iOS 12, Android 9時代の
今からでも始められるモバイル開発入門

第36回 エンバカデロ・デベロッパーキャンプ

株式会社シリアルゲームズ
取締役 / AppDiv3 マネージャー　細川 淳

embarcadero®
DEVELOPER CAMP

アジェンダ

- 最近のモバイル事情
- Delphi 10.3 Rio について
- Delphi 10.3 Rio によるモバイル開発

最近のモバイル事情

embarcadero DEVELOPER CAMP

iOS 最近のトピック

- 2018/12 現在のバージョンは iOS 12.1
- OpenGL ES が非推奨になり Metal に移行
  - FireMonkey は OpenGL ES で記述されている
  - FireMonkey が Metal で書き直されれば、既存コードも移行される？
- ギャンブル製のあるゲームが削除された
  - 削除の基準が良くわからない
- iPhone X からアスペクト比 2:1 の端末が出現
- iPhone X で「ノッチ」が出現
  - ノッチへの対応

Android 最近のトピック

- 2018/12 現在のバージョンは Android 9 (Pie)
- Android SDK Level 26 以上が必須に
  - 2019/8には 64bit が必須に
- Android 6 から実行時権限の取得が必要に
- Android 7 から動的リンクの禁止
- バックグラウンド動作の制限
  - 9 からはセンサーイベントやカメラなどへのアクセスが禁止される
- アスペクト比 2:1 の端末の増加

Delphi 10.3 Rio のトピック

- iOS 12 に対応
- Android SDK Level 26 に対応
- 言語の拡張
- ARC の廃止（まずは Linux コンパイラから）
- Linux AnsiString / AnsiChar 対応
- FireMonkey が Android ネイティブコントロールに対応
- FireMonkey で Android Z オーダーに対応

今がむしろ最適なタイミング！

Delphi 10.3 Rio について

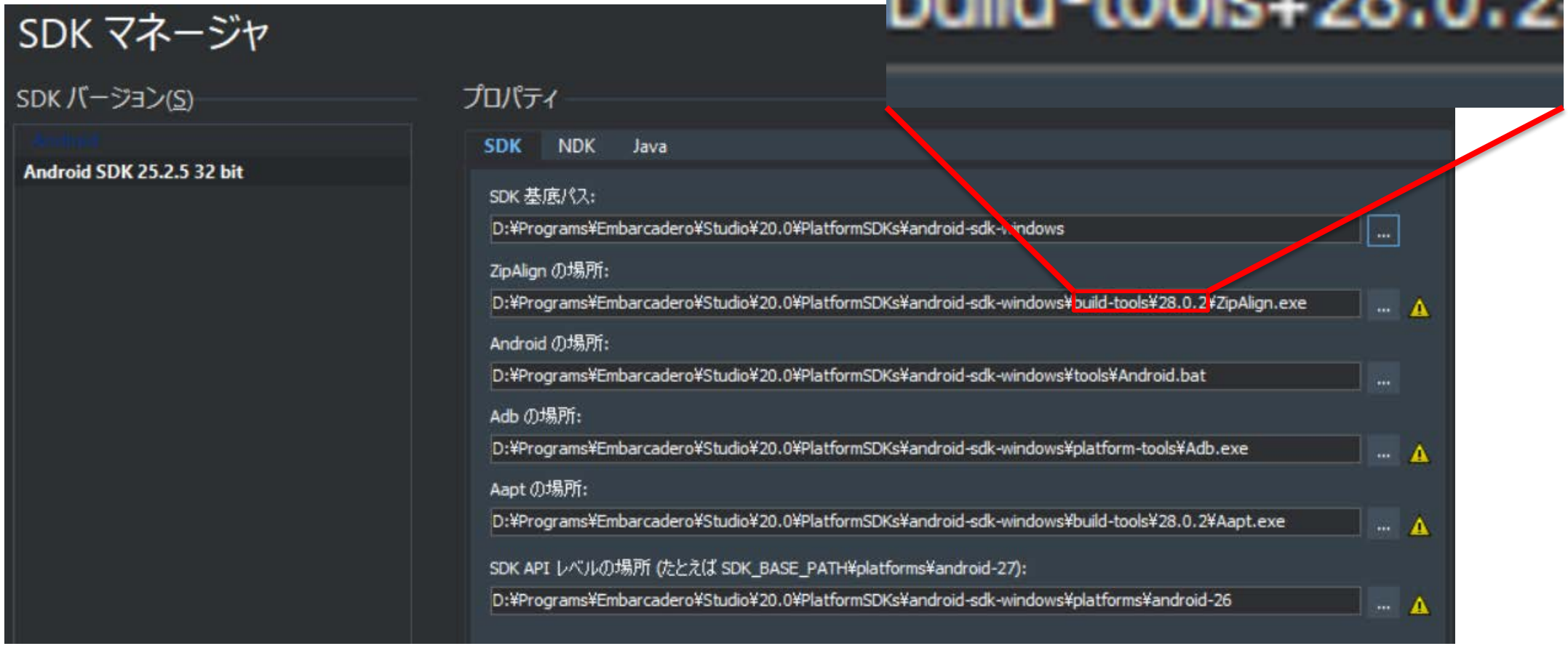
embarcadero DEVELOPER CAMP

モバイルプラットフォームに対する変更

- iOS
  - iOS 12 対応
  - Xcode 10 に対応
  - Xcode Command Line Tools も 10 に
- Android
  - Android SDK Level 26 対応
  - Android SDK 対応以外にも多くの変更がありました

