Inhalt

Einleitung

Die Template-Engine verarbeitet neben der normalen pagina-XML Instanz zusätzlich sogenannte TPL-Templates. Dies sind XHTML-Dateien mit der Dateiendung *.tpl, die entweder reihenspezifisch in einem EPUB-ConfigSetting oder titelspezifisch neben der XML-Datei abgelegt sind. Sie liegen bereits im Zielformat XHTML vor und sind nicht in pagina-XML-Syntax verfasst. Dennoch werden die TPL-Templates vom EPUB-Konverter geparsed – genau genommen werden sie von der TPL-Engine prozessiert und verarbeitet.

Sinn und Zweck

Sinn und Zweck der TPL-Templates ist das Einbringen von statischen oder variablen Inhalten die z.B. reihenübergreifend in jedes EPUB eingebracht werden sollen. Gemäß der Tagging-Konventionen sollten solche Inhalte NICHT in die XML-Daten selbst eingebracht werden.

Beispiel: ein Verlags-Hinweis zum EPUB-Format, der in jedes E-Book als letzte Seite eingebunden werden soll.

Neben statischen Inhalten können TPL-Templates auch Variablen enthalten. Diese Variablen sind Platzhalter die während der EPUB-Generierung von der Template-Engine mit Inhalt aus der XML-Instanz befüllt werden. Soll im Text eines TPL-Templates bspw. auf den Titel und den Autor des Buches verwiesen werden, so können dafür folgende Platzhalter verwendet werden:

```
Wir hoffen, Ihnen hat "{{TITEL}}" von {{AUTOR}} gefallen, und Sie...
```

Durch die Template-Engine wird dies – bspw. für Schillers Wilhelm Tell – wie folgt aufgelöst:

Wir hoffen, Ihnen hat "Wilhelm Tell" von Friedrich Schiller gefallen, und Sie...

So gut wie alle Meta-Elemente der parsX-DTD lassen sich als Variable in den TPL-Templates definieren. Sie folgen allerdings einer besonderen Syntax, weshalb die Lektüre der folgenden Seiten unbedingt empfohlen wird um selbst TPL-Templates zu schreiben.

TPL-Templates vs. generierte Zusatzseiten

Als generierte Zusatzseiten werden im EPUB-Konverter die Seiten bezeichnet, die aus den E-Book-Metadaten (<titelei_ebook>) der XML-Instanz vor oder nach dem eigentlichen Content erstellt werden. Dazu gehören z.B. das Cover, die Titelseite, das Inhaltsverzeichnis oder das Impressum. Diese generierten Seiten sind in einer festen Reihenfolge angeordnet und lassen sich auch über das EPUB-ConfigFile nicht umsortieren. Ein Impressum im "Vorspann", also vor dem Buch-Content, war bis zur Einführung der Template-Engine nicht möglich.

Diese generierten Seiten lassen sich allerdings im EPUB-ConfigFile abschalten und nun komplett durch TPL-Templates ersetzen:

- Sie können dadurch beliebigen Inhalt VOR oder NACH den XML-Content platzieren.
- Sie können für jedes TPL-Template bestimmen ob es im TOC aufgeführt werden soll oder nicht.
- Sie können für jedes TPL-Template Bedingungen definieren die zutreffen müssen, damit das Template verarbeitet wird.

> **Beispiel**
>
> Das TPL-Template "Über den Autor" soll nur verarbeitet und eingebunden werden, wenn das entsprechende Meta-Element <meta_copyright> auch Inhalt enthält.

Grundlagen zur Verwendung von TPL-Templates

Um in einem EPUB-ConfigSetting ein TPL-Template zu verwenden, gehen Sie wie folgt vor:

- Erstellen Sie in oXygen eine neue XML- oder HTML-Datei, bspw. über STRG+N (Windows) / CMD+N (Mac).
- Löschen Sie allen vordefinierten Inhalt um eine leere Datei zu erhalten.
- Fügen Sie den folgenden Inhalt per Copy & Paste in die leere Datei ein

```
<?xml version="1.0" encoding="UTF-8"?>
 <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
           <meta http-equiv="Content-Type"
content="application/xhtml+xml; charset=UTF-8" />
           <title>TOC-Eintrag</title>
           <link rel="stylesheet" type="text/css"
href="stylesheet.css" id="style-standard" />
      </head>

      <body class="meta_XXX">

           <!-- hier kann nun weiterer Inhalt folgen -->

      </body>
</html>
```

Wichtig

In der TPL-Datei darf kein DOCTYPE deklariert sein! Diese wird automatisch ergänzt!

