

ソフトウェア工学

オブジェクト指向プログラミング

4I No.4 aridai

(GitHub: @aridai / Twitter: @aridai_net)

今回話すこと

- できたもの
- 使ったもの
- プロジェクトの構成
- WPFについて (フレームワーク/アーキテクチャ)
- Rxについて
- P/Invokeについて
- テストについて
- 感想

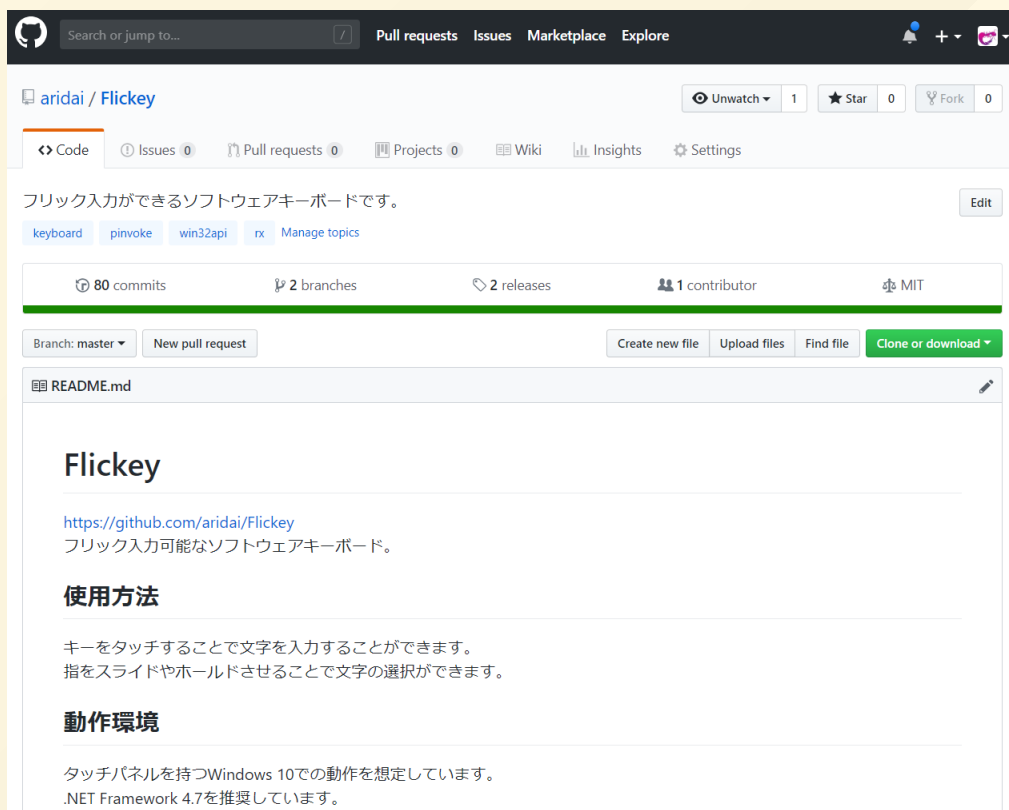
できたもの

- フリック入力可能なソフトウェアキーボード。
- Windows 10で動く。(タッチパネルが必要。)
- 指をスライドさせたり、ホールドさせたりして入力できる。

