Modelo genérico de web scraping que facilite la integración, visualización y normalización de datos inmobiliarios en EE. UU

Daniel Felipe Bravo Calvache
Luis Fernando Gómez Gómez

Director: PhD. Carlos Alberto Cobos Lozada

Anexo B - Lenguaje de instrucciones para acceso a la información de las páginas web

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Área de interés en Gestión de la Información
Popayán, octubre de 2021

LISTA DE FIGURAS

Lenguaje de instrucciones

La obtención de la información inmobiliaria de las páginas de real state con la técnica de web scraping se realizó por medio del framework de Python Selenium. Sin embargo, la extracción de los datos inmobiliarios en las páginas usando Selenium implican tener conocimientos relativamente avanzados tanto en programación (en el lenguaje python) como conocer las funciones y herramientas específicas propias del framework, lo que puede llegar a ser una tarea más dispendiosa y limita la cantidad de usuarios que pueden hacer uso del framework y que no tengan mucho conocimiento sobre programación.

Por esta razón, se hizo necesaria la creación de un lenguaje de instrucciones para realizar web scraping, que traduce sentencias más simples y sintácticamente más entendibles, que, por medio de un procesador de instrucciones, traduce las sentencias simplificadas en funciones de Selenium que se ejecutan y generan el resultado indicado en las instrucciones. Este lenguaje a su vez se divide en dos tipos de instrucciones: Un tipo de instrucciones diseñado específicamente para realizar acciones sobre los elementos de la página web, como por ejemplo hacer click sobre el botón o ingresar texto sobre un campo de entrada de algún formulario, y otro tipo de instrucciones para recorrer la estructura html de la página y sacar los datos presentados en ella.

El procesador de lenguaje de instrucciones para realizar acciones sobre la página web se le denominó FolioAccessProcessor, porque por medio de estas acciones se pueden realizar las búsquedas y consultas en las páginas web para que muestren la información de un registro de propiedad específico.

El procesador de instrucciones para acceder al contenido de los elementos html dentro de una página web se le denominó PathProcessor, el cuál obtiene la referencia a un objeto indicado dentro de una página, para luego interactuar con él o extraer su contenido.

La estructura de las instrucciones se define como una secuencia de sentencias separadas por el carácter slash (‘/’), y que dentro de cada una se pueden recibir de cero a muchos argumentos separados por dos puntos (‘:’), dependiendo del tipo de instrucción. Cuando una instrucción no es la última dentro de la secuencia, el resultado de ésta termina siendo el punto de partida para la siguiente instrucción.

IntruccionUno:argumentoUno:argumentoDos/instruccionDos:argumentoUno/InstruccionFinal

En el ejemplo anterior, la instrucción uno que recibe dos argumentos, da como resultado haber encontrado un elemento dentro de la página. La referencia a este elemento encontrado es la base de la instrucción dos, la cuál empieza a buscar un nuevo elemento a partir del encontrado en la primera instrucción. La última instrucción obtiene el resultado final de toda la secuencia.

Instrucciones del PathProcessor

Este procesador de instrucciones se encarga de recibir una secuencia de instrucciones para cada dato que se quiera encontrar y obtener en la página web especificada en la petición del scraper.

findById:idElemento

Esta función se encarga de buscar un elemento dentro de la página web o elemento web que coincida con el valor especificado dentro de su atributo id del elemento html.

Ejemplo:

```
findById:property_info
```

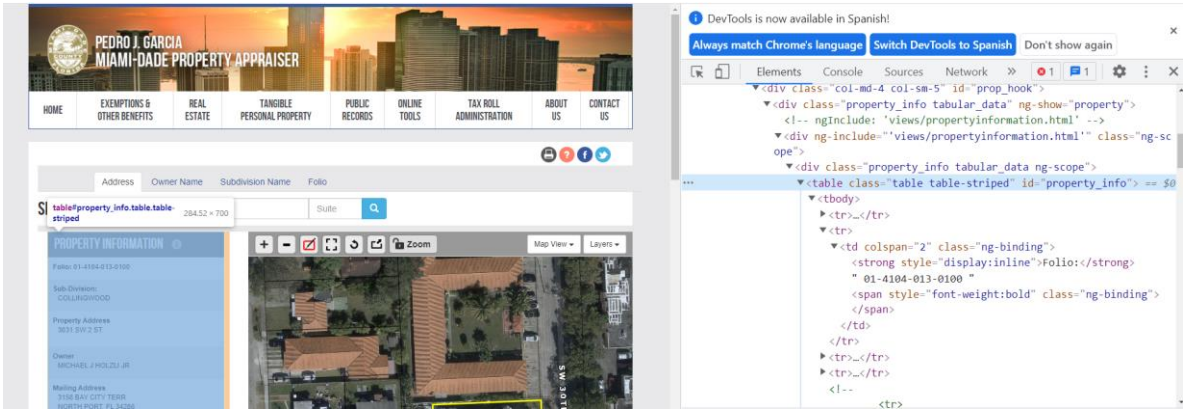

Figura 1: Ejemplo de instrucción findById

Resultado: referencia al elemento table que tiene como atributo id el valor “property_info”

findByString:textoEnElemento

Devuelve el elemento cuyo objeto html tiene dentro de todo su contenido el texto especificado.

