XML und Clarion

hi allesamt,

1. Vorbemerkungen

1. Vorbemerkungen

- XML eXtensible Markup Language – gibt Daten eine Struktur Info
  - Substandart von SGML (Standard Generalized ML)
  - Definition eigener Tag's
  - wohlgeformt – keine Syntaxfehler
  - gültig – keine Struktur Fehler (validiert) und wohlgeformt

- „XML ist ASCII mit Regeln für die innere Struktur“
- ... und das kann schließlich jeder !

1. Vorbemerkungen

- XML
  - Schema
    - DTD
    - XSD

    - Definition & Validierung von Daten/Strukturen
  - XSL
    - XSL-FO - Formatierung
    - XSLT - Transformation
    - XPath - Navigation
  - diverses
    - XQuery - Abfrage Sprache
    - XForms - Browser Formulare
    - XLink - Links im XML Doc
    - XPointer - Links zu einem XML Doc
    - VoiceXML - Sprache in XML
    - SOAP - Simple Object Access Protokol (XML)
    - WSDL - Web Services Description Lang. (XML)
    - SMIL - Synchronized Multimedia Integration Lang.
    - ...

2.1. Grundlagen


```
<cdsammlung>
 <cd>
  <gruppe>Haggard</gruppe>
  <titel>Eppur Si Muove</titel>
  <jahr>2004</jahr>
 </cd>
 <cd>
  <gruppe>Blutengel</gruppe>
  <titel>Demon Kiss</titel>
  <jahr>2004</jahr>
 </cd>
 <!-- weitere cd's -->
</cdsammlung>
```

- Tag basiert
- Attribute basiert

Datensatz

```
<cdsammlung>
 <cd gruppe="Haggard" titel="Eppur Si Muove" jahr="2004" />
 <cd gruppe="Blutengel" titel="Demon Kiss" jahr="2004" />
</cdsammlung>
```

2.1. Grundlagen


```
...
<gnr V="31347">
 <allgemein>
  <service_tmr V="2004-07-01.."/>
  <legende>
   <kap_bez V="31.2.13">
    <bereich V="IV"/>
    <kapitel V="31"/>
    <abschnitt V="2"/>
   </kap_bez>
   <kurztext V="Laserchirurgischer Eingriff der Kategorie W7"/>
   <quittungstext V="Augenärztlicher Eingriff mit Laserchirurgie"/>
  </legende>
  <anmerkung V="Im Anschluss an die Leistung nach der Nr. 31347 kann für ..."/>
  <leistungsinhalt>
   <komplex V="1" S="urn::kbv/keytabs/komplex" SV="1.00">
    <leistung V="Chirurgischer Eingriff der Kategorie W7 entsprechend Anhang 2"/>
   </komplex>
  </leistungsinhalt>
  <bewertung_liste>
   <bewertung V="9370" U="1" U-DOMAIN="urn::kbv/keytabs/punktwerte"/>
  </bewertung_liste>
  <zeitbedarf_liste>
   <zeit V="148" U="2" U-DOMAIN="urn::kbv/keytabs/zeiten">
    <leistung_typ V="1" S="urn::kbv/keytabs/leistungsart" SV="1.00"/>
   </zeit>
  </zeitbedarf_liste>
  <leistungsgruppe V="8" S="urn::kbv/keytabs/lg" SV="1.00"/>
 </allgemein>
 <bedingung>
  <fachgruppe_liste V="true">
   <versorgungsbereich V="1" S="urn::kbv/keytabs/versorgungsbereich" SV="1.00">
    <fachgruppe V="020" S="urn::kbv/keytabs/fachgruppen" SV="1.00"/>
    <fachgruppe V="030" S="urn::kbv/keytabs/fachgruppen" SV="1.00"/>
    ...
```

- reale Welt: Mischform
- nicht def. Datentypen (~unendlich)
- Verweise
- Rekursion

2.1. Grundlagen

- Fehler im Doc führen zwingend zum Abbruch (W3C)

  → Validierung mit „XML Schema“

- XSD
  - XML Schema Definition
  - selbst XML Doc
  - definieren die Struktur eines XML Docs
  - Typ, Reihenfolge, Anzahl, Default Werte von Elementen und Attributen
  - Datentypen, Gültigkeiten, Namensräume
  - kann relationale Datenmodelle beschreiben
  - angelegt für zukünftige Erweiterungen
- DTD
  - Document Type Definition
  - veraltet

- ist sowas wie ein Data Dictionary
- nach Validierung sind XML Docs „garantiert“ lesbar
- XSD erlauben eine „Ansicht“ der Struktur → XMLSpy

2.1. Grundlagen

- neben Verwendung als Datenaustausch Format ist XML für das Internet gedacht
- pures HTML, CSS, ... gehen – aber zu unflexibel

→ XSL eXtensible Stylesheet Language

- besteht aus den Elementen
  - XSL-FO - Formatierung (Browser)
  - XSLT - Transformation (XML,HTML,WML,SVG,RTF,TeX,...)
  - XPath - Navigation (Elemente,Knoten und Mengen davon)

