

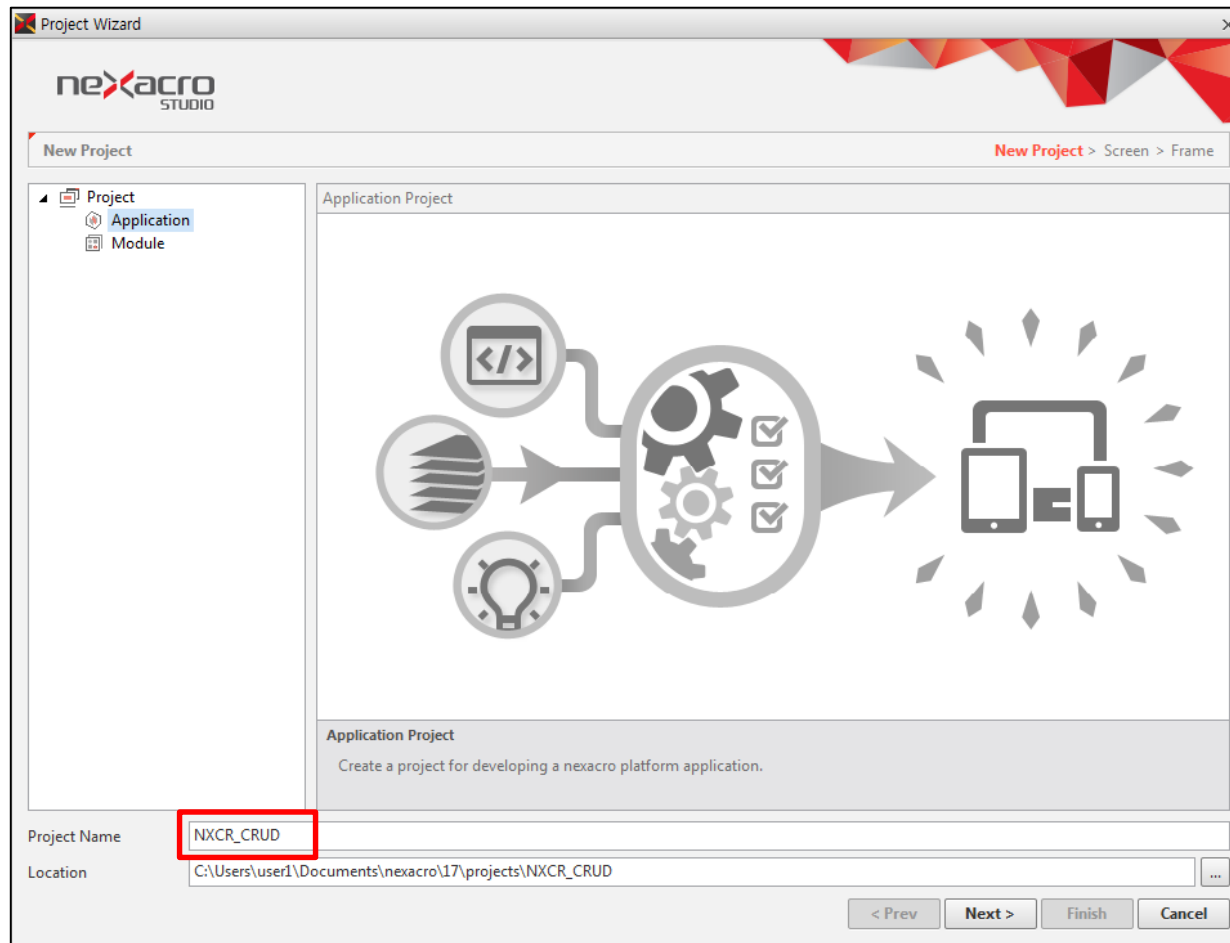


nexacro\_CRUD

1

Project 생성

# 1. 프로젝트 생성



File → New → Project 선택  
새 프로젝트 생성

# 1. 프로젝트 생성

Project Wizard

nexacro STUDIO

New Project

New Project > Screen > Frame

**Desktop**

☒

Width 650 px

Height 670 px

Screen ID Desktop\_screen

App ID App\_Desktop

Screen suitable for desktop environment.  
You can set the size to suit your desktop resolution.

**Tablet**

☐

※ Flexible size

Screen ID Tablet\_screen

App ID App\_Tablet

Screen suitable for tablet environment.  
- Apple iPad series  
- Samsung Galaxy Tab series  
- LG G Pad series

**Phone**

☐

※ Flexible size

Screen ID Phone\_screen

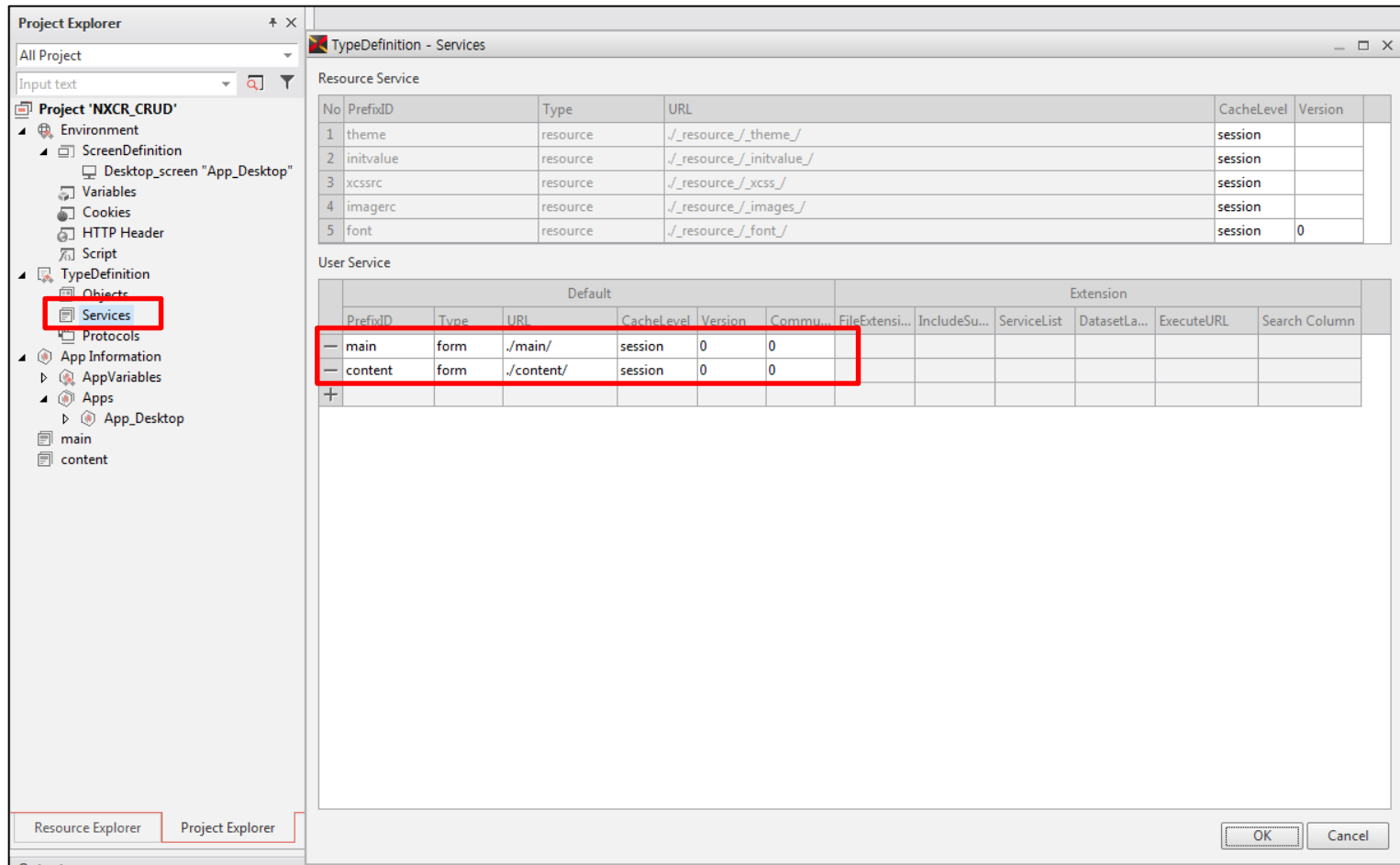
App ID App\_Phone

Screen suitable for phone environment.  
- Apple iPhone series  
- Samsung Galaxy series  
- LG Optimus series

< Prev Next > Finish Cancel

새 창을 띄워서 관리자 페이지를 운용할 계획이므로 크기를 650px X 670px로 생성

# 1. 프로젝트 생성



서비스 그룹 2개 생성 main(base를 변경), content

# 1. 프로젝트 생성



2

form 작성

## 2. form 작성(main)

Form Wizard

neXacro STUDIO

New Form Information > Position (Optional)

Name: main

Location: main

Layout: default

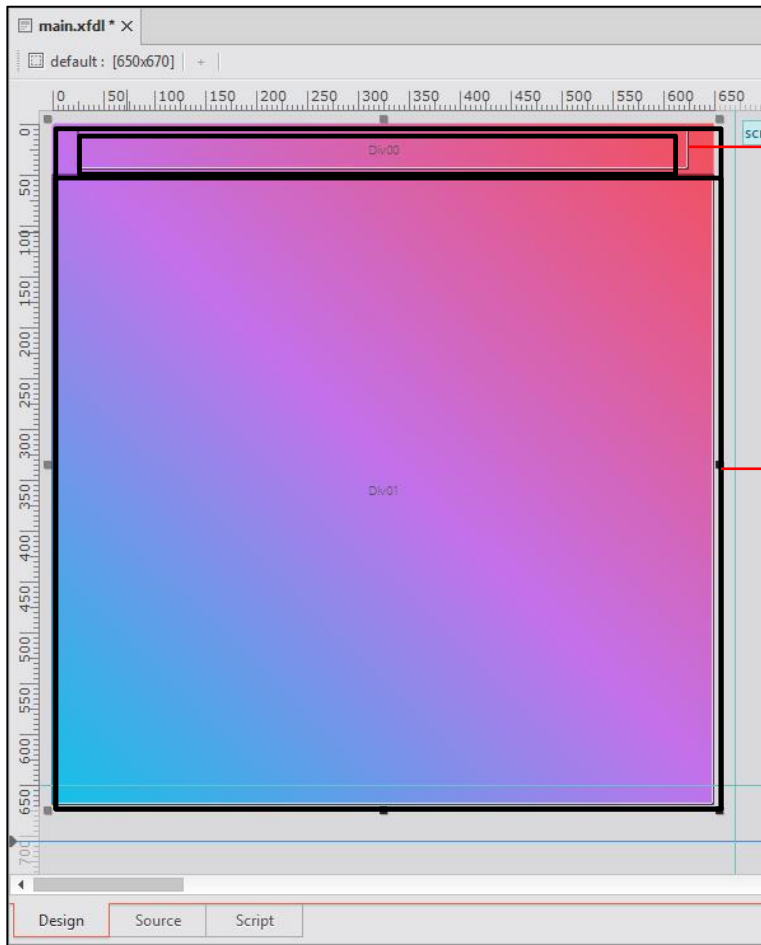
Layout Information	
name	default
screenid	
width	650 px
height	670 px
stepcount	
stepindex	
description	
mobileorientation	landscape

