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

1 Projekübersicht

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 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 enthält. 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. Zunächst sollen die Daten auf einer HTML-Seite dargestellt werden. Dazu muss zunächst die XSLT-Transformationsdatei erzeugt werden. Darüber hinaus soll die Transformation in ein PDF erfolgen. Hierfür wurde die FOP-Transformation eingesetzt. Eine übersichtliche Darstellung dieser Transformationen zeigt Abbildung 1.

Der dritte Aufgabenteil beschäftigt sich mit der XML-Programmierung. Dabei soll eine Apllikation erstellt werden, welche die Validierung, Abfrage und Transformation der XML-Datei ermöglicht.

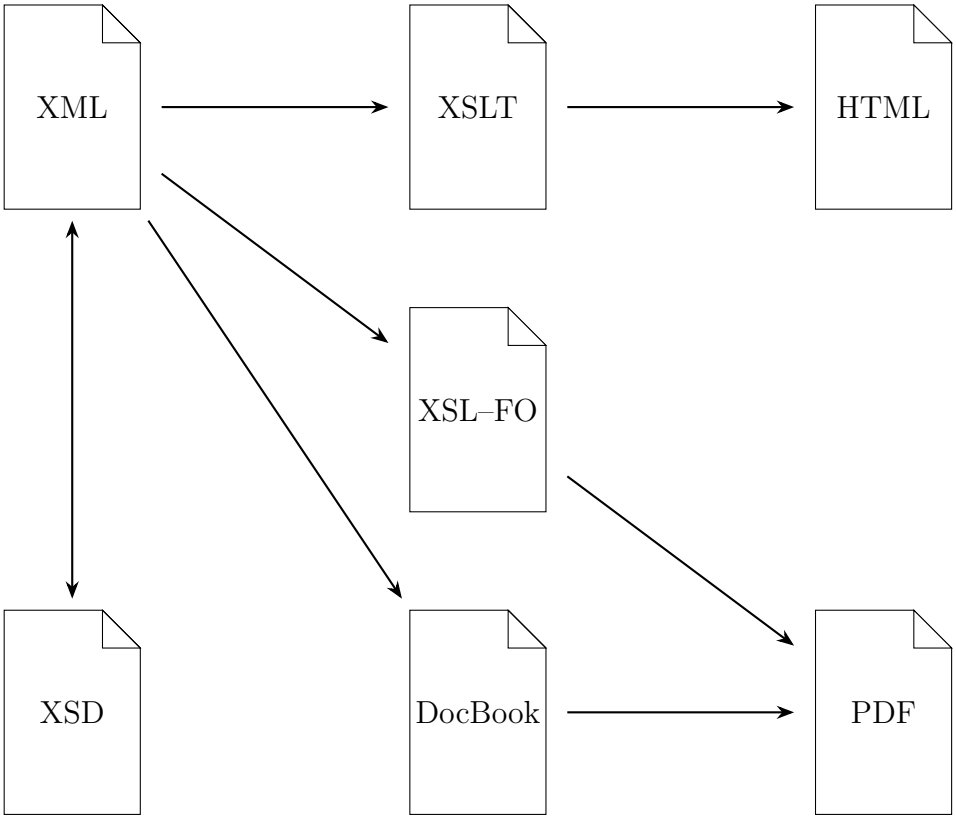
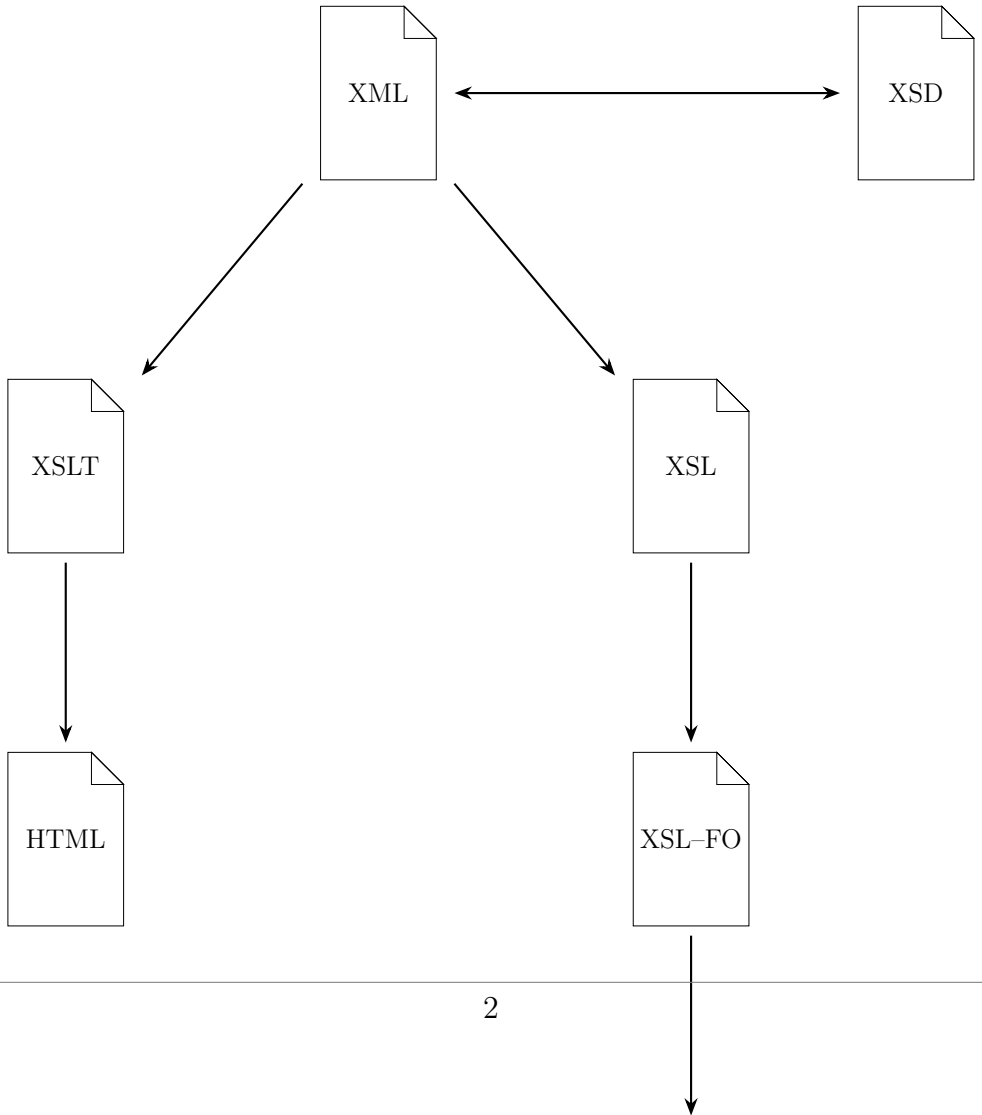


