Creating Generators

Below are a series of scripts that run through the creation of an Insert, Extend, Sort and Filter Generator.

JQL Inserter

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreGeneratorParameters
import java.util.Map

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureName = "test" // name of the structure you want to add this generator to
def jql = 'project = stmb and type = story' // this is the JQL query you want the generator to executre

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def generatorManager = structureComponents.getGeneratorManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureId = structureManager.getStructuresByName(structureName, permission)[0].getId()

// inserter
def params = new java.util.HashMap()
params.put(CoreGeneratorParameters.JQL, jql)
def jqlinserterId = generatorManager.createGenerator("com.almworks.jira.structure:inserter-jql", params,
structureId)
def generatorItem = CoreIdentities.generator(jqlinserterId); // item to add to forest
def forestSource = forestService.getForestSource(ForestSpec.structure(structureId)) //resolving forest source for
structure
forestSource.apply(new ForestAction.Add(generatorItem, 0, 0, 0))
```

Extend stories under epics and extend by blocking link

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.item.ItemForestBuilderImpl;
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreStructureGenerators
import com.atlassian.jira.issue.link.IssueLinkTypeManager
import com.almworks.jira.structure.api.generator.CoreStructureGenerators
import java.util.Map

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureName = "test"
def structureId = structureManager.getStructuresByName(structureName, permission)[0].getId()
def forestBuilder = new ItemForestBuilderImpl()
def generatorManager = structureComponents.getGeneratorManager()

def epicsExtenderItem = generatorManager.createGenerator(CoreStructureGenerators.EXTENDER_AGILE, [:], null)
forestBuilder.nextRow(CoreIdentities.generator(epicsExtenderItem))

def issueLinkTypeManager = ComponentAccessor.getComponent(IssueLinkTypeManager)
def blocksLinkTypes = issueLinkTypeManager.getIssueLinkTypesByName('Blocks')
def blocksLinkTypeId = blocksLinkTypes[0].id

def params = [
        'linkTypeId': blocksLinkTypeId, // id of the Blocks issue link type
 'direction': 'outward',         // direction of the link type, could be either 'inward' or 'outward'
 'disableActions': false,        // enable structure actions, i.e. issues DnD
 'from': 2,
        'to': 2 // number of levels to extend issue at
]
def blocksExtenderItem = generatorManager.createGenerator(CoreStructureGenerators.EXTENDER_LINKS, params,
structureId)
forestBuilder.nextRow(CoreIdentities.generator(blocksExtenderItem))

def forestService = structureComponents.getForestService()
def forestSource = forestService.getForestSource(ForestSpec.structure(structureId))
def forestToAdd = forestBuilder.build()
forestSource.apply(new ForestAction.Add(forestToAdd, 0, 0, 0))
```

Sorting by a formula result and manual sorter

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.item.ItemForestBuilderImpl;
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreStructureGenerators

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureName = "test"
def structureId = structureManager.getStructuresByName(structureName, permission)[0].getId()
def forestBuilder = new ItemForestBuilderImpl()
def generatorManager = structureComponents.getGeneratorManager()

def manualSorterItem = generatorManager.createGenerator(CoreStructureGenerators.SORTER_MANUAL, [:], null)
forestBuilder.nextRow(CoreIdentities.generator(manualSorterItem))
/* adding formula-based sorter
*/
def formulaSorterParams = [
        attribute: [
                id : "expr",
                format: "order",
                params : [
                        formula: "assignee + key",
                        variables: [
                                assignee: [ id: "Assignee", format: "text"],
                                key: [ id: "Key", format: "text"]
                        ]
                ]
        ]
]
def formulaSorterItem = generatorManager.createGenerator(CoreStructureGenerators.SORTER_ATTRIBUTE,
formulaSorterParams as Map, null)
forestBuilder.nextRow(CoreIdentities.generator(formulaSorterItem))
def forestService = structureComponents.getForestService()
def forestSource = forestService.getForestSource(ForