XML: Übungen

Arbeitsanweisungen,
Hintergründe

(noch im Entstehen)

Vorbereitungen

- Linux-Rechner ggf. starten. Einloggen, ggf. GUI starten
- Shell öffnen, Arbeitsverzeichnis wählen/anlegen:
  - Vorschlag: $HOME/kurse/xml/uebungen
- Bereitgestellte Musterdateien finden Sie in folgendem Verzeichnis:
  - **/local0/werntges/kurse/xml/uebungen**
  - Kopieren Sie daraus die Datei xml.dcl in Ihr Arbeitsverzeichnis:
    - > cd ~/kurse/xml/uebungen
    - > cp /local0/werntges/kurse/xml/uebungen/xml.dcl . # Punkt am Ende beachten!
- Emacs starten, XML-Modus wählen
  - Von der Shell aus: > emacs &
  - Im Emacs: M-x xml-mode # M-x = ESC, x nacheinander
  - Rundgang durch die neuen Menüpunkte
    - SGML / Validate: Unser Standardweg, um XML-Dokumente zu validieren und Fehlermeldungen zu prüfen sowie Fehler zu korrigieren.

Vorbereitungen

- Allgemeiner Hinweis:
  - Alternativ können Sie auch von der Kommandozeile aus validieren:
  - > nsgmls -s -wxml xml.dcl myfile.xml # myfile.xml sinngemäß ersetzen
    - Bitte die Warnung „SGML declaration was not implied“ ignorieren
    - Bitte andere Warnungen, die sich nicht auf Ihre Datei(en) beziehen, ebenfalls ignorieren.
- Verfügbarkeitstest: Mozilla (Browser) starten
  - Von der Shell aus: > mozilla & # alternativ per Icon auf GUI
  - Homepage des Dozenten anwählen, Scriptmaterial (PDF) laden:
    Wählen Sie folgende URL an:
    - http://www.informatik.fh-wiesbaden.de/~werntges/home/index.html

Übung 1: XML-Daten betrachten

- Ziele der Übung:
  - Eigenschaften des XML-Modus von Emacs kennenlernen:
    - Syntax highlighting
    - Einrückungen
    - Folding
    - Validierung
  - Einfache Veränderungen austesten
    - Kommentare einbauen, abspeichern, validieren
    - Fehler im Aufbau provozieren, Validiererantwort auswerten.
    - Problemstellen im Text lokalisieren
- Testdatei
  - tempest.xml (Shakespeare!) kopieren nach **ueb01.xml**, diese bearbeiten
- Anzeigetest
  - Datei mit Mozilla laden. Was zeigt Mozilla an?
    - Nur den Inhalt, ohne *markup*!
  - Vergleich mit IE (per Projektor gezeigt)
    - Baumstruktur, incl. Markup - ein ganz anderes Konzept!

Übung 2: Sonderzeichen, *character references*

- Beschreibung
  - Wir üben den Umgang mit Sonderzeichen anhand einiger beispielhaft ausgewählter mathematischen Beziehungen.
  - Zur Erfassung dieser Beziehungen verwenden Sie die in der Vorlesung vorgestellte Syntax für *character references* und die in XML vordefinierten *entities*.
  - Verwenden Sie in dieser Übung folgende Unicodes:
    - alpha: #x3B1, beta: #x3B2, gamma: #x3B3, Grad: #xB0
- Abzubildende mathematische Beziehungen:
  - Transitivität: a < b und b < c => a < c (1)
  - (Variante): a < b & c > b => a < c (2)
  - Ebenes Dreieck: alpha + beta + gamma = 180° (3)
- Hinweise
  - Verwenden Sie in Gl (1) und (2) die in XML vordefinierten entities.
  - Verwenden Sie in Gl. (3) die o.g. Unicodes.

Übung 2: Sonderzeichen, *character references*

- Als Dokumententyp verwenden wir folgende Struktur:
  - <MathAussagen>
    <MathAussage Typ="x" Nummer='n'>
    <linkeSeite>….</linkeSeite>
    <rechteSeite>…</rechteSeite>
    </MathAussage>
    </MathAussagen>
    mit x aus { Gleichheit, Ungleichheit, Folgerung, Äquivalenz },
    mit n = (optionale) Nummer der Gleichung
- Aufgabe:
  - Erstellen Sie eine Datei ueb02.xml, die aus einem XML-Dokument vom Typ „MathAussagen" besteht und die drei o.g. Beziehungen als drei Instanzen vom Element „MathAussage" codiert. Codieren Sie die Sonderzeichen in geeigneter Art und Weise!
  - Validierung (Emacs, nsgmls).
  - Anzeige mit mozilla

