by Floris Lambrechts
<floris/at/linuxfocus.org>

About the author:

Ich war der niederländische LinuxFocus Editor für einige Zeit. Ich studiere 'Industrieingenieur für Elektronik' in Leuven, Belgien und verbummle meine Zeit mit Linux, PHP, XML, LinuxFocus und beim Lesen von Büchern von Stephen Hawking und momentan Jef Raskin ('The Human Interface').

Erste Schritte in XML

Abstract:

Dieser Text ist eine wirklich knappe Einführung in XML. Dabei werden wir Eddy, die Metakatze, die XML Syntaxregeln und einige DTDs kennenlernen. Aber keine Angst, das wird ja alles erklärt ;–)

Einleitung

Im Sommer 2001 trafen sich einige der LinuxFocus Editoren in Bordeaux während des LSM. Viele Gespräche und Diskussionen drehten sich bei der LSM Gruppe für Interessierte in Dokumentation um das gleiche Thema: XML. Lange (fröhliche) Stunden vergingen, um zu klären, was XML genau ist und wie man es einsetzen kann. Und um genau das soll es in diesem Artikel gehen.

Ich möchte die Gelegenheit nutzen um Egon Willighagen und Jaime Villate für ihre Einführung in XML zu danken. Dieser Artikel basiert zum Teil auf Jaimes Informationen, die ich natürlich weiter unten auch verlinkt habe.

Was ist XML?

Wir Dokumentationsleute wussten schon immer, was XML ist, zumindest mehr oder weniger. Eigentlich ist die Syntax wie bei HTML und es ist bloß eine weitere Auszeichnungssprache wie SGML und HTML, richtig? Richtig. Aber es geht noch weiter.
XML besitzt einige Eigenschaften, die es zu einem sehr nützlichen Datenformat machen, mit dem man fast alles erfassen kann. Es scheint, man kann damit selbst die komplexesten Sachen beschreiben und trotzdem bleibt es für den Menschen verständlich und läßt sich einfach durch Programme verarbeiten. Wie kann das sein? Untersuchen wir die verheißungsvolle Sprache etwas näher.

Eddy, die Metakatze

Zuerst einmal, XML ist eine *Auszeichnungssprache (markup language)*. Dokumente, die in einer solchen Auszeichnungssprache verfasst wurden, enthalten 2 Dinge: *Daten* und *Metadaten*. Falls jemand weiß, wie man Daten definiert, wäre ich froh, wenn er es mir mitteilen könnte. Bis dahin sprechen wir lieber über Metadaten ;). Metadaten stellen Informationen bereit, die einen Kontext oder eine Bedeutung für die Daten liefern. Ein einfaches Beispiel: nehmen wir den Satz: *'Meine Katze heißt Eddy'*. Ein Mensch würde schlussfolgern, dass *'Katze'* die Bezeichnung für eine Tierart ist und *'Eddy'* der Name des Tiers. Computerprogramme sind keine Menschen und wissen das alles nicht. Deshalb benutzt man Metadaten, um den Daten eine Bedeutung zu geben (hier in XML Syntax natürlich):

```
<satz>
  Meine <tier>Katze</tier> heißt <name>Eddy</name>.
</satz>
```

Jetzt kann sogar der dümmste Computer sagen, dass 'Katze' eine Art und 'Eddy' ein Name ist. Wollen wir nun ein Dokument erstellen, in dem alle Namen blau und alle Arten rot dargestellt werden, macht es uns XML relativ einfach. Wir würden also erhalten:

Meine Katze heißt Eddy.

