Groovy Shell Scripts

This page has been migrated to Git. Please add new scripts here: groovy-shell-scripts

Following some helpful scripts

Also see Useful shell scripts (some duplication, feel free to merge)

A note on usage

Magnolia CMS providing you with a Groovy app where you can create and edit your custom Groovy scripts for some development and administration purposes. For below scripts, just copy and paste them into the console, run it for your interest.

Access Content

Magnolia 4.5 & 5

```
session = MgnlContext.getJCRSession("website");
node = session.getNode("/demo-project")
```

Magnolia 4.4

```
hm = MgnlContext.getHierarchyManager("website")
node = hm.getContent("/demo-project")
```

Useful imports & initialization

```
import info.magnolia.cms.core.*;
import info.magnolia.cms.util.*;
import info.magnolia.context.*;

ctx = MgnlContext.getSystemContext();
MgnlContext.setInstance(ctx);
hm = ctx.getHierarchyManager("config");
```

Using the Visitor

```
visitor = {node ->
  println(node.handle)
}

ContentUtil.visit(root, visitor as ContentUtil.Visitor, new NodeTypeFilter(ItemType.CONTENT))
```

Visit hierarchy

```
root = hm.root;
ContentUtil.visit(root, new ContentUtil.Visitor(){
   visit(Content node) {
     if(node.getIndex()>1){
       print(node);
     }
   }
});
```

Dump a node

```
print(info.magnolia.cms.util.DumperUtil.dump(node));
```

Get Activation Status

Magnolia 5

```
java.lang.String querySql2 = "select * from [nt:base]";
java.lang.String queryWorkspace = "website";
javax.jcr.NodeIterator nodeList = info.magnolia.cms.util.QueryUtil.search(queryWorkspace, querySql2, javax.jcr.
query.Query.JCR_SQL2, "mgnl:page");

println "Iterating through nodes...";
while(nodeList.hasNext()) {
  javax.jcr.Node currNode = nodeList.nextNode();
  java.lang.Integer status = info.magnolia.jcr.util.NodeTypes.Activatable.getActivationStatus(currNode);

  if (status == NodeTypes.Activatable.ACTIVATION_STATUS_MODIFIED){
    println "modified : ${currNode.path}";
  }
  else if (status == NodeTypes.Activatable.ACTIVATION_STATUS_ACTIVATED){
    println "activated : ${currNode.path}";
  }
  else {
    println "not-activated : ${currNode.path}";
  }
}
println "Node iterations done";
```

Set Activation Status (green)

```
  md = node.metaData
  md.setActivated()
  md.setActivatorId(MgnlContext.user.name)
  md.setLastActivationActionDate()
```

Find something in Bootstrap-files

```
files = ClasspathResourcesUtil.findResources(".*/mgnl-bootstrap/.*")
files.each(){name ->
  text = getClass().getResourceAsStream(name).getText()
  if (text =~ "microsoft")
    println(name)
}
```

Export dialogs/templates/paragraphs from a module at once to the filesystem

Each template, paragraph and dialog will be export in its own file, which is much more handy than one big file for all.

```
import info.magnolia.module.admininterface.pages.DevelopmentUtilsPage;

DevelopmentUtilsPage p = new DevelopmentUtilsPage(null, null, null);
// remember to write this file to a directory where you have the necessary permissions
// i.e. /var/lib/tomcat7/webapps/magnoliaAuthor_transa/tmp/export.xml when executing from Author
p.setRootdir("/some/place/on/your/servers/filesystem/");
p.setRepository("config");

p.setParentpath("/modules/myModule/templates");
p.backupChildren();
p.setParentpath("/modules/myModule/paragraphs");
p.backupChildren();
p.setParentpath("/modules/myModule/dialogs");
p.backupChildren();
```

Export configuration to be used in diffs

```
import groovy.xml.*

hm = MgnlContext.getHierarchyManager("config")
root = hm.getContent("/modules/standard-templating-kit")
// remember to write this file to a directory where you have the necessary permissions
// i.e. /var/lib/tomcat7/webapps/magnoliaAuthor_transa/tmp/export.xml when executing from Author
out = new FileWriter(new File("/Users/pbracher/workspaces/magnolia-4.2/export.xml"))

buffer = new ByteArrayOutputStream()

hm.workspace.session.exportDocumentView("/modules/standard-templating-kit", buffer, false, false)

str = buffer.toString()

def root = new XmlParser(false, false).parseText(str)

root.depthFirst().each({node ->
  if(node.name()=="MetaData"){
    node.parent().remove(node)
  }

  node.attributes().remove("jcr:uuid")
  node.attributes().remove("jcr:created")
})

def traverse(builder, node) {
    builder."${node.name()}"() {
        node.attributes().each(){attribute ->
            builder."${attribute.key}"(attribute.value)
        }
        node.children().each {
            traverse(builder, it)
        }
    }
}

builder = new MarkupBuilder(out)
traverse(builder,root)
```

Diff two xml files

Files exported as described above

```
import groovy.xml.*

homeDir = new File("/Users/pbracher/workspaces/magnolia-4.2")

def update = new XmlParser(false, false).parse(new FileReader(new File(homeDir,"update.xml")))
def installation = new XmlParser(false, false).parse(new FileReader(new File(homeDir,"installation.xml")))

def path(node){
  if(node.parent()){
    return path(node.parent()) + "/" + node.name()
  }
  return node.name()
}

def diff(left, right){
  left.children().each(){child ->
      // if text value
      if(child instanceof String){
         if(child != right.text()){
             println "${path(left)}: ${child} --> ${right.text()}"
         }
      }
      // if element
      else{
          if(right[child.name()].size() == 0){
              println "${path(child)}: removed}"
          }
          else{
              diff(child, right[child.name()][0])
          }
      }
  }
  right.children().each(){child ->
      // if text value
      if(!(child instanceof String)){
          if(left[child.name()].size() == 0){
              println "${path(child)}: added}"
          }
          else{
              if(!(child.children[0] instanceof String)){
                  diff(right[child.name()][0], child)
              }
          }
      }
  }

}

diff(update, installation)
```

