EKON 7

Consic

XML-Grundlagen

Einführung in die Extensible Markup Language:
Syntax, Aufbau, Einsatz, Beispiele

Stefan Heymann
Consic Software Engineering · www.consic.de · heymann@consic.de

Umgebungsbedingungen

- Handy aus?
- Bei Fragen: fragen!
- www.consic.de > Downloads > Talks

Inhalts-Übersicht

- Begriffsbestimmung, Geschichte
- Syntax (Elemente, Tags, usw.)
- XML-Design
- Misc (Beispiele, Parsing, etc.)

Einleitung

Der XML-Standard

- Extensible Markup Language
- Ziele
  - SGML-kompatibel
  - Einfacher als SGML
  - Fehler von HTML vermeiden
  - Einfach zu erzeugen und verarbeiten
- WorldWideWeb Consortium (W3C)
- Version 1.0: 10. Februar 1998
- Version 1.0SE: 6. Oktober 2000
- Version 1.1: ???

Was ist XML?

- Keine Programmier-Sprache
- Keine Layout-Sprache
- Nicht der HTML-Nachfolger
- Ja was denn nu?
- Ein Dateiformat zur Speicherung strukturierter Informationen.

Beispiel: Bestellung

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

<ORDER>

  <HEAD>
    <name>Stefan Heymann</name>
    <address>Tübingen</address>
  </HEAD>

  <DATA>
    <item prodno="10238" count="1">
      <name>Nikon F80 Gehäuse</name>
      <price curr="EUR" amt="390" />
    </item>

    <item prodno="20032" count="1">
      <name>Nikon 1.8/50 mm AF/D Objektiv</name>
      <price curr="EUR" amt="178" />
    </item>
  </DATA>

</ORDER>
```

XML-Syntax

Die XML-Deklaration

- XML-Version
- Encoding (Zeichensatz)
- Steht ganz am Anfang jeder Datei

Beispiel:

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

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

Das Element

Start-Tag	Inhalt	End-Tag
`<name>`	`Heinz Rühmann`	`</name>`
`<preis>`	`10.50`	`</preis>`

Das Element: Start-Tag

- Steht in <spitzen> Klammern
- Name: Beginnt mit Buchstabe, ‘_’ oder ‘:’
- Weitere Zeichen: A..Z, a..z, 0..9, ‘.’, ‘-’, ‘_’, ‘:’
- Wie alles in XML: Name ist **CaSe-SeNsItIvE!**

Beispiele:

```
<name>
<long-description>
<Short-Description>
```

Das Element: Attribute

- Attribute eines Elements werden im Start-Tag notiert
- Allgemein: name="wert"
- Anführungszeichen (einfach oder doppelt) sind Pflicht!
- Wertangabe ist Pflicht

Beispiele:

```
<preis waehrung="DEM" >
<custom-attribute name='template' >
```

Das Element: Inhalt

Inhalt [Element Content] kann sein:

- Elemente (Schachtelung)
- Text
- CDATA-Sektionen
- Entity-Referenzen
- Processing Instructions
- Kommentare

Das Element: End-Tag

- Abschluss eines Elements
- Schrägstrich, gefolgt vom Element-Namen

Beispiele:

```
</preis>
</custom-attribute>
```

“Leere” Elemente

- [Empty Elements]
- = Elemente ohne Content. Die Information ist:
  - in den Attributen
  - bzw. das *Vorhandensein* des Elements
- Spezielle Notation erlaubt

Beispiele:

```
<preis waehrung=“DEM” wert=“12.50” />
<typ name=“Schuhe” />
<br/>
```

Das Element: Vordefinierte Attribute

Es gibt in XML zwei vordefinierte Attribute:

- xml:lang gibt die Sprache eines Elements an
  `xml:lang="de-DE"`
  `xml:lang="en-US"`

- xml:space gibt an, ob Whitespaces erhalten werden sollen
  `xml:space="preserve"`
  `xml:space="default"`.

Text-Inhalte

- Normaler Text gemäß dem definierten Zeichensatz
- Nicht erlaubte Zeichen (< > & " ') müssen durch die *vordefinierten Entities* notiert werden:

<	<
>	>
&	&
'	'
"	"

Text-Inhalte

- Bei nicht darstellbaren Zeichen wird der Unicode-Wert durch “Character References” notiert

- € € [EURO SIGN]

- & & [AMPERSAND]

CDATA-Sektionen

- Andere Notation für Text
- Wenn viele Markup-Zeichen vorkommen (z.B. Quellcodes)

Syntax:

```
<![CDATA[IF A < B THEN Min := A ]]>
```

Müsste sonst “gequotet” werden

Schachtelung von Elementen

- Elemente können geschachtelt werden
- Content = ein oder mehrere Elemente
- Dadurch entsteht die Baum-Struktur einer XML-Instanz
- Ein XML-Dokument *ist* genau ein Element

Beispiel: Bestellung

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

<ORDER>
  <HEAD>
    <name>Stefan Heymann</name>
    <address>Tübingen</address>
  </HEAD>

  <DATA>
    <item prodno="10238" count="1">
      <name>Nikon F80 Gehäuse</name>
      <price curr="EUR" amt="390" />
    </item>

    <item prodno="20032" count="1">
      <name>Nikon 1.8/50 mm AF/D Objektiv</name>
      <price curr="EUR" amt="178" />
    </item>
  </DATA>
</ORDER>
```

Schachtelung von Elementen

```
<HEAD>
  <name>Elvis Presley</name>
  <address>Graceland</address>
</HEAD>
```

- Schachtelungstiefe beliebig
- End-Tags *müssen* vorhanden sein
- Elemente *müssen* korrekt geschachtelt werden

<a><B>...</B>

falsch!

<a><B>...</B>

OK

Processing Instructions (PI)

- Spezielle Notation für Anweisungen an den Parser
- Inhalt beliebig. Gängig sind “Pseudo-Attribute”

Syntax:

```
<?target inhalt ?>
```

Beispiel:

```
<?xml-stylesheet type="text/xml" href="#style1” ?>
```

Kommentare

- Wie bei HTML `<!-- Kommentar -->`
- Kommentar darf keine aufeinander folgenden Bindestriche enthalten
- Keine Schachtelung möglich

Beispiel:

```
<!-- Ich bin ein Kommentar. -->
```

XML-Design

Strukturen

- Sequenzen und Verschachtelungen in einer Datei
- Dadurch praktisch alle Informationen abbildbar
- Ausnahme: Binäre Daten
  (→ MIME base64)

Verarbeitung

- XML ist eine “Meta-Sprache”, d. h. es muss erst auf Basis von XML eine (Applikations-) Sprache entwickelt werden.
- Ein **Parser** zerlegt die XML-Daten in ihre Teile und reicht sie weiter an eine **Applikation,** die sie dann interpretiert und verarbeitet.
- Dabei kann die **Gültigkeit** [Validity] der XML-Daten geprüft werden.

Gültigkeit

- Es gibt drei Arten von XML-Instanzen:
  - Ungültige (falsche) [invalid]
  - Wohlgeformte [well-formed]
  - Gültige [valid]

Design: Struktur

- Analog der Erstellung eines DB-Design:
  - Identifikation der Strukturen
  - Abbildung auf XML-Strukturen:
    - Sequenzen (= Listen/Tabellen)
    - Verschachtelungen (= Hierarchien)
    - Festlegung von Element- und Attribut-Namen

Design: Inhalte

- Text-Inhalt oder Attribut-Wert?
  - Text-Inhalt: Normalisierung durch Applikation beeinflussbar

```
<adresse>Hessenring 9
64546 Mörfelden/Walldorf</adresse>
```

  - Attribut-Wert: Normalisierung gemäß Spezifikation. Üblicherweise “kurze” Werte

```
<name first=“Elvis  A.”  last=“ Presley” />
```

Misc

Anwendungen von XML

- Datenspeicherung (Dateiformat)
- Datenaustausch
  - B2B-Transaktionen (EDI, E-Business)
    - Bestellungen, Liefer-Abrufe
    - Produkt-Kataloge, Produkt-Kategorisierung
  - Datenbank-Export/Import
- Publishing
  - Generierung von HTML, WML, PDF, usw. aus XML + Stylesheet (CSS/XSL)