**height**  
Form 에 정의된 현재 Layout 의 높이를 설정하는 속성입니다.

< Prev   Next >   Finish   Cancel



## 2. form 작성(main)



메뉴Div

Position			
positionstep	0		
left	25	px	▼
top	5	px	▼
width	600	px	▼
height	40	px	▼
right		px	▼
bottom		px	▼

컨텐츠 Div

Position			
positionstep	0		
left	0	px	▼
top	50	px	▼
width	650	px	▼
height	620	px	▼
right		px	▼
bottom		px	▼

main form 설정 : linear-gradient(to right top,#12c2e9,#c471ed,#f64f59) 뒷배경색  
컨텐츠 Div는 따로작성해둔 후 메뉴 클릭 시 url을 변경하여 설정할것임

## 2. form 작성(main)

Position	
positionstep	0
left	0 px
top	0 px
width	100 px
height	40 px

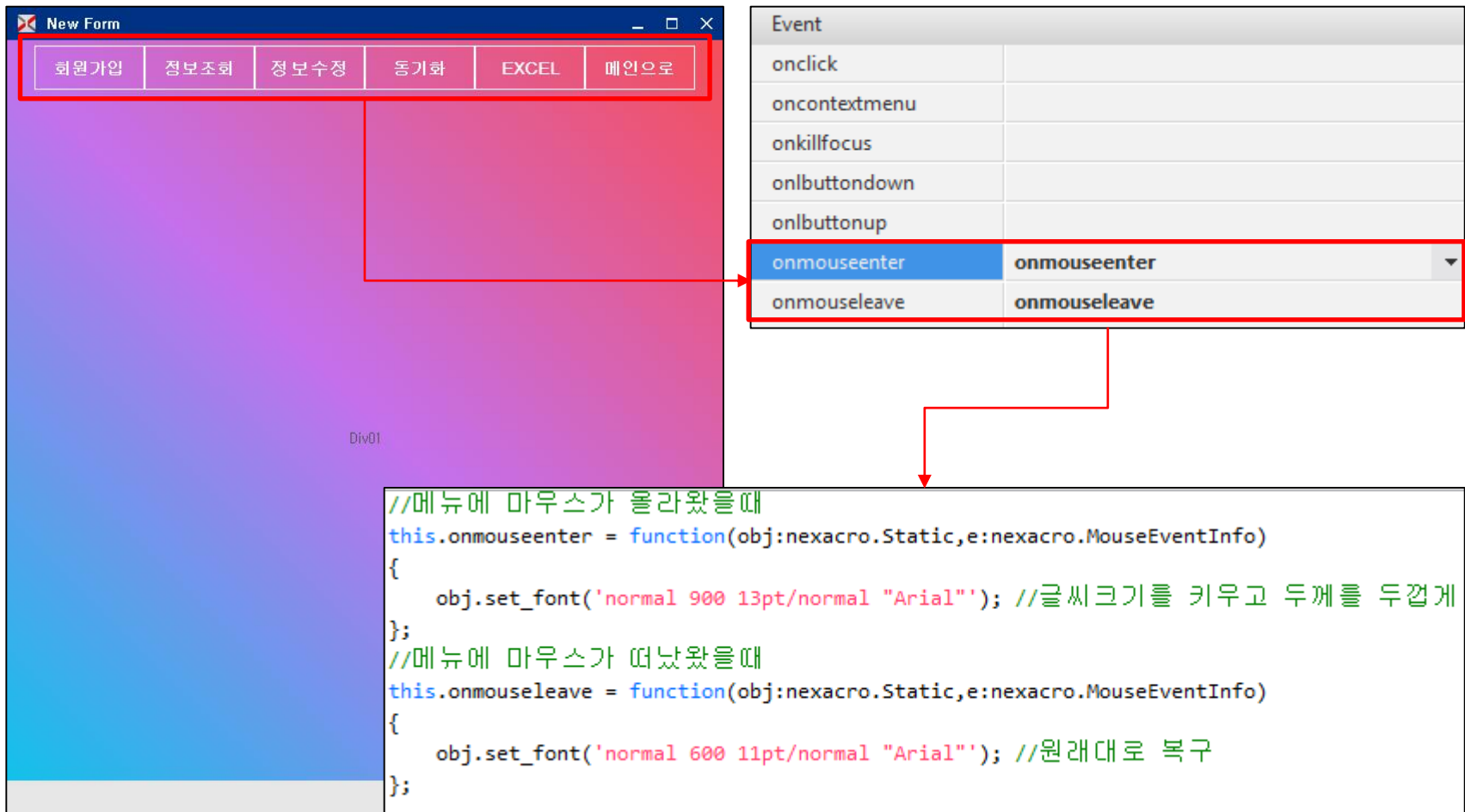
Style	
background	
border	1px solid white
borderRadius	
boxShadow	
color	<input type="checkbox"/> white
cursor	
edge	
font	normal 600 11pt/normal "Arial"
letterSpacing	px
opacity	
padding	
textAlign	center

Position	
positionstep	0
left	0px
top	0px
width	100px
height	40px

Style	
background	
border	1px solid white
borderRadius	
boxShadow	
color	<input type="checkbox"/> white
cursor	
edge	
font	normal 600 11pt/normal "Arial"
letterSpacing	px
opacity	
padding	
textAlign	center

Div00안에 Static 컴포넌트를 통한 메뉴 작성 (버튼이나 다른 것을 사용해도 됨)  
Static의 다른 설정은 모두 동일하며 left를 100px씩 증가시키면서 text를 수정하면 됨  
font: 설정

## 2. form 작성(main)



The screenshot shows a web form editor interface. At the top, there is a menu bar with buttons: "회원가입", "정보조회", "정보수정", "동기화", "EXCEL", and "메인으로". Below the menu bar is a large blue area labeled "Div01". To the right of the form is a table titled "Event" with two columns: "Event" and "Handler". The table lists various events, and the "onmouseenter" and "onmouseleave" events are highlighted with a red box. A red arrow points from the "onmouseenter" event in the table to the corresponding JavaScript code block below.

Event	Handler
onclick	
oncontextmenu	
onkillfocus	
onmousedown	
onmouseup	
onmouseenter	onmouseenter
onmouseleave	onmouseleave

```
//메뉴에 마우스가 올라왔을때
this.onmouseenter = function(obj:nexacro.Static,e:nexacro.MouseEventInfo)
{
    obj.set_font('normal 900 13pt/normal "Arial"'); //글씨크기를 키우고 두께를 두껍게
};
//메뉴에 마우스가 떠났을때
this.onmouseleave = function(obj:nexacro.Static,e:nexacro.MouseEventInfo)
{
    obj.set_font('normal 600 11pt/normal "Arial"'); //원래대로 복구
};
```

각 메뉴에 마우스가 올라왔을때 변화를 주는 이벤트 작성(hover)  
공통 기능이므로 마우스가 올라왔을 때, 떠났을 때를 작성 후 연결

## 2. form 작성(join)

Form Wizard

neXacro STUDIO

New Form Information > Position (Optional)

Name: join

Location: content

Layout: default

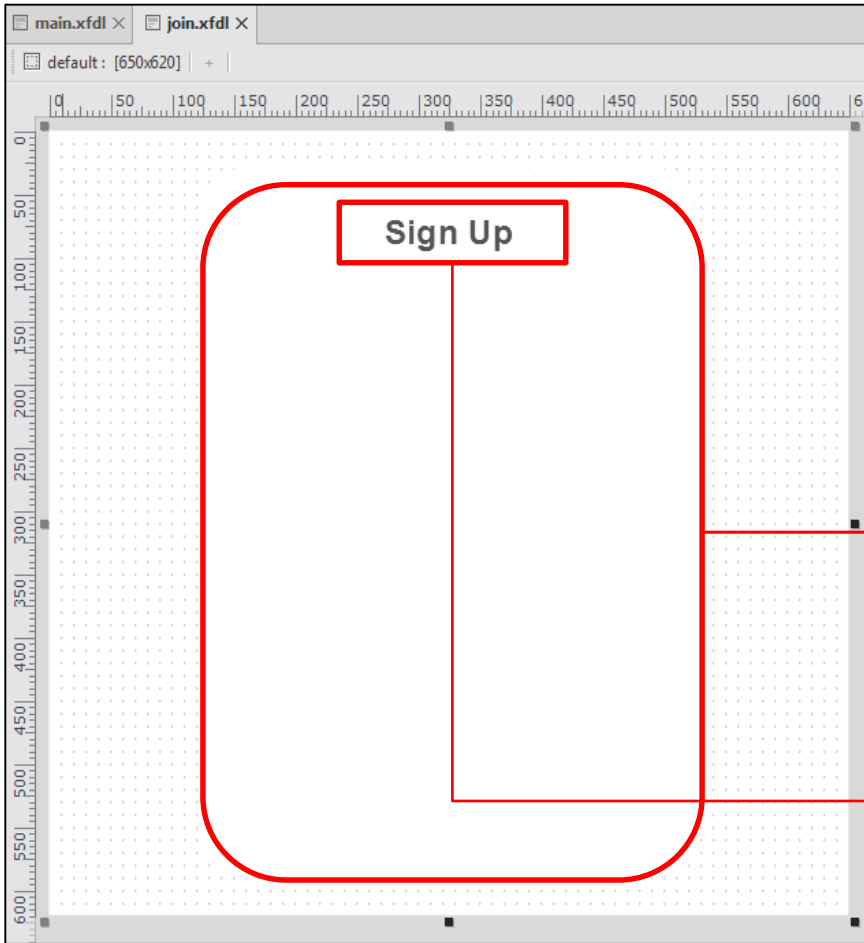
Layout Information	
name	default
screenid	
width	650 px
height	620 px
stepcount	
stepindex	
description	
mobileorientation	landscape

**height**  
Form 에 정의된 현재 Layout 의 높이를 설정하는 속성입니다.

< Prev   Next >   Finish   Cancel

회원 가입부터 생성

## 2. form 작성(join)



Position		
positionstep	0	
left	125	px ▾
top	30	px ▾
width	400	px ▾
height	560	px ▾

Style	
background	white
border	
borderRadius	30px

Position		
positionstep	0	
left	100	px ▾ ▾
top	25	px ▾ ▾
width	200	px ▾ ▾
height	50	px ▾ ▾

컴포넌트를 담은 Div 생성

Static 컴포넌트 회원가입 Sign Up(font : normal 900 20pt/normal "Arial", text-align:center)

## 2. form 작성(join)

New Form

### Sign Up

ID

PASSWORD

NAME

AGE

ADDRESS

회원가입

Static컴포넌트

font	normal 500 10pt/normal "Arial"	...
font-style	normal	
font-weight	500	
font-size	10	pt
line-height	normal	
font-family	"Arial"	

Position		
positionstep	0	
left	50	px
top	95	px
width	100	px
height	20	px

각 항목의 이름으로 top만 변경됨(95, 170, 245, 320, 395)

## 2. form 작성(join)

New Form

### Sign Up

ID

PASSWORD

NAME

AGE

ADDRESS

회원가입

Edit컴포넌트

Style	
background	
border	0px none,0px none,1px solid gray
+ border-top	0px none
+ border-right	0px none
+ border-bottom	1px solid gray
+ border-left	0px none

Position	
positionstep	0
left	50 px
top	125 px
width	300 px
height	30 px

각 항목을 입력 받는 필드로 이름으로 top만 변경됨(125, 200, 275, 425)  
패스워드는 보이면 안되므로 password 속성 true로 변경

