XML in perspectief

Leereenheid 1

XML in perspectief

INTRODUCTIE

Deze leereenheid geeft een introductie in XML en webservices en in de nieuwe mogelijkheden die deze technologieën hebben geschapen.

In paragraaf 1 wordt aan de hand van een voorbeeld geïllustreerd wat XML is, hoe 'data' en 'document' bij XML tot één begrip zijn samengesmolten en hoe via XML-grammatica's wordt afgedwongen dat uitgewisselde documenten een voorgeschreven vorm hebben.

Paragraaf 2 gaat in op de problematiek die ontstaat wanneer een aantal applicaties met elkaar gegevens moeten uitwisselen, waarbij het dataformaat van de ene applicatie niet aansluit op dat van de andere applicatie. In dat geval moet conversie plaatsvinden van het ene formaat naar het andere. XML kan als platformonafhankelijke taal helpen om de complexiteit van deze conversies te vereenvoudigen.

Paragraaf 3 gaat over uiteenlopende XML-toepassingen en illustreert dat XML een 'taalmaker' is die voor elk terrein waarbij gegevensuitwisseling in geheel of gedeeltelijk gestandaardiseerde vorm dient plaats te vinden, een taal op maat kan definiëren.

Paragraaf 4 geeft een inleiding in een belangrijk type XML-toepassing: webservices. In blok 4 wordt uitvoerig op webservices ingegaan.

LEERDOELEN

Na het bestuderen van deze leereenheid wordt verwacht dat u:

- inzicht hebt in de semantiek en de globale structuur van een XML-document
- inzicht hebt in de wijze waarop de structuur van een XML-document kan worden afgedwongen via een XML-grammatica
- enig inzicht hebt in de complexiteit van gegevensuitwisseling tussen applicaties en de rol die XML daarbij kan spelen.
- enkele XML-toepassingen kunt schetsen met voorbeelden van XML-documenten daarbij
- inzicht hebt in de aard van webservices

Studeeraanwijzingen

De benodigde voorbeelden of bouwstenen van deze leereenheid zijn gebundeld in twee oXygen-projecten: xml01_perspectief en (voor paragraaf 4) xml01_tempconvertor.
De leereenheid heeft géén zelftoets.

Studielast

De studielast van deze leereenheid bedraagt 5 uur.

LEERKERN

1 Wat is XML?

In deze paragraaf bekijken we allereerst een voorbeeld van een XML-document. Van het XML-document bestuderen we enkele in het oog lopende kenmerken en tevens zullen we het vanuit grammaticaal gezichtspunt beschouwen. Ook zullen we zien waarom XML en het web elkaars ontwikkeling zo enorm hebben versterkt.

1.1 VOORBEELD VAN EEN XML-DOCUMENT

Als voorbeeld bekijken we een XML-document dat op het internet wordt uitgewisseld tussen twee applicaties die met elkaar communiceren in het kader van de elektronische marktplaats.

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

   <klant>
      <klantnaam>Inkoper</klantnaam>
      <adres land="Netherlands">
         <straat>Drienerlolaan</straat>
         <huisnr>999</huisnr>
         <postcode>7500 AE</postcode>
         <plaats>Enschede</plaats>
      </adres>
   </klant>

   <leverancier>
      <bedrijfnaam>ScannerCo</bedrijfnaam>
      <adres land="Netherlands">
         <straat>Uranusstraat</straat>
         <huisnr>28</huisnr>
         <postcode>5432 AS</postcode>
         <plaats>Amsterdam</plaats>
      </adres>
   </leverancier>

   <logistiek>
      <logistiekeDienst>
         <bedrijfnaam>Fast Delivery Service</bedrijfnaam>
      </logistiekeDienst>
      <snelheid>within 24 hours</snelheid>
   </logistiek>

   <producten>
      <product aantal="3">
         <productnaam>Ultrascan</productnaam>
      </product>
   </producten>

</bestelling>
```

Toelichting

XML-declaratie

1 Het document begint met een zogenaamde *XML-declaratie*. Deze zegt zoiets als: ‘dit is een XML-document, geschreven in XML-versie 1.0 en gecodeerd en met character-codering UTF-8 ’. Versie 1.0 is, op het moment dat deze cursus uitkomt, nog steeds de actuele versie van XML.

De codering UTF-8 is een bepaalde binaire codering van de door XML gebruikte tekenverzameling (characterset) Unicode.

Tag

2 Uit HTML herkent u de *tags*. De voorschriften voor tags zijn in XML stricter dan bij HTML . Zo moet elke openingstag worden gevolgd door een sluittag en is het overlappen van tag-paren verboden.

Element

Root-element

3 Een tagpaar met de inhoud daarbinnen heet een *element*. Een welgevormd XML-document heeft altijd één hoofdelement, *root* of *root-element* genoemd, waarvan alle andere elementen subelementen zijn. In dit document is dat het element bestelling.

Leeg element

Net als in HTML kan een element leeg zijn en dan mogen openings- en sluittag tot één tag worden samengevoegd. Bijvoorbeeld: de XML-code voor een leeg element spoed luidt <spoed></spoed> en kan worden verkort tot <spoed/>.

Attribuut

4 De meeste informatie komt voor als elementinhoud, tussen de openingstag en de sluittag. Sommige informatie is echter gegeven als waarde van een *attribuut*, zoals de landnamen en de productaantallen. Tot op zekere hoogte is de keuze tussen elementinhoud of attribuut-waarde willekeurig. Hier komen we later in de cursus op terug.

5 Het element 'producten' heeft één subelement 'product'. Dit zouden er echter meer mogen zijn, bijvoorbeeld:

```
<producten>
   <product aantal="3">
      <productnaam>Ultrascan</productnaam>
   </product>
   <product aantal="1">
      <productnaam>ScanJet</productnaam>
   </product>
</producten>
```

6 Aan de structuur van het XML-document liggen keuzen ten grondslag: de gegeven structuur is het resultaat van een bepaalde *modellering*. Er zijn dan ook vele alternatieven denkbaar. De interne structuur van het element klant had bijvoorbeeld ook als volgt kunnen zijn:

```
<klant>
   <klantnaam>Inkoper</klantnaam>
   <adres>Drienerlolaan 999</adres>
   <postcode>7500 AE</postcode>
   <plaats>Enschede</plaats>
   <land>Netherlands</land>
</klant>
```

De leverancier-adresinformatie moet dan op analoge wijze worden aangepast. We benadrukken dat XML niet is ontworpen vanuit het oogpunt van informatiemodellering. Wie gewend is opslagstructuren te ontwerpen in bijvoorbeeld een relationele-databaseomgeving, zal wellicht zo nu en dan de wenkbrauwen optrekken. Hier is inderdaad heel wat over te zeggen, want informatiestructuren zijn 'van zichzelf' (om dat maar even zo te noemen) veelal niet hiërarchisch. Alleen wanneer de XML-data intrinsiek hiërarchisch zijn, levert XML een natuurlijk opslagformaat, geschikt voor specifieke XML-databases. In andere gevallen is XML in de eerste plaats een communicatieformaat.

OPGAVE 1.1

De voor deze opgave benodigde handleidingen zijn te openen en eventueel te downloaden via de cursussite. De software (Eclipse en oXygen) staat op de met de cursus meegeleverde cd-rom.

a Installeer Eclipse volgens de installatiehandleiding.

b Bestudeer de Basishandleiding Eclipse. (Specifieke Java-onderwerpen zijn pas van belang vanaf leereenheid 9.)

c Installeer de oXygen-plugin.

d Bestudeer de Basishandleiding oXygen t/m paragraaf 3.

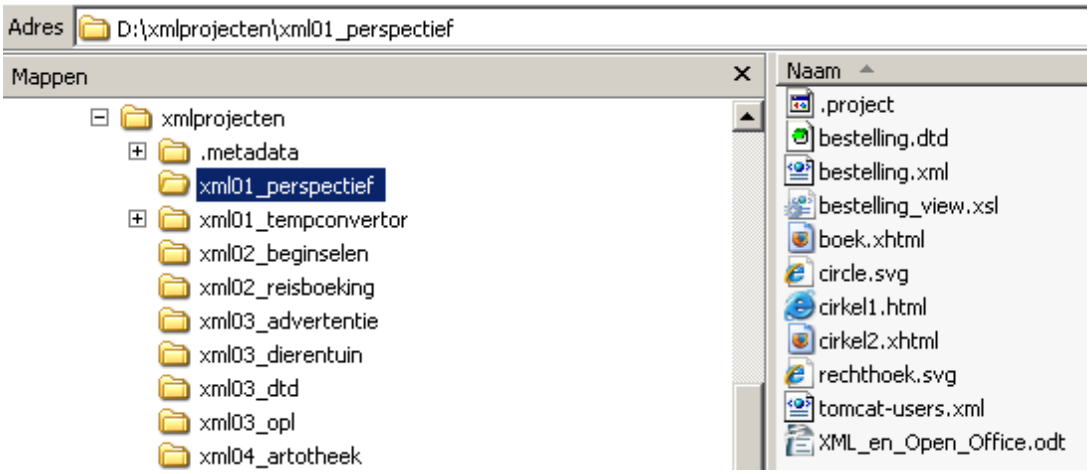
In alle hierna volgende opgaven wordt aangenomen dat u over basisvaardigheden beschikt in het omgaan met Eclipse en oXygen.

Klaarzetten van projectmap (workspace)

In opgave 1.2 zet u alle in deze cursus te gebruiken Eclipse/oXygen-projecten klaar in één map, genaamd xmlprojecten. Door de aanwezige meta-informatie (de submap .metadata en de bestanden .project in de andere submappen) vormt deze een Eclipse-*workspace*.

OPGAVE 1.2

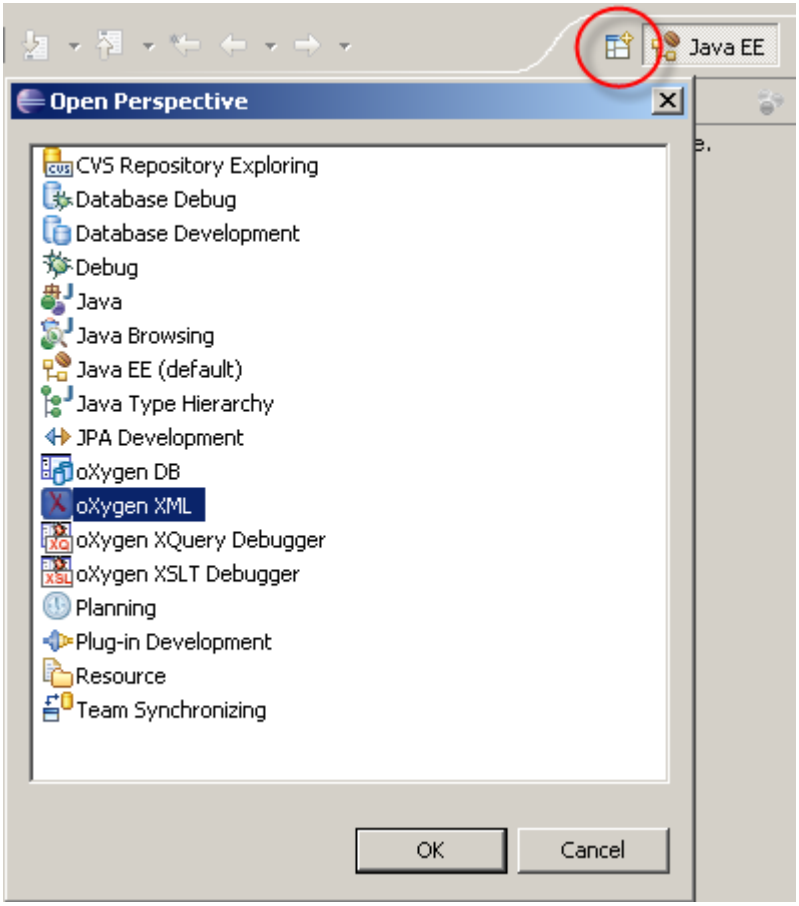
a Download de zip-file met projectbestanden en pak deze uit op een plek naar keuze. In de tekst zullen we aannemen dat dit d:\ is. Alle projectmappen zijn dan submappen van d:\xmlprojecten (zie figuur 1.1).



FIGUUR 1.1 Projectmappen

b Selecteer in Eclipse de map xmlprojecten als actieve workspace. (Eclipse start hierna opnieuw op.)

c Selecteer het Eclipse-perspectief 'oXygen XML' (zie figuur 1.2). U krijgt hierdoor de beschikking over een op XML toegespitste GUI.



FIGUUR 1.2 Keuze van perspectief oXygen XML

OPGAVE 1.3

Het XML-document in de voorgaande tekst is beschikbaar in het project xml01_perspectief, onder de naam bestelling.xml. In deze opgave echter bekijken we bestelling.xml nog niet via Eclipse/oXygen maar via achtereenvolgens een teksteditor en een browser.

a Open bestelling.xml eerst in een eenvoudige teksteditor, zoals Windows Kladblok. U hoort nu een 'kale' weergave van de XML-tekst te zien.

b Open bestelling.xml in uw browser. U zult een weergave krijgen met elementen (knooppunten) die via een muisklik zijn in- of uit te klappen. Probeer dit uit. Figuur 1.3 geeft de weergave in Internet Explorer, na het inklappen van alle elementen die direct onder het root-element liggen.

FIGUUR 1.3 Weergave van XML-document met ingeklapte elementen

Dat u het XML-document in de browser anders ziet weergegeven, komt doordat het XML-document door de browser is getransformeerd naar een HTML-document. Dit bevat JavaScript-code die voor het dynamische gedrag zorgt.

OPGAVE 1.4

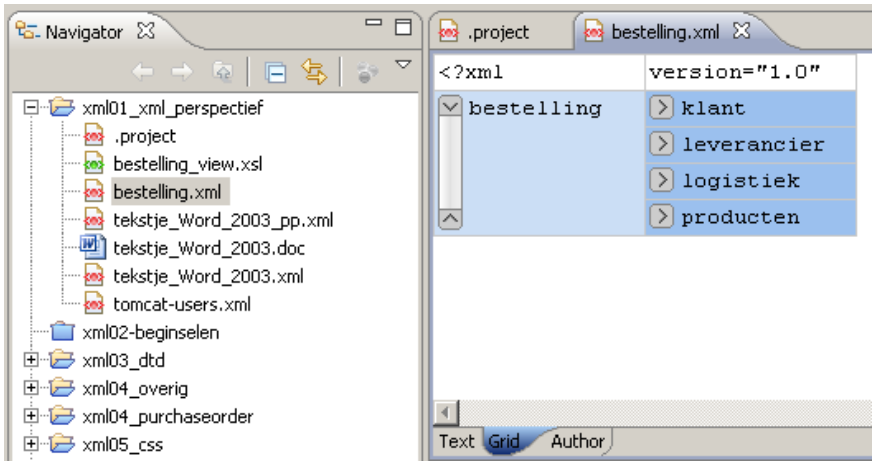
In deze opgave bekijken we bestelling.xml via Eclipse/oXygen.

a Open in Eclipse het project xml01_perspectief.

b Open bestelling.xml.

c Experimenteer met de verschillende manieren van weergeven: tekstueel (tabblad Text) en grafisch met in- en uitklapbare elementen (tabblad Grid). Zie figuur 1.4.

d Open ook het XML-bestand .project, met meta-informatie over het project. (Dit bestand is ook geopend in figuur 1.4, getuige het tabblad .project.)



FIGUUR 1.4 Bestelling.xml in grid-view

In de volgende paragraaf gaan we dieper in op weergave van XML-documenten.

1.2 SCHEIDING VAN BETEKENIS EN WEERGAVE

XML schrijft een strenge scheiding voor tussen betekenis en weergave van gegevens. Het XML-document bestelling.xml van de vorige paragraaf bevat alleen tag- en attribuutnamen die een betekenis weergeven. Zolang dit document alleen deel uitmaakt van de communicatie tussen twee applicaties, is dat voldoende. Er zijn echter veel situaties waarin XML-gegevens ook moeten worden afgebeeld. In opgave 1.1 zagen we al wat de standaardmanier van afbeelden is in een browser: een min of meer letterlijke weergave, echter met dynamiek in de vorm van in- en uitklapbare elementen. Er zijn echter vele manieren om XML-documenten of een selectie van de gegevens erin, precies zo weer te geven als we willen, of dat nu in een browser is of op papier. Enkele van de technieken hiervoor worden in latere leereenheden behandeld.

Stylesheet

In opgave 1.5 geven we een voorproefje, om de scheiding van betekenis en weergave te illustreren. We hebben gekozen voor weergave via een *stylesheet* (opmaakmodel), geschreven in de taal XSLT. Dit is een programmeertaal, speciaal ontworpen om XML-gegevens te transformeren naar andere XML-structuren. Zo'n andere XML-structuur kan die van een HTML-document zijn, mits dat voldoet aan de XML-welgevormdheidsregels. (Feitelijk hebben we het dan over XHTML, de XML-vorm van HTML.) Een aardige (hoewel niet per se goede) eigenschap van XSLT is dat het zelf een XML-taal is.

OPGAVE 1.5

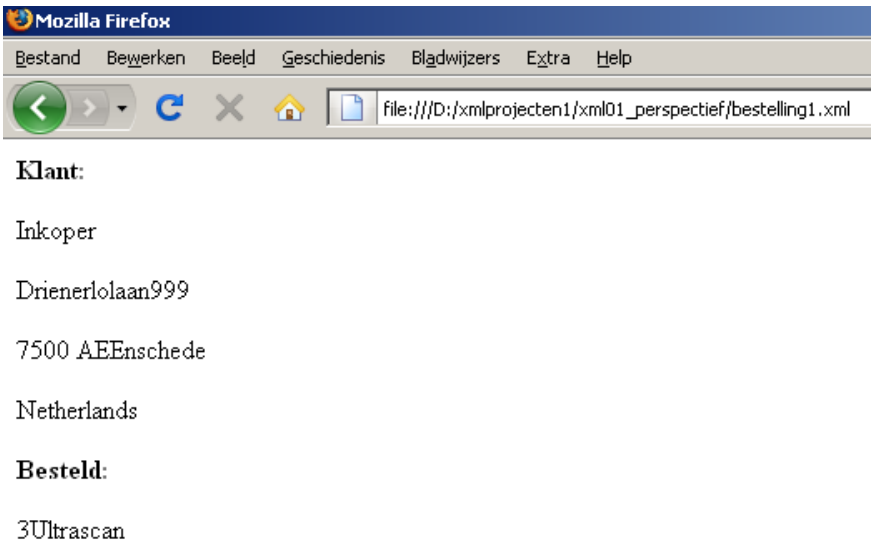
a Open de XSLT-stylesheet bestelling_view.xsl in een teksteditor.

b Het is in dit stadium niet de bedoeling dat u alle codedetails begrijpt. Ga echter na dat in deze stylesheet (die u ook een programma mag noemen) een HTML-document wordt geassembleerd uit stukjes letterlijke HTML-code en gegevens die uit het brondocument (bestelling.xml) worden gehaald. In de uitwerking geven we een beknopte toelichting.

c Neem in bestelling.xml direct na de XML-declaratie de volgende regel op, waarmee wordt aangegeven dat dit document moet worden gepresenteerd via de stylesheet bestelling_view.xsl:

```
<?xml-stylesheet type="text/xsl" href="bestelling_view.xsl"?>
```

d Sla het gewijzigde bestelling.xml op en open het in uw browser. U hoort nu een weergave te zien zoals in figuur 1.5. Zo niet, dan is uw browser waarschijnlijk niet 'XSLT-enabled'. Probeer het dan met een meer recente versie of een andere browser.
Merk op dat de stylesheet nog niet perfect is, gezien de ontbrekende spaties her en der. We zullen nog meermalen zien dat whitespace (spaties, tabs, regelovergangen) ons voor lastige problemen stelt.



FIGUUR 1.5 Weergave in Firefox van de inhoud van bestelling.xml via de XSLT-stylesheet bestelling_view.xsd

1.3 DOCUMENTSTRUCTUREN EN DATASTRUCTUREN

Het XML-document van paragraaf 1.1 is zowel een instantie van een *documentstructuur* als een instantie van een *datastructuur*. Het is typerend voor XML dat tussen de begrippen 'document' en (instantie van) 'datastructuur' geen formeel onderscheid wordt gemaakt. In andere contexten denken we bij een document aan een nogal 'vrij' format en bij een datastructuur aan een strak format zoals in de context van programmeertalen. XML kan echter met alle mogelijke formats overweg, met elke mate van vrijheid daarin. Het betreffende onderscheid vervalt daardoor.

1.4 WELGEVORMDHEID EN VALIDITEIT

Er zijn verschillende manieren om het format van een XML-document vast te leggen, dat wil zeggen: de structuur aan beperkingen te binden door middel van een *grammatica*.

Welgevormde XML-documenten

Welgevormde XML-documenten
Allereerst zijn er algemene grammaticale regels waaraan elk XML-document moet voldoen. Deze regels heten *XML-welgevormdheidsregels*. Een voorbeeld van zo'n welgevormdheidsregel is dat tags alleen maar genest mogen voorkomen. Zo is <a> … … … toegestaan, maar <a> … … … niet. Een document dat aan alle welgevormdheidsregels voldoet heet een *welgevormd* XML-document. Eigenlijk is hier sprake van een pleonasme: de eigenschap 'welgevormdheid' zit al opgelsoten in het begrip 'XML-document'. In leereenheid 2 gaan we dieper in op de 'welgevormdheidsgrammatica' van XML.

Valide XML-documenten

Valide XML-documenten
Voor de meeste toepassingen is welgevormdheid niet genoeg, maar moeten de XML-documenten aan specifieke vormeisen voldoen, zoals tags en attributen, in een voorgeschreven structuur. Die vormeisen

worden beschreven in een *grammatica*. Een XML-document dat aan zo'n grammatica voldoet, heet *valide*.

We geven een voorbeeld van een grammatica voor XML-documenten zoals bestelling.xml:

```
<?xml encoding="UTF-8"?>
<!ELEMENT bestelling (klant, leverancier, logistiek, producten)>
<!ELEMENT klant (klantnaam, adres)>
<!ELEMENT leverancier (bedrijfnaam, adres)>
<!ELEMENT logistiek (logistiekeDienst,snelheid)>
<!ELEMENT producten (product)>
<!ELEMENT klantnaam (#PCDATA)>
<!ELEMENT logistiekeDienst (bedrijfnaam)>
<!ELEMENT snelheid (#PCDATA)>
<!ELEMENT product (productnaam)>
<!ATTLIST product aantal CDATA #REQUIRED>
<!ELEMENT productnaam (#PCDATA)>
<!ELEMENT adres (straat, huisnr, postcode, plaats)>
<!ATTLIST adres land NMTOKEN #REQUIRED>
<!ELEMENT straat (#PCDATA)>
<!ELEMENT huisnr (#PCDATA)>
<!ELEMENT postcode (#PCDATA)>
<!ELEMENT plaats (#PCDATA)>
<!ELEMENT bedrijfnaam (#PCDATA)>
```

Documenttype-definitie (DTD)

Deze grammatica is een zogenaamde *documenttypedefinitie* (*DTD*). Hij is in de projectmap beschikbaar als bestelling.dtd.
DTD's vormen een ouder type XML-grammatica. Ze zijn eenvoudig te begrijpen: de tweede regel zegt bijvoorbeeld dat een bestelling-element moet bestaan uit de achtereenvolgende subelementen klant, leverancier, logistiek en producten.
Hoofdstuk 3 is in zijn geheel aan DTD's gewijd. In opgave 1.6 zien we een voorproefje: validatie van bestelling.xml tegen bestelling.dtd.
Een nieuwer type XML-grammatica, geschreven in de taal XML Schema, wordt behandeld in hoofdstuk 4.

OPGAVE 1.6

In deze opgave valideren we bestelling.xml tegen bestelling.dtd. Dat gaat niet vanzelf, omdat bestelling.xml nog geen verwijzing bevat naar bestelling.dtd.

a Open in oXygen zowel bestelling.xml als bestelling.dtd.

b Neem in bestelling.xml de volgende regel op, direct na de XML-declaratie (dus als tweede regel):

