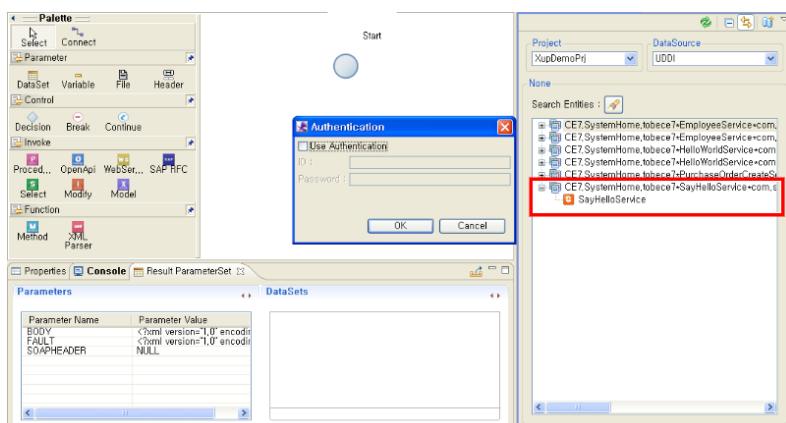
Password:

UDDI를 이용하여 WebService 데이터소스 생성하기

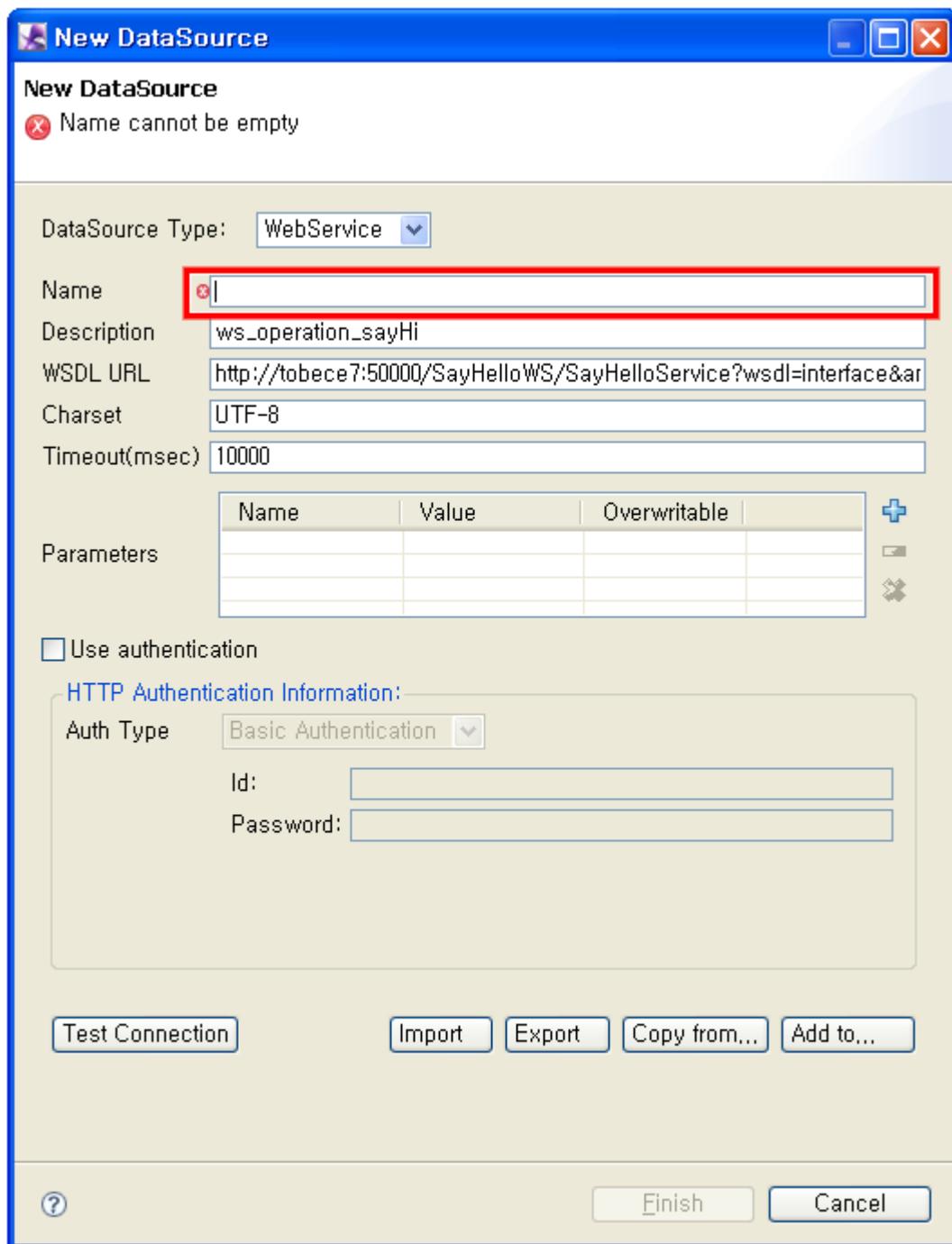
1. 화면 우측의 X-UP DataSource 뷰에서 생성 된 UDDI 데이터 소스를 선택 합니다..
2. ‘Search Entities’ 버튼 [] 을 클릭하여 ‘Business Entity’ 명칭을 입력하여 데이터 엘리먼트를 검색 합니다.



3. ‘Entity’ 중 하나의 ‘Entity’를 선택 하면 하위의 ‘Business Service’를 검색 합니다.
4. 검색 된 ‘Business Service’ 중 하나의 Service를 선택 하였을 때 인증이 필요 한 경우 해당 정보를 입력 하고 OK 버튼을 클릭 합니다.



5. 최종 WSDL 이 검색 되고 WSDL 서비스의 인터페이스를 확인 할 수 있습니다.
6. WebService 로 개발 할 WSDL을 에디터로 드래그 앤 드롭 하면 New DataSource 위자드가 실행이 되며 다른 데이터 소스와 중복 되지 않는 이름을 사용자가 지정해야 합니다.



7. 중복 되지 않는 이름을 입력 한 후 'Finish' 버튼을 클릭 할 경우 WebService의 Invoke WebService 위자드가 실행 되며 Invoke를 생성 하게 됩니다.

WebService Invoke 개발하기

8.6 WebService Invoke를 이용한 모델 개발하기를 참조 합니다.

9.

X-UP Transaction 모델 개발하기

이 장에서는 DBMS와의 연동 및 DB와 관계없는 사용자 임의의 로직을 구현하기 위한 트랜잭션 모델(Transaction Model)을 이용하여 예제를 통해 직접 개발하는 방법에 대하여 설명합니다.

X-UP 모델 개발단계는 다음과 같습니다.

1. X-UP Builder를 이용하여 모델을 개발한다.
2. X-UP Builder에서 개발된 모델을 테스트한다.
3. 개발된 모델을 X-UP Builder의 deploy 기능을 이용하여 X-UP 서버에 배치시킨다.
4. X-UP 테스트 웹 페이지에서 배치된 모델을 테스트한다.

X-UP Builder는 개발자가 쉽게 코드를 생성 할 수 있도록 아래 다섯가지 종류의 마법사(Wizard)를 제공합니다.

- DB Binding Code 마법사
- Transaction JDBC Code 마법사
- DB Procedre Calling Code 마법사
- X-UP Model Invoking 마법사
- DataSet Creating 마법사

이 장에서는 위 두가지 종류의 마법사를 사용하여 주소록(ADDRESS_BOOK) 테이블을 변경하는 방법을 설명합니다.

이 장의 구성은 다음과 같습니다.

- DB Binding Code 마법사를 이용한 모델 개발하기
- Transaction JDBC Code 마법사를 이용한 모델 개발하기

9.1 DB Binding Code 마법사를 이용한 모델 개발하기

DB Binding은 데이터셋과 데이터베이스 테이블의 데이터를 동기화하는 기능입니다. 데이터셋은 insert, update, delete된 트랜잭션 정보를 담고 있습니다. DB Binding은 데이터셋의 트랜잭션 정보를 데이터베이스 테이블에 반영합니다.



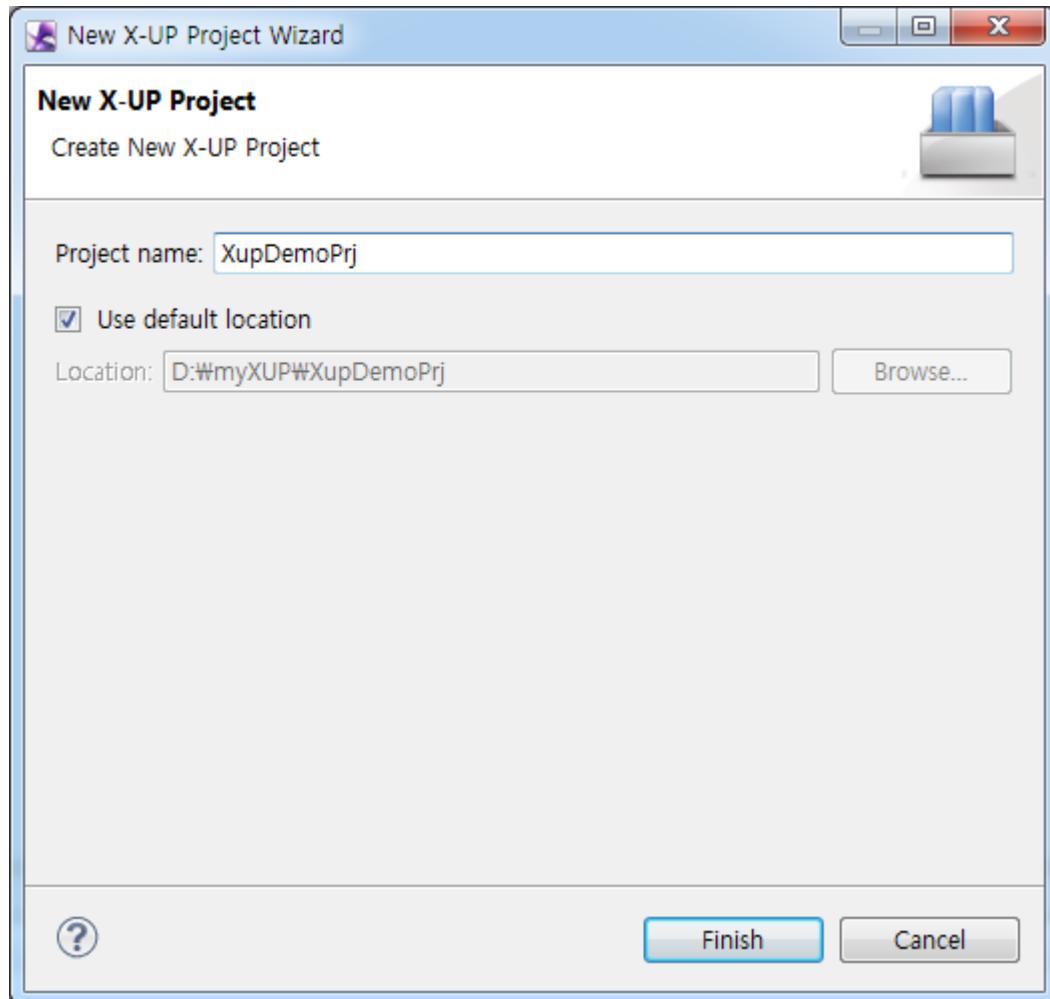
이 절에서 사용한 데이터베이스 테이블 정보는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 참조하십시오.

이 절에서 설명하는 Transaction 모델의 DB Binding Code 마법사를 이용한 개발단계는 다음과 같습니다.

- X-UP 프로젝트 생성하기
- 트랜잭션 모델 생성하기
- 입출력 파라메터 및 데이터셋 설정
- 모델 테스트

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 New X-UP Project Wizard를 실행합니다.
2. New X-UP Project Wizard에서 Project name 필드에 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭합니다.

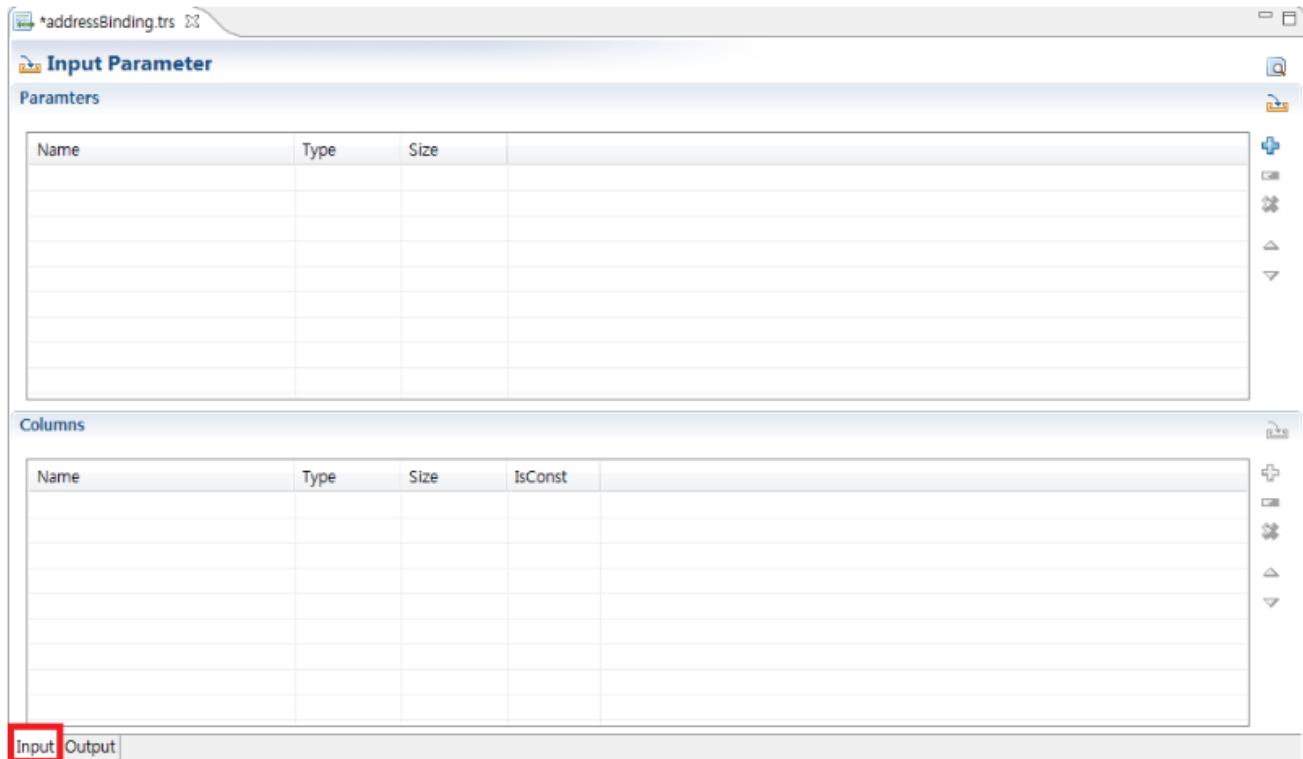


트랜잭션 모델 생성하기

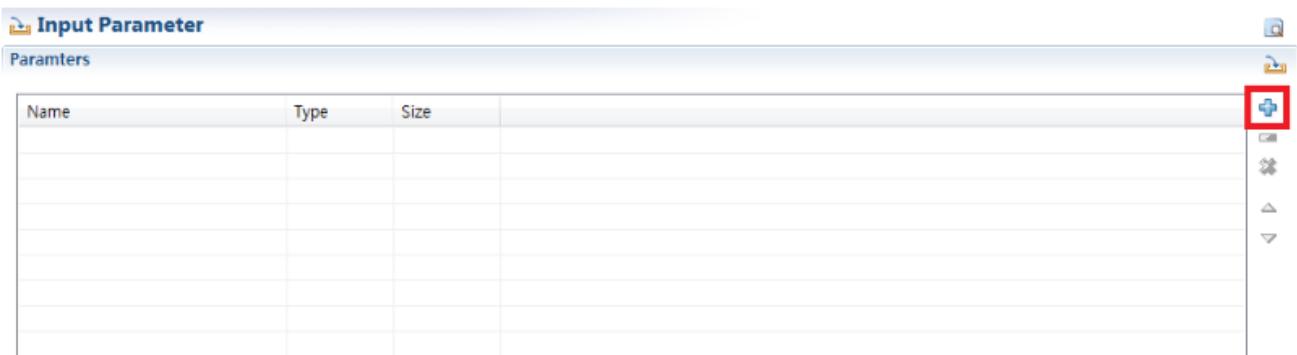
1. X-UP Explorer 뷰에서 생성한 프로젝트를 선택합니다.
2. [File > New > X-UP Transaction Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
3. New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 'Finish' 버튼을 클릭합니다.

입출력 파라미터 및 데이터셋 설정

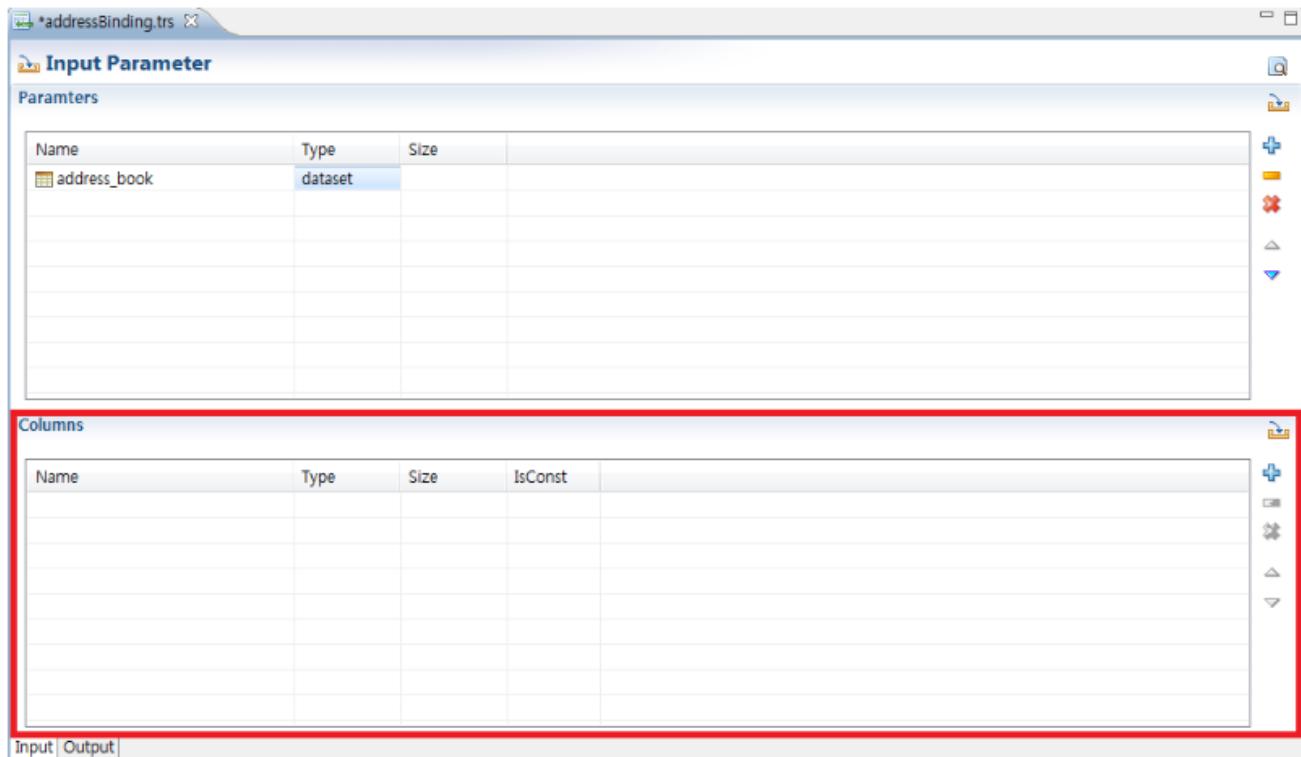
1. 트랜잭션 모델 설정 에디터에서 Input탭을 선택합니다.



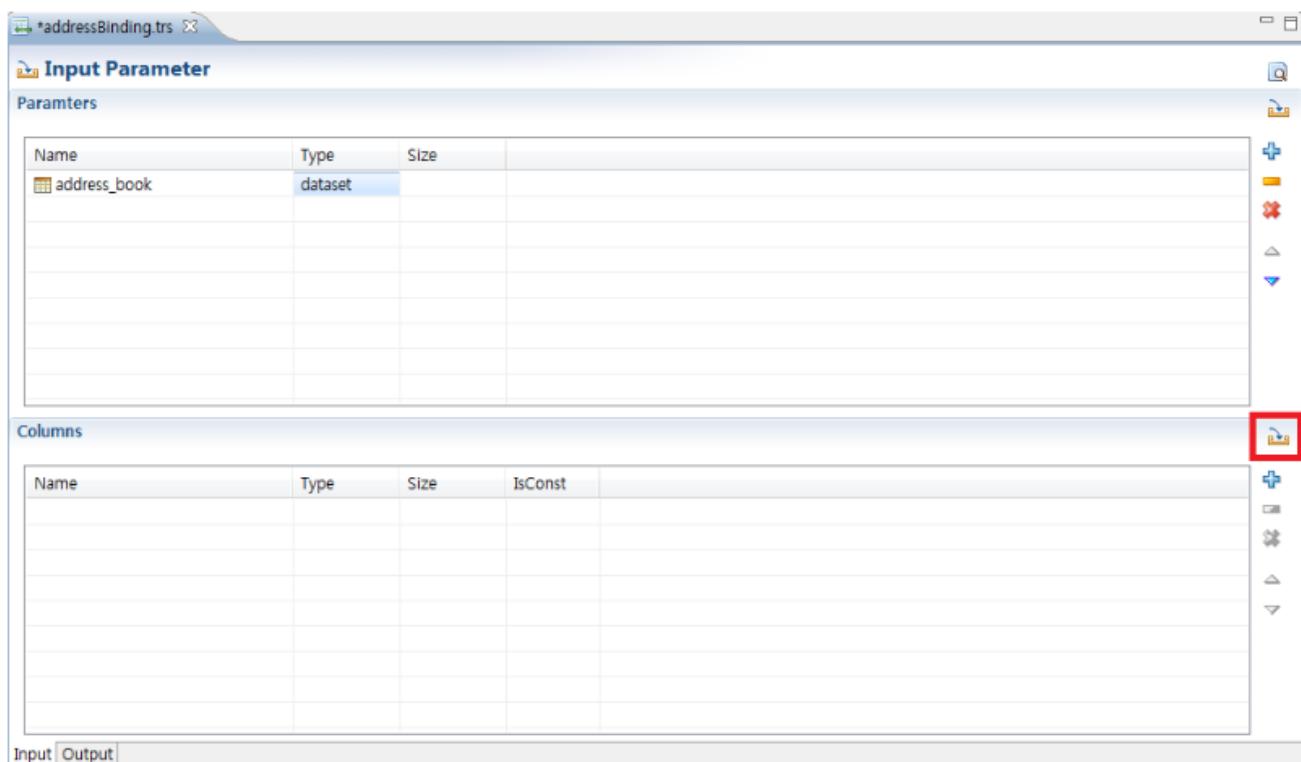
2. Parameters 설정 원도우에서 Add 버튼[+]을 클릭합니다.



3. 파라미터의 Name은 'address_book'으로 Type은 'dataset'으로 설정합니다. 파라미터의 Type이 데이터셋인 경우에는 Columns뷰가 활성화됩니다.

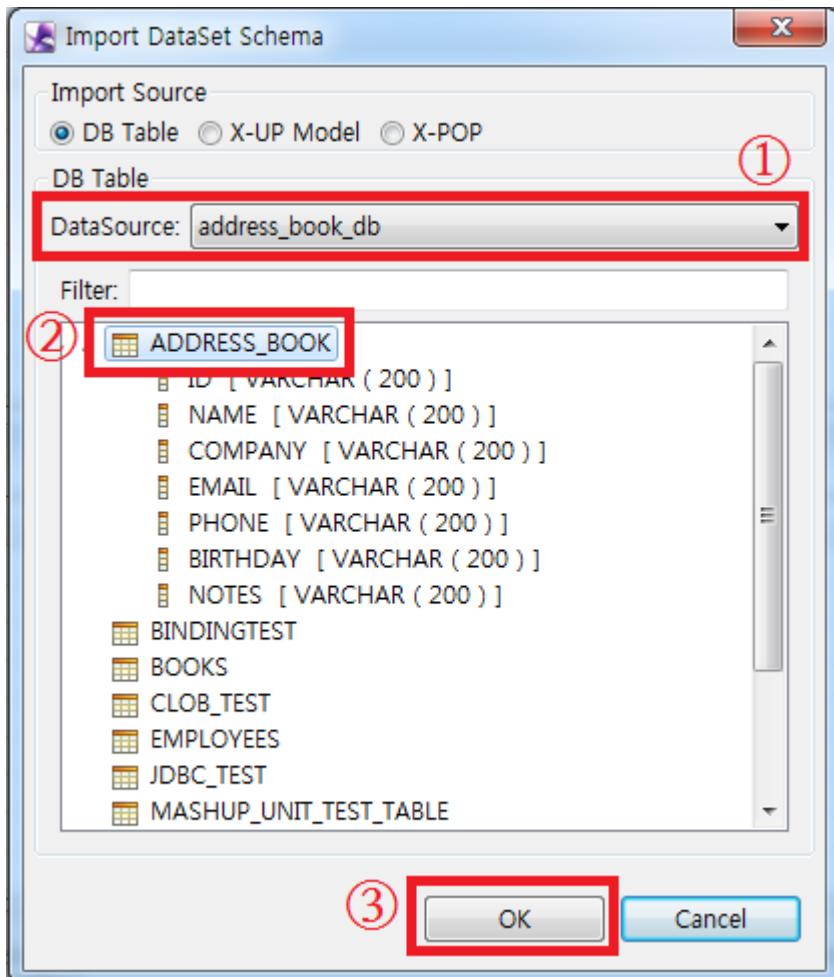


- 추가한 'address_book' 데이터셋의 컬럼을 설정합니다. Column 위치드의 Import Schema 버튼[]을 클릭하여 Import DataSet Schema 위치드를 실행합니다.



- Import 데이터셋 Schema 위치드에서 데이터소스를 선택합니다(1번). 주소록 테이블(ADDRESS_BOOK)을 선택하고(2번) OK 버튼을 클릭(3번)합니다.

 데이터셋 컬럼 설정은 Columns 원도우에서 'Add' 버튼[+]을 클릭하여 사용자가 직접 입력하거나 'Import Schema' 버튼[+]을 클릭하여 데이터베이스 테이블 또는 다른 X-UP 모델로부터 컬럼 정보를 가져올 수 있습니다.



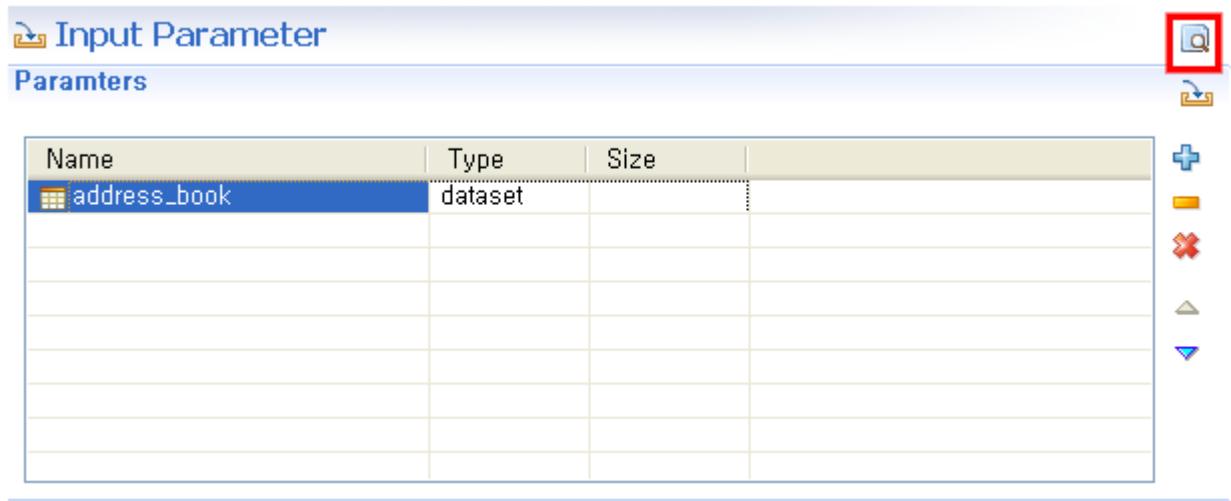
Import Source	설명
DB Table	데이터베이스의 테이블로부터 컬럼 정보를 가져옵니다. 현재 X-UP 프로젝트에 하나 이상의 DB 데이터소스가 설정되어 있어야 사용 가능합니다.
X-UP Model	현재 X-UP 프로젝트에 존재하는 다른 모델의 출력 데이터셋을 이용하여 컬럼 정보를 가져옵니다.

6. Import DataSet Schema 위자드에서 설정한 컬럼 정보가 Columns 뷰에 아래와 같이 표시됩니다.

Columns			
Name	Type	Size	IsConst
ID	string	200	<input type="checkbox"/>
NAME	string	200	<input type="checkbox"/>
COMPANY	string	200	<input type="checkbox"/>
EMAIL	string	200	<input type="checkbox"/>
PHONE	string	200	<input type="checkbox"/>
BIRTHDAY	string	200	<input type="checkbox"/>
NOTES	string	200	<input type="checkbox"/>

위자드를 이용하여 모델 로직 클래스 수정하기

- 트랜잭션 모델 에디터 상단에 위치한 **Open logic class** 버튼 []을 클릭하여 로직 클래스 에디터를 실행합니다.



```

addressBinding.trs addressBindingTransactionLogic.java
XupDemoPrj > logics > (default package) > addressBindingTransactionLogic >
import java.sql.Connection;

public class addressBindingTransactionLogic extends addressBindingBaseTransactionLogic {

    public DataSet[] execute(ParameterSet parameterSet) throws TransactionFailException{

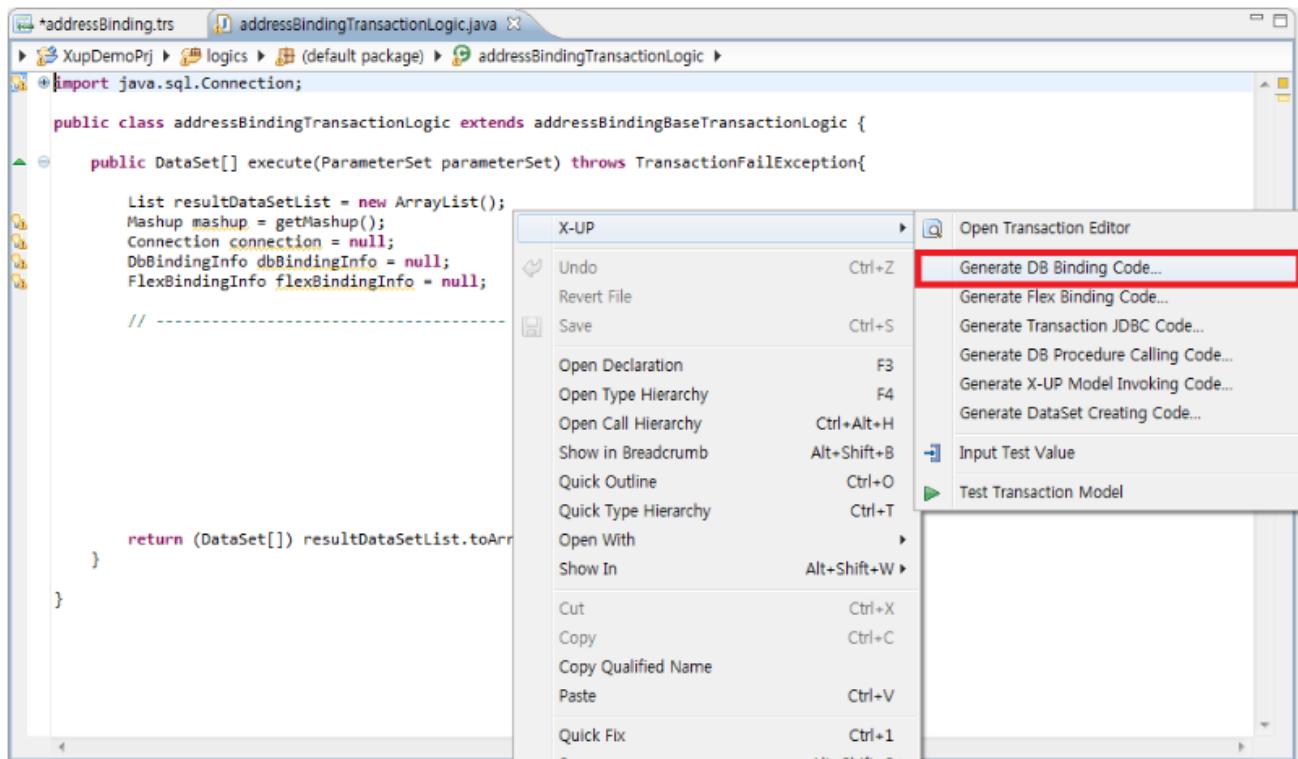
        List resultDataSetList = new ArrayList();
        Mashup mashup = getMashup();
        Connection connection = null;
        DbBindingInfo dbBindingInfo = null;
        FlexBindingInfo flexBindingInfo = null;

        // ----

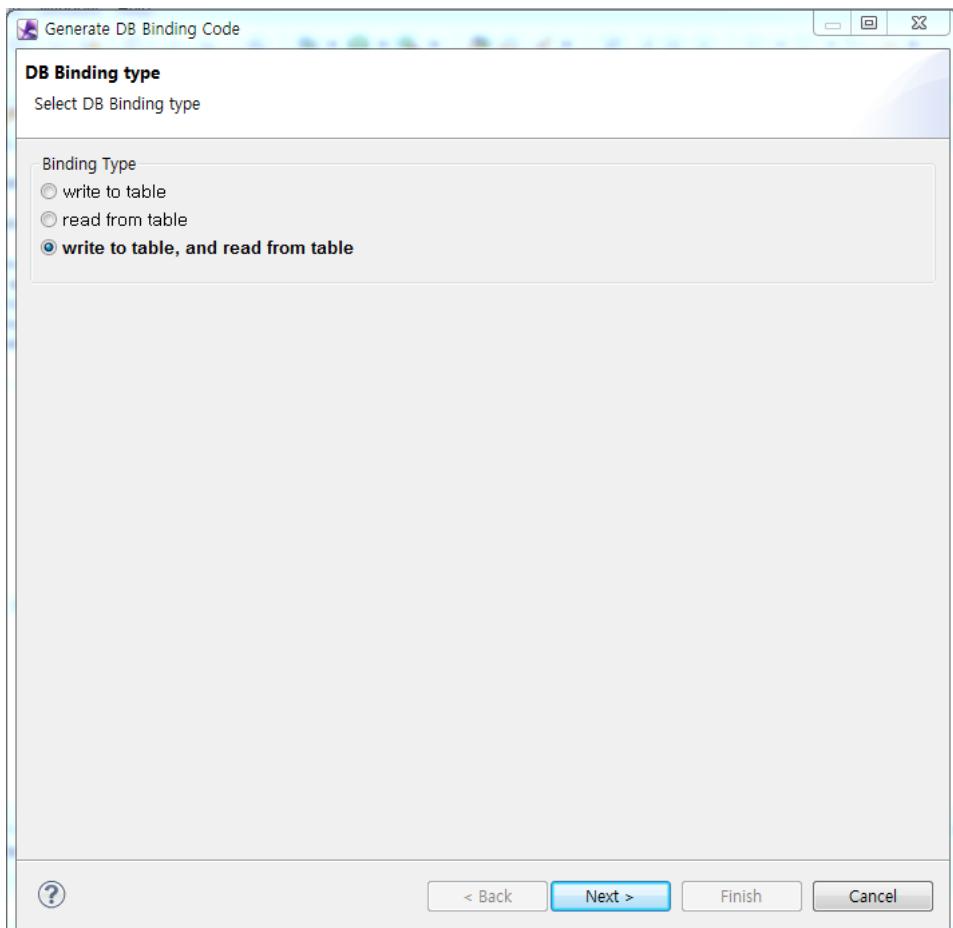
        return (DataSet[]) resultDataSetList.toArray(new DataSet[resultDataSetList.size()]);
    }
}

```

- 로직 클래스 에디터에서 주석문 '// ----- ...' 다음 라인으로 커서를 이동합니다. 마우스 오른쪽 버튼을 클릭하고 **X-UP Generate DB Binding Code**를 선택합니다.



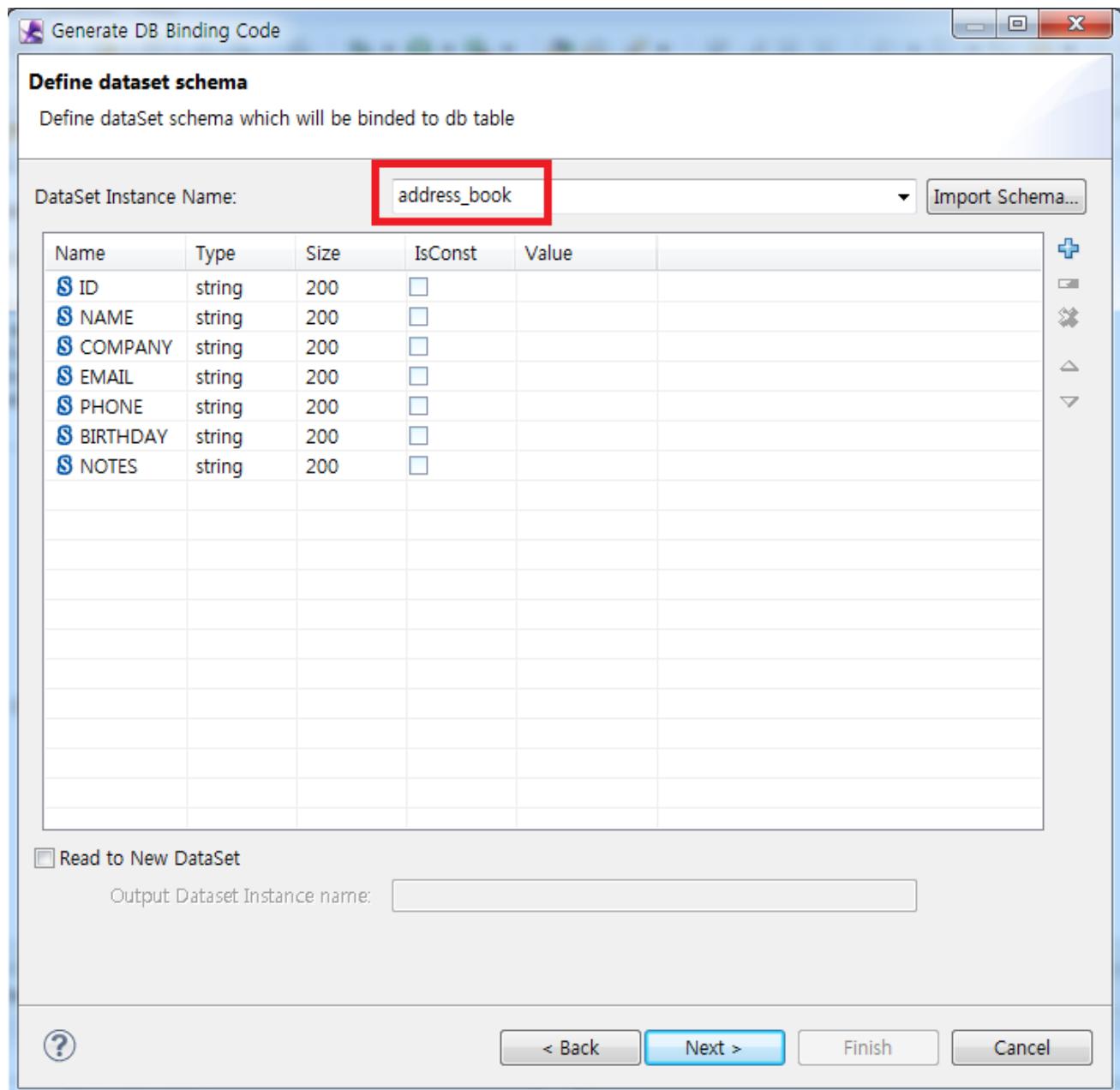
3. Generate DB Binding Code 위자드의 첫번째 단계는 DB Binding type 설정입니다. write to table, and read from table을 선택하고 Next 버튼을 클릭합니다.



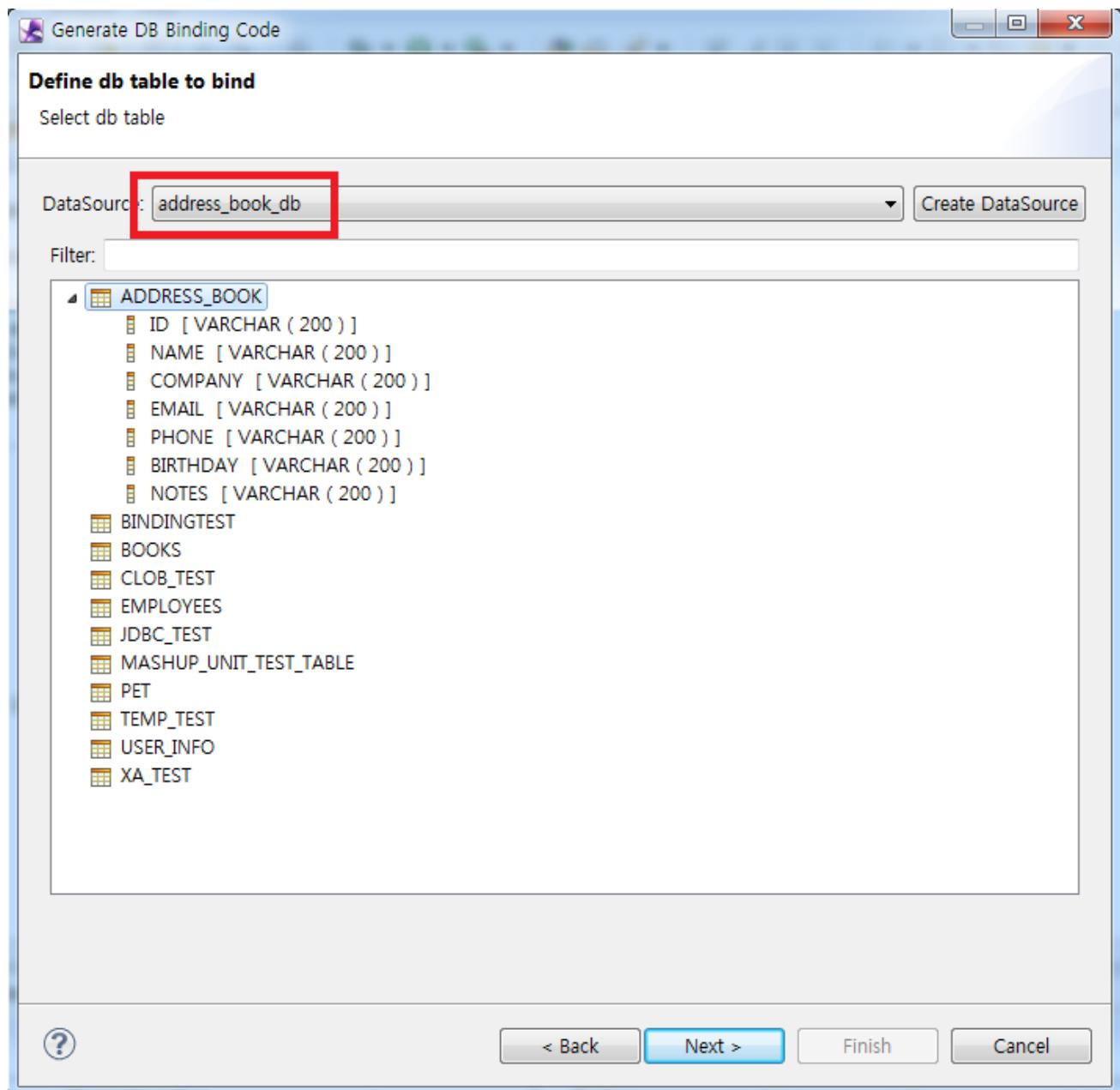


'write to table, and read from table' 옵션은 입력 데이터셋의 내용을 DB table에 적용한 후 그 결과를 다시 조회하여 출력 데이터셋으로 보내고자 할 경우 사용합니다.

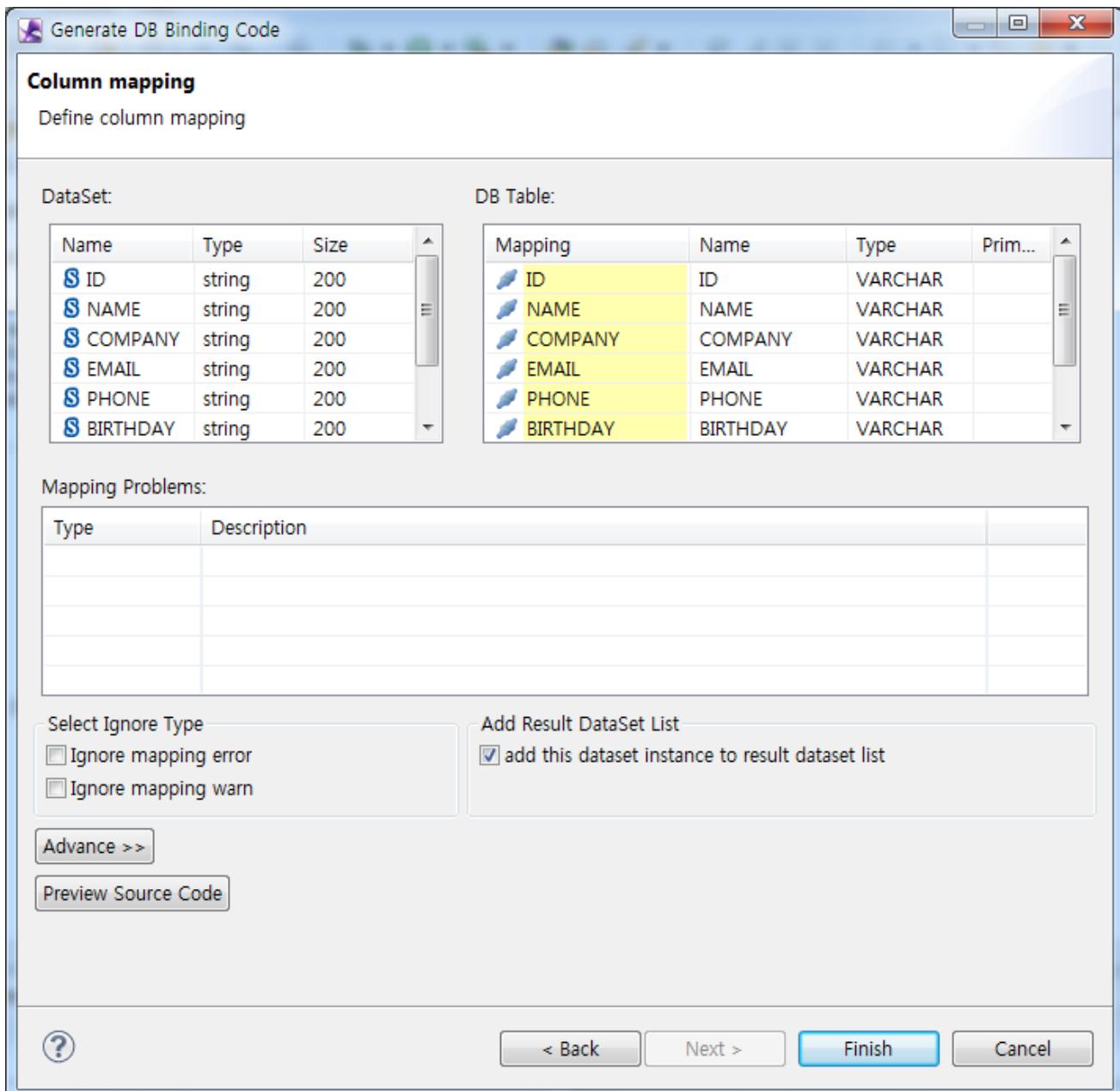
- 데이터셋 스키마를 설정합니다. 트랜잭션 모델 에디터에서 설정한 입력 데이터셋(address_book)을 선택하고 **Next** 버튼을 클릭합니다.



- DataSource 콤보박스에서 address_book_db 데이터소스를 선택하면 테이블 리스트가 조회됩니다. 예제에서 사용할 주소록 테이블(ADDRESS_BOOK)을 더블클릭하고 **Next** 버튼을 클릭합니다.



6. 데이터셋과 테이블을 맵핑합니다. X-UP Builder는 데이터셋과 테이블의 컬럼명이 동일한 경우에는 자동으로 맵핑합니다. 예제는 동일한 스키마의 데이터셋과 DB 테이블을 사용하므로 직접 맵핑 할 필요가 없습니다. 'Finish' 버튼을 클릭합니다.



