



## **XSLT & XSLT 2.0**

Prof. Dr.-Ing. Robert Tolksdorf  
Freie Universität Berlin  
Institut für Informatik  
Netzbasierende Informationssysteme  
[tolk@ag-nbi.de](mailto:tolk@ag-nbi.de)

---

# Wie geht es weiter?

letzte Woche

☑ XSLT

heutige Vorlesung

- XSLT-Editoren
- mehrere Ursprungsdokumente
- Sortieren
- Gruppieren
- XSL-FO
- XSLT 2.0

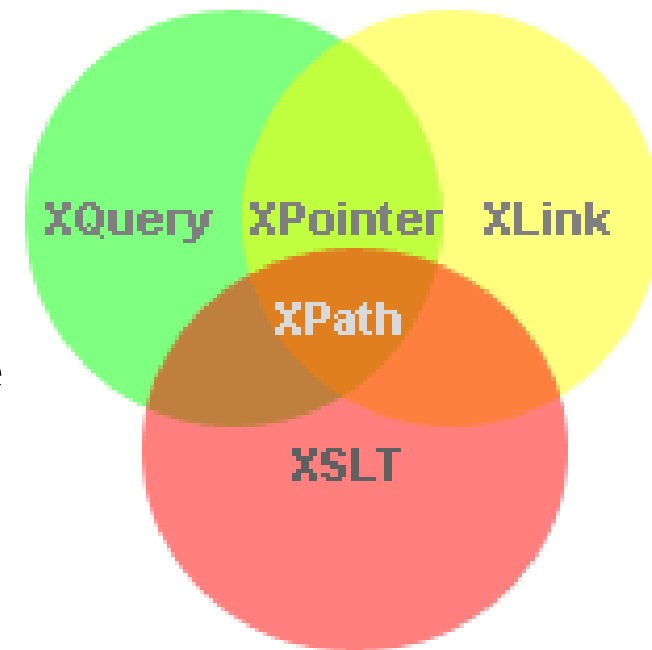


Bild-Quelle: <http://www.w3schools.com/xquery/default.asp>



**XSLT Tools...**

---

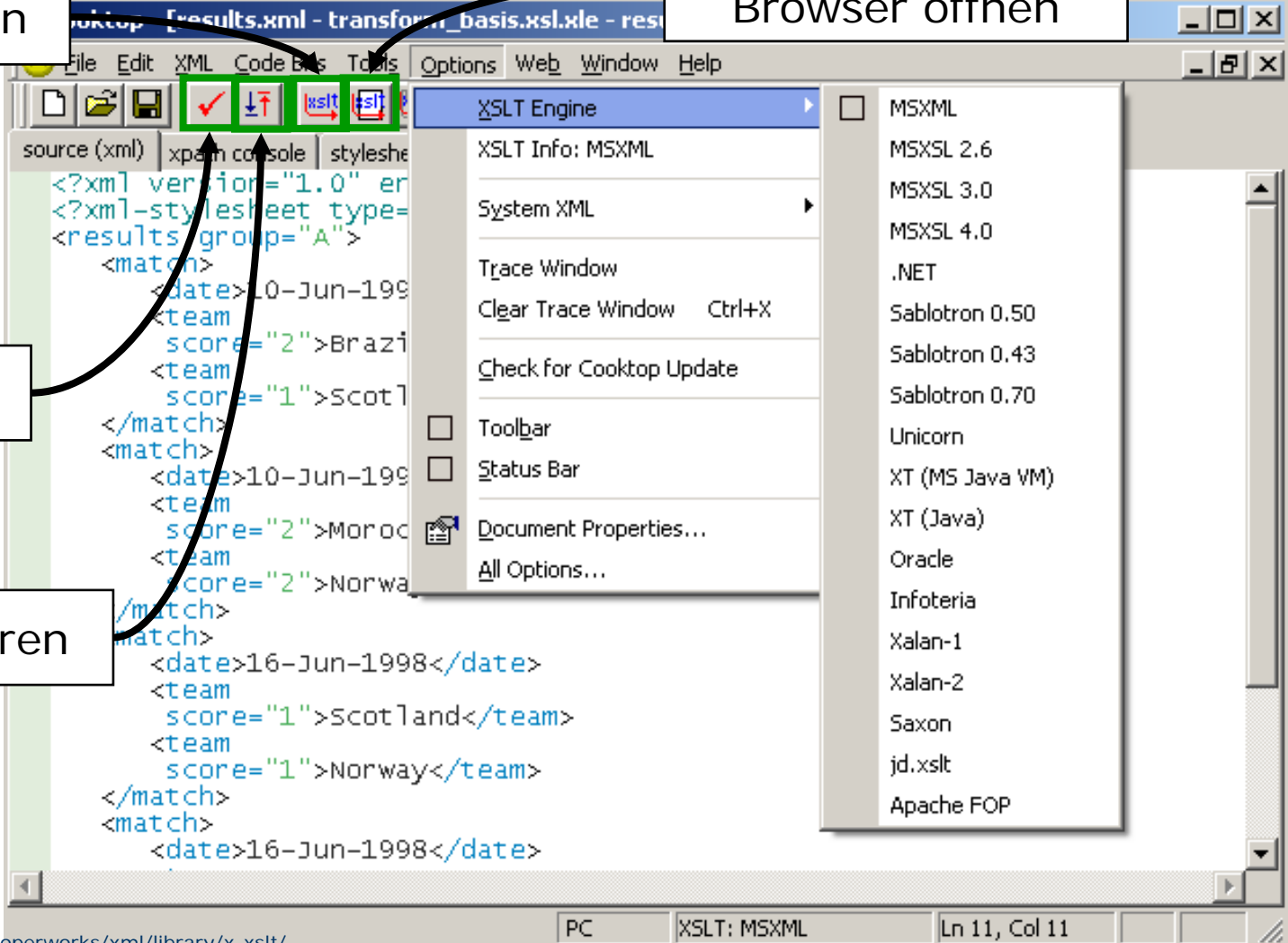
- CookTop:
  - text-basiert
  - Free Software
  - Download: <http://xmlcooktop.com/>
  
- xsl:easy:
  - Drag & Drop (Textbearbeitung nur bedingt möglich)
  - Download Trial: <http://xsl-easy.com/>



**CookTop**

---

# CookTop



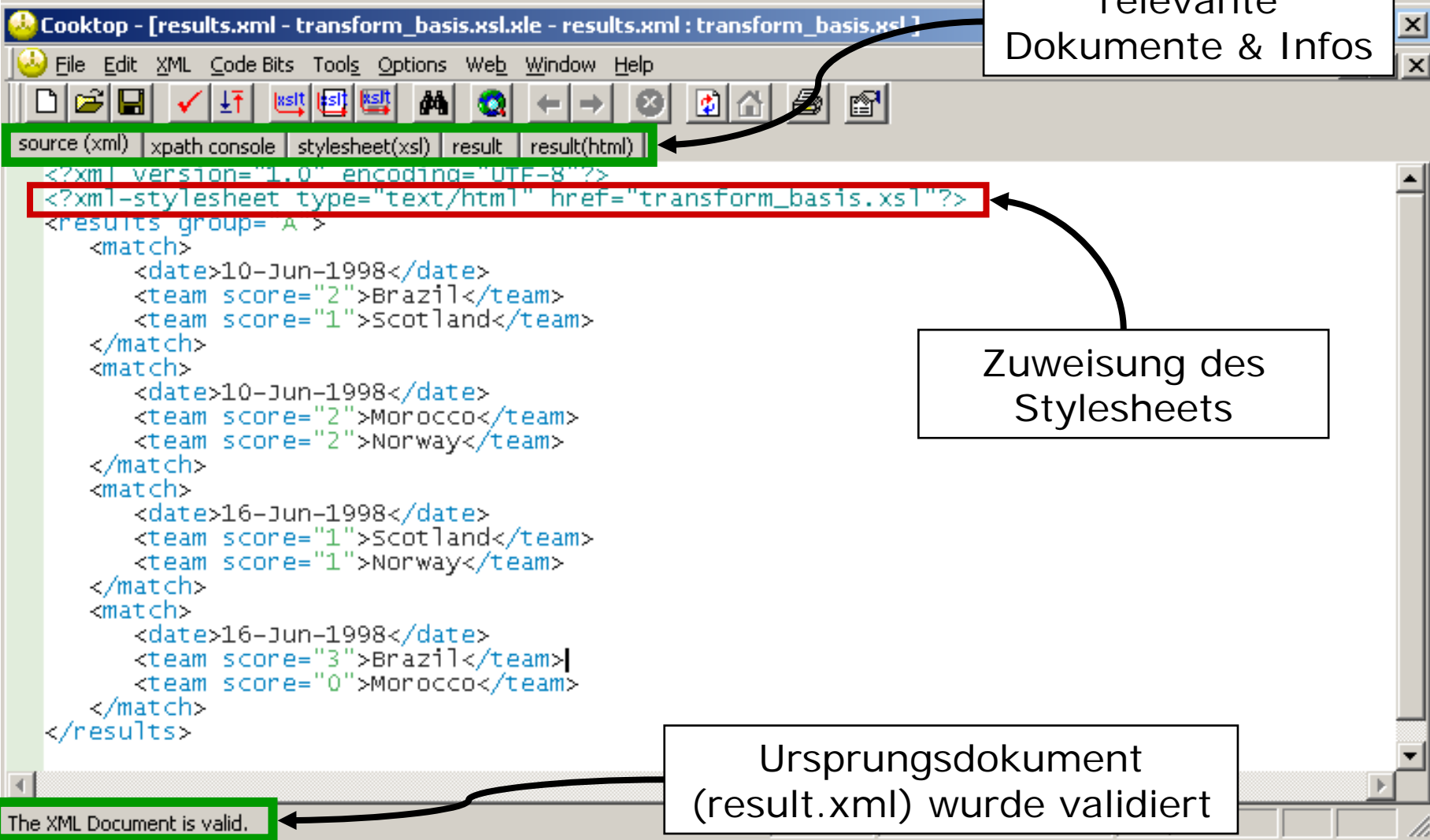
The screenshot shows the CookTop application window with the following annotations:

- XSLT ausführen**: Points to the 'Execute' button (a green checkmark icon) in the toolbar.
- Validieren**: Points to the 'Validate' button (a blue downward arrow icon) in the toolbar.
- Formatieren**: Points to the 'Format' button (a blue upward arrow icon) in the toolbar.
- Ergebnis in default-Browser öffnen**: Points to the 'Open in Default Browser' button (a blue document icon) in the toolbar.

The main window displays XML source code and a list of XSLT engines. The XML code includes elements like `<date>`, `<team>`, and `</match>`. The XSLT Engine list includes options like MSXML, Sablotron, and Xalan.

Beispiel von: <http://www.ibm.com/developerworks/xml/library/x-xslt/>

# Cooktop – Ursprungsdokument



The screenshot shows the Cooktop XML editor interface. The title bar reads "Cooktop - [results.xml - transform\_basis.xsl.xle - results.xml : transform\_basis.xsl]". The menu bar includes "File", "Edit", "XML", "Code Bits", "Tools", "Options", "Web", "Window", and "Help". The toolbar contains various icons for file operations and editing. The main text area displays XML source code with the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/html" href="transform_basis.xsl"?>
<results group= A >
  <match>
    <date>10-Jun-1998</date>
    <team score="2">Brazil</team>
    <team score="1">Scotland</team>
  </match>
  <match>
    <date>10-Jun-1998</date>
    <team score="2">Morocco</team>
    <team score="2">Norway</team>
  </match>
  <match>
    <date>16-Jun-1998</date>
    <team score="1">Scotland</team>
    <team score="1">Norway</team>
  </match>
  <match>
    <date>16-Jun-1998</date>
    <team score="3">Brazil</team>
    <team score="0">Morocco</team>
  </match>
</results>
```

Annotations in the image include:

- A box labeled "relevante Dokumente & Infos" with an arrow pointing to the "stylesheet(xsl)" tab in the editor's tab bar.
- A box labeled "Zuweisung des Stylesheets" with an arrow pointing to the line `<?xml-stylesheet type="text/html" href="transform_basis.xsl"?>` in the source code.
- A box labeled "Ursprungsdokument (result.xml) wurde validiert" with an arrow pointing to the status bar message "The XML Document is valid."

Beispiel von: <http://www.ibm.com/developerworks/xml/library/x-xslt/>

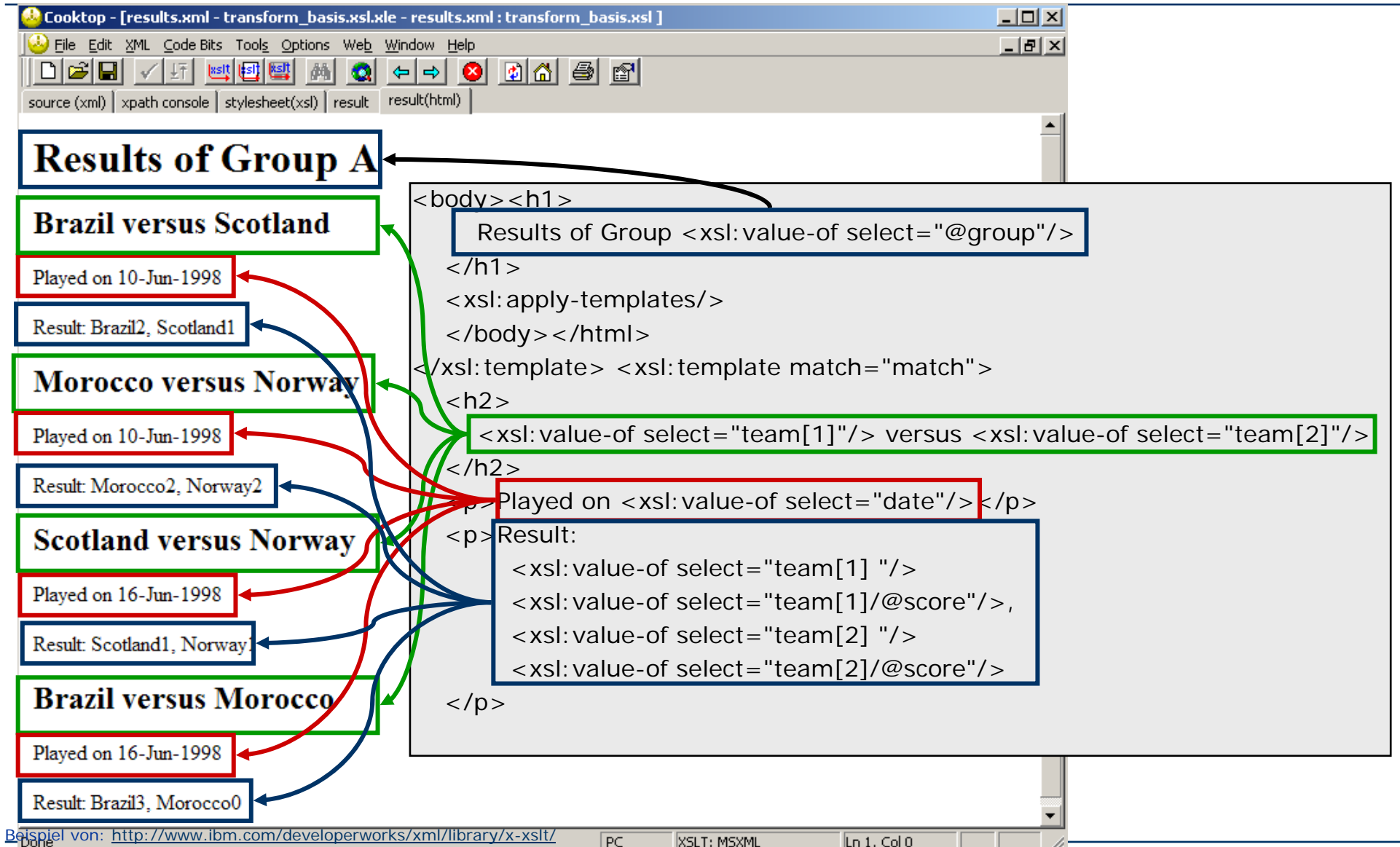
# Cooktop – XSLT Stylesheet (einfach)

