XML und RDF: Rückblick und Ausblick

Markus Stocker

25. Juni 2018

Rückblick

XML

- Extensible Markup Language
- Eine erweiterbare, beschreibende *meta* Auszeichnungssprache
- Hierarchisch (semi-) strukturierte Daten (Baumstruktur)

XML: Wichtige Sprachkonstrukte

- Tag
  - Beginnt mit < und endet mit >
  - Drei Arten: *opening*, *closing*, *empty element*
- Element
  - Beginnt mit *opening tag* und endet mit entsprechendem *closing tag*
  - Inhalt entweder Text oder andere Elemente
  - Beachte Gross-/Kleinschreibung, Sonderzeichen, Verschachtelung
- Attribut
  - In *opening* oder *empty-element tags*
  - Innerhalb Klammern < > als `name="value"` Paare
- Dokument
  - Enthält ein Wurzelelement
  - Ist wohlgeformt wenn syntaktisch korrekt

XML: Bemerkungen

- XML is mensch- und maschinenlesbar
- Tags sind allerdings nur für Menschen Bedeutungsvoll
- Verwendet für Datenaustausch, zwischen Anwendungen, im Internet
- Interoperabilität beruht allerdings auf Einigung (Schema)
- XML kann in Programmiersprachen gelesen/geschrieben werden
- XML hat auch Nachteile, insb. Ballast wegen tagging

XPath

- Ermöglicht die Verarbeitung von XML Daten
- Deklarativer Zugriff auf Teile eines XML Dokuments
- Selektion von Elementen und Inhalten
- Operationen auf Inhalten

XPath: Wichtige Sprachkonstrukte

- Lokalisierungspfad
  - Adressierung von Knotenmengen
  - Setzt sich aus mehreren Einzelschritten zusammen
  - Diese werden mittels / getrennt
  - Absolut/relativ und ausführlich/verkürzt
- Schritt
  - A::KT[P]*
  - Achse: Navigationsrichtung aus dem Kontextknoten
  - Knotentest: Gewünschte Knotenmenge
  - Prädikat: Filterbedingungen
- Prädikat
  - Filtrierung/Einschänkung der XPath Ergebnissmenge
  - Definition genauer Zielmenge
  - Komplexere Problemstellungen

DTD

- Sprache zur Spezifikation von XML Dokumente
- Beschreibung gültiger Elemente, Attribute, Struktur
- Diese werden in einer DTD deklariert
- Ermöglicht die Validierung von XML Dokumente
- Sprich diese auf Gültigkeit zu prüfen

DTD: Deklarationen

- Elemente
  - `<!ELEMENT name inhalt>`
  - Inhalt: Kindelemente, `PCDATA`, `EMPTY`
  - Inhaltsmodelle: Sequenz, Alternative, Wiederholungen
- Attribute
  - `<!ATTLIST element attribut typ wert>`
  - Typen: `CDATA`, enumerierte Liste, `ID`, `ENTITY`
  - Werte: Standardwert, `#REQUIRED`, `#IMPLIED`, `#FIXED value`
- Entitäten
  - `<!ENTITY name "wert">`

XML Schema

- Weitere Sprache zur Spezifikation von XML Dokumente
- Adressiert einige Schwächen der DTD
- Insb. Datentypen, Namensräume, XML Syntax
- Stellt vordefinierte Datentypen zur Verfügung
- Definition benutzerdefinierte Datentypen
- Ermöglicht detailliertere Spezifikation von XML Dokumente
- Strengere Restriktionen die auf Gültigkeit geprüft werden können

XML Schema: Datentypen

- Datentypen sind wichtig
- `2018-02-10` und `0.5` sind Daten von verschiedenem Typ
- In XML Schema spezifiziert man diese nicht als `PCDATA`
- Sondern als `xs:dateTime` und `xs:decimal`
- Zudem spezifiziert `xs:dateTime` das Format `YYYY-MM-DD`
- `2018-02-10` ist somit 10. Februar nicht 2. Oktober

XML Schema: Deklarationen

- Einfache Elemente (keine Kindelemente oder Attribute)
  - `<xs:element name="..." type="..."/>`
  - Typen: `xs:string`, `xs:boolean`, `xs:int`, ...
  - Vorgegebene (`default`) oder festgelegte (`fixed`) Werte
  - Häufigkeit des Elements (`minOccurs`, `maxOccurs`)
- Attribute
  - `<xs:attribute name="..." type="..."/>`
  - Können vorgegebene oder festgelegte Werte haben
- Komplexe Elemente
  - `<xs:element><xs:complexType>...`
  - Zwei Varianten, eine mit benutzerdefiniertem Datentyp
  - Ordnung mittels `xs:all`, `xs:choice`, `xs:sequence`

