

Logical Theme System

- Anzeige der Revisionen in einem PDF Export
- URL mit Mausklick kopieren
- tabellarische Darstellung der Tags in einem PDF Export
- PDF Export mit QR-Code
- News-Seite / schwarzes Brett
- Regale zu denen ein Buch gehört anzeigen

Anzeige der Revisionen in einem PDF Export

getestet mit Version **24.02**

Anforderung

Um die Revisionen und ggfs. den Changelog in einem PDF Export ganz zum Schluss anzeigen lassen zu können sind einige Anpassungen nötig.

Zuerst habe ich dafür einen zusätzlichen Link in das Export-Menü eingebaut um einmal eine Version ohne Revisionen und eine mit Revisionen exportieren zu können.

betroffene Dateien

Dateien müssen sich in der entsprechenden Struktur unterhalb des Themes befinden.

Ordner = *kursiv*

Dateien = **fett**

- *entities*
 - **export-menu.blade.php**
- *exports*
 - *parts*
 - **revisions-index-row-compact.blade.php**
 - **page.blade.php**

Inhalte der Dateien

export-menu.blade.php

Hier wurde ein weiterer Link in **Zeile 6** eingefügt, der mittels GET den Wert `history=true` übergibt. Diese GET Variable kann dann später abgefragt werden.

[...]

```
<ul refs="dropdown@menu" class="wide dropdown-menu" role="menu">
  <li><a href="{{ $entity->getUrl('/export/html') }}" target="_blank" class="label-item"><span>{{
trans('entities.export_html') }}</span><span>.html</span></a></li>
  <li><a href="{{ $entity->getUrl('/export/pdf') }}" target="_blank" class="label-item"><span>{{
trans('entities.export_pdf') }}</span><span>.pdf</span></a></li>
  <li><a href="{{ $entity->getUrl('/export/pdf?qr=true') }}" target="_blank" class="label-
item"><span>{{ trans('entities.export_pdf') }} + QR</span><span>.pdf</span></a></li>
  <li><a href="{{ $entity->getUrl('/export/pdf?history=true') }}" target="_blank" class="label-
item"><span>{{ trans('entities.export_pdf') }} + History</span><span>.pdf</span></a></li>
  <li><a href="{{ $entity->getUrl('/export/plaintext') }}" target="_blank" class="label-
item"><span>{{ trans('entities.export_text') }}</span><span>.txt</span></a></li>
  <li><a href="{{ $entity->getUrl('/export/markdown') }}" target="_blank" class="label-
item"><span>{{ trans('entities.export_md') }}</span><span>.md</span></a></li>
</ul>
```

[...]

revisions-index-row-compact.blade.php

Ich habe in dem Ordner eine weitere Datei angelegt und mit folgendem Inhalt gefüllt:

```
<tr>
  <td>{{ $revision->created_at->isoFormat('D MMMM Y') }}</td>
  <td>{{ $revision->revision_number == 0 ? " : $revision->revision_number }}</td>
  <td>@if($revision->createdBy) {{ $revision->createdBy->name }} @else {{
trans('common.deleted_user') }} @endif</td>
  <td>{{ $revision->summary }}</td>
</tr>
```

page.blade.php

In dieser Datei habe ich an das Ende der Seite folgenden Code eingefügt und die alte Meta Ansicht deaktiviert. Der geänderte Code beginnt in **Zeile 7** (in diesem Codeschnipsel).

```

[...]
```

```

    {!! $page->renderedHTML ?? $page->html !!}
</div>

@if(request()->query('history'))
    <hr>
    <h2>Dokumentenhistorie</h2>
    <table>
        <tr>
            <th>Datum</th>
            <th>Version</th>
            <th>Autor</th>
            <th>Anmerkungen</th>
        </tr>
        @if(count($page->revisions) > 0)
            @foreach($page->revisions as $index => $revision)
                @include('exports.parts.revisions-index-row-compact', ['revision' => $revision, 'current' =>
$page->revision_count === $revision->revision_number])
            @endforeach
        @else
            <p>{{ trans('entities.pages_revisions_none') }}</p>
        @endif
    </table>
@endif

<!-- <hr>

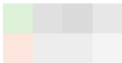
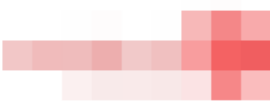
<div class="text-muted text-small">
    @include('exports.parts.meta', ['entity' => $page])
</div> -->
@endsection

```

Der Codeabschnitt kann auch an jeder anderen Stelle hinterlegt werden, für mich hat es aber am Ende des Dokuments den meisten Sinn gemacht.

Ich habe bei den Anpassungen auf Übersetzungen verzichtet, das würde sich aber problemlos ändern lassen.

Screenshots



Dokumentenhistorie

| Datum | Version | Autor | Anmerkungen |
|---------------|---------|----------------|--------------------------|
| 10 April 2024 | 3 | Sascha Jelinek | Anpassungen Formatierung |
| 10 April 2024 | 2 | Sascha Jelinek | Formatierungen angepasst |
| 10 April 2024 | 1 | Sascha Jelinek | Erste Veröffentlichung |

URL mit Mausklick kopieren

getestet mit Version **24.02**

Anforderung

Wenn man Bookstack als PWA nutzt ist es nur umständlich möglich, die URL zu kopieren wenn man diese jemandem schicken möchte.

betroffene Dateien

Dateien müssen sich in der entsprechenden Struktur unterhalb des Themes befinden.

Ordner = *kursiv*

Dateien = **fett**

- *entities*
 - **share-link.blade.php**
- *shelves*
 - **show.blade.php**
- *books*
 - **show.blade.php**
- *chapters*
 - **show.blade.php**
- *page*
 - **show.blade.php**

Inhalte der Dateien

share-link.blade.php

```

<button type="button"
  id="share-link-button"
  data-success-text="Link copied to clipboard!"
  class="icon-list-item text-link">
  <span>@icon('share')</span>
  <span>{{ trans('common.share') }}</span>
</button>
<script nonce="{{ $cspNonce }}">
  (async function() {
    const shareButton = document.getElementById('share-link-button');
    shareButton.addEventListener('click', event => {
      copyTextToClipboard(window.location.href);
      window.$events.success(shareButton.dataset.successText);
    });

    async function copyTextToClipboard(text) {
      if (window.isSecureContext && navigator.clipboard) {
        await navigator.clipboard.writeText(text);
        return;
      }

      // Backup option where we can't use the navigator.clipboard API
      const templInput = document.createElement('textarea');
      templInput.style = 'position: absolute; left: -1000px; top: -1000px;';
      templInput.value = text;
      document.body.appendChild(templInput);
      templInput.select();
      document.execCommand('copy');
      document.body.removeChild(templInput);
    }
  })()
</script>

```

show.blade.php (gleich für jede Datei)

Hier wurde die Zeile 10 hinzugefügt (Zeilennummer nur für diesen Ausschnitt)










```
@if($watchOptions->canWatch() && !$watchOptions->isWatching())
    @include('entities.watch-action', ['entity' => $page])
@endif
@if(user()->hasAppAccess())
    @include('entities.favourite-action', ['entity' => $page])
@endif
@if(userCan('content-export'))
    @include('entities.export-menu', ['entity' => $page])
@endif
@include('entities.share-link', ['entity' => $page])
</div>

</div>

@stop
```

Screenshots

Actions

-  Edit
 -  Copy
 -  Move
 -  Revisions
 -  Permissions
 -  Delete
-
-  Favourite
 -  Export
 -  Share (copy link)



tabellarische Darstellung der Tags in einem PDF Export

getestet mit Version **24.02**

Anforderung

Für diverse Zertifizierungsdokument wird ein "Dokumentenheader" benötigt. Da ich die Informationen in Tags versteckt habe lasse ich diese einfach als Tabelle bei einem PDF Export darstellen.

betroffene Dateien

Dateien müssen sich in der entsprechenden Struktur unterhalb des Themes befinden.

Ordner = *kursiv*

Dateien = **fett**

- *exports*
 - *parts*
 - **tag-export-table.blade.php**
 - **page.blade.php**

Inhalte der Dateien

tag-export-table.blade.php

```
<tr>
  <th>{{ $tag->name }}</th>
  @if($tag->value)<td>{{ $tag->value }}</td>@else<td>&nbsp;</td>@endif
```

</tr>

page.blade.php

An der gewünschten Stelle muss folgender Code eingefügt werden:

```
@if($page->tags->count() > 0)
    <h2>Dokumenteninformationen</h2>
    <table>
        @foreach($page->tags as $tag)
            @include('exports.parts.tag-export-table', ['tag' => $tag])
        @endforeach
    </table>
</div>
<hr>
@endif
```

Screenshots



- [blurred item]

Dokumenteninformationen

| | |
|-------------------------|----------------|
| Freigabe | zur Freigabe |
| Klassifizierung | intern |
| Gültigkeit | unbegrenzt |
| Überarbeitungsintervall | jährlich |
| Verantwortlicher | [blurred name] |
| Überarbeitungszeit | 10.2022 |

PDF Export mit QR-Code

getestet mit Version **24.02**

Anforderung

Ich wollte gerne die Möglichkeit haben, exportierte Daten einfach wieder in der digitalen Welt zu finden. Was ist dazu besser geeignet als ein **QR-Code**.

betroffene Dateien

Dateien müssen sich in der entsprechenden Struktur unterhalb des Themes befinden.

