



XML

# XML

**Prof. Dr.-Ing. Martin Gaedke**

Technische Universität Chemnitz

Fakultät für Informatik

Professur Verteilte und Selbstorganisierende  
Rechnersysteme

<http://vsr.informatik.tu-chemnitz.de>



# Chapter 7

## **EXTENSIBLE STYLESHEET LANGUAGE (XSL)**



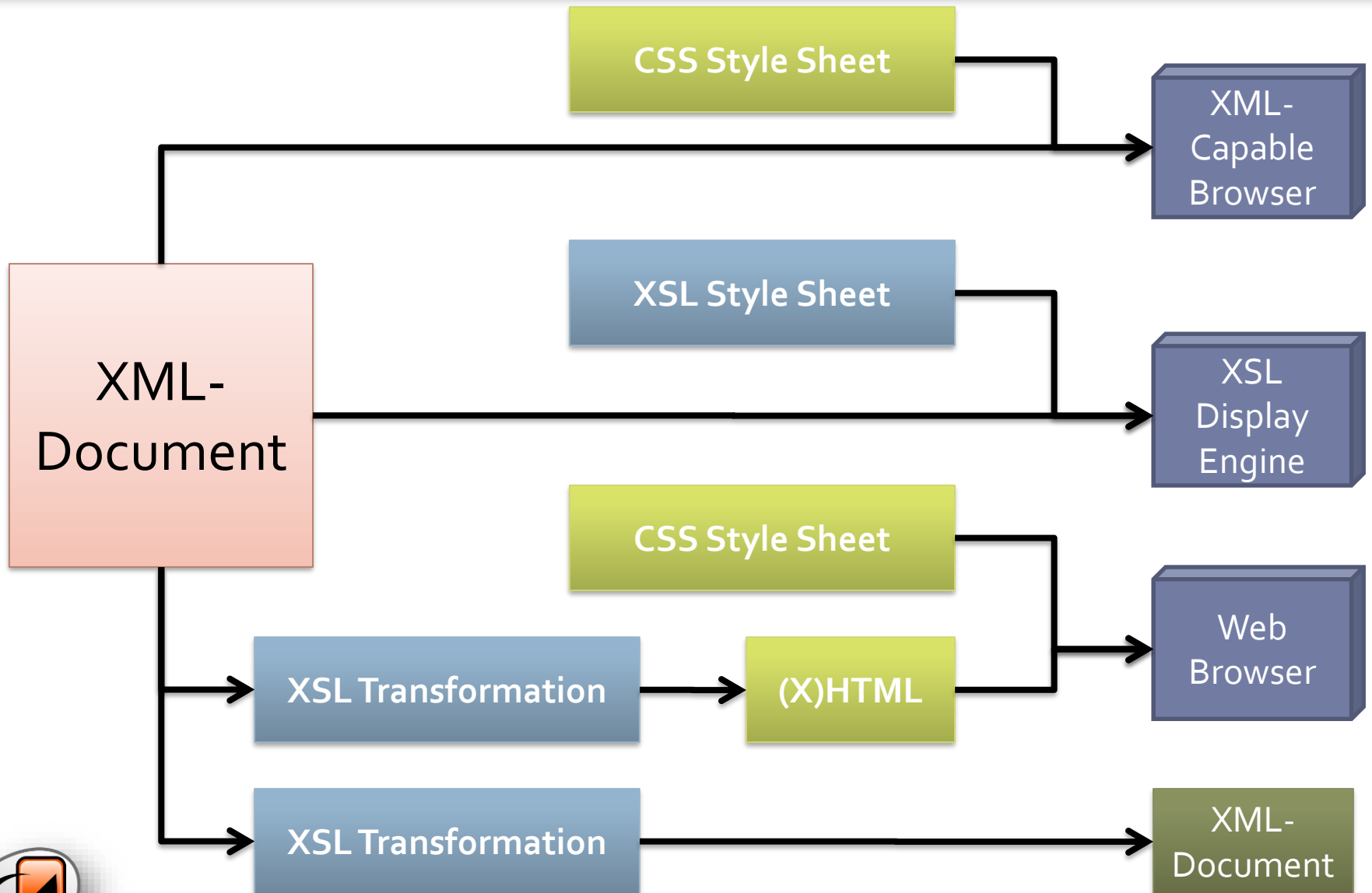
# XSL – Motivation

---

- Extensible Stylesheet Language (XSL)
  - W<sub>3</sub>C Recommendation
  - Focus: How can XML applications be represented?
- Initial situation / requirements:
  - XML application has content
  - Could require: Different representation descriptions (colors, text sizes etc.), different output formats (HTML, PDF, etc.), content reuse and representation descriptions.



# XML in Representation Context



# Example

- From personal data to postal address...

```
K07-01.xml
<?xml version="1.0" encoding="utf-8"?>
<Person>
  <Vorname>Peter</Vorname>
  <Nachname>Pater</Nachname>
  <Tel>123</Tel>
  <Fax>888</Fax>
  <PW>pass123</PW>
  <M>p@ter.com</M>
  <PLZ>09111</PLZ>
  <Strasse>Ahorn Weg 3</Strasse>
  <Ort>Chemnitz</Ort>
</Person>
```

?

```
-01.xml K07-02.xml
<?xml version="1.0" encoding="utf-8"?>
<Anschrift>
  <Vorname>Peter</Vorname><Nachname>Pater</Nachname>
  <Strasse>Ahorn Weg 3</Strasse>
  <PLZ>09111</PLZ><Ort>Chemnitz</Ort>
</Anschrift>
```

