Easy JCR File streaming with Magnolia CMS

Recently Magnolia Partners and Customers who started using our REST content delivery endpoints wondering how could they 'stream' their files which are stored / uploaded into our Jackrabbit repository. This kind of request has increased due to more and more people are developing single page apps with Magnolia CMS as a backend management system (Hybrid Headless). Even though we are providing a DAM (Digital Asset Manager) Download Servlet in our DAM Core module, it is not enough when people start also working with different content types from different workspaces - provided by our Content Types module.

This guideline provide you with an easy, quick, temporarily solution for file streaming requirement when waiting for a stable and well tested out of the box supported function officially delivered by Magnolia CMS.

Let's start designing our use-case

As a web page developer, you are working quite well with our provided Node response from *REST content delivery endpoints* however still missing 'something' when you just want to stream **just** the content of your uploaded file. Because the content of the file is large (a few MB image for instance) and you don't want to parse the whole JSON response with lots of redundancy data from the containing node to get file content from your front-end side. If the content of the file is pretty small so no one care about the parsing or resources using when doing that.

Then from your point of view, you will need to get the file content when you know exactly where it is. Also you would expect that requests to file streaming would be specific so that people will not accidentally stream a large number of files from your server. Then the function API is quite simple that we'll get the prefix of the request somehow like our ".rest" one, let's take ".file" in this example. Then workspace as the physical storage place where your file is locating within the big Jackrabbit repository, and then its path.

Proposed request pattern

```
http(s)://host:port/contextPath/.file/your_workspace_name_such_as_contacts/parent_node(s)
/your_node_that_containing_the_file
```

An example

```
http://localhost:8580/mgnltc8-webapp/.file/contacts/vvangogh/photo
```

Where

- my host/port/context would not be so strange to any of this article reader.
- ".file" is our 'randomly' picked for our file streaming purpose
- "contacts" is our demo workspace which is available to most of customers
- "vvangogh" is a node name of our demo data for Vincent Van Gogh
- "photo" is the node that contain the image file which is hidden from customers point of view if you are using our Contacts app, however it's quite easy to discover if you're using our JCR Browser app and look into "contacts" workspace.

Our expected result

- Would be a streaming session starts when the request come and you get the full image after all. Let's say this cUrl request would end up with an image to your current folder:

```
curl -u username:password http(s)://host:port/contextPath/.file/your_workspace_name_such_as_contacts/parent_node
(s)/your_node_that_containing_the_file --output filename.extension
```

- An example is:

```
curl -u superuser:superuser http://localhost:8580/mgnltc8-webapp/.file/contacts/vvangogh/photo --output vangogh.
jpg
```

- Or directly put the request to your browser will let you download the image file with name and extension correctly.

Then absolutely we need a Servlet class

The bottom line is you need something really stream the file for you, and it is a Java servlet class as below:

FileDownloadServlet

```
package info.magnolia.cms.servlets;

import java.io.IOException;

import javax.jcr.Node;
import javax.jcr.Property;
import javax.jcr.RepositoryException;
import javax.jcr.Session;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.jackrabbit.commons.JcrUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import info.magnolia.cms.filters.ContextFilter;
import info.magnolia.cms.util.ServletUtil;
import info.magnolia.context.MgnlContext;
import info.magnolia.jcr.util.PropertyUtil;

public class FileDownloadServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    public static Logger log = LoggerFactory.getLogger(ContextFilter.class);

    private String contentDisposition = "attachment";

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        doGet(request, response);
    }

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException, ServletException {
        String path = ServletUtil.stripPathParameters(request.getPathInfo());
        if (path == null) {
            log.error("Cannot stream file from an empty path.");
            throw new ServletException("Cannot stream file from an empty path.");
        }
        try {
            log.info("Streaming {}...", path);
            path = path.substring(1); // remove first slash
            int pos = path.indexOf("/");
            String workspaceName = path.substring(0, pos);
            Session session = MgnlContext.getJCRSession(workspaceName);
            Node node = session.getNode(path.substring(pos));
            if (node.hasNode(Node.JCR_CONTENT)) {
                node = node.getNode(Node.JCR_CONTENT);
            }
            response.setContentType(PropertyUtil.getString(node, Property.JCR_MIMETYPE, "application/octet-
```

```
stream"));
            String fileName = PropertyUtil.getString(node, "fileName", "noname");
            String extension = PropertyUtil.getString(node, "extension", ".bin");
            String downloadFileName = fileName.endsWith(extension) ? fileName : fileName + "." + extension;
            response.setHeader("Content-Disposition", contentDisposition + "; filename=\"" + downloadFileName +
"\"");
            JcrUtils.readFile(node, response.getOutputStream());
            log.debug("Stream file {} successful!", node.getPath());
        } catch (RepositoryException e) {
            String errorMessage = "Cannot stream file from path: " + path;
            log.error(errorMessage);
            throw new ServletException(errorMessage, e);
        }
    }

    public String getContentDisposition() {
        return contentDisposition;
    }

    public void setContentDisposition(String contentDisposition) {
        this.contentDisposition = contentDisposition;
    }

}
```

