JAVA Mag inkl. CD

OpenCMS: Mehr Struktur im Web »68

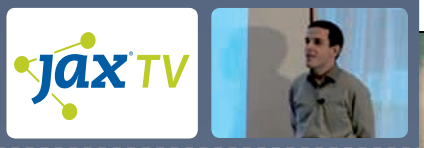
Deutschland €8,50 Österreich €9,80 Schweiz sFr 16,80 4.2011

Java magazin

Java • Architekturen • Web • Agile

www.javamagazin.de

CD-INHALT



Quo vadis Dependency Injection?

Video von Agim Emruli auf der W-JAX 2010

HIGHLIGHTS

Java-Editor
BlueJ
Greenfoot

WEITERE INHALTE

JBoss Seam
OpenCMS
uvm.

Alle CD-Infos ab Seite 3

4 194586 708507 04

Bean Validation
Türsteher für Bohnen »100

Multi-Threading
Parallelisierung leicht gemacht »18

jax 2011
Alle Infos im Heft »35

CDI

Enterprise-Trend für 2011? »38

JavaFX

Java-Raupe – FX-Schmetterling »61

JSF in der GAE

Das Tutorial »74

Software Development Governance

In kleinen Schritten zur besseren Entwicklung »81

Redaktionell pflegbare Webseiten mit OpenCms

Mehr Struktur im Web

Eine redaktionell pflegbare Webseite als Java-Anwendung – das sollte ja kein großes Problem darstellen. Seiteninhalte werden mittels eines O/R Mappers verwaltet, die Auslieferung erledigt ein beliebiges Webframework. Aber Redakteure wollen Inhalte auch im Voraus bearbeiten und erst später freischalten. Versionierung von Änderungen, komfortables Editieren, Berechtigungen auf Inhalte, leichte Durchsuchbarkeit – schnell stößt eine selbstgestrickte Lösung an ihre Grenzen. Da sollte es doch eigentlich schon etwas Fertiges geben?

von Florian Hopf

Content Management dient der Verwaltung von Inhalten, häufig zur Ausgabe auf einer Webseite. Es wird eine möglichst neutrale Struktur der Inhalte angestrebt, um sie in unterschiedlichen Kontexten ausgeben zu können. Beispielsweise kann ein einmal erstellter Newsbeitrag auf einer Webseite in einer Übersichtsliste, in der Detailansicht oder in einem Container mit den neuesten Beiträgen dargestellt werden. Im System existiert der Beitrag genau einmal, nur die Darstellung unterscheidet sich. Dieses Konzept kann noch ausgeweitet werden auf die Art des Mediums, ein Newsbeitrag kann in unterschiedlichen HTML-Varianten, als RSS oder PDF dargestellt werden. OpenCms [1] ist ein weit verbreitetes Java-basiertes CMS, das sich über die Jahre auch in großen Internet- und Intranetanwendungen bewährt hat. Es wird hauptsächlich von der Firma Alkacon [2] aus Köln weiterentwickelt und steht unter der LGPL zur Verfügung. Mit den Jahren hat sich eine große Community um das System gebildet, die zusätzliche Dokumentation bereitstellt und Module und Erweiterungen zur Verfügung stellt [3], [4]. Aktuell ist die Version 7.5.3 verfügbar, für Mai 2011 ist OpenCms 8 angekündigt. OpenCms wird als Webanwendung ausgeliefert und benötigt neben einem Servlet-Container eine Datenbank zur Ablage der Inhalte. Über einen Installations-Wizard werden alle benötigten Tabellen angelegt und mit den initial benötigten Daten befüllt.

