すべてのカードは同じように見えますが、再利用可能なカードを使用してそれぞれのスタイルを設定し、拡張ウィジェットを使用してレイアウトすることができますが、コンテンツはすべて異なります。通常、Flutterでは、Flutterが作成したウィジェットを使用しているときに、子プロパティを持つことでこれに対処する傾向があります。

そのため、たとえば、展開されたウィジェットには子があり、コンテナには子があります。

これにより、同じウィジェットを再利用できますが、使用するウィジェットごとにカスタムコンテンツを追加できます。

カスタムの再利用可能なもので同じことができるように、先に進んで新しい子プロパティを作成しましょう

カード。まず最初に、カスタムの子プロパティを作成し、cardChildと呼びます。

また、ウィジェットは不変なので、これは最終的なものでなければなりません。

そして、これはウィジェットのタイプを持ち、cardChildと呼ばれます。

新しいプロパティはありますが、最終的な変数car childを初期化する必要があることを示すエラーが表示されるため、ここで初期化する必要があります。

それをここで行います。

今、私は子のプロパティが必要になるとは思わない。

そのため、必要な注釈を追加する必要はありません。必要なのは、this.cardChildを書くだけです。

つまり、新しいreusableCardを作成すると、ウィジェット内に子を追加することができます。

ただし、コンテンツが必要ない場合は、そのままの状態でそのまま使用できます。最後の最後は、この子をウィジェットツリー内のどこに配置するかです。さて、コンテナの子になります。

したがって、コンテナの子プロパティを利用することができ、この再利用可能なカードに渡されるcardChildは、コンテナの子になります、それだけです。

それだけです。

reusableCardを更新したので、実際にコンテンツを追加します。

そして、最初に対処するのは、ユーザーが性別を選択できるようにするこれらのカードです。

フォントの素晴らしいFlutterパッケージに由来するアイコンがあります。

そして、その下に説明として少しのテキストがあります。

それでは、最初の行で再利用可能なカードをカスタマイズし、機能をコーディングする前にまずスタイルを整えることから始めましょう。

まず最初に、素晴らしいフォントからアイコンを取得する必要があります。

font\_awesome\_flutterパッケージが必要になります。

それでは、実際にカスタムパッケージをまったく含んでいないpubspec.yamlにこれを追加してみましょう。

クパチーノのアイコンのすぐ下に、font\_awesome\_flutterを追加します。それをここに貼り付けます。そして、ここにコロンを含める義務的なYAMLマークアップを追加し、キャレットサインアップもここに追加します。

このようにして、8.something.somethingのfont\_awesome\_flutterのメジャーバージョンを使用して満足していることを示しますが、8〜9を超えることはありません。

YAMLを更新したので、packages.getをクリックし、パッケージをインポートページにインポートします。

したがって、フォントawesomeパッケージをインポートし、reusableCard内で使用できるようにします。まず最初に、ここで最初のカードをターゲットにします。これは次のようになります。

したがって、すべてをカードの中央に配置し、素晴らしいフォントのMarsアイコンと以下のテキストを表示します。

したがって、これはおそらく、子として列を使用して簡単に実現できます。

先に作成したcardChildをプロパティとして追加すると、cardChildは2つの子を持つ列になります。そして、これらの2人の子供はまずフォントの素晴らしいアイコンから作成されるアイコンになります。そして、私たちが必要とするものは火星と呼ばれるものであり、これは一般に男性のシンボルです。アイコンが入ったので、

アイコンのサイズを80に増やし、通常よりもずっと大きくします。

そして、アイコンとテキストの間に少しギャップを追加します。それを行うには、通常sizedBoxを使用します。また、カスタムの高さプロパティを提供して、アイコンとテキストの間の距離を指定できます。

そして、この場合、15を選択します。そして最後に、テキストウィジェットを追加し、すべて大文字で「男性」と言うだけです。それから、テキストスタイルウィジェットになるスタイルを少し追加します。フォントサイズを18に増やし、フォントの色をカスタム色に変更するだけです。 0xFFになるため、完全に不透明な8D8E98です。そして、これは一種のこの薄いグレー色であり、これらの色はすべて、カラーピッカーツールを使用するだけで元のDribbbleデザインから来ていることがわかります。すべてのコンマを追加して、保存時にDartがコードを再フォーマットするようにします。この時点でデザインを見ると、フォントの素晴らしいアイコンを実際に引っ張っていないことがわかります。これは、新しいパッケージを追加してpackages.getを実行するたびに、実際にはアプリを停止してゼロから実行することをお勧めします。これにより、コールドリスタートを実行し、必要なものすべて、つまりフォントの素晴らしいアイコンを組み込むことができます。そして今、私たちが望む実際のアイコンを画面上に見ることができますが、この再利用可能なカードの子である私たちのコラムは、あちこちにあります。ただし、以前から覚えている場合、実際に列を親の中央に配置するmainAxisAlignmentと呼ばれる列プロパティを追加できます。

