



samen sterk voor werk

<XML FUNDAMENTALS/>

Deze cursus is eigendom van VDAB Competentiecentra ©

Peoplesoftcode:

Wettelijk depot:

versie: 13/3/2015

INHOUD

1	Inleiding.....	5
1.1	Doelstelling.....	5
1.2	Nodige software	5
1.3	Algemeen.....	5
1.4	Well formed.....	5
1.5	Veel gebruikte toepassingen van XML	5
1.5.1	Office documenten	5
1.5.2	Data uitwisselen op het internet	5
1.5.3	Configuratiebestanden	6
1.5.4	SVG.....	6
2	Editors	7
2.1	NetBeans	7
2.1.1	Een XML document inzien	7
2.1.2	Een XML document indenteren	7
2.1.3	Controleren of een XML document well formed is	7
2.2	Visual Studio	8
2.2.1	Een XML document inzien	8
2.2.2	Een XML document indenteren	8
2.2.3	Controleren of een XML document well formed is	8
3	Structuur van een XML document	9
3.1	XML declaration.....	9
3.2	Commentaar.....	9
4	Elementen met data tussen begintag en eindtag	10
4.1	Algemeen.....	10
4.2	Geneste elementen	10
4.3	Child, Parent, Sibling, Root.....	11
4.3.1	Child elementen.....	11
4.3.2	Parent element	11
4.3.3	Siblings	11
4.3.4	Root element	11
4.3.5	Mixed content.....	12
4.3.6	Meerdere child elementen met soortgelijke informatie.....	12
5	Elementen met data in attributen.....	13
5.1.1	Algemeen	13

5.1.2	Een element zonder eindtag	13
5.1.3	Dubbele of enkele quotes	13
5.2	Root element	13
6	Elementen met data in attributen én in child elementen	14
7	XML document als boomstructuur	16
8	Speciale tekens	17
8.1	Het teken vervangen door de bijbehorende entity	17
8.2	CDATA	17
9	Samenvatting well formed	18
10	XPath	19
10.1	Algemeen	19
10.2	Installatie en gebruik van de XPathUtil plugin in NetBeans	19
10.3	Installatie en gebruik van de XPath Navigator extension in Visual Studio	19
10.4	Slashes	20
10.4.1	Een XPath expressie die begint met één slash	20
10.4.2	Een XPath expressie die begint met twee slashes	20
10.4.3	Een XPath expressie die begint zonder slash	20
10.5	Ster	20
10.6	Twee punten	20
10.7	Text node zoeken	21
10.8	Attribuut waarde zoeken	21
10.9	Elementen zoeken op hun volgnummer ten opzichte van hun parent	21
10.10	Elementen zoeken op attribuut waarden	21
10.11	Elementen zoeken op het al of niet aanwezig zijn van attributen	22
10.12	Elementen zoeken op text node waarden	22
10.13	Statistische functies	23
11	Namespaces	24
11.1	Algemeen	24
11.2	URI's als namespaces	24
11.3	Praktisch	25
11.4	Default namespace	26
12	Regular expression	27
12.1	Controleren of een stukje tekst <i>voorkomt</i> in een tekst	27
12.2	Regular expressions uitproberen	27
12.3	Het begin van de tekst	28

12.4	Het einde van de tekst	28
12.5	Speciale tekens hun letterlijke betekenis geven	28
12.6	Een regular expression visualiseren	28
12.7	De or operator 	28
12.8	Subexpressie tussen ()	29
12.9	Om het even welk teken	29
12.10	Herhalingsoperatoren	30
12.10.1	Nul of meerdere keren *	30
12.10.2	Één of meerdere keren +	30
12.10.3	Nul of één keer ?	30
12.10.4	Een vast aantal keer {aantal}	31
12.10.5	Een minimum aantal keer {minimum,}	31
12.10.6	Een minimum én maximum aantal keer {minimum, maximum}	31
12.11	Één teken uit een lijst van tekens []	32
12.12	Één teken uit een bereik van tekens [-]	32
12.13	Alles behalve [^]	32
12.14	Metacharacters \	33
13	Schema	34
13.1	Algemeen	34
13.2	DTD	34
13.3	Een eerste schema	34
13.4	Vanuit een XML bestand verwijzen naar een schema	35
13.5	Controleren of XML data voldoet aan een schema	35
13.5.1	NetBeans	35
13.5.2	Visual Studio	35
13.6	XML verbonden met een namespace	35
13.7	XMLSchema.xsd	36
13.8	Attributen	36
13.9	Ingebouwde simple types	37
13.9.1	Tekst categorie	37
13.9.2	Getal categorie	37
13.9.3	Logische categorie	38
13.9.4	Datum – tijd categorie	38
13.10	Zelf gedefinieerde simple types	38
13.10.1	Ondergrens en/of bovengrens	39
13.10.2	Tekstlengte	40

13.10.3	Maximum aantal cijfers na de komma – maximum totaal aantal cijfers.....	40
13.10.4	Lijst	40
13.10.5	Regular expression	41
13.11	Child elementen.....	41
13.11.1	sequence – choice -all	41
13.11.2	Simple type – complex type	41
13.11.3	sequence	41
13.11.4	all.....	42
13.11.5	choice	42
13.11.6	minOccurs – maxOccurs.....	43
13.12	Een complex type met child elementen én attributen.....	44
13.13	Een complex type erft van een ander complex type	45
13.14	key.....	46
13.15	keyref	46
13.16	Documentatie	48
13.16.1	NetBeans	48
13.16.2	Visual Studio.....	48
13.17	Stappenplan	48
14	Libraries.....	50

1 Inleiding

1.1 Doelstelling

Je leert data voorstellen met XML.

1.2 Nodige software

NetBeans of Visual Studio

1.3 Algemeen

XML (eXtensible Markup Language) is een taal waarmee je data voorstelt in tekstvorm.

Zowel een mens als een programma kan de data lezen die je met XML voorstelt.

In de praktijk leest een informaticus data in XML formaat, maar niet een eindgebruiker, want XML is geen gebruikersvriendelijke voorstelling van data.

De informaticus toont de XML data in een gebruikersvriendelijk formaat aan een eindgebruiker.

XML is een standaard van het World Wide Web Consortium.

Dit is de organisatie die ook andere standaarden beheert, zoals HTML en CSS.

De standaardisatie van XML heeft veel voordelen

- Veel tools (Microsoft Office, Open Office, ...) ondersteunen XML
- Alle programmeertalen ondersteunen XML
- Alle besturingssystemen ondersteunen XML

1.4 Well formed

Een XML document moet voldoen aan een aantal regels, beschreven in de XML standaard.

Een XML document dat voldoet aan deze regels is een *well formed* XML document.

Je leert deze regels kennen in deze cursus.

1.5 Veel gebruikte toepassingen van XML

1.5.1 Office documenten

Microsoft Office en Open Office slaan documenten op in XML formaat.

Je ziet daarvan een voorbeeld met volgende stappen

- Je wijzigt de extensie van EenWordDocument.docx (bij deze cursus) van docx naar zip.
- Je opent in dit ZIP bestand de folder word
- Je opent in die folder document.xml
- Je ziet onder andere de regel van de eerste paragraaf van het document
(Dit is een eerste paragraaf)
en de regel van de tweede paragraaf van het document
(Dit is een tweede paragraaf).

Alle programmeertalen hebben libraries om XML te verwerken.

Je kan dus met je programmacode Office documenten maken, lezen, wijzigen ...

1.5.2 Data uitwisselen op het internet

Veel data wordt op het internet uitgewisseld in XML formaat.

Je kan met alle programmeertalen een applicatie schrijven om die data te verwerken.

Een voorbeeld is de koers van vreemde munten ten opzichte van de Euro.

Je ziet deze data als je met een browser volgende URL opent

<https://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml>

Je ziet in de regel die begint met <Cube currency="USD" een getal met de \$ koers.

Je ziet in de regel die begint met <Cube currency="GBP" een getal met de £ koers.

1.5.3 Configuratiebestanden

- Elke Java web applicatie kan een bestand `web.xml` hebben.
Dit bevat de configuratie van de applicatie.
Deze configuratie bevat bijvoorbeeld of de applicatie al of niet cookies gebruikt.
- Elke .net web applicatie bevat een bestand `web.config` dat XML bevat.
Dit bevat de configuratie van de applicatie.
Deze configuratie bevat bijvoorbeeld of de applicatie al of niet cookies gebruikt.
- ...

1.5.4 SVG

SVG (Scalable Vector Graphics) is een documentformaat dat een afbeelding voorstelt als een verzameling punten, lijnen en krommen. Deze verzameling is beschreven in XML.

Je kan met je browser met de toetsencombinatie `Ctrl O` het bestand `Fuji.svg` openen.

Als je blijft inzoomen op de afbeelding met de toetsencombinatie `Ctrl +` wordt de tekening nooit rafelig, omdat die op iedere keer hertekend wordt als een verzameling punten, lijnen en krommen.

Als je blijft inzoomen op de afbeelding `Fuji.png` wordt de tekening wel rafelig, want het PNG formaat slaat de tekening op als pixels. Ook de formaten BMP, GIF en JPEG doen dit.

Alle programmeertalen hebben libraries om XML te verwerken.

Je kan dus met je programmacode SVG afbeeldingen aanmaken.

2 Editors

Een XML document is een tekstdocument.

Je kan een XML document dus openen met elke tekst editor, zelfs met de eenvoudige Notepad.

Notepad heeft veel beperkingen

- ➔ Notepad gebruikt geen kleuren om de XML syntax te verduidelijken
- ➔ Notepad kan een XML document niet indenteren (per niveau dieper inspringen)
- ➔ Notepad kan niet controleren of een XML document well formed is

Er bestaan specifieke editors om XML documenten te verwerken

- XMLSpy Je kunt gratis een 30 dagen trial versie downloaden.
- Stylus Studio Je kunt gratis een 15 dagen trial versie downloaden.
- XRay XML Editor Dit is een gratis product.
- ...

Je hebt als ontwikkelaar niet zo'n specifieke XML editor nodig.

De IDE's waarmee je applicaties ontwikkelt, zijn ook goede XML editors.

We leggen hier uit hoe je volgende IDE's gebruikt

- NetBeans (Java en PHP ontwikkeling)
- Visual Studio (.net ontwikkeling)

In het vervolg van de cursus kan je handelingen op een XML document met één van deze IDE's doen.

2.1 NetBeans

2.1.1 Een XML document inzien

- Je kiest in het menu File de opdracht Open File
- Je duidt het XML document aan dat je wil inzien. Je kiest als voorbeeld België.xml.

2.1.2 Een XML document indenteren

- Je kiest in het menu Source de opdracht Format

2.1.3 Controleren of een XML document well formed is

- Je klikt met de rechtermuisknop in het document en je kiest Check XML.
- Je ziet in het venster Output – XML check (onder in NetBeans) enkele boodschappen. Geen enkele is een foutboodschap. Het document is dan well formed.

Je wijzigt in de eerste regel 1.0 naar 6.0 (een onbestaande versie van XML).

- Je klikt met de rechtermuisknop in het XML document en je kiest Check XML.
- Je ziet in het venster Output – XML check nu ook een foutboodschap:
XML version "6.0" is not supported, only XML 1.0 is supported. [1]
 Het document is niet well formed. Als je de foutboodschap aanklikt, plaatst NetBeans de cursor in de regel met de fout en toont die regel met een rode achtergrond.

Je corrigeert in de eerste regel 6.0 terug naar 1.0

- Je klikt met de rechtermuisknop in het XML document en je kiest Check XML.
- Je ziet in het venster Output – XML check geen foutboodschap meer.

2.2 Visual Studio

2.2.1 Een XML document inzien

- Je kiest in het menu FILE de opdracht Open, File
- Je duidt het XML document aan dat je wil inzien. Je kiest als voorbeeld België.xml.

2.2.2 Een XML document indenteren

- Je kiest in het menu EDIT de opdracht Advanced, Format Document

2.2.3 Controleren of een XML document well formed is

Een well formed XML document bevat geen enkel stukje regel dat rood golvend onderlijnd is. Dit is het geval in België.xml

Je wijzigt in de eerste regel 1.0 naar 6.0 (een onbestaande versie van XML).

6.0 wordt rood golvend onderlijnd. Het document is niet well formed.

Als je de muisaanwijzer laat rusten op 6.0, zie een foutmelding.

Je corrigeert in de eerste regel 6.0 terug naar 1.0.

De rode golvende onderlijning verdwijnt. Het document is well formed.

3 Structuur van een XML document

De belangrijkste onderdelen van een XML document zijn

- XML declaration
- commentaar
- elementen.

Deze bevatten de data van het document (zie volgend hoofdstuk)

3.1 XML declaration

De XML declaratie bevat

- De versie van de gebruikte XML. Er bestaan twee versies: 1.0 en 1.1
De versie 1.1 verduidelijkt enkele minimale onduidelijkheden in de versie 1.0.
Weinig tools en programmeertalen ondersteunen de versie 1.1
In de praktijk gebruikt men daarom versie 1.0.
- Met welke codering de tekens in de data zijn voorgesteld.
De meest gebruikte codering is UTF-8. Deze codering kan elk menselijk teken voorstellen.
UTF-8 gebruikt zoveel mogelijk één byte om één menselijk teken voor te stellen.
Dit is het geval bij tekens uit de Westerse cultuur. UTF-8 gebruikt meerdere bytes om één menselijk teken voor te stellen bij tekens die niet tot de Westerse cultuur behoren.
Naast UTF-8 bestaat ook UTF-16. UTF-16 gebruikt minstens twee bytes om één menselijk teken voor te stellen. UTF-16 is pas interessant als je data veel niet-Westerse tekens bevat.