7. 'Generate DB Binding Code' 위자드를 정상적으로 완료하면 로직 클래스 에디터에 다음과 같은 소스가 추가 됩니다.

```
public DataSet[] execute(ParameterSet parameterSet) throws TransactionFailException {
    List resultDataSetList = new ArrayList();
    Mashup mashup = getMashup();
    Connection connection = null;
    DbBindingInfo dbBindingInfo = null;
    FlexBindingInfo flexBindingInfo = null;

    // -----
    // Generate DB Binding Code
}
```

```

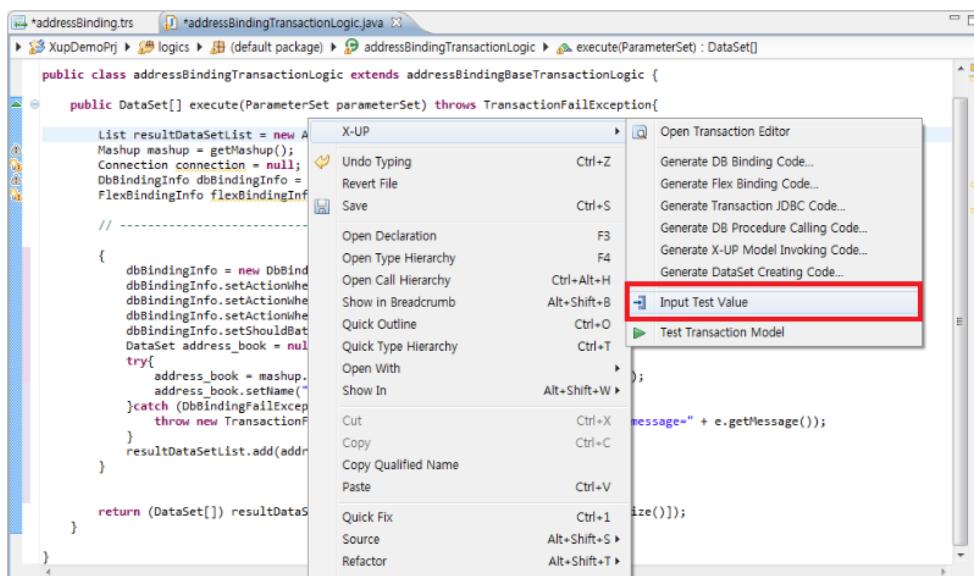
{
    dbBindingInfo = new DbBindingInfo("address_book_db", "ADDRESS_BOOK");
    dbBindingInfo.setActionWhenUpdateFailed(DbBindingInfo.Action.INSERT);
    dbBindingInfo.setActionWhenInsertFailed(DbBindingInfo.Action.UPDATE);
    dbBindingInfo.setActionWhenDeleteFailed(DbBindingInfo.Action.IGNORE);
    dbBindingInfo.setShouldBatchCheckForRecordExist(false);
    DataSet address_book = null;
    try {
        address_book =
            mashup.writeAndReadWithDb(parameterSet.getDataSet("address_book"),
            dbBindingInfo);
    } catch (DbBindingFailException e) {
        throw new TransactionFailException("transaction failed. e=" + e +
            message=" + e.getMessage());
    }
    resultDataSetList.add(address_book);
}
return (DataSet[])resultDataSetList.toArray(new DataSet [resultDataSetList.size()]);
}

```

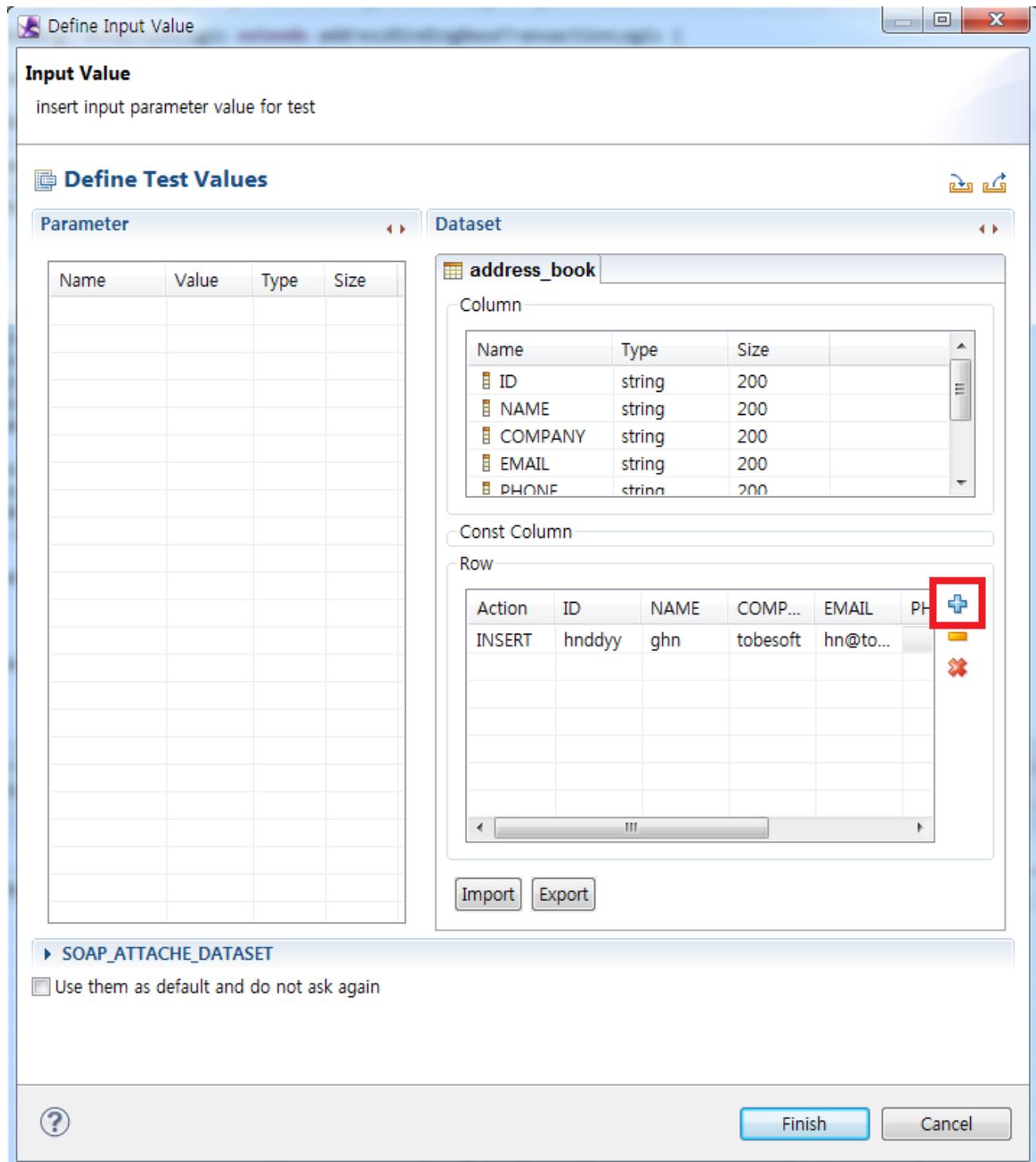
8. File Save 메뉴를 클릭하여 개발한 모델을 저장합니다.

모델 테스트하기

1. 트랜잭션 모델을 테스트합니다. 테스트에 앞서 테스트 데이터를 입력 합니다. 로직 클래스 에디터에서 마우스 오른쪽 버튼을 클릭하여 X-UP Input Test Value를 선택 합니다.



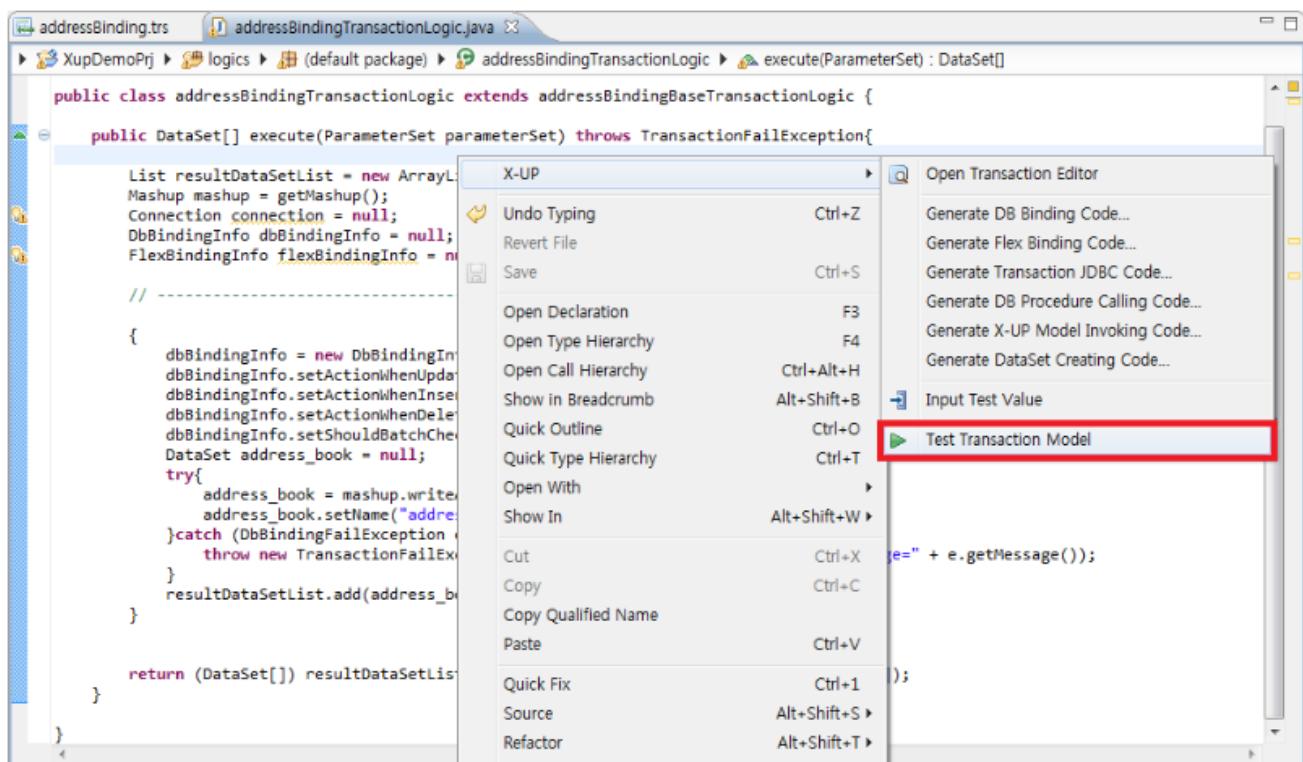
2. 트랜잭션 모델 테스트에 사용할 데이터를 입력합니다. **Input Value** 원도우에서 **Add** 버튼[+]을 클릭하여 데이터를 입력합니다.



Action	설명
INSERT	'INSERT'가 설정된 Row는 DB 테이블에 추가됩니다.
UPDATE	'UPDATE'가 설정된 Row는 테이블에서 동일한 레코드를 찾아 업데이트를 수행합니다. 동일한 레코드는 기본 키(Primary Key) 컬럼의 값이 입력 데이터셋의 값과 동일한 레코

Action	설명
	드를 의미합니다. 그러므로 하나 이상의 기본 키 컬럼이 설정된 테이블에서만 ‘UPDATE’를 수행 할 수 있습니다.
DELETE	‘DELETE’가 설정된 Row는 테이블에서 동일한 레코드를 찾아 삭제합니다. 동일한 레코드는 기본 키(Primary Key)컬럼의 값이 입력 데이터셋의 값과 동일한 레코드를 의미합니다. 그러므로 하나 이상의 기본 키 컬럼이 설정된 테이블에서만 ‘DELETE’를 수행할 수 있습니다.
NORMAL	입력 데이터로만 사용되며 DB 트랜잭션에는 관여하지 않습니다.

3. 로직 클래스 에디터에서 마우스 오른쪽 버튼을 클릭하여 X-UP Test Transaction Model을 선택합니다.



테스트가 완료되면 Result ParameterSet부에서 결과를 확인 할 수 있습니다.

Result DataSet							
ID	NAME	COMPANY	EMAIL	PHONE	BIRTHDAY	NOTES	
11	yang	tobesoft	ysh@tobesof...	NULL	NULL	NULL	
0	Sheriff Woody	NULL	woody@exa...	NULL	19951122	NULL	
1	Buzz Lightyear	NULL	buzz@exam...	NULL	19951122	NULL	
2	Mr. Potato H...	NULL	potato@exa...	NULL	19951122	NULL	
3	Slinky Dog	NULL	slinky@exa...	NULL	19951122	NULL	
4	Flex	NULL	rex@example...	NULL	19951122	NULL	
5	Hamm	NULL	hamm@exa...	NULL	19951122	NULL	
6	Bo Peep	NULL	peep@example...	NULL	19951122	NULL	
7	Sarge	NULL	sarge@example...	NULL	19951122	NULL	
8	Lightning Mc...	NULL	lightning@ex...	NULL	20060609	NULL	
9	Mater	NULL	mater@example...	NULL	20060609	NULL	
10	Sally Carrera	NULL	sally@example...	NULL	20060609	NULL	

9.2 JDBC Transaction Code 마법사를 이용한 모델 개발하기

앞에서 설명한 **Generate DB Binding Code**는 매우 간단하게 DB 트랜잭션을 처리하는 코드를 생성하지만 테이블에 기본 키(Primary Key)가 존재해야 하고 하나의 테이블에서만 트랜잭션을 처리할 수 있는 등, 제한된 환경에서만 사용 할 수 있습니다. 이번에 설명하는 **Generate Transaction JDBC Code** 위자드는 개발자가 직접 SQL문을 설정 할 수 있으므로 **Generate DB Binding** 보다는 좀 더 세밀한 제어가 가능합니다. 예제는 앞에서 설명한 **DB Binding Code** 마법사를 이용한 모델 개발하기에서 개발한 모델과 동일한 트랜잭션을 처리하는 코드를 **Generate Transaction JDBC Code** 위자드를 사용하여 생성하는 방법을 설명합니다.



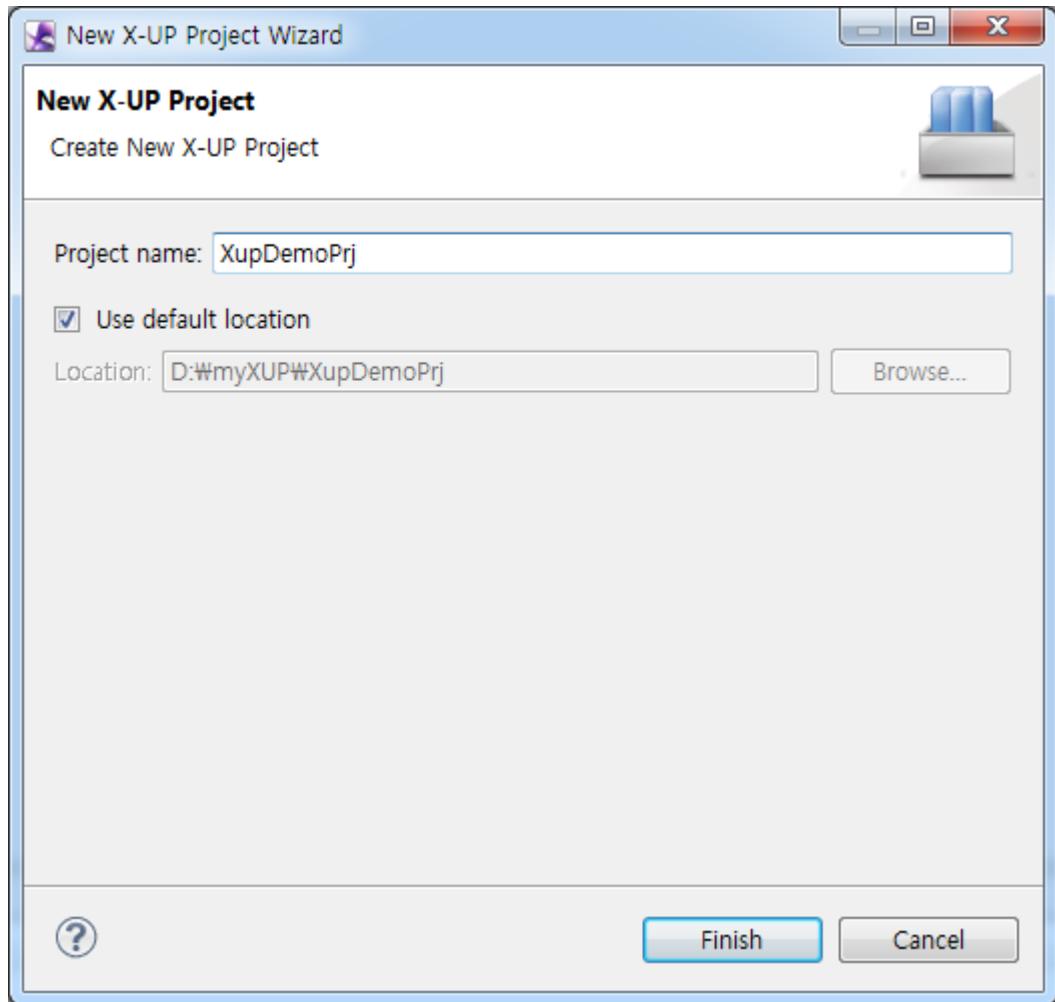
이 절에서 사용한 데이터베이스 테이블 정보는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 참조하십시오.

이 절에서 설명하는 Transaction 모델의 JDBC Transaction Code 마법사를 이용한 개발 단계는 다음과 같습니다.

- X-UP 프로젝트 생성하기
- 트랜잭션 모델 생성하기
- 입출력 파라메터 및 데이터셋 설정
- 위자드를 이용하여 모델 로직 클래스 수정하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭합니다.

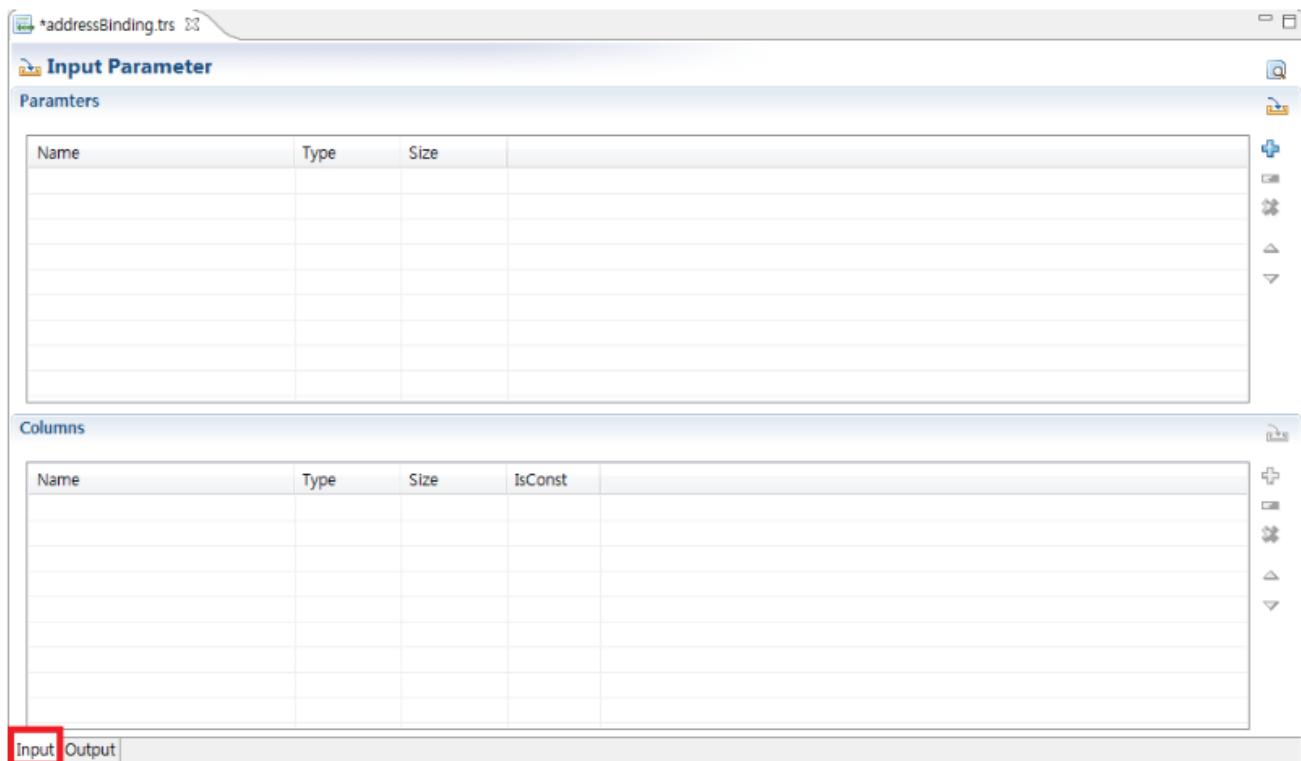


트랜잭션 모델 생성하기

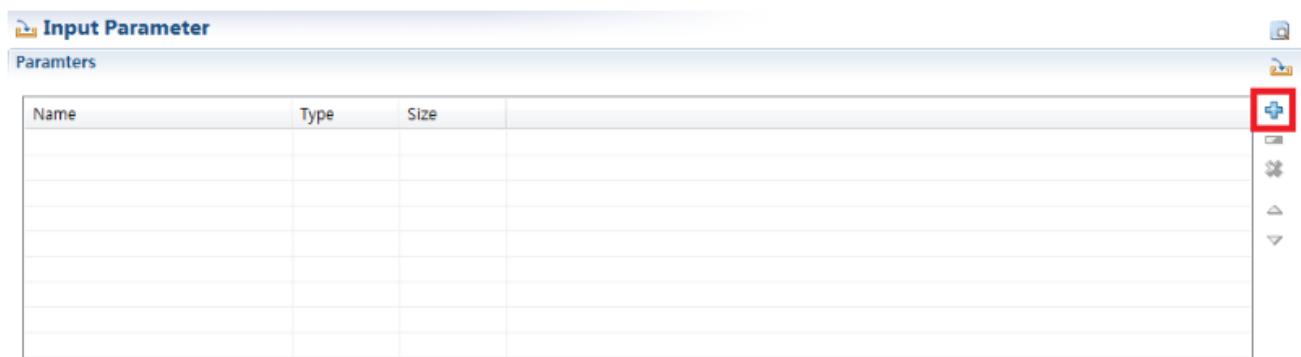
1. X-UP Explorer 뷰에서 생성한 프로젝트를 선택합니다.
2. [File > New > X-UP Transaction Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
3. New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 'Finish' 버튼을 클릭합니다.

입출력 파라메터 및 데이터셋 설정

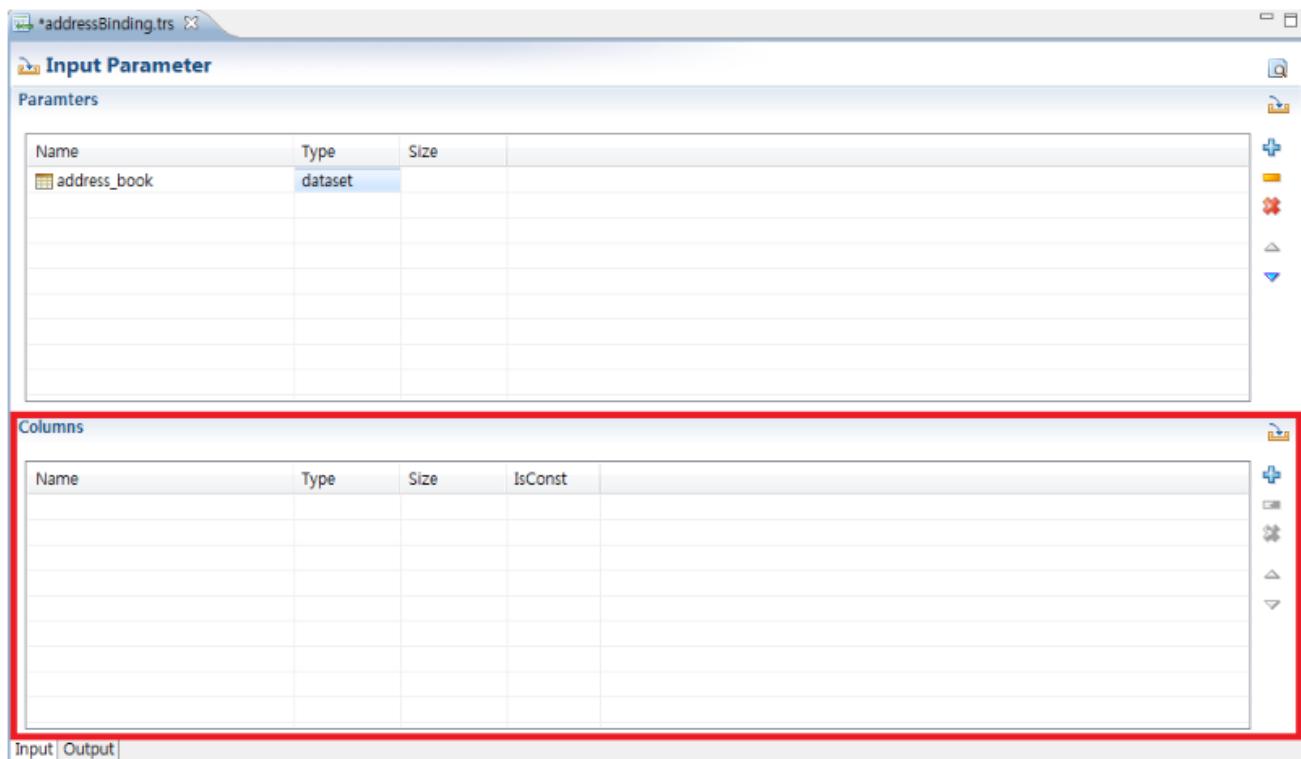
1. 트랜잭션 모델 설정 에디터에서 Input탭을 선택합니다.



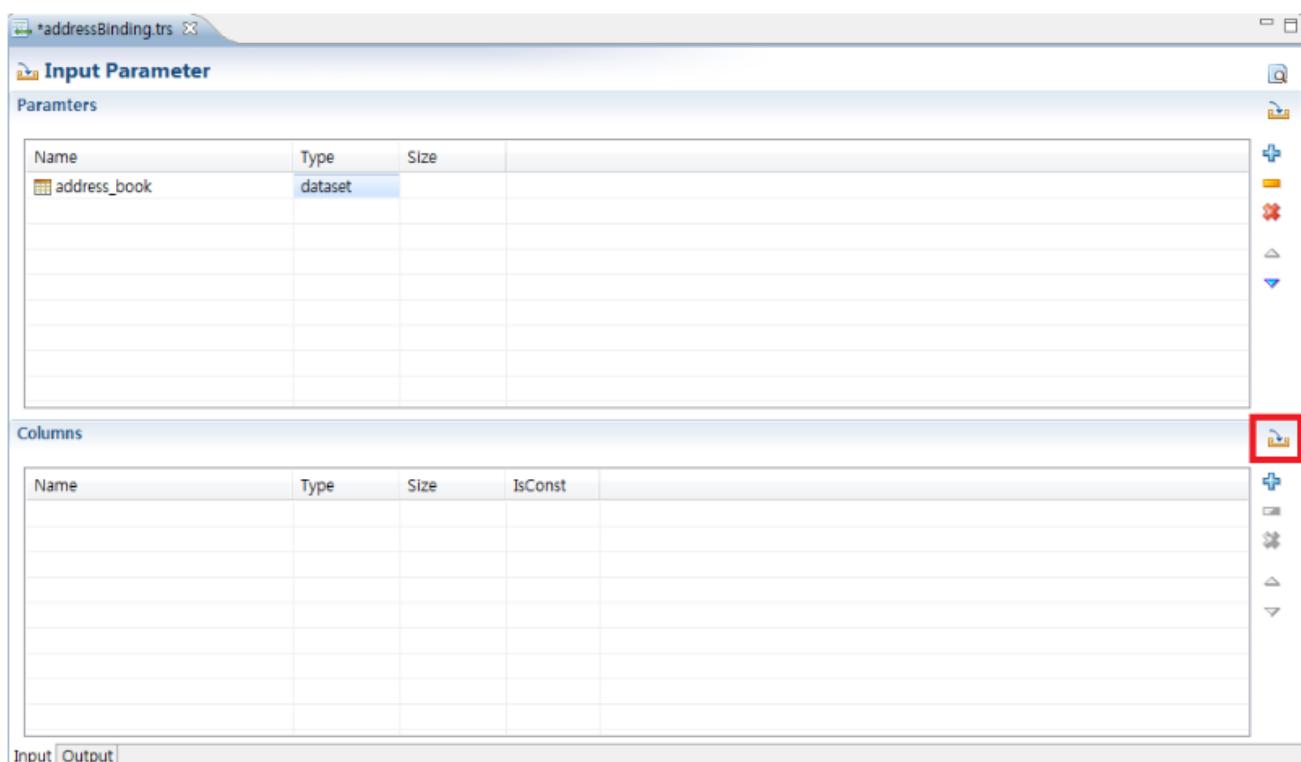
2. Parameters 설정 원도우에서 Add 버튼[+]을 클릭합니다.



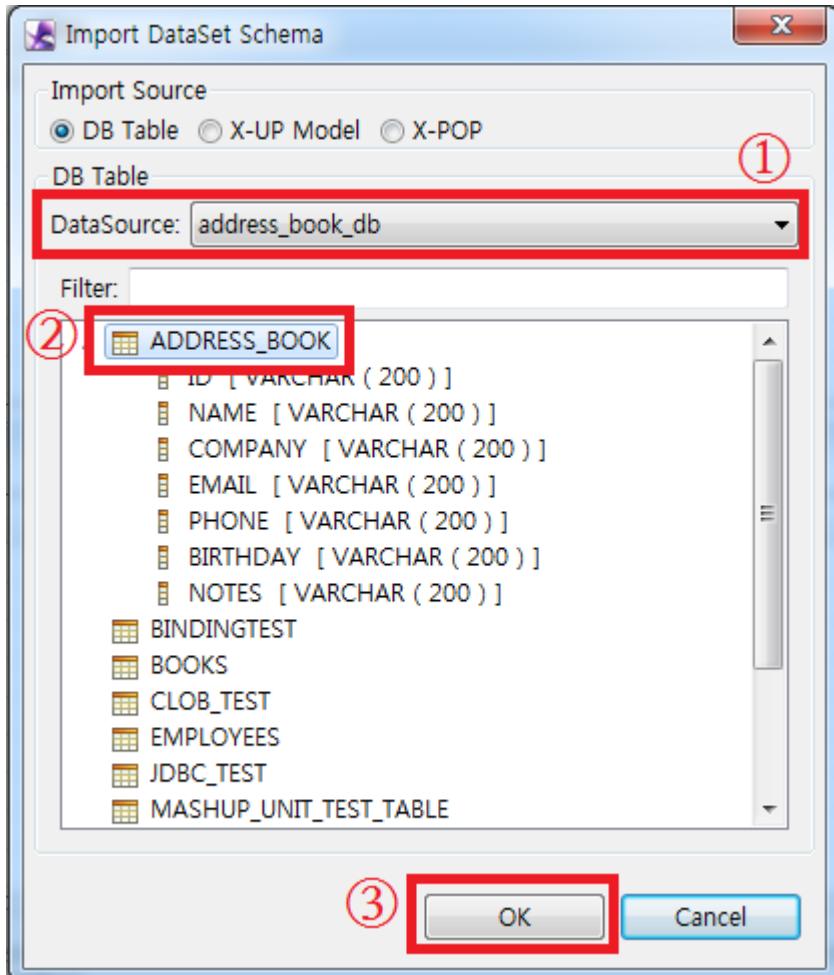
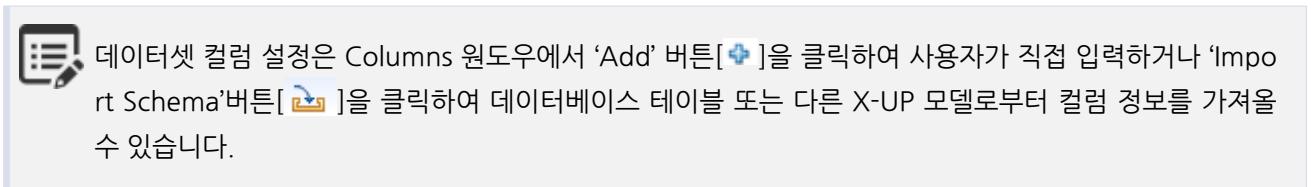
3. 파라메터의 Name은 'address_book'으로 Type은 'dataset'으로 설정합니다. 파라메터의 Type이 데이터셋인 경우에는 Columns 뷰가 활성화됩니다.



- 추가한 'address_book' 데이터셋의 컬럼을 설정합니다. Column 위치드의 Import Schema 버튼[]을 클릭하여 Import DataSet Schema 위치드를 실행합니다.



- Import 데이터셋 Schema 위치드에서 데이터소스를 선택합니다(1번). 주소록 테이블(ADDRESS_BOOK)을 선택하고(2번) OK 버튼을 클릭합니다(3번).



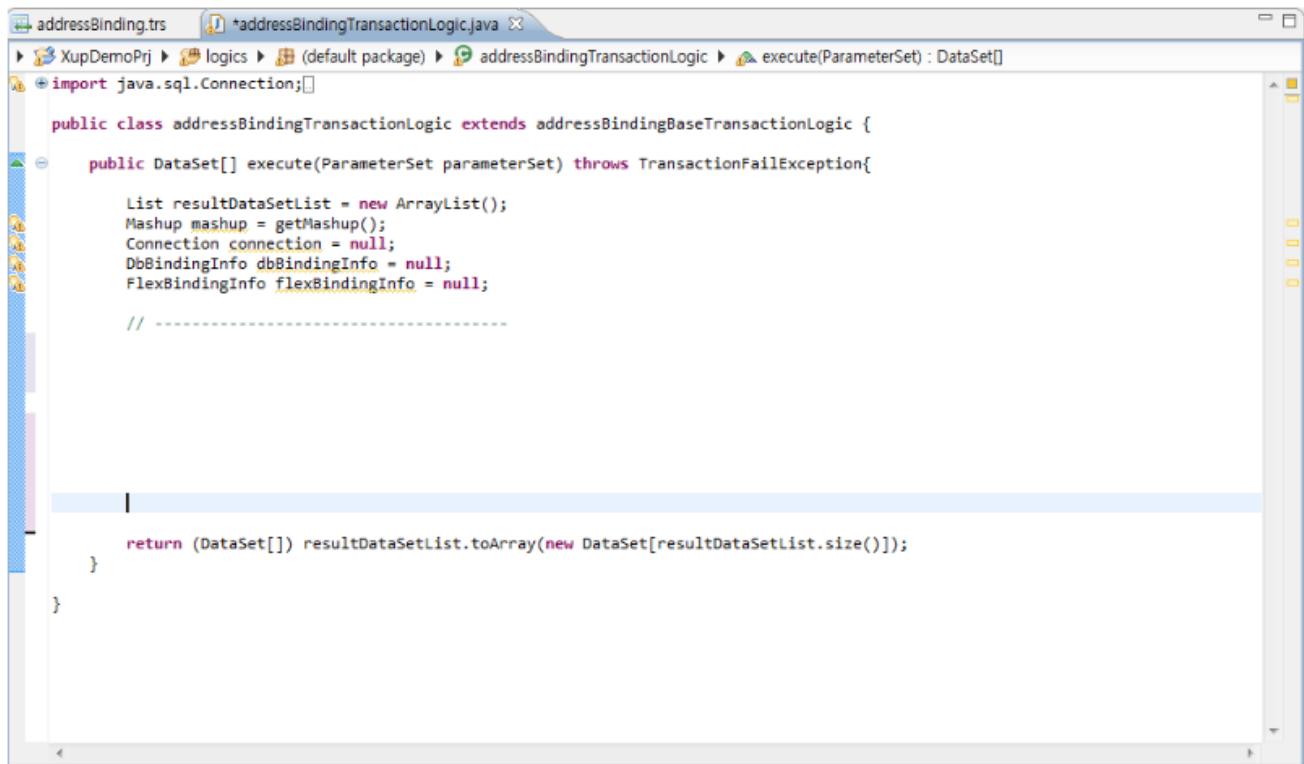
Import Source	설명
DB Table	데이터베이스의 테이블로부터 컬럼 정보를 가져옵니다. 현재 X-UP 프로젝트에 하나 이상의 DB 데이터소스가 설정되어 있어야 사용 가능합니다.
X-UP Model	현재 X-UP 프로젝트에 존재하는 다른 모델의 출력 데이터셋을 이용하여 컬럼 정보를 가져옵니다.

6. Import DataSet Schema 위자드에서 설정한 컬럼 정보가 Columns 뷰에 아래와 같이 표시됩니다.

Name	Type	Size	IsConst
ID	string	200	<input type="checkbox"/>
NAME	string	200	<input type="checkbox"/>
COMPANY	string	200	<input type="checkbox"/>
EMAIL	string	200	<input type="checkbox"/>
PHONE	string	200	<input type="checkbox"/>
BIRTHDAY	string	200	<input type="checkbox"/>
NOTES	string	200	<input type="checkbox"/>

위자드를 이용하여 모델 로직 클래스 수정하기

- 트랜잭션 모델 에디터 상단에 위치한 **Open logic class** 버튼 []을 클릭하여 로직 클래스 에디터를 실행합니다.



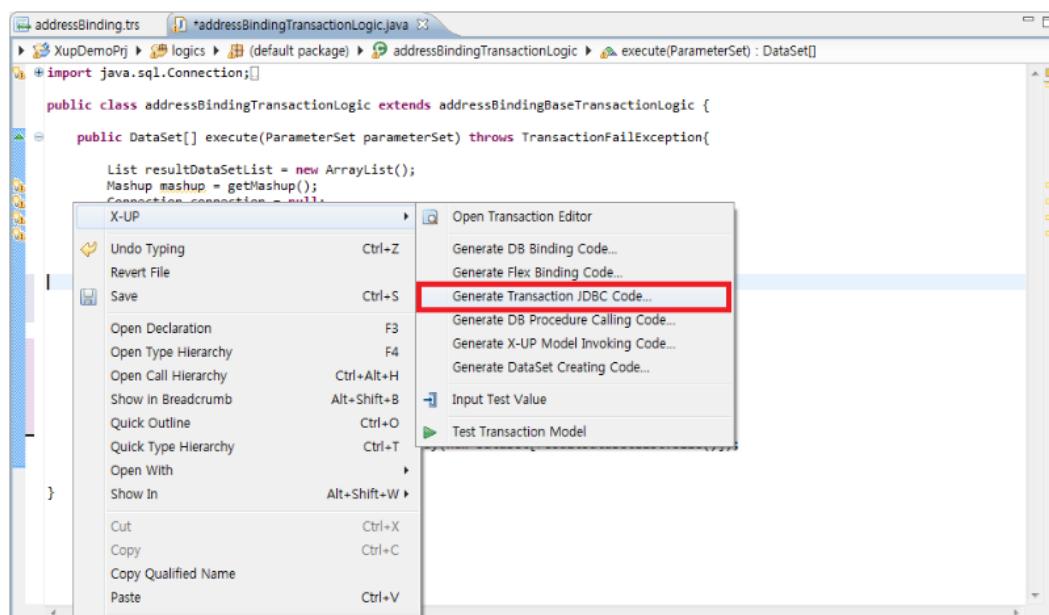
```

addressBinding.trs *addressBindingTransactionLogic.java
XupDemoPrj > logics > (default package) > addressBindingTransactionLogic > execute(ParameterSet) : DataSet[]
import java.sql.Connection;

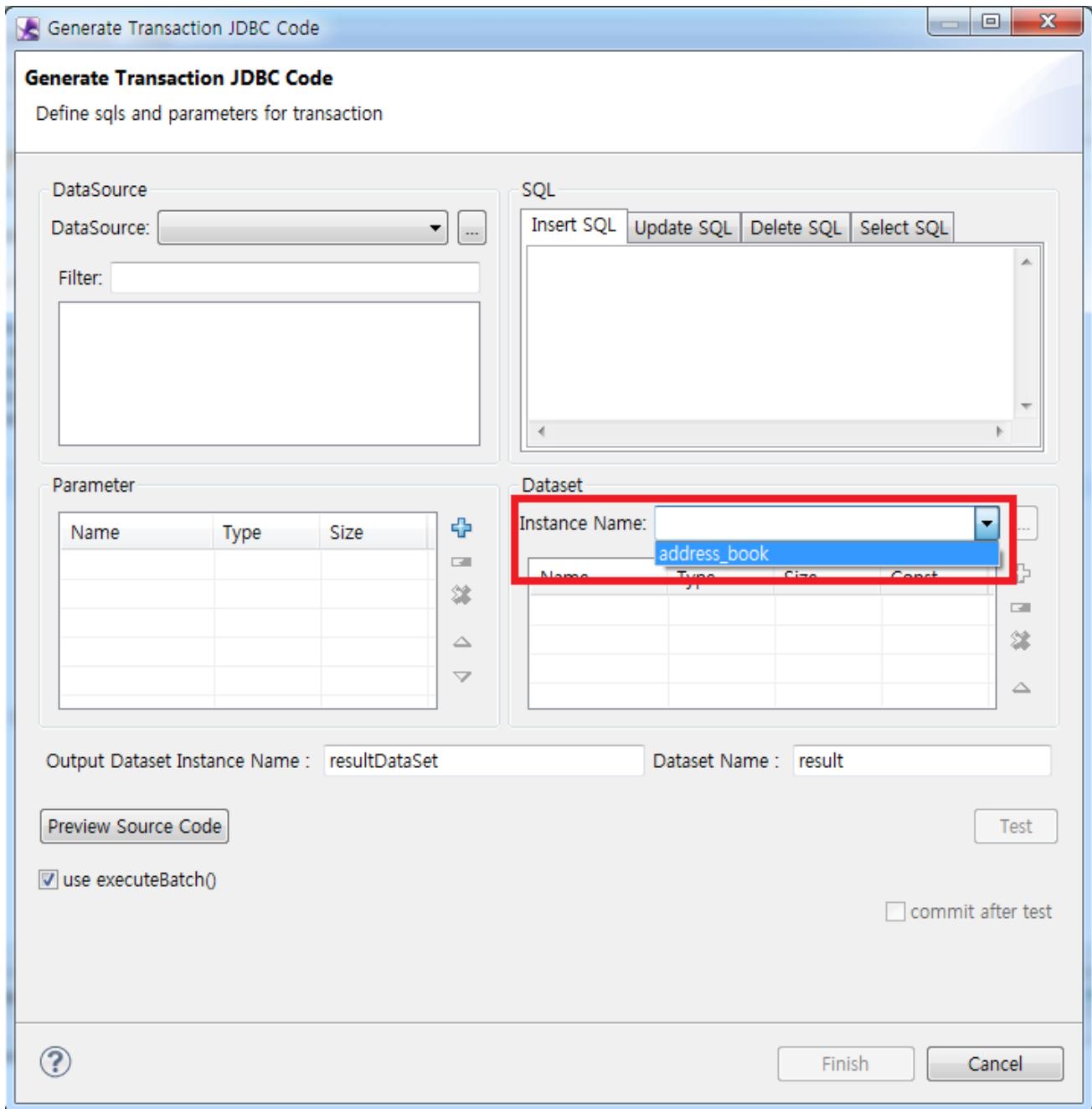
public class addressBindingTransactionLogic extends addressBindingBaseTransactionLogic {
    public DataSet[] execute(ParameterSet parameterSet) throws TransactionFailException{
        List resultDataSetList = new ArrayList();
        Mashup mashup = getMashup();
        Connection connection = null;
        DbBindingInfo dbBindingInfo = null;
        FlexBindingInfo flexBindingInfo = null;
        // -----
        return (DataSet[]) resultDataSetList.toArray(new DataSet[resultDataSetList.size()]);
    }
}

```

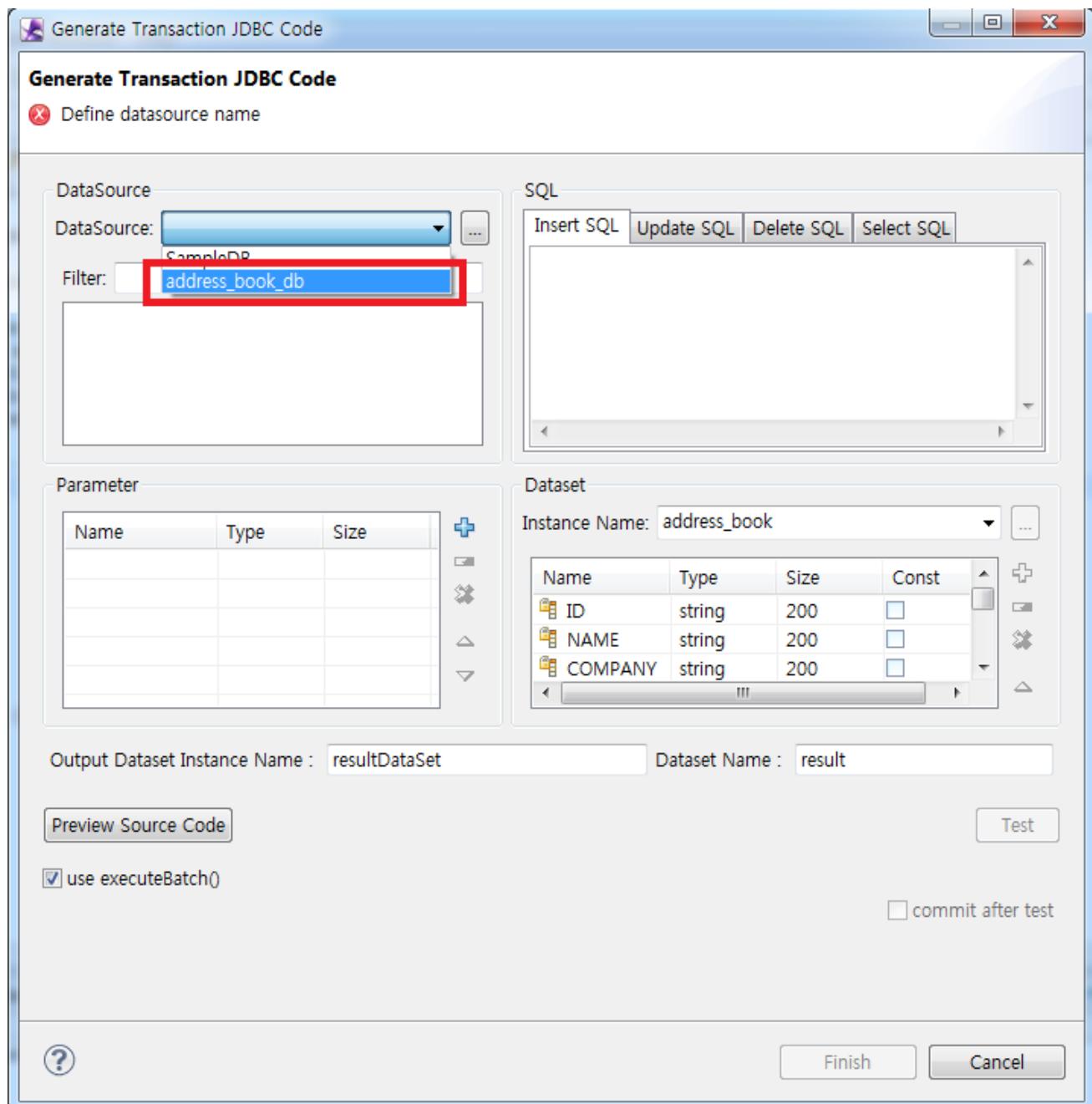
- 로직 클래스 에디터에서 주석문 ‘// ----- …’ 다음 라인으로 커서를 이동합니다. 위자드로부터 자동으로 생성된 소스는 클래스 에디터의 커서가 위치한 곳에 입력됩니다. 커서를 이동한 후 마우스 오른쪽 버튼을 클릭하고 **X-UP Generate Transaction JDBC Code**를 선택합니다.



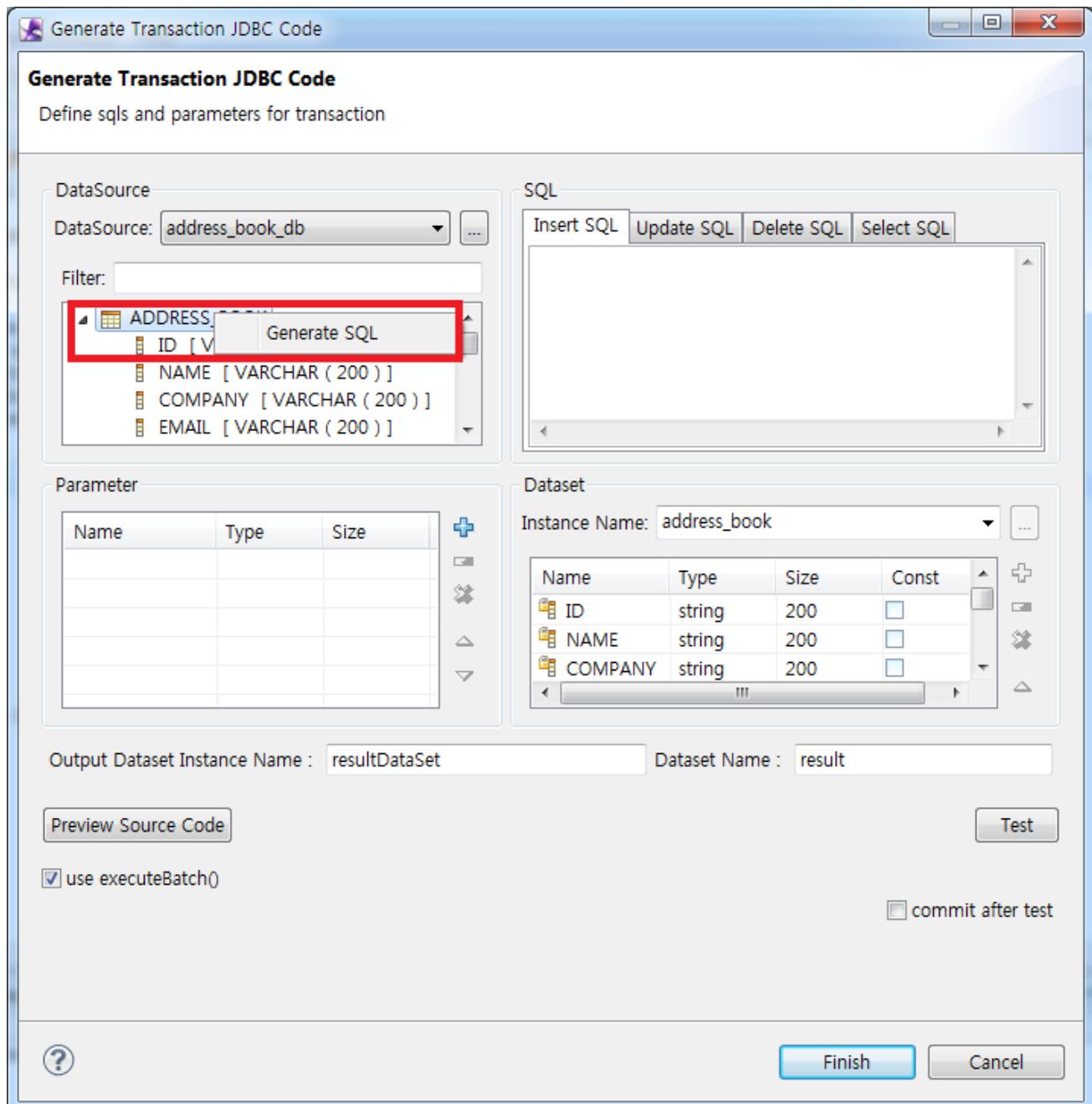
3. 'Generate Transaction JDBC Code' 위자드의 데이터셋 Instance Name 콤보박스에서 트랜잭션에 사용할 입력 데이터셋(address_book)을 선택합니다.

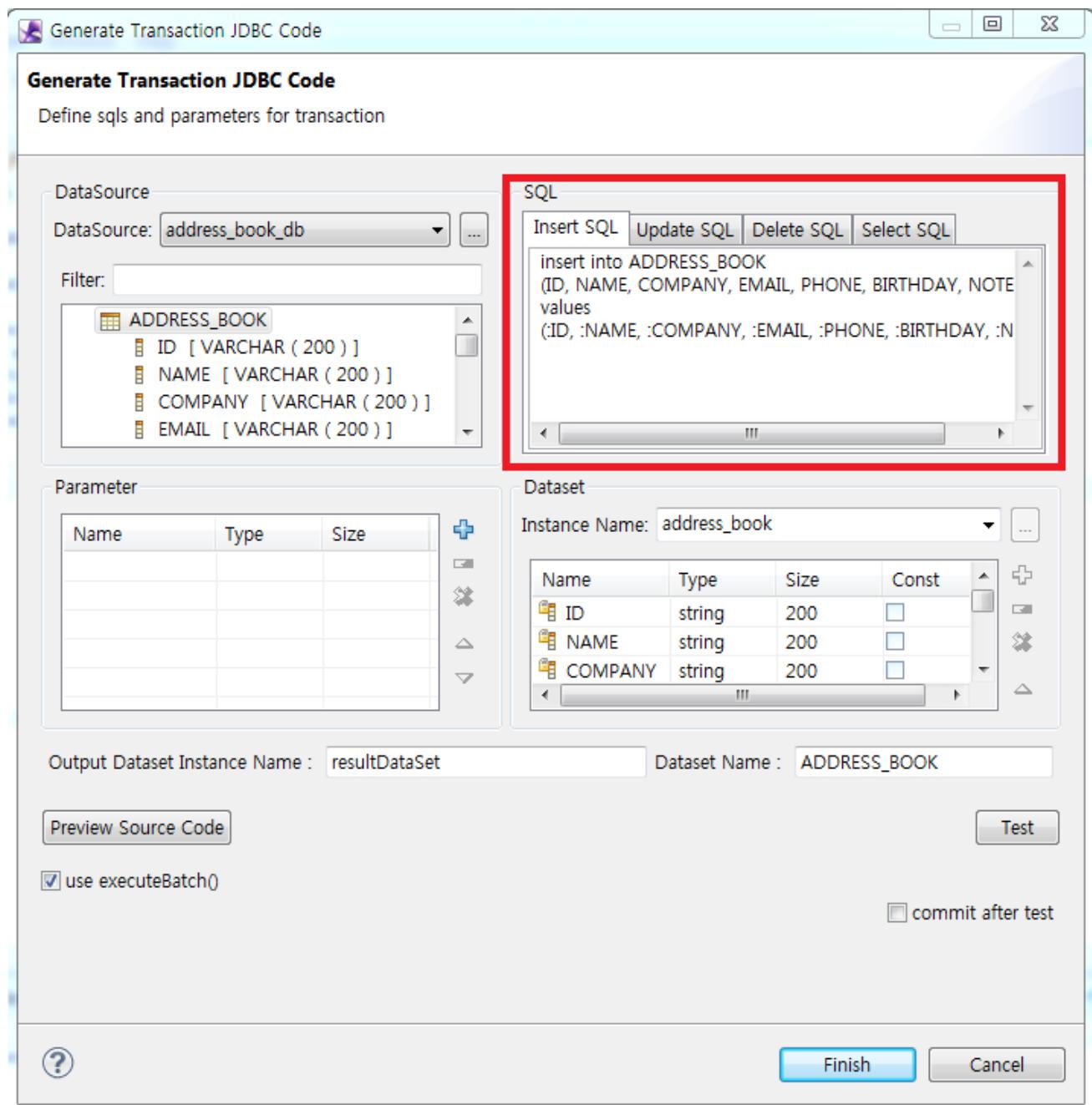


4. 'Generate Transaction JDBC Code' 위자드의 DataSource 콤보박스에서 address_book_db 데이터소스를 선택합니다.