# XSL – Specification

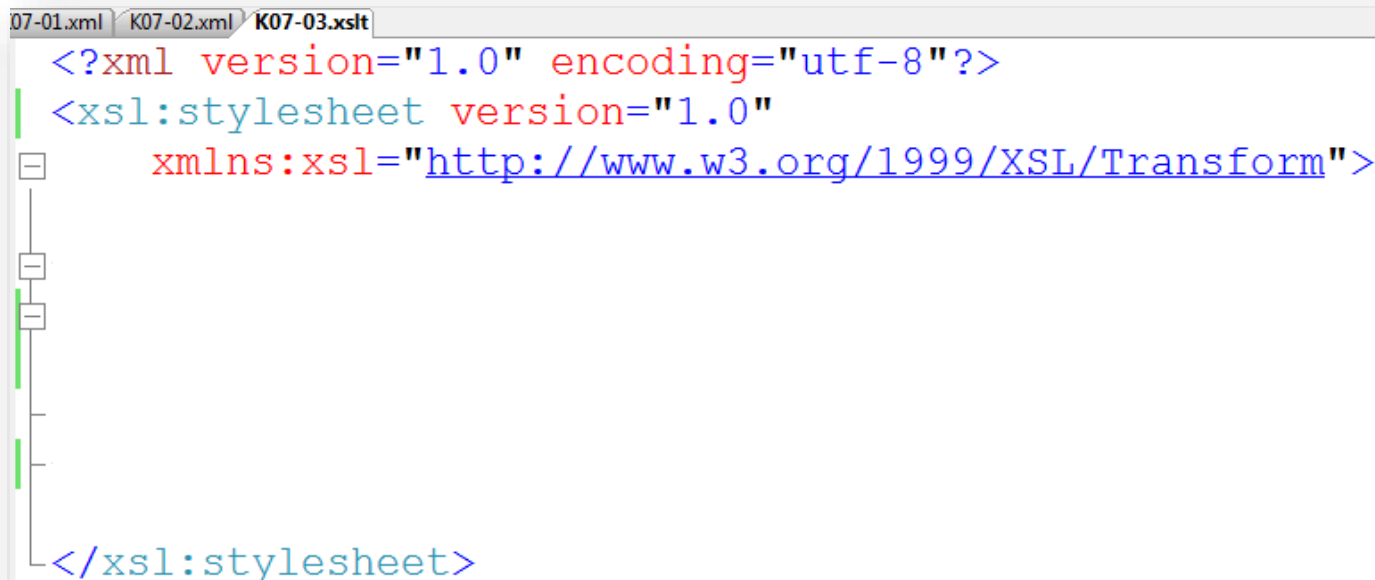
---

- XSL is a family of specifications for representing the contents of XML applications
- XSL contains three areas
  - **XSL Transformations (XSLT)**
    - Language for XML transformation
  - **XML Path Language (XPath)**
    - Language for formulation of embodiments referencing areas (Nodelists) or targets (Nodes) in an XML document
  - **XSL Formatting Objects (XSL-FO)**
    - Language describing formatting/representation directives (is introduced in presentation context)
- Languages from these three areas are used in **XSL Stylesheets**



# Stylesheets Specification

- Root-Element: ***stylesheet***
  - ***transform*** can be used as a synonym
  - Processing/transformation can be executed on the server- or client side



```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

</xsl:stylesheet>
```

# XSL Stylesheets

---

- **XSL Stylesheets** consist of a set of template rules (XSLT rules)
  - They describe the build-up of an XML document in the target representation via XSL transformations
- **Process:**  
**Input vocabulary → Processing → Output**
  - Input vocabulary:
    - **XML Source Tree**
  - Processing:
    - Constructs an **XML Result Tree** (target vocabulary) via XSL transformations (Result Tree is the result of XSLT processing)
  - Output: Formatting of XML Result Tree
    - Does not necessarily produce an XML document, is processed by a **Formatter**, for example, for serializing the XML Result Tree





# XSL Transformations (XSLT)

- **XSLT** – Language describing transformations of XML documents
  - W3C-Standard since 1999: <http://www.w3.org/TR/xslt>
  - Platform independent, since is an application of XML itself
  - Concept: Nodes of a Source Tree are processed using template rules to produce fragments of the Result Tree
  - Processing of all template rules leads to the final Result Tree

- **Template rule:**

```
<xsl:template match="sourceelement">  
  <h1>Representation</h1>  
</xsl:template>
```

- **Processing:**
- Input: <sourceelement/> (Source Tree)
- Intermediate result: <h1>Representation</h1> (Result Tree)
- Final result: <h1>Representation</h1> (Formatting)