Volgende regel is de XML declaration

```
<?xml version="1.0" encoding="utf-8" ?>
```

- Het XML document moet met deze regel beginnen: er mogen geen regels voor komen.
- Er mogen geen spaties of tab tekens komen voor het teken <
- De regel is hoofdlettergevoelig, behalve de tekst utf-8
- Het onderdeel version is verplicht en moet voor het onderdeel encoding komen.
- Als je het onderdeel encoding weglaat, schatten tools en programmeertalen dit onderdeel in door de eerste bytes van het bestand te lezen. Gezien deze inschatting soms verkeerd is, wordt aangeraden het onderdeel encoding altijd te vermelden.
- Als je de regel met de XML declaration weglaat, wordt XML versie 1.0 verondersteld, en wordt de encoding 'ingeschat' zoals eerder vermeld.
Er wordt aangeraden altijd de regel met de XML declaratie te vermelden.

3.2 Commentaar

Je tikt commentaar tussen <!-- en -->

Je tikt deze commentaar ten dienste van een mens die het XML document inkijkt.

Tools en programmeertalen interpreteren de tekst in de commentaar niet.

Je mag commentaar over meerdere regels tikken.

Een XML document kan meerdere commentaar onderdelen bevatten.

Voorbeeld:

```
<!--  
Dit XML document bevat  
  pizza's en  
  de ingrediënten van die pizza's  
-->
```



Declaration en commentaar: zie takenbundel

4 Elementen met data tussen begintag en eindtag

Elementen bevatten data. Dit kan op twee manieren.

- De data bevindt zich tussen de begintag en de eindtag van het element
- De data bevindt zich in attributen in de begintag. Je leert dit in het volgende hoofdstuk

Je kan beide manieren mengen. Je leert dit in nog eens een volgend hoofdstuk

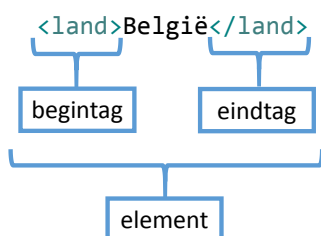
4.1 Algemeen

NaamVanEenLand.xml bevat één stukje data: de naam van een land

```
<?xml version="1.0" encoding="UTF-8" ?>
<land>België</land>
```

De data (België) is omsloten door <land> (dit heet de begintag) en </land> (dit heet eindtag)

- De begintag bevat tussen < en > een omschrijving van de data.
- De eindtag is verplicht en moet tussen </ en > dezelfde omschrijving bevatten als in de begintag. Deze gelijkheid is hoofdlettergevoelig.



- Als men spreekt over een element, bedoelt men de begintag tot en met de eindtag
- Als men spreekt over de naam van een element, bedoelt men de omschrijving in de begintag. De naam van het element is dus land
- In XML kies je zelf de namen van elementen. In HTML zijn die namen gedefinieerd in de standaard.

Er is geen standaard voor de namen van elementen. Meest gebruikt:

- namen helemaal in kleine letters (land, oppervlakte, ...)
- namen met het eerste woord in kleine letters en de daarop volgende woorden met de eerste letter als hoofdletter (aantalKinderen, gehuwd, ...)

4.2 Geneste elementen

Als je *meerdere* stukken data wil opnemen binnen de begintag en eindtag van een element, tik je tussen de begintag en de eindtag van dat element geneste elementen met die data.

DrieStukkenData.xml bevat de naam, de hoofdstad en de oppervlakte van een land

```
<?xml version="1.0" encoding="utf-8"?>
<land>
  <naam>België</naam>
  <hoofdstad>Brussel</hoofdstad>
  <oppervlakte>30528</oppervlakte>
</land>
```

Je moet de geneste elementen afsluiten vooraleer je het bovenliggend element afsluit.

Volgend voorbeeld is dus verkeerd, omdat het er geen eindtag </oppervlakte> is.

```
<?xml version="1.0" encoding="utf-8"?>
<land>
  <naam>België</naam>
  <hoofdstad>Brussel</hoofdstad>
  <oppervlakte>30528
</land>
```



Daltons: zie takenbundel

Je mag nesten op meerdere niveaus, zoals in BelgiëBrussel.xml

```
<?xml version="1.0" encoding="utf-8"?>
<land>
  <naam>België</naam>
  <hoofdstad>
    <naam>Brussel</naam>
    <oppervlakte>32.61</oppervlakte>
  </hoofdstad>
  <oppervlakte>30528</oppervlakte>
</land>
```

4.3 Child, Parent, Sibling, Root

Je kan elementen aanduiden als child elementen, parent element, sibling elementen en root element. We leggen dit uit aan de hand van BelgiëBrussel.xml

4.3.1 Child elementen

Geneste elementen noemt men *child* elementen van het element waarin ze direct genest zijn.

- Het element naam (met België) is child van het element land
- Het element hoofdstad is child van het element land.
- Het element naam (met Brussel) is child van het element hoofdstad maar niet van het element land, want er is een tussenliggend element hoofdstad
- Het element oppervlakte (met 32.61) is child van het element hoofdstad
- Het element oppervlakte (met 30528) is child van het element land

4.3.2 Parent element

Het element waarin de child elementen direct genest zijn noemt men het *parent* element van die child elementen.

- Het element land is het parent element van de elementen naam (met data België), hoofdstad en oppervlakte (met 30528).
- Het element hoofdstad is het parent element van de elementen naam (met data Brussel) en oppervlakte (met 32.61)

4.3.3 Siblings

Elementen met eenzelfde parent element noemt men siblings

- De elementen naam (met België), hoofdstad en oppervlakte (met 30528) zijn siblings
- De elementen naam (met Brussel) en oppervlakte (met 32.61) zijn siblings

4.3.4 Root element

Een well formed XML document moet één root element hebben.

Dit is een element dat zelf geen child element is: het element land.

BelgiëNederlandLuxemburg.xml is niet well formed: het heeft drie root elementen.

```
<?xml version="1.0" encoding="utf-8"?>
<land>België</land>
<land>Nederland</land>
<land>Luxemburg</land>
```

Met een kleine ingreep wordt dit well formed, met één root element benelux, zoals in benelux.xml

```
<?xml version="1.0" encoding="utf-8"?>
<benelux>
  <land>België</land>
  <land>Nederland</land>
  <land>Luxemburg</land>
</benelux>
```

4.3.5 Mixed content

Bij mixed content heeft een element tussen zijn begintag en eindtag zowel tekst als child elementen.

Je ziet een voorbeeld in `PacManMetMixedContent.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<pacman>
  De speler bestuurt en geel "happertje" genaamd Pac-man, waarmee hij zich
  een weg moet banen door een speelveld gevuld met muntjes
  <producent>Namco</producent>
</pacman>
```

De tekst kan zich *voor* de child elementen (zoals hier), *tussen* de child elementen en/of *na* de child elementen bevinden.

Mixed content wordt afgeraden, omdat de tekst `De speler ... muntjes` geen eigen omschrijving heeft met een begintag en eindtag.

`PacMan.xml` bevat een oplossing: de tekst bevindt zich in een extra child element `spelverloop`

```
<?xml version="1.0" encoding="utf-8"?>
<pacman>
  <spelverloop>
    De speler bestuurt en geel "happertje" genaamd Pac-man, waarmee hij zich
    een weg moet banen door een speelveld gevuld met muntjes.
  </spelverloop>
  <producent>Namco</producent>
</pacman>
```

4.3.6 Meerdere child elementen met soortgelijke informatie

Meerdere child elementen kunnen *soortgelijke* informatie bevatten.

Je kan dit op twee manieren uitdrukken.

- de child elementen rechtstreeks in het parent element opnemen.
In `TalenInBelgië1.xml` bevatten de child elementen `taal` soortgelijke informatie. Deze child elementen bevinden zich rechtstreeks in het element `land`

```
<?xml version="1.0" encoding="utf-8"?>
<land>
  <naam>België</naam>
  <taal>Nederlands</taal>
  <taal>Frans</taal>
  <taal>Duits</taal>
</land>
```
- deze child elementen in een extra child element van het parent element.
Ook in `TalenInBelgië2.xml` bevatten alle child elementen `taal` soortgelijke informatie. Deze child elementen bevinden zich in een extra child element `talen` in het element `land`

```
<?xml version="1.0" encoding="utf-8"?>
<land>
  <naam>België</naam>
  <talen>
    <taal>Nederlands</taal>
    <taal>Frans</taal>
    <taal>Duits</taal>
  </talen>
</land>
```

De meeste informatici geven de voorkeur aan de tweede manier, wegens de duidelijkere *structuur*.



CD verzameling: zie takenbundel

5 Elementen met data in attributen

5.1.1 Algemeen

BelgiëAttributen1.xml bevat drie stukken data: naam, hoofdstad en oppervlakte van een land:

```
<?xml version="1.0" encoding="utf-8"?>
<land naam="België" hoofdstad="Brussel" oppervlakte="30528"></land>
```

Een begintag kan een variabel aantal attributen bevatten. Elk attribuut bevat één stuk data.

Een attribuut heeft een omschrijving van de data, daarna =, daarna de data zelf tussen quotes.

Elke onderdeel met gele achtergrond is één attribuut:

```
<land naam="België" hoofdstad="Brussel" oppervlakte="30528"></land>
```

Elk attribuut in een element moet een unieke naam hebben. Volgend voorbeeld is dus verkeerd:

```
<persoon familienaam="Desmet" voornaam="Bart" voornaam="Jan" voornaam="Frans"/>
```

5.1.2 Een element zonder eindtag

Je kan de eindtag weglaten, als een element geen child elementen en geen tekst tussen de begintag en eindtag bevat. Je sluit de begintag dan met />

Je ziet dit in BelgiëAttributen2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<land naam="België" hoofdstad="Brussel" oppervlakte="30528"/>
```

5.1.3 Dubbele of enkele quotes

Je kan de data van een attribuut omsluiten met dubbele of enkele aanhalingstekens.

Dubbele worden aangeraden, omdat de data zelf soms enkele bevat:

AttributenDubbeleQuotes.xml gebruikt dubbele quotes en is well formed

```
<?xml version="1.0" encoding="utf-8"?>
<gemeente naam=" 's-Hertogenbosch" />
```

AttributenEnkeleQuotes.xml gebruikt enkele quotes en is *niet* well formed

```
<?xml version="1.0" encoding="utf-8"?>
<gemeente naam=' 's-Hertogenbosch' />
```

XML interpreteert de tweede regel als een attribuut naam met een lege inhoud (' ')

en een attribuut s-Hertogenbosch zonder inhoud, gevolgd door een ' die er niet mag staan.

5.2 Root element

Ook bij gebruik van attributen geldt nog de regel dat er *één* root element moet zijn.

BeneLuxNietWellFormed.xml is daarom *niet* well formed

```
<?xml version="1.0" encoding="utf-8"?>
<land naam="België" hoofdstad="Brussel" oppervlakte="30528"/>
<land naam="Nederland" hoofdstad="Amsterdam" oppervlakte="41543"/>
<land naam="Luxemburg" hoofdstad="Luxemburg" oppervlakte="2586"/>
```

Met een kleine ingreep wordt dit well formed XML, zoals in BeneLuxWellFormed.xml

```
<?xml version="1.0" encoding="utf-8"?>
<beneLux>
  <land naam="België" hoofdstad="Brussel" oppervlakte="30528"/>
  <land naam="Nederland" hoofdstad="Amsterdam" oppervlakte="41543"/>
  <land naam="Luxemburg" hoofdstad="Luxemburg" oppervlakte="2586"/>
</beneLux>
```



CD verzameling 2: zie takenbundel

6 Elementen met data in attributen én in child elementen

Een element kan én attributen én child elementen hebben.

Je ziet dit in BeneluxGemengd.xml in het element benelux

```
<?xml version="1.0" encoding="utf-8"?>
<benelux jaarOprichting="1944">
  <plaatsOprichting>Londen</plaatsOprichting>
  <landen>
    <land>België</land>
    <land>Nederland</land>
    <land>Luxemburg</land>
  </landen>
</benelux>
```

De XML standaard heeft geen regels of je data opneemt in attributen of in child elementen.

Er is zeker één vuistregel die je altijd respecteert: als data kan opgesplitst worden in kleinere stukken data, plaats je die data *niet* in één attribuut of één tekst tussen een begin en eindtag, maar in child elementen of in meerdere attributen.

OlivierGerardEenAttribuut.xml voldoet niet aan die vuistregel

```
<?xml version="1.0" encoding="utf-8"?>
<persoon naam="Olivier Gerard"/>
```

Je kan een naam opsplitsen in een voornaam en een familienaam.

Hier bevindt de naam zich in één attribuut.

Deze structuur geeft niet aan of de voornaam Olivier of Gerard is.

In OlivierGerardChildElementen.xml geeft de structuur wel aan dat de voornaam Olivier is

```
<?xml version="1.0" encoding="utf-8"?>
<persoon>
  <voornaam>Olivier</voornaam>
  <familienaam>Gerard</familienaam>
</persoon>
```

In OlivierGerardMeerdereAttributen.xml geeft de structuur dit ook aan

```
<?xml version="1.0" encoding="utf-8"?>
<persoon voornaam="Olivier" familienaam="Gerard"/>
```

De tweede oplossing met meerdere attributen gebruikt minder bytes (89) dan de eerste oplossing met child elementen (126). Dit is een besparing met 30 %.

