Met XML van database naar LaTeX

Het automatisch publiceren van database informatie in LaTeX documenten

Keywords
conversie, transformatie, XML, XSLT, XPath, Saxon, database, Access, publiceren.

Abstract
In dit artikel wordt de geautomatiseerde aanmaak van het programmaboek voor de Bacheloropleiding Technische Wetenschappen bij de Hogere Defensie Opleidingen (HDO) toegelicht. Een belangrijk aspect hierbij vormt de omzetting van een Microsoft Access database naar LaTeX tabellen en bijlages in de uiteindelijke documentatie: XML-bestanden worden als tussenstap gehanteerd.

Inleiding en aanleiding

De Hogere Defensie Opleidingen (HDO, de samenvoeging van de Koninklijke Militaire Academie in Breda, het Koninklijk Instituut voor de Marine in Den Helder en het Instituut Defensie Leergangen in Rijswijk) zijn begonnen met de introductie van nieuwe opleidingsprogramma's volgens de Bachelor-Master structuur. In het kader van die transformatie is ook een opleiding Technische Wetenschappen (TW) ontwikkeld; een opleidingsteam heeft een ontwerp en de documentatie gegenereerd.

Het ontwerpproces kent een aantal uitgangspunten en karakteristieken:

□ Aangezien het team veelal geografisch gescheiden en in een uiteenlopende IT-omgeving werkt is bij dit proces zoveel mogelijk gebruik gemaakt van IT-hulpmiddelen. Zo vond de interne informatievoorziening via een Website plaats, waar met name een archief van alle tussenproducten ingericht was.

□ De standaard software in de defensieomgeving is Microsoft Office. Die draait op alle PC's en veel gebruikers zijn ermee vertrouwd. Daarom moesten Word brondocumenten en zoveel mogelijk Office applicaties gehanteerd worden in de ontwikkeling;

□ Desalniettemin is in een vroeg stadium door beide auteurs nagedacht over eenduidigheid en consistentie in de brongegevens, het werken in teamverband en beheersbaarheid van de gegevens op lange termijn. Dan kom je al gauw uit op het gebruik van een centrale database.

□ Daarbij stond het streven naar *Open Standards* voorop en dat bracht als oplossingsrichting al snel het gebruik van XML en LaTeX in beeld[1]. Vooral het artikel van Berend de Boer [1] gaf de doorslag om met XML en ConTeXt/LaTeX verder te gaan.

□ Gezocht is naar een zoveel mogelijk geautomatiseerd proces bij de aanmaak van documentatie, waaronder het programmaboek voor de bachelor Technische Wetenschappen (TW). Teksthoofdstukken in het programmaboek TW veranderen qua inhoud zeer beperkt. Daartoe is brontekst direct in LaTeX ingevoerd of heeft een eenmalige conversie met *Word2LaTeX* plaatsgevonden. Gedetailleerde curriculumgegevens veranderen frequent en zijn daarom als tabellen en bijlages in het boek opgenomen.

□ Belangrijk is daarbij een keuze voor het conversieproces en de integratie van de gegevens in de documentatie. Berend de Boer noemde in [1] de mogelijkheid om XML data direct om te zetten naar ConTeXt code, zonder gebruik te maken van de door hem beschreven ConTeXt macro's. Deze directe omzetting, en dan naar LaTeX, is voor het invoegen van curriculumgegevens in het programmaboek TW gebruikt, enerzijds omdat deze flexibeler gevonden werd, anderzijds omdat ConTeXt binnen het Koninklijk Instituut voor de Marine (KIM) niet gebruikt wordt.

De geautomatiseerde aanmaak van de documentatie is het onderwerp van dit artikel.

Wat is XML

Er is momenteel sprake van een XML-hype. Er kan geen programma meer uitkomen zonder dat de fabrikant uitgebreid verklaart dat het programma XML ondersteunt.

De afkorting XML staat voor *Extensible Markup Language*. Een bekende markup taal is HyperText Markup

Language (HTML). Het kenmerk van een *markup taal* is dat gegevens, weergegeven als reeksen van karakters, op een of andere manier gemarkeerd zijn, zodat de *betekenis* van die gegevens duidelijk wordt. Pas als de betekenis duidelijk is kan een computerprogramma of de mens er iets mee doen. De gegevens worden dan *informatie*. LaTeX is ook een markup taal, immers in LaTeX worden stukken tekst (secties, paragrafen, lijsten, titel etc.) gemarkeerd zodat een typesettingprogramma deze op de juiste manier kan weergeven.

