

Amazon Webscraping using Automation Test Suite

Aaryaman Gupta

Why this testing suite?

- This project is a one stop shop for all automation needs like web testing, web scraping, monitoring and data population using simple commands.
- It enhances flexibility of maintaining the continuity of data, in lengthy automation scenarios through excel workbook.
- The excel tables and predefined columns makes it easy to visualize the dataflow.
- Suite is useful in simplifying the test and flow management in most scenarios, for those having no code knowledge.
- The available source code can be altered using minimum steps.
- Data is generated sequentially in logs and json files in easy to reuse format.
- Pytest and AI is in queue. Possibility of exploring similar features are endless.

Will this work for me?

- This suite can cater from simplest to highly complex programs. Features to add short code snippets in excel itself is inbuilt. Any user with minimal knowledge of Python can use the feature as entire automation is handled by the application.
- New features are being developed to minimise coding requirement for user and maximise flexibility & functionality in excel sheet.

Tech?

- The code engages with browser automation using Playwright. Possibilities are being explored to support Selenium, API testing and mobile testing (Appium) using the same excel format.

Building the project

1) **Clone the project to your desired directory** `git clone git@github.com:BoneyGupta/Amazon-Webscraping.git`

2) **Add the required libraries**

Prerequisites:

- **Python:** Ensure you have Python 3.7 or later installed on your system. You can check your Python version by running `python --version` in your terminal or command prompt.
- **pip:** Make sure pip, the Python package installer, is also installed. If not, you can install it by following the instructions for your operating system:
 - > Windows: Download and run the get-pip.py script from <https://bootstrap.pypa.io/get-pip.py>.
 - > macOS/Linux: Open your terminal and run `python3 -m pip install --upgrade pip`.

Installation:

- Open your terminal or command prompt.
- Install Playwright using pip: `pip install playwright`
- Install the Playwright browser binaries: `playwright install`
- Install openpyxl: `pip install openpyxl`

5) **Add Chrome Application folder to your directory (If cdp is required)**

- Install Chrome Browser on your device.
- Copy the folder `C:\Program Files\Google\Chrome\Application` and add it to your directory.
- Open command line runner and run the command `chrome.exe --remote-debugging-port=9988 --user-data-dir=..\chromedata`
- Wait for the browser to open and in the running browser login to the desired website.

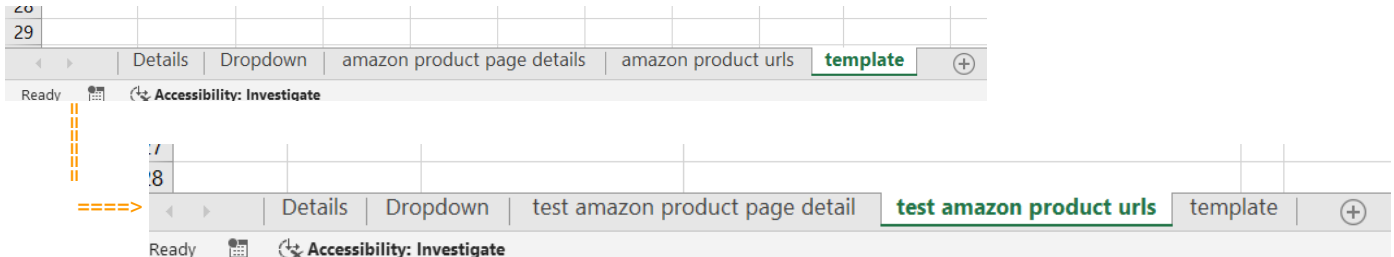
6) **Run the program** `python main.py`

7) **Enter the query when prompted**

8) **Get the reports from Reports folder.**

Step 1: First steps to start the process

- 1) Go to the automation testing suite directory >> open excel directory
- 2) Open the Test.xlsx for your first program(you can refer to 'Template.xlsx' and 'Automation Test Suite Commands.xlsx')
Presently, only the excel file named as 'Test.xlsx' will be run for automation.
- 3) A pre-filled sheets will be present for your aide.
- 4) There are three required sheets(Details, Sheet having 'test' in its name and Dropdown) for the execution of test cases.
Update the name of all the sheets which are up for testing with 'test' as prefix.



