## 프로젝트 생성

1. AndroidManifest.xml 편집

<manifest xmlns:android="http://schemas.android.com/apk/res/android">

<application android:allowBackup="true"

android:icon="@mipmap/appicon"

android:roundIcon="@mipmap/appicon\_round"

android:supportsRtl="true"

android:usesCleartextTraffic="true">

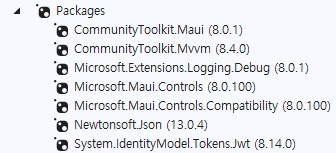
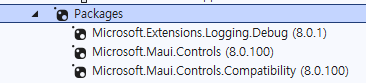
</application>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

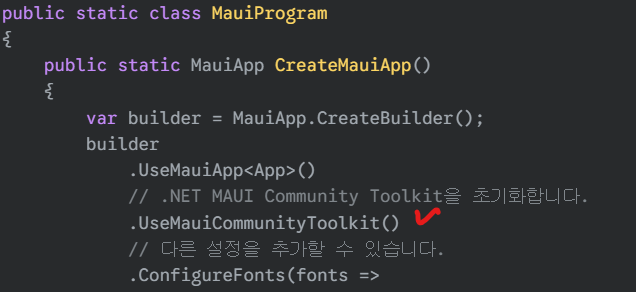
<uses-permission android:name="android.permission.INTERNET" />

</manifest>

## Nuget package 설치

>> MauiProgram.cs 편집: UseMauiCommunityToolkit() 추가



## PM90 SDK device.sdk.jar : pm90\_proj.docx참조

**. Android Java Binding Library 프로제트 생성**

**>> 프로젝트 속성**

**: Target framework 속성을 .NET 8.0 으로 변경 ( 기본 .NET 9.0 )**

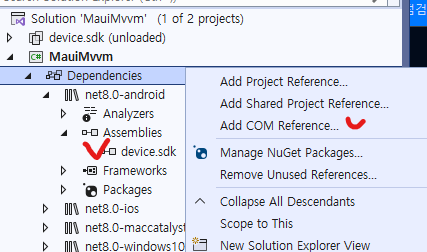
**: Target OS Version을 34로 변경**

**>> 프로젝트 루트에** device.sdk.jar 파일 추가 >> 파일 속성을 AndroidLibrary

>> **특정폴더에 dll, aar 파일 복사 (빌드 과정에서 이 네이티브 파일들이 최종 앱에 포함 )**

>> 생성된 dll을 프로젝트 참조에 추가

>> 프로젝트 unload





## 폴더 생성

. Services , Utils, Views, ViewModels, Models

## 서비스 추가

builder.Services.AddSingleton<IAuthService, AuthService>();

builder.Services.AddSingleton<IWebApiService, WebApiService>();

#if ANDROID

builder.Services.AddSingleton<ISimpleScannerService, MauiMvvm.Platforms.Android.SimpleScannerService>();

#endif

## 화면 추가하면 아래 순서로 진행

* .NET MAUI ContentPage(XAML) 추가 : LoginPage
* Viewmodel추가 ; LoginViewModel
* LoginPage 에서 ViewModel DI 후 BindingContext = viewModel;

public LoginPage(LoginViewModel viewModel)

{

InitializeComponent();

BindingContext = viewModel;

}

* AppShell.xaml.cs 에 Route추가

Routing.RegisterRoute(nameof(ListTest), typeof(ListTest));

* MauiProgram.cs DI를 위한 서비스 추가

builder.Services.AddSingleton<LoginPage>();

builder.Services.AddSingleton<LoginViewModel2>();

