Einführung in XML

Universität Osnabrück
- Rechenzentrum -
Dipl.-Math. Frank Elsner
Frank.Elsner@rz.uni-osnabrueck.de
Version 1.1 vom 01.03.04

Inhaltsverzeichnis

Einleitung

Diese Einführung in XML setzt grundlegende Kenntnisse in HTML und der Bedienung von Windows Programmen (oder UNIX Programmen) sowie grundlegende Kenntnisse des Internets voraus.

Was ist XML

Die Abkürzung XML steht für "Extensible Markup Language", zu deutsch in etwa "erweiterbare Auszeichnungssprache". XML ist ein internationaler Standard des World Wide Web Konsortiums (W3C), das u.a. auch HTML standardisiert hat.

XML wird als erweiterbar (englisch: extensible) bezeichnet, weil es keine feste Anzahl von Auszeichnungselementen (englisch: tags) vorgibt wie zum Beispiel HTML. Stattdessen ist XML in Wirklichkeit eine 'Metasprache' – eine Sprache zur Beschreibung von Sprachen –, die den Entwurf eigener, angepaßter Auszeichnungssprachen für beliebig viele unterschiedliche Dokumenttypen ermöglicht.

XML-basierte Auszeichnungssprachen

Auf Basis von XML sind u.a. folgende wichtigen Auszeichnungssprachen definiert worden:

1. XML for HTML (XHTML) [als Ersatz für HTML]
2. XML for Scalable Vector Graphics (SVG)
3. XML for Synchronized Multimedia Integration Language (SMIL)
4. XML for Mathematical Markup Language (MathML)
5. XML for DocBook
6. XML for Wireless Application Protocol / Wireless Markup Language (WAP/WML)
7. XML for StarOffice / OpenOffice

[Auszüge aus: COVER_XMLAPPS und WEBREF_XML_REFS]

Merkmale von XML

Das World Wide Web Konsortium (W3C) beschreibt XML mit den folgenden sieben charakteristischen Merkmalen:

1. XML bietet eine Methode zum Einfügen strukturierter Daten in eine Textdatei.
2. XML ähnelt HTML ein wenig.
3. XML wird von Maschinen gelesen, ist aber dem Menschen verständlich.
4. XML umfasst eine ganze Familie von Technologien.
5. XML ist wortreich.
6. XML ist relativ neu, hat aber bedeutende Wurzeln.
7. XML ist lizenzfrei, plattformunabhängig und wird breit unterstützt.

[gekürzte Übersetzung aus W3C_10P]

Wie ist ein XML Dokument aufgebaut?

In diesem Kapitel lernen Sie die grundlegenden Bestandteile eines XML Dokumentes kennen.

Element und Inhalt

Ein XML Dokument besteht mindestens aus einem Element und dem zugehörigen Inhalt:

```
<person>Minette Walters</person>
```

1. Ein XML-Dokument muss mindestens ein Element (englisch: tag) und kann mehrere Elemente beinhalten. Im Beispiel gibt es genau ein Element mit Namen "person".
2. Ein Element besteht aus einem Start-Bezeichner, dem Inhalt und einem Ende-Bezeichner. Im Beispiel lautet der Bezeichner "person" und bezieht sich auf eine Person; der Inhalt besteht aus dem Text "Minette Walters" (einer Krimi-Autorin).

Wurzel-Element (englisch: root element)

Ein XML Dokument besteht aus genau einem Wurzel-Element und weiteren, untergeordneten Elementen:

```
<person>
   <name>Minette Walters</name>
   <title>Shapes of Snakes</title>
</person>
```

1. Es gibt genau ein Element, das Wurzel-Element genannt wird (auch Basis-Element, englisch: root element), und keiner seiner Teile befindet sich innerhalb eines anderen Elements. Im Beispiel ist das Element "person" das Wurzel-Element.
2. Das Wurzel-Element kann weitere Elemente enthalten, die korrekt ineinander verschachtelt sein müssen. Im Beispiel gibt es die untergeordneten Elemente "name" (Vorname und Name) und "title" (Buchtitel).

Wiederholungen untergeordneter Elemente

Ein XML Dokument kann untergeordnete Elementen wiederholt enthalten:

```
<person>
   <name>Minette Walters</name>
   <title>Shapes of Snakes</title>
   <title>Echo</title>
   <title>Wave Breakers</title>
</person>
```

1. Im Beispiel gibt es das untergeordnete Element "title", das mehrfach auftaucht.

Schachtelungen untergeordneter Elemente

Ein XML Dokument kann ineinander verschachtelte untergeordnete Elementen enthalten:

```
<person>
   <name>Minette Walters</name>
   <books>
      <title>Shapes of Snakes</title>
      <title>Echo</title>
      <title>Wave Breakers</title>
   </books>
</person>
```

1. Im Beispiel gibt es das untergeordnete Element "books", das seinerseits das Element "title" als mehrfaches Unter-Element besitzt.

Attribute von Elementen

Ein XML Dokument kann Elemente mit Attributen enthalten. Attribute geben genauere Informationen zu einem Element.

```
<person>
   <name>Minette Walters</name>
   <born date="17.5.1948" />
</person>
```

1. Im Beispiel gibt es das Element "born" mit dem Attribut "date" (Geburtsdatum). Da das Element "born" keinen weiteren Inhalt benötigt, wird es in einer Kurzform notiert, die einen Ende-Bezeichner ersetzt. Beachten Sie das Leerzeichen vor dem Zeichen "/".
2. Der Wert eines Attributs ist von dessen Namen durch "=" getrennt. Der Wert wird innerhalb von Apostrophen ('single-quotes') oder Anführungszeichen ("double-quotes") notiert.
3. Wenn Anführungsstriche mit Apostroph (single-quotes) oder Anführungszeichen (double-quotes) für den Wert des Attributs verwendet werden, muß dieser innerhalb des Wertes in dem jeweils anderen Zeichen notiert werden.

```
<person>
   <name>Minette Walters</name>
   <born>
      <day>17</day>
      <month>5</month>
      <year>1948</year>
   </born>
</person>
```

1. In diesem Beispiel ist das Geburtsdatum "born" über drei untergeordnete Elemente "day", "month" und "year" definiert. Hieraus wird deutlich, daß sich Attribute auch in untergeordnete Elemente umsetzen lassen. Es ist Geschmackssache (und ggf. auch abhängig von der Art der Verarbeitung), welche Variante verwendet wird.