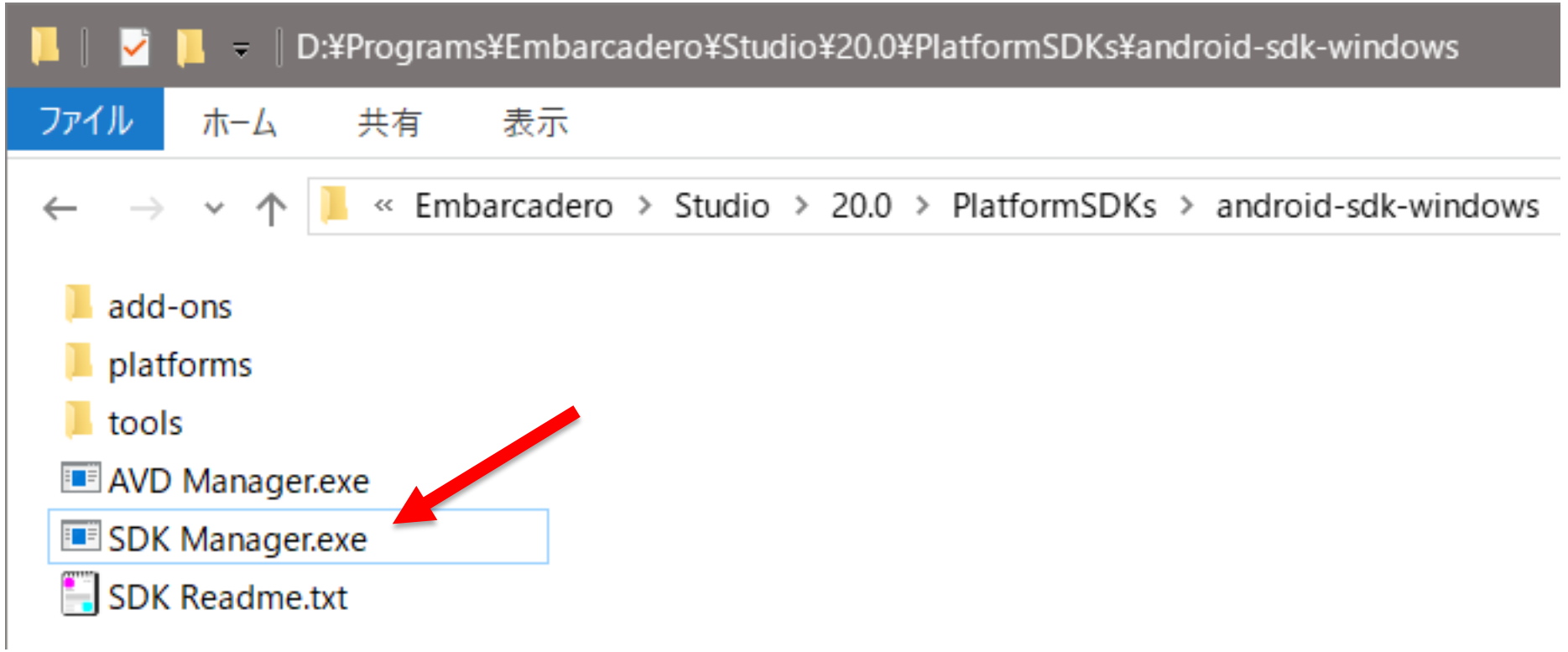
Android SDK Level 26 対応

- Android SDK
  - Level 28 がインストールされている…？

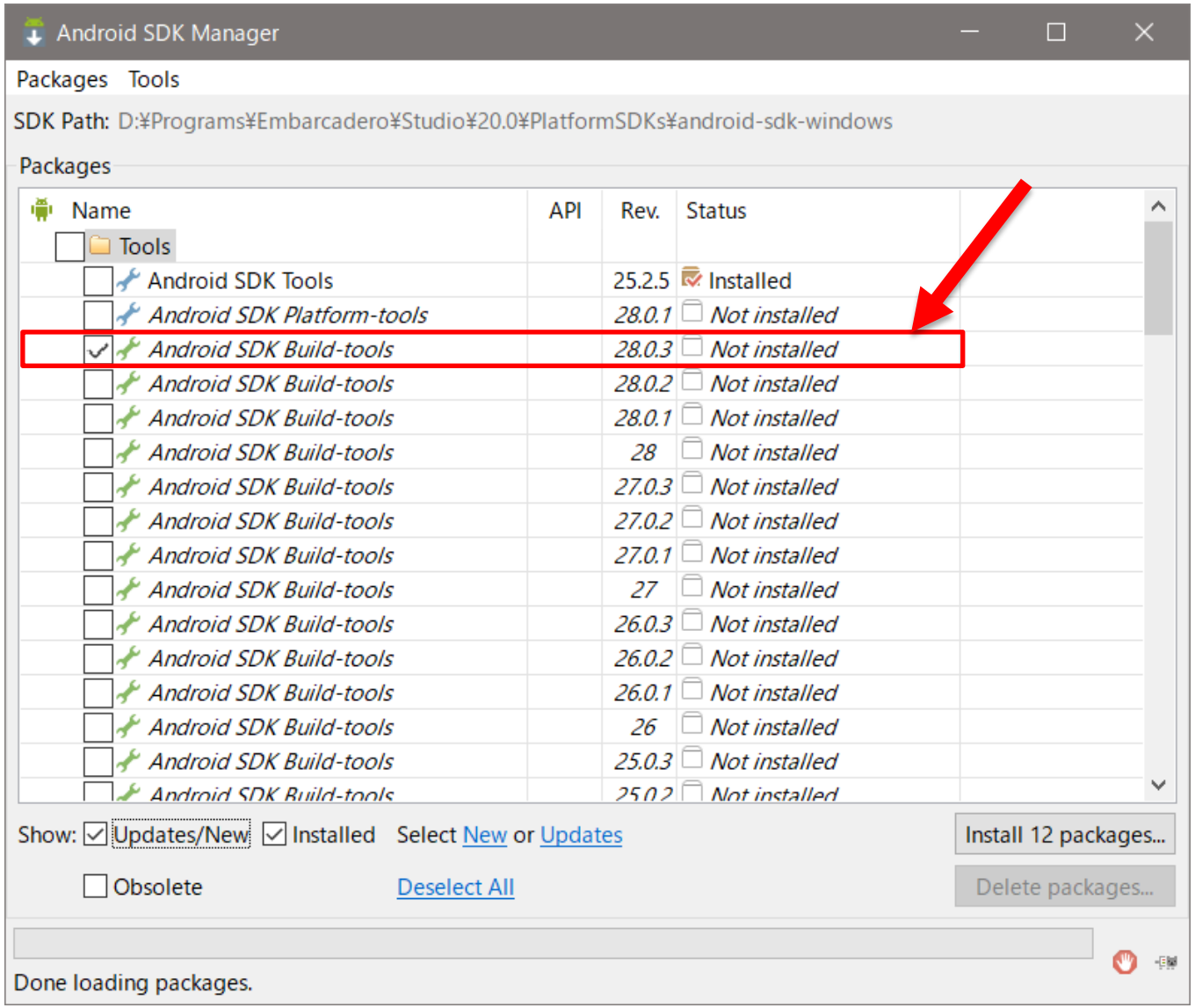


Android SDK Level 26 対応

- SDKマネージャでバージョンアップ

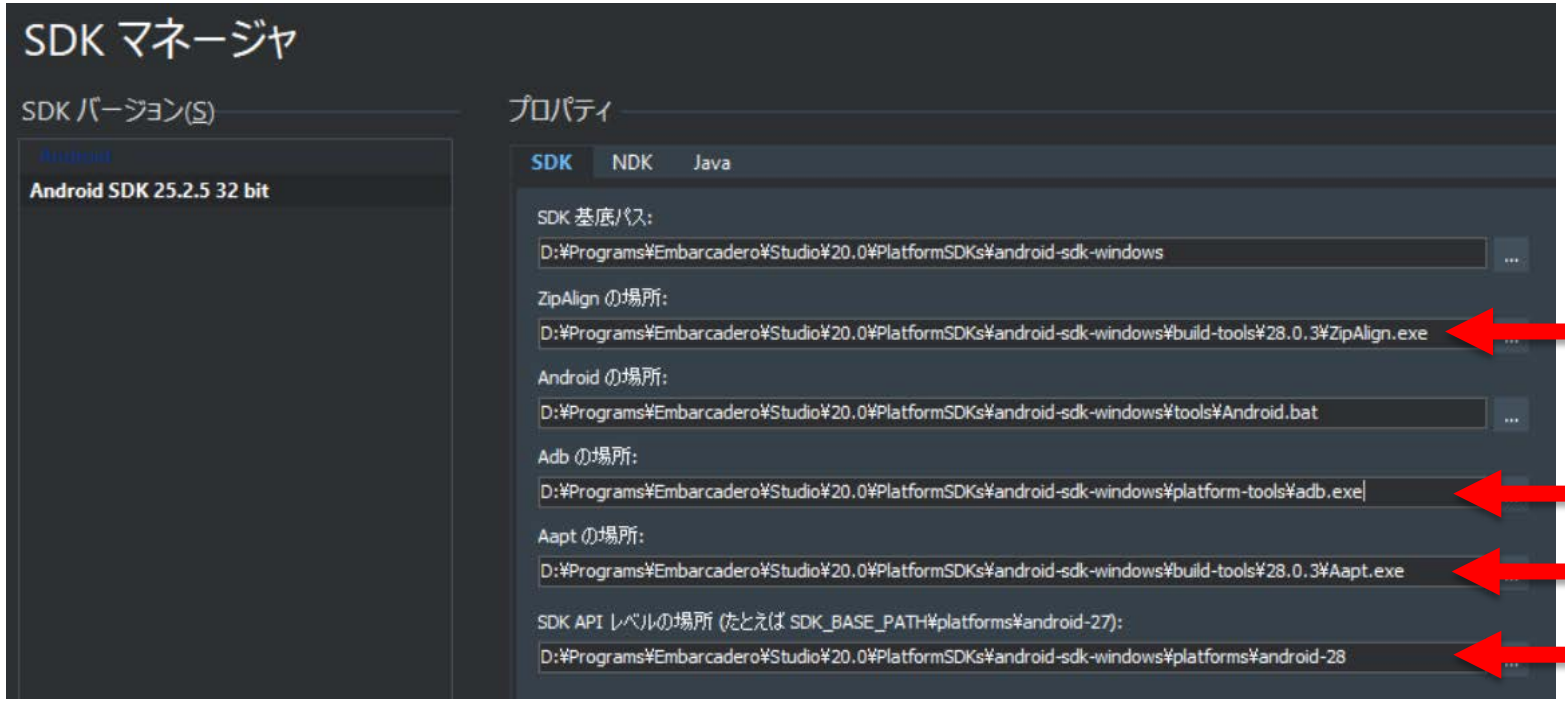


Android SDK Level 26 対応



Android SDK Level 26 対応

- パスを修正



28.0.2 ⇒ 28.0.3

Adb.exe ⇒ adb.exe

28.0.2 ⇒ 28.0.3

android-26 ⇒ android-28

言語拡張

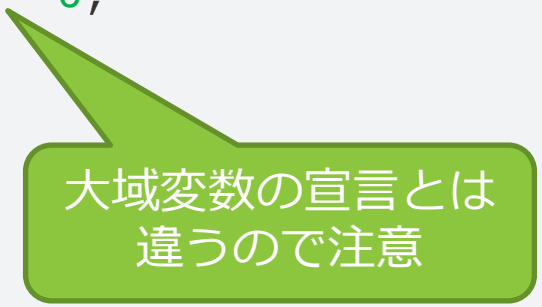
- 変数宣言ブロック var が必要なくなりました。
- Inline var 宣言
- Inline var 宣言によって型推論が可能に！
  - 今までの var ブロックによる宣言では使用される目的（右辺値）が不明なためやりたくてもできなかった
- 変数のスコープ・生存期間はブロック毎

言語拡張

- var ブロック不要の変数宣言(Inline var 宣言)

```
// 文中での変数宣言
procedure Foo;
begin
  var Bar: Integer;
  Bar := 0;
end;

// 文中での変数宣言と初期化を同時に
procedure Foo;
begin
  var Bar: Integer := 0;
end;
```



言語拡張

- 型推論

```
// 型推論
procedure Foo;
begin
  var Bar := 0; // 右辺値で自動的に型を特定する
  var Baz := 'Delphi 10.3 Rio !';

  // 型推論と組み合わせると今までの for 文に var を付けるだけ！
  for var i := 0 to 100 do
  begin
    // 何か処理
  end;
end;
```

言語拡張

- 型推論

```
// 型推論
procedure Foo;
begin
  var Qux := TDictionary<String, Integer>.Create;

  // 型推論と組み合わせると TPair<String, Integer> が不要に！
  for var Item in Qux do
  begin
    // 何か処理
  end;
end;
```

言語拡張

- 生存期間

```
procedure Foo;
begin
  var Bar := 0;
  if Bar = 0 then
  begin
    var Baz := 1;
  end;

  // ここでは Bar は見えるが Baz は見えない
  Writeln(Bar, Baz); // Bazが見えないのでコンパイルエラー
end;
```

言語拡張

- 生存期間

```
// 型推論
type
  IBar = Interface
    procedure Hello;
  end;

  TBar = class(TInterfacedObject, IBar)
  public
    procedure Hello;
  end;

procedure TBar.Hello;
begin
  Writeln('Hello, Delphi 10.3 Rio !');
end;
```

言語拡張

- 生存期間

```
procedure Foo;
begin
  if something = 0 then
  begin
    var Bar: IBar := TBar.Create;
    Bar.Hello;
  end; // Bar はここを抜けるとき廃棄される
end;
```

ARC 廃止

- ARC = Automatic Reference Counting
  - 参照カウントによるオブジェクト管理
- NEXTGEN 定義済みシンボル
  - Linux では無視される
- まずは Linux コンパイラからだが全てのコンパイラで廃止予定
  - 開発中の macOS 64bit コンパイラは最初から廃止されている

- 理由
  - TComponent の持つ Owner モデルとの関係
  - ARC 対応によるコンパイラの開発負荷増大
  - 自動挿入コードによる速度の低下

ARC 対応

- ARC 対応コンパイラでは…

```
type
  TBar = class(TObject)
  pubic
    procedure Baz;
  end;

procedure Foo;
var
  Bar: TBar;
begin
  Bar := TBar.Create;
  Bar.Baz;
end; // Bar はここで自動的に破棄
```

ARC 対応

- ARC 未対応コンパイラでは…

```
type
  TBar = class(TObject)
  pubic
    procedure Baz;
  end;

procedure Foo;
var
  Bar: TBar;
begin
  Bar := TBar.Create;
  try
    Bar.Baz;
  finally
    Bar.DisposeOf; // 自分で破棄
  end;
end;
```

ARC が廃止されるため破棄メソッドは Free でも
良くなるかも知れません

Free: オブジェクトを廃棄
DisposeOf: オブジェクトを廃棄しても良いと伝える

ARC 廃止

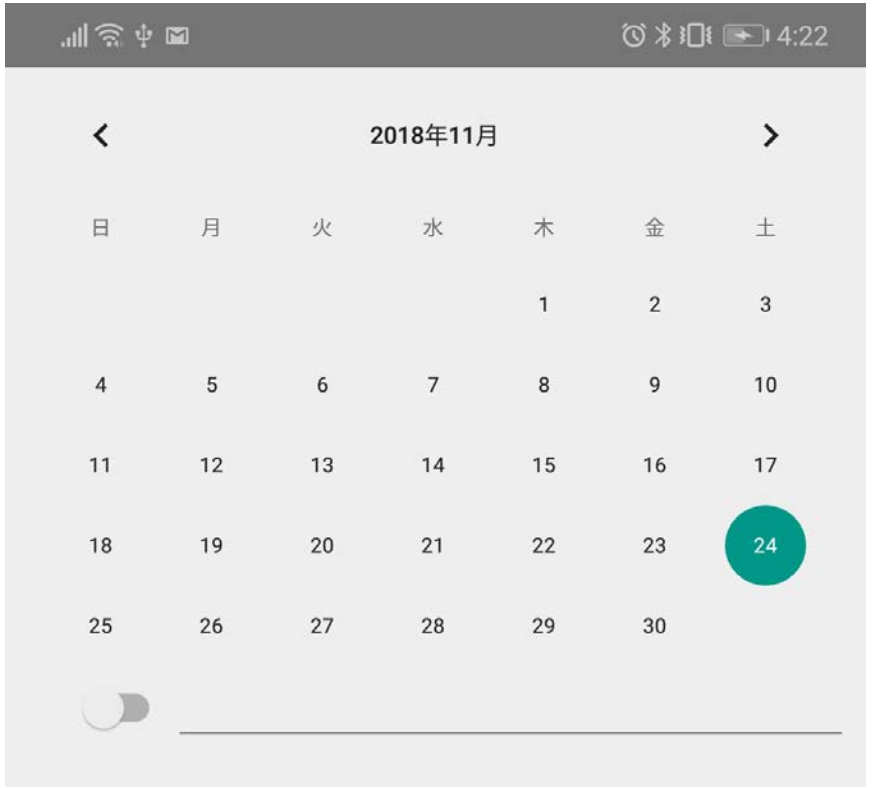
- TInterfacedObject の参照カウントは維持される
  - TOCImport, TJavaImport は TInterfacedObject を継承している
  - API へのアクセスコードはそのままでOK

```
function SaveJavaBitmapToFile(const iSrc: JBitmap; const iFileName: String): Boolean;
var
  PngFile: JFile;    // 自動廃棄
  OS: JOutputStream; // 自動廃棄
begin
  Result := False;
  PngFile := TJFile.JavaClass.init(TAndroidHelper.JStringToString(iFileName));
  OS := TJFileOutputStream.JavaClass.init(PngFile);
  try
    iSrc.compress(TJBitmap_CompressFormat.JavaClass.PNG, 100, OS);
    Result := True;
  finally
    OS.close;
  end;
end;
```

Android ネイティブコントロール

- ControlType プロパティが有効化された
  - TCalendar
  - TEdit
  - TSwitch

ControlType = Styled



ControlType = Platform

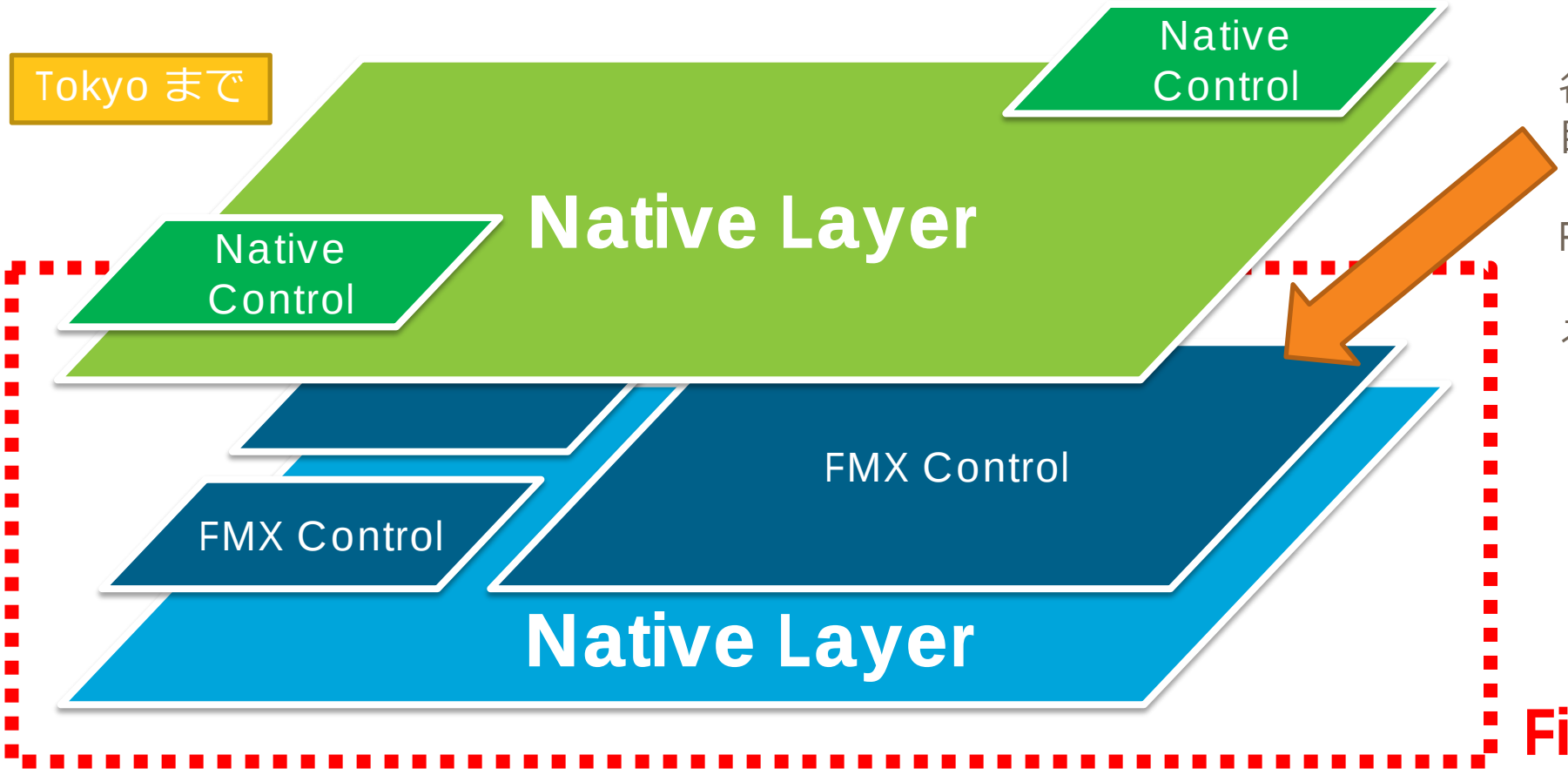
Android ネイティブコントロール

- Styled でも適切なスタイルを選べば、とても美しい！

DelphiStyle.com で購入した Calypso
※DelphiStyle.com は IDE のテーマを作っている所

Android Z Order

- Delphi 10.2 Tokyo までは下記の様な構造になっていて、FireMonkey のコントロールは Native Control より上には描画できませんでした。