5. 테이블 목록에서 주소록 테이블(address_book)을 선택하고 마우스 오른쪽 버튼을 클릭하면 **Generate SQL** 팝업메뉴를 볼 수 있습니다. **Generate SQL**을 선택하면 SQL 입력 원도우에 자동으로 SQL문이 생성됩니다. 생성된 SQL문을 사용자가 수정, 삭제할 수 있습니다. 예제에서는 기본으로 생성된 SQL문을 그대로 사용합니다.





6. ‘Finish’ 버튼을 클릭하여 생성한 코드를 모델 로직 클래스에 추가합니다.
7. Generate Transaction JDBC Code 위치드가 정상적으로 완료되면 로직 클래스 에디터에 다음과 같은 소스가 추가됩니다.

```
public DataSet[] execute(ParameterSet parameterSet) throws TransactionFailException {
    List resultDataSetList = new ArrayList();
    Mashup mashup = getMashup();
    Connection connection = null;
    DbBindingInfo dbBindingInfo = null;
    FlexBindingInfo flexBindingInfo = null;
```

```

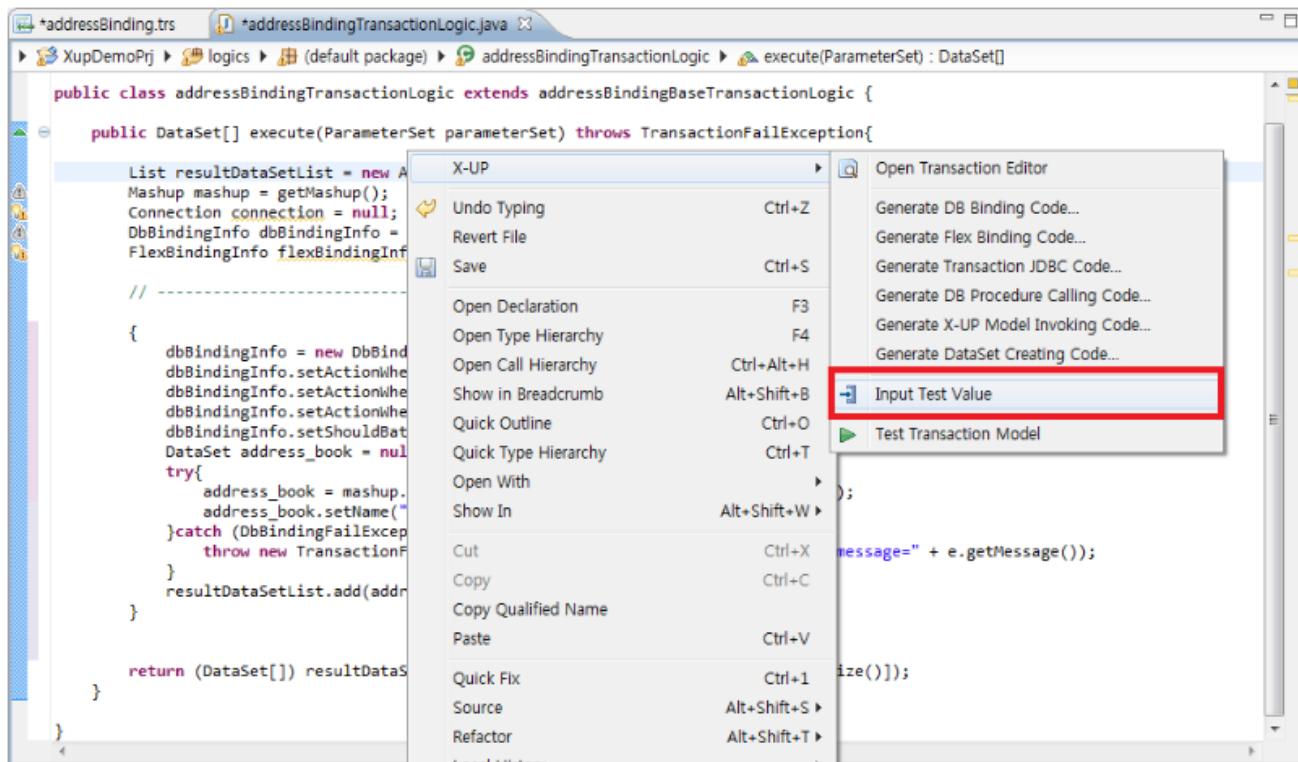
// -----
// Generate Transaction JDBC Code
{
    DataSet resultDataSet = null;
    try {
        connection = getConnection( "address_book_db" );
        JdbcActor jdbcActor = new addressBindingJdbcActor0();
        jdbcActor.setDataSet(parameterSet.getDataSet( "address_book" ));
        jdbcActor.setParameterSet(parameterSet);
        jdbcActor.setConnection(connection);
        resultDataSet = jdbcActor.run();
        resultDataSet.setName( "ADDRESS_BOOK" );
    } catch (InvalidDataSourceException e) {
        throw new TransactionFailException( "transaction failed. e= " + e+
            " , message= " +e.getMessage());
    } catch (SQLException e) {
        throw new TransactionFailException( "transaction failed. e= " + e+
            " , message= " +e.getMessage());
    } finally {
    }
    resultDataSetList.add(resultDataSet);
}
return (DataSet[]) resultDataSetList.toArray(new
    DataSet [resultDataSetList.size()]);
}

```

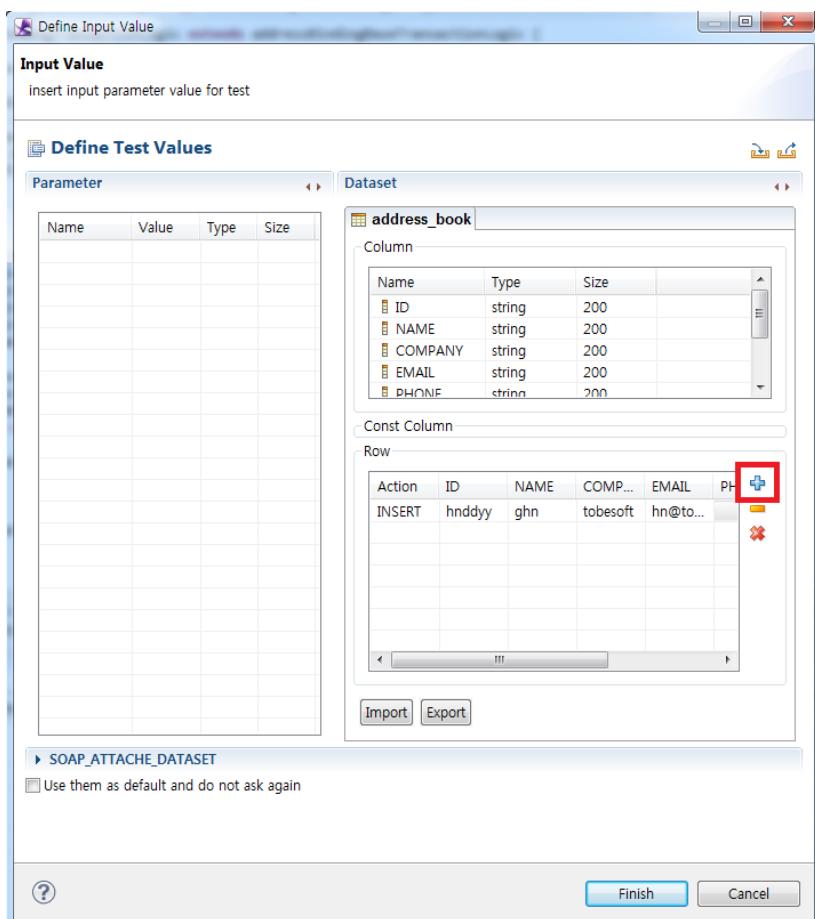
8. File Save 메뉴를 클릭하여 개발한 모델을 저장합니다.

모델 테스트하기

1. 트랜잭션 모델을 테스트합니다. 테스트에 앞서 테스트 데이터를 입력 합니다. 로직 클래스 에디터에서 마우스 오른쪽 버튼을 클릭하여 X-UP Input Test Value를 선택 합니다.

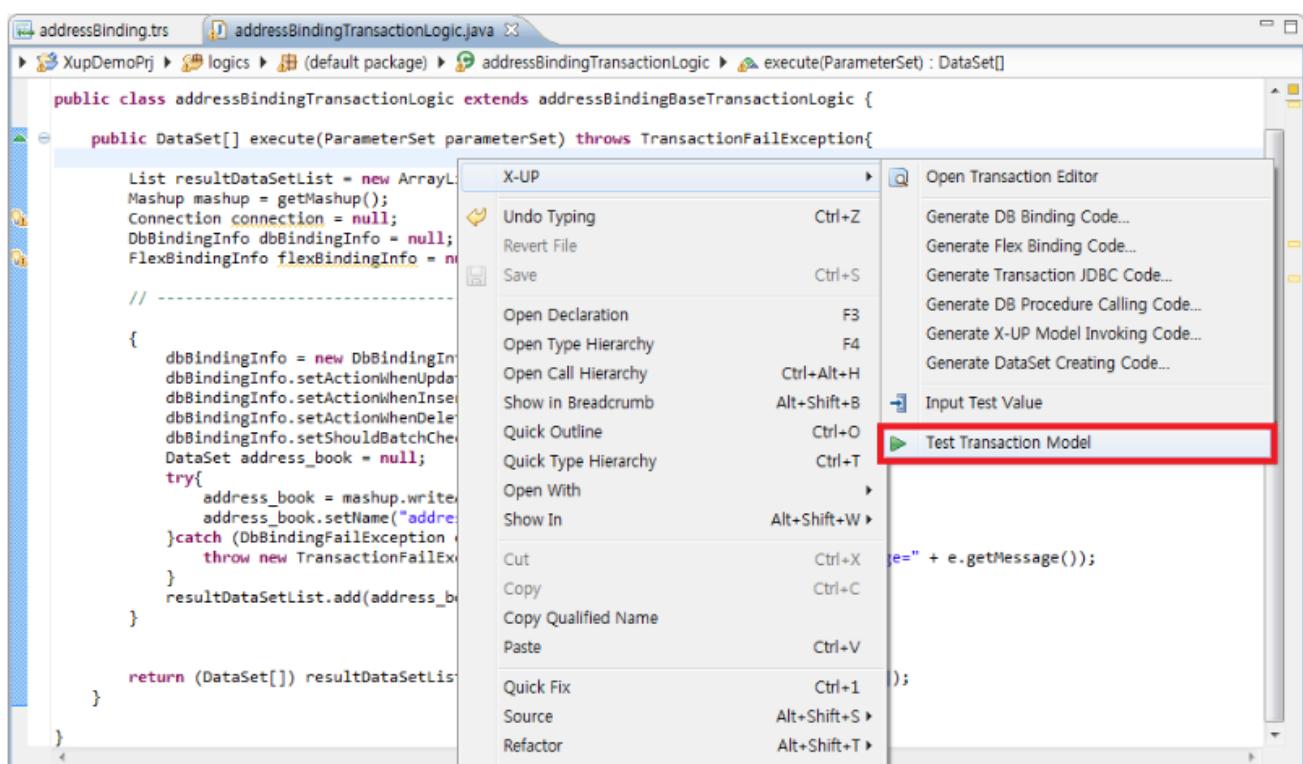


2. 트랜잭션 모델 테스트에 사용할 데이터를 입력합니다. **Input Value** 원도우에서 **Add** 버튼[]을 클릭하여 데잍를 입력합니다.



Action	설명
INSERT	'INSERT'가 설정된 Row는 DB 테이블에 추가됩니다.
UPDATE	'UPDATE'가 설정된 Row는 테이블에서 동일한 레코드를 찾아 업데이트를 수행합니다. 동일한 레코드는 기본 키(Primary Key) 컬럼의 값이 입력 데이터셋의 값과 동일한 레코드를 의미합니다. 그러므로 하나 이상의 기본 키 컬럼이 설정된 테이블에서만 'UPDATE'를 수행 할 수 있습니다.
DELETE	'DELETE'가 설정된 Row는 테이블에서 동일한 레코드를 찾아 삭제합니다. 동일한 레코드는 기본 키(Primary Key) 컬럼의 값이 입력 데이터셋의 값과 동일한 레코드를 의미합니다. 그러므로 하나 이상의 기본 키 컬럼이 설정된 테이블에서만 'DELETE'를 수행할 수 있습니다.
NORMAL	입력 데이터로만 사용되며 DB 트랜잭션에는 관여하지 않습니다.

3. 로직 클래스 에디터에서 마우스 오른쪽 버튼을 클릭하여 X-UP Test Transaction Model을 선택합니다.



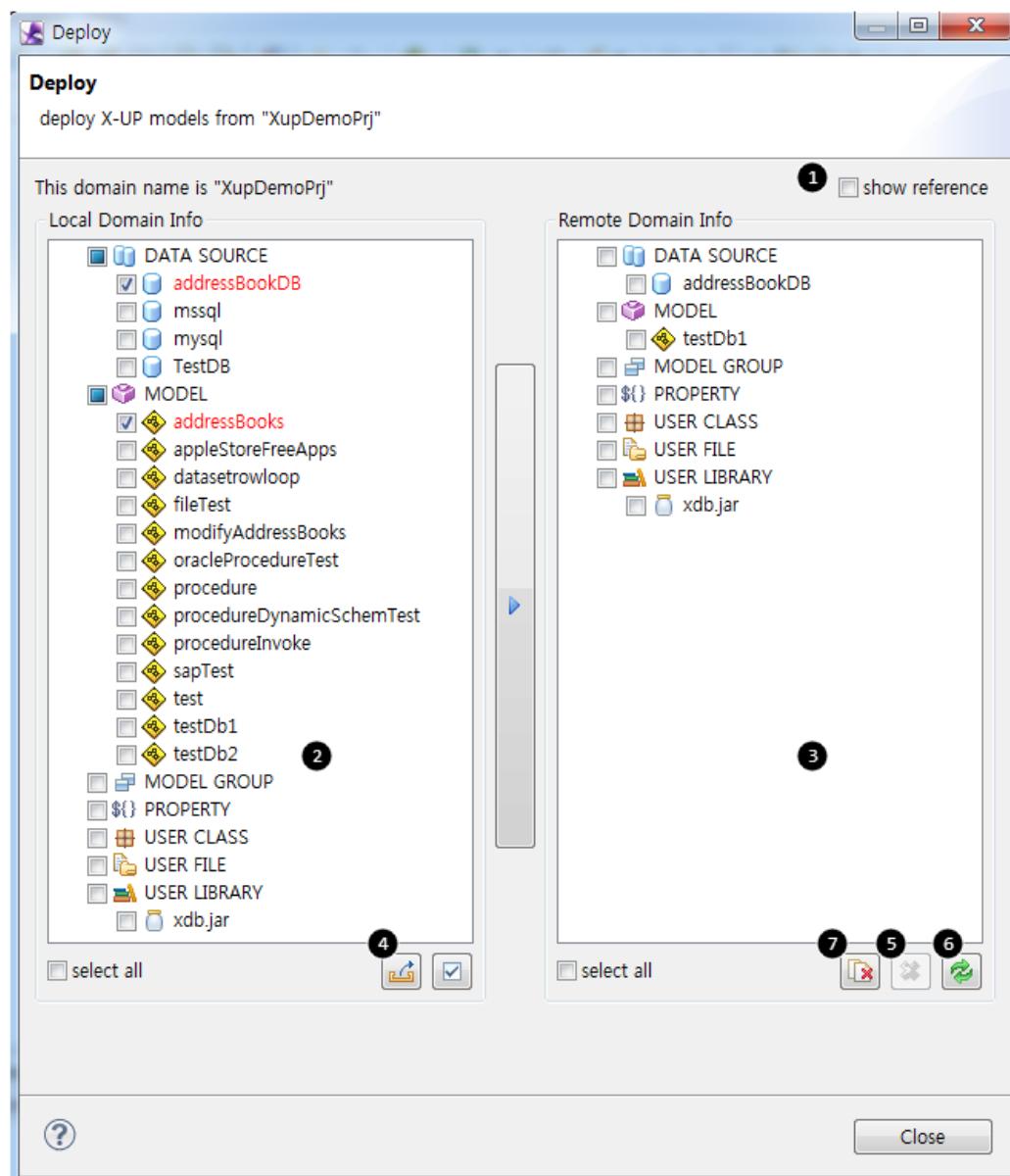
테스트가 완료되면 Result ParameterSet 뷰에서 결과를 확인 할 수 있습니다.

10.

Deploy & Undeploy

Deploy X-UP Model Dialog는 서버에 배치된 모델들을 조회하고 개발 완료된 X-UP 모델들을 서버에 배치하거나 서버에 배치된 모델을 삭제할 수 있는 기능을 제공합니다.

Deploy X-UP Model Dialog는 select project  Deploy X-UP Model을 통해 실행할 수 있습니다.



	Name	Description
①	show reference	각 항목별 연관된 모든 정보를 트리에 표시.
②	Local	로컬 리스트
③	Remote	서버 리스트
④	Deploy	선택한 항목을 서버에 배치
⑤	Remove	선택한 항목을 서버에서 삭제
⑥	Refresh	서버 리스트 갱신
⑦	Clear All in Domain	서버에서 현 도메인 삭제

10.1 모델을 X-UP서버에 Deploy하기

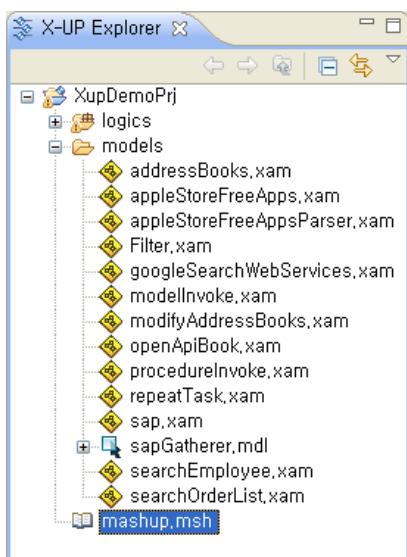
이 절에서는 X-UP Builder에서 사용자가 개발한 모델에 대해 서버에 배치하는 방법에 설명합니다.

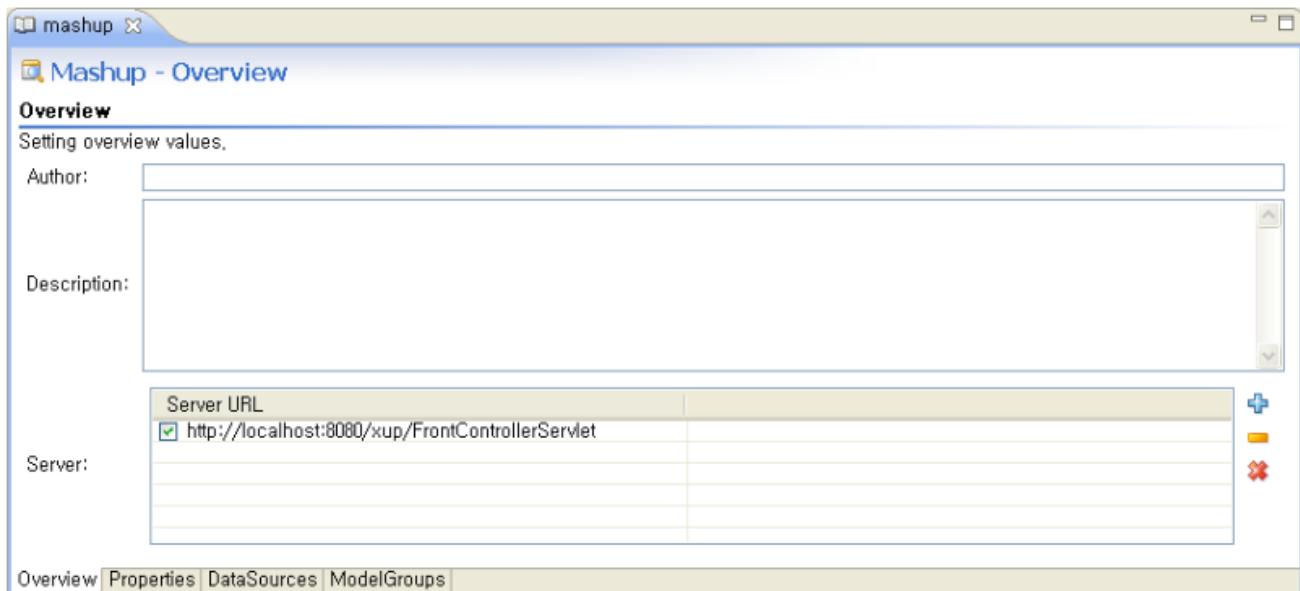
이 절에서 설명하는 모델 배치 단계는 다음과 같습니다.

- 배치할 서버 설정하기
- X-UP 모델 배치하기
- 웹브라우저로 배치된 모델 테스트 하기

배치할 서버 설정하기

1. X-UP Explorer 화면의 `mashup.msh` 파일을 더블 클릭하면 아래와 같은 Mashup - Overview 화면이 나타납니다.



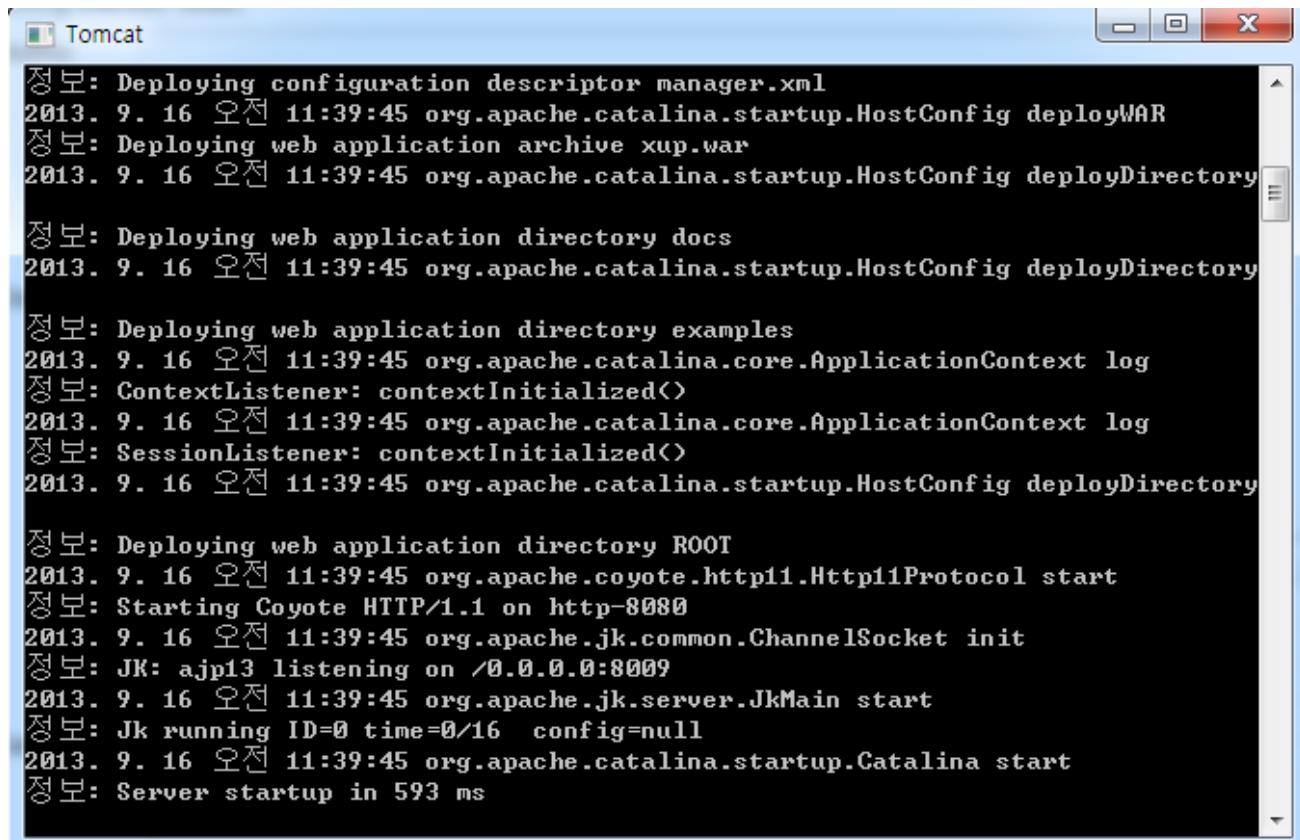


2. Mashup-Overview 화면에서 모델 배치를 위한 아래 사항들을 입력합니다.

Field Name	Field Value
Author	'xup tester' 필수입력사항이 아니라 입력하지 않으셔도 무방합니다.
Description	'This is a X-UP Server' 배치해야 할 X-UP서버에 대한 설명을 입력합니다. 필수입력사항이 아니라 입력하지 않으셔도 무방합니다.
Servers	'http://localhost:8080/xup/FrontControllerServlet.do' [+]버튼을 입력하여 배치해야 할 X-UP서버 URL을 입력합니다. X-UP 서버 URL은 http://[host name:port]/xup/[servlet mapping name] 으로 구성됩니다. 복수의 X-UP 서버를 설정하고 체크버튼으로 배치할 서버를 선택할 수 있습니다. X-UP 서버의 기본 Servlet Url Pattern 매팅정보는 *.do 입니다.

X-UP 모델 배치하기

1. WAS를 실행합니다. 본 매뉴얼에서는 톰캣 6.0을 기준으로 작성하였습니다.



```
정보: Deploying configuration descriptor manager.xml
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployWAR
정보: Deploying web application archive xup.war
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

정보: Deploying web application directory docs
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

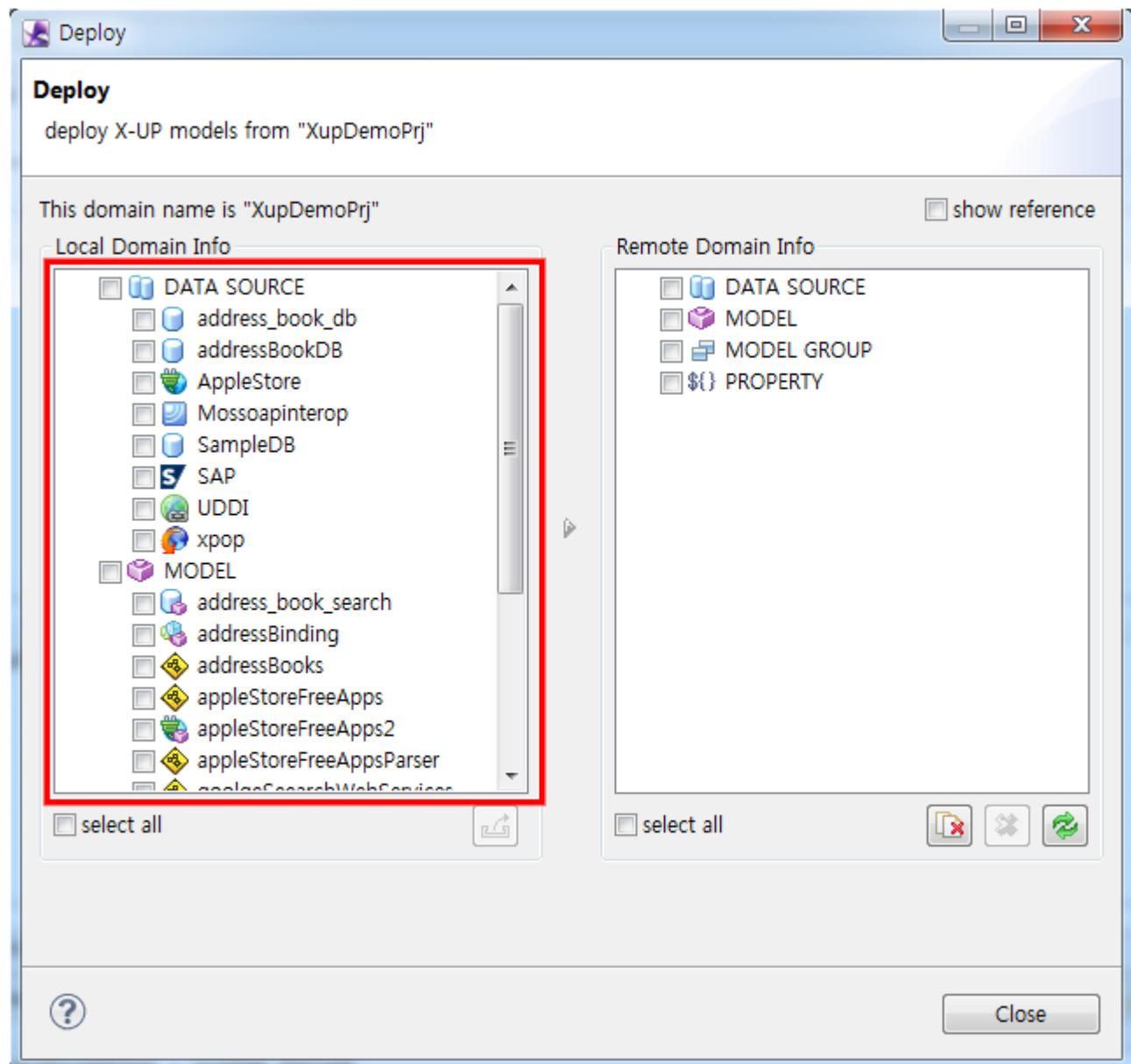
정보: Deploying web application directory examples
2013. 9. 16 오전 11:39:45 org.apache.catalina.core.ApplicationContext log
정보: ContextListener: contextInitialized()
2013. 9. 16 오전 11:39:45 org.apache.catalina.core.ApplicationContext log
정보: SessionListener: contextInitialized()
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

정보: Deploying web application directory ROOT
2013. 9. 16 오전 11:39:45 org.apache.coyote.http11.Http11Protocol start
정보: Starting Coyote HTTP/1.1 on http-8080
2013. 9. 16 오전 11:39:45 org.apache.jk.common.ChannelSocket init
정보: JK: ajp13 listening on /0.0.0.0:8009
2013. 9. 16 오전 11:39:45 org.apache.jk.server.JkMain start
정보: Jk running ID=0 time=0/16 config=null
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.Catalina start
정보: Server startup in 593 ms
```

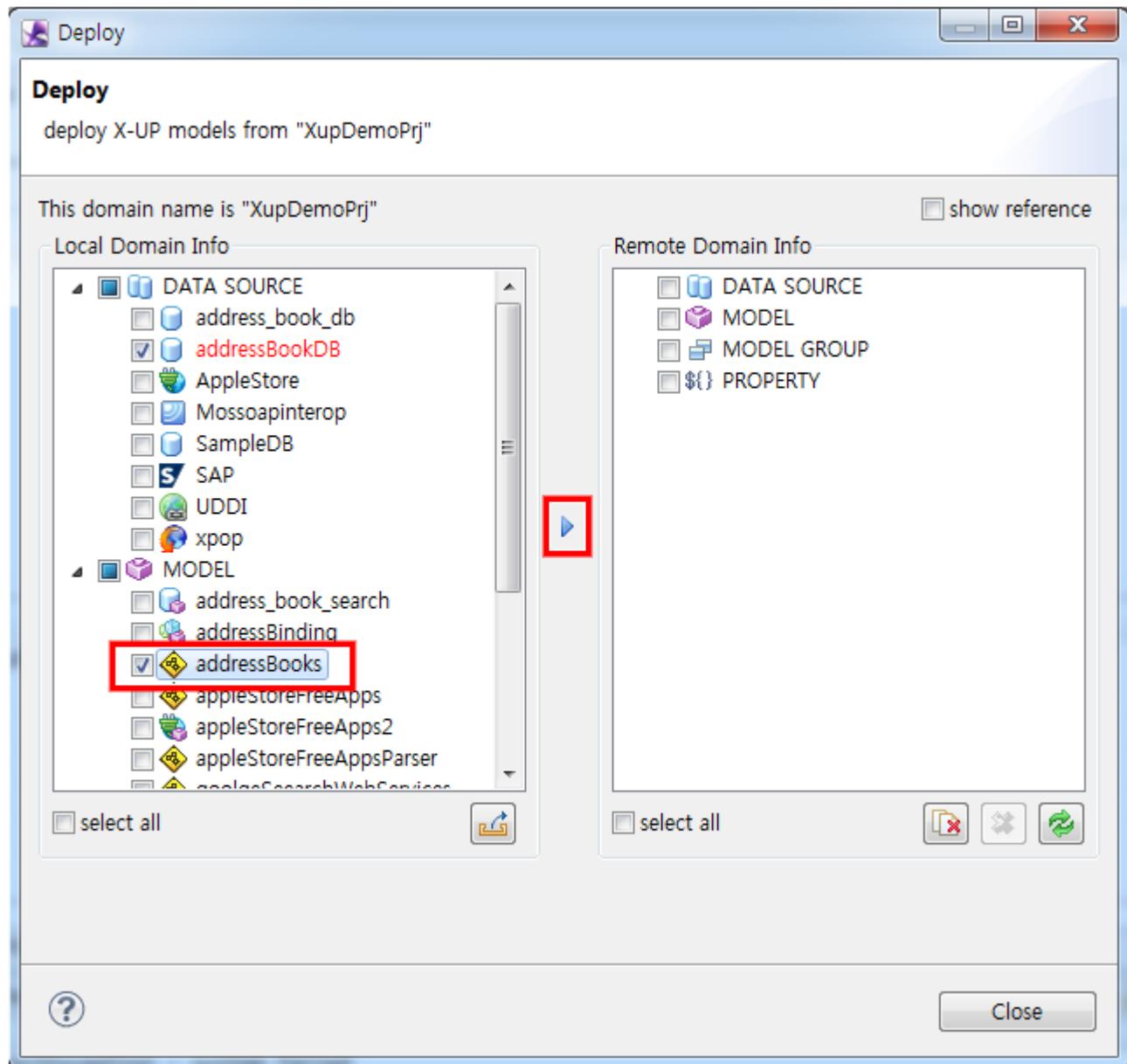
2. X-UP Explorer의 mashup.msh 파일을 선택한 후 툴바의 Deploy X-UP Model [] 아이콘을 클릭 합니다.



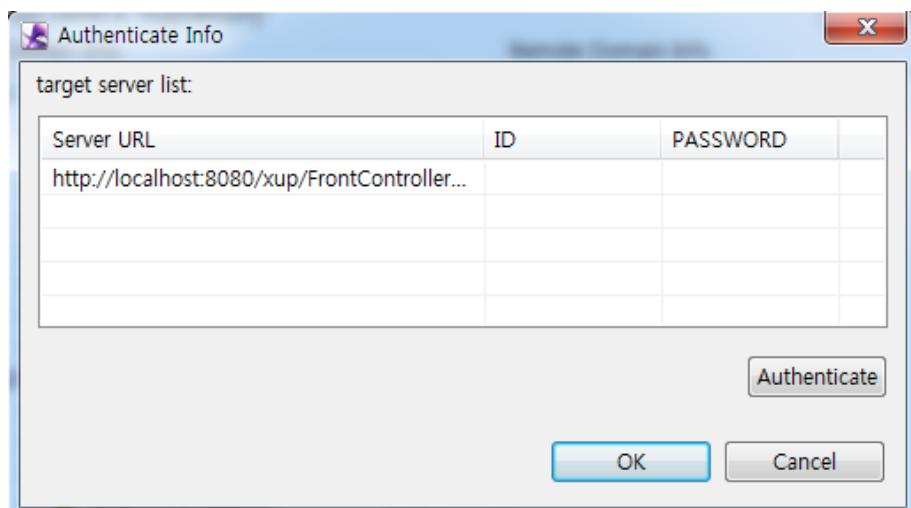
3. 아래 그림과 같이 Deploy 위치드가 활성화 되면 Local에서 개발한 모델 리스트가 나타납니다.



4. 모델 리스트에서 배치할 모델을 선택한 후 [▶] 아이콘을 클릭하면 서버에 배치를 수행 합니다.



5. 배치를 수행 하기 전 아래의 그림과 같이 서버의 인증 화면이 나타납니다.

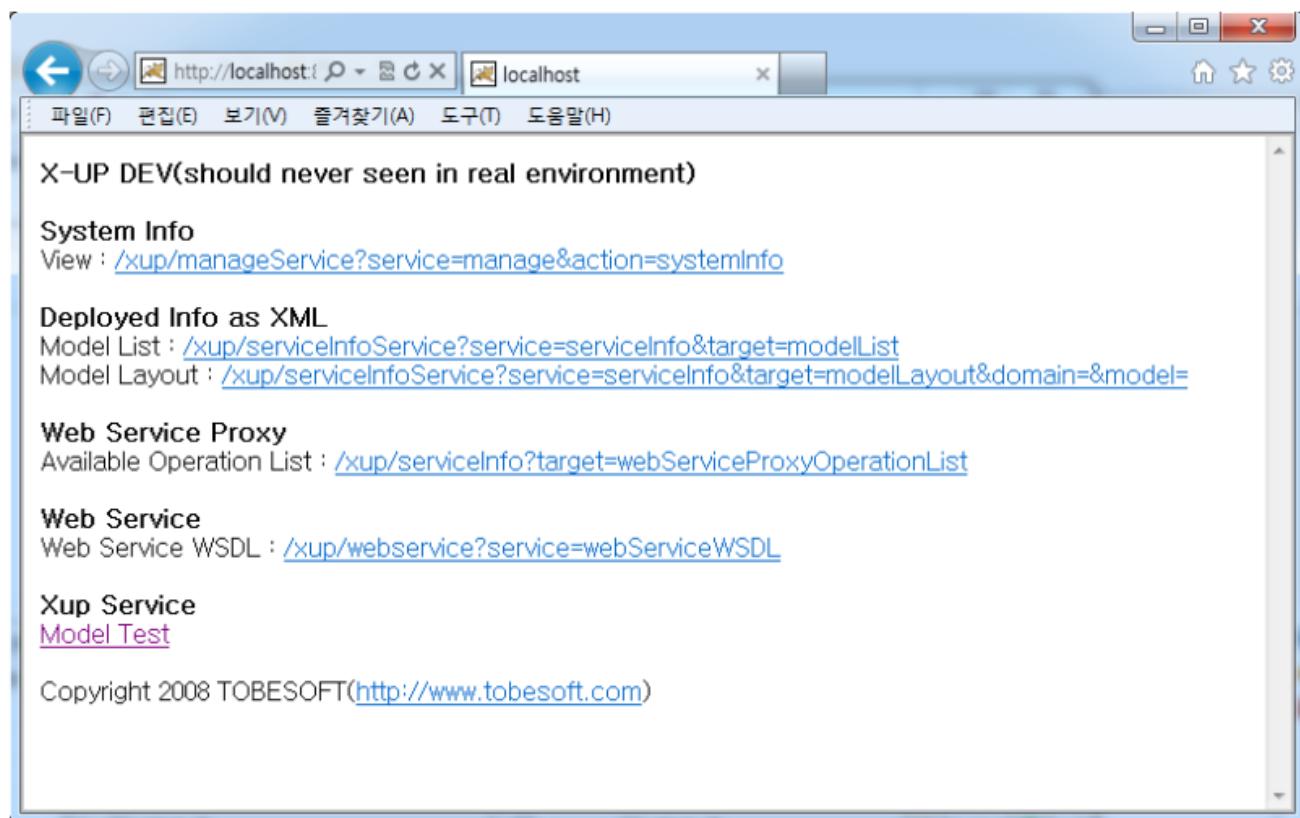


6. X-UP 설치 시 기본 계정 정보는 xup/xup 입니다. 그림과 같이 id/password를 입력 후 “OK” 버튼을 클릭하여 배치를 완료 합니다.

웹브라우저로 배치된 모델 테스트 하기

이 절에서는 X-UP 서버에 배치된 모델을 웹브라우저에서 테스트하는 방법을 설명합니다.

1. 웹브라우저 주소 입력창에 X-UP 서버 URL(<http://localhost:8080/xup/FrontControllerServlet.do>) 을 입력하여 X-UP 테스트 웹페이지를 호출합니다.



2. X-UP 테스트 웹페이지에서는 X-UP 서버에 배치된 모델리스트를 확인하고 테스트 할 수 있습니다.
3. Xup Service 항목의 Model Test 링크를 클릭하면 아래와 같은 X-UP Model List 가 화면에 보입니다.

TestForModule	Model Name	Description	Transaction Type
TestForModule	msssoapinteropTest	this is msssoapinteropTest model	Msssoapinterop
TestForModule	appleStoreTop10Apps	this is appleStoreTop10Apps model	appleStoreFreeApp, msssoapinteropTes
TestForModule	testXpop	this is testXpop model	xpop
TestForModule	xpop	this is xpop model	xpop
TestForModule	appleStoreFree	null	
TestForModule	webservicetest	this is webservicetest model	SampleWebService
TestForModule	addressBinding	this is addressBinding model	
TestForModule	address_book_search	this is address_book_search model	SampleDB
XupDemoPrj	addressBooks	ADDRESS_BOOK of DB to query data from a table.	

4. 테스트에 사용 할 모델명 “address_book_search” 을 클릭하여 나타나는 X-UP Model Test화면은 아래와 같습니다.

X-UP Model TEST

Domain : XupDemoPrj

Model : addressBooks

Parameter	Alias	Type	Value	Size	Description	Default Value
IN_DS	IN_DS	dataset		-1		

Service Type and Format: MiPlatform XML

[RUN]

5. 아래와 같이 모델 호출 시 필요한 Input Parameter를 입력하고 [RUN] 버튼을 클릭합니다.

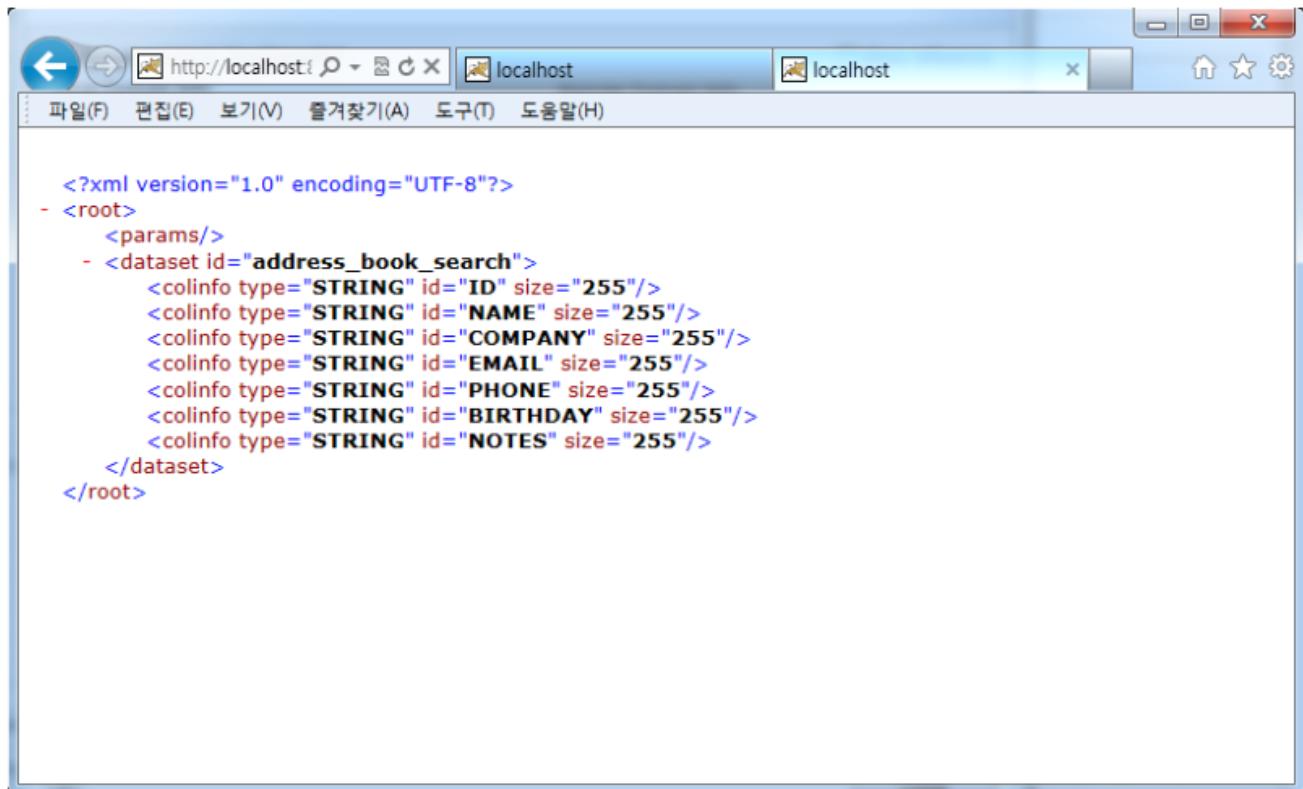
X-UP Model TEST

Domain : XupDemoPrj

Model : address_book_search

Parameter	Alias	Type	Value	Size	Description	Default Value
NAME	NAME	string	Rex	255		

6. 테스트 결과 화면은 아래와 같습니다



The screenshot shows a web browser window with the URL `http://localhost`. The page content displays the XML configuration for a dataset named `address_book_search`. The XML structure includes a root node `<root>`, which contains a `<params>` node and a `<dataset id="address_book_search">` node. The `<dataset>` node is defined with the following parameters:

```

<?xml version="1.0" encoding="UTF-8"?>
- <root>
  <params/>
  - <dataset id="address_book_search">
    <colinfo type="STRING" id="ID" size="255"/>
    <colinfo type="STRING" id="NAME" size="255"/>
    <colinfo type="STRING" id="COMPANY" size="255"/>
    <colinfo type="STRING" id="EMAIL" size="255"/>
    <colinfo type="STRING" id="PHONE" size="255"/>
    <colinfo type="STRING" id="BIRTHDAY" size="255"/>
    <colinfo type="STRING" id="NOTES" size="255"/>
  </dataset>
</root>

```

10.2 X-UP 서버의 모델을Undeploy하기

이 절에서는 X-UP Builder에서 사용자가 개발한 모델에 대해 Undeploy하는 방법에 설명합니다.

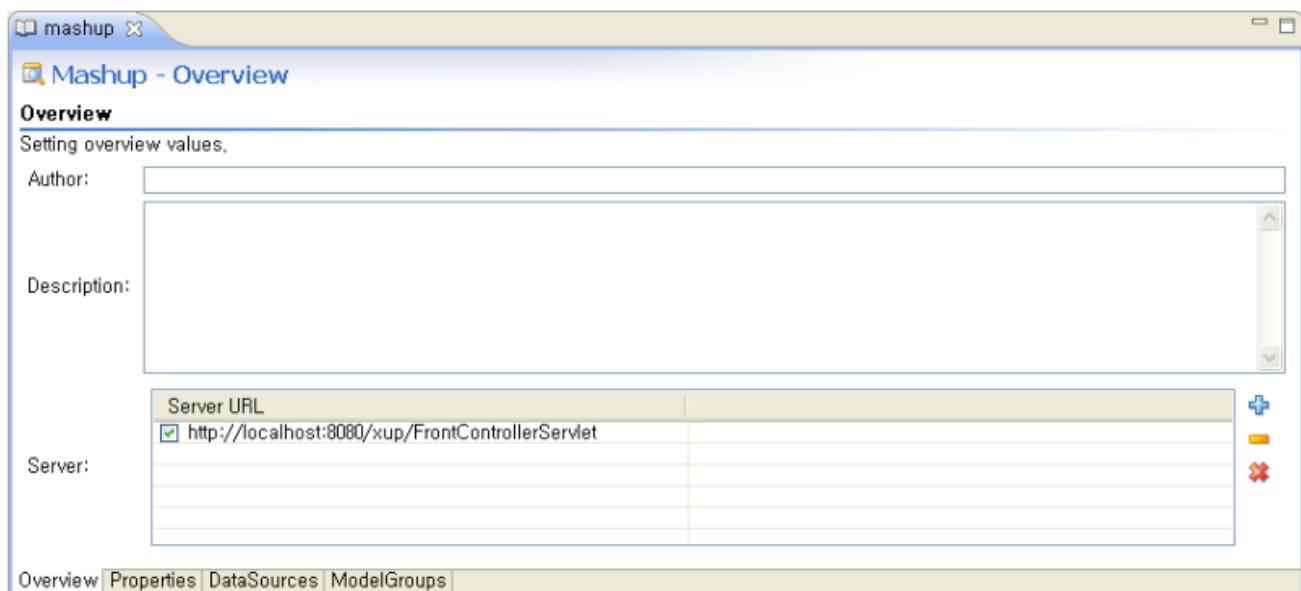
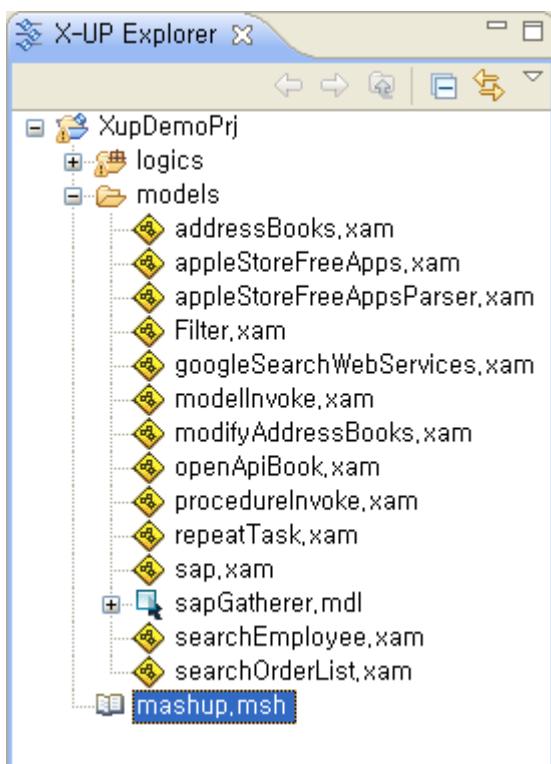
이 절에서 설명하는 Undeploy 단계는 다음과 같습니다.

