RÉFÉRENCE RAPIDE XPATH

Axes, prédicats, fonctions, opérateurs, sélection de nœuds

Syntaxe

Expressions de chemin

`/`	Nœud racine (début de chemin absolu)
`/bookstore/book`	Sélection d'enfant direct
`//book`	Sélectionner tous les nœuds book n'importe où
`.`	Nœud de contexte actuel
`..`	Parent du nœud actuel
`@lang`	Attribut nommé lang
`node()`	Tout nœud de tout type
`*`	Tout nœud élément
`@*`	Tout attribut

Exemples de base

```
/html/body/div          # chemin absolu vers <div>
//input[@type='text']   # tous les inputs de type texte
//div[@class='main']/*  # enfants de div.main
//a/@href               # tous les attributs href
```

Combinaison de chemins

```
//book/title | //book/price   # union de deux chemins
//h1 | //h2 | //h3           # plusieurs types d'éléments
```

Axes

Directions d'axe

`child::`	Enfants directs (axe par défaut)
`parent::`	Parent direct
`ancestor::`	Tous les ancêtres jusqu'à la racine
`ancestor-or-self::`	Ancêtres + nœud actuel
`descendant::`	Tous les descendants
`descendant-or-self::`	Descendants + nœud actuel
`following::`	Tous les nœuds après l'actuel dans le document
`following-sibling::`	Frères après l'actuel
`preceding::`	Tous les nœuds avant l'actuel dans le document
`preceding-sibling::`	Frères avant l'actuel
`self::`	Nœud actuel uniquement
`attribute::`	Attributs du nœud actuel
`namespace::`	Nœuds d'espace de noms

Exemples d'axe

```
//div/child::p            # enfants <p> de <div>
//td/parent::tr           # parent <tr> de <td>
//h2/following-sibling::p  # <p> après <h2>
//li/ancestor::ul         # <ul> contenant <li>
```

Prédicats

Filtrer avec des prédicats

```
//book[1]                 # premier élément book
//book[last()]            # dernier élément book
//book[position() < 3]    # deux premiers books
//book[@lang='en']        # books avec lang="en"
//book[price > 30]        # books avec prix > 30
```

Modèles de prédicat

`[n]`	Élément à la position n (base 1)
`[last()]`	Dernier élément
`[last()-1]`	Avant-dernier
`[@attr]`	Possède l'attribut
`[@attr='val']`	L'attribut est égal à la valeur
`[element]`	Possède un élément enfant
`[element='text']`	L'élément enfant contient le texte
`[not(@attr)]`	Ne possède pas l'attribut

Prédicats enchaînés

```
//div[@class='list']//a[1]        # premier <a> dans div.list
//input[@type='text'][@name='q']  # condition ET
//book[price>30][@lang='en']      # conditions multiples
```

Fonctions

Fonctions de chaîne

`contains(s, sub)`	Vrai si s contient sub
`starts-with(s, pre)`	Vrai si s commence par pre
`string-length(s)`	Longueur de la chaîne
`normalize-space(s)`	Rogner et réduire les espaces
`concat(a, b, ...)`	Concaténer des chaînes
`substring(s, pos, len)`	Extraire une sous-chaîne (base 1)
`translate(s, from, to)`	Remplacement caractère par caractère

Fonctions numériques

`sum(node-set)`	Somme des valeurs numériques
`count(node-set)`	Nombre de nœuds
`floor(n)`	Arrondir vers le bas
`ceiling(n)`	Arrondir vers le haut
`round(n)`	Arrondir à l'entier le plus proche
`number(val)`	Convertir en nombre

Exemples de fonctions

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

Opérateurs

Opérateurs de comparaison

`=`	Égal
`!=`	Différent
`<`	Inférieur à
`<=`	Inférieur ou égal
`>`	Supérieur à
`>=`	Supérieur ou égal

Logiques et arithmétiques

`and`	ET logique
`or`	OU logique
`not()`	NON logique (fonction)
`+`	Addition
`-`	Soustraction
`*`	Multiplication
`div`	Division
`mod`	Modulo
`|`	Union d'ensembles de nœuds

Exemples d'opérateurs

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

Tests de nœud

Types de nœud

`node()`	Tout nœud (élément, texte, commentaire, PI)
`text()`	Nœud texte uniquement
`comment()`	Nœud commentaire uniquement
`processing-instruction()`	Nœud d'instruction de traitement
`*`	Tout nœud élément
`@*`	Tout nœud attribut
`element-name`	Élément avec un nom spécifique

Exemples de test de nœud

```
//p/text()                # contenu textuel de <p>
//div/comment()           # commentaires dans <div>
//body/node()             # tous les nœuds enfants de <body>
//div/*                   # tous les enfants éléments de <div>
```

Fonctions booléennes

`true()`	Booléen vrai
`false()`	Booléen faux
`boolean(expr)`	Convertir en booléen
`not(expr)`	Inverser un booléen
`lang(code)`	Vrai si la langue du nœud correspond

Abréviations

Forme courte vs longue

`(aucune)`	child:: (axe par défaut)
`@`	attribute::
`//`	/descendant-or-self::node()/
`.`	self::node()
`..`	parent::node()
`[n]`	[position()=n]

Exemples abrégés

```
# Ces paires sont équivalentes :
child::div              →  div
attribute::href         →  @href
/descendant-or-self::node()/p  →  //p
self::node()            →  .
parent::node()          →  ..
```

Modèles abrégés courants

```
//div[@id='main']       # div avec id="main"
//table//td             # tous les <td> dans tout <table>
../sibling              # frère via le parent
.//span                 # descendants span du contexte
```

Modèles courants

Web scraping / tests

```
//input[@name='username']          # input de formulaire par nom
//button[text()='Submit']          # bouton par texte
//div[contains(@class, 'error')]   # élément par classe partielle
//a[contains(@href, 'login')]      # lien par href partiel
```

Sélection conditionnelle

```
//div[@class='item'][.//span[@class='price']]
//tr[td[1]='Active']              # ligne où 1re cellule = Active
//*[contains(text(), 'Warning')]   # tout élément avec le texte
```

XPath en Python (lxml)

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

XPath dans Selenium

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