SOFTBase

xFrame5
개발 가이드

VERSION 1.0

xFrame5

서울특별시 구로구 디지털로 272, 1110 (구로동, 한신 IT 타워)
Phone 02-2108-8030 • Fax 02-2108-8031
www.softbase.co.kr

목차

1 장: xDataSet 개요

이 장에서는 xFrame5 솔루션의 xDataSet 기능에 대한 기술적인 개요에 대하여 설명한다. 여기에서는 xDataSet 내부 구조에 대한 개념적인 부분을 설명하며, 이 장에서 기술하는 내용은 아래와 같다.

- ✓ xDataSet 개요
- ✓ xDataSet 내부 아키텍처
- ✓ xDataSet 데이터 처리 흐름 아키텍처

xDataSet 개요

xDataSet 이란?

xDataSet 이란 xFrame5 솔루션에서 UI 및 서버 부분에서 데이터를 처리하기 위한 아키텍처이다.

xDataSet 아키텍처는 UI 부분에서 처리하는 부분과 서버에서 처리하는 부분이 상호 연관적인 관계를 가지고 유기적으로 데이터를 처리하기 기반을 제공한다.

전체적인 아키텍처 구조는 아래의 그림과 같다.

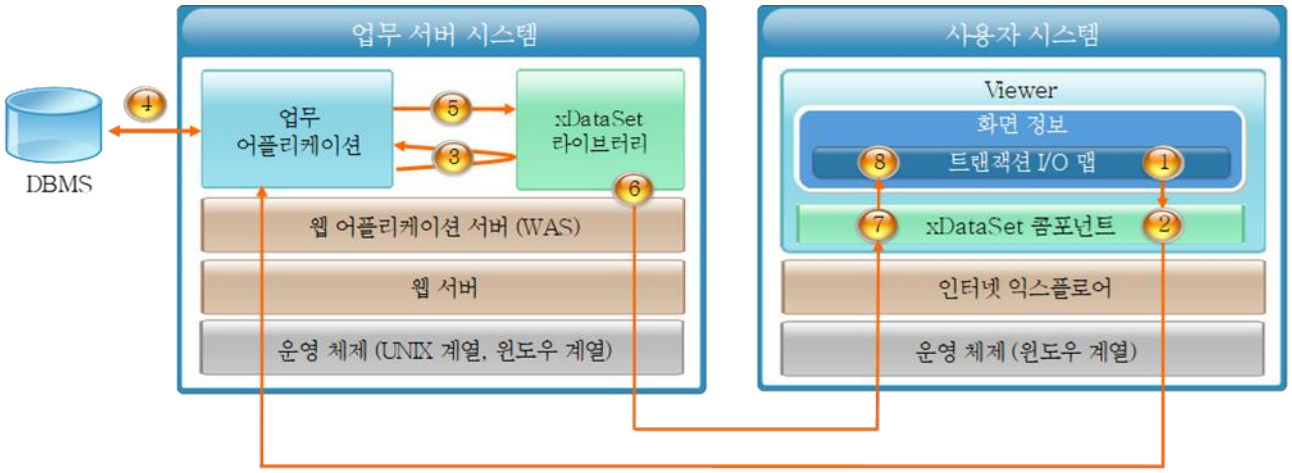


그림. xDataSet 아키텍처

위의 그림에서 데이터 흐름에 대한 설명은 아래와 같다.

구분	설명
1	사용자 시스템에서 사용자가 트랜잭션을 발생시키기 위해서 트랜잭션 I/O 맵에 정의한 트랜잭션 ID 정보를 지정하여 트랜잭션 발생 함수를 호출한다.
2	xFrame-Viewer 내부의 xDataSet 컴포넌트가 트랜잭션 ID 의 입력 맵 정보를 기준으로 데이터 셋에서 업무 서버 시스템으로 송신할 데이터를 조립한다. 이때, 데이터 셋에 저장되어 있는 데이터뿐만 아니라, 화면의 메타 정보도 함께 조립하여 업무 서버 시스템으로 송신한다.
3	업무 서버 시스템의 업무 어플리케이션은 사용자 시스템으로부터 수신 받은 정보를 xdataset5.jar 라이브러리를 이용하여 필요한 데이터를 추출한다.
4	업무 서버 시스템의 업무 어플리케이션은 DBMS 와 연동하여 데이터 처리를 수행한다.
5	업무 서버 시스템의 업무 어플리케이션은 사용자 시스템으로 송신할 데이터를 xdataset5.jar 라이브러리를 이용하여 설정한다.
6	클라이언트로 송신될 데이터가 모두 설정된 이후에, 데이터를 사용자 시스템으로 송신하여, 업무 처리를 완료한다.
7	사용자 시스템에서는 업무 서버로부터 수신 받은 데이터를 xDataSet 컴포넌트에 의해서 자동 분석한다.
8	사용자 시스템의 xDataSet 시스템은 트랜잭션 ID 의 출력 맵 정보를 기준으로 데이터를 화면에 자동으로 표시한다.

xDataSet 주요 기능

xDataSet 의 주요 기능은 아래의 표와 같다.

구분	주요 기능
사용자 시스템 (개발자 시스템)	▪ 글로벌 데이터 셋 기능 ▪ 데이터 셋과 UI 컴포넌트간 1:N 데이터 바인딩 기능 ▪ Query를 이용한 데이터 셋 자동 생성 기능 및 초기 데이터 설정 기능 ▪ 데이터 셋을 이용한 화면 UI 컴포넌트 생성 및 자동 바인딩 기능 ▪ 트랜잭션 맵 기능을 통한 데이터 셋 I/O 설정 기능 ▪ First-Row 방식의 데이터 표시 기능
업무 서버 시스템	▪ 사용자 시스템의 UI로부터 수신된 데이터의 자동 분석 ▪ 사용자 시스템으로 송신할 데이터에 대한 설정 기능 ▪ 화면에 대한 Meta 정보 추출 기능 ▪ 데이터 압축 송수신 기능 ▪ First-Row 방식으로 데이터를 사용자 시스템의 UI로 송신하는 기능

xDataSet 내부 아키텍처

사용자 시스템에서 처리되는 xDataSet 내부 아키텍처는 아래와 같다.

데이터 셋 구분

데이터 셋은 화면 데이터 셋과 글로벌 데이터 셋으로 구분되며, 각각의 특징은 아래와 표와 같다.

구분	주요 기능
화면 데이터 셋	▪ 각각의 개별 화면에서 정의한 데이터 셋 ▪ 각각의 개별 화면에서만 접근 가능
글로벌 데이터 셋	▪ 전역적으로 정의한 데이터 셋 ▪ 개별 화면에서 접근 가능

송수신 데이터 구조

업무 서버로 송신되는 데이터에 대한 구조는 아래의 그림과 같다.

그림. xDataSet 송수신 데이터 구조

송수신 데이터의 각각의 항목에 대한 설명은 아래의 표와 같다.

구분	주요 기능
트랜잭션 URL 파라미터	▪ 트랜잭션 맵에서 트랜잭션 URL 부분에 사용자가 지정한 파라미터 정보 ▪ 파라미터 정보는 GET 방식의 URL 지정 방식과 동일 ▪ 파라미터 이름 중 [XDATASET] 은 xFrame 내부에서 이용하므로 사용하면 안됨

화면 데이터 셋	▪ 각각의 개별 화면에서 지정한 데이터 셋 ▪ 트랜잭션 맵에서 지정한 데이터 셋의 데이터만 송수신 됨.
글로벌 데이터 셋	▪ 글로벌 데이터 셋으로 트랜잭션 맵에서 지정한 데이터 셋만 송수신 됨
화면 메타 정보	▪ 화면 메타 정보는 아래의 데이터로 구성됨 ▪ 화면 속성의 [trancode]와 [transactionheader] 프로퍼티 정보 ▪ 화면 경로 및 이름 : 화면이 개발 툴 내에서 저장되어 있는 경로 및 이름 ▪ 사용자 시스템 IP 주소 : 실제 사용자 IP 주소 ▪ 트랜잭션 맵 ID : 사용자가 발생시킨 트랜잭션 ID

내부 데이터 셋 구조

xDataSet ID

각각의 xDataSet 은 고유의 ID 를 가지며, 각각의 ID 는 다음과 같은 제약 사항이 있다.

구분	주요 기능
화면 데이터 셋	▪ 각각의 화면내의 xDataSet ID는 유일한 값을 가짐 ▪ 화면간 xDataSet의 이름은 동일해도 상관 없음 ▪ 화면내의 xDataSet ID는 글로벌 데이터 셋의 ID와 중복될 수 없음
글로벌 데이터 셋	▪ 전역적으로 유일한 값을 가짐 ▪ 화면내의 xDataSet ID 값과 중복될 수 없음

xDataSet 컬럼 정의

xDataSet 컬럼은 다음과 같은 속성을 가진 컬럼들로 구성된다.

구분	주요 기능
Column	▪ 데이터 셋의 컬럼 이름 ▪ 단일 데이터 셋 내에서는 중복되지 않는 값을 가짐
Description	▪ 컬럼에 대한 부가 설명 ▪ Query 실행기를 통해서 컬럼 정보를 설정할 경우에는, 테이블 정의 시에 지정한 컬럼의 [Comment] 값으로 설정됨
Length	▪ 컬럼에 들어갈 수 있는 데이터의 최대 길이 ▪ Query 실행기를 통해서 컬럼 정보를 설정할 경우에는, 테이블 정의 시에 지정한 컬럼의 [Length] 값으로 설정됨

데이터 셋 내부 구조

xFrameViewer 에서 사용하는 xDataSet 은 아래 그림과 같이 데이터 셋의 데이터 이외에도 부가적인 정보를 가지고 있다.

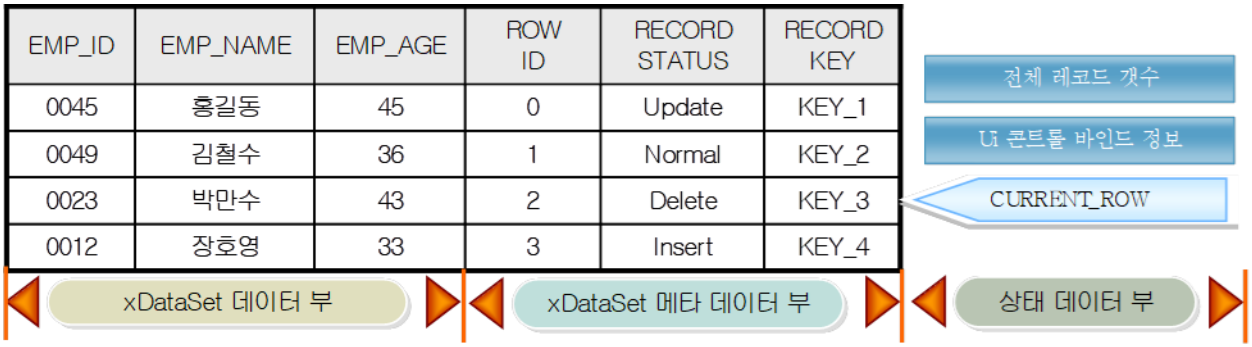


그림. xDataSet 내부 구조

위의 그림에서 예시한 데이터 셋 정보와 같이 데이터 셋 내부는 아래와 같이 데이터 부와 메타 데이터 부로 구성되며, 각 부분에 대한 설명은 아래와 같다.

구분		주요 기능
xDataSet 데이터 부		▪ xDataSet에 저장되어 있는 실제 데이터 ▪ 데이터 셋의 컬럼별/로우별로 데이터가 저장됨
xDataSet 메타 데이터 부	ROW ID	▪ 데이터 셋 레코드의 로우 값
	RECORD_STATUS	▪ 데이터 셋의 변경 상태 정보이며, 트랜잭션이 끝나면 모두 기본 상태의 NORMAL로 초기화됨 ▪ NORMAL : 기존 데이터에서 값이 변경되지 않은 상태 (기본 상태) ▪ UPDATE : 기존 데이터에서 값이 변경된 레코드 상태 ▪ DELETE : 기존 데이터에서 값이 삭제된 레코드 상태 ▪ INSERT : 기존 데이터에 새롭게 추가된 데이터 레코드 상태
	RECORD_KEY	▪ 각 레코드를 구별하는 유일한 값
상태 데이터 부	UI 콘트롤 바인드 정보	▪ 데이터 셋과 BIND된 UI 콘트롤 정보 리스트 ▪ 하나 이상의 UI 콘트롤이 하나의 데이터 셋에 바인딩 될 수 있음
	전체 레코드 개수	▪ 현재 데이터 셋에 저장되어 있는 전체 레코드 개수
	CURRENT_ROW	▪ 현재 사용자가 보고 있는 레코드 위치 ▪ 비어있는 데이터 셋 내에 처음으로 데이터가 저장되는 경우는 첫번째 레코드 위치로 설정

		▪ 데이터 셋이 그리드가 아닌 일반 텍스트성 콘트롤에 바인드 되어 있을 경우, UI 콘트롤에 표시되는 데이터는 CURRENT_ROW에 해당하는 데이터 셋의 정보가 표시됨 ▪ 데이터 셋이 그리드 콘트롤에 바인딩 되어 있을 경우, 이 정보는 현재 UI 콘트롤에서 사용자가 선택한 ROW가 됨

트랜잭션 맵 구조

트랜잭션 맵은 화면에서 업무 서버 시스템과의 통신 시 송수신 데이터에 사용되는 데이터 셋의 정보와 업무 서버 시스템의 URL 정보를 관리하는 맵이다. 트랜잭션 맵은 아래와 같은 정보를 정의한다.

