

arago DocMe Information Modelling

**arago Institut für komplexes
Datenmanagement AG**

Erstellt:

Version 1.0.3

Autor(en):

Hans-Christian Boos
Peter Steffan





Inhaltsverzeichnis

- 1 EINLEITUNG 4**
- 2 OBJEKTORIENTIERUNG (OO) 6**
 - 2.1 ELEMENTE DER OBJEKTORIENTIERUNG 6
 - 2.1.1 Objekte..... 6
 - 2.1.2 Klassen 7
 - 2.1.3 Attribute 7
 - 2.1.4 Klassenstruktur 8
 - 2.1.5 Vererbung 8
 - 2.1.6 Sammlung..... 8
 - 2.2 NUTZEN AUS DER OO..... 9
 - 2.2.1 Struktur 9
 - 2.2.2 Erweiterbarkeit 9
 - 2.2.3 Minimierung des Pflegeaufwands 9
- 3 CONTENT POOL STRUKTUR..... 10**
 - 3.1 FACHLICHE ANFORDERUNGEN 10
 - 3.2 PRAKTISCHE INFORMATIONSERSTELLUNG 11
 - 3.2.1 Klassifizierung des Content..... 11
 - 3.2.2 Informationstypisierung des Content 12
 - 3.3 INFORMATIONSSCHNITTSTELLEN 15
 - 3.4 TECHNISCHE ASPEKTE 15
- 4 CONTENT ZIELSYSTEME..... 18**
 - 4.1 ZWECK DER ZIELSYSTEME 18
 - 4.2 STRUKTUR DER ZIELSYSTEME 18
 - 4.3 WELCHER CONTENT WIRD BENÖTIGT? 19
 - 4.4 WELCHER CONTENT AUS DEM CONTENT POOL? 19
 - 4.4.1 Qualifikation des Content 19
 - 4.4.2 Attribut und Strukturanpassung 19
- 5 ZIELSYSTEME CONTENTVERWENDUNG..... 21**
 - 5.1 BASISLAYOUT DER ZIELSYSTEME 21
 - 5.1.1 Basislayoutcode 21
 - 5.1.2 Attributzuordnung – Mapping..... 22
 - 5.1.3 Defaults 22
 - 5.2 CONTENT BEHANDLUNG 23
 - 5.2.1 Formatkonvertierung 23
 - 5.2.2 Storage..... 24
 - 5.2.3 StyleMe 25
 - 5.2.4 Hooks 26
 - 5.2.5 Klassentypen 28
 - 5.2.6 Aufnahmegrenzen 29
 - 5.3 CONTENT EINBINDEN MIT DOCME STANDARDS - INDICES 29
 - 5.3.1 Einfache Indices 33
 - 5.3.2 Komplexe Indices 33
 - 5.3.3 Umfassende Indices..... 34



Einleitung

- 5.3.4 Direktanzeige 34
- 5.4 CONTENT EINBINDEN ERWEITERT 34
 - 5.4.1 Hook Vererbung 34
 - 5.4.2 Download verschiedener Formate 35
- 5.5 CONTENT IN APPLIKATIONEN 36
 - 5.5.1 Indices 37
 - 5.5.2 Navigation 37
 - 5.5.3 Sitemap 37
 - 5.5.4 Adverstimentboxen 37
 - 5.5.5 Suchmaschinen 38
 - 5.5.6 Personalisierung 38
 - 5.5.7 Newsletter 39
 - 5.5.8 DB Zugriff 39
 - 5.5.9 Server Side Include (SSI) 39
 - 5.5.10 Caching 39
- 6 CONTENT VEREDELUNG 40**
 - 6.1 STYLEME 40
 - 6.2 LINKAGE 40
 - 6.2.1 Keywords 40
 - 6.2.2 Attribute 41
 - 6.2.3 Attachmanagement 41
 - 6.2.4 Linkmanagement 41
 - 6.3 VARIABLEN 42
 - 6.3.1 Einfache Variablen 42
 - 6.3.2 SQL Variablen 42
 - 6.3.3 LOADFILE 43
 - 6.3.4 JAVASCRIPT 43
- 7 REGELWERKE 45**
 - 7.1 CONTENT POOL REGELN 45
 - 7.2 CONTENT POOL – CONTENT POOL REGELN 45
 - 7.3 INDEX REGELN 46
- 8 ANHANG 48**
 - 8.1 VARIABLEN 48
 - 8.1.1 Klassenvariablen 48
 - 8.1.2 Indexvariablen 50
 - 8.2 HOOKS 55
 - 8.3 HOOKS 55
 - 8.3.1 Class Tree Hooks 55
 - 8.3.2 Index Hooks (Modul Indices) 60
 - 8.3.3 Globalhooks (Modul Basic Opitons) 60
- 9 LITERATUR 61**



1 Einleitung

Informationsmodellierung ist ein Ansatz zur Erfassung aller vorhandenen und benötigten Informationen, deren Ausprägung, Änderungsfrequenz, Lifecycle, Relevanz sowie die zur Dokumentation der benötigten Darstellungsarten, -formen und Zeiten für jeden vorhandenen Informationstyp des Content Management System (CMS). Dabei wird auf größere oder kleinere Entitäten innerhalb des Unternehmens ein Modell erstellt, dem folgende Frage zu Grunde liegt:

“**Wer** produziert **welche** Informationen, und **wer** will diese dann **wann, wo** und **wie** sehen.“

Die vorliegende Ausarbeitung soll dabei die zur praktischen Umsetzung in einem DocMe Projekt notwendigen theoretischen Grundlagen vermitteln.

Nun gibt es sicher viele Wege sich der Beantwortung dieser Fragen zu nähern. Das vorliegende Werk beschreibt einen Weg. Die behandelten Themen sind in einer oft benutzten Vorgehensweise angeordnet, können jedoch teilweise projektspezifisch individuelle arrangiert und in der benötigten Tiefe genutzt werden.

Wesentlich ist aber, dass man sich die Frage stellt, diese hinreichend genau beantwortet, die Ergebnisse auf Grund der Komplexität notiert, visualisiert und vor allem reflektiert.

Je detaillierter dabei die Kenntnisse der verfügbaren Werkzeuge zur Modellierung ist, desto besser und flexibler wird das resultierende Informationsmodell und davon direkt abhängig die Qualität der Integration, der Umsetzung, der Akzeptanz und der Erweiterbarkeit des gesamten Vorhabens.

Organisation

Leider gibt es keine für alle Vorhaben allgemeingültige Empfehlung zur Tiefe, in der ein Informationsmodell zu Erstellen ist. Es gilt aber folgende Abhängigkeit:

Je umfangreicher, verteilter und diversifizierter ein Vorhaben zur Informationsverwaltung, -verteilung und -gestaltung ist und werden soll, desto komplexer wird auch das zu erstellende Informationsmodell. Damit steigt auch das Risiko, bei nicht hinreichender Tiefenbetrachtung Strukturfehler zu implementieren, die sich in höheren Wartungsaufwänden, unflexiblen Systemen und letztlich in laufenden und fixen Kosten ausdrücken.

Leider kann aber auch eine zu tiefe Modellierung, die meist im Bestreben



DocMe Information Modelling

Einleitung

mündet technische Lösungen um jeden Preis zu kreieren, zu unnötiger Komplexität, verminderter Transparenz und letztlich wieder zu höheren laufenden und fixen Kosten führen.

Eine angemessenen Modellierungstiefe, verbunden mit dem Bestreben möglichst Standards zu schaffen und von Standards abweichende Prozesse einer besonderes kritischen Hinterfragung, insbesondere die Kosten/Nutzenrelation betreffend, zu unterziehen führen zu guten Lösungen.

Hinweis: Prüfen Sie daher, insbesondere bei vom Standards abweichenden Prozessen, ob eine organisatorische Lösung nicht den gleichen Effekt, bei weit geringeren Kosten, erzeugen kann.



2 Objektorientierung (OO)

Die OO bietet über die Definition von Strukturen, Klassen, Objekten und deren Vererbungsregeln die Möglichkeit selbst komplexe Strukturen mit geringem Aufwand zu verwalten und anzupassen. Dabei bleibt die Möglichkeit der Individualisierung der Objekte über die Attribute bestehen.

In Fachkreisen besteht jedoch seit Jahren ein Glaubenskrieg über das Für und Wider zum OO Ansatz, welcher sich in den folgenden kurz beschriebenen Grundgedanken ausdrückt:

- **Systeme ohne OO**
...sind stärker individualisierbar und ggf. schneller einführbar.
Probleme entstehen meist bei wachsendem Systemumfang.
- **Systeme mit OO**
... sind schnell erweiterbar und minimieren die Systempflege.
Probleme entstehen meist bei starker Individualisierung.

Auf eine weitere Bewertung oder gar eine Zustimmung zu der einen oder anderen Methode wird hier verzichtet, es käme doch immer ein pro OO heraus.

DocMe lehnt sich stark an ein objektorientiertes Modell und spielt daher seine Stärken besonders gut aus bei einer die OO Herkunft von DocMe unterstützenden Grundlage – dem Informationsmodell.

Deshalb zunächst ein kurzer Überblick zu den oft verwendeten Begriffen aus der OO Welt.

2.1 Elemente der Objektorientierung

2.1.1 Objekte

Was ist ein Objekt?

- **Allgemein**
Ein Objekt ist immer einer bestimmten Klasse zugeordnet und stellt ein Exemplar (Instanz) dieser Klasse da. Ein Objekt erhält von der Klasse die Struktur der Attribute. Jedes Objekt verfügt über individuelle Werte der Attribute.
- **DocMe**
Ein Objekt ist ein DocMe Dokument. Es erhält von der Klasse die Struktur, die System- und Uservariablen. Die Variablen haben individuelle Werte.
Sehr oft enthält ein DocMe Dokument eine Hauptdatei und ggf.



weitere anhängende Dateien.

Info:

Natürlich sind z.B. auch Indices, Converter etc. Objekte. Damit aber die Lesbarkeit dieser Ausarbeitung nicht durch die allgemeine Verwendung des Begriffs „Objekt“ leidet, wird im weiteren bewusst auf eine exakte, rein theoretische Nutzung der OO Begriffe verzichtet.

Was ist eine Objektidentität?

- **Allgemein**
Die Objektidentität ist eine Objekteigenschaft welche Objekte selbst bei gleichen Attributwerten unterscheidet.
- **DocMe**
Die für jedes DocMe Dokument (Objekt) einzigartige Dokumentenidentifikationsnummer (DocId) stellt die Qualifikation selbst bei gleichen Attributwerten sicher.

2.1.2 Klassen

Was ist eine Klasse?

- **Allgemein**
Eine Klasse ist eine Zusammenfassung (Aggregat) von gleichartigen Objekten. Dabei beschreibt die Klasse die Eigenschaften dieser Objekte mit Attributen. Die Klasse selbst ist somit ein Metaobjekt. Eine Klasse erlaubt das Erzeugen von Objekten.
- **DocMe**
Eine Klasse fasst gleichartige DocMe Dokumente (Objekte) zusammen. Die Klasse beschreibt die Eigenschaft der Dokumente mit System- und Uservariablen.

2.1.3 Attribute

Was sind Klassenattribute?

- **Allgemein**
Werden pro Klasse definiert und stehen allen Objekten der Klasse zur Verfügung.
- **DocMe**
System- und Uservariablen welche jedem DocMe Dokument (Objekt) auf Grund der Klassenzugehörigkeit zur Verfügung gestellt werden.



DocMe Uservariablen können in beliebiger Zahl und Ausprägung selbst definiert werden.

2.1.4 Klassenstruktur

Was ist eine Oberklasse bzw. Unterklasse?

- **Allgemein**

Eine Oberklasse ist eine Verallgemeinerung ausgewählter Eigenschaften ihrer Unterklassen.

Eine Unterklasse ist die Spezialisierung einer Oberklasse und erbt alle Eigenschaften der Oberklasse.

- **DocMe:**

Eine Oberklasse hängt oberhalb einer Klasse oder eines Klassenverzeichnisses (Directory) im Klassenbaum und fasst meist mehrere gleichartige Unterklassen zusammen.

Eine Unterklasse hängt unterhalb eines Klassenverzeichnisses (Directory) im Klassenbaum und erbt die Struktur der Oberklasse.

2.1.5 Vererbung

Was ist Vererbung?

- **Allgemein**

Definiert die Regeln zur Vererbung von Eigenschaften, Attributen und Werten von Klassen.

- **DocMe**

System- und Uservariablen sowie deren Werte werden von der „Elternklasse“ an die „Kindklasse“ vererbt.

Typischerweise werden zur Vererbung dienende Klassen als sog. „Abstrakte Klassen“ definiert. „Abstrakte Klassen“ nehmen keine DocMe Dokumente (Objekte) auf, sondern dienen lediglich der Vererbung im Unterschied zu den sog. „Konkreten Klassen“

2.1.6 Sammlung

Was ist eine Sammlung

- **Allgemein**

Sammlungen sind Objekte welche über Regeln definierte Mengen an anderen Objekten referenzieren und Funktionen zum Zugriff auf die referenzierten Objekte bereitstellen.



- **DocMe**

Indices und Keywordverlinkung fassen über Regeln DocMe Dokumente (Objekte) zusammen und ermöglichen über die Erstellung von Indexpages einen Zugriff auf die DocMe Dokumente.

2.2 Nutzen aus der OO

2.2.1 Struktur

OO Systeme erlauben eine Strukturdefinition die sowohl über die hierarchische Baumstruktur, als auch über die Attribute erzeugt bzw. definiert werden kann.

Die Möglichkeit beide Verfahren gleichzeitig zu nutzen bietet die Grundlage in DocMe selbst komplexe Strukturen relativ einfach abbilden zu können ohne dabei auf eine einfache Erweiterbarkeit und Wartbarkeit zu verzichten.

