Kapitel 6

XML - Extensible Markup Language

[1]Die Extensible Markup Language (engl. „erweiterbare Auszeichnungssprache“), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird u. a. für den plattform- und implementationsunabhängigen Austausch von Daten zwischen Computersystemen eingesetzt, insbesondere über das Internet.

Die vom World Wide Web Consortium (W3C) herausgegebene XML-Spezifikation (Recommendation, erste Ausgabe vom 10. Februar 1998) definiert eine Metasprache, auf deren Basis durch strukturelle und inhaltliche Einschränkungen anwendungsspezifische Sprachen definiert werden. Diese Einschränkungen werden durch Schemasprachen wie DTD oder XML Schema ausgedrückt. Beispiele für XML-Sprachen sind: RSS, MathML, GraphML, XHTML, XAML, Scalable Vector Graphics (SVG), GPX, aber auch XML-Schema.

Ein XML-Dokument besteht aus Textzeichen, im einfachsten Fall in ASCII-Kodierung, und ist damit menschenlesbar. Binärdaten enthält es per Definition nicht.

6.1 Einführung - Theorie

XML ist eine textbasierte Meta-Auszeichnungssprache, die die Beschreibung, den Austausch, die Darstellung und die Manipulation von strukturierten Daten erlaubt, so dass diese, vor allem über das Internet, von einer Vielzahl von Anwendungen genutzt werden können. XML ist keine Markup-Sprache in dem Sinn, dass die Menge der Auszeichnungselemente (tags) festgelegt ist, wie z.B. bei HTML. Die Namen der Tags können von XML-AnwenderInnen frei erfunden werden. Die tag-Namen können so gewählt werden, dass sie die Bedeutung des Inhalts ausdrücken. Ein tag <`buchtitel`> erklärt sich selbst.

XML gilt als Vereinfachung der Standard Generalized Markup Language (SGML). Die Verwendung von XML betrifft zwei große Bereiche. Als erstes lässt sich XML für die Entwicklung von unterschiedlichen Beschreibungssprachen einsetzen (siehe erster Absatz). Der zweite Anwendungsbereich liegt in der Definition von Austauschformaten zwischen verschiedenen Anwendungen in heterogenen Netzen. Hierzu dienen die Standards SOAP[2] und XML-RPCs.

6.2 Begriffe und Definitionen

- Markups
  Markups sind Informationen, die einem Dokument hinzugefügt werden, um auf bestimmte Weise dessen Bedeutungsinhalt zu erweitern, indem es die einzelnen Teile kennzeichnet und festlegt, wie diese zueinander in Beziehung stehen.

[1]Quelle: http://www.wikipedia.org
[2]Simple Object Access Protocol

- DTD und XML Schema
  Neben der Einhaltung der Syntaxvorschriften der Markup-Sprache selbst kann man verlangen, dass Dokumente eigendefinierten Strukturregeln genügen. Dazu wurde in XML zuerst die sog. Document Type Definition (DTD) von SGML übernommen.
- XSL
  Die Extensible Stylesheet Language (XSL) besteht aus drei Hauptteilen:
  - XSL Transformation Language dient zur Definition von Transformationen von XML-Dokumenten in andere XML-basierte Sprachen.
  - Formating Objects bildet eine ausgereifte Formatierungsprache, die in ihrer Funktion etwa PDF ähnelt.
  - XPath ist eine von XSLT verwendete Sprache für die Adressierung von Elementen innerhalb eines XML-Dokuments.
- DOM
  Das Document Object Model (DOM) ist eine baumartige interne Darstellung eines XML-Dokuments (oder auch anderer Dokumentarten). Mithilfe der DOM-Schnittstelle können Anwendungen XML-Dokumente dynamisch verändern und haben einen wahlfreien, nicht sequentiellen Zugriff auf den Inhalt eines Dokuments.
- XML-Parser
  Ein XML-Parser ist ein Programm, das XML-Dokumente auf syntaktische Korrektheit überprüft. Ein sog. validierender XML-Parser validiert zusätzlich ein XML-Dokument anhand einer DTD oder eines XML Schemas.