```
Cooktop - [results.xml - transform_basis.xsl:transform_basis.xml]
File Edit XML Code Bits Tools Options Web Window Help
[Icons]
source (xml) xpath console stylesheet(xsl) result result(html)
<?xml version="1.0"?>
<xsl:transform
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="results">
  <html>
  <head><title>
    Results of Group <xsl:value-of select="@group"/>
  </title></head>
  <body><h1>
    Results of Group <xsl:value-of select="@group"/>
  </h1>
  <xsl:apply-templates/>
  </body></html>
</xsl:template> <xsl:template match="match">
  <h2>
    <xsl:value-of select="team[1]"/> versus <xsl:value-of select="team[2]"/>
  </h2>
  <p>Played on <xsl:value-of select="date"/></p>
  <p>Result:
    <xsl:value-of select="team[1]"/>
    <xsl:value-of select="team[1]/@score"/>,
    <xsl:value-of select="team[2]"/>
    <xsl:value-of select="team[2]/@score"/>
  </p>
</xsl:template> </xsl:transform>
```

```
<results group="A">
  <match>
    <date>10-Jun-1998</date>
    <team score="2">Brazil</team>
    <team score="1">Scotland</team>
  </match>
  <match>
    <date>10-Jun-1998</date>
    <team score="2">Morocco</team>
    <team score="2">Norway</team>
  </match>
  ...
</results>
```

Beispiel von: <http://www.ibm.com/developerworks/xml/library/x-xslt/>

# Cooktop – Ergebnis



The screenshot shows the Cooktop application window with the following content:

**Results of Group A**

- Brazil versus Scotland**
  - Played on 10-Jun-1998
  - Result: Brazil2, Scotland1
- Morocco versus Norway**
  - Played on 10-Jun-1998
  - Result: Morocco2, Norway2
- Scotland versus Norway**
  - Played on 16-Jun-1998
  - Result: Scotland1, Norway
- Brazil versus Morocco**
  - Played on 16-Jun-1998
  - Result: Brazil3, Morocco0

**XSLT Code Snippets:**

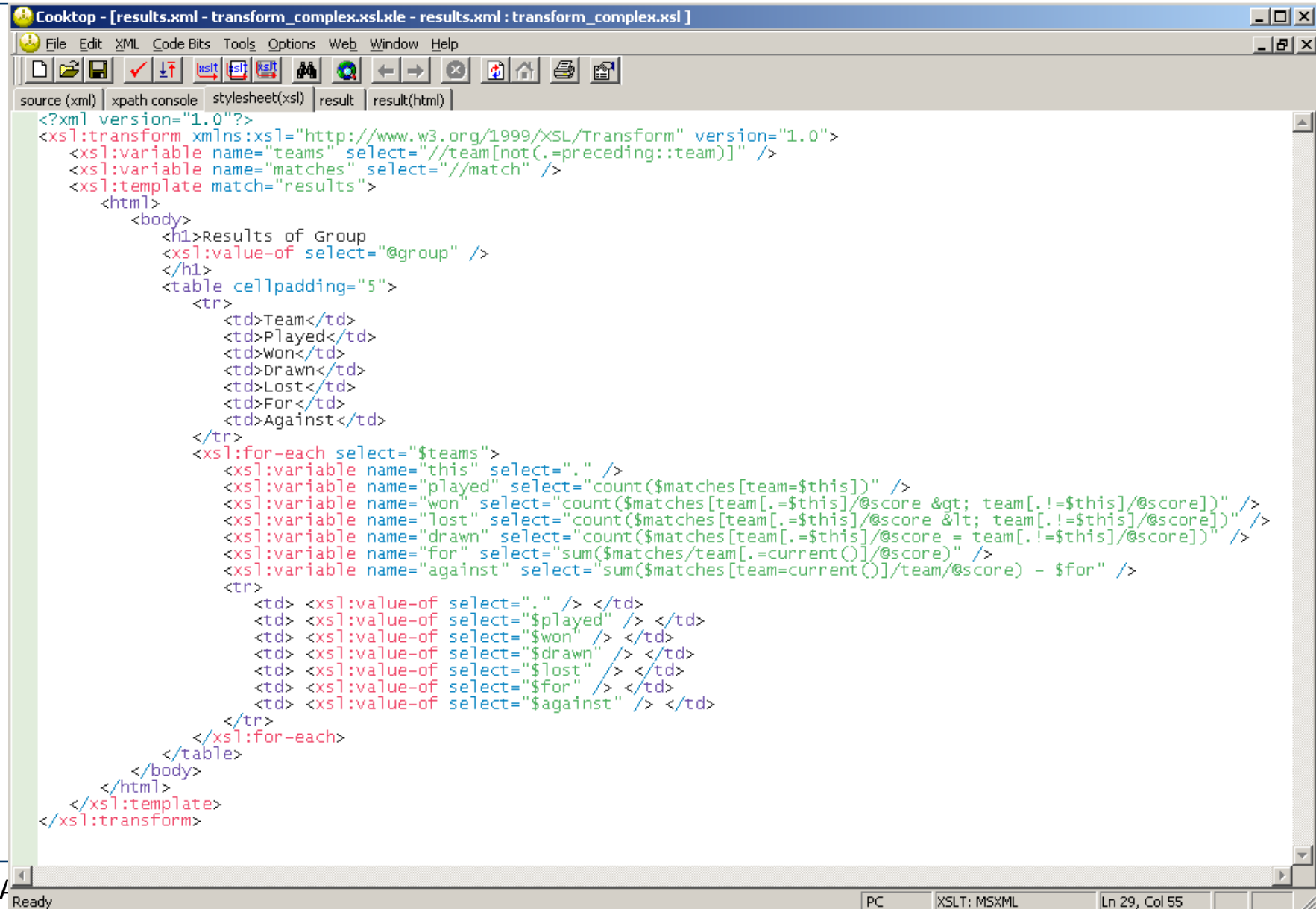
```
<body><h1>  
Results of Group <xsl:value-of select="@group"/>  
</h1>  
<xsl:apply-templates/>  
</body></html>  
</xsl:template> <xsl:template match="match">  
<h2>  
<xsl:value-of select="team[1]"/> versus <xsl:value-of select="team[2]"/>  
</h2>  
<p>Played on <xsl:value-of select="date"/></p>  
<p>Result:  
<xsl:value-of select="team[1]"/>  
<xsl:value-of select="team[1]/@score"/>,&br/><xsl:value-of select="team[2]"/>  
<xsl:value-of select="team[2]/@score"/>  
</p>
```

Annotations in the image include:

- A blue box around the title **Results of Group A** with an arrow pointing to the `<xsl:value-of select="@group"/>` snippet.
- Green boxes around the match titles (e.g., **Brazil versus Scotland**) with arrows pointing to the `<xsl:value-of select="team[1]"/> versus <xsl:value-of select="team[2]"/>` snippet.
- Red boxes around the dates (e.g., **Played on 10-Jun-1998**) with arrows pointing to the `<xsl:value-of select="date"/>` snippet.
- Blue boxes around the results (e.g., **Result: Brazil2, Scotland1**) with arrows pointing to the `<xsl:value-of select="team[1]"/>` and `<xsl:value-of select="team[2]"/>` snippets.

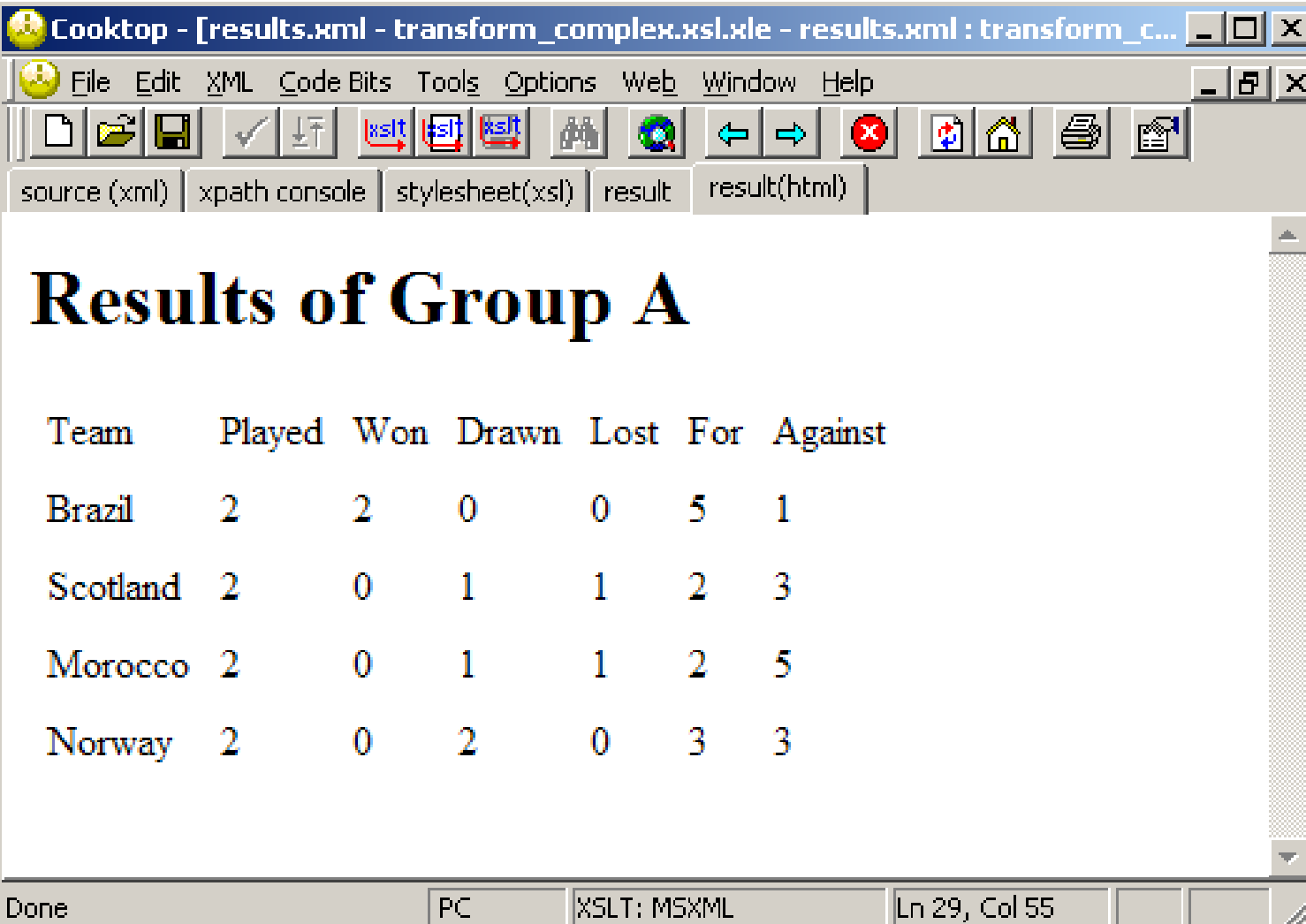
Beispiel von: <http://www.ibm.com/developerworks/xml/library/x-xslt/>

# Cooktop – XSLT Stylesheet (komplex)