> In der TPL-Datei dürfen keine benannten HTML-Entities (z.B.) verwendet werden!

- Speichern Sie die Datei im OEBPS/-Verzeichnis des EPUB-ConfigSettings unter dem folgenden Namen: 10-test.xhtml.vorspann.tpl

> **Wichtig**
>
> Achten Sie beim Speichern in oXygen darauf, dass Sie den Dateinamen komplett mit Endung angeben und oXygen nicht noch ein zusätzliches ".xhtml" anhängt!

- Wenn Sie nun aus Ihrer XML-Instanz ein EPUB generieren, dann wird das TPL-Template im Vorspann zwischen Navigationsseite und Inhaltsübersicht eingebunden.
  - Um es an anderer Stelle im Vorspann einzubinden, müssen Sie die entsprechenden generierten Seiten im EPUB-ConfigFile ausschalten und ebenfalls als TPL-Template umsetzen.
  - Über die Nummer am Anfang des Dateinamens kann die Reihenfolge der TPL-Templates geändert werden.

Dateinamenskonvention

Der Dateiname `10-test.xhtml.vorspann.tpl` aus dem vorigen Beispiel folgt einer logischen Konvention:

- `10-` : Laufende Nummer gefolgt von einem Bindestrich. Angabe ist optional. Dient der Definition der Reihenfolge der TPL-Templates. Es gelten die folgenden Regeln:
  - Falls keine Nummer angegeben ist, wird alphabetisch sortiert.
  - Es kann keine natürliche Sortierung angewendet werden.

    Das heißt, es wird wie folgt sortiert: `10, 30, 4, 40` und nicht

wie wir erwarten würden: 4, 10, 30, 40. Verwenden Sie deshalb bei einstelligen Ziffern immer eine führende 0 um eine korrekte Sortierung zu erhalten 04, 10, 30, 40

- test : Dateiname. Angabe ist verpflichtend. Die Wahl des Dateinamens bleibt Ihnen überlassen. Es wird empfohlen, sprechende Angaben wie "haupttitel", "schmutztitel", etc. und keine Sonder- oder Leerzeichen zu verwenden.
- .xhtml : XHTML-Dateiendung. Angabe ist verpflichtend. Muss immer .xhtml lauten!
- .vorspann : Hinweis zur Positionierung des TPL-Templates im Buch. Angabe ist optional. Es sind folgende Werte erlaubt:
  - *keine Angabe*: Das TPL-Template wird im Nachspann eingeordnet. Der Übersichtlichkeit halber empfehlen wir, immer einen der folgenden Werte 'vorspann' oder 'nachspann' zu verwenden!
  - .vorspann : Das TPL-Template wird im Vorspann eingeordnet.
  - .nachspann : Das TPL-Template wird im Nachspann eingeordnet.
- .tpl : TPL-Dateiendung. Angabe ist verpflichtend. Muss immer auf .tpl enden!

Steuerung des TOC-Eintrags

Für die Steuerung des TOC-Eintrags ist das HTML-Element <title> in der TPL-Datei zuständig. Ist es vorhanden und befüllt, wird automatisch ein TOC-Eintrag erstellt dessen Wert dem des <title>-Elements entspricht.

Beispiel 1.1. TOC-Eintrag aus <title>

```
<head>
   <title>Über das Buch</title>
   [...]
</head>
```

Ergebnis: Der TOC-Eintrag dieses TPL-Templates lautet im EPUB "*Über das Buch*"

Ebenso ist es möglich, Textvariablen im <title>-Element zu verwenden:

Beispiel 1.2. TOC-Eintrag mit Textvariable

```
<head>
   <title>Über {{AUTOR}}</title>
   [...]
</head>
```

Ergebnis: Der TOC-Eintrag dieses TPL-Templates lautet im EPUB dann bspw. "*Über Friedrich Schiller*"

Kein TOC-Eintrag erzeugen

Wird das `<title>`-Element leer gelassen (es darf nicht ganz fehlen, sonst ist die XHTML-Datei invalide), wird kein TOC-Eintrag erzeugt:

Beispiel 1.3. TOC-Eintrag mit Textvariable

```
<head>
   <title></title>
   [...]
</head>
```

Ergebnis: Die Seite taucht zwar an entsprechender Stelle im Lesefluss auf, wird aber im TOC nicht erwähnt.

