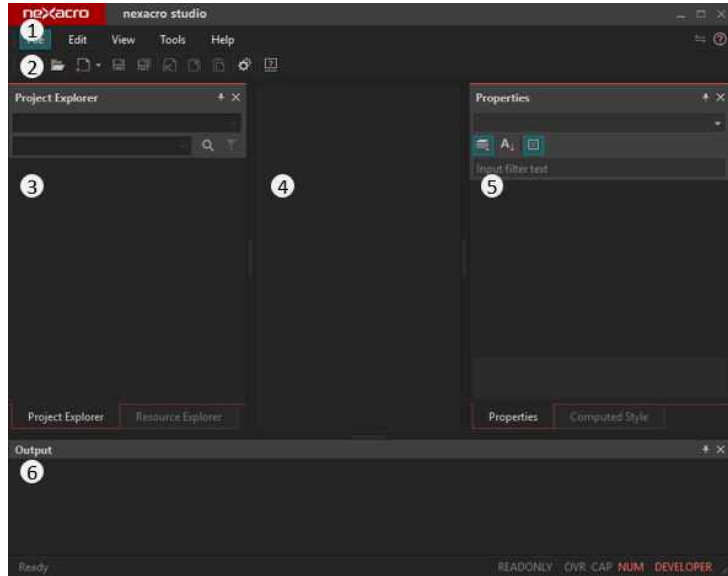


## **Chapter 3 넥사크로 프로젝트 시작하기**

# 1 nexacro studio Interface

프로젝트 만들기에 앞서 Nexacro studio 의 기본구성은 다음과 같이 이루어져 있다.

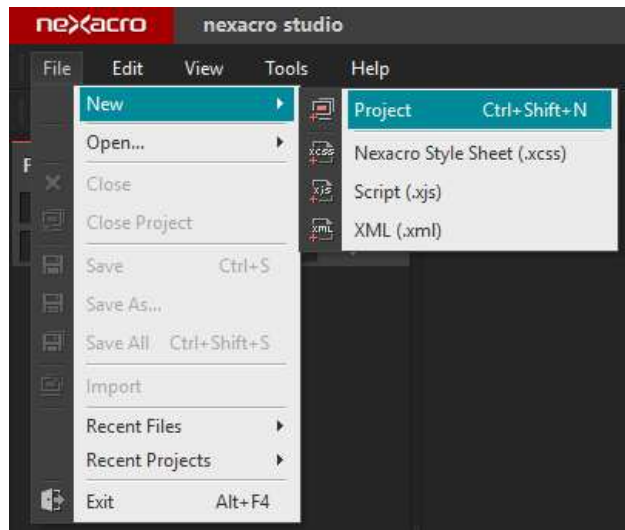


구성요소	설명
① 메뉴바	Nexacro studio의 기본 메뉴 구성
② 툴바	프로젝트에 자주 사용되는 tool 모음
③ 프로젝트 탐색기	프로젝트를 탐색기 형식으로 보여준다
④ Form Design	현재 작업 중인 영역에 대한 화면
⑤ 속성 창	폼, 컴포넌트, Dataset 컴포넌트의 속성을 보여주고 편집하는 공간
⑥ Output	오류 메시지나 Generate 메시지 등을 볼 수 있는 공간

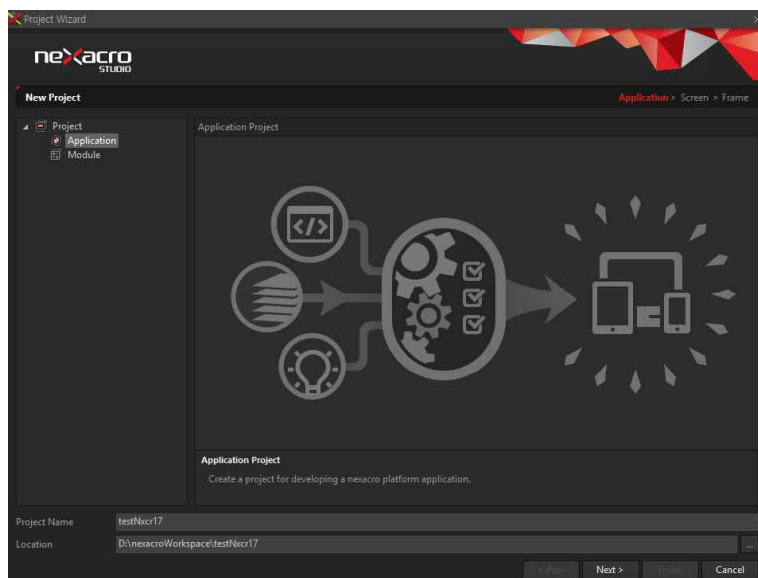
Form Design 창이 활성화되면 세 개의 탭이 생기는데, Design 탭은 화면에 표시될 위지윅(WYSIWYG) 작업창을 의미하며, Source 탭은 해당 컴포넌트들이 실제 XML 파일로 생성되는 것을 보여준다. 또한 Script 탭은 내부에서 사용될 메소드나 변수를 코딩할 때 사용한다.

