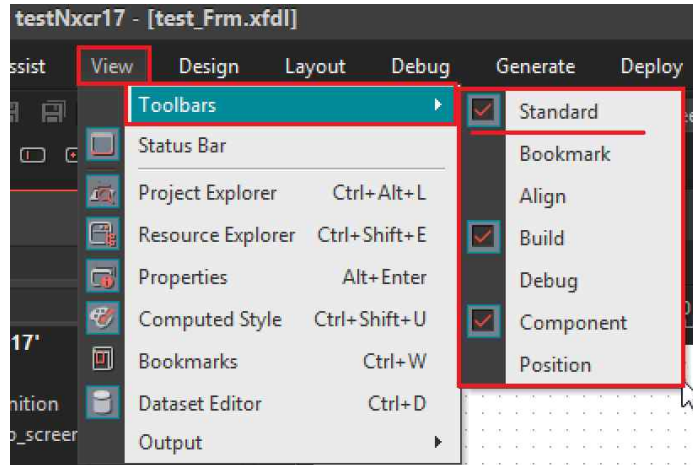



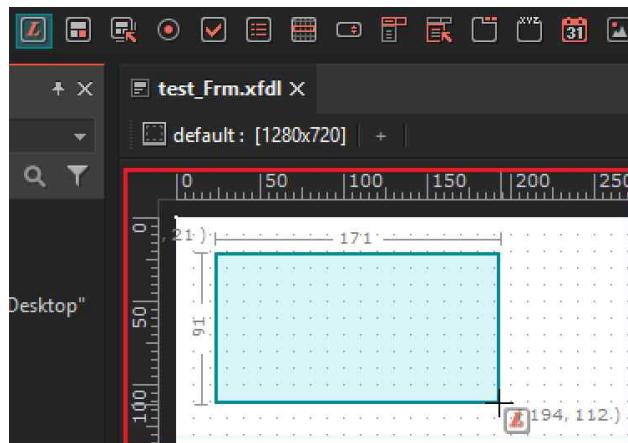
Chapter 4 컴포넌트 소개 및 화면 구성하기

1 컴포넌트 배치하기

새로운 폼이 만들어지면 Form Design 창이 활성화되고 파일 기본 저장, 복사 등과 관련된 'Standard' 툴바, 폼 디자인에 필요한 컴포넌트를 배치할 때 사용하는 'Component' 툴바, 프로젝트 js 변환과 관련된 'Build' 툴바가 활성화 된다. 다른 툴이 필요하다면 View > Toolbars 에서 원하는 툴 메뉴를 추가할 수 있다.

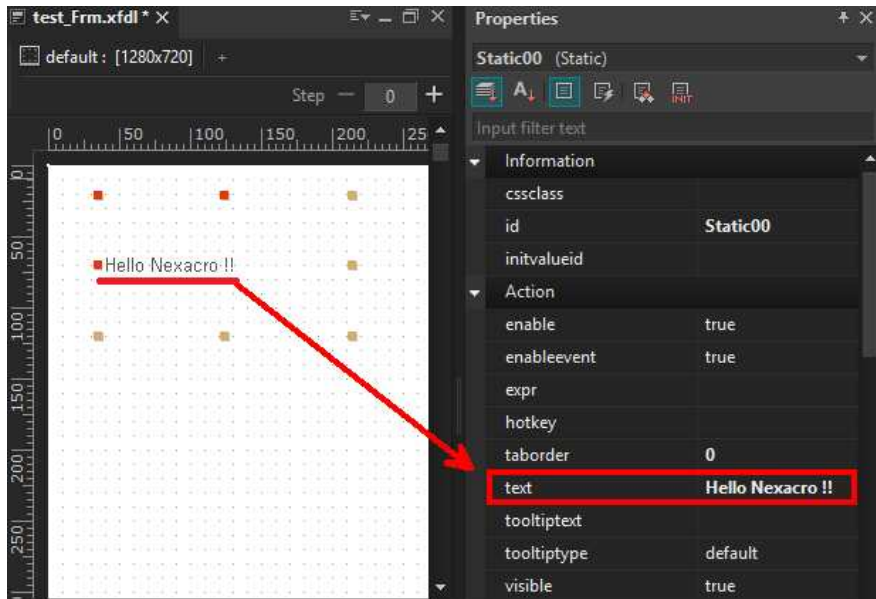


넥사크로플랫폼 애플리케이션에서 글자를 표기할 때는 Static 컴포넌트를 사용한다. 'Component' 툴바에서  모양의 Static 컴포넌트를 마우스로 선택하고 Form Design 창에서 배치하기 원하는 위치를 마우스로 클릭하면 기본값으로 정해진 크기로 Static 컴포넌트가 배치된다.

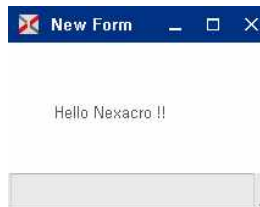


Form Design 창에서 마우스를 클릭하지 않고 마우스 왼쪽 버튼을 누른 상태에서 드래그하면 원하는 크기로 컴포넌트를 배치할 수 있다.

Static 컴포넌트의 텍스트를 수정할 때는 컴포넌트를 선택한 후 텍스트 영역을 클릭하면 편집 모드로 전환되며 직접 수정할 수 있다. Form Design 창이 아닌 속성 창에서도 해당 속성을 수정할 수 있는데, Static 컴포넌트를 선택하고 속성 창에서 text 속성 항목을 수정하면 된다.