```
<?xml version="1.0"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:variable name="teams" select="//team[not(.=preceding::team)]" />
  <xsl:variable name="matches" select="//match" />
  <xsl:template match="results">
    <html>
      <body>
        <h1>Results of Group
        <xsl:value-of select="@group" />
        </h1>
        <table cellpadding="5">
          <tr>
            <td>Team</td>
            <td>Played</td>
            <td>won</td>
            <td>Drawn</td>
            <td>Lost</td>
            <td>For</td>
            <td>Against</td>
          </tr>
          <xsl:for-each select="$teams">
            <xsl:variable name="this" select="." />
            <xsl:variable name="played" select="count($matches[team=$this])" />
            <xsl:variable name="won" select="count($matches[team[.= $this]/@score > team[!=$this]/@score])" />
            <xsl:variable name="lost" select="count($matches[team[.= $this]/@score < team[!=$this]/@score])" />
            <xsl:variable name="drawn" select="count($matches[team[.= $this]/@score = team[!=$this]/@score])" />
            <xsl:variable name="for" select="sum($matches/team[.=current()]/@score)" />
            <xsl:variable name="against" select="sum($matches[team=current()]/team/@score) - $for" />
            <tr>
              <td> <xsl:value-of select="." /> </td>
              <td> <xsl:value-of select="$played" /> </td>
              <td> <xsl:value-of select="$won" /> </td>
              <td> <xsl:value-of select="$drawn" /> </td>
              <td> <xsl:value-of select="$lost" /> </td>
              <td> <xsl:value-of select="$for" /> </td>
              <td> <xsl:value-of select="$against" /> </td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:transform>
```

# Cooktop – Ergebnis (II)



The screenshot shows the Cooktop application window with the following menu: File, Edit, XML, Code Bits, Tools, Options, Web, Window, Help. The toolbar includes icons for file operations, XSLT processing, and navigation. The main content area displays the following table:

Team	Played	Won	Drawn	Lost	For	Against
Brazil	2	2	0	0	5	1
Scotland	2	0	1	1	2	3
Morocco	2	0	1	1	2	5
Norway	2	0	2	0	3	3

The status bar at the bottom shows: Done, PC, XSLT: MSXML, Ln 29, Col 55.

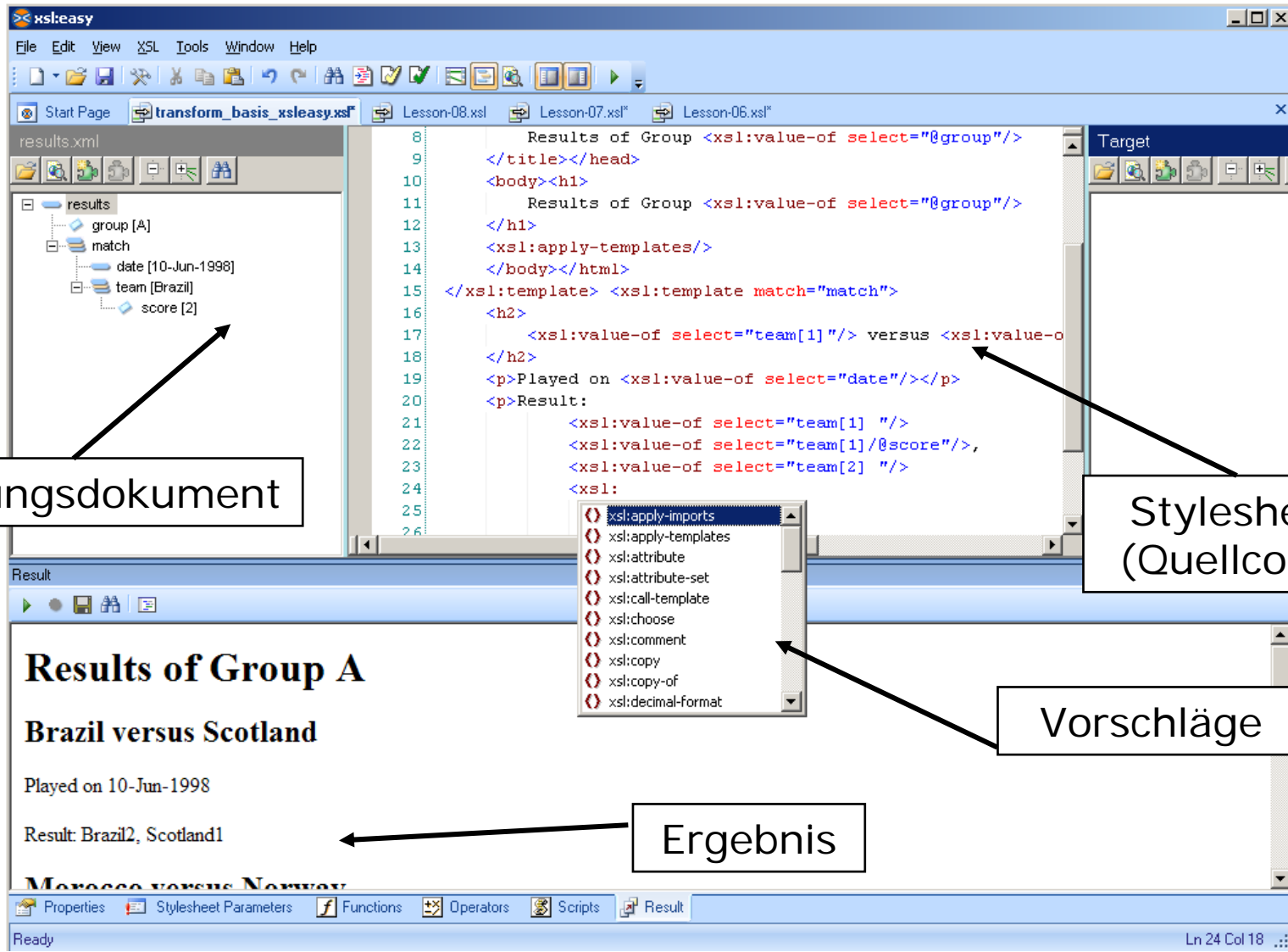
Beispiel von: <http://www.ibm.com/developerworks/xml/library/x-xslt/>



**xsl:easy**

---

# xsl:easy



The screenshot shows the xsl:easy application interface. On the left, a tree view displays the source XML document 'results.xml' with a structure: results > group [A] > match > date [10-Jun-1998] > team [Brazil] > score [2]. The main editor shows XSLT code for transforming this XML into HTML. The code includes a template for 'match' that generates a page header, a title, and a body with an h1, h2, and paragraphs. A dropdown menu is open, showing a list of XSLT elements: xsl:apply-imports, xsl:apply-templates, xsl:attribute, xsl:attribute-set, xsl:call-template, xsl:choose, xsl:comment, xsl:copy, xsl:copy-of, and xsl:decimal-format. The 'Result' pane at the bottom displays the output HTML: 'Results of Group A', 'Brazil versus Scotland', 'Played on 10-Jun-1998', and 'Result: Brazil2, Scotland1'.

Ursprungsdokument

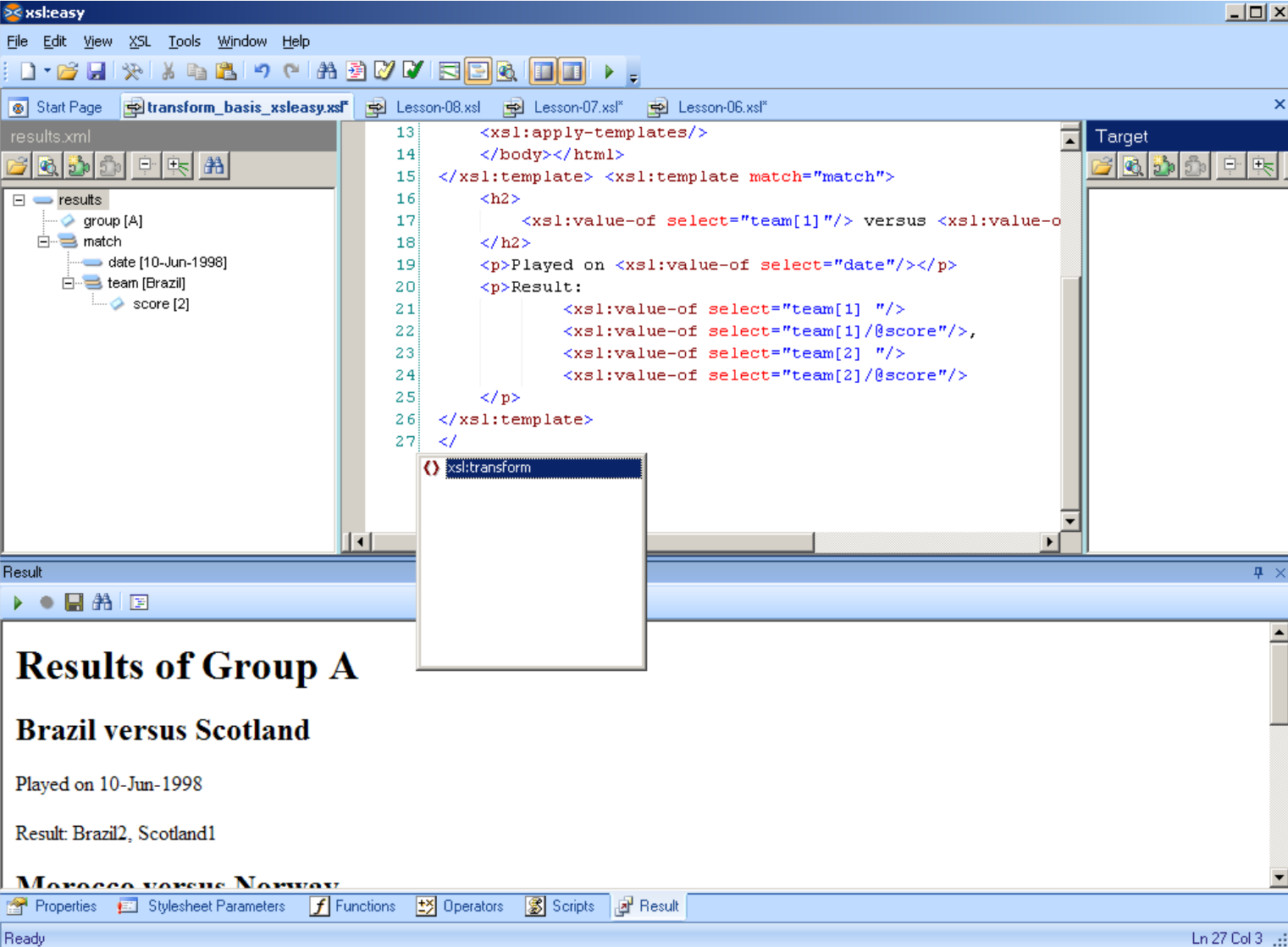
Stylesheet  
(Quellcode)

Vorschläge

Ergebnis

Beispiel von: <http://www.ibm.com/developerworks/xml/library/x-xslt/>

# xsl:easy – End-Tags

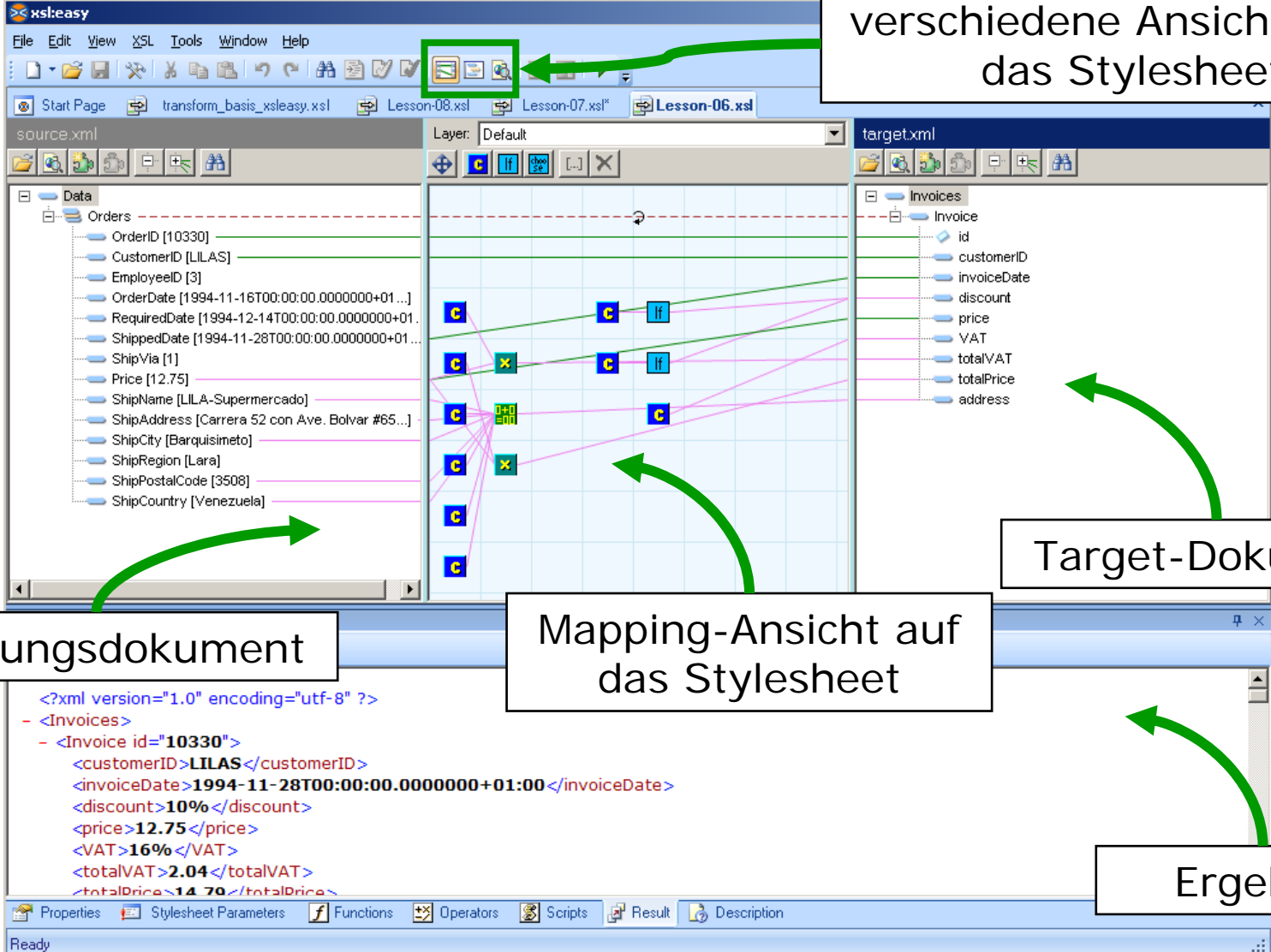


The screenshot shows the xsl:easy application interface. The main window displays XSL code for transforming XML data. The code includes an `<xsl:apply-templates/>` tag, followed by an `</xsl:template>` block with a `match="match"` attribute. Inside the template, there are several XSL instructions: `<h2>`, `<xsl:value-of select="team[1]"/>`, `versus`, `<xsl:value-of select="team[2]"/>`, `</h2>`, `<p>Played on <xsl:value-of select="date"/></p>`, `<p>Result:`, `<xsl:value-of select="team[1]"/>`, `<xsl:value-of select="team[1]/@score"/>`, `<xsl:value-of select="team[2]"/>`, `<xsl:value-of select="team[2]/@score"/>`, `</p>`, `</xsl:template>`, and `</>`. The left sidebar shows a tree view of the source XML with nodes: `results`, `group [A]`, `match`, `date [10-Jun-1998]`, `team [Brazil]`, and `score [2]`. The bottom pane shows the resulting HTML output:

```
Results of Group A  
Brazil versus Scotland  
Played on 10-Jun-1998  
Result: Brazil2, Scotland1  
Morocco versus Norway
```

The status bar at the bottom indicates the current position is `Ln 27 Col 3`.

# xsl:easy – Fenster & Ansichten



The screenshot shows the xsl:easy application interface. At the top, a menu bar includes File, Edit, View, XSL, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main workspace is divided into four panes:

- source.xml**: A tree view of the source XML document. The root is 'Orders', which contains elements like OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Price, ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, and ShipCountry.
- Mapping-Ansicht auf das Stylesheet**: A central mapping view showing connections between source and target elements. It features a grid with colored boxes (blue 'c', green 'x', pink 'f') and lines representing mappings.
- target.xml**: A tree view of the target XML document. The root is 'Invoices', which contains elements like id, customerID, invoiceDate, discount, price, VAT, totalVAT, totalPrice, and address.
- Ergebnis**: A text view showing the resulting XML output. The output is an XML document with a root element 'Invoices' containing one 'Invoice' element with attributes and child elements corresponding to the source data.

Annotations with green arrows point to specific parts of the interface:

- A box labeled "verschiedene Ansichten auf das Stylesheet" points to the toolbar icons.
- A box labeled "Ursprungsdokument" points to the source.xml tree view.
- A box labeled "Mapping-Ansicht auf das Stylesheet" points to the central mapping grid.
- A box labeled "Target-Dokument" points to the target.xml tree view.
- A box labeled "Ergebnis" points to the resulting XML output.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Invoices>
- <Invoice id="10330">
  <customerID>LILAS</customerID>
  <invoiceDate>1994-11-28T00:00:00.0000000+01:00</invoiceDate>
  <discount>10%</discount>
  <price>12.75</price>
  <VAT>16%</VAT>
  <totalVAT>2.04</totalVAT>
  <totalPrice>14.70</totalPrice>
```

# xsl:easy "Angebot"

- Funktionen, Constanten, Parameter, Operationen, if und choose Konstrukte über Drag&Drop einfügen



**Functions**

<b>B</b> Boolean	key	String
Ceiling	lang	String Length
Concat	Last	Substring After
Contains	local-name	Substring Before
Count	Name	Substring
element-available	namespaceURI	Sum
false	Normalize Space	system-property
Floor	Not	Translate
Format Number	Number	true
function-available	Position	unparsed-entity-uri
generate-id	Round	
id	Starts With	

