Snippets & Rezepte

Fertige Code-Bausteine für häufige Aufgaben.

Action-Parameter anlegen

Es kann sein, daß man innerhalb eines Actions Parameter benötigt, die im Request nicht übergeben wurden. Hierzu kann man den Action-Baustein "Batix Quelltext ausführen" im Zusammenhang mit dem Tag "bx:pagedata.setscriptattribute" benutzen. Hierzu ein paar Beispiele:

Umwidmen eines Request-Parameters

```
<bx:pagedata.setscriptattribute name="Email"><bx:pagedata.request
name="Imehl"/></bx:pagedata.setscriptattribute>
```

Bestandteil einer Botfalle

Quelltext der Seite

```
<bx:containerfilter.Mitarbeiter pool="Mitarbeiter" idfield="mid" force="single">
  <bx:pagedata.setscriptattribute
name="Email"><bx:recordfield.Email/></bx:pagedata.setscriptattribute>
</bx:containerfilter.Mitarbeiter>
```

Wenn man im nachfolgenden Action die Email braucht, diese aber nicht übergeben wurde (z.B. für Email-Versand-Action)

Link für Weiterleitung zusammenbasteln

```
<bx:pagedata.setscriptattribute name="weiterleitung">
  <bx:pagedata.request name="abgeschlossen"
value="j">/test/naechsteSeite.htm</bx:pagedata.request>
  <bx:pagedata.request name="abgeschlossen"
value="n">/test/gleicheSeite.htm</bx:pagedata.request>
</bx:pagedata.setscriptattribute>
```

Im anschließenden Weiterleitungsaction kann der Parameter mit [[abgeschlossen]] eingesetzt werden.

Blättern (Pagination)

Einfaches Listenblättern

sieht so aus:

Die Leiste oben und unten kann in Bausteine ausgegliedert werden. Dann kann Sie für jede Liste im Web genutzt werden. Voraussetzung: die Liste heißt dann immer "Liste", also `<bx:containerfilter.Liste pool="...">`

HTML in der Seite

```
<bx:include modul="blaetternOben"/>

<bx:containerfilter.Liste pool="Witze" orderby="Titel" max="5">
  <div class="listitem"<bx:recorddata.if last> style="margin-bottom:0;"</bx:recorddata.if>>
    <a href="detail.htm?rezeptid=<bx:recorddata.id/>"><bx:recordfield.Titel/></a>
  </div>
</bx:containerfilter.Liste>

<bx:include modul="blaetternUnten"/>
```

ausgelagertes Include "blaetternOben"

```
<div class="blaetternoben">
  <bx:recorddata.nav object="Liste">
    <bx:recorddata.startindex/> - <bx:recorddata.endindex/> von <bx:recorddata.total/>
Einträgen
  </bx:recorddata.nav>
```

```
</div>
```

ausgelagertes Include "blaetternUnten"

```
<div class="blaetternunten">
  <bx:recorddata.nav object="Liste">
    <bx:if><bx:recorddata.previouslist><< zurück</bx:recorddata.previouslist><bx:if.else><span
style="color:#8f8f8f;"><< zurück</span></bx:if.else></bx:if>
      | <strong>Seite:</strong> 
    <bx:recorddata.navlink move="-5"/>
    <bx:recorddata.navlist max="5" boundary=" · "/>
    <bx:recorddata.navlink move="5"/>
     |  
    <bx:if><bx:recorddata.nextlist>weiter >></bx:recorddata.nextlist><bx:if.else><span
style="color:#8f8f8f;">weiter >></span></bx:if.else></bx:if>
  </bx:recorddata.nav>
</div>
```

wenn nichts zum Blättern da ist, nicht anzeigen

```
<bx:tabledata.nav object="Aktuelles">
  <bx:iF type="any">
    <div class="blaetternunten">
      <bx:iF.ignore>
        <bx:if><bx:tabledata.previouslist><<
zurück</bx:tabledata.previouslist><bx:if.else><span style="color:#8f8f8f;"><<
zurück</span></bx:if.else></bx:if>
      </bx:iF.ignore>
        | <strong>Seite:</strong> 
      <bx:tabledata.navlink move="-5"/>
      <bx:tabledata.navlist max="5" boundary=" · "/>
      <bx:tabledata.navlink move="5"/>
       |  
      <bx:iF.ignore>
        <bx:if><bx:tabledata.nextlist>weiter >></bx:tabledata.nextlist><bx:if.else><span
style="color:#8f8f8f;">weiter >></span></bx:if.else></bx:if>
      </bx:iF.ignore>
    </div>
  </bx:iF>
</bx:tabledata.nav>
```

zugehöriges CSS

```
.blaetternunten,.blaetternoben{
  background-color:#999999;
  text-align:center;
  padding:5px;
  color:#ffffff;
  }
```

Etwas ausführlicheres Stylen

sieht so aus:

Code

```
<bx:recorddata.nav object="Liste">
  <div id="blaetterbox">
    <div id="blaetterelements">
      <span id="blaettern-left">
       <bx:if><bx:recorddata.previouslist>«
zurück</bx:recorddata.previouslist><bx:if.else><span style="color:#B2B2B2;">«
zurück</span></bx:if.else></bx:if>
      </span>
      <span id="blaettern-middle">
        <bx:recorddata.navlist max="5" boundary="">

<bx:recorddata.navlistcurrent><span><bx:recorddata.navlistpage/></span></bx:recorddata.navlist
current>
          <bx:recorddata.navlistcurrent not><a
```

```
href="?index=<bx:recorddata.navlistindex/>"><bx:recorddata.navlistpage/></a></bx:recorddata.navlistcurrent>
        </bx:recorddata.navlist>
      </span>
      <span id="blaettern-right">
        <bx:if><bx:recorddata.nextlist>weiter »</bx:recorddata.nextlist><bx:if.else><span style="color:#B2B2B2;">weiter »</span></bx:if.else></bx:if>
      </span>
    </div>
  </div>
</bx:recorddata.nav>
```

zugehöriges CSS

```
#blaetterbox {
  width:100%;
  height:28px;
  background-color:#b2b2b2;
  -moz-border-radius: 3px;
  -webkit-border-radius: 3px;
  border-radius: 3px;
  }
#blaetterbox #blaetterelements {
  width:365px; margin:auto;
  font-size:10pt;
  font-weight:bold;
  padding-top:6px;
  text-align:center;
  }
#blaetterbox #blaetterelements #blaettern-middle{#
  padding:0 20px;
  }
#blaetterbox #blaetterelements #blaettern-left a, #blaetterbox #blaetterelements #blaettern-right a{
  text-decoration:none;
  color:#605639;
  filter: dropshadow(color=#949494, offx=-1, offy=-1);
  }
#blaetterbox #blaetterelements #blaettern-left span, #blaetterbox #blaetterelements
```

```
#blaettern-right span{
  text-decoration:none;
  color:#ffffff;
  text-shadow: -1px -1px 0px #949494;
  filter: dropshadow(color=#949494, offx=-1, offy=-1);
  }
#blaetterbox #blaetterelements #blaettern-middle span{
  text-decoration:none;
  letter-spacing:5px;
  color:#ffffff;
  background-color:#968659;
  padding:6px 1px 5px 7px;
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
  }
#blaetterbox #blaetterelements #blaettern-middle a{
  text-decoration:none;
  letter-spacing:5px;
  color:#605639;
  padding:6px 1px 7px 7px;
  }
```

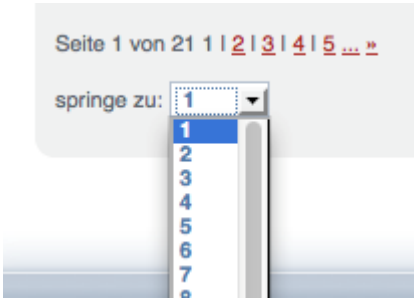
Springe-zu-Funktion

Ergänzung mit:

Code

```
<br><br>springe zu:
<select name="change" size="1" style="width:50px;" onchange="">
  <bx:recorddata.navlist object="MeineSuchergebnisliste" max="0">
    <option value="<bx:recorddata.navlistpath encoding="html"
/>"><bx:recorddata.navlistpage/></option>
  </bx:recorddata.navlist>
</select>
```

sieht dann so aus:



Dieser Quelltext-Schnipsel kann beliebig außerhalb des recorddata.nav-Blocks in das HTML-Template eingebunden werden (auf Wunsch auch innerhalb, kein Ding) und baut eine SELECT-Box in den HTML-Quellcode ein inkl. aller verfügbaren Seitenergebnisse und dem entsprechenden Blätter-Parameter.

Hinweis:
Man muss abschließend nur noch die SELECT-Box via JavaScript animieren etwas zu tun, wenn eine Auswahl getroffen wurde (Parameter versteckt setzen+Form abschicken oder URL-Aufruf ...)

Zusätzlich kann eine kleine Abfrage um die Select-Box gebaut werden, dass diese bspw. nur angezeigt wird, wenn es mehr als 5 Seiten zu blättern gibt. Somit bleibt die Webseite schön clean und wenn es wirklich mehrere Suchseiten gibt die auch nicht mehr über die bisherige Darstellung sinnvoll angezeigt werden, dann ist die Springe-zu-Funktion hilfreich. Bei 2 Seiten ist diese neue Erweiterung nicht so hilfreich wie bei 15 Suchergebnisseiten.

Bootstrap4-Pagination

```
<bx:recorddata.empty object="{listenname}" not>
  <bx:recorddata.nav object="Telefonate" indexparam="{parametername_index}">
    <nav aria-label="Page navigation" class="mt-4">
      <ul class="pagination">
        <li class="page-item"><a class="page-link"
href="./?parametername_index=0<bx:if>&searchvalue=<bx:pagedata.request
name="searchvalue"/></bx:if>" title="zur ersten Seite"><i class="fa fa-fast-backward" aria-
hidden="true"></i> zur Seite 1</a></li>

        <bx:recorddata.previouslist type="if"><li class="page-item"><a class="page-link"
href="./?parametername_index=<bx:Recorddata.previouslist
type="index"/><bx:if>&searchvalue=<bx:pagedata.request name="searchvalue"/></bx:if>"
title="eine Seite zurück"><i class="fa fa-backward" aria-
hidden="true"></i></a></li></bx:recorddata.previouslist>

        <bx:recorddata.navlist max="5">
          <bx:recorddata.navlistcurrent not>
```

```
            <li class="page-item"><a class="page-link"
href="./?parametername_index=<bx:recorddata.navlistindex/><bx:if>&searchvalue=<bx:pagedata.req
uest name="searchvalue"/></bx:if>" title="Zur Seite
<bx:recorddata.navlistpage/>"><bx:recorddata.navlistpage/></a></li>
          </bx:recorddata.navlistcurrent>
          <bx:recorddata.navlistcurrent>
            <li class="page-item active"><a class="page-link"
href="#"><bx:recorddata.navlistpage/> <span class="sr-only">(current)</span></a></li>
          </bx:recorddata.navlistcurrent>
        </bx:recorddata.navlist>

        <bx:recorddata.nextlist type="if"><li class="page-item"><a class="page-link"
href="./?parametername_index=<bx:Recorddata.nextlist
type="index"/><bx:if>&searchvalue=<bx:pagedata.request name="searchvalue"/></bx:if>"
title="eine Seite vor"><i class="fa fa-forward" aria-
hidden="true"></i></a></li></bx:recorddata.nextlist>

        <li class="page-item"><a class="page-link"
href="./?parametername_index=<bx:recorddata.lastlist
type="index"/><bx:if>&searchvalue=<bx:pagedata.request name="searchvalue"/></bx:if>"
title="zur letzten Seite">zur Seite <bx:recorddata.totalpages/> <i class="fa fa-fast-forward"
aria-hidden="true"></i></a></li>
      </ul>
    </nav>
  </bx:recorddata.nav>
</bx:recorddata.empty>
```

