bx:recordfield

Das Tag `recordfield` stellt ein einzelnes Element eines Container-Datensatzes dar. Dies kann ein simpler Typ wie z.B. Text, Datum oder Bild, aber auch eine Schleife über Untercontainer sein.

Allgemeine Syntax

```
<bx:recordfield.{field} [parent | parent="{n}"] [baseloop="{schleifenname}"] [level=".."]
zusätzliche Parameter />
<bx:recordfield.{field} [parent | parent="{n}"] [baseloop="{schleifenname}"] [level=".."]
zusätzliche Parameter [dummy]> Inhalt </bx:recordfield.{field}>
```

Je nach Feld-Typ stellt sich bx:recordfield unterschiedlich dar und akzeptiert verschiedene Parameter.

field	Name des Feldes im Container
level	level=".." - übergeordneter Container Zugriff auf die Felder des übergeordneten Containers (zur Zeit nur direkt-übergeordnet möglich) sollte nicht mehr verwendet werden, lieber `parent`
baseloop	Name der Containerschleife (wie er nach dem Punkt im Tag steht) Bezug auf eine bestimmte Containerschleife (z.B. bei Verschachtelungen)
parent	parent: Zugriff auf direkt übergeordneten Container parent="2": Zugriff auf n-ten übergeordneten Container
dummy	bewirkt, daß, wenn noch kein DS verknüpft ist, ein neuer DS erzeugt wird Dazu darf beim Speicher-Action weder ja noch nein bei createnew gewählt sein.

Beispiele

```
<bx:containerfilter.Kategorie pool="Kategorien" orderby="Titel">
  <h2><bx:recordfield.Titel/></h2>
  <bx:containerfilter.Artikel pool="Artikel" orderby="Titel">
    <a href="detail.htm?artikel=<bx:recorddata.id/>&amp;kat=<bx:recorddata.id
baseloop="Kategorie"/>"><bx:recordfield.Titel/></a>
```

```
  </bx:containerfilter.Artikel>
  <br>
</bx:containerfilter.Artikel>
```

Linkaufruf mit Übergabe der Kategorie in der inneren Schleife. Zugriff auf die Kategorie-ID mit `baseloop`.

```
<bx:containerfilter.Anbieter pool="Anbieter" orderby="Titel">
  <h2><bx:recordfield.Titel/></h2>
  <bx:containerfilter.Kategorie pool="Kategorien" orderby="Titel">
    <h2><bx:recordfield.Titel/></h2>
    <bx:containerfilter.Artikel pool="Artikel" orderby="Titel">
      <a href="detail.htm?artikel=<bx:recorddata.id/>&amp;kat=<bx:recorddata.id
parent="1"/>&amp;anbieter=<bx:recorddata.id parent="2"/>"><bx:recordfield.Titel/></a>
    </bx:containerfilter.Artikel>
    <br>
  </bx:containerfilter.Artikel>
</bx:containerfilter.Anbieter>
```

Ähnliches Beispiel wie oben mit zusätzlicher Ebene. Zugriff auf die Kategorie-ID und die Anbeiter-ID mit `parent`.

Filter

```
<bx:recordfield.{field} [filter="{name}" | dynamicfilter="{feldname}"] [quiet] .../>
```

Bei einfachen Typen (Text, Datum, Zahl, Wahrheitswert) lassen sich auch JSP-Filter einsetzen. Die Schnittstelle zum Filter bildet das Request-Attribut "attrValue". Über dieses wird dem Filter der Inhalt des Tags übergeben und aus diesem wird nach dem Filteraufruf auch der neue Tag-Inhalt gelesen.

filter	Angabe eines Filters (direkt über JSP-Baustein)
dynamicfilter	Angabe eines Feldes im Datensatz, wo der Name des Filters drin steht (dynamisch)
quiet	Fehler werden nicht auf der Seite ausgegeben

Beispiele für Filter

Filtername im Feld

```
...
Preis ohne MWSt.: <bx:recordfield.Preis_netto/>
Preis incl. MWSt: <bx:recordfield.Preis_netto filter="Bruttopreis"/>
...
```

Innerhalb des Filters "Bruttopreis" wird der (Netto)Preis in Bruttopreis umgerechnet und anstelle des Wertes im Recordfield ausgegeben.

Filtername aus Container

```
...
Preis ohne MWSt.: <bx:recordfield.Preis_netto/>
Preis incl. MWSt: <bx:recordfield.Preis_netto dynamicfilter="Filter"/>
...
```

Gleiches Beispiel wie oben, nur daß der Filtername (also Bruttopreis) im Feld "Filter" steht und von dort genommen wird. Vorteil: es kann pro DS ein völlig anderer Filter bestimmt werden.

einzeiliger Text

```
<bx:recordfield.{field} [encode|type="(quoted-printable | javascript | url | plain |
csvfield)"] [maxwords="{n}"] [maxchars="{n}" [ellipsis]]  />
```

Je nach Feld-Typ stellt sich bx:recordfield unterschiedlich dar und akzeptiert verschiedene Parameter.