Properties Stylesheet Parameters Functions Operators  
Ready

**Operators**

- + Adding
- & AND
- / Divide
- = Equal
- > Greater
- >= Greater Or Equal
- < Less
- <= Less Or Equal
- mod Mod
- x Multiply
- != Not Equal
- or OR
- Subtract

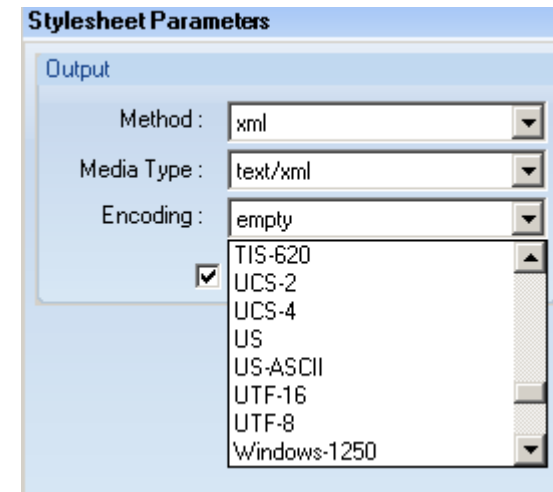
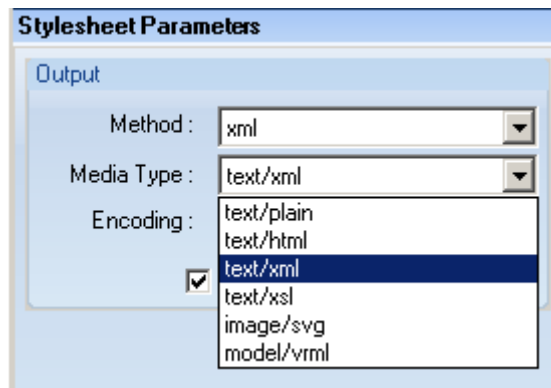
**Scripts**

abs Abs	ssp GetSubstringPosition	mul Multipl
add AddAll	hex Hex	oct Oct
ada AddDays	int Int	orf OrForA
and AndForAll	isd IsDate	pow Power
arc ArcTan	isn IsNumeric	rgt Right
asc Asc	iss IsString	rnd Round
chr Chr	lca LCCase	rtr RTrim
concat ConcatAll	lef Left	sin Sin
cos Cos	ln Ln	sqr Sqrt
cda CurrentDate	l10 Log10	sub SubtractAll
cdt CurrentDateAndTime	lon LogN	tan Tan
cti CurrentTime	ltr LTrim	uca UCase
e10 Exp10	mx Max	ucf UCaseFirstChar
exe ExpE	min Min	

Properties Stylesheet Parameters Functions Op  
Ready

# Output Parameter

- Methode
  - `<xsl:output method="xml"/>`
  - `<xsl:output method="text"/>`
  - `<xsl:output method="html"/>`
- Ausgabebetyp
- Encoding-Typ





**Mehrere Ursprungsdokumente**

---

## Mehrere Ursprungsdokumente

- Verbinden von
  - Buchkapitel
  - Brief-Templates
  - Adresslisten
- Erstellen von Inhaltverzeichnissen, die mehrere Dateien erfassen
- Wiederverwenden von Bildbeschreibungen in verschiedenen Galerien

# 3 Dokumente

```
<?xml version="1.0"?>
<ph:photo
  xmlns:ph="http://page.mi.fu-berlin.de/mochol/xml_xsl/" >
  <ph:title>Jet d Eau</ph:title>
  <ph:location>Genf (Geneve)</ph:location>
  <ph:date>Juni 2008</ph:date>
  <ph:description>
    Der Jet d Eau (franz. Wasserstrahl) ist
    hoher Springbrunnen im Genfersee und
    Wahrzeichen der Stadt Genf.
  </ph:description>
</ph:photo>
```

geneve.xml

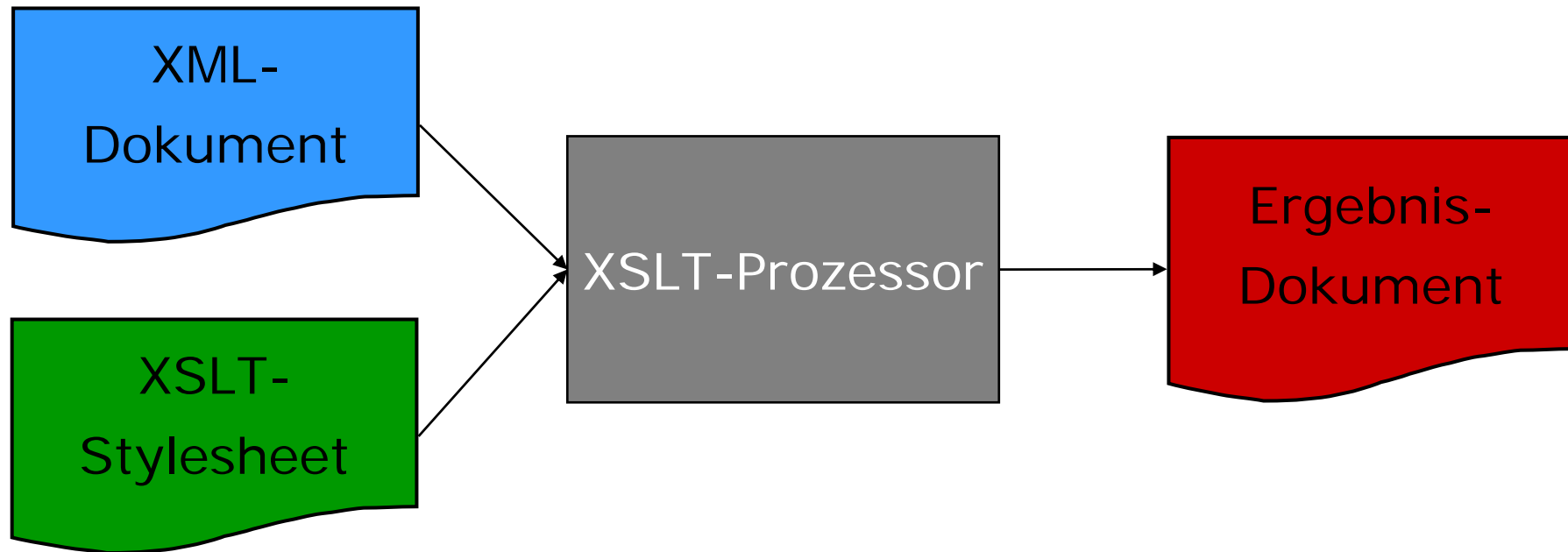
```
<?xml version="1.0"?>
<ph:photo xmlns:ph="http://page.mi.fu-berlin.de/mochol/xml_xsl/" >
  <ph:title>Brandenburger Tor</ph:title>
  <ph:location>Berlin</ph:location>
  <ph:date>Juni 2008</ph:date>
  <ph:description>
    Das Brandenburger Tor am Pariser Platz in der Dorotheenstadt im
    Ortsteil Mitte (Bezirk Mitte) von Berlin wurde in den Jahren von
    1788 bis 1791 auf Anweisung des preussischen
    Koenigs Friedrich Wilhelm II. ....
  </ph:description>
</ph:photo>
```

berlin.xml

```
<?xml version="1.0"?>
<ph:photo xmlns:ph="http://page.m
  <ph:title>Tower-Bridge</ph:title>
  <ph:location>London</ph:location>
  <ph:date>Juni 2008</ph:date>
  <ph:description>
    Die Tower Bridge ist eine Strassenbruecke ueber den Fluss
    Themse in London. Sie wurde 1894 eroeffnet und verbindet die
    City of London auf der Nordseite mit dem Stadtteil Southwark im
    Stadtbezirk London Borough of Southwark auf der Suedseite.
  </ph:description>
</ph:photo>
```

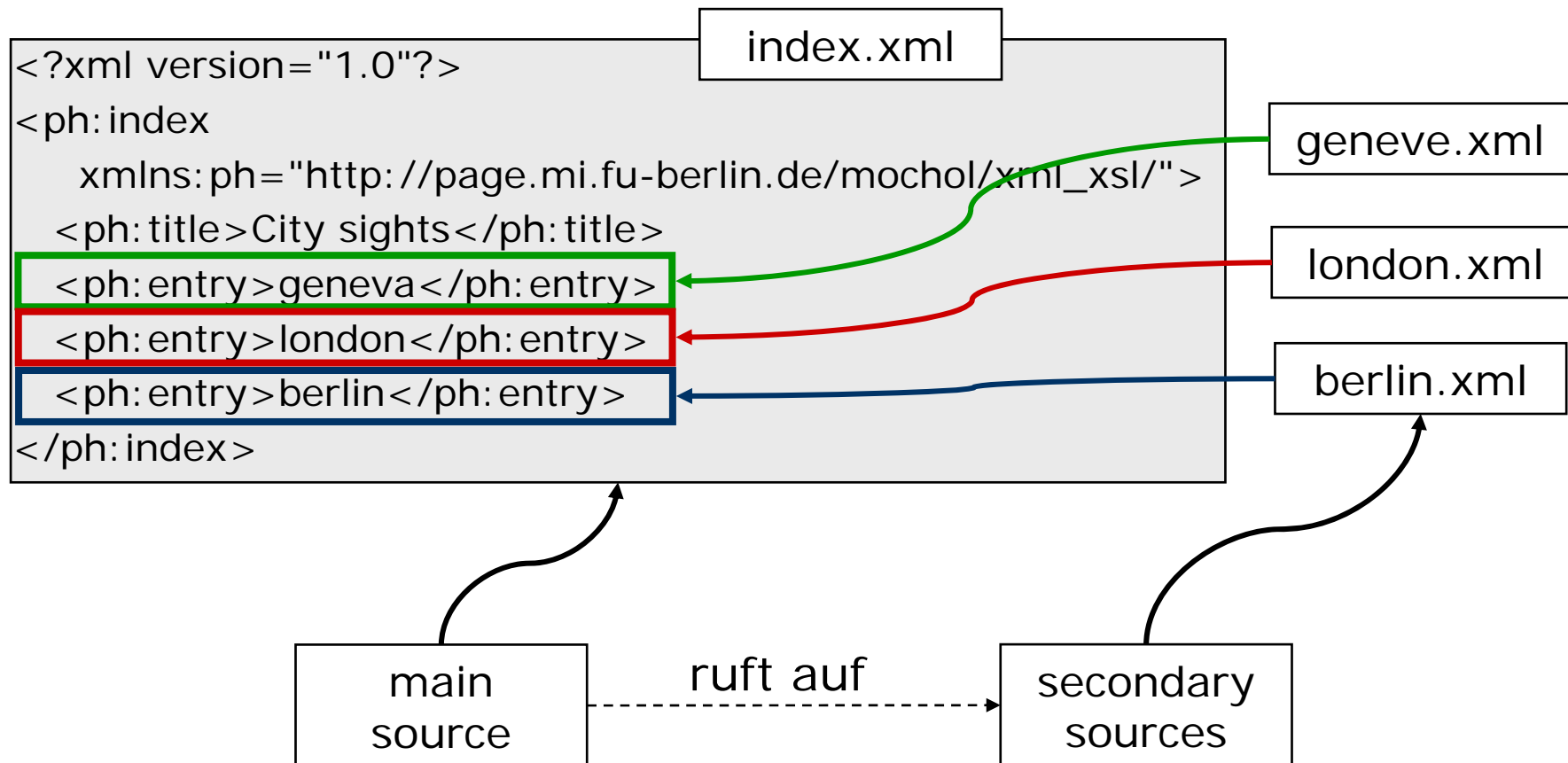
london.xml

## Allg. Schema der Transformation



- *Problem:*
  - 3 XML-Dokumente (*geneve.xml*, *london.xml* & *berlin.xml*) aber erlaubt ist nur ein XML-Dokument als Input

# Main & secondary sources



Hier: main source = Ursprungsdokument

# Stylesheet (I)

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:ph="http://page.mi.fu-berlin.de/mochol/xml_xsl/">
<xsl:output method="html" media-type="text/html" />

<xsl:template match="ph:index">
  <html>
    <head><title><xsl:value-of select="ph:title"/></title></head>
    <xsl:apply-templates/>
  </html>
</xsl:template>

<xsl:template match="ph:index/ph:title">
  <h1><xsl:apply-templates/></h1>
</xsl:template>

<xsl:template match="ph:entry">
  
  <xsl:apply-templates select="document(concat(., '.xml'))"/>
  <br clear="right"/>
</xsl:template>

<xsl:template match="ph:photo/ph:title">
  <h2><xsl:apply-templates/></h2>
</xsl:template>

<xsl:template match="ph:location">
  <h3>in <xsl:apply-templates/></h3>
</xsl:template>

<xsl:template match="ph:date">
  <p>Date: <xsl:apply-templates/></p>
</xsl:template>

<xsl:template match="ph:description">
  <p><xsl:apply-templates/></p>
</xsl:template>
</xsl:stylesheet>

```

Namespace

index-Knoten aus index.xml

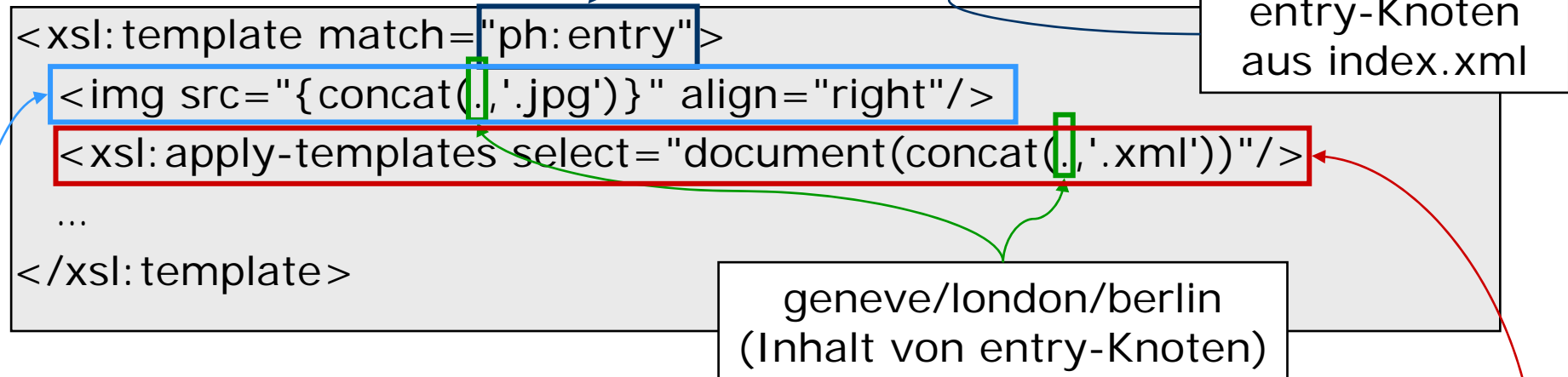
titel-Knoten aus index.xml

entry-Knoten aus index.xml

## concat()

- *string* **concat**(*string* *s1*, *string* *s2*)
  - *string* **concat**(*string* *s1*, *string* *s2*, *string* *s3*)
  - *string* **concat**(*string* *s1*, *string* *s2*, *string* *s3*, ...)
- 
- Argumente werden von links nach rechts verkettet
  - Anzahl der Argumente beliebig
  - nicht-Strings werden mit der Funktion `string()` konvertiert

# Auflösung von concat()