2.2.2 Erweiterbarkeit

Innerhalb der bestehenden Struktur (Baumstruktur) können neue Klassen und weitere Strukturen nach Bedarf hinzugefügt werden ohne dabei die Funktionen der bestehenden Struktur mit ihren zugehörigen Objekten zu verändern.

Die Klassen können weiter nach ihrer Bedeutung und Bedarf detailliert werden, ohne die Funktionen von zur gleichen Oberklasse (Directory im Klassenbaum) gehörenden Klassen zu tangieren.

2.2.3 Minimierung des Pflegeaufwands

Eine gute Strukturierung garantiert nicht nur die spätere Erweiterbarkeit, sondern auch die Minimierung des Pflegeaufwands. Insbesondere die Nutzung der Vererbung senkt den ansonsten zu erwartenden Pflegaufwand erheblich.



3 Content Pool Struktur

Der Content Pool stellt die Summe aller von DocMe verwalteten Dokumente (Objekte) in einem DocMe Netzwerk dar.

Die Struktur, des Content Pools ist von essentieller Bedeutung und bestimmt im Wesentlichen die spätere Erweiterbarkeit und die Akzeptanz durch die Endanwender.

3.1 Fachliche Anforderungen

Zunächst unabhängig von jeder Technik, stehen die fachlichen Anforderungen für den Content Pool im Vordergrund. Dabei sollte man sich folgende Fragen stellen.

Welche Contentthemen Bereiche sollen im Content Pool bereitgestellt werden?

Welche Contentthemen Bereiche werden in Zukunft adressiert?
(Hier hilft weder eine sehr konservative noch eine euphorische Sichtweise wirklich weiter.)

Dabei ergeben die bisherigen Unternehmensprozesse eine Grundlage für die zu erstellende Content Pool Struktur. Der Content Pool soll ja den Unternehmensprozessen dienen indem er die Basis zur Informationsverteilung bildet..

Dies schließt die Möglichkeit zu einer Umstellung oder Anpassung des einen oder anderen Unternehmensprozesses zu diesem Zeitpunkt nicht aus. Oft gibt ja gerade die Einführung neuer Systeme die Chance auch bestehende Prozesse kritisch zu hinterfragen.

Sehr wichtig ist deshalb die Beteiligung der Fachabteilung und der Organisationsabteilung bei der Erstellung der Content Pool Struktur. Eine rein durch die IT betriebene Erstellung ist unbedingt zu vermeiden.

Hinweis: Es ist unbedingt zu vermeiden die Content Pool Struktur mit einer am Zielsystem erwarteten WebStruktur gleichzusetzen! Es liegt zunächst sehr nahe diesen Bezug herstellen zu wollen. Aber WebSites werden eher häufigen Umstrukturierungen unterworfen, die sicher nicht zu einer Umstrukturierung des gesamten Informationsmodells führen sollen!



3.2 Praktische Informationserstellung

Ziel der praktischen Informationserstellung ist ein erster grober Entwurf der Content Pool Struktur, auch Klassendiagramm genannt.

Dieser grobe Entwurf wird nach der später folgenden Betrachtung der Zielsysteme einer Überprüfung und Anpassung unterzogen. So nähert sich die Content Pool Struktur iterativ der dann umzusetzenden Struktur.

Zur Erstellung des ersten groben Entwurfs dienen meist Tabellen in denen die folgenden Punkte berücksichtigt werden.

3.2.1 Klassifizierung des Content

Zu Beginn steht eine Sammlung möglichst aller Contentthemen, die dem Content Pool zur Verfügung gestellt werden sollen.

Zur Findung der ersten Gliederungskriterien stellt man sich die folgenden Fragen:

Gibt es Content mit gleichen Grundvoraussetzungen?

Gibt es Content mit gleichen Eigenschaften?

Gibt es Content der immer nur zu einem Thema gehört?

Die so erhaltenen Klassen (Gruppierungen) haben alle die gleichen Eigenschaften. Unterklassen leiten sich davon ab und werden weiter spezialisiert.

Die Sammlung der Contentthemen wird auf diese Weise in eine erste hierarchische Baumstruktur überführt.

Die Detaillierung kann einerseits eine beliebige Tiefe erreichen, andererseits wirken sich aus der Praxis heraus Klassentiefen ab 4-6 Ebenen eher negativ auf die Akzeptanz der Autoren und Redakteure aus. Es ist an dieser Stelle aber noch nicht ausschlaggebend, ob die Detaillierung über die Struktur abgebildet wird, da eine spätere Zusammenfassung der Struktur über Metainformationen (Attribute) erfolgen kann.

Hinweis: Es bietet sich an, sich selbst dazu zu verpflichten, mindestens zwei grundsätzlich verschiedene Ansatzmodelle anzufertigen um sicher zu gehen, dass die Vor- und Nachteile nicht nur in sich, sondern auch gegenüber anderen Modellen überprüft werden können.

Hinweis: Zu diesem Zeitpunkt kann die spätere Verwendung des Content am Zielsystem völlig außer Acht gelassen werden.



Es besteht sonst die Gefahr, die Content Pool Struktur an die Bedürfnisse der Zielsysteme anzugleichen. Dies wäre aber nur in einem einzigen Fall sinnvoll und zwar, wenn es auch in Zukunft nur ein einziges Zielsystem gäbe und dieses von möglichen Veränderungen in der Zukunft nicht berührt würde.

Tip: Zur Unterstützung der Akzeptanz sollten die Namenswahl der Klassen denen der im Unternehmen gebräuchlichen Syntax angepasst werden.

3.2.2 Informationstypisierung des Content

Standard Attribute (Metainformationen)

Attribute beschreiben Objekte (Dokumente) zur weiteren Klassifizierung näher. Diese Beschreibungen dienen den Zuordnungen, Verarbeitungen und den benötigten Sichtweisen.

Es gibt allerdings einen Satz an Standard Metainformationen, der sich als ein quasi Minimumsatz herauskristallisiert hat. Diese Attribute sollten jeder Klasse zugeordnet werden und damit jedem Objekt zur Verfügung stehen. Die Attributwerte können dabei manuell und/oder automatisiert erzeugt werden.

- Titel
- Beschreibung
- Autor
- Email Adresse des Autor
- Erstellungsdatum
- Veröffentlichungsdatum
- Verfallsdatum

Wenn es Content- oder Contentbereiche übergreifende Standard Attribute gibt sollten diese zu diesem Zeitpunkt qualifiziert werden, da die auf dieser höchsten Ebene definierten Attribute viel Aufwand bei der Definition der einzelnen Klassen erspart und den Blick auf das Wesentliche, eben die wenigen individuellen Klassenattribute lenkt.

Die Standardattribute sind fast vollständig klassenunabhängig, sind aber zu weiteren Klassifizierung unabdingbar. Sie stellen den höchsten Grad der zu vererbenden Attributwerte dar und werden deshalb in einer [abstrakten Klasse](#) definiert von welcher alle anderen Klassen, ob abstrakt oder konkret, erben. Es gilt dabei die folgende Vererbungsregel:



- **Vererbungsregel:**
Ist ein Attributwert einer Parentklasse in der Parentklasse gesetzt, so wird er von der Childklasse übernommen. Ist in der Childklasse selbst ein Attributwert gesetzt, so hat dieser Vorrang. Diese Regel gilt über beliebig viele Vererbungsgenerationen hinweg.

Objektanzahltyp

Pro Klasse können beliebig viele Objekte (Dokumente) erzeugt werden. Es gibt aber auch Fälle in denen nur eine ganz bestimmte definierte Menge oder gar nur ein Objekt erzeugt werden kann.

- **Multi Dokumenten Klasse**
Ist die Standard Klasse und nimmt beliebig viele Objekte auf. Neu erzeugte Objekte sind immer zusätzliche Exemplare der Klasse.
- **Multi Dokumenten Klasse mit Replacekriterium**
Die Anzahl der Objekte richtet sich nach dem Replacekriterium. Objekte mit gleichem Replacekriterium werden ersetzt (es wird eine neue Version im Archiv erzeugt). Objekte mit noch nicht vorhandenem Replacekriterium bilden ein zusätzliches Exemplar der Klasse.
- **Single Dokumenten Klasse**
Erzeugt genau ein Objekt. Wird ein weiteres Objekt erzeugt, so wird das bestehende damit ersetzt (es wird eine neue Version im Archiv erzeugt) und es existiert wieder genau ein Objekt.

Den beiden letzten Typen kann ein sog. Main Link Argument mit übergeben werden, welches den dedizierten Zugriff auf ein solches Objekt gewährleistet. Dies trifft auch für die Multi Dokumenten Klasse zu, wobei eine Eindeutigkeit des späteren Zugriffs am Zielsystem zumindest nicht direkt gegeben ist.

Der Einsatz von Single Dokumenten Klassen sollte kritisch hinterfragt werden, da ein vermehrter Einsatz dieses Typs zwangsläufig die Struktur aufbläht.

Änderungsfrequenz

Die Änderungsfrequenz von Informationen wirkt sich nicht unbedingt sofort auf das Informationsmodell selbst aus, hat aber ggf. starke Auswirkungen auf die technische Verwendbarkeit von Informationen. Deshalb wird die Änderungsfrequenz hier als ein Kriterium der Informationstypisierung herangezogen.

Statische Informationen

Statische Informationen werden exakt einmal erstellt und haben dann für



ihre gesamte, gewöhnlich recht lange Lebenszeit volle Gültigkeit. Statische Informationen sind daher keine aktuellen Informationen und werden im Allgemeinen auf dynamische Prozesse innerhalb eines Unternehmens wenig Einfluss haben: sie können aber sehr wohl die Grundlage für solche Prozesse bilden, wenn es sich bei den Informationen z.B. um Verfahrensvorschriften handelt.

Dynamische Informationen

Dynamische Informationen sind das eigentlich exakte Gegenteil zu statischen Informationen. Dynamische Informationen haben ein sehr beschränktes Bedeutungsfeld und eine sehr kurze Lebensdauer, sie werden jedoch ständig durch aktuellere Informationen gleicher Bedeutung ersetzt: auf diese Weise stellt eine Reihe von dynamischen Informationen wieder eine statische oder semidynamische Information dar.

Semidynamische Informationen

Semidynamische Informationen haben eine lange Gültigkeit und werden im Gegensatz zu rein dynamischen Informationen nur selten komplett automatisch produziert. Dennoch werden von jenem semidynamischen Typ Informationen in gewissen Intervallen übermittelt, welche die bestehenden Informationen, ähnlich wie beim dynamischen Modell, ersetzen oder diese erweitern.

Der weitaus größte Teil der in einem Unternehmen produzierten Informationen fällt in diese Kategorie, da die Informationen immer wieder den gleichen definierten Typ haben und trotzdem entweder einen Ersatz oder eine Erweiterung des bereits bestehenden Informationsbestandes darstellen.

Freigabe

Die benötigten Freigabeprozesse haben zwar keinen direkten Einfluss auf das Informationsmodell, wohl aber einen starken Einfluss auf die technische Realisierung. Je nach Anforderungen bedingen die Freigabeprozesse technische Erweiterungen bis hin zur Integration von Workflow Systemen.

Hier erfolgt die Definition ob Content, z.B. wegen gesetzlicher Vorschriften, oder aus Haftungsgründen einem Freigabeprozess unterzogen werden muss.

4,6,8... Augen Prinzip

Erstellter Content muss von einem/mehreren weiteren, zur Freigabe berechtigten User freigegeben werden.

Individueller Workflow

Erstellter Content muss einen individuellen Weg der Freigabe, meist mit Eskalationsregeln versehenen Weg, durchlaufen.



3.3 Informationsschnittstellen

Die Informationsschnittstellen beschreiben die Quellen der Systeme, die Content an ein DocMe System liefern.

Dabei nimmt DocMe Content über die folgenden zwei grundsätzlich verschiedene Verfahren an.

Autoren

Informationen nehmen ihren Ursprung immer bei den Autoren. Die Autoren in ihrer gewohnten Umgebung zu belassen, aus der heraus sie publizieren können, ist Grundvoraussetzung für das Informationsmodell. Denn ein Autor soll seiner Kernkompetenz, dem Verfassen von Informationen nachkommen und sich nicht auf die IT oder die Informationsverteilungsprozesse konzentrieren.

Die Autoren nutzen als Schnittstellen die DocMe Client Systeme wie den DocMe Manager, den DocMe Java Publisher, den DocMe Administrator, sowie auf den DocMe Manager aufsetzende COM+ Anwendungen wie die DocMe Office Makros und Anwendungen und selbst erstellte Anwendungen.

Automatisierte Prozesse

Individuelle, zu automatisierende Prozesse zur Content Einstellung müssen definiert werden und können über die DocMe LDAP, File, XML, ICE oder Hook Schnittstellen realisiert und integriert werden.

3.4 Technische Aspekte

Zur Definition der technischen Content Pool Struktur dient die Frage

- **Wo** wird Content gebraucht?

Zur Definition der Zielsysteme, also der Informationskonsumenten und deren Aufbereitung, dient hingegen die Frage.

- **Wer** braucht Content **wie**?

Die Zielsysteme werden mit dem nächsten Kapitel beginnend behandelt.

Je nach geographischer bzw. sprachlicher oder inhaltlicher Verschiedenheit kann es sinnvoll sein, den Content Pool aufzusplitten und lediglich die Schnittmenge zwischen den Content Pools auszutauschen.

