**eBook Gratuit**

# APPRENEZ xpath

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#xpath

# Table des matières

# À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: xpath

It is an unofficial and free xpath ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official xpath.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapitre 1: Démarrer avec xpath

## Remarques

XPath est un langage permettant d'adresser des parties d'un document XML.

Il est utilisé dans XSLT et est un sous-ensemble de XQuery. Les bibliothèques sont également disponibles pour la plupart des autres langages de programmation.

XPath est une norme internationale avec des spécifications publiées par le W3C:

- XPath 1.0: XML Path Language (XPath), version 1.0
- XPath 2.0: XML Path Language (XPath) 2.0 (deuxième édition)
- XPath 3.0: XML Path Language (XPath) 3.0

## Versions

| Version | Date de sortie |
|---|---|
| 1.0 | 1999-12-16 |
| 2.0 | 2007-01-23 |
| 3.0 | 2014-04-08 |
| 3.1 (Recommandation du candidat du W3C) | 2015-12-17 |

## Examples

### Exemple de XML (sans espaces de noms)

Voici quelques exemples de XML avec lesquels XPath peut être écrit:

```
<r>
  <e a="1"/>
  <f a="2" b="1">Text 1</f>
  <f/>
  <g>
    <i c="2">Text 2</i>
    Text 3
    <j>Text 4</j>
  </g>
</r>
```

### Sélectionnez le texte

Pour l'exemple XML (sans espaces de noms):

Ce XPath,

```
/r/f/text()
```

sélectionnera le noeud de texte avec cette valeur de chaîne:

```
"Text 1"
```

Et ce XPath,

```
string(/r/f)
```

renverra la valeur de chaîne de `f` , qui est aussi:

```
"Text 1"
```

## Sélectionnez un élément

Pour l'exemple XML (sans espaces de noms):

Ce XPath,

```
/r/e
```

sélectionnera cet élément:

```
<e a="1"/>
```

## Opérations HTML communes

Si l'entrée HTML DOM est

```
<html>
    <body>
        <a>link</a>
        <div class='container' id='divone'>
            <p class='common' id='enclosedone'>Element One</p>
            <p class='common' id='enclosedtwo'>Element Two</p>
        </div>
    </body>
</html>
```

Trouver un élément avec un identifiant spécifique dans la page entière

```
//*[@id='divone'] # Returns <div class='container' id='divone'>
```

Trouver un élément avec un identifiant spécifique dans un chemin particulier

```
/html/body/div/p[@id='enclosedone'] # Returns <p class='common' id='enclosedone'>Element
One</p>
```

Sélectionnez un élément avec un identifiant et une classe particuliers

```
//p[@id='enclosedone' and @class='common'] # Returns <p class='common'
id='enclosedone'>Element One</p>
```

Sélectionnez le texte d'un élément particulier

```
//*[@id='enclosedone']/text() # Returns Element One
```

## Test de Xpaths dans la console du navigateur

Un moyen rapide de tester votre xpath se trouve dans la console de votre outil de développement de navigateur.

Le format est

```
   $x('//insert xpath here')
```

$ - spécifie qu'il s'agit d'un sélecteur.

x - spécifie qu'il utilise xpaths

Exemple:

```
$x("//button[text() ='Submit']")
```

Lorsque cette commande est entrée, elle renvoie toutes les occurrences des éléments qui sont des boutons avec un texte égal à Submit.

Lire Démarrer avec xpath en ligne: https://riptutorial.com/fr/xpath/topic/883/demarrer-avec-xpath

# Chapitre 2: Chemins de localisation et axes

## Remarques

Un *chemin d'emplacement* XPath est une série d' *étapes d'emplacement* séparées par un caractère `/` :

```
step1/step2/step3
```

Une *étape d'emplacement* contient un *axe* , un *test de noeud* et une liste facultative de *prédicats* . L' *axe* et le *test de noeud* sont séparés par deux deux-points `::` . Les *prédicats* sont entre crochets:

```
axis::nodeTest[predicate1][predicate2]
```

L'évaluation d'un *chemin d'emplacement* commence par un ensemble de nœuds contenant le *nœud de contexte* donné par le contexte de l'expression ou par le *nœud racine* , si le chemin d'emplacement commence par un `/` . A chaque étape, chaque nœud *N* de l'ensemble de nœuds d'origine est remplacé par l'ensemble des nœuds

- peut être atteint à partir de *N* suivant l' *axe* donné,
- correspond au *test de noeud* et
- correspond à tous les *prédicats* .

Le résultat d'une expression de *chemin d'emplacement* correspond au dernier ensemble de nœuds obtenu après le traitement de toutes *les étapes d'emplacement* .

