



Gemeinsam weiterbilden



Wiederholung

TEIL 2 ABSCHLUSSPRÜFUNG IT-BERUFE Fachinformatiker AE

- gestreckte Abschlussprüfung Teil 2

PV-AP2 XML Grundlagen / Wiederholung

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Abschlussprüfung AP2 FIAE

- XML
 - Was ist XML?
 - XML Dokumente
 - Elemente, Tags, Zeichendaten und mehr
 - Attribute
 - CDATA - Abschnitte
 - Dokumenttyp-Definition
 - XML - Schema



<https://media.geeksforgeeks.org/wp-content/cdn-uploads/20200424201448/XML.png>

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

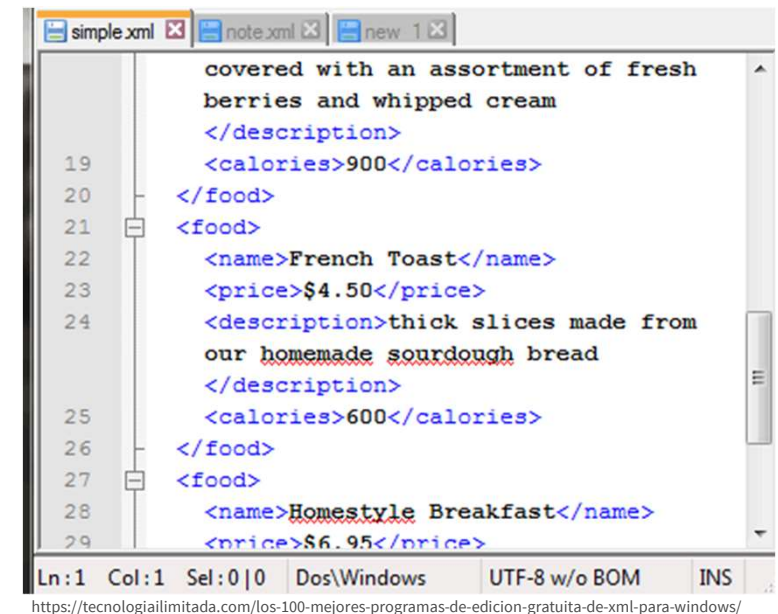
Was ist XML?

- XML steht für **E**xtensible **M**arkup **L**anguage
- Standardisiertes und flexibles Format vom **W3C** für Datenauszeichnung
- generische Syntax, mit derer Hilfe sich Daten mit einfachen, von Menschen lesbaren, Tags auszeichnen lassen
- XML war damit so erfolgreich, dass es für unterschiedlichste Problemfelder (Websites, Datenaustausch, Vektorgrafiken, Serialisierung, uvm.) eingesetzt wurde und wird.
- Viele frei verfügbare Bibliotheken für unterschiedlichste Programmiersprachen stehen zur Verfügung, um XML-Dateien zu lesen und oder zu manipulieren

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Was ist XML?

- XML ist eine Meta – Markup – Language
- X steht für **eX**tensible (dt.: erweiterbar)
- Im Gegensatz zu HTML, das vorgeschriebene Tags hat, ist das Ziel von XML die einfache Erweiterbarkeit von Tags
 - Bei Bedarf können Entwickler neue für ihre Branche passende Tags entwickeln
- Bei Tags ist XML sehr flexibel
- XML ist aber restriktiv gegen seiner Spezifikation: XML definiert eine Grammatik, wo und wie dürfen Tags auftauchen, wo dürfen Attribute auftauchen usw.



The screenshot shows a text editor window with three tabs: 'simple.xml', 'note.xml', and 'new 1'. The 'simple.xml' tab is active and displays XML code for food items. The code is as follows:

```
covered with an assortment of fresh
berries and whipped cream
</description>
<calories>900</calories>
19 </food>
20
21 <food>
22   <name>French Toast</name>
23   <price>$4.50</price>
24   <description>thick slices made from
our homemade sourdough bread
</description>
25   <calories>600</calories>
26 </food>
27 <food>
28   <name>Homestyle Breakfast</name>
29   <price>$6.95</price>
```

The status bar at the bottom indicates 'Ln:1 Col:1 Sel:0|0 Dos\Windows UTF-8 w/o BOM INS'. A URL is visible at the bottom: <https://tecnologiailimitada.com/los-100-mejores-programas-de-edicion-gratuita-de-xml-para-windows/>

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Dokumente (Dateien)

- XML – Dokumente enthalten Text, keine Binärdaten
- Bild zeigt einfachstes, aber „valides“ und wohlgeformtes XML-Dokument
- Das Dokument könnte person.xml heißen
 - XML – Parser ist nicht wählerisch... dieses Dokument könnte auch person.txt, person.data heißen, solange der Inhalt XML ist
- Valides XML kann nicht nur in einer Datei stehen: es könnte auch ein Datensatz oder ein Feld in einer DB sein, könnte auch spontan durch ein anderes Programm als Antwort einer Anfrage erzeugt werden

```
1 | <person>  
2 |   Alan Turing  
3 | </person>
```

Abb1.: Einfaches, aber vollständiges XML-Dokument

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Dokumente (Dateien)

- Bei Webdokumenten sollte der MIME-Typ (Internet Media Type, Content – Type) application/xml sein (hier gibt es aber auch noch speziellere Varianten, je nachdem, was man gerade nutzt)

```
1 | <person>  
2 |   Alan Turing  
3 | </person>
```

Abb1.: Einfaches, aber vollständiges XML-Dokument

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Elemente und Tags

- Beispiel – Bild rechts besteht aus einem **Element** vom Typ person
- Das Element wird eingegrenzt durch den Start Tag <person> und dem Endtag </person>
- Alles zwischen Start – Tag und End – Tag einschließlich White-Space ist der Inhalt des Elements
- Die Tags sind das Markup, der String Alan Turing sind **Zeichendaten**

Element und Starttag von person

```
1 | <person>
2 |   Alan Turing
3 | </person>
```

Abb. 1

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Elemente und Tags

- XML Dokument rechts besteht noch immer nur aus **einem** Element, person
- Das Wurzelement (en.: root element) ist das Element OHNE Elternelement (also ein Element, das über dem Element steht)
 - dementsprechend ist in diesem XML – Dokument das Wurzelement **person**
- Alle anderen Dokumente sind innerhalb des Wurzelements
- Das Wurzelement wird auch manchmal Dokumentenelement genannt
- Jedes wohlgeformte XML – Dokument hat nur exakt **ein** Wurzelement

```
1  <person>
2    <name>
3      <vorname>Alan</vorname>
4      <nachname>Turing</nachname>
5    </name>
6    <beruf>Informatiker</beruf>
7    <beruf>Mathematiker</beruf>
8    <beruf>Kryptograph</beruf>
9  </person>
```

Abb. 2

Achtung: Kleine Info, weil ich nicht weiß wohin damit:
XML ist case – sensitive, achtet also auf Groß-, Kleinschreibung: <PERSON> ist ungleich <person> oder <Person>

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Elemente und Tags

- Im Dokument rechts sehen wir Elternelemente und Kindelemente
 - name und die drei Beruf Elemente sind Kindelemente von person, person ist Elternelement von den eben genannten Kindelementen
 - vorname und nachname sind Kindelemente von name, dementsprechend ist name Elternelement von vorname und nachname
- Die Elemente name und die drei Beruf – Elemente werden auch „Geschwister“ (en.: siblings) genannt
- Diese Struktur ist Informatikern bekannt als Baumstruktur...

```
1 <person>
2   <name>
3     <vorname>Alan</vorname>
4     <nachname>Turing</nachname>
5   </name>
6   <beruf>Informatiker</beruf>
7   <beruf>Mathematiker</beruf>
8   <beruf>Kryptograph</beruf>
9 </person>
```

