# XML-Workshop für Editionsprojekte

Christian Sonder M. A.

Wiss. Mitarbeiter SSRQ, Universität St. Gallen

Mail: christian.sonder@unisg.ch

# Was wir in der letzten Sitzung gelernt haben

**1. Sitzung**

1) Was ist XML? Eine erweiterbare Auszeichnungssprache.
2) Wofür benutzt man XML? Strukturieren, Klassifizieren, Annotieren und Kommentieren von Daten.
3) Was sind die zentralen Konzepte und Grundbegriffe? Trennung von Wortlaut, Struktur und Aussehen eines Textes mithilfe von Markup.
4) XML-Syntax, Teil 1: Elemente, Attribute.

# Was wir in dieser Sitzung lernen

**2. Sitzung**

1) Wie kodiert man einfache Daten mit XML?
2) XML-Syntax, Teil 2: Escapen, Kommentare, CDATA Sections, XML-Deklarationen.
3) Wie ist ein vollständiges XML-Dokument aufgebaut?
4) XML und Unicode, Character References.

# Wie kodiert man einfache Daten mit XML?

XML dient dazu, Daten aller Art auszuzeichnen. Hierfür stehen vor allem Elemente und Attribute zur Verfügung.

Wann benutzt man Elemente (bzw. verschachtelte Elemente) und wann Attribute?

Ein paar Grundregeln:

- Attribute sind für Metadaten und nicht für die Daten selbst gedacht.
- Der Inhalt der Elemente sind die Daten selbst.
- Die Namen der Elemente klassifizieren die Daten.
- Die Verschachtelung bzw. Anordnung der Elemente nutzt man, um die Struktur der Daten wiederzugeben.

# Beispiel: Kodierung von Adressdaten

Datensatz: Max Mustermann, Musterstrasse 1, CH-1234 Musterstadt

- Variante 1: Alles wird mit Attributen ausgedrückt.
- <datensatz typ="Adresse" name="Max Mustermann" strasse="Musterstrasse 1" plz="CH-1234" ort="Musterstadt"/>
- Einziger Vorteil: Wenig Markup.
- Nachteile:
  - Attributwerte enthalten Daten statt Metadaten => Es können keine Metadaten mehr erfasst werden.
  - Daten sind so schlecht strukturiert, dass Sortierung und Abfrage sehr schwer sind.
  - Ein Attribut enthält mehr als ein Datum.

# Beispiel: Kodierung von Adressdaten

- Variante 2: Verwendung von Elementen mit flacher Hierarchie.

```
<adresse>
    <name>Max Mustermann</name>
    <strasse>Musterstrasse 1</strasse>
    <plz>CH-1234</plz>
    <ort>Musterstadt</ort>
</adresse>
```

- Vorteil:
  - Daten sind jetzt leicht mit Metadaten anzureichern.
  - Elementnamen drücken nun die Art der Daten aus.
- Nachteile:
  - Manche Elemente enthalten mehr als ein Art von Daten.
  - Verschachtelung drückt den Sachverhalt nicht gut aus.

# Beispiel: Kodierung von Adressdaten

- Variante 3: Verwendung von Elementen mit tiefer Hierarchie und Atomisierung der Inhalte, d. h. jedes Element enthält genau eine Information.

```
<adresse>
   <name>
        <vorname>Max</vorname>
        <nachname>Mustermann</nachname>
   </name>
   <strasse>Musterstrasse</strasse>
   <hausnummer>1</hausnummer>
   <plz>1234</plz>
   <ort>Musterstadt</ort>
   <land>Schweiz</land>
</adresse>
```

# Beispiel: Kodierung von Adressdaten

Fazit: Es gibt in XML viele unterschiedliche Möglichkeiten, einen Sachverhalt zu kodieren. Vor dem Arbeiten mit XML sollte daher geklärt werden:

- Welche Arten von Daten habe ich (Tabellen, Briefe, Gedichte, Romane, …)?
- Was sind meine Daten und welche Metadaten möchte ich zu den Daten erfassen (IDs, Normdaten, Links zu anderen Quellen, …)?
- Was zeichne ich aus und was nicht? (Muss wirklich alles kodiert werden? Jede Auszeichnung kostet Zeit und erhöht die Komplexität des XMLs.)
- Warum zeichne ich aus und zu welchem Zweck?
- Wie zeichne ich die einzelnen Arten von Daten und Sachverhalten aus? (Am Besten nach einem weit verbreiteten Standard, z. B. TEI!)