## 배포

* Release 모드에서 프로젝트 속성 변경
* 프로젝트 우클릭 >> Edit project file

>> <ApplicationId>com.nakdong.mauimvvm</ApplicationId>

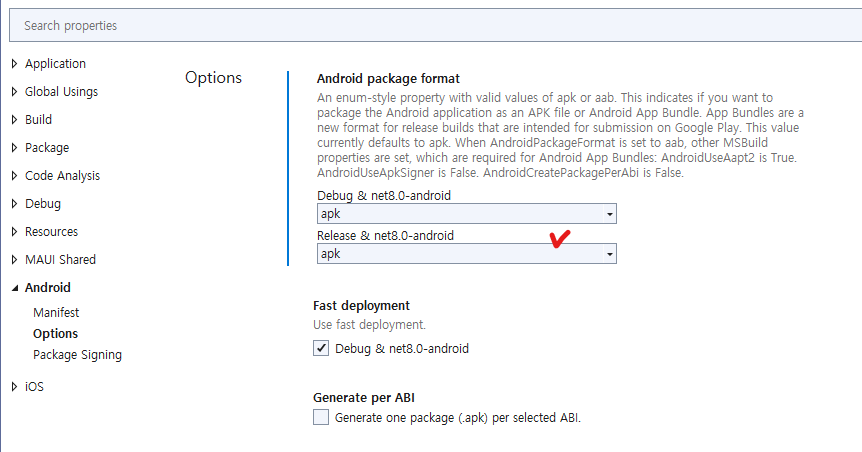
>>

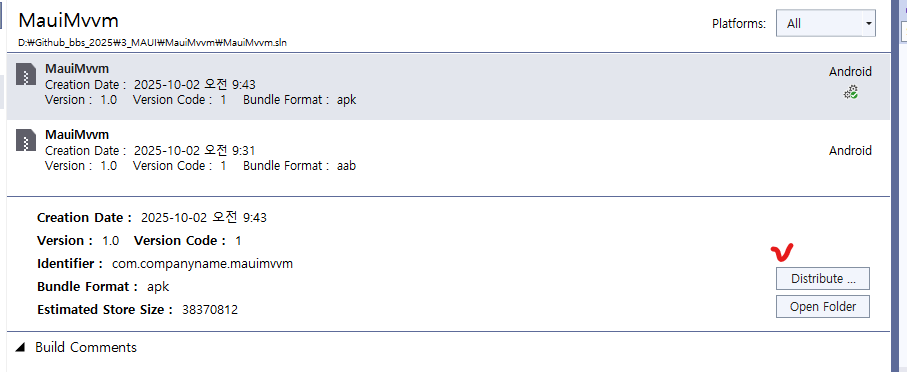
<!-- Versions -->

<ApplicationDisplayVersion>1.0</ApplicationDisplayVersion>

<ApplicationVersion>1</ApplicationVersion>

* Release 모드로 변경 >> 빌드
* Publish >> Distribution >> Ad Hoc >> Key선택 > Save As
* Apk 파일을 파일 다운로드 위치로 복사 > PDA에서 다운로드하여 설치





1. Edit Project File 에서 ApplicationId 변경 후 publish하니까 아래 오류

XABLD7000: Xamarin.Tools.Zip.ZipException: Renaming temporary file failed: Permission denied

**>> 프로세스가 잡고 있는듯. 노트북 리부팅 후 배포 성공**

제공해주신 오류 메시지 \*\*XABLD7000: Xamarin.Tools.Zip.ZipException: Renaming temporary file failed: Permission denied\*\*는 .NET MAUI (또는 Xamarin.Android) 빌드 과정에서 발생하는 **가장 흔한 파일 권한 오류** 중 하나입니다.

이 오류는 컴파일러가 임시 파일을 생성하고 정리하는 과정에서, 해당 파일이나 폴더에 접근하거나 이름을 변경할 **권한이 없기 때문에** 발생합니다.

**오류 원인: Permission denied (권한 거부)**

이 오류의 핵심 원인은 OS나 다른 프로그램이 빌드 관련 파일에 **잠금(Lock)을 걸고 있기 때문**입니다.

1. **Visual Studio나 IDE의 잔여 프로세스:** 이전 빌드나 디버깅 세션이 비정상적으로 종료되어, 백그라운드 프로세스가 일부 파일 핸들을 놓지 않고 있을 때 발생합니다.
2. **보안 소프트웨어/백신:** 때로는 백신 프로그램이나 보안 소프트웨어가 빌드 프로세스의 파일 접근을 의심스러운 행위로 판단하고 차단할 수 있습니다.
3. **파일 탐색기의 잠금:** 빌드 폴더(bin, obj)를 파일 탐색기나 터미널에서 열어둔 상태로 빌드를 시도할 경우 파일 잠금이 발생할 수 있습니다.

**해결책: 빌드 환경 초기화 (Clean-up)**

이 문제는 대부분 프로젝트의 **빌드 캐시를 완전히 제거**하고 시스템의 파일 잠금을 해제하면 해결됩니다.

**1. Visual Studio에서 정리 및 재시작**

가장 먼저 시도해야 할 표준적인 방법입니다.

1. Visual Studio를 **완전히 종료**합니다.
2. 작업 관리자(Task Manager)를 열어 **모든 C# 및 Visual Studio 관련 프로세스** (특히 VBCSCompiler.exe, MSBuild.exe 등)가 완전히 종료되었는지 확인합니다.

**2. 수동으로 캐시 폴더 제거 (강력 권장)**

이 방법은 빌드 캐시를 완전히 초기화하여 거의 모든 XABLD7000 오류를 해결합니다.

1. 솔루션 폴더를 엽니다.
2. **프로젝트 폴더** 내부로 이동합니다.
3. **bin** 폴더와 **obj** 폴더를 **모두 완전히 삭제**합니다. (이 폴더들은 다시 빌드할 때 재생성됩니다.)
4. Visual Studio를 다시 시작합니다.

**3. 보안 소프트웨어 확인**

위의 방법으로 해결되지 않는다면, 일시적으로 백신 프로그램이나 보안 소프트웨어의 **실시간 감시 기능**을 **잠시 중단**한 후 다시 빌드를 시도해 보세요. 만약 이 상태에서 빌드가 성공한다면, 해당 프로젝트 폴더를 백신 프로그램의 **예외 목록**에 추가해야 합니다.

캐시를 수동으로 정리하고 재빌드하면 문제가 해결될 것입니다.