Content Management in der Cloud mit JCR und Oracle

Dominic Weiser
esentri consulting GmbH
Ettlingen

Schlüsselworte
JCR, Oracle, RDBMS, Java, DMS, JSR-170, Tomcat 6, Apache Jackrabbit, 11g XE

Einleitung
Aktuellen Studien zufolge geht auch 2013 der Trend in Unternehmen weiterhin stark in Richtung der Nutzung zentraler Cloud-Anwendungen. Ein Schwerpunkt bei der Nutzung von Cloud-Dienstleistungen liegt auf der Speicherung von Daten. Dabei ist ein elementarer Vorteil, beim Einsatz von Speicherkapazitäten in der Cloud, in der zentralen und ortsunabhängigen Verfügbarkeit aller Daten zu sehen. Der Zugriff kann daher über die Grenzen des Unternehmens hinweg, mit Hilfe eines beliebigen internetfähigen Endgeräts, erfolgen.

Content Repository for Java API
JCR ist ein Standard welcher die Schnittstellen zwischen Content Repository und Anwendung beschreibt. Den Ursprung hatte JCR im Jahr 2005 als Resultat des JSR-170. Bis heute wurde der Standard bereits in der zweiten Version (JSR-283) veröffentlicht. Aktuell wird an Version 2.1 (JSR-333) gearbeitet.

Der JCR-Standard wird in zwei Teile untergliedert, die um einen optionalen Teil erweitert werden. Die beiden Teile unterscheiden wie auf ein DMS zugegriffen werden kann. Der erste Teil des Standards definiert alle lesenden Operationen. Eine Applikation, welche den ersten Teil umsetzt, ist zum Beispiel ein Datei-Browser. Der Anwender kann sich die Dateien anzeigen lassen, sie aber nicht verändern. Der zweite Teil definiert die schreibenden Operationen. Dieser wird zum Beispiel von einem Datei-Explorer umgesetzt worin Dateien angelegt verändert oder gelöscht werden können. Darüber hinaus wird durch den Standard ein optionaler Teil beschrieben, welcher zusätzliche Themenbereiche wie Transaktionen, Sperren oder Observation beschreibt. Die Trennung sorgt dafür, dass je nach der individuell umzusetzenden Anforderung nicht der ganze Standard implementiert werden muss.

Inhaltsstrukturierung
Die Struktur in JCR wird durch unterschiedliche Bereiche abgebildet. Im Folgenden soll dargestellt werden aus welchen Teilen sich die Struktur zusammensetzt.

Repository
Das Repository bildet den Rahmen, in den verschiedene Workspaces eingebettet werden können. Dabei werden innerhalb des Repositories die grundlegenden Eigenschaften wie Persistenz, Suchindizes oder Versionierungsverhalten definiert. Falls diese Einstellungen nicht, durch Workspace spezifische Konfigurationen, überschrieben werden, erbt und verwendet jeder Workspace die Einstellungen des Repositories.

Workspace
Innerhalb eines Repositories können mehrere Workspaces definiert werden, die unabhängig voneinander Daten verwalten. Es ist möglich, Dokumente über die Workspace-Grenze hinweg miteinander zu verknüpfen. Dadurch können gleiche Inhalte in unterschiedlichen Bereichen erstellt werden. Dieses Verhalten ist mit dem Erstellen eines „Branches“ innerhalb von Versionierungswerkzeugen in der Softwareentwicklung zu vergleichen.

