Jackrabbit Repository Configuration File

Your Rating: ☆☆☆☆☆ Results: ★★★★½ 28 rates

Magnolia uses Apache Jackrabbit content repository. You are free to use other content repository implementations as long as they conform to JSR 283. In this page we attempt to demystify the repository configuration file Magnolia uses to configure Jackrabbit.

The configuration options available are outlined in the API here `RepositoryConfig`.

```
<!DOCTYPE Repository PUBLIC "
-//The Apache Software Foundation//DTD Jackrabbit 2.0//EN"
"http://jackrabbit.apache.org/dtd/repository-2.0.dtd">
<Repository>
  <DataSources .../>
  <FileSystem .../>
  <Security .../>
  <DataStore .../>
  <Workspaces .../>
  <Workspace> <!-- Blueprint for all new workspaces -->
    <FileSystem .../>
    <PersistenceManager .../>
    <SearchIndex .../>
    <WorkspaceSecurity .../>
  </Workspace>
  <Versioning .../>
</Repository>
```

See http://jackrabbit.apache.org/jcr/jackrabbit-configuration.html

Magnolia	Jackrabbit
6.2	2.20
6.1	2.18
5.7	2.16
5.6	2.12
5.5	2.12
5.4	2.8

Other Resources

- Jackrabbit Index Debugging
- Jackrabbit Repository Splitting
- Search Index Configuration File
- Jackrabbit Repository Configuration File
- Jackrabbit Workspace Configuration File

Summary

Magnolia ships with a few examples of popular repository configurations that customers might use as-is or the basis for a customized configuration. If we look at what is provided with Magnolia 5 and 6 bundles we find 5 files in both author and public.

The repo config folders are located here:

- `/magnolia-x.x.x/apache-tomcat-x.x.x/webapps/magnoliaAuthor/WEB-INF/config/repo-conf`
- `/magnolia-x.x.x/apache-tomcat-x.x.x/webapps/magnoliaPublic/WEB-INF/config/repo-conf`

OOTB Examples:

- `jackrabbit-bundle-derby-search.xml`
- `jackrabbit-bundle-h2-search.xml` (Magnolia 5.5+)
- `jackrabbit-bundle-ingres-search.xml`
- `jackrabbit-bundle-mysql-search.xml`
- `jackrabbit-bundle-postgres-search.xml`
- `jackrabbit-memory-search.xml`

Other examples:

- `jackrabbit-pool-mssql-search.xml`
- `jackrabbit-pool-h2-search.xml`
- `jackrabbit-pool-oracle-search-db.xml` (db FS, db DS)
- `jackrabbit-pool-mysql8-search.xml` (MySQL 8)

Two properties are required for the content repository (see Configuration management).

- Repository home directory: specified in the `magnolia.properties` file as `magnolia.repositories.home`
- Repository configuration file: specified in the `magnolia.properties` file as `magnolia.repositories.jackrabbit.config`

The repository configuration file specifies global options like security, data sources, and versioning. A default workspace configuration template is also included in the repository configuration file. For each workspace that was created, there will also be a `workspace.xml` file created inside the workspace home directory that will be used for the workspace.

Configuration

DataSources

The data source(s) used by jackrabbit can be configured as child elements of the `Repository` using the `DataSources` element. Typically, you configure one `DataSource` per repository and each workspace will use the exact same data source connection. Technically, a workspace could use a different connection. Some workspaces could connected to an embedded datasource while others connect to a database. There is a lot of flexibility.

Database

The configuration options available are outlined in the API here `DataSourceConfig`.

```
<DataSources>
  <DataSource name="magnolia">
    <param name="driver" value="com.mysql.jdbc.Driver" />
    <param name="url" value="jdbc:mysql://localhost:3306/magnolia" />
    <param name="user" value="root" />
    <param name="password" value="password" />
    <param name="databaseType" value="mysql"/>
    <param name="validationQuery" value="select 1"/>
  </DataSource>
</DataSources>
```

`driver`	**required** Depending which database you choose to work with make sure to include the jar with the appropriate driver in the classpath. • MySQL Connector/J is the official JDBC driver for MySQL • Ingres JDBC Driver Downloads - Ingres Community Wiki • PostgreSQL JDBC Driver • Oracle JDBC Drivers • Microsoft JDBC Drivers
`url`	**required** The connection url for the database.
`password`	**required** The password associated with the schema.