Inhalt mit reservierten Zeichen

Einige Zeichen dürfen nicht im Text (und auch nicht in Namen) verwendet werden, weil sie zur Abgrenzung von Bezeichner (markup, tag) und Inhalt bzw. zur Abgrenzung von Werten eines Attributes verwendet werden.

```
<example>

&gt;
&lt;
&quot;
&apos;<
&amp;

</example>
```

1. Die Zeichen < und & dürfen nicht im Inhalt eines Elementes vorkommen, da sie Teil der Auszeichnungssprache sind.
2. Wenn diese Zeichen gebraucht werden, wird < statt < und & statt & notiert.
3. Die Zeichen >, " und ' sind durch >, " und ' zu ersetzen

Zulässige Namen für Bezeichner und Attribute

Die Namen für Bezeichner und Attribute können relativ frei gewählt werden:

```
<The_Person_Is_Named>Minette Walters</The_Person_Is_Named>
```

1. Der Name eines Elements (und eines Attributes) darf Buchstaben, Ziffern, Binde- oder Unterstriche oder Punkte enthalten.

2. Ein Doppelpunkt darf nur in einem bestimmten Fall benutzt werden, in dem er den Elementnamen von seinem Namensraum ("namespace") trennt.
3. Elementnamen, die mit "xml" (egal ob in Groß- oder Kleinbuchstaben oder einer Kombination von Groß- und Kleinbuchstaben) anfangen, sind vom XML-Standard reserviert und dürfen nicht verwendet werden.
4. Alle anderen Sonderzeichen wie Leerzeichen " ", Fragezeichen "?" usw. sind nicht zulässig.

Deklaration, Verarbeitungsanweisungen und Kommentare

Ein vollständiges XML Dokument kann - muß aber nicht - folgende weiteren Bestandteile enthalten:

1. XML Deklaration (XML declaration)
2. Processing Instructions (PI)
3. Kommentare

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!-- This is data about Minette Walters, a famous thriller author. -->

<person>
   <name>Minette Walters</name>
   <title>Shapes of Snakes</title>
   <title>Echo</title>
   <title>Wave Breakers</title>
</person>

<?php
   echo "Current date: ...."
?>
```

1. Im Beispiel wird das XML Dokument in der ersten Zeile mit der Versionsnummer 1.0 von XML deklariert.
2. Die zweite Zeile enthält einen Kommentar.
3. Kommentare können im gesamten Dokument, außer in einem Bezeichner (Markup, tag) vorkommen. Ein XML-Prozessor kann einer Anwendung den Text der Kommentare zur Verfügung stellen, muss das aber nicht.
4. Die Zeichenkette "--" darf in Kommentaren nicht vorkommen.
5. Processing Instructions (PI) sind speziell für Anwendungen vorgesehen, die ein XML Dokument verarbeiten. Ein Beispiel lautet <?php ... ?>, wobei hiermit der PHP Prozessor instruiert wird, die eingebetteten Anweisungen auszuwerten.

Beispiele für fehlerhafte XML Dokumente

```
<books>
   <book>Waves</Book>
   <Book>Snakes</book>
</books>
```

1. Der Name im Ende-Bezeichner muss mit dem Namen im Start-Bezeichner übereinstimmen. Groß- und Kleinschreibung wird unterschieden; d.h. Namen für Elemente sind „empfindlich“ für Groß- oder Kleinschreibung (case-sensitive). Im Beispiel sind Anfangs- und Ende-Bezeichner jeweils unterschiedlich.

```
<person>
   <name>Minette Walters
   <title>Shapes of Snakes</name></title>
</person>
```

1. Wenn sich der Start-Bezeichner im Inhalt eines anderen Elements befindet, muss sich der Ende-Bezeichner im gleichen Element befinden. Anders ausgedrückt: Elemente müssen nur korrekt ineinander verschachtelt sein. Im Beispiel befindet sich der Ende-Bezeichner von "name" innerhalb des Inhaltes von "title" und ist damit fehlerhaft verschachtelt.

```
<person>
   <name>
</person>
```

1. Ein Element ohne Inhalt kann in einer Sonderform mit nur einem Start-Bezeichner notiert werden

(<name/>). Der Schrägstrich vor der schließenden Klammer ersetzt dann den Ende-Bezeichner. Im Beispiel fehlt der Ende-Bezeichner von "name" bzw. die entsprechende Kurzform "</name">.

[überarbeitete Fassung von ZVON_XMLBASICS]

Regeln für wohlgeformte XML Dokumente

Zusammengefaßt gelten folgende wichtige Regeln für wohlgeformte XML Dokumente:

1. Es gibt genau ein Wurzel-Element (englisch: root element).
2. Elemente (englisch: elements) können ineinander verschachtelt sein (englisch: nested), sie dürfen sich jedoch nicht überlappen (englisch: non-overlapping).
3. Jeder öffnende Bezeichner (englisch: start tag) benötigt genau einen schließenden Bezeichner (englisch: end tag).
4. Werte von Attributen (englisch: attributes) sind entweder in Apostrophen ('...') oder in Anführungszeichen ("...") eingeschlossen.
5. Ein Element darf nicht zwei Attribute mit dem selben Namen haben.
6. Kommentare (englisch: comments) und Verarbeitungsanweisungen (englisch: processing instructions, PI) dürfen nicht innerhalb von Elementen auftreten.
7. Die Zeichen "<" oder "&" müssen in spezieller Darstellung "<" und "&" verwendet werden.
8. Die Namen von Elementen und Attributen dürfen keine Sonderzeichen enthalten und nicht mit "xml" beginnen.

Übungen

Sind die folgenden XML Dokument wohlgeformt (englisch: well formed) oder nicht? Falls nicht, gegen welche Regel(n) für wohlgeformte XML Dokumente wird verstoßen?

	Dies ist ein XML Dokument!
	<text>Dies ist ein XML Dokument!
	<text>Dies ist ein XML Dokument!</text/>
	<text>Dies ist ein XML Dokument!</text>
	< text>Dies ist ein XML Dokument!</text>
	<Text>Dies ist ein XML Dokument!</text>
	<Über>Minette Walters </Über>
	<Über Die Schriftstellerin>Minette Walters </Über Die Schriftstellerin>
	<xmlFormat>Minette Walters </xmlFormat>
	<born date="18.05.1970" />
	<born date='18.05.1970' />
	<born place="Stratford upon Avon' />
	<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?> <!-- This is data about Minette Walters -- a famous thriller author. --> <person>Minette Walters is famous about writing about ...</person>

Lösungen

TODO

Einführendes Beispiel - Teil 1

In diesem Kapitel wird ein Dokumenttyp für eine Person entwickelt, und anschließend ein XML Dokument dieses Dokumenttyps in einem WWW Browser dargestellt.

Anschließend wird für den Dokumenttyp eine Document Type Definition (DTD) entwickelt.

Aufbau eines Dokuments

Ein neuer Dokumenttyp "person" für eine Person soll folgenden Aufbau haben:

```
<person>
            <first_name>Vorname</first_name>
            <given_name>Nachname</given_name>
            <born>Geburtsdatum</born>
            <profession>Beruf</profession>
            <children>
                        <child>
                                    <child_first_name>Name des 1. Kindes</child_first_name>
                                    <child_born>Geburtsdatum des 1. Kindes</child_born>
                        </child>
            </children>
</person>
```

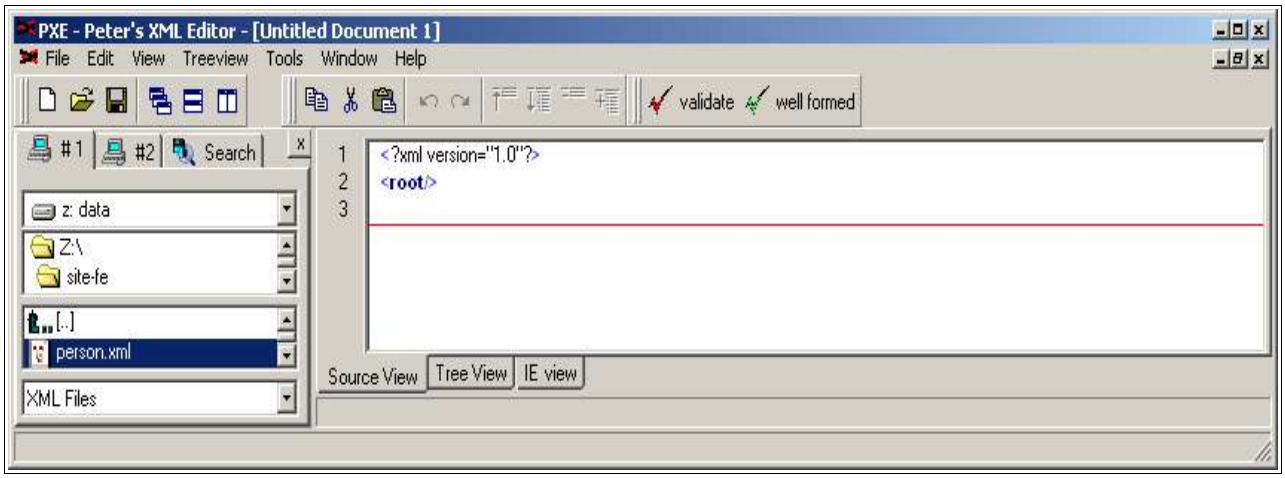
Erfassen eines XML Dokumentes

Installieren Sie zunächst einen XML Editor, um die folgenden XML Dokumente und Stylesheets zu schreiben. Im folgenden wird der Freeware XML Editor Peter's XML Editor, kurz PXE, verwendet.

Ein zugehöriges XML Dokument zum Dokumenttyp "person" könnte folgendermaßen aussehen:

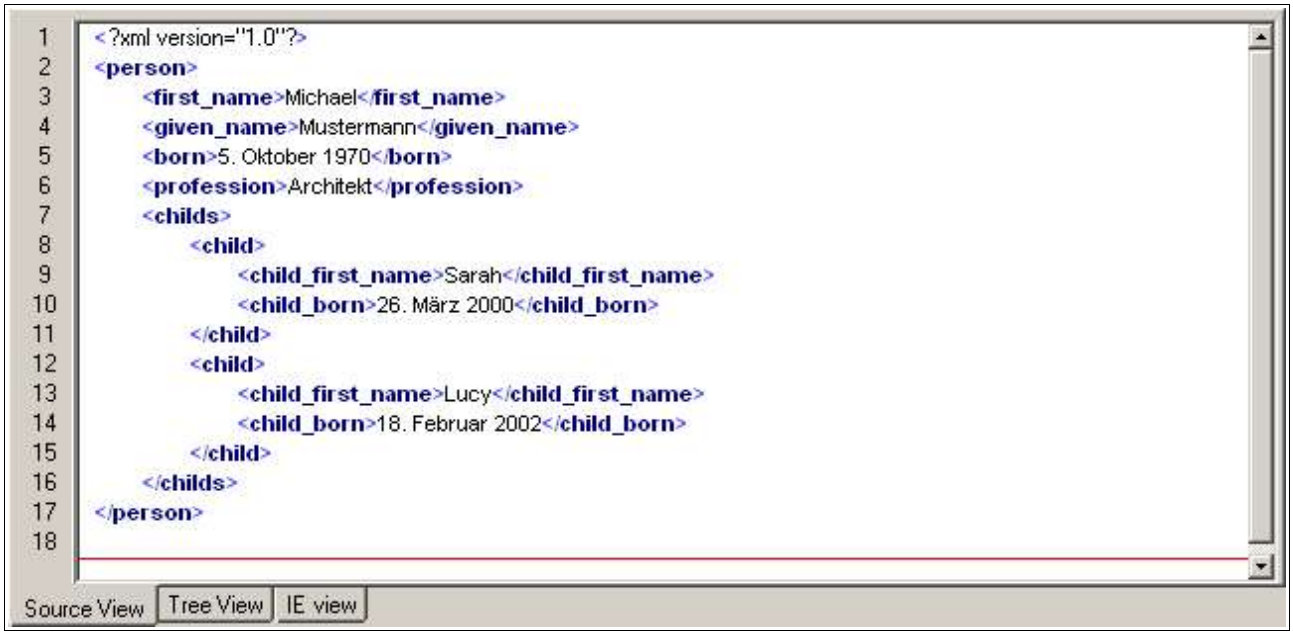
```
<?xml version="1.0"?>

