ACCELQ

Mastering the art of Object Recognition

Industry Best Practices for a sustainable Automation

Element Identification consists of selecting the attributes or properties of a control that uniquely identify it in the page. The higher the quality of the element identification, the more resilient and maintainable your automated tests become. A few moments spent optimizing your element identification today can eliminate hours of test maintenance tomorrow. Learn and apply best practices from this guide for a **Sustainable Automation**.

Maintainability is the key!

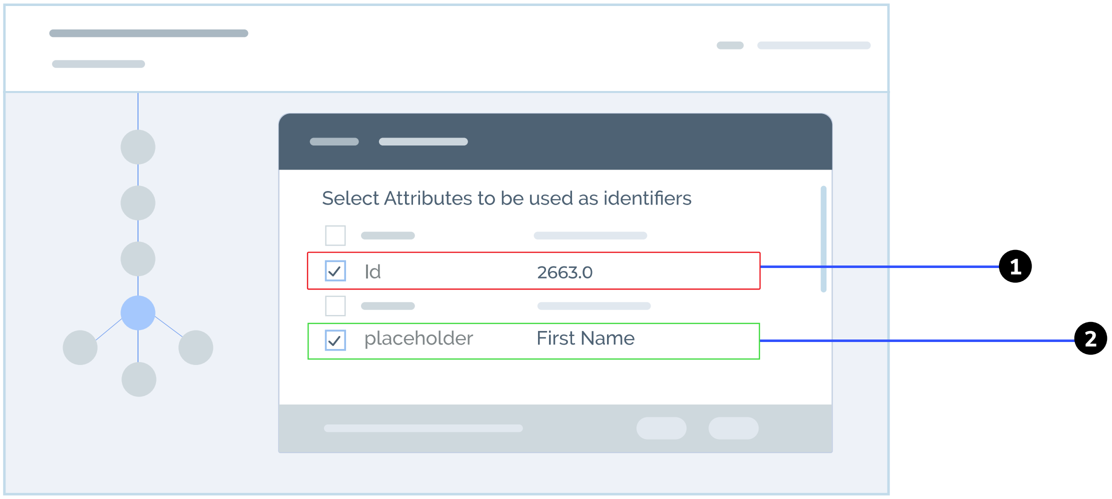
- Multiple attribute combinations may yield uniqueness, but the key is to choose right set of attributes
- When it comes to Element attributes, *change* is a *constant*. Be prepared to adapt to changes from the get-go.
- Element Explorer in ACCELQ provides powerful capabilities to construct stable Element ID.
- Poor element design has significant impact on long term reliability and change management.
- Several techniques discussed in this guide along with applied examples.

Select right attributes

Random or numeric values

Avoid attributes with random numeric or non-functional text/value

1. `ID` in the example, may change any time or may be different for various use cases on the application.
2. `Placeholder` may be a better alternative.


Select Attributes to be used as identifiers

✓	Id	2663.0
✓	placeholder	First Name

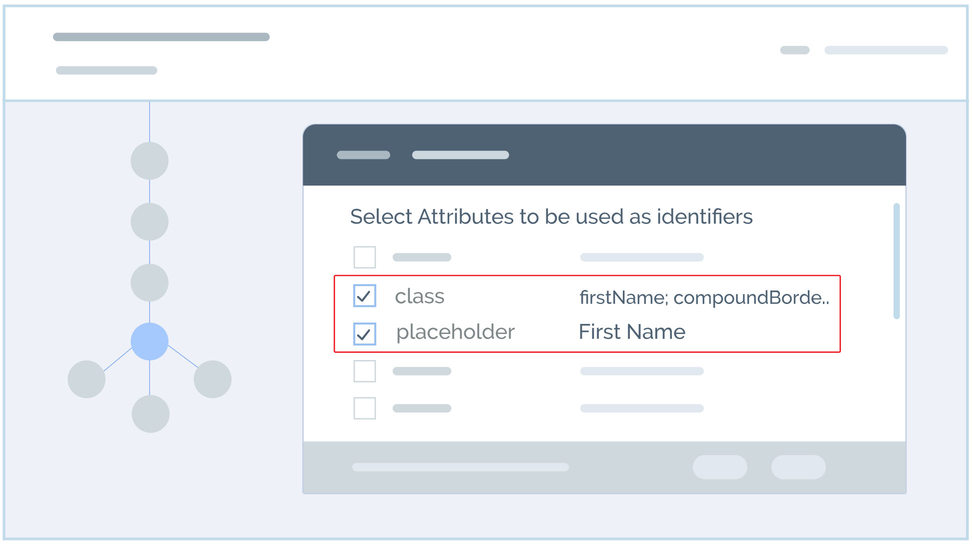
Select right attributes

Use Minimal set of attributes

Judicious choice is important when you have multiple alternatives

Both `placeholder` and `class` provide uniqueness – one of them is enough.

Increased risk of future change when relying on more attributes than necessary.