## 2 넥사크로플랫폼 프로젝트 만들기

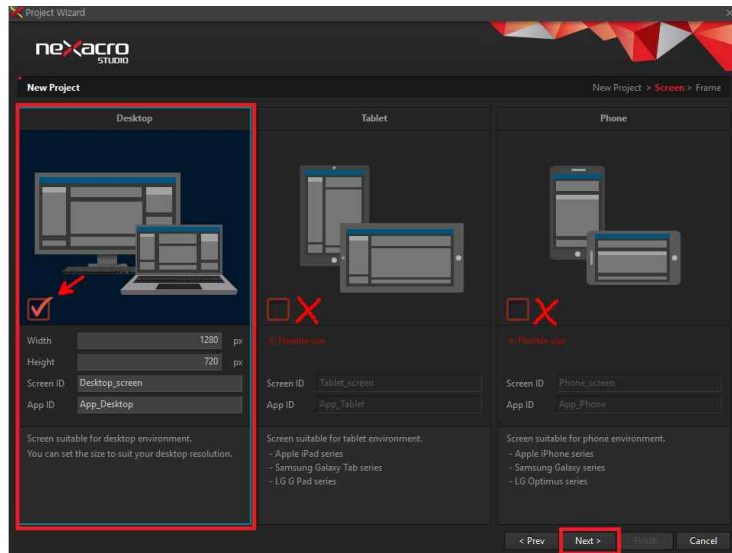
넥사크로플랫폼 애플리케이션을 만들기 위해서는 프로젝트를 먼저 만들어야 한다.  
먼저 메뉴 바에서 File > New > Project를 통해 새 프로젝트를 작성한다.



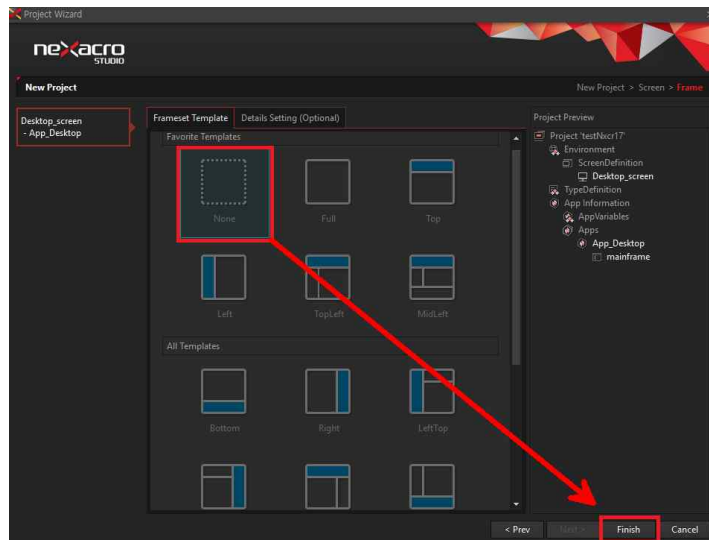
프로젝트는 새로 만들거나(Application) 기존의 프로젝트에 추가하는 구성(Module)으로 제작할 수 있으며, 본 챕터에서는 새로운 프로젝트로 생성하여 진행하도록 한다.  
프로젝트의 이름(Name)을 'testNxcr17'로 짓고 위치(Location) 설정을 마친 후 Next> 를 클릭하자.



그럼 화면과 같이 반응형으로 화면을 다양한 폼으로 제작할 것인지 묻는 창이 뜨는데 이번 예제에서는 기본 설정인 Desktop으로 진행한다. 확인이 완료되면 Next>를 클릭하자.



