



Praktikum zur Veranstaltung XML-Technologie: **Übung 05**

Das „MOM“-Beispiel mittels
XML Schema, Teil 1



Organisatorisches



- Arbeitsverzeichnis:

`~/lv/xmltech/05/`

- Dateinamen:

`05-bestell.xml`

`XMLSchema.dtd`

`05-bestell.dtd`

`datatypes.dtd`

`05-bestell.xsd`

`05-myns.ent`

- Abzugeben:

`05-bestell.xml, 05-bestell.xsd [, 05-myns.ent]`

- Werkzeuge:

`emacs`

`# oder X-Emacs`

`sval`

`# zur Schemavalidierung`

`firefox`

`# zur Nachkontrolle`



- Zur Aufgabe:
 - Ziel ist der Umbau der DTD zur Bestellung in ein XML Schema.
 - Zweistufiges Vorgehen:
 - Aufgabe 05:
 - Zunächst reiner Umbau DTD → Schema
 - Dabei Kennenlernen der neuen Werkzeuge
 - Aufgabe 06:
 - Präzisere Datentypen u.a. Schema-Features!
- Abgabezeitpunkt:
 - Ausnahmsweise spätestens in 14 Tagen.
 - Bem.: Für Aufgaben 05 & 06 stehen insgesamt 3 Wochen zur Verfügung. Es steht Ihnen frei, bereits nach 1 Woche mit Aufgabe 06 zu beginnen.



- Dateien:
 - 05-bestell.xml, 05-bestell.dtd
Aus dem Dozentenverzeichnis kopieren
Dann geeignet modifizieren!
 - Hinweis:
Die beiden Dateien entsprechen den Versionen aus Aufgabe 03, sind aber um das Element „Bestellungen“ erweitert worden, um auch mehrere Bestellungen in einem Dokument erfassen zu können.
 - XMLSchema.dtd, datatypes.dtd
Aus dem Dozentenverzeichnis kopieren
 - 05-bestell.xsd
Selbständig aufzubauen!
ACHTUNG: MIT Namensraum-Unterstützung!



- Werkzeuge:
 - Der DTD-Validierer „nsgmls“ des Emacs ist nicht für Namensräume und XML Schema geeignet.
 - Wir verwenden als Schema-Validierer daher ein neues Werkzeug, das auf der C++ Version der Bibliothek „Xerces“ der Apache Foundation basiert.
 - Xerces ist ein validierender XML-Parser und auch voll *namespace*- und XML Schema-kompatibel!
 - Das Kommandozeilen-Werkzeug „SAXPrint“, das Xerces enthält, schreibt XML-Dateien eigentlich nur „nett formatiert“ nach **stdout**. Implizit prüft der Xerces-Parser die Quelldaten aber und schreibt alle Fehler und Warnungen nach **stderr**.
 - `~werntges/bin/sval` ist ein Wrapper-Script um SAXPrint, das nur die Fehler-Outputs liefert – und dadurch als Validierer arbeitet.
 - Bem.: Der Validierer „xmllint“ ist eine mögliche Alternative. Er hat sich aber noch nicht bewährt im Schema-Kontext. → Option „--schema“



Aufgabe



- A: Verbindung der XML-Datei mit der Schemadatei
 - Legen Sie eine zunächst leere Schemadatei `05-bestell.xsd` an.
 - Bauen Sie nun in `05-bestell.xml` die Attribute ein, mit deren Hilfe der Validierer die Schema-Datei finden kann.
 - Vergeben Sie dazu einen Namensraum-URL
 - Basis: `http://www.informatik.fh-wiesbaden.de/~account/2009/ns`
 - Testen Sie, ob der Validierer die Schema-Datei findet:

```
$ ~werntges/bin/sval 05-bestell.xml
```

Erst wenn Sie mindestens eine Fehlermeldung aus der Schemadatei `05-bestell.xsd` erhalten, können Sie sicher sein, dass diese Datei auch gefunden wurde.
Provozieren Sie ggf. einen Fehler, etwa durch Angabe einer syntaktisch falschen Zeile in der Schemadatei!



