**Variáveis, Constantes e Argumentos no Studio**

**Criando, usando e gerenciando variáveis**

Você pode imaginar variáveis como contêineres que contêm dados (valor) de um determinado tipo. O valor de uma variável pode mudar durante a execução do programa devido a uma entrada externa, manipulação de dados ou como resultado da passagem de uma atividade para outra. Em outras palavras, as variáveis armazenam dados dinamicamente.

As variáveis no Studio são configuradas por meio de quatro propriedades principais. São eles:

1. Name: É o atributo exclusivo usado para identificar uma variável. Este é um campo obrigatório. Se você não adicionar um nome a uma variável, um será gerado automaticamente. Os nomes das variáveis devem ser significativos e o mais descritivos possível. Embora não seja a única opção, recomendamos o uso da convenção de nomenclatura PascalCase. Por exemplo: ItemValue, LastName ou KeyItem.
2. Type: Ele define o tipo de dados armazenados em uma variável. Este é um campo obrigatório. Alguns dos tipos de dados comuns incluem:

* Booleano
* Int32
* String
* Object
* System.Data.DataTable
* Matriz de [T]

1. Scopo: Ele define o contexto no qual uma variável pode ser usada no projeto. O escopo é um campo obrigatório. Pode ser definido como o arquivo de fluxo de trabalho atual ou qualquer atividade de contêiner dentro do arquivo de fluxo de trabalho. Além disso, também pode ser definido como global, o que significa que elas são acessíveis a todas as atividades e fluxos de trabalho em um projeto de automação.
2. Default: É o valor padrão da variável. Este é um campo opcional. Se uma variável for declarada com esse campo vazio, um valor padrão correspondente ao tipo de dados da variável será atribuído a ela. Por exemplo, para um Int32, o valor padrão é 0.

Variáveis são locais de armazenamento que podem conter vários valores do mesmo tipo de dados. Podemos pensar em variáveis como uma caixa contendo informações que podemos acessar. Podemos sempre abrir esta caixa para verificar o que está dentro ou colocar algo mais dentro dela.

Por exemplo, em uma loja de varejo, as variáveis podem ser usadas para rastrear os níveis de estoque de diferentes produtos. Uma variável pode ser usada para armazenar o número atual de um produto específico em estoque. À medida que o produto é vendido, o valor da variável é atualizado para refletir o novo nível de estoque.

Para descrever com precisão o uso de variáveis em todo o projeto, devemos fornecer nomes significativos para variáveis. Fornecer nomes significativos é tão bom quanto rotular a caixa com base em seu conteúdo para facilitar a identificação.

Como os projetos de RPA podem ficar bastante complexos e muitas vezes são colaborativos, é uma boa prática dar nomes descritivos às atividades e adicionar anotações.

Vamos conferir as principais maneiras de criar uma nova variável no UiPath Studio:

* No painel Variáveis, adicionando uma nova linha, ..
* No painel Propriedades da atividade, pressione Ctrl + K em um campo de entrada que aceite variáveis.
* No corpo de uma Atividade, pressionando Ctrl + K em um campo de entrada que possa aceitar um nome de variável, ou clicando com o botão direito do mouse no campo de entrada e selecionando a opção criar variável.
* No Painel Gerenciador de Dados, clique em Nova Variável.
* E clicando com o botão direito do mouse na atividade e selecione Criar variável. Isso funciona em um escopo, ou seja, dentro de uma atividade de tipo de sequência.

Podemos selecionar um dos tipos mostrados no menu suspenso, mas também podemos clicar em Procurar e Selecionar um Tipo .Net na janela exibida.

Se houver duas variáveis com o mesmo nome, a variável definida no escopo mais interno estará no topo da lista de prioridades. O valor Default define um valor inicial para a variável quando o fluxo de trabalho é iniciado.

A variável aparece nos painéis Variáveis e Gerenciador de Dados. Uma atividade de atribuição sempre funciona da direita para a esquerda. Quando criamos uma variável usando esse método, o tipo de dados padrão é String e o Scope é definido como o contêiner que o inclui, no nosso caso, Assign Sequence.

Um fato interessante sobre o método Ctrl+K é que ele declarará uma variável do tipo necessário no campo TO. Por exemplo, se o tipo de dados necessário no campo for UIElement, esse será o tipo de dados que ele atribuirá à variável recém-criada. Caso contrário, o tipo de dados é uma cadeia de caracteres como no nosso caso.

Podemos alterar a mensagem da atividade Mensagem de log para uma concatenação das cadeias de caracteres nas duas variáveis. A concatenação é obtida usando os operadores VB.Net + ou "ampersand".