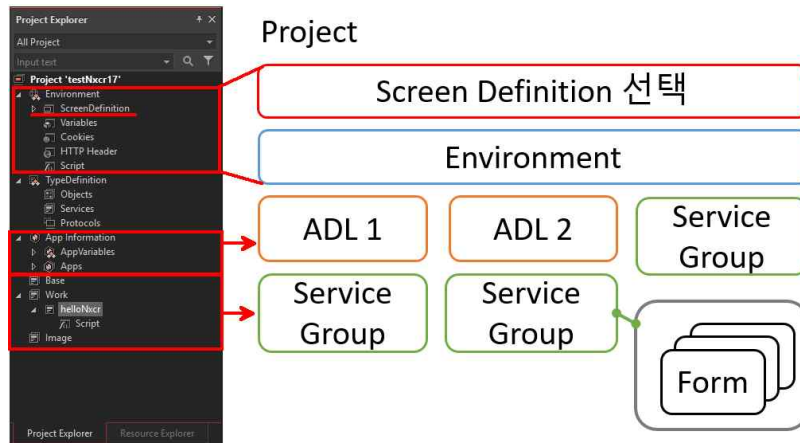
화면 사이즈까지 지정했다면 마지막으로 화면의 구성을 어떻게 할 것인지 묻는 frameset Templates 창을 볼 수 있다. 이것 역시 기본 설정인 'None'을 선택하고 'Finish'를 클릭하도록 하자.



이제 실습을 위한 기본 프로젝트 생성이 완료되었다.  
다음 목차로 이동하여 넥사크로 플랫폼을 사용해보자.

### 3 넥사크로플랫폼 애플리케이션 만들기

넥사크로플랫폼은 다음과 같은 구조를 지니는데, 하나의 애플리케이션 내에 직접 Form이 위치하는 것이 아니라 Service Group이라는 폴더 이내에 각각 기능을 담당할 Form이 위치하게 된다.



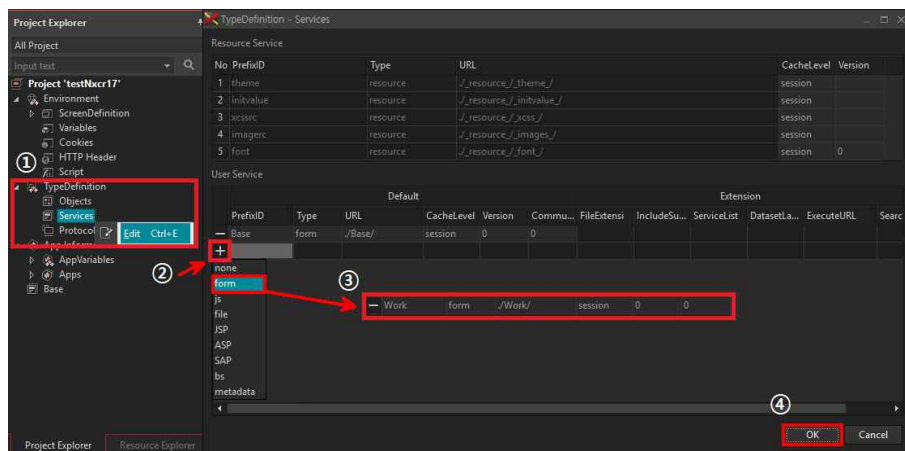
구성 요소		
<b>Environment</b>	실행 환경 정보를 가지고 있는 설정 파일로 애플리케이션에서 사용하는 Screen, Variables, Cookies 등을 설정 여러 개의 애플리케이션으로 구성 시 공유하여 사용 가능	
<b>TypeDefinition</b>	넥사크로플랫폼에서 사용되는 오브젝트와 생성한 물리적인 파일이 존재하는 디렉토리의 경로 등을 관리	
<b>App Information</b>	애플리케이션과 화면에서 공통으로 사용할 변수, 이차원 형태의 데이터를 지정	
<b>생성 산출물</b>	_resource_	애플리케이션에서 사용하는 Theme, UserFont 등의 정보를 관리하는 Resource 폴더
	Base(Service)	애플리케이션에서 생성된 화면, script, 라이브러리, 이미지 등의 파일들이 저장되는 폴더
	Form, Popup	TypeDefinition > Service 항목에서 추가한 서비스
	App_Desktop.xadl	Apps > 'App_Desktop' 항목의 화면 설정 정보
	*.xprj	개발 시에만 필요한 프로젝트 작업 파일

기본 프로젝트 생성시에 Base라는 서비스 그룹이 생성되나, 본 예제에서는 해당 서비스 그룹을 사용하지 않고, 직접 'Work'라는 서비스 그룹을 만들어 그 안에 Form까지 작성해 보는 것을 연습해 보도록 하겠다.