## Examples

### Traversant des éléments enfants

Passage du noeud racine à un élément descendant à l'aide de l'axe `child` :

```
/child::html/child::body/child::div/child::span
```

L'axe `child` étant l'axe par défaut, il peut être abrégé en:

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

### Traversant tous les descendants

Les axes `descendant` et `descendant-or-self` peuvent être utilisés pour trouver tous les éléments descendants d'un nœud à n'importe quelle profondeur. En revanche, l'axe `child` ne traverse que les enfants immédiats.

```
/child::html/descendant::span
/child::html/descendant-or-self::*
```

La double barre oblique `//` est un raccourci pour `/descendant-or-self::node()/` . Les expressions suivantes sont donc équivalentes:

```
table//td
child::table/descendant-or-self::node()/child::td
child::table/descendant::td
table/descendant::td
```

## Traversant les ancêtres

L'axe `parent` ne contient que le parent d'un noeud. L'expression suivante sélectionne l'élément `html` en faisant un détour sur l'élément `body` :

```
/child::html/child::body/parent::html
```

`..` est un raccourci pour `parent::node()`

Les axes `ancestor` et `ancestor-or-self` traversent tous les ancêtres d'un nœud. L'expression suivante renvoie tous les éléments `div` qui sont les ancêtres du noeud de contexte:

```
ancestor::div
```

## L'axe "self"

L'axe `self` ne contient que le noeud de contexte lui-même. L'expression `.` est un raccourci pour `self::node()` et correspond toujours au noeud de contexte. Le `.` Le raccourci est utile pour énumérer les descendants du nœud de contexte. Les expressions suivantes sont équivalentes:

```
.//span
self::node()/descendant-or-self::node()/child::span
descendant::span
```

L'axe `self` peut être utile dans les prédicats XPath 1.0. Par exemple, pour sélectionner tous les enfants `h1` , `h2` et `h3` du noeud de contexte:

```
*[self::h1 or self::h2 or self::h3]
```

## Passage des nœuds suivants et précédents

Les axes `following-sibling` et `preceding-sibling` suivants contiennent les frères avant ou après le nœud de contexte, et les axes `following` et `preceding` contiennent tous les nœuds du document avant ou après le nœud de contexte, mais:

- Aucun de ces axes ne contient de nœuds d'attribut ou d'espace de noms.
- L'axe `following` ne contient pas de descendants.

- L'axe `preceding` ne contient aucun ancêtre.

Exemples:

```
following::span[1]
following-sibling::*[last()]
```

### Traversée d'attributs et de nœuds d'espace de noms

L' `attribute` et `namespace` axes d' `namespace` contiennent tous les nœuds d'attribut et d'espace de nom d'un élément. Le raccourci `@` représente l' `attribute::` , donc les suivants sont équivalents:

```
child::div/attribute::class
div/@class
```

Lire Chemins de localisation et axes en ligne: https://riptutorial.com/fr/xpath/topic/6171/chemins-de-localisation-et-axes

# Chapitre 3: Espaces de noms

## Remarques

XPath 1.0 n'a pas le concept d'un espace de noms par défaut.

De plus, les préfixes d'espace de nommage définis dans le document XML d'origine n'affectent pas XPath - les préfixes d'espace de noms doivent être explicitement enregistrés avec le fournisseur XPath, sinon les préfixes ne peuvent pas être utilisés du tout dans l'expression XPath.

## Examples

### Fonctions sensibles aux espaces de noms

```
<root xmlns="http://test/">
    <element xmlns:example="http://foobar/">
        <example:hello_world attribute="another example" />
    </element>
</root>
```

L'expression `/root` ne renverra rien car il n'y a pas d'élément non nommé appelé `root` au niveau racine du document. Toutefois, ne retournera le `<root xmlns="http://test/">` élément.

```
/*[namespace-uri() = 'http://test/' and local-name() = 'root']
```

Lire Espaces de noms en ligne: https://riptutorial.com/fr/xpath/topic/2324/espaces-de-noms

# Chapitre 4: Recherche d'éléments contenant des attributs spécifiques

## Examples

### Trouver tous les éléments avec un certain attribut

Imaginez le XML suivant:

```
<root>
    <element foobar="hello_world" />
    <element example="this is one!" />
</root>
```

```
/root/element[@foobar]
```

et renverra l' `<element foobar="hello_world" />` .