So haben international agierende Unternehmen oft eine gemischt sprachliche und geographische Umgebung. Kommen nun stark abweichende Content Themen hinzu, kann anstatt der Umsetzung einer sehr komplexen Unternehmensentität eine den



Content Pool Struktur

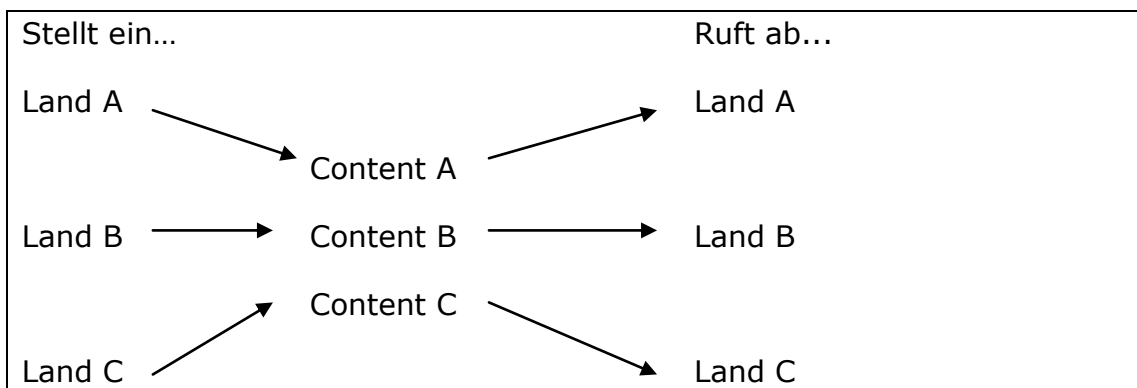
Unternehmensgegebenheiten entsprechend unterstützende Aufspaltung in kleinere Entitäten gewählt werden.

Indizien für eine Content Pool Aufspaltung zeigen sich über die Schnittmenge des Contentbedarfs und vor allem über die geringe Deckungsgleichheit der Teilstrukturen.

Beispiel:

Gegeben sei ein Konzern, der in drei Ländern getrennt die Fertigung von Komponenten betreibt. Dazu ergeben sich folgenden drei grundsätzlich verschiedenen Ansätze.

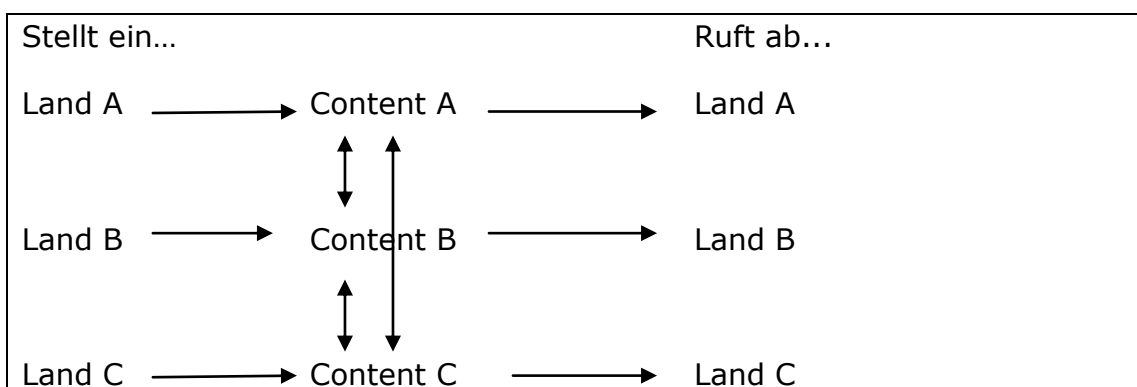
Zentraler Content Pool



Vorteil:
Ein zentraler Content Pool

Nachteil:
Von zwei Ländern muss der Content in das zentrale „Pool Land B“ verbracht werden und auch wieder von dort abgerufen werden.

Dezentrale Content Pools



Vorteil:
Der überwiegende landesspezifische Content bleibt im eigenen Land und

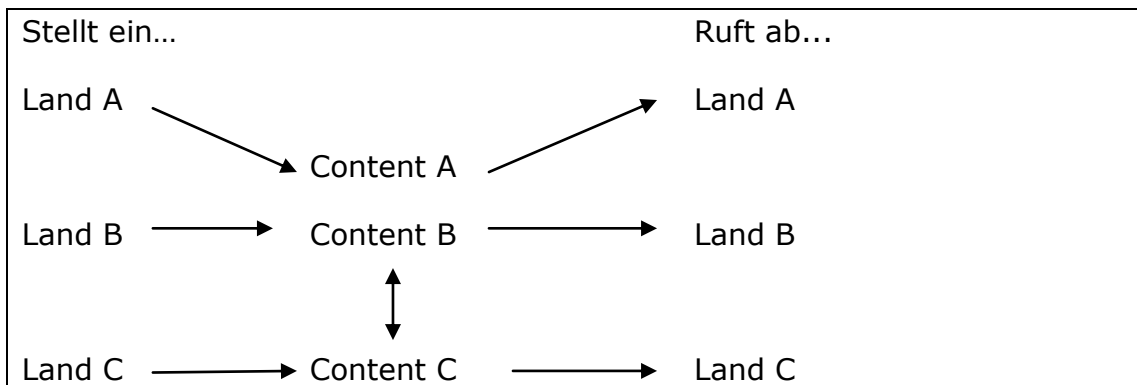


Content Pool Struktur

wird auch von dort abgerufen. Nur die in jedem Land benötigte Schnittmenge wird ausgetauscht.

Nachteil:
Drei Content Pools.

Verteilte Content Pools



Verteilte Content Pools stellen eine Mischform des zentralen und dezentralen Content Pools dar.



4 Content Zielsysteme

Der Content Pool kann gleichzeitig beliebig viele und im Zweck unterschiedliche Zielsysteme mit Informationen versorgen.

Besonders interessant ist dabei, die Mehrfachnutzung und Wiederverwendung des meist teuer erstellten oder erworbenen Content. So kann ein im Content Pool verfügbarer Content in verschiedenen Ausprägungen und gleichzeitig mehreren Zielsystem zur Verfügung gestellt werden.

Zielsysteme stellen aus DocMe Sicht die von den PublishD Servern belieferten Systeme dar, welche den Content ihrerseits zweckgebunden ausliefern.

4.1 Zweck der Zielsysteme

Jedes Zielsystem erfüllt immer einen bestimmten Zweck der Informationsverteilung. Dieser Zweck des Zielsystems bestimmt bereits auf einer Metaebene grob die später zu definierende Menge des Contents.

Beispiele für DocMe Zielsysteme:

- Web Server (Inter-, Extra- und Intranet)
- Web Anwendungen (z.B. Newsletter)
- Portal Server
- WAP Server
- Fax Server
- CD-ROM, DVD Server
- Video Server
- Print Server
- Individuelle Anwendungen

4.2 Struktur der Zielsysteme

In der Regel entspricht die Klassenstruktur des Zielsystems einer Untermenge des Content Pools.



Ausnahmen bestätigten diese Regel und sollten daher auch Ausnahmen bleiben. Mehr dazu unter [Cross Distribution](#).

4.3 Welcher Content wird benötigt?

Für jedes der Zielsysteme ist die Summe des benötigten Contents zu definieren. Dabei ist zunächst noch nicht von Bedeutung, ob der Content vom DocMe Content Pool oder von anderen Quellen stammt. Im Vordergrund steht die vollständige Erfassung und eine hinreichende Vorstellung der zeitlichen Entwicklung des benötigten Contents.

Die Behandlung des nicht vom DocMe Content Pool stammenden Contents kann hier natürlich nicht weiter erörtert werden. Es ist aber ratsam zu prüfen, ob der nicht via Content Pool bereitgestellte Content ebenfalls über DocMe bereitgestellt werden könnte um die Vorteile, z.B. Content Archivierung, allgemeiner Zugriff auf den Content u.s.w. nutzen zu können.

4.4 Welcher Content aus dem Content Pool?

4.4.1 Qualifikation des Content

Der am jeweiligen Zielsystem benötigte Content aus dem Content Pool muss qualifiziert werden. Als erstes Kriterium dient dazu der Grobentwurf der Klassenstruktur.

Aus der benötigten Menge leiten sich später die zu erstellenden Verteilregeln für den Content Pool ab.

4.4.2 Attribut und Strukturanpassung

Oft stellt man beim Versuch den benötigten Content zu qualifizieren fest, dass der gegebene Entwurf aus der Content Pool Grobstruktur nicht ausreicht um den benötigten Content klar und eindeutig für das Zielsystem zu qualifizieren.

Dann ist zu prüfen, ob die betroffene Klasse oder die Teilstruktur durch das Hinzufügen von weiter detaillierenden Attributen die gewünschte Qualifizierungsmöglichkeit erhalten kann, oder ob gar Teilbereiche der Klassenstruktur neu zu gestalten sind.

In jedem Fall hat die Überprüfung der Qualifizierbarkeit des Content am Zielsystem zur Folge, dass die Grobstruktur des Content Pools entweder bestätigt oder aber weiter detailliert, angepasst oder ergänzt wird. Je näher der Grobentwurf des Content Pools den Bedürfnissen kommt, desto eher ist anstatt mit Strukturänderungen am Content Pool, mit Anpassungen der Attribute zu rechnen.



Hinweis: Je nach Umfang der gesamten Struktur, wird der Focus ggf. mehrmals zwischen der [Definition des Content Pools](#) und der Qualifizierbarkeit am Zielsystem wechseln. Erst wenn dieser Bearbeitungsschritt hinreichende Stabilität und damit Verfahrenssicherheit gibt, sollte mit der weiteren Betrachtung der Contentverwendung fortgefahren werden, wo weitere Attribute auf Grund des [Basislayouts](#) hinzugefügt werden.



5 Zielsysteme Contentverwendung

Wenn der benötigte Content für das jeweilige Zielsystem definiert und qualifiziert ist und das zugehörige Klassenmodell erstellt ist, kann man sich der Frage nach der Contentverwendung widmen.

Je nach Verwendung des Contents muss dieser entsprechenden Prozeduren zur Content Behandlung unterzogen werden, so dass die Zielsysteme den Content in verarbeitbarer Form erhalten. Ebenso zählt die pro Zielsystem individuelle Aufbereitung, meist nach Unternehmens CI, zur Aufbereitung des Content.

5.1 Basislayout der Zielsysteme

Ebenso, wie pro Zielsystem verschiedene Zielformate benötigt werden, ist auch die individuelle Aufbereitung des Content höchst verschieden. So ist beispielsweise Content einer Klasse an einem Inter- und Intranet gleich, aber die optische Aufbereitung erfolgt unterschiedlich. Hier fließt also zum ersten mal die Frage nach dem optischen Design des benötigten Contents ein.

Meist wird das Basislayout für das Zielsystem von einer Agentur erstellt und geliefert.

Tip: Es wird bereits an dieser Stelle darauf hingewiesen, dass die Umsetzung des von der Agentur gelieferten Layouts in Form von Bildern und StyleGuide vollauf genügt da der Code nach CMS Richtlinien in Templates zerlegbar sein muss, die Agenturen aber, ihrer Aufgabe entsprechend, einen stärkeren Focus auf die Optik legen.

5.1.1 Basislayoutcode

Aufgrund des Basislayouts wird ein zielsystemspezifischer Basislayoutcode erstellt.

Der Basislayoutcode dient nicht nur der einheitlichen „[Content Behandlung](#)“ sondern ist ebenso Basis für die Erstellung der Indices über die [Indextemplates](#).

Das Basislayout gibt den Rahmen der Darstellung vor. Weitere Detaillierungen von Elementen innerhalb dieses Rahmens sind im StyleGuide zu definieren.

Sind Ausprägungen dieser Elemente von der Struktur abhängig, so können die benötigten Informationen ebenfalls in Klassenattributen festgehalten und sehr effizient vererbt werden.



Hier einige Beispielelemente:

- Hauptfarbe
- Grundfarbe
- Grundfarbe 2 (z.B. für alternierende Farben)
- Hintergrundfarbe
- Bild
- Hintergrundbild
- Rahmen
- Printversion (ja/nein)
- u.s.w.

Es würde wenig Sinn machen, den Autoren alle genannten Attribute pro Klasse zur Verfügung zu stellen. Vielmehr bietet es sich an, diese Attribute entweder als „hidden“ mit Vorgabewerten zu definieren, oder aber noch viel einfacher, gar nicht erst zuzuweisen, sondern lediglich die Attributwerte zu vererben, bzw. zu setzen mit sog. „defaults“.

5.1.2 Attributzuordnung – Mapping

DocMe unterscheidet zwischen zugeordneten (gemappten) Attributen und nicht zugeordneten Attributen.

Grundsätzlich verfügt jede Klasse über alle Systemattribute und die Anzahl definierter sog. Uservariablen, welche technisch gesehen die Attribute darstellen.

Lediglich die davon zugeordneten (gemappten) Attribute werden dem Autoren zur Verfügung gestellt. Die anderen Attribute sind dennoch verfügbar und können mit sog. „defaults“ belegt werden.

5.1.3 Defaults

Die „defaults“ gestatten es Werte von Attributen zu erfassen, ohne dass diese Klasse selbst über die definierten Attribute verfügt.

Ebenso wie die Werte von gemappten Attributen, können auch die Werte der „defaults“ vererbt werden und bieten so eine optimale Möglichkeit zur weiteren automatischen Informationsstrukturierung.



Die „defaults“ können in Kombination mit gemappten Attributen einerseits dem Autor die Möglichkeit geben ein Objekt durch Angabe bzw. Selektion eines Attributwerts weiter zu qualifizieren und andererseits einen „Defaultwert“ für den Fall setzen, dass der Autor keinen Gebrauch von einer weiteren Qualifizierung macht.

Es empfiehlt sich bei der Definition der zu nutzenden „defaults“ ein weitestgehend kongruentes Vorgehen mit den [Indextemplates](#) zu erzielen.

5.2 Content Behandlung

Zur Behandlung des Content stehen die folgenden grundsätzlichen Verfahren zur Verfügung.