Het verschil tussen HTML en LaTeX enerzijds en XML anderzijds is dat XML *extensible* is. HTML en LaTeX hebben een min of meer vaste set van *markup codes*, waarmee het gebruik van de informatie beperkt is tot één toepassing, te weten presentatie in een browser respectievelijk printen (typesetting) van de gegevens. XML daarentegen schrijft slechts de *syntax* voor van de markup codes en laat de betekenis ervan volledig vrij. Om iets met XML te kunnen doen, moeten eerst afspraken over de betekenis gemaakt zijn. De toegestane markup codes en hun onderlinge relaties en beperkingen worden in een *Document Type Definition (DTD)* vastgelegd, die ingevoegd wordt in het XML-bestand of waarnaar verwezen wordt. Er bestaan gestandaardiseerde DTD's, bijvoorbeeld XHTML (HTML versie die aan de XML standaarden voldoet), MathML (wiskunde) en XFRML (accountants).

Om de hype te relativeren, eerst een opsomming wat XML *niet* is (uit [2]):

□ XML is geen programmeertaal. XML kan niet uitgevoerd worden en er bestaan geen XML compilers;

□ XML is geen netwerkprotocol. XML-bestanden kunnen net als ieder ander bestand over een netwerk getransporteerd worden, waarbij gebruik gemaakt wordt van protocollen als HTTP, FTP, NFS;

□ XML is geen database. XML is een representatie van informatie en kan daarmee wel gebruikt worden om informatie tussen databases uit te wisselen;

□ XML is niet zaligmakend. Hoewel vele databases en teksverwerkers XML kunnen importeren en exporteren, maken ze meestal gebruik van eigen DTD's, zodat ze elkaars XML-bestanden jammer genoeg niet accepteren!

Om te laten zien wat XML *wel* is volgt hier een fragment uit een XML-bestand:

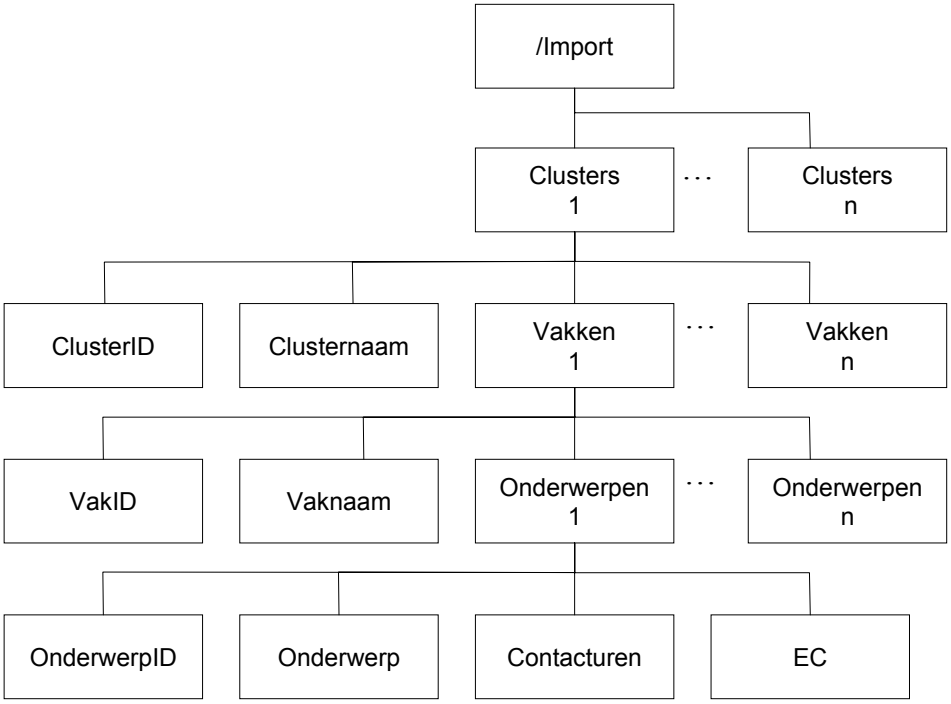
```
<?xml version="1.0" encoding="UTF-8"?>
<Import>
  <Clusters>
    <ClusterID>11</ClusterID>
    <Clusternaam>Wiskunde</Clusternaam>
    <Vakken>
      <VakID>25</VakID>
      <Vaknaam>Analyse 1</Vaknaam>
      <Vaksoort>Grondslag</Vaksoort>
      <Datum>19-11-2004</Datum>
      <Onderwerpen>
        <OnderwerpID>21</OnderwerpID>
        <Onderwerp>Integralen</Onderwerp>
        <Werkvorm>Werkcollege</Werkvorm>
        <Contacturen>8</Contacturen>
        <ECTS>0,6</ECTS>
      </Onderwerpen>
      <Onderwerpen>
        <OnderwerpID>22</OnderwerpID>
        .....
      </Onderwerpen>
    </Vakken>
    <Vakken>
      <VakID>27</VakID>
      <Vaknaam>Statistiek</Vaknaam>
      .....
    </Vakken>
  </Clusters>
  <Clusters>
    <ClusterID>10</ClusterID>
    .....
</Import>
```

In bovenstaande is te zien dat gegevens in XML gemarkeerd worden met zogenaamde *tags*: `<tag>Te markeren informatie</tag>`. Zo'n samenstel van tags en gegevens heet een *element*. Elementen representeren dus een stuk informatie. Markeren gaat met een begin `<tag>` en eind `</tag>`. Elementen kunnen ook leeg zijn: `<tag/>` (let op de plaats van de "/"). Elementen kunnen *genest* worden d.w.z. de gegevens kunnen zelf ook weer elementen bevatten. In dat geval is er sprake van een *gelaagde informatieboom*. Zie Figuur 1 voor een grafische representatie van bovenstaand XML-bestand.

Conversie van XML naar tekst

Op zich is XML al nuttig om informatie in een bestand vast te leggen. Pas echt krachtig wordt XML door de diverse gerelateerde standaarden en definities waarmee op een standaard manier met XML omgegaan kan worden. En, zoals het een echte *open standaard* betaamt, zijn deze standaarden gepubliceerd (`www.w3c.org`) en zijn er veel (gratis) programma's en bibliotheken te krijgen voor XML. De volgende standaarden zijn relevant voor dit artikel (er zijn er echter veel meer, zie [3] voor een compleet overzicht):

□ *XPath*: een krachtige set van commando's waarmee in een XML-bestand elementen *geselecteerd* kunnen worden;



Figuur 1. De gelaagde informatie in het XML voorbeeld.

□ *XSLT: Extensible Stylesheet Language Transformations*, een taal waarmee na selectie van elementen *iets met de informatie gedaan* kan worden. Elegant is dat XSLT-bestanden zelf qua formaat voldoen aan de XML standaard. XSLT maakt gebruik van XPath.

Zoals Berend de Boer in [1] al vermeldde, kan XSLT een XML-bestand transformeren naar:

□ een ander XML-bestand. Dan wordt de informatie in een nieuwe elementenstructuur (met een andere DTD) vertaald;

□ een HTML-bestand. Logisch, want HTML lijkt erg veel op XML;

□ een tekstbestand. En dat kan dus een LaTeX-bestand worden, immers in LaTeX worden alle formateringscommando's in tekst gecodeerd;

□ formatted output (zgn. XSL-FO), bijvoorbeeld naar PDF of naar TeX.

In dit artikel zullen we de XSLT transformatie van XML naar een *tekstbestand* demonstreren. Het doel zal zijn het eerder gegeven XML-bestand om te zetten naar onderstaande LaTeX code. Deze is na typesetting te herkennen als Figuur 2.

Hoofdstuk 1: Cluster- en vakbeschrijvingen

1.1 Wiskunde

Vakken	CU	EC
⇒Analyse 1	64	5
⇒Statistiek	45	4
...		

1.2 ...