```





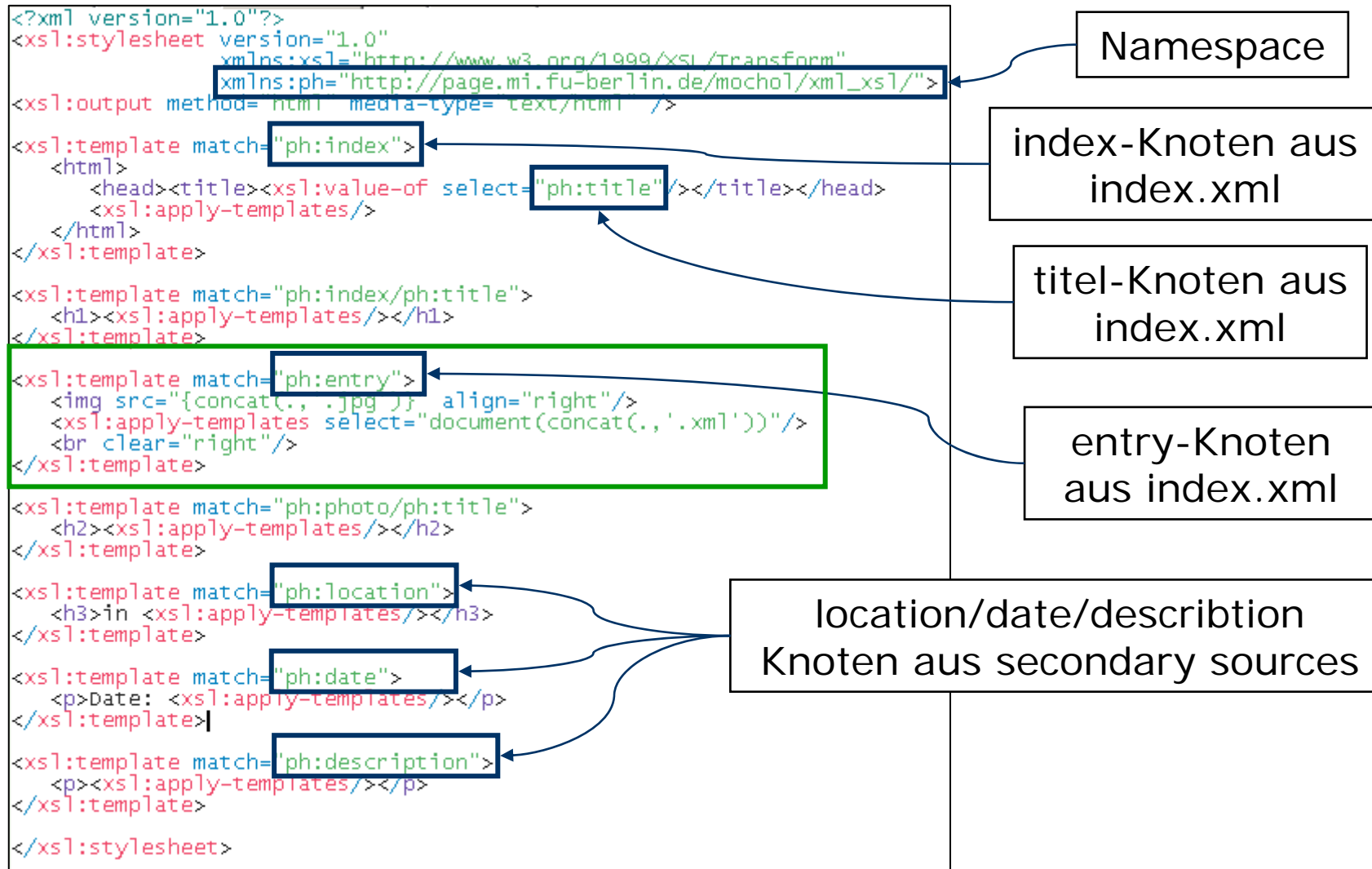
```

```

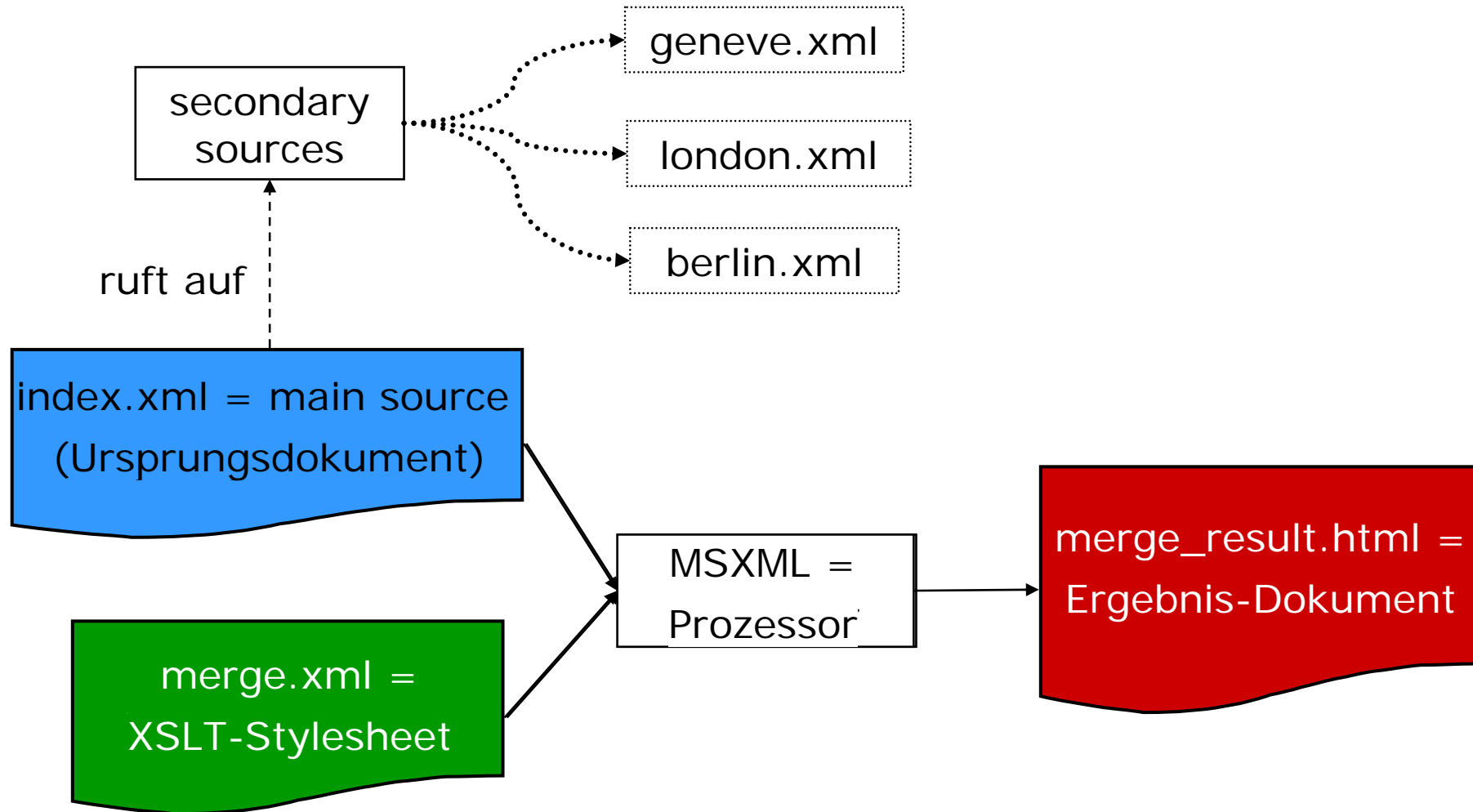
<xsl:apply-templates select="document(geneve.xml)"/>
<xsl:apply-templates select="document(london.xml)"/>
<xsl:apply-templates select="document(berlin.xml)"/>

```

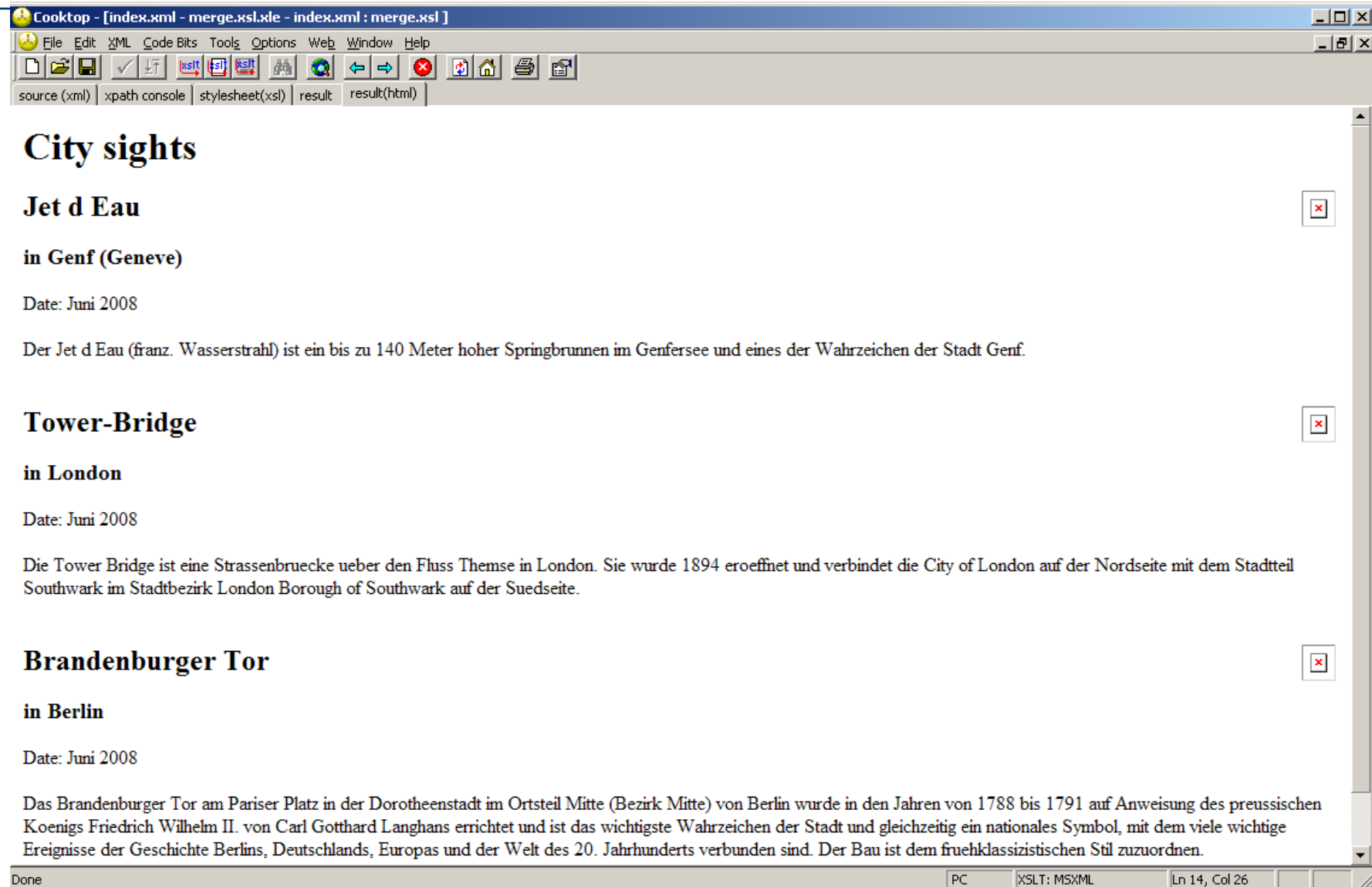
# Stylesheet (II)



# Ablauf der Verarbeitung



# Ausgabe in CookTop



**City sights**

**Jet d Eau**

**in Genf (Geneve)**

Date: Juni 2008

Der Jet d Eau (franz. Wasserstrahl) ist ein bis zu 140 Meter hoher Springbrunnen im Genfersee und eines der Wahrzeichen der Stadt Genf.

**Tower-Bridge**

**in London**

Date: Juni 2008

Die Tower Bridge ist eine Strassenbruecke ueber den Fluss Themse in London. Sie wurde 1894 eroeffnet und verbindet die City of London auf der Nordseite mit dem Stadtteil Southwark im Stadtbezirk London Borough of Southwark auf der Suedseite.

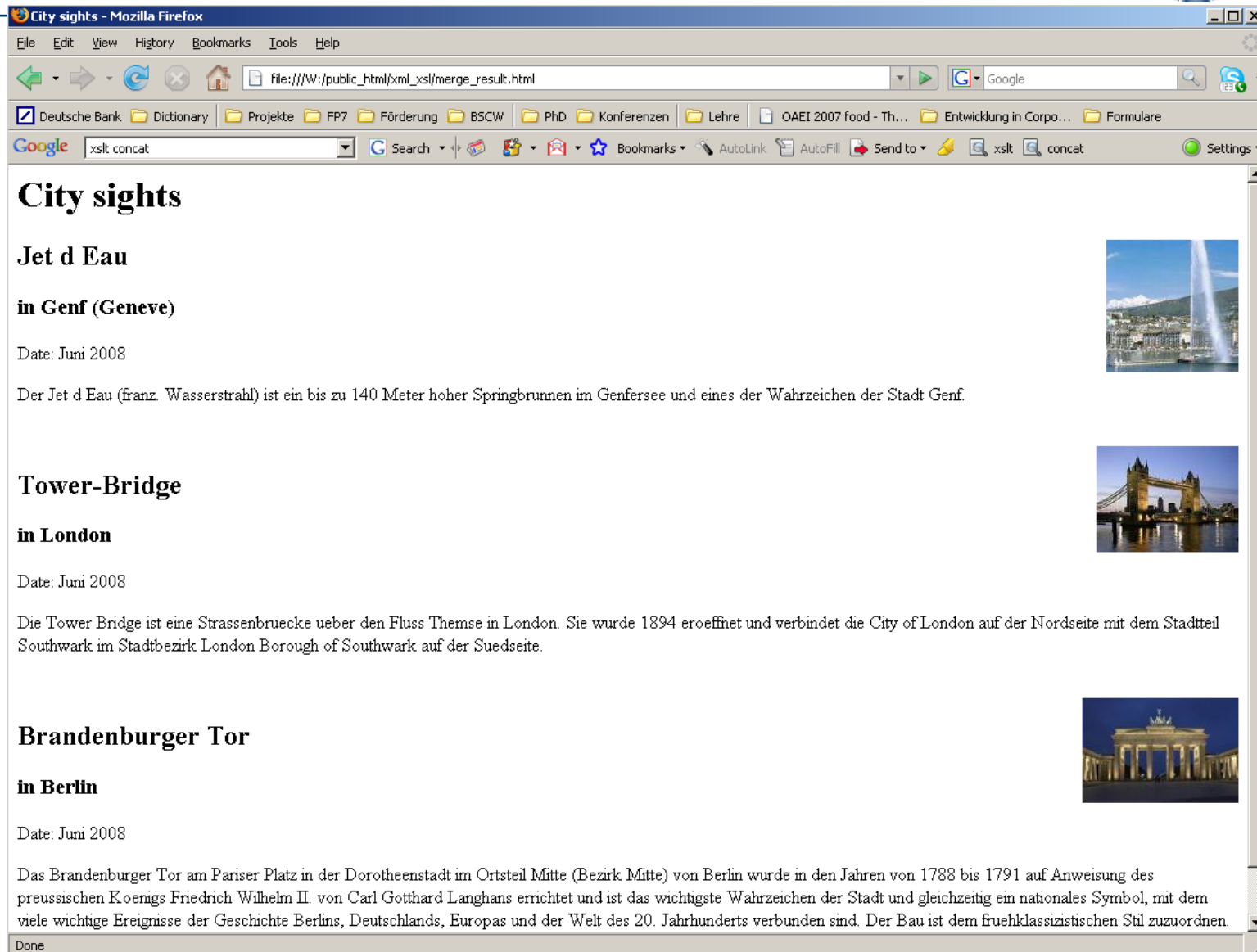
**Brandenburger Tor**

**in Berlin**

Date: Juni 2008

Das Brandenburger Tor am Pariser Platz in der Dorotheenstadt im Ortsteil Mitte (Bezirk Mitte) von Berlin wurde in den Jahren von 1788 bis 1791 auf Anweisung des preussischen Koenigs Friedrich Wilhelm II. von Carl Gotthard Langhans errichtet und ist das wichtigste Wahrzeichen der Stadt und gleichzeitig ein nationales Symbol, mit dem viele wichtige Ereignisse der Geschichte Berlins, Deutschlands, Europas und der Welt des 20. Jahrhunderts verbunden sind. Der Bau ist dem fruehklassizistischen Stil zuzuordnen.

# Ausgabe in Firefox





**Sortieren mit `<xsl:sort>`**

---

## <xsl:sort>

- definiert in welcher Reihenfolge die Knoten bearbeitet werden
- mehrere Sortierungsschlüssel möglich
- die Elemente werden nach dem ersten <xsl:sort>, dann nach dem zweiten <xsl:sort> usw. sortiert
- Position:
  - innerhalb von <xsl:apply-templates> oder <xsl:for-each>

## Attribute von <xsl:sort>