Ejemplo:

```
findByString:Property Address
```

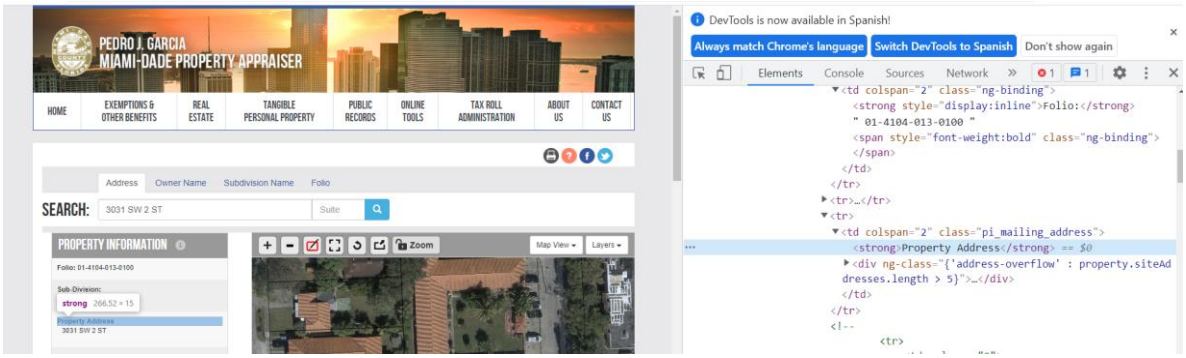

Figura 2: Ejemplo de instrucción findByString

Resultado: referencia al elemento que contiene el texto “Property Address”, en este caso un elemento .

findByClassName:nombreClaseCss

Encuentra el primer elemento html cuyo atributo class sea igual al nombre de clase especificado.

Ejemplo:

```
findByClassName:ng-binding
```

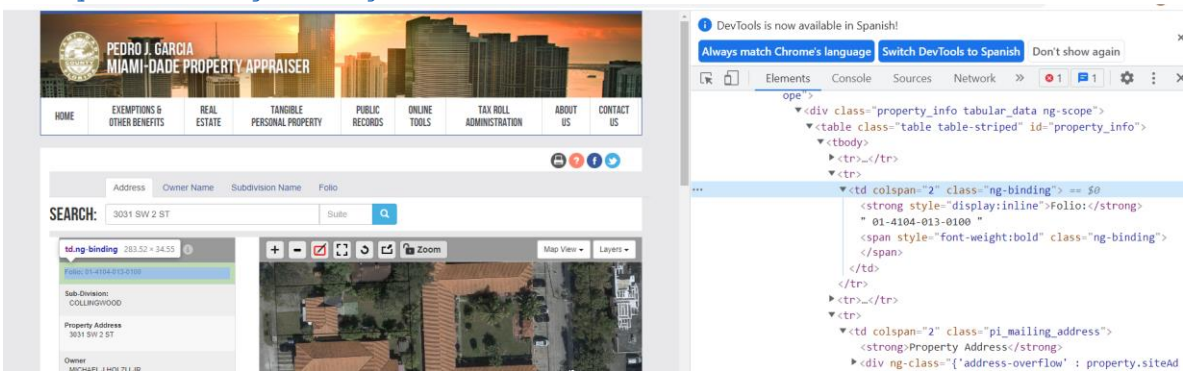

Figura 3: Ejemplo de instrucción findByClassName

Resultado: referencia a un elemento <td> cuyo atributo clase contiene “ng-binding”.

findAllByClassName:nombreClaseCss

Similar a la instrucción findByClassName, con la diferencia que el resultado de esta instrucción no es la referencia a un único elemento sino a una lista de elementos que correspondan con el nombre de clase especificado.

findByStringExact:textoEnelemento

Encuentra el primer elemento html que contenga exactamente el texto especificado, ni más ni menos caracteres.

Ejemplo:

```
findByStringExact:PA Primary Zone
```

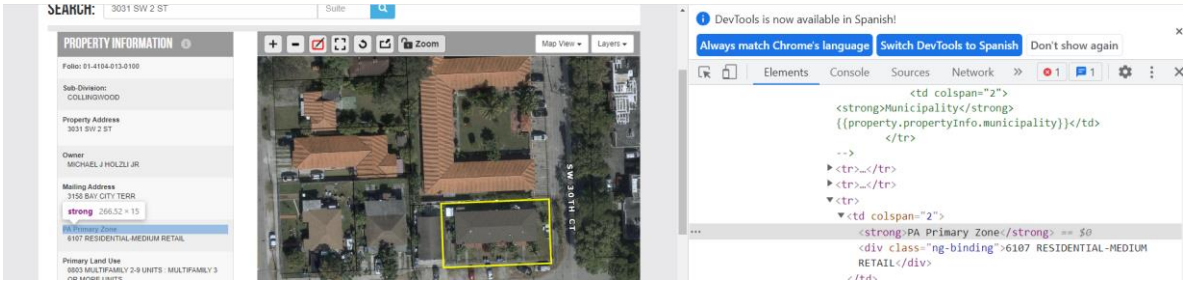

Figura 4: Ejemplo de instrucción findByStringExact

Resultado: referencia a un elemento que contiene exactamente el texto “PA Primary Zone”.

findByValue:valor

Encuentra el primer elemento html que contenga en el atributo value el valor especificado.

