# Die Suche in OpenCms

Die Suche in OpenCms basiert auf dem Lucene Framework. Die Jakarta Lucene ist eine leistungsstarke, voll funktionsfähige Text-Such-Engine, geschrieben in Java. Sie ist eine Technologie, die für fast jede mögliche Anwendung verwendbar ist, die Volltextsuche erfordert, besonders eine Cross-Plattform. Lucene ist allerdings keine fertige Suchmaschine, sondern stellt Klassen und Funktionen zur Verfügung, um für beliebige Projekte eine eigene Suchmaschine zu bauen.

Lucene unterstützt standardmäßig reichhaltige Suchoptionen. Allerdings besteht auch die Möglichkeit, eine eigene Suchsyntax zu entwickeln.

## 1. Operatoren

Welche Suchmöglichkeiten und Operatoren bietet die Lucene Engine in OpenCms.

| Operatoren | Lucene Search Engine in OpenCms |
|---|---|
| **Fuzzy Suche**<br><br>undeutliche Suche zu formulieren mit Tilde-Symbol ("~")<br><br>Beispiel: seit~liefert: Seite, seinen, weitere, Leiter... | Ja |
| **Boolesche Suche**<br><br>Verknüpfen von Wörtern mit UND, OR, AND<br>**Beispiel:**<br>*Guten AND Tag*<br>+ für soll vorkommen<br><br>**Beispiel:**<br>oder aber *+Tag + guten*<br>NOT nicht vorkommen | Ja |
| **Proximity-Suche/Distanzsuche**<br><br>*"Guten wiedersehen"~10*<br>(zeigt Ergebnisse, indem die beiden Wörter maximal 10 Wörter zwischen sich haben). | Ja |
| **Wildcard**<br><br>Platzhalter (?) z. B. *te?t* | Ja, aber nicht am Anfang des Wortes:<br><br>z. B. ?wort, ?gang **geht nicht**<br>z. B. Haus?, Baum? **geht** |

| | |
|---|---|
| **Feldsuche**<br><br>Lucene unterstützt Felddaten.<br><br>Beispiel: *"title:index AND text: Tag".* | Ja, z. B. Suche kann über Volltext oder Seitentitel gesteuert werden.<br><br>Dazu muss das Feld indiziert sein. |
| **Bereichssuche**<br><br>Ein Bereich kann die Zeit zwischen Erstelldatum und Bearbeitungsdatum sein oder zwischen 1. Januar 2010 und 8. Januar 2010 inklusive Start- und Enddatum.<br>**oder Beispiel:**<br>Findet alle Dokumente ohne den Begriff Wissensmanagement. | Ja, ist möglich.<br><br>created: [20100101 TO 20100108]<br><br>lastmodified: [20100101 TO 20100108]<br><br>title:{Wissensmanagement} |
| **Verstärkungsfaktor**<br><br>um einen Verstärkungsfaktor zuzuordnen, wird das "^"-Symbol verwendet<br><br>Tag^4 du = Tag ist 4 x stärker gewichtet | Ja |
| **Weitere Funktionen** | |
| **Tokenisierung**<br><br>Zerlegen eines oder mehrerer Texte in kleine Einheiten, so genannte Tokens. Diese werden anschließend gespeichert in der Token-Dokument-Beziehung. | Ja |
| **Indexierung**<br><br>Mit Indexierungsverfahren wird durch die Ermittlung von Worthäufigkeiten und Wortrelevanz eine Auswahl getroffen und somit Wörter in den Index aufgenommen.<br><br>Anhand des erstellten Index kann dann die Volltextsuche, wie später beschrieben, effizient durchgeführt werden. | Ja<br><br>Bei der Installation von OpenCms erstellt die Lucene Engine automatisch aus den Inhalten in der Datenbank einen Index.<br><br>Neue Dokumente werden sofort indiziert.<br><br>Änderungen werden auch sofort berücksichtigt.<br>Lucene untersucht den von OpenCms übergebenen Text und schreibt Informationen über die Häufigkeit der im Text enthaltenen Worte sowie deren Relevanz in den Index.<br>Das hat den Vorteil, dass bei einer Suche nicht das gesamte Dokument durchsucht werden muss (bei großen Dokumenten würde das sehr lange dauern), sondern nur nach dem Wort in dem Index geschaut werden muss und sofort die Trefferstelle und Häufigkeit feststeht. |