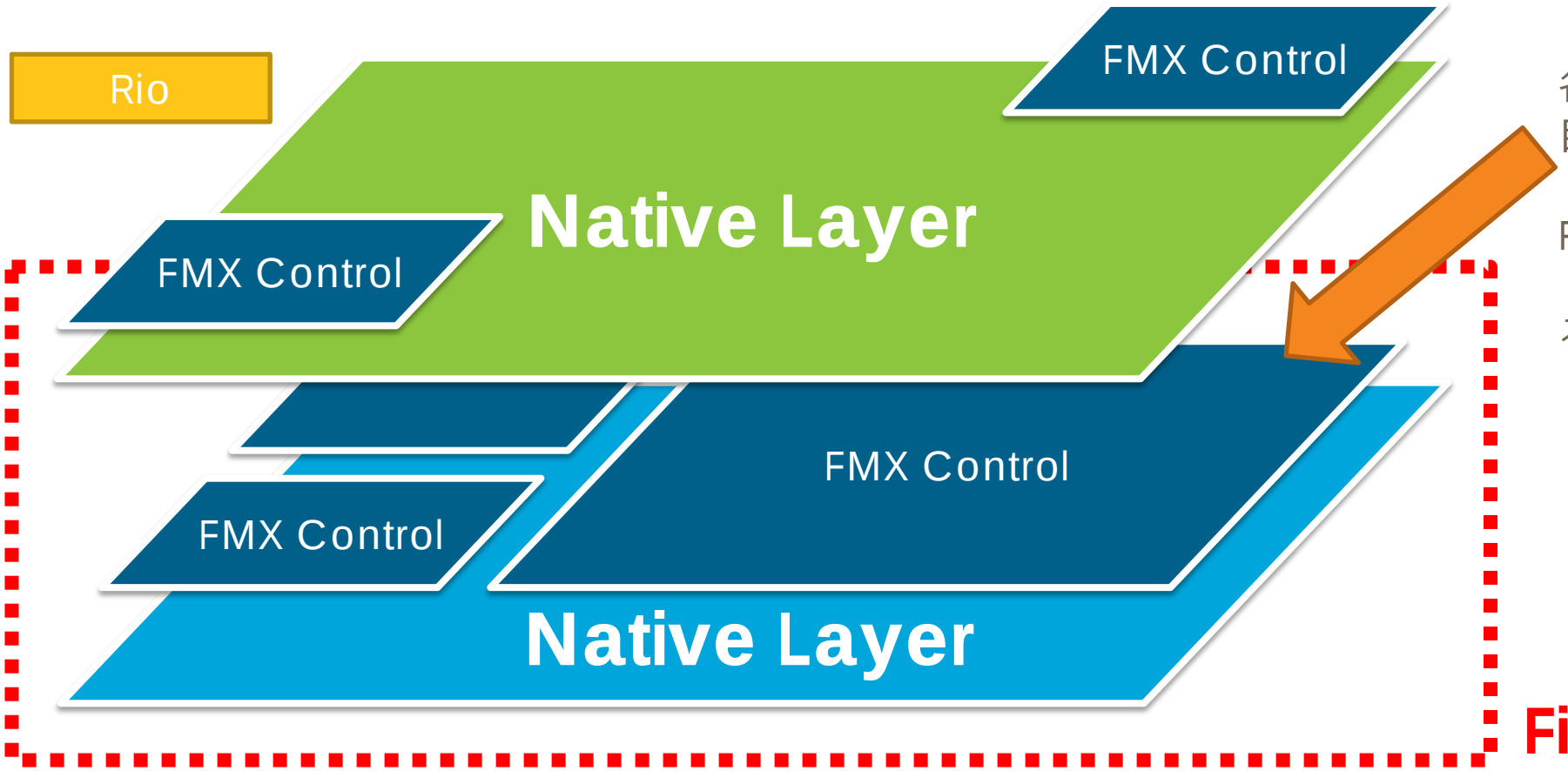


各 OS に依存した構造上に自分で描画している

FMX Control は「絵」！
OS から見た時に実体があるわけではない

Android Z Order

- 特定の条件を満たしたコントロールは**「Native コントロール上に置ける」**になりました。



各 OS に依存した構造上に自分で描画している

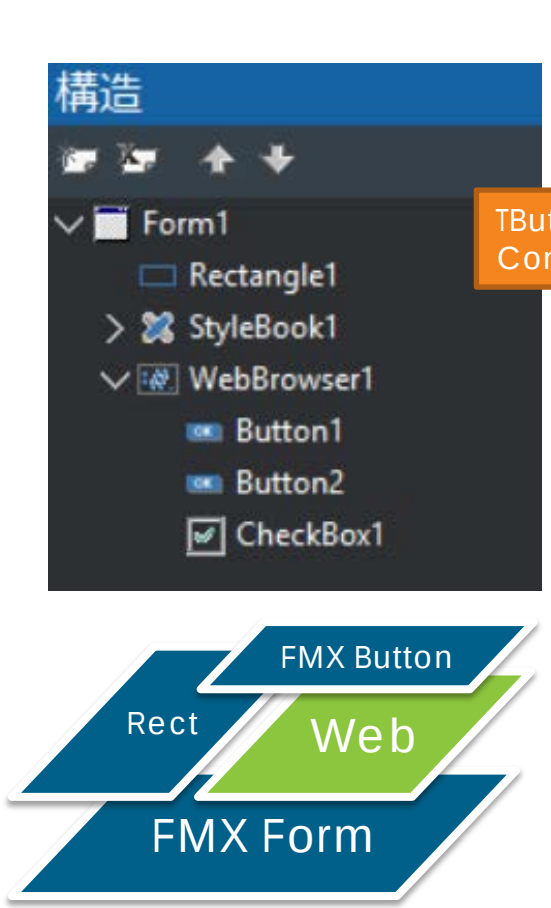
FMX Control は「絵」！
OS から見た時に実体があるわけではない

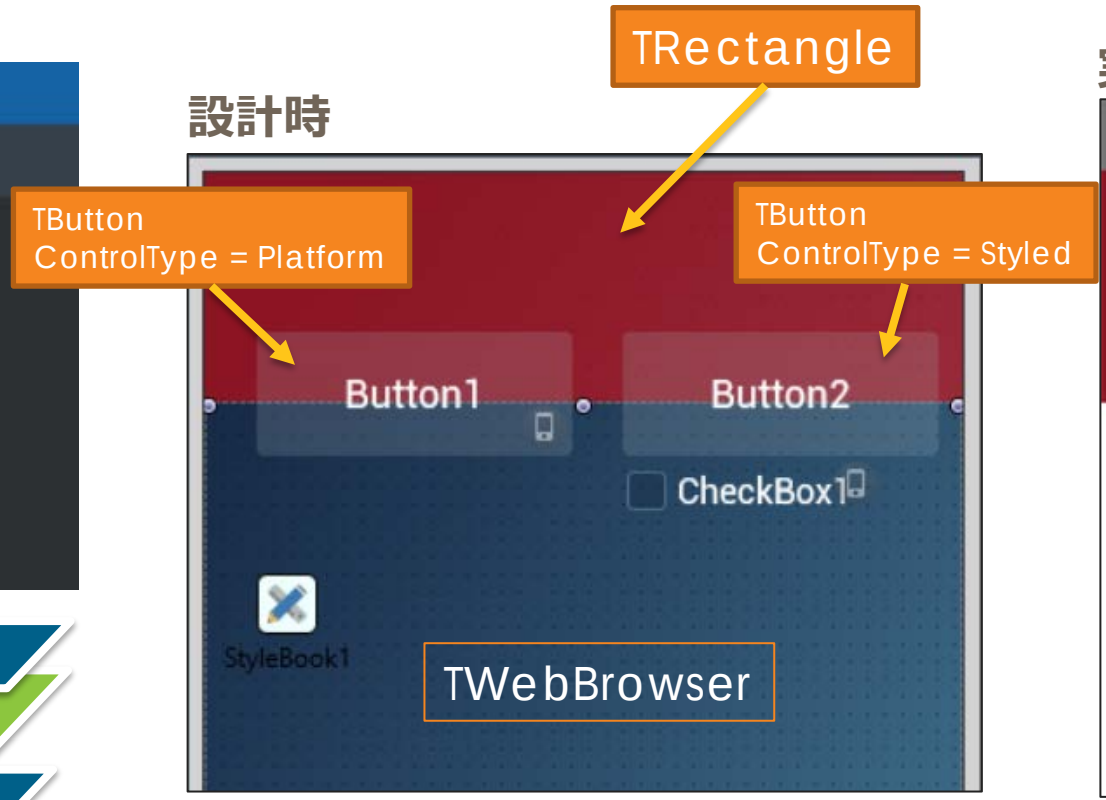
Androd Z Order

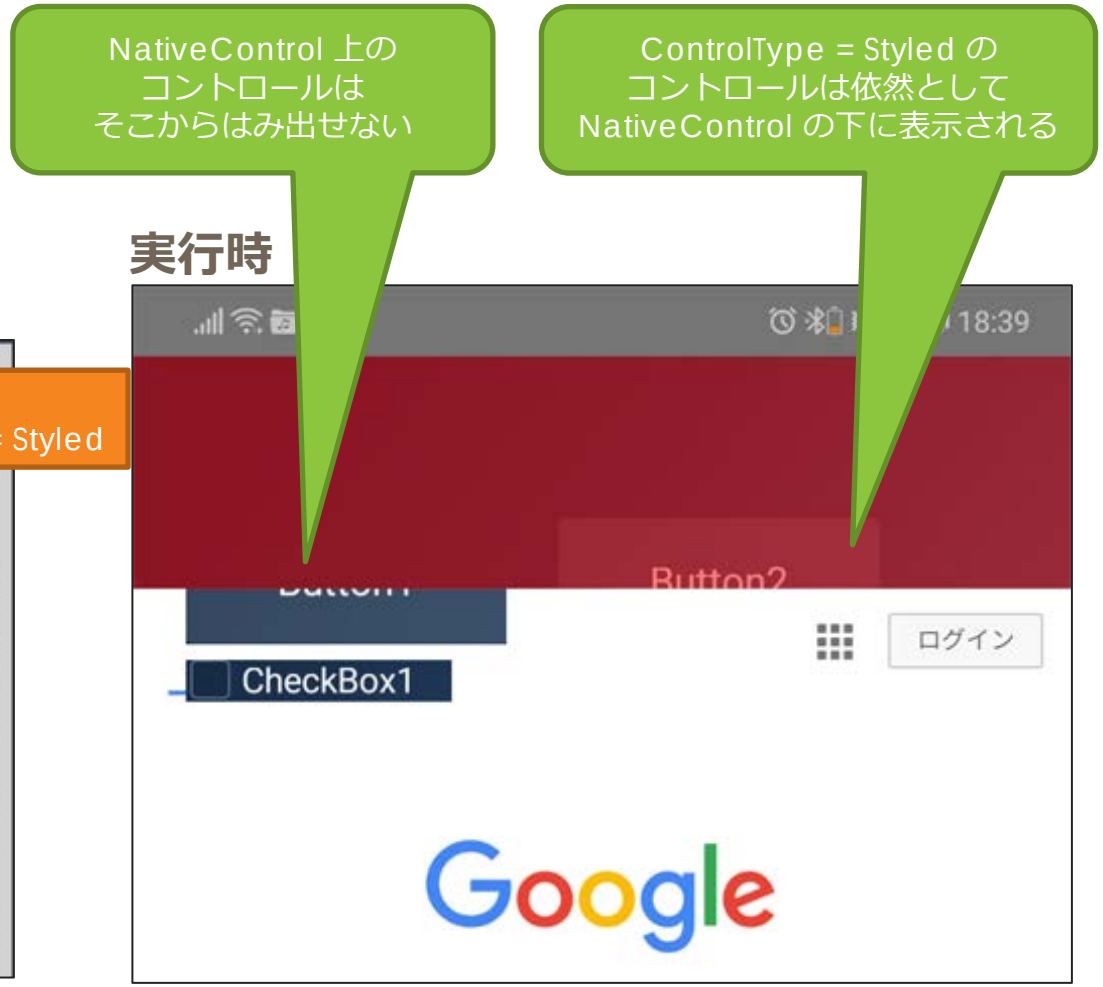
- 上に置ける条件
  - 親が Native Control であること
  - コントロールに ControlType プロパティがあること
    - 少し奇妙ですが ControlType = Platform にすると Native コントロールの上に置けます。

Android Z Order

- WebView の上に置いた例です







Android 対応 端末のアスペクト比

- アスペクト比 2:1 の端末に対応する方法
- AndroidManifest.template.xml の Application タグに " android.max_aspect " を追加します

AndroidManifest.template.xml 抜粋

```
<application android:hardwareAccelerated="true"
android:persistent="False"
android:restoreAnyVersion="False"
android:label="Project1"
android:installLocation="auto"
android:debuggable="True"
android:largeHeap="False"
android:icon="@…/ic_launcher"
android:theme="@…/AppTheme">

<meta-data android:name="android.max_aspect" android:value="2.1" />
```

