Chapter 8 Frame 설정하기

1. 프레임을 이용한 화면배치

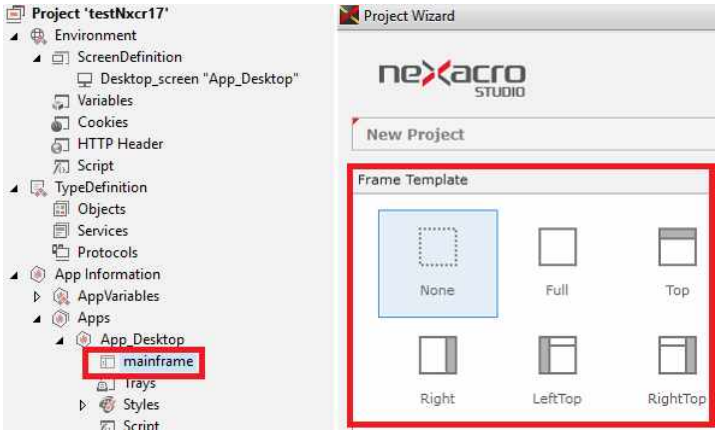
넥사크로플랫폼 애플리케이션은 화면 처리를 위한 기본 프레임 구조를 제공하는데, 최상위 MainFrame 아래에 ChildFrame 을 가지고 그 아래에 화면(Form)을 배치한다. 프레임은 Form 을 보여주기 위한 무대장치라고 할 수 있다. 각 화면을 이루는 Form 이 따로 떨어져 있지만 같은 무대 위에서 보여주기 때문에, 공통으로 사용할 수 있는 요소를 배치해 화면 간 다양한 요소를 공유할 수 있다.

기본 제공되는 프레임은 MainFrame, ChildFrame으로 나뉘며 화면 레이아웃에 따라 FrameSet, VFrameSet, HFrameSet, TileFrameSet으로 구분 지을 수 있다.

프레임 종류	Frame	설명
Root Frame	MainFrame	- 최상위 프레임 - 하위로 Node Frame 또는 ChildFrame을 가질 수 있다 - 하위에 ChildFrame을 가진 경우, 프레임을 추가할 수 없다
Terminal Frame	ChildFrame	- 하위로 어떤 프레임도 가질 수 없다 - 하위로 하나의 폼만 가질 수 있다
Node Frame	FrameSet	- 특별한 형태 없이 하위 프레임을 배치 - 2개 이상의 하위 프레임이 추가되면 계단식으로 배치하며 위치 속성값을 지정하면 해당 위치에 배치 - 하위로 Node Frame 또는 ChildFrame을 가질 수 있다
	VFrameSet	- 세로 형태로 하위 프레임을 배치 - separatesize 속성으로 하위 프레임 배치 비율을 지정 - 하위로 Node Frame 또는 ChildFrame을 가질 수 있다
	HFrameSet	- 가로 형태로 하위 프레임을 배치 - separatesize 속성으로 하위 프레임 배치 비율을 지정 - 하위로 Node Frame 또는 ChildFrame을 가질 수 있다
	TileFrameSet	- 바둑판 형태로 하위 프레임을 배치 - separatetype, separatecount 속성으로 가로, 세로 방향에 배치될 하위 프레임을 지정 - 하위로 Node Frame 또는 ChildFrame을 가질 수 있다

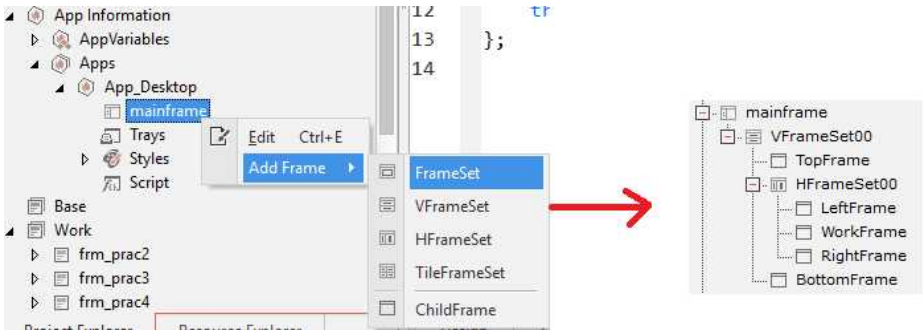
1.1 SDI (Single Document Interface)

프로젝트 생성 시 Default 템플릿을 선택하고 다른 추가적인 작업을 하지 않았다면 하나의 프레임만으로 프로젝트가 구성된다. 이 경우, MainFrame이라는 하나의 프레임 화면을 통해 Form의 구성 내용을 보여 주게 되는데 이를 SDI(Single Document Interface)라고 한다.

