IK WIL

XML Introductie

vijfhart
IT-OPLEIDINGEN

Introductie

- Voorstelrondje
- Agenda
- Dagindeling
- Cursusmateriaal
- Doel van deze cursus

Agenda

Dag 1	H1	Inleiding
	H2	XML Documenten
	H3	XML in de praktijk
	H4	DTD's en XML Schema
Dag 2	H5	Transformatie en opmaak
	H6	XML documenten verwerken

Dagindeling

- 8:45 -..... Begin cursusdag
  - Introductie
  - .....
  - Hoofdstuk
    - theorie
    - opdrachten maken
    - opdrachten bespreken
- 10:30 -10:45 Koffiepauze
- 12:00 -12:45 Lunchpauze
- 14:30 -14:45 Koffiepauze
- ..... -16:00 Einde cursusdag

Cursusmateriaal

- Map met sheets
- Cursistenmap
  - Samenvatting theorie
  - Bijlagen
  - Opdrachten
- Computer
  - XML editor (XMLBlueprint)
  - Cursusbestanden

Doel van deze cursus

- In deze cursus leert u wat XML is en waar u het tegen kunt komen
- Hoe u XML in de praktijk kunt toepassen
- Hoe XML documenten worden gevalideerd met DTD's en schema's
- Hoe XML documenten worden getransformeerd

Hoofdstuk 1 Inleiding

Leerdoelen:

Begrijpen wat XML is

Weten waarvoor XML wordt gebruikt

De belangrijkste verschillen tussen HTML en XML kunnen aangeven

Weten wat een XML-editor is

Weten wat een parser is

Inleiding

- We willen tegenwoordig al onze gegevens opslaan
- Het opslaan kan op veel verschillende manieren
  - Databases
  - Bestanden in verschillende bestandsformaten
- Gegevens moeten regelmatig met anderen worden uitgewisseld
- De vele vormen van opslag kunnen hierbij voor problemen zorgen
- Tekst kan eenvoudig worden verzonden en gelezen
- Met XML worden gegevens als tekst opgeslagen
- Dit maakt XML erg geschikt voor het uitwisselen van gegevens

SGML

- XML afgeleid van SGML (Standard Generalized Markup Language)
- SGML bestaat uit een hiërarchische structuur van elementen
- Deze elementen worden gemarkeerd met tags
- Tags worden omsloten door < >
  - <tagnaam>inhoud</tagnaam>
- Regels hiervoor zijn als standaard vastgelegd door W3C
- W3C is World Wide Web Consortium
- SGML kent verschillende subsets met striktere regels o.a.:
  - HTML en XHTML beschrijven van webpagina's
  - XML (beschrijven van gegevens
  - SVG en VRLM beschrijving van vector en 3D figuren
  - SOAP WSDL beschrijven communicatie protocol
  - MathML beschrijven van wiskundige symbolen en formules
  - …

Wat is XML

- XML is een gemoderniseerde en vereenvoudigde versie van SGML
- In XML worden gegevens gemarkeerd volgens bepaalde regels
- Deze regels zijn strikter dan bij SGML
- XML is de afkorting van: Extensible Markup Language
  - Extensible: uitbreidbaar met nieuwe "eigen" vocabulaire
  - Markup: Markeren van elementen uit een document (vb. HTML)
- XML tags zijn zelf beschrijvend
- XML is uitbreidbaar met eigen tags
- XML is tekst dus platformonafhankelijk
- Voorbeeld:

```
<cursus>
  <naam>XML introductie</naam>
  <dagen>2</dagen>
</cursus>
```

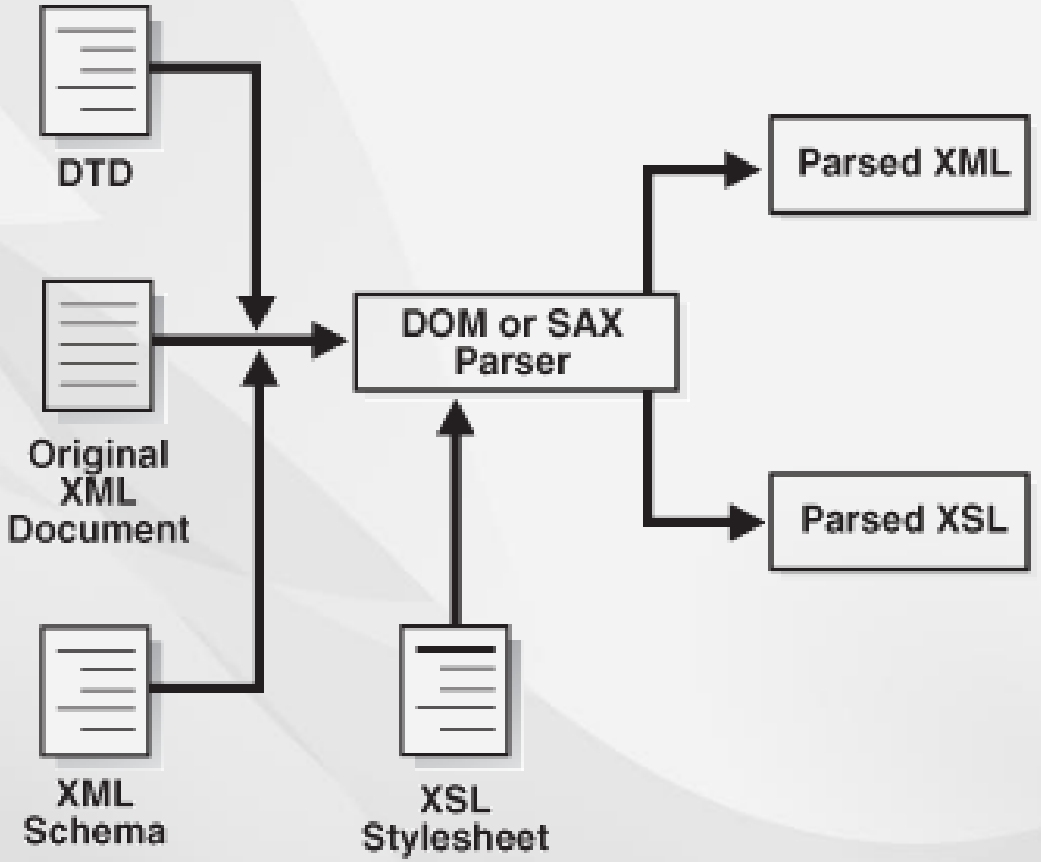
XML begrippen

- XML Editor: tekstverwerker voor XML, bv. XMLBluePrint, XMLSpy
- XML Validator: valideren van XML op basis van:
  - DTD: Document Type Definition
  - XSD: XML Schema Definition
  - Online (syntax): http://www.w3schools.com/xml/xml_validator.asp

XML begrippen

- XML Parser: vertalen van XML naar leesbaar formaat
  - Validerende parser of non-validerende parser
  - DOM: Document Object Model op basis van compleet document
  - SAX: Streaming, gebaseerd op events, gedeeltelijk
- Taalondersteuning voor XML: o.a. voor XML processing en binding
  - JAXP: Java XML Processing API (o.a. DOM en SAX)
  - JAXB: Java XML Binding API (XML <-> Java object)
  - JAX-WS: Java Web Services (XML/SOAP bericht <-> Java methode)
  - SQL binnen verschillende databases
  - …

XML Parser



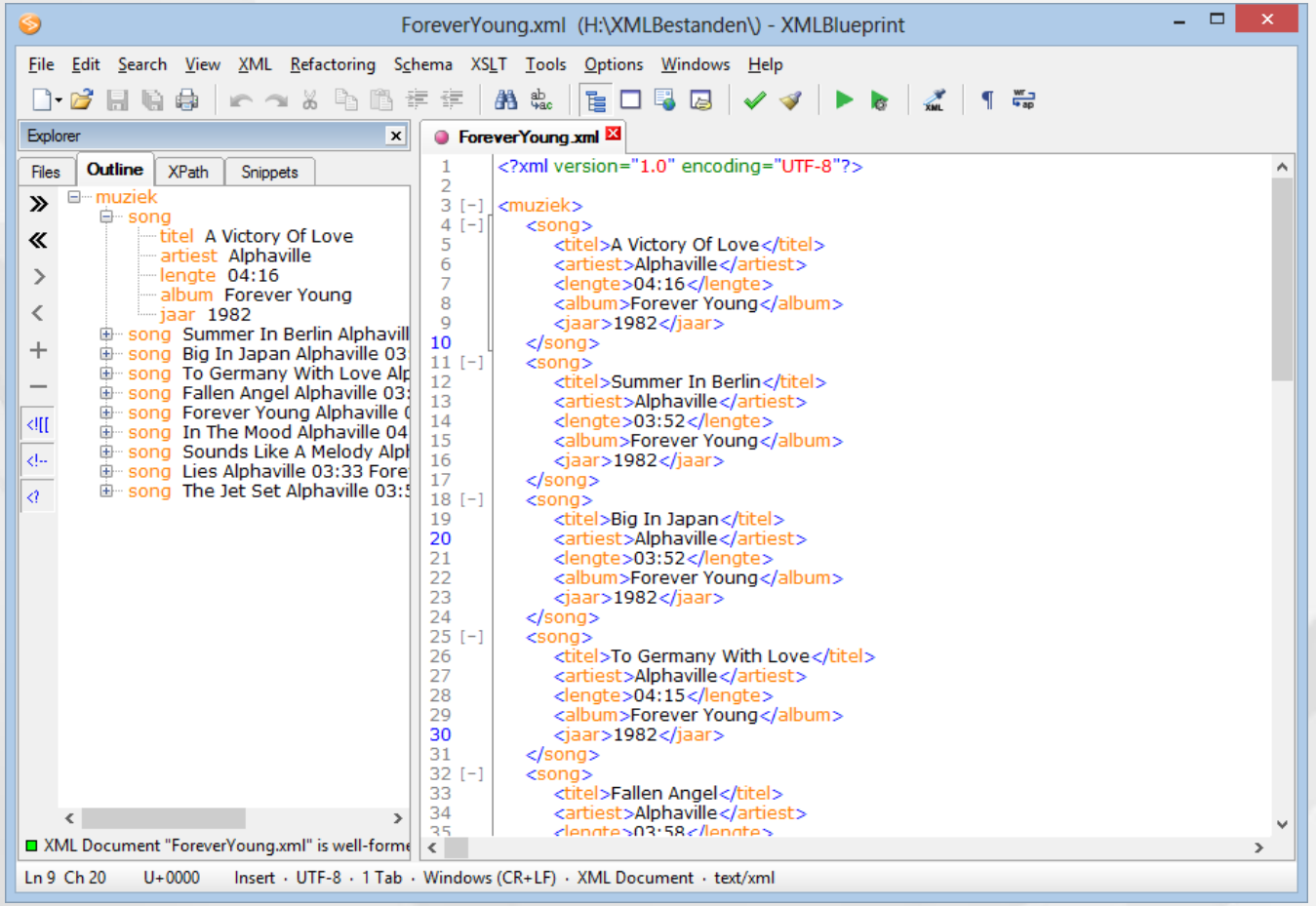
XML editor

- XML documenten maken of wijzigen kan in elke teksteditor
  - Notepad
  - Word
- Maar en specifieke XML editor biedt veel meer mogelijkheden
  - Eenvoudige Opmaak
  - XML documenten controleren
  - XML documenten transformeren
- Er zijn veel XML editors met elk hun eigen mogelijkheden
  - Wij gebruiken tijdens deze cursus XMLBlueprint

XML Blueprint

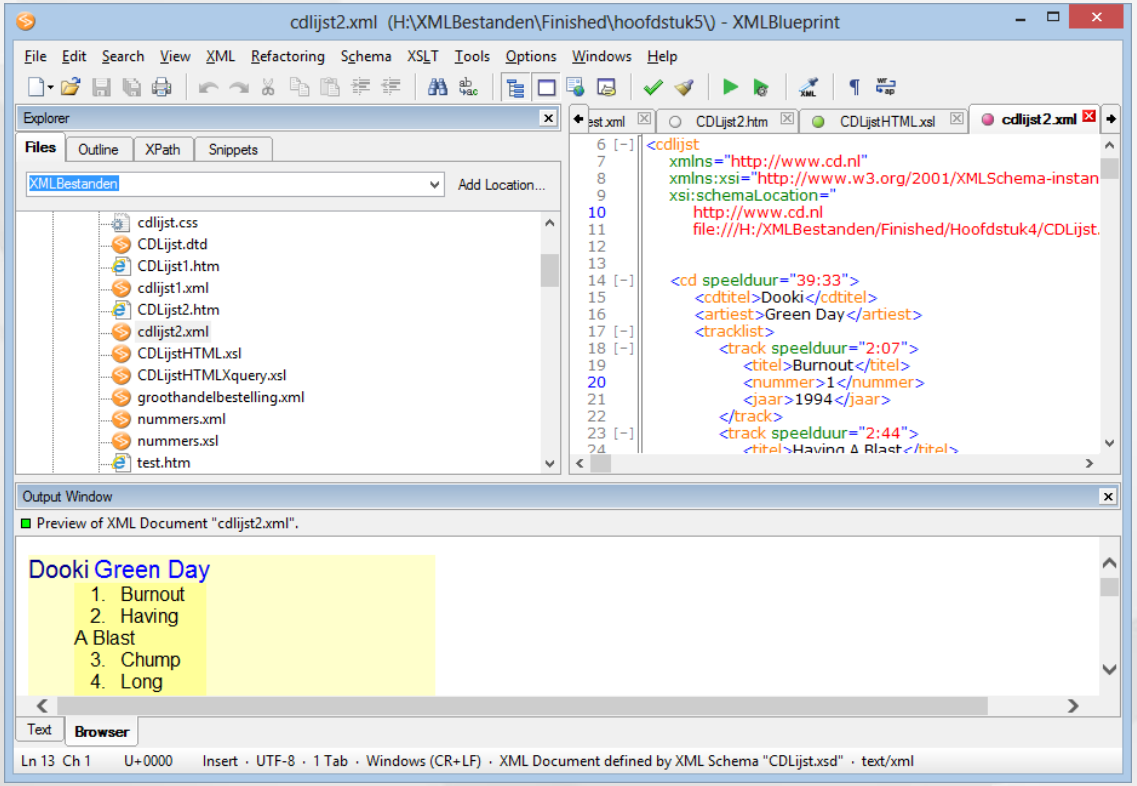
- XML Blueprint is een uitgebreide editor
- Zo kunnen we o.a.
  - Met een klik XML in een overzichtelijke structuur tonen
  - De XML structuur controleren (is het geldige XML)
  - XML documenten valideren (voldoet het aan de voorwaarden)
  - XML omzetten in ander document (XSLT-transformatie)
  - XPath expressies evalueren

XMLBlueprint



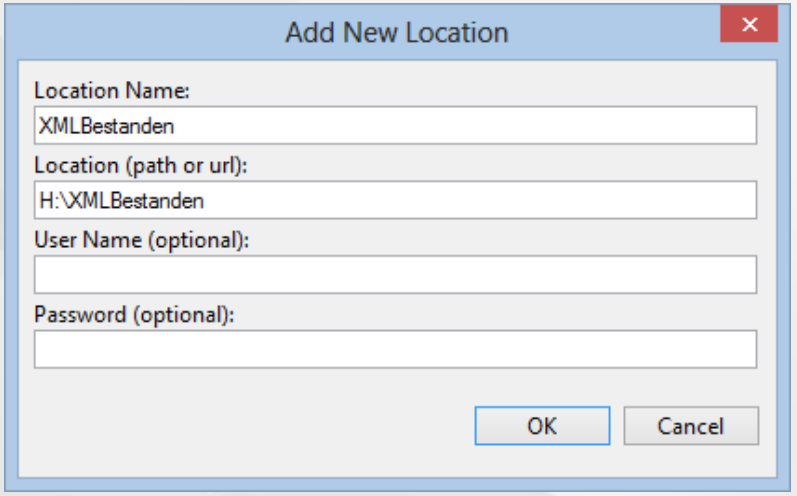
XMLBlueprint

- Output op twee tabs Tekst en Browser
- zichtbaar maken met Preview in Output Window 



XMLBlueprint

- Opdracht
  - Start XMLBlueprint
  - Klik bij Explorer op de knop op Add Location (tabblad Files)
  - Vul het venster in als volgt in en klik op OK



  - Klik in het invul vak en kies XMLBestanden uit de lijst
  - De oefenbestanden zijn nu te openen van uit Explorer

Hoofdstuk 2 XML Documenten

Leerdoelen:

De structuur van een XML document herkennen

Weten wat tags, elementen en attributen zijn

Een eenvoudig well-formed XML document kunnen maken

XML documenten

- Een XML documenten is opgebouwd uit tags
- Deze tags kunnen inhoud (gegevens) of andere tags bevatten
- Voorbeeld

```
<muziek>
      <song>
          <titel>Summer In Berlin</titel>
          <artiest>Alphaville</artiest>
          <lengte>03:52</lengte>
          <album>Forever Young</album>
          <jaar>1982</jaar>
      </song>
      <song>
          <titel>Big In Japan </titel>
          <artiest>Alphaville</artiest>
          <lengte>03:52</lengte>
          <album>Forever Young</album>
          <jaar>1982</jaar>
      </song>
</muziek>
```

XML Elementen en Tags

- Een tag is code tussen een < en > teken
- De begintag <…> en eindtag </…> vormen een element
- De inhoud van het element staat tussen de begin- en eindtag
- Begin- en eindtag hebben dezelfde naam (dit is ook case sensitive!)
- Voorbeeld:
  - `<cursus>XML Introductie</cursus>`
- Een element mag ook leeg mag zijn
  - `<cursus></cursus>`
  - `<cursus/>`

XML Elementen en Tags

- De inhoud van een element kan bestaan uit tekst of elementen
- Tags kunnen worden genest:

```
<cursus>  <!-- dit is het root element en de parent -->
  <naam>XML introductie</naam>  <!-- dit is een child, een sibling -->
  <dagen>2</dagen>
</cursus>
```

- Ook child elementen kunnen weer geneste elementen bevatten
- Een element is een child van slechts één element (parent element)
- Elke element heeft een parent element behalve de bovenste
- Dit bovenste element wordt wel *root* element genoemd

XML Elementen en Tags

- De namen van tags mogen zelf bedacht worden
- Tags kunnen een semantische betekenis hebben
  - Ze vormen een koppeling met de echte wereld
  - Mensen begrijpen wat er bedoeld wordt
- Voorbeeld

```
<naam>
    <voornaam>Michiel</voornaam>
    <tussenvoegsel>de</tussenvoegsel>
    <achternaam>Vries</achternaam>
 </naam>
```

- Tags kunnen ook betekenis hebben voor de applicatie
  - Uit de naam is het soort gegeven niet meer duidelijk te herleiden
  - Voor een applicatie is de semantische betekenis niet van belang
- Voorbeeld

```
<p><n1>Michiel></n1><n2>de</n2><n3>Vries</n3> </p>
```

Leeg of afwezig?

