bx:tools

Das Tag `tools` vereint kleine Hilfen, um häufig auftretende Aufgaben zu vereinfachen.

Hilfsmittel für Requestparameter

tools.request

```
<bx:tools.request name="{param}" />
<bx:tools.request name="{param}" [value="{text}"] [not]> Inhalt </bx:tools.request>
```

Es wird ein Request-Parameter ausgegeben oder dieser ausgewertet.

Die erste Form gibt den Inhalt von `<param>` aus, falls vorhanden. In der zweiten Form wird der Tag-Inhalt nur ausgegeben, wenn der Request-Parameter existiert, oder, falls zusätzlich "value" angegeben ist, einen bestimmten Wert enthält. Die Bedingung ist mit not umkehrbar. Eine vergleichbare Funktion beinhaltet bx:pagedata.

tools.listrequest

```
<bx:tools.listrequest [include="{filter}" | exclude="{filter}"] />
```

Alle Parameter aus dem Request werden als Formularfelder vom Typ "hidden" in die Seite geschrieben. Falls nur bestimmte Request-Parameter ausgegeben (include) oder bestimmte Parameter ausgeschlossen (exclude) werden sollen, können diese als Komma-getrennte Liste angegeben werden (z.B.: include="artikelID,kundenID").

tools.querystring

```
<bx:tools.querystring [include="{filter}" | exclude="{filter}"] />
```

Diese Funktion wandelt alle Request-Parameter in einen QueryString um. Falls nur bestimmte Request-Parameter umgewandelt (include) oder bestimmte Parameter ausgeschlossen (exclude) werden sollen, können diese als Komma-getrennte Liste angegeben werden (z.B.:

exclude="counter,index").

Kodierung (Encodings)

tools.htmlencode

```
<bx:tools.htmlencode> Inhalt </bx:tools.htmlencode>
```

Der Tag-Inhalt wird HTML-kodiert. Wenn man nur `< > & "` encoden möchte, muß man `<bx:tools.xmlencode>` nehmen. Es werden folgende Ersetzungen vorgenommen:

Originalwert	ausgegebener Wert
<	<
>	>
"	"
&	&
'	'
äöüÄÖÜß§€, deutsche Anführungszeichen unten und oben und langen Bindestrich	
alles ab ASCII-Code 128	zu "&#[code];"

tools.urlencode

```
<bx:tools.urlencode [charset="{set}"]> Inhalt </bx:tools.urlencode>
```

Es wird eine URL-Kodierung des Tag-Inhalts vorgenommen. Mit dem Parameter `charset` kann eine andere Zeichenkodierung angegeben werden (Standard ist "iso-8859-1"). Es wird java.net.URLEncoder benutzt.

tools.urlpathencode

```
<bx:tools.urlpathencode [charset="{set}"]> Inhalt </bx:tools.urlencode>
```

Zum Kodieren von Pfadbestandteilen (z. B. Dateiname). Führt ein URL-Encoding des Tag-Bodys aus. Mit dem Parameter 'charset' kann ein anderes Encoding angegeben werden (Standard=iso-8859-1). Zum encoden wird `URLEncoder.encode(String, String)` verwendet.

tools.entityencode

```
<bx:tools.entityencode> Inhalt </bx:tools.entityencode>
```

Es werden alle Zeichen mit einem ASCII-Code > 127 (0x80) in die Form &#nnn; gebracht, wobei nnn für den ASCII-Code in dezimaler Schreibweise steht. *Ab v2.5.9 werden auch eventuelle Steuerzeichen (0x00-0x1F) entfernt.*

tools.xmlencode

```
<bx:tools.xmlencode> Inhalt </bx:tools.xmlencode>
```

Dieses Tag führt zunächst ein .htmlencode aus und lässt das Ergebnis nochmals durch .entityencode laufen. * *Ab v2.5.9 werden auch eventuelle Steuerzeichen (0x00-0x1F) entfernt.**

tools.removectrlchars

```
<![CDATA[<bx:tools.removectrlchars> Inhalt mit seltsamen Zeichen
</bx:tools.removectrlchars>]]>
```