```
<xsl:sort select="price" order="descending" data-type="number"/>
```

- **order** {"ascending | descending"} - aufsteigende oder absteigende Reihenfolge
- **case-order** {"upper-first | lower-first"}  
Großbuchstaben vor Kleinbuchstaben oder Kleinbuchstaben vor Großbuchstaben folgen
- **lang** {language-code} Sprach-Code, z.B.: "ä" nach "a" für den deutschen Sprach-Code oder "ä" nach "z" für den schwedischen Sprach-Code
- **data-type** {"text | number"} alphabetische oder numerische Sortierung

## Zu sortieren

```
<employees>
  <employee>
    <name>
      <given>James</given>
      <family>Clark</family>
    </name>
    ...
  </employee>
</employees>
```

[XSL Transformations (XSLT) Version 1.0 W3C  
Recommendation 16 November 1999]

# Templates

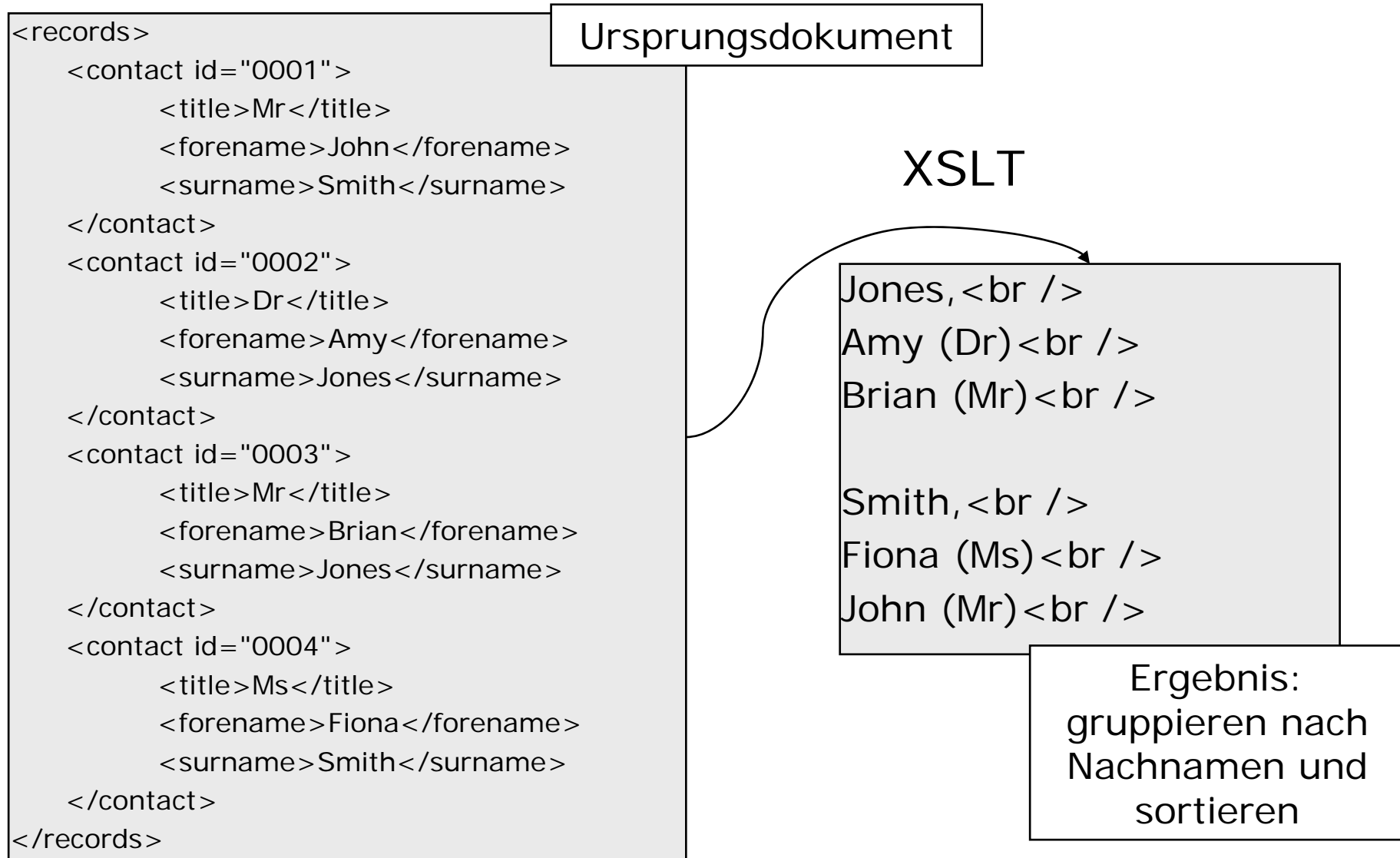
```
<xsl:template match="employees">
  <ul>
    <xsl:apply-templates select="employee">
      <xsl:sort select="name/family"/>
      <xsl:sort select="name/given"/>
    </xsl:apply-templates>
  </ul>
</xsl:template>
```

```
<xsl:template match="employee">
  <li>
    <xsl:value-of select="name/given"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="name/family"/>
  </li>
</xsl:template>
```



## Gruppieren von XML-Daten

# Gruppieren in XSLT 1.0



## 1. Ermitteln der auftretenden Nachnamen

- jeweils einen Eintrag für jeden Nachnamen bestimmen
- alle Einträge bestimmen, die einen Nachnamen besitzen, der nicht bereits bei einem vorhergehenden Eintrag zu finden ist

```
contact[not(surname = preceding-sibling::contact/surname)]
```

## 2. Bestimmen aller Einträge (contact) auf den gleichen Nachnamen

```
<xsl:apply-templates  
  select="/records/contact[surname = current()/surename]" />
```

- zwei XPath Ausdrücke, die bearbeitet werden müssen → viel Prozessor-Zeit im Falle von großen XML-Dateien
- Durchsuchen aller Vorgänger mit 'preceding-siblings'-Achse sehr aufwändig (besonders am Ende der Liste)
- Bestimmung aller Einträge mit einem bestimmten Nachnamen erfordert jeweils den Zugriff auf jeden Eintrag

→ sehr ineffiziente Vorgehensweise

- effizientere Gruppierung durch Benutzung von Keys (entwickelt von Steve Muench)
- **Key** weist einem XML-Knoten einen **Schlüssel** zu
- Zugriff auf Knoten über den dazugehörigen Schlüssel

→ bei vielen Knoten mit dem gleichen Schlüsselwert werden alle diese Knoten durch Benutzung dieses Schlüsselwertes ermittelt

Keys sehr effizient fürs Gruppieren einer bestimmten Anzahl von Knoten bezüglich einer gewissen Eigenschaft

## Zugriffsschlüssel <xsl:key>

- xsl:key - definiert einen Zugriffsschlüssel auf Elemente oder Attribute
  - key() – Verwendung des Zugriffsschlüssels
- Attribute von xsl:key
  - name(*obligatorisch*) – legt einen Namen für den Schlüssel fest, unter dem er angewendet werden kann.
  - use(*obligatorisch*) gibt an, woraus der Wert des Schlüssels ermittelt wird
  - match(*optional*) – gibt an, wo der Schlüssel im Elementenbaum ansetzen soll

## Zurück zum Beispiel ...

1. Schlüssel, der jedem Eintrag als Schlüsselwert den dazugehörigen Nachnamen zuweist

```
<xsl:key name="contacts-by-surname"  
         use="surname"  
         match="contact" />
```

Name des Keys

Wert des Schlüssels aus dem Element  
<surname>

Schlüssel soll im Element  
<contact> ansetzen

## Zurück zum Beispiel (II) ...

---

2a. Bestimmung alle Einträge mit einem bestimmten Nachnamen (d.h. Nachname bekannt, z.B.: "Smith")

```
(key('contacts-by-surname', 'Smith'))
```

2b. Bestimmung aller Einträge mit dem gleichen Nachnamen (d.h. Nachname unbekannt)

```
<xsl:apply-templates select="key('contacts-by-surname', surname)" />
```

3. Überprüfung, ob der Eintrag der erste in der durch einen Schlüssel zurückgegebenen Liste ist
  - Vergleich der mittels `generate-id()` erzeugten eindeutigen Bezeichner für die beiden Knoten:

```
contact[generate-id() = generate-id(key('contacts-by-surname', surname)[1])]
```

- „The **generate-id** function returns a string that uniquely identifies the node in the argument node-set that is first in document order. [...] the string is syntactically an XML name.“



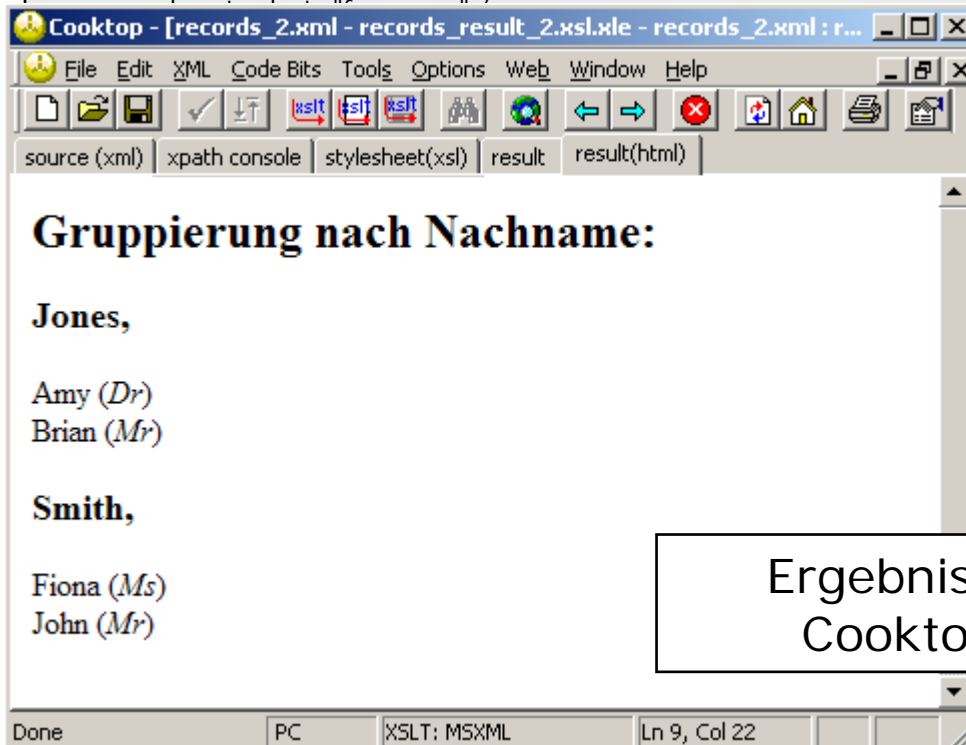
# Ergebnis

## XSLT-Stylesheet

```
<records>
  <contact id="0001">
    <title>Mr</title>
    <forename>John</forename>
    <surname>Smith</surname>
  </contact>
  <contact id="0002">
    <title>Dr</title>
    <forename>Amy</forename>
    <surname>Jones</surname>
  </contact>
  <contact id="0003">
    <title>Mr</title>
    <forename>Brian</forename>
    <surname>Jones</surname>
  </contact>
  <contact id="0004">
    <title>Ms</title>
    <forename>Fiona</forename>
    <surname>Smith</surname>
  </contact>
</records>
```

## Ursprungsdokument

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" media-type="text/html" />
  <xsl:key name="contacts-by-surname" match="contact" use="surname" />
  <xsl:template match="records">
    <h2>Gruppierung nach Nachname:</h2>
    <xsl:for-each
      select="contact[generate-id() = generate-id(key('contacts-by-surname', surname)[1])]">
      <xsl:sort select="surname" />
      <h3><xsl:value-of select="surname" />,</h3>
      <xsl:for-each select="key('contacts-by-surname', surname)">
```



The screenshot shows the Cooktop application interface. The title bar reads "Cooktop - [records\_2.xml - records\_result\_2.xsl.xle - records\_2.xml : r...". The menu bar includes "File", "Edit", "XML", "Code Bits", "Tools", "Options", "Web", "Window", and "Help". The toolbar contains icons for file operations and XSLT processing. The main window has tabs for "source (xml)", "xpath console", "stylesheet(xsl)", "result", and "result(html)". The "result" tab is active, displaying the output of the XSLT transformation:

```
Gruppierung nach Nachname:

Jones,

Amy (Dr)
Brian (Mr)

Smith,

Fiona (Ms)
John (Mr)
```

The status bar at the bottom shows "Done", "PC", "XSLT: MSXML", and "Ln 9, Col 22".