5.2.1 Formatkonvertierung

Wenn der Autor in seiner gewohnten Umgebung Informationen erstellen soll, so muss sich danach ein automatisierter Konvertierungsprozess anschließen, der das vom Konsumenten benötigte Zielformat erzeugt.

Da die gleiche Information an verschiedenen Zielsystemen den Konsumenten zur Verfügung gestellt werden kann, muss sich die Formatierung individuell pro Zielsystem den Bedürfnissen der dortigen Konsumenten anpassen.

Pro Zielsystem ist daher für jede Klasse das bzw. die benötigten Formate zu definieren.

Hinweis: Die Konvertierung des Content findet zwar vom DocMe GatherD Server gesteuert am Content Pool statt, es ist aber zum Zeitpunkt der Modellierung der Klassenstruktur des Content Pool meist nicht bekannt, in welchen Formaten die einzelnen Zielsysteme den Content benötigen.

Zur Konvertierung von Office Formaten empfehlen wir den Einsatz des Konverters HTML-Transit von Stellent.

Dieser Konverter bietet bei Einsatz von Word Formatvorlagen über eine reine Konvertierung hinaus die Möglichkeit, Word Absatzformate zu erkennen und in frei definierbaren Code umzusetzen.

Diese Möglichkeit lässt sich für die folgenden zwei Aufgaben hervorragend nutzen.

Absatzformatierungen

Absätze können qualifiziert und mit Codeteilen zur späteren Formatierung über Cascadeing Stylesheets vorbereitet werden. Damit wird eine konsequente Umsetzung der CI unterstützt.



Die folgenden drei Schritte sind dabei zu definieren.

- Es empfiehlt sich die Schrift, Tabellenformatierungen etc. des StyleGuides bereits in den Word Formatvorlagen in allgemeingültigen Word Absatzformaten umzusetzen.
- Der Konverter HTML-Transit setzt die qualifizierbaren Word Absatzformate dann in CSS tauglichen Code um.
- Das verwendete CSS stellt dann den Content in der anzuzeigenden Form dar.

Beispiel:

Word Absatzformat:	headline1 (Arial, 14 pt, ...)
HTML-Transit:	<div class="headline1">.....</div>
CSS:	headline1{font-family: arial; font-size: 14pt;}

Content Assets

Contentteile können qualifiziert und definierten Klassenattributen zugewiesen werden. Damit steht nicht nur der komplett konvertierte Code, sondern auch Teile davon als sog. Content Assets zur Verfügung. Hierzu dienen insbesondere Word Formularvorlagen.

Die folgenden drei Schritte sind dabei zu definieren.

- Definition eines allgemeingültigen Word Absatzformates.
- Der Konverter HTML-Transit setzt die qualifizierbaren Word Absatzformate dann in Hook tauglichen Code um.
- Ein Hook überführt den qualifizierten Code in Klassenattribute

Beispiel:

Word Absatzformat:	container1
HTML-Transit:	<container1>.....</container1>
Hook:	Schreibt das Content Asset zwischen den <container> tags als Wert in die Variable \${User1}

5.2.2 Storage

Hier wird die Frage nach der Ablage und damit der Ansprache des Contents auf den Zielsystemen behandelt.

Content wird im Standardverfahren in der vorgegebenen Struktur, dem aus dem Klassennamen abgeleiteten PATH, abgelegt. Der komplette Klassenname dient dabei als erster Teil des Pfades des Ablageorts.



Daran schließt sich der zweite Teil des Pfades, ein maschinencodierter Verzeichnisname und letztlich der Name der Hauptdatei des Dokuments an.

Beispiel:

Teil 1: /Test
Teil 2: /ab83995cec2e64975da27.0.0/test.html
Summe: /Test/ab83995cec2e64975da27.0.0/test.html

Je nach [Objektanzahltyp](#) kann über ein Replacekriterium zusätzlich ein dedizierter, fixer Ablageort des Content über einen symbolischen Link generiert werden. Dieser bleibt auch beim Neueinstellen von Content gleich und verweist fix auf die gültige Adresse.

Beispiel:

Teil 1: /Test
Teil 2: /MeinReplaceArgument/test.html
Summe: /Test/MeinReplaceArgument/test.html

Hinweis: Das Replacekriterium kann auch erst am Zielsystem definiert werden. Dies sollte aber nur dann der Fall sein, wenn die betroffenen Objekte nur an einem der Zielsysteme, in der dadurch reduzierten Menge, vorhanden sein sollen.

5.2.3 StyleMe

StyleMe ist eine von arago AG entwickelte <tag>-basierte, aus nur vier Befehlen bestehende Sprache, zur Manipulation von Content.

Der roh konvertierte HTML Content wird über die <tag> Manipulation durch das Ausführen des StyleMe Skripte in eine dem Basislayoutcode oder Schnipsel davon entsprechenden Code überführt.

An Platzhaltern, sog. [DocMe Variablen](#), können Attributwerte der Objekte eingefügt werden. So werden auch die im Styleguide definierten Elemente voll automatisiert integriert.

StyleMe kann beliebigen <tag>-basierten Code manipulieren und daraus je nach Bedarf z.B. HTML, SHTML, XML, XSLT, PHP, ASP, JS, TXT, u.s.w. Code erzeugen. Ebenso können Mischformen erzeugt werden z.B. HTML & Java Script.

Allgemeine Definition

- StyleMe Skripte werden pro Zielsystem zentral erstellt und stehen dann allen Klassen bei Bedarf zur Verfügung.



- Pro Klasse können nach dem Prozessabschnitt PreContentHandler beliebig viele StyleMe Skripte definiert werden.
- Die Reihenfolge der Ausführung der StyleMe Skripte kann individuell pro Klasse definiert werden.

Arbeitsweise

- Erreicht ein Verarbeitungsprozess den Prozessabschnitt PreContentHandler so wird das/die StyleMe Skripte ausgeführt.
- Dabei stehen dem StyleMe Skript alle zu diesem Zeitpunkt verfügbaren Klassenattribute und deren Werte in Form von Umgebungsvariablen zur Verfügung.
- Während der Ausführung eines StyleMe Skripts können die folgenden grundsätzlichen Aufgaben bearbeitet werden:
 - Manipulation des Content
 - Integration von Content externer Systeme

Die vier Grundbefehle von StyleMe

- Select
- Insert
- Remove
- Replace

Durch Erweiterung der Befehle mit StyleMe Attributen können sehr mächtige Manipulationen am Content vorgenommen werden.

Weiter Informationen zu StyleMe finden Sie in der separaten Dokumentation „StyleMe“ von arago AG.

5.2.4 Hooks

DocMe Hooks stellen Prozessabschnitte im Verarbeitungsprozess dar, zu denen externe Anwendungen zur weiteren Verarbeitung integriert werden können.

Die DocMe Hook-Technologie ist damit eine außerordentlich flexible und sehr mächtige Methode zur Manipulation des Content entlang des gesamten Contentverarbeitungsprozesses..



An dieser Stelle wird deshalb bewusst nur auf die Möglichkeiten zur Manipulation des Content am Zielsystem eingegangen.

Weiterführende technische Informationen zu den einzelnen Prozessabschnitten finden Sie im DocMe Administrator Tutorial.

Allgemeine Definition

- Hook Skripte werden pro Zielsystem zentral erstellt und stehen dann allen Klassen bei Bedarf zur Verfügung.
- Pro Klasse können zu jedem Prozessabschnitt beliebig viele Hook Skripte definiert werden.
- Sets von Hookskripten können sehr effizient, ebenso wie andere Klassenattribute [vererbt](#) werden.
- Die Reihenfolge der Ausführung der Hook Skripte kann individuell pro Klasse definiert werden.
- Hook Skripte können in beliebigen Sprachen entwickelt werden, sofern sie in einer Shell gestartet werden können.
- Insgesamt gibt es am Content Pool 10 Klassen Hooks und am Zielsystem 6 Klassen Hooks, 4 Index Hooks sowie 2 Global Hooks.

Arbeitsweise

- Erreicht ein Verarbeitungsprozess einen Prozessabschnitt zu dem ein Hook Skript gestartet werden kann und der betreffenden Klasse ist ein Hook Skript zugeordnet, so wird es ausgeführt.
- Dabei stehen dem Hook Skript alle zu diesem Zeitpunkt verfügbaren Klassenattribute und deren Werte in Form von Umgebungsvariablen zur Verfügung.
- Während der Ausführung eines Hook Skripts können die folgenden grundsätzlichen Aufgaben bearbeitet werden:
 - Manipulation des Content/Attributwerten
 - Erzeugung von neuem Content/ Attributwerten
 - Integration von Content/Attributwerten externer Systeme
 - Übergabe von Content/Attributen und Werten an externe Systeme



- Manipulation, Erzeugung, Integration und Übergabe von Content/Attributwerten auf Grund der Information von externen Systemen

Hinweis: Zu beachten ist bei den schier unendlichen Möglichkeiten, die sich durch die DocMe Hook-Technologie bieten, dass z.B. bei der Erzeugung von neuem Content das verwaltende DocMe System ggf. keine Rückinformation bezüglich der Existenz dessen hat.

Eine sehr diversifizierte Nutzung kann ebenfalls die Übersichtlichkeit des Gesamtsystems, insbesondere bei Fehlersuchen, vermindern.

5.2.5 Klassentypen

Bisher wurde unterstellt, dass ein Objekt aus Attributen und Content in Form mindestens einer Datei besteht. Aber es kann auch sein, dass ein Objekt lediglich aus Attributen besteht. vgl. dazu [Objekte](#)

Klassen können deshalb für die folgenden Klassentypen deklariert werden.

- **Standard**
Die Objekte der Klasse enthalten mindestens je eine Datei (die Hauptdatei).

- **Nur Meta**
Die Objekte der Klasse enthalten keine Datei.

Beispiel:

FAQ Objekte bestehen in der Regel aus einer Frage, einer Antwort und einer Kategorisierung. Wenn dabei die Fragen und Antworten aus reinen Textanteilen und ohne weitere Formatierung bestehen können, so kann eine Klasse als Typ „Nur Meta“ mit dementsprechenden Attributen definiert werden. Ergebnis der Definition ist eine Klasse, in welche der Autor lediglich die Objektattribute (Metainformationen) erzeugen kann. Diese genügen aber auch schon zur Erstellung z.B. einer FAQ Site.

Der Vorteil liegt u.a. in der schnellen Verarbeitung, da keine Konvertierung stattfinden muss.

Auf diese Weise können also DocMe Formulare zur Erfassung von Informationen erstellt werden.

- **Beides**
Die Objekte der Klasse können wahlweise Dateien enthalten



oder nicht.

Beispiel:

Eine Klasse zu einem Thema enthält Objekte. Es sollte aber auch ein Objekt als „Link zu einer externen Site“ als Objekt bereitgestellt werden können. Dazu benötigt es keine Datei, sondern lediglich die Attribute und ein Linkattribut.

5.2.6 Aufnahmegrenzen

Zur Informationstypisierung kann der Dateityp und/oder die Dateigröße und Anzahl der Dateien aus denen ein Objekt besteht relevant sein. Die Aufnahmegrenzen wirken bei den Klassentypen ‚Standard‘ und ‚Beides‘.

Die Aufnahmegrenzen stellen aber im Wesentlichen einen ersten wichtigen Schritt zur Qualitätssicherung dar. Die folgenden ‚Limitations‘ stehen zur Verfügung

- **Maximale Anzahl Dateien**
Maximale Anzahl der Dateien aus der ein Dokument bestehen darf.
- **Maximale Größe einer Datei**
Maximale Größe jeder einzelnen Datei aus der ein Dokument bestehen darf.
- **Maximale Größe aller Dateien**
Maximale Größe der Summe aller Dateien aus der ein Dokument bestehen darf.
- **Dateinamenserweiterung**
Die Hauptdatei darf nur vom Typ der Dateinamenserweiterungsliste sein.

5.3 Content Einbinden mit DocMe Standards - Indices

Wurde der Content pro Zielsystem entsprechend formatiert und optisch aufbereitet, sowie ggf. über DocMe Hooks modifiziert, gilt es den Content zu präsentieren.

DocMe verfügt hierzu über ein sehr mächtiges Werkzeug, die sog. Indices.

Ein Index kann sowohl Dateien mit Content von einem Objekt und/oder dessen Attribute, als auch Dateien von mehreren Objekten und/oder Attributen bzw. Referenzen zu diesen Objekten erzeugen.



DocMe Information Modelling

Zielsysteme Contentverwendung

Indices dienen sowohl zur Darstellung von Content als auch zur Erstellung von contentspezifischen Navigationen.

Darüber hinaus können Indices dazu benutzt werden, verbleibenden, bisher nicht von DocMe erzeugten statischen Content, auch in DocMe zu integrieren und verwaltbar zu machen.

Allgemeine Definition

- Jeder Index enthält ein Regelwerk, welches die zu präsentierende Menge der Objekte definiert.
- Ein Objekt kann beliebig vielen Indices über die Regelwerke zugewiesen werden.
- Die entstehende Objektsammlung kann nach beliebigen Kriterien sortiert werden.
- Jedem Index können Templates zugewiesen werden, welche den optischen Aufbau und die Inhalte der zu präsentierenden Informationen der Objekte repräsentieren.
- Indextemplates werden pro Zielsystem zentral erstellt und stehen dann allen Indices bei Bedarf zur Verfügung.

Arbeitsweise

- Indexregeln erzeugen zur Wahrung der referenziellen Integrität bei Veränderungen der DocMe Dokumente (Objekte) nach dem Prozessabschnitt PrePublishDocument die resultierenden Indexdateien automatisch neu. Eine Pflege bestehender Indices entfällt damit.

