

RPA学習コース

第2回目：UiPathでロボプロセスをサクッと作ってみる

2021年3月18日 19時

1HB-8S 黄 蔚菁

前回のおさらい

第1回目：RPA 紹介と無料な UiPath 開発環境構築

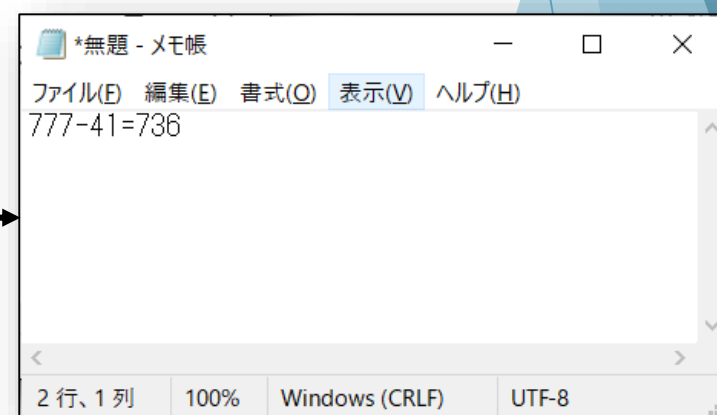
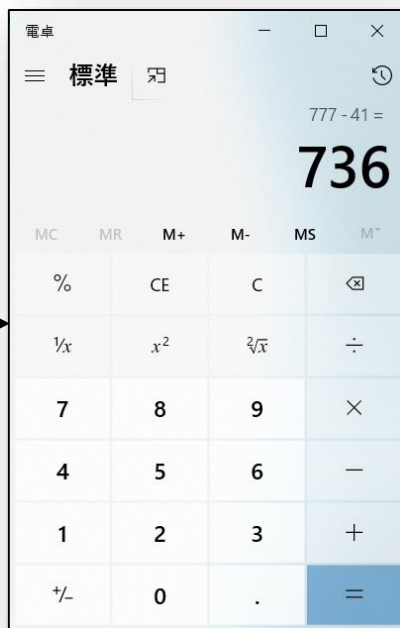
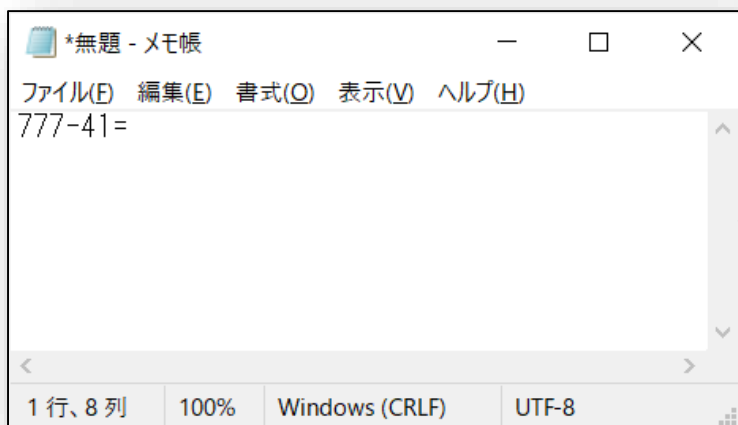
- ▶ RPA概要
 - ▶ RPAとは／RPAデモ／RPAが求められる理由／伝統的なプログラミング言語と比較
- ▶ 無料なUiPath開発環境構築の方法を把握した
- ▶ 録画機能で初めてのロボを作ってみた

前回の宿題

- ▶ まだStudioのインストールを完了していない方はスライド「10分で無料な開発環境構築してみよう」を参照して、次回の勉強会までに完了してください。
(必須)
- ▶ 普段の生活や仕事の中で、ロボ化できそうなことってありますか。いくつか考えてみてください。(オプション)

本日の勉強会を終えると、あなたは...

- ▶ セレクターで画面UI要素の操作ができる。
- ▶ 変数を利用して、アプリケーションの間にデータ連携ができる。
- ▶ 分岐／繰り返し／サブフローを使って処理フローの制御ができる。
- ▶ 上記知識を用い、こんなロボを作れるようになる（デモ）



アジェンダ

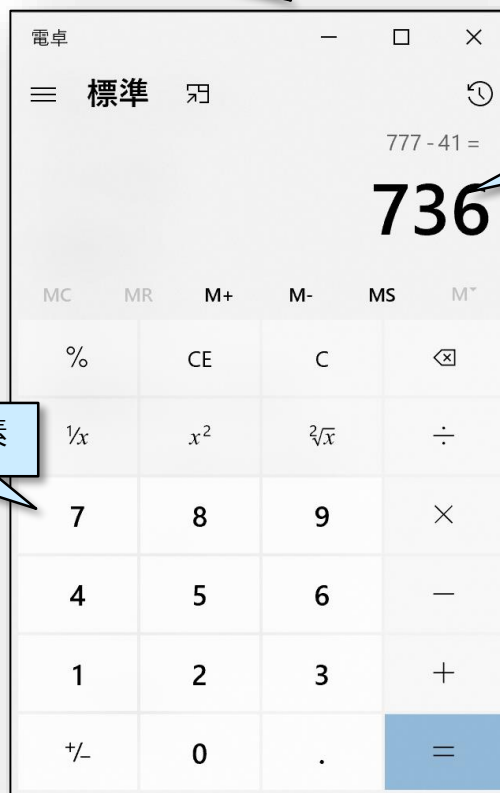
- ▶ UI要素の操作
- ▶ アプリケーション間のデータ連携
- ▶ フロー制御
- ▶ 宿題