Prüfbedingungen

Mit Hilfe einer Prüfbedingung bzw. eines Checks kann über die Einbindung der gesamten Seite oder eines einzelnen Elements in das EPUB entschieden werden.

Nützlich ist dies für Seiten, die abhängig vom Inhalt oder Wert eines bestimmten XML-Elements sein sollen. Ein gutes Beispiel hierfür ist die Autorenseite die nur generiert werden soll,

wenn <meta_bio> auch mit Inhalt befüllt ist.

Oder ein Leseproben-Zusatz wie *"Unverkäufliche Leseprobe zu:"* der nur angezeigt werden soll, wenn auch gerade eine Leseprobe produziert wird.

Inhaltsprüfungen (<check>)

Zum Festlegen einer Prüfbedingung wird im <head>-Element der TPL-Templatedatei ein <check>-Element eingefügt. Das Element muss mindestens das valueOf-Attribut besitzen:

- @valueOf="" – Mit diesem Attribut wird der Name des Elementes in der XML-Instanz angegeben, dessen Inhalt geprüft werden soll.

> **Hinweis**
>
> Als Prüfbedingung für Element-Inhaltsprüfungen können nur die TPL-Element-Aufrufe (vollständige Liste im folgenden Kapitel) verwendet werden!
>
> XPath-Abfragen sind nicht erlaubt!

- @valueOf_empty="(true|false)" – Mit diesem optionalen Attribut kann angegeben werden, dass das, durch das @valueOf-Attribut überprüfte XML-Element, leer (Attributwert true) oder nicht leer (Attributwert false) sein muss.

- @valueOf_equal="" – Mit diesem optionalen Attribut kann angegeben werden, dass der Inhalt des, durch das @valueOf-Attribut überprüften XML-Elements, dem hier angegebenen Attributwert gleichen muss.

> **Hinweis**
>
> Da man in Attributwerten keine Tags verwenden kann, macht diese Prüfungen nur für Plain-Text-Abfragen Sinn (z.B. ISBNs, Namen*, etc.)*.

@valueOf_notEqual="" – Mit diesem optionalen Attribut kann angegeben werden, dass der Inhalt des, durch das @valueOf-Attribut überprüften XML-Elements, ungleich(!) dem hier angegebenen Attributwert sein muss.

> **Hinweis**
>
> Da man in Attributwerten keine Tags verwenden kann, macht diese Prüfungen nur für Plain-Text-Abfragen Sinn (z.B. ISBNs, Namen, etc.).

Beispiel 1.4. Prüfung ob das XML-Element <meta_bio> Inhalt enthält

```
<head>
   [...]
   <check valueOf="meta_bio"/>
</head>
```

Oder ab parsX 3.3:

```
<head>
   [...]
   <check valueOf="meta_bio" valueOf_empty="false"/>
</head>
```

Falls <meta_bio> Inhalt enthält, wird das Template ganz normal verarbeitet und ins EPUB aufgenommen.

Falls <meta_bio> nicht existiert oder leer ist, wird das TPL-Template nicht weiter verarbeitet und somit auch nicht ins EPUB aufgenommen. Im Logfile des EPUB-Konverters wird darüber informiert.

Beispiel 1.5. Prüfung ob die E-Book-ISBN einem bestimmten Wert gleicht

```
<head>
   [...]
   <check valueOf="meta_e-isbn" valueOf_equal="978-3-455-
50250-3"/>
</head>
```

Falls die E-Book-ISBN "978-3-455-50250-3" lautet, wird das Template ganz normal verarbeitet und ins EPUB aufgenommen.

Falls nicht, wird das TPL-Template nicht weiter verarbeitet und somit auch nicht ins EPUB aufgenommen. Im Logfile des EPUB-Konverters wird darüber informiert.

Beispiel 1.6. Prüfung ob die Print-ISBN nicht einem bestimmten Wert gleicht

```
<head>
   [...]
   <check valueOf="meta_isbn" valueOf_notEqual="978-3-455-
50250-3"/>
</head>
```

Falls die Print-ISBN nicht "978-3-455-50250-3" lautet, wird das Template ganz normal verarbeitet und ins EPUB aufgenommen.