6.3 Beispiel

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE TeilnehmerS SYSTEM "teilnehmer0.dtd">
<TeilnehmerS vorlesung='Einführung in XML'>
       <Teilnehmer matrNr='4711'>
              <nachname>Kulator</nachname>
              <vorname>Karl</vorname>
              <semester>6</semester>
              <fachb>16</fachb>
       </Teilnehmer>
       <Teilnehmer matrNr="4722">
              <nachname>Ralwasser</nachname>
              <vorname>Mina</vorname>
              <semester>8</semester>
              <fachb>17</fachb>
       </Teilnehmer>
</TeilnehmerS>
```

- Die Anweisung <?xml version='1.0'?> in der ersten Zeile ist eine sog. XML-Deklaration (XML Processing Instruction). Sie teilt einem XML-Prozessor mit, dass dieses Dokument ein XML-Dokument ist. Zusätzlich wird die verwendete XML-Version 1.0 und der Zeichensatz ISO-8859-1 für westeuropäische Sprachen angegeben.
- Die Tags <TeilnehmerS> und <Teilnehmer> enthalten weitere Unterelemente. Sie dienen also der Strukturierung des Dokuments. Das nachname-Element dagegen enthält einen Text als Inhalt. In XML können Elemente auch gemischt Unterelemente und Texte als Inhalt haben, z. B. XML in der Praxis.

- Die Angaben `matrNr='...'` und `version='...'` sind sog. Attribute. Ein Element kann mehrere Attribute enthalten, die weitere Informationen über das Element angeben. Die Attributwerte sind in Hochkommata (einfache oder doppelte) eingeschlossen. In unserem Beispiel gibt z. B. das Attribut `matrNr` die Matrikelnummer eines Teilnehmers an.

6.3.1 Die XML-Deklaration

Die erste Zeile ist die sog. XML-Deklaration. Sie sollte in jedem XMLDokument enthalten sein.Sie kann die folgenden Attribute haben:

- `version`
  Spezifiziert die verwendete XML-Version im Dokument.
- `encoding`
  Definiert die im Dokument verwendete Zeichenkodierung. Einige verbreitete Zeichenkodierungen sind: US-ASCII, ISO-8859-1, UTF-8 und UTF-16. Falls encoding nicht angegeben wird, wird die Standard-Zeichenkodierung UTF-8 verwendet.
- `standalone`
  Sagt dem XML-Prozessor, ob irgendwelche andere Dateien geladen werden müssen. Falls externe Markup Declarations vorhanden sind, aber keine Standalone Document Declaration angegeben ist, wird `standalone=''no''` angenommen.

6.3.2 Die Dokumenttyp-Deklaration

Falls eine DTD (Document Type Definition) verwendet werden soll, so muss diese deklariert werden. Die Instruktion kann gegenwärtig folgende zwei Formen annehmen:

```
<!DOCTYPE root−element PUBLIC "name" "URL−of−DTD">
<!DOCTYPE root−element SYSTEM "URL−of−DTD">
```

oder innerhalb eines Dokuments:

```
<?xml version="1.0"?>
<!DOCTYPE root−element [!ELEMENT ...
...
]>
```

6.3.3 Elemente

Elemente sind die Grundbausteine von XML-Dokumenten. In XML werden Elemente durch Tags in spitzen Klammern in der Form <`element-name`> `...` </`element-name`> eingeklammert. Der Elementname kann eine beliebige Anzahl von Buchstaben, Zahlen, Bindestrichen, Punkten und Unterstrichen enthalten, darf aber nicht mit Punkt, Bindestrich oder einer Ziffer beginnen. Das Doppelpunktzeichen (:) wird als Trenner für Namensräume verwendet. Freizeichen, Tabulatoren, Zeilenumbrüche, Gleichheitzeichen und Anführungszeichen sind Separatoren und dürfen deshalb in Element- und Attributnamen nicht verwendet werden. Sonderzeichen wie `&` oder `?` dürfen ebenfalls nicht in Elementnamen enthalten sein.

6.3.3.1 Start- und Endtags

Ein Element besteht aus einem Start-Tag, einem Inhalt und einem End- Tag. Der Inhalt kann aus reinem Text bestehen, aus weiteren Unterelementen oder einer Mischung aus beidem. Die Ende-Markierung (End-Tag) wird durch </`elementName`> angegeben, und darf nicht, wie in HTML, weggelassen werden. Ist der Inhalt leer (leeres Element), so kann man Start- und End-Tag zu einem Tag zusammenziehen. Da das Kleinerzeichen (<) für den Beginn eines Tags steht, darf es nicht direkt im Text verwendet werden. Dazu können die vordefinierten Entities für < (`<`) und für > (`>`) verwendet werden.

6.3.4 Attribute

Mit Attributen kann man Elementen zusätzliche oder genauere Informationen hinzufügen. Man kann z. B. Elementen eindeutige Bezeichner durch ein Attribut geben, oder eine Eigenschaft über dieses Element beschreiben (Beispielsweise das `href`-Attribut für das `a`-Element in HTML). Eine Attributangabe besteht aus einem Attributnamen, einem Gleichheitszeichen und einem Wert in Anführungszeichen. Dabei können einfache oder doppelte Anführungszeichen verwendet werden.

6.3.5 Prozessor-Instruktionen

Prozessor-Instruktionen (PI, Processing Instructions) sind dazu gedacht, Anweisungen an externe Anwendungen in einem XML-Dokument einzubauen. Diese werden nur von der adressierten Anwendung verstanden und interpretiert. Sie werden von anderen Anwendungen ignoriert.

Eine PI sieht wie folgt aus:

```
<?anwendung attr1='Wert 1' ...?>
```

Beispiel ist der Hinweis für die Verwendung eines Stylesheets:

```
<?xml−stylesheet type="text/xsl" href="filename.xsl"?>
```

Dies teilt einem Stylesheet-Prozessor mit, das XML-Dokument anhand des Stylesheets `filename.xsl` zu transformieren.

6.3.6 Entities

Ähnlich wie bei Makros lassen sich Zeichenfolgen mit einem Bezeichner versehen. Einige Standardvorgaben (z. B. `&`) werden von XML vordefiniert und müssen nicht gesondert angegeben werden. Andere können in einer DTD definiert werden.

6.3.7 Kommentare

Kommentare werden in XML durch `<!-- ... -->` angegeben. In einem Kommentar können beliebige Texte vorkommen.

6.4 Namensräume

In XML-Dokumenten können Elemente aus verschiedenen Sprachen verwendet werden. Sie können z. B. eine mathematische Formel als MathML-Element in einem HTML-Dokument einbauen. Damit der XML-Prozessor (z.B. HTML-Browser) zwischen den Elementen beider Sprachen unterscheiden kann und diese unterschiedlich behandelt (z. B. das Plugin für die Darstellung von MathML aufruft), wurden in XML Namensräume eingeführt.

Bevor ein Namensraum verwendet werden kann, muss dieser zuerst im Dokument deklariert werden. Die Deklaration erfolgt in der Form eines Attributes innerhalb eines Elements:

```
<namensraum:elementName> < ... xmlns:namensraum="url">
```

Man kann in einem Dokument mehrere Namensräume verwenden. Alle Nachkommen dieses Elements werden Teil des Namensraumes. Zur Verwendung von Namensräumen werden sie durch ein Namensraum-Präfix gefolgt von einem Doppelpunkt gefolgt vom lokalen Namen des Elements angegeben. Der Wert des `xmlns:`-Attributs ist eine URL, die gewöhnlich zu der Organisation gehört, die den Namensraum unterhält. Es ist jedoch nicht erforderlich, dass der XML-Prozessor diese URL benutzt. Die Angabe der URL dient lediglich der eindeutigen Bezeichnung des Namensraums.

Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bestellung xmlns:produkt="http://localhost/XML/produkt" xmlns:kunde="http://localhost/XML/kunde">
       <produkt:nummer>p49393</produkt:nummer>
       <produkt:name>JXY Rasierer VC100</produkt:name>
       <produkt:menge>1</produkt:menge>
       <produkt:preis>69</produkt:preis>

       <kunde:nummer>k2029</kunde:nummer>
       <kunde:name>Meier, Fritz</kunde:name>
       <kunde:lieferadresse>Donnerbalkenstr.14, 80111 München</kunde:lieferadresse>
</bestellung>
```

6.5 Wohlgeformt vs. Gültig

Generell muss man bei allen Parsern noch unterscheiden, auf Grund welcher Syntax sie entscheiden, ob ein Dokument korrekt ist, oder nicht. Es wurde ja bereits erwähnt, dass es im Allgemeinen zwei Grammatiken gibt, die für ein XML Dokument gelten können: die Grammatik für XML selbst und die DTD für eine bestimmte XML-basierte Sprache.

Daran anlehnend spricht man im Bereich der XML Parser von einem validierenden Parser, wenn er neben der XML Grammatik auch noch die Gültigkeit eines Dokumentes bezüglich seiner DTD überprüft. Ein nicht-validierender Parser prüft nur die Wohlgeformtheit des Dokumentes, also nur die XML Grammatik. XML Dokumente, die nicht wohlgeformt sind, können in der Regel nicht verarbeitet werden.

Wenn statt DTDs zukünftig XML Schema Definitionen zur Spezifikation von XML Dokumenten benutzt werden, dann können Parser prüfen, ob ein gegebenes XML Dokument mit dem Schema übereinstimmt (Schema Validation).

6.6 DTD - Document Type Definition

In einer DTD werden mit Hilfe einer erweiterten Backus-Naur-Form beschrieben, wie der Dokumentkörper eines XML Dokumentes aufgebaut sein muss, also welche Tag- Namen erlaubt sind, welche Elemente wie geschachtelt sein dürfen, welche Elemente welche Attribute haben dürfen, etc.

6.6.1 Elemente

Das entscheidende Strukturprinzip in XML bilden die Elemente. Ein Element wird über seinen Namen identifiziert und hat einen definierten Inhalt:

```
<!ELEMENT Name Inhalt>
```

Der Inhalt eines Elements kann aus einer Kombinationen von anderen Elementen und/oder Text bestehen oder leer sein. Es existieren mehrere Möglichkeiten der Kombination:

- Sequenz

  ```
  <!ELEMENT Nachricht (Absender, Empfänger, Nachrichtentext)>
  ```