```
<!DOCTYPE bestelling SYSTEM "bestelling.dtd">
```

Wanneer oXygen geen fouten meldt, betekent dit dat boekbestelling.xml valide is.

1.5 XML EN DE ONTWIKKELING VAN HET WEB

XML is niet uit de lucht komen vallen. Er is een relatie met zowel HTML (*HyperText Markup Language*) als SGML (*Standardized General Markup Language*), die elk al een – voor internetbegrippen – lange voorgeschiedenis hebben. In deze paragraaf kijken we naar de relatie tussen HTML, SGML en XML. Kort ook bespreken we de nieuwere XML-versie (XHTML) van HTML.
We sluiten de paragraaf af met wat informatie over het orgaan dat de ontwikkeling van internet (en daarbij ook XML) in goede banen probeert te leiden: het World wide web consortium (W3C).

HTML en XML

HTML en XML

HTML wordt in deze cursus bekend verondersteld. Wat we hier nog willen benadrukken, is dat de meeste HTML-markup dient voor presentatiedoeleinden en dus niets zegt over de betekenis van de gegevens.

In tegenstelling tot HTML is XML primair op betekenis gericht en niet op presentatie: XML is 'semantisch rijker' dan HTML. Verder zijn de welgevormdheidsregels van XML veel strakker dan die van HTML (zie paragraaf 2.1 voor enkele van die regels). Een onwelgevormd XML-document wordt door de verwerkende software doorgaans geweigerd, terwijl een browser van elk HTML-document nog wel iets probeert te maken, ook al is het nog zo'n rommeltje. Dat wijst op nog een ander verschil: de weergave van een HTML-document is gericht op een menselijke lezer via tussenkomst van een browser, terwijl XML-documenten primair worden geschreven of geassembleerd voor machine-machinecommunicatie (juister: software-software-communicatie).

Bij elke opsomming van overeenkomsten en verschillen moeten we één fundamenteel verschil in het oog houden: HTML is een taal, maar XML is een mechanisme om talen te maken. Er zijn vele XML-talen. Ook de nieuwere versies van HTML zijn volledig gebaseerd op XML, en elke versie is te beschouwen als één XML-taal. Hier komen we nog op terug, maar eerst moeten we iets uitleggen over de oudere, en zeer krachtige taal SGML, waarmee XML en HTML beide een – zij het nogal verschillende – relatie hebben.

SGML en XML

SGML en XML

Wat de talen SGML en XML gemeen hebben, is dat zij beide beschikken over een mechanisme om, door het opleggen van extra grammaticale regels, subtalen voor specifieke doeleinden te definiëren. Bijvoorbeeld subtalen voor speciale toepassingen. Het SGML-mechanisme hiervoor was de DTD (*Document type definitie*). Een DTD is niets anders dan een grammaticale specificatie, in aanvulling op de algemene (en zeer ruime) welgevormdheidsregels. We zouden kunnen zeggen: SGML + DTD geeft een specifieke SGML-taal. SGML heeft enorm veel mogelijkheden, maar werd voor grootschalige toepassing op internet te complex bevonden.

XML was aanvankelijk niets anders dan een vereenvoudiging van SGML. Men zou kunnen zeggen: met de eenvoud van HTML. Ook XML werkte (en werkt nog steeds) met DTD's en is als zodanig, evenals SGML, als een 'taalmaker' te beschouwen. De DTD's, die dus een erfenis zijn van SGML,

worden inmiddels als verouderd beschouwd. Hoewel ze net als de meeste andere verouderde software nog een lang leven zullen hebben, wordt voor nieuwe en geavanceerde toepassingen een ander type grammatica gebruikt: *schema's* (geschreven in een schemataal, zoals XML Schema). Zoals reeds vermeld zijn de leereenheden 3 en 4 achtereenvolgens gewijd aan DTD's en schema's. Daar zal ook uit de doeken worden gedaan waarom DTD's voor bepaalde toepassingen ontoereikend zijn.

SGML en HTML

SGML en HTML

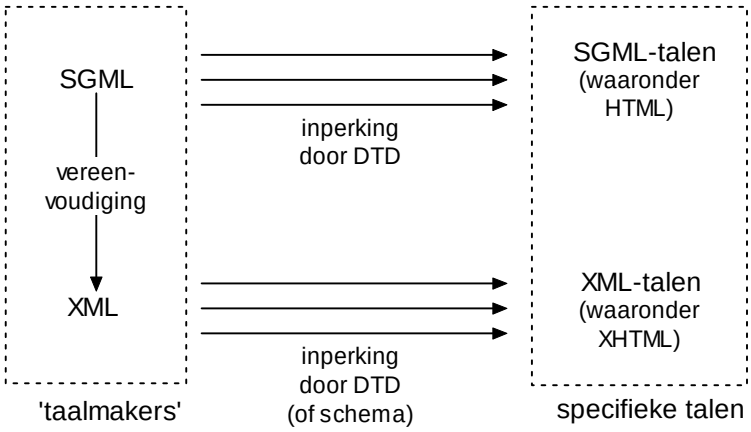
HTML (tot en met versie 1) is niets anders dan één specifieke SGML-taal. U mag hieruit dus concluderen dat er een specifieke 'SGML-DTD voor HTML' bestaat, die de welgevormdheidsregels van SGML inperkt tot precies de taal HTML. De 'vrijheid' in HTML (bijvoorbeeld om tags te laten overlappen, of openingstags te gebruiken zonder sluittag) heeft HTML geërfd van SGML. Die vrijheid, hoewel misschien lelijk vanuit taaltheoretisch standpunt, heeft wel bijgedragen aan het succes van HTML. Evenals de welwillendheid van browsers, die altijd wel 'een oogje dichtknijpen' als een HTML-document niet aan de regels voldoet.

XHTML

XHTML

De 'netheid' en eenvoud van XML heeft zijn uitwerking op HTML niet gemist. Was HTML tot en met versie 1 gebaseerd op SGML, vanaf versie 2 is het gebaseerd op XML. Vanaf deze versie wordt gesproken over XHTML. Bedenk hierbij dat XML weliswaar niet primair op presentatie is gericht, maar dat aan tags in specifieke XML-talen best wel een 'presentatie-betekenis' (!) kan worden gegeven. Het is maar wat we afspreken, en met name wat we afspreken om de verwerkende software te laten doen. Leereenheid 5 'Presentatie van XML-inhoud' is voor een deel aan XHTML gewijd.

In figuur 1.6 wordt de samenhang tussen SGML, DTD's, HTML, XML en XHTML, zoals in de voorgaande subparagrafen besproken, overzichtelijk weergegeven.



FIGUUR 1.6 De relatie tussen SGML, DTD's, HTML, XML en XHTML

World wide web consortium (W3C)

World wide web consortium

Het World Wide Web Consortium (W3C), voorgezeten door internetpionier Tim Berners-Lee, is het belangrijkste orgaan dat probeert de ontwikkeling van

internet zodanig te sturen dat deze ontwikkeling wordt gestimuleerd, en niet door wildgroei en tegenstrijdige standaards wordt afgeremd. In het W3C zijn wetenschappers en marktpartijen verenigd. Zeker bij de ontwikkelingen rondom XML is het tot nu toe wonderwel gelukt om de partijen (denk aan Microsoft, IBM en dergelijke) niet al te ver van elkaar te laten afdrijven.
