過去のいくつかのレッスンでは、すでにほとんどのユーザーインターフェイスをセットアップしました。バックグラウンドレイアウトがあり、列構造のセットアップのほとんどすべてがあり、リストビューもあります。しかし、モックアップの2番目の部分を見る必要があり、ユーザーがリストタスクを追加できるように、下から表示されるこの小さなポップアップを作成する必要があります。

Flutterのドキュメントに目を向けると、一番下のウィジェットインデックスを見ると、スクロールでき、ボトムシートと呼ばれるものに出くわすかもしれません。

したがって、下のシートが下から上にスライドして、より多くのコンテンツを表示します。

それは、まさに私たちが望むものとまったく同じように聞こえますか？

それをクリックして、もう少し読みましょう。

また、ボトムシートには2種類あることがわかります。1つは永続的なもので、プライマリコンテンツを補足する情報を表示するものです。

少し余分なコンテンツがありますが、ユーザーがアプリの他の部分と対話しても表示されたままです。

だからそれはおそらく私たちが望んでいるものではないでしょう。

ずっとそこにいてほしくありませんよね？

代わりに、モーダルボトムシートが必要です。これは、ユーザーがアプリの残りの部分と対話できないようにするものです。強調表示され、ユーザーが入力できる場所にポップアップが表示されるためです。新しいタスクを実行し、完了したらすぐにポップアップ表示され、メインユーザーインターフェイスに戻ります。したがって、これらのモーダルボトムシートを作成する方法は、showModalBottomSheet関数と呼ばれるものを使用することです。

それをクリックして、もう少し詳しく見てみましょう。

この関数には2つの必須プロパティがあることがわかります。

テーマ内のナビゲーターをルックアップして、アプリの他の部分との整合性を保つためにこのボトムシートをどのようにレイアウトする必要があるかを確認できる現在のコンテキストを取り込む必要があります。また、ビルダー、つまりウィジェットビルダーも必要です。ウィジェットビルダーは、現在のコンテキストを取得してウィジェットを返すだけの関数です。

この場合、ボトムシートウィジェットが返されます。

それでは、どのように実装できるのか見てみましょう。

したがって、onPressedに戻ると、これがボトムシートのポップアップをトリガーする部分です。

それでは、showModalBottomSheetと呼びましょう。

ここで、永続的な型を表示するshowBottomSheetがあることに注意する必要がありますが、実際に必要なのは、アニメーションを表示する小さなポップアップです。

したがって、渡すコンテキストはもちろん、タスク画面からのウィジェットの現在のコンテキストです。

ですから、私たちの場合、テーマにはあまりありませんが、実際に渡されるのはそれほど多くありませんが、ナビゲーションスタックはあります。そのため、ナビゲーションスタックのどこにあるかを伝えることができます。

しかし、このビルダーメソッドも用意されているため、このことには関数を渡す必要があることがわかります。

ここで関数を作成しましょう。この関数は、覚えている場合はウィジェットを返さなければならず、入力としてコンテキストを持っている必要があります。

それで、これが実際に私たちがいつも使っているビルド方法に本当に似ていることに注意してください。

したがって、出力はウィジェットです。そして、メソッドにbuildBottomSheetという名前を付けてから、ビルドコンテキスト型の入力を取得し、それをコンテキストと呼ぶことができます。その後、ウィジェットを返します。

したがって、この場合、ボトムシート内に表示するものは何でも返します。

それでは、何を見せたいですか？

本当に単純なことを示すコンテナを返すだけだとしましょう。

そしてもちろん、私とは違って実際にスペルできることを確認してください。WigetではなくWidgetを手に入れてください。

これで、ビルダー内で、作成したばかりのBuild BottomSheetというメソッドを渡すことができます。

そして、先に進んでセミコロンを追加すると、showModalBottomSheetがセットアップされました。

そして実際に見てみると、実際にはかなり短いです。

しかし、今ではボタンを押すと、それが真実の瞬間であり、下からポップアップし、私たちを表示するコンテナウィジェットを構築するボトムシートを取得します。

では、実際にコンテナを持っていることをどのようにして確認できますか？

それでは、テキストを提供しましょう。

「これはボトムシートです」というテキストを中央に配置します。保存してみましょう。

これで、このボトムシート領域にアクセスして、必要なコンテンツを配置できるようになったことがわかります。

そして、それがどのようにユーザーインターフェイスの残りを暗くするかを見ることができますので、残りの部分と対話することはできません。

しかし、クリックするとすぐにアニメーション化し、下にドロップします。そして、アニメーションが上に、アニメーターが下に動きます。

私は一日中これで遊ぶことができましたが、あなたの正気のために私はしません。

これらのボトムシートを作成する方法がわかったので、ダミーのコンテンツを少し削除します。ダミーのコンテンツを削除し、実際のコンテンツを作成しましょう。

この非常に冗長なメソッドをここでどのように使用できるかに注意してください。

buildBottomSheetに名前を付け、戻り値も設定します。

ただし、1つのアイテム、この場合はコンテナーを返すだけの場合、セミコロンを実際に削除し、returnキーワードと中かっこをすべて削除して、矢印関数で置き換えることができることを知っています。