encode | type	**Der Text wird (falls gewünscht umgewandelt) ausgegeben.** Standard: htmlencode quoted-printable:** das Format in einer Email, siehe entsprechenden Wikipedia-Eintrag javascript: Den Zeichen "\n \r \t " ' " wird ein Backslash vorangestellt. url: zum Kodieren von Request-Parametern. Es wird java.net.URLEncoder benutzt. urlpath: zum Kodieren von Pfadbestandteilen (z. B. Dateiname) plain: unformatierter Text csvfield: (ab v2.5.9) der Text wird mit " umgeben und allen " im Text wird ein zusätzliches " vorangestellt**
maxwords	nur eine bestimmte Anzahl an Wörtern ausgeben
maxchars	nur bestimmte Anzahl Buchstaben ausgeben

encode | type	**Der Text wird (falls gewünscht umgewandelt) ausgegeben.** Standard: htmlencode quoted-printable:** das Format in einer Email, siehe entsprechenden Wikipedia-Eintrag javascript: Den Zeichen "\n \r \t " ' " wird ein Backslash vorangestellt. url: zum Kodieren von Request-Parametern. Es wird java.net.URLEncoder benutzt. urlpath: zum Kodieren von Pfadbestandteilen (z. B. Dateiname) plain: unformatierter Text csvfield: (ab v2.5.9) der Text wird mit " umgeben und allen " im Text wird ein zusätzliches " vorangestellt**
ellipsis	ohne Parameter: fügt "..." an den text an, wenn dieser abgeschnitten wurde. ellipsis=" usw." (mit Parameterwert) fügt usw. an den Text an oder halt irgendwas anderes sinnvolles.

Feldinhalt vergleichen

```
<bx:recordfield.{field} matches="{regex}" [not]> Inhalt </bx:recordfield.{field}>
<bx:recordfield.{field} equals="{Vergleichs-Feldinhalt}" [not]> Inhalt
</bx:recordfield.{field}>
<bx:recordfield.{field} contains="{Suchbegriff im Feld}" [not]> Inhalt
</bx:recordfield.{field}>
```

Der Inhalt des Tags wird nur dann angezeigt, wenn der Inhalt des Container-Feldes mit den Testkriterien übereinstimmt (umkehrbar mit `not`).

matches	**Regulärer Ausdruck auf "leer" prüfen:** `matches="^$"(Achtung: es wird immer der gesamte String verglichen, "^" vor und "$" nach dem Ausdruck sind also implizit!)`
equals	String, der mit dem Feldinhalt übereinstimmen muß
contains	String, der im Feldinhalt enthalten sein muß

Beispiele

```
<bx:recordfield.Email matches="^[_a-zA-Z0-9-](\.{0,1}[_a-zA-Z0-9-])*@([_a-zA-Z0-9-
]{2,}\.){0,}[_a-zA-Z0-9-]{3,}(\.[_a-zA-Z]{2,4}){1,2}$" not> keine gültige E-Mail-Adresse
</bx:recordfield.Email>
```

Achtung: `matches` funktioniert nicht oder nicht korrekt wenn RegEx-Sonderzeichen wie Klammern, „+“ „?“, „*“ im Request stehen Mit equals funktioniert es dann:

```
<selected>
  <option value="bx:recorddata.id/>"<bx:recordfield.Kategoriename equals="request:param">
selected</bx:recordfield.Kategoriename>><bx:recordfield.Kategoriename/></option>
</selected>
```

Innerhalb einer Schleife: wenn übergebener Wert des Parameters `param` mit dem Kategorienamen übereinstimmt, wird dieser DS vorausgewählt.

```
<bx:recordfield.Email contains="batix.com">class="intern"</bx:recordfield.Email>
```

Jedem Link, der den String "batix.com" enthält, wird die Klasse "intern" zugewiesen (um ihn z.B. auf www.batix.com von externen Links zu unterscheiden)

mehrzeiliger Text

```
<bx:recordfield.{field} [encode="(quoted-printable | javascript | csv | plain | csvfield)"]
[maxwords="{n}"] [maxchars="{n}" [ellipsis]] />
```

csv	
encode | type	Der Text wird (falls gewünscht umgewandelt) ausgegeben.** Standard: htmlencode quoted-printable:** das Format in einer Email, siehe entsprechenden Wikipedia-Eintrag **javascript:** Den Zeichen "\n \r \t " ' " wird ein Backslash vorangestellt. **csv:** alle Zeilenumbrüche durch Leerzeichen ersetzt und sonstige Steuerzeichen entfernt. **plain:** unformatierter Text **csvfield:** (ab v2.5.9) der Text wird mit " umgeben und allen " im Text wird ein zusätzliches " vorangestellt url: gibt es für dieses Tag nicht, man kann bx:tools.urlencode drumrum machen **htmltext:** zeigt plain-Feld in Verwaltung aber macht s im Frontend rein
maxwords	nur eine bestimmte Anzahl an Wörtern ausgeben
maxchars	nur bestimmte Anzahl Buchstaben ausgeben
ellipsis	ohne Parameter: fügt "..." an den text an, wenn dieser abgeschnitten wurde. ellipsis=" usw." (mit Parameterwert) fügt usw. an den Text an oder halt irgendwas anderes sinnvolles.