Het middel waarvan het W3C zich bedient, is dat van *recommendations* (aanbevelingen). De W3C-recommendations zijn de-factostandaarden, maar worden minder dwingend opgelegd dan bijvoorbeeld ISO-standaarden, waardoor de acceptatie soepeler verloopt.

De website http://www.w3.org van het W3C heeft een aantal subwebs die uitstekende vertrekpunten vormen voor het zoeken van informatie over XML:
– http://www.w3.org/xml
– http://www.w3.org/tr/#recommendations (zie onder meer de site-index).

In andere leereenheden zullen we nog wel vaker naar W3- of W3C-webs verwijzen. Niet alleen vormen ze een primaire bron, ze zijn vaak ook redelijk tot zeer toegankelijk geschreven en vormen daardoor een geschikt naslagwerk.

De ontwerpdoelen van XML
Aan de website van W3C (zie http://www.w3.org/tr/rec-XML) ontlenen we de volgende tien ontwerpdoelen, waarop de specificatie van XML versie 1.0 is gebaseerd:

1 XML moet zonder belemmeringen zijn te gebruiken over het internet.
2 XML moet een ruime variëteit aan applicaties ondersteunen.
3 XML moet compatibel zijn met SGML.
4 Het moet eenvoudig zijn programma's te schrijven die XML-documenten verwerken.
5 Het aantal optionele kenmerken van XML moet tot het absolute minimum worden beperkt en dient idealiter nul te zijn.
6 XML-documenten moeten door mensen gelezen kunnen worden en redelijk helder zijn.
7 Het ontwerp van XML dient snel te worden voorbereid.
8 Het ontwerp van XML dient formeel en beknopt te zijn.
9 XML-documenten moeten gemakkelijk te creëren zijn.
10 Beknoptheid van XML-markup is van minimaal belang.

Opmerkingen

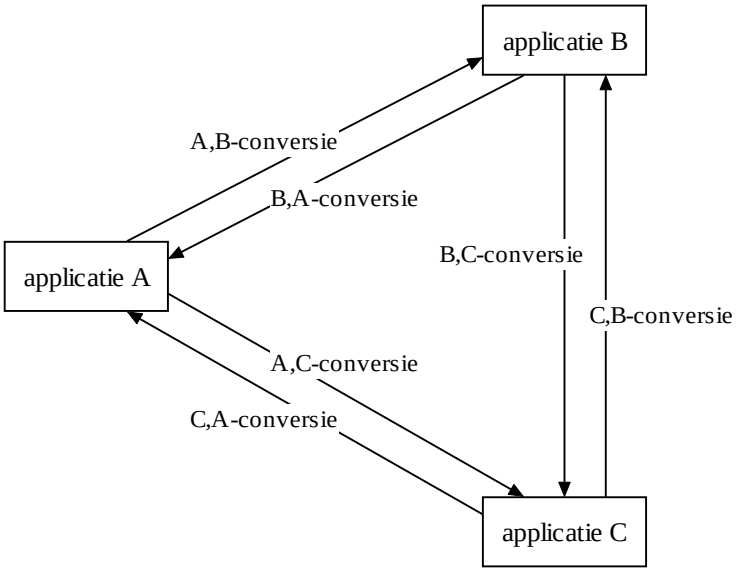
Ontwerpdoel 7 ('The XML design should be prepared quickly') lijkt ons niet helemaal in het rijtje thuis te horen: het is immers geen eis aan het ontwerp zelf, maar aan het ontwerpproces.
Inmiddels is duidelijk geworden dat alle doelstellingen die met eenvoud te maken hebben, bij veel toepassingen niet zijn gehaald. Dit komt ten dele door de complexiteit van de toepassingen, ten dele ook – zoals we nog zullen bespreken – door ontwerpkeuzen voor XML zelf.

2 XML als uitwisselingsformaat

Het gebeurt vaak dat applicaties gegevens met elkaar moeten uitwisselen maar dat het dataformaat van de ene applicatie niet aansluit op dat van de andere applicatie. In dat geval is conversie nodig van het ene gegevensformaat naar het andere. XML kan als platformonafhankelijke taal helpen de complexiteit van de conversieproblematiek te vereenvoudigen.

2.1 COMPLEXITEIT VAN DE CONVERSIEPROBLEMATIEK

Wanneer er geen algemene afspraken zijn over een neutrale uitwisselingstaal, moet voor elk tweetal applicaties dat onderling gegevens uitwisselt, een specifieke conversie plaatsvinden, vaak zowel de ene als de andere kant op. Figuur 1.7 brengt dit in beeld voor drie applicaties die onderling gegevens uitwisselen. Dit geeft 3×2 = 6 verschillende soorten van dataconversie. Voor n applicaties zijn dat maximaal n(n-1) = n^2-n verschillende manieren van conversie. Het conversieprobleem is – globaal bezien – dus van 'kwadratische complexiteit'. Per applicatie is de complexiteit lineair: maximaal 2(n-1) verschillende manieren van conversie van of naar de applicatie.



FIGUUR 1.7 Gegevenuitwisseling zonder neutrale uitwisselingstaal: een probleem van kwadratische complexiteit

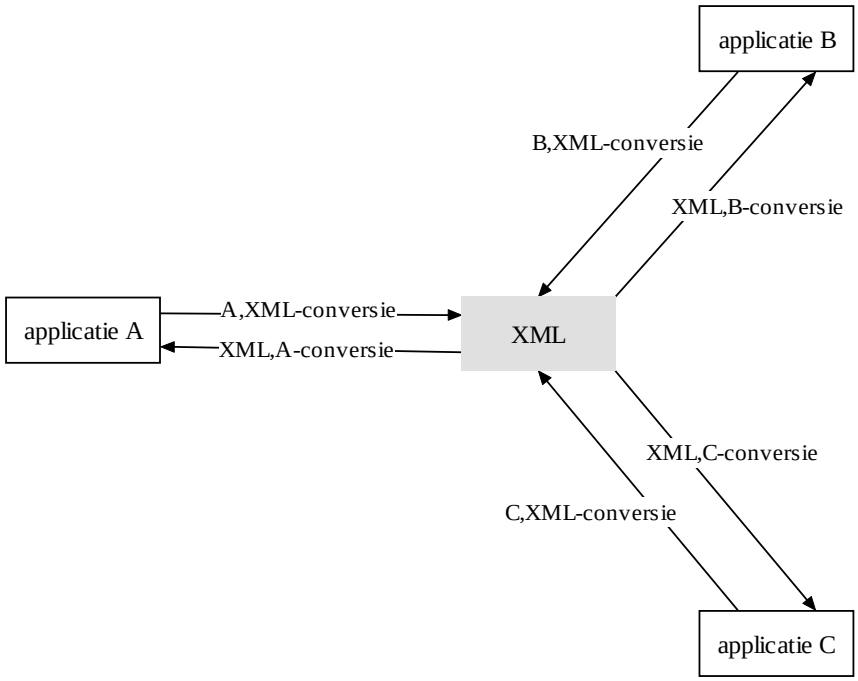
Afgezien van het aantal conversies is niet a priori duidelijk waar de verantwoordelijkheid ligt voor de conversie tussen een applicatie A en een applicatie B. Voor conversie van A richting B kan A worden voorzien van een B-exportconvertor, maar evengoed kan B worden voorzien van een A-importconvertor.

2.2 XML ALS LINGUA FRANCA

Een oplossing voor het probleem van de grote, namelijk kwadratische complexiteit van gegevensuitwisseling en voor het probleem waar de verantwoordelijkheid ligt, is gelegen in het gebruik van een neutrale uitwisselingstaal. Op deelgebieden bestaan dergelijke talen al heel lang, bijvoorbeeld in de databasewereld het comma separated format. Het gebruik en de mogelijkheden van dergelijke talen waren echter altijd heel beperkt.

De laatste jaren is XML naar voren gekomen als een universeel neutraal uitwisselingsformaat, een ware 'lingua franca' voor informatie-uitwisseling. Figuur 1.8 brengt dit in beeld voor de drie applicaties van

figuur 1.8. Elke applicatie exporteert haar gegevens in XML-vorm. En niet alleen de gegevens maar ook de grammatica (DTD of schema) die de structuur van de gegevens beschrijft. De XML-gegevens zijn voor elke applicatie waarmee gegevens moeten worden uitgewisseld beschikbaar. De globale complexiteit is niet langer kwadratisch maar lineair: maximaal 2n manieren van conversie, en per applicatie maar twee: één exportconversie en één importconversie.



FIGUUR 1.8 Gegevensuitwisseling met neutrale uitwisselingstaal: een probleem van lineaire complexiteit

2.3 VOORBEELD: XML-EXPORT VAN DATABASE

MySQL is een voorbeeld van een databasemanagementsysteem met een XML-exportoptie. Allereerst volgt hier een voorbeeld van hoe MySQL via een DOS-interface zijn data op een character-gebaseerde, maar min of meer grafische manier exporteert. De gegevens zijn afkomstig uit een kleine MySQL-boekendatabase boekenDB, met daarin een tabel Boek met twee rijen.

Het commando

C:\>mysql -uroot -e "use boekenDB; SELECT * FROM Boek;"

geeft als output:

```
+-----------------------------------------+----------------------------+---------------+-------+-----------+
| titel                                   | auteur                     | isbn          | prijs | categorie |
+-----------------------------------------+----------------------------+---------------+-------+-----------+
| XML Development with Java 2             | Michael Daconta et al.     | 0-321-45687-4 | 44.99 | Java      |
| An introduction to XML and Web technologies | Anders Moller et al.   | 0-334-10293-3 |    53 | XML       |
+-----------------------------------------+----------------------------+---------------+-------+-----------+
```

En nu precies dezelfde data, maar als XML-export. Het commando wordt hiertoe uitgebreid met de optie `--xml`:

```
C:\>mysql --xml -uroot -e "use boekenDB; SELECT * FROM Boek;"
```

De output is nu in XML:

```
<?xml version="1.0"?>
<resultset statement="SELECT * FROM Boek" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <row>
    <field name="titel">XML Development with Java 2</field>
    <field name="auteur">Michael Daconta et al.</field>
    <field name="isbn">0-321-45687-4</field>
    <field name="prijs">44.99</field>
    <field name="categorie">Java</field>
  </row>
  <row>
    <field name="titel">An introduction to XML and Web technologies</field>
    <field name="auteur">Anders Moller et al.</field>
    <field name="isbn">0-334-10293-3</field>
    <field name="prijs">53</field>
    <field name="categorie">XML</field>
  </row>
</resultset>
```

Toelichting
1 Het root-element heet resultset, niet vreemd omdat het document de resultaatverzameling is van de XML-export. Dit element heeft een attribuut genaamd statement, met als waarde het gekozen SQL-exportstatement. Het resultaat daarvan is – zoals van elk SQL-statement – een tabel. De subelementen van het element resultset zijn dan ook rijen, aangegeven door de elementnaam row. Elke 'row' heeft kolomwaarden, aangegeven door subelementen met de naam field.
2 Merk op dat we de metadata van de tabel (de kolomnamen) in de XML-export terugzien als attribuutwaarden en de tabelinhoud als elementinhoud.

3 Enkele XML-toepassingen

Het aantal XML-toepassingen is inmiddels 'ontelbaar'. Hun succes is nauw verbonden met hét sterke punt van XML, namelijk dat elke XML-taal een platform- en programmeertaalonafhankelijk specificatie- of uitwisselingsformaat vormt. De toepassingen vormen dan ook veelal een volgende generatie ten opzichte van voorgangers, waarbij het formaat wel nauw verbonden was met een specifiek platform en specifieke programmatuur.

Ter illustratie bespreken we enkele voorbeelden van XML-talen. Elke taal is te beschouwen als een XML-toepassing of een onderdeel daarvan. De nadruk bij de voorbeelden in deze paragraaf ligt op XML-talen voor visuele weergave. Dat is niet helemaal representatief, maar het is gedaan omdat dat laagdrempelige voorbeelden zijn. We beginnen echter met een andersoortig voorbeeld, een simpele specificatie van gebruikers voor een webserver.

3.1 XML ALS SPECIFICATIETAAL

XML is primair een uitwisselingstaal tussen applicaties. XML wordt echter ook zeer veel gebruikt om specificaties voor één applicatie in uit te drukken en op te slaan. Een voorbeeld is het gebruikersbestand van de Tomcat-webserver (waar u in leereenheid 12 mee zult kennismaken). Een mogelijke inhoud van dit gebruikersbestand, tomcat-users.xml genaamd, luidt aldus:

```
<tomcat-users>
  <user name="tom" password="jerry2501" roles="tomcat" />
  <user name="cat1" password="walk37" roles="role1" />
  <user name="cemanager" password="lim3945" roles="manager" />
  <user name="both" password="tomcat" roles="tomcat, role1" />
</tomcat-users>
```

In dit voorbeeld nemen de specificaties de vorm aan van tweetallen, bestaande uit een attribuutnaam en een (eventueel samengestelde) attribuutwaarde, binnen een eenvoudige hiërarchie van elementen. De rollen corresponderen met bepaalde bevoegdheden. Het XML-bestand koppelt gebruikers aan een of meer rollen en aan een wachtwoord. Opmerking: dit XML-voorbeeld bevat geen XML-declaratie. Zo'n declaratie is gebruikelijk, maar niet verplicht.

Merk op dat de relatie tussen gebruikers (users) en rollen (roles) veel-op-veel is: bij één gebruiker kunnen meerdere rollen horen en één rol kan van toepassing zijn op meerdere gebruikers. In de XML-code wordt dit gerealiseerd door toe te staan dat een roles-attribuut een opsomming is van meerdere rollen, zoals het vierde user-element illustreert.

3.2 MATHML

MathML

MathML is een XML-taal voor mathematische expressies, ontwikkeld in het kader van het XMath-project. Het doel van MathML is het gebruik en hergebruik van mathematische en wetenschappelijke 'content' op het web te bevorderen en te vergemakkelijken. MathML biedt markup voor zowel de semantiek van mathematische expressies (uiteraard!) als de hoogwaardige visuele weergave daarvan.
Als voorbeeld geven we een codefragment met de MathML-expressie voor de kwadratische polynoom $x^2 + 5x + 6$:

```
<apply>
   <plus/>
   <apply>
      <power/>
      <ci>x</ci>
      <cn>2</cn>
   </apply>
   <apply>
      <times/>
      <cn>5</cn>
      <ci>x</ci>
   </apply>
   <cn>6</cn>
</apply>
```

Toelichting
- De 'apply'-tags vormen een soort van haakjes; ook de expressie als geheel staat tussen dergelijke 'haakjes'.
- Het element <plus/> is de opteloperator voor de erop volgende elementen van hetzelfde niveau; twee daarvan staan tussen apply-tags.
- De op te tellen operanden zijn achtereenvolgens: een 'power' (macht) van de variabele x en het getal 2, een 'times' (vermenigvuldiging) van het getal 5 en de variabele x, en tenslotte het getal 6.
- Deze structuur zegt nog niets over de manier waarop de polynoom moet worden weergegeven. Hiervoor kent MathML aparte presentatiemarkup.

De homepage van MathML is de volgende W3C-site: http://www.w3.org/math/.

OPGAVE 1.7

De polynoom $x^2 + 5x + 6$ is equivalent met de productschrijfwijze $(x + 2)(x + 3)$. Schrijf deze tweede vorm in MathML.

3.3 XHTML

XHTML

XHTML is – als XML-versie van HTML – de moderne presentatietaal voor webpagina's. Meer in het algemeen is het de taal voor de gebruikersinterface in XML-gebaseerde informatiesystemen. Hier volgt een voorbeeld van XHTML-code dat ook wordt gebruikt in leereenheid 5 'Presentatie van XML-inhoud'. Het bestand is beschikbaar onder de naam boekCSS.xhtml in de projectmap xml01_perspectief.

```
1   <?xml version="1.0" encoding="iso-8859-1"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3       "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

4   <html xmlns="http://www.w3.org/1999/xhtml">
5     <head>
6       <title>XML Ontwikkeling met Java 2</title>
7       <link href="boek.css" rel="stylesheet" type="text/css" />
8     </head>

9     <body>
10      <h1>XML Ontwikkeling met Java 2</h1>
11      <div class="kaft">
12        <img src="XML-Java2.jpg" alt="boek kaft"/>
13      </div>
14      <table>
15        <tr>
16          <td>Title:</td>
17          <td>XML Development with Java 2</td>
18        </tr>
19        <tr>
20          <td>Door:</td>
21          <td>Michael Daconta et al.</td>
22        </tr>
23        <tr>
24          <td>Kost:</td>
25          <td>44,99</td>
26        </tr>
27        <tr>
```

```
28          <td>Categorie:</td>
29          <td>Java</td>
30        </tr>
31      </table>
32      <h2>Beschrijving</h2>
33      <p>
34        <span class="titel">XML Development with Java 2</span> verstrekt
35        de informatie en de technieken die een ontwikkelaar van Java
36        nodig heeft om XML in Java-gebaseerde toepassingen te integreren.
37      </p>
38      <p>
39        <a href="koop.asp">Koop dit book</a>
40      </p>
41    </body>
42  </html>
```

Zie figuur 1.9 voor een weergave van dit XHTML-document. In zowel code als weergave zijn de kernpresentatieconstructen van XHTML te zien: titels, koppen, paragrafen en tabellen. Door tekst en/of andere media-inhoud in die constructen te plaatsen, maakt men deze inhoud presenteerbaar op het web.

FIGUUR 1.9 XHTML-document weergegeven in Firefox

Toelichting

1 Regels 2 t/m 4 hebben betrekking op de DTD xhtml11.dtd, die de syntaxis bevat van geldige XHTML-bestanden (XHTML versie 1.1). XML-toepassingen die dit XHTML-bestand verwerken kunnen met behulp van een XML-parser controleren of het XHTML-bestand de correcte syntax heeft.

Namespace

2 Het root-element xhtml (zie regel 4 voor de openingstag) heeft het attribuut xmlns met als waarde "http://www.w3.org/1999/xhtml". Deze string duidt een zogenaamde *namespace* aan. De element-tagnamen xhtml, head, body, h1, enzovoort worden door de parser automatisch aan deze string gekoppeld. De eigenlijke elementnamen zijn daardoor samengestelde namen: de namespace-string gevolgd door de naam van de tag, bijvoorbeeld:

```
http://www.w3.org/1999/xhtml:xhtml
http://www.w3.org/1999/xhtml:head
http://www.w3.org/1999/xhtml:body
http://www.w3.org/1999/xhtml:h1
…
```

Bij verwerking van een XML-document door een parser worden de namespace-strings daadwerkelijk gekoppeld aan de tagnamen.
Op deze wijze worden zeer specifieke en onvervreemdbare XHTML-namen gevormd: de XHTML-taal heeft haar eigen unieke woordenschat.

Meer over namespaces leest u in de leereenheden 2 t/m 5.

Namespaces maken globale (wereldwijd unieke) naamgeving mogelijk, wat uitermate belangrijk is op het moderne internet. De namespace http://www.w3.org/1999/xhtml is vanuit dat gezichtpunt niet alleen op te vatten als een string, maar ook als een 'naamruimte', dat wil zeggen een verzameling unieke namen. (Ook attribuutnamen kunnen tot een namespace behoren, al is dat hier niet het geval.)

OPGAVE 1.8

a Open boek.xhtml in een browser.
b Open www.w3.org/TR/xhtml11/DTD/xhtml11.dtd op het net en bekijk de inhoud in vogelvlucht. Probeer het voor uzelf aannemelijk te maken dat hier de syntaxis van toelaatbare XHTML-bestanden wordt vastgelegd.

3.4 SVG

SVG

SVG (*Scalable vector graphics*) is een XML-taal voor het beschrijven van (tweedimensionale) plaatjes die zijn opgebouwd uit vectorgrafiek. Alle moderne browsers ondersteunen SVN. Alleen voor Internet Explorer is een plugin nodig, die gratis beschikbaar is via de Adobe-site (http://www.adobe.com/svg/viewer/install/).

Voorbeeld: cirkel

SVG-voorbeeld

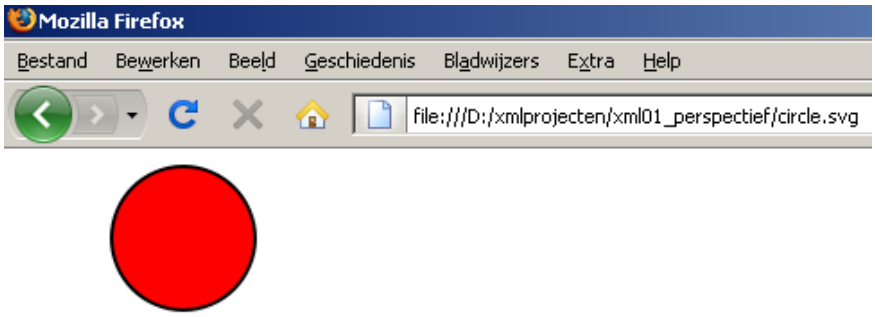
Het volgende XML-bestand (SVG-bestand) bevat de definitie van een cirkel in SVG. Het staat in de projectmap xml01_perspectief onder de naam cirkel.svg.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE svg PUBLIC
3     "-//W3C//DTD SVG 1.1//EN"
4     "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

5   <svg xmlns="http://www.w3.org/2000/svg"
6     width="100%" height="100%" version="1.1">

7     <circle cx="100" cy="50" r="40"
8       stroke="black" stroke-width="2"
9       fill="red"/>

10  </svg>
```

Toelichting
1 Regels 2 t/m 4 hebben betrekking op de DTD svg11.dtd, die de syntaxis bevat van geldige SVG-bestanden (SVG versie 1.1).
2 Regel 5 bevat de openingstag van het root-element svg (ook genoemd in regel 2). De attribuutwaarde http://www.w3.org/2000/svg

correspondeert met de namespace van specifieke SVG-elementnamen (zoals svg en circle).
3 Regels 7 t/m 9 definiëren een cirkel, met pixelcoördinaten x en y, straal r (ook in pixels), de randkleur en -dikte en ten slotte de vulkleur.

Er zijn meerdere browsers die SVG-bestanden kunnen tonen. Figuur 1.10 toont (helaas in zwart-wit) de inhoud van circle.svg in Firefox.



FIGUUR 1.10 SVG-bestand cirkel.svg, weergegeven in Firefox

SVG in HTML-pagina

Aan de weergave van een los SVG-bestand in een browser hebben we meestal niet zoveel. Het is dan ook de bedoeling om SVG te combineren met HTML of XHTML om webpagina's met vectorgrafiek te produceren. Hiervoor zijn verschillende mogelijkheden. We geven twee voorbeelden.

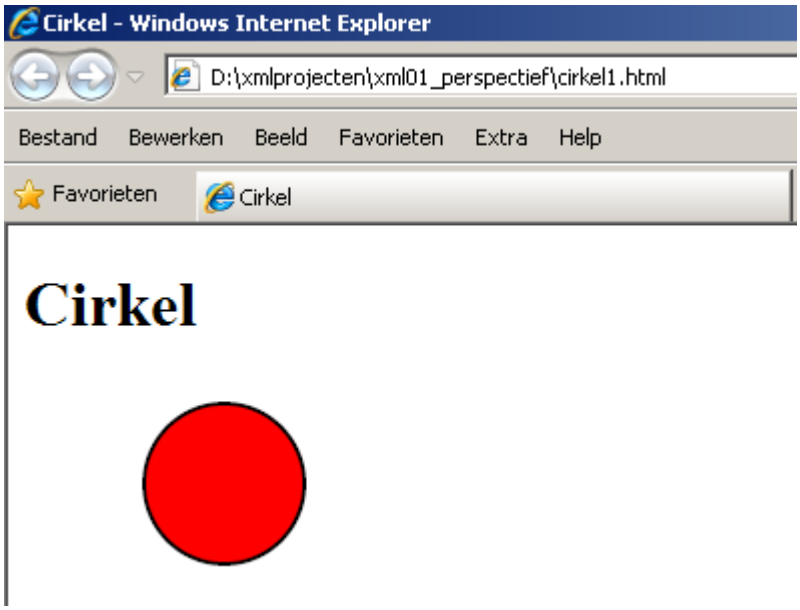
In het eerste voorbeeld nemen we een verwijzing naar cirkel.svg op via het embed-element, dat door de meeste browsers geaccepteerd zal worden:

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <!DOCTYPE html PUBLIC
3        "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG 1.1//EN"
4        "http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg.dtd">
5    <html xmlns="http://www.w3.org/1999/xhtml">
6      <head>
7        <title>Cirkel</title>
8      </head>
9      <body>
10       <h1>Cirkel</h1>

11   <embed src="circle.svg" width="300" height="100"
12       type="image/svg+xml"
13       pluginspage="http://www.adobe.com/svg/viewer/install/" />

14     </body>
15   </html>
```

Dit bestand is in de projectmap te vinden onder de naam cirkel1.html. Zie figuur 1.11 voor een weergave in Internet Explorer.

Toelichting
1 Het embed-element bevat als 'source' een verwijzing naar circle.svg en zorgt ervoor dat op de webpagina hiervoor een ruimte van 300 bij 100 pixels wordt gereserveerd.

2 Voor browsers die SVG niet standaard weergeven (Internet Explorer) wordt de plugin-locatie opgegeven.



FIGUUR 1.11 Een HTML-pagina met tekst en vectorgrafiek, weergegeven in Internet Explorer

Een nadeel van cirkel1.html is dat het geen valide XHTML is. XHTML kent namelijk het embed-element niet. Daarnaast zou het natuurlijk mooier zijn om SVG-code direct met (X)HTML te kunnen combineren. Dit is inderdaad mogelijk, maar niet alle browsers kunnen hiermee overweg. Het nu volgende XHTML-bestand cirkel2.xhtml kunt u openen in bijvoorbeeld Firefox, met hetzelfde resultaat als van figuur 1.11:

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <!DOCTYPE html PUBLIC
3        "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG 1.1//EN"
4        "http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg.dtd">
5    <html xmlns="http://www.w3.org/1999/xhtml">
6      <head>
7        <title>Cirkel</title>
8      </head>
9      <body>
10       <h1>Cirkel</h1>

11       <svg xmlns="http://www.w3.org/2000/svg"
12         width="100%" height="100%" version="1.1">

13         <circle cx="100" cy="50" r="40"
14           stroke="black" stroke-width="2"
15           fill="red"/>

16       </svg>

17     </body>
18   </html>
```

Toelichting
1 Het bestand wordt op regel 4 gevalideerd tegen een DTD-grammatica xhtml-math-svg.dtd. Deze staat toe XHTML te combineren met XMath en SVG.
2 De regels 11 t/m 16 bevatten de SVG-code. Vergelijk deze met die van cirkel.svg.