Virtuelles Dateisystem

Inhalte werden in OpenCms im so genannten Virtual Filesystem (VFS) verwaltet. Alle Ressourcen (Dateien und Ordner sowie zugehörige Metadaten) sind in der zugrunde liegenden Datenbank abgelegt, können jedoch wie in einem Filesystem dargestellt werden. OpenCms bietet eine komplett browserbasierte Bedienoberfläche, die an bekannte Dateisystembrowser angelehnt ist. **Abbildung 1** zeigt den Workplace Explorer. Im linken Teilfenster wird die Ordnerstruktur der aktuell gewählten Seite angezeigt. Rechts daneben befindet sich das Hauptarbeitsfenster, in dem diverse Operationen auf den Dateien ausgeführt wer-

den können, darunter Bearbeiten, Kopieren und Löschen der Ressourcen und Anzeigen der Versionshistorie.

Die Struktur einer Seite sieht im VFS wie ein Baum im Dateisystem aus. Normalerweise liegt auf der obersten Ebene eine Datei mit dem Namen *index.html* (die jedoch nicht zwingend eine HTML-Datei ist, sondern auch eine JSP oder ein strukturierter Inhalt sein kann). Ordner, die weitere Dateien enthalten, strukturieren die Seite und können genutzt werden, um automatisch eine Navigationsstruktur zu erzeugen. Zusätzlich bestimmt die Position einer Datei im Ordnerbaum den URL, unter dem sie zu erreichen ist. Angenommen, es existiert eine Datei *kontakt.html* im Ordner *unternehmen*. Dann lautet der URL, über den diese Datei zu erreichen ist */unternehmen/kontakt.html*. Diese lesbaren URLs werden einerseits von Suchmaschinen ausgewertet und ermöglichen dem Besucher ein leichtes Einordnen der aktuell angezeigten Seite in den Kontext. Ressourcen im VFS (Dateien und Ordner) bestehen im Wesentlichen aus drei verschiedenen Datensätzen:

- Attribute
- Properties
- Dateiinhalt

Attribute beschreiben die Verwaltungsinformationen zu einer Datei. Dazu gehören neben aus normalen Filesystemen bekannte Informationen wie Erzeugungsdatum oder letzter Bearbeiter auch CMS-spezifische Informationen wie das Gültigkeitsdatum. Ist das Gültigkeitsdatum einer Datei abgelaufen, wird diese vom System nicht mehr nach außen ausgeliefert. So können einzelne Dateien oder Teilbäume der Seite gezielt nur über einen bestimmten Zeitraum zur Verfügung gestellt werden.

Properties sind zusätzliche Metadaten zu Ressourcen und sind als Schlüssel-Wert-Paare abgelegt. Sie bieten eine uniforme Zugriffsmöglichkeit auf Inhalte und werden zum Beispiel verwendet, um den Titel einer Ressource darzustellen. Angenommen, es wird eine Liste von Dateien und Links auf der Seite dargestellt. Das Property *Title* speichert sowohl zu binären Dateitypen wie PDFs als auch für einzelne Seiten einen lesbaren Namen. In der Liste wird dann der Pfad zu der Datei als Link verwendet und das zugehörige Property *Title* als Beschreibung ausgeben. Im Code spielt es keine Rolle, welchen Typ ein Eintrag in der Liste hat, solange das entsprechende Property gesetzt ist. Häufig benutzt werden noch die Properties zur Speicherung der Keywords und Description zur Ausgabe als HTML-Meta-Tags sowie die Naviagtions-Properties zur Einordnung einzelner Ressourcen in eine Navigationsstruktur.

Workflow

Redakteure bearbeiten den Inhalt der Seite immer in einem Offlineprojekt. Wie in **Abbildung 2** angedeutet, kann sich der Stand der Dateien dieses Projekts vom Onlineprojekt, das die Besucher der Webseite sehen, unterscheiden. Ressourcen können nur im Offlineprojekt bearbeitet werden, im Onlineprojekt besteht nur lesender Zugriff. Um die Inhalte vom Offline- in das Onlineprojekt zu überführen, muss ein Publikationsmechanismus angestoßen werden.