Ejemplo:

```
findByValue:Search
```

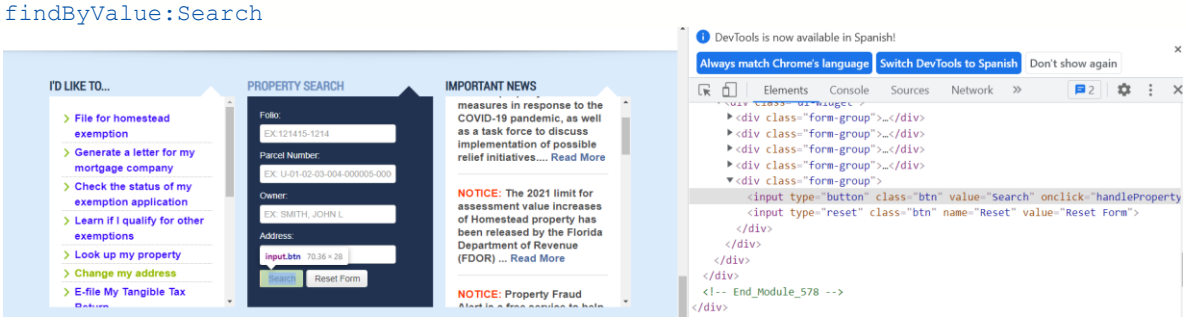

Figura 5: Ejemplo de instrucción findByValue

Resultado: referencia a un botón de búsqueda cuyo atributo value es “Search”

findNextSibling:nombreEtiqueta

Encuentra el primer elemento hermano o consecutivo (del mismo nivel) a partir del elemento base que corresponda al nombre de etiqueta especificado.

Ejemplo:

`findByString:Folio/findNextSibling:td`

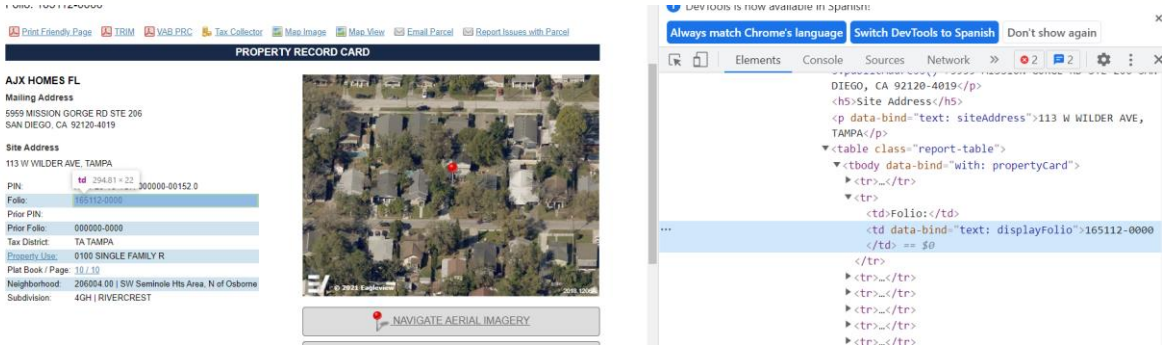

Figura 6: Ejemplo de instrucción findNextSibling

Resultado: inicialmente se indica que se obtenga un elemento que contenga el texto “Folio”. A partir de este, se le indica que obtenga el primer elemento hermano o del mismo nivel jerárquico cuya etiqueta sea <td>.

parent

Encuentra el elemento padre o de nivel jerárquico superior del elemento actual.

Ejemplo:

`findByString:Property Use/parent`

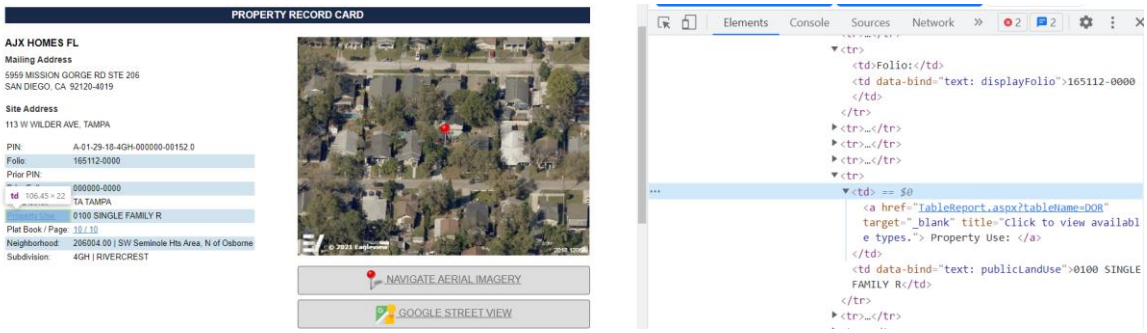

Figura 7: Ejemplo de instrucción parent

Resultado: Inicialmente se obtiene el elemento cuyo contenido sea el texto “Property Use”, esto da como resultado el elemento con la etiqueta <a>, a partir de este elemento se obtiene otro de orden jerárquico inmediatamente superior, es decir el elemento que lo contiene, que en este caso es un <td>.

findByTagName:nombreElemento

Encuentra el elemento que corresponda a la etiqueta especificada a partir de la página o elemento base.

