Formación Profesional: RPA con UiPath
Guía de Práctica – Segundo Ejercicio

Tema: Extracción de Datos Semi-Estructurados

1. Objetivo Técnico

El objetivo de esta práctica es implementar un proceso de web scraping para extraer y parsear datos de múltiples elementos de una página web. Se extrae datos de contenedores de UI (tarjetas de producto) donde la información es semi-estructurada.

El proceso final leerá todos los productos de una categoría, extraerá un bloque de texto de cada uno, se obtendrán datos de interés (nombre, precio, descripción) y finalmente, estructurará estos datos en un formato tabular para su posterior análisis.

2. Arquitectura del Proceso

El algoritmo se implementará siguiendo estos pasos lógicos:

1. **Inicialización:** Definir la estructura de datos de salida (una DataTable) que almacenará los resultados.
2. **Identificación de Elementos:** Utilizar Find Children para obtener una colección de todos los elementos de UI que representan las tarjetas de producto en la página.
3. **Iteración y Procesamiento:** Recorrer en un bucle la colección de elementos obtenida. Dentro de cada iteración:
   a. Extraer el texto completo del elemento actual.
   b. Obtener los datos de interés descripción y precio.
   c. Poblar una nueva fila en la DataTable con los datos extraídos.
4. **Almacenamiento de Resultados:** Escribir el contenido de la DataTable a un archivo Excel.

3. Implementación Paso a Paso

Sitio Web: https://webscraper.io/test-sites/e-commerce/allinone/computers/laptops

Paso 3.1: Inicialización de la DataTable

1. Crea un nuevo Proceso.
2. Utiliza la actividad **Build Data Table** para crear la estructura de salida.
3. Define las columnas: Descripcion (String), Precio (String).
4. Guarda la salida en una variable DataTable llamada dt_Productos.

Paso 3.2: Identificación de las Tarjetas de Producto

1. Arrastra una actividad **Use Application/Browser** y apunta a la URL objetivo.

2. Dentro, añade la actividad **Find Children**.
3. Indica como objetivo el contenedor que agrupa todas las tarjetas de productos.
4. En el panel de Propiedades:
    - **Filter:** Configura un selector que identifique de forma unívoca las tarjetas. Por ejemplo: <webctrl tag='DIV' class='card-body' />. Esto puede requerir el uso de UI Explorer para refinarlo.
    - **Children:** Crea una variable para almacenar la lista de tarjetas tipo IEnumerable<UiElement>.

Paso 3.3: Iteración

1. Arrastra un bucle **For Each** después del Find Children.
2. En el campo in, introduce la variable creada.
3. **Importante:** Cambia el TypeArgument del bucle a UiPath.Core.UiElement. *Si se prefiere renombrar al ítem de la iteración*.

Paso 3.4. Lógica de Procesamiento Dentro del Bucle

Dentro del bucle For Each, para cada elemento UI (tarjetaActual) que representa una tarjeta de producto, se implementará la siguiente secuencia de procesamiento:

3.4.1. Extracción del Bloque de Texto

La primera operación es obtener todo el contenido textual del elemento UI actual en su formato crudo.

1. Arrastre una actividad **Get Text**.
2. El objetivo (Input element) de esta actividad debe ser la variable de la iteración actual del bucle (ej. tarjetaActual).
3. Guarde el resultado en una nueva variable de tipo String que siga la convención de nomenclatura (ej. str_textoCrudo, str_tarjetaActual).

3.4.2. División (Split) y Limpieza del Texto en un Array

Una vez obtenido el texto crudo, el siguiente paso es dividirlo en líneas individuales y limpiar los datos para eliminar inconsistencias como espacios en blanco o líneas vacías.

1. Arrastre una actividad **Assign**.
2. En el campo Save to, cree una nueva variable de tipo Array de String (String[]). Siga la convención de nomenclatura (ej. list_LineasDeTexto, list_TextoParseado).
3. En el campo Value to save, introduzca la siguiente expresión. Esta expresión es robusta y realiza varias operaciones de limpieza en una sola línea:

code Vb.net

---

variableStringDeTextoCrudo.Split({vbCr, vbLf}, StringSplitOptions.RemoveEmptyEntries).Where(Function(x) Not String.IsNullOrWhiteSpace(x)).Select(Function(x) x.Trim).ToArray()

(Reemplace variableStringDeTextoCrudo con el nombre de la variable que utilizó en el paso 3.4.1)

Análisis de la Expresión:

- **.Split({vbCr, vbLf}, StringSplitOptions.RemoveEmptyEntries)**: Divide el string en un array, utilizando tanto el retorno de carro (vbCr) como el salto de línea (vbLf) como delimitadores. La opción RemoveEmptyEntries elimina cualquier línea que quede completamente vacía tras la división.
- **.Where(Function(x) Not String.IsNullOrWhiteSpace(x))**: Es un filtro adicional y más potente. Elimina cualquier línea que, aunque no esté vacía, contenga únicamente espacios en blanco.
- **.Select(Function(x) x.Trim)**: Itera sobre cada línea restante en el array y aplica la función Trim, eliminando cualquier espacio en blanco al principio y al final de cada línea.
- **.ToArray()**: Convierte el resultado final de la consulta LINQ de nuevo a un formato de Array de String.

