XML Vorlesung ETHZ, Sommersemester 2006

XML Grundlagen I

Erik Wilde

11.4.2006

http://dret.net/lectures/xml-ss06/

Übersicht

- Aufbau von XML
  - XML Dokumenten-Syntax
  - wichtigste Spracheigenschaften
  - XML Dokumententypen (DTDs)
- Gültigkeit von XML-Dokumenten
  - syntaktische Gültigkeit
  - Validierung anhand bestehender DTDs

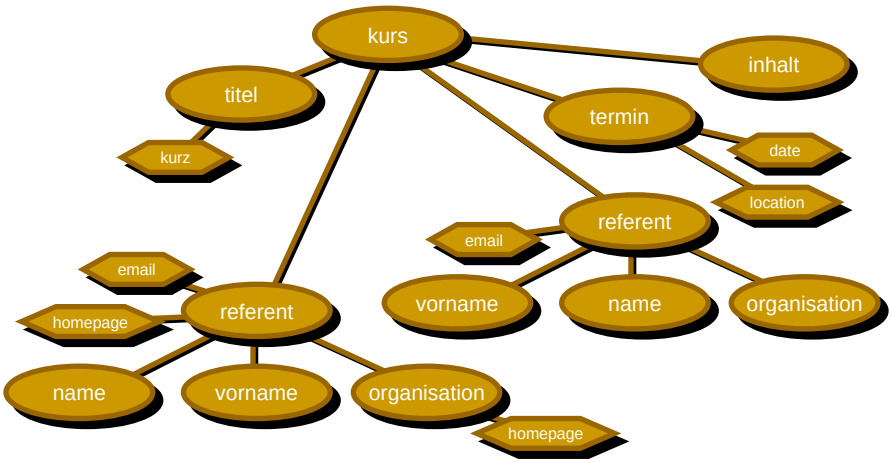
Markup

- Markup ist die physische Form eines Dokuments
  - Markup ist immer menschenlesbar
  - *XML Text = Character Data + Markup*
- Markup wird durch spezielle Zeichen markiert
  - Tags sind in < und > eingeschlossen (`<titel>`)
  - Entities sind in & und ; eingeschlossen (`&#252;`)
- XML benutzt immer die gleichen Zeichen
  - wichtiger Unterschied zu SGML (*SGML Declaration*)
  - ermöglicht einfachere Implementierungen
- Markup-Analyse ist eine Standardaufgabe
  - eingebaut in Software (z.B. MSXML in IE)

Beispiel (XML)

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

<kurs>
<titel kurz="XML">XML - Grundlagen und Umfeld</titel>

<referent email="xml@dret.net"
          homepage="http://dret.net/">
  <vorname>Erik</vorname>
  <name>Wilde</name>
  <organisation homepage="http://www.tik.ee.ethz.ch/">ETH
  Zürich</organisation>
</referent>

<referent> ... </referent>

<termin date="20000512" location="technopark"/>

<inhalt> ... </inhalt> </kurs>
```

Baumstruktur eines XML Dokuments



Elemente

- Elemente sind der grundlegende Mechanismus
  - Strukturierung von hierarchischen Daten
  - "beliebige" Namensgebung für Elemente
  - Definition gemäss inhaltlichen Strukturen
  - Kernpunkt des DTD-Designs
- Elementtypen haben zwei wichtige Aspekte
  - ein *content model* für erlaubten Inhalt
  - Attribute (optionales Vorkommen oder notwendig)
- DTD deklariert Typ, den Dokument verwendet

