Controlling Information Structures with Schemas and XSLT

Tutorial

XML Europe 2001

21.5.2001, Berlin

Information | Doing | Basics | Motivation

Overview

- Motivation
  - What's the problem?
- Basics
  - Schemas and XSLT - short intro
- Doing
  - How to define constraints?
- Information
  - How to start?

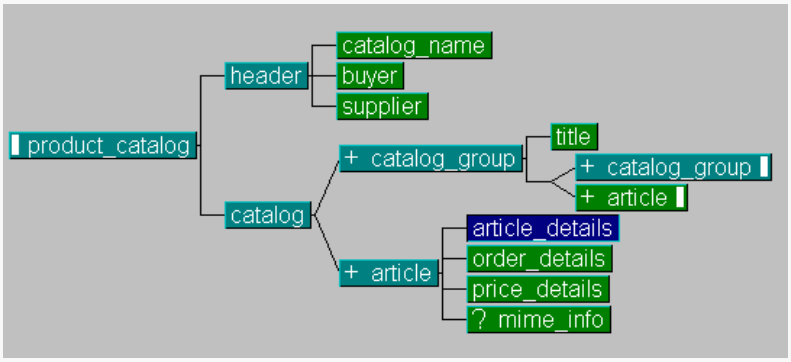
Motivation

What's the problem?
Examples

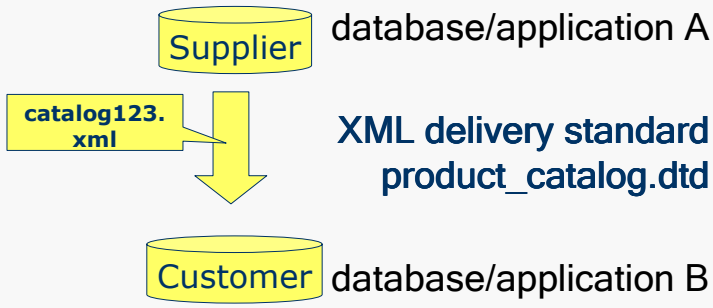
Example

Transforming a product catalog

DTD for Product Catalog



Scenario



Transformation Task

- The DTD is standardized (like BMEcat, or others)
- The catalogs must be transformed from the supplier structure to the structure of product_catalog.dtd and then will be imported to the target application

Problem

- Of course the DTD is used in order to check the result, ***but***
- The DTD does not express all the constraints of the delivery standard (e.g. Constraint 1)
- There exist some additional constraints in the target application (e.g. Constraint 2)

Constraint 1

- Some element contents are restricted, e.g. the child elements of <price_details>:
  - <amount> must contain a number
  - There exists a list of possible values for <currency>
  - <tax> must contain a number

Constraint 2

- <catalog_group> is recursive: The standard does not restrict the levels
- But the target application does! The group levels are restricted to 3

Solution - and New Problems

- One can define *additional constraints* in a transformation / conversion DTD
- *But:* Some restrictions
  - can not be defined in the DTD without changing the whole structure
  - can not be defined in a DTD at all

Some Other Example Constraints

- Attribute A and B are optional. But if attribute A is specified, attribute B must be specified, too
- The content model of Element C depends on the context: E.g. if it has parent D, attribute E is required
- If Attribute F has value 1, attribute G can only have value (2 | 3 | 4)

checking for tree relations

checking for contents

Result

What do we have?
What do we need?

We have...

- (Relational) databases for defining relationships, data types, and other constraints
- DTDs for defining tree structures, orders, frequencies, and other constraints
- Work-arounds: Define additional constraints within the processing programs

We have not...

- a language that combines the power of data models (databases) and the power of doctype models (DTDs)
- a language that allows contextual constraints (checking for tree relations; checking for contents; etc.)

We can solve this problem now

- by combining DTDs (i.e. XML validating) and individual checking (e.g. within the transformation program in the example scenario above)

But we want to do it better

- by separating *validation* from *processing* more strictly:
- defining constraints separated from data processing programs (as we do in DTDs, as we do in data models)
- Schemas or/and XSLT can be used for that purpose

Basics

Schemas and XSLT
Short Introduction

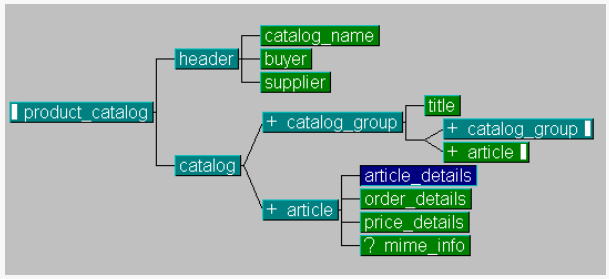
Schemas

Beyond DTDs

XML Schemas, RELAX, and others

What are Schemas?