Dieses Tag entfernt eventuelle Steuerzeichen (0x00-0x1F, außer Tab und Zeilenumbruch). Das Ergebnis kann dann innerhalb eines CDATA-Blocks verwendet werden (ab Version 2.5.9).

tools.html2plain

```
<bx:tools.html2plain> Inhalt </bx:tools.html2plain>
```

Der Inhalt wird zunächst durch `.htmlencode` umgewandelt, danach werden folgende Ersetzungen vorgenommen:

Originalwert	ausgegebener Wert		Originalwert	ausgegebener Wert
\r	*nichts*		\n	*Leerzeichen*
 	\r\n		 	\r\n

Originalwert	ausgegebener Wert		Originalwert	ausgegebener Wert
 	\r\n		<p>	\r\n\r\n
<sonstige Tags>	*nichts*			*Leerzeichen*
ü	ü		Ü	Ü
ä	ä		Ä	Ä
ö	ö		Ö	Ö
ß	ß		"	" (gerade Anführungsz. oben)
<	<		>	>
&	&		„	" (dopp. Anführungsz. unten)
“	" (dopp. Anführungsz.oben)		–	- (Breite n)
…	...		–	- (Breite n)
€	€		´	´
’	'		§	§

Zuletzt wird noch eventueller Whitespace am Anfang und Ende des Textes entfernt.

tools.plain2html

```
<bx:tools.plain2html> Inhalt </bx:tools.plain2html>
```

Wandelt den plain-Text im body in HTML-Code um, indem er HTML encoded wird und bei jeden Zeilenumbruch ein
oder
gesetzt wird

Wirkungsweise

`<bx:tools.plain2html>Erste Zeile Zweite Zeile Dritte Zeile Vierte Zeile Fünfte Zeile</bx:tools.plain2html>`

Wird zu

`«Erste Zeile<br> Zweite Zeile<br> Dritte Zeile<br> Vierte Zeile<br> F&uuml;nfte Zeile»`

tools.csvfield

```
<bx:tools.csvfield> Inhalt </bx:tools.csvfield>
```

Der Text wird mit " umgeben und allen " im Text wird ein zusätzliches " vorangestellt. (ab v2.5.9)

tools.scriptencode

```
<bx:tools.scriptencode> Inhalt </bx:tools.scriptencode>
```

Diese Funktion dient zum Encoden eines Textes, damit dieser direkt als String in JavaScript verwendet werden kann. (ab v2.5.9)

Originalwert	ausgegebener Wert
\n	\\n
\r	\\r
"	\"
'	\'
\t	\\t

Zusätzlich wird jeder Backslash (), der nicht zum Escapen verwendet wird (also **nicht** vor folgenden Zeichen steht: n r t " ') verdoppelt.

tools.scripteventencode

```
<bx:tools.scripteventencode > Inhalt </bx:tools.scripteventencode >
```

Hier wird ein String so encoded, dass er direkt als Event-Code für ein HTML-Element festgelegt werden kann. (ab v2.5.9)

Originalwert	ausgegebener Wert
\n	\\n
\r	\\r
&	&amp;
"	&quot;
'	\'

Originalwert	ausgegebener Wert
\t	\\t
\	\\

tools.umlautencode

```
<bx:tools.umlautencode > Inhalt </bx:tools.umlautencode >
```

Deutsche Umlaute werden hier in die Schreibweise mit normalen Buchstaben überführt. (ab v2.5.9)

Originalwert	ausgegebener Wert
ß	ss
Ä	Ae
ä	ae
Ö	Oe
ö	oe
Ü	Ue
ü	ue

tools.mysqlencode

```
<bx:tools.mysqlencode>Gasthaus 'Zur Tanne'</bx:tools.mysqlencode>
```

