XML-Technologien

Tutorium im Sommersemester 2012

niels.hoppe@fu-berlin.de

13. Juni 2012

24. April
- Musterlösung
  - XML schreiben
  - XML verstehen
- Namensräume
  - Erklärung
  - Beispiele

8. Mai
- Offene Fragen
- Musterlösung
  - DTDs entwerfen
  - DTDs anwenden
- DTDs
  - Elemente deklarieren
  - Attribute deklarieren
  - Entitäten deklarieren
  - Teilmengen definieren
- XML-Schema
- Übungen

15. Mai
- Offene Fragen
- Musterlösung
- XML-Schema
- Übungen

22. Mai
- Musterlösung
- XSLT und XPath in der Praxis
- Übungen
- Wiederholung REST

29. Mai
- Twitter als Beispiel einer REST-API
  - REST-Basics
  - REST-Constraints
  - Exkurs: Basic authentication vs. OAuth
- XML vs. JSON

5. Juni
- Was ist eigentlich dieses „Linked Data“?
- XML-Technologien im Überblick
  - XML Grundlagen
  - DTD und Schema
  - XSL
- XPath
- Übungen

1. Tutorium - XML Grundlagen

Quelltext anzeigen

Quelltext anzeigen

Erfüllt das XML-Dokument die Anforderungen?

(a) Es ist kompatibel mit XML-1.0-Parsern, da diese erwarten, dass die Version 1.0 oder nicht angegeben ist.
Es ist auch kompatibel mit XML-1.1-Parsern, da diese erwarten, dass die Version angegeben ist und selbst kompatibel zu XML-1.0 sein müssen.

(b) Um ein Element einem Namensraum zuzuordnen, kann man entweder einen Standardnamensraum angeben,

```
<purchaseOrder
        xmlns="http://www.altova.com/IPO"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        orderDate="1999-12-01">
    <shipTo export-code="1" xsi:type="ipo:EU-Address">
    <!-- ... -->
```

(b) Um ein Element einem Namensraum zuzuordnen, kann man entweder einen Standardnamensraum angeben,

```
<purchaseOrder
        xmlns="http://www.altova.com/IPO"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        orderDate="1999-12-01">
    <shipTo export-code="1" xsi:type="ipo:EU-Address">
    <!-- ... -->
```

oder dem Element ein Präfix voranstellen, das zuvor als Namensraum deklariert wurde.

```
<ipo:purchaseOrder
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ipo="http://www.altova.com/IPO"
        orderDate="1999-12-01">
    <ipo:shipTo export-code="1" xsi:type="ipo:EU-Address">
    <!-- ... -->
```

(c) Die Daten `orderDate`, `type`, `export-code` und `partNum` werden als Attribute repräsentiert.

(c) Die Daten `orderDate`, `type`, `export-code` und `partNum` werden als Attribute repräsentiert.

(d) Die Attribute `orderDate`, `export-code` und `partNum` haben kein Präfix. Da Attribute nur durch ein Präfix einem Namensraum zugeordnet werden können, sind sie nicht namensraumeingeschränkt und liegen damit im Null-Namensraum.

(c) Die Daten `orderDate`, `type`, `export-code` und `partNum` werden als Attribute repräsentiert.

(d) Die Attribute `orderDate`, `export-code` und `partNum` haben kein Präfix. Da Attribute nur durch ein Präfix einem Namensraum zugeordnet werden können, sind sie nicht namensraumeingeschränkt und liegen damit im Null-Namensraum.

(e) Das Attribut `type` wird mit dem Präfix `xsi` verwendet, das als der Namensraum `http://www.w3.org/2001/XMLSchema-instance` deklariert ist. Dadurch liegt es in diesem Namensraum.

(a) Um auch die Attribute `orderDate`, `export-code` und `partNum` dem gleichen Namensraum wie die Elemente zuzuordnen, müssen sie mit einem Präfix verwendet werden.

```
<ipo:purchaseOrder
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ipo="http://www.altova.com/IPO"
        ipo:orderDate="1999-12-01">
    <ipo:shipTo ipo:export-code="1" xsi:type="ipo:EU-Address">
    <!-- ... -->
```

(b) Nein, das Dokument ist nicht wohlgeformt bezüglich XML 1.0 und 1.1. Ein XML-Dokument ohne Angabe der Version oder mit Version 1.0 ist nicht wohlgeformt bezüglich XML 1.1 und ein XML-Dokument mit Version 1.1 ist nicht wohlgeformt bezüglich XML 1.0.

(c) Die klassische Betrachtungsweise geht davon aus, dass eine URI entweder ein URL oder ein URN ist. Sowohl für URLs als auch für URNs gibt es dabei verschiedene Schemata, deren Verwendung durch ein Präfix angegeben wird, wie zum Beispiel `http` oder `isbn`.

(c) Die klassische Betrachtungsweise geht davon aus, dass eine URI entweder ein URL oder ein URN ist. Sowohl für URLs als auch für URNs gibt es dabei verschiedene Schemata, deren Verwendung durch ein Präfix angegeben wird, wie zum Beispiel `http` oder `isbn`.

Die aktuelle Betrachtungsweise nimmt keine klare Unterteilung von URIs mehr vor. Schemata werden nur noch für URIs verwendet und als Namensräume bezeichnet. URNs mit dem Präfix `urn` sind ein möglicher Namensraum. URLs werden nur noch als informelle Bezeichnung für solche Ressourcen verwendet, deren Namensraum ihre primäre Zugriffsmethode angibt.

Definition

An **XML namespace** is identified by a URI reference [RFC3986]; element and attribute names may be placed in an XML namespace using the mechanisms described in this specification.

Quelle: `http://www.w3.org/TR/REC-xml-names/#concepts`

Definition

An **XML namespace** is identified by a URI reference [RFC3986]; element and attribute names may be placed in an XML namespace using the mechanisms described in this specification.

Definition

An **expanded name** is a pair consisting of a *namespace name* and a *local name*.

Quelle: `http://www.w3.org/TR/REC-xml-names/#concepts`

Definition

An **XML namespace** is identified by a URI reference [RFC3986]; element and attribute names may be placed in an XML namespace using the mechanisms described in this specification.

Definition

An **expanded name** is a pair consisting of a *namespace name* and a *local name*.

Definition

For a name N in a namespace identified by a URI I, the **namespace name is I**. For a name N that is not in a namespace, the **namespace name has no value**. In either case the **local name is N**.

Quelle: `http://www.w3.org/TR/REC-xml-names/#concepts`

Definition

An **XML namespace** is identified by a URI reference [RFC3986]; element and attribute names may be placed in an XML namespace using the mechanisms described in this specification.

Definition

An **expanded name** is a pair consisting of a *namespace name* and a *local name*.

Definition

For a name N in a namespace identified by a URI I, the **namespace name is I**. For a name N that is not in a namespace, the **namespace name has no value**. In either case the **local name is N**.

Definition

A **qualified name** is a name subject to namespace interpretation.

Quelle: `http://www.w3.org/TR/REC-xml-names/#concepts`

Man verwendet *qualified names* anstelle von *expanded names*. Sie werden entweder als *prefixed names* oder als *unprefixed names* geschrieben.

```
<root>
    <foo
        xmlns="http://www.example.org/default-namespace"
        xmlns:pre="http://www.example.org/another-namespace">

        <bar /><!-- this is in default-namespace -->

        <pre:baz /><!-- this is in another-namespace -->

        <!-- this is in default-namespace -->
    </foo>
    <!-- this is in null-namespace -->
</root>
```

Attribute in XML sind sogenannte assoziierte Knoten. Sie werden nicht wie normale Kindelemente eines Elements behandelt.

