bx:pagedata

Mithilfe des Tags `pagedata` lassen sich die verschiedensten Informationen die Seite betreffend ausgeben.

pagedata.parentnavid

```
<bx:pagedata.parentnavid/>
```

Es wird die Navigations-ID des Elternpunktes (oder leer) ausgegeben.

pagedata.webid

```
<bx:pagedata.webid/>
<bx:pagedata.customerid/><!-- veraltet -->
```

Gibt die ID des Webs (Projekts) aus

pagedata.webdir

```
<bx:pagedata.webdir/>
<bx:pagedata.customerdir/><!-- veraltet -->
```

gibt das virtuelle Verzeichnis des Webs aus.

pagedata.webhost

```
<bx:pagedata.webhost [addscheme]/>
```

(Verfügbar ab Version 2.7)

Der beim Projekt eingegebene Haupthost wird ausgegeben. Wenn "addscheme" angegeben ist, wird auch das beim Projekt eingestellte Protokoll ausgegeben. Dies ist bei Canonical-Angaben nützlich, wenn nicht der aktuelle Host (wie bei `<bx:pagedata.hostname/>`) gebraucht wird, sondern die Hauptdomain.

pagedata.virtualpath

```
<bx:pagedata.virtualpath [original]/>
```

Der virtuelle Pfad des Menüpunktes wird ausgegeben. z.B. von http://www.beispielseite.de/produkte/textiles/pullover/ wird ausgegeben: produkte/textiles/pullover

original

es wird der ursprüngliche Pfad ausgegeben (bei forwards)

pagedata.navid

```
<bx:pagedata.navid [level="{n}"]/>□□□□□□□<!-- gibt nur die ID aus -->
<bx:pagedata.navid is="{id}"> Inhalt </bx:pagedata.navid>□<!-- Inhalt wird nur dann ausgegeben,
wenn die ID des aktuellen Menüpunktes mit der ID von "is" übereinstimmt" -->
```

Navigations-ID des aktuellen Menüpunktes oder eines Elternpunktes Um auch mehrere IDs, die Eltern-ID oder den Pfad abfragen zu können, benutzen Sie bitte `pagedata.nav`.

level

Ebene des gewünschten Navigationspunktes. Hauptmenüpunkt entspricht `level="0"`, Untermenüpunkte liegen jeweils eine Ebene höher

is

vorgegebene ID

not

funktioniert hier nicht, bitte pagedata.nav verwenden

pagedata.navpath

```
<bx:pagedata.navpath [previous | next] [active | inactive | all] />
```

Gibt den virtuellen Pfad des Menüpunktes aus, z.B. von http://www.beispielseite.de/produkte/textiles/pullover/ wird ausgegeben: pullover

previous / next

sucht den vorherigen bzw. nächsten Menüpunkt im gleichen Level

active / inactive / all

- aktive anzeigen (Standard)
- nur inactive anzeigen
- alle anzeigen

Beispiele

```
<div style="float:left;width:100px;">
  <a href="<bx:pagedata.navpath previous/>">&laquo; vorherige Seite</a>
</div>
<bx:pagedata.navname />
<div style="float:right;width:100px;text-align:right;">
  <a href="<bx:pagedata.navpath/>">nächste Seite &raquo;</a>
</div>
```

Ein Blättern zwischen den (aktiven) Menüpunkten einer Ebene. Sieht dann ungefähr so aus:

« vorherige Seite	Angebote für Juli	nächste Seite »

pagedata.navname

```
<bx:pagedata.navname [level="{n}"] />
```

Gib den Name des Menüpunktes oder eines Elternpunktes aus.

level

Ebene des gewünschten Navigationspunktes (nur höhere Ebenen vom aktuellen Menüpunkt aus). Hauptmenüpunkt entspricht `level="0"`, Untermenüpunkte liegen jeweils eine Ebene höher

pagedata.nav

```
<bx:pagedata.nav {id="{id}" | parent="{id}" | path="{pfad}"} [not] [original]> Inhalt
</bx:pagedata.nav>
```