2.1. Grundlagen


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <h2>Ein paar CD's</h2>
   <table border="1">
   <tr bgcolor="#9acd32">
     <th align="left">CD Name</th>
     <th align="left">die Gruppen</th>
   </tr>
   <xsl:for-each select="katalog/cdsammlung/cd">
   <tr>
     <td><xsl:value-of select="titel" /></td>
     <td><xsl:value-of select="gruppe"/></td>
   </tr>
   </xsl:for-each>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

*.XSL

*.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdbsp.xsl"?>
<katalog>
 <cdsammlung>
   <cd>
     <gruppe>Haggard</gruppe>
     <titel>Eppur Si Muove</titel>
     <jahr>2004</jahr>
   </cd>
   ...
```

Browser Bsp.

2.2. Parser

- stellen Funktionalität über einem XML Doc zur Verfügung
- diverse Hersteller mit unterschiedlichen Implementationen
- ein Muss: neuste Browser
- zwei Modelle (in Zukunft evtl. noch STX – Streaming Transformations for XML; Event orientiert, keine Baum Repräsentation)

SAX

- Simple API for XML

- nur lesend
- liest ein Doc Eelement- (zeilen)weise (DataStream basiert)
- arbeitet mit Callback Events
- nicht im .NET implementiert

- geeignet für sehr große XML Docs
- einfach und performant

DOM

- Document Object Model

- lesend und schreibend
- liest immer das ganze Doc ein !!!
- Baum Repräsentation im RAM
- integraler Bestandteil von .NET

- nicht geeignet für sehr große Docs (1MB XML werden bis zu 100MB im RAM je nach Impl.) !

3.1. Übersicht

- SV: XML Generator Class - Report Output Generators
  - nur schreibend
  - können „missbraucht“ werden , es gibt einen CMag Artikel
- SV & MS: TPS nach XML via ASP und ADO
  - Konvertierung, siehe ASP
- SV & CP: XML Klassen
  - Wrapper für die CenterPoint Klassen, Punkt 3.2.
- ThinkData: xmlFUSE
  - Wrapper für Microsoft XML SDK's, Punkt 3.3.
- jeder: ASCII

3.2. CenterPoint Wrapper

- CP ist Open Source Project
- SAX und DOM Implementation, in C++
- basiert selbst auf dem XML Parser Toolkit von expat
- unterstützt Schema und XSL(XPath)

  - SV stellt einen Wrapper über die CP Klassen bereit
  - Code in cpxml.inc, .clw (komplexer Code)
  - DLL's: C60cpxml.dll, C60xmlty.dll
  - implementiert SAX und DOM (kein XPath !?)

  - m.e. unzureichend dokumentiert (ClarionXmlSupport.pdf) !
  - es gibt nur ein DOM basiertes Template Beispiel
  - rudimentäre Artikel im CMag zu simplen DOM Anwedungen
  - absolut nichts zu SAX

3.2. CenterPoint Wrapper

in C61 existieren folgende Einstiegpunkte in XML:

Einfachheit

- Klassen (3.2.1.)
  - über Interfaces werden die CP Klassen direkt angesprochen,
  - DOM und SAX
  - kein DOM Code in .clw >> nur bei CP
- API (Prozedur basiert, 3.2.2.)
  - nutzen die Klassen über Referenzen, Code in .inc und .clw
  - DOM
  - es gibt zwei SAX Prozeduren, wie Callbacks ?, allein anwendbar ?
- Code Templates (3.2.3.)
  - nutzen die API
  - nur DOM

Flexibilität

3.2.1. CP Klassen

- SAX
  - SAXParserClass mit komplettem Callback Interface

- DOM
  - XMLExchange — Hauptklasse für XML Im- und Export
  - XMLNavigator
  - XMLSchema — (im PDF noch nicht dokumentiert)
