アプリケーションプログラミングインターフェイスとは何かを少し学習したので、実際のインターフェイスとやり取りして、Open Weather Mapからデータを取得してみましょう。

そのため、インターネットからデータを取得するために、これを行う方法に関する非常に便利なクックブックドキュメントがあります。

また、httpパッケージの使用方法を順を追って説明します。

これは、Dartチームによって作成されたFlutterパッケージです。

あなたはそれが良いことを知っているし、それも同様に本当に使いやすいです。

そのため、このhttpライブラリを使用して、Open Weather Mapに現在の天気データを取得するよう要求します。

このライブラリの使用方法を見てみましょう。

いつものように、プロジェクトへの依存関係として追加します。

そのため、pubspec.yamlを開き、geolocatorの下に別の依存関係を追加し、packages.getを押してライブラリのすべてのコードをプロジェクトに組み込みます。そして今、私たちがしなければならないことは、ライブラリをインポートすることです。

それでは、ロード画面でそれを行ってみましょう。

location.dartをインポートした場所のすぐ下で、httpライブラリをインポートします。http/ http.dartは、ここでインポートする必要のあるものです。そして今、私はそのライブラリからいくつかのコードを使い始めることができます。ネットワーク要求を行うには、httpライブラリからgetメソッドを呼び出してから、データを取得するURLを呼び出すだけです。ここで新しいメソッドを作成して、getDataと呼びます。ここで、このgetDataメソッド内で、http.dartパッケージをインポートしたためにアクセスできるgetメソッドを使用します。

そのため、このgetsメソッドは入力の1つとしてURLを予期し、そのURLからデータを取得しようとし、将来の応答を返します。

したがって、これは非同期メソッドです。これは、すべてのワイヤを通過してOpen Weather Mapサーバーと通信するのに時間がかかるため、このメソッドには時間がかかる可能性があるためです。

そのため、非同期で動作します。このgetメソッドへの入力として渡すURLは、Open Weather Mapから取得します。

したがって、現在の天気データのAPIドキュメントを下にスクロールすると、地理座標によって天気データを取得することもできます。したがって、lat値とlon値を追加することにより、関心のある場所を渡すことができます。

そして、私たちのために既に構築されているAPIの例を次に示します。

したがって、このサンプルAPI呼び出しをクリックすると、samples.openweathermap.orgと呼ばれるものにアクセスしており、緯度35と経度139を通過していることがわかります。

また、ここには何らかのアプリIDもあります。

サンプルサブドメインを使用する必要がないように、任意の緯度と経度にリクエストを送信できるように、独自のアプリIDも少し後で登録します。

しかし今のところ、ブラウザがそのURLにアクセスして同じURLにアクセスできるのと同じ方法で対応するデータを取得できるように、アプリがこのURLに対して実際にこのリクエストを行うことができるようにすることです。このデータのリクエストを取得します。

このため、このURLをURLバーからコピーし、getMethodに文字列として貼り付けます。そのため、単一引用符で囲みます。

今、このgetメソッドは、将来の応答を返すことを覚えていますか？

したがって、返されるデータを保持したい場合は、応答タイプの新しい変数を作成する必要があり、必要なものを呼び出すことができます。必要に応じて、応答と呼ぶことができます。

そして、この将来の応答をタイプ応答に割り当てることができないことに注目してください。

したがって、このメソッドは非同期で動作するため、将来いつでも解決する可能性があります。

したがって、たとえばコンソールに出力するなど、この応答をどこかで使用したい場合は、他の場所で行ったようにデータを非同期メソッドに変換または取得する必要があります。また、getメソッドの前にawaitキーワードを追加して、応答が出力される前にこれが解決されるのを待ちます。この段階で、先に進んでコードgetDataをトリガーする場合、initState、getDataに追加するか、ビルドメソッド内にアプリを配置するたびにアプリをホットリスタートする必要がない場合セーブをヒットするたび、またはホットリロードをヒットするたびに呼び出されることがわかっています。

どちらの方法でも大丈夫です。

かなり多くテストするので、ここに入れます。

そして、これはこのメソッドgetDataを再トリガーする非常に簡単な方法です。

アプリを実行してコンソール内を見ると、レスポンスのインスタンスが出力されていることがわかります。

応答オブジェクトとは正確には何ですか？

httpパッケージのドキュメントを見て、そのオブジェクトが実際にどのように見えるかを見てみましょう。

そのため、Aboutセクションに進むと、APIリファレンスがあります。これにより、httpパッケージのドキュメントが表示されます。

ここで、使用している応答クラスを検索してみましょう。

また、これは、応答本文全体が事前にわかっているhttp応答であることがわかります。

これが、私たちが取り戻している応答です。新しい応答オブジェクトを作成すると、いくつかのプロパティがあります。それは文字列としての応答の本文である本文を持ち、ヘッダーやリクエストやステータスコードなどのものも持っています。実際に関心のある部分、つまりデータを含む部分は、実際には応答の本文に保持されます。