| | |
|---|---|
| **Anfrageanalyse oder effiziente Suche**<br><br>Analyse von Text-, PDF-, Word-, Excel- und OpenOffice-Dokumenten. | Für jedes Dokumentformat existiert ein „Analyzer“. Dieser durchsucht und indexiert in optimierter Weise verschiedene Dokument-formate. OpenCms unterstützt das. |
| **Rekursives Crawling**<br><br>Suche durch mehrere Verzeichnisse und externe Laufwerke | Nein, dazu muss die Lucene Engine erweitert werden. |
| **Paging**<br><br>Blättern durch Ergebnislisten. | Ja |
| **TagCloud**<br><br>Eine Schlagwortwolke –ist eine Methode zur Informationsvisualisierung, bei der eine Liste aus Schlagworten, oft alphabetisch sortiert, flächig angezeigt wird, wobei Wörter, die häufiger gesucht werden größer angezeigt werden als weniger gesuchte. | Standardmäßig gibt es sowas aber in OpenCms noch nicht. |
| **Semantische Suche – Beispiel**<br><br>Eine Suchanfrage nach ‚Baum' an eine bedeutungserschließende Suchmaschine verwendet zur Suche auch Begriffe, die im Zusammenhang mit ‚Baum' genannt werden, auch wenn sie in der Anfrage selbst nicht genannt werden. Es werden Ergebnisse, die die Wörter (z. B. Ahorn, Eiche, Linde) beinhalten, angezeigt. | Nein |

## 2. Verhalten von OpenCms und Lucene

1. OpenCms erkennt, dass eine neue Ressource in die Suche aufgenommen werden soll.
2. OpenCms schaut, um welche Ressourcenart es sich handelt und extrahiert suchrelevante Daten - bei XML-Content z. B. Titel, Body, Dateipfad des Anhangs etc., bei Bildern z. B. Bildname, Bildpfad etc. - als Text.
3. OpenCms ruft Lucene auf und übergibt die aus der neuen Ressource suchrelevanten extrahierten Daten; dabei kann OpenCms noch angeben, ob diese eine besondere Gewichtung - "boost" - haben sollen. Außerdem wird Text, welcher aus wenigen Einzelworten besteht, höher gewichtet, so z. B. beim Titel, der oft nur aus einem Wort besteht, im Gegensatz zum Content, der aus Tausenden Wörtern bestehen kann.
4. Lucene untersucht den von OpenCms übergebenen Text und schreibt Informationen über die Häufigkeit der im Text enthaltenen Worte sowie deren Relevanz in den Index.

# 3. Indexierung

Die Indexierung dient der Aufbereitung des Suchraums und wird typischerweise zu Beginn und bei Änderungen des Suchraums durchgeführt. Anhand des erstellten Index kann dann die Volltextsuche effizient durchgeführt werden.