(直下に)追加

Delphi 10.3 Rio によるモバイルアプリ開発

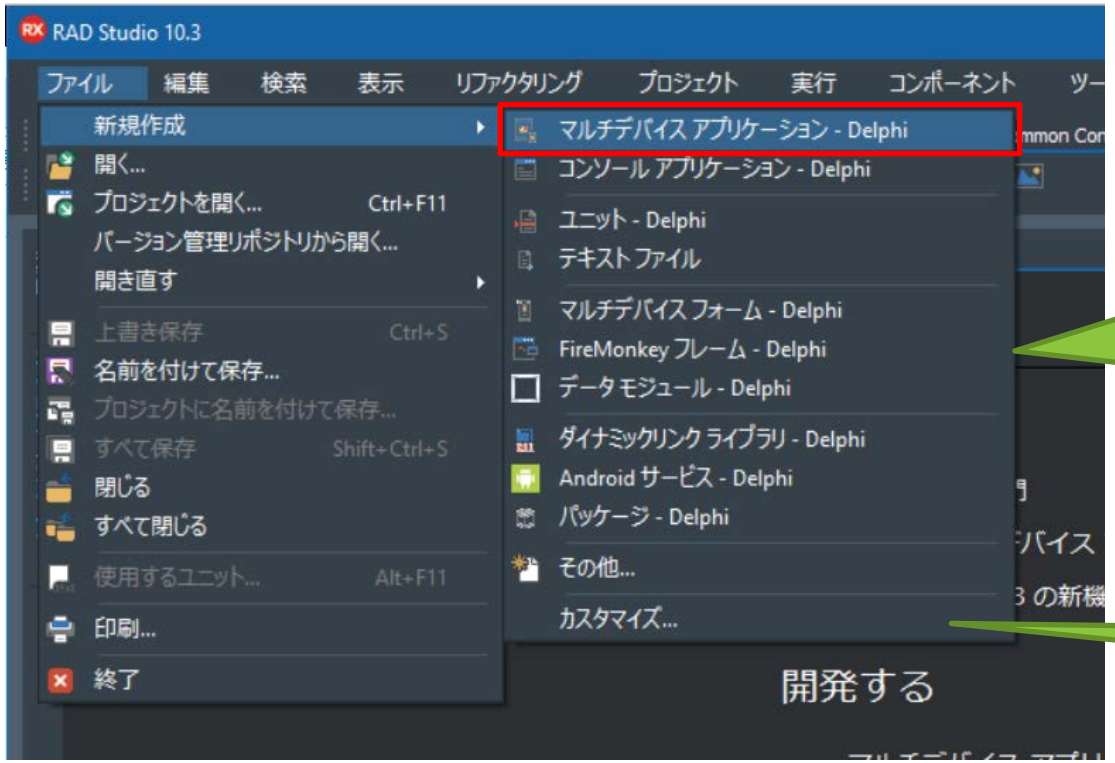
embarcadero®
DEVELOPER CAMP

Delphi 10.3 Rio によるモバイルアプリ開発

- FireMonkey や RTL が変わりましたが、基本的には今までと同じ！
- ですが、今までアプリケーションの作成を最初から最後まで紹介したことはありませんでした。
- そこで、今回はチュートリアル的に全ての流れを紹介します。
  - 画像ビューアを作ります

ImageViewer の作成

- まずは「ファイル」⇒「新規作成」から「マルチデバイス アプリケーション - Delphi」を選びます。

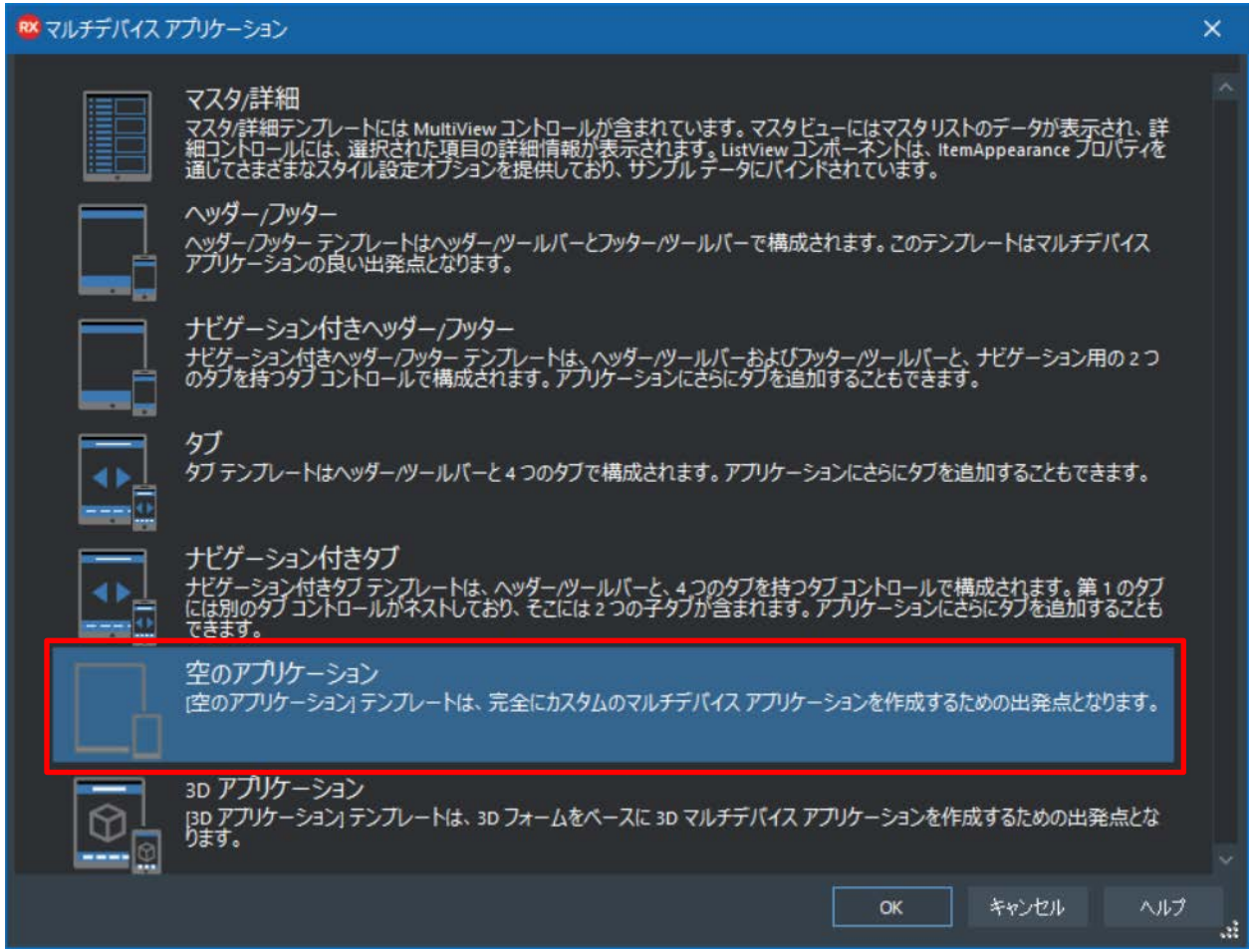


新規作成のメニューを
カスタマイズしているので
デフォルトの並びには
なっていません

ちなみに、カスタマイズはここから

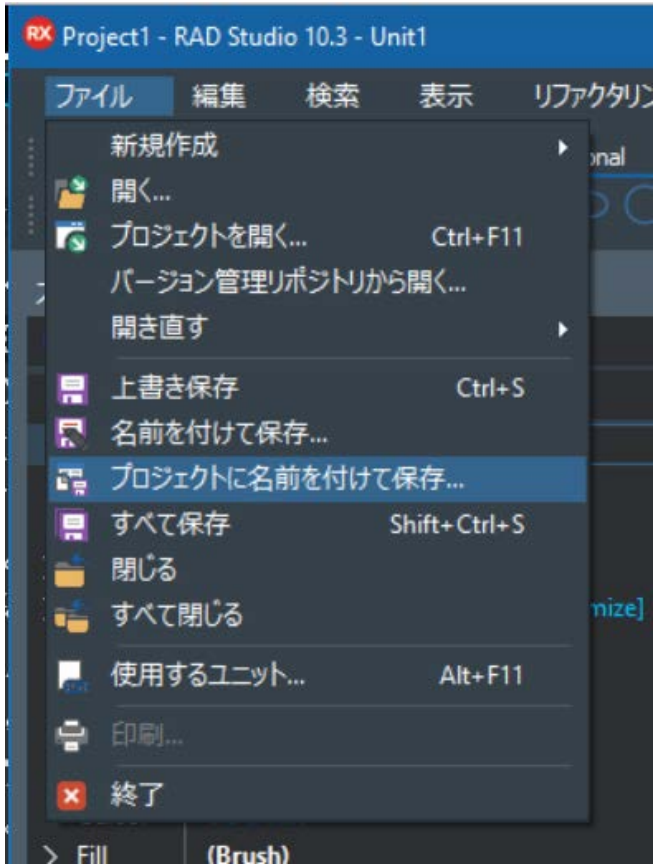
ImageViewer の作成

- 今回は「空のアプリケーション」を選びます



ImageViewer の作成

- まずは、「ファイル」⇒「プロジェクトに名前を付けて保存」でプロジェクトを保存してしまいましょう。

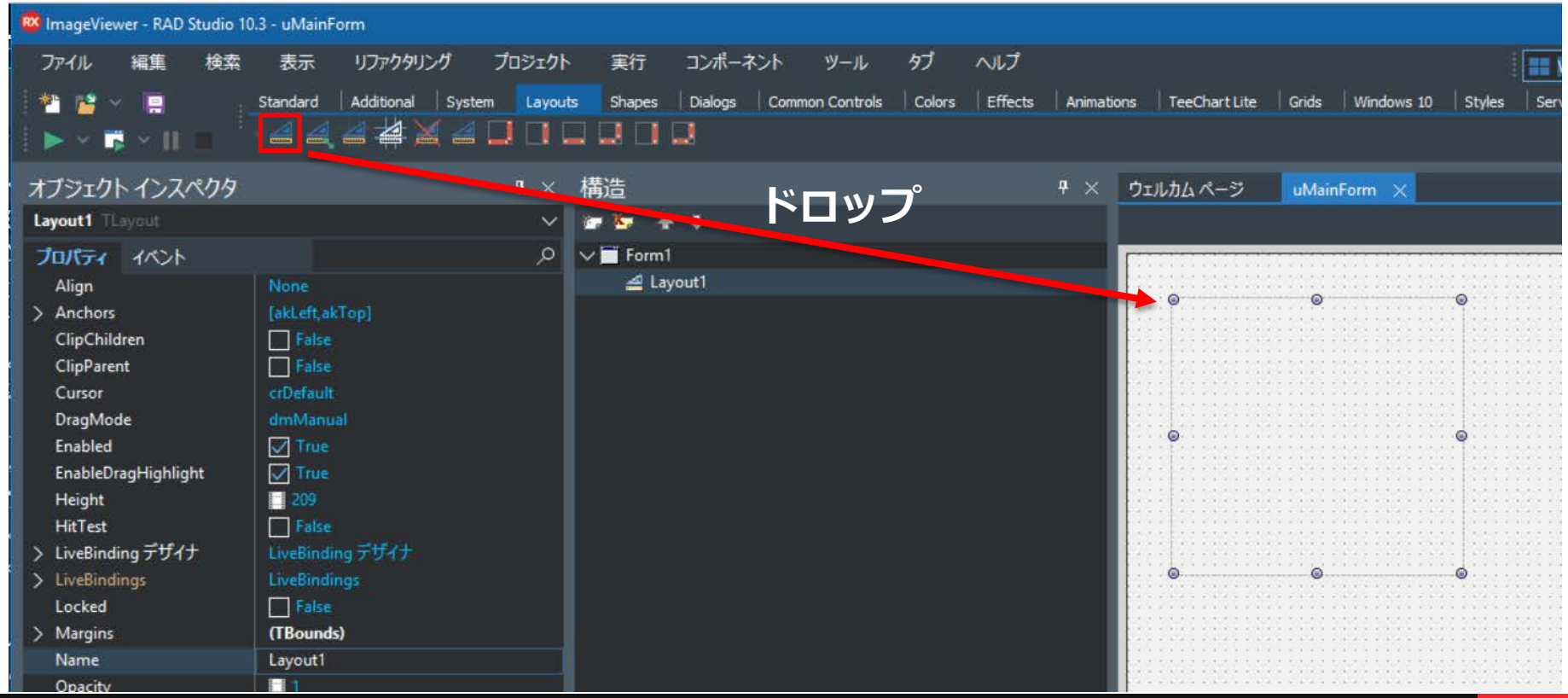


今回はこのようにしました。

プロジェクト名：Project1 ⇒ ImageViewer
ユニット名：　　Unit1 ⇒ uMainForm

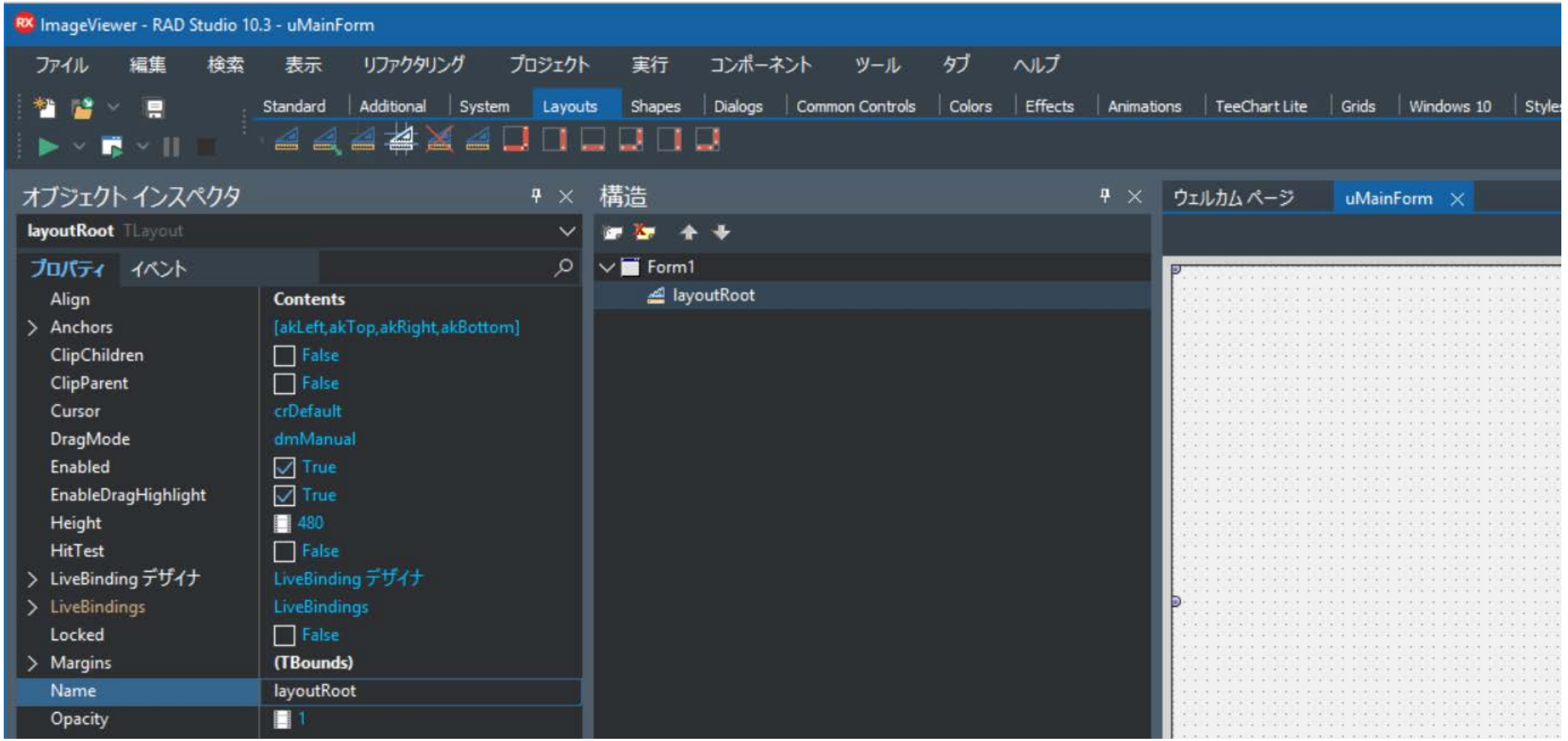
ImageViewer の作成

- まずは TLayout をフォームに置きます。
- この Layout の上にコントロールを配置していきます。
- Form に直接置かないのは大幅なコントロールの配置し直しの時にやりやすいのと、**AdjustPan** がやりやすいためです。



