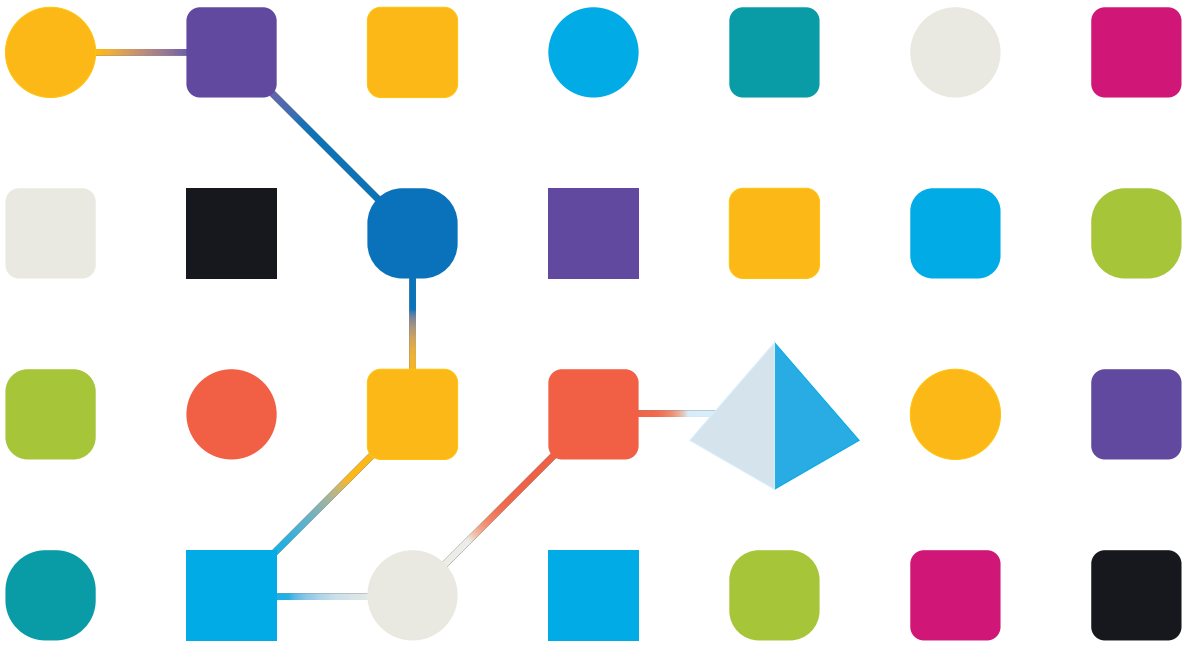
Blue Prism

Attribute Tuning Best Practices for Web Applications

Document Revision: 1.0



Trademarks and Copyright

The information contained in this document is the proprietary and confidential information of Blue Prism Limited and should not be disclosed to a third-party without the written consent of an authorized Blue Prism representative. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying without the written permission of Blue Prism Limited.

© 2023 Blue Prism Limited

"Blue Prism", the "Blue Prism" logo and Prism device are either trademarks or registered trademarks of Blue Prism Limited and its affiliates. All Rights Reserved.

All trademarks are hereby acknowledged and are used to the benefit of their respective owners. Blue Prism is not responsible for the content of external websites referenced by this document.

Blue Prism Limited, 2 Cinnamon Park, Crab Lane, Warrington, WA2 0XP, United Kingdom.
Registered in England: Reg. No. 4260035. Tel: +44 370 879 3000. Web: www.blueprism.com

Contents

Introduction

This attribute tuning best practice guide is aimed at arming SS&C | Blue Prism® developers and testers with the skills to fix compatibility issues brought on by migration to modern Chromium browsers. It also serves as a standards guide for developers who are interested in what to avoid when developing new Blue Prism automations.

Best practice guides

Best practice guides are recommended standards that define specific functions, installation configurations, technical capabilities, and APIs, or other controlling standards. They reflect the collective knowledge gained by senior Blue Prism developers and architects over years of developing and implementing scalable, robust, resilient automations.

The purpose of best practice guides is to remove ambiguity by saying, "this is the standard recommendation for coding or configuring a Blue Prism process, action, or capability for best outcome". They should be viewed as a supplement to current documentation, not a replacement.

Best practices are not tied to a specific Blue Prism version or product extension (for example, browser plug-in).

Prerequisites

Best practice guides assume readers have mastered Blue Prism development and implementation to an advanced level.

For more information on Blue Prism fundamentals, please consult the following Blue Prism University courses:

- Blue Prism Foundation
- Blue Prism Advanced Consolidation Exercise (ACE)
- Attribute Matching Guide
- Blue Prism – Solution Design Overview
- Object Design Guide
- Browser Automation Guide

Assertions

Implementing successful attribute tuning is based on the following assumptions:

- A web browser of some kind is being used.
- Application Modeller is used to discover and define the element attributes used.

Internet Explorer – The end of a legacy and what that means for your company

All organizations with applications that were created specifically for the legacy Internet Explorer browser face a dilemma. On the one hand, they can continue using the unsupported web browser to keep their legacy applications working as previously implemented. But doing so may expose the enterprise to security risks and privacy issues. Alternatively, they can migrate to a modern web browser and risk having application compatibility issues break their automations.

Cross-browser inconsistency

Application developers have long wrestled with coding for the proliferation of browsers used by end users viewing web pages. Browser-based applications behave differently in different browsers, in different resolutions, and sometimes in different operating systems. This inconsistency between browsers and resultant faulty HTML/CSS rendering is exacerbated by the recent requirement to migrate from legacy Internet Explorer browser applications to modern Chromium browsers like Google Chrome and Microsoft Edge.

Blue Prism automations are not immune to the cross-browser inconsistencies and faulty rendering behaviors of browser-based applications.

Regression testing

To overcome these problems, tuning Blue Prism Application Modeller attributes and regression testing are integral to the quality assurance process for automations that interact with the user interface (UI) of browser-based applications. Changes to how the UI is rendered will, in some cases, require changes to the automation technology.

Web sites – The RPA challenge

Web pages

A web page is a document that is hosted on a remote web server and accessed by entering a unique address, known as a uniform resource locator (URL), in a web browser, such as Google Chrome or Microsoft Edge. A web page is created as a text file, written in Hyper Text Markup Language (HTML), which the web browser uses to determine how to display the contents of the page to the user.

A web site is a collection of related web pages that are accessed from the same domain address.

Automation developer versus web sites

Early static web pages, which described all page elements at render time, made the automation developer's job much easier. Static web page elements, like buttons, links, input fields, and navigation locations, were easily detected by Blue Prism. Even when page elements were moved, a Blue Prism automation could detect the location using page attributes, rather than page coordinates.

Enter dynamic web pages and server-side scripting

To improve the user experience and application performance, today's web sites have moved away from static pages and have become very dynamic. A dynamic web page is a web page whose construction (rendering) is controlled by an application server processing server-side scripts (for example, JavaScript).

Adding JavaScript to dynamic web pages enabled web developers to implement client-side, event-driven page elements to dynamically display input fields, links, buttons, and navigation that are driven by the actions taken by the end user. This decreased roundtrip web server lag time and improved end user experience opened the door to delivering robust, responsive web applications.