### Trouver tous les éléments avec une certaine valeur d'attribut

Imaginez le XML suivant:

```
<root>
    <element foobar="hello_world" />
    <element example="this is one!" />
</root>
```

L'expression XPath suivante:

```
/root/element[@foobar = 'hello_world']
```

renverra l' `<element foobar="hello_world" />` .

des guillemets doubles peuvent également être utilisés:

```
/root/element[@foobar="hello_world"]
```

Lire Recherche d'éléments contenant des attributs spécifiques en ligne: https://riptutorial.com/fr/xpath/topic/6488/recherche-d-elements-contenant-des-attributs-specifiques

# Chapitre 5: Recherche d'éléments contenant un texte spécifique

## Examples

### Trouver tous les éléments avec un certain texte

Imaginez le XML suivant:

```
<root>
    <element>hello</element>
    <another>
        hello
    </another>
    <example>Hello, <nested> I am an example </nested>.</example>
</root>
```

L'expression XPath suivante:

```
//*[text() = 'hello']
```

renverra l' `<element>hello</element>` , mais pas l'élément `<another>` . C'est parce que l'élément `<another>` contient des espaces blancs entourant le texte `hello` .

Pour récupérer à la fois `<element>` et `<another>` , on pourrait utiliser:

```
//*[normalize-space(text()) = 'hello']
```

ou

```
//*[normalize-space() = 'hello']
```

qui va couper les espaces blancs avant de faire la comparaison. Ici, nous pouvons voir que le spécificateur de noeud `text()` est facultatif lors de l'utilisation `normalize-space` .

Pour trouver un élément *contenant* un texte spécifique, vous pouvez utiliser la fonction `contains` . L'expression suivante retournera l'élément `<example>` :

```
//example[contains(text(), 'Hello')]
```

---

Si vous souhaitez rechercher du texte couvrant plusieurs enfants / noeuds de texte, vous pouvez utiliser `.` au lieu de `text()` . `.` fait référence à l'intégralité du contenu textuel de l'élément et à ses enfants.

```
//example[. = 'Hello,  I am an example .']
```

Pour voir les différents nœuds de texte, vous pouvez utiliser:

```
//example//text()
```

qui reviendra:

- "Bonjour, "
- "Je suis un exemple"
- "."

Et pour voir plus clairement tout le contenu textuel d'un élément, on peut utiliser la fonction `string` :

```
string(//example[1])
```

ou juste

```
string(//example)
```

> Bonjour, je suis un exemple.

Ce dernier fonctionne car, si un ensemble de nœuds est passé dans des fonctions telles que des `string` , XPath 1.0 se contente d'examiner le premier nœud (dans l'ordre des documents) dans cet ensemble de nœuds et ignore le reste.

alors:

```
string(/root/*)
```

retournerais:

> Bonjour

Lire Recherche d'éléments contenant un texte spécifique en ligne:
https://riptutorial.com/fr/xpath/topic/1903/recherche-d-elements-contenant-un-texte-specifique

# Chapitre 6: Rechercher des nœuds ayant un attribut spécifique

## Syntaxe

1. À l'intérieur d'un noeud spécifique
    - / chemin d'accès à / element [nom_attribut_attribut]
2. Partout dans le document
    - //*[@Nom d'attribut]
3. A l'intérieur d'un noeud spécifique avec une certaine valeur
    - / chemin vers / element [@ nom_attribut = 'valeur de recherche']
    - / chemin vers / element [@ nom_attribut = "valeur de recherche"]
4. Partout dans le document avec une certaine valeur
    - // * [@ attribute_name = 'recherche chaîne']
    - // * [@ nom_attribut = "chaîne de recherche"]

## Paramètres

| Sélecteur | fonction |
|---|---|
| @Nom d'attribut | Il sélectionne la valeur d'attribut d'un noeud, s'il est présent |

## Remarques

En utilisant [@attribute_name], nous pouvons sélectionner des noeuds qui ont l'attribut indépendamment de la valeur.

Nous pouvons utiliser n'importe quelle fonction ou combinaison de fonctions telles que les débuts avec et les minuscules, par exemple, avec ce sélecteur pour répondre à nos besoins.

## Examples

### Rechercher des nœuds avec un attribut spécifique

XML

```
<Galaxy>
    <name>Milky Way</name>
    <CelestialObject name="Earth" type="planet"/>
    <CelestialObject name="Sun" type="star"/>
</Galaxy>
```

XPATH

```
/Galaxy/*[@name]
```

ou

```
//*[@name]
```

SORTIE

```
<CelestialObject name="Earth" type="planet" />
<CelestialObject name="Sun" type="star" />
```

## Rechercher des nœuds avec une valeur d'attribut spécifique

XML

```
<Galaxy>
    <name>Milky Way</name>
    <CelestialObject name="Earth" type="planet"/>
    <CelestialObject name="Sun" type="star"/>
</Galaxy>
```

XPATH

```
/Galaxy/*[@name='Sun']
```

ou

```
//*[@name='Sun']
```

SORTIE

```
<CelestialObject name="Sun" type="star" />
```

## Recherche de noeuds par sous-chaîne correspondant à la valeur d'un attribut

XML

```
<Galaxy>
    <name>Milky Way</name>
    <CelestialObject name="Earth" type="planet"/>
    <CelestialObject name="Sun" type="star"/>
</Galaxy>
```