- Een element kan leeg of afwezig zijn
- Overwegingen
  - verwacht de applicatie het element?
  - is het optioneel of nu even geen inhoud?
- *Leeg:*

```
<naam>
   <voornaam></voornaam>
   <achternaam>Smits</achternaam>
</naam>
```

- *Afwezig:*

```
<naam>
   <achternaam>Smits</achternaam>
</naam>
```

- Afhankelijk van wat er gevraagd wordt

Lege elementen

- Een leeg element wordt op verschillende manieren weer gegeven
- Alleen een begin en eind tag zonder inhoud
  - Wordt meestal gebruikt voor elementen die soms leeg zijn
  - `<voornaam></voornaam>`
- Er is één tag die begin en eind tag tegelijk is
  - Wordt meestal gebruikt voor elementen die altijd leeg zijn
  - `<gehuwd/>`

Attributen

- Het is ook mogelijk om aan een element attributen toe te voegen
  - `<tag attribuut1="waarde1"/>`
- Een attribuut is een eigenschap van een element
- Attributen zijn tekst waarden net als de tekstinhoud van elementen
- Meerdere attributen zijn mogelijk
- Voorbeeld:
  - `<cursus naam="XML" jaar="2014"></cursus>`

Attributen

- Een attribuut kan alleen een tekstwaarde bevatten
  - Een attribuut kan dus niet genest worden
- Een attribuut bevat meestal meta-gegevens voor het element
  - Het attribuut voegt niets toe aan het element zelf
- Een attribuutnaam mag maar één keer in het element voorkomen
  - Een attribuut is uniek per element

Attributen of elementen?

- Wanneer is een gegeven een attribuut en wanneer een child element
- Vooral van belang voor de leesbaarheid (semantische betekenis)
- Afwegingen die hierin een rol spelen
  - Gegevens die mede bepalen wat het element precies is, worden als children opgenomen
  - Beschrijvende (meta gegevens) worden opgenomen als attributen
  - Als de gegevens bedoeld zijn om te worden gelezen door een persoon, gebruik dan elementen i.p.v. attributen.
  - Neem eigenschappen van een element op als attributen
- Soms is het best lastig om hierin een keuze te maken
- Bedenk dat het niet veel verschil maakt alleen
  - Maar het is wel van belang voor de afhandeling van het document

Attributen of elementen?

- De prijs van volkorenbrood kan als element of attribuut worden opgenomen

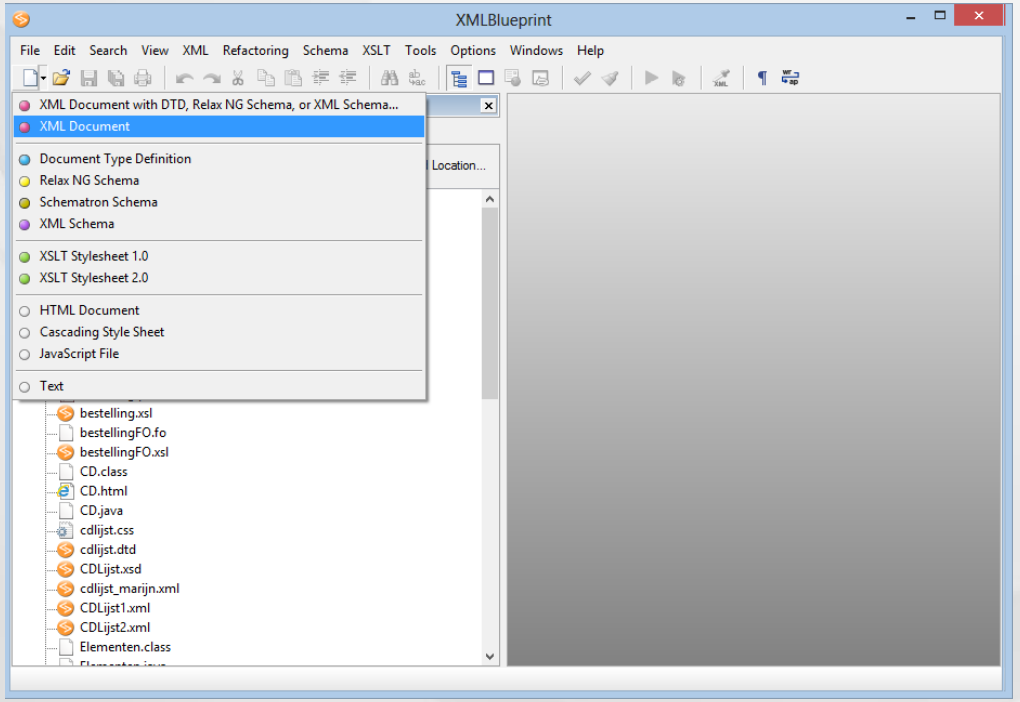
```
<product prijs="1.80" eenheid="euro">
    <naam>volkorenbrood</naam>
</product>
```

```
<product>
    <naam>volkorenbrood</naam>
    <prijs eenheid="euro">1.80</prijs>
</product>
```

- De tweede manier heeft de voorkeur
  - Prijs en eenheid zijn geen eigenschappen van het product
  - De prijs wordt daarom een child element
  - De eenheid is wel een eigenschap van de prijs
  - De eenheid wordt daarom een attribuut van prijs

XML documenten maken

- Een XML document is een tekst document met de extensie .xml
- Kan in elke tekst editor gemaakt worden
- In XMLBlueprint gebruiken we de knop New File



XML documenten maken

- Het nieuwe XML document bevat al één regel
- Een tag die begint met <? En eindigt met ?>
- De wordt de processing instruction (PI) genoemd
  - <?xml version="1.0" encoding="UTF-8"?>
  - De PI is wel een tag maar geen element
  - Hiermee wordt aangegeven wat voor document het is
  - Is vooral bedoeld voor verwerking in applicaties
  - De attributen version en encoding zijn verplicht
    - version geeft de XML versie aan is eigenlijk altijd 1.0
    - encoding definieert de karakterset
- De PI is niet noodzakelijk
- Het is wel aan te raden om een PI op te nemen
- Niet elk document met tags is correct XML
- Het document moet aan regels voldoen, Well-formed zijn

Well-formed XML

- Well-formed XML bevat:
  - Minimaal en maximaal één root element
    - Andere elementen zijn gedefinieerd als childs van de root
  - Elementen bestaan uit een begin en een eind tag
  - Correct geneste structuur
  - Attribuutwaarden opgenomen binnen dubbele quotes
    `<cursus naam="XML"></cursus>`
- Let op:
  - Tags zijn hoofdlettergevoelig, idealiter kleine letters
  - Tags bevatten geen `!"#$%&'()*+,/;<=>?@[\]^`{|}~`, of white space
  - Tags mogen niet beginnen met de letters xml, een punt of een getal
  - Tags mogen geen speciale tekens bevatten zoals &, en <, of > behalve wanneer speciale betekenis gewenst is.

Correct genest

- Een internet browser kan omgaan met verkeerd geneste tags,
- Het onderstaand is bijvoorbeeld toegestaan:
  - `<i><b>Cursief en vetgedrukt</i></b>`

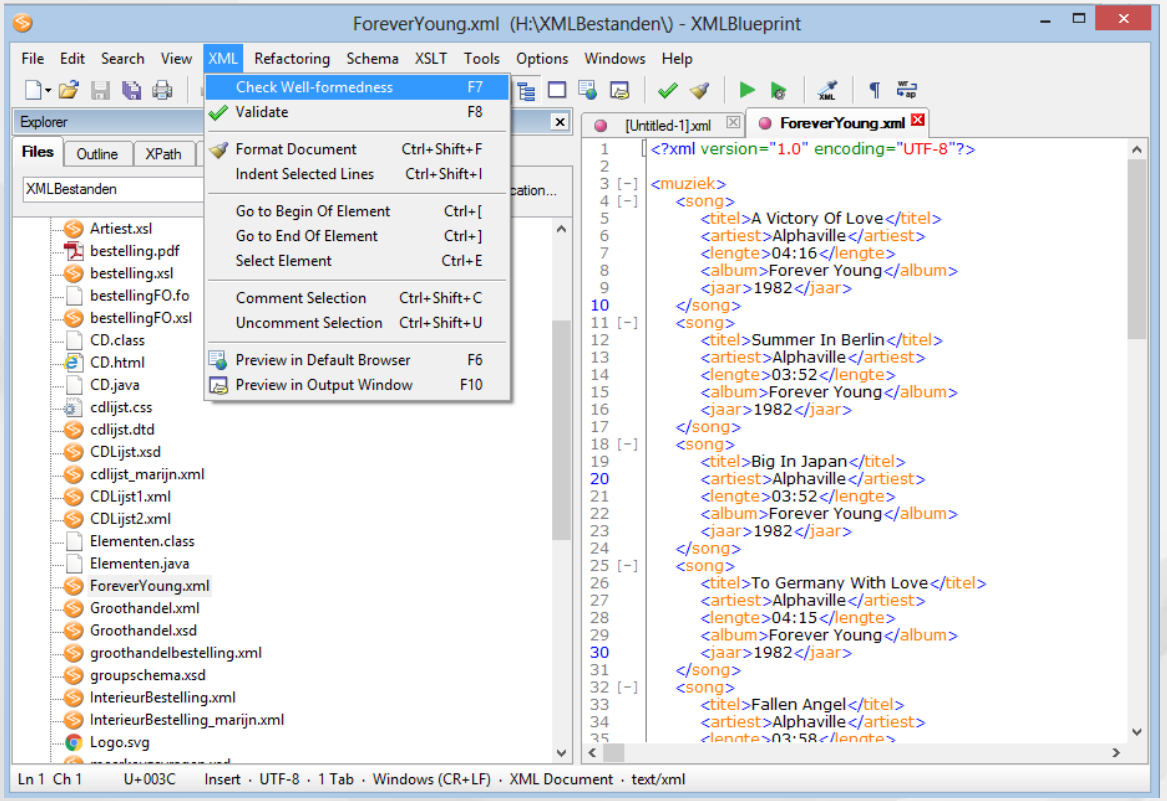
- In XML is dit niet toegestaan:
- het ene element moet geheel binnen, of geheel buiten het andere element staan, dus:
  - `<i><b>Cursief en vetgedrukt</b></i>`

    of:

    `<b><i>Cursief en vetgedrukt</i></b>`

Well-formed XML

- XMLBlueprint kan controleren of een document Well-formed is
  - Via het menu XML of met de toets [F7]



XML met alleen elementen of tekst

- XML beschrijft vaak strikt gestructureerde gegevens
- In dat geval bevat een element tekst OF een of meer andere elementen
- Voorbeeld: gegenereerde XML content uit een database of een applicatie
- Voorbeeld

```
<persoon>
  <naam>
     <voornaam>Donald</voornaam>
     <achternaam>Duck</achternaam>
  </naam>
</persoon>
```

XML met mixed content

- XML wordt ook gebruikt als hulpmiddel om delen van een document nader te specificeren.
- De inhoud van het document wordt verder meestal in natuurlijke taal geformuleerd: de structuur is niet eenduidig vastgelegd.
- Een XML element kan hierbij zowel tekst als elementen bevatten.
  - Dit wordt wel mixed content genoemd
- Voorbeeld: brieven, beschrijving van een verzekering, etc.

```
<brief>
<aanhef>Geachte lezer</aanhef>,
Dit is een voorbeeld van XML met <onderwerp>mixed
content</onderwerp>.
fijn dat u dit wilt lezen.

<ondertekening>Hoogachtend,
D. Duck</ondertekening>
</brief>
```

Speciale tags en karakters

- Er zijn een aantal speciale XML tags
  - XML Processing Instructie hebben we al gezien

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

  - XML Commentaar

```
<!-- Commentaar, alles hiertussen wordt genegeerd -->
```

- Commentaar wordt gebruikt om extra informatie
  - Tekst die niet als gegevens hoeft worden gezien

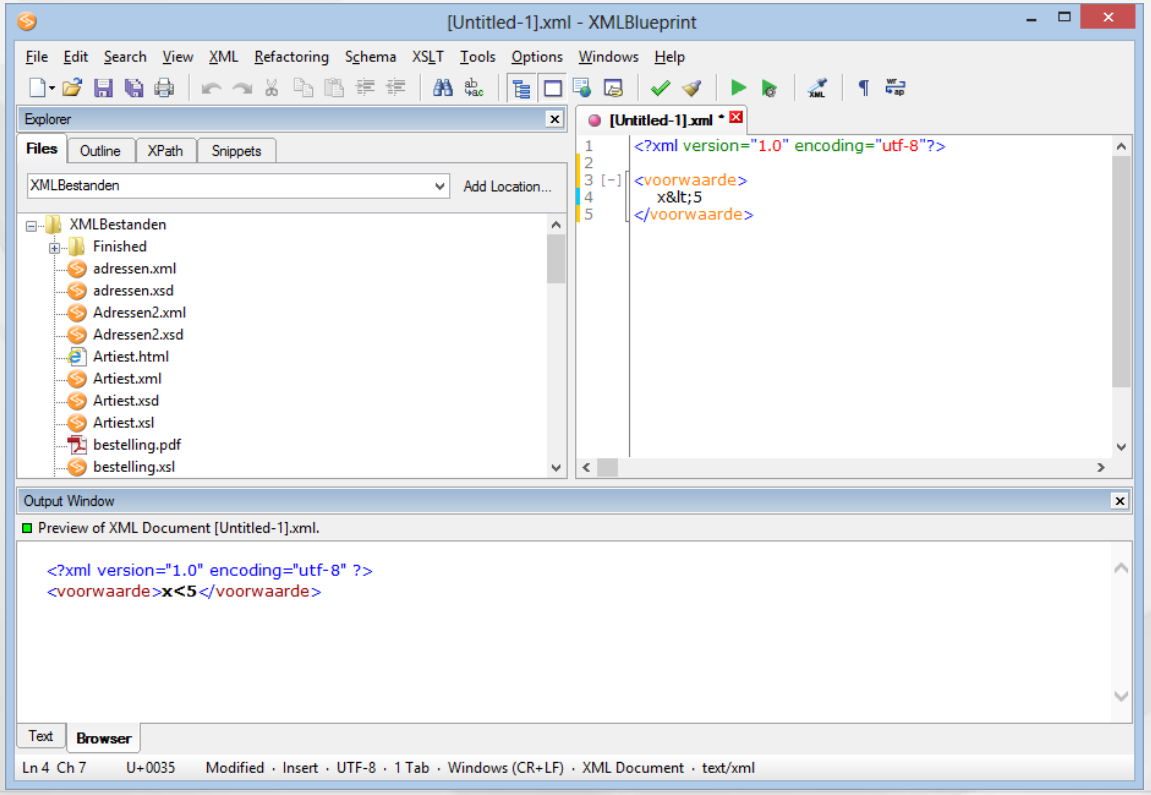
Speciale karakters

- Een aantal speciale karakters kunnen in niet worden gebruikt XML
  - Namen van Tags en Attributen
- In de inhoud van elementen en attributen kunnen wel speciale tekens wel voorkomen
- Om deze tekens op te nemen zijn speciale notaties noodzakelijk
  - Entity: tekstweergave van een speciaal karakter

Karakter	Entity
<	&lt;
>	&gt;
'	&apos;
"	&quot;
&	&amp;

