

# **Dokumentation zum XML-Projekt im Master-Modul Internettechnologien**

**Lennart Mende**

**Richard Mende**

HTWK Leipzig

Wintersemester 2025/26

Prof. Dr.-Ing. Andreas Pretschner

30. Dezember 2025

# Inhaltsverzeichnis

<b>1</b>	<b>Projektübersicht</b>	<b>1</b>
1.1	Zielstellung . . . . .	1
<b>2</b>	<b>Datensatz</b>	<b>3</b>
<b>3</b>	<b>Transformationen</b>	<b>3</b>
3.1	XSLT von XML in HTML . . . . .	3
3.2	FOP-Transformation . . . . .	4
3.3	Vergleich der Transformationen . . . . .	4
<b>4</b>	<b>Programmierung</b>	<b>4</b>
<b>5</b>	<b>Quellenverzeichnis</b>	<b>4</b>

# 1 Projektübersicht

## 1.1 Zielstellung

Als Grundlage dieses Projekts dient das Anlegen eines XML-Datensatzes. Dieser sollte aus mehreren Objekten mit jeweils mindestens einem Attribut und einer Datensequenz bestehen. Die daran anschließende Aufgabe besteht im Erstellen einer XSD-Schemadatei. Damit kann validiert werden, ob die gegebene XML-Datei die gewünschte Struktur erfüllt und alle benötigten Angaben gemacht wurden. Darüber hinaus soll eine in *editiX* automatisch generierte XSD-Schemadatei erstellt werden.

Der zweite Teil des Projekts befasst sich mit der Transformation der XML-Datei. Als Erstes sollen die Daten auf einer HTML-Page dargestellt werden. Dazu muss zunächst die XSLT-Transformationsdatei erzeugt werden. Darüber hinaus soll die Transformation in eine PDF erfolgen. Dazu gibt es die Möglichkeit der direkten FOP-Transformation oder die Docbook-Transformation. Eine übersichtliche Darstellung dieser Transformationen zeigt Abbildung 1.

Der dritte Aufgabenteil setzt sich mit der XML-Programmierung auseinander. Dabei soll eine Applikation erstellt werden, welche die Validierung, Abfrage und Transformation der XML-Datei ermöglichen soll.