Entscheidet, ob Inhalt ausgegeben wird anhand von ID(s) oder Pfad

id

ID des gesuchten Menüpunktes ab V 2.6.2: es können mehrere ID mit Komma getrennt angegeben werden

parent

ID des Elternmenüpunktes (findet auch sich selbst als Punkt) ab V 2.7: es können mehrere ID mit Komma getrennt angegeben werden

path

Pfad des aktuellen Menüpunktes

not

kehrt die Bedingung um

original

es wird die Angabe von der ursprünglichen Seite genommen (z.B. bei einem jsp-include)

pagedata.request

Ab V 2.6.3 RC14 wird in den meisten Fällen ein HTML-Encoding gemacht, wo kein encode-Parameter angegeben ist und auch kein `<bx:tools.htmlencode>` außen herum ist. Dies verursacht manchmal **Probleme**:

- SQL-Filter: wenn encode=sql fehlt, wird jetzt HTML-Encoded (aus WHERE Ort LIKE '%Köln%' wurde WHERE Ort LIKE '%Köln%' und es wurde nichts mehr gefunden)
- `<bx:clipboard.cut><bx:pagedata.request/></bx:clipboard.cut>` ... `<bx:tools.htmlencode><bx:clipboard.paste/></bx:tools.htmlencode>` führt jetzt zu doppelten kodieren.

```
<bx:pagedata.request name="{param}" [boundary="{text}"] [empty] encode="{ html | javascript | sql | sql-like | sql-rlike | url | xml | plain}"/>□<!-- Ausgabe des Request-Parameters -->
<bx:pagedata.request name="{param}" value="<text>" [true="{text}"] [false="{text}"] />

<bx:pagedata.request name="{param}" [not] [empty]> Inhalt </bx:pagedata.request>□□□□□<!-- Auswertung des Request-Parameters -->
<bx:pagedata.request name="{param}" value="{text}" [not]> Inhalt </bx:pagedata.request>
```

Request-Parameter auswerten

`<bx:pagedata.request ... encode="html|javascript|sql|sql-like|sql-rlike|url|xml">`

javascript und url gibt's schon seit v2.3, den Rest ab v2.6.2

name

Name des Request-Parameters, kann auch einen Platzhalter in der Form `name="[platz]id"` enthalten (siehe Beispiel)

encode

Codierung des Wertes, für JavaScript-Code (Sonderzeichen werden escaped) oder als URL (%nn). Hiermit kann der ausgegebene Text für verschiedene Formate kodiert werden:

- html
- javascript
- sql (zur Verwendung in SQL für Strings)

- sql-like (zur Verwendung in SQL mit LIKE)
- sql-rlike (zur Verwendung in SQL mit RLIKE und REGEXP)
- url
- xml
- plain (überschreibt automatisches Encoding)
- htmltext: siehe Beipspiel (3)

Das erspart ein bx:tools, was ansonsten um das bx:pagedata gepackt werden müsste.

boundary

Falls der Parameter mehrfach vorkommt (?para=wert1&para=wert2) wird ohne Angabe von `boundary` nur der Erste ausgegeben, andernfalls wird der Wert von `boundary` als Trennzeichen benutzt.

not / empty

Falls `empty` gesetzt ist, wird ein leerer Request-Parameter wie ein nicht vorhander behandelt.

	seite.htm	**seite.htm?p=**	**seite.htm?p=wert**
`<bx:pagedata.request name="p">...`	leer	angezeigt	angezeigt
`<bx:pagedata.request name="p" not>...`	angezeigt	leer	leer
`<bx:pagedata.request name="p" empty>...`	leer	leer	angezeigt
`<bx:pagedata.request name="p" not empty>...`	angezeigt	angezeigt	leer

value

Text, den der Request-Parameter enthalten soll

true / false

Wenn der Wert des Parameters mit `value` verglichen wird kann entweder `true` (wird ausgegeben, wenn gleich) und/oder `false` (wird ausgegeben, wenn ungleich) verwendet werden oder man setzt den Inhalt, welcher dann bei Übereinstimmung ausgegeben wird (umkehrbar durch `not`). ab v2.6.6: es kann auch abgefragt werden, ob der Parameter übergeben wurde oder nicht (siehe Beispiel)

Beispiele

Fall 1 true/false

```
<bx:pagedata.request name=“test“ value=“3“ true=”ist drei” false=”ist nicht drei” />
```

Fall 2 true/false

```
<bx:pagedata.request name=“test“ true=”test ist übergeben” false=”test ist nicht übergeben” />
```

es wird nicht der Wert abgefragt, sondern ob überhaupt der Parameter übergeben wurde

```
<bx:pagedata.request name=“test“ encode=”htmltext”/>
```