- Undeploy할 서버 설정하기

- X-UP 모델 Undeploy하기
- 도메인안에 있는 전체 모델에 대해 Undeploy 하기

Undeploy할 서버 설정하기

1. X-UP Explorer 화면의 `mashup.msh` 파일을 더블 클릭하면 아래와 같은 Mashup - Overview 화면이 나타납니다.



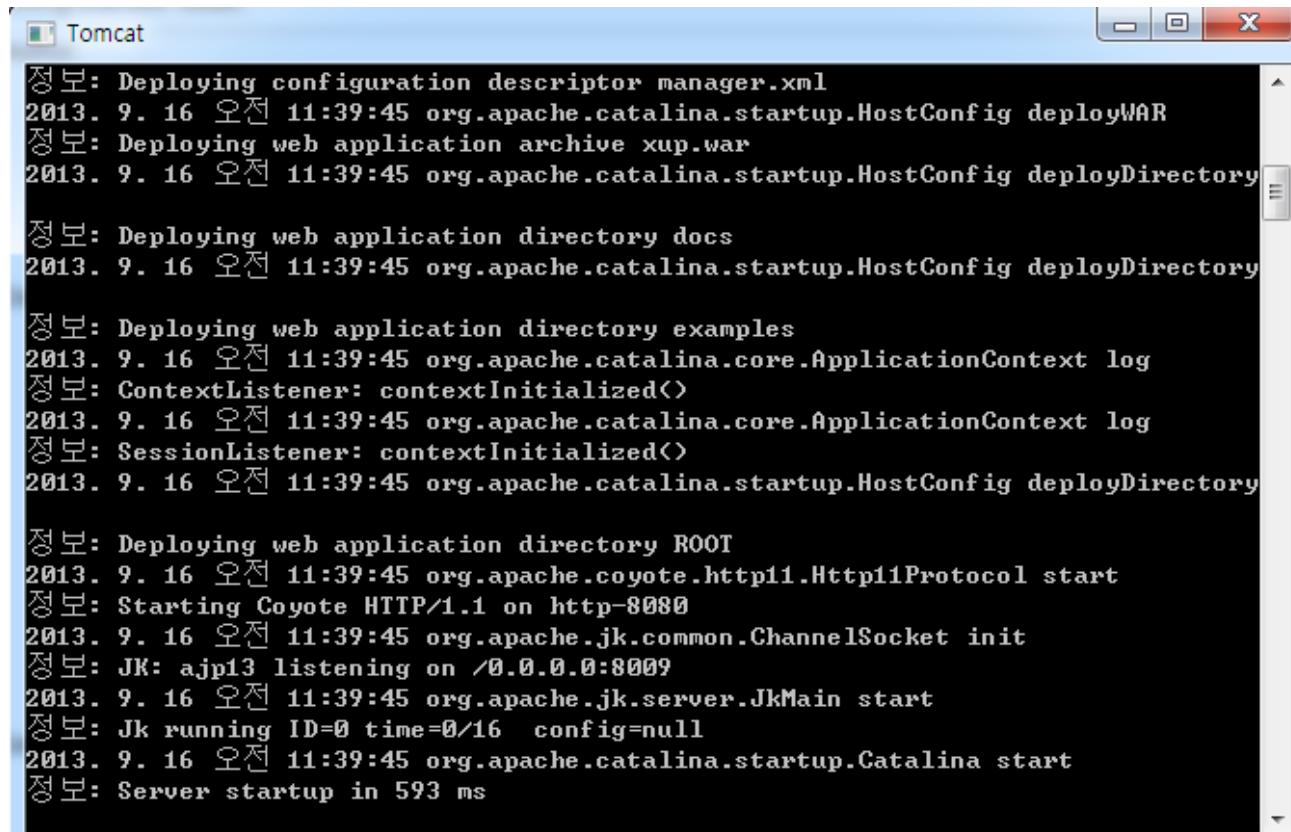
1. Mashup-Overview 화면에서 모델 Undeploy를 위한 아래 사항들을 입력합니다.

Field Name	Field Value
Author	'xup tester' 필수입력사항이 아니라 입력하지 않으셔도 무방합니다.
Description	'This is a X-UP Server' 배치해야 할 X-UP서버에 대한 설명을 입력합니다. 필수입력사항이 아니라 입력하지 않으셔도 무방합니다.
Servers	'http://localhost:8080/xup/FrontControllerServlet.do' [+]버튼을 입력하여 Undeloy할 모델이 위치한X-UP서버 URL을 입력합니다. X-UP 서버 URL은 http://[host name:port]/xup/[servlet mapping name] 으로 구성됩니다. 복수의 X-UP 서버를 설정하고 체크버튼으로 배치할 서버를 선택할 수 있습니다. X-UP 서버의 기본 Servlet Url Pattern 맵핑정보는 *.do 입니다.

X-UP 모델 Undeploy하기

위에서 설정한 서버에 배치된 모델이 있다는 가정하에 설명합니다.

1. WAS를 실행합니다. 본 매뉴얼에서는 톰캣 6.0을 기준으로 작성하였습니다.



```

정보: Deploying configuration descriptor manager.xml
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployWAR
정보: Deploying web application archive xup.war
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

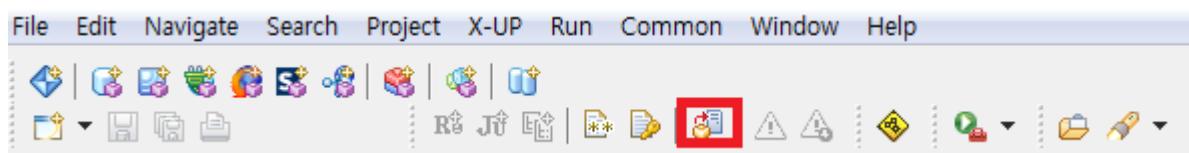
정보: Deploying web application directory docs
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

정보: Deploying web application directory examples
2013. 9. 16 오전 11:39:45 org.apache.catalina.core.ApplicationContext log
정보: ContextListener: contextInitialized()
2013. 9. 16 오전 11:39:45 org.apache.catalina.core.ApplicationContext log
정보: SessionListener: contextInitialized()
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

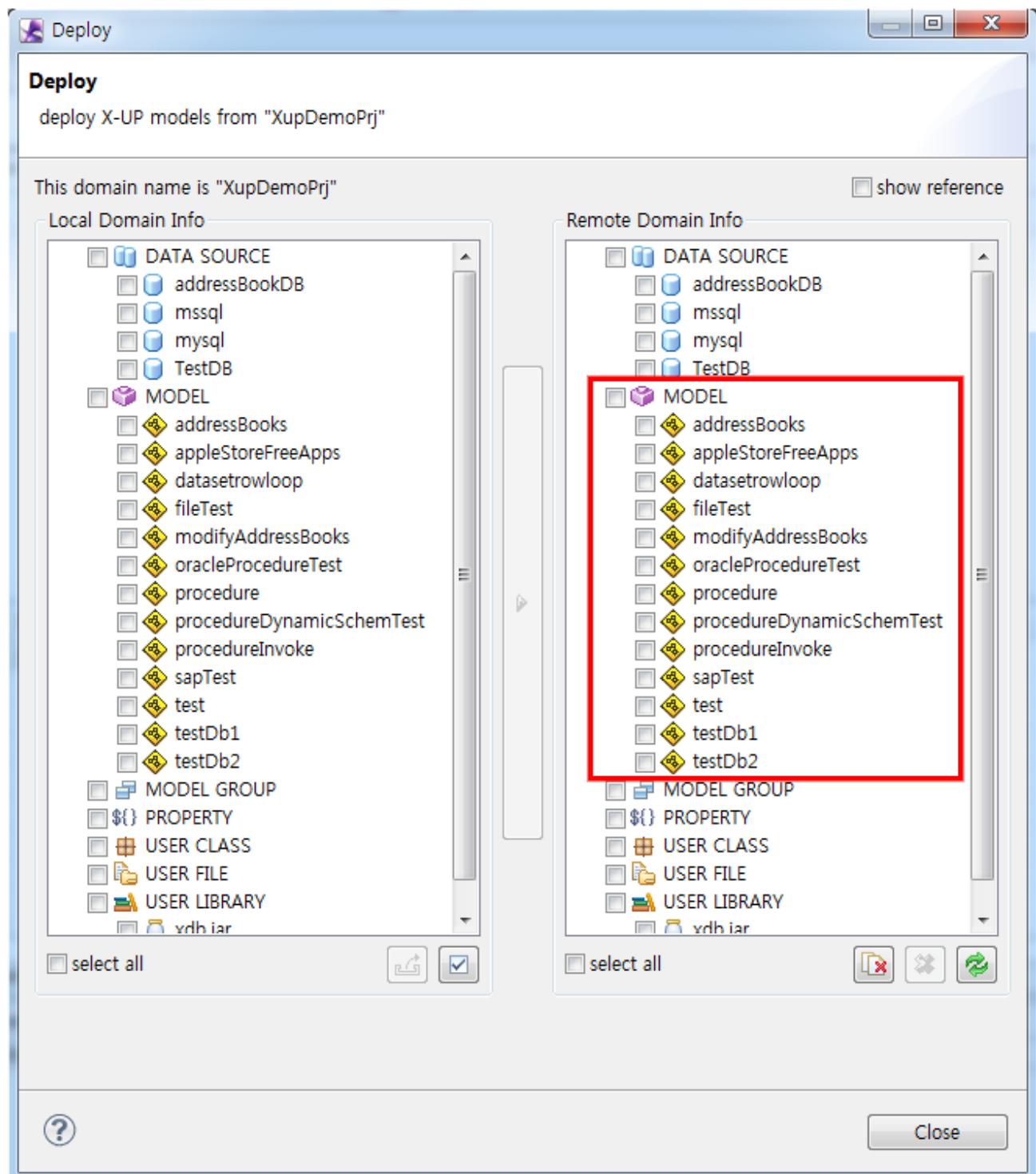
정보: Deploying web application directory ROOT
2013. 9. 16 오전 11:39:45 org.apache.coyote.http11.Http11Protocol start
정보: Starting Coyote HTTP/1.1 on http-8080
2013. 9. 16 오전 11:39:45 org.apache.jk.common.ChannelSocket init
정보: JK: ajp13 listening on /0.0.0.0:8009
2013. 9. 16 오전 11:39:45 org.apache.jk.server.JkMain start
정보: Jk running ID=0 time=0/16 config=null
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.Catalina start
정보: Server startup in 593 ms

```

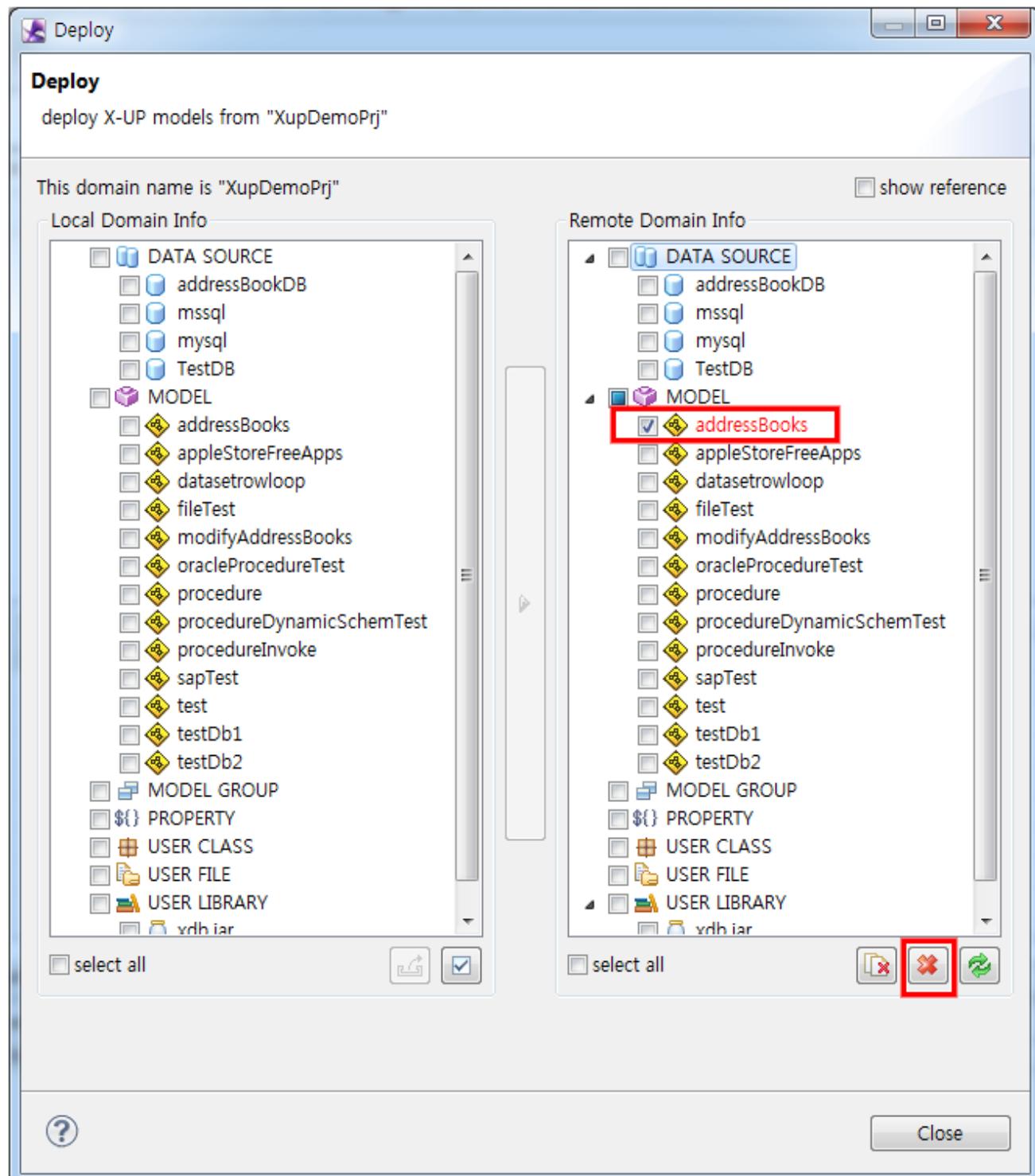
2. X-UP Explorer의 `mashup.msh` 파일을 선택한 후 툴바의 Deploy X-UP Model [] 아이콘을 클릭 합니다.



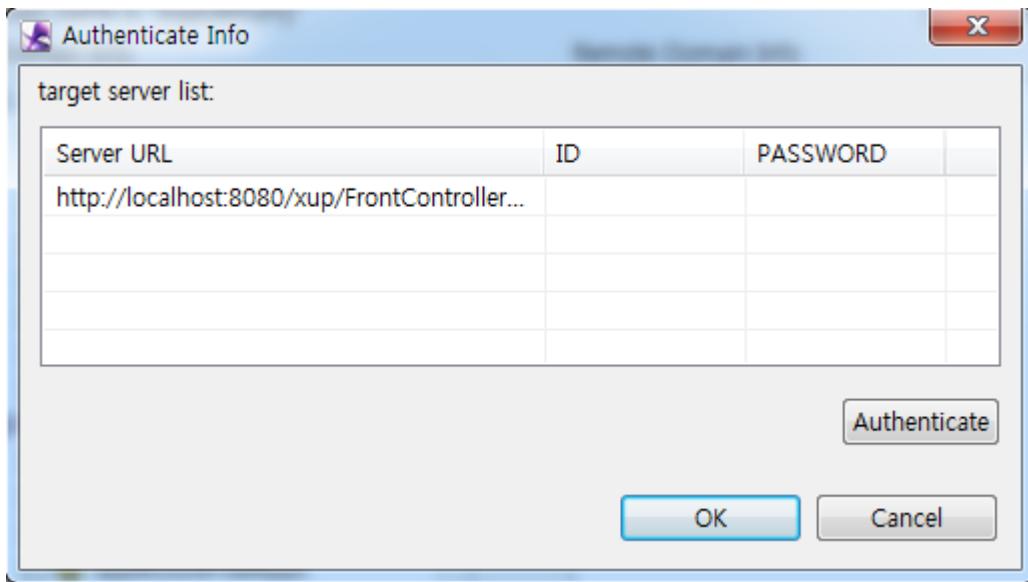
3. 아래 그림과 같이 Deploy 위치드가 활성화 되면 서버에 Deploy된 모델 리스트가 나타납니다.



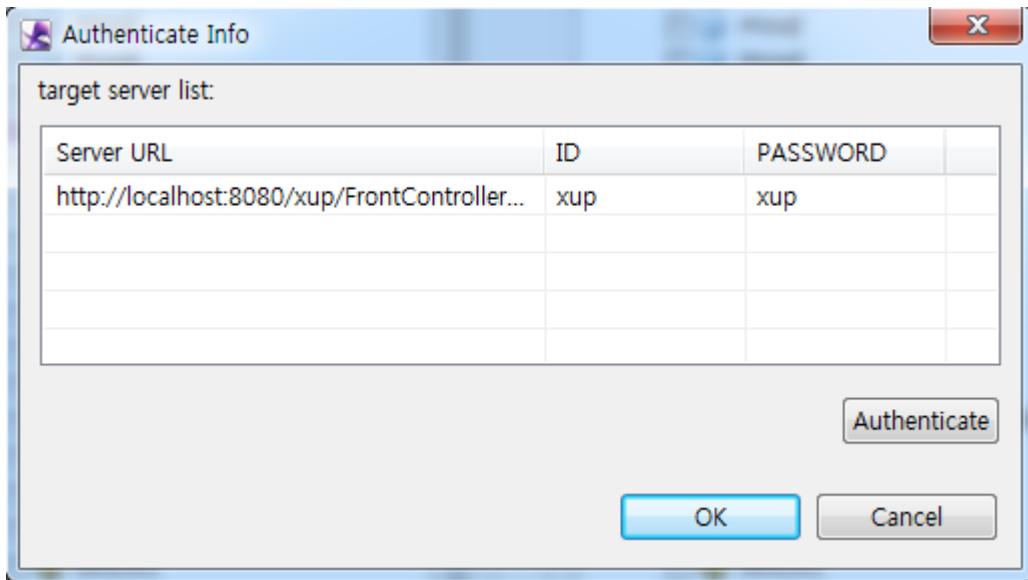
4. Remote Domain Info 리스트에서 Undeploy할 모델을 선택한 후 [✖] 아이콘을 클릭하면 서버에서 Undeploy를 수행 합니다. 'addressBooks' 모델을 선택합니다.
5. 아래 [✖] 를 선택합니다.



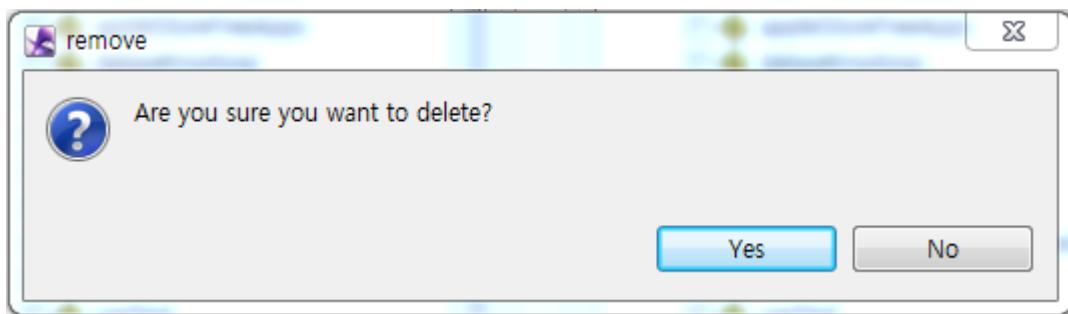
6. 배치를 수행 하기 전 아래의 그림과 같이 서버의 인증 화면이 나타납니다.



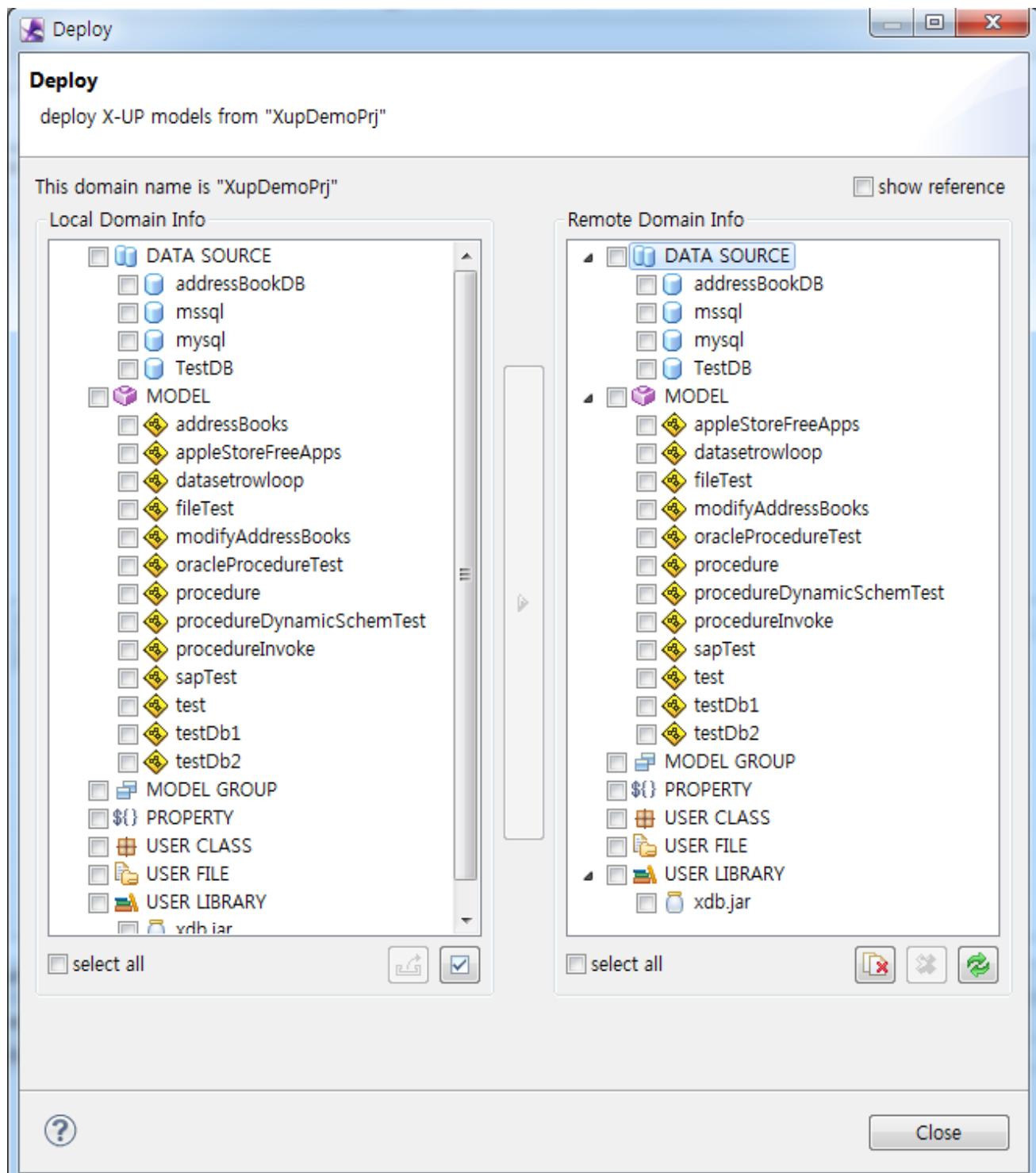
7. X-UP 설치 시 기본 계정 정보는 xup/xup 입니다. 그림과 같이 id/password를 입력 후 'OK' 버튼을 클릭하여 Undeploy를 완료 합니다.



8. remove를 수행할 지에 대한 확인창이 뜨면 'Yes'를 선택합니다.



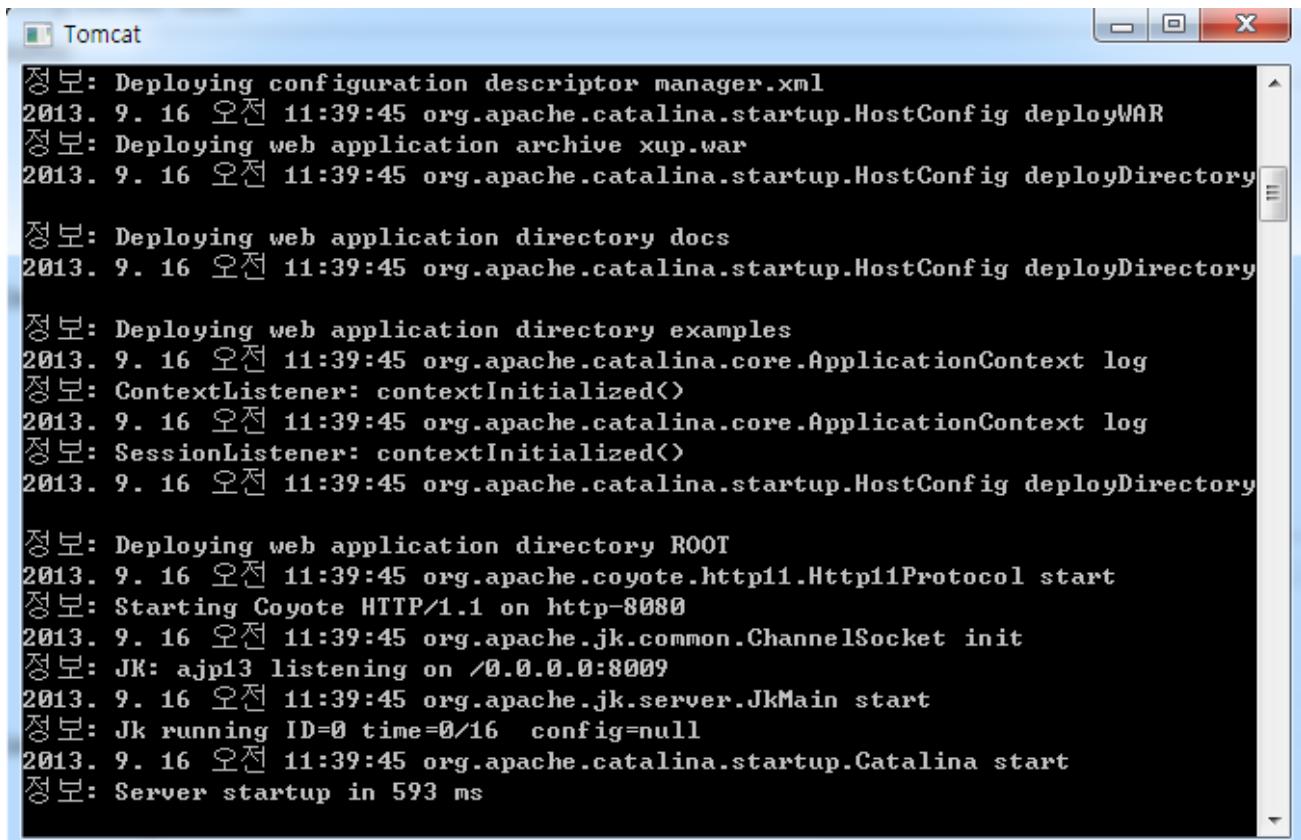
9. Remote Domain Info 리스트에 'addressBooks' 모델이 삭제된 것을 확인합니다.



도메인 안에 있는 전체 모델에 대해 Undeploy 하기

위에서 설정한 서버에 배치된 모델이 있다는 가정하에 설명합니다.

- WAS를 실행합니다. 본 매뉴얼에서는 톰캣 6.0을 기준으로 작성하였습니다.



```
정보: Deploying configuration descriptor manager.xml
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployWAR
정보: Deploying web application archive xup.war
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

정보: Deploying web application directory docs
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

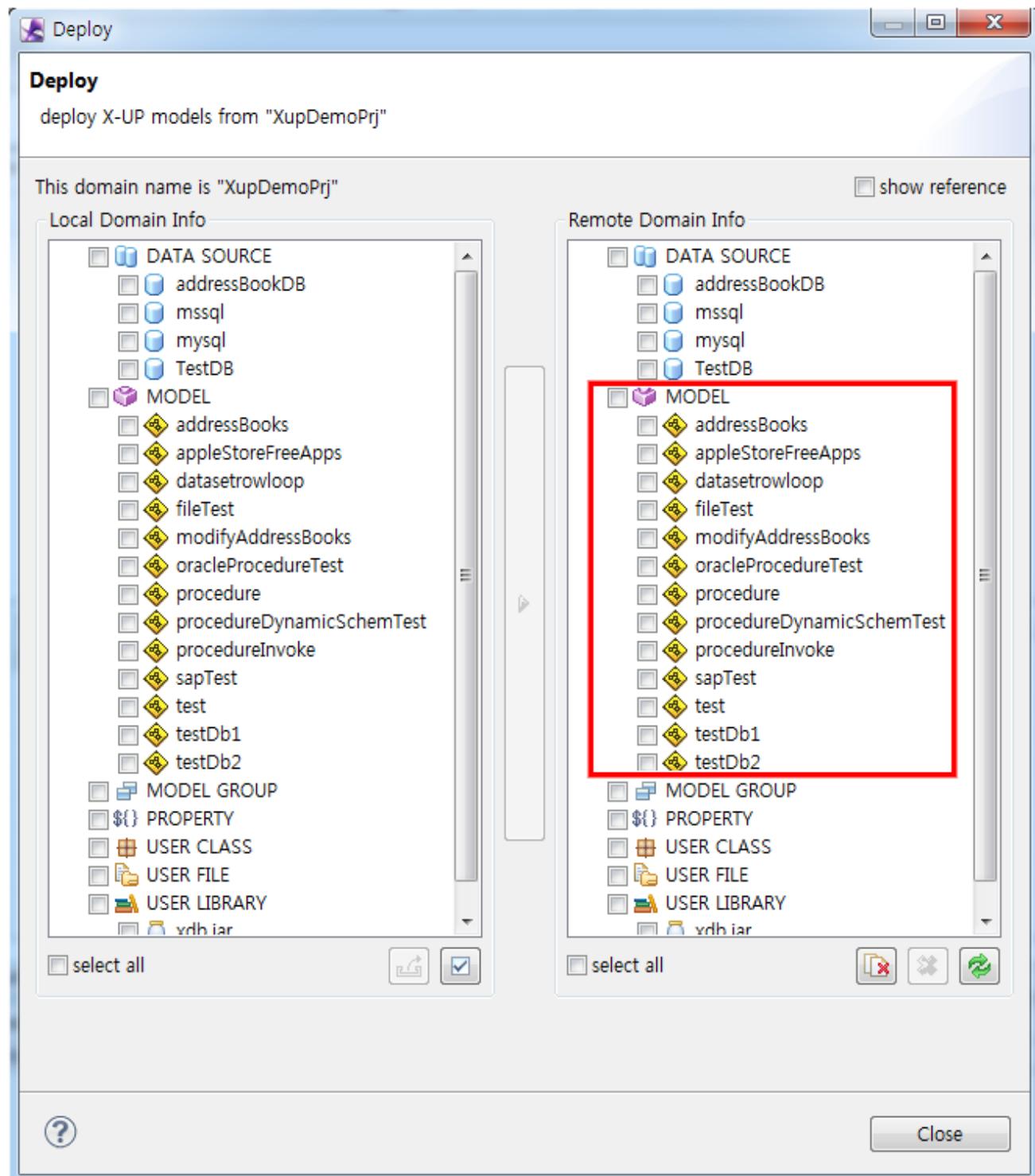
정보: Deploying web application directory examples
2013. 9. 16 오전 11:39:45 org.apache.catalina.core.ApplicationContext log
정보: ContextListener: contextInitialized()
2013. 9. 16 오전 11:39:45 org.apache.catalina.core.ApplicationContext log
정보: SessionListener: contextInitialized()
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.HostConfig deployDirectory

정보: Deploying web application directory ROOT
2013. 9. 16 오전 11:39:45 org.apache.coyote.http11.Http11Protocol start
정보: Starting Coyote HTTP/1.1 on http-8080
2013. 9. 16 오전 11:39:45 org.apache.jk.common.ChannelSocket init
정보: JK: ajp13 listening on /0.0.0.0:8009
2013. 9. 16 오전 11:39:45 org.apache.jk.server.JkMain start
정보: Jk running ID=0 time=0/16 config=null
2013. 9. 16 오전 11:39:45 org.apache.catalina.startup.Catalina start
정보: Server startup in 593 ms
```

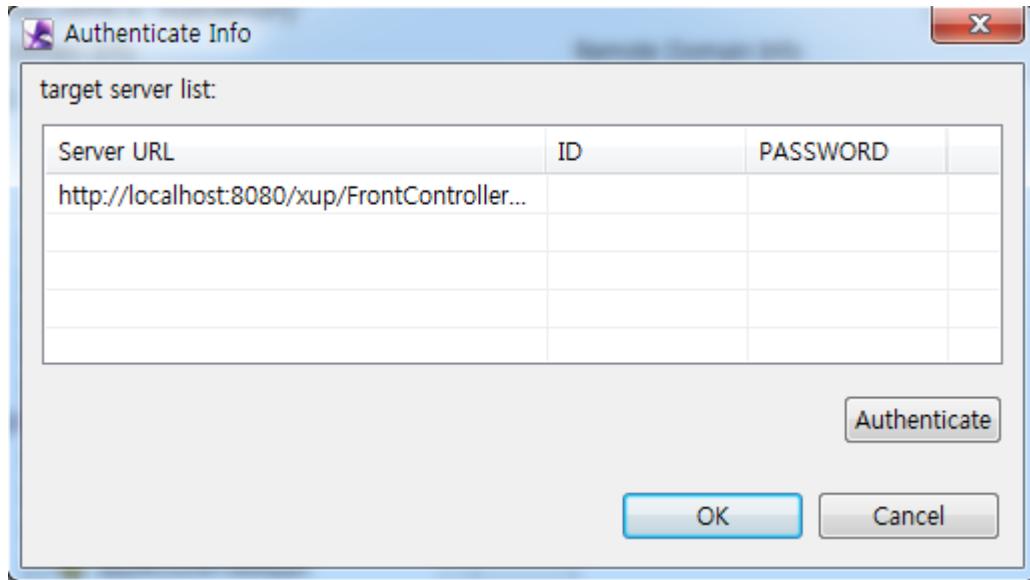
2. X-UP Explorer의 mashup.msh 파일을 선택한 후 툴바의 Deploy X-UP Model [] 아이콘을 클릭 합니다.



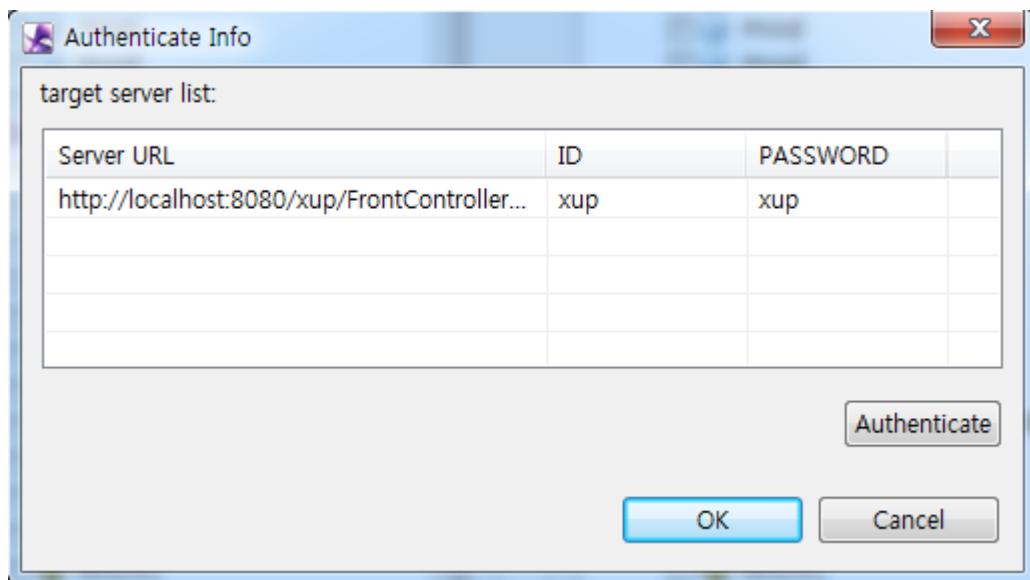
3. 아래 그림과 같이 Deploy 위치드가 활성화 되면 서버에 Deploy된 모델 리스트가 나타납니다.



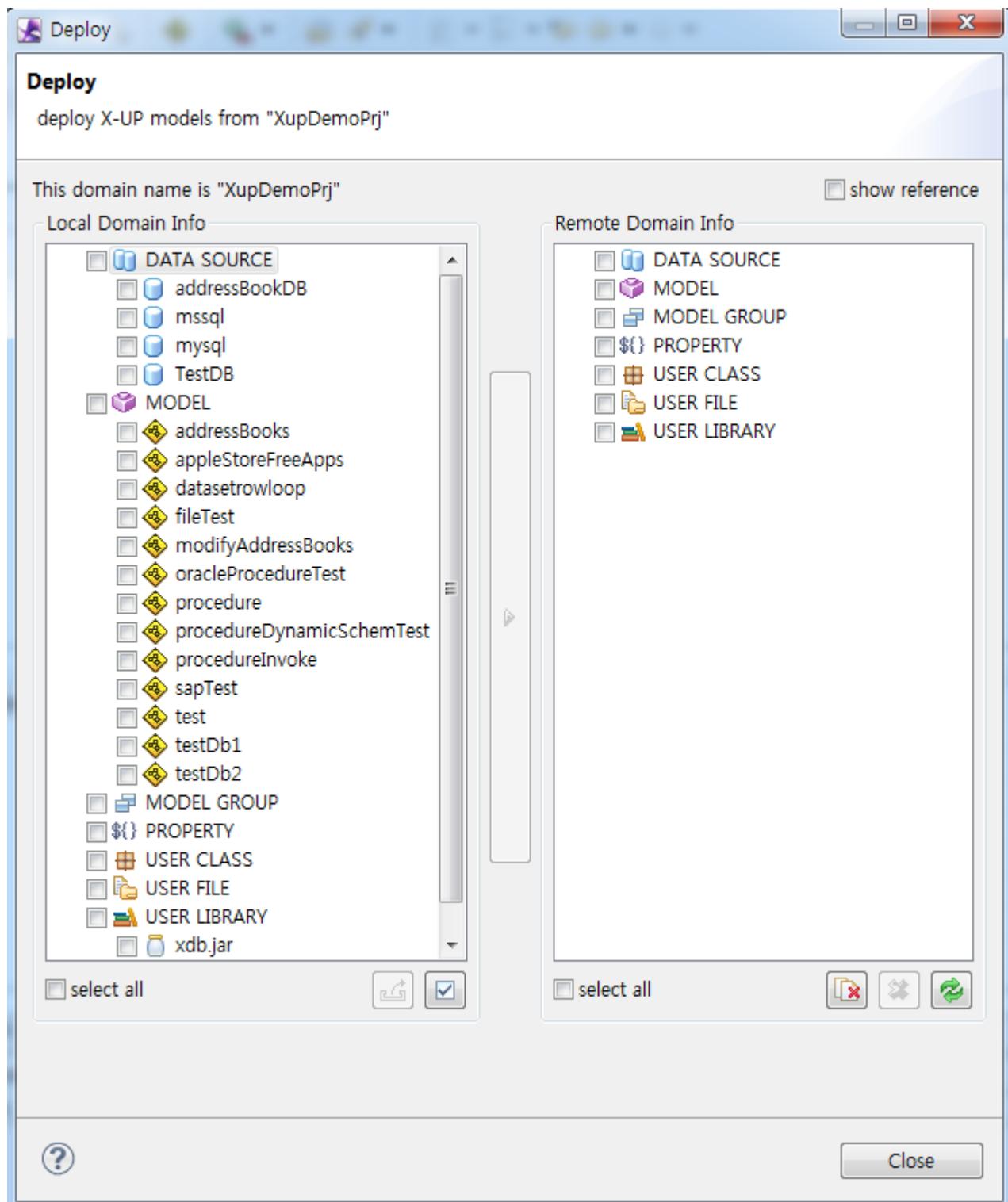
4. Remote Domain Info 리스트에서 전체 모델을 Undeploy 하기 위해 [] 아이콘을 클릭합니다. 서버에서 Undeploy를 수행 합니다.
5. 배치를 수행 하기 전 아래의 그림과 같이 서버의 인증 화면이 나타납니다.



6. X-UP 설치 시 기본 계정 정보는 xup/xup 입니다. 그림과 같이 id/password를 입력 후 'OK' 버튼을 클릭하여 Undeploy를 완료 합니다.



7. Remote Domain Info 리스트에 도메인의 전체 모델과, 모델의 DataSource 및 라이브러리들이 삭제된 것을 확인합니다.



11.

XPLATFORM 어플리케이션에서 X-UP 모델 사용하기

이 장에서는 [8.5 OpenApi Invoke를 이용한 모델 개발하기](#) 절에서 만든 모델을 사용하여 XPLATFORM 어플리케이션을 개발하는 방법에 대해 설명합니다. XPLATFORM 어플리케이션은 UX-Studio 툴을 이용하여 개발합니다. 모델 개발과 배치 방법은 해당 장을 참고하십시오.



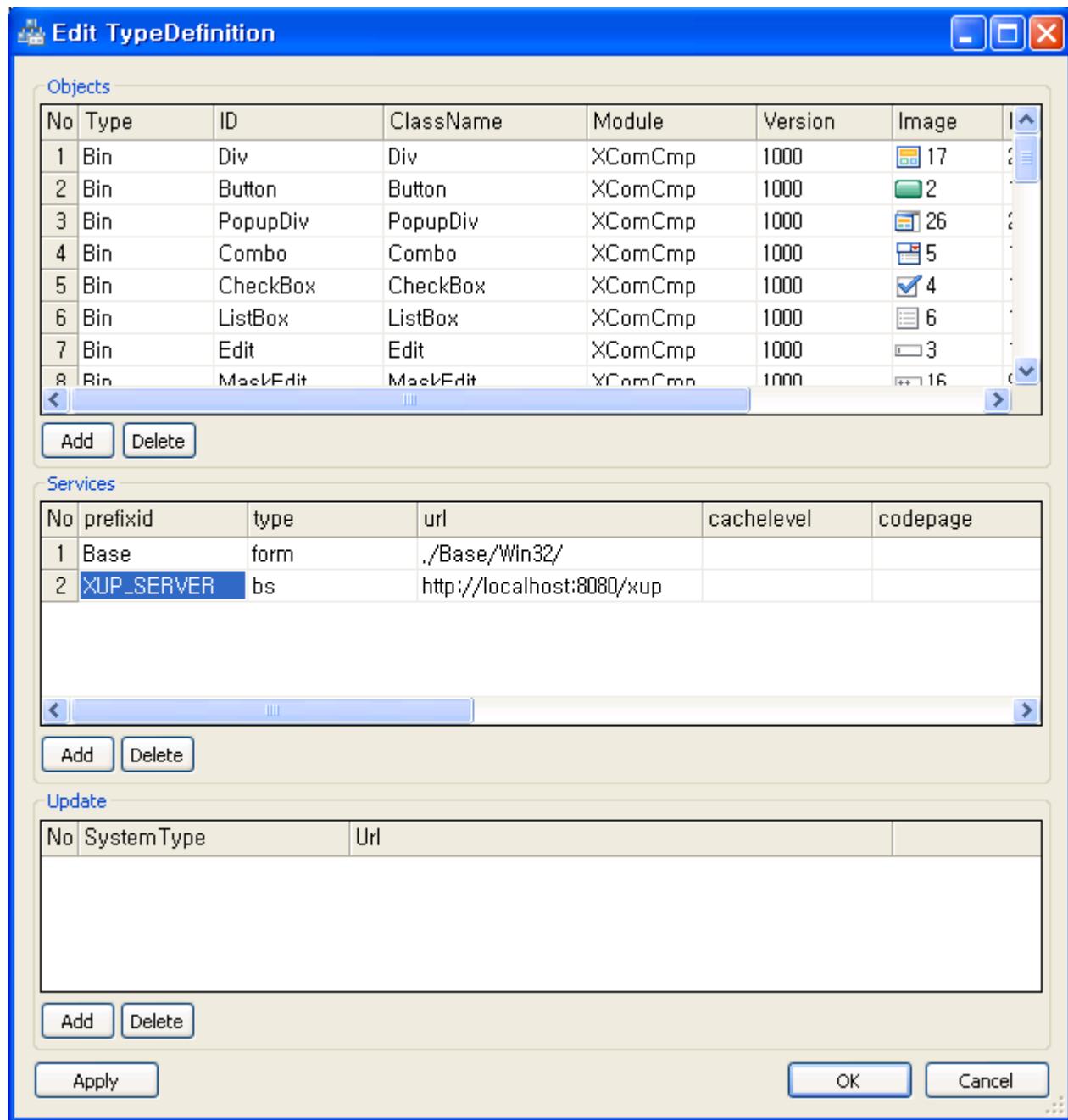
이번 장은 사용자가 XPLATFORM 및 UX-Studio에 익숙하다는 전제하에 작성 되었습니다. 따라서, 기본적인 UX-Studio를 이용한 XPLATFORM 어플리케이션 개발 단계에 대한 상세 정보는 설명하지 않습니다.

이 장에서 설명하는 UX-Studio를 이용한 X-UP 사용 어플리케이션 개발 단계는 다음과 같습니다.

- TypeDefinition-Service 등록하기
- 모델 리스트와 인터페이스 가져오기
- 완성된 모델 적용하기
- 모델 테스트하기

11.1 TypeDefinition-Server 등록하기

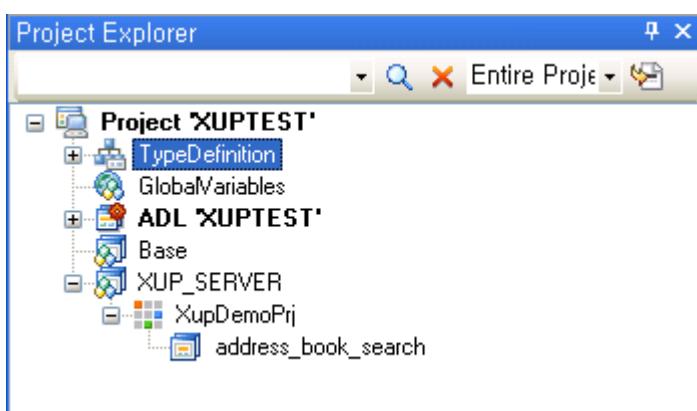
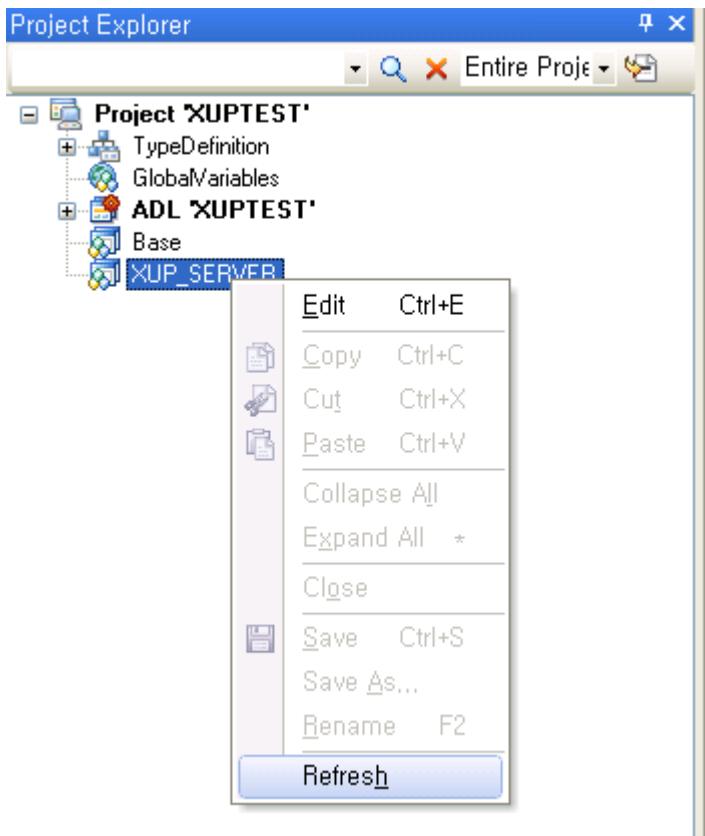
1. 아래의 그림과 같이 TypeDefinition에서 Service를 추가 합니다.



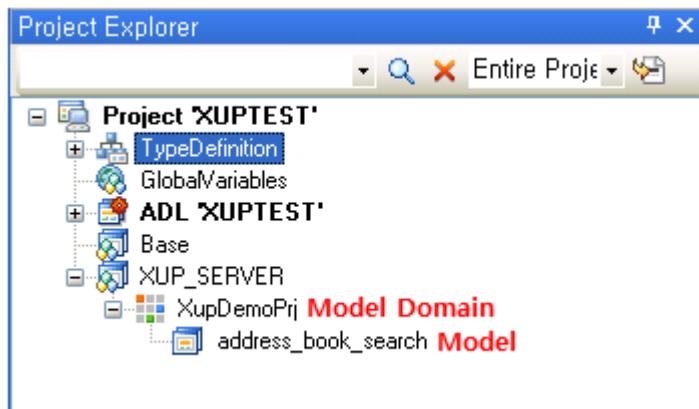
Field Name	Field Value
Prefix	'XUP_SERVER' 모델 호출 스크립트 코드에서 사용되는 Prefix를 입력합니다.
Type	'bs' X-UP 모델을 호출하기 위해서는 반드시 bs 타입을 선택해야 합니다.
URL	'http://localhost:8080/xup/FrontControllerServlet.do' X-UP 서버 URL을 입력합니다.
ServiceList	'?service=serviceInfo&target=modelList' 서비스 리스트 호출 url을 입력합니다.
데이터셋Layout	'?service=serviceInfo&target=modelLayout' 입출력 정보를 획득하기 위한 url을 입력합니다.

11.2 모델 리스트와 인터페이스 가져오기

1. Project Explorer에서 해당 Service를 Refresh하여 X-UP서버의 모델 리스트를 가져옵니다.

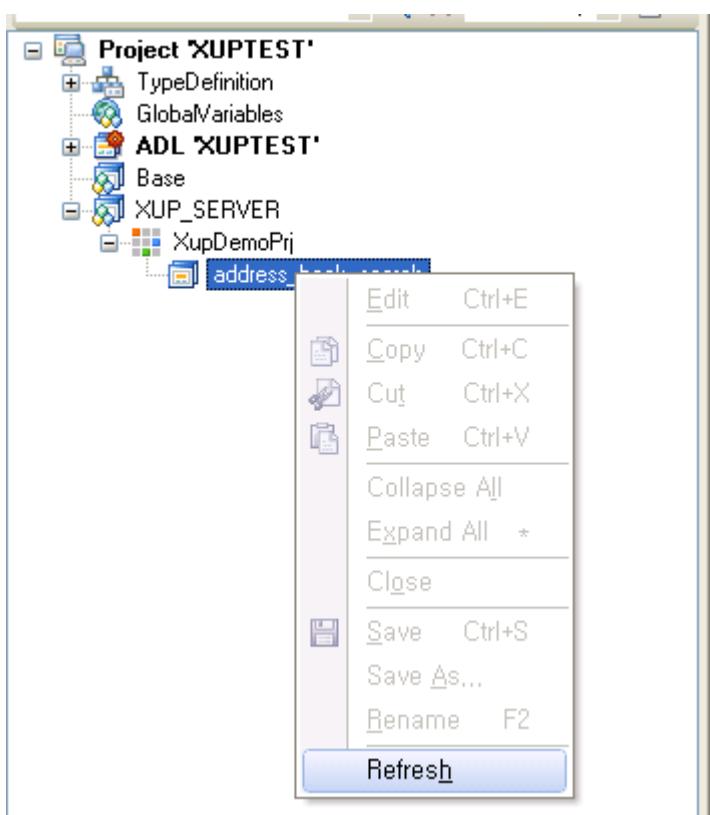


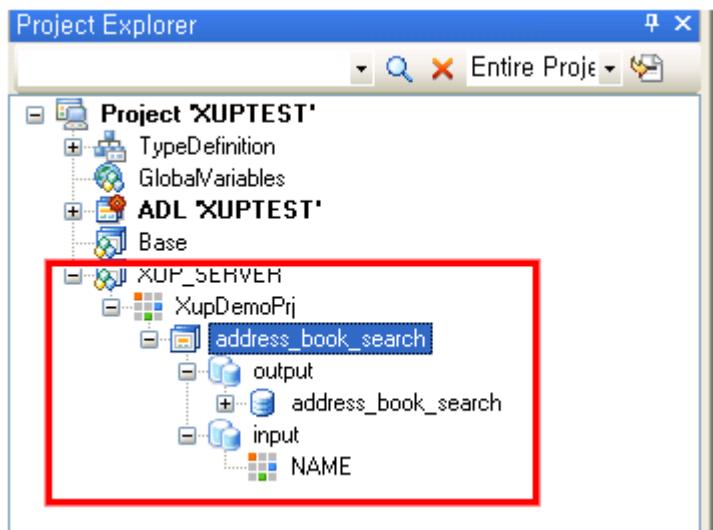
가져온 모델 리스트는 모델 도메인별로 모델들이 표시됩니다.



모델 도메인이란 모델을 구분하기 위한 네임스페이스로 X-UP Builder의 프로젝트 이름으로 설정됩니다.
즉, X-UP 프로젝트 하나가 모델 도메인 하나에 해당됩니다.