Beispiele

```
<bx:recordfield.Titel/>
<bx:recordfield.Text maxwords="20" ellipsis />
<a href="detail.htm?beitragsid=<bx:recorddata.id/>">mehr</a>
```

Anteasern in einer Liste: es sollen max. 20 Wörter ausgegeben werden, dann wird ein ... gesetzt.

Datum/Zeit

```
<bx:recordfield.{field} [locale="{lang}"] [pattern="{format}"] [year="{mod}"] [month="{mod}"]
[day="{mod}"] [hour="{mod}"] [minute="{mod}"] [second="{mod}"] [type="holiday"]/>
```

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.**
pattern	Formatierung des Datums (Standard für Datum / Datum+Zeit / Zeit ist "EEEEEE, d. MMMMMM yyyy" / "dd.MM.yyyy HH:mm" / "HH:mm")
year month day minute second	Datum direkt setzen oder das aktuelle Datum manipulieren folgende Angaben bei `{mod}`: **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 **Achtung:** Monate laufen von 0-11
type="holiday"	prüft, ob der Tag auf einen Feiertag fällt und schreibt statt des Datums die Feiertagsbezeichnung hin (z. B. Silvester)(ab V2.6.0)(Thüringer Feiertage)

Beispiele

```
<bx:recordfield.Datumsfeld day="15" month="+1" />
```

zeigt anstatt des gespeicherten Datums den 15. des folgenden Monats an.

Abfrage

```
<bx:recordfield.{field} if="( past | future | empty | today)" [not] > Inhalt
</bx:recordfield.{field}>
```

Mittels dieser Variante wird *Inhalt* nur ausgegeben wenn das Datum im Datensatz (inklusive eventueller Modifikationen) in der Vergangenheit liegt (`past`), in der Zukunft liegt (`future`) oder leer ist (`empty`). `not` kehrt die Bedingung um.

Beispiele

```
<bx:clipboard.cut name="aktJahr"><bx:tools.datum pattern="yyyy"/></bx:clipboard.cut>
<bx:recordfield.Geburtsdatum year="clipboard:aktJahr" if="today"> Herzlichen Glückwunsch zum Geburtstag! </bx:recordfield.Geburtsdatum >
```

Das Feld Geburtsdatum wird mit dem heutigen Datum verglichen. Da das Jahr ja nicht stimmt, wird es auf das aktuelle Jahr gesetzt und kann nun verglichen werden.

Datumsmanipulation

```
<bx:recordfield.{field} weekday="{angabe}"/>
```

mo", "di", "mi", "do", "fr", "sa", "so" oder "mon", "tue", "wed", "thu", "fri", "sat", "sun" wechselt auf den entsprechenden Wochentag, der Woche in dem das Calendar-Objekt ist. vorangestelltes Plus- oder Minuszeichen wechslet zum folgenden bzw. vorigen entsprechenden Wochentag. Wenn calendar bereits auf dem gleichen Wochentag steht, wird nichts geändert. Dieser Fall wird mit doppelten plus/minus anders behandelt. Dann wird nämlich +/-7 Tage gerechnet.

Vom gespeicherten Wert aus können folgende Werte ermittelt werden:

weekday	**Do = der Donnerstag dieser Woche (egal ob schon vergangen oder nicht) +Do = der nächste Donnerstag (heute, falls heute Donnerstag) -Do = der vorige Donnerstag (heute, falls heute Donnerstag) ++Do bzw --Do = der nächste/vorige Donnerstag (nie heute)**

Zahl/Preis

Formatierte Ausgabe

```
<bx:recordfield.{field} [locale="{lang}"] [pattern="{format}"] [gs="{c}"] [ds="{c}"] />
<bx:recordfield.{field} isempty [not]> Inhalt </bx:recordfield.{field}>
```

locale	**Sprachformatierung (z.B. "de" oder "en_US")**

pattern	Formatierung festlegen, (Standard bei Zahl: "0.#####", bei Preis: "0.00", bei Preis mit Tausenderpunkt: "#,##0.00")
gs	Tausender-Trennzeichen festlegen
ds	Dezimal-Trennzeichen festlegen
isempty	Abfrage, ob das Feld leer ist

Abfrage des Betrages

```
<bx:recordfield.{field} equals="{n}" [not]> Inhalt </bx:recordfield.{field}>
<bx:recordfield.{field} equals="{n}" [true="{text}"] [false="{text}"] />
```

(Ausgabe von Quelltext abhängig vom Zahlenwert)

equals	Vergleich des Feldinhalts mit einem Wert. Ausgabe bei Übereinstimmung
true, false	eingetragener Text wird bei Übereinstimmung bzw. Nicht-Übereinstimmung ausgegeben.

```
<bx:recordfield.{field} {gt | gte | lt | lte}="{n}"> Inhalt </bx:recordfield.{field}>
```

Zahl mit einem Wert vergleichen. Ausgabe bei Übereinstimmung. Ab *Version 2.5.7* kann anstatt der festen Vergleichszahl für `{n}` auch "`request:{Parametername}`", "`attribute:{Attributname}`" oder "`clipboard:{Name}`" verwendet werden.

gt **gte** **lt** **lte**	**größer größer gleich kleiner kleiner gleich**

Alternativwert

```
<bx:recordfield.Preis alt="0" pattern="0.00"/>
<bx:recordfield.Anzahl alt="-1" lte="0">Anzahl fehlt oder ist 0</bx:recordfield.Anzahl>
```

Ab *Version 2.5.8* kann der Parameter `alt` verwendet werden um einen Alternativwert zu verwenden, falls keine Zahl gespeichert ist (auch keine 0).

```
<bx:recordfield.Preis alttext="gratis" pattern="0.00"/>
```

Ab *Version 2.5.8* kann der Parameter `alttext` verwendet werden um einen Alternativtext anzuzeigen, falls keine Zahl gespeichert ist (auch keine 0).