- Option

  ```
  <!ELEMENT Nachrichtentext (Betreff?, Textkörper)>
  ```

- Alternative

```
<!ELEMENT Block (PARA | TABLE | IMG)>
```

- Iteration

```
<!ELEMENT Liste (Name, Item+)>
<!ELEMENT Bus (Fahrgast*)>
```

- Text-Inhalt

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

- gemischter Inhalt

```
<!ELEMENT PARA (#PCDATA | I | B | A)*>
```

- beliebiger Inhalt

```
<!ELEMENT Divers ANY>
```

- leere Elemente

```
<!ELEMENT BR EMPTY>
```

6.6.2 Attribute

Elemente in XML können jedoch nicht nur einen Inhalt haben; zusätzlich besteht die Möglichkeit, sie mit Attributen zu versehen. In der DTD wird dies dadurch definiert, dass einem Element Attributlisten zugewiesen werden.

```
<!ATTLIST Bus
        Fahrzeugtyp CDATA #IMPLIED
        Kennzeichen CDATA #REQUIRED>

<!ATTLIST Fahrgast
        id ID #REQUIRED
        stammkunde (ja|nein) "nein">
```

Das Attribut vom Typ ID hat in XML eine besondere Bedeutung: keine ID eines Fahrgast-Elements darf den gleichen Wert haben wie die ID irgend eines anderen Fahrgast-Elements. Diese Eigenschaft wird auch vom Parser überprüft. Ein solches Attribut spielt damit die Rolle eines eindeutigen Schlüssels (vergleichbar mit einem Primärschlüssel) und kann beim Suchen und Navigieren entsprechend ausgenutzt werden.

Bei Attribut-Deklarationen wird CDATA (character data) angegeben im Gegensatz zu PCDATA bei Element-Deklarationen. PCDATA steht für parsable character data. Im Gegensatz zu PCDATA wird CDATA nicht mehr vom Parser analysiert. Dadurch werden innerhalb von CDATA keine Markup-Zeichen wie spitze Klammern erkannt.

6.7 XML Schema

Die Nachteile, ein XML-Dokument mit Hilfe einer DTD zu beschreiben sind:

- DTDs unterstützen keine Namensräume.
- DTDs haben eine sehr beschränkte Anzahl an Datentypen, z.B. gibt es keine Datentypen für Zahlen, Datum etc.
- DTDs sind selbst keine XML-Dokumente.

XML Schema (kurz XSchema) ist ein Standardisierungsvorschlag des W3C, der DTDs ersetzen soll. XSchema selbst ist eine XML-basierte Sprache, wodurch die Verarbeitung von XSchema-Dokumenten mit den gleichen Werkzeugen wie für XML-Dokumente erfolgen kann (Parser, DOM-Schnittstelle etc.).

6.7.1 Grundlagen

XSchema-Dokumente haben üblicherweise die Endung *.xsd. Ein XSchema hat den folgenden Aufbau:

```
<xsd:schema
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <xsd:annotation>
              <xsd:documentation xml:lang="de-DE">
                     Dokumentation über das Schema
              </xsd:documentation>
       </xsd:annotation>
       <xsd:element name="x" type="xType"/>
       ...
</xsd:schema>
```

Der Element `xsd:annotation` ist optional und dient zur Angabe von allgemeinen Informationen über das Schema.

6.7.2 Schemadateien referenzieren

Zur Wiederholung nochmals ein Beispiel zum Einbinden eines DTD-Dokuments:

```
<?xml version="1.0"?>
<!DOCTYPE person SYSTEM "person.dtd">
<person>
       <fullname> Andreas Brachinger </fullname>
       <tel> +43 7412 52575513 </tel>
</person>
```

Bei Schemadateien ist folgende Syntax anzugeben, wobei in diesem Beispiel die Angabe inkl. Verwendung von Namensräumen gezeigt wird:

```
<?xml version="1.0"?>
<person xmlns="http://www.mysite.com"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://www.mysite.com person.xsd">
       <fullname> Andreas Brachinger </fullname>
       <tel> +43 7412 52575513 </tel>
</person>
```

Verwendet man keine Namensräume so wird das Attribut `xsi:noNamespaceSchemaLocation` verwendet.

```
<?xml version="1.0"?>
<person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="person.xsd">
       <fullname> Andreas Brachinger </fullname>
       <tel> +43 7412 52575513 </tel>
</person>
```