<person>
            <first_name>Michael</first_name>
            <given_name>Mustermann</given_name>
            <born>5. Oktober 1970</born>
            <profession>Architekt</profession>
            <children>
                        <child>
                                    <child_first_name>Sarah</child_first_name>
                                    <child_born>26. April 2000</child_born>
                        </child>
                        <child>
                                    <child_first_name>Lucy</child_first_name>
                                    <child_born>18. Februar 2002</child_born>
                        </child>
            </children>
</person>
```

Starten Sie Peter's XML Editor, im folgenden PXE:

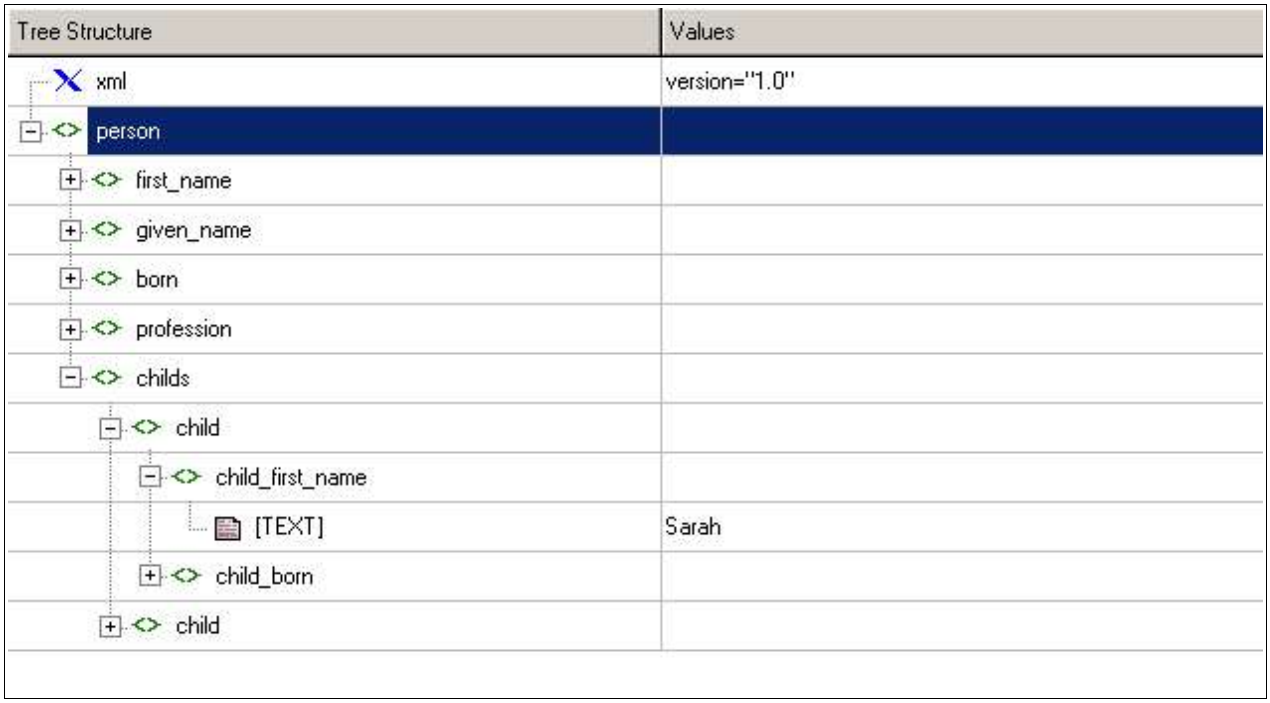


Geben Sie im "Source View"-Modus den Quelltext ein (siehe oben). Die folgenden Abbildungen zeigen das erfaßte XML Dokument im PXE in verschiedenen Ansichten.

Die folgende Abbildung zeigt die Quellcode-Ansicht (Source View):



Die folgende Abbildung zeigt die Baumstruktur-Ansicht (Tree View), wobei das Element "childs" zum Teil aufgeklappt wurde:



Die folgende Abbildung zeigt die Browser-Ansicht (IE Internet Explorer View):

```
<?xml version="1.0" ?>
- <person>
    <first_name>Michael</first_name>
    <given_name>Mustermann</given_name>
    <born>5. Oktober 1970</born>
    <profession>Architekt</profession>
  - <childs>
    - <child>
        <child_first_name>Sarah</child_first_name>
        <child_born>26. März 2000</child_born>
      </child>
    - <child>
        <child_first_name>Lucy</child_first_name>
        <child_born>18. Februar 2002</child_born>
      </child>
    </childs>
  </person>
```

Hinzufügen eines CSS Stylesheets

Die Anzeige im WWW Browser kann über ein Cascading Stylesheet (CSS Stylesheet) manipuliert werden.

Verwenden Sie folgende Datei "person.css" im selben Verzeichnis wie "person.xml", um die Darstellung der Elemente zu defineren:

```
person
{ font-family: Arial;
  font-size: 12pt;
  color : blue
}

first_name, given_name, profession
{ font-weight: bold
}

given_name
{ text-transform   : uppercase;
  background-color : lightgray
}

born
{ font-style: italic
}

child_first_name, child_born
{ font-size: 10pt;
  font-style: italic
}

born, profession, childs
{ display: block
}

child_first_name
{
  display: block;
  margin-left: 5%
}

child_born
{
  display: block;
  margin-left: 10%
}
```

Ergänzen Sie das XML Dokument um eine Processing Instruction (PI) mit der Zeile <?xml-stylesheet ...>, um

das CSS Stylesheet einzufügen und speichern Sie das XML Dokument unter dem Namen "person-with-css.xml" ab.

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

<!-- Add CSS Stylesheet -->
<?xml-stylesheet type="text/css" href="person.css" ?>

...(wie zuvor) ..
```

Beachten Sie, wie sich die Darstellung in der Browser-Ansicht (IE Internet Explorer View):verändert hat:

Definieren einer Document Type Description (DTD)

In der Dokumentbeschreibung (englisch: document type description, DTD) "person.dtd" wird die grundlegende Struktur eines XML Dokuments des Dokumenttyps "person" festgelegt:

1. Das Wurzel-Element "person" beinhaltet einige untergeordnete Elemente in einer festgelegten Reihenfolge.
2. Bis auf das Element "childs" handelt es sich um Elemente, die nur aus Inhalt bestehen und keine untergeordneten Elemente besitzen.
3. Das Element "childs" besitzt kein, ein oder mehrere Unter-Elemente "child".

Diese Festlegungen werden hier in der Datei "person.dtd" im selben Verzeichnis wie "person.xml" getroffen:

```
<!DOCTYPE person [

<!ELEMENT person ( first_name, given_name, born, profession, children ) >
<!ELEMENT first_name ( #PCDATA ) >
<!ELEMENT given_name ( #PCDATA ) >
<!ELEMENT profession ( #PCDATA ) >
<!ELEMENT born ( #PCDATA ) >
<!ELEMENT children ( child* ) >
<!ELEMENT child ( child_first_name, child_born ) >
<!ELEMENT child_born ( #PCDATA ) >
<!ELEMENT child_first_name ( #PCDATA ) >

]>
```

Für jedes Element wird ein Eintrag der folgenden allgemeinen Form verwendet:

```
<!ELEMENT bezeichner ( Inhaltsmodell )>
```

Im einfachsten Fall besteht ein Element nur aus Inhalt und enthält keine untergeordneten Elemente wie zum Beispiel hier für das Element "given_name":

```
<!ELEMENT given_name ( #PCDATA )>
```

Ein Element mit untergeordneten Elemente enthält diese in einer durch Komma separierten Liste.

```
<!ELEMENT bezeichner ( unter-element-1, unter-element-2 ... ) >
```

Dies trifft im Beispiel für das Element "child" zu:

```
<!ELEMENT child ( child_first_name, child_born ) >
```

Ein Element mit mehrfach wiederholten untergeordneten Elemente enthält diese in einer durch Komma separierten Liste wie zum Beispiel für das Element "children", wobei die Wiederholungen durch die Symbole "?" (keinmal oder einmal) , "+" (einmal oder mehrmals) oder "*" (keinmal, einmal oder mehrmals) gesteuert werden.

```
<!ELEMENT bezeichner ( unter-element-1*, unter-element-2?, unter-element-3+ ) >
```

Im Beispiel kann das Element "child" beliebig häufig (oder eben auch gar nicht) als Unter-Element für "children" auftreten:

```
<!ELEMENT children ( child* ) >
```

Automatisches Generieren einer DTD aus einem XML Dokument

Verwenden Sie den XML-To-DTD-Generator (http://www.hitsw.com/xml_utilites/), um sich eine DTD automatisch auf Grundlage eines Beispiel-Dokumentes erzeugen zu lassen.

