RIFERIMENTO RAPIDO XPATH

Assi, predicati, funzioni, operatori, selezione nodi

Sintassi

Espressioni di Percorso

`/`	Nodo radice (inizio percorso assoluto)
`/bookstore/book`	Selezione del figlio diretto
`//book`	Seleziona tutti i nodi book ovunque
`.`	Nodo di contesto corrente
`..`	Genitore del nodo corrente
`@lang`	Attributo chiamato lang
`node()`	Qualsiasi nodo di qualsiasi tipo
`*`	Qualsiasi nodo elemento
`@*`	Qualsiasi attributo

Esempi di Base

```
/html/body/div          # percorso assoluto verso <div>
//input[@type='text']   # tutti gli input di testo
//div[@class='main']/*  # figli di div.main
//a/@href               # tutti gli attributi href
```

Combinazione di Percorsi

```
//book/title | //book/price   # unione di due percorsi
//h1 | //h2 | //h3           # più tipi di elementi
```

Assi

Direzioni degli Assi

`child::`	Figli diretti (asse predefinito)
`parent::`	Genitore diretto
`ancestor::`	Tutti gli antenati fino alla radice
`ancestor-or-self::`	Antenati + nodo corrente
`descendant::`	Tutti i discendenti
`descendant-or-self::`	Discendenti + nodo corrente
`following::`	Tutti i nodi dopo il corrente nel documento
`following-sibling::`	Fratelli dopo il corrente
`preceding::`	Tutti i nodi prima del corrente nel documento
`preceding-sibling::`	Fratelli prima del corrente
`self::`	Solo il nodo corrente
`attribute::`	Attributi del nodo corrente
`namespace::`	Nodi namespace

Esempi di Assi

```
//div/child::p             # figli <p> di <div>
//td/parent::tr            # genitore <tr> di <td>
//h2/following-sibling::p  # <p> dopo <h2>
//li/ancestor::ul          # <ul> che contiene <li>
```

Predicati

Filtro con Predicati

```
//book[1]                  # primo elemento book
//book[last()]             # ultimo elemento book
//book[position() < 3]     # primi due book
//book[@lang='en']         # book con lang="en"
//book[price > 30]         # book con prezzo > 30
```

Pattern di Predicati

`[n]`	Elemento in posizione n (base 1)
`[last()]`	Ultimo elemento
`[last()-1]`	Penultimo elemento
`[@attr]`	Ha l'attributo
`[@attr='val']`	Attributo uguale al valore
`[element]`	Ha un elemento figlio
`[element='text']`	Elemento figlio contiene il testo
`[not(@attr)]`	Non ha l'attributo

Predicati Concatenati

```
//div[@class='list']//a[1]         # primo <a> in div.list
//input[@type='text'][@name='q']   # condizione AND
//book[price>30][@lang='en']       # condizioni multiple
```

Funzioni

Funzioni Stringa

`contains(s, sub)`	Vero se s contiene sub
`starts-with(s, pre)`	Vero se s inizia con pre
`string-length(s)`	Lunghezza della stringa
`normalize-space(s)`	Rimuove spazi iniziali/finali e comprime
`concat(a, b, ...)`	Unisce le stringhe
`substring(s, pos, len)`	Estrae sottostringa (base 1)
`translate(s, from, to)`	Sostituzione carattere per carattere

Funzioni Numeriche

`sum(node-set)`	Somma dei valori numerici
`count(node-set)`	Numero di nodi
`floor(n)`	Arrotonda verso il basso
`ceiling(n)`	Arrotonda verso l'alto
`round(n)`	Arrotonda all'intero più vicino
`number(val)`	Converte in numero

Esempi di Funzioni

```
//div[contains(@class, 'active')]
//a[starts-with(@href, 'https')]
//p[string-length(text()) > 100]
//ul[count(li) > 5]
```

Operatori

Operatori di Confronto

`=`	Uguale
`!=`	Diverso
`<`	Minore di
`<=`	Minore o uguale
`>`	Maggiore di
`>=`	Maggiore o uguale

Logici e Aritmetici

`and`	AND logico
`or`	OR logico
`not()`	NOT logico (funzione)
`+`	Addizione
`-`	Sottrazione
`*`	Moltiplicazione
`div`	Divisione
`mod`	Modulo
`|`	Unione di set di nodi

Esempi di Operatori

```
//book[price > 20 and price < 50]
//item[@type='a' or @type='b']
//span[not(contains(@class, 'hidden'))]
```

Test dei Nodi

Tipi di Nodo

`node()`	Qualsiasi nodo (elemento, testo, commento, PI)
`text()`	Solo nodo di testo
`comment()`	Solo nodo commento
`processing-instruction()`	Nodo istruzione di elaborazione
`*`	Qualsiasi nodo elemento
`@*`	Qualsiasi nodo attributo
`element-name`	Elemento con nome specifico

Esempi di Test dei Nodi

```
//p/text()              # contenuto testuale di <p>
//div/comment()         # commenti dentro <div>
//body/node()           # tutti i nodi figlio di <body>
//div/*                 # tutti i figli elemento di <div>
```

Funzioni Booleane

`true()`	Booleano vero
`false()`	Booleano falso
`boolean(expr)`	Converte in booleano
`not(expr)`	Nega il booleano
`lang(code)`	Vero se il lang del nodo corrisponde

Abbreviazioni

Forma Breve vs Estesa

`(nessuno)`	child:: (asse predefinito)
`@`	attribute::
`//`	/descendant-or-self::node()/
`.`	self::node()
`..`	parent::node()
`[n]`	[position()=n]

Esempi Abbreviati

```
# Queste coppie sono equivalenti:
child::div            →  div
attribute::href       →  @href
/descendant-or-self::node()/p  →  //p
self::node()          →  .
parent::node()        →  ..
```

Pattern Abbreviati Comuni

```
//div[@id='main']       # div con id="main"
//table//td             # tutti i <td> in qualsiasi <table>
../sibling              # fratello tramite genitore
.//span                 # discendenti span del contesto
```

Pattern Comuni

Web Scraping / Testing

```
//input[@name='username']          # input form per nome
//button[text()='Submit']          # button per testo
//div[contains(@class, 'error')]   # elemento per classe parziale
//a[contains(@href, 'login')]      # link per href parziale
```

Selezione Condizionale

```
//div[@class='item'][.//span[@class='price']]
//tr[td[1]='Active']              # riga dove 1a cella = Active
//*[contains(text(), 'Warning')]   # qualsiasi elemento con testo
```

XPath in Python (lxml)

```
from lxml import html
tree = html.fromstring(page_content)
links = tree.xpath('//a/@href')
titles = tree.xpath('//h2/text()')
```

XPath in Selenium

```
driver.find_element(By.XPATH, "//input[@id='search']")
driver.find_elements(By.XPATH, "//li[@class='result']")
```