Select Attributes to be used as identifiers

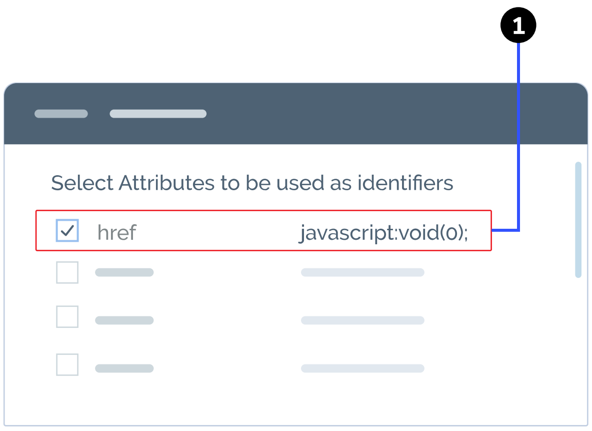
✓	class	firstName; compoundBorde..
✓	placeholder	First Name

Select right set of attributes

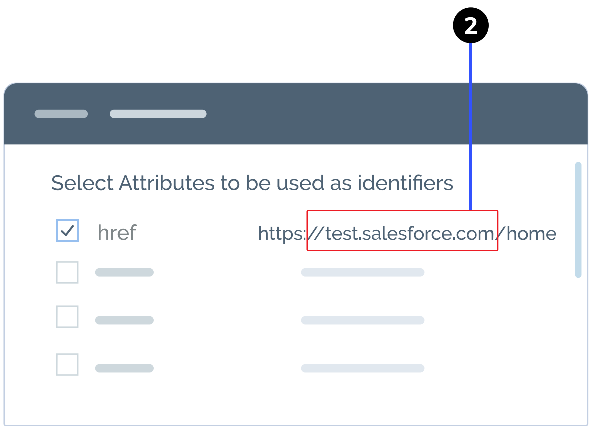
Pay attention to `href` attribute

Look out for gotchas when using `href` attribute

1. `href` attribute pointing to a JavaScript function is not a good candidate - uniqueness is usually short-lived.
2. When pointing to an absolute path of a resource, make sure to remove environment dependency. Learn more in regex section.


Select Attributes to be used as identifiers

✓	href	javascript:void(0);


Select Attributes to be used as identifiers

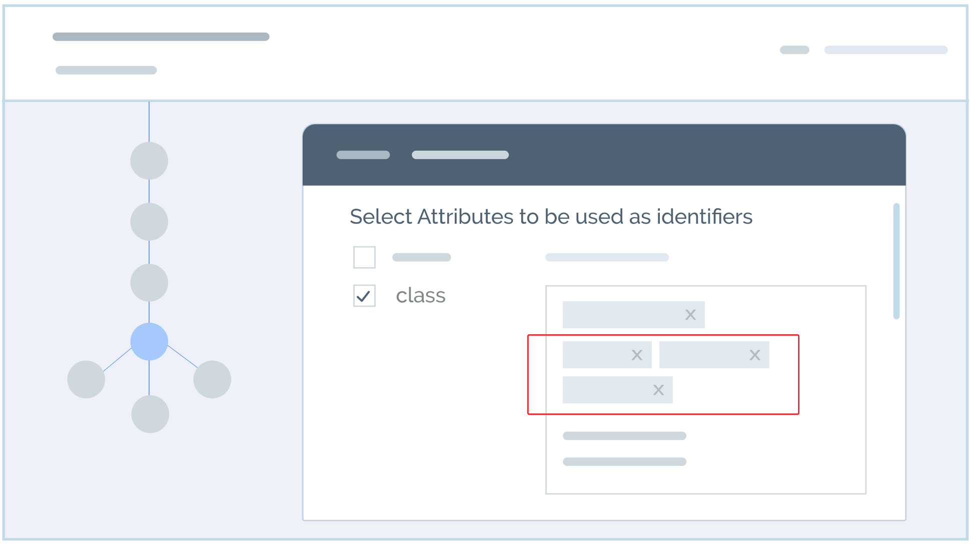
✓	href	https://test.salesforce.com/home

Select right set of attributes

It may not be *All or None* when using `Class` attribute

`Class` attribute refers to CSS classes the element is rendered with. You may have multiple classes for a given element.

- When using this attribute, be sure to remove unnecessary classes.
- Make sure the selected classes have a functional connotation.
- Particularly, remove framework specific classes such as bootstrap col division (col-md-*) or angular (ng-*) classes.



Select right set of attributes

Context is important with `text` attribute

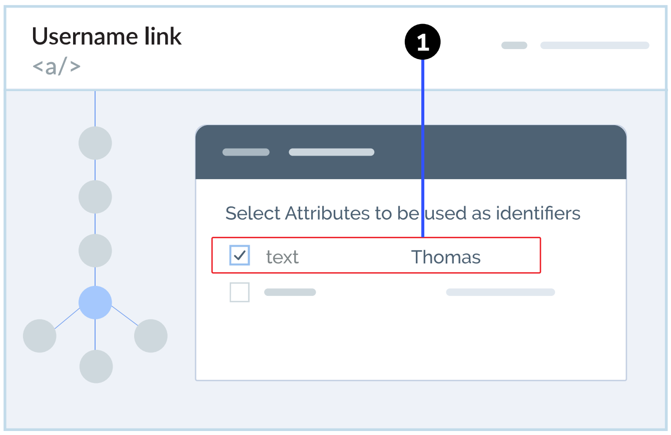
`text` attribute is extremely useful, but take a note of the value of this attribute

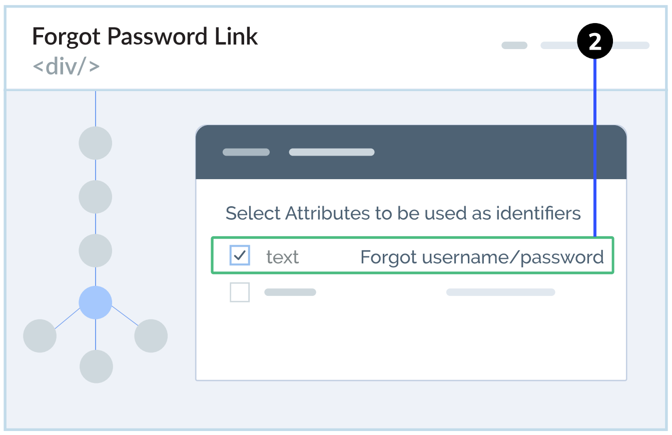
1. First example (Username Link), you will notice that the text value will change if someone else logs in.

