XML (eXtensible Markup Language) und XPath

Dozent: Prof. Dr. Michael Eichberg

Kontakt: michael.eichberg@dhbw.de, Raum 149B

Version: 1.0.2

Folien: https://delors.github.io/ds-xml/folien.de.rst.html
https://delors.github.io/ds-xml/folien.de.rst.html.pdf

Fehler melden: https://github.com/Delors/delors.github.io/issues

1. XML (eXtensible Markup Language)

Markup Languages

- Sprachen, die verwendete werden, um Texte zu strukturieren und zu formatieren
- maschinenlesbar
- Beispiele:
  - HTML
  - XML
  - LaTeX
  - Markdown
  - reStructuredText
  - ...

Auch wenn Markup-Sprachen für Menschen lesbar sind, sind sie in erster Linie für Maschinen gedacht. Darüber hinaus sollte im Allgemeinen vermieden werden, dass der Markup dem Formatieren dient/zum formatieren verwendet wird.

YAML hat keinen Dokumentenfokus und ist nicht (mehr) als Markup-Sprache klassifiziert.

XML - Hintergrund

- Aufbauend auf **Standard Generalized Markup Language (SGML)**
  - SGML ist Standardisiert als ISO 8879:1986
  - In SGML ist die Basis für jedes Dokument eine Formatbeschreibung mit Hilfe einer *Document type definition* (DTD)

    Beschreibt welche Elemente es gibt und wie diese ineinander geschachtelt werden können

```
<!ELEMENT note (head, body)>
<!ELEMENT head (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

- XML ist eine vereinfachte Version von SGML und wurde 1998 standardisiert.
- XML dient der Kodierung und Strukturierung einzelner Instanzen von Dokumenten.

XML[1]

- Ein XML Dokument kann man sich als einen Baum von Elementen vorstellen, die Informationen enthalten.
- Dokumentenstruktur kann durch DTDs oder XML-Schemas beschrieben werden.
- Eine explizite Beschreibung der Dokumentenstruktur ist nicht zwingend erforderlich (aber häufig sinnvoll).
- XML Dokumente müssen stringente Anforderungen an die Syntax erfüllen (*Well-formed XML Dokumente*).
- XML bildet die Basis für viele weitere Sprachen wie MathML, GraphML, SVG, ...
- Abfragen auf XML basierenden Dokumenten können mittels XPath oder XQuery durchgeführt werden.
- Auf XML basierende Dokumenten können durch XSLT transformiert werden.

In Hinblick auf XML betrachten wir Dokumente als Instanzen von Informationen, die eine Struktur haben. Unter dieser Perspektive ist vieles ein Dokument:

- Artikel, Bücher, Notizen, Gedichte, Romane
- Technische Handbücher, Beiblätter, Produktverpackungen
- Mails, Nachrichten
- Rechnungen, Bestellungen, Lieferscheine
- Log Dateien, Protokolle, Konfigurationsdateien

Wesentliche Anforderungen bzgl. der Syntax eines XML Dokuments (*Well-formed* XML Dokumente):

- es gibt nur ein Wurzelelement
- Element überlappen sich nicht; d. h. für alle Elemente (außer dem Wurzelelement) gilt: Befindet sich das Start-Tag im Inhalt eines anderen Elements, so befindet sich das End-Tag im Inhalt desselben Elements. Es ergibt sich somit ein Baum.

---

[1] XML 1.0: eXtensible Markup Language, https://www.w3.org/TR/xml/ (Aktuell)
XML 1.1: https://www.w3.org/TR/2006/REC-xml11-20060816/ (nur für Spezialfälle)

Was bietet XML?

- Internationalisierung durch die Verwendung von Unicode.
- Validierung von Instanzen (d. h. von Dokumenten).
- Lokalisierung von Namen über Namensräume (z. B. *Mein* Haus ist nicht dein *Haus*).
- Ein *menschenlesbares* Format.
- Hierarchische Struktur.
- Erweiterbarkeit.

Wie auch in HTML (HyperText Markup Language) kann auch in XML jedes Zeichen als Referenz auf ein Unicode-Zeichen kodiert werden.

Beispiel

```
&#x2200;&#x03b1;&#x2208;&#x0393;
```

entspricht:
∀α∈Γ

XML Dokument - Beispiel

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<lehrveranstaltungen status="akkreditiert">
    <!-- Modul muss überarbeitet werden... -->
    <modul>
        <vorlesung>Web Entwicklung</vorlesung>
        <vorlesung>Verteilte Systeme</vorlesung>
    </modul>
</lehrveranstaltungen>
```

XML-Deklaration: `<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`

Start-Tags: `<lehrveranstaltungen>`, `<modul>`, `<vorlesung>`

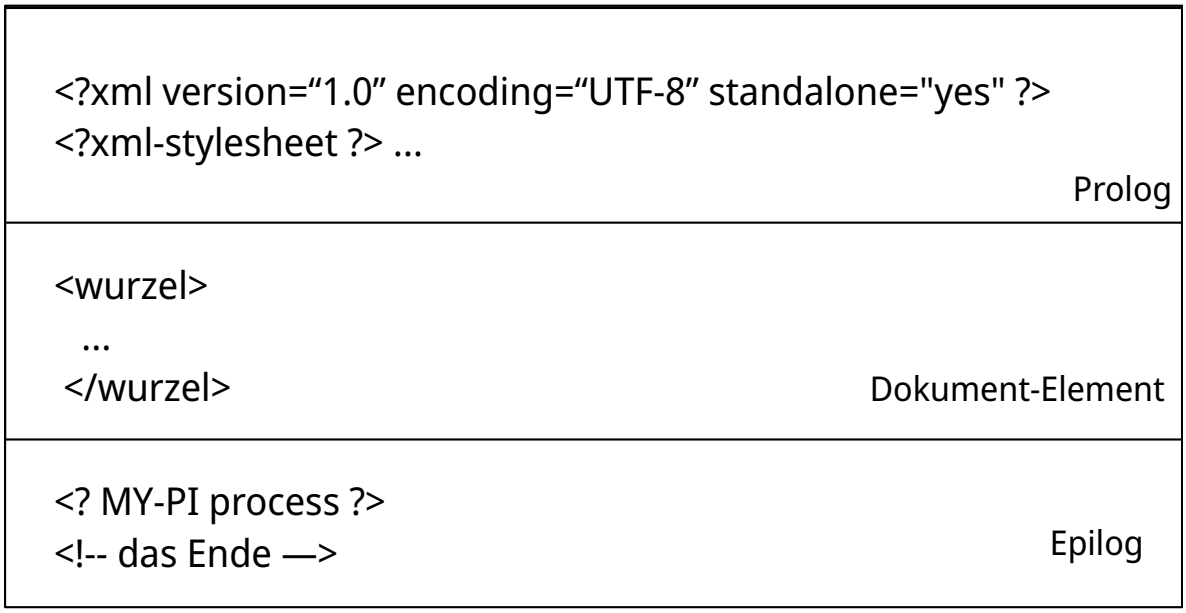
End-Tags: `</lehrveranstaltungen>`, `</modul>`, `</vorlesung>`

Attribute: `status`

#Text Nodes: `Web Entwicklung`, `Verteilte Systeme`

Die Spezifikationen bzgl. `encoding` (Kodierung des Dokuments) und `standalone` (Ist das Dokument von anderen Dokumenten abhängig) sind *nur* Pseudoattribute, da sie zum Prolog des Dokuments gehören.

XML Dokument - allgemeine Struktur



Formale Beschreibung der XML Syntax

- die Syntax von XML Dokumenten wird durch eine *formale Grammatik* (hier: EBNF) beschrieben.

  Beispiel - Beschreibung des Prologs von XML Dokumenten in EBNF:

```
1 prolog      ::= XMLDecl? Misc* (doctypedecl Misc*)?
2 XMLDecl     ::= "<?xml" VersionInfo EncodingDecl? SDDecl? S? "?>""
3 VersionInfo ::= S "version" Eq ("'" VersionNum "'" | '"' VersionNum '"')
4 Eq          ::= S? "=" S?
5 VersionNum  ::= "1." [0-9]+
6 Misc        ::= Comment | PI | S
```

  Wir werden uns auf eine informelle Beschreibung der Syntax der wichtigsten Konstrukte beschränken.

EBNF (*Extended Backus-Naur Form*) 101:

- '+' bedeutet 'eins oder mehr',
- '?' bedeutet 'optional'
- '*' bedeutet 'null oder mehr'.
- Klammerkonstrukte werden gruppiert.
- '|' (Pipe-Zeichen) bedeutet 'oder'.
- 'S' steht für Leerzeichen (hier).
- 'string' bedeutet das Vorkommen der wörtlichen Zeichenkette.
- [c-c] ist eine Zeichenklasse und steht für ein einzelnes Zeichen im angegebenen Bereich.

EBNFs sind eng mit regulären Ausdrücke verwandt. EBNFs können jedoch auch rekursive Strukturen beschreiben und werden häufig für die Beschreibung von Programmiersprachen verwendet.

Elemente

- Im Allgemeinen bestehen Elemente aus einem Start-Tag (z. B. `<start>`), seinem Inhalt und einem End-Tag (z. B. `</start>`).
- Der Inhalt eines Elements ist geordnet.
- Start-Tags können Attribute haben - Name/Wert-Paare (z. B. `<start kind="slow"/>`).
- Die Elemente müssen wohlgeformt sein: balanciert, konforme Syntax, gültige Attribute, keine Duplikate, usw.
- Elemente können leer sein (z. B. `<empty/>`); d. h. sie haben keinen Inhalt, können aber Attribute haben.

Attribute

- Attribute sind *ungeordnete* Name/Wert-Paare, die in einem Start-Tag eines Elements enthalten sind.
- Jedes Attribut darf nur einmal in einem Element vorkommen.
- Ausgewählte Zeichen müssen maskiert werden, wenn sie im Wert vorkommen sollen.
- Die Werte von Attributen werden normalisiert (z. B. werden Zeilenumbrüche entfernt).

Vordefinierte *Entity References*

Entity Reference	**Zeichen**
`<`	<
`>`	>
`&`	&
`&quot;`	"
`&apos;`	'

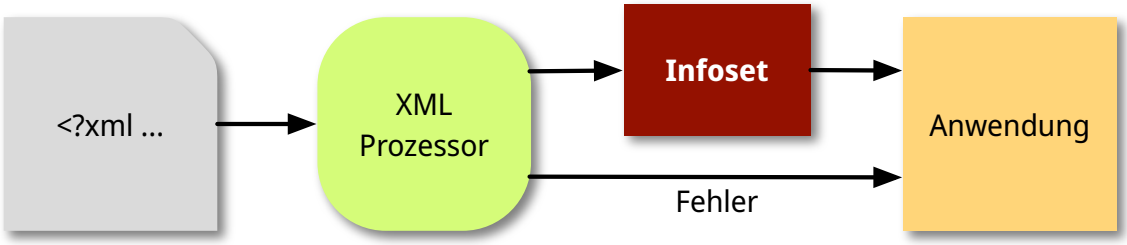
Whitespace in XML

- Oft wird Leerraum (Leerzeichen, Zeilenumbrüche, Tabulatoren usw.) hinzugefügt, um das XML "lesbarer" zu machen.
- Leerzeichen können als nicht signifikant gekennzeichnet werden; dies erfordert jedoch einen validierenden XML Prozessor.

XML für Anwendungen - *Infosets*

Infosets (Information Sets)

- Ein *Infoset* ist eine (abstrakte) Darstellung eines XML Dokuments; losgelöst von der konkreten Syntax (z. B. ob der Wert eines Attributs in `" "` oder `' '` gefasst wurde; oder ob *Entity References* verwendet wurden, etc.).
- Ein *Infoset* enthält alle Informationen, die in einem XML Dokument enthalten sind.



Ein Infoset ist eine Hierarchie (oder ein Baum) von Elementen mit benannten Eigenschaften.

Ausgewählte *Info Items*

Die verschiedenen *Info Items* eines *Infosets* stellen z. B. die folgenden Informationen bereit:

Document Info Item:
: Kinder, Wurzelelement, Basis-URI.

Element Info Item: lokaler Name, Kinder, Attribute, Vorgänger

Attribute Info Item: lokaler Name, normalisierter Wert, deklarierendes Element

Es gibt weitere *Info Items* für Kommentare, Verarbeitungsanweisungen, Text, etc.

2. XML Namensräume (🇺🇸 *XML Namespaces*)

Namensräume in XML - Motivation

Wenn wir nur einen Namen(sraum) haben sollten...

- Was würde passieren, wenn wir Markup von zwei verschiedenen Autoritäten nutzen wollten?
- Wie assoziiere ich Semantik mit gemischtem Markup?
- Wie verbinde ich ein Schema (oder Regeln) mit dem gemischten Markup?

Variante 1:

```
<date>1/27</date>
```

Variante 2:

```
<date><year>2004</year><day>1</day><month>27</month></date>
```

Wie kann ich beide unterscheiden?

XML - Namen und Namensräume

Namen werden in zwei Teile unterteilt:

`Präfix`: Ein Bezeichner für einen Namensraum.

`lokaler Name`: Ein Bezeichner für einen Namen in diesem Namensraum

Diese Teile werden durch einen Doppelpunkt getrennt und **QNames** (*Qualified Names*) genannt.

Beispiel:

```
<c:pseudocode>
  <c:comment xlink:href="http://somewhere..."/>
</c:pseudocode>
```

Dies gilt nur für Element- und Attributnamen.

Jedes Präfix, das "xml" enthält, ist für das W3C reserviert.

XML Präfixe und Namensräume

- Präfixe müssen durch assoziierte Präfixe mit Namensräumen deklariert werden, *bevor* sie verwendet werden.
- Diese Assoziation kann nur für Elemente deklariert werden.
- Die Syntax lautet: `xmlns:prefix="some:uri"`.

  Beispiel:

```
<c:pseudocode xmlns:c="urn:publicid:IDN+mathdoc.org">
  <c:comment xlink:href="http://somewhere..."
             xmlns:xlink="http://www.w3.org/..."/>
</c:pseudocode>
```

- *Bevor* bedeutet, dass der Präfix auf dem Element, in dem das Präfix vorkommt - oder auf einem Vorgängerelement - deklariert werden muss.

Das Präfix `xml` ist vordefiniert und die URI ist: `http://www.w3.org/XML/1998/namespace`.

Mit Hilfe einer URI (Uniform Resource Identifier) wird ein Namensraum identifiziert. Die URI muss nicht aufgelöst werden können.

URI-Werte können Webadressen sein (z. B. `http://youdomain.com`), aber auch andere Werte wie URNs (Namen): `urn:...` oder andere Schemata: `scheme:scheme-specific-part`.

Default Namespace

- Der Standardnamensraum kann vorgegeben werden.
- Dies gilt nur für Elementnamen ohne Präfixe.
- Die Syntax lautet: `xmlns="some:uri"`.

Beispiel

```
<c:pseudocode xmlns:c="urn:publicid:IDN+mathdoc.org">
<c:comment xmlns="http://www.w3.org/1999/xhtml">
    <p>Dieser Code macht folgendes:</p>
    ...
</c:comment>
</c:pseudocode>
```

Mit `xmlns=""` kann der gesetzte Standardnamensraum aufgehoben werden.

Achtung!

Attribute ohne Präfix befinden sich immer im leeren Namensraum, d. h. sie haben keinen Namensraum

Geltungsbereich von Namensräumen[2]

- Der Geltungsbereich einer Deklaration eines Namensraums ist das Element, in dem sie vorkommt.
- Es gibt keinen Unterschied zwischen Deklarationen auf dem Wurzelelement und anderswo.
- Das Element, seine Attribute und seine Kinder können dieses Präfix in ihren Namen verwenden.
- Namespaces können redefiniert werden.

---

[2] **eng:** *Namespace Scoping*

Der Name des Namensraums

- Das Präfix ist nur eine Abkürzung des eigentlichen Namens des Namensraumes (d. h. des Wertes der Deklaration).
- Ein Name besteht nun aus zwei Teilen:
  1. der Name des Namensraum, der mit dem Präfix verbunden ist.
  2. der lokale Name; d. h. der Teil des Namens nach dem Doppelpunkt.

Namensräume und das XML Information Set (Infoset)

Elemente

Name des Namensraums:
: der Name des Namensraums oder `no value`, wenn es keinen gibt.

Lokaler Name:
: der lokale Teil des Namens (d. h. nach dem Doppelpunkt).

Präfix:
: der für das Element verwendete Namensraumpräfix oder `no value`, wenn es keinen gibt.

Im Geltungsbereich definierte Namensräume:
: Eine ungeordnete Liste von *Namespace Info Items*.

Deklarationen von Namensräumen:
: Eine ungeordnete Liste aller Attribute des Elements, die Namensräume deklarieren.

Attribute

Name des Namensraums:
: der Name des Namensraums oder `no value`, wenn es keinen gibt.

Lokaler Name:
: der lokale Teil des Namens (d. h. nach dem Doppelpunkt).

Präfix:
: der für das Attribut verwendete Namensraumpräfix oder `no value`, wenn es keinen gibt.

Namensräume

Setzen des Standardnamensraums

```
<pseudocode xmlns="urn:publicid:IDN+mathdoc.org">
    <comment>e = mc^2</comment>
</pseudocode>
```

Definition eines Präfixes (hier: „m“)

```
<m:pseudocode xmlns:m="urn:publicid:IDN+mathdoc.org">
    <m:comment>e = mc^2</m:comment>
</m:pseudocode>
```

Redefinition eines Präfixes (hier: „m“)

```
<m:pseudocode xmlns:m="urn:publicid:IDN+mathdoc.org">
    <m:comment xmlns:m="urn:comment">e = mc^2</m:comment>
</m:pseudocode>
```

Beispiel: OpenOffice Dokumentenformat

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content
    [...]
        xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
        xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
        xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
        office:version="1.3">
    <office:scripts/>
    <office:font-face-decls>[...]</office:font-face-decls>
    <office:automatic-styles>
        <style:style style:name="P1" [...]>[...]</style:style>
    </office:automatic-styles>
    <office:body>
        <office:text> [...]
            <text:p text:style-name="P1">Test</text:p>
        </office:text>
    </office:body>
</office:document-content>
```

Übung: XML Dokument mit Namensräumen

Erstellen Sie ein XML Dokument nach folgenden Vorgaben:

- Das Wurzelelement ist `document`.
- Das Dokument fasst mehrere Bestellungen (`order`-Elemente) zusammen.
- Es gibt vier Bestellungen (d.h. vier `order`-Elemente).
- Jede Bestellung enthält mehrere Produkte (d. h. `product`-Elemente).
- Pro Produkt soll angegeben werden um welches Produkt es sich handelt und wie viele davon bestellt wurden. Fügen Sie den Bestellungen zwischen einem und drei Produkte hinzu.
- Die Bestellungen gehen an verschiedenen Partnersysteme und sollen deswegen durch entsprechende Namensräume voneinander getrennt sein.

3. XPath

XPath - Übersicht

- XPath ist eine Syntax/Sprache zur Adressierung von Knoten in einem Dokument.
- XPath-Ausdrücke sind *Pfadausdrücke* ( *path expressions*).
- Erlaubt es folgende Dinge auszudrücken:
  - Selektiere alle `vorlesung`-Kinderelemente des `lehrveranstaltungselements`-Elements.
  - Finde die Geschwisterknoten des Elements `vorlesung`.
  - Finde das Element `lehrveranstaltung`, bei dem das Attribut `status` den Wert `aufgekündigt` hat.
- Es handelt sich um einen eigenen Mini-Standard, der von vielen Spezifikationen verwendet wird (XSLT, XQuery, ...).
- Implementationen sind in vielen Programmiersprachen verfügbar (z. B. Java, JavaScript, Python, ...) und alle Browser unterstützen XPath-Ausdrücke für die Selektion von Elementen.

XPath - Pfadausdrücke

- Ein Pfadausdruck besteht aus einer Folge von Schritten, die durch Schrägstriche getrennt sind. (Ähnlich wie bei Dateipfaden.)
- Ein einzelner Schrägstrich ("`/`") steht für das Wurzelelement.
- Nachfolgende benannte Schritte im Pfad stellen Kinder dar:

  ```
  /lehrveranstaltungen/modul
  ```

  Wählt das untergeordnete Element `modul` des Dokumentenelements `lehrveranstaltungen` aus.
- XPath-Ausdrücke müssen nicht bei der Wurzel starten:

  ```
  modul/vorlesung
  ```

  Wählt das `vorlesung`-Kinderelement des `modul`-Elements aus.

Resultat eines XPath-Ausdrucks

- Das Ergebnis der Auswertung eines XPath-Ausdrucks ist ein *Node Set* oder ein einzelner Wert (ein String, eine Zahl oder ein Boolean).[3]
- Ein `Node` ist nur ein anderer Begriff für *Info Item*.
- Beispiel

  Sei das folgende XML-Dokument gegeben:

```
<modul>
    <vorlesung>Eins</vorlesung>
    <vorlesung>Zwei</vorlesung>
</modul>
```

  Dann gibt der folgende Ausdruck zwei `vorlesung`-Elemente zurück:

```
/modul/vorlesung
```

[3] Die Reihenfolge der Ergebnisse muss nicht über alle Implementierungen (z. B. Browser) hinweg konsistent sein. (vgl. XPathResult)

Attribute Selektieren

- Attribute können über den entsprechenden Schritt: `@Name` ausgewählt werden.
- Beispiel

  Sei das folgende XML-Dokument gegeben:

  ```
  <modul>
      <vorlesung mhb="123">Eins</vorlesung>
      <vorlesung mhb="456">Zwei</vorlesung>
  </modul>
  ```

  Dann würde der Ausdruck:

  `/modul/vorlesung/@mhb`

  Die beiden `mhb` Attribute als Menge zurückgeben.

Namen und Namensräume

- Jeder Schritt eines XPath-Ausdrucks kann einen *QName* verwenden: `<Präfix>:<Lokaler Name>`
- Das Matching basiert auf dem lokalen Namen und dem Namen des Namespaces und nicht auf dem Präfix.
- Beispiele für XPath-Ausdrücke mit Namensraum:

```
/dhbw:modul/dhbw:vorlesung
/dhbw:modul/dhbw:vorlesung/@mhb
/dhbw:modul/dhbw:vorlesung/@i:mhb
```

Hinweis

Die Präfixbindung wird außerhalb des Ausdrucks definiert (i. d. R. anwendungsspezifisch).

In dem gezeigten Beispiel müsste die Anwendung die Präfixe (`dhbw` und `i`) mit den entsprechenden Namensräumen verknüpfen.

kein Präfix = kein Namensraum

Ein Namenstest innerhalb eines Pfadausdrucks, der kein Präfix spezifiziert ist nur für Namen ohne Namensraum erfolgreich!

Zum Beispiel:

```
m:section/title
```

selektiert das Element `title` im folgenden Beispiel, da es keinen Namensraum hat:

```
<m:section xmlns:m='urn:...'>
  <title>Kein Namespace</title>
</m:section>
```

in folgendem Beispiel jedoch nicht:

```
<m:section xmlns:m='urn:...'
           xmlns='urn:something-else...'>
  <title>Ich habe einen Namensraum...</title>
</m:section>
```

Der Namensabgleich basiert auf dem lokalen Namen und dem Namen des Namensraums.

Wildcards in xPath

- `*` wird als Platzhalter für Namen verwendet werden.
- Beispiele:
  - Alle Elemente, die in einem `modul`-Element enthalten sind:

    ```
    /modul/*
    ```
  - Alle Attribute eines `vorlesung`-Elements:

    ```
    /modul/vorlesung/@*
    ```
  - Verwendung von Namensräumen:

    ```
    /dhbw:modul/dhbw:*
    /dhbw:modul/dhbw:vorlesung/@i:*
    ```

Der Namensraum Präfix kann nicht durch ein *Wildcard* ersetzt werden.

Kontextknoten

- Die Auswertung erfolgt immer in Bezug auf einen Kontextknoten.
- Der Kontextknoten wird mit . (Punkt) referenziert.
- Beispiel - Selektion der Attribute des Kontextknotens:

  ```
  ./@*
  ```

  Der Kontextknoten ist implizit.
- Der Kontextknoten muss nicht zwingend ein Element sein.

Bedingtes Matching

- Prädikate erlauben die Angaben von Bedingungen und folgen der Deklaration des *Schrittes*.
- Prädikate sind in eckigen Klammern (`[` und `]`) eingeschlossen.
- Verschachtelte Prädikate sind möglich.
- **Beispiel**