`databaseType`	**optional**, *depends on database type* Use the appropriate setting for the database. • `postgresql` • `mysql` • `mssql` • `oracle`
`validationQuery`	**optional**, *depends on database type* The SQL query that will be used to validate connections from this pool before returning them to the caller. The query depends on the database type. • MySQL: `select 1` • MSSQL: `select 1` • Oracle: `select 1 from dual`
`maxPoolSize`	**optional** Restrict the number of connections in the pool to a max value.

JNDI

Jackrabbit supports JNDI data sources. The container you use will determine how you setup your JNDI data source.

> Note that you can use data sources from JNDI in both the configuration of the data sources and in configuration of the components. Jackrabbit will use these JNDI data sources as-is and will not wrap pools around them. See ConnectionPooling.

- Tomcat: JNDI Resources HOW-TO Tomcat 9.0.
- JBoss: JBoss AS JNDI Datasource Setup.
- WebLogic: JNDI for Oracle WebLogic Server.
- WebSphere: JNDI namespace bindings for WebSphere

See: https://jackrabbit.apache.org/archive/wiki/JCR/UsingJNDIDataSource.html

Embedded

Jackrabbit provides persistence manager implementations for both the H2 and Derby databases. Using these databases does not require a concrete datasource configuration. You simply provide the connection URL at the persistence manager configuration. You do need to make sure that you have the jar file in your classpath.

- H2 Database Engine
- Apache Derby

FileSystem

The virtual file system used by the repository to store things like registered namespaces and node types.

```
<FileSystem class="org.apache.jackrabbit.core.fs.local.LocalFileSystem">
    <param name="path" value="${rep.home}/repository" />
</FileSystem>
```

Jackrabbit provides a lot of choices for how you can configure the `FileSystem`. Choose the class that best fits your use case and click the link to see your configuration options.

- org.apache.jackrabbit.core.fs.local.LocalFileSystem: The file system will be created in the location specified by `magnolia.repositories.home`.
- org.apache.jackrabbit.core.fs.db.DbFileSystem: Persists file system entries in a database table.
- org.apache.jackrabbit.core.fs.db.DerbyFileSystem: Persists file system entries in an embedded Derby database.
- org.apache.jackrabbit.core.fs.db.MSSqlFileSystem: Persists file system entries in an MS SQL database.
- org.apache.jackrabbit.core.fs.db.OracleFileSystem: Persists file system entries in an Oracle database.
- org.apache.jackrabbit.core.fs.mem.MemoryFileSystem: An in-memory file system implementation (can be useful for development).

See: http://jackrabbit.apache.org/jcr/jackrabbit-configuration.html#file-system-configuration

Security

The security configuration element is used to specify authentication and authorization settings for the repository.

```
<Security appName="magnolia"> <SecurityManager class="org.apache.jackrabbit.core.DefaultSecurityManager"/>
    <AccessManager class="org.apache.jackrabbit.core.security.DefaultAccessManager"></AccessManager>
    <!-- Login module defined here is used by the repo to authenticate every request.
         Not by the webapp to authenticate user against the webapp context
         (this one has to be passed before thing here gets invoked).
    -->
    <LoginModule class="info.magnolia.jaas.sp.jcr.JackrabbitAuthenticationModule"></LoginModule>
</Security>
```

Jackrabbit uses the Java Authentication and Authorization Service (JAAS) to authenticate users who try to access the repository. The `appName` parameter in the `<Security/>` element is used as the JAAS application name of the repository.

Once a user has been authenticated, Jackrabbit will use the configured `AccessManager` to control what parts of the repository content the user is allowed to access and modify.