XPATH

```
/Galaxy/*[contains(@name,'Ear')]
```

ou

```
//*[contains(@name,'Ear')]
```

Les guillemets doubles peuvent également être utilisés à la place des guillemets simples:

```
/Galaxy/*[contains(@name, "Ear")]
```

SORTIE

```
<CelestialObject name="Earth" type="planet" />
```

## Recherche de nœuds par sous-chaîne correspondant à la valeur d'un attribut (insensible à la casse)

XML

```
<Galaxy>
    <name>Milky Way</name>
    <CelestialObject name="Earth" type="planet"/>
    <CelestialObject name="Sun" type="star"/>
</Galaxy>
```

XPATH

```
/Galaxy/*[contains(lower-case(@name),'ear')]
```

ou

```
//*[contains(lower-case(@name),'ear')]
```

ou, avec la chaîne entre guillemets:

```
//*[contains(lower-case(@name), "ear")]
```

SORTIE

```
<CelestialObject name="Earth" type="planet" />
```

## Rechercher des nœuds par sous-chaîne correspondant au début de la valeur d'un attribut

XML

```
<Galaxy>
    <name>Milky Way</name>
    <CelestialObject name="Earth" type="planet"/>
    <CelestialObject name="Sun" type="star"/>
</Galaxy>
```

XPATH

```
/Galaxy/*[starts-with(lower-case(@name),'ear')]
```

ou

```
//*[starts-with(lower-case(@name),'ear')]
```

SORTIE

```
<CelestialObject name="Earth" type="planet" />
```

## Rechercher des nœuds par sous-chaîne correspondant à la fin de la valeur d'un attribut

XML

```
<Galaxy>
    <name>Milky Way</name>
    <CelestialObject name="Earth" type="planet"/>
    <CelestialObject name="Sun" type="star"/>
</Galaxy>
```

XPATH

```
/Galaxy/*[ends-with(lower-case(@type),'tar')]
```

ou

```
//*[ends-with(lower-case(@type),'tar')]
```

SORTIE

```
<CelestialObject name="Sun" type="star" />
```

Lire Rechercher des nœuds ayant un attribut spécifique en ligne:
https://riptutorial.com/fr/xpath/topic/3096/rechercher-des-nouds-ayant-un-attribut-specifique

# Chapitre 7: Récupère le nombre de nœuds

## Syntaxe

- compter (ensemble de nœuds)

## Paramètres

| fonction | résultats |
|---|---|
| compter | nombre total de noeuds dans l'ensemble de noeuds |

## Remarques

Nous pouvons l'utiliser en combinaison avec d'autres fonctions et axes pour répondre à nos besoins.

## Examples

### Combien d'enfants a Goku?

XML

```
<Goku>
    <child name="Gohan"/>
    <child name="Goten"/>
</Goku>
```

XPATH

```
count(/Goku/child)
```

SORTIE

```
2.0
```

### Combien de plantes y a-t-il dans la maison?

XML

```
<House>
    <LivingRoom>
        <plant name="rose"/>
    </LivingRoom>
```

```
    <TerraceGarden>
        <plant name="passion fruit"/>
        <plant name="lily"/>
        <plant name="golden duranta"/>
    </TerraceGarden>
</House>
```

### XPATH

```
count(/House//plant)
```

### SORTIE

```
4.0
```

Lire Récupère le nombre de nœuds en ligne: https://riptutorial.com/fr/xpath/topic/4463/recupere-le-nombre-de-nouds

# Chapitre 8: Récupère les nœuds relatifs au nœud actuel

## Syntaxe

1. Tous les ancêtres d'un nœud
    - / chemin du noeud / ancêtre :: node ()
2. Un ancêtre spécifique d'un noeud
    - / chemin vers le noeud / ancêtre :: nom_principal
3. Parent d'un noeud
    - / chemin du noeud / parent :: node ()
4. Frères et soeurs suivants d'un noeud
    - / chemin du noeud / soeur suivant :: node ()
5. Un frère spécifique suivant un nœud
    - / chemin du noeud / soeur-suivant :: nom_sager
6. Frères et soeurs précédents d'un noeud
    - / chemin du noeud / precedent-sibling :: node ()
7. Un frère spécifique précédant un noeud
    - / chemin vers le noeud / precedent-sibling :: nom-frère
8. Tous les noeuds enfant immédiats d'un noeud
    - / chemin du noeud / enfant :: node ()
9. Un noeud enfant immédiat spécifique d'un noeud
    - / chemin du noeud / enfant :: nom_chid
10. Tous les descendants d'un nœud
    - / chemin du noeud / descendant :: node ()
11. Tous les descendants spécifiques d'un noeud
    - / chemin vers le nœud / descendant :: nom_ descendant

## Paramètres

| Axe | sélectionne |
|---|---|
| ancêtre | tous les nœuds ancêtres |
| parent | nœud parent |
| frère suivant | frères et soeurs suivant le nœud |
| frère précédent | frères et soeurs précédant le nœud |
| enfant | enfants immédiats |
| descendant | tous les descendants indépendamment du niveau de nidification |

# Remarques

Ces axes peuvent être utilisés en combinaison avec d'autres fonctions pour répondre à nos besoins.

