

NE > < ACRO

PLATFORM
Education for SeedIT

Concept & 5 Demo

Nexacro N

1. 넥사크로 앱을 실행하기 위해서는 실행 환경이 필요합니다. 별도의 실행 환경을 구축하지 않아도 넥스크로 스튜디오에서 Windows NRE, 에뮬레이터, 로컬 웹 서버로 테스트해볼 수 있습니다.
2. 넥사크로 앱 동작 뿐 아니라 실제 데이터를 조회하고 저장하는 데이터 트랜잭션까지 테스트해보고자 한다면 WAS(web Application Server)를 설정해야 합니다. 이를 위해 WAS 설치를 위한 JDK 설치를 위한 JDK 설치와 아파치 톰캣 설정에 대해 설명합니다.

1.로컬 웹서버와 WAS의 차이점

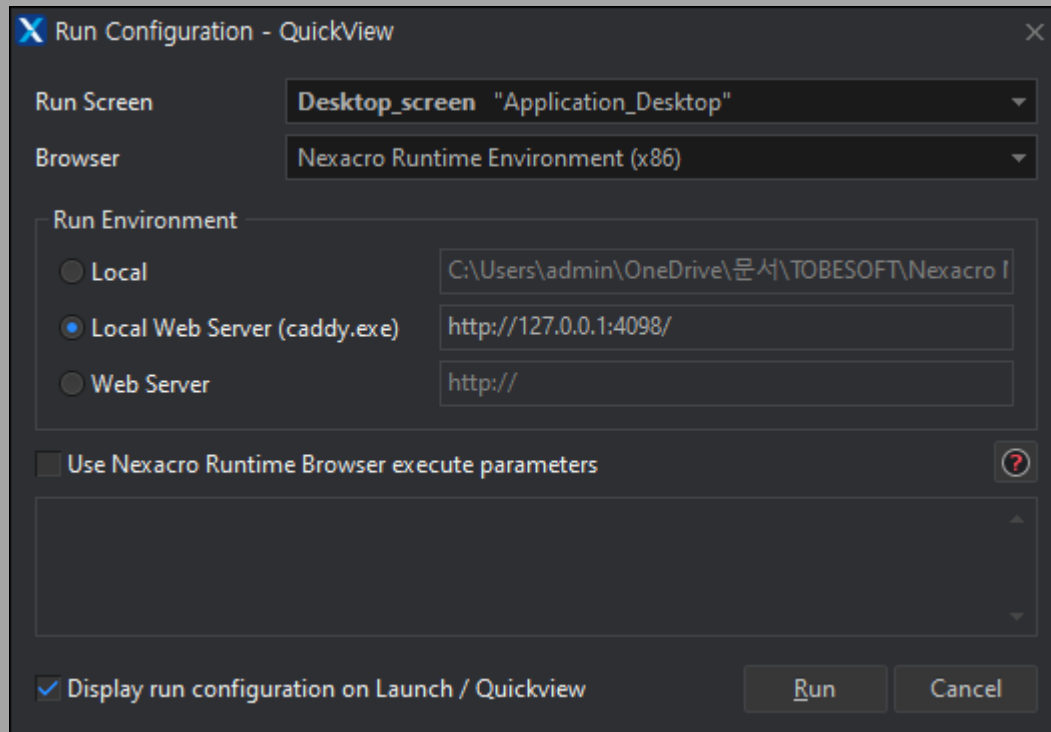
	로컬 웹서버	Apache Tomcat
설치파일	없음 (넥사크로 스튜디오 설치에 포함)	설치 파일 제공 (바이너리 소스로 받아서 직접 배치 파일을 실행시킬 수도 있음)
JDK 설치	필요 없음	JDK 설치 후 사용 가능
Context 설정	없음 (Generate 폴더가 웹서버 root로 지정)	정해진 절차에 따라 컨테스트를 추가하거나 war 파일을 디플로이해서 컨텍스트 생성
X-API 지원	사용할 수 없음	사용 가능 (별도 설정)

-로컬 웹서버는 넥사크로 스튜디오를 실행하는 중에만 사용할 수 있습니다. 넥사크로 스튜디오를 실행한지 않고 앱을 테스트하려면 직접 웹서버나 WAS를 설치해야 합니다.

-Nexacro N에서 X-API를 설치해서 외부 DB연동 작업을 진행하겠습니다.

2.로컬 웹서버 사용하기

-넥사크로 스튜디오 설치 후 기본 옵션은 로컬 웹서버를 사용하도록 설정됩니다. 로컬 웹서버를 사용하는 경우에는 별도의 설정 없이 넥사크로 스튜디오에서 "Launch" 또는 "Quick View" 실행 시 웹 브라우저를 선택한 경우에는 로컬 웹서버가 자동으로 실행되고 넥사크로 스튜디오를 종료하면 자동으로 로컬 웹서버도 종료됩니다.



3.WAS(Web Application Server) 사용하기

※ X-API 사용 시 Jakarta EE 스펙으로 구현된 WAS는 지원되지 않습니다.

WAS는 톰캣(Tomcat)이나 제티(Jetty), 윈스톤(Winstone)과 같이 무료로 제공된 것을 사용하거나 상용 제품을 사용할 수 있습니다. 여기에서는 간단하게 사용할 수 있는 톰캣을 설치하는 기준으로 설명하겠습니다.

톰캣은 자바기반으로 만들어졌기 때문에 실행하기 위해서는 JDK(Java SE Development Kit)환경이 설정되어 있어야 합니다.
(여기서는 자바 11 버전을 설치해서 사용하겠습니다. -자바 설치 생략)

아파치 톰캣(Apache Tomcat)

넥사크로 앱을 웹브라우저에서 동작하게 하려면 웹서버 환경을 만들어주어야 합니다. 아파치 톰캣은 간단한 설치만으로 이런 환경을 만들 수 있습니다.
-톰캣 설치 생략(컨텍스트 설정부터 진행합니다.)

웹브라우저에서 로컬 주소에 설정된 URL값을 입력해 정상 설치 여부를 확인합니다.

<http://localhost:8087>(톰캣의 http 기본포트는 8080인데 Oracle http 포트와 충돌로 인해서 톰캣포트를 8087로 변경해서 설치하였습니다.)

4.컨텍스트 설정

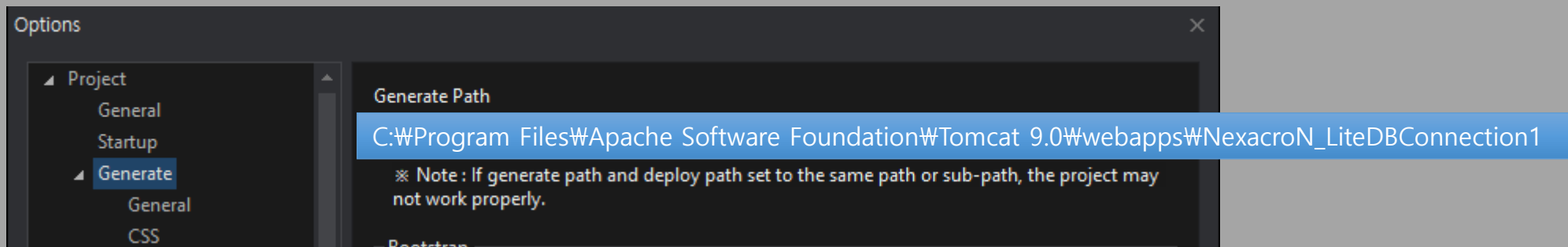
톰캣에서 원하는 정보를 보여주려면 컨텍스트를 추가해주어야 합니다. 컨텍스트를 추가하는 방법은 몇 가지가 있지만, 이번 장에서는 컨텍스트 파일을 생성하고 임의의 폴더를 설정하는 방법을 살펴보겠습니다.