Figuur 2. Resultaat van de conversie.

```
\chapter{Cluster- en vakbeschrijvingen}
\label{chap:archdetail}

\section{Wiskunde}
\label{sec:11}

\begin{tabular}{@{}l@{}lrr}\hline
     & Vakken & CU & EC \\\hline
\htmlref{$\Rightarrow$}{vak:25}&
                          Analyse 1 & 64 & 5\\
\htmlref{$\Rightarrow$}{vak:27}&
                          Statistiek & 45 & 4\\
```

```
%volgende vakken in wiskunde cluster
\hline
\end{tabular}
\newline

\section{....
%volgend cluster

%etc...
```

Hierin is te zien dat de cluster- en vakinformatie, afkomstig uit het XML-bestand, ingevoegd is tussen LaTeX commando's. Om dit te demonstreren het begin van het XSLT-bestand dat deze conversie verzorgt:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
     "http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"
                omit-xml-declaration="yes"/>
<!-- Einde header, begin XSLT code -->
<xsl:template match="/import|Import">
  <xsl:text>
\chapter{Cluster- en vakbeschrijvingen}
\label{chap:archdetail}
  </xsl:text>
```

De eerste regel is de identificatie van de gebruikte XML standaard. Ieder XML-bestand moet met een dergelijke regel beginnen. Te zien is dat XSLT gecodeerd is als XML. De tweede regel verwijst naar de DTD van XSLT, deze verwijzing is verplicht in XSLT. De derde regel geeft aan dat er alleen tekst geschreven mag worden. Commentaar in een XML-bestand staat tussen `<!--` en `-->`.

Bij het uitvoeren van het XSLT-bestand (*hoe* dat uitgevoerd wordt zien we verderop) wordt door het XML-bestand heen gelopen en wordt telkens gekeken welk "template" van toepassing is. Templates worden gedefinieerd met het element `<xsl:template match=...>`, dat als gegevens de XSLT commando's bevat die bij een "match" toegepast moeten worden. In ons geval bevat het XSLT-bestand slechts één `template`, te weten `<xsl:template match="/import|Import">`. Daarmee wordt het element `/import` *of* `/Import`[2] uit het XML-bestand gematched. De gegevens in het element `Import` worden daarna als *huidig niveau* beschouwd voor de verdere verwerking. De "/" staat voor de *root*, deze is met "|" (*of-operator*) afkomstig uit XPath. De XPath commando's staan in XSLT tussen dubbele aanhalingstekens.

Het `<xsl:text>Informatie</xsl:text>` commando schrijft de `Informatie` direct naar de output, met behoud van lege regels, spaties etc. In bovenstaand voorbeeld worden daarmee de LaTeX commando's geschreven.

We gaan nu verder met het invoegen van de gewenste informatie:

```
  <xsl:for-each select="Clusters">
    <xsl:text>

\section{</xsl:text>
    <xsl:value-of select="Clusternaam"/>
    <xsl:text>}
\label{sec:</xsl:text>
    <xsl:value-of select="ClusterID"/>
    <xsl:text>}

\begin{tabular}{@{}l@{}lrr}\hline
&amp; Vakken &amp; EC &amp; CU \\\hline
    </xsl:text>
```

Met `<xsl:for-each select="Clusters">` worden alle cluster-elementen om de beurt geselecteerd en als *huidig niveau* beschouwd. Per cluster wordt eerst een lege regel gevolgd door `\section{` geschreven. Daarna moet de clusternaam ingevoegd worden: dat gebeurt met `<xsl:value-of select="Clusternaam"/>`. Deze zoekt het actuele element `Clusternaam` en copieert de bijbehorende informatie (`Wiskunde`) naar de uitvoer. Merk op dat we inmiddels twee niveau's diep in het XML-bestand zitten (/Import/Clusters) in de eerste `Cluster` en dat `Clusternaam` dus *eenduidig bepaald* is.

Hierna wordt het `\section` commando afgesloten, waarna `\label{sec:11}` (immers het *huidige* `ClusterID` is 11) geschreven wordt. Belangrijk is dat het XSLT-bestand de volgorde van de informatie in de uitvoer bepaalt, niet het XML bronbestand. XPath commando's in XSLT selecteren de gewenste elementen. Als je bijvoorbeeld de naam van het vijfde cluster wilt hebben, en je zit op het niveau `<Import>`, dan kan dat met

```
<xsl:value-of select="Clusters[5]/Clusternaam"/>
```

Ook selecties zijn mogelijk:

```
<xsl:value-of select="Clusters
          [Clusternaam='Wiskunde']/ClusterID"/>
```

geeft het `ClusterID` van het cluster met `Clusternaam = 'Wiskunde'`.

Na het schrijven van het `ClusterID` wordt de LaTeX tabeldefinitie geschreven. Merk daarbij op dat de "&" in XML een speciaal teken is en daarom altijd als "`&`" (voor ampersand) geschreven moet worden. Het hebben van verschillende speciale tekens en afwijkende tekensets in XML en LaTeX compliceert de conversie erg!

Na deze basis volgt de rest van de XSLT code:

```
    <!--List of Vakken within Cluster-->
    <xsl:for-each select="Vakken">
      <xsl:sort select="Vaknaam"/>
```

```
      <!-- create a reference -->
      <xsl:text>
\htmlref{$\Rightarrow$}{vak:</xsl:text>
      <xsl:value-of select="VakID"/>
      <xsl:text>} &amp; </xsl:text>
      <xsl:value-of select="Vaknaam">
      <xsl:text> &amp; </xsl:text>
      <!-- compute total Contacturen -->
      <xsl:value-of select=
              "sum(Onderwerpen/Contacturen)"/>
      <xsl:text> &amp; </xsl:text>
      <!-- compute total EC -->
      <xsl:value-of select=
     "round(10*sum(Onderwerpen/ECTS)) div 10"/>
      <xsl:text>\\</xsl:text>
    </xsl:for-each> <!--Vakken-->
    <xsl:text>
\hline
\end{tabular}
\newline
    </xsl:text>
  </xsl:for-each> <!--Clusters-->
</xsl:template> <!--template end-->
</xsl:stylesheet>
```

We zien hier de volgende relevante zaken:

□ Het `<xsl:sort>` commando sorteert de uitvoer, in dit geval de vakken, op vaknaam.

□ Met `\htmlref{$\Rightarrow$}{vak:xx}` (uit het HTML package, onderdeel van LaTeX2HTML) wordt een hyperlink gemaakt. Als de ID's in het XML-bestand *uniek* zijn, geeft dat correcte links door het hele document.

□ Het `<xsl:value-of select="sum(Onderwerpen /Contacturen)"/>` commando sommeert de getallen die in het *huidige vak* per onderwerp (dat kunnen er meerdere zijn) in het element `Contacturen` gegeven zijn. `sum()` Is een XPath functie.

□ XSLT gaat niet goed om met de sommatie van *niet-gehele* getallen. `<xsl:value-of select="round (10*sum(Onderwerpen/ECTS)) div 10"/>` sommeert de EC's (European Credits, maat voor studielast) per vak en doet een afronding. Ook `round()` en `div` komen uit XPath.

Tot zover de behandeling van de conversie van XML naar LaTeX met XSLT/XPath. Realiseer je dat hier slechts een fractie behandeld is van de mogelijkheden die XSLT en XPath bieden! Zie [3] voor een gedetailleerd overzicht van alle mogelijkheden. We gaan nu kijken hoe het gehele documentatiesysteem gerealiseerd is.

Het document systeem

In dit deel wordt behandeld hoe de hiervoor beschreven conversie gebruikt wordt om informatie vanuit een master-database via LaTeX te publiceren. Figuur 3 geeft een overzicht van de diverse stappen met de tussenformaten.

Stap 1: De master-database
Als relationeel database-programma is *Microsoft (MS) Access* gebruikt. Dit heeft de volgende voordelen:

□ Een *relationele database* garandeert consistentie van de informatie, omdat een gegeven slechts eenmaal in een *tabel* opgeslagen wordt, met een *unieke identificatie*. Informatie in verschillende tabellen kan vervolgens onderling *gerelateerd* worden. De tabellen met de onderlinge relaties kunnen worden weergegeven in een zgn. *Entity-Relation Diagram (ERD)* of *Informatie model*. Figuur 4 geeft het vereenvoudigde informatiemodel van de gebruikte master-database. Daarin is de informatie te herkennen die we ook in het XML-bestand gezien hebben, zoals clusters, vakken en onderwerpen.

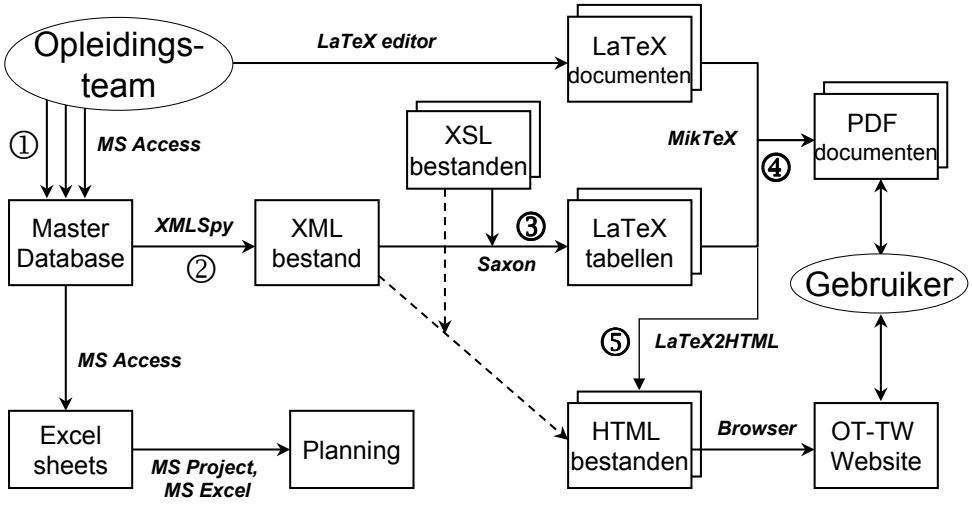
□ MS Access is het standaard database-programma binnen Defensie en is een redelijk eenvoudig te leren. Het heeft goede invoer- en uitvoermogelijkheden die gebruikt zijn bij de vulling van de master-database door het opleidingsteam.

□ MS Access voorziet in een replicatiemechanisme waardoor gegevensinvoer parallel gedaan kan worden. Bij het samenvoegen in de master-database worden eventuele conflicten gemeld, die dan opgelost kunnen worden.

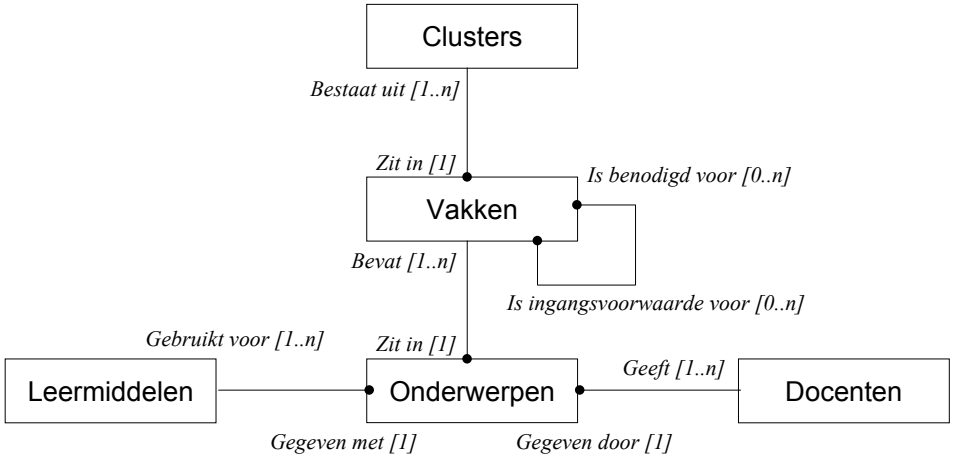
□ In MS Access kunnen geavanceerde programmeerfuncties ingebouwd worden met Visual Basic. Dit is gebruikt om volgorderelaties binnen clusters automatisch te berekenen.

Naast de in dit artikel beschreven LaTeX uitvoer kunnen vanuit de master-database eenvoudig allerlei overzichten gemaakt worden voor bijvoorbeeld analyses, brainstorm sessies en planningen. De master-database is de primaire bron voor overzichten over de opleiding.

Stap 2: Maken van de XML-bestanden uit de master-database
Ondanks dat de nieuwste MS Access versie volgens de fabrikant XML ondersteunt, wordt het programma *XMLSpy Professional Edition* [4] gebruikt om vanuit MS Access de juiste XML-bestanden te maken[3]. Belangrijk is dat in deze stap een gelaagd XML-bestand conform Figuur 1 gemaakt wordt door informatie uit meerdere



Figuur 3. Overzicht van de conversie van de informatie.



Figuur 4. Het informatiemodel van de master-database (vereenvoudigd).

tabellen te combineren tot één uitvoer, daarbij gebruik makend van de relaties. Dit is niet triviaal omdat daarvoor selectie-opdrachten (*queries*) nodig zijn. Of MS Access deze mogelijkheid biedt is ons onbekend, de gebruikte MS Access versie (versie 2000) kent geen XML uitvoer.

XMLSpy haalt de benodigde informatie direct uit het MS Access `.mdb`-bestand. In totaal zijn er 5 XML-bestanden nodig vanuit de master-database. Voor de verdere bewerking moeten de XML-bestanden omgezet worden naar *UTF-8*, dat afwijkt van de standaard windows karakterset. Dit kan met XMLSpy.

Stap 3: Converteren van informatie uit de XML-bestanden naar LaTeX

De techniek van de XML-naar-LaTeX conversie is al uitgebreid aan de orde gekomen, we gaan nu kijken hoe dat gedaan is.

De conversie wordt uitgevoerd door een zgn. *XSLT-processor*. Deze leest de XML en XSLT-bestanden en produceert de gewenste uitvoer. Voorbeelden van (freeware) XSLT-processoren zijn MsXML (in Internet Explorer), Saxon en Xalan.

Voor ons systeem hebben we gebruik gemaakt van *Instant-Saxon* [5]. Het voordeel van dit programma is dat het via een *Command Line Interface (CLI)* aan te roepen is, op de volgende manier: `saxon -o apparch.tex clusters.xml genclusterlijst.xslt`, waarbij `apparch.tex` het uitvoerbestand is. Omdat Saxon via een CLI aangeroepen wordt, kunnen meerdere commando's gecombineerd worden in één *batchbestand* dat met een muisklik uitgevoerd wordt. De volgende stappen worden doorlopen voor de conversie naar LaTeX.