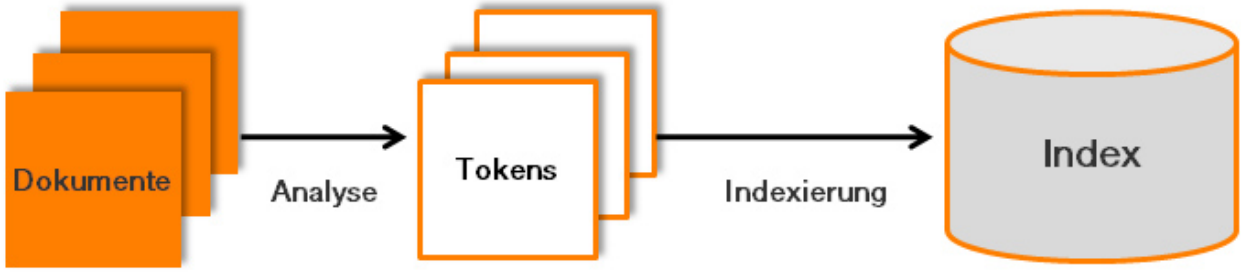


**Abbildung: Allgemeiner Aufbau einer Volltextindexierung**

Zunächst zum Begrifflichen: Ein Index mehrerer Dokumente bezeichnet im Allgemeinen ein Abbild von Elementen (Tokens oder Terme) dieser Dokumente auf eine Liste der Dokumente, in denen das jeweilige Token vorkommt. Unter der Indexierung versteht man die Generierung eines Index aus einem oder mehreren Texten. Der Indexierungsprozess in Lucene benötigt die folgenden Grundkonzepte mit entsprechenden Java-Klassen:

- **Dokument:** Die von Lucene zu indexierenden Dokumente sind in der Klasse org.apache.lucene.document.Document abstrahiert. Ein Dokument kann hierbei alles von einer Buchseite über eine Text- oder HTML-Datei bis zu einem Datenbankeintrag umfassen.
- **Feld:** Ein Dokument besteht aus mehreren (benannten) Feldern (org.apache.lucene.document.Field), die auf Wunsch indexiert werden können. Diese Aufteilung ermöglicht eine feiner strukturierte Suche. Beispielsweise könnte man bei einem Word-Dokument neben dem eigentlichen Text als zusätzliche Felder Autor und Beschreibung indexieren.

**Textvorverarbeitung (Analyse):** Der zu indexierende Text aus den Feldern der Dokumente wird nicht direkt in den Index geschrieben, sondern zunächst von einem sogenannten Analyzer, einem Objekt vom Typ org.apache.lucene.analysis.Analyzer, geparst. Dadurch wird der in den Index einzutragende Text vorverarbeitet - das kann von so trivialen Dingen wie Normalisierung auf Kleinbuchstaben über das Weglassen bestimmter trivialer Wörter bis hin zum Eintrag nur des Wortstammes reichen. Alle Suchanfragen werden dann auch vom entsprechenden Analyzer vorverarbeitet.

**Indexerstellung**: Der eigentliche Index wird von der Klasse org.apache.lucene.index.IndexWriter geschrieben. Dazu muss man angegeben, wo der Index gespeichert werden soll - sei es direkt im Dateisystem, in einer Datenbank oder nur im Arbeitsspeicher - und welcher Analyzer verwendet werden soll. In den Index lassen sich dann mehrere Dokumente eintragen. Auch können mehrere Indizes zusammengefügt und optimiert werden.

Indexierung hat den Vorteil, dass bei einer Suche nicht das gesamte Dokument durchsucht werden muss, sondern nur nach dem Wort in dem Index geschaut werden muss und sofort die Trefferstelle und Häufigkeit feststeht. Bei großen Dokumenten bzw. Dokumentenmengen würde das sehr lange dauern und die Performance verschlechtern.

# 4. Die eigentliche Suche

1. Benutzer gibt in OpenCms einen Suchbegriff ein.
2. OpenCms gibt den Begriff an Lucene weiter.
3. Lucene bereitet den Begriff ggf. auf (Operatoren wie AND, OR, Wildcards etc.).
4. Lucene holt sich Informationen über den Begriff aus dem Index und gibt eine Liste von Treffern zurück.
5. Die Liste ist dann so geordnet, dass relevante Treffer am Anfang stehen.

## 4.1. Darstellung der Suchergebnisse

- Standardmäßig werden in OpenCms die Suchergebnisse nach relevanten Treffern gelistet.
- Standardmäßig ist das Suchwort markiert oder hervorgehoben.
- Quelle ist ersichtlich, wenn die Suche über mehrere Websites geht.
- Standardmäßig wird die Trefferanzahl angezeigt.