Eingabe: test.htm?test=erste+Zeile%0azweite+Zeile («%0a» ist ein Zeilenumbruch) → schreibt «erste Zeile
zweite Zeile» und damit sieht man einen Zeilenumbruch im Browser

pagedata.attribute

Beispiele

```
speichern.act?feld=[]/Name[containerID]&containerID=12345
```

Bissl weit hergeholtes Beispiel: ... ich hab keine Ahnung!

```
speichern.act?feld=[]/Name[containerID]&containerID=12345
xml
test.htm?test=erste+Zeile%0azweite+Zeile  // "%0a" ist ein Zeilenumbruch

<bx:pagedata.request name="test" encode="htmltext"/> // schreibt erste Zeile<br>zweite Zeile
und damit sieht man einen Zeilenumbruch im Browser
xml
<bx:pagedata.attribute name="{attrib}" />□□□□□□□□□□□□□□□<!-- schreibt den Wert des Attributes (f
vorhanden) in die Seite -->
<bx:pagedata.attribute name="{attrib}" [not]> Inhalt </bx:pagedata.attribute>□□□□□<!-- gibt den
```

```
Tag-Inhalt aus, wenn das Attribut vorhanden ist -->
<bx:pagedata.attribute name="{attrib}" value="{text}" [not]> Inhalt </bx:pagedata.attribute>
<!-- vergleicht den Wert des Attributes mit dem Wert von value und gibt den Tag-Inhalt bei
Übereinstimmung aus -->
```

Ein Request-Attribute, das z.B. durch JSP-Includes oder das bx:setattribute gesetzt wurde, kann ausgewertet werden.

name

Name des Request-Attributs

value

beliebiger Text, der mit dem Attribut übereinstimmen muß (oder auch nicht bei `not`)

pagedata.setattribute

```
<bx:pagedata.setattribute mode="[copy | set | remove]" name="{name}"> Inhalt
</bx:pagedata.setattribute>
```

Ein Request-Attributekann gesetzt, geändert oder gelöscht werden.

name

Name des Request-Attributs

mode

`copy:` Zeigt den Taginhalt in der Seite an und kopiert ihn in das Request-Attribut `set:` Kopiert den Inhalt in das Request-Attribut aber zeigt ihn nicht in der Seite an `remove:` löscht das Request-Attribut (das Tag wird geschlossen verwendet)

pagedata.scriptattribute

```
<bx:pagedata.scriptattribute name="{attrib}" />□□□□□□□□□□□□□□□□□<!-- gibt den Wert des Script-At
vorhanden) aus -->
<bx:pagedata.scriptattribute name="{attrib}" [not]> Inhalt </bx:pagedata.scriptattribute>□□□□□<
gibt den Tag-Inhalt aus, wenn das Script-Attribut vorhanden ist -->
<bx:pagedata.scriptattribute name="{attrib}" value="{text}" [not]> Inhalt
</bx:pagedata.scriptattribute>□<!-- vergleicht den Wert des Script-Attributes mit dem Wert von
value und gibt den Tag-Inhalt bei Übereinstimmung aus -->
```

Ein Script-Attribut, das z.B. durch Actions gesetzt wurde, kann ausgewertet werden.

name

Name des Script-Attributs

value

beliebiger Text, der mit dem Attribut übereinstimmen muß (oder auch nicht bei `not`)

pagedata.setscriptattribute

```
<bx:pagedata.setscriptattribute mode="[copy | set | remove]" name="{name}" [trim]> Inhalt
</bx:pagedata.setscriptattribute>
```

Ein Script-Attribut kann gesetzt, geändert oder gelöscht werden.

name

Name des Script-Attributs

mode

`copy:` Gibt den Taginhalt aus und kopiert ihn in das Script-Attribut `set:` Kopiert den Taginhalt in das Script-Attribut aber zeigt ihn nicht an `remove:` löscht das Script-Attribut (das Tag wird geschlossen verwendet)

pagedata.listrequest

```
<bx:pagedata.listrequest/>
```

alle Request-Parameter werden als 'hidden' Formularfeld in die Seite geschrieben

pagedata.querystring

```
<bx:pagedata.querystring [original]/>
```

QueryString ausgeben

Ein '?' am Anfang ist in der Ausgabe nicht enthalten. Ge**POST**ete Parameter sind nicht enthalten! Falls dieses Tag im HTML-Quelltext (z.b. für Links) verwendet wird, muss die Ausgabe noch durch bx:tool.urlencode umgewandelt werden.

