EBook Gratuito

APPRENDIMENTO

xpath

Free unaffiliated eBook created from
Stack Overflow contributors.

#xpath

Sommario

Di

Capitolo 1: Iniziare con xpath

Osservazioni

XPath è un linguaggio per indirizzare parti di un documento XML.

È utilizzato in XSLT ed è un sottoinsieme di XQuery. Le librerie sono disponibili anche per la maggior parte degli altri linguaggi di programmazione.

XPath è uno standard internazionale con specifiche pubblicate dal W3C:

- XPath 1.0: XML Path Language (XPath), Versione 1.0
- XPath 2.0: XML Path Language (XPath) 2.0 (Seconda edizione)
- XPath 3.0: XML Path Language (XPath) 3.0

Versioni

Versione	Data di rilascio
1.0	1999/12/16
2.0	2007-01-23
3.0	2014/04/08
3.1 (Raccomandazione del Candidato W3C)	2015/12/17

Examples

XML di esempio (senza spazi dei nomi)

Ecco alcuni esempi di XML rispetto agli esempi di XPath che possono essere scritti:

```
<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>
```

Seleziona il testo

Per l'XML di esempio (senza spazi dei nomi):

Questo XPath,

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

selezionerà il nodo di testo con questo valore stringa:

```
"Text 1"
```

E questo XPath,

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

restituirà il valore stringa di `f` , che è anche:

```
"Text 1"
```

Seleziona un elemento

Per l'XML di esempio (senza spazi dei nomi):

Questo XPath,

```
/r/e
```

selezionerà questo elemento:

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

Operazioni HTML comuni

Se il DOM HTML di input è

```
<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>
```

Trova un elemento con un ID specifico nell'intera pagina

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

Trova un elemento con un ID specifico in un particolare percorso

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

Seleziona un elemento con un particolare ID e classe

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

Seleziona il testo di un particolare elemento

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

Testare Xpath nella console del browser

Un modo rapido per testare xpath è nella console dello sviluppatore del browser.

Il formato è

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

$ - specifica che è un selettore.

x - specifica che sta usando xpaths

Esempio:

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

Quando viene immesso questo comando, verranno restituite tutte le occorrenze di elementi che sono pulsanti con testo uguale a Invia.

Leggi Iniziare con xpath online: https://riptutorial.com/it/xpath/topic/883/iniziare-con-xpath

Capitolo 2: Controlla se il testo di un nodo è vuoto

Sintassi

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

Osservazioni

La funzione booleana ha altri usi

1. Controlla se un nodo è presente
2. Controlla se l'argomento non è un numero (NaN) o è 0

La funzione stringa viene utilizzata per restituire il valore stringa di un nodo.

Examples

Controlla se Deborah ha un master e il suo valore di testo non è vuoto

XML

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

XPATH

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

O

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

PRODUZIONE

```
true
```

Controlla se Dobby ha un master e il suo valore di testo non è vuoto

XML

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

XPATH

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

O

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

PRODUZIONE

```
false
```

Leggi Controlla se il testo di un nodo è vuoto online:
https://riptutorial.com/it/xpath/topic/7445/controlla-se-il-testo-di-un-nodo-e-vuoto

Capitolo 3: Controlla se un nodo è presente

Sintassi

- booleana (path_to_node)

Osservazioni

La funzione booleana ha altri usi

1. Controlla se una stringa è vuota
2. Controlla se l'argomento non è un numero (NaN) o è 0

Examples

L'animale ha le zanne?

XML

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

XPATH

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

PRODUZIONE

```
false
```

L'animale ha le corna?

XPATH

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

XPATH

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

PRODUZIONE

```
true
```

Ci sono piante in casa?

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)
```

PRODUZIONE

```
true
```

Leggi Controlla se un nodo è presente online: https://riptutorial.com/it/xpath/topic/7432/controlla-se-un-nodo-e-presente

Capitolo 4: Namespace

Osservazioni

XPath 1.0 non ha il concetto di uno spazio dei nomi predefinito.

Inoltre, i prefissi dei namespace definiti nel documento XML originale non influiscono su XPath - i prefissi dei namespace devono essere esplicitamente registrati con il provider XPath, altrimenti i prefissi non possono essere utilizzati affatto nell'espressione XPath.

Examples

Funzioni per lo spazio dei nomi

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

L'espressione `/root` non restituirà nulla, perché non esiste un elemento non nominato dallo spazio chiamato `root` al livello root del documento. Tuttavia, i seguenti *restituirà* il `<root xmlns="http://test/">` elemento.

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

Leggi Namespace online: https://riptutorial.com/it/xpath/topic/2324/namespace

Capitolo 5: Ottieni il conteggio dei nodi

Sintassi

- contare (node-set)

Parametri

funzione	ritorna
contare	numero totale di nodi nel set di nodi

Osservazioni

Possiamo usarlo in combinazione con altre funzioni e assi per soddisfare le nostre esigenze.

Examples

Quanti bambini ha Goku?

XML

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

XPATH

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

PRODUZIONE