- Für Namensräume bedeutet das, dass Attribute nicht im Namensraum des Elements stehen, in dem sie notiert sind, sondern im Null-Namensraum.
- Will man den Namensraum eines Attributes ändern, muss es mit einem Präfix versehen werden:

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
     <a xlink:href="grafik2.svg">Link zur Grafik</a>
</svg>
```

- Es gibt für Attribute keine Möglichkeit, sie ohne Präfix in einen bestimmten Namensraum zu setzen.

Beispiel 1

```
<pre:foo xmlns:pre="http://www.example.org/some-namespace">
    <bar />
</pre:foo>
```

Beispiel 2

```
<foo xmlns="http://www.example.org/some-namespace">
    <bar xmlns="http://www.example.org/another-namespace">
        <baz />
    </bar>
</foo>
```

Beispiel 3

```
<foo xmlns="http://www.example.org/some-namespace">
    <bar xmlns:pre="http://www.example.org/another-namespace">
        <pre:baz attr="value" />
    </bar>
</foo>
```

Beispiel 4

```
<foo xmlns="http://www.example.org/some-namespace">
    <bar xmlns="">
        <baz />
    </bar>
</foo>
```

2. Tutorium - DTDs

Frage Übernehmen Attribute den Namensraum des Elementes zu dem sie gehören, wenn dieser explizit mittels Präfix angegeben ist?

Antwort Nein. Ein Attribut ist **immer** im Null-Namensraum, außer es hat selbst ein Namensraum-Präfix.

Frage Wird Whitespace in der Deklaration des Namensraumes als Null-Namensraum interpretiert?

Antwort Nein. Jedenfalls erkennen einige Validatoren Whitespace als eigenen Namensraum.

DTDs entwerfen

(a) Das `shipTo`- und das `billTo`-Element kann entweder aus den Elementen `name`, `street`, `city`, und `postcode` oder aus den Elementen `name`, `street`, `city`, `state` und `zip` bestehen.

```
<!ELEMENT shipTo (name, street, city,
    (postcode | (state, zip)))>
<!ELEMENT billTo (name, street, city,
    (postcode | (state, zip)))>
```

DTDs entwerfen

(a) Das `shipTo`- und das `billTo`-Element kann entweder aus den Elementen `name`, `street`, `city`, und `postcode` oder aus den Elementen `name`, `street`, `city`, `state` und `zip` bestehen.

```
<!ELEMENT shipTo (name, street, city,
    (postcode | (state, zip)))>
<!ELEMENT billTo (name, street, city,
    (postcode | (state, zip)))>
```

Zusammengefasst mittels einer Parameter-Entity:

```
<!ENTITY % Address "(name, street, city,
␣␣␣␣(postcode | (state, zip)))">
<!ELEMENT shipTo %Address;>
<!ELEMENT billTo %Address;>
```

DTDs entwerfen

(b) Sowohl das `shipTo`- als auch das `billTo`-Element enthält immer ein Attribut `type` aus dem Namensraum http://www.w3.org/2001/XMLSchema-instance, jeweils mit dem Werten `ipo:EU-Address` oder `ipo:US-Address`.

```
<!ATTLIST purchaseOrder xmlns:xsi CDATA #FIXED
"http://www.w3.org/2001/XMLSchem-instance">

<!ATTLIST shipTo xsi:type
    (ipo:EU-Address | ipo:US-Address) #REQUIRED>
<!ATTLIST billTo xsi:type
    (ipo:EU-Address | ipo:US-Address) #REQUIRED>
```

DTDs entwerfen

(b) Sowohl das `shipTo`- als auch das `billTo`-Element enthält immer ein Attribut `type` aus dem Namensraum http://www.w3.org/2001/XMLSchema-instance, jeweils mit dem Werten `ipo:EU-Address` oder `ipo:US-Address`.

```
<!ATTLIST purchaseOrder xmlns:xsi CDATA #FIXED
"http://www.w3.org/2001/XMLSchem-instance">

<!ATTLIST shipTo xsi:type
    (ipo:EU-Address | ipo:US-Address) #REQUIRED>
<!ATTLIST billTo xsi:type
    (ipo:EU-Address | ipo:US-Address) #REQUIRED>
```

`ipo` ist als Namensraumpräfix für http://www.altova.com/IPO definiert.

```
<!ATTLIST purchaseOrder xmlns:ipo CDATA #FIXED
    "http://www.altova.com/IPO">
```

DTDs entwerfen

(c) Das Element `items` besteht aus beliebig vielen `item`-Elementen, mindestens aber aus einem `item`-Element.

```
<!ELEMENT items (item+)>
```

DTDs entwerfen

(d) Alle Elemente einer gültigen Instanz sollen dem Namensraum http://www.altova.com/IPO zugeordnet sein.

```
<!ATTLIST purchaseOrder xmlns CDATA #FIXED
    "http://www.altova.com/IPO">
```

DTDs anwenden

Eine DTD kann entweder direkt eingebettet werden,

```
<?xml version="1.0"?>
<!DOCTYPE BookInfo [
    <!ELEMENT purchaseOrder (shipTo, billTo, Items)>
    <!ATTLIST purchaseOrder
        xmlns:xsi CDATA #FIXED "http://www.w3.org/2001/XMLSchema-i
        xmlns CDATA #FIXED "http://www.altova.com/IPO"
        xmlns:ipo CDATA #FIXED "http://www.altova.com/IPO"
        orderDate CDATA #REQUIRED>
    <!-- ... -->
]>
<ipo:purchaseOrder
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ipo="http://www.altova.com/IPO"
        orderDate="1999-12-01">
<!-- ... -->
```

oder es kann ein Pfad zu einer DTD, die lokal oder im Netzwerk gespeichert ist, angegeben werden.

```
<?xml version="1.0"?>
<!DOCTYPE html SYSTEM "xml_ueb02_lsg.dtd">
<ipo:purchaseOrder
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ipo="http://www.altova.com/IPO"
        orderDate="1999-12-01">
<!-- ... -->
```

DTDs

- Elemente deklarieren
- Attribute deklarieren
- Entitäten deklarieren

Elemente deklarieren

Elemente können verschiedene Arten von Inhalten haben:

```
<!ELEMENT my-element EMPTY>

<!ELEMENT my-element ANY>

<!ELEMENT my-element (#PCDATA)>

<!ELEMENT my-element (#CDATA)>

<!ELEMENT my-element (specified, contents)>
```

Elemente deklarieren

Die Häufigkeit enthaltener Elemente lässt sich festlegen:

```
<!ELEMENT my-element (
    zero-or-once?,
    zero-or-more*,
    once-or-more+,
    exactly-once
)>
```

Elemente deklarieren

Man kann Alternativen angeben:

```
<!ELEMENT my-element (
    either-this
    | or-this
    | or-that
)>
```

Elemente deklarieren

Man kann Gruppierungen vornehmen:

```
<!ELEMENT my-element (
    alice
    | (bob, charlie)
)>
```

Attribute deklarieren

Attribute eines Elementes werden nacheinander aufgelistet:

```
<!ATTLIST my-element some-attribute CDATA "default-value">
```

Attribute deklarieren

Attribute eines Elementes werden nacheinander aufgelistet:

```
<!ATTLIST my-element some-attribute CDATA "default-value">

<!ATTLIST my-element
    required-attribute CDATA #REQUIRED>
    optional-attribute CDATA #IMPLIED>
```

Attribute deklarieren

Attribute eines Elementes werden nacheinander aufgelistet:

```
<!ATTLIST my-element some-attribute CDATA "default-value">

<!ATTLIST my-element
    required-attribute CDATA #REQUIRED>
    optional-attribute CDATA #IMPLIED>

<!ATTLIST my-element fixed-attribute
    CDATA #FIXED "always-this-value">
```

Attribute deklarieren

Attribute eines Elementes werden nacheinander aufgelistet:

```
<!ATTLIST my-element some-attribute CDATA "default-value">

<!ATTLIST my-element
    required-attribute CDATA #REQUIRED>
    optional-attribute CDATA #IMPLIED>

<!ATTLIST my-element fixed-attribute
    CDATA #FIXED "always-this-value">

<!ATTLIST my-element choice-attribute (
    either-this-value
    | or-this-value
    | or-that-value) "either-this-value">
```

Attribute deklarieren

Es gibt eine ganze Menge von Attributtypen:

CDATA	The value is character data
(en1 |en2 |..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

Quelle: `http://www.http://w3schools.com/dtd/dtd_attributes.asp`

Entitäten deklarieren

Entitäten sind Konstanten, die innerhalb von PCDATA-Blöcken ausgewertet werden. Es gibt verschiedene Arten von Entitäten:

- Built-in Entities (z.B. `&amp;` und `&lt;`)
- Character Entities (z.B. `&#243;` und `&#x00F3;`)
- General Entities, selbst definiert
- Parameter Entities, selbst definiert

Entitäten können sowohl innerhalb der DTD deklariert, als auch extern referenziert werden.

Entitäten deklarieren

Beachte bei General Entities:

- Der Name muss ein gültiger XML-Name sein.
- Der Wert interner Entitäten muss wohlgeformt sein.
- Der Wert externer Entitäten muss nicht wohlgeformt sein, wohl aber das Dokument, nachdem sie ausgewertet wurden.
- Der Wert kann andere Entitäten enthalten.

Entitäten deklarieren

Beachte bei General Entities:

- Der Name muss ein gültiger XML-Name sein.
- Der Wert interner Entitäten muss wohlgeformt sein.
- Der Wert externer Entitäten muss nicht wohlgeformt sein, wohl aber das Dokument, nachdem sie ausgewertet wurden.
- Der Wert kann andere Entitäten enthalten.

Parameter Entities ermöglichen es, Strukturen wieder zu verwenden.

Entitäten deklarieren

Und so sehen sie aus:

```
<!ENTITY internal-entity "entity-value">

<!ENTITY external-entity SYSTEM
    "http://www.example.org/entities.dtd">
```

Entitäten deklarieren

Und so sehen sie aus:

```
<!ENTITY internal-entity "entity-value">

<!ENTITY external-entity SYSTEM
    "http://www.example.org/entities.dtd">

<!ENTITIY %parameter-entity "(some,(complex|construct))">
```

Teilmengen definieren

Es ist möglich, in einer internen DTD eine Teilmenge einer externen DTD zu definieren:

```
<!DOCTYPE BookStore SYSTEM "book.dtd" [
    <!ELEMENT BookStore (Book | BookShelf)*>
]>
```

Teilmengen definieren

Es ist möglich, in einer internen DTD eine Teilmenge einer externen DTD zu definieren:

```
<!DOCTYPE BookStore SYSTEM "book.dtd" [
    <!ELEMENT BookStore (Book | BookShelf)*>
]>
```

Beachte dabei:

- Elemente und Attribute können nicht verändert werden.
- Entitäten können überschrieben werden.

XML-Schema

Wer war schon mal auf http://www.w3schools.com?

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Ja.

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Ja.

```
<?xml standalone="yes" ?>
```

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Ja.

```
<?xml standalone="yes" ?>
```

Nein.

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Ja.

```
<?xml standalone="yes" ?>
```

Nein.

```
<?xml version="1.1" encoding="ISO-8859-1" standalone="no" ?>
```

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Ja.

```
<?xml standalone="yes" ?>
```

Nein.

```
<?xml version="1.1" encoding="ISO-8859-1" standalone="no" ?>
```

Ja.

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Ja.

```
<?xml standalone="yes" ?>
```

Nein.

```
<?xml version="1.1" encoding="ISO-8859-1" standalone="no" ?>
```

Ja.

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

Übungen

Welche XML-Deklaration ist zulässig?

```
<?xml encoding="UTF-8" version="1.0" ?>
```

Nein.

```
<?XML version="1.0" encoding="" ?>
```

Ja.

```
<?xml standalone="yes" ?>
```

Nein.

```
<?xml version="1.1" encoding="ISO-8859-1" standalone="no" ?>
```

Ja.

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

Ja.

Übungen

Bei der XML-Deklaration beachten:

- „`xml`“ klein schreiben
- Reihenfolge der Attribute beachten
- `version` immer angeben
- `encoding` ist optional
- `standalone` ist optional

Übungen

Formuliere eine DTD für folgendes XML und beachte dabei:

- Es darf nur einen Baecker geben.
- Der Ofentyp ist optional und im Zweifel ein „Backofen“.
- Der Ofen muss leer sein.

```
<Baeckerei>
    <Baecker>Bernd Backfix</Baecker>
    <Ofen Typ="Steinofen" />
</Baeckerei>
```

Übungen

Formuliere eine DTD für folgendes XML und beachte dabei:

- Es darf nur einen Baecker geben.
- Der Ofentyp ist optional und im Zweifel ein „Backofen“.
- Der Ofen muss leer sein.

```
<Baeckerei>
    <Baecker>Bernd Backfix</Baecker>
    <Ofen Typ="Steinofen" />
</Baeckerei>
```

```
<!ELEMENT Baeckerei (Baecker,Ofen)>
<!ELEMENT Baecker (#PCDATA)>
<!ELEMENT Ofen EMPTY>
<!ATTLIST Ofen Typ CDATA "Backofen">
```

Übungen

Beschreibe kurz, was die fogende DTD über die Keksfabrik aussagt:

```
<!ELEMENT Keksfabrik (Baecker+, Kunden*, Chef?)>
```

Übungen

Beschreibe kurz, was die fogende DTD über die Keksfabrik aussagt:

```
<!ELEMENT Keksfabrik (Baecker+, Kunden*, Chef?)>
```

- Es gibt mindestens einen Bäcker.
- Es kann beliebig viele (auch keine) Kunden geben.
- Es gibt höchstens einen (oder keinen) Chef.

Übungen

Worum handelt es sich bei dem folgenden XML-Fragment?

```
<Inhalt>
    <Anfang>Hier geht es los</Anfang>
    <Ende>Und hier ist Schluss</Ende>
</Inhalt>
```

Übungen

Worum handelt es sich bei dem folgenden XML-Fragment?

```
<Inhalt>
    <Anfang>Hier geht es los</Anfang>
    <Ende>Und hier ist Schluss</Ende>
</Inhalt>
```

Um strukturierten Inhalt.

Übungen

Worum handelt es sich bei dem folgenden XML-Fragment?

```
<Inhalt>
    <Anfang>Hier geht es los</Anfang>
    Und hier ist Schluss
    <Ende />
</Inhalt>
```

Übungen

Worum handelt es sich bei dem folgenden XML-Fragment?

```
<Inhalt>
    <Anfang>Hier geht es los</Anfang>
    Und hier ist Schluss
    <Ende />
</Inhalt>
```

Um gemischten Inhalt.

Übungen

Welche Arten von Inhalten gibt es in XML?

Übungen

Welche Arten von Inhalten gibt es in XML?

- unstrukturierten Inhalt
- strukturierten Inhalt
- gemischten Inhalt
- leeren Inhalt

Übungen

```
<xs:element name="ethnos">
    <xs:complexType>
        <xs:attribute name="self" type="xs:string" use="required"
    </xs:complexType>
</xs:element>
```

Welche der folgenden DTD ist mit der obigen Schema-Definition äquivalent?

```
<!ATTLIST ethnos self (#PCDATA) #REQUIRED>
<!ATTLIST ethnos self PCDATA #REQUIRED>
<!ATTLIST ethnos self CDATA #FIXED>
<!ATTLIST ethnos self PCDATA #IMPLIED>
<!ATTLIST ethnos self CDATA #REQUIRED>
```

Übungen

```
<xs:element name="ethnos">
    <xs:complexType>
        <xs:attribute name="self" type="xs:string" use="required"
    </xs:complexType>
</xs:element>
```

Welche der folgenden DTD ist mit der obigen Schema-Definition äquivalent?

```
<!ATTLIST ethnos self (#PCDATA) #REQUIRED>
<!ATTLIST ethnos self PCDATA #REQUIRED>
<!ATTLIST ethnos self CDATA #FIXED>
<!ATTLIST ethnos self PCDATA #IMPLIED>
<!ATTLIST ethnos self CDATA #REQUIRED>

<!ATTLIST ethnos self CDATA #REQUIRED>
```

Übungen

Was könnte gültig sein?

```
<!xml version="1.0"?>
<!DOCTYPE galaxy SYSTEM "cosmos.dtd">
<galaxy>M85</galaxy>
```

Übungen

Was könnte gültig sein?

```
<!xml version="1.0"?>
<!DOCTYPE galaxy SYSTEM "cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist falsch geöffnet.

Übungen

Was könnte gültig sein?

```
<!xml version="1.0"?>
<!DOCTYPE galaxy SYSTEM "cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist falsch geöffnet.

```
<?XML version="1.0" encoding="UFTF-8"?>
<!DOCTYPE galaxy [<!ELEMENT galaxy (#PCDATA)>]>
<galaxy>M85</galaxy>
```

Übungen

Was könnte gültig sein?

```
<!xml version="1.0"?>
<!DOCTYPE galaxy SYSTEM "cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist falsch geöffnet.

```
<?XML version="1.0" encoding="UFTF-8"?>
<!DOCTYPE galaxy [<!ELEMENT galaxy (#PCDATA)>]>
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag muss klein geschrieben werden.

Übungen

Was könnte gültig sein?