Escaped einige Zeichen, die für SQL-Injections verwendet werden könnten. (ab v2.5.9) Beispiel im Quellcode: wird in `Gasthaus \'Zur Tanne\'` umgewandelt

tools.mysqlencodeall

```
... WHERE KEYWORDS LIKE '%<bx:tools.mysqlencodeall><bx:pagedata.request
name="suche"></bx:tools.mysqlencodeall>%' ...
```

Escaped einige Zeichen, die für SQL-Injections verwendet werden könnten (auch % und _). (ab v2.5.9) Dies ist zur Verwendung in einem SQL-`LIKE`-Befehl gedacht (siehe Quellcode) Das %-

Zeichen ist im LIKE ein Platzhalter für beliebig viele Zeichen. Ohne diesen Befehl würde z. B. im LIKE-Ausdruck `WHERE Feld LIKE '100%'` alles gefunden werden, das mit "100" beginnt.

Sonstige Tools

tools.lastmodified

```
<bx:tools.lastmodified [pattern="{format}"] />
```

Gibt Datum/Zeit der letzten Änderung des aktuellen Menüpunktes aus (Meta-Wert "dataPublished" oder "contentPublished"). Das Format der Ausgabe lässt sich über `pattern` steuern (Standard ist "dd.MM.yyyy HH:mm").

tools.positiontracker

```
<bx:tools.positiontracker [boundary="{text}"] [showall] [startlevel="{n}"] />
```

Für Breadcrumb. Es wird ein Position-Tracker in die Seite geschrieben. Die übergeordneten Menüpunkte werden jeweils mit einer entsprechenden Anzahl von "../" verlinkt.

boundary

Text zwischen den einzelnen Ausgaben (Standard-Wert ist " >> ")

showall

auch inaktive Hauptmenüpunkte werden ausgegeben

startlevel

Angabe, wenn in einem anderen Level gestartet werden soll (Hauptmenü = Level 0)

Beispiele

```
<bx:tools.positiontracker boundary=" | " startlevel="1"/>
```

Ausgabe der Menüpunkte ab der 2. Ebene, Trenner ist ein Pipe-Zeichen zwischen zwei Leerzeiche.

tools.datum

```
<bx:tools.datum [show="{month-begin | month-end}" |
                [year="{mod}"] [month|month1="{mod}"] [day="{mod}"] [hour="{mod}"]
[minute="{mod}"] [second="{mod}"]]
                [locale="{lang}"] [pattern="{format}"] />
```

Dieses Tag kann benutzt werden, um das aktuelle oder ein bestimmtes Datum auszugeben. Das Datum kann auch manipuliert werden (siehe bei bx:recordfield)

> **Achtung:** Monate laufen von 0-11.

show

month-begin: 1. des aktuellen Monats (0.00.00)
month-end: letzter des aktuellen Monats (23.59.59)
ohne Angabe von show: jetzt

year, month, day, hour, minute, second

mit diesen Parametern kann ein Datum manipuliert werden.
n: das entsprechende Feld wird auf den Wert **n** gesetzt
+n: das entsprechende Feld wird um den Wert **n** erhöht
-n: das entsprechende Feld wird um den Wert **n** verringert

locale

Sprachformatierung (z.B. "de" oder "en_US")
`locale` sollte immer angegeben werden, da sonst die Standardeinstellung vom Server genutzt wird, die sich aber nach Updates oder Umzügen ändern kann.

month1

wie bei month, aber beginnt nicht bei 0 sondern bei 1, d.h. bei month1="1" wird Januar und nicht Februar ausgegeben

pattern

legt die Formatierung des Datums fest (Standard ist "dd.MM.yyyy")