UI要素の操作

UI要素とは

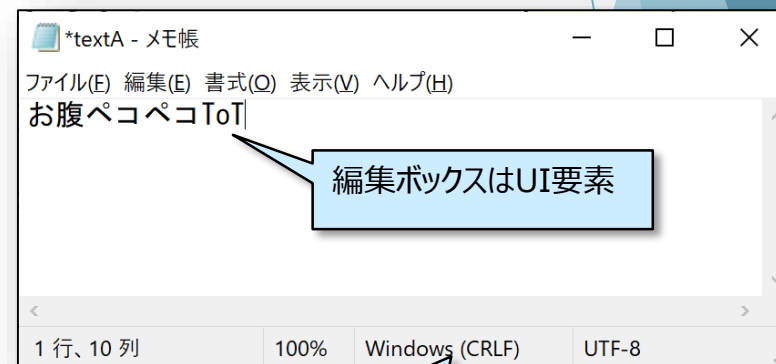
- ▶ ウィンドー
- ▶ ラベル
- ▶ ボタン
- ▶ メニュー
- ▶ 編集ボックス
- ▶ リストボックス
- ▶ コンボボックス
- ▶ スクロールバー
- ▶ その他

ウィンドー全体はUI要素



表示ラベルはUI要素

ボタンはUI要素



編集ボックスはUI要素

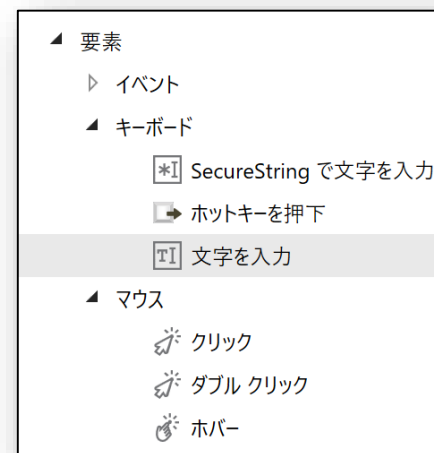
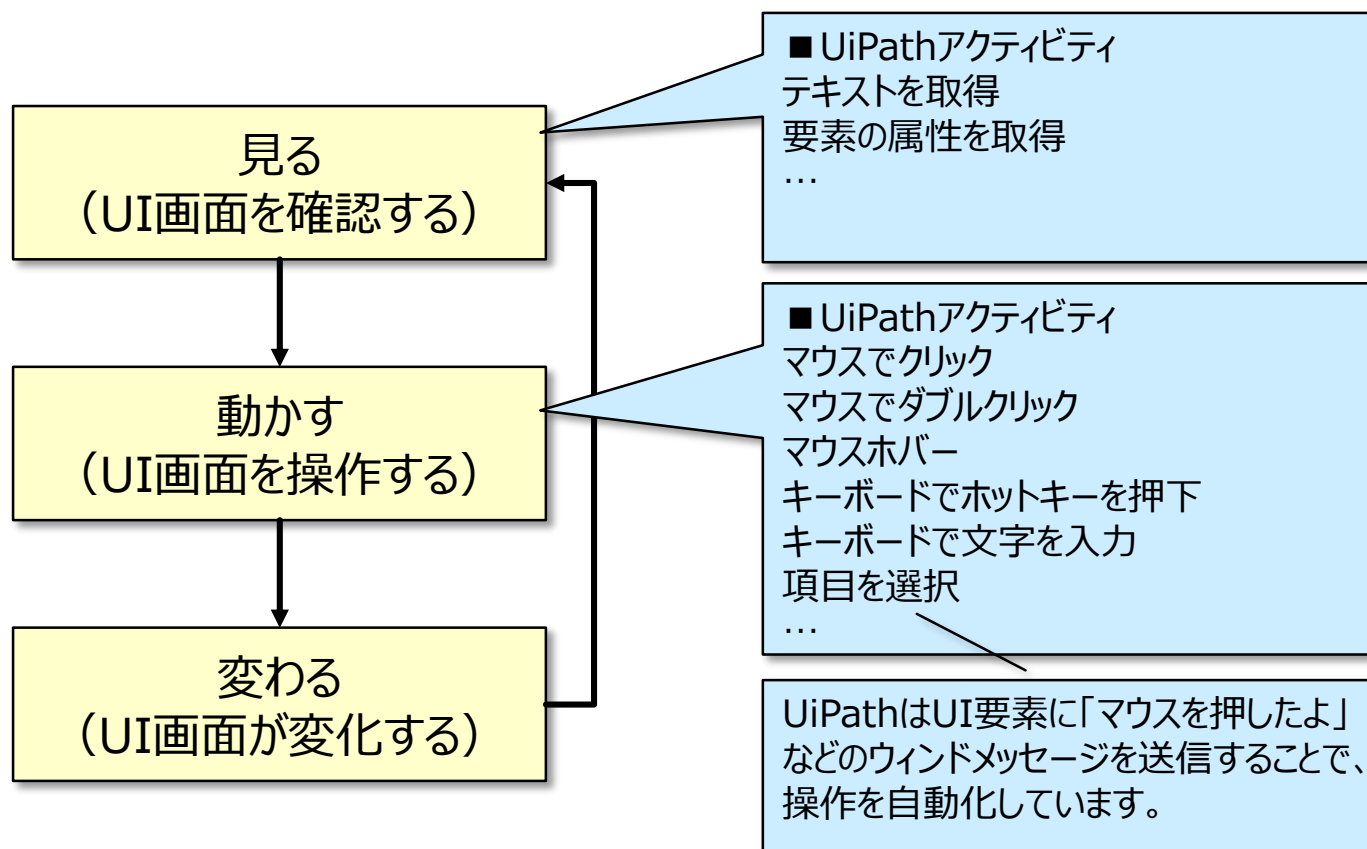
ステータスバーはUI要素

