EBook Gratis

APRENDIZAJE xpath

Free unaffiliated eBook created from
Stack Overflow contributors.

#xpath

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con xpath

Observaciones

XPath es un lenguaje para direccionar partes de un documento XML.

Se utiliza en XSLT y es un subconjunto de XQuery. Las bibliotecas también están disponibles para la mayoría de los otros lenguajes de programación.

XPath es un estándar internacional con especificaciones publicadas por W3C:

- XPath 1.0: XML Path Language (XPath), versión 1.0
- XPath 2.0: XML Path Language (XPath) 2.0 (Segunda edición)
- XPath 3.0: Lenguaje de ruta XML (XPath) 3.0

Versiones

Versión	Fecha de lanzamiento
1.0	1999-12-16
2.0	2007-01-23
3.0	2014-04-08
3.1 (Recomendación del candidato del W3C)	2015-12-17

Examples

Muestra de XML (sin espacios de nombres)

Aquí hay algunos ejemplos de XML contra los que se pueden escribir ejemplos XPaths:

```
<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>
```

Seleccionar texto

Para la muestra XML (sin espacios de nombres):

Este XPath,

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

seleccionará el nodo de texto con este valor de cadena:

```
"Text 1"
```

Y este XPath,

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

devolverá el valor de cadena de `f` , que también es:

```
"Text 1"
```

Selecciona un elemento

Para la muestra XML (sin espacios de nombres):

Este XPath,

```
/r/e
```

seleccionará este elemento:

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

Operaciones comunes de HTML

Si la entrada HTML DOM es

```
<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>
```

Encuentra un elemento con una identificación específica en toda la página.

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

Encuentra un elemento con una identificación específica en una ruta particular

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

Seleccione un elemento con un id y clase particular

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

Selecciona el texto de un elemento particular.

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

Probando Xpaths en la consola del navegador

Una forma rápida de probar su xpath es en la consola de herramientas de desarrollo de su navegador.

El formato es

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

$ - especifica que es un selector.

x - especifica que está utilizando xpaths

Ejemplo:

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

Cuando se ingrese este comando, se devolverán todas las apariciones de elementos que sean botones con texto igual a Enviar.

Lea Empezando con xpath en línea: https://riptutorial.com/es/xpath/topic/883/empezando-con-xpath

Capítulo 2: Compruebe si el texto de un nodo está vacío

Sintaxis

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

Observaciones

La función booleana tiene otros usos.

1. Compruebe si un nodo está presente
2. Compruebe si el argumento no es un número (NaN) o es 0

La función de cadena se utiliza para devolver el valor de cadena de un nodo.

Examples

Compruebe si Deborah tiene un maestro y su valor de texto no está vacío

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) != ''
```

SALIDA

```
true
```

Compruebe si Dobby tiene un master y su valor de texto no está vacío

XML

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

XPATH

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

O

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

SALIDA

```
false
```

Lea Compruebe si el texto de un nodo está vacío en línea:
https://riptutorial.com/es/xpath/topic/7445/compruebe-si-el-texto-de-un-nodo-esta-vacio

Capítulo 3: Compruebe si un nodo está presente

Sintaxis

- boolean (path_to_node)

Observaciones

La función booleana tiene otros usos.

1. Compruebe si una cadena está vacía
2. Compruebe si el argumento no es un número (NaN) o es 0

Examples

¿El animal tiene colmillos?

XML

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

XPATH

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

SALIDA

```
false
```

¿El animal tiene cuernos?

XPATH

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

XPATH

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

SALIDA

```
true
```

¿Hay plantas en la 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)
```

SALIDA

```
true
```

Lea Compruebe si un nodo está presente en línea:
https://riptutorial.com/es/xpath/topic/7432/compruebe-si-un-nodo-esta-presente

Capítulo 4: Encontrar elementos que contengan atributos específicos.

Examples

Encuentra todos los elementos con un determinado atributo.

Imagina el siguiente XML:

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

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

y devolverá el `<element foobar="hello_world" />` .

Encuentra todos los elementos con un determinado valor de atributo.

Imagina el siguiente XML:

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

La siguiente expresión XPath:

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

devolverá el `<element foobar="hello_world" />` .

También se pueden utilizar comillas dobles:

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

Lea Encontrar elementos que contengan atributos específicos. en línea: https://riptutorial.com/es/xpath/topic/6488/encontrar-elementos-que-contengan-atributos-especificos-

Capítulo 5: Encontrar elementos que contengan texto específico.

Examples

Encuentra todos los elementos con cierto texto.