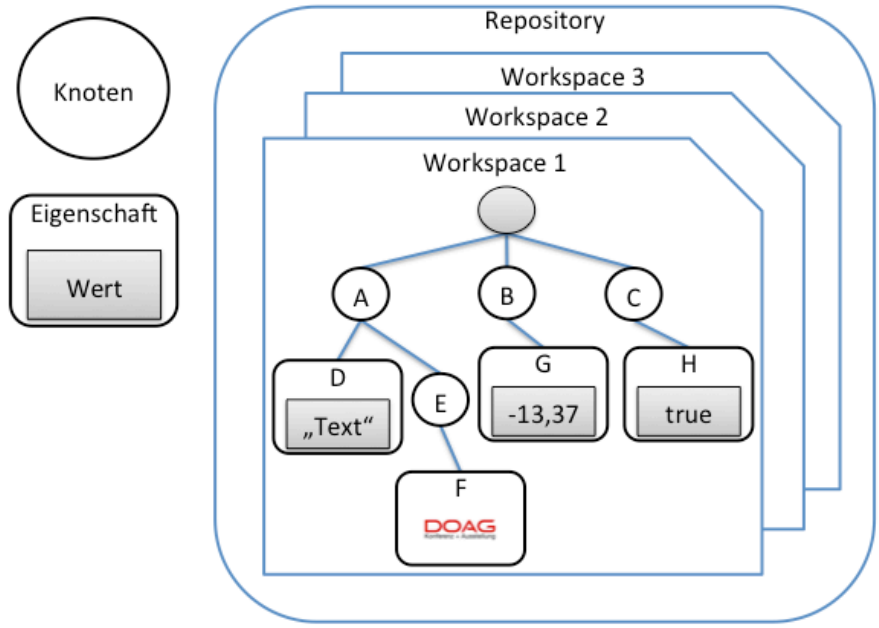


Abb. 1: Inhaltsstruktur JCR

Root-Node
Der Root-Node ist der Ursprung des Content-Trees. Er ist der zentrale Ankerpunkt für alle weiteren Knoten. Darüber hinaus hat er keine weitere Funktionalität. Der Root-Node kann nicht gelöscht werden und ist in jedem Workspace Initial vorhanden.

Knoten
Jedem Knoten kann eine beliebige Anzahl weiterer Knoten und Eigenschaften zugewiesen werden. Dabei wird für jeden Knoten ein Nodetype und ein Name vergeben. Über den Nodetype wird reguliert, welcher Struktur ein Knoten unterliegt. Wird einem Knoten kein Nodetype zugewiesen, ist er unstrukturiert und es können beliebige Knoten und Eigenschaften hinzugefügt werden.

Eigenschaften
Eigenschaften beinhalten die Informationen eines Repositories. Eine Eigenschaft kann ein primitiver Datentyp oder eine Liste von primitiven Datentypen sein. Im Allgemeinen kann man diese als Metadaten von Dateien betrachten. Hier wird der binäre Datenstrom einer Datei abgespeichert.

Persistenzschichten
JCR als Standard beschreibt ausschließlich die API zwischen Content Repository und Applikation. Über die Art wie die Daten festgeschrieben werden, wird keine Vorschrift gemacht. Dennoch haben sich bei den führenden Anbietern von DMS auf JCR Basis zwei unterschiedliche Persistenzschichten durchgesetzt.

1. Das Festschreiben auf einem virtuellen Dateisystem.
   Das virtuelle Dateisystem wird auf die Festplatte des Servers installiert. Ähnlich wie beim Dateisystem auf dem heimischen Rechner werden hier die Dateien im Key – Value verfahren abgespeichert. Durch die von JCR vorgeben Vater – Kind Beziehungen und den „Ordner“-Knoten lässt sich ein beliebiges Dateisystem erstellen.
2. Das Festschreiben auf einer Relationalen Datenbank
   Hier wird die Struktur des Repository in der Datenbank abgespeichert. Dazu werden unterschiedliche Tabellen erstellt, in welchen die „Pfade“ als Wert abgespeichert werden. Werden die Daten ebenfalls auf der Datenbank gespeichert, werden diese in einer separaten Tabelle als Blob geschrieben.

Welcher Ansatz ist der schnellere?
Diese elementare Frage stellt sich bereits zu beginn eines Projekt. Die Antwort der Hersteller von DMS Implementierung fallen hingegen sehr politisch aus. Häufig werden beide Lösungen als gleichwertig dargestellt. Jedoch geht die Tendenz in Richtung des virtuellen Dateisystems. Vereinzelt werden auch Hybride Lösungen empfohlen, so dass die produktiven Daten auf das virtuelle Dateisystem gespeichert und die Backups auf die Datenbank geschrieben werden.

Versuchsaufbau
DMS: Jackrabbit auf einem Tomcat 6
Datenbank: Oracle Datenbank 11g XE
Testrechner: Apple iMac (3,33 GHz Intel Core 2 Duo, 16GB 1067 DDR 3, Mac OS X 10.6.8)

