さて、最後のレッスンでは、scratch.dartファイルを使用して、futuresおよびasync and awaitについてさらに学習しました。

ただし、このファイルはもう必要ないので、削除するか、将来参照するために保管しておくことができます。

プロジェクトには影響しません。

それでは、ロード画面に戻りましょう。コードを見てみると、もう少しわかりやすいはずです。

geolocator.dartからの新しいgeolocatorオブジェクトを作成していることがわかります。そして、そのオブジェクトを使用して現在の位置を取得しています。 getCurrentPositionを呼び出しており、これはgeolocator内のメソッドです。そして、この場所に必要な精度を提供します。

そして、すべてのバッテリーを使い果たさないように、低を選択します。また、現在の位置を取得するのに時間がかかるため、このメソッドは非同期であり、試行しようとする他のこととは無関係に発生します。しかし、ほとんどの場合、たとえば、戻ってきた位置を印刷したり、天気データを取得する際にその位置を使用したりすることしかできないため、関数を変更するためにasyncキーワードを追加しています。そして、それは基本的に、あなたが何かを続ける前にこれが完了するまで待つと言うawaitキーワードへのアクセスを与えます。このawaitキーワードがなかった場合、ポジションはまだありますが、実際のポジションにはなりません。

それは将来の位置になります。

注文番号のようなものです。

このプロセスが完了すると、将来価値があります。ただし、このコードが呼び出された時点またはトリガーされた時点では、これは単なる領収書です。

実際のコーヒーではありません。

そして今、この位置変数を印刷する場合、実際に出てくるのは実際の位置ではありません。そして、これをscratch.dartの実行からmain.dartに変更することでそれを示すことができます。

そして今、実行を押して場所を取得をクリックすると、main.dartのコンソールに印刷されているものが表示されるので、必要に応じてscratch.dartを閉じることができますポジション。これは、友人にコーヒーを飲みに行くように約束するようなもので、あなたがしたことは領収書を持って戻ってくるだけです。

今、彼らは領収書を望んでいません。

彼らは実際のコーヒーが欲しくなるでしょう。

そのため、メソッド呼び出しの前にawaitキーワードを追加して、この値が位置に割り当てられる前にこれが完了するのを待ちます。

こうすることで、将来のコーヒーの約束だけでなく、実際の位置や実際のコーヒーを実際に取得できます。

そして、現在位置取得をクリックすると、未来だけでなく実際の位置が印刷されるのを見ることができます。

そのため、このボタンを押したときに場所を取得できるのは非常に良好であり、良いことです。

しかし、画面が読み込まれたらすぐに場所が必要な場合はどうでしょうか？天気アプリでは、アプリを開くとすぐに現在地を検出しようとし、その場所に基づいて天気が取得されるためです。

ユーザーにボタンを押して天気を表示したり場所を取得したりするように強制することは意味がありませんが、少し余分に思えます。それで、画面がロードされたらすぐにそれを実行したい場合、どのように位置取得を呼び出すのでしょうか？

これを行うには、ウィジェットのライフサイクルについて学習する必要があります。

ステートレスウィジェットは基本的に非常にシンプルなレゴブロックのようなものです。

それらを半分で見ることはできませんし、変更することもできません。

それらを破壊して新しいものを作成することを決定しない限り、あなたはそれらで何もすることができません。

また、ステートレスウィジェットの変更が必要になるたびに、新しいものを破棄して作成し続ける必要があります。

そのため、これらのウィジェットのライフサイクルメソッドは非常にシンプルです。心配する必要があるのは1つだけで、それがビルド方法です。ウィジェットが構築されると、このメソッドが呼び出され、ここで、画面に表示する外観またはウィジェットを作成します。

一方、ステートフルウィジェットもあり、これらのステートフルウィジェットを組み合わせて、状態オブジェクトを使用して状態を追跡できることがわかります。

状態オブジェクトは、ウィジェットの構成、ウィジェットのプロパティなどの変数を追跡します。

また、設定状態を使用してこれらの変数をすべて変更すると、アプリが更新されます。現在、この場合、状態オブジェクトの実際の寿命ははるかに長いため、ライフサイクルメソッドが増えています。

初期化状態があり、その状態が最初に初期化されるとトリガーされます。

もちろん、ウィジェットが実際にビルドされたときにトリガーされ、画面に表示されるビルドメソッドがあります。そして、ステートフルウィジェットが破棄されたときに呼び出されるdeactivateメソッドがあります。

ですから、人間が生まれ、成長するたびに、さまざまなライフステージを経て、死んでしまうのです。ステートフルウィジェットもそうです。しかし、さまざまな時点で異なることを実行したい場合は、ライフサイクルのこれらの各段階を活用できます。

ここで、これらのライフサイクルメソッドがトリガーされるタイミングを示します。そのために、以前に使用したナビゲーションデモである以前のファイルを取得します。

これで、コードを記述する必要はありません。

私は実際に何が起こるかを見せたいだけです。

従うだけで十分です。ここには、ステートレスウィジェットである画面1がまだあります。

現在、ステートフルウィジェットであるscreen 2があります。そして、最初のルートである画面1からアプリを開始しました。

そのため、アプリを実行すると、画面1に移動します。画面1には、画面2にプッシュするボタンが1つしかありません。

画面2に移動したら、これらのライフサイクルメソッドが実際に呼び出されたときに表示できるようにします。

状態オブジェクトの内部では、親クラスである状態クラスからのいくつかのメソッドを利用します。その1つがinitStateメソッドです。

このinitStateをクリックし、CONTROL + JまたはCONTROL + Qを押すと、オブジェクトがツリーに挿入されたときに呼び出されることがわかります。