컨텍스트 파일 생성

컨텍스트 파일은 넥사크로 스튜디오에서 Generate Path를 지정하고 해당 경로를 톰캣과 연결해주는 역할을 합니다. 톰캣이 설치된 경로 아래에 지정된 폴더에 XML 파일로 만든 컨텍스트 파일을 추가해줍니다.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\conf\Catalina\localhost\NexacroN_LiteDBConnection1.xml

XML 파일을 아래와 같이 생성합니다. path 항목에는 URL 주소에 포함될 문자열을 입력하고 docBase 항목에는 Generate Path로 지정된 경로를 입력합니다. 아래와 같이 path 항목을 지정한 경우에 연결되는 URL은 [http://localhost:8087/NexacroN_LiteDBConnection1/...](http://localhost:8087/NexacroN_LiteDBConnection1/) 형식이 됩니다. 생성하는 파일명은 path 항목에 이름과 같아야 합니다. 아래 예제에서는 NexacroN_LiteDBConnection1.xml 파일을 생성합니다.



4.NexacroN_LiteDBConnection1.xml 파일 생성

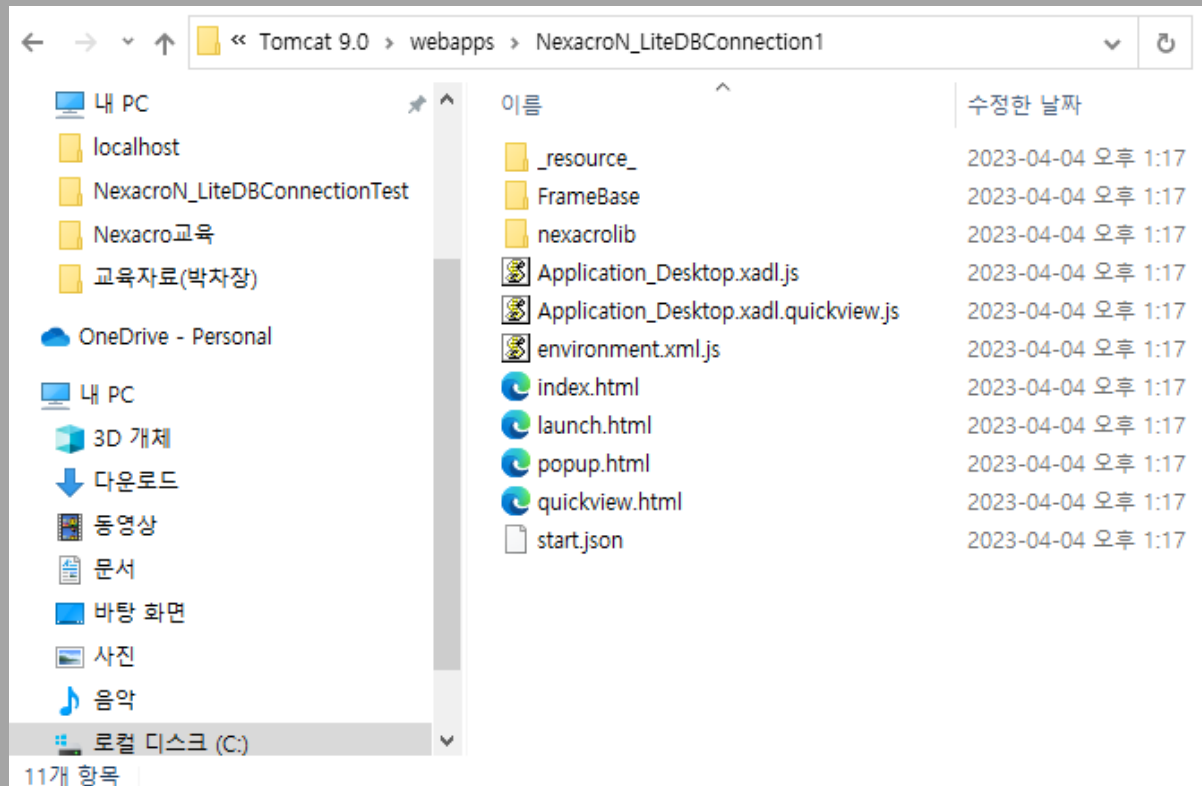
Generate Path 설정후 OK버튼을 클릭하면 Nexacro Studio 가 Refresh 되면서 아래화면과 같이 WAS(톰캣) 경로상에 파일들이 업로드되는것을 확인할 수 있습니다.

NexacroN_LiteDBConnection1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Context path="/NexacroN_LiteDBConnection1" docBase="C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\NexacroN_LiteDBConnection1"
    debug="0" privileged="true" reloadable="true">
    <Logger className="org.apache.catalina.logger.FileLogger"
        directory="logs" prefix="localhost_log." suffix=".txt"
        timestamp="true"
    />
</Context>
```

5.컨텍스트 설정

Generate Path 설정후 OK버튼을 클릭하면 Nexacro Studio 가 Refresh 되면서 아래화면과 같이 WAS(톰캣) 경로상에 파일들이 업로드되는것을 확인할 수 있습니다.



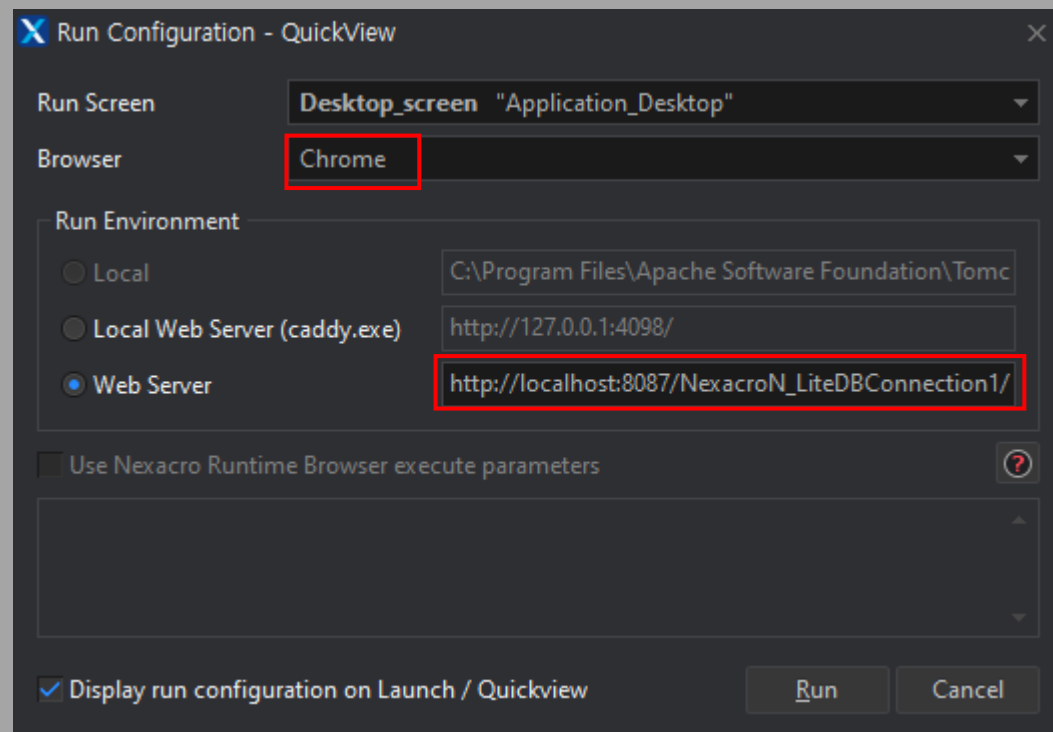
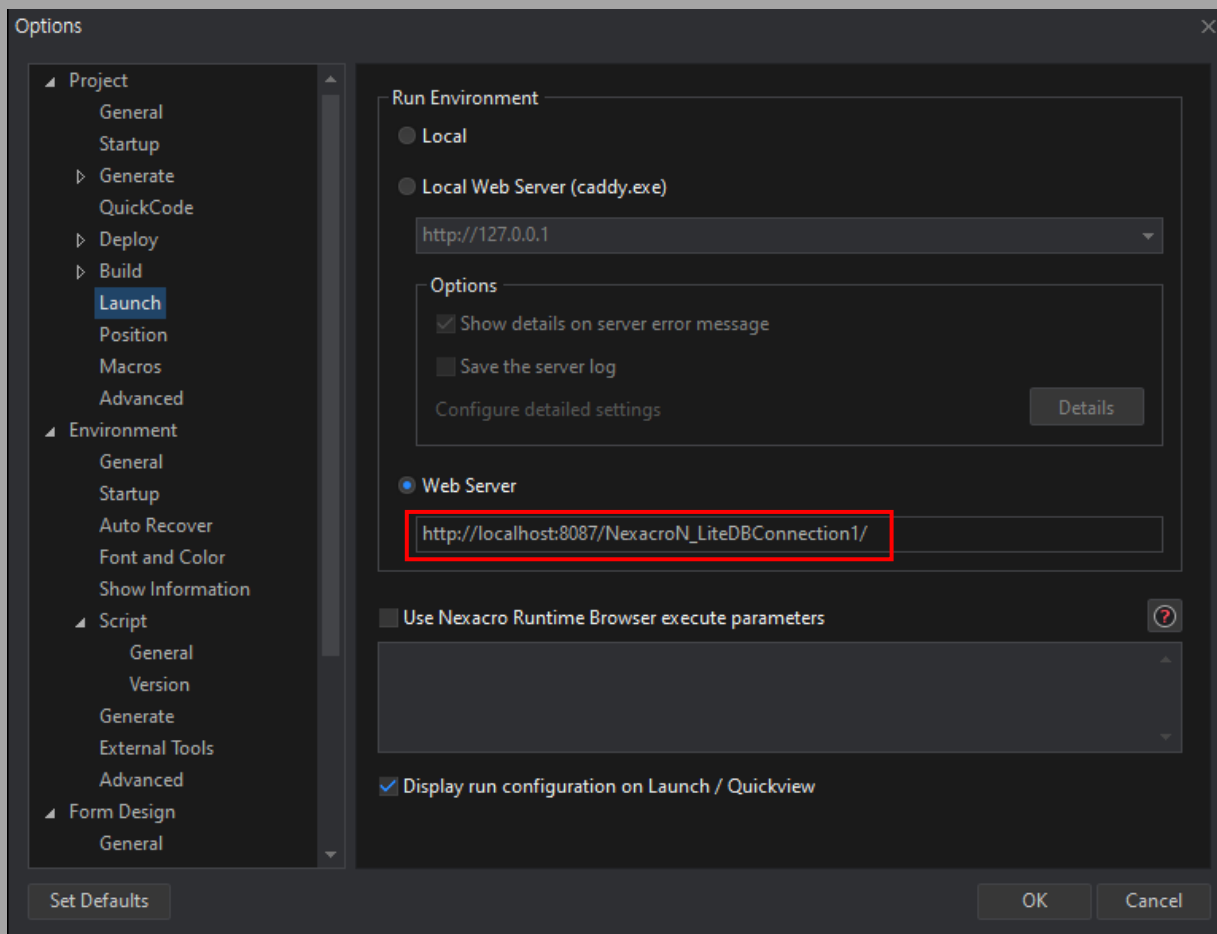
Nexacro N-DB연동

Nexacro N 외부 DB연동

6. 넥사크로 앱 확인

로컬 웹서버가 아닌 다른 웹서버 또는 WAS를 사용하는 경우에는 Web Server 설정을 해주어야 합니다.

메뉴[Tools > Options > Project > Launch > Web Server] 항목에 URL을 설정해주면 이후 실행 시 해당 경로 기준으로 실행이 됩니다.



6. 넥사크로 앱 확인

웹 브라우저의 주소표시줄에서 확인해보면 Project > Launch > Web Server에 입력한 것과 같은것을 확인할 수 있습니다.

The screenshot shows a web browser window with the address bar displaying `localhost:8087/NexacroN_LiteDBConnection1/launch.html?screenid=Desktop_screen`. The application interface includes a menu on the left with items like Main_Menu1, Sub_Menu1, Form_Hello, Form_Emp, Sub_Menu2, Form_Emp1, Main_Menu2, Sub_Menu1, Form1, and Form2. The main content area is titled 'Employees' and contains a table with columns EMPL_ID, FULL_NAME, DEPT_CD, POS_CD, and HIRE_DATE. The table lists employees: AA001 (Olivia), AA002 (John), BB001 (Jackson), BB002 (Maia), CC001 (Adam), and DD001 (Tyler). To the right of the table is a form for editing employee details, including fields for Name, Emp ID, Department, Position, Gender, Married, Hire Date, Annual Salary, and Memo. The form is currently displaying details for Olivia (AA-001), Accountant Team, Officer, Division Manager, Female, Married, 2010-10-03, 83,000, and Memo: ivory.

EMPL_ID	FULL_NAME	DEPT_CD	POS_CD	HIRE_DATE
AA001	Olivia	01	02	2010-10-03
AA002	John	02	02	2005-10-11
BB001	Jackson	02	03	2007-02-06
BB002	Maia	04	01	2009-05-12
CC001	Adam	04	04	2001-01-09
DD001	Tyler	04	03	2006-09-07

Employees

retrieve Add Del Save

Department find Gender ☐ All ☐ Male ☐ Female reset

Name

Emp ID

Department

Position

Gender ☐ male ☒ Female

Married ☒

Hire Date

Annual Salary

Memo

7.X-API

X-API 배포 파일

- 넥사크로 X-API 라이브러리는 데이터 처리를 위한 서비스 구현 시 필요한 기능을 라이브러리 형태로 구현해 제공하고 있습니다. 제공하는 파일은 아래와 같습니다.
 - nexacro-xapi-java-1.0.1.jar(X-API 라이브러리 파일)
 - json-simple-1.1.1.jar(json 통신 파일)
 - commons-logging-1.1.1.jar (<http://commons.apache.org/proper/commons-logging/>)
 - NexacroN_server_license.xml(라이선스 파일)

Nexacro N-DB연동

Nexacro N 외부 DB연동

7.X-API

X-API 배포 파일은 기술지원사이트에서 내려받을 수 있습니다.

<http://support.tobesoft.co.kr> > PRODUCT > Nexacro N > Download > Server [API, XENI]

개발용 라이선스 파일은 LICENSE 메뉴 아래 [개발용라이선스] 링크에서 신청할 수 있습니다.

※ 여기서는 체험판 라이선스파일을 이용하여 작업하겠습니다.(NexacroN_server_license.xml)

The screenshot shows the TOBESOFT support website. The top navigation bar includes 'TOBESOFT', 'PRODUCT' (highlighted with a red box), 'Q&A', 'EDUCATION', and 'LICENSE'. The breadcrumb trail is 'Home > Product > Nexacro N > Download'. Below this, there are three tabs: 'Developer', 'Server [API,XENI]' (highlighted with a red box), and 'Etc.'. A list of notices is displayed, followed by the section 'Nexacro N Excel Export/Import' with a download link for 'NexacroN_XENI_JAVA_20230313(1.2.22)_1.zip'. At the bottom, the 'Nexacro N X-API' section is highlighted with a red box, showing a download link for 'NexacroN_XAPI_JAVA_20230308(1.0.1)_1.zip'.

TOBESOFT

PRODUCT Q&A EDUCATION LICENSE

Home > Product > Nexacro N > Download

Developer Server [API,XENI] Etc.

- 유지보수 미계약 고객(사)의 경우 패치 이후 발생하는 문제와 관련한 일체의 책임을 지지 않습니다
- 유지보수 미계약 고객(사)의 경우 관련 영업대표를 통해 유지보수를 체결해 주시기 바랍니다.
- NexacroN 시스템 요구사항 보기

Nexacro N Excel Export/Import

NexacroN_XENI_JAVA_20230313(1.2.22)_1.zip

Nexacro N X-API

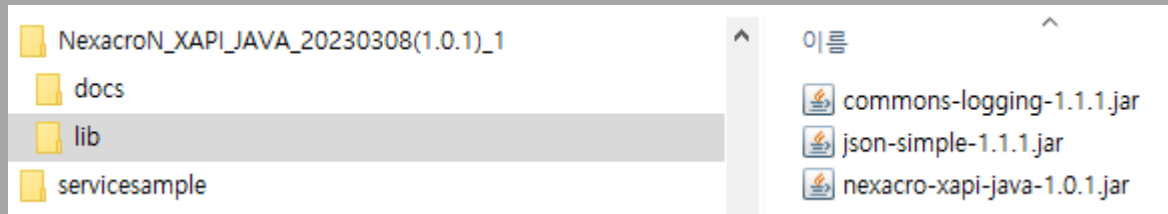
NexacroN_XAPI_JAVA_20230308(1.0.1)_1.zip

Nexacro N-DB연동

Nexacro N 외부 DB연동

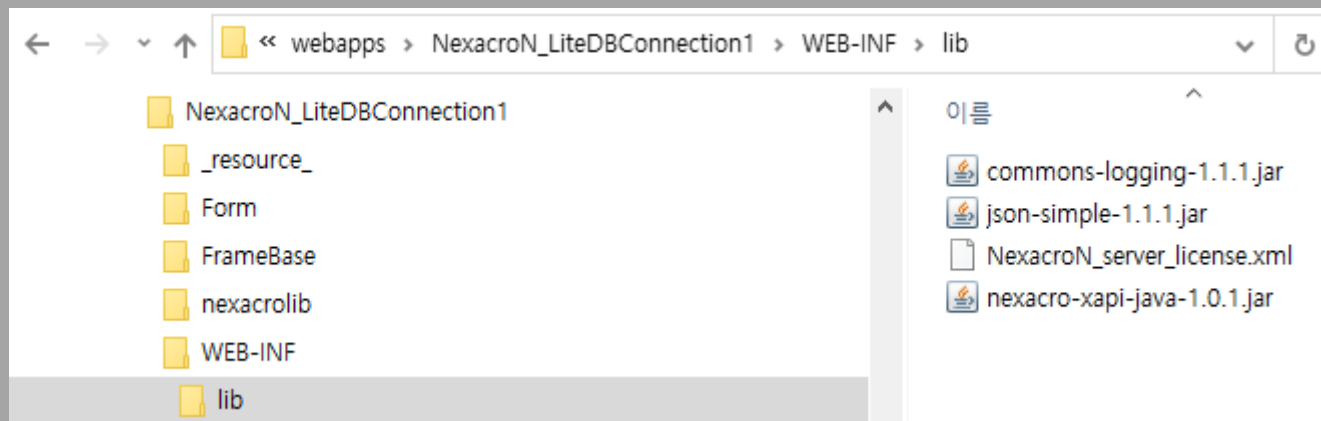
8.X-API 배포

-내려받은 X-API 파일의 압축을 해제합니다. 압축을 해제하면 docs, lib 폴더 2개가 보이는데 그 중에서 lib 폴더 안에 있는 3개의 jar 파일을 사용할 겁니다.



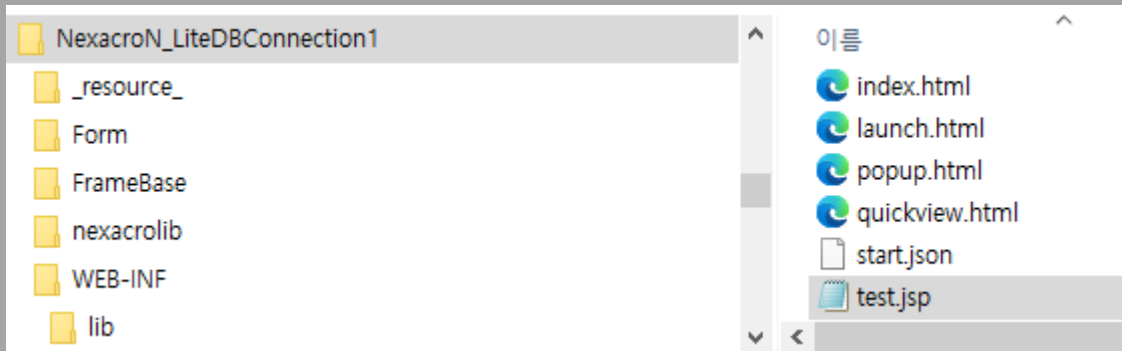
-docBase로 지정한 폴더를 생성하고 해당 폴더 안에 WEB-INF 폴더를 생성합니다. 그리고 WEB-INF 폴더 안에 lib 폴더를 생성하고 jar 파일과 라이선스 파일을 복사합니다.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\NexacroN_LiteDBConnection1\WEB-INF\lib



8.X-API 배포(X-API 배포 테스트)

-docBase로 지정한 폴더 아래에 테스트를 위한 jsp 파일을 생성합니다.



8.X-API 배포(X-API 배포 테스트)

-test.jsp 파일을 아래와 같이 작성합니다.

test.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>

<html>
  <head>
    <title>JarInfo</title>
    <style>
      * { font-family: Verdana }
    </style>
  </head>
  <body>
    <pre>
<%
new com.nexacro.java.xapi.util.JarInfo().info(out);
%>
    </pre>
  </body>
</html>
```

9. test.jsp 파일 실행하여 X-API 설치 확인하기

웹브라우저서 아래 주소로 접근했을 때 설치 정보가 정상적으로 표시되는지 확인합니다. 설치정보가 표시된다면 정상적으로 X_API가 설치된 겁니다.

http://localhost:8087/NexacroN_LiteDBConnection1/test.jsp

```
← → ↻ ⓘ localhost:8087/NexacroN_LiteDBConnection1/test.jsp
nexacroTest
--- Manifest ---
Manifest-Version: 1.0
Built-By: Server Team
Created-By: 1.4.2_19-b04 (Sun Microsystems Inc.)
Ant-Version: Apache Ant 1.7.1
Main-Class: com.nexacro.java.xapi.util.JarInfo
Built-Date: March 8 2023

Name: com.nexacro.java.xapi
Specification-Title: X-API
Implementation-Title: X-API
Specification-Vendor: TOBESOFT CO., LTD.
Specification-Version: 1.0.0
Implementation-Vendor-Id: com.nexacro
Implementation-Vendor: TOBESOFT CO., LTD.


--- Log ---
INFO Loaded license file in JAR dir: path=C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\NexacroN_LiteDBConnection1\WEB-INF\lib\NexacroN_server_license.xml

--- License ---
product.name: nexacro platform
product.version: 21
product.function: Runtime,WebBrowser
customer.name: TOBESOFT
server.cpuCoreCount: 0
server.ipAddress: 0.0.0.0
date.activation: 2023-03-01
date.term: 2
date.term.unit: month
customer.targetPlatform: Windows,WindowsRT,macOS
```


10. Sqlite JDBC Driver 설정하기

Nexacro와 Sqlite를 연동하기 위해서는 Sqlite JDBC Driver를 다운받아서 Java와 톰캣에 설정해야 사용할 수 있다.

<https://dbschema.com/jdbc-driver/Sqlite.html>

 DbSchema

BlogFeatures ▾DocsDatabasesEditionsSupport ▾Download

The Sqlite JDBC Driver

- [1. What are JDBC Drivers](#)
- [2. What is the JDBC URL](#)
- [3. Download the Sqlite JDBC Driver](#)
- [4. How to Connect using the DbSchema Sqlite Client](#)

What are JDBC Drivers?

JDBC drivers are Java library files with the extension '.jar', used by Java applications to connect to the database. Usually they are provided by the same company which developed the database software. [DbSchema](#) is an Sqlite Client which already includes the Sqlite JDBC driver. DbSchema can configure the Sqlite JDBC URL and test the connectivity.

What is the JDBC URL?