したがって、今すぐ保存を押すと、アイコンとテキストが、再利用可能なカードである親の真ん中に来ることがわかります。

これで、最終設計では、基本的にほとんど同じように見える2枚のカードが作成されることがわかりました。

どちらにもアイコンと少しのテキストがあり、スタイルはまったく同じです。

そのため、アイコンとギャップとテキストウィジェットを作成しているこの部分を実際に再利用して、別のウィジェットとして抽出して、どこでも再利用できるようにすることは理にかなっています。

だからここに挑戦です。

そこで、カスタムアイコンコンテンツウィジェットを作成します。そして、挑戦として、cardC内にあるすべてのものを、2枚のカードにまたがって繰り返す別のウィジェットに抽出してみてください。

したがって、reusableCardに対してどのようにそれを行ったかを思い出して、ビデオを一時停止し、このチャレンジを完了してみてください。

大丈夫。

そのため、これを行うには、抽出するウィジェットの最高レベルを選択します。

したがって、いくつかの子を持つ列があることはわかっていますが、実際にはメインツリーから取り出して別のウィジェットを作成する列です。カーソルが選択されている限り、Flutterアウトラインに移動でき、メインウィジェットツリーで確認できます。対応する列も選択されています。

そして今、私たちがする必要があるのは、それを右クリックして抽出ウィジェットをクリックすることです。

そして今、それに名前を付けて、そのiconContentを呼び出すことができます。これを別のウィジェットにリファクタリングします。

Dartの最新バージョンの時点で前述したように、新しいキーワードは必要ありませんが、アイコンコンテンツウィジェットがあります。

そして、ここに抽出されました。

したがって、次の課題はこのウィジェットを抽出したことですが、2つの場所で再利用できるようにする必要があります。

iconContentにあるものはすでに入手済みですが、2番目のカードでは2番目の場所があります。アイコンの内容は、アイコンと少しのテキストがある列を記述するためです。

しかしもちろん、アイコンとテキストは実際にはこれらの2つのカードで異なります。

そのため、アイコンが何であってもアイコンコンテンツウィジェットを再利用して、これらの各ウィジェットを表示できるようにする必要があります。つまり、アイコンコンテンツウィジェットに現在含まれているコードを少し変更する必要があります。

最終目標は、同じアイコンコンテンツウィジェットを使用して、今作成したカスタムウィジェットを使用して、男性アイコンカードと女性アイコンカードの両方を作成できるようにすることです。

そのため、ビデオを一時停止して、このチャレンジを完了してみてください。そのため、これを実現するために、カスタムreusableCardで行ったのとまったく同じことを行います。つまり、作成するウィジェットで使用できるプロパティを作成します。

そのため、ここではまず、アイコンコンテンツを抽出した場所にFlutterの抽出機能を使用するときに付属したデフォルトコンストラクターを削除し、代わりに、独自のカスタムコンストラクターを作成して、アイコンの値およびテキストのカスタム値。

それではまず、プロパティを作成しましょう。

繰り返しになりますが、最終キーワードがあり、ウィジェットのプロパティを作成するたびにそれが表示されます。

そして、データ型を持つことになりますので、この場合はアイコンデータになります。

さて、それがアイコンデータデータタイプになることをどうやって知るのですか？

さて、いつでもアイコンウィジェットを確認できます。また、iconというプロパティについては、アイコンデータ型を期待していることがわかります。

したがって、アイコンデータを取得し、単にアイコンと呼びます。

これで、文字列になる2番目のプロパティを追加できます。

これは、女性カードを作成しているときに女性を、私たちが女性カードを作成しているときに男性を渡すことができるように、ここで行くラベルを渡すことになります男性カードを作成します。

これらの最終プロパティをまだ初期化していないため、エラーが発生しています。

最終変数には値が1つしかなく、それが変数の作成時点にあることに注意してください。

したがって、このアイコンコンテンツが作成されると、トリガーされる部分はもちろんコンストラクタです。

そこで、コンストラクタを作成します。そして、コンストラクターにはいくつかの名前付きプロパティがあり、this.iconとthis.labelもあります。

これで両方が必要になるようにできましたが、そのままにしておくこともできます。

コンストラクタを作成したので、プロパティを作成しました。

メインのビルドメソッドで、これらのプロパティを配置する場所を実際に指定します。

したがって、カスタムになる最初のことはもちろんアイコンです。

そのため、火星アイコンとしてハードコードされることはもうありません。

また、テキストをlabelプロパティに変更します。

したがって、このプロパティを介して渡されるものはすべてここに移動します。

それは男性と女性になります。

メインウィジェットツリーであるBMI計算機に入ったら、アイコンコンテンツを更新して、フォントAwesomeIcon.marsになるアイコンを含めることができます。