# Examples

## Trouver mes ancêtres

XML

```
<GrandFather name="Bardock" gender="male" spouse="Gine">
    <Dad name="Goku" gender="male" spouse="Chi Chi">
        <Me name="Gohan" gender="male"/>
        <brother name="Goten" gender="male"/>
    </Dad>
</GrandFather>
```

XPATH

```
//Me/ancestor::node()
```

SORTIE

```
<GrandFather name="Bardock" gender="male" spouse="Gine">
  <Dad name="Goku" gender="male" spouse="Chi Chi">
    <Me name="Gohan" gender="male" />
    <brother name="Goten" gender="male" />
  </Dad>
</GrandFather>
<Dad name="Goku" gender="male" spouse="Chi Chi">
  <Me name="Gohan" gender="male" />
  <brother name="Goten" gender="male" />
</Dad>
```

## Trouver mon parent

XML

```
<GrandFather name="Bardock" gender="male" spouse="Gine">
    <Dad name="Goku" gender="male" spouse="Chi Chi">
        <Me name="Gohan" gender="male"/>
        <brother name="Goten" gender="male"/>
    </Dad>
</GrandFather>
```

XPATH

```
//Me/ancestor::Dad
```

ou

```
//Me/parent::node()
```

SORTIE

```
<Dad name="Goku" gender="male" spouse="Chi Chi">
  <Me name="Gohan" gender="male" />
  <brother name="Goten" gender="male" />
</Dad>
```

## Trouver mon grand père

XML

```
<GrandFather name="Bardock" gender="male" spouse="Gine">
  <Dad name="Goku" gender="male" spouse="Chi Chi">
    <Me name="Gohan" gender="male" />
    <brother name="Goten" gender="male" />
  </Dad>
</GrandFather>
```

XPATH

```
//Me/ancestor::GrandFather
```

ou

```
//Me/parent::node()/parent::node()
```

SORTIE

```
<GrandFather name="Bardock" gender="male" spouse="Gine">
  <Dad name="Goku" gender="male" spouse="Chi Chi">
    <Me name="Gohan" gender="male" />
    <brother name="Goten" gender="male" />
  </Dad>
</GrandFather>
```

## Trouver mon frere

XML

```
<GrandFather name="Bardock" gender="male" spouse="Gine">
  <Dad name="Goku" gender="male" spouse="Chi Chi">
    <brother name="Goten" gender="male" />
    <Me name="Gohan" gender="male" />
    <brother name="Goten" gender="male" />
  </Dad>
</GrandFather>
```

XPATH

```
//Me/following-sibling::brother
```

SORTIE

```
<brother name="Goten" gender="male" />
```

## Obtenez tous les avatars avant Parashurama

XML

```
<Dashavatar>
    <Avatar name="Matsya"/>
    <Avatar name="Kurma"/>
    <Avatar name="Varaha"/>
    <Avatar name="Narasimha"/>
    <Avatar name="Vamana"/>
    <Avatar name="Balabhadra"/>
    <Avatar name="Parashurama"/>
    <Avatar name="Rama"/>
    <Avatar name="Krishna"/>
    <Avatar name="Kalki"/>
</Dashavatar>
```

XPATH

```
//Avatar[@name='Parashurama']/preceding-sibling::node()
```

SORTIE

```
<Avatar name="Matsya"/>
<Avatar name="Kurma"/>
<Avatar name="Varaha"/>
<Avatar name="Narasimha"/>
<Avatar name="Vamana"/>
<Avatar name="Balabhadra"/>
```

## Obtenez tous les avatars après Parashurama

XML

```
<Dashavatar>
    <Avatar name="Matsya"/>
    <Avatar name="Kurma"/>
    <Avatar name="Varaha"/>
    <Avatar name="Narasimha"/>
    <Avatar name="Vamana"/>
    <Avatar name="Balabhadra"/>
    <Avatar name="Parashurama"/>
    <Avatar name="Rama"/>
    <Avatar name="Krishna"/>
```

```
    <Avatar name="Kalki"/>
</Dashavatar>
```

XPATH

```
//Avatar[@name='Parashurama']/following-sibling::node()
```

SORTIE

```
<Avatar name="Rama" />
<Avatar name="Krishna" />
<Avatar name="Kalki" />
```