original

es wird der ursprüngliche QueryString ausgegeben (bei forwards)

pagedata.meta

```
<bx:pagedata.meta name="<meta>" />□□□□□□□□□□□□□□□<!-- schreibt den Wert der Meta-Eigenschaft (fal
vorhanden) in die Seite -->
<bx:pagedata.meta name="<meta>" [not]> Inhalt </bx:pagedata.meta>□□□□□<!-- gibt den Tag-Inhalt
wenn die Meta-Eigenschaft vorhanden ist -->
<bx:pagedata.meta name="<meta>" value="<text>" [not]> Inhalt </bx:pagedata.meta>□<!--
vergleicht den Wert der Eigenschaft mit dem Wert von value und gibt den Tag-Inhalt bei
Übereinstimmung aus -->
```

Metadaten auswerten

name

Name der Meta-Eigenschaft

value

beliebiger Text, der mit der Meta-Eigenschaft übereinstimmen muß (oder auch nicht bei `not`)

pagedata.cookie

```
<bx:pagedata.cookie name="{cookie}" />□□□□□□□□□□□□□□<!-- schreibt den Wert des Cookies (falls vo
die Seite -->
<bx:pagedata.cookie name="{cookie}" [not]> Inhalt </bx:pagedata.cookie>□□□□□<!-- gibt den Tag-
Inhalt aus, wenn der Cookie vorhanden ist -->
<bx:pagedata.cookie name="{cookie}" value="{text}" [not]> Inhalt </bx:pagedata.cookie>  <!--
vergleicht den Wert des Cookies mit dem Wert von value und gibt den Tag-Inhalt bei
Übereinstimmung aus -->
```

Wertet vom Browser übergebene Cookies aus.

name

Name des Cookies

value

beliebiger Text, der mit dem Cookie-Wert übereinstimmen muß (oder auch nicht bei `not`)

pagedata.hostname

```
<bx:pagedata.hostname [addscheme] />□□□□□□□□□□□□□□<!-- gibt den Hostnamen des Servers aus (wenn
Ausgabe = hostname:port -->
<bx:pagedata.hostname match="{regex}" [addscheme] [not]> Inhalt </bx:pagedata.hostname>□<!--
vergleich des Hostnames mit einem regulären Ausdruck (ab v2.5.5)-->
```

Der Hostname (und evtl. Port) der aktuellen Anfrage wird ausgegeben.

match

Regulärer Ausdruck

addscheme

gibt zusätzlich das Protokoll (http:// oder https://) des aktuellen Seitenaufrufs aus

Beispiele

Wenn man im Web von batix. com folgende Tags einbindet:

```
<bx:pagedata.hostname/>
```

wir ausgegeben: www.batix.com

```
<bx:pagedata.hostname addscheme/>
```

wird ausgegeben: http://www.batix.com (ab v2.5.8)

```
<bx:pagedata.hostname match=".*\.[com|net]"> Sie haben unsere net- oder com-Domain aufgerufen.</bx:pagedata.hostname>
```

beim Aufruf von www.batix.de oder www.batix.eu wird der Text nicht ausgegeben, beim Aufruf von www.batix.com oder www.batix.net schon.

pagedata.sessionid

```
<bx:pagedata.sessionid/>
```

Die Session-ID wird ausgegeben. Wenn es noch keine Session gibt, dann wird eine erstellt und dann ausgegeben, außer man gibt "no-create" an* (ab V 2.6.9)*

pagedata.sessionurl

```
<bx:pagedata.sessionurl/>
```

Session-URL-Komponente (;jsessionid=) ausgeben, falls Cookies ausgeschaltet sind

pagedata.url

```
<bx:pagedata.url [encode="html"] />
```

Dieser Befehl schreibt die komplette Request-URL inklusive Query-String in die Seite. Falls `encode="html"` angegeben ist, werden <, >, ", & und ' durch ihre HTML-Umschreibungen ersetzt.

pagedata.filename

```
<bx:pagedata.filename [original]/> □□□□□□□□□□□□□□□□<!-- der Dateiname wird ausgegeben (z. B. "in
<bx:pagedata.filename equals="{vergleichsname}" [not] [original]> ... </bx:pagedata.filename>□
<!-- Taginhalt wird nur ausgeführt, wenn der aktuelle Dateiname dem angegebenen Namen
entspricht (ab v2.5.7) -->
```

Gibt den virtuellen Dateinamen aus der gerade aufgerufenen URL zurück. Wenn die Seite mit "/" aufgerufen wird, behandelt das Tag den Namen als "index.htm".