- 하나의 업무 화면에서는 하나 이상의 트랜잭션을 정의할 수 있으며, 각각의 트랜잭션은 트랜잭션 ID로 구분된다.
- 트랜잭션 ID별로 업무 서버 시스템의 URL 정보와 송수신 데이터 셋의 정보를 정의한다.
- 송수신 대상 데이터 셋은 각각의 데이터 셋의 송수신 대상 컬럼을 정의한다.
- 데이터 셋의 송수신 대상 레코드 기준을 정의한다.

업무 서버 시스템 URL 정보 설정 방법

트랜잭션 ID 별로 업무 서버 시스템 URL 을 지정하는 방법은 아래와 같다.

- 업무 서버 시스템 URL = BASE_URL + TRANSACTION_URL

BASE_URL 은 업무 서버 시스템의 기본 URL 정보를 지정하는 값으로 트랜잭션마다 반복되는 정보를 미리 정의하고, 업무 시스템의 URL 정보를 일괄적으로 적용하기 위해서 사용된다.

설정방법은 [UI 개발 환경 설정] – [UI 환경 설정] 항목을 참고한다.

데이터 셋의 송수신 대상 레코드 기준 정의

트랜잭션 ID 에 대한 송수신 대상 데이터 셋을 지정하는 방법은 아래의 표와 같이 입력/출력에 따라 구분되어 지정하며, 설정 값에 따라 데이터를 송수신하는 대상 데이터가 결정된다.

구분		설명
입출력 구분	설정 값	
입력	공백	▪ 트랜잭션 발생시 업무 서버로 송신되지 않는 데이터 셋
	ALL	▪ 트랜잭션 발생시 업무 서버로 데이터 셋의 모든 데이터가 송신됨
	UPDATE	▪ 트랜잭션 발생시 업무 서버로 데이터 셋의 데이터 중에서 [RECORD STATUS] 상태가 "UPDATE, DELETE, INSERT"로 설정된 레코드만 송신됨
	CHECKED	▪ 트랜잭션 발생시 업무 서버로 데이터 셋의 레코드 상태와 무관하게 그리드 콘트롤에서 Check된 레코드만 송신됨. ▪ Check 방식을 사용하기 위해서는 데이터 셋과 링크된 그리드 콘트롤의 [use_checkrow] 및 [multi_checkrow] 속성을 사용함. ▪ 이때, 레코드의 상태는 변경 없음.
출력	공백	▪ 트랜잭션 발생시 업무 서버로부터 수신처리 되지 않음. ▪ 업무 서버에서 해당 데이터 셋에 대한 데이터를 설정한 경우에도 데이터를 처리하지 않음
	UPDATE	▪ 업무 서버로부터 수신된 데이터에서 데이터의 [RECORD KEY] 값을 기준으로 기존 데이터를 업데이트함 ▪ 업무 서버로부터 수신된 데이터에서 [RECORD KEY] 값에 해당하는 데이터가 기존에 없는 경우에는 데이터 셋에 새롭게 추가됨
	CLEAR	▪ 기존에 저장된 데이터 셋의 모든 데이터를 삭제하고, 업무 서버로부터 수신된 데이터로 데이터 셋의 데이터를 설정함

xDataSet 데이터 처리 흐름 아키텍처

일반 데이터 처리 흐름도

xDataSet 의 데이터는 업무 서버 시스템과 사용자 UI 시스템간에 데이터를 아래의 그림과 같이 처리한다.

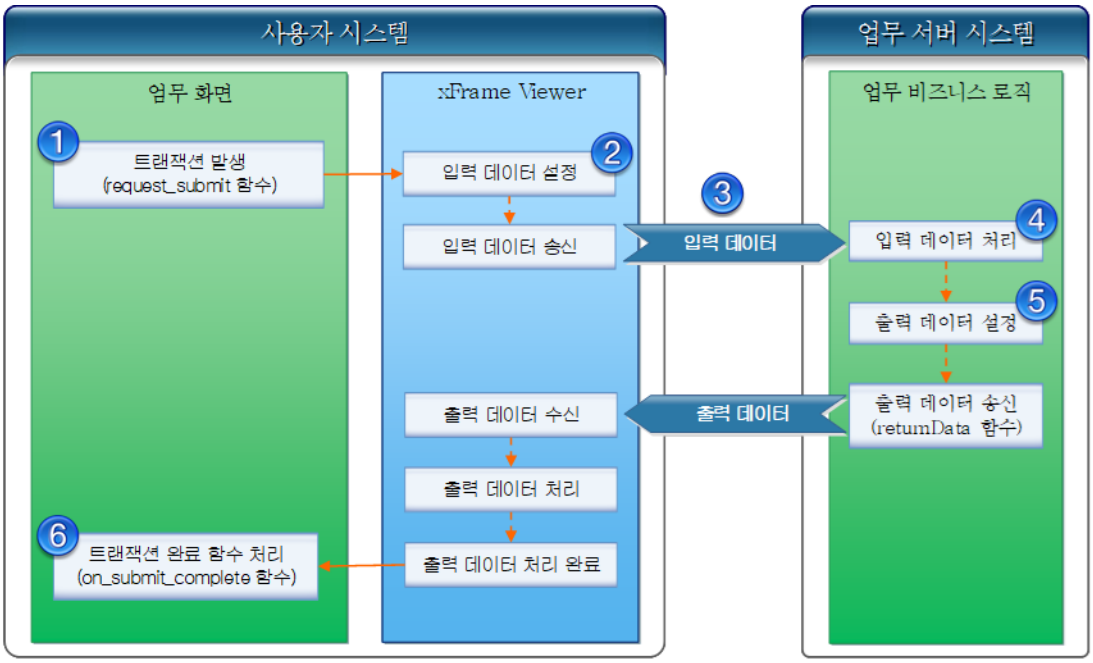


그림. 일반 데이터 처리 흐름도

구분	설명
1	▪ 사용자는 UI 이벤트(예: [조회] 버튼 클릭)를 발생시켜 업무 화면의 request_submit() 함수를 호출하여 정의된 트랜잭션 ID에 해당하는 트랜잭션을 발생시킴
2	▪ xFrame Viewer에서 트랜잭션 ID에 해당하는 업무 서버 시스템의 URL 정보와 트랜잭션 IO 맵 정보를 기준으로 업무 서버 시스템으로 송신할 데이터를 설정
3	▪ 송신 데이터는 데이터 셋의 데이터 이외에도 부가적인 데이터가 송신됨 ▪ 송신 데이터의 구조는 [xDataSet 내부아키텍처 - 송수신 데이터 구조] 참조
4	▪ xdataset5.jar 라이브러리를 이용해서 입력 데이터를 얻어오고 비즈니스 업무 요건에 따라 처리
5	▪ xdataset5.jar 라이브러리를 이용해서 출력 데이터를 설정
6	▪ xFrame Viewer에서 출력 데이터에 대한 수신 및 데이터 처리가 완료되면, 개발자가 정의한 트랜잭션 완료 시 처리할 비즈니스 로직을 처리하기 위해, 화면의 on_submit_complete() 함수를 호출

2 장: xFrame5 개발 환경 설정

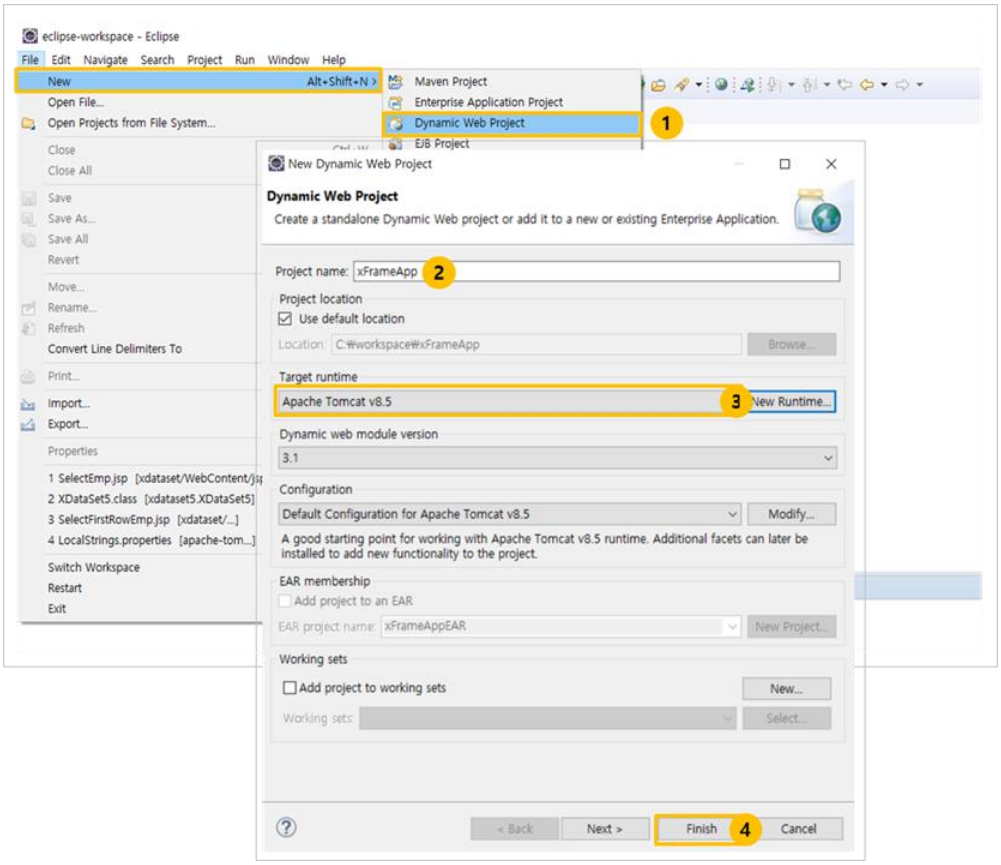
이 장에서는 xFrame5 개발 환경을 설정하는 방법을 설명한다.

