Bases de données documentaires et distribuées
Cours NFE04
XPath, l'essentiel

Auteurs : Raphaël Fournier-S'niehotta, Philippe Rigaux, Nicolas Travers
prénom.nom@cnam.fr

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

XPath

XPath est un langage **d'expression de chemins**.

Il permet l'accès à des nœuds d'un documents XML en décrivant les chemins menant ces nœuds.

XPath est un outil de base dans la manipulation de documents XML. Utilisé dans

XQuery	XPath fait partie du langage d'interrogation XQuery,
XSLT	pour la sélection de la partie des données à transformer,
XML Schema	pour les expressions de contraintes d'unicité,
XPointer	pour l'identification de fragments,
XLink	pour ancrer les liens hypertextes,
...	

Ce qui est montré dans ce qui suit : XPath 1.0

Plan de la présentation

Une intuition

Permet de “pointer” (localiser) des nœuds dans un document XML.

Intuition

Par **navigation** : en se déplaçant dans l'arbre de "étape par étape" à partir de la racine, on va chercher à atteindre certaines de ses parties.

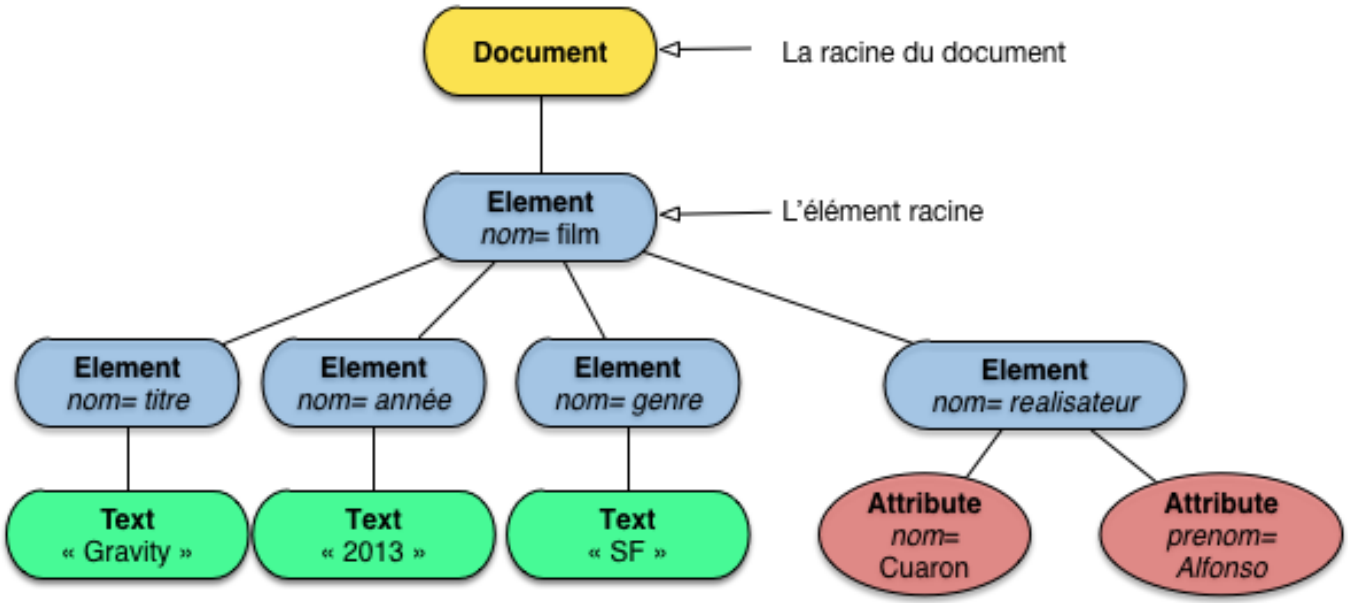
Résultat d'une expression XPath

Le résultat de l'évaluation d'une expression XPath est un **ensemble de nœuds** sélectionnés dans le document en entrée.

Attention. Un nœud peut être racine d'un sous-arbre ; dans ce cas c'est l'ensemble du sous-arbre qui fait partie du résultat.