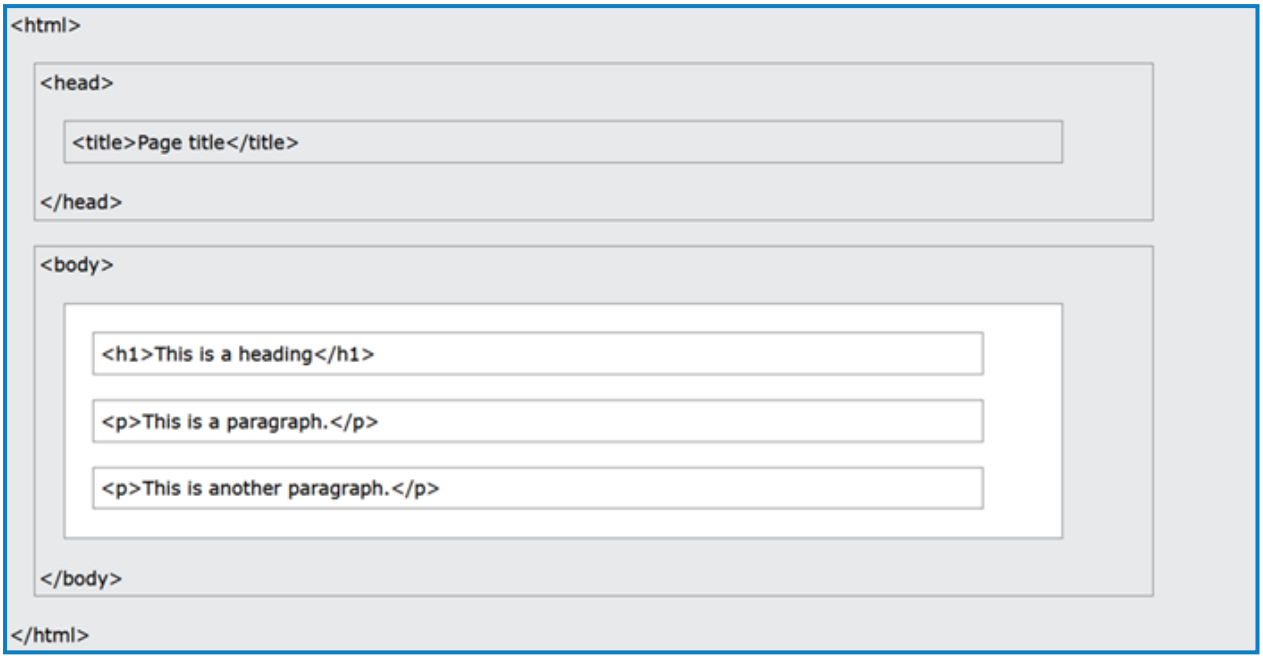
RPA and dynamic pages

Dynamically rendered web pages make the robotic process automation (RPA) developer's job challenging, as the RPA tool needs to have a consistent, unambiguous way to identify and interact with page elements that are dynamically rendered based on the interaction between the end user and the application server.

Web site structure

Web sites are built around a basic structural Document Object Model (DOM). The fundamental architecture that governs the layout and navigation of the web site is designed to inform the user's visual senses and guide them to information and how it is organized.

An example of a basic HTML page structure is shown below:



HTML5 imposes relatively forgiving rules when rendering web page layouts, giving web developers a great deal of flexibility in page design and implementation.

Categories of HTML tags

HTML tags are broken into multiple categories. Those categories include basic HTML, formatting, forms and input, images, audio/video, links, lists, tables, styles, and semantics (for example, section and article).

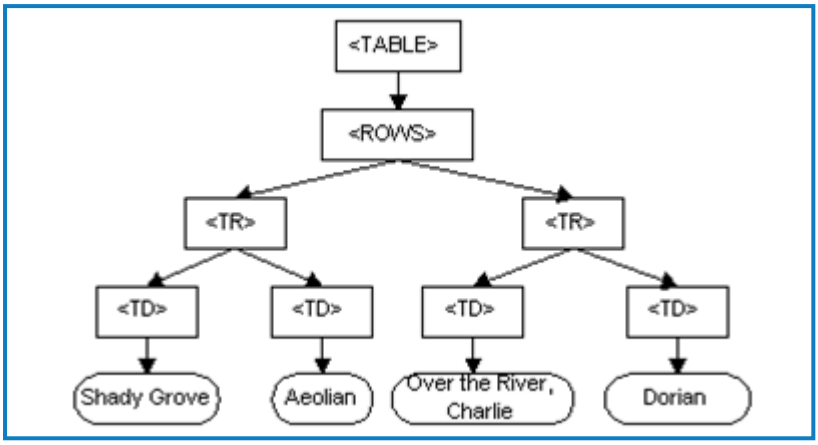
Page tags, like <SECTION> and <DIV>, create the hierarchy of page content and the scope. Tags like <H1> and <P> describe how content is displayed, while tags like <INPUT> and <TABLE> describe where end user input is expected.

However, the ability to use elements with page tags like <A>, <UL>, and <SPAN>, or tag modifiers, like class, id, and inline style, give web pages nearly infinite rendering flexibility. This same flexibility can present a challenge to RPA tools like Blue Prism.

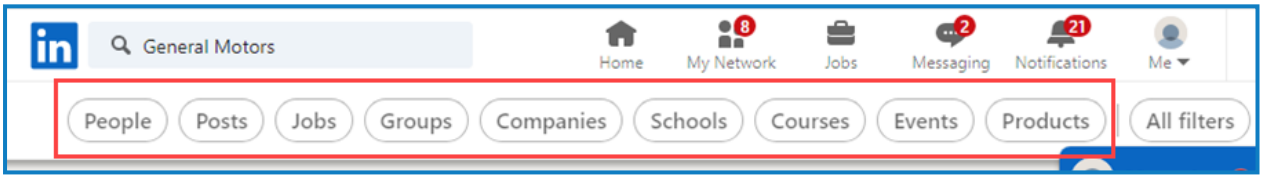
Example: This data from https://www.w3.org/TR/WD-DOM/introduction.html shows the structure of a simple table:

```
<TABLE>
    <ROWS>
        <TR>
            <TD>Shady Grove</TD>
            <TD>Aeolian</TD>
        </TR>
        <TR>
            <TD>Over the River, Charlie</TD>
            <TD>Dorian</TD>
        </TR>
    </ROWS>
</TABLE>
```