6.7.3 Vergleich einer DTD zu einem XML-Schema

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       targetNamespace="http://www.w3schools.com"
       xmlns="http://www.w3schools.com"
       elementFormDefault="qualified">
       <xs:element name="note">
              <xs:complexType>
                     <xs:sequence>
                            <xs:element name="to" type="xs:string"/>
                            <xs:element name="from" type="xs:string"/>
                            <xs:element name="heading" type="xs:string"/>
                            <xs:element name="body" type="xs:string"/>
                     </xs:sequence>
              </xs:complexType>
       </xs:element>
</xs:schema>
```

Erklärung:

- Der Auszug `xmlns:xs=''http://www.w3.org/2001/XMLSchema''` gibt an, dass alle Elemente und Datentypen von dem Namensraum `http://www.w3.org/2001/XMLSchema` kommen. Daher ist auch der Präfix xs anzugeben.
- Der Auszug `targetNamespace=''http://www.w3schools.com''` gibt an, dass die Elemente, die in diesem Schema definiert werden (to, from, heading, body) vom Namensraum `http://www.w3schools.com` stammen.
- `elementFormDefault=''qualified''` - Dieses Attribut legt fest, ob lokal definierte Elemente dem Namensraum des Schemas angehören (qualified) oder keinem Namensraum angehören (unqualified)

Um die folgenden Aufgaben lösen zu können, verweise ich auf folgende Webseite:

https://www.w3schools.com/xml/schema_intro.asp.

6.8 Aufgaben

6.8.1 Verein

Erstellen Sie ein XML Dokument, das der beschriebenen DTD entspricht und fügen Sie fünf Spieler ein. Das XML Dokument soll die unterschiedlichen Möglichkeiten veranschaulichen und daher möglichst viele der beschriebenen Einschränkungen abdecken (optionale Elemente, verschiedene Werte für Attribute etc.). Folgende Punkte sollen dabei erfüllt sein:

- Es gibt ein Element verein.
- Es gibt genau ein Subelement name von verein.
- Es gibt ein oder mehrere Subelemente spieler von verein.
- Das Element spieler hat ein erforderliches Attribut trikotnr, und ein optionales Attribut bevorzugt das ausschließlich “mittelfeld”, “tor”, “flanke” oder “ersatzbank” beinhalten darf.
- Das Element spieler hat Subelemente name (genau einmal) und position (beliebig oft, auch keines).
- Das Element spieler hat weiters ein Subelement news (beliebig oft, auch keines).
- news besteht aus gemischtem Inhalt (beliebige Kombination von name-Elementen und Text).
- Nicht näher beschriebene Elemente und Attribute enthalten lediglich Text.

Die zugehörige DTD hat folgenden Aufbau:

```
<?xml version=”1.0” encoding=”UTF−8”?>
<!ELEMENT verein (name, spieler+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT spieler (name, position∗, news∗)>
<!ELEMENT position (#PCDATA)>
<!ELEMENT news (#PCDATA | name)∗>
<!ATTLIST spieler trikotnr CDATA #REQUIRED>
<!ATTLIST spieler bevorzugt (mittelfeld|tor|flanke|ersatzbank) #IMPLIED>
```

6.8.2 Kino

Gegeben sei folgendes XML-Dokument:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE kino SYSTEM "kino.dtd">
<kino>
	<standort>
		<saal kennung="am">
			<name>Audi Max</name>
		</saal>
		<saal kennung="s2">
			<name>Saal 2</name>
		</saal>
		<saal kennung="s3">
			<name>Saal 3</name>
		</saal>
		<saal kennung="s4">
			<name>Saal 4</name>
		</saal>
	</standort>
	<filme>
		<film nr="f123">
			<name>Krieg der Sterne IV</name>
			<regie>George Lucas</regie>
			<darsteller>Mark Hamill</darsteller>
			<darsteller>Harrison Ford</darsteller>
		</film>
		<film nr="f124">
			<name>Der Herr der Ringe - Die Gefährten</name>
			<regie>Peter Jackson</regie>
```

```
                        <darsteller>Elijah Wood</darsteller>
                        <darsteller>Ian McKellen</darsteller>
                </film>
                <film nr="f125">
                        <name>Der Pate</name>
                        <regie>Francis Ford Coppola</regie>
                        <darsteller>Marlon Brando</darsteller>
                        <darsteller>Al Pacino</darsteller>
                </film>
        </filme>
        <spielplan>
                <auffuehrung datum="2013-10-22" film="f123">
                        <ort uhrzeit="22:00" saal="am"/>
                        <ort uhrzeit="22:30" saal="s4"/>
                </auffuehrung>
                <auffuehrung datum="2013-10-22" film="f124">
                        <ort uhrzeit="19:15" saal="am"/>
                        <ort uhrzeit="21:00" saal="s2"/>
                        <ort uhrzeit="20:45" saal="s3"/>
                </auffuehrung>
                <auffuehrung datum="2013-10-22" film="f125">
                        <ort uhrzeit="20:15" saal="s3"/>
                </auffuehrung>
                <auffuehrung datum="2013-10-23" film="f123"/>
        </spielplan>
</kino>
```

Geben Sie eine DTD `kino.dtd` an, sodass das xml-Dokument `kino.xml` bezüglich dieser DTD gültig ist.

Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Die Reihenfolge der Elemente soll wie in `kino.xml` angegeben fixiert werden.
- Das Wurzelelement beinhaltet jeweils genau ein Element `standort, filme` und `spielplan`.
- Ein `saal` beinhaltet genau ein `name` Element.
- Ein `standort` beinhaltet ein oder mehrere `saal` Elemente.
- Ein `film` beinhaltet genau ein `name` und `regie` Element, jedoch zwei oder mehr `darsteller` Elemente.
- Alle weiteren Elemente können beliebig oft vorkommen.
- Alle Attribute sollen verpflichtend sein.
- Die Attribute `kennung` und `nr` sollen vom Typ `ID` sein, und `film` sowie `saal` sollen Referenzen auf die ID-Attribute sein.