### 3.1 Service Group 만들기

서비스 그룹은 프로젝트의 서브디렉토리로, 폼/파일(이미지 등)/경로/css/js등 여러 자원들을 그 기능에 맞게 분류하여 저장한다.

서비스 그룹을 만들기 위해 좌측의 프로젝트 탐색기 상단의 TypeDefinition 을 더블 클릭하여 열고 Service 탭을 우클릭한 뒤 Edit 메뉴를 클릭한다.



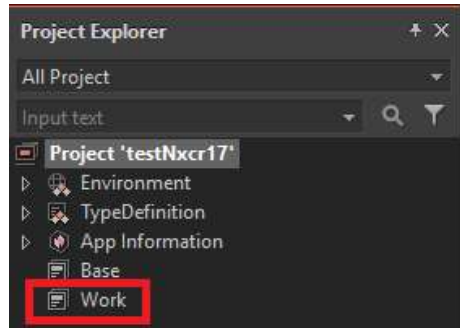
그럼 Base 라는 기본으로 제공하는 서비스 그룹이 보이는데, 하단의 '+'를 클릭하여 Type을 form으로 선택하고, 'PrefixID'를 'Work'라고 작성하여 실습을 위한 서비스 그룹을 하나 생성하자.

User Service					
Default					
PrefixID	Type	URL	CacheLevel	Version	Commu...
Base	form	./Base/	session	0	0
Work	form	./Work/	session	0	0
+					

\*\* 작성을 완료하였다면 상단의 그림처럼 등록되어야 한다.

Service 를 접근할 URL 은 자동으로 변경되나 차후 폼 화면 접근에 사용할 것이기 때문에 './Work/'로 잘 변경되었는지 반드시 확인하고 하단의 'OK'버튼을 클릭하면 된다.

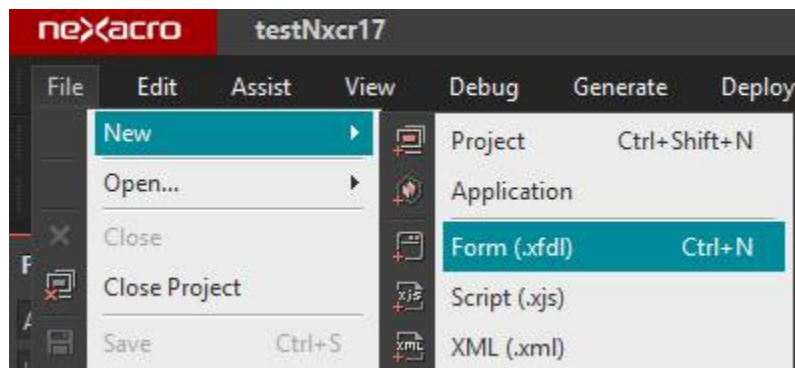
OK를 누르면 Project Explorer 항목에 추가된 것을 확인 할 수 있다.