- ✓ xFrame5 서버 환경 설정
- ✓ xFrame5 UI 개발 환경 설정

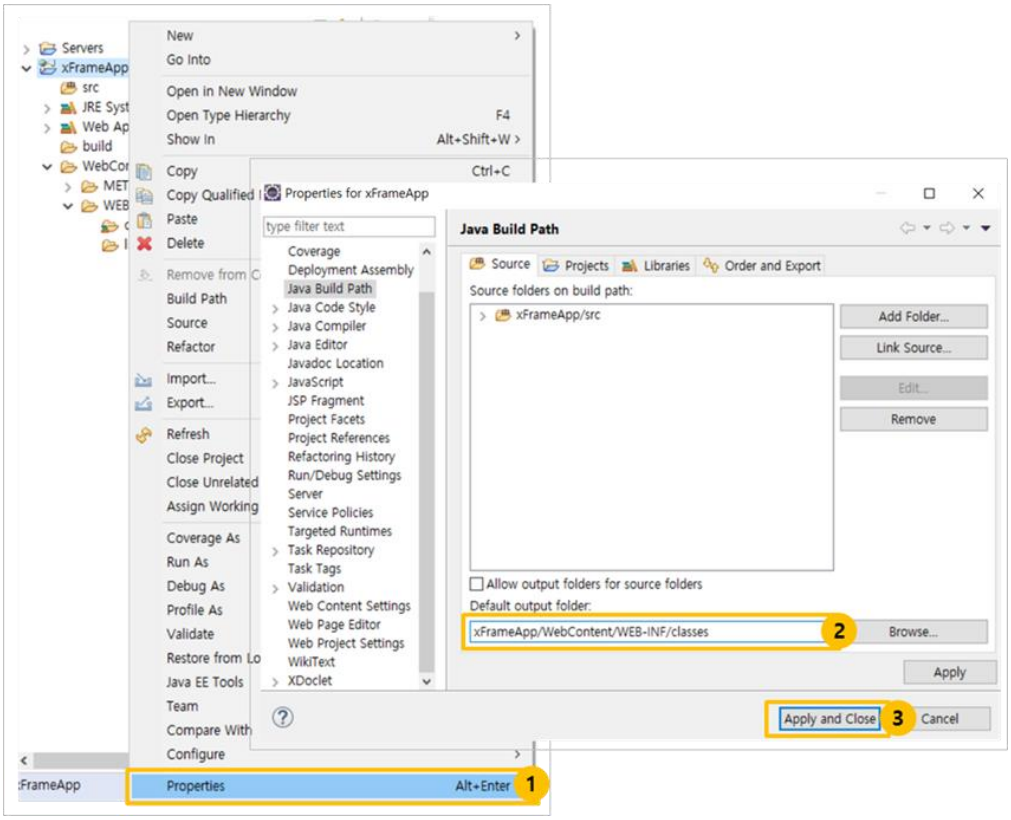
서버 환경 설정

Eclipse 프로젝트 생성

Eclipse 에서 다음과 같이 신규 프로젝트를 생성한다. (project name : xFrameApp)

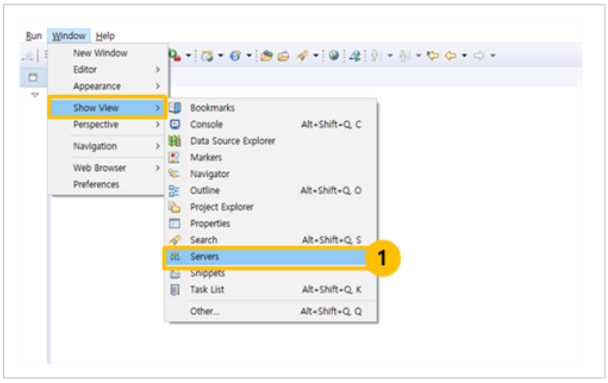


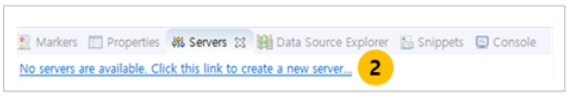
새로 생성된 프로젝트의 [Properties]를 선택하여 Default Output Folder 를 xFrameApp/WebContent/WEB-INF/classes 로 선택한다. classes 폴더가 없을 경우에는 생성한 후 설정한다.

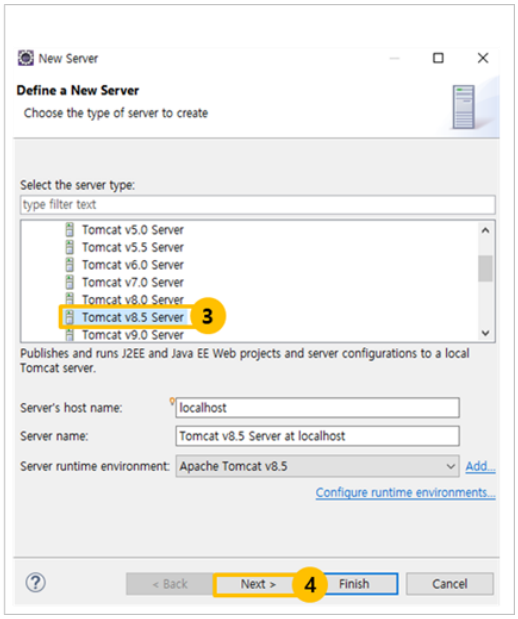


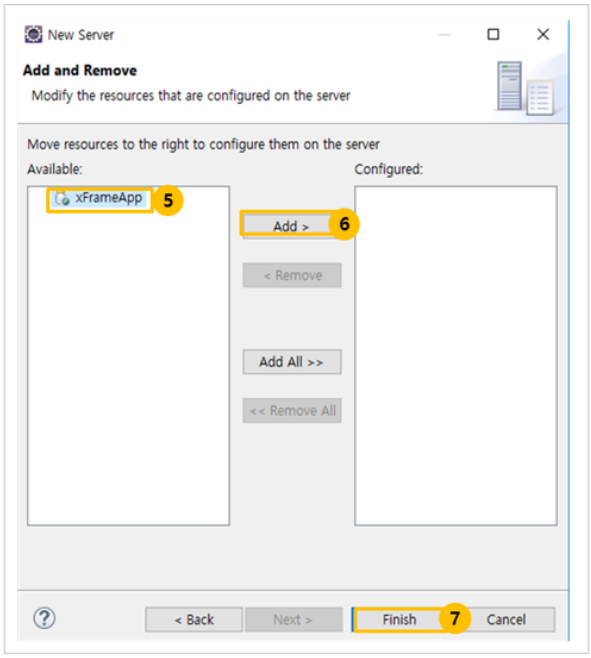
톰캣 서버 설정

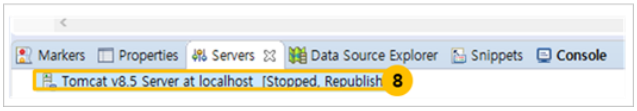
[Window] – [Show View] – [Servers] 를 클릭한 후 [Servers] 탭에서 새로운 서버를 생성하고 앞서 생성한 프로젝트(xFrameApp)을 추가한다. 톰캣 서버의 버전은 해당 환경에 맞게 선택한다.





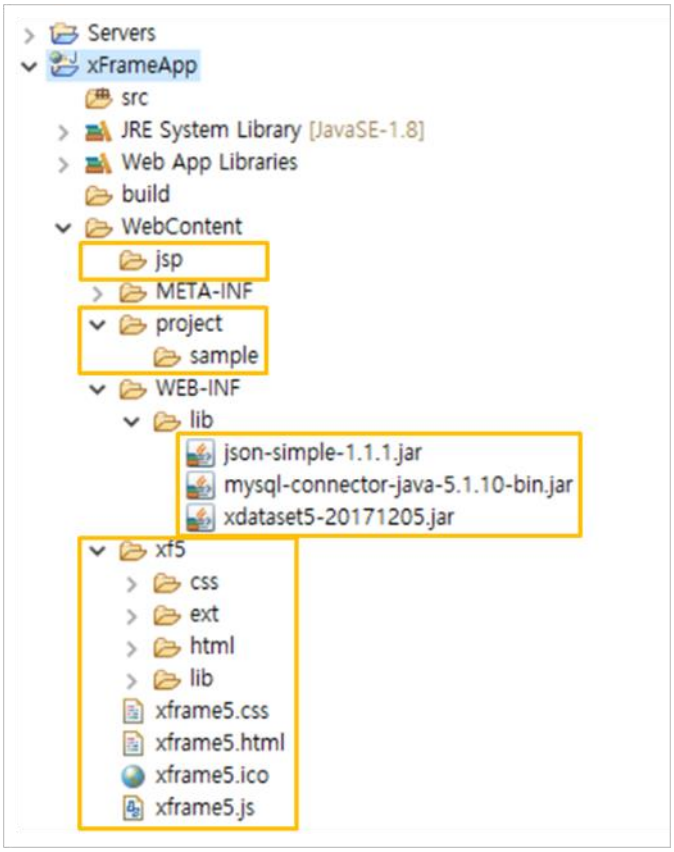



New Server
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
xFrameApp
5
Add >
6
< Remove
Add All >>
<< Remove All
< Back
Next >
Finish
7
Cancel


Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Tomcat v8.5 Server at localhost [Stopped, Republish
8

xFrame5 서버 환경 설정

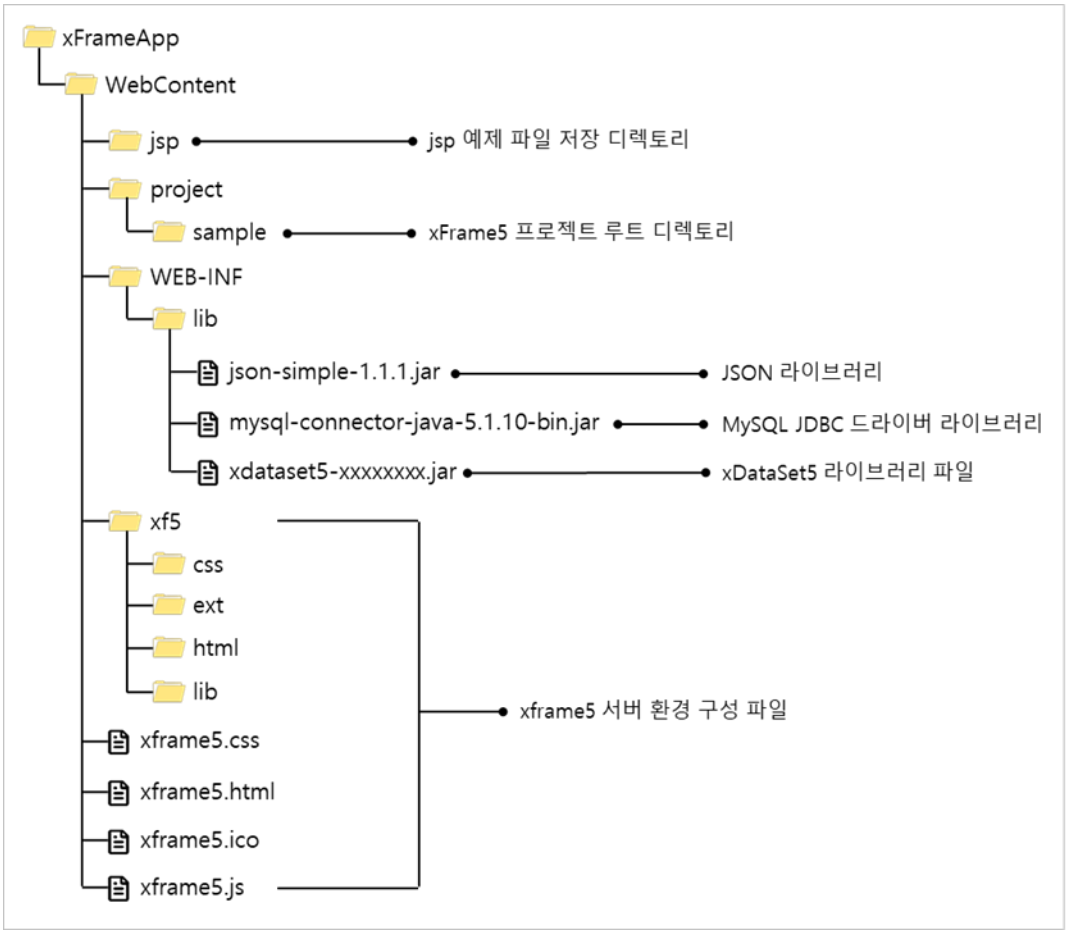
다음 그림을 참고로 xFrame5 의 서버 환경을 구성한다.



- WebContent 디렉토리 하위에 project를 생성하고 그 하위에 sample 디렉토리를 생성한다.
- C:₩xFrame5 디렉토리에 xframe5.css, xframe5.html, xframe5.ico, xframe5.js 파일을 WebContent 디렉토리에 복사한다.
- xFrame5 설치 디렉토리인 C:₩xFrame5 디렉토리 하위에 xfs 디렉토리 중, css, ext, html, lib 폴더를 WebContent 디렉토리로 복사한다.
- C:₩xFrame5₩jar₩ 디렉토리에 xdataset5-xxxxxxxx.jar, mysql-connector-java-5.1.10-bin.jar 및 json-simple-1.1.1.jar 파일을 WebContent₩WEB-INF₩lib₩ 에 복사한다
- WebContent 하위에 jsp 폴더를 생성한다.

프로젝트 디렉토리 구조 및 파일 개요

프로젝트 디렉토리 구조 및 내용은 아래와 같다.



XDATASET_BASEURL 설정

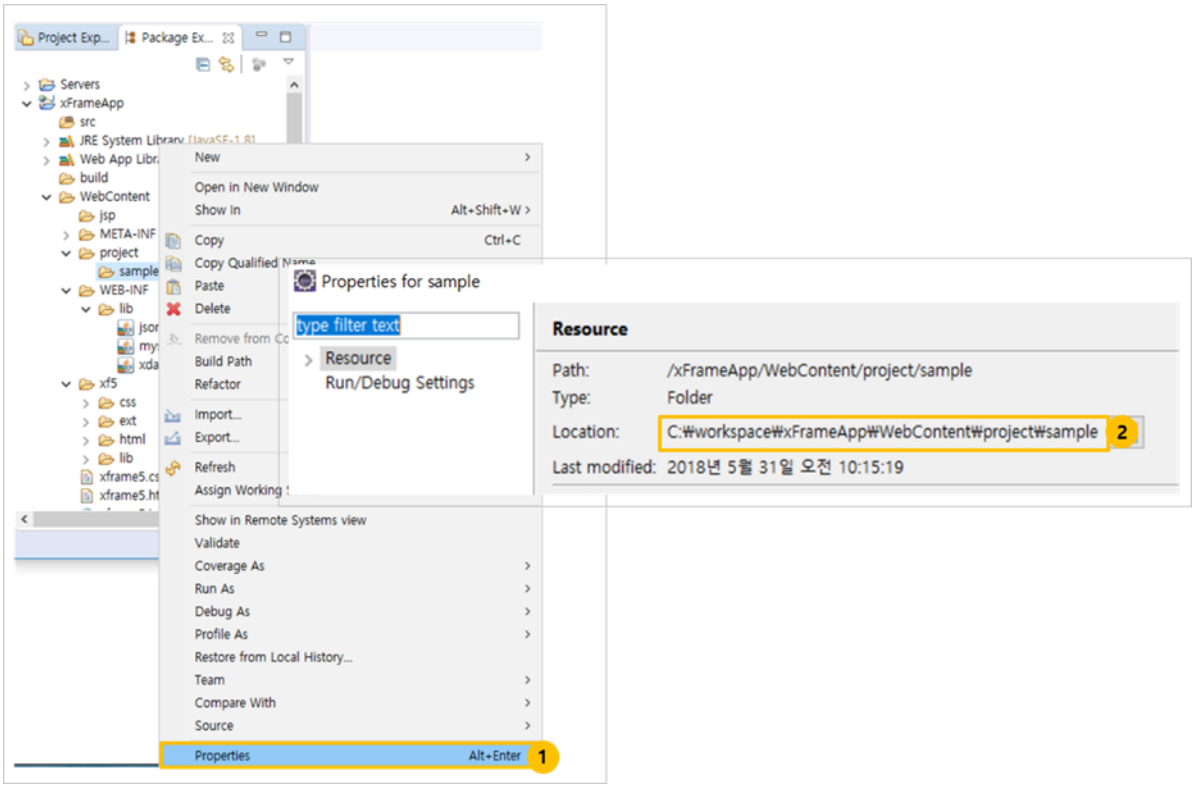
xframe5.js 파일 내 XDATASET_BASEURL 을 http://127.0.0.1:8080/xFrameApp/jsp/ 로 설정한다.

```
SCREENBASEURL: './project/sample/screen',
IMAGEBASEURL: './project/sample/image',
PICKLISTBASEURL: './project/sample/picklist',
MENUBASEURL: './project/sample/menu',
STYLEBASEURL: './project/sample/style',
COMMONMODULEBASEURL: './project/sample/screen/common_module',
COMMONXDATASETBASEURL: './project/sample/screen/common_xdataset',
XDATASET_BASEURL : 'http://127.0.0.1:8080/xFrameApp/jsp/',
XTRAN_COMPRESS: true,
XTRAN_TIMEOUT: 1000 * 60 * 60,
XTRAN_DATAFORMAT: 'json',
XTRAN_SENDDATAKEY: 'XDATASET5',
XTRAN_URLENCODING: true,
XTRAN_XMLSENDDATAFUNC: 'XTRAN_XMLSENDDATAFUNC',
XTRAN_XMLRECVDATAFUNC: 'XTRAN_XMLRECVDATAFUNC',
```

UI 개발 환경 설정

xFrame5 프로젝트 생성 디렉토리 확인

Eclipse 에서 생성한 xFrameApp 프로젝트 내 WebContent₩project₩sample 폴더를 선택 후 마우스 우클릭하여 [Properties] 를 선택하여 xFrame5@DevStudio 에서 프로젝트 루트 디렉토리로 사용할 디렉토리를 확인한다.

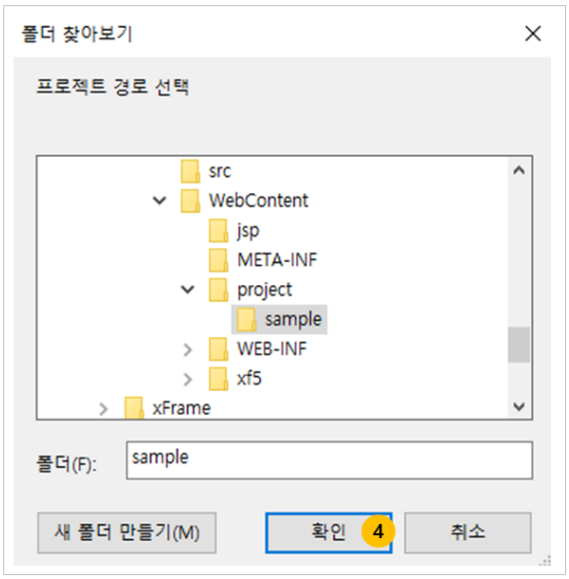


xFrame5@DevStudio 시작

바탕화면에 xFrame5@DevStudio 아이콘을 더블클릭하여 DevStudio 를 실행한다.

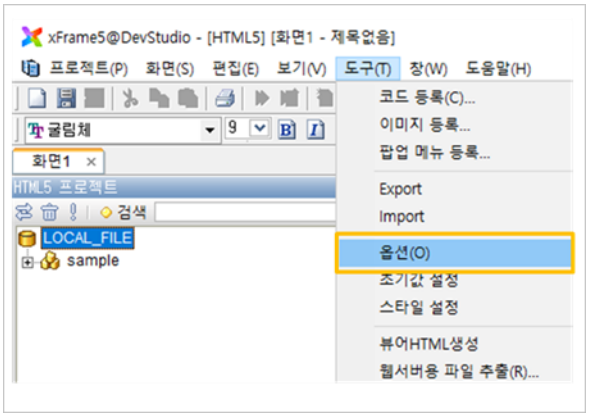
시작 화면에서 [File]탭 > [프로젝트 생성] 탭을 선택한 후 프로젝트 생성 경로를 [xFrame5 프로젝트 생성 디렉토리 확인]에서 확인한 디렉토리를 선택한다.

프로젝트 아이디와 이름을 [sample]로 입력한 후 생성 버튼을 클릭한다.

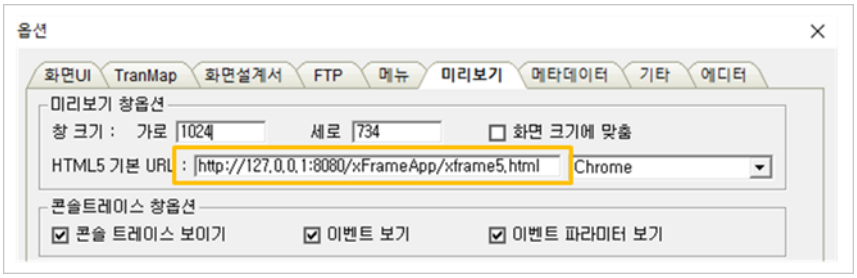


UI 환경 설정

xFrame5@DevStudio [도구] 메뉴에서 [옵션]을 선택한다.



[미리보기] 탭에서 HTML5 기본 URL 을 http://127.0.0.1:8080/xFrameApp/xframe5.html 로 입력한다.



[기타] 탭에서 Transaction 탭 보기를 선택하고, 인코딩 방식을 UTF-8 로 선택한다.


옵션
화면UI
TranMap
화면설계서
FTP
메뉴
미리보기
메타데이터
기타
에디터
xFrameViewer 클래스아이디 : 525CE85A-4C03-49a2-BB50-DB94C12F8446
Browser Type : Default Web Browser xFrame Browser
뷰어 경로 : C:₩xFrame5₩bin₩xFrameBrowser.exe
전용브라우저 환경파일 경로: C:₩xFrame5₩bin₩xFrameConfig.ini
코드 설정창 초기경로 :
이미지 설정창 초기경로 :
xDataSet Base URL : http://127.0.0.1:8080/xFrameApp/jsp/
Transaction 탭 보기
Transaction 탭에서 xTranMap 사용
TranMap 탭 보기
TranLink 탭 보기
AttributeMap 탭 보기
TranMap/TranLink, Transaction 탭 사용 옵션과 다르게 설계된 화면을 여는 경우 경고창 보이기
입력 인덱스 자동 매김
인덱스 시작 : 0
인덱스 간격 : 1
한글 이름 사용 : 프로젝트 화면 아이디 오브젝트 이벤트 함수
파일 추출시 : 인코딩 방식 : UNICODE(UTF-8)
자바스크립트 난독화
압축여부 물어보기 압축하기 압축안하기
화면 도움말 작성 확인하기

3 장: xDataSet 활용 예제

이 장에서는 xDataSet 을 활용한 예제를 통해 xDataSet 의 사용법을 익힌다.

- ✓ 예제 개요
- ✓ 예제 DB구성
- ✓ 예제 UI 개발 및 서버 개발

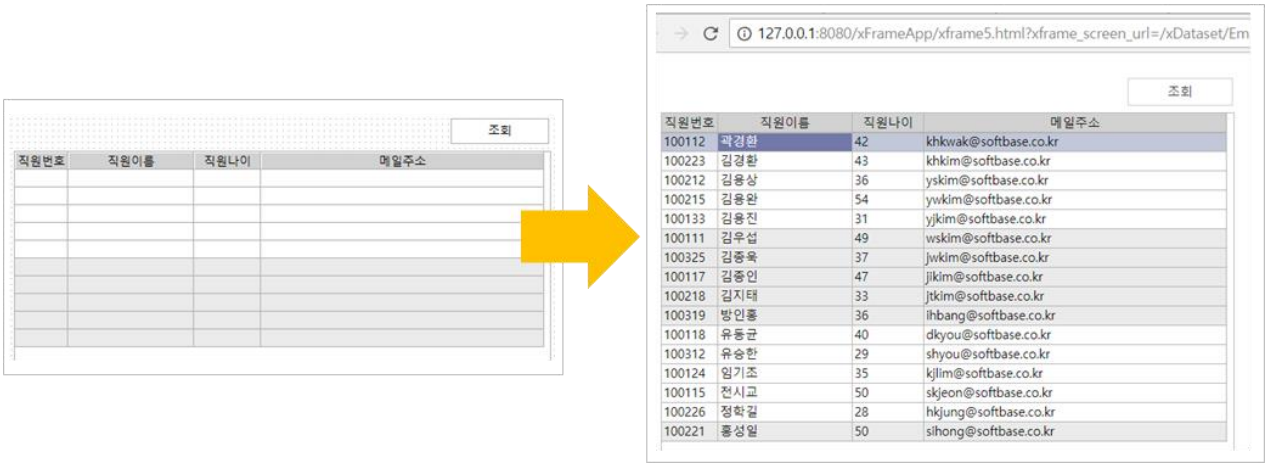
예제 개요

xDataSet 을 이용한 직원정보조회 예제를 따라 해보며, xDataSet 의 기본적인 사용방법을 학습한다. 이 예제를 통해 트랜잭션과 데이터 셋의 사용방법도 함께 살펴보도록 한다.

본 예제는 다음과 같은 개발 환경을 전제로 기술한다.

구분	소프트웨어	참고사이트
개발 툴	Eclipse	https://www.eclipse.org/
웹 서버	Tomcat	http://tomcat.apache.org/
DB	MySQL	https://www.mysql.com/

다음은 이 예제의 최종 실행화면이다.



예제 DB 구성

테이블 생성

테이블 스키마

예제에서 활용될 스키마는 아래와 같으며, [테이블 생성 스크립트] 항목을 참고로 DB 에 해당 테이블을 생성한다.