Falls hingegen die Print-ISBN genau diesem Wert entspricht, wird das TPL-Template nicht weiter verarbeitet und somit auch nicht ins EPUB aufgenommen. Im Logfile des EPUB-Konverters wird darüber informiert.

Leseprobenmodus-Prüfung (<check_modus>)

Seit parsX 3.3 kann mit Hilfe des neuen Elements <check_modus> im <head>-Element der TPL-Templatedatei geprüft werden, ob der EPUB-Konverter derzeit im "normalen Modus" ausgeführt wird, oder im "Leseprobe-Modus".

Beispiel 1.7. Prüfung ob Modus=Leseprobe

```
<head>
   <title>Über {{AUTOR}}</title>
   <check_modus if="modus-leseprobe"/>
   [...]
</head>
```

Falls gerade eine Leseprobe produziert wird, wird das Template ganz normal verarbeitet und ins EPUB aufgenommen.

Falls keine Leseprobe produziert wird, wird das TPL-Template nicht weiter verarbeitet und somit auch nicht ins EPUB aufgenommen. Im Logfile des EPUB-Konverters wird darüber informiert.

Erlaubt sind die folgenden Werte für das `if`-Attribut:

```
modus-leseprobe | modus-normal
```

Abbildungsprüfung (<check_image>)

Seit parsX 3.3 kann mit Hilfe des neuen Elements <check_image> im <head>-Element der TPL-Templatedatei geprüft werden, ob eine Abbildung (nicht) existiert.

Beispiel 1.8. Prüfung ob Abbildung existiert

```
<head>
   [...]
   <check_image name="tpl-anzeige.jpg" exists="true" />
</head>
```

Oder in Kurzschreibweise:

```
<head>
   [...]
   <check_image name="tpl-anzeige.jpg" />
</head>
```

Beide Beispiele prüfen die **Verfügbarkeit** der Abbildung tpl-anzeige.jpg im XML-Abbildungsordner UND im ConfigSetting-Image-Ordner.

Wird die Abbildung an einem der beiden Orte gefunden wird das TPL-Template eingebunden, andernfalls nicht.

Beispiel 1.9. Prüfung ob Abbildung *nicht* existiert

```
<head>
   [...]
   <check_image name="tpl-anzeige.jpg" exists="false" />
</head>
```

Dieses Beispiel prüft die **Nicht-Verfügbarkeit** der Abbildung tpl-anzeige.jpg sowohl im XML-Abbildungsordner als auch im ConfigSetting-Image-Ordner.

Wenn die Abbildung in beiden Ordnern (also im EPUB) fehlt, dann wird das TPL-Template eingebunden, andernfalls nicht.

Tipp

Ein möglicher Anwendungsfall wäre ein Text-TPL-Template als Fallback für ein Bild-Anzeigen-Template das nur eingebunden wird, wenn die Bild-Anzeige fehlt.

If-Prüfbedingungen für Content-Elemente

Seit parsX 3.3 können If-Prüfbedingungen auch auf Element-Ebene vorgenommen werden. So ist es z.B. möglich, einen Leseprobenzusatz (z.B. "Unverkäufliche Leseprobe zu:") nur dann auf dem Haupttitel anzuzeigen, wenn auch eine Leseprobe produziert wird.

Beispiel 1.10.

```
<body>
   <p class="t_leseprobe-zusatz" if="modus-leseprobe">Unverkäufliche Leseprobe zu:</p>
   <p class="t_leseprobe-zusatz" if="modus-normal">Volle E-Book-Ausgabe!</p>
   [...]
</body>
```

Der erste Absatz wird nur übernommen, wenn gerade eine Leseprobe produziert wird. Ansonsten wird er entfernt.

Beim zweiten Absatz verhält es sich genau umgekehrt.

Erlaubt sind die folgenden Werte für die If-Prüfbedingung:

`modus-leseprobe | modus-normal`

Nutzung variabler Inhalte

Die folgenden Kapitel beschreiben die Verwendung von Textvariablen, Element-Abfragen und spezieller XSLT-Templates.

Textvariablen

Textvariablen in doppelt-geschweiften Klammern (z.B. {{TITEL}}) geben den Inhalt des entsprechenden Metadatenfelds in Plain-Text zurück.

Dabei greift die Template-Engine ausschließlich auf die E-Book-Metadaten aus dem Element <titelei_ebook> zu. Einzige Ausnahme ist hier die Print-ISBN aus den "normalen" Metadaten die mit {{ISBN}} oder {{ISBN-clean}} abgefragt werden kann.