JanWillemLincolnEenAttribuut.xml voldoet niet aan de vuistregel

```
<?xml version="1.0" encoding="utf-8"?>
<persoon voornamen="Jan Willem Lincoln" familienaam="van de Wetering"/>
```

De structuur geeft niet aan of deze persoon drie voornamen heeft (Jan, Willem en Lincoln) of twee voornamen (Jan Willem en Lincoln).

In JanWillemLincolnChildElementen.xml geeft de structuur dit wel aan

```
<?xml version="1.0" encoding="utf-8"?>
<persoon familienaam="van de Wetering">
  <voornamen>
    <voornaam>Jan Willem</voornaam>
    <voornaam>Lincoln</voornaam>
  </voornamen>
</persoon>
```


In OlivierGerardMeerdereAttributen.xml geeft de structuur dit ook aan

```
<?xml version="1.0" encoding="utf-8"?>  
<persoon familienaam="van de Wetering" voornaam="Jan Willem" voornaam2="Lincoln"/>
```

De meeste informatici vermijden deze oplossing: ze geeft niet zo duidelijk aan dat de attributen voornaam en voornaam2 behoren tot één en dezelfde structuur (voornamen). De eerste oplossing geeft dit wel duidelijk aan.



Quatre quarts: zie takenbundel

7 XML document als boomstructuur

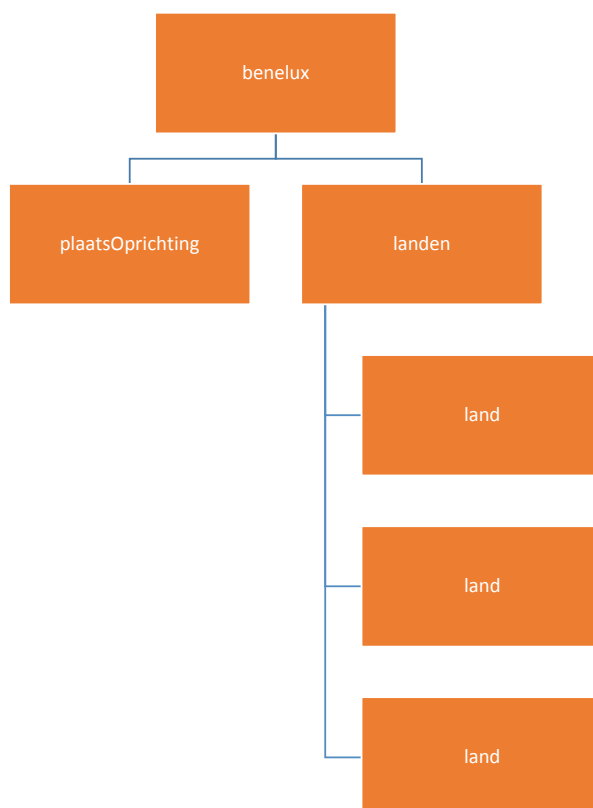
Elk XML document is een boomstructuur.

De boom begint bij het root element en vertakt zich in child elementen.

Je kan `BeneLuxGemengd.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<benelux jaarOprichting="1944">
  <plaatsOprichting>Londen</plaatsOprichting>
  <landen>
    <land>België</land>
    <land>Nederland</land>
    <land>Luxemburg</land>
  </landen>
</benelux>
```

visualiseren met volgende boomstructuur (de boom 'staat op zijn kop' met de wortel bovenaan)



Boomstructuur: zie takenbundel

8 Speciale tekens

Je mag de tekens <, & en > niet letterlijk tikken in tekst tussen een begintag en een eindtag.

Er bestaan hiervoor twee oplossingen.

- Het teken vervangen door de bijbehorende entity
- CDATA

8.1 Het teken vervangen door de bijbehorende entity

Een entity is een tekenreeks die één teken vervangt dat je niet letterlijk mag tikken.

teken dat je niet letterlijk mag tikken	vervangende entity
<	<
&	&
>	>

Film.xml bevat een voorbeeld: we willen de film Wallace & Grommit voorstellen

```
<?xml version="1.0" encoding="utf-8"?>
<film>Wallace &amp; Grommit</film>
```

XML data, waarin < en/of & en/of > veel voorkomen, worden onleesbaar voor een mens.

Voorbeeld: VergelijkingsoperatorenMetEntities.xml bevat volgende menselijke tekst

Je kan in een programmeertaal met de operatoren ==, !=, <, <=, > en >= twee waarden vergelijken.

```
<?xml version="1.0" encoding="utf-8"?>
<vergelijkingsoperatoren>
```

Je kan in een programmeertaal met de operatoren ==, !=, <, <=, > en >= twee waarden vergelijken.

```
</vergelijkingsoperatoren>
```

Je lost dit hier onder op met CDATA.

8.2 CDATA

Als je de tekst voorafgaat door <![CDATA[en afsluit met]]>, mag je in de tekst de tekens <, & en > wel letterlijk tikken.

Je ziet dit in VergelijkingsoperatorenMetCDATA.xml

```
<?xml version="1.0" encoding="utf-8"?>
<vergelijkingsoperatoren>
<![CDATA[
```

Je kan in een programmeertaal met de operatoren ==, !=, <, <=, > en >= twee waarden vergelijken.

```
]]>
</vergelijkingsoperatoren>
```

9 Samenvatting well formed

Je vindt hier onder nog eens de regels voor een well formed XML document

- het document begint met de XML declaration
- het document heeft één root element
- start tags hebben bijbehorende eindtags, tenzij de start tag eindigt met />
- tag namen zijn hoofdlettergevoelig
- elementen moeten correct genest zijn
- de waarden van attributen zijn omsloten met aanhalingstekens
- speciale karakters moeten als entities of met CDATA voorgesteld worden



Tip: iedere keer je een XML bestand afgewerkt hebt, controleer je het best of het well formed is met de controle mogelijkheden in NetBeans of Visual Studio

10 XPath

10.1 Algemeen

XPath is een taal waarmee je een zoekopdracht schrijft.

Je zoekt met die opdracht data in een XML document.

Zo'n zoekopdracht heet ook een XPath expressie.

XPath is ook een standaard van het World Wide Web Consortium.

Je gebruikt een XPath expressie normaal in je Java, PHP of C# code.

Je tikt in deze cursus XPath expressies in NetBeans of Visual Studio en voert deze uit, om XPath te leren kennen. Je moet daartoe een uitbreiding op NetBeans of Visual Studio installeren.

10.2 Installatie en gebruik van de XPathUtil plugin in NetBeans

Je installeert met volgende stappen een plugin in NetBeans

waarmee je XPath zoekopdrachten kan intikken en uitvoeren vanuit NetBeans

1. Je surft met een browser naar <http://plugins.netbeans.org/plugin/18522/xpathutil>
2. Je kiest de knop Download
3. Je kiest in NetBeans in het menu Tools de opdracht Plugins
4. Je kiest het tabblad Downloaded
5. Je kiest de knop Add Plugins
6. Je kiest het bestand dat je bij punt 2 op je computer plaatste
7. Je kiest de knop Install, Next, I accept ..., Install, Continue, Finish, Close

Je leert nu hoe je met de plugin een eerste XPath expressie intikt en uitvoert.

Je leert de syntax van de expressie verder in de cursus kennen.

1. Je opent België.xml
2. Je kiest in het menu Window de opdracht XPath
3. Je tikt bij Expression volgende XPath expressie: //hoofdstad
4. Je ziet daaronder het resultaat van deze expressie: het element hoofdstad



Opmerking: de plugin toont attributen van elementen in alfabetische volgorde, niet in de volgorde zoals ze in het XML document voorkomen.

10.3 Installatie en gebruik van de XPath Navigator extension in Visual Studio

Je installeert met volgende stappen een extension in Visual Studio

waarmee je XPath zoekopdrachten kan intikken en uitvoeren vanuit Visual Studio

1. Je kiest in Visual Studio in het menu TOOLS de opdracht Extensions and Updates
2. Je kiest links Online
3. Je tikt xpath navigator in het tekstvak rechts boven en je wacht enkele seconden
4. Je kiest Download naast XPath Navigator
5. Je kiest de knop Install, Restart now

Je leert nu hoe je met de extension een eerste XPath zoekopdracht intikt en uitvoert.

Je leert de syntax van de zoekopdracht verder in de cursus kennen.

1. Je opent België.xml
2. Je kiest in het menu VIEW de opdracht Toolbars, XML Editor
3. Je tikt in de toolbar bij XPath: volgende XPath expressie: //hoofdstad
4. Je ziet het resultaat van deze expressie: het element hoofdstad krijgt een roze achtergrond
Je ziet het resultaat van de zoekopdracht ook in het venster Output.

10.4 Slashes

Een XPath expressie kan beginnen met één slash, twee slashes of zonder slash.

Je opent BelgiëBrussel.xml om de uitgelegde XPath expressies uit te proberen.

10.4.1 Een XPath expressie die begint met één slash

Als een XPath expressie begint met één slash, moet na die slash het root element komen.

XPath zoekt dan data ten opzichte van dit root element.

- De XPath expressie kan na dit root element een slash bevatten gevolgd door de naam van een child element dat je zoekt.
- De XPath expressie kan na de naam van dit child element terug een slash bevatten, gevolgd door de naam van een child element (van het eerste child element in de expressie) dat je zoekt.
- Als de XPath expressie enkel een slash bevat, krijg je alle elementen van het document.

Voorbeelden:

XPath expressie	Resultaat van de expressie
/land	<land>...</land>
/	<land>...</land>
/land/oppervlakte	<oppervlakte>30528</oppervlakte>
/land/hoofdstad/naam	<naam>Brussel</naam>

10.4.2 Een XPath expressie die begint met twee slashes

Als een XPath expressie begint met twee slashes doorzoekt XPath het volledige document naar het element (of de elementen) vermeld in de expressie. Voorbeelden:

XPath expressie	Resultaat van de expressie
//hoofdstad	<hoofdstad>...</hoofdstad>
//hoofdstad/naam	<naam>Brussel</naam>
//naam	<naam>België</naam> <naam>Brussel</naam>

10.4.3 Een XPath expressie die begint zonder slash

Als een XPath expressie niet begint met een slash, doorzoekt XPath het document vanaf het *huidig* element. Je kan zo'n expressie niet uittesten in NetBeans of Visual Studio, omdat je geen *huidig* element kan aanduiden vooraleer je de XPath expressie uitvoert.

Je kan zo'n expressie wel gebruiken in je Java, PHP of C# code. Voorbeelden:

XPath expressie	Huidig element	Resultaat van de expressie
hoofdstad	land	<hoofdstad>...</hoofdstad>
naam	hoofdstad	<naam>Brussel</naam>

10.5 Ster

Een ster betekent "alle elementen op dit niveau".

XPath expressie	Resultaat van de expressie
/land/hoofdstad/*	<naam>Brussel</naam> <oppervlakte>32.61</oppervlakte>

10.6 Twee punten

Twee punten betekenen "het parent element"

XPath expressie	Resultaat van de expressie
//hoofdstad/..	<land naam="België" .../> ... <land>

10.7 Text node zoeken

Een text node is de tekst tussen een begintag en een eindtag.

Je duidt een text node in een XPath expressie aan met de XPath functie `text()`

Je kan zo'n expressie niet uittesten in Visual Studio, wel in NetBeans. Voorbeelden:

XPath expressie	Resultaat van de expressie
<code>/land/naam/text()</code>	België
<code>//hoofdstad/naam/text()</code>	Brussel

10.8 Attribuut waarde zoeken

Je duidt een attribuut in een XPath expressie aan met het teken `@` gevolgd door de naam van het attribuut. Voorbeelden in `OlivierGerardMeerdereAttributen.xml`

XPath expressie	Resultaat van de expressie
<code>/persoon/@voornaam</code>	Olivier
<code>//@familienaam</code>	Gerard

10.9 Elementen zoeken op hun volgnummer ten opzichte van hun parent

Je kan in een XPath expressie een volgnummer (nummering vanaf 1) tikken tussen vierkante haakjes. XPath zoekt dan de elementen die het zoveelste child zijn van hun parent.

Je kan tussen vierkante haakjes ook de functie `last()` vermelden.

XPath zoekt dan het element dat het laatste child is van zijn parent.

Je kan `last()` combineren met een berekening. `last() - 1` duidt bvb. het voorlaatste element aan.

Voorbeelden in `BeneLux.xml`:

XPath expressie	Resultaat van de expressie
<code>//land[2]</code>	<code><land>Nederland</land></code>
<code>//land[last()]</code>	<code><land>Luxemburg</land></code>
<code>//land[last()-1]</code>	<code><land>Nederland</land></code>

10.10 Elementen zoeken op attribuut waarden

Je kan in een XPath expressie na een element tussen vierkante haakjes een boolean expressie tikken waarin je een attribuut van dit element vergelijkt met een waarde.

XPath zoekt dan de elementen waarvan die boolean expressie `true` teruggeeft.

Voorbeelden in `BeneLuxWellFormed.xml`

XPath expressie	Resultaat van de expressie
<code>//land[@naam="Nederland"]</code>	<code><land naam="Nederland" .../></code>
<code>//land[@naam!="Nederland"]</code>	<code><land naam="België" .../></code> <code><land naam="Luxemburg" .../></code>
<code>//land[@naam=@hoofdstad]</code>	<code><land naam="Luxemburg" .../></code>
<code>//land[@oppervlakte>10000]</code>	<code><land naam="België" .../></code> <code><land naam="Nederland" .../></code>

Je kan in de boolean expressie (die je vermeldt tussen de vierkante haakjes) ook de wiskundige operatoren `+`, `-`, `*`, `div` (delen) en `mod` (restbepaling) gebruiken.

Voorbeeld in `Rechthoeken.xml`: de rechthoeken met een oppervlakte vanaf 10:

XPath expressie	Resultaat van de expressie
<code>//rechthoek[@lengte*@breedte>=10]</code>	<code><rechthoek lengte="5" breedte="2"/></code> <code><rechthoek lengte="6" breedte="3"/></code>

Je kan ook de logische operatoren and, or en not() gebruiken

XPath expressie	Resultaat van de expressie
<code>//rechthoek[@lengte>=4 and @breedte=2]</code>	<code><rechthoek lengte="5" breedte="2"/></code> <code><rechthoek lengte="4" breedte="2"/></code>

Je kan een element zoeken op basis van een attribuut in een *child* element van het te zoeken element. Je vermeldt het child element tussen vierkante haakjes, gevolgd door nog eens vierkante haakjes en de boolean expressie waarin je het attribuut vergelijkt met een waarde. Voorbeeld in BelgiëNederland.xml

XPath expressie	Resultaat van de expressie
<code>//land[hoofdstad[@naam="Brussel"]]</code>	<code><land naam="België" .../> ... </land></code>

Je kan van een element dat je zoekt op basis van een attribuut niet dit element zelf, maar een *child* element van dit element opzoeken. Voorbeeld:

XPath expressie	Resultaat van de expressie
<code>//land[@naam="België"]/hoofdstad</code>	<code><hoofdstad naam="Brussel" ... /></code>

10.11 Elementen zoeken op het al of niet aanwezig zijn van attributen

Voorbeelden in Sporters.xml

XPath expressie met daaronder de betekenis	Resultaat van de expressie
<code>//sporter[@bijnaam]</code> sporter elementen die een attribuut bijnaam hebben	<code><sporter ... bijnaam="De adelaar uit de Andes"/></code> <code><sporter ... bijnaam="The animal"/></code>
<code>//sporter[not(@bijnaam)]</code> sporter elementen die geen attribuut bijnaam hebben	<code><sporter voornaam="Jean" .../></code> <code><sporter voornaam="Fons" .../></code> <code><sporter/></code>
<code>//sporter[*]</code> sporter elementen die om het even welk attribuut hebben	<code><sporter voornaam="Luis" .../></code> <code><sporter voornaam="Jean" .../></code> <code><sporter voornaam="Ellen" .../></code> <code><sporter voornaam="Fons" .../></code>
<code>//sporter[not(*)]</code> sporter elementen die geen enkel attribuut hebben	<code><sporter/></code>

10.12 Elementen zoeken op text node waarden

Je kan in een XPath expressie na een element tussen vierkante haakjes een child element tikken dat je vergelijkt met een waarde. XPath vergelijkt de *text node* van dit child element met die waarde. XPath zoekt dan de elementen waarvan die boolean expressie true teruggeeft.

Voorbeelden in BelgiëNederlandElementen.xml:

XPath expressie	Resultaat van de expressie
<code>//land[hoofdstad="Brussel"]</code>	<code><land naam="België"/> ... </land></code>
<code>//land[oppervlakte>40000]</code>	<code><land naam="Nederland"/> ... </land></code>
<code>//land[oppervlakte>40000]/hoofdstad</code>	<code><hoofdstad>Amsterdam</hoofdstad></code>

10.13 Statistische functies

Je kan in een XPath expressie de statistische functies `count()` en `sum()` gebruiken.

Voorbeelden in `FrietBestelling.xml`

XPath expressie met daaronder de betekenis	Resultaat van de expressie
<code>//bestelling[count(product)=1]</code>	<pre><bestelling> <product naam="friet" prijs="3"/> </bestelling></pre>
<code>//bestelling[sum(product/@prijs)=2.5]</code>	<pre><bestelling> <product naam="friet" prijs="2"/> <product naam="saus" prijs="0.5"/> </bestelling></pre>



XPath: zie takenbundel

11 Namespaces

11.1 Algemeen

Sommige woorden hebben meerdere betekenissen. Voorbeelden:

- Account betekent bankrekening
maar betekent ook de informatie van een gebruiker op een computer
- Gerecht betekent voedsel maar ook rechtbank
- String betekent een stuk textiel maar ook een verzameling tekens in een computer

Je gebruikt in XML woorden als namen van elementen.

Je ziet in `BankAccounts.xml` het woord `account` gebruikt in zijn betekenis van bankrekening