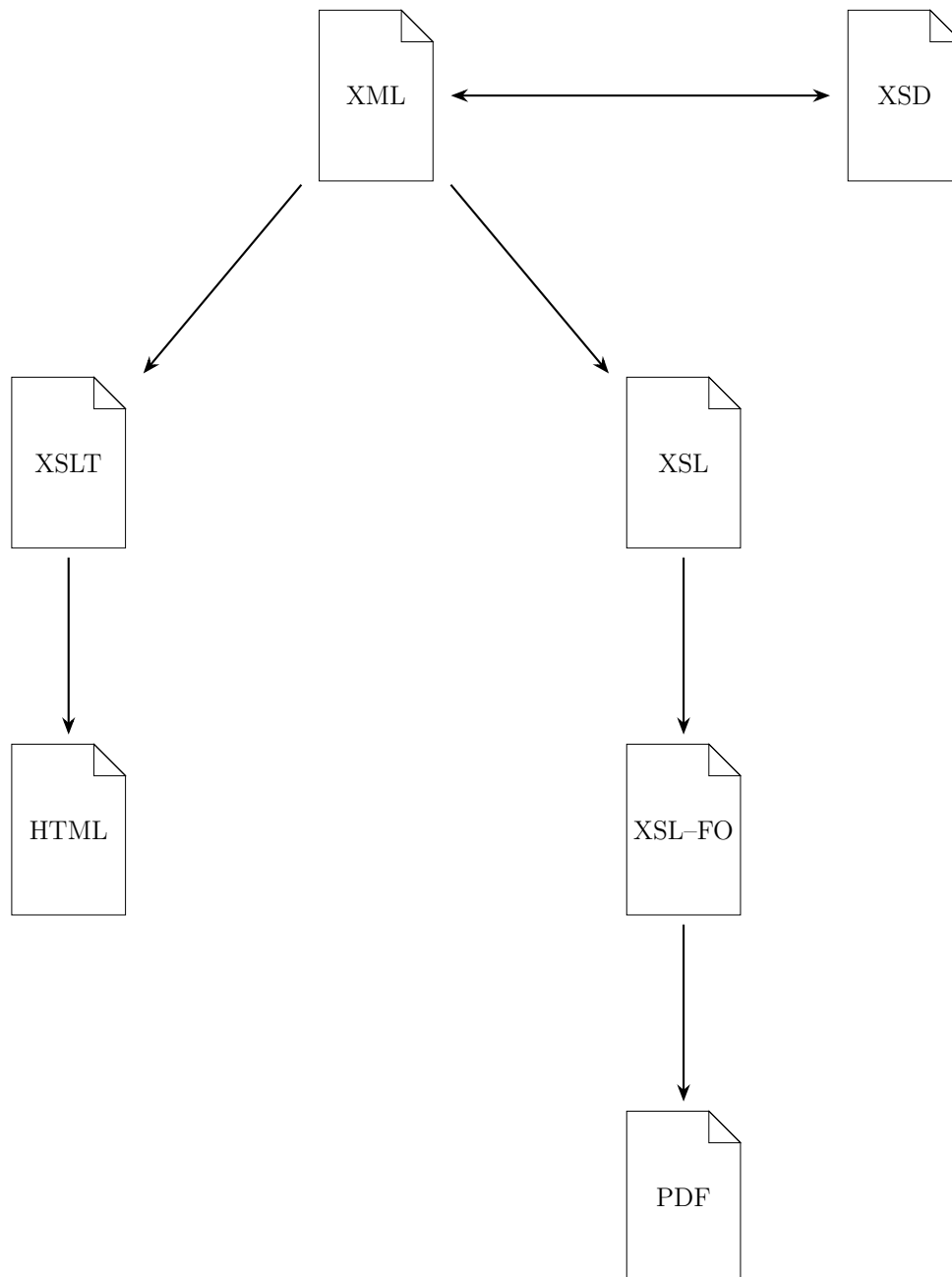


Abbildung 1: Transformations- und Validierungsbeziehungen zwischen XML-basierten Dokumentformaten

Für dieses Projekt wurde ein GitHub-Repositorium angelegt, dass diese Dateien enthält. Es gibt ein Verzeichnis `docs` mit der Dokumentation (als tex- und pdf-Dokument). Das Verzeichnis `xml-pipeline` enthält die für Abschnitt 2 und Abschnitt 3 relevanten Dateien. Der Ordner `programming` beinhaltet die Dateien für Abschnitt 4.

## 2 Datensatz

Der Datensatz beschreibt Produkte eines fiktiven Elektronikshops. Er liegt als `products.xml` vor.

Das zugehörige Schema befindet sich in `products.xsd`. Es definiert Struktur, Reihenfolge und Häufigkeit der verwendeten Elemente. Beim Entwurf des XSD wurde darauf geachtet, jedes Element über einen eigenen Typ zu kapseln, sodass spätere Änderungen gezielt und unabhängig vorgenommen werden können. Die verwendeten Datentypen wurden bewusst gewählt, um eine valide und konsistente Datenstruktur sicherzustellen. Standardwerte kommen dort zum Einsatz, wo Attribute fehlen, um sinnvolle und einheitliche Einstellungen zu gewährleisten. Feste Werte werden verwendet, um Abweichungen frühzeitig zu erkennen und als Fehler zu kennzeichnen.

## 3 Transformationen

### 3.1 XSLT von XML in HTML

Um die XML in HTML transformieren zu können, wurde `products.xslt` erstellt. Die tabellenförmige Ausgabe wurde getrennt für Widerstände, Spulen und Kondensatoren sowie für Dioden vorgenommen, da für erstere jeweils eine physikalische Größe angegeben ist, für letztere zwei physikalische Größen sowie der Untertyp.

Die physikalischen Größen selbst wurden mit einem gemeinsamen Template ausgewertet, da alle einen Wert und eine Einheit sowie einen optionalen Exponenten und eine optionale Toleranz aufweisen. Mit einer `xsl:when`-Abfrage wurde sichergestellt, dass bei Abwesenheit einer Toleranz dieses Feld mit einem Bindestrich gefüllt wird. Da das Feld für die physikalische Größe in der entsprechenden Tabelle jedoch auch ohne einen explizit vorgegebenen Exponenten korrekt ausgefüllt wird und alle weiteren Angaben in der Tabelle obligatorisch sind, ist dieses Vorgehen nur für die Toleranz notwendig.

Im Allgemeinen wurden die Templates so generisch wie möglich erstellt, um eine hohe Flexibilität für mögliche Veränderungen zu gewährleisten.

## 3.2 FOP-Transformation

Für die FOP-Transformation wurde hauptsächlich die XSLT für den HTML-Output wiederverwendet, da die gleichen Informationen mittels XPath ermittelt wurden. Es wurde lediglich die HTML-Syntax durch FO-Syntax ersetzt, zusätzlich wurde ein Master-Layout definiert.

Der einzige Unterschied besteht in der Behandlung des  $\Omega$ . Dieses musste im Vergleich zu HTML und zu den anderen Zeichen des Dokuments stets um 0.145em nach oben verschoben werden, da es sonst eine halbe Zeile zu tief steht.

## 3.3 Vergleich der Transformationen

Die erste Transformation wandelt eine XML in ein HTML-Dokument, wohingegen die zweite Transformation ein FO erzeugt, das als PDF gerendert werden kann.

Zudem sind die ersten beiden Transformationen sehr generisch – mit ihnen kann stets das angestrebte Output-Dokument erzeugt werden, sofern die Input-XML den in Abschnitt 2 erläuterten Anforderungen genügt.

# 4 Programmierung

# 5 Quellenverzeichnis