Ejemplo:

```
findByClassName:propcard-section/findByTagName:h4
```

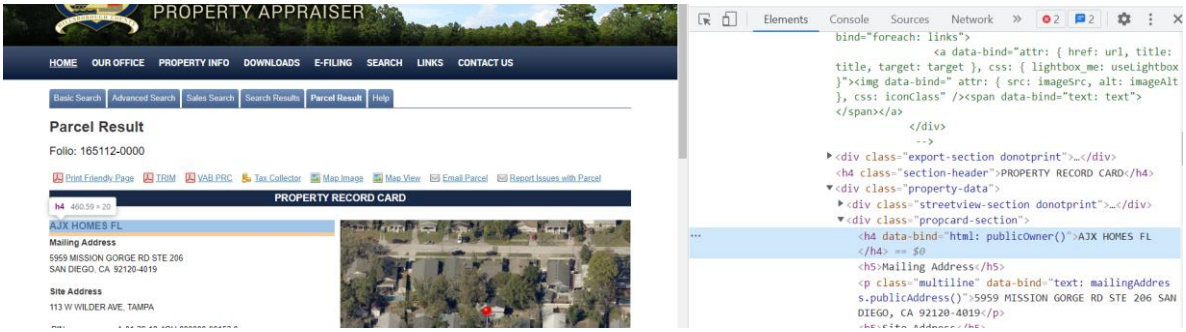

Figura 8: Ejemplo de instrucción findByTagName

Resultado: Con la primera instrucción se obtiene un elemento con nombre de clase “propcard-section”, y a partir de este elemento se obtiene el primer elemento <h4> contenido dentro del anterior elemento.

findChildAtPosition:posición

Encuentra el elemento en la posición indicada de la lista de elementos html

Ejemplo:

```
findByStringExact:Total/parent/findChildAtPosition:3
```

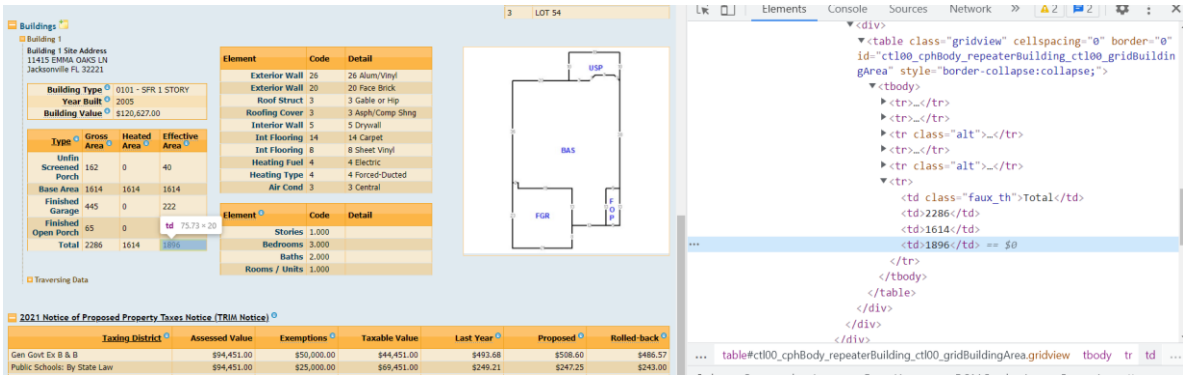

Figura 9: Ejemplo de instrucción findChildAtPosition

Resultado: En la primera instrucción se encuentra un elemento que contiene el texto “Total” que resulta ser un <td> con clase “faux_th”. A partir de este se obtiene el

elemento padre, el cuál es un <tr>. A partir de este último se indica encontrar el elemento hijo en la tercera posición, el cual contiene “1896”.

findAll:nombreEtiqueta:posición

Encuentra todos los elementos con el nombre de etiqueta especificado. Recibe opcionalmente un argumento de posición en caso de que se quiera obtener un elemento especifico de la lista de elementos.

