2021年11月16日現在

サポート情報	たった2日でマスターするiPhoneアプリ開発集中講座 Xcode 12 Swift 5.3 対応

本書の出版時は、Xcode12.3 での動作確認を行っていますが、Xcode・Swift・iOS のバージョンアップに伴い、本書の記載内容やサンプルアプリに変更が必要なときがあります。
変更が必要な箇所を、このサポート情報にてご連絡いたします。

ご使用されている **Xcode** のバージョンをご確認の上で、このサポート情報を利用してください。

また、正誤も合わせて記載させて頂きます。

■変更内容：

なし

■正誤表：

本書において下記のとおり、誤りがございました。
内容を訂正すると共に、読者の皆様にご迷惑をおかけしたことを、深くお詫び申し上げます。
恐れ入りますが、本正誤表をご確認の上、ご利用いただきますようお願い申し上げます。
赤字が修正箇所になります。

P43

【誤】	実際の利用方法は、サンプリアプリの開発を通して繰り返し操作して慣れていきますので、今は、こんな画面の区切りがあるという認識をしていただければ大丈夫です。
【正】	実際の利用方法は、サンプルアプリの開発を通して繰り返し操作して慣れていきますので、今は、こんな画面の区切りがあるという認識をしていただければ大丈夫です。

P185

【誤】	図「ギターを鳴らすコード」で、39行目、40行目のコードに誤りがあります。 **self.**guitarPlayer = try AVAudioPlayer(data: **self.**guitarData) **self.**guitarPlayer.play()
【正】	**self**は省略できるため、不要です。 guitarPlayer = try AVAudioPlayer(data: guitarData) guitarPlayer.play()

P264

【誤】	図「画面遷移を実装」で、15行目のコメント「// VSaack ここまで」に誤りがあります。
【正】	「// VSaack ここまで」ではなく「**// VStack** ここまで」が正しい