代わりにresponse.bodyを出力しましょう。

それでは、もう一度実行してみましょう。

そして、ここで見ているのとまったく同じデータを取得していることがわかります。

応答本文は次のとおりです。

現在、ステータスコードと呼ばれるものもあり、応答オブジェクトのプロパティとしてアクセスできます。

これにより、応答のstatusCodeが取得されます。

そのため、代わりに印刷すると、コード200が返されていることがわかります。

では、ステータスコードとは何ですか？

さて、外部サーバーとやり取りする場合、外部サーバーとやり取りしたときに正確に何が起こったのかを、簡潔で統一された方法で伝える必要があります。

これは、特定の結果のコードのようなものです。

404ページが表示されましたか？

それで、ピクサーやレゴ、またはドリブルの様々なデザイナーから有名な人がいます。また、本質的に404ステータスコードは、外部サーバーからアクセスしようとしたリソースが実際には存在しないことを意味します。

したがって、送り返すことができるさまざまなコードがたくさんあり、200にあるものはすべて、リクエストが成功したことを意味します。そして具体的には、コード

200はOKを意味し、すべてがOKであり、ここに結果があります。

404の内容は、リクエストを試みたときに何か悪いことが起こったことを意味します。また、500にあるものはすべて、実際にはサーバーに問題があることを意味します。

そのため、このWebサイトへのリンクを含めて、これらの各コードの意味について詳しく説明します。

また、実際に取得しようとしているstatusCodeが何を伝えようとしているのかを把握しようとしているときに参照できます。

しかし、実際、サンダーホーゲンドールンによるこのチートシートは、彼が行った講演の1つにあります。

1で始まるコードが戻ってくるということは

ちょっと待って、何かが起こっています。

2で戻ってくるものはすべて、成功したときにここに行くことを意味します。 3つで始まるものはすべてなくなり、この特定のリソースを表示する権限がありません。

4で始まるものはすべて、失敗したことを意味します。 5で始まるものはすべて、私がめちゃくちゃになったことを意味します。私はそのメッセージを送信する外部サーバーであるということです。

たとえば、存在しないリソースを取得してみましょう。

天気のようなものを取得しようとする代わりに、ヘザーのようなものを取得しようとすることを知ってみましょう。そして、これはAPIが実際に処理方法を知らないものを参照するようになります。

そのため、もう一度保存ボタンを押して取得データを再実行すると、ステータスコードとして表示されるのは従来の404であることがわかります。したがって、通常、response.bodyが私たちが考えているものであると推測し始める前に、ステータスコードが何であるかを確認することをお勧めします。

というのは、今、response.bodyを利用して何かをしようとした場合、実際に取得しているものは何もないということがわかるからです。

私は体として何も戻ってきません。

また、それを表示するためにアプリでレンダリングしようとすると、予期しないエラーになります。

代わりに、response.statusCodeが200に等しいかどうかを確認してみましょう。これは、すべてが完全に実行されたことを意味します。

次に、この場合、dataと呼ばれる新しい文字列を作成し、response.bodyに設定します。

ただし、応答ステータスコードが200でない場合は、実際に何がうまくいかなかったかを確認するために、単にresponse.statusCodeを出力します。

さて、この時点で、getメソッドが非常に奇妙に見えるということに気付くかもしれないことが1つあります。

パッケージから来たように見えませんか？

通常、外部パッケージを使用している場合、パッケージで作成された何らかのオブジェクトを利用する必要があるためです。

ただし、このgetメソッドとその場所（http.dartファイル内）を見ると、

コマンドを押しながらgetをクリックすると、http.dartファイルが表示されます。このファイルは、ここで読み込み画面に組み込まれています。

その後、ここで実際に作成されたクラスがないことがわかります。

httpクラスはありません。

代わりに、getメソッド、headメソッド、またはpostメソッドのすべてのメソッドにアクセスできます。

これにより、コードはgetgetというメソッドを作成し、独自のメソッドを使用しているように見えます。他の人が来て、彼らがこれが実際にhttpパッケージから来ていることを知らなかった場合、これは一種の混乱を招きます。

このため、Flutterコードでこのhttp.dartパッケージを使用する最も一般的な方法の1つは、キーワード「as」を追加し、そのパッケージに名前を付けることです。通常はhttpになります。

そのため、このhttpという単語を使用して、このパッケージ内にあるものなら何でも使用できます。

したがって、getの代わりにhttp.getを作成し、responseの代わりにhttp.responseを作成することができます。これらはすべてそのパッケージからのものだからです。

したがって、次にこのコードを見るとき、または他の誰かがこのコードを見るとき、このgetメソッドがどこから来たのか、そのhttpパッケージから来たのか、そしてこのオブジェクト応答はhttpパッケージから来たのかは非常に明白です。あなたがそれを知らず、その応答オブジェクトが何であるかを理解しようとすると、私たちはちょうどFlutterドキュメントに行って、応答と呼ばれるこのものを検索しようとしましたが、実際にはここに存在しませんか？

これはそのhttpパッケージのドキュメントと同じではありません。なぜなら、応答オブジェクトはここから来ているからです。

そのため、そのデータを再度出力し、保存をクリックしてコンソールに表示すると、これは文字列として構造化されていることがわかりますが、それ以上のものです。実際には、キーと値のペアに基づいて特定のものを利用できるようにフォーマットされています。そして、これがJSON形式のAPIレスポンスと呼ばれるものです。そして、次のレッスンでは、実際にそれが何を意味するのか、そしてそれをどのように使用するのかについてもう少し学びます。