Modulentwicklung

Alle zum Betrieb der eigenen Seite notwendigen Dateien wie JSPs, kompilierte Klassen und Bibliotheken werden ebenfalls im VFS verwaltet. Über Module können Anwendungen gruppiert und leicht verteilt werden. Ein Modul entspricht einem Zip-Archiv, das die zugehörigen Dateien sowie eine Manifest-Datei zur Beschreibung der Inhalte enthält. Jars und Klassen werden nach dem Import automatisch an die vom Servlet-Container erwarteten Stellen ins Dateisystem exportiert.

Bedingt durch die Ablage im VFS ist das Entwickeln mit OpenCms etwas aufwändiger als bei einer normalen Webapplikation. Mindestens die zur Darstellung verwendeten JSPs müssen ins VFS importiert werden und

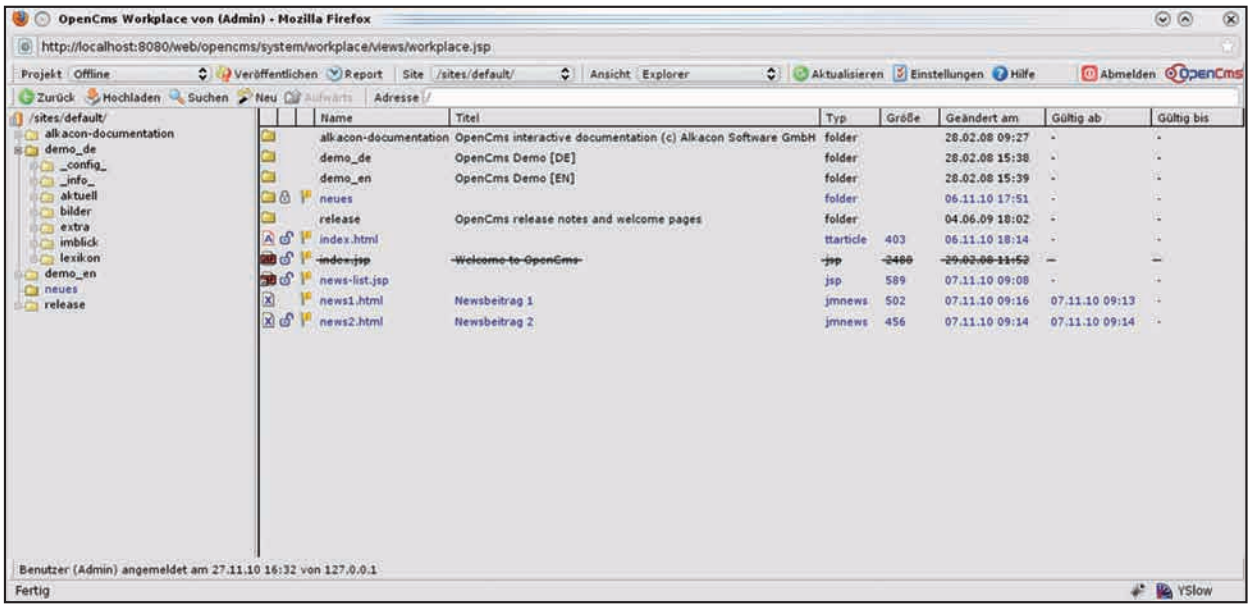


Abb. 1: Workplace Explorer

können nicht direkt im Dateisystem editiert werden. Gängige Build-Mechanismen können damit nicht ohne Weiteres verwendet werden. Mittlerweile existieren jedoch einige Tools, die das Arbeiten per Maven, Ant, NetBeans oder Eclipse erleichtern [5].

Templating

Die Darstellung der Inhalte erfolgt über Java Server Pages. Meist existiert ein Haupt-Template, das das Grundgerüst der Seite festlegt, also Kopfbereich und Fußbereich, die Position der Navigationselemente und

Listing 1

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms"%>

<cms:template element="head">
<html>
  <head>
    <title><cms:property name="Title"/></title>
    <cms:editable/>
  </head>
  <body>
</cms:template>