All the sheets and fields are described with the help of test case: amazon product page details

Step 2: Details Entry

Open the 'Details' sheet in 'Test.xlsx'

Test Name	Amazon Test
Browser	Chrome
Website	https://www.amazon.in
Headless	TRUE
cdp	TRUE

Field entry options:

Any alphanumeric value
Chrome, Firefox, Edge
Complete website url
TRUE, FALSE
TRUE, FALSE

Description:

Declares the name of the Test
Select a browser
Website names should be complete with https:// or similar prefix
Headless testing is a testing which allows the browser to run in the background without GUI.
cdp is required when we want to work with already logged on browser with websites which requires login or captcha. Do not support Headless testing.

Step 3: Create a test Sheet

Create a new sheet with the below columns and add 'test' prefix if you want to run it as a test.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	1	2	3	4	5	6	7	8	13	9	10	11	12	15	16	17
2	Element ID	Description	Execute	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Wait(ms)	URL	File Path	Conditional Key	Conditional Statement dictionary: dict={}
3																
4																
5																

*Next steps might be optional depending on the test.

- 1) Fill in the Element ID(User can add any alphanumeric value depending on their organization)
- 2) Performing a basic action
 - >> Add the locator of the element
 - >> Add nth (in case of multiple elements with same locator)
 - >> Add the action you want to perform (Actions can be viewed from Dropdown sheet)
 - >> Add the value (Search Item name) you wish to pass through the action
 - >> add a wait timer (default is 3000 millisecond) which waits for the element to be interactable.
 Now we can open [amazon.in](https://www.amazon.in) go to the search box, fill up 'smartphone' and click search.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	1	2	3	4	5	6	7	8	13	9	10	11	12	15	
2	Element ID	Description	Execute	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Wait(ms)	URL	File Path	Conditional Key
3	2	Fill the Search Box in amazon.in		//input[@id="twotabse		fill	Smartphone					30000			
4	3	Click Search		//input[@id='nav-sear		click						15000			
5															

- 3) Add an assertion
 - >> hard assertion means the test will terminate on error
 - >> soft means the test will continue with logging the case as failed
 - >> if assertion is left blank, in case of any error, the test will terminate.
 - >> fill the assert type, the assert condition and the assert value if required.
 In element id 5 and 6, we are checking for the elements to be attached to the DOM and storing the text content of product title and rating in a dictionary.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	1	2	3	4	5	6	7	8	13	9	10	11	12	15	
2	Element ID	Description	Execute	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Wait(ms)	URL	File Path	Conditional Key
3	2	Fill the Search Box in amazon.in		//input[@id="twotabse		fill	Smartphone					30000			
4	3	Click Search		//input[@id='nav-sear		click						15000			
5															

- 4) Add a loop

There are 3 types of looping sequence we can use

Loop Type 1 : Simple Loop									Description
Description	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	This is a simple loop. The loop will iterate through all the elements in the element list created through the given locator. The key is given to store loop data in the reference loop dictionary and data dictionary.
start loop	locator							key	
...test rows goes here...									
end loop									

Loop Type 2 : end case									Description
Description	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Same as the above but the loop may end prematurely if the end loop assertion fails
start loop	locator							key	
...test rows goes here...									
end loop	Locator				soft/hard	condition	assert value		

Loop Type 3 : end case continuous									Description	
Description	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	The loop can continue endlessly until the end loop assertion fails. The length of element list does not matter in this loop type	
start loop	locator							key		
...test rows goes here...										
end loop										
continuous	locator				soft/hard	condition	assert value			

Step 4: Get the results

There are 5 documents of results that are extracted from the test.

- **Code Progression (code_prog.log)**

Details of the code that is run sequentially due to our test.
- **Logs (logs.log)**

Logs of the test run.
- **Sequential Data (data.json)**

Data in the form of sequence. We can get the flow of the code using this json along with the retrieved data
- **Reference Data (reference_data.json)**

We can obtain all the data retrieved in the form of json
- **Reference Loop Data (ref_loop_data.json)**

We can get the data retrived in each loop
- Tasks which require python knowlege are given in this pdf in the Dev sheet
- There are other functionalities and data entry method which will enable user to perform other tasks. The methods of which are given in details in the next sheets.