Die Objekte selbst müssen dazu keiner weiteren Behandlung unterzogen werden, sie verbleiben an ihrem Ablageort, sie werden nicht kopiert und alleine das „Auftauchen“ eines zum Regelwerk gehörenden Objektes genügt zur Neugenerierung und damit zur Aktualisierung des Index.

- Zum Zeitpunkt der Neugenerierung stehen dem Index in allen Templates alle Indexattribute zur Verfügung. Die Werte der Klassenattribute und der Content der Objekte stehen im Entrytemplate zur Verfügung.
- Ergebnis der Neugenerierung ist die Erstellung der Indexdatei bzw. Indexdateien.

Vererbung

Indices sind im OO Sinne sog. Sammlungen und verfügen über die



Eigenschaft der Vererbung.

Dieser Umstand erlaubt es, ebenso wie bei den Klassen, sehr effizient neue Indices zu erzeugen. Aber, wie bei den Klassen, ist auch bei den Indices eine definierte Struktur der Vererbung, meist über mehrere Generationen, vorher zu definieren.

Vererbbar sind alle Attribute eines Index außer dem Namen der zu generierenden Indexdatei und dem Regelwerk zur Definition der zugehörigen Objekte.

Ebenso wie es „Abstrakte Klassen“ gibt, gibt es „Abstrakte Indices“ welche ihrerseits keine Indexdateien erzeugen, sondern lediglich ihre Attributwerte vererben. Dadurch erreicht man eine gute Abgrenzung zu den konkreten Indices und erhöht die Wartbarkeit. Es gilt dabei die folgenden Vererbungsregel:

- **Vererbungsregel:**
Ist ein Attributwert eines Parentindex im Parentindex gesetzt, so wird er von Childindex übernommen. Ist im Childindex selbst ein Attributwert gesetzt, so hat dieser Vorrang. Diese Regel gilt über beliebig viele Vererbungsgenerationen hinweg.

Indexattribute (Indexmetavariablen)

Analog den „[defaults](#)“, den nicht gemappten Attributen der Klassen, gibt es Indexattribute, die sog. Index Metavariablen.

Jeder Index verfügt über alle Indexattribute. Über die Vererbung der Attributwerte können neu Indices einerseits sehr schnell erzeugt werden, verfügen dabei aber über mehrere Generationen vererbter Attributwerte oder über direkt zugewiesene Attributwerte

Tip: Die Indexattribute sollten weitestgehend kongruent mit der Definition des Basislayouts respektive den StyleMe Skripten gehalten werden.

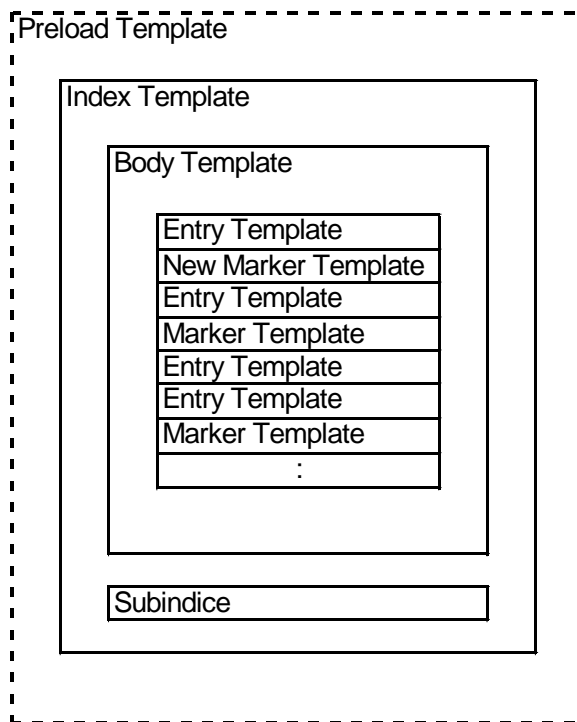
Templates

Sind Dateien mit Schnipsel vom [Basislayoutcode](#) welche an bestimmten Stellen Platzhalter, sog. DocMe Variablen, zum Einfügen von Attributwerten und/oder Content enthalten.

Platzhalter sind einerseits die Indexattribute, und in den Entrytemplates aber auch die Klassenattribute.

Vgl. auch „[StyleMe](#)“.

Templates werden geschachtelt nach dem folgenden Schema aufgebaut:



- **Pre Templates**
Nehmen global definierte Variablen und global verfügbare verfügbare Java Script Funktionen auf.
- **Index Templates**
Nehmen den Rahmen [Basislayoutcodes](#), versehen mit Platzhaltern für Indexattributwerte, auf.
- **Body Templates**
Nehmen den Hauptkörper des Basislayoutcodes, versehen mit Platzhaltern für Indexattributwerte, auf.
- **Entry Templates**
Nehmen den objektspezifischen Teil des Basislayoutcodes, versehen mit Platzhaltern für Indexattributwerte und den Platzhaltern für die Objektattributwerte, auf.
- **Marker Templates**
Nehmen den Teil des Basislayoutcodes für definierte Listverfahren auf. Z.B. Einspaltig, Zweispaltig, von links nach rechts oder von oben nach unten auffüllend, etc.
- **New Marker Templates**
Nehmen den Teil des Basislayoutcodes für die Anzeige eines Aktualitätshinweises auf.



Ergebnis einer Indexerzeugung ist anschließend ein kompletter, dem Basislayoutcode entsprechender Code.

Die Templates repräsentieren in der Summe syntaktisch vollständigen Zielsystemcode, können aber auch sog. „raw“ Code zur dynamischen Integration über [Server Side Include](#) erzeugen.

Templates können je nach Bedarf beliebigen Code erzeugen, z.B. HTML, SHTML, XML, XSLT, PHP, Perl, ASP, JS, TXT, u.s.w. Ebenso können Mischformen erzeugt werden z.B. HTML & Java Script.

Weitere Informationen zu den Indextemplates finden Sie im „DocMe Administrator Tutorial“.

5.3.1 Einfache Indices

Einfache Indices sind Objektsammlungen einer Klasse. Im einfachsten Fall wird eine Liste mit Referenzen zu den Objekten erstellt, oder direkt der Objektcontent und / oder Attributwerte erzeugt.

5.3.2 Komplexe Indices

Divided Indices

Komplexe Indices erlauben die automatische Rubrizierung der Objektsammlung nach einem sog. „Dividekriterium“. Dies kann beliebig aus Objektattributen und Textinformationen zusammengesetzt werden.

Durch die Rubrizierung der Objektsammlung können komplexe Indices sehr große Mengen an Objekten darstellen. Sie stellen daher ein ideales Werkzeug zur vollständig automatisierten Erzeugung von Contentsammlungen unbestimmter Größen dar und sind wann immer möglich zu bevorzugen.

Beispielsweise können so sehr effizient sich über die Datumsattribute oder die Titelattribute selbst aktualisierende Archive aufgebaut werden. Ohne jeglichen weiteren Pflegeaufwand!

Sub Indices

Stellen eine weitere Form von komplexen Indices dar. Grundsätzlich ist jeder Index auch als Subindex aufrufbar.

Aus jedem Template kann ein bereits existierender Index als Subindex integriert werden.

Sie dienen zur Zusammenfassung von Informationen die aufgrund ihrer



Klassenstruktur nur sehr schwer in einem Index zu packen oder sinnvoll darzustellen wären.

So braucht ein Index ggf. selbst keine eigenen Regeln und Objekte, sondern fasst nur bereits bestehende Indices als Subindices zusammen.

So lassen sich durch das Zusammenfassen mehrerer Indices sog. Matrixindices erstellen.

5.3.3 Umfassende Indices

Die Indexregeln sind nicht auf eine Klasse, oder auf eine Oberklasse beschränkt, sondern sie können klassenübergreifend definiert werden.

Typischerweise wird diese Eigenschaft bei „Aktuelle Information“ und/oder Newslettern benutzt. Dort steht die Aktualität eines Objektes im Vordergrund und nicht die Klassifizierung.

5.3.4 Direktanzeige

An Stelle der Referenz zu den Objekten kann der Content auch direkt dargestellt werden..

Wenn der Content bereits in der gewünschten optisch aufbereiteten Form vorliegt, kann er direkt aus einem Entrytemplate übernommen werden.

Ist die optische Aufbereitung über bestehende Templates gewünscht, so kann die betreffende Klasse über einen Hook aus dem Content einen sog. „Rawcontent“ erstellen. Dieser „Rawcontent“ kann, unter Beibehaltung von syntaktisch korrektem Code, integriert werden.

Eine Alternative zur Direktanzeige über Indices stellt die Verwendung von [Replacekriterien](#) in Zusammenhang mit dedizierten symbolischen Links dar.

5.4 Content Einbinden Erweitert

Über die Standard Einbindung von Content hinaus

5.4.1 Hook Vererbung

[Hooks](#) stellen eine außerordentlich flexible Möglichkeit zur Manipulation von Objekten wie zur Erzeugung von neuem Content dar.

Hooks stellen im Sinne des objektorientierten Modell ebenfalls Objekte dar. Deshalb können die Hook-Attribute, ebenso wie die Klassen- oder Indexattribute, vererbt werden. Eine Zuweisung kann sehr effizient



gemeinsam mit den sonstigen Klassen- bzw. Indexattributen vererbt werden.

Dabei wird je nach Hook Typ unterschieden zwischen:

- **Klassen Hooks**
Die zu einem Prozessabschnitt auszuführenden Hooks werden einer bestimmten Klasse zugeordnet.

Beispiel:

Ein Hook transformiert in einer Klasse „Bilder“ jeweils die Hauptdatei zum Prozessabschnitt „PreProcess“ in eine genormte Bildgröße. Anschließend wird eine zusätzliche neue Datei „teaser.jpg“ zur Voranzeige in Indices erzeugt.

- **Index Hooks**
Die zu einem Prozessabschnitt auszuführenden Hooks werden einem bestimmten Index zugeordnet.

Beispiel:

Wird einem Index ein neues Objekt hinzugefügt, erzeugt der hook eine Infomail.

Natürlich könnte man einen solchen Hook auch direkt der Klasse zuordnen. Wenn der Index aber mehrere Klassen mit nur bestimmten Attributwerten umfasst, so kann die Zuordnung über einen Index Hook sehr viel effizienter erfolgen.

- **Global Hooks**
Nur der Vollständigkeit halber werden hier die globalen Hooks aufgeführt. Sie haben kaum Relevanz für die Informationsmodellierung und dienen z.B. der Information über Systemstarts.

Eine Liste aller Hooks finden Sie im [Anhang](#).

5.4.2 Download verschiedener Formate

Oft ist die Bereitstellung des gleichen Content in verschiedenen Formaten für die Konsumenten notwendig. Beispielsweise ist in Intranets einerseits das schnelle Browsen und das einfache Suchen gewünscht, andererseits aber auch die Nutzung der Originaldatei z.B. ein Musterbrief zur weiteren Vervollständigung.

In einem solchen Fall sind mindestens die Formate „.html“ und „.doc“.

Die DocMe Server können dazu wahlweise konfiguriert werden, ob nur das



konvertierte oder aber auch das Originalformat dem Zielsystem übergeben werden sollen.

Ist aber ein weiteres Zielformat z.B. „.PDF“ notwendig, so kann dieses am Zielsystem über einen Hook zusätzlich erzeugt werden.

Die Ablage erfolgt typischerweise im gleichen Objektverzeichnis und der Zugriff wird über eine Ableitung des originalen Dateinamens realisiert.

Auf diese Weise können mehrere Formate zum Download bereitgestellt werden.

Hinweis: Für die Zukunft ist eine Erweiterung der Konvertiermöglichkeiten bereits am Content Pool geplant, so dass dann selbst ohne Hooks beliebig viele Formate des gleichen Contents bereitgestellt werden können.

5.5 Content in Applikationen

Einen wichtigen Teil der Applikationen stellen die Navigationen dar. Sie sollen die Erreichbarkeit des Content ermöglichen. Dabei sollte der Content immer auf mehreren Wegen dem Konsumenten bereitgestellt werden. So gibt es die folgenden grundsätzlich verschiedenen Wege

- Indices (Teilnavigation)
- Navigationen
- Sitemaps
- Advertimentboxen
- Suchmaschinen
- Personalisierung
- Newsletter
- DB Zugriff

Über die Navigation auf einer WebSite hinaus, wird der gleiche Content weiteren Applikationen zur Erschließung des Contents zur Verfügung gestellt.

- Suchmaschinen
- Personalisierung



- Newsletter

5.5.1 Indices

Schon wieder Indices? Ja. Diesmal als Applikation für einfache Navigationen oder Teilnavigationen. Sie können über Indices erstellt und sowohl in Singleframe als auch in Multiframe integriert werden.

Dabei können sowohl syntaktisch vollständige Dateien oder aber sog. „Rawcontent“ zum einbinden via SSI erzeugt werden.

Besonders interessant ist die Möglichkeit, über komplexe Indices Teilnavigationen vollständig automatisch aufbauen zu lassen. Insbesondere in Intranetanwendungen werden solche Autonavigationen integriert.

5.5.2 Navigation

Navigationen beinhalten einerseits eine statische Komponente – man wird dem Konsumenten kaum jeden Tag eine neue Navigationsstruktur zumuten – aber auch eine dynamische Komponente zur Ergänzung, Granulierung oder Hervorhebung von Informationen.

Das DocMe Navigationsmodul ist daher unabhängig von der Content Pool Struktur. Es stellt Verknüpfungen zu Objekten, Objektsammlungen, anderem statischen Content oder Anwendungen her. Die aus dieser Struktur erzeugte Navigation verknüpft dann die Navigationsknoten mit dem abgegebenen Content.

Hinweis: Eine feste Verknüpfung zwischen der Content Pool Struktur und der Navigationsstruktur ist im Grundsatz abzulehnen. Sonst würde ein Relaunch eine Umstrukturierung der Content Pool Struktur nach sich ziehen!