Beispiele

```
<bx:tools.datum/>□□□□□□□□□□<-- "heute", z.B. 08.06.2014 -->
<bx:tools.datum pattern="d. MMM yy HH:mm:ss"/>□□<-- "jetzt" wird ausgegeben, z.B. 3. Feb 12
13:42:55 -->
<bx:tools.datum show="month-end" pattern="dd"/>□□<-- Tag des Monatsende vom aktuellen Monat,
z.B. beim Aufruf im Februar: 28 -->
<bx:tools.datum lang="en_US"/>□□□□□□<-- Ausgabe von "heute" im US-Format, z.B. 2001-07-04 -->
```

```
<bx:tools.datum month="+1" day="1"/> □□□□□□□<-- nächster Monats-Erster -->
<bx:tools.datum month=+1 day=1 hour=-24"/>□□□□□□<-- Monatsletzter   -->
<bx:tools.datum day="+7"/>□□□□□□□□□□<-- in einer Woche -->
<bx:tools.datum day="-1" hour="12" minute="0" second="0"/>□□<-- gestern Mittag -->
<bx:tools.datum year="-1" month="11" day="31"/>□□□□□<-- Silvester letztes Jahr -->
```

tools.timeframe

```
<bx:tools.timeframe after="{zeit}" before="{zeit}" week="{bits}" [not] [now={...}]> Inhalt
</bx:tools.timeframe>
```

Zeigt den Inhalt an, wenn die aktuelle Serverzeit in einem bestimmten Rahmen liegt. Wenn before kleiner als after ist, wird der Uhrzeitbereich außerhalb der Angaben verwendet (ab v2.6.8. - vorher wäre es nie ausgeführt worden, weil keine Uhrzeit vor 7 und nach 20 Uhr sein kann)

after

wird im 24h-Format angegeben (HH:mm)

before

wird im 24h-Format angegeben (HH:mm)

week

Angabe als eine sogenannte Bitmaske, d.h. jede Stelle steht für einen Tag.
So steht z.B. "1111100" für *nur werktags* oder "1010100" für *montags, mittwochs, freitags*.

not

zur Umkehrung der Uhrzeitangaben. Es beeinflußt nur die Uhrzeit und nicht den Parameter `week`

now

Mit dem Parameter `now` kann man eine andere Uhrzeit testen, ohne bis zu dieser Uhrzeit warten zu müssen. (siehe Beispiel 3)

Beispiele

Beispiel 1

```
<bx:tools.timeframe after="06:00" before="18:00" week="1000000">Wir haben
geöffnet.</bx:tools.timeframe>
```

Inhalt wird nur Montags von 6.00 bis 18.00 Uhr angezeigt.

Beispiel 2

```
<bx:tools.timeframe before="6:00" after="18:00">Wir haben geschlossen.</bx:tools.timeframe>
```

entspricht: `<bx:tools.timeframe not before="18:00" after="6:00">`

Beispiel 3

```
<bx:tools.timeframe after="22:00" now="7.3.2018 22:01">Bettruhe!!!</bx:tools.timeframe>
```

tools.for / tools.forchar

```
<bx:tools.for start="{n}" end="{n}" [inkrement="{n}"]> Inhalt </bx:tools.for><!-- für Zahlen
-->
<bx:tools.forchar start="{zeichen}" end="{n}"> Inhalt </bx:tools.forchar><!-- für Buchstaben
-->
```

Dieses Tag führt den Inhalt einer Schleife mehrfach aus und ersetzt jedesmal einen Platzhalter durch eine laufende Nummer. Im Inhalt des Tags können dann "{i}" und "{ii}" verwendet werden (ii = vorangestellte 0, wenn kleiner als 10)

start

Startet bei angegebener Zahl bzw. angegebenem Zeichen (z.B. `a`)

end

Endet bei angegebener Zahl bzw. angegebenem Zeichen (z.B. `z`)

inkrement

der Wert, um den erhöht werden soll (Standard: 1).
Falls `start` > `end` ist, muss hier eine negative Zahl eingetragen werden!