The Document Object Model (DOM) is shown here:



Particularly in the case of a table structure, the developer may find that an element has one or more tags that change value each time the target page is loaded. For example, on the LinkedIn site, the buttons shown in the image below don't always render in the same order:



Creating Blue Prism automations

When creating a web page automation, Blue Prism must be able to uniquely identify a given element without ambiguity. Looking at the DOM example above, there are two table rows (TR) with four table data values (TD), which have text values of "Shady Grove", "Aeolian", "Over the River, Charlie", and "Dorian".

A Blue Prism developer could define uniqueness for a given element in a number of ways. They could find the TD that has a value of "Shady Grove", but could also define uniqueness by the HTML path/XPath to select nodes down a hierarchical relationship path, such as ...<TABLE><TR><TD>Shady Grove</TD>.

Another method that would also find uniqueness would be to find the first occurrence of ...<TABLE><TR><TD>, which, in our example, would be "Shady Grove". It is the ability to define a unique set of attributes for a given element that enables Blue Prism to quickly identify a specific element that is both reliable and performant.

Getting to know the target web site

One of the most important steps to be taken in preparing for your automation is to understand the nature of the target web site. As the target web site is the sole entity that Blue Prism will be working with, a significant amount of time should be invested in this exercise. This can be done using Blue Prism tools and supplemented by browser development tools that are supplied with modern browsers, such as Chrome.

It is beneficial to involve a business subject matter expert (SME), as they will have hands-on experience with the site. The business SME need not be technical, but their insight into the nature of the site will be invaluable.

Analyzing a web site

Analyzing a web site requires a combination of visual observations and web page element inspection.

Visual clues

The developer should look at the web page's navigation, buttons, and links to observe how the hyperlink (or URL) navigates document presentation. This is the same perspective that an end user would experience interacting with the web application.

Navigation

Navigation is about the clues that direct users to specific locations in the web application. The navigation flow is all about how the site behaves when moving from page to page (or section to section). Navigation includes, but is not limited to, top, left, and right menus (or shortcuts). In some web designs, a top menu will navigate to a new category of the web site (like a table of contents pointing to chapters). Left and right rail menus frequently drill deeper into sub-categories.

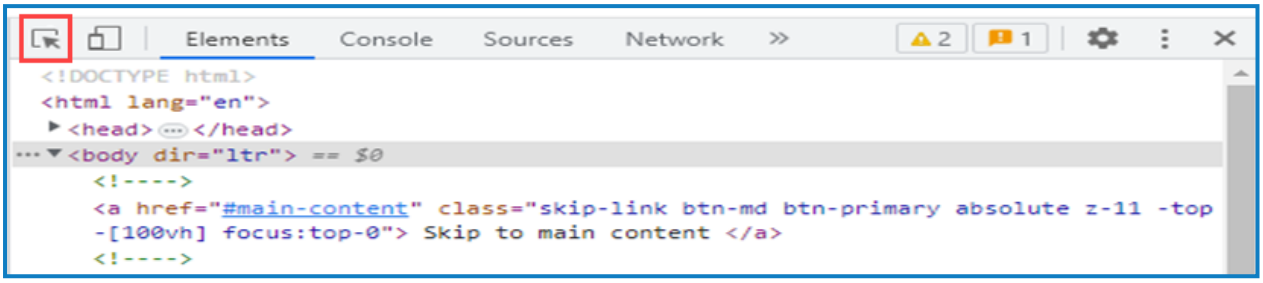
The developer should pay attention to the URLs used, as they may indicate whether they navigate to a new category, or to a sub-category. A URL specifies both the protocol (secure or unsecure), and the name or number of a target page. A Blue Prism developer can leverage the URL to navigate by a fully qualified URL path, or use a wildcard character (*) that will allow a digital worker to verify screen location, and/or dynamically build the URL pointer based on the automation functionality desired (for example, search, create, or change). The developer may also want to utilize shortcut keys that are available, to quickly navigate to a certain page or function, such as log out.