6.8.3 Oper

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE oper SYSTEM "oper.dtd">
<oper>
        <werke>
                <werk id="w1">
                        <komponist>Verdi</komponist>
                        <name>La Traviata</name>
                        <urauffuehrung>1853</urauffuehrung>
```

```
		</werk>
		<werk id="w2">
			<komponist>Puccini</komponist>
			<name>Tosca</name>
			<urauffuehrung>1900</urauffuehrung>
		</werk>
		<werk id="w3">
			<komponist>Puccini</komponist>
			<name>Turandot</name>
			<urauffuehrung>1926</urauffuehrung>
		</werk>
		<werk id="w4">
			<komponist>Mozart</komponist>
			<name>Don Giovanni</name>
			<urauffuehrung>1787</urauffuehrung>
		</werk>
	</werke>
	<inszenierungen>
		<inszenierung id="i1" werk="w4">
			<regie>Willy Decker</regie>
			<ort>Sachsische Staatsoper</ort>
			<premiere>1993</premiere>
		</inszenierung>
			<inszenierung id="i2" werk="w1">
			<regie>Hans Gratzer</regie>
			<ort>Wiener Volksoper</ort>
			<premiere>2007</premiere>
		</inszenierung>
		<inszenierung id="i3" werk="w2">
			<regie>Margarethe Wallmann</regie>
			<ort>Wiener Staatsoper</ort>
			<premiere>1957</premiere>
		</inszenierung>
	</inszenierungen>
	<wertungen>
		<bewertung inszenierung="i1" wert="2"/>
		<bewertung inszenierung="i3" wert="5"/>
	</wertungen>
</oper>
```

Geben Sie eine DTD `oper.dtd` an, sodass das xml-Dokument `oper.xml` bezüglich dieser DTD gültig ist.

Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Die Reihenfolge der Elemente soll wie in `oper.xml` angegeben xiert werden.
- Alle Attribute sind verpflichtend.
- Das Wurzelelement (`oper`) beinhaltet jeweils genau ein Element `werke, inszenierungen` und `wertungen`.
- `werke` enthält ein oder mehrere `werk` Elemente.
- `werk` enthalt genau ein `komponist, name` und `urauffuehrung` Element. Das Attribut `id` soll global eindeutig sein.
- `inszenierungen` enthält ein oder mehrere `inszenierung` Elemente.
- `inszenierung` enthält jeweils ein `regie, ort` und `premiere` Element. Das Attribut `id` von `inszenierung` soll global eindeutig sein, und das Attribut `werk` von `inszenierung` soll vom Typ `IDREF` sein.

- `wertungen` enthält beliebig viele `bewertung` Elemente.
- `bewertung` ist ein leeres Element und hat zwei Attribute `wert` und `inszenierung`. Stellen Sie sicher, dass das Attribut `wert` nur die Werte 1 bis 5 annehmen kann. Das Attribut `inszenierung` soll vom Typ `IDREF` sein.

Lösung:

```
<?xml version=”1.0” encoding=”UTF−8”?>
<!ELEMENT komponist (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT urauffuehrung (#PCDATA)>
<!ELEMENT regie (#PCDATA)>
<!ELEMENT ort (#PCDATA)>
<!ELEMENT premiere (#PCDATA)>
<!ELEMENT oper (werke,inszenierungen,wertungen)>
<!ELEMENT werke (werk+)>
<!ELEMENT werk (komponist, name, urauffuehrung)>
<!ATTLIST werk id ID #REQUIRED>
<!ELEMENT inszenierungen (inszenierung+)>
<!ELEMENT inszenierung (regie, ort, premiere)>
<!ATTLIST inszenierung id ID #REQUIRED werk IDREF #REQUIRED>
<!ELEMENT wertungen (bewertung∗)>
<!ELEMENT bewertung EMPTY>
<!ATTLIST bewertung inszenierung IDREF #REQUIRED wert (1|2|3|4|5) #REQUIRED>
```

6.8.4 Anwendung einer DTD

Betrachten Sie die folgende DTD `test.dtd`:

```
<?xml version=”1.0” encoding=”UTF−8”?>
<!ELEMENT A (B∗,C∗)>
<!ELEMENT B (C+,B?)>
<!ATTLIST B K ID #REQUIRED>
<!ELEMENT C (#PCDATA|D|E)∗>
<!ELEMENT D (#PCDATA)>
<!ELEMENT E EMPTY>
<!ATTLIST C L IDREF #IMPLIED>
```

Betrachten Sie weiters die acht verschiedenen XML-Dateien, die unten angeführt sind.

Hinweise:

- Gehen Sie davon aus, dass allen folgenden Dateien die Zeile `<!DOCTYPE A SYSTEM ''test.dtd''>` vorangestellt ist.
- Sie können auch davon ausgehen, dass alle folgenden XML-Dateien wohlgeformt sind. Es geht also lediglich darum, ihre Gültigkeit bezüglich test.dtd zu entscheiden.

Geben Sie an, welche der folgenden XML-Dateien gültig bezüglich `test.dtd` sind.

