Automatically Remove Issues Based on JQL Query

The following script will automatically review all issues in a manually built structure and remove any that do not satisfy the defined JQL query.

```
package examples.docs.structure

import com.atlassian.query.Query
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.row.StructureRow
import com.almworks.jira.structure.api.row.RowManager
import com.almworks.jira.structure.api.item.ItemIdentity
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.util.JiraComponents
import com.almworks.integers.LongArray
import com.almworks.integers.LongIterator
import com.almworks.integers.LongOpenHashSet

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version = '16.10.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
StructureComponents structureComponents

def plugin = com.atlassian.jira.component.ComponentAccessor.pluginAccessor.getPlugin('com.almworks.jira.
structure')

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def permission = PermissionLevel.valueOf("ADMIN")



// Here we are going to get our structure and then get the forest that it is built on
def structureName = "name1"
if (structureManager.getStructuresByName(structureName, permission).isEmpty()){
    log.warn "Something went wrong, couldn't find the structure."
 return
}
def struct = structureManager.getStructuresByName(structureName, permission)[0]
def forestSpec = ForestSpec.structure(struct.getId())
```

```
def forestSrc = forestService.getForestSource(forestSpec)

RowManager rowManager = structureComponents.getRowManager()

def forest = forestSrc.getLatest().getForest()


// This will allow us access to useful helper functions, in this case the application of our JQL query to our rows.

def helper = plugin.getModuleDescriptor('helper').module


// This variable will hold all our issues that match our JQL query.

def matchingIssues = new LongOpenHashSet()

// This variable will store all the elements in our structure that are issues (vs folders or generators).

LongArray onlyIssues = new LongArray()

// This variable will hold the rows that that are eventually removed from the Structure.

LongArray matchingRows = new LongArray();


// This will turn our JQL string into a Jira query.

def jqlQuery = "assignee = Eve"

Query query = JiraComponents.getComponent(com.atlassian.jira.jql.parser.JqlQueryParser).parseQuery(jqlQuery);




// Here we are iterating over all the rows of our structure to get the issues.

for (LongIterator ri : forest.getRows()) {

    StructureRow row = rowManager.getRow(ri.value())

    ItemIdentity itemId = row.getItemId()

    if (CoreIdentities.isIssue(itemId)) {

        onlyIssues.add(itemId.getLongId())

    }

}


// Here we are evaluating which items match the query. If we change the boolean value to false, we get the opposite.

helper.matchIssues(onlyIssues, query, true, matchingIssues);


// Now we are iterating over our structure again, row by row, to translate the issues that we want to remove to their respective rows.

for (LongIterator ri : forest.getRows()) {

    StructureRow row = rowManager.getRow(ri.value())

    ItemIdentity itemId = row.getItemId()

    if (CoreIdentities.isIssue(itemId) && matchingIssues.contains(itemId.getLongId())) {

        matchingRows.add(ri.value())

    }

}


// Here we pass the rows we identified as unwanted to our remove function.

forestSrc.apply(new ForestAction.Remove(matchingRows.subList(0,matchingRows.size())))
```