5.5.3 Sitemap

Eine Sitemap sollte sich automatisch aus der definierten Navigationsstruktur ableiten. Genau dies wird vom DocMe Sitemapmodul erstellt.

5.5.4 Adverstimentboxen

Sie stellen eine einfache aber optisch stark hervorgehobene Methode dar, auf Content aufmerksam zu machen. Das DocMe Advertisementmodul arbeitet, wie das Navigationsmodul, auf einer frei zu definierenden Struktur.



5.5.5 Suchmaschinen

Sie können ohne weiteres Zutun vom CMS System die resultierenden WebTrees regelbasiert scannen und indizieren.

Interessant ist aber auch die Möglichkeit, Content auf Grund von geerbten Attributen zu selektieren. Hierzu kann der Suchmaschine ein direkter Zugriff auf die DocMe Datenbank des Zielsystems gewährt werden.

Zur Informationsmodellierung sind ggf. eigene Rubriken an Attributen zur gezielten Information der Suchmaschinen über diese Methode zu berücksichtigen.

5.5.6 Personalisierung

Personalisierungssysteme sollen Content profilorientiert zur eigenen Assemblierung durch den Konsumenten für sog. „MySites“ bereitstellen. Dabei gibt es zwei grundsätzlich verschiedene Methoden zur Qualifizierung des Content welcher dem Konsumenten zur individuellen Zusammenstellung zur Verfügung steht.

- Contentverfügbarkeit auf Grund der Gruppenzugehörigkeit
- Contentverfügbarkeit auf Grund des Verhaltens

Gruppenzugehörigkeit

Die Mitgliedschaft eines Konsumenten in einer Gruppe definiert den für diesen selektierbaren „Personal Content Offer“ am Zielsystem. Eine MySite Anwendung regelt dann die Möglichkeit zur individuellen Zusammenstellung des Angebots.

Dazu muss für die jeweiligen Personalisierungsgruppen [Content qualifizierbar](#) gemacht werden und hat damit Rückwirkung auf das Informationsmodell.

Das DocMe Personalisierungsmodul stellt sehr umfangreiche Möglichkeiten zu regelbasierte Definition und Bereitstellung von Content zur Verfügung.

Verhalten

Das Verhalten eines Konsumenten mündet in sog. Profilen welche wiederum die Zuordnung weitere Gruppen und damit die Erweiterung des „Personal Content Offer“ nach sich ziehen können.

Weiter können über die Analyse der allgemeinen Akzeptanz weitere oder veränderte Informationsangebote abgeleitet und definiert werden.

Das DocMe Personalisierungsmodul stellt mit dem Statistikmodul Möglichkeiten zu Verhaltensanalyse zur Verfügung und bietet die



Grundlage zur Anpassung des Informationsangebots an das Konsumentenverhalten.

5.5.7 Newsletter

Newsletter bieten eine gute Möglichkeit die Konsumenten zu binden und darüber hinaus die ersten Kontaktinformationen sowie Verhaltensinformationen zu erhalten.

Für Newsletter bestimmte Objekte sind als solche zu qualifizieren und in bestimmten Rhythmen zu sammeln, in das gewünschte bzw. abonnierte Format zu bringen und zu verteilen.

Das DocMe Newsletter Modul arbeitet nach dem „double-opt-in“ Verfahren, um sicherzustellen, dass ein abonnierter Content auch wirklich vom Adressaten abonniert wurde.

5.5.8 DB Zugriff

Grundsätzlich können Anwendungen auch direkten Zugriff auf die DocMe Datenbank am Zielsystem erhalten.

Nach Vertragsabschluss legen wir unseren Kunden dazu das DocMe Datenmodell am Zielsystem offen.

5.5.9 Server Side Include (SSI)

Über die Server Side Include Technik können Informationen in allen Templates oder StyleMe Skripten definiert und beim Abruf über den WebBrowser dynamisch integriert werden.

Diese Technik kann sowohl dazu genutzt werden dynamische als auch statische Informationen zu integrieren.

Dynamisches Beispiel:
Aufruf eines PHP Skriptes mit Datenbankabfrage

Statisches Beispiel:
Aufruf bestehender [Indexdateien](#) im „raw“ Format.

5.5.10 Caching



6 Content Veredelung

6.1 StyleMe

StyleMe Skripte können nicht nur während des Standard Publikationsprozess, sondern auch über die Hookschnittstelle benutzt werden.

StyleMe wird dazu von der Commandline aus, z.B. in einem Shellscript aufgerufen.

Wenn der „Rawcontent“ eines Objektes benötigt wird, so kann dieser über einen Hook, der wiederum ein StyleMe Skript startet, erzeugt werden. Das StyleMe Skript markiert dazu lediglich die gewünschten Codepassagen und verwirft den Rest.

Dieser neu erzeugte Content kann dann z.B. zusätzlich zum kompletten Code, als „.raw“ abgelegt werden.

Damit der „Rawcontent“ bei jeder Klasse gut markiert werden kann, empfiehlt es sich einen für diesen Zweck eigens definierten tag, z.b. `<rawmarker>` `</rawmarker>` vor und nach den eigentlichen Code im StyleMe Skript zu positionieren.

Vgl dazu [Direktanzeige](#)

6.2 Linkage

Nicht nur die sich vollständig verlinkenden Indices, das DocMe Navigations Modul oder das DocMe Advertising Modul stellen eine Verlinkung und damit Navigationsmöglichkeit dar. Auch die Möglichkeit Content selbst, regelbasiert und voll automatisiert zu verlinken oder aber die Möglichkeit gezielt vom Autor selbst bereits bei der Erfassung Verlinkungen zu initiieren, sind integrierbar.

6.2.1 Keywords

DocMe verfügt bereits im Standard über ein sehr mächtiges Keyword Modul. Dieses erlaubt die voll automatisierte, regelbasierte Verlinkung des Content untereinander. Dabei werden fest vorgegebene Begriffe ebenso wie vom Autor beim Publizieren frei erfasste Keywords unterstützt.

Eine Liste mit zu verlinkenden Standardkeywords mit deren Zugehörigkeitsbereich ist zu erstellen und zu gruppieren. Klassen in denen Autoren freie oder aus Listen stammende Keywords zur automatisierten Keywordverlinkung



Es ist darauf zu achten, dass die verwendeten Begriffe so gewählt sind, dass sie innerhalb des Zugehörigkeitsbereichs nicht zigfach pro Objekt vorkommen. Eine automatische Verlinkung des Contents würde in diesem Fall die Lesbarkeit des Contents auf Grund der häufigen Links vermindern.

Hinweis: Aus diesem Grund ist es ratsam, die automatische Keywordverlinkung ggf. erst in einem zweiten Schritt, nachdem die Häufigkeit der Keywords pro Bereich analysiert worden ist, zu integrieren.

6.2.2 Attribute

Ebenso können Klassenattribute als Informationsträger für Links genutzt werden.

Diese können klassenspezifisch als „defaults“ zugeordnet oder aber dem Autoren zur Auswahl gestellt werden.

Es können pro Klasse sowohl individuelle als auch vererbte Linklisten definiert werden. Diese stehen dann dem Autor zur Auswahl zur Verfügung.

Die Attributwerte können anschließend über StyleMe am Dokument oder über Entrytemplates in Indices gesetzt werden.

So können auf einfache Art und Weise Erklärungs und Hilfesysteme erstellt werden.

6.2.3 Attachmanagement

Objekte können ihrerseits aus einer Hauptdatei mit mehreren untereinander verlinkten, anhängenden Dateien bestehen. Das DocMe Attachmanagement erlaubt die einfache Verlinkung von einer Word Datei zu weiteren anhängenden Dateien.

DocMe setzt diese Links beim Publizieren des gesamten Objektes um und verwaltet sie automatisch.

6.2.4 Linkmanagement

Das DocMe Linkmanagement Modul ermöglicht das Setzen eines Links zu bereits in DocMe verwalteten Objekten und der Dateien bereits bei der Erstellung des Content aus Word heraus.

DocMe setzt diese Links beim Publizieren des gesamten Objektes um und verwaltet sie automatisch.



6.3 Variablen

DocMe Variablen repräsentieren, je nach Nutzungsort, Klassen-, Index- oder Systemattributwerte.

Die allgemeine Form einer DocMe Variablen lautet

```
#{NameDerVariablen}
```

ggf. verfügen die Variablen über weitere Attribute

```
#{NameDerVariablen:Attribut}
```

Wird ein solcher Code in einem StyleMe Skript, Indextemplate, Hook- oder Converter Skript angetroffen, so wird die Variable durch den entsprechenden Attributwert ersetzt.

Eine Liste aller Variablen ist im [Anhang](#) beigefügt.

6.3.1 Einfache Variablen

Die DocMe Variablen untergliedern sich in

Systemvariablen

Sind vom DocMe System grundsätzlich vorhandene Klassen-, Index- oder Systemattribute oder Funktionen.

Uservariablen (Klassenattribute)

Frei durch den Administrator zu definierende Anzahl an Klassenattributen, die jeder Klasse, zumindest nicht gemappt, zur Verfügung stehen.

Metavariablen (Indexattribute)

Frei durch den Administrator zu definierende Anzahl an Indexattributen, die jeder Klasse, zumindest nicht gemappt, zur Verfügung stehen.

Eine Liste aller Variablen ist im Anhang beigefügt.

6.3.2 SQL Variablen

Besonders leistungsfähig ist die Nutzung der SQL Variablen. Sie bieten die Möglichkeit SQL Abfragen an die DocMe Datenbank zu senden. Hierüber stehen alle in DocMe verfügbaren Informationen zur Verfügung.

```
#{SQL:SELECT Profil FROM Autoren WHERE Autor = '#{AUTHOR}'}
```



6.3.3 LOADFILE

Oft benötigte statische Codeschnipsel können an zentraler Stelle abgelegt und von mehreren Templates und StyleMe Scripten während der Erzeugung geladen werden.

```
#{LOADFILE:Name_und_Pfad_der_Datei}
```

Beispielsweise kann man statische Bereiche der <Meta> tag Informationen in HTML Dateien einmal erstellen und sie von allen Templates und Skripten aus nutzen.

So kann bei einer ggf. späterer folgenden Erweiterung ein Republish und Neuindizierung in allen resultierenden Objekten die neuen Codeschnipsel aktualisieren.

6.3.4 JAVASCRIPT

Über die Nutzung der SQL Abfragen hinaus, kann innerhalb der Templates und StyleMe Skripte, der vollständige Sprachumfang von Java Script als Logikelement benutzt werden.

Hier ein Beispiel für ein PreLoad Template, das die Werte von zwei Indexmetavariablen in Java Script Variablen überführt und die Funktion ‚color‘ definiert

```
#{JAVASCRIPT:  
var color1 = "##{META5}";  
var color2 = "##{META6}";  
function color(n)  
  {  
    if((n%2) != 0) return color1;  
    return color2  
  }  
}
```

Für dieses Beispiel könnte im zugehörigen Entry Template z.B. folgender Code benutzt werden um einen farblich alternierenden Hintergrund zu erzeugen.

```
<tr bgcolor="##{JAVASCRIPT:color(#{PARTDOCCOUNT})}">  
  <td>  
    <a href="##{DOCDIR}/#{MAINFILE}">#{'TITLE'}</a>  
  </td>  
</tr>
```



DocMe Information Modelling

Content Veredelung

Hinweis: Nicht zu verwechseln mit der Erzeugung von Java Script Code der am WebBrowser zur Ausführung kommt! Dieser kann immer und jederzeit aus jedem Template oder StyleMe Skript erzeugt werden. Die Variable `#{JAVASCRIPT:... }` übergibt ihre Attributwerte an einen Java Script Interpreter, der das Ergebnis der Anfrage zurückschreibt. Nach der Generierung des StyleMe oder Templates ist also nicht mehr der in der Variablen enthaltene Java Script Code enthalten, sondern dessen interpretiertes Ergebnis.



7 Regelwerke

7.1 Content Pool Regeln

Die Content Pool Regeln, die sog. Verteilregeln, definieren die von einem Content Pool an ein Zielsystem zu verteilende Informationsmenge.

Die Content Pool Regeln arbeiten nach dem gleichen unter „Index Regeln“ beschriebenen Verfahren.

Tip: Wenn es die Vertraulichkeit von Contentmengen zulässt, kann es sinnvoll sein, anstatt einer sehr tiefen Diversifizierung der Verteilregeln, lieber „etwas mehr“ Content an das Zielsystem zu verteilen und damit die Verteilregeln schlank und übersichtlich zu halten.

7.2 Content Pool – Content Pool Regeln

Die Content Pool – Content Pool Regeln definieren die zwischen zwei Content Pools auszutauschende Informationsmenge. Gehören die Content Pools dabei zum gleichen Konzern, so kann eine sehr ähnliche, oder gar deckungsgleiche Content Pool Struktur ermöglicht werden. Je höher die Übereinstimmung der Content Pool Strukturen ist, desto geringer ist der zu erwartende Supportaufwand im Falle einer Fehlersuche.

Bei völlig verschiedenen Content Pool Strukturen wie z.B. zur Content Syndication oder konzernweiter Mehrfachnutzung von Content bei sehr großer Selbständigkeit der einzelnen Konzerngesellschaften, ist von einer sehr divergenten Content Pool Struktur auszugehen.

Hierzu gibt es das sog. „Cross Distribute“ Verfahren. Es ermöglicht bei der Erstellung einer Verteilregel, nicht nur die Objektmenge zu qualifizieren, sondern darüber hinaus die „Umlenkung“ des Contents in eine andere Klassenstruktur.

Zusätzlich können bei dieser Umlenkung die Attribute angepasst werden, da ja nicht davon auszugehen ist, dass eine fremde Content Pool Struktur die gleichen Attribute nutzt.