UI要素の操作

UiPathアクティビティ

■どんなUI操作でも、分解すると下記3種類の動作になる。

「見る」、「動かす」ためのUiPathアクティビティが大量に提供されている（皆さんの武器になる）

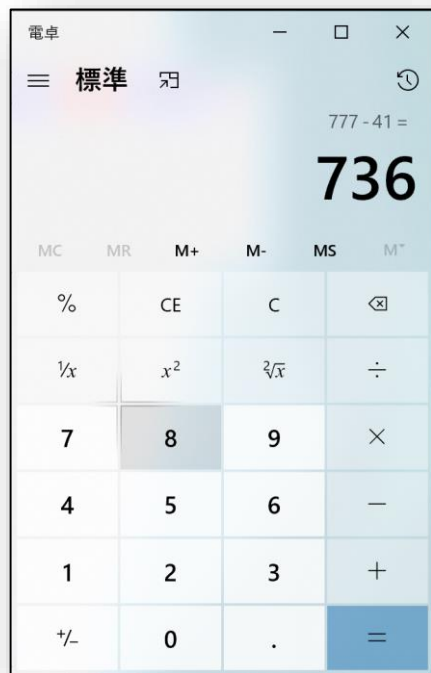


UiPathのアクティビティ
(抜粋)

UI要素の操作

セクターとは

- ▶ セクターは、アクティビティが操作対象とする**UI要素を特定するためのテキストデータ**。操作したいUI要素を選択（セレクト）するからセクターと呼ばれる。



```
<wnd app='applicationframehost.exe' appid='Microsoft.WindowsCalculator_8wekyb3d8bbwe!App' title='電卓' />
<uia cls='LandmarkTarget' />
<uia automationid='NumberPad' cls='NamedContainerAutomationPeer' name='数字パッド' />
<uia automationid='num8Button' cls='Button' name='8' />
```