Preprocessing. Deze XSLT conversie is nodig voor een aantal taken:

- conversie van de "," in numerieke waarden naar ".". MS Access hanteert de Nederlandse conventie, XML gaat uit van de Amerikaanse conventie[4];
- conversie van *harde returns* in de database informatie naar `\newline` commando's om de formattering van de gegevens in de master-database te behouden;
- conversie van *speciale karakters*, gecodeerd in XML of UTF-8 naar de LaTeX equivalent. Voorbeelden zijn: "`&iuml;`" naar "`\"{\i}`", "`>`" naar "`$>$`" en "`&`" naar "`\&`". Om gebruik van LaTeX commando's in de master-database informatie mogelijk te maken, worden de "\", "{" en "}" *niet* geconverteerd.

Genereren. Ons systeem maakt gebruik van meerdere XSLT conversies. In totaal worden voor het programmaboek vijf XSLT-bestanden gebruikt voor 16 XSLT conversies, die resulteren in evenzovele LaTeX bestanden. Die bestanden bevatten tabellen, annexen etc. XSLT kent een *parameter mechanisme*, waarmee vanuit de CLI extra besturingsparameters aan XSLT meegegeven wordt. Daardoor zijn er "maar" vijf XSLT-bestanden nodig.

Opslaan. Door meerdere batch-bestanden te maken kan de informatie uit de XML-bestanden naar meerdere sets van LaTeX-bestanden geconverteerd worden. In ons geval zijn dat naast het programmaboek ook een studiegids en een opleidingsreglement. Het systeem garandeert *eenduidige informatie over meerdere documenten*. Door de LaTeX-bestanden meteen in de juiste TeX directory te zetten blijft het geheel overzichtelijk.

Stap 4: Compileren van het uiteindelijke LaTeX document
De gegenereerde bestanden zijn voorzien van vooraf gedefinieerde labels en worden met `\include` of `\input` commando's in het LaTeX moederdocument ingevoegd. Dat `.tex` moederdocument bepaalt de structuur van het document en bevat de diverse hoofdstukken met grote delen tekst. Met `pdflatex` wordt het geheel geconverteerd naar een Acrobat `.pdf`-document, waarbij de `hyperref` package wordt gebruikt om een navigeerbare browser versie te verkrijgen. Alle tabellen, gedetailleerde annexen en hyperlinks in het document bevatten *eenduidige informatie*, ze komen immers uit één bron.

Stap 5: Maken van een HTML document
Een extra stap is de conversie van LaTeX naar HTML, via LaTeX2HTML. Hiermee wordt het programmaboek op de in de inleiding genoemde TW website gepubliceerd. Het oorspronkelijke plan was om dit direct uit het XML-bestand te gaan doen met Java-scripts, met een zoek- en selecteerfunctie (stippellijn in Figuur 3). Door de complexe structuur van de opleiding bleek dit moeilijker dan gedacht en is dit wegens tijdgebrek niet verder opgepakt.

Conclusie

De ervaringen met dit documentsysteem zijn als volgt samen te vatten:

□ De benodigde hulpmiddelen voor de opzet van de geautomatiseerde conversie zijn in korte tijd geïdentificeerd (aangekochte software) of ontwikkeld (XSLT en XPath). Nodig zijn een beperkt budget, enige programmeerervaring en wat doorzettingsvermogen; het totale project is door de auteurs binnen enkele maanden in deeltijd gerealiseerd.