# XSL Template Rules

---

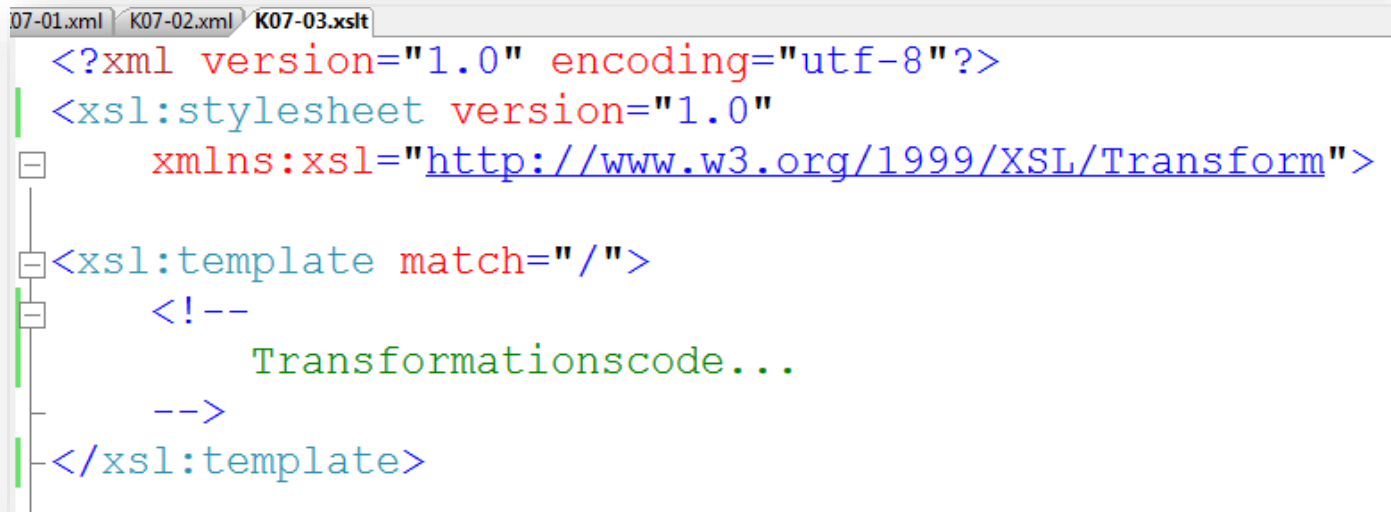
## ■ Processing model

- List of nodes (NodeList) of a **Source Tree (ST)** is processed to create a **Result Tree (RT)**
- RT is created by processing a NodeList which contains the root of the ST (all transformations start with this rule)
- NodeList of the ST is processed sequentially according to the nodes in the list, the processing results are attached to the RT
- Node of the ST is processed according to the template rule which best describes the node in the ST (node description by sample / pattern)
  - The chosen template rules are instantiated
  - The current ST node is forwarded as a **current node** and a NodeList of the currently chosen ST Node - as **current node list**
  - The directives of the instantiated template rules are executed (which, in turn, leads to execution of further template rules)
- Such recursive processing will carry on until no new nodes are chosen in the ST



# XSL Templates <xsl:template>

- **template-Element** specifies template rules
  - **match-Attribute** provides a sample, which identifies root node or nodes in the Source Tree for template rule application (instantiation)