> **Achtung**
>
> Es wird kein MarkUp generiert! Inline-Auszeichnungen werden nicht ausgegeben!
>
> Enthält z.B. das Element <meta_titel> einen Zeilenumbruch mittels
-Element, so wird er NICHT ausgegeben!

> **Anmerkung**
>
> Erlaubt sind die folgenden Textvariablen:
>
> {{TITEL}} | {{UNTERTITEL}} | {{ISBN}} | {{ISBN-clean}} | {{eISBN}} | {{eISBN-clean}} | {{DOI}} | {{URN}} | {{ID}} | {{LOVELYBOOKS.ALLEAUTOREN}} | {{LOVELYBOOKS.AUTORENLINK}} | {{UEBERSETZER}} | {{UEBERSETZER1}} | {{UEBERSETZER2}} | {{UEBERSETZER3}} | {{UEBERSETZER4}} | {{UEBERSETZER5}} | {{ILLUSTRATOREN}} | {{ILLUSTRATOR}} | {{ILLUSTRATOR1}} | {{ILLUSTRATOR2}} |

{{ILLUSTRATOR3}} | {{ILLUSTRATOR4}} | {{ILLUSTRATOR5}} | {{GATTUNG}} | {{eLOGO}} | {{LOGO}} | {{HERAUSGEBER}} | {{HERAUSGEBER1}} | {{HERAUSGEBER2}} | {{HERAUSGEBER3}} | {{HERAUSGEBER4}} | {{HERAUSGEBER5}} | {{AUTOREN}} | {{AUTOR}} | {{AUTOR1}} | {{AUTOR2}} | {{AUTOR3}} | {{AUTOR4}} | {{AUTOR5}} | {{CO-AUTOREN}} | {{CO-AUTOR1}} | {{CO-AUTOR2}} | {{CO-AUTOR3}} | {{CO-AUTOR4}} | {{CO-AUTOR5}} | {{DL-NAME}} | {{DATUM-SPERRFRIST}} | {{DATUM-SPERRFRIST-FREITEXT}} | {{DATUM-ERSCHEINUNGSTERMIN}} | {{DATUM-ERSCHEINUNGSTERMIN-FREITEXT}}

Beispiel 1.11. Verwendung im Fließtext
TPL-Template:
Die E-Book-ISBN dieses Titels lautet: {{eISBN}}
HTML-Ausgabe im EPUB:
Die E-Book-ISBN dieses Titels lautet: 978-3-844-23718-4

Beispiel 1.12. MarkUp wird entfernt
pagina-XML:
<meta_titel>Ein <kursiv>ziemlich</kursiv> langer Beispieltitel
für ein E-Book!</meta_titel>
TPL-Template:
Der Titel dieses Buches lautet: {{TITEL}}
HTML-Ausgabe im EPUB:
Der Titel dieses Buches lautet: Ein ziemlich langer Beispieltitelfür ein E-Book!

Achtung

MarkUp wird ersatzlos entfernt. Dadurch kann es zu Darstellungsfehlern kommen! Mit einem Leerzeichen vor dem

-Element könnte man dies umgehen. Eleganter ist allerdings die Verwendung von Element-Abfragen (siehe nächstes Kapitel).

Da kein MarkUp generiert wird, können Textvariablen auch in HTML-Attributen verwendet werden:

Beispiel 1.13. Verwendung in Attributen

```
<p><a href="http://www.lovelybooks.de/m/epub/{{ISBN-clean}}/ihre-meinung/">Schreiben Sie hier Ihre Meinung zum Buch</a></p>
```

HTML-Ausgabe im EPUB:

```
<p><a href="http://www.lovelybooks.de/m/epub/9783844237184/ihre-meinung/">Schreiben Sie hier Ihre Meinung zum Buch</a></p>
```

Element-Abfragen

Um Schwächen der Textvariablen zu umgehen, sind seit Version 2 der Template-Engine Element-Abfragen möglich.

Element-Abfragen werden über das Attribut `@valueOf="element"` an einem beliebigen HTML-Element im TPL-Template erzeugt.

Dabei greift die Template-Engine ausschließlich auf die E-Book-Metadaten aus dem Element `<titelei_ebook>` zu. Einzige Ausnahme ist hier die Print-ISBN aus den "normalen" Metadaten die mit `meta_isbn` abgefragt werden kann.