## 2. form 작성(join)

New Form

### Sign Up

ID  
|

PASSWORD

NAME

AGE

ADDRESS

회원가입

Button 컴포넌트

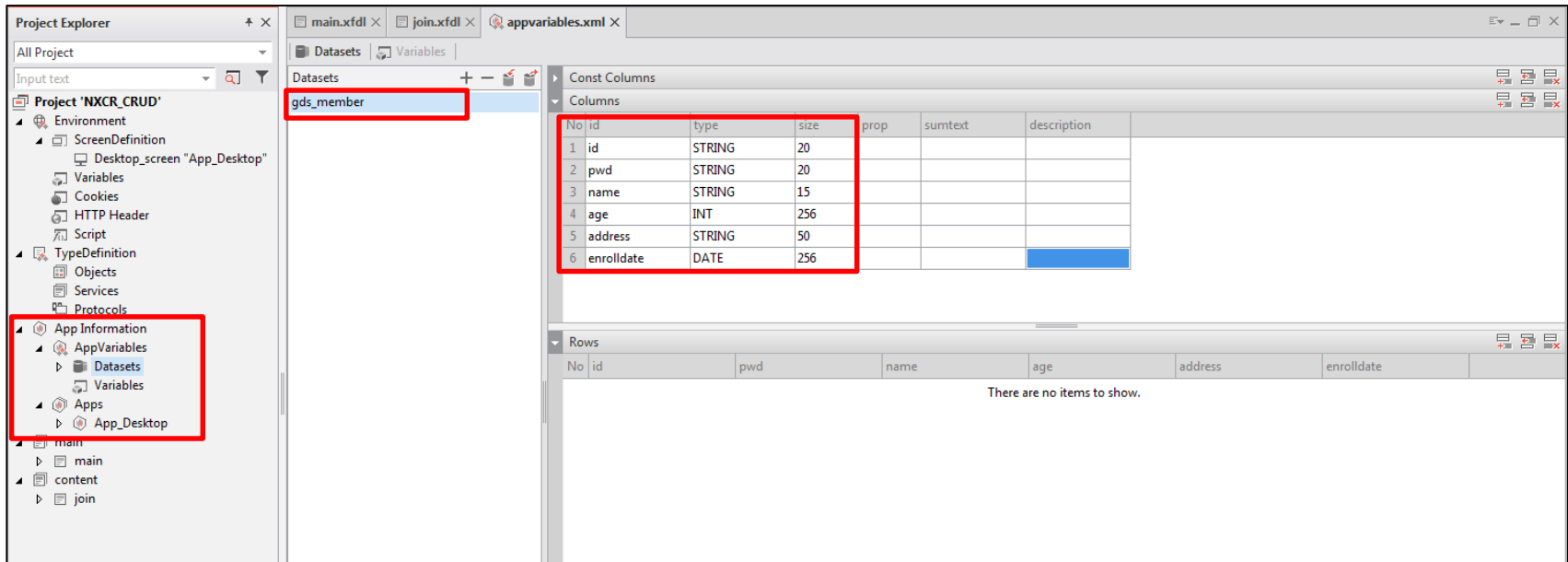
Style	
background	linear-gradient(to right,#12c2e9,#c471ed,;
border	1px solid transparent
borderRadius	50px
boxShadow	
color	<input type="checkbox"/> white
cursor	pointer
edge	
font	normal 600 13pt/normal "Arial"

Position		
positionstep	0	
left	100	px ▾
top	490	px ▾
width	200	px ▾
height	50	px ▾

각 필드 입력 후 회원가입버튼을 누르면 회원이 가입되도록 승인하는 버튼



## 2. form 작성(join)



회원 정보를 저장할 DataSet 생성(gds\_member)  
DataSet은 Join폼에서만 사용하는 것이 아니기 때문에 글로벌 변수로 설정

## 2. form 작성(join)

Event	
onclick	Div00_Button00_onclick
oncontextmenu	
ondblclick	
ondrag	
ondragenter	
ondragleave	
ondragmove	
ondrop	
onkeydown	
onkeyup	
onkillfocus	
onmousedown	
onmouseup	
onmouseenter	Div00_Button00_onmouseenter
onmouseleave	Div00_Button00_onmouseleave

다음 페이지

```
this.Div00_Button00_onmouseenter = function(obj:nexacro.Button,e:nexacro.MouseEventInfo)
{
    this.Div00.form.Button00.set_font('normal 900 15pt/normal "Arial"');
};

this.Div00_Button00_onmouseleave = function(obj:nexacro.Button,e:nexacro.MouseEventInfo)
{
    this.Div00.form.Button00.set_font('normal 600 13pt/normal "Arial"');
};
```

버튼이벤트 작성  
버튼영역에 마우스 들어왔을 때 / 나갈 때 / 클릭 할 때

## 2. form 작성(main)

```
this.Div00_Button00_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    //각 입력값 읽어오기
    //password설정은 text로 읽어오면 **** 을 읽어오기때문에 value를 읽어온다
    var id = this.Div00.form.Edit00.text;
    var pwd = this.Div00.form.Edit01.value;
    var name = this.Div00.form.Edit02.text;
    var age = this.Div00.form.MaskEdit00.text;
    var address = this.Div00.form.Edit03.text;
    //빈값이 있는지 검사
    if(id==" " || pwd==" " || name==" " || age==" " || address==""){
        alert("정보를 모두 입력해주세요");
    }else{
        var addRow = nexacro.Application.gds_member.addRow();           //gds_member에 row 추가
        nexacro.Application.gds_member.setColumn(addRow,"id",id);       //id값 입력
        nexacro.Application.gds_member.setColumn(addRow,"pwd",pwd);     //pw값 입력
        nexacro.Application.gds_member.setColumn(addRow,"name",name);   //name값 입력
        nexacro.Application.gds_member.setColumn(addRow,"age",age);     //age값 입력
        nexacro.Application.gds_member.setColumn(addRow,"address",address); //address값 입력
        var d = new Date();           //Date를 이용하여 오늘 날짜를 구함
        var yyyy = d.getFullYear();
        var mm = (d.getMonth()+1);
        var dd = d.getDate();
        mm = (mm>9?"":"0")+mm;
        dd = (dd>9?"":"0")+dd;
        nexacro.Application.gds_member.setColumn(addRow,"enrolldate",yyyy+mm+dd); //enrolldate값 입력
        alert("회원 추가완료");
    }
};
```

각 Edit 및 MaskEdit의 값을 읽어와서 gds\_member에 저장하는 함수

## 2. form 작성(search)

### Search

회원 목록

아이디	이름	주소
none		

아이디

이름

검색

<

>

Static

Position		
positionstep	0	
left	50	px ▾
top	100	px ▾
width	300	px ▾
height	50	px ▾

Style	
background	linear-gradient(to right,#12c2e9,#c471ed,
border	
borderRadius	
boxShadow	
color	<input type="checkbox"/> white
cursor	
edge	
font	normal 900 15pt/normal "Arial"
letterSpacing	px
opacity	
padding	
textAlign	center

join과 동일한 구조로 search로 생성  
Static 컴포넌트 생성

## 2. form 작성(search)

**Search**

회원 목록

아이디	이름	주소
-----	----	----

아이디      이름

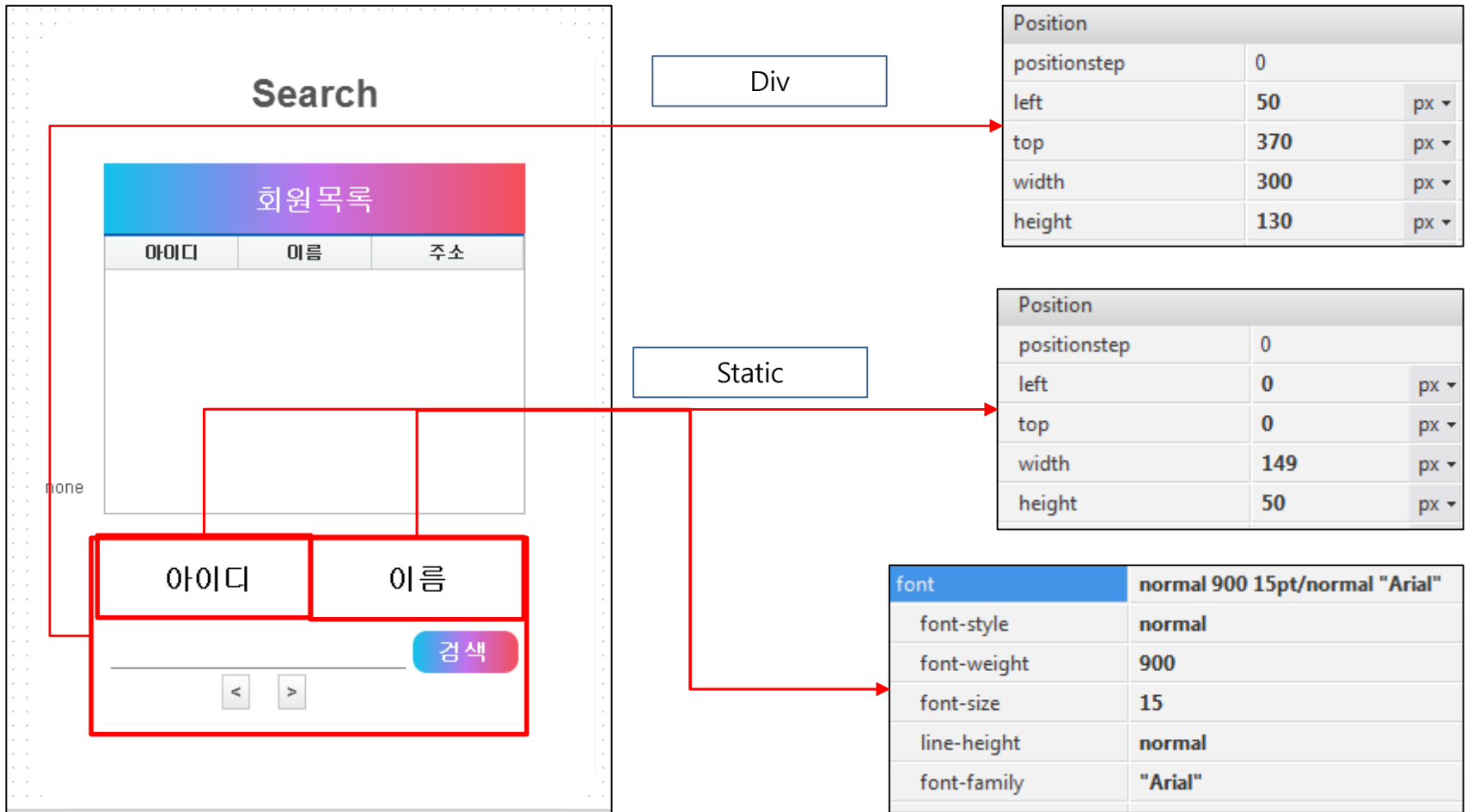
Binding	
binddataset	gds_member
formatid	
formats	<Formats> <Format id="default"> <Column