電卓アプリの「ボタン 8」を表すセクター

「ボタン 8」のセクター



マウスクリック
アクティビティ



「ボタン 8」をクリック

セクターの使用例その一



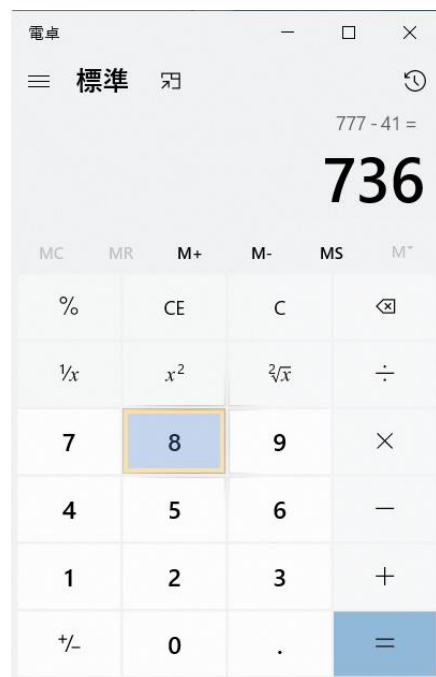
UI要素の操作

UI要素のセレクターを取得するには

①ドラッグ&ドロップでアクティビティ追加



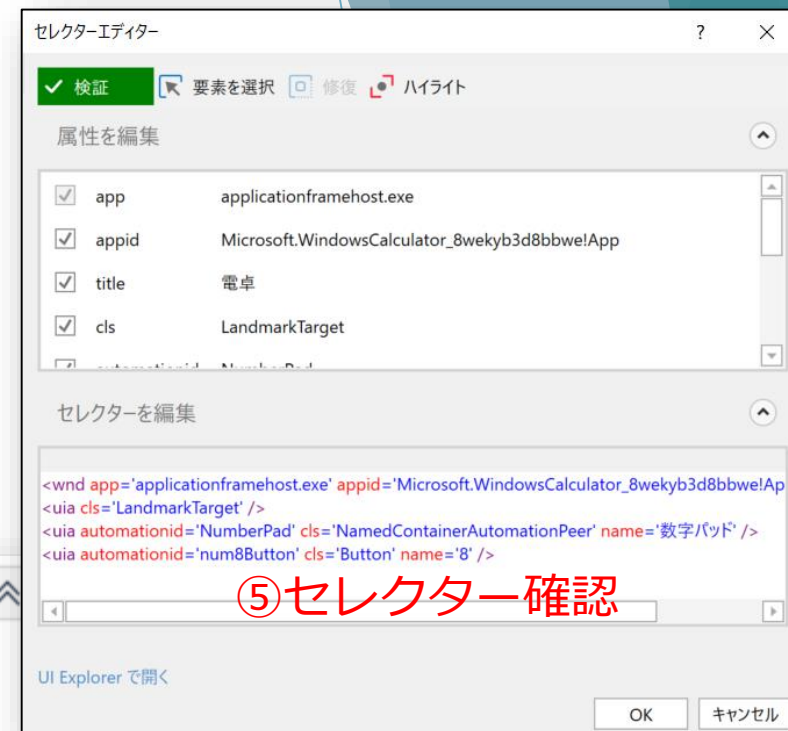
②「画面上で指定」をクリック



③操作したいUI要素をクリック



④セレクター編集で取得したセレクターを確認



画面上で指定

セレクターを編集

UI Explorer で開く

参考スクリーンショットを変更

参考スクリーンショットを削除

参考スクリーンショットを表示 (ダブルクリック)

UI要素の操作

UI要素のセレクターを取得するには

The screenshot displays the UiPath Studio Pro interface. The top toolbar contains various icons for file operations, package management, and development tools. The 'UI Explorer' icon is highlighted with a red box. A red arrow points from this box to the UI Explorer window below.

The UI Explorer window is divided into several sections:

- ビジュアルツリー (Visual Tree):** Shows the application hierarchy. The selected element is 'button 9'.
- プロパティエクスプローラー (Property Explorer):** Lists the properties of the selected element, including 'aaname', 'aastate', 'app', 'AppPath', and 'automationid'.
- セレクターエディター (Selector Editor):** Provides a list of selected and unselected items for the selector. The selected items are:
 - <wnd app='applicationframehost.exe' appid='Microsoft.WindowsCalculator_8wekyb3d8bbwe!App' title='電卓' />
 - <uia cls='LandmarkTarget' />
 - <uia automationid='NumberPad' cls='NamedContainerAutomationPeer' name='数字パッド' />
 - <uia automationid='num9Button' cls='Button' name='9' />

The bottom status bar indicates 'ターゲット要素: 'button 9'' and 'UIAutomation v20.10.9'.

UI Explorer

UI要素の操作

自動取得したセレクトターを編集する

```
<wnd app='notepad.exe' cls='Notepad' title='textA - メモ帳' />  
<wnd aaname='水平' cls='Edit' />  
<ctrl name='テキスト エディター' role='editable text' />
```



メモ帳を編集する処理がある。編集ボックスを特定するための、左記のセレクトターは、textAというファイルが開かれた場合のみ有効になるため、宜しくない一例となる。

```
<wnd app='notepad.exe' cls='Notepad' />  
<wnd aaname='水平' cls='Edit' />  
<ctrl name='テキスト エディター' role='editable text' />
```



Titleを削除することで、titleにどんな文字が入っても、対象として識別される。

```
<wnd app='notepad.exe' cls='Notepad' title='*' />  
<wnd aaname='水平' cls='Edit' />  
<ctrl name='テキスト エディター' role='editable text' />
```



