画面1を開いて、アプリバーが赤で、画面1という名前があり、画面2に進むと言うボタンがあることを確認できます。したがって、画面2はほとんど同じ話ですが、青があります。アプリバーには、画面1に戻るというボタンがあります。

したがって、先に進んでアプリを実行すると、約束のScreen 1が非常にシンプルであることがわかります。アプリバーとボタンがあります。

このボタンをクリックして画面2に移動した時点では、何も起こりません。そして、onPressedではそのコールバックで完全に空であるため、これは予想どおりです。

それでは、画面2へのルートを作成して、そのボタンを押したときに表示されるようにする方法を見てみましょう。

Flutterは、コースで見てきたこれらのクックブックを作成する上で本当に良い仕事をしました。

そして、これらの1つは、新しい画面へのナビゲーションを作成する方法と戻る方法を正確に示しています。

そのため、画面1と画面2の2つのルートを既に作成しました。

ここで必要なのは、navigator.pushと呼ばれるものを使用して、ルートのスタックにルートを追加することだけです。

そして、私たちを押したのは、パンケーキをスタックに正しく押し込んだときと非常に似ていますか？

それらを1つずつ追加するだけです。

一番上にあるスタックで、パンケーキを食べたい、またはパンケーキから飛び出したいと決めたとき、それが一番上になります。

したがって、これはアプリで画面を整理する方法と非常に似ています。

それらはそれぞれ互いに重なり合っており、実際に背後にある画面が表示されるのは、一番上のものが外れたときだけです。

それでは、このパンケーキ操作をコードで試してみましょう。ここで言うように、navigator.pushメソッドを使用する必要があり、2つのことを渡す必要があります。コンテキストと、マテリアルページルートを構築し、行き先を伝える必要があります。これで、コンテキストはビルドコンテキストを参照します。

これは少し前に、現在のスライダーテーマをコピーしようとしたときに見ました。そしてそれを行うために、現在のコンテキストのスライダーテーマを教えてくれました。

これが現在のビルドコンテキストであり、すべてのウィジェットに1つがあります。

これは、この特定のウィジェットがウィジェットツリーのどこに存在するかを把握するための単なる方法です。そして、コンテキストを渡すことで、Flutterが私たちがどこにいて、どこに行く必要があるかを把握するのに役立ちます。

それでは、画面2を画面1にプッシュしてみましょう。

したがって、onPressed内のscreen 1というファイル内で、navigatorを呼び出します。

また、すべてのステートレスウィジェットには1つあることを忘れないでください。

足場が独自のウィジェット内にある限り、navigator.pushを呼び出すことができ、コンテキストはスクリーン1ウィジェットの現在のビルドコンテキストになります。これは、ウィジェットツリー全体におけるウィジェットの現在の場所です。 。

したがって、これは小さなウィジェットツリーであり、アプリの実行時に大きなウィジェットツリーに埋め込まれ、そのコンテキストによってその場所が決定されます。

それが最初に必要なことです。

次に必要なのはルートです。そして、そのマテリアルページルートを使用して、2番目のルートを構築します。

MaterialPageRouteを渡して、ビルダーが機能を取得するようにします。

したがって、コンテキストと呼ばれる入力を持つ関数になり、次に、ナビゲートしたいものを何でも返すボディを持ちます。

この場合、これは新しいスクリーン2オブジェクトです。また、セミコロンでそれを締めくくり、すべてのコンマを追加して、Dartがコードを再フォーマットする必要があるようにします。

現時点では、画面2があることがはっきりとわかりますが、画面2が何であるかはわかりません。これは、screen2.dartファイルをまだインポートしていないためです。

保存ボタンを押してアプリをチェックアウトすると、このボタンをクリックすると画面2に移動します。

そして、例えばiOSでは新しい画面が上に来ると、通常右から来て、この種のアニメーションでは上に押し出されることを知るのは十分賢明です。

したがって、このマテリアルページルートを使用することにより、実際に必要なアニメーションの種類を自動的に決定するのに役立ちます。

したがって、Flutterインスペクターに移動し、ここでこのボタンをクリックすると、プラットフォームが切り替わります。

そのため、iOSシミュレーターで実行しているにもかかわらず、Androidのような画面が表示されます。これは、タイトルがアプリバーの左隅にあるためです。

そして、ボタンをクリックして画面2に移動すると、実際に下部からポップアップされます。これはiOSとAndroidの違いです。

そして、それはすべてFlutterチームによってプリベークされており、あなたがしなければならないことは、このマテリアルページルートビルダーを使用することだけです。

必要に応じて、プラットフォームをiOSに戻すか、Androidに保持します。

そして、このボタンを押して画面1に戻る方法を試してみましょう。前に言ったように、プッシュすることもポップすることもできます。したがって、ポップは、トップ画面を破棄して、下の画面に戻るだけなので、はるかに簡単です。 screen2.dart内で、[画面1に戻る]ボタンを押すと、Navigator.popを書き込むだけで、現在の画面がポップされて破棄されます。

