XML

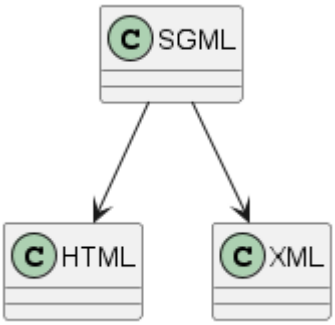
1. Grundlagen

TIP	XML = Extensible Markup Language

Ist eine strukturierte Sprache, um Daten und Datenstrukturen abzuspeichern.
HTML basiert auf demselben Grundprinzip wie XML.

1.1. SGML

SGML ist die Muttersprache für heutzutage Anwendungen wie HTML und XML.



1.2. Einsatzgebiete

1.2.1. Internet

In den Anfangszeiten des Internets wurde XML verwendet, um REST-Schnittstellen zu betreieben, und daten zu übertragen

Heutzutage hat **JSON** diese Rolle im Internet übernommen.
Aufgrund backwards compatability unterstützen aber selbst moderne Browser weiterhin XML

1.2.2. Maven

Maven verwendet seit ewigkeiten XML für die Konfiguration, wie aber auch für die sogenannte pom.xml, welche in jedem Maven Projekt eine wichtige Rolle für die Konfiguration von Depedencies, Plugins, und weiteres bietet.

1.2.3. Weitere XML Sprachen

- XHMTL (XML HTML)
- SVG (Scalable Vector Graphics)
- MathML (Mathe Formeln)

1.3. Struktur von XML

Beispiel XML (Premiere Pro rendering presets)

```
<PremiereData Version="3">
    <Key Version="1" Index="0">
        <DirectoryPath>/presets/</DirectoryPath>
        <Key Version="3" Index="0">
            <PresetModifiedTime>20230207T173300.529816</PresetModifiedTime>
            <PresetFileType>1211250228</PresetFileType>
            <PresetClassID>1313424203</PresetClassID>
            <PresetPath>/presets/</PresetPath>
            <PresetName>Gemeinde</PresetName>
            <PresetID>3740cd45-4da5-46ac-9006-741865229db7</PresetID>
            <FolderDisplayPath>User Presets & Groups</FolderDisplayPath>
            <IngestPreset>false</IngestPreset>
        </Key>
        <Key Version="3" Index="1">
            <PresetModifiedTime>20221122T195107.744243</PresetModifiedTime>
            <PresetFileType>1211250228</PresetFileType>
            <PresetClassID>1313424203</PresetClassID>
            <PresetPath>/presets/</PresetPath>
            <PresetName>School 720p low No Audio</PresetName>
            <PresetID>aaec6bcc-015b-4e9c-80c3-554512f0d2c6</PresetID>
            <FolderDisplayPath>User Presets & Groups</FolderDisplayPath>
            <IngestPreset>false</IngestPreset>
        </Key>
        <Key Version="3" Index="2">
            <PresetModifiedTime>20221122T193829.687985</PresetModifiedTime>
            <PresetFileType>1211250228</PresetFileType>
            <PresetClassID>1313424203</PresetClassID>
            <PresetPath>/presets/</PresetPath>
            <PresetName>School 720p low</PresetName>
            <PresetID>7d913976-df8e-4704-930f-d70c00e70902</PresetID>
            <FolderDisplayPath>User Presets & Groups</FolderDisplayPath>
            <IngestPreset>false</IngestPreset>
            </Key>
        </Key>
</PremiereData>
```

2. Vorteile von XML

- Human readable
- Format der Daten wird beibehalten

3. XML - Schnittstellen

3.1. SAX

TIP	SAX = Simple API for XML

SAX ist eine API, um einzelne Datenwerten von XML Dateien zu erhalten.
Hierbei wird das Dokument sequenziell (von oben nach unten) eingelesen. Bereits während des Leseprozesses kann eine Applikation über einzelne Bestandteile des XML Dokumentes informiert werden.