<cms:template element="foot">
  </body>
</html>
</cms:template>
```

Listing 2

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                                          elementFormDefault="qualified">

  <xsd:include schemaLocation="opencms://opencms-xmlcontent.xsd"/>

  <xsd:element name="NewsElements" type="OpenCmsNewsElements"/>

  <xsd:complexType name="OpenCmsNewsElements">
    <xsd:sequence>
      <xsd:element name="NewsElement" type="OpenCmsNewsElement"
                                    minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="OpenCmsNewsElement">
    <xsd:sequence>
      <xsd:element name="Title" type="OpenCmsString" minOccurs="1"
                                                    maxOccurs="1" />
      <xsd:element name="Date" type="OpenCmsDateTime" minOccurs="1"
                                                    maxOccurs="1" />
      <xsd:element name="Text" type="OpenCmsHtml" minOccurs="1"
                                                    maxOccurs="1" />
      <xsd:element name="Links" type="OpenCmsVarLink" minOccurs="0"
                                                maxOccurs="unbounded"/>
      <xsd:element name="Author" type="OpenCmsString" minOccurs="1"
                                                    maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="language" type="OpenCmsLocale" use="required"/>
  </xsd:complexType>

  <xsd:annotation>
    <xsd:appinfo>
      <defaults>
        <default element="Author" value="${currentuser.fullname}"/>
        <default element="Date" value="${currenttime}"/>
      </defaults>
      <mappings>
        <mapping element="Title" mapto="property:Title" />
        <mapping element="Date" mapto="attribute:datereleased" />
      </mappings>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:schema>
```

des Content-Bereichs. Die seitenabhängigen Inhalte werden über einzelne JSPs ausgegeben, die meist durch konfigurierbare Eigenschaften des aufgerufenen Inhalts bestimmt werden. Ein einfaches Template mit Kopf- und Fußbereich, das das HTML-Grundgerüst bereitstellt, kann wie in Listing 1 angegeben aussehen.

Über die <*cms:template*>-Tags werden Bereiche definiert, die an geeigneten Stellen eingebunden werden können. Der Titel wird automatisch aus dem Property *Title* des aktuell ausgelesenen Inhalts bestimmt. Die Namen *head* und *foot* für die Bereiche sind Konventionen, die von vielen bestehenden Modulen beachtet werden.

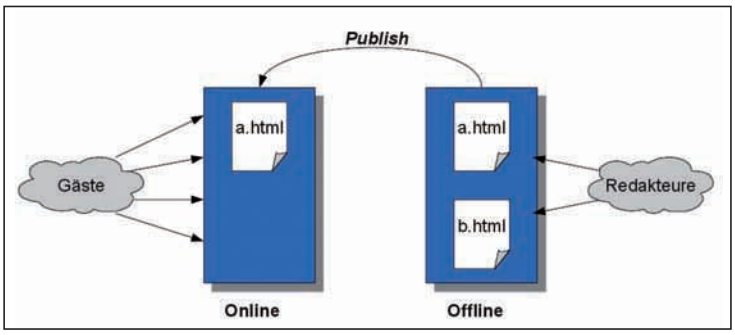


Abb. 2: Online/Offline-Workflow

Strukturierte Inhalte

Die Definition von Inhaltstypen wie Artikeln, Newsbeiträgen oder Veranstaltungshinweisen wird mithilfe von speziellen XML-Schema-Dokumenten vorgenommen. Jedes Element der Schema-Datei entspricht einem Feld des Inhalts, beispielsweise kann ein Inhaltstyp für Artikel Elemente für den Titel, den Vorspann und den Textkörper enthalten. Neben einigen vorgefertigten primitiven Typen wie *String*, *Boolean* oder *Date* können strukturierte Inhalte verschachtelt werden, ein Beispiel ist die Einbindung eines Schemas für Adressen in den strukturierten Inhalt einer Person. Jede registrierte Inhaltsdefinition kann anhand eines Typnamens im System referenziert werden. Redakteure können konkrete XML-Instanzen des Schematyps anlegen, kommen dabei jedoch nicht mit XML in Berührung. Automatisch generierte Editoren ermöglichen eine leichte Pflege der Inhalte. Listing 2 enthält die beispielhafte Definition eines Newsbeitrags mit fünf Elementen: *Title*, *Date*, *Text*, *Links* und *Author*.

