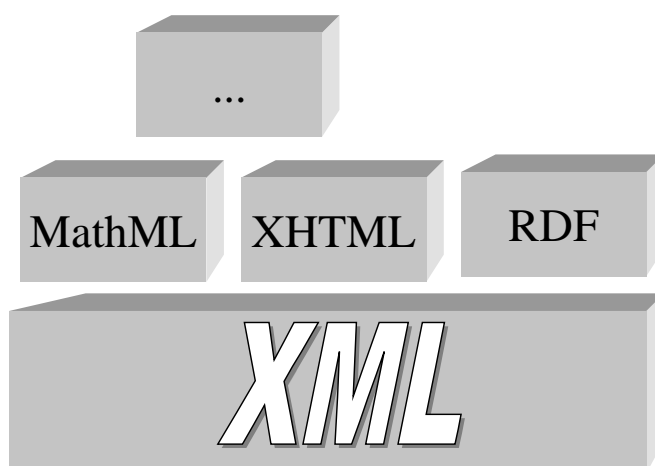




Einführung in XML

Die eXtensible Markup Language [XML] ist eine Metasprache für die Definition von Markup Sprachen. Sie unterscheidet sich durch ihre Fähigkeit, Markup Befehle und Attribute definieren zu können, wesentlich von bekannten Markup Sprachen wie z.B. HTML, das nur die Nutzung einer bestimmten (und unveränderlichen) Menge von Markup Befehlen erlaubt.

Das seit Januar 2000 als Recommendation des W3C vorliegende XHTML [XHTML] ist eine XML konforme Reformulierung von HTML. Inzwischen existieren eine Fülle weiterer XML Anwendungen:



XML ist als Teilmenge von SGML nicht ganz so mächtig wie SGML, dafür aber sehr viel einfacher anzuwenden. Im Folgenden werden die grundlegenden Konzepte von XML vorgestellt.

XML Dokumente

Das folgende Beispiel zeigt ein typisches XML Dokument:

```
<?xml version="1.0" ?>
<document class="H.3.3">
  <author>John Smith</author>
  <title>XML Retrieval</title>
  <chapter>
    <heading>Introduction</heading>
    This text explains all about XML and IR.
```

```

</chapter>
<chapter>
  <heading>Extensible Style
    Language</heading>
  <section>
    <heading>Examples</heading>
  </section>
  <section>
    <heading>Syntax</heading>
  </section>
</chapter>
</document>

```

Im Gegensatz zu HTML wird in XML zwischen Groß- und Kleinschreibung unterschieden. Die erste Zeile gibt an, um welche XML Version es sich bei den vorliegenden Dokument handelt.

```
<?xml version="1.0" ?>
```

Bislang gibt es nur die Version "1.0". Nach dieser sogenannten Processing Instruction beginnt der eigentliche Inhalt des XML Dokumentes, der aus einer Reihe von Markup Befehlen und den Daten besteht, die in einer baumartigen Struktur angeordnet werden. Die eigentlichen Daten stehen zwischen dem Markup, z.B. `<heading>Syntax</heading>`. XML ist in Bezug auf die Syntax strenger als HTML, da jedes geöffnete Tag wieder geschlossen werden muss und keine überlappenden Tags (z.B. `<a>`) erlaubt sind.

Im Folgenden werden die wichtigsten Sprachkonstrukte im Einzelnen vorgestellt.

Processing Instructions

Die Processing Instructions (PI) geben darüber Aufschluss, wie das folgende XML Dokument verarbeitet werden soll. Die erste Zeile eines XML Dokumentes muss die Processing Instruction `<?xml version="1.0" ?>` enthalten. Der innerhalb eines Dokumentes verwendete Zeichensatz kann über das Schlüsselwort `encoding` in der PI definiert werden, z.B. `<?xml version="1.0" encoding="UTF-8"?>`. Wird kein Zeichensatz spezifiziert, erwartet der XML Parser UTF-8. Es können beliebige weitere PIs angegeben werden, die Metadaten über das XML Dokument enthalten. Sie dürfen nur nicht mit dem reservierten Wort `xml` starten.

Tags

Die Struktur der Dokumente ist durch die Tags gegeben und entspricht einem Baum. Insbesondere muss es ein Wurzelement geben. Der eigentliche Inhalt der Daten steht innerhalb der Tags. Ein leeres Tag `<a>` kann durch den Ausdruck `<a/>` abgekürzt

werden. Jeder Tag muss geschlossen werden und es sind keine überlappenden Tags (z.B. <a>) erlaubt.

Attribute

Tags können Attribute besitzen. Diese spezifizieren meistens die Daten, die der Tag enthält, etwas genauer. Ein Attribut darf höchstens einmal auftreten, d.h. eine Konstruktion wie <mitarbeiter name="hans" name="emil"> ist verboten. Ob es sinnvoller ist, Daten in Attribute oder in eigenständigen Tags zu speichern, hängt von der jeweiligen Anwendung ab und ist oft nicht einfach zu entscheiden.

Kommentare

Kommentare <!-- Hier kommt ein Kommentar --> können wie normale Tags an beliebiger Stelle im XML Dokument eingefügt werden.

Textknoten

Analog zu HTML kann man auch bei XML Markup und Text vermischen:

```
<chapter>
  <heading>Introduction</heading>
  This text explains all about XML and IR.
</chapter>
```

Diesen Text bezeichnet man als Textknoten.

CDATA

Unter Character Data (CDATA) lassen sich beliebige Zeichenketten abspeichern, die beim Parsen des XML Dokumentes nicht weiter überprüft werden. Auf diese Weise lässt sich zum Beispiel ein nicht korrekter XML Textabschnitt in einem XML Dokument abspeichern:

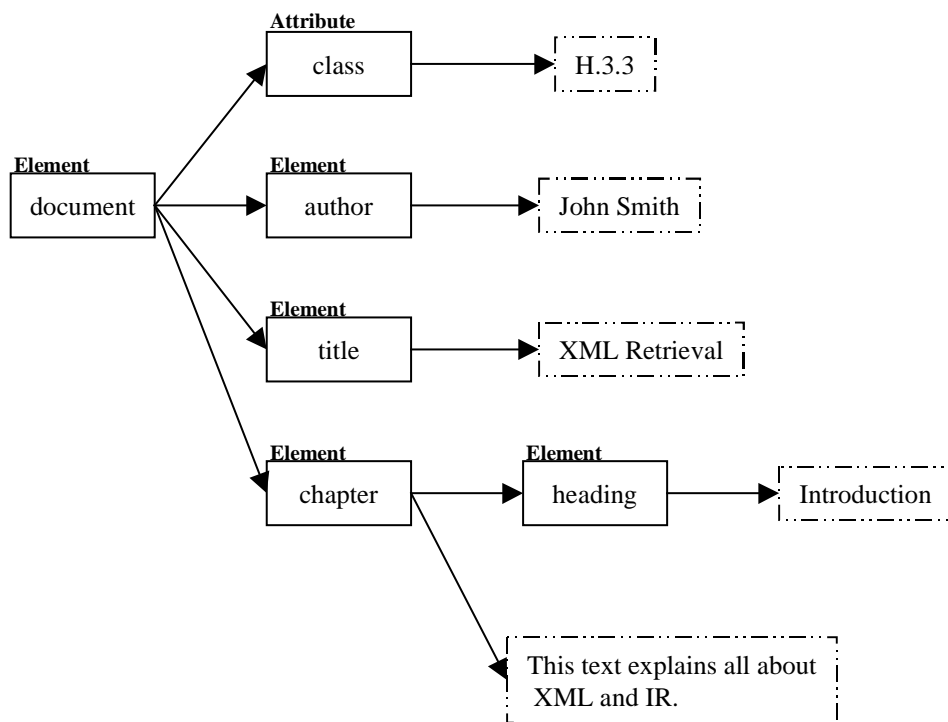
```
<oldHTML >
<![CDATA[
  <html>
  <p>Introduction
  <br>This text explains <BR>
  all about XML and IR.
  <HTML>
</oldHTML >
```

Baumstruktur

Jedes XML Dokument lässt sich als Baum darstellen, dessen Knoten den im letzten Kapitel vorgestellten Typen entsprechen.

```
<document class="H.3.3">
  <author>John Smith</author>
  <title>XML Retrieval</title>
  <chapter>
    <heading>Introduction</heading>
    This text explains all about XML and IR.
  </chapter>
</document>
```

Die Baumstruktur dieses XML Dokumentes sieht wie folgt aus:



Validität

Well-Formed vs Valid

Bislang ist nur die grobe syntaktische Struktur von XML Dokumenten thematisiert worden. Dazu gehört etwa die Unterscheidung zwischen Groß- und Kleinschreibung, die notwendige Baumstruktur der Tags usw. Entspricht ein XML Dokument den im letzten Abschnitt erläuterten Bedingungen, so bezeichnet man dieses Dokument als wohlgeformt (well-formed).

Dieser Korrektheitsbegriff reicht in vielen Fällen jedoch nicht aus, da er keine Informationen über die erlaubten Tags und ihre Beziehung zueinander repräsentieren kann. Aus diesem Grund gibt es in XML die Möglichkeit, die komplette syntaktische Struktur von XML Dokumenten in einer Document Type Definition (DTD) festzulegen. Für jedes XML Dokument ist es dann entscheidbar, ob es einer bestimmten DTD entspricht oder nicht. Zum Beispiel ist in der DTD von XHTML genau festgelegt, welche Tags für die Erstellung eines XHTML Dokumentes benutzt werden dürfen. Die Gültigkeit (Valid) eines XML Dokumentes bezüglich einer DTD schließt die Wohlgeformtheit des Dokumentes ein. Die Umkehrung gilt im Allgemeinen nicht.

Document Type Definition

Im Folgenden wird die Grundfunktionalität einer DTD anhand eines Beispiels demonstriert. Angenommen, man möchte XML Dokumente für die Speicherung von Adressen verwenden.

```
<?xml version="1.0" ?>
<adresse>
  <name>Karl Mustermann</name>
  <strasse> Musterweg 15</strasse>
  <ort>Berlin</ort>
</adresse>
```

Eine DTD für diesen Dokumenttyp wird wie folgt definiert:

```
<!DOCTYPE adressen [
<!ELEMENT adresse (name, strasse, ort)>
  <!ELEMENT name #PCDATA>
  <!ELEMENT strasse #PCDATA>
  <!ELEMENT ort #PCDATA>
]>
```

Hier wird festgelegt, dass das Wurzelement `adresse` vom Dokumenttyp `adressen` die Kindelemente `name`, `strasse` und `ort` besitzt. Die DTD lässt sich direkt in das XML Dokument mit einbinden.

```
<?xml version="1.0" ?>
<!DOCTYPE adressen [
```

```

<!ELEMENT adresse (name, strasse, ort)>
  <!ELEMENT name #PCDATA>
  <!ELEMENT strasse #PCDATA>
  <!ELEMENT ort #PCDATA>
]>

```

```

<adresse>
  <name>Karl Mustermann</name>
  <strasse> Musterweg 15</strasse>
  <ort>Berlin</ort>
</adresse>

```

Alternativ kann die DTD in einer externen Datei abgelegt und in das Dokument über einen Verweis eingebunden werden.

```

<?xml version="1.0" ?>
<!DOCTYPE adressen SYSTEM="adressen.dtd">

<adresse>
  <name>Karl Mustermann</name>
  <strasse> Musterweg 15</strasse>
  <ort>Berlin</ort>
</adresse>

```

Dieses Beispiel sollte nur einen kleinen Einblick in die Erstellung von DTDs geben. Da bezüglich RDF die Wohlgeformtheit von Dokumenten das entscheidende Kriterium ist, wird im Folgenden nicht weiter auf DTDs eingegangen.

Namespaces

Ein Problem mit Namensräumen ergibt sich immer dann, wenn in einem XML Dokument Tags aus verschiedenen Communities benutzt werden sollen. Zum Beispiel könnte man eine Dublin Core Metadatenbeschreibung sehr einfach als XML Dokument repräsentieren:

```

<?xml version="1.0" ?>
<metadata>
  <creator>Karl Mustermann</creator>
  <title>XML </title>
</metadata>

```

Um auszudrücken, wo diese Tags definiert wurden, kann man einen Namespace angeben. Ein Namespace beschreibt eine Menge von Elementen und Attributen, die einer URI zugeordnet sind.

```

<?xml version="1.0" ?>
<metadata xmlns:dc="http://purl.org/dc/elements/">

```

```
<dc:creator>Karl Mustermann</dc:creator>
<dc:title>XML </dc:title>
</metadata>
```

Nun lassen sich auch andere Tags für die Beschreibung von Metadaten nutzen. Zum Beispiel könnte man den Namen des Creators in Vor- und Nachname auflösen, indem man vCard Elemente in die Beschreibung aufnimmt.

```
<?xml version="1.0" ?>
<metadata xmlns:dc="http://purl.org/dc/elements/
  xmlns:vCard="http://imc.org/vCard/3.0#">
  <dc:creator>
    <vCard:Given>Karl </vCard:Given>
    <vCard:Family>Mustermann </vCard:Family>
  </dc:creator>
  <dc:title>XML </dc:title>
</metadata>
```

Durch den Einsatz von Namespaces lassen sich Tags benutzen, die in verschiedenen Communities definiert wurden. Offensichtlich führt ein solcher Mix von Tags im Allgemeinen nicht zu einem bezüglich einer DTD gültigen XML Dokument.

Referenzen

[XML] Extensible Markup Language (XML)
<http://www.w3.org/XML/>

[XHTML] XHTML 1.0: The Extensible HyperText Markup Language
A Reformulation of HTML 4 in XML 1.0
<http://www.w3.org/TR/xhtml1/>