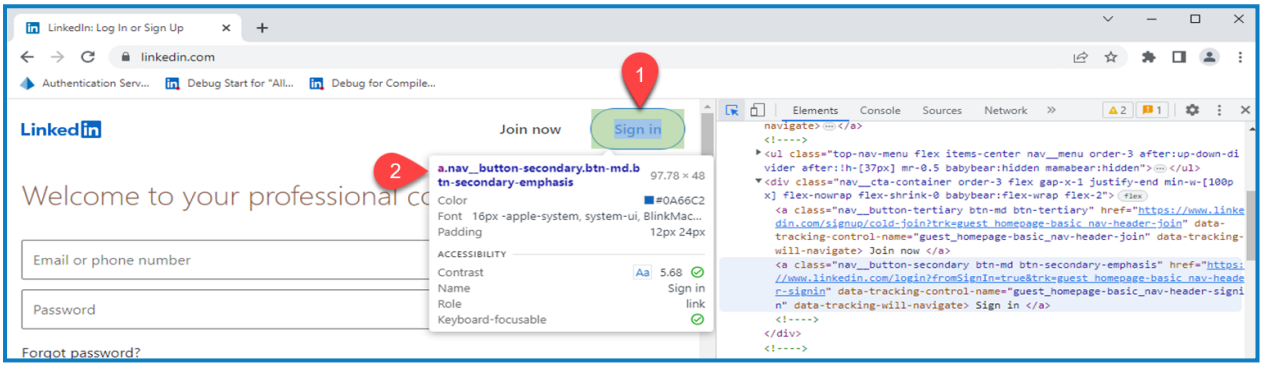
Web page inspection

The second method used by Blue Prism developers to improve browser automation is using a web developer tool to inspect the web page HTML code and CSS. Blue Prism Application Modeller provides multiple methods to identify (or spy) web page elements. This works well for most web page elements. However, there are instances when looking at the web page structure is the only option.

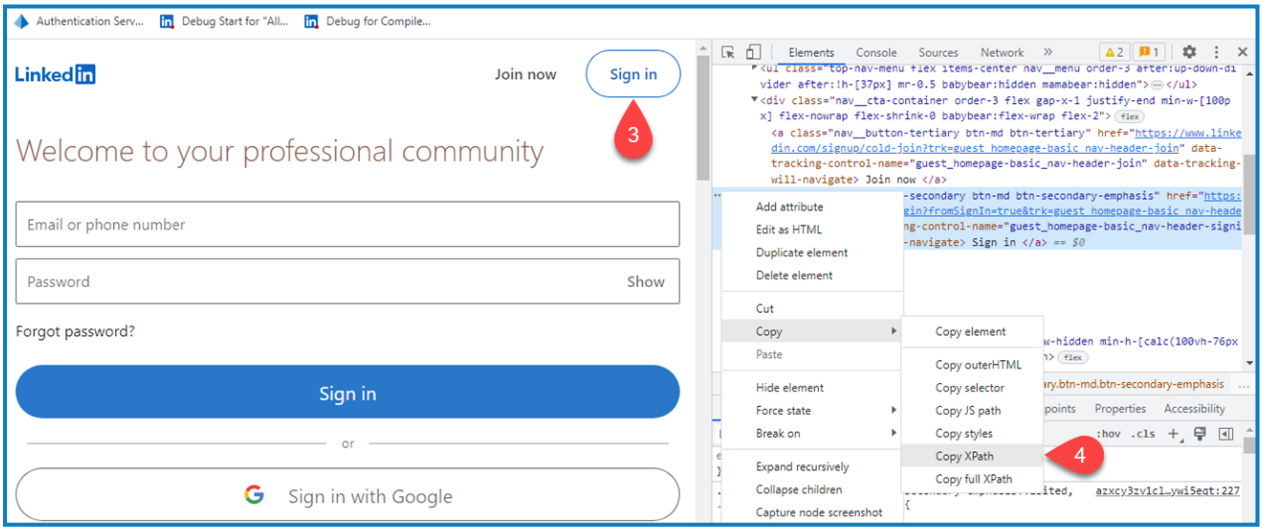
Web page inspection tools can enable a developer to drill into the actual page code, which may be helpful when HTML page elements are nested, or coding hygiene does not properly open or close HTML tags. These tools can also reveal HTML and script errors that prevent proper identification of web page elements.

