Open Weather Mapのドキュメントで、地理座標に基づいてデータを取得するには、API呼び出しは次のようになっていることがわかります。

api.openweathermap.org

そして、緯度と経度を追加するだけです。

ただし、このリンクをクリックすると、実際には別のサブドメインに移動します。

samples.openweathermap.orgにアクセスして、動作するサンプルと独自のアプリIDを提供します。

これは、サービスに登録して独自のアプリIDを取得する前にサンプルデータを確認できるようにするためです。

しかし、このレッスンでは、ジオロケーターから取得した緯度と経度を使用して、その場所の実際の現在のライブ気象データを取得する方法を学習します。

そのためには、まずAPIに登録する必要があります。

それでは、APIタブをクリックして、新しいアカウントにサインアップしてください。そして、それが完了したら、Open Weather Mapにサインインできます。

そして、サインインしたら、このようなものを見ているは​​ずです。

この2番目のタブリストの下に、APIキーと呼ばれるものがあります。そこで、独自のAPIキーを取得します。そこで、そのキーをコピーしてください。

そして、プロジェクトに貼り付けます。

ロード画面の一番上から、新しい定数を作成します。これは変更されないためです。

これをapiKeyと呼び、今コピーした文字列と等しくなるように設定します。

そのため、このアプリIDを独自のAPIキーに置き換えることができます。したがって、ここに$を追加してAPIキーを追加できます。次に、実際のライブデータを取得するためにURLをサンプルからAPIに変更し、緯度と経度を実際の緯度と経度に変更します

ジオロケーターから戻ってきています。

状態クラス\_LoadingScreenState内で、緯度となるdouble型の新しい変数を作成します。

また、経度用にもう1つ作成します。

そして、このデータを保存します。コンソールに印刷する代わりに、

これらの新しい変数に保存します。

そのため、緯度はlocation.latitudeと等しくなり、経度はlocation.longitudeと等しくなります。

緯度と経度にアクセスできるようになったので、それらをURLに入れることができます。

緯度はここにあります。したがって、等号の後、アンパサンドの前に、ドル記号と緯度変数を追加します。

そして、139の代わりに経度が現在の経度になります。

したがって、これらの値が一致していることを確認してください。latは緯度に等しく、longは経度に等しくなります。

そして、実行を開始する前に、呼び出しを移動して、init状態の後に発生することがわかっているビルドメソッドからデータを取得します。

代わりに、ここでgetLocationに追加します。そして、getDataを呼び出す前に、実際に現在の場所を取得していることを確認します。

この方法では、空の緯度と経度の値を解析することはありません。

それでは、アプリを再実行して、コンソールを確認しましょう。エミュレータで検出された現在の値はクパチーノにあり、温度は285Kで、気象条件は値803であり、完璧です。これで、コードはデバイスの現在のライブ緯度と経度を取得し、Open Weather Map APIを使用してその場所に対応する気象データを取得できるようになりました。そして、JSONから取得した結果を、温度、条件コード、都市名など、必要なさまざまな値に渡すことができます。

今やらなければならないのは、コードを整理してリファクタリングすることです。そうすれば、現在よりも少し良く見えるようになります。ロード画面内ですべてのネットワーキングとリクエストおよびレスポンスのチェックを行う代わりに、カスタムのnetworking.dartサービスファイルに分割します。

そこで、dart：convertとhttp Dartの両方のインポートについて説明します。

ここから切り取り、networking.dartに貼り付けます。ここで、NetworkHelperという新しいクラスを作成します。

そして、このクラスが初期化されると、クラスのコンストラクターで、URLを渡します。

したがって、文字列になる最終変数を持ち、それをurlと呼びます。クラスが初期化されると、そのurlの値を与えます。次に、getDataという新しいメソッドを作成します。

また、ネットワークが完了するのを待ってから、取得したデータを処理する必要があるため、非同期になります。そのため、http getを使用し、ステータスコードを確認し、データをデコードしているすべてのコードを移動します。

そこで、これをここから切り取り、この新しいgetData関数に貼り付けます。

次に、ステートメントのelse部分も含めます。

これも切り取ります。ここに貼り付けます。

したがって、データを非同期的に取得することを言っており、このURLに対してhttp get要求が行われるのを待ちます。そして、応答が200であるかどうかを確認します。この場合、応答本文をdataという変数として保存します。そして、jsonDecodeを使用してそのデータをデコードします。

そのため、緯度と経度、APIキーがわからないため、エラーが発生しています。

そのため、これらすべてを削除し、代わりに、渡されるこのurlプロパティを使用します。

したがって、このgetメソッド内では、そのURLを使用します。次に、ロード画面内で、場所の取得とデータの取得を1つのメソッドに結合します。これをgeLlocation Dataと呼びます。そして、ここで新しいネットワークヘルパーを初期化します。そしてもちろん、これを認識するためには、そ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、servicesフォルダーとnetworking.dartファイルをインポートします。そして、NetworkHelperが何であるかを認識するはずです。

そして、これをnetworkHelperと呼び、URLを渡す必要のあるコンストラクターから作成されます。

そこで、先ほどコピーしたURLを貼り付けます。ジオロケーターから緯度、ジオロケーターから経度、そして定数からアプリキーを取得します。

そのため、このgetDataメソッドを削除し、代わりにそこにあるすべてのものをこれに移動できます

getLocationData。そして、余分な中括弧を削除できます。 printステートメントもすべて削除し、weatherDataという新しい変数を作成して、取得した気象データを利用し、networkHelperがデータを取得するまで等しく待機するように設定します。ですから、先物しか待てないということです。

したがって、このgetDataメソッドを変更して、voidの代わりに返すようにし、futureを返す必要があります。

このようにして、この非同期関数を実行し、それが発生するのを待ってから、このデコードされたデータをメソッドの出力として返します。そのため、このように構成するか、単純にするためにJSONでデコードされたデータを単に返すことができます。ここで、デコードされたデータから必要な特定のものを取り出すこれらの部分をすべて移動し、それをロケーション画面に貼り付けます。

単にファイルの最後に配置し、後でこれらの値を実際にウィジェットに入れる必要があるときに使用するためにコメントアウトします。

したがって、ロード画面ははるかに単純になり、geoLocationパッケージを使用して緯度と経度を取得するだけです。

そして、それをURLに入れ、ネットワークヘルパーを使用して、ここで提供したOpen Weather MapのURLとネットワーク接続してデータを取得し、ステータスコードが何であるかを確認し、返されたデータを返しますそのJSONデコーダーを使用して解析します。

そして、getDataとgetLocationDataを呼び出す必要がなくなりました。

ここで、単一の呼び出しに結合できます。

そのため、このデータをロケーション画面に渡すだけで、これらのコード行を使用してデータをフォーマットし、結果をテキストウィジェットに入れることができます。

それで、そのすべてについて、次のレッスンでお会いしましょう。