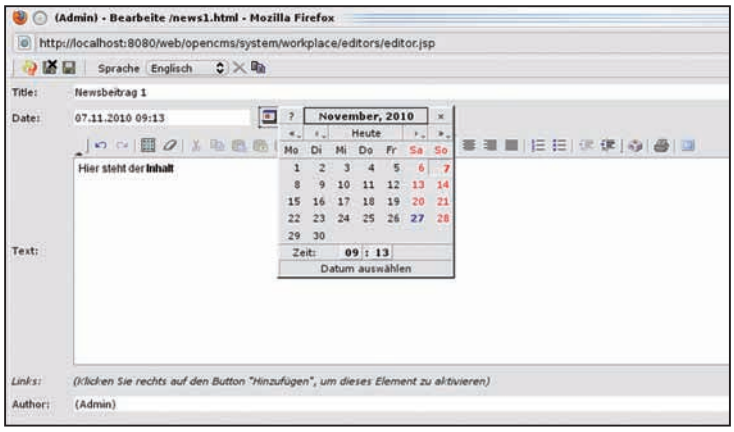


Abb. 3: Structured Content Editor

Links können 0- bis n-mal vorkommen, alle anderen Felder sind genau einmal vorhanden. Im Annotationsbereich sind Metadaten zum Inhaltstyp hinterlegt. Autor und Datum werden mit Default-Werten vorbelegt, die Überschrift wird auf das Property *Title* gemappt, um den uniformen Zugriff zu gewährleisten. Das Datum wird automatisch auf das Erscheinungsdatum abgebildet. Wenn hier ein Zeitpunkt in der Zukunft angegeben wird, ist der Beitrag für Besucher erst zu diesem Datum auf der Seite sichtbar.

OpenCms generiert aus der Schema-Datei den in **Abbildung 3** dargestellten Editor, der unterschiedliche Eingabe-Widgets für die Datentypen enthält. Das Datum wird mittels eines JavaScript-Kalenders eingegeben, Überschrift und Autor sind einzeilige Texteingabefelder. Die Auswahl der Links erfolgt über einen Baum-Selektor, in dem weitere Dateien auf der Seite gewählt werden können. Dazu können auch externe Links gehören, die im System als wiederverwendbare Content-Elemente abgelegt werden. Für den Textinhalt wird durch die Angabe des Typs *OpenCmsHtml* ein Rich-Text-Editor generiert, dessen Funktionalität durch weitere Angaben im Annotationsbereich bestimmt werden kann.

Der Zugriff auf die Daten zur Darstellung erfolgt über Taglibs oder JSTL-Funktionen. Dabei werden die Elementnamen, wie sie im Schema angegeben sind, verwendet, um XPath-artige Zugriffe auf die Daten zu ermöglichen. In Listing 3 wird ein Newseintrag für eine Detailseite aufbereitet.

Die <*cms:include*>-Tags im Kopf- und Fußbereich binden die aus dem konfigurierbaren Template stammenden Bereiche ein. Da hier kein fester Pfad angegeben ist, kann ein beliebiges Template verwendet und so das Layout also einfach ausgetauscht werden. Der Tag <*cms:contentload*> dient dem Laden von Inhalten und wird sowohl für einzelne Datensätze als auch für aggregierte Listen verwendet. Das Attribut *collector* im Zusammenspiel mit den über *params* angegebenen Eigenschaften bestimmt, welche Daten bereitgestellt werden. In diesem Fall wird über das Makro *%(opencms.uri)* die aktuell ausgelieferte Datei eingelesen. <*cms:contentaccess*> stellt den eingelesenen Inhalt in einem speziellen Zugriffsobjekt zur Verfügung. Damit können die Tags der JSTL in Verbindung mit der Expression Language verwendet werden. Die in den Ausdrücken über *value[...]* verwendete Methode *getValue()* liefert eine dynamischen Map zurück, die anhand des übergebenen Keys das passende Element ausliest. Mit diesem Trick wird die Übergabe von Methodenparametern simuliert, die ansonsten in der Expression Language nur über Funktionen möglich ist. Damit ist ein sprechender Zugriff auf die Elemente möglich.

