**Repeating Web Objects Handle in Automation Anywhere**

The key to handling repeating web objects is to use a combination of object capture, looping, and variables. The bot needs to be able to identify a collection of similar objects and then process each one individually.

**Core Concepts and Actions:**

1. **Object Cloning (Capture Action):**
   * The primary tool is the Capture action from the **Recorder package**. This action allows you to interact with specific web objects.
   * When you capture an object, Automation Anywhere identifies it using its properties (HTML Path, tag, ID, class, etc.). For repeating objects, these properties often follow a predictable pattern.
2. **Loop Action:**
   * This is the engine that drives the repetition. A Loop action is used to iterate through a collection of items.
   * **Looping on a Web Table:** The most common scenario is processing data in a web table. You can use the Capture action with the **"Get table"** action to extract the entire table into a **Table variable**. Then, you use a Loop action with the "For each row in Table variable" iterator to process each row.
   * **Looping on a List of Objects:** If the repeating items aren't in a formal table (e.g., product cards on an e-commerce site, search results), you can use a Loop with the "For each element in list" iterator (if the list can be captured as a whole).
3. **Using Variables for Dynamic Identification:**
   * The key to processing repeating objects is to make the object's identification dynamic. The properties of each repeating item often contain a numerical index (e.g., id="item-1", id="item-2", item\_path="/items[1]/", item\_path="/items[2]/").
   * You can use a variable in place of this index.
   * **Example:**
     + You create a Number variable, let's say $counter$.
     + You start a Loop that increments $counter$.
     + Inside the loop, you use a Capture action.
     + In the Capture action's object properties, you dynamically build the path using your counter variable: HTMLPath = /html/body/div[2]/ul/li[\*\*$counter$\*\*]/a
     + This allows the bot to find the first item, then the second, and so on, for the duration of the loop.

**A Typical Workflow for Repeating Web Objects:**

1. **Identify the repeating pattern:** Manually inspect the HTML of the web page to find the unique properties that identify the repeating objects. Look for an index or a consistent class name.
2. **Create a Loop:** Determine the number of repetitions. You could get the count of items using a capture action or just loop for a fixed number if known.
3. **Define a counter variable:** Initialize a number variable (e.g., $i$) to 1 before the loop starts.
4. **Use Capture inside the loop:**
   * Drag and drop the Capture action inside the loop.
   * Select the first object in the repeating sequence.
   * In the Capture action's object properties, find the property that contains the index.
   * **Replace the hardcoded index with your counter variable.** For example, index='1' becomes index='$i$'.
5. **Perform actions:** Inside the loop, perform the desired actions on the dynamically identified object (e.g., Click, Get text, Set text).
6. **Increment the counter:** Make sure to increment your counter variable at the end of the loop so it moves to the next object in the next iteration.
7. **Handle exit conditions:** Define an exit condition for the loop, such as a check to see if the object exists.

**Interview Questions and Answers**

**1. How do you handle repeating web objects, like rows in a web table, in Automation Anywhere?**

**Answer:** The most robust way to handle repeating web objects is to combine the Capture action with a Loop. I would use the Capture action's "Get table" functionality to extract the entire web table into a **Table variable**. Then, I would use a Loop action with the "For each row in Table variable" iterator to process each row of the extracted data. Inside the loop, I can access individual cell data using the table variable (e.g., $myTableVariable["ColumnName"]$).

**2. What if the repeating web objects are not in a table, but are in a list (e.g., product cards on an e-commerce site)? How would you handle that?**

**Answer:** For repeating objects not in a table, I would use a combination of a Loop and a dynamic Capture action.

1. I would first identify the common property or HTML path pattern of the repeating objects, noting that only a numerical index changes.
2. I would then create a Number variable to act as a counter (e.g., $counter$), initialized to 1.
3. I would start a Loop that runs for the expected number of items or until a condition is met (e.g., the object no longer exists).
4. Inside the loop, I would use a Capture action. In its object properties, I would replace the hardcoded index with my $counter$ variable, making the path dynamic (e.g., id="item\_$counter$" ).
5. I would then perform my actions and, at the end of the loop, increment the $counter$ variable to move to the next item in the next iteration.

**3. Why is using a variable for dynamic object identification more robust than relying on object cloning for each individual object?**

**Answer:** Creating a separate Capture action for each repeating object is not scalable and is very fragile. If the number of objects changes, the bot would break. Using a variable for dynamic identification is robust because:

* **Scalability:** The same logic can process 10 items or 1,000 items without code changes.
* **Maintainability:** You only have one Capture action to manage and debug, not dozens or hundreds.
* **Flexibility:** It adapts to scenarios where the number of items changes dynamically on the web page.

**4. How would you handle a situation where the loop needs to stop when it reaches the last item and a "Next Page" button needs to be clicked to continue?**

**Answer:** This is a classic pagination scenario.

1. I would wrap the entire loop (that processes the items on the current page) in a parent Loop action that runs until the "Next Page" button is no longer visible.
2. Inside the parent loop, after the inner loop finishes processing all items on the current page, I would use an If condition to check if the "Next Page" button exists.
3. If the button exists, I would use a Capture action to click it.
4. If the button does not exist, I would use a Break action to exit the parent loop, signifying the end of all pages.

This approach ensures the bot processes all available data across multiple pages.

**5. What are some key properties you look for when trying to dynamically identify repeating web objects?**

**Answer:** When trying to identify repeating web objects, I look for properties that have a predictable pattern. The most common and useful properties are:

* **HTML Path:** This often contains an index (e.g., div[2]/ul/li[1]/a) that can be replaced with a variable.
* **ID:** The ID might contain a number that increments (e.g., row-1, row-2).
* **Class:** The class name might be consistent across all repeating items (e.g., class="product-card").
* **Attributes:** I would look for other unique attributes or a combination of attributes that uniquely identifies the items.

I would also use the wildcard character (\*) if only a portion of the property is dynamic.