ワイルドカードを指定することで、titleにどんな文字が入っても、対象として識別される。

TIPS : セレクトターの定義をよ〜く観察して、開発環境に依存しない、かつ対象UI要素を十分に特定できるセレクトターを定義することが大事！

UI要素の操作

練習時間

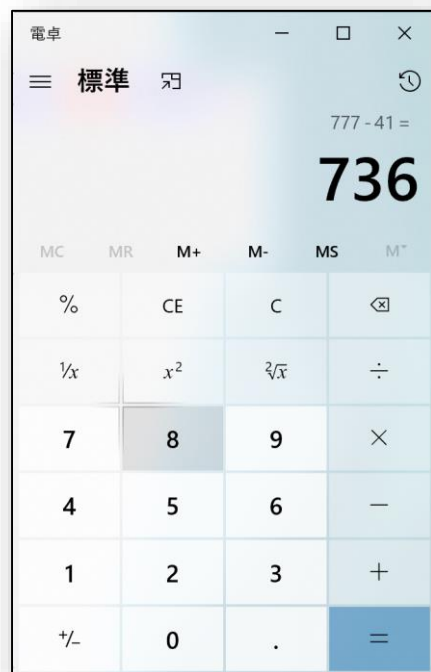
～～練習時間（5分間）～～

- ①マウスクリックアクティビティを利用して、電卓の「ボタン8」をクリックする。
- ②セレクトーのテキストを直して、「ボタン9」を押すように修正してください。

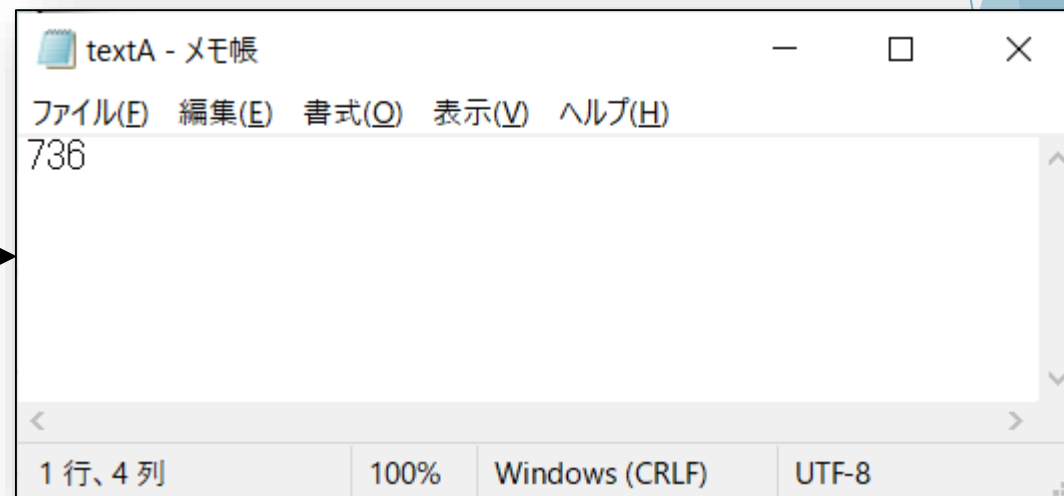
アプリケーション間のデータ連携

変数について

- ▶ 処理と処理の間で引き渡されるデータは、それぞれを区別できるように、名前がついた箱に入れられる。この箱を変数という。変数といっても、数値とは限らない。文字列や日付など、いろいろな種類のデータを変数で扱うことができる。



電卓出力：
"736"



アプリケーション間のデータ連携

変数を定義する

名前	変数の型	スコープ	既定値
学生名	String	シーケンス	C# の式を入力してください
学籍番号	String	シーケンス	C# の式を入力してください
所属クラス番号	String	シーケンス	C# の式を入力してください
変数の作成			
変数パネルで変数値を定義する			

変数 引数 インポート

100%

TIPS : 変数名は日本語で定義しても構わない！むしろ日本語を推奨する！

アプリケーション間のデータ連携

よく見かける変数タイプ

▶ 変数と引数の設定

データタイプ	型	
整数	Int32	123
小数	Double	421.5
文字列	String	“がちゃがちゃ”
論理	Boolean	true/false
期間	TimeSpan	1.01:30:00.100（1日間1時間30分0秒と100ミリ秒）
日時	DateTime	2021/03/18
配列	Array of [T]	new String[] { "abc", "cde" }

配列は頻繁に使われているが、コレクションと合わせて、次回に紹介させていただきます。

★UiPathは.NET Framework との親和性が高く、NETの各種データタイプはそのまま使用可能
データタイプ辞書：Microsoft社の.Net Framework公式リファレンス

<https://docs.microsoft.com/ja-jp/dotnet/api/system.string?view=net-5.0>

アプリケーション間のデータ連携

変数の代入

- ▶ 代入
- ▶ 複数代入
- ▶ 情報取得用のアクティビティで設定

A*B 代入

学生名	=	"黄"
-----	---	-----

等号の右側には代入したい値
(あるいは変数や引数)を入れる

A*B 複数代入

学生名	...	=	"黄"	...	×
学籍番号	...	=	"10052510137"	...	×
所属クラス番号	...	=	"SEI0501"	...	×

追加

等号の左側には設定したい変数や引数を入れる

アクティビティ「テキストを取得」

テキストを取得 'text 表示は 736 です'

777 - 41 =

736

プロパティ

UiPath.Core.Activities.GetValue

その他

プライベート ☐

入力

ターゲット Target

共通

エラー発生時に実行を継続 現在のアクティビティ ☒

表示名 テキストを取得 'text 表示は 736 です'

出力

値 指定した UI 要素から取得 ...

出力欄に変数や引数を入れることで、実行後に取得したテキストが該当引数や変数に代入される。

アプリケーション間のデータ連携

変数中身の確認

▶ 変数の中身を見てみたい時は？

→ 「メッセージをログ」 アクティビティでプリントアウトしてみよう！

シーケンス

複数代入

学生名 ... = "黄" ...

学籍番号 ... = "10052510137" ...

所属クラス番号 ... = "SEI0501" ...

追加

メッセージをログ

ログレベル Info

メッセージ 学生名

出力

検索

次のファイルの実行を開始しました: シーケンス

フロー制御の実行が開始されました。

黄

フロー制御の実行が終了しました。in: 00:00:04

出力パネルで変数値が出力

アプリケーション間のデータ連携

変数のメソッドを活用する（オプション）

- ▶ 前述したようにUiPathは.Net Frameworkのデータタイプをそのまま使用しているので、.Netの提供メソッドはすべて使える。

The screenshot shows the 'シーケンス' (Sequence) editor in UiPath. It contains two main sections: 'A+B 複数代入' (Multiple Assignment) and 'メッセージをログ' (Log Message). In the 'A+B 複数代入' section, three variables are being assigned: '学生名' (Student Name) to '黄' (Yellow), '学籍番号' (Student ID) to '10052510137', and '所属クラス番号' (Class Number) to 'SEI0501'. Below these is a '追加' (Add) button. In the 'メッセージをログ' section, the 'ログレベル' (Log Level) is set to 'Info' and the 'メッセージ' (Message) is '所属クラス番号.ToLower()'. A green arrow points from this section towards the output window on the right.