The URL is a string (text) with a specific format containing information about the host where the database is running, the port, username, database name, etc. The URL format is specific to each driver. Any wrong character in the URL may make the database connectivity fail. Therefore we recommend installing [DbSchema](#) and try to get connected, and you will find the JDBC URL in the connection dialog.

Driver Information

- Required File(s): sqlite-jdbc-xxx.jar
- Java Driver Class: org.sqlite.JDBC
- URL: jdbc:sqlite:FILE
- Website: [Sqlite](#)

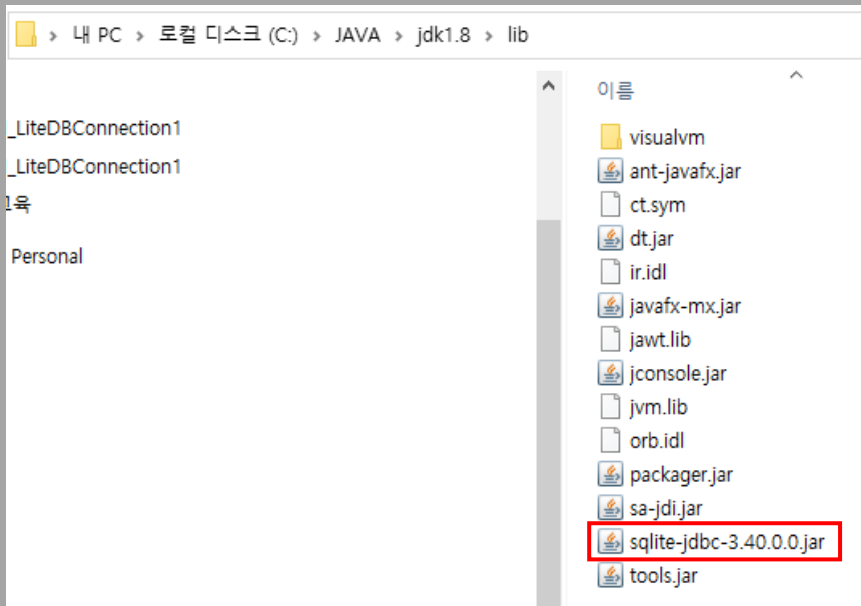
For Android developers we include in DbSchema help detailed information how to get the Sqlite database file from an Android phone.

Download Sqlite JDBC Driver

10. Sqlite JDBC Driver 설정하기-java 설정

Sqlite JDBC Driver를 다운받아서 압축을 풀면 아래와 같이 sqlite-jdbc-3.40.0.0.jar 파일을 java 설치된 경로의 lib폴더에 붙여넣는다

C:\JAVA\jdk1.8\lib\sqlite-jdbc-3.40.0.0.jar

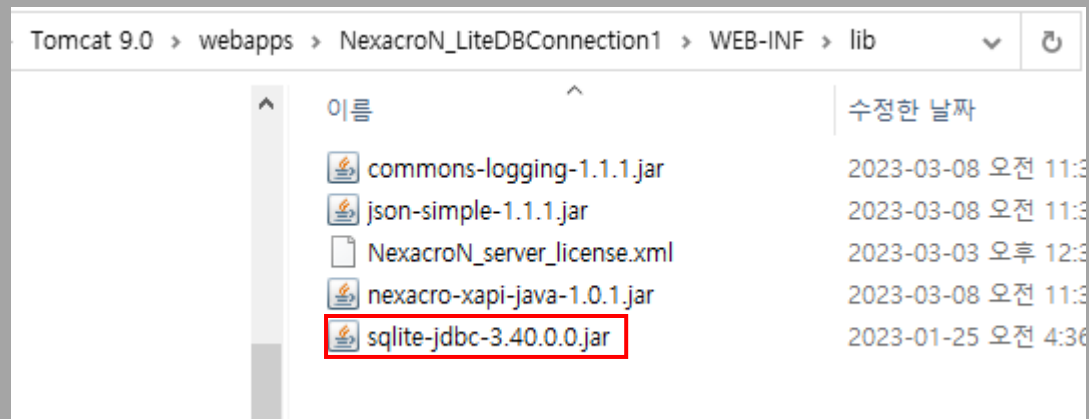


10. Sqlite JDBC Driver 설정하기-was(톰캣) 설정

Sqlite JDBC Driver(sqlite-jdbc-3.40.0.0.jar) 파일을 톰캣의 WEB-INF > lib폴더에 붙여넣는다

Sqlite JDBC Driver(sqlite-jdbc-3.40.0.0.jar) 파일을 자바와 톰캣에 각자 설정하여 연동한다.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\NexacroN_LiteDBConnection1\WEB-INF\lib\sqlite-jdbc-3.40.0.0.jar

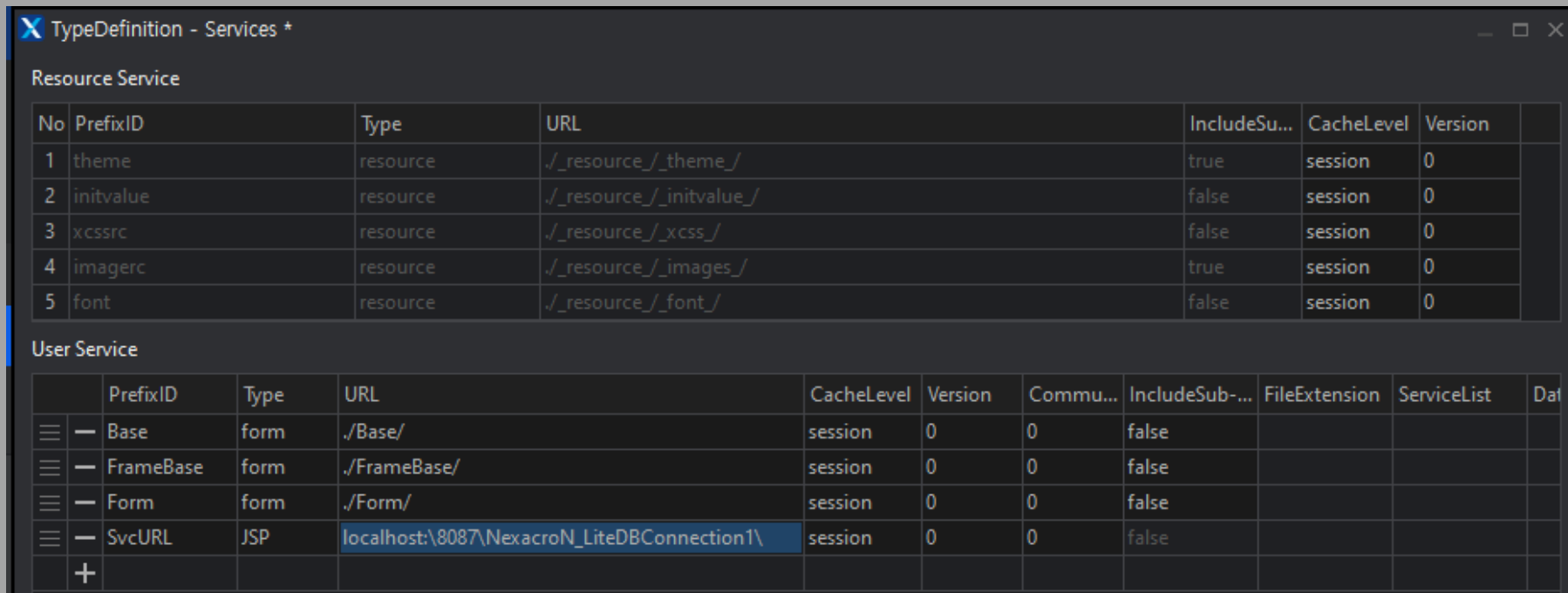


11. 서버 경로 설정

톰캣 서버에 JSP 서비스 파일이 위치 할 디렉토리를 다음과 같이 구성합니다.

위에서 지정한 서버 경로를 화면에서 호출 하기 위하여 넥사크로 스튜디오에서 지정 합니다.

