XPath Guide de Référence & Aide-mémoire

1. Chemins Absolus & Relatifs

Un chemin absolu part de la racine. Très précis, mais fragile en cas de modification du DOM.

```
/html/body/div[1]/span/div/div/section/p[2]
```

Un chemin relatif (// ou .) commence n'importe où. Plus robuste.

```
//div[@class='menu']/a[@id='accueil']
```

2. Séparateurs et Chemin d'Accès

/ (Slash simple) : Sépare un nœud de son enfant direct. Au début, désigne la racine.

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

// (Double Slash) : Sépare un nœud de n'importe lequel de ses descendants (n'importe où).

```
//div[@class='card']//a
```

3. Structure Fondamentale

Une étape de localisation est composée de 3 éléments : `Axe::Test\_de\_Nœud[Prédicat]`

Axe (Direction) : Ex: child:: (par défaut), parent::, attribute::.

Test de Nœud : Ex: div, span, input, * (tout élément).

Prédicat (Filtre) : Ex: [@class='menu'], [@id='valeur'], [1].

Exemple (Les 3 éléments) :

```
parent::div[@class='menu']
```

4. Utilisation d'index

Position et Stabilité

Index de position ://li[3] ou //li[last()].

Fonction position() ://li[position() > 5].

⚠️ **Danger de l'Index : L'utilisation d'index numériques (comme [1]) est extrêmement fragile. Si un élément est ajouté ou retiré, le chemin casse. Privilégiez les attributs et les fonctions texte pour la stabilité.**

5. Opérateurs & Prédicats Avancés

Opérateurs Logiques

Utilisés dans les prédicats pour combiner des conditions :

AND : Les deux conditions doivent être vraies.

```
//input[@type='text' and @id='prenom']
```

OR : Au moins une condition doit être vraie.

```
//button[@name='Ajouter' or @name='Supprimer']
```

Prédicats Complexes

Prédicat sur un descendant :

```
//div[.//span[@class='active']]
```

Filtre basé sur l'enfant direct :

```
//tr[./td[@class='type']]
```

Filtre basé sur un descendant avec opérateur logique :

```
//div[.//a[@class='link' and text()='Détails']]
```

6. Axes de Navigation (Relations)

Les axes définissent la direction de la navigation par rapport au nœud courant. Ils sont fondamentaux pour localiser des éléments parents, frères, ou descendants indirects.

Axe	Abr.	Rôle
child::	(omis)	Nœuds enfants directs.
parent::	..	Nœud parent.
self::	.	Le nœud courant.
attribute::	@	Nœuds attributs.
descendant::	//	Tous les descendants.
ancestor::		Tous les ancêtres du nœud courant.
following-sibling::		Frères/sœurs qui apparaissent APRÈS le nœud courant.
preceding-sibling::		Frères/sœurs qui apparaissent AVANT le nœud courant.
following::		Tous les nœuds suivants.
preceding::		Tous les nœuds précédents.

7. Sélecteurs Spéciaux & opérateurs

Utilisation de caractères génériques (wildcards) pour trouver des éléments ou attributs avec souplesse.

Sélecteur	Rôle	Exemple
*	Sélectionne n'importe quel élément.	`//div/*/a`
@*	Sélectionne un élément avec n'importe quel attribut.	`//button[@*]`
//text()	Sélectionne uniquement le contenu textuel du nœud.	`//p[1]/text()`

Opérateurs de Comparaison et Types de Nœuds

Opérateur	Signification	Type de Nœud	Explication
=	Égalité	element	Sélectionne une balise HTML/XML (Ex: div).
!=	Non Égalité	attribute	Sélectionne un attribut (Ex: @id).
<, >	Inférieur, Supérieur	text()	Sélectionne le contenu textuel.
<=, >=	Inf. ou Ég., Sup. ou Ég.	comment()	Sélectionne un commentaire.

8. Fonctions Courantes (Texte)

Fonction	Rôle	Exemple
contains(s, sub)	Vrai si s contient sub.	`//p[contains(text(), 'partiel')]`
starts-with(s, sub)	Vrai si s commence par sub.	`//input[starts-with(@id, 'utilisateur')]`
normalize-space(s)	Nettoie les espaces multiples et début/fin.	`//div[normalize-space(text())='Texte']`
concat(s1, s2, ...)	Concatène des chaînes.	`concat('Nom: ', @nom)`
substring(s, start, len)	Extrait une sous-chaîne (position 1-basée).	`substring('ABCD', 2, 2) → 'BC'`
substring-before(s, delim)	Extrait avant le délimiteur.	`substring-before('nom@dom.com', '@')`
substring-after(s, delim)	Extrait après le délimiteur.	`substring-after('nom@dom.com', '@')`