- "Schemas define the characteristics of classes of objects"



- a DTD is a schema

What's New? The Language

- A declaration in a DTD:

```
<!ELEMENT article_details (description_short,
          description_long?, ean?, supplier_id) >
```

- A declaration in an XML Schema:

```
<xsd:element name="article_details">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="description_short" type="xsd:string"/>
            <xsd:element name="description_long" type="xsd:string" minOccurs="0"/>
            <xsd:element name="ean" type="xsd:string" minOccurs="0"/>
            <xsd:element name="supplier_id" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```


It's Instance syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v3.5 NT (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="article">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="article_details">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="description_short" type="xsd:string"/>
                            <xsd:element name="description_long" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="ean" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="supplier_id" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
```

What's new? The features

- Depending on the schema language; in general
  - Constraints on the element content (data types, and other constraints)
  - Advanced modelling features (e.g. derivation of elements and data types)

Schema Languages

- XML Schema Language
  - W3C Proposed Recommendation
  - two parts: structures and data types
- RELAX
  - Regular Language Description for XML
  - JIS Technical Recommendation
- And others
  - XDR: XML Data Reduced
  - TREX (James Clark)

XML Schema: Features (1)

- Constraints not only on the content models, but also on the content
  - Constraining Data types of element contents and attribute values
  - Defining Patterns for element contents and attribute values
  - ...

INFOTAKT information design

Information
Doing
Basics
Motivation

XML Schema: Features (2)

- Modelling features
  - re-use of content models
  - deriving elements and data types
  - defining abstract elements and data types
  - combining different schemas easily
  - ...

Example

- `<description_short>` must not contain more than 80 characters

```
<xsd:element name="description_short">
   <xsd:simpleType>
         <xsd:restriction base="xsd:string">
               <xsd:maxLength value="80"/>
         </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

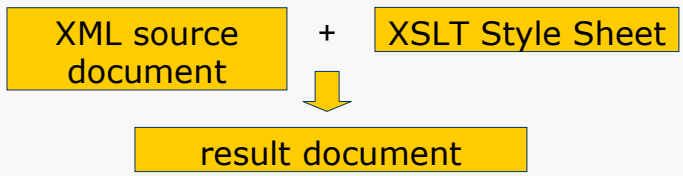

XSLT

XSL Transformations

XSL: EXtensible Stylesheet Language

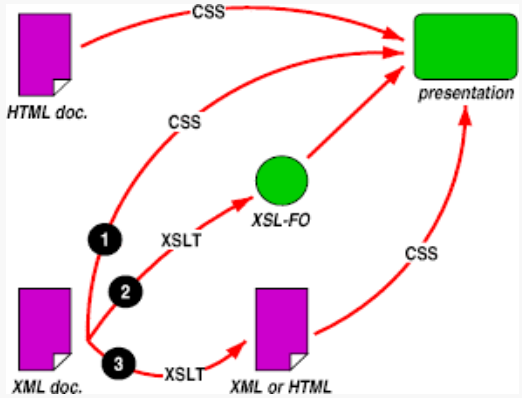
- a language for expressing stylesheets
- consists of two parts:
  - XSLT - XSL Transformations. A language for transforming XML documents
  - XSLFO - a vocabulary for specifyng formatting semantics

Transformations with XSLT



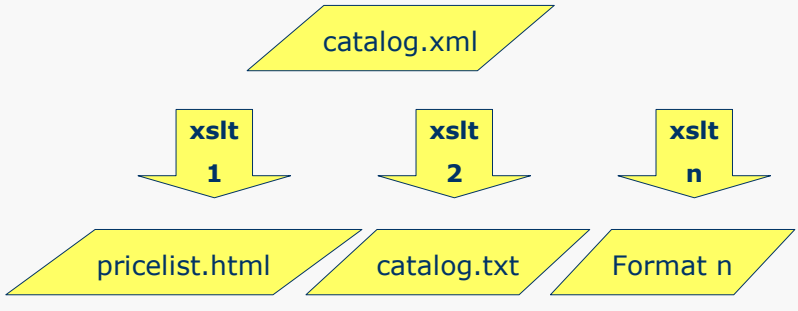
- The result document may be
  - XML, e.g. HTML
  - ASCII
  - ...

How XSLT can be used for publication



But XSLT can be used also...

- ... as a transformation language in general:



How does XSLT work?

- The source document is viewed as a tree
- The XSLT style sheet declares rules for nodes of this tree
- There are different types of nodes, like element nodes, attribute nodes, and text nodes
- The rules define how to transform a node

The rules

- Two parts
  - MATCH RULE
    - which node to match?
  - PROCESS RULE
    - how to transform that node?

Example: Task

catalog.
xml