Speciale karakters

- In de tekst x <5 wordt he teken < gezien als het begin van een tag
- Dit is op te lossen door < te gebruiken



CDATA

- Andere mogelijkheid is gebruik maken van CDATA
- CDATA staat voor Character Data
- Alle gegevens binnen een CDATA element worden als tekst gezien, niet als gestructureerde inhoud
- Syntax:

```
<![CDATA[ tekst ]]>
```

- Voorbeeld

```
<![CDATA[ tekst met speciale karakters zoals &, <, >, ', en "
worden gezien als tekst en genegeerd ]]>
```

- CDATA wordt vaak gebruikt om code op te nemen, zoals javascript
- CDATA kan niet gebruikt worden bij attributen

Opdrachten Hoofdstuk 2 (succes)

Hoofdstuk 3 XML in de praktijk

Leerdoelen:

Ontdekken hoe XML in de praktijk gebruikt wordt

Verschillende methoden van verwerking kunnen onderscheiden

XML in de praktijk

- Het internet heeft het mogelijk gemaakt meer informatie te delen
- Vroeger vooral als "vraagbaak" t.b.v. de consument
- Nu ook (commercieel) als middel om informatie te delen
- Delen van informatie wordt belangrijker om te kunnen concurreren
- Applicaties van verschillende bedrijven zijn vaak niet compatible
- Gebruik daarom een taal die alle applicaties kunnen lezen: XML
- Wel zijn er vaak domein specifieke transformaties nodig (XSLT)

XML in de praktijk

- Elke organisatie heeft zijn eigen XML met eigen tags en structuren
- Dit is vaak niet door anderen te verwerken
- Daarom is er een vertaling (transformatie) met een stylesheet nodig
- Ook zijn er een aantal XML vocabularies ontwikkeld
- XML vocabularies XML als Domain Specific Language (DSL)
  - MathML: wiskunde
  - ebXML: electronic banking XML
  - MusicXML:bladmuziek
  - HL7:Health Level Seven
  - Open Office XML: office applicaties
  - Legal XML: juridisch domein
  - OSB Koppelvlak Standaard: Overheid Service Bus

XML in de praktijk

- XML als communicatiemiddel
  - Web services
  - Transformatie door middel van XSLT
- XML voor presentatie
  - XML in combinatie met Cascading Style Sheets (CSS)
  - XML in combinatie met XSLT Style Sheets
  - Scalable Vector Graphics (SVG)

XML als communicatiemiddel

- XML wordt veelvuldig toegepast binnen web services
- Het standaard berichtformaat voor web services is een envelop structuur in XML
- Applicaties geschreven in verschillende talen kunnen zo met elkaar communiceren
- De SOAP envelop structuur biedt ondersteuning voor adressering

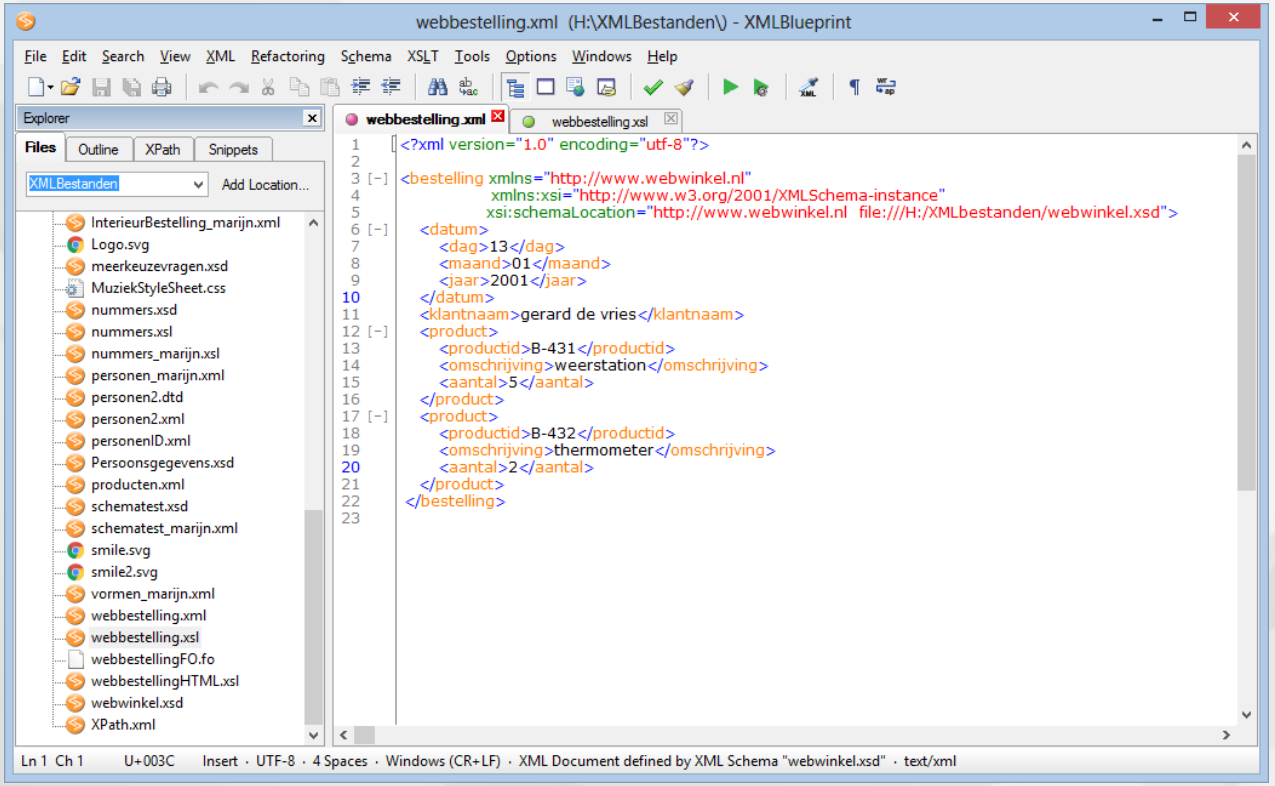
```
<?xml version="1.0"?>

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>Geboekte cursus</soap:Header>
  <soap:Body>
    <curs:getPrijs xmlns:curs="http://www.vijfhart.nl/cursussen">
      <curs:cursus>XML Introductie</curs:cursus>
    </curs:getPrijs>
  </soap:Body>
</soap:Envelope>
```

XML als communicatiemiddel

- Elke organisatie heeft eigen XML die niet uitwisselbaar is
- Er is een transformatie met een XSLT Stylesheet nodig
- Een XSLT Stylesheet vertaalt de tags van de zender naar tags die de ontvanger begrijpt
- *Extensible Stylesheet Language Transformations*
- Transformaties naar verschillende formats mogelijk zoals
  - XML
  - HTML
  - PDF
- Dit kan eenvoudig met XMLBlueprint

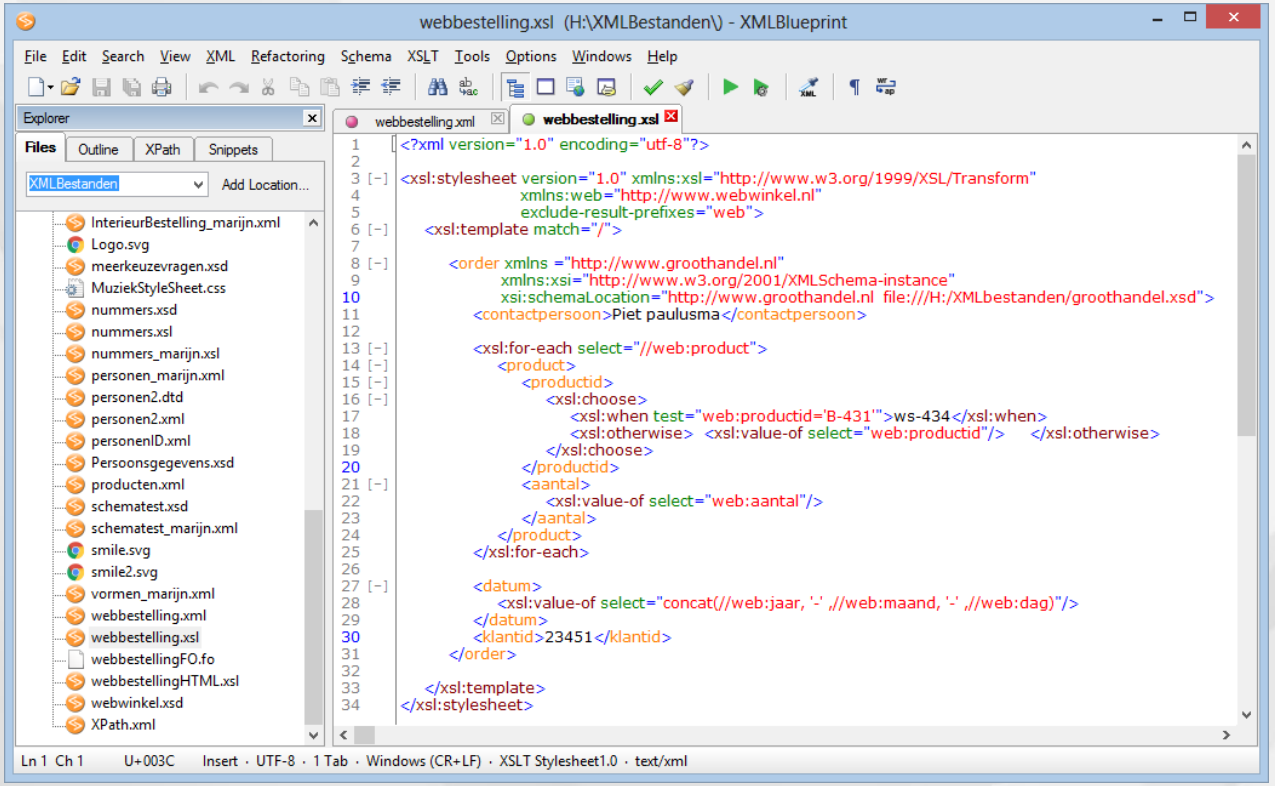
XML stylesheets

- Het document webbestelling.xml moet verzonden worden
- De groothandel gebruikt andere xml tags



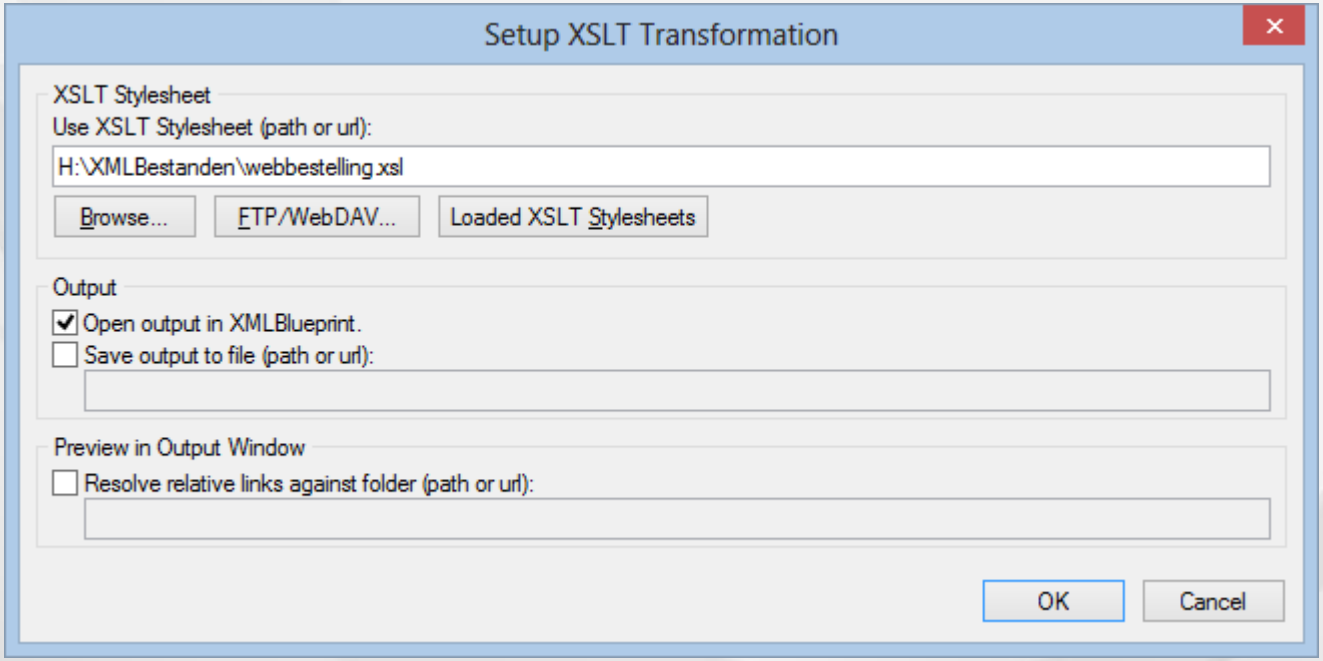
XML stylesheets

- Het stylesheet webbestelling.xsl transformeert onze xml naar een xml document dat de groothandel kan begrijpen



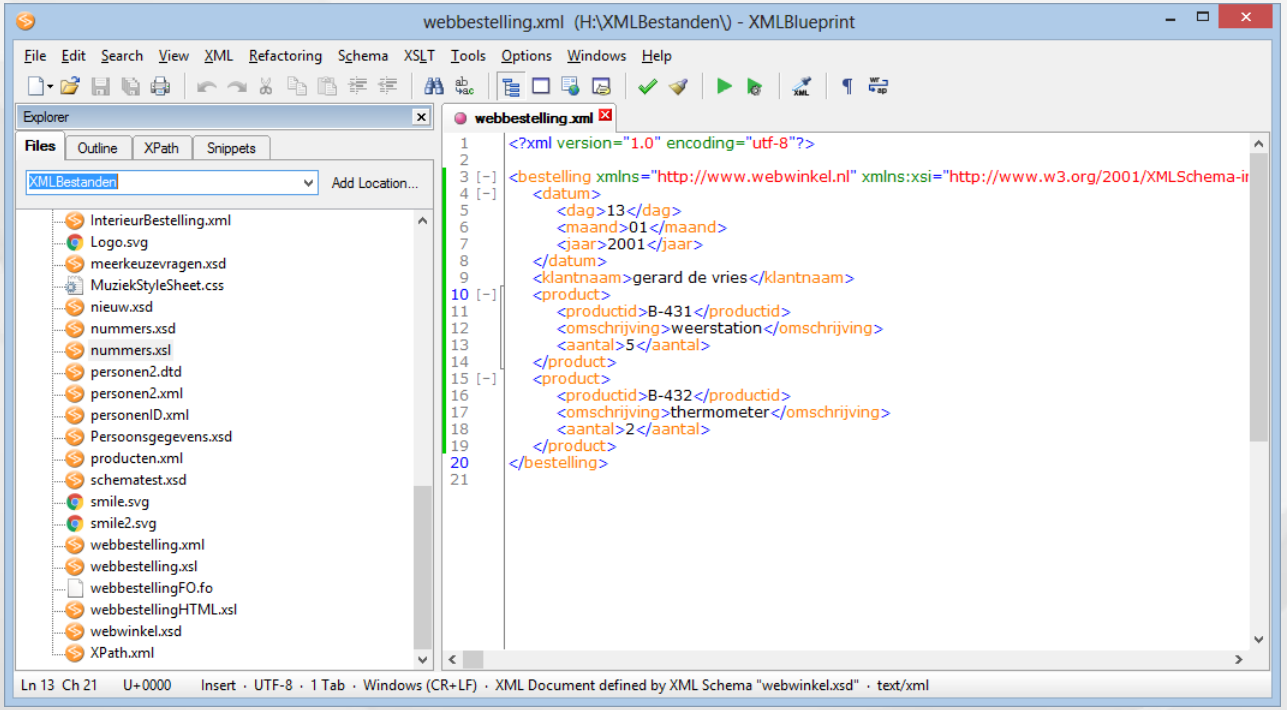
XML stylesheets

- Transformatie uit te voeren met de knop Run XSLT transformatie



XML stylesheets

- De uitvoer is een nieuw XML document
- Leesbare opmaak met de knop Document Format



DEMO

XML stylesheets

- Met behulp van tags zijn transformaties te automatiseren
- Bijvoorbeeld.
  - Koppelen van een XSLT stylesheet voor opmaak in browser:

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

  - Koppelen van een CSS stylesheet (bovenaan in xml bestand):

```
<?xml-stylesheet type="tekst/css" href ="vijfhartstylesheet.css"?>
```

  - Start van stylesheet zelf:

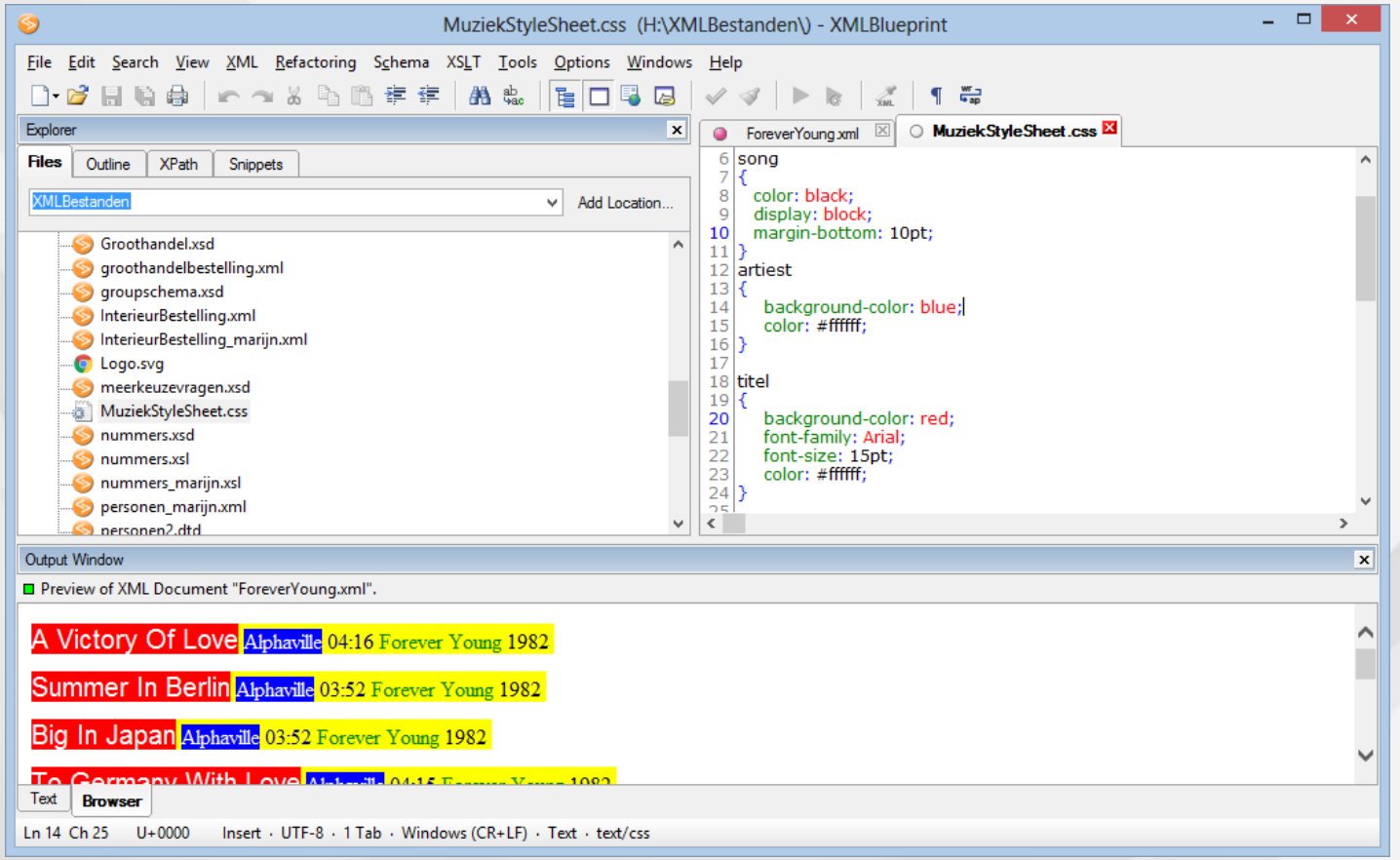
```
<xsl:stylesheet mlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:opld="http://www.vijfhart.nl">[...]</xsl:stylesheet>
```

- Transformaties worden in hoofdstuk 5 uitgebreid behandeld

CascadingStyleSheets (CSS)

- Met CascadingStyleSheets (CSS) documenten voorzien van opmaak
- Dit wordt veel gebruikt binnen HTML
- In een CSS wordt de opmaak van HTML elementen gedefinieerd
- CSS kan ook gebruikt worden voor de opmaak van XML tags
- In het CSS wordt per element een opmaak gedefinieerd zoals:
  - Lettertype
  - Tekengrootte
  - Tekenkleur
  - Achtergrondkleur
  - Uitlijning

CascadingStyleSheets (CSS)



DEMO

Scalable Vector Graphics (SVG)

- SVG is een standaard XML vorm voor vectorafbeeldingen
- Met voor gedefinieerde tags kunnen figuren worden beschreven
- Goedgekeurd door het W3C
- Meeste webbrowsers kunnen afbeeldingen weergeven

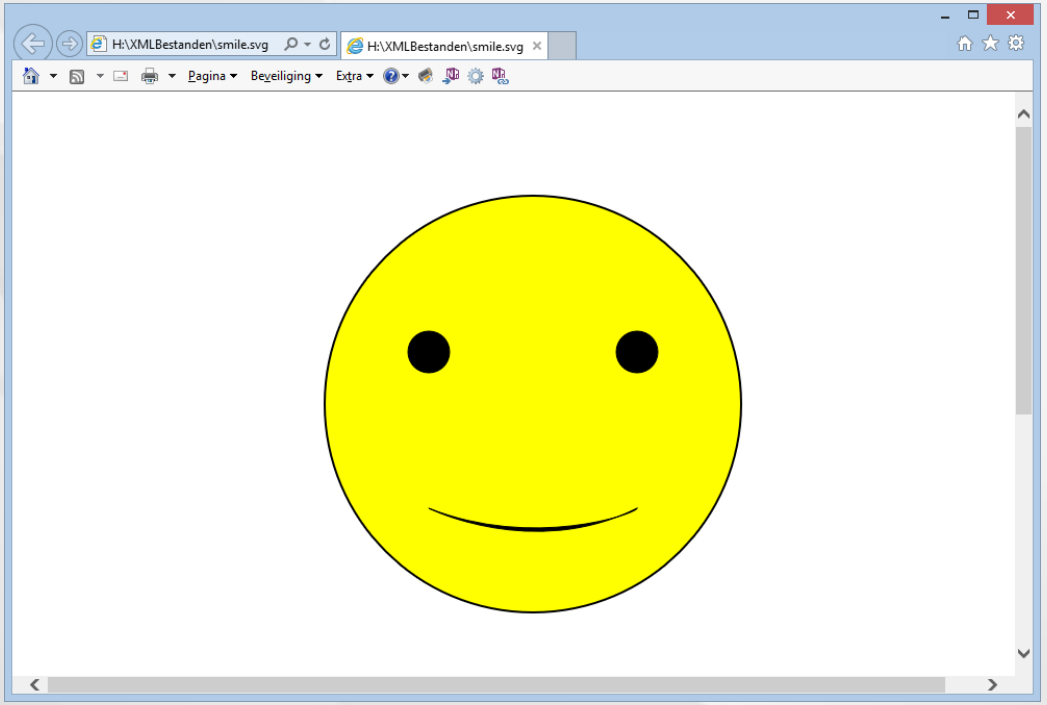
Scalable Vector Graphics (SVG)

- Onderstaande code is de beschrijving van een smiley in SVG
- De elementen en attributen zijn voor gedefinieerd

```
<?xml version="1.0" encoding="iso-8859-1"?>

<svg width="1000px" height="1000px">
  <g id="a">
    <circle cx="500" cy="300" r="200" fill="yellow" stroke="black" stroke-width="2"/>
    <circle cx="400" cy="250" r="20" fill="black"/>
    <circle cx="600" cy="250" r="20" fill="black"/>
    <path d="M400 400 C400 400 500 450 600 400C600 405 500 440 400 400" fill="black"/>
  </g>
</svg>
```

Scalable Vector Graphics (SVG)

- In een webbrowser wordt een svg document als afbeelding getoond



DEMO

XML verwerken

- Er zijn nog veel meer manieren om XML documenten te verwerken
- Veel databases hebben de mogelijkheid om XML te verwerken
  - Opslaan en lezen van XML gegevens
  - XML gegevens genereren uit relationele tabellen
- Ook veel programmeertalen kunnen XML gegevens verwerken
  - Lezen
  - Genereren

Hoofdstuk 4 DTD's en XML Schema

Leerdoelen:

Kunnen werken met DTD's

Een DTD kunnen ontwikkelen

Een document kunnen valideren

Kunnen werken met namespaces

Kunnen werken met XML schema's

Een XML schema kunnen ontwikkelen

Valideren

- De juiste opbouw van XML document is belangrijk voor het verwerken
- Well formedness is hierbij meestal niet voldoende
  - Controleert of de XML document syntactisch juist is opgebouwd
  - Geen controle op de structuur van het XML document
- De juiste structuur is belangrijk voor het verwerken van de XML
- Verschillende structuren voor verschillende situaties
- Controleren op de juiste structuur noemt men valideren
- Structuur eisen worden vastgelegd in DTD of XSD
  - DTD kan los maar ook embedded worden opgenomen
  - XSD is een XML document die de structuur beschrijft
- De meeste XML editors hebben een mogelijkheid om een XML document te valideren aan de hand van een DTD of een XML Schema

Document Type Definition

- Een DTD is een tekst bestand en beschrijft alle elementen die in een XML document voor kunnen komen
- In een DTD is aangegeven of elementen verplicht of optioneel zijn
  - Het is aan te geven of een element vaker voor kan komen
- De verwijzing naar een DTD wordt als DOCTYPE Declaration toegevoegd aan het begin van het XML document

Document Type Definition

- Een DOCTYPE declaratie begint altijd met <!DOCTYPE vervolgens de naam van het root element
- Na het root element zijn er twee mogelijkheden
  - SYSTEM "met een verwijzing naar het DTD bestand"
  - Tussen blokhaken een opsomming van elementen
    - Dit wordt wel een embedded DTD genoemd

Document Type Definition

- In een DTD wordt elk element word beschreven
- beginnend bij het root element
- Haakjes geven aan als een element een ander element bevat
- Qualifiers geven aan hoeveel een element mag voorkomen
- #PCDATA: Parsed Character Data
  - (de parser doorloopt de tekst,
    mogelijk bevat deze nog andere elementen)

Document Type Definition

- In een DTD wordt elk element met behulp van tags beschreven
  - `<!ELEMENT naam(type)>`
- Elke element heeft een naam
  - Dit is de naam van het element in het XML document
- Tussen de haakjes wordt het type aangegeven
  - #PCDATA om aan te geven dat het element tekst bevat
  - Een opsomming van één of meer child elementen
    - , vaste volgorde
    - | keuze
    - () groep
- Ook **alle** child elementen moeten in de DTD worden gedefinieerd

Document Type Definition

- Gescheiden door komma's
  - Definieert een vaste volgorde van een groep van child elementen
    - `<!ELEMENT adres (straat, huisnummer, postcode, woonplaats)>`
    - Het element adres heeft vier child elementen in de opgeven volgorde
- Gescheiden door |
  - Geeft een keuze aan tussen de opgeven elementen aan
    - `<!ELEMENT betaling (pin | cash)>`
    - Het elementen betaling bevat één child element dit is of pin of cash
- Gegroepeerd met ()
  - Definieert een groep van child elementen
    - `<!ELEMENT persoon ( naam, (gebdatum | leeftijd))>`
    - Het element persoon heeft twee child elementen
      - naam en gebdatum
      - of naam en leeftijd

Document Type Definition

- Met behulp van een qualifier kan worden aangeven hoe vaak een element of set aan elementen mag of moet voorkomen
- Mogelijke qualifiers
  - ? Optioneel (0 of 1 keer)
  - + Verplicht (1 of meer)
  - * Optioneel (0 of meer)
- Geen qualifier
  - Het element moet verplicht één keer voorkomen

Voorbeeld DTD

- Voorbeeld van de inhoud van een DTD:

```
<!ELEMENT cursussen (cursus)+>
<!ELEMENT cursus (naam, dagen, prijs?)>
<!ELEMENT naam (#PCDATA)>
<!ELEMENT dagen (#PCDATA)>
<!ELEMENT prijs (#PCDATA)>
```

- Het rootelement van de XML is het element `<cursussen>`
  - Er is altijd maar één rootelement
- Het element `<cursussen>` bevat zelf een element `<cursus>`
  - Er kunnen meerdere cursussen in het document staan vanwege `+`
- Elk element `<cursus>` bevat zelf weer drie child elementen
  - Dit zijn `<naam>` , `<dagen>` en `<prijs>`
  - Het element `<prijs>` is optioneel vanwege `?`
- De elementen `<naam>`, `<dagen>` en `<prijs>` hebben tekst als inhoud

Document Type Definition

- Voorbeeld XML bestand gebaseerd op het DTD cursussen.dtd:

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

<!DOCTYPE cursussen SYSTEM "cursussen.dtd">

<cursussen>
    <cursus>
        <naam>XML deel 1:introductie</naam>
        <dagen>2</dagen>
        <prijs>950</prijs>
    </cursus>
    <cursus>
        <naam>XSLT & XPATH</naam>
        <dagen>2</dagen>
        <prijs>950</prijs>
    </cursus>
    <cursus>
        <naam>XML Overview</naam>
        <dagen>1</dagen>
    </cursus>
</cursussen>
```

Document Type Definition

- Voorbeeld XML bestand met embedded DTD:

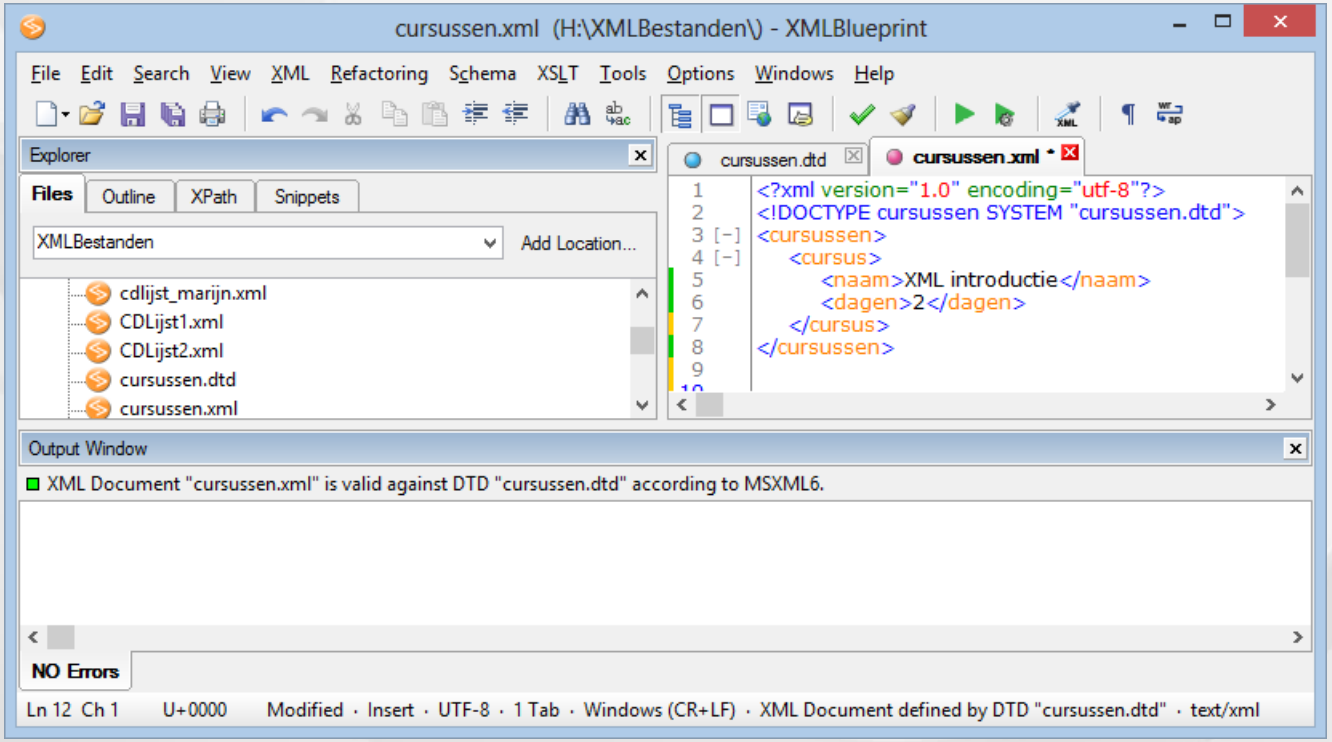
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cursussen [
        <!ELEMENT cursussen (cursus)+>
        <!ELEMENT cursus (naam, dagen, prijs?)>
        <!ELEMENT naam (#PCDATA)>
       <!ELEMENT dagen (#PCDATA)>
        <!ELEMENT prijs (#PCDATA)>
      ]>

<cursussen>
    <cursus>
        <naam>XML deel 1:introductie</naam>
        <dagen>2</dagen>
        <prijs>950</prijs>
    </cursus>
</cursussen>
```

Valideren

- De DOCTYPE declaratie boven het document geeft niet aan dat het document geldig (valid) is
- De DOCTYPE declaratie boven het document geeft alleen aan met welke DTD het document gevalideerd moet worden
- Om te bepalen of een XML document aan de structuur voldoet moet het XML document gevalideerd worden
- Het valideren doen we in XML Blueprint met de knop: **Validate** 