SQL query example

Deletes nodes with null value of mgnl:template MetaData property

```
hm = ctx.getHierarchyManager("website")

query = hm.getQueryManager().createQuery("SELECT * FROM nt:base WHERE mgnl:template = ''", "sql")
result = query.execute()
contentCollection = result.getContent(ItemType.CONTENTNODE.getSystemName())
println "found ${contentCollection.size()} nodes"

contentCollection.each { c ->
  println "deleting ${c.name}"
  c.delete()
  }
hm.save()
```

Delete pages with given template type & descendant of given path

Deletes nodes that match both criterias:
- template type "website-module:pages/myShopBrand"
- descendant of "/Home/Brands"

```
java.lang.String querySql2 = "select * from [nt:base] WHERE ISDESCENDANTNODE('/Home/Brands') AND [mgnl:
template] = 'website-module:pages/myShopBrand'";
java.lang.String queryWorkspace = "website";

javax.jcr.NodeIterator nodeList = info.magnolia.cms.util.QueryUtil.search(queryWorkspace, querySql2, javax.jcr.
query.Query.JCR_SQL2, "mgnl:page");

println "Removing nodes :";
while(nodeList.hasNext()) {
  javax.jcr.Node currNode = nodeList.nextNode();
  println "${currNode.path}";
  currNode.remove();
}

info.magnolia.context.MgnlContext.getJCRSession(queryWorkspace).save();
println "Nodes removal finished";
```

Export a node with no children

As the ability to export a node without children is not available in the UI, you might resort to this script in case of need.

```
hm = MgnlContext.getHierarchyManager("website")
out = new FileWriter(new File("/Users/me/mynode-export.xml"))
buffer = new ByteArrayOutputStream()

/*
 *  last two boolean params are
 *  skipBinary A <code>boolean</code> governing whether binary properties are to be serialized.
 *  noRecurse  A <code>boolean</code> governing whether the subgraph at absPath is to be recursed.
 */
hm.workspace.session.exportDocumentView("/mynode", buffer, true, true)

str = buffer.toString()

out.write(str)

out.close()
```

Export a page to JSON

```
import info.magnolia.jcr.predicate.NodeTypePredicate;
import info.magnolia.jcr.predicate.RuleBasedNodePredicate;
import info.magnolia.jcr.util.NodeUtil;
import info.magnolia.jcr.wrapper.ChildFilteringNodeWrapper;

import java.io.StringWriter;

import javax.jcr.Node;
import javax.jcr.RepositoryException;
import javax.jcr.Session;

import org.apache.sling.commons.json.jcr.JsonItemWriter;

pathToNode = "/home";

session = MgnlContext.getJCRSession("website");
page = session.getNode(pathToNode);

Rule rule = new Rule("mgnl:page", "");
rule.reverse();
ChildFilteringNodeWrapper wrapped =
    new ChildFilteringNodeWrapper(page, new RuleBasedNodePredicate(rule));

jsonWriter = new JsonItemWriter(null);
stringWriter = new StringWriter();

jsonWriter.dump(wrapped, stringWriter, -1, true);
json = stringWriter.toString();
```

Export page and subpages into separate xml files

Magnolia 4.5 & 5

```
import info.magnolia.cms.util.Rule;
import info.magnolia.context.MgnlContext;
import info.magnolia.importexport.filters.MetadataUuidFilter;
import info.magnolia.jcr.predicate.RuleBasedNodePredicate;
import info.magnolia.jcr.util.NodeUtil;
import info.magnolia.jcr.wrapper.ChildFilteringNodeWrapper;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

import javax.jcr.Node;
import javax.jcr.RepositoryException;
import javax.jcr.Session;

import org.apache.commons.io.IOUtils;
import org.apache.jackrabbit.commons.xml.SystemViewExporter;
import org.apache.jackrabbit.commons.xml.ToXmlContentHandler;
import org.apache.xml.serialize.OutputFormat;
import org.apache.xml.serialize.XMLSerializer;

import org.xml.sax.ContentHandler;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;

outputDirectory = "/path/to/directory/";
pathToNode = "/demo-project/about";

session = MgnlContext.getJCRSession("website");
page = session.getNode(pathToNode);
subpages = NodeUtil.collectAllChildren(page, new RuleBasedNodePredicate(new Rule("mgnl:page", "")));

generateXML(page);

for(Node subpage : subpages) generateXML(subpage);

def generateXML(node) {
  tempFile = File.createTempFile("file", ".xml");
  contentHandler = new ToXmlContentHandler(new FileOutputStream(tempFile));
  exporter = new SystemViewExporter(session, contentHandler, true, true);
  rule = new Rule("mgnl:page" + "," + "mgnl:reserve", ",");
  rule.reverse();
  wrapped = new ChildFilteringNodeWrapper(node, new RuleBasedNodePredicate(rule));
  exporter.export(wrapped);

  file = new File(outputDirectory + node.getPath().replace("/", ".").replaceFirst(".", "/") + '.xml');
  reader = XMLReaderFactory.createXMLReader(org.apache.xerces.parsers.SAXParser.class.getName());
  outputStream = new FileOutputStream(file);
  inputStream = new FileInputStream(tempFile);
  outputFormat = new OutputFormat();
  outputFormat.setPreserveSpace(false);
  outputFormat.setIndenting(true);
  outputFormat.setIndent(2);
  outputFormat.setLineWidth(120);
  metadataUuidFilter = new MetadataUuidFilter(reader, true);
  metadataUuidFilter.setContentHandler(new XMLSerializer(outputStream, outputFormat));
  metadataUuidFilter.parse(new InputSource(inputStream));

  IOUtils.closeQuietly(inputStream);
  outputStream.flush();
  IOUtils.closeQuietly(outputStream);
  tempFile.delete();
}
```

