Custom Activityの
デバッグ方法

Shingo Mori
November 2018

Custom Activityのデバッグ方法

- Custom Activityとは
- いわゆる「Printデバッグ」の方法
- デバッガによるデバッグ方法

Custom Activityとは

Custom Activityとは

- UiPaht Studioでは、ビルトインのコアアクティビティと、必要に応じて特定のフィードからインストール出来る専用アクティビティ（PDF、メール、Excel、等）が提供されている
- Custom Activity = 開発者が独自に実装した機能を持つ専用アクティビティ

Custom Activityの作り方

- Visual StudioでC#を使ってクラスライブラリ（.dll）を作り、その後パッケージ化（.nupkg）する
- Custom Activityの詳しい作り方は公式ガイド「カスタムアクティビティの作成」参照

いわゆる「Printデバッグ」の方法

Printデバッグとは？

- コードに標準出力関数やデバッグ出力関数を埋め込んで出力を確認するデバッグ方法
- C#のWindowsアプリの場合、System.Diagnostics.Debug.WriteLineの出力をDebugViewで確認可能

メリット

- 開発環境のないコンピューター上でも確認できる

デメリット

- 対話的実行が出来ない

Printデバッグ手順

1. コードにデバッグ出力関数を埋め込む

```
//例
System.Diagnostics.Debug.Flush();
System.Diagnostics.Debug.WriteLine("Hello, UiPath!");
System.Diagnostics.Debug.WriteLine("Hello, Developer Community!");
```

2. プロジェクトをビルドしてパッケージを作成、ワークフローに組み込む
   ※詳細な手順は「カスタムアクティビティの作成」参照
3. DebugViewを管理者権限で起動
4. ワークフローを実行し、DebugViewで出力を確認

デバッガによるデバッグ方法

Visual Studioのデバッガを使用してデバッグを行う

- Visual Studioでは、実行中のUiPathExecutor.exeプロセスにデバッガーをアタッチしてデバッグを行う事が可能

メリット

- 対話的実行が出来る
  - 好きな箇所でブレークポイントが張れる
  - 変数の中身が見れる
  - コールスタックを辿れる

デメリット

- デバッグ手順がPrintデバッグに比べて複雑

プロセスへアタッチによるデバッグ手順

1. プロジェクトをビルドしてパッケージを作成、ワークフローに組み込む
2. ワークフローを実行中の状態にする
    1. Custom Activityを実行する手前にMessage Box Activityを配置する
    2. Workflowを実行する -> Message Boxが表示され、ワークフロー実行中の状態になる
3. Custom Activityライブラリをプロセスにアタッチする
    1. Visual Studioの[デバッグ] - [プロセスにアタッチ]によりUiPathExecutor.exeプロセスにアタッチする
4. Custom ActivityのC#コードにブレークポイントを設定する
5. C#コードのブレークポイントまで処理を進める
    1. Message Boxのボタンを押してワークフローを再開する
    2. C#コード内でブレークする

-> デバッグ可能になったので、テップ実行等を行ってデバッグを進める

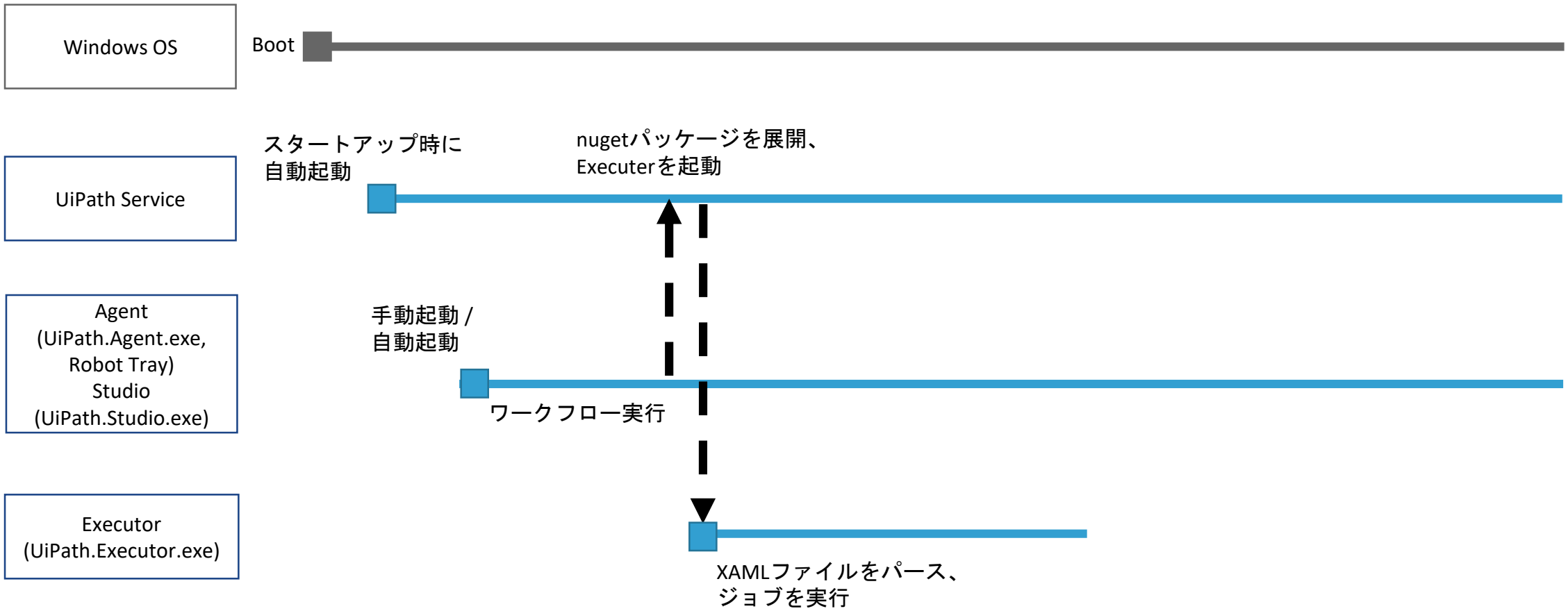
※2018.2よりも前のバージョンではアタッチするプロセスがUiPathExecutor.exeではなくUiRobot.exeなので注意

補足

なぜワークフロー実行中の状態にしてからプロセスアタッチを行うのか？

➢ アタッチ対象であるUiPathExecutor.exeプロセスはワークフローが実行されるまで存在しないため

ジョブ実行時のプロセスライフサイクル（※2018.2～）



UiPath

Thank you