테이블 명 : EMP_INFO

컬럼 명	데이터 형식	길이	Key	Comment
emp_no	CHAR	6	PK	직원번호
emp_name	VARCHAR	20		직원이름
emp_age	INT	3		직원나이
mail_addr	VARCHAR	100		메일주소

테이블 생성 스크립트

다음 스크립트를 참고로 예제 테이블을 생성한다.

```
CREATE TABLE `emp_info` (
  `emp no` CHAR(6) NOT NULL COMMENT '직원번호',
  `emp_name` VARCHAR(20) DEFAULT NULL COMMENT '직원이름',
  `emp_age` INT(3) DEFAULT NULL COMMENT '직원나이',
  `mail_addr` VARCHAR(100) DEFAULT NULL COMMENT '메일주소',
  PRIMARY KEY  (`emp_no`)
) ENGINE=INNODB DEFAULT CHARSET=utf8;
```

데이터 생성

데이터 입력 스크립트

다음 스크립트를 참고로 예제 데이터를 입력한다.

```
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100111','김우섭',49,'wskim@softbase.co.kr');
INSERT  INTO `emp_info`(`emp_no`,`emp_name`,`emp_age`,`mail_addr`) VALUES
('100112','곽경환',42,'khkwak@softbase.co.kr');
INSERT  INTO `emp_info`(`emp_no`,`emp_name`,`emp_age`,`mail_addr`) VALUES
('100115','전시교',50,'skjeon@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100117','김종인',47,'jikim@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
```

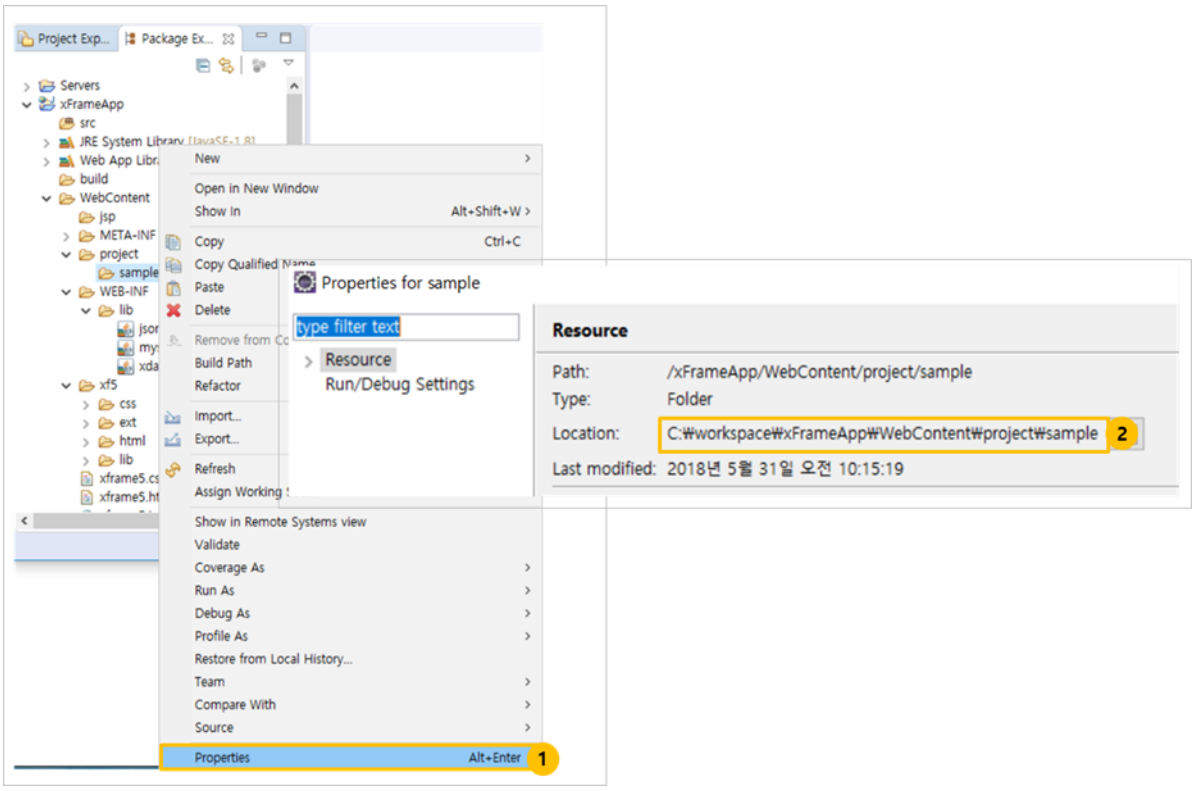
```
('100118','유동균',40,'dkyou@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100124','임기조',35,'kjlim@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100133','김용진',31,'yjkim@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100212','김용상',36,'yskim@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100215','김용완',54,'ywkim@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100218','김지태',33,'jtkim@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100221','홍성일',50,'sihong@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100223','김경환',43,'khkim@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100226','정학길',28,'hkjung@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100312','유승한',29,'shyou@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100319','방인홍',36,'ihbang@softbase.co.kr');
INSERT  INTO `emp info`(`emp no`,`emp name`,`emp age`,`mail addr`) VALUES
('100325','김종욱',37,'jwkim@softbase.co.kr');
commit;
```

UI 개발

UI 개발 환경 설정

xFrame5 프로젝트 생성 디렉토리 확인

Eclipse 에서 생성한 xFrameApp 프로젝트 내 WebContent₩project₩sample 폴더 선택 후 마우스 우클릭하여 [Properties] 를 선택하여 xFrame5@DevStudio 에서 프로젝트 루트 디렉토리로 사용할 디렉토리를 확인한다.



xFrame5@DevStudio 시작

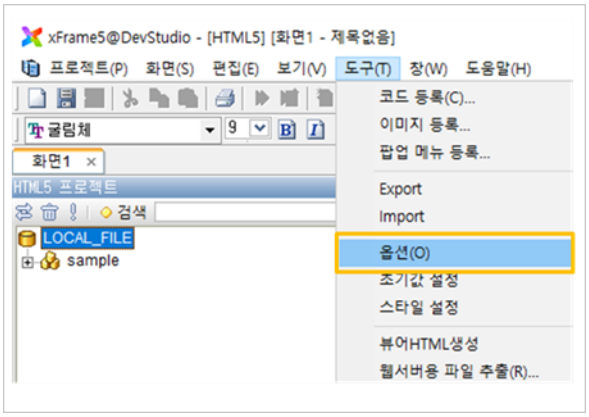
바탕화면에 xFrame5@DevStudio 아이콘을 더블클릭하여 DevStudio 를 실행한다.

시작 화면에서 [File]탭 > [프로젝트 생성] 탭을 선택한 후 프로젝트 생성 경로를 [xFrame5 프로젝트 생성 디렉토리 확인]에서 확인한 디렉토리를 선택한다.

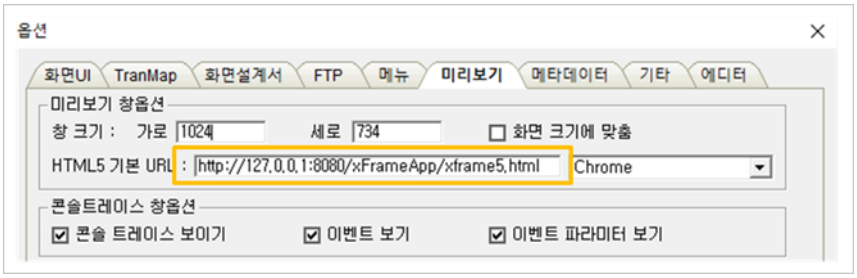
프로젝트 아이디와 이름을 [sample]로 입력한 후 생성 버튼을 클릭한다.

UI 환경 설정

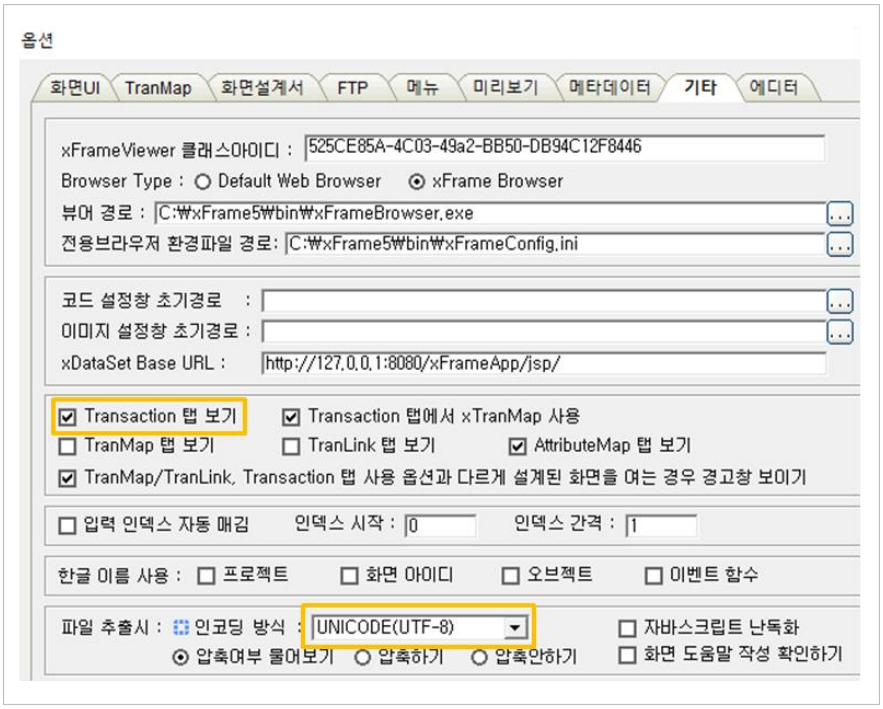
xFrame5@DevStudio [도구] 메뉴에서 [옵션]을 선택한다.



[미리보기] 탭에서 HTML5 기본 URL 값으로 http://127.0.0.1:8080/xFrameApp/xframe5.html 을 입력한다.



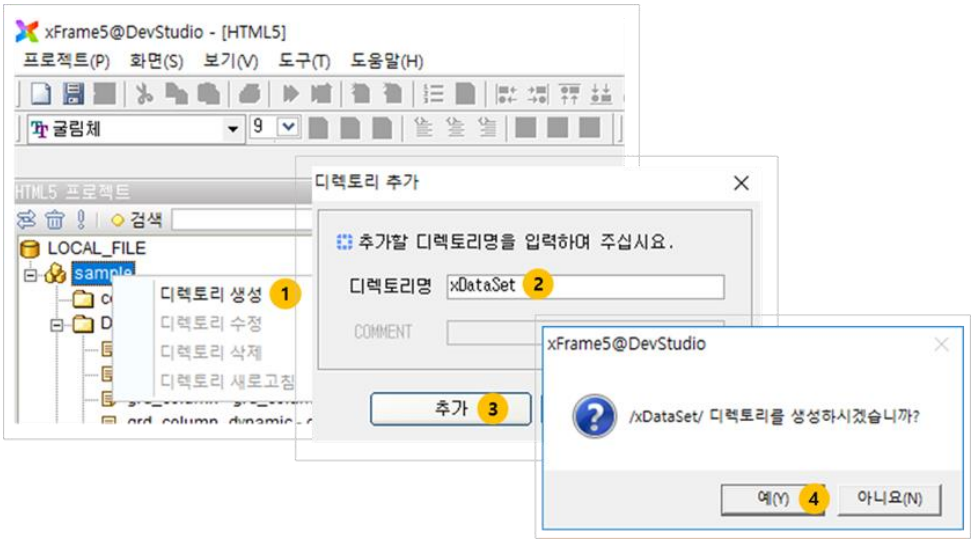
[기타] 탭에서 Transaction 탭 보기를 선택하고, 인코딩 방식을 UTF-8 로 선택한다.



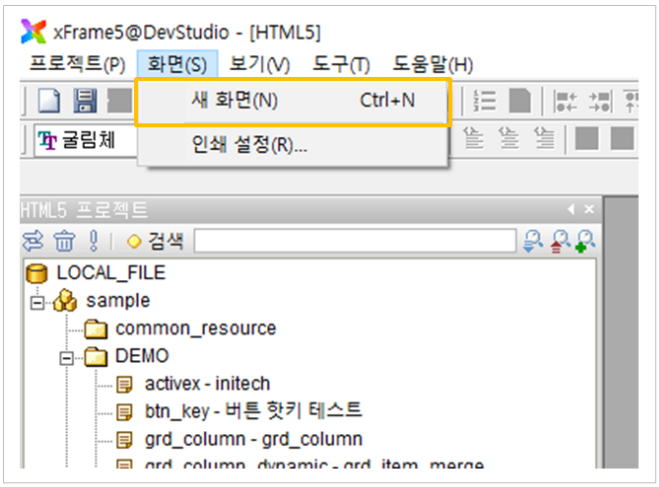
UI 화면 구성

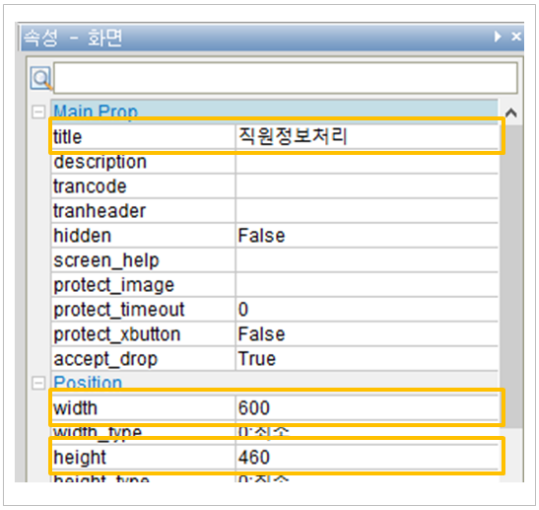
새 화면 생성

프로젝트 창에서 프로젝트명을 선택 후 마우스 우 클릭하고 [디렉토리 생성] 메뉴를 선택한다. 디렉토리명은 "xDataset"으로 입력하여 디렉토리를 만든다.