そして、単に「MALE」という単語になるラベルもあります。

そして、これが文字列を期待する理由です。

セーブを押して、2番目のカードについても同じことをしましょう。

したがって、2番目のreusableCardにもcardChildがあり、子はアイコンコンテンツウィジェットであるウィジェットになります。

また、アイコンも表示されますが、今回は、アイコンが金星のシンボル、女性用の金星になり、ラベルが女性になります。涼しい。

したがって、保存ボタンを押してアプリに移動すると、両方のカードが作成され、すべてのスタイルがIconContentから取得されていることがわかります。

そのため、アイコンのサイズ、アイコンとテキストの間のギャップのサイズなど、すべてを再作成する必要はありませんでした。

代わりに、ここではカスタムウィジェットIconContentに依存しています。そして、実際のアイコンや実際のラベルなど、変更が必要な部分にいくつかの異なる値を渡しているだけです。

すぐに使用可能なFlutterウィジェットに依存するのではなく、独自のウィジェットを作成し始めていることがわかります。そして、これらのプロパティとコンストラクターを使用して、作成するウィジェットごとにカスタムコンテンツを指定できます。

この時点で、メインのinput\_page.dartファイルでカスタムウィジェットが多くのスペースを占有し始めていることがわかります。

この場合、実際には、IconContentと再利用可能なCardを2つの別々のDartファイルに分離することをお勧めします。

これは、他のファイル、おそらくアプリの他のページや画面でもこれを使用する可能性があるためです。

したがって、すべてを1つのファイルに詰め込むのは意味がありません。

それでは、libに進み、新しいDartファイルを作成しましょう。そして、最初にicon\_contentを呼び出してから、[はい]をクリックしてGitに追加します。また、reusableCard、reusable\_cardを保持する新しいDartファイルを作成し、Dartファイルも作成します。

それでは、先に進んでIconContentに関連するすべてのものを取得し、input\_page.dartファイルから切り取って、代わりにここに貼り付けてみましょう。

そしてもちろん、マテリアルウィジェットを使用するときは常に、マテリアルパッケージをインポートする必要があります。

また、reusableCardをreusable\_car.dartに移植しますが、material.dartパッケージをインポートする前には移植しません。

これで、reusableCardに貼り付けることができます。

IconContentを見ると、多くのことがハードコーディングされていることがわかりますか？アイコンのサイズ、またはアイコンとテキスト間のギャップのサイズ、またはテキストのスタイルなど。

今では、これらのパーツを単に取り出して、ファイルの先頭に定数として作成することが理にかなっています。

そして、このようにlabelTextStyleを変更したいと思う場合、ここの一番上にそれを見る必要があり、必要に応じてそれを更新および編集できます。

そしてここでは、単にlabelTextStyleを参照しています。そして、そのように多くのラベルがあるか、その特定のスタイルが必要な場合、それらはすべてこの定数を参照できます。

したがって、ここで修正する最後のことは、再利用可能なカードが何であるか、またはこれらのウィジェットがこのファイルから削除されたためにIconContentが何であるかがわからなくなるため、多くのエラーが発生することです。

そのため、これらの他のウィジェットについてファイルに伝えるために、作成したばかりのファイルをインポートする必要があります。

したがって、icon\_content.dartおよびreusable\_card.dart

したがって、すべてを個別のファイルにリファクタリングする必要があり、必要に応じてこれらの各コンポーネントを個別に操作するのがはるかに簡単になります。

ネイティブのFlutterウィジェットが独自のファイルを持つように、たとえばWindowsでCOMMANDまたはCONTROLを押しながらテキストをクリックすると、すべてのコードを含むtext.dartファイルがあることがわかりますテキストウィジェット。

同様に、独自のファイルに独自のカスタムウィジェットがあります。したがって、プロジェクト全体で再利用できます。

大丈夫。

そのため、インポートが少し多くなりすぎて、スペースを取りすぎていることがわかった場合、単にそれを折りたたむことは悪い考えではありません。

コードを作成するときにインポートを参照する必要はありません。そして今、メイン入力ページには、ほとんどがレイアウトでいっぱいのウィジェットツリーがあり、ここでツリーを完全に展開し、100万行の長さを表示するのではなく、一目でそれが何であるかをより簡単に見ることができます。何が起こっているかを一目で理解するのは非常に簡単です。

次のレッスンでは、カードをクリックしたときにカードの色を変更しようとします。現時点では何も起こらないからです。

これらのいずれかを選択できるようには思えません。

そのため、次のレッスンでその問題を解決します。

それで、それ以上のために、あなたに会いましょう。