Export page and subpages into separate xml files (6.2.x)

```
import info.magnolia.cms.util.Rule;
import info.magnolia.context.MgnlContext;
import info.magnolia.importexport.filters.MetadataUuidFilter;
import info.magnolia.jcr.predicate.RuleBasedNodePredicate;
import info.magnolia.jcr.util.NodeUtil;
import info.magnolia.jcr.wrapper.ChildFilteringNodeWrapper;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

import javax.jcr.Node;
import javax.jcr.RepositoryException;
import javax.jcr.Session;

import org.apache.commons.io.IOUtils;
import org.apache.jackrabbit.commons.xml.SystemViewExporter;
import org.apache.jackrabbit.commons.xml.ToXmlContentHandler;

import org.xml.sax.ContentHandler;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.sax.TransformerHandler;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.sax.SAXTransformerFactory;

outputDirectory = "/path/to/directory/";
pathToNode = "/travel";

session = MgnlContext.getJCRSession("website");
page = session.getNode(pathToNode);
subpages = NodeUtil.collectAllChildren(page, new RuleBasedNodePredicate(new Rule("mgnl:page", "")));

generateXML(page);

for(Node subpage : subpages) generateXML(subpage);

def generateXML(node) {
  tempFile = File.createTempFile("file", ".xml");
  contentHandler = new ToXmlContentHandler(new FileOutputStream(tempFile));
  exporter = new SystemViewExporter(session, contentHandler, true, true);
  rule = new Rule("mgnl:page" + "," + "mgnl:reserve", ",");
  rule.reverse();
  wrapped = new ChildFilteringNodeWrapper(node, new RuleBasedNodePredicate(rule));
  exporter.export(wrapped);

  file = new File(outputDirectory + node.getPath().replace("/", ".").replaceFirst(".", "/") + '.xml');

  SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
  saxParserFactory.setNamespaceAware(true);
  SAXParser saxParser = saxParserFactory.newSAXParser();
  reader = saxParser.getXMLReader();

  outputStream = new FileOutputStream(file);
  inputStream = new FileInputStream(tempFile);

  SAXTransformerFactory transformerFactory = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
  TransformerHandler transformerHandler = transformerFactory.newTransformerHandler();
  Transformer transformer = transformerHandler.getTransformer();
  transformer.setOutputProperty("indent", "yes");
```

```
  transformer.setOutputProperty("doctype-public", "yes");
  transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", String.valueOf("2"));
  transformerHandler.setResult(new StreamResult(outputStream));

  metadataUuidFilter = new MetadataUuidFilter(reader, true);
  metadataUuidFilter.setContentHandler(transformerHandler);
  metadataUuidFilter.parse(new InputSource(inputStream));

  IOUtils.closeQuietly(inputStream);
  outputStream.flush();
  IOUtils.closeQuietly(outputStream);
  tempFile.delete();
}
```

List of dms files ordered by their size

```
hm = MgnlContext.getHierarchyManager("dms")
query = hm.getQueryManager().createQuery("select * from mgnl:resource", "sql")
result = query.execute()
contentCollection = result.getContent(ItemType.CONTENTNODE.getSystemName())
def sortClosure = {a, b ->
    def aValue = Integer.parseInt(a.getNodeData("document").getAttribute("size"))
    def bValue = Integer.parseInt(b.getNodeData("document").getAttribute("size"))
    return bValue.compareTo(aValue)
}
contentCollection.sort(sortClosure)
contentCollection.each { c ->
  print c.getHandle() + " ";
  nodeData = c.getNodeData("document");
  println nodeData.getAttribute("size") + "B";
  }
```

Clean mgnlSystem of mgnl: nodes

```
import javax.jcr.Node
import javax.jcr.NodeIterator

repo = "mgnlSystem"
println "Cleaning ${repo}"
sysCtx = MgnlContext.getSystemContext()
session = sysCtx.getHierarchyManager(repo)

def items = 0
try {
    Content root = session.getRoot()
    NodeIterator nodes = root.getJCRNode().getNodes()

    while (nodes.hasNext()) {
        Node child = nodes.nextNode();
        if (child.getPrimaryNodeType().getName().startsWith('mgnl:')) {
            println "Removing backup copy of '${child.getPath()}' from ${repo}, please verify that page was
activated correctly to this public instance."
            child.remove()
            items++
        }
    }
} catch (Exception e) {
    println e
}
println "${items} nodes removed."
session.save()
```

Dump Java object properties recursively

```
import info.magnolia.objectfactory.Components
import info.magnolia.module.templatingkit.sites.*
import info.magnolia.objectfactory.guice.GuiceComponentProvider
import org.codehaus.groovy.runtime.NullObject

/**
* Usage sample. We want to dump the state of the default Site object currently in memory. Filtering out
'inheritance' field.
*/
def filtered = ['inheritance']
def defSite = Components.componentProvider.getSingleton(SiteManager.class).defaultSite
dumpObject(defSite, filtered, 0)
return 'done'

/**
* Implementation.
*/
def dumpObject(object, filtered, level) {

def space = ''
level.times {space += "\t"}
if (object instanceof Map)
   properties = object
 else
   properties = object.properties

sortedProps = properties?.sort{it.key}?.sort{!isPrimitive(it.value)}
filteredProps = sortedProps.collect{it}.findAll{!filtered.contains(it.key)}
filteredProps.each {
    if(isLeaf(it.value)){
      println space + '- '+ it
    } else {
      println space + '+ ' + it.key
      dumpObject(it.value,filtered, level+1)
    }
  }
}
def isLeaf(value) {
    return [GuiceComponentProvider, NullObject, Boolean, String, Double, Integer, Class].any { it.
isAssignableFrom(value.getClass())}
}
```

Perform a bulk operation on nodes

Say you have 30000 users under */users/admin* and you need to delete them all. Notice that the *save()* method is called every 1000 items so to flush the JCR session periodically and avoid a possible *OutOfMemory* exception if we did it only once at the end of the loop.

```
hm = ctx.getHierarchyManager('users')
users = hm.getContent('/admin').getChildren('mgnl:user')

count = 0
users.eachWithIndex { user, i ->
  println "Deleting user $user.name..."
  user.delete()
  count++
  if(i % 1000 == 0)
    hm.save()
}
//perform a final saving because there might be some items left in the current JCR session
hm.save()
return "Done deleting $count users."
```