Bei der Bestimmung des Ziels eines eingebundenen Links wird zur Generierung des URL der <*cms:link*>-

Tag angegeben. Dieser hat mehrere Aufgaben: Er fügt, wenn benötigt, automatisch den Context-Namen der Webanwendung und den Servlet-Namen hinzu und kann veranlassen, dass einzelne Ressourcen (meist binäre Typen) nicht jedes Mal aus der Datenbank ausgelesen, sondern durch eine Exportfunktion direkt aus dem zugrunde liegenden Filesystem ausgeliefert werden.

Listing 3

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<cms:include property="template" element="head" />

<cms:contentload collector="singleFile" param="%(opencms.uri)" editable="auto">
   <cms:contentaccess var="news" scope="page" />
   <div class="news">
      <h3><c:out value="${news.value['Title']}"/></h3>
      <div class="content">
         <c:out value="${news.value['Text']}" escapeXml="false"/>
      </div>
      <div class="author">
         (<c:out value="${news.value['Author']}"/>)
      </div>
   </div>

   <c:if test="${not empty news.valueList['Links']}">
      <div class="newslinks">
         <h4>Links</h4>
         <ul>
         <c:forEach var="link" items="${news.valueList['Links']}">
            <li>
               <a href="<cms:link>${link}</cms:link>">
                  <cms:property name="Title" file="${link}"/>
               </a>
            </li>
         </c:forEach>
         </ul>
      </div>
   </c:if>
</cms:contentload>

<cms:include property="template" element="foot" />
```

Listing 4

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<cms:include property="template" element="head" />

<ul>
<cms:contentload collector="allInFolderDateReleasedDesc" param=
                                                  "%(opencms.folder)|jmnews|5">
   <c:set var="path"><cms:contentshow element="%(opencms.filename)"/></c:set>
   <c:set var="title"><cms:property name="Title" file="${path}"/></c:set>
   <li><a href="<cms:link>${path}</cms:link>"><c:out value="${title}"/></a></li>
</cms:contentload>
</ul>

<cms:include property="template" element="foot" />
```

Um eine Liste unserer Newseinträge darzustellen, kann die in Listing 4 angegebene JSP verwendet werden.

Wie bei der Detailansicht, dient der Tag *<cms:contentload>* dem Laden der Inhalte. Über die Parameter ist bestimmt, dass aus dem aktuellen Ordner alle Dateien vom Typ *jmnews* eingelesen werden sollen, und zwar maximal fünf Einträge. Im Gegensatz zum vorherigen Beispiel stellt der Tag die Inhalte jetzt nicht nur einmal bereit, sondern iteriert über mehrere Inhalte. Über *<cms:contentshow>* wird mittels eines weiteren Makros der Pfad zur aktuell verarbeiteten Datei ausgegeben. Das *Title*-Property dient wiederum der Ausgabe des Namens des Newsbeitrags.

Direktes Editieren

Neben dem Editieren von Seiteninhalten über den Workplace haben Redakteure auch die Möglichkeit, Inhalte direkt aus der Seitenvorschau zu erzeugen und zu bearbeiten. Auf der Seite werden die editierbaren Bereiche durch Rahmen markiert, unterschiedliche Buttons stehen zum Erzeugen, Bearbeiten und Löschen von Ressourcen bereit. Mithilfe dieser Funktion können angemeldete Redakteure sowie Onlinebenutzer durch die Seite navigieren und Inhalte direkt bearbeiten, ohne im Workplace die Datei finden zu müssen. Direct Edit wird durch den im Template angegebenen Tag *<cms:editable/>* in der JSP aktiviert, der den benötigten JavaScript-Code lädt. Die bearbeitbaren Inhalte werden beim Einbinden als editierbar markiert. Für OpenCms 8 sind in diesem Bereich zahlreiche Erweiterungen geplant. Mit dem Advanced Direct Edit sollen Redakteure alle häufig benötigten Aktionen im Frontend durchführen können, ohne in den Workplace wechseln zu müssen.

