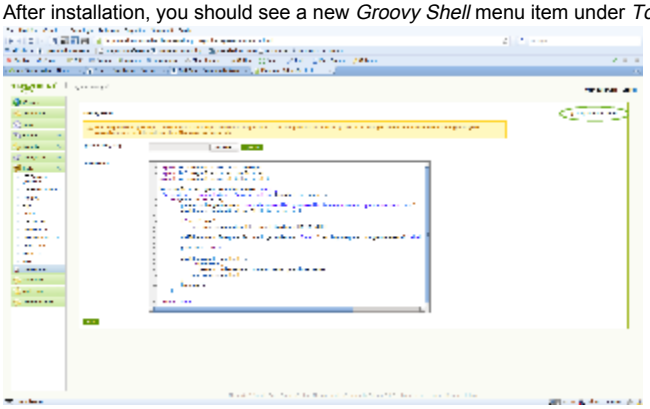
Magnolia groovy shell module

Your Rating: Results: 195 rates

This is a brief description of the recently released openutils-mgnlgroovy module.

openutils-mgnlgroovy is an open source custom Magnolia module which adds a console for running groovy scripts inside the Magnolia adminCentral. Nothing fancy, but a very handy tool, which allows for leveraging all the power of Groovy+Java within Magnolia for all kinds of administrative tasks, without the need for precompiling and deploying any Java class.

After installation, you should see a new *Groovy Shell* menu item under *Tools*, like in the picture below



You can upload your scripts or write them directly in the editor. In both cases you get 'real-time' syntax highlighting and line numbers which can be very useful to spot problematic instructions in case of errors during script compiling or execution.

In the upper right corner of the module you can see the groovy version detected in the classpath at module bootstrapping.

As an example of what you can do with Groovy+ Magnolia here is a small script to lexicographically sort some nodes in the jcr repository based on one of their properties.

More scripts can be found under Groovy Shell Scripts

```
import info.magnolia.cms.core.ItemType
import info.magnolia.context.MgnlContext
import info.magnolia.cms.util.ContentUtil

hm = MgnlContext.getHierarchyManager('config')
['promotion', 'marketplace','communication'].each { category ->
    for(counter in 1..3) { counter->
        parent = hm.getContent("/modules/dms/dialogs/dmsEdit/$category/country$counter/options")
        children = ContentUtil.getAllChildren(parent)

        if (children)
            children = parent.getChildren(ItemType.CONTENTNODE)

        children.sort([compare:{a,b-> b.getNodeData('label').string.compareTo(a.getNodeData('label').string) }
] as Comparator)

        previous = null

        children.each{ content ->
            if(previous)
               parent.orderBefore(content.name, previous.name)
            previous = content
        }
        hm.save()
    }
}
return 'done'
```

Notice how the dynamic nature of Groovy and the syntactic sugar it adds to plain Java allows for a very compact, clean and readable code. This is just a trivial example but the facilities Groovy provides to easily carry out all sorts of tasks, from interacting with relational databases, to read and write xml, spreadsheets etc makes it, in my experience, an indispensable aid to working with Magnolia.