  - XMLNameMap — mapt Felder und Attribute Namen
  - XMLStructWrapper — für ~File, ~Group, ~View, ~TreeView, ~Queue
  - diverse Helper Classes — ?

3.2.2. CP API

DOM Prozeduren

- Typ 1 — elementar
  - XMLΣToDOM — Erzeugen eines DOM Obj. aus einer XML Quelle
  - DOMToXMLΣ — Speichern des DOM Obj. in ein XML Ziel
  - ΦToDOM — Export des Inhalts einer Clarion Struktur in ein DOM Obj.
  - DOMToΩ — Import des DOM Obj. in die Clarion Struktur
  - div. Helper
- Typ 2 — zusammengesetzt aus Typ 1 und Helpern
  - ToXMLFile — Export einer Clarion Struktur nach XML Datei
    Impl. ~ ΦToDOM + DOMToXMLFile
  - FromXMLFile — Import einer XML Datei in Clarion Struktur
    Impl. ~ XMLFileToDOM + DOMToΩ
  - ViewXML — Ansicht eines XML Doc

Σ – {Datei,String}
Φ – {File,Queue,View}
Ω – {File,Queue}

3.2.3. Code Templates

- ViewXML

  öffnet ein Fenster mit 4 mehr oder weniger uunterschiedlichen Ansichten

- ImportFromXML / ExportToXML

  sehr vereinfacht, nur die vorgegebene triviale XML Struktur möglich

- FromXML / ToXML

  etwas mehr Kontrolle, Schema Unterstützung, nur die triviale Struktur

- Templates sind triviale Wrapper der gleichnamigen API Prozeduren

3.2.4. Probleme

- alles „höhersprachliche“ ist zu eingeschränkt bzw. zu trivial
  - keine gemischt Tag und Attribute basierten XML Docs
  - keine Dereferenzierung im XML Doc
  - keine referentiellen Datenmodelle möglich

- sehr wichtig - für weitere Einschränkungen unbedingt lesen:
  - „Built-in Caveats and Limitations“ in ClarionXMLSupport.pdf

3.3. xmlFUSE

- ThinkData
- COM Implementation von MS-XML 4.0 (basierend auf den COM Klassen von Plugware)
- mit im Packet: SOAP, HTTP, Webservices

- SAX und DOM Modell
- implementiert Schema, XSL, XPath u.v.m.
- kommt komplett mit Source aller Wrapper und dem Plugware COM Layer
- kommt mit Beispiel zu jeder einzelnen Implementation
- es gibt keine Templates

3.3. xmlFUSE

- hier auch: Trennung von Klassen und Prozedur API (diese Proz. sind Wrapper der Klassen Methoden
- umfangreiche API - kommt mit einem Satz von Prozeduren, die ausführlich im PDF und im Code beschrieben sind
- Massen an Interfaces werden bereitgestellt – kaum dokumentiert, Verweis auf das MS-XML SDK

- Probleme
  - MS-SOAP SDK wird eingestellt (geht in .NET auf) ... ?
  - sehr kompl(ex)(iziert)

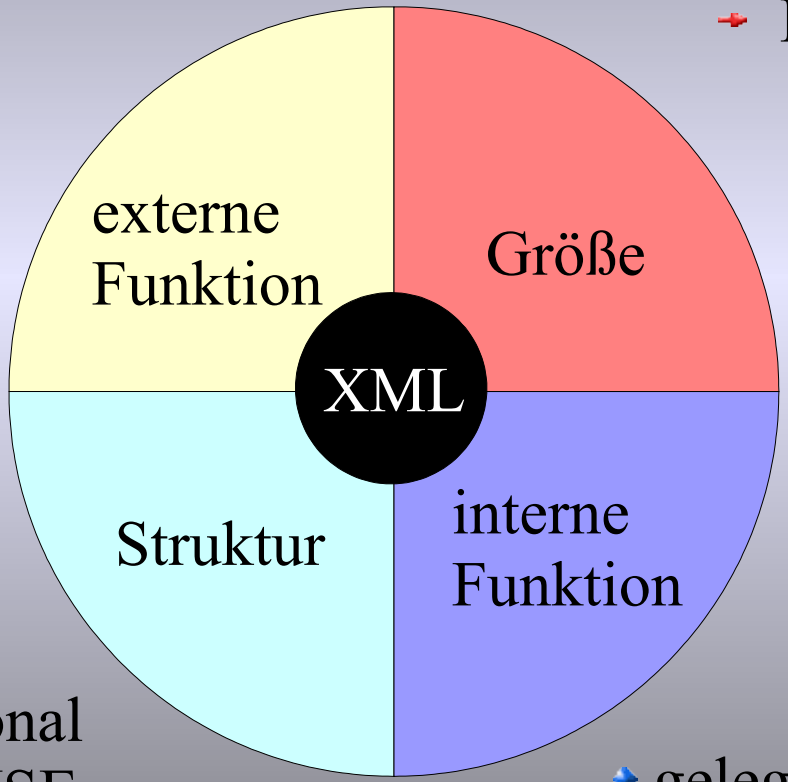
3.4. ASCII

- „Parser“ selber schreiben
- ASCII einlesen entspricht etwa dem SAX Modell
- schnell, schlank, „kennt/kann jede innere Struktur“

- zum Einlesen ist es eine Alternative
- zum Schreiben nur für triviale Strukturen geeignet

4. Auswahl Kriterien

- lesend und/oder schreibend
  - SAX, DOM
- klein oder groß
  - DOM, SAX, ASCII, ...



- trivial oder relational
  - ASCII oder FUSE
  - OOP, API, Templates
- gelegentl. oder häufiger Zugriff
  - DOM, XPath, XQuery, ...

5. Literatur

- XML
  - www.w3schools.de
  - www.w3c.org
  - www.xml.org
  - www.perfectxml.com
  - http://msdn.microsoft.com/xml
- CenterPoint
  - www.cpointc.com
- xmlFUSE
  - www.thinkdata.com
- XMLSpy
  - www.altova.com

Das war:

XML und Clarion

ciaotschüß, tg.