2. In the second example (forgot password link), text value is pretty functional and safe to use!

Watch out for extraneous blank space characters, leading or trailing spaces, multiple consecutive spaces etc.

Make sure to utilize Regex so you are not surprised at runtime!



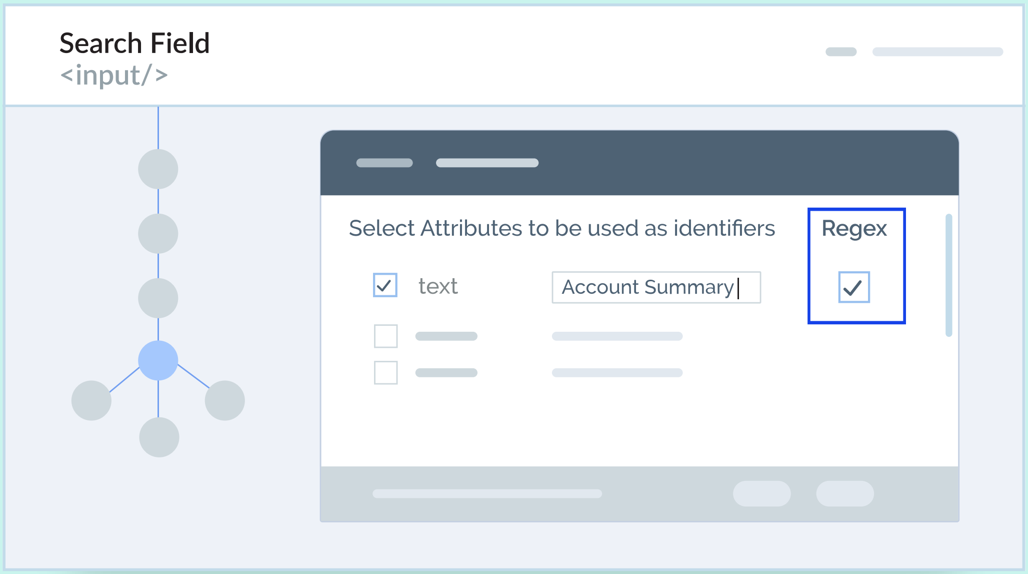


Dynamic Content?

Regular Expressions to rescue!

- Powerful tool for dealing with dynamic element attribute values.
- Allows using attributes which are not stable as a whole, but may present portions of functional and stable text.
- ACCELQ follows standard JavaScript patterns for regular expressions.

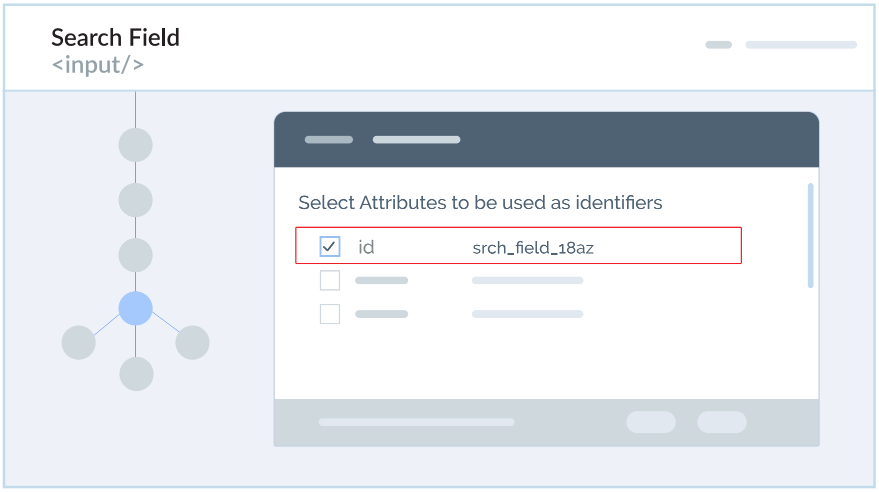
Checkbox appears against an attribute when you are editing its value in ACCELQ



Regular Expression by Example

Using Regular Expression for ID attribute

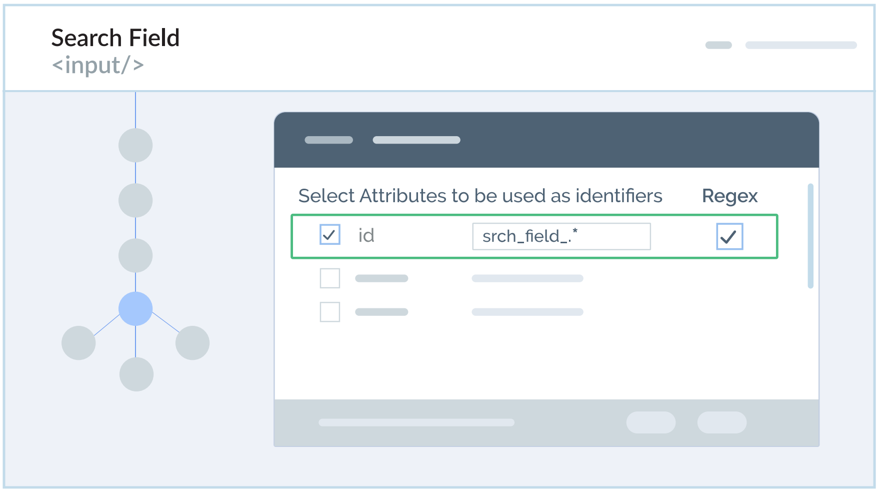
- Let's remove dynamic/random portion from the id attribute below.
- `srch_field` portion of text in the ID seems perfectly appropriate for the element.