The screenshot shows the '出力' (Output) window in UiPath. It displays a list of messages from the sequence execution. The messages are: '次のファイルの実行を開始しました: シーケンス' (Started execution of the next file: Sequence), 'フロー制御の実行が開始されました。' (Flow control execution started.), 'sei0501' (The class number variable), and 'フロー制御の実行が終了しました。in: 00:00:00' (Flow control execution ended. in: 00:00:00). A blue arrow points from the 'sei0501' message to a callout box that says '小文字化して出力' (Output in lowercase).

小文字化して出力

アプリケーション間のデータ連携

変数のメソッドを活用する（オプション）

<https://docs.microsoft.com/ja-jp/dotnet/api/system.string?view=net-5.0>

String クラス (System) | Microsoft | X

docs.microsoft.com/ja-jp/dotnet/api/system.string?view=net-5.0#methods

バージョン: .NET 5

検索

String

- コンストラクター
- > フィールド
- > プロパティ
- > メソッド
- > 演算子
- > 明示的なインターフェイスの実装
- > StringComparer
- StringComparison
- > StringNormalizationExtensions
- StringSplitOptions
- > SystemException
- > ThreadStaticAttribute
- > TimeoutException
- > TimeSpan

ToCharArray(Int32, Int32)	このインスタンスの指定した部分文字列の文字を Unicode 文字配列へコピーします。
ToLower()	この文字列のコピーを小文字に変換して返します。
ToLower(CultureInfo)	指定されたカルチャの大文字と小文字の規則を使用して、この文字列のコピーを小文字に変換して返します。
ToLowerInvariant()	インバリアント カルチャの大文字と小文字の規則を使用して、この String オブジェクトのコピーを小文字に変換して返します。
ToString()	String のこのインスタンスを返します。実際の変換処理は実行されません。
ToString(IFormatProvider)	String のこのインスタンスを返します。実際の変換処理は実行されません。
ToUpper()	この文字列のコピーを大文字に変換して返します。
ToUpper(CultureInfo)	指定されたカルチャの大文字と小文字の規則を使用して、この文字列のコピーを大文字に変換して返します。
ToUpperInvariant()	インバリアント カルチャの大文字と小文字の規則を使用して、この String オブジェクトのコピーを大文字に変換して返します。

このページは役に立ちましたか?

Yes No

この記事の内容

- 定義
- 注釈
- コンストラクター
- フィールド
- プロパティ
- メソッド**
- 演算子
- 明示的なインターフェイスの実装
- 拡張メソッド
- 適用対象
- スレッド セーフ
- こちらをご覧ください

すべて表示

Microsoft社の公式.Net Frameworkリファレンス

先ほどのString.ToLower()の仕様が詳しく書かれている。

アプリケーション間のデータ連携

変数をセレクトターに入れる

- ▶ 変数をセレクトターに入れることで、ダイナミックにUI要素の特定ができる。

```
<wnd app='applicationframehost.exe' appid='Microsoft.WindowsCalculator_8wekyb3d8bbwe!App' title='電卓' />  
<uia cls='LandmarkTarget' />  
<uia automationid='NumberPad' cls='NamedContainerAutomationPeer' name='数字パッド' />  
<uia automationid='num{{入力値}}Button' cls='Button' name='{{入力値}}' />
```