Commençons par un exemple : `/film/titre/text()`

Vous vous souvenez sans doute de notre document XML représentant le film Gravity, dans sa forme arborescente. Notez bien les types DOM des nœuds

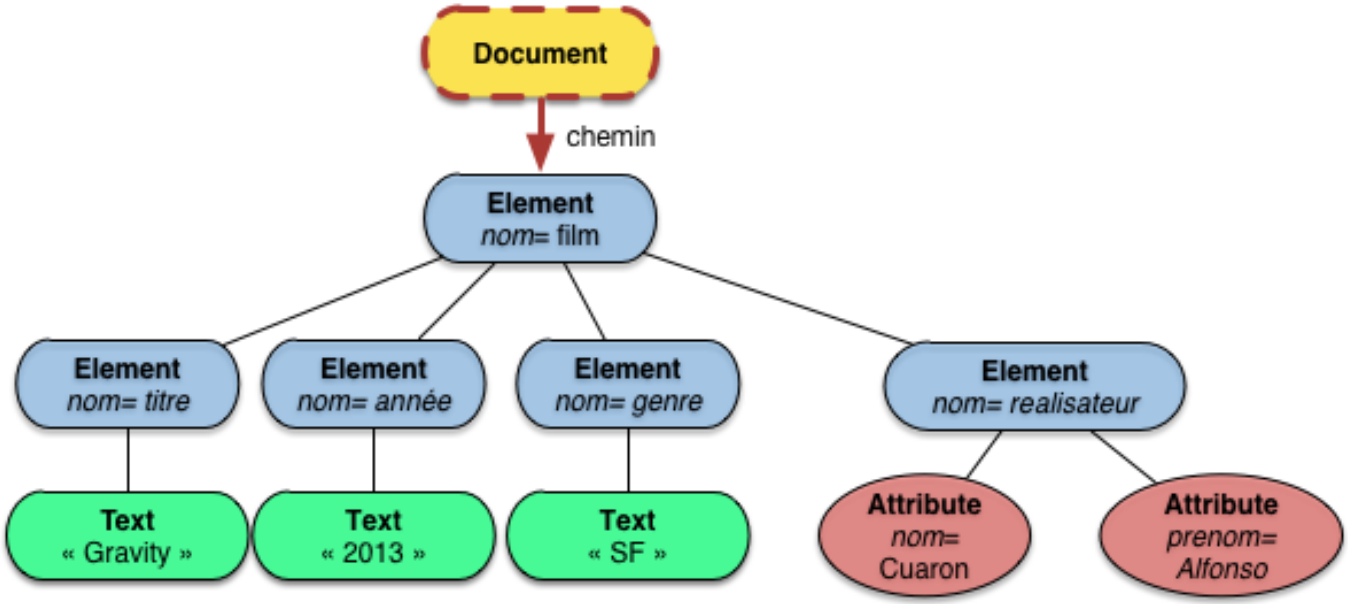


Remarque

Dans ce qui suit le nœud courant est marqué par des **bords rouges en pointillés**, les nœuds sélectionnés par des **bords rouges pleins** et les chemins par des flèches.

Commençons par un exemple : `/film/titre/text()` (suite)

Notre expression commence par un '/' : c'est une **expression absolue** qui prend pour nœud courant initial la **racine du document**.

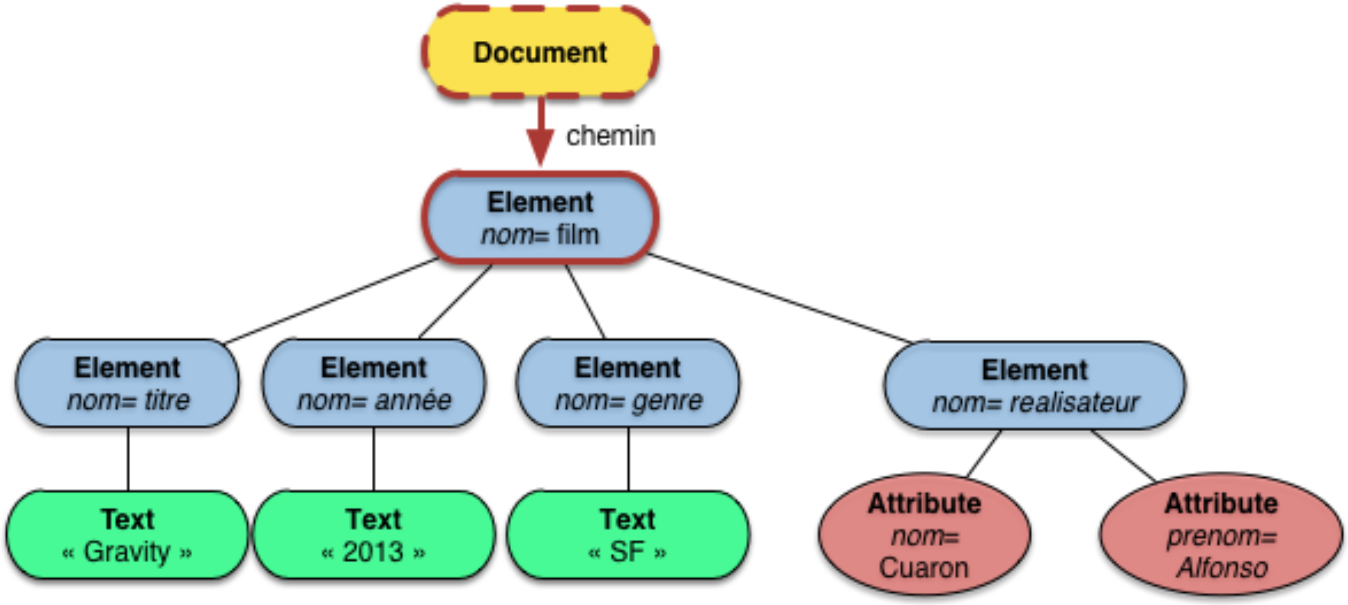


La navigation

On navigue vers le bas, vers les enfants du nœud courant.

Commençons par un exemple : `/film/titre/text()` (suite)

Une expression XPath est constituée **d'étapes**. La première étape est `film` : parmi les enfants du nœud courant, on prend celui dont le **nom** est `film`.

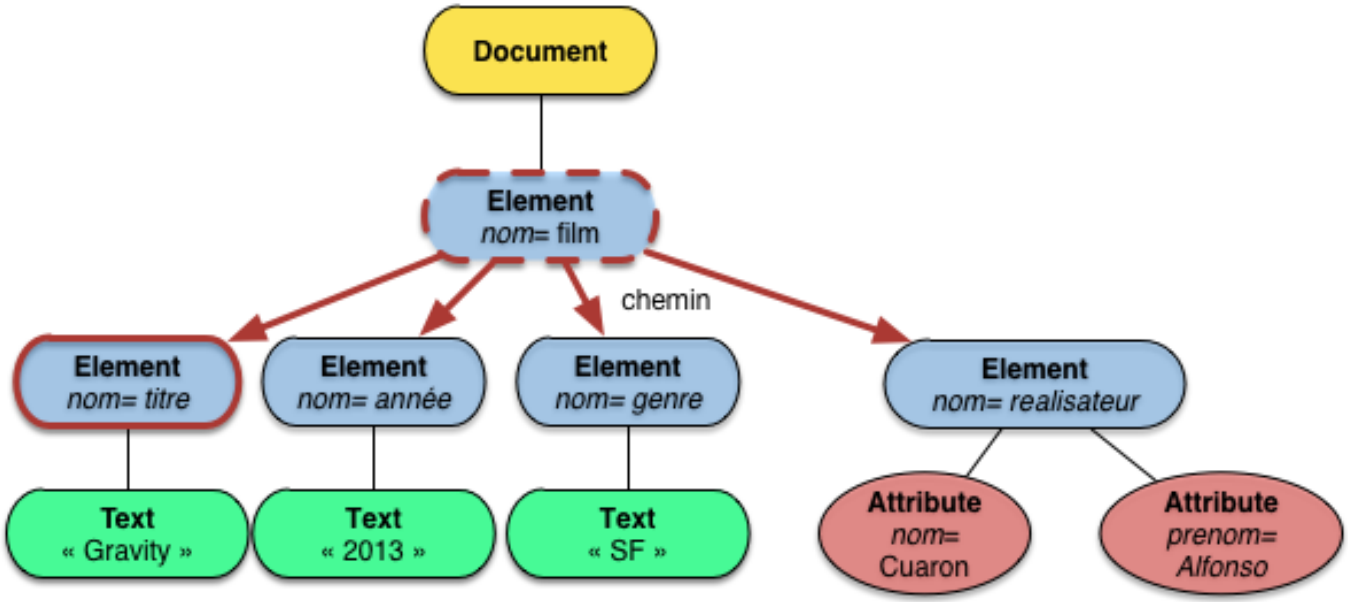


Les tests de nœud

La sélection d'un nœud par son nom est un exemple de **test de nœud** (*node test*). Avec XPath on peut appliquer des filtres sur la **structure** avec les tests de nœuds et sur le **contenu** avec les **prédicats**.

Commençons par un exemple : `/film/titre/text()` (suite)

On passe à l'étape suivante : `titre`. Maintenant, le **nœud courant** est `film`, résultat de l'étape précédente. On regarde les enfants et on sélectionne celui dont le nom est `titre`



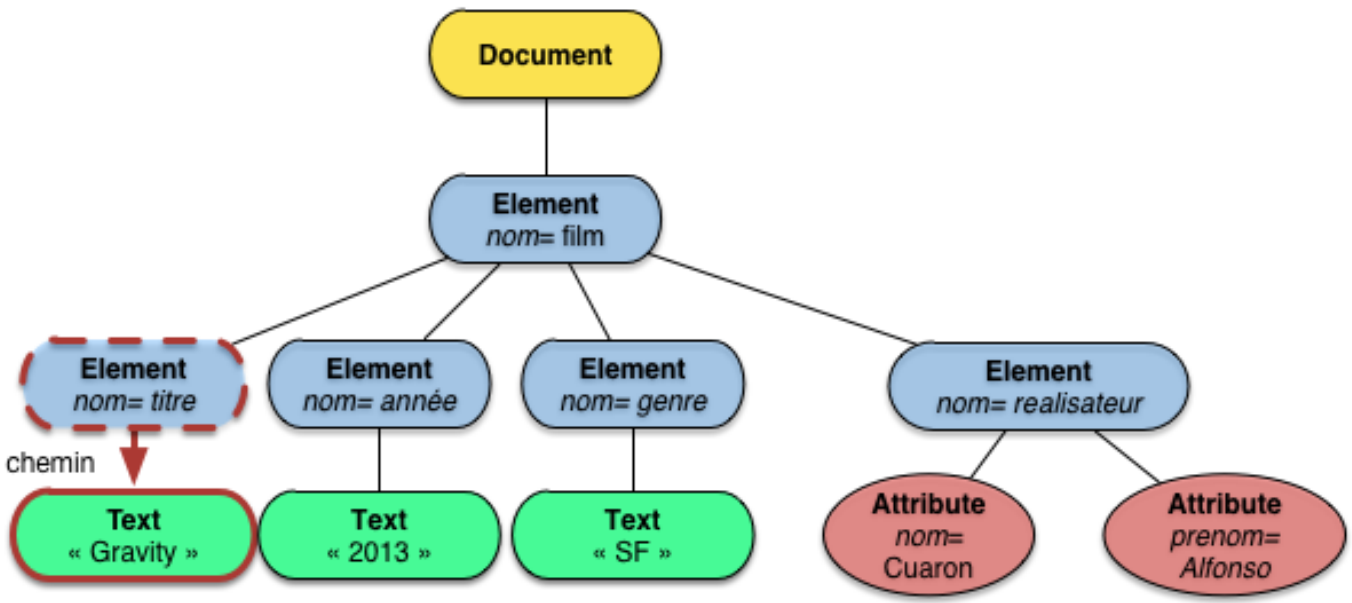
Etape et nœud courant

Le nœud courant à l'étape n est un nœud sélectionné à l'étape $n-1$. On répète l'étape n autant de fois qu'il y a de nœuds sélectionnés.

Commençons par un exemple : `/film/titre/text()` (suite)

On arrive au bout ! La dernière étape est `text()`. C'est un test de nœud sur le **type** du nœud, et non sur le nom.

On prend donc, parmi les enfants du nœud courant `titre`, ceux de type **Text**.



C'est fini : l'expression a donné chemin qui va de la racine du document vers le nœud-texte `Gravity`.

Plan de la présentation

Qu'est-ce qu'une **étape** XPath

Une **étape** est une expression de la forme `axis::node-test[predicate]`

`axis` est la direction de la navigation
Par défaut, on descend vers les enfants, mais beaucoup d'autres axes sont possibles.

`node-test` est un test de structure
Il porte sur le type (DOM) du nœud ou sur le nom du nœud (éléments et attributs seulement)

`[predicate]` (optionnel) est un test sur le contenu
On prend les valeurs de nœuds-texte ou des attributs, on les compare à des constantes ou entre eux : un peu à la SQL.

Dans `child::realisateur[@nom='Cuaron']`, `child` est l'axe, `library` le test, le prédicat est entre crochets.

Interprétation d'une expression

Principe général

Une expression XPath est une suite d'étapes (*steps*), chaque étape comprenant éventuellement des tests sur les nœuds atteints pour en éliminer certains.

A chaque étape :

1. on considère un **nœud courant**, pris comme point de départ ;
2. on "désigne" à partir du nœud courant un ensemble d'autres nœuds ;
3. on applique à chacun les tests de nœud ;
4. on applique les prédicats (optionnels) ;
5. enfin on prend chacun des nœuds restant comme nœud courant pour l'étape suivante.

Les axes dans XPath

À partir d'un nœud de l'arbre (nœud courant), les axes permettent de se déplacer dans tous les sens.

On va se contenter du minimum (qui est aussi le plus utile !)

- L'axe `child` : les enfants du nœud courant.
- `descendant` : tous les descendants du nœud courant.
- `parent` : l'ascendant direct du nœud courant.
- `attribute` : les attributs du nœud courant.
- `self` : le nœud courant lui-même.

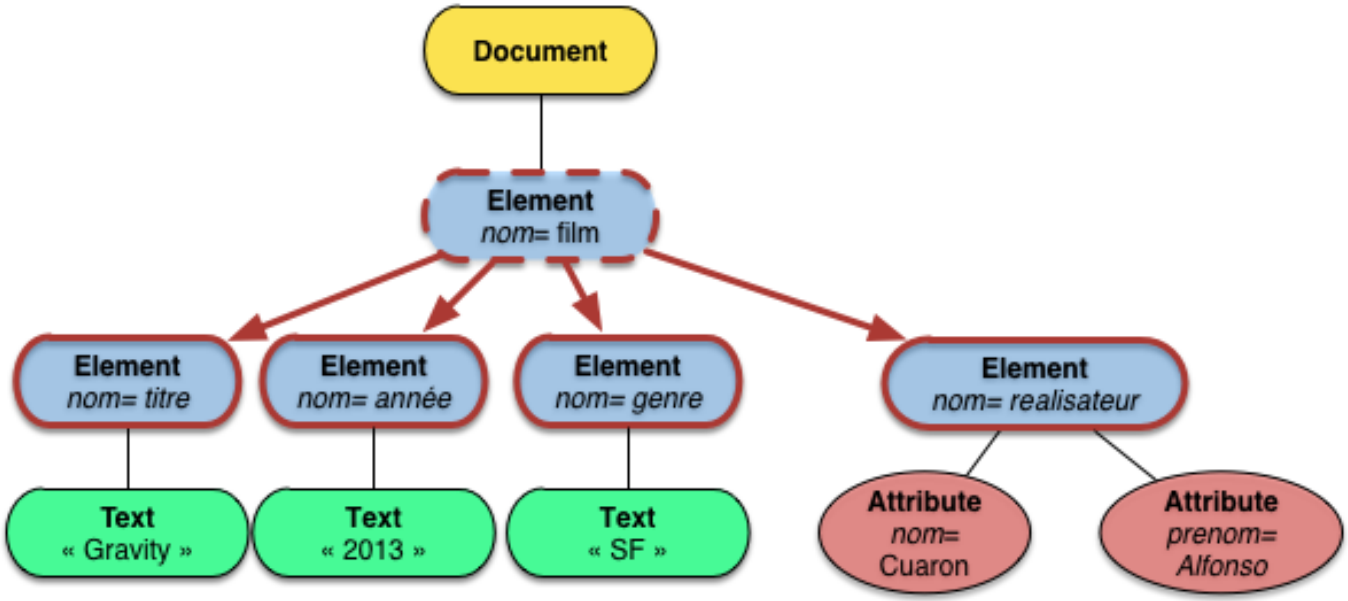
Remarque

Si vous êtes amenés à utiliser XPath intensivement, il faudra aller plus loin : reportez-vous au livre `http://webdam.inria.fr/jorge` pour une présentation complète.

L'axe `child`

Pas de surprise : à chaque **étape**, on se dirige vers les enfants du nœud courant.

La figure montre les nœuds désignés par l'expression `child::node()`, pour le nœud courant `film`.



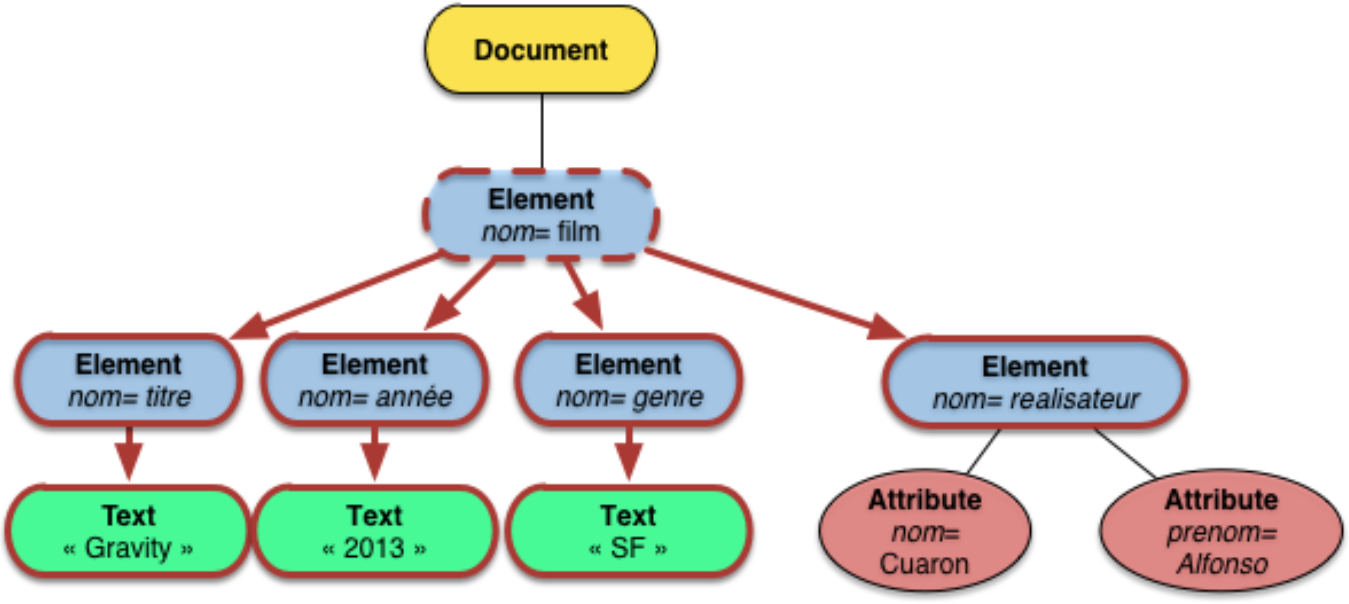
Remarque

Rappel : `node()` est le test "neutre" : il renvoie vrai pour tous les nœuds (ici, sans tenir compte du **nom** du nœud.

L'axe `descendant`

On parcourt les enfants, puis les enfants des enfants, jusqu'aux feuilles.

La figure montre les nœuds désignés par l'expression `descendant::node()`, pour le nœud courant `film`.



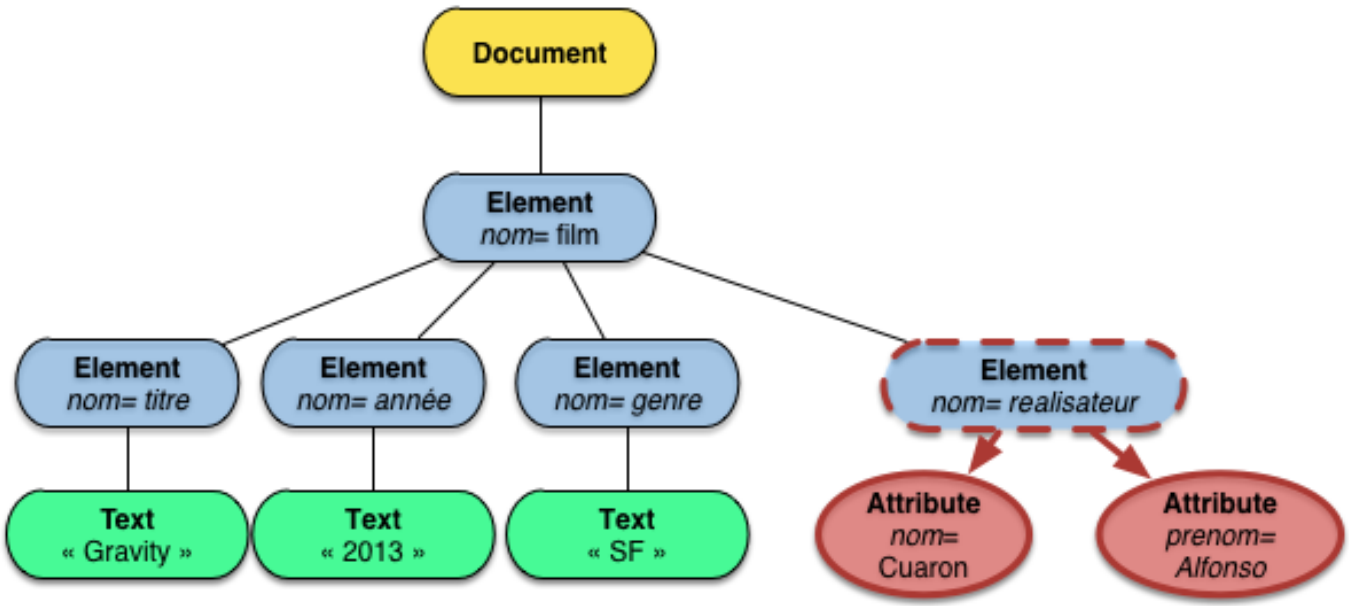
Remarque

Si, quand même une petite surprise : les **attributs** ne sont pas concernés.

L'axe `attribute`

Les attributs sont spéciaux : on ne peut y accéder qu'en utilisant explicitement l'axe `@` (ou `attribute`).

La figure montre les nœuds désignés par l'expression `@*`, pour le nœud courant `réalisateur`.



Plan de la présentation

Les tests de nœuds

Un axe désigne un ensemble de nœuds (par exemple, les enfants du nœud courant).

On les **filtre** avec un test (`node-test`).

On peut tester sur le type (DOM) du nœud :

`node()`	sélectionne les nœuds quel que soit leur type
`text()`	sélectionne les nœuds de type **Text**.
`element()`	sélectionne les nœuds de type **Element**.
	et quelques autres...

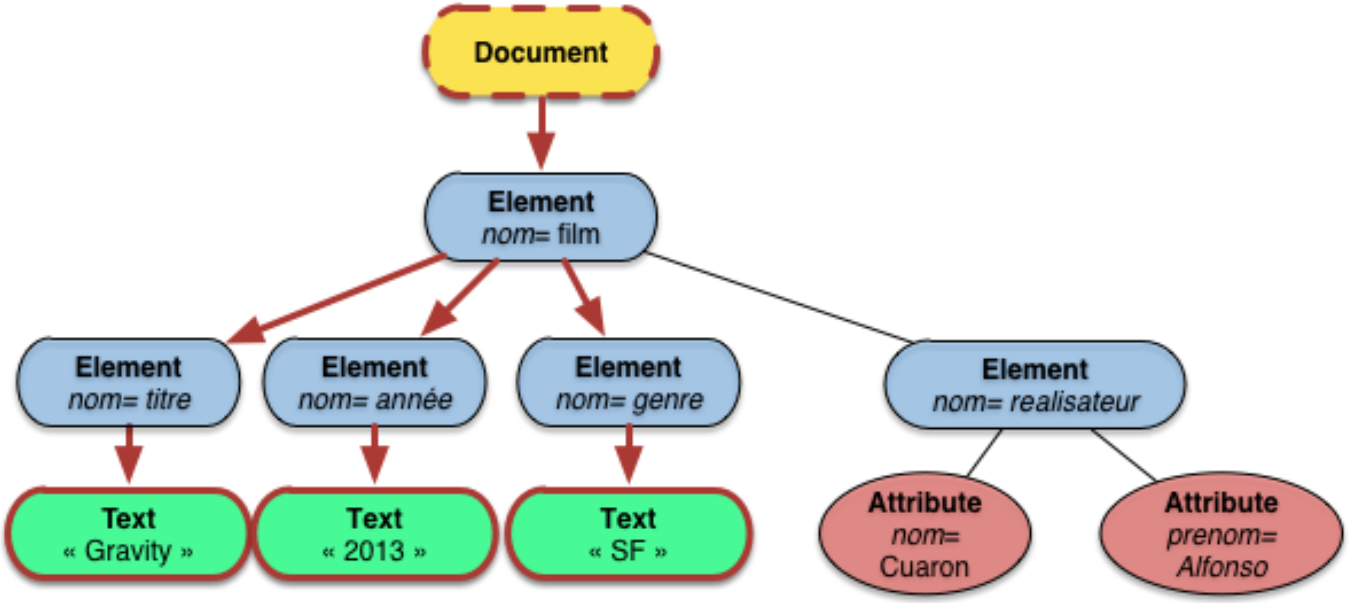
Ou bien tester sur le nom du nœud (seulement pour éléments et attributs).

`leNom`	sélectionne les éléments ou attributs de nom `leNom`
`*`	sélectionne les nœuds qui ont un nom, quel que soit le nom.

Exemple : sélection des nœuds texte

On veut tous les nœuds de type **Text**.

Expression : `/film/*/text()`.



Quels sont les tests dans cette expression ? Quels sont les axes ?

Les prédicats

Alors que les `node-test` portent sur la **structure** du document, les **prédicats** portent sur le contenu.

Quelques exemples, mieux que les discours.

- Le titre du film réalisé par M. Cuaron
  `/film[realisateur/@nom='Cuaron']/titre`
- Les films de SF : `/film[genre='SF']/titre`
- Les personnes dont on connaît la date de naissance :
  `/personne[@date_naissance]`
- Les films dont le genre est soit SF, soit comédie :
  `/film[genre='SF' or genre='Comedy']/titre`

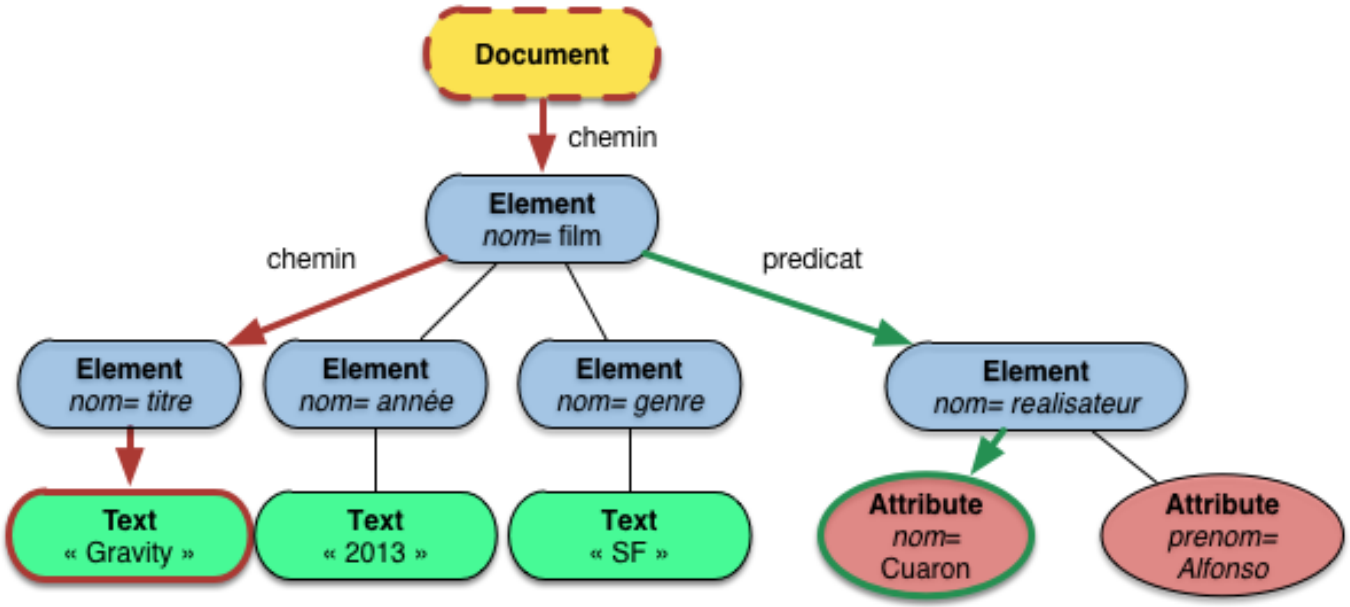
Le principe

Un nœud *n* est sur un chemin. Pour savoir si on le garde ou pas : tests sur des **valeurs** ramenées par des expressions **relatives** à *n*.

Un exemple détaillé

Films réalisés par M. Cuaron :

```
/film[realisateur/@nom='Cuaron']/titre
```



Ce qu'il vaut mieux cacher...

Que se passe-t-il quand l'expression d'un prédicat désigne **plusieurs** nœuds que l'on compare à **une** valeur ? Passons pudiquement...

Abréviations

Certaines expressions très courantes ont une abréviation.

. désigne le nœud courant (abrégé de `self::node()`)

.. désigne le parent du nœud courant (abrégé de `parent::node()`)

`//` est une expression abrégée désignant tous les descendants du nœud courant, y compris le nœud courant lui-même.

- `//a` désigne tous les nœuds du document dont le nom est `a`.
- `.//b` désigne tous les descendants du nœud courant (ou lui-même) dont le nom est `b`.

Opérations et fonctions

XPath permet aussi d'effectuer des opérations et fonctions. Mini sélection :

- `count(nset)` nombre d'éléments dans l'ensemble de nœuds.
- `name(nœud)` nom du nœud.
- `concat(ch1, ..., chn)`, concaténation de textes
- `contains(ch1,ch2)`, vrai si `ch1` contient `ch2`.

Fonctions sur les booléens

- `not(bool)`

Remarque

La version 2 de XPath est un sous-ensemble de XQuery : toutes les fonctions XQuery sont disponibles, il y en a beaucoup et c'est extensible.

Bilan

Nous en savons assez pour faire beaucoup de choses intéressantes.

- Tous les nœuds texte du document : `//text()`
- Les nœuds texte dont le nom du parent est `titre` : `//text()[parent::titre]`
- Les nœuds texte dont le nom du parent est `titre` et qui contiennent la sous-chaîne `'ty'`. `//text()[parent::titre and contains(., 'ty')]`
- Les nœuds qui ont plus de deux enfants. `//*[count(*) >= 2]`

Pour conclure

Deux remarques

- XPath sélectionne des nœuds existant ; on ne peut pas construire de nouveaux documents, de nouvelles valeurs.
- Typique d'une approche où on peut interroger à la fois la structure et le contenu.