Project Explorer > TypeDefinition 의 Services 항목을 더블클릭하여 Services Edit창을 엽니다.
+버튼을 이용하여 Service를 다음과 같이 등록 합니다.



Resource Service

No	PrefixID	Type	URL	IncludeSu...	CacheLevel	Version
1	theme	resource	./_resource/_theme/	true	session	0
2	initvalue	resource	./_resource/_initvalue/	false	session	0
3	xcsrc	resource	./_resource/_xcsrc/	false	session	0
4	imagerc	resource	./_resource/_images/	true	session	0
5	font	resource	./_resource/_font/	false	session	0

User Service

	PrefixID	Type	URL	CacheLevel	Version	Commu...	IncludeSub...	FileExtension	ServiceList	Dat
≡	Base	form	./Base/	session	0	0	false			
≡	FrameBase	form	./FrameBase/	session	0	0	false			
≡	Form	form	./Form/	session	0	0	false			
≡	SvcURL	JSP	localhost:\8087\NexacroN_LiteDBConnection1\	session	0	0	false			
+										

User Service항목 하단에 SvcURL에 지정된 경로에 json 통신에 필요한 jsp파일들을 위치시켜서 NexacroN과 연동할 수 있습니다.

12. 조회 처리

화면 조회 버튼 스크립트

Retrieve 버튼의 onclick 이벤트에 'btn_retrieve onclick'으로 함수를 생성 후, 다음과 같이 스크립트를 작성합니다.

```
this.btn_retrieve onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    var id = "svcSelect";
    var url = "SvcURL::select_emp.jsp";//SvcURL에 설정된 경로에 jsp파일 지정
    var reqDs = "";
    var respDs = "ds_emp=out_emp";//Form_Emp 폼에 ds_emp에 transaction으로 가져온 out_emp dataset을 입력한다.
    var args = "";
    var callback = "fn_callback";//비동기 통신 결과를 받을 콜백 함수 지정

    this.transaction(id, url, reqDs, respDs, args, callback);
    //this.grd_list.createFormat();
};
this.fn_callback = function(svcID, errCD, errMsg){
    if(errCD == 0){//errCD값이 0이면 transaction 성공
        var rowcount = this.ds_emp.rowcount;
        this.alert(rowcount + " numbers of data have been found.");
    }else{//errCD값이 0이아니면 transaction 실패
        this.alert("Error: " + errMsg + ", errCD:" + errCD + ", svcID:" + svcID);
        return;
    }
    if(svcID == "svcSelect"){
        this.alert("Retrieve Success!!!");
    }
}
```

12. 조회 처리

transaction

데이터 처리를 위해 서비스를 호출하는 메소드로 기본 비동기 방식으로 동작 합니다.

-Syntax

```
this.transaction(strSVCID, strURL, strInDatasets, strOutDatasets, strArgument, strCallbackFunc [,bAsnc [,nDataType [,bCompress]]])
```

구분	설명
strSvcID	transaction을 구분하기 위한 ID입니다.
strURL	transaction을 요청할 서버 페이지 주소 입니다.
strInDatasets	서버로 보낼 Dataset 입니다.
strOutDatasets	서버에서 보내는 데이터를 받을 Dataset입니다.
strArgument	서버로 보낼 파라미터(변수) 입니다.
strCallbackFunc	서버의 처리 결과를 받을 콜백함수를 문자열로 지정합니다.
[bAsync]	비동기 통신 여부를 설정합니다. true : 비동기 통신 / false : 동기 통신
[nDataType]	전송할 데이터의 형태를 설정합니다. 0 : XML / 1 : Binary / 2 : SSV
[bCompress]	서버 통신시 데이터의 압축 여부를 지정합니다. Default : false

12. 조회 처리

조회 서비스 페이지(JSP) 작성-1

조회 서비스를 'select_emp.jsp'로 생성하고 다음과 같이 작성 합니다.

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page language="java"%>
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.sql.*"%>
<%@ page import="org.apache.commons.logging.*"%>
<%@ page import="com.nexacro.java.xapi.tx.*" %>
<%@ page import="com.nexacro.java.xapi.data.*" %>

<%
//PlatformData(비동기 통신으로 transaction callback 함수에 데이터 전송시 데이터를 담아서 전송할때 사용)
PlatformData pData = new PlatformData();

int nErrorCode = 0;
String strErrorMsg = "START";
try{
    /***** JDBC Connection *****/

    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
```

12. 조회 처리

조회 서비스 페이지(JSP) 작성-2

```
try{
    Class.forName("org.sqlite.JDBC");
    conn = DriverManager.getConnection("jdbc:sqlite:C:\\WWW\\JAVAWWW\\sqlite_LiteDB\\WWW\\nexacro_sample_db.sqlite");
    stmt = conn.createStatement();
    String sql = "select * from tb_emp";

    rs = stmt.executeQuery(sql);
    //DataSet 생성 후 DataType 설정
    DataSet ds = new DataSet("out_emp");
    ds.addColumn("EMPL_ID", DataTypes.STRING, 10);
    ds.addColumn("FULL_NAME", DataTypes.STRING, 50);
    ds.addColumn("DEPT_CD", DataTypes.STRING, 10);
    ds.addColumn("POS_CD", DataTypes.STRING, 10);
    ds.addColumn("GENDER", DataTypes.STRING, 10);
    ds.addColumn("HIRE_DATE", DataTypes.DATE, 10);
    ds.addColumn("MARRIED", DataTypes.STRING, 10);
    ds.addColumn("SALARY", DataTypes.INT, 10);
    ds.addColumn("MEMO", DataTypes.STRING, 10);
```


12. 조회 처리

조회 서비스 페이지(JSP) 작성-3

```
//DataSet에 tb_emp 테이블에서 조회한 값 입력하기
int row = 0;
while(rs.next()){
    row = ds.newRow();
    ds.set(row, "EMPL_ID", rs.getString("EMPL_ID"));
    ds.set(row, "FULL_NAME", rs.getString("FULL_NAME"));
    ds.set(row, "DEPT_CD", rs.getString("DEPT_CD"));
    ds.set(row, "POS_CD", rs.getString("POS_CD"));
    ds.set(row, "GENDER", rs.getString("GENDER"));
    ds.set(row, "HIRE_DATE", rs.getString("HIRE_DATE"));
    ds.set(row, "MARRIED", rs.getString("MARRIED"));
    ds.set(row, "SALARY", rs.getInt("SALARY"));
    ds.set(row, "MEMO", rs.getString("MEMO"));
}
// #1 dataset -> PlatformData
pData.addDataSet(ds);//pData 객체에 DataSet ds를 입력
nErrorCode = 0;
strErrorMsg = "SUCC";
}catch(ClassNotFoundException e){
    e.printStackTrace();
    nErrorCode = -1;
    strErrorMsg = e.getMessage();
}catch(SQLException e){
    e.printStackTrace();
    nErrorCode = -1;
    strErrorMsg = e.getMessage();
}
```

12. 조회 처리

조회 서비스 페이지(JSP) 작성-4

```
        /***** JDBC Close *****/
        if( stmt != null ) try { stmt.close(); } catch(Exception e) {nErrorCode = -1; strErrorMsg = e.getMessage();}
        if ( conn != null ) try {conn.close(); } catch(Exception e) {nErrorCode = -1; strErrorMsg = e.getMessage();}
    }catch(Throwable th){
        nErrorCode = -1;
        strErrorMsg = th.getMessage();
    }
    //에러코드와 에러메시지를 VariableList에 담아 클라이언트에 전송
    VariableList varList = pData.getVariableList();
    varList.add("ErrorCode", nErrorCode);
    varList.add("ErrorMsg", strErrorMsg);

    HttpPlatformResponse res = new HttpPlatformResponse(response, PlatformType.CONTENT_TYPE_XML, "UTF-8");
    res.setData(pData);