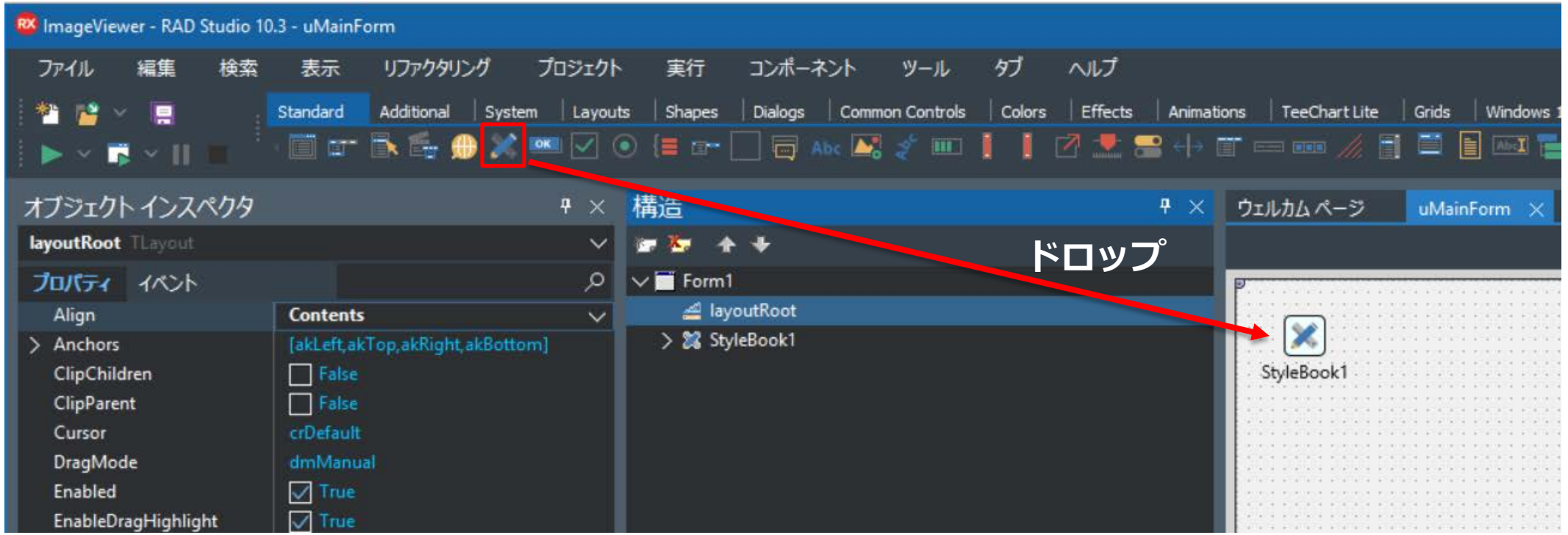
ImageViewer の作成

- Layout1 を layoutRoot として、Align を Contents にします。



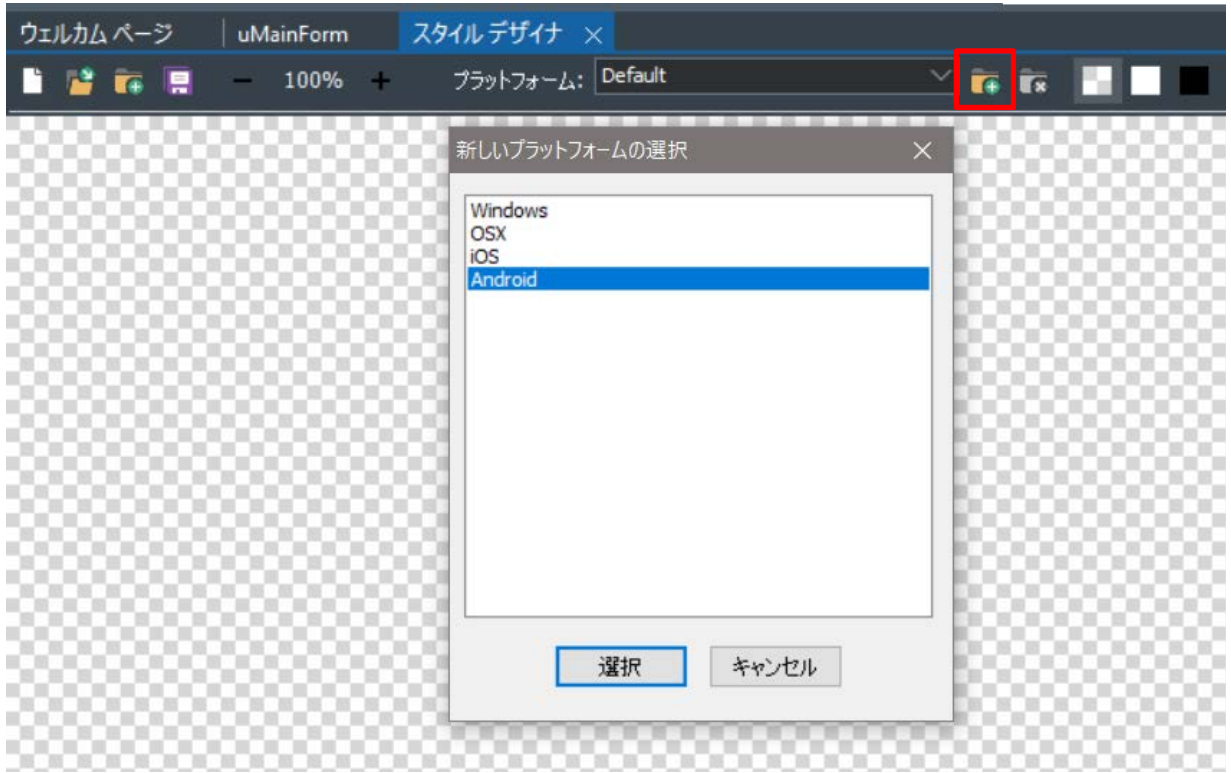
ImageViewer の作成

- 次に StyleBook を置きます。



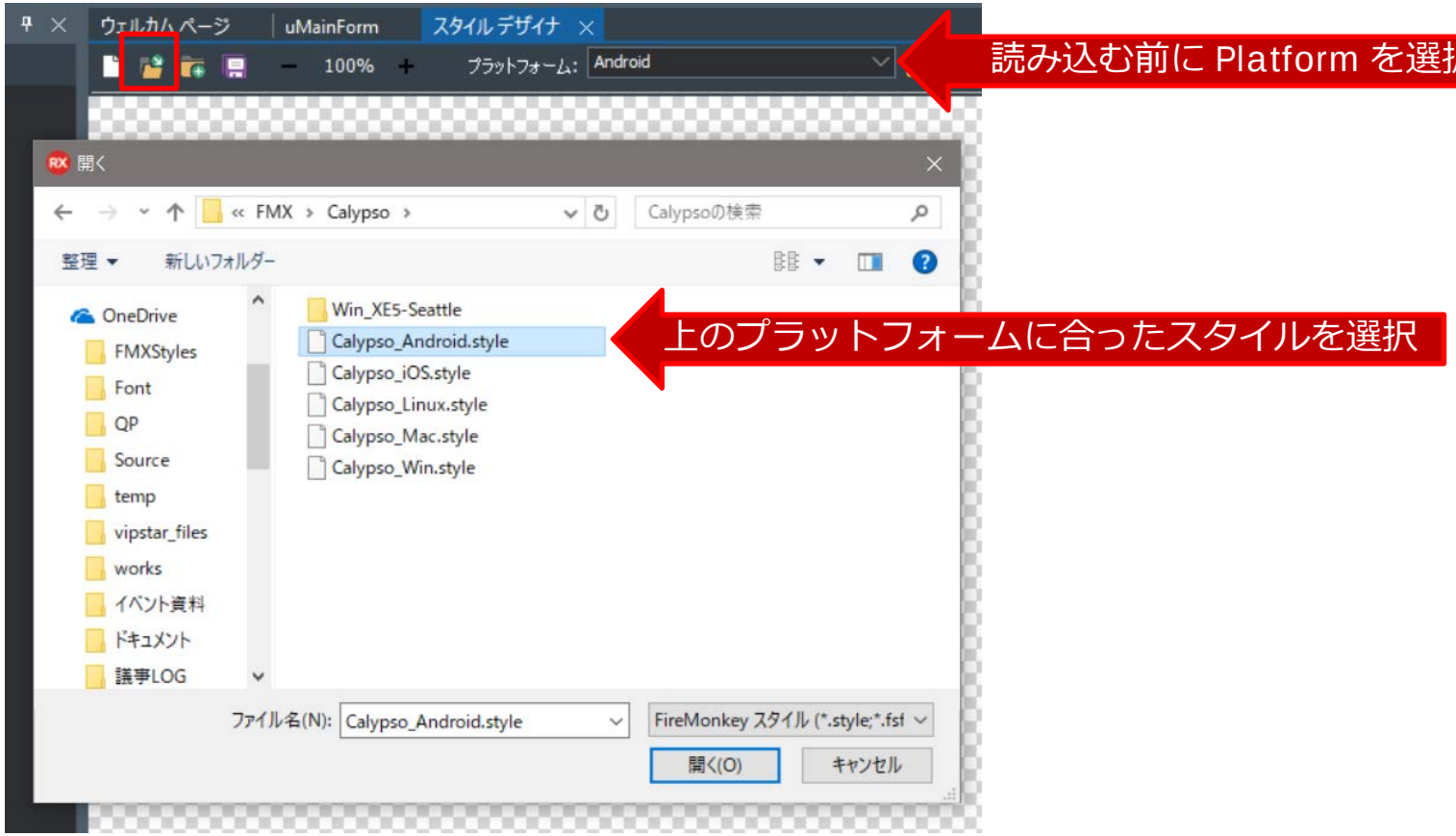
ImageViewer の作成

- StyleBook にスタイルを読み込ませます。
- このとき、それぞれの OS に合ったスタイルがある場合は「＋」ボタンを押してプラットフォームを追加します。
- ここでは、Android と iOS を追加します。



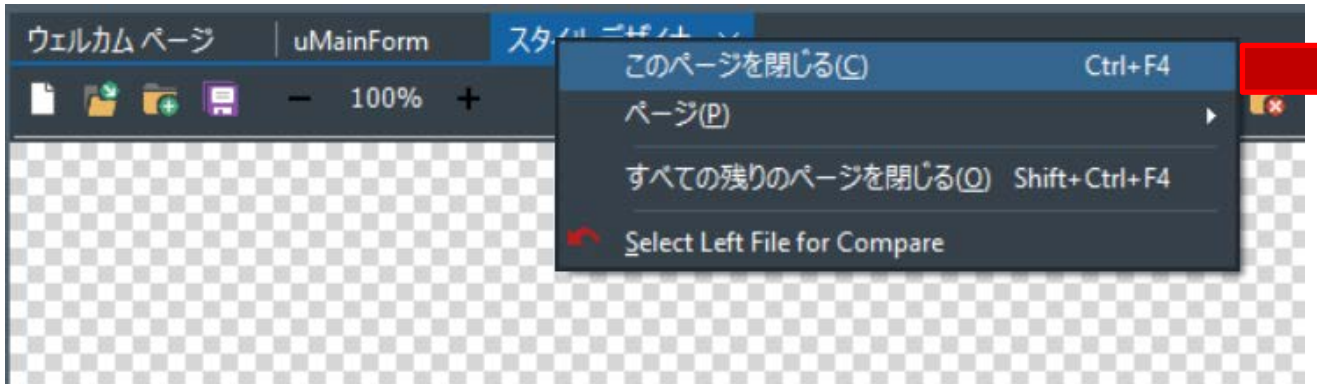
ImageViewer の作成

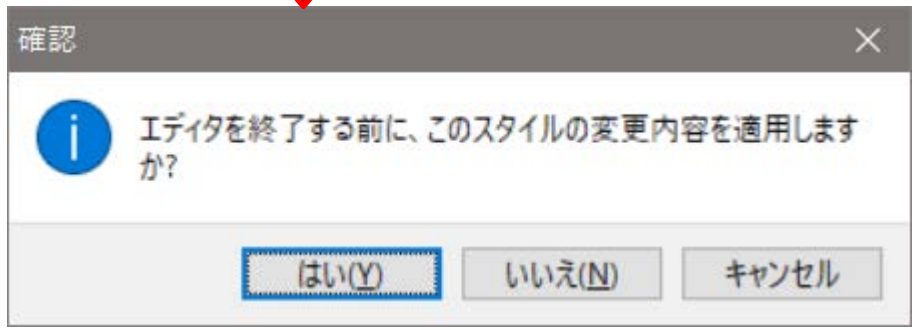
- フォルダを開くアイコンをクリックして、それぞれのスタイルを読み込みます。
- ここでは、DelphiStyles.com で購入した Calypso スタイルを使用します。