Position		
positionstep	0	
left	50	px ▼
top	150	px ▼
width	300	px ▼
height	200	px ▼

autofittype	col
-------------	-----

gds\_member를 출력 할 Grid 생성

## 2. form 작성(search)



검색을 하는 영역구분을 위한 Div, 아이디/이름 클릭 시 해당 영역으로 검색

## 2. form 작성(main)

**Search**

회원 목록

아이디	이름	주소
none		

아이디

이름

검색

<

>

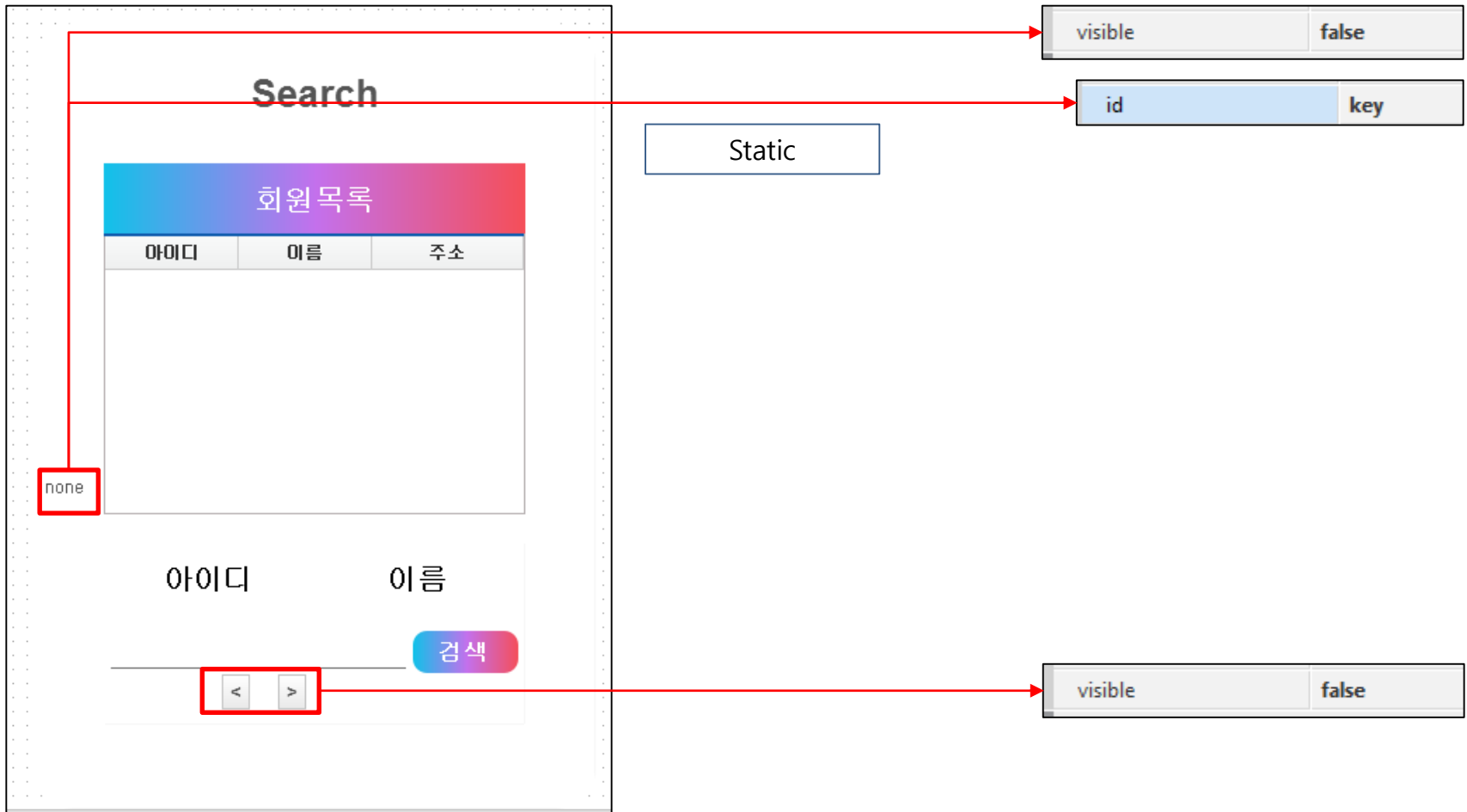
Edit

Position		
positionstep	0	
left	5	px ▾
top	60	px ▾
width	200	px ▾
height	30	px ▾

Style	
background	
border	0px none,0px none,1px solid gray
borderRadius	
boxShadow	
color	<input type="checkbox"/>
cursor	
edge	
font	normal 15pt/normal "Arial"

검색어를 입력하는 Edit

## 2. form 작성(search)



아이디/이름 선택 시 어떤걸 선택한지 알기 위해 만든 Static(기본값 none이며 보이지 않게 설정)  
검색결과가 여러 개인 경우 다음과 이전을 표시(보이지않게 설정)



## 2. form 작성(search)

```
this.Div00_Div00_Static00_onclick = function(obj:nexacro.Static,e:nexacro.ClickEventInfo)
{
    //아이디 static 클릭 시 배경변경, 글씨색 변경, key의 text값 변경
    this.Div00.form.Div00.form.Static00.set_background("linear-gradient(to right,#12c2e9,#c471ed,#f64f59)");
    this.Div00.form.Div00.form.Static00.set_color("white");
    this.Div00.form.Div00.form.Static01.set_background("");
    this.Div00.form.Div00.form.Static01.set_color("black");
    this.Div00.form.key.set_text("id");
};
```

아이디 클릭 시

```
this.Div00_Div00_Static01_onclick = function(obj:nexacro.Static,e:nexacro.ClickEventInfo)
{
    //이름 static 클릭 시 배경변경, 글씨색 변경, key의 text값 변경
    this.Div00.form.Div00.form.Static01.set_background("linear-gradient(to right,#12c2e9,#c471ed,#f64f59)");
    this.Div00.form.Div00.form.Static01.set_color("white");
    this.Div00.form.Div00.form.Static00.set_background("");
    this.Div00.form.Div00.form.Static00.set_color("black");
    this.Div00.form.key.set_text("name");
};
```

이름 클릭 시

검색 조건 선택 시 표시변경 및 구분자로 사용할 key의 text값 변경

## 2. form 작성(search)

```
this.Div00_Div00_Button00_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    var key = this.Div00.form.key.text;           //key의 값 읽어와서 변수에 저장
    var value = this.Div00.form.Div00.form.Edit00.text; //검색어 읽어와서 변수에 저장
    if(key == "none"){ //아이디나 이름을 선택하지 않고 검색한 경우
        alert("아이디 또는 이름을 선택하세요");
    }else{
        searchRow = new Array(); //검색된 행의 번호를 담을 배열 생성
        index=0; //검색된 row를 저장하는 배열의 다음과 이전 버튼에서 사용 할 index
        var i = 0; //searchRow에 값을 추가할 때 사용할 index용변수
        var row = -1; //DataSet을 검색할 때 사용하는 변수
        do{
            row++;
            //key 컬럼의 값이 value인 행번호를 row부터 찾음(첫번째 찾고 종료)
            //해당되는 row가 없으면 -1반환
            row = nexacro.Application.gds_member.findRow(key,value,row);
            if(row != -1){ //해당되는 값이 있으면
                searchRow[i] = row; //배열의 행번호를 담고 index를 증가
                i++;
            }
        }while(row != -1)
    }
    this.Div00.form.Grid00.selectRow(searchRow[index],true); //검색결과 중 첫번째 행을 Grid에서 선택
    if(searchRow.length>1){
        this.Div00.form.Div00.form.Button01.set_visible(true); //이전버튼 보임
        this.Div00.form.Div00.form.Button01.set_enable(false); //이전버튼 비활성화
        this.Div00.form.Div00.form.Button02.set_visible(true); //다음버튼 보임
        this.Div00.form.Div00.form.Button02.set_enable(true); //다음버튼 활성화
    }else{
        this.Div00.form.Div00.form.Button01.set_visible(false); //이전/다음버튼 둘다 안보임
        this.Div00.form.Div00.form.Button02.set_visible(false);
    }
};
```

검색버튼 클릭 시 이벤트 - 검색타입, 값을 읽어와서 로직처리

## 2. form 작성(main)

```
this.Div00_Div00_Button01_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    //이전버튼 클릭 시
    index--;    //선택 된 인덱스를 1뺀다
    this.Div00.form.Grid00.selectRow(searchRow[index],true);    //해당 row를 선택
    this.Div00.form.Div00.form.Button02.set_enable(true);    //이전으로 왔기때문에 다음 버튼 활성화
    if(index == 0){    //인덱스 가장 앞번호인경우 이전으로 더이상 갈 수 없기 때문에 비활성화
        this.Div00.form.Div00.form.Button01.set_enable(false);
    }
};
```

이전버튼 클릭 시

```
this.Div00_Div00_Button02_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    //다음버튼 클릭 시
    index++;    //선택 된 인덱스를 1더한다
    this.Div00.form.Grid00.selectRow(searchRow[index],true);    //해당 row를 선택
    this.Div00.form.Div00.form.Button01.set_enable(true);    //다음으로 왔기때문에 이전 버튼 활성화
    if(index+1 == searchRow.length){    //인덱스가 가장 뒷번호면 다음버튼 비활성화
        this.Div00.form.Div00.form.Button02.set_enable(false);
    }
};
```

다음버튼 클릭 시

이전/다음 버튼 클릭 시의 이벤트  
작성이 후 main에서 정보조회 클릭 시 나올 수 있도록 연계

## 2. form 작성(modify)

The image shows two windows from a software application. The 'New Form' window on the left is titled 'Modify' and contains a form with the following fields: ID, NAME, AGE, ADDRESS, and EnrollDate. The EnrollDate field is a date picker. At the bottom of the form are two buttons: '수정' (Modify) and '삭제' (Delete). The 'Bind Item' window on the right is a dialog box with a table containing the following data:

Bind ID	Object	Property	Dataset	Column ID
item0	Div00.form.Edit00	value	gds_member	id
item1	Div00.form.Edit01	value	gds_member	name
item2	Div00.form.MaskEdit00	value	gds_member	age
item3	Div00.form.Edit02	value	gds_member	address
item4	Div00.form.Calendar00	value	gds_member	enrolldate
+				

At the bottom of the 'Bind Item' window are 'OK' and 'Cancel' buttons. Below the 'Bind Item' window, there is a horizontal bar with the text 'readonly' and 'true'.

기존방법과 거의 동일한 방식으로 Modify 생성, gds\_member와 binding search폼에서 선택된 row의 정보가 modify에 출력 됨

## 2. form 작성(modify)

```
this.Div00_Button00_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    if(this.Div00.form.Button00.text == "수정"){           //버튼의 글씨가 수정이면
        this.Div00.form.Edit01.set_readonly(false);
        this.Div00.form.MaskEdit00.set_readonly(false);
        this.Div00.form.Edit02.set_readonly(false);       //이름, 나이, 주소를 수정가능하게 변경
        this.Div00.form.Button00.set_text("수정완료");    //버튼 글씨 변경
    }else{         //버튼의 글씨가 수정완료면
        this.Div00.form.Edit01.set_readonly(true);
        this.Div00.form.MaskEdit00.set_readonly(true);
        this.Div00.form.Edit02.set_readonly(true);       //이름, 나이, 주소를 readonly로 설정
        this.Div00.form.Button00.set_text("수정");       //버튼 글씨 변경
    }
};
```

수정/수정완료 버튼 클릭 시

```
this.Div00_Button01_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    if(this.confirm("삭제 하시겠습니까?")){               //삭제 여부 확인
        nexacro.Application.gds_member.deleteRow(nexacro.Application.gds_member.rowposition); //현재 포지션의 행을 삭제
    }
};
```

삭제 버튼 클릭 시

수정버튼 클릭 시 readonly 속성을 false로 변경하여 변경 가능하도록 설정  
수정완료 버튼 클릭시에는 다시 readonly속성이 true로 변경되고, 정보는 실시간 변경  
삭제버튼 클릭 시에는 한번 확인 후 삭제함

## 2. form 작성(main)

The diagram illustrates the form structure and the corresponding JavaScript code for the '회원가입' (Sign Up) button.