Dies kann z. B. auch für Intranet-Eingabeseiten verwendet werden, um zu markieren wo Zahlenwerte leer sind.

Wahrheitswert

```
<bx:recordfield.{field}> Inhalt </bx:recordfield.{field}>□□□□<!-- Ausgabe, wenn Feld im Contain
angehakt ist -->
<bx:recordfield.{field} not> Inhalt </bx:recordfield.{field}>□□□<!-- Ausgabe, wenn Feld im
Container NICHT angehakt ist -->
<bx:recordfield.{field} empty> Inhalt </bx:recordfield.{field}>         <!-- Ausgabe, wenn
Feld im Container "nicht gewählt" ist -->
<bx:recordfield.{field} not empty> Inhalt </bx:recordfield.{field}>     <!-- Ausgabe, wenn
Feld nicht "nicht gewählt", also entweder ja oder nein ist -->
```

Zusätzlich zu den beiden Statussen "j" und "n" also true und false, gibt es noch den Status "empty". Diesen Status hat das Feld, wenn es noch nie ausgewählt wurde (im Backend wurde beim Speichern des DS keine Feld mit dem entsprechenden Namen übergeben) oder wenn im Container direkt dieser Status ausgewählt wurde (siehe Bild "Datenansicht").

Ob dieser Status "nicht gewählt" erlaubt ist, kann in der Containerdefinition festgelegt werden. Dort kann auch ein Wert den entsprechenden Statussen zugewiesen werden (Felder "angezeigter Text...".

Definitionsansicht: image2015-7-21 12:54:54.png	Datenansicht: image2015-7-21 12:51:38.png

not	**Ausgabe, wenn das Feld nicht angehakt wurde (ohne: Ausgabe, wenn das Feld angehakt wurde) veraltet für `not` : invert und else**
empty	Ausgabe, wenn Status "nicht gewählt" zutriffe (Erklärung siehe Text oben)

Beispiele

```
<bx:recordfield.Infomaterial_gewuenscht empty>
  <input type="radio" name="Infomaterial_gewuenscht" value="j"> Infomaterial gewünscht
  <input type="radio" name="Infomaterial_gewuenscht" value="n"> kein Infomaterial gewünscht
</bx:recordfield.Infomaterial_gewuenscht>
```

```
<bx:recordfield.Infomaterial_gewuenscht not empty>
  Du hast dich bereits entschieden.
</bx:recordfield.Infomaterial_gewuenscht>
```

Das Feld "Infomaterial_gewuenscht" kann nur einmal ausgewählt werden und dann nie wieder.

```
<bx:recordfield.Häkchenfeld show="( real | config )" />  <!-- Ausgabe der in der Verwaltung
gespeicherten Einstellungen -->
<bx:recordfield.{field} show="config" [true="{text}"] [false="{text}"] [empty="{text}"] />
<!-- direkt im Tag angegebene Werte überschreiben die Einstellung -->
```

Eine Ausgabe entspechend des gespeicherten Statuses erfolgt.

true	**Text, der ausgegeben wird, wenn das Feld angehakt wurde**
false	Text, der ausgegeben wird, wenn das Feld nicht angehakt wurde
empty	Text, der ausgegeben wird, wenn das Feld den Status "nicht ausgewählt" hat (Erklärung siehe oben)
show	ab V 2.6.2 real: gibt den wirklich gespeicherten Wert (z.B. "j", "n", "1", "0") aus config: gibt den in der Verwaltung konfigurierten Wert aus (Standard: "wahr", "falsch")

Bisher gab es noch keine Option um zu sagen „nicht angehakt oder leer“ bzw. „angehakt oder leer“.

```
<bx:recordfield.{häkchenfeld} [not empty]
alt="{true|false}">...</bx:recordfield.{häkchenfeld}>
```

	`<bx:rf>`	`<bx:rf not>`	`<bx:rf empty>`	`<bx:rf not empty>`	`<bx:rf not alt="..">`	`<bx:rf alt="..">`
angehakt	angezeigt			angezeigt		angezeigt
nicht angehakt		angezeigt		angezeigt	angezeigt	
leer			angezeigt		angezeigt (wenn alt="false")	angezeigt (wenn alt="true")

einen der drei Werte abfragen	**Gegenteil(die beiden anderen Möglichkeiten)**

`<bx:recordfield.häkchen>` zeigt Block an, wenn das Feld angehakt ist	`<bx:recordfield.häkchen not alt="false">` zeigt Block an, wenn das Feld nicht angehakt oder leer ist
`<bx:recordfield.häkchen not>` zeigt Block an, wenn das Feld als nicht angehakt gespeichert ist	`<bx:recordfield.häkchen alt="true">` zeigt Block an, wenn das Feld angehakt oder leer ist
`<bx:recordfield.häkchen empty>` zeigt Block an, wenn das Feld leer ist	`<bx:recordfield.häkchen not empty>` zeigt Block an, wenn das Feld als angehakt oder nicht angehakt gespeichert ist