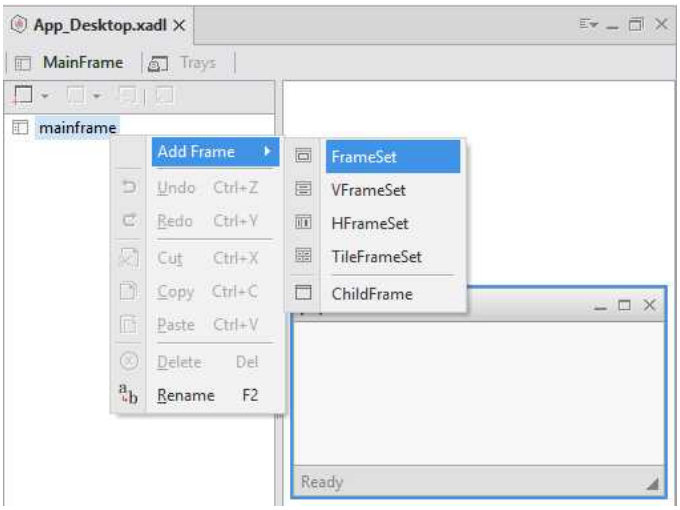


1.2 MDI (Multi Document Interface)

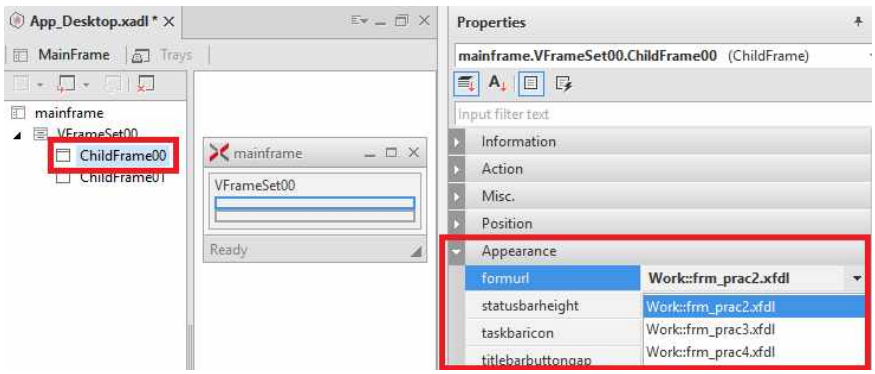
상단 메뉴, 하위 메뉴, 콘텐츠, 툴바 형식으로 화면을 구성하는 경우, 전체적인 레이아웃 구조를 원하는 형식에 맞게 구성해야 한다. 넥사크로플랫폼에서 MDI 구성은 Default 템플릿 선택 후 필요한 프레임을 추가하거나 HFrame, VFrame 템플릿 또는 사용자가 지정한 프로젝트 템플릿을 선택해 만들 수 있다.



프레임 추가는 프로젝트 탐색기에서 MainFrame 또는 Node Frame 선택 후 컨텍스트 메뉴에서 [Add Frame] 항목을 선택하거나 MainFrame 또는 Node Frame 항목을 더블 클릭 후 나타나는 편집화면 컨텍스트 메뉴에서 [Add Frame] 항목을 선택해 추가할 수 있다.



각각의 Frame 에서 각자의 폼을 연동할 경우, 해당 Frame 을 클릭한 후, Properties 창에서 Appearance > formurl 의 버튼을 클릭하여 화면에 보여질 폼을 연결하면 된다.

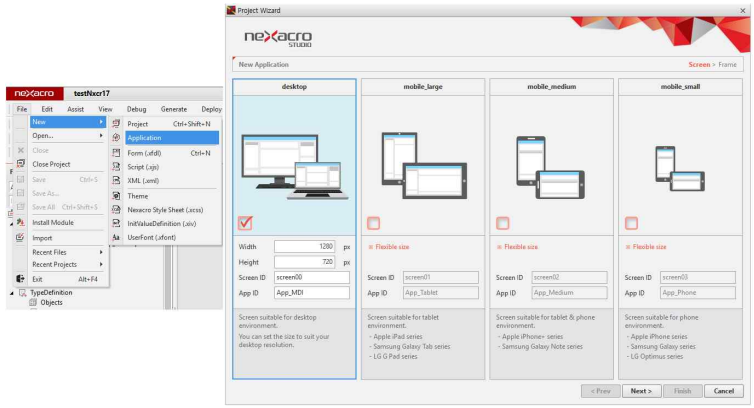


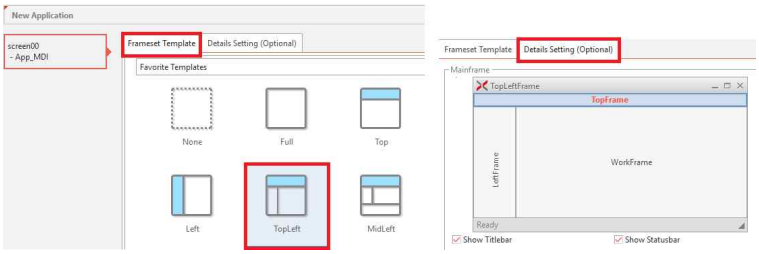
2. Frame 구조화 하기

이제까지 우리는 Form 을 작성한 후, Quick View 를 통해 페이지 구성을 테스트 하였는데, 이 방식은 MainFrame 에서 하나의 Form 만 볼 수 있는 SDI 프로젝트에 해당한다.

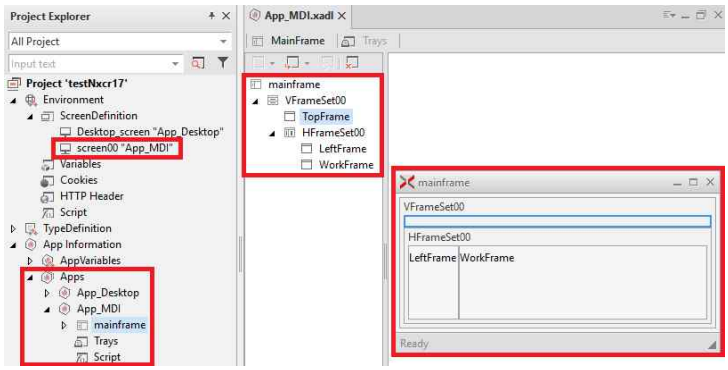
이번 실습에서는 메뉴 프레임과 컨텐츠 프레임을 나누어 각각의 화면에 내용을 별도로 보여주는 MDI 프로젝트를 만들어 보자.