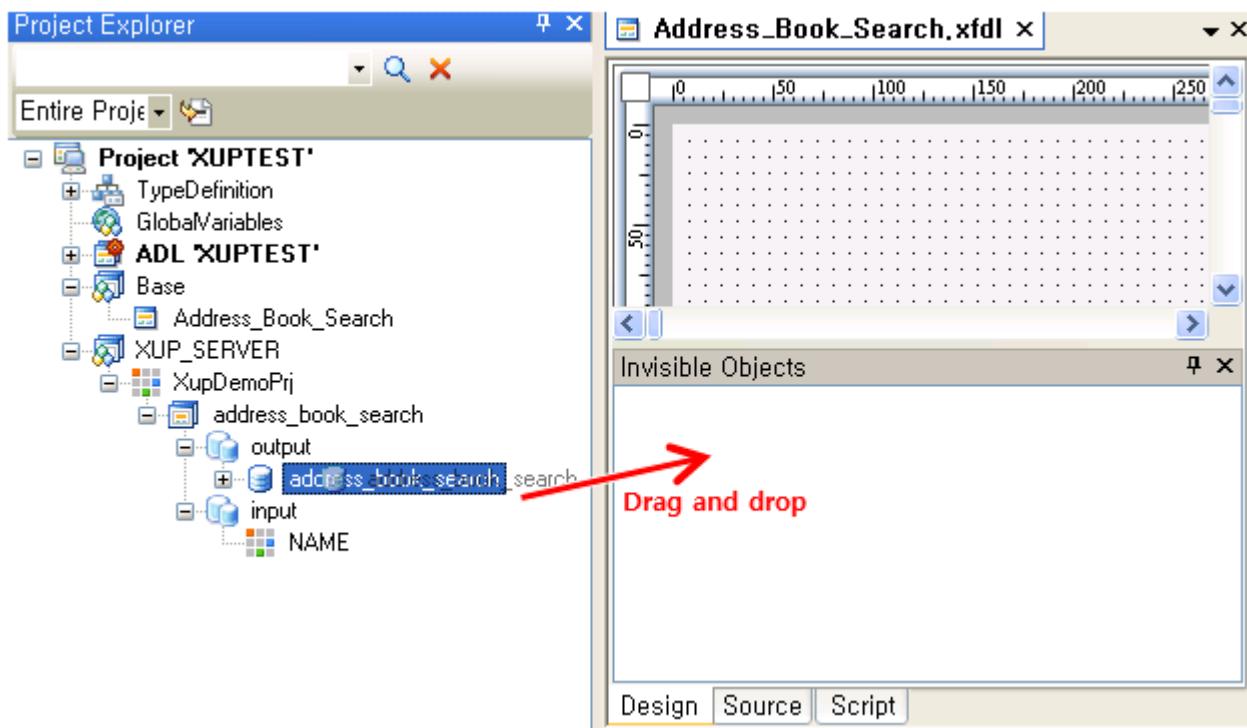
2. 모델 리스트에서 “naverBookSearch” 모델을 Refresh하여 인터페이스 정보(입출력 정보)를 가져옵니다.





11.3 모델 호출 스크립트 작성하기

1. Project Explorer의 모델의 인터페이스 정보에서 출력 데이터셋을 Invisible Objects 에디터로 드래그&드롭 합니다.



2. Script 에디터에서 아래와 같이 모델 호출 코드를 작성합니다.

```

function btnSearchBlog_OnClick(obj:Button, e:ClickEventArgs)
{
    // Model Input parameters
    var param = " NAME=" + edSearch.Text;

    var svcparam = "domain=" + "XUPDemoPrj"      // Model Domain name
        +"&model=" + "address_book_search" // Model name
        +"&format=" + "xml"
        +"&version=" + "xplatform";

    var svcurl = "XUP_SERVER:::service=xupservice&" + svcparam;
    Transaction("address_book_search", svcurl, "",
        "address_book_search=address_book_search", param, "CallbackFunc");
}

function CallbackFunc(strSvcID, nErrorCode, strErrorMag)
{
    if (nErrorCode != 0)
    {
        alert(nErrorCode + " : " + strErrorMag);
        return;
    }

    if(strSvcID=="address_book_search")
    {
        trace(address_book_search.SaveXML());
    }
}

```



XPLATFORM에서 X-UP 모델 호출하기 위한 정보는 아래와 같습니다.

서비스 URL:

`http://[host]:[port]/xup/FrontControllerServlet.do?service=xupservice&domain=[도메인 이름]
&model=[모델이름]&format=xml&version=miplatform`

입력 파라미터 : 각 모델에서 요구하는 입력 파라미터를 GET 또는 POST방식으로 전달

3. Quick View를 실행하여 결과를 확인합니다.

12.

WebApplication(eGovFrame)에서 X-UP 사용하기

이 장에서는 [8.1 SAP RFC Invoke를 이용한 모델 개발하기](#) 절에서 만든 모델을 사용하여 WebApplication에서 X-UP에 존재하는 모델을 사용하는 방법에 대해 설명합니다.



이번 장은 eGovFrame 2.6.0 을 기준으로 작성되었으며, 사용자가 WebApplication 개발에 익숙하다는 전제하에 작성 되었습니다.

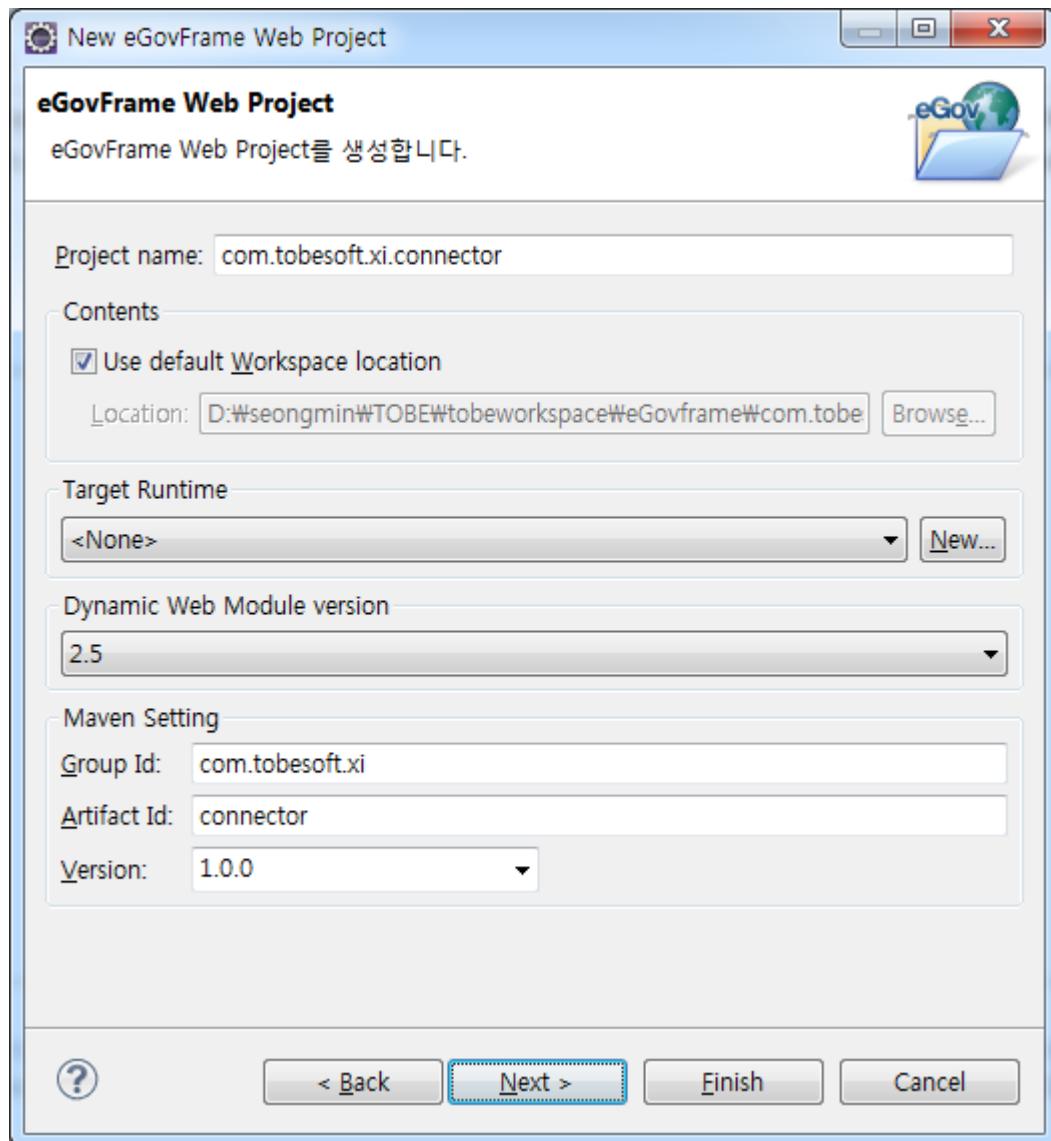
이 장에서 설명하는 UX-Studio를 이용한 X-UP 사용 어플리케이션 개발단계는 다음과 같습니다.

- WebApplication 생성
- WebApplication 에 X-UP 적용
- SAP RFC Invoke 를 이용한 모델 개발
- WebApplication 서버에 개발 된 모델을 Export하기
- WebApplication 에서 모델 호출 하기
- 테스트 하기

12.1 WebApplication 생성

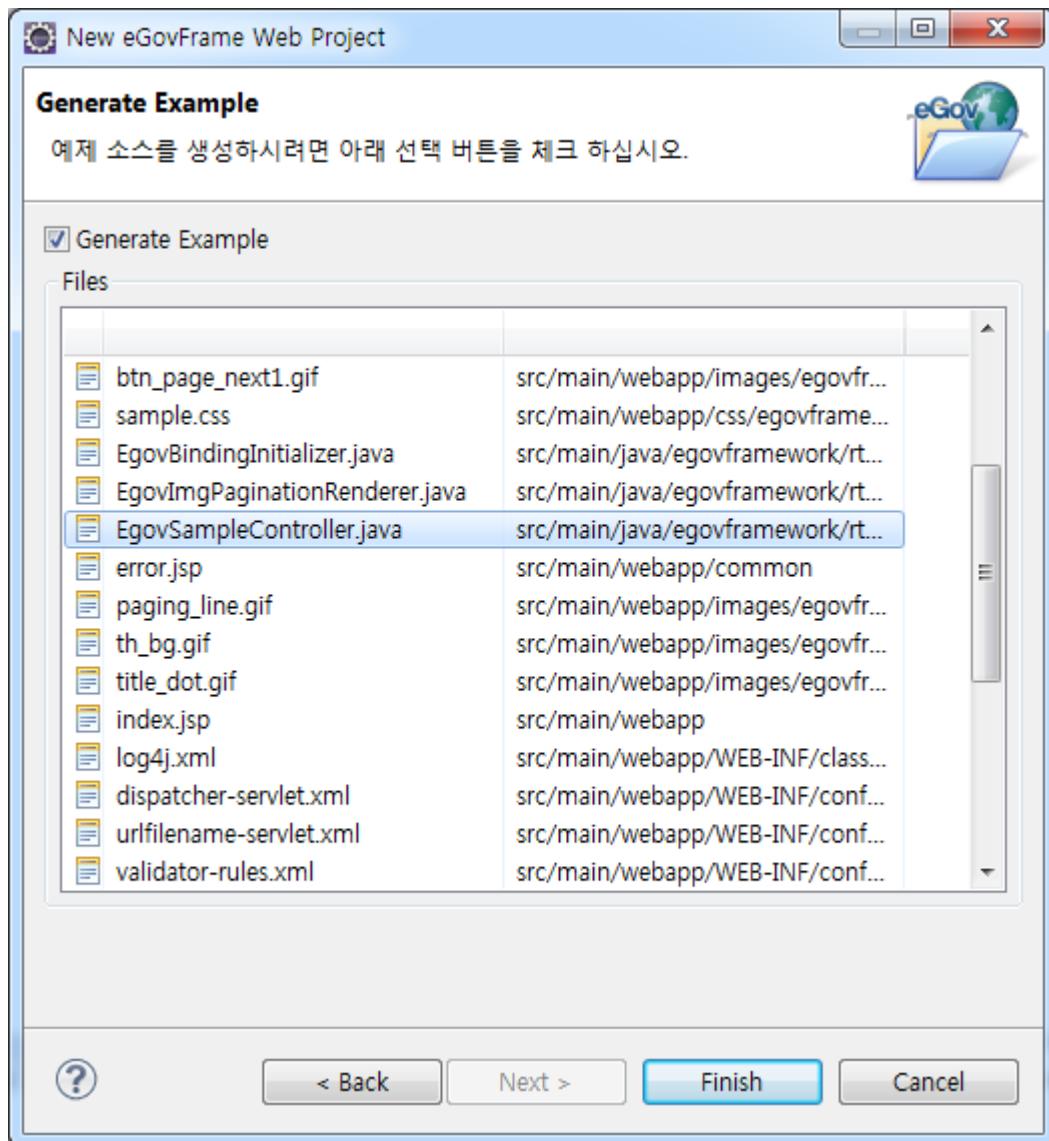
WebApplication 에 X-UP 을 설정하기 위해선 하나의 WebApplication 이 필요합니다.

프로젝트를 생성하기 위하여 File > new eGovFrame Web Project 를 선택합니다.

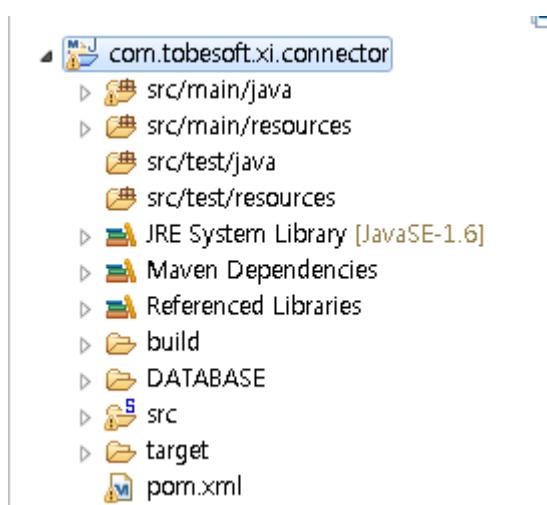


Project 명을 명시하고 next 버튼을 클릭합니다.

Generate Example (EgovSampleController.java) 를 통해 예제를 생성합니다.



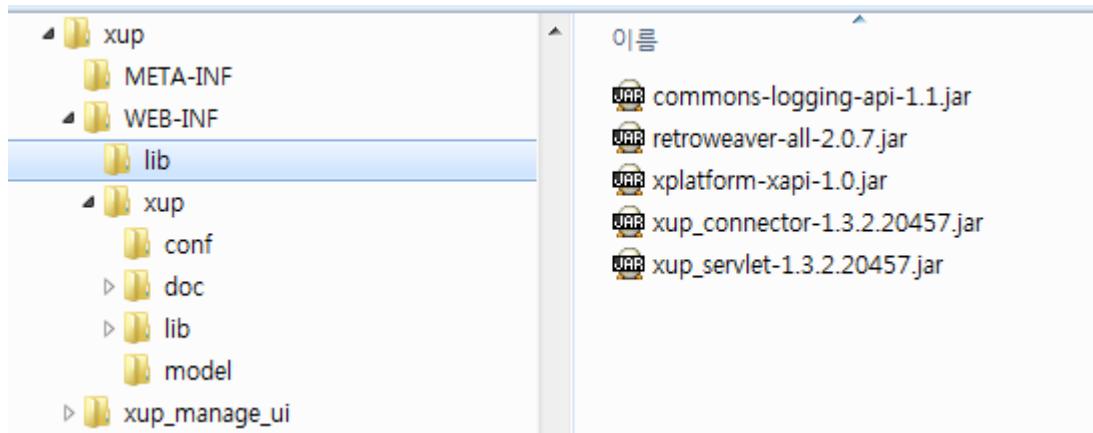
생성 된 프로젝트의 모습입니다.



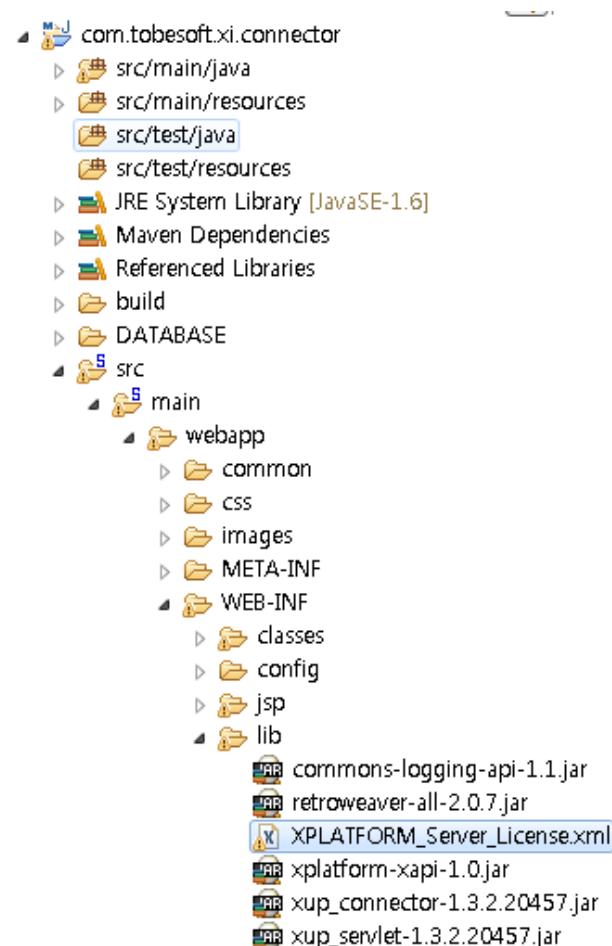
12.2 WebApplication 에 X-UP 적용.

WebApplication 에 X-UP 을 적용하기 위해선 다음과 순서로 진행됩니다.

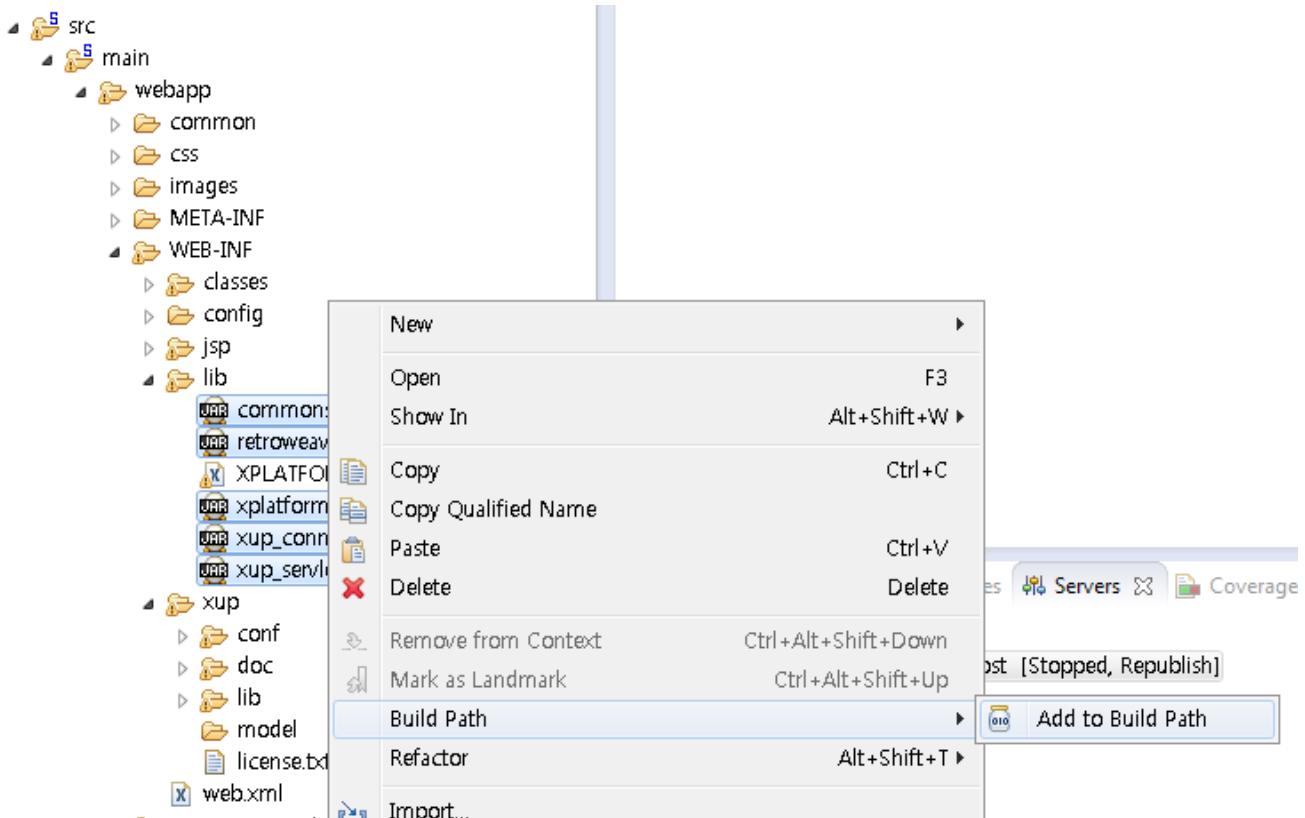
xup.war 의 내용을 압축을 풀어 xup/WEB-INF/lib 폴더 하단의 library 를 생성 된 프로젝트의 src/main/webapp/WEB-INF/lib 하단으로 copy 합니다.



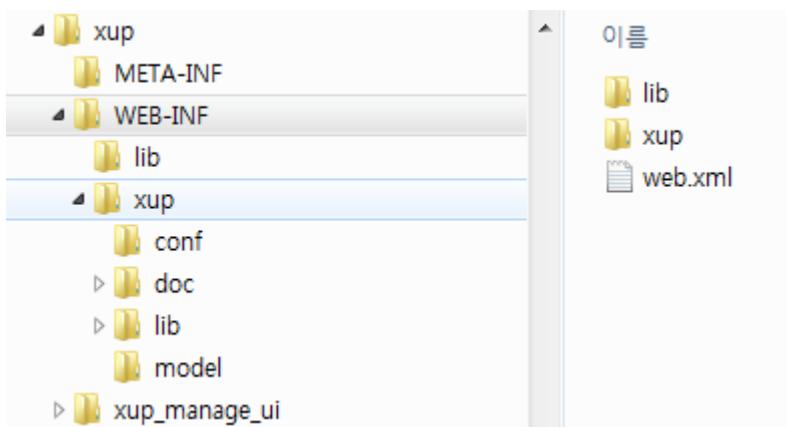
X-API 를 사용하기 위해 발급받은 XPLATFORM_Server_License.xml 파일을 copy 합니다.



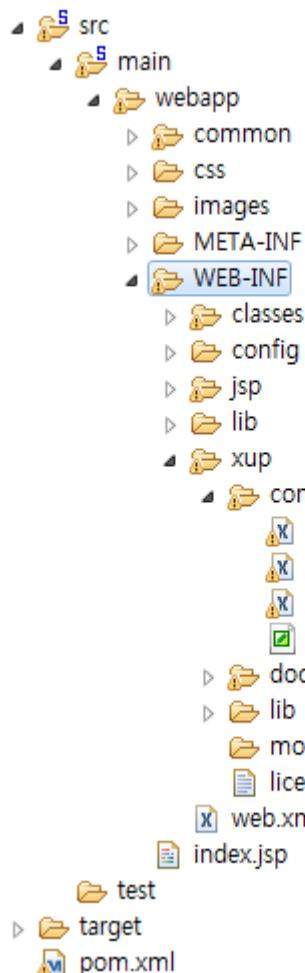
WebApplication에서 사용하기 위해 해당 library 파일들을 Build Path에 추가합니다.



xup.war의 압축이 풀어진 xup/WEB-INF 하단의 xup 폴더를 src/main/webapp/WEB-INF 하단으로 copy 합니다.



Copy 된 후 해당 프로젝트에 생성된 디렉토리 구조 및 파일은 다음과 같습니다.



X-UP 을 사용하기 위해 발급받은 X-UP_Server_License.xml 파일을 src/main/webapp/WEB-INF/xup/conf/ 디렉토리 하단으로 copy 합니다.

추가적으로 X-UP 의 모든 기능을 사용하기 위해 X-UP 의 Servlet 을 web.xml 에 등록합니다.

Servlet 의 명칭은 FrontControllerServlet 이며, 해당 Servlet 의 url-pattern 은 eGovFrame 에서 제공하는 Dispatcher-Servlet 의 url-pattern 과 명칭이 동일하기 때문에 '.xup' 으로 변경합니다.

xup.war 파일에 존재하는 web.xml 안의 servlet 정보를 전자정부의 web.xml 에 추가해 줍니다.

xup/WEB-INF/web.xml > src/main/webapp/WEB-INF/web.xml

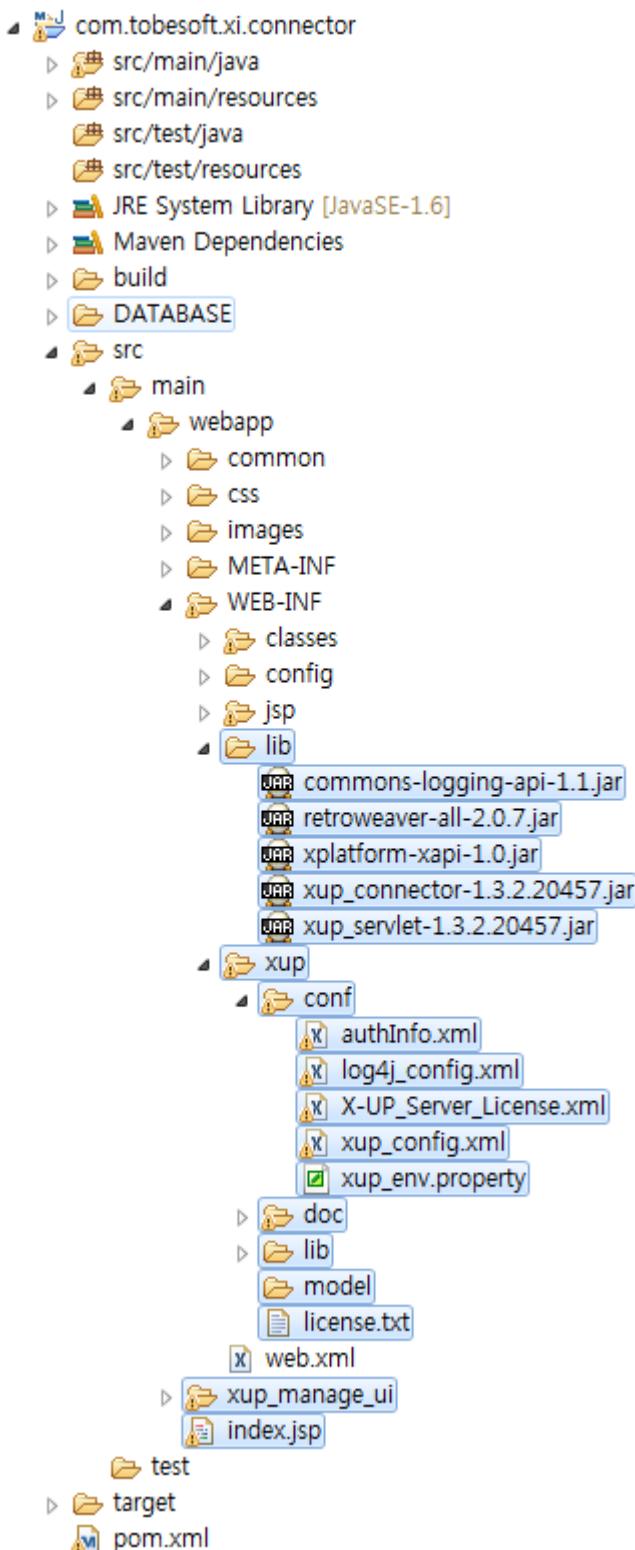
```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/config/egovframework/springmvc/dispatcher-servlet.xml,/WEB-INF/config/egovframework/springmvc/urlfilename-servlet.xml
    </param-value>
```

```
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>FrontControllerServlet</servlet-name>
    <servlet-class>com.tobesoft.xup.service.FrontControllerServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>FrontControllerServlet</servlet-name>
    <url-pattern>/FrontControllerServlet.xup</url-pattern>
</servlet-mapping>
```

또한 X-UP 의 관리자 기능을 사용한다면 xup.war 에 존재하는 xup_manage_ui 폴더 및 index.jsp를 copy 합니다.

설정된 최종 화면은 다음과 같습니다.



12.3 SAP RFC Invoke 를 이용한 모델 개발 하기

SAP RFC Invoke 를 개발 하기 위해선 다음 [8.1 SAP RFC Invoke를 이용한 모델 개발하기](#)을 참조합니다.

12.4 WebApplication 서버에 개발 된 모델을 Export 하기

WebApplication 서버에 개발 된 모델을 Export 하기 위한 방법을 명시합니다.

WAS 를 구동 한 후 X-UP Builder 의 mashup.msh 파일을 오픈합니다.

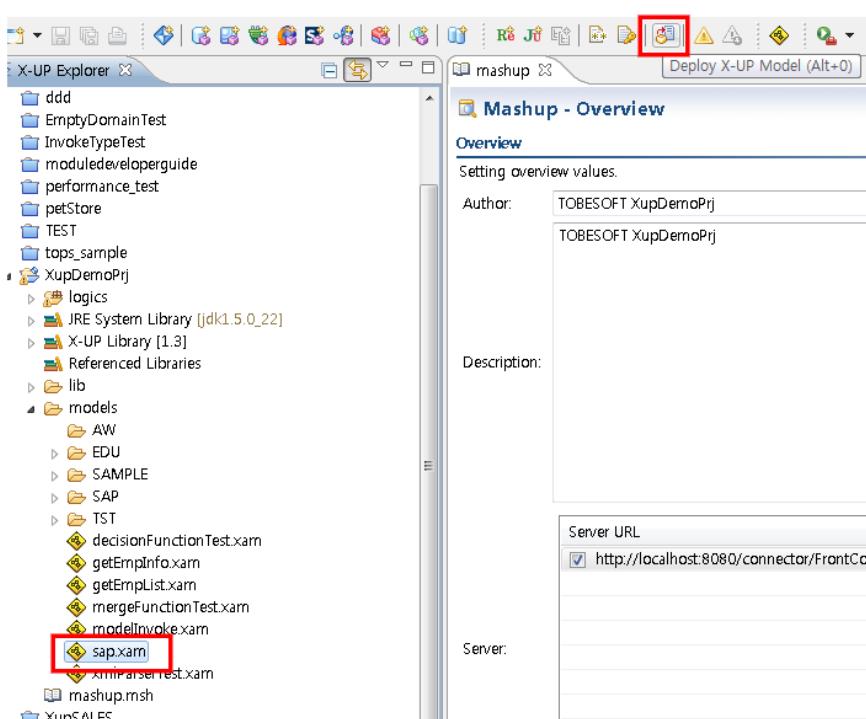
오픈 된 mashup.msh 파일의 Overview 의 기본 Server Url 은 다음과 같이 정의 되어 있습니다.

http://localhost:8080/xup/FrontControllerServlet.do

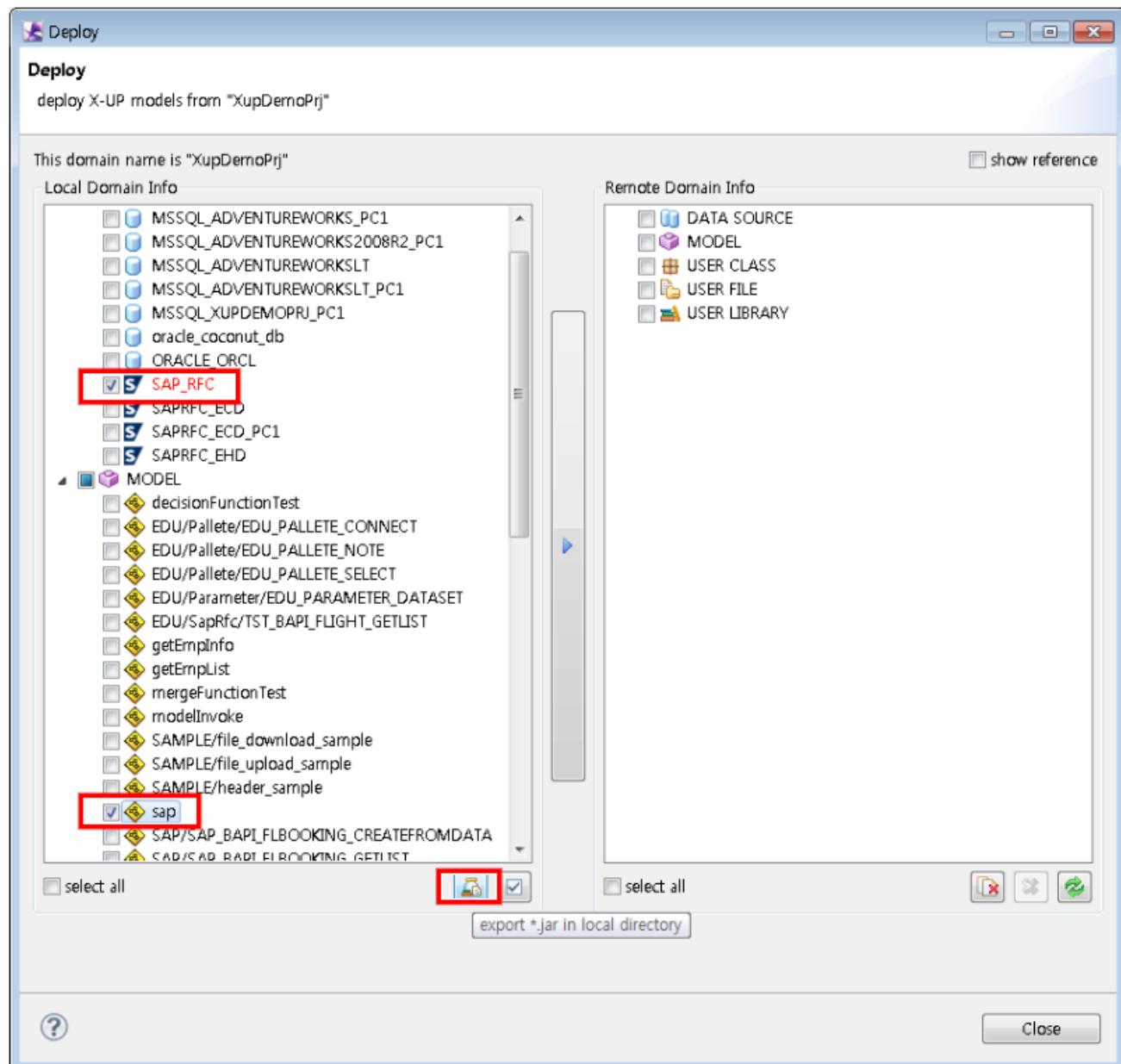
구동 된 WAS 의 WebModule 의 명칭으로 ServerUrl 을 변경합니다. (url-pattern 도 변경)

http://localhost:8080/connector/FrontControllerServlet.xup

모델을 선택 한 뒤 Deploy X-UP Model 을 클릭합니다.



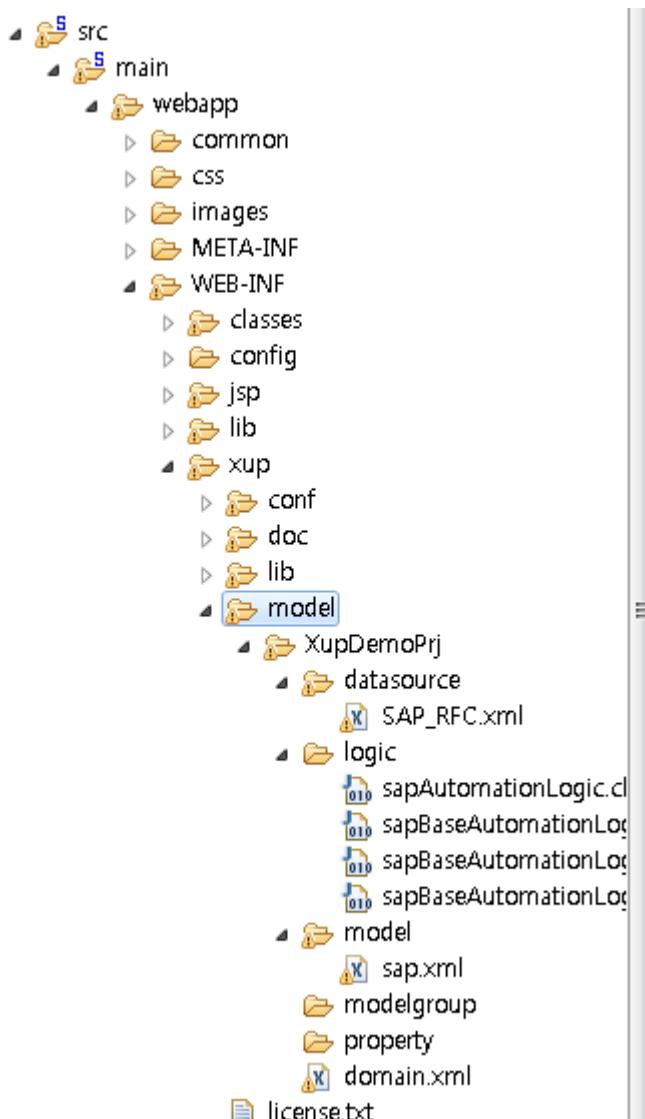
활성화된 Deploy 위자드에서 **export *.jar in local directory** 버튼을 클릭하여 jar 파일을 생성합니다.



생성 된 jar 파일을 압축을 해제하고 WebApplication 에 위치합니다.

해당 디렉토리는 다음과 같습니다.

src/main/webapp/WEB-INF/xup/model/ 폴더 하단에 위치합니다.



12.5 WebApplication에서 모델 호출하기.

WebApplication에서 모델을 호출하기 위해서 다음과 같은 Controller를 생성합니다.

아래의 코드를 통해 개발된 X-UP 모델을 호출 할 수 있습니다.

```
packageegovframework.rte.sample.web;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.context.annotation.Scope;
```

```

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import com.tobesoft.xplatform.data.PlatformData;
import com.tobesoft.xplatform.tx.HttpPlatformRequest;
import com.tobesoft.xplatform.tx.HttpPlatformResponse;
import com.tobesoft.xup.connector.XIConnector;
import com.tobesoft.xup.service.xi.XIServiceRequest;
import com.tobesoft.xup.service.xi.XIServiceResponse;

@Controller
public class XIServiceController {

    @RequestMapping(value="/xi/getStockList.do")
    public String getStockList(HttpServletRequest request, HttpServletResponse response) throws
Exception {
        // HttpServletRequest를 이용하여 HttpPlatformRequest 생성
        HttpPlatformRequest req = new HttpPlatformRequest(request);
        // XML 데이터 분석
        req.receiveData();
        // 데이터를 PlatformData 형태로 저장
        PlatformData platformData = req.getData();

        // XIConnector 생성
        XIConnector connector = XIConnector.getInstance();
        // X-UP 으로 전송하기 위한 Request 생성
        XIServiceRequest serviceRequest = connector.createRequest("xupDemoPrj", "sap",
platformData);

        XIServiceResponse serviceResponse =
(XIServiceResponse)connector.execute(serviceRequest);
        PlatformData resultPlatformData = serviceResponse.getPlatformData();

        //HttpServletResponse를 이용하여 HttpPlatformResponse 생성
        HttpPlatformResponse res = new HttpPlatformResponse(response);
        res.setData(resultPlatformData);
        //데이터 송신
        res.sendData();
        return null;
    }
}

```

```
}
```

12.6 테스트 하기

개발 된 Controller 를 호출 하여 X-UP 모델을 정상적으로 호출 하는지에 대해 확인합니다.

호출하기위해 getStockList 메서드의 RequestMapping 의 value 값으로 호출합니다.

브라우저에서 확인하기 위한 url 은 다음과 같습니다.

http://localhost:8080/connector/xi/getStockList.do

대상 WAS 서버 / ApplicationContext 이름 / RequestMapping

결과 데이터를 확인합니다. (sample)



The screenshot shows a browser window with the URL `localhost:8080/connector/xi/getStockList.do`. The page content is an XML document. At the top, it says "This XML file does not appear to have any style information associated with it." Below this, the XML structure is displayed:

```

<Root xmlns="http://www.tobesoft.com/platform/dataset" ver="5000">
  <Parameters>
    <Parameter id="009_MESSAGE" type="string"/>
    <Parameter id="009_STAT_CD" type="string"/>
  </Parameters>
  <Dataset id="E_FAIR0111">
    <ColumnInfo>
      <Column id="#WERKS" type="string" size="4"/>
      <Column id="#WERKS_NAME" type="string" size="30"/>
    </ColumnInfo>
    <Rows>...</Rows>
  </Dataset>
  <Dataset id="E_FAIR0121">
    <ColumnInfo>
      <Column id="#WERKS" type="string" size="4"/>
      <Column id="MATNR" type="string" size="18"/>
      <Column id="MAKTX" type="string" size="40"/>
      <Column id="LBKUM" type="bigdecimal" size="14"/>
      <Column id="MEINS" type="string" size="3"/>
    </ColumnInfo>
    <Rows>
      <Row>
        <Col id="#WERKS">1000</Col>
        <Col id="MATNR">F226</Col>
        <Col id="MAKTX">FIN226, MTO, PD, Batch-Fifo, SerialNo</Col>
        <Col id="LBKUM">1000,000</Col>
        <Col id="MEINS">PC</Col>
      </Row>
      <Row>
        <Col id="#WERKS">1000</Col>
        <Col id="MATNR">F234-2</Col>
        <Col id="MAKTX">FIN234-2, Forecast, Batch</Col>
        <Col id="LBKUM">5050,000</Col>
        <Col id="MEINS">PC</Col>
      </Row>
      <Row>
        <Col id="#WERKS">1000</Col>
        <Col id="MATNR">H21</Col>
        <Col id="MAKTX">Trading Good, Reorder Point, Batch-FIFO</Col>
        <Col id="LBKUM">3110,000</Col>
        <Col id="MEINS">PC</Col>
      </Row>
    </Rows>
  </Dataset>

```



다음과 같은 에러가 발생 할 경우 OS Platform , JRE , SAP dll bit version 을 동일하게 설정 합니다.

caused by java.lang.NoClassDefFoundError :

message=Could not initialize class com.sap.conn.jco.rt.JCoRuntimeFactory

혹은 Error getting the version of the native layer:

java.lang.UnsatisfiedLinkError: no sapjco3 in java.library.path

13.

기타 기능

이 장에서는 8~11장에서 다루지 않은 X-UP의 주요 서비스에 대해 소개하고 핸들링 하는 방법에 대하여 설명합니다.

13.1 Session Handling

이 절에서는 X-UP에서 Session을 Handling 하는 방법에 대해 설명합니다.

X-UP에서는 session 을 처리할 수 있는 UserSession 객체를 제공합니다.

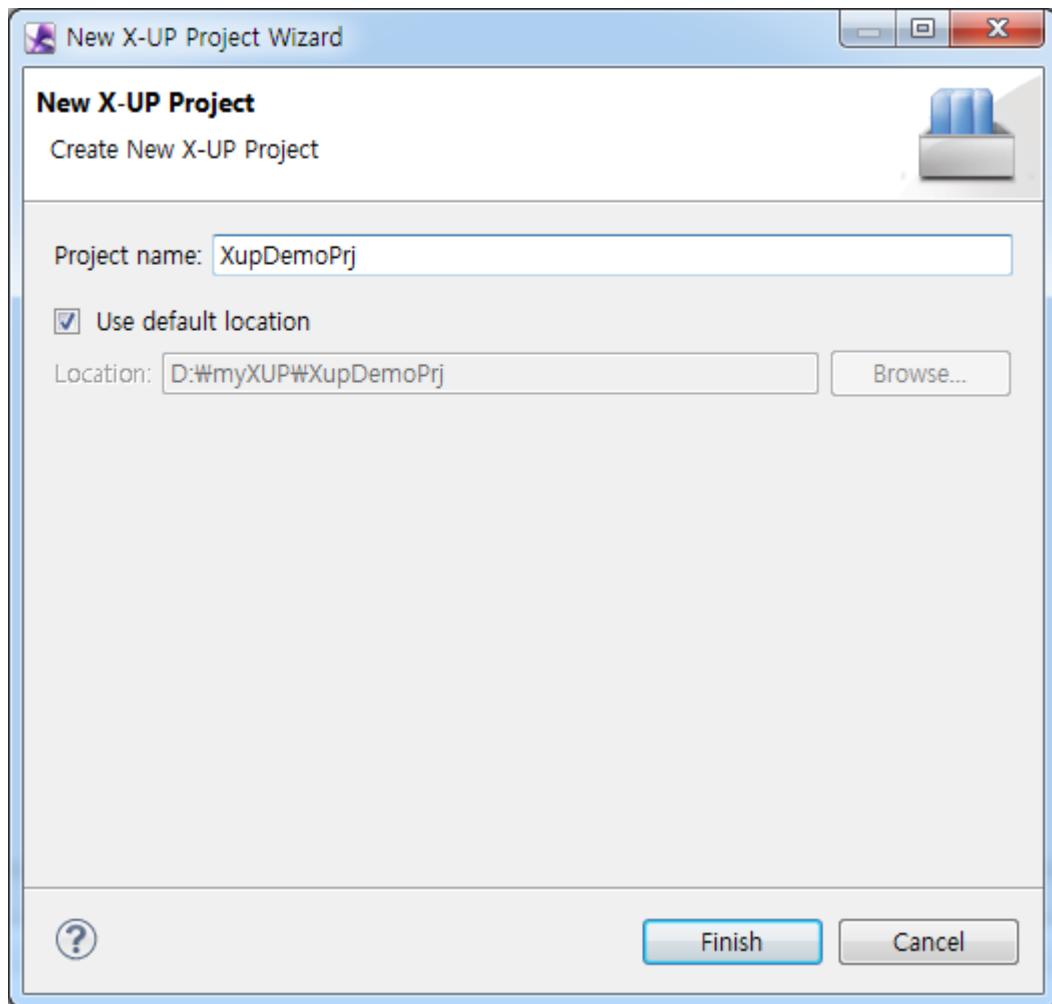
UserSession은 HttpSession객체를 Wrapping한 객체로 ,이 절에서는 UserSession객체를 사용하여 세션의 생성 , 세션값 체크 , 세션해제등의 기능을 로그인, 사용자정보가져오기, 로그인세션 해제하기라는 샘플 모델로 만들어 확인하는 방법을 설명합니다 .

다음은 Session 처리하는 서비스를 개발하는 단계는 다음과 같습니다.

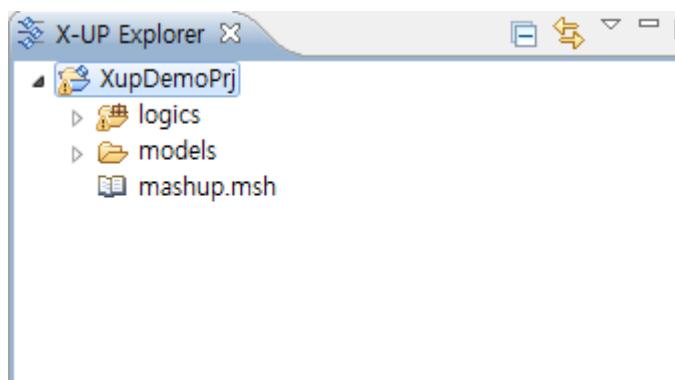
- X-UP 프로젝트 생성하기
- 로그인 Automation 모델 생성하기
- UserMethod 생성
- 세션값 체크 Automation 모델 생성하기
- user_id , user_pw 파라메터와 UserMethod 생성
- 세션해제 Automation 모델 생성하기
- UserMethod 생성
- 모델 테스트하기

X-UP 프로젝트 생성하기

1. [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
2. **New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭 합니다.

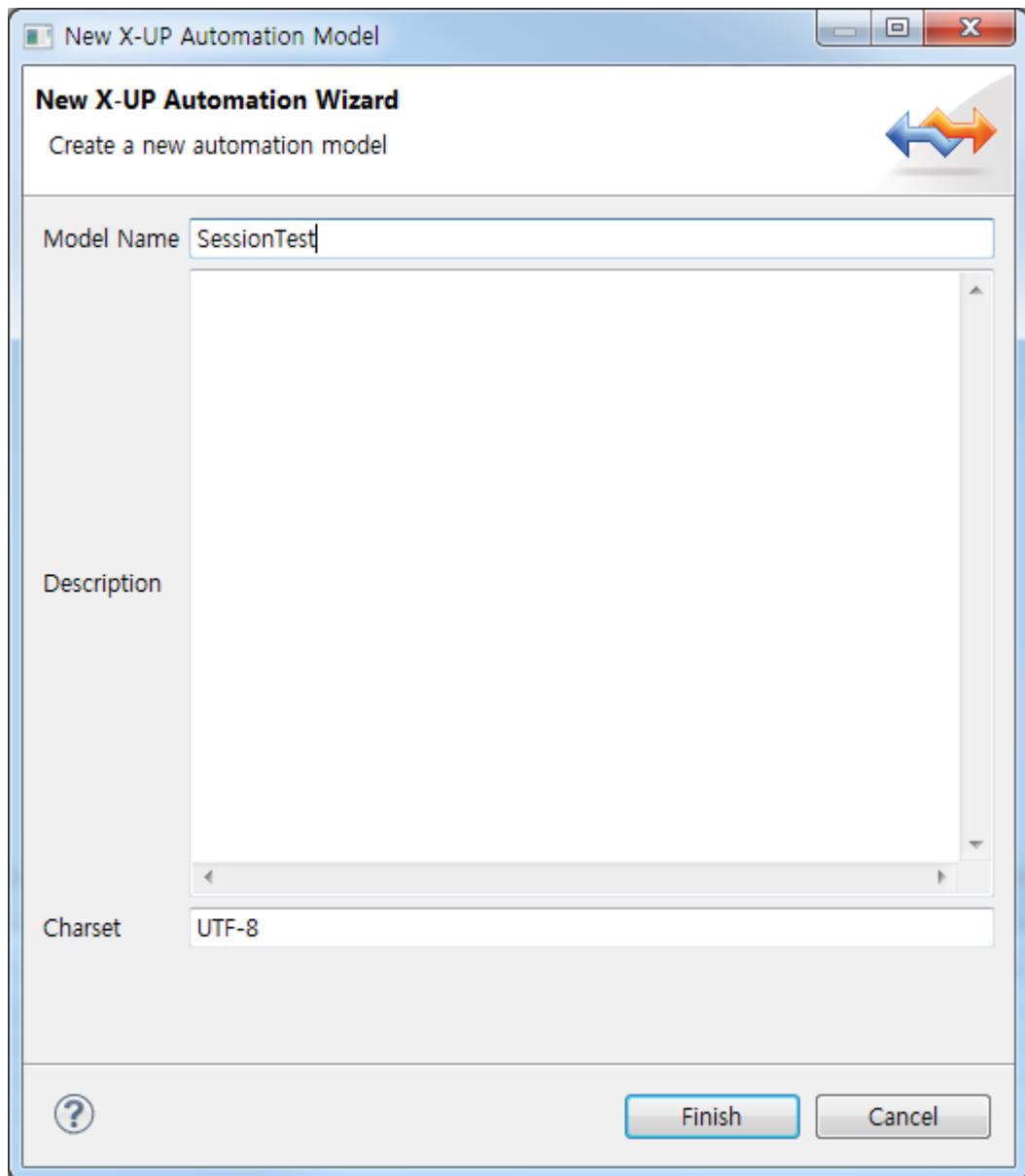


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

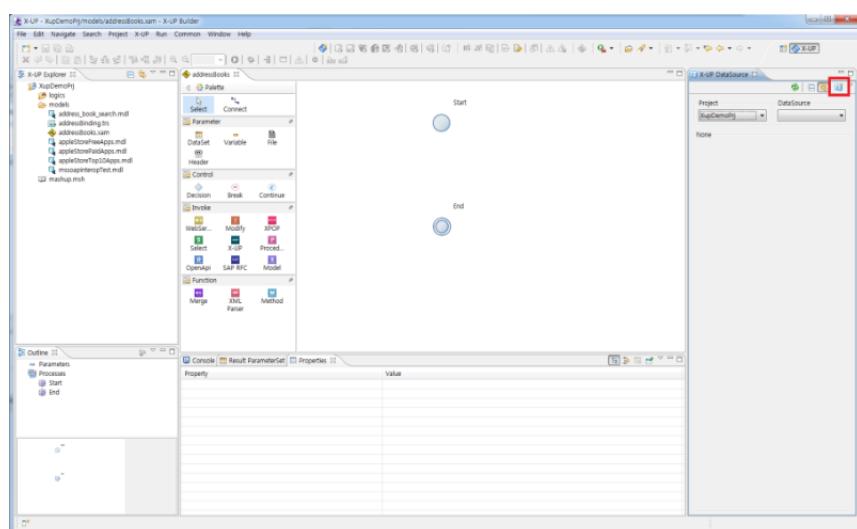


Automation 모델 생성하기

- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 OK 버튼을 클릭합니다.