**Form Structure:**

- Top Navigation Bar: 회원가입, 정보조회, 정보수정, 동기화, EXCEL, 메인으로
- Static Elements: Static00, Static01, Static02, Static03, Static04, Static05

**JavaScript Code:**

```
this.Div00_Static00_onclick = function(obj:nexacro.Static,e:nexacro.ClickEventInfo)
{
    this.Div00.form.Static00.set_border("1px solid white,1px solid white,1px solid transparent");
    this.Div00.form.Static01.set_border("1px solid white");
    this.Div00.form.Static02.set_border("1px solid white");
    this.Div00.form.Static04.set_border("1px solid white");
    this.Div01.set_url("content::join.xfdl");
};

this.Div00_Static01_onclick = function(obj:nexacro.Static,e:nexacro.ClickEventInfo)
{
    this.Div00.form.Static00.set_border("1px solid white");
    this.Div00.form.Static01.set_border("1px solid white,1px solid white,1px solid transparent");
    this.Div00.form.Static02.set_border("1px solid white");
    this.Div00.form.Static04.set_border("1px solid white");
    this.Div01.set_url("content::search.xfdl");
};

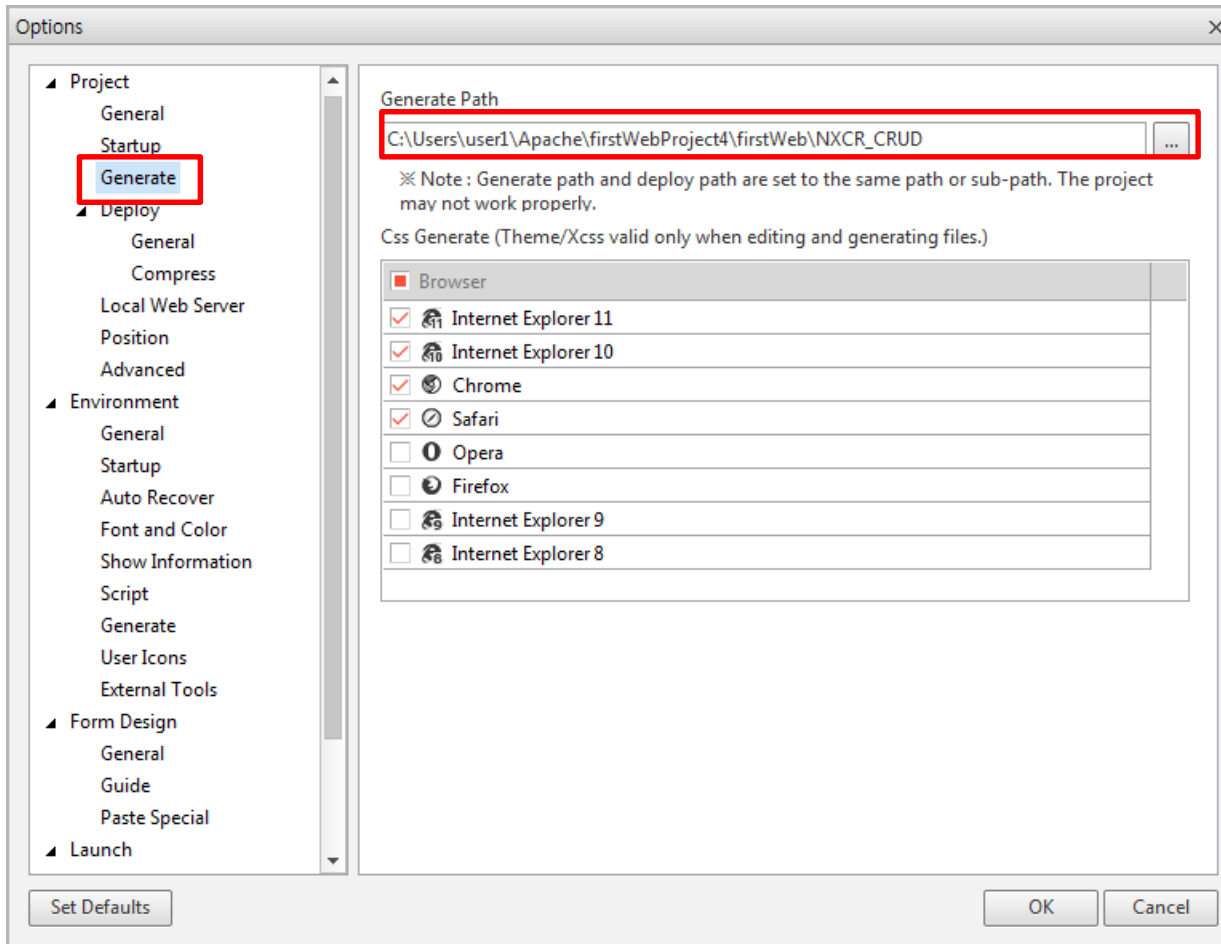
this.Div00_Static02_onclick = function(obj:nexacro.Static,e:nexacro.ClickEventInfo)
{
    this.Div00.form.Static00.set_border("1px solid white");
    this.Div00.form.Static01.set_border("1px solid white");
    this.Div00.form.Static02.set_border("1px solid white,1px solid white,1px solid transparent");
    this.Div00.form.Static04.set_border("1px solid white");
    this.Div01.set_url("content::modify.xfdl");
};
```

글로벌 변수인 gds\_membe와 CRUD를 하는 것까지는 완료함.  
이제 gds\_membe를 Servlet을 통해서 DataBase와 연동하는 것을 진행

3

서버 연동

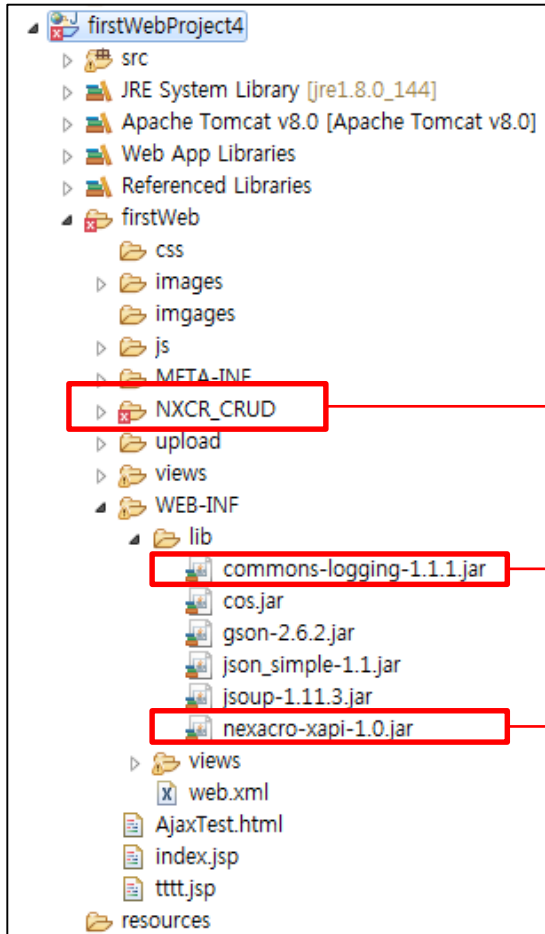
### 3. 서버 연동



Tools – options – Generate – Generate Path를 웹프로젝트 WebContent 안에 디렉토리를 생성하여 지정  
넥사크로 스튜디오에서 작성하면 이후에 generate버튼으로 서버와 동기화함



### 3. 서버 연동



generate를 통해 연동 된 폴더

넥사크로를 사용하기 위한 라이브러리 추가

### 3. 서버 연동(DB생성)

```
-- 계정생성 및 권한부여(system계정)
CREATE USER nxcr IDENTIFIED BY nxcr;
GRANT CONNECT, RESOURCE TO nxcr;
-- 테이블 생성 및 데이터입력(nxcr계정)
-- 테이블 생성 시 DataSet(gds_member)와 동일한 형태로 생성
CREATE TABLE NXCR_MEMBER (
  ID VARCHAR2(20) PRIMARY KEY,
  PWD VARCHAR2(20) NOT NULL,
  NAME VARCHAR2(15) NOT NULL,
  AGE NUMBER,
  ADDRESS VARCHAR(50),
  ENROLLDATE DATE
);
INSERT INTO NXCR_MEMBER VALUES('admin','1234','관리자',30,'서울',SYSDATE);
INSERT INTO NXCR_MEMBER VALUES('manager','1111','매니저',25,'서울',SYSDATE);
COMMIT;
SELECT * FROM NXCR_MEMBER;
```

DB와 연동을 위해 DB 계정 및 Table생성(테스트 데이터도 생성함)

### 3. 서버 연동(transaction)

transaction

- transaction 메소드를 통해서 실제 서버의 데이터를 가지고 옴(DB정보)
- 넥사크로 플랫폼에서 사용되는 데이터형식(XML)에 따라 만들어진 데이터를 가져옴
- transaction 메소드는 6개의 파라미터로 구성

transaction 메소드 파라미터

1. id : 트랜잭션을 구분하는 id
2. url : 트랜잭션을 요청하는 서비스의 주소(호출하는 Servlet)
3. reqDs : 트랜잭션 요청 시 전달 할 DataSet
4. resDs : 트랜잭션 처리 결과를 받을 때 사용하는 DataSet
5. args : 트랜잭션 요청 시 전달되는 파라미터 값
6. callback : 트랜잭션 결과를 처리할 콜백 함수

비동기통신, 바이너리통신, 압축통신을 지정하는 파라미터가 더 있지만 생략 시 기본값 적용

### 3. 서버 연동(transaction)

넥사크로 라이브러리 객체

1. PlatformData : 데이터를 보관하는 기본 객체
2. PlatformRequest : 서비스 요청 시 XML Format Data를 읽고 객체화하는 Input 객체
3. PlatformResponse : 서비스 요청 시 XML Format Data를 출력하는 Output 객체
4. DatasetList & DataSet : 데이터를 2차원 table 형태 또는 table array타입으로 보관
5. VariableList & Variable : I/O 인자 값으로 사용되는 단일 값을 보관하는 객체