CSS-Menü

Beispielcode für ein einfaches Menü

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Batix Testmenü mit CSS</title>
<link rel="stylesheet" href="batix-css-menue.css" type="text/css">
</head>
<body>

<div class="BX_cssmenue">
  <ul>
    <bx:sitemap.0>
     <li><!--[if IE]><!--><a<bx:sitemap.if hasChilds> class="submenue"</bx:sitemap.if>
href="/www/<bx:pagedata.webdir/>/<bx:navigation.path/>/" target="<bx:navigation.target
default="_self"/>" title="<bx:navigation.meta name="meta"/>"><!--<![endif]--
><bx:sitemap.name/>
      <!--[if gt IE 6]><!--></a><!--<![endif]--><!--[if lte IE 6]><table><tr><td><![endif]-->
        <bx:sitemap.if hasChilds><ul></bx:sitemap.if>
          <bx:sitemap.1>
           <li class="<bx:sitemap.if first>firstitem</bx:sitemap.if><bx:sitemap.if
last>lastitem</bx:sitemap.if>"><a<bx:sitemap.if hasChilds> class="submenue"</bx:sitemap.if>
href="/www/<bx:pagedata.webdir/>/<bx:navigation.path/>/" target="<bx:navigation.target
default="_self"/>" title="<bx:navigation.meta name="meta"/>"><bx:sitemap.name/>
            <!--[if gt IE 6]><!--></a><!--<![endif]--><!--[if lte IE
6]><table><tr><td><![endif]-->
              <bx:sitemap.if hasCHilds><ul></bx:sitemap.if>
                <bx:sitemap.2>
                 <li class="<bx:sitemap.if first>firstitem</bx:sitemap.if><bx:sitemap.if
last>lastitem</bx:sitemap.if>"><a<bx:sitemap.if hasChilds> class="submenue"/</bx:sitemap.if>
href="/www/<bx:pagedata.webdir/>/<bx:navigation.path/>/" target="<bx:navigation.target
default="_self"/>" title="<bx:navigation.meta name="meta"/>"><bx:sitemap.name/></a></li>
                </bx:sitemap.2>
              <bx:sitemap.if hasCHilds></ul></bx:sitemap.if>
```

```
            <!--[if lte IE 6]></td></tr></table></a><![endif]-->
           </li>
          </bx:sitemap.1>
        <bx:sitemap.if hasChilds></ul></bx:sitemap.if>
       <!--[if lte IE 6]></td></tr></table></a><![endif]-->
      </li>
   </bx:sitemap.0>
  </ul>
</div>

</body>
</html>
```

Beispiel für die dazugehörige css-Datei

```
 .BX_cssmenue {
  width:600px;
  height:25px;
  border:1px solid #98b1c4;
  background:#bdcedc;
  z-index:100;
  position:relative;
}
.BX_cssmenue ul {
  float:left;
  list-style-type:none;
  position:relative;
  margin:0;
  padding:0;
}
.BX_cssmenue li {
  float:left;
}
.BX_cssmenue li:hover,.BX_cssmenue li a:hover {
  position:relative;
}
.BX_cssmenue a {
  display:block;
  float:left;
  height:25px;
```

```
  white-space:nowrap;
  padding:0 10px 0 10px;
  line-height:25px;
  font-size:12px;
  font-weight:bold;
  font-family:Arial,Helvetica,sans-serif;
  text-align:center;
  color:#293d6b;
  text-decoration:none;
}
.BX_cssmenue li:hover a,.BX_cssmenue a:hover {
  background:#98b1c4;
}
.BX_cssmenue a.submenue {
  background:url(/static/dev/css-menu/top-sub.gif) no-repeat center right;
  padding:0 20px 0 10px;
}
.BX_cssmenue li:hover a.submenue,.BX_cssmenue a.submenue:hover {
  background:#98b1c4 url(/static/dev/css-menu/top-sub.gif) no-repeat center right;
}
.BX_cssmenue ul ul {
  position:absolute;left:-5000px;
  top:-5000px;
  width:202px;
  height:auto;
  border:none;
  background:#bdcedc;
}
.BX_cssmenue table {
  margin-top:-1px;
  border-collapse:collapse;
}
.BX_cssmenue ul :hover ul {
  left:0px;
  top:25px;
}
.BX_cssmenue li li {
  width:200px;
  border:1px solid #98b1c4;
  border-width:0 1px;
```

```
  padding:0
}
.BX_cssmenue li li.firstitem {
  border-top:1px solid #98b1c4;
}
.BX_cssmenue li li.lastitem {
  border-bottom:1px solid #98b1c4;
}
.BX_cssmenue ul ul a,.BX_cssmenue ul :hover ul a,.BX_cssmenue ul :hover ul :hover ul a {
  float:none;
  margin:0;
  width:180px;
  height:auto;
  white-space:normal;
  padding:4px 10px 5px 10px;
  font:bold 12px Arial,Helvetica,sans-serif;
  text-decoration:none;
  text-align:left;
  color:#293d6b;
  background:none;
}
.BX_cssmenue ul ul :hover a,.BX_cssmenue ul ul a:hover,.BX_cssmenue ul ul :hover ul :hover
a,.BX_cssmenue ul ul :hover ul a:hover {
  background:#98b1c4;
}
.BX_cssmenue ul ul a.submenue,.BX_cssmenue ul :hover ul a.submenue {
  width:180px;
  padding:4px 10px 5px 10px;
  background:url(/static/dev/css-menu/sub-sub.gif) no-repeat center right;
}
.BX_cssmenue ul ul :hover a.submenue,.BX_cssmenue ul ul a.submenue:hover  {
  background:#98b1c4 url(/static/dev/css-menu/sub-sub.gif) no-repeat center right;
}
.BX_cssmenue ul :hover ul ul {
  position:absolute;
  left:-5000px;
  top:-5000px;
}
.BX_cssmenue ul :hover ul :hover ul {
  left:190px;
```

```
  top:-1px;
}
.BX_cssmenue br {
  clear:both;
  height:0;
  font-size: 1px;
  line-height: 0px;
 }
```

Link zur Demo-Seite mit diesem Code http://dev.batix.local/www/dev/purecssmenu/

CSV-Import in Container

Die Daten einer zuvor hochgeladene CSV-Datei werden in einen Container gespeichert. Es ist auch möglich, eine ZIP-Datei mit verschiedenen CSV-Dateien hochzuladen.

Datei
Feldname der zuvor hochgeladenen Datei
ID_ZielContainer
Angabe der ID, wenn der Name nicht anderweitig ermittelt werden kann (z.B. durch Dateiname oder 1. Zeile in der Datei)
ZielContainer_aus_Datei
noch nicht benutzt
ZielContainer_aus_Dateiname
wenn angehakt, dann wird als Zielcontainer der Name der Datei genommen.
ZielContainer_in_Zeile
Angabe der Zeile, in der der Name des Zielcontainers steht (im Beispiel unten wäre das die 1)
Feldnamen_in_Zeile
Zeilennummer der Zeile, in der die Feldname stehen (im Beispiel unten wäre das die 2)
Trennzeichen
Trennzeichen, mit der die Dateifelder getrennt sind (z.B. ,)
Daten_ab_Zeile
Nummer der Zeile, ab der die Daten beginnen (im Beispiel unten wäre das die 3)
Feldnamenliste
Wenn in der Importdatei keine Feldnamen enthalten sind, dann können diese hier angegeben werden.
Namens-Umwandlung
Zeilenweise Liste: CSV-Feldname=Feldname im Container oder Dateiname=ZielContainerID

Beispielaufbau einer CSV-Datei:

Kontakt
Name,Vorname,Abteilung,Telefon,Email

Mustermann,Paul,Datensicherheit,03671/223344,test@batix.com

Wurst,Hans,Marketing,090/345678,hawu@dotcom.com

...

Eingeloggt bleiben

Variante 1 - über Sessiongültigkeit

Im Login-Action trägt man im Baustein "Login" im Feld "minimale Sessionlänge" ein, wie lange die Session haltbar ist (in Minuten, 1 Monat = 43200 Minuten).

Danach legt man einen Groovy-Baustein an, wo geprüft wird, ob das Häkchen "eingeloggt bleiben" gesetzt ist. Wenn ja, wird ein Cookie "JSESSIONID" geschrieben, der dann im Browser mit der gleichen Haltbarkeit wie im vorherigen Login-Baustein gespeichert wird (in Sekunden, 1 Monat = 2592000 Sekunden).

```
import javax.servlet.http.Cookie;


if (action.getParameter("remember") == "j") {
  Cookie c = new Cookie("JSESSIONID", session.id)
  c.maxAge = 2592000 // Sekunden
  c.path ="/"
  action.originalResponse.addCookie(c)
}
```

Allerding liegen dann viele offene Sessions auf dem Server rum - ist dann wohl eher für kleinere Projekte interessant.

Formularversand mit Ajax

Dieses Beispiel veranschaulicht die Möglichkeit Formulardaten mittels jQuery per Ajax versenden. Voraussetzung hierfür ist das Einbinden des jQuery-Frameworks und des speziellen Scriptes jquery.form.js

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Beispieldokument</title>

<script type="text/javascript" src="jquery-1.4.1.min.js"></script>
<script type="text/javascript" src="jquery.form.js"></script>
<script type="text/javascript">
  // warten bis die DOM-Struktur geladen ist
  $(document).ready(function() {
    var options = {
      target:        '#div_suchergebnisse'   // Element in das die Serverantwort geschrieben
wird
      //beforeSubmit:  showRequest,  // optionale Funktion vor dem Absenden des Formulares
      //success:       showResponse  // optionale Funktion nach dem Absenden des Formulares
    };
    // Anbindung des betreffenden Formulares, dass per Ajax versendet werden soll (mehrere
möglich)
    $('#testform').ajaxForm(options)
  });
</script>

</head>
<body>

  <form name="testform" id="testform" method="post" action="zieladresse.htm">
    <input type="hidden" name="field1"  id="field1" value="">
    <input type="text" name="field2"  id="field2" value=""><br>
    <input type="text" name="field3"  id="field3" value="">
    <input type="submit" value="absenden">
```

```
  </form>

  <div id="div_suchergebnisse">
    Hier landet das Ergebnis der Suche
  </div>

</body>
</html>
```

Es sind noch wesentlich mehr Optionen für diese Funktion verfügbar.