W3schools

De site 'W3schools' is een W3C-site met tal van zeer toegankelijke tutorials voor alle mogelijke XML-toepassingen. Zo ook voor SVG: zie http://www.w3schools.com/svg/default.asp. Maak hiervan gebruik bij de volgende opgave.

OPGAVE 1.9

a Maak een SVG-document rechthoek.svg en toon dit in een browser naar keuze. Als dat niet lukt, kies dan Firefox of een andere browser.
b Toon de SVG-inhoud vervolgens via een XHTML-document, analoog aan cirkel2.xhtml.

3.5 XML-GEBASEERDE TEKSTVERWERKERS

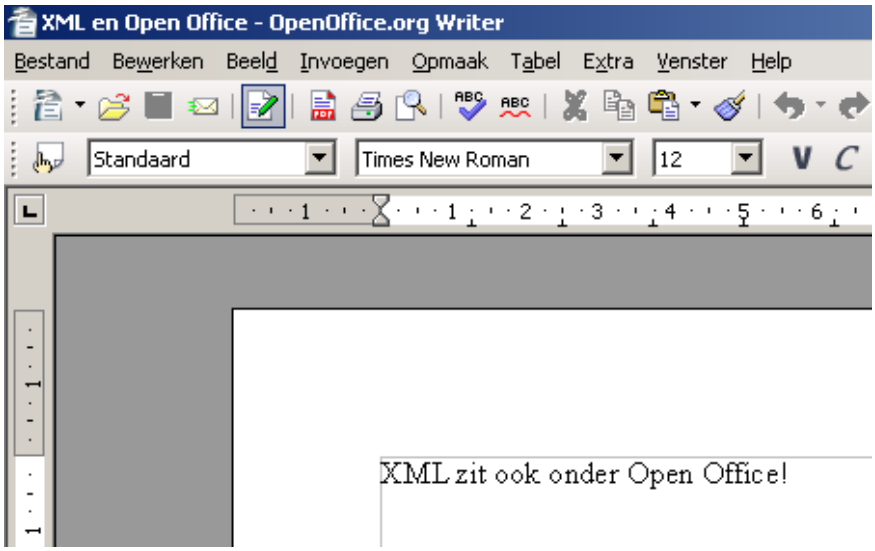
Eenvoudige teksteditors
Teksten gemaakt met een eenvoudige teksteditor (veelal met bestandsextensie txt) worden getoond in een vorm die maar weinig afwijkt van de onderliggende tekenverzameling. Dat komt omdat die onderliggende tekenverzameling is samengesteld uit een vrij klein 'alfabet' van voornamelijk afdrukbare tekens (meestal ASCII). Zijn ingewikkelder constructies nodig (zoals figuren of automatische verwijzingen) of bevat de tekst ook opmaak (zoals een speciaal lettertype of tabellen) dan is een complexere tekstverwerker nodig. De gemaakte bestanden bevatten dan voornamelijk code die niet of slechts voor een klein deel leesbaar is.

Doc-formaat van Word
Beschouwen we als voorbeeld een doc-bestand van Word, dan valt daar voor ons als menselijke lezer weinig aan te begrijpen. De code is grotendeels binair en alleen toegankelijk voor het tekstverwerkende programma (Word) zelf. Dit blijkt bijvoorbeeld wanneer u een doc-bestand opent in een simpele editor zoals Windows Kladblok.

XML-gebaseerd docx-formaat
Het doc-formaat is Word-specifiek en erg ontoegankelijk. Transformaties van en naar Word zijn daardoor niet eenvoudig. Met de grote vlucht die XML heeft genomen, zijn er zoveel nieuwe gereedschappen beschikbaar gekomen voor bewerking, verwerking en transformatie, dat het voordelig bleek om op XML als 'onder water'-formaat over te stappen. In Office 2007 is het doc-formaat vervangen door het op XML gebaseerde docx-formaat. Een docx-bestand is geen XML-bestand maar een archief dat net als bijvoorbeeld een zip-bestand of een Java-archief (jar-bestand) geopend kan worden met programma's zoals WinZip.

Voorbeeld: XML-formaat van Open Office
Voor een voorbeeld nemen we niet Word maar het vrij verkrijgbare Open Office Writer. Writer-bestanden hebben de extensie odt.

De projectmap d:\xmlprojecten\xml01_perspectief bevat het Writer-bestand XML_en_Open_Office.odt. Figuur 1.12 toont dit bestand geopend in Open Office Writer.



FIGUUR 1.12 Weergave van Writer-document (Open Office)

Het bestand XML_en_Open_Office.odt is net als een docx-bestand van Word een archief. Wanneer u dit archief opent (in bijvoorbeeld WinZip), ziet u onder andere het XML-bestand content.xml. We tonen een deel van de inhoud hiervan, met de in figuur 1.12 zichtbare tekst gemarkeerd:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <office:document-content
3      xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
4      xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"...>
       ...
5      <office:body>
6         <office:text>
          ...
7         <text:p  ... >XML zit ook onder Open Office!</text:p>
8         </office:text>
9      </office:body>
10  </office:document-content>
```

Toelichting
1 De woordenschat van dit XML-Office-document bevat nu woorden uit verschillende namespaces. In het codefragment zijn woorden uit twee namespaces zichtbaar:
– de woorden document-content (regels 2 en 10), body (regels 5 en 9) en text (regels 6 en 8) behoren tot de namespace "urn:oasis:names:tc:opendocument:xmlns:office:1.0"
– het woord p (twee keer op regel 7) hoort tot de namespace "urn:oasis:names:tc:opendocument:xmlns:text:1.0".

Alias

2 De koppeling tussen de twee typen XML-Office-woorden en de namespaces (de strings op de regels 3 en 4) vindt plaats via de *aliassen* office en text (gevolgd door een dubbele punt).

Dit is een wat andere manier om aan namespaces gekoppelde namen te vormen dan in de paragrafen 3.3 (XHTML) en 3.4 (SVG), waarbij maar één namespace voorkwam.
Het volledige XML-Officedocument bevat nog meer namespaces met bijbehorende aliassen.

Opmerkingen
– Het codefragment bevat zowel een alias 'text' als een Office-woord 'text'. Deze hebben niets met elkaar te maken. De alias staat vóór een dubbele punt, de Office-naam achter een dubbele punt.
– Zonder verandering van betekenis kan een alias vervangen worden door een andere alias. Het is immers maar een koppelwoordje. De alias 'text' kan dus in het hele document (o.a. op de regels 4 en 7) vervangen worden door bijvoorbeeld xyz zonder dat er iets wezenlijks verandert. Dit geeft nog eens het verschil aan met het Office-woord 'text' op de regels 6 en 9.

OPGAVE 1.10

Open XML_en_Open_Office.odt in WinZip of een vergelijkbaar programma en bekijk vervolgens content.xml. Uiteraard kunt u het bestand ook als eindgebruiker openen in (het vrij verkrijgbare) Open Office Witer zelf.
Als alternatief kunt u – indien u over Word 2008 beschikt – op een vergelijkbare manier het 'onder water'-formaat van een docx-bestand onderzoeken.

TERUGKOPPELING

Uitwerking van de opgaven

1.1 t/m 1.4 Geen uitwerking

1.5 b Beknopte toelichting op de code:
- de eerste regel is de XML-declaratie, verplicht omdat de stylesheet zelf een XML-document is
- de tweede en de laatste regel bevatten de openings- en sluittag van het root-element van deze stylesheet zelf; ze zeggen zoiets als 'dit is een stylesheet'
- de derde regel zegt dat de transformatie uitgaat van het element bestelling in het brondocument
- het stylesheet bevat letterlijke stukjes HTML-code (zoals <html><body>, Klant en <p/>)
- in de regels met het element xsl:value-of wordt via het select-attribuut een waarde uit het brondocument (bestelling.xml) gelezen.

1.6 Geen uitwerking

1.7 Het product $(x + 2)(x + 3)$ in MathML:

```
<apply>
   <times/)
   <apply>
      <plus/>
      <ci>x</ci>
      <cn>2</cn>
   </apply>
   <apply>
      <plus/>
      <ci>x</ci>
      <cn>3</cn>
   </apply>
</apply>
```

1.8 t/m 1.11 Geen uitwerking.