```
<!xml version="1.0"?>
<!DOCTYPE galaxy SYSTEM "cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist falsch geöffnet.

```
<?XML version="1.0" encoding="UFTF-8"?>
<!DOCTYPE galaxy [<!ELEMENT galaxy (#PCDATA)>]>
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag muss klein geschrieben werden.

```
<?xml version="1.0"?>
<!DOCTYPE SYSTEM "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Übungen

Was könnte gültig sein?

```
<!xml version="1.0"?>
<!DOCTYPE galaxy SYSTEM "cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist falsch geöffnet.

```
<?XML version="1.0" encoding="UFTF-8"?>
<!DOCTYPE galaxy [<!ELEMENT galaxy (#PCDATA)>]>
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag muss klein geschrieben werden.

```
<?xml version="1.0"?>
<!DOCTYPE SYSTEM "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Im `DOCTYPE` ist kein Wurzel-Element angegeben.

Übungen

```
<?xml version="1.0">
<!DOCTYPE galaxy SYSTEM
    "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Übungen

```
<?xml version="1.0">
<!DOCTYPE galaxy SYSTEM
    "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist nicht korrekt geschlossen.

Übungen

```
<?xml version="1.0">
<!DOCTYPE galaxy SYSTEM
    "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist nicht korrekt geschlossen.

```
<?xml version="1.0"?>
<!DOCTYPE PUBLIC "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Übungen

```
<?xml version="1.0">
<!DOCTYPE galaxy SYSTEM
    "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Das `xml`-Tag ist nicht korrekt geschlossen.

```
<?xml version="1.0"?>
<!DOCTYPE PUBLIC "http://www.example.com/cosmos.dtd">
<galaxy>M85</galaxy>
```

Nein. Im `DOCTYPE` ist kein Wurzel-Element angegeben.

Übungen

Welchen Wert hat die Entität `&color;` in einem Dokument, das folgende DTD verwendet?

```
<!ENTITY color "red">
<!ENTITY color "black">
<!ENTITY color "">
```

- „red“
- „black“
- „“
- Die DTD ist gar nicht gültig.

Übungen

Welchen Wert hat die Entität `&color;` in einem Dokument, das folgende DTD verwendet?

```
<!ENTITY color "red">
<!ENTITY color "black">
<!ENTITY color "">
```

- „red“
- „black“
- „“ ←
- Die DTD ist gar nicht gültig.

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train />
```

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train />
```

Ja.

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train />
```

Ja.

```
<train><engine>Steamy</engine></train>
```

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train />
```

Ja.

```
<train><engine>Steamy</engine></train>
```

Ja.

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train />
```

Ja.

```
<train><engine>Steamy</engine></train>
```

Ja.

```
<train><engine>Steamy
<wagon>Diner</wagon>
</engine></train>
```

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train />
```

Ja.

```
<train><engine>Steamy</engine></train>
```

Ja.

```
<train><engine>Steamy
<wagon>Diner</wagon>
</engine></train>
```

Nein. Gemischter Inhalt ist für `engine` nicht erlaubt.

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train><engine>Steamy</engine>
<wagon>Diner</wagon><wagon>Sleeper</wagon>
</train>
```

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train><engine>Steamy</engine>
<wagon>Diner</wagon><wagon>Sleeper</wagon>
</train>
```

Ja.

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train><engine>Steamy</engine>
<wagon>Diner</wagon><wagon>Sleeper</wagon>
</train>
```

Ja.

```
<train><engine>Steamy</engine><engine>Diesel</engine>
<wagon>Sleeper</wagon>
</train>
```

Übungen

```
<!ELEMENT train (engine?, wagon*)>
<!ELEMENT engine (#PCDATA)>
<!ELEMENT wagon (#PCDATA)>
```

Ist das folgende XML-Fragment valide bezüglich obiger DTD?

```
<train><engine>Steamy</engine>
<wagon>Diner</wagon><wagon>Sleeper</wagon>
</train>
```

Ja.

```
<train><engine>Steamy</engine><engine>Diesel</engine>
<wagon>Sleeper</wagon>
</train>
```

Nein. Es darf nur ein engine geben.

3. Tutorium - XML Schema

Frage Darf das Attribut `version` in der XML-Deklaration fehlen?

Antwort Nein, es ist sowohl in XML 1.0 als auch in XML 1.1 obligatorisch.

Frage Darf ein Element mit Inhalt vom Typ `ANY`, `#CDATA` oder `#PCDATA` auch leer sein?

Antwort Ja. Nur wenn es leer sein **muss**, muss es als Typ `EMPTY` haben.

Frage Darf in einem Element mit Inhalt vom Typ CDATA eine öffnende spitze Klammer vorkommen?

Antwort Nein. Nur in einem Bereich, der gesondert mit `<![CDATA[ ...]]>` gekennzeichnet ist.

Schemata entwerfen

Quelltext ansehen

Schemata entwerfen

Quelltext ansehen Erfüllt das Schema die Anforderungen?

(a) Das `shipTo`- und das `billTo`-Element einer gültigen Instanz muss ein Attribut type aus dem Namensraum http://www.w3.org/2001/XMLSchema-instance enthalten.

```
<complexType name="Address" abstract="true">
    <sequence>
        <element name="name" type="string" />
        <element name="street" type="string" />
        <element name="city" type="string" />
    </sequence>
    <attribute name="type">
        <simpleType>
            <restriction base="string">
                <enumeration value="ipo:US-Address" />
                <enumeration value="ipo:EU-Address" />
            </restriction>
        </simpleType>
    </attribute>
</complexType>
```

Und kann entweder ...

(a) i. aus den Elementen `name`, `street`, `city`, und `postcode` bestehen, dann muss `type` den Wert `ipo:EU-Address` haben, sowie das Attribut `export-code` mit dem Wert `1` vorhanden sein,

```
<complexType name="EU-Address">
    <complexContent>
        <extension base="ipo:Address">
            <sequence>
                <element name="postcode" type="ipo:EU-Postcode" />
            </sequence>
            <attribute name="export-code" type="positiveInteger" fix
        </extension>
    </complexContent>
</complexType>
```

(a)

ii. oder aus den Elementen `name`, `street`, `city`, `state` und `zip` bestehen, dann muss das Attribut `type` den Wert `ipo:US-Address` haben und das Attribut `export-code` darf nicht vorhanden sein.

```
<complexType name="US-Address">
    <comlexContent>
        <extension base="ipo:Address">
            <sequence>
                <element name="state" type="ipo:US-State" />
                <element name="zip" type="positiveInteger" />
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

(b) Um Elementinhalte und Attribut-Werte (wie z.B. bei `orderDate`, `shipDate`, `quantity`, `price` und `state`) sinnvoll einzuschränken, verwenden wir die eingebauten Datentypen `date`, `decimal` und einen eigenen Datentypen `ipo:Sku`.

```
<element name="shipDate" type="date" />

<element name="price" type="decimal" />

<element name="quantity">
    <simpleType>
        <restriction base="positiveInteger">
            <maxExclusive value="1000" />
        </restriction>
    </simpleType>
</element>

<attribute name="partNum" type="ipo:Sku" />
<simpleType name="Sku">
    <restriction base="string">
        <pattern value="\d{3}-[A-Z]{2}" />
    </restriction>
</simpleType>
```

(c) Das Element `items` besteht aus beliebig vielen `item`-Elementen, mindestens aber aus einem item-Element. Dazu geben wir die Häufigkeit nach oben hin frei. Die untere Beschränkung ist voreingestellt Eins.

```
    <sequence>
        <element name="item" maxOccurs="unbounded">
<!-- ... -->
```

(d) Das Attribut `partNum` des `item`-Elementes soll immer aus 3 Ziffern gefolgt von einem Bindestrich und schließlich zwei Buchstaben bestehen. Dazu schränken wir den Datentyp `string` mit Hilfe eines regulären Ausdruckes ein:

```
<simpleType name="Sku">
    <restriction base="string">
        <pattern value="\d{3}-[A-Z]{2}" />
    </restriction>
</simpleType>
```

(e) Alle Elemente einer gültigen Instanz werden durch die Angabe des Attributes `taregtNamespace` dem Namensraum http://www.altova.com/IPO zugeordnet.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ipo="http://www.altova.com/IPO"
        targetNamespace="http://www.altova.com/IPO"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
```

Schemata anwenden

Um ein XML-Dokument mit einem Schema zu verknüpfen, verwendet man das Attribut `xsi:schemaLocation` im Wurzelelement des Dokuments:

```
<?xml version="1.0"?>
<ipo:purchaseOrder
        xmlns:ipo="http://www.altova.com/IPO"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3schools.com␣note.xsd"
        orderDate="1999-12-01">
```

XML-Schema

Übung

Welches der folgenden Elemente ist ein gültiges Wurzelelement eines XML-Schemas?

1.
```
<schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org">
```

2.
```
<xs:schema targetNamespace="http://www.example.org">
```

3.
```
<xs:schema
xmlns="http://www.example.org">
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

4.
```
<schema xmlns="http://www.example.org">
```

Übung

```
<data href="http://www.example.com/datAdat"/>
```

Welche der folgenden Schema-Definitionen ist am geeignetsten, um das Attribut `href` zu definieren?