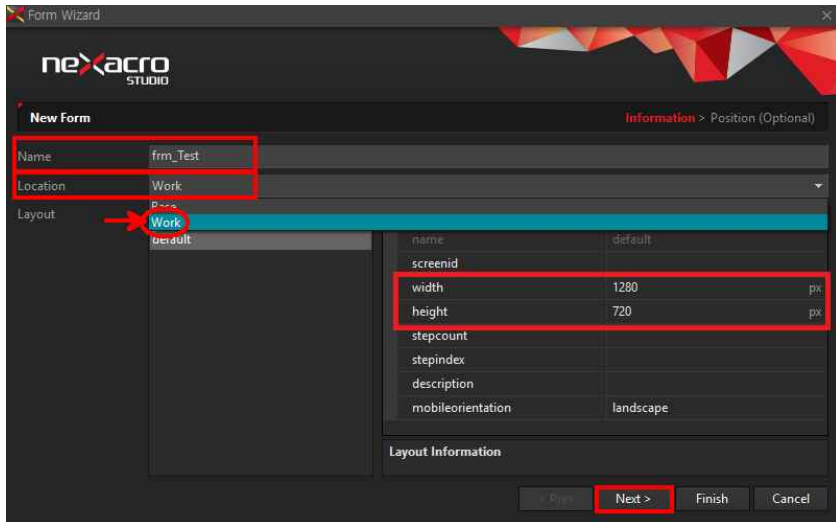
## 3.2 Form 만들기

넥사크로플랫폼 애플리케이션은 폼(Form)을 기반으로 동작한다. 즉, 프로젝트가 애플리케이션을 위한 무대를 만든 것이라면 무대를 꾸미고 동선을 배치하고 효과를 추가하는 작업은 폼에서 담당한다는 것이다. 그럼 실제 화면에서 기능을 담당할 폼을 작성해 보자.

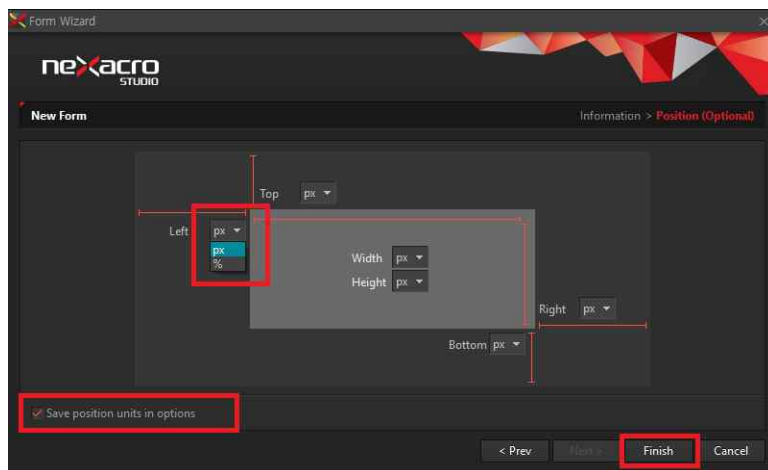
폼은 메뉴 바에서 File > New > Form을 선택하여 만든다.



팝업된 Form Wizard에서 폼의 이름을 기입하고, 해당 form의 이름을 'frm\_Test'로 짓고 해당 폼이 위치할 서비스 그룹을 Work로 정한 뒤, Next를 눌러 화면에 표시할 사이즈를 정한다.



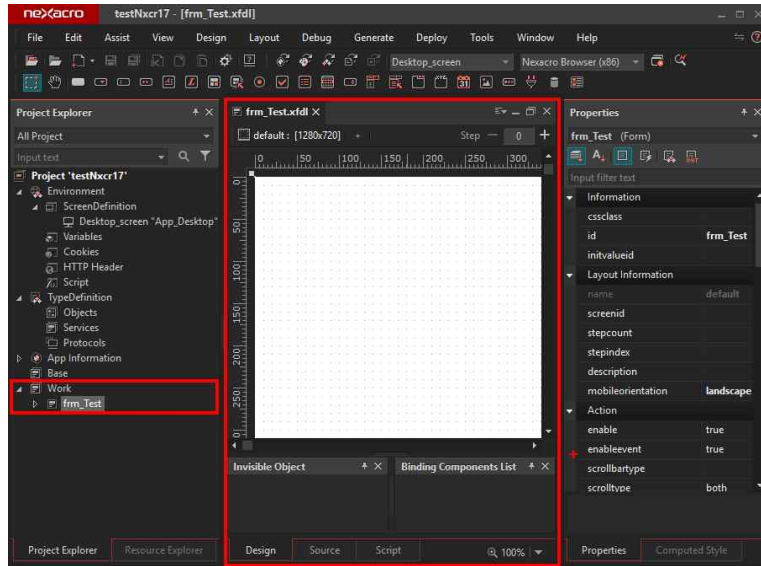
그 뒤 Next를 눌러 Form에서 사용할 위치의 지정 값을 고른다.  
클라이언트 입장에서 사용하는 각 기기마다 화면의 크기는 다를 수 있으므로 이를 절대 값으로 잡아 주기 위해 값 지정 방식을 px 단위로 정한다.




하단의 Save position in option을 체크하면, 폼 작성 이후 속성 창의 Properties-Position에서 해당 값을 직접 제어할 수 있다.