Mehr Informationen und Script-Download unter http://jquery.malsup.com/form/#getting-started

Google-Sitemap Datei erzeugen

Diese Beispiel realisiert folgende Funktionen:

- eine dynamische Sitemap im XML-Format generieren, die Suchmaschinen beim Indizieren hilft

Folgende Tags wurde verwendet:

- Bx:pagedata
- Bx:sitemap
- Bx:tools

Quellcode

Dieser Code wird als Designtemplate gespeichert und im Startseiten-Menüpunkt z.B. als sitemap.xml eingebunden. Es ist wichtig, dass diese Datei im obersten Verzeichnis des Projektes liegt, da immer nur URLs unterhalb des Pfades dieser Datei indiziert werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
<bx:sitemap.1>
<url>
<loc>http://<bx:pagedata.hostname/>/www/<bx:pagedata.webdir/>/<bx:sitemap.path/></loc>
<lastmod><bx:tools.lastmodified pattern="yyyy-MM-dd" />T<bx:tools.lastmodified
pattern="hh:mm:ss" />+00:00</lastmod>
<changefreq>weekly</changefreq>
<priority>1</priority>
</url>
<bx:sitemap.2>
<url>
<loc>http://<bx:pagedata.hostname/>/www/<bx:pagedata.webdir/>/<bx:sitemap.path/></loc>
<lastmod><bx:tools.lastmodified pattern="yyyy-MM-dd" />T<bx:tools.lastmodified
pattern="hh:mm:ss" />+00:00</lastmod>
<changefreq>weekly</changefreq>
<priority>0.75</priority>
</url>
<bx:sitemap.3>
```

```
<url>
<loc>http://<bx:pagedata.hostname/>/www/<bx:pagedata.webdir/>/<bx:sitemap.path/>/</loc>
<lastmod><bx:tools.lastmodified pattern="yyyy-MM-dd" />T<bx:tools.lastmodified
pattern="hh:mm:ss" />+00:00</lastmod>
<changefreq>weekly</changefreq>
<priority>0.5</priority>
</url>
<bx:sitemap.4>
<url>
<loc>http://<bx:pagedata.hostname/>/www/<bx:pagedata.webdir/>/<bx:sitemap.path/>/</loc>
<lastmod><bx:tools.datum pattern="yyyy-MM-dd" day="-3"/></lastmod>
<changefreq>weekly</changefreq>
<priority>0.25</priority>
</url>
<bx:sitemap.5>
<url>
<loc>http://<bx:pagedata.hostname/>/www/<bx:pagedata.webdir/>/<bx:sitemap.path/>/</loc>
<lastmod><bx:tools.datum pattern="yyyy-MM-dd" day="-3"/></lastmod>
<changefreq>weekly</changefreq>
<priority>0.20</priority>
</url>
</bx:sitemap.5>
</bx:sitemap.4>
</bx:sitemap.3>
</bx:sitemap.2>
</bx:sitemap.1>
</urlset>
```

Es werden alle Menüpunkte bis zu einer Tiefe von 5 Ordnern ausgegeben. `priority` ist lediglich eine relative, projekt-interne Angabe und wird von Suchmaschinen genutzt, um bestimmte Links bei einer Auswahl von mehreren Links im Projekt zu priorisieren, diese Angabe hat keine Auswirkung auf das Ranking in der Suchmaschine.

Bei `lastmod` wurden 2 verschiedene Implementierungen genutzt. Mittels bx:tools.lastmodified wird allerdings nur das Bearbeitungsdatum des Hauptmenüpunktes zurückgegeben (nicht das der einzelnen Menüpunkte), bx:tools.datum dagegen liefert immer "Heute - 3 Tage". Welche Möglichkeit genutzt wird, kann nicht verallgemeinert werden (optimal wäre das Bearbeitungsdatum des jeweiligen Menüpunktes).

HTML-Mails für MS-Outlook

E-Mails im HTML-Format benötigen ein sehr spezielles Design, um in Outlook korrekt angezeigt zu werden. Es gibt eine ganze Reihe HTML- und CSS-Elemente, die nicht erlaubt sind (z.B. Div's, Hintergrundbilder, Positionierungen etc.).

Eine komplette Liste aller erlaubten/nicht erlaubten Elemente und Parameter finden Sie unter http://msdn.microsoft.com/en-us/library/aa338201.aspx.

Die große Kunst beim HTML-Mail-Design besteht darin, dass die Mails nicht nur in allen Browsern, sondern auch in allen E-Mail-Clients aussehen müssen. Während fast alle Mail-Clients und Webmailer keine Probleme mit der Darstellung haben, kommt Outlook mit modernem HTML-Design überhaupt nicht zurecht und verlangt nach Quellcode, wie er in den frühen 90er Jahren zu finden war. Mail-Designs müssen für Outlook wieder umständlich mit Tabellen, zerschnippelten Bildern und Platzhalter-GIF's erzeugt werden.

Wer hierfür Beispiele braucht, sollte die Newsletter von ALTEC Solartechnik oder Waldbad Brunn abonnieren.

Kalender bauen

Benutzt werden die beiden Tags `bx:calendarloop` und `bx:calenderfield`

```
<table class="calendar1">
    <tr class="head">
      <td colspan="7">
        <div class="month">
          <span><a href="./?<bx:calendarfield.month link="prev" object="Kalender"/>"
title="einen Monat zurück">&laquo;</a></span>
          <span><bx:calendarfield.month object="Kalender"/></span>
          <span><a href="./?<bx:calendarfield.month link="next" object="Kalender"/>"
title="einen Monat vor" >&raquo;</a></span>
        </div>
        <div class="year">
          <span><a href="./?<bx:calendarfield.year link="prev" object="Kalender"/>" title="ein
Jahr zurück">&laquo;</a></span>
          <span><bx:calendarfield.year object="Kalender"/></span>
          <span><a href="./?<bx:calendarfield.year link="next" object="Kalender"/>" title="ein
Jahr vor">&raquo;</a></span>
        </div>
        <div style="clear:both;"></div>
      </td>
    </tr>
    <tr class="daynames">
      <td>Mo</td>
      <td>Di</td>
      <td>Mi</td>
      <td>Do</td>
      <td>Fr</td>
      <td class="we">Sa</td>
      <td class="we">So</td>
    </tr>
  <bx:calendarloop.Kalender activedayrequest="filter" pool="12549B83FF8"
datestart="Startdatum" dateend="Endedatum">
    <bx:calendarfield.if firstweekday><tr class="days"></bx:calendarfield.if>
      <td class="
      <bx:calendarfield.if currentmonth="false">othermonth</bx:calendarfield.if>
```

```
      <bx:calendarfield.if weekend>we</bx:calendarfield.if>
      <bx:calendarfield.if mark>marked</bx:calendarfield.if>
      <bx:calendarfield.if today>today</bx:calendarfield.if>">
        <a href="./?filter=<bx:calendarfield.day pattern="dd."/><bx:calendarfield.month
pattern="MM."/><bx:calendarfield.year pattern="yyyy"/>"><bx:calendarfield.day
pattern="d"/></a>
      </td>
    <bx:calendarfield.if lastweekday></tr></bx:calendarfield.if>
  </bx:calendarloop.Kalender>
  <tr class="footer">
    <td colspan="7">
      <a href="">Heute</a>
    </td>
  </tr>
</table>
```

zugehöriges CSS

```
table.calendar1 a{color:#555555;}
table.calendar1 td{width:20px;height:20px;text-align:center;}
table.calendar1 .head td{background-color:#FFB689;}
table.calendar1 .head .month{float:left;padding-left:5px;}
table.calendar1 .head .year{float:right;padding-right:5px;}
table.calendar1 .daynames td{background-color:#aaaaaa;}
table.calendar1 .daynames .we{color:#962420;font-weight:bold;}
table.calendar1 .days td{background-color:#cccccc;}
table.calendar1 .days td.marked{background-color:#FBD9CA;}
table.calendar1 .days td.today{outline:1px solid red;}
table.calendar1 .days td.active{outline:1px solid green;}
table.calendar1 .days td.othermonth{background-color:#eeeeee;}
table.calendar1 .days td.othermonth a{color:#ffffff !important;}
table.calendar1 .days td.we a{color:#962420;font-weight:bold;}
table.calendar1 .footer td{background-color:#FFB689;}
```

Locale bei Date-Pattern hinzufügen

1. Projekt auswählen
2. bei **Dokumentvorlagen** rechts unter Tools auf **Quelltext ersetzen** klicken
3. in das erste Textfeld (**Suchbegriff**) eintragen:
   `pattern="([^"]*(?:MMM|E)[^"]*)"(?! locale)`
4. in das zweite Textfeld (**Ersetzen durch**) eintragen:
   `pattern="$1" locale="de"`
5. Haken **regulärer Ausdruck** setzen
6. auf **Vorschau** klicken und prüfen, ob alles passt
7. auf **jetzt ersetzen** klicken
8. ggf. Schritte 6 und 7 wiederholen, falls viele Quelltexte betroffen sind

Manipulationsgeschützte Formulare mit Secureform

Vom System generierte Werte in Formularen sind leicht veränderbar. Falls in den Aktionen keine Überprüfungen dieser Werte vorhanden sind, entstehen schnell Sicherheitslecks. Parameter können dann beliebig manipuliert werden und die Aktionsbausteine verrichten unbedarft ihren Dienst mit gefälschten Daten. So können z.B. IDs ausgetauscht werden und somit fremde Datensätze überschrieben werden.

Damit nicht in jeder Aktion selbst eine Überprüfung in JSP oder Groovy geschrieben werden muss, gibt es das Tag `bx:secureform` und die Aktion Sicheres Formular überprüfen (SecureformAction). Im Zusammenspiel stellen diese beiden Komponenten sicher, dass ausgewählte Parameter nicht manipuliert werden können.

Dieser Artikel bietet eine detaillierte Erklärung der Funktion und Vorgehensweise. Es sollten zusätzlich die Seiten des Tags und der Aktion durchgelesen werden.

Parameter sammeln

Im Formular

Im ersten Schritt werden die zu sichernden Formularelemente gesammelt. Das sind jene Parameter, die später an das Action geschickt werden. Es müssen nicht alle Parameter abgesichert werden, nur jene die im Action als zu sichern angegeben wurden.

Zunächst wird um das gesamte Formular ein `<bx:secureform>` Tag gesetzt. Dies definiert einen Bereich, in dem Parameter gesammelt werden. Dadurch können auf einer Seite mehrere solche Bereiche definiert werden, falls es mehrere Formulare gibt.

Bereich definieren

```
<bx:secureform>
  <form action="...">
  </form>
</bx:secureform>
```

Danach werden die zu schützenden Parameterwerte gesammelt. In den meisten Fällen wird der `value` eines `<input type="hidden">` einfach mit `<bx:secureform.field>` umschlossen.

Parameter sammeln

```
<bx:secureform>
  <form action="...">
    <input type="hidden" name="Besitzer" value="<bx:secureform.field
name="Besitzer"><bx:userdata.id /></bx:secureform.field>">
  </form>
</bx:secureform>
```

Falls ein Parameter im Formular geschützt wird, muss er auf jeden Fall auch in den Aktionseinstellungen angegeben werden (s.u.) - da die Prüfung im Baustein sonst fehlschlägt.

Parameter, die nicht mitgesendet werden, falls sie keinen Wert haben, dürfen nicht mit zur Hashbildung herangezogen werden. Das sind z.B. Datensatz-ID Felder, die nur bei fehlerhaften Validierungen gefüllt werden, damit beim erneuten Absenden des Formulars kein neuer Datensatz erzeugt wird. Meist kommt dabei `<bx:containerfilter dummy>` oder `<bx:record dummy>` zum Einsatz. Der Parameter `hidden-when-empty` bewirkt, dass ein Feld mit leerem Wert von `<bx:secureform.field>` ignoriert wird.

hidden-when-empty

```
 <bx:secureform>
  <form action="...">
    <bx:if><input type="hidden" name="kid" value="<bx:secureform.field name="kid" hidden-when-
empty><bx:recorddata.id /></bx:secureform.field>"></bx:if>
    <input type="hidden" name="Besitzer" value="<bx:secureform.field
name="Besitzer"><bx:userdata.id /></bx:secureform.field>">
  </form>
</bx:secureform>
```

Zum Schluss wird der Hash generiert und in das Formular geschrieben. Die einfachste Methode ist vom System direkt ein `<input type="hidden">` generieren zu lassen.

Hash ausgeben

```
<bx:secureform>
  <form action="...">
    <bx:if><input type="hidden" name="kid" value="<bx:secureform.field name="kid" hidden-when-
empty><bx:recorddata.id /></bx:secureform.field>"></bx:if>
    <input type="hidden" name="Besitzer" value="<bx:secureform.field
name="Besitzer"><bx:userdata.id /></bx:secureform.field>">
    <bx:secureform.hash />
```

```
  </form>
</bx:secureform>
```

Dies erzeugt ein verstecktes Formularfeld mit passendem Name und Wert. Alternativ kann der Name oder Wert auch einzeln ausgegeben werden, siehe dazu die Tag Doku.

Im Link

Das Tag kann auch benutzt werden, um einen Link / Button abzusichern:

```
<bx:secureform>
  <a href="delete.act?kid=<bx:secureform.field name="kid"><bx:recorddata.id
/></bx:secureform.field>&<bx:secureform.hash name />=<bx:secureform.hash value />">löschen</a>
</bx:secureform>
```

Hier werden die Parameter `name` und `value` von `bx:securefom.hash` benutzt, denn beide Werte sind dynamisch.

Falls URL-Parameter mit z.B. `bx:if` umschlossen sind, muss auch wieder `hidden-when-empty` benutzt werden (s.o.).

Erlaubte Werte festlegen

Falls ein Parameter mehrere, bestimmte Werte annehmen darf, ist dies mit `bx:secureform.values` abbildbar. In diesem Fall wird nicht das Feld mit dem Wert gehasht (da es vom Nutzer ja geändert werden kann), sondern ein zweites Feld eingefügt. Es enthält die erlaubten Werte und wird anstelle des veränderlichen Feldes mitgehasht.

So kann sich der Inhalt des Feldes (z.B. ein `<select>`) ändern, es wird aber trotzdem eine abgesicherte Information über die erlaubten Werte mitgeschickt.

Der Parametername wird normal in die Aktion eingetragen, diese unterscheidet aber, ob ein Info-Parameter mit den erlaubten Werten übergeben wurde oder nur der Normale und zieht dann den entsprechenden Parameter zur Hashbildung heran.

Im Formular

mehrere erlaubte Werte (Formular)

```
<select name="Anrede">
  <option value="<bx:secureform.values name="Anrede">m</bx:secureform.values>">Herr</option>
```

```
  <option value="<bx:secureform.values name="Anrede">w</bx:secureform.values>">Frau</option>
</select>
<bx:secureform.values name="Anrede" />
```

Das Tag in der geschlossenen Variante erzeugt automatisch ein `<input type="hidden">` mit passenden `name` und `value` Attributen.

Im Link

mehrere erlaubte Werte (Link)

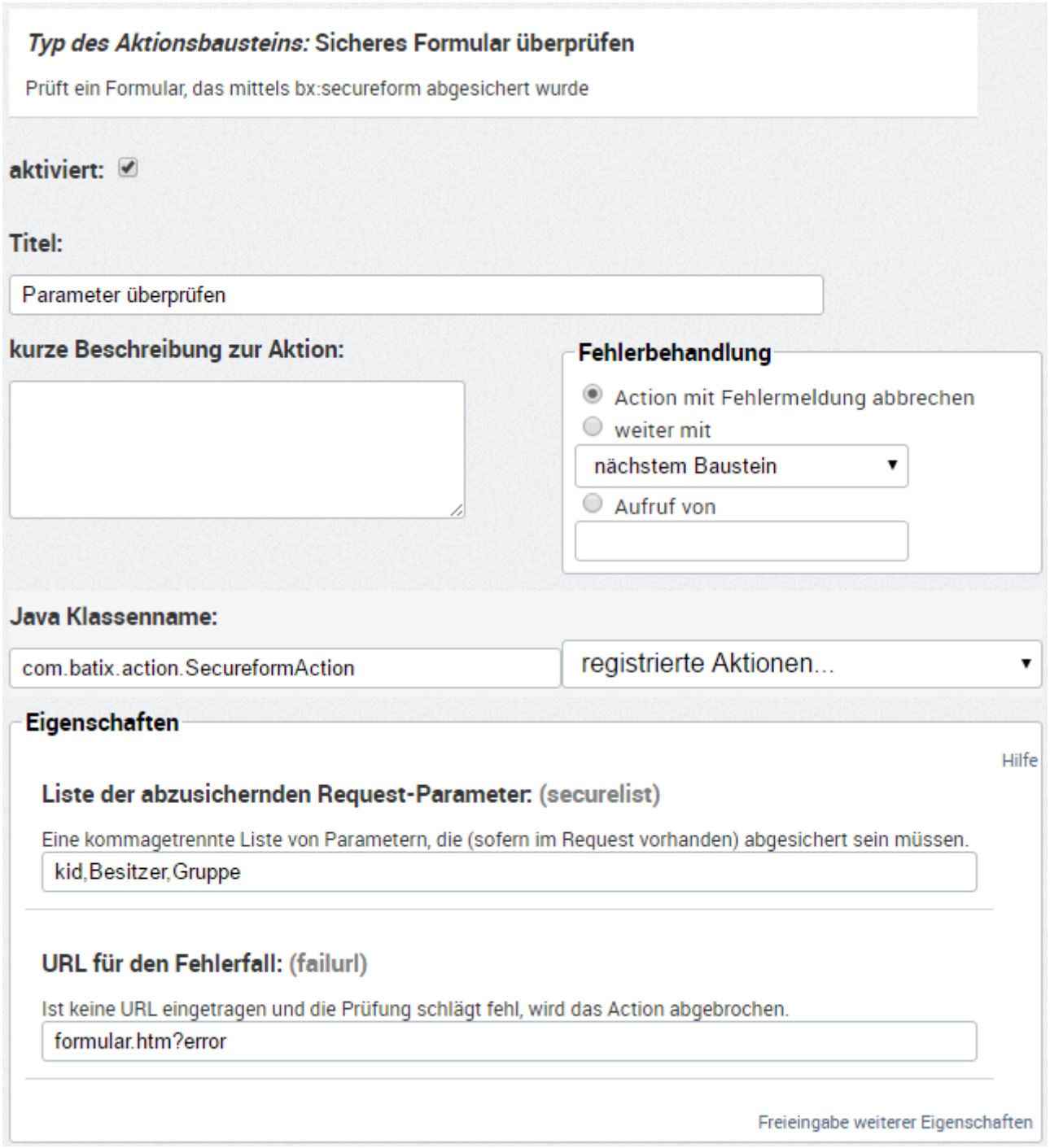
```
var url = "save.act?";
// ...andere (ggf. abgesicherte) Parameter, wie z.B. ID...
url += "&Anrede=" + encodeURIComponent($("[name=Anrede]").val());
url += "&<bx:secureform.values name="Anrede" param encode="javascript"
/>=<bx:secureform.values name="Anrede" values encode="javascript" />"
```

Aktion absichern

Um eine Aktion abzusichern, wird ein neuer Sicheres Formular überprüfen (SecureformAction) Baustein eingefügt. **Dieser muss an erster Stelle stehen**, damit bei Manipulation die Aktion direkt abgebrochen wird und nachgelagerte Bausteine nicht ausgeführt werden.

In der *Liste der abzusichernden Request-Parameter* werden all die Parameter aufgezählt, die (falls übergeben) abgesichert sein müssen - d.h. diese Parameter werden zur Hashbildung im Baustein herangezogen. Das bedingt natürlich, dass diese Parameter auch im Formular eingesammelt wurden. Umgedreht gilt dasselbe: Parameter, die im Formular gesammelt wurden, müssen auch im Aktionsbaustein gelistet sein.

Im unteren Feld kann eine URL für den Fehlerfall spezifiziert werden. Hierhin wird weitergeleitet, sobald die Prüfung fehlschlägt. Wurde keine URL angegeben und die Prüfung schlägt fehl, so wird die gesamte Aktion mit einer Fehlermeldung abgebrochen.



Steht ein Parameter in der Liste, wurde aber nicht übergeben, so wird er vom Baustein nicht zu Hashbildung herangezogen.

> Die Liste kann auch für Parameter verwendet werden, die der Aktion niemals übergeben werden dürfen (falls z.B. andere Bausteine schlecht darauf reagieren).
>
> Da diese nicht im Formular gesammelt werden und somit nicht in den Hash einfließen, wird vom Formular ein anderer Hash erzeugt, als von der Aktion (da diese den Parameter mit in die Berechnung integriert, da er in der Liste steht).

Seite absichern

Analog zum Absichern einer Aktion existiert die Möglichkeit ein Template abzusichern. Dies kann z.B. verwendet werden, um bestimmte Parameter einer Listen- oder Detail-Seite abzusichern ("Kontakttyp darf nur Kunde sein" oder "Gruppe muss 123ABC oder 456DEF sein").

Hierfür gibt es das Tag `<bx:secureform.check>`, welches ganz oben in der Seite eingebaut wird und die Abarbeitung der Seite abbricht, sobald etwas manipuliert wurde. Optional kann auch eine URL angegeben werden, auf die weitergeleitet wird.

einfache Prüfung in Template

```
<bx:secureform.check securelist="Kategorie" />
<!DOCTYPE html>
<html>
...
```

Funktionsweise

Das Verfahren beruht auf kryptografischen Hashes. Dies sind Einwegfunktionen, die einen beliebigen Input in einen Output bestimmter Länge transformieren. Diese Berechnung ist nicht umkehrbar und somit ist der Input nicht rekonstruierbar.

Es fließen verschiedene Sachen in die Berechnung des Hashes ein. Allein die Parameternamen und -werte genügen nicht, da sonst der Hash von einem Angreifer selbst gebildet werden könnte. Der Algorithmus ist im Allgemeinen bekannt oder kann rekonstruiert werden.

Daher werden zusätzlich serverseitig generierte bzw. gespeicherte Werte für die Berechnung des Hashes benutzt. Diese Werte werden dem Client nicht mitgeteilt und er kann diese auch nicht auslesen. Nur das Tag und das Action kennen diese und können damit den korrekten Hash bilden.

Alle Parameter, die im Formular zum Hashen gesammelt und im Request mitgeschickt wurden, müssen auch in den Aktioneinstellungen angegeben werden. Umgekehrt müssen aber nicht alle Parameter, die im Aktionsbaustein stehen, auch im Request vorhanden sein. Falls sie vorhanden sind, müssen sie allerdings auch abgesichert sein. Das Formular und die Aktion müssen im Grunde denselben Satz an Parametern zur Hashbildung benutzen, sonst sind die Ergebnisse verschieden.

Menü mit CSS

Ein CSS-basiertes Menü, das zur Not auch ohne JavaScript funktioniert und beliebig viele Ebenen darstellen kann.

Zum Betrieb des Menüs sind nur zwei Textbausteine in den HTML-Quelltext zu integrieren.

Textbaustein 1 für den HTML-Body

```
<!-- ################### Beginn CSS-Menü ################### -->

<ul id="qm0" class="qmmc">
  <bx:sitemap.0>
    <li><a class="<bx:sitemap.if hasChilds>qmparent</bx:sitemap.if><bx:sitemap.IF open>
qmactive</bx:sitemap.IF>"
href="/www/<bx:pagedata.webdir/>/<bx:navigation.path/>/"><bx:sitemap.name/></a>
    <bx:sitemap.iF hasChilds>
      <ul>
        <bx:sitemap.1>
          <li><a<bx:sitemap.if hasChilds> class="qmparent<bx:sitemap.IF open>
qmactive</bx:sitemap.IF>"</bx:sitemap.if>
href="/www/<bx:pagedata.webdir/>/<bx:navigation.path/>/"><bx:sitemap.name/></a>
            <bx:sitemap.IF hasChilds>
              <ul style="width:auto;">
                <bx:sitemap.2>
                  <li><a<bx:sitemap.if hasChilds> class="qmparent"</bx:sitemap.if>
href="/www/<bx:pagedata.webdir/>/<bx:navigation.path/>/"><bx:sitemap.name/></a>
                    <bx:sitemap.if hasChilds>
                      <ul>
                        <bx:sitemap.3>
                          <li><a
href="/www/<bx:pagedata.webdir/>/<bx:navigation.path/>/"><bx:sitemap.name/></a></li>
                        </bx:sitemap.3>
                      </ul>
                    </bx:sitemap.if>
                  </li>
                </bx:sitemap.2>
              </ul>
            </bx:sitemap.IF>
```

```
          </li>
        </bx:sitemap.1>
      </ul>
    </bx:sitemap.iF>
    </li>
  </bx:sitemap.0>
  <li class="qmclear"> </li>
</ul>

<!-- ##################### Ende CSS-Menü ##################### -->

<!-- Create Menu Settings: (Menu ID, Is Vertical, Show Timer, Hide Timer, On Click ('all' or
'lev2'), Right to Left, Horizontal Subs, Flush Left, Flush Top) -->
<script
type="text/javascript">qm_create(0,false,100,500,false,false,false,false,false);</script>
```

Textbaustein 2 für den HTML-Head

```
<!-- ################### Beginn Styles für CSS-Menü ################### -->
<style type="text/css">