The screenshot shows a code editor with three tabs: 07-01.xml, K07-02.xml, and K07-03.xslt. The active tab, K07-03.xslt, contains the following XSLT code:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <!--
      Transformationscode...
    -->
  </xsl:template>
```

# XSL Templates Instantiation

- Template instantiation
  - **Template content:** Template content contains transformations to be executed and further possible processing rules
  - It is applied to the Current Node and has access to the Current Node List
  - Result of the execution is attached to the Result Tree
- Some further processing rules:
  - `<xsl:apply-templates/>`
  - `<xsl:apply-templates select="PATTERN"/>`
  - `<xsl:for-each select="PATTERN">`
  - See <http://w3.org/TR/xslt>



# XSL Template Rules

---

- **Node description** by a sample (pattern)
  - Pattern describes nodes in ST
  - Pattern is applied to the Current Node to find further nodes in ST
  - Pattern is defined by a language



# Chapter 8

## **XML PATH LANGUAGE (XPath)**



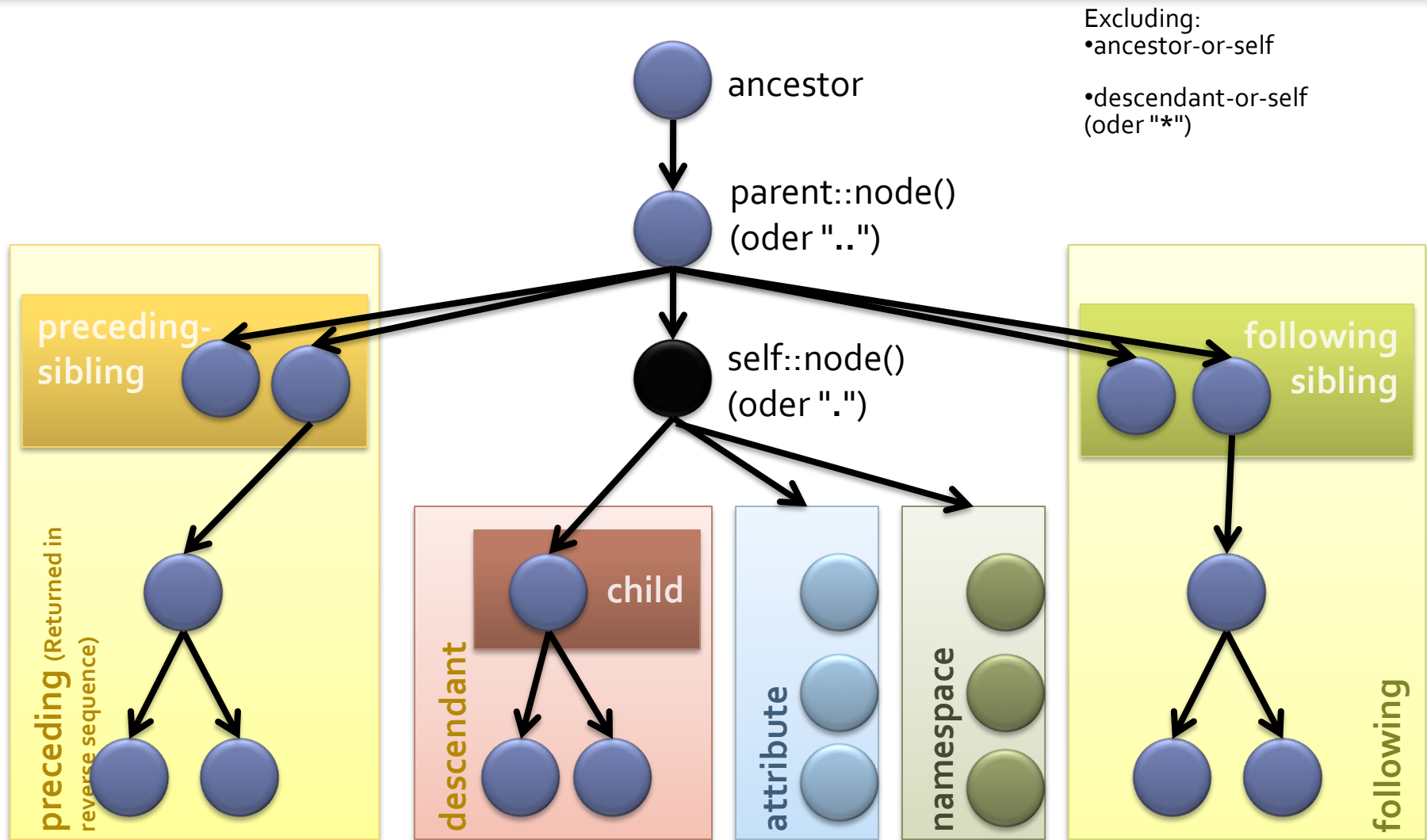
# XPath Introduction

---

- XML Path Language (XPath)
  - XPath is a language for addressing parts of an XML document
  - W3C Recommendation since 16. November 1999 (XPath 1.0)
  - XPath 2.0 – Recommendation since 23. January 2007
  - XPath is used in other standards, such as XLink, XPointer, XQuery, XSL – example match in XSLT
- XPath enables
  - Node selection in a XML document
  - Node condition observation
  - Result of an Xpath request is a Node set or (since XPath 2.0) any sequence of values allowed by the Xpath data model (XDM)
- Here we'll concentrate on XPath 1.0



# XPath Axis (Navigation Axis)





# XPath – Expressions

---

- So-called **Xpath Expressions** lie at the base.
  - Expressions are assigned to an object, which has one of the four types: **node-set**, **boolean**, **number**, **string**
- Expression evaluation is always regarded in context
- The language used (such as XSLT) defines how the context is determined
  - Contexts are nodes, variable bindings
  - The most important expression is **LocationPath** (**path expression**)



# Simple Path Expressions

- Example:

```
<cupboard>
  <plate price="12.00">A</plate>
  <plate price="18.00">B</plate>
  <plate price="18.00">C
    <plate price="1">D</plate>
  </plate>
</cupboard>
```
- **cupboard**
  - Addresses all **<cupboard>** nodes in the current context
- **cupboard//plate**
  - Addresses all **plates** within **cupboard**
- **cupboard/plate**
  - Addresses all direct children called **plate** in **cupboard** context
- **//plate**
  - Addresses all **plates** in the whole document
- **./plate**
  - Addresses all **plates** within the current nodes (context)



# Syntax

Syntax (short)	Syntax (long)	Meaning
ename	<b>child::</b> ename	All child nodes with the name ename (in current context)
.	<b>self::</b> node()	Current Nodes
..	<b>parent::</b> node()	Parent Nodes (in current context)
*	<b>descendant-or-self::</b>	All children or Current Nodes
/		Root Node
//		All children of the Root Node
[filter]		Filter attached to the Path Expression. Only holds for the Elements that adhere to the filter.
[n]		Filter chosen by the n-th Element in the Path Expression
@attr	<b>attribute::</b> attr	Attribute called attr of the current Element

# Further Path Expressions

- Example: `<cupboard>`  
    `<plate price="12.00">A</plate>`  
    `<plate price="18.00">B</plate>`  
    `<plate price="18.00">C`  
        `<plate price="1">D</plate>`  
    `</plate>`  
    `</cupboard>`
- **`cupboard/*/plate`**
  - Addresses all `<plate>` nodes, which are grandchildren of the cupboard
- **`cupboard|plate`**
  - Addresses all cupboard or plate elements



# Node Test

Typical Node Test	Restriction
*	Wildcard (no restriction)
Name	All children with Name
node()	Returns all sub-elements
text()	Returns all text sub-elements
comment()	Returns all comment sub-elements
document-node()	Document nodes

- Example:
- `<xsl:value-of select = "text()" />`
- Delivers the text of the current content, i.e.  
`<nodes> Test </nodes> → Test`



# Predicates

- Predicates allow node test constraints (so-called filter)
  - Multiple predicates can be applied via chaining
  - All results are converted to boolean values
- Relationship to path expressions
  - **Path expression** consists of navigation axis, node test, predicates
  - `child::plate[position()=last()]`
  - *child* is the navigation axis, *plate* is the node test and *[position()=last()]* is the predicate
- Examples
  - **plate[@price]**
    - Addresses all plates, which have a price attribute
  - **plate[@price="18.00"]**
    - Addresses all plates, which have price attribute with a value of 18.00
  - **cupboard/plate[1]**
    - Addresses the first element **plate** in the cupboard context



# XPath Functions

XPath Function	Meaning
number <b>last()</b>	Size of the current context set
number <b>position()</b>	Current position in the current context set
number <b>count(node-set)</b>	Number of nodes in the node set
node-set <b>id(object)</b>	Returns nodes with the provided object-ID
string <b>name(node-set?)</b>	Returns the name of the node set, where element and attributes are of expanded-name (with namespace)
boolean <b>starts-with(string1, string2)</b>	Returns True, if string1 starts with the value of string2

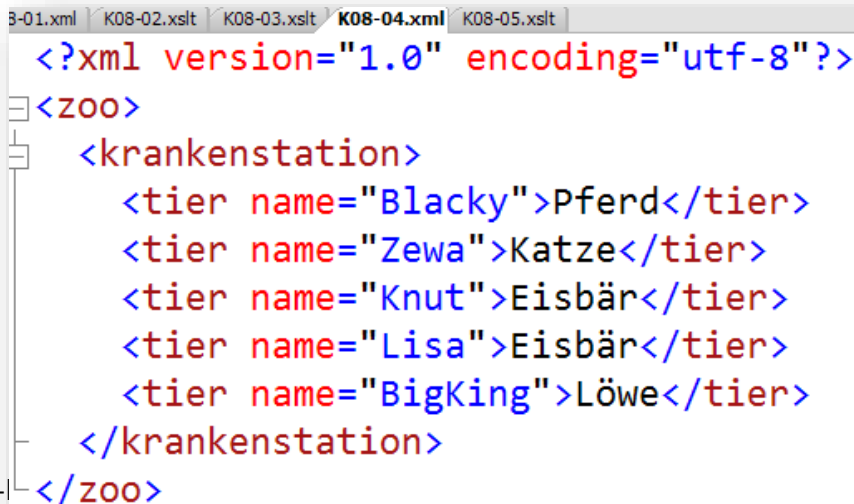
- Many further functions
  - Function areas: Node set, String, Boolean, Number
  - Examples: contains, substring, concat, not, true, false, number, sum, floor, ceiling



# XSLT and XPath in Action

## ■ Idea:

- XML data as input
- XSL representation approach in HTML
  - With recursion and
  - Iteration as template approach
  - Additionally, sort, count etc.



The screenshot shows an XML editor with a tab labeled 'K08-04.xml'. The XML content is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<zoo>
  <krankenstation>
    <tier name="Blacky">Pferd</tier>
    <tier name="Zewa">Katze</tier>
    <tier name="Knut">Eisbär</tier>
    <tier name="Lisa">Eisbär</tier>
    <tier name="BigKing">Löwe</tier>
  </krankenstation>
</zoo>
```