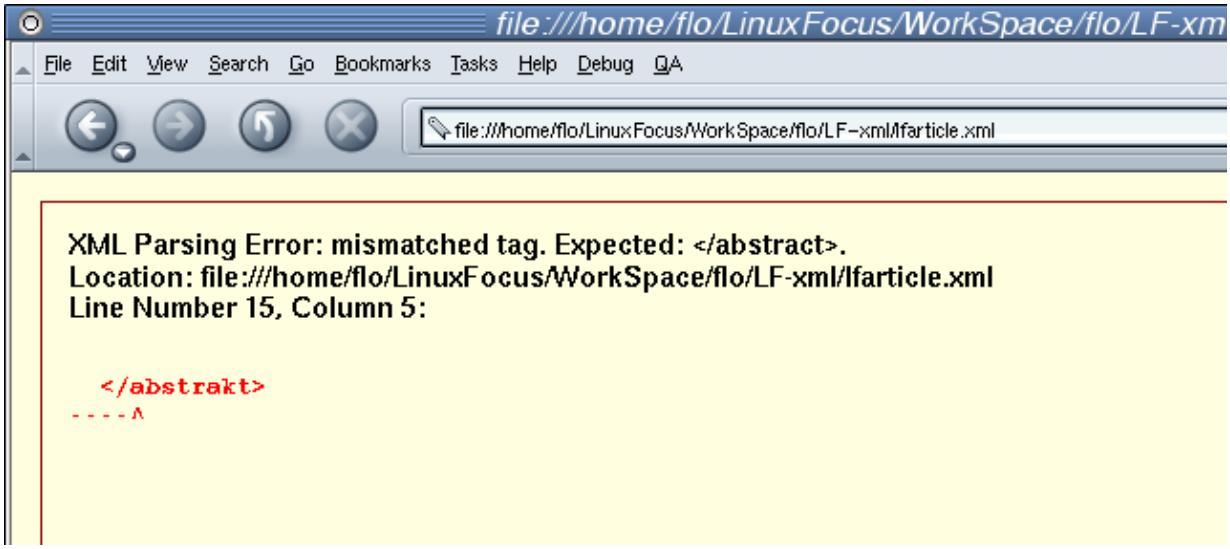
Nun könnte man, theoretisch zumindest, die Darstellungsinformationen (in unserem Fall die unterschiedlichen Farben) in einer Extradatei, einem Stylesheet, ablegen. Dadurch trennen wir Inhalt und Darstellung. Einige betrachten das als den Heiligen Gral des Webdesigns™. Bis jetzt haben wir aber noch nichts besonderes getan; Metadaten hinzufügen ist die Grundaufgabe jeder Markupsprache. Also was ist an XML so besonders?

Die Syntaxregeln

XML hat eine sehr strenge Syntax. So muss z.B. in XML jedes <tag> von einem </tag> geschlossen werden. [Hinweis: da es etwas umständlich wäre zu schreiben <tag></tag>, wenn eigentlich nichts dazwischen liegt, kann man auch schreiben <tag /> und sich einige Arbeit ersparen].
Eine weitere Regel ist, dass man keine Tags vermischen darf. Man muss die Tags in der umgekehrten Reihenfolge schließen, in der man sie geöffnet hat. Folgendes wäre nicht zulässig:

<B> **Fettdruck** <I> ***Fett– und Kursivdruck*** </B> *Kursivdruck* </I>

Die Regel besagt, dass man das </I> Tag vor dem </B> schließen muss.
Man muss weiterhin darauf achten, dass _alle_ Elemente in einem XML Dokument von Tags eingeschlossen werden müssen (außer die 2 äußeren Tags natürlich!). Deshalb haben wir in dem Beispiel weiter oben den Satz durch das <satz> Tag eingeschlossen. Ohne das Tag wären einige Wörter nicht von Tags umgeben worden. Das ist ein Punkt, an dem einen die XML Syntaxregeln in den Wahnsinn treiben können.



Mozilla's Syntaxüberprüfung bei der Arbeit...

Eine strenge Überprüfung der Syntaxregeln hat aber auch seine Vorteile: es sorgt für Ordnung. Seit XML diesen strikten Regeln folgt, ist es für Programme sehr einfach, die Dokumente zu lesen. Weiterhin sind die Daten in den Dokumenten sehr gut strukturiert, was es für Menschen auch einfacher macht.
Mit XML ist es sogar möglich, Datenbanken zu beschreiben (versuch das mal mit HTML! :p). Genau das hat Egon Willighagen für die niederländische Ausgabe des LinuxFocus getan. Sein Artikel ist unten in der Linkliste aufgeführt.
Wenn man sich mit den Syntaxregeln gut anfreundet, dann kann man sich durch sie einiges an Arbeit erledigen lassen. Dazu muss man aber die DTDs nutzen...

