Search Index Configuration File

Summary

Jackrabbit allows you to control which properties of a node are indexed and how much they will affect the `jcr:score` value of that node in the result. You also have the option to configure different analyzers on a property-by-property basis. The index configuration file instructs lucene how to index the content of a workspace. This page is based on Jackrabbit 2.18 API and Lucene 3.6.

Apache Lucene is a Java-based indexing and search technology, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities. Jackrabbit 2.18 ships with Lucene 3.6.

Feel free to ask questions at the bottom of the page for further clarification.

Important Changes

- 5.5.9 Release notes - Separate indexing configuration files.
- 5.6.3 Release notes - Separate indexing configuration files.
- 5.6.5 Release notes - Fixed full text search of documents.

Indexing Configuration

The configuration parameter `indexingConfiguration` is not set by default. This means all properties of a node are indexed.

If you wish to configure the indexing behavior you need to add a parameter to the `SearchIndex` element of either your repository configuration file or your workspace configuration file.

Any time you make changes to the indexing configuration do not forget to recreate the index from scratch.

See https://wiki.apache.org/jackrabbit/IndexingConfiguration

Configuration files

There is a default indexing configuration file used by all workspaces in the repository. It is located in the core module in the package `info.magnolia.jackrabbit`.

```
<param name="indexingConfiguration" value="/info/magnolia/jackrabbit/indexing_configuration_default.xml"/>
```

As of Magnolia 6.1 that file looks like this:

indexing_configuration_default.xml

```
<?xml version="1.0"?>
<!DOCTYPE configuration SYSTEM "http://jackrabbit.apache.org/dtd/indexing-configuration-1.2.dtd">
<configuration xmlns:nt="http://www.jcp.org/jcr/nt/1.0" xmlns:mgnl="http://www.magnolia.info/jcr/mgnl" xmlns:
jcr="http://www.jcp.org/jcr/1.0">
  <!--
      A global, generic indexing configuration used for all workspaces in Magnolia.
      It excludes some well known properties from the node scope fulltext index.
 -->
<index-rule nodeType="nt:base">
    <!-- unlike other system properties, we want mgnl:tags to be indexed -->
    <property>mgnl:tags</property>
    <property isRegexp="true" nodeScopeIndex="false">mgnl:.*</property>
    <property isRegexp="true" nodeScopeIndex="false">jcr:.*</property>
    <!-- notice how the catch-all property should come at the end after all specific ones -->
    <property isRegexp="true">.*:.*</property>
</index-rule>

</configuration>
```

There is also a website workspace specific file in the same package.

That file looks like this:

indexing_configuration_website.xml

```
<?xml version="1.0"?>
<!DOCTYPE configuration SYSTEM "http://jackrabbit.apache.org/dtd/indexing-configuration-1.2.dtd">
<configuration xmlns:nt="http://www.jcp.org/jcr/nt/1.0" xmlns:mgnl="http://www.magnolia.info/jcr/mgnl" xmlns:
jcr="http://www.jcp.org/jcr/1.0">
  <!--
      A global, generic indexing configuration used for all workspaces in Magnolia.
      It excludes some well known properties from the node scope
      fulltext index and boosts mgnl:page title.
 -->
<index-rule nodeType="nt:base">
    <!-- unlike other system properties, we want mgnl:tags to be indexed -->
    <property>mgnl:tags</property>
    <property isRegexp="true" nodeScopeIndex="false">mgnl:.*</property>
    <property isRegexp="true" nodeScopeIndex="false">jcr:.*</property>
    <!-- notice how the catch-all property should come at the end after all specific ones -->
    <property isRegexp="true">.*:.*</property>
</index-rule>
<index-rule nodeType="mgnl:page">
    <property boost="3.0">title</property>
</index-rule>
<!--
     index aggregate configuration for mgnl:page. Properties of mgnl:area and mgnl:component should also be
part of the full-text index for mgnl:page
     as they all constitute the contents of a page. This should simplify searching for content within pages by
simply issuing a JCR query
     e.g. "select * from [mgnl:page] as p where contains(p.*, 'foo')"
-->
<aggregate primaryType="mgnl:page">
    <include primaryType="mgnl:area">*</include>
    <include primaryType="mgnl:component">*</include>
</aggregate>
<!-- areas can be nested. See http://wiki.apache.org/jackrabbit/IndexingConfiguration for recursion -->
<aggregate primaryType="mgnl:area" recursive="true">
    <include primaryType="mgnl:component">*</include>
    <include primaryType="mgnl:area">*</include>
</aggregate>
<!-- components can contain areas with components -->
<aggregate primaryType="mgnl:component" recursive="true">
    <include primaryType="mgnl:area">*</include>
    <include primaryType="mgnl:component">*</include>
</aggregate>
</configuration>
```

Index Rules

To optimize the index size you can index only certain properties of a node type. Index rules are processed top down and the first matching rule gets applied and all remaining ones are ignored.