Übung 3: Einfache *entities* selbst definieren

- Viele Symbole wären über „sprechende“ Abkürzungen leichter handhabbar, analog zu den eingebauten Sonderzeichen
  - Die math. Zeichen <=>, =>, ^ (logisches UND)
  - Das Gradzeichen
  - Die griechischen Buchstaben.
- *Entities* bieten dazu die Möglichkeit
  - Im Prolog des XML-Dokuments lassen sich eigene „Makros“ definieren.
- Aufgabe:
  - Wählen Sie als Ausgangspunkt Ihre math. Testdatei aus Übung 2:
    - cp ueb02.xml ueb03.xml
  - Besorgen Sie sich eine hier zu verwendende DTD-Datei:
    - cp /local0/werntges/kurse/xml/uebungen/ueb03.dtd . # Punkt beachten!
  - Definieren Sie im Prolog Ihrer Dateikopie:
    - „ist-äquivalent“, „logisches-und“, „daraus-folgt“
    - „Grad“
    - „alpha“, „beta“, „gamma“

Übung 3: (Fortsetzung)

- Aufgabe (Forts.)
  - Verwenden Sie dazu die *char. references* aus Übung 2
  - Legen Sie eine document type declaration wie folgt an:
    - <!DOCTYPE MathAussagen SYSTEM „ueb03.dtd“ [
    - <!ENTITY Grad “&xB0;”> <!-- zur Definition und &*name*; zum „Aufruf“ -->
    - <!ENTITY *name* “value”> <!-- usw., analog für *jedes* Sonderzeichen -->
    -]> <!-- zum Schließen der Doctypedecl. -->
    - Einbau so vieler *entity*-Deklarationen in den Prolog, wie Sie benötigen
  - Neben den Unicode-Werten aus Übung 2 verwenden Sie ferner:
    - ä: #xE4, Ä: #xC4
    - logisches-und: #x2227, logisches-oder: #x2228
    - daraus-folgt: #x21D2, ist-äquivalent: #x21D4
  - Ersetzen Sie nun die *character references* in Ihrem Dokumenttext durch die neu definierten *entity references*.
  - Tauschen Sie in Gl. (2) das kaufmännische UND aus gegen das nun definierte „logische UND“.

Übung 3: (Fortsetzung)

- Testen Sie Ihre *entities*
  - Validierung
    - Korrigieren Sie Ihre Datei so lange, bis die Validierung gelingt.
  - Bemerkung:
    - Da erstmals eine DTD zur Verfügung steht (die wier hier nur benutzen, deren Inhalt aber noch keine Rolle spielt), prüfen wir nicht nur auf Wohlgeformtheit, sondern tatsächlich auf Gültigkeit der Datei.
  - Ergebniskontrolle
    - korrekte / erwartete Anzeige in Mozilla?
- Probleme?
  - Möglicherweise lehnte Ihr Parser den Umlaut „ä“ im Wort „äquivalent“ ab.
  - Falls ja, ersetzen Sie ihn durch „ae“

Übung 4: Externe *entities, parameter entities*

- Hintergrund:
  - Symbolsammlungen wie die math. Sonderzeichen oder das griechische Alphabet sind in vielen XML-Dokumenten nützlich.
  - Sie sollten daher als externe entities gepflegt werden und vom jeweiligen Dokument nur „aufgerufen“ / „eingebunden“ werden.
  - Dies ist ein typischer Fall, warum XML-Dokumente physisch oft aus mehreren Dateien / Quellen bestehen!
- Aufgabe
  - cp ueb03.xml ueb04.xml # Auf Übung 3 aufsetzen
  - Hinweis:
    - Führen Sie die folgenden Auslagerungen erst mit einer der drei Dateien aus und testen Sie, bis der Lösungsweg ermittelt ist.
    - Lagern Sie erst dann die anderen beiden Fälle aus!
  - Auslagern aller *entity*-Deklarationen (Ausnahme: s.u.)
    - für die griechischen Buchstaben in die Datei „greek_letters.ent“
    - für math. Sonderzeichen in die Datei „math_symbols.ent“
    - für sonstige Sonderzeichen in die Datei „sonderzeichen.ent“
    - Ausnahme: Verbleib der internen Definition zu „Grad“!