Hinzufügen einer DTD zu einem XML Dokument

Die DTD kann direkt in das XML Dokument integriert werden, hier: "person-with-css-with-dtd.xml":

Übungen

1. Erweitern Sie die DTD "person.dtd" zu einer DTD für mehrere Personen "person.dtd". Ermöglichen Sie, daß eine Person mehrere Berufe (oder keinen) besitzt und mehrere mittlere Namen (oder keinen) besitzt.
2. Erweitern Sie "persons.dtd" um weitere Elemente.
3. Testen Sie "persons.dtd" mit folgendem XML Dokument "persons-test.xml":

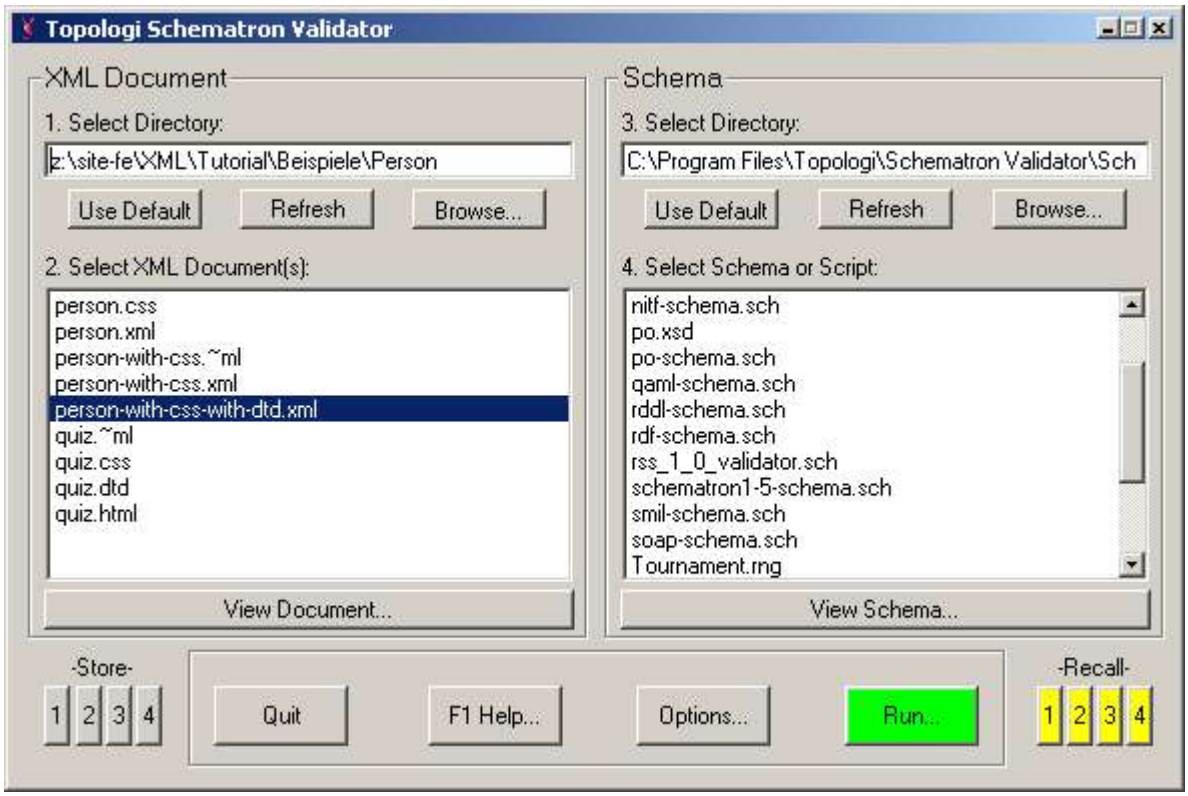
Lösungen

Eine Lösung zu Aufgabe 1 könnte folgendermaßen aussehen:

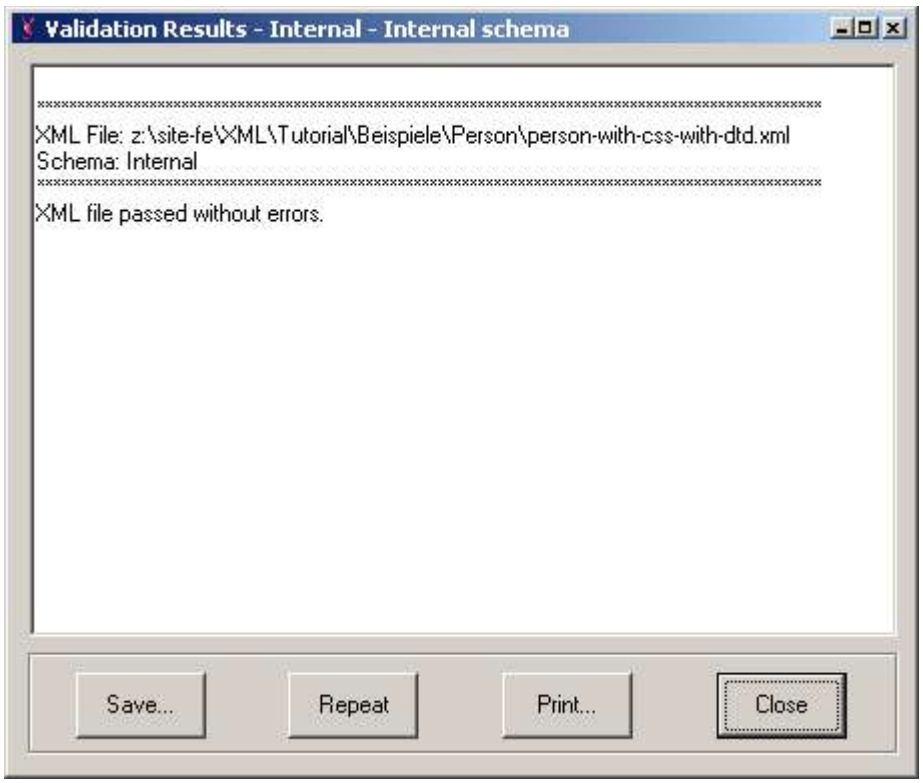
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="person.css" ?>

<!DOCTYPE person [
<!ELEMENT person ( first_name, middle_name*, given_name, born, profession*, children ) >
<!ELEMENT first_name ( #PCDATA ) >
<!ELEMENT given_name ( #PCDATA ) >
<!ELEMENT middle_name ( #PCDATA ) >
<!ELEMENT profession ( #PCDATA ) >
<!ELEMENT born ( #PCDATA ) >
<!ELEMENT children ( child* ) >
<!ELEMENT child ( child_first_name, child_born ) >
<!ELEMENT child_born ( #PCDATA ) >
<!ELEMENT child_first_name ( #PCDATA ) >
]>