Search Field
<input/>

Select Attributes to be used as identifiers

✓	id	srch_field_18az



Search Field
<input/>

Select Attributes to be used as identifiers			Regex
✓	id	srch_field_.*	✓

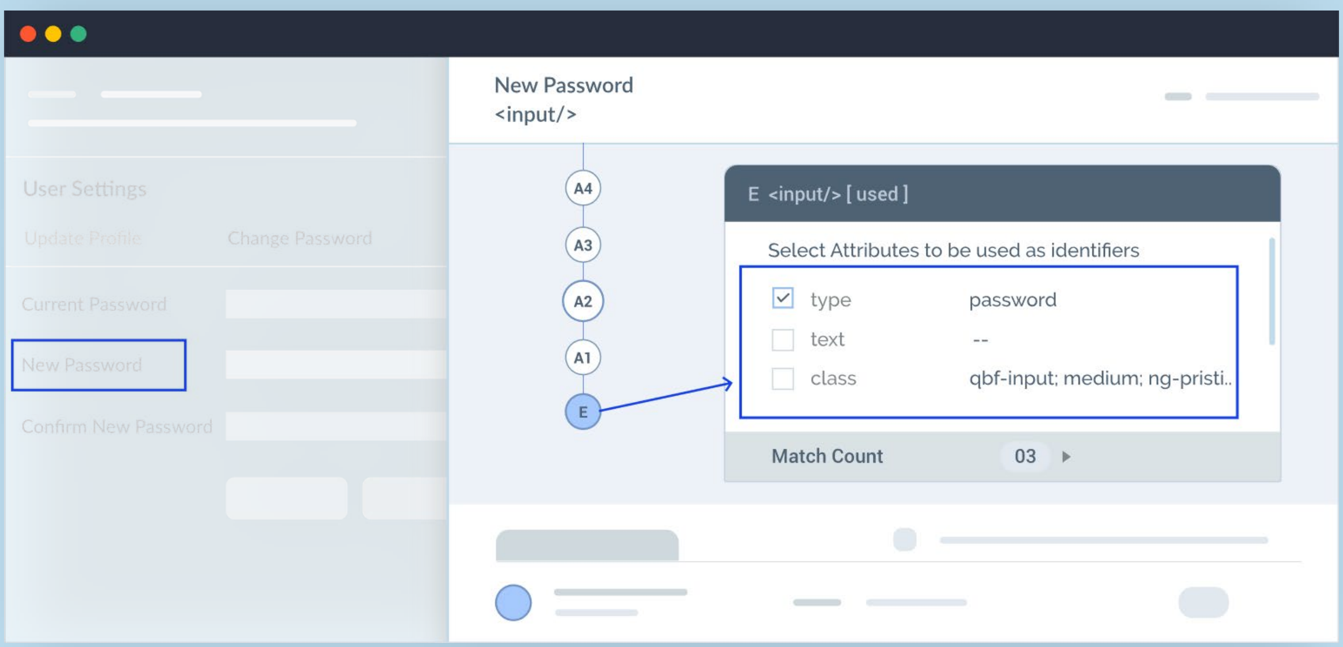
Regular Expression Examples

EXAMPLE		REGEX PATTERN
text with leading and trailing spaces	" Select Hotel "	"Select Hotel" (remove spaces and mark as "regex")
Multiple consecutive spaces	" Select Hotel "	Select[ ]+Hotel"
Text with unwanted/random numerics	" Select Hotel 1873 "	"Select Hotel [0-9]+"
Text with unwanted alphanumeric text	" Select Hotel a3X0kj8U "	"Select Hotel.*"
Text with multiple variable components	" 18 of 32 flights from $320 "	"[0-9]+ of [0-9]+ flights from \$[0-9]+"
Generic Phone number	"972-333-1322"	"[0-9]{3}\-[0-9]{3}\-[0-9]{4}"
Generic email address	"dilbert@accelq.com"	"[a-z]+@[a-z]+\.[a-z]+"

Reference Link https://www.rexegg.com/regex-quickstart.html

When self-attributes are not unique, **you may count on neighborhood!**

- At times, an element itself may not have any unique attributes of its own.
- Notice the "New Password" field has no unique attributes (3 matches with its attributes)



E <input/> [used]