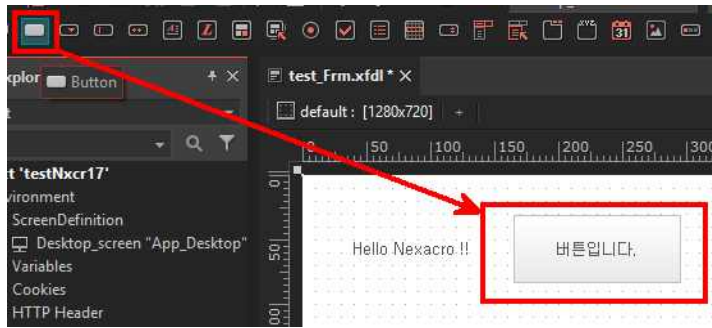
작성이 완료되었다면 Generate 후 Quick View를 통해 실행하여 해당 결과를 확인 해보자.



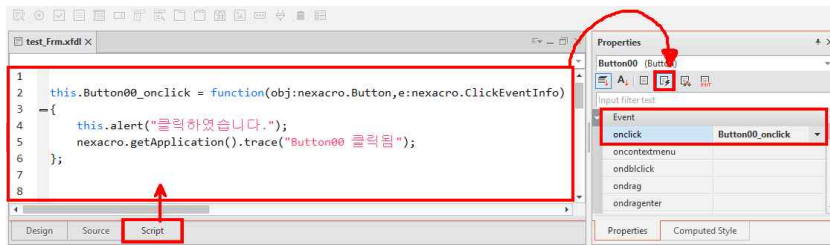
2 컴포넌트와 이벤트

앞서 만들었던 Static 컴포넌트 옆에 버튼을 하나 만들고, 해당 버튼을 클릭했을 때 이벤트가 발생하는 이벤트를 만들어 보자.

상단의 툴바에서 버튼을 클릭하고 Form Design 창을 클릭하여 버튼을 생성한다.



생성된 버튼을 더블 클릭하면 Form Script 탭에 '[Button의 ID]_onclick' 이라는 함수가 자동으로 생성되고, 해당 버튼 컴포넌트의 Properties 내 이벤트 속성에 생성된 함수가 바인딩되는 것을 볼 수 있다.



확인이 끝났으면 버튼을 클릭 했을 때 alert 창과 Output의 trace log가 발생하도록 내부 소스를 적는다.

```
this.alert("클릭하였습니다.");
nexacro.getApplication().trace("Button00 클릭됨");
```

여기서 중점으로 알아야 할 것은 this와 application 이라는 유효 범위(Scope)이다. this는 현재 Form 영역을 가리키는 Scope이고, application은 넥사크로 애플리케이션에서 사용하는 전역 Scope를 뜻한다.

보기의 alert() 함수도 Form과 Application에서 각각 제공하는 메소드이기 때문에, 앞에 Scope를 붙였다. alert()과 trace() 메소드에 한해서 범위지정자를 생략할 수 있지만, 일반적으로 함수를 선언하거나 호출 할 때, 변수를 호출할 때에는 반드시 이러한 범위를 고려하고 사용해야 한다.

이렇게 버튼 클릭 이벤트는 자동 구문생성을 통해 만들어 지지만, 직접 이벤트를 작성하여 바인딩을 하는 경우, 하기 형식으로 소스 코드를 작성해야 하며, 작성 후 이벤트 속성 창에서 해당 함수를 바인딩 하거나 EventHandler 함수를 통해 제어해야 한다.

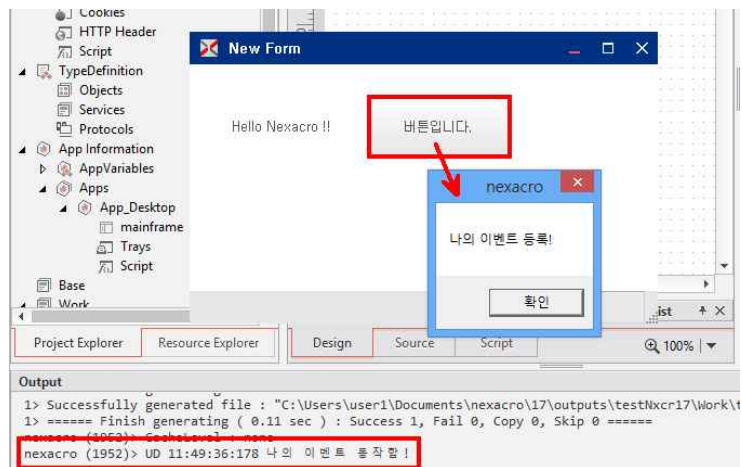


```

// addEventHandler( event명, 함수명, 대상 Form );
Ex) this.Button00.setEventHandler("onclick", this.myFunc, this);

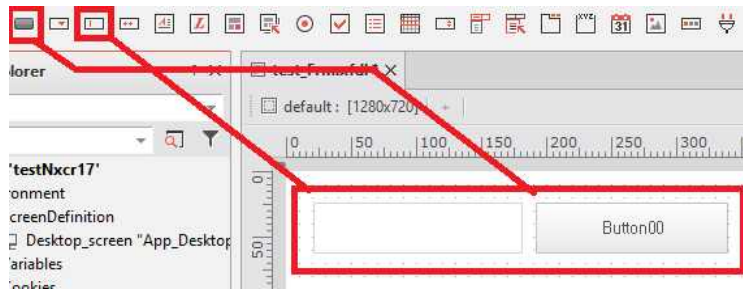
```

직접 만든 메소드로 마우스 클릭 이벤트를 바인딩한 뒤, Generate 후 Quick View를 통해 실행하여 확인하면, 알림창과 Output에 직접 작성한 내용이 기록되는 것을 볼 수 있다.



