ContentServer 2

1 Einleitung

Der ContentServer 2 verfolgt die Idee, den bisherigen monolithischen ContentServer in verschiedene wieder verwendbare Komponenten aufzuteilen:

- ContentLib: Eine grundlegende Bibliothek für einfache Bild- (drehen, skalieren, Formatkonversion) und PDF Operationen (Bilder zu PDF, Metadatenanreicherung).
- ContentServlet(s): Servlets, die diese Funktionen als Webdienst bereitstellt (kann z.B. in späteren Stadien einfach für den Zugriffsschutz erweitert werden).
- ContentCache: Eine Komponente, die die Zwischenspeicherung übernimmt.
- ContentFrontend: Ein Typo3 Plugin, dass in der Lage ist die Servlets zu „bedienen“, es ist ebenfalls für die Authentifizierung und das Zwischenspeichern zuständig.

Das hat unterschiedliche Vorteile:

- durch die Bibliothek wird Dopplung von Code vermieden.
- Das Servlet kann als PDF Export in Goobi.Flow eingesetzt werden.
- Der modulare Aufbau ermöglicht eine Nachnutzung in auftragsspezifischen Implementierungen.

Ausgangsbasis ist eine Modifizierte Variante des ContentServers von Markus Enders (Java 5, Logger, Eclipse Codestyle, explizite Imports usw.), sowie einige Fragmente, die im Rahmen des TIB Auftrags implementiert wurden.

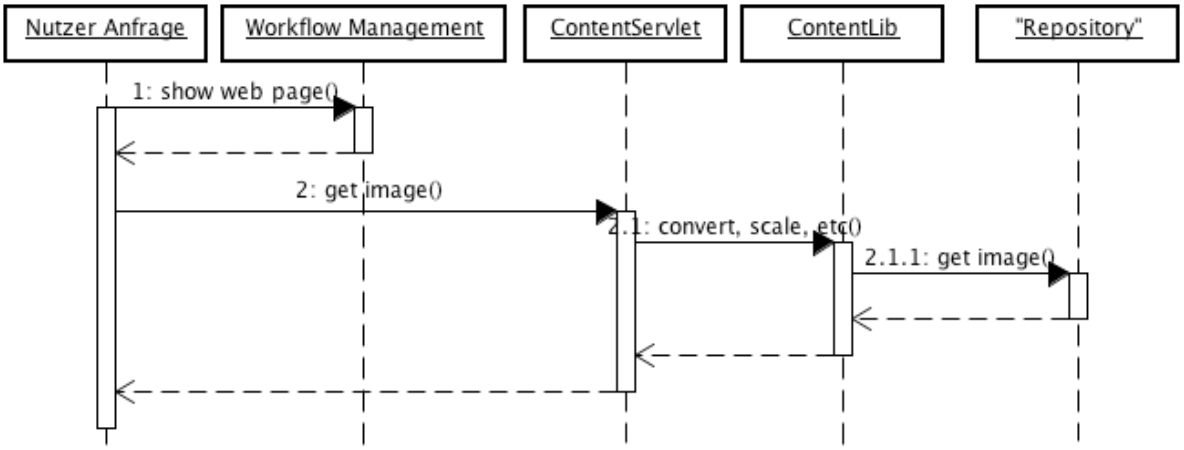
Um einer möglichst flexible und auf Durchsatz optimierte Implementierung zu erhalten müssen bei der Entwicklung folgenden Prinzipien eingehalten werden:

Die Übergabe von Speicherorten benötigter Ressourcen erfolgt ausschließlich über URLs.