1.
```
<xs:attribute name="href" type="xs:string"/>
```

2.
```
<xs:attribute name="href" type="xs:anyURI"/>
```

3.
```
<xs:attribute name="href" type="xs:NMTOKEN"/>
```

4.
```
<xs:attribute name="href" type="xs:anyType"/>
```

5.
```
<xs:attribute name="href" type="xs:QName"/>
```

Übung

In welchen XML-Schema Elementen sollte die folgende Definition enthalten sein?

```
<xsd:extension base="xsd:integer">
<xsd:attribute name="currency" type="xsd:string" />
</xsd:extension>
```

1. `<xsd:simpleType>` `<xsd:simpleContent>`

   `<xsd:simpleType>` `<xsd:complexContent>`

3. `<xsd:complexType>` `<xsd:simpleContent>`

   `<xsd:complexType>` `<xsd:complexContent>`

4. Tutorium - XSL Transformation

XSLT entwerfen und verwenden

Entwerfen Sie ein XSLT-Stylesheet, das aus der Musterlösung der 1. Übung durch eine Transformation folgendes HTML-Dokument erzeugt. Das Stylesheet soll auch für jedes andere XML-Dokument, das dem Schema aus der Musterlösung der 2. Übung genügt, eine sinngemäße Transformation durchführen.

Die Grundstruktur

Zuerst matchen wir auf das Wurzelelement `purchaseOrder` und erzeugen die Grundstruktur unserer HTML-Seite:

```
<xsl:template match="purchaseOrder">
<html>
    <head>
        <title>Order of <xsl:value-of select="@orderDate" />
                </title>
    </head>
    <body>
        <h1>Order of <xsl:value-of select="@orderDate" /></h1>

        <!-- display shipTo and billTo -->

        <h2>Order details</h2>
        <xsl:apply-templates />
    </body>
</html>
</xsl:template>
```

shipTo und billTo

```
<h2>Shipping address</h2>
<xsl:call-template name="address">
    <xsl:with-param name="address" select="shipTo" />
</xsl:call-template>

<h2>Billing address</h2>
<xsl:call-template name="address">
    <xsl:with-param name="address" select="billTo" />
</xsl:call-template>
```

address

Dieses Template erzeugt aus dem als Parameter übergebenen Element eine Adresse abhängig von dessen Attributwert für `xsi:type`:

```
<xsl:template name="address">
    <xsl:param name="address" />
    <xsl:choose>
        <xsl:when test="$address/@xsi:type='ipo:EU-Address'">
            <address class="export">
            <xsl:value-of select="$address/name" /><br />
            <xsl:value-of select="$address/street" /><br />
            <xsl:value-of select="$address/city" /><br />
            <xsl:value-of select="$address/postcode" />
            </address>
        </xsl:when>
        <!-- ... -->
```

address

```
            <!-- ... -->
            <xsl:when test="$address/@xsi:type='ipo:US-Address'">
                <address>
                <xsl:value-of select="$address/name" /><br />
                <xsl:value-of select="$address/street" /><br />
                <xsl:value-of select="$address/city" />,&amp;nbsp;
                <xsl:value-of select="$address/state" />&amp;nbsp;
                <xsl:value-of select="$address/zip" />
                </address>
            </xsl:when>
        </xsl:choose>
    </xsl:template>
```

Tabelle für `items`

Dieses Template matched auf `items` und erzeugt daraus:

4. Berechnungen für die Fußzeile
- Kopfzeile
- Fußzeile

```
<xsl:template match="items">
    <!-- set variables -->
    <table>
        <thead>
            <!-- ... -->
        </thead>
        <tbody>
            <xsl:apply-templates />
        </tbody>
        <tfoot>
            <!-- ... -->
        </tfoot>
    </table>
</xsl:template>
```

Berechnungen für die Fußzeile

```
<xsl:variable name="total-items">
    <xsl:value-of select="count(item)" />
</xsl:variable>
<xsl:variable name="total-pieces">
    <xsl:value-of select="sum(item/quantity)" />
</xsl:variable>
<xsl:variable name="total-price">
    <xsl:call-template name="recursivesum">
        <xsl:with-param name="items" select="item" />
    </xsl:call-template>
</xsl:variable>
```

Kopfzeile . . .

```
<thead>
    <tr>
    <th>partNum</th>
    <th>productName</th>
    <th>quantity</th>
    <th>price</th>
    <th>shipDate</th>
    <th>comment</th>
    </tr>
</thead>
```

. . . und Fußzeile

```
<tfoot>
    <tr>
    <td>Total:</td>
    <td><xsl:value-of select="$total-items" /> items</td>
    <td><xsl:value-of select="$total-pieces" /> pieces</td>
    <td><xsl:value-of select="$total-price" /> &amp;curren;</td>
    <td></td>
    <td></td>
    </tr>
</tfoot>
```

recursivesum

Berechnet die Summe der durch die quantity gewichteten Preise der einzelnen items.

```
<xsl:template name="recursivesum">
    <xsl:param name="items" />
    <xsl:param name="sum" select="0" />
    <xsl:variable name="head" select="$items[1]" />
    <xsl:variable name="tail" select="$items[position()>1]" />
    <xsl:variable name="thissum"
            select="$head/price * $head/quantity" />
    <!-- ... -->
```

recursivesum

```
        <!-- ... -->
        <xsl:choose>
            <xsl:when test="not($tail)">
                <xsl:value-of select="$sum+$thissum" />
            </xsl:when>
            <xsl:otherwise>
                <xsl:call-template name="recursivesum">
                    <xsl:with-param name="sum"
                                    select="$sum+$thissum" />
                    <xsl:with-param name="items" select="$tail" />
                </xsl:call-template>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
```

Alternative in XPath 2.0

Alternativ sollte in XPath 2.0 folgender Ausdruck zu dem gewünschten Ergebnis führen (nicht getestet!):

```
<xsl:variable name="total-price">
    <xsl:value-of select="sum(item/(quantity␣*␣price)" />
</xsl:variable>
```

Tabellenkörper

Innerhalb des Tabellenkörpers werden noch einmal alle Templates angewendet. Es matched hier nur noch das Template, das die Ausgabe der `items` als Tabellenzeilen ausführt.

```
<xsl:template match="item">
    <tr>
        <td><xsl:value-of select="@partNum" /></td>
        <td><xsl:value-of select="productName" /></td>
        <td><xsl:value-of select="quantity" /></td>
        <td><xsl:value-of select="price" /></td>
        <td><xsl:value-of select="shipDate" /></td>
        <td><xsl:value-of select="comment" /></td>
    </tr>
</xsl:template>
```

Templates überschreiben

Mehrere Templates die auf dasselbe Element matchen aber mit unterschiedlichen gleichpriorisierten Kriterien sind ein Zeichen schlechten Programmierstils!
Stattdessen einmal auf das Element matchen, dann innerhalb des Templates bedingte Ausgaben machen und evtl. benannte Templates verwenden:

```
<xsl:if test="price &gt; 10">


<xsl:choose>
<xsl:when> ... </xsl:when>
</xsl:choose>
```

Prioritäten von Templates

- Wenn mehr als ein Template mit der gleichen *import precedence* auf ein Element matched...

Prioritäten von Templates

- Wenn mehr als ein Template mit der gleichen *import precedence* auf ein Element matched. . . (*import precedence*: default Templates $<$ importierte Templates $<$ Templates im eigentlichen Stylesheet)

Prioritäten von Templates

- Wenn mehr als ein Template mit der gleichen *import precedence* auf ein Element matched. . . (*import precedence*: default Templates $<$ importierte Templates $<$ Templates im eigentlichen Stylesheet)
- . . . dann wird das mit der höheren Priorität ausgewählt.