```
DTD:       <!ELEMENT titel      (#PCDATA) >
Dokument: <titel>XML - Grundlagen und Umfeld</titel>
```

Element-Markup

- ein Element ist eine logische Einheit
- im XML Dokument eingeschlossen durch Tags
  - Start-Tag für den Beginn des Elements: `<title>`
  - End-Tag für das Ende des Elements: `</title>`
- Elemente können Attribute haben
  - immer im Start-Tag angegeben
- Elemente können leer sein
  - `<title></title>`
  - alternative Kurzschreibweise: `<title/>`
  - für eine Applikation nicht unterscheidbare Fälle

Regeln für Elementtypdeklarationen

- der Inhalt von Elementen kann sein
  - nur Elemente *(element content model)*
  - Text gemischt mit Elementen *(mixed content model)*
  - kein Inhalt erlaubt (`EMPTY` Keyword)
- das *content model* eines Elements
  - optionales Vorkommen mit `?`, wiederholbar mit `*`
  - notwendig und wiederholbar mit `+`
  - Sequenz mit `,`
  - Alternativen (Exklusiv-oder) mit `|`
- nicht erlaubt sind folgende SGML Konstrukte
  - vertauschbare Vorkommen mit `&` und Exceptions

Regeln für Elemente

- jedes XML-Dokument hat genau eine Wurzel
  - *document element*
- jedes Element hat ein Eltern-Element
  - das *document element* hat kein *parent element*
- direkt untergeordnete Elemente sind Kinder
  - falls keine Kinder: Blätter (*leave element*)
- untergeordnete Elemente sind Nachkommen
  - *descendant* (Kinder und Kindeskinder usw.)
- übergeordnete Elemente sind Vorfahren
  - *ancestor* (Eltern und Grosseltern usw.)

Attribute

- Attribute sind Informationen zu Elementen
  - Attribute geben Zusatzinformationen
  - Entscheidung Attribut/Element nicht immer klar
- optional (`#IMPLIED`) oder notwendig (`#REQUIRED`)
- Attribute können verschiedene Typen haben
  - ein Konzept, das für Elemente nicht existiert
  - deutliche Einschränkungen (siehe HTML DTD)

```
DTD:      <!ATTLIST titel kurz CDATA #REQUIRED >
Dokument: <titel kurz="XML">XML - Grundlagen...
```

Regeln für Attributlistendeklarationen

- erlaubt sind
  - mehrere Attributtypen in einer Attributliste
  - mehrere Attributlisten für ein Element
  - bei Namenskonflikten zählt das erste Vorkommen
  - gleiche Attributnamen für verschiedene Elemente
- nicht erlaubt sind
  - eine Attributliste für mehrere Elemente (erlaubt in SGML!)
  - einige der SGML-Typen für Attribute
- erlaubte Typen sind
  - String types (beliebiger String als Wert)
  - Enumerated types (Auswahl aus definierter Liste)
  - Tokenized types (XML Namen verschiedener Art)
    - insbesondere `ID/IDREF(S)` als Referenzierungsmechanismus

Attributtypen

- es gibt eine Reihe verschiedener Typen
  - `CDATA` definiert beliebige Strings (keine Elemente!)
  - Enumeration listet mögliche Werte auf
  - `ID` zur Identifikation, Eindeutigkeit wird überprüft
  - `IDREF` definiert eine Referenz auf eine **`ID`**
  - `IDREFS` definierte eine Liste von Referenzen auf **`ID`**s
  - `ENTITY` referenziert ein deklariertes Entity
  - `ENTITIES` gibt eine Liste von Entity-Referenzen an
  - `NMTOKEN` verlangt ein Name Token (XML Name)
  - `NMTOKENS` verlangt eine Liste von Name Tokens
  - `NOTATION` Enumeration listet Notations auf

Regeln für Attribute

- ein Attribut ist immer ein Name/Value-Paar
- Attributnamen müssen also angegeben werden
  - in SGML/HTML dürfen sie u.U. weggelassen werden
- Attributwerte müssen in Quotes gesetzt werden
- Attribute können weggelassen werden
  - vom Parser ersetzt falls auf `#IMPLIED` gesetzt
  - nicht erlaubt falls auf `#REQUIRED` gesetzt
- Attribute werden immer im Start-Tag verwendet
  - konzeptionell Information am Element-Knoten

Wann Elemente, wann Attribute?

- Elemente immer, wenn...
  - weitere Strukturierung notwendig
  - Re-use in unterschiedlichen Kontexten nötig
  - wiederholtes Vorkommen notwendig ist
- Attribute immer, wenn...
  - spezielle Datentypen notwendig (z.B. `ID/IDREF`)
  - Bindung an das Element sehr eng
- keine abschliessenden Regeln möglich
  - Entscheidungen im Einzelfall
  - Strategie definieren und konsequent durchhalten

Document Type Definition (DTD)

- Beschreibung der Datenstrukturen in einem Schema
  - Schema beschreibt eine Klasse von Dokumenten
  - SGML/XML DTD ist nur eine mögliche Variante
  - *XML Schema* als Weiterentwicklung (später mehr dazu...)
- Beschreibung von Datenblöcken
  - Elemente als Strukturmittel
  - Attribute als Daten zu Elementen
- Beschreibung der erlaubten Kombinationen
  - Definition einer Grammatik
  - Verwendung für die Validierung von Daten
  - Verwendung für die Generierung von Daten
- Schema Modellierung als Kern von XML

Beispiel (Verweis auf DTD)

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

<kurs>
<titel kurz="XML">XML - Grundlagen und
  Umfeld</titel>

<referent email="xml@dret.net"
  homepage="http://dret.net/">
  <vorname>Erik</vorname>
  <name>Wilde</name>
```

