

**Using XPath in Selenium Python to Find List Items with Specific Role and Class Attributes**

Before diving into the solution, I want to explain that combining role attributes with class selectors in XPath allows for precise targeting of elements even when class names alone are insufficient. This approach is particularly useful for working with accessibility-focused web applications.

**Understanding XPath for Role-Based Element Selection**

XPath provides powerful ways to locate elements based on multiple attributes. In your case, you need to:

1. Find parent elements with role='list'
2. Within those elements, find items with both role='listitem' and a specific class

**Basic XPath Strategies**

There are several approaches to achieve this combination:

**Strategy 1: Direct Parent-Child Relationship**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
  
# Initialize driver  
driver = webdriver.Chrome()  
driver.get("your\_url")  
  
# Find all listitems with specific class inside a list  
list\_items = driver.find\_elements(By.XPATH,   
 "//div[@role='list']//div[@role='listitem' and contains(@class, 'your-class-name')]")  
  
# Process the list items  
for item in list\_items:  
 print(item.text)

This XPath expression //div[@role='list']//div[@role='listitem' and contains(@class, 'your-class-name')] targets:

* //div[@role='list'] - finds div elements with role='list'
* // - looks for descendants (at any level)
* div[@role='listitem' and contains(@class, 'your-class-name')] - finds div elements that have both role='listitem' AND a class containing 'your-class-name'

**Strategy 2: Two-Step Approach (More Efficient)**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
  
# Initialize driver  
driver = webdriver.Chrome()  
driver.get("your\_url")  
  
# Step 1: Find the list element  
list\_element = driver.find\_element(By.XPATH, "//div[@role='list']")  
  
# Step 2: Find list items within that list  
list\_items = list\_element.find\_elements(By.XPATH,   
 ".//div[@role='listitem' and contains(@class, 'your-class-name')]")  
  
# Process the list items  
for item in list\_items:  
 print(item.text)

The two-step approach is often more efficient because:

1. It narrows the search scope
2. It can be more performant than a single complex XPath expression

**Best Practices and Optimization**

**1. Use contains() for Class Attributes**

Since HTML elements can have multiple classes, it's better to use contains() rather than exact matching:

# Good - handles elements with multiple classes  
"//div[@role='listitem' and contains(@class, 'your-class-name')]"  
  
# Less reliable - requires exact match of the entire class attribute  
"//div[@role='listitem' and @class='your-class-name']"

The contains() function is especially useful when class names are dynamically generated or when elements have multiple classes[[1]](#fn1).

**2. Handling Multiple Lists**

If you have multiple lists and need to target a specific one, you can add more conditions:

# Target list items in a specific list with an ID  
list\_items = driver.find\_elements(By.XPATH,   
 "//div[@role='list' and @id='specific-list-id']//div[@role='listitem' and contains(@class, 'your-class-name')]")

Or use a two-step approach with more specific first selection:

# Find a specific list first  
specific\_list = driver.find\_element(By.XPATH,   
 "//div[@role='list' and contains(@class, 'specific-list-class')]")  
  
# Then find list items within that list  
list\_items = specific\_list.find\_elements(By.XPATH,   
 ".//div[@role='listitem' and contains(@class, 'your-class-name')]")

**3. Error Handling**

Always include error handling when working with Selenium:

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.common.exceptions import NoSuchElementException  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
  
driver = webdriver.Chrome()  
driver.get("your\_url")  
  
try:  
 # Wait for the list to be present  
 list\_element = WebDriverWait(driver, 10).until(  
 EC.presence\_of\_element\_located((By.XPATH, "//div[@role='list']"))  
 )  
   
 # Find listitems within the list  
 list\_items = list\_element.find\_elements(By.XPATH,   
 ".//div[@role='listitem' and contains(@class, 'your-class-name')]")  
   
 if list\_items:  
 for item in list\_items:  
 print(item.text)  
 else:  
 print("No list items found.")  
   
except NoSuchElementException:  
 print("List element not found.")  
except Exception as e:  
 print(f"An error occurred: {e}")  
finally:  
 driver.quit()

**Advanced XPath Techniques**

**Excluding Elements**

If you need to exclude certain elements, you can use the not() function:

# Find listitems that don't have a specific class  
list\_items = driver.find\_elements(By.XPATH,   
 "//div[@role='list']//div[@role='listitem' and not(contains(@class, 'exclude-this-class'))]")

**Direct Children Only**

If the list items are direct children of the list element (not nested deeper), use a single slash instead of double slash for better performance:

# Only direct children   
list\_items = driver.find\_elements(By.XPATH,   
 "//div[@role='list']/div[@role='listitem' and contains(@class, 'your-class-name')]")

**Conclusion**

To find elements with role='listitem' and a specific class within a parent element with role='list', the most reliable approach is:

# Two-step approach (recommended)  
list\_element = driver.find\_element(By.XPATH, "//div[@role='list']")  
list\_items = list\_element.find\_elements(By.XPATH,   
 ".//div[@role='listitem' and contains(@class, 'your-class-name')]")

This method is robust, performs well, and precisely targets the elements you need. Remember to adapt the tag names (div in this example) to match your actual HTML structure.

**References**

1. XPath strategies for targeting elements with role attributes and classes
2. Combining multiple attributes in XPath expressions[[2]](#fn2)
3. Using contains() and logical operators in XPath[[1]](#fn1)[[3]](#fn3)
4. Advanced XPath techniques for element selection[[4]](#fn4)[[5]](#fn5)

⁂

1. <https://www.h2kinfosys.com/blog/xpath-contains-and-or-parent-start-with-axes-in-selenium-webdriver/>

1. <https://webkul.com/blog/xpath-with-multiple-elements/>

1. <https://www.edureka.co/community/9957/selecting-xpath-multiple-conditions-using-selenium-python>

1. <https://stackoverflow.com/questions/6029232/how-to-select-two-attributes-from-the-same-node-with-one-expression-in-xpath>

1. <https://www.roborabbit.com/blog/how-to-find-elements-by-xpath-in-selenium/>