Prioritäten von Templates

- Wenn mehr als ein Template mit der gleichen *import precedence* auf ein Element matched. . . (*import precedence*: default Templates $<$ importierte Templates $<$ Templates im eigentlichen Stylesheet)
- . . . dann wird das mit der höheren Priorität ausgewählt.
- Es ist nicht zulässig, wenn zwei oder mehr Templates mit der gleichen Priorität auf ein Element matchen. (Die meisten XSLT Prozessoren melden aber keinen Fehler, sondern wählen jeweils das **letzte**.)

Prioritäten von Templates

- Rules with match patterns of a single element or attribute name have priority 0
- Rules with match patterns of prefix:* have priority -0.25
- Rules with match patterns which have only a wildcard test (*, @* ,node() ...) have priority -0.5
- Rules with any other patterns, e.g. with location paths or predicates, have priority 0.5
- One can explicitly give a template a priority (`priority="..."`)

Übung

Nach welchen Eigenschaften teilt man XML-Parser ein?

Übung

Nach welchen Eigenschaften teilt man XML-Parser ein?

- pull / push

Übung

Nach welchen Eigenschaften teilt man XML-Parser ein?

- pull / push
- one-step / multi-step

Übung

Nach welchen Eigenschaften teilt man XML-Parser ein?

- pull / push
- one-step / multi-step
- validating / non-validating

Übung

Nach welchen Eigenschaften teilt man XML-Parser ein?

- pull / push
- one-step / multi-step
- validating / non-validating

Zusatzfage: Wann eignet sich welcher Parser?

Übung

Nach welchen Eigenschaften teilt man XML-Parser ein?

- pull / push
- one-step / multi-step
- validating / non-validating

Zusatzfage: Wann eignet sich welcher Parser?
`www.cs.nmsu.edu/~epontell/courses/XML/material/xmlparsers.html`

Übung

Wie geht ein SAX-Parser mit Syntax- und Strukturfehlern in XML-Dokumenten um?

- Er terminiert ohne Ergebnis
- Fängt Syntaxfehler ab, aber keine Strukturfehler
- Fängt Strukturfehler ab, aber keine Syntaxfehler
- Fängt sowohl Syntax- als auch Strukturfehler ab

Übung

Wie geht ein SAX-Parser mit Syntax- und Strukturfehlern in XML-Dokumenten um?

- Er terminiert ohne Ergebnis
- Fängt Syntaxfehler ab, aber keine Strukturfehler
- Fängt Strukturfehler ab, aber keine Syntaxfehler
- Fängt sowohl Syntax- als auch Strukturfehler ab ←

Übung

Welche Eigenschaften hat ein DOM-Parser?

- Liefert in einem Schritt einen kompletten DOM-tree
- Ist ein one-step push-parser
- Ermögicht direkten Zugriff auf Elemente des DOM-trees anhand ihres Namens
- Erlaubt nicht das Modifizieren und Erstellen von XML-Dokumenten

Übung

Welche Eigenschaften hat ein DOM-Parser?

- Liefert in einem Schritt einen kompletten DOM-tree ←
- Ist ein one-step push-parser
- Ermögicht direkten Zugriff auf Elemente des DOM-trees anhand ihres Namens
- Erlaubt nicht das Modifizieren und Erstellen von XML-Dokumenten

Übung

Welche Eigenschaften hat ein DOM-Parser?

- Liefert in einem Schritt einen kompletten DOM-tree ←
- Ist ein one-step push-parser
- Ermögicht direkten Zugriff auf Elemente des DOM-trees anhand ihres Namens ←
- Erlaubt nicht das Modifizieren und Erstellen von XML-Dokumenten

Parser

	one-step / multi-step	push / pull	validating?
SAX	multi-step	push	
StAX	multi-step	pull	
DOM	one-step	pull	

Übung

Schreibe einen XPath-Ausdruck, der

- im gesamten Dokument
- alle `book`-Elemente
- mit einem `author`-Element als Kind

auswählt.

Übung

Schreibe einen XPath-Ausdruck, der

- im gesamten Dokument
- alle `book`-Elemente
- mit einem `author`-Element als Kind

auswählt.

```
//book[author]
```

Übung

Welche Knoten werden mit folgendem XPath-Ausdruck beschrieben?

```
order/item[@item-id]
```

- `item`-Elemente, de Kind von `order` sind und `item-id` als Attribut haben
- `order`-Elemente, die `item-id` als Attribut haben
- die Werte des Attributs `item-id`

Übung

Welche Knoten werden mit folgendem XPath-Ausdruck beschrieben?

```
order/item[@item-id]
```

- `item`-Elemente, de Kind von `order` sind und `item-id` als Attribut haben ←
- `order`-Elemente, die `item-id` als Attribut haben
- die Werte des Attributs `item-id`

Übung

Welcher XPath-Ausdruck wählt den Preis von „Borshch“?

1. `//entree[@cuisine='Borshch']/price`
2. `//entree[dish='Borshch']/price`
3. `//entree[@dish='Russian']/price`
4. `//entree[text()='Borshch']/price`
5. `//entree['Borshch']/price`

Übung

Welcher XPath-Ausdruck wählt den Preis von „Borshch“?

1. `//entree[@cuisine='Borshch']/price`
2. `//entree[dish='Borshch']/price` ←
3. `//entree[@dish='Russian']/price`
4. `//entree[text()='Borshch']/price`
5. `//entree['Borshch']/price`

Übung

Welcher XPath-Ausdruck wählt das erste `manifold`-Element, das ein `riemann`-Attribut hat?

1. `manifold[@riemann[1]]`
2. `manifold[position()=1][@riemann]`
3. `manifold[1]/[@riemann]`
4. `manifold[@riemann][1]`
5. `[1]manifold[@riemann]`

Übung

Welcher XPath-Ausdruck wählt das erste `manifold`-Element, das ein `riemann`-Attribut hat?

1. `manifold[@riemann[1]]`
2. `manifold[position()=1][@riemann]`
3. `manifold[1]/[@riemann]`
4. `manifold[@riemann][1]` ←
5. `[1]manifold[@riemann]`

Übung

Welcher XPath-Ausdruck wählt alle `species`-Elemente, die ein `mutation`-Element haben?

1. `species[mutation]`
2. `species(mutation)`
3. `species/mutation`
4. `species[@mutation]`

Übung

Welcher XPath-Ausdruck wählt alle `species`-Elemente, die ein `mutation`-Element haben?

1. `species[mutation]` ←
2. `species(mutation)`
3. `species/mutation`
4. `species[@mutation]`

5. Tutorium

REST-Basics

Eine REST-Architektur kennt folgende Begriffe

Komponenten Client, (Proxy, Gateway, . . . ,) Server

Ressourcen sind logische Objekte

Repräsentationen bilden den Zustand einer Ressource ab

REST-Constraints

1. Client-Server
2. Stateless
3. Cacheable
4. Layered System
5. (Code-on-demand)
6. Uniform interface

REST-Constraints: Client-Server

Ermöglicht bessere **Aufgabenteilung** (separaton of concerns) und verbessert die **Skalierbarkeit**.

REST-Constraints: Client-Server

Ermöglicht bessere **Aufgabenteilung** (separaton of concerns) und verbessert die **Skalierbarkeit**.

Und bei Twitter?

REST-Constraints: Client-Server

Ermöglicht bessere **Aufgabenteilung** (separaton of concerns) und verbessert die **Skalierbarkeit**.

Und bei Twitter?

Server `http://api.twitter.com`

Client z.B. unser Browser

REST-Constraints: Stateless

Jede Nachricht enthält alle notwendigen Informationen, die dem Empfänger die Verarbeitung erlauben.

- kein gespeicherter Kontext am Server
- Sitzungszustand beim Client (z.B. mittels Cookies, Session IDs)

REST-Constraints: Stateless

Jede Nachricht enthält alle notwendigen Informationen, die dem Empfänger die Verarbeitung erlauben.

- kein gespeicherter Kontext am Server
- Sitzungszustand beim Client (z.B. mittels Cookies, Session IDs)

Und bei Twitter?

REST-Constraints: Stateless

Jede Nachricht enthält alle notwendigen Informationen, die dem Empfänger die Verarbeitung erlauben.

- kein gespeicherter Kontext am Server
- Sitzungszustand beim Client (z.B. mittels Cookies, Session IDs)

Und bei Twitter?
Ja, alle Parameter müssen in der Anfrage stehen (z.B. `screen_name` bei `home_timeline`).