これで、ここに1行のコードができました。

名前付き関数buildBottomSheetを使用する代わりに、これをさらに短くすることができます。単に匿名関数にすることができます。

それでは、先に進んでこの部分をすべて取り、それをカットして、ここでの機能を以前のものに置き換えます。

したがって、名前と戻り値の型を削除できます。もちろん、このコンテキスト入力もここで型を必要としません。

ビルドコンテキストタイプである、ここから渡されるコンテキストから推測できます。

そのため、今では私たちのコードは非常に短く、ほとんど痛いほど短いです。

ただし、機能はまったく同じです。

そして、コンテナを作成してここに巨大なツリーを構築する代わりに、実際にこれを取得して、画面内に新しいファイルを作成します。

それでは、add\_task\_screenという新しいDartファイルを作成しましょう。

そして、この小さなボトムシートの内部に表示するウィジェットツリーを作成します。

そのため、コンテナを返す代わりに、AddTaskScreenを返し、スクリーンフォルダーとadd\_task\_screen.dartファイルをインポートします。

それでは、ここで実際にAddTaskScreenと呼ばれるものを作成し、この画面内に何が欲しいのでしょうか？

モックアップを見てみると、タイトル、つまりテキストウィジェットとテキスト入力、ユーザーが実際のタスクを記述できるもの、さらには追加というフラットなボタンがあります。しかし、このすべてのインターフェイスを作成する前に、1つだけがあります。つまり、モックアップでは、このボトムシートに、アプリのデザインが実際に一貫して見えるようにするためにリストを作成したのと同じように、曲線状のエッジが必要でした。

しかし、現時点では、この非常に正方形に見えるボトムシートがあります。それは非常にクールですが、それでもまだ私たちが望んでいたものではありません。

これを作成する方法を見てみましょう。

さて、私たちはコンテナを手に入れました。私がやろうとしているのは、この背景色とまったく同じ色を与えることです。 Macには、デジタルカラーメーターと呼ばれるものがあります。

そして、それが私にできることは、基本的にそれが私にカラードロッパーを与えることであり、同様のカラードロッパーツールをWindowsにもインストールすることができます。

ここにカーソルを合わせて、この色の値を確認します。それで、現時点では、それは117のようなものだと思いますか？

したがって、赤は117、緑は117、Bは117です。

そこで、これらの値を使用して、クロムに入り、「rgb to hex」を検索します。そして、Chromeのこの小さなカラーピッカーツールを使用して、これらの値を変更できます。したがって、117、117、117がありました。

そのため、この灰色で、＃757575のような16進コードがここにあります。

それでは、これをコードにコピーしてみましょう。

そのため、色は色ウィジェットを使用して構築され、整数値を取ります。

そのため、以前のことを思い出すと、ハッシュタグをポンド記号に置き換えるために0xffを実行する必要があり、その後、色の整数値になります。

したがって、保存ボタンを押すと、このポップアップが背景とほとんど区別できなくなります。

それを最初のコンテナとして使用し、このコンテナ内で別のコンテナを追加します。

さて、これは少し奇妙に思えます。私がやっていることですが、すぐに明らかになります。

このコンテナに、装飾を追加します。

それは箱の装飾になるだろうし、色は白い色になるだろう。

そしてもちろん、境界半径があります。

そのため、左上と右上に境界半径を追加し、両方に20の円形半径を追加できます。

そして今、保存ボタンを押すと、この下のシートがポップアップするときに、実際には背景と一致するグレーの色を持つ下のシートが背面に1つあることがわかります。

そして、コンテナ内に丸いエッジを持つ別のものがあります。

これにより、この小さな湾曲したエッジのボトムシートを作成できます。デザインの残りの部分に収まるので、本当にきれいに見えます。

したがって、モックアップに表示される残りの部分を追加するだけです。

そのため、「タスクの追加」という大きなタイトル、テキストフィールド、フラットボタンがあります。

だから、あなたのための別の課題があります。ビデオを一時停止し、ウィジェットについて学習したことを使用してこの正確なレイアウトを再作成できるかどうかを確認し、Flutterウィジェットを使用してUIをレイアウトします。