먼저 실습을 위한 새로운 Application 'App_MDI'를 생성한다.





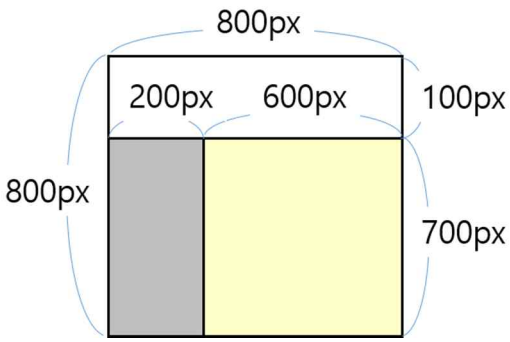
** Next 이후 Frameset Template 에서 TopLeft를 선택하자. Detail Setting 에서 표현될 화면의 내용을 미리 확인해 볼 수 있다.



VFrameSet, HFrameSet, FrameSet의 구성과 같이 하나의 FrameSet에는 반드시 1개 이상의 ChildFrame이 존재해야 하며, 우리가 실제 작성한 Form의 내용이 해당 ChildFrame에 위치하게 된다.

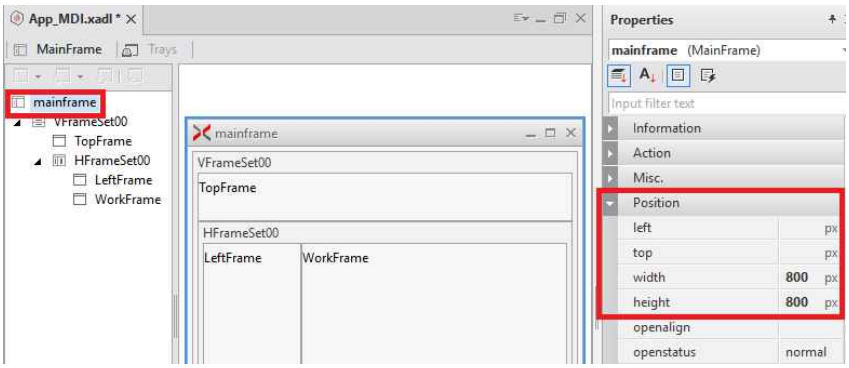
2.1 Frame 구조 설계

각 FrameSet의 크기를 Properties에서 설정해보자.



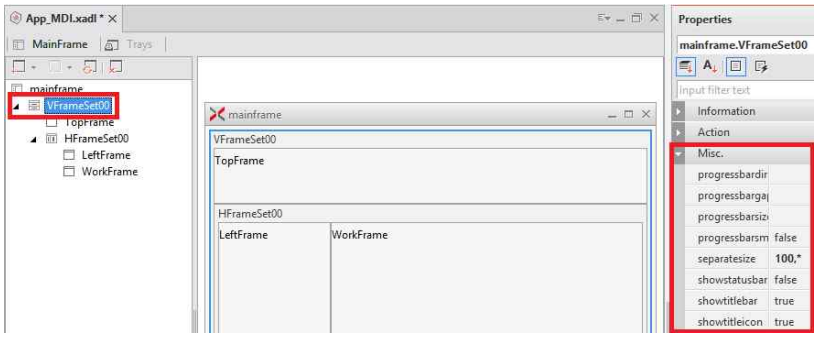
2.1.1 MainFrame

MainFrame을 선택하여 Properties > Position 탭에서 width와 height 속성을 각각 800px로 설정한다.



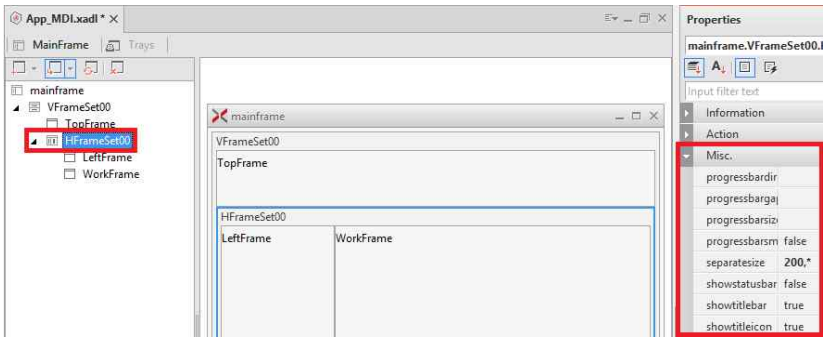
2.1.2 VFrameSet

VFrameSet0를 선택 한 후, Properties > Misc. 탭에서 separatesize 속성을 100,* 로 설정한다. '*'은 VFrameSet의 상단을 100px로 하고 남은 나머지 부분의 사이즈를 뜻한다.



