XML 2010

Erik Tjong Kim Sang
erikt(at)xs4all.nl

4 oktober 2010

Vorige week

Omzetten van XML-documenten naar andere formaten met XSLT (*Extensible Style Language for Transformation*)

Toepassing van XSLT voor:

- omzetten van XML-documenten naar HTML
- selecteren van informatie uit XML-documenten

Deze week

Zoeken in XML-documenten met XPath

Wat is XPath?

XPath is een taal waarmee kan worden gezocht in XML-documenten

In XPath kan je zoekopdrachten geven, bijvoorbeeld: *geef mij de inhoud van het tweede element* `beschrijving` *in het* `toneelstuk.xml`

Het resultaat van de zoekopdracht is een deel van het XML-document

Hoe zien XPath-zoekopdrachten eruit?

XPath-zoekopdrachten bestaan uit een zoekpad in een boom die de structuur van een XML-document beschrijft

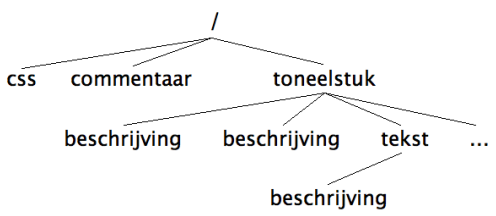
Het pad kan beginnen in de root; dan heet het een absoluut pad

Het pad kan ook beginnen in een andere knoop dan de root; dan heet het een relatief pad

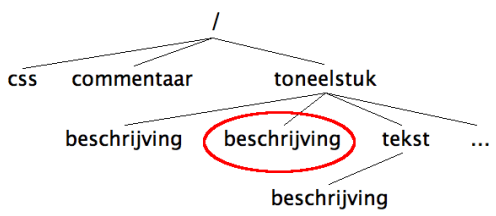
Voorbeelddocument: `toneelstuk.xml`

```
<?xml-stylesheet type="text/css" href="toneelstuk.css"?>
<!-- markup toegevoegd door: erikt -->
<toneelstuk>
  <beschrijving>
    _Veneti&euml;. Een Straat._
  </beschrijving>
  <beschrijving>
    ANTONIO, SALARINO _en_ SOLANIO _komen op._
  </beschrijving>
  <tekst spreker="ANTONIO">
    'k Weet waarlijk niet waarom 'k zoo somber ben,
```


Een boomstructuur voor `toneelstuk.xml`



Resultaat van `/toneelstuk/beschrijving[2]`



Onderdelen van zoekpaden

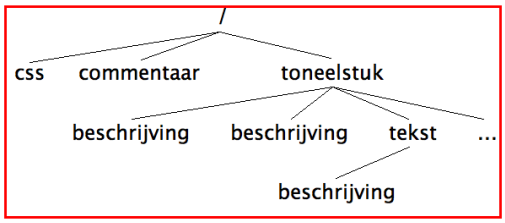
Een absoluut zoekpad begint met een slash en bevat daarna een reeks van elementnamen gescheiden door slashes (/)

Het eerste element is een kind van de root

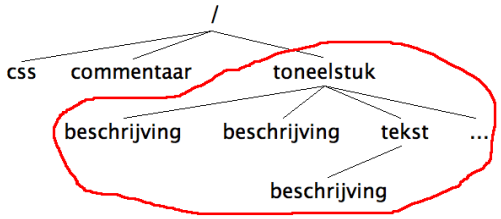
Het tweede element is een kind van het eerste element, enzovoorts

Het laatste element in de reeks wordt geselecteerd door de zoekopdracht (dit kunnen ook meerdere elementen zijn)

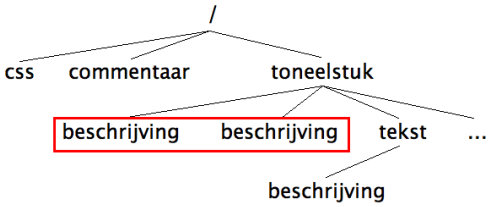
Resultaat van zoekopdracht `/`



Resultaat van zoekopdracht `/toneelstuk`



Resultaat van `/toneelstuk/beschrijving`



Descendant-or-self

De zoekopdracht `/toneelstuk/beschrijving` vindt alleen de elementen `beschrijving` die kinderen zijn van `toneelstuk`

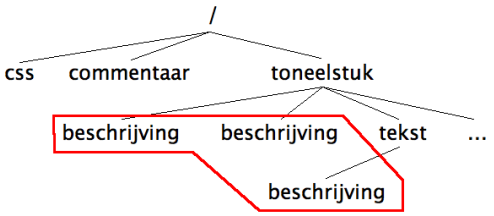
Vaak willen we alle elementen vinden met een bepaalde naam: kinderen, kleinkinderen, enzovoorts