SAX eignet sich besonders gut für große XML Dokumenten, bei denen nur wenige Daten benötigt werden.

3.2. DOM

TIP	DOM = Document Object Model

DOM ist eine standardisierte API für die Erstellung und Manipulation eines XML Information Sets.
Hierbei wird das XML Dokument eingelesen, und in Entities im Hauptspeicher abgespeichert (parsing).
Sollten nun Änderungen am Dokument vorgenommen werden, kann dieses ebenfalls wieder gespeichert werden (serialisieren).

DOM eignet sich besonders gut für kleiner XML Dokumenten, auf die oft Zugriffe erfolgen. Ebenfalls ist es eine einfache Methode, XML Dokumente zu bearbeiten oder zu speichern. Für einzelne Abfragen ist SAX besser geeignet.
Ebenfalls kann es bei großen XML-Dokumenten vorkommen, dass der Hauptspeicher nicht ausreichend für das gesamte Dokument ist. Eine Lösung für dies ist, dass nur einzelne Teile einer XML-Datei in einem DOM angelegt werden.

Moderne Webbrowser parsen ein HTML Dokument in eine DOM, da diese sehr klein sind (wenige Megabyte), allerdings konstanten Zugriff auf dieser erforderlich ist.

4. XML-Prozessoren (Beispiele)

4.1. Xerces

- Im Rahmen des Apache-Projektes entstanden.
- Validierende Parser für Java und C++
- DOM Level 1 & 2, wie auch SAX
- Unterstützung von XML Schemas und validierung

4.2. MSXML Parser

Microsoft XML Parser.

- DOM und SAX Prozessoren
- überprüfung von Konformität und Schema

5. Validierung von XML

5.1. DTD

TIP	DTD = Document Type Definition

In einer DTD werden Grundregeln für die Struktur eines Dokumentes festgelegt:

DTD Definition

```
<!-- DTD-Definition Adressen -->
<!ELEMENT address(vorname, name, strasse, plz, ort)>
<!ELEMENT vorname #PCDATA>
<!ELEMENT name #PCDATA>
<!ELEMENT strasse #PCDATA>
<!ELEMENT plz #PCDATA>
<!ELEMENT ort #PCDATA>
```

DTD in XML

```
<?xml version="1.0" encoding="iso-8859-2" ?>
<!DOCTYPE customers[
    <!ELEMENT customers (customer*)>
        <!ELEMENT customer (name, address+)>
            <!ELEMENT name (#PCDATA)>
            <!ELEMENT address (street, city, state?, postal)>
                <!ATTLIST address country NMTOKEN #REQUIRED> ①
                <!ELEMENT street (#PCDATA)>
                <!ELEMENT city (#PCDATA)>
                <!ELEMENT state (#PCDATA)>
                <!ELEMENT postal (#PCDATA)>
]>
<customers>
    <customer>
        <name>Baeckerei Huber</name>
        <address country=" AT ">
            <street>Landstrasse 63</street>
            <city>Linz</city>
            <postal>4020</postal>
        </address>
    </customer>
```

```
<customer>
```

① Bei der Adresse muss ein Land angegeben sein!

Die Struktur der XML-Datei dar nicht von der DTD abweichen, sonst ist die XML-Datei ungültig:

Ein DTD kann auch in einem externen .DTD File gespeichert werden.

5.1.1. Nachteile von DTD:

- Eine weite Syntax für Entwickler
- Nicht erweiterbar
- schlechte Unterstützung von XML Namespaces
- schwache Unterstützung von Datentypen (numerisch?)
- geringe Möglichkeiten der exakten definition von Tags
- keine Erweiterbarkeit

5.2. XML Schema

```
<?xml version="1.0" encoding="iso-8859-1"?> ①
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"> ②
    <xsd:element name="Buch">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Titel" type="xsd:string"/>
                <xsd:element name="Inhalt" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

① Angabe der XML-Versionsnummer und des Encodings

② Angabe der Datenstrukturen innerhalb des XMLs