Imagina el siguiente XML:

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

La siguiente expresión XPath:

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

devolverá el `<element>hello</element>` , pero no el elemento `<another>` . Esto se debe a que el elemento `<another>` contiene espacios en blanco que rodean el texto de `hello` .

Para recuperar tanto el `<element>` como el `<another>` , uno podría usar:

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

o

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

que recortará el espacio en blanco circundante antes de hacer la comparación. Aquí podemos ver que el especificador de nodo `text()` es opcional cuando se utiliza `normalize-space` .

Para encontrar un elemento *que contiene* un texto específico, puede utilizar la `contains` la función. La siguiente expresión devolverá el elemento `<example>` :

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

Si desea buscar texto que abarque varios nodos hijos / texto, puede usarlo `.` en lugar de `text()` . . Se refiere a todo el contenido del texto del elemento y sus hijos.

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

Para ver los múltiples nodos de texto, puede utilizar:

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

que volverá:

- "Hola, "
- "Yo soy un ejemplo"
- "."

Y para ver más claramente el contenido de texto completo de un elemento, se puede usar la función de `string` :

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

o solo

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

> Hola soy un ejemplo

El último funciona porque, si un conjunto de nodos se pasa a funciones como una `string` , XPath 1.0 solo mira el primer nodo (en el orden del documento) en ese conjunto de nodos, e ignora el resto.

asi que:

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

volvería:

> Hola

Lea Encontrar elementos que contengan texto específico. en línea: https://riptutorial.com/es/xpath/topic/1903/encontrar-elementos-que-contengan-texto-especifico-

Capítulo 6: Encuentra nodos que tienen un atributo específico

Sintaxis

1. Dentro de un nodo específico
    - / ruta a / elemento [@attribute_name]
2. En cualquier parte del documento
    - //*[@Nombre del Atributo]
3. Dentro de un nodo específico con algún valor
    - / ruta a / elemento [@ attribute_name = 'valor de búsqueda']
    - / ruta a / elemento [@ attribute_name = "valor de búsqueda"]
4. En cualquier parte del documento con algún valor.
    - // * [@ nombre_atributo = 'cadena de búsqueda']
    - // * [@ attribute_name = "cadena de búsqueda"]

Parámetros

Selector	función
@Nombre del Atributo	Selecciona el valor del atributo para un nodo, si está presente

Observaciones

Usando [@attribute_name] podemos seleccionar nodos que tienen el atributo independientemente del valor.

Podemos usar cualquiera de las funciones o la combinación de las funciones, como comics con y minúsculas, por ejemplo, con este selector para satisfacer nuestras necesidades.

Examples

Encuentra nodos con un atributo específico

XML

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

XPATH

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

o

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

SALIDA

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

Encuentra nodos con un valor de atributo específico

XML

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

XPATH

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

o

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

SALIDA

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

Encuentra nodos por subcadena que coincida con el valor de un atributo

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')]
```

Las comillas dobles también se pueden utilizar en lugar de comillas simples:

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

SALIDA

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

Encuentre nodos por subcadena que coincida con el valor de un atributo (no distingue mayúsculas y minúsculas)

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')]
```

o, con la cadena entre comillas dobles:

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

SALIDA

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

Encuentre nodos por subcadena que coincidan con el inicio del valor de un atributo

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')]
```

SALIDA

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

Encuentre nodos por subcadena que coincidan con el final del valor de un atributo

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')]
```

SALIDA

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

Lea Encuentra nodos que tienen un atributo específico en línea:
https://riptutorial.com/es/xpath/topic/3096/encuentra-nodos-que-tienen-un-atributo-especifico

Capítulo 7: Espacios de nombres

Observaciones

XPath 1.0 no tiene el concepto de un espacio de nombres predeterminado.

Además, los prefijos de espacio de nombres definidos en el documento XML original no afectan a XPath: los prefijos de espacio de nombres tienen que ser registrados explícitamente con el proveedor de XPath, de lo contrario, los prefijos no se pueden usar en absoluto en la expresión XPath.

Examples

Funciones conscientes del espacio de nombres

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

La expresión `/root` no devolverá nada, porque no hay ningún elemento sin espacio de nombre llamado `root` en el nivel raíz del documento. Sin embargo, la siguiente *devolverá* el `<root xmlns="http://test/">` elemento.

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

Lea Espacios de nombres en línea: https://riptutorial.com/es/xpath/topic/2324/espacios-de-nombres

Capítulo 8: Obtener el recuento de nodos.

Sintaxis

- cuenta (conjunto de nodos)

Parámetros

función	devoluciones
contar	número total de nodos en el conjunto de nodos

Observaciones

Podemos usar esto en combinación de otras funciones y ejes para satisfacer nuestras necesidades.

Examples

¿Cuántos hijos tiene Goku?