Abbildung 1: Übersicht der Dokumenttypen und Transformationspfade



1.2 Grundlegende Informationen

Für eine einfachere Bearbeitung dieses Projekts und zur Versionskontrolle wurde ein *GitHub-Repository* erstellt. Dieses vereinfacht insbesondere die Zusammenarbeit mehrerer Teilnehmer. Das *Repository* enthält alle angehängten Dateien und ermöglicht den Zugriff auf ältere Zwischenstände.

Abb.

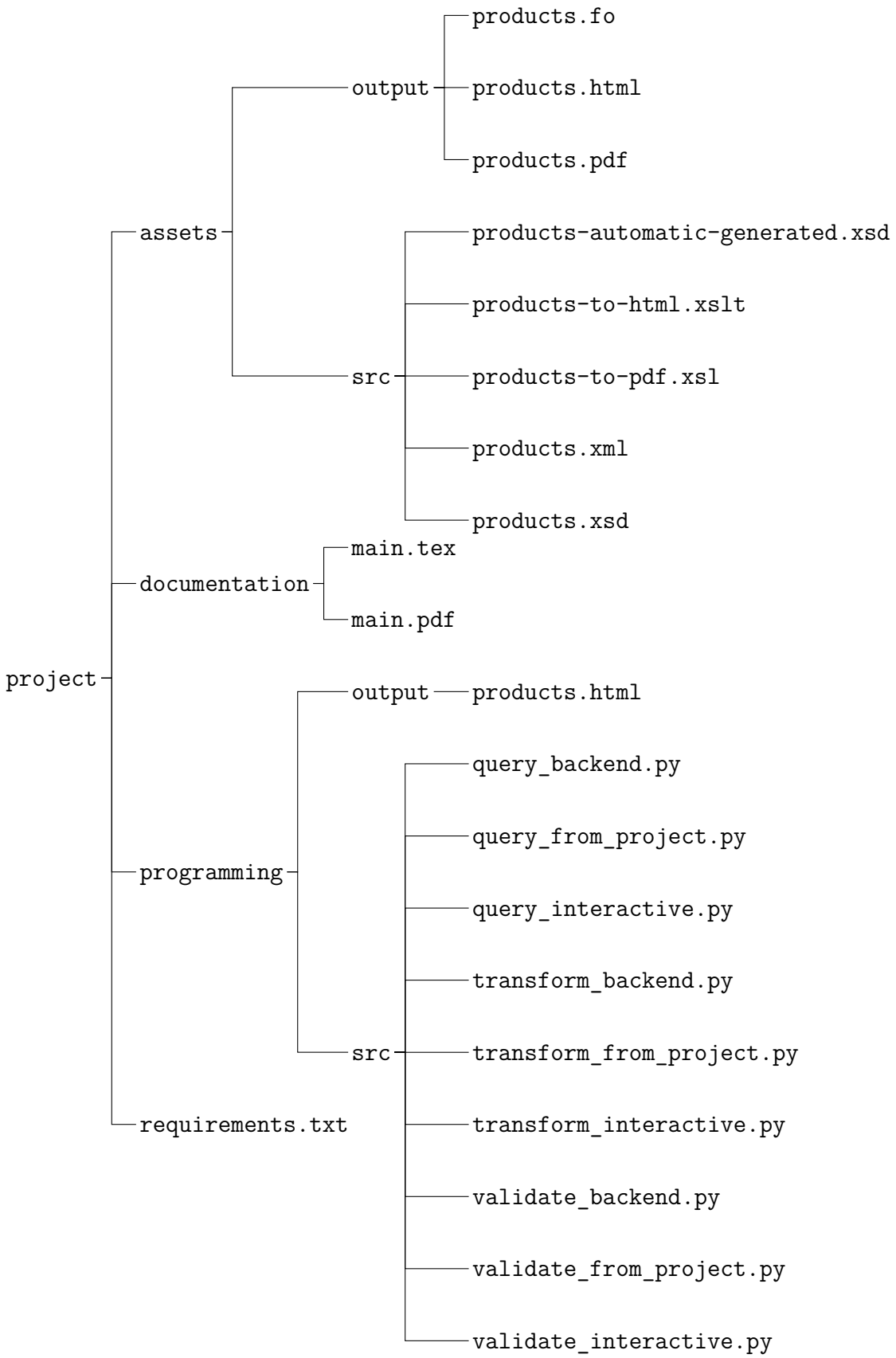


Abbildung 3: Projektordnerstruktur

Für dieses Projekt wurden folgende Quellen verwendet:

- Moodle-Unterlagen für das Modul Internettechnologien
- https://www.digikey.de
- https://www.mouser.de