REST-Constraints: Cacheable

Antworten auf Anfragen implizit oder explizit als cachable oder non-cachable klassifizieren

REST-Constraints: Cacheable

Antworten auf Anfragen implizit oder explizit als cachable oder non-cachable klassifizieren

Und bei Twitter?

REST-Constraints: Cacheable

Antworten auf Anfragen implizit oder explizit als cachable oder non-cachable klassifizieren

Und bei Twitter?

Siehe `https://dev.twitter.com/docs/working-with-timelines`

REST-Constraints: Layered System

Unabhängigkeit der einzelnen Komponenten durch beschränkte Sicht nur bis zum Interaktionspartner

REST-Constraints: Layered System

Unabhängigkeit der einzelnen Komponenten durch beschränkte Sicht nur bis zum Interaktionspartner

Und bei Twitter?

REST-Constraints: Layered System

Unabhängigkeit der einzelnen Komponenten durch beschränkte Sicht nur bis zum Interaktionspartner

Und bei Twitter?

Ja, durch die Verwendung der üblichen WWW-Infrastruktur.

REST-Constraints: Uniform interface

Standardisierter Zugriff auf Komponenten zur Vereinfachung der Infrastruktur und verbesserung der Sichtbarkeit von Interaktionen

- universelle Syntax zur Identifikation von Ressourcen
- Manipulation durch wohldefinierte Aktionen auf einer Repräsentation
- alle Inhalte und Informationen zum Zustandsübergang (Hyperlinks) werden an den Client weitergegeben

REST-Constraints: Uniform interface

Standardisierter Zugriff auf Komponenten zur Vereinfachung der Infrastruktur und verbesserung der Sichtbarkeit von Interaktionen

- universelle Syntax zur Identifikation von Ressourcen
- Manipulation durch wohldefinierte Aktionen auf einer Repräsentation
- alle Inhalte und Informationen zum Zustandsübergang (Hyperlinks) werden an den Client weitergegeben

Und bei Twitter?

REST-Constraints: Uniform interface

Standardisierter Zugriff auf Komponenten zur Vereinfachung der Infrastruktur und verbesserung der Sichtbarkeit von Interaktionen

- universelle Syntax zur Identifikation von Ressourcen
  Ja, durch **Verwendung von URIs**
- Manipulation durch wohldefinierte Aktionen auf einer Repräsentation

- alle Inhalte und Informationen zum Zustandsübergang (Hyperlinks) werden an den Client weitergegeben

Und bei Twitter?

REST-Constraints: Uniform interface

Standardisierter Zugriff auf Komponenten zur Vereinfachung der Infrastruktur und verbesserung der Sichtbarkeit von Interaktionen

- universelle Syntax zur Identifikation von Ressourcen
  Ja, durch **Verwendung von URIs**
- Manipulation durch wohldefinierte Aktionen auf einer Repräsentation
  JA, durch **Verwendung der HTTP-Methoden GET und POST**
- alle Inhalte und Informationen zum Zustandsübergang (Hyperlinks) werden an den Client weitergegeben

Und bei Twitter?

REST-Constraints: Uniform interface

Standardisierter Zugriff auf Komponenten zur Vereinfachung der Infrastruktur und verbesserung der Sichtbarkeit von Interaktionen

- universelle Syntax zur Identifikation von Ressourcen
  Ja, durch **Verwendung von URIs**
- Manipulation durch wohldefinierte Aktionen auf einer Repräsentation
  JA, durch **Verwendung der HTTP-Methoden GET und POST**
- alle Inhalte und Informationen zum Zustandsübergang (Hyperlinks) werden an den Client weitergegeben
  Nicht unbedingt!

Und bei Twitter?

HTTP-Methoden

Im Allgemeinen sollten die HTTP-Methoden in einer REST-API folgende Bedeutungen haben:

Method	Collection	Element
GET	**List** members	**Retrieve** representation
PUT	**Replace** entire collection	**Replace or create** new member
POST	**Create** new member	**Create** new member of element treated as a new col
DELETE	**Delete** entire collection	**Delete** member of collection

Quelle: `https://en.wikipedia.org/wiki/Representational_State_Transfer#RESTful_web_services`

HTTP-Methoden

Im Allgemeinen sollten die HTTP-Methoden in einer REST-API folgende Bedeutungen haben:

Method	Collection	Element
GET	**List** members	**Retrieve** representation
PUT	**Replace** entire collection	**Replace or create** new member
POST	**Create** new member	**Create** new member of element treated as a new col
DELETE	**Delete** entire collection	**Delete** member of collection

Quelle: `https://en.wikipedia.org/wiki/Representational_State_Transfer#RESTful_web_services`

Und bei Twitter?

HTTP-Methoden

Im Allgemeinen sollten die HTTP-Methoden in einer REST-API folgende Bedeutungen haben:

Method	Collection	Element
GET	**List** members	**Retrieve** representation
PUT	**Replace** entire collection	**Replace or create** new member
POST	**Create** new member	**Create** new member of element treated as a new col
DELETE	**Delete** entire collection	**Delete** member of collection

Quelle: `https://en.wikipedia.org/wiki/Representational_State_Transfer#RESTful_web_services`

Und bei Twitter?

Nur GET und POST werden verwendet, Bedeutung jeweils der Dokumentation zu entnehmen.

Basic authentication

In jedem Aufruf Nutzername und Passwort mitsenden.

```
GET /private/index.html HTTP/1.1
Host: localhost
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

OAuth

Anfragen werden signiert und vom Besitzer einer Ressource authorisiert.

```
POST /1/statuses/update.json?include_entities=true HTTP/1.1
Accept: */*
Connection: close
User-Agent: OAuth gem v0.4.4
Content-Type: application/x-www-form-urlencoded
Authorization:
        OAuth oauth_consumer_key="xvz1evFS4wEEPTGEFPHBog",
              oauth_nonce="kYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS4cg",
              oauth_signature="tnnArxj06cWHq44gCs1OSKk%2FjLY%3D",
              oauth_signature_method="HMAC-SHA1",
              oauth_timestamp="1318622958",
              oauth_token="370773112-GmHxMAgYyLbNEtIKZeRNFsMKPR9EyMZeS9weJAEb",
              oauth_version="1.0"
Content-Length: 76
Host: api.twitter.com