/*!!!!!!!!!!! Diese Styles nicht ändern !!!!!!!!!!!!!*/
.qmmc .qmdivider{display:block;font-size:1px;border-width:0px;border-
style:solid;position:relative;z-index:1;}.qmmc .qmdividery{float:left;width:0px;}.qmmc
.qmtitle{display:block;cursor:default;white-space:nowrap;position:relative;z-index:1;}.qmclear
{font-size:1px;height:0px;width:0px;clear:left;line-height:0px;display:block;float:none
!important;}.qmmc {position:relative;zoom:1;z-index:10;}.qmmc a, .qmmc li
{float:left;display:block;white-space:nowrap;position:relative;z-index:1;}.qmmc div a, .qmmc
ul a, .qmmc ul li {float:none;}.qmsh div a {float:left;}.qmmc
div{visibility:hidden;position:absolute;}.qmmc li {z-index:auto;}.qmmc ul {left:-
10000px;position:absolute;z-index:10;}.qmmc, .qmmc ul {list-
style:none;padding:0px;margin:0px;}.qmmc li a {float:none}.qmmc li:hover>ul{left:auto;}#qm0 ul
{top:100%;}#qm0 ul li:hover>ul{top:0px;left:100%;}



/*!!!!!!!!! Diese Styles zur individuellen Anpassung verändern !!!!!!!!!*/

  /*""""""""" (MAIN) Container""""""""""""*/
  #qm0
```

```
{
  padding:0px;
  margin:20px 0 0 15px;
}

/*""""""""" (MAIN) Items""""""""""""*/
#qm0 a
{
  padding:2px 40px 2px 10px;
  background-color:#A20002;
  color:#FFFFFF;
  font-family:Arial;
  font-size:12px;
  text-decoration:none;
  font-weight:bold;
  text-align:left;
  border-width:1px 0px 1px 2px;
  border-style:solid;
  border-color:#7C0000;
  -moz-border-top-left-radius:7px;
  border-top-left-radius:7px;
  -moz-border-top-right-radius:7px;
  border-top-right-radius:7px;
}

/*""""""""" (MAIN) Hover State""""""""""""*/
#qm0 a:hover
{
  background-color:#7C0000;
  color:#FFFC85;
}

/*""""""""" (MAIN) Active State""""""""""""*/
body #qm0 .qmactive, body #qm0 .qmactive:hover
{
  background-color:#7C0000;
  color:#FFFC85;
}

/*""""""""" (MAIN) Persistent State""""""""""""*/
```

```
body #qm0 .qmpersistent, body #qm0 .qmpersistent:hover
{
  background-color:#7C0000;
}

/*""""""""" (SUB) Container""""""""""""*/
#qm0 div, #qm0 ul
{
  background-color:#A20002;
  border-width:1px;
  border-style:solid;
  border-color:#7C0000;
}

/*""""""""" (SUB) Items""""""""""""*/
#qm0 div a, #qm0 ul a
{
  padding:4px 20px 4px 15px;
  color:#FFFFFF;
  font-size:12px;
  font-weight:normal;
  text-align:left;
  border-width:0px 0px 1px;
  border-style:solid;
  border-color:#7C0000;
}

/*""""""""" (SUB) Hover State""""""""""""*/
#qm0 div a:hover, #qm0 ul a:hover
{
  color:#FFFC85;
  border-color:#7C0000;
}

/*""""""""" (SUB) Active State""""""""""""*/
body #qm0 div .qmactive, body #qm0 div .qmactive:hover
{
  color:#FFFC85;
}
```

```
  /*""""""""" (SUB) Persistent State""""""""""""*/
  body #qm0 div .qmpersistent, body #qm0 div .qmpersistent:hover
  {
    background-color:#7C0000;
  }


</style>

<!-- ###################### Ende Styles für CSS-Menü ##################### -->

<!-- ###################### Beginn Scripts für CSS-Menü ################### -->

<!-- Core MyCSSMenu Code -->
<script type="text/javascript">/* <![CDATA[ */var
qm_si,qm_li,qm_lo,qm_tt,qm_th,qm_ts,qm_la,qm_ic,qm_ib;var qp="parentNode";var
qc="className";var qm_t=navigator.userAgent;var qm_o=qm_t.indexOf("Opera")+1;var
qm_s=qm_t.indexOf("afari")+1;var qm_s2=qm_s&&qm_t.indexOf("ersion/2")+1;var
qm_s3=qm_s&&qm_t.indexOf("ersion/3")+1;var qm_n=qm_t.indexOf("Netscape")+1;var
qm_v=parseFloat(navigator.vendorSub);;function qm_create(sd,v,ts,th,oc,rl,sh,fl,ft,aux,l){var
w="onmouseover";var ww=w;var
e="onclick";if(oc){if(oc=="all"||(oc=="lev2"&&l>=2)){w=e;ts=0;}if(oc=="all"||oc=="main"){ww=e;
th=0;}}if(!l){l=1;qm_th=th;sd=document.getElementById("qm"+sd);if(window.qm_pure)sd=qm_pure(sd
);sd[w]=function(e){qm_kille(e)};document[ww]=qm_bo;if(oc=="main"){qm_ib=true;sd[e]=function(e
vent){qm_ic=true;qm_oo(new
Object(),qm_la,1);qm_kille(event)};document.onmouseover=function(){qm_la=null;clearTimeout(qm_
tt);qm_tt=null;};}sd.style.zoom=1;if(sh)x2("qmsh",sd,1);if(!v)sd.ch=1;}else
if(sh)sd.ch=1;if(oc)sd.oc=oc;if(sh)sd.sh=1;if(fl)sd.fl=1;if(ft)sd.ft=1;if(rl)sd.rl=1;sd.style.
zIndex=l+""+1;var lsp;var sp=sd.childNodes;for(var i=0;i<sp.length;i++){var
b=sp[i];if(b.tagName=="A"){lsp=b;b[w]=qm_oo;if(w==e)b.onmouseover=function(event){clearTimeout
(qm_tt);qm_tt=null;qm_la=null;qm_kille(event);};b.qmts=ts;if(l==1&&v){b.style.styleFloat="none
";b.style.cssFloat="none";}}else
if(b.tagName=="DIV"){if(window.showHelp&&!window.XMLHttpRequest)sp[i].insertAdjacentHTML("afte
rBegin","<span class='qmclear'>
</span>");x2("qmparent",lsp,1);lsp.cdiv=b;b.idiv=lsp;if(qm_n&&qm_v<8&&!b.style.width)b.style.w
idth=b.offsetWidth+"px";new qm_create(b,null,ts,th,oc,rl,sh,fl,ft,aux,l+1);}}};/* ]]>
*/</script>

<!-- Add-On Core Code (Remove when not using any add-on's) -->
<style type="text/css">.qmfv{visibility:visible !important;}.qmfh{visibility:hidden
```

```
!important;}</style>
<script type="text/JavaScript">var qmad = new Object();qmad.bvis="";qmad.bhide="";</script>


  <!-- Add-On Settings -->
  <script type="text/JavaScript">

    /*******  Menu 0 Add-On Settings *******/
    var a = qmad.qm0 = new Object();

    // Sub Menu Fade Animation Add On
    a.fade_in_frames = 10;
    a.fade_out_frames = 20;

    // Item Bullets (CSS - Imageless) Add On
    a.ibcss_apply_to = "parent";
    a.ibcss_main_type = "arrow";
    a.ibcss_main_direction = "down";
    a.ibcss_main_size = 5;
    a.ibcss_main_position_x = -16;
    a.ibcss_main_position_y = -5;
    a.ibcss_main_align_x = "right";
    a.ibcss_main_align_y = "middle";
    a.ibcss_sub_type = "arrow";
    a.ibcss_sub_direction = "right";
    a.ibcss_sub_size = 5;
    a.ibcss_sub_position_x = -22;
    a.ibcss_sub_align_x = "right";
    a.ibcss_sub_align_y = "middle";

    // Drop Shadow Add On
    a.shadow_offset = 3;
    a.shadow_color = "#AFAFAF";
    a.shadow_opacity = 1;

    // Keyboard Access Add On
    a.keyboard_access_active = true;

  </script>
```

Passwort vergessen

Nach Drücken von "Paßwort vergessen" wird auf eine Seite geleitet (pwvergessen.htm), wo man die E-Mail, mit der man angemeldet ist, eingeben muß. Nach Prüfung wird ein DS im Container "Token" angelegt und eine E-Mail mit einem Link (Token und Email im Request) an die angegebene Adresse geschickt. Nach Klicken des Links öffnet sich eine Seite mit Paßwort-Eingabefeldern. Nach Prüfung der Token-Daten wird das neue PW in der Benutzerverwaltung gespeichert.

Containeraufbau:

Token - einfaches Textfeld
Erstelldatum - Datumsfeld
Email - einfaches Textfeld
BC - BC-Verknüpfung
Kontakt - Einzelverknüpfung mit Kontakt-Container

Actions anlegen:

pwvergessen.act

- Filteraction: prüft, ob's die angegebenen Email im Kontakt-Container gibt und ruft *savetoken.act* auf
- Weiterleitung auf Info-Seite

Beispiele

1.) Actionbaustein "Container-Daten Filteraction" - Prüfen, ob's Email im "Kontakte" gibt

list: ID des Kontakte-Containers

xml:

<filter>
<filter-object required="true">
<field>Kontakt_Email</field>
<type>4</type>
<request-value>email</request-value>
</filter-object>

<limit max="1"/>
</filter>

url:

savetoken.act?Token=<bx:tools.uuid/>&Kontakt=<bx:recorddata.id/>&BCVerknuepfung=<bx:recordfield.Kontakt_BC><bx:recorddata.id/></bx:recordfield.Kontakt_BC>&Erstelldatum=<bx:tools.urlencode><bx:tools.datum pattern="dd.MM.yyyy HH:mm"/></bx:tools.urlencode>&Email=<bx:recordfield.Kontakt_Email/>

failurl:

pwvergessen.htm?fehler=1&email=`[[`email`]]`

2.) Actionbaustein "Weiterleitung" - Weiterleitung

destination: mailverschickt.htm

savetoken.act

- Speicheraction: erzeugt neuen Token-DS
- CMS-Seite verschicken: mit Link (mit Token und Email)

Beispiele

1.) Actionbaustein "Container-Daten speichern" - Neuen Token-DS anlegen

listid: ID des Token-Containers

recordidfield: tokenid

active: ja

2.) Actionbaustein "CMS-Seite versenden" - Email mit Token versenden

alle vorherigen Felder ensprechend ausfüllen

file: pwemail.htm?tokid=`[[`tokenid`]]`

checkpw.act

- prüft, ob Paßwörter da und gleich sind
- Filteraction: sucht Token anhand von Parametern und Datum raus (prüfen, ob noch gültig), ruft *changepw.act* auf

Beispiele

1.) Aktionsbaustein "vorhandene Bedingung abfragen" - Paßwörter da?

requestField: Passwort

compareValue: #nodata

conditionTrue: changepw.htm?fehler=1&token=`[[`token`]]`&email=`[[`email`]]`

2.) Aktionsbaustein "vorhandene Bedingung abfragen" - Paßwörter gleich?

requestField: Passwort

compareValue: `[[`Passwortwdhlg`]]`

conditionFalse: changepw.htm?fehler=2&token=`[[`token`]]`&email=`[[`email`]]`

3.) Aktionsbaustein "Container-Daten Filteraction" - prüfen, ob Token-DS gültig ist

list: ID des Token-Containers

xml:

```
<filter>
  <filter-object required="true">
    <field>Token</field>
    <type>4</type>
    <request-value>token</request-value>
  </filter-object>
  <filter-object required="true">
    <field>Email</field>
    <type>4</type>
    <request-value>email</request-value>
  </filter-object>
  <filter-object>
    <field>Erstelldatum</field>
    <type>4</type>
    <special-value days="-1">TODAY</special-value>
  </filter-object>
```