P267

【誤】

図「[▼ スタート]ボタンの色の定義の追加」で、❶の位置に誤りがあります。

「スタート」ボタンで使用する色の定義を追加します。この後、「スタート」ボタンを配置するときに利用します。

❶［＋］を選択し、❷［Color Set］を選択します。

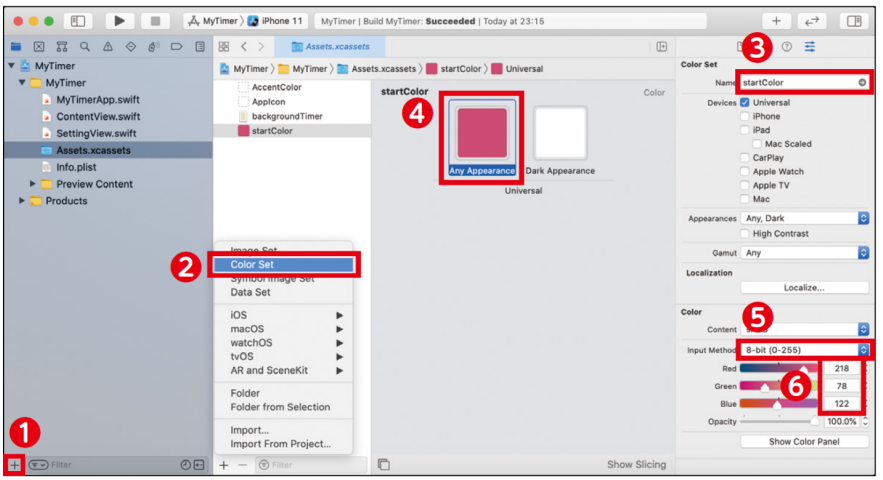
❸［Name］を「startColor」と入力します。

❹［Any Appearance］を選択します。［Any Appearance］で定義した色がライトモードで使われます。

❺［Input Method］のリストボックスから「8-bit (0-255)」を選択します。

❻［Red］に「218」、［Green］に「78」、［Blue］に「122」と入力します。

▼「スタート」ボタンの色の定義の追加



【正】

「スタート」ボタンで使用する色の定義を追加します。この後、「スタート」ボタンを配置するときに利用します。

❶［＋］を選択し、❷［Color Set］を選択します。

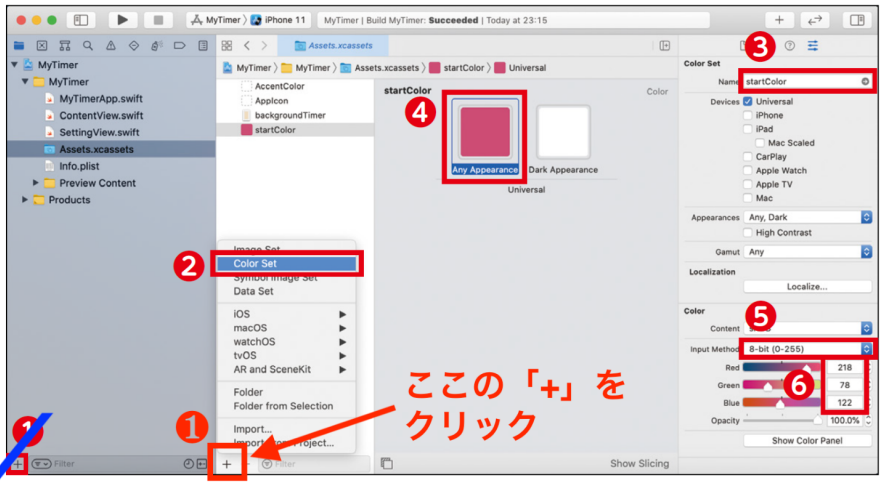
❸［Name］を「startColor」と入力します。

❹［Any Appearance］を選択します。［Any Appearance］で定義した色がライトモードで使われます。

❺［Input Method］のリストボックスから「8-bit (0-255)」を選択します。

❻［Red］に「218」、［Green］に「78」、［Blue］に「122」と入力します。

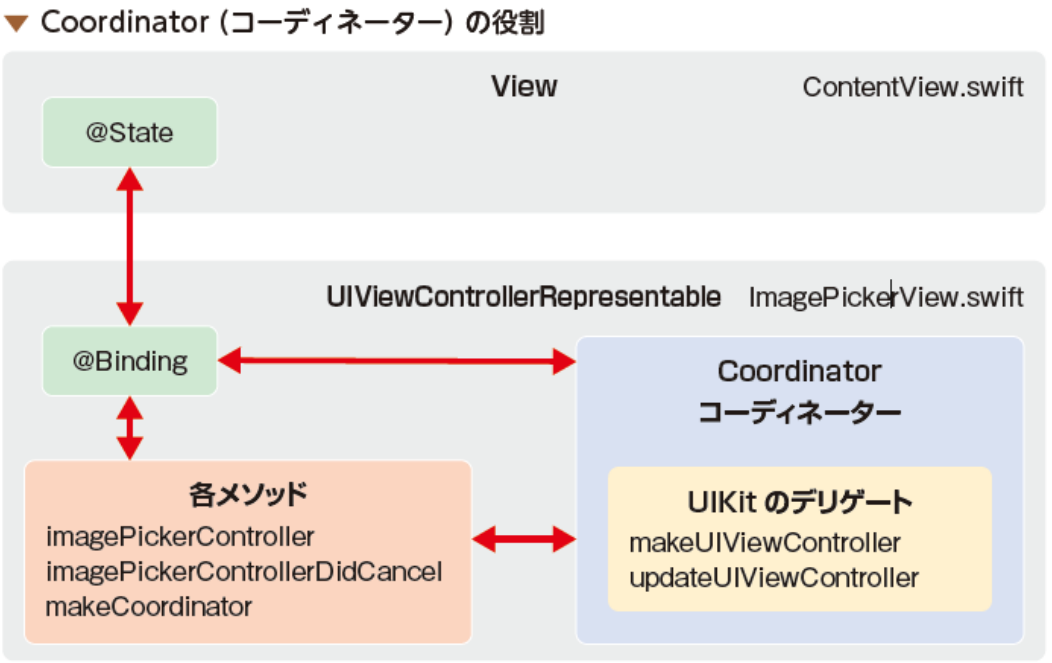
▼「スタート」ボタンの色の定義の追加



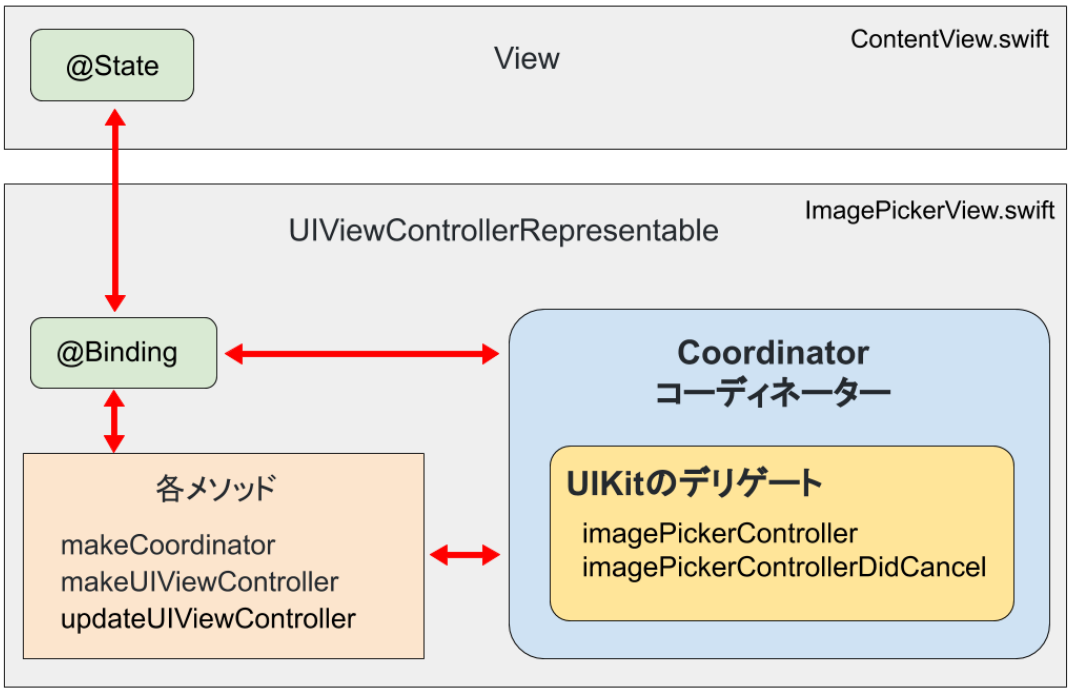