Anmerkung

Erlaubt sind die folgenden Element-Aufrufe:

meta_titel | meta_untertitel | meta_e-isbn | meta_isbn | meta_doi | meta_urn | meta_id | meta_uebersetzer-abs | meta_uebersetzer-1 | meta_uebersetzer-2 | meta_uebersetzer-3 | meta_uebersetzer-4 | meta_uebersetzer-5 |

meta_illustratoren-abs | meta_illustrator-1 | meta_illustrator-2 | meta_illustrator-3 | meta_illustrator-4 | meta_illustrator-5 | meta_gattung | meta_herausgeber-abs | meta_herausgeber-1 | meta_herausgeber-2 | meta_herausgeber-3 | meta_herausgeber-4 | meta_herausgeber-5 | meta_autoren-abs | meta_autor-1 | meta_autorvorname-1 | meta_autornachname-1 | meta_autor-2 | meta_autorvorname-2 | meta_autornachname-2 | meta_autor-3 | meta_autorvorname-3 | meta_autornachname-3 | meta_autor-4 | meta_autorvorname-4 | meta_autornachname-4 | meta_autor-5 | meta_autorvorname-5 | meta_autornachname-5 | meta_co-autoren-abs | meta_co-autor-1 | meta_co-autorvorname-1 | meta_co-autornachname-1 | meta_co-autor-2 | meta_co-autorvorname-2 | meta_co-autornachname-2 | meta_co-autor-3 | meta_co-autorvorname-3 | meta_co-autornachname-3 | meta_co-autor-4 | meta_co-autorvorname-4 | meta_co-autornachname-4 | meta_co-autor-5 | meta_co-autorvorname-5 | meta_co-autornachname-5 | meta_joker-1 | meta_joker-2 | meta_joker-3 | meta_joker-4 | meta_joker-5 | meta_copyright | meta_bio | meta_kurztext | meta_dl-name | meta_datum-sperrfrist | meta_datum-sperrfrist-freitext | meta_datum-erscheinungstermin | meta_datum-erscheinungstermin-freitext

parsX 3.0 - veraltete Element-Aufrufe

Mit parsX 3.0 wurden im Zuge der Metadaten-Überarbeitung einige Element-Aufrufe als "veraltet" markiert. Diese funktionieren auch weiterhin noch, erzeugen im EPUB-Log jedoch eine Warnung. Im folgenden Eine vollständige Liste mit Ersetzungen:

- doi => meta_doi
- urn => meta_urn
- meta_e-copyright => meta_copyright
- meta_e-bio => meta_bio
- meta_e-kurztext => meta_kurztext
- meta_e-joker-1 => meta_joker-1
- meta_e-joker-2 => meta_joker-2
- meta_e-joker-3 => meta_joker-3
- meta_e-joker-4 => meta_joker-4
- meta_e-joker-5 => meta_joker-5
- meta_illustrator => meta_illustrator-1
- meta_uebersetzer => meta_uebersetzer-1
- meta_hrsg-1 => meta_herausgeber-1
- meta_hrsg-2 => meta_herausgeber-2
- meta_hrsg-3 => meta_herausgeber-3
- meta_hrsg-4 => meta_herausgeber-4
- meta_hrsg-5 => meta_herausgeber-5

Beispiel 1.14. Ausgabe des Titels

HTML-Ausgabe im EPUB:
Buchtitel

Beispiel 1.15. Ausgabe des Titels als Überschrift 6. Ebene
<h6>Titel ist: ""</h6>
HTML-Ausgabe im EPUB:

```
<h6>Titel ist: "<span>Buchtitel</span>"</h6>
```

Falls eine Element-Abfrage kein Ergebnis liefert, wird das Element welches das @valueOf-Attribut enthält, entfernt.

Beispiel 1.16. Leere Ergebnis-Elemente werden gelöscht

```
<div valueOf="meta_joker-5"/>
```

HTML-Ausgabe im EPUB:

Falls dieses Verhalten nicht erwünscht ist, das Element also trotzdem leer erzeugt werden soll, kann das Attribut @preserveEmptyElement="true" gesetzt werden:

Beispiel 1.17. Leere Ergebnis-Elemente erhalten

```
<div valueOf="meta_joker-5" preserveEmptyElement="true"/>
```

HTML-Ausgabe im EPUB:

```
<div></div>
```

@class Attribute am abfragenden Element werden übernommen:

Beispiel 1.18. Attribute werden übernommen

```
<div valueOf="meta_untertitel" class="t_untertitel"/>
```

HTML-Ausgabe im EPUB:

```
<div class="t_untertitel">Buch-Untertitel</div>
```

Achtung

Fehlerhafte Element-Abfragen erzeugen im Oxygen-Logfile UND im EPUB eine rot hinterlegte Fehlermeldung.