Details Data Entry	
--------------------	--

Functionality Not Developed Yet

- Declares the name of the Test
- Select a browser
- Website names should be complete with https:// or similar prefix
- Headless testing is a testing which allows the browser to run in the background without GUI.
- cdp is required when we want to work with already logged on browser with websites which requires login or captcha. Do not support Headless testing

Action Data Entry and Code

Description	Locator	nth	Action	Value	Stored Value Key	URL	File Path	Code
Clicks on an element.	locator	nth	click					page.locator(locator).nth(nth).click()
Fills an input element with text.	locator	nth	fill	value				page.locator(locator).nth(nth).fill(value)
Presses a specific key or combination of keys.	locator	nth	press	value				page.locator(locator).nth(nth).press(value)
Double-clicks on an element.	locator	nth	double click					page.locator(locator).nth(nth).double_click()
Sets the focus on an element.	locator	nth	focus					page.locator(locator).nth(nth).focus()
Hovers the mouse over an element.	locator	nth	hover					page.locator(locator).nth(nth).hover()
Selects a specific option from a select element.	locator	nth	select option	value				page.locator(locator).nth(nth).select_option(value)
Selects multiple options from a select element.	locator	nth	select options	"value1" , "value2"				page.locator(locator).nth(nth).select_options(value)
Gets the text content of an element.	locator	nth	text content		key			data = page.locator(tr.locator).first.text_content()
Gets the inner text content of an element.	locator	nth	inner text		key			data = page.locator(locator).nth(nth).inner_text()
Gets the value of a specified attribute from an element.	locator	nth	get attribute		key			data = page.locator(locator).nth(nth).get_attribute(f'"{tr.value}"')
Checks if a checkbox or radio button is checked.	locator	nth	is checked		key			data = page.locator(locator).nth(nth).is_checked()
Checks if an element is disabled.	locator	nth	is disabled		key			data = page.locator(locator).nth(nth).is_disabled()
Checks if an element is visible.	locator	nth	is visible		key			data = page.locator(locator).nth(nth).is_visible()
Checks if an element is hidden.	locator	nth	is hidden		key			data = page.locator(locator).nth(nth).is_hidden()
Checks if an element is enabled.	locator	nth	is enabled		key			data = page.locator(locator).nth(nth).is_enabled()
Gets the number of matching elements.	locator	nth	count	value	key			data = page.locator(locator).nth(nth).count()
Gets the inner text of all matching elements.	locator	nth	all inner texts		key			data = page.locator(locator).nth(nth).all_inner_texts()
Gets the text content of all matching elements.	locator	nth	all text contents		key			data = page.locator(locator).nth(nth).all_text_contents()
Opens a link in a new tab.			open in new tab			url		new_window = page.context.new_page() new_window.goto(url) new_window.bring_to_front()
Gets the parent tab of a window.			parent tab					total_pages = page.context.pages total_pages[0].bring_to_front()
Closes the current tab.			close tab					page.close()
In a loop, it navigates to the newly opened tab.			loop new tab					
Opens a link.			open link			url		page.goto(url)
Takes a screenshot of the entire page.			page screenshot				screenshot.png	page.wait_for_load_state("load") page.screenshot(path=f"{logs.directory_path}\\pgscreenshot{time.strftime("%H%M%S")}{tr.filepath}", timeout=100000)
Takes a screenshot of a specific element.	locator	nth	element screenshot				screenshot.png	page.wait_for_load_state("load") page.locator(tr.locator).screenshot(path=f"{logs.directory_path}\\pgscreenshot{time.strftime("%H%M%S")}{tr.filepath}")
Gets the current URL of the page.	locator	nth	get page url		key			data = page.url
Stores the url an row to restart the program incase of failure			master url					
Launches a new browser instance an open master url of current page			fresh browser					browser = pw.chromium.launch(headless=headless) context = browser.new_context() page = context.new_page() page.goto(master_url)
Create html file which has the links of all the urls from reference dictionary.			create html	key(which has the urls from reference dictionary)			url.html	
Selects all matching elements.			all					
Stores the value of an element in a variable.			store					
Sets the files for a file input element.			set Input Files					
Gets the inner HTML content of an element.			inner HTML					
Waits for a specific condition to be met.			wait for					
Waits for an element to reach a specific state (e.g., visible, hidden, stable).			wait for element state					
Takes a screenshot of the page or an element.			screenshot					
Gets the bounding box of an element.			bounding box					
Combines locators using logical AND.			and					
Filters a list of elements based on a condition			filter					