P311

【誤】

図「Coordinator（コーディネーター）の役割」で、makeUIViewController、updateUIViewControllerの配置に誤りがあります。



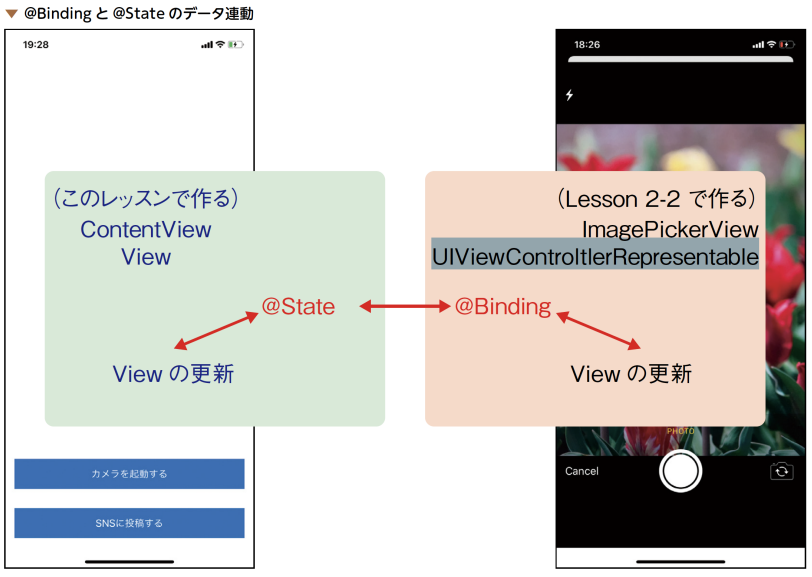
【正】



P311

【誤】	「let parent」の解説に誤りがあります。 ``` // ImagePickerView型の変数を用意 let parent: ImagePickerView ``` Coordinator クラスのインスタンス生成（最初に使用する）のときに、Coordinator を利用する「親」を指定します。Coordinator の中で「親」の値を直接に編集できるようにすることで利用しやすくします。ここでの「親」は「struct ImagePickerView {}」のことです。 そのため、「親」を格納する変数 parent には、構造体 ImagePickerView のデータ型を指定します。
【正】	「let parent」は変数ではなく、定数が正しいです。 定数 ``` // ImagePickerView型の変数を用意 let parent: ImagePickerView ``` Coordinator クラスのインスタンス生成（最初に使用する）のときに、Coordinator を利用する「親」を指定します。Coordinator の中で「親」の値を直接に編集できるようにすることで利用しやすくします。ここでの「親」は「struct ImagePickerView {}」のことです。 そのため、「親」を格納する~~変数~~ parent には、構造体 ImagePickerView のデータ型を指定します。 定数