  ```
  /modul/vorlesung[@mhb='123']
  ```

  Wählt das `vorlesung`-Element aus, das das Attribut `mhb` mit dem Wert `123` hat.
- Die Verwendung von (komplexen) Pfadausdrücken in Bedingungen ist ebenfalls möglich.

  Beispiel

  ```
  lehrveranstaltungen/modul[vorlesung/@mhb='123']
  ```

Bedingtes Matching - Operatoren und Funktionen

- boolesche Operatoren: (`or` und `and`)
- boolesche Funktionen: `not ( boolean ),lang ( string ),true(),false(),...`
- Mathematische Funktionen: `sum( node-set ),number( object ),...`
- Zeichenketten: `string( object ),concat( string, string, string* ),starts-with( string, string ),contains( string, string ),substring( string, number, number ),string-length( string ),normalize-space( string ),...`
- Node-set Funktionen: `last(),position(),count( node-set ),id( object ),local-name( node-set ),namespace-uri( node-set ),...`

Beispiel

Alle Element, die den lokalen Namen `modul` haben:

```
//*[local-name()='modul']
```

vgl. XPath 1.0 Funktionen

Selektion von Elternknoten und Vorgängerknoten

- Über den Kontextknoten kann auf übergeordnete und vorgelagerte Elemente zugegriffen werden.
- `..` steht für das übergeordnete Element; wie bei Verzeichnissen.

Beispiel

```
/modul/vorlesung[@mhb='123']/..
```

Wählt das `modul`-Element aus, das das `vorlesung`-Element mit dem Attribut `mhb` und dem Wert `123` enthält.

Selektion von Kindknoten

- mit dem `//` können Elemente, die keine direkten Kinder sind abgeglichen werden

  Es werden somit die Nachkommen des *aktuellen Kontexts* durchsucht.

Beispiel

```
lehrveranstaltungen//vorlesung[@mhb='123']/..
```

Wählt alle `vorlesung`-Elemente mit dem Attribut `mhb` und dem Wert `123`, die Nachkommen des `lehrveranstaltungen`-Elements sind aus.

Auswahl von Knoten, die keine Elemente oder Attribute sind

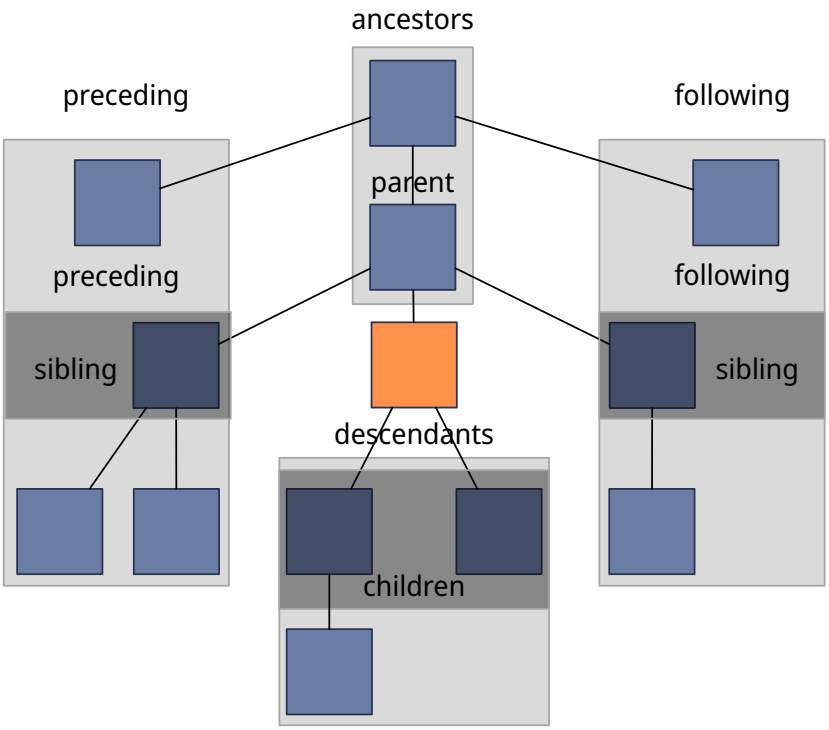
Funktion	Beschreibung
`text()`	Wählt den Textinhalt eines Elements aus.
`comment()`	Wählt Kommentare aus.
`processing-instruction()`	Wählt Verarbeitungsanweisungen aus.
`node()`	Wählt alle Knoten aus.

Beispiel

Alle Kommentare, die Kinder des `document`-Elements sind:

```
/document/comment()
```

Beziehungen zwischen Knoten



Weitere Beziehungen

Attribute:
Jedes Element kann Attribute haben (welche keine Kinder im Baum sind).

Namensraum:
Jedes Element kann Namensräume haben (welche keine Kinder bzgl. des Baums sind).

Axen in XPath beschreiben die Richtungen von Beziehungen zwischen Knoten.

- Baumbeziehungen:
  - `ancestor`, `ancestor-or-self`
  - `parent`, `child`, `self`
  - `descendant`, `descendant-or-self`
  - `following`, `following-sibling`
  - `preceding`, `preceding-sibling`
- Weitere Beziehungen:
  - *Attribute*
  - *Namensräume*

- **Beispiel**

  `//modul/ancestor::lehrveranstaltungen`

  Wählt das `lehrveranstaltungen`-Element aus, das das `modul`-Element enthält.

- **Beispiel**

  `//modul/child::vorlesung`

  Wählt das `vorlesung`-Element aus, das ein Kind des `modul`-Elements ist.

Alle gängigen Browser unterstützen XPath 1.0.

Gängige Bibliotheken (z. B. Saxon) unterstützen XPath 3.1.

https://www.saxonica.com/welcome/welcome.xml

Übung: XPath

3.1. Erste XPath Ausdrücke

Schreiben Sie XPath-Ausdrücke, um die folgenden Anfragen zu beantworten:

- Wählen Sie alle `orders`-Elemente aus.
- Wählen Sie alle `product`-Elemente aus, die im Namensraum `http://fruits.com` sind.
- Berechnen Sie die Summe der Werte der `quantity`-Attribute, die zu Bestellungen aus dem Namensraum von `http://fruits.com` gehören.
- Berechnen Sie die Summe der Werte der `quantity`-Attribute, die im Namensraum `http://meat.com` sind.
- Berechnen Sie die Summe der Werte *aller* `quantity`-Attribute; unabhängig von dem konkreten Ziel der Bestellung.
- Wählen Sie alle `order`-Elemente aus.
- Wählen Sie das erste `product`-Element jeder Bestellung aus, die `http://meat.com` zugeordnet ist.
- Wählen Sie alle `order`-Elemente aus, bei denen mehr als fünf Produkte bestellt wurden.
- Bestimmen Sie wie viele Bestellungen es gibt.
- Selektieren Sie alle Produkte der Bestellungen, die genau vier Produkte umfassen.

Hinweis

Verwenden Sie das XML Dokument aus der Musterlösung zur letzten Aufgabe als Grundlage.

Voraussetzungen

Installation von node.js

Installieren Sie node.js von https://nodejs.org/en (Version 21 und 22 sind getestet). Benutzen Sie bitte *eine getestet Version*!

Installieren Sie die benötigten node.js Pakete

Am Besten einfach im "aktuellen Verzeichnis" in dem die Übungsdateien liegen ausführen:

```
npm install jsonschema@~1.4.1
npm install xpath@~0.0.34
npm install @xmldom/xmldom@~0.8.10 #ACHTUNG: 0.9.0 funktioniert nicht!
```

Ausführen der XPath Ausdrücke

Nutzen Sie den XPath Evaluator, um die XPath-Ausdrücke auf dem XML-Dokument auszuführen:

https://gist.github.com/Delors/189629b86265463e4a625924a9f705c8

(Speichern Sie das Script in der Datei `xpaths_evaluator.js` und führen Sie es mit `node xpaths_evaluator.js <xpath specifications>` aus.)

In der Datei finden Sie am Anfang eine Beschreibung wie die Dateien auszusehen haben. Alternativ können Sie auch die XML Datei demo.xml und die Datei demo.xpaths.json herunterladen und als Grundlage nutzen.