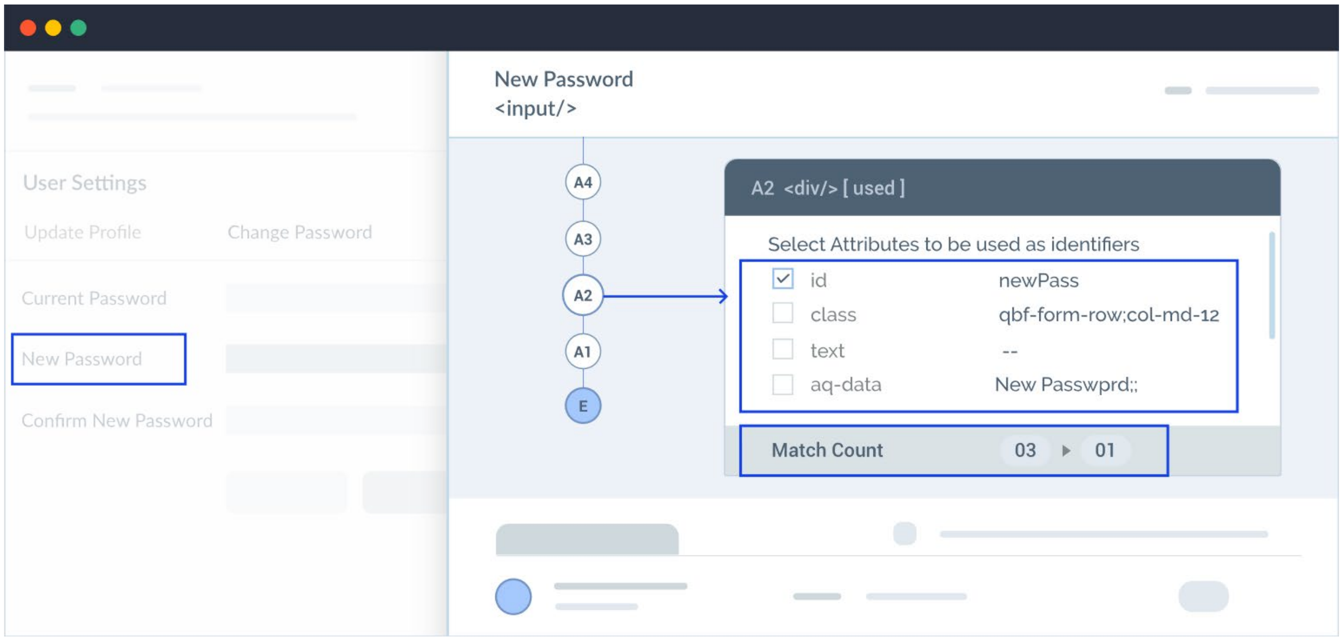
Select Attributes to be used as identifiers

✓	type	password
	text	--
	class	qbf-input; medium; ng-pristi..

Match Count 03

Utilize neighborhood nodes for uniqueness

- Utilize one or more Ancestor or Descendant nodes attributes for additional options.
- You “add” nodes to Selector, which typically only contains the Element itself.
- In this example, we added “A2” node to selector with “id” attribute.



Select Attributes to be used as identifiers

✓	id	newPass
	class	qbf-form-row;col-md-12
	text	--
	aq-data	New Passwprd;;

Match Count 03 ▸ 01

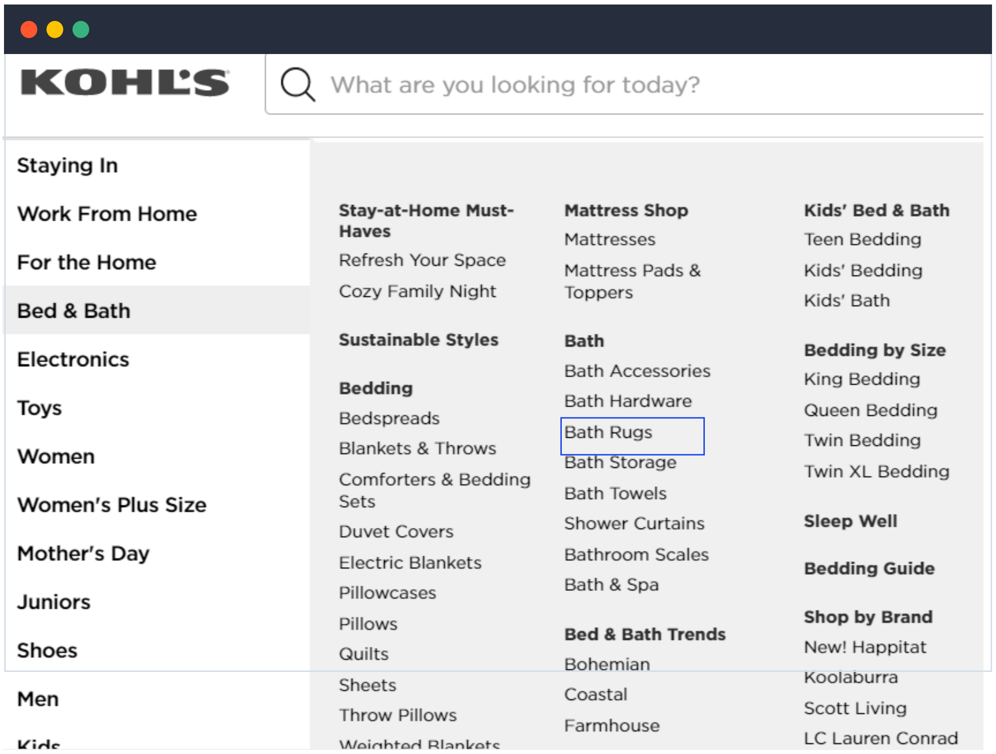
Similar Element Pattern

- Group of elements that are functionally similar in the context of the application.
- Differ among each other by one or two attribute values.
- Typically list type of elements in one functional context.

Example with large set of menu links

Similar elements are all-pervasive!

- There is a total of 906 category links across all Departments on Kohl's home page!
- It is obvious we do not want to save 906 elements in our Repository. How do we optimize this?