そして、完了したら、試してみてから戻って、解決策を紹介します。

今すぐビデオを一時停止します。

大丈夫。

それでどうなったの？

さて、これを行う方法の私のバージョンがあります。

また、Flutterの場合と同じことを行う方法は100万通りあります。

あなたのコードが私のコードと異なるからといって、それが間違っているという意味ではありません。この最終結果が得られれば、目的はほぼ達成されました。コンテナ内

再びすべてが垂直方向にレイアウトされるため、列になる子を追加します。

このコラムでは、最初の子がテキストウィジェットになり、2番目の子がテキストフィールドウィジェットになり、最後の子がフラットボタンウィジェットになります。

それで、3つのことができました。テキストウィジェットに「タスクの追加」という名前を付けます。

そのため、保存をクリックすると、3つのウィジェットが表示されます。

どれも本当に見苦しいですが、今すぐ変更します。

だから、まずテキストウィジェット。スタイルを追加してみましょう。

それでは、スタイルプロパティを追加して、テキストスタイルにし、フォントサイズを変更して、さらに大きくし、30にします。そして、色を変更して、とにかくこのモジュールのお気に入りの色、colors.lightBlueAccent、そして今では少し良く見えます。

それでは先に。

次はテキストフィールドです。テキストフィールドは実際にこの長さ全体を囲んでいます。クリックすると、青い下線が表示され、カーソルが一番左にほとんど見えなくなるからです。

もう1つ変更したいのは、この小さな下の画面をポップアップするときに、このテキストフィールドを自動的に有効にすることです。COMMAND+ Kを押してキーボードを表示するか、Androidで切り替えると、この小さなボタンは、このキーボードがポップアップするたびに自動的に表示されるので、クリックする必要はありません。テキストフィールドが1つしかないため、このテキストフィールドを自動的に有効にして、これをタップする余分な手順を必要とせずにすぐに入力できるようにすることは理にかなっています。これを行うには、オートフォーカスプロパティを追加し、trueに設定します。

そのため、保存ボタンを押して下のシートを開くと、自動的に有効になっていることがわかります。この下の線は青で、キーボードが毎回ポップアップ表示されます。

もう1つやりたいことは、カーソルが画面からほとんど外れているという事実に対処することです。また、テキストのalignプロパティをTextAlign.centerに変更するだけで、中央に移動できます。

追加ボタンを押すと、カーソルが真ん中に表示され、すでに有効になっていて準備ができていることがわかります。

最後の1つはもちろんフラットボタンです。

それでは、テキストウィジェットである子を指定してテキストを追加し、「追加」と読みます。

そして、淡いブルーのアクセントになる色を付けたいと思います。

しかし、保存してホットリロードを押すと、実際には表示されません。

そして、その理由は、フラットボタンが現在、パラメーターonPressedが必要であるという警告を表示しているためです。

また、ボタンにonPressedが指定されていない場合、デフォルトでは無効なボタンになります。

そのため、ボタンは現在無効になっているため、色は表示されません。

それでは先に進み、必要なonPressedを追加します。ここで後日、おそらくいくつかのコードを追加して、タスクをリストに追加します。

しかし、今はそのままにしておきます。しかし、保存ボタンを押して、オンになったらすぐに「押した」ボタンを押して、フラットボタンが無効にならないようにすると、背景が表示されます。そして今、私はそのテキストを修正したい。

それで、スタイルを白に変更してみましょう。そして、テキストのスタイルと色はcolors.whiteです。

さて、保存ボタンをクリックして、最後に気付くのはそのボタンとモックアップの違いです。このボタンは端から端まで行きますか？

または、テキストフィールドとほぼ同じ幅です。

では、どのようにコードでそれを行うのでしょうか？

列の横軸の配置を変更して左右に引き伸ばすことができます。次に、先に進み、テキストウィジェットのtextAlignプロパティを同様に中央に変更します。

ここで、テキストの最初のプロパティが「タスクの追加」であることに注意してください。

そのため、その前に別のプロパティを追加しようとしても、あまり好きではありません。これには、パラメータ名がないためです。テキストコロン付きのテキストウィジェットのようなものはありません...

そのため、位置データとその位置がウィジェットの最初でなければなりません。

だから私たちはそれを最初にしなければなりません。

そして、特別な注意を払ってそれを扱わなければなりません。

これで、ここにあるすべてのものを作成したので、列の周りにパディングを追加するだけです。

それでは、先に進んで、ここで列をラップするパディングをコンテナに追加しましょう。

そして、おそらくすべての辺を単に追加するだけで済ませることができ、古き良き20ピクセルを追加しましょう。これは、ここで見たものとほとんど同じように見えます。

もちろん、このフィールドの間にパディングを追加し、必要に応じてボタンを追加することもできますが、実際にはうまく見える瞬間だと思います。

ただし、もちろん自由に外観を変更できます。

これで、モックアップ全体をほぼ複製できました。

リストを説明する領域が上部にあり、リスト項目を含む実際のリストビューがあり、アニメーション化されたボトムシートがポップアップ表示され、そのスクリーンショー内にすべてが表示されていますアウトも。

これで、次の段階に進む準備がほぼ整いました。次の段階では、どのようにすべてを機能させ、すべてをまとめるかを検討します。

そして、私が最初に対処したいのは、なぜ私たちのチェックボックスが機能しないのですか？

だからそれについて考えて、解決策を見つけるために、

次のレッスンでお会いしましょう。