<person>
            <first_name>Michael</first_name>
            <given_name>Mustermann</given_name>
            <born>5. Oktober 1970</born>
            <profession>Architekt</profession>
            <children>
                        <child>
                                    <child_first_name>Sarah</child_first_name>
                                    <child_born>26. April 2000</child_born>
                        </child>
                        <child>
                                    <child_first_name>Lucy</child_first_name>
                                    <child_born>18. Februar 2002</child_born>
                        </child>
            </children>
</person>
```

Validieren eines XML Dokumentes gegen eine DTD

Zur Kontrolle, ob das XML Dokument die Regeln der DTD befolgt, verwenden Sie den Topologi Schematron Validator:



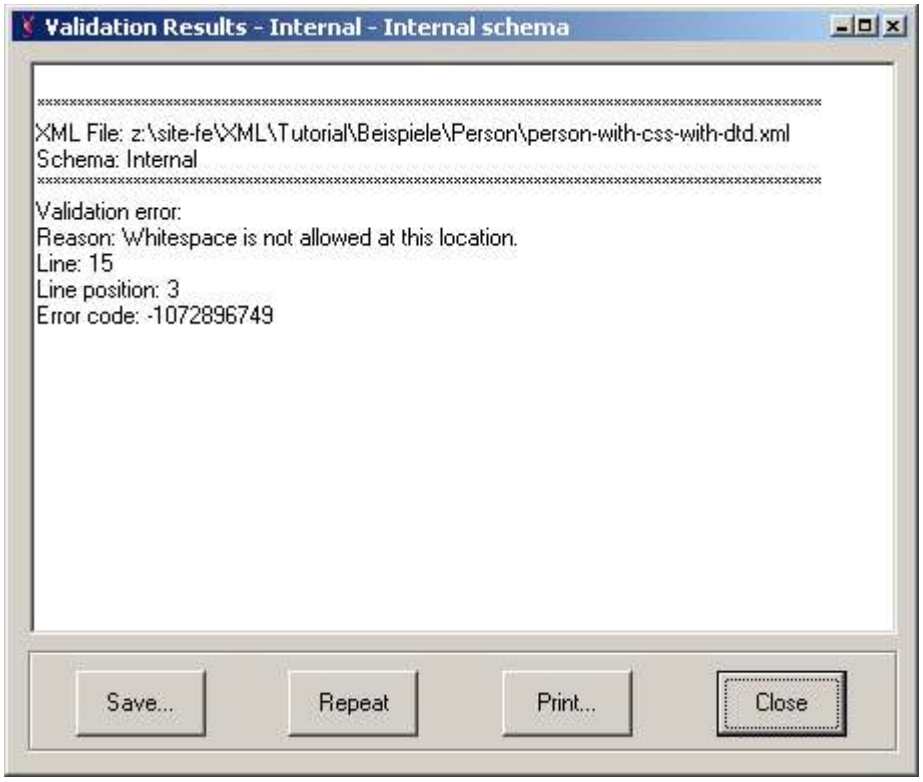
Klicken Sie auf "Run". Im "Validation Results"-Fenster wird eine erfolgreiche Validierung bzw. die entdeckten Fehler angezeigt, hier: keine Fehler.



Verändern Sie beispielsweise den Eintrag für den Vornamen (fehlerhaft) wie folgt:

```
< first_name>Michael</first_name>"
```

Starten Sie die Validierung erneut:



Sie erhalten eine sehr ausführliche Fehlermeldung, die beim Lokalisieren des Fehlers helfen sollte.

Verwenden einer externen DTD

Sie können eine DTD alternativ aus einer externen Datei oder sogar von einer Webseite laden:

Im ersten Beispiel wird die DTD aus dem selben Verzeichnis geladen:

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

<?xml-stylesheet type="text/css" href="person.css" ?>

<!DOCTYPE person SYSTEM "person.dtd">

...(wie bekannt) ...
```

Im zweiten Beispiel liegt die DTD auf einem Web Server:

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

<?xml-stylesheet type="text/css" href="http://myserver/mydtds/person.css" ?>

<!DOCTYPE person SYSTEM "person.dtd">

...(wie bekannt)...
```

Übungen

1. Erweitern Sie die DTD für Person um folgende Elemente:

Lösungen

Die Datei person.dtd enthält eine mögliche Lösung der Aufgabe 1:

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

<?xml-stylesheet type="text/css" href="persons.css" ?>

<!DOCTYPE persons [
<!ELEMENT persons (person* ) >
<!ELEMENT person ( first_name, middle_name*, given_name, born, professions, children ) >
<!ELEMENT first_name ( #PCDATA ) >
<!ELEMENT middle_name ( #PCDATA ) >
<!ELEMENT given_name ( #PCDATA ) >
<!ELEMENT professions ( profession* ) >
<!ELEMENT profession ( #PCDATA ) >
<!ELEMENT born ( #PCDATA ) >
<!ELEMENT children ( child* ) >
<!ELEMENT child ( child_first_name, child_middle_name*, child_born ) >
<!ELEMENT child_born ( #PCDATA ) >
<!ELEMENT child_first_name ( #PCDATA ) >
<!ELEMENT child_middle_name ( #PCDATA ) >
]>

<persons>
<person>
<first_name>Michael</first_name>
<given_name>Mustermann</given_name>
<born>5. Oktober 1970</born>
<professions>
<profession>Architekt</profession>
<profession>Maurer</profession>
</professions>
<children>
<child>
<child_first_name>Sarah</child_first_name>
<child_middle_name>Marie</child_middle_name>
<child_born>26. april  2000</child_born>
</child>
<child>
<child_first_name>Lucy</child_first_name>
<child_born>18. Februar 2002</child_born>
</child>
</children>
</person>
</persons>
```

Einführendes Beispiel - Teil 2

In dieser Fortsetzung des Beispiels wird die Darstellung im WWW Browser durch ein Cascading Style Sheet (CSS Stylesheet) gesteuert.

Abschließend wird ein einfaches XSLT Stylesheet entwickelt, um die Darstellung in einem (XSLT-fähigen) WWW Browser zu verbessern.

Hinzufügen eines CSS Stylesheets

Die Anzeige im WWW Browser kann über ein Cascading Stylesheet (kurz, aber falsch: CSS Stylesheet) manipuliert werden.