{{変数名}}でセレクトターに変数を設定できる。
上記の定義で変数：「入力値」という変数の値が変わると、セレクトターで特定する電卓ボタンも変わる

変数とセレクトタの組み合わせ利用はUiPathのコアスキルである。やり方をぜひ覚えてください！

アプリケーション間のデータ連携

練習時間

～～練習時間（5分間）～～

- ①メモ帳の内容を取り出してログに出力してください。
- ②メモ帳に入れられた数字で電卓ボタンを押す処理を作ってみよう。

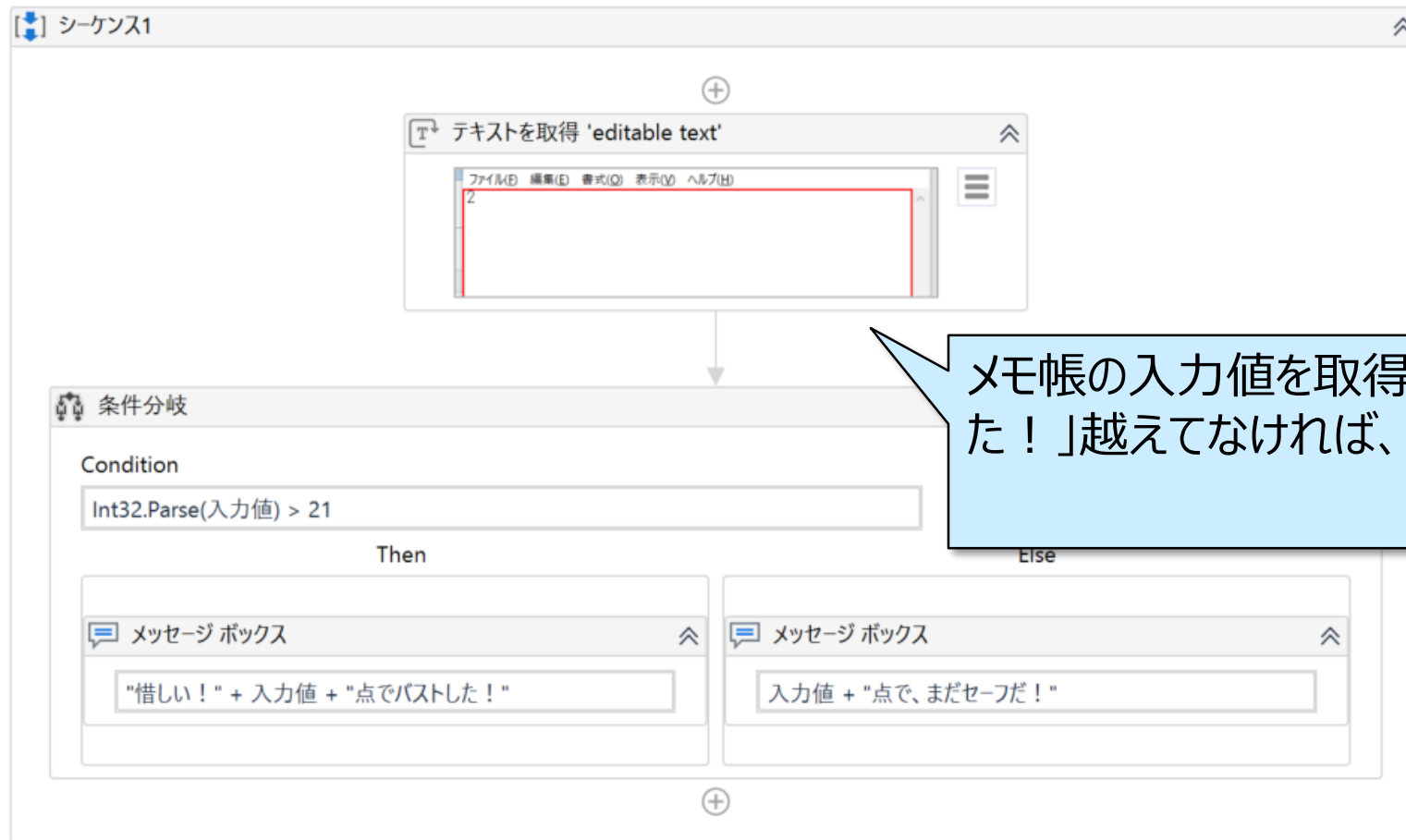
フロー制御

- ▶ 下記のように、プログラムと同じようなフロー制御をUiPathも提供している。これらを組み合わせることによって、どんなフローでも作れる。本日はデモをお見せしながらフロー制御を皆さんにマスターして頂く。

Java/C#の世界でこう呼ぶ	UiPathの世界でこうよぶ
IF文	条件分岐
FORループ	繰り返し
メソッド	サブフロー

フロー制御

条件分岐のサンプル



メモ帳の入力値を取得し、21点を超えたら、「惜しい、バストした！」越えてなければ、「まだセーフだ！」とポップアップされる。

フロー制御

繰り返しのサンプル

【】 繰り返しサンプル

+

🔄 繰り返し (コレクションの各要素)

繰り返し 次のコレクション内の各要素:

本文

【】 本文

+

📝 メッセージをログ

ログ レベル

メッセージ

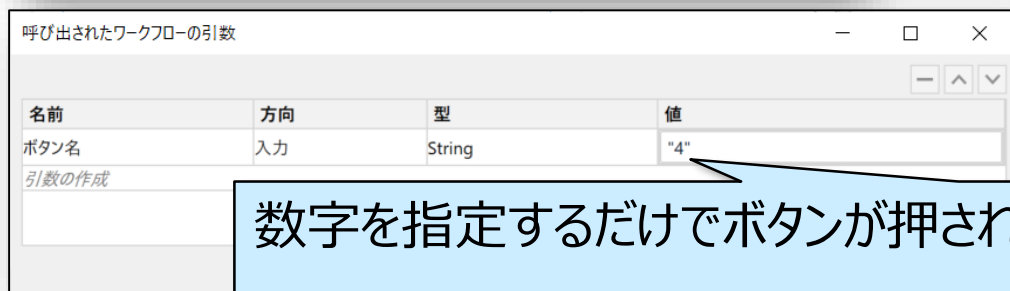
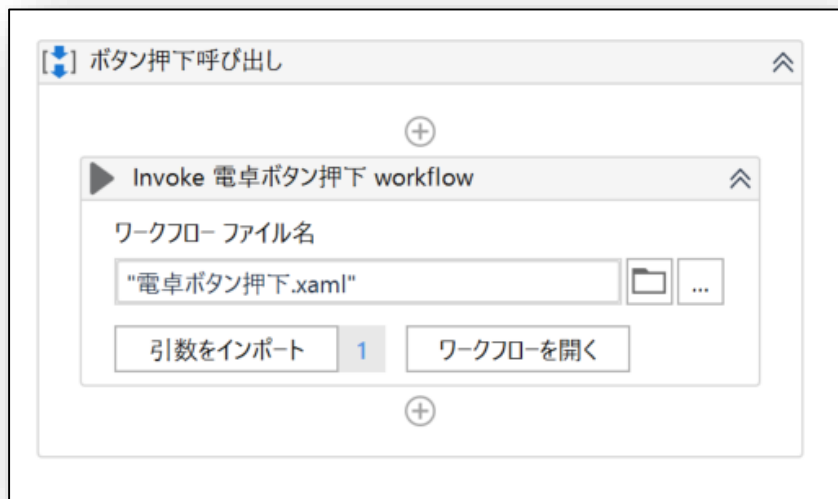
+