Dit kan met de zoekopdracht `descendant-or-self` die kort wordt opgeschreven als //

Voorbeeld: `/toneelstuk//beschrijving`

Dit pad selecteert alle elementen `beschrijving` in `toneelstuk`

Resultaat van `/toneelstuk//beschrijving`



Selectie van attributen

Een attribuut kan worden geselecteerd door XPath met een pad waaraan `/@attribuutnaam` is toegevoegd

Voorbeeld: `/toneelstuk/tekst/@spreker`

Dit pad selecteert de attributen `spreker` van alle elementen `tekst` die een kind zijn van `toneelstuk`

Zoekpaden met positie-informatie

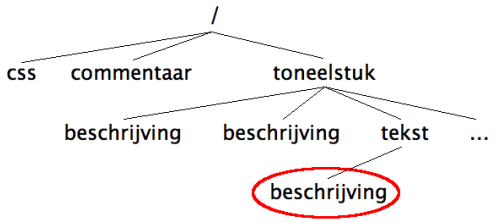
Als een element meerdere kinderen heeft met dezelfde naam dan hebben we soms een zoekpad dat slechts 1 van hen aanwijst

Indien het kindelement aanwijsbaar is door een positie dan kan het aanwijzen gebeuren door `[positie]` toe te voegen aan het pad

Voorbeeld: `/toneelstuk/tekst[1]/beschrijving[1]`

Dit pad wijst naar het eerste kind `beschrijving` van het eerste kind `tekst` van `toneelstuk`

Effect `/toneelstuk/tekst[1]/beschrijving[1]`



Zoekpaden met extra beperkingen

Een zoekpad kan ook beperkingen bevatten voor de attributen:

Deze moeten dan tussen vierkante haken achter het laatste element worden geplaatst

Voorbeeld: `/toneelstuk/tekst[@spreker='ANTONIO']`

Dit pad selecteert alleen de elementen `tekst` waarvan het attribuut `spreker` de waarde `ANTONIO` heeft

Het is ook mogelijk om te testen of de tekstinhoud van een element een bepaalde waarde heeft: `/toneelstuk[beschrijving='STOP!']`

Voorbeeld van gebruik XPath vanuit XSLT

XPath-zoekopdrachten kan je uitvoeren via een XSLT-programma

Plaats de zoekopdracht in het attribuut `select` van `apply-values`:

```
<xsl:template match="/">
  <xsl:apply-templates
   select="/toneelstuk/tekst[@spreker='ANTONIO']"/>
</xsl:template>
```

Het XSLT-programma kan vervolgens worden verwerkt door `xsltproc`:

```
$ xsltproc mijnprogramma.xsl mijnbestand.xml
```


Voorbeeld van gebruik XPath vanuit Javascript

```
<script type="text/javascript">
/* bron: http://www.w3schools.com/xpath/tryit.asp?filename=try_xpath_select_cdnodes */

/* laad XML-bestand; definieer zoekpad; zoek; pak le treffer */
xml = loadXMLDoc("books.xml");  /* functie loadXMLDoc is gedefinieerd op bronpagina */
path = "/bookstore/book/title";
nodes = xml.evaluate(path, xml, null, XPathResult.ANY_TYPE, null);
result = nodes.iterateNext();

/* zolang treffer beschikbaar */
while (result) {
  /* laat treffer zien in nieuwe paragraaf */
  document.write("<p>");
  document.write(result.childNodes[0].nodeValue);
  document.write("</p>");
  /* zoek naar volgende treffer */
  result = nodes.iterateNext();
}
</script> /* plaats dit script in een html-pagina en bekijk deze in een browser */
```


Voorbeeld van gebruik XPath vanuit Saxon

Saxon is een programma dat XLST- en XQuery-instructies kan verwerken (zie: http://saxon.sourceforge.net/)

XPath-instructies kunnen worden opgenomen in XQuery-programma's:

```
(: dit is het bestand xquery.xq :)
for $t in /toneelstuk//tekst
return <resultaat>{$t/@spreker}</resultaat>
```

Deze programma's kunnen worden gedraaid in een Java-omgeving:

```
$ java -cp saxon9he.jar net.sf.saxon.Query
       -s toneelstuk.xml -q xquery.xq
```


Voorbeelden van XPath-taken

1. Wat staat er in de beschrijvingen in teksten in `toneelstuk.xml`?
2. Wie sprak de troonrede uit?
3. Tegen wie speelde FC Groningen thuis?
4. Welke uitwedstrijden heeft PSV gewonnen?
5. Aan welk land gaf Griekenland 12 punten?