비동기통신, 바이너리통신, 압축통신을 지정하는 파라미터가 더 있지만 생략 시 기본값 적용

### 3. 서버 연동

```
this.main_onload = function(obj:nexacro.Form,e:nexacro.LoadEventInfo)
{
    this.selectAllFunc();           //mainform이 로드되면 DB의 정보를 읽어옴
};
this.selectAllFunc = function(){   //DB에 있는 전체 회원 정보를 조회하여 gds_member에 저장
    var id = "selectAll";          //요청 ID
    var url = "http://localhost/first/selectAll"; //요청 URL(전체경로입력)
    var reqDs = "";                //전체조회이므로 넘겨주는 DataSet없음
    var resDs = "gds_member=out_ds" //요청결과를 out_ds에 담아서 리턴하고, 해당 값을 gds_member에 저장
    var args="";                   //요청 시 전달하는 변수 없음
    var callback = "";             //transaction수행 후 실행할 함수명
    //미리 작성한 파라미터를 사용하여 transaction메소드 호출
    this.transaction(id,url,reqDs,resDs,arts,callback);
}
```

transaction 메소드를 사용하여 selectAll Servlet을 호출

### 3. 서버 연동(select)

```
//회원전체조회이므로 request패킷에 전달되는 데이터가 없음 -> DB조회결과를 리턴만 하면 됨
ArrayList<NXCRMMember> list = new NXCRMMemberService().selectAll();           //전체 조회 결과를 받을 ArrayList를 생성하고, Service호출
//Service와 Dao는 기존에 사용하는 방식과 동일 함
//결과값을 저장 할 DataSet객체 생성(out_ds는 넥사크로로 데이터전달할때 구분하는 명칭 -> resDs에 명시함(gds_member=out_ds))
DataSet ds = new DataSet("out_ds");           //gds_member와 같은 DataSet 컬럼을 생성
ds.addColumn("id",DataTypes.STRING, (short)20);
ds.addColumn("pwd",DataTypes.STRING, (short)20);
ds.addColumn("name",DataTypes.STRING, (short)15);
ds.addColumn("age",DataTypes.INT, (short)256);
ds.addColumn("address",DataTypes.STRING, (short)50);
ds.addColumn("enrolldate",DataTypes.STRING, (short)256);
//DB조회 결과를 DataSet에 저장
for(int i=0;i<list.size();i++) {
    int row = ds.newRow();           //DataSet에 행을 추가하고 그 index를 row변수에 저장
    //생성된 row의 컬럼에 맞춰서 결과값을 저장
    ds.set(row, "id", list.get(i).getId());
    ds.set(row, "pwd", list.get(i).getPwd());
    ds.set(row, "name", list.get(i).getName());
    ds.set(row, "age", list.get(i).getAge());
    ds.set(row, "address", list.get(i).getAddress());
    ds.set(row, "enrolldate", list.get(i).getEnrolldate());
}
//PlatformData는 데이터를 보관하는 기본객체임
PlatformData outData = new PlatformData();           //결과데이터를 담기위한 outData객체 생성
outData.addDataSet(ds);           //outData에 넥사크로로 전달할 DataSet인 ds 추가함
//HttpPlatformResponse : XML Format Data를 출력하는 output객체
HttpPlatformResponse pRes = new HttpPlatformResponse(response, PlatformType.CONTENT_TYPE_XML,"UTF-8");
pRes.setData(outData); //pRes에 outData를 저장
try {
    pRes.sendData();           //데이터 전송
} catch (PlatformException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

SelectAllServlet 작성

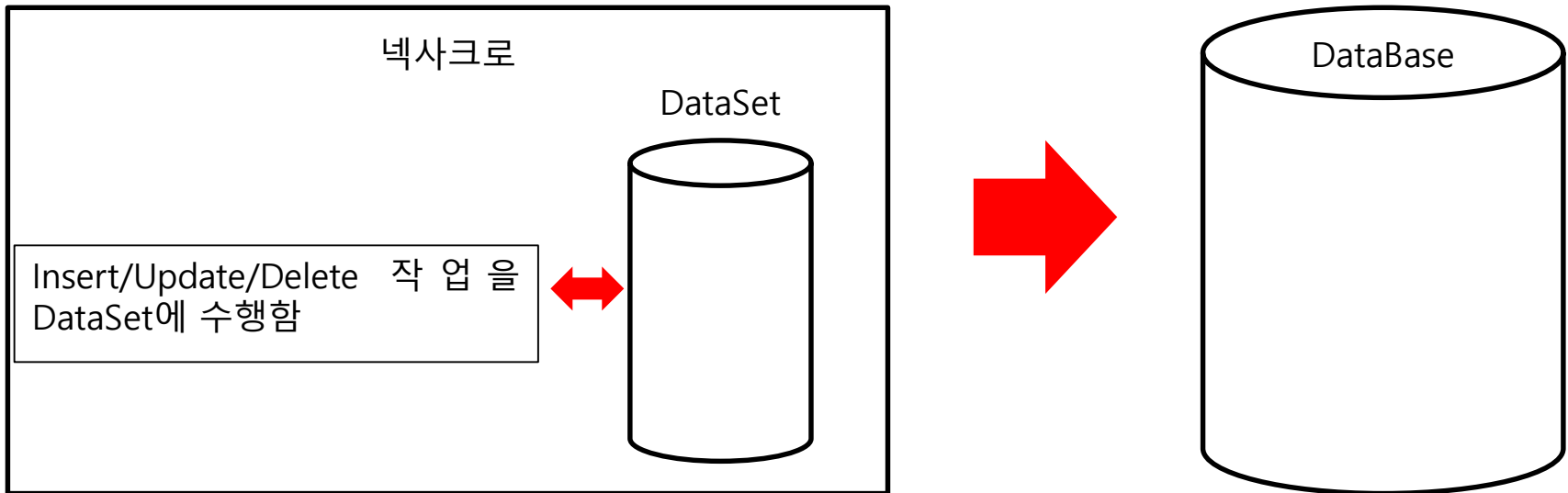
### 3. 서버 연동(select)

```
public ArrayList<NXCRMMember> selectAll(){
    Connection conn = JDBCTemplate.getConnectionNXCR();
    ArrayList<NXCRMMember> list = new NXCRMMemberDao().selectAll(conn);
    JDBCTemplate.close(conn);
    return list;
}
```

```
public ArrayList<NXCRMMember> selectAll(Connection conn){
    Statement stmt = null;
    ResultSet rset = null;
    String query = "select * from nxcr_member";
    ArrayList<NXCRMMember> list = null;
    try {
        stmt=conn.createStatement();
        rset=stmt.executeQuery(query);
        list=new ArrayList<NXCRMMember>();
        while(rset.next()) {
            NXCRMMember m = new NXCRMMember();
            m.setId(rset.getString("id"));
            m.setPwd(rset.getString("pwd"));
            m.setName(rset.getString("name"));
            m.setAge(rset.getInt("age"));
            m.setAddress(rset.getString("address"));
            m.setEnrolldate(rset.getDate("enrolldate"));
            list.add(m);
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        JDBCTemplate.close(rset);
        JDBCTemplate.close(stmt);
    }
    return list;
}
```

Service, DAO 작성(기존 작성법과 동일함)

### 3. 서버 연동



Insert, update, delete는 각 건마다 DB에 요청하는 방식이 아닌 넥사크로 내부의 DataSet에 insert, update, delete를 수행하고 DataSet과 DB를 동기화하는 방식 (한번의 Servlet으로 가능함)



### 3. 서버 연동

```
this.Div00_Static03_onclick = function(obj:nexacro.Static,e:nexacro.ClickEventInfo)
{
    if(confirm("동기화 하시겠습니까?")){
        this.transaction(
            "sync",                                //요청 ID
            "http://localhost/first/syncDB",        //요청 URL
            "in_ds=gds_member:A",                  //현재 gds_member를 전송
            "",
            "",
            "tr_callbackFunc"                      //성공 실패 여부를 판단할 콜백 함수
        )
    }
    this.selectAllFunc();
};
this.tr_callbackFunc=function(id,errCd,errMsg){
    alert(errMsg);                                //넘어온 메시지를 alert함
}
```

insert, update, delete가 발생한 gds\_member를 transaction메소드를 통해서 Servlet을 호출하면서 전달함.  
동기화 이후에 다시 조회해오는 구조임

### 3. 서버 연동

```
int totalQueryCount=0;
int successCount=0;
int failCount=0;
int nErrorCode=0;
String strErrMsg = "";
int result = 0;
HttpPlatformRequest pReq = new HttpPlatformRequest(request);
try {
    pReq.receiveData();
} catch (PlatformException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
PlatformData inData = pReq.getData();
DataSet ds=inData.getDataSet("in_ds");
//DataSet의 메소드를 이용하여 Delete/Insert/Update를 구분 할 수 있기때문에 한번의 Servlet 요청으로 처리가 가능함

//Insert, Update, Delete가 발생한 횟수
//Insert, Update, Delete가 성공한 횟수
//Insert, Update, Delete가 실패한 횟수
//callback 함수에서 사용 할 에러코드(성공실패 구분)
//callback함수에서 사용 할 메세지(실패 시 경고창에 넣을 메세지)
//DB 작업결과를 저장하는 변수
//XML포맷 데이터를 읽는 객체 생성

//XML 데이터를 분석

//PlatformData 형태로 저장
//백사크로에서 전달해 준 DataSet을 저장
```

사용할 변수생성 및 gds\_member를 읽어오는 과정

### 3. 서버 연동

```
//insert와 update 로직
for(int i=0;i<ds.getRowCount();i++) {
    int rowType = ds.getRowType(i);
    try {
        String id = ds.getString(i, "id");
        String pwd = ds.getString(i, "pwd");
        String name = ds.getString(i, "name");
        int age = Integer.parseInt(ds.getString(i, "age"));
        String address = ds.getString(i, "address");
        String tmp = ds.getString(i, "enrolldate");
        Date enrolldate = Date.valueOf(tmp.substring(0, 4)+"-"+tmp.substring(4, 6)+"-"+tmp.substring(6, 8));
        NXCRMMember m = new NXCRMMember(id, pwd, name, age, address, enrolldate);
        //rowType에 따라 NXCRMMemberService의 다른 메소드를 호출함(insert, update가 아닌 경우에는 DB작업을 진행하지 않음)
        if(rowType == DataSet.ROW_TYPE_INSERTED) {
            totalQueryCount++;
            result = new NXCRMMemberService().insertMember(m);
            if(result>0) {
                successCount++;
            }else {
                failCount++;
            }
        }else if(rowType == DataSet.ROW_TYPE_UPDATED){
            totalQueryCount++;
            result = new NXCRMMemberService().updateMember(m);
            if(result>0) {
                successCount++;
            }else {
                failCount++;
            }
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

//삭제를 제외한 나머지 전체 행의 수만큼 반복함
//insert, update, 변화없음을 구분하는 코드(0->변화없음, 1->삽입, 2->수정)

//getString메소드로 값을 꺼내와 변수에 저장(매개변수는 row번호, 컬럼명)

//String형태로 읽어온 후 Date타입으로 변환하기 위해 tmp변수에 먼저 저장

//객체를 만들어서 전달함

//insert 작업이 발생하면 DB작업 카운트 증가
//insert 처리
//결과에 따라 성공, 실패 카운트

//update 작업이 발생하면 DB작업 카운트 증가
//update 처리
//결과에 따라 성공, 실패 카운트
```

insert, update 로직

### 3. 서버 연동

```
public int insertMember(NXCRMMember m) {
    Connection conn = JDBCTemplate.getConnectionNXCR();
    int result = new NXCRMMemberDao().insertMember(conn,m);
    if(result>0) {
        JDBCTemplate.commit(conn);
    }else {
        JDBCTemplate.rollback(conn);
    }
    JDBCTemplate.close(conn);
    return result;
}

public int updateMember(NXCRMMember m) {
    Connection conn = JDBCTemplate.getConnectionNXCR();
    int result = new NXCRMMemberDao().updateMember(conn,m);
    if(result>0) {
        JDBCTemplate.commit(conn);
    }else {
        JDBCTemplate.rollback(conn);
    }
    JDBCTemplate.close(conn);
    return result;
}
```