-- File FileDownloadServlet.java

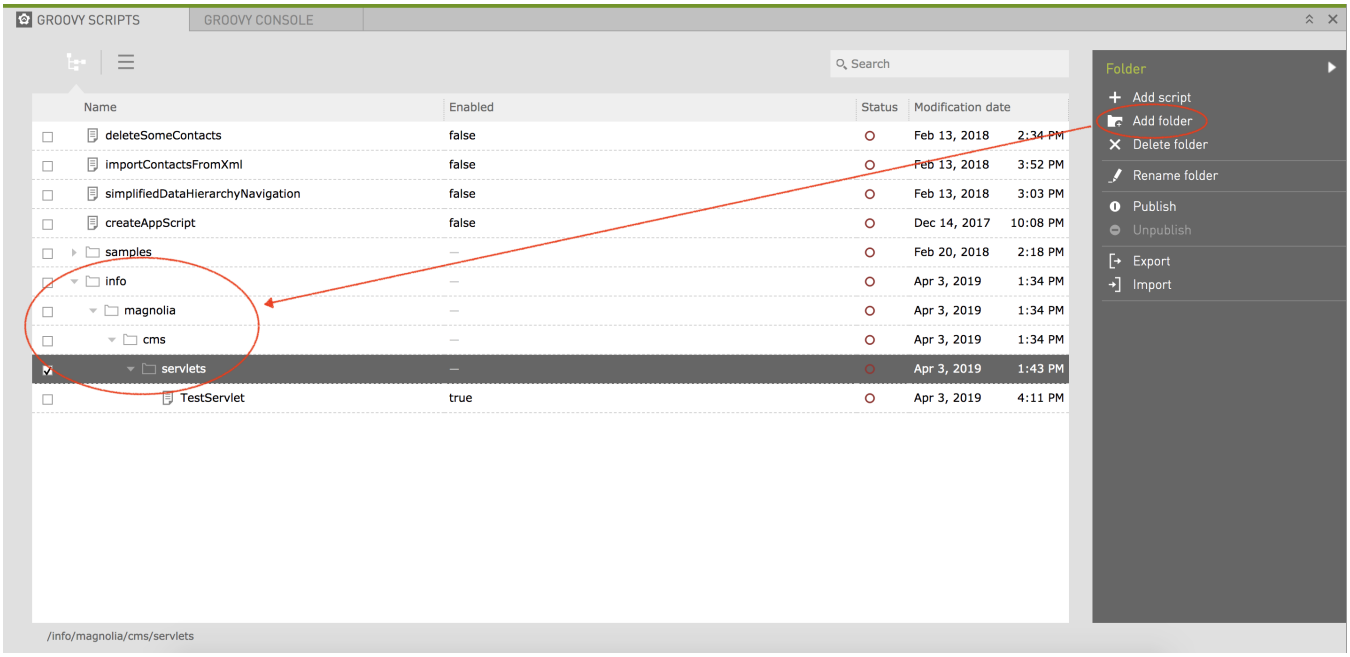
Deploying FileDownloadServlet to your container

Note that you need to choose either Light development deployment way or Custom Magnolia Maven module (Java development) deployment way.

