Database-Only Repositories with JNDI Datasources

Your Rating: Results: 141 rates

Huh, what's this about?

This page gives a quick example for how to set up magnolia with Jackrabbit JCR writing to Database only (as far as possible).

The default setup for magnolia uses the built-in DerbyDB, as well as the filesystem for storing the JCR Content. The sample configurations provided for mysql also only puts part of the repository in the database. The remaining parts land on the file system, by default within the webapp folder of magnolia.

There are a number of disadvantages to this "mixed" setup:

1. Mixing file-system and DB provides for 2 points of failure.
2. Mixing file-system and DB makes consistent backups more difficult. Basically, to guarantee a consistent backup, magnolia has to be shut down.
3. Filesystem-backed storage means Repository is not clusterable in Jackrabbit.

Switching to a database-only setup gets rid of these disdvantages.

These instructions have been tested with Magnolia 4.3 and 4.4.

Structure of JCR Repositories

Basically, a JCR repository can have one or more workspaces. Each workspace is what is called a "repository" in magnolia: website, dms, data, imageing, config, etc...

Each workspace requires a number of different "Storage-Backends" in JCR in order to store all the different data-elements. These are:

- A "FileSystem" for content
- A "PersistanceManager" for content
- A "Datastore" for large content (blobs)
- A "FileSystem" for versions
- A "PersistanceManager" for versions

Also required is:

- A general "FileSystem" for the repository (all workspaces)

Confusingly, even though the element is called "FileSystem", both "FileSystem" and "PersistanceManager" can be configured to use a number of different backends, either file-system based, database based, or others.

Our objective is to configure everything to use database-backed storage.

So will everything be in the DB?

Unfortunately, the answer is "no". Even after configuring all storage to use database-backends, Jackrabbit will still write the following into the repositories folder:

- a config file per workspace. If this file is missing, the workspace will be reinitialized, so make sure you don't delete these!!
- the search-indexes per workspace. These can be deleted any time, and will be recreated as needed.

For more information, see the Jackrabbit Documentation and Wiki.

How to set it up?

Note: Your old repository will be gone, so make a backup fo your content first!!

1. Create a database and appropriate users on the database server of your choice. You will need a seperate database for each magnolia instance, eg. one for author and one for public.
2. Install appropriate JDBC drivers for your database in either WEB-INF/lib or TOMCAT_HOME/lib
3. Create JNDI Datasource definitions in the web.xml, context.xml or server.xml files, see JNDI Datasources in the Apache Tomcat documentation for more details.
4. Configure Jackrabbit like in the example file below. Your Jackrabbit Config file goes in the folder *TOMCAT_HOME/webapps/magnoliaAuthor /WEB-INF/config/repo-conf*
5. Configure magnolia to use the new jackrabbit configuration. Edit *TOMCAT_HOME/webapps/magnoliaAuthor/WEB-INF/config/default/magnolia. properties*. Set the property "magnolia.repositories.jackrabbit.config",
   eg: magnolia.repositories.jackrabbit.config=WEB-INF/config/repo-conf/jackrabbit-mysql.xml
6. Configure the repository home dir (where Jackrabbit will still write a few config files and the indices) to lie outside Tomcat's webapps folder.
   eg: magnolia.repositories.home=c:/dev/magnolia/repo-author/repositories
7. Repeat steps 4-6 for the public instance.

See the following example JackRabbit config file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Repository PUBLIC "-//The Apache Software Foundation//DTD Jackrabbit 1.5//EN" "http://jackrabbit.
apache.org/dtd/repository-1.5.dtd">
<Repository>
  <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
          <param name="driver" value="javax.naming.InitialContext"/>
          <param name="url" value="java:comp/env/jdbc/magnoliaAuthorDS"/>
          <param name="schema" value="mysql"/>
      <param name="schemaObjectPrefix" value="fsrep_"/>
  </FileSystem>
  <Security appName="Jackrabbit">
    <AccessManager class="org.apache.jackrabbit.core.security.SimpleAccessManager"></AccessManager>
    <LoginModule class="org.apache.jackrabbit.core.security.SimpleLoginModule">
      <param name="anonymousId" value="anonymous" />
    </LoginModule>
  </Security>
  <DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
          <param name="driver" value="javax.naming.InitialContext"/>
          <param name="url" value="java:comp/env/jdbc/magnoliaAuthorDS"/>
          <param name="databaseType" value="mysql"/>
      <param name="schemaObjectPrefix" value="ds_" />
  </DataStore>
  <Workspaces rootPath="${rep.home}/workspaces" defaultWorkspace="default" />
  <Workspace name="default">
        <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
                  <param name="driver" value="javax.naming.InitialContext"/>
                  <param name="url" value="java:comp/env/jdbc/magnoliaAuthorDS"/>
                  <param name="schema" value="mysql"/>
          <param name="schemaObjectPrefix" value="fsws_${wsp.name}_"/>
        </FileSystem>
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.MySqlPersistenceManager">
          <param name="driver" value="javax.naming.InitialContext"/>
          <param name="url" value="java:comp/env/jdbc/magnoliaAuthorDS"/>
      <param name="schema" value="mysql" /><!-- warning, this is not the schema name, it's the db type -->
      <param name="schemaObjectPrefix" value="pm_${wsp.name}_" />
```