The slightly more advanced `SimpleJBossAccessManager` class is designed for use with the JBoss Application Server, where it maps JBoss roles to Jackrabbit permissions.

See: http://jackrabbit.apache.org/jcr/jackrabbit-configuration.html#security-configuration

DataStore

The data store is optionally used to store large binary values. Normally all node and property data is stored in a persistence manager, but for large binaries such as files special treatment can improve performance and reduce disk usage.

The main features of the data store are:

- **Space saving**: only one copy per unique object it kept
- **Fast copy**: only the identifier is copied
- Storing and reading does not block others
- Multiple repositories can use the same data store
- Objects in the data store are immutable
- Garbage collection is used to purge unused objects
- Hot backup is supported

File System DataStore

The file data store stores each binary in a file. The file name is the hash code of the content. When reading, the data is streamed directly from the file (no local or temporary copy of the file is created). The file data store does not use any local cache, that means content is directly read from the files as needed. New content is first stored in a temporary file, and later renamed / moved to the right place.

from jackrabbit-bundle-mysql-search.xml

```
<DataStore class="org.apache.jackrabbit.core.data.FileDataStore">
    <param name="path" value="${rep.home}/repository/datastore"/>
    <param name="minRecordLength" value="1024"/> <!-- default is 100 bytes -->
</DataStore>
```

The configuration options available are outlined in the API here `FileDataStore` .

See: https://wiki.apache.org/jackrabbit/DataStore#File_Data_Store

Database DataStore

The database data store stores data in a relational database. All content is stored in one table, the unique key of the table is the hash code of the content. When reading, the data may be first copied to a temporary file on the server, or streamed directly from the database (depending on the `copyWhenReading` setting). New content is first stored in the table under a unique temporary identifier, and later the key is updated to the hash of the content.

MySQL does not support sending very large binaries from the JDBC driver to the database. Therefore a database `DataStore` should be avoided when using MySQL.

```
<DataStore class="org.apache.jackrabbit.core.data.db.DbDataStore">
    <param name="url" value="java:jboss/datasources/jackrabbit"/> <!-- JNDI Datasource example -->
    <param name="driver" value="javax.naming.InitialContext"/>
    <param name="databaseType" value="oracle"/>
    <param name="schemaObjectPrefix" value="repo_ds" />
</DataStore>
```

The configuration options available are outlined in the API here `DbDataStore` .

See: https://wiki.apache.org/jackrabbit/DataStore#Database_Data_Store

Workspaces

The `Workspaces` element of the repository configuration specifies where and how the workspaces are managed. The configuration of this element gets stored in the class `RepositoryConfig`.

```
<Workspaces rootPath="${rep.home}/workspaces" defaultWorkspace="default" />
```

The `Workspaces` element has the following configuration options, set as attributes of the element and not as `param` sub-elements.

`rootPath`	The native file system directory for workspaces. A subdirectory is automatically created for each workspace, and the path of that subdirectory can be used in the workspace configuration as the `${wsp.path}` variable.
`defaultWorkspace`	Name of the default workspace. This workspace is automatically created when the repository is first started.
`configRootPath`	By default the configuration of each workspace is stored in a `workspace.xml` file within the workspace directory within the `rootPath` directory. If this option is specified, then the workspace configuration files are stored within the specified path in the virtual file system (see above) configured for the repository.
`maxIdleTime`	By default Jackrabbit only releases resources associated with an opened workspace when the entire repository is closed. This option, if specified, sets the maximum number of seconds that a workspace can remain unused before the workspace is automatically closed.

See: http://jackrabbit.apache.org/jcr/jackrabbit-configuration.html#workspace-configuration

Workspace

The configuration specified in the `Workspace` element becomes the template for all workspaces created by Jackrabbit. Each workspace will have its own `workspace.xml` file generated from this template. See Jackrabbit Workspace Configuration File for more information.

FileSystem

Workspace level virtual file system passed to the persistence manager and search index. The same configuration options are available here as described above for the repository level virtual file system.