+

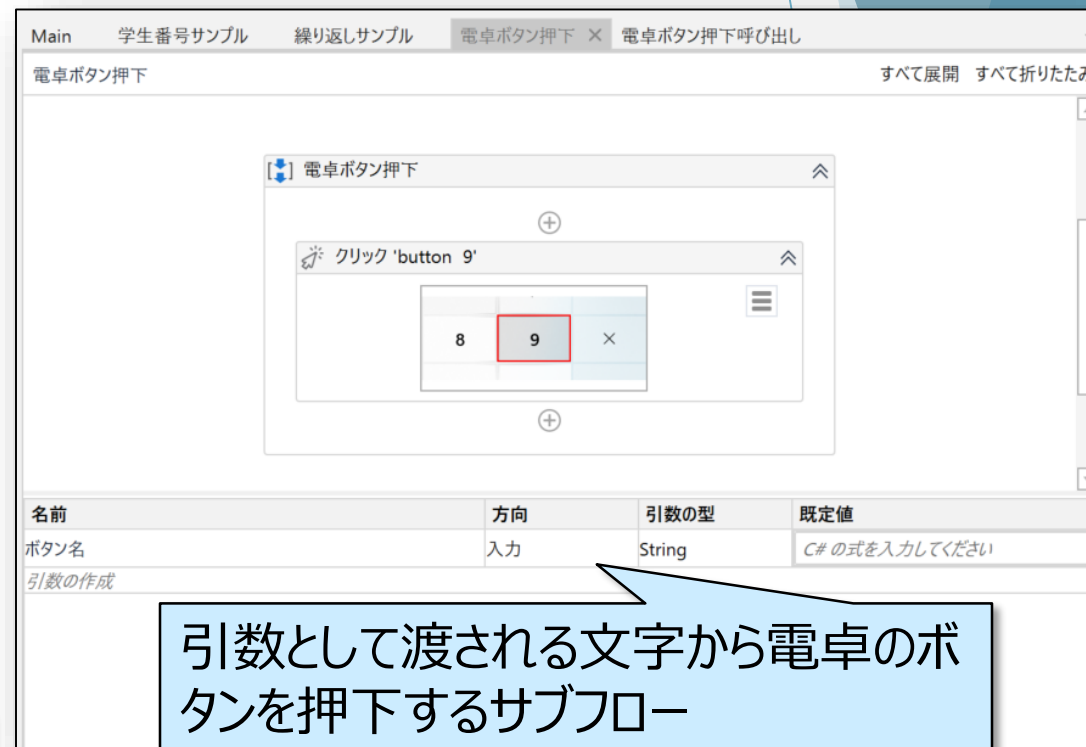
`new string[] { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" }`
曜日の英語略をログに順番に出力するサンプル。

フロー制御

サブフローサンプル



数字を指定するだけでボタンが押される



引数として渡される文字から電卓のボタンを押下するサブフロー

フロー制御

サブフローを利用するメリット

- ▶ 1つのワークフローで取り扱う問題を小さくする。
- ▶ 重複する処理を集約して、同一のワークフローとして部品化する。
- ▶ ワークフロー単位でテストができるようになる。
- ▶ チーム開発を容易にする。

フロー制御

練習時間

～～練習時間（10分間）～～

メモ帳に書かれている計算式を拾い、電卓に移して計算し、結果をまたメモ帳に転記するロボットを作ってください。

フロー制御

練習時間

～～練習時間（10分間）～～

ヒント 1 : 電卓の「+ - × ÷」ボタンのセクターを確認してください。

ヒント 2 : Stringを繰り返しアクティビティに入れたら、文字毎に繰り返される。

質問コーナー

宿題

- ▶ スライド「フロー制御：練習時間」の電卓転機ロボを完成してください。

次回予告

第三回目：高度なUiPath機能（1）（4/1（木） 19時）

- ▶ データテーブルとコレクション
- ▶ エラーハンドリング
- ▶ 並列とキャンセルスコープ
- ▶ C#でカスタマイズ処理を作ってみよう

次回からは皆さんの武器を増やす会です！
では、ご参加お待ちしております！！