できたもの

設計図と配布物は設計図共有サイトでMITライセンスで公開します。

<https://github.com/aridai/Flickey>



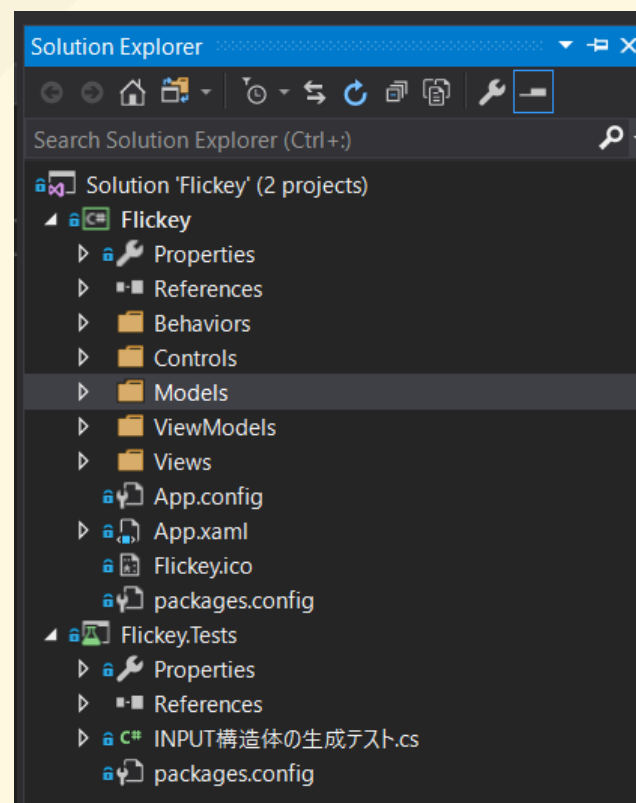
使ったもの

- 言語: C# 7.3
- GUIフレームワーク: WPF
- テストフレームワーク: MsUnit
- ライブラリ: Prism / ReactiveProperty
- IDE: Visual Studio Community 2017
- バージョン管理: Git + git-flow + 設計図共有サイト

