Search revisited

Problem

At its current implementation status, search presents several issues which make it unpractical if not unusable in most cases. (For the current search implementation see the old concept page at Concept - Search and Sort for Content Apps)

The main problem we want to look at here is well described by the following JIRA ticket

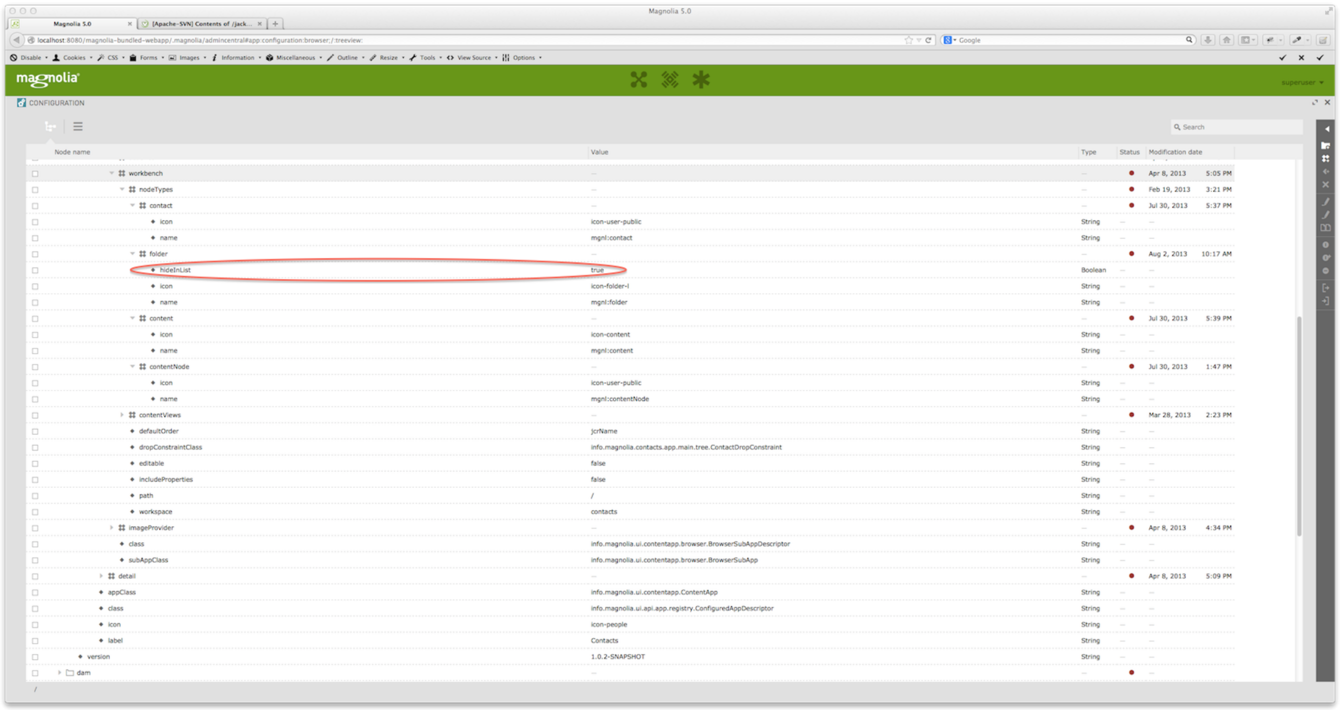
- **MGNLUI-1568** - Getting issue details... STATUS

Proposal

We need to enable searching for multiple node types by issuing **one single JCR query** which will also be used for "paginating" the results with the same mechanism used by the list view**.** In JCR SQL-2 this is feasible with the following syntax

```
select * from [nt:base] where ([jcr:primaryType] = 'foo' or [jcr:primaryType] = 'bar' or [jcr:primaryType] =
'qux')
```

Each node type declared in a *workbench* might have the **additional boolean property *hideInList*** . If not hidden, the node type is used by default in the JCR sql2 query for both list and search jcr containers.


localhost:8080/magnolia-bundled-webapp/.magnolia/admincentral#app:configuration:browser;/:treeview:

CONFIGURATION

Node name	Value	Type	Status	Modification date
workbench	—	—	●	Apr 8, 2013 5:05 PM
nodeTypes	—	—	●	Feb 19, 2013 3:21 PM
contact	—	—	●	Jul 30, 2013 5:37 PM
icon	icon-user-public	String	—	—
name	mgnl:contact	String	—	—
folder	—	—	●	Aug 2, 2013 10:17 AM
hideInList	true	Boolean	—	—
icon	icon-folder-l	String	—	—
name	mgnl:folder	String	—	—
content	—	—	●	Jul 30, 2013 5:39 PM
icon	icon-content	String	—	—
name	mgnl:content	String	—	—
contentNode	—	—	●	Jul 30, 2013 1:47 PM
icon	icon-user-public	String	—	—
name	mgnl:contentNode	String	—	—
contentViews	—	—	●	Mar 28, 2013 2:23 PM
defaultOrder	jcrName	String	—	—
dropConstraintClass	info.magnolia.contacts.app.main.tree.ContactDropConstraint	String	—	—
editable	false	String	—	—
includeProperties	false	String	—	—
path	/	String	—	—
workspace	contacts	String	—	—
imageProvider	—	—	●	Apr 8, 2013 4:34 PM
class	info.magnolia.ui.contentapp.browser.BrowserSubAppDescriptor	String	—	—
subAppClass	info.magnolia.ui.contentapp.browser.BrowserSubApp	String	—	—
detail	—	—	●	Apr 8, 2013 5:09 PM
appClass	info.magnolia.ui.contentapp.ContentApp	String	—	—
class	info.magnolia.ui.api.app.registry.ConfiguredAppDescriptor	String	—	—
icon	icon-people	String	—	—
label	Contacts	String	—	—
version	1.0.2-SNAPSHOT	String	—	—
dam	—	—	●	—