For example, Chrome provides developer tools that enable the developer to identify the structure of a given web page, much like Blue Prism spying, but more in-depth. Once the Chrome developer tools are selected, the following view is presented:



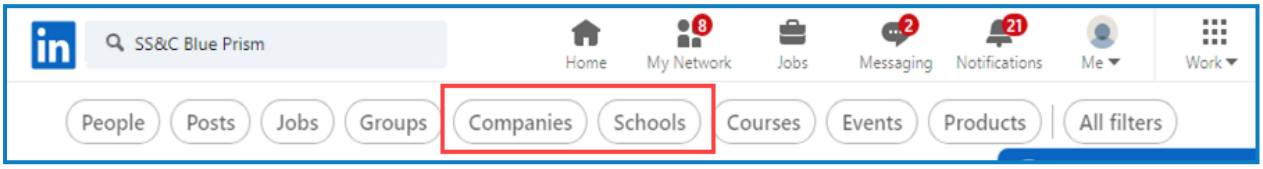


Once a given element is selected (3), the developer can then copy information for that element, such as XPath (4), to their clipboard:



Other attributes, such as Selector (CSS Selector) can also be copied and used within Blue Prism. In summary, the developer has many tools available to help them determine the best approach to working with a given element or web site.

The developer can use Notepad to compare various attribute values. For example, the developer may find that the only difference from one element to the next is a single attribute value, as shown in the images below. The HTML path to the Companies button and the Schools button differs by only the value of the LI tag (LI[5] for the Companies button and LI[6] for the Schools button). For this situation, the developer may determine that using a dynamic attribute is preferable to using a different element for each button.




```
/HTML[1]/BODY[1]/DIV[6]/DIV[3]/DIV[2]/SECTION[1]/DIV[1]/NAV[1]/DIV[1]/UL[1]/LI[5]/BUTTON[1]
/HTML[1]/BODY[1]/DIV[6]/DIV[3]/DIV[2]/SECTION[1]/DIV[1]/NAV[1]/DIV[1]/UL[1]/LI[6]/BUTTON[1]
```

Observation

There are any number of hurdles that web page code may present to an automation developer. Some are by design, like the use of unique session identifiers or GUIDs embedded in URL paths. Other hurdles may be the result of the dynamic nature of CSS and JavaScript. HTML tag modifiers are designed to allow web interaction to easily reference groups of similar elements by their class name. Examples might include the use of branding (for example, font types and color) that is applied to content, sections, and viewports (screen sizes). An automation developer can view these coding techniques using a browser development tool to inspect the code for areas that may require additional anchors to uniquely identify a unique row or data value.

Performance

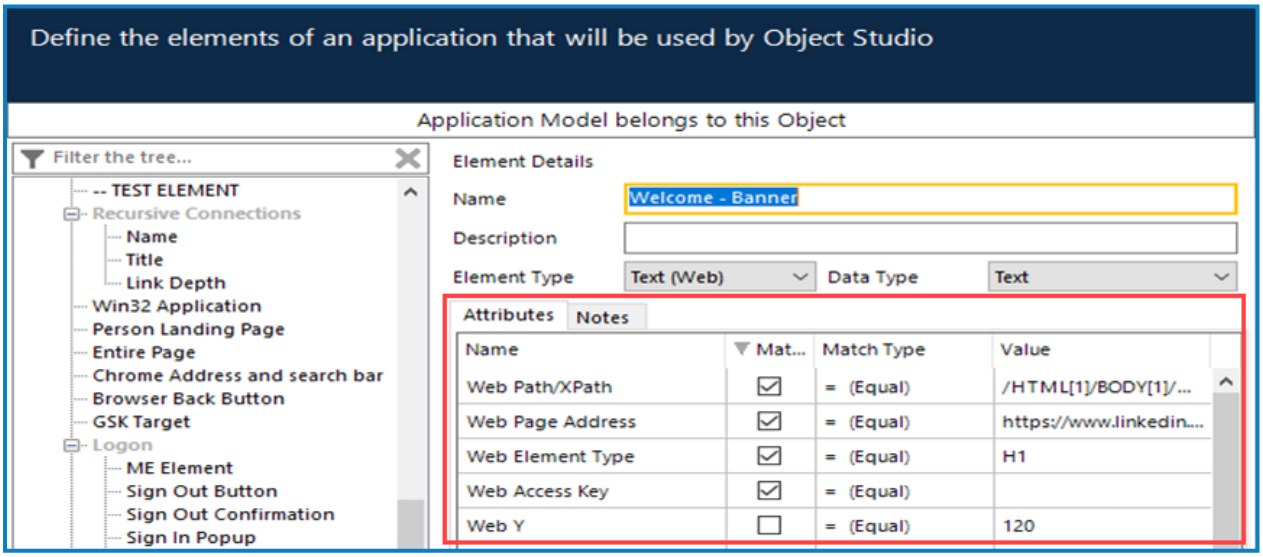
The benefit of automation is the ability to perform work more quickly and accurately than its human partner. Using the URL to bypass navigation hierarchy clicks can improve automation performance. Using the URL as a shortcut, by sending the URL path to the browser's address bar (using a Write stage) and using simulating keys and send key events like ENTER can be executed using Global Send Keys (for more information, see the "Guide to Send Keys and Send Keys Events" from the Blue Prism University). Using tag modifiers can enable developers to quickly identify content, links, and input fields to rapidly control the behavior of the page.