3.4.3. Asignación de Datos Específicos desde el Array

Con el array limpio y estructurado, se procede a asignar los valores específicos a variables, accediendo a ellos mediante su índice. ***Recordatorio:*** *Los arrays en .NET son de base cero (el primer elemento es el índice 0).*

1. Arrastre una actividad **Assign** para el precio.
    - Save to: Cree una nueva variable String para el precio (ej. str_Precio).
    - Value to save: variableArrayDeLineas(0). Se asume que el precio es el primer elemento de la lista.
2. Arrastre otra actividad **Assign** para la descripción.
    - Save to: Cree una nueva variable String para la descripción (ej. str_Descripcion).
    - Value to save: variableArrayDeLineas(2). Se asume que la descripción es el tercer elemento de la lista.

Nota de Ingeniería: Los índices (0, 2, etc.) dependen directamente de la estructura del texto en las tarjetas de la página web. Si la estructura de la web cambia, estos índices podrían

necesitar ser re-evaluados. *Es fundamental analizar el contenido de la variable array durante la depuración para validar que los índices son correctos.*

3.4.4. Población de la DataTable

Finalmente, los datos extraídos y asignados se añaden como una nueva fila a la DataTable que fue inicializada al principio del proceso.

1. Arrastre una actividad **Add Data Row**.
2. Configure las propiedades:
    - DataTable: Indique la variable DataTable creada en el paso 3.1 (ej. dt_Productos).
    - ArrayRow: Introduzca las variables en un array, **en el orden exacto en que definió las columnas**.
      {str_Descripcion, str_Precio}

Recordatorio: La estructura de la fila en ArrayRow ({variable1, variable2}) debe coincidir exactamente en orden y número con las columnas definidas en la actividad Build Data Table.

3.5. Almacenamiento Final en Formato Excel.

Una vez que el bucle For Each ha iterado y procesado todas las tarjetas de producto, el paso final es serializar la DataTable en memoria a un archivo de libro de Excel (.xlsx). Para esta operación, se utilizará la actividad Write Range Workbook, que forma parte del paquete de actividades de Excel y tiene la ventaja de no requerir que Microsoft Excel esté instalado en la máquina de ejecución.

1. Fuera y después del bucle For Each, arrastre la actividad Write Range Workbook.
2. Configure los parámetros de la actividad de la siguiente manera:
    - Primer campo (Workbook path / Ruta del libro): Especifique la ruta y el nombre del archivo de salida. El nombre debe incluir la extensión .xlsx.
        - Ejemplo: "Precios_Allinone.xlsx"
    - Segundo campo (Sheet name / Nombre de la hoja): Indique el nombre de la hoja de cálculo dentro del libro de Excel donde se escribirán los datos. Si la hoja no existe, la actividad la creará automáticamente.
        - Ejemplo: "Laptop"
    - Tercer campo (DataTable / Tabla de datos): Indique la variable DataTable que contiene todos los datos procesados y que se escribirá en el archivo.
        - Ejemplo: dt_Productos
3. Configuración de Propiedades Adicionales (Panel de Propiedades):

- Seleccione la actividad Write Range Workbook y diríjase al panel de "Propiedades".
- Asegúrese de que la opción AddHeaders (Añadir encabezados) esté activada. *Si esta opción está desactivada, solo se escribirán los datos, sin el contexto de los encabezados.*

Análisis de la Actividad:
La actividad Write Range Workbook interactúa directamente con los archivos .xlsx a nivel de archivo (utilizando la librería Open XML SDK), lo que la hace más rápida y eficiente para operaciones de escritura que no requieren la manipulación de fórmulas complejas o macros. Es la opción preferida para la generación de reportes en entornos de servidor o en robots desatendidos donde la instalación de la suite de Office no es viable o deseable.

4.Actividad Práctica

Modificar el robot existente para que sea capaz de extraer la información de las categorías Laptops, Tablets y Teléfonos en una sola ejecución.

- El resultado final debe ser un **único archivo Excel**
- Cada categoría de producto debe tener su propia hoja de cálculo (Sheet) dentro del libro de Excel. El nombre de cada hoja debe corresponder al nombre de la categoría que contiene (ej. una hoja llamada "Tablets" con los datos de las tablets).