```
<product_catalog>
  <header>
    <catalog_name>Soennecken</catalog_name>
</header>
  ...
</product_catalog>
```

cat2htm.
xslt

catalog.
htm

```
<html>
  <head><title>Soennecken</title></head>
  <body>...</body>
</html>
```


Example: Style Sheet

```
<!-- TEMPLATE -->
<xsl:template match="product_catalog">
  <!-- PROCESSING RULES -->
  <html>
    <head><title>
      <xsl:value-of select="header/catalog_name"/>
    </title><head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

XPath: XML Path Language

- a standard language for addressing parts of an XML document
- basic facilities for manipulation of strings, numbers, and booleans
- models an XML document as a tree of nodes

Look at the example above

```
<!-- NODE: <product_catalog> (root element) -->
<xsl:template match="product_catalog">
  <html>
    <head><title>
      <!-- NODE: <catalog_name>
        in <header>
        in <product_catalog>
      -->
      <xsl:value-of select="header/catalog_name"/>
    </title><head>
    ...
  </html>
</xsl:template>
```

Examples for XPath Expressions

- `chapter1`
  - all <chapter1> child elements of the current node
- `chapter1 | chapter2`
  - all <chapter1> and <chapter2> child elements of the current node
- `chapter1/title`
  - all child elements <title> of <chapter1> child elements of the current node
- `body//title`
  - all descendant elements <titel> of the child elements <body> of the current node
- `@day`
  - attribute "day" of the current node

How to Do It

Defining Constraints with Schemas and XSLT

Yes, Schemas. But XSLT?

- Schemas are made for defining constraints. Schema languages are alternatives to the DTD language
- But what about XSLT? I'ts for transformations, not for defining constraints!

Example

```
<!-- matching a tree node -->
<xsl:template match="catalog_group">
  <!-- checking for a constraint -->
  <xsl:if test="count(ancestor::catalog_group)>=3">
   <!-- answering with an error message -->
    too many group levels!
  </xsl:if>
  <xsl:apply-templates/>
</xsl:template>
```

Remember the example constraints

- <amount> and <tax> must contain a number
- There exists a list of possible values for <currency>
- The <catalog_group> levels are restricted to 3
- Attribute A and B are optional. But if attribute A is specified, attribute B must be specified, too
- The content model of Element C depends on the context: E.g. if it has parent D, attribute E is required
- If Attribute F has value 1, attribute G can only have value (2 | 3 | 4)

For some of them we need…

- constraints like data types, i.e. constraints not only for content models, but also for content
- check for tree relations: If node A is like this, node B must be like that
- know the document instance: If node C has value 1 (not part of an enumeration list!), node D must be like this...
- ...

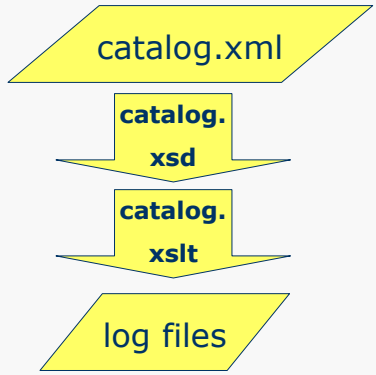
Concepts

Schemas	XSLT
Defining a document *type*	Transforming *instances* of a document type
Defining the structure of the *document type* tree	Checking the *document instance* tree by "walking through" it and finding *tree patterns*
Defining a *grammar*	Building *tree traversion* rules

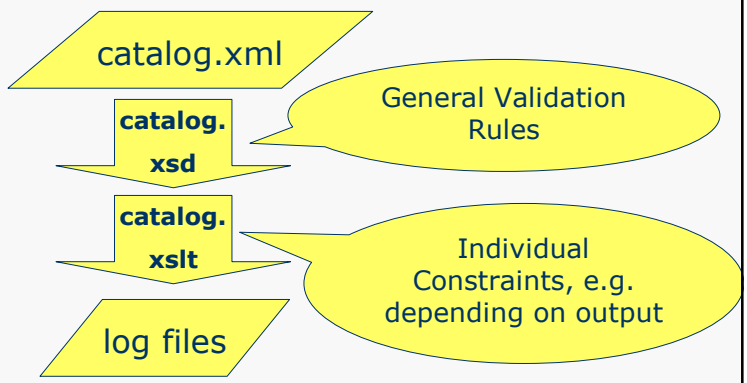
Result and Preview

- Schemas and XSLT have completely different approaches. They can enhance each other.
  - Schemas: Defining constraints on structure, semantics, and content
  - XSLT: Checking for constraints by matching and comparing tree nodes
- There are some rules that can't be defined within existing schema languages (although they should?)