See: http://jackrabbit.apache.org/jcr/jackrabbit-configuration.html#file-system-configuration

PersistenceManager

The PM is an internal Jackrabbit component that handles the persistent storage of content nodes and properties. Property values are also stored in the persistence manager, with the exception of large binary values (those are usually kept in the DataStore). Each workspace of a Jackrabbit content repository uses a separate persistence manager to store the content in that workspace.

```
<PersistenceManager class="org.apache.jackrabbit.core.persistence.pool.MySqlPersistenceManager">
    <param name="dataSourceName" value="magnolia"/> <param name="schemaObjectPrefix" value="pm_${wsp.name}_"
/>
</PersistenceManager>
```

Jackrabbit provides a lot of choices for how you can configure the `PersistenceManager`. Choose the class that best fits your use case and click the link to see your configuration options.

> All `BundlePersistenceManager` implementations that do not use a pool of JDBC connections have been marked as deprecated. Replace them with the pooled version.

- org.apache.jackrabbit.core.persistence.pool.BundleDbPersistenceManager

- org.apache.jackrabbit.core.persistence.pool.DerbyPersistenceManager
- org.apache.jackrabbit.core.persistence.pool.H2PersistenceManager
- org.apache.jackrabbit.core.persistence.pool.MSSqlPersistenceManager
- org.apache.jackrabbit.core.persistence.pool.MySqlPersistenceManager
- org.apache.jackrabbit.core.persistence.pool.OraclePersistenceManager
- org.apache.jackrabbit.core.persistence.pool.PostgreSQLPersistenceManager
- org.apache.jackrabbit.core.persistence.mem.InMemPersistenceManager (can be useful for development)

SearchIndex

The search index in Jackrabbit is pluggable and has a default implementation based on Apache Lucene. It is configured in the file workspace.xml once the workspace is created. For more detailed information on the settings here see *Jackrabbit Workspace Configuration File* and *Search Index Configuration File*.

```
<SearchIndex class="org.apache.jackrabbit.core.query.lucene.SearchIndex">
    <param name="path" value="${wsp.home}/index" /> <!-- SearchIndex will get the indexing configuration
from the classpath, if not found in the workspace home -->
    <param name="indexingConfiguration" value="/info/magnolia/jackrabbit/indexing_configuration.xml"/>
    <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="queryClass" value="org.apache.jackrabbit.core.query.QueryImpl" />
    <param name="respectDocumentOrder" value="true" />
    <param name="resultFetchSize" value="100" />
    <param name="extractorPoolSize" value="3" />
    <param name="extractorTimeout" value="100" />
    <param name="extractorBackLogSize" value="100" /> <!-- needed to highlight the searched term -->
    <param name="supportHighlighting" value="true"/> <!-- custom provider for getting an HTML excerpt in a
query result with rep:excerpt() -->
    <param name="excerptProviderClass" value="info.magnolia.jackrabbit.lucene.SearchHTMLExcerpt"/>
</SearchIndex>
```

See: http://wiki.apache.org/jackrabbit/Search

WorkspaceSecurity

Workspace security is handled by the class `MagnoliaAccessProvider`. It is a Magnolia specific ACL provider. This class will compile the set of permissions a user has for a given workspace. If the user does not have any permissions for the workspace then root-only access is returned. If the user is detected as admin or superuser, which is checked first, then an implementation of `CompiledPermissions` that grants everything is returned.

The class `MagnoliaAccessProvider` has DEBUG output available that can be switched on using the Log Tools app.

```
<WorkspaceSecurity>
    <AccessControlProvider class="info.magnolia.cms.core.MagnoliaAccessProvider" />
</WorkspaceSecurity>
```

Versioning

FileSystem

Versioning level virtual file system passed to the persistence manager. The same configuration options are available here as described above for the repository level virtual file system.

See: http://jackrabbit.apache.org/jcr/jackrabbit-configuration.html#file-system-configuration

PersistenceManager

Persistence configuration for the version store. The versioning configuration is much like workspace configuration as they are both used by Jackrabbit for storing content. The same configuration options are available here as described above for the workspace level persistence manager.