Application Modeller – An example for tuning attributes

The primary tool used to discover the various elements on a given web page is the Application Modeller. The developer can use their mouse to point to a given element, then Blue Prism attempts to ascertain an initial set of element attributes that identify the given element. The Application Modeller selects a default set of attributes as a baseline. In most cases, this initial selection of attributes will not be optimized. It is incumbent on the developer to have an understanding of which attributes work best for a given situation. Application Modeller inspects the construction of the page elements and associated attributes of the selected page element spied. Spying identifies the baseline structure of attached pages, providing the developer with attributes that can be tuned to achieve the most performant and reliable interaction possible. In all situations, the developer should try various spy modes to uncover which may provide the most performant and resilient options.

Example: The following is an illustration that highlights how a Blue Prism developer can spy a dynamic single page application (SPA) to discover an attribute, such as a welcome banner on the LinkedIn web page:

Once the developer points to the welcome banner, Blue Prism will choose a default set of attributes, such as:



- **Web Path/XPath** – The **Web Path/XPath** has been selected. For modern, dynamic web sites, the Web Path/XPath value will change dynamically. It cannot be used as a static anchor value.

  ⚠ Selecting this value will cause errors that result from the dynamic value changing in the web page. If the web site dynamically updates the value, or its web developer changes the path, then Blue Prism will fail as the original path will no longer exist. You can set this attribute with a Match Type of Dynamic and read it in every time, however, this is an advanced technique and caution should be used.

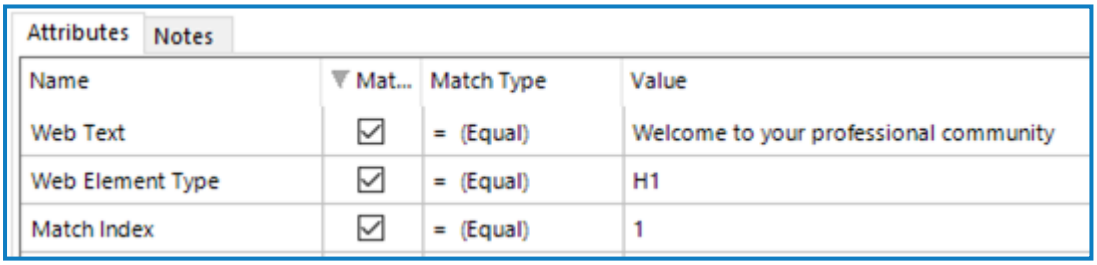
- **Web Page Address** – The **Web Page Address** contains the page address element URL value. A static value indicated here will only work when viewing the exact URL address. This will limit the ability to create reusable objects. Caution is also advised here, as the link provided could be changed, which would again cause Blue Prism to fail.
- **Web Element Type** – The **Web Element Type** is selected and has the value H1, which stands for header 1. This indicates the level of the heading and is typically the first heading on a page. Though the web site developer could certainly change the appearance of this value, it is probably a more reliable attribute to use in the Application Modeller.
- **Missing Attribute Values** – One (or more) of the selected attributes may be null. This indicates that the attribute cannot be used to identify the page element. Any attribute that Blue Prism identifies with a null value should be unchecked to limit stability issues.

Initial tuning

It should be noted that the process of tuning or selecting the correct combination of attributes is iterative and highly dependent on the target web site. What works perfectly on one web site may not work at all on another web site.