Beispiele

```
...
Infomaterial gewünscht? <bx:recordfield.Infomaterial_gewuenscht true="ja" false="nein"
empty="noch nicht entschieden" />
...
xml
<div id='status' style='background-color:<bx:recordfield.aktiviert true="green"
false="red"/>;'>...</div>
```

Bild

```
<bx:recordfield.{field} [width="{n}"] [height="{n}"] [nogifshrink] [cover]/>
```

Es wird das Bild (also ein Tag) ausgegeben, ggf. verkleinert, wenn gewünscht.

width	Angabe der max. Breite (abhängig von der Höhe)
height	Angabe der max. Höhe (abhängig von der Breite)
nogifshrink	keine Verkleinerung von GIF-Dateien
cover	(ab V 2.6.3, ansonsten bei 2.6.2 reinkopieren) Bild wird nur soweit verkleinert, daß es das angegebene Element ganz ausfüllt (siehe Beispiel 2)
cover="crop"	Originalbild wird abgeschnitten und auch so gespeichert

Beispiele

Beispiel 1

```
<div style="width:150px;height:150px;border:1px solid black;">
  <img src="<bx:recordfield.Bild width="150" height="150" type="path"/>" alt="">
```

```
</div>
```

Bild wird in den genannten Dimensionen ausgegeben. Es wird entweder auf die Breite oder Höhe verkleinert.

Beispiel 2

```
css:
.qimg{
  width:150px;
  height:150px;
  overflow:hidden;
  border:1px solid black;
  background-position:center;
  }
_____________________

<div class="qimg" style="background-image:url('<bx:recordfield.Bild width="150" height="150"
cover type="path"/>');">
  <img src="" alt="">
</div>
```

Bild wird deckend in den genannten Dimensionen ausgegeben. Es wird entweder in der Breite (bei Querformt) oder in der Höhe (bei Hochformat) beschnitten.

Unterschied: siehe Bild, halbtransparente Bildteile werden durch css abgeschnitten

beispiel-cover.jpg

Bild-Daten ausgeben

```
<bx:recordfield.{field} type="(path | url | title | alt | author | empty | id | inline)" [not]
/>
```

type	statt des eigentlichen Bildes kann folgendes ausgegeben werden: path: relativer Pfad url: komplette URL title, name: Titel des Bildes (Metadaten, Eingabe unter "Grafiken") originalname : Originaldateiname vom Upload licence: Lizenzangabe description: Beschreibung des Bildes alt: Alternativtext des Bildes (Metadaten, Eingabe unter "Grafiken") author: Autor des Bildes (Metadaten, Eingabe unter "Grafiken") empty: Abfrage, ob Feld leer ist (ab v 2.6.2) id: ID in der Bildergalerie (ab v 2.6.2) inline: erzeugt img-Tag mit Binärdaten in src (ab 2.6.1), sinnvoll bei Newsletter, da steht das Bild direkt in der Seite, also nicht als Link oder Anhang base64data: Tagerweiterung für vcf (Outlook-Visitenkarte, siehe Beispiel 2) mit Foto ab v2.6.6
not	in Verbindung mit empty, kehrt die Abfrage um

Beispiele

```
<bx:recordfield.Bild type="empty" not>
  <div style="background-image:url('<bx:recordfield.Bild type="path"/>');"
title="<bx:recordfield.Bild type="path"/>"> Inhalt </div>
</bx:recordfield.Bild>
```

Anzeige als Hintergrundbild, aber nur, wenn auch ein Bild vorhanden ist.

```
<bx:record.Visitenkarte pool="Ansprechpartner" id="request:editID">
BEGIN:VCARD
VERSION:2.1
N:<bx:recordfield.Name/>;<bx:if><bx:recordfield.Titel/> </bx:if><bx:recordfield.Vorname/>;;;
FN:<bx:if><bx:recordfield.Titel/> </bx:if><bx:recordfield.Vorname/> <bx:recordfield.Name/>
<bx:if>PHOTO;<bx:recordfield.Bild type="base64data"
pattern="TYPE=$type;ENCODING=BASE64:\n$data\n" width="200" height="200"/></bx:if>
ORG:<bx:recordfield.Unternehmen><bx:recordfield.Titel/></bx:recordfield.Unternehmen>;
<bx:if>TITLE: <bx:recordfield.Taetigkeit/></bx:if>
NOTE;ENCODING=QUOTED-PRINTABLE: <bx:recordfield.Beschreibung type="plain"/>
contact=0D=0A
TEL;WORK;VOICE:<bx:recordfield.Telefon/>
<bx:if>TEL;HOME;VOICE:</bx:if>
<bx:if>TEL;CELL;VOICE:<bx:recordfield.Mobil/></bx:if>
<bx:if>TEL;WORK;FAX:<bx:recordfield.Fax/></bx:if>
ADR;WORK:;;<bx:recordfield.Straße/>;<bx:recordfield.Ort/>;<bx:recordfield.PLZ/>;;
LABEL;WORK;ENCODING=QUOTED-PRINTABLE:<bx:recordfield.Straße/>=0D=0A<bx:recordfield.Ort/>,
```

```
<bx:recordfield.PLZ/> =0D=0A
<bx:if>URL:<WWLINK TYPE="GENERIC"
VALUE="<bx:recordfield.Internet/>"><bx:recordfield.Internet/></WWLINK></bx:if>
EMAIL;PREF;INTERNET:<bx:recordfield.Email/>
REV:
END:VCARD
</bx:record.Visitenkarte>
```