    // Send data
    res.sendData();
    %>
```

13. 조회 처리 결과 확인

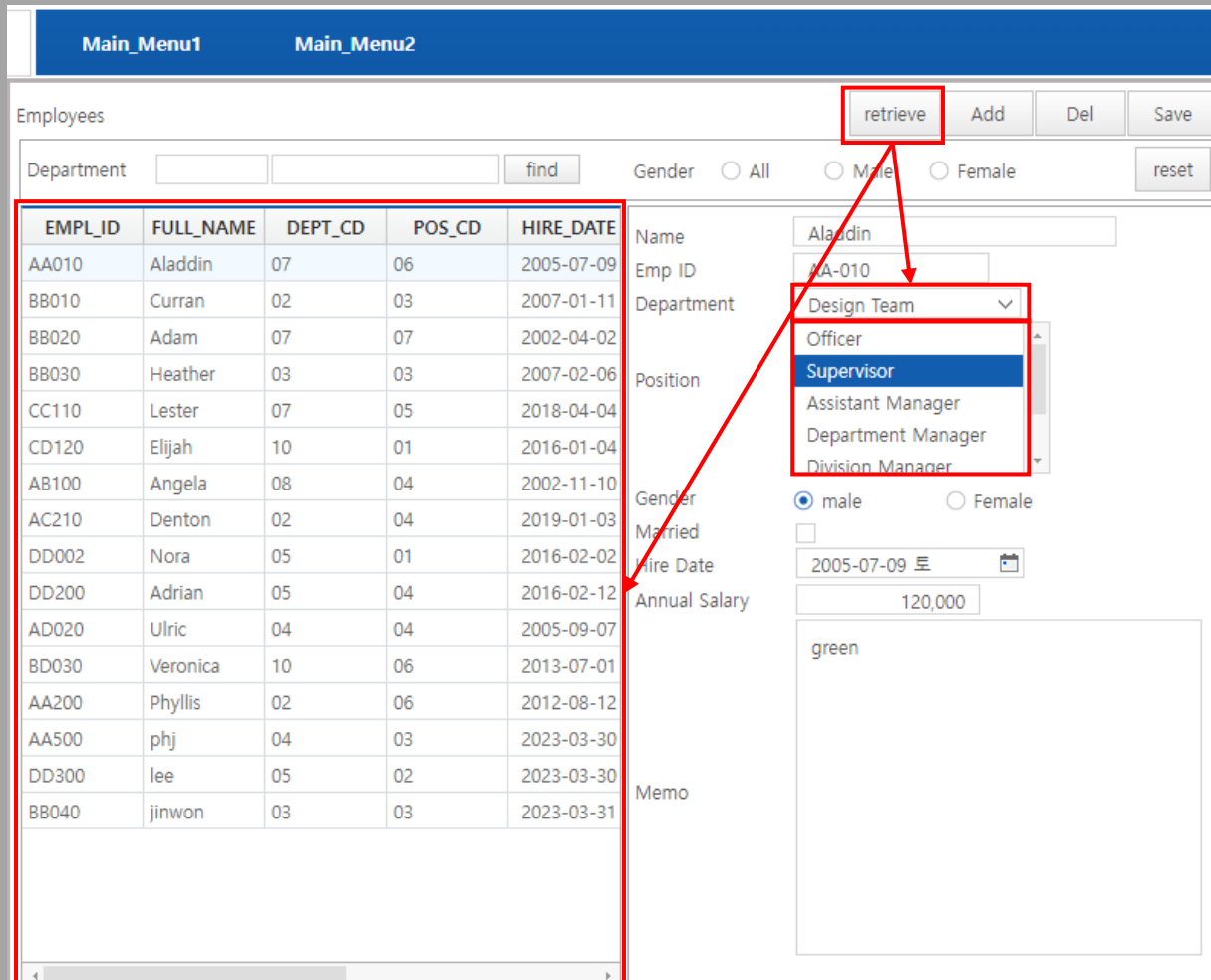
retrieve 버튼을 click 하면 sqlite와 연동하여 DB값이 grd_list에 출력되는것을 확인할 수 있다.

The screenshot displays a web application interface for managing employee data. At the top, there are two menu items: "Main_Menu1" and "Main_Menu2". Below the menu, the title "Employees" is visible. The interface includes a search section with a "Department" dropdown, a "find" button, and radio buttons for "Gender" (All, Male, Female), along with a "reset" button. A "retrieve" button is highlighted with a red box, and a red arrow points from it to the table. The table, titled "grd_list", contains 20 rows of employee data. The first row is highlighted in blue. To the right of the table is a form for editing an employee, with fields for Name, Emp ID, Department, Position, Gender, Married, Hire Date, Annual Salary, and Memo. The form is pre-filled with data for the first employee in the table.

EMPL_ID	FULL_NAME	DEPT_CD	POS_CD	HIRE_DATE
AA010	Aladdin	07	06	2005-07-09
BB010	Curran	02	03	2007-01-11
BB020	Adam	07	07	2002-04-02
BB030	Heather	03	03	2007-02-06
CC110	Lester	07	05	2018-04-04
CD120	Elijah	10	01	2016-01-04
AB100	Angela	08	04	2002-11-10
AC210	Denton	02	04	2019-01-03
DD002	Nora	05	01	2016-02-02
DD200	Adrian	05	04	2016-02-12
AD020	Ulric	04	04	2005-09-07
BD030	Veronica	10	06	2013-07-01
AA200	Phyllis	02	06	2012-08-12
AA500	phj	04	03	2023-03-30
DD300	lee	05	02	2023-03-30
BB040	jinwon	03	03	2023-03-31

14. cbo_dept, lst_pos 조회

retrieve 버튼을 click 하면 sqlite와 연동하여 DB값이 grd_list, cbo_dept, lst_pos에 출력되는것을 확인할 수 있다.



Employees

retrieve Add Del Save

Department find Gender ☐ All ☐ Male ☐ Female reset

EMPL_ID	FULL_NAME	DEPT_CD	POS_CD	HIRE_DATE
AA010	Aladdin	07	06	2005-07-09
BB010	Curran	02	03	2007-01-11
BB020	Adam	07	07	2002-04-02
BB030	Heather	03	03	2007-02-06
CC110	Lester	07	05	2018-04-04
CD120	Elijah	10	01	2016-01-04
AB100	Angela	08	04	2002-11-10
AC210	Denton	02	04	2019-01-03
DD002	Nora	05	01	2016-02-02
DD200	Adrian	05	04	2016-02-12
AD020	Ulric	04	04	2005-09-07
BD030	Veronica	10	06	2013-07-01
AA200	Phyllis	02	06	2012-08-12
AA500	phj	04	03	2023-03-30
DD300	lee	05	02	2023-03-30
BB040	jinwon	03	03	2023-03-31

Name Aladdin

Emp ID AA-010

Department Design Team

Position Supervisor

Gender ☒ male ☐ Female

Married ☐

Hire Date 2005-07-09 토

Annual Salary 120,000

Memo green

14. cbo_dept, lst_pos 조회

retrieve 버튼의 btn_retrieve_onclick 이벤트의 스크립트를 아래와 같이 변경한다.

```
this.btn_retrieve_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    var id = "empSelect";
    var url = "SvcURL::select_emp.jsp";//SvcURL에 설정된 경로에 jsp파일 지정
    var reqDs = "";
    var respDs = "ds_emp=out_emp";//Form_Emp 폼에 ds_emp에 transaction으로 가져온 out_emp dataset을 입력한다.
    var args = "";
    var callback = "fn_callback";//비동기 통신 결과를 받을 콜백 함수 지정

    this.transaction(id, url, reqDs, respDs, args, callback);

    id = "deptSelect";
    url = "SvcURL::select_dept.jsp";//SvcURL에 설정된 경로에서 select_dept.jsp 파일을 지정한다.
    respDs = "ds_dept=out_dept";//Form_Emp 폼에 ds_dept에 transaction으로 가져온 out_dept dataset을 입력한다.

    this.transaction(id,url,reqDs,respDs,args,callback);

    id = "posSelect";
    url = "SvcURL::select_pos.jsp";//SvcURL에 설정된 경로에서 select_pos.jsp 파일을 지정한다.
    respDs = "ds_pos=out_pos";//Form_Emp 폼에 ds_pos에 transaction으로 가져온 out_pos dataset을 입력한다.

    this.transaction(id,url,reqDs,respDs,args,callback);
};
```

14. cbo_dept, lst_pos 조회

transaction에 callback으로 설정된 fn_callback 함수를 아래와 같이 수정한다.