Die DTD

In unserem kleinen 'Eddy, die Metakatze' Beispiel haben wir unsere eigenen XML Tags erfunden. Es ist klar, so ein kreativer Akt wird von den Wächtern der Syntax nicht toleriert! Die 'Men in Blue' wollen wissen was wir tun, wann und (wenn möglich) warum. Aber hey, kein Problem, man kann ja alles in der DTD erklären...

Mit der DTD kann man neue Tags einführen. Genauer gesagt, man kann komplett neue Sprachen erfinden, solange diese der XML Syntax folgen.
Die DTD (**D**ocument **T**ype **D**efinition) ist eine Datei mit den Beschreibungen zur XML Sprache. Es handelt sich eigentlich nur um eine Liste mit möglichen Tags, deren Attributen und Kombinationen. Die DTD beschreibt was in der Sprache möglich und ummöglich ist. Wenn man also über eine 'XML Sprache' spricht, meint man eine spezielle DTD.

Regeln aufstellen

Manchmal *zwingt* die DTD einen dazu, etwas bestimmtes zu tun. So kann einen die DTD z.B. zwingen, ein Tag mit dem Titel des Dokuments einzufügen. Das ist eine praktische Sache, denn es gibt Software (z.B. ein Emacs Modul), die automatisch für die Erstellung der benötigten Tags sorgt.
Dadurch werden einige Teile der Dokumentenstruktur automatisch ausgefüllt. Da die Syntax so strikt und wohldefiniert ist, dient sie als Leitfaden beim Verfassen eines Dokumentes. Und wenn dann doch mal ein Fehler passiert, wie ein vergessenes Endtag, sorgen die Syntaxregeln für die nötigen Informationen zur Behebung. In der realen Welt wird einem gesagt, du darfst das und das nicht, in der XML Welt wird gesagt: 'Syntaxfehler in Zeile xx :' :)
Da einem die Regeln diese Arbeit abnehmen, kann man sich auf die eigentliche Aufgabe konzentrieren.

In the mix

Es sei noch auf eine weitere großartige Eigenschaft von XML hingewiesen. Man kann mehrere DTDs mischen und so auf verschiedene Datentypen im selben Dokument zurückgreifen.

Dieser Mix wird durch XML Namensräume (namespaces) realisiert. Nehmen wir an, man hat die Docbook DTD im XML Dokument integriert ('dbk' *Prefix* in diesem Beispiel).
Jedes Docbook Tag kann sofort im Dokument in folgender Form (sagen wir mal es gibt folgendes Docbook Tag: <ein_tag>) genutzt werden:

<dbk:ein_tag> einige Worte </dbk:ein_tag>

Mit den Namensräumen kann man jedes Tag und jedes Attribut jeglicher XML DTD ansprechen. Daraus ergibt sich eine Welt von Möglichkeiten, wie man im folgenden Kapitel sehen kann...

Verfügbare DTD's

Hier ist eine kleine Sammlung von DTDs, die bereits (zumindest teilweise) genutzt werden.

- **Docbook−XML**
  Docbook ist eine Sprache zum Schreiben von strukturierten Dokumenten wie Büchern und Schriften. Aber es gibt noch andere Anwendungsgebiete. Docbook ist eigentliche eine SGML DTD (SGML ist die Muttersprache von XML), aber es gibt auch eine sehr populäre XML Version. Es ist überhaupt eine der wohl am meist gebrauchtesten XML DTDs.

- **MathML**
  MathML ist die Mathematical Markup Language, mit der man mathematische Ausdrücke und Formeln beschreiben kann. Es ist ein sehr wichtiges Werkzeuge für die Mathewelt. Die Chemiker müssen aber nicht auf ihre Mathematikkollegen neidisch sein. Für sie gibt es die CML, Chemical Markup Language. Übrigens unterstützt Mozilla 1.0 nun standardmäßig MathML.