Beispiele

```
<bx:clipboard.cut name="anzTage"><bx:tools.datum show="month-end"
pattern="dd"/></bx:clipboard.cut>
<select>
  <bx:tools.for start="1" end="clipboard:anzTage">
    <option value="{ii}">{i}</option>
  </bx:tools.for>
</select>
```

Die Tage des aktuellen Monats (Anzahl wurde vorher mit `bx:tools.datum` ermittelt und in einem Clipboard zwischengespeichert) werden in einem Drop-Down ausgegeben (value mit vorangestellter 0, wenn unter 10)

```
Ergebnisse finden beginnend mit:<br>
|
<bx:tools.forchar start="A" end="Z">
  <a href="suche.act?init={i}">{i}</a>|
</bx:tools.forchar>
```

Es wurd eine verlinkte Buchstabenleiste mit den Großbuchstaben von "A" bis "Z" erzeugt.

tools.for-split

```
<bx:tools.for-split source="[Daten]" regex="[Trenner]" clipboard="[key]" attribute="[key]"
placeholder="[Platzhalter]"> Auswertung mit [Platzhalter] </bx:tools.for-split>
```

Dieses Tag führt den Inhalt einer Schleife mehrfach aus und ersetzt jedesmal einen Platzhalter durch eine laufende Nummer. Im Inhalt des Tags können dann "{i}" und "{ii}" verwendet werden (ii = vorangestellte 0, wenn kleiner als 10)

source

Quelle

regex

Angabe des Trenners, z.B. Komma, Leerzeichen oder Semikolon

clipboard

Wert kann wird im Clipboard gespeichert

attribute

Wert kann wird im Request-Attribute gespeichert

placeholder

Es kann ein anderer Platzhalter definiert werden, ansonsten lautet der Platzhalter {i}

Beispiele

```
<bx:tools.for-split source="Hallo Welt" regex=" "> :{i}: </bx:tools.for-split>
```

ergibt " :Hallo: :Welt: "

```
<bx:clipboard.cut name="idliste">AAA,BBB,CCC</bx:clipboard.cut>
<bx:tools.for-split source="clipboard:idliste" regex="," clipboard="einzelneID">
<bx:record id="clipboard:einzelneID">
 ... <bx:recordfield...> ...
</bx:record>
</bx:tools.for-split>
```

tools.random

```
<bx:tools.random/>□□□□□□□□□□□□□<-- zufälliger double-Wert zwischen 0 und 1, ca. 15 Nachkommastel
<bx:tools.random max="{zahl}"/>□□□□□□□□□□<-- zufälliger int-Wert zwischen 0 und {zahl} -->
<bx:tools.random chance="{zahl}"> [Inhalt] </bx:tools.random>□<-- zeigt den Inhalt mit der
Wahrscheinlichkeit von {zahl} Prozent an (1≤{zahl}≤99) -->
```

Dieser Befehl zeigt eine Zufallszahl an bzw. zeigt einen Inhalt mit einer bestimmten Wahrscheinlichkeit an.

tools.randstring

```
<bx:tools.randstring [length={len}] [lower] [upper] [num] [alphanum] [extra] [extra2] [all] />
```

Dieser Befehl erzeugt einen zufälligen String aus bestimmten Charakter-Klassen. Dies ist z.B. für Passwortgenerierung nützlich.

> Es muss mindestens eine Charakter-Klasse angegeben werden.

length

Länge des zu erzeugenden Strings (Standard 12 Zeichen)

lower

Kleinbuchstaben a-z

upper

Großbuchstaben A-Z

num

Ziffern 0-9

alphanum

lower + upper + num

extra

! ? - _ ( ) + * \ / | $ ~ # @

extra2

< > { } [ ] " § % & ' ^ ° ; . : =

all

alphanum + extra + extra2

Beispiele

```
<bx:tools.randstring length=10 alphanum/>
```

Es wird ein Paßwort mit 10 Zeichen erzeugt, das Groß- und Kleinbuchstaben sowie Zahl enthält.

tools.uuid

```
<bx:tools.uuid />
```

Hiermit wird eine zufällige UUID (Typ 4) erzeugt (z.B. f3910a2f-63cf-433b-8117-1ac0d22bb4f5).

tools.id

```
<bx:tools.id />
```

Dieses Tag erzeugt eine neue Batix-ID und gibt diese aus (z.B. 12E241E47CB).