Light development deployment way

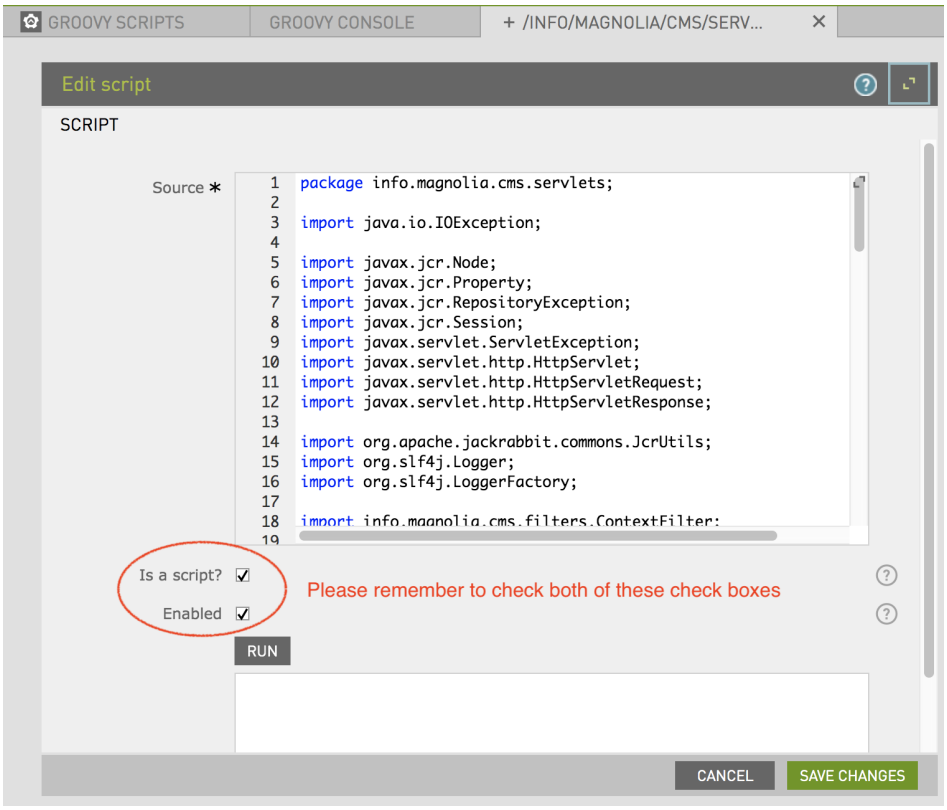
1. Open Groovy App under our DEV drawer in admincentral and create corresponding folder structure for our Groovy Class as below:

The folder is "/info/magnolia/cms/servlets" corresponding to above JAVA source code, package definition "info.magnolia.cms.servlets".



2. Then Click on "Add script" to open next screen, copy and paste all our JAVA code above to the new Code Editor screen as below:

Note that you need to check on both "Is a script?" and "Enabled" check boxes



3. Save changes then you have your class loaded by our Groovy Class Loader using Light development deployment way.

Custom Magnolia Maven module deployment way

Create corresponding class in your developing module then deploy the module to your developing webapp. If you are new to Magnolia Maven module development, please follow Setting up a Magnolia website project and Adding a custom module to your website project written by Richard Gange to getting start with this.