3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



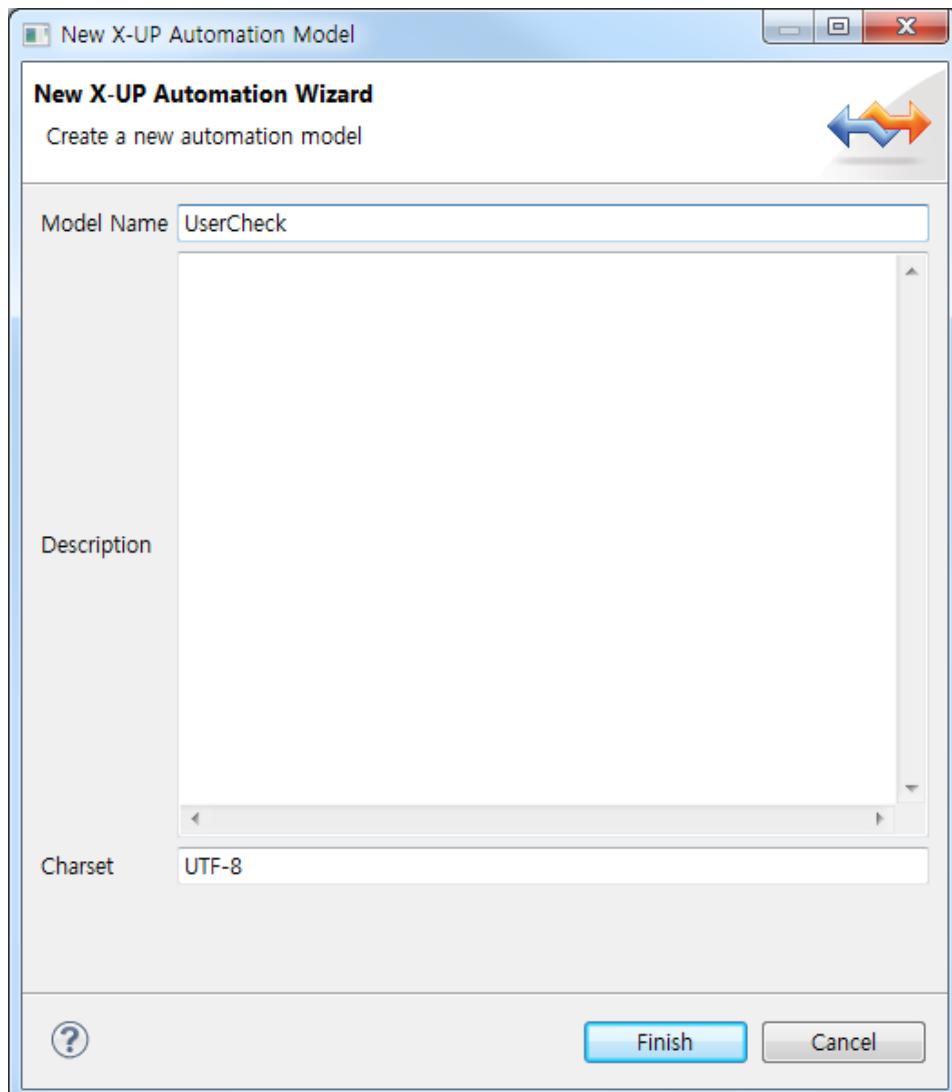
UserMethod 생성하기

1. UserMethod를 더블클릭하여 아래와 같이 소스코드를 입력합니다.

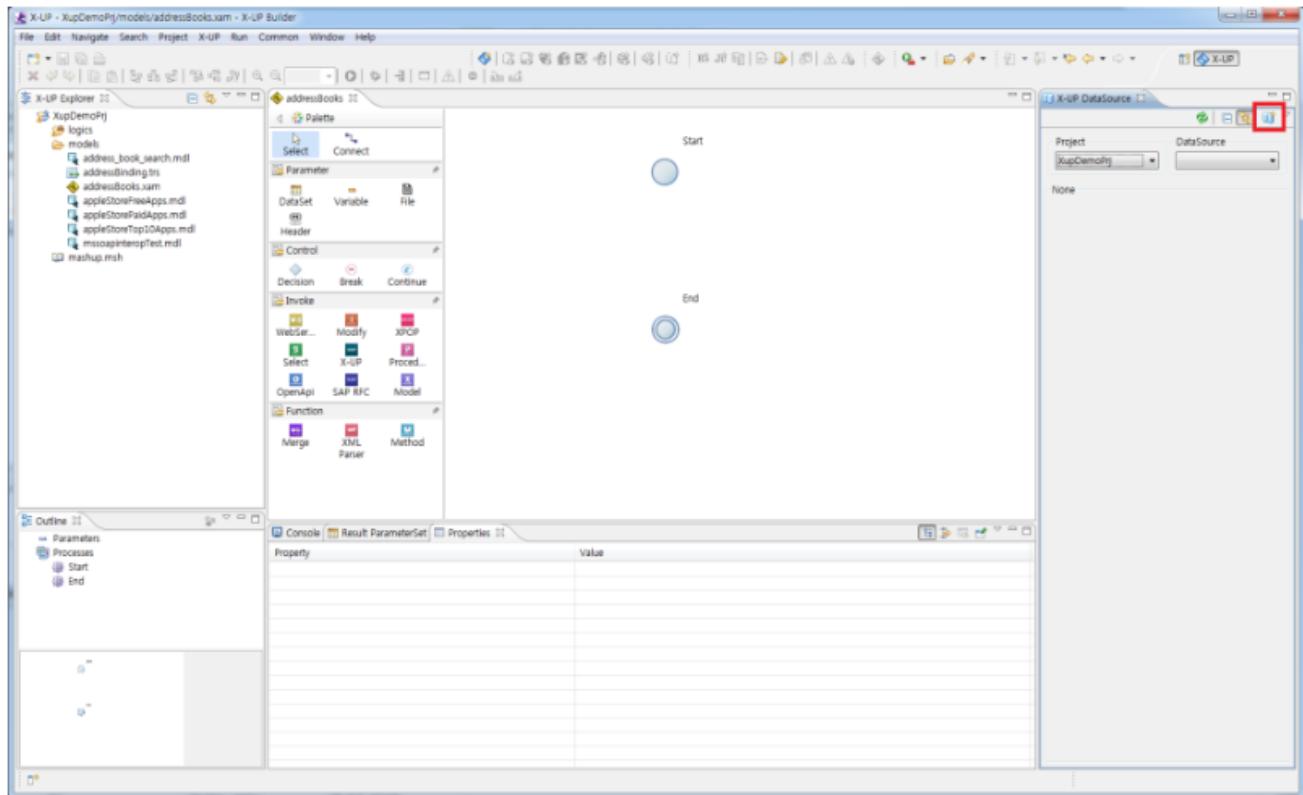
```
// session 생성  
UserSession session = this.getUserSession();  
session.setAttribute("USER_ID", "test");  
session.setAttribute("USER_PW", "1234");
```

세션값 체크 Automation 모델 생성하기

1. [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 OK 버튼을 클릭합니다.



3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.

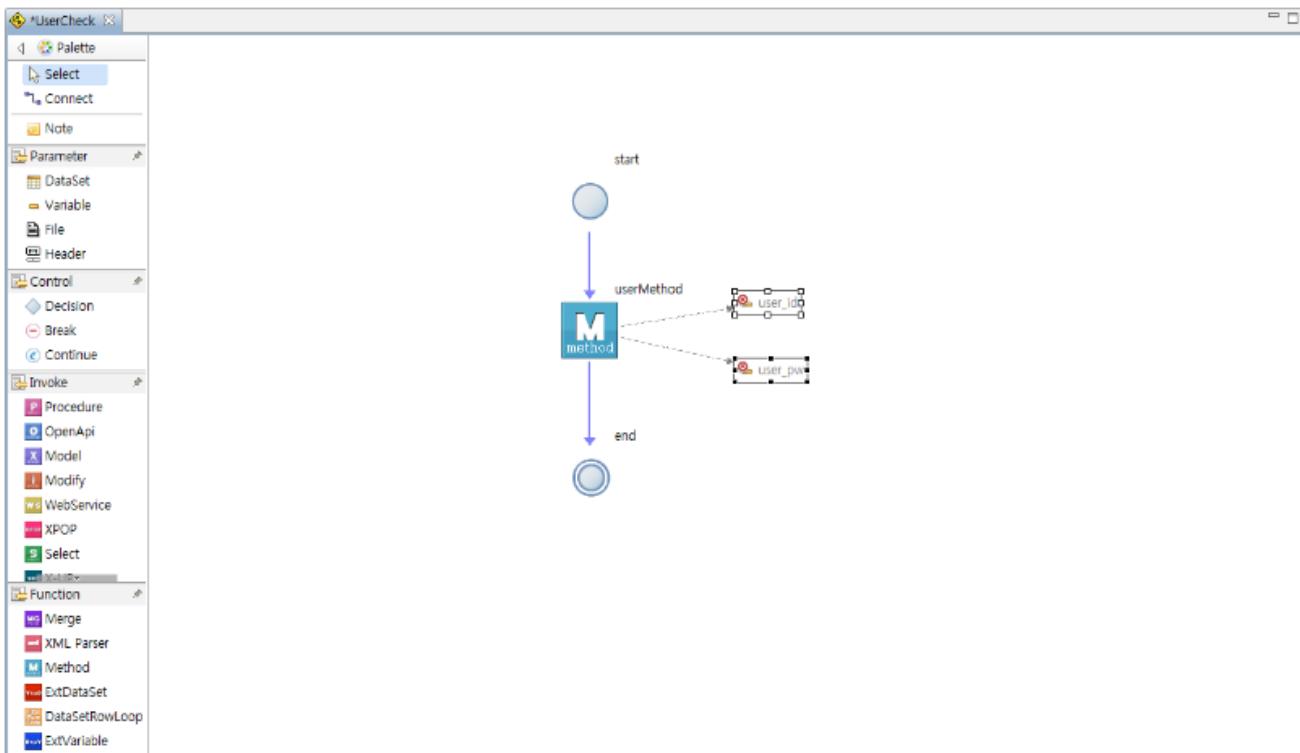


user_id , user_pw 파라메터와 UserMethod 생성하기

1. Variable 파라메터 두개를 선언하고 이름을 user_id , user_pw로 지정합니다.
2. UserMethod 컴포넌트를 더블클릭하여 아래와 같이 소스코드를 입력합니다.

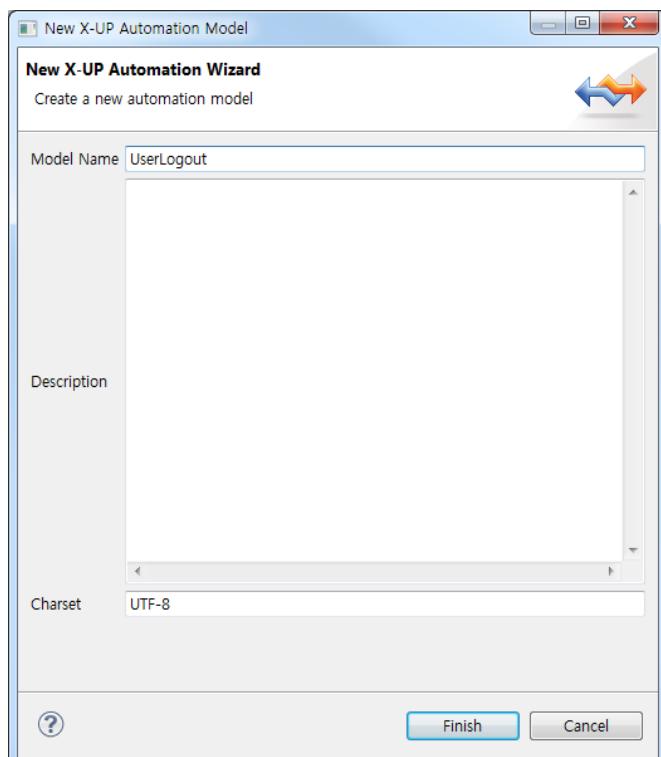
```
// session 생성
UserSession session = this.getUserSession();
globalParameterSet.add("user_id",session.getAttribute("USER_ID"));
globalParameterSet.add("user_pw",session.getAttribute("USER_PW"));
```

3. user_id , user_pw variable을 usermethod에 output 파라메터로 Connect합니다

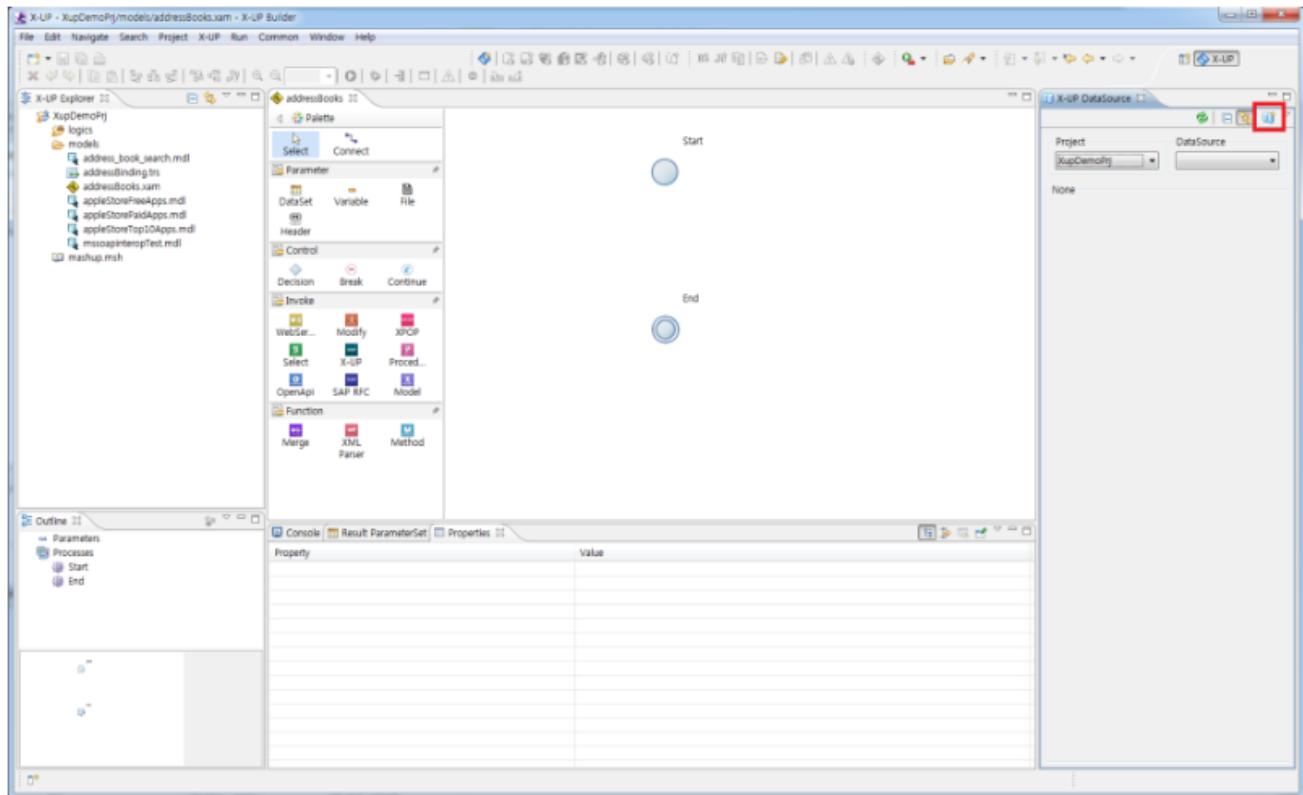


세션해제 Automation 모델 생성하기

- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 OK 버튼을 클릭합니다.



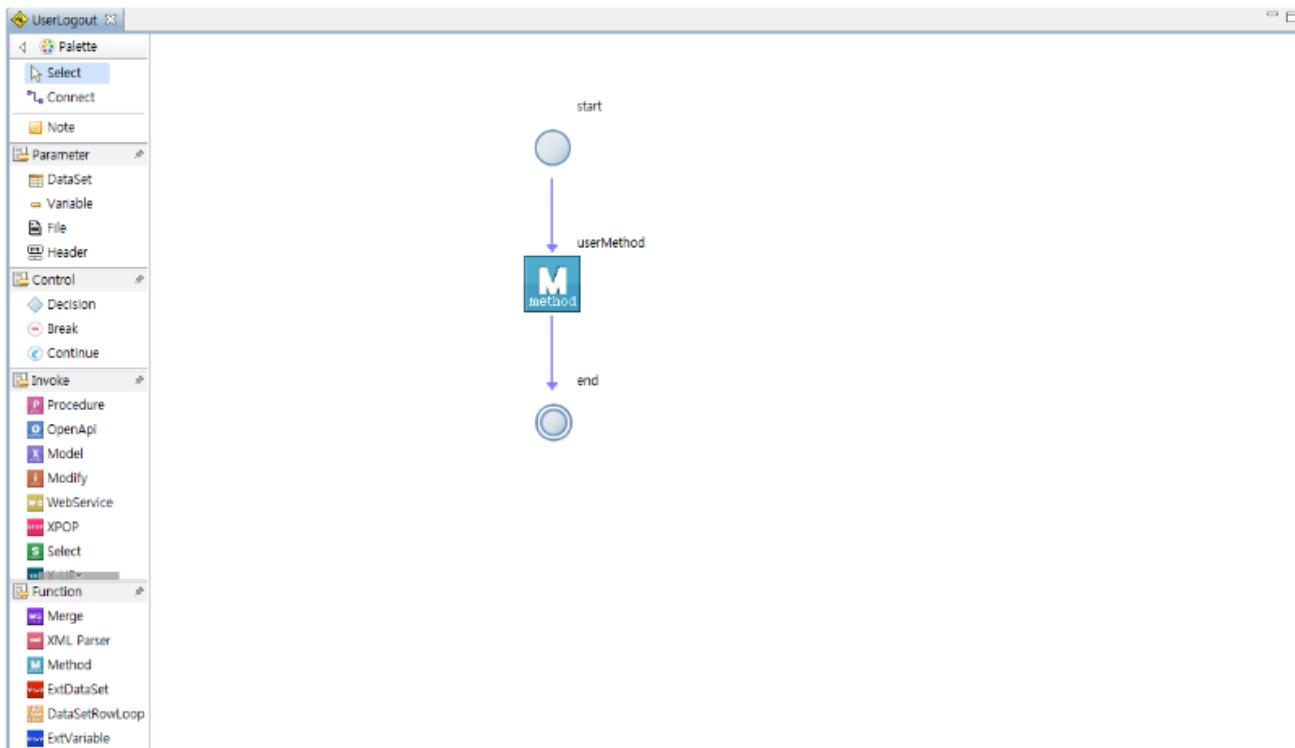
3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



UserMethod 생성하기

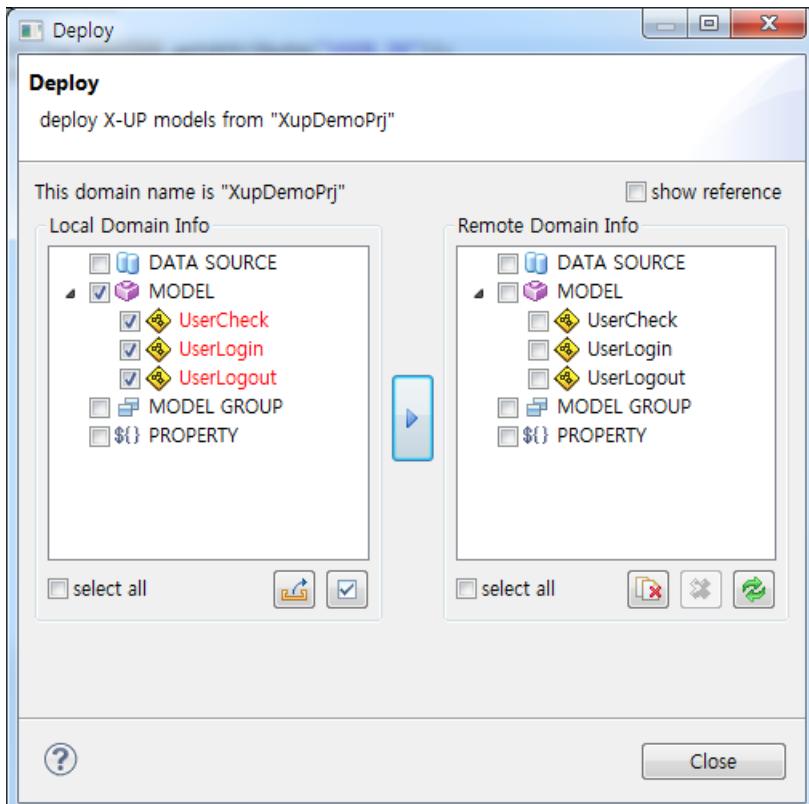
1. Variable 파라메터 두개를 선언하고 이름을 user_id , user_pw로 지정합니다.
2. UserMethod 컴포넌트를 더블클릭하여 아래와 같이 소스코드를 입력합니다.

```
// session 해제
UserSession session = this.getUserSession();
session.invalidate(true);
```

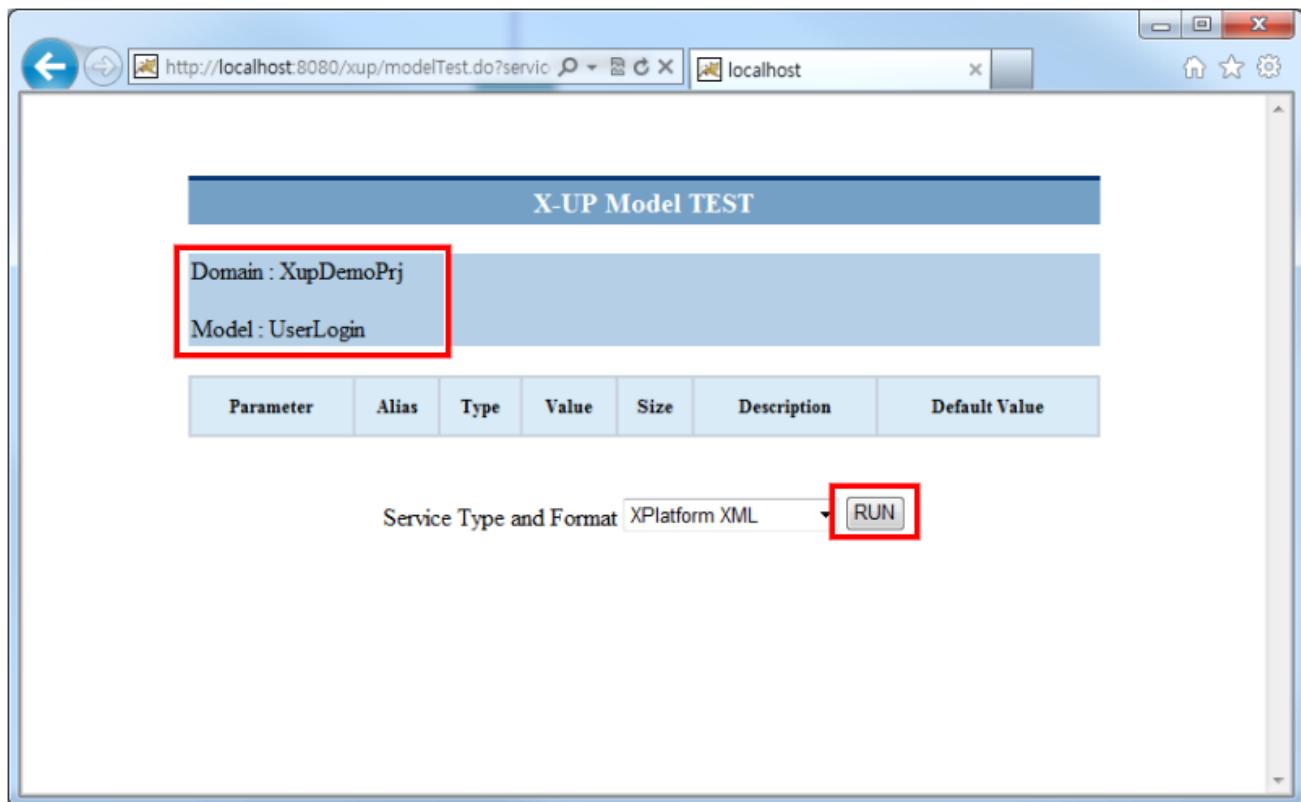


테스트

1. 위에서 작성한 모델 3개를 서버로 디플로이 합니다.



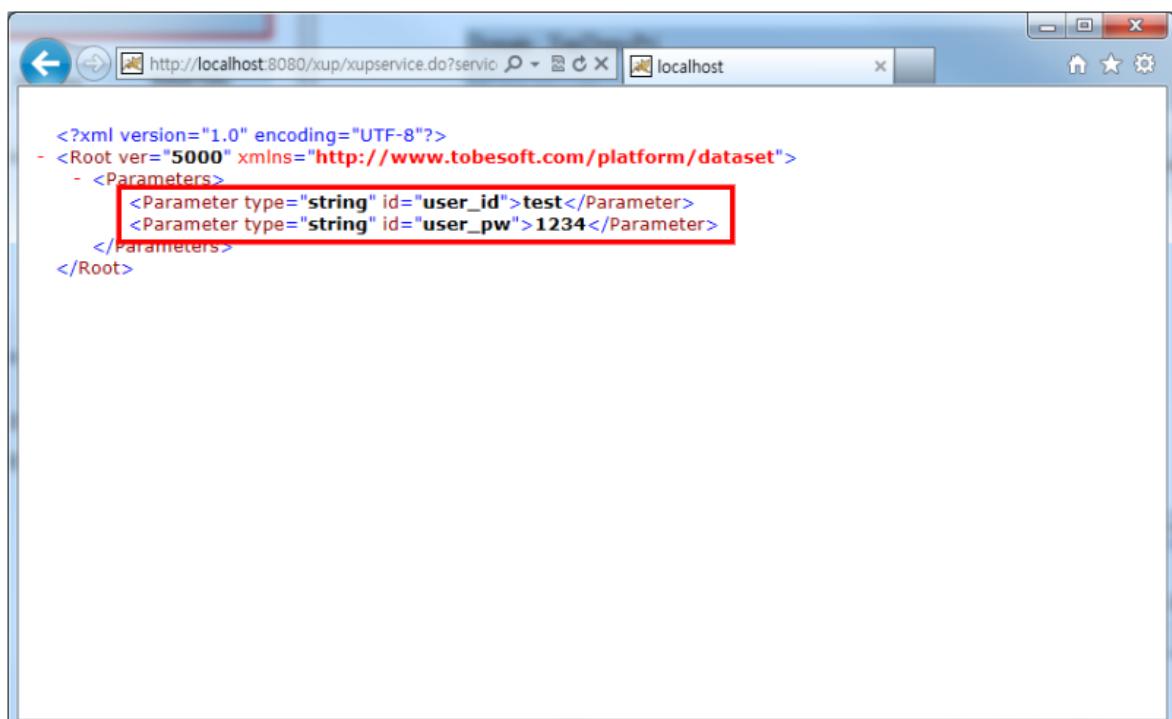
2. UserLogin 모델을 호출합니다.



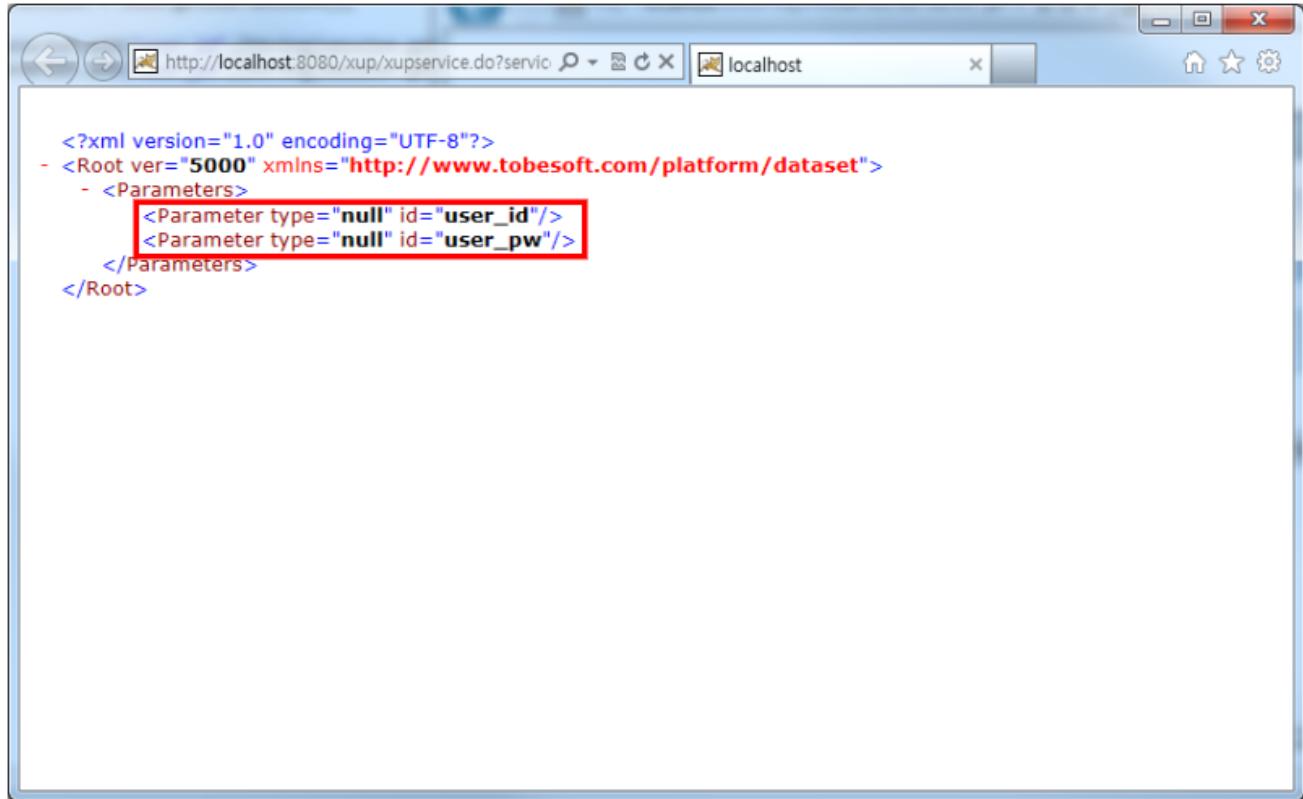
3. 위 모델을 호출을 했으면 UserCheck 모델도 호출합니다.

UserLogin 모델이 정상적으로 호출되었으면 session 값에 user_id, user_pw 값이 할당되어있어야 합니다.

4. User_id, user_pw 값이 정상적으로 출력되는지 확인합니다.



5. 위 정보를 확인하고 세션을 해제하는 UserLogout 모델을 호출합니다. 호출하게되면 이전에 생성한 세션정보를 잃게 됩니다.
6. 위 단계를 실행하고 UserCheck 모델을 호출해 이전에 저장한 세션값이 null이 되었는지 확인합니다.



```

<?xml version="1.0" encoding="UTF-8"?>
- <Root ver="5000" xmlns="http://www.tobesoft.com/platform/dataset">
  - <Parameters>
    <Parameter type="null" id="user_id"/>
    <Parameter type="null" id="user_pw"/>
  </Parameters>
</Root>

```

13.2 Cookie Handling

Cookie란 웹서버가 클라이언트에게 전송한 텍스트로 클라이언트에 저장되어 동일한 웹서버에 접속할 때마다 요청 헤더에 key=value 형식으로 자동으로 서버에 전달되는 값입니다.

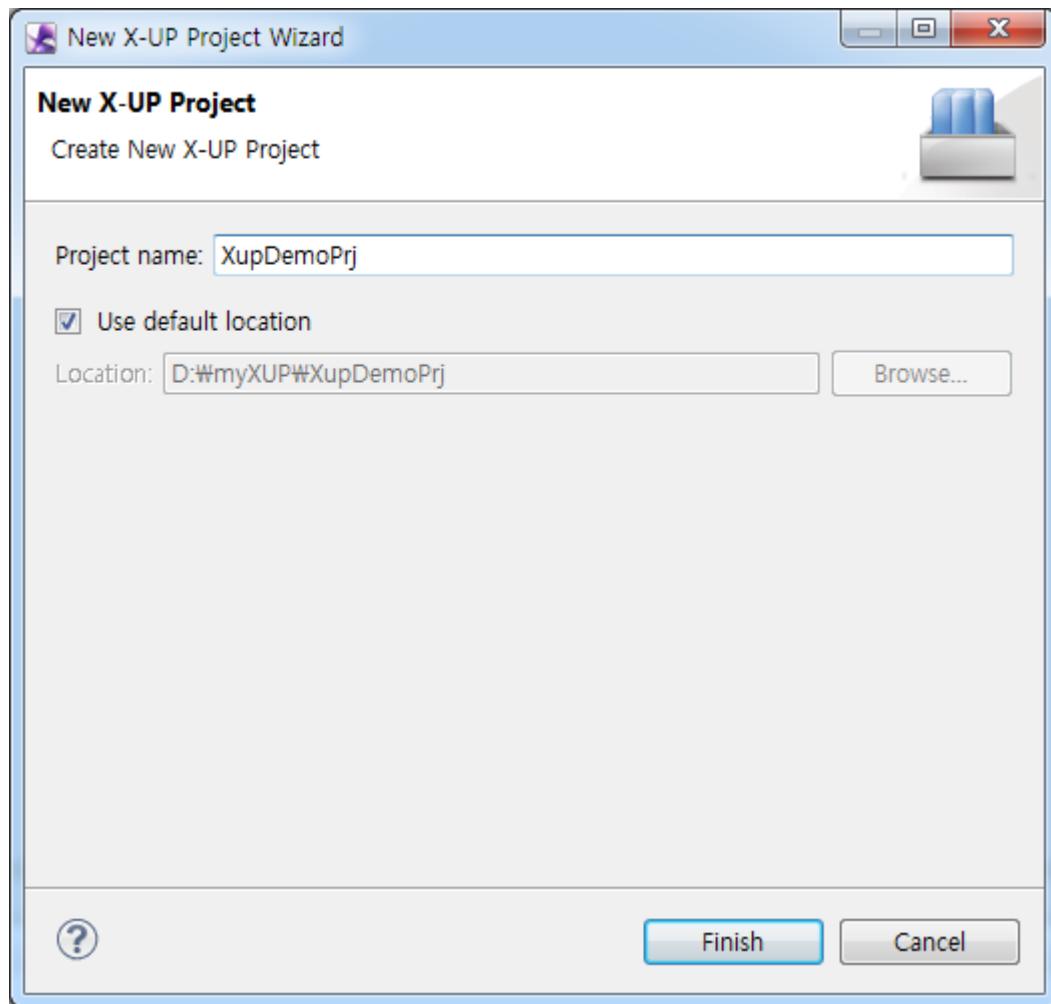
이 절에서는 Cookie Handling 하는 방법을 설명합니다. X-UP의 header 파라미터를 이용하여 클라이언트에서 넘어오는 header 값을 읽어 그 중 cookie 값을 변경하는 방법을 설명합니다.

이 절에서의 모델 개발 단계는 다음과 같습니다.

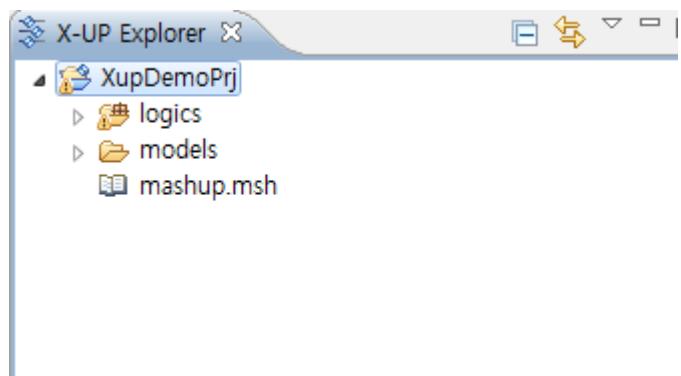
- X-UP 프로젝트 생성하기
- Automation 모델 생성하기
- Header 파라미터와 UserMethod 생성하기
- UserMethod 로직 구현하고 output 파라미터 설정하기
- 모델 테스트하기

X-UP 프로젝트 생성하기

- [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.
- New X-UP Project Wizard**에서 **Project name** 필드에 원하는 프로젝트 이름을 입력하고 ‘Finish’ 버튼을 클릭합니다.

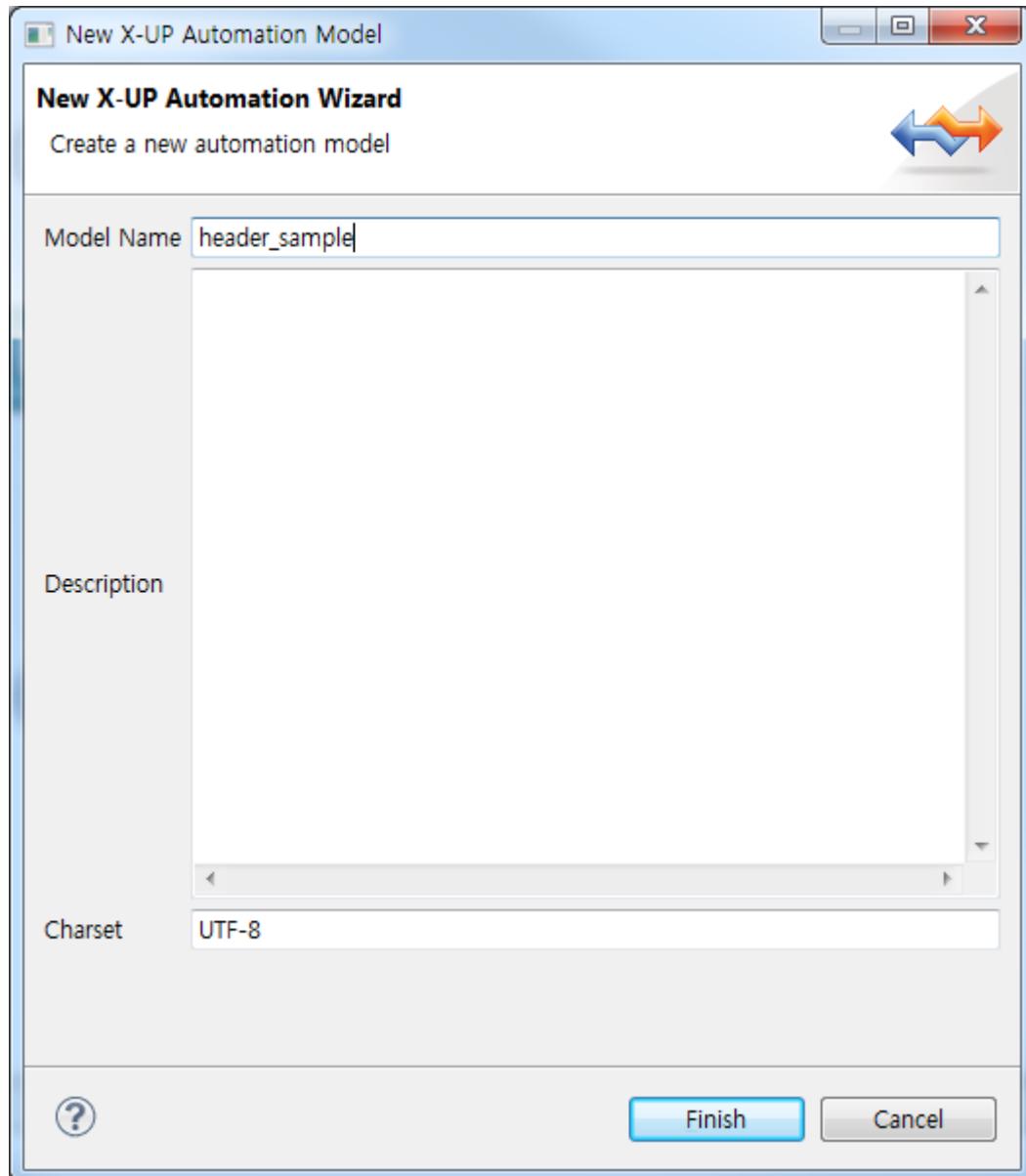


- X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

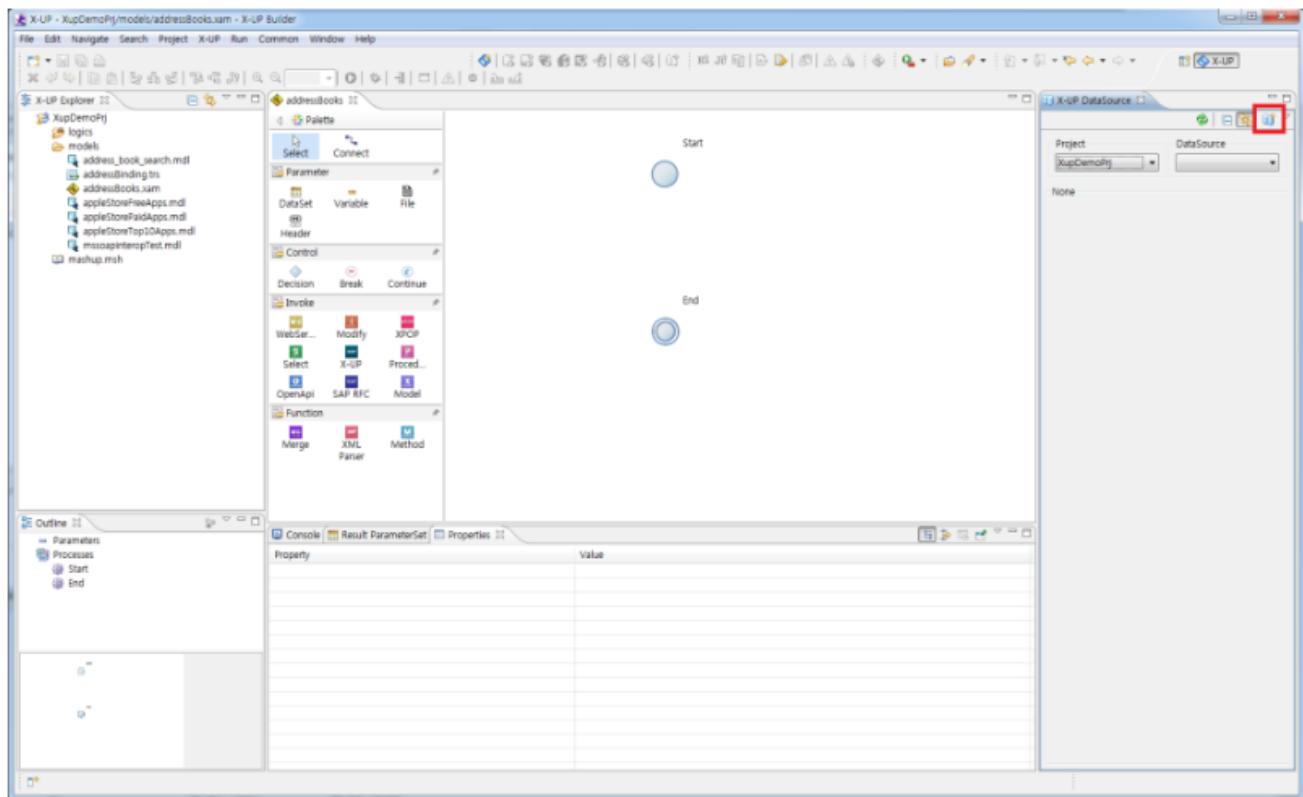


Automation 모델 생성하기

- [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
- New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 OK 버튼을 클릭합니다.

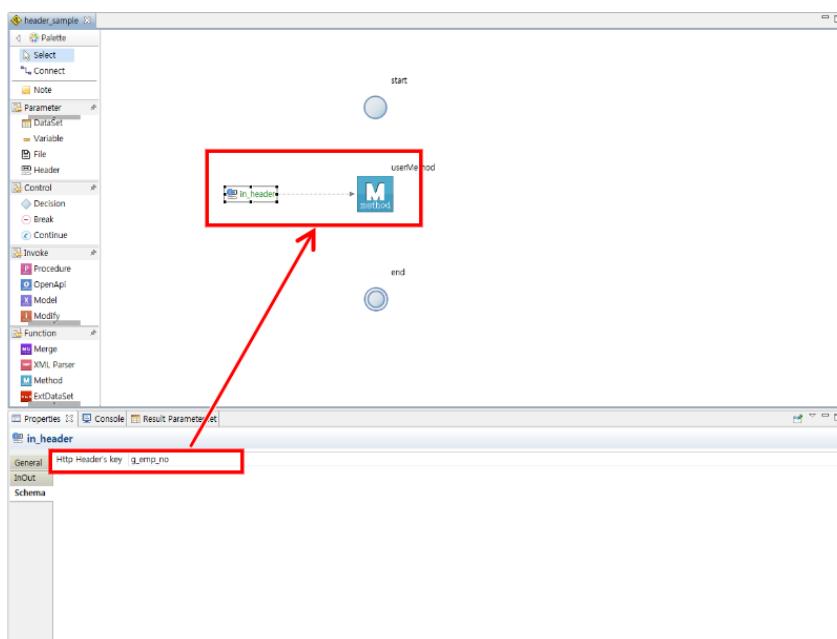


3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



Header 파라미터와 UserMethod 생성하기

1. header 파라미터를 추가하고, 이름은 " cookie ", Schema는 "cookie"로 입력해줍니다.
2. UserMethod를 추가한 후 생성한 UserMethod로 input 파라미터로 cookie를 Connect합니다.

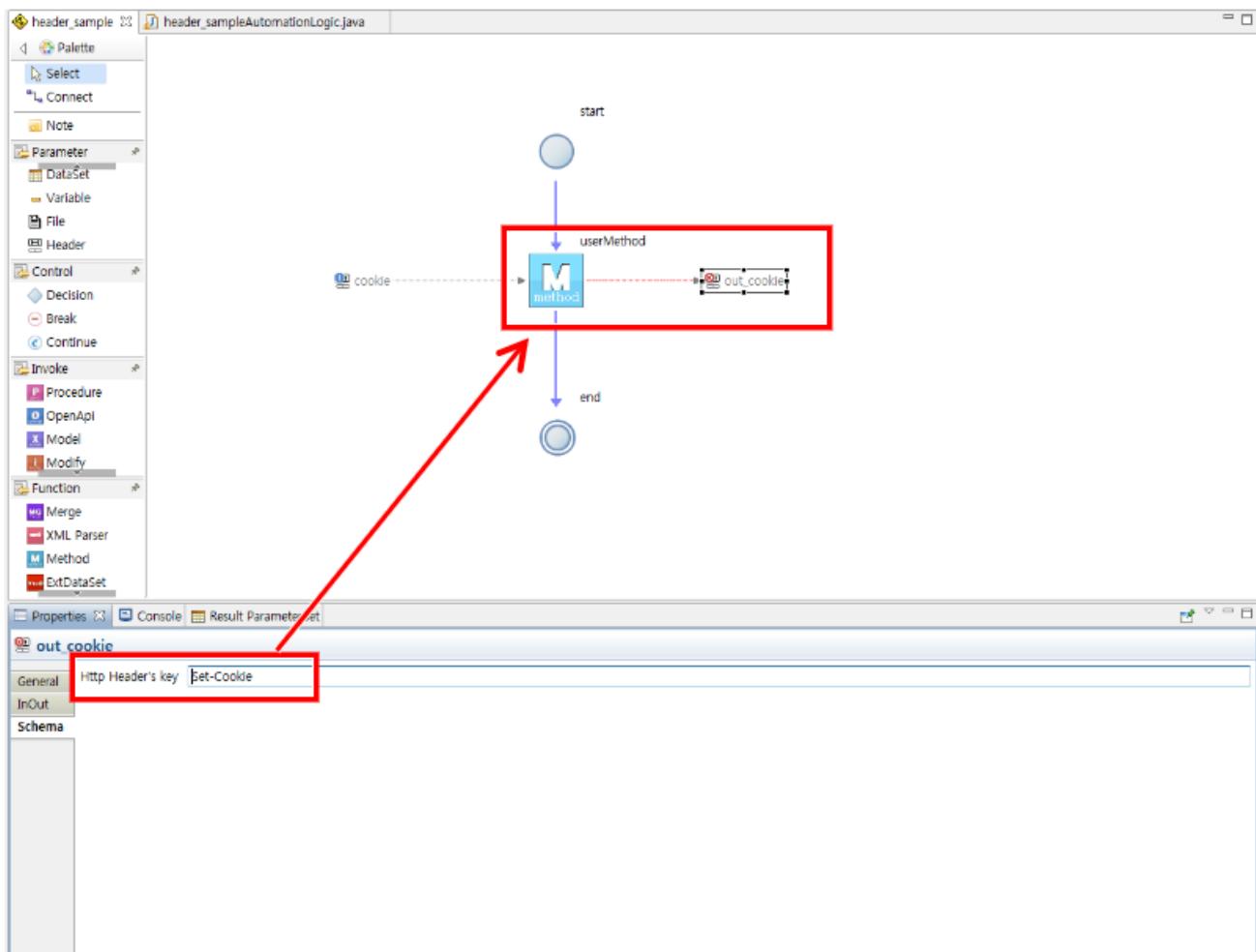


UserMethod 로직 구현하고 output 파라미터 설정하기

1. UserMethod를 더블클릭하여 아래와 같이 소스코드를 입력합니다.

```
//클라이언트로 부터 받아온 header 중 cookie 값을 얻는다.
MashupHeader cookie= globalParameterSet.getHeader("cookie");
//key = value 형식으로 되어있는 값을 파싱한다.
String [] arrCookie = cookie.getValue().trim().split(";");
String cookieFullValue = "";
for(int iKey = 0; iKey < arrCookie.length; iKey++) {
    String[] cookieKeyVal = arrCookie[iKey].trim().split("=");
    String key = "";
    String value = "";
    if(cookieKeyVal.length == 1) {
        key = cookieKeyVal[0];
        value = "";
    } else {
        key = cookieKeyVal[0];
        value = cookieKeyVal[1];
    }
    if(key.equals("g_emp_no")) {
        value = "54321";
    }
    cookieFullValue += key+"="+value+"; ";
}
MashupHeader cookie1= globalParameterSet.getHeader("out_cookie");
cookie1.setValue(cookieFullValue);
```

2. header 파라메터를 추가하고, 이름은 " out_cookie ", Schema는 "Set-Cookie" InOut타입은 “out”으로 입력해줍니다.
3. UserMethod와 생성한 out_cookie 파라미터를 connect 합니다.



- Start 노드와 End 노드까지 위에서 생성한 모든 컴포넌트를 Connect 합니다

테스트 화면만들기

UX-Studio에서 프로젝트 만들기, 폼화면 만들기 설명은 생략합니다.