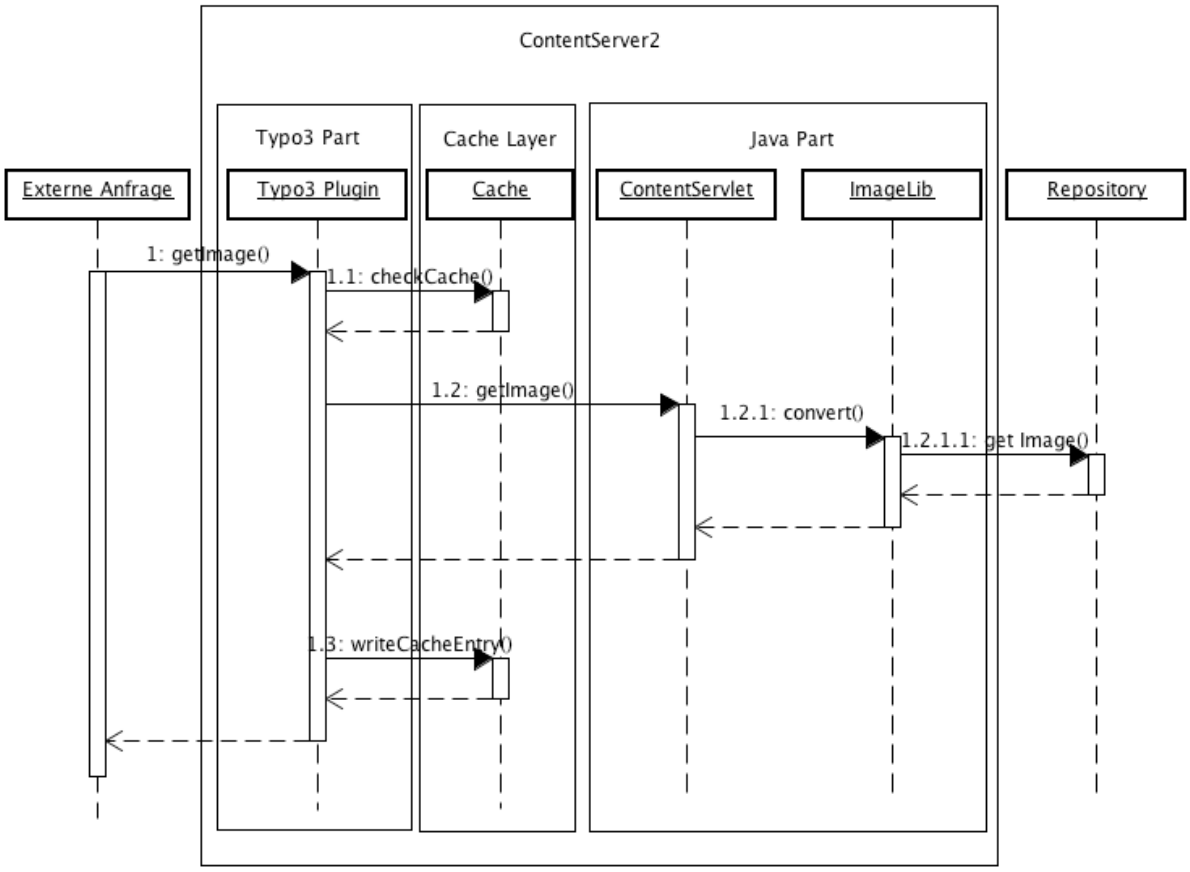
Das interne Handling größerer Dateimengen erfolgt ausschließlich als Datenstrom (Stream).

1.1 Schematische Abläufe

1.1.1 Nutzung des Contentservlet aus der Worklowkomponente



1.1.2 Nutzung des GoobiContentServlets aus der Präsentation für die Erstellung eines einzelnen Bildes oder einer PDF Datei



2 Komponenten

2.1 ContentLib

Hier werden Teile des bestehenden ContentServers (und ggf. andere bestehender Implementierungen wie für den TIB Auftrag nachgenutzt). Diese Komponente hat zwei Subkomponenten:

2.1.1 Bildmanipulation

Die Bibliothek soll in der Lage sein folgende Tätigkeiten auszuführen:

- Drehen des Bildes in einer (ganzzahligen) Gradzahl (Nachnutzung aus TIB).
  - Es werden Konstanten für 90°, 180° und 270° bereitgestellt
- Skalieren mit verschieden Parametern.
  - Breite in Pixel (Beibehaltung des Seitenverhältnisses)
  - Höhe in Pixel (Beibehaltung des Seitenverhältnisses)
  - Anpassen auf (Pixel-) Box (Beibehaltung des Seitenverhältnisses)
  - Prozent
- Auflösung (DPI) anpassen.
  - Es werden Konstanten für die gebräuchlisten Varianten vorgehalten (z.B. DISPLAY, PRINT, COLOR und BITONAL)
- Konversion in unterschiedliche Bildformate.
  - Es werden Konstanten für die unterschiedlichen Formate bereitgehalten
- Rechtecke ins Bild zeichnen, notwendig für das Hervorheben von Treffern.
  - Ein Polygon
  - Ein Rechteck
  - Eine Liste von Polygonen
- Aneinanderfügen von Bildern
  - Die Bibliothek muss in der Lage sein zwei Bilder aneinander zu fügen. Dabei lassen sich die Position, ggf. nötige Auffüllung (inklusive Farbe für den Füllraum) bzw. Skalierung als Parameter übergeben.
- Text in ein Bild schreiben
  - Für z.B. URLs in Fußzeilen ist es Notwendig, das die Bibliothek Text in Bilder schreiben kann. Als Parameter lassen sich dabei die Farbe, Schriftart und -größe, Koordinaten und natürlich der Text selber übergeben.