# Steuerzeichen und Escapen (Umschreiben)

- XML hat 5 Steuerzeichen, die für die Syntax von XML unerlässlich sind: <, >, ", ' und &.
- Wenn diese Zeichen wörtlich im Text vorkommen und nicht Teil des Markups sein sollen, dann müssen Sie in Form von «Entity References» «escaped», d. h. umschrieben, werden.
- Eine «Entity Reference» besteht aus: & + Name der Entity + ;
- Die 5 Steuerzeichen werden wie folgt escaped:

| Entity | Entity Reference | Erläuterung |
|---|---|---|
| < | < | (lt = less than) |
| > | > | (gt = greather than) |
| " | " | (quot = quotation mark) |
| ' | ' | (apos = apostrophe) |
| & | & | (amp = ampersand) |

# Wo muss was escaped (umschrieben) werden?

- In Elementen dürfen nicht vorkommen: <, &
- In Attributen dürfen nicht vorkommen: <, & und
  " (wenn das Attribut mit " eingeleitet wird) oder
  ' (wenn das Attribut mit ' eingeleitet wird)
- Beispiele:

| | |
|---|---|
| statt <aussage>a < b </aussage> | <aussage>a < b</aussage> |
| statt <firma>Hektor & Co. KG</firma> | <firma>Hektor & Co. KG</firma> |
| statt <a value="a < b & c"/> | <a value="a < b & c "/> |
| statt <a zitat="Er sagte: "Ach!""/> | <a zitat="Er sagte: "Ach!""/> |
| statt <a zitat='Er sagte: 'Ach!''/> | <a zitat='Er sagte: 'Ach!'"/> |

# Kommentare

- Ein XML-Kommentar besteht aus:

  <!-- + Inhalt des Kommentars + -->

- Beispiele:

  <!-- ToDo: Was steht hier genau? -->
  <!-- Folgende Person kann ich nicht identifizieren, welche ID soll ich hier vergeben? -->
  <!-- In einem XML-Kommentar <foo val="bar"/> muss man übrigens nichts escapen & das ist auch gut so! -->

- Ein XML-Kommentar darf nicht die Zeichenfolge «--» enthalten:

- Negativbeispiele:

  <!-- Die Noten: 1+, 1 und 1--->
  <!-- Das hier -- ist auch verboten. -->

# Einsatz von Kommentaren

- XML-Kommentare kommen ausserhalb von anderem Markup (z. B. Tags) vor.
- Beispiel:

```
<!-- Ich stehe vor dem Wurzelelement -->
<wurzel>
   <!-- Ich komme vor einem Element -->
   <element>Foo <!-- Drinnen! --></element>
</wurzel>
<!-- Und ich komme nach dem Wurzelelement -->
```

- Kommentare sind zwar Teil des XML-Dokuments, zählen aber nicht zum eigentlichen Text und müssen von XML-Prozessoren nicht zwingend weiterverarbeitet werden. Daher sollten sie nur temporäre Inhalte wie z. B. ToDos enthalten, aber keine relevanten Daten. Vor der Publikation einer XML-Datei sollten die Kommentare nach Möglichkeit abgearbeitet werden.

# CDATA Sections

- Neben Kommentaren gibt es noch CDATA Sections (CDATA = Character Data). Sie bestehen aus:

  <![CDATA[ + Inhalt + ]]>

- Eine CDATA Section gehört zum Text des XML-Dokuments und muss immer verarbeitet werden. Wie beim Kommentar muss nichts escaped (umschrieben) werden, weil der Inhalt wörtlich wiedergegeben wird. Deswegen eignet sie sich besonders gut, um XML-Markup zu zitieren, ohne dass dieses ausgewertet wird.