```
<?xml version="1.0" encoding="utf-8"?>
<accounts>
  <account id="GR16011012500000000123006" firstname="Larry" lastname="Ellison"/>
  <account id="KU00000000000001234560101" firstname="Bill" lastname="Gates"/>
</accounts>
```

Je ziet in `ComputerAccounts.xml` het woord `account` gebruikt in zijn betekenis van informatie over een gebruiker op een computer

```
<?xml version="1.0" encoding="utf-8"?>
<accounts>
<account id="JB007007007007007007" firstname="James" lastname="Bond"/>
</accounts>
```

XML data uit verschillende bronnen (bijvoorbeeld van over het internet) wordt regelmatig gefusioneerd tot één XML document.

`Accounts.xml` is de fusie van `BankAccounts.xml` en `ComputerAccounts.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<accounts>
  <account id="GR16011012500000000123006" firstname="Larry" lastname="Ellison"/>
  <account id="KU00000000000001234560101" firstname="Bill" lastname="Gates"/>
  <account id="JB007007007007007007" firstname="James" lastname="Bond"/>
</accounts>
```

Als mens is het moeilijk hier het onderscheid te maken tussen de twee betekenissen van `account`.

Vanuit code is dit onderscheid nog moeilijker:

je kan geen XPath expressie verzinnen die enkel de bankrekeningen uit dit bestand leest.

Namespaces lossen dit probleem op.

11.2 URI's als namespaces

Om het onderscheid te maken tussen de twee betekenissen van het woord `account`

- associeer je het element `Account` in `ComputerAccounts.xml` met een unieke identificatie
- associeer je het element `Account` in `BankAccounts.xml` met een andere unieke identificatie

Je vermeldt deze unieke identificatie ook bij elk `Account` element in `Accounts.xml`.

Je zal zo in dit bestand het onderscheid kunnen maken tussen de twee betekenissen van "Account".

Zo'n unieke identificatie heet bij XML een namespace.

Het is belangrijk dat de namespace waarmee je `account` associeert met "bankrekening" verschilt van de namespace waarmee je `account` associeert met "informatie van een gebruiker op een computer".

Om dit te garanderen, over de wereld heen, begint een namespace met de URL van de firma of organisatie die de namespace maakt: elke firma of organisatie heeft een unieke URL.

De namespace bevat na die URL nog stukken tekst, zoals de naam van de afdeling binnen de firma of organisatie die de namespace maakt en/of de naam van de productfamilie die met de namespace te maken heeft.

De volledige namespace is een URI (Universal Resource Identifier).



Opmerking: URI (Universal Resource Identifier) is een identifier voor een stukje data.

Elke URL is een URI, maar niet elke URI is een URL.

Een URI is een URL als de URI een netwerkprotocol (zoals http) gebruikt.

file:///c:/data/mijngezin.jpg is een URI maar geen URL. file is geen netwerkprotocol, maar een protocol dat een bestand op de eigen PC aanduidt.

Voorbeelden van namespaces:

- <http://www.vdab.be/opleidingen/ict>
- <http://www.belgie.be/belastingsaangiften>

Belangrijk hierbij is:

- Namespaces zijn geen URL's: er hoeft geen webpagina te bestaan met een URL gelijk aan de namespace. Sommige firma's maken wel zo'n pagina.
- Je gebruikt namespaces in XML bestanden. De tools en libraries die deze XML bestanden verwerken proberen deze namespaces nooit als URL's op het internet te zoeken.

11.3 Praktisch

Je wijzigt BankAccounts.xml: je associeert de elementen met de namespace

<http://www.bestebank.be/sparen>

```
<?xml version="1.0" encoding="utf-8"?>
<accounts xmlns="http://www.bestebank.be/sparen">
  <account id="GR16011012500000000123006" firstname="Larry" lastname="Ellison"/>
  <account id="KU0000000000001234560101" firstname="Bill" lastname="Gates"/>
</accounts>
```

- (1) Je gebruikt het standaard attribuut xmlns om een namespace te associëren met het huidige element (accounts). Alle child elementen van dit element (in dit geval de account elementen) zijn dan ook met die namespace geassocieerd. Ook child elementen van die child elementen zijn geassocieerd met die namespace, ...

Je wijzigt op soortgelijke manier ComputerAccounts.xml:

je associeert de elementen met de namespace <http://www.computers.org/accounts>

```
<?xml version="1.0" encoding="utf-8"?>
<accounts xmlns="http://www.computers.org/accounts">
  <account id="JB007007007007007007" firstname="James" lastname="Bond"/>
</accounts>
```

Je wijzigt nu ook Accounts.xml.

Je hebt hier meer werk: je gebruikt *meerdere* namespaces in één XML bestand.

```
<?xml version="1.0" encoding="utf-8"?>
<accounts xmlns:bank="http://www.bestebank.be/sparen"
  xmlns:pc="http://www.computers.org/accounts">
  <bank:account id="GR16011012500000000123006" firstname="Larry"
    lastname="Ellison"/>
  <bank:account id="KU0000000000001234560101" firstname="Bill" lastname="Gates"/>
  <pc:account id="JB007007007007007007" firstname="James" lastname="Bond"/>
</accounts>
```

- (1) Je associeert de eerste namespace met een vrij te kiezen prefix (afkorting) die je verder in het XML bestand zal gebruiken.
Het element accounts zelf is door deze handeling niet met een namespace geassocieerd.
- (2) Je associeert de tweede namespace met een andere vrij te kiezen prefix.

- (3) Je tikt voor een element, dat je met de eerste namespace wil associëren, de prefix die je met die eerste namespace associeerde, gevolg door :.
Zo is het element met die namespace geassocieerd.
Dezelfde prefix wordt met hetzelfde doel gebruikt op de volgende regel.
- (4) Je tikt voor een element, dat je met de tweede namespace wil associëren, de prefix die je met die tweede namespace associeerde. Zo is het element met die namespace geassocieerd.



Opmerking: als je voor het element accounts de prefix bank zou tikken (bank:accounts) associeer je ook dit element met de namespace `http://www.bestebank.be/sparen`

Als mens kan je nu wel het onderscheid te maken tussen de twee betekenissen van account.

Vanuit code is dit onderscheid ook mogelijk.

Als je een XPath expressie uitvoert vanuit Java, PHP of c#, associeer je eerst een vrij te kiezen prefix met de namespace van de elementen die je met XPath opzoekt.

Je gebruikt die prefix daarna in je XPath expressie.

Als je de prefix tmp associeert met de namespace `http://www.bestebank.be/sparen`, zoek je met de XPath expressie `//tmp:account` de bankrekeningen in `Accounts.xml`.

De prefix die je in XPath met een namespace associeert hoeft dus niet dezelfde te zijn als de prefix die je in het XML document met die namespace associeert.

Je kan met de NetBeans plugin of de Visual Studio extension geen XPath expressie uitvoeren die met een namespace geassocieerd is.



Opmerking: als je een prefix tikt in een begintag én er is een bijbehorende eindtag, moet je in die eindtag dezelfde prefix tikken.

11.4 Default namespace

Een namespace die je met een element associeert zonder prefix (zoals in `BankAccounts.xml` en `ComputerAccounts.xml`) heet de default namespace.

Alle child elementen, waarvoor geen prefix staat, zijn automatisch met die namespace geassocieerd.



Namespaces: zie takenbundel

12 Regular expression

Sommige waarden uit de werkelijkheid moeten voldoen aan een bepaald patroon. Voorbeelden:

- een ISBN (Internationaal Standaard Boeknummer) is de cijfers 978 gevolgd door 10 cijfers
- een Visa kaart nummer begint met het cijfer 4 en daarna 13 of 16 cijfers

Een regular expression stelt een tekstpatroon voor. Volgende regular expression hoort bij een ISBN:

`^978\d{10}$` (uitleg verder in de cursus).

- 9789027439642 voldoet aan die regular expression
- 6669027439642 voldoet niet aan die regular expression.

Je kan in elke programmeertaal (Java, C#, PHP, Javascript, ...) én in XML controleren of een tekst voldoet aan een regular expression. Een regular expression heeft in al die talen dezelfde syntax.

Je leert in dit hoofdstuk de syntax van regular expressions. Je leert in het volgende hoofdstuk een XML waarde controleren ten opzichte van een regular expression.

12.1 Controleren of een stukje tekst *voorkomt* in een tekst

De meeste tekens in een regular expression hebben hun letterlijke betekenis.

Als je een tekst controleert ten opzichte van een regular expression, controleer je of de tekenreeks van de regular expression *voorkomt* in die tekst. Regular expressions zijn hoofdlettergevoelig.

regular expression	tekst	controle
pizza	pizza's zijn lekker	👍
pizza	frietjes zijn lekker	👎
pizza	een pizza voor mij	👍
pizza	Pizza for ever	👎

12.2 Regular expressions uitproberen

Je kan in de website <https://regex101.com> regular expressions uitproberen

The screenshot shows the regex101.com interface. In the 'REGULAR EXPRESSION' section, the pattern is `/ een / g`. In the 'TEST STRING' section, the text is `een landschap in een mooie morgen`. The 'EXPLANATION' section explains that `een` matches the characters `een` literally (case sensitive) and `g` is the global modifier. The test string shows two matches highlighted in light blue: `een` and `een`. A green 'MATCH' button is visible next to the first match.