```
2.0
```

Quante piante ci sono in casa?

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)
```

PRODUZIONE

```
4.0
```

Leggi Ottieni il conteggio dei nodi online: https://riptutorial.com/it/xpath/topic/4463/ottieni-il-conteggio-dei-nodi

Capitolo 6: Ottieni nodi relativi al nodo corrente

Sintassi

1. Tutti gli antenati di un nodo
    - / percorso per il nodo / ancestor :: node ()
2. Un antenato specifico di un nodo
    - / percorso per il nodo / antenato :: nome_antentore
3. Padre di un nodo
    - / percorso per il nodo / parent :: node ()
4. Seguendo fratelli di un nodo
    - / percorso per il nodo / follow-sibling :: node ()
5. Un fratello specifico che segue un nodo
    - / percorso per il nodo / seguente-fratello :: nome_bibbia
6. Fratelli e sorelle di un nodo
    - / percorso per il nodo / precedente-sibling :: nodo ()
7. Un fratello specifico che precede un nodo
    - / percorso per il nodo / precedente-fratello :: nome_bibbia
8. Tutti i nodi figlio immediati di un nodo
    - / percorso per il nodo / child :: node ()
9. Uno specifico nodo figlio immediato di un nodo
    - / percorso per il nodo / child :: chid_name
10. Tutti i discendenti di un nodo
    - / percorso per il nodo / discendente :: nodo ()
11. Tutti i discendenti specifici di un nodo
    - / percorso il nodo / discendente :: nome discendente

Parametri

Asse	seleziona
antenato	tutti i nodi degli antenati
genitore	nodo genitore
following-sibling	fratelli che seguono il nodo
preceding-sibling	fratelli che precedono il nodo
bambino	bambini immediati
discendente	tutto il discendente indipendentemente dal livello di nidificazione

Osservazioni

Questi assi possono essere utilizzati in combinazione con altre funzioni per soddisfare le nostre esigenze.

Examples

Trova i miei antenati

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()
```

PRODUZIONE

```
<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>
```

Trova il mio genitore

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
```

o

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

PRODUZIONE

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

Trova mio nonno

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
```

o

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

PRODUZIONE

```
<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>
```

Trova mio fratello

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
```

PRODUZIONE

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

Ottieni tutti gli avatar prima di 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()
```

PRODUZIONE

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

Ottieni tutti gli avatar dopo 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()
```

PRODUZIONE

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

Ottieni tutti gli avatar tranne quello attuale (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
```

PRODUZIONE

```
<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" />
```

Ottieni tutti i dettagli (nodi figli) di House

XML

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

XPATH

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

PRODUZIONE

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

Ottieni tutte le stanze (bambini immediati di nome Room) in House

XML

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

XPATH

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

o

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

PRODUZIONE

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

Ottieni tutte le stanze (indipendentemente dalla posizione) in House

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
```

PRODUZIONE

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

Leggi Ottieni nodi relativi al nodo corrente online: https://riptutorial.com/it/xpath/topic/6495/ottieni-nodi-relativi-al-nodo-corrente

Capitolo 7: Percorsi di posizione e assi

Osservazioni

Un *percorso di posizione* XPath è una serie di *passaggi* di *posizione* separati da un `/` personaggio:

```
step1/step2/step3
```

Un *passaggio di posizione* contiene un *asse* , un *test del nodo* e un elenco facoltativo di *predicati* . L' *asse* e il *test* del *nodo* sono separati da due caratteri di due punti `::` . I *predicati* sono racchiusi tra parentesi quadre:

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

La valutazione di un *percorso di ubicazione* inizia con un set di nodi contenente il *nodo di contesto* dato dal contesto dell'espressione o dal *nodo radice* , se il percorso di ubicazione inizia con un `/` . Ad ogni passaggio, ogni nodo *N* nel set di nodi originale viene sostituito con l'insieme di nodi che

- può essere raggiunto da *N* seguendo l' *asse* dato,
- corrisponde al *test* del *nodo* e
- corrisponde a tutti i *predicati* .

Il risultato di un'espressione del *percorso di posizione* è il set di nodi finale ottenuto dopo l'elaborazione di tutti i *passaggi di posizione* .