ImageViewer の作成

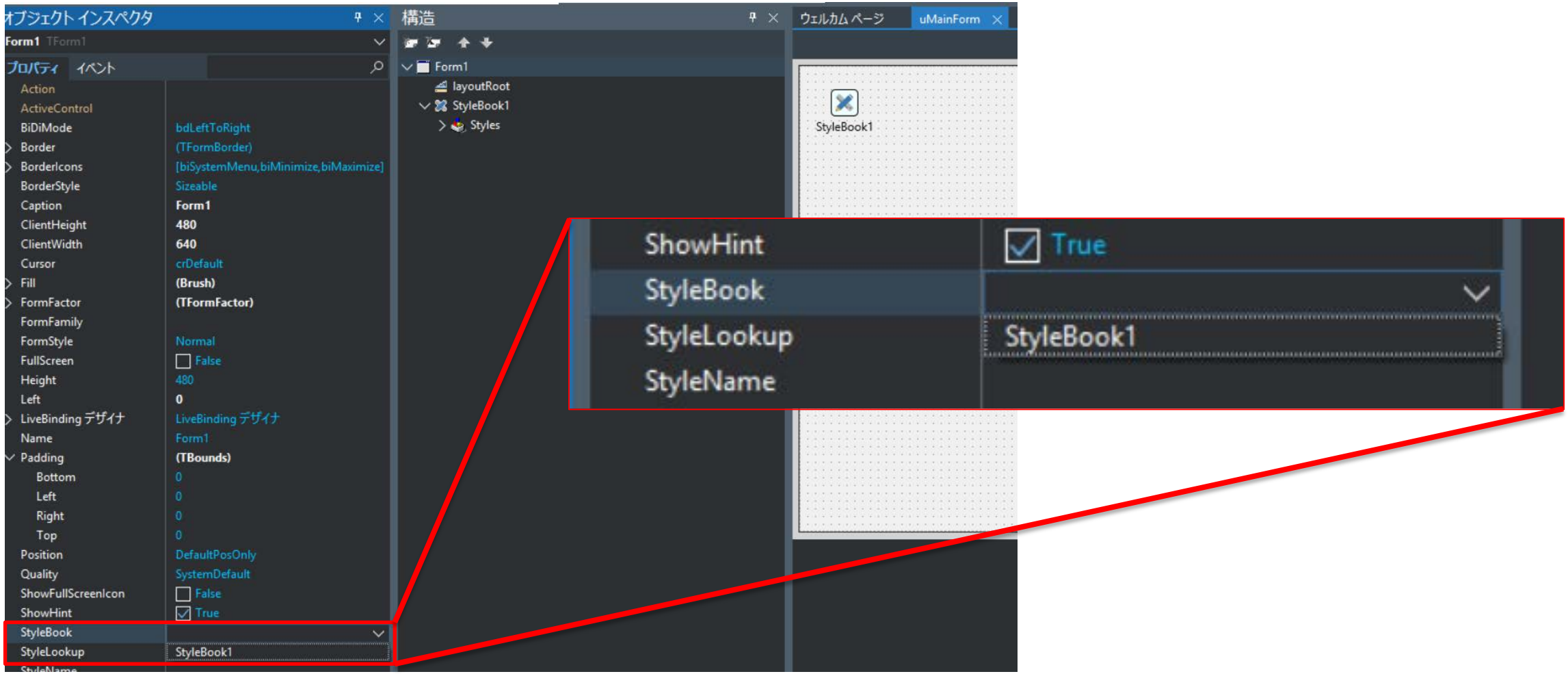
- 読み込ませ終わったら、「スタイルデザイナ」タブを右クリックして「このページを閉じる」を選びます。
- すると、ダイアログが表示されるので「はい」を選びます。