Ejemplo:

```
findByStringExact:Total/parent/findAll:td
```

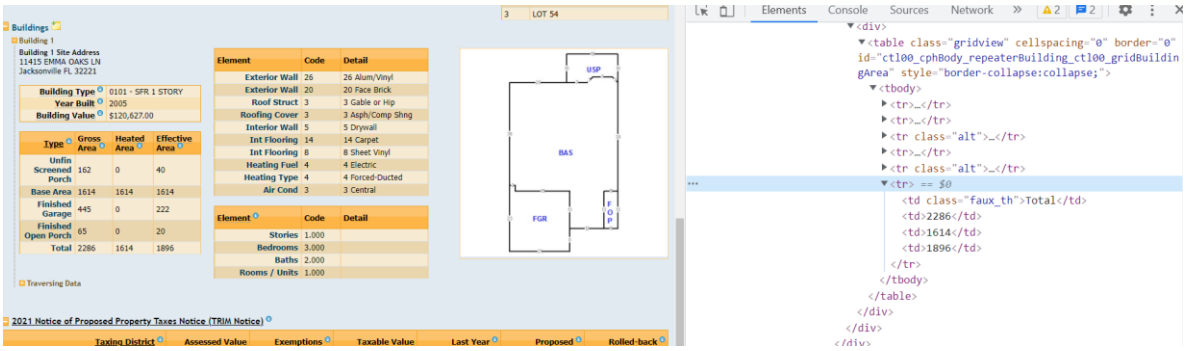

Figura 10: Ejemplo de instrucción findAll

Resultado: Tomando como referencia el ejemplo anterior, una vez obtenida la referencia al elemento padre <tr>, se indica que se obtengan todos los elementos <td> dentro del elemento padre. El resultado es una lista de referencias a cada uno de esos elementos.

getString

Devuelve la cadena de caracteres contenida dentro del elemento html encontrado en la instrucción inmediatamente anterior.

Ejemplo:

```
findByStringExact:PA Primary Zone/findNextSibling:div/getString
```

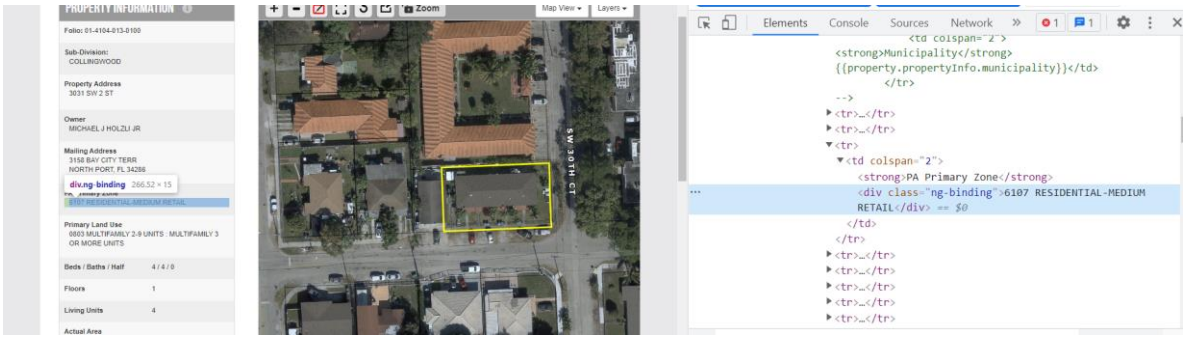

Figura 11: Ejemplo de instrucción getString

Resultado: Inicialmente se obtiene el elemento con el texto “PA Primary Zone”, posteriormente se obtiene el elemento hermano cuyo nombre de etiqueta es . A este último se le extrae el contenido o texto. El resultado final de esta secuencia de instrucciones es: “6107 RESIDENTIAL-MEDIUM RETAIL”.

getStrings

Devuelve la cadena de caracteres contenida dentro de una lista de elementos html encontrados en la instrucción inmediatamente anterior. Se usa conjuntamente con las instrucciones findAll o findByClassName que pueden devolver una lista de elementos.

Ejemplo:

```
findById:property_info/findByString:Owner/parent/findByClassName:ng-binding/getStrings
```

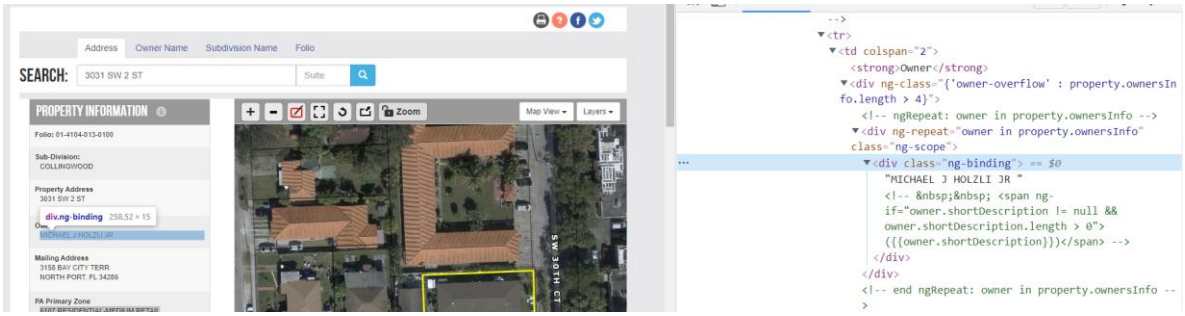

Figura 12: Ejemplo de instrucción getStrings

Resultado: al final de esta secuencia de instrucciones, se extrae el texto contenido en cada uno de los elementos encontrados con la instrucción findByClassName. Debido a que solo se encontró un elemento, el resultado es una lista de un solo elemento: [“MICHAEL J HOLZLI JR”]

splitString:caracterDeDivision:posición

Toma el texto extraído de un elemento, lo divide por el carácter especificado y de esa división devuelve el elemento en la posición especificada. Esta función es útil cuando varios atributos de información se encuentran dispuestos en una misma línea de texto dentro de la página, como se muestra en la siguiente imagen:

Figura 13: Ejemplo de varios datos en una misma línea de texto

El parámetro de carácter de división solo puede reconocer los siguientes valores:

slash: Para dividir la cadena por el carácter “/”

middle_dash: Para dividir la cadena por el carácter “-”

back_slash: Para dividir la cadena por el carácter “\”

white_space: Para dividir la cadena por el carácter “ ”

línea_break: Para dividir la cadena por salto de línea.