Práticas recomendadas ao trabalhar com variáveis:

* Atribuir nomes significativos: Nomes significativos devem ser atribuídos a variáveis para descrever com precisão seu uso em um projeto.
* Seguir uma convenção de nomenclatura: Para melhorar a legibilidade, os nomes de variáveis também devem estar alinhados a uma convenção de nomenclatura. Recomendamos o uso da convenção de nomenclatura PascalCase. Nessa convenção, a primeira letra de cada palavra em uma variável é maiúscula. Ex: First1Name2, First1Name.
* Mantenha as variáveis no escopo mais interno: As variáveis devem ser mantidas no escopo mais interno para reduzir a desordem no painel Variáveis e mostrar apenas o que é relevante em um ponto específico do fluxo de trabalho.

**Explorando Tipos de Dados**

**O que são tipos de dados?**

Como você pode adivinhar pelo nome, os tipos de dados descrevem o tipo de dados que uma variável pode conter. Por exemplo, se o tipo de dados for Int32, a variável deverá conter um inteiro. Da mesma forma, se o tipo de dados for um String, a variável deverá conter um texto.

**Por que os tipos de dados são importantes?**

Variáveis e argumentos no Studio precisam ter um tipo de dados específico definido. Eles garantem que o formato de dados correto seja aplicado em cada estágio de um processo automatizado. Os tipos de dados no UiPath são emprestados do Visual Basic.Net. Abaixo estão listados alguns dos mais usados.

1. String: System.String armazena textos. Esse tipo de dados vem com muitos métodos de manipulação de dados que serão explicados em detalhes em outra lição, ou seja, Manipulação de Dados com Strings no Studio.
2. Number: Essas variáveis armazenam números. Existem diferentes subtipos de variáveis numéricas. Por exemplo:

* Int32 - System.Int32 (inteiros assinados): 10, 299, -100
* Longo - System.Int64 (inteiros longos): 54354353430, -11332424D
* Duplo - System.Double (permite decimais, precisão de 15-16 dígitos)

1. Booleano: System.Boolean armazena um dos dois valores: True ou False.
2. Array: ArrayOf<T> ou System.DataType[] armazena vários valores do mesmo tipo de dados. O tamanho (número de objetos) é definido na criação.
3. Lista: System.Collections.Generic.List<T> armazena vários valores do mesmo tipo de dados, assim como Matrizes. Mas, ao contrário de Arrays, o tamanho é dinâmico.
4. Dictionary: System.Collections.Generic.Dictionary<TKey, TValue> armazena objetos na forma de pares (chave, valor), onde cada um dos dois pode ser um tipo de dados separado.
5. DataTable: DataTable representa variáveis que podem armazenar grandes informações e atuar como um banco de dados ou uma planilha simples com linhas e colunas.
6. DateTime: System.DateTime armazena coordenadas de hora específicas (mm/dd/aaaa hh:mm:ss). Esse tipo de variável fornece uma série de métodos de processamento específicos, como subtrair dias, calcular o tempo restante versus hoje, e assim por diante. Por exemplo, para obter a hora atual, atribua a expressão DateTime.Now a uma variável do tipo DateTime.
7. TimeSpan: System.TimeSpan armazena informações sobre uma duração (dd:hh:mm:ss). Você pode usá-lo para medir a duração entre duas variáveis do tipo DateTime. Por exemplo, você pode salvar a hora no início do processo em uma variável (do tipo DateTime), a hora no final em outra (do tipo DateTime) e armazenar a diferença em uma variável do tipo TimeSpan.

**Métodos de Conversão de tipos de dados**

Haverá cenários em que precisaremos alterar o tipo de dados de uma variável para outro formulário. Por exemplo, a conversão de um inteiro em uma cadeia de caracteres ou vice-versa. Nesses casos, podemos usar os métodos de conversão disponíveis no Studio. Abaixo está uma lista de alguns dos métodos de conversão mais comumente usados.