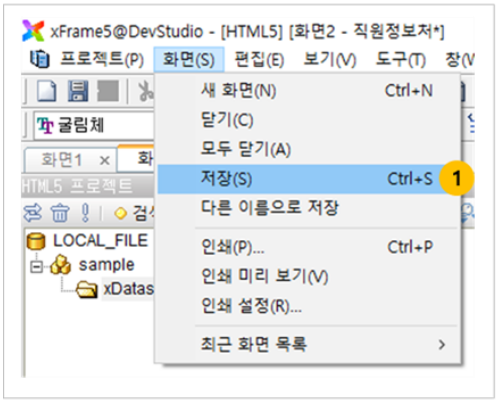


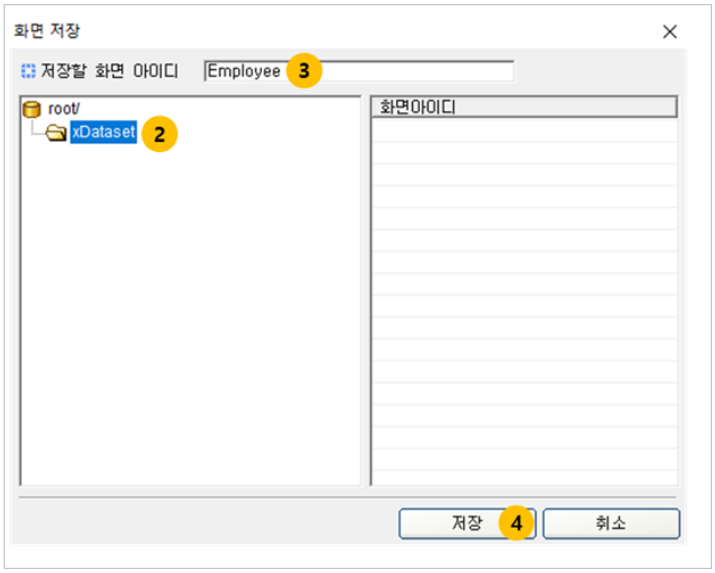
[화면] -> [새 화면] 메뉴를 선택하여 새 화면을 생성한다. 속성 창에서 title, width, height 을 설정한다.





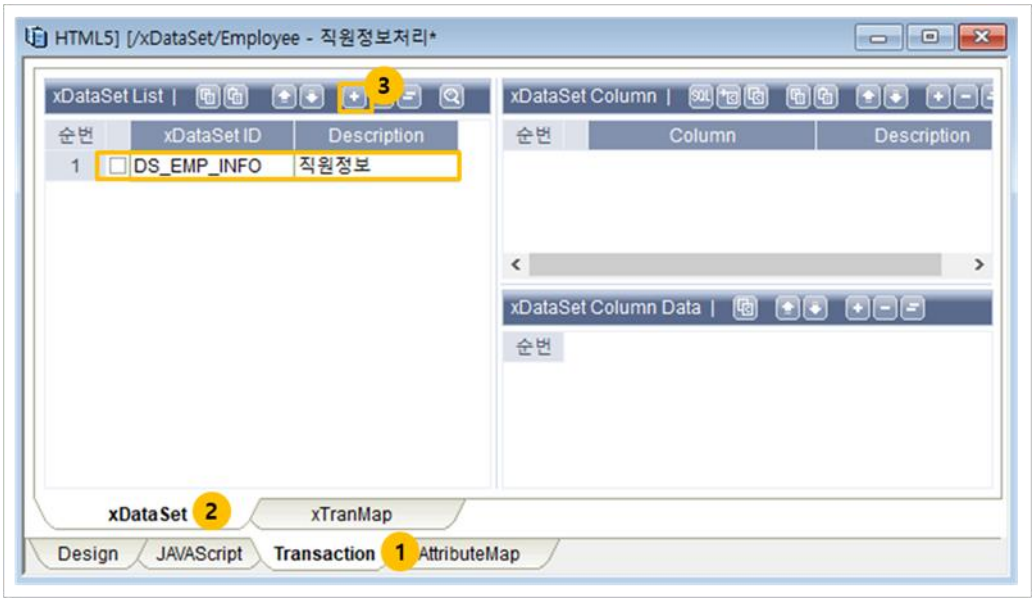
[화면] -> [저장] 메뉴를 선택하고 [xDataset] 디렉토리를 선택한 후 "Employee" 라는 이름으로 저장한다.





데이터 셋 설정

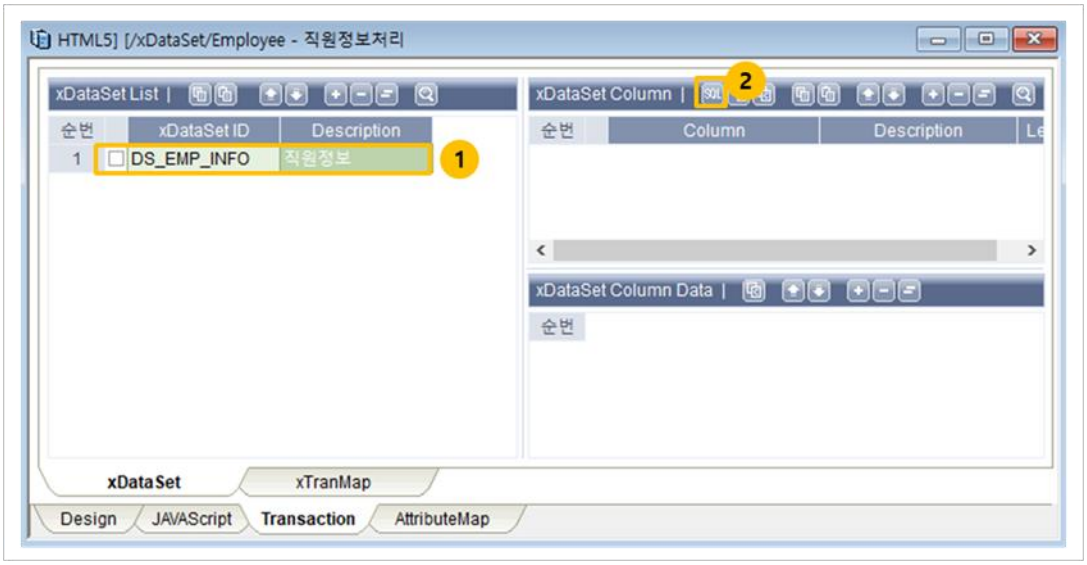
화면 하단의 [Transaction] 탭의 [xDataSet] 탭을 선택한다. xDataSet List 에 xDataSet 을 추가한 후 아래와 같이 [xDataset ID]와 [Description]을 입력한다.



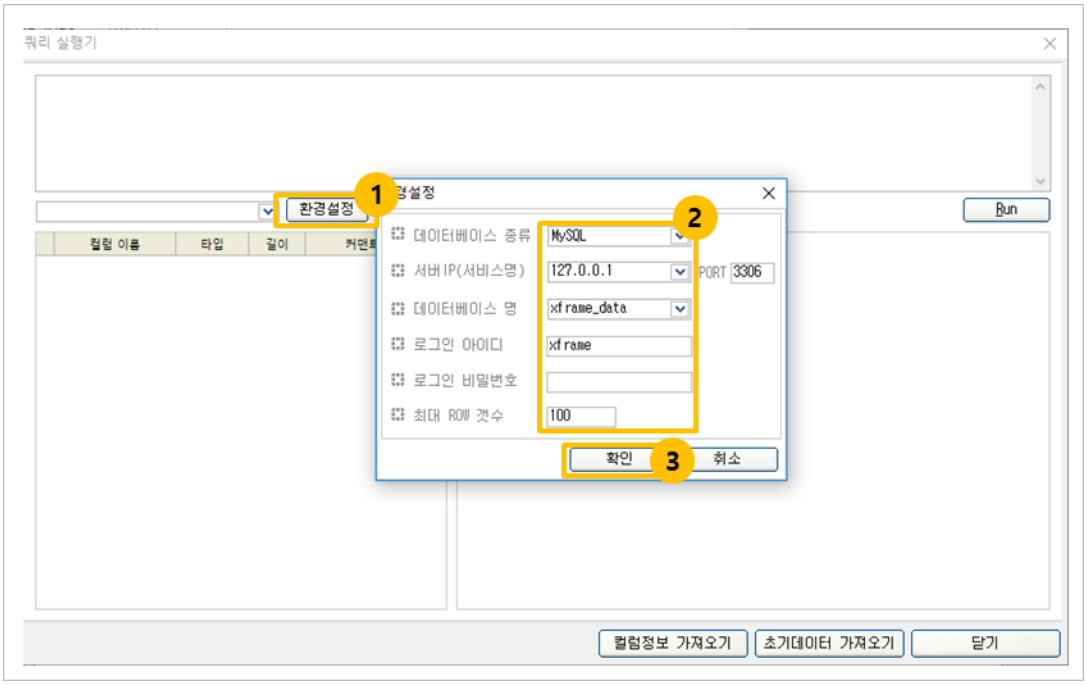
데이터 셋 컬럼 정의

직원 정보를 저장할 DS_EMP_INFO 데이터 셋의 컬럼을 정의한다. DB 상에 존재하는 테이블을 가져와 데이터 셋의 컬럼을 정의하기 위해서 쿼리 실행기를 이용하여 [Column]을 정의한다.

다음 그림과 같이 xDataSet List 에서 [DS_EMP_INFO]를 클릭하고 [xDataSet Column]에 있는 [쿼리 실행기]를 클릭한다.



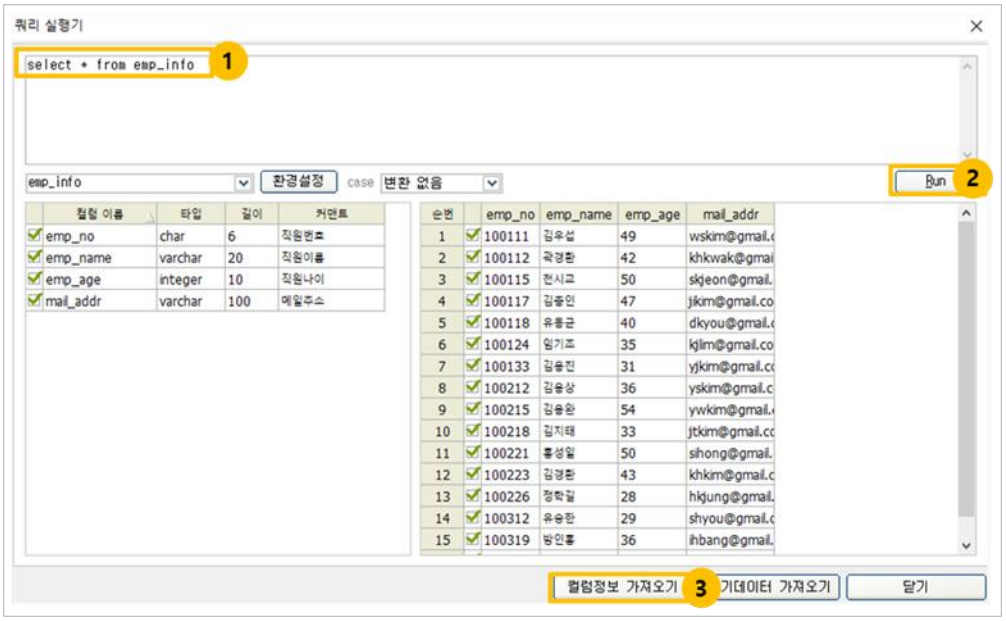
[쿼리 실행기]가 실행 되면 [환경 설정] 버튼을 클릭하여 현재 사용하고 있는 데이터베이스 정보를 선택 및 입력한다.



[쿼리 실행기]에서 "select * from emp_info" 질의문을 입력하고 [Run]버튼을 클릭하면 DB 의 테이블과 값들을 가져온 것을 확인 할 수 있다.