それでは、保存して、チェックしてみましょう。画面1に戻ります。

それがまさにそれです。

そして、必要に応じて1〜2の間で移動できます。とてもクールです。

しかし、多くのルートを持つより複雑なアプリがある場合はどうでしょうか？

たとえば、スクリーン0には2つのボタンがあり、1つはスクリーン1に移動し、2番目のボタンはスクリーン2に移動します。

名前付きルートを使用すると、非常に簡単に実現できます。クックブックでも、これを行う方法を説明しています。

名前付きルートでナビゲートするものに行きましょう。

そして、すでに2つの画面を作成しました。次に、ルートを定義する必要があります。

そして、マテリアルアプリを作成する場所でそれを行います。

この場合、main.dartファイルに含まれます。

それで、私たちの家や出発点を画面1にする代わりに、画面0に変更して、どのように見えるかを示しましょう。

もちろん、screen0.dartをインポートするには使用する必要があります。

これが正しいscreen0.dartで、保存をクリックします。

そして、最初に戻りたいので、ホットリスタートをクリックします。

したがって、main.dartに戻り、スクリーン0に進みます。

したがって、画面0には2つのボタンがあり、画面1に移動するか、画面2に移動します。

そして、それらをクリックすると、正しいルートに移動するはずです。

これを実装するにはどうすればよいですか？

マテリアルアプリには、routesというプロパティもあり、これにより、マップと呼ばれるものを定義できます。

そのため、マップは他のプログラミング言語の辞書に似ており、基本的にはコレクション型です。リストがアイテムのコレクションであるように、マップもコレクションです。

ただし、マップとリストの違いは、マップにはキーと値があることです。辞書に用語と定義があるように、マップにも同じことがあります。

したがって、次のレッスンで詳しく説明するこのマップコンストラクトを使用してルートを定義できます。

そのため、実際に動作を確認しますが、Dartマップの理解を深めたい場合は、このレッスンを続けてください。

そして、次のレッスンに進んだら、それをさらに詳しく説明します。しかし、それは非常に使いやすいです

それを見るだけで、おそらくそれがどのように機能するかをすでに理解しているでしょう。

ルートの定義方法を見てみましょう。

キーと値を持つマップを作成できます。

つまり、/であるルートが要求された場合、最初の画面を構築するか、/ secondと呼ばれるルートが要求された場合、2番目の画面を構築する必要があります。

これまでのところとても簡単です。

それをアプリに定義してみましょう。

先に進んでマップを作成しましょう。中括弧を開いてEnterキーを押すだけでマップを作成できます。そのため、ここの内側はマップ上にあり、いくつかのキーと値があり、キーと値が一致します。

そのため、キーはルートの名前です。

したがって、「/」だけのルートを作成しましょう。

そして、そのルートの値を指定する呼び出しを追加すると、すべてのルートが関数をマップの値として期待していることがわかります。

したがって、文字列はになり、この場合はルートの名前になります。そして、値はコンテキストを入力として受け取り、ウィジェットを返す関数になります。

したがって、この場合、入力としてコンテキストを取り、このルートが要求されたときに表示したいウィジェットになるウィジェットを返します。

したがって、必要なのは、ルートが/の場合のscreen0になります。

これで、先に進み、さらにいくつかのルートを定義できます。 / firstを持ちましょう。

そしてこれが鍵です。

そのため、この場合の値は、おそらくスクリーン1を返す別のコンテキストになります。

したがって、/の名前をスクリーン0に移動し、/ firstをスクリーン1に移動しました。

ここで必要なのは、最後の1つ/ secondを追加することです。

そして、これはもちろん2番目の画面、つまり画面2に移動するはずです。

そして、このscreen2.dartファイルについて伝えるために、これをインポートに追加する必要があります。

名前付きルートをすべて作成しました。

先に進んで使用しましょう。

通常、マテリアルアプリのホームプロパティを使用して、アプリを開始する場所、最初に表示する画面を定義していることがわかります。

ただし、これの代わりにinitialRouteというプロパティを使用することもできます。

また、定義したすべてのルートのうち、どのルートを使用するかを伝える文字列を指定するだけです。

そこで、/という文字列だけを指定して、スクリーン0から開始するように指示しましょう。

そのため、先に進んでアプリをホットリスタートすると、画面0に移動するため、まったく同じように見えるはずです。

ただし、最初のルートがあり、ホームプロパティも定義されている場合、スクリーン0である前のものを追加して、これら2つが互いに競合するため、注意が必要です。

そして、今すぐ保存を押すと、アプリがクラッシュすることがわかります。

そして、それらのうちの1つだけが存在できることを教えてくれます。

したがって、開始と開始の両方を定義することはできません。これはコードでは意味がありません。

最初のルートを使用しているときはいつでもホームを削除しましょう。

また、必要に応じて、最初のルートの代わりにホームを使用することもできますが、すべてのルートが既に定義されているため、この方法で行うのは理にかなっています。

ずっと簡単です。

それでは、screen0.dart内でこれらのルートを使用して、スクリーン1に移動する場所を試してみましょう。

Navigator.pushと言うだけで名前のないルートを使用してこれを実行できますが、ルートをプッシュして作成する代わりに、名前付きルートをプッシュできます。

したがって、この場合、アプリ内の現在のコンテキストがありますが、ここで作成した文字列だけであるルート名も定義します。したがって、最初の画面に移動したい場合、およびこの2番目の画面ボタンをクリックしたときに2番目の画面に移動したい場合は、単に/ firstと書くことができます。ここで作成したルート。