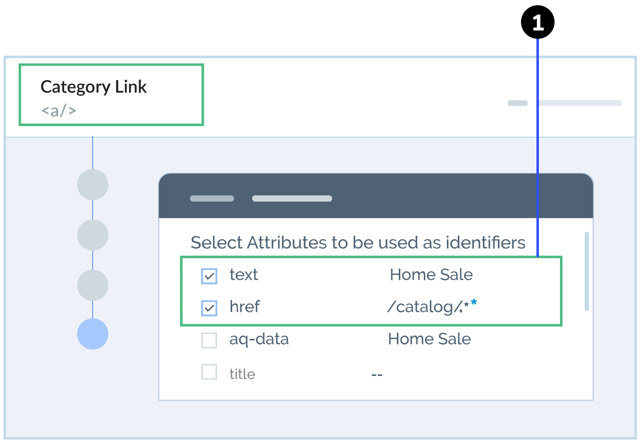
Notice the Category Links are Similar Elements

- All these links are indicating a shopping category.
- 'text' value is the only difference among all the Category links in the example.

Bringing Element Repo & Logic Editor together

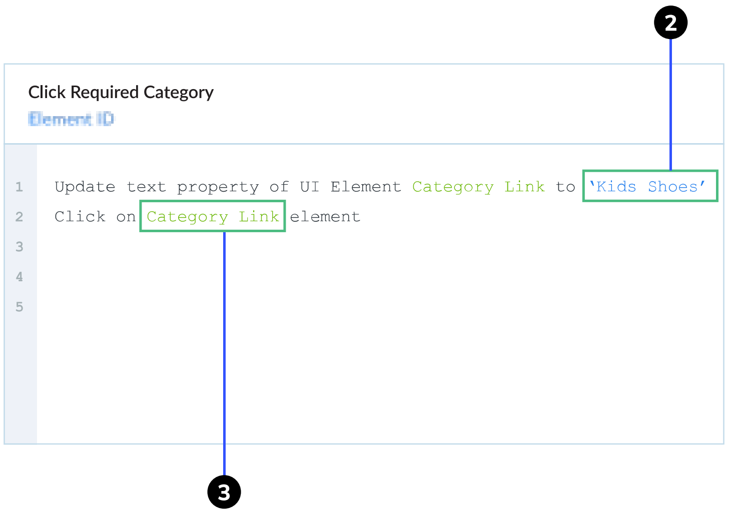
3 Steps to optimization

1. Define one of the category links with a generic name, “Category Link”.
2. In the action logic, update the text attribute with the value for the link you are interested. (`Update Element Text Property`).
3. In the action logic, operate on the element now. For example, a `click` operation.



Select Attributes to be used as identifiers

✓	text	Home Sale
✓	href	/catalog/.*
	aq-data	Home Sale
	title	--



```
Update text property of UI Element Category Link to 'Kids Shoes'
Click on Category Link element
```

Repeat Elements

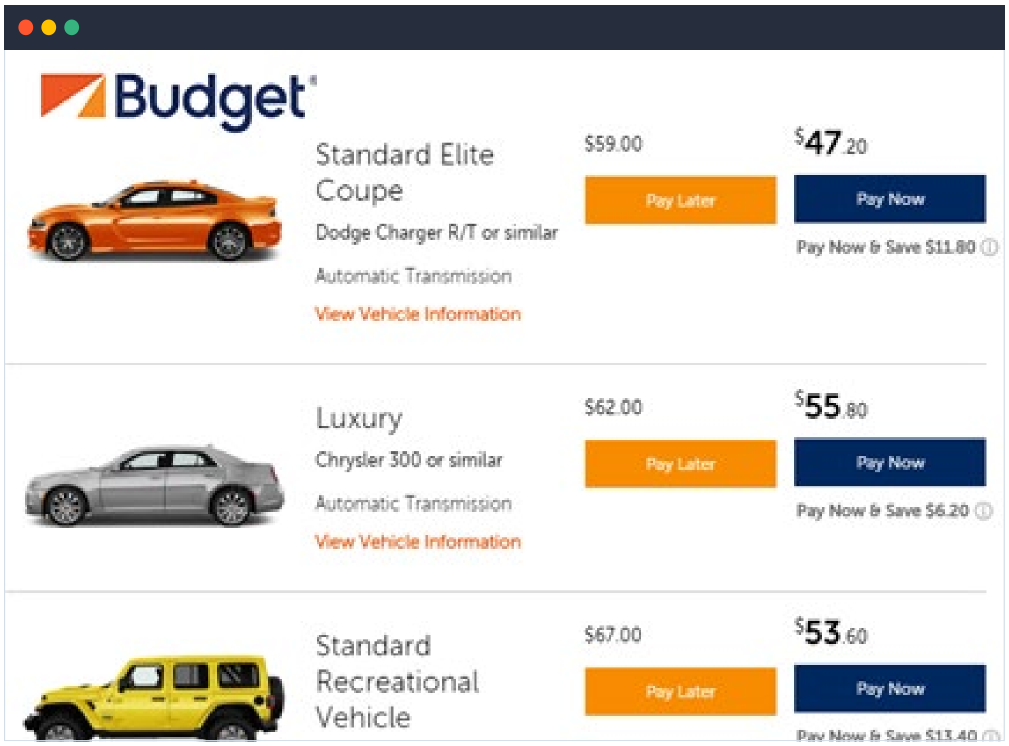
- Set of elements that repeat on a page, and functionally same. Page contains multiple instances of the same element!

- Differentiate from Similar Elements, which are group of equivalent elements, but not the same.

Uniqueness not to be expected

Characteristics of Repeat Elements

- Multiple occurrences of exactly the same element on a screen.
- Every element is functionally exactly same.
- Number of occurrences is not fixed and there is no need to differentiate one instance from the other.
- Don't look for *uniqueness* while setting up element ID.