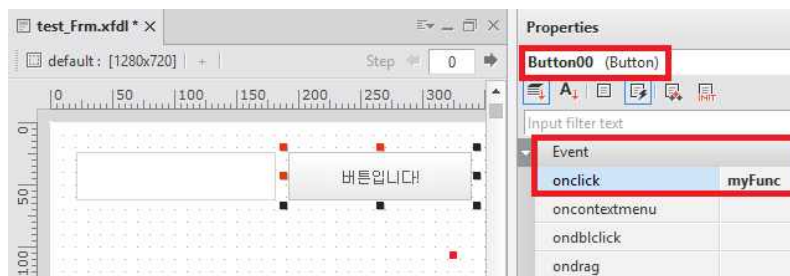
그럼 조금 더 나아가 Edit 이라는 텍스트 필드를 하나 만들고, 그 안의 값을 출력하고 변경하는 이벤트를 만들어 보자.

먼저, 툴 바에서 Edit과 Button Component를 선택하고 각각 Form Design에 추가한다.

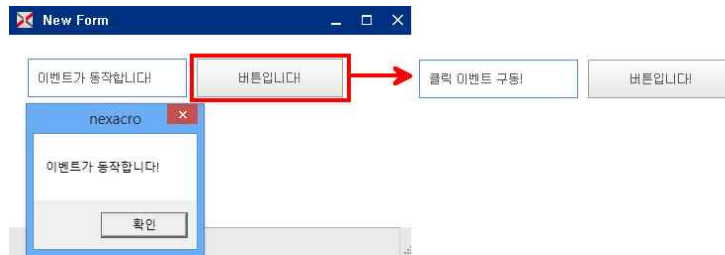


다음으로 Form Script 탭으로 넘어가서 myFunc라는 사용자 정의함수를 다음과 같이 만들고 생성한 버튼에 해당 이벤트를 바인딩한다.

```
this.myFunc = function(obj, e){  
    // Edit00의 값을 받아 저장할 변수 생성  
    // Edit00은 Form 내부에 있으므로 this 범위 지정자로 접근이 가능하다.  
    var str = this.Edit00.value;  
  
    // Edit00의 값을 알림창으로 띄운다.  
    alert(str);  
  
    // 현재 Edit00의 값을 변경한다.  
    // Setter는 제공되는 Property 앞에 'set_'를 붙여 사용한다.  
    this.Edit00.set_value("클릭 이벤트 구동!");  
}
```



작성이 완료되면 Generate 후 Quick View를 통해 확인이 가능하다.

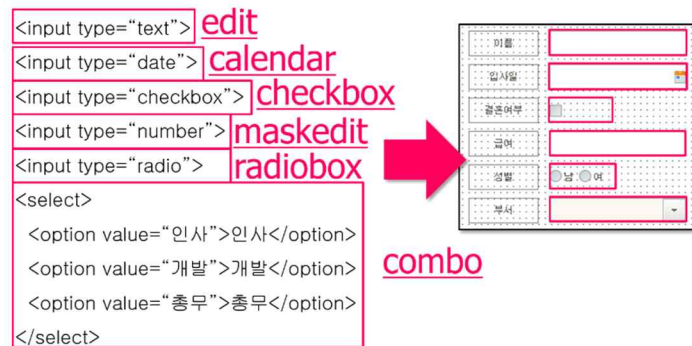


'F1'를 통해 매뉴얼을 조회하여 Edit 항목을 검색하면 해당 컴포넌트에 대해 자세하게 알 수 있으며, 내부 Property 중 value 를 찾아 들어가보면 Getter와 Setter의 사용법에 대해 알아 볼 수 있다.

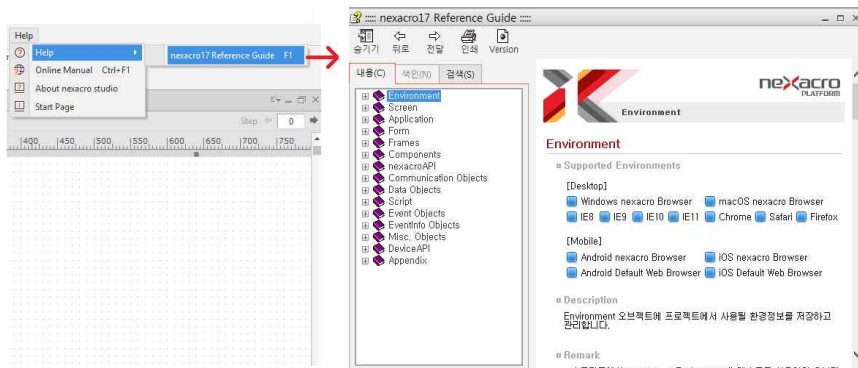


3 컴포넌트의 종류

컴포넌트는 Design 폼에서 WISIWYG 기반 작업환경 제공하는 도구로써, 해당 컴포넌트를 만든 뒤, generate 하게 되면, 웹표준에 근거하여 HTML태그로 변환된다.

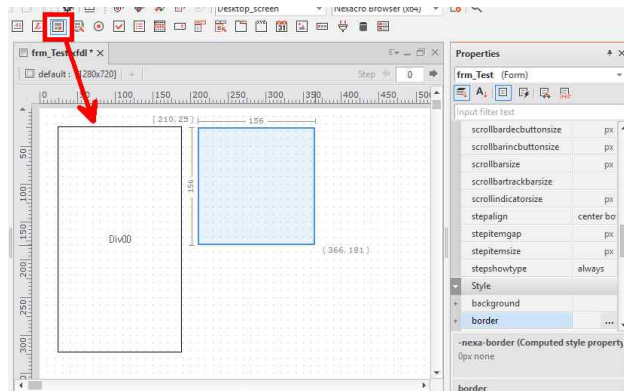


넥사크로 스튜디오에서는 Help > (단축키 F1) 를 통해 각 컴포넌트 요소들에 대한 풍부한 API를 제공하고 있으며, 이후 소개하는 컴포넌트들에 대한 자세한 사용법을 알아 볼 수 있다.