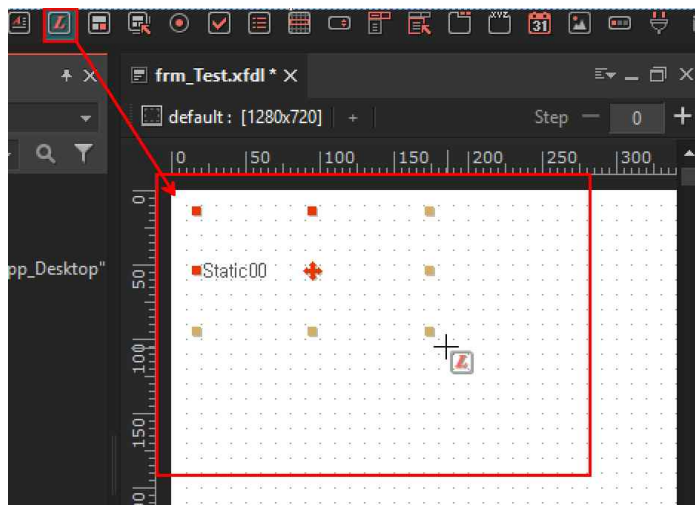


그럼 그림과 같이 Work 하위 요소로 우리가 만든 frm\_Test 가 위치하는 것을 확인할 수 있으며, 가운데의 Form Design 탭에서 우리가 작업할 영역이 활성화된 것을 알 수 있다



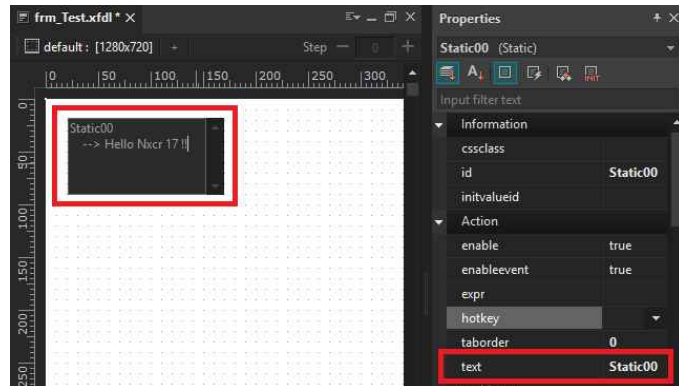
### 3.3 기본 컴포넌트 만들기

그럼 이제 우리가 만든 form에 간단한 메시지를 작성하여 화면에 표시해보자. 먼저, 상단의 컴포넌트 중  모양의 박스 아이콘을 선택하여 화면에 글자를 표시하는 Static 컴포넌트를 클릭한 뒤 frm\_Test 화면에서 드래그하여 생성한다.



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

생성한 Static 컴포넌트를 클릭한 뒤 F2키를 누르면 글자 편집이 가능하며, Ctrl + Enter 를 통해 줄바꿈도 사용할 수 있다. 또한 우측 Properties -> Action, 탭의 text 에서도 글의 수정이 가능하다. 그럼 위의 단축키를 활용하여 'Static00' 을 'Hello Nxcr 17'로 바꿔보자.

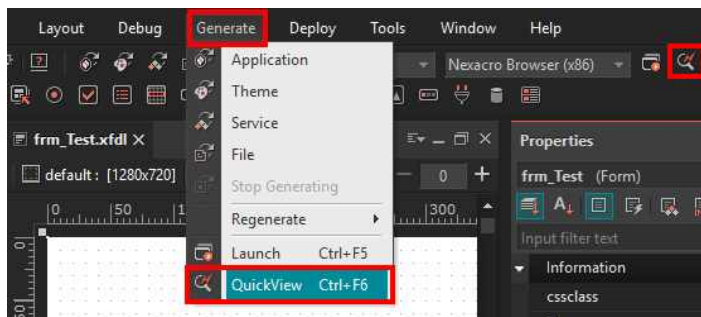


작성을 완료하였다면 다음 장으로 넘어가 화면을 직접 실행하여 확인하면 되겠다.