- UX-Studio를 실행하고 “header_Form”이라는 이름의 form 화면 하나를 생성합니다.
- GlobalVariables 의 variable을 아래와 같이 생성합니다.

globalvars.xml [Variables]			
No	id	initval	usecookie
1	g_emp_no		true

- 생성한 화면에 버튼을 추가하고 더블클릭하여 아래와 같은 function 스크립트를 입력합니다.

```

MashupFile file = globalParameterSet.getParameter("file").getFile();
function Button00_onclick(obj:Button, e:ClickEventInfo)
{
    alert(g_emp_no);
    var svcID = "testService";
    var svcparam = "domain=XupCOMM" // X-UP 도메인 이름
        +"&model=header_sample&service=xupservice" // X-UP 모델 이름
        +"&format=xml"
        +"&version=xplatform";

    var svcurl = "SERVER::xupservice.do?" + svcparam;
    var inputDataset = "";
    var outputDataset = "";
    var strArgument = "";
    transaction(svcID, svcurl, inputDataset, outputDataset, strArgument,
        "CallbackFunc",false);
}

function CallbackFunc(strSvcID, nErrorCode, strErrorMag)
{
    if (nErrorCode != 0)
    {
        alert(nErrorCode + " : " + strErrorMag);
        return;
    }
    alert(g_emp_no);
}

```

4. 폼 화면을 실행시켜 버튼을 클릭합니다.
5. 최초 쿠키 값이 빈값으로 출력되는지 확인합니다.



6. 다시 한번 OK 버튼을 클릭해 변경된 쿠키값이 들어오는지 확인합니다.



13.3 File Handling

이 절에서는 File 파라메터를 이용한 모델 개발방법에 대해 설명합니다.



이 절에서 사용한 데이터베이스 테이블 정보는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 참조하십시오.

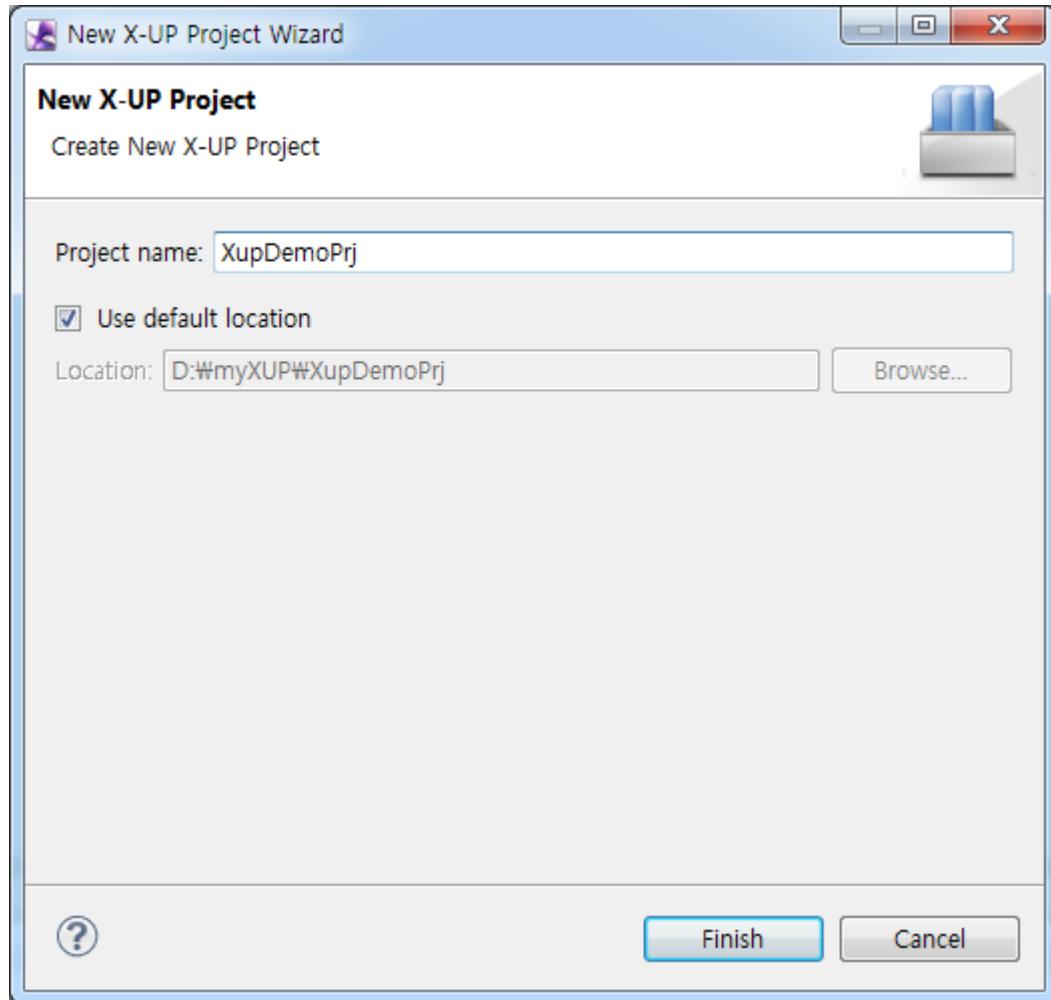
이 절에서 설명하는 File 파라메터를 이용한 모델 개발단계는 다음과 같습니다.

- X-UP 프로젝트 생성하기
- Automation 모델 생성하기
- File 파라메터와 UserMethod 생성하기
- UserMethod 로직 구현하고 테스트 파라메터 설정하기
- Modify Invoke 및 Select Invoke 생성하고 출력 로직 설정하기
- 모델 테스트하기

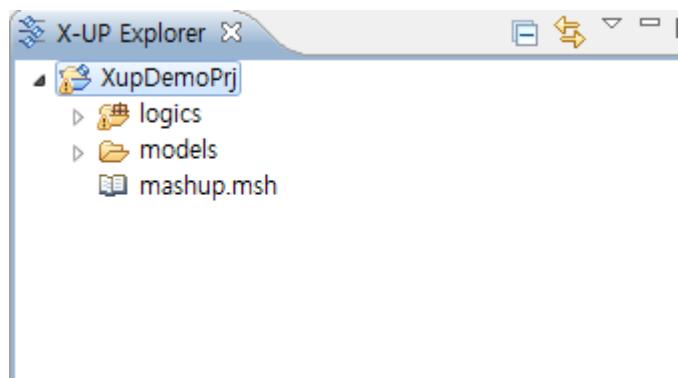
X-UP 프로젝트 생성하기

- [File > New > X-UP Project] 메뉴를 선택하여 **New X-UP Project Wizard**를 실행합니다.

2. New X-UP Project Wizard에서 Project name 필드에 원하는 프로젝트 이름을 입력하고 'Finish' 버튼을 클릭 합니다.

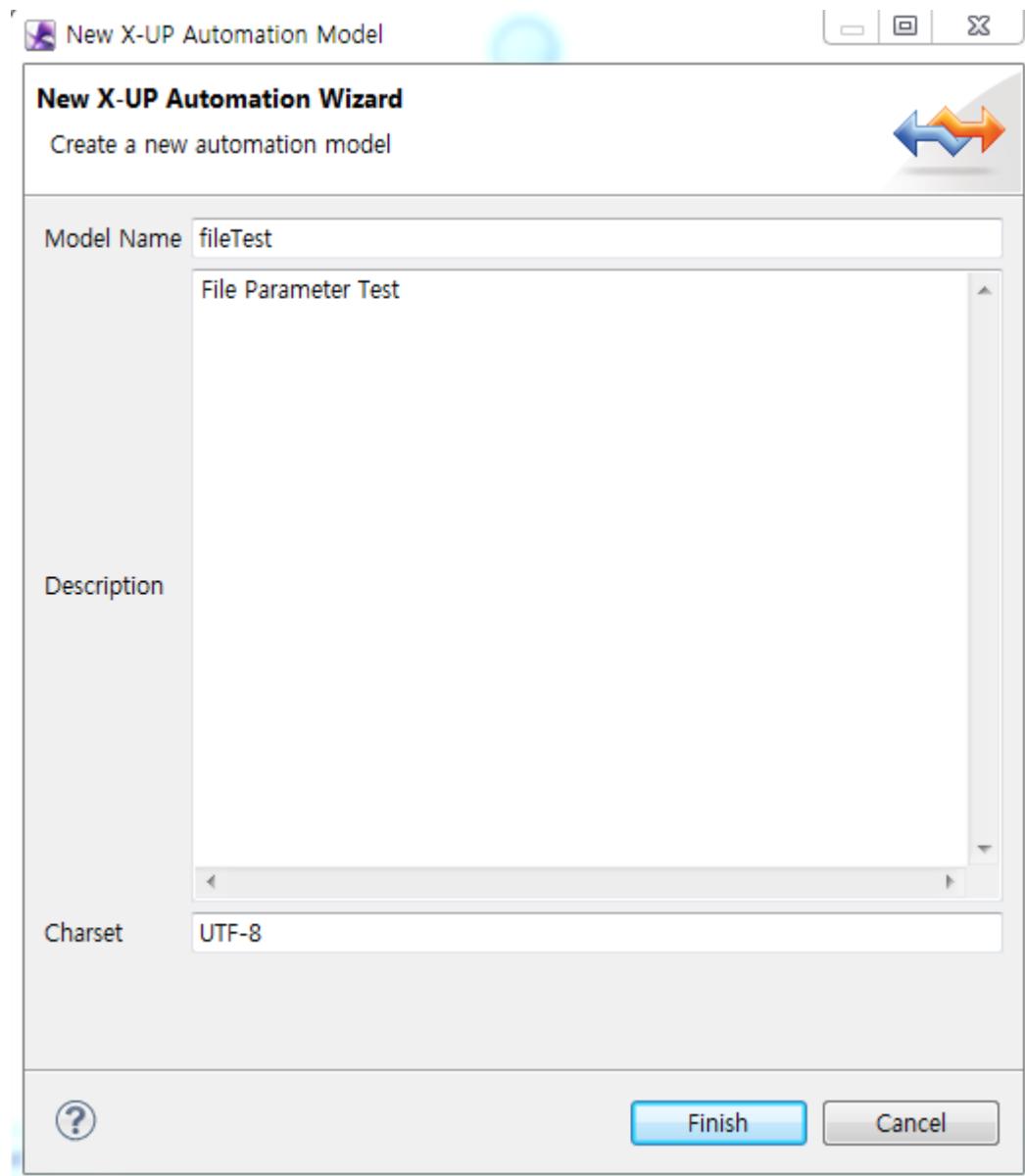


3. X-UP Explorer에 아래와 같이 X-UP 프로젝트가 생성된 것을 확인합니다.

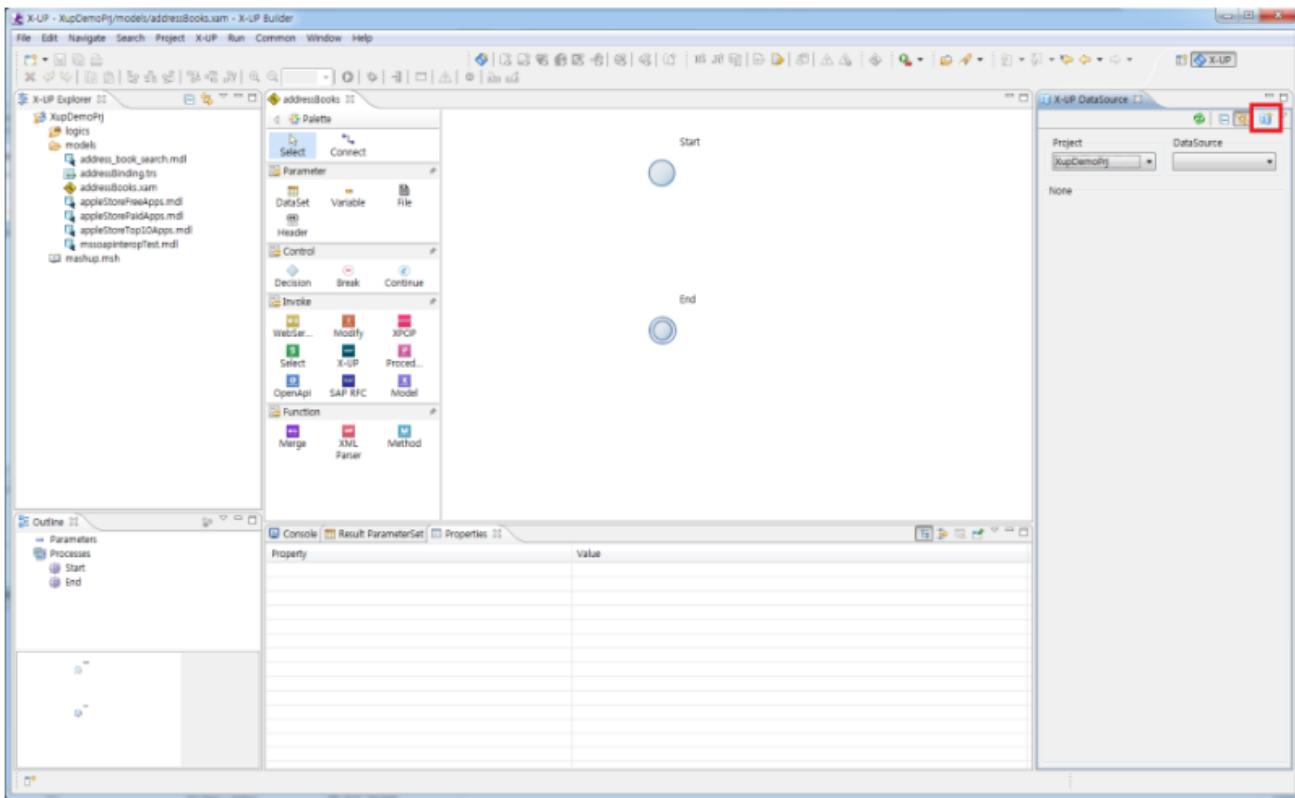


Automation 모델 생성하기

1. [File > New > X-UP Automation Model] 메뉴를 선택하여 New Model Wizard를 실행합니다.
2. New Model Wizard에서 Model Name 필드에 모델 이름을 입력하고 OK 버튼을 클릭합니다.

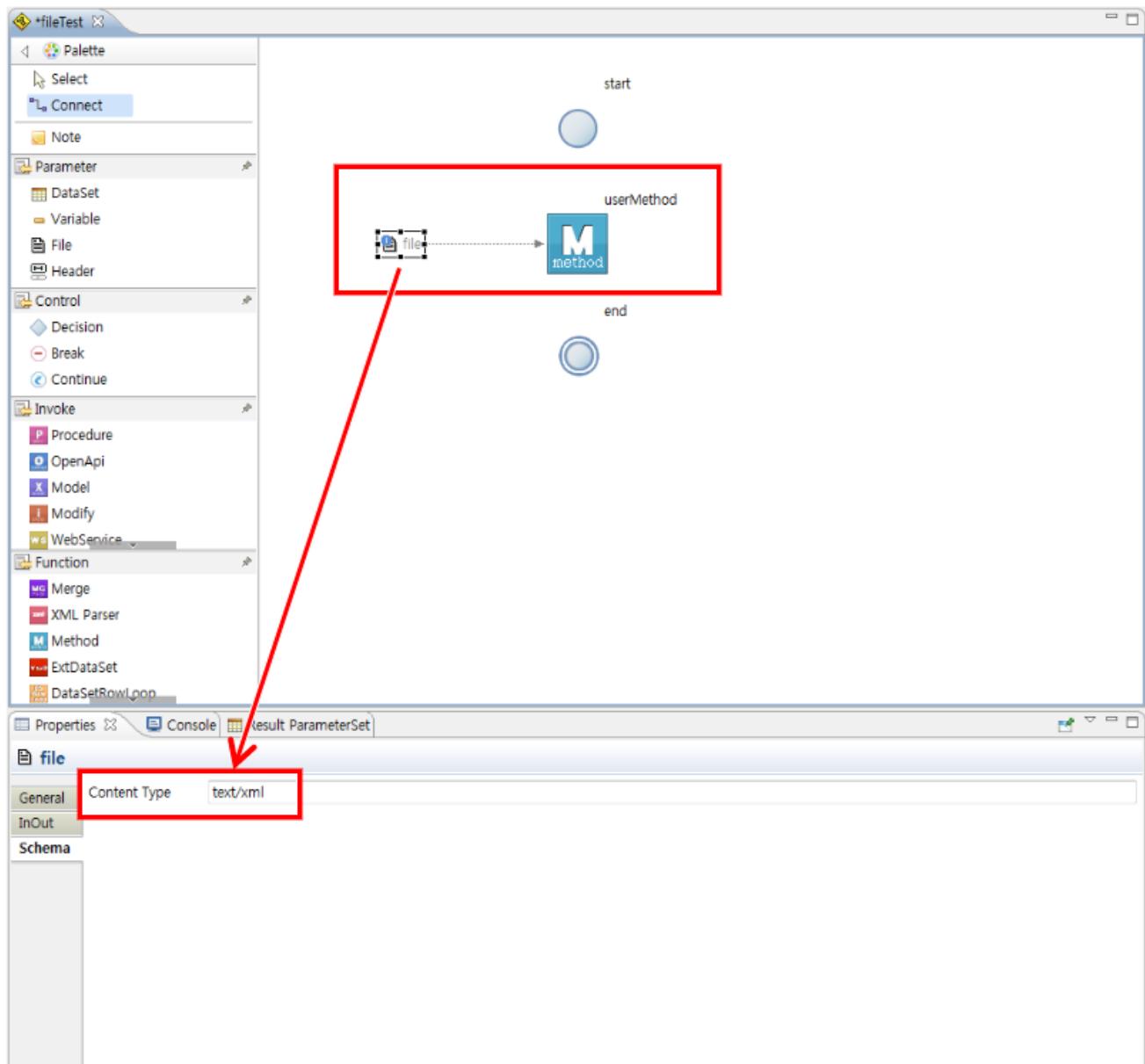


3. New Model Wizard가 완료 후 나타나는 화면은 아래와 같습니다.



File 파라미터와 UserMethod 생성하기

1. File 파라미터를 추가하고, 이름은 ‘file’, Schema는 ‘text/xml’로 입력해줍니다.
2. UserMethod 함수를 추가한 후 생성한 UserMethod함수에 input 파라미터로 위에서 추가한 파일 파라미터 file을 Connect합니다.



UserMethod 로직 구현하고 테스트 파라미터 설정하기

1. UserMethod를 더블클릭하여 아래와 같이 소스코드를 입력합니다.

```

MashupFile file = globalParameterSet.getParameter("file").getFile();

String name = file.getName();
String path = file.getPath();

DataSet dataset1 = globalParameterSet.getDataSet("dataset1");

```

```

int row = dataset1.newRow();
dataset1.set(row, "NAME", name);
dataset1.set(row, "PATH", path);

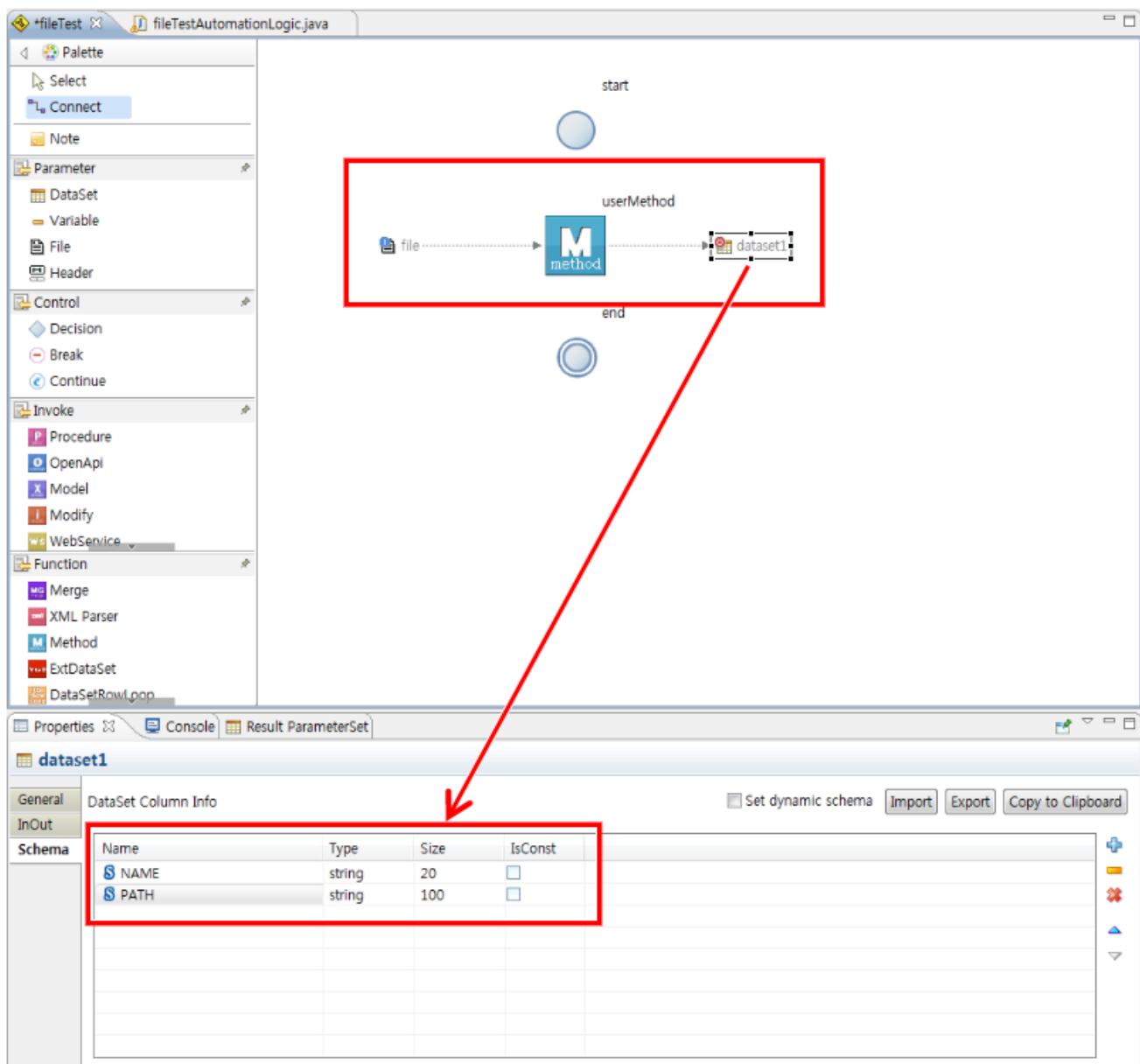
globalParameterSet.add("dataset1", dataset1);

```

2. DataSet을 추가하고 이름은 "dataset1"로하고 스키마는 아래와 같이 정의합니다.

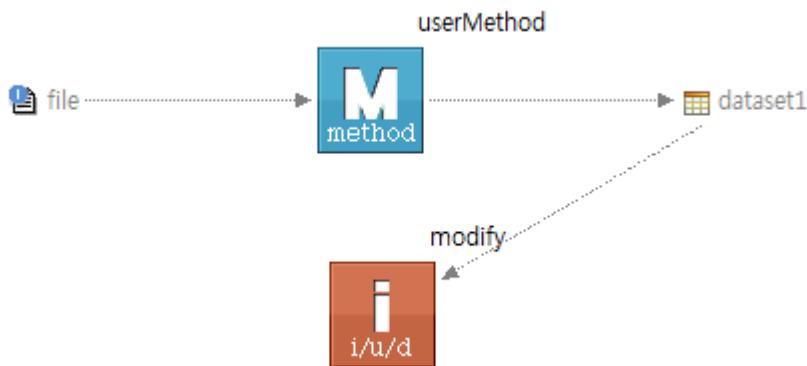
Name	Type	Size
NAME	string	20
PATH	string	100

3. UserMethod로 부터 output 파라메터를 dataset1로 Connect합니다.



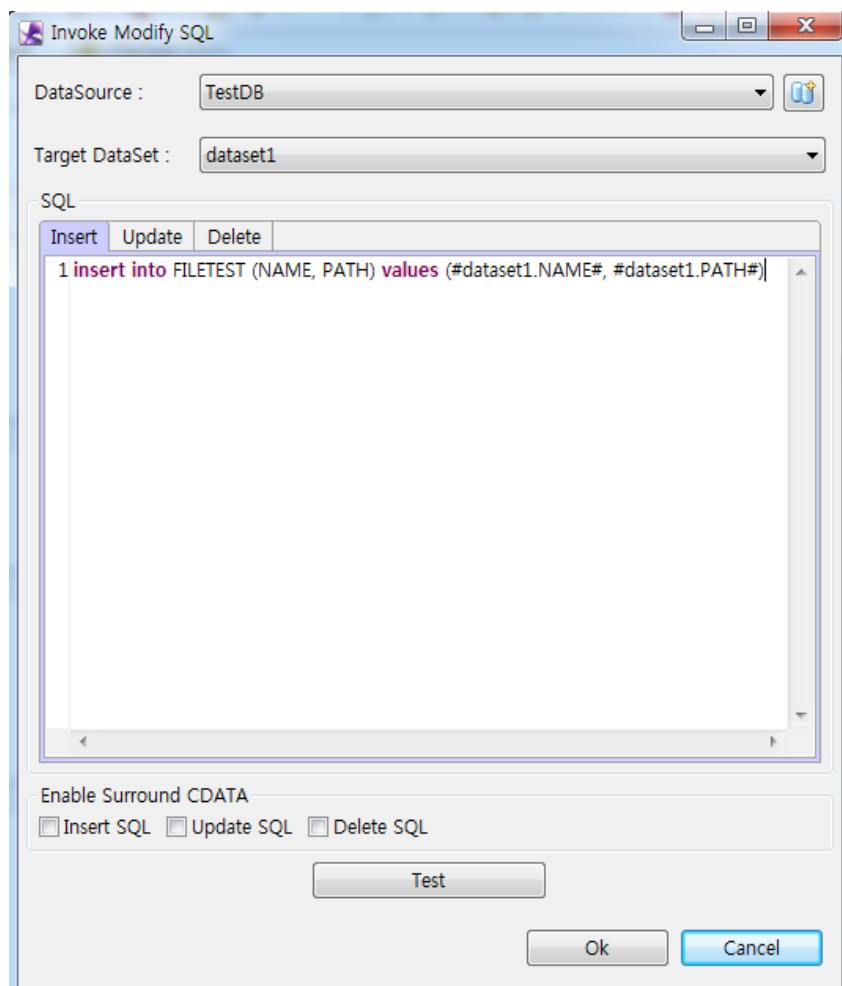
Modify Invoke 및 Select Invoke 생성하고 출력 로직 설정하기

1. Modify Invoke를 추가합니다.
2. 타겟 DataSet은 dataset1이며 Modify Invoke에 input 파라미터로 Connect해줍니다.



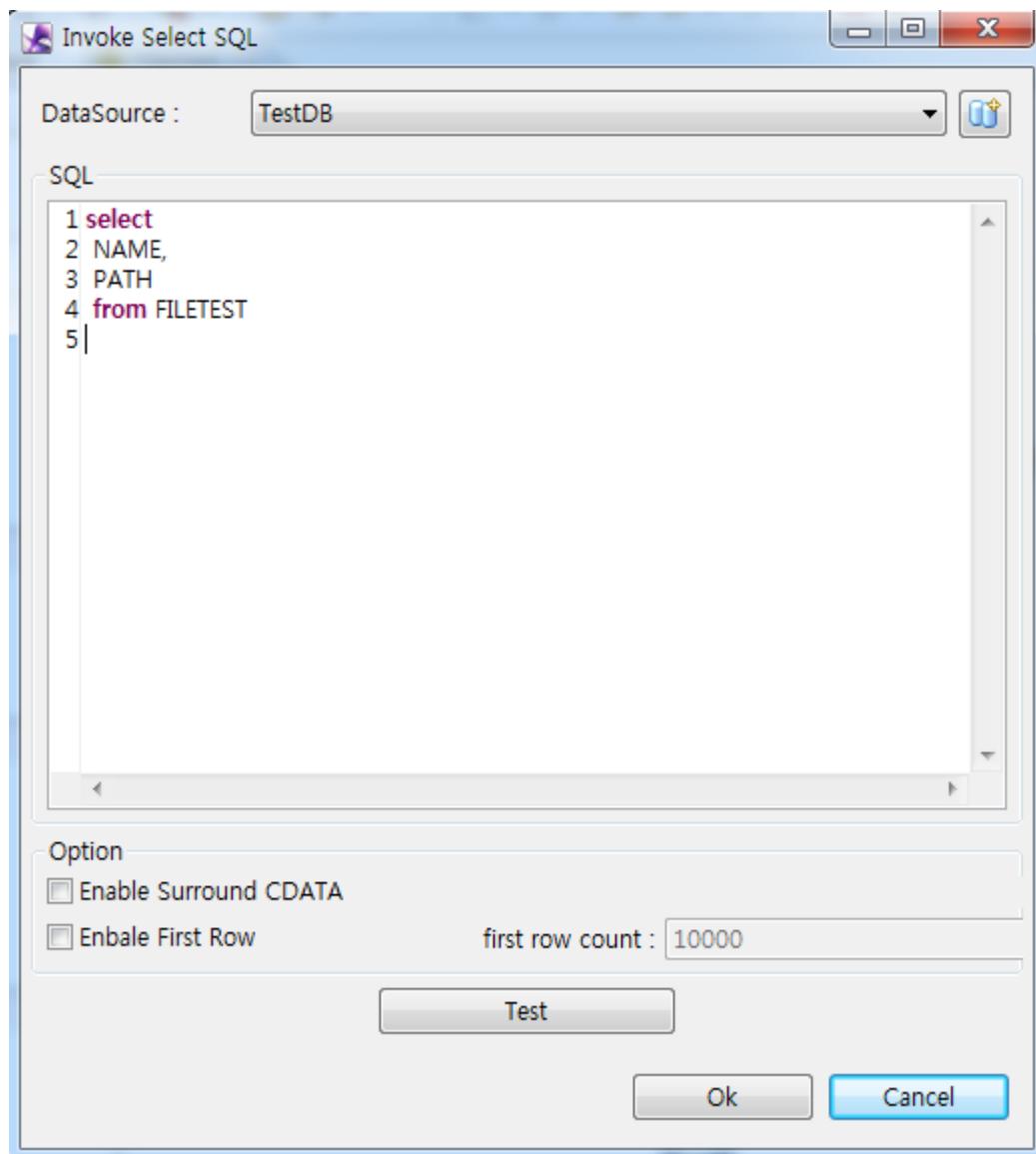
3. Modify Invoke를 더블클릭하고 insert 쿼리에 아래와 같이 입력합니다.

insert into FILETEST (NAME, PATH) values (#dataset1.NAME#, #dataset1.PATH#)

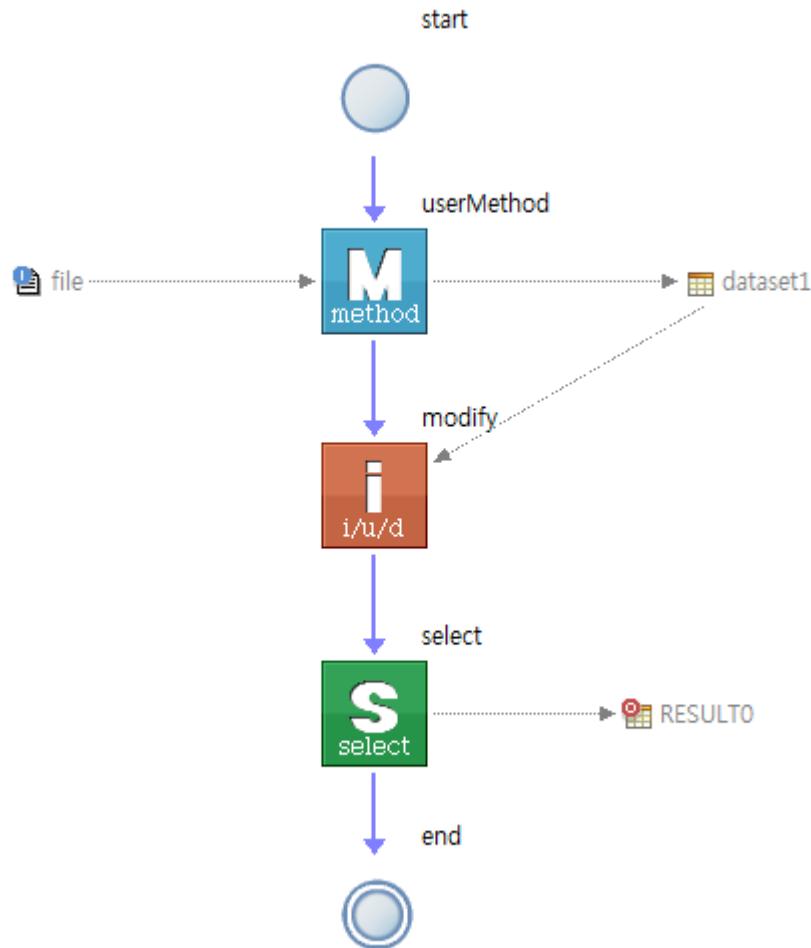


4. Test버튼을 클릭 합니다.
5. Test 후에 자동 생성된 RESULT0 output 파라메테는 삭제합니다.
6. Select Invoke를 추가하고 Invoke Select SQL 창에 아래와 같이 쿼리를 입력합니다.

```
select
    NAME,
    PATH
from FILETEST
```

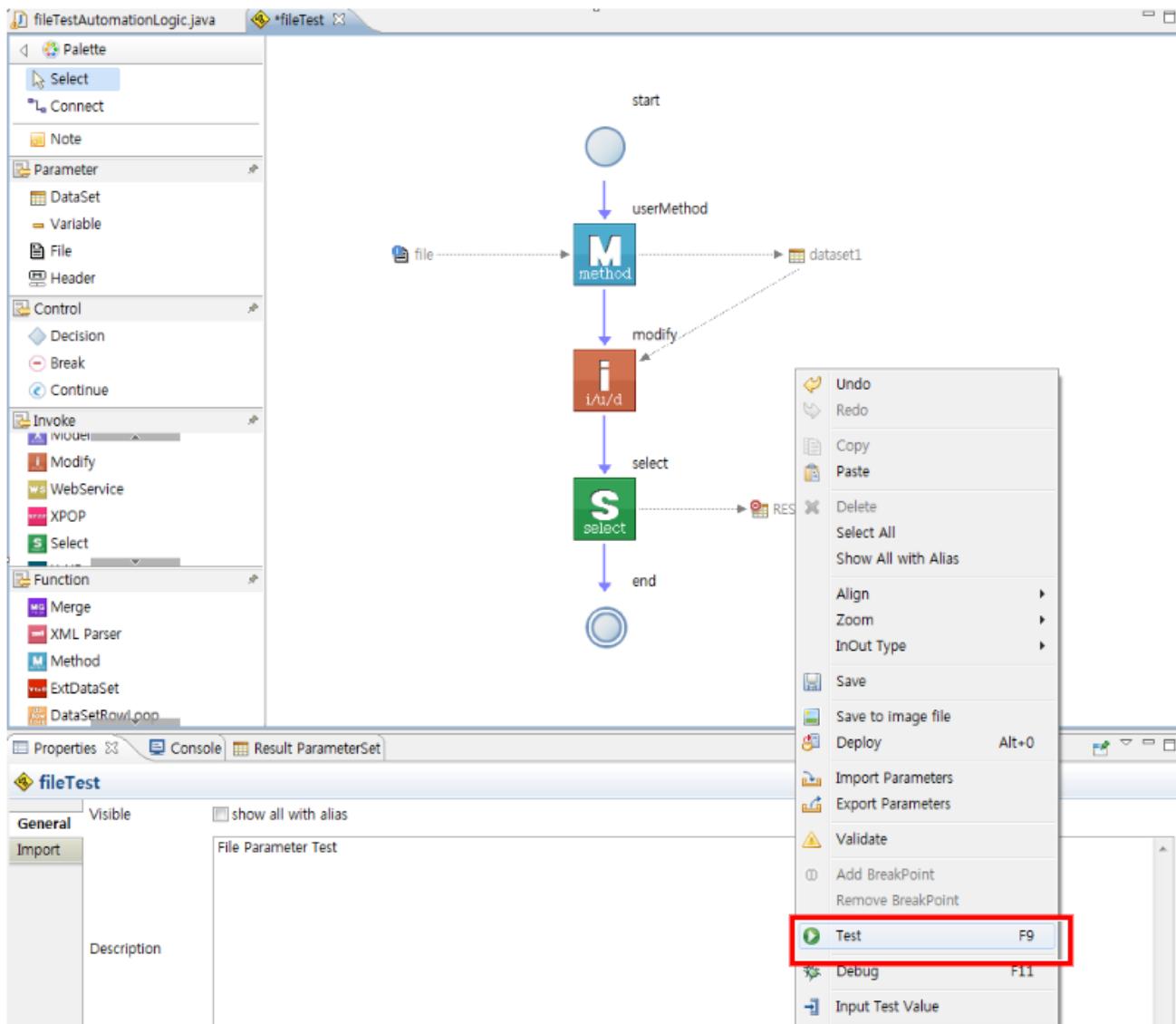


7. Test 버튼을 클릭하고 Ok 버튼을 클릭하여 Invoke Select SQL창을 종료합니다.
8. Start 노드와 End 노드까지 위에서 생성한 모든 컴포넌트를 Connect 합니다.



모델 테스트하기

1. 모델 에디터에서 마우스를 오른쪽 클릭하면 팝업 메뉴가 나타납니다. 팝업 메뉴에서 Test를 선택하면 모델을 테스트 할 수 있습니다.



2. 테스트가 끝나면 모델의 출력을 Result ParameterSet 뷰에서 보여줍니다.

The screenshot shows the 'Result ParameterSet' view. It contains two tables: 'Parameters' and 'DataSets'. The 'Parameters' table has columns 'Parameter Name' and 'Parameter Value', but is currently empty. The 'DataSets' table has columns 'NAME' and 'PATH', and contains the following data:

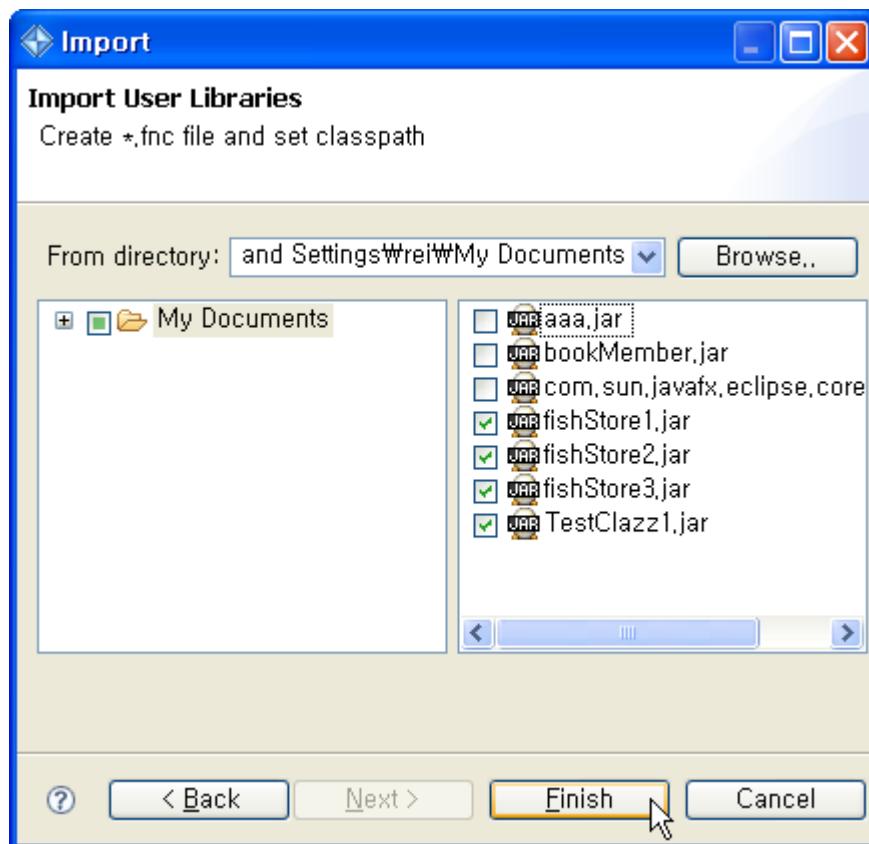
NAME	PATH
HttpRequest.xml	C:\#Users\gan...
RESULTO	

13.4 User Library Handling

X-UP 모델 생성 시 에디터에서 제공하는 X-UP 함수 외에 사용자가 라이브러리를 추가하여 사용자 지정 함수를 사용할 수 있습니다.

User Library는 Import User Library Dialog에서 X-UP Model을 개발할 때 사용할 수 있도록 프로젝트에 등록해줍니다.

Import User Library Dialog는 [Select project > Click right mouse > import > X-UP > User Libraries] 를 통해 실행할 수 있습니다.

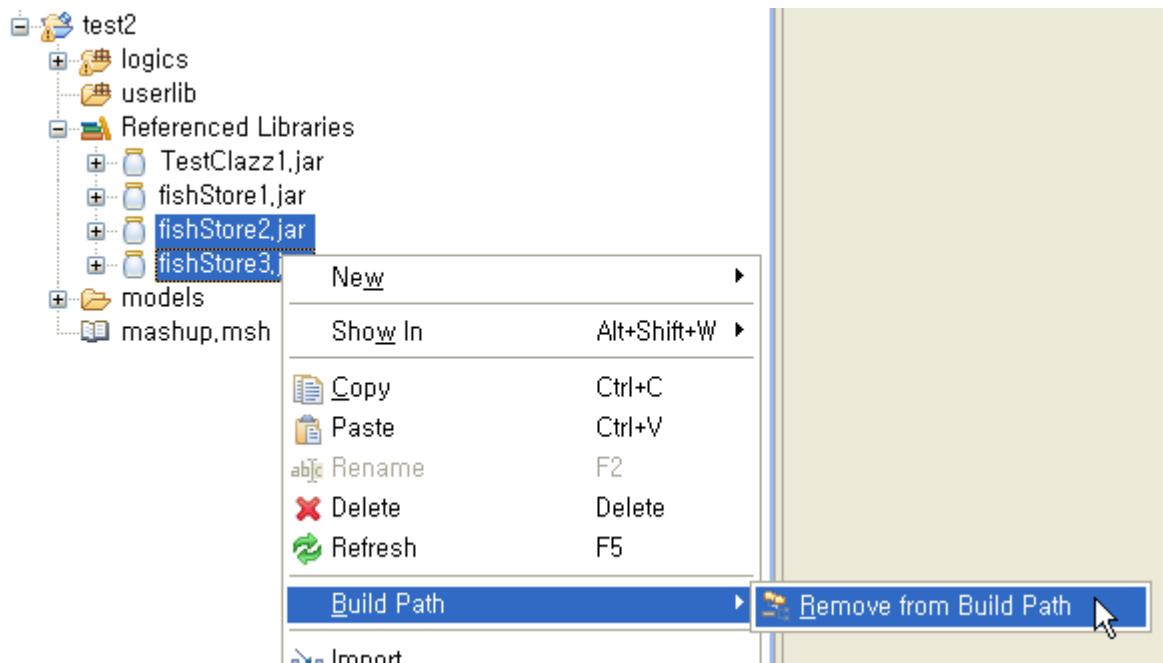


유저 라이브러리를 선택하고 Finish를 하면 프로젝트에 userlib 폴더가 없다면 자동 생성되고, 선택한 jar파일을 userlib폴더에 복사하며, 자동으로 선택한 jar 파일이 클래스패스에 등록이 됩니다.



유저 라이브러리를 Build Path에서 제외하는 과정은 다음과 같습니다.

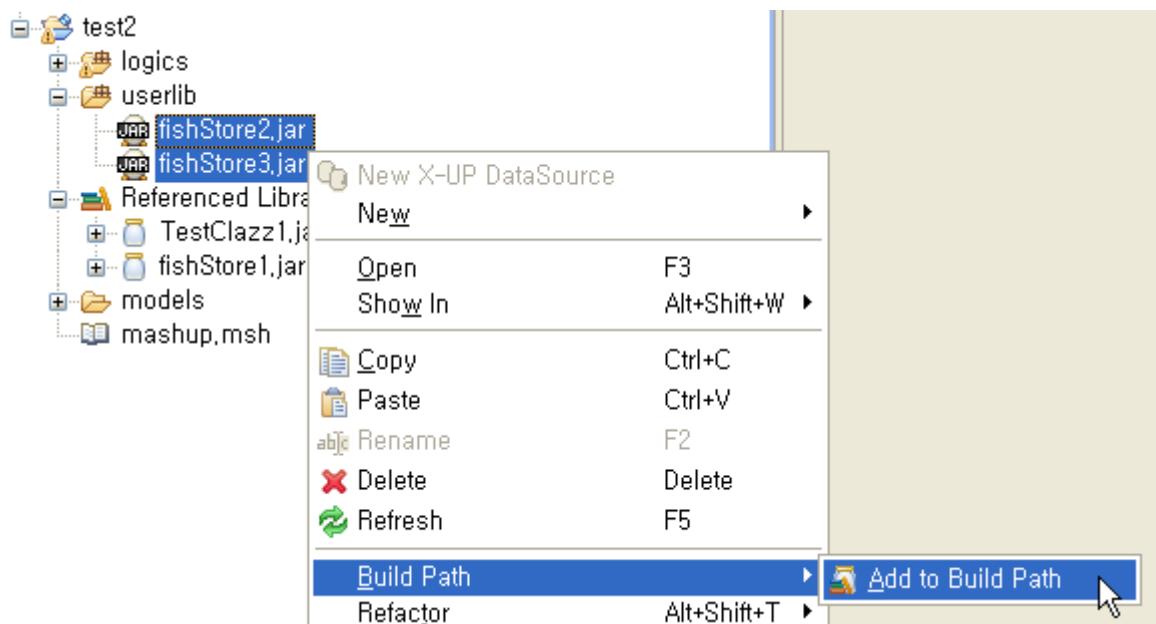
삭제하고자 하는 라이브러리를 선택한 후 [Build Path > Remove from Build Path]를 선택합니다.



Remove from Build Path를 선택하면 해당 파일은 빌드패스에서 제외되고 userlib 폴더밑에 파일이 보이게 됩니다.
userlib폴더밑에 보이는 파일은 영구히 삭제하거나 나중에 다시 빌드패스에 추가할 수 있습니다.

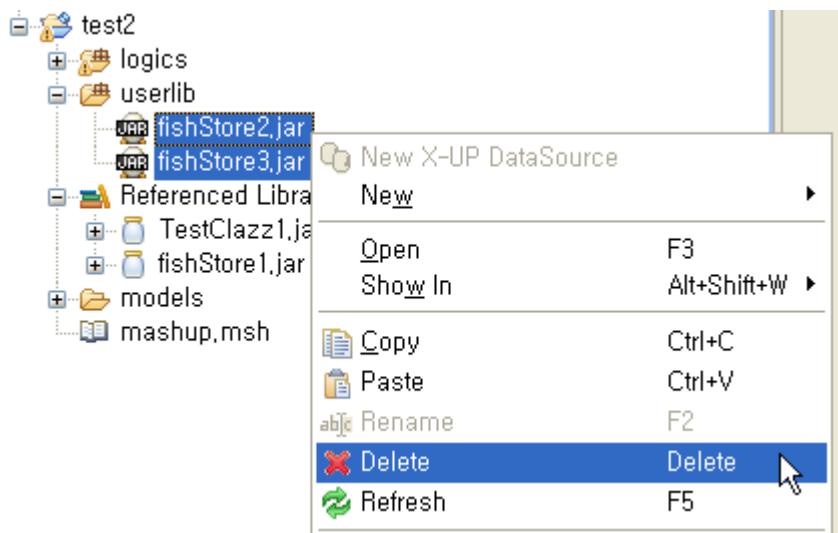
Build Path에서 제외된 유저라이브러리를 다시 Build Path에 추가하는 과정은 다음과 같습니다.

추가하고자 하는 라이브러리를 선택한 후 [Build Path > Add to Build Path]를 선택합니다



Build Path에서 제외된 유저라이브러리를 프로젝트에서 삭제하는 과정은 다음과 같습니다.

삭제하고자 하는 라이브러리를 선택한 후 Delete를 선택합니다



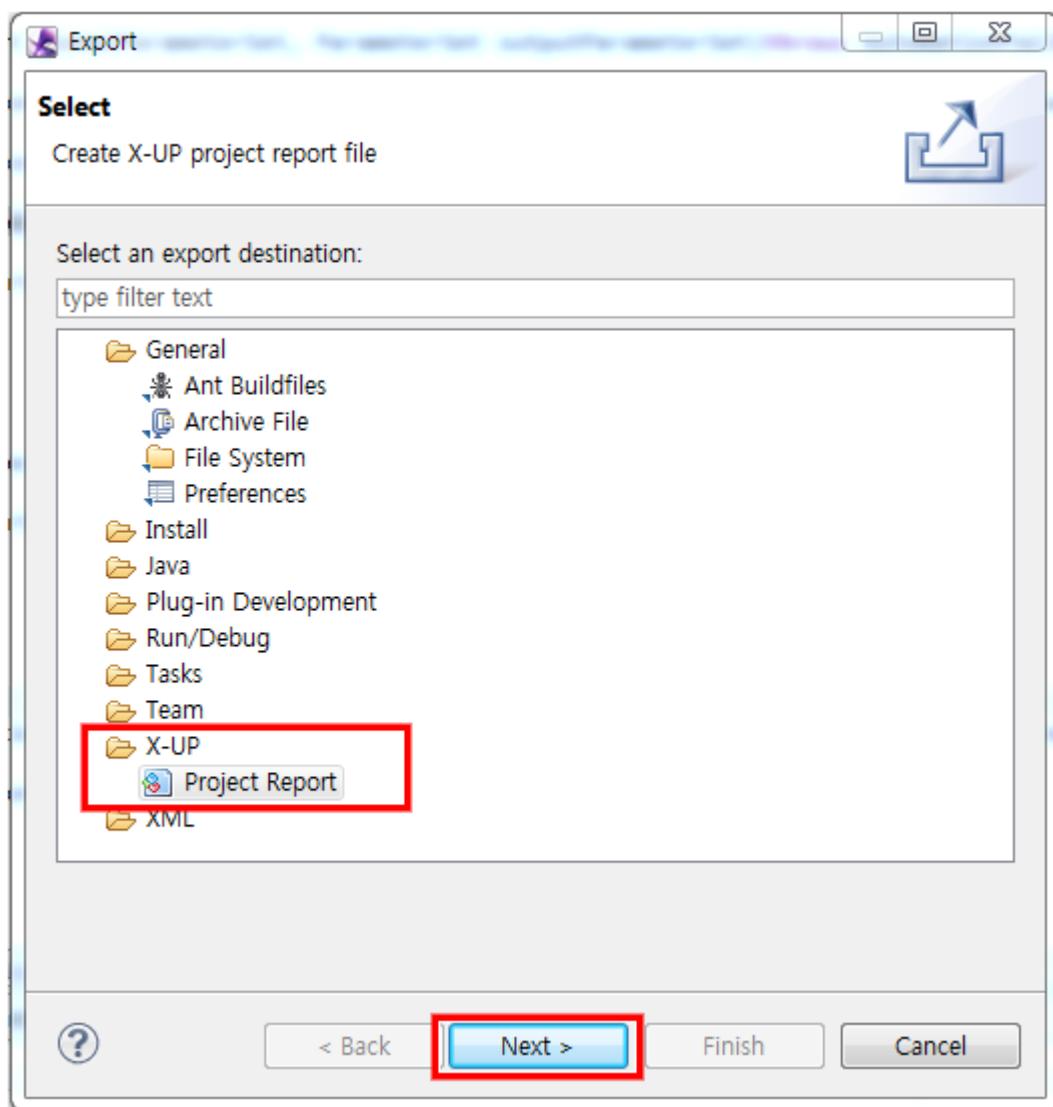
13.5 X-UP Server 관리

실행과 중지

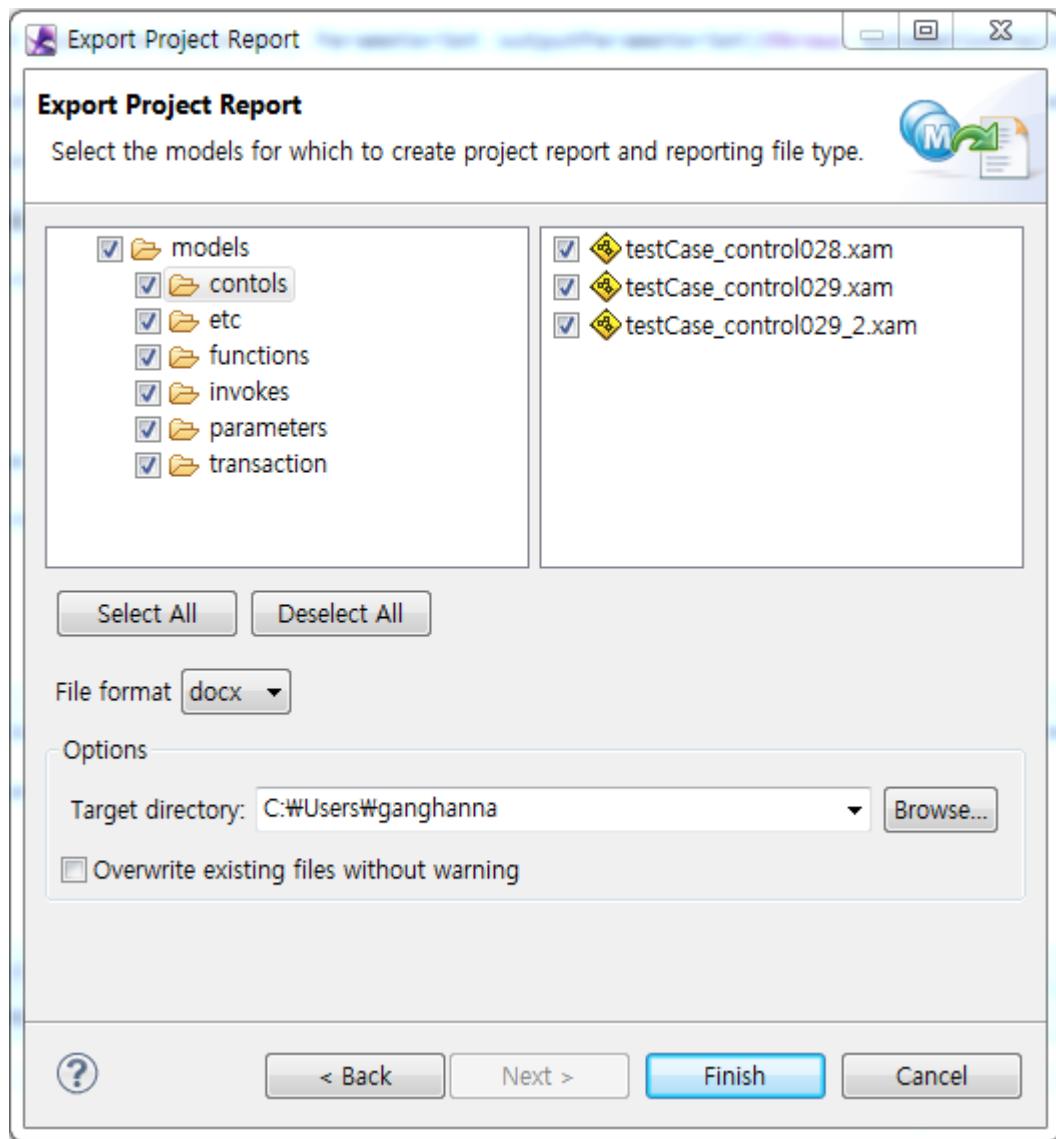
X-UP은 WAS(Web Application Server)위에서 웹 애플리케이션으로 동작합니다. 그렇기 때문에 서버의 구동과 중지는 X-UP이 설치된 WAS의 구동방법에 의존합니다.