As of Jackrabbit 2.0 you can also use the match all regexp for the namespace prefix part of a property name. However that's currently the only supported regular expression.

Please note that you have to declare the namespace prefixes in the `configuration` element that you are using throughout the XML file.

Node Scope

With the `nodeScopeIndex` attribute set to `false` the property will not be in the full-text index. Meaning it would be available for all searches except for those using `contains(...)` in `sql` and `sql2` or `jcr:contains(...)` for `xpath`.

Here we are applying an index rule against nodes of type `nt:base`. This also applies to nodes with a type that extends from `nt:base`. Since `nt:base` is the base node type of all primary nodes types this rule will apply everywhere.

```
<index-rule nodeType="nt:base">
  <property isRegexp="true" nodeScopeIndex="false">mgnl:.*</property> <!-- Exclude Magnolia metadata from the
full-text index. -->
  <property isRegexp="true" nodeScopeIndex="false">jcr:.*</property> <!-- Exclude JCR metadata from the full-
text index. -->
  <property isRegexp="true">.*:.*</property> <!-- Include all properties from any namespace, even the empty
namespace. -->
</index-rule>
```

Boost

It is possible to configure `boost` value on both nodes and/or properties that match an index rule. The default `boost` value is `1.0`. Higher `boost` values (a reasonable range is `1.0 - 5.0`) will yield a higher score value and appear as more relevant.

Here we are applying a `boost` value of `3.0` added to the `title` property on nodes of type `mgnl:page`.

```
<index-rule nodeType="mgnl:page">
  <property boost="3.0">title</property>
</index-rule>
```

Conditions

You may also add a condition to the index rule and have multiple rules with the same node type.

For example, let's say that we only want to boost page titles when the paged has been marked with a `priority` property. Further more let's assume we also have a requirement to provide three priority levels of low, medium, and high.

```
<!-- Since the default boost it 1.0 we don't need to specify it. Anything not medium or high will be considered
low. -->
<index-rule nodeType="mgnl:page"
            condition="@priority = 'medium'">
  <property boost="3.0">title</property>
</index-rule>
<index-rule nodeType="mgnl:page"
            condition="@priority = 'high'">
  <property boost="5.0">title</property>
</index-rule>
```

Finally, add a radio button to your page dialog for controlling page priority levels.

You may also reference properties in the condition that are not on the current node and/or specify the type of a node in the condition.

Aggregates

Sometimes it is useful to include the contents of descendant nodes into a single node to easier search on content that is scattered across multiple nodes.

Here we create an index aggregate on `mgnl:page` that includes the content of `mgnl:area` and `mgnl:component`. This will make it easier to search content on a page that is located in one of it's area or component subnodes.

```
<aggregate primaryType="mgnl:page">
  <include primaryType="mgnl:area">*</include>
  <include primaryType="mgnl:component">*</include>
</aggregate>
```

Analyzers

With this configuration part, you define how a property should be analyzed.

For example, let's say I wanted to target properties which I know store German language content with a German language analyzer.

```
<analyzer class="org.apache.lucene.analysis.de.GermanAnalyzer">
   <property>text_de</property>
</analyzer>
```

Custom configuration

You can create a custom indexing configuration for any workspace. Once created the file can be configured at the workspace.xml file of the workspace you wish to target. Changes to this configuration require a reindexing of the workspace.

An example of this would be the website specific example shown above or the dam specific configuration here:

This shows an example of node data aggregation. Since the magnolia metadata is stored on the mgnl:asset node and the image metadata/data is stored on a mgnl:resource subnode we can aggregate this into one lucene document.

indexing_configuration_dam.xml

```
<?xml version="1.0"?>
<!DOCTYPE configuration SYSTEM "http://jackrabbit.apache.org/dtd/indexing-configuration-1.2.dtd">
<configuration xmlns:nt="http://www.jcp.org/jcr/nt/1.0" xmlns:mgnl="http://www.magnolia.info/jcr/mgnl" xmlns:
jcr="http://www.jcp.org/jcr/1.0">
  <!--
      A global, generic indexing configuration used for all workspaces in Magnolia.
      It excludes some well known properties from the node scope fulltext index.
 -->
  <index-rule nodeType="nt:base">
    <!-- unlike other system properties, we want mgnl:tags to be indexed -->
    <property>mgnl:tags</property>
    <property isRegexp="true" nodeScopeIndex="false">mgnl:.*</property>
    <property isRegexp="true" nodeScopeIndex="false">jcr:.*</property>
    <!-- notice how the catch-all property should come at the end after all specific ones -->
    <property isRegexp="true">.*:.*</property>
  </index-rule>
  <!--
     The dam specific part: aggregates the actual contents of a document with mgnl:asset so that a full-text
query like

     select * from [mgnl:asset] as t where contains(t.*, 'some text')

     would return documents whose contents contain the searched terms.
  -->
  <aggregate primaryType="mgnl:asset">
    <include primaryType="mgnl:resource">*</include>
  </aggregate>
</configuration>
```