

Skriptsprachen: Python

xml.dom.minidom



Jan Krüger, Alexander Sczyrba

Technische Fakultät
Universität Bielefeld

30. September 2016

XML – eXtensible Markup Language

- Daten sind sturkturiert (Texte, Bilder, Messergebnisse)
- Maschinelle Verarbeitung erfordert Kenntnis der Struktur
- Automatisierte Verarbeitung erfordert einheitliches Datenformat

Gesucht: Formalismus, um beliebige Strukturen zu beschreiben

Lösung: XML

- XML kann textuelle Daten strukturieren
- XML ist eine Metasprache zur Definition von Auszeichnungssprachen
 - Welche Elemente/Attribute sind vorhanden?
 - Wie dürfen diese verschachtelt sein?

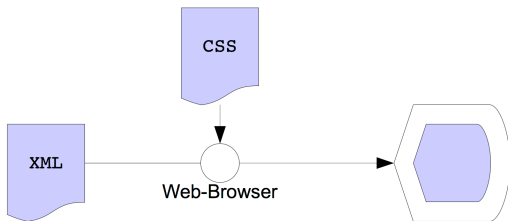
XML-Sprachbeispiele

- XHTML 1.1
- SVG
- RSS 2.0
- RDF
- MathML

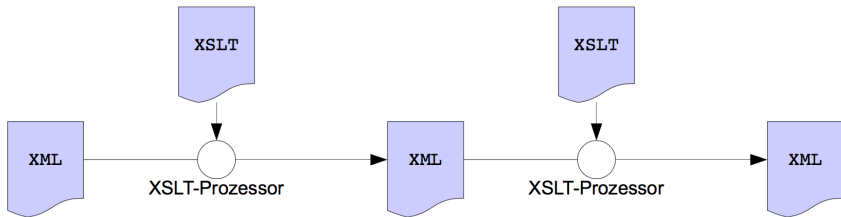
```
<?xml version="1.0" encoding="UTF-8"?>
<Wurzelement>
  <Element>...</Element>
  <Element>
    <Standalone-Element Attribut="" />
  </Element>
  <Standalone-Element />
</Wurzelement>
```

XML - Und dann?

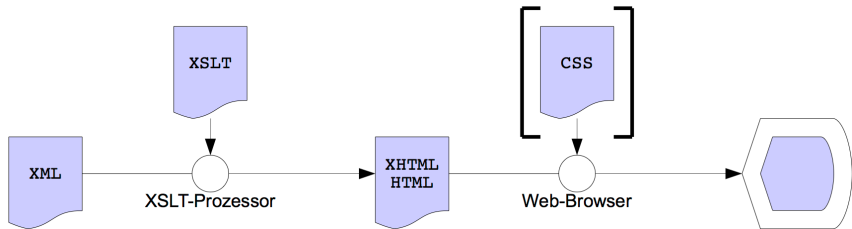
- XML dient rein zur Datenrepräsentation
- Darstellung sekundäres Problem
- XML muss weiterverarbeitet werden



XML transformieren



XML transformieren & darstellen



Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
    </item>
      <item>XML instances can be <emph>well formed</emph>
or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

XML-Deklaration

Optional Kodierung:
encoding="UTF-8"
encoding="ISO-8859-1"

Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
    </item>
      <item>XML instances can be <emph>well formed</emph>
or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

Root

XML-Dokumente besitzen
genau ein Root-Element
(Wurzel).

Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML & Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
    </item>
      <item>XML instances can be <emph>well formed</emph>
or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

Element

Ab der Wurzel können beliebig viele Kindelemente folgen. Sie werden auch Elementknoten genannt.

Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
    </item>
      <item>XML instances can be <emph>well formed</emph>
or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

Öffnendes und
schließendes Tag

Ein Element besitzt genau
ein öffnendes und ein
schließendes Tag:

<elem> – öffnend
</elem> – schließend

Alternativ kann ein Tag
auch leer sein: <elem/>

Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML & Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
    </item>
      <item>XML instances can be <emph>well formed</emph>
or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

Textknoten

Elemente können als Inhalt
Text enthalten, sodass sie
als Kindknoten einen
Textknoten besitzen.

Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
    </item>
      <item>XML instances can be <emph>well formed</emph>
or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

Entities (Entitäten)

Spezielle Zeichen werden als Entitäten gesetzt – Textmakro.

Sie dienen zur Kodierung von Sonderzeichen, reservierten Zeichen und oft verwendeten Zeichenketten.

Neben bereits vorgegebenen, können neue definiert werden. Sieger hierzu: DTD.

Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
    </item>
      <item>XML instances can be <emph>well formed</emph>
or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

Attribute

Elemente können Attribute besitzen, um Zusatzinformationen zu hinterlegen. Das key/value-Paar kommt nur im öffnenden Tag vor und muss eindeutig benannt sein.

Ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)
      </item>
      <item>XML instances can be <emph>well formed</emph>
      or even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

Nested Elements

In Textknoten können eingebettete Elemente vorkommen, wobei hier auf korrekte Schachtelung geachtet werden muss:

```
<a><b>... </b></a>
vs.
<a><b>... </a></b>
```

Besonderheiten

- Groß-/Kleinschreibung ist relevant:

<elem>GEHT NICHT</Elem>

- Reservierte Sonderzeichen:

< – < ; > – > ; & – & ;
' – ' ; " – " ;

- Mehrzeilige Kommentare: <!--KOMMENTAR-->

- Darf nicht vorkommen: --
- Keine Verschachtelung möglich

Wohlgeformtheit und Validierbarkeit

Wenn alle Vorgaben eingehalten wurden, ist das XML-Dokument wohlgeformt und kann korrekt verarbeitet werden, wenn ein DTD/Schema angegeben wurde.

Überprüfung durch Validierer:

Kommandozeile xmlwf, xmllint

Onlinetools W3C, Validome

Übung – Mediendatenbank

Erstellt ein XML-Dokument, das eine Mediendatenbank für Musikstücke repräsentiert. Überprüfe die Wohlgeformtheit mittels der vorgestellten Kommandozeilentools und Onlinetools.

Fügt beispielhaft drei Einträge ein(inkl. leerer Elemente in einem Eintrag)

Namespaces

- Verwendung unterschiedlicher XML-Sprachen in einem Dokument
- Vergleichbar mit Modulen (Python, Java) oder einer Telefonvorwahl
→ Unterscheidung zwischen gleichnamigen Elementen
- Werden durch URLs (Uniform Resource Identifier) dargestellt
 - Müssen keine gültige URL (Uniform Resource Locator) darstellen
 - Wenn gültige URL, dann sollte dort zu finden sein:
 - DTD (Dokumenttypdefinition)
 - XML-Schema

Namespaces

```
<html xmlns="http://www.w3.org/1999/xhtml">
  ... XHTML-Elemente
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    ... MathML-Elemente
  </math>
  ... XHTML-Elemente
</html>
```

Attribut `xmlns:pf="http://..."` binden den Namensraum `pf` an die URI `http://...`

