Klassifikation mittels Barcodetrennung

> Ab SQUEEZE Version 2.5.3 möglich

Um ein Dokument in SQUEEZE mittels Barcodes zu trennen und die Splits direkt anhand des Barcodes zu klassifizieren kann der folgende UserExit verwendet werden.

Voraussetzungen

An der Stapelklasse müssen für den Barcode Split die Stapelklasseneigenschaften

- `SplitBarcodeType`
- `SplitBarcodePattern`

gesetzt werden.

Funktionsweise

Die Splits werden durch den gefundenen Barcode des entsprechenden Typs identifiziert.

Dabei kann die Seite, welche den Split identifiziert (also auf dem der Barcode gefunden wurde) selbst weggelassen werden.
Dafür wird das Flag `isSplitPage` genutzt.

```
// dismiss split pages
if ($isSplitPage) {
$logger->debug(sprintf("Barcode split page %s will be skipped", $page), $logContext);
} else {
  $splits[$currentSplit][] = [
    'page' => $page,
    'type' => $barcodeType,
    'value' => $barcodeValue,
    'fields' => $fieldValuesToKeep,
    'suppressOcr' => $suppressOCR,
    "documentClassId" => $documentClassId
  ];
}
```

Alternativ kann die Seite, welche den Split identifiziert (also auf dem der Barcode gefunden wurde), dem Split hinzugefügt werden.
Dafür einfach das Array weiter befüllen und das `isSplitPage` Flag ignorieren.

```
// include split pages
$splits[$currentSplit][] = [
    'page' => $page,
    'type' => $barcodeType,
    'value' => $barcodeValue,
    'fields' => $fieldValuesToKeep,
    'suppressOcr' => $suppressOCR,
    "documentClassId" => $documentClassId
];
```

Die Dokumentenklassen ID kann dem Split mit dem Attribut `documentClassId` zugewiesen werden.

> Pro Split wird für das Attribut `documentClassId` der ersten Seite des Splits (erstes Array Element des Splits) ausgewertet.

> Es dürfen nur valide Dokumentenklassen der Stapelklassen gesetzt werden! Invalide Dokumentenklassen werden ignoriert.

Beispielhafter UserExit

In dem folgenden Beispiel werden

- Barcode Pages entfernt
- Die Dokumentenklasse des Splits anhand des Barcodewertes ermittelt, welcher den Namen der Dokumentenklasse enthält.

```
/**
 * Retrieves the document class ID by name.
 *
 * @param string $name The name of the document class.
 * @return int The ID of the document class.
 */
function getDocumentClassIdByName(string $name): int
{
    $documentClass = new xDocumentClass();
    $documentClass->getByName($name);
    if (is_null($documentClass->getId())) {
```

```
        return 0;
    }
    return (int)$documentClass->getId();
}
```

```
if (isMatchingBarcode($splitBarcodeType, $splitBarcodePattern, $barcode["value"],
$barcode["type"])) {
  // ...
  $documentClassId = getDocumentClassIdByName($barcodeValue);
  // ...
}
```