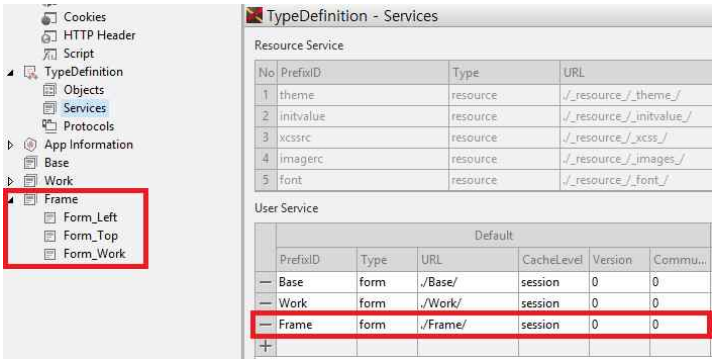
2.1.3 HFrameSet

HFrameSet0를 선택 한 후, Properties > Misc. 탭에서 separatesize 속성을 200,* 로 설정한다.



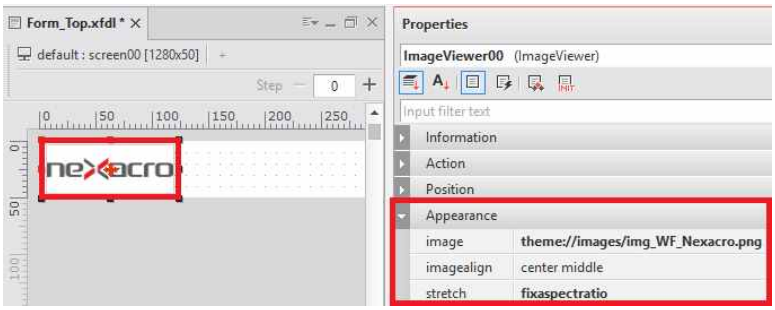
2.2 Form Menu 설계

TypeDefinition > Services 탭에서 새 서비스그룹 Frame을 추가하고, 메뉴 바의 File > New > Form으로 서비스그룹 Frame에 상단 메뉴와 좌측 메뉴를 작성하기 위한 Form_Left, Form_Top, Form_Work를 각각 생성한다. 생성 시 템플릿을 사용하여 작성하였다면 자동으로 함께 생성된다.

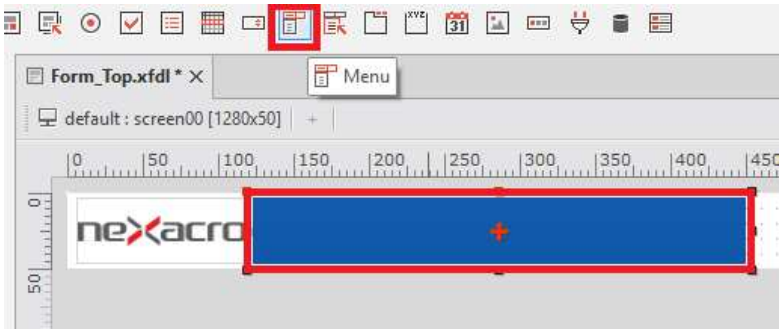


2.2.1 Form_Top 구성

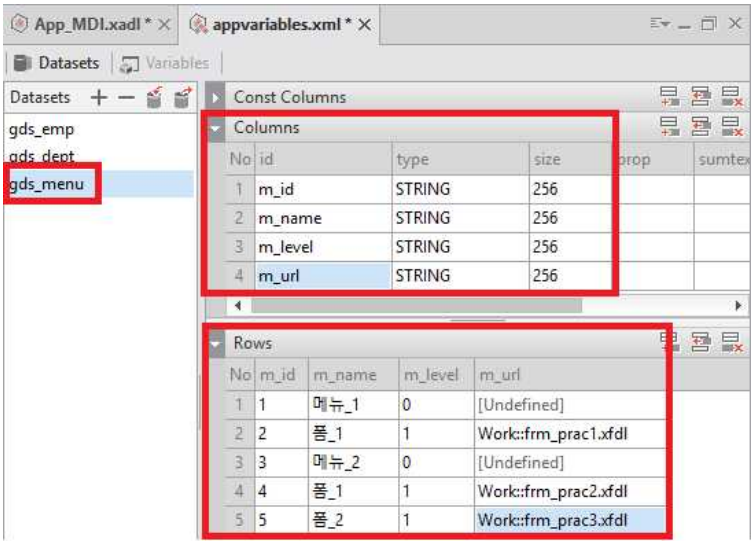
Form_Top은 본 프로젝트의 상단 주 메뉴를 담당한다. 먼저 ImageViewer를 하나 생성하고, Properties > Appearance > image 에서 ▼를 클릭하여 dropdown 메뉴로 전역변수나 theme에 저장한 이미지를 설정한다.



다음으로 상단 메뉴에서 Menu 컴포넌트를 찾아 직사각형의 형태로 ImageViewer 우측에 추가한다.



전역 변수 DataSet으로 gds_menu를 만들어 다음과 같이 내부 Data를 작성한다.