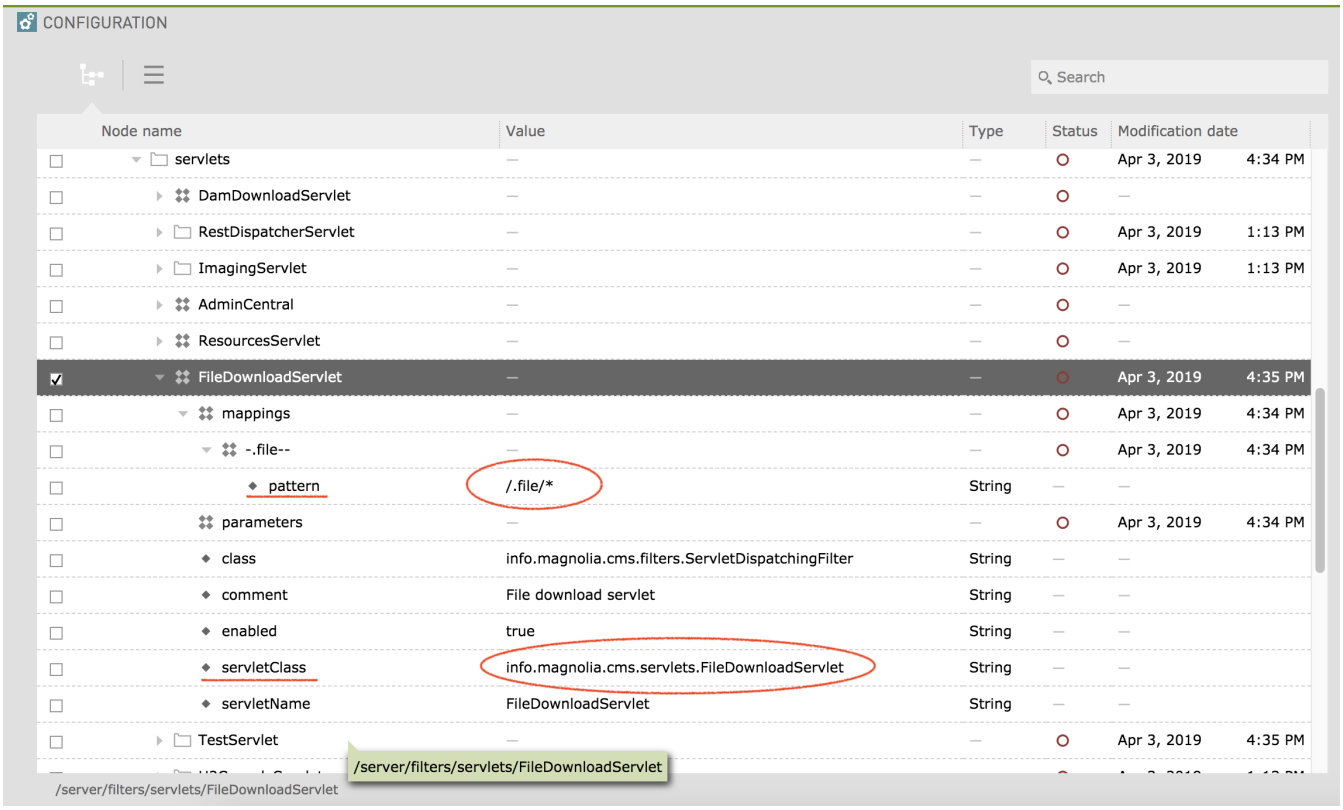
Configure your servlet

Since you have got the servlet and it is loaded in your classpath, then you can follow our guideline Configure the servlet in the filter chain to make it up and run.

Here is the detail:

- Your configuration path is /server/filters/servlets/FileDownloadServlet - note that the node "FileDownloadServlet" has not existed yet, please create one or just import this file into (under) /server/filters/servlets node config.server.filters.servlets.FileDownloadServlet.yaml
- Your servlet mapping pattern should be "/.file/*" as our defined from beginning of this article
- And finally your servletClass should be the class that we just implemented.

See below for double checking:



Finally please verify the result by accessing our proposed URL or delivering cURL commands with username and password included to see if the file is streamed correctly for you.

Please note that you always able to customize the ACL (access control list) for your workspaces, nodes, subnodes, etc. for each of your role using our Security app - Roles and ACL.

Hope this helps and have a good day!