

# XSLT – wichtige Elemente

## Templates

`<xsl:template match="XPath-Muster" name="String" mode="String">`

- Eine Schablone, die alles verarbeitet, was auf das XPath-Muster passt. Schablonen können benannt sein oder einen bestimmten Modus haben.
- Mindestens @match oder @name muss vorhanden sein.

`<xsl:apply-templates select="XPath-Ausdruck" mode="String" />`

- Aufruf in einer Schablone, der besagt, dass für diese Eingabedaten noch weitere Schablonen berücksichtigt werden sollen.
- Optional mit @select: für welche Elemente genau sollen weitere Schablonen berücksichtigt werden?
- Optional mit @mode: nur Schablonen mit diesem Modus sollen berücksichtigt werden

## Wertausgabe & Kopieren

`<xsl:value-of select="XPath-Ausdruck" />`

- Schreibt den Wert (den Text) von bestimmten Teilen des Eingabedokuments (@select) in das Ergebnisdokument
- Kurzsyntax innerhalb von Attributkonstruktionen: `attribut= "{XPath-Ausdruck}"`

`<xsl:copy-of select="XPath-Ausdruck" />`

- Kopiert die mit @select ausgewählten Elemente vollständig (mit allen Attributen und Kindelementen)

## Schleifen & Sortieren

`<xsl:for-each select="XPath-Ausdruck">`

- Mit jedem Element, das vom XPath-Ausdruck zurückgegeben wird, soll etwas getan werden

`<xsl:sort select="XPath-Ausdruck" data-type="text | number" order="ascending | descending" case-order="upper-first | lower-first" lang="de | en | ..." />`

- Sortiert die Knotenmenge, die mit @select ausgewählt wird
- Kann vorkommen in: `<xsl:for-each>`, `<xsl:apply-templates>`
- Wird als leeres Element direkt nach dem öffnenden Tag von z.B. `for-each` notiert

## Bedingungen

`<xsl:if test="XPath-Ausdruck">`

- Bedingung: die in `<xsl:if>` enthaltenen Anweisungen werden nur ausgeführt, wenn der Ausdruck in @test wahr ist

`<xsl:choose>`

`<xsl:when test="XPath-Ausdruck"></xsl:when>`

`<xsl:otherwise></xsl:otherwise>`

`</xsl:choose>`

- Es werden mehrere Bedingungen nacheinander (`<xsl:when test=" ">`) überprüft und die Anweisungen der ersten erfüllten Bedingung ausgeführt.
- Trifft keine der Bedingungen in @test zu, werden die Anweisungen in `<xsl:otherwise>` ausgeführt.

## Zieldokumente

`<xsl:result-document href="String">`

- Erstellen eines Zieldokuments
- In `@href` können der Pfad und Dateiname des Zieldokuments angegeben werden
- Dadurch Erstellen mehrere Zieldokumente möglich

`<xsl:output method="html | text | xhtml | xml" encoding=" UTF-8 | ISO-8859-1" indent="yes | no">`

- Gibt an, wie der Ergebnisbaum/das Zieldokument ausgegeben werden soll
- Top-Level-Element!
- `@method`: Angabe des Formats
- `@encoding` gibt die Zeichencodierung an
- `@indent`: bei „yes“ werden die Elemente im Ergebnisbaum eingerückt (für lesbaren Quelltext)

## Leerraum

`<xsl:strip-space elements="String"/>`

- Leerraum, der im Ausgangsdokument zwischen Elementen stand, wird entfernt
- In `@elements` wird eine Liste von Elementnamen angegeben, getrennt durch Leerzeichen
- Top-Level-Element

`<xsl:preserve-space elements="String"/>`

- Leerraum, der im Ausgangsdokument zwischen Elementen stand, wird beibehalten
- In `@elements` wird eine Liste von Elementnamen angegeben, getrennt durch Leerzeichen
- Top-Level-Element