Die Farbe und Stärke der Füllung der Rechtecke sind über die API zu bestimmen.

Die Bibliothek soll folgende Bildformate bearbeiten bzw. ineinander Konvertieren können:

- JPEG
- PNG
- TIFF (mit verschiedenen Kompressionsverfahren: JPEG, FAX G4, LZW)
- JPEG2000

Bilder werden als RenderedImage übergeben. Eine statische Methode für die Konvertierung in ein BufferedImage wird bereitgestellt.

Der für die Skalierung verwendete Algorithmus soll konfigurierbar sein, es werden entsprechende Konstanten bereitgestellt. Für unterschiedliche Farbtiefen gibt es entsprechend angepasste Vorgaben.

Die Funktionalität muss (ausschließlich) auf Basis von JAI (bzw. ImageIO) implementiert sein.

Die Bibliothek sollte bei Fehlern entsprechende Ausnahmen auslösen.

2.1.2 PDF Erstellung

- Erstellen von PDF Dateien aus Bildern (Formate s.o.) und PDF Dateien Dabei müssen z.B. in den PDF Dateien enthaltene Volltexte erhalten bleiben.
- Metadaten (Nachnutzung aus TIB) manipulierbar.
  Folgende MPDF Metadatenfelder müssen direkt manipulierbar sein. Alternativ muss eine Methode diese Daten auf Basis einer Map (Groß/Kleinschreibung wird für die Schlüssel ignoriert) schreiben können.
  - Creator
  - Author
  - Title
  - Subject
  - Keywords

  Formate.
  - XMP
  - PDF „klassisch“
- Seitenzählung manipulierbar.
  Dazu muss eine Struktur Map<Integer, String> (physische Bildnummer, Seitenbezeichner) ausreichen.
  Bei der Zusammenführung mehrer PDF Dateien ist die Seitenbezeichnung zu erhalten.
- Anreichen mit Strukturdaten
- Für den Austausch von Strukturdaten (Bookmarks) ist eine eigene Klasse (eine Serialisierung ist in der vorliegenden Version nicht notwendig) zu definieren.
- "Wasserzeichen" konfigurierbar.
- Titelseiten Konfiguration (XML), hier sollte die bestehende Implementierung aus dem ContentServer nachgenutzt werden.
- Bei Zusammenführung bestehender PDF Dateien sind einige Dinge zu beachten:
  Die Serielle Abarbeitung von mehreren PDF Dokumenten nacheinander sorgt dafür, das die verschiedenen Typen Metadaten nicht vorher sinnvoll zusammengefasst werden können. Das sollte kein Problem sein solange sie für das Gesamtdokument vor dem Start der Verarbeitung angegeben wurden. Falls das nicht geschehen ist, werden normalerweise die entsprechenden Daten des ersten Dokuments übernommen (soweit dies möglich ist). Ein alternatives Verhalten, das entfernen der Metadaten, auch für die erste Seite, ist konfigurierbar.

Die Bibliothek soll folgende PDF „Typen“ unterstützen:

- PDF (1.5)
- PDF/A[1]

Für den gewünschten Ausgabetyp werden Konstanten bereitgehalten.

[1] PDF/A mit iText
http://blog.rubypdf.com/2007/08/10/itext-and-itextsharp-support-pdfa-1-now/

Um eine die Rückgabe als Stream zu realisieren, müssen die Metadaten vor der eigentlichen Verarbeitung bereit stehen. (andere „Aufrufreihenfolgen“ lösen eine „IllegalStateException“ aus).

Für PDF/A muss ein Farbprofil übergeben werden können, es ist zusätzlich eines Hinterlegt, falls keines übergeben wurde.

Die Übergabe von Informationen über Strukturdaten erfolgt über ein „Bookmark“ Objekt, nicht in der Form, die von der iText Bibliothek vorgegeben wird. Dadurch ist es einfacher möglich später ggf. eine andere Bibliothek zu nutzen.

Die Funktionalität muss auf Basis von iText implementiert sein.