プロジェクトの構成

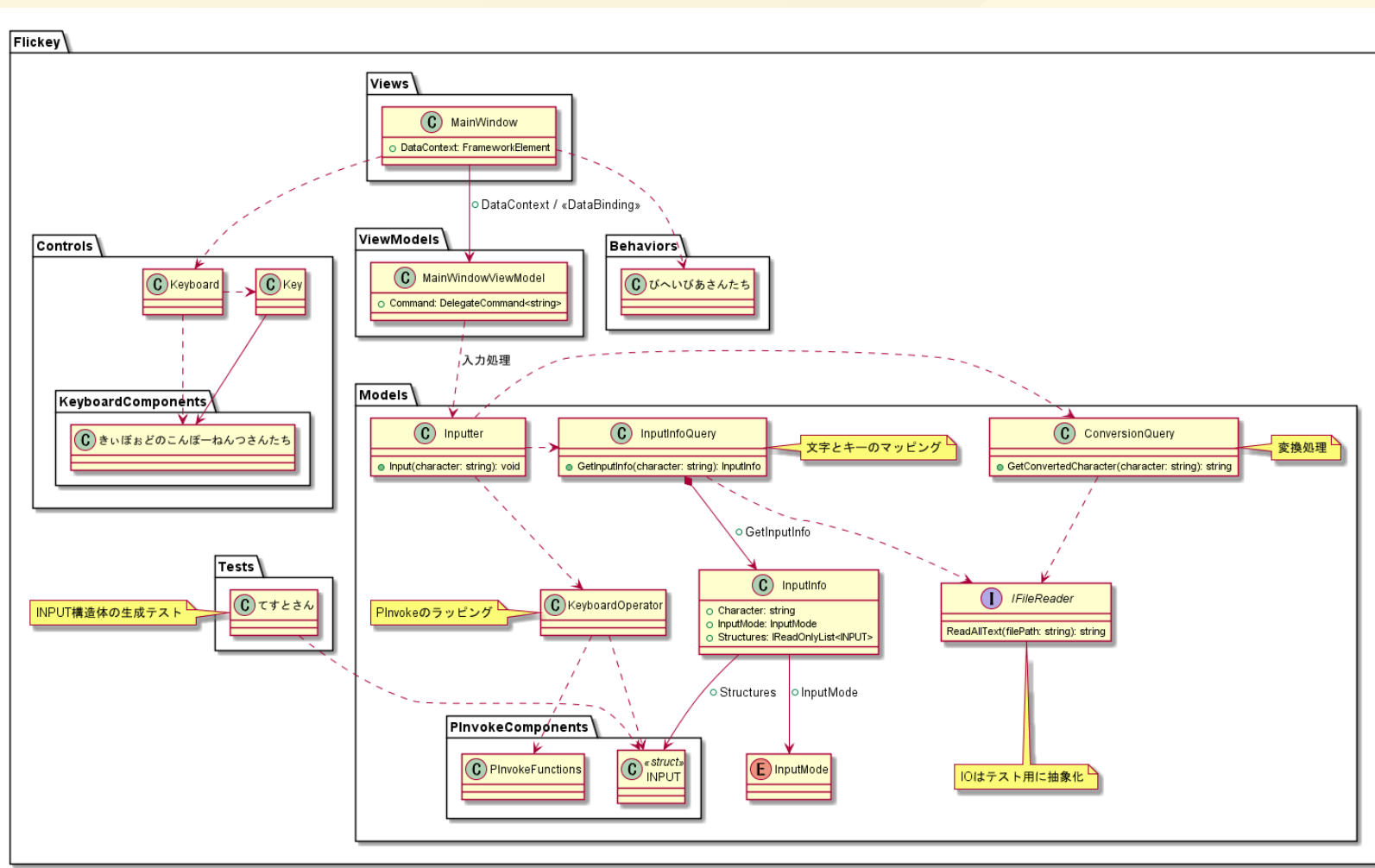
Visual Studioソリューションがこんな感じに。

- 2プロジェクト。
(WPFApp + UnitTest)
- MVVMパターン +
Behaviorパターン。
- UI部分はユーザコントロール
として作成。



プロジェクトの構成

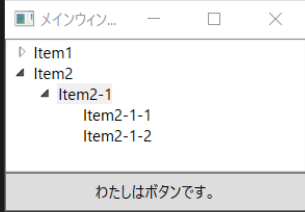
クラス図はこう。一応、頭の片隅に**SOLID**原則を。



WPFについて

Windowsアプリケーションのフレームワーク。
UIをXAMLを使って記述する。

```
1 <Window
2   x:Class="WpfApp.MainWindow"
3   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
7   mc:Ignorable="d"
8   Title="メインウィンドウ" Width="500" Height="400">
9
10  <Grid>
11    <Grid.RowDefinitions>
12      <RowDefinition Height="*" />
13      <RowDefinition Height="30" />
14    </Grid.RowDefinitions>
15    <Button Grid.Row="1" Content="わたしはボタンです。" />
16    <TreeView Grid.Row="0">
17      <TreeViewItem Header="Item1">
18        <TreeViewItem Header="Item1-1">
19          <TreeViewItem Header="Item1-1-1" />
20          <TreeViewItem Header="Item1-1-2" />
21        </TreeViewItem>
22        <TreeViewItem Header="Item1-2">
23          <TreeViewItem Header="Item1-2-1" />
24          <TreeViewItem Header="Item1-2-2" />
25        </TreeViewItem>
26      </TreeViewItem>
27      <TreeViewItem Header="Item2" IsExpanded="True">
28        <TreeViewItem Header="Item2-1" IsExpanded="True" IsSelected="True">
29          <TreeViewItem Header="Item2-1-1" />
30          <TreeViewItem Header="Item2-1-2" />
31        </TreeViewItem>
32      </TreeViewItem>
33    </TreeView>
34  </Grid>
35</Window>
```



WPFについて

- **MVVMパターン**

Model・**View**・**ViewModel**の3つの層に分けて設計・実装を行うアーキテクチャパターン。

- **Behaviorパターン**

ViewのUI処理を**Behavior**の層で行う。(語弊あり)