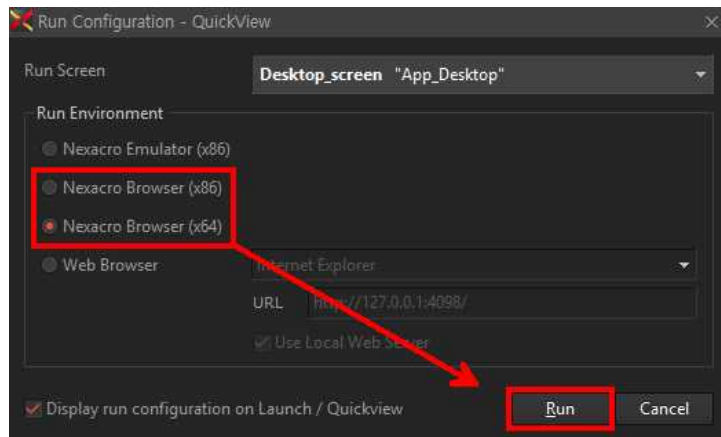
### 3.4 넥사크로 플랫폼 실행하기 (Generate)

넥사크로 스튜디오에서 작성된 코드는 바로 실행하지 않고 자바스크립트 코드로 변환(Generate)하는 과정이 필요한데, Form 을 생성하거나 수정 후 저장할 때 자동으로 처리된다. 또한 Generate > Application 을 통해 직접 Generate를 처리할 수도 있다.

자신이 만들거나 수정한 애플리케이션에서 하나의 Form을 바로 확인하고 싶을 때에는 Quick View를 사용한다.

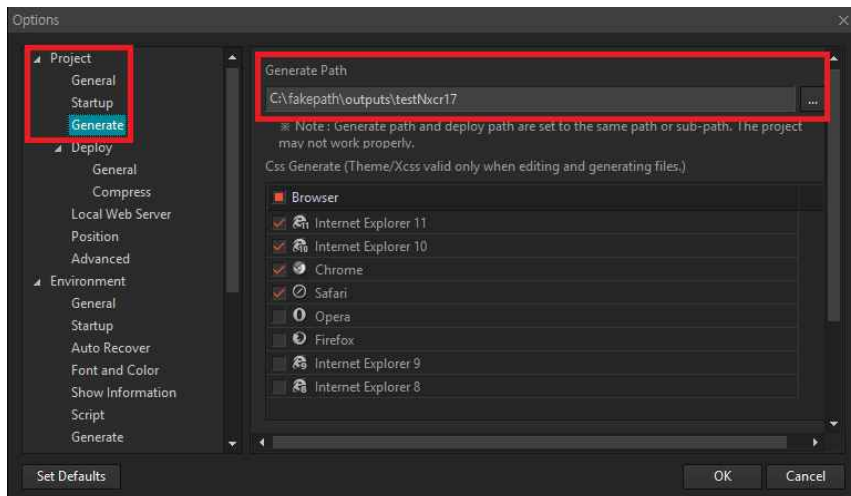


상단의 메뉴에서 Generate > Quick View 를 클릭하거나 툴 바에 있는 돋보기 모양 아이콘을 클릭하면 현재 작업 중인 Form 의 화면을 바로 볼 수 있는데, 실행 시에는 다음과 같은 확인 메시지가 발생한다.



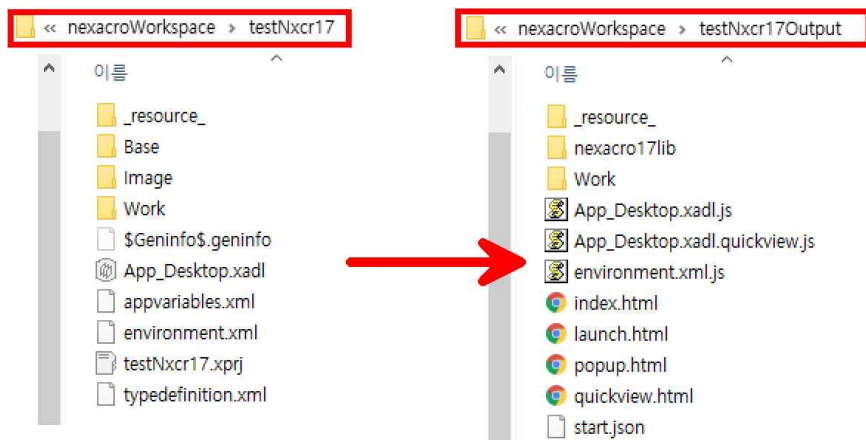
해당 설정 창에서 자신의 운영체제와 일치하는 넥사크로 브라우저나 평소에 사용하던 웹 브라우저를 통해 확인할 수 있다.