```
1 <A><B K="A1">abc</B><C>cde<D>fgh</D></C></A>
2 <A><B K="A1"><C>abc</C><C>fgh</C></B></A>
3 <A><B K="A1"><C>abc</C></B><C L="A2">def</C></A>
4 <A><C><E/><E></E></C></A>
5 <A><C><E><E/></E></C></A>
6 <A><B K="A1"><C><D>abc</D></C><B K="A2"><C>def</C></B></B></A>
7 <A><B K="A1"><C><D>abc</D></C></B><B K="A2"><C>def</C></B></A>
8 <A><C></C></A>
```

6.8.5 Dancing Stars

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dancingstars SYSTEM "dancing.dtd">
<dancingstars>
	<paar nr="1" ergebnis="ausgeschieden in Runde 1">
		<partner>Alexander Kreissl</partner>
		<promi>Claudia Stoeckl</promi>
	</paar>
	<paar nr="2" ergebnis="ausgeschieden in Runde 3">
		<partnerin>Julia Polai</partnerin>
		<promi>Oliver Stamm</promi>
	</paar>
	<paar nr="3" ergebnis="Finale verloren">
		<promi>Elisabeth Engstler</promi>
		<partner>Alexander Zaglmaier</partner>
	</paar>
	<paar nr="4" ergebnis="ausgeschieden in Runde 5">
		<partnerin>Kelly Kainz </partnerin>
		<promi>Marc Pirchner</promi>
	</paar>
		<paar nr="5" ergebnis="ausgeschieden in Runde 4">
		<promi>Christine Reiler</promi>
	<partner>Manfred Zehender</partner>
	</paar>
	<paar nr="6" ergebnis="ausgeschieden in Runde 7">
		<partnerin>Alice Guschelbauer</partnerin>
		<promi>Waterloo</promi>
	</paar>
	<paar nr="7" ergebnis="Finale gewonnen">
		<partnerin>Nicole Kuntner</partnerin>
		<promi>Dorian Steidl</promi>
	</paar>
	<paar nr="8" ergebnis="ausgeschieden in Runde 6">
		<partner>Balazs Ekker</partner>
		<promi>Jeannine Schiller</promi>
	</paar>
	<paar nr="9" ergebnis="ausgeschieden in Runde 2">
		<partner>Michaela Heintzinger</partner>
		<promi>Peter Tichatschek</promi>
	</paar>
	<paar nr="10" ergebnis="ausgeschieden im Semifinale">
		<partner>Andy Kainz</partner>
		<promi>Elke Winkens</promi>
	</paar>
</dancingstars>
```

Geben Sie eine DTD `dancing.dtd` an, sodass das xml-Dokument `dancing.xml` (siehe Anhang) bezüglich dieser DTD gültig ist. Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Das Wurzelelement beinhaltet eine beliebige Anzahl von „paar“-Elementen (jedoch zumindest eines).
- Das „paar“-Element beschreibt ein Tanzpaar. Ein solches besteht aus genau einem „promi“ sowie entweder einem „partner“ oder einer „partnerin“. Beachten Sie, dass die Reihenfolge jedoch nicht fixiert ist!
- Alle Attribute sollen verpflichtend sein.

Lösung:

```
<!ELEMENT dancingstars (paar)+>
<!ELEMENT paar
(((partnerin|partner),promi)|(promi,(partnerin|partner)))>
<!ATTLIST paar
nr CDATA #REQUIRED
ergebnis CDATA #REQUIRED>
<!ELEMENT partnerin (#PCDATA)>
<!ELEMENT partner (#PCDATA)>
<!ELEMENT promi (#PCDATA)>
```

Überprüfen Sie Ihre Lösung mit folgendem Validator: http://www.xmlvalidation.com

6.8.6 Anwendung eines Schemas

Betrachten Sie die folgende xml-Schema Datei `test.xsd`:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
	<xsd:element name="a" type="atype"/>

	<xsd:complexType name="atype">
		<xsd:choice minOccurs="1" maxOccurs="2">
			<xsd:element name="b" type="btype" maxOccurs="2"/>
			<xsd:element name="c" type="xsd:int" />
		</xsd:choice>
	</xsd:complexType>

	<xsd:complexType name="btype" mixed="true">
		<xsd:sequence>
			<xsd:element name="b" type="btype" minOccurs="0" maxOccurs="2"/>
			<xsd:element name="c" type="xsd:int" minOccurs="0" maxOccurs="unbounded" />
		</xsd:sequence>
	</xsd:complexType>
</xsd:schema>
```

Betrachten Sie weiters die acht verschiedenen xml-Dateien, die unten angeführt sind. Sie können davon ausgehen, dass alle folgenden xml-Dateien wohlgeformt sind. Es geht also lediglich darum, ihre Gültigkeit bezüglich `test.xsd` zu entscheiden.