3.1 DIV

Div 는 다른 컴포넌트들을 한데 묶어 그룹화 할 수 있는 컴포넌트로, Div를 생성 후, 원하는 컴포넌트를 해당 Div 내부에 완전히 포함 되도록 생성하면 된다.

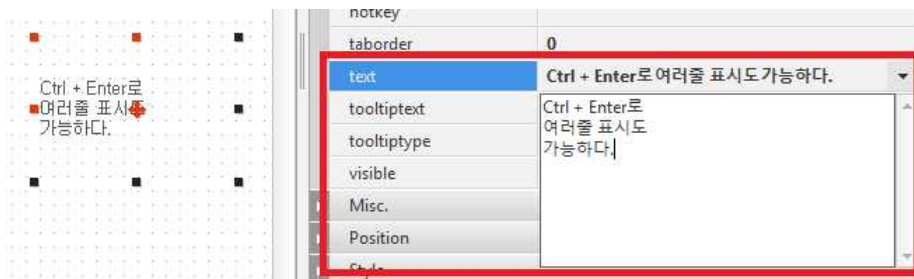


Div는 내부에서 xml 처리되며, 해당 부분은 Form Source 탭에서 확인할 수 있다.



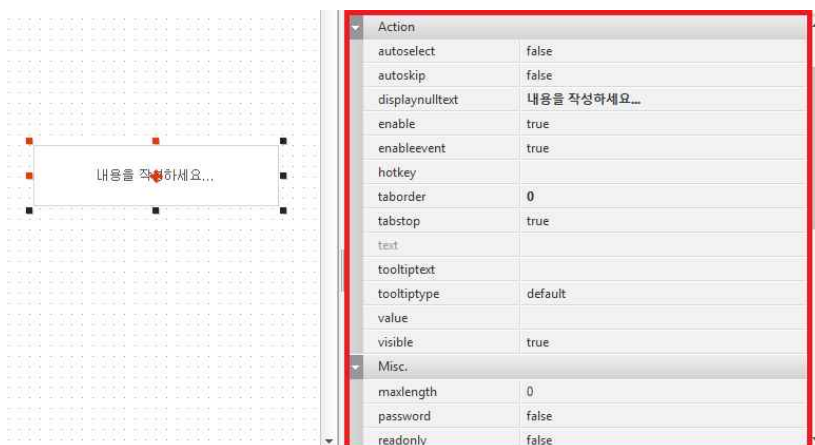
3.2 Static

화면에 일정한 문자열을 출력하기 위해 사용한다. 출력할 문자열은 후에 설명할 Dataset과 바인드하여 수시로 변하게 하거나, 고정적으로 출력하게 할 수도 있다.



3.3 Edit

<input type="text">와 거의 동일하게 사용하는 컴포넌트로, 사용자로부터 문자열을 입력 받거나 암호문자열을 입력 받을 때 사용한다.

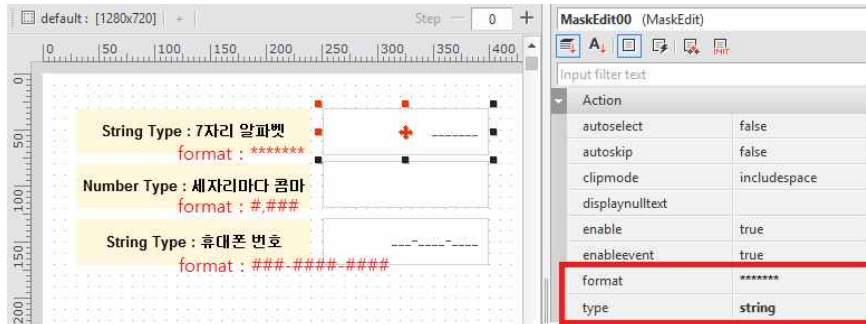


주로 사용되는 속성은 다음과 같다.

속성 명	설명
autoskip	입력되는 문자열의 길이가 maxlength속성에 지정된 길이만큼 입력 되었을 때 자동으로 다음 컴포넌트로 이동
readonly	읽기전용(비활성화) 유/무를 설정
tabborder	사용자가 tab키를 눌렀을 때 이동하는 순서 지정
displaynulltext	input 태그의 placeholder 속성과 같은 역할
maxlength	입력할 수 있는 문자열의 최대 길이를 설정
password	입력/출력되는 문자를 '*'로 숨길지 여부를 설정

3.4 MaskEdit

정해진 규격에 따라 입력/출력이 필요한 경우 사용되는 컴포넌트로, 주민등록번호, 전화번호 등 일정한 포맷을 지정하여 표현이 가능하다.



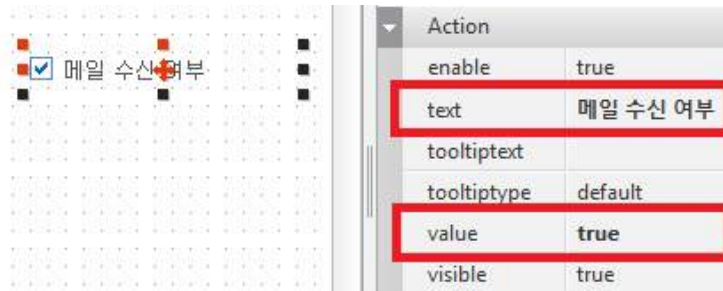
Properties – Action의 type 속성에 따라, mask 속성으로 입력 받을 문자의 형식을 지정할 수 있다.