Wozu Schema

- Beispiel NASA, ESA und JAXA die Daten austauschen
- Einigung über verwendete Tags
- Wie auch die Struktur (Elemente, Attribute)
- Ziel ist es Unterschiede zu vermeiden
- Und so den Datenaustausch zu erleichtern
- Die Interoperabilität der Systeme erhöhen

Wohlgeformtheit und Gültigkeit

- Ein XML Dokument mit korrekter Syntax ist wohlgeformt
- Validiert ein wohlgeformtes Dokument einem Schema ist es gültig
- Ein gültiges Dokument ist immer auch wohlgeformt (*well-formed*)
- Ein wohlgeformtes Dokument ist nicht zwingend gültig (*valid*)

RDF

- Resource Description Framework
- Beschreibung von Ressourcen, digitale, physische, imaginäre Entitäten
- Grundlegender Baustein des *Semantic Web*
- Formale Spezifikation der Bedeutung von Daten
- Graphbasiertes Datenformat, Knoten und gerichtete Kanten
- Mittels URI benannt, Literale oder *blank nodes*

RDF: Literale und *blank nodes*

- Literale repräsentieren Datenwerte
- Können typisiert oder untypisiert sein
- Typisierte Literale haben einen Datentyp
- Untypisierte Literale können ein *language tag* haben
- *Blank nodes* sind unbenannte Ressourcen
- Haben strukturelle Funktion für mehrwertige Relationen

RDF: Tripel

- Die Elementareinheit in RDF
- Eine Struktur die aus drei Elementen besteht
- Das Subjekt, das Prädikat und das Objekt
- Subjekte und Objekte entsprechen Knoten
- Prädikate entsprechen gerichteten Kanten
- Prädikate sind immer benannt (URI)
- Subjekte und Objekte können unbenannt sein (*blank node*)
- Subjekte und Objekte können benannt sein (URI)
- Objekte können Literale sein

RDF: Syntax

- Tripelmengen können entsprechend einer Syntax serialisiert werden
- Besprochene Syntaxen: N-Triples, Turtle, RDF/XML
- Jede Syntax hat Vor- und Nachteile
- Man kann zwischen Syntaxen convertieren (automatisch)

SPARQL

- SPARQL Protocol And RDF Query Language
- Abfragesprache für RDF
- Ermöglicht deklarativer Zugriff auf RDF Daten
- *Tripel pattern* (Tripelmuster) fundamentale Struktur
- Besteht aus drei Elementen: Subjekt, Prädikat und Objekt
- Wobei diese auch variabel sein können (im Unterschied zum Tripel)

SPARQL

- Eine *triple pattern* Menge nennt man *basic graph pattern*
- Resultatformate: `SELECT`, `CONSTRUCT`, `ASK`, `DESCRIBE`
- Modifizierer: `FILTER`, `ORDER BY`, `LIMIT`, `OFFSET`
- SPARQL Update um RDF Daten deklarativ zu ändern
- SPARQL Endpoints für RDF Datenbank als Web Service
- Abfrageoptimierung damit man Antworten möglichst schnell erhält

RDF Schema

- RDFS unterstützt das Organisieren von Ressourcen einer Tripelmenge
- Grundlegend dabei ist die Gruppierung von Ressourcen
- Dafür stellt RDFS das Konstrukt der Klasse zur Verfügung
- RDFS ermöglicht Klassen als solche zu definieren
- Als Instanzen der Klasse aller Klassen (`rdfs:Class`)
- Dabei spielt das Prädikat `rdf:type` eine wichtige Rolle

RDF Schema

- RDFS ermöglich die Definition von Unterklassen (`rdfs:subClassOf`)
- Und somit die Erzeugung von Klassenhierarchien
- Prädikathierarchien sind ebenfalls möglich (`rdfs:subPropertyOf`)
- Klassenzugehörigkeiten (`rdfs:domain`, `rdfs:range`)

Ontologien

- Formalisierung von Wissen über eine Domäne
- Begriffe und Beziehungen eines Gegenstandsbereiches
- Informationsaustausch unter Beihaltung der Bedeutung
- Bestandteile: Klassen, Relationen, Instanzen
- Aufbau: Schema, Inhalt
- Wissensquellen: Mensch, Bücher, Web, Datenbanken

Ausblick