Ejemplo:

```
findById:property_info/findByString: Baths /parent/parent/findByClassName:ng-binding/splitString:slash:0
```

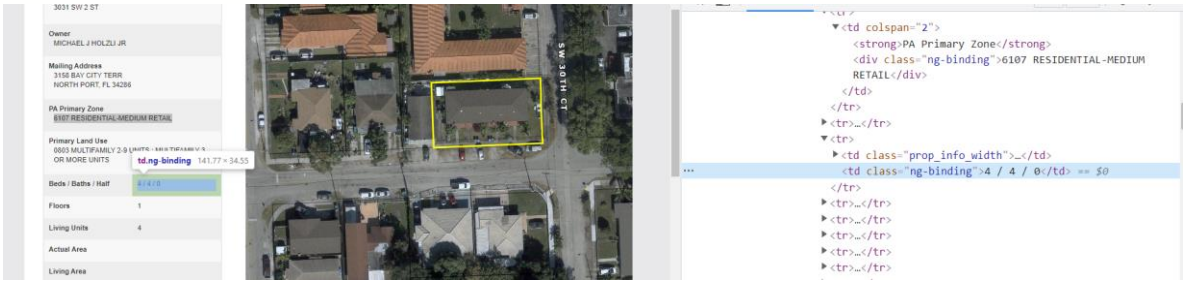

Figura 14: Ejemplo de instrucción splitString

Resultado: En la penúltima instrucción de esta secuencia, se obtiene una referencia al elemento <td> con clase “ng-bindig”. Finalmente, el texto contenido en este elemento se divide por el carácter slash (“/”), quedando separado en un vector con los elementos [“4”,”4”,”0”], de entre los cuales se pide extraer el que ocupa la posición número cero. El resultado final de esta secuencia de instrucciones es “4”.

FolioAcessProcessor

El lenguaje para el acceso a los registros específicos de las propiedades hace uso de la mayoría de las funciones del procesador de rutas o PathProcesor, pero extiende su funcionalidad al agregar nuevos argumentos a esas mismas instrucciones para producir interacción con los elementos de las páginas web.

openInNewTab

Intenta abrir el último elemento encontrado en caso de que sea posible abrirlo en una nueva pestaña. Se usa en el atributo “iterate” del objeto de la petición al scraper.

click

Esta función se usa como argumento final dentro de una función destinada a obtener un elemento html clickeable. Ejecuta todas las acciones de click asociadas al elemento.

Ejemplo:

```
findById:search_submit:click
```

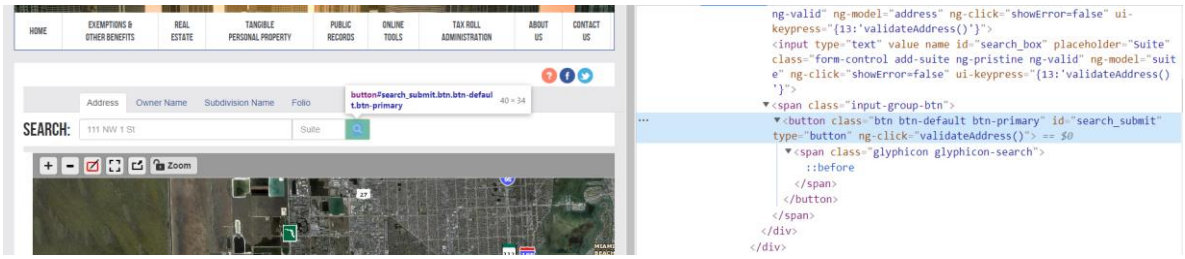

Figura 15: Ejemplo de instrucción click

Resultado: Se obtiene elemento cuyo id es “search_submit”, que en este caso es un botón, y sobre él se hace un click, lo que dirige a una pantalla de resultados de búsqueda.

clear

Este argumento se usa en elementos html que funcionan como campos de entrada. Elimina el contenido escrito dentro de la caja de texto.

putText

El funcionamiento de esta instrucción/argumento depende del contenido del atributo “address”, que tiene los datos completos y seccionados de las direcciones

de casas dentro del objeto de la petición. El atributo Address presenta la siguiente estructura:

```
"addresses": [
      {
         "unit":"",
         "number":"3031",
         "street":" SW 2 ST",
         "fullAddress":"3031 SW 2 ST"
      }
]
```

Este argumento a recibe su vez un nuevo argumento, una cadena de texto que se obtiene del objeto “addresses” que se insertará sobre el elemento html encontrado.