Plattform-Unabhängigkeit

- XML-Dateien sind Text-Dateien
- Alle XML-Dateien sind Unicode
  (€ €)
- Offene Standards
  - Unicode
  - XML selbst ist “offen”

Beispiele

Intershop enfinity Produkt-Katalog

```
<product sku="0 601 120 503">
  <template>gw/product.isml</template>
  <image>/images/gw/products/GW_BG/GW_BG_BM/gbm16-2re_thumb.jpg</image>
  <product-type name="GEW_WKZ_BOHR_4"/>
  <list-price-used>0</list-price-used>
  <category-links>
    <category-link name="GEW_WKZ_BOHR"/>
  </category-links>
  <name xml:lang="de-DE">GBM 16-2 RE</name>
  <short-description xml:lang="de-DE">Steuer-Electronic - fÃ¼r exaktes Anbohren</short-descript
  <custom-attributes>
    <custom-attribute name="PDF" xml:lang="de-DE">/pdf/GBM-16-2re.pdf</custom-attribute>
    <custom-attribute name="Produktbild_gross">/images/gbm16-2re_gross.jpg</custom-attribute>
    <custom-attribute name="Abgabeleistung" xml:lang="de-DE">570 W</custom-attribute>
    <custom-attribute name="Maschinengewicht" xml:lang="de-DE">3,7 kg</custom-attribute>
    <custom-attribute name="Bohrdurchmesser in Stahl" xml:lang="de-DE">8 mm</custom-attribute>
    <custom-attribute name="Bohrdurchmesser in Holz" xml:lang="de-DE">40 mm</custom-attribute>
    <custom-attribute name="Bohrdurchmesser in Alu" xml:lang="de-DE">13 mm</custom-attribute>
  </custom-attributes>
</product>
```

Beispiele

Die XML-Spezifikation selbst

```
<p>XML documents are made up of storage units called
<termref def="dt-entity">entities</termref>, which contain
either parsed or unparsed data.Parsed data is made up of
<termref def="dt-character">characters</termref>,some of
which form <termref def="dt-chardata">character data</termref>,
and some of which form <termref def="dt-markup">markup</termref>.
Markup encodes a description of the document's storage layout
and logical structure.
XML provides a mechanism to impose constraints on the storage
layout and logical structure.</p>
```

Beispiele

Wireless Markup Language (WML)

Beispiele: WML

```
<?xml version="1.0"?>
<wml>  [...]
  <card id="startup">
    <onevent type="onenterbackward">      <prev/>    </onevent>
    <onevent type="ontimer">
      <go
href="http://wap.otto.de/servlet/WelcomeServlet?clientId=1"/>
    </onevent>
    <timer value="25"/>
    <p align="center">
      <br/>
      <img src="http://wap.otto.de/ottoproj/wap/ottologo.wbmp"
           alt="Willkommen bei Otto!"/>
      <br/> <br/>
      <a href="http://wap.otto.de/servlet/WelcomeServlet?clientId=1"
            title="Hauptmen&#252;">--> Hauptmen&#252;</a>
      <br/> <br/>
      <anchor title="Zur&#252;ck">--> Zur&#252;ck
        <prev/>
      </anchor>
    </p>
  </card>
</wml>
```

Beispiele: Datenbank-Export

```
<?xml version="1.0" encoding="UTF-8" ?>
<DBEXPORT version="1.0">
<sql-select>select * from p$catalogcategories</sql-select>

<fielddefs>
  <fielddef name="CCAT_ID" type="number" mandatory="true" />
  <fielddef name="PARENT_CCAT_ID" type="number" />
  <fielddef name="NAME" type="string" len="150“ mandatory="true" />
  <fielddef name="SHORTDESC" type="string" len="2000" />
</fielddefs>

<content>
  <record index="0">
    <field name="CCAT_ID">1</field>
    <field name="PARENT_CCAT_ID"></field>
    <field name="NAME">Marken</field>
    <field name="SHORTDESC">Bosch</field>
  </record>
</content>
</DBEXPORT>
```