Fehlersuche (Debugging)

tools.showrequest

```
<bx:tools.showrequest/>
```

Erzeugt eine Debug-Ausgabe des Request-Typs sowie sämtlicher Parameter, die im Request enthalten sind.

tools.duration

```
<bx:tools.duration save="{name}"/>
<bx:tools.duration show="{name"/>
```

Erzeugt eine Debug-Ausgabe der Bearbeitungszeit einer im Aufbau befindlichen Seite.

save

Anfang der Zeitmarke wird gesetzt Durch die Benennung sind mehrere Zeitmarken möglich

show

verstrichene Zeit seit `save` wird angezeigt (in Sekunden mit 3 Nachkommastellen) Weitere `show`-Tags liefern fortlaufende Messungen.

Beispiele

```
<html>
  <head></head>
  <body>
    ... Quelltext ...
    <bx:tools.duration save="dauer1"/>
    ... komplizierte SQL ...
    Verarbeitungsdauer SQL1: <bx:tools.duration show="dauer1"/>
    <bx:tools.duration save="dauer2"/>
    ... zu vergleichende SQL ...
    Verarbeitungsdauer SQL2: <bx:tools.duration show="dauer2"/>
    ... Quelltext ...
  </body>
</html>
```

Für beide SQL-Aufrufe wird die Ausführungszeit gemessen und ausgegeben - ein direkter Vergleich ist möglich.

tools.log

```
<bx:tools.log level="{stufe}" [copy]>zu loggender Text</bx:tools.log>
```

Schreibt den Inhalt in die Logdatei, anstatt in die Seite.

level

Logstufe `error`, `warn`, `info` oder `debug` Standard: `info`

copy

Text soll auch auf der Seite angezeigt werden

Beispiele

```
<bx:tools.duration save="dauer"/>
... aufwändige Filterungen ...
<bx:tools.log>Verarbeitungsdauer: <bx:tools.duration show="dauer"/></bx:tools.log>
```

Die Zeit, die die "aufwändige Filterung" braucht, wird in die Log-Datei geschrieben

Zeichenketten-Funktionen

tools.replacetext

Ersetzen-Modus

```
<bx:tools.replacetext regex="<ausdruck>" replacement="<text>"> Inhalt </bx:tools.replacetext>
```

Es wird nach bestimmten Ausdrücken im Tag-Inhalt gesucht und diese werden durch den gewünschten Text ersetzt. Es wird die Java-Methode String.replaceAll verwendet, dadurch ergeben sich vielfältige Möglichkeiten.

regex

regulärer Ausdruck, nach dem im Text gesucht werden soll

replacement

Ersetzungs-Text

Beispiele

```
<bx:tools.replacetext regex="World" replacement="Batix">Hello, World!</bx:tools.replacetext>
```

Hier wird aus "Hello, World!" der Text "Hello, Batix!"

```
<bx:tools.replacetext regex="(.+?),(.+?),(.+?)" replacement="Wert 2=$2; Wert 1=$1; Wert 3=$3">
  Alice,123,ABC
</bx:tools.replacetext>
```

Umwandeln von CSV-Daten: Das Ergebnis wäre hier der Text "Wert 2=123; Wert 1=Alice; Wert 3=ABC".

```
<bx:clipboard.cut name="temp"><a href="$1">$1<a></bx:clipboard.cut>
<bx:tools.replacetext regex="(http://[^\ ]+)" replacement="clipboard:temp">
  Einen Text mit http://einer.url/zu/einem Text mit einem Link machen.
</bx:tools.replacetext>
```

Um in einen Text mit einer URL einen Link um die URL einzufügen: Das bx:clipboard wird benötigt, weil im replacement-Parameter die Zeichen ", < und > nicht verwendet werden können. *ab v2.5.9*

> Allgemeine Hilfe und Beispiele zu regulären Ausdrücken gibt es u.a. auf http://www.regular-expressions.info (englisch).