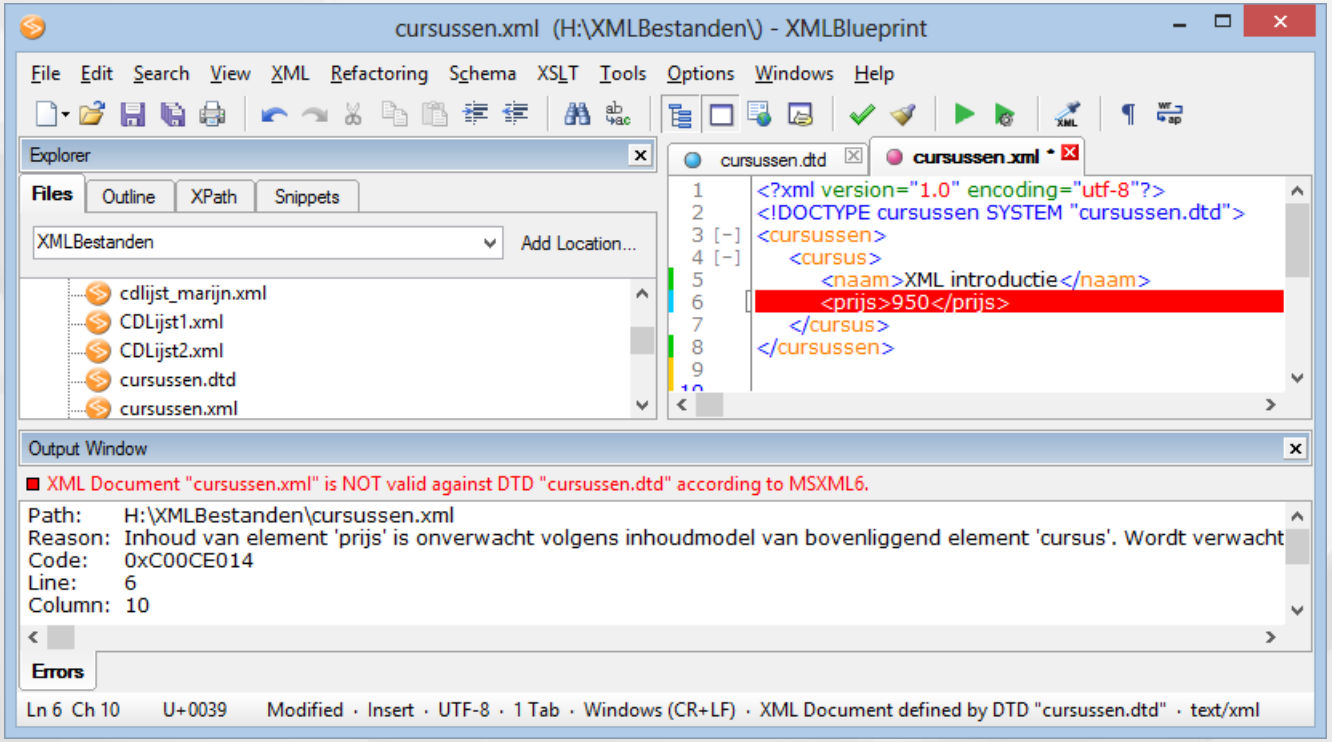
Valideren

- Meldingen over valideren verschijnen in het Ouput Window



Valideren

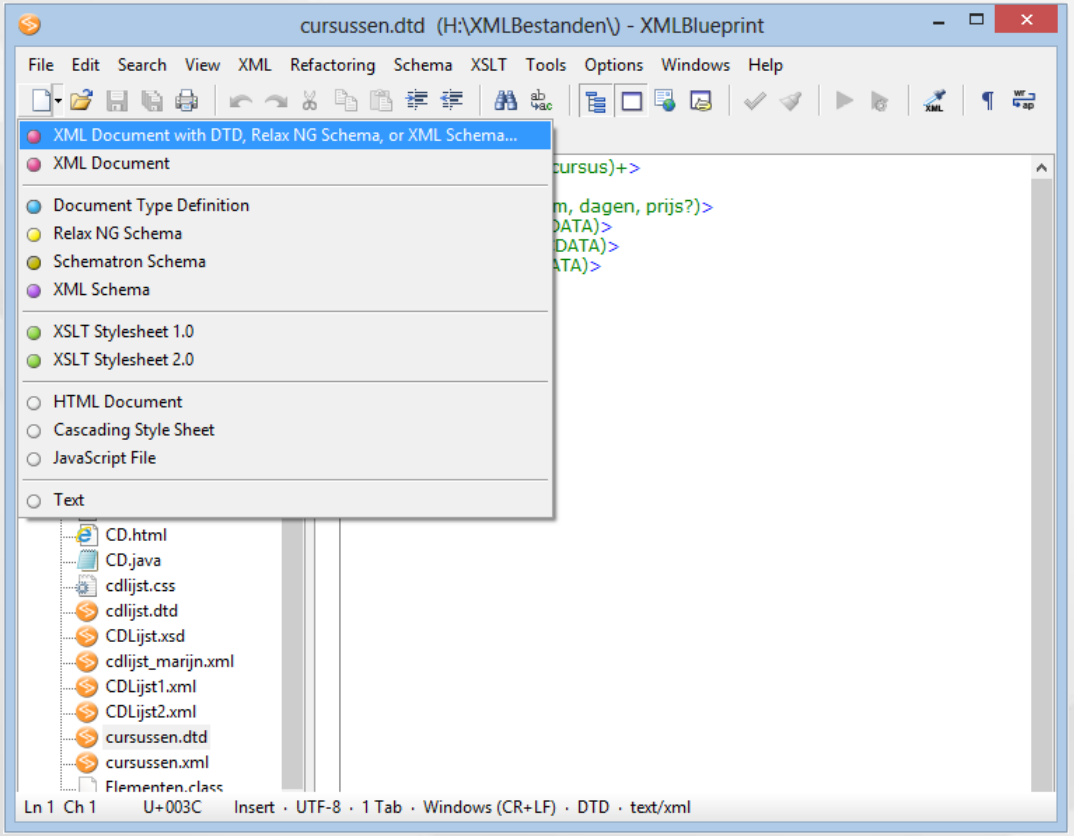
- Fouten worden gemarkeerd als het document niet valid is



DEMO

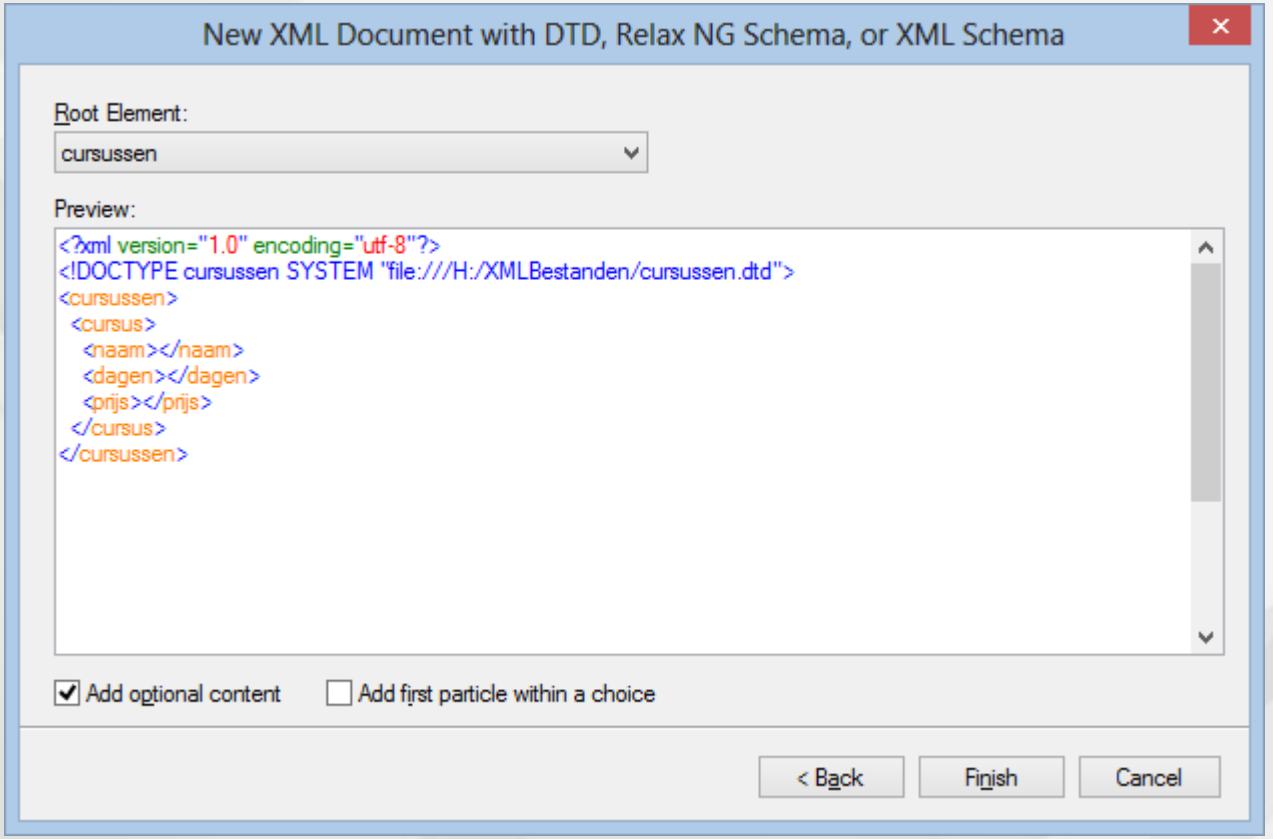
Document maken op basis van DTD

- Een nieuw document op basis van een DTD kan via de knop New File



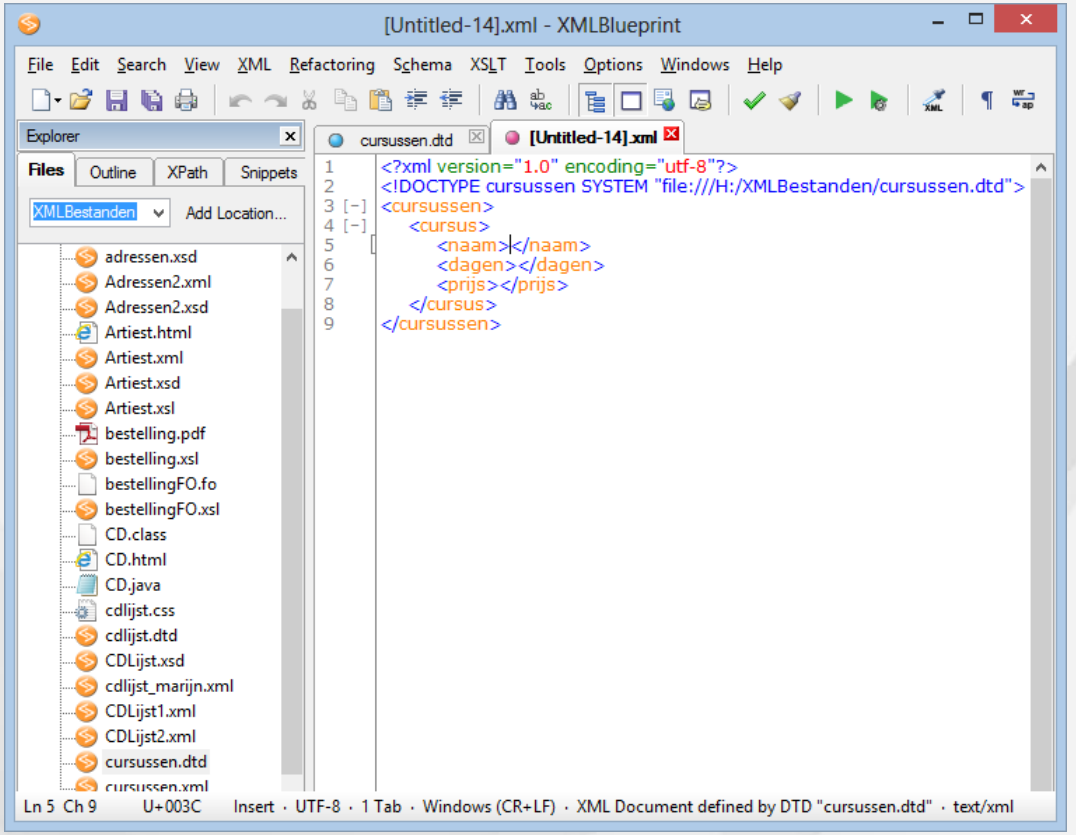
Document maken op basis van DTD

- Met een van de knoppen het gewenste DTD kiezen



Document maken op basis van DTD

- Het nieuwe XML document bevat de benodigde elementen en is gelijk te valideren



DEMO

Mixed content in DTD

- Als elementen met mixed content bevatten kan dit niet komma gescheiden als opsomming in het DTD worden opgenomen
- In een DTD kun je mixed content aangeven als: nul of meer elementen van type #PCDATA of ander type gescheiden door |:

```
<!DOCTYPE taak [
<!ELEMENT taak (#PCDATA|prioriteit|datum)*>
<!ELEMENT datum (#PCDATA)>
<!ELEMENT prioriteit (#PCDATA)>
]>


<taak>
   XML cursus
   <datum>2014-04-2017</datum>
</taak>
```

Lege elementen en willekeurige inhoud

- Soms mag een elementen willekeurige inhoud of elementen bevatten
  - Dit is in een DTD aan te geven met ANY
- Met Empty wordt afgedwongen dat een element geen inhoud bevat

```
<!DOCTYPE notes [
<!ELEMENT notes (note)*>
<!ELEMENT note ANY >
<!ELEMENT ready EMPTY >
<!ELEMENT important ANY >
<!ELEMENT secret (#PCDATA)>
]>

<notes>
    <note><important>blijven ademhalen</important></note>
    <note>voorbeeld</note>
    <note><secret>psst...</secret></note>
    <note><ready/>boodschappen doen</note>
</notes>
```

Validatie op attributen

- Elementen kunnen ook attributen hebben
- Dit moet ook in het DTD worden opgenomen met de tag ATTLIST
  - `<!ATTLIST elementnaam attribuutnaam CDATA optie>`
- De attribuutnaam moet worden opgeven en het attribuut hoort dan bij het element met de opgegeven elementnaam
- CDATA geeft aan dat het attribuut gewone tekst bevat
  - Zelfde type als `<![CDATA[...]]>` (Unparsed Character Data)
- Er moet altijd een optie mee geven worden Mogelijke Opties:
  - "waarde": standaard waarde indien niet opgegeven in XML
  - #REQUIRED: verplicht op te geven binnen XML
  - #IMPLIED: geen standaard waarde maar optioneel
  - #FIXED: vaste waarde, niet aan te passen via XML
- Voor meerdere attributen moet per attribuut per element een ATTLIST declaraties worden opgenomen

Voorbeeld ATTLIST

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

<!DOCTYPE personen [
<!ELEMENT personen (persoon+)>
<!ELEMENT persoon (#PCDATA)>
<!ATTLIST persoon geslacht CDATA #REQUIRED>
]>


<personen>
  <persoon geslacht="m">Hans</persoon>
  <persoon geslacht="v">Aniek</persoon>
</personen>
```

DTD - vaste waarden voor attributen

- In plaats van CDATA mag ook een lijst met vaste waarden opgeven worden tussen haakjes gescheiden door |
- Er moet dan tussen " " een standaard waarde opgegeven worden

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE vormen[
   <!ELEMENT vormen(vorm)+>
   <!ELEMENT vorm (#PCDATA)>
   <!ATTLIST vorm kleur (blauw|rood) "blauw">
]>

<vormen>
  <vorm kleur="rood">cirkel</vorm>
  <vorm>rechthoek</vorm>
  <vorm kleur="blauw">driehoek</vorm>
</vormen>
```

De attributen ID en IDREF

- Met een attributen ID en IDREF kunnen verwijzingen gemaakt worden
- Met een ID attribuut krijgt het element een uniek ID toegekend
- De waarde van het ID is voor het hele XML document uniek
  - Valideren gaat fout als ID niet uniek is
- Met een IDREF kan verwezen worden naar het betreffende ID
  - Valideren gaat fout als het betreffende ID niet bestaat

Voorbeeld ID en IDREF

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE personen [
<!ELEMENT personen (persoon+)>
<!ELEMENT persoon (#PCDATA)>
<!ATTLIST persoon persid ID #REQUIRED>
<!ATTLIST persoon vader IDREF #IMPLIED>
<!ATTLIST persoon moeder IDREF #IMPLIED>
]>


