さて、これまでに見てきたように、ウィジェットにプロパティとして追加することで、ルートを介して転送されるデータを解析できることがわかりました。そのため、ここにプロパティを追加し、finalと呼び、その変数locationWeatherを呼び出すことができます。少し後で、別のソースから天気を取得するためです。

これで、このステートフルウィジェットが初期化されると、ロケーション画面が表示されます。

そしてここにthis.locationWeatherを追加できます。

そのため、ロケーション画面を初期化すると、locationWeatherというプロパティがあり、新しい変数に設定できます。

そして、ここでその気象データを解析します。

新しいロケーション画面を初期化するとすぐに、そのロケーションデータを取得できます。

しかし、自分の場所の状態からどのようにアクセスしますか？これが中心的な問題であり、これはFlutterの仕組みを理解するための基本です。

位置情報画面のテキストウィジェット内に読み込み画面から受信した天気情報を表示する必要があります。

これは、ロケーション画面オブジェクトからロケーション画面の状態オブジェクトにデータを取得することを意味します。

ロケーション画面のステートフルウィジェットは、実際にはその状態とは別のオブジェクトであることに注意してください。また、テキストウィジェットはロケーション画面内に存在しません。

代わりに、テキストウィジェットは状態オブジェクトのビルドメソッドに存在します。

ただし、ロケーション画面オブジェクトとロケーション画面の状態オブジェクトはリンクされています。

つまり、状態オブジェクトは、どのステートフルウィジェットに属しているかを知っています。状態オブジェクトには、親のステートフルウィジェットを指すwidgetというプロパティがあります。

コードでこれがどのように機能するかを示します。

ロケーション画面の状態のウィジェットプロパティを介してコンソールに天気データを印刷します。

先に進んでinit状態を追加し、状態オブジェクトが初期化された瞬間をタップすると、ウィジェットオブジェクトを出力できます。

これで、すべての単一状態オブジェクト内のこのウィジェットオブジェクトにアクセスできます。そしてそれは、現在の構成です。

したがって、状態オブジェクトの構成は、対応するステートフルウィジェットインスタンスです。

したがって、基本的にウィジェットをタップすると、locationWeatherのプロパティを持つロケーション画面ステートフルウィジェットにアクセスできます。したがって、print widget.locationWeatherと言うことができます。そして今、このホットポテトをロード画面からロケーション画面に渡すことができます。そして、ロケーション画面の状態から、ウィジェットを介してロケーション画面をタップし、ここでロケーション天気を介してそのプロパティをタップすることができます。

ここで取得するために2つの解析を行います。

それを印刷して、天気がここから実際に通過したことを確認します。

そのため、ロード画面に印刷ステートメントがありません。コンソールを見ると、ロケーション画面に移動するとすぐに、その場所の天気がコンソールに印刷されています。

確実に機能しています。私はここで間違いなくその場所の天気にアクセスしています。

ですから、その場所の天気を使用して温度を取得し、条件コードを取得し、そこから都市名を取得するだけです。

それで、前から持っていたこれらの3行のコードをコピーし、updateUIと呼ばれる新しいメソッドに貼り付けます。

そのため、これを使用してユーザーインターフェイスを更新します。このupdateUIには、weatherDataと呼ばれる単一の入力があります。また、locationWeatherはこの気象データ変数から派生した変数であることに注意してください。そのため、入力のデータ型を動的として追加できます。

ここで動的であり、ここで動的であり、渡すときにも動的になります。したがって、これらの3行のコードを貼り付け、更新して、デコードされたデータを使用する代わりに、その天気データオブジェクトを利用するための天気データ。

したがって、ウィジェット内でこれらのプロパティを使用する場合は、状態オブジェクト内にいくつかのプロパティを作成する必要があります。

そのため、温度と呼ばれるdoubleと、条件となるintと、cityNameとなる文字列を作成します。

そして今度は、温度をweatherData、条件付きequalweatherDataおよび都市名に等しく設定できます。

それでは、CONTROL +保存を使用してコードを再フォーマットし、updateUIを作成します。この段階では、widget.locationWeatherを印刷する代わりに、それをupdateUIメソッドに渡し、そのwidget.locationWeatherをそのメソッドへの入力として提供できます。