## 4.2. Suchvorschläge

Standardmäßig macht OpenCms keine Suchvorschläge. Dazu muss der Suchindex oder die Metadatenverwaltung an die Sucheingabemaske angebunden werden. Nur so wird während des Schreibens in das Suchfeld ein Wort aus dem Index oder der Metadatenverwaltung vorgeschlagen, das dort indexiert bzw. aufgenommen wurde. Den Vorschlägen liegt eine Technologie zugrunde, die sich daran orientiert, was der Nutzer mit seiner Suchanfrage am ehesten gemeint haben könnte. Die bereitgestellten Vorschläge laufen dabei im Hintergrund mit und werden erst eingeblendet, wenn der Nutzer das Drop-Down-Menü direkt unter dem Suchfenster öffnet. Die angezeigten weiteren Begriffe und Themenvorschläge sollen dabei helfen, das gesuchte Thema noch präziser einzugrenzen. Ziel wäre es, den User bei seiner Suche schneller und bequemer ans Ziel zu führen.

# 5. Option Solr

Für eine hochskalierbare Suche kommt die führende Open Source Software für Enterprise Search Applications Solr zum Einsatz. Solr ist ein Enterprise Search Server auf der Basis der Lucene Java-Bibliothek und kann somit unabhängig vom Portal betrieben werden. Dies führt zu einer besseren Performance, Skalierbarkeit, Lastverteilung und Wartbarkeit von Solr. Zu den Schnittstellen von Solr gehört beispielsweise eine HTTP-API, mit der Dokumente hinzugefügt, geändert oder gelöscht werden können. Zu den weiteren Funktionen gehören XML/HTTP und JSON APIs, Hit-High-Lighting, facettierte Suche, Caching, Replikation sowie eine Web-Administrations-Oberfläche.

Im Endeffekt steht Solr zwischen Client-Anwendung (z. B. OpenCms) und Lucene. Der Vorteil ist, dass man dadurch die Suche vereinheitlichen kann. Man kann OpenCms an Solr anschließen und dadurch einen Suchindex erzeugen lassen, welcher Dokumente beider Systeme enthält.

## 5.1. Vorteile von Solr

- **Individualisierung:** Flexible Anpassung der Suchalgorithmen.
- **Transparenz:** offene API, Protokolle, Formate und Suchalgorithmen.
- **Portabilität:** Lucene/Solr läuft auf allen Plattform-Systemen, welche Java unterstützen; die erstellten Indizes sind unabhängig vom Plattform-System und können somit ohne Probleme zwischen verschiedenen Plattformen ohne Anpassungen portiert werden.
- **Sicherheit:** Solr wird bereits in geschäftskritischen Anwendungen bei über 4.000 Unternehmen weltweit eingesetzt, darunter Branchengrößen wie MySpace, AOL, Nike, LinkedIn oder Monster.com.
- **Performance:** Schnelle Antwortzeiten (intern oft unter 50 ms), da keine Datenbankzugriffe nötig sind; verbessert die Geschwindigkeit Ihrer Seite und gleichzeitig das Ranking in Suchmaschinen.
- **Skalierbarkeit:** Auch bei wachsenden Datenbeständen sind keine überproportionalen Investitionen in Hardware nötig - dies schont Ihre IT-Budgets; große Anwender können von den Replikationsmöglichkeiten und Load-Balancer-Systemen für Solr profitieren.

**comundus realisiert auf Basis von OpenCms Mitarbeiterportale und Internetauftritte. In beiden Unternehmenslösungen spielt die Suche eine zentrale Rolle. Eine intelligente Suchfunktion kann darüber entscheiden, ob aus Besuchern Kunden werden und Mitarbeiter effizient und effektiv ihr Wissen teilen können.**

comundus

**comundus GmbH**

Heerstraße 111
D-71332 Waiblingen
Telefon: +49 7151 96528-0
E-Mail: info@comundus.com
Internet: www.comundus.com