<personen>
      <persoon persid="p1">Hans</persoon>
      <persoon persid="p2">Aniek</persoon>
      <persoon persid="p3" vader="p1" moeder="p2">Gert</persoon>
      <persoon persid="p4" vader="p1" moeder="p2">Marit</persoon>
      <persoon persid="p5">Kirsten</persoon>
      <persoon persid="p6" vader="p3" moeder="p5">Merijn</persoon>
</personen>
```

Validatie op attributen

- Minder gebruikte typen
  - NMTOKEN en NMTOKENS
    - Wordt gebruikt om aan te geven dat de waarde van een attribuut moet voldoen aan de eisen die ook voor XML namen gelden.
      Bijvoorbeeld als het document wordt verwerkt met *java* of *javascript*
  - ENTITY en ENTITIES
    - Wordt gebruikt om een externe (niet XML) verwijzing te maken.
      Bijvoorbeeld afbeeldingen
  - NOTATION
    - Als er in een XML document gegevens voorkomen die geen XML zijn dan kan met behulp van *NOTATION* aangeven wat voor soort gegevens dit wel zijn

DTD maken

- Bij het maken van een DTD moeten alle elementen en attributen gedefinieerd worden
- De volgorde waarin dit gebeurd is willekeurig
- Zinvol om gestructureerd te werken
  - Begin bij het rootelement
  - Beschrijf alle child elementen en attributen van het root element
  - Beschrijf alle child elementen en attributen van het volgende element enz.

Opdracht 1 Hoofdstuk 4 (succes)

Namespaces

- Iedereen ontwikkelt zijn eigen XML documenten
- Dit kan bij het verwerken problemen geven met identieke tag namen
- Voorbeeld: naam komt zowel voor in cursussen.xml als cursisten.xml
- Dit kan opgelost worden met namespaces
- De namespace wordt aan de naam van een element toegevoegd
- Daarmee wordt aangegeven bij welk 'domein' een element hoort

Namespace Declaratie

- Gebruikte namespace moeten vooraf gedeclareerd worden
- Een namespace bestaat uit een prefix en een URI
  - Prefix is een zelf gekozen korte naam of afkorting
  - URI (Uniform Resource Identifier) is een unieke naam
    - De URI Is vaak een webadres (URL) omdat deze al uniek is
- Te declareren met het attribuut xmlns:prefix binnen een element
  - `<bestelling xmlns:super="http://super.nl/namespaces/">`

Namespace gebruiken

- Als de namespace gedeclareerd is kan deze gebruikt worden
- Dit kan door prefix: voor tagname te plaatsen
  - Dit moet in de begintag en in de eindtag
  - `<cur:naam>XML deel 1:introductie</cur:naam>`
- De namespace is voor alle children van het element te gebruiken
  - De namespace wordt daarom vaak in het root element gedeclareerd

Voorbeeld met namespaces

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cursussen SYSTEM "cursussen.dtd">
<cursussen xmlns:opld="http://vijfhart.nl/opleidingen">
    <opld:cursus>
        <opld:naam>XML deel 1:introductie</opld:naam>
        <opld:dagen>2</opld:dagen>
        <opld:prijs>950</opld:prijs>
    </opld:cursus>
</cursussen>
```

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cursisten SYSTEM "cursisten.dtd">
<cursisten xmlns:cust= "http://vijfhart.nl/cursisten">
    <cust:cursist>
        <cust:naam>Anouk de Bruijn</cust:naam>
        <cust:adres>Iepenhoeve 39</cust:adres>
        <cust:woonplaats>Nieuwegein</cust:woonplaats>
    </cust:cursist>
</cursisten>
```

Voorbeeld met namespaces

- De namespace kan ook een niveau dieper op het element cursus worden gedefinieerd

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cursussen SYSTEM "cursussen.dtd">

<cursussen>
  <opld:cursus xmlns:opld="http://vijfhart.nl/opleidingen">
    <opld:naam>XML deel 1:introductie</opld:naam>
    <opld:dagen>2</opld:dagen>
    <opld:prijs>950</opld:prijs>
  </opld:cursus>
</cursussen>
```

Voorbeeld namespaces

- Dit gaat niet goed bij meerdere cursussen
- het tweede element cursus is geen child van het element waarop de namespace is gedefinieerd

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cursussen SYSTEM "cursussen.dtd">
<cursussen>
  <opld:cursus xmlns:opld="http://vijfhart.nl/opleidingen">
     <opld:naam>XML deel 1:introductie</opld:naam>
     <opld:dagen>2</opld:dagen>
     <opld:prijs>950</opld:prijs>
  </opld:cursus>
  <opld:cursus>
     <opld:naam>XML deel 2:XSLT en XPath</opld:naam>
     <opld:dagen>2</opld:dagen>
     <opld:prijs>950</opld:prijs>
  </opld:cursus>
</cursussen>
```

- Het document is nu niet meer well-formed!!

Voorbeeld namespaces

- Bij declaratie in het element cursussen is dit geen probleem

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cursussen SYSTEM "cursussen.dtd">

<cursussen xmlns:opld="http://vijfhart.nl/opleidingen">
  <opld:cursus >
     <opld:naam>XML deel 1:introductie</opld:naam>
     <opld:dagen>2</opld:dagen>
     <opld:prijs>950</opld:prijs>
  </opld:cursus>
  <opld:cursus>
     <opld:naam>XML deel 2:XSLT en XPath</opld:naam>
     <opld:dagen>2</opld:dagen>
     <opld:prijs>950</opld:prijs>
  </opld:cursus>
</cursussen>
```

Meerdere namespaces

- Binnen een document zijn meerdere namespaces te declareren
- Bij gebruik van meerdere namespaces in een document kan een namespace worden aangeduid als de default, deze heeft geen prefix

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cust:cursisten SYSTEM "cursisten.dtd">
<cust:cursisten xmlns:cust= "http://vijfhart.nl/cursisten">
  <cust:cursist>
    <cust:naam>Anouk de Bruijn</cust:naam>
    <cust:adres>Iepenhoeve 39</cust:adres>
    <cust:woonplaats>Nieuwegein</cust:woonplaats>
    <opld:cursussen xmlns:opld="http://vijfhart.nl/opleidingen">
      <opld:cursus >
        <opld:naam>XML deel 1:introductie</opld:naam>
        <opld:dagen>2</opld:dagen>
        <opld:prijs>950</opld:prijs>
      </opld:cursus>
    </opld:cursussen>
  </cust:cursist>
</cust:cursisten>
```

Meerdere namespaces

- Beide namespaces worden ook wel in het rootelement gedeclareerd

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cust:cursisten SYSTEM "cursisten.dtd">
<cust:cursisten xmlns:cust= "http://vijfhart.nl/cursisten"
                xmlns:opld="http://vijfhart.nl/opleidingen">
  <cust:cursist>
    <cust:naam>Anouk de Bruijn</cust:naam>
    <cust:adres>Iepenhoeve 39</cust:adres>
    <cust:woonplaats>Nieuwegein</cust:woonplaats>
    <opld:cursussen>
      <opld:cursus >
        <opld:naam>XML deel 1:introductie</opld:naam>
        <opld:dagen>2</opld:dagen>
        <opld:prijs>950</opld:prijs>
      </opld:cursus>
    </opld:cursussen>
  </cust:cursist>
</cust:cursisten>
```

Default namespaces

- Een namespace mag ook als default namespace worden gedeclareerd
- De default namespace wordt gedeclareerd zonder prefix
  - Hiervoor wordt het attribuut **xmlns** gebruikt
- Er kan slechts één namespace de default namespace zijn
- Elementen zonder prefix horen bij de default namespace

Voorbeeld default namespace

- In het volgende voorbeeld is een default namespace gedeclareerd

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cursisten SYSTEM "cursisten.dtd">
<cursisten xmlns="http://vijfhart.nl/cursisten"
           xmlns:opld="http://vijfhart.nl/opleidingen">
  <cursist>
    <naam>Anouk de Bruijn</naam>
    <adres>Iepenhoeve 39</adres>
    <woonplaats>Nieuwegein</woonplaats>
    <opld:cursusen>
      <opld:cursus>
        <opld:naam>XML deel 1:introductie</opld:naam>
        <opld:dagen>2</opld:dagen>
        <opld:prijs>950</opld:prijs>
      </opld:cursus>
    </opld:cursusen>
  </cursist>
</cursisten>
```

Aanpassing DTD voor gebruik namespaces

- Bij gebruik van namespaces moeten ook DTD aangepast worden om te kunnen valideren
- In de DTD moet de prefix van de namespace aan de elementnamen toegevoegd worden
- Het element met de namespace declaratie moet een ATTLIST hebben

```
<!ELEMENT opld:cursussen (opld:cursus)+>
<!ATTLIST opld:cursussen xmlns:opld CDATA #FIXED 5hart.nl/opleidingen">
<!ELEMENT opld:cursus (opld:naam, opld:dagen, opld:prijs?)>
<!ELEMENT opld:naam (#PCDATA)>
<!ELEMENT opld:dagen (#PCDATA)>
<!ELEMENT opld:prijs (#PCDATA)>
```

- In de XML moet ook de DOCTYPE definitie aan gepast worden

```
<!DOCTYPE opld:cursusisten
```

Namespaces

- Door W3C zijn een aantal standaard namespaces vastgelegd
- Worden gebruikt voor het werken met XML of specifieke vocabulaires
- Standaard (industriële) namespaces:
  - XML Schema http://www.w3.org/2001/XMLSchema
  - XSLT http://www.w3.org/1999/XSL/Transform
  - XHTML http://www.w3.org/1999/xhtml
  - XSL-FO http://www.w3.org/1999/XSL/Format

XML Schema's

- Het valideren van XML documenten met een DTD heeft beperkingen
- Een DTD kijkt naar de structuur en niet naar de inhoud
- Een DTD controleert bijvoorbeeld niet het type van de gegevens
- Uitgebreidere validatie kan met een XML schema
- Een XML schema is een bestand waarin de structuur van een XML document wordt beschreven
- Een XML schema is zelf ook een XML bestand met de extensie .xsd
  - Omdat het xml is moet het ook aan de regels van xml voldoen

XML Schema's

- Een schema wordt ook wel XSD genoemd (XML Schema Definition)
- XSD biedt meer mogelijkheden dan DTD:
  - Hoe vaak mag een element voorkomen (minimaal en maximaal)
  - Van welk type moet de inhoud zijn
  - Wat is de maximale lengte van de inhoud
  - Welke waarden mag een element hebben
- XSD gebruikt de namespace http://www.w3.org/2001/XMLSchema
- In deze namespace zijn alle elementen van een schema vastgelegd
- Deze namespace krijgt meestal de prefix xsd of xs

XML Schema Definition

- Een schema is een XML document met een rootelement schema
- Dit element hoort bij de schema namespace
- De verwijzing naar de namespace is een attribuut van dit element
- Het element schema kan meerdere attributen hebben

```
<xsd:schema xmlns="default_namespace"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="doel_namespace"
elementFormDefault="qualified">
```

- targetNamespace geeft aan bij welke namespace de gedefinieerde elementen en typen horen
- targetNamespace is vaak gelijk aan de default namespace
- elementFormDefault geeft aan dat gedeclareerde elementen behoren tot de targetNamespace, ipv tot de null namespace

XML Schema Definition

- In een XML moet naar het schema verwezen worden
- Dat kan met het attributen schemaLocation
- Dit attribuut hoort bij de namespace http://www.w3.org/2001/XMLSchema-instance
- Deze namespace moet daarom wel opgenomen worden

```
<cursussen  xmlns="www.cursussen"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="www.cursussen
file:///H:/XMLBestanden/cursussen.xsd">
```

- Omdat een xsd zelf ook gevalideerd kan worden zien we de attributen schema-instance en schemaLocation ook vaak terug in een xsd

DEMO

XML Schema - element

- Met element **element** kunnen elementen gedefinieerd worden
- Het attribuut name is de naam van het element en is verplicht
- Daarnaast nog enkele veelgebruikte attributen
  - type: het datatype van het element
  - minOccurs: minimum aantal van het element
    - 0: optioneel
    - 1 (default) of meer
  - maxOccurs: maximum aantal van het element
    - 1(default) of meer
  - fixed: om af te dwingen dat er een vaste waarde wordt opgegeven
  - default: om een standaard waarde op te geven
  - id:definieert een uniek ID voor een element
  - ref: om een verwijzing naar een bovenliggende element te maken

XML Schema datatypen

- Belangrijkste datatypen:
  - xsd:string — tekst
  - xsd:int — geheel getal tussen -2^{31} en $2^{31}-1$
  - xsd:float — decimaal getal
  - xsd:decimal — decimaal getal
  - xsd:boolean — booleaanse waarden [true, false, 0, 1]
  - xsd:date — datum [YYYY-MM-DD]
  - xsd:time — tijd [hh-mm-ss]
  - xsd:id — uniek id voor het element

XML Schema - simpleType

- De door W3C vastgelegde datatypen zijn niet altijd toereikend
- Daarom kunnen zelf nieuwe typen gemaakt worden
  - Een waarde moet tussen 0 en 50 liggen
  - Lijst met voor gedefinieerde waarden
  - Een postcode moet 4 cijfers en 2 hoofdletters hebben
- Een eigen type maken kan met het element **simpleType**
- Een simpleType kan gebruikt worden voor een element dat geen children bevat en geen attributen heeft

XML Schema - simpleType

- Binnen simpleType moet minimaal één van de volgende elementen gebruikt worden
  - **restriction** voor het definiëren van een beperking
  - **list** inhoud XML is een lijst; items gescheiden door spaties
  - **union** geeft aan dat inhoud bestaat uit 1 of meer simpleTypes
  - **annotation** voor het weergeven van in-line opmerkingen
- Meestal wordt binnen simpleType restriction gebruikt
- Restriction heeft altijd een basistype
- Binnen restriction zijn elementen voor voorwaarden op te geven zoals
  - Enumeration
  - minLength, maxLength, length, fractionDigits, totalDigits
  - maxInclusieve, maxExclusive, minInclusive, minExclusive
  - pattern (reguliere expressie)
  - whiteSpace

XML Schema - simpleType

- Voorbeelden restriction

```
<xsd:simpleType name="werkdagen">
    <xsd:restriction base="weekdagen">
            <xsd:maxInclusive value="5"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name= "weekdagen">
    <xsd:restriction base="xsd:int">
        <xsd:enumeration value= "1"/>
        <xsd:enumeration value= "2"/>
        <xsd:enumeration value= "3"/>
        <xsd:enumeration value= "4"/>
        <xsd:enumeration value= "5"/>
        <xsd:enumeration value= "6"/>
        <xsd:enumeration value= "7"/>
    </xsd:restriction>
</xsd:simpleType>
```

XML Schema – reguliere expressies

- Het element restriction kent een childelement **pattern**
- Pattern heeft een reguliere expressie als patroon
- De inhoud van het XML element moet aan het patroon voldoen
- Het patroon is op te geven bij het attribuut value
  - <xsd:pattern value= "reguliere expressie">

XML Schema – reguliere expressies

- Belangrijkste reguliere expressies:
  - [AaXYZ1] lijst van karakters
  - [A-Z] range van karakters
  - [^Aa] niet toegestane karakters
  - (Alfa) groep van karakters
  - . willekeurig karakter
  - ? voorgaande karakter/groep mag 0 of 1 keer voorkomen
  - * voorgaande karakter/reeks mag 0 of 1 keer voorkomen
  - + voorgaande karakter/groep moet 1 of meer keer voorkomen
  - {n} voorgaande karakter/groep moet n keer voorkomen
  - {n,m} voorgaande karakter/groep moet tussen n en m keer voorkomen
  - | de ene of de andere karaktergroep
  - \s witruimte
  - \S geen witruimte
  - \d numeriek
  - \D niet numeriek
  - \w alfanumeriek
  - \ niet alfanumeriek

XML Schema – reguliere expressies

- Voorbeeld postcode moet bestaan uit 4 cijfers en 2 hoofdletters :

```
<xsd:simpleType name="postcode">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[1-9]{4} ?[A-Z]{2}" />
    </ xsd:restriction>
</xsd:simpleType>
```

DEMO

XML Schema – complexType

- Een simpleType is niet bruikbaar voor elementen met attributen en childelementen
- In dat geval moet er een complexType gebruikt worden
- De belangrijkste childelementen binnen een complexType zijn:
  - **sequence** Definieert een volgorde van elementen
  - **choice** Keuze uit één van de opgegeven elementen
    Bij maxOccurs groter dan 1 mogen meer opgegeven elementen gebruikt worden
  - **all** Alle opgeven elementen moeten precies één keer voor moeten komen. De volgorde maakt niet uit.
  - **group** Verwijzing naar een eerder genoemde groep

XML Schema – complexType

- Voorbeeld complexType:

```
<xsd:element name="cursussen">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="cursus" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="naam" type="xsd:string"/>
            <xsd:element name="dagen" type="xsd:int"/>
            <xsd:element name="prijs" type="xsd:float"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

XML Schema – attribute

- Om binnen een complexType attributen te definiëren wordt het element **attribute** gebruikt
- Een attribuut moet een naam hebben en een type
- Het type moet een simpleType of voor gedefinieerd type zijn
- In attribute kan ook het attribuut **use** worden opgenomen
- use kan de waarden **optional** (default), **required**, of **prohibited** hebben

```
<xsd:element name="prijs">
    <xsd:complexType>
       <xsd:attribute name="valuta" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>

<prijs valuta="€"></prijs>
```

XML Schema – attribute

- Op deze manier attributen definiëren heeft beperkingen
  - Het element kan nu geen inhoud of childelementen hebben
- Dit kan wel bij gebruik van **simpleContent** of **complexContent**
- Deze elementen worden daarom veel gebruikt binnen complexType

XML Schema – simpleContent

- Als elementen met attributen alleen tekst als inhoud en hebben kan simpleContent gebruikt worden
- Een simpleContent bevat altijd **extension** of **restriction**
- Zo wel restriction als extension zijn gebaseerd op een ander type
  - Op te geven in het attribuut base