Type	Mask value
Number	# : 모든 10진수 숫자 (0-9) 9 : 모든 10진수 숫자 (0-9) 0 : 모든 10진수 숫자 (0-9)(값이 없을 경우 0로 설정) . : 소숫점 자릿수 , : 콤마 표시 여부 (Text에서만 나타남)
String	@ : 모든 Ascii 문자(한글 등 다국어 불가) # : 모든 10진수 숫자(0-9) * : 모든 알파벳 문자(a-zA-Z) 9 : 모든 알파벳, 숫자(a-zA-Z0-9) A : 대문자 알파벳(A-Z) a : 소문자 알파벳(a-z) Z : 대문자 알파벳, 숫자(A-Z0-9) z : 소문자 알파벳, 숫자(a-z0-9)

기타 자세한 형식은 F1을 눌러 매뉴얼의 MaskEdit 파트를 확인하기 바란다.

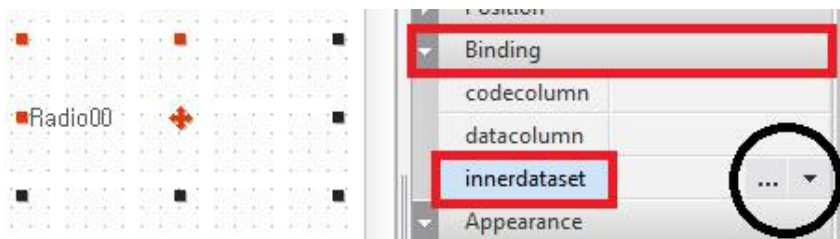
3.5 CheckBox

<input> 태그의 체크박스와 동일한 컴포넌트로, true/false value에 따라 기본값을 지정할 수 있다.

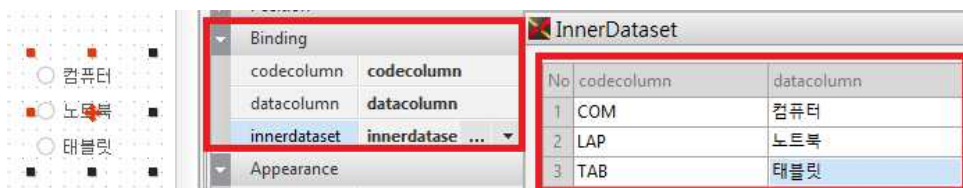


3.6 Radio

<input> 태그의 radio 타입과 동일하나 각각의 요소를 name 속성으로 그룹핑 짓지 않고, innerDataset 이라는 내부 Dataset을 바인딩하여 화면에 보여질 부분과 실제 값 부분을 매칭시킨다.



먼저 Radio 컴포넌트를 생성한 뒤, 우측 Properties-Binding 탭에서 innerdataset의 '...'을 클릭한다. 그럼 아래의 그림과 같이 내부 Dataset을 만들 수 있는데, codecolumn에는 원하는 값을, datacolumn에는 화면에 보여질 값을 적으면 된다. `



innerDataset의 내용을 화면 상에서 추가하고 싶다면, 메소드에 다음과 같은 script를 작성하여 구현하면 된다.

```
var innerDS = this.Radio00.getInnerDataset();
innerDS.insertRow(0);
innerDS.setColumn(0,"codecolumn",'code값');
innerDS.setColumn(0,"datacolumn","data값");
```

3.7 Combo

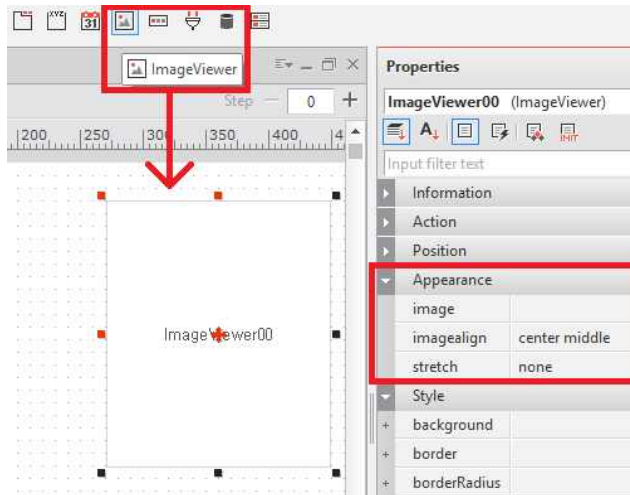
HTML 에서의 <select> 태그와 <option> 태그로 사용되는 콤보박스와 유사하며, Radio 컴포넌트와 동일하게 innerdataset을 사용하여 구현한다.



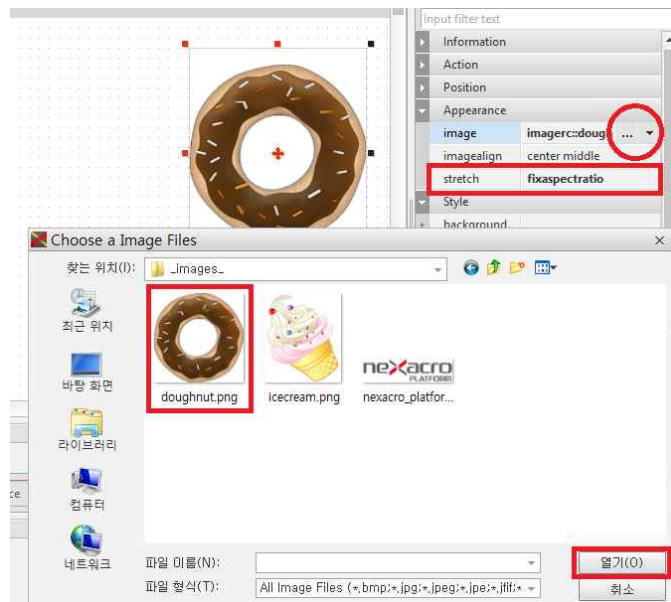
3.8 ImageViewer

ImageViewer 컴포넌트는 이미지 콘텐츠를 화면에 보이도록 처리한다. 웹에서 이미지를 처리하는 것은 이미지를 보여주고 정렬하는 수준이라면 ImageViewer 컴포넌트는 해당 영역에 텍스트를 추가하고 편집할 수 있다.