```
</filter>
```

url: changepw.act?user=<bx:recordfield.BCVerknuepfung type="id"/>&Token=&tokid=<bx:recorddata.id/>&password=`[[`Passwort`]]`

failurl: changepw.htm?fehler=3

4.) Aktionsbaustein "Weiterleitung" - Weiterleitung

destination: ok.htm

changepw.act

- PW ändern: ändert in der Benutzerverwaltung das Paßwort
- speichern: löscht den Token im Token-DS, DS selbst bleibt aber erhalten
- Weiterleitung an Info-Seite (Benutzer muß sich nun mit neuem Paßwort einloggen)

Beispiele

1.) Actionbaustein "Intranet-User anlegen/ändern" - geändertes PW speichern

Parametername des Benutzer-ID-Feldes: user

Parametername des Passwortfeldes: password

2.) Actionbaustein "Container-Datensatz speichern" - Container-Daten speichern

listid: ID des Token-Containers

recordidfield: tokid

Seiten bauen

pwvergessen.htm: Seite mit E-Mail-Eingabe, ruft *pwvergessen.act* auf, Ausgabe, wenn Email nicht gefunden

Beispiele

```
<form action="pwvergessen.act" method="post">
  <label for="email">Ihre E-Mail-Adresse: </label>
```

```
  <input type="text" id="email" name="email" value="<bx:pagedata.request name="email"/>">
  <bx:pagedata.request name="fehler" value="1">Bitte geben Sie Ihre E-Mail-Adresse
an.</bx:pagedata.request>
  <bx:pagedata.request name="fehler" value="2">Mit dieser Email sind Sie nicht
angemeldet.</bx:pagedata.request>
  <br>
  <input type="submit" value="abschicken">
</form>
```

mailverschickt.htm: Info-Seite, "So geht' weiter...", nachdem Email mit dem Token verschickt wurde

changepw.htm: Seite mit Feldern zur Paßwortänderung, ruft checkpw.act auf, Ausgabe, wenn Token nicht mehr gültig

Beispiele

```
<bx:pagedata.request name="fehler" value="3">Der Link aus der E-Mail ist nicht mehr gültig.
Bitte fordern Sie einen neuen Link an.</bx:pagedata.request>
<form action="checkpw.act" method="post">
  <input type="hidden" name="token" value="<bx:pagedata.request name="token"/>">
  <input type="hidden" name="email" value="<bx:pagedata.request name="email"/>">
  <label for="Passwort">neues Paßwort:*</label>
  <input id="Passwort" type="text" name="Passwort" value=""> <bx:pagedata.request
name="fehler" value="1">Bitte geben Sie Ihr Paßwort incl. Wiederholung
an.</bx:pagedata.request><br>
  <label for="Passwortwdhlg" style="width:145px;">Paßwortwiederholung:*</label>
  <input id="Passwortwdhlg" type="text" name="Passwortwdhlg" value=""> <bx:pagedata.request
name="fehler" value="2">Die beiden Paßwörter sind nicht gleich.</bx:pagedata.request><br>
  <input type="submit" value="speichern" style="margin-left:145px;">
</form>
```

ok.htm: Info-Seite, "So geht' weiter...", nachdem PW erfolgreich geändert wurde

pwemail.htm: Mail mit dem Token

Beispiele

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Paßwort vergessen</title>
</head>
<body>
<bx:containerloop.User pool="132AF649539" idfield="tokid" force="single">
<bx:recordfield.Kontakt>
Guten Tag, <bx:recordfield.Kontakt_Anrede/> <bx:if><bx:recordfield.Kontakt_Vorname/>
</bx:if><bx:recordfield.Kontakt_Vorname/> <bx:recordfield.Kontakt_Name/>,
<br><br>
</bx:recordfield.Kontakt>
Sie hatten auf "Paßwort vergessen" geklickt. Wenn nicht, dann ignorieren Sie diese Mail.
<br><br>
Bitte folgenden Link anklicken - Sie können dann ein neues Paßwort vergeben. Dieser Link ist
einen Tag gültig und nur einmal verwendbar.
<br>
**********************************************************************************************
**********<br>
http://mrp.batix.net/www/nab/mitgliederbereich/changepw.htm?token=<bx:recordfield.Token/>&amp;
email=<bx:recordfield.Email/>
**********************************************************************************************
**********<br>
Bitte beachten Sie: Bei einigen Browsern und E-Mail-Programmen lässt sich der Link nicht
anklicken. Sollte dies der Fall sein, kopieren Sie bitte den kompletten Link in die
Adresszeile Ihres Browsers und bestätigen mit der Return-Taste.
<br><br>
Mit freundlichen Grüßen<br>
Das Notarzt-Börse-Team
</bx:containerloop.User>
</body>
</html>
```

Records – Hilfsklasse für einfaches Filtern

Für die programmatische Erstellung einer Datensatz-Filterung gab es bisher nur die Klasse `SQLGenerator`, welche teilweise schwierig zu bedienen war. Nun gibt es eine einfachere Möglichkeit mit besserer Typunterstüzung: die Klasse `com.batix.table.Records`. In Groovy-Code gibt es einen ähnlichen Helfer (findRecords), aber auch dort sollte `Records` vorgezogen werden.

Diese hat z. B. den Vorteil, dass die möglichen Filter für die unterschiedlichen Feldtypen ausdefiniert sind - man bekommt sie also per Code-Completion von der IDE vorgeschlagen und muss sich keine Zahlen merken. Außerdem wurden bestimmte Filter nochmals unterteilt, um beispielsweise `NULL` Werte zuzulassen oder nicht.

Auch in Bezug auf Sicherheit sollte der Einsatz von `Records` favorisiert werden: Es können aus Versehen keine Filter definiert werden, die in manchen Fällen dazu führen, dass die Filterung ausgehebelt wird (also man alle Datensätze sieht). Das war z. B. bisher der Fall, wenn man davon ausging, dass bestimmte Requestparameter immer da sind, es dann aber durch einen Fehler oder gezielten Angriff doch nicht so war.

> Verfügbar ab CMS Version 2.7.3

Beispiel

Zur schnellen Übersicht ist hier ein Beispiel, in dem viele Methoden verwendet werden.

```
tmd.records()
  .connection(conn) // optional
  .activeOrInactive() // default active()
  .asc("timestamp").descNullsLast("sent_at") // nach mehreren Sachen sortieren
  .index(10)
  .limit(5)
  .exclude("123ABC") // bestimmte DS-IDs includen/excluden
  .filterCheckbox("valid", "ignoreValid").checked() // mehrere Felder mit oder verknüpft, d.
h. mind. eins von beiden muss angehakt sein
  .filterSingleLink("Kategorie").anyId(collectionOfStrings)
  .filterMultipleLink("docs").containsAny("123ABC", "456DEF")
```

```
  .filterInteger("age").nullOr().greaterOrEquals(18)
  .filterFloat("price").lesser(100)
  .filterString("Vorname").notNullAnd().contains("test")
  .filterString("Nachname").nullOr().startsWith("test")
  .filterDate("Geburtsdatum").onYear(2000)
  .filterDate("Geburtsdatum").onMonthDay(MonthDay.now())
  .filterTime("wann").onHour(14)
  .filterTime("wann").later(LocalTime.of(13, 37))
  .filterDocument("doc").id("123ABC")
  //.count() -> int
  //.firstOrNull() -> ContainerRecord
  .forEach(rec -> { /*...*/ });
```

Allgemeines

Um den Zusammenbau der Konfiguration zu starten, wird eine Instanz von `TableMetadata` benötigt (im Beispiel oben `tmd`). Dann kann einfach `tmd.records()` oder `Records.from(tmd)` aufgerufen werden.

Die Klasse ist im Builder-Style gehalten. Es können also mehrere Bedingungen hintereinander definiert werden. Die Reihenfolge ist egal, da noch nichts ausgeführt, sondern nur die Abfrage-Konfiguration erstellt wird. Es sollte auf sinnvolle Zeilenumbrüche zur besseren Lesbarkeit geachtet werden (wie im Beispiel oben).

> Es muss nicht zwangsläufig alles hintereinander geschrieben werden. Man kann sich die `Records` Instanz auch auf eine Variable legen und die Methoden nur in bestimmten Fällen aufrufen, z. B. anhand von Request-Parametern.

```
final Records.RecordsBuilder recs = tmd.records()
  .filterCheckbox("deleted").notChecked()
  .asc("Titel")
  .limit(10);

if (reqSuche != null && reqSuche.length() > 0) {
  recs.filterString("Beschreibung").notNullAnd().contains(reqSuche);
}

int cnt = recs.count();
```

In Kotlin kann auch z. B. `apply` benutzt werden:

```
val cnt = tmd.records().apply {
  filterCheckbox("deleted").notChecked()

  if (!reqSuche.isNullOrEmpty()) {
    filterString("Beschreibung").notNullAnd().contains(reqSuche)
  }

  asc("Titel")
  limit(10)
}.count()
```

> Man kann sich keinen alten Stand der Konfiguration merken, da von der `Records` Instanz, die man konfiguriert bei den einzelnen Methoden keine Kopie erzeugt wird, sondern diese direkt verändert wird.

Damit die Abfrage ausgeführt wird, ist als letztes eine der Ergebnis-Methoden aufzurufen. Diese können auch mehrfach aufgerufen werden, die Abfrage wird dann immer wieder mit den aktuellen Einstellungen neu gestartet.

> Bei Methoden, die mehrere Testwerte entgegennehmen, kann auch eine Collection des entsprechenden Typs anstatt einzelner Parameter übergeben werden.

Connection

```
.connection(/* Connection */)
.connection(request)
```

Mittels `.connection()` kann eine Datenbank-Connection übergeben werden. Dies ist nicht unbedingt nötig, da `Records` ansonsten selbst eine Connection im Hintergrund aufmacht. Das kann aber bei vielen Abfragen zu Performance-Problemen führen, da immer wieder eine neue Connection aufgemacht wird. Daher sollte man sich angewöhnen, eine Connection zu übergeben. Das geht direkt mit `.connection(conn)` oder, falls man nur einen Request zur Hand hat, mit `.connection(request)`.

Filtern

Eine Suche nach Feldwerten kann mit den `.filter...()` Methoden definiert werden. Für jeden Feldtyp gibt es eine entsprechende Methode.

Der `.filter...()` Methode wird dann der Name des zu filternden Feldes übergeben.

> Einer `.filter...()` Methode können auch mehrere Feldnamen übergeben werden (als einzelne Parameter). Das führt zu einer ODER-Verknüpfung der Felder: Mindestens eins dieser Felder muss also die Bedingung erfüllen.
>
> Das entspricht im XML-Filter der Angabe mehrerer `<field>` Elemente.

Beispiel

```
.filterString("Vorname", "Nachname").notNullAnd().contains("muster")
```

> Alle Filtermethoden prüfen ihre Argumente auf `null` bzw. leer (z. B. bei Strings) und lehnen solche Werte mit einer Exception ab.
> Das bedeutet man kann weder explizit, noch aus Versehen, einen Testwert übergeben, der nicht filtert (beispielsweise einen Leerstring beim Filtern einer "Besitzer" Verknüpfung).
> Somit werden bestimmte Bugs und Angriffe zur Laufzeit verhindert. Soll ein Feld nur in manchen Fällen gefiltert werden, ist das obige Beispiel unter [Allgemeines](#bkmrk-allgemeines) anzuwenden.

Nachfolgend sind die Feldtypen mit ihren zugehörigen Filter-Methoden aufgeführt.

Text

```
.filterString("Feldname")...
```

Leer abfragen

```
.filterString("Feldname").empty()
.filterString("Feldname").notEmpty()
```

Leer erlaubt oder nicht erlaubt

```
.filterString("Feldname").notNullAnd()...
.filterString("Feldname").nullOr()...
```

Der Text-Filter teilt sich in zwei Unterbereiche, die aber im Prinzip dieselben Methoden unterstützen. Sie unterscheiden sich dadurch, dass leere Felder entweder von vornherein ausgeschlossen werden, oder alternativ, dass leere Felder auch die Bedingung erfüllen (diese

Variante wird z. B. angewendet, falls es ein einschränkendes Feld gibt, das aber auch leer sein kann, was dann bedeutet, dass die Einschränkung nicht greift und demzufolge der Datensatz auch mit gelistet werden soll).

