Making POST, PUT, DELETE Calls from Groovy

Making POST, PUT, DELETE Calls from Groovy

POST & DELETE

Here is an example POST and DELETE (and login) using HTTPBuilder. Since it is using Grape there is no need to download the libraries. You can find out more about HTTPBuilder here: http://groovy.codehaus.org/modules/http-builder/

PUTs (updates) can also be easily done by changing the Method to put and using the full url of the item as the path.

```
import groovy.grape.Grape
@Grab(group='org.codehaus.groovy.modules.http-builder', module='http-builder', version='0.5.2')

import groovyx.net.http.*
import groovyx.net.http.ContentType.*
import groovyx.net.http.Method.*
import net.sf.json.*

def scibase
scibase = new HTTPBuilder(https://www.sciencebase.gov/catalog)
scibase.get( query: [josso_username: '<harvester username>', josso_password: '<harvester password>'] )
def newItemId

//Create a json body with all tested fields
def jsonBody = [:]
// Test title
jsonBody.put("title", "Test title")
// Test parentID
jsonBody.put("parentId", "4f5fc39de4b098845cbcb45e")


// Test Subtitle
jsonBody.put("subTitle", "subtitle")
// Test Summary
jsonBody.put("summary", "summary")
// Test Body
jsonBody.put("body", "body")
// Test purpose
jsonBody.put("purpose", "purpose")
// Test rights
jsonBody.put("rights", "rights")
// Test edition
jsonBody.put("edition", "edition")
// Test MaterialRequestInstructions
jsonBody.put("materialRequestInstructions", "materialRequestInstructions")

// Test Provenance
jsonBody.put("provenance", ["html": "provenance",
        "harvesterCode": "harvester code",
        "harvesterSourceCode": "harvester source code",
        "harvesterConfigCode": "harvester config code",
        "harvesterJob":"harvester job"])


// Test alternate titles
def alternateTitles = ["Alternate", "Titles", "For item"]
def altTitlesArray = []
alternateTitles.each{ altTitle ->
        altTitlesArray.add(altTitle.toString())
}
jsonBody.put("alternateTitles", altTitlesArray)

// Test narratives
def narratives = [[type: "type", content: "content"],[type: "type2", content: "content2"]]
def narrativeArray = []
narratives.each{ narrative ->
        narrativeArray.add(["type": narrative.type.toString(), "content": narrative.content.toString()])
}
```

```
jsonBody.put("narratives", narrativeArray)

// Test identifers
def identifiers = [[scheme: "scheme", type: "type", key: "key"],[scheme: "scheme2", type: "type2", key: "key2"]]
def identifierArray = []
identifiers.each{ identifier ->
        identifierArray.add(["scheme": identifier.scheme.toString(),
                                                "type": identifier.type.toString(),
                                                "key": identifier.key.toString()])

}
jsonBody.put("identifiers", identifierArray)

// Test tags
def tags = ["Tag1", "Tag2"]
def tagsArray = []
tags.eachWithIndex{ tag, i ->
        tagsArray.add(["name": tag.toString()])
}
jsonBody.put("tags", tagsArray)

// Test Contacts
def contacts = [[type: "Data Owner", name: "Data Owner"], [type: "Project Leader", name: "Project Leader"]]
def contactsArray = []
contacts.each { contact ->
        contactsArray.add(["type": contact.type, "name": contact.name])
}
jsonBody.put("contacts", contactsArray)

// Test tableOfContents
jsonBody.put("tableOfContents", "table of contents")


/* Put more fields here */



// POST
scibase.request(Method.POST, ContentType.JSON){ req ->
        uri.path = "/catalog/item/"
        uri.query = [format:'json']
        body = jsonBody

        response.success = { resp, json ->
                assert resp.status == 200
                newItemId = json.id
        }
        // not logged in response
        response.'302' = { resp ->
                throw new Exception("Stopping at item POST: uri: " + uri + "\n" +
                        "   You are not logging in properly. Item will not be created.")
        }
        response.failure = { resp, json ->
                throw new Exception("Stopping at item POST: uri: " + uri + "\n" +
                        "   Unknown error trying to create item: ${resp.status}, not creating Item." +
            "\njson = ${json}")
        }
}

println "New item Id = " + newItemId

System.console().readLine "Press enter when ready to delete item"

scibase.request(Method.DELETE){ req ->
        uri.path = "/catalog/item/" + newItemId
        uri.query = [format:'json']

        // Item exists, change POST to PUT
        response.success = { resp, json ->
                println "SUCCESS: Successfully deleted item '${newItemId}'"
        }
```

```
        response.failure = { resp ->
                throw new Exception("FAILURE: Could not delete item '${newItemId}'. Error response: \n${resp.
statusLine}")
        }
}
```