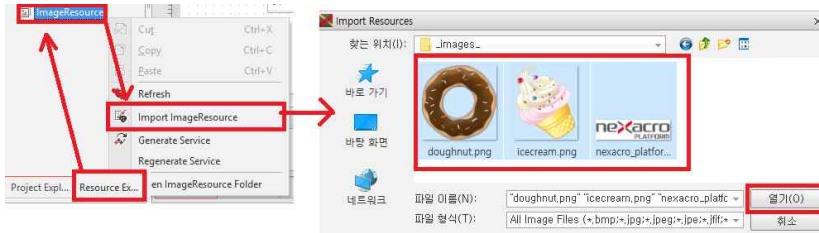
먼저 ImageViewer를 선택하여 Form 화면에 드래그 앤 드랍을 통해 컴포넌트를 하나 생성한다. 그럼 다음과 같이 빈 화면 하나가 생성된 것을 확인할 수 있다.



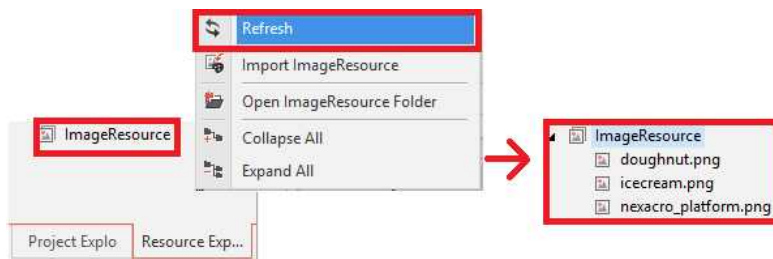
다음으로 properties 창의 Appearance > image 항목을 클릭하여 이미지를 하나 선택한 후 화면에 출력되는 것을 확인한다. 이미지가 주어진 박스 크기에 맞지 않을 경우 stretch 항목의 'fit', 'fixaspectratio' 값을 통해 비율을 조절할 수 있다.



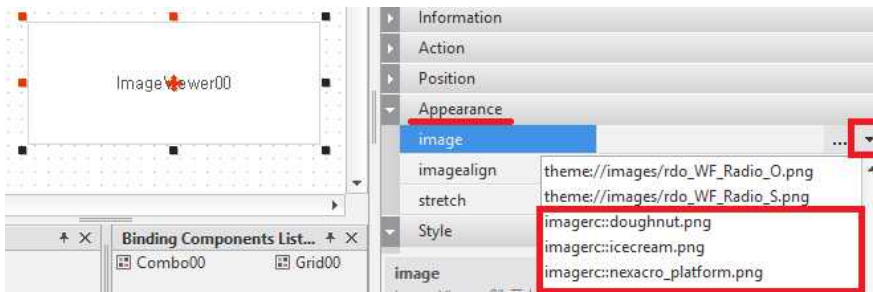
만약 여러 개의 이미지 경로를 저장하여 사용하고자 할 경우 Project Explorer 에서 Resource Explorer 탭을 통해 이미지를 등록 해놓고 사용할 수도 있다.



먼저 Resource Explorer 에서 ImageResource 항목을 선택하고 컨텍스트 메뉴에서 [Import ImageResource] 항목을 선택한다. 파일 대화 상자를 열어 이미지 파일을 선택하면 ImageResource 를 추가할 수 있다.



추가된 이미지파일은 ImageResource 를 우클릭하여 Refresh 를 하면 추가된 것을 확인할 수 있다.



이렇게 추가된 이미지 경로는 ImageViewer 의 properties > Appearance > image 항목을 통해 손쉽게 사용할 수 있으며 이미지를 추가한 후에는 반드시 generate 하여 해당 내용을 js 코드로 변환해야 실제 프로젝트에 이미지가 반영된다.

이미지를 동적으로 변경하고자 할 경우 ImageViewer 컴포넌트의 image 속성을 통해 변경이 가능하다. 다음은 이러한 동적 접근에 대한 코드를 정리한 표이다.

이미지 동적 접근	
절대경로 이미지	ImageViewer.set_image('url("file://C:/imageID")');
상대경로 이미지	ImageViewer.set_image('url("./imageID")');
테마 이미지	ImageViewer.set_image('url("theme://images/imgFile.jpg")');
TypeDefinition Prefix 이미지	ImageViewer.set_image('url("Images::image.jpg")');
외부 이미지	ImageViewer.set_image('url("http://abc.com/image.jpg")');
Base64 인코딩 이미지	ImageViewer.set_image("data:image/png;base64,iVBORw0~....");

그럼 이러한 속성을 활용하여 버튼을 클릭 시 이미지가 변경되는 화면 이벤트를 만들어보자. 먼저 화면에 Button 컴포넌트와 ImageViewer 컴포넌트를 생성하여 다음과 같이 배치한다.