Schleifen-Modus

```
<bx:tools.replacetext regex="<ausdruck>" source="<quelle>">
  Inhalt
</bx:tools.replacetext>
```

Durch die Angabe von `source` schaltet das Tag in den Schleifen-Modus. Hierbei wird der Tag-Inhalt für jeden Match ausgeführt. Ausgaben, die im Tag-Inhalt erfolgen werden ignoriert (stattdessen kann ein Clipboard geschrieben werden, um den Match zu modifizieren). Es werden Informationen (Match-Index, Match-Gruppen) über Clipboards bereitgestellt, der gematchte Text kann so auch ersetzt / verändert werden.

Folgende Clipboards werden pro Durchlauf gesetzt:

Clipboard-Name	Beschreibung
`replacetext_match`	Der komplette gematchte String. In dieses Clipboard kann im Tag-Inhalt auch geschrieben werden, die gematchte Textstelle wird nach jedem Durchlauf mit dem Inhalt dieses Clipboards ersetzt.
`replacetext_index`	Laufende Nummerierung der Matches, beginnend bei Null. Im ersten Durchlauf steht also "0" im Clipboard, danach "1", usw..
`replacetext_group_<n>`	`<n>` ist eine Zahl. Für jede Gruppe des Regexes (mit runden Klammern umschlossen) werden die gematchten Textstellen pro Durchlauf in nummerierte Clipboards gepackt, wobei - wie bei Regex üblich - replacetext_group_0 noch mal den kompletten Match enthält, replacetext_group_1 die erste Gruppe, replacetext_group_2 die zweite usw..

Man kann sich den Tag-Inhalt wie eine Funktion vorstellen, die pro Match aufgerufen wird: Parameter werden über die Clipboards übergeben, der Rückgabewert läuft auch über ein Clipboard (replacetext_match).

Beispiele

Batix Quelltext:

```
<bx:clipboard.cut name="zutaten" trim>
300#g#Mehl
5##Eier
150#ml#Milch
</bx:clipboard.cut>
<bx:tools.replacetext regex="(\d+)#([^#]+)?#(.*)" source="clipboard:zutaten">
  1 = Anzahl
  2 = Einheit
  3 = Zutat
  <bx:clipboard.cut name="replacetext_match">
    Zutat <bx:clipboard.paste name="replacetext_index" />:
    <bx:clipboard.paste name="replacetext_group_3" />
    (<bx:clipboard.paste name="replacetext_group_1" />
    <bx:clipboard.paste name="replacetext_group_2" />)
    <br>
  </bx:clipboard.cut>
</bx:tools.replacetext>
```

Ausgabe:

```
Zutat 0: Mehl (300 g)
Zutat 1: Eier (5 )
Zutat 2: Milch (150 ml)
```

Erkärung:

- Der zu durchsuchende Text wird in das Clipboard "zutaten" geschrieben, welches als `source` an `replacetext` übergeben wird.
- Der Regex lautet: `(\d+)#([^#]+)?#(.*)`
  1. Gruppe: `\d+` (nur Ziffern)
  2. `[^#]+` (alles außer #)
  3. `.*` (egal was)
- Beispielhaft die Clipboards im ersten Durchlauf:

replacetext_match	300#g#Mehl
replacetext_index	0
replacetext_group_0	300#g#Mehl
replacetext_group_1	300
replacetext_group_2	g
replacetext_group_3	Mehl