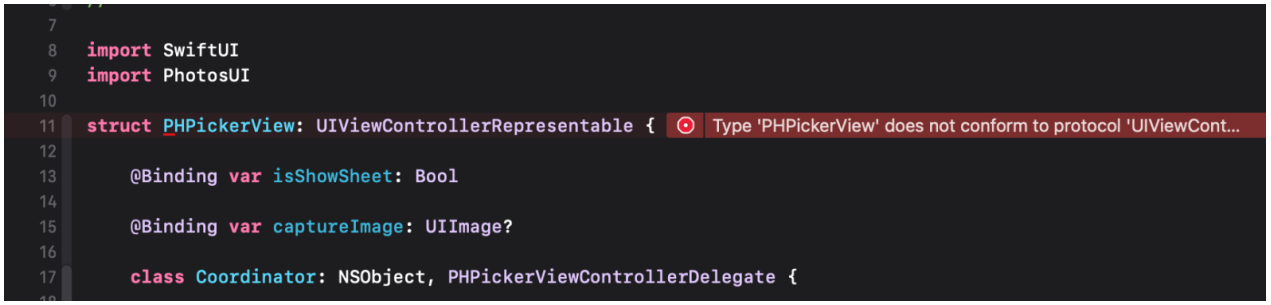
P332

【誤】	図「@Binding と@Stateのデータ連動」で、UIViewControltlerRepresentableの記述に誤りがあります。 
【正】	UIViewControltlerRepresentableではなく**UIViewControllerRepresentable**が正しい

P336

【誤】	図「カメラ使用許可の確認」のタイトルに誤りがあります。 
【正】	「カメラ使用許可の確認」ではなく「フォトライブラリー使用許可の確認」が正しい

P351

【誤】	UIViewControllerRepresentableプロトコルを追加するとエラーが表示される。 
【正】	UIViewControllerRepresentableプロトコルの追加だけでは不完全であるため、環境によってはエラーが表示されることがあります。このエラーは、P356のコードを入力すると解消されますので、エラーが表示されたとしても、そのまま学習を継続してください。

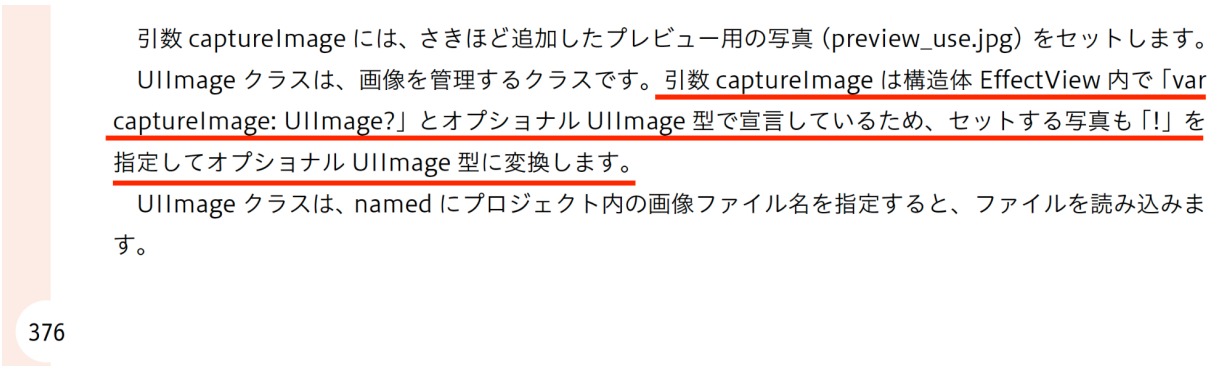
P363

【誤】	ActionSheetの「self.isPhotolibrary」での「self」は不要です。 actionSheet モディファイアは、状態変数 isShowAction が「true」になったときに実行します。 `ActionSheet(title: Text("確認"), message: Text("選択してください"), buttons: [` `  .default(Text("カメラ"), action: {` `  // カメラを選択` `  self.isPhotolibrary = false` `  }),` `  .default(Text("フォトライブラリー"), action: {` `  // フォトライブラリーを選択` `  self.isPhotolibrary = true` `  }),` `  // キャンセル` `  .cancel(),` `]) // ActionSheet ここまで` ActionSheet 構造体は、表示するタイトル、メッセージ、ボタンメニューを定義します。ここでは「カ
【正】	「self.isPhotolibrary = false」ではなく「**isPhotolibrary = false**」が正しい 「self.isPhotolibrary = true」ではなく「**isPhotolibrary = true**」が正しい

P370

【誤】	図「状態変数の追加」の20行目のコードに誤りがあります。 // 表示有無を管理する状態変数 @State var isShowAcitivity = false
【正】	@State var **isShowActivity** = false