そのため、/ firstに一致するキーを調べて、値から何をすべきかを判断します。

したがって、保存をクリックしてアプリをチェックアウトすると、画面1に移動をクリックすると画面1に移動します。

画面2に移動をクリックすると、画面2に移動します。

これで、ルートを最初から作成するナビゲーターを使用するすべてのボタンが引き続き機能します。

したがって、画面1から画面2に移動できます。しかし、スタックには3つの画面があります。0、1、および2がすべて重なり合っています。

そして、[戻る]ボタンをクリックし続けると、最初まで戻ります。ナビゲーターは、画面間を移動したり、ルート間を移動したりするための非常に簡単な方法です。そして、アプリがより複雑になると、おそらくより多くのルートが必要になり、それらの間をナビゲートする複雑な方法が必要になります。

したがって、2つまたは3つの画面しかない非常にシンプルなアプリがある場合は、おそらく必要な時点でルートを作成すれば十分でしょう。

ただし、4つ以上の画面を持つ複雑なアプリの場合は、最初にすべてのルートをリストし、ナビゲーションコードを実際の画面でより表現力豊かで簡単に記述できるようにします。

ルートとナビゲーションについてすべて学習したので、何を推測したら、新しいページを作成し、そこに移動する必要があります。

したがって、libフォルダーを右クリックして、results\_pageという新しいDartファイルを作成し、[OK]をクリックして新しいresults\_page.dartを作成します。マテリアルパッケージをインポートし、ここで非常にシンプルなステートレスウィジェットを作成します。

そのため、ResultsPageという名前になります。ここでは、単にBMI CALCULATORというタイトルを持つアプリバーを含むスキャフォールドを返します。

そして、こんにちは、またはあなたが望むものは何でも言う少しのテキストを持っているつもりであるボディも持っているでしょう。

この新しいページに移動したときに、実際にコードが正しく設定されていることを確認できるようにするためです。結果ページができたので、入力ページに戻り、コンテナがある最後の部分で、テキストを少し表示します。したがって、テキストウィジェットである子を追加し、CALCULATEと言います。コンテナーがタップを受信するには、コンテナーをジェスチャー検出器でラップする必要があります。

そして、それがタップされたら、先に進んでルートを作成します。

今、挑戦はあなたの法廷にあります。ユーザーが計算ボタンをクリックすると、すべての入力がある最初の画面から2番目の画面にユーザーを誘導できるように、ナビゲーションへのルートについて学習したことを実装します。ビデオを一時停止し、チャレンジを完了してみてください。

大丈夫。

したがって、この場合、2つの画面間を移動するだけなので、おそらく名前付きルートを作成しません。ただし、代わりに信頼できるNavigator.pushメソッドを使用できます。そして、現在のコンテキストとしてコンテキストを取得し、ルートはマテリアルページルートになります。

そして、それを構築するために、入力としてコンテキストを取り、それを返す関数を与える必要があります...

そのため、中括弧のセットを使用してreturnを記述するか、コードが1行しかない場合の簡単な方法は太い矢印であることに注意してください。したがって、等号は山かっこです。そして、ResultsPageを返すことができます。また、まだインポートされていないため表示されません。

それを上部にインポートして、results\_page.dartをインポートします。

保存ボタンを押してアプリを実行すると、このピンク色の領域内にある計算ボタンをクリックすると、helloという単語とアプリバーのある2番目の画面が表示されます。 BMI CALCUATOR。

また、これは、アプリのテーマを定義するために行ったハードワークのすべてが、同じ原色と同じ足場の背景色で、2番目の画面にも引き継がれることを指摘する良い点です。 テキストの色。 ページ間を移動できるようになったので、あとはスタイリングをもう少し更新し、実際にユーザーのBMIを計算して画面に表示するだけです。

しかし、約束されたように、次のレッスンでは、Dartマップの詳細を調べて、namedRouteで表示されているこれらのすべてのマップと、これらのキーと値のペアがどのように機能するかについてもう少し学ぶことができます その中で。

マップまたはディクショナリとそれらがどのように見え、Dartでどのように機能するかにすでに慣れている場合は、次のレッスンをスキップして、その後レッスンに進んでアプリを作成し続けてください。

しかし、いずれにせよ、次のレッスンでお会いしましょう。