Standard Elite Coupe
Dodge Charger R/T or similar
Automatic Transmission
View Vehicle Information
$59.00
$47.20
Pay Now & Save $11.80

Luxury
Chrysler 300 or similar
Automatic Transmission
View Vehicle Information
$62.00
$55.80
Pay Now & Save $6.20

Standard Recreational Vehicle
$67.00
$53.60
Pay Now & Save $13.40

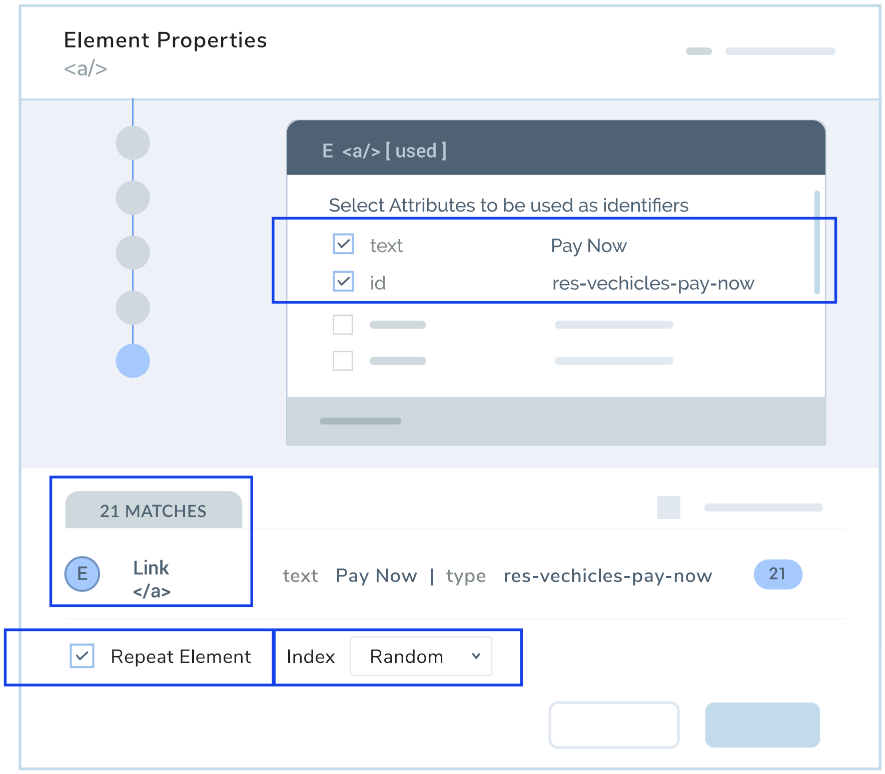
Working with Element Explorer

Setting up a Repeat Element

- Make sure the match-count displayed on the Element Explorer matches the number of occurrences displayed on the screen.
- Mark the checkbox for “Repeat Element” at the footer
- Setup an index policy based on the need. This refers to the specific instance you are interested.
  - First, Last, n-th from first, n-th from last, Random

Even when you add an element which is Repetitive in nature, ACCELQ might try to find attributes for unique match in the Element Explorer.

Be sure to modify the Selector (often by removing some attributes/nodes) to get multi-match count as per the page display.



✓ text	Pay Now
✓ id	res-vechicles-pay-now

text Pay Now | type res-vechicles-pay-now 21

✓ Repeat Element | Index Random

Generalizing repeat element access

Updating the *instance* of Repeat Element

- Your test logic may require pointing to a specific instance of Repeat Element
- Logic editor provides ability to override the `index` setting in the Element Explorer
- Useful commands
    - `Set Repeat Element by Ordinal Index`
    - `Set Repeat Element by Text`
    - `Set Repeat Element by Property`

```
Point the repeat element Pay Now Button, to an instance
with ordinal index'3'
Click on Pay Now Button element
```

Container Elements

Container is an element that encompasses multiple “related” elements in a page.

Elements related with a common characteristic

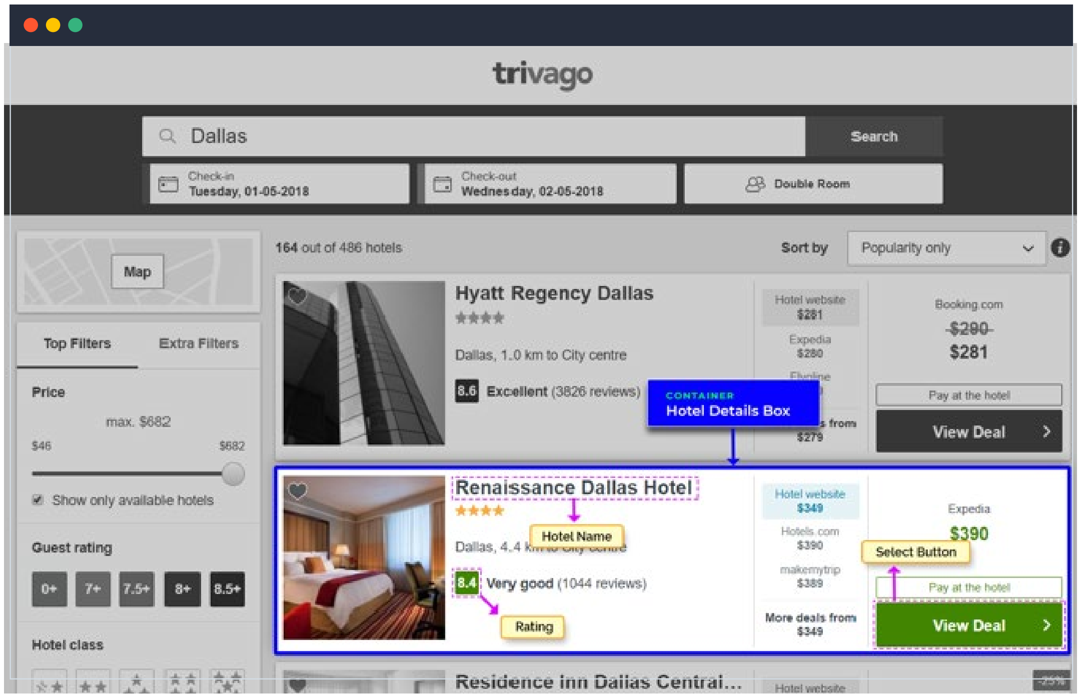
It’s all in the family!