Hinweis: Cross Distribution kann theoretisch auch von einem Content Pool zu einem zugehörigen Zielsystem erfolgen. Eine solche Nutzung sollte aber die absolute Ausnahme bleiben. Ansonsten leidet die Übersichtlichkeit für die Autoren und damit die Nachvollziehbarkeit der Prozesse darunter. Sind sie dennoch unumgänglich, so ist es ratsam, diese Ausnahmen



zu dokumentieren und den Autoren bekannt zu machen.

7.3 Index Regeln

Die zu einem Index gehörenden Objekte werden über Regeln definiert.

- Ein Regelsatz schränkt die Menge der Objekte über die Klasse oder Oberklasse ein.
 - Beispiel:
Es sollen alle Objekte aus der Klasse „Wertpapiere“ zu einem Index gehören
Klasse: Wertpapiere
- Innerhalb eines Regelsatzes können Einschränkungen die Menge der Objekte reduzieren.
 - Beispiel:
Es sollen alle Objekte aus der Klasse „Wertpapiere“ zu einem Index gehören, allerdings nur, wenn das Attribut „Wertpapiertyp“ den Wert „Aktien“ enthält. (Das Attribut „Wertpapiere“ soll in der Variablen User1 abgelegt worden sein.)
Klasse: Wertpapiere
User1: Aktien
- Pro Regelsatz können mehrere Einschränkungen definiert werden.
 - Beispiel:
Es sollen alle Objekte aus der Klasse „Wertpapiere“ zu einem Index gehören, allerdings nur, wenn das Attribut „Wertpapiertyp“ den Wert „Aktien“ enthält und der Autor „Peter Steffan“ ist.
Klasse: Wertpapiere
Author: Peter Steffan
User1: Aktien
- Einschränkungen können auch negiert erfolgen.
 - Beispiel:
Es sollen alle Objekte aus der Klasse „Wertpapiere“ zu einem Index gehören, allerdings nur, wenn das Attribut „Wertpapiertyp“ nicht den Wert „Renten“ enthält und der Autor „Peter Steffan“ ist.
Klasse: Wertpapiere
Author: Peter Steffan
User1: !Renten
- Wildcards ermöglichen es bestimmte Textanteil zu qualifizieren.



Regelwerke

- **Beispiel:**
Es sollen alle Objekte aus der Klasse „Wertpapiere“ zu einem Index gehören, allerdings nur, wenn das Attribut „Wertpapiertyp“ nicht den Wert „Renten“ enthält und der Autor den String „Peter“ enthält.
Klasse: Wertpapiere
Author: *Peter*
User1: !Renten

- Pro Index können beliebig viele Regelsätze definiert werden. Jeder Regelsatz definiert eine weitere zur Sammlung gehörende Objektmenge.

- **Beispiel:**
Es sollen alle Objekte aus der Klasse „Wertpapiere“ zu einem Index gehören, allerdings nur, wenn das Attribut „Wertpapiertyp“ nicht den Wert „Renten“ enthält und der Autor den String „Peter“ enthält. Weiter sollen alle Objekte aus der Oberklasse „DAX“ zum Index gehören.
1.Regelsatz:
Klasse: Wertpapiere
Author: *Peter*
User1: !Renten
2.Regelsatz:
Klasse: DAX.*



8 Anhang

8.1 Variablen

8.1.1 Klassenvariablen

Variable	Beschreibung
<code>\${AUTHOR}</code>	Name des Dokumentenautor
<code>\${AUTHOREMAIL}</code>	Email Adresse des Autor
<code>\${BASE}</code>	Dateinamen ohne Extension von <code>\${MAINFILE}</code>
<code>\${CLASS}</code>	Klassenname, identisch mit <code>\${CLASSNAME}</code>
<code>\${CLASSNAMEPART:x}</code>	Die Argumente x sind positive und negative Ganzzahlen. Bsp. Die Klasse sei "Beispiele.Karriere.Schueler" <code>\${CLASSNAMEPART:1}</code> liefert "Beispiele" <code>\${CLASSNAMEPART:2}</code> liefert "Karriere" <code>\${CLASSNAMEPART:3}</code> liefert "Schueler" <code>\${CLASSNAMEPART:4}</code> liefert "" <code>\${CLASSNAMEPART:-1}</code> liefert "Schueler" <code>\${CLASSNAMEPART:-2}</code> liefert "Karriere" <code>\${CLASSNAMEPART:-3}</code> liefert "Beispiele" <code>\${CLASSNAMEPART:-4}</code> liefert "" <code>\${CLASSNAMEPART:0}</code> liefert immer ""
<code>\${DATE}</code>	Erstellungsdatum, identisch mit <code>\${DOCDATE}</code> Bsp: <code>\${DATE:%1%A, %d. %B, %Y}</code>
<code>\${DOCDATE}</code>	Erstellungsdatum, identisch mit <code>\${DATE}</code>
<code>\${DOCDIR}</code>	relativer Pfad zum Dokument
<code>\${DOCDIRFULL}</code>	Pfad zum Dokument ab dem DocMe DocRoot
<code>\${DOCID}</code>	eindeutige Dokumenten Identifikationsnummer
<code>\${DOCROOT}</code>	Kompletter Pfad zum Dokument
<code>\${DOCTIME}</code>	Erstellungszeit, identisch mit <code>\${TIME}</code>
<code>\${EXT}</code>	Fileextension von <code>\${MAINFILE}</code>
<code>\${EXPIRE}</code>	Verfallsdatum in Sekunden von 1970



DocMe Information Modelling

Anhang

<code>#{EXPIREDATE}</code>	Verfallsdatum
<code>#{EXPIRETIME}</code>	Verfallszeit
<code>#{FILEx}</code>	<code>#{FILE0}</code> gibt den Namen der 1. Datei (Haupt oder anhängende zu konvertierende Datei) zurück. <code>#{FILE1}</code> gibt den Namen der 2. Datei (Haupt oder anhängende zu konvertierende Datei) zurück
<code>#{FILECOUNT}</code>	Anzahl der Dateien in <code>#{FILELIST}</code>
<code>#{FILELIST}</code>	Durch Kommata getrennte Liste der Dateinamen
<code>#{GATHER}</code>	Einstellungsdatum in Sekunden von 1970
<code>#{GATHERDATE}</code>	Einstellungsdatum
<code>#{GATHERTIME}</code>	Einstellungszeit
<code>#{PUBLISHDATE}</code>	Veröffentlichungsdatum
<code>#{KEYWORD}</code>	Keywords
<code>#{MAINFILE}</code>	Name der ggf. konvertierten Hauptdatei des DocMe Dokuments. In Klassen vom Typ "Only Meta" und "Both" ist bzw. kann <code>#{MAINFILE}</code> leer sein und dient so zur Qualifizierung ob Links gesetzt werden sollen.
<code>#{ORIGEXT}</code>	Fileextension der Originaldatei
<code>#{ORIGFILE}</code>	Originaldatei, falls mit am PublishD verfügbar, wenn nicht, identisch mit <code>#{MAINFILE}</code>
<code>#{PUBLISH}</code>	Veröffentlichungsdatum in Sekunden von 1970
<code>#{PUBLISHDATE}</code>	Veröffentlichungsdatum
<code>#{PUBLISHTIME}</code>	Veröffentlichungszeit
<code>#{TIME}</code>	Erstellungszeit, identisch mit <code>#{DOCTIME}</code>
<code>#{TITLE}</code>	Titel
<code>#{TYPE}</code>	Fileextension von <code>#{MAINFILE}</code>



Anhang

<p><code>#{USER0}</code> bis <code>#{USERx}</code></p>	<p>Dokumentenvariablen, werden individuell vom Administrator angelegt und Klassen zugeordnet (gemappt)</p>
--	--

8.1.2 Indexvariablen

Variable	Beschreibung
<code>#{BODY}</code>	Nächstes Template wird aufgerufen
<code>#{BODY:x}</code>	<p>Das Argument x ist der Name eines existierenden Index, der als Subindex integriert werden soll</p> <p>Bsp: <code>#{BODY:NAME_DES_SUBINDEX}</code></p>
<code>#{DOCCOUNT}</code>	Anzahl der Dokumente eines Index
<code>#{DOCTOTAL}</code>	Anzahl der Dokumente eines Index
<code>#{INDEX}</code>	Name des Index, identisch mit <code>#{INDEXNAME}</code> und <code>#{INDEXNAME:-1}</code>
<code>#{INDEXFILE}</code>	Pfad ab DocRoot mit Indexfilename
<code>#{INDEXNAME}</code>	Name des Index, identisch mit <code>#{INDEX}</code> und <code>#{INDEXNAME:-1}</code>
<code>#{INDEXNAME:x}</code>	<p>Die Argumente x sind positive und negative Ganzzahlen</p> <p>Bsp. Die Index Vererbung sei "Hauptvorlage.Bereichsvorlage.Abtteilungsindex"</p> <p><code>#{INDEXNAME:1}</code> liefert "Hauptvorlage"</p> <p><code>#{INDEXNAME:2}</code> liefert "Bereichsvorlage"</p> <p><code>#{INDEXNAME:3}</code> liefert "Abteilungsindex"</p> <p><code>#{INDEXNAME:4}</code> liefert ""</p> <p><code>#{INDEXNAME:-1}</code> liefert "Abteilungsindex" (ist identisch mit <code>#{INDEX}</code> und <code>#{INDEXNAME}</code>)</p> <p><code>#{INDEXNAME:-2}</code> liefert "Bereichsvorlage"</p> <p><code>#{INDEXNAME:-3}</code> liefert "Hauptvorlage"</p> <p><code>#{INDEXNAME:-4}</code> liefert ""</p> <p><code>#{INDEXNAME:0}</code> liefert immer ""</p>
<code>#{INDEXPATH}</code>	Liefert den Pfad und den Dateinamen der Indexdatei



DocMe Information Modelling

Anhang

<code>\${INDEXPATHONLY}</code>	Liefert den Pfad zur Indexdatei
<code>\${INDEXROOT}</code>	Liefert den Pfad zur Ablage der Indices, z.B. "/opt/docme/publishd/docs"
<code>\${INDEXURL:x}</code>	Das Argument x ist der Name eines Index Liefert den WebServer Pfad zum Index zurück Bsp: <code>\${INDEXURL:MeinIndex}</code> liefert /indexablage/meinindex.html zurück.
<code>\${META0}</code> bis <code>\${METAx}</code>	<code>\${META0}</code> bis <code>\${METAx}</code> Indexvariablen, werden individuell pro Index angelegt bzw. vererbt
<code>\${NEW}</code>	Fügt den Code des New Marker Templates ein, so lange das Dokument nicht älter als die definierte Zeitangabe ist
<code>\${NEWEST}</code>	Sekunden von 1970 des jüngsten Dokuments eines Index
<code>\${NEWESTDATE}</code>	<code>\${DOCDATE}</code> des jüngsten Dokuments eines Index
<code>\${NEWESTTIME}</code>	<code>\${DOCTIME}</code> des jüngsten Dokuments eines Index
<code>\${OLDEST}</code>	Sekunden von 1970 des ältesten Dokuments eines Index
<code>\${OLDESTDATE}</code>	<code>\${DOCDATE}</code> des ältesten Dokuments eines Index
<code>\${OLDESTTIME}</code>	<code>\${DOCTIME}</code> des ältesten Dokuments eines Index
<code>\${PARTCOUNT}</code>	Anzahl der Index Seiten eines Divided Index
<code>\${PARTDOCCOUNT}</code>	Anzahl der Dokumente pro Index Seite eines Divided Index
<code>\${PARTFROM}</code>	Laufende Nummer mit der eine Index Seite eines Divided Index startet



<code>\${PARTMAIN}</code>	URL des Divided Index
<code>\${PARTNAME}</code>	Durch das Dividekriterium gefundener Indexeintrag, identisch mit <code>\${REALPARTNAME}</code>
<code>\${PARTNEWEST}</code>	<code>\${DOCDATE}</code> des jüngsten Dokuments des Index eines Divided Index
<code>\${PARTNEXT}</code>	URL der nächsten Index Seite eines Divided Index
<code>\${PARTNUM}</code>	Laufende Nummer der Einträge eines Index
<code>\${PARTOLDEST}</code>	<code>\${DOCDATE}</code> des ältesten Dokuments des Index eines Divided Index
<code>\${PARTPREV}</code>	URL der vorhergehenden Index Seite eines Divided Index
<code>\${PARTTO}</code>	Laufende Nummer mit der eine Index Seite eines Divided Index endet
<code>\${PARTTOTAL}</code>	Summe der Index Seiten eines Divided Index
<code>\${REALPARTNAME}</code>	Durch das Dividekriterium gefundener Indexeintrag, identisch mit <code>\${PARTNAME}</code>
<code>\${SUBINDEXCOUNT}</code>	Anzahl der Subindices
<code>\${TEMPLATE}</code>	Liefert den Namen des Entry, Body oder Index Templates Nicht nutzbar für Marker oder Link Templates.
<code>\${UPDATE}</code>	Sekunden seit dem letzten Updates des Index
<code>\${UPDATEDATE}</code>	Datum des letzten Updates des Index
<code>\${UPDATETIME}</code>	Zeit des letzten Updates des Index



Sonstige Variablen

Beinhalten sonstige Variablen zu Hooks, Navigationen, Encoding und sonstige DocMe Funktionen und Aufrufen