□ De doelstellingen zijn gehaald: er wordt gewerkt met eenduidige brongegevens, de gegevens zijn in een open standaard beschikbaar en er is een snel en betrouwbaar documentsysteem tot stand gebracht (van database naar 225 pagina's programmaboek binnen enkele minuten). Het heeft ertoe geleid dat de HDO deze werkwijze nu waarschijnlijk voor al haar documentatie zal overnemen. Ook een aantal TU faculteiten heeft belangstelling getoond.

□ Het vastleggen van layout-details in grote curriculumtabellen via een XSLT style sheet is geen sinecure. Na de conversie kan de layout niet meer gerepareerd worden (dit zou na iedere conversie opnieuw moeten gebeuren) dus moet de layout, kloppend voor alle informatie, in het XSLT-bestand vastgelegd worden. Ver-

der maken keuzemogelijkheden en uitzonderingen in in het curriculum het geheel complexer dan in dit artikel beschreven.

□ De auteurs zijn er niet in geslaagd om andere teamleden (voor zover nog geen gebruiker) over de streep te trekken om LaTeX te gebruiken. De drempel is te hoog gebleven, ondanks demo's in vriendelijkere (WYSIWYG) omgevingen zoals SciWriter of Publicon. Ook het stringente beleid voor software installatie binnen Defensie speelt mee.

□ Dit project was tevens bedoeld om te onderzoeken of ook de curriculuminhoud zelf (het lesmateriaal) eenduidig en gemakkelijk in XML vast te leggen was, om van daaruit automatisch HTML browser files en via LaTeX `pdf` hardcopies te genereren. In dit opzicht is de conclusie negatief: eenduidigheid is maar al te moeizaam te verkrijgen (pakketten gebruiken allerlei definities voor hun XML) en ook betrouwbare hulpmiddelen voor de omzetting van bestaand LaTeX materiaal zijn niet gevonden. Daar is dus voorlopig gekozen voor `.tex` bronfiles.

Noten

1. Geen der auteurs had eerdere ervaring met XML; de eerste auteur bezat enige ervaring met TeX in het kader van de aanmaak van software-documentatie bij de Koninklijke marine, de tweede auteur is sinds 1987 een frequente gebruiker van computer typesetting.
2. Het element `Import` wordt door omzetting naar XML gemaakt en heeft geen betekenis voor de documentgeneratie.
3. XMLSpy Professional Edition voorziet o.a. in een XSLT programmeeromgeving, die gebruikt is voor de ontwikkeling van de XSLT-bestanden
4. Er zijn andere mogelijkheden bekend, maar voor ons was deze manier de meest eenvoudige.

Referenties

[1] Boer, B. de, EuroTeX^{2}001 presentatie "From database to presentation via XML, XSLT and ConTeXt", *MAPS 26* (2001), 27 - 39.
[2] Harold, E.R. en Scott Means, W., *XML in a Nutshell, 2nd Edition*, O-Reilly, 2002.
[3] Sall, K.B., *XML Family of Specifications*, Addison-Wesley, 2002.
[4] *XMLSpy Professional Edition, Version 2004 rel. 3*, http://www.altova.com, 2004.
[5] Kay, M., *Instant Saxon Version 6.5.3*, http://saxon.sourceforge.net, 2003.

Oscar Boot, Frans Absil
Sectie Sensor-, Wapen- en Commandosystemen
Koninklijk Instituut voor de Marine
Den Helder
o.boot@kim.nl