- **データバインディング**

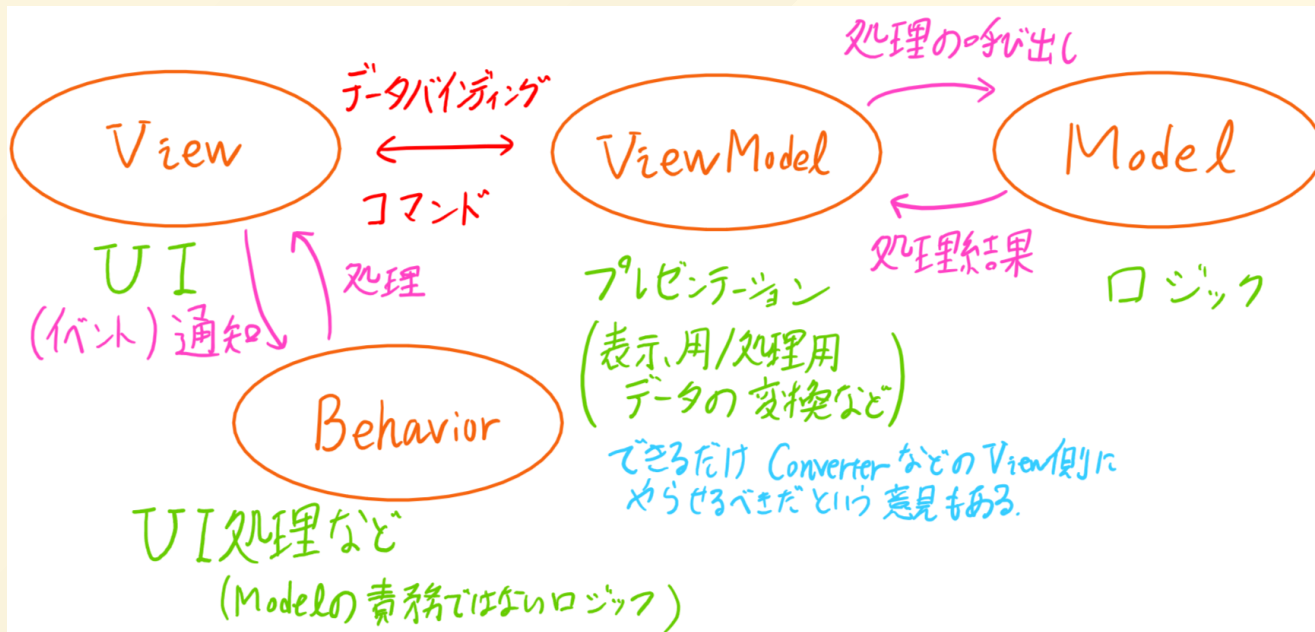
一方のデータが変更されると他方のデータも更新される仕組み。

- **コマンド**

操作とそのパラメータ (引数/利用可能状態) をまとめて1つのものとして扱ったもの。

WPFについて

今回の開発も**MVVMパターン**に従って、画面部分と処理部分と分けて作るようにした。
具体的には、キーボードの制御 (表示の切り替えなど) に関しては、キーの入力処理が知らなくてもいいようになっている。(疎結合でみんな幸せ!)



Rx (Reactive Extensions) について

LINQの**observer-pattern**版みたいなやつ。

キーボードのタイミング制御などが複雑になりそうだったため、これを使うと、時間やスレッドをまたぐ制御が楽になった。

```
// 長押し判定の例
var holdDetection = operationTarget
    .Select(target => target.deviceId) // データの選択
    // IDが有効で、
    .Where(id => id != null) // 通知のフィルタ
    // 一定時間経過したら、
    .Delay(timeOut) // 時間差通知
    .ObserveOnDispatcher() // スレッド変更
    .Where(id => id == deviceId)
    // 長押し判定にする。
    .Select(_ => OperationType.Hold);
```