```
this.fn_callback = function(svcID, errCD, errMsg){
    if(errCD == 0){//errCD값이 0이면 transaction 성공
        if(svcID=="empSelect"){
            var rowcount = this.ds_emp.rowcount;
            this.alert(rowcount + " numbers of data have been found.");
            this.alert(svcID + " Retrieve Success!!!");
        }else if(svcID=="deptSelect"){
            var rowcount = this.ds_dept.rowcount;
            this.alert(rowcount + " numbers of data have been found.");
            this.alert(svcID + " Retrieve Success!!!");
        }else if(svcID=="posSelect"){
            var rowcount = this.ds_pos.rowcount;
            this.alert(rowcount + " numbers of data have been found.");
            this.alert(svcID + " Retrieve Success!!!");
        }
    }else{//errCD값이 0이아니면 transaction 실패
        this.alert("Error: " + errMsg + ", errCD:" + errCD + ", svcID:" + svcID);
        return;
    }
}
```

15. 조회 서비스 select_dept.jsp 파일 작성-1

cbo_dept 콤포넌트에 조회되도록 아래와 같이 파일을 작성한다.

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page language="java"%>
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.sql.*"%>
<%@ page import="org.apache.commons.logging.*"%>
<%@ page import="com.nexacro.java.xapi.tx.*" %>
<%@ page import="com.nexacro.java.xapi.data.*" %>

<%
//PlatformData(비동기 통신으로 transaction callback 함수에 데이터 전송시 데이터를 담아서 전송할때 사용)
PlatformData pData = new PlatformData();
int nErrorCode = 0;
String strErrorMsg = "START";
try{
    /***** JDBC Connection *****/

    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
```

15. 조회 서비스 select_dept.jsp 파일 작성-2

cbo_dept 컴포넌트에 조회되도록 아래와 같이 파일을 작성한다.

```
try{
    Class.forName("org.sqlite.JDBC");
    conn = DriverManager.getConnection("jdbc:sqlite:C:\\WWW\\JAVA\\WWW\\sqlite_LiteDB\\WWW\\nexacro_sample_db.sqlite");
    stmt = conn.createStatement();
    String sql = "select * from tb_dept";

    rs = stmt.executeQuery(sql);
    //DataSet 생성 후 DataType 설정
    DataSet ds = new DataSet("out_dept");
    ds.addColumn("DEPT_CD", DataTypes.STRING, 2);
    ds.addColumn("DEPT_NAME", DataTypes.STRING, 50);

    //DataSet에 tb_emp 테이블에서 조회한 값 입력하기
    int row = 0;
    while(rs.next()){
        row = ds.newRow();
        ds.set(row, "DEPT_CD", rs.getString("DEPT_CD"));
        ds.set(row, "DEPT_NAME", rs.getString("DEPT_NAME"));
    }
    // #1 dataset -> PlatformData
    pData.addDataSet(ds);//pData 객체에 DataSet ds를 입력

    nErrorCode = 0;
    strErrorMsg = "SUCC";
}
```


15. 조회 서비스 select_dept.jsp 파일 작성-3

cbo_dept 콤포넌트에 조회되도록 아래와 같이 파일을 작성한다.

```
        }catch(ClassNotFoundException e){
            e.printStackTrace();
            nErrorCode = -1;
            strErrorMsg = e.getMessage();
        }catch(SQLException e){
            e.printStackTrace();
            nErrorCode = -1;
            strErrorMsg = e.getMessage();
        }
        /***** JDBC Close *****/
        if( stmt != null ) try { stmt.close(); } catch(Exception e) {nErrorCode = -1; strErrorMsg = e.getMessage();}
        if ( conn != null ) try {conn.close(); } catch(Exception e) {nErrorCode = -1; strErrorMsg = e.getMessage();}
    }catch(Throwable th){
        nErrorCode = -1;
        strErrorMsg = th.getMessage();
    }
    //에러코드와 에러메시지를 VariableList에 담아 클라이언트에 전송
    VariableList varList = pData.getVariableList();
    varList.add("ErrorCode", nErrorCode);
    varList.add("ErrorMsg", strErrorMsg);

    HttpPlatformResponse res = new HttpPlatformResponse(response, PlatformType.CONTENT_TYPE_XML, "UTF-8");
    res.setData(pData);

    // Send data
    res.sendData();
    %>
```

16. 조회 서비스 select_pos.jsp 파일 작성-1

lst_pos 콤포넌트에 조회되도록 아래와 같이 파일을 작성한다.

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page language="java"%>
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.sql.*"%>
<%@ page import="org.apache.commons.logging.*"%>
<%@ page import="com.nexacro.java.xapi.tx.*" %>
<%@ page import="com.nexacro.java.xapi.data.*" %>

<%
//PlatformData(비동기 통신으로 transaction callback 함수에 데이터 전송시 데이터를 담아서 전송할때 사용)
PlatformData pData = new PlatformData();
int nErrorCode = 0;
String strErrorMsg = "START";
try{
    /***** JDBC Connection *****/

    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
```

16. 조회 서비스 select_pos.jsp 파일 작성-2

lst_pos 컴포넌트에 조회되도록 아래와 같이 파일을 작성한다.

```
try{
    Class.forName("org.sqlite.JDBC");
    conn = DriverManager.getConnection("jdbc:sqlite:C:\\WWW\\JAVA\\WWW\\sqlite_LiteDB\\WWW\\nexacro_sample_db.sqlite");
    stmt = conn.createStatement();
    String sql = "select * from tb_pos";

    rs = stmt.executeQuery(sql);
    //DataSet 생성 후 DataType 설정
    DataSet ds = new DataSet("out_pos");
    ds.addColumn("POS_CD", DataTypes.STRING, 2);
    ds.addColumn("POS_NAME", DataTypes.STRING, 50);

    //DataSet에 tb_emp 테이블에서 조회한 값 입력하기
    int row = 0;
    while(rs.next()){
        row = ds.newRow();
        ds.set(row, "POS_CD", rs.getString("POS_CD"));
        ds.set(row, "POS_NAME", rs.getString("POS_NAME"));
    }
    // #1 dataset -> PlatformData
    pData.addDataSet(ds);//pData 객체에 DataSet ds를 입력

    nErrorCode = 0;
    strErrorMsg = "SUCC";
}
```

15. 조회 서비스 select_pos.jsp 파일 작성-3

lst_pos 콤포넌트에 조회되도록 아래와 같이 파일을 작성한다.

```
        }catch(ClassNotFoundException e){
            e.printStackTrace();
            nErrorCode = -1;
            strErrorMsg = e.getMessage();
        }catch(SQLException e){
            e.printStackTrace();
            nErrorCode = -1;
            strErrorMsg = e.getMessage();
        }
        /***** JDBC Close *****/
        if( stmt != null ) try { stmt.close(); } catch(Exception e) {nErrorCode = -1; strErrorMsg = e.getMessage();}
        if ( conn != null ) try {conn.close(); } catch(Exception e) {nErrorCode = -1; strErrorMsg = e.getMessage();}
    }catch(Throwable th){
        nErrorCode = -1;
        strErrorMsg = th.getMessage();
    }
    //에러코드와 에러메시지를 VariableList에 담아 클라이언트에 전송
    VariableList varList = pData.getVariableList();
    varList.add("ErrorCode", nErrorCode);
    varList.add("ErrorMsg", strErrorMsg);

    HttpPlatformResponse res = new HttpPlatformResponse(response, PlatformType.CONTENT_TYPE_XML, "UTF-8");
    res.setData(pData);

    // Send data
    res.sendData();
    %>
```

감사합니다.

- Fin -