Beispiel (Teil einer DTD)

```
<!ELEMENT kurs         (titel, referent+, termin+, inhalt) >

<!ELEMENT titel        (#PCDATA) >
<!ATTLIST titel
          kurz         CDATA #REQUIRED >

<!ELEMENT referent     (vorname, name, organisation?) >
<!ATTLIST referent
          email        CDATA #IMPLIED
          homepage     CDATA #IMPLIED >

<!ELEMENT vorname      (#PCDATA) >
<!ELEMENT name         (#PCDATA) >
<!ELEMENT organisation (#PCDATA) >
<!ATTLIST organisation
          homepage     CDATA #IMPLIED >
```


Prolog eines XML Dokuments

- eine optionale XML Deklaration
  - `<?xml version="1.0"?>`
  - optional Zeichencodierung (Default ist UTF-8)
  - optional standalone-Deklaration
- Kommentare oder Processing Instructions
  - `<!-- kommentar -->`
  - `<?php instructions ?>`
- optional Document Type Declaration
  - definiert den Typ des Dokuments
  - notwendig für die Validierung eines Dokuments

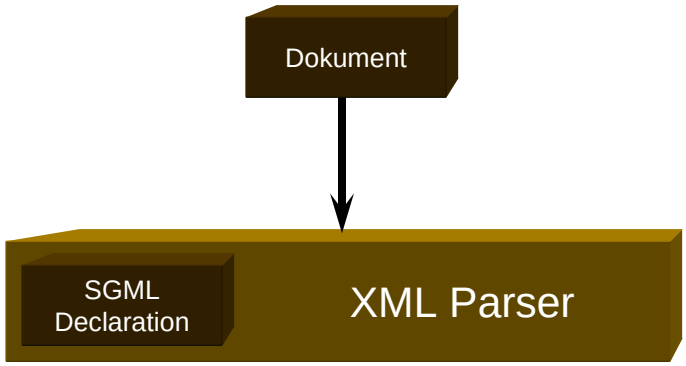
XML Parser

- Programm zur Interpretation von XML
  - offizieller Name in der Spec: XML Processor
- viele bestehende Software-Pakete
  - eines der wichtigen Argumente für XML
  - Markup-Analyse erledigt bestehende Software
  - eigene Programme verwenden Parser
- XML Spec beschreibt das Verhalten
  - was muss oder darf ein Parser akzeptieren
  - was muss oder darf ein Parser zurückweisen
  - keine Beschreibung einer konkreten Schnittstelle

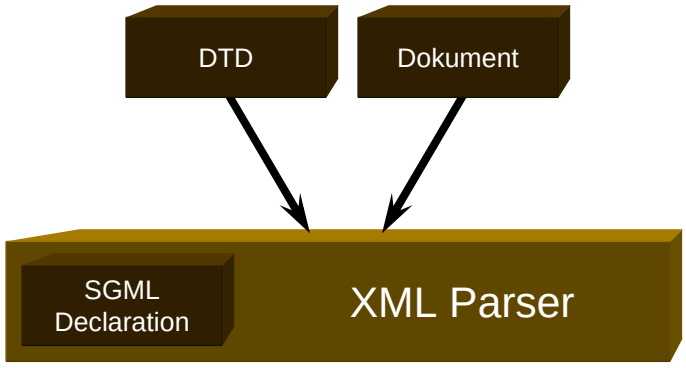
Well-formed und *valid* XML

- XML unterscheidet zwischen zwei "Levels"
  - *well-formed* gehorchen dem XML-Standard
  - *valid* sind well-formed und gehorchen einer DTD
- *well-formed* Dokumente sind korrektes XML
  - falls keine DTD vorhanden (nicht immer nötig!)
  - falls DTD nicht verfügbar
  - falls keine Weiterverarbeitung notwendig
- *valid* Dokumente sind korrekt gemäss DTD
  - Validierung anhand einer DTD
  - sinnvolle Kontrolle zur Weiterverarbeitung
  - im B2B Umfeld wohl ausnahmslos valid XML

XML Parser (Well-formedness)



XML Parser (Validierung)



Zusammenfassung

- XML als Syntax für strukturierte Daten
  - Regeln werden in DTDs beschrieben
  - Unterscheidung *valid/well-formed* XML Dokumente
  - hierarchische Organisation der Daten
  - Strukturierung in Elemente/Attribute