- Je tikt de regular expression onder REGULAR EXPRESSION
- Je tikt de tekst onder TEST STRING
- Als de tekst klopt ten opzichte van de regular expression zie je naast REGULAR EXPRESSION een blokje met daarin onder andere **MATCH**
- Het stuk tekst dat klopt krijgt onder TEST STRING een lichtblauwe achtergrond
Als meerdere onderdelen in de tekst kloppen, toont de website standaard enkel het eerste onderdeel met een lichtblauwe achtergrond. Als je wil dat alle onderdelen een lichtblauwe achtergrond krijgen, tik je de optie **g** in het vakje met **?**
- Als de tekst *niet* klopt ten opzichte van de regular expression zie je naast REGULAR EXPRESSION een blokje met daarin onder andere **NO MATCH**
- Je ziet rechts, onder EXPLANATION, uitleg over je regular expression

Vanaf nu geven we in voorbeelden de tekst onderdelen die kloppen ook een lichtblauwe achtergrond

12.3 Het begin van de tekst

Het teken `^` heeft in een regular expression een speciale betekenis: het *begin* van de tekst

regular expression	tekst	controle
<code>^pizza</code>	pizza's zijn lekker	👍
<code>^pizza</code>	een pizza a.u.b.	👎

12.4 Het einde van de tekst

Het teken `$` heeft in een regular expression een speciale betekenis: het *einde* van de tekst

regular expression	tekst	controle
<code>pizza\$</code>	ik lust pizza	👍
<code>pizza\$</code>	een pizza a.u.b.	👎
<code>^pizza\$</code>	pizza	👍
<code>^pizza\$</code>	pizza's zijn lekker	👎

12.5 Speciale tekens hun letterlijke betekenis geven

`^` en `$` zijn voorbeelden van speciale tekens.

Als je die tekens letterlijk hun letterlijke betekenis wil geven, tik je er `\` voor.

regular expression	tekst	controle
<code>\\$</code>	veel \$ in mijn zak	👍
<code>\\$\\$</code>	dollar symbool: \$	👍

Je zal leren dat nog tekens een speciale betekenis hebben: `|`, `(`, `)`, `*`, `+`, `?`, `{`, `}`, `[`, `]`

Je geeft ook deze tekens hun letterlijke betekenis door er `\` voor te tikken

12.6 Een regular expression visualiseren

Je kan een regular expression tikken in de website <https://jex.im/regex>.

Je ziet een grafische voorstelling, die je helpt de regular expression te begrijpen.

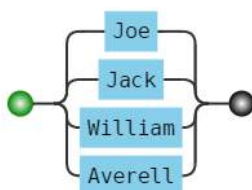


12.7 De or operator |

Het teken `|` is de *or* operator. Het betekent de tekst voor `|` of de tekst na `|`

regular expression	tekst	controle
<code>Joe Jack William Averell</code>	De leider van de Daltons is Joe	👍
<code>Joe Jack William Averell</code>	Averell heeft altijd honger	👍
<code>Joe Jack William Averell</code>	Pa Dalton is een kluizenkraker	👎

visualisatie bovenstaande expressie



12.8 Subexpressie tussen ()

Een subexpressie is een onderdeel van een regular expression tussen ronde haakjes.

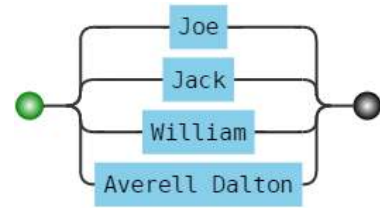
Een subexpressie laat toe voor of na een lijst met keuzes, gescheiden door | een tekst te tikken die *niet* tot deze keuzes behoort.

Je wil uitdrukken dat na Joe of Jack of William of Averell een spatie en Dalton moet komen.

Dit lukt niet met de regular expression

Joe|Jack|William|Averell Dalton

Deze betekent: Joe of Jack of William of Averell Dalton



Dit lukt wel met de regular expression

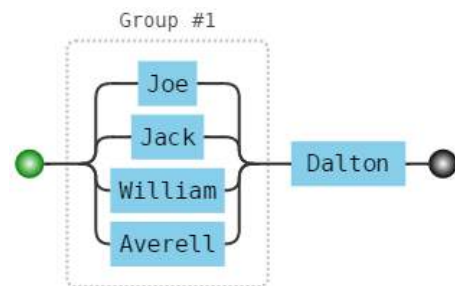
(Joe|Jack|William|Averell) Dalton

De | operator beperkt de keuzes nu

tot de waarden tussen de ronde haakjes.

De spatie en Dalton er na wordt aanzien

als een vervolg van de regular expression



regular expression	tekst	controle
(Joe Jack William Averell) Dalton	Joe Dalton is de leider	👍
(Joe Jack William Averell) Dalton	Averell Dalton heeft honger	👍
(Joe Jack William Averell) Dalton	Pa Dalton is een kluizenkraker	👎

12.9 Om het even welk teken .

Het teken . heeft een speciale betekenis: om het even welk teken (behalve een enter)

regular expression	tekst	controle
.Daltons	Daar komen de Daltons	👍
.Daltons	Daltons zijn gevaarlijk	👎

visualisatie bovenstaande expressie:



12.10 Herhalingsoperatoren

Een herhalingsoperator betekent dat het teken of de subexpressie *voor* die operator herhaald wordt.

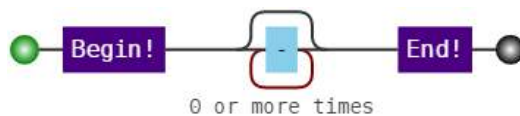
12.10.1 Nul of meerdere keren *

Het teken * betekent dat het teken of de subexpressie voor * nul of meerdere keren herhaald wordt.

Volgende regular expression stelt een streepjeslijn voor die ook leeg kan zijn

regular expression	tekst	controle
^_*\$	-----	👍
^_*\$		👍
^_*\$	- :)	👎

visualisatie bovenstaande expressie



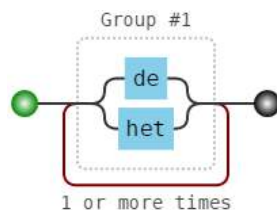
12.10.2 Één of meerdere keren +

Het teken + betekent dat het teken of de subexpressie voor + één of meerdere keren herhaald wordt.

Volgende regular expression betekent dat lidwoorden (**de** of **het**) minstens één keer voorkomen

regular expression	tekst	controle
(de het)+	de dieven zijn gevat	👍
(de het)+	de paarden eten het hooi	👍

visualisatie bovenstaande expressie



12.10.3 Nul of één keer ?

Het teken ? betekent dat het teken of de subexpressie voor ? nul of één keer herhaald wordt

Volgende regular expression betekent dat **email** of **e-mail** moet voorkomen

regular expression	tekst	controle
e-?mail	Lucky Luke leest geen email	👍
e-?mail	Lucky Luke leest geen e-mail	👍
e-?mail	de mails voor Lucky Luke gaan verloren	👎

visualisatie bovenstaande expressie



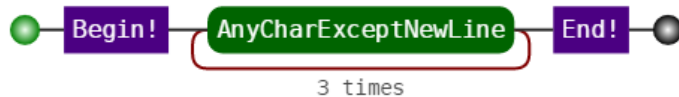
12.10.4 Een vast aantal keer {aantal}

Een getal tussen { en } betekent dat het teken of de subexpressie voor { zoveel keer herhaald wordt als dit getal

Volgende regular expression betekent dat de tekst 3 tekens moet bevatten

regular expression	tekst	controle
^. {3}\$	BEF	👍
^. {3}\$	BE	👎

visualisatie bovenstaande expressie



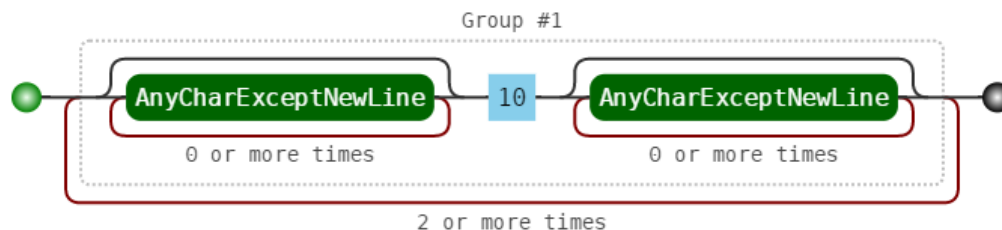
12.10.5 Een minimum aantal keer {minimum,}

Een getal tussen { en , } betekent dat het teken of de subexpressie voor { minstens zoveel keer herhaald wordt als dit getal

Volgende regular expression betekent dat het getal 10, eventueel voorafgegaan en/of gevolgd door tekst, minstens 2 keer moet voorkomen

regular expression	tekst	controle
(.*10.*){2,}	Mieke kreeg 10 keer 10/10	👍
(.*10.*){2,}	Jan kreeg een 10	👎

visualisatie bovenstaande expressie



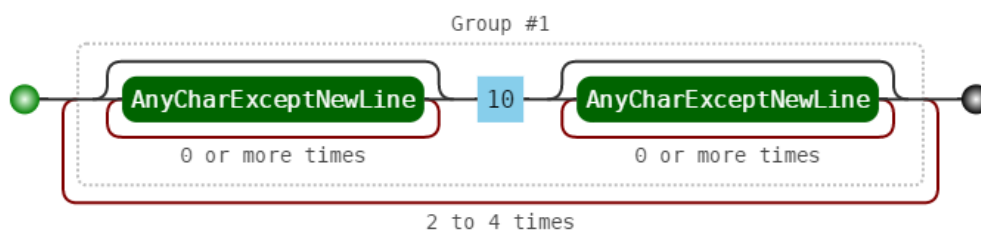
12.10.6 Een minimum én maximum aantal keer {minimum, maximum}

Twee getallen tussen { en } en gescheiden door , betekenen dat het teken of de subexpressie voor { minstens zoveel keer herhaald wordt als het 1° getal en hoogstens zoveel keer als het 2° getal

Volgende regular expression betekent dat het getal 10, eventueel voorafgegaan en/of gevolgd door tekst, tussen 2 en 4 keer moet voorkomen

regular expression	tekst	controle
(.*10.*){2,4}	Mieke kreeg 10 keer 10/10	👍
(.*10.*){2,4}	Jan kreeg een 10	👎

visualisatie bovenstaande expressie



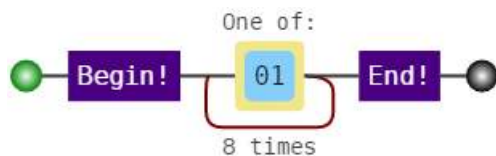
12.11 Één teken uit een lijst van tekens []

Tekens tussen vierkante haakjes betekent: één van deze tekens

Volgende expressie stelt een byte voor uitgedrukt als 8 bits

regular expression	tekst	controle
<code>^[01]{8}\$</code>	10100101	👍
<code>^[01]{8}\$</code>	20100101	👎

visualisatie bovenstaande expressie



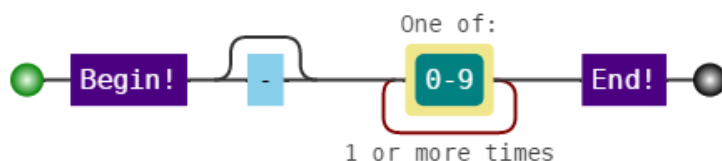
12.12 Één teken uit een bereik van tekens [-]

Twee tekens tussen vierkante haakjes én gescheiden door het teken - betekent de tekens die volgens Unicode liggen tussen het ene en het andere teken.

Volgende expressie stelt een geheel getal voor

regular expression	tekst	controle
<code>^-?[0-9]+\$</code>	123	👍
<code>^-?[0-9]+\$</code>	-69	👍
<code>^-?[0-9]+\$</code>	3.14	👎

visualisatie bovenstaande expressie



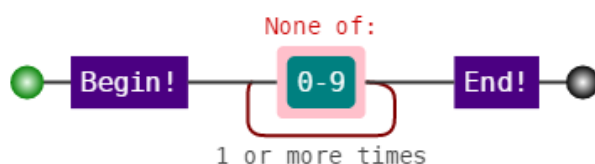
12.13 Alles behalve [^]

Als onmiddellijk na [het teken ^ komt, betekent dit alles behalve het bereik van tekens vermeld tussen [^en]

Volgende expressie stelt alles behalve een positief geheel getal voor

regular expression	Tekst	controle
<code>^[^0-9]+\$</code>	Alibaba	👍
<code>^[^0-9]+\$</code>	40	👎

visualisatie van bovenstaande expressie





12.14 Metacharacters \

Een metacharakter is een teken waarvoor \ staat en zo een speciale betekenis krijgt.