- Beispiel:

  <![CDATA[ Hier kann beliebiges <element n="1">XML-Markup</element> & so stehen. ]]>

# XML-Deklaration

- Woher weiss ein Computer, dass er es mit XML zu tun hat? Eine XML-Deklaration klärt, dass XML vorliegt und um welches XML es sich handelt.
- Die Deklaration besteht aus folgenden Bestandteilen:

  <?xml + Attribut version + Attribut encoding + Attribut standalone + ?>
- Beispiel: <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
- version="1.0" (Gibt die verwendete XML-Version an, also 1.0 oder 1.1)
- encoding="UTF-8" (Optional, gibt den verwendeten Zeichensatz an, in der Regel UTF-8, aber auch UTF-16, ISO-8859-1 u. a. möglich.)
- standalone="yes" oder standalone="no" (Optional, gibt an, ob man zum Interpretieren des Dokuments Inhalte aus anderen Dateien parsen muss oder nicht. Ohne Angabe gilt standalone="no")

# XML-Dokument

- Ein vollständiges XML-Dokument besteht aus maximal drei Bestandteilen:
  1) Prolog (= XML-Deklaration, Kommentare, Weissraum, Doctype-Deklaration*, Processing Instructions**)
  2) Wurzelelement
  3) Sonstiges (= Kommentare, Weissraum oder Processing Instructions**)
- Der Prolog und das Sonstige sind optional.
- Das minimal mögliche XML-Dokument besteht also nur aus einem Wurzelelement, z. B.: <a/>

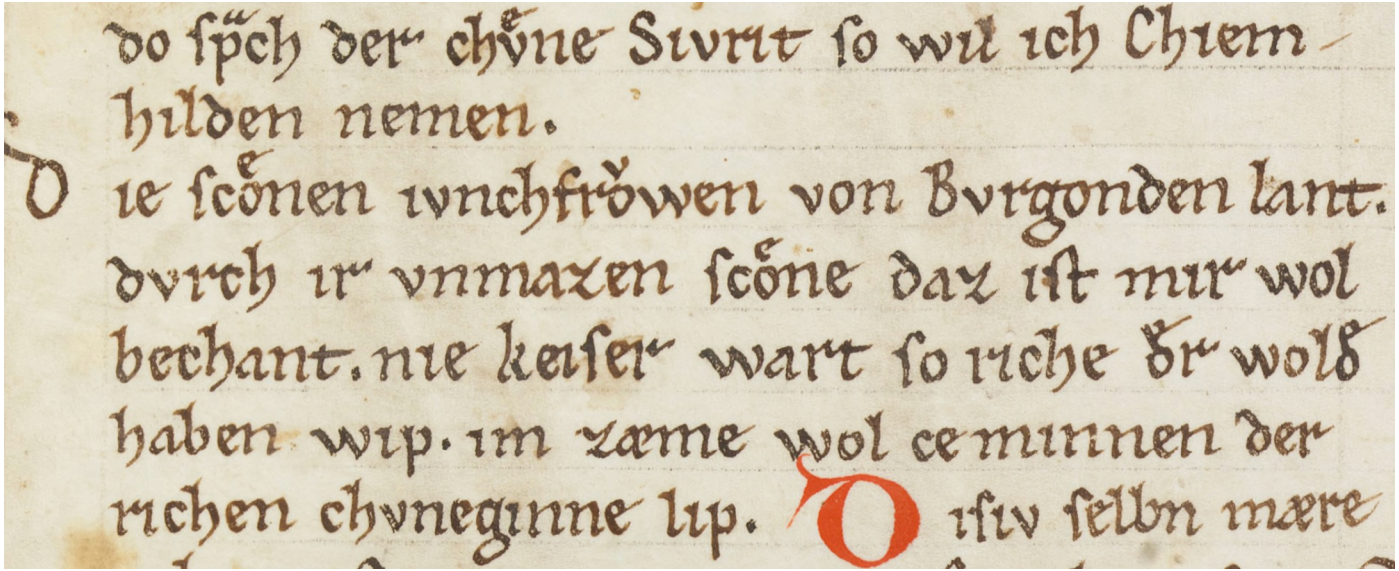
* Doctype-Deklarationen, kurz DTD, werden wir nicht behandeln.
** Processing Instructions lernen wir in der dritten Sitzung kennen