1. Método Convert.ToString: Esse método converte o valor especificado em sua representação de cadeia de caracteres equivalente. Por exemplo, de um inteiro para String. Ex: StrVar = Convert.Tostring(IntVar)
2. Método Convert.Int32: Esse método converte um valor especificado em um inteiro. Por exemplo, de um String para inteiro ou de um número de ponto flutuante para inteiro. Eg: IntVar = Convert.ToInt32(StrVar) / IntVar = Convert.ToInt32(DblVar)
3. Há outro método para converter uma cadeia de caracteres em um inteiro - CInt(String): Ex: IntVar = CInt(StrVar)
4. Método Double.ToString: Esse método converte o valor numérico de um número de ponto flutuante em sua representação de cadeia de caracteres equivalente. Ex: StrVar = DblVar.ToString
5. Método Double.Parse: Esse método converte a representação de cadeia de caracteres de um número em seu equivalente de número de ponto flutuante.Ex: DblVar = Double.Parse(StrVar)
6. Método Boolean.ToString: Esse método converte o valor de um Boolean em sua representação de cadeia de caracteres equivalente ("True" ou "False"). Ex: ToString()
7. Método Convert.ToBoolean: Esse método converte um valor especificado em um valor booleano equivalente. Por exemplo, cobiçar um inteiro para um valor booleano equivalente. Ex: Convert.ToBoolean(IntVar)
8. Converter data e hora em cadeia de caracteres: Esse método converte o valor do objeto DateTime especificado em sua representação de cadeia de caracteres equivalente. Ex: DateTimeVar.ToString("dd-MM-aaaa")

**Fluxo de Trabalho**

Um fluxo de trabalho representa uma parte relativamente pequena de um projeto de automação, normalmente executando uma parte específica do processo. Uma vez construído, ele pode ser reutilizado em diferentes projetos.

Um fluxo de trabalho é feito de atividades do Studio, interligadas através de variáveis para formar uma rotina. A rotina normalmente tem uma entrada e uma saída. Basicamente, ele define o fluxo de automação. Daí o nome, fluxo de trabalho. O UiPath Studio fornece layouts de fluxo de trabalho predefinidos para atender a todas as necessidades de um processo de automação rápido e confiável. Os layouts de fluxo de trabalho são:

* Sequências (abre em uma nova guia)
* Fluxogramas (abre em uma nova aba)
* Máquinas de Estado (abre em uma nova guia)(abre em uma nova guia)

A maneira mais rápida, confiável e útil de automatizar um processo é dividi-lo em bits menores. Isso permite o teste independente de componentes, permite a colaboração da equipe e a reutilização de componentes. Portanto, a maioria dos projetos de automação requer o uso de vários fluxos de trabalho que se unem para fornecer uma solução sólida de automação comercial.

**Argumentos**

Agora que você já sabe sobre fluxos de trabalho, vamos voltar a abordar a diferença entre uma variável e um argumento. Enquanto as variáveis passam dados entre atividades, os argumentos passam dados entre fluxos de trabalho.

Uma propriedade adicional associada aos argumentos é a direção. Os argumentos têm direções específicas: Entrada, Saída e Entrada/Saída. Isso diz ao robô para onde as informações armazenadas neles devem ir.

Dividir processos em fluxos de trabalho menores é fundamental para um bom design de projeto. Os fluxos de trabalho dedicados permitem testes independentes de componentes, ao mesmo tempo em que incentivam a colaboração da equipe trabalhando em arquivos separados ao mesmo tempo.

Como você acha que passamos dados entre os fluxos de trabalho pai e filho? Ao trabalhar com vários fluxos de trabalho, em vez de variáveis, usamos argumentos para passar dados entre os fluxos de trabalho pai e filho. Os argumentos passam dados entre fluxos de trabalho, enquanto as variáveis passam dados entre atividades. Algumas exceções se aplicam, sobre as quais falaremos um pouco mais adiante. É hora de começar a construir!

Um argumento é uma forma de fornecer mais informações. A direção de um argumento diz ao aplicativo para onde as informações armazenadas nele devem ir. Agora, há três direções que podemos escolher para um argumento: In, Out e In/Out. Os argumentos "In" podem ser usados apenas para passar dados para o fluxo de trabalho invocado, "out" apenas para extrair dados do fluxo de trabalho invocado e "in/out" funciona nos dois sentidos.

A opção "Propriedade" é usada para indicar que um argumento não está em uso no momento, sem ter que excluí-lo da lista. Os tipos que estão disponíveis para escolher para variáveis também estão disponíveis para argumentos. Uma direção de argumento deve sempre prefixar o nome do argumento. Ex: out\_soma / in\_FirstValue. Assim que criarmos argumentos no Painel de Argumentos, eles também aparecerão no Painel do Gerenciador de Dados.

Para invocar um fluxo de trabalho precisamos de uma atividade Invocar Fluxo de Trabalho para fazer isso. No entanto, podemos arrastar e soltar o fluxo de trabalho que queremos invocar do painel Projeto, tendo assim a atividade configurada automaticamente. Além disso, se clicarmos com o botão direito do mouse no nome do projeto, temos a opção de Importar arquivos e selecionar o arquivo .xaml que queremos usar.