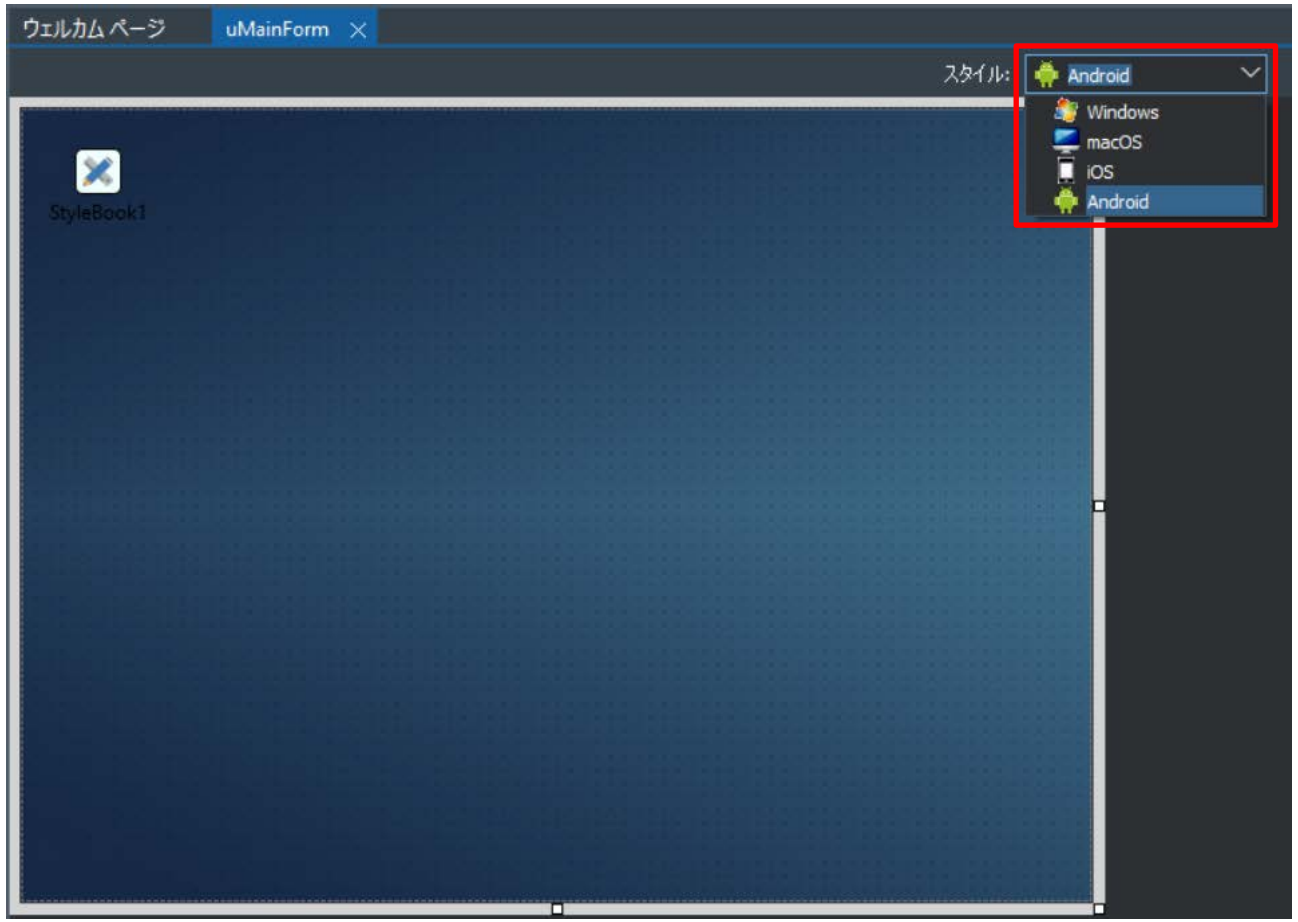
ImageViewer の作成

- Form1 に StyleBook1 を設定します。



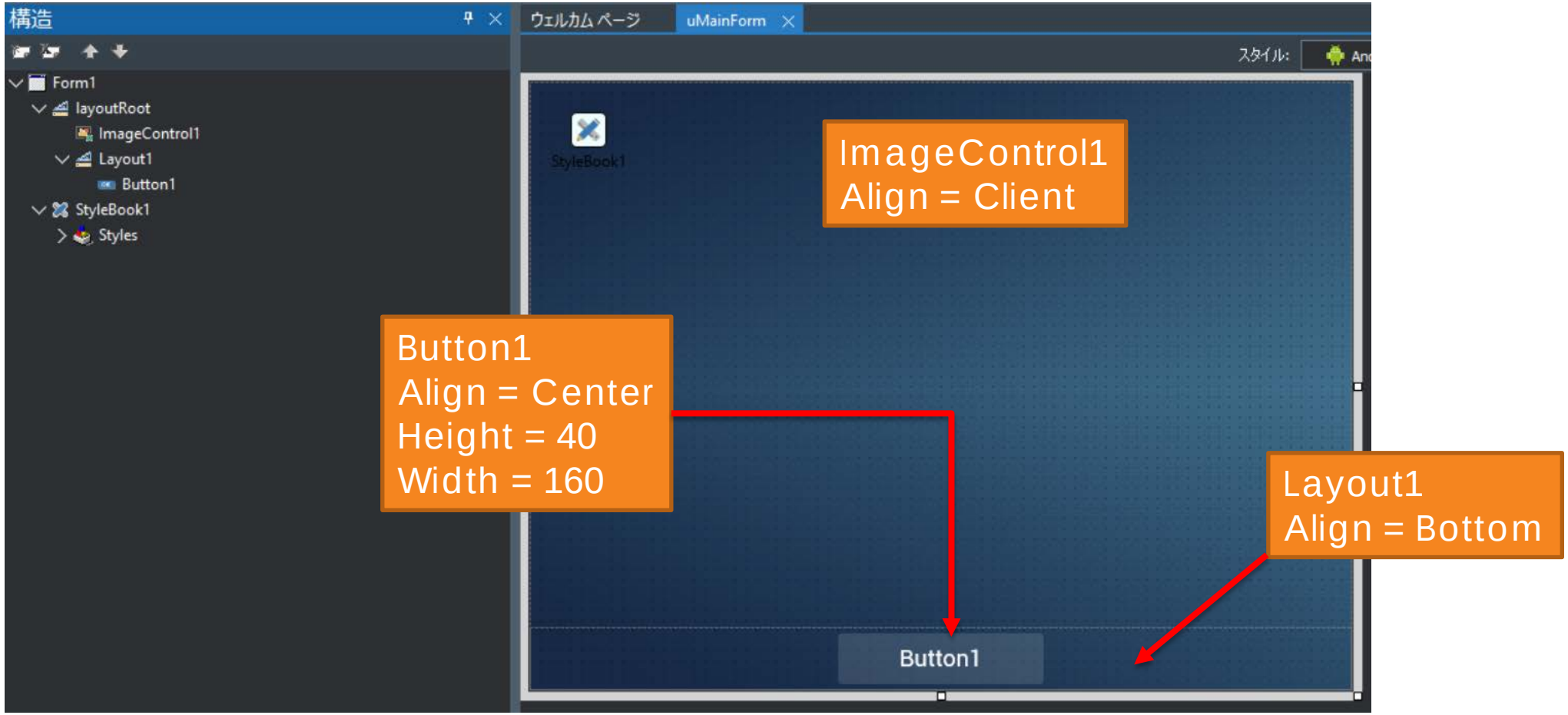
ImageViewer の作成

- 次にエディタの上にある「スタイル」ドロップダウンで「Android」や「iOS」を選ぶと設定されたスタイルによって見た目が変わります。



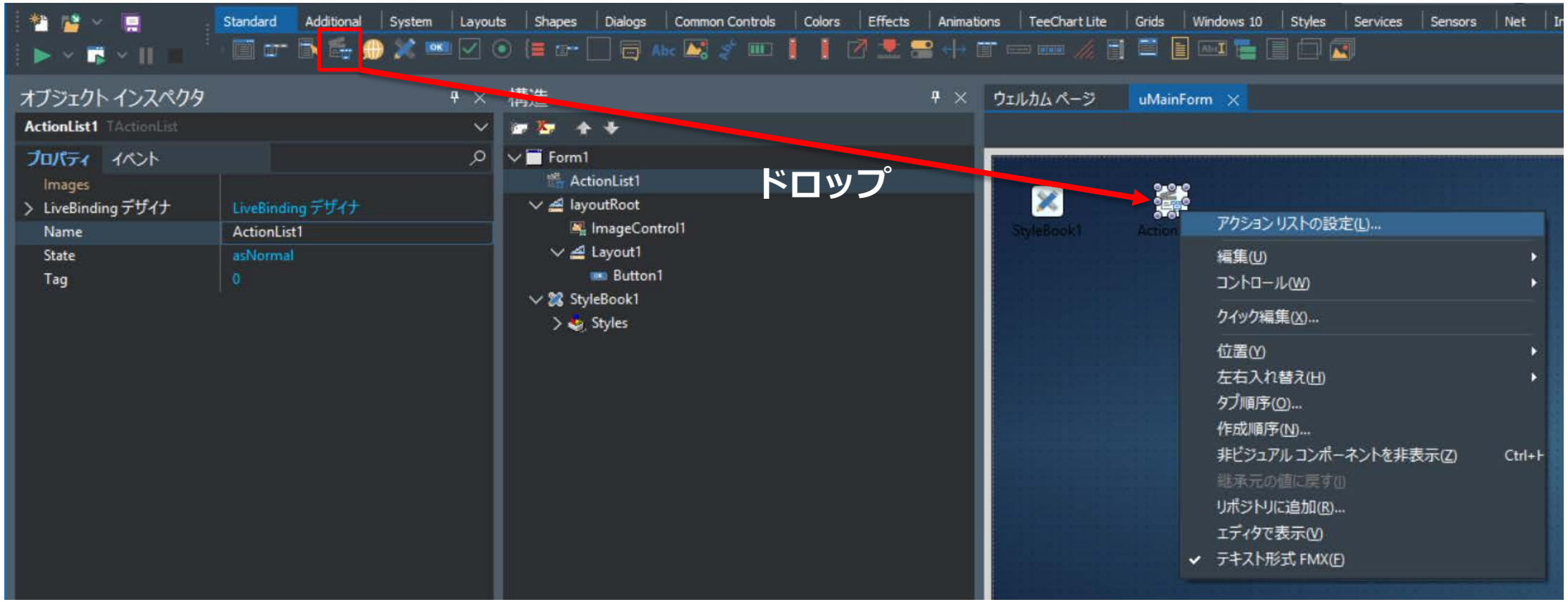
ImageViewer の作成

- ここから UI を構築します。
- まずは下記の様に配置しました



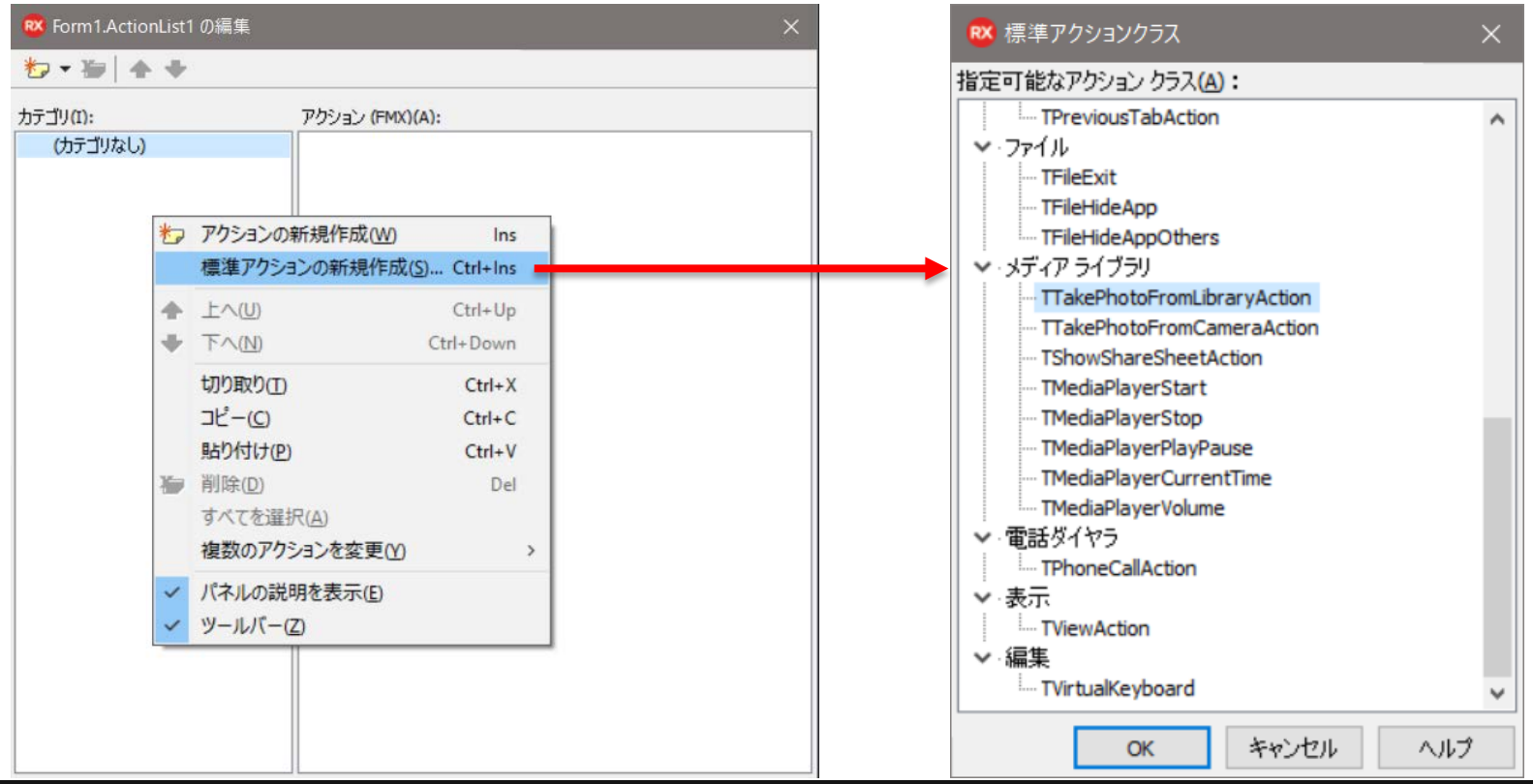
ImageViewer の作成

- 「ActionList」を置き、右クリック「アクションリストの設定」をクリックします



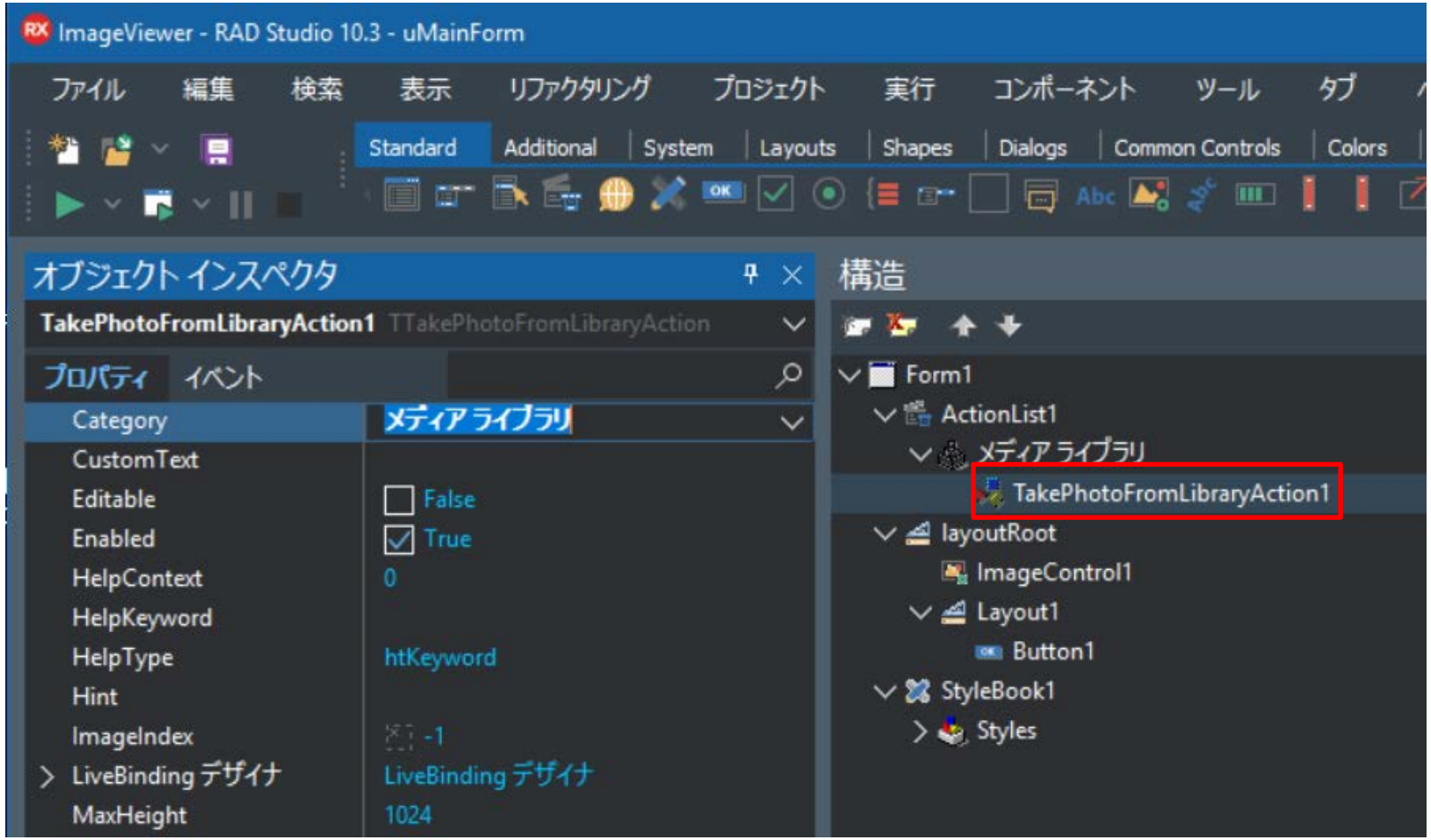
ImageViewer の作成

- 開いた編集ウィンドウの「カテゴリ」ペインを右クリックして「標準アクションの新規作成」を選びます。
- 標準アクションクラスウィンドウが開くので「TTakePhotoFromLibraryAction」を選択します。



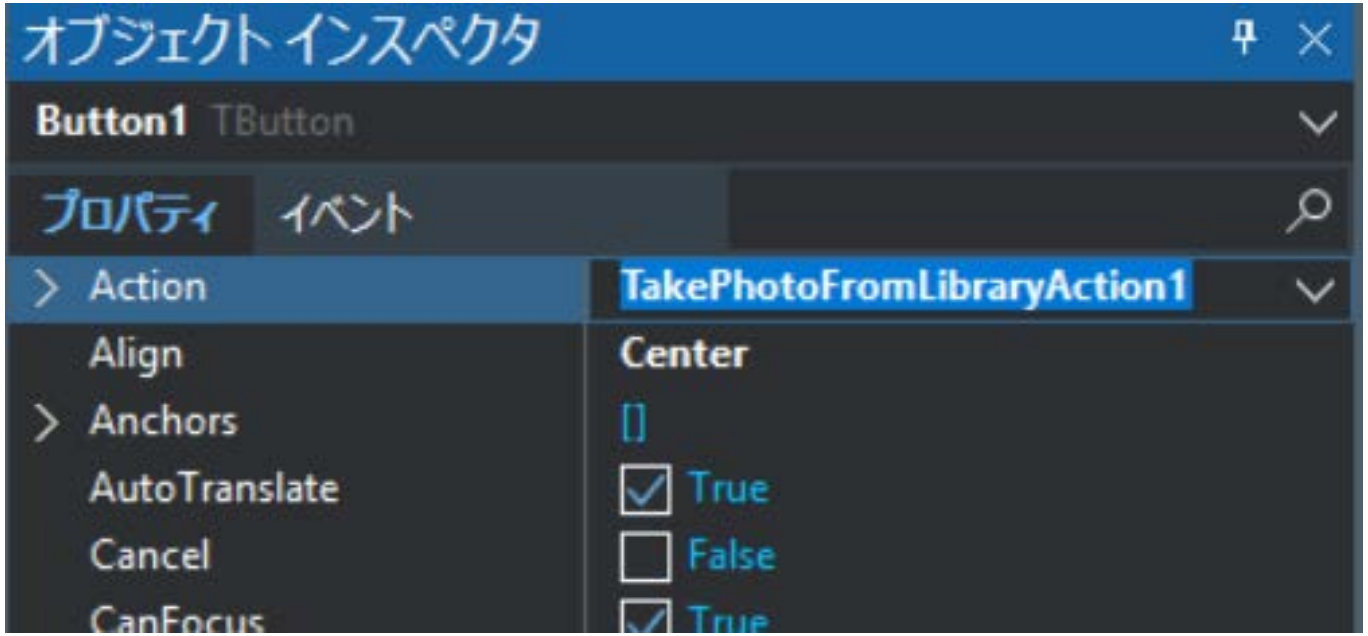
ImageViewer の作成

- ActionList1 の下に TakePhotoFromLibraryAction1が生成されます



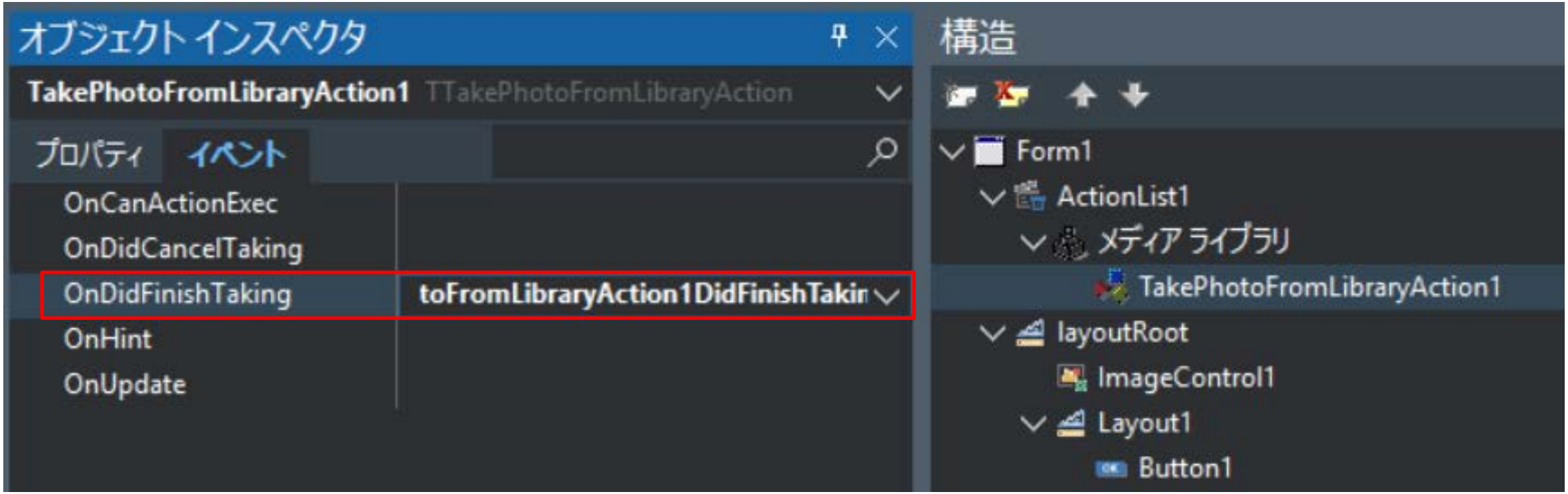
ImageViewer の作成

- このアクションを Button1 に割り当てます。
- ※アクションを割り当てると Text プロパティの値が代わり自動的に「フォトライブラリ」となります。必要に応じて変更します。