Code

```
<?php

declare(strict_types=1);

namespace Squeeze\UserExits\Common;

use App\V2\Base\Config\ServerConfig;
use App\V2\Base\DI\Dependencies;
use App\V2\Base\Logging\Logging;
use Exception;
use Psr\Log\LoggerInterface;
use Squeeze\xBatchClass;
use Squeeze\xDoc;
use Squeeze\xDocumentClass;
use Squeeze\xQueueEntry;
use Squeeze\xReturnObject;
use Squeeze\xTools;

/**
 * @param xDoc $xDoc
 * @param string $nextStep
 * @param string $newStatus
 * @return xReturnObject
 * @throws Exception
 */
```

```
function BarcodeSeparation(xDoc $xDoc, string $nextStep, string $newStatus): xReturnObject
{
    $logger = Logging::get();

    $queueEntry = new xQueueEntry();
    $queueEntry->getByxDocId($xDoc->getId());
    $batchClass = new xBatchClass();
    $batchClass->getById($queueEntry->getBatchClassId());
    // Get barcode engine from batch class settings
    $splitBarcodeType = $batchClass->getSettingValue('SplitBarcodeType') ?? "";
    $splitBarcodePattern = $batchClass->getSettingValue('SplitBarcodePattern') ?? "";

    if (!empty($splitBarcodePattern) && !empty($splitBarcodeType)) {
        return splitByBarcode($xDoc, $logger, $splitBarcodeType, $splitBarcodePattern);
    }

    $splitFixPages = (int)($batchClass->getSettingValue('SplitFixPages') ?? 0);
    if ($splitFixPages > 0) {
        return splitFixPages($xDoc, $logger, $splitFixPages);
    }

    return new xReturnObject(false, 400, 'No Split configured', []);
}

/**
 * Splits the given xDoc into fixed pages and returns an xReturnObject containing the splits.
 *
 * @param xDoc $xDoc The xDoc to be split.
 * @param LoggerInterface $logger The logger instance.
 * @param int $fixedPageSize The size of each fixed page.
 * @throws Exception If an error occurs during the split process.
 * @return xReturnObject The xReturnObject containing the splits.
 */
function splitFixPages(xDoc $xDoc, LoggerInterface $logger, int $fixedPageSize): xReturnObject
{
    // Define which field values to keep
    $fieldNamesToKeep = [];
    $fieldValuesToKeep = getFieldValuesToKeep($xDoc, $fieldNamesToKeep);

    // initialize
    $logContext = [Logging::CTX_DOCUMENT_ID => $xDoc->getId()];
```

```
    $container = Dependencies::get();
    $config = $container->get(ServerConfig::class);
    $repo = $config->get("repository.work");

    $splits = [];
    $absolutePath = xTools::buildAbsolutePath($repo, $xDoc->getRepoPath(), false);
    $files = array_diff(scandir($absolutePath . "Viewer"), ['.', '..']);
    $currentPage = 0;
    foreach ($files as $page => $file){
        $currentPage++;
        if ($currentPage % $fixedPageSize === 0) {
            $logger->debug(sprintf('Splitting page %s due to fixed page size %s',
$currentPage, $fixedPageSize), $logContext);
            $splits[$currentPage][] = [
                'page' => $currentPage,
                'type' => 'FixedPageSplit',
                'value' => 'FixedPageSplit-' . $fixedPageSize,
                'fields' => $fieldValuesToKeep,
                'suppressOcr' => false
            ];
        }
    }
    return new xReturnObject(true,200, 'Fixed Page Split', $splits);
}

/**
 * Splits the given xDoc into multiple parts based on barcode information.
 *
 * @param xDoc $xDoc The xDoc object to be split.
 * @param LoggerInterface $logger The logger object for logging debug information.
 * @param string $splitBarcodeType The types of barcodes to split on, separated by semicolons.
 * @param string $splitBarcodePattern The pattern to match barcodes against.
 * @return xReturnObject The xReturnObject containing the split parts of the xDoc object.
 */
function splitByBarcode(xDoc $xDoc, LoggerInterface $logger, string $splitBarcodeType, string
$splitBarcodePattern): xReturnObject
{

    // Define which field values to keep
    $fieldNamesToKeep = [];
    $fieldValuesToKeep = getFieldValuesToKeep($xDoc, $fieldNamesToKeep);
```

```
    // initialize
    $logContext = [Logging::CTX_DOCUMENT_ID => $xDoc->getId()];
    $currentSplit = 0;
    $splits = [];
    $documentClassId = 0;
    $barcodeType = '';
    $barcodeValue = '';

    foreach ($xDoc->getBarcodes() as $page => $barcodes) {
        $isSplitPage = false;
        $suppressOCR = 0; // used to suppress OCR at desired page