そして、これは、ステートフルウィジェットがツリーに挿入されるとすぐに作成するときに、initStateを呼び出すことを意味します。

これが最初に起こることです。

そのため、ここに「initState called」という印刷ステートメントを追加します。そして、このメソッドがトリガーされるたびに印刷されます。そして、この状態オブジェクトの寿命の特定の段階で自動的にトリガーされます。

トリガーされる2番目の要素、つまりライフサイクルの2番目の要素は、ビルドメソッドになります。

ここで、「build called」というprintステートメントも追加します。

最後に、非アクティブ化と呼ばれる親クラスからの別のメソッドも紹介します。また、このステートフルウィジェットが破棄されると、非アクティブ化がトリガーされます。

それで、ここにもprintステートメントを追加しましょう。

次に、変更を実行するために[実行]をクリックします。コンソールでは、このボタンをクリックするとすぐに、画面2に移動する必要があることがわかります。

ステートフルウィジェットが作成され、ウィジェットツリーに挿入されるとすぐに、initStateが呼び出されます。

ここに印刷されます。

そして、それがちょうど私のボタンや私の足場などである画面内にすべてのウィジェットを構築するとすぐに、私たちは呼ばれます。これで、画面を変更したり、ウィジェットが依存するプロパティの1つを変更したりするたびに、ビルドが何度も何度も呼び出されます。

そして、これは最も頻繁に使用されるライフサイクルメソッドの1つです。

しかし、最後にクリックして画面1に戻ると、画面2が飛び出します。つまり、画面2は破壊されます。

存在しなくなります。

そして、それはdeactivateが呼び出される時です。

そのため、他のライフサイクルメソッドもありますが、これらはおそらく最も有用であり、Flutterアプリを作成するときに実際に出会って必要になるものです。

そのため、ステートフルウィジェットが作成されてツリーに追加されたときに何かを発生させたい場合は、コードをinitState内に配置することがわかります。ステートフルウィジェットが再構築されるたびに何かを発生させたい場合は、ビルドメソッドに入れます。

最後に、ステートフルウィジェットが破棄されたときに何かを実行したい場合は、deactivateメソッド内にコードを配置します。

そこで、ライフサイクルメソッドで学んだことを使用して、ここでの課題を説明します。

コードを変更して、ボタンをタップせずにコンソールに位置が印刷されるようにしてください。

実際、足場の内容全体を削除することができます。

起こるべきことは、アプリが実行されるとすぐに、この読み込み画面が電話内に表示されるとすぐに、コンソールに自分の位置が印刷されるのが見えることです。

そして、これはあなたがこのレッスンから学んだことに依存します。

ビデオを一時停止し、そのチャレンジを完了してみてください。

そのため、現在、アプリを実行すると、空の画面しか表示されません。

ビルドの中には何もありません、ただの足場です。

そのため、ボタンを使用して場所の取得をトリガーするつもりはありません。

代わりに、initStateをロード画面に追加します。ステートフルウィジェットが作成されるとすぐにトリガーされることがわかっているので、画面に表示される瞬間です。

ここで、initStateが呼び出されるのは1回だけであり、その状態が初期化されて作成されるのはその瞬間だけであることに注意することが重要です。

しかし、ウィジェットが画面上に構築されると呼び出されますが、ビルドのようなものですが、ウィジェットが再構築されるたびにキャッチされます。

そのため、テキストの一部、画像、またはこのアニメーションが変更されるたびに、ビルドが何度も何度も呼び出されます。また、非常に高価なため、繰り返し呼び出されるコードをそこに入れたくない場合が非常に多くあります。

代わりに、initState内に配置します。

そしてinitState内で、get locationを呼び出します。

そして、この特定のステートフルウィジェットを再初期化するためにアプリをホットリスタートすると、画面に表示されるとすぐに表示され、コンソールに自分の位置が表示されます。

何も押す必要はありませんでした。したがって、ステートフルウィジェットのライフの特定の瞬間を利用したい場合、ライフサイクルメソッドは本当に便利です。データを保存したい場合

ステートフルウィジェットが破棄される前、またはメモリから何かの割り当てを解除する場合、またはステートフルウィジェットが初期化された直後に新しいオブジェクトを作成する場合、これらは特定のコードを実行するために利用できるメソッドです時間。そして、私たちの場合、ロード画面の状態が初期化されるとすぐにget locationメソッドを実行します。これは最初の段階で、1回だけ実行します。現時点では、現在位置を取得しており、ジオロケーターを読み込み画面内にすべてインポートしていますが、これは仕事ではありませんか？

そのため、次のレッスンでは、ユーザーが場所を取得する許可を与えているかどうかを確認する方法について詳しく学習します。

また、場所に関連するすべての作業をlocation.dartファイルにリファクタリングします。

それで、それ以上については、次のレッスンで説明します。