## Obtenez tous les avatars sauf celui actuel (Parusharama)

XML

```
<Dashavatar>
    <Avatar name="Matsya"/>
    <Avatar name="Kurma"/>
    <Avatar name="Varaha"/>
    <Avatar name="Narasimha"/>
    <Avatar name="Vamana"/>
    <Avatar name="Balabhadra"/>
    <Avatar name="Parashurama"/>
    <Avatar name="Rama"/>
    <Avatar name="Krishna"/>
    <Avatar name="Kalki"/>
</Dashavatar>
```

XPATH

```
//Avatar[@name='Parashurama']/following-sibling::Avatar |
//Avatar[@name='Parashurama']/preceding-sibling::Avatar
```

SORTIE

```
<Avatar name="Matsya" />
<Avatar name="Kurma" />
<Avatar name="Varaha" />
<Avatar name="Narasimha" />
<Avatar name="Vamana" />
<Avatar name="Balabhadra" />
<Avatar name="Rama" />
<Avatar name="Krishna" />
<Avatar name="Kalki" />
```

## Obtenez tous les détails (noeuds enfants) de House

XML

```
<House>
    <Rooms>10</Rooms>
    <People>4</People>
    <TVs>4</TVs>
    <Floors>2</Floors>
</House>
```

XPATH

```
/House/child::node()
```

SORTIE

```
<Rooms>10</Rooms>
<People>4</People>
<TVs>4</TVs>
<Floors>2</Floors>
```

## Obtenez toutes les chambres (enfants immédiats nommés chambre) dans la maison

XML

```
<House>
    <numRooms>4</numRooms>
    <Room name="living"/>
    <Room name="master bedroom"/>
    <Room name="kids' bedroom"/>
    <Room name="kitchen"/>
</House>
```

XPATH

```
/House/child::Room
```

ou

```
/House/*[local-name()='Room']
```

SORTIE

```
<Room name="living" />
<Room name="master bedroom" />
<Room name="kids' bedroom" />
<Room name="kitchen" />
```

## Obtenez toutes les chambres (indépendamment de la position) dans la maison

XML

```
<House>
    <numRooms>4</numRooms>
    <Floor number="1">
        <Room name="living"/>
        <Room name="kitchen"/>
    </Floor>
    <Floor number="2">
        <Room name="master bedroom"/>
        <Room name="kids' bedroom"/>
    </Floor>
</House>
```

XPATH

```
/House/descendant::Room
```

SORTIE

```
<Room name="living" />
<Room name="kitchen" />
<Room name="master bedroom" />
<Room name="kids' bedroom" />
```

Lire Récupère les nœuds relatifs au nœud actuel en ligne:
https://riptutorial.com/fr/xpath/topic/6495/recupere-les-nouds-relatifs-au-noud-actuel

# Chapitre 9: Sélectionner des noeuds dont le nom est égal ou contenant une chaîne

## Syntaxe

1. À l'intérieur d'un noeud spécifique:

   {path-to-parent} / name () = 'chaîne de recherche']

2. N'importe où dans le document:

   // * [name () = 'chaîne de recherche']

## Paramètres

| fonction | valeur de retour |
| --- | --- |
| nom local () | le nom du noeud sans préfixe |

## Remarques

Le nom local-name () n'inclut pas le préfixe (la fonction XPATH de lookup name ())

## Examples

### Rechercher des nœuds portant le nom Light, Device ou Sensor

XML

```
<Galaxy>
    <Light>sun</Light>
    <Device>satellite</Device>
    <Sensor>human</Sensor>
    <Name>Milky Way</Name>
</Galaxy>
```

XPATH

```
/Galaxy/*[local-name()='Light' or local-name()='Device' or local-name()='Sensor']
```

ou

```
//*[local-name()='Light' or local-name()='Device' or local-name()='Sensor']
```

SORTIE

```
<Light>sun</Light>
<Device>satellite</Device>
<Sensor>human</Sensor>
```

## Rechercher des noeuds dont le nom contient Light

XML

```
<Data>
    <BioLight>
        <name>Firefly</name>
        <model>Insect</model>
    </BioLight>
    <ArtificialLight>
        <name>Fire</name>
        <model>Natural element</model>
        <source>flint</source>
    </ArtificialLight>
    <SolarLight>
        <name>Sun</name>
        <model>Star</model>
        <source>helium</source>
    </SolarLight>
</Data>
```

XPATH

```
/Data/*[contains(local-name(),"Light")]
```

ou

```
//*[contains(local-name(),"Light")]
```

SORTIE

```
<BioLight>
  <name>Firefly</name>
  <model>Insect</model>
</BioLight>
<ArtificialLight>
  <name>Fire</name>
  <model>Natural element</model>
  <source>flint</source>
</ArtificialLight>
<SolarLight>
  <name>Sun</name>
  <model>Star</model>
  <source>helium</source>
</SolarLight>
```