P/Invokeについて

- **P/Invoke**

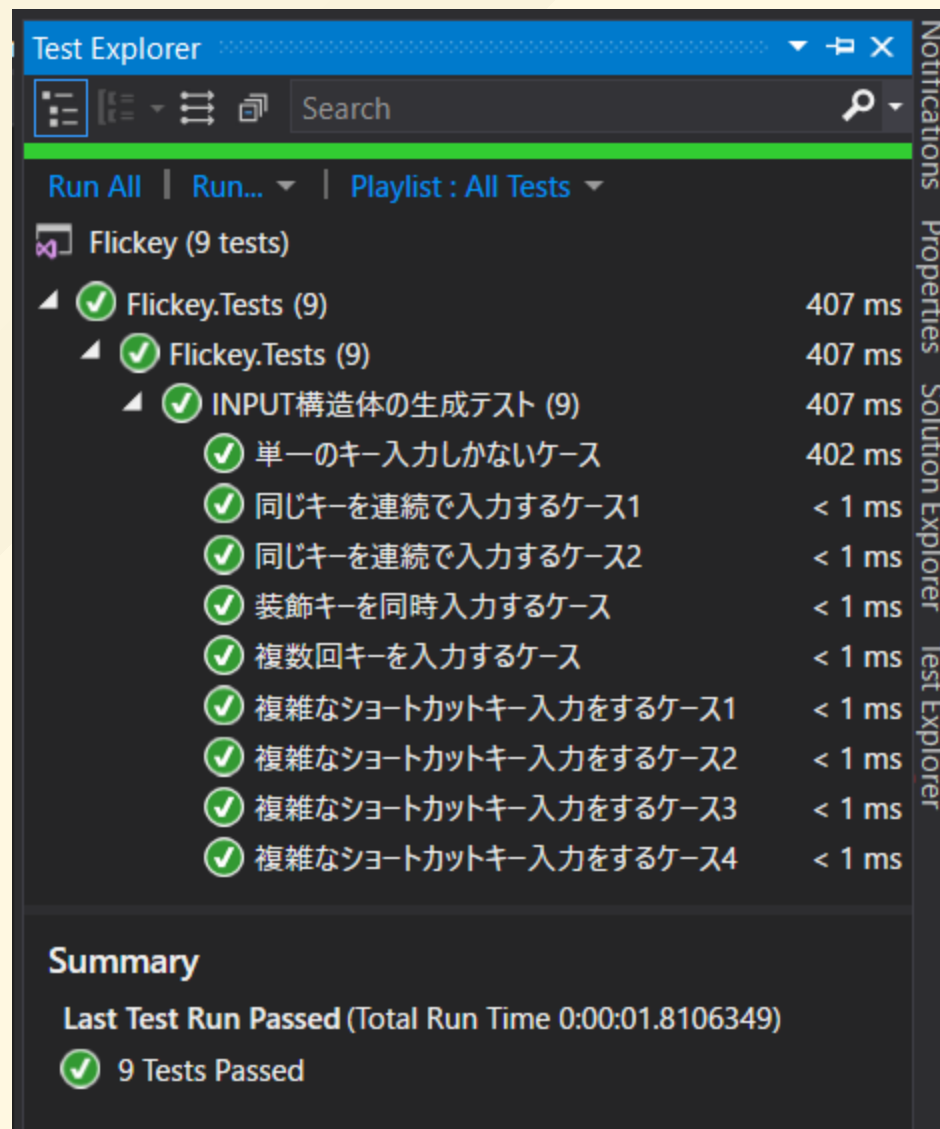
C# から **C++** のコードを呼び出す機能。(語弊あり)

IMEの処理をするには、**Win32 API** を利用しなくてはならず、それは **C++** で書かれているため、**P/Invoke** を利用して呼び出した。(語弊あり)

```
[DllImport("user32.dll")]  
public static extern int SendMessage(  
    IntPtr hWnd, uint Msg, uint wParam, int lParam);  
  
[DllImport("user32.dll")]  
public static extern int SendInput(  
    int nInputs, INPUT[] pInputs, int cbSize);
```

テストについて

- MsUnitを利用した。
(xUnitと迷った。)
- 入力データの生成が正しくできているかをテストした。
- 先にテストケースを書いて実装した。
- テストが通るとめっちゃ気持ちいい!



感想

前期に座学でやったことを意識しながら開発ができてよかったと思います。

今回は規模的/時間的にできなかったけれども、**TDD**や**DIコンテナ**、**CIサーバ**なども試してみたいと思った。

タッチパネル付きのWindowsPCを持っている人は是非使ってみてね。

issues / pull requests 大歓迎です。

リポジトリ: <https://github.com/aridai/Flickey>