Beide Subkomponenten müssen in der Lage sein mit Streams und / oder URLs zu arbeiten. Es werden jeweils zum Speichern Methoden bereitgehalten, die einen Dateinamen als Parameter annehmen. Die Implementierung ist ausschließlich über die API konfigurierbar (z.B Skalierungsalgorythmus).

Die Bibliothek sollte zwei Fehlerbehandlungsmodi besitzen: Entweder wird bei Fehlern mit „darunterliegenden“ Bildern ein Fehlerbild eingebunden oder die angeforderte Bearbeitung mit einem Fehler quittiert.

2.2 ContentServlet(s)

Das ContentServlet ist primär eine Webschnittstelle für die Bildmanipulation, es besteht aus zwei Schichten, die zweite Schicht ist von der ersten abgeleitet:

- ContentServlet: Die erste Schicht kann nur Operation auf Basis von URLs ausführen und braucht keine Kenntnis über seine Umgebung zu haben. Dieser Teil wird in eine zukünftigen Version der Workflowkomponente eingebettet, um Dopplung an Quellcode zu vermeiden.
- GoobiContentServlet: Die zweite Schicht ist eine Spezialisierung für den Goobi.Presentation Kontext.

2.2.1 ContentServlet

Das ContentServlet ist eine Web API für die ImageLib. Damit lassen sich die durch die ContentLib bereitgestellten Funktionalitäten dezentral über das Web nutzen. Das Ergebnis der Operation wird über HTTP zurückgegeben, im HTTP Header ist der MimeType entsprechend zu setzen. Auf Wunsch (Parameter) wird ein Vorschlag für den Dateinamen im HTTP Header gemacht. Das ContentServlet (und somit die Ableitungen) verfügen über einen Echo Mechanismus. Dieser gibt lediglich seine Argumente zurück. Das ContentServlet (und somit das GoobiContentServlet) muss, sofern Möglich, in der Lage sein mehrere CPUs bzw. CPU Kerne zu nutzen.

2.2.2 GoobiContentServlet

Das GoobiContentServlet stellt ein Spezialisierung des ContentServlets da, es ist auf die Anforderungen im Goobi.Presentation Kontext angepasst. Zu diesem Zweck verfügt es über folgende zusätzlich Fähigkeiten:

- METS Dateien können auf Basis einer URL eingelesen werden
- METS Dateien werden (mit Hilfe der von METS Beans) interpretiert.
- Beliebige (sinnvolle) Unterstrukturen können mit Hilfe einer ID aus einer METS Datei extrahiert werden.
- Pfade zu Bildern können aus METS ausgelesen werden, relative Angaben werden auf Basis der URL der METS Datei aufgelöst.
- Titelseite auf Basis einer XML Konfiguration (die via URL anzugeben ist) werden erstellt (Nachnutzung aus dem bestehenden ContentServer)
- Generierung von Bildbeschriftungen für Bilder auf Basis einer erweiterten Konfiguration für die Titelseiten.
- Sämtliche Bild- und PDF Operationen können auf den aus einer METS Datei extrahierten Informationen angewendet werden.
- Es ist in der Lage Anfrage nach dem Strukturdatentyp zu filtern.
- Alle Operationen können auf Basis mehrerer METS Dateien in einer Anfrage abgearbeitet werden, dabei ist die Anzahl der zu verarbeitenden Dateien pro Anfrage nicht beschränkt.

Der Aufruf durch das ContentFrontend erfolgt mit folgenden Parametern:

- URL(s) zur METS Datei(en).
- Identifier des bzw. der gewünschten Strukturelemente innerhalb der METS Datei, falls nicht vorhanden ist das gesamte Dokument(e) angefordert, als Identifier werden hier die XML IDs der „ “ elemente verwendet.
- Dateigruppe („METS:fileGrp“) aus der die Quelldaten zu entnehmen sind (optional, Vorgabe für „USER“ ist konfigurierbar s.u.)
- URL zur XML Datei, die die Titelseite beschreibt (optional).

Das Ergebnis wird mit einem Vorschlag für einen Dateinamen zurückgegeben. Das Format ist dabei für PDFs „Identifier-IDStruktureinheit.pdf“, falls eine Struktureinheit angefordert wurde, ansonsten „ Identifier.pdf“. Bei Bildern ist der Name „ Identifier-Bildnummer.FormatEndung“.

Folgende Parameter sind konfigurierbar:

- Wasserzeichen und Titelseite (unabhängig von Titelseite, die als URL übergeben werden kann)
- Fehlerbilder (Sonstiger Fehler)
- Dateigruppe falls keine übergeben wurde.