Export nodes with certain node types to single files

```
import info.magnolia.jcr.predicate.NodeTypePredicate;
import info.magnolia.importexport.DataTransporter

repository = "users"
nodeType = "mgnl:user"

session = ctx.getJCRSession(repository)
collection = NodeUtil.collectAllChildren(session.getRootNode(), new NodeTypePredicate(nodeType))
collection.each { node->
  try {
    xmlFileOutput = new FileOutputStream("/your/path/" + node.name + ".xml")
    DataTransporter.executeExport(xmlFileOutput, false, false, session, node.path, repository, DataTransporter.
XML)
    xmlFileOutput.close()
  } catch (Exception e) {
    println "can't export node: " + node.path
  }
}
```

Remove properties from some given bootstrap files and export the clean up files

Caveat: for some reason the files output by the script are missing the prolog <?xml version="1.0" encoding="UTF-8"?>

However, one can insert it back after generating the files with a sed command, e.g.

```
sed -i '.original' "1s/^/<?xml version="1.0" encoding="UTF-8"?> /" dam*.xml
```

```
modules = ['ui-admincentral', 'pages', 'dam', 'contacts', 'ui-mediaeditor', 'security-app', 'ui-framework',
'sample-app', 'messages-app']
propertiesToRemove = ['label', 'icon', 'confirmationMessage', 'i18nBasename', 'description']
modules.each { module ->
  println "Cleaning up module ${module}..."
  files = ClasspathResourcesUtil.findResources(".*/mgnl-bootstrap/.*config.modules.${module}.*")
  files.each{ name ->

      text = getClass().getResourceAsStream(name).getText()
      println "Checking bootstrap file name ${name}..."
      xml = new XmlParser(false,false).parseText(text)
      foundPropertiesToRemove = false

      xml.depthFirst().each{node ->
        if(node.name()=="sv:property" &amp;amp;amp; propertiesToRemove.contains(node.attributes()['sv:name'])){
          //println "Node named [${node.parent().attributes()['sv:name']}] has a property named [${node.
attributes()['sv:name']}] which will be removed"
          foundPropertiesToRemove = true
          node.parent().remove(node)
        }
     }
     if(foundPropertiesToRemove) {
       filePath = "/Users/me/test${name.substring(name.lastIndexOf("/"))}"
       println "Found properties to remove. Creating file $filePath ..."
       file = new File(filePath)
       xmlNodePrinter = new XmlNodePrinter(new PrintWriter(new FileWriter(file)))
       xmlNodePrinter.setPreserveWhitespace(true)
       xmlNodePrinter.print(xml)
     }
   }
}
return 'done'
```

Permissions List

This script produces a list of permission which includes:

- which user has which group and role
- which group has which other group
- which group has which role
- what acl is defined for the role

It's meant to be used on the 4.4 branch since 4.5 has a permissions tool built it.

```
import info.magnolia.cms.security.*;
import info.magnolia.cms.beans.config.ContentRepository;
import info.magnolia.cms.security.auth.ACL;
import org.apache.commons.lang.StringUtils;
import info.magnolia.cms.security.PermissionImpl;
allUsers = SecuritySupport.Factory.getInstance().getUserManager().getAllUsers();
out.println '- which user has which group and role';
out.println '************************************';
for (aUser in allUsers)
{
  out.println 'User Name: ' + aUser.name;

  out.println '\tgroups:';
  groups = aUser.getAllGroups();
  for (group in groups) out.println '\t\t' + group;

  out.println '\troles:';
  roles = aUser.getAllRoles();
  for (role in roles) out.println '\t\t' + role;

  out.println '';
}

out.println '';

allGroups = SecuritySupport.Factory.getInstance().getGroupManager().getAllGroups();
out.println '- which group has which other group';
out.println '- which group has which role';
out.println '- what acl is defined for the role';
out.println '**********************************';
for (aGroup in allGroups)
{
  out.println 'Group Name: ' + aGroup.name;

  out.println '\tgroups:';
  groups = aGroup.getAllGroups();
  for (group in groups) out.println '\t\t' + group;

  out.println '\troles:';
  roles = aGroup.getRoles();
  for (role in roles) {
    out.println '\t\t' + role;

    HierarchyManager hm = MgnlContext.getSystemContext().getHierarchyManager(ContentRepository.USER_ROLES);
    node = hm.getContent(role);
    Iterator it = node.getChildren(ItemType.CONTENTNODE.getSystemName(), "acl*").iterator();
    while (it.hasNext()) {
      Content aclEntry = (Content) it.next();
      String name = StringUtils.substringAfter(aclEntry.getName(), "acl_");

      String repositoryName;
      String workspaceName;
      if (!StringUtils.contains(name, "_")) {
        repositoryName = name;
        workspaceName = ContentRepository.getDefaultWorkspace(name);
        name += ("_" + workspaceName); // default workspace must be added to the name
      }
      else {
        String[] tokens = StringUtils.split(name, "_");
        repositoryName = tokens[0];
        workspaceName = tokens[1];
```

```
      }

      Iterator it2 = aclEntry.getChildren().iterator();
      while(it2.hasNext()) {
        acl = it2.next();
        path = acl.getJCRNode().getProperty('path').getValue().getString();
        permissions = acl.getJCRNode().getProperty('permissions').getValue().getLong();

        out.print '\t\t\t';
        switch (permissions) {
          case 0: out.print 'Deny access'; break;
          case 8:
            if ('uri'.equals(repositoryName)) { out.print 'Get'; break; }
            else { out.print 'Read only'; break; }
          case 11: out.print 'Post (Write)'; break;
          case 63:
            if ('uri'.equals(repositoryName)) { out.print 'Get &amp;amp; Post'; break; }
            else { out.print 'Read/Write'; break; }
          case 75: out.print 'Moderate (and Post)'; break;
          case 79: out.print 'Moderate and Delete'; break;
          case 111: out.print 'Administer (Moderate, Delete and Activate)'; break;
          default: out.print permissions;
        }

        out.println ' permission in the workspace ' + repositoryName + ' with path ' + path;
      }
    }
  }

  out.println '';
}

out.println '';
```