Gleichheit

```
.filterString("Feldname").notNullAnd().value("Testwert")
.filterString("Feldname").notNullAnd().valueAny("Testwert 1", "Testwert 2")

.filterString("Feldname").notNullAnd().notValue("Testwert")
.filterString("Feldname").notNullAnd().notValueAny("Testwert 1", "Testwert 2")

.filterString("Feldname").nullOr().value("Testwert")
.filterString("Feldname").nullOr().valueAny("Testwert 1", "Testwert 2")

.filterString("Feldname").nullOr().notValue("Testwert")
.filterString("Feldname").nullOr().notValueAny("Testwert 1", "Testwert 2")
```

Um den kompletten Feldinhalt zu prüfen, wird eine der `value` Methoden verwendet.

> Da die `...Any()` Methoden intern die multiplen Testwerte in einen kommagetrennten String schreiben, ist die Verwendung eines Kommas in Testwerten für diese Methoden nicht möglich bzw. führt zu fehlerhaften Ergebnissen!

Partielle Übereinstimmung

```
.filterString("Feldname").notNullAnd().contains("Testwert")
.filterString("Feldname").notNullAnd().containsAny("Testwert 1", "Testwert 2")
.filterString("Feldname").notNullAnd().containsAll("Testwert 1", "Testwert 2")
.filterString("Feldname").notNullAnd().startsWith("Testwert")
.filterString("Feldname").notNullAnd().endsWith("Testwert")
.filterString("Feldname").notNullAnd().regex("Pattern")

.filterString("Feldname").notNullAnd().notContains("Testwert")
.filterString("Feldname").notNullAnd().notContainsAny("Testwert 1", "Testwert 2")
.filterString("Feldname").notNullAnd().notContainsAll("Testwert 1", "Testwert 2")
.filterString("Feldname").notNullAnd().notStartsWith("Testwert")
.filterString("Feldname").notNullAnd().notEndsWith("Testwert")
.filterString("Feldname").notNullAnd().notRegex("Pattern")

.filterString("Feldname").nullOr().contains("Testwert")
```

```
.filterString("Feldname").nullOr().containsAny("Testwert 1", "Testwert 2")
.filterString("Feldname").nullOr().containsAll("Testwert 1", "Testwert 2")
.filterString("Feldname").nullOr().startsWith("Testwert")
.filterString("Feldname").nullOr().endsWith("Testwert")
.filterString("Feldname").nullOr().regex("Pattern")

.filterString("Feldname").nullOr().notContains("Testwert")
.filterString("Feldname").nullOr().notContainsAny("Testwert 1", "Testwert 2")
.filterString("Feldname").nullOr().notContainsAll("Testwert 1", "Testwert 2")
.filterString("Feldname").nullOr().notStartsWith("Testwert")
.filterString("Feldname").nullOr().notEndsWith("Testwert")
.filterString("Feldname").nullOr().notRegex("Pattern")
```

Teile des Feldinhaltes können mit diesen Methoden überprüft werden.

> Da die `...Any()` und `...All()` Methoden intern die multiplen Testwerte in einen kommagetrennten String schreiben, ist die Verwendung eines Kommas in Testwerten für diese Methoden nicht möglich bzw. führt zu fehlerhaften Ergebnissen!

Bild

```
.filterPicture("Feldname")...
```

Leer abfragen

```
.filterPicture("Feldname").empty()
.filterPicture("Feldname").notEmpty()
```

Ziel-Datensatz existiert

```
.filterPicture("Feldname").exists()
.filterPicture("Feldname").notExists()
```

Diese Filter überprüfen, ob es ein Bild mit der im Feld hinterlegten ID auch wirklich gibt.

ID abfragen

```
.filterPicture("Feldname").id("123456ABCDEF")
```

Datei

```
.filterDocument("Feldname")...
```

Leer abfragen

```
.filterDocument("Feldname").empty()
.filterDocument("Feldname").notEmpty()
```

ID abfragen

```
.filterDocument("Feldname").id("123456ABCDEF")
```

Datum mit oder ohne Uhrzeit

```
.filterDate("Feldname")...
```

> Die Filtermethoden nehmen aus Sicherheitsgründen keinen String entgegen, da dieser evtl. nicht im vom Filter erwarteten Format ist. Es muss also eine Instanz von `Date | LocalDate | LocalDateTime` übergeben werden. Bei Feldern ohne Uhrzeit wird nur der Datumsteil des Testwertes gelesen.
> Falls man nur einen String hat, muss man diesen vorher in ein `LocalDate(Time)` Objekt umwandeln. Man erstellt dazu einen Formatter mit dem passenden Pattern und parst den String zum gewünschten Objekt.

Beispiele

```
LocalDateTime.parse("zu parsender String", DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss"))
LocalDate.parse("zu parsender String", DateTimeFormatter.ofPattern("dd.MM.yyyy"))
```

Leer abfragen

```
.filterDate("Feldname").empty()
.filterDate("Feldname").notEmpty()
```

(Teil)Gleichheit

```
.filterDate("Feldname").onDate(/* Date | LocalDate | LocalDateTime */)
.filterDate("Feldname").onDay(/* int (1-31) */)
.filterDate("Feldname").onMonth(/* int (1-12) */)
.filterDate("Feldname").onMonthDay(/* MonthDay */)
.filterDate("Feldname").onYear(/* int */)
```

```
.filterDate("Feldname").onYearMonth(/* YearMonth */)

.filterDate("Feldname").notOnDate(/* Date | LocalDate | LocalDateTime */)
```

Abfrage neuer / älter bzw. gleich

```
.filterDate("Feldname").newer(/* Date | LocalDate | LocalDateTime */) // Feld > Testwert
.filterDate("Feldname").newerEquals(/* Date | LocalDate | LocalDateTime */) // Feld >=
Testwert

.filterDate("Feldname").older(/* Date | LocalDate | LocalDateTime */) // Feld < Testwert
.filterDate("Feldname").olderEquals(/* Date | LocalDate | LocalDateTime */) // Feld <=
Testwert
```

Abfrage neuer / älter oder Leer

```
.filterDate("Feldname").nullOr().newerEquals(/* Date | LocalDate | LocalDateTime */)
.filterDate("Feldname").nullOr().olderEquals(/* Date | LocalDate | LocalDateTime */)
```

Uhrzeit

```
.filterTime("Feldname")...
```

> Auch hier nehmen die Methoden - wie oben bei Datum mit oder ohne Uhrzeit - keinen String entgegen. Dieser muss zunächst in ein passendes Objekt umgewandelt werden. Bei Objekten vom Typ `Date` oder `LocalDateTime` wird nur der Uhrzeitteil gelesen.

Beispiele

```
LocalTime.parse("zu parsender String", DateTimeFormatter.ofPattern("HH:mm:ss"))
LocalDateTime.parse("zu parsender String", DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss"))
```

Leer abfragen

```
.filterTime("Feldname").empty()
.filterTime("Feldname").notEmpty()
```

(Teil)Gleichheit

```
.filterTime("Feldname").onTime(/* Date | LocalTime | LocalDateTime */)
.filterTime("Feldname").onHour(/* int (0-23) */)
.filterTime("Feldname").onMinute(/* int (0-59) */)
.filterTime("Feldname").onSecond(/* int (0-59) */)

.filterTime("Feldname").notOnTime(/* Date | LocalTime | LocalDateTime */)
```

Abfrage neuer / älter bzw. gleich

```
.filterTime("Feldname").later(/* Date | LocalTime | LocalDateTime */) // Feld > Testwert
.filterTime("Feldname").laterEquals(/* Date | LocalTime | LocalDateTime */) // Feld >=
Testwert

.filterTime("Feldname").earlier(/* Date | LocalTime | LocalDateTime */) // Feld < Testwert
.filterTime("Feldname").earlierEquals(/* Date | LocalTime | LocalDateTime */) // Feld <=
Testwert
```

Abfrage neuer / älter oder Leer

```
.filterTime("Feldname").nullOr().laterEquals(/* Date | LocalTime | LocalDateTime */)
.filterTime("Feldname").nullOr().earlierEquals(/* Date | LocalTime | LocalDateTime */)
```

Ganzzahl

```
.filterInteger("Feldname")...
```

Leer abfragen

```
.filterInteger("Feldname").empty()
.filterInteger("Feldname").notEmpty()
```

Gleichheit

```
.filterInteger("Feldname").value(/* Number */)
.filterInteger("Feldname").notValue(/* Number */)

.filterInteger("Feldname").nullOr().value(/* Number */)
.filterInteger("Feldname").nullOr().notValue(/* Number */)
```

Abfrage größer / kleiner

```
.filterInteger("Feldname").greater(/* Number */)
.filterInteger("Feldname").greaterOrEquals(/* Number */)
.filterInteger("Feldname").lesser(/* Number */)
.filterInteger("Feldname").lesserOrEquals(/* Number */)

.filterInteger("Feldname").nullOr().greater(/* Number */)
.filterInteger("Feldname").nullOr().greaterOrEquals(/* Number */)
.filterInteger("Feldname").nullOr().lesser(/* Number */)
.filterInteger("Feldname").nullOr().lesserOrEquals(/* Number */)
```

Dezimalzahl, Preis

```
.filterFloat("Feldname")...
```

Die Methoden sind identisch zu Ganzzahl, es muss lediglich `filterFloat` als Methode verwendet werden.

> Alle Vergleiche auf Gleichheit bzw. Ungleichheit verwenden eine maximale Genauigkeit von 0,00001.
> Das bedeutet, dass z. B. 0,000004 und 0,000005 für den Filter gleich sind.

Wahrheitswert (Checkbox)

```
.filterCheckbox("Feldname")...
```

Prüfen ob angehakt

```
.filterCheckbox("Feldname").checked()
.filterCheckbox("Feldname").notChecked()
```

Prüfen ob leer

```
.filterCheckbox("Feldname").empty()
.filterCheckbox("Feldname").notEmpty()
```

> Dies ist in den meisten Fällen nicht der richtige Filter und es sollte "Prüfen ob angehakt" benutzt werden.
> Hier wird nur geprüft, ob noch kein expliziter Wert im Feld steht, egal ob angehakt oder nicht angehakt.

> Wurde die Checkbox also als **nicht angehakt** gespeichert, ist das Feld **nicht leer** (da "n" oder 0 drin steht).

Koordinaten

```
.filterCoordinate("Feldname")...
```

> Dieser Feldtyp ist veraltet und wird nicht mehr benutzt. Er wird hier nur der Vollständigkeit halber erwähnt.

Prüfen ob leer

```
.filterCoordinate("Feldname").empty()
.filterCoordinate("Feldname").notEmpty()
```

Gleichheit der Koordinaten

```
.filterCoordinate("Feldname").x(/* Number */)
.filterCoordinate("Feldname").y(/* Number */)
```

Einzelverknüpfung

```
.filterSingleLink("Feldname")...
```

Prüfen ob leer

```
.filterSingleLink("Feldname").empty()
.filterSingleLink("Feldname").notEmpty()
```

Prüfen ob Datensatz existiert

```
.filterSingleLink("Feldname").exists()
.filterSingleLink("Feldname").notExists()
```

Es wird geprüft, ob es auch einen Datensatz mit der hinterlegten ID gibt bzw. nicht gibt.

Prüfen ob Datensatz aktiv

```
.filterSingleLink("Feldname").active()
.filterSingleLink("Feldname").notActive()
```

Es wird geprüft, ob es den Datensatz gibt. Dieser muss außerdem aktiviert bzw. deaktiviert sein.