Übung 4: Externe *entities, parameter entities*

- Aufgabe (Fortsetzung)
  - Deklaration der externen *entities* wie folgt ("*external entity declaration*"):
    <! ENTITY greek SYSTEM "greek_letters.ent">
  - Aufruf / Expansion testen:
    - a) innerhalb des *root elements*:
      &greek; <!-- Ergebnis? Aha, mal in der *doc. decl.* versuchen… -->
    - b) innerhalb der Document-Deklaration.
      &greek; <!-- Ergebnis? Hmm, wenn „&" hier nicht erlaubt ist, was dann? -->
  - Für derartige Zwecke führte XML die ***parameter entity*** *ein:*
    <! ENTITY % greek SYSTEM "greek_letters.ent">
  - Aufruf / Expansion testen:
    - a) innerhalb des root elements:
      %greek; <!-- Ergebnis? als „char data" gelesen, so geht's also nicht. -->
    - b) innerhalb der Document-Deklaration:
      %greek; <!-- Ergebnis? Aha, dies wird nun validiert. -->
  - Verfahren Sie nun analog mit den anderen Fällen:
    - <! ENTITY % math SYSTEM "math_symbols.ent"> <!-- an %math; und -->
    - <! ENTITY % xtra SYSTEM "sonderzeichen.ent"> <!-- %xtras; denken! -->

Übung 4: Externe *entities (Forts.)*

- Tests
  - a) Bespiele/Varianten durchtesten, bis die Validierung gelingt. Danach:
  - b) Anzeigentest: Gleiches Ergebnis wie in Übung 3?
- B) Falls der Browser streikt…
  - Sollte die Validierung mit nsgmls gelingen, aber der *Parser* des Browsers mit Fehler abbrechen, versuchen Sie den Weg über interne *parameter entities*, wie folgt am Beispiel „greek" erläutert:
    <! ENTITY % greek "<!ENTITY alpha 'α'> <!…> <!…> ">
    %greek;
  - Vorbereitung:
    cp ueb04.xml cp ueb04b.xml # dann ueb04b.xml editieren
  - Ergänzen Sie sinngemäß. Es dürfen mehrere entities innerhalb der Anführungszeichen aufgelistet werden.
  - Verwenden Sie in diesem Beispiel nur interne *entities.*
  - Tip: Mit „vi" lassen sich die externen entities leicht importieren.
  - (*) Was wäre eine mögliche Erklärung für das Browser-Verhalten?

Übung 4: Externe *entities (Forts.)*

- Test auf Vorrang
  - Versuchen Sie, „alpha“ umzudefinieren, ohne die externe *entity* zu modifizieren, indem Sie „alpha“ intern nochmal definieren, aber mit dem neuen Inhalt „A“.
    - Variante 1: Vor der Deklaration der externen *entity*.
    - Variante 2: Nach der Deklaration der externen *entity*.
  - Fragen
    - Akzeptiert der Validierer die doppelte Definition?
    - Wenn ja: Welche genießt Vorrang? Gilt dies unabhängig von der Position der internen Deklaration (in Bezug auf die externe)?
  - Variante:
    - Verlagern Sie die erneute Definition von „alpha“ in „greek_letters.ent“ hinein, einmal vor und einmal nach der Originaldefinition.
    - Testen Sie das Ausgabeverhalten. „Gewinnt“ die erste oder die letzte Definition?

Übung 5: Zeichensätze

- XML-Deklaration:
  - version
    - Testen des nsgmls-Verhaltens bei falscher Angabe „1.1“
    - Testen des mozilla-Verhaltens bei falscher Angabe „1.1“
  - encoding
    - Test: Parser sollten Zeichensatznamen unabhängig. von Klein/Großschrift akzeptieren. Wirklich?
    - Übung: ISO-8859-1 testen, „ä“ wieder verwenden, falls vorher ersetzt.
      - Liste der üblichen Zeichensätze: s. Vorlesung
  - Unterscheide „Textdeklaration“
    - erste Zeile untergeordneter *entities*. Testen in `greek_letters.ent`

Übung 5: Zeichensätze

- Hinweise
  - `cp ueb04.xml ueb05.xml`
  - `ueb05.xml` modifizieren.
- Ausblick auf UTF-16 codierte Dateien:
  - 0xFEFF - erstes Zeichen bei UTF-16 codierten Dateien:
    - Kein Markup, kein Inhalt - dient nur der Codierungserkennung
    - Hilft, *little-endian* von *big-endian* Maschinen zu unterscheiden.
    - Bedeutung: *Zero-length, non-breakable white space*