status=Hello%20Ladies%20%2b%20Gentlemen%2c%20a%20signed%20OAuth%20request%21
```

https://dev.twitter.com/docs/auth/authorizing-request

XML vs. JSON

Vergleich zweier Anfragen an Twitter:

- `https://api.twitter.com/1/statuses/user_timeline.json?include_entities=true&include_rts=true&screen_name=MLuczak&count=10`
- `https://api.twitter.com/1/statuses/user_timeline.xml?include_entities=true&include_rts=true&screen_name=MLuczak&count=10`

XML vs. JSON

Vergleich zweier Anfragen an Twitter:

- `https://api.twitter.com/1/statuses/user_timeline.json?include_entities=true&include_rts=true&screen_name=MLuczak&count=10`
- `https://api.twitter.com/1/statuses/user_timeline.xml?include_entities=true&include_rts=true&screen_name=MLuczak&count=10`

	JSON	XML
Datenvolumen	22.0 kB	33.3 kB
Benutzbarkeit?		

6. Tutorium

Die Idee hinter Linked Data

Verknüpfe alle öffentlichen Informationsressourcen im Internet zu einer globalen Datenbank,

Die Idee hinter Linked Data

Verknüpfe alle öffentlichen Informationsressourcen im Internet zu einer globalen Datenbank, dem sogenannten **semantic web** oder auch **web of data** (im Gegensatz zum gegenwärtigen web of documents)

Die Idee hinter Linked Data

Verknüpfe alle öffentlichen Informationsressourcen im Internet zu einer globalen Datenbank, dem sogenannten **semantic web** oder auch **web of data** (im Gegensatz zum gegenwärtigen web of documents)
Diese Datenbank enthält Informationen über Dinge (Entitäten, Ressourcen) und ihre Beziehungen zueinander.

Linked Data Prinzipien

1. Dinge werden durch URIs identifiziert.

Linked Data Prinzipien

1. Dinge werden durch URIs identifiziert.
2. URIs sollen über HTTP dereferenzierbar sein.

Linked Data Prinzipien

1. Dinge werden durch URIs identifiziert.
2. URIs sollen über HTTP dereferenzierbar sein.
3. Wenn nach einem Ding gefragt wird, gib sinnvolle Informationen darüber in einem standardisierten Format an.

Linked Data Prinzipien

1. Dinge werden durch URIs identifiziert.
2. URIs sollen über HTTP dereferenzierbar sein.
3. Wenn nach einem Ding gefragt wird, gib sinnvolle Informationen darüber in einem standardisierten Format an.
4. Wenn nach einem Ding gefragt wird, gibt seine Beziehung zu anderen Dingen an und verknüpfe diese.

Die Prinzipien 1. und 2. werden durch die uns bekannten URLs erreicht, z.B. `http://example.org/entity/some-thing`

Die Prinzipien 1. und 2. werden durch die uns bekannten URLs erreicht, z.B. `http://example.org/entity/some-thing`

Welches sind geeignete Formate?

Die Prinzipien 1. und 2. werden durch die uns bekannten URLs erreicht, z.B. `http://example.org/entity/some-thing`

Welches sind geeignete Formate?

RDF, **microdata**, **microformats**

Die Prinzipien 1. und 2. werden durch die uns bekannten URLs erreicht, z.B.
`http://example.org/entity/some-thing`

Welches sind geeignete Formate?

RDF, **microdata**, **microformats**
aber auch XML, HTML, JSON, YAML...

Linked Data Technologien

Die Prinzipien 1. und 2. werden durch die uns bekannten URLs erreicht, z.B. `http://example.org/entity/some-thing`

Welches sind geeignete Formate?

RDF, **microdata**, **microformats**
aber auch XML, HTML, JSON, YAML...

Und wie entscheiden?

Die Prinzipien 1. und 2. werden durch die uns bekannten URLs erreicht, z.B. `http://example.org/entity/some-thing`

Welches sind geeignete Formate?

RDF, **microdata**, **microformats**
aber auch XML, HTML, JSON, YAML...

Und wie entscheiden?

Frag den Interessenten: **Content negotiation**

RFD

RDF macht Aussagen über Ressourcen:

RFD

RDF macht Aussagen über Ressourcen:

`http://example.org/entity/some-thing`
is
`http://example.org/vocab/in-this-or-that-way`
related to
`http://example.org/entity/another-thing`

microdata und microformats

- http://webdesign.about.com/od/microdata/a/what-is-microdata.htm
- http://webdesign.about.com/od/microdata/a/what-are-microformats.htm

Content Negotiation

- Der Interessent wünscht sich ein Datenformat

```
GET / HTTP 1.0
Host: www.example.com
Accept: text/html
```

Content Negotiation

- Der Interessent wünscht sich ein Datenformat

```
GET / HTTP 1.0
Host: www.example.com
Accept: text/html
```

- Der Anbieter prüft die Verfügbarkeit und
  - leitet auf entsprechende Repräsentation weiter

```
HTTP/1.1 303 See Other
Location: http://example.org/other.html
```

Content Negotiation

- Der Interessent wünscht sich ein Datenformat

```
GET / HTTP 1.0
Host: www.example.com
Accept: text/html
```

- Der Anbieter prüft die Verfügbarkeit und
  - leitet auf entsprechende Repräsentation weiter

```
HTTP/1.1 303 See Other
Location: http://example.org/other.html
```

  - oder sendet im entsprechenden Format

Content Negotiation

- Der Interessent wünscht sich ein Datenformat

```
GET / HTTP 1.0
Host: www.example.com
Accept: text/html
```

- Der Anbieter prüft die Verfügbarkeit und
  - leitet auf entsprechende Repräsentation weiter

```
HTTP/1.1 303 See Other
Location: http://example.org/other.html
```

  - oder sendet im entsprechenden Format
  - oder sendet HTTP Status 406 „Not Acceptable“

XML-Technologien im Überblick

An der Tafel.

XML Grundlagen

Beachte besonders:

- Wohlgeformtheit
- XML-Deklaration
- Syntax
- Regeln für Namen
- ...

DTD

Anwendung Beschreibung der Struktur von XML-Dokumenten.

DTD

Anwendung Beschreibung der Struktur von XML-Dokumenten.

Prinzip Beschreibe Struktur von Elementen und ihren Attributen mittels einfacher regulärer Ausdrücke.

DTD

Anwendung Beschreibung der Struktur von XML-Dokumenten.

Prinzip Beschreibe Struktur von Elementen und ihren Attributen mittels einfacher regulärer Ausdrücke.

Markup

```
<!ELEMENT my-element (first-child,
    (second-child-either|second-child-or))>
<!ATTLIST my-element
    some-attribute CDATA #REQUIRED
    another-attribute PCDATA #IMPLIED
>
```

Schema

Anwendung Beschreibung der Struktur von XML-Dokumenten

Schema

Anwendung Beschreibung der Struktur von XML-Dokumenten

Prinzip Beschreibe Struktur von Elementen und ihren Attributen mittels komplexer XML-Strukturen

Schema

Anwendung Beschreibung der Struktur von XML-Dokumenten

Prinzip Beschreibe Struktur von Elementen und ihren Attributen mittels komplexer XML-Strukturen

Markup

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema
    xmlns:xsd="">
    <xsd:element name="my-element">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="first-child">
                <xsd:choice>
                    <xsd:element ref="second-child-eit
                    <xsd:element ref="second-child-or"
                </xsd:choice>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Schema - restriction und extension

restriction Einschränkung eines simpleType oder eines complexType

```
<xsd:simpleType>
    <xsd:restriction base="positiveInteger">
        <xsd:maxExclusive value="10" />
    </xsd:restriction>
</xsd:simpleType>
```

Siehe auch http://w3schools.com/schema/schema_facets.asp.

Schema - restriction und extension

restriction Einschränkung eines simpleType oder eines complexType

```
<xsd:simpleType>
    <xsd:restriction base="positiveInteger">
        <xsd:maxExclusive value="10" />
    </xsd:restriction>
</xsd:simpleType>
```

Siehe auch http://w3schools.com/schema/schema_facets.asp.

extension Erweiterung eines complexType

```
<xsd:complexType>
    <xsd:extension base="my-other-type">
        <xsd:element ref="new-element" />
        <xsd:attribute ref="new-attribute" />
    </xsd:extension>
</xsd:complexType>
```

DTD vs. Schema

DTDs und XML Schemas haben die gleiche Anwendung, aber

- DTD ist einfach und kann Entitäten deklarieren.
- XML Schema hat viel mehr Funktionen und ist dadurch komplexer.

XSL

Anwendung Transformation von XML-Dokumenten in beliebige Darstellung (anderes XML, SVG, HTML, JSON, YAML, . . . oder auch unstrukturierter Text).

XSL

Anwendung Transformation von XML-Dokumenten in beliebige Darstellung (anderes XML, SVG, HTML, JSON, YAML, ... oder auch unstrukturierter Text).

Prinzip Templates matchen (XPath) auf Teile des Dokuments und erzeugen Ausgabe.

XSL

Anwendung Transformation von XML-Dokumenten in beliebige Darstellung (anderes XML, SVG, HTML, JSON, YAML, ... oder auch unstrukturierter Text).

Prinzip Templates matchen (XPath) auf Teile des Dokuments und erzeugen Ausgabe.

Markup

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:transform version="1.0"
    xmlns:xsl="">
    <xsl:template match="@*|node()" />
</xsl:transform>
```

XPath

Anwendung Auswahl bestimmter Teile eines XML-Dokumentes.

XPath

Anwendung Auswahl bestimmter Teile eines XML-Dokumentes.

Prinzip Finde Knoten durch Pattern-Matching und Filter.

XPath

Anwendung Auswahl bestimmter Teile eines XML-Dokumentes.

Prinzip Finde Knoten durch Pattern-Matching und Filter.

Beispiel

```
child::section[position()<6]
/descendant::cite
/attribute::href
```

XPath - Achsen

`http://cs.au.dk/~amoeller/XML/linking/axes.html`

Übungen

XML Grundlagen `http://w3schools.com/quiztest/quiztest.asp?qtest=XML`

XPath `http://www.andong.co.uk/programming/quiz.aspx?id=xslt&action=start`

Fragen? Jetzt stellen!

Fragen oder Hinweise auf Fehler in den Folien gerne auch per E-Mail mit Betreff „[xml] . . . “ an niels.hoppe@fu-berlin.de.