Restore deleted pages recursively

```
import javax.jcr.version.Version;
import info.magnolia.jcr.predicate.NodeTypePredicate;


visitor = { node ->
  def lastGoodVersion

  if (node.isNodeType("mgnl:deleted")) {
  versions = versionMan.getAllVersions(node);
  while( versions.hasNext()) {
    item = versions.next();
    lastGoodVersion = item.name
    frozen = item.frozenNode
  //  print item.getName() + "::"
    if (frozen.hasProperty("mgnl:comment")) {
 //     println frozen.getProperty("mgnl:comment").string
      if ("Automatically created version prior deletion" == frozen.getProperty("mgnl:comment").string) {
        break;
      }
    } else {
  //      println "no comment"
    }
}


  println "for page " + node.path  + " the lucky winner is: " + lastGoodVersion

  version = versionMan.getVersion(node, lastGoodVersion)
  versionMan.restore(node, version, true)


}
}







versionMan = info.magnolia.objectfactory.Components.getComponent(info.magnolia.cms.core.version.VersionManager.
class);
session = MgnlContext.getSystemContext().getJCRSession("website");
NodeUtil.visit(session.getNode("/myTopLevelMaybeDeletedPage"), visitor as NodeVisitor, new NodeTypePredicate
("mgnl:page"))
```

Restore deleted contacts recursively (could be reused for any workspace/content)

```
import info.magnolia.jcr.util.NodeUtil
import org.apache.jackrabbit.commons.predicate.NodeTypePredicate
import info.magnolia.cms.core.version.VersionManager
import info.magnolia.objectfactory.Components
import javax.jcr.version.Version
import javax.jcr.version.VersionIterator



session = ctx.getJCRSession("contacts")

root = session.getNode("/")
allChildren = NodeUtil.collectAllChildren(root, new NodeTypePredicate("mgnl:contact", true))

allChildren.each{
    templateID = NodeTypes.Renderable.getTemplate(it)
    if(templateID == "ui-admincentral:deleted"){
        println("Found deleted contact: "+it.getPath())
        println("Restoring last version of: "+it.getPath())


        VersionManager versionManager = Components.getComponent(info.magnolia.cms.core.version.VersionManager.
class);
        Version previousVersion = null;
        VersionIterator versionIterator = versionManager.getAllVersions(it);

        while (versionIterator.hasNext()) {
            previousVersion = versionIterator.nextVersion();
        }
        if(previousVersion != null){

            versionManager.restore(it, previousVersion, true);
            println("Version Sucessfully restored of: "+it.getPath())
        } else{
            println("No Version found for node: "+it.getPath())
        }


    } else {
         println("Is NOT a deletd contact: "+it.getPath())
    }
}


session.save()
```

Migration from one DB system to another

This script lets you migration from one DB system (e.g. Derby DB) to another one (e.g. MySQL DB). Make sure that no-one is writing any data to the repository you want to migrate from to prevent inconsistencies.

Should work with any Magnolia that uses jackrabbit 1.6+.

Magnolia 4.5 & 5

```
import info.magnolia.repository.RepositoryConstants;
import javax.jcr.Repository;
import org.apache.jackrabbit.core.RepositoryCopier;
import org.apache.jackrabbit.core.RepositoryImpl;
import org.apache.jackrabbit.core.config.RepositoryConfig;
import java.io.File;

// configuration to which you want to migrate
String repositoryConfig = ""; // e.g. /some/path/WEB-INF/config/repo-conf/jackrabbit-bundle-mysql-search.xml
String repositoryFolder = ""; // target folder where files that are stored on FS will be transferred

// get current repository
Repository source = ctx.getJCRSession(RepositoryConstants.WEBSITE).getRepository();
// create RepositoryConfig from given settings
RepositoryConfig targetConfig = RepositoryConfig.create(new File(repositoryConfig), new File(repositoryFolder));
// do the migration
RepositoryCopier.copy((RepositoryImpl) source, targetConfig);
```

Magnolia 4.4

```
import javax.jcr.Repository;
import org.apache.jackrabbit.core.RepositoryCopier;
import org.apache.jackrabbit.core.RepositoryImpl;
import org.apache.jackrabbit.core.config.RepositoryConfig;
import java.io.File;

// configuration to which you want to migrate
String repositoryConfig = ""; // e.g. /some/path/WEB-INF/config/repo-conf/jackrabbit-bundle-mysql-search.xml
String repositoryFolder = ""; // target folder where files that are stored on FS will be transferred

// get current repository
Repository source = ctx.getHierarchyManager("website").getWorkspace().getSession().getRepository();
// create RepositoryConfig from given settings
RepositoryConfig targetConfig = RepositoryConfig.create(new File(repositoryConfig), new File(repositoryFolder));
// do the migration
RepositoryCopier.copy((RepositoryImpl) source, targetConfig);
```

Removing / Cleaning the versions (5.x)

```
import info.magnolia.cms.core.version.VersionManager;
import info.magnolia.context.MgnlContext;
import info.magnolia.jcr.predicate.NodeTypePredicate;
import info.magnolia.jcr.util.NodeUtil;
import info.magnolia.jcr.util.NodeVisitor;
import javax.jcr.RepositoryException;
import javax.jcr.Session;

// specify which website you want to clean here
String path = "/travel";

VersionManager vm = VersionManager.getInstance();
Session session =  MgnlContext.getJCRSession("website");
javax.jcr.Node root =session.getNode(path);
visitor = { node ->
    vm.removeVersionHistory(node);
}

NodeUtil.visit(root,  visitor as NodeVisitor, new NodeTypePredicate("mgnl:page"));
session.save();
```

Remove Node (versions 4.x, 5.x)

```
try {
session = MgnlContext.getJCRSession("website");
session.removeItem("/hh");
session.save();
}catch (Exception e) {
    print "Error " +  e
}
return "done removing"
```

Fix activation status (e.g. after manual import of content into public instance) (versions 4.5.x)

Please backup your website workspace beforehand.