Nach ID suchen

```
.filterSingleLink("Feldname").id("123456ABCDEF")
.filterSingleLink("Feldname").notId("123456ABCDEF")

.filterSingleLink("Feldname").anyId("123456ABCDEF", "ABCDEF123456")
```

Mehrfachverknüpfung

```
.filterMultipleLink("Feldname")...
```

Prüfen ob leer

```
.filterMultipleLink("Feldname").empty()
.filterMultipleLink("Feldname").notEmpty()
```

Nach ID suchen

```
.filterMultipleLink("Feldname").contains("123456ABCDEF")
.filterMultipleLink("Feldname").notContains("123456ABCDEF")

.filterMultipleLink("Feldname").containsAny("123456ABCDEF", "ABCDEF123456")
.filterMultipleLink("Feldname").containsAll("123456ABCDEF", "ABCDEF123456")
```

Bei `containsAny` reicht es, wenn eine der übergebenen IDs im Feld verknüpft ist. Für `containsAll` müssen alle übergebenen IDs verknüpft sein. In beiden Fällen können aber auch noch andere IDs verknüpft sein.

Untertabelle (Untercontainer)

```
.filterSubList("Feldname")...
```

Prüfen ob leer

```
.filterSubList("Feldname").empty()
.filterSubList("Feldname").notEmpty()
```

Aktiv / Inaktiv

```
.active()
.activeOrInactive()
.inactive()
```

Standardmäßig werden nur aktive Datensätze zurückgegeben. Dies lässt sich aber anpassen.

IDs inkludieren / exkludieren

```
.include("123456ABCDEF", "ABCDEF123456")
.exclude("123456ABCDEF", "ABCDEF123456")
```

Zusätzlich zu den Feldfiltern besteht die Möglichkeit nur bestimmte Datensätze anhand ihrer ID zuzulassen oder auszuschließen.

Werden Feldfilter **und** ID-Filter benutzt, müssen beide zutreffen. Ein mittels `include` angegebener Datensatz taucht also nicht zwangsläufig auf, sondern nur wenn alle anderen Bedingungen auch passen (sofern welche definiert wurden).

Paginierung

```
.limit(/* Integer */)
.index(/* Integer */) // 0-basiert
```

Die Anzahl der Ergebnisse kann mit `limit` begrenzt werden. Ebenso kann die Startposition mit `index` festgelegt werden.

> Da `index` nullbasiert ist, startet man bei einer beispielhaften Seitenanzahl von zehn Ergebnissen mit dem elften Ergebnis (also Seite zwei), indem man `index(10)` angibt und nicht etwa `11` .

Sortierung

```
.asc("Feldname") // aufsteigend
.desc("Feldname") // absteigend
```

```
.ascNullsLast("Feldname")
.descNullsLast("Feldname")
```

Es kann nach mehreren Feldern sortiert werden. Dabei ist die Reihenfolge entscheidend. Es wird zuerst nach dem ersten angegebenen Sortierfeld sortiert, dann nach dem zweiten, usw. für alle angegebenen Sortierfelder.

Standardmäßig werden `NULL` Werte an den Anfang sortiert. Dies kann mit den `...NullsLast()` Methoden geändert werden.

Im folgenden Beispiel wird also zunächst nach `timestamp` aufsteigend sortiert. Datensätze mit leerem `timestamp` Feld werden zuerst zurückgegeben.
Danach werden Datensätze mit gleichem Wert bei `timestamp` nochmals anhand `sent_at` absteigend sortiert. Dabei werden solche mit leerem `sent_at` ans Ende der Untersortierung gesetzt.

```
.asc("timestamp")
.descNullsLast("sent_at")
```

Ergebnisse laden

Es gibt verschiedene Möglichkeiten die Ergebnisse abzurufen. Wie bereits erwähnt kann dies auch mehrfach geschehen, indem man sich den `Records` Builder auf eine Variable legt und dessen Ergebnismethoden mehrfach aufruft.

Anzahl

```
.count() // int
```

Diese Methode gibt lediglich die Anzahl der gefundenen Datensätze zurück.

Erster Datensatz

```
.firstOrNull() // ContainerRecord oder null
```

Um nur den ersten Datensatz zu laden, kann diese Methode benutzt werden. Entspricht kein Datensatz dem Filter, wird `null` zurückgegeben, ansonsten ein `ContainerRecord`.

Datensätze durchgehen

```
.forEach((ContainerRecord record) -> {
  // ...
});
```

Hier werden die Datensätze aus der Datenbank gestreamt und dem Lambda (der Closure) einzeln übergeben. Die Methode `forEach` blockiert solange, bis alle Datensätze vom Lambda verarbeitet wurden.
Es werden dabei **nicht** zunächst alle Datensätze in den Speicher geladen, was der Performance zuträglich ist. Daher ist dies die präferierte Methode um Ergebnisse zu durchlaufen.

Stream

```
.stream() // Stream<ContainerRecord>
```

Genau wie bei `forEach` werden hier die Datensätze aus der Datenbank gestreamt. Man erhält allerdings direkten Zugriff auf den Stream.

> Der Stream muss geschlossen werden, da sonst Leaks entstehen, da die Datenbankobjekte nicht aufgeräumt werden (weil der Stream ja nicht weiß, dass er fertig ist).
>
> Das geschieht z. B. automatisch durch terminale Stream-Methoden oder manuell mittels `close()`.

Alle Datensätze laden

```
.loadAll() // ContainerRecord[]
```

Es wird ein Array zurückgegeben. Falls keine Elemente gefunden wurden, ist das Array leer (nicht `null`).

> Diese Methode lädt alle gefundenen Datensätze in den Speicher, was zu Abstürzen fehlen kann. Daher sind die anderen Methoden vorzuziehen.

Regelmäßiges Neuladen von DIVs

Aufbau der Hauptseite

```
<html>
<head>
<script type="text/javascript" src="mootools-1.2.js"></script>
<script type="text/javascript">
function registerRedirect(timeout, url, target){
  target.callUpdate = function(){
    new Request.HTML({
      url: url,
      noCache: true,
      update: target,
      onFailure: function(){
        setTimeout(target.callUpdate, 5 * 1000)
      }
    }).get();
  }
  setTimeout(target.callUpdate, timeout * 1000)
}

window.addEvent('domready', function() {
  registerRedirect(2, "oben.html", $("div1"))
  registerRedirect(2, "unten.html", $("div2"))
});
</script>
</head>
<body>
<div id="div1">div 1</div>
<br><hr><br>
<div id="div2">div 2</div>
</body>
</html>
```

Der Inhalt einer nachgeladenen Seite sieht so aus:

```
<script type="text/javascript">
registerRedirect(2, "oben.html", $("div1"))
</script>
... Content für oben steht hier ...
```

Sitemap bauen

Einfaches Sitemap, kann noch verschönert werden.

Styles

```
.ebene0{
  font-weight:bold;
  text-decoration:underline;
  }
.ebene1{
  margin-left:30px;
  }
.ebene2{
  margin-left:60px;
  }
```

Quellcode

```
<bx:sitemap.0>
  <div class="ebene0"><a href="<bx:navigation.target default="http://" /><bx:pagedata.hostname
/>/www/<bx:pagedata.webdir/>/<bx:navigation.path/>"><bx:navigation.name/></a></div>
  <bx:sitemap.1>
    <div class="ebene1"><a href="<bx:navigation.target default="http://"
/><bx:pagedata.hostname
/>/www/<bx:pagedata.webdir/>/<bx:navigation.path/>"><bx:navigation.name/></a></div>
    <bx:sitemap.2>
      <div class="ebene2"><a href="<bx:navigation.target default="http://"
/><bx:pagedata.hostname
/>/www/<bx:pagedata.webdir/>/<bx:navigation.path/>"><bx:navigation.name/></a></div>
    </bx:sitemap.2>
  </bx:sitemap.1>
</bx:sitemap.0>
```

Support für „not" hinzufügen

Dieses Beispiel realisiert folgende Funktionen:

- `not` für ein Tag emulieren, dass `not` nicht unterstützt

Folgende Tags wurden verwendet:

- Bx:clipboard
- Bx:if
- Bx:pagedata

Quellcode

Als Beispiel dient hier das Tag bx:pagedata.navid, welches `not` für seine Bedingung nicht unterstützt.

```
<bx:clipboard.cut name="check">
<bx:pagedata.navid is="12345678901"><!-- --></bx:pagedata.navid>
</bx:clipboard.cut>
<bx:clipboard.cut name="content">
Inhalt der bei allen Menüpunkten außer dem Gesuchten angezeigt wird
</bx:clipboard.cut>
<bx:if type="one">
<bx:clipboard.paste name="check" />
<bx:clipboard.paste name="content" />
</bx:if>
```

Das "check" Clipboard wird nur gefüllt, wenn ein bestimmter Menüpunkt aufgerufen wird. Dagegen wird das "content" Clipboard immer mit dem gewünschten Inhalt gefüllt. Da <bx:if> aber im Modus `"one"` läuft, wird der Inhalt auch nur ausgegeben, wenn sich nur im "content" Clipboard etwas befindet, d.h. wenn die NavID des aktuellen Menüpunktes **nicht** mit der zu überprüfenden übereinstimmt!

Validation

Validation eines Container-Datensatzes

Mit einem Aktionsbaustein vom Typ ValidationAction werden die zu validierenden Felder und ihre Gültigkeitsbedingungen konfiguriert. Danach folgt ein Aktionsbaustein vom Typ ValidationResultAction der das Validationsergebnis im XML-Format in ein Feld des Containerdatensatzes schreibt.

Auf der Formularseite können durch das Tag bx:validation die fehlerhaften Eingabefelder markiert, oder ein Kommentar angezeigt werden.

Validation von übergebenen Requestparametern

Der Aktionsbaustein FormValidatorAction validiert Requestparameter gegen reguläre Ausdrücke. Die regulären Ausdrücke werden als `<input type="hidden">`-Felder an die Action-URL übergeben.

Validation eines einzelnen Requestparameters

Der Aktionsbaustein ConditionAction dient eigentlich zur Verzweigung des Ablaufs innerhalb einer Aktion. Ein übergebener Requestparameter wird auf einen in der Aktion festgelegen wert geprüft. Abhängig von Ergebnis kann an andere URLs weitergeleitet werden.

Websuche

Voraussetzung für ein Funktionieren der Websuche ist eine Indizierung der Seiten, z.B. via Zeitsteuerung. Dies geht erst, wenn das Web fertig, also ungeschützt und unter der endgültigen Url erreichbar ist.

Suchformular

```
<div id="suchdiv">
  Suche:
  <form action="/www/<bx:pagedata.webdir/>/sonstiges/suche/suche.act" method="post">
    <input type="text" name="search" size="18" value="<bx:websearch.query/>"><input
type="submit" value="los">
  </form>
</div>
```

Action:

suche.act

Typ der Aktion: Suche im Web

resultpage: Pfad zur Ausgabeseite, z.B. /suche/

Ausgabeseite

```
<strong>Sie suchten nach: <bx:websearch.query/></strong><br>

<div style="text-align:center;">
  <div style="float:left;"><bx:websearch.previous hide/></div>
  <div style="float:right;"><bx:websearch.next hide/></div>
  <bx:websearch.if not empty>
  Ergebnis <bx:websearch.firstshown/> bis <bx:websearch.lastshown/> von insgesamt
<bx:websearch.totalhits/>
  </bx:websearch.if> 
</div>
<hr>
<bx:websearch.if empty>Es wurden leider keine Ergebnisse gefunden.</bx:websearch.if>
<bx:websearch.if not empty>
  <bx:websearch max="10">
```

```
<strong><bx:websearch.rank/>.</strong> <bx:websearch.link><bx:websearch.title/></bx:websearch.link>  <bx:websearch.stars max="5"><img src="star.jpg"> </bx:websearch.stars><br>
    <div style="margin-left:30px;"><bx:websearch.description/></div><br><br>
  </bx:websearch>
</bx:websearch.if>
```