2 Datensatz

Als Thema für den Beispieldatensatz wurden in diesem Projekt elektronische Bauteile eines fiktiven Elektronikshops gewählt. Der vollständige Datensatz dieses Projekts ist in der Datei *products.xml* enthalten. Er umfasst 9 Objekte, davon 3 Widerstände, 2 Spulen, 2 Kondensatoren sowie 2 Dioden. Hintergrund ist, dass die vorhandenen Daten auf einer Produkt-Website dargestellt werden können. Beim Inhalt des Datensatzes wurde sich deshalb auf die dafür wichtigsten Daten konzentriert. Das Root-Element ist *Products*, das alle *Product*-Elemente enthält. Diese können über den Eintrag im *id*-Attribut eindeutig identifiziert werden. Jedes dieser *Product*-Elemente umfasst die Elemente Anzahl, Preis und Hersteller sowie, abhängig vom mit dem *type*-Attribut spezifizierten Bauteil, eine oder zwei bauteilspezifische Datensequenzen. Diese beinhalten die physikalischen Größen des jeweiligen Bauteils, wobei der Wert, die Einheit und die Toleranz angegeben werden. Die Einheit Einheit??? % ist keine Einheit der Toleranz kann mit dem *unit*-Attribut eingestellt werden, ebenso die Einheit des Preises. Die Bezeichnung dieser Datensequenz ist für jedes Bauteil spezifisch. Während die Widerstände, Spulen und Kapazitäten jeweils nur eine spezifische physikalische Größe umfassen, können für Dioden mehrere angegeben werden. Diese konkrete Zuordnung der physikalischen Größen vereinfacht die anschließende Validierung der XML-Datei mit der XSD-Schemadatei.

Das zugehörige Schema befindet sich in `products.xsd`. Es definiert Struktur, Reihenfolge und Häufigkeit der verwendeten Elemente. Das Schema dient dazu, die Angaben in der XML-Datei auf formale Richtigkeit zu überprüfen. Damit kann sichergestellt werden, dass die geforderten Angaben in der XML-Datei gemacht wurden. Diese Richtigkeit ist für die Transformation in andere Datenformate unerlässlich. Außerdem kann überprüft werden, ob die angegebenen Datentypen korrekt sind. Die erstellte XSD-Datei dieses Projekts ist in *products.xsd* zu finden. Beim Entwurf des XSD-Schemas

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. Darüber hinaus wurde die Datei *products-automatic-generated.xsd* automatisch durch die Dokumentenerstellung mit *EditiX* erzeugt. Diese Datei ist jedoch nicht so präzise wie *products.xsd*.Es können nicht an allen Stellen die gewünschten Datentypen angegeben werden. Außerdem ist es nicht möglich, ein *choice*-Element zu verwenden. Es wird daher empfohlen, zur Validierung der XML-Datei die XSD-Datei *products.xsd* zu verwenden.

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 Ω. Dieses musste im Vergleich zu HTML und zu den anderen Zeichen des Dokuments stets um 0.145 em 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.

Beide Transformationen sind sehr generisch – mit ihnen kann stets das angestrebte Output-Dokument erzeugt werden, sofern die übergebene XML den in Abschnitt 2 erläuterten Anforderungen der Schemadatei genügt.

4 Programmierung

4.1 Grundlegende Informationen

Durch die XML-Programmierung sollen folgende Funktionalitäten implementiert werden:

1. Validierung eines XML-Dokuments gegen ein XSD-Schema

2. Abfrage von Informationen aus einem XML-Dokument

3. Transformation eines XML-Dokuments in eine HTML-Website

Zur Umsetzung dieser Funktionalitäten wird im Rahmen dieses Projekts die Programmiersprache *Python* verwendet. Um das die in den folgenden Kapiteln beschriebenen Befehle in der Kommandozeile ausführen zu können, muss *Python* von der offiziellen Website https://www.python.org unter dem Reiter *Downloads* installiert werden. Für dieses Projekt wurde Version 3.12.8 genutzt. Außerdem müssen die beiden Module *lxml* und *tkinter* installiert werden.

Durch das Modul *lxml* ist es möglich, diese Funktionalitäten umzusetzen.

Damit die Befehle über die Kommandozeile aufgerufen werden können, muss zunächst ein lokaler Python Download erfolgen.

HIER MUSS ICH ES MAL MIT MEINEM EIGENEN LINUX SYSTEM PROBIEREN!