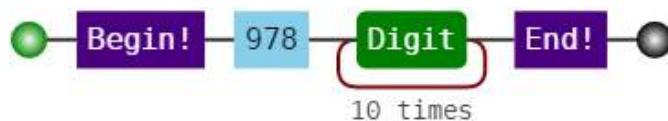
De meest gebruikte metacharacters

Metacharakter	betekenis
\t	Tab
\r	Carriage return
\n	Line feed
\s	Witruimte (spatie, tab, carriagereturn of line feed)
\S	Alles behalve witruimte
\d	Cijfer
\D	Alles behalve cijfer
\w	Alfanumeriek teken (tussen a-z of tussen A-Z of tussen 0-9)
\W	Alles behalve alfanumeriek teken

Volgende expressie stelt een ISBN nummer voor (978 gevolgd door 10 cijfers)

regular expression	Tekst	controle
<code>^978\d{10}\$</code>	9789027439642	
<code>^978\d{10}\$</code>	9781	

visualisatie bovenstaande expressie



Regular expression: zie takenbundel

13 Schema

13.1 Algemeen

Je beschrijft in een XML schema (een bestand met extensie xsd) de structuur van een XML bestand. Dit is belangrijk bij het uitwisselen van XML bestanden over dezelfde data:

pas als al deze bestanden dezelfde structuur hebben, kan je ze vlot verwerken in Java, PHP, c#, ...

Je beschrijft in een schema

- Welke de verplichte en optionele attributen zijn van een XML element
- Het soort inhoud van deze attributen: tekst of getallen of datums, ...
- Andere controles op deze attributen: de inhoud moet 3 tekens bevatten
- Welke child elementen verplicht moeten voorkomen in een element
- Welke child elementen optioneel kunnen voorkomen in een element
- Het soort inhoud van de tekst nodes van een element: tekst of getallen of datums, ...
- Andere controles op deze tekst nodes: de inhoud moet 3 tekens bevatten
- ...

Je associeert een XML data bestand met het bijbehorende schema.

Tools en libraries van programmeertalen kunnen dan controleren of de inhoud van het XML data bestand voldoet aan de structuur gedefinieerd in het schema.

XML die voldoet aan het schema waartoe deze XML behoort heet "valid XML".



13.2 DTD

XML schema is een standaard van het World Wide Web Consortium.

Dit consortium heeft nog een tweede standaard om de structuur van een XML bestand te definiëren: DTD (Document Type Definition). DTD wordt, wegens onderstaande beperkingen nog zelden gebruikt en wordt daarom niet uitgelegd in de cursus. Beperkingen van DTD, die niet bestaan in XML schema:

- DTD heeft een eigen syntax, XML schema gebruikt de XML syntax
- DTD kent geen namespaces, XML schema wel
- DTD kent geen types (tekst, getal, datum), XML schema wel
- Je kan in een DTD niet zelf types definiëren, in XML schema wel

13.3 Een eerste schema

Je opent Persoon.xsd. Je kiest bij Visual Studio

Use the XML editor to view and edit the underlying XML schema file

```

<?xml version="1.0" encoding="utf-8"?> (1)
<schema xmlns="http://www.w3.org/2001/XMLSchema" (2)
  <element name="persoon"/> (3)
</schema>
  
```

- (1) Een schema is zelf ook een XML bestand.
Dit betekent dat je ook kan controleren of een schema well formed is.
- (2) De elementen die je gebruikt in een XML schema om structuur te beschrijven behoren tot de namespace `http://www.w3.org/2001/XMLSchema`.
Het World Wide Web Consortium definieerde deze namespace.
schema en element zijn voorbeeldelementen die tot deze namespace behoren.
- (3) Je definieert dat in een XML bestand, dat zal verwijzen naar dit schema, de naam van het root element moet persoon zijn.

13.4 Vanuit een XML bestand verwijzen naar een schema

Je verwijst vanuit een XML bestand naar een schema, om aan te geven dat de structuur van dit XML bestand overeenkomt met de structuur-definitie in dit schema.

Je opent Jean.xml

```
<?xml version="1.0" encoding="utf-8"?>
<persoon xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"          (1)
    xsi:noNamespaceSchemaLocation="Persoon.xsd"/>                        (2)
```

- (1) Je verwijst bij (2) naar het schema met het attribuut noNamespaceSchemaLocation. Dit attribuut behoort tot de namespace `http://www.w3.org/2001/XMLSchema-instance`. Het World Wide Web Consortium definieerde ook deze namespace. Opgepast, dit is een andere namespace dan die in het schema zelf.
- (2) Je tikt bij noNamespaceSchemaLocation de naam van het schema dat bij dit XML bestand hoort. Schema's bevinden zich regelmatig op het internet, niet op je harde schijf. Je tikt dan bij noNamespaceSchemaLocation de URL waar dit schema zich bevindt.

13.5 Controleren of XML data voldoet aan een schema

13.5.1 NetBeans

- Je klikt met de rechtermuisknop in Jean.xml en je kiest Validate XML.
- Je ziet onder in het venster Output- XML check enkele boodschappen, maar geen foutboodschappen. Dit betekent dat de XML data voldoet aan het schema.

Je wijzigt in Jean.xml persoon naar mens

1. Je klikt terug met de rechtermuisknop in Jean.xml en je kiest Validate XML.
2. Je ziet onder in het venster Output- XML check enkele boodschappen, met deze keer ook een foutboodschap: `cvc-elt.1: Cannot find the declaration of element 'mens'`. Dit betekent dat de XML data niet voldoet aan het schema: dit bevat geen definitie van het element mens.

13.5.2 Visual Studio

Jean.xml bevat geen enkel stukje regel dat blauw golvend onderlijnd is.

Dit betekent dat dit bestand voldoet aan het schema waarnaar verwezen wordt.

Je wijzigt in Jean.xml persoon naar mens

mens wordt blauw golvend onderlijnd. Als je er de muisaanwijzer laat op rusten zie je de foutmelding `The 'mens' element is not declared`. Je kan dit lezen als: het element mens is niet gedefinieerd in het schema.



Schema zonder namespace: zie takenbundel

13.6 XML verbonden met een namespace

De XML data in Jean.xml was niet verbonden met een namespace. XML data is meestal wél verbonden met een namespace. Je leert wat je hiertoe doet in het schema en in het XML bestand.

In het voorbeeld is de XML verbonden met de namespace `http://www.vdab.be/personen`

Je opent PersoonMetNamespace.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.vdab.be/personen"          (1)
    <element name="persoon"/>
</schema>
```

- (1) Je tikt bij targetNamespace de namespace van de XML waarvoor dit schema zal dienen.

Je opent JeanMetNamespace.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<persoon xmlns="http://www.vdab.be/personen" (1)
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.vdab.be/personen PersonMetNamespace.xsd"/> (2)
```

- (1) Je associeert de XML met de namespace `http://www.vdab.be/personen`
- (2) Je tikt bij `schemaLocation` dezelfde namespace, gevolgd door een spatie, gevolgd door de naam van het schema bestand dat bij deze namespace hoort. Als dit schema zich op het internet bevindt, komt na de spatie de URL waar dit schema zich bevindt.



Schema met namespace: zie takenbundel

13.7 XMLSchema.xsd

Een schema is ook een XML bestand met een structuur.

`XMLSchema.xsd` beschrijft deze structuur. `XMLSchema.xsd` is zo "het schema van alle schema's".

Je vindt dit bestand op `http://www.w3.org/2009/XMLSchema/XMLSchema.xsd`.

Tools, libraries en IDE's bevatten een kopie van dit bestand,

zodat ze het bestand ook kunnen raadplegen als het internet niet ter beschikking is.

Je kan controleren of de structuur van je schema correct is ten opzichte van `XMLSchema.xsd`

- Je klikt in NetBeans met de rechtermuisknop in je schema en je kiest `Validate XML`
- Je controleert in Visual Studio of geen onderdeel van je schema blauw golvend onderlijnd is

13.8 Attributen

Je leert hier hoe je in je schema beschrijft welke attributen een element kan hebben.

Je doet dit op het root element. Je doet dit later op om het even welk element.

Je opent `Attributen.xsd`

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/personen" (1)
  targetNamespace="http://www.vdab.be/personen">
```

```
  <complexType name="Persoon"> (2)
```

```
    <attribute name="voornaam" use="required"/> (3)
```

```
    <attribute name="familienaam" use="required"/>
```

```
    <attribute name="aantalKinderen"/> (4)
```

```
    <attribute name="geboorteDatum"/>
```

```
    <attribute name="wedde"/>
```

```
    <attribute name="gehuwd"/>
```

```
  </complexType>
```

```
  <element name="persoon" type="vdab:Persoon"/> (5)
```

```
</schema>
```

- (1) Je associeert de namespace die hoort bij de XML, die bij dit schema zal horen, met de prefix `vdab`. Je gebruikt die prefix bij (5).
- (2) Je definieert een `complexType`. Een `complexType` beschrijft een prototype van een element dat attributen en/of child elementen heeft. Je geeft elk `complexType` een naam. Je verwijst verder in het schema naar dit `complexType` met deze naam. Je definieert bij (5) een element gebaseerd op dit prototype.
- (3) Je definieert dat elk element, gebaseerd op dit `complexType` een *verplicht* attribuut `voornaam` heeft. Als je `use="required"` weglaat, definieer je dat dit attribuut *kan* aanwezig zijn, maar niet *moet* aanwezig zijn.

- (4) Je definieert dat elk element, gebaseerd op dit complexType een optioneel attribuut aantalKinderen *kan* hebben.
- (5) Je definieert dat het root element persoon gebaseerd is op het prototype met de naam Persoon. Je tikt voor Persoon de prefix vdab. Als je deze prefix niet tikt, wordt Persoon gezocht in de default namespace, wat verkeerd is: de default namespace van het schema is <http://www.w3.org/2001/XMLSchema>



Opmerking: de volgorde waarmee je de attributen in XML beschrijft moet niet dezelfde zijn als de volgorde waarmee je de attributen beschrijft in het bijbehorende schema. In XML wordt de volgorde van attributen aanzien als onbelangrijk.

Je opent JeanAttributen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<persoon xmlns="http://www.vdab.be/personen"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/personen Attributen.xsd"
  voornaam="Jean" familienaam="Smits"/>
```

- Als je met je IDE dit bestand valideert ten opzichte van het schema is alles OK.
- Als je het attribuut voornaam en/of familienaam weglaat krijg je een fout.
- Als je het attribuut aantalKinderen, geboorteDatum, wedde of gehuwd toevoegt is alles OK
- Als je een ander attribuut toevoegt krijg je een fout

Als je in NetBeans een attribuut moet tikken, kan je Ctrl+Spatie drukken.

Je ziet dan een lijst (gebaseerd op het schema) met de mogelijke attributen waaruit je kan kiezen.

Als je in Visual Studio een attribuut moet tikken, kan je een spatie tikken.

Je ziet dan een lijst (gebaseerd op het schema) met de mogelijke attributen waaruit je kan kiezen.



Attributen: zie takenbundel

13.9 Ingebouwde simple types

Je kan tot nu definiëren welke attributen een element kan/moet hebben.

Je leert hier te definiëren welk *type* data een attribuut moet hebben. Je zal bijvoorbeeld definiëren dat het attribuut geboorteDatum een datum moet bevatten, niet zomaar wat tekst.

Schema heeft veel ingebakken types om het type van een attribuut te beschrijven.

Deze types heten "simple types". Je leert hier onder de belangrijkste simple types kennen.

We hebben ze in categorieën ingedeeld.

13.9.1 Tekst categorie

simple type	mogelijke inhoud
string	om het even welke tekst

13.9.2 Getal categorie

simple type	mogelijke inhoud
integer	geheel getal
positiveInteger	geheel getal groter dan 0
nonNegativeInteger	geheel getal groter of gelijk aan 0
negativeInteger	geheel getal kleiner dan 0
nonPositiveInteger	geheel getal kleiner of gelijk aan 0
decimal	getal met mogelijk decimalen

13.9.3 Logische categorie

simple type	mogelijke inhoud
boolean	true of false

13.9.4 Datum – tijd categorie

simple type	mogelijke inhoud
date	datum voorgesteld als jaar-maand-dag. Je tikt het jaar over 4 posities, de dag en maand over 2 posities. 1 augustus 1966 is dus 1966-08-01
time	tijd voorgesteld als uren:minuten:seconden. Je tikt uren, minuten en seconden over 2 posities. 8 uur 's avonds wordt dus 20:00:00
dateTime	datum-tijd combinatie. Je tikt de datum zoals bij het type date. Na de datum komt een T, gevolgd door de tijd, getikt zoals bij het type time. 1 augustus 1966 om acht uur 's avonds wordt dus 1966-08-01T20:00:00

Je wijzigt volgende regels in Attributen.xsd

```
<attribute name="aantalKinderen" type="nonNegativeInteger"/> (1)
<attribute name="geboorteDatum" type="date"/> (2)
<attribute name="wedde" type="decimal"/> (3)
<attribute name="gehuwd" type="boolean"/> (4)
```

(1) aantalKinderen moet minstens 0 zijn.

Je leert verder in de cursus hoe je ook een maximum kan instellen voor dit attribuut.

(2) geboorteDatum moet een datum zijn.

(3) wedde moet een getal (met mogelijk decimalen) zijn.

Je leert verder in de cursus hoe je ook een minimum kan instellen voor dit attribuut.

(4) gehuwd moet true of false zijn.