P376

【誤】	次の赤線に誤りがあります。  引数 captureImage には、さきほど追加したプレビュー用の写真（preview_use.jpg）をセットします。 UIImage クラスは、画像を管理するクラスです。引数 captureImage は構造体 EffectView 内で「var captureImage: UIImage?」とオプショナル UIImage 型で宣言しているため、セットする写真も「!」を指定してオプショナル UIImage 型に変換します。 UIImage クラスは、named にプロジェクト内の画像ファイル名を指定すると、ファイルを読み込みます。 376
【正】	正しくは、以下の解説になります。 引数captureImageは構造体EffectView内で「var captureImage: UIImage」とUIImage型で宣言しています。UIImage(named:)は、画像がないときにはnilが設定されるオプショナル型であるため、「!」（強制アンラップ）を指定してUIImage型に変換して利用します。

P378

【誤】	表「Core Image でよく使用するクラス」のCIImageクラスの解説に誤りがあります。 画像に関連するデータを管理するためのクラス。Core Image の各クラス（CIFilter、CIContext など）では、CIImage クラスで生成されたCIImage ブジェクトを使用
【正】	「CIImage ブジェクト」ではなく「**CIImage オブジェクト**」が正しい

P401

【誤】	Pointのコードコメント解説に誤りがあります。 a += 1　// a = a + 1　1 を加算 a -= 1　// a = a - 1　1 を減算 a *= 1　// a = a * 1　1 を乗算 a /= 1　// a = a / 1　**1** を除算
【正】	a /= 1　// a = a / 1　**1** で除算

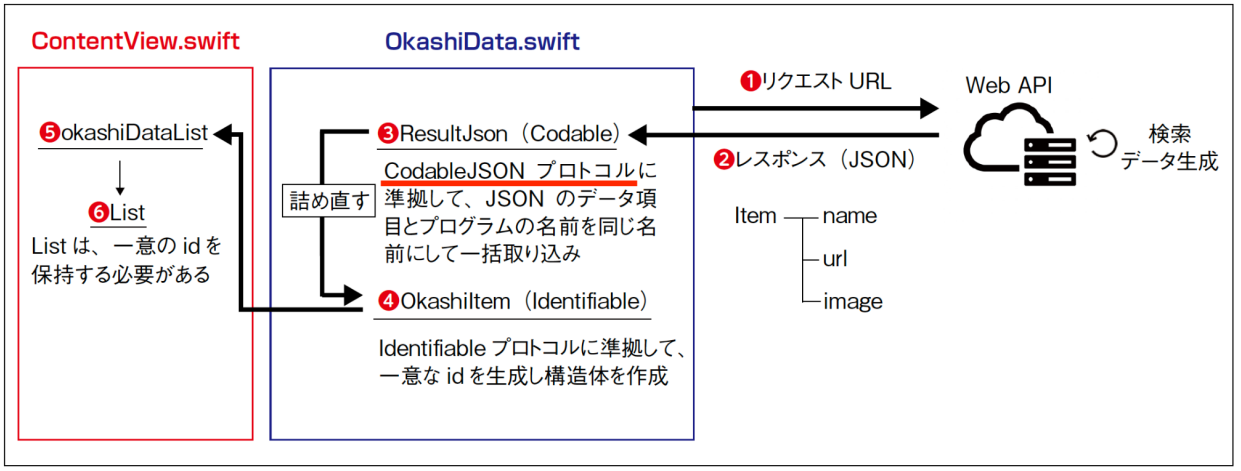
P437

【誤】	本文の「データ取得が完了したタイミンで、セッションを終了します。」に誤りがあります。 ``` // セッションを終了 session.finishTasksAndInvalidate() ``` データ取得が完了したタイミンで、セッションを終了します。 ``` // JSONDecoderのインスタンス取得 let decoder = JSONDecoder() ```
【正】	データ取得が完了したタイミングで、セッションを終了します。

P437

【誤】	「変数json」の解説に誤りがあります。 decoder.decode で、取得した JSON データ（data!）をパースして構造体 ResultJson のデータ構造に合わせて、変数 json に格納します。 この 1 行で取得したデータが、変数 json に格納されます。この仕組みは前述の Codable を活用しています。
【正】	「変数json」ではなく「定数json」が正しい。 decoder.decode で、取得した JSON データ（data!）をパースして構造体 ResultJson のデータ構造に合わせて、~~変数~~定数 json に格納します。 この 1 行で取得したデータが、~~変数~~定数 json に格納されます。この仕組みは前述の Codable を活用しています。

P443

【誤】	図「データ変換ロジックのイメージ」で、「CodableJSONプロトコル」の記述に誤りがあります。 ▼ データ変換ロジックのイメージ 
【正】	「Codable**JSON**プロトコル」ではなく「**Codableプロトコル**」が正しいです。

以上