Scriptrunner for Confluence

https://scriptrunner.adaptavist.com/latest/confluence/rest-endpoints.html#_configuration

Gettting Userinfo

This works

```
import com.onresolve.scriptrunner.runner.rest.common.CustomEndpointDelegate
import groovy.json.JsonBuilder
import groovy.transform.BaseScript

import javax.ws.rs.core.MultivaluedMap
import javax.ws.rs.core.Response

import com.atlassian.sal.api.component.ComponentLocator
import bucket.user.UserAccessor

@BaseScript CustomEndpointDelegate delegate

def userAccessor = ComponentLocator.getComponent(UserAccessor)

user(
        httpMethod: "GET", groups: ["confluence-administrators"]
) {

    // validate we have username as a url parameter
    // extraPath is already available to use
    def extraPath = extraPath as String
    assert extraPath =~ "^/[a-zA-Z]+"
    def username = extraPath.split("/").last()

    def user = userAccessor.getUser(username)
    // user must exist in Confluence
    if (!user) {
        return Response.serverError().entity([error: "User $username does not exist"]).build()
    }

    def userResponse = [
            username: user.name,
            fullname: user.fullName,
            email: user.email
    ]

    return Response.ok(new JsonBuilder(userResponse).toString()).build()
}
```

Output:

```
jarvis:bin npn$ curl -X GET -u admin:Welcome1 http://localhost:8090/rest/scriptrunner/latest/custom/user/admin
{"username":"admin","fullname":"admin","email":"npn@netic.dk"}jarvis:bin npn$
jarvis:bin npn$
```

Including getting rights to a Page (hardcoded id)

```
import com.onresolve.scriptrunner.runner.rest.common.CustomEndpointDelegate
import groovy.json.JsonBuilder
import groovy.transform.BaseScript

import javax.ws.rs.core.MultivaluedMap
import javax.ws.rs.core.Response

import com.atlassian.sal.api.component.ComponentLocator
import bucket.user.UserAccessor

import com.atlassian.confluence.pages.PageManager
import com.atlassian.confluence.security.PermissionManager
import com.atlassian.confluence.security.Permission

@BaseScript CustomEndpointDelegate delegate

def userAccessor = ComponentLocator.getComponent(UserAccessor)

user(
        httpMethod: "GET", groups: ["confluence-administrators"]
) {

    // validate we have username as a url parameter
    // extraPath is already available to use
    def extraPath = extraPath as String
    assert extraPath =~ "^/[a-zA-Z]+"
    def username = extraPath.split("/").last()

    def user = userAccessor.getUser(username)
    // user must exist in Confluence
    if (!user) {
        return Response.serverError().entity([error: "User $username does not exist"]).build()
    }

    def pageManager = ComponentLocator.getComponent(PageManager)
    def page = pageManager.getPage(1540098)
        def permissionManager = ComponentLocator.getComponent(PermissionManager)
        // Permissions - https://developer.atlassian.com/confdev/development-resources/confluence-developer-faq
/how-do-i-tell-if-a-user-has-permission-to
    boolean canEdit = permissionManager.hasPermission(user, Permission.EDIT, page);

    def userResponse = [
            username: user.name,
            fullname: user.fullName,
            email: user.email,
                edit: canEdit
    ]

    return Response.ok(new JsonBuilder(userResponse).toString()).build()
}
```

Output:

Notice that bnp=false, but admin=true

```
curl -X GET -u admin:Welcome1 http://localhost:8090/rest/scriptrunner/latest/custom/user/bnp
{"username":"bnp","fullname":"Normann","email":"bnp@mos-eisley.dk","edit":false}
jarvis:bin npn$

curl -X GET -u admin:Welcome1 http://localhost:8090/rest/scriptrunner/latest/custom/user/admin
{"username":"admin","fullname":"admin","email":"npn@netic.dk","edit":true}
jarvis:bin npn$
```