Oplossingen voor de XPath-taken

1. `//tekst/beschrijving`
2. `/troonrede/@spreker`
3. `//result[hometeam='FC Groningen']/awayteam`
4. `//result[awayteam='PSV'][homegoals < awaygoals]`
5. `//points-456[.='12'][@from='Greece']/../@name`

Relatieve paden

Een relatief pad is een pad dat niet begint met / (het rootelement)

In XSLT kunnen we zo'n pad bijvoorbeeld plaatsen in templates die voor een ander element dan / zijn gedefinieerd:

```
<xsl:template match="points-456">
  <xsl:value-of select="../@name"/>
</xsl:template>
```

Het pad `../@name` selecteert de waarde van het attribuut `name` van de ouder van het huidige element (`points-456`)

Basisbegrippen in XPath

Een zoekpad in XPath bestaat uit een aantal stappen die zijn gescheiden door slashes (/)

Iedere stap kan drie onderdelen bevatten:

- **as** (Engels: *axis*): de richting van de stap
- **element** (*node test*): de naam van een element
- **predikaat** (*predicate*): voorwaarden voor het element

Een stap ziet er schematisch uit als: `as::element[predikaat]`

Tot nu toe hebben we alleen onderdelen element en predikaat gebruikt

Wat is een as?

In een pad geeft de as de richting van de stap in een XML-boom aan

Dit kan worden vergeleken met de richtingen die iemand op kan gaan op een kruising van twee wegen: links, rechts, vooruit en terug

XPath 1.0 maakt gebruik van 13 assen

Bekende assen

We zijn 4 van de 13 assen eerder tegengekomen:

- `child`: selecteert kindelementen
  afkorting: als de as niet wordt vermeld dan wordt `child` gebruikt
- `attribute`: selecteert attributen
  afkorting: `@`
- `parent`: selecteert de ouder
  afkorting: `..`
- `descendant-or-self`: afstammelingen en het huidige element
  afkorting: `//`

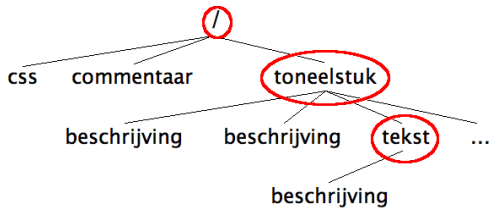
Oplossingen voor de taken met assen

1. `//tekst/beschrijving`
   `/descendant-or-self::tekst/child::beschrijving`
2. `/troonrede/@spreker`
   `/child::troonrede/attribute::spreker`
3. `//result[hometeam='FC Groningen']/awayteam`
   `/descendant-or-self::result[hometeam='FC Groningen']/child::awayteam`
4. `//result[awayteam='PSV'][homegoals < awaygoals]`
   `/descendant-or-self::result[awayteam='PSV'][homegoals < awaygoals]`
5. `//points-456[.='12'][@from='Greece']/../@name`
   `/descendant-or-self::points-456[.='12'][@from='Greece']/parent::*/attribute::name`

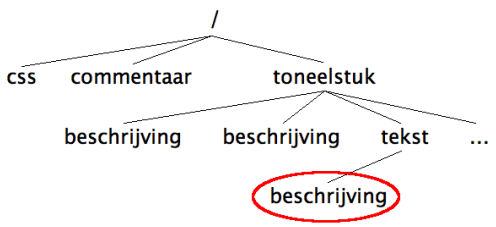
De negen andere assen

- `ancestor`: voorouders
- `ancestor-or-self`: voorouders inclusief het huidige element
- `descendant`: afstammelingen
- `following`: opvolgers (in XML-document)
- `following-sibling`: opvolgers en kind van dezelfde ouder
- `preceding`: voorgangers (in XML-element)
- `preceding-sibling`: voorgangers en kind van dezelfde ouder
- `self`: het element zelf (afkorting: .)
- `namespace`: namespace (niet meer in gebruik)

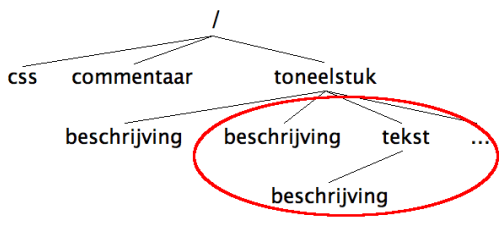
Resultaat van `//beschrijving[1]/ancestor::*`