# XML-Dokument mit XML-Deklaration

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Das ist ein kleines Test-Dokument -->
<wurzel>
    <element>Hier kommt mein Inhalt hin.</element>
</wurzel>
<!-- Hier endet das Test-Dokument -->
```

# XML und Zeichensätze

- Digitale Editionen, die XML nutzen, haben einen grossen Bedarf an Sonderzeichen.
- Sonderzeichen lassen sich mithilfe von Zeichensätzen und Fonts darstellen.




Quelle: St. Gallen, Stiftsbibliothek, Cod. 857, p. 293a.

# XML und Zeichensätze

- XML kann mit allen gängigen Zeichensätzen eingesetzt werden.
- Ein Zeichensatz ist eine Zuordnung einer Menge von Zeichen zu Codepunkten:

| Zeichen | Codepunkt (hexadezimal) | Codepunkt (dezimal) |
|---|---|---|
| A | 0041 | 65 |
| B | 0042 | 66 |
| C | 0043 | 67 |
| … | … | … |
| Y | 0059 | 89 |
| Z | 005A | 90 |

# XML und Unicode

- Der wichtigste Zeichensatz der Welt ist Unicode (http://www.unicode.org).
- Unicode (Version 16.0 vom 10.09.2024) enthält z. Z. 154.998 unterschiedliche Zeichen aus nahezu allen Sprachen der Welt. Jedes Zeichen ist einem eigenen Codepunkt zugeordnet.
- Will man Unicode in XML-basierten Editionen nutzen, muss man angeben, welches Unicode-Format man einsetzt (UTF-8, UTF-16, UTF-32 usw.). Das Format legt fest, mit wie vielen Bytes ein Zeichen gespeichert wird. UTF-8 ist am weitesten verbreitet und kann standardmässig eingesetzt werden.
- Beispiel: <?xml version="1.0" encoding="UTF-8"?>
- Doch wie kann man bestimmte Unicode-Zeichen in XML-Dokumente einfügen?

# XML und Unicode

Es gibt zwei Wege Unicode-Zeichen in XML-Dokumente einzufügen.

Variante 1: Man gibt das Zeichen mithilfe seines Codepunkts direkt über die Tastatur ein oder kopiert das Zeichen aus einem anderen Dokument oder einer Tabelle.

Variante 2: Man referenziert den Codepunkt mit einer «Character Reference».

- Eine Character Reference funktioniert ähnlich wie eine Entity Reference:

  Hexadezimal: & + # + x + Codepunkt + ;

  Dezimal: & + # + Codepunkt + ;

  Beispiel: Das so genannte Schaft-s hat in Unicode den Codepunkt 017F (in hexadezimaler Darstellung) bzw. 383 (in dezimaler Darstellung):

- ſ = ſ (hexadezimal) = ſ (dezimal)

# XML und Unicode

- Variante 1: <untertitel>Erſte Periode.</untertitel>
- Variante 2 hexadezimal: <untertitel>Erſte Periode.</untertitel>
- Variante 2 dezimal: <untertitel>Erſte Periode.</untertitel>
- Alle drei Varianten sind völlig bedeutungsgleich. Die letzte ist allerdings ungewöhnlich, da Unicode in der Regel mit Hexadezimalzahlen ausgedrückt wird.
- Variante 1 ist leichter zu lesen, es kann aber sein, dass aus Versehen falsche (weil ähnlich aussehende) Zeichen eingesetzt werden.
- Variante 2 ist schwerer zu lesen, aber der Codepunkt ist klar benannt.
- Welche Variante zum Einsatz kommt, hängt von den benötigten Zeichen und den zur Verfügung stehenden Fonts ab.

# Der komplette Kurs auf GitHub

**https://github.com/ChristianSonderUniSG/xml-kurs**

**Dort finden Sie alle Folien, Übungsaufgaben, Hausaufgaben und weitere Materialien.**