```
      <param name="externalBLOBs" value="false" />
    </PersistenceManager>
    <SearchIndex class="org.apache.jackrabbit.core.query.lucene.SearchIndex">
      <param name="path" value="${wsp.home}/index" />
      <param name="useCompoundFile" value="true" />
      <param name="minMergeDocs" value="100" />
      <param name="volatileIdleTime" value="3" />
      <param name="maxMergeDocs" value="100000" />
      <param name="mergeFactor" value="10" />
      <param name="maxFieldLength" value="10000" />
      <param name="bufferSize" value="10" />
      <param name="cacheSize" value="1000" />
      <param name="forceConsistencyCheck" value="false" />
      <param name="autoRepair" value="true" />
      <param name="analyzer" value="org.apache.lucene.analysis.standard.StandardAnalyzer" />
      <param name="queryClass" value="org.apache.jackrabbit.core.query.QueryImpl" />
      <param name="respectDocumentOrder" value="true" />
      <param name="resultFetchSize" value="2147483647" />
      <param name="extractorPoolSize" value="3" />
      <param name="extractorTimeout" value="100" />
      <param name="extractorBackLogSize" value="100" />
      <param name="textFilterClasses"
        value="org.apache.jackrabbit.extractor.MsWordTextExtractor,
               org.apache.jackrabbit.extractor.MsExcelTextExtractor,
               org.apache.jackrabbit.extractor.MsPowerPointTextExtractor,
               org.apache.jackrabbit.extractor.PdfTextExtractor,
               org.apache.jackrabbit.extractor.OpenOfficeTextExtractor,
               org.apache.jackrabbit.extractor.RTFTextExtractor,
               org.apache.jackrabbit.extractor.HTMLTextExtractor,
               org.apache.jackrabbit.extractor.PlainTextExtractor,
               org.apache.jackrabbit.extractor.XMLTextExtractor" />
    </SearchIndex>
  </Workspace>
  <Versioning rootPath="${rep.home}/version">
        <FileSystem class="org.apache.jackrabbit.core.fs.db.DbFileSystem">
                  <param name="driver" value="javax.naming.InitialContext"/>
                  <param name="url" value="java:comp/env/jdbc/magnoliaAuthorDS"/>
                  <param name="schema" value="mysql"/>
          <param name="schemaObjectPrefix" value="fsver_"/>
        </FileSystem>
    <PersistenceManager class="org.apache.jackrabbit.core.persistence.bundle.MySqlPersistenceManager">
          <param name="driver" value="javax.naming.InitialContext"/>
          <param name="url" value="java:comp/env/jdbc/magnoliaAuthorDS"/>
      <param name="schema" value="mysql" /><!-- warning, this is not the schema name, it's the db type -->
      <param name="schemaObjectPrefix" value="version_" />
      <param name="externalBLOBs" value="false" />
    </PersistenceManager>
  </Versioning>
</Repository>
```

Things to note:

My JNDI datasource in this example is called "magnoliaAuthorDS". This is for the author instance. For the public instance, replace all occurrences of "magnoliaAuthorDS" with the JNDI name of your public instance datasource.

Notes on MySQL

When using mysql for JackRabbit, you will probably need to configure the number of connections allowed by the database server to about 200. JackRabbit opens a lot of connections.

If you are using MySQL as the Database, and want to move the Datastore (Blob storage) into the DB as well (as in the setup above) then you will need to configure mysql to handle larger binary objects via JDBC. There are instructions for doing this on the Jackrabbit wiki as well as in the mysql documentation, but for my version of mysql (5.1) it was enough to add the following to my.cnf:

```
max_connections = 200
max_allowed_packet = 32M
```

The limit specified here (in my example 32M for 32 megabytes) will be a hard limit on the maximum file size you will be able to upload to the repository.

MySQL datasource definition

You can define the datasource in your servlet container in the normal way. A standard **javax.sql.Datasource** definition, as described in the tomcat documentation will work. However, JackRabbit does not need the connection pooling offered by the standard datasource, and there is no easy way to disable the connection pool.

Some configurations could even be harmful, as JackRabbit keeps connections open for a very long time without using them, so if you have configured recovery of abandoned connections (recoverAbandoned=true), the connection pool may "steal back" connections JackRabbit is still using, mistakenly believing them to be abandoned.

To avoid this kind of thing from the outset you might consider configuring an unpooled datasource, for example as follows:

```
<Resource name="jdbc/magnoliaAuthorDS" auth="Container"
    type="com.mysql.jdbc.jdbc2.optional.MysqlDataSource"
    factory="com.mysql.jdbc.jdbc2.optional.MysqlDataSourceFactory"
    user="***" password="***" driverClassName="com.mysql.jdbc.Driver"
    explicitUrl="true" url="jdbc:mysql://localhost:3306/magnolia_author"
/>
```

You need to specify the `type` so that Tomcat does not try to instantiate its own pooled DS. ⚠️ Two other important differences:

- the `user` property - in Tomcat's regular DS, this is called `username`.
- `explicitUrl` needs to be set to true unless you configure **all** parameters explicitly outside the url (including database name, which we don't do in this example).

Connection idle timeouts

In addition to the abandoned connection recovery at the tomcat end of things, there are also timeouts for idle connections configured at the mysql side.

This should not be a problem on a production server, where requests can be expected to come in at a somewhat constant rate. On development setups, where there may be no use of the system at all for a whole weekend, the connection-idle-timeout needs to be increased.

For mysql, add something like the following to your my.ini, and restart the server:

```
wait_timeout = 302400
interactive_timeout = 302400
```