배치가 끝났다면 다음과 같은 스크립트를 Script 탭에서 추가해보자.

```

/// 현재 이미지를 도넛으로 변경하기
this.Button00_onclick =
    function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
    {
        this.ImageViewer00.set_image("imagerc::doughnut.png");
        this.ImageViewer00.set_stretch("fit");
    };

```



```

/// 현재 이미지를 아이스크림으로 변경하기
this.Button01_onclick =
    function(obj:nexacro.Button,e:nexacro.ClickEventInfo)
    {
        this.ImageView00.set_image("imagerc::icecream.png");
        this.ImageView00.set_stretch("fixaspectratio");
    };

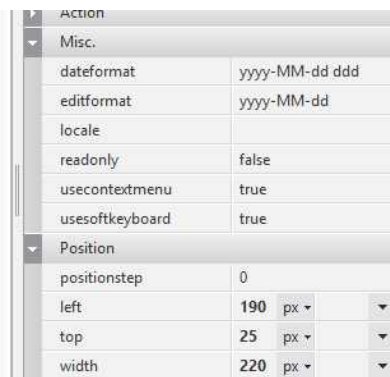
```



위의 이미지처럼 이벤트 바인딩과 각 컴포넌트의 속성을 활용하면 동적인 화면을 쉽게 구현해낼 수 있다.

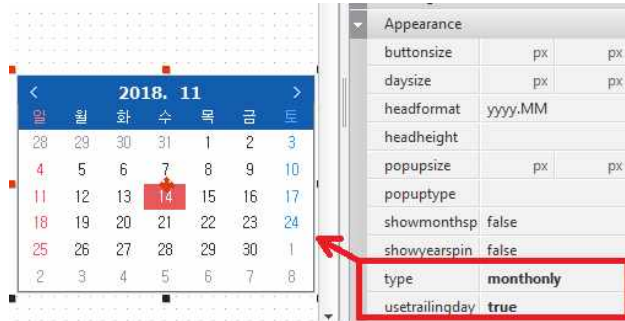
3.9 Calendar

HTML의 <input> 태그의 date, datetime-local 타입과 유사한 컴포넌트로, 날짜와 시간을 입력 받아 특정 포맷으로 표현한다.



Properties > Misc에서 포맷, 지역 설정을 정할 수 있으며 'dateformat'으로 편집 중이 아닐 때 보여지는 날짜 형식을, 'editformat'으로 편집 중일 때 보여지는 날짜 형식을 지정할 수 있다. 또한 Properties > Appearance 의 type, usetrailingday 속성 변경을 통해 화면

에 표현할 양식도 변경이 가능하다.



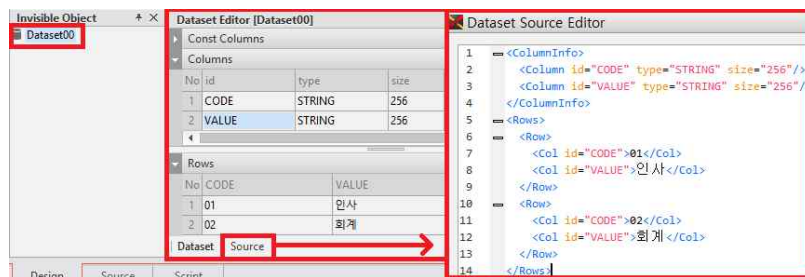
기타 자세한 사항은 매뉴얼(F1)을 참고하기 바란다.

3.10 Dataset

Dataset 컴포넌트는 넥사크로플랫폼 애플리케이션 내에서 데이터를 관리하거나 서버와 통신할 때 주고받는 데이터 형식으로, 사용자가 입력하거나 서버에서 가져온 데이터를 보관하고 데이터의 변경(추가/수정/삭제) 상태를 관리할 수 있다.

데이터셋은 2차원 테이블 형태로 데이터를 관리하며, 컬럼(Column) 구조와 로우(Row, Record) 단위로 데이터를 다룰 수 있다. 하나의 폼에는 여러 개의 Dataset을 가질 수 있으며 GlobalVariables로 Dataset을 만들어 여러 폼에서 공통으로 사용할 수도 있다.

Dataset은 화면에 보이는 컴포넌트가 아니라 애플리케이션 내부에서 데이터 관리를 위해 사용되는 컴포넌트이기 때문에 Invisible Objects 항목에 나타나며 해당 컴포넌트를 드래그하여 다른 컴포넌트에 값에 대한 바인딩처리를 할 수 있다.

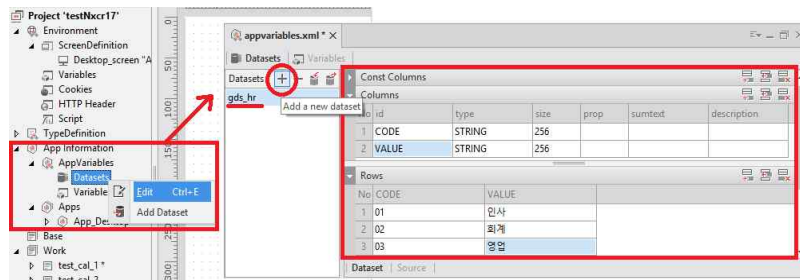


Dataset은 다음 3곳에서 정의할 수 있으며, Source 탭을 통해 XML형태로 변환되는 것을 확인할 수 있다.

선언 범위	설 명
전역 Level	해당 Application에서 사용하는 모든 Form에서 접근 가능
Form Level	해당 Form에서만 접근이 가능
innerDataset	해당 컴포넌트에서만 접근 가능


1. 전역 레벨에서 dataset 선언

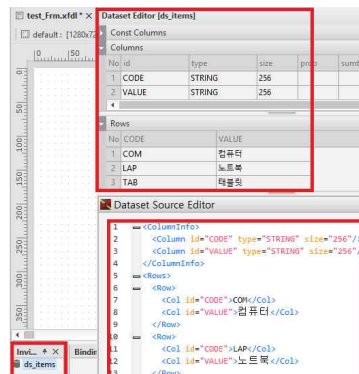
AppVariables > Datasets 에서 선언하며, 해당 App의 모든 Form 에서 접근할 수 있다.