Mixins

If a mixin type is declared under /workbench/nodeTypes it will be included in the list and search views with the following syntax (we assume in the following example that a node type named *baz* is a mixin and is declared under /workbench/nodeTypes)

```
select * from [nt:base] where ([jcr:primaryType] = 'foo' or [jcr:mixinTypes] ='baz')
```

By default, mixins beginning with **jcr: nt: mix: rep:** will be discarded. The same rule for *hideInList* applies here. See more about mixin types in JCR 2.0 specs

Subtypes

Subtypes of node types declared in a workbench will be added to the list and search views, provided their parents are not defined as hidden (*hideInList = true*) nor strict (*strict = true*). By default, in node type definitions **hideInlist** and **strict** are **false**.

Performance

As the new query syntax involving a *where* clause on several node types may arise doubts about performance compared to a plain *select* on one node type, some basic manual tests have been conducted comparing the two syntaxes *(the groovy script used to help doing the tests is attached to this page).*

- Contacts workspace - **50000** nodes all of type *mgnl:contact*

Query	**first run**	**subsequent runs**
`select * from [mgnl:contact]`	`50000 nodes returned in ~5000ms`	`50000 nodes returned in ~1000ms`
`select * from [nt:base] where ([jcr:primaryType] = 'mgnl:contact')`	`50000 nodes returned in ~5000ms`	`50000 nodes returned in ~1000ms`

- Contacts workspace - **50000** nodes: 20000 *mgnl:contact,* 10000 *mgnl:folder,* 10000 *mgnl:content,* 10000 *mgnl:contentNode*

Query	**first run**	**subsequent runs**
`select * from [nt:base]`	`70000 nodes returned in ~ 6800ms`	`70000 nodes returned in ~ 1000ms`
`select * from [nt:base] where ([jcr:primaryType] = 'mgnl:contact' or [jcr:primaryType] = 'mgnl:content' or [jcr:primaryType] = 'mgnl:contentNode' or [jcr:primaryType] = 'mgnl:folder')`	`50000 nodes returned in ~ 6000ms`	50000 nodes returned in ~ `1000ms`

Query with limit 100 and offset 500	**first run**	**subsequent runs**
`select * from [nt:base] where ([jcr:primaryType] = 'mgnl:contact' or [jcr:primaryType] = 'mgnl:content' or [jcr:primaryType] = 'mgnl:contentNode' or [jcr:primaryType] = 'mgnl:folder')`	`~ 250ms`	`~ 250ms`

The above results seem to show that there is no performance penalty in querying for multiple node types with the where clause syntax.

Limiting search results to relevant matches

One negative feedback regarding current search concerns the large amount of matches a search sometimes produces which are often completely unrelated to the term queried for. Consider the following example.

We want to search for all properties in config named **admin.** We know there is only one in a typical CE setup, that is the one directly under **/server.** However , searching for admin will return more than 4900 nodes!, that is all config nodes. That's because the current search implementation will perform a full text search on all properties

of all nodes in the workspace, something like *select * from [mgnl:contentNode] as t where contains(t.*, 'admin').* The problem in this case is that each node has by default a *jcr:createdBy* property which is automatically added upon node creation and whose value is by default *admin*

To solve this issue we need to exclude some well known propertites from the full text search index. This is doable thanks to http://wiki.apache.org/jackrabbit/IndexingConfiguration.

"Per default the configured properties are fulltext indexed if they are of type STRING and included in the node scope index. That is, you can do a *jcr:contains(., 'foo')* and it will return nodes that have a string property that contains the word foo. This behaviour can be disabled: ..." .

Here is the *indexing_configuration.xml* file we will use

```
<?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">
    <property isRegexp="true" nodeScopeIndex="false">mgnl:.*</property>
    <property isRegexp="true" nodeScopeIndex="false">jcr:.*</property>
    <property isRegexp="true">.*:.*</property>
</index-rule>
</configuration>
```

The file will be placed under ***src/main/resources/info/magnolia/jackrabbit*** in Magnolia's core module which means it will eventually be available in the JVM classpath once the artifact is built. To make JR aware of the indexing configuration, each provided repository configuration (*jackrabbit-bundle-derby-search.xml* and friends) will have an added parameter called *indexingConfiguration* as in the following excerpt

```
...
<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"/>
...
```

The query eventually produced will be something like this

```
select * from [nt:base] as t where (([jcr:primaryType] = 'mgnl:content' or [jcr:primaryType] = 'mgnl:
contentNode') and (localname() like 'admin%' or t.admin is not null or contains(t.*, 'admin')))
```

localname() like 'admin%' will search for nodes whose jcr name begins with "*admin*"

t.admin is not null will look for the existence of properties called *admin*

contains(t.*, 'admin') will perform a full-text search for *admin* within **all** properties across all nodes **but** those excluded by our indexing configuration, meaning, in our case, that j*cr:createdBy* won't be searched. The actual result of the above query is 7 matching results (there are some nodes whose name contains the word *admin*), one of which is the node ***/server*** containing the *admin* property (as far as I could see, it is not possible to get directly the property rather than its parent node).

The three above conditions are needed as we don't know beforehand if the user means to search for a node name, a property name or a value. It is noteworthy to mention that excluding certain properties from the full-text index still makes them available in other types of query. For example *select * from [nt:base] as t where t.[jcr:createdBy] = 'admin'* **will find all the expected matches.**