13.6 Reporting

X-UP Builder에서는 X-UP Builder로 개발한 X-UP Model들의 자세한 사항들을 문서로 만들어 한 눈에 볼 수 있는 기능인 Export Project Report Dialog를 제공합니다. . X-UP Project Reporting은 [Select Project > Click right mouse > Export > X-UP > Project Report] 를 통해 실행할 수 있습니다.



Next 버튼을 누르면 아래와 같이 진행됩니다.



	Name	Description
①	Project Select List	프로젝트 선택 리스트
②	Model Select List	모델 선택 리스트
③	File Format	파일 형식 선택 (docx / xlsx / html / text)
④	Target Directory	Project Reporting File 저장 디렉토리 경로
⑤	Browse	Project Reporting File 저장 디렉토리 경로 편집
⑥	Finish	Project Reporting File 생성

Project Reporting을 원하는 모델을 선택한 뒤 원하는 File Format을 선택합니다. File Format은 MS Word(docx), MS Excel(xlsx), HTML(html), Text(txt) 총 4개의 형식을 선택할 수 있습니다. Target Directory는 Project Reporting 파일이 만들어질 디렉토리를 정해줍니다. 그리고 Finish 버튼을 누르게 되면 원하는 형식의 파일이 만들어지게 됩니다.

13.7 Filtering

X-UP에서 한 Row에 대한 action(insert, update, delete) 을 처리하기 전 특정 Rule에 맞는 Row일 경우 해당 Row는 실행을 하지 않고 Filtering 됩니다.

다음은 Filtering에 대한 예입니다.

The screenshot shows the SAP BusinessObjects Designer environment. On the left is the Palette, which includes sections for Select, Parameter, Control, Invoke, and Function. The Invoke section has a tab for X-UP selected, showing icons for Web..., Modify, XPOP, Select, and X-UP. The main area displays a process flow starting with a 'Start' node, followed by a connector labeled 'EMPLOYEES', then a rectangular activity node labeled '1/1/d' with a green 'modifyDs' input and a red 'RESULTO' output. The flow ends with an 'End' node. Below the palette is the Properties panel, which is open for the 'EMPLOYEES' object. The 'Filtering' tab is selected and highlighted with a red box. The 'Filtering Rule' table contains one row:

Rule Name	Parameter	Comparator	Value
idValidation	modifyDs.EMPLOYEEID	empty	not empty empty = != > => <= < between

Name	Description
Rule Name	Rule을 구분하기 위한 값으로 다른 Rule과 중복되지 않는 값을 설정합니다.
Parameter	Rule에서 사용할 입력 파라미터를 설정합니다.
Comparator	입력 파라미터의 조건을 설정합니다.
Value	Comparator에서 설정한 값이 “not empty, empty, custom”이 아닌 경우 조건 상수를 입력합니다.



Filtering은 Modify Invoke 시에만 적용되는 사항입니다. 자세한 예제는 [8.3 Modify Invoke를 이용한 모델 개발하기](#)를 참조하세요.

13.8 다국어 처리

X-UP은 다음과 같은 원칙으로 다국어를 처리합니다.

- 클라이언트 요청에 포함된 문자셋으로 데이터셋에 요청하고 받은 데이터를 같은 문자셋으로 클라이언트에 전달
- 만약 클라이언트 요청의 문자셋과 데이터소스에서 수집한 데이터의 문자셋이 다를 경우 데이터의 캐릭터셋으로 클라이언트에 전달

예를 들어 클라이언트의 요청에 담긴 문자셋이 ‘euc_kr’일 경우 데이터 수집 시에 문자셋을 ‘euc_kr’”를 사용하여 데이터를 가져옵니다. 그리고 수집하고 가공된 데이터를 ‘euc_kr’로 클라이언트에 응답합니다. 하지만 만약 데이터소스에서 가져온 데이터가 ‘UTF-8’이라면 이를 ‘euc_kr’로 다시 인코딩하지 않고 그대로 ‘UTF-8’로 클라이언트에 전달합니다.

다음은 클라이언트의 요청에서 문자셋을 결정하는 로직입니다.

- 데이터 자체에 명시된 문자셋을 구한다.(xml과 같은 경우)
- 없으면, HttpRequest에 의한 요청일 경우 request에서 문자셋을 구한다.
 - 헤더 ‘CONTENT-TYPE”에 담긴 문자셋을 구한다.
 - 없으면, 헤더 ‘ACCEPT-CHARSET’에 담긴 문자셋을 구한다.
 - 없으면, 헤더 ‘ACCEPT-LANUAGE’에 담긴 언어를 가지고 문자셋을 구한다.
- 없으면, 모델에 설정된 문자셋으로 한다.
 - 모델 metadata에 설정된 문자셋을 구한다.
 - 없으면, 모델이 사용하는 datasource에 설정된 문자셋을 구한다.
- 없으면, ‘UTF-8’을 문자셋으로 한다.

13.9 Junit & Remote class 생성

X-UP은 사용자에게 디버깅의 편리성을 제공하기 위한 기능으로 JUnit 클래스와 Remote 클래스를 자동 생성해주는 기능을 제공하고 있습니다.

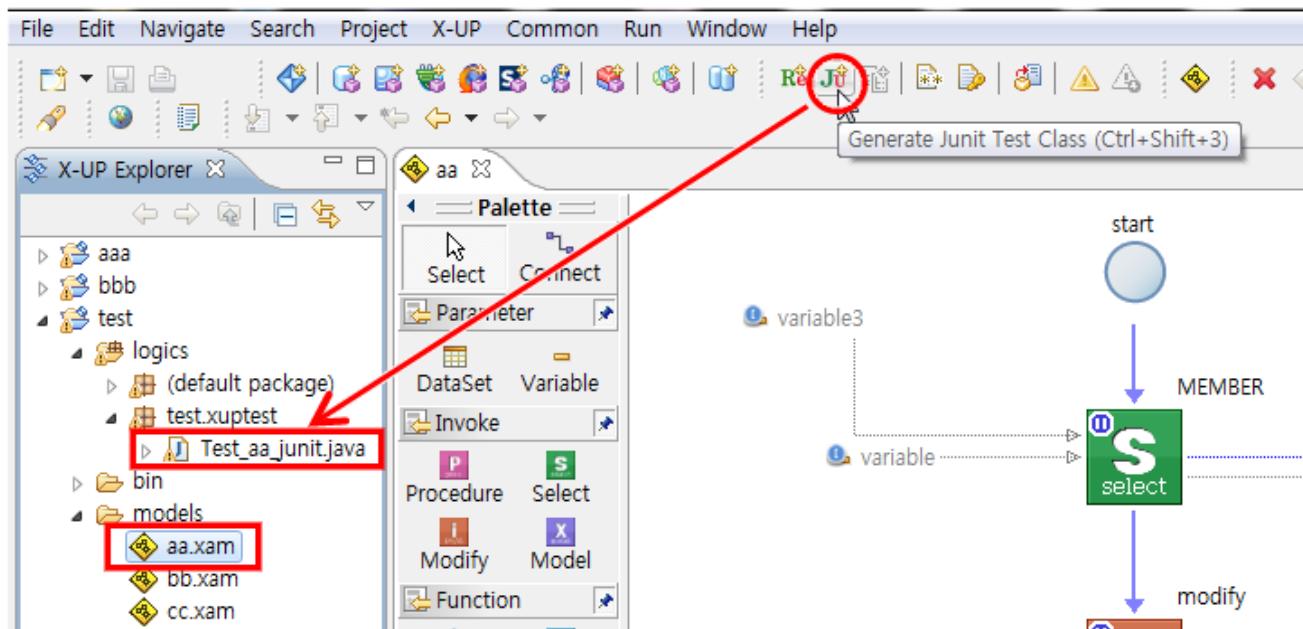
사용자가 테스트할 모델을 선택하고 아래 메뉴를 클릭하게 되면 간단한 JUnit 클래스와 Remote 클래스를 생성합니다.

다.

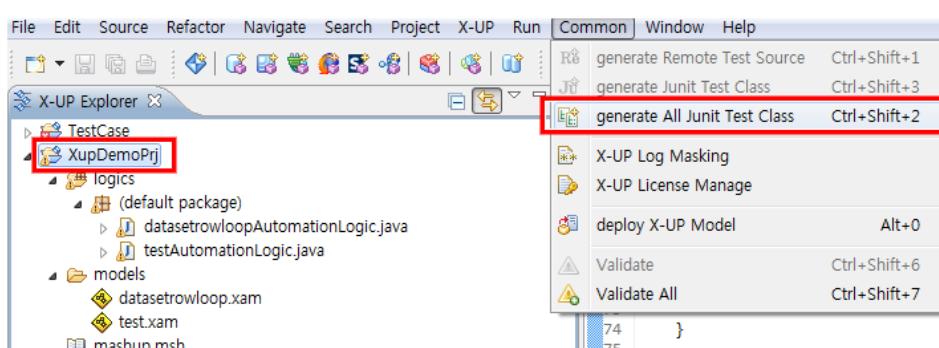
Name	Description
generate Remote Test Source	선택한 모델을 기반하여 Remote 테스트를 할 수 있는 테스트 클래스를 생성
generate Junit Test Class	선택한 모델을 테스트할 수 있는 간단한 JUnit Test Case를 생성
generate All Junit Test Class	선택한 프로젝트에 존재하는 모든 모델을 대상으로 JUnit Test Case를 생성

JUnit Test Class 생성 방법

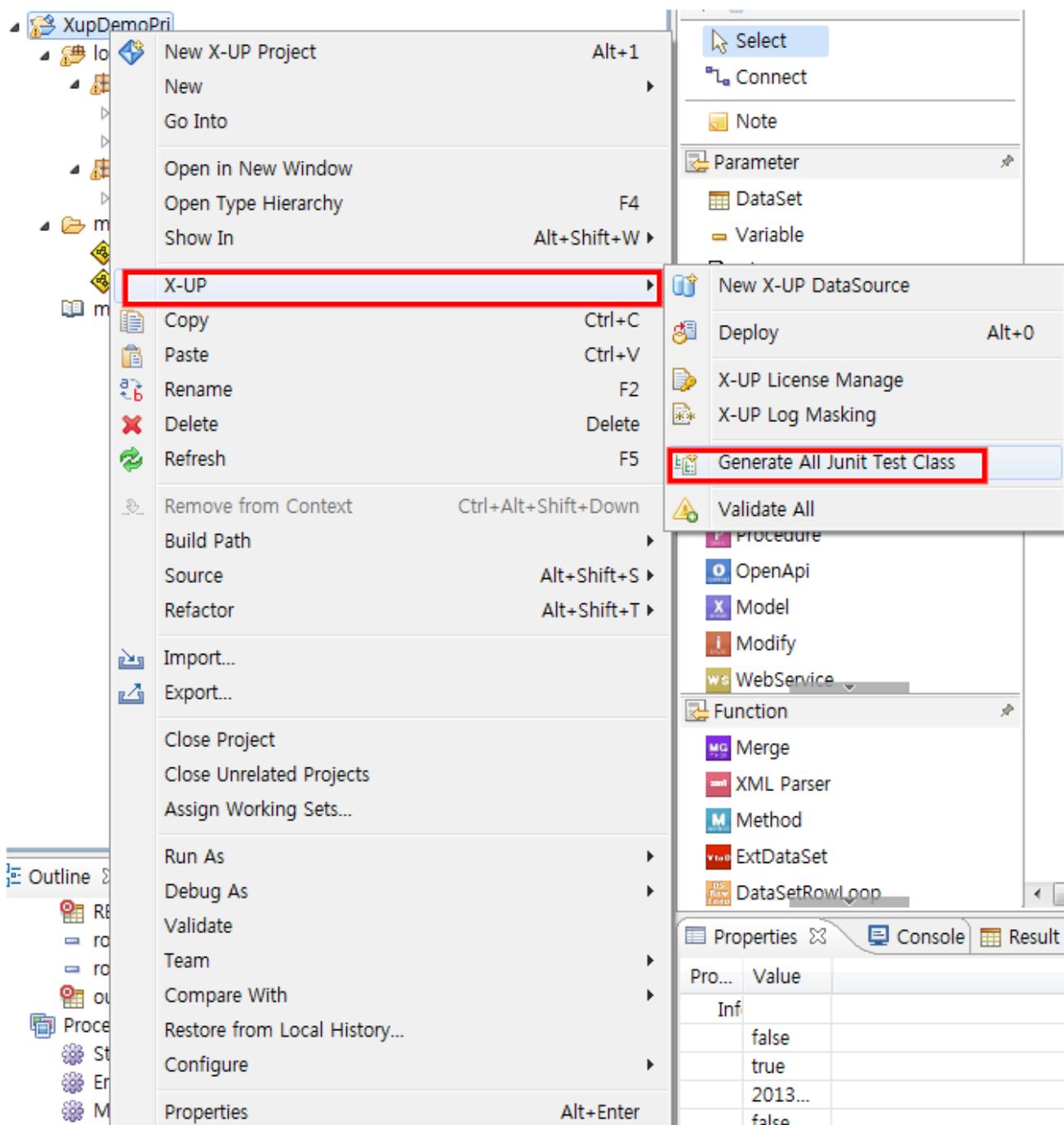
- 테스트할 모델을 선택합니다.
- 개발도구 Bar에 [JUnit] 아이콘을 선택합니다.
- logics 폴더에 본래 클래스이름 뒤에 '_junit'가 덧붙여진 JUnit Test 클래스가 생성되었음을 확인합니다.



- 또 다른 방법으로는 Menu Bar에서 [Common > generate Junit Test Class]를 선택한다.
- 프로젝트내에 있는 모든 모델에 대해 JUnit Test 클래스를 생성하기 위해서는 프로젝트를 선택한 후에 [Common > generate All Junit Test Class]를 선택한다.

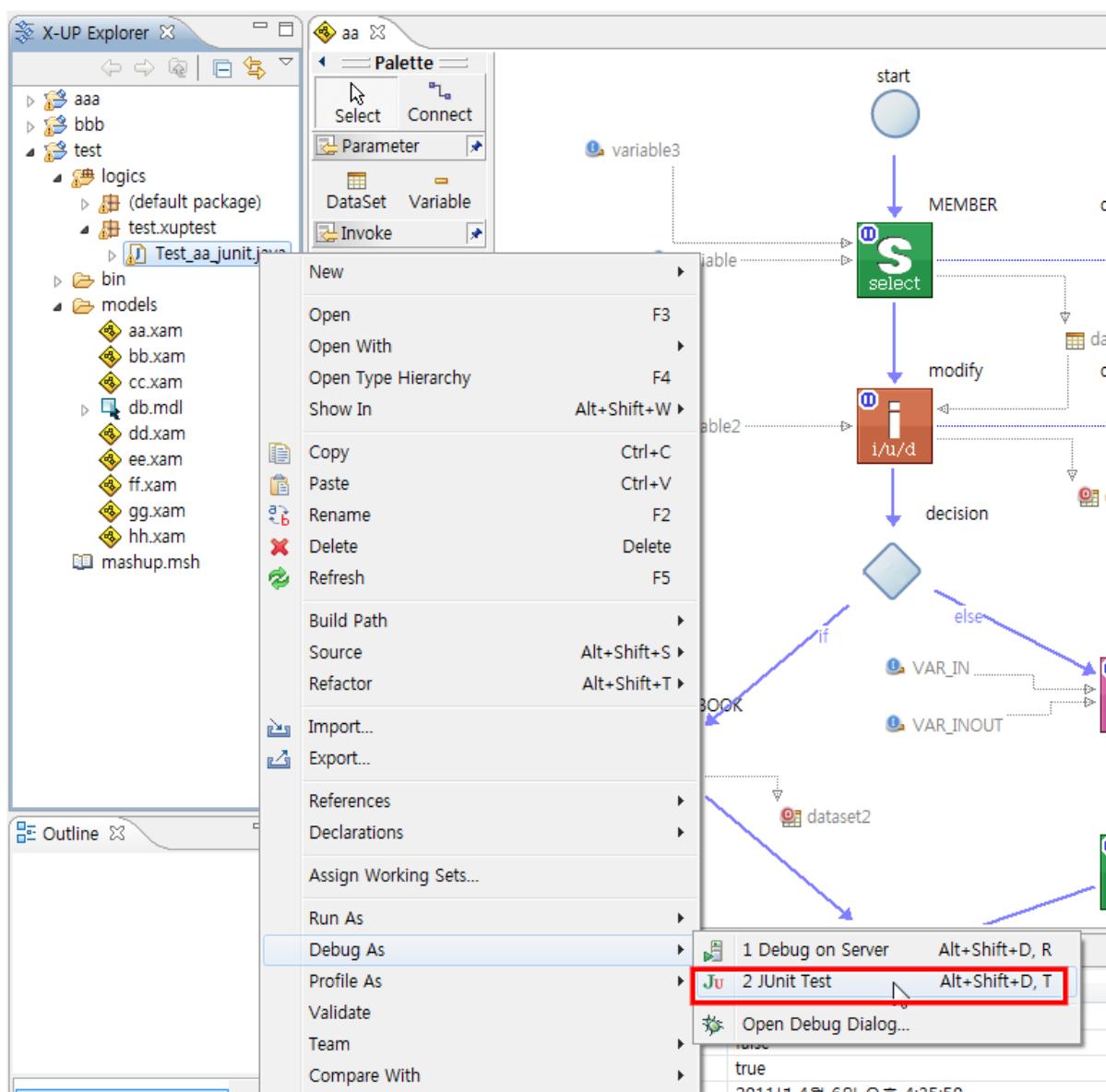
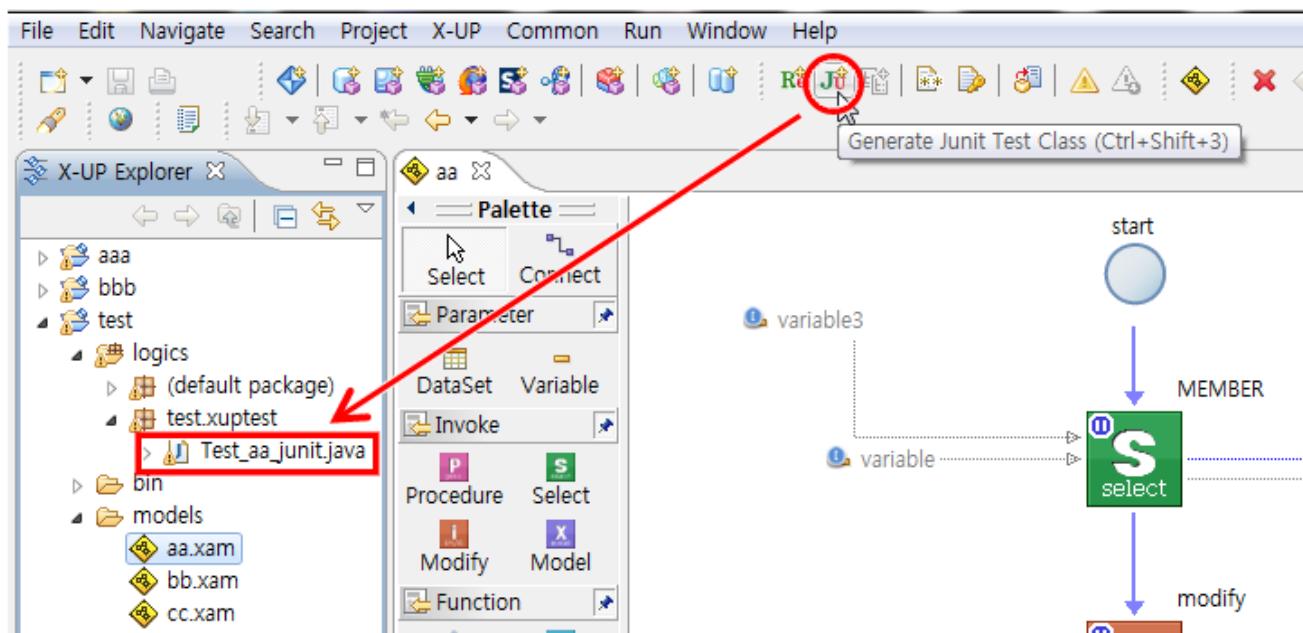


6. 또 다른 방법으로는 프로젝트를 선택한 후에 [Click right mouse > X-UP > Generate All Junit Test Class]



JUnit Test Class를 생성한 후 디버그 모드로 실행

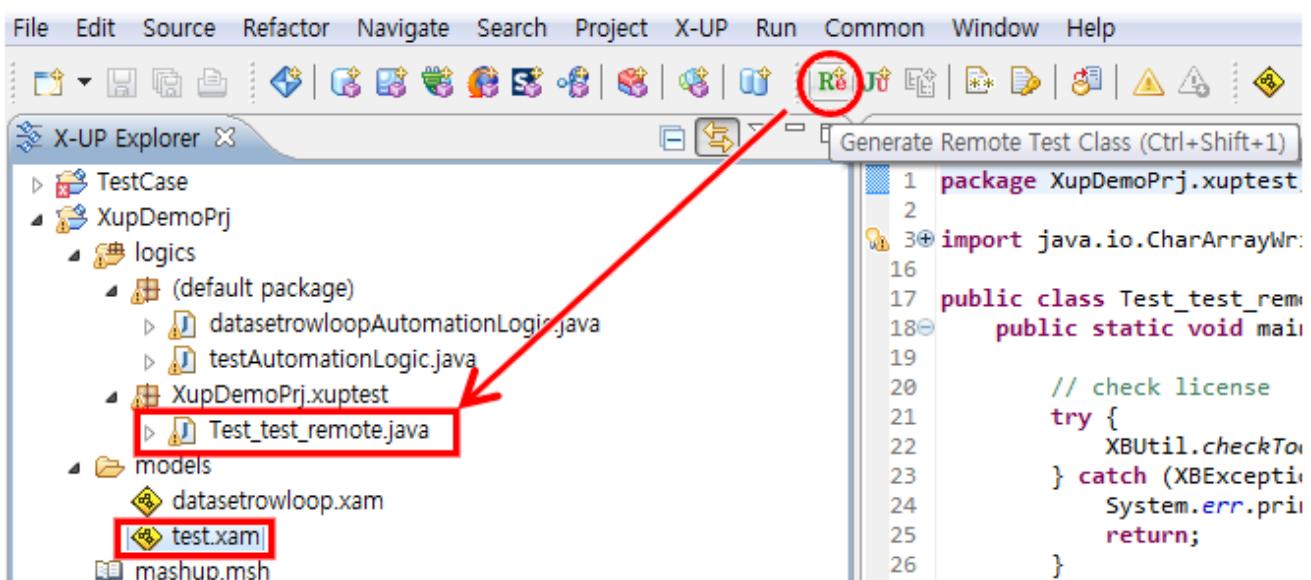
BreakPoint를 추가한 후 모델 디버깅을 하기 위해선 해당 모델의 Junit Test Case 자동 생성 메뉴를 통해 테스트케이스 자바파일을 생성한 후 자바 디버그 모드로 실행합니다.



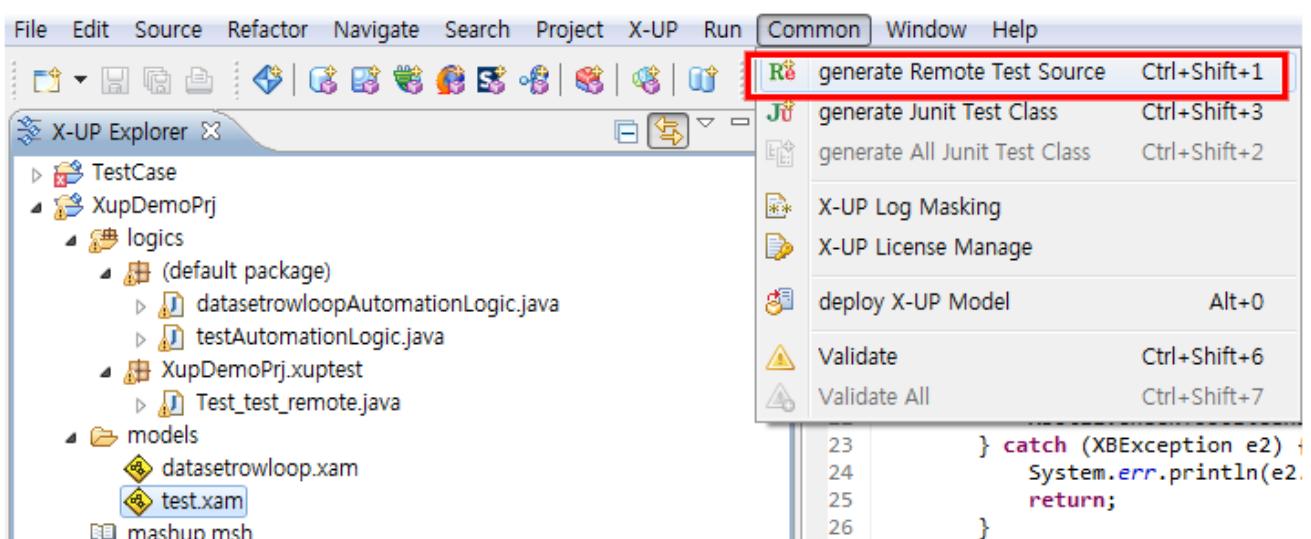
이후는 일반 자바 디버그와 동일합니다.

Remote Test Souce 생성 방법

1. 테스트할 모델을 선택합니다.
2. 개발도구 Bar에 [] 아이콘을 선택합니다.
3. logics 폴더에 본래 클래스이름 뒤에 '_reomte'가 덧붙여진 Remote Test Source 클래스가 생성되었음을 확인합니다.



4. 또 다른 방법으로는 Menu Bar에서 Common generate Remote Test Source를 선택합니다.



14.

Exception 처리

X-UP에서는 하나의 Automation 모델에 대해 Global Transaction이 적용되기 때문에 개별적인 트랜잭션을 사용할 수 없습니다. 하지만 Invoke 별 Exception을 처리 할 수 있는 property 가 있습니다. 해당 property 에서 처리 할 수 있는 예외는 InvokeSkip, RowSkip 두 가지를 제공 합니다.

- InvokeSkip : 하나의 Invoke에서 예외가 발생했을 경우 InvokeSkip을 하게 되면 해당 Invoke의 예외를 무시하여 다음 Invoke 를 처리하게 됩니다.
- RowSkip : Modify Invoke에 적용되는 사항이며 Row단위 처리 시 특정 Row를 처리 하는 중 예외가 발생했을 경우 해당 Row에 대한 예외를 무시하고 다음 Row를 처리하게 됩니다.

다음은 RowSkip에 대한 예입니다.

```
Properties <--> Console <--> Result ParameterSet  
ADDRESS_BOOK1  
General  
InvokeInfo  
TypeBinding  
Filtering  
Default Value  
Exception  
public void onExceptionOccured(ParameterSet globalParameterSet, InvokingInfo invokingInfo, Throwable e, Invok  
try {  
    InvokingModifyErrorInfo modifyErrorInfo = (InvokingModifyErrorInfo) errorInfo;  
    int type = modifyErrorInfo.getExecuteType();  
    if(type == InvokingModifyErrorInfo.DELETE) {  
        return;  
    }  
    int erridx = modifyErrorInfo.getRowIndex();  
    DataSet addrs = globalParameterSet.getDataSet("addressDsModify");  
    if("TOBESOFT".equals(addrs.getString(erridx, "COMPANY"))){  
        log("DEBUG", "modifyAddressBooks model was ignored "+erridx+" index.");  
        System.out.println("test");  
        throw new Rowskip("modifyAddressBooks model was ignored "+erridx+" index.");  
    }  
    throw new Exception("invoke failed. e=" + e+ ", message=" + e.getMessage());  
} catch(Rowskip x) {  
    throw x;  
} catch(Exception x) {  
    throw new AutomationFailException("select invoke failed. e=" + e+ ", message=" + e.getMessage());  
}  
}
```

위의 그림 중 onExceptionOccured(ParameterSet globalParameterSet, Throwable e, InvokingErrorInfo errInfo) 메서드는 Invoke 실행 중 예외가 발생 했을 경우 호출되는 메서드 입니다.

해당 메서드의 매개변수(parameter)에 대한 설명입니다.

- globalParameterSet : Invoke 시 필요한 parameter와 데이터소스로부터 획득한 모든 parameter들이 있는 ParameterSet입니다.

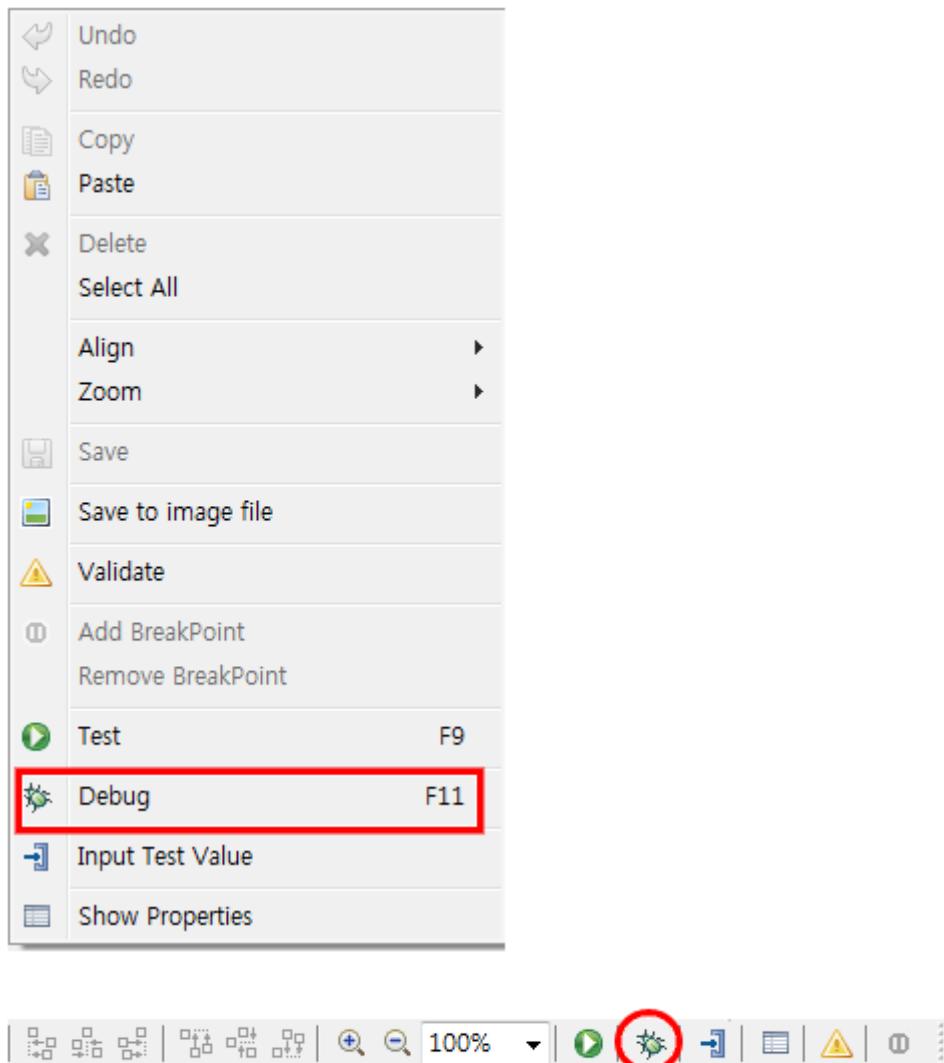
- e : 실제 로직에서 발생한 예외입니다.
- errInfo : 발생한 예외에 대한 정보를 담고 있는 객체입니다.

15.

Debugging

X-UP Builder에서의 Debug 메뉴를 통한 빠른 디버깅 :

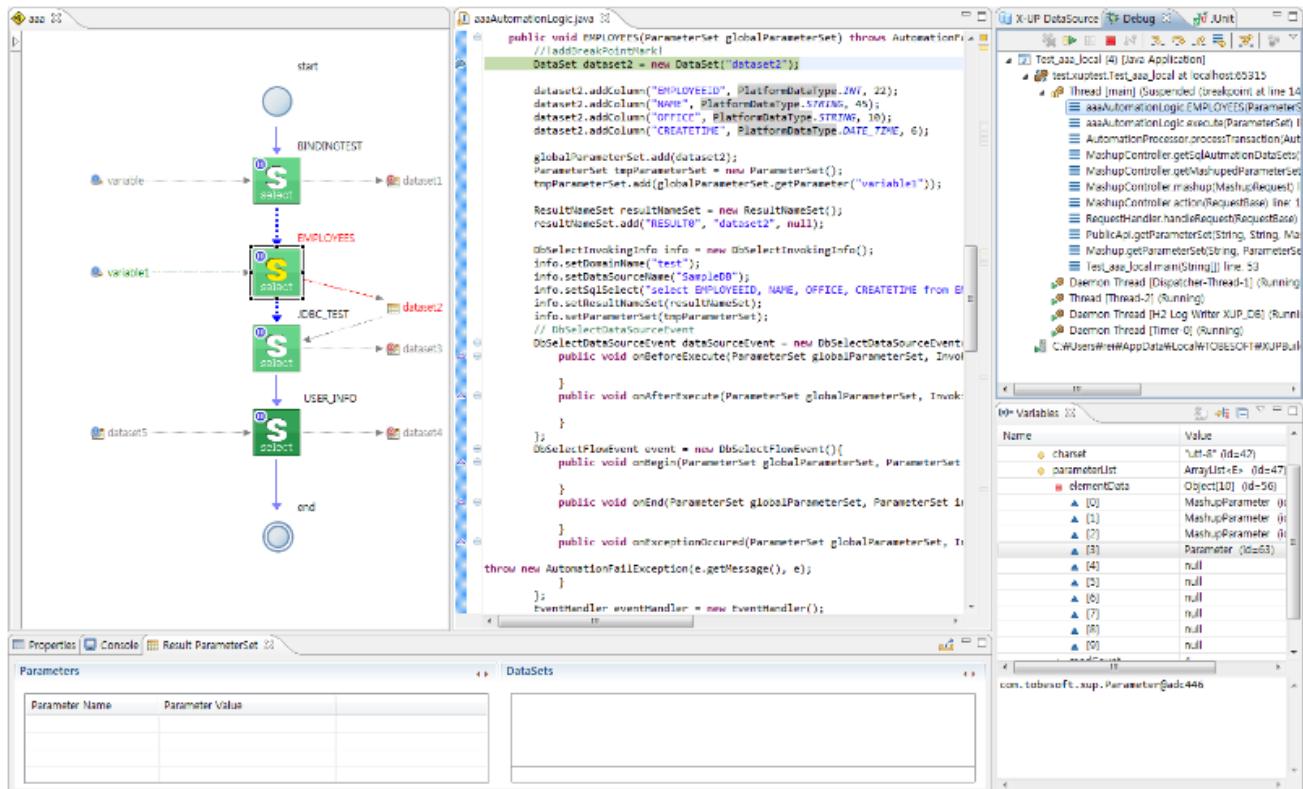
에디터에서 모델 개발을 한 후 디버그 시 멈추고 싶은 노드를 선택하여 break point를 등록합니다. 그런 다음 에디터 메뉴에 Debug를 실행하여 디버깅합니다.



디버그 메뉴는 에디터 툴바와 오른쪽 마우스 메뉴에 존재하며 디버그 실행시 자바 디버그 모드로 자동 실행됩니다. 디버그 진행 시 에디터화면과 자바 소스파일을 보여주며 Resume이나 기타 디버그 이벤트 진행시마다 자바 메소드와

에디터 노드간의 동기화가 이루어집니다.

에디터 화면과 자바 파일을 같이 볼 수 있도록 하면 현재 자바 소스 코드상의 메소드가 실제 에디터의 어느 노드인지 한눈에 확인할 수 있습니다.



16.

오류 및 Error확인 및 대응

X-UP서비스로 구축된 시스템을 관리 하다 보면 X-UP 서비스 호출 시 오류가 나는 경우가 있습니다. 이는 개발한 X-UP모델 자체 오류 또는 시스템의 환경적 오류로 구분되어 질 수 있으며, 이를 보다 빨리 파악하기 위해 xup.log 내용을 확인합니다.

xup.log는 사용자가 설정한 loglevel에 의해 로그가 남겨집니다.

17.

배포 안내

17.1 정책 및 일정 안내

고객의 기능 요구사항에 보다 효과적으로 대응하고, 신뢰성을 향상시키기 위해 배포 프로세스에 따라 X-UP 신규 개발 및 유지 보수를 진행하고 있습니다.

배포 일정

신규개발 및 유지보수 처리에 대한 월 일정은 다음과 같습니다. 배포일자는 매월 마지막주 금요일입니다.

월	화	수	목	금
	<ul style="list-style-type: none">• 요구사항 접수• 개발 및 수정• Engine, Builder 단위 테스트• 서버팀 내부 회의 및 동료 검토	<ul style="list-style-type: none">• 테스트 케이스 검토• 익월 개발 항목 리스트 서버팀 내부회의	<ul style="list-style-type: none">• 당월 배포 기능 리스트 검토(SAP 사업부, 서버팀)	<ul style="list-style-type: none">• 배포 (매 월 마지막주 금요일)

릴리즈 시작

→ 수정

배포 종류

배포 종류는 크게 정기배포와 비 정기배포가 있으며, 차이점은 다음과 같습니다.

구분	정기배포	비 정기배포
정의	X-UP을 정기적으로 정해진 날짜에 배포 (매월 마지막 주 금요일)	고객의 요구에 따라 배포
배포 기능 리스트	배포 기능 리스트 도출 순서도에 따라 확정	협의를 통해 확정
테스트	배포 기능 리스트에 대한 요구사항 처리 확인 X-UP Engine 단위 테스트 (JUnit) X-UP Builder 단위 테스트 Builder 통합 테스트 Stress 테스트 Porting(배포/설치 포함) 테스트	배포 기능 리스트에 대한 요구사항 처리 확인 X-UP Engine 단위 테스트 (JUnit) Builder 통합 테스트
History	X-UP 팀에서 작업 후 배포	X-UP 팀에서 작업 후 배포
매뉴얼	X-UP 팀에서 작업 후 배포	배포하지 않고 억월 정기 배포에 포함함

17.2 배포 사이트 안내

부록 A.

Dynamic Sql

iBatis는 SQL에 기반한 데이터베이스와 자바 등을 연결시켜 주는 역할을 하는 영속성 프레임워크입니다. 프로그램의 소스코드에서 SQL문장을 분리하여 별도의 XML파일로 저장하고 이 둘을 서로 연결시켜주는 방식으로 작동합니다.

실무에서 SQL문을 작성하다 보면 동적 쿼리문 작성은 작성해야 할 경우가 많이 생깁니다. 이럴 경우 제어문을 반복해서 사용하다보면 소스 코드의 가독성을 떨어트립니다. 이 때 iBatis의 동적 쿼리문을 사용함으로써 소스를 깔끔하게 구현할 수 있습니다.

아래는 몇가지 동적 쿼리문의 사용방법과 예제입니다.

변수는 #변수#를 이용하여 쿼리를 작성합니다.

X-UP Builder에서 Automation 모델인 Select Invoke와 본 매뉴얼에서 제공하는 [부록 B. 예제 DB 테이블 생성 스크립트](#)를 이용하여 아래 Example 쿼리를 테스트할 수 있습니다.

1. 동적 column

컬럼을 동적으로 할당할 경우에 사용

```
select
    empno,
    empname,
    job
<dynamic>
    <isEqual prepend="," property="inParam" compareValue="sal">
        sal
    </isEqual>
    <isEqual prepend="," property="inParam" compareValue="hiredate">
        to_char(hiredate,'YYYY-MM-DD') as hiredate
    </isEqual>
</dynamic>
from emp
```

2. 동적 where 조건절

프로퍼티에 따라 동적으로 where 조건절을 사용

```

select empno, ename, job
from emp
where 1=1
<dynamic>
    <isEmpty prepend="AND" property="inParam">
        comm is null
    </isEmpty>
    <isNotNull prepend="AND" property ="inParam">
        comm = #inParam#
    </isNotNull>
</dynamic>

```

3. <isEqual>

프로퍼티 속성값이 compare프로퍼티 값이나 compare 값과 같은지 검사

```

select empno, ename, job
from emp
where 1=1
<dynamic>
    <isEqual property="inParam" compareValue="WARD" prepend="AND">
        ename=#inParam#
    </isEqual>
</dynamic>

```

4. <isGreaterThan>

프로퍼티 속성값이 compare프로퍼티 값이나 compare값보다 큰지 검사

```

select empno, ename, job
from emp
where 1=1
<dynamic>
    <isGreaterThan prepend="AND" property="inParam" compareValue="2000">
        sal  >= #inParam#
    </isGreaterThan>
</dynamic>

```

5. <isGreaterEqual>

프로퍼티 속성값이 compare프로퍼티 값이나 compare값보다 크거나 같은지 검사

```

select empno, ename, job
from emp
where 1=1
<dynamic>
  <isGreaterEqual prepend="AND" property="inParam" compareValue="3000">
    sal >= #inParam#
  </isGreaterEqual>
</dynamic>

```

6. <isLessThan>

프로퍼티 속성값이 compare프로퍼티 값이나 compare값보다 작은지 검사

```

select empno, ename, job
from emp
where 1=1
<dynamic>
  <isLessThan prepend="AND" property="inParam" compareValue="1000">
    sal <= #inParam#
  </isLessThan>
</dynamic>

```

7. <isLessEqual>

프로퍼티 속성값이 compare프로퍼티 값이나 compare값보다 작거나 같은지 검사

```

select empno, ename, job
from emp
where 1=1
<dynamic>
  <isLessEqual prepend="AND" property="inParam" compareValue="2000">
    sal <= #inParam#
  </isLessEqual>
</dynamic>

```

8. <isPropertyAvailable>

프로퍼티가 유효한지 검사

```
<isPropertyAvailable property="memberName" prepend=",">
```

9. <isEmpty>

명시된 프로퍼티가 null이거나 빈 문자열(""), 빈 컬렉션이나 빈 String.valueOf()인지를 검사

```
select empno, ename, job
from emp
where 1=1
<dynamic>
  <isEmpty property="inParam" prepend="AND">
    comm is null
  </isEmpty>
</dynamic>
```

10. <isNotEmpty>

명시된 프로퍼티가 null이거나 빈 문자열(""), 빈 컬렉션이나 빈 String.valueOf()가 아닌지 검사

```
select empno, ename, job
from emp
where 1=1
<dynamic>
  <isNotEmpty property="inParam" prepend="AND">
    deptno = #inParam#
  </isNotEmpty>
</dynamic>
```

11. <isParameterPresent>

파라미터 객체가 존재하는지 평가

```
select empno, ename, job
from emp
where 1=1
<dynamic>
  <isParameterPresent prepend="AND">
    ename = #inParam#
  </isParameterPresent>
</dynamic>
```

12. <iterate>요소

컬렉션이나 배열로 된 프로퍼티를 받아서, 그 값들로부터 SQL의 반복적인 부분을 생성

```
Select empno,ename,job from emp
<dynamic prepend="WHERE empno IN ">
    <iterate property="inDs.ENAME" open="(" close=")" conjunction=",">
        ename = #inDs.ENAME#
    </iterate>
</dynamic>
```

13. <iterate>요소-다수

여러 개의 컬렉션이나 배열 프로퍼티를 받아 SQL의 반복적인 부분을 생성

```
select empno,ename,job
from emp
<dynamic prepend="WHERE">
    <iterate property="inDs" open="" close="" conjunction="OR">
        lower(empno) like #inDs.empno# OR
        lower(ename) like #inDs.ename# OR
        lower(job) like #inDs.job#
    </iterate>
</dynamic>
```

14. 복합 dynamic sql

여러 개의 동적 쿼리 사용

```
select empno , ename, job,
sal,to_char(hiredate,'YYYY-MM-DD') hiredate
from emp
where 1=1
<dynamic>
    <isNotEmpty prepend="AND" property="inParam1">
        (ename = #inParam1#
        <isNotEmpty prepend="OR" property="inParam2">
            empno = #inParam2#
        </isNotEmpty>
    )
    </isNotEmpty>
    <isGreaterThan prepend="AND" property="inParam3" compareValue="0">
        sal >= 2000
    </isGreaterThan>
</dynamic>
```

부록 B.

예제 DB 테이블 생성 스크립트

아래는 데이터베이스 예제를 통해 모델 개발 시 사용되는 테이블 생성 스크립트입니다. 본 매뉴얼은 Oracle 데이터베이스를 기준으로 하여 작성하였습니다.

```
drop table address_book;

create table address_book (
    id number primary key,
    name varchar2 (128) not null,
    company varchar2 (128),
    email varchar2 (128),
    phone varchar2 (128),
    birthday date,
    notes varchar2 (256)
);

insert into address_book (id, name, company, email, birthday)
values (0, 'Sheriff Woody', 'TOBESOFT', 'woody@example.com',
to_date('1995/11/22','YYYY/MM/DD'));

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Buzz Lightyear', 'TOBESOFT', 'buzz@example.com',
to_date('1995/11/22','YYYY/MM/DD')
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Mr. Potato Head', 'TOBESOFT', 'potato@example.com',
to_date('1995/11/22','YYYY/MM/DD')
```

```
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Slinky Dog', 'TOBESOFT', 'slinky@example.com',
to_date('1995/11/22', 'YYYY/MM/DD')
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Rex', 'TOBESOFT', 'rex@example.com', to_date('1995/11/22', 'YYYY/MM/DD')
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Hamm', 'TOBESOFT', 'hamm@example.com', to_date('1995/11/22', 'YYYY/MM/DD')
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Bo Peep', 'TOBESOFT', 'peep@example.com',
to_date('1995/11/22', 'YYYY/MM/DD')
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Sarge', 'HN', 'sarge@example.com', to_date('1995/11/22', 'YYYY/MM/DD')
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Lightning McQueen', 'HN', 'lightning@example.com',
to_date('2006/06/09', 'YYYY/MM/DD')
from address_book;

insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Mater', 'HN', 'mater@example.com', to_date('2006/06/09', 'YYYY/MM/DD')
from address_book;
```

```
insert into address_book (id, name, company, email, birthday)
select max(id)+1, 'Sally Carrera', 'HN', 'sally@example.com',
to_date('2006/06/09','YYYY/MM/DD')
from address_book;
```

```
commit;
```

```
drop table dept;
drop table emp;
drop table salgrade;
```

```
create table dept(
    deptno number(2),
    dname varchar2(14),
    loc varchar2(13),
    constraint DEPT_DEPTNO_PK primary key(deptno)
);
```

```
insert into dept values(10,'ACCOUNTING','NEW YORK');
insert into dept values(20,'RESEARCH','DALLAS');
insert into dept values(30,'SALES','CHICAGO');
insert into dept values(40,'OPERATIONS','BOSTON');
```

```
create table emp(
    empno number(4),
    ename varchar2(10),
    job varchar2(9),
    mgr number(4),
    hiredate date,
    sal number(7,2),
    comm number(7,2),
    deptno number(2) not null,
    constraint EMP_DEPTNO_FK FOREIGN key(deptno) REFERENCES dept(deptno),
    constraint EMP_EMPNO_PK primary key(empno)
);
```

```
insert into emp values(7369,'SMITH','CLERK',7902,'80/12/17',800, NULL,20);
insert into emp values(7499,'ALLEN','SALESMAN',7698,'81/02/20',1600,300,30);
insert into emp values(7521,'WARD','SALESMAN',7698,'81/02/22',1250,500,30);
insert into emp values(7566,'JONES','MANAGER',7839,'81/04/02',2975,NULL,20);
insert into emp values(7654,'MARTIN','SALESMAN',7698,'81/09/28',1250,1400,30);
insert into emp values(7698,'BLAKE','MANAGER',7839,'81/05/01',2850,NULL,30);
insert into emp values(7782,'CLARK','MANAGER',7839,'81/05/09',2450,NULL,10);
insert into emp values(7788,'SCOTT','ANALYST',7566,'87/04/19',3000,NULL,20);
insert into emp values(7839,'KING','PRESIDENT',NULL,'81/11/17',5000,NULL,10);
insert into emp values(7844,'TURNER','SALESMAN',7698,'81/09/08',1500,0,30);
insert into emp values(7876,'ADAMS','CLERK',7788,'87/05/23',1100,NULL,20);
insert into emp values(7900,'JAMES','CLERK',7698,'81/12/03',950,NULL,30);
insert into emp values(7902,'FORD','ANALYST',7566,'81/12/03',3000,NULL,20);
insert into emp values(7934,'MILLER','CLERK',7782,'82/01/23',1300,NULL,10);
```

```
create table salgrade(
    grade number,
    losal number,
    hisal number
);
```

```
insert into salgrade values(1,700,1200);
insert into salgrade values(2,1201,1400);
insert into salgrade values(3,1401,2000);
insert into salgrade values(4,2001,3000);
insert into salgrade values(5,3001,9999);
```

```
commit;
```