したがって、ロケーション画面が初期化されると、widget.locationWeatherを介してタップできるlocationWeatherを渡し、updateUIを呼び出して、再びその気象データを渡します。そして、温度条件と都市名を取得するためにそれを渡します。この時点で、温度値を印刷しようとすると、もう一度実行してアプリをホットリスタートしてみましょう。印刷される温度はケルビンであることがわかります。したがって、摂氏ではなく華氏でもありません。

ここで、237ポイント1を差し引くことにより、この温度をケルビンから摂氏に変換することができます。

ここで、温度を285.61とし、273を引くと、摂氏12度または華氏54度になります。

しかし、これは非常に一般的な機能であり、Open Weather Mapは実際にその処理方法をすでに知っています。

したがって、APIドキュメントを見て、単位形式を見ると、単位に等しいメトリックを末尾に追加するか、単位が帝国に等しいことで、温度をメトリック値または帝国値として返すことができることがわかります最後まで。

だから私はメートル単位が欲しいので、摂氏として欲しい。

それをコピーし、それを貼り付けてURLに追加します。

APIキーの後の最後に、アンパサンドを貼り付けて、単位がメトリックに等しいと言います。

したがって、ホットリスタートを再度実行すると、印刷する温度が摂氏になっていることがわかります。

華氏の温度が必要な場合は、それを帝国に変更できます。さて、ささいなことを整理したので、ウィジェット内で温度を使用できるようになりました。そして、それを表示する場所はここにあり、32度ではなく、32度を削除して、ここに温度変数をドル記号で追加します。そして、その小さな程度のシンボルを文字列として保持します。

したがって、保存をクリックしてアプリを見ると、12ポイント4度になっており、残りのコンテンツが画面から押し出されていることがわかります。

さて、誰もその正確さの程度の温度を知る必要はありませんか？

必要なのは、小数点以下の桁のない最初の数字だけです。

では、どうすればこれを行うことができますか？温度を倍精度で保存する代わりに、整数、小数点以下のない整数に変更できます。そして、2倍の温度を保存するローカル変数を作成できるようになったので、単にそのtempを呼び出します。そして、実際の値をtemp.toIntとして温度変数に保存できます。これにより、doubleが整数に変換されます。

そして、ホットリスタートを押すと、温度が表示されたときに整数しか得られないことがわかります。

したがって、残りの小数点以下の桁なしで12度を取得します。

そのため、現在の場所の天気データをロケーション画面に正常に解析し、温度、状態、都市名を取得するためにそれを渡すことができ、ウィジェットで既に温度の使用を開始しました。最後にしたいことは、この読み込み画面を少しリファクタリングすることです。

ここに緯度と経度を倍精度で保存する代わりに、非常に多くの変数を作成する代わりに、ここでlocation.latitudeとlocation.longitudeを使用できます。

唯一の問題は、1つのドル記号を使用して変数値を挿入すると、1つのステップしか実行できないことです。そのため、それが識別する最初の変数のみを取ることができます。

location.latitudeまたはlocation plus 3を利用したり、それを使用したい場合は、中かっこを追加する必要があります。

したがって、location.latitudeの周りにいくつかの中括弧を追加し、locationの周りにもいくつか追加します。 経度：この方法では、これらの2行を削除することでコードをさらに簡素化でき、また、すぐにそれらの変数を削除できます。 そして今、私たちのコードは非常に短くなり、それでも表現力は同じです。 取得した場所に等しい緯度を渡します。緯度と経度はlocation.longitudeです。 そして、ロード画面がほぼ完成しました。

そのため、次のレッスンでは、Open Weather Mapから取得したデータを使用して、画面の他の部分を更新する方法を検討します。

それだけではありませんが、次のレッスンでお会いしましょう。