**Assignment 6 – Element Collections**

When you were looking at the rdoc for the **Watir::Container** *module*, you may have noticed that for every locator *method* like div there was also a pluralized version: divs. The difference is that while the div *method* returns a single element, the divs *method* returns a *collection* of elements that match your locator.

Continuing with our Travelocity example, let’s look at the Search Results page. If we look at the elements that make up the flights that match our search we will see this HTML:

<ol id="flight-cards">

<li id="result-1\_58" class="module resultCardB" style="">

<li id="result-5\_96-76" class="module resultCardB" style="">

<li id="result-9\_91-84" class="module resultCardB" style="">

<li id="result-14\_89-72" class="module resultCardB" style="">

\*\*\*Additional Code Omitted\*\*\*

</ol>

The HTML <ol> tag stands for Ordered List. It contains a bunch of List Items: <li> tags. Each <li> contains the data of a single search result. If we want to grab all of the search results we can do so:

browser.ol(id: “flight-cards).lis

or

browser.lis(class: “module resultCardB”)

In the first example, I’m using element scope to grab all of the <li> tags that are a child of the Ordered List with id=”flight-cards”.

In the second example, I’m searching for any <li> with a class=” module resultCardB”. The second example is less specific. It will return any <li> with that class, even if they are in separate lists!

The first example is usually the safest way to do it.

Okay, so you know how to get a collection of elements. Now what? We could count how many there are:

browser.ol(id: “flight-cards).lis.size

=> 20

We could get the first or last element:

browser.ol(id: “flight-cards).lis.first

browser.ol(id: “flight-cards).lis.last

We could get an element by its index in the collection:

browser.ol(id: “flight-cards).lis[1]

**Note:** Element Collections, like Arrays in Ruby, use zero-based indexing. That means the first element is at index 0 (zero), so lis[1] would return the second search result.

Usually, when we get a collection of elements it is to do something with each element. We do this by iterating trough the collection using a *each* loop.

browser.ol(id: “flight-cards).lis.each do |li|

puts li.text

end

The code above basically says: For each li element in the lis collection, output it’s text to the console.

Technically, we are calling the each method which iterates over the *collection*. We then assign the element to a *variable* named li, which is then passed to a ‘do’ block. We then execute the code within the ‘do’ block: outputting the text of the <li> element to the console.

Let’s put this into practice:

* Create a new script: assignment\_6.rb
* Go to Travelocity and login
* Do a search with whatever values you chose
* Grab a collection of the search results
* Iterate through the collection and output the fair price to the screen.
  + Hint: use element scope (using the li variable) to search for a div with class=”amt”

**Bonus**

While iterating through the collection also output the # of stops for each result.