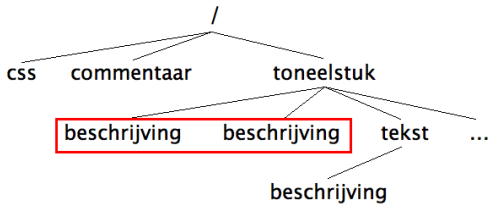
Resultaat van `//tekst[1]/descendant::*`



Resultaat van `//beschrijving[1]/following::*`



Resultaat van `//tekst[1]/preceding::*`



Inhoud van predikaten

In predikaten kunnen tests worden geplaatst om beperkingen op te leggen aan elementen of attributen

De tests maken gebruik van functies, bijvoorbeeld:

- `last()`: de laatste van een groep elementen
- `text()`: de `#PCDATA`-inhoud van een element
- `position()`: de positie van het element (getal)
- `starts-with(s1,s2)`: string `s1` begint met `s2`
- `substring(s1,n1,n2)`: geeft letters `n1`-`n2` uit string `s1`

Voorbeelden van predikaten met functies

- `/toneelstuk/tekst[last()]`
  het laatste `tekst`-element in `toneelstuk`
- `/toneelstuk[text()]`
  alle tekst in `toneelstuk`; `/toneelstuk` heeft hetzelfde effect
- `/toneelstuk/tekst[position()=1]`
  het eerste `tekst`-element; meestal afgekort tot `tekst[1]`
- `//tekst[starts-with(@spreker,'A')]`
  alle tekst van sprekers waarvan de naam begint met een `A`
- `//tekst[substring(@spreker,1,1)='S']`
  alle tekst van sprekers waarvan de naam begint met een `S`

Functies die elementen als argument nemen

XPath bevat ook functies waarmee elementen geteld kunnen worden:

- `count(pad)`: telt alle elementen die voldoen aan `pad`
- `sum(pad)`: berekent de som van de getallen in de elementen die voldoen aan `pad`
- `name(pad)`: de naam van het eerste element dat voldoet aan `pad`

Deze functies leveren geen elementen op als resultaat

In XSLT worden zij geplaatst in `<xsl:value-of select="...">`

Voorbeelden van functies met elementargumenten

- `count(/toneelstuk//beschrijving)`
  telt alle `beschrijving`-elementen in het toneelstuk
- `sum(/eurovision//points-456)`
  telt alle punten in `points-456`-elementen bij elkaar op
- `name(/toneelstuk/*)`
  geeft de naam van het eerste kind van `toneelstuk`

Case-studie: zoeken in taalkundige annotaties

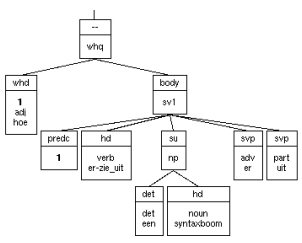
In Groningen heeft de afdeling Informatiekunde syntactisch geannoteerde teksten van ongeveer 2 miljard woorden verzameld: het LASSY-corpus

Het corpus bevat informatie over woorden en hun relaties

De extra informatie is gecodeerd in XML:

```
<alpino_ds version="1.3">
    <node id="8" pos="adj" rel="mod" word="Frans"/>
    <node id="9" pos="noun" rel="hd" word="schrijver"/>
  ...
  <sentence>Alexandre Dumas père , Frans schrijver</sentence>
```


Voorbeeld van een syntaxboom



Taken voor XPath

1. Hoe vaak zit het woord *Groningen* in het corpus?
2. Laat de zinnen met het woord *Groningen* zien
3. Hoe vaak komt het woord *Frans* voor in het corpus?
4. Hoe vaak is het woord *Frans* een bijvoegelijk naamwoord?
5. Laat de zelfstandige naamwoorden bij *Frans* zien

Taken voor XPath

1. `count(//node[@word='Groningen'])`
2. `//node[@word='Groningen']/ancestor::alpino_ds/sentence`
3. `count(//node[@word='Frans'])`
4. `count(//node[@word='Frans' and @pos='adj'])`
5. `//node[@word='Frans' and @pos='adj']/../node[@rel='hd' and @pos='noun']/@word`

Samenvatting

We hebben gezien hoe we met XPath kunnen zoeken in XML-documenten

Anders dan bij standaardzoeken, kunnen we met XPath exact aangeven naar welk deel van een document we op zoek zijn

XPath is een nuttig gereedschap voor het zoeken in gestructureerde informatie

Achtergrondliteratuur

David Hunter, Jeff Rafter, Joe Fawcett and Eric van der Vlist, *Beginning XML (Programmer to Programmer)*. John Wiley & Sons, 2004 (chapter 7: XPATH)

W3schools, *CSS Tutorial*, http://www.w3schools.com/xpath/

EINDE