Verwenden Sie folgende Datei "person.css" im selben Verzeichnis wie "person.xml", um die Darstellung (Formatierung) der Elemente im WWW Browser zu definieren:

```
person
{ font-family: Arial;
  font-size: 12pt;
  color : blue
}

first_name, given_name, profession
{ font-weight: bold
}

given_name
{ text-transform   : uppercase;
  background-color : lightgray
}

born
{ font-style: italic
}

child_first_name, child_born
{ font-size: 10pt;
  font-style: italic
}

born, profession, childs
{ display: block
}

child_first_name
{
  display: block;
  margin-left: 5%
}

child_born
{
  display: block;
  margin-left: 10%
}
```

Ein CSS Stylesheet hat folgenden prinzipiellen Aufbau:

```
bezeichner
{ Attribut-1: "Wert-1";
  Attribut-2: "Wert-2"
}
```

Folgende Attribute werden häufig gesetzt:

TODO

Ergänzen Sie das XML Dokument um eine Verarbeitungsanweisung (englisch: Processing Instruction, PI) mit der Zeile <?xml-stylesheet ...>, um das CSS Stylesheet einzubinden und speichern Sie das XML Dokument unter dem Namen "person-with-css.xml" ab.

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

<!-- Add CSS Stylesheet -->
<?xml-stylesheet type="text/css" href="person.css" ?>

...(wie zuvor) ..
```

Beachten Sie, wie sich die Darstellung des XML Dokumentes "person-with-css.xml" in der Browser-Ansicht (IE Internet Explorer View):verändert hat:

Hinzufügen eines XSLT Stylesheets

Die Anzeige im WWW Browser kann über ein Extensible Stylesheet Language Tranformations Stylesheet (XSLT Stylesheet) manipuliert werden. XSLT Stylesheets können viel allgemeiner dazu verwendet werden, um ein XML Dokument in ein anderes XML Dokument zu transformieren (siehe unten).

In diesem Abschnitt wird Schritt für Schritt ein einfaches XSLT Stylesheet entwickelt, das den Dokumenttyp "person" in einen Dokumenttyp "einfaches XHTML" wandelt.

Verwenden Sie folgende Datei "person-1.xslt" im selben Verzeichnis wie "person.xml", um das XML Dokument "person.xml" (kurz A) in eine neue Form(kurz B) zu transformieren:

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

<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

</xsl:stylesheet>
```

Zur Transformation des XML Dokumentes (A) in ein anderes XML Dokument (B) muß ein XSLT Prozessor verwendet werden.

Im folgenden wird der freie XML Editor Architag XRay verwendet, der einen integrierten XSLT Prozessor besitzt.

Starten Sie XRay, laden Sie XML Dokument und XSLT Stylesheet und definieren Sie eine Transformation. Die folgende Abbildung zeigt XRay mit 4 Fenstern für (von oben links nach unten rechts) (1) XML Dokument (2) XSLT Stylesheet (3) Transformation und (4) HTML Ansicht:

Reines Kopieren von A nach B

Die erste Version des XSLT Stylesheets "person-1.XSLT" sorgt für ein reines Kopieren der Inhalte der Elemente in das Ergebnis-Dokument - das ist natürlich nicht zufriedenstellend.

Hinzufügen des HTML Gerüsts in B

Im nächsten Schritt mit der Datei "person-2.XSLT" wird eine für HTML passende Form erzeugt:

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

<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="person">

<html>
<head>
</head>
<body>
   <xsl:apply-templates />
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

Beachten Sie, daß zunächst eine Regel für das Wurzel-Element "person" definiert wird, die einen passenden "HTML-Rahmen" erzeugt, bevor rekursiv weitere Regeln angewendet werden. In diesem Fall werden die Standard-Regeln angewendet, die einfach nacheinander den Inhalt aller Elemente ausgeben. Folgendes Ergebnis wird bei der Transformation erreicht:

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html">
</head>
<body>
Vorname: Michael<br>
Nachname: Mustermann<br>
Geburtsdatum: 5. Oktober 1970<br>
Beruf: Architekt<br><p></p>Kinder:<ol>
<li> Name: Sarah,
geboren: 26. Juni 2000</li>
<li> Name: Lucy,
geboren: 18. Februar 2002</li>
</ol>
</body>
</html>
```

Hinzufügen von erläuterndem Text in B

Im abschließenden Schritt werden Regeln für die weiteren Elemente in der Datei "person-3.XSLT" hinzugefügt:

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

<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="person">
<html>
<head>
</head>
<body>
Vorname: <xsl:value-of select="first_name"  /><br/>
Nachname: <xsl:value-of select="given_name"  /><br/>
Geburtsdatum: <xsl:value-of select="born"  /><br/>
Beruf: <xsl:value-of select="profession"  /><br/>
   <xsl:apply-templates select="childs" />
</body>
</html>
</xsl:template>

<xsl:template match="childs">
<p/>Kinder:<ol>
 <xsl:apply-templates select="child" />
 </ol>
</xsl:template>

<xsl:template match="child">
<li> Name: <xsl:value-of select="child_first_name"  />,
geboren: <xsl:value-of select="child_born"  /></li>
</xsl:template>

</xsl:stylesheet>
```

Hiermit wird folgende Transformation erreicht:

TODO

Die HTML-Ansicht ist bereits als gelungen zu bezeichnen:

Verarbeiten der XSLT Transformation im WWW Browser

Sofern ein WWW Browser verwendet wird, der XSLT Stylesheets auswerten kann, kann das XSLT Stylesheet direkt über eine Verarbeitungsanweisung (PI) eingebunden werden:

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

<!-- Add XSLT Stylesheet -->
<?xml-stylesheet type="application/xml" href="person-1.XSLT" ?>

...(wie zuvor) ..
```

WWW Browser mit XSLT Unterstützung sind z.Zt. Internet Explorer 6 und Mozilla 1.0.

Von HTML nach XHTML

HTML ist sicher heute und auch in ferner Zukunft die wichtigste Sprache im Web.

In diesem Kapitel werden die Zusammenhänge und die Unterschiede zwischen HTML und XHTML, einer auf XML basierenden Erweiterung von HTML, erläutert.

Ein Beispiel für ein XHTML Dokument

Das folgende Dokument "xhtml.htm" ist ein gültiges HTML Dokument und ebenfalls ein gültiges XHTML Dokument:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="generator" content="HTML Tidy for Windows (vers 1st October 2002), see www.w3.org"/>
<link rel="STYLESHEET" type="text/css" href="xhtml.css" />
<title>FAQ (Frequently Asked Questions)</title>
</head>
<body>
<h1>FAQ zu XML</h1>
<h2><a id="top" name="top">Inhaltsverzeichnis</a></h2>
<ol>
<li><a href="#f1">Frage #1</a></li>
<li><a href="#f2">Frage #2</a></li>
</ol>
</body>
</html>
```

Es werden in diesem HTML Dokument folgende wichtigen Regeln beachtet, die XHTML von (unsauberem) HTML unterscheiden:

- Include the correct DOCTYPE declaration at the beginning of the file.
- Add the attribute xmlns="http://www.w3.org/1999/xhtml" to the <html> tag.
- Close all tags; omitting is not valid anymore.
- Change all tags to lowercase: <P> becomes .
- Correctly specify empty elements: <hr> becomes <hr />.
- Quote all attribute values: .
- Always add attribute values: <hr noshade="noshade">.
- Always use & for & in attributes: <a href="?a=1&b=2">.