Public instance

```
import info.magnolia.jcr.util.MetaDataUtil
import info.magnolia.jcr.predicate.NodeTypePredicate

file = "" // path of the file where to store result of the script
path = "/" // path of the tree to traverse

buffer = new BufferedWriter(new OutputStreamWriter(
    new FileOutputStream(file), "UTF-8"))

root = ctx.getJCRSession("website").getNode(path)

visitor = { node ->
  if (node.getPrimaryNodeType().getName().startsWith("mgnl:")) {
    date = MetaDataUtil.getLastModification(node)
    str = node.identifier + "|" + node.path + "|" + (date != null ? date.timeInMillis : "0")
    println str
    buffer.write(str)
    buffer.newLine()
  }
}

NodeUtil.visit(root, visitor as NodeVisitor, new NodeTypePredicate("mgnl:page", false))

buffer.close()
println "done"
```

Author instance

```
import org.apache.commons.lang.StringUtils
import javax.jcr.RepositoryException
import info.magnolia.jcr.util.MetaDataUtil
import info.magnolia.jcr.predicate.NodeTypePredicate
import org.apache.commons.io.FileUtils

filePath = ""
rootPath = ""

pagePaths = []
lastModDates = []
missing = []

for (String line : FileUtils.readLines(new File(filePath))) {
  pagePaths.add(StringUtils.split(line, "|")[1])
  lastModDates.add(StringUtils.split(line, "|")[2])
}

session = ctx.getJCRSession("website")

i = 0
visitor = { node ->
  if (node.getPrimaryNodeType().getName().startsWith("mgnl:")) {
    if (pagePaths.contains(node.getPath())) {
        lastModDate = Long.valueOf(lastModDates.get(pagePaths.indexOf(node.getPath())))
        metaData = MetaDataUtil.getMetaData(node)
        date = metaData.getModificationDate()
        if (date != null) {
          if (date.timeInMillis == lastModDate) {
            // modification date of same page on author &amp;amp; public equals
            // this means page is up to date and we can set its status to activated
            if (!metaData.getIsActivated()) {
              println "Set green activation status on page: " + node.path
              metaData.setActivated()
            }
          } else if (date.timeInMillis > lastModDate) {
            // modification date on author is greater then on public
```

```
            // we need to set status to activated &amp;amp; also check if lastaction date
            // is before lastmodification date
            activationDate = metaData.getLastActionDate()
            if (!date.after(activationDate)) {
              newDate = date.clone()
              newDate.add(Calendar.HOUR_OF_DAY, -1)
              metaData.setProperty("mgnl:lastaction", newDate)
            }
            metaData.setActivated()
            println "Set orange activation status on page: " + node.path
          } else {
            // skip
          }
        }
        missing.add(node.getPath())
      } else {
        MetaDataUtil.getMetaData(node).setUnActivated()
        println "Set red activation status on page: " + node.path
      }
    if (i++ % 100 == 0) {
      session.save()
      println "session saved"
    }
  }
}

NodeUtil.visit(session.getNode(rootPath), visitor as NodeVisitor, new NodeTypePredicate("mgnl:page", false))

session.save()
println "session saved"


for (String page : missing) {
    println "Page " + page + " is missing on author but is present on public!"
}
```

Fix activation status (e.g. after manual import of content into public instance) (versions 5.x)

Please backup your website workspace beforehand.

Public instance

```
import info.magnolia.jcr.util.MetaDataUtil
import info.magnolia.jcr.predicate.NodeTypePredicate

path = "" // path of the tree to traverse
file = "" // path of the file where to store result of the script

buffer = new BufferedWriter(new OutputStreamWriter(
    new FileOutputStream(file), "UTF-8"))

root = ctx.getJCRSession("website").getNode(path)

visitor = { node ->
  if (node.getPrimaryNodeType().getName().startsWith("mgnl:")) {
    date = NodeTypes.LastModified.getLastModified(node)
    str = node.getIdentifier() + "|" + node.getPath() + "|" + (date != null ? date.timeInMillis : "0")
    println str
    buffer.write(str)
    buffer.newLine()
  }
}

NodeUtil.visit(root, visitor as NodeVisitor, new NodeTypePredicate("mgnl:page", false))

buffer.close()
```

Author instance

```
import org.apache.commons.lang.StringUtils
import javax.jcr.RepositoryException
import info.magnolia.jcr.util.MetaDataUtil
import info.magnolia.jcr.predicate.NodeTypePredicate
import org.apache.commons.io.FileUtils

filePath = "/Users/jsimak/f.txt"
rootPath = "/travel"

pagePaths = []
lastModDates = []

for (String line : FileUtils.readLines(new File(filePath))) {
  pagePaths.add(StringUtils.split(line, "|")[1])
  lastModDates.add(StringUtils.split(line, "|")[2])
}

session = ctx.getJCRSession("website")

i = 0;
visitor = { node ->
  if (node.getPrimaryNodeType().getName().startsWith("mgnl:")) {
    if (pagePaths.contains(node.getPath())) {
        lastModDate = Long.valueOf(lastModDates.get(pagePaths.indexOf(node.getPath())))
        date = NodeTypes.LastModified.getLastModified(node)
        if (date != null) {
          if (date.timeInMillis == lastModDate) {
            // modification date of same page on author &amp;amp; public equals
            // this means page is up to date and we can set its status to activated
            if (NodeTypes.Activatable.getActivationStatus(node) == 0) {
              println "Set green activation status on page: " + node.getPath()
              node.setProperty(NodeTypes.Activatable.ACTIVATION_STATUS, true);
            }
          } else if (date.timeInMillis > lastModDate) {
            // modification date on author is greater then on public
            // we need to set status to activated &amp;amp; also check if lastaction date
            // is before lastmodification date
            activationDate = NodeTypes.Activatable.getLastActivated(node)
            if (!date.after(activationDate)) {
              newDate = date.clone();
              newDate.add(Calendar.HOUR_OF_DAY, -1)
              node.setProperty(NodeTypes.Activatable.LAST_ACTIVATED, newDate);
              node.setProperty(NodeTypes.Activatable.ACTIVATION_STATUS, true);
              println "Set orange activation status on page: " + node.getPath()
            }
          } else {
            // skip
          }
        }
      } else {
        node.setProperty(NodeTypes.Activatable.ACTIVATION_STATUS, false);
        println "Set red activation status on page: " + node.getPath()
      }
    if (i++ % 100 == 0) {
      session.save()
      println "session saved"
    }
  }
}

NodeUtil.visit(session.getNode(rootPath), visitor as NodeVisitor, new NodeTypePredicate("mgnl:page", false))

session.save()
println "session saved"

for (String page : pagePaths) {
    if (!session.nodeExists(page)) {
      println "Page " + page + " is missing on author but is present on public!"
    }
}
```