## Ergebnis in Cooktop

- + erfordert nicht das Durchsuchen einer großen Anzahl von Knoten
- + kann sehr effizient ausgeführt werden
  
- nicht alle XSLT-Prozessoren unterstützen Keys
- kann speicherintensiv sein, weil alle Knoten & ihre Schlüsselwerte im Speicher gehalten werden müssen
- reichlich kompliziert, wenn die Knoten über mehrere Quelldokumente verteilt sind



## XSLT 2.0

- XSLT 2.0 wurde entwickelt um
  - die Gestaltungsmöglichkeiten durch Stylesheets zu erweitern
  - einige Schwächen von XSLT 1.0 zu beseitigen
  - hinsichtlich der Sprachen viele neue und gleichzeitige prägnantere, effizientere, flexiblere und erheblich leistungsstärkere Features
- XPath 2.0 liefert
  - Vereinfachung & Erweiterung der Zugriffe auf Daten

**XSLT 2.0 & XPath 2.0 rückwärts kompatibel !**

## Was ist neu in XSLT 2.0?

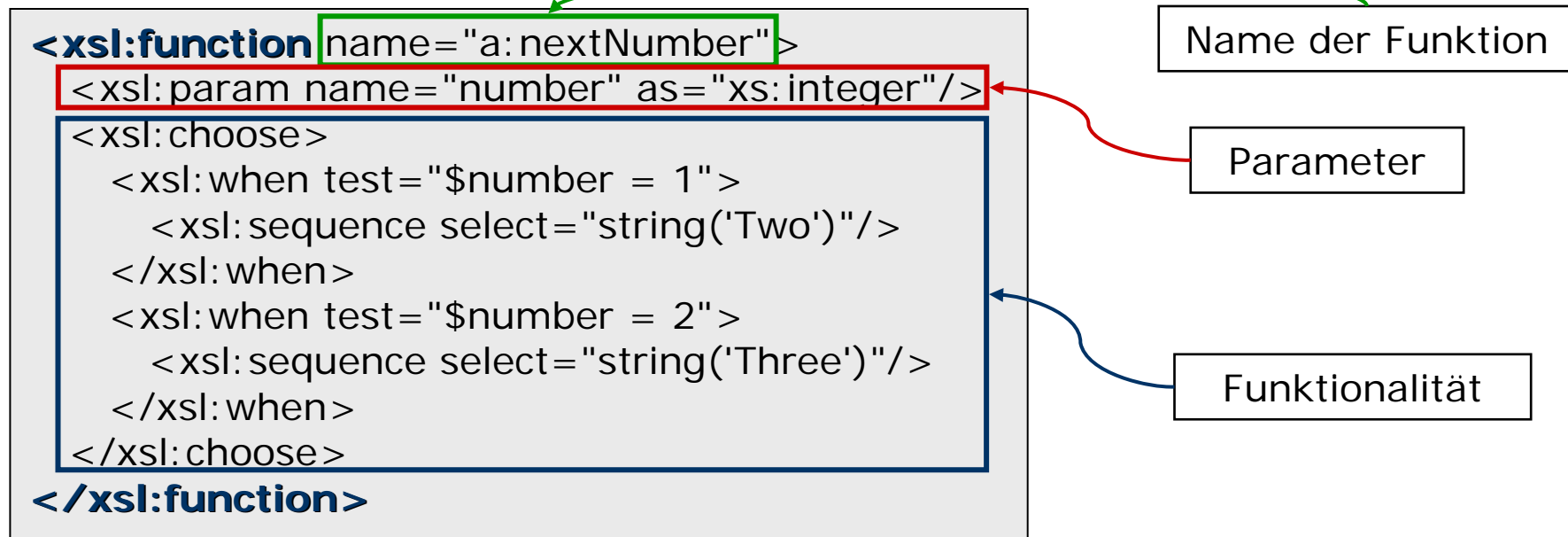
1. neue Funktionen und Operatoren
2. Benutzerdefinierte Funktionen
3. mehrere Ergebnisdokumente
4. Unterstützung der Ausgabe in XHTML
5. weitere neue Elemente
  
6. einfachere Gruppierung

## 1. Neue Funktionen

- **format-date** – formatiert ein Datum
- **format-dateTime** – formatiert einen Datums/Zeit-Wert
- **format-time** – formatiert einen Zeitwert
  
- **type-available** – prüft die Verfügbarkeit eines bestimmten Datentyps

## 2. Benutzerdefinierte Funktionen mit `<xsl:function>`

- Erzeugen von benutzerdefinierten XSLT-Funktionen
- benutzt XPath-Ausdrücke
- legt den Namen der Funktion fest
- legt die Benötigten Parameter fest
- implementier die gewünschte Funktionalität



## <xsl:function> Beispiel

```
<xsl:function name="a:nextNumber">  
  <xsl:param name="number" as="xs:integer"/>  
  <xsl:choose>  
    <xsl:when test="$number = 1">  
      <xsl:sequence select="string('Two')"/>  
    </xsl:when>  
    <xsl:when test="$number = 2">  
      <xsl:sequence select="string('Three')"/>  
    </xsl:when>  
  </xsl:choose>  
</xsl:function>
```

Definition der Funktion  
a:nextNumber

Aufruf der Funktion  
a:nextNumber

<xsl:value-of select="a:nextNumber( 1 )"/>

- Benutzerdefinierte Funktionen sind an allen Stellen im Stylesheet für XPath-Ausdrücke verfügbar

### 3. Mehrere Ergebnisdokumente

- Transformation produziert mehrere Ergebnisbäume
- → Basis für Erzeugung mehrerer Ergebnisdokumenten
  - z.B. gleichzeitige Erzeugung von dem Gesamtdokument und einer Dokumentübersicht

```
<xsl:template match="/">  
  <xls:result-document href="output_1.xml" >  
    Beschreibung des ersten Ausgabebaumes (z.B.: Gesamtdokument)  
  </xls:result-document>  
  
  <xls:result-document href="output_2.xml">  
    Beschreibung des zweiten Ausgabebaumes (z.B.: Dokumentübersicht)  
  </xls:result-document>  
</xsl:template>
```

## 4. Unterstützung der Ausgabe in XHTML

- in XSLT 1.0 nur:
  - `<xsl:output method="xml"/>` - erzeugt wohlgeformtes XML
  - `<xsl:output method="text"/>` - HTML-Elemente & -Attribute werden erkannt
  - `<xsl:output method="xml"/>` - gibt die String-Werte aller Textknoten, die im Ausgabebaum enthalten sind
- in XSLT 2.0 auch XHTML als Ausgabeformat vorgesehen

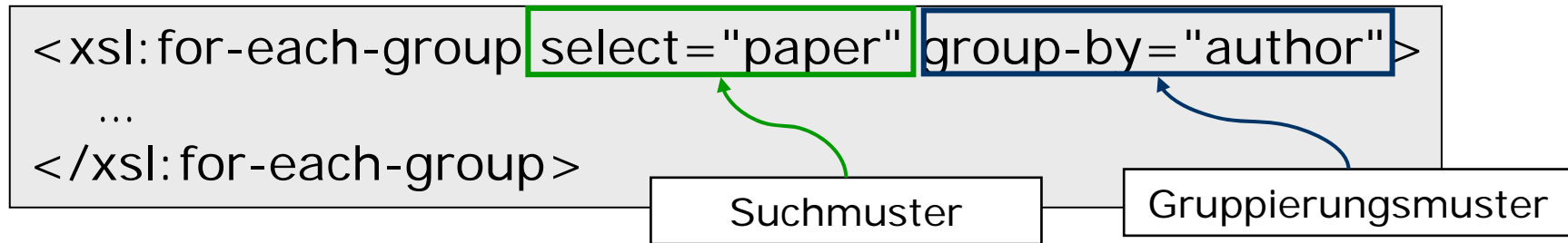
```
<xsl:output method="xhtml"/>
```

## 5. Weitere neue Elemente

- **<xsl:character-map>** – verwendet, um während der Serialisierung bestimmte Zeichen durch andere zu ersetzen
- **<xsl:document>** – erzeugt einen neuen Dokumentknoten
- **<xsl:result-document>** – erweitert die Möglichkeiten von output in XSLT 1.0; erlaubt die Eigenschaften des Ausgabebaum detailliert festzulegen
- **<xsl:sequence>** – verwendet innerhalb eines Sequenzkonstruktors, um eine Sequenz von Knoten oder Einzelwerten zu bilden

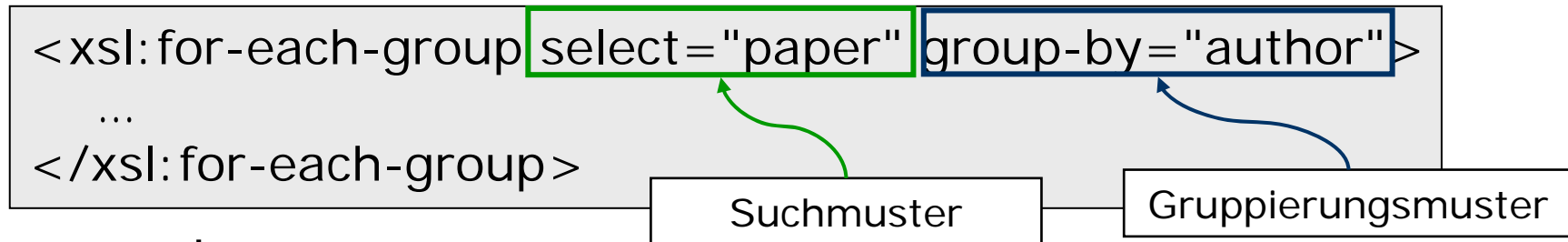


## Gruppieren in XSLT 2.0



- **<xsl:for-each-group>** vereinfacht die Gruppierung von Sequenzen
- **current-group()** – liefert die Inhalte der aktuellen Gruppe in `<xsl:for-each-group>`
- **current-grouping-key()** – liefert den aktuellen Gruppenschlüssel in `<xsl:for-each-group>`

## Attribute in <xsl:for-each-group>



- group-by
  - arbeitet mit Gruppierungsschlüssel
  - fasst alle Datenelemente zusammen, die in Bezug auf diesen Schlüssel denselben Wert liefern
- group-adjacent
  - Fasst alle Datenelemente zusammen, die in Bezug auf diesen Schlüssel denselben Wert liefern und die Datenelemente innerhalb der Eingabesequenz benachbart sind
- group-starting-with / group-ending-with
  - Identifiziert Start / Ende einer Gruppe

# Beispiel: Ursprung & Ergebnis

XML-Ursprungsdokument

```
<paper>
  <title>Publishing Workflows</title>
  <author>Myers, Charles</author>
</paper>
<paper>
  <title>On the Way to XML</title>
  <author>Parsons, Jonathan</author>
  <author>Caisley, Phil</author>
</paper>
```

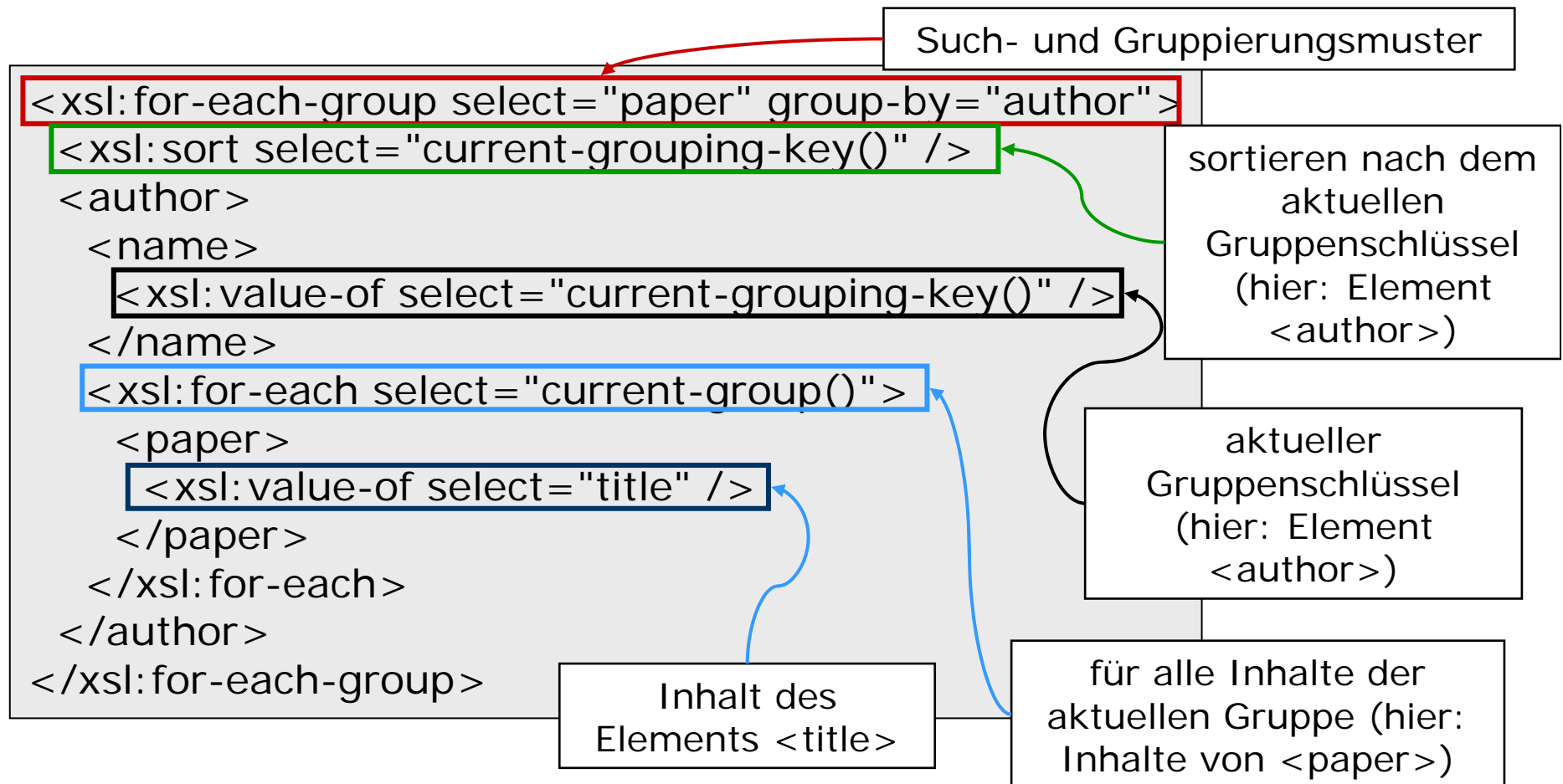
XML-Ergebnisdokument

```
<author>
  <name>Caisley, Phil</name>
  <paper>On the Way to XML</paper>
</author>
<author>
  <name>Myers, Charles</name>
  <paper>Publishing Workflows</paper>
</author>
<author>
  <name>Parsons, Jonathan</name>
  <paper>On the Way to XML</paper>
</author>
```

