

Uploading Files with Groovy

Uploading Files with Groovy

See [Uploading Files to ScienceBase](#) for more detailed information on uploading files

In this example we use `httpClient` and `httpmime` from `HTTPComponents` to login and upload items to ScienceBase. We are doing this from a stand alone groovy script, so we just "grab" the libraries directly from their source at the beginning of the script. If you are planning on sending files from a Grails application it is recommended that you download them and put them in your libraries.

HTTPComponents: <http://hc.apache.org/downloads.cgi>

Login and Obtain a JOSSO_SESSIONID

Before you can begin to upload files you must log in and get a `JOSSO_SESSIONID`. This is covered here:

Uploading files example

In this script we log in and obtain a `JOSSO_SESSIONID`, create a list of files, create some a `sbJSON` string, and send the files.



It is important that we check for a 200 response before we try to parse any JSON or we will more than likely throw an exception.

```
package gov.usgs

@Grab(group='org.apache.httpcomponents', module='httpClient', version='4.2.1')
@Grab(group='org.apache.httpcomponents', module='httpmime', version='4.2.1')

import org.apache.http.client.*
import org.apache.http.impl.client.DefaultHttpClient
import org.apache.http.client.methods.*
import groovy.json.JsonSlurper
import org.apache.http.entity.mime.*
import org.apache.http.entity.mime.content.*
import org.apache.http.util.EntityUtils
import org.apache.http.cookie.Cookie

//ScienceBase URL
def baseUrl = "https://www.sciencebase.gov/catalog"
//ScienceBase beta URL
//def baseUrl = "https://beta.sciencebase.gov/catalog"

//Username and password, these should be a service account. Usernames must be all lowercase, check logging in to
//http://my.usgs.gov with the exact username and password if logging in fails.
def username = "sbdocumentationtesting"
def password = "k+I7!iAIZ^4"

//Login
String josso_sessionid = login(baseUrl, username, password)

//Check that we logged in correctly
if(!josso_sessionid) {println "fail"; System.exit(1)}

//Create a list of files to send, this list has a few instances of the same file (the current script)
List<File> files = new ArrayList<File>();
files.add(new File("uploadFileExample.groovy"));
files.add(new File("uploadFileExample.groovy"));

//The id to our File Upload Tests item
//https://www.sciencebase.gov/catalog/item/5033ff1ae4b068b9cdc54853
def id = "5033ff1ae4b068b9cdc54853"

def sbJSON = "{title: 'File Upload Test', provenance:{html: 'File upload from example in the ScienceBase Documentation'}}"
//Send the files
```

```

sendFiles(baseUrl, josso_sessionid, id, files, sbJSON, true)

/**
 * Login to ScienceBase and return a HttpClient
 *
 * @param baseUrl Base URL to authenticate on, eg. https://my.usgs.gov/catalog
 * @param username The plain text username to authenticate on, use your own or a service account
 * @param password The plain text password to authenticate on
 * @return The josso_sessionid from the cookie received when logging in
 */
String login(String baseUrl, String username, String password){
    def encodedUsername = URLEncoder.encode(username)
    def encodedPassword = URLEncoder.encode(password)

    //Create a client for sending files, they will share cookies through this
    HttpClient httpClientLogin = new DefaultHttpClient()
    //Login
    HttpGet httpGet = new HttpGet(("${baseUrl}/?josso_username=${encodedUsername}
&josso_password=${encodedPassword}").toURI())
    httpGet.addHeader("Accept", "application/json")
    //login
    def loginResponse = httpClientLogin.execute(httpGet)
    //Must do this to complete the login and release httpClientLogin
    def loginString = EntityUtils.toString(loginResponse.getEntity())

    def josso_sessionid_cookie = httpClientLogin.getCookieStore().getCookies().find{it.name ==
"JOSSO_SESSIONID"}
    if(loginResponse.statusLine.toString().contains("200") && josso_sessionid_cookie){
        return josso_sessionid_cookie.value
    }
    else return null
}

/**
 * Send files to ScienceBase
 *
 * Sends files to ScienceBase one at a time, can create new items
 *
 * @param baseUrl The base URL you want to send to, eg. https://my.usgs.gov/catalog. This should
be the
 * same url you authenticated on
 * @param josso_sessionid The josso_sessionid that you got from logging in
 * @param id The id that you want to send to OR the parent id of the item you want to create
 * @param files A list of Files that you want to send
 * @param createNewItem True if you want to create a new item below the item id you input, false if you
want to
 * update the item id you input
 * @return A list of JSON responses for each of the items you sent
 */
List<Map> sendFiles(String baseUrl, String josso_sessionid, String id, List<File> files, String sbJSON, Boolean
createNewItem = false){
    def uploadPath
    if(createNewItem){
        uploadPath = "${baseUrl}/file/uploadAndCreateItem/${id}?josso=${josso_sessionid}".toURI()
    } else {
        uploadPath = "${baseUrl}/file/uploadAndUpdateItem/${id}?josso=${josso_sessionid}".toURI()
    }

    //For sending Posts
    HttpPost httpPost = new HttpPost()

    //Create a JjsonSlurper to process the returned strings into json
    def jsonSlurper = new JjsonSlurper()

    //Returned Jjson for each file
    List<Map> jsonUploadResponses = new ArrayList<Map>()

```

```

//Set the URI for our initial upload
httpPost.setURI(uploadPath)
//Add headers to our upload post
httpPost.addHeader("Accept", "application/json")

//Create a multipart entity and add our file and json
def mpEntity = new MultipartEntity()
files.each{ file ->
    assert file.exists()
    mpEntity.addPart("file", new FileBody(file))
}
//The json to go along with the test
mpEntity.addPart("item", new StringBody(sbJSON))
httpPost.setEntity(mpEntity)

//Execute upload
//println "executing request ${httpPost.getRequestLine()}"
HttpClient httpClient = new DefaultHttpClient()
def uploadResponse = httpClient.execute(httpPost)
def httpUploadEntity = uploadResponse.getEntity()

//We can only "consume" httpScrapeEntity once, so let's set it to a string, this will make for easier error
finding at some point
def uploadRespString = EntityUtils.toString(httpUploadEntity)

//Parse our response to JSON
def uploadRespJson
if(uploadResponse.statusLine.toString().contains("200")){
    uploadRespJson = jsonSlurper.parseText(uploadRespString) as Map
    jsonUploadResponses.add(uploadRespJson)
} else {
    //If we don't get a 200 back, throw an error
    println uploadResponse.statusLine.toString()
    throw new Exception(uploadRespString)
}

//Set the id to the new item we created in case we are uploading multiple files
id = uploadRespJson.id

return jsonUploadResponses
}

```