Erzeugung einer Visitenkarte für Outlook

Bild-Größenangaben ausgeben

```
<bx:recordfield.{field} type="(origwidth | origheight | calcwidth | calcheight)" />
```

Hier werden nur die Größen als Zahl ausgegeben (z.B.: für Javascript-Popups um die Größe zu übergeben).

origwidth	Ausgabe der Originalbreite
origheight	Ausgabe der Originalhöhe
calcwidth	Ausgabe der (berechneten) Endbreite
calcheight	Ausgabe der (berechneten) Endhöhe

Dokument

```
<bx:recordfield.{field} [type="(link | path | url | id | name | size | description | meta)"]
[name="{meta}"] [disposition="attachment|inline|none"]/>
<bx:recordfield.{field} [type="link"]> Inhalt </bx:recordfield.{field}>
```

Falls `type` nicht angegeben wurde, wird der Standardwert `link` verwendet.

type	**link: erzeugt einen klickbaren Link auf das Dokument (falls Inhalt angegeben, wird dieser als Linktext verwendet) path: gibt den relativen Pfad url: gibt die komplette URL aus id: gibt die ID aus name: gibt den Namen aus size: gibt die Dateigröße aus, description: gibt die Beschreibung aus meta: Metadatenabfrage, Name des Meta-Feldes wird mit `name` übergeben (siehe unten) empty: (ab V 2.6.2) Abfrage, ob Feld leer ist**
name	(nur in Verbindung mit `type="meta")` Name der Meta-Eigenschaft

type	link: erzeugt einen klickbaren Link auf das Dokument (falls Inhalt angegeben, wird dieser als Linktext verwendet) path: gibt den relativen Pfad url: gibt die komplette URL aus id: gibt die ID aus name: gibt den Namen aus size: gibt die Dateigröße aus, description: gibt die Beschreibung aus meta: Metadatenabfrage, Name des Meta-Feldes wird mit `name` übergeben (siehe unten) empty: (ab V 2.6.2) Abfrage, ob Feld leer ist
unit	(nur in Verbindung mit `type="size")` auto: wählt automatisch die entsprechende Einheit, je nach Größe der Datei b: Angabe in Byte kb: Angabe in Kilobyte mb: Angabe in Megabyte
pattern	(nur in Verbindung mit `type="size")` Formatierung der Zahl (Standardwert ist "#,##0")
locale	(nur in Verbindung mit `type="size")` Sprachen-Formatierung der Zahl (standardmäßig "de_DE")
disposition	siehe Beispiele

Beispiele

```
<bx:recordfield.Dokument disposition="inline"/>
```

Ausgabe: `http://domain/files/{Batix-ID}/{Dateiname}` setzt content-disposition:inline; filename="{Dateiname}", wird als Standard genommen, wenn nichts angegeben wird

```
<bx:recordfield.Dokument disposition="attachment"/>
```

Ausgabe: `http://domain/files/download/{Batix-ID}/{Dateiname}` bewirkt bei PDFs einen Zwangsdownload anstatt eine Anzeige im Browser-Plugin, dasselbe auch durch `http://domain/files/{Batix-ID}/{Dateiname}?dl=true`

```
<bx:recordfield.Dokument disposition="none"/>
```

Ausgabe: `http://domain/files/media/{Batix-ID}/{Dateiname}` bewirkt bei MP4-Videos, die in Apple-Mobilgeräten abgespielt werden sollen, daß das funktioniert, weil der Header nicht geschrieben wird (scheint nur Apple-Geräte zu betreffen)

Pixelkoordinaten

```
<bx:recordfield.{field} type="(x | y) [multiply="{n}"] [add="{n}"] [alt="{n}"] />
```

type	Je nach `type` wird entweder die x- oder die y-Koordinate ausgegeben
multiply	Angabe zum Multiplizieren des Koordinatenwertes (bei Kartenvergrößerund bzw. -verkleinerung)
add	Wert, der zum Koordinatenwerte dazuaddiert werden kann
alt	Alternativwert, der ausgegeben wird, wenn keine Zahl festgelegt ist

