XPath Guide

XPath (XML Path Language) is a powerful language for selecting nodes from an XML document. It can be used in various contexts, including HTML documents, to navigate through elements and attributes. Here are the details of XPath, covering its syntax, expressions, and functions.

Basic XPath Syntax

- / : Selects from the root node.
- // : Selects nodes in the document from the current node that match the selection no matter where they are.
- . : Selects the current node.
- .. : Selects the parent of the current node.
- @ : Selects attributes.

XPath Expressions

1. Selecting Nodes

- Absolute Path: Starts from the root node.

  ```
  /html/body/div
  ```

- Relative Path: Starts from the current node.

  ```
  div[@class='product']
  ```

2. Wildcards

- Any Node (*): Matches any element node.

  //div/*

- Any Attribute (@*): Matches any attribute.

  //div[@*]

3. Predicates

- Used to find a specific node or a node that contains a specific value.

  //div[@class='product']

  //div[1] // First div element

  //div[last()] // Last div element

  //div[position()<3] // First two div elements

4. Axes

- Child (child::): Selects children of the current node (default axis).

  child::div

- Descendant (descendant::): Selects all descendants of the current node.

  descendant::div

- Parent (parent::): Selects the parent of the current node.

  parent::div

- Following (following::): Selects everything in the document after the closing tag of the current node.

following::div

- Preceding (preceding::): Selects all nodes that appear before the current node in the document.

preceding::div

- Self (self::): Selects the current node.

self::div

XPath Functions

1. Node Functions

- text(): Selects the text content of a node.

//div[@class='product-name']/text()

- node(): Selects any type of node.

//div[@class='product']/node()

2. String Functions

- contains(): Checks if a string contains a substring.

//div[contains(@class, 'product')]

- starts-with(): Checks if a string starts with a substring.

  //div[starts-with(@class, 'product')]

- string-length(): Returns the length of a string.

  //div[string-length(@class)=7]

- substring(): Returns a part of a string.

  //div[substring(@class, 1, 7)='product']

3. Number Functions

- position(): Returns the position of a node.

  //div[position()=1]

- last(): Returns the last position of a node.

  //div[last()]

4. Boolean Functions

- not(): Negates a condition.

  //div[not(@class='product')]

Examples in HTML Context

Given the HTML structure provided earlier, here are a few more examples:

XPath Guide

1. Select all product names:

   //div[@class='product-name']/text()

2. Select the price of the second product:

   (//div[@class='product']/p[@class='product-price'])[2]

3. Select all buttons within products:

   //div[@class='product']//button

4. Select the product names of products priced over 100:

   //div[@class='product'][p[@class='product-price'] > 100]/h4[@class='product-name']

Conclusion

XPath is a versatile language for navigating and selecting elements within XML or HTML documents. By understanding its syntax and functions, you can craft precise queries to locate the nodes you need. The key is to understand the structure of your document and how to traverse it using XPath expressions.