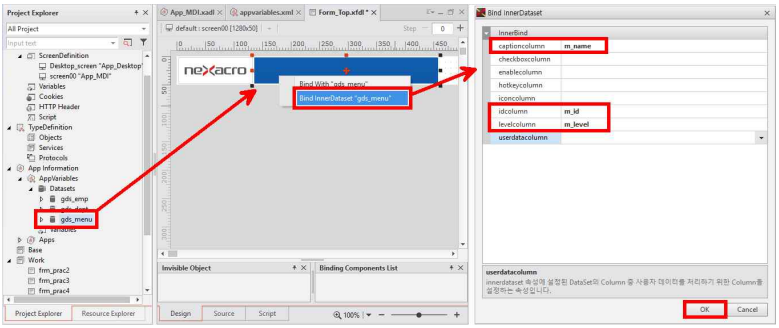
** m_url 컬럼은 메뉴별로 보여질 폼의 주소 값을 담는다.

** 주소는 '서비스그룹명::form명.xfdl' 로, Form의 실제 파일명까지 기술한다.

사용한 코드는 다음과 같다.

```
<ColumnInfo>
  <Column id="m_id" type="STRING" size="256"/>
  <Column id="m_name" type="STRING" size="256"/>
  <Column id="m_level" type="STRING" size="256"/>
  <Column id="m_url" type="STRING" size="256"/>
</ColumnInfo>
<Rows>  <Row>
    <Col id="m_id">1</Col>
    <Col id="m_name">메뉴_1</Col>
    <Col id="m_level">0</Col>
  </Row>
  <Row>
    <Col id="m_id">2</Col>
    <Col id="m_name">폼_1</Col>
    <Col id="m_level">1</Col>
    <Col id="m_url">Work::frm_prac1.xfdl</Col>
  </Row>
  <Row>
    <Col id="m_id">3</Col>
    <Col id="m_name">메뉴_2</Col>
    <Col id="m_level">0</Col>
  </Row>
  <Row>
    <Col id="m_id">4</Col>
    <Col id="m_name">폼_1</Col>
    <Col id="m_level">1</Col>
    <Col id="m_url">Work::frm_prac2.xfdl</Col>
  </Row>
  <Row>
    <Col id="m_id">5</Col>
    <Col id="m_name">폼_2</Col>
    <Col id="m_level">1</Col>
    <Col id="m_url">Work::frm_prac3.xfdl</Col>
  </Row>  </Rows>
```

DataSet의 작성이 완료되었다면, 먼저 만들어 둔 menu 컴포넌트에 드래그하여 inner Dataset 바인드를 시키고, Inner Bind Dataset 옵션을 다음과 같이 설정한다.

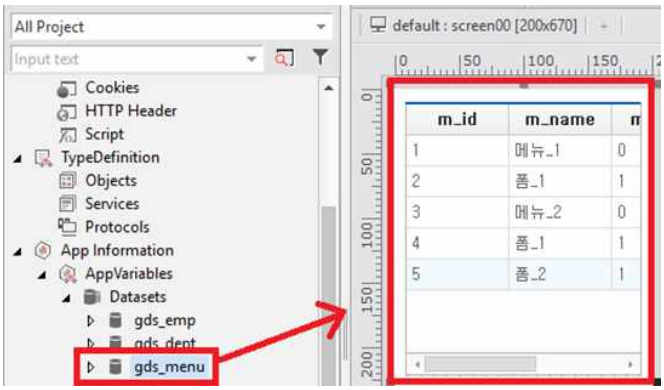


captioncolumn: 사용자에게 보여질 값 (m_name)
idcolumn: 각 메뉴의 고유 식별 값 (m_id)
levelcolumn: 메뉴 별 계층구조 구분 값 (m_level)

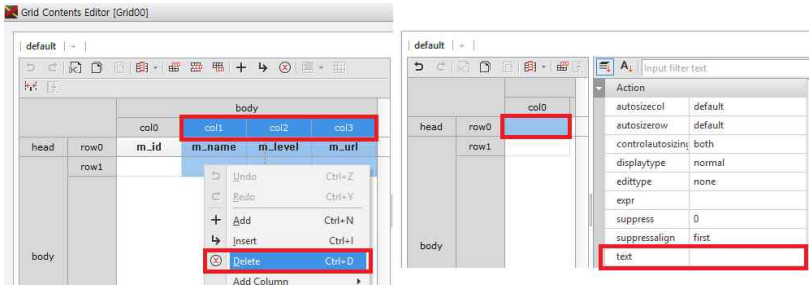
설정이 완료되었다면 generate 후에 QuickView로 결과 화면을 확인한다.
그럼 다음과 같이 화면의 상단을 구성하는 프레임이 생성된 것을 확인할 수 있다.

2.2.2 Form_Left 구성

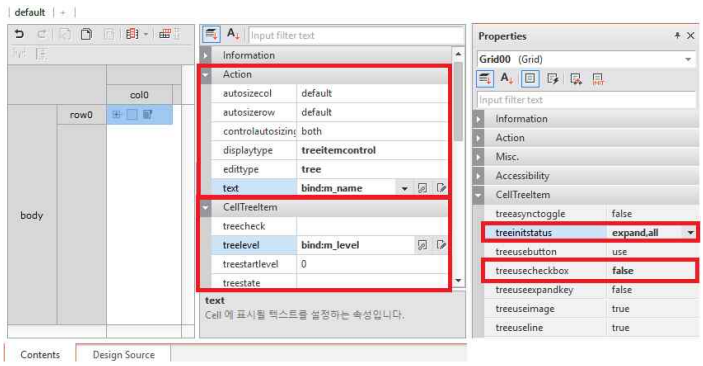
Form_Left에는 메뉴 내용을 tree 형태로 구현한다. 먼저 Grid를 하나 생성한 뒤, gds_menu를 드래그하여 바인드한다.