** 전역 변수로 Dataset을 선언할 경우 id 변경 시 바로 적용되지 않는 경우가 있으나, 해당 dataset을 저장하고 탭을 닫은 뒤, 다시 열면 변경사항이 적용된다.

2. Form 레벨에서의 Dataset 선언

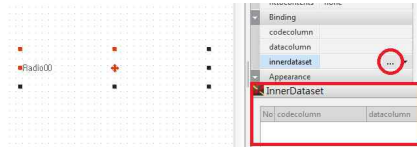
툴 바에서 Dataset  을 선택한 상태에서 Form Design 영역에서 다시 한번 클릭하면 Dataset이 추가된다.



** Source 탭을 보면 Dataset이 XML로 변환되는 것을 확인할 수 있다.

3. Component 레벨에서의 innerdataset으로 Component에 바인딩 :

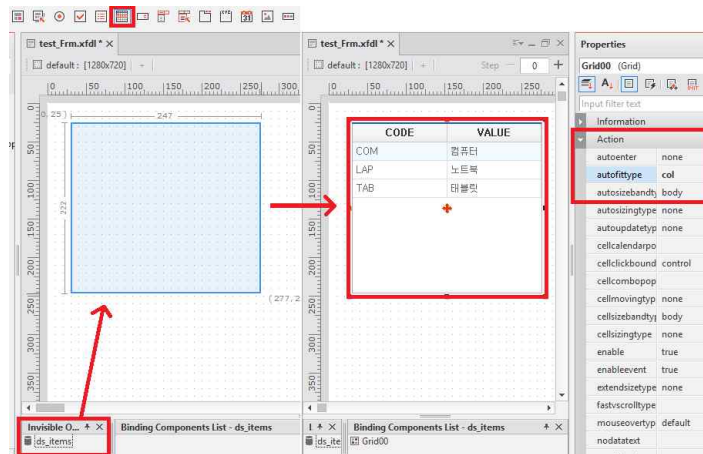
(radio/combo컴포넌트 등 참조)



** innerdataset을 사용하는 컴포넌트는 Calendar, Combo, ListBox, Menu, PopupMenu, Radio가 있다.

3.11 Grid

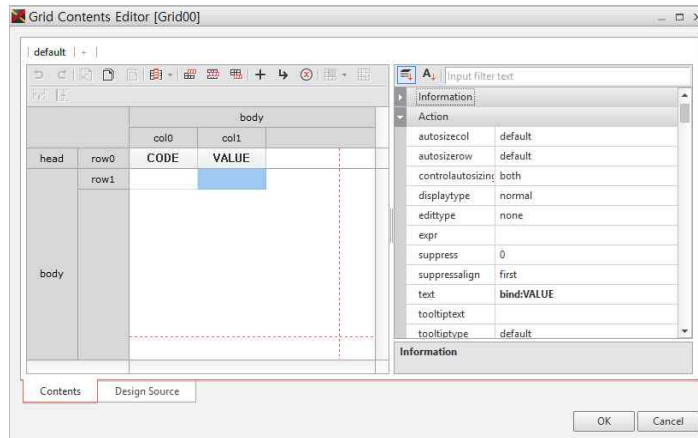
Dataset의 내용을 격자 모양으로 표현하는 컴포넌트로, Dataset에 변경 사항이 생기면 실시간으로 Grid에 반영되고, 반대로 Grid에 변경 사항이 생기면 실시간으로 Dataset에 반영된다.



** Grid 생성 후 Properties – Action의 autofittyp에서 col을 선택하면
현재 grid의 크기에 맞게 내부 컬럼 사이즈를 수정한다.

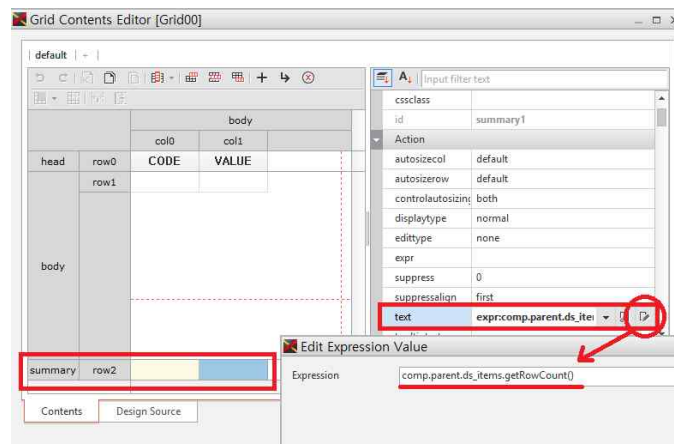
- Grid Contents Editor

현재 추가한 컴포넌트를 더블 클릭하게 되면, grid의 세부설정을 할 수 있는 Grid Contents Editor가 팝업된다.



Formats 탭은 head, body 와 summary 로 구성된다. head 는 Databases 의 컬럼 명에, body는 row의 실제 값 표시 란에 해당되며, summary는 모든 row에 대한 요약 표시할 때 쓰인다.

Grid에서는 값을 표현할 때 넥사크로플랫폼에서 제공하는 표현 식(expression)을 사용할 수 있는데, grid도 하나의 컴포넌트이므로, form 이나 application의 dataset에 접근할 때 해당 접근에 맞는 scope를 함께 작성해야 한다.



```
/* 현재 Row의 총 개수를 출력해보자. */
//전역 Dataset일 경우
nexacro.getApplication().gds_items.getRowCount();
//Form 영역 Dataset일 경우
comp.parent.ds_items.getRowCount();
```