Selects the first matching element.	first		
Selects the last matching element.	last		
Selects the nth matching element.	nth		
Selects elements based on their text content.	with text		
Checks a checkbox or radio button.	check		

Wait Data Entry, Code and Exception Handling
--

Description	Locator	nth	Assert	Wait(ms)	Code	try/catch
Wait	locator	nth	soft	timeout	page.locator(locator).nth(nth).wait_for(timeout=timeout)	Yes
	locator		soft	timeout	page.wait_for_selector(locator, timeout=timeout)	Yes
	locator	nth		timeout	page.locator(locator).nth(nth).wait_for(timeout=timeout)	No
	locator			timeout	page.wait_for_selector(locator, timeout=timeout)	No
	locator		hard	timeout	print("No explicit waiting can be performed")	No

- The wait function can be merged with Action, Assert, Execute and Loop functions. Further details available in their respective sheets.
- This is not meant to be a standalone function but can be used as such.

Assertion Data Entry and Code

Description	Locator	nth	Assert	Condition	Assert Value	Code
The element is attached to the DOM.	locator	nth	soft/hard	to be attached		expect(page.locator(locator).nth(nth),element not present in page/dynamic element).to_be_attached()
The element is checked (for checkboxes or radio buttons).	locator	nth	soft/hard	to be checked		expect(page.locator(locator).nth(nth),element not present in page/dynamic element).to_be_checked()
The element is disabled.	locator	nth	soft/hard	to be disabled		expect(page.locator(locator).nth(nth),element is not disabled).to_be_disabled()
The element is editable (e.g., input fields).	locator	nth	soft/hard	to be editable		expect(page.locator(locator).nth(nth),element is not editable).to_be_editable()
The element's value is empty.	locator	nth	soft/hard	to be empty		expect(page.locator(locator).nth(nth),element is not empty).to_be_empty()
The element is enabled (not disabled).	locator	nth	soft/hard	to be enabled		expect(page.locator(locator).nth(nth),element is not enabled).to_be_enabled()
The element has focus.	locator	nth	soft/hard	to be focused		expect(page.locator(locator).nth(nth),element is not focused).to_be_focused()
The element is hidden (not visible).	locator	nth	soft/hard	to be hidden		expect(page.locator(locator).nth(nth),element is not hidden).to_be_hidden()
The element is partially or fully visible within the viewport.	locator	nth	soft/hard	to be in viewport		expect(page.locator(locator).nth(nth),element is not in viewport).to_be_in_viewport()
The element is visible.	locator	nth	soft/hard	to be visible		expect(page.locator(locator).nth(nth),element is not visible).to_be_visible()
The element contains the specified text.	locator	nth	soft/hard	to contain text	partial_text	expect(page.locator(locator).nth(nth),element does not contain text).to_contain_text(assert_value)
The element has an accessible description.	locator	nth	soft/hard	to have accessible description	value	expect(page.locator(locator).nth(nth),element does not have accessible description).to_have_accessible_description(assert_value)
The element has an accessible name.	locator	nth	soft/hard	to have accessible name	value	expect(page.locator(locator).nth(nth),element does not have accessible name).to_have_accessible_name(assert_value)
The element has the specified attribute.	locator	nth	soft/hard	to have attribute	"attribute","expected_value"	expect(page.locator(locator).nth(nth),element is not empty).to_have_attribute(assert_value)
The element has the specified class.	locator	nth	soft/hard	to have class	class	expect(page.locator(locator).nth(nth),element does not have class).to_have_class(assert_value)
The locator matches the specified number of elements.	locator	nth	soft/hard	to have count	count	expect(page.locator(locator).nth(nth),element does not have count).to_have_count(assert_value)
The element has the specified CSS property.	locator	nth	soft/hard	to have css	css	expect(page.locator(locator).nth(nth),element does not have css).to_have_css(assert_value)
The element has the specified ID.	locator	nth	soft/hard	to have id	id	expect(page.locator(locator).nth(nth),element does not have id).to_have_id(assert_value)