## Rechercher des noeuds dont le nom commence par Star

XML

```
<College>
    <FootBall>
        <Members>20</Members>
        <Coach>Archie Theron</Coach>
        <Name>Wild cats</Name>
        <StarFootballer>David Perry</StarFootballer>
    </FootBall>
    <Academics>
        <Members>100</Members>
        <Teacher>Tim Jose</Teacher>
        <Class>VII</Class>
        <StarPerformer>Lindsay Rowen</StarPerformer>
    </Academics>
</College>
```

XPATH

```
/College/*/*[starts-with(local-name(),"Star")]
```

ou

```
//*[starts-with(local-name(),"Star")]
```

SORTIE

```
<StarFootballer>David Perry</StarFootballer>
<StarPerformer>Lindsay Rowen</StarPerformer>
```

## Rechercher des noeuds dont le nom se termine par Ball

XML

```
<College>
    <FootBall>
        <Members>20</Members>
        <Coach>Archie Theron</Coach>
        <Name>Wild cats</Name>
        <StarPlayer>David Perry</StarPlayer>
    </FootBall>
    <VolleyBall>
        <Members>24</Members>
        <Coach>Tim Jose</Coach>
        <Name>Avengers</Name>
        <StarPlayer>Lindsay Rowen</StarPlayer>
    </VolleyBall>
    <FoosBall>
        <Members>22</Members>
        <Coach>Rahul Mehra</Coach>
        <Name>Playerz</Name>
        <StarPlayer>Amanda Ren</StarPlayer>
    </FoosBall>
</College>
```

XPATH

```
/College/*[ends-with(local-name(),"Ball")]
```

ou

```
//*[ends-with(local-name(),"Ball")]
```

SORTIE

```
<FootBall>
  <Members>20</Members>
  <Coach>Archie Theron</Coach>
  <Name>Wild cats</Name>
  <StarPlayer>David Perry</StarPlayer>
</FootBall>
<VolleyBall>
  <Members>24</Members>
  <Coach>Tim Jose</Coach>
  <Name>Avengers</Name>
  <StarPlayer>Lindsay Rowen</StarPlayer>
</VolleyBall>
<FoosBall>
  <Members>22</Members>
  <Coach>Rahul Mehra</Coach>
  <Name>Playerz</Name>
  <StarPlayer>Amanda Ren</StarPlayer>
</FoosBall>
```

## Rechercher des nœuds avec un nom léger (insensible à la casse)

XML

```
<Galaxy>
    <Light>sun</Light>
    <Device>satellite</Device>
    <Sensor>human</Sensor>
    <Name>Milky Way</Name>
</Galaxy>
```

XPATH

```
/Galaxy/*[lower-case(local-name())="light"]
```

ou

```
//*[lower-case(local-name())="light"]
```

SORTIE

```
<Light>sun</Light>
```

Lire Sélectionner des noeuds dont le nom est égal ou contenant une chaîne en ligne: https://riptutorial.com/fr/xpath/topic/3095/selectionner-des-noeuds-dont-le-nom-est-egal-ou-contenant-une-chaine

# Chapitre 10: Sélectionner des nœuds en fonction de leurs enfants

## Examples

### Sélectionner des nœuds en fonction du nombre d'enfants

*Exemple XML*

```
 <Students>
    <Student>
        <Name>
            <First>Ashley</First>
            <Last>Smith</Last>
        </Name>
        <Grades>
            <Exam1>A</Exam1>
            <Exam2>B</Exam2>
            <Final>A</Final>
        </Grades>
    </Student>
    <Student>
        <Name>
            <First>Bill</First>
            <Last>Edwards</Last>
        </Name>
        <Grades>
            <Exam1>A</Exam1>
        </Grades>
    </Student>
</Students>
```

*XPath*

Sélectionnez tous les élèves ayant au moins 2 notes enregistrées

```
//Student[count(./Grades/*) > 1]
```

*Sortie*

```
<Student>
    <Name>
        <First>Ashley</First>
        <Last>Smith</Last>
    </Name>
    <Grades>
        <Exam1>A</Exam1>
        <Exam2>B</Exam2>
        <Final>A</Final>
    </Grades>
</Student>
```

## Sélectionner des noeuds basés sur un noeud enfant spécifique

*Exemple XML*

```
 <Students>
    <Student>
        <Name>
            <First>Ashley</First>
            <Last>Smith</Last>
        </Name>
        <Grades>
            <Exam1>A</Exam1>
            <Exam2>B</Exam2>
            <Final>A</Final>
        </Grades>
    </Student>
    <Student>
        <Name>
            <First>Bill</First>
            <Last>Edwards</Last>
        </Name>
        <Grades>
            <Exam1>A</Exam1>
        </Grades>
    </Student>
</Students>
```