Abb. 2

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Baumstruktur

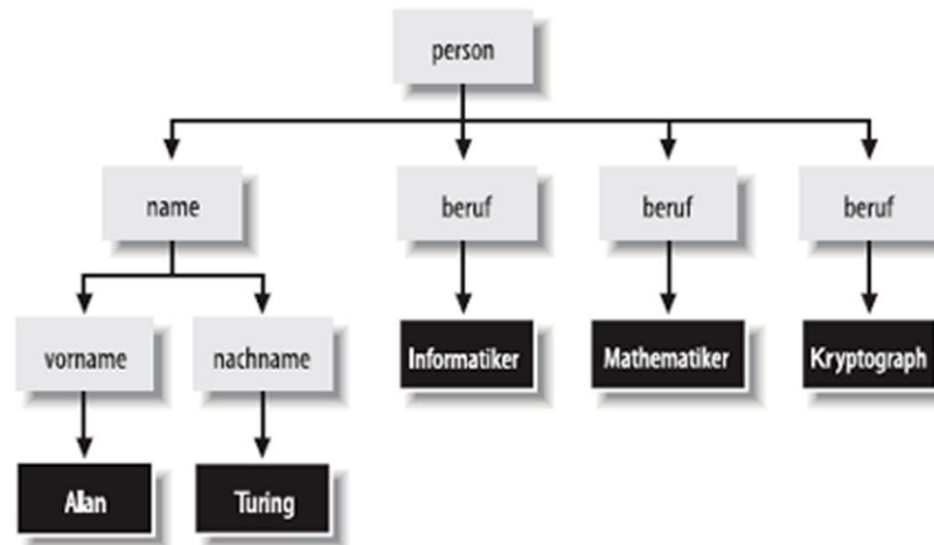


Abb. 3

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Attribute

- XML – Elemente können wie auch HTML – Elemente Attribute haben
- Attribut ist ein Key – Value Pair, das dem Start – Element zugewiesen ist
- Der Key wird vom Value durch ein Gleichheitszeichen getrennt, optional mit Whitespace vor und nach dem Gleichheitszeichen (s. Abb. 4 und Abb. 5)
- Whitespace dient allein der persönlichen Ästhetik, ist aber auch in manchen Styleguides vorgeschrieben
- In der Regel verwendet man Attribute, um die Elemente weitergehend zu beschreiben, bzw. Meta – Daten zum Element zur Verfügung zu stellen z.B. eine URL – Adresse, Quellen, generell Hyperlinks

```
1 <person geboren="1912-06-23" gestorben="1954-06-07">
2   Alan Turing
3 </person>
```

Abb. 4

```
1 <person gestorben = '1954-06-07' geboren = '1912-06-23'>
2   Alan Turing
3 </person>
```

Abb. 5

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Wohlgeformtheit

- Ausnahmslos alle Dokumente müssen wohlgeformt sein, ansonsten weist der XML – Parser sie ab, grundlegende Regeln, an die man sich halten muss sind:
 1. Jedes Start – Tag muss ein dazugehöriges End-Tag haben
 2. Elemente dürfen geschachtelt sein, sich aber nicht überlappen
 3. Es muss genau ein Wurzelement geben
 4. Attribute müssen in Anführungszeichen stehen
 5. Ein Element darf nicht zwei Attribute mit dem gleichen Namen besitzen
 6. Kommentare dürfen nicht in innerhalb von Tags stehen
 7. keine ungeschützten < > Klammern oder & - Ampersand Zeichen

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

CDATA

- Möchte man XML oder HTML – Quellcode innerhalb eines XML – Dokuments verwenden, ohne die eckigen **Klammern** `<>` sowie auch das **&** **Ampersand** zu escapen mit `<` und `&`;
- Das wäre bei langen Code – Abschnitten eher lästig, deswegen kann diese in ein CDATA Abschnitt packen – alles bis auf die schließenden `]]` – Klammern können im CDATA vorkommen
- CDATA – Inhalt ist Text, der nicht weiter interpretiert wird
- Im Beispiel – Screenshot sieht man ein XHTML – Dokument, dass ein Scalable Vector Grapics (SVG) CDATA Abschnitt enthält

```
<p>Sie können ein vorgegebenes <code>xmlns</code>-Attribut benutzen,
damit Sie nicht allen Ihren Elementen das Präfix svg hinzufügen müssen:</p>
<pre><![CDATA[
  <svg xmlns="http://www.w3.org/2000/svg" width="12cm" height="10cm">
    <ellipse rx="110" ry="130" />
    <rect x="4cm" y="1cm" width="3cm" height="6cm" />
  </svg>
]]></pre>
```