[DS_EMP_INFO]의 [xDataSet Column]을 DB 와 동일하게 정의하기 위해서 [컬럼정보 가져오기] 버튼을 클릭한다.

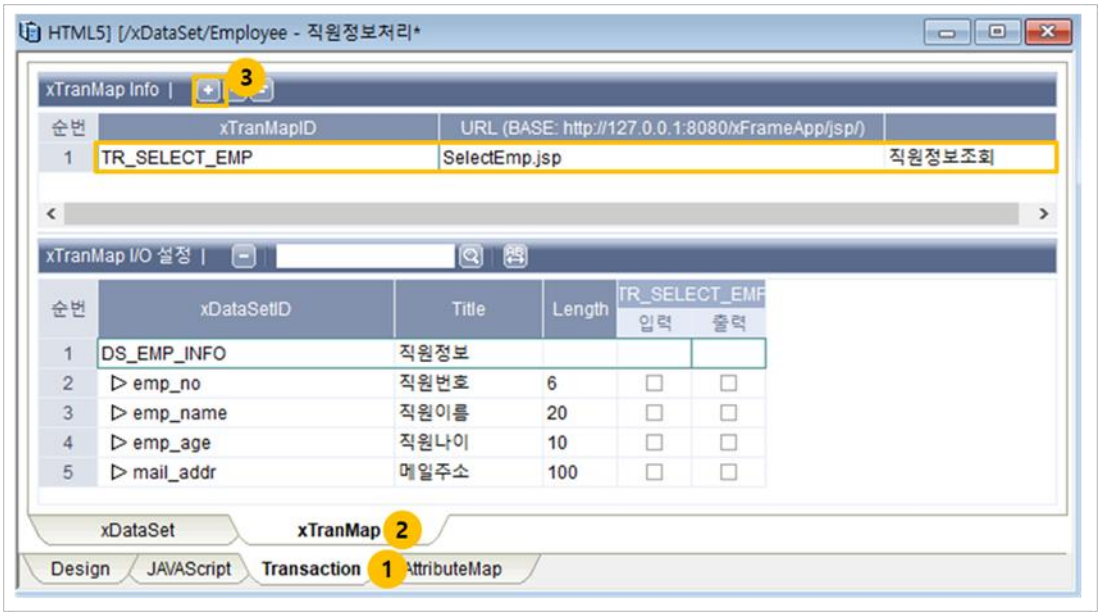
참고로 [초기 데이터 가져오기] 버튼을 누르면 DB 에 저장된 데이터를 [Column Data]에 가져올 수도 있다.



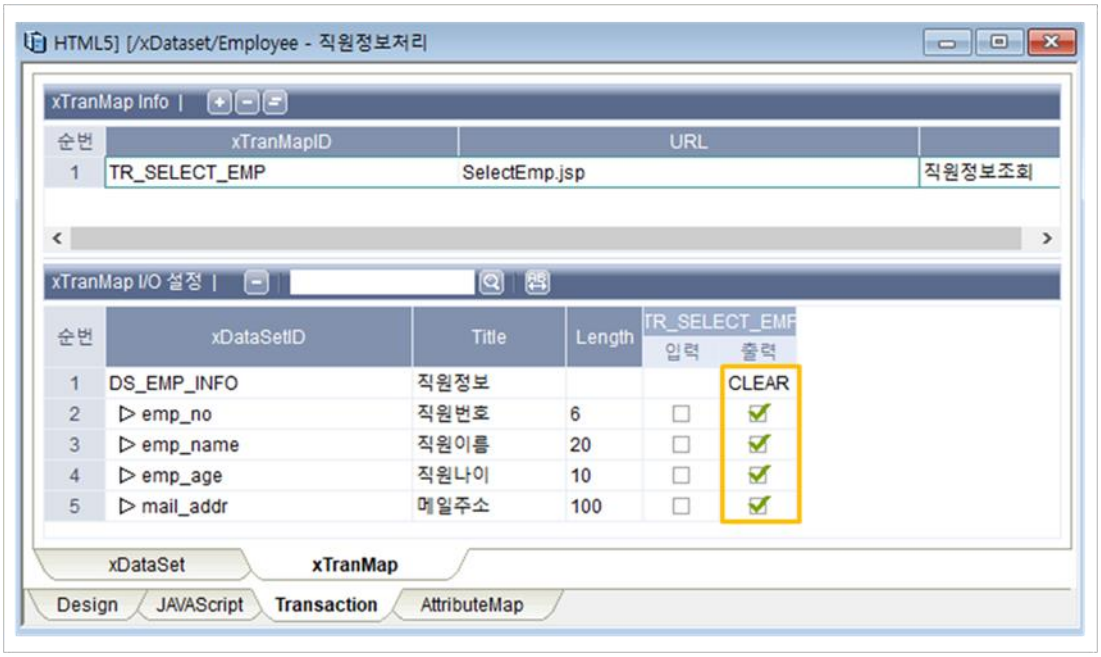
트랜잭션 설정

트랜잭션은 정보의 교환이나 데이터베이스 갱신 등 일련의 작업들에 대한 연속처리 단위를 의미한다. 화면에서 발생하는 트랜잭션을 정의하기 위하여 [Transaction]탭내에 [xTranMap] 탭을 선택한다.

xTranMap Info 에 트랜잭션을 추가한 후 아래와 같이 [xTranMapID], [URL], [Comment] 값을 입력한다.



[xTranMap I/O]를 설정한다. 입력 및 출력 항목에 대한 자세한 설명은 [xDataSet 내부 아키텍처 - 내부 데이터 셋 구조]를 참고한다.



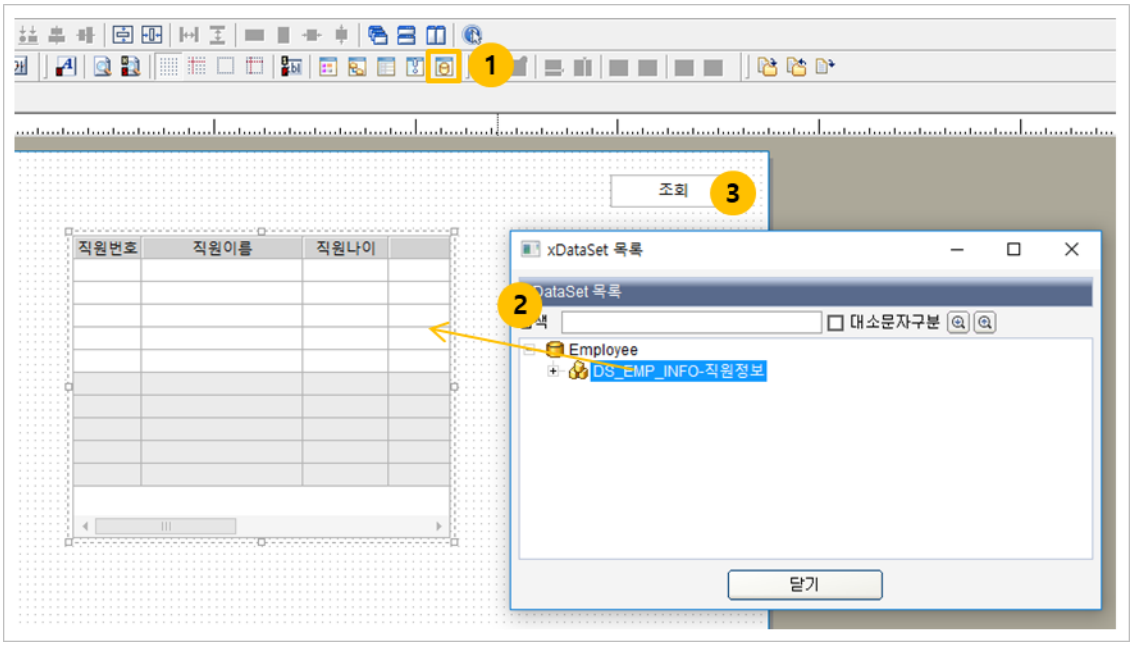
Transaction I/O 설정

xTranMapID	출력데이터	데이터구분	설명
TR_SELECT_EMP	DS_EMP_INFO	CLEAR	직원 정보를 저장하고 있는 데이터 셋의 데이터를 삭제하고 업무 서버로부터 수신한 데이터를 이용해서 기존 데이터로 저장한다.

UI 화면 개발

직원 정보 그리드는 xDataSet 과 연결 되어 서버에서 보내주는 데이터를 가져와 화면에 출력 한다.

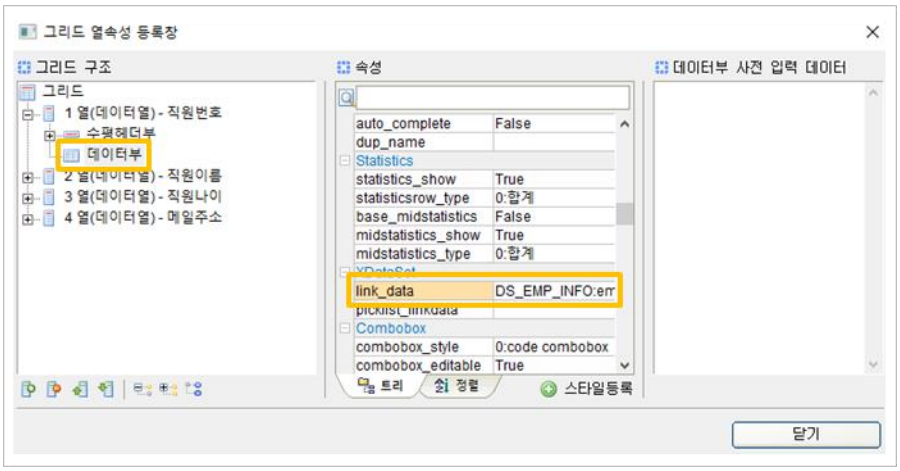
다음 그림과 같이 xDataSet 목록 창에서 [DS_EMP_INFO]를 클릭하고 Drag & Drop 으로 그리드를 생성하고, 이벤트를 처리할 [조회] 버튼을 생성한다.



화면 구성 컨트롤 및 속성

구분	컨트롤	속성	값
1	버튼	name	btnSelect
		text	조회
2	그리드	name	grdEmpList

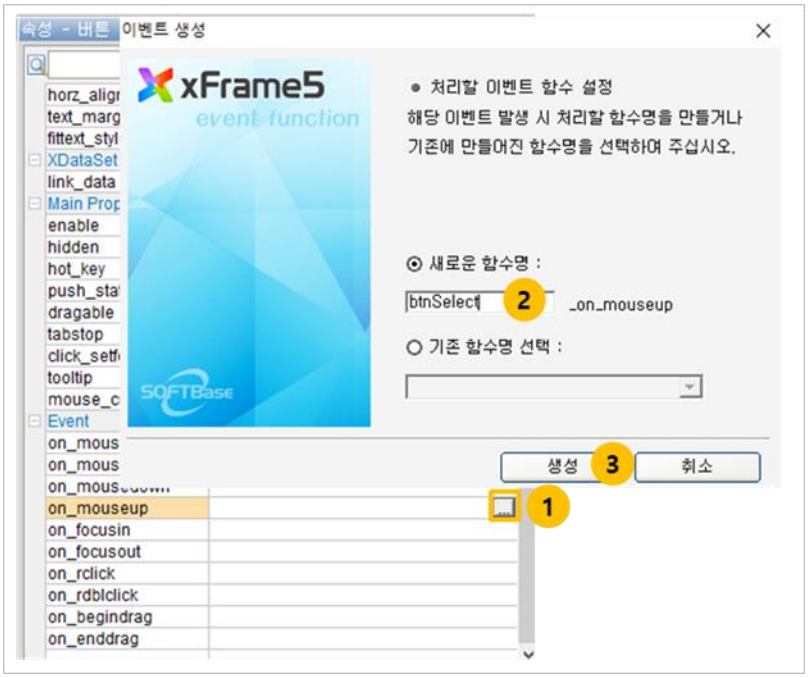
그리드를 더블클릭 하면 [그리드 열 속성 등록 창]이 나온다. 각 열의 [데이터부]에 [link_data] 속성을 확인해보면 xDataSet 과 연동되는 ID 의 Column 과 연결되어 있는 것을 알 수 있다. 예를 들어, [직원번호] 열의 [link_data]는 [DS_EMP_INFO]에 있는 [emp_no] Column 과 연결되어 있다.



이벤트 처리

트랜잭션 발생 버튼 이벤트 처리

사용자가 [조회] 버튼을 클릭할 경우, 업무 서버와의 데이터 트랜잭션이 발생한다. 이벤트를 처리할 함수를 생성하기 위해 [조회]버튼 속성에서 on_mouseup 항목을 클릭하여 다음과 같이 함수를 생성한다.