Für alle 3 Teilaufgaben existieren 3 getrennte Python-Dateien:

1. Backend-Datei (`*_backend.py`): enthält ausschließlich die Logik der jeweiligen Aufgabe und können nicht direkt ausgeführt werden

2. Projektmodus-Datei (`*_from_project.py`): führt die jeweilige Aufgabe mit den im Projekt erarbeiteten Dateien aus

3. Interaktiv-Datei (`*_interactive.py`): ermöglicht die Auswahl beliebiger Dateien und führt die jeweilige Aufgabe für diese durch

Die jeweiligen Dateien enthalten den beschriebenen Code, der ausführlich kommentiert ist. Die Dateien können ausgeführt werden, indem man zunächst die virtuelle Umgebung aktiviert und in der Kommandizeile in das Verzeichnis `programming/src` wechselt und den Befehl `python desired_file.py` ausführt. Dabei ist `desired_file.py` ein Platzhalter für den eigentlichen Dateinamen, der an dieser Stelle natürlich eingesetzt werden muss.

4.2 Abfrage

Bei der Abfrage werden gezielt Informationen aus einem XML-Dokument extrahiert, indem XPath verwendet wird. Hierbei werden 3 verschiedene Abfragen durchgeführt:

1. Welche Produkt-IDs gibt es?

2. Wie heißen die Hersteller? (Hierbei sollen mehrfach aufgeführte Hersteller nur einmal aufgeführt werden.)

3. Was ist die Gesamtanzahl aller Widerstände im Lager?

Führt man die Datei `query_from_project.py` für den Projektmodus aus, dann wird in der Kommandozeile folgendes ausgegeben:

```
=== QUERY ===

Query 1: Retrieve all product IDs.
Result: ['R001', 'R002', 'R003', 'L001', 'L002', 'C001', 'C002',
        'D001', 'D002']

Query 2: Retrieve unique manufacturer names.
Result: ['Diodes Incorporated', 'Micro Commercial Components',
        'Murata Power Solutions', 'Samsung Electro-Mechanics',
        'Stackpole Electronics', 'Taiyo Yuden']

Query 3: Calculate the total number of resistors in stock.
Result: 864907
```

Möchte man die interaktive Abfrage durchführen, kann man über die Datei `query_from_project.py` eine XML-Datei aus dem Datei-Explorer auswählen. Anschließend wird zunächst überprüft, ob die angegebene Datei das vorgegebene XSD-Schema erfüllt. Ist dies nicht der Fall, können die Abfragen nicht durchgeführt werden und es wird eine Fehlermeldung ausgegeben. Ist die XML-Datei gültig, werden ebenfalls die 3 Abfragen getätigt und die Ergebnisse ausgegeben.

4.3 Validierung

Um eine XML-Datei gegen ein XSD-Schema validieren zu können, kommen die Dateien `validate_from_project.py` und `validate_interactive.py` zur Anwendung. Dabei kann in ersterem Fall die XML-Datei gegen die XSD-Datei aus dem gegebenen Projekt

validiert werden. Für die zweite Datei ist die Auswahl einer beliebigen XML- und XSD-Datei im Datei-Explorer möglich. Die Ausgabe in der Kommandozeile für eine gültige oder nicht gültige XML-Datei lautet folgendermaßen:

```
VALIDATION RESULT: products.xml is valid according to products.xsd
```

```
VALIDATION RESULT: products.xml is NOT valid according to products.xsd
```

4.4 Transformation

Die Transformation erfolgt über die Dateien `transfrom_from_project.py` und `transform_interact`
Dabei wird entweder die XML- und XSLT-Datei des gegebenen Projekts verwendet oder im interaktiven Modus werden diese beiden Dateitypen vom Anwender übergeben. Anschließend wird die Transformation in eine HTML-Datei durchgeführt und in den *output*-Ordner (in *programming*) gelegt, sodass sich der Anwender die entstehende HTML-Website anschauen kann. Im interaktiven Modus kann sich der Anwender den Zielordner selsbt aussuchen. In der Kommandozeile kommt der Hinweis:

```
HTML generated: products.html
```

Im Unterordner *programming/output* befindet sich bereits die HTML-Datei, die durch die durchgeführte Transformation für die im Projekt gegebenen Dokumente entstanden ist.