Je wijzigt JeanAttributen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<persoon xmlns="http://www.vdab.be/personen"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/personen Attributen.xsd"
  voornaam="Jean" familienaam="Smits"
  aantalKinderen="1" geboorteDatum="1966-08-01" wedde="5432.1" gehuwd="true"/>
```

Je controleert of de XML well formed en valid is.

Je tikt bij gehuwd de waarde soms en je controleert dat de XML nu niet valid is.

Je corrigeert gehuwd terug en je controleert of de XML well formed en valid is.



Ingebouwde simple types: zie takenbundel

13.10 Zelf gedefinieerde simple types

Je kan verfijnder de attribuut waarde controleren met een zelf gedefinieerd simple type.

Je baseert dit simple type op

- een ingebouwd simple type of
- een ander zelf gedefinieerd simple type

en je voegt extra controles toe aan je simple type

Deze extra controles op de attribuut waarde kunnen volgende vormen aannemen

- een ondergrens en/of bovengrens definiëren
- de tekstlengte definiëren
- een lijst definiëren waartoe de waarde moet behoren
- maximum aantal cijfers na de komma of maximum totaal aantal cijfers definiëren
- een regular expression definiëren waaraan de waarde moet voldoen

13.10.1 Ondergrens en/of bovengrens

Je definieert 777 als de bovengrens voor het attribuut `aantalKinderen`

Je maakt daartoe een eigen simple type `AantalKinderen`, gebaseerd op het ingebakken simple type `nonNegativeInteger`.

Je tikt volgende regels in `Attributen.xsd` voor de regel `<complexType name="Persoon">`

```
<simpleType name="AantalKinderen"> (1)
  <restriction base="nonNegativeInteger"> (2)
    <maxInclusive value="777"/> (3)
  </restriction>
</simpleType>
```

(1) Elk simple type moet een unieke naam hebben.

Er is geen naamgevingconventie voor types. We starten in deze cursus met een hoofdletter.

Je type behoort tot de namespace bij `targetNamespace`: `http://www.vdab.be/personen`

(2) Je vermeldt een simple type waarop je het huidig simple type baseert

(3) Je stelt een maximum waarde in. Ook `maxExclusive`, `minInclusive`, `minExclusive` bestaan

Je gebruikt dit type bij het attribuut `aantalKinderen` van het complexType `Persoon`

```
<attribute name="aantalKinderen" type="vdab:AantalKinderen"> (1)
```

(1) Als je de prefix `vdab` niet zou vermelden, zou XML `AantalKinderen` zoeken in de default namespace (`http://www.w3.org/2001/XMLSchema`)

Je definieert 0 als ondergrens en 10000 als bovengrens voor het attribuut `wedde`

Je maakt daartoe eerst een eigen simple type `Wedde`

```
<simpleType name="Wedde">
  <restriction base="decimal">
    <minInclusive value="0"/>
    <maxInclusive value="10000"/>
  </restriction>
</simpleType>
```

Je gebruikt dit type bij het attribuut `wedde` van het complexType `Persoon`

```
<attribute name="wedde" type="vdab:Wedde">
```

Je definieert dat het attribuut `geboorteDatum` moet liggen tussen 1/1/1900 en 31/12/2999.

Je maakt daartoe eerst een eigen simple type `GeboorteDatum`

```
<simpleType name="GeboorteDatum">
  <restriction base="date">
    <minInclusive value="1900-01-01"/>
    <maxInclusive value="2999-12-31"/>
  </restriction>
</simpleType>
```

Je gebruikt dit type bij het attribuut `wedde` van het complexType `Persoon`

```
<attribute name="geboorteDatum" type="vdab:GeboorteDatum">
```

Je slaat het schema op

Je kan in `JeanAttributen.xml` attributen wijzigen naar een juiste of verkeerde waarde en controleren of de XML valid is.

13.10.2 Tekstlengte

Je zal de taal van een persoon bijhouden als een taalcode. Deze heeft een vaste lengte van 2 tekens. Je maakt daartoe eerst een eigen simple type TaalCode.

```
<simpleType name="TaalCode">
  <restriction base="string">
    <length value="2"/>
  </restriction>
</simpleType>
```

(1)

(1) Je definieert dat de attribuut waarde 2 tekens moet bevatten.

Ook minLength (minimum lengte te definiëren) en maxLength (maximum lengte definiëren) bestaan. Je kan minLength en maxLength combineren (samen gebruiken) in je simple type

Je gebruikt dit type bij een nieuw attribuut taalCode in het complexType Persoon

```
<attribute name="taalCode" type="vdab:TaalCode"/>
```

Je kan in JeanAttributen.xml het attribuut taalCode="nl" toevoegen en controleren of de XML valid is.

13.10.3 Maximum aantal cijfers na de komma – maximum totaal aantal cijfers

Je verfijnt het type Wedde: het bevat maximaal 2 cijfers na de komma

```
<simpleType name="Wedde">
  <restriction base="decimal">
    <minInclusive value="0"/>
    <maxInclusive value="10000"/>
    <fractionDigits value="2"/>
  </restriction>
</simpleType>
```

(1)

(1) Je definieert dat het getal maximaal 2 cijfers na de komma heeft.

Ook totalDigits bestaat. Je definieert daarmee het maximaal aantal cijfers in het volledige getal (voor plus na de komma) te definiëren.

Je kan fractionDigits en totalDigits combineren.

Je kan in JeanAttributen.xml het attribuut wedde wijzigen en controleren of de XML valid is.

13.10.4 Lijst

Je zal het geslacht van een persoon bijhouden als man of vrouw. Je laat geen andere waarden toe.

Je maakt daartoe eerst een eigen simple type Geslacht.

```
<simpleType name="Geslacht">
  <restriction base="string">
    <enumeration value="man"/>
    <enumeration value="vrouw"/>
  </restriction>
</simpleType>
```

(1)
(2)

(1) man is één van de mogelijke waarden

(2) vrouw is een andere mogelijke waarde

Je gebruikt dit type bij een nieuw attribuut geslacht in het complexType Persoon

```
<attribute name="geslacht" type="vdab:Geslacht"/>
```

Je kan in JeanAttributen.xml volgend attribuut toevoegen: geslacht="man" en controleren of de XML valid is.

13.10.5 Regular expression



Een regular expression in een schema controleert of de *volledige* tekst klopt ten opzichte van de regular expression. Je kan en mag de tekens ^ en \$ niet gebruiken om het begin en het einde van de tekst aan te geven. Deze tekens behouden hun letterlijke betekenis.

Je verfijnt het type `TaalCode`. Het moet uit twee tekens bestaan die elk liggen tussen a en z

```
<simpleType name="TaalCode">
  <restriction base="string">
    <pattern value="[a-z]{2}" />
  </restriction>
</simpleType>
```

(1)

(1) Deze regular expression definieert een tekst met twee tekens tussen a en z

Je kan in `JeanAttributen.xml` het attribuut `taalCode` wijzigen en controleren of de XML valid is.



Zelf gedefinieerde simple types: zie takenbundel

13.11 Child elementen

13.11.1 sequence – choice – all

Je leert hoe je in je schema beschrijft welke child elementen een element kan hebben.

Je beschrijft de child elementen van een element als een *sequence*, een *choice* of een *all*

sequence	De child elementen beschreven in het schema moeten allen voorkomen in de XML. De volgorde van de child elementen de XML moet dezelfde zijn zoals in het schema.
all	De child elementen beschreven in het schema moeten allen voorkomen in de XML. De volgorde in de XML mag verschillen van de volgorde in het schema.
choice	Één child element van degene beschreven in het schema mag voorkomen in de XML

13.11.2 Simple type – complex type

- Als één van de child elementen enkel een tekst node heeft (geen attributen en/of child elementen), kan je bij dit child element een (ingebouwd of zelf gedefinieerd) simple type vermelden. De tekst node moet dan voldoen aan dit simple type.
- Als één van de beschreven child elementen attributen en/of child elementen heeft, kan je bij dit het child element een complex type vermelden.
De structuur van dit child element moet dan voldoen aan dit complex type.

13.11.3 sequence

Je opent `Adres.xsd`

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/adressen"
  targetNamespace="http://www.vdab.be/adressen"
  elementFormDefault="qualified">
```

(1)

```
  <simpleType name="Postcode">
    <restriction base="int">
      <minInclusive value="1000"/>
      <maxInclusive value="9999"/>
    </restriction>
  </simpleType>
  <complexType name="Adres">
    <sequence>
```

(2)

```
    <element name="straat"/>
    <element name="huisnummer"/>
    <element name="postcode" type="vdab:Postcode"/>
```

```

    <element name="gemeente"/>
  </sequence>
</complexType>
<element name="adres" type="vdab:Adres"/>
</schema>

```

- (1) Standaard behoren child elementen niet tot de namespace bij targetNamespace, tenzij je bij ieder child element volgend attribuut zou tikken: form="qualified". Door één keer elementFormDefault="qualified" in het element schema te tikken behoren child elementen wél tot de namespace bij targetNamespace
- (2) Alle child elementen, beschreven tussen <sequence> en </sequence> zullen in de XML moeten voorkomen, in dezelfde volgorde zoals beschreven in het schema.

Je opent Hoofdbestuur.xml

```

<?xml version="1.0" encoding="utf-8"?>
<adres xmlns="http://www.vdab.be/adressen"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/adressen Adres.xsd">
  <straat>Keizerslaan</straat>
  <huisnummer>11</huisnummer>
  <postcode>1000</postcode>
  <gemeente>Brussel</gemeente>
</adres>

```

Je kan controleren of de XML valid is.

Als je de volgorde van de child elementen wijzigt is de XML niet meer valid.

13.11.4 all

Je wijzigt in Adres.xsd <sequence> naar <all> en </sequence> naar </all>.

Nu moeten alle child elementen, beschreven tussen <all> en </all> voorkomen in de XML, maar de volgorde mag anders zijn.

Je kan dit nazien door Hoofdbestuur.xml te wijzigen en te valideren.

13.11.5 choice

Je opent Locatie.xsd. Het geeft de keuze (choice) om een locatie te beschrijven met

- een adres (straat, huisnummer, postcode en gemeente) of
- met een point (longitude en latitude).

```

<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/Locaties"
  targetNamespace="http://www.vdab.be/Locaties"
  elementFormDefault="qualified">
  <simpleType name="Postcode">
    <restriction base="int">
      <minInclusive value="1000"/>
      <maxInclusive value="9999"/>
    </restriction>
  </simpleType>
  <complexType name="Adres">
    <all>
      <element name="straat"/>
      <element name="huisnummer"/>
      <element name="postcode" type="vdab:Postcode"/>
      <element name="gemeente"/>
    </all>
  </complexType>

```

```

<complexType name="Point">
  <attribute name="longitude" type="decimal" use="required"/>
  <attribute name="latitude" type="decimal" use="required"/>
</complexType>
<complexType name="Locatie">
  <choice>
    <element name="adres" type="vdab:Adres"/>
    <element name="point" type="vdab:Point"/>
  </choice>
</complexType>
<element name="Locatie" type="vdab:Locatie"/>
</schema>

```

(1)

- (1) Van de child elementen, beschreven tussen <choice> en </choice> zal in de XML slechts één mogen voorkomen

Je opent Hoofdbestuur2.xml

```

<?xml version="1.0" encoding="utf-8"?>
<locatie xmlns="http://www.vdab.be/Locaties"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/Locaties Locatie.xsd">
  <adres>
    <straat>Keizerslaan</straat>
    <huisnummer>11</huisnummer>
    <postcode>1000</postcode>
    <gemeente>Brussel</gemeente>
  </adres>
</locatie>

```

Je kan controleren of de XML valid is.

Je vervangt in Hoofdbestuur2.xml de regels <adres> tot en met </adres> door

```
<point longitude="50.84289889999999" latitude="4.352501200000006"/>
```

Je kan terug controleren of de XML valid is.

13.11.6 minOccurs – maxOccurs

Standaard mag een child element één keer voorkomen in zijn parent element.

Je kan hiervan afwijken met de attributen minOccurs en maxOccurs.

Je past deze attributen toe op <element ...> elementen in je schema.

minOccurs definieert hoeveel keer het child element minstens voorkomt in zijn parent element.

Standaard (als je minOccurs niet vermeldt) staat dit op 1. Je kan 0 tikken bij minOccurs.

Je geeft zo aan dat het child element *optioneel* voorkomt in zijn parent element.

maxOccurs definieert hoeveel keer het child element maximaal voorkomt in zijn parent element.

Standaard (als je maxOccurs niet vermeldt) staat dit op 1. Je kan unbounded tikken bij maxOccurs.

Je geeft zo aan dat het child element een *oneindig* aantal keer voorkomt in zijn parent element.

Je kan minOccurs en maxOccurs samen gebruiken bij een <element ...>

Je opent NaamVoornamenGeboorte.xsd

```

<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/naamVoornamenGeboorte"
  targetNamespace="http://www.vdab.be/naamVoornamenGeboorte"
  elementFormDefault="qualified">
  <complexType name="Voornamen">
    <sequence>
      <element name="voornaam" maxOccurs="unbounded"/>
    </sequence>

```

(1)

```

</complexType>
<complexType name="NaamVoornamen">
  <sequence>
    <element name="naam"/>
    <element name="voornamen" type="vdab:Voornamen"/>
    <element name="geboorte" type="date" minOccurs="0"/>
  </sequence>
</complexType>
<element name="naamVoornamen" type="vdab:NaamVoornamen"/>
</schema>

```

(2)

- (1) Je tikt geen minOccurs. Het element voornaam moet dus minstens één keer voorkomen in zijn parent element. Je tikt unbounded bij maxOccurs.
Het element voornaam kan dus oneindig keer voorkomen in zijn parent element.
- (2) Je tikt 0 bij minOccurs. Het element geboorte is dus een optioneel child element van zijn parent element. Je tikt geen maxOccurs.
Het element geboorte kan dus maximaal één keer voorkomen in zijn parent element.