Einfachverknüpfung

```
<bx:recordfield.{field} [chkuser] [show="active|inactive|all"] <not> > Inhalt
</bx:recordfield.<field>>
<bx:recordfield.{field} type="id" />
<bx:recordfield.{field}> />
<bx:recordfield.{field} type="if" value="{ID}" <not>> Inhalt </bx:recordfield.<field>>
```

Innerhalb des Tags kann wiederum mittels `bx:recordfield` auf die Elemente der Verknüpfung zugegriffen werden. Falls kein Vergleich passt, wird der Inhalt nicht ausgegeben. Falls kein Inhalt angegeben ist, wird der Titel des Datensatz-Feldes ausgegeben.

chkuser	ID des Datensatzes mit der User-ID des aktuell eingeloggten Users bzw. mit den IDs seiner Gruppen verglichen.
type="id"	gibt die ID des Datensatzes aus
type="if"	(ab V *2.5.8*) Inhalt des Tags wird nur dann angezeigt, wenn die ID des verknüpften Datensatzes mit der in `value` angegebenen übereinstimmt
value	zu prüfenden ID. Mehrere zu prüfende IDs können durch Leerzeichen oder Kommas getrennt werden. **#null**: "nichts verknüpft" wird abgefragt **#all**: hebt alle Bedingungen auf, zeigt alles an (siehe Beispiel)
not	kehrt die Bedingung um
show	active: nur aktive DS (Standard) inactive: nur inaktive DS all: aktive und inaktive DS werden angezeigt

Beispiele

Beispiel für den Einsatz von nofilter

```
<bx:clipboard.cut>#all</bx:clipboard.cut>
 <bx:irgendeineBedingung><bx:clipboard.cut>eine
Kategorieid</bx:clipboard.cut></bx:irgendeineBedingung>
 <bx:recordfield.Einzelkategorie type="if"
value="clipboard:default">...</bx:recordfield.Einzelkategorie>
```

Clipboard wird mit `#all` initialisiert Clipboard wird mit einer ID gefüllt oder auch nicht wenn nicht gefüllt, greift das #all und hebt alle Bedingungen auf (`ID` = gibt aus, wenn es diese ID ist, `#null` = gibt aus, wenn nichts verknüpft ist) und gibt bei jedem DS aus

Mehrfachverknüpfung

```
<bx:recordfield.{field} [forceindex="<n>" | indexparam="<param>"] [max="<n>"]
[idfield="<param>"] [force = {single | list}] [orderby="<field>"] [desc] [chkuser]
[type="quantity"] [show="{active | inactive | all}"]>
 Inhalt
</bx:recordfield.{field}>

<bx:recordfield.{field} zusätzliche Parameter [boundary="<text>"] [type="ids"] />
```

Hier stellt sich bx:recordfield als Schleife dar, welche für jeden Unter-Datensatz durchlaufen wird.

forceindex	Stelle, an der die Ausgabe beginnen soll
indexparam	Parametername, wo die Stelle, an der die Ausgabe beginnen soll, steht
max	maximale Anzahl der auszugebenden Datensätze (z.B. max="5")
idfield	Parametername, der die ID des Datensatzes enthält (Standard ist der Name des Feldes)
force	Ausgabemodus (single oder list) kann erzwungen werden
orderby	Feldangabe, nach der die Ausgabe sortiert sein soll (Datum, Name etc.)
desc	dreht die Sortierrichtung um
chkuser	DS-ID und ID seiner zugeordneten Gruppen wird mit der ID des eingeloggten Users verglichen, dh: Wenn im DS eine Gruppe zugeordnet ist, dann wird der Inhalt ausgegeben

forceindex	Stelle, an der die Ausgabe beginnen soll
type	quantity: Anzahl der verknüpften Datensätze anzuzeigen
boundary	Trennzeichen, wenn die Titel aller Elemente ausgegeben werden (Standard ","), sollen nur die IDs ausgegeen werden, muß `type="ids"` angegeben werden (Tag ohne Inhalt)
show	auch inaktive anzeigen lassen ab v2.6.6

Erweiterung für neue Container (v2.5.8) Werte für `type`:

```
<bx:recordfield.{field} type="[ ids | titel | quantity ]" [boundary="{text}"]/>

<bx:recordfield.{field} type="[ if-any | if-none | if-all ]" value="{Liste von IDs}"> ...
</bx:recordfield.{field}>

<bx:recordfield.{field} type="[ if-empty | computebody | chkuser ]" [not] [nosv]> Inhalt
</bx:recordfield.{field}>
```

type=	ids	gibt die verknüpften ID's mit Leerzeichen getrennt aus
	titel	gibt zur Zeit den Titel aus z. B. "2 verknüpfte Datensätze" (Standard wenn type nicht angegeben und geschlossenes Tag)
	quantity	gibt die Datensatzanzahl aus. "0" wenn keine Datensätze verknüpft
	if-any	gibt den Inhalt aus, wenn ein verknüpfter Datensatz die angegebene ID hat
	if-none	gibt den Inhalt nur aus, wenn keiner der verknüpften Datensätze die angegebene ID hat
	if-all	gibt den Inhalt aus, wenn alle angegebenen IDs im Datensatz verknüpft sind
	if-empty	gibt den Inhalt aus wenn kein Datensatz verknüpft ist not: wenn mindestens 1 Datensatz verknüpft ist

