# web crawler

A web crawler, also known as a spider, is an automated software program that systematically scans and indexes websites on the internet. The primary purpose of a web crawler is to gather information about web pages, including their content, links, and other data, so that search engines can provide users with relevant search results.

Web crawlers start by visiting a seed URL, which is typically provided by the search engine, and then follow links on that page to other pages on the same website and other websites. As the crawler visits each page, it collects information about the page and adds it to a database, which is used by search engines to generate search results.

Web crawlers are used by search engines to create their indexes and by other applications, such as web archiving tools and data mining applications, to gather information about the web. However, web crawling can also be used for malicious purposes, such as harvesting email addresses or scraping sensitive dat

# XPath

XPath (XML Path Language) is a language used to navigate through XML/html documents and select specific elements or attributes. It is a query language that can be used to extract data from an XML document, and it provides a way to navigate the hierarchical structure of an XML document.

XPath expressions are used to identify specific parts of an XML document by matching patterns of elements or attributes. For example, an XPath expression can be used to select all elements with a certain tag name, or all elements with a certain attribute value. XPath is commonly used in web scraping and parsing XML data in programming languages such as Python and Java.

XPath has a rich set of operators and functions that allow you to manipulate and extract data from an XML document. Some common XPath expressions include selecting all elements of a certain type, selecting elements based on their attributes, and selecting elements based on their position in the document.

# CSS Selector

CSS Selector is a pattern that is used to select and style HTML elements based on their attributes, such as their tag name, class, ID, or other properties. CSS Selectors are a fundamental part of Cascading Style Sheets (CSS), which is used to define the visual style and layout of web pages.

CSS Selectors allow web developers to target specific elements or groups of elements in an HTML document, and apply styles such as color, font, size, and positioning. They can be used to select elements by their tag name, class, ID, attribute values, and their relationships with other elements in the document, such as parent-child or sibling relationships.

There are many types of CSS Selectors, including the universal selector, type selector, class selector, ID selector, attribute selector, descendant selector, child selector, adjacent sibling selector, and more. Each selector has its own syntax and behavior, and they can be combined to create complex selection patterns.

CSS Selectors are commonly used in web development to create responsive and visually appealing websites, and they are supported by all modern web browsers.

# selenium webdriver

Selenium WebDriver is a popular open-source framework for automating web browsers. It provides a way to programmatically interact with a web page, including clicking links, filling out forms, and simulating user input. WebDriver supports multiple programming languages, including Java, Python, and JavaScript, and works with a variety of web browsers, including Chrome, Firefox, and Safari.

WebDriver works by sending commands to a browser's driver, which is a separate executable that provides a way to communicate with the browser. The driver executes the commands and returns the results to the WebDriver instance. This allows developers to write tests and scripts that can simulate user interactions with a web page, allowing them to automate repetitive tasks and perform more thorough testing of web applications.

Selenium WebDriver is widely used in the software development industry for web testing and web scraping, as well as for other tasks that involve automating web browser interactions. Its popularity is due to its flexibility, ease of use, and the fact that it is an open-source tool that is supported by a large community of developers.

# Examples

example of using XPath with HTML using the same scenario as before - extracting the titles of all books published after 1950:

Suppose we have an HTML document that contains a list of books and their authors, and we want to extract the titles of all the books that were published after a certain year. We can use the **span** element and the built-in **substring()** function in XPath to filter the books by publication year.

<!DOCTYPE html>

<html>

<head>

<title>Books</title>

</head>

<body>

<div class="book">

<span class="title">Harry Potter and the Philosopher's Stone</span>

<span class="author">J.K. Rowling</span>

<span class="year">1997</span>

</div>

<div class="book">

<span class="title">The Great Gatsby</span>

<span class="author">F. Scott Fitzgerald</span>

<span class="year">1925</span>

</div>

<div class="book">

<span class="title">1984</span>

<span class="author">George Orwell</span>

<span class="year">1949</span>

</div>

</body>

</html>

Here's an example code snippet in Python that uses XPath to extract the titles of all books published after 1950:

from lxml import html

# Parse the HTML document

doc = html.parse('books.html')

# Use XPath to find all span elements with a year greater than 1950

titles = doc.xpath("//div[span[@class='year'] > '1950']/span[@class='title']")

# Loop through the selected titles and extract their text content

for title in titles:

print(title.text\_content())

This code uses the **parse()** method from the **lxml** library to parse the HTML document. It then uses XPath to select all **div** elements that contain a **span** element with a **class** attribute equal to **'year'** and a value greater than 1950. It then selects the **span** element with a **class** attribute equal to **'title'** that is a child of the selected **div** elements. Finally, it loops through the selected titles and extracts their text content using the **text\_content()** method.

Note that in the XPath expression, we use the **>** operator to compare the year element with a string representing the year 1950. We also use the **text\_content()** method to extract the text content of the **title** element in the Python code.

Xpath tutorial

<https://www.w3schools.com/xml/xpath_intro.asp>

https://www.tutorialspoint.com/xpath/index.htm