

PREWORK SESIÓN 2

Objetivos

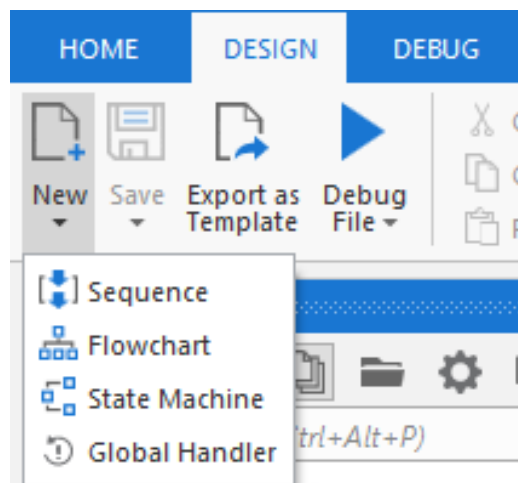
1. Conocer los diversos flujos de trabajo.
2. Comprender el uso e importancia del flujo de trabajo *Sequence*.
3. Conocer las variables y argumentos.
4. Aprender las buenas prácticas para las convenciones de nomenclaturas.

Desarrollo

Flujos de Trabajo

UiPath ofrece cuatro diagramas para integrar actividades en una estructura de trabajo al desarrollar un archivo de flujo de trabajo:

- *Sequence*
- *Flowchart*
- *State Machine*
- *Global Handler*



Dato importante: Todos los tipos de flujos de trabajo anteriores en UiPath se pueden usar en combinaciones, según sea necesario.

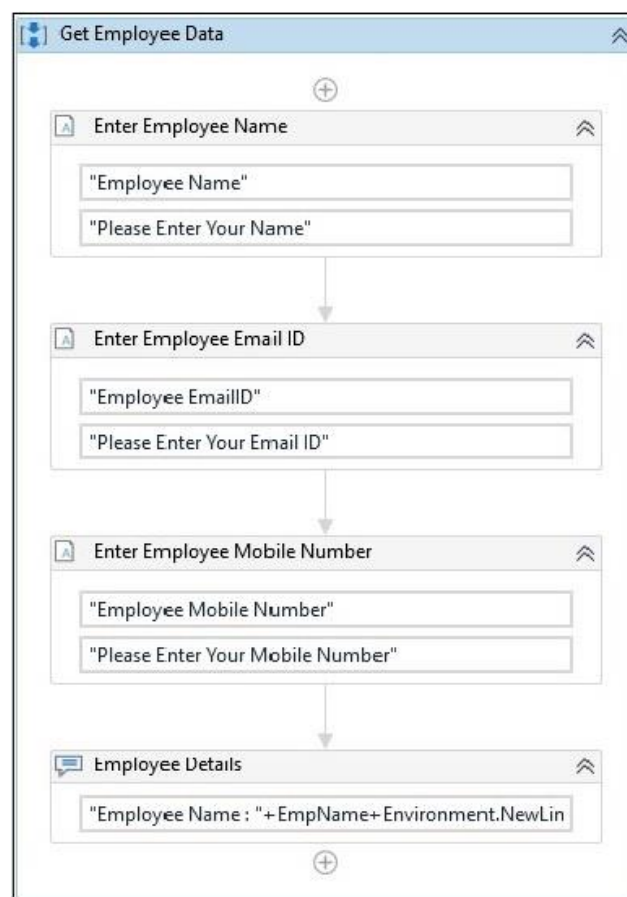
En la sesión 2 conoceremos más detalle sobre el diagrama *Sequence*.

Secuencia (*Sequence*)

Las secuencias son adecuadas para procesos lineales, ya que le permiten pasar de una actividad a otra sin problemas y actuar como una actividad de un solo bloque. Por ejemplo, son útiles en la automatización de la interfaz de usuario, cuando la navegación y la escritura ocurren con un clic / pulsación de tecla a la vez. Debido a que las secuencias son fáciles de ensamblar y comprender, son el diseño preferido para la mayoría de los flujos de trabajo.

Ejemplo de secuencia

A continuación, se muestra una secuencia que solicita diversos datos de un empleado (nombre, identificación de correo electrónico y número de teléfono móvil). Por último, muestre los datos obtenidos.



Dato importante: La razón para crear una secuencia aquí es que queremos que las actividades se ejecuten una tras otra y no implica estructuras de rama complejas o decisiones.

Variables y tipos de datos

Las variables son contenedores que pueden contener valores únicos o múltiples. Las variables se utilizan para almacenar información para ser referenciada y manipulada en un proyecto UiPath. El valor de una variable puede cambiar a través de entradas externas, manipulación de datos o pasar de una actividad a otra. Podemos configurarlas en el panel de Variables. Las principales propiedades de las variables son las siguientes:

- **Nombre:** Los nombres de las variables deben ser lo más descriptivos posible para que nuestra automatización sea fácil de leer por otros desarrolladores y para ahorrar tiempo.
- **Tipo:** Define qué tipo de datos se pueden almacenar en la variable. Por ejemplo, emailAddress puede ser una variable de tipo String que contiene el valor contact@intellectrpa.com.
- **Alcance:** El alcance de una variable significa parte del flujo de trabajo en el que se puede utilizar la variable.
- **Valor por defecto:** Podemos establecer un valor predeterminado o inicial para la variable, y que se puede manipular durante todo el proceso.