Examine if a Group has Edit Access:

```
import bucket.user.UserAccessor
import com.atlassian.sal.api.component.ComponentLocator
import com.atlassian.user.GroupManager
import com.atlassian.user.impl.DefaultUser
import com.onresolve.scriptrunner.runner.rest.common.CustomEndpointDelegate
import groovy.json.JsonBuilder
import groovy.transform.BaseScript
import org.codehaus.jackson.map.ObjectMapper

import javax.ws.rs.core.MultivaluedMap
import javax.ws.rs.core.Response

import static com.atlassian.user.security.password.Credential.unencrypted

import com.atlassian.sal.api.component.ComponentLocator
import bucket.user.UserAccessor

import com.atlassian.confluence.pages.PageManager
import com.atlassian.confluence.security.PermissionManager
import com.atlassian.confluence.security.Permission
import com.atlassian.confluence.security.ContentPermission
import com.atlassian.confluence.core.ContentPermissionManager
import com.atlassian.confluence.security.ContentPermissionSet

@BaseScript CustomEndpointDelegate delegate

def userAccessor = ComponentLocator.getComponent(UserAccessor)

user(
        httpMethod: "GET", groups: ["confluence-administrators"]
) {

    // validate we have username as a url parameter
    // extraPath is already available to use
    def extraPath = extraPath as String
    assert extraPath =~ "^/[a-zA-Z]+"
    def groupname = extraPath.split("/").last()

    def group = userAccessor.getGroup(groupname)
    // user must exist in Confluence
    if (!group) {
        return Response.serverError().entity([error: "Group $groupname does not exist"]).build()
    }

    def pageManager = ComponentLocator.getComponent(PageManager)
    def page = pageManager.getPage(1540098)
    def permissionManager = ComponentLocator.getComponent(PermissionManager)

    boolean CanEdit=false

    def PermList = page.getContentPermissionSet(ContentPermission.EDIT_PERMISSION)
    // def ContentPermissionManager = ComponentLocator.getComponent(ContentPermissionManager)
    //List<ContentPermissionSet> cpsl = ContentPermissionManager.getContentPermissionSets(page,
ContentPermission.EDIT_PERMISSION);
    //for(int i=0; i<PermList.size(); i++){
          //        System.out.println(PermList(i) + i.toString());
        //}

    for (ContentPermission cp : page.getContentPermissionSet(ContentPermission.EDIT_PERMISSION)) {
                if (cp.toString().contains("groupName=" + groupname)) {
                    System.out.println(cp)
                CanEdit=true
                }
        }



    return Response.ok(CanEdit.toString()).build()
}
```

Output:

Notice that bnp=false, but admin=true

```
curl -X GET -u admin:Welcome1 http://localhost:8090/rest/scriptrunner/latest/custom/user/confluence-
administrators
true
jarvis:bin npn$
```

Remove EDIT Rights for a page:

```
import bucket.user.UserAccessor
import com.atlassian.sal.api.component.ComponentLocator
import com.atlassian.user.GroupManager
import com.atlassian.user.impl.DefaultUser
import com.onresolve.scriptrunner.runner.rest.common.CustomEndpointDelegate
import groovy.json.JsonBuilder
import groovy.transform.BaseScript
import org.codehaus.jackson.map.ObjectMapper

import javax.ws.rs.core.MultivaluedMap
import javax.ws.rs.core.Response

import static com.atlassian.user.security.password.Credential.unencrypted

import com.atlassian.sal.api.component.ComponentLocator
import bucket.user.UserAccessor

import com.atlassian.confluence.pages.PageManager
import com.atlassian.confluence.security.PermissionManager
import com.atlassian.confluence.security.Permission
import com.atlassian.confluence.security.ContentPermission
import com.atlassian.confluence.core.ContentPermissionManager
import com.atlassian.confluence.security.ContentPermissionSet

@BaseScript CustomEndpointDelegate delegate

def userAccessor = ComponentLocator.getComponent(UserAccessor)

user(
        httpMethod: "GET", groups: ["confluence-administrators"]
) {

    // validate we have username as a url parameter
    // extraPath is already available to use
    def extraPath = extraPath as String
    assert extraPath =~ "^/[a-zA-Z]+"
    def groupname = extraPath.split("/").last()

    String PageId="1540098"

    def group = userAccessor.getGroup(groupname)
    // user must exist in Confluence
    if (!group) {
        System.out.println("Group: " + groupname + " was not found in the System.")
        return Response.serverError().entity([error: "Group $groupname does not exist"]).build()
    }

    def pageManager = ComponentLocator.getComponent(PageManager)
    def page = pageManager.getPage(PageId.toLong())
    if (!page) {
        System.out.println("Page: " + PageId + " was not found in the System.")
        return Response.serverError().entity([error: "Page $PageId does not exist"]).build()
```

```
    }

    def permissionManager = ComponentLocator.getComponent(PermissionManager)
    def contentPermissionManager = ComponentLocator.getComponent(ContentPermissionManager)

    boolean CanEdit=false

    // cp is a ContentpermissionSet - https://docs.atlassian.com/confluence/5.8.4/com/atlassian/confluence
/security/ContentPermissionSet.html (unique and also bound to a page)
    for (ContentPermission cp : page.getContentPermissionSet(ContentPermission.EDIT_PERMISSION)) {
                if (cp.toString().contains("groupName=" + groupname)) {
                // Remove the Permission
                System.out.println("Group: " + groupname + " was found having EDIT Permissions for Page Id: " +
PageId + ".")
                contentPermissionManager.removeContentPermission(cp);
                System.out.println("Removed EDIT permissions for Group: " + groupname + "for Page Id: " +
PageId + ".")
              }
        }



    return Response.ok(CanEdit.toString()).build()
}
```