- A: Hinweis
 - Der Namensraum-URL muss insgesamt viermal in zwei verschiedenen Dateien vergeben werden.
 - Sie können die drohende Redundanz vermeiden durch:
 - Anlegen einer kleinen Entity-Datei „05-myns.ent“, darin:
 - Eintrag einer geeigneten Entity-Deklaration
 - Einbinden dieses ext. Entity mittels Parameter Entity-Referenzen in den internen Subsets des XML-Dokuments und der Schemadatei
 - Nun möglich: Verwendung von Referenzen an den Stellen, die den Namensraum-URL erhalten sollen.
 - Hinweis: Diese Redundanzvermeidung ist **optional!**



- B: Vorbereitung der Schemadatei
 - Tragen Sie in die Schemadatei `05-bestell.xsd` ggf. einen XML-Prolog ein (s. nächste Seite).
 - Legen Sie Namensräume & Präfix-Werte fest, erstellen Sie dann entsprechend das *start tag* und *end tag* für „Schema“ (incl. der gewählten Präfix-Angaben).
 - Ein Beispiel dazu finden Sie in der Vorlesung - passen Sie es an!
 - Testen Sie mittels Schema-Validierung, ob Ihre Erweiterung funktioniert.
 - Sie sollten nur (zahlreiche) Fehler über inhaltliche Probleme wie unbekannte Elemente und nicht deklarierte Attribute erhalten, aber keine über das Dokumenten-Element selbst.



- B: Vorbereitung der Schemadatei (**optional**)
 - Auch für XML Schema gibt es eine DTD. Sie ist nicht normativ, aber nützlich, etwa für den XML-Modus des Emacs.
 - Sie besteht aus den Dateien **XMLSchema.dtd** und **datatypes.dtd**. Kopieren Sie beide Dateien aus dem Dozentenverzeichnis zu dieser Aufgabe in Ihr Arbeitsverzeichnis „05“.
 - Bauen Sie eine Dokumententyp-Deklaration in den Prolog Ihrer Schema-Datei ein mit Verweis auf **XMLSchema.dtd** .
 - Nun können Sie die Schema-Datei mit dem Emacs komfortabler erstellen und auch wie gewohnt (DTD-) validieren.
 - Hinweise:
 - Man muss *parameter entities* „p“ und „s“ (Präfix und Suffix) sowie das Attribut „xmlns“ noch geeignet im internen *subset* deklarieren.
 - „p“ und „s“ werden einfach als leere Strings deklariert, wenn Sie kein Namensraumpräfix für Schema verwenden.



- **C: Übertragung der DTD - Anlegen der Elemente**
 - Legen Sie für jedes Element der DTD im Schema ein `element`-Element an.
 - Verwenden Sie als Datentyp des Inhalts zunächst nur „string“, wo bisher „#PCDATA“ stand.
 - Verweisen Sie auf enthaltene Elemente mittels Attribut „ref“ – bitte KEINE Elemente implizit anlegen.
 - Prüfen Sie mittels `sva1`, ob Sie Fortschritte machen (weniger Fehler) oder neue Fehler entstehen.
- Hinweise:
 - Beispiele für die Übertragung von DTD-Komponenten finden Sie im Vorlesungsmaterial.



- **D: Übertragung der DTD – Die Attribute**

- Ergänzen Sie die `element`-Elemente nun ggf. um `attribute`-Elemente. Bauen Sie wenn nötig die Elemente so um, dass sie Attribute aufnehmen können.
- Verwenden Sie als Datentypen der Attribute zunächst nur die direkt aus den DTDs stammenden Typen wie „string“ für „CDATA“, „NMTOKEN“, „NOTATION“, „ID“, etc.
- Verweisen Sie ggf. auf `notation`-Elemente.
- Prüfen Sie mittels `sval`, ob Sie Fortschritte machen (weniger Fehler) oder ob neue Fehler entstehen.
- Es sollten nur noch Fehler bez. vermisster Notationen übrig bleiben.



- E: Übertragung der DTD – Die Notationen
 - Was in der DTD eine Notation war, bleibt auch in Schema eine. Legen Sie daher die entsprechenden **notation**-Elemente an, sofern Sie Notationen verwendeten.
 - Prüfen Sie erneut mittels **sval**. Die Fehler sollten nun verschwinden!
- Hinweise
 - Das Verständnis der Attribute **system** und **public** ist offenbar in Schema nicht direkt an XMLs „PublicID“ angelehnt. Verwenden Sie Attribut „**public**“, wenn „**system**“ allein nicht akzeptiert wird!



- Übertragung der DTD in ein XML Schema
 - Wir haben nun die DTD in ein Schema „1:1“ übersetzt. Damit sind die Voraussetzungen geschaffen, um die eigentlichen Vorzüge von XML Schema kennen zu lernen!
 - Teil 2 (Aufgabe 06) zu XML Schema besteht i.w. aus dem Definieren eigener, an die Anforderungen aus Aufgabe 03 an Bestelldaten angepasster Datentypen sowie deren Verwendung, und im Einsatz präziser Wiederholfaktoren für Elemente und Elementgruppen
 - Ziel ist dabei die möglichst präzise Datenmodellierung und damit die Erkennung von fehlerhaften Inhalten möglichst schon auf Senderseite.
 - Der Übergang zwischen den Aufgabenteilen ist fließend...