XML

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

XPATH

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

SALIDA

```
2.0
```

¿Cuántas plantas hay en la 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)
```

SALIDA

```
4.0
```

Lea Obtener el recuento de nodos. en línea: https://riptutorial.com/es/xpath/topic/4463/obtener-el-recuento-de-nodos-

Capítulo 9: Obtener nodos en relación con el nodo actual

Sintaxis

1. Todos los ancestros de un nodo.
    - / ruta al nodo / ancestor :: node ()
2. Un ancestro específico de un nodo.
    - / ruta al nodo / ancestor :: ancestor_name
3. Padre de un nodo
    - / ruta al nodo / padre :: nodo ()
4. Siguiendo a los hermanos de un nodo
    - / ruta al nodo / following-sibling :: node ()
5. Un hermano específico que sigue un nodo.
    - / ruta al nodo / following-sibling :: sibling_name
6. Hermanos anteriores de un nodo
    - / ruta al nodo / preced-sibling :: node ()
7. Un hermano específico que precede a un nodo.
    - / ruta al nodo / preced-sibling :: sibling_name
8. Todos los nodos hijos inmediatos de un nodo.
    - / ruta al nodo / child :: node ()
9. Un nodo secundario inmediato específico de un nodo
    - / ruta al nodo / child :: chid_name
10. Todos los descendientes de un nodo.
    - / ruta al nodo / descendant :: node ()
11. Todos los descendientes específicos de un nodo.
    - / ruta al nodo / descendant :: descendant_name

Parámetros

Eje	selecciona
antepasado	todos los nodos ancestros
padre	nodo padre
siguiente hermano	hermanos siguiendo el nodo
hermano precedente	hermanos que preceden al nodo
niño	niños inmediatos
descendiente	Todo el descendiente independientemente del nivel de anidación.

Observaciones

Estos ejes se pueden utilizar en combinación con otras funciones para satisfacer nuestras necesidades.

Examples

Encontrar mis ancestros

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()
```

SALIDA

```
<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>
```

Encontrar a mi padre

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()
```

SALIDA

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

Encuentra a mi abuelo

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()
```

SALIDA

```
<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>
```

Encontrar a mi hermano

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
```

SALIDA

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

Consigue todos los avatares antes de 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()
```

SALIDA

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

Consigue todos los avatares después de 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()
```

SALIDA

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

Consigue todos los avatares excepto el actual (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
```

SALIDA

```
<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" />
```

Consigue todos los detalles (nodos hijos) de la casa.

XML

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

XPATH

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

SALIDA

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

Obtener todas las habitaciones (niños inmediatos nombradas habitación) en casa

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']
```

SALIDA

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

Obtener todas las habitaciones (independientemente de la posición) en casa

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
```

SALIDA

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

Lea Obtener nodos en relación con el nodo actual en línea:
https://riptutorial.com/es/xpath/topic/6495/obtener-nodos-en-relacion-con-el-nodo-actual

Capítulo 10: Seleccionar nodos con nombres iguales o que contengan alguna cadena

Sintaxis

1. Dentro de un nodo específico:

   {ruta al padre} / nombre () = 'buscar cadena']

2. En cualquier parte del documento:

   // * [nombre () = 'cadena de búsqueda']

Parámetros

función	valor de retorno
nombre local()	el nombre del nodo sin prefijo

Observaciones

el resultado del nombre local () no incluye el prefijo (nombre de búsqueda () para la función XPATH)

Examples

Buscar nodos con nombre como luz, dispositivo o 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']
```

o

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

SALIDA

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

Buscar nodos que tengan nombre que contenga luz.

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")]
```

SALIDA

```
<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>
```

Busca nodos que tengan un nombre que comience 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")]
```

SALIDA

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

Busca nodos que tengan un nombre que termine 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")]
```

SALIDA

```
<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>
```

Buscar nodos con luz de nombre (mayúsculas y minúsculas)

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"]
```

SALIDA

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

Lea Seleccionar nodos con nombres iguales o que contengan alguna cadena en línea: https://riptutorial.com/es/xpath/topic/3095/seleccionar-nodos-con-nombres-iguales-o-que-contengan-alguna-cadena

Capítulo 11: Seleccionar nodos en función de sus hijos.

Examples

Seleccionar nodos basados en el recuento de niños

XML de muestra

```
 <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

Seleccionar todos los estudiantes que tengan al menos 2 calificaciones registradas