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:m="http://www.w3.org/1998/Math/MathML">
  ... XHTML-Elemente
  <m:math>
    <span>Hier steht XHTML. Variable x: <m:i>x</m:i></span>
  </m:math>
  ... XHTML-Elemente
</html>
```

XPath & DOM-Repräsentation

Es existieren unterschiedliche Techniken, um durch ein XML-Dokument zu navigieren (und dieses zu verändern)

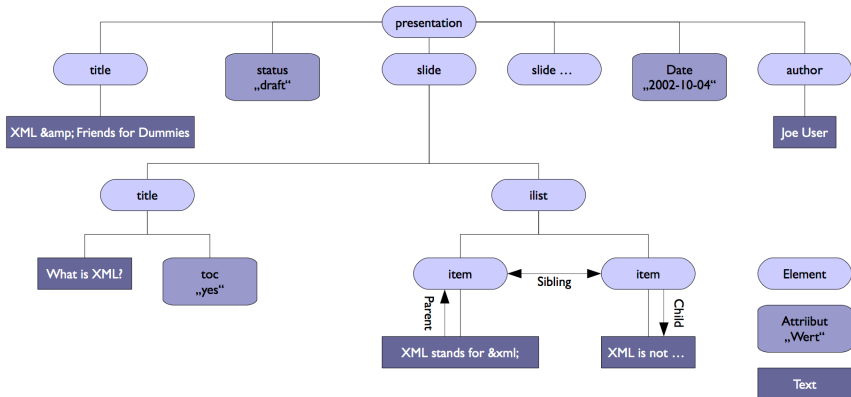
XPath Navigieren

DOM-Repräsentation Navigieren und Modifizieren

Beide Verfahren sehen das Dokument als Baum an, mit:

- Knoten: Element, Attribut, Text, Kommentar, Namespace, PI(Processing Instruction)
- Navigation entlang einer Achse
- Optionale Knotentests (Bedingungen)

Baumrepräsentation



XPath

- Beginnt mit der Suche ab dem aktuellen Kontextknoten
- Finden eines Zielknotens während der Lokalisierung:
Achse::Knotentest [Praedikat1] [Praedikat2]
 - Achse** Pfad, getrennt mit / oder Achsangabe
 - Knotentest** * alle Kindknoten, NAME eines Knotens oder Funktionsaufruf wie z.B. Text() oder comment()
 - Prädikat** Index, Relationszeichen, Zeichenkettenfunktion, Operatoren, Knotenmengenfunktionen

XPath – Beispiele

`/*` – Wurzelemente

`//Interpret` – Alle Elemente auf der Interpret-Achse

`//Titel/Produzent[2]` – Der zweite Produzent
eines Titels

`//Titel/Produzent[2][@gema='ja']` – Der zweite
Produzent eines Titels, der GEMA-Mitglied ist

`attribute::*` – Alle Attribute des Kontextknotens

`child::text()` – Alle Textknoten des Kontextknotens

`.` – Der aktuelle Kontextknoten

DOM-Repräsentation

- Document Object Model (DOM): Spezifikation für Zugriff auf XML-Dokumente
- Implementationen in vielen Sprachen verfügbar: JavaScript, JDOM für Java, LibXML jeweils für C, Python, Perl...
- Durch Spezifikation: Einheitliche Funktionen, Funktionsnamen, Funktionsweise
- Lesender und schreibender Zugriff

Arbeiten mit DOM

- XML-Dokument wird eingelesen und geparkt – Parser
- XML-Dokument liegt als Baumstruktur vor, mit folgenden Knotentypen:

Document	Node	Element
Attr	Text	Comment
CDATA	Namespace	DTD
Processing	Instructions	

- Variablen repräsentieren Knoten eines bestimmten Typs → Funktionen auf Variablen sind kontextabhängig
- Knoten können anhand einer eindeutigen ID, Tarnnamen (mit/ohne Namespace), Namen, Verwandtschaftsverhältnisses und mittels XPath-Ausdrücken angesteuert werden

Das Modul `xml.dom.minidom` ist eine besonders leichtgewichtige DOM-Implementierung für Python.

- Implementiert DOM 1.0 vollständig
- DOM 2-Featureliste nur teilweise, aber für uns ausreichend
- Voll OO-Ansatz

Parsen eines Dokuments

Modulimport `import xml.dom.minidom as dom`

Datei parsen `dom.parse(<FILENAME>)`

String parsen `dom.parseString(<STRING>)`

Leerer DOM-Baum `dom.Document()`

DOM-Baum serialisieren

als `String` `toxml([ENCODING])`, Default-Encoding: UTF-8

in `Datei` `writexml(WRTR[, indent[, addindent [, nl ]]])` , mit:

`WRTR` Oftmals `file` -Instanz

`indent` Einschub des Startknotens

`addindent` Einschub aller weiter Kinderknoten

`nl` Newline-Charakter

```
> f = open("stuff.xml", "w")
> domTree.writexml(f, "", "\t", "\n")
> f.close()
```

Elemente erstellen

- Elementknoten

```
element = doc.createElement(qname)  
element = doc.createElementNS(nsURI, qname)
```

- Textknoten

```
text = doc.createTextNode(data)
```

- Kommentarknoten

```
comment = doc.createComment(data)
```

- Attributknoten

```
attribute = doc.createAttribute(nsURI, qname)
```

Attributknoten

Attributknoten erstellt, und dann? Was ist mit dem Wert?

```
> aNode.value = <StrAsValue>  
> node.setAttributeNode(aNode)
```

Oder direkt am Knoten erstellen:

```
node.setAttribute(<NAME>, <StrAsValue>)
```

Und wieder weg damit...

```
node.removeAttribute(name)  
node.removeAttributeNode(aNode)
```

Finden von Knoten(listen

- über die Element id

```
node = doc.getElementById(id)
```

Achtung! Das Attribute muß vom Typ id sein!

- über den Element Namen

```
nodes = doc.getElementsByTagName(tagname)
```

```
nodes = doc.getElementsByTagNameNS(nsURI, tagname)
```

Anhängen, austauschen, löschen von Knoten

```
child = node.appendChild(children_node)
```

```
node.replaceChild(new_node, old_node)
```

```
child = node.removeChild(child_node)
```

Übung – minidom ganz schön groß

Erweitert das in der vorherigen Übungsaufgabe erstellte XML-Dokument mit Hilfe von minidom in Python um einen weiteren Eintrag. Lest hierfür euer Dokument ein, findet den zu erweiternden

Knoten, erstellt neue Knoten, hängt diese an den DOM-Baum an entsprechender Stelle wieder ein und speichert das XML-Dokument unter einem neuen Dateinamen ab.