The element has the specified JavaScript property.	locator	nth	soft/hard	to have js property	"property","expected_value"	expect(page.locator(locator).nth(nth),element does not have the js properties).to_have_js_property(assert_value)
The element has the specified ARIA role.	locator	nth	soft/hard	to have role	role	expect(page.locator(locator).nth(nth),element does not have the role).to_have_role(assert_value)
The element contains text.	locator	nth	soft/hard	to have text	text	expect(page.locator(locator).nth(nth),element does not have the text).to_have_text(assert_value)
The element has the specified value.	locator	nth	soft/hard	to have value	value	expect(page.locator(locator).nth(nth),element does not have the value).to_have_value(assert_value)
The element has multiple values (e.g., for select elements).	locator	nth	soft/hard	to have values	"value1","value2"	expect(page.locator(locator).nth(nth),element do not have the values).to_have_values([assert_value])
The element has the specified title attribute.			soft/hard	to have title	title	expect(page,page does not have title).to_have_title(assert_value)
The element has the specified URL (e.g., for links).			soft/hard	to have url	url	expect(page,page does not have url).to_have_url(assert_value)
The page load was successful and there are no errors.	locator	nth	soft/hard	to be ok		expect(page.locator(locator).nth(nth),element is not ok).to_be_ok()
The element is not attached to the DOM.	locator	nth	soft/hard	not to be attached		expect(page.locator(locator).nth(nth),element is present in page/dynamic element).not_to_be_attached()
The element is not checked (for checkboxes or radio buttons).	locator	nth	soft/hard	not to be checked		expect(page.locator(locator).nth(nth),element is checked).not_to_be_checked()
The element is not disabled.	locator	nth	soft/hard	not to be disabled		expect(page.locator(locator).nth(nth),element is disabled).not_to_be_disabled()
The element is not editable (e.g., input fields).	locator	nth	soft/hard	not to be editable		expect(page.locator(locator).nth(nth),element is editable).not_to_be_editable()
The element's value is not empty.	locator	nth	soft/hard	not to be empty		expect(page.locator(locator).nth(nth),element is empty).not_to_be_empty()
The element is not enabled (disabled).	locator	nth	soft/hard	not to be enabled		expect(page.locator(locator).nth(nth),element is enabled).not_to_be_enabled()
The element does not have focus.	locator	nth	soft/hard	not to be focused		expect(page.locator(locator).nth(nth),element is focused).not_to_be_focused()
The element is not hidden (visible).	locator	nth	soft/hard	not to be hidden		expect(page.locator(locator).nth(nth),element is hidden).not_to_be_hidden()
The element is not partially or fully visible within the viewport.	locator	nth	soft/hard	not to be in viewport		expect(page.locator(locator).nth(nth),element is in viewport).not_to_be_in_viewport()
The element is not visible.	locator	nth	soft/hard	not to be visible		expect(page.locator(locator).nth(nth),element is visible).not_to_be_visible()
The element does not contain the specified text.	locator	nth	soft/hard	not to contain text	partial_text	expect(page.locator(locator).nth(nth),element contains text).not_to_contain_text(assert_value)
The element does not have an accessible description.	locator	nth	soft/hard	not to have accessible description	value	expect(page.locator(locator).nth(nth),element has accessible description).not_to_have_accessible_description(assert_value)
The element does not have an accessible name.	locator	nth	soft/hard	not to have accessible name	value	expect(page.locator(locator).nth(nth),element has accessible name).not_to_have_accessible_name(assert_value)
The element does not have the specified attribute.	locator	nth	soft/hard	not to have attribute	"attribute","expected_value"	expect(page.locator(locator).nth(nth),element has attribute).not_to_have_attribute(assert_value)
The element does not have the specified class.	locator	nth	soft/hard	not to have class	class	expect(page.locator(locator).nth(nth),element has class).not_to_have_class(assert_value)
The locator does not match the specified number of elements.	locator	nth	soft/hard	not to have count	count	expect(page.locator(locator).nth(nth),element has count).not_to_have_count(assert_value)