그 뒤 Grid를 더블 클릭하여 Grid Contents Editor를 연 뒤 다음과 같이 하나의 Body 컬럼만을 남기고 전부 delete한다.

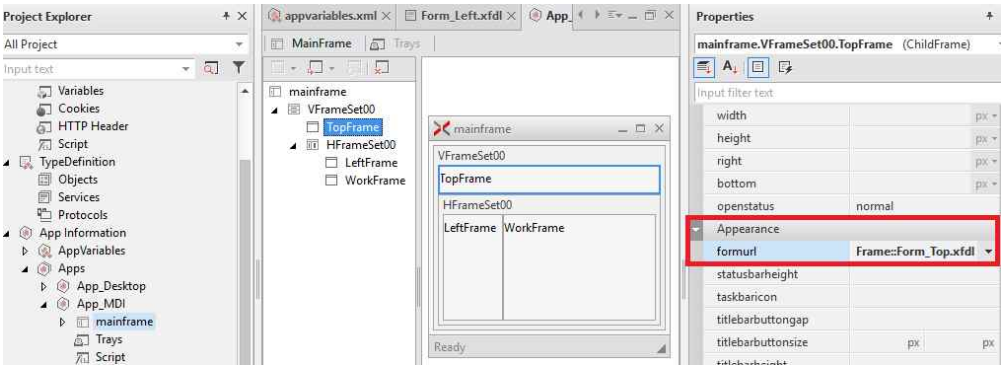


남은 한 개의 컬럼을 선택, 우측의 Properties 설정을 다음과 같이 변경하고 OK 한다. 이 후, Form에서 Grid 컴포넌트의 Properties > CellTreeItem의 속성과 생성된 grid 컴포넌트의 속성을 다음과 같이 변경한다.

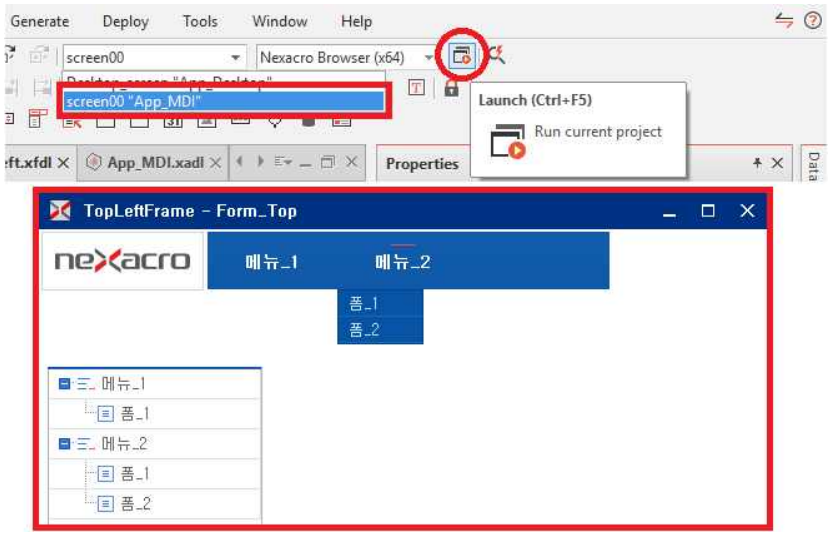


2.3 Frame – Form 간 연결 설정

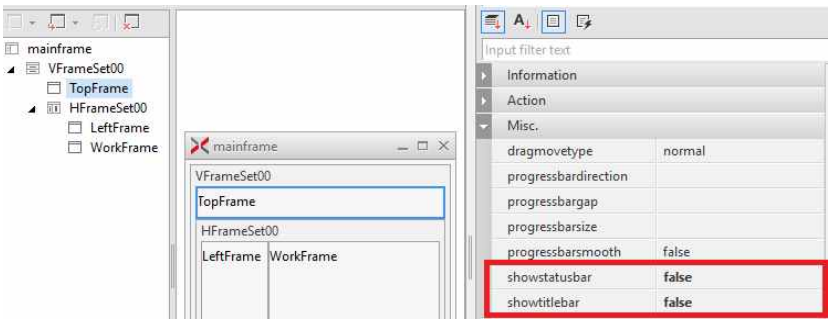
이제 구현이 완료된 From_Top과 From_Left를 각각 VFrameSet0 > ChildFrame0, HFrameSet0 > WorkFrame 의 Properties > Appearance > formurl 에 등록한다.



설정이 완료되었다면 상단 툴 바에서 단일 폼만 볼 수 있던 QuickView가 아닌 Launch Project로 프레임 전체를 확인 해 보자.



만약, 프레임 Form 상단에 Title Bar 가 표시되는 것을 원치 않는다면,
해당 ChildFrame의 Properties > Misc. 의 showtitlebar 속성을 false로 체크하면 보이지 않는다.



2.4 메뉴 클릭 이벤트 구현

이제 Top메뉴, 또는 좌측 TreeMenu에서 폼을 선택하면, Section 영역에 해당 폼을 불러오는 기능을 구현해보자. 현재 Section 영역은 FrameSet으로 지정되어 있고, WorkFrame이 존재하는 상태다. 모든 폼은 ChildFrame에서 보여져야 하며, 각각의 폼을 생성한 ChildFrame안에 formurl을 통해 연결하고, 이 ChildFrame을 다시 FrameSet에서 보이게 구현하는 과정을 거친다.