Generate 를 통해 변환된 코드는 넥사크로 스튜디오의 Generate Path 안에 생성되며, 해당 설정은 상단 메뉴 바의 Tools > Options > Project > Generate > Generate Path 에서 확인하고 수정할 수 있다.

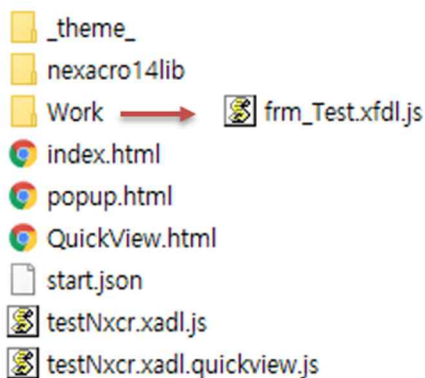


넥사크로 스튜디오에서 기본 설정으로 지정하는 Generate Path  
C:\Users\[사용자명]\Documents\nexacro\outputs\[프로젝트명] 이다.

현재 프로젝트는 최초 생성 경로에 있지만, javascript 로 변환되어 output 된 프로젝트는 Generate Path 로 등록된 위치에 생성된다.



이렇게 넥사크로 스튜디오에서 변환된 파일은 다음과 같이 js 파일로 변환되며, 실제 파일의 내부 구조는 다음과 같이 JavaScript 로 되어 있는 것을 확인할 수 있다.



```

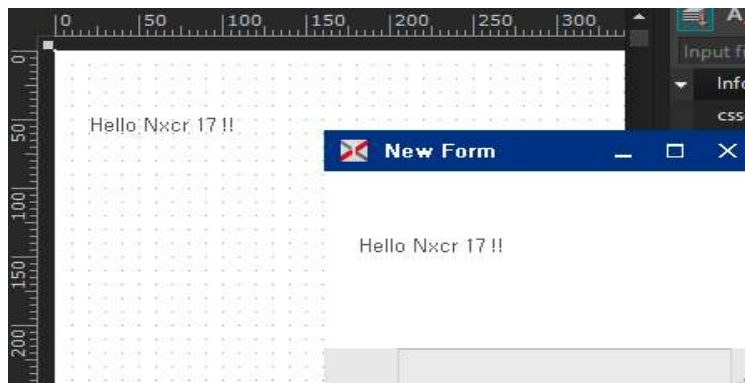
1 (function()
2 {
3     return function()
4     {
5         if (!this._is_form)
6             return;
7
8         var obj = null;
9
10        this.on_create = function()
11        {
12            this.set_name("frm_Test");
13            this.set_titletext("New Form");
14            if (Form == this.constructor)
15            {
16                this._setFormPosition(1280,720);
17            }
18        }
19    }
20 }

```

또한 Generate 과정을 하단의 Output 창에서 다음과 같이 확인할 수 있다.

```
Output
1> Successfully generated file : "D:\nexacroWorkspace\testNxcr17Output\start.json"
1> Successfully generated file : "D:\nexacroWorkspace\testNxcr17Output\index.html"
1> Successfully generated file : "D:\nexacroWorkspace\testNxcr17Output\launch.html"
1> Successfully generated file : "D:\nexacroWorkspace\testNxcr17Output\quickview.html"
1> Successfully generated file : "D:\nexacroWorkspace\testNxcr17Output\popup.html"
1> ===== Finish generating ( 2.91 sec ) : Success 22, Fail 0, Copy 0, Skip 0 =====
1> ===== Start generating =====
1> This file is already generated : "D:\nexacroWorkspace\testNxcr17Output\Work\frm_Test.xfdl.js"
1> ===== Finish generating ( 0.06 sec ) : Success 0, Fail 0, Copy 0, Skip 1 =====
```

위의 Generate 과정을 통해 변환된 자바스크립트 파일이 만들어지면서 Quick View 가 실행되는 것을 확인할 수 있는 것이다. 따라서 구성요소를 수정 한 뒤 실행을 원한다면 반드시 **Generate / Regenerate** 작업이 수행됨을 확인 해주어야 한다.



\*\* 킥 뷰를 통해 올바르게 실행하였다면 다음과 같은 창을 볼 수 있다.