The element does not have the specified CSS property.	locator	nth	soft/hard	not to have css	css	expect(page.locator(locator).nth(nth),element has css).not_to_have_css(assert_value)
The element does not have the specified ID.	locator	nth	soft/hard	not to have id	id	expect(page.locator(locator).nth(nth),element has id).not_to_have_id(assert_value)
The element does not have the specified JavaScript property.	locator	nth	soft/hard	not to have js property	"property","expected_value"	expect(page.locator(locator).nth(nth),element has the js properties).not_to_have_js_property(assert_value)
The element does not have the specified ARIA role.	locator	nth	soft/hard	not to have role	role	expect(page.locator(locator).nth(nth),element has the role).not_to_have_role(assert_value)
The element does not contain text.	locator	nth	soft/hard	not to have text	text	expect(page.locator(locator).nth(nth),element has the text).not_to_have_text(assert_value)
The element does not have the specified value.	locator	nth	soft/hard	not to have value	value	expect(page.locator(locator).nth(nth),element has value).not_to_have_value(assert_value)
The element does not have multiple values (e.g., for select elements).	locator	nth	soft/hard	not to have values	"value1","value2"	expect(page.locator(locator).nth(nth),element has values).not_to_have_values([assert_value])
The element does not have the specified title attribute.			soft/hard	not to have title	title	expect(page,page has title).not_to_have_title(assert_value)
The element does not have the specified URL (e.g., for links).			soft/hard	not to have url	url	expect(page,page has url).not_to_have_url(assert_value)
The page load was not successful or there are errors.	locator	nth	soft/hard	not to be ok		expect(page.locator(locator).nth(nth),element is ok).not_to_be_ok()

Loop Structure and Data Entry

Loop Type 1 : Simple Loop										Description
Description	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Wait(ms)	
start loop	locator							key		
end loop										
...test rows goes here...										This is a simple loop. The loop will iterate through all the elements in the element list created through the given locator. The key is given to store loop data in the reference loop dictionary and data dictionary.

Loop Type 2 : end case										Description
Description	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Wait(ms)	
start loop	locator							key		
end loop	Locator				soft/hard	condition	assert value			
...test rows goes here...										Same as the above but the loop may end prematurely if the end loop assertion fails

Loop Type 3 : end case continuous										Description
Description	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Wait(ms)	
start loop	locator							key		
end loop continu.	Locator				soft/hard	condition	assert value			
...test rows goes here...										The loop can continue endlessly until the end loop assertion fails. The length of element list does not matter in this loop type

13

Description	Locator	nth	Action	Value	Assert	Condition	Assert Value	Stored Value Key	Wait(ms)	URL	File Path
start loop			click		Hard	to be attached					
end loop			fill	value	Soft	to be checked					
end loop continuous			press	value	hard	to be disabled					
			double click		soft	to be editable					
			focus			to be empty					
			hover			to be enabled					
			select option	value		to be focused					
			select options	"value1", "value2"		to be hidden					
			text content			to be in viewport		key			
			inner text			to be visible		key			
			get attribute			to contain text	partial_text	key			
			is checked			to have accessible description	value	key			
			is disabled			to have accessible name	value	key			
			is visible			to have attribute	"attribute", "expected_value"	key			
			is hidden			to have class	class	key			
			is enabled			to have count	count	key			
			count	value		to have css	css	key			
			all inner texts			to have id	id	key			
			all text contents			to have js property	"property", "expected_value"	key			
			open in new tab			to have role	role			url	
			parent tab			to have text	text				
			close tab			to have value	value				
			loop new tab			to have values	"value1", "value2"				
			open link			to have title	title			url	
			page screenshot			to have url	url				screenshot.png
			element screenshot			to be ok					screenshot.png
			get page url			not to be attached		key			
			master url			not to be checked					
			fresh browser			not to be disabled					
			create html	key(which has the urls fi		not to be editable					url.html
			all			not to be empty					
			store			not to be enabled					
			set Input Files			not to be focused					
			inner HTML			not to be hidden					
			wait for			not to be in viewport					
			wait for element state			not to be visible					
			screenshot			not to contain text	partial_text				
			bounding box			not to have accessible descripti	value				
			and			not to have accessible name	value				
			filter			not to have attribute	"attribute", "expected_value"				
			first			not to have class	class				
			last			not to have count	count				
			nth			not to have css	css				
			with text			not to have id	id				
			check			not to have js property	"property", "expected_value"				
						not to have role	role				
						not to have text	text				
						not to have value	value				
						not to have values	"value1", "value2"				
						not to have title	title				
						not to have url	url				
						not to be ok					