type=	ids	gibt die verknüpften ID's mit Leerzeichen getrennt aus
	computebody	gibt eine Schleife für jeden verknüpften Datensatz aus (Standard, wenn type nicht angegeben und offenes Tag) Die Parameter von oben sind noch gültig (force, idfield, max, indexparame, orderby nofilter)
	chkuser	wenn das Feld eine BC-Verknüpfung ist, wird geprüft, ob es mit dem eingeloggten Benutzer/Gruppe übereinstimmt (wie bei if-empty)
boundary		damit kann man den Trenner angeben, z.B. boundary="," Standard ist Leerzeichen
nosv		wenn Admin, der ja in allen Gruppen drin ist, eingeloogt, wird er keiner Gruppe zugewiesen

Untercontainer

```
<bx:recordfield.{field} [forceindex="{n}" | indexparam="{param}"] [max="{n}"]
[idfield="{param}"] [force="{single | list}"] [orderby="{field}"] [desc] [type="empty |
quantity"] [show="active|inactive|all"] [not]> Inhalt </bx:recordfield.<field>>

<bx:recordfield.{field} [filterfield="{field}" [filtertype="{n}"] filtervalue="{text}"
[filterrequired="{true|false}"] [nofilter]> Inhalt </bx:recordfield.<field>>
```

Dieser Typ ist der Mehrfachverknüpfung ähnlich. Für jeden Datensatz wird der Inhalt des Tags angezeigt. Ab *v2.5.?* gibt es die Möglichkeit einen Filter zu definieren, nach dem die Unterdatensätze gefiltert werden

forceindex	Stelle, an der die Ausgabe beginnen soll
indexparam	Parametername, wo die Stelle, an der die Ausgabe beginnen soll, steht
max	maximale Anzahl der auszugebenden Datensätze (z.B. max="5")
idfield	Parametername, der die ID des Datensatzes enthält (Standard ist der Name des Feldes)
force	Ausgabemodus (single oder list) kann erzwungen werden

forceindex	Stelle, an der die Ausgabe beginnen soll
orderby	Feldangabe, nach der die Ausgabe sortiert sein soll (Datum, Name etc.) Sortierung nach Anlegereihenfolge: Angabe des Feldes, das die Parent-ID enthält
desc	dreht die Sortierrichtung um
nofilter	Filtereigenschaften werden nicht vom übergeordneten Container übernommen (siehe Beispiel unten)
filterfield	gibt den Namen des Feldes im Untercontainer an, das durchsucht werden soll
filtertype	Vergleichstyp (siehe Filter , standardmäßig auf 8 gestellt)
filtervalue	Static-Wert oder Parameter-Werte
filterrequired	required-Flag für das Filter-Objektsetzen (standardmäßig false)
type not	empty: Abfrage, ob der UK leer ist. `not` kehrt die Abfrage um quantity: gibt die Anzahl aus
show	active: nur aktive DS (Standard) inactive: nur inaktive DS all: aktive und inaktive DS werden angezeigt

Beispiele

Beispiel für eine Filterung des Untercontainers

```
<bx:recordfield.Sub filterfield="Text" filtertype="1" filtervalue="request:suche">
  <bx:recordfield.Text />
</bx:recordfield.Sub>
```

Beispiel für den Einsatz von nofilter

```
<bx:containerfilter.Kuecheninhalt pool="Kuecheninhalt">
  ...
  <bx:recordfield.Lebensmittel>
    ..
  </bx:recordfield.Lebensmittel>
  ..
</bx:containerfilter.Kuecheninhalt>
```

Filter für den Container "Kücheninhalt" (alle DS, die im UK "Lebensmittel" das Wort "brot" im Feld "Titel" enthalten).):

```
<filter-object>
  <field>Lebensmittel/Titel</field>
  <type>1</type>
  <static-value>brot</static-value>
</filter-object>
```

Verhalten, wenn `nofilter` **nicht gesetzt** ist:

Bei Ausgabe des Recordfield "Lebensmittel" werden nur die DS, die im Titel "brot" enthalten, ausgegeben, da der Filter vom Containerfilter übernommen wurde. `nofilter` verhindert das und alle DS werden ausgegeben.

Veraltete Tags

Die folgenden Befehle sind veraltet und werden durch bx:recorddata ersetzt.

```
<bx:recordfield.parentid />□□          <!-- Gibt die ID des übergeordneten Objekes zurück. -->
<bx:recordfield.recordid />                <!-- Gibt die ID des Feldes aus. -->
<bx:recordfield.rootid />                  <!-- Die ID des zugehörigen Hauptcontainers wird
ausgegeben. -->
<bx:recordfield.previous object="<name>" />  <!-- veraltet: stattdessen
bx:recorddata.previouslist benutzen -->
<bx:recordfield.next object="<name>" />     <!-- veraltet: stattdessen bx:recorddata.nextlist
-->
```

Verweise auf diese Seite:

- Container-Filteraction
- bx:tablefield
- Container
- bx:recordfield
- Container
- bx:tablefield
- Container-Filteraction
- bx:recordfield

(8 Verweise)