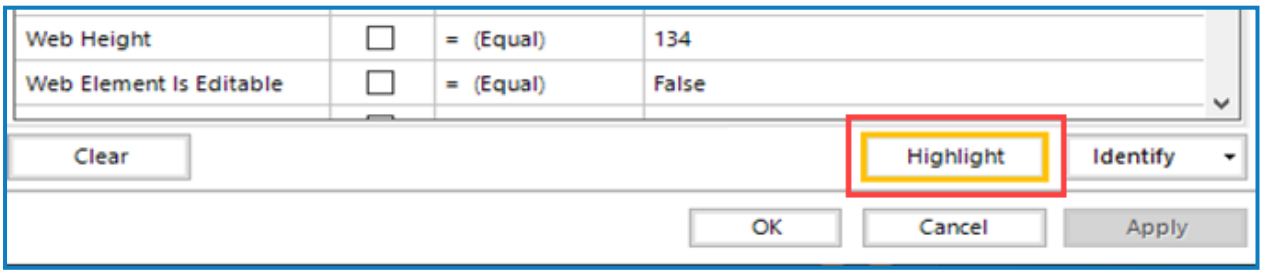
Similarly, what works in one browser is likely to require testing and attribute tuning if the automation is updated to work with a different browser. The recommended standards in this guide should be accompanied by regression testing for each browser. The developer should use the tools described in this guide to understand the page code and attributes available and make the best decision moving forward.

Once the element has been discovered, and the Blue Prism developer has tuned the attribute list, the following attributes are selected:



Match Index is selected, and all attributes and blank values have been deselected. Also, note that the list of attributes is very short, only three in the example above. For performance reasons, this list should be as short as possible, balancing performance with reliability.

After the initial selection of attributes has been made, the developer then clicks the **Highlight** button:

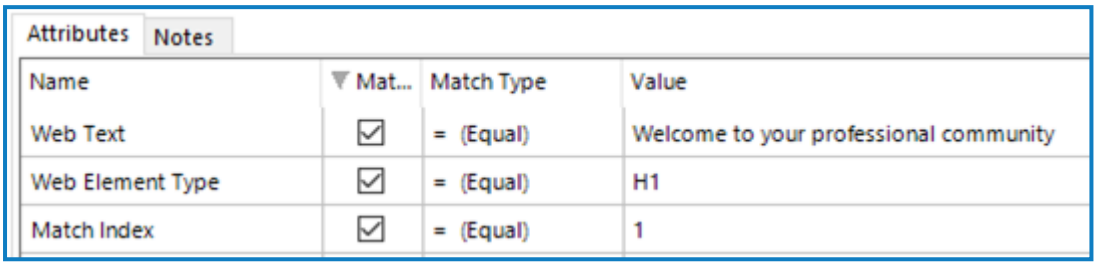


Then notes how long it takes Blue Prism to identify the given element by surrounding it with a red border:

The developer should try various spying identification modes to determine and improve how quickly each element is found (highlighted in a red box).

Attribute tuning – An iterative process

With the following set of attributes selected, the response time to highlight the welcome banner in the example was about one second, which is certainly respectable in regard to performance.

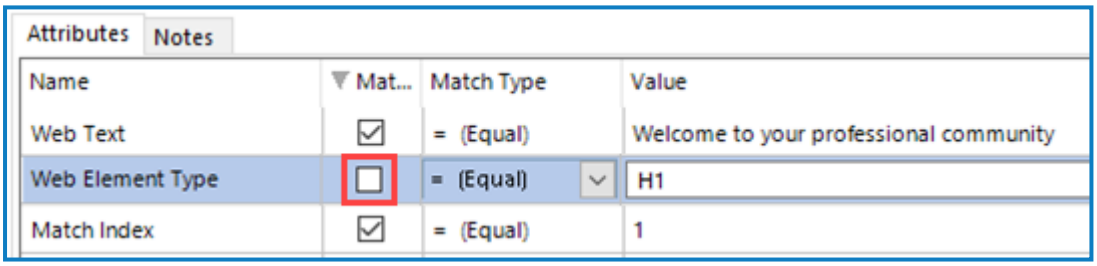


By selecting a web page anchor tag value like the H1 welcome banner text "Welcome to your professional community", the developer commits to its likelihood of being found consistently.

Setting the Match Index to 1 tells Blue Prism to look for the attribute starting at the top of the DOM and stop searching once it finds the first instance of an H1 element that has the defined text string.

The Match Index ensures uniqueness, but does not guarantee correctness. This is the job of the developer, possibly assisted by the business SME, who works with the site regularly.

In the interest of trying to cut down the list of attributes even further, the developer might deselect the **Web Element Type**, which in this example has a value of H1, as shown below:



The developer clicks **Apply**, then **Highlight**, and then notes that it now takes Blue Prism around 30 seconds to identify and highlight the selected element. Considering that a given application model can have hundreds of elements, it is apparent that improper attribute tuning can make the difference between a high performing automation or an attribute that is underperforming in terms of speed.

Developers must endeavor to tune each element used, so that they are performant and reliable. As this example shows, no matter how much memory or CPU is added to the associated Blue Prism server, resource, or database, nothing can overcome a poorly tuned automation.