```
public int insertMember(Connection conn, NXCRMMember m) {
    PreparedStatement pstmt = null;
    int result = 0;
    String query = "insert into nxcr_member values(?,?,?,?,?,?)";
    try {
        pstmt=conn.prepareStatement(query);
        pstmt.setString(1, m.getId());
        pstmt.setString(2, m.getPwd());
        pstmt.setString(3, m.getName());
        pstmt.setInt(4, m.getAge());
        pstmt.setString(5, m.getAddress());
        pstmt.setDate(6, m.getEnrolldate());
        result = pstmt.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }finally {
        JDBCTemplate.close(pstmt);
    }
    return result;
}

public int updateMember(Connection conn, NXCRMMember m) {
    PreparedStatement pstmt =null;
    int result = 0;
    String query="update nxcr_member set name=?, age=?, address=? where id=?";
    try {
        pstmt = conn.prepareStatement(query);
        pstmt.setString(1, m.getName());
        pstmt.setInt(2, m.getAge());
        pstmt.setString(3, m.getAddress());
        pstmt.setString(4, m.getId());
        result = pstmt.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }finally {
        JDBCTemplate.close(pstmt);
    }
    return result;
}
```

insert, update의 Service, Dao 작성

### 3. 서버 연동

```
//삭제로직
for(int i=0;i<ds.getRemovedRowCount();i++) {
    totalQueryCount++;
    String delId = ds.getRemovedData(i, "id").toString();
    result = new NXCRMService().deleteMember(delId);
    if(result>0) {
        successCount++;
    }else {
        failCount++;
    }
}
//에러코드 및 메시지 작성
strErrMsg = "요청 : "+totalQueryCount+"\n성공 : "+successCount+"\n실패 : "+failCount;    //요청/성공/실패 카운트를 메시지에 저장
if(failCount>0) {
    nErrorCode=-1;
}
```

//getRemovedRowCount() -> 지워질 행 갯수  
//Delete 발생 시 total카운트 증가  
//삭제되는 DataSet의 id컬럼값을 저장(PrimaryKey인 id를 읽어옴)  
//삭제 서비스 호출  
//결과에 따라서 성공,실패 카운트  
//DB작업이 1개라도 실패하면 에러코드를 음수로 변경 함

삭제로직 요청

insert, update, delete가 끝나면 총 요청 수와 성공/실패 수를 Msg로 전달

### 3. 서버 연동

```
public int deleteMember(String delId) {
    Connection conn = JdbcTemplate.getConnectionNXCR();
    int result = new NXCRMemberDao().deleteMember(conn, delId);
    if(result>0) {
        JdbcTemplate.commit(conn);
    }else {
        JdbcTemplate.rollback(conn);
    }
    JdbcTemplate.close(conn);
    return result;
}
```

```
public int deleteMember(Connection conn, String delId) {
    PreparedStatement pstmt = null;
    int result = 0;
    String query = "delete from nxcr_member where id=?";
    try {
        pstmt=conn.prepareStatement(query);
        pstmt.setString(1, delId);
        result = pstmt.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }finally {
        JdbcTemplate.close(pstmt);
    }
    return result;
}
```

Service, DAO 작성

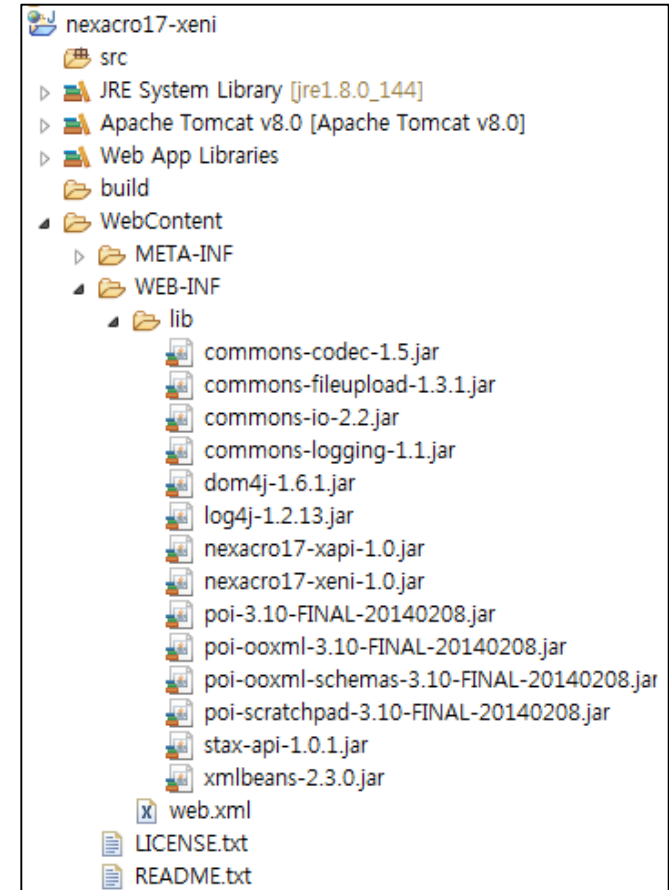
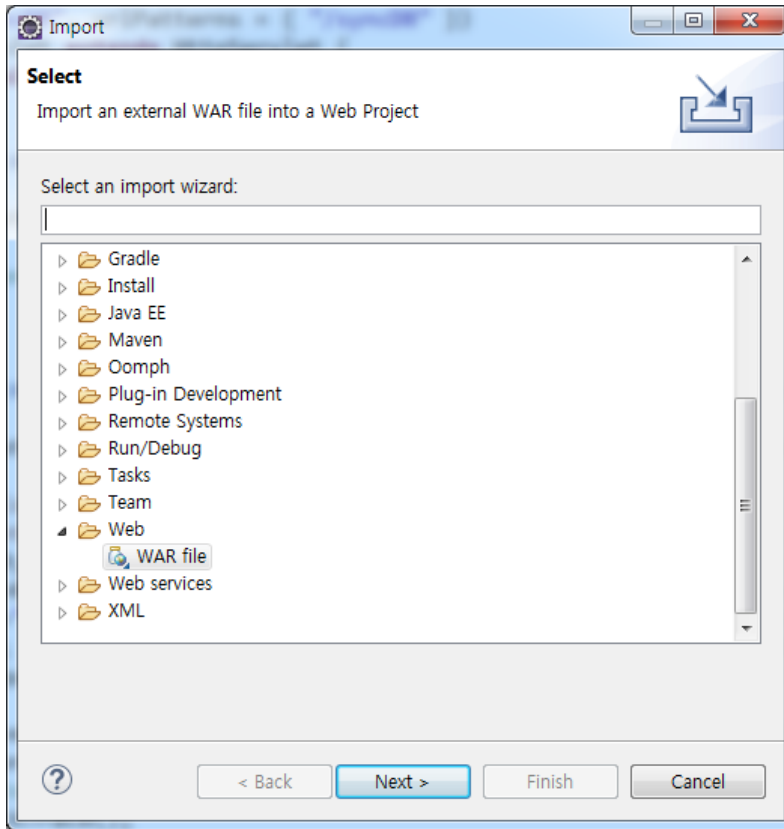
### 3. 서버 연동

```
//PlatformData는 데이터를 보관하는 기본객체
PlatformData outData = new PlatformData();
VariableList outVarList = new VariableList();
outVarList.add("ErrorCode", nErrorCode);
outVarList.add("ErrorMsg", strErrMsg);
outData.setVariableList(outVarList);
//HttpPlatformResponse : XML Format Data를 출력하는 output객체
HttpPlatformResponse pRes = new HttpPlatformResponse(response, PlatformType.CONTENT_TYPE_XML, "UTF-8");
pRes.setData(outData);
try {
    pRes.sendData();
} catch (PlatformException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

//결과데이터를 담기 위한 outData객체 생성  
//에러코드와 메시지 변수를 전송하기 위한 List객체 생성  
//list객체에 에러코드 추가  
//list객체에 에러메세지 추가  
//outData에 추가된 list객체를 저장  
//pRes에 outData를 저장  
//데이터 전송

View페이지로 내용을 전달하는 코드

### 3. 서버 연동(EXCEL)



넥사크로를 사용하면 EXCEL파일의 Export/Import가 손쉽게 가능  
사용하기 위해서는 nexacro-xeni 라이브러리를 사용해야 함.  
war형태로 제공하기때문에 war파일을 받아서 서버에 적용



### 3. 서버 연동

Dataset Editor [Dataset00]						
▶ Const Columns						
▼ Columns						
No	id	type	size	prop	sumtext	description
1	id	STRING	20			
2	pwd	STRING	20			
3	name	STRING	15			
4	age	INT	256			
5	address	STRING	50			
6	enrolldate	DATE	256			

DB에서 원하는 정보만 조회해와서 저장할 DataSet 생성(Dataset00 품 DataSet)

### 3. 서버 연동

The screenshot shows a 'New Form' window with a form titled 'EXCEL'. The form contains a table with columns 'id', 'name', 'age', 'address', and 'enrolldate'. Below the table are 'EXPORT' and 'IMPORT' buttons, and a search bar with a '검색' button. A date field 'enrolldate' is shown with a calendar icon. Red boxes highlight the table, the 'EXPORT' and 'IMPORT' buttons, and the 'enrolldate' field. Arrows point from these elements to explanatory text boxes.

Binding

binddataset	Dataset00
formatid	
formats	<Formats> <Format id="default"> <Column

gds\_search와 binding

Excel로 Export/Import하기 위한 버튼

col4		autosizerow	default
enrolldate		controlautosizingtype	both
0000-00		displaytype	calendarcontrol

검색결과 나타나는 이전, 다음버튼도 삭제  
enrolldate의 셀서식 변경할 것

### 3. 서버 연동

```
this.Div00_Div00_Button00_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    var key = this.Div00.form.key.text;           //key의 값 읽어와서 변수에 저장
    var value = this.Div00.form.Div00.form.Edit00.text; //검색어 읽어와서 변수에 저장
    if(key == "none"){ //아이디나 이름을 선택하지 않고 검색한 경우
        alert("아이디 또는 이름을 선택하세요");
    }else{
        this.transaction(
            "selectOne",           //요청 ID
            "http://localhost/first/searchKeyword", //요청 Servlet
            "",                    //reqDs 없음
            "Dataset00=out_ds",    //outDs로 결과를 받아서 gds_search에 저장
            "type="+key+" keyword="+value, //검색타입과 검색값을 파라미터 변수로 보냄
            ""
        )
    }
};
```