In the example below, *specific instances* of Hotel name, Rating and Select Button are all part of one *specific* hotel we are interested in.

There are multiple instances of such elements in the page, but they belong to different hotels (families).

Container is any element that encompasses all the elements in the family we are interested in.

Hotel Details Box in the example here, is the Container.



In this example, the Container (Hotel Details Box) itself is a Repeat Element, but once you decide on a specific Container, members of that Container are not repeating.

Adding Container and member elements to repository

Step 1

Find the Container element and add it to the Repository. (Hotel Details Box)

Hover on the View and find a rectangle that encompasses all the member elements that we want to work with.

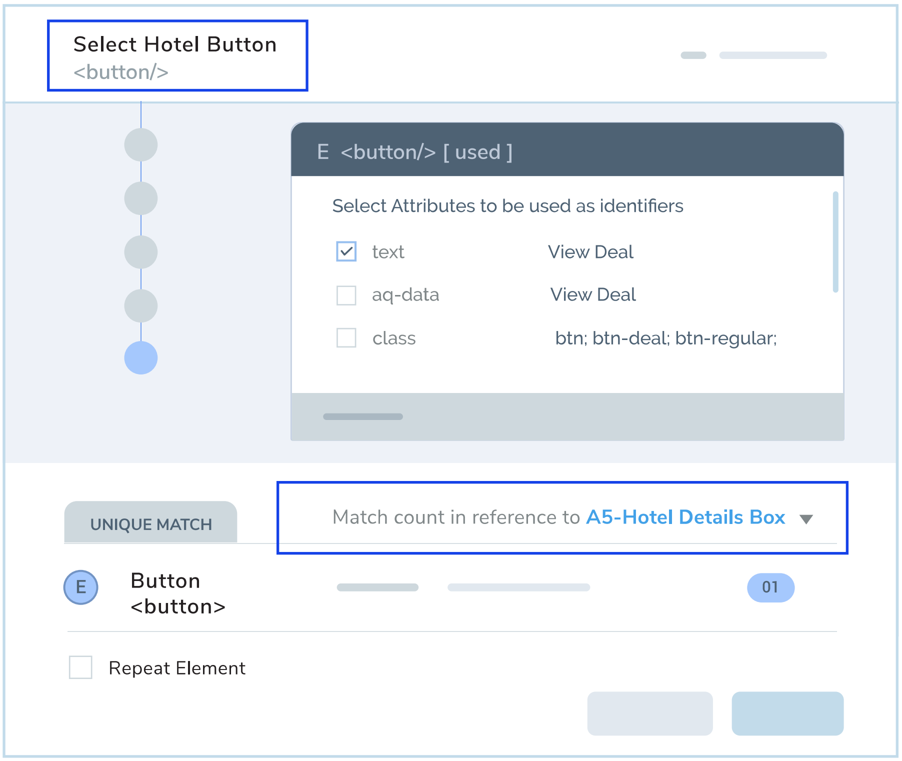
Container element may itself be a Repeat Element, as the same containers may be repeating multiple times in the page.

Step 2

Hover on a member element and add to repository. (Select Hotel Button)

Mark the Container element from step# 1 as the "reference" while defining the child element (figure below).

Unique identification of a member element is evaluated in the limited "context" of the Container element.



Select Hotel Button
<button/>

E <button/> [used]

Select Attributes to be used as identifiers

✓	text	View Deal
☐	aq-data	View Deal
☐	class	btn; btn-deal; btn-regular;

UNIQUE MATCH

Match count in reference to A5-Hotel Details Box

E Button <button> 01

☐ Repeat Element

Point to required Container & work with members

Updating Container instance in Action Logic

Step 1

Set Container element to required instance

Locate Container by
Text Content
Example: Select hotel by name.

Locate Container by
Child Element
Example: Selecting a hotel which has strike-through price displayed.

Locate Container by
Ordinal Index
Example: Selecting 3rd hotel from the top.

Locate Container by
Random Ordinal Index
Example: Selecting a random hotel

Step 2

Operate on the member/family element.

Just perform normal operations such as enter-text, get-element-text, click etc.

Since the Container is already set, no further tweaking necessary for the member.

```
Find a hotel with given name and get the pricing details
Locate Container element, Hotel Details Box by text content: contains 'Renaissance'
Get text from Best Price element. Store result in hotel best price
Click on Select Hotel Button element
```

ACCELQ

ACCELQ is the only cloud-based Continuous Test Automation and Test Management platform that seamlessly automates API and web testing without writing a single line of code. IT teams of all sizes use ACCELQ to accelerate their testing by automating critical aspects of lifecycle like test design, planning, test generation and execution.

ACCELQ customers typically save over 70% of the cost involved in the change and maintenance efforts in testing - addressing one of the major pain points in the industry. ACCELQ makes this possible with an AI-powered core that brings self-healing and resilience as a critical dimension to test development capability.

WWW.ACCELQ.COM San Francisco | Dallas | Hyderabad