ImageViewer の作成

- 次に TakePhotoFromLibraryAction1 の「OnDidFinishTaking」イベントをダブルクリックしてイベントハンドラを生成します。



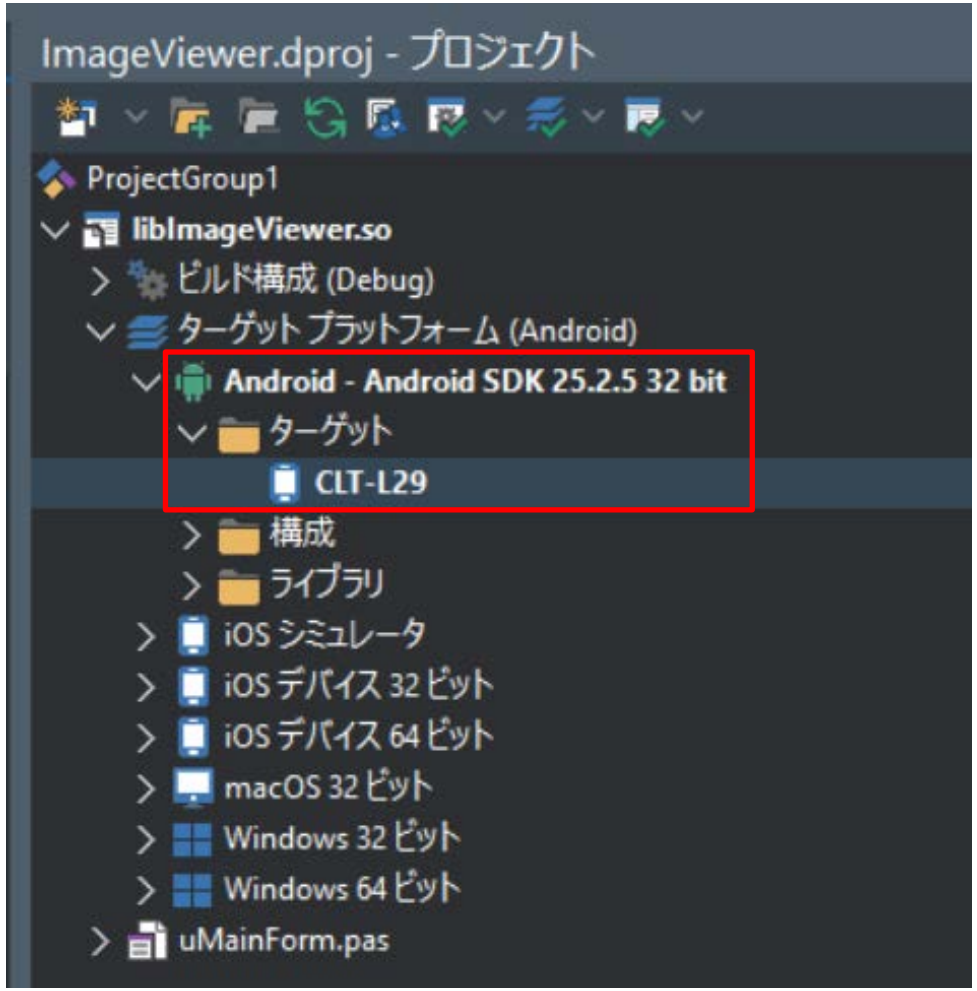
ImageViewer の作成

- イベントハンドラは次のようにします。

```
procedure TForm1.TakePhotoFromLibraryAction1DidFinishTaking(Image: TBitmap);
begin
  ImageControl1.Bitmap.Assign(Image);
end;
```

ImageViewer の作成

- プロジェクトマネージャでターゲットをダブルクリックして設定後、実行（F9 か Shift+F9）します。



ImageViewer の作成

- ボタンを押して画像を選ぶと…

エラー表示も
無く落ちる！

ImageViewer の作成

- Android 6.0 から実行時に権限を取得しなければならなくなりました。
- Delphi 10.3 Rio であれば、今までと比べて簡単に取得できます。
  - 新しく追加された TPermissionsService を使います。

```
unit uMainForm;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Layouts,
  System.Actions, FMX.ActnList, FMX.StdActns, FMX.MediaLibrary.Actions,
  FMX.Controls.Presentation, FMX.StdCtrls, FMX.Objects, System.Permissions;
```

interface 部の uses に
System.Permissions ユニットを追加

ImageViewer の作成

```
type
  TForm1 = class(TForm)
    layoutRoot: TLayout;
    StyleBook1: TStyleBook;
    ImageControl1: TImageControl;
    Layout1: TLayout;
    Button1: TButton;
    ActionList1: TActionList;
    TakePhotoFromLibraryAction1: TTakePhotoFromLibraryAction;
    procedure TakePhotoFromLibraryAction1DidFinishTaking(Image: TBitmap);
    procedure FormCreate(Sender: TObject);
  private
    procedure RequestPermissionResult(
      Sender: TObject;
      const APermissions: TArray<string>;
      const AGrantResults: TArray<TPermissionStatus>);
  public
  end;
```

結果を受け取るイベントハンドラを追加

ImageViewer の作成

```
uses
  FMX.DialogService;

{$R *.fmx}

procedure TForm1.FormCreate(Sender: TObject);
begin
  TPermissionsService.DefaultService.RequestPermissions(
    [
      'android.permission.READ_EXTERNAL_STORAGE' // 権限文字列
    ],
    RequestPermissionResult
  );
end;
```

FormCreate で権限を要求

権限文字列は、Androidapi.JNI.Os, Androidapi.Helpers ユニットを uses して
JStringToString(TJManifest_permission.JavaClass.READ_EXTERNAL_STORAGE)
とすることもできます。
その場合、{$IFDEF ANDROID} などでプラットフォームに依存しないようにします

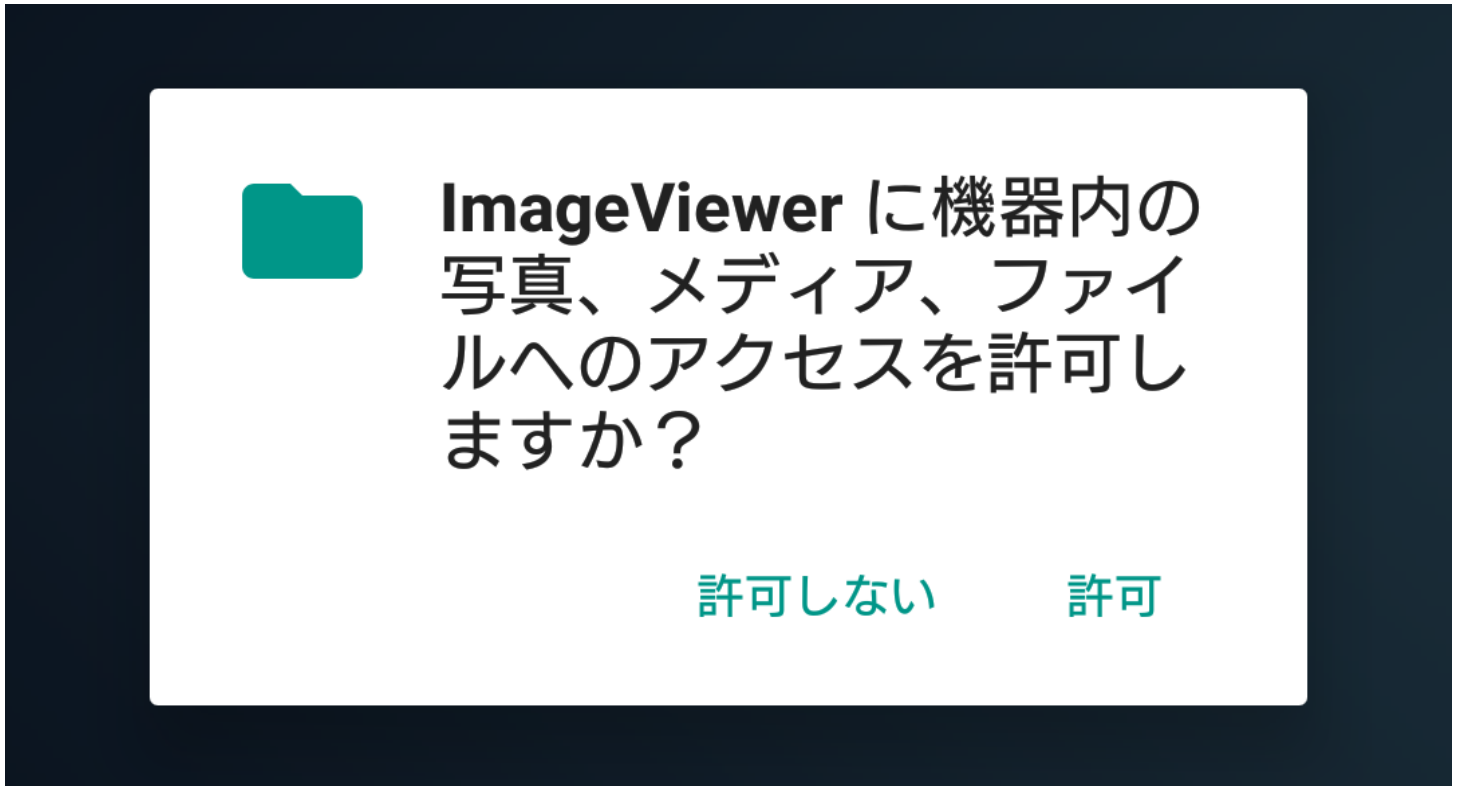
ImageViewer の作成

```
procedure TForm1.RequestPermissionResult(
  Sender: TObject;
  const APermissions: TArray<string>;
  const AGrantResults: TArray<TPermissionStatus>);
begin
  if AGrantResults[0] <> TPermissionStatus.Granted then
  begin
    TDialogService.MessageDialog(
      '権限がないため終了します',
      TMsgDlgType.mtError,
      [TMsgDlgBtn.mbOK],
      TMsgDlgBtn.mbOK,
      0,
      procedure(const AResult: TModalResult)
      begin
        Application.Terminate;
      end
    );
  end;
end;
```

権限を要求した結果
貰えなかったら終了

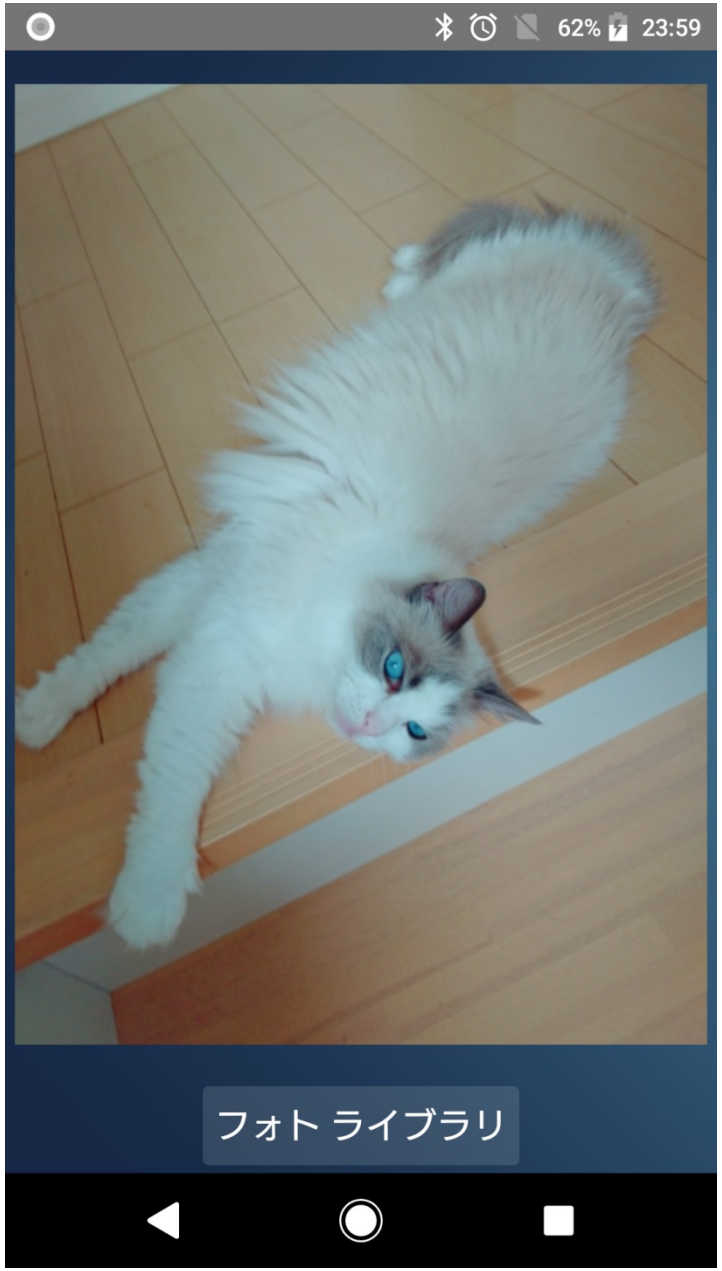
ImageViewer の作成

- これで再度実行すると…権限取得ダイアログが表示されます。



ImageViewer の作成

- 許可すると無事に起動して、画像を取得できました。



まとめ

embarcadero DEVELOPER CAMP

Delphi 10.3 Rio で加速されたモバイルアプリ開発をやるなら？

THANKS!

www.embarcadero.com/jp

第36回 エンバカデロ・デベロッパーキャンプ