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Dokumenttyp – Defintion (DTD, en.: Document type definition)

- DTDs erklären Anwendungen, wie XML – Dokumente zu lesen sind, welche Regeln für Elemente gelten
- Dadurch können Aussagen getroffen werden wie:
 - ein `` - Tag muss immer in einem `` oder `` Tag stehen
 - oder jedes `<angestellter>` - Element muss ein `<sozialversicherungsnummer>` haben
- Ein validierender Parser vergleicht DTD – Dokument mit XML – Dokument und zeigt Abweichungen auf
- Programm entscheidet selbst, wie es mit diesen Problemen umgeht:
 - Weißt es das XML – Dokument ab? Versucht es das Problem selbst zu lösen?

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Dokumenttyp – Definition (DTD, en.: Document type definition)

- DTDs listen alle Elemente, Attribute und Entities auf, die das Dokument verwendet, sowie den jeweiligen Kontext
- DTD kann auch Einträge enthalten, die das Dokument nicht nutzt
- Alles, was nicht explizit in der DTD erlaubt ist, ist **verboten**
 - alles im Dokument muss einer Deklaration im DTD entsprechen

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Einfaches DTD Beispiel

- Abb. 6 bezieht sich auf Abb. 2
- person hat ein name - Element auf das null oder n – beruf Elemente folgen
- name hat exakt ein vorname – Element, auf das ein nachname – Element folgt
- die Elemente vorname, nachname und Beruf enthalten Text, das man mit #PCDATA definiert
- jede Zeile ist eine Element - Deklaration

```
1 <!ELEMENT person (name, beruf*)>
2 <!ELEMENT name (vorname, nachname)>
3 <!ELEMENT vorname (#PCDATA)>
4 <!ELEMENT nachname (#PCDATA)>
5 <!ELEMENT beruf (#PCDATA)>
```

Abb. 6

```
1 <person>
2   <name>
3     <vorname>Alan</vorname>
4     <nachname>Turing</nachname>
5   </name>
6   <beruf>Informatiker</beruf>
7   <beruf>Mathematiker</beruf>
8   <beruf>Kryptograph</beruf>
9 </person>
```

Abb. 2

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Einfaches DTD Beispiel

```
1 <person>
2   <name>
3     <vorname>Alan</vorname>
4     <nachname>Turing</nachname>
5   </name>
6   <beruf>Informatiker</beruf>
7   <beruf>Mathematiker</beruf>
8   <beruf>Kryptograph</beruf>
9 </person>
```

Abb. 7: gültig nach DTD

```
1 <person>
2   <name>
3     <vorname>Alan</vorname>
4     <nachname>Turing</nachname>
5   </name>
6 </person>
```

Abb. 8: gültig nach DTD, **beruf** darf 0 – n-mal
enthalten sein

```
1 <!ELEMENT person (name, beruf*)>
2 <!ELEMENT name (vorname, nachname)>
3 <!ELEMENT vorname (#PCDATA)>
4 <!ELEMENT nachname (#PCDATA)>
5 <!ELEMENT beruf (#PCDATA)>
```

DTD das Regeln beschreibt

```
1 <person>
2   <beruf>Informatiker</beruf>
3   <beruf>Mathematiker</beruf>
4   <beruf>Kryptograph</beruf>
5 </person>
```