Examples

Attraversando elementi figlio

Spostamento dal nodo radice a un elemento discendente utilizzando l'asse `child` :

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

Poiché l'asse `child` è l'asse predefinito, questo può essere abbreviato in:

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

Attraversando tutti i discendenti

Gli assi `descendant` e `descendant-or-self` possono essere utilizzati per trovare tutti gli elementi discendenti di un nodo a qualsiasi profondità. Al contrario, l'asse `child` attraversa solo i bambini immediati.

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

La doppia barra `//` è una scorciatoia per `/descendant-or-self::node()/` . Quindi le seguenti espressioni sono equivalenti:

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

Attraversando gli antenati

L'asse `parent` contiene solo il genitore di un nodo. L'espressione seguente seleziona l'elemento `html` effettuando una deviazione sull'elemento `body` :

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

`..` è una scorciatoia per `parent::node()`

Gli assi degli `ancestor` e degli `ancestor-or-self` attraversano tutti gli antenati di un nodo. L'espressione seguente restituisce tutti gli elementi `div` che sono gli antenati del nodo di contesto:

```
ancestor::div
```

L'asse "auto"

L'asse `self` contiene solo il nodo di contesto stesso. L'espressione `.` è una scorciatoia per `self::node()` e corrisponde sempre al nodo di contesto. Il `.` il collegamento è utile per enumerare i discendenti del nodo di contesto. Le seguenti espressioni sono equivalenti:

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

L' `self` asse può essere utile in XPath 1.0 predicati. Ad esempio, per selezionare tutti i figli `h1` , `h2` e `h3` del nodo di contesto:

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

Attraversando i nodi seguenti e quelli precedenti

I `following-sibling` assi `following-sibling` e `following-sibling` `preceding-sibling` contengono i fratelli prima o dopo il nodo di contesto e gli assi `following` e `preceding` contengono tutti i nodi nel documento prima o dopo il nodo di contesto, ma:

- Nessuno di questi assi contiene nodi attributo o namespace.
- L'asse `following` non contiene discendenti.
- L'asse `preceding` non contiene alcun antenato.

Esempi:

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

Attributo di attraversamento e nodi dello spazio dei nomi

Gli assi `attribute` e `namespace` contengono tutti i nodi attributo e spazio dei nomi di un elemento. La scorciatoia `@` sta per `attribute::` , quindi i seguenti sono equivalenti:

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

Leggi Percorsi di posizione e assi online: https://riptutorial.com/it/xpath/topic/6171/percorsi-di-posizione-e-assi

Capitolo 8: Seleziona i nodi in base ai loro figli

Examples

Seleziona i nodi in base al numero di bambini

Esempio di 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

Seleziona tutti gli studenti che hanno registrato almeno 2 voti

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

Produzione

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

Selezionare i nodi in base al nodo figlio specifico

Esempio di 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

Seleziona tutti gli studenti che hanno registrato un punteggio per Exam2

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

o

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

Produzione

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

Leggi Seleziona i nodi in base ai loro figli online:
https://riptutorial.com/it/xpath/topic/6504/seleziona-i-nodi-in-base-ai-loro-figli

Capitolo 9: Seleziona nodi con nomi uguali o contenenti una stringa

Sintassi

1. All'interno di un nodo specifico:

   {path-to-parent} / name () = 'search string']

2. Ovunque nel documento:

   // * [name () = 'search string']

Parametri

funzione	valore di ritorno
local-name ()	il nome del nodo senza prefisso

Osservazioni

il risultato nome-locale () non include prefisso (funzione lookup name () XPATH per esso)

Examples

Cerca nodi con nome come Luce, Dispositivo o Sensore

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']
```

o

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

PRODUZIONE

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

Cerca nodi con nome che contiene Luce

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")]
```

o

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

PRODUZIONE

```
<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>
```

Cerca nodi con nomi che iniziano con 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")]
```

o

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

PRODUZIONE

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

Cerca i nodi che hanno un nome che termina con 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")]
```

o

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

PRODUZIONE

```
<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>
```

Cerca nodi con luce del nome (maiuscole / minuscole)

XML

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

XPATH

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

o

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

PRODUZIONE

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

Leggi Seleziona nodi con nomi uguali o contenenti una stringa online:
https://riptutorial.com/it/xpath/topic/3095/seleziona-nodi-con-nomi-uguali-o-contenenti-una-stringa

Capitolo 10: Trova nodi che hanno un attributo specifico

Sintassi

1. All'interno di un nodo specifico
    - / percorso a / elemento [@attributo_nome]
2. Ovunque nel documento
    - //*[@nome attributo]
3. All'interno di un nodo specifico con un certo valore
    - / percorso a / elemento [@ attribute_name = 'valore di ricerca']
    - / percorso a / elemento [@ attribute_name = "valore di ricerca"]
4. Ovunque nel documento con un certo valore
    - // * [@ attribute_name = 'stringa di ricerca']
    - // * [@ attribute_name = "stringa di ricerca"]

Parametri

Selettore	funzione
@nome attributo	Seleziona il valore dell'attributo per un nodo, se presente

Osservazioni

Usando [@attributo_nome] possiamo selezionare nodi che hanno l'attributo indipendentemente dal valore.

Ad esempio, con questo selettore possiamo adattare qualsiasi funzione o combinazione di funzioni come start-in e minuscolo, in base alle nostre esigenze.

Examples

Trova nodi con un attributo specifico

XML

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

XPATH

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

o

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

PRODUZIONE

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

Trova nodi con un valore di attributo specifico

XML

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

XPATH

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

o

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

PRODUZIONE

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

Trova i nodi per corrispondenza di sottostringa del valore di un attributo

XML

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

XPATH

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

o

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

Le doppie virgolette possono anche essere usate al posto delle virgolette singole:

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

PRODUZIONE

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

Trova i nodi per sottostringa che corrisponde al valore di un attributo (senza distinzione tra maiuscole e minuscole)

XML

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

XPATH

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

o

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

oppure, con la stringa tra virgolette:

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

PRODUZIONE

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

Trova i nodi per sottostringa che corrisponde all'inizio del valore di un attributo

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')]
```

o

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

PRODUZIONE

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

Trova i nodi per sottostringa che corrisponde alla fine del valore di un attributo

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')]
```

o

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

PRODUZIONE

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

Leggi Trova nodi che hanno un attributo specifico online:
https://riptutorial.com/it/xpath/topic/3096/trova-nodi-che-hanno-un-attributo-specifico

Capitolo 11: Trovare elementi contenenti attributi specifici

Examples

Trova tutti gli elementi con un determinato attributo

Immagina il seguente XML:

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

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

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

Trova tutti gli elementi con un determinato valore di attributo

Immagina il seguente XML:

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

La seguente espressione XPath:

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

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

le doppie virgolette possono anche essere usate:

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

Leggi Trovare elementi contenenti attributi specifici online:
https://riptutorial.com/it/xpath/topic/6488/trovare-elementi-contenenti-attributi-specifici

Capitolo 12: Trovare elementi contenenti testo specifico

Examples

Trova tutti gli elementi con un determinato testo

Immagina il seguente XML:

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

La seguente espressione XPath:

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

restituirà l' `<element>hello</element>` , ma non l'elemento `<another>` . Questo perché l'elemento `<another>` contiene spazi bianchi che circondano il testo `hello` .

Per recuperare sia `<element>` che `<another>` , si potrebbe usare:

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

o

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

che taglierà lo spazio circostante circostante prima di fare il confronto. Qui possiamo vedere che lo specificatore del nodo `text()` è opzionale quando si usa lo `normalize-space` .

Per trovare un elemento *contenente* testo specifico, puoi utilizzare la funzione `contains` . La seguente espressione restituirà l'elemento `<example>` :

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

Se vuoi trovare un testo che si estende su più bambini / nodi di testo, puoi utilizzarlo `.` invece di `text()` . `.` si riferisce all'intero contenuto testuale dell'elemento ed è figlio.

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

Per vedere i diversi nodi di testo, puoi usare:

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

che restituirà:

- "Ciao, "
- "Sono un esempio"
- ""

E per vedere più chiaramente l'intero contenuto del testo di un elemento, si può usare la funzione `string` :

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

o semplicemente

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

Ciao, sono un esempio.

Quest'ultimo funziona perché, se un set di nodi viene passato in funzioni come `string` , XPath 1.0 guarda solo il primo nodo (nell'ordine del documento) in quel set di nodi e ignora il resto.

così:

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

ritornerebbe:

Ciao

Leggi Trovare elementi contenenti testo specifico online:
https://riptutorial.com/it/xpath/topic/1903/trovare-elementi-contenenti-testo-specifico

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con xpath	Community, hielsnoppe, kjhughes, Vinay, Wolfgang Schindler
2	Controlla se il testo di un nodo è vuoto	suchitra nair
3	Controlla se un nodo è presente	suchitra nair
4	Namespace	4444, Keith Hall
5	Ottieni il conteggio dei nodi	suchitra nair
6	Ottieni nodi relativi al nodo corrente	suchitra nair
7	Percorsi di posizione e assi	nwellnhof
8	Seleziona i nodi in base ai loro figli	Matthew
9	Seleziona nodi con nomi uguali o contenenti una stringa	Dimitre Novatchev, suchitra nair
10	Trova nodi che hanno un attributo specifico	Keith Hall, miken32, suchitra nair
11	Trovare elementi contenenti attributi specifici	Keith Hall
12	Trovare elementi contenenti testo specifico	Keith Hall