        $logger->debug(sprintf('Checking barcodes for page %s', $page), $logContext);

        if (count($barcodes) == 0) {
            $logger->debug(sprintf('No barcodes found on page %s', $page), $logContext);
            if ($page == 1) {
                $currentSplit = $page;
                $logger->debug(sprintf('Start split at page %s', $page), $logContext);
            }
        } else {
            foreach ($barcodes as $barcode) {
                if (isMatchingBarcode($splitBarcodeType, $splitBarcodePattern,
$barcode["value"], $barcode["type"])) {
                    $currentSplit = $page - 1;
                    $barcodeType = $barcode['type'] ?? '';
                    $barcodeValue = $barcode['value'] ?? '';
                    $documentClassId = getDocumentClassIdByName($barcodeValue);
                    $isSplitPage = true;
                    $logger->debug(
                        sprintf(
                            'Split at page %s - Barcode "%s" of type "%s" found - Use document
class id %s',
                            $page,
                            $barcode["value"],
                            $barcode["type"],
                            $documentClassId
                        ),
                        $logContext
                    );
                    break;
```

```
                }
            }
        }

        // dismiss split pages
        if ($isSplitPage) {
            $logger->debug(sprintf("Barcode split page %s will be skipped", $page),
$logContext);
        } else {
            $splits[$currentSplit][] = [
                'page' => $page,
                'type' => $barcodeType,
                'value' => $barcodeValue,
                'fields' => $fieldValuesToKeep,
                'suppressOcr' => $suppressOCR,
                "documentClassId" => $documentClassId
            ];
        }
    }

    return new xReturnObject(true, 200, 'Barcode Split', $splits);
}

/**
 * Retrieves the values of the specified fields from the given xDoc object.
 *
 * @param xDoc $xDoc The xDoc from which to retrieve the field values.
 * @param array $fieldNames An array of field names to retrieve the values for.
 * @return array An array of associative arrays containing the field names and values.
 */
function getFieldValuesToKeep(xDoc $xDoc, array $fieldNames): array
{
    $fieldValues = [];
    foreach ($fieldNames as $fieldName) {
        if ($xDoc->getDocumentFieldByName($fieldName) !== null) {
            $fieldValue = $xDoc->getDocumentFieldByName($fieldName)->getValue();
            if (!is_null($fieldValue)) {
                $fieldValues[] = ['name' => $fieldName, 'value' => $fieldValue->getValue()];
            }
        }
    }
```

```
    return $fieldValues;
}

/**
 * Retrieves the document class ID by name.
 *
 * @param string $name The name of the document class.
 * @return int The ID of the document class.
 */
function getDocumentClassIdByName(string $name): int
{
    $documentClass = new xDocumentClass();
    $documentClass->getByName($name);
    if (is_null($documentClass->getId())) {
        return 0;
    }
    return (int)$documentClass->getId();
}

/**
 * Checks if a given barcode matches the specified barcode type and pattern.
 *
 * @param string $splitBarcodeType The type of the barcode to match.
 * @param string $splitBarcodePattern The pattern to match against the barcode value.
 * @param string $barcodeValue The value of the barcode to match.
 * @param string $barcodeType The type of the barcode to match against.
 * @return bool Returns true if the barcode matches the specified type and pattern, false
otherwise.
 */
function isMatchingBarcode(string $splitBarcodeType, string $splitBarcodePattern, string
$barcodeValue, string $barcodeType): bool
{
    if (strtolower($splitBarcodeType) === strtolower($barcodeType)) {
        $matches = null;
        xTools::mb_preg_match_all("/$splitBarcodePattern/i", $barcodeValue, $matches,
PREG_OFFSET_CAPTURE);
        $hit = $matches[0];
        foreach ($hit as $match) {
            if ($match[0] == '') {
                continue;
```

```
            }
            return true;
        }
    }
    return false;
}
```

Revision #7
Created 2023-10-30 15:34:00 UTC by Maximillian Weitze
Updated 2023-11-01 12:00:03 UTC by Maximillian Weitze