```
1. <a/>
2. <a><c>1</c></a>
3. <a><b/><b/><b/><b/></a>
4. <a><c>1</c><c>2</c><b/></a>
5. <a><b><b><c>Zahl 3</c></b></b></a>
6. <a><b><b/><b/></b><b/></a>
7. <a><b><b/><b/><b/></b></a>
8. <a><b><b>Zahl <c>3</c></b></b></a>
```

Lösung: u,g,u,u,u,g,u,g

6.8.7 Schach

Vervollständigen Sie die XML Schema Datei `schach.xsd` zur Speicherung von Spiel-Eröffnungen, sodass XML-Dokumente in der Gestalt von `schach.xml` (siehe Anhang) bezüglich dieses Schemas gültig sind.

Berücksichtigen Sie beim Erstellen des Schemas folgende Punkte:

- Das Element schach beinhaltet mindestens ein eroffnung Element. Jedes eroffnung Element hat ein Attribut name und genau ein Kindelement weiß.
- Elemente weiß haben gemischten Inhalt und potentiell mehrere Kindelemente schwarz. Elemente schwarz sollen keinen gemischten Inhalt haben und maximal ein Kindelement weiß besitzen.
- Definieren Sie die Attribute der Elemente weiß und schwarz entsprechend. Hinweise: Nutzen Sie attributeGroup um Schreibaufwand zu sparen.
- Sollten bei bestimmten Elementen oder Attributen keine näheren Angaben bezüglich des genauen Typs vorgegeben sein, wählen Sie selbst einen sinnvollen Typ aus.

`schach.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<schach>
	<eröffnung name="Englische Eröffnung">
	<weiß von="c2" nach="c4">
	Englische Symmetrievariante:
		<schwarz von="c7" nach="c5"/>
		Sizilanisch im Anzuge:
		<schwarz von="e7" nach="e5"/>
		Neutraler Entwicklungszug:
		<schwarz von="g8" nach="f6"/>
	</weiß>
	</eröffnung>
	<eröffnung name="Damengambit">
		<weiß von="d2" nach="d4">
			<schwarz von="d7" nach="d5">
				<weiß von="c2" nach="c4">
					Angenommenes Damengambit:
					<schwarz von="d5" nach="c4" schlagt="true"/>
					Abgelehntes Damengambit:
					<schwarz von="e7" nach="e6"/>
				</weiß>
			</schwarz>
		</weiß>
	</eröffnung>
	<eröffnung name="Spanische Eröffnung">
		<weiß von="e2" nach="e4">
			<schwarz von="e7" nach="e5">
				<weiß von="g1" nach="f3">
					<schwarz von="b8" nach="c6">
						<weiß von="f1" nach="b5"/>
					</schwarz>
				</weiß>
			</schwarz>
		</weiß>
	</eröffnung>
</schach>
```

6.8.8 Movies

Vervollständigen Sie die XML Schema Datei `movies.xsd`, sodass XML-Dokumente in der Gestalt von `movies.xml` (siehe Anhang) bezüglich dieses Schemas gültig sind. XML-Dokumente der Form `movies.xml` sollen eine Liste von Filmen abspeichern. Berücksichtigen Sie beim Erstellen des Schemas folgende Punkte:

- Das Element `movies` ist das Wurzelelement und besteht aus mindestens zwei `movie`-Elementen.
- Jedes `movie`-Element beinhaltet Information zu einem Film. Ein Film besteht aus genau einem `title`-Element, genau einem `releaseYear`-Element und genau einem `imdbRating`-Element gefolgt von genau einem `characters`-Element. Die Reihenfolge dieser Elemente soll eingehalten werden.
- Das `characters`-Element hat mindestens zwei `name`-Elemente.
- Das `imdbRating`-Element darf einen Wert von 0 bis 10 beinhalten mit einer Nachkommastelle.
- Das `releaseYear`-Element darf einen Wert nicht kleiner als 1900 und nicht größer als 2014 beinhalten.
- Wählen Sie anhand des Dokuments im Anhang sinnvolle Typen und Häufigkeiten aus.
- Alle Attribute sollen verpflichtend sein. Dies soll in Ihrer XML-Schema Definition explizit ablesbar sein. Es sind keine Schlüssel zu definieren.

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
	<movie url="http://www.imdb.com/?move=....">
		<title>Die Verurteilten</title>
		<releaseYear>1994</releaseYear>
		<imdbRating>9.2</imdbRating>
		<characters>
			<name>Tim Robbins</name>
			<name>Morgan Freeman</name>
		</characters>
	</movie>

	<movie url="http://www.imdb.com/?move=....">
		<title>Der Pate</title>
		<releaseYear>1972</releaseYear>
		<imdbRating>9.2</imdbRating>
		<characters>
			<name>Marlon Brando</name>
			<name>Al Pacino</name>
		</characters>
	</movie>

	<movie url="http://www.imdb.com/?move=....">
		<title>Der Pate 2</title>
		<releaseYear>1974</releaseYear>
		<imdbRating>9.0</imdbRating>
		<characters>
			<name>Al Pacino</name>
			<name>Robert De Niro</name>
		</characters>
	</movie>

	<movie url="http://www.imdb.com/?move=....">
		<title>Pulp Fiction</title>
		<releaseYear>1994</releaseYear>
		<imdbRating>8.9</imdbRating>
		<characters>
			<name>John Travolta</name>
			<name>Samuel L. Jackson</name>
		</characters>
	</movie>
</movies>
```