- Nach dem Durchlauf wird in das Clipboard "replacetext_match" zurückgeschrieben, was die Zeile im Original ersetzt.
- Hinweis: Da die zweite Gruppe mit einem Fragezeichen quantifiziert ist, ist sie optional. Im zweiten Durchlauf (die 5 Eier) ist das Clipboard "replacetext_group_2" also leer.

tools.uppercase

```
<bx:tools.uppercase> bring mich gross raus </bx:tools.uppercase>
```

Wandelt alle Buchstaben im Body in Großbuchstaben um und gibt diese aus.

tools.lowercase

```
<bx:tools.lowercase> jetzt werd ich ganz klein </bx:tools.lowercase>
```

Wandelt alle Buchstaben im Body in Kleinbuchstaben um und gibt diese aus.

tools.substring

```
<bx:tools.substring maxlength="{Zahl}" [startindex="{Zahl}"] [fillbefore="{ein Zeichen}" |
fillafter="{ein Zeichen}"]>
```

Schneidet aus dem übergebenen Tagbody den durch Indexpositionen angegebenen Bereich heraus und gibt nur diesen zurück.

maxlength

muß mindestens 1 sein

startindex

Stelle, ab wann der Substring herausgelöst werden soll (startet mit 0) Standard: 0 (bei Fehlen des Parameters)

fillbefore, fillafter

optionales Füllzeichen (keine Entity) wenn Text kürzer ist als bei `maxlength` angegeben, wird er mit dem angegeben Zeichen aufgefüllt

Beispiele

```
<bx:tools.substring maxlength="5" startindex="6">Hello World</bx:tools.substring>
```

Gibt "World" aus

```
<bx:tools.substring maxlength="5" fillbefore="0">7318</bx:tools.substring>
```

gibt "07318" aus (wenn z. B. PLZ als Zahl gespeichert ist und demzufolge nur 4 Ziffern hat)

bx:tools.trim

```
<bx:tools.trim [lines] [spaces] [single-line]> Inhalt </bx:tools.trim>
```

Entfernt alle Whitespace-Zeichen (Zeilenumbrüche, Leerzeichen, Tabulator) am Anfang und Ende des inneren Bereiches

lines

reduziert mehrfache Zeilen zu einem Einzelnen (siehe Beispiel)

spaces

reduziert mehrfache Leerzeichen zu einem einzelnen

single-line

entfernt alle Zeilenumbrüche und schreibt alles leerzeichengetrennt in eine Zeile

Beispiele

```
<bx:tools.trim lines>   Hallo


          Welt


                        </bx:tools.trim>
```

reduziert mehrfache Zeilenumbrüche zu einem Einzelnen würde also nur „Hallo“ und darunter „Welt“ schreiben

Links erzeugen

tools.absoluteurl

```
<bx:tools.absoluteurl> Pfadangabe </bx:tools.absoluteurl>
```

Erzeugt aus dem relativen URL-Pfad im Tag-Body eine absolute URL und schreibt sie anstelle des Tags in die Seite.

Beispiele

```
<bx:tools.absoluteurl>../test.htm</bx:tools.absoluteurl>
```

Im Beispiel würde das Tag in der Seite durch eine URL "http[s]://{hostname}/www/{pfad}/test.htm" ersetzt werden, die auf die Seite test.htm im übergeordneten Menüpunkt verweist.

tools.rooturl

```
<bx:tools.rooturl> Pfadangabe </bx:tools.rooturl>
```

Erzeugt aus dem relativen URL-Pfad im Tag-Body eine absolute URL und schreibt einen absoluten Pfad beginnend mit "/" anstelle des Tags in die Seite.

Beispiele

```
<bx:tools.rooturl> ../test.htm</bx:tools.rooturl>
```

Im diesem Beispiel würde das Tag in der Seite durch eine URL "/www/{pfad}/test.htm" ersetzt werden, die auf die Seite test.htm im übergeordneten Menüpunkt verweist.