Demo



# Final Remarks

- XPath 2.0
  - Focus: Use of the data model (XPath 2.0 Data Model (XDM))
    - Check differences here: <http://www.w3.org/TR/2010/REC-xpath20-20101214/xpath20-diff-from-REC20070123.html>
  - Example:
    - `<eur>10</eur> <eur>10.0</eur> <eur>10.00</eur>`
    - Search for particular eur-values
    - Solution in the text world  
`eur = '10' || eur = '10.0' || eur = '10.00'`
    - Solution in the data model world  
`number(eur) = 10`
  - XPath 2.0 can additionally process sequences and sets
- Problem: There still exist only few implementations of the recommendation.



# Chapter 9

# **XPOINTER**



# Motivation

---

- Motivation <http://www.w3.org/TR/NOTE-xlink-principles>
- Need for Anchors (cf. HTML Anchor) in XML
- XPointers address into XML documents
  - XPointers shall be straightforwardly usable in URI's
  - The XPointer syntax shall be reasonably compact and human-readable



# XPointer

---

- XML Pointer Language (XPointer)
  - W3C Last Call Working Draft 8 January 2001
  - W3C Candidate Recommendation 11 September 2001
  - Language to be used as the basis for a fragment identifier for an XML resource
  - W3C Working Draft 16 August 2002 → XPointer document has been superseded!
- Now: XPointer Framework



# XPointer Framework

---

- XPointer Framework
  - W3C Recommendation 25 March 2003
  - <http://www.w3.org/TR/xptr-framework/>
  - Supports: shorthand and scheme-based pointer, namespace binding
- Example for `http://server/resource.xml` append:
  - `#xpointer(/order/price)`
  - `#xpointer(id('orderId')/price[1])element(//price)`



# Chapter 10

## **XML BASE**



# XML Base

---

## ■ XML Base

- W3C Recommendation 28 January 2009
- Tool for defining base URIs for XML document parts
- Tool approach is realized via attributes

## ■ **xml:base** Attribute

- Attribute can be added to an XML document to explicitly set the base URI (unless it already corresponds to the existing base URI of the document in question), see RFC 3986 w.r.t. URI



# xml:base Example

- `<doc xml:base="http://example.org/abc/">`
  - `<a>`
    - `<link href="new.xml">A-News</link>!`
  - `</a>`
  - `<b xml:base="/news/">`
    - `<link href="new.xml">B-News</link>!`
  - `</b>`
- `</doc>`
- A-News
  - → `http://example.org/abc/new.xml`
- B-News
  - → `http://example.org/abc/news/new.xml`