Caching

Um das Ausliefern von Inhalten zu beschleunigen, bietet OpenCms Caching-Mechanismen auf mehreren Ebenen an. Neben einem Cache für die Datenbankobjekte ist das hauptsächlich der Flex-Cache, der HTML-Fragmente enthält und diese direkt aus dem Speicher ausliefern kann. Dabei ist man nicht auf komplette Seiten festgelegt; beliebige Elemente einer Seite können mit unterschiedlichen Caching-Regeln konfiguriert werden. Ein häufiges Beispiel ergibt sich für Seiten, die einen Benutzerbereich anbieten: Wenn ein Benutzer angemeldet ist, soll ein Logout-Link im Menü dargestellt werden, für Gastbenutzer ein Login-Link. Die komplette umgebende Seite kann aus dem Speicher ausgeliefert werden, nur der Linkbereich wird dynamisch berechnet.

Suche

Um Inhalte leicht auf der Webseite auffindbar zu machen, kann die Lucene-Integration von OpenCms verwendet werden. Die Erstellung eines Lucene-Index erfolgt konfigurativ. Dabei werden die zu indizierenden Ressourcentypen und die eigentlichen Ressourcen angegeben. Oft wird ein Index für den kompletten Seitenbaum erstellt, es ist jedoch auch möglich, einzelne Teile getrennt zu indizieren oder auszuschließen. Für viele häufig verwende-

Direkt loslegen

Wer die Entwicklung mit OpenCms ausprobieren will, lädt sich am besten die Webanwendung von der OpenCms-Webseite [1] herunter. Nach der Installation steht neben der Dokumentation auch eine Beispielseite zur Verfügung, die man entweder als Grundlage für eine eigene Seite oder zur Inspiration verwenden kann. Um die Grundkonzepte kennen zu lernen, bietet sich ein englischsprachiges Buch zum Thema an, in dem dem Leser anhand von konkreten Beispielen die Entwicklung mit OpenCms nähergebracht wird [6]. Zusätzlich bieten sich die folgenden Ressourcen im Web an:

- das OpenCms-Wiki [3]
- ein Webforum [4]
- die Mailingliste [7]

te Dateitypen wie PDF- oder Word-Dokumente werden bereits passende Mechanismen mitgeliefert, mit denen Textinhalte automatisch extrahiert werden.

Was noch?

Neben den in dieser kurzen Einführung angesprochenen Features bietet OpenCms noch eine Menge weiterer Funktionen. Über die WebDAV können Ressourcen ohne Zugriff auf die Weboberfläche abgelegt werden. Das Benutzermanagement ermöglicht die feingranulare Vergabe von Rechten für Benutzer oder Gruppen auf einzelne Dateien oder Ordner. Für so gut wie jede Anforderung gibt es Erweiterungspunkte im System, beispielsweise können Aktionen registriert werden, die bei jedem Aufruf einer Ressource oder bei Systemevents wie Publikation oder Löschen einer Ressource ausgeführt werden.

Florian Hopf arbeitet als Entwickler bei der Synyx GmbH & Co. KG in Karlsruhe. Neben Content Management gelten seine Interessen momentan der Dokumentenerzeugung auf Basis von Open Document und dem Information Retrieval mit Lucene und Solr.

Links & Literatur

[1] http://opencms.org

[2] http://www.alkacon.com

[3] http://opencms-wiki.org

[4] http://opencms-forum.de

[5] http://opencms-wiki.org/Development_Environment_Setup_Options

[6] Dan Liliedahl: Opencms 7 Development, Packt Publishing, 2008

[7] http://opencms.org/de/development/mailinglist.html