XML Schema – simpleContent

- Voorbeeld complexType met simpleContent

```
<xsd:element name="dagen">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="dag" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="weekdagen">
              <xsd:attribute name="uren" type="xsd:int/>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<dag uren="8">maandag</dag>
```

- Het type weekdagen is een zelf gedefinieerd simpleType (sheet 105)

Restriction en extension samen

```
<xsd:simpleType name="prijsType">
  <xsd:restriction base="xsd:float">  <!-- beperking op xsd:float -->
     <xsd:minExclusive value="0" />   <!-- alleen bedragen boven 0 -->
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="valutaType">
  <xsd:restriction base="xsd:string">  <!-- beperking op xsd:string -->
    <xsd:enumeration value="€"/>       <!-- mag alleen € of $ zijn -->
    <xsd:enumeration value="$"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="prijs">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="prijsType">   <!-- uitbreiding op prijsType -->
        <xsd:attribute name="valuta" type="valutaType"/> <!-- met attribuut -->
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

XML Schema – complexContent

- Voor elementen met attributen die wel childelementen bevatten moet complexContent worden gebruikt
- Ook complexContent bevat altijd **extension** of **restriction**
- En ook hier wordt het basistype opgeven in het attribuut base

XML Schema – complexContent

```
<xsd:complexType name="persoon">
  <xsd:sequence>
    <xsd:element name="voornaam" type="xsd:string"/>
    <xsd:element name="achternaam" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="werknemers">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="werknemer" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="persoon">
              <xsd:attribute name="leeftijd" type="xsd:int" use="required"/>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

XML Schema - group en attributeGroup

- Soms is het handig om attributen te groeperen zodat deze hergebruikt kunnen worden binnen andere elementen
- Hiervoor wordt met **group** een groep gedefinieerd
- Via **ref** wordt binnen elementen verwezen naar die groep
- Binnen een groep kunnen ook **sequence**, **choice**, of **all** gedefinieerd worden
- Met attributeGroup wordt een groep van attributen gedefinieerd

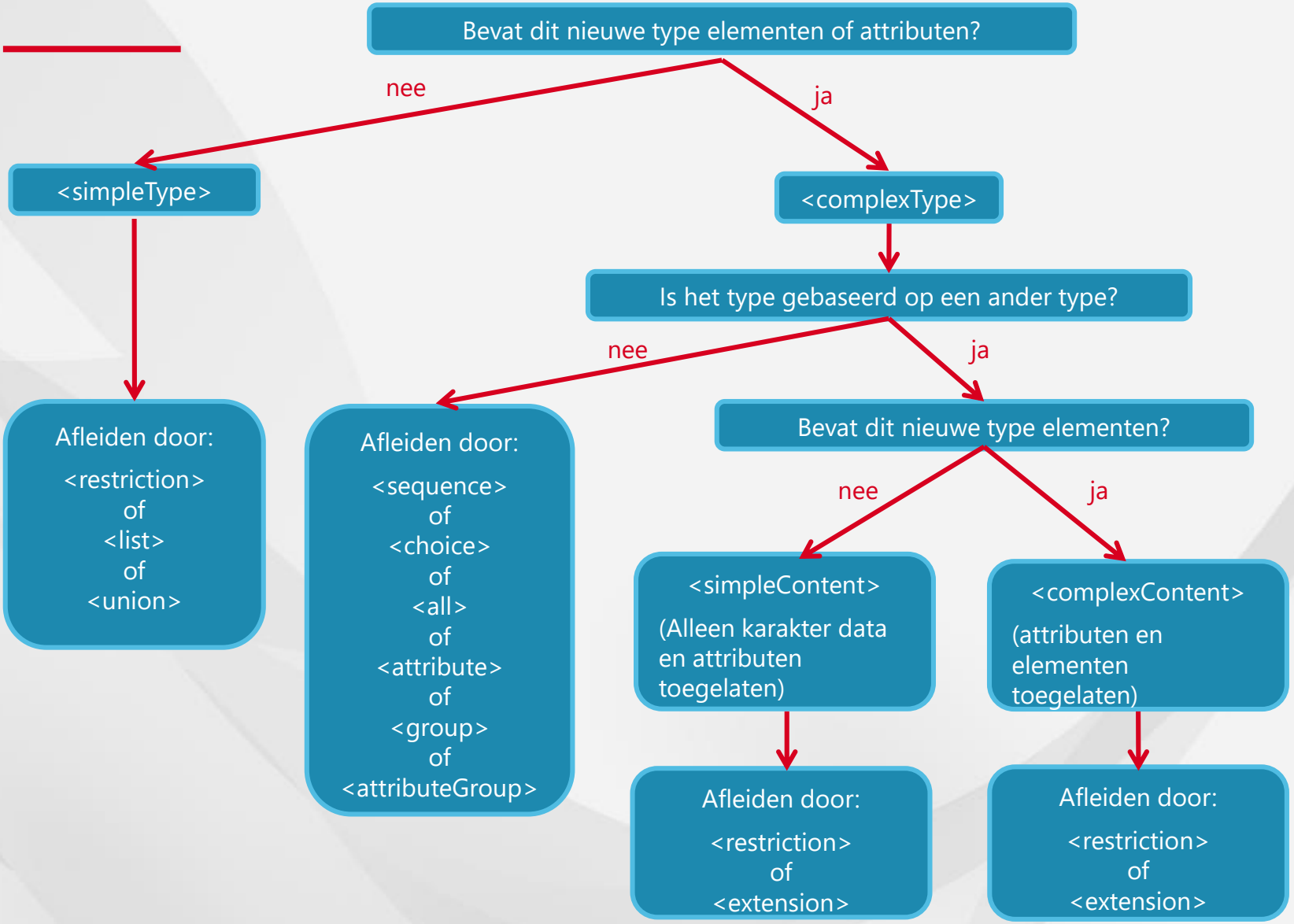
Group en attributeGroup

```
<xsd:group name="opleiding">
  <xsd:sequence>
    <xsd:element name="Naam" type="xsd:string" />
    <xsd:element name="Dagen" type="xsd:int" />
    <xsd:element name="Prijs" type="xsd:int" />
  </xsd:sequence>
</xsd:group>

<xsd:attributeGroup name= "revisie">
  <xsd:attribute name="Dag" type="xsd:string" />
  <xsd:attribute name="Maand" type="xsd:string" />
  <xsd:attribute name="Jaar" type="xsd:integer" />
</xsd:attributeGroup>

<xsd:complexType name="cursus">
  <xsd:sequence>
    <xsd:group ref="opleiding"/>
      <xsd:element name="..." type="..."/>
    </xsd:sequence>
  <xsd:attributeGroup ref="revisie"/>
</xsd:complexType>
```

DEMO

Een schema maken

- Een schema maken is niet eenvoudig
- Eerst de structuur van het document goed kennen
- In XML Blueprint kan een schema gemaakt worden met het menu **file**
- Een aantal standaard namespaces worden dan al toegevoegd
- Vervolgens moet stap voor stap alle elementen beschreven worden
- Bedenk steeds welk type nodig is
  - Standard type
  - simpleType
  - complexType
    - simpleContent
    - complexContent
- Het volgende schema kan een handig hulpmiddel zijn



XML Schema modulair opbouwen

- Basistypen definiëren
- Basiselementen onderscheiden
- Groepen opbouwen
- Documentstructuur opbouwen
- Grote schema's eventueel opsplitsen in kleinere:
  - xsd:include
    Invoegen van schema voor elementen van zelfde namespace
  - xsd:import
    Invoegen van schema voor elementen van andere namespace

Opdrachten Hoofdstuk 4 (succes)

Hoofdstuk 5 Transformatie en opmaak

Leerdoelen:

Omzetten van XML naar HTML

Omzetten van XML naar XML

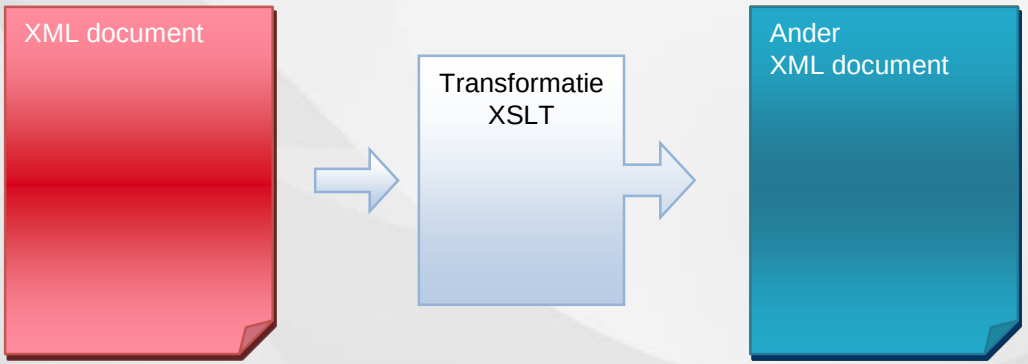
Een XML document weergeven met CSS

Een XSL-FO document maken

Een XML document weergeven met XSL-FO

XSLT Stylesheets

- Niet iedere organisatie kan met één uniform (XML) document werken
- Ook verschillende afdelingen kunnen verschillende XML gebruiken
- Bij gebruik van XML als communicatie zijn vaak vertaalslagen nodig
- Voorbeelden XML naar ander XML, XML naar PDF of XML naar HTML
- Voor transformaties worden XSLT Stylesheets gebruikt



XSLT Stylesheets

- In een XSLT document worden regels vastgelegd op basis waarvan een nieuw document wordt gegenereerd
  - Hoe komt het nieuwe document er uit te zien
  - Welke bestaande objecten worden gebruikt
- Een minimaal XSLT document:
  Het resultaat is een leeg XML document

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

<xsl:stylesheet version="1.0"
                mlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:template match="/">

     </xsl:template>

</xsl:stylesheet>
```

XSLT Stylesheets: stylesheet

- Vaak verwijzingen naar de namespaces uit het bron document opgenomen in het element **template**
- Deze namespaces krijgen allemaal een prefix die in de xsl gebruikt wordt
- Namespaces die in het nieuwe XML document niet nodig zijn mogen bij het attribuut **exclude_result_prefixes** worden aangegeven

```
<xsl:stylesheet version="1.0"
                mlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:web="http://www.webwinkel.nl"
                exclude-result-prefixes="web">
```

XSLT Stylesheets

- Het template element bepaalt wat er moet gebeuren zodra een match gevonden wordt met het attribuut **match**
- In het eenvoudige voorbeeld verwijst de **/** naar het root element
  - Bij match mag wel naar andere elementen verwezen worden
- Binnen het template wordt de structuur voor het nieuwe document opgesteld.
- Binnen de structuur kan content uit het bron document gemengd worden met de nieuwe structuur

XSLT Stylesheets: template

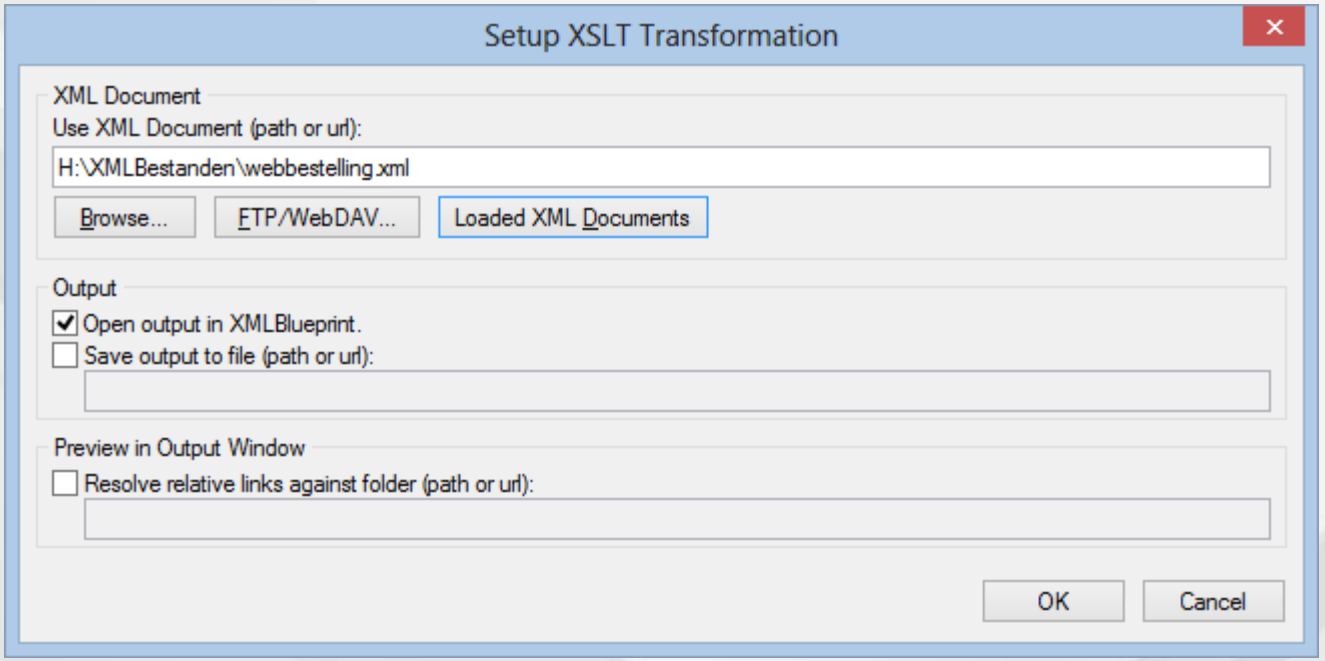
- Door nieuwe tags binnen het element **template** toe te voegen kan de structuur van het nieuwe document opgegeven worden

```
<xsl:template match="/">
  <order>
    <contactpersoon>Piet Paulusma</contactpersoon>
    <product>
      <productid>??</productid>
      <aantal>??</aantal>
    </product>
    <datum>??</datum>
    <klantid>12345</klantid>
  </order>
</xsl:template>
```

- Vaste waarde voor de inhoud kunnen eenvoudig worden toegevoegd
- Andere waarden moeten worden opgehaald uit de oorspronkelijke XML
  - Deze waarden worden hier nog aangegeven met **??**

Transformatie uitvoeren

- Transformatie uit te voeren met de knop Run XSLT transformatie 



- De transformatie opnieuw uitvoeren kan met de knop 
- De transformatie instellingen aan passen kan met de knop 

Transformatie uitvoeren

- De optie Open output in XMLBlueprint geeft na transformatie een nieuw XML document
- Dit XML document kan worden opgeslagen
- Het document krijgt een leesbare opmaak met Document Format 

DEMO

XSLT Stylesheets: value-of

- Om waarden uit het bron document toe te voegen moeten deze eerst worden opgezocht
- Hiervoor kan het element **value-of** gebruikt worden
- Dit element heeft een attribuut **select** waarmee wordt bepaald wat er uit het bron document wordt opgehaald
- De waarde van dit attribuut is een XPath expressie

```
<aantal>
  <xsl:value-of select="//web:aantal"/>
</aantal>
```

- In dit voorbeeld wordt uit het bron document de waarde van het element aantal opgehaald
  - De **//** in de XPath expressie zoekt naar alle elementen aantal

XPath expressies

- In een XPath expressie voldoet // vaak niet
  - Als een element meerdere keren voorkomt wordt de eerste getoond
- Om de juiste waarden op te halen moet er met een XPath expressie door het bron document genavigeerd worden
- Een XPath expressie beschrijft de weg door de boomstructuur van het bron document
- Hiervoor moet de structuur van het bron document goed bekend zijn
- Met XPath expressie kunnen zowel elementen als attributen ophalen
- De term node is de verzamelnaam voor elementen en attributen
  - Een node kan zowel een element als een attribuut zijn
- XMLBluePrint biedt ondersteuning voor XPath

XPath expressies

- Binnen XPath zijn de volgende expressies mogelijk

Expressie	beschrijving
nodename	Selecteert alle nodes van met deze naam op deze plek in het pad
/	Selecteert de root (waarvan het root element het enige child element is).
//nodename	Selecteert alle nodes in het document die aan de opgegeven naam voldoen het maakt niet uit waar ze in de structuur staan.
.	Selecteert de huidige node.
..	Selecteert de parent van de huidige node.
@attributename	Selecteert een attribuut met deze naam.
*	Een wildcard voor elke element node.
@*	Een wildcard voor elke attribute node.

XPath expressies

- Met XPath kan door de nodes van een XML boomstructuur worden genavigeerd
- Voorbeeld bron document:

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

<cursussen>
  <cursus>
    <naam>XSLT en XPATH</naam>
    <dagen>2</dagen>
    <prijs valuta="€">950</prijs>
  </cursus>
</cursussen>
```

- Voorbeeld XPath expressies:
  - /cursussen/cursus geeft XSLT en XPATH 2 950
  - /cursussen/cursus/naam geeft XSLT en XPATH
  - /cursussen/cursus/prijs/@valuta geeft €

XPath expressies

- Voorbeeld Extra expressies:

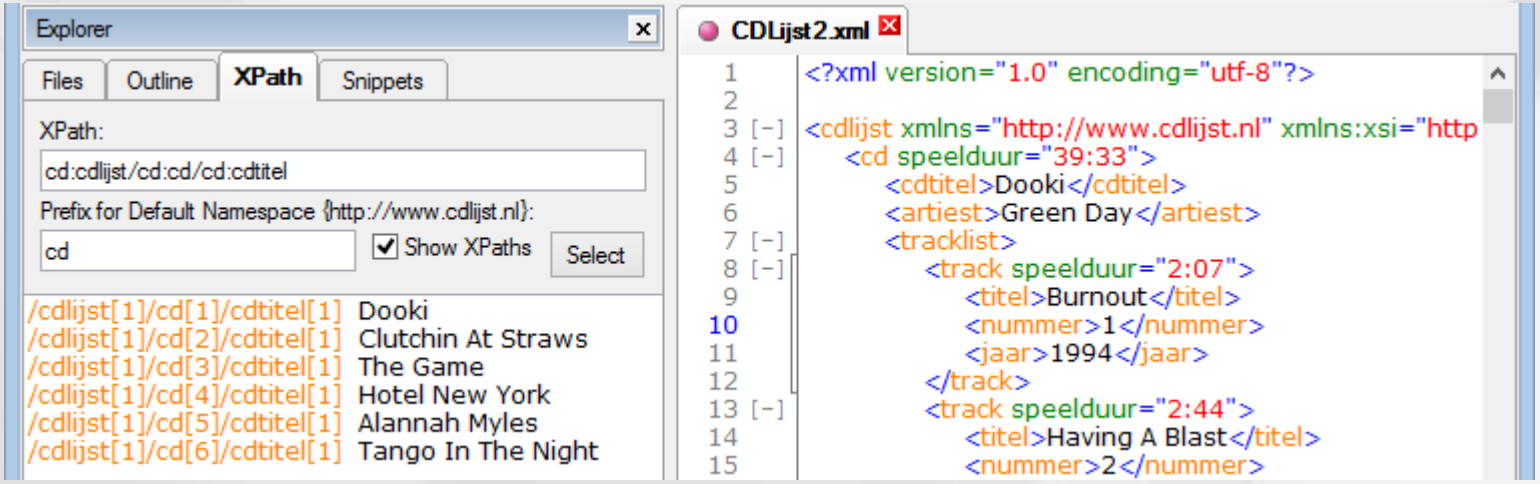
/cursussen/cursus[1]	De eerste cursus uit de lijst
/cursussen/cursus[last()]	De laatste cursus uit de lijst
/cursussen/cursus[position()<3]	De eerste twee cursussen
//prijs[@valuta="€"]	Alle nodes met de naam prijs met attribuut valuta met waarde €
//cursus[1]/prijs	Prijs van de eerste cursus

XSLT en XPath

- In een bron document worden vaak namespaces gebruikt
- Deze namespaces zullen dan ook in het stylesheet moeten worden opgenomen
- Dit geldt ook voor de default namespace van het bron document
  - Deze krijgt in het stylesheet wel een prefix
  - Het stylesheet heeft zelf al een andere default namespace
- In XPath expressies moet naar de juiste namespace verwezen worden

XPath evaluator

- Met een XPath evaluator kan een XPath expressie getest worden
- In XML blueprint is een XPath evaluator aanwezig
  - Tabblad XPath in de Explorer
- Vul een XPath expressie in en klik op Select
- Vul in het onderste vak een prefix voor de default namespace in
  - Gebruik hier de prefix die ook in de xsl wordt gebruikt



XSLT elementen

- Om gegevens uit een bron document om te zetten zijn er meer elementen beschikbaar binnen een stylesheet
- Al deze elementen hebben hun eigen mogelijkheden en kunnen waar nodig worden toegepast
- Elke element heeft zijn eigen specifieke attributen en of child elementen
- Vaak maken deze elementen gebruik van een XPath expressie
- Een XPath expressie zoekt niet telkens vanaf het begin in de boom structuur, maar gaat verder vanaf het punt van de vorige stap
  - Het attribuut match binnen template speelt hierin ook een rol

XSLT Elementen

<xsl:template>	Definieert een sjabloon
<xsl:apply-templates>	Past templates toe op child elementen of op door XPath expressie gekozen elementen (select='...')
<xsl:output>	Definieert output formaat bv xml, html
<xsl:value-of>	Haalt de waarde op via de gegeven XPath expressie
<xsl:for-each>	Wordt gebruikt om een bepaalde node-set van een opgeven XML element te doorlopen.
<xsl:if>	Wordt alleen uitgevoerd als aan de opgeven voorwaarde bij het attribuut test wordt voldaan
<xsl:choose>	Combinatie van verschillende tests. gecombineerd met minimaal één <xsl:when>
<xsl:sort>	Sorteert de opgegeven lijst op basis van een element gegeven bij het attribuut select.

XPath functie

- Binnen een XPath expressie kunnen ook functies gebruikt worden
- Bijvoorbeeld de functie concat waarmee verschillende waarden aan elkaar gekoppeld kunnen worden
- Onderstaande manieren leveren hetzelfde resultaat

```
<datum>
  <xsl:value-of select="web:jaar"/>-<xsl:value-of select="web:maand"/>-
  <xsl:value-of select="web:dag)"/>
</datum>
```

```
<datum>
  <xsl:value-of select="concat(//web:jaar, '-' ,//web:maand, '-' ,//web:dag)"/>
</datum>
```

- In XML deel 2 komt het gebruik van functies uitgebreider aan de orde

Voorbeeld XSLT en XPath

- Dit voorbeeld maakt van cursussen.xml een lijst van cursussen met alleen de naam van de cursus uit het bron document:

```
<xsl:stylesheet version="1.0" xmlns:cr="http://www.vijfhart.nl"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <cursussen>
            <xsl:for-each select="cr:cursussen/cr:cursus">
                <xsl:sort select="cr:naam"/>
                <cursus>
                    <naam>
                        <xsl:value-of select="cr:naam"/>
                    </naam>
                </cursus>
            </xsl:for-each>
        </cursussen>
    </xsl:template>
</xsl:stylesheet>
```

DEMO

Voorbeeld XSLT en XPath

- Het volgende voorbeeld levert hetzelfde resultaat op

```
<xsl:stylesheet version="1.0" xmlns:cr="http://www.vijfhart.nl"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/cr:cursussen">
        <cursussen>
            <xsl:for-each select="cr:cursus">
                <xsl:sort select="cr:naam"/>
                <cursus>
                    <naam>
                        <xsl:value-of select="cr:naam"/>
                    </naam>
                </cursus>
            </xsl:for-each>
        </cursussen>
    </xsl:template>
</xsl:stylesheet>
```

- Het attribuut match is aangepast waardoor ook select van for-each aangepast moet worden

pull versus push

- De elementen for-each en value-of doorlopen zelf gekozen elementen
- dit wordt de pull benadering genoemd:
- de stylesheet maakt gebruik van kennis over het xml document om benodigde informatie eruit te trekken
- Dit is goed te gebruiken als de structuur van het document bekend is en niet verandert
- Het element apply-templates past templates toe op elementen die de stylesheet tegenkomt
- Dit wordt de push benadering genoemd: het xml document wordt als het ware door templates van de stylesheets heen geduwd
- beter toepasbaar als de structuur van het document mogelijk uitgebreid wordt (XML = **Extensible** Markup Language)
- In de cursus XML2 wordt dit uitgebreid behandeld

Stylesheest en opmaak

- Een stylesheet kan ook gebruikt worden om XML om te zetten naar andere type bestanden
  - HTML
  - PDF
- Hierbij kunnen dan opmaak kenmerken worden gedefinieerd
- Daarnaast kunnen Cascading Style Sheets (CSS) gebruikt worden om XML documenten in een mooie opmaak te tonen
- Hiervoor is wel kennis van de betreffende bestands indelingen nodig

XML naar HTML Voorbeeld

- Dit voorbeeld zet cursussen.xml naar een HTML tabel

```
<xsl:stylesheet version="1.0" xmlns:cr="http://www.vijfhart.nl"
                mlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" version="4.01" encoding="UTF-8"/>
  <xsl:template match="/">
    <html>
      <head>
        <title>Cursussen</title>
      </head>
      <body>
        <table>
          <xsl:for-each select="cr:cursussen/cr:cursus">
            <tr>
              <td><xsl:value-of select="cr:naam"/></td>
              <td><xsl:value-of select="cr:prijs"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

DEMO

XML met CSS

- Met behulp van CSS kan aan verschillende elementen opmaak worden meegegeven
- Hierbij wordt per node een naam opgegeven
- Vorm:

```
elementnaam{
            opmaak-regel1:waarde1;
            opmaak-regel2: waarde2;
             …
            }
```

- Voorbeeld:

```
titel {
       background-color:red;
       font-family: Arial;
       font-size: 15pt;
       folor:#ffffff;
      }
```

XML met CSS

- Belangrijkste opmaak eigenschappen voor tekst:
  - font-family te gebruiken lettertype
  - font-size tekengrootte
  - font-style tekststijl (normaal, italic)
  - font-weight tekstdikte(none, bold, bolder, lighter)
  - color tekstkleur in hexadecimale waarde
  - text-indent tekst inspringen
  - text-align tekst uitlijnen
  - text-decoration tekst-decoratie (underline, overline, line-through)
  - line-height regelafstand
  - text-transform hoofdletter weergave (none, uppercase, lowercase)
- De gebruikte eenheid voor lengte is centimeter(cm), inches(in), punten (pt), pixels(px), en em spaces (em)
- Em space is de ruimte die de hoofdletter M inneemt voor het gegeven lettertype en grootte

XML met CSS

- Eigenschappen voor gebieden:
  - background-color — achtergrondkleur in hexadecimaal
  - background-image — achtergrond afbeelding
  - border-color — kleur van de rand in hexadecimaal
  - border-style — lijnstijl van de rand (bv. solid, dotted)
  - border-width — dikte van de rand
  - margin — grootte van de marge
  - padding — grootte van de padding

XML met CSS

- Het display attribuut geeft nog meer mogelijkheden voor weergave van een element:
  - none — element wordt niet getroond
  - block — het element wordt als block (nieuwe regel) getoond
  - inline — het element wordt inline getoond (geen nieuwe regel)
  - list-item — de elementen worden als een lijst getoond
  - table — de elementen worden als een tabel getoond
  - table-row — de elementen worden als een tabel rij getoond
  - table-column — de elementen worden als een tabel kolom getoond
  - table-cell — de elementen worden als een tabel cel getoond

DEMO

Automatisch stylesheet toepassen

- Met behulp van een PI kan een stylesheet automatisch worden toegepast als het document wordt geopend
- Voorbeelden

```
<?xml-stylesheet type="text/css" href="cdlijst.css"?>
<?xml-stylesheet type="text/xsl" href="nummers.xsl"?>
```

XML en XSL-FO

- FO staat voor Formatting Objects
- Bij deze techniek wordt XML getransformeerd naar een .fo document
- Voor FO wordt gebruik gemaakt van de namespace:
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
- Een FO document heeft altijd een <fo:root>
- De basis opmaak wordt gedefinieerd met <fo:layout-master-set>
- Met <fo:simple-page-master> worden een aantal pagina typen gedefinieerd te onderscheiden via het attribuut master-name
- Uit het .fo document kan bijvoorbeeld een .pdf, .rtf, of postscript document worden gegenereerd
- Voor de conversie van FO is een XSL-FO processor nodig, zoals Apache FOP

DEMO

Opdrachten Hoofdstuk 5 (succes)

Hoofdstuk 6 XML documenten verwerken

Leerdoelen:

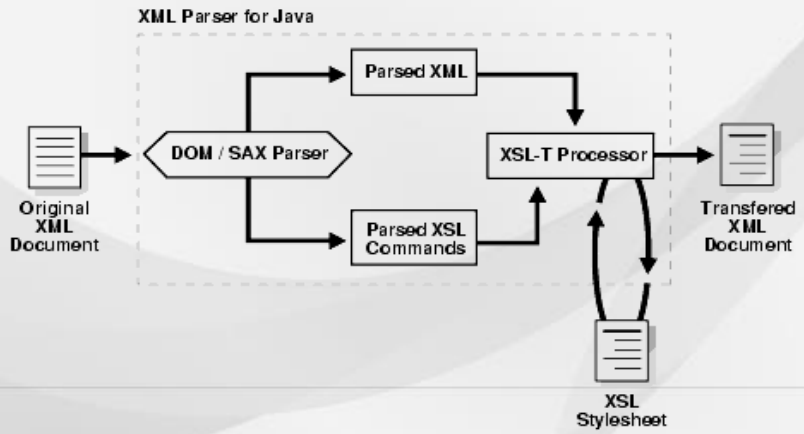
Een document kunnen benaderen met DOM

Een document kunnen benaderen met SAX

De verschillende tussen de beide methoden kunnen benoemen

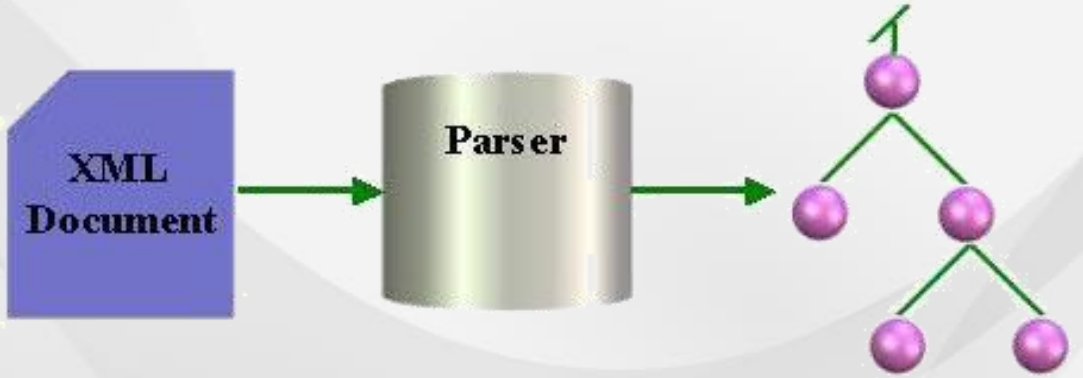
XML verwerken

- Als de XML documenten klaar zijn moeten ze verwerkt worden
  - Dit wordt wel parsing genoemd
- Verschillende applicaties kunnen XML verwerken
- Hierbij wordt gebruik gemaakt van API
  - Aplication Programming Interface
- Met een API wordt een verzameling definities vastgelegd
- Zo kunnen programmertalen als Java, JavaScript of .NET communiceren



DOM API

- Vaak wordt XML verwerkt volgens het Document Object Model (DOM)
- Bij deze manier van verwerking wordt een in memory boom structuur van het gehele XML document gemaakt
- Voordeel van DOM is de eenvoudige API en het makkelijker kunnen selecteren van een element
- Nadeel van DOM is de lagere performance bij grote XML documenten



Document Object Model

- Voorbeeld JavaScript functie om de DOM tree te creëren:

```
function laadDocument()
{
      xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
      xmlDoc.load(window.prompt("geef de naam van het XML
                                 document","cursussen.xml"));
      root= xmlDoc.documentElement;
}
```

Document Object Model

- Deze functie toont alle cursusnamen via getElementsByTagName:

```
function cursusLijst()
{
      cursusnamen=xmlDoc.getElementsByTagName("naam");
      namen = "cursussen<br>"
      for(i=0;i<cursusnamen.length;i++) {namen = namen +
                                          cursusnamen(i).text + "<br>"}
}
```

- In dit eenvoudige voorbeeld wordt met het hele XML document gewerkt, de referentie hieraan is xmlDoc

Simple API for XML (SAX)

- Een SAX parser beschouwt het XML document als een serie events
- SAX is efficiënter dan DOM
- Nadeel van SAX is de ingewikkeldere API en het is moeilijker een selectie op een element te doen

Simple API for XML (SAX)

- SAX gebruikt callback functies om aan te geven waar de parser binnen het XML document is aangekomen:
  - startDocument()
  - endDocument()
  - startElement(String uri, String naam, String elementNaam, Attributes attrs)
  - endElement(String uri, String naam, String elementNaam)
  - characters (char ch[], int start, int length)

XML en JAVA

- Voor werken met XML in java programmatuur zijn er verschillende mogelijkheden zoals
  - JAXB Java Architecture for XML Binding
    - Gebruikt om java Objecten te serialiseren naar XML
    - XML terug te lezen naar Objecten
  - JAX-WS Java API for XML Web Services
    - JAX-WS maakt het mogelijk om WSDL webservices in Java aan te maken
      WSDL beschrijft de functionaliteit van een webservice
      Met JAX-WS wordt een mapping gemaakt tussen SOAP berichten die binnenkomen, en aan te roepen Java methods
    - De SOAP specificatie definieert de envelope structuur, die wordt verzonden als SOAP-berichten (XML-bestanden) en terug gelezen naar objecten
  - JAX-RS De RESTFul variant van JAX-WS
    - Hierin wordt geen WSDL gebruikt: REST beschrijft een algemene set van operaties die op een RESTFul service kan worden aangeroepen

Bedankt voor uw aandacht!