スプリント2: アプリのインストール

スプリント 1 の振り返りで、次のことを決定しました。

> アプリのインストーラーを作成しているため、プレースホルダー アプリを配置して実際にインストールする必要があります。

私たち自身も同意しているので、それをスプリント 2 のテーマにしましょう。スプリントのタスクは次のとおりです。

- アプリ プロジェクトを作成します。
- プロジェクト参照を追加します。
- プレースホルダー コンポーネントを app.exe に置き換えます。
- ExampleComponents の名前を AppComponents に変更します。
- 社名を修正します。
- ビルドに成功しました。

私たちのスプリントの終了基準は引き続き非常にシンプルです。変更によってビルドが壊れていないことを確認するだけです。

この章のページ

‹ スプリント 2: アプリのインストール　上　スプリント 2: アプリ プロジェクトを作成する ›

スプリント2: アプリプロジェクトを作成する

アプリ チームの仲間たちは予定より遅れています。どうやら、彼らの日々は、アプリ ウィンドウの配色と、それを WinForms と WPF のどちらで構築するかの議論にほとんど費やされているようです。そのため、今のところは、インストールできるものを用意するために、プレースホルダー アプリを作成するだけにします。アプリ チームが最終的に行うことを間違って推測した場合は、それを修正するだけです。

Visual Studio でアプリ プロジェクトを作成するには:

1. ソリューションを右クリックして を選択します `Add|New Project` 。
2. `C#` 言語ドロップダウンで選択します。
3. `Windows Forms App (.NET Framework)` プロジェクト テンプレート リストで選択します。
4. 選ぶ `Next` 。
5. `App` プロジェクト名として入力します。
6. 選ぶ `Create` 。

> `App` よりももっと説明的なプロジェクト名にすべきだと思うかもしれません。しかし、マーケティング チームは、多くのケータリング オフサイトの 1 つで、製品の名前と、名前を「とてもクール」にするために母音を省く方法についてまだ議論しています。待つのではなく、彼らが思いついたものに対応するだけです (そして、ケータリングの残りの一部を解放するかもしれません)。

‹ スプリント 2: アプリのインストール　上　スプリント 2: プロジェクト参照を追加する

スプリント2: プロジェクト参照を追加する

これで、コンパイルしてインストーラーに含めるアプリ プロジェクトができました。そのためには、少なくとも 2 つのことを行う必要があります (他にもいくつかあります)。

1. パッケージ プロジェクトの前にアプリ プロジェクトがビルドされていることを確認してください。そうすることで、パッケージがビルドされたときにアプリ自体が使用できるようになります。
2. パッケージ プロジェクトがアプリ プロジェクトの出力を見つけるための何らかの方法を提供します。

WiX でこれらを実現する最善の方法は、プロジェクト参照(具体的には、パッケージ プロジェクトからアプリ プロジェクトへの参照) を使用することです。プロジェクト参照を使用すると、Visual Studio (および CI ビルド システムの MSBuild) は、最初にアプリ、次にパッケージという正しい順序でプロジェクトをビルドします。

次に、WiX MSBuild ターゲットが起動し、アプリ プロジェクトからの出力を簡単に取得できるように特別な処理を実行します。これについては、このスプリントの後半で説明します。

Visual Studio でプロジェクト参照を追加するには:

1. ソリューション エクスプローラーでパッケージ `Dependencies` の下を右クリックします。 `WixTutorialPackage`
2. 選ぶ `Add Project Reference` 。
3. プロジェクトを確認し `App` 、 を選択します `OK` 。

‹ スプリント 2: アプリ プロジェクトを作成する

上

スプリント 2: プレースホルダーコンポーネントの置き換え ›

スプリント2: プレースホルダーコンポーネントを置き換える

前回のスプリントを覚えているかもしれませんが、HeatWave テンプレートはインストーラーにパッケージ化されるコンポーネントを提供してくれました...しかし、それは単なるプレースホルダーでした。そこで、それを実際のものに置き換えてみましょう。まだ実際には本物ではありませんが、テンプレートのプレースホルダーをアプリのプレースホルダーに置き換えます。これは私たちのものであり、それが特別なのです。