Je opent JanWillemLincolnMetSchema.xml

```

<?xml version="1.0" encoding="utf-8"?>
<naamVoornamen xmlns="http://www.vdab.be/naamVoornamenGeboorte"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/naamVoornamenGeboorte
    NaamVoornamenGeboorte.xsd">
  <naam>van de Wetering</naam>
  <voornamen>
    <voornaam>Jan Willem</voornaam>
    <voornaam>Lincoln</voornaam>
  </voornamen>
</naamVoornamen>

```

Je kan controleren of de XML valid is. Als je een geboortedatum toevoegt blijft de XML valid.
Als je geen enkele voornaam tikt is de XML niet meer valid.



Complex type: zie takenbundel

13.12 Een complex type met child elementen én attributen

Als een complex type child elementen én attributen heeft, definieer je eerst de child elementen, daarna de attributen.

Je opent NaamVoornamenGeboorteAttributen.xsd

```

<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/naamVoornamenGeboorte"
  targetNamespace="http://www.vdab.be/naamVoornamenGeboorte"
  elementFormDefault="qualified">
  <complexType name="Voornamen">
    <sequence>
      <element name="voornaam" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="NaamVoornamen">
    <sequence>
      <element name="voornamen" type="vdab:Voornamen"/>
    </sequence>
    <attribute name="naam" use="required"/>
    <attribute name="geboorte" type="date"/>
  </complexType>

```

(1)

(2)

```
<element name="naamVoornamen" type="vdab:NaamVoornamen"/>
</schema>
```

- (1) Je definieert eerst de child elementen
- (2) Je definieert daarna de attributen

Je opent JanWillemLincolnAttributenMetSchema.xml

```
<?xml version="1.0" encoding="utf-8"?>
<naamVoornamen xmlns="http://www.vdab.be/naamVoornamenGeboorte"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/naamVoornamenGeboorte
    NaamVoornamenGeboorteAttributen.xsd"
  naam="van de Wetering">
  <voornamen>
    <voornaam>Jan Willem</voornaam>
    <voornaam>Lincoln</voornaam>
  </voornamen>
</naamVoornamen>
```

Je kan controleren of de XML valid is.

13.13 Een complex type erft van een ander complex type

Een complex type kan de definitie van een ander complex type erven en daarop uitbreiden.

Je opent Punt.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/punt"
  targetNamespace="http://www.vdab.be/punt">
  <complexType name="TweeDimensieelPunt">
    <attribute name="x" type="integer" use="required"/>
    <attribute name="y" type="integer" use="required"/>
  </complexType>
  <complexType name="DrieDimensieelPunt">
    <complexContent>
      <extension base="vdab:TweeDimensieelPunt">
        <attribute name="z" type="integer" use="required"/>
      </extension>
    </complexContent>
  </complexType>
  <element name="punt" type="vdab:DrieDimensieelPunt"/>
</schema>
```

- (1) Je vermeldt het complex type waarvan je erft
- (2) Je voegt een attribuut als uitbreiding toe

Je opent Punt.xml

```
<?xml version="1.0" encoding="utf-8"?>
<punt xmlns="http://www.vdab.be/punt"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/punt Punt.xsd"
  x="6" y="28" z="496"/>
```

Je kan controleren of de XML valid is

13.14 key

Je definieert met key een attribuut of child element dat ingevuld én uniek moet zijn bij ieder element uit een lijst. Dit lijkt op een primary key in een table in een database.

Je opent Strips.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/strips"
  targetNamespace="http://www.vdab.be/strips"
  elementFormDefault="qualified">
  <simpleType name="ISBN">
    <restriction base="integer">
      <pattern value="978\d{10}"/>
    </restriction>
  </simpleType>
  <complexType name="Strip">
    <attribute name="titel" use="required"/>
    <attribute name="isbn" use="required" type="vdab:ISBN"/>
  </complexType>
  <complexType name="Strips">
    <sequence>
      <element name="strip" type="vdab:Strip" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <element name="strips" type="vdab:Strips">
    <key name="isbnKey">                                (1)
      <selector xpath="vdab:strip"/>                  (2)
      <field xpath="@isbn"/>                            (3)
    </key>
  </element>
</schema>
```

- (1) Je definieert een primary key met een element key. Elke primary key krijgt een unieke name.
- (2) Je tikt bij selector een xpath expressie. Deze moet verwijzen naar herhalende child elementen binnen het huidig gedefinieerd element. Het huidig gedefinieerd element is strips, op de regel boven (1). Dit bevat herhalende child elementen met de naam strip.
- (3) Je tikt bij field een xpath expressie. Deze expressie verwijst naar een attribuut of child element van de herhalende elementen bij (2). Dit attribuut of child element moet bij ieder herhalend element een unieke waarde hebben. In dit voorbeeld geef je aan dat het attribuut isbn van ieder strip element een unieke waarde moet bevatten.

Je opent Strips.xml. Dit verwijst naar Strips.xsd.

Dit bestand is valid. Als je echter twee strips hetzelfde ISBN geeft, is het bestand niet meer valid.

13.15 keyref

Je definieert met keyref een attribuut of child element dat moet bestaan als primary key.

Dit lijkt op een foreign key in een table in een database.

Je opent LuckyLuke.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/LuckyLuke"
  targetNamespace="http://www.vdab.be/LuckyLuke" elementFormDefault="qualified">
  <simpleType name="ISBN">
    <restriction base="integer">
      <pattern value="978\d{10}"/>
    </restriction>
```



```

</simpleType>
<complexType name="Medewerker">
  <attribute name="id" use="required" type="positiveInteger"/>
  <attribute name="naam" use="required"/>
</complexType>
<complexType name="Strip">
  <attribute name="titel" use="required"/>
  <attribute name="isbn" use="required" type="vdab:ISBN"/>
  <attribute name="tekenaarId" use="required" type="positiveInteger"/>
  <attribute name="schrijverId" use="required" type="positiveInteger"/>
</complexType>
<complexType name="Medewerkers">
  <sequence>
    <element name="medewerker" type="vdab:Medewerker" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<complexType name="Strips">
  <sequence>
    <element name="strip" type="vdab:Strip" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<complexType name="LuckyLuke">
  <sequence>
    <element name="medewerkers" type="vdab:Medewerkers"/>
    <element name="strips" type="vdab:Strips"/>
  </sequence>
</complexType>
<element name="LuckyLuke" type="vdab:LuckyLuke">
  <key name="idKey">
    <selector xpath="vdab:medewerkers/vdab:medewerker"/>
    <field xpath="@id"/>
  </key>
  <key name="isbnKey">
    <selector xpath="vdab:strips/vdab:strip"/>
    <field xpath="@isbn"/>
  </key>
  <keyref name="tekenaarKeyRef" refer="vdab:idKey">
    <selector xpath="vdab:strips/vdab:strip"/>
    <field xpath="@tekenaarId"/>
  </keyref>
  <keyref name="schrijverIdKeyRef" refer="vdab:idKey">
    <selector xpath="vdab:strips/vdab:strip"/>
    <field xpath="@schrijverId"/>
  </keyref>
</element>
</schema>

```

- (1) Deze key definieert dat de id van iedere medewerker een unieke waarde moet hebben.
- (2) Deze key definieert dat de isbn van iedere strip een unieke waarde moet hebben.
- (3) Deze keyref definieert dat de tekenaarId van iedere strip een bestaande id van een medewerker moet bevatten.
- (4) Deze xpath expressie verwijst naar strip elementen.
- (5) Deze xpath expressie verwijst naar het attribuut tekenaarId van een strip element.
- (6) Deze keyref definieert dat de schrijverId van iedere strip een bestaande id van een medewerker moet bevatten.

Je opent LuckyLuke.xml. Deze verwijst naar LuckyLuke.xsd

Dit bestand is valid. Als je echter bij een strip als tekenaarId of als schrijverId een onbestaande medewerker id tikt, is het bestand niet meer valid.

13.16 Documentatie

Je kan de menselijke betekenis van attributen en elementen documenteren in een schema. Je doet dit met een child element annotation, daarin een child element documentation en daarin de menselijke betekenis.

Je probeert dit uit in Punt.xsd. Je wijzigt `<attribute name="x" ... />` naar

```
<attribute name="x" type="integer" use="required">
  <annotation>
    <documentation>De X coördinaat van het punt</documentation>
  </annotation>
</attribute>
```

Je IDE toont deze documentatie terwijl je XML intikt die bij dit schema past.

13.16.1 NetBeans

Je verwijdert in Punt.xml de attributen x, y en z en je tikt Ctrl+Spatie. Je ziet een lijstje met de mogelijke attributen, waarbij het attribuut x geselecteerd is. Je ziet onder dit lijstje een venster met daarin onder andere de tekst "De X coördinaat van het punt"

13.16.2 Visual Studio

Je verwijdert in Punt.xml de attributen x, y en z en je tikt Spatie. Je ziet een lijstje met de mogelijke attributen, waarbij het attribuut x geselecteerd is. Je ziet naast dit attribuut de tekst "De X coördinaat van het punt"

13.17 Stappenplan

Je kan met volgend stappenplan een schema maken voor een bestaand XML bestand.

De uitleg is gebaseerd op Boek.xml. Een beroepencode moet 6 cijfers bevatten.

```
<?xml version="1.0"?>
<boek boeknr="405" isbn="1590590104">
  <titel>Database Programming with C#</titel>
  <schrijver>Thomsen</schrijver>
  <uitgever>Apress</uitgever>
  <curriculum nr="38">
    <naam>.NET ontwikkelaar met C#</naam>
    <beroepencode>750412</beroepencode>
  </curriculum>
</boek>
```

1. Je opent een nieuw XML schema. Je definieert in dit schema een namespace, bijvoorbeeld `http://www.vdab.be/boek`.

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:vdab="http://www.vdab.be/boek"
  targetNamespace="http://www.vdab.be/boek" elementFormDefault="qualified">
</schema>
```

2. Je maakt een lijst met de attributen. Je vermeldt naast ieder attribuut het bijbehorende datatype. Dit kan een ingebouwd simple type zijn of een zelf gedefinieerd simple type (als een ingebouwd simple type niet volstaat)
 - o boeknr positiveInteger
 - o isbn ISBN (zelf gedefinieerd)
 - o nr positiveInteger

3. Je maakt de zelf gedefinieerde simple types die je in punt 2 ontdekte

```
<simpleType name="ISBN">
  <restriction base="integer">
    <pattern value="978\d{10}"/>
  </restriction>
</simpleType>
```

4. Je maakt een lijst met de elementen die enkel een tekst node hebben (geen child elementen en/of attributen). Je vermeldt naast ieder element het bijbehorende datatype. Dit kan een ingebouwd simple type zijn of een zelf gedefinieerd simple type (als een ingebouwd simple type niet volstaat)

- o titel string
- o schrijver string
- o uitgever string
- o naam string
- o beroepencode Beroepencode (zelf gedefinieerd)

5. Je maakt de zelf gedefinieerde simple types die je in punt 4 ontdekte

```
<simpleType name="Beroepencode">
  <restriction base="integer">
    <pattern value="\d{6}"/>
  </restriction>
</simpleType>
```

6. Je maakt complex types. Je start met de elementen op het laagste niveau. Je eindigt met het root element.

```
<complexType name="Curriculum">
  <sequence>
    <element name="naam" type="string"/>
    <element name="beroepencode" type="vdab:Beroepencode"/>
  </sequence>
  <attribute name="nr" use="required" type="positiveInteger"/>
</complexType>

<complexType name="Boek">
  <sequence>
    <element name="titel" type="string"/>
    <element name="schrijver" type="string"/>
    <element name="uitgever" type="string"/>
    <element name="curriculum" type="vdab:Curriculum"/>
  </sequence>
  <attribute name="boeknr" use="required" type="positiveInteger"/>
  <attribute name="isbn" use="required" type="vdab:ISBN"/>
</complexType>
```

7. Je definieert het root element

```
<element name="boek" type="vdab:Boek"/>
```

8. Je definieert eventueel keys en keyrefs. Dit is in Boek.xml niet nodig.

Je kan nu Boek.xml verbinden met dit schema

```
<?xml version="1.0" encoding="utf-8"?>
<boek xmlns="http://www.vdab.be/boek"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.vdab.be/boek Boek.xsd"
  boeknr="405" isbn="1590590104">
```

(1)

...



Bieren: zie takenbundel

14 Libraries

Je kan de libraries, waarmee je vanuit een programmeertaal XML leest, in twee categorieën verdelen

- **DOM (Document Object Model) libraries**
Deze lezen de XML *volledig* binnen in het interne geheugen en bieden die XML aan als een boomstructuur
- **SAX (Simple API for XML) libraries**
Deze lezen sequentieel element per element binnen in het interne geheugen.
Het interne geheugen bevat op ieder moment slechts *één* XML element

Beide categorieën hebben voor- en nadelen

Kenmerk	SAX	DOM
RAM verbruik	Laag	Hoger, afhankelijk van omvang XML data
Snelheid	Zeer hoog	Redelijk
Leesrichting	Enkel voorwaarts door elementen	Willekeurig (je kan bvb. ook achteruit lezen)
ReadOnly	Ja	Nee. Je kan waarden wijzigen en de gewijzigde XML opslaan
XPath	Niet bruikbaar	Bruikbaar

Meestal worden DOM libraries gebruikt, tenzij de XML zeer groot is (tientallen megabytes of meer)

COLOFON

Domeinexpertisemanager	Rita Van Damme
Moduleverantwoordelijke	
Auteurs	Hans Desmet
Versie	16/11/2015
Codes	Peoplesoftcode: Wettelijk depot:

Omschrijving module-inhoud

Abstract	Doelgroep	Opleiding Java Ontwikkelaar
	Aanpak	Zelfstudie
	Doelstelling	XML kunnen gebruiken
Trefwoorden		XML
Bronnen/meer info		