Name	Variable type	Scope	Default
variable1	String	Flowchart	"valor"
variable2	Int32	Flowchart	0
variable3	Boolean	Flowchart	True
variable4	DataTable	Flowchart	Enter a VB expression
variable5	Int32[]	Flowchart	{1,2,3,4,5}
Create Variable			
Variables Arguments Imports			

Algunos de los tipos de variables más utilizados en UiPath

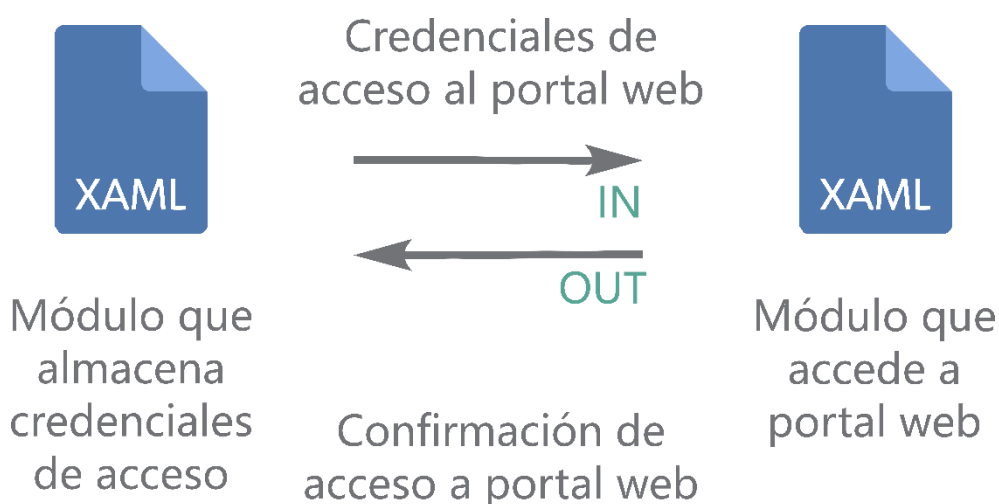
- **System.Int32, System.Int64, System.Double:** Para almacenar datos numéricos.
- **System.Boolean:** Para almacenar uno de dos valores (ya sea 'verdadero' o 'falso').
- **System.String:** Para almacenar datos textuales.
- **ArrayOf o System.DataType[]:** Para almacenar varios valores del mismo tipo de datos.
- **System.DateTime, System.TimeSpan:** Para almacenar coordenadas de tiempo específicas y duración respectivamente.
- **GenericValue:** Para almacenar cualquier tipo de datos, incluidos texto, números, fechas y matrices. Este tipo se utiliza principalmente en actividades en las que no estamos seguros de qué tipo de datos recibiremos, aunque en general el uso de estos es temporal.

Argumentos

Los argumentos se utilizan para pasar los datos dentro y entre los proyectos. También almacenan datos como variables, tienen los mismos tipos de datos y admiten los mismos métodos, pero no son exactamente iguales que las variables. La diferencia es que las variables pasan datos entre actividades mientras que los argumentos pasan datos entre proyectos o archivos.

Los argumentos tienen una propiedad adicional llamada **dirección** desde/hacia la cual se pasan los datos. La dirección puede ser Entrada (In), Salida (Out) o Entrada/Salida (In/Out).

Por ejemplo: En tu proyecto de automatización tienes un módulo que almacena un conjunto de credenciales de acceso, entre ellas las credenciales de acceso a un portal web. Tienes otro módulo, este te permite acceder a un portal web, pero requiere las credenciales de acceso. Es por ello que recibe como **argumento de entrada** las credenciales, las utiliza y puede o no ser exitoso el acceso al portal web, pero el resultado lo comparte como **argumento de salida**.







Las principales propiedades de los argumentos son las siguientes:


- **Nombre:** Los nombres de las variables deben ser lo más descriptivos posible para que nuestra automatización sea fácil de leer por otros desarrolladores y para ahorrar tiempo.
- **Tipo:** Define qué tipo de datos se pueden almacenar en el argumento. Por ejemplo, emailAddress puede ser un argumento de tipo String que contiene el valor contact@intellectrpa.com.

- Valor por defecto:** Podemos establecer un valor predeterminado o inicial para el argumento, y que se puede manipular durante todo el proceso.

Name	Direction	Argument type	Default value
argument1	In	String	<i>Enter a VB expression</i>
argument2	Out	Int32	<i>Default value not supported</i>
argument3	In/Out	String	<i>Default value not supported</i>
<i>Create Argument</i>			

Variables Arguments Imports



 108.27%
 



Información complementaria: Existen varias recomendaciones para nombrar una variable o un argumento. Te comparto este documento que detalla las buenas prácticas para las convenciones de nomenclatura: [Descargar archivo](#).