さらに思い出していただくために、コンポーネントは次のようになります。

```
<Component>
  <File Source="ExampleComponents.wxs" />
</Component>
```

私たちの目標は、WiX ソース ファイルがインストールされるのではなく (意味がないため)、このスプリントの前半で作成したアプリがインストールされるように変更することです。

ExampleComponents.wxs という名前のファイルの代わりにアプリをインストールするには、属性の値を変更するだけでよいことに驚かないでしょう `Source` 。しかし、どのような値を使用すればよいのでしょうか。

ハードコードされた絶対パスを使用することもできます。

```
<Component>
  <File Source="x:\path\to\App\bin\Debug\App.exe" />
</Component>
```

しかし、同僚は、チームが使用する Git リポジトリのクローンに対してまったく同じパスを設定する必要があることに不満を感じるかもしれません。また、ハードコードされたパスがクラウドでホストされている CI/CD ビルド システムで機能する可能性はまったくありません。

もう 1 つの方法は、相対パスを使用することです。アプリ プロジェクトとパッケージ プロジェクトは同じ Visual Studio ソリューションと同じディレクトリ ツリーにあるため、相対パスが機能する可能性があります。C# .NET Framework プロジェクトは、 のようなディレクトリに .exe 出力をビルドします `bin\Debug\App.exe` 。このパスは C# プロジェクトに対する相対パスであるため、パッケージ プロジェクトからは、次のような相対パスを使用できます。

```
<Component>
  <File Source="..\App\bin\Debug\App.exe" />
</Component>
```

しかし、出力パスは常にこのように相対的でしょうか? 通常はそうなりますが、常にそうであるとは限りません。 `Debug` パスの一部をハードコーディングしていることにも気付いたかもしれません。ビルドに使用しているソリューション構成を尊重する方法はありますか? 幸いなことに、WiX MSBuild ターゲットはこの種の問題を予測し、 `Configuration` MSBuild プロパティを同じ名前の WiX プリプロセッサ変数に変換します。MSBuild が使用するのと同じ構文を使用して参照できます。 `$(variable_name)` したがって、その変数を使用した相対パスは次のようになります。

```
<Component>
  <File Source="..\App\bin\$(Configuration)\App.exe" />
</Component>
```

これは悪くないですが、もっと良いものはないでしょうか? きっとできるはずです。WiX MSBuild ターゲットは、それ自体は知覚力はありませんが、参照されているプロジェクトからのファイルを簡単にパッケージ化できるべきという WiX 開発者の考えを反映しています。これは、参照されているすべてのプロジェクトの出力に対してバインド パスを生成することによって実現されます。バインド パスは、WiX がパッケージ化されるファイルを検索するパスです。バインド パスにはディレクトリが含まれているため、ファイル名自体を参照するだけで済みます。

```
<Component>
  <File Source="App.exe" />
</Component>
```

これで完璧です。

‹ スプリント 2: プロジェクト参照を追加する

上

スプリント 2: ExampleComponents の名前を AppComponents に変更 ›

スプリント 2: ExampleComponents の名前を AppComponents に変更する

OK、これは私の個人的な不満です。HeatWave テンプレートによって生成されたプロジェクトでは、`ExampleComponents` いくつかの場所で名前が使用されていますが、テンプレートでは意味があるかもしれませんが、プロジェクトにはふさわしくありません。これらは例ではなく、私たちのアプリです。または、最終的にはそうなるでしょう。 の代わりに `ExampleComponents` を使用しましょう `AppComponents` 。

Visual Studio の「ファイル内を検索」検索を使用して、使用されているすべての場所を見つけることができます `ExampleComponents` 。修正してみましょう。

ExampleComponents.wxs を開き、 `ExampleComponents` 次の ID を置き換えます `ComponentGroup` :

```
<ComponentGroup Id="AppComponents" Directory="INSTALLFOLDER">
```