Zugriffsbeschränkungen für beide Servlets können über den Servlet Container oder eine Apache Anbindung konfiguriert werden. Die Überprüfung ob eine Struktureinheit überhaupt abgerufen werden darf ist Aufgabe des ContentFrontends.

2.3 ContentCache

Der ContentCache ist ein generischer Zwischenspeicher. Die Realisierung sollte ähnlich eines Proxy Servers erfolgen: Die Anfrage wird an die Zieladresse weitergeleitet, der Rückgabedatenstrom „gespalten“. Ein „Zweig“ geht dabei direkt weite an den anfragenden Browser, der andere wird gespeichert.

Der ContentCache ist als „dummer“ Zwischenspeicher gedacht, er verfügt nur über wenige administrative Funktionalitäten:

Durch einen zusätzlichen Übergabeparameter kann er veranlasst werden ein Objekt neu anzufordern, die bisherige Version im Zwischenspeicher wird dabei gelöscht.

Bei Fehlern sendet er eine entsprechende Nachricht im angeforderten Format.

Es muss eine Speicherbegrenzung konfigurierbar sein. Falls diese Grenze überschritten ist, wird das Speichern (nicht aber die Funktionalität) eingestellt

Da in der beschriebenen Ausbaustufe das Management des Cache extern erfolgen kann, muss eine gewisse Fehlertoleranz vorhanden sein. Das Löschen von Inhalten aus dem Zwischenspeicher zur Laufzeit sollte keine negativen Seiteneffekte haben.

2.4 ContentFrontend

Das ContentFrontend ist ein Typo3 Plugin, das die Nutzung der ContentServlets aus der Goobi.Presentation erlaubt. Das Frontend ist als Typo3 Plugin zu realisieren, dies ist Aufgabe der SLUB Dresden.

3 Umsetzung

Der in diesem Dokument beschriebenen Komponenten werden in zwei Phasen realisiert. Dabei umfasst die erste Phase folgende Funktionalitäten:

- einfache Bildmanipulationen (Skalierung, drehen, Rechecke hinein zeichnen, Formatkonversionen)
- Erstellung von PDF Dateien (in der ersten Phase nur aus Bildern, kein PDF/A, keine XMP Metadaten)
- Auswerten von METS Datei

Im Rahmen von Phase zwei werden die verbleibenden beschriebenen Funktionen implementiert

4 Konventionen

4.1 Namensräume

4.1.1 ContentLib

de.unigoettingen.sub.gdz.util.contentlib

4.1.2 ContentServlet

de.unigoettingen.sub.gdz.util.contentServlet

4.1.3 GoobiContentServlet

de.unigoettingen.sub.gdz.goobi.contentServlet

4.2 Kodierungskonventionen

4.2.1 Java

Es ist der Eclipse Codestyle zu benutzen. Zusätzlich gelten folgende Regelungen:

- Es ist (mindestens) Java 5 zu benutzen (Generische Datentypen, „for each“, usw.)
- Wildcard Importe sind nicht zulässig.
- Nach Möglichkeit sind „for each“ Konstrukte „nackten“ Iteratoren vorzuziehen.
- Bei Über- bzw. Rückgabe sollten, wenn möglich, Interfaces benutzt werden (also z.B. List<?> statt ArrayList<?>).
- Über- bzw. Rückgabe dürfen keine primitiven Datentypen sein.
- Auch wenn ein Block nur aus einer Zeile besteht, darf auf geschweifte Klammern nicht verzichtet werden.
- Methoden müssen kurz im Java Doc Stil dokumentiert werden (zumindest Parameter, Rückgabewert)
- Primitive Arrays (z.B. „File[]“) sind nicht erlaubt.
- Variablen dürfen nicht vor ihrem Gültigkeitsraum (Scope) definiert werden.
- Der Garbage Collector darf nicht explizit aufgerufen werden.
- Die JVM darf nicht explizit beendet werden.
- Für Meldungen muss Log4J benutzt werden, System.(out|err).* ist nicht erlaubt.
- Beim überladen von Methoden sind Annotations zu verwenden (z.B. „@Override“).

4.3 Lizenzen (sofern anwendbar)

Nach Möglichkeit sind freie Lizenzen (GPL, LGPL, Apache bzw. BSD Lizenzen zu vergeben). Für Bibliotheken ist die LGP der GPL vorzuziehen. Jede Datei, die Quellkode enthält muss einen Lizenz Header haben.