Make N copies of a Node at the same level (maybe useful for testing things like deleting multiple Nodes):

```
// Copy "/modules/groovy/apps/groovy" 50 times, under "/modules/groovy/apps"

// get the config workspace:
session = MgnlContext.getJCRSession("config");

// get a node:
node = session.getNode("/modules/groovy/apps/groovy");

// copy the node n times:
(50..1).each {
    NodeUtil.copyInSession(node, "/modules/groovy/apps/${it}")
}

session.save();
```

Automatically synchronize all registered workspaces

```
subscriptions = NodeUtil.collectAllChildren((ctx.getJCRSession('config')).getNode('/modules/synchronization
/commands/synchronization/synchronize/subscriber/subscriptions'))

subscriptions.each { node ->
    workspace = (NodeUtil.unwrap(node)).getProperty('repository').getString()

    catalog = (info.magnolia.commands.CommandsManager.getInstance()).getCatalogByName('synchronization')

    command = catalog.getCommand('synchronize')
    command.setRepository(workspace)

    command.setPath('/')
    command.setRecursive(true)

    command.execute(ctx)
}
```

Change all Templates

```
    import info.magnolia.jcr.predicate.NodeTypePredicate;

    session = ctx.getJCRSession('website')

    pages = NodeUtil.collectAllChildren(session.getRootNode(), new NodeTypePredicate("mgnl:page"))

    pages.each { page ->
        node = NodeUtil.unwrap(page)
        template = node.getProperty("mgnl:template").getValue().getString()

       // println template

        newt = template.replace("mtk:pages/basic", "fluffy-bunny-slippers")

        node.setProperty("mgnl:template", newt)

       // printlin newt

    }

    session.save()
```

Export all workspaces to separate files (old skool)

```
// Programmatically Export all your workspaces as XML:
import info.magnolia.repository.RepositoryManager
import info.magnolia.objectfactory.Components
import info.magnolia.importexport.DataTransporter
tmpFilesPath = '/Users/bandersen/Desktop/TEST/'
workspaces = Components.getSingleton(RepositoryManager.class).getWorkspaceNames()
workspaces.each { workspace ->
    hm = ctx.getHierarchyManager(workspace)
    //hm = ctx.getJCRSession(workspace)
    workspaceRoot = hm.getContent('/')
    //workspaceRoot = hm.getRootNode()
    xmlFileOutput = new FileOutputStream(tmpFilesPath + workspace + '.xml')
         println 'Backing up workspace [' + workspace + '] to ' + tmpFilesPath + workspace + '.xml ...'
         DataTransporter.executeExport(xmlFileOutput, false, false, hm.getWorkspace().getSession(),
workspaceRoot.getHandle(), workspace, DataTransporter.XML)
         //DataTransporter.executeExport(xmlFileOutput, false, false, hm, workspaceRoot.getHandle(), workspace,
DataTransporter.XML)
    xmlFileOutput.close()
 }
```

Export all workspaces to separate files (new skool)

```
// Programmatically Export all your workspaces as XML:
import info.magnolia.repository.RepositoryManager
import info.magnolia.objectfactory.Components
import info.magnolia.importexport.DataTransporter
tmpFilesPath = '/Users/bandersen/Desktop/TEST/'
workspaces = Components.getSingleton(RepositoryManager.class).getWorkspaceNames()
workspaces.each { workspace ->
    //hm = ctx.getHierarchyManager(workspace)
    hm = ctx.getJCRSession(workspace)
    //workspaceRoot = hm.getContent('/')
    workspaceRoot = hm.getRootNode()
    xmlFileOutput = new FileOutputStream(tmpFilesPath + workspace + '.xml')
         println 'Backing up workspace [' + workspace + '] to ' + tmpFilesPath + workspace + '.xml ...'
         //DataTransporter.executeExport(xmlFileOutput, false, false, hm.getWorkspace().getSession(),
workspaceRoot.getHandle(), workspace, DataTransporter.XML)
         DataTransporter.executeExport(xmlFileOutput, false, false, hm, workspaceRoot.getHandle(), workspace,
DataTransporter.XML)
    xmlFileOutput.close()
 }
```

Import all exported workspaces

```
// Re-Import all workspaces:
import groovy.io.FileType
import info.magnolia.importexport.DataTransporter
import javax.jcr.ImportUUIDBehavior

def list = []
directory = "/Users/bandersen/Desktop/TEST/"

def dir = new File(directory)
dir.eachFileRecurse (FileType.FILES) { file ->
    workspace = file.name - ~/\.\w+$/   // remove the file extension, e.g., '.xls'

    xmlFile = new File(directory + file.name)
    DataTransporter.importFile(xmlFile, workspace, "/", false, ImportUUIDBehavior.IMPORT_UUID_CREATE_NEW, true,
true)
}
```

Export of embedded templating to ftl files

```
import info.magnolia.jcr.util.NodeUtil;
import info.magnolia.jcr.util.NodeVisitor;
import info.magnolia.context.MgnlContext;
import info.magnolia.init.MagnoliaConfigurationProperties;
import info.magnolia.cms.core.SystemProperty;

import javax.jcr.Session;

import java.io.File;
import java.io.FileWriter;

def loggingVisitor(workspace,path) {

visitor = { node ->
  if (node.hasProperty("mgnl:template")) {
    if (node.getProperty("mgnl:template").getString() == "resources:processedHtml") {
        file = new FileWriter(new File(SystemProperty.getProperty(MagnoliaConfigurationProperties.
MAGNOLIA_APP_ROOTDIR) + "/modules" + node.getPath() + '.ftl'));
        file.write(node.getProperty("text").getString());
        file.close();
        println "resource: ${node.getPath()} exported";
    }
  }
}

Session session = MgnlContext.getJCRSession(workspace);
node = session.getNode(path);

NodeUtil.visit(node,  visitor as NodeVisitor);
}

//example
loggingVisitor("resources", "/");
```

Export DAM files to your server folder

This script helps export all your DAM files to a specific directory from your server side.