Beispiel 1.19. Fehlerhafte Abfrage

HTML-Ausgabe im EPUB:
<span style="background-color:
red;">FEHLER! Es sind nur die folgenden
Element-Aufrufe erlaubt: `meta_titel`, `meta_untertitel`,
[...]
Anzeige im EPUB:

Liefern Sie niemals ein EPUB aus, welches eine solche Fehlermeldung enthält!

Aufruf spezieller XSLT-Templates

XSLT-Templates kapseln vorgefertigte komplexe XPath- und XSLT-Abfragen, die mit den oben genannten Möglichkeiten der Element-Abfragen nicht umzusetzen sind. Dazu gehört unter anderem die Ausgabe des verlinkten Inhaltsverzeichnisses.

Template-Aufrufe werden über das spezifische Attribut @callTemplate="template-name" an einem beliebigen HTML-Element im TPL-Template erzeugt.

Erlaubt sind lediglich die Aufrufe der folgenden Templates:

- alleAutoren
  - Listet alle Autoren (inkl. MarkUp), getrennt durch das im ConfigFile für Urheber definierte Trennzeichen (Abschnitt "Einstellungen für den Haupttitel" => "Trennzeichen für mehrere Autoren und Urheber")
- alleCoAutoren
  - Listet alle Co-Autoren (inkl. MarkUp), getrennt durch das im ConfigFile für Urheber definierte Trennzeichen (Abschnitt

"Einstellungen für den Haupttitel" => "Trennzeichen für mehrere Autoren und Urheber")

- alleHerausgeber
  - Listet alle Herausgeber (inkl. MarkUp), getrennt durch das im ConfigFile für Urheber definierte Trennzeichen (Abschnitt "Einstellungen für den Haupttitel" => "Trennzeichen für mehrere Autoren und Urheber")
- alleUebersetzer
  - Listet alle Übersetzer (inkl. MarkUp), getrennt durch das im ConfigFile für Urheber definierte Trennzeichen (Abschnitt "Einstellungen für den Haupttitel" => "Trennzeichen für mehrere Autoren und Urheber")
- alleIllustratoren
  - Listet alle Illustratoren (inkl. MarkUp), getrennt durch das im ConfigFile für Urheber definierte Trennzeichen (Abschnitt "Einstellungen für den Haupttitel" => "Trennzeichen für mehrere Autoren und Urheber")
- inhaltsverzeichnis
  - Gibt das generierte Content-Inhaltsverzeichnis aus
  - Dieses enthält nur die eigentlichen Inhaltskapitel aus dem Hauptteil und keine Zusatzseiten wie Cover, Haupttitel, etc.
- inhaltsverzeichnis-toc
  - Gibt das komplette TOC-Inhaltsverzeichnis aus
  - Dieses enthält neben den Inhaltskapiteln aus dem Hauptteil auch Zusatzseiten wie Cover, Haupttitel, etc. und entspricht damit dem Reader-TOC.

> **Anmerkung**
>
> Falls ein Template keine Ausgabe erzeugt, z.B. wenn alle Herausgeber gelistet werden sollen, in der XML-Instanz aber kein

Element <meta_herausgeber> existiert, dann wird das Element – im Gegensatz zum oben genannten Verhalten bei Element-Abfragen – trotzdem generiert!

Beispiel 1.20. Abfrage aller Autoren

```
<div callTemplate="alleAutoren"/>
```

HTML-Ausgabe im EPUB:

```
<div>Siegfried <span class="unterstr">Lenz</span> | <span class="durchgestr">Tobias</span> Fischer | Vorname <strong>Nachname</strong></div>
```

Beispiel 1.21. @class Attribute am abfragenden Element werden übernommen

```
<div callTemplate="alleAutoren" class="t_autor"/>
```

HTML-Ausgabe im EPUB:

```
<div class="t_autor">Siegfried <span class="unterstr">Lenz</span> | <span class="durchgestr">Tobias</span> Fischer | Vorname <strong>Nachname</strong></div>
```