Ejemplo:

```
findById:search_box:clear:putText:fullAddress
```

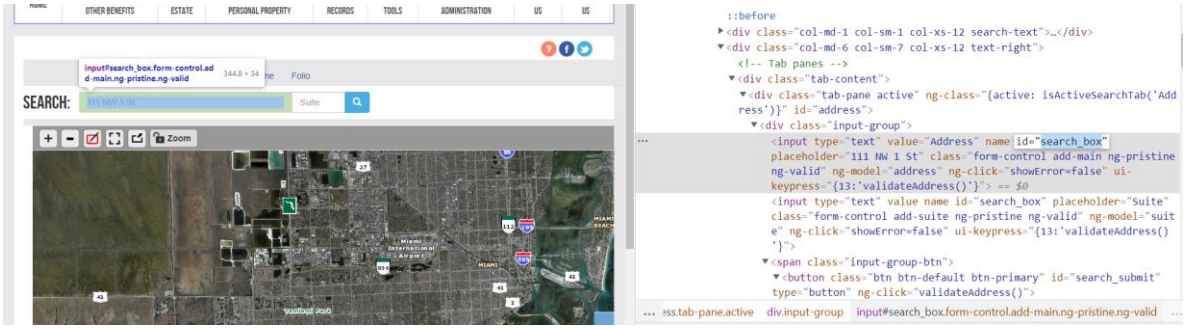

Figura 16: Ejemplo de instrucción putText

Resultado: Se encuentra el elemento <input> identificado como “search_box”, se limpia cualquier contenido que haya podido tener, y se inserta sobre él la cadena de caracteres que representa el argumento fullAddress, es decir “3031 SW 2 ST”.

Estructura de las peticiones HTTP para la API del scraper

El scraper atiende las solicitudes de scraping mediante un servicio REST implementado en el microframework Flask. El cuerpo de las peticiones se define de la siguiente forma para cada uno de los tipos de scraper:

```
{
   "city":"nombre-ciudad (opcional)",
   "url":"url-pagina-web",
   "data_access_route":[
       "instrucion1",
```

```
        "instruccion2",
        "instruccion3", ...
        ],
    "routes":[
        {
            "atributo1":"secuenciaDeInstrucciones1",
            "atributo2":"secuenciaDeInstrucciones2", ...
        }
    ],
    "addresses":[
       {
          "unit":"unidad",
          "number":"numero",
          "street":" calle",
          "fullAddress":"direccionCompleta"
       }
    ]
}
```

Para el caso de los scrapers de Taxes y Documents, el cuerpo de la petición recibe dos atributos adicionales, uno que guarda instrucción para recorrer una estructura repetitiva como una tabla. El otro atributo guarda los parámetros de búsqueda que generalmente son obtenidos al realizar scraping en las páginas de información general.

```
"iterate": "instruccionARepetir",

"search_parameters": [
        {
            "propertyId": "numeroPropiedad",
            "owner_name": "nombreDueño",
            "address": "direccionPropiedad"
        }
    ]
```

url: En este atributo se encuentra la dirección web a la que el scraper accederá para empezar a ejecutar las instrucciones de los otros argumentos.

data_access_route: Este atributo recibe un arreglo de instrucciones que generalmente es ejecutado por el FolioAccessProcessor ya que en él se especifican la serie de acciones a realizar para acceder al registro que contiene la información a extraer.

routes: Una vez ejecutadas las instrucciones encontradas en el atributo data_access_route, el scraper empieza a ejecutar esta lista de instrucciones en las

que al final se devuelven los valores obtenidos para cada uno de los atributos correspondientes a las instrucciones.

addresses: Este atributo es fundamental para realizar las búsquedas ya que en él está representada la estructura de las direcciones de las propiedades obtenidas desde el servicio de Google Maps o cualquier otra fuente de direcciones. El FolioAccessProcessor usa esta información para insertarla en los cuadros de búsqueda.

iterate: Esta instrucción es usada en los scrapers de Taxes y Documents debido a que los resultados encontrados en estas páginas generalmente son presentados en tablas de links que demuestran el histórico de registros que tienen, por lo que se debe abrir uno por uno cada uno de estos elementos y obtener la información específica que de se encuentra dentro de ellos.

search_parameters: En este atributo se encuentran los datos de búsqueda para que requieran las páginas de impuestos y documentos que son obtenidos inicialmente al hacer scraping sobre las páginas de información general.

Ejemplos de cuerpos de petición para realizar web scraping

Para los tres scrapers la estructura de la petición es prácticamente la misma, salvo por el scraper de información general que no requiere de los atributos iterate y search_parameters.