original

es wird der ursprüngliche Dateiname ausgegeben (bei forwards)

pagedata.errortext

```
<bx:pagedata.errortext/>
```

Fehlertext in eine Fehlerseite schreiben. Der Befehl ist nur in der Fehlerseite "/www/[host-dir]/errorpage/index.htm" verfügbar. (ab* v2.3.15)*

pagedata.errorstacktrace

```
<bx:pagedata.errorstacktrace/>
```

Schreibt den Stacktrace einer Exception in die Fehlerseite. Der Befehl ist nur in der Fehlerseite "/www/[host-dir]/errorpage/index.htm" verfügbar. *seit v2.5.8*

pagedata.remoteip

```
<bx:pagedata.remoteip/>□□□<!-- schreibt die IP-Adresse des Clients (der die Abfrage macht) in d
Seite. (z. B. um sie in einen Container zu speichern)  -->
<bx:pagedata.remoteip equals="{testip}" [not]> Inhalt </bx:pagedata.remoteip>□□<!-- Vergleich
der IP und entsprechende Ausgabe des Tag-Inhalts -->
<bx:pagedata.remoteip host="{hostname}" [not]> Inhalt  </bx:pagedata.remoteip>
<bx:pagedata.remoteip matches="{regex}" [not]> Inhalt  </bx:pagedata.remoteip>
```

(ab v2.5.7)

Beispiele

```
<bx:pagedata.remoteip matches="123\.210\.12\.[0-9]+">
 <a href="...">Intranet</a>
</bx:pagedata.remoteip>
```

Inhalt nur für IP-Adressen im Bereich "123.210.12.*" zeigen

pagedata.scheme

```
<bx:pagedata.scheme/>
```

Gibt das Protokoll zurück. ("http" oder "https") (ab* v2.5.8*)

pagedata.compressed

```
<bx:pagedata.compressed />□□□<!-- es wird ".bxmin" ausgegeben* -->
<bx:pagedata.compressed>.bxmin</bx:pagedata.compressed>  <!-- der Inhalt des Body (im Beispiel
ebenfalls ".bxmin")wird ausgegeben, ansonsten wird keine Ausgabe gemacht* -->
```

Unterscheidet abhängig vom Projektstatus die Einstellung der Scriptkomprimierung (ab* v2.6.2*). *Greift, wenn das Projekt auf Produktiv und "Style- und Script-Compressor" im Projekt angehakt ist.

pagedata.status

```
<bx:pagedata.status type="{dev | prod | debug}"> ... </bx:pagedata.status>
```

Zeigt abhängig vom Projektstatus den Taginhalt an (ab* v2.6.2*).

Beispiele

```
<script src="tools<bx:pagedata.status type="prod">.bxmin</bx:pagedata.status>.js"></script>
```

Wenn das System auf "Produktiv" gestellt ist, wird tools.bxmin.js ausgegeben, sonst tools.js

pagedata.charset

```
<bx:pagedata.charset />
```

Gibt den Zeichensatz aus, der beim Projekt bzw. beim Design angegeben wurde. Wenn bei beiden nichts angegeben wurde, wird nichts ausgegeben und es gilt der Standard der Servlet-API (iso-8859-1)

pagedata.writecontenttype

```
<bx:pagedata.writecontenttype/>
```

Gibt folgendes aus: `<meta http-equiv="Content-Type" content="[mimetype]; charset=[Zeichensatz]">`
Bisher wird es immer von Hand reingeschrieben und bei Änderung des Seiteneincodngs passen die Zeichensätze nicht mehr zusammen.

pagedata.header

```
<bx:pagedata.header name="..." (value|matches|contains)="[Vergleichswert]" [not]>
```

Beispiele

```
<a href="<bx:pagedata.header name="Referer"/>">zurück</a>
<bx:pagedata.header name="Referer" not>Direkt eingetippt</bx:pagedata.header>
<bx:pagedata.header name="accept-language" contains="es">¡hola!</bx:pagedata.header>
<bx:pagedata.header name="User-Agent" matches=".*(google|bing|yahoo).*">Hallo
Suchmaschine!</bx:pagedata.header>
```

Verweise auf diese Seite:

- Container-Filteraction
- bx:tools
- bx:tools
- Support für not hinzufügen
- Google-Sitemap Datei erzeugen
- Container-Filteraction
- Google-Sitemap Datei erzeugen
- Support für not hinzufügen

(8 Verweise)