コンポーネント グループの名前を変更したので、その参照の名前も変更する必要があります。

Package.wxs を開き、次の ID を更新します `ComponentGroupRef` 。

```
<Feature Id="Main">
  <ComponentGroupRef Id="AppComponents" />
</Feature>
```

忘れないように、WiX ソース ファイル ExampleComponents.wxs の名前を AppComponents.wxs に変更しましょう。ソリューション エクスプローラーでファイル ExampleComponents.wxs を右クリックし、[名前の変更] を選択して、ファイルの名前を AppComponents.wxs に変更します。

‹ スプリント 2: プレースホルダー コンポーネントを置き換える　上　スプリント2: 社名を修正する ›

スプリント2: 社名を修正する

スプリント 1で議論したように、 `Package` 要素の必須属性の 1 つは `Manufacturer` 、パッケージに所有者を与える です。HeatWave は、すぐにビルドされないプロジェクトを与えないように、デフォルトを提供します。また、注目されて修正される可能性のある製造元名 も提供します。 `TODO Manufacturer` それでは、その名前を修正しましょう。

`Manufacturer` Package.wxs を開き、属性の値を から に変更し `TODO Manufacturer` ます `Edgerock Concepts` 。

```
<Package
  Name="WixTutorialPackage"
  Manufacturer="Edgerock Concepts"
  Version="1.0.0.0"
  UpgradeCode="64deef2a-cf99-4a0c-be41-5faa802a9502">
```

簡単。

‹ スプリント 2: ExampleComponents を AppComponents に名前変更　上　スプリント 2: プロジェクトの構築 ›

スプリント2: プロジェクトの構築

HeatWave を使用して Visual Studio で MSI パッケージ プロジェクトをビルドするには、 `WixTutorialPackage` ソリューション エクスプローラーでプロジェクトを右クリックし、 を選択します `Build` 。出力ウィンドウには、App プロジェクトが WixTutorialPackage プロジェクトの前にビルドされ、このスプリント中に他の変更が何も起こらなかったことが表示されます。よかったです!

```
Build started...
1>------ Build started: Project: App, Configuration: Debug Any CPU ------
1>  App -> x:\path\to\App\bin\Debug\App.exe
2>------ Build started: Project: WixTutorialPackage, Configuration: Debug x86 ------
2>WixTutorialPackage -> x:\path\to\WixTutorialPackage\bin\x86\Debug\en-US\WixTutorialPackage.msi
========== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build started at 8:35 PM and took 01.579 seconds ==========
```

‹ スプリント2: 社名を修正する 上 スプリント2の振り返り ›

スプリント2の振り返り

スプリントの振り返りはスプリントを終了し、チームが何がうまくいったか、何がうまくいかなかったか、そしてチームがどのように改善できるかについて正直に話し合う機会を与えます。スプリント 2 では、どうでしたか?

何がうまくいきましたか?

パッケージ プロジェクトの変更をランダムに行うだけでなく、アプリのインストールに備えて変更を開始することができました。

もっとうまくいく方法は何でしょうか?

まだ実際のアプリをインストールしていませんが (チームはカラー スキームをほぼ決定しています)、アプリとそのインストーラーの両方をどのようにテストするかについて考え始める必要があります。幸いなことに、私たちはアプリ チームより先に進んでいるので、ペースを維持できれば、チーム全員にインストーラーを使用してアプリをテストするよう説得できるはずです。これは二重取りの良い例です。つまり、アプリだけでなくインストーラーも専用にテストすることになります。

次のスプリントで何を改善しますか?

わかりました。テストが重要であることはわかりました。次のスプリントでは、インストーラーを安全にテストする方法を考えましょう。アプリ チームが準備を整えれば、準備は完了です。アプリ チームが準備を整えれば。

‹ スプリント 2: プロジェクトの構築 上 スプリント 3: 仮想テスト ›