XSLT



# Beispiel: XSLT-Stylsheet



# group-starting-with: Beispiel

Quelle

```
<h1>Überschrift</h1>
<p>Absatz</p>
<p>Absatz</p>
<h1>Überschrift</h1>
<p>Absatz</p>
<p>Absatz</p>
```

Ziel

```
<section>
  <title>Überschrift</title>
  <para>Absatz</para>
  <para>Absatz</para>
</section>
<section>
  <title>Überschrift</title>
  <para>Absatz</para>
  <para>Absatz</para>
</section>
```

```
<xsl:for-each-group select="*" group-starting-with="h1">
  <section>
    <title>
      <xsl:value-of select="current-group()[self::h1]"/>
    </title>
    <xsl:for-each select="current-group()[self::p]">
      <param><xsl:apply-templates/></param>
    </xsl:for-each>
  </section>
</xsl:for-each-group>
```

Stylesheet

## Vor- und Nachteile von XSLT 2.0

- + mächtiger als XSLT 1.0
- + unterstützt Datentypen von XML-Schema
- + wichtige Funktionen verfügbar, die in Version 1.0 fehlten
- + ermöglicht etwas einfachere und effizientere Entwicklung
- + rückwärts kompatibel
- + flache Lernkurve für die neue Spezifikation
  
- immer noch wenig Prozessoren (Tools), die XSLT 2.0 unterstützen



## **eXtensible Stylesheet Language Formatting Objects (XSL-FO)**

---

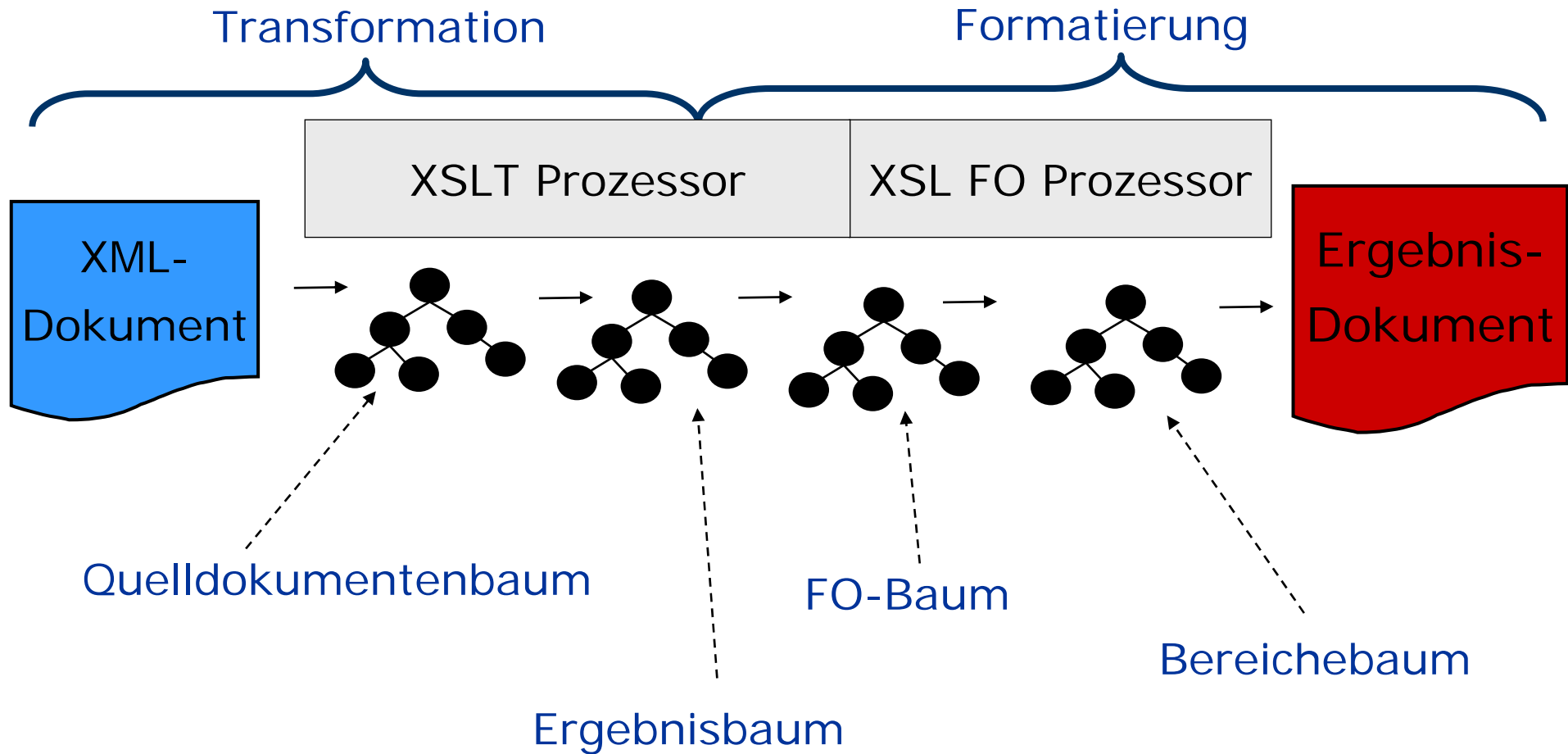
## **XSLT**

- erlaubt Transformation von XML → HTML
- ungeeignet für druckfähige Formatierungen (PDF, RTF)

## **XSL-FO**

- erlaubt XML-Dokumente mit druckfähigen Layout zu versehen
- Transformation XML → PDF oder RTF möglich
- basiert auf auf Cascading Style Sheets (CSS2)
- W3C-Standard von 2001

# Mehrstufiger XSL-Prozess



Quelle: H. Vonhoegen „Einstig in XML: Grundlagen, Praxis, Referenzen“, ISBN 978-3-8362-1074-4, 2007

<b>CSS</b>	<b>XSL-FO</b>
Darstellung auf <b>Bildschirm</b>	Darstellung auf <b>seitenorientiertem</b> Ausgabemedium
Ausgabe durch <b>Webbrowser</b>	Ausgabe durch <b>Drucker</b> und andere <b>Seitenausgabegeräte</b>
Formatierungsinformation für vorhandenes <b>Markup</b>	Komplette Ersetzung von Markup durch - <b>Formatierungsmarkup</b>

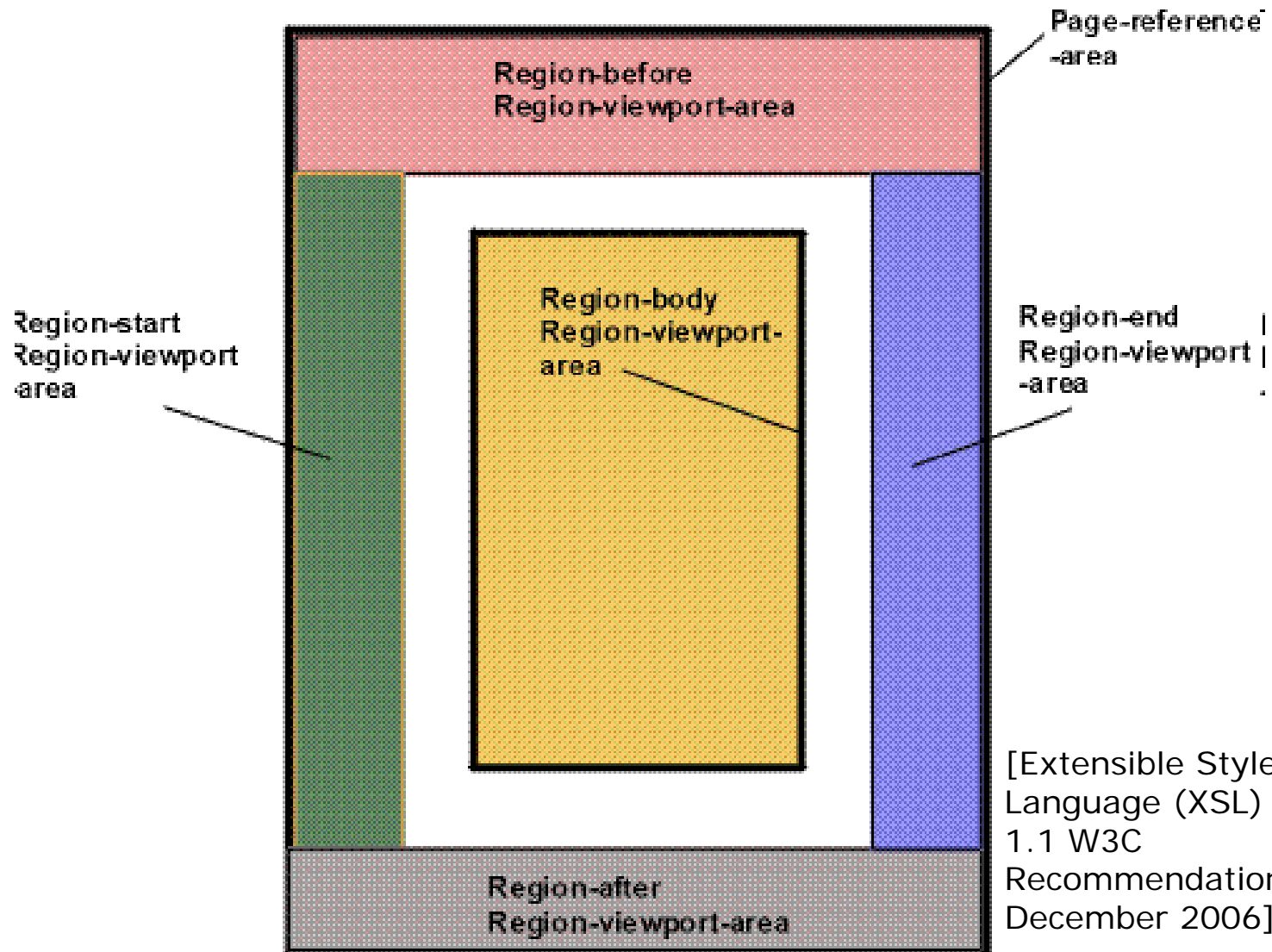
- Massensatz, z.B.: bei der technischen Dokumentation
- gleichzeitige Ausgabe derselben Inhalte in unterschiedlichen Formaten:
  - verschiedene Medien
  - gleiches Medium aber verschiedene Bedürfnisse der Nutzer
- Individualisierung bzw. Personalisierung von Dokumenten

- Wie in Office-Programmen Vorlagen für Seitenstruktur:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format" >
  <fo:layout-master-set >
    <fo:simple-page-master master-name="my-page" >
      <fo:region-body margin="1in"/>
    </fo:simple-page-master >
  </fo:layout-master-set >

  <fo:page-sequence master-reference="my-page" >
    <fo:flow flow-name="xsl-region-body" >
      <fo:block>Hello, world! </fo:block>
    </fo:flow >
  </fo:page-sequence >
</fo:root >
```

Quellenhinweis: XSL-FO Beispiele auf den folgenden Folien aus Nikolai Grigoriev. XSL Formatting Objects Tutorial. <http://www.renderx.com/tutorial.html>



[Extensible Stylesheet Language (XSL) Version 1.1 W3C Recommendation 05 December 2006]

- CSS artige Darstellungseigenschaften:

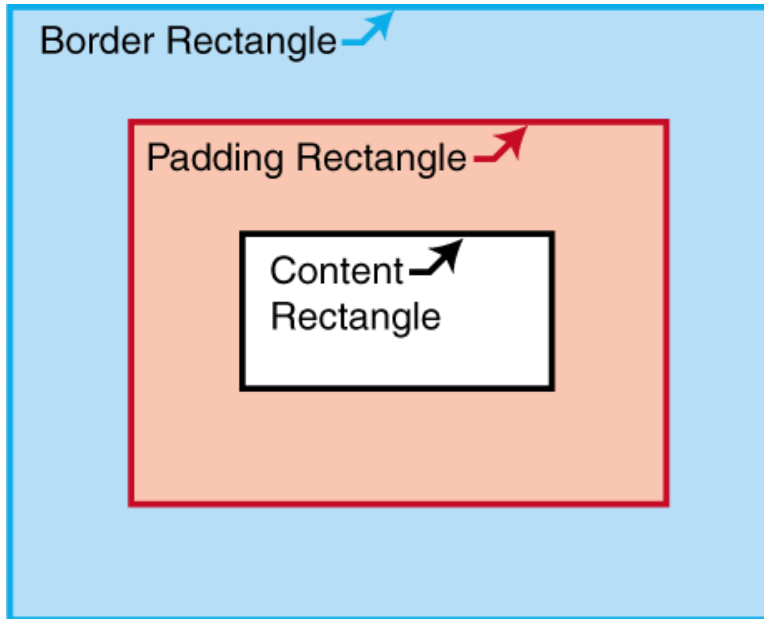
```
<fo:block font="italic 14pt Times" >  
  <fo:inline color="red" >H</fo:inline>ello,  
  <fo:inline font-weight="bold" >world!</fo:inline>  
</fo:block>
```

- Für Blöcke:

```
<fo:block text-align="justify" text-indent="1in"  
  text-align-last="end" last-line-end-indent="1in" >
```

This is an example of double-justified text with an indented first line. The last line of the text is aligned to the right, and indented by 1 inch from the right.

```
</fo:block>
```



[Extensible Stylesheet Language (XSL) Version 1.1  
W3C Recommendation 05 December 2006]

```
<fo:block border="thin solid navy"  
  text-align="center"  
  padding-before="18pt" padding-bottom="18pt">  
  <fo:block border="thin solid maroon">  
    The outer block has a 18 pt padding from top and bottom  
  </fo:block>  
</fo:block>
```

```
<fo:list-block provisional-distance-between-starts="18pt"  
    provisional-label-separation="3pt">
```

```
<fo:list-item>
```

```
<fo:list-item-label end-indent="label-end()">
```

```
<fo:block>#x2022;</fo:block>
```

```
</fo:list-item-label>
```

```
<fo:list-item-body start-indent="body-start()">
```

```
<fo:block>First item</fo:block>
```

```
</fo:list-item-body>
```

```
</fo:list-item>
```

```
<fo:list-item>
```

```
<fo:list-item-label end-indent="label-end()">
```

```
<fo:block>#x2022;</fo:block>
```

```
</fo:list-item-label>
```

```
<fo:list-item-body start-indent="body-start()">
```

```
<fo:block>Second item</fo:block>
```

```
</fo:list-item-body>
```

```
</fo:list-item>
```

```
</fo:list-block>
```

```
<fo:table border="0.5pt solid black" text-align="center">
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell padding="6pt" border="0.5pt solid black">1
        <fo:block> upper left </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> upper right </fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> lower left </fo:block>
      </fo:table-cell>
      <fo:table-cell padding="6pt" border="0.5pt solid black">
        <fo:block> lower right </fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>
```



**Warum XSLT 2.0 (nicht) benutzen?**

---

# Wie geht es weiter?

---

## heutige Vorlesung

- ☑ XSLT-Editoren
- ☑ mehrere Ursprungsdokumente
- ☑ Gruppieren
- ☑ XSLT 2.0
- ☑ XSL-FO

## Vorlesung nächste Woche

- neuer Block → Web Services