IMPORTANT Compatibility Note:
To make your XHTML compatible with today's browsers, you should add an extra space before the "/" symbol like this:
, and this: <hr />.

Automatische Konvertieren von HTML nach XHTML

Dave Raggett's HTML TIDY is a free utility for cleaning up HTML code. It also works great on the hard-to-read markup generated by specialized HTML editors and conversion tools.

HTML Tidy kann zum Beispiel im Zusammenspiel mit dem freien HTML Editor Phase 5 eingesetzt werden.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
            <head>
                        <meta name="generator" content="HTML Tidy for Windows (vers 1st October
2002), see www.w3.org"/>
<link rel="STYLESHEET" type="text/css" href="xhtml.css" />
                        <title>FAQ (Frequently Asked Questions)</title>
            </head>
            <body>
                        <h1>FAQ zu XML</h1>
                        <h2><a id="top" name="top">Inhaltsverzeichnis</a></h2>
                        <ol>
                                    <li><a href="#f1">Frage #1</a></li>
                                    <li><a href="#f2">Frage #2</a></li>
                        </ol>

            </body>
</html>
```

http://webreference.com/xml/reference/xhtml.html
[XHTML_TAGREF]

Weitere Themen

In diesem Kapitel werden weitere Themen behandelt.

TODO: Work in Progress ...

XML for Scalable Vector Graphics (SVG)

TODO

XML for Mathematical Language (MathML)

TODO

XML for Synchronized Multimedia Language (SMIL)

TODO

Von XML nach XML

TODO

Zugreifen auf Elemente eines XML Dokumentes mit XPath

XSLT und XPath

Beispiel: Inhaltsverzeichnis
Beispiel: Link-Liste

TODO

XML und PHP

TO DO

XML to PDF

TODO

XML Datenbanken

TODO

XML CBIR

TODO

Zum Weiterlesen

In dieser Einführung sind einige wichtige Themen nur am Rande gestreift oder gar nicht behandelt worden. Hierzu zählen ohne Anspruch auf Vollständigkeit:

1. XML Path
2. XML Pointer
3. XML Link
4. XSL-FO (wichtig für die Transformation von XML nach PDF)
5. XML Query Language
6. XML Namespaces
7. XML Include
8. SAX und DOM
9. XML API for C, C++, Java, Python, Perl
10. SOAP und WDDX

Folgende Online Bücher und Online Tutorials sind zur Vertiefung des Stoffes geeignet:

1. XML in 10 points
   http://www.w3.org/XML/1999/XML-in-10-points
   [W3C_10P]
2. ZVON XML Tutorial
   http://zvon.org/xxl/XMLTutorial/General_ger/contents.html
   [ZVON_XMLBASICS]
3. Learn XML in 11,5 Minutes
   http://www.geocities.com/SiliconValley/Peaks/5957/10minxml.html
4. What is XML?
   http://www.geocities.com/SiliconValley/Peaks/5957/wxml.html
5. XML in der Praxis
   Henning Behme und Stefan Mintert
   http://www.mintert.com/xml/buch/
6. Migrating from HTML to XML
   http://www.webtechniques.com/archives/2000/07/fischer/
7. Why XML
   http://webdevelopersjournal.com/articles/why_xml.html
8. Learning XML (Ausschnitte aus dem Buch)
   http://www.oreilly.com/catalog/learnxml/chapter/ch02.html#application
9. XML Quickstart
   http://archive.devx.com/projectcool/developer/xmlz/xmlzquick/
   [XMLQUICK]
10. XML – a professional alternative to HTML (iX Artikel)
    http://www.heise.de/ix/artikel/E/1997/06/106/
11. Extending your markup – a XML tutorial
    http://computer.org/internet/xml/xml.tutorial.pdf

Folgende Spezifikationen, Bücher und Artikel sind für fortgeschrittene Autoren und Entwickler geeignet:

1. **XML at W3C**
   http://www.w3.org/XML/
2. **Extensible Markup Language** (XML) 1.0 (Zweite Auflage)
   http://edition-w3c.de/TR/2000/REC-xml-20001006/
3. Intro
   http://xml.coverpages.org/xmlIntro.html
4. XML FAQ
   http://www.ucc.ie/xml/faq.xml
5. XML at W3 Schools

http://www.w3schools.com/xml/

6. The XML Bible (einige Kapitel Online)
   http://www.ibiblio.org/xml/books/bible2/

7. XSL-FO – How to use Apache FOP under Windows
   http://www.pinkjuice.com/howto/FOP_win_start.txt

8. Books about XML
   http://www.ibiblio.org/xml/books/index.html

9. W3C XSL Tools Page
   http://www.w3.org/Style/XSL/tools.html

10. XML Applications
    http://xml.coverpages.org/xmlApplications.html

11. XHTML Tag Reference
    http://webreference.com/xml/reference/xhtml.html
    [XHTML_TAGREF]

12. OpenOffice File Format
    http://xml.gov/presentations/sun/openoffice.pdf

13. StarOffice2HTML Filter
    http://xml.openoffice.org/sx2ml/

14. XML and the Second Generation Web
    http://www.sciam.com/print_version.cfm?articleID=0008C786-91DB-1CD6-B4A8809EC588EEDF

XML Software

Aus dem großem Angebot an freier XML Software (Freeware, OpenSource) und kommerzieller XML Software seien ohne Anspruch auf Vollständigkeit die folgenden Programme empfohlen:

XML Editoren

XML Editoren dienen in der Regel zum Erfassen und Validieren von XML Dokumenten. Gelegentlich stehen zusätzlich Möglichkeiten zur XSLT Transformation zur Verfügung.

1. HTML Kit
   http://www.chami.com/html-kit/
   (Basis-Version: Freeware, Pro-Version: Shareware)
2. Plugins zu HTML Kit
   http://www.chami.com/html-kit/plugins/
3. XMLMind Editor
4. XMLMind FO Converter
5. Peters XML Editor
   http://www.iol.ie/~pxe/download.html

Weitere XML Editoren sind auf folgenden Seiten gelistet:

1. XML Software
   http://www.xmlsoftware.com/editors.html
   http://www.perfectxml.com/Software.asp

XSL-FO Anwendungen

Fortgeschrittene Beispiele für XSL-FO wie z.B. einen "Barcode Generator" oder einen "Schachspiel-Generator" finden Sie unter:

1. AntennaHouse
   RenderX XEP

XEP Beispiele
http://www.renderx.com/barcodes.html

Verwendete Werkzeuge

Dieses Tutorial ist mit **StarOffice für Windows** geschrieben worden und direkt aus StarOffice in das PDF Format exportiert worden. Die XML Beispiele sind mit dem Editor Dreamweaver MX for Windows erstellt worden.