Petición para scraper de información general en página de Miami:


```
{
   "city":"Miami",
   "url":"https://www.miamidade.gov/Apps/PA/propertysearch/#/",
   "data_access_route":[
       "findById:search_box:clear:putText:fullAddress",
       "findById:search_submit:click",
       "findById:property_info"
       ],
   "routes":[

       {
           "par_parcel_id":"findById:property_info/findByString:Folio/parent/splitString:white_space:1"
       },
       {
           "own_name":"findById:property_info/findByString:Owner/parent/findByClassName:ng-binding/getStrings"
       },
       {
           "add_address":"findById:property_info/findByString:Property Address/parent/findAllByClassName:inline:getStrings"
       },
       {
           "par_use_type":"findById:property_info/findByString:Primary Land Use/parent/findByClassName:ng-binding/getString"
       },
       {
           "par_floors":"findById:property_info/findByString:Floors/parent/parent/findByClassName:ng-binding/getString"
       },
       {
           "par_lot_size":"findById:property_info/findByString:Lot Size/parent/parent/findByClassName:ng-binding/getString"
       },
       {
           "par_year_built":"findById:property_info/findByString:Year Built/parent/parent/findByClassName:ng-binding/getString"
       },
       {
```

```
            "par_market_value":"findByString:Market Value/parent/parent/findByClas
sName:ng-binding/getString"
        },
        {
            "par_book_record":"findByString:Sales Information/parent/findNextSibli
ng:tr/findNextSibling:tr/findByClassName:text-center/splitString:middle_dash:0"
        },
        {
            "par_page_record":"findByString:Sales Information/parent/findNextSibli
ng:tr/findNextSibling:tr/findByClassName:text-center/splitString:middle_dash:1"
        },
        {
            "str_number_bedrooms":"findById:property_info/findByString: Baths /par
ent/parent/findByClassName:ng-binding/splitString:slash:0"
        },
        {
            "str_number_baths":"findById:property_info/findByString: Baths /parent
/parent/findByClassName:ng-binding/splitString:slash:1"
        },
        {
            "str_effective_area":"findById:property_info/findByString:Actual Area/
parent/parent/findByClassName:ng-binding/getString"
        },
        {
            "str_building_type":"findByStringExact:PA Primary Zone/findNextSibling
:div/getString"
        },
        {
            "str_building_area":"findByStringExact:Adjusted Area/parent/findNextSi
bling:td/getString"
        }
    ],
    "addresses":[
        {
            "unit":"",
            "number":"3031",
            "street":" SW 2 ST",
            "fullAddress":"3031 SW 2 ST"
        }
    ]
}
```

Petición para scraper de impuestos en página de Miami:


```
{
    "city": "Miami Taxes",
    "url": "https://miamidade.county-
taxes.com/public/search/property_tax?action_location=Header",
    "data_access_route": [
        "findById:search_query:clear:putText:propertyId",
        "findByClassName:btn-primary:click",
        "findByClassName:results:findByTagName:a:click",
        "findByStringExact:Account history/findNextSibling:div/findSelector:table
/findAllByClassName:description"
    ],
    "iterate": "findSelector:a/openInNewTab",
    "routes": [
        {
            "tax_year": "findById:bill/getString",
            "tax_paid": "findByClassName:col-12 col-md-4 bill-
details/findChildAtPosition:14/splitString:line_break:1",
            "tax_due": "findByClassName:table table-
hover bills/findByClassName:balance for-cart/getString"
        }
    ],
    "search_parameters": [
        {
            "propertyId": "01-4104-013-0100",
            "owner_name": "MICHAEL J HOLZLI JR",
            "address": "3031 SW 2 ST"
        }
    ]
}
```

Petición para scraper de documentos en página de Fort Lauderdale:


```
{
    "city": "Fort Lauderdale Mortgage",
    "url": "https://officialrecords.broward.org/AcclaimWeb/search/SearchTypeParcel",
    "data_access_route": [
        "findById:btnButton:click",
        "findById:ParcelId:clear:putText:propertyId",
        "findById:btnSearch:click",
        "findById:SearchGridContainer/findById:RsltsGrid/findByClassName:t-grid-content/findAllBySelectorName:tr"
    ],
    "iterate": "click",
    "routes": [
        {
            "doc_date": "",
            "doc_rec_date": "findByClassName:detailsTop2/findByString:Record Date/findNextSibling:div/getString",
            "boo_type": "findByClassName:detailsTop2/findByString:Book Type/findNextSibling:div/getString",
            "doc_page": "findByClassName:detailsTop2/findByString:Page/findNextSibling:div/splitString:slash:1",
            "doc_number": "findByClassName:detailsTop2/findByString:Instrument Number/findNextSibling:div/getString",
            "doc_type": "findByClassName:docBlock/findByString:Doc Type/findNextSibling:div/getString",
            "doc_own_grantor": "findByClassName:docBlock/findByString:Grantor/findNextSibling:div/getString",
            "doc_own_grantee": "findByClassName:docBlock/findByString:Grantee/findNextSibling:div/getString",
            "doc_consideration": "findByClassName:docBlock/findByString:Consideration/findNextSibling:div/getString",
            "doc_assumption": "findByClassName:docBlock/findByString:Mtg Assumption Amt/findNextSibling:div/getString"
        }
    ],
    "search_parameters": [
        {
            "propertyId": "494234014270",
            "owner_name": "MICHAEL J HOLZLI JR",
            "address": "3031 SW 2 ST"
        }
    ]
}
```