- **RDF**
  RDF ist das Resource Description Framework. Es wurde entwickelt, um bei der Kodierung und Wiedergewinnung von Metadaten zu helfen. In der Praxis wird es sehr häufig von Webseiten genutzt, um anderen Seiten mitzuteilen, welche Neuigkeiten angezeigt werden. So nutzt die niederländische Seite linuxdot.nl.linux.org z.B. RDF Dateien von anderen Seiten, um einen Newsticker bereitzustellen. Die meisten populären Newsseiten (wie z.B. Slashdot) haben eine RDF Datei verfügbar und so kann man deren Neuigkeiten einfach in einer eigenen Spalte auf der eigenen Homepage anzeigen lassen.

- **SOAP**
  SOAP steht für Simple Object Access Protocol. Diese Sprache wird für die Interprozesskommunikation genutzt (Datenaustausch und entfernte Funktionsaufrufe). Mit SOAP können Prozesse z.B. über das HTTP Protokoll (Internet) miteinander kommunizieren. Ich denke, Atif hier von LF kann mehr darüber erzählen, einfach mal bei den Links schauen :−)

- **SVG**
  Scalable Vector Graphics. Das Trio PNG, JPEG2000 und SVG soll die Zukunft von Bildern im Netz sichern. PNG wird die Rolle von GIF (verlustfrei komprimierte Bilder mit Transparenz) übernehmen und JPEG2000 wird irgendwann wohl .jpg ablösen (verlustbehaftete Kompression mit einstellbarer Qualität). SVG ist kein Bitmap basiertes Format, sondern Vektorgrafik. Das bedeutet Bilder werden nicht durch Bildpunkte sondern durch mathematische Gebilde (Linien, Rechtecke, ...) beschrieben.

SVG bietet weiterhin Funktionen für Skripts und Animationen, man kann es mit Flash von Macromedia vergleichen. Man kann JavaScript in .svg Dateien speichern und dieser Code kann neuen JavaScript Code einfügen. Richtig flexibel, oder?
SVG ist aber noch sehr neu. Momentan gibt es nur ein qualitativ hochwertiges Browser Plugin von Adobe für Windows & Mac. Mozilla arbeitet an einem integrierten SVG Viewer, aber dieser ist noch nicht fertig und man muss eine spezielle Version des Browsers herunterladen, um es verwenden zu können.
Hinweis: .svg Dateien können sehr groß werden; deshalb werden oft .svgz Dateien benutzt. Diese sind komprimierte Versionen unter Nutzung des gzip Algorithmus.

- **XHTML**
  XHTML ist die XML Variante von HTML Version 4.01. Wegen der strikten XML Syntax bestehen einige Unterschiede. Einige Dinge, die man von HTML gewohnt war, sind in XHTML nicht mehr gültig. Andererseits ist eine richtige XHTML Seite auch eine gültige HTML Seite. Das Programm HTML tidy kann eine HTML Seite nach XML konvertieren.

- **andere**
  Viele neue Dateiformate nutzen XML, oft in Kombination mit .gz oder .zip Kompression. Z.B.: KOffice Dateien werden in XML DTDs beschrieben. Das ist sehr hilfreich, da dies die Kombination von verschiedenen Funktionalitäten in einem Dokument ermöglicht. So kann man in einem KWord Dokument durchaus auch eine KChart Tabelle integrieren.

Links

Das W3C, World Wide Web Consortium
Mit Informationen zu XML, MathML, CML, RDF, SVG, SOAP, XHTML, Namespaces...
www.w3.org

Einige Sachen von Jaime Villate (evt. erst durch einen Onlineübersetzer jagen, damit man es lesen kann)
Einführung in XML (in Spanisch)
Wie erstellt man HTML mit XML(in Spanisch)
LSM Folien

HTML tidy, das Programm:
www.w3.org/People/Raggett/tidy

Docbook
www.docbook.org

Mozilla.org's SVG Projekt
www.mozilla.org/projects/svg

Relevante LinuxFocus Artikel:
Benutzen von XML und XSLT zum Bauen von LinuxFocus.org
PDF Dokumente erzeugen mittels DocBook