구현할 스크립트는 다음을 참고한다.

```
1  function btnSelect_on_mouseup()
2  {
3      screen.requestsubmit("TR_SELECT_EMP", true);
4  }
```

화면에 설정된 xTranMapID 를 기준으로 서버와 데이터를 송수신한다.

화면과 서버가 데이터를 주고받기 위해서는 screen 오브젝트의 requestsubmit API 를 사용하여 트랜잭션을 발생시키며, screen.requestsubmit(strTranMapID, bProcAsync)의 파라미터는 다음 표를 참고 한다.

파라미터	설명
strTranMapID	트랜잭션을 발생시킬 트랜잭션 ID 를 지정한다. 트랜잭션 ID 는 [xTranMapInfo]에서 정의됨 트랜잭션 ID 는 업무 서버의 URL 정보와 송수신될 데이터 셋의 정보를 구별함
bProcAsync	트랜잭션에 대한 통신 처리는 Sync 방식과 Async 방식으로 처리할 것을 지정함 true : 트랜잭션이 Async 방식으로 처리되고, on_submitcomplete 함수가 콜백 처리됨 false : 트랜잭션이 Sync 방식으로 처리되고, on_submitcomplete 함수가 콜백 처리되지 않음

Sync 방식과 Async 방식의 차이

트랜잭션 발생시에 처리 방식인 Sync 방식과 Async 방식의 차이는 아래의 표와 같다.

항목	Sync 방식	Async 방식
함수 리턴 시점	업무 서버로 데이터를 송신하고, 업무 서버로부터 데이터를 수신한 후, 수신 데이터에 대한 처리가 완료되면 함수 리턴	업무 서버로 데이터가 송신 완료되면 함수 리턴
트랜잭션 완료	requestsubmit 함수 호출이 리턴되면 거래 처리 완료	screen_on_submitcomplete 함수가 호출되면 거래 처리 완료
트랜잭션 처리 결과 정보 확인	screen 오브젝트의 특정함수를 호출하여 결과 확인	screen_on_submitcomplete 함수의 파라미터로 확인
특징	트랜잭션이 완료되기 전에 사용자는 아무런 동작을 수행할 수 없음	트랜잭션을 완료되기 전에 사용자가 다른 동작이나 또 다른 트랜잭션을 발생시킬 수 있음

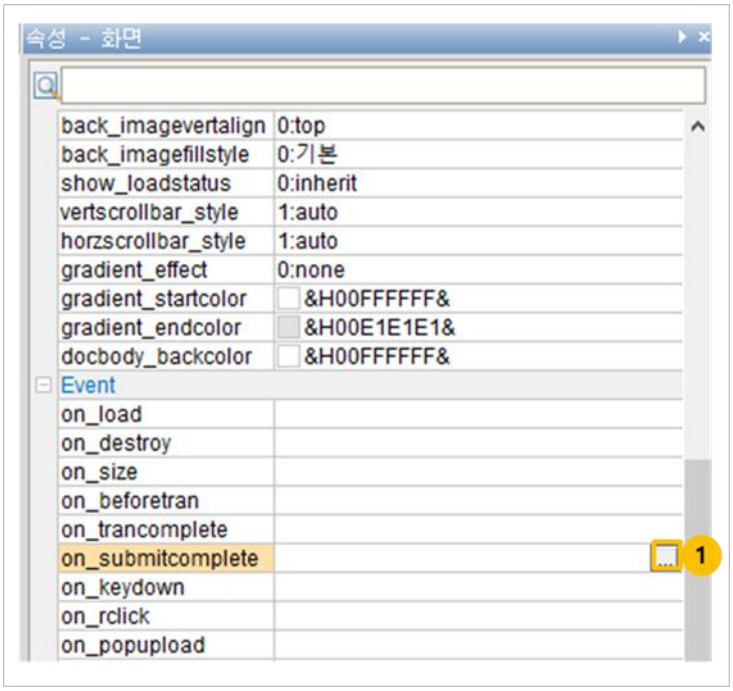
트랜잭션 완료 버튼 이벤트 처리

트랜잭션이 완료되면 화면의 on_submitcomplete 함수가 xFrame-Viewer 에 의해서 자동으로 호출된다.

트랜잭션 완료 시에 자동으로 호출되는 screen_on_submitcomplete 함수의 역할은 트랜잭션의 정상 처리 유무에 대한 정보를 전달하기 위한 것이며, 함수의 파라미터 값은 아래의 표와 같다.

파라미터	설명
mapid	처리 완료된 트랜잭션의 ID 로 툴에서 설정한 xTranMapID 값에 해당함
result	처리 완료에 대한 정보로 '1'인 경우에는 정상이고, 그 이외의 경우에는 장애임
recv_userheader	업무 서버 시스템에서 setUserHeader 함수를 통해서 지정한 값
recv_code	업무 서버 시스템에서 setErrorMessage 함수를 통해서 지정한 코드 값
recv_msg	업무 서버 시스템에서 setErrorMessage 함수를 통해서 지정한 메시지 값

본 예제에서는 트랜잭션이 완료되었을 경우, 아래의 그림과 같이 해당 정보를 개발콘솔에 표시하도록 개발할 것이다. 화면에서 on_submitcomplete 이벤트를 선택한다.



트랜잭션 완료 처리시에 호출되는 실제 함수 소스는 아래와 같다.

```
1   function screen_on_submitcomplete(mapid, result, recv_userheader,
2                                     recv_code, recv_msg)
3   {
4       // 처리 결과를 콘솔에 출력
5       factory.consoleprint("mapid = " + mapid);
6       factory.consoleprint("result = " + result);
7       factory.consoleprint("recv_userheader = " + recv_userheader);
8       factory.consoleprint("recv_code = " + recv_code);
9       factory.consoleprint("recv_msg = " + recv_msg);
10  }
```

서버 개발

예제 구현

직원 정보 조회 처리 개발

WebContent₩jsp 디렉토리 아래 SelectEmp.jsp 를 생성하고, 다음 내용을 파일에 복사한다.

라인 27 ~ 30 은 예제 테이블이 생성된 DB 환경에 알맞게 수정한다.

```
1  <%@ page import="java.sql.DriverManager" %>
2  <%@ page import="java.sql.Connection" %>
3  <%@ page import="java.sql.PreparedStatement" %>
4  <%@ page import="java.sql.ResultSet" %>
5  <%@ page import="xdataset5.XDataSet5" %>
6
7  <%
8  Connection conn = null;          // DB Connection Object
9  PreparedStatement pstmt = null; // JDBC PreparedStatement Object
10 ResultSet rs = null;             // Query Result Set Object
11 XDataSet5 xDataSet5 = null;      // XFrame XDataSet Object
12
13 try {
14     // Clear out's buffer
15     out.clearBuffer();
16
17     // Create XDataSet object
18     xDataSet5 = new XDataSet5(request, response);
19
20     System.out.println("ScreenNo : " + xDataSet5.getScreenNo());
21     System.out.println("IP : " + xDataSet5.getTerminalIpAddress());
22     System.out.println("Map ID : " + xDataSet5.getTransactionMapId());
23     System.out.println("User Header : " + xDataSet5.getUserHeader());
24
25     // For MySQL
27     String driverClass = "com.mysql.jdbc.Driver";
28     String dbUrl = "jdbc:mysql://localhost:3306/xframe_data";
29     String dbUserId = "xframe";
30     String dbUserPasswd = "xframe";
31
32     // Load JDBC Driver and connect to database
33     Class.forName(driverClass);
34     conn = DriverManager.getConnection(dbUrl, dbUserId, dbUserPasswd);
35
36     // Create a select query for EMP_INFO table
37     String sQuery =
38        "SELECT " +
39        "     emp_name, emp_no, emp_age, mail_addr " +
40        "FROM " +
41        "     EMP_INFO " +
42        "ORDER BY " +
```

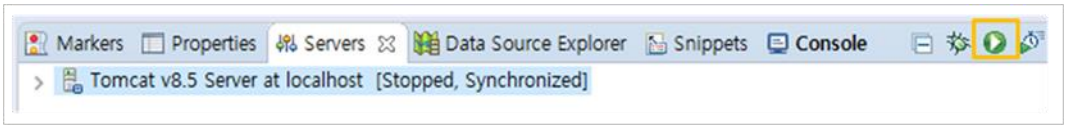
```
43          "     emp_name";
44
45      // Creae a statement and execute a statement
46      pstmt = conn.prepareStatement(sQuery);
47      rs = pstmt.executeQuery();
48
49      // Loop a select result records
50      for(int nRow = 0; rs.next(); nRow++) {
51          // Get data from resultset
52          String emp_name = rs.getString("emp_name");
53          if(emp_name == null) emp_name = "";
54
55          String emp_no = rs.getString("emp_no");
56          if(emp_no == null) emp_no = "";
57
58          String emp_age = rs.getString("emp_age");
59          if(emp_age == null) emp_age = "";
60
61          String mail_addr = rs.getString("mail_addr");
62          if(mail_addr == null) mail_addr = "";
63
64          // Set output data to XDataSet object
65          xDataSet5.setData("DS_EMP_INFO", "emp_name", nRow, emp_name);
66          xDataSet5.setData("DS_EMP_INFO", "emp_no", nRow, emp_no);
67          xDataSet5.setData("DS_EMP_INFO", "emp_age", nRow, emp_age);
68          xDataSet5.setData("DS_EMP_INFO", "mail_addr", nRow, mail_addr);
69      }
70
71      // Return data to XFrame
72      xDataSet5.returnData();
73  }
74  catch(Exception e) {
75      try {
76          if(xDataSet5 != null) {
77              // Set error code and error message
78              xDataSet5.setErrorMessage("XDATASET_ERROR", e.getMessage());
79              System.out.println(e.getMessage());
80              xDataSet5.returnData();
81          }
82      }
83      catch(Exception ex) {
84          System.out.println("Exceptino Msg = " + ex.toString());
85      }
86  }
87  finally {
88      // Release a database resources
89      if(rs != null) { try { rs.close(); } catch(Exception ignore) {} }
90      if(pstmt != null) { try { pstmt.close(); } catch(Exception ignore) {} }
91      if(conn != null) { try { conn.close(); } catch(Exception ignore) {} }
92  }
93  %>
```

예제 실행

구현된 [직원정보조회] 예제를 확인해 본다.

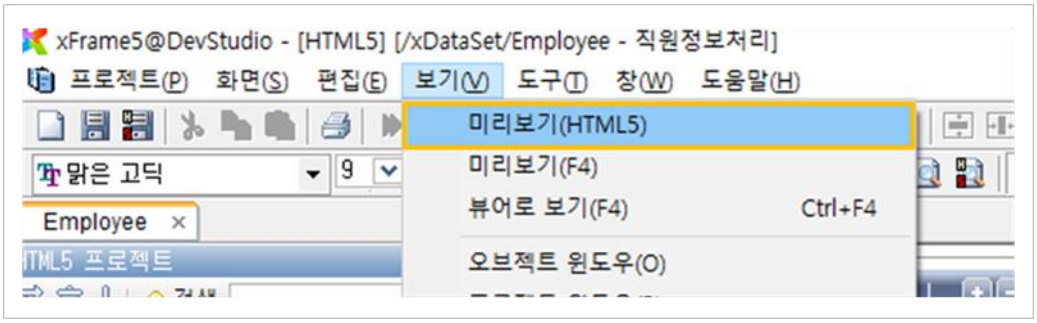
톰캣 서버 시작

톰캣 시작 버튼을 클릭하여 서버를 시작한다.



예제 확인

[보기] – [미리보기(HTML5)]를 클릭하여 구현한 예제를 실행한다.



[조회] 버튼을 클릭하여 직원정보가 정상적으로 조회되는 지 확인한다.

직원번호	직원이름	직원나이	메일주소
100112	곽경환	42	khkwak@gmail.com
100223	김경환	43	khkim@gmail.com
100212	김용상	36	yskim@gmail.com
100215	김용완	54	ywkim@gmail.com
100133	김용진	31	yjkim@gmail.com
100111	김우섭	49	wskim@gmail.com
100325	김종욱	37	jwkim@gmail.com
100117	김종인	47	jikim@gmail.com
100218	김지태	33	jtkim@gmail.com
100319	방인홍	36	ihbang@gmail.com
100118	유동균	40	dkyou@gmail.com
100312	유승한	29	shyou@gmail.com
100124	임기조	35	kjlim@gmail.com
100115	전시교	50	skjeon@gmail.com
100226	정학길	28	hkjung@gmail.com
100221	홍성일	50	sihong@gmail.com