Used in Combination



The Ideal Combination



Example

The product catalog

Remember Constraint 1

- Some element contents are restricted, e.g. the child elements of <price_details>:
  - <amount> must contain a number
  - There exists a list of possible values for <currency>
  - <tax> must contain a number

Constraint Definition in a Schema

```
<xsd:element name="amount" type="xsd:decimal"/>

<xsd:element name="currency">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Euro"/>
      <xsd:enumeration value="USD"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```


Remember Constraint 2

- <catalog_group> is recursive: The standard does not restrict the levels
- But the target application does! The group levels are restricted to 3

Constraint Definition in XSLT

```
<!-- matching a tree node: <catalog_group> -->
<xsl:template match="catalog_group">
  <!-- checking for ancestors
       <catalog_group> in the tree -->
  <xsl:if test="count(ancestor::catalog_group)>=3">
   <!-- output an error message
        in the target file -->
    too many group levels!
  </xsl:if>
  <xsl:apply-templates/>
</xsl:template>
```

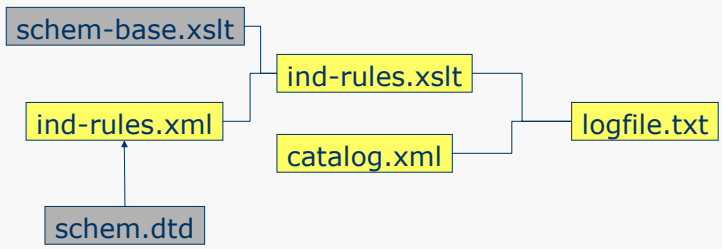

A Useful Language

Schematron

Schematron

- The Schematron Assertion Language. Specified by Rick Jellife
- "A feather duster for the furthest corners of a room where the vacuum cleaner (DTD) cannot reach." (Rick Jellife)

Processing Architecture



- + some variants
  - replacing schem-base.xslt, resulting in an other error output, e.g. html

How to define the rules

- The constraints are defined as an XML instance (e.g. ind-rules.xml)
- There can be defined
  - <assert>ion rules: What happens if the source file is allright according to the rule?
  - <report>ing rules: What happens if the source file is wrong according to the rule?

Remember Constraint 2

- <catalog_group> is recursive: The standard does not restrict the levels
- But the target application does! The group levels are restricted to 3

Constraint Definition in a Schematron Instance

```
<pattern name="catalog group levels">
  <rule context="catalog_group">
    <report
        test="count(ancestor::catalog_group)>=3">
      too many group levels!
    </report>
  </rule>
</pattern>
```


Information

How to start?
Hints and References

How to start? Recommendations. Step 1: DTD

- Of course: Find out your requirements. Standards are fine. Ideas are fine. But can it be helpful to *you*? If yes:
- Write a DTD. Start small

- Useful Tools:
  - ASCII Editor
  - Near&Far Designer 3.0 www.nearfar.com

How to start? Recommendations. Step 2: Schema

- Convert the DTD to a Schema (using e.g. XML Authority or XML Spy)
- Enhance the constraints within the schema, e.g. for data types of PCDATA contents

- Useful Tools:
  - XML Spy www.xmlspy.com
  - XML Authority www.xmlauthority.xom
- Useful Information:
  - XML Schema Tutorial www.infotakt.de/literatur.htm

How to start? Recommendations. Step 3: XSLT

- If some constraints are missing that require to check for tree patterns: Write an XSLT style sheet or a schematron instance.

- Useful Tools
  - XMLSpy www.xmlspy.com
  - Schematron www.ascc.net/xml/resource/Schematron/
- Useful Information
  - XSLT Tutorials on the Web
  - Schematron Examples www.zvon.org/HTMLonly/SchematronTutorial/General/contents.html

Other Information

- Schema Languages
  - W3C Proposed Recommendation
    - www.w3.org/TR/xmlschema-0/
    - www.w3.org/TR/xmlschema-1/
    - www.w3.org/TR/xmlschema-2/
  - RELAX JIS TR
    - www.xml.gr.jp/relax/
  - XDR (XML Data Reduced)
    - www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm
  - TREX
    - www.coverpages.org./trex/trexAnnounce.html
    - www.thaiopensource/trex
- XSLT W3C Recommendation
  - www.w3.org/TR/xslt/

Discussion

- You are welcome!
  - Have a talk at the conference
    21 - 23 April (wednesday)
    +49-173-9270185
  - Contact me later - by e-mail or phone
    ulrike.schaefer@infotakt.de
    +49-931-27049776

INFOTAKT information design

Thank you for your interest!