해당 기능은 자바 스크립트를 통해 구현하게 되는데, 함수의 선언 위치에 따라 3 종류로 구분할 수 있다.

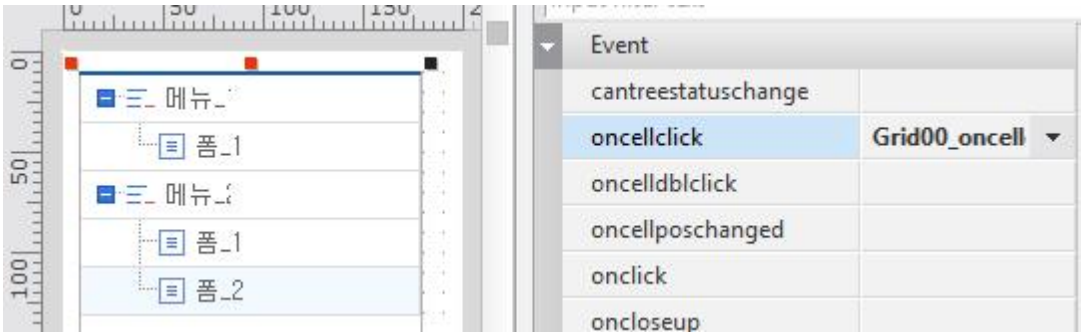
1. 폼 내부함수
2. Application 전역함수 선언 : Apps 내 스크린 설정에 스크립트 추가
3. JS 서비스그룹을 만들고, 해당 js파일을 include하기

2.4.1 폼 내부 함수 선언

form을 Section 영역에 불러오는 기능을 구현하는 순서는 다음과 같다.

1. ChildFrame이 위치할 부모 경로를 생성한다.
2. 화면에 표시할 ChildFrame을 동적 생성하고 초기화한다.
3. ChildFrame의 formurl을 설정한다.
4. 부모 경로에 만들어 둔 ChildFrame을 추가한다.
5. ChildFrame show 메소드를 호출한다.
6. 반복 호출을 체크하고, 반복 호출할 경우 포커스를 전환시키는 로직을 추가한다.

폼 스크립트에 내부 함수를 선언하기 위해 Form_Left.xfdl 로 이동하여 Grid를 선택 후, Properties > Event Info > oncellclick 이벤트를 더블 클릭하여 이벤트 핸들러를 선언한다.



코드 작성에 앞서, 우리가 알아야 할 것은 해당 메뉴 row의 m_url값과 m_id값, 각각의 ChildFrame이 구분하기 위한 ChildFrameID를 위한 고유값이다. 우리가 Grid 내의 메뉴를 선택했을 때 해당 메뉴의 url 값과 id 값을 찾을 수 있어야 그 값에 맞는 form을 화면에 표시할 수 있기 때문이다.

다음의 코드는 이러한 값을 찾기 위해 Grid에 해당 스크립트를 작성한 것이다.

- Grid Click Event

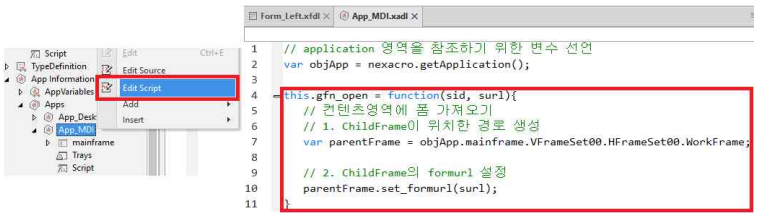
```
01 // application 영역을 참조하기 위한 변수 선언
02 var objApp = nexacro.getApplication();
03
04 this.Grid00_oncellclick = function(obj:Grid, e:nexacro.GridClickEventInfo)
05 {
06     //id와 url을 찾아 변수에 저장한다.
07     var sid = objApp.gds_menu.getColumn(e.row, "m_id");
08     var surl = objApp.gds_menu.getColumn(e.row, "m_url");
09
10     // ChildFrame설정을 위한 내부함수를 호출한다.
11     this.fn_open(sid,surl);
12 }
```

- fn_open (해당 Form Script탭의 grid click event 밑에 작성)

```
01 this.fn_open = function(sid, surl){
02     // 컨텐츠영역에 폼 가져오기
03     // 1. ChildFrame이 위치한 경로 생성
04     var parentFrame =
05         objApp.mainframe.VFrameSet00.HFrameSet00.WorkFrame;
06
07     // 2. ChildFrame의 formurl 설정
08     parentFrame.set_formurl(surl);
09 }
```

2.4.2 Application 전역함수 선언 : ADL에 스크립트추가

이전과 같은 내용을 frmTop에도 적용하되 Application 전 범위에서 사용할 수 있는 함수를 만들어 적용해보자. 작업중인 ADL을 우클릭하고, Edit Script를 선택하여 Script창을 연다. 그 후 이전에 작업한 fn_open 함수를 복사해서 함수명을 gfn_open으로 변경 후 붙여 넣는다.