검색 버튼 클릭 시 DB에서 조회해서 결과를 가지고 옴

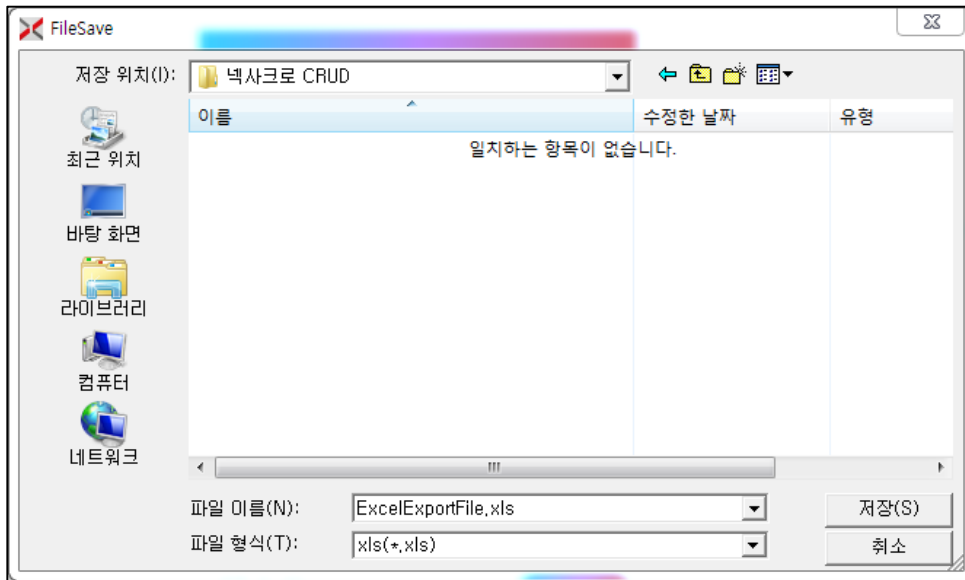
### 3. 서버 연동

```
this.Div00_Button00_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    this.exportObj = new ExcelExportObject("Export00",this);
    this.exportObj.set_exportfilename("ExcelExportFile");
    this.exportObj.set_exporturl("http://localhost/nexacro17-xeni/XExportImport");
    this.exportObj.addExportItem(nexacro.ExportItemTypes.GRID, this.Div00.form.Grid00, "Sheet1!A1");
    this.addEventHandler("onsuccess",this.Export00_onsuccess, this);
    this.addEventHandler("onerror", this.Export00_onerror, this);
    var intExportedItem = this.exportObj.exportData();
    trace("Number of Exported Item : "+intExportedItem);
};
this.Export00_onsuccess = function(obj:ExcelExportObject, e:nexacro.ExcelExportEventInfo)
{
    trace("Export00_onsuccess");
}
this.Export00_onerror = function(obj:ExcelExportObject, e:nexacro.ExcelExportEventInfo)
{
    trace("Export00_onerror");
}
```

//엑셀 내보내기 객체 생성  
//내보낼 파일 이름  
//Export를 처리할 URL(war로 배포된 서버를 서비스에 올림)  
//Grid데이터를 첫번째 시트 A1 cell부터  
//성공 시 이벤트  
//에러 시 이벤트  
//Export파일 수 확인  
//출력

Export 버튼 클릭시 Export객체를 생성하여 내보내는 함수 작성

### 3. 서버 연동



	A	B	C	D	E
1	id	name	age	address	enrolldate
2	test01	테스트	20	서울	2018-10-25 목
3	test02	테스트	20	동서울	2018-10-25 목
4	test03	테스트	222	서울	2019-01-04 금
5	test04	테스트	11	강원도	2019-01-04 금

엑셀로 저장되며, 저장파일을 열어보면 grid와 동일한 형태임

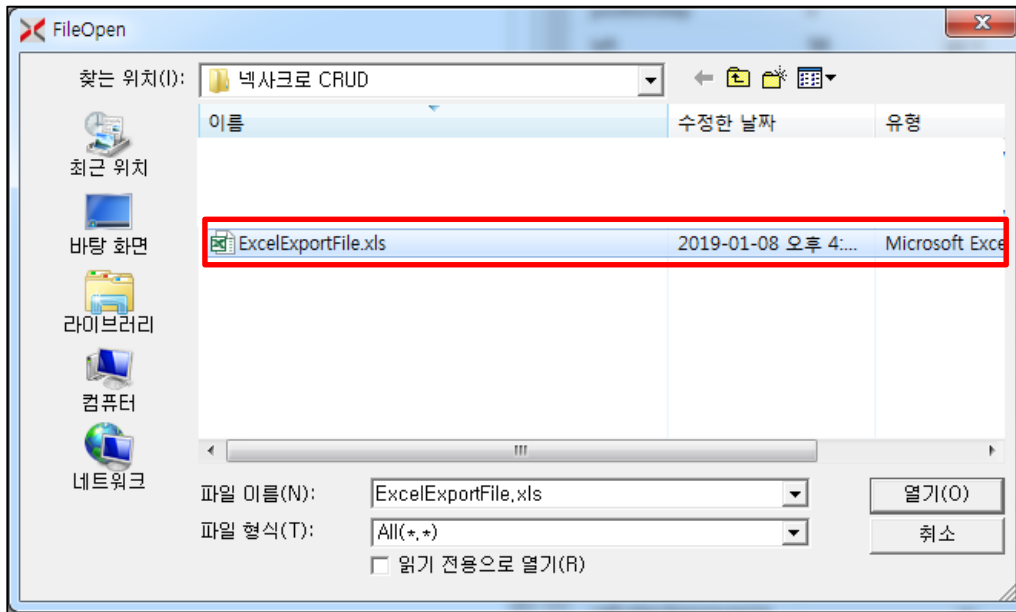
### 3. 서버 연동

```
this.Div00_Button01_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    this.importObj = new ExcelImportObject("Import00", this);           //Excel Import객체 생성
    this.importObj.set_importtype(nexacro.ImportTypes.EXCEL);         //불러오는 파일 타입 설정
    this.importObj.set_importurl("http://localhost/nexacro17-xeni/XImport"); //Import를 처리할 URL(war로 배포된 서버를 서비스에 올림)
    this.importObj.addEventHandler("onsuccess",this.Import00_onsuccess, this); //성공 시 이벤트
    this.importObj.addEventHandler("onerror",this.Import00_onerror, this); //실패 시 이벤트
    this.importObj.importData("", "Sheet1!A1:E5", "Dataset00");        //DataSet00에 Import함
};
this.Import00_onsuccess = function(obj:ExcelImportObject, e:nexacro.ExcelImportEventInfo)
{
    trace("Import00_onsuccess");

    this.Div00.form.Grid00.createFormat();
}
this.Import00_onerror = function(obj:ExcelImportObject, e:nexacro.ExcelImportErrorEventInfo)
{
    trace("Import00_onerror");
}
```

Dataset00에 엑셀파일을 읽어와서 Import하면 바인드 된 Grid에 자동으로 표시

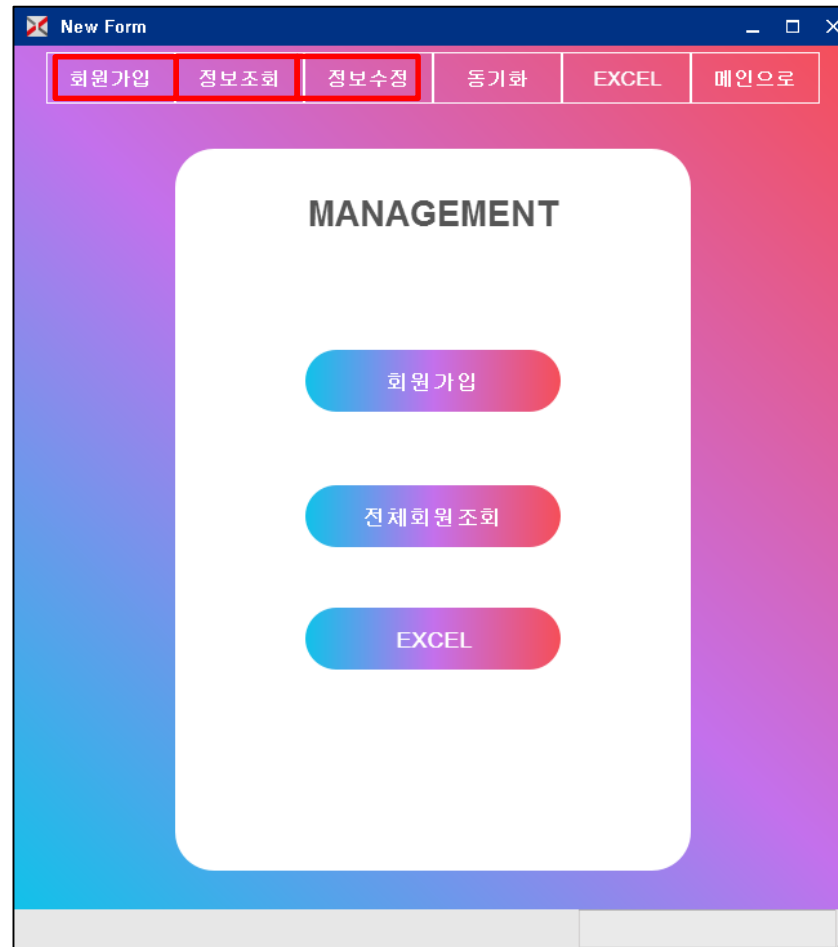
### 3. 서버 연동



회원 목록				
id	name	age	address	enrollda
test01	테스트	20	서울	2018-10-2
test02	테스트	20	동서울	2018-10-2
test03	테스트	222	서울	2019-01-0
test04	테스트	11	강원도	2019-01-0
EXPORT			IMPORT	

내보냈던 파일 Export파일을 다시 Import해서 Grid에 표현

### 3. 서버 연동



메인페이지를 생성(기능은 동일하지만 상위요소를 선택하는 것을 해보기 위함)  
회원가입, 전체회원조회, EXCEL을 main폼에 접근하여 클릭  
main페이지 시작 시 div01의 url을 이페이지로 설정



### 3. 서버 연동

```
this.onmouseenter = function(obj:nexacro.Button,e:nexacro.MouseEventInfo)
{
    this.Div00.form.Button00.set_font('normal 900 15pt/normal "Arial"');
};

this.onmouseleave = function(obj:nexacro.Button,e:nexacro.MouseEventInfo)
{
    this.Div00.form.Button00.set_font('normal 600 13pt/normal "Arial"');
};
//this는 index이며 parent는 Div01, parent.parent는 main이 된다.
//main.Div00_Static00 onclick();을 클릭하는 구조
this.Div00_Button00_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    this.parent.parent.Div00_Static00_onclick();
};

this.Div00_Button01_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    this.parent.parent.Div00_Static01_onclick();
};

this.Div00_Button02_onclick = function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
{
    this.parent.parent.Div00_Static04_onclick();
};
```

this는 현재 form을 의미하며 parent는 상위요소를 의미함

### 3. 서버 연동

url	content::index.xfdl
-----	---------------------

```
this.Div00_Static05_onclick = function(obj:nexacro.Static,e:nexacro.ClickEventInfo)
{
    this.Div00.form.Static00.set_border("1px solid white");
    this.Div00.form.Static01.set_border("1px solid white");
    this.Div00.form.Static02.set_border("1px solid white");
    this.Div00.form.Static04.set_border("1px solid white");
    this.Div01.set_url("content::index.xfdl");
};
```

```
var addRow = nexacro.Application.gds_member.addRow();           //gds_member에 row 추가
nexacro.Application.gds_member.setColumn(addRow,"id",id);       //id값 입력
nexacro.Application.gds_member.setColumn(addRow,"pwd",pwd);     //pw값 입력
nexacro.Application.gds_member.setColumn(addRow,"name",name);   //name값 입력
nexacro.Application.gds_member.setColumn(addRow,"age",age);      //age값 입력
nexacro.Application.gds_member.setColumn(addRow,"address",address); //address값 입력
var d = new Date();                                             //Date를 이용하여 오늘 날짜를 구함
var yyyy = d.getFullYear();
var mm = (d.getMonth()+1);
var dd = d.getDate();
mm = (mm>9?'':'0')+mm;
dd = (dd>9?'':'0')+dd;
nexacro.Application.gds_member.setColumn(addRow,"enrolldate",yyyy+mm+dd); //enrolldate값 입력
alert("회원추가완료");
this.parent.parent.Div00_Static05_onclick();
```

최초 main페이지 로드 시 index폼 연결  
메인으로 클릭 시 index폼 연결  
회원가입 완료 후 index폼으로 이동