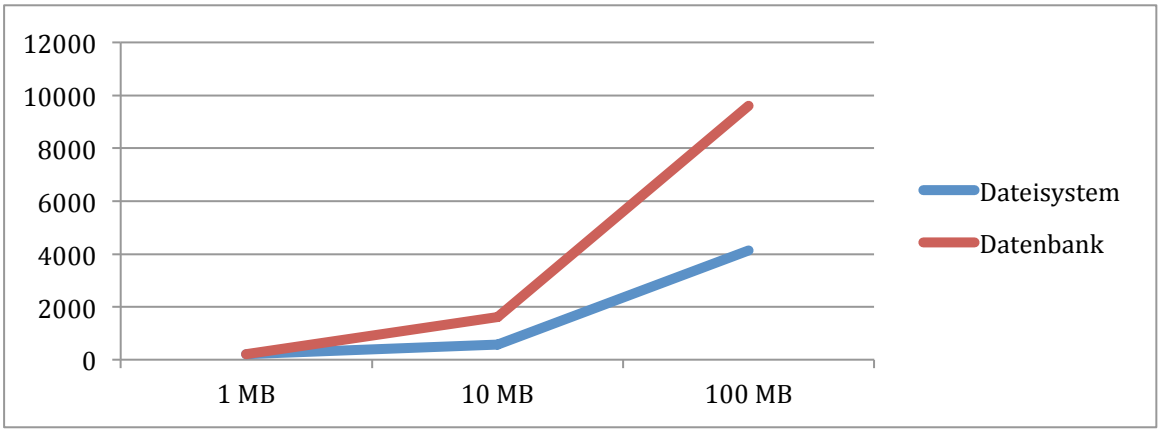
Installation
Im ersten Schritt muss ein DMS installiert werden welches den JCR Standard implementiert. Dazu bietet sich Apache Jackrabbit als Referenzimplementierung des Standards an. Für die Installation auf einem Webserver (z.B. Tomcat 6) kann die vorkompilierte war Datei direkt von der Projektseite geladen werden. Nach der Installation kann durch die von Jackrabbit zur Verfügung gestellten Betriebsmittel ein Standard Repository erstellt werden. Dabei wird als Persistenzschicht ein virtuelles Dateisystem angelegt. Im Anschluss wird die Datenbank als Persistenzschicht angepasst. Dazu wird in der reporsitory.xml die Einstellungen für das Dateisystem und den Persistencemanager bearbeitet.

```
<FileSystem class="org.apache.jackrabbit.core.fs.db.OracleFileSystem">
  <param name="driver" value="oracle.jdbc.driver.OracleDriver"/>
  <param name="url" value="jdbc:oracle:thin:@localhost:1521:XE"/>
  <param name="user" value="jackrabbit" />
  <param name="password" value="test"/>
</FileSystem>

<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
  <param name="driver" value="oracle.jdbc.driver.OracleDriver"/>
  <param name="url" value="jdbc:oracle:thin:@localhost:1521:XE"/>
  <param name="user" value="jackrabbit" />
  <param name="password" value="test"/>
</DataStore>

<PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.
                          OraclePersistenceManager">
  <param name="url" value="jdbc:oracle:thin:@localhost:1521:XE"/>
  <param name="user" value="jackrabbit" />
  <param name="password" value="test"/>
  <param name="tablespace" value=""/>
  <param name="externalBLOBs" value="false"/>
</PersistenceManager>
```

Diese Einstellungen können für die Workspaces und die Versionierung unterschiedlich sein. Das wird immer dann verwendet wenn die Daten und Versionierung auf unterschiedlichen Persistenzschichten abgelegt werden sollen. Am häufigsten trifft man hier die Konfiguration Daten auf den Dateisystem Versionierung auf der Datenbank an. Dabei handelt es sich um die Konfiguration zu Testzwecken. Im produktiven System sollte das Passwort als base64 Verschlüsselt werden.

```
<param name="password" vlaue="{base64}dGVzdA=="/>
```

Vergleich



Wie in dem Schaubild zu sehen, werden nehmen die Zugriffszeiten mit der Größe der Datei zu. Bei kleinen Dateien (bis 1MB) ist kein Unterschied zwischen Datenbank und virtuellem Dateisystem zu erkenne. Jedoch stellt sich bereits ab einer Dateigröße von ca. 10MB eine Verdopplung der Zugriffszeiten ein.

Fazit
Die durch die Messung ergebenen Zeiten decken sich mit den Erwartungen und den politischen Empfehlungen der DMS Herstellern. Daraus ergibt sich, dass die Datenbank als Persistenzschicht für kleine Dateien sehr gleich gut zu verwenden ist wie das virtuelle Dateisystem. Jedoch wenn abzusehen ist, dass vornehmlich größere Daten gespeichert werden, sollte das virtuelle Dateisystem die erste Wahl bleiben. Jedoch bleibt immer noch fraglich welcher Mehrwert durch die Datenbank geschaffen wird. Durch die verwendeten Volltextsuchen und die damit verbunden Indizes ermöglichen einen sehr schnellen Zugriff auf die Daten. Das relationale Zugreifen fällt dabei zunehmend in Hintertreffen und wird nur noch durch die erste Version des Standard unterstützt.

Kontaktadresse:
Dominic Weiser
esentri consulting GmbH
Pforzheimer Straße 132
76275 Ettlingen

Telefon: +49 (0) 7243-354 90 0
E-Mail dominic.weiser@esentri.com
Internet: http://www.esentri.com