*XPath*

Sélectionnez tous les étudiants qui ont un score pour Exam2 enregistré

```
//Student[./Grades/Exam2]
```

ou

```
//Student[.//Exam2]
```

*Sortie*

```
<Student>
    <Name>
        <First>Ashley</First>
        <Last>Smith</Last>
    </Name>
    <Grades>
        <Exam1>A</Exam1>
        <Exam2>B</Exam2>
        <Final>A</Final>
    </Grades>
</Student>
```

Lire Sélectionner des nœuds en fonction de leurs enfants en ligne:
https://riptutorial.com/fr/xpath/topic/6504/selectionner-des-nouds-en-fonction-de-leurs-enfants

# Chapitre 11: Vérifier si un noeud est présent

## Syntaxe

- booléen (path_to_node)

## Remarques

La fonction booléenne a d'autres utilisations

1. Vérifiez si une chaîne est vide
2. Vérifiez si l'argument n'est pas un nombre (NaN) ou s'il s'agit de 0

## Examples

### Est-ce que l'animal a des défenses?

XML

```
<Animal>
    <legs>4</legs>
    <eyes>2</eyes>
    <horns>2</horns>
    <tail>1</tail>
</Animal>
```

XPATH

```
boolean(/Animal/tusks)
```

SORTIE

```
false
```

### Est-ce que l'animal a des cornes?

XPATH

```
<Animal>
    <legs>4</legs>
    <eyes>2</eyes>
    <horns>2</horns>
    <tail>1</tail>
</Animal>
```

XPATH

```
boolean(/Animal/horns)
```

SORTIE

```
true
```

## Y a-t-il des plantes dans la maison?

XML

```
<House>
    <LivingRoom>
        <plant name="rose"/>
    </LivingRoom>
    <TerraceGarden>
        <plant name="passion fruit"/>
        <plant name="lily"/>
        <plant name="golden duranta"/>
    </TerraceGarden>
</House>
```

XPATH

```
boolean(/House//plant)
```

SORTIE

```
true
```

Lire Vérifier si un noeud est présent en ligne: https://riptutorial.com/fr/xpath/topic/7432/verifier-si-un-noeud-est-present

# Chapitre 12: Vérifiez si le texte d'un nœud est vide

## Syntaxe

- booléen (path_to_node / text ())
- string (path_to_node)! = ''

## Remarques

La fonction booléenne a d'autres utilisations

1. Vérifier si un noeud est présent
2. Vérifiez si l'argument n'est pas un nombre (NaN) ou s'il s'agit de 0

La fonction String est utilisée pour renvoyer la valeur de chaîne d'un noeud.

## Examples

### Vérifiez si Deborah a un maître et que sa valeur textuelle n'est pas vide

XML

```
<Deborah>
    <address>Dark world</address>
    <master>Babadi</master>
    <ID>#0</ID>
    <colour>red</colour>
    <side>evil</side>
</Deborah>
```

XPATH

```
boolean(/Deborah/master/text())
```

OU

```
string(/Deborah/master) != ''
```

SORTIE

```
true
```

### Vérifiez si Dobby possède un maître et que sa valeur textuelle n'est pas vide

XML

```
<Dobby>
    <address>Hogwartz</address>
    <master></master>
    <colour>wheatish</colour>
    <side>all good</side>
</Dobby>
```

XPATH

```
boolean(/Dobby/master/text())
```

OU

```
string(/Dobby/master) != ''
```

SORTIE

```
false
```

Lire Vérifiez si le texte d'un nœud est vide en ligne:
https://riptutorial.com/fr/xpath/topic/7445/verifiez-si-le-texte-d-un-noud-est-vide

# Crédits

| S. No | Chapitres | Contributeurs |
|---|---|---|
| 1 | Démarrer avec xpath | Community, hielsnoppe, kjhughes, Vinay, Wolfgang Schindler |
| 2 | Chemins de localisation et axes | nwellnhof |
| 3 | Espaces de noms | 4444, Keith Hall |
| 4 | Recherche d'éléments contenant des attributs spécifiques | Keith Hall |
| 5 | Recherche d'éléments contenant un texte spécifique | Keith Hall |
| 6 | Rechercher des nœuds ayant un attribut spécifique | Keith Hall, miken32, suchitra nair |
| 7 | Récupère le nombre de nœuds | suchitra nair |
| 8 | Récupère les nœuds relatifs au nœud actuel | suchitra nair |
| 9 | Sélectionner des noeuds dont le nom est égal ou contenant une chaîne | Dimitre Novatchev, suchitra nair |
| 10 | Sélectionner des nœuds en fonction de leurs enfants | Matthew |
| 11 | Vérifier si un noeud est présent | suchitra nair |
| 12 | Vérifiez si le texte d'un nœud est vide | suchitra nair |