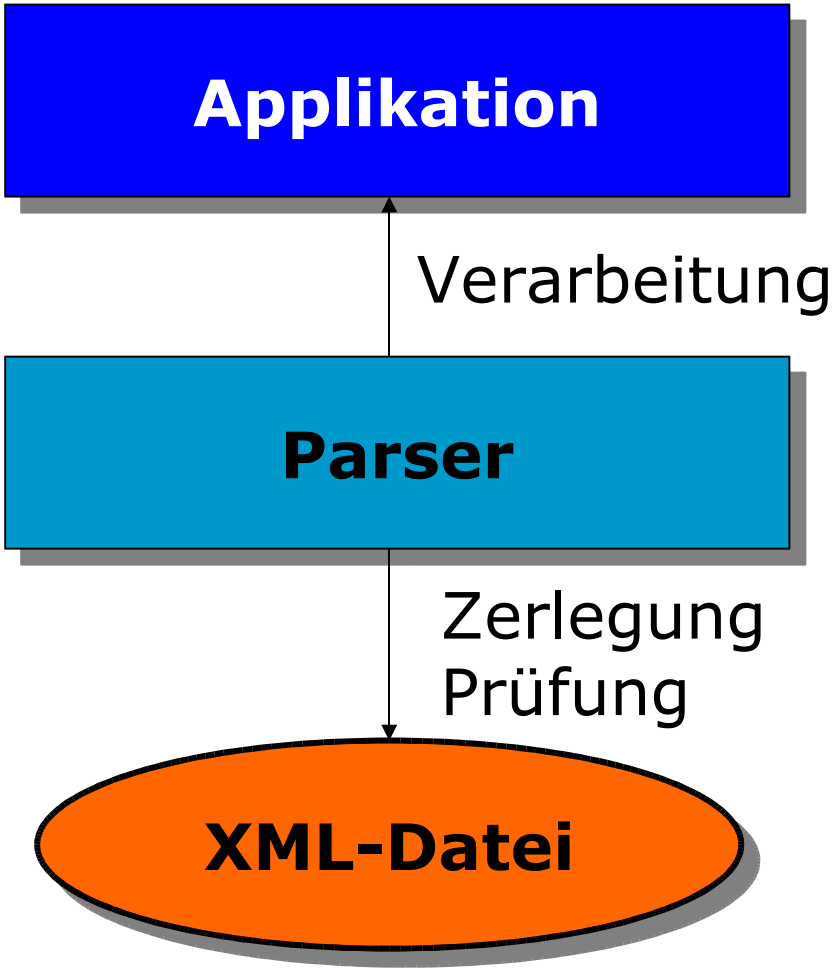
“Fertige” XML-Standards

- SOAP (Simple Object Access Protocol)
- WML (Wireless Markup Language)
- XHTML
- SVG (Scalable Vector Graphics)
- MathML (Mathematical Markup Language)
- Intershop enfinity
- BMEcat (Produkt-Katalog-Datenaustausch)
- Biztalk, cXML (Commercial XML)
- XSL, XSLT, XML-Schema
- u. v. a. m.

XML Parsing

- Validating/non-validating Parser
- Event-gesteuert: **SAX**
- Aufbau eines Objekt-Baums im Speicher: **DOM**
- Progressive: destructor.de-Parser
- Produkte: MSXML, OpenXML, XmlPartner, destructor.de, AElfredPascal, SAX for Pascal, …

XML-Verarbeitung



Wo ist der Kick?

XML-Vorteile

- Struktur
- Überprüfbarkeit
- Unicode
- In 100 Jahren noch lesbar
- Gut maschinell verarbeitbar – trotzdem noch lesbar
- Struktur einfach erweiterbar
- Alle Informationen in einer Datei
- Entities/Textbausteine/Modularität
- Kein Syntax-Wildwuchs
- Gute Komprimierbarkeit

Probleme

- „Laberhaftigkeit“
- Komplexer als es sein müsste
- Unüberschaubar viele XML-bezogene Techniken
- Aufwändig zu parsen
- DTD-Konzept zu schwach

Fragen?

Consic

</talk>

Stefan Heymann
Consic Software Engineering · www.consic.de · heymann@consic.de