Abb. 9: nicht gültig, Kind **name** fehlt

```
1 <person>
2   <beruf>Informatiker</beruf>
3   <name>
4     <vorname>Alan</vorname>
5     <nachname>Turing</nachname>
6   </name>
7   <beruf>Mathematiker</beruf>
8   <beruf>Kryptograph</beruf>
9 </person>
```

Abb. 10: nicht gültig, Element **beruf** vor Element **name**

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML - Schema

- XML – Schema ist sind die komplizierteren, dafür aber auch mächtigeren DTDs
- Empfehlung des W3C zum Definieren von Strukturen für XML – Dokumente
- Wird selbst in Form eines XML – Dokuments beschrieben
- Unterstützt eine große Zahl Datentypen
- XML - Schema wird auch als XSD (XML Schema Definition) bezeichnet, wird häufig aber nur Schema genannt
 - endet häufig auf die Dateiendung **.xsd**

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Schema, einfacher Typ

- `xs:string`, `xs:decimal`, `xs:integer`, `xs:float`, `xs:boolean`, `xs:date`, `xs:time`
- Schema – Datentypen ähneln den Datentypen, die wir aus uns bekannten Programmiersprachen kennen
- An dem Screenshot rechts erkennt man beispielhaft den Aufbau eines einfachen Typen, sowie Restriktionen und weitere Regeln

```
<xs:simpleType name="monatInt">  
  <xs:restriction base="xs:integer">  
    <xs:minInclusive value="1"/>  
    <xs:maxInclusive value="12"/>  
  </xs:restriction>  
</xs:simpleType>  
<xs:simpleType name="monate">  
  <xs:list itemType="monatInt"/>  
</xs:simpleType>
```

Abb. 11

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

XML – Schema, komplexer Typ

- XML – Datentypdefinition bietet auch die Möglichkeit, komplexe Elementstrukturen zusammenhängend zu definieren
- Diese Strukturen können weitere Elemente und Attribute enthalten
- Die Abbildung soll beispielhaft zeigen, was mit komplexen Typen möglich ist

```
<xs:complexType name="pc-Typ">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="hersteller" type="xs:string"/>
    <xs:element name="prozessor" type="xs:string"/>
    <xs:element name="mhz" type="xs:integer" minOccurs="0"/>
    <xs:element name="kommentar" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:integer"/>
</xs:complexType>
```

Abb. 12

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Bild Quellen

- Abb. 1: <https://www.data2type.de/xml-xslt-xslfo/xml/xml-in-a-nutshell/xml-grundlagen/xml-dokumente-und-xml-dateien>
- Abb. 2: <https://www.data2type.de/xml-xslt-xslfo/xml/xml-in-a-nutshell/xml-grundlagen/elemente-tags-und-zeichendaten>
- Abb. 3: https://www.data2type.de/fileadmin/images/XML_In_A_Nutshell/Baumdiagramm_fuer_Code-Beispiel.png
- Abb. 4 & Abb. 5: <https://www.data2type.de/xml-xslt-xslfo/xml/xml-in-a-nutshell/xml-grundlagen/attribute>
- Abb. 6, 7, 8, 9, 10: <https://www.data2type.de/xml-xslt-xslfo/xml/xml-in-a-nutshell/dokumenttyp-definitionen/validierung>
- Abb. 11, 12: https://de.wikipedia.org/wiki/XML_Schema

TEIL 2 - ABSCHLUSSPRÜFUNG IT-BERUFE

Recherche Quellen:

- <https://www.data2type.de/xml-xslt-xslfo/xml/xml-in-a-nutshell>
- https://de.wikipedia.org/wiki/XML_Schema
- <http://www.edition-w3.de/TR/2001/REC-xmlschema-0-20010502/>
- <http://www.edition-w3c.de/TR/xmlschema-1/>
- <http://www.edition-w3c.de/TR/xmlschema-2/>



Gemeinsam weiterbilden



Ende der Wiederholung

**TEIL 2 ABSCHLUSSPRÜFUNG
IT-BERUFE
Fachinformatiker AE**

- gestreckte Abschlussprüfung Teil 2

PV-AP2 XML Grundlagen / Wiederholung