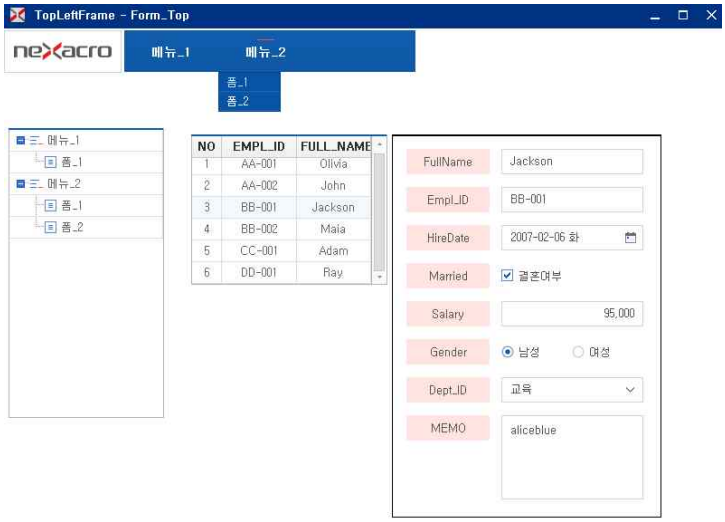


다음으로 Form_Top.xfdl을 열고, Form Design 창의 menu 컴포넌트를 더블 클릭하여 click 이벤트 핸들러를 추가한다.

- Menu Click Event

```
01 var objApp = nexacro.getApplication();
02
03 this.Menu00_onmenuclick =
04       function(obj:nexacro.Menu,e:nexacro.MenuClickEventInfo)
05 {
06     //id와 url을 찾아 변수에 저장
07     //innerDataset으로 바인딩 하였다면 바로 접근이 가능하다
08     var sid = e.id;
09
10     // Dataset.lookup(strColID,varCmpVal,strOutputColID);
11     // sid에 해당하는 열의 url값을 찾는 함수
12     // 엑셀의 lookup 함수와 유사하다.
13     var surl = objApp.gds_menu.lookup("m_id", sid, "m_url");
14
15     // ADL 전역변수에 선언된 함수
16     objApp.gfn_open(sid, surl);
17 };
```

해당 변경 사항을 저장한 후, generate > Launch Project 메뉴 기능을 확인해보자.



2.4.3 JS 서비스그룹을 만들고, 해당 js파일을 include하기

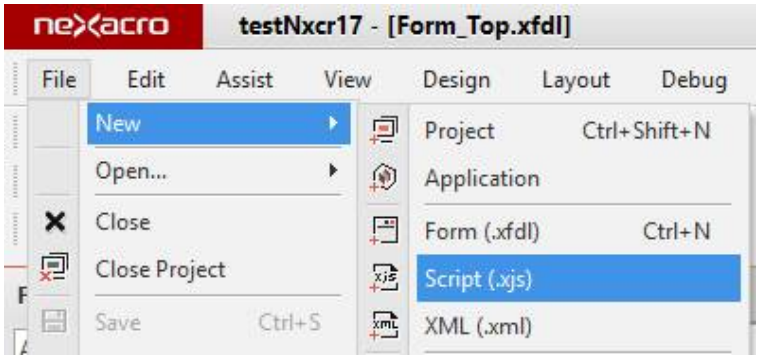
여러 스크립트를 공용으로 함께 사용하고자 한다면 별도의 서비스 그룹을 생성하여, 해당하는 자바 스크립트 파일들을 관리해야 한다. 이번 파트에서는 스크립트 전용 서비스 그룹을 만들고 이를 include하여 사용하는 방법에 대해 실습하겠다.

먼저 TypeDefinition > Services 탭에서 'xLib' 새 서비스그룹 'js'를 추가한다.

User Service

	Default					
	PrefixID	Type	URL	CacheLevel	Version	Commu...
—	Base	form	./Base/	session	0	0
—	Work	form	./Work/	session	0	0
—	commonJS	js	./commonJS/	session	0	0
—	Frame	form	./Frame/	session	0	0
—	xLib	js	./xLib/	session	0	0

xLib 서비스 그룹이 생성되었다면, 메뉴 바에서 File > New > File > Script(.xjs)를 클릭하여 menu라고 이름을 지은 스크립트 파일을 xLib 경로에 생성한다.



파일 생성을 완료했다면, Project Explorer에서 xLib 서비스 그룹과 안에 생성된 menu.xjs를 확인할 수 있다. 이제 해당 스크립트에 이전의 fn_open 함수를 xfn_open 함수로 바꾸어 붙여 넣은 후 저장한다.

** 스크립트 파일을 작성하면 넥사크로플랫폼에 의해 내부적으로 '.xjs'파일로 변환되어 저장된다.

이렇게 공용 스크립트로 작성된 스크립트 파일은 해당 함수를 필요로 하는 Form에서 Script 창의 최상단에 include 구문을 기술하여 사용할 수 있다.

이를 응용하여 기존 Form_Left.xfdl 에 작성된 내부 함수를 지우고 xfn_open(sid, surl)을 include하여 수정해보자.

- Grid Click Event (include 활용)

```
01  include "xLib::menu.xjs"      // xLib의 menu.xjs 를 include
02
03  var objApp = nexacro.getApplication();
04
05  this.Grid00_oncellclick = function(obj:Grid, e:nexacro.GridClickEventInfo)
06  {
07      var sid = objApp.gds_menu.getColumn(e.row, "m_id");
08      var surl = objApp.gds_menu.getColumn(e.row, "m_url");
09  .
10      this.xfn_open(sid,surl);
11  }
```

작성한 수정사항을 저장하고, generate > Launch Project 를 통해 정상적으로 작동하는지 확인해보자.