Variable	Beschreibung
<code>\${CONVNAME}</code>	Name des Converter Scripts
<code>\${HOOKCMDNAME}</code>	Name des Hook Scripts
<code>\${LEADTEXT/x:y}</code>	Das Argument x ist die Anzahl maximaler Zeichen auf welche die Zeichenfolge y gekürzt werden soll. Das Letzte Wort wird immer vollständig geliefert. Bsp: <code>\${LEADTEXT/14:Dieser Text soll gekürzt werden.}</code> liefert "Dieser Text"
<code>\${JAVASCRIPT:x}</code>	Führt während der Erstellung von Indices und während der Ausführung von StyleMe Scripten den JavaScript Code x aus. Variablen und Funktionen sind in anderen JAVASCRIPT Blöcken wieder verfügbar. Das Argument muss gültigen Java Script Code enthalten.
<code>\${JAVASCRIPT/LOCAL:x}</code>	Führt während der Erstellung von Indices und während der Ausführung von StyleMe Scripten JavaScript Code x aus. Variablen und Funktionen sind nur im aktuellen Block gültig. Das Argumente muss gültigen Java Script Code enthalten.
<code>\${LOADFILE:x}</code>	Das Argument x ist die zu ladende Datei. Bsp: <code>\${LOADFILE:/opt/docme/publishd-demo-online/docs/HTMLSCHNIPSEL.RAW}</code>
<code>\${LOCAL}</code>	Systemdatum des Servers in Sekunden von 1970
<code>\${LOCALDATE}</code>	Systemdatum des Server
<code>\${LOCALTIME}</code>	Systemzeit des Server
<code>\${NAVIID}</code>	Identifizier des derzeitigen Navigationsknotens
<code>\${NAVIID:x}</code>	Die Argumente x sind positive und negative Ganzzahlen Bsp. Die Navigation sei "Beispiele.Karriere.Schueler"



Anhang

	<pre> \${NAVIID:1} liefert "Beispiele" \${NAVIID:2} liefert "Karriere" \${NAVIID:3} liefert "Schueler" \${NAVIID:4} liefert "" \${NAVIID:-1} liefert "Schueler" \${NAVIID:-2} liefert "Karriere" \${NAVIID:-3} liefert "Beispiele" \${NAVIID:-4} liefert "" \${NAVIID:0} liefert immer "" </pre>
<code>\${NAVINAME}</code>	Anhand des NavigationEntryTemplates erstellter Navigationspfad des aktuellen Dokuments oder Indizes
<code>\${SQL:x}</code>	<p>Führt während der Erstellung von Indices und während der Ausführung von StyleMe Scripten den SQL Befehl x aus</p> <p>Beispiele:</p> <p>1.) <code>\${SQL:SELECT IF('\${USER1}' = 'TRUE', '\${TITLE}', '')}</code></p> <p>2.) <code>\${SQL:SELECT IF('\${MAINFILE}' = '', 'Nur Meta', 'Standard')}</code></p>
<code>\${SERVERNAME}</code>	Name des Servers
<code><EMBED></code>	<p>Embed tag zum Qualifizieren von Content, der über das DocMe Browser Plugin pflegbar sein soll</p> <p>Bsp:</p> <pre> <EMBED type="application/x-arago-docme" DocId="\${DOCID}" DocMeDocInfo="Autor \${AUTHOR} Titel \${TITLE} Datum \${DOCDATE} Klasse \${CLASS}" width="0" height="0" hspace="0" vspace="0"> </pre>
<code><!-- #include --></code>	<p>Beispielaufruf am Apache WebServer zum dynamischen includieren von Content</p> <pre> <!-- #include virtual="\${DOCDIR}/\${MAINFILE}" --> </pre>
-	<p>Liefert Kleinbuchstaben</p> <p>Bsp: <code>\${-TITLE}</code></p> <p>"Ä ' Sonderz."->"ä ' sonderz."</p>
+	<p>Liefert Grossbuchstaben</p> <p>Bsp: <code>\${+TITLE}</code></p>



Anhang

	"Ä ' Sonderz."->"Ä ' SONDERZ."
&x&	Liefert HTML encoding Bsp: \${&TITLE&} "Ä ' Sonderz."->"Ä ' Sonderz."
%x%	Liefert URL encoding Bsp: \${%TITLE%} "Ä ' Sonderz."->"Ä%20'%20Sonderz."
\x\	Liefert Shell encoding Bsp: \${\TITLE\} "Ä ' Sonderz."->"Ä\ \' Sonderz."
'	escaped alle Singelquots Bsp:\${'TITLE} "Ä ' Sonderz."->"Ä \' Sonderz."
"	escaped alle Doublequots Bsp:\${"TITLE} 'Ä " Sonderz. '->'Ä \' Sonderz.'

8.2 Hooks

8.3 Hooks

8.3.1 Class Tree Hooks

Hook	Beschreibung
Pre Process Doc	Verarbeitungsschritt, bevor ein Dokument vom GatherD Server im DocMe Archiv abgelegt wird. Beispiel: Alle Dokumente sollen, bevor sie verarbeitet werden von einer externen Anwendung auf Viren überprüft werden.
Doc Received	Verarbeitungsschritt, nachdem ein Dokument von einem GatherD Server oder vom Autor über den DocMe Manager Client, also direkt per DocMe Protokoll übermittelt wurde. Dieser Verarbeitungsschritt stellt den Beginn des Publikationsprozesses für Dokumente dar, die



Anhang

	<p>über das interne DocMe Protokoll von einem GatherD Server oder von einer DocMe Client Anwendung empfangen wurden.</p> <p>Beispiel: Zur Unterstützung der Betriebssicherheit der Systeme könnte hier z.B. ein Hook über das Limitation Modul hinausgehende Überprüfungen der empfangenen Dokumente durchführen. Zusätzlich könnte der Hook den Administrator per mail über ein solches Ereignis informieren.</p> <p>Verarbeitungsschritt, nachdem ein Dokument per FTP oder per mail in das ‚DocSource‘ Verzeichnis des GatherD eingestellt wurde und vom GatherD erfasst wurde.</p>
<p>G Doc Gathered</p>	<p>Dieser Verarbeitungsschritt stellt den Beginn des Publikationsprozesses für Dokumente dar, die über externe Transferanwendungen wie z.B. FTP oder mail in das DocRoot Verzeichnis des GatherD eingestellt wurden.</p>
<p>G Doc Released</p>	<p>Verarbeitungsschritt, nachdem ein Dokument von einem für die Freigabe autorisierten User, einem Redakteur, freigegeben wurde.</p> <p>Beispiel: Mail an den Autor, ob sein Dokument vom Redakteur freigegeben wurde.</p>
<p>G Doc Rejected</p>	<p>Verarbeitungsschritt, nachdem ein Dokument von einem für die Freigabe autorisierten User, einem Redakteur, abgelehnt wurde. Das Dokument wird damit nicht publiziert.</p> <p>Beispiel: Mail an den Autor, dass sein Dokument vom Redakteur abgelehnt wurde.</p>
<p>G Doc Delivered</p>	<p>Verarbeitungsschritt, nachdem ein Dokument an einen Publisd oder GatherD Server publiziert wurde.</p> <p>Nach diesem Verarbeitungsschritt ist der Publizierungsprozess am GatherD beendet.</p> <p>Beispiel:</p>









Anhang

	<p>Wenn Autoren für international arbeitende Unternehmen tätig sind, wird ein Dokument oft an verschiedene GatherD bzw. PublishD weltweit verteilt. So kann ein Hook zur automatischen Beantwortung von Rückfragen z.B. eine Protokolldatei speziell für bestimmte Dokumente erstellen. Diese Protokolldatei kann wiederum im web über DocMe zur Verfügung gestellt werden.</p>
<p>G Doc Queued</p>	<p>Verarbeitungsschritt, bevor ein Dokument konvertiert wurde.</p>
<p>G Doc Converted</p>	<p>Verarbeitungsschritt, nachdem ein Dokument konvertiert wurde.</p> <p>Beispiel: Nach der Konvertierung, können sich weitere Prozesse darauf verlassen, ein bestimmtes Dateiformat vorzufinden. Ein Hook könnte anschließend auf Grund von bestimmten Werten in den Metainformationfeldern HOST Anfragen generieren und dessen Ergebnisse in die Datei integrieren. Im Intranet bietet es sich an Budgetwerte, Absatzwerte etc. aus der ERP Anwendung automatisch in die Abteilungsseiten zu integrieren.</p>
<p>G Doc Part Converted</p>	<p>Verarbeitungsschritt, nach dem ein Konvertierschritt auf dem Weg vom Dateiausgangsformat zum Dateizielformat beendet wurde.</p> <p>Beispiel: Soll vom Dateiausgangsformat ZIP in das Dateizielformat HTML konvertiert werden, so sind zwei Konvertierschritte notwendig. 1. Konvertierschritt ZIP entpacken in Word Format 2. Konvertierschritt Word in HTML Format</p>
<p>G Doc Republish</p>	<p>Verarbeitungsschritt, nach einem erneuten Publizieren eines bestehenden Dokuments. Entspricht im Prozessablauf dem Verarbeitungsschritt ‚Doc Delivered‘.</p> <p>Beispiel: Wurde ein Dokument an einem PublishD nach</p>







Anhang

	Erreichen des Expiredate gelöscht, so kann es bei Bedarf durch ein Republish aus dem GatherD Archiv erneut publiziert werden.
 PreProcessDoc	<p>Verarbeitungsschritt, bevor ein Dokument vom PublishD Server im Dokumentenverzeichnis abgelegt wird.</p> <p>Beispiel: Alle Dokumente sollen, bevor sie verarbeitet werden von einer externen Anwendung auf Viren überprüft werden.</p>
 Pre Publish Document	<p>Verarbeitungsschritt bevor ein Dokument am PublishD publiziert wird.</p> <p>Dieser Verarbeitungsschritt stellt den Beginn des Publikationsprozess am PublishD Server dar.</p> <p>Beispiel: Das vom GatherD übermittelte Dokument kann über diesen Verarbeitungsschritt für jeden PublishD individuell mit zusätzlichen Informationen versehen werden.</p>
 PreContent Handler	Verarbeitungsschritt, bevor SyleMe, Linkmanagement und Keywords druchgeführt wird.
 PostContent Handler	Verarbeitungsschritt, nachdem SyleMe, Linkmanagement und Keywords druchgeführt wurde.
 Post Publish Document	<p>Letzter Verarbeitungsschritt nach dem ein Dokument am PublishD publiziert wurde.</p> <p>Beispiel: Oft möchten Autoren, Redakteure oder Administratoren informiert werden, wenn ein Dokument publiziert wurde. In diesem Fall würde ein Hook zur Versendung eines mails den Verarbeitungsschritt ‚Post PublishDocument‘ nutzen.</p>
 Delete Document	<p>Verarbeitungsschritt, bevor ein Dokument vom PublishD gelöscht wird.</p> <p>Beispiel:</p>





DocMe Information Modelling

Anhang



	<p>Beim Löschen von Dokumenten müssen auch evtl. Suchmaschinen über das beabsichtigte Entfernen von Dokumenten informiert werden um möglichst zeitgleich zum aktuellen Stand die Suchmaschinenindexe entsprechend anzupassen.</p>
<p> Index Add Document</p>	<p>Verarbeitungsschritt, nachdem ein Dokument einem Index zugefügt wurde.</p> <p>Beispiel: Über diesen Verarbeitungsschritt lässt sich über den Versand von mails kontrollieren ob alle Dokumente bereits im Index eingefügt wurden.</p>
<p> Index Del Document</p>	<p>Verarbeitungsschritt, bevor ein Dokument (Link) aus einem Index gelöscht wird.</p> <p>Beispiel: Siehe Verarbeitungsschritt Delete Document</p>
<p> Publish Error</p>	<p>Ereignis, wenn während des Publikationsprozess am Publisher ein Fehler auftritt.</p> <p>Beispiel: Mail an den IT-Administrator.</p>
<p> Deadlink Doc</p>	<p>Das Ereignis tritt auf, wenn innerhalb des DocMe Datenbestandes ein DeadLink festgestellt wird.</p> <p>Siehe auch im Modul 'Basic Options' im Karteireiter 'Extended'.</p> <p>Beispiel: Mail mit Informationen zu dem betreffenden von DocMe verwalteten Dokument mit einem Dead Link an den IT-Administrator.</p>



8.3.2 Index Hooks (Modul Indices)

Hook	Beschreibung
 Index Divide	Verarbeitungsschritt, nach der Generierung jedes einzelnen Divided Parts eines Divided Index.
 Index Generated	Verarbeitungsschritt, nach dem ein Index neu generiert wurde.

8.3.3 Globalhooks (Modul Basic Opitons)

Hook	Beschreibung
 Advertisement Changed	Verarbeitungsschritt, nach dem ein Änderung am DocMe Advertisement Modul in der DocMe Datenbank gespeichert wurde.
 Navigation Changed	Verarbeitungsschritt, nach dem ein Änderung am DocMe Navigation Modul in der DocMe Datenbank gespeichert wurde.



9 Literatur

Weitere Informationen finden Sie in den folgenden Dokumentationen

- **Information Modelling**
Allgemeine Informationen zum Information Modelling
von Hans-Christian Boos, Vorstand
arago Institut für komplexes Datenmanagement AG
- **DocMe Administrator Tutorial**
Dokumentation zur Nutzung des DocMe Administrator Client, dem
Werkzeug für Administratoren
von Peter Steffan
arago Institut für komplexes Datenmanagement AG
- **DocMe Manager Tutorial**
Dokumentation zur Nutzung des DocMe Manager Client, dem
Werkzeug für Autoren und Redakteure
von Peter Steffan
arago Institut für komplexes Datenmanagement AG
- **StyleMe**
Detaillierte technische Dokumentation zur Nutzung der StyleMe
Skriptsprache
von Michael Reutlinger
arago Institut für komplexes Datenmanagement AG
- **DocMe FactSheet**
Allgemeine Beschreibung der DocMe Funktionen
arago Vertrieb

Bitte wenden Sie sich an Ihren arago Kundenbetreuer unter Tel:

0 69/4 05 68-0. Wir helfen Ihnen gerne weiter.