Ordner = *kursiv*

Dateien = **fett**

- *layouts*
 - *parts*
 - **export-body-start.blade.php**
- *entities*
 - **export-menu.blade.php**

Inhalte der Dateien

export-body-start.blade.php

Vor dem ersten `div`-Container muss hier folgendes eingetragen werden:

```
@inject('totp', 'BookStack\Access\Mfa\TotpService')

@php
$qrcode = $totp->generateQrCodeSvg($page->getUrl());
```

```
$imgStr = 'data:image/svg+xml;base64,' . base64_encode($qrCode);  
@endphp
```

Im Anschluss kann an einer beliebigen Stelle das Bild an die PDF übergeben werden. Dazu muss folgender Abschnitt hinzugefügt werden:

```
@if(request()->query('qr'))  
    <div style="float: right;">  
        getUrl() }}">  
    </div>  
@endif
```

export-menu.blade.php

Hier muss nun einfach nach der Zeile gesucht werden mit dem Inhalt `/export/pdf`.

Danach die Zeile kopieren und den String `?qr=true` anhängen an die selbe Stelle.

Es sollte dann wie folgt aussehen:

```
<ul refs="dropdown@menu" class="wide dropdown-menu" role="menu">  
    <li><a href="{{ $entity->getUrl('/export/html') }}" target="_blank" class="label-item"><span>{{  
trans('entities.export_html') }}</span><span>.html</span></a></li>  
    <li><a href="{{ $entity->getUrl('/export/pdf') }}" target="_blank" class="label-item"><span>{{  
trans('entities.export_pdf') }}</span><span>.pdf</span></a></li>  
    <li><a href="{{ $entity->getUrl('/export/pdf?qr=true') }}" target="_blank" class="label-  
item"><span>{{ trans('entities.export_pdf') }} + QR</span><span>.pdf</span></a></li>  
    <li><a href="{{ $entity->getUrl('/export/plaintext') }}" target="_blank" class="label-item"><span>{{  
trans('entities.export_text') }}</span><span>.txt</span></a></li>  
    <li><a href="{{ $entity->getUrl('/export/markdown') }}" target="_blank" class="label-  
item"><span>{{ trans('entities.export_md') }}</span><span>.md</span></a></li>  
</ul>
```

In diesem Ausschnitt ist in Zeile 4 der neue Export Link hinzugefügt.

Screenshots



Favoriten



Ex



HTML-Datei .html

PDF-Datei .pdf

PDF-Datei + QR .pdf

Textdatei .txt

Markdown-Datei .md

News-Seite / schwarzes Brett

getestet mit Version 24.12

Anforderung

Zum Abbilden der Funktion eines schwarzen Bretts bzw. einer News-Seite um aktuelle Infos anzuzeigen.

betroffene Dateien

Dateien müssen sich in der entsprechenden Struktur unterhalb des Themes befinden.

Ordner = *kursiv*

Dateien = **fett**

- *home*
 - **specific-page.blade.php**

Inhalte der Dateien

specific-page.blade.php

Innerhalb der `section('left')` oder `section('right')`, je nach Präferenz, muss folgender `<div>`-Block eingefügt werden:

```
<div class="card mb-xl">
    @php
        $newsBookId = 34;
        $newsItems = \BookStack\Entities\Models\Page::visible()
            ->where('book_id', $newsBookId)
```



```






->orderBy('created_at', 'desc')
->take(7)
->get();
@endphp
<h3 class="card-title" style="font-weight: bold; font-size: 15pt;">{{ trans('common.actualnews')
}}</h3>
<div class="px-m">
    @include('entities.list', [
        'entities' => $newsItems,
        'style' => 'compact',
    ])
</div>
<a href="{{ url('/books/firmen-aushange-schwarzes-brett') }}" class="card-footer-link">{{
trans('common.view_all') }}</a>
</div>

```

die ID für das Buch welches hier genutzt werden soll lässt sich einfach herausfinden, in dem ein Buch öffnet und dann die Entwicklertools startet (**F12**). Hier sucht man dann nach folgendem Begriff: `option:entity-search:entity-id`
Direkt dahinter steht die ID des Buches, welches dann in der Anpassung hinterlegt werden muss.

Screenshots

Latest News

-  Bookstack Update 24.02
-  Bookstack Update 23.12 - Standard-Vorlagen
-  Das neue Ticketsystem ist live!
-  neue Startseite
-  Link in Zwischenablage kopieren

[Alle anzeigen](#)

Regale zu denen ein Buch gehört anzeigen

getestet mit Version **25.02**

Anforderung

Ich wollte sehen in welchen Regalen ein Buch steht, da ein Buch in mehreren Regalen stehen kann und weil die Brotkrumen Navigation das Regal nicht anzeigt wenn man nicht das Buch z.B. über die Suche öffnet.

betroffene Dateien

Dateien müssen sich in der entsprechenden Struktur unterhalb des Themes befinden.

Ordner = *kursiv*

Dateien = **fett**

- *books*
 - **show.blade.php**

Inhalte der Dateien

show.blade.php

Unter `@section('left')` einfach an der gewünschten Stelle den folgenden Code einfügen

```
[...]
```

```
@if(count($bookParentShelves) > 0)
```

```
<div class="actions mb-xl">
  <h5>{{ trans('entities.shelves') }}</h5>
  @include('entities.list', ['entities' => $bookParentShelves, 'style' => 'compact'])
</div>
@endif
[...]
```

Screenshots