Note that you need to create a writable folder (+w) from your server side and put its absolute path like this "/Users/vietnguyen/Downloads/dam-files/" into the script below.

Export DAM files server side

```
javax.jcr.NodeIterator nodeList =  info.magnolia.cms.util.QueryUtil.search("dam", "select * from [mgnl:
resource] as t  where name(t) = 'jcr:content'");
while(nodeList.hasNext()) {
  javax.jcr.Node currNode = nodeList.nextNode();
  println "downloading ${currNode.path}";
  try {
    javax.jcr.Binary binary = currNode.getProperty("jcr:data").getBinary()
    InputStream inputStream = binary.getStream();

    byte[] buffer = new byte[inputStream.available()];
    inputStream.read(buffer);
    String fileName = info.magnolia.jcr.util.PropertyUtil.getString(currNode, "fileName", "noname");
    String extension = info.magnolia.jcr.util.PropertyUtil.getString(currNode, "extension", ".bin");
    String downloadFileName = fileName.endsWith(extension) ? fileName : fileName + "." + extension;

    File targetFile = new File("/Users/vietnguyen/Downloads/dam-files/" + downloadFileName);
    OutputStream outStream = new FileOutputStream(targetFile);
    outStream.write(buffer);
    println("File Downloaded in " + targetFile.getAbsolutePath());
  } catch (Exception x) {
      println("Error:" + x);
  }
}
```

Manually run Jackrabbit Garbage Collection

> The data store never deletes entries except when running data store garbage collection. Similar to Java heap garbage collection, data store garbage collection will first mark all used entries, and later remove unused items.
>
> Data store garbage collection does not delete entries if the identifier is still in the Java heap memory. To delete as many unreferenced entries as possible, call System.gc() a few times before running the data store garbage collection. Please note System.gc() does not guarantee all objects are garbage collected.
>
> -- *Apache Jackrabbit DataStore Garbage Collection Introduction*

This would useful for customers who used Magnolia CMS for such a long time. In the snipped of code below, I am using "dam" as the highest user of Jackrabbit DataStore. Other workspaces and repositories seem not to be necessary.

```
org.apache.jackrabbit.core.RepositoryImpl rpi = info.magnolia.context.MgnlContext.getJCRSession("dam").
getRepository();
org.apache.jackrabbit.api.management.DataStoreGarbageCollector gc = rpi.createDataStoreGarbageCollector();
gc.mark();
gc.sweep();
gc.close();
```

Update a system property

Did you accidentally go into production with `magnolia.develop` set true? Using this script you can set any System property at runtime until you get a chance to restart later.

```
import info.magnolia.init.MagnoliaConfigurationProperties
import info.magnolia.objectfactory.Components
import java.lang.System

propertyToBeChanged = "magnolia.develop"
// propertyToBeChanged = "magnolia.resources.filesystem.observation.excludedDirectories"

value = "false"
// value = "META-INF,WEB-INF,cache,docroot,logs,repositories,tmp"

props = Components.getComponent(info.magnolia.init.MagnoliaConfigurationProperties)
str = props.getProperty(propertyToBeChanged)

println "Changing " + propertyToBeChanged + " from " + str

System.setProperty(propertyToBeChanged, value)
str = props.getProperty(propertyToBeChanged)

println "Property " + propertyToBeChanged + " changed to " + str
```

Changes can be verified from the Config info subapp in the About app.

Alphabetically order a group of sibling nodes

Here is an example of how to order the modules in the config workspace.

```
workspace = "config";
path = "modules";

session = ctx.getJCRSession(workspace);
node = session.getRootNode().getNode(path);
outOfOrder = true;
iterator = null;
previous = null;
current = null;


while (outOfOrder) {
    outOfOrder = false;
    iterator = node.getNodes();

    if (iterator.hasNext()) {
        previous = iterator.nextNode();

        while (iterator.hasNext()) {
            current = iterator.nextNode();

            if (current.getName().compareToIgnoreCase(previous.getName()) < 0) {
                node.orderBefore(current.getName(), previous.getName());
                outOfOrder = true;
            }

            previous = current;
        }
    }
 }

session.save();
```

List component mappings

Provide a list of component mappings.

```
import info.magnolia.objectfactory.Components

components = Components.componentProvider.getTypeMappings()

components.keySet().each { key ->
    println "type: " + key
    println "implementation: " + components.get(key)
    println ''
}
```

Send a mail

Example of how to send a mail using a template.

```
import info.magnolia.module.mail.*;
import info.magnolia.module.mail.templates.*;
import info.magnolia.objectfactory.*;
import java.util.*;

Map<String, Object> mailParameters = new HashMap<String, Object>();
MailModule mailModule = Components.getComponent(MailModule.class);
MgnlMailFactory mgnlMailFactory = mailModule.getFactory();
MgnlEmail email = mgnlMailFactory.getEmailFromTemplate("testFreemarker", mailParameters)
email.setToList("mgnl@mail.ch");
email.setBodyFromResourceFile();
mailModule.getHandler().sendMail(email);
```

Remove individual version of a node

```
versionMan = info.magnolia.objectfactory.Components.getComponent(info.magnolia.cms.core.version.VersionManager.
class);

node = ctx.getJCRSession("workspace").getNode("nodePath");
verIt = versionMan.getAllVersions(node);

// Optionally list names and UUIDs for all versions
verIt.each({version ->
  println("VERSION NAME: " + version.getName() + " VERSION ID: " + version.getIdentifier());
})

verHis = versionMan.getVersionHistory(node);

verHis.removeVersion("versionName");
```