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

Salida

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

Seleccionar nodos basados en nodos secundarios específicos

XML de muestra

```
 <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

Selecciona a todos los estudiantes que tienen un puntaje para Exam2 registrado

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

o

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

Salida

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

Lea Seleccionar nodos en función de sus hijos. en línea: https://riptutorial.com/es/xpath/topic/6504/seleccionar-nodos-en-funcion-de-sus-hijos-

Capítulo 12: Ubicación de caminos y ejes

Observaciones

Una *ruta de ubicación de* XPath es una serie de *pasos* de *ubicación* separados por un carácter `/` :

```
step1/step2/step3
```

Un *paso de ubicación* contiene un *eje* , una *prueba de nodo* y una lista opcional de *predicados* . El *eje* y la *prueba de nodo* están separados por dos caracteres de dos puntos `::` . Los *predicados* están encerrados entre corchetes:

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

La evaluación de una *ruta de ubicación* comienza con un conjunto de nodos que contiene el *nodo de contexto* dado por el contexto de la expresión, o el *nodo raíz* , si la ruta de ubicación comienza con un `/` . En cada paso, cada nodo *N* en el conjunto de nodos original se reemplaza con el conjunto de nodos que

- Se puede llegar desde *N* siguiendo el *eje* dado,
- coincide con la *prueba de nodo* , y
- coincide con todos los *predicados* .

El resultado de una expresión de *ruta de ubicación* es el conjunto final de nodos obtenido después de procesar todos *los pasos de ubicación* .

Examples

Atravesando elementos infantiles

Recorrido desde el nodo raíz a un elemento descendiente utilizando el eje `child` :

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

Dado que el eje `child` es el eje predeterminado, se puede abreviar como:

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

Atravesando a todos los descendientes

Los ejes `descendant` y `descendant-or-self` pueden usarse para encontrar todos los elementos descendientes de un nodo a cualquier profundidad. En contraste, el eje `child` solo atraviesa niños inmediatos.

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

La barra doble `//` es un acceso directo para `/descendant-or-self::node()/` . Entonces las siguientes expresiones son equivalentes:

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

Travesía antepasados

El eje `parent` contiene solo el padre de un nodo. La siguiente expresión selecciona el elemento `html` tomando un desvío sobre el elemento del `body` :

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

`..` es un acceso directo para `parent::node()`

Los ejes `ancestor` y `ancestor-or-self` atraviesan todos los antepasados de un nodo. La siguiente expresión devuelve todos los elementos `div` que son ancestros del nodo de contexto:

```
ancestor::div
```

El eje "yo"

El eje `self` solo contiene el nodo de contexto. La expresión `.` es un acceso directo para `self::node()` y siempre coincide con el nodo de contexto. El `.` El acceso directo es útil para enumerar descendientes del nodo de contexto. Las siguientes expresiones son equivalentes:

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

El eje `self` puede ser útil en los predicados de XPath 1.0. Por ejemplo, para seleccionar todos los hijos `h1` , `h2` y `h3` del nodo de contexto:

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

Atravesando nodos siguientes y anteriores

Los `following-sibling` ejes `following-sibling` y `following-sibling` `preceding-sibling` contienen los hermanos antes o después del nodo de contexto, y los ejes `following` y `preceding` contienen todos los nodos en el documento antes o después del nodo de contexto, pero:

- Ninguno de estos ejes contiene atributos o nodos de espacio de nombres.
- El `following` eje no contiene descendientes.

- El eje `preceding` no contiene ningún ancestro.

Ejemplos:

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

Atravesando nodos de atributo y espacio de nombres

Los ejes de `attribute` y `namespace` contienen todos los nodos de atributo y espacio de nombres de un elemento. El atajo `@` representa `attribute::` , por lo que los siguientes son equivalentes:

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

Lea Ubicación de caminos y ejes en línea: https://riptutorial.com/es/xpath/topic/6171/ubicacion-de-caminos-y-ejes

Creditos

S. No	Capítulos	Contributors
1	Empezando con xpath	Community, hielsnoppe, kjhughes, Vinay, Wolfgang Schindler
2	Compruebe si el texto de un nodo está vacío	suchitra nair
3	Compruebe si un nodo está presente	suchitra nair
4	Encontrar elementos que contengan atributos específicos.	Keith Hall
5	Encontrar elementos que contengan texto específico.	Keith Hall
6	Encuentra nodos que tienen un atributo específico	Keith Hall, miken32, suchitra nair
7	Espacios de nombres	4444, Keith Hall
8	Obtener el recuento de nodos.	suchitra nair
9	Obtener nodos en relación con el nodo actual	suchitra nair
10	Seleccionar nodos con nombres iguales o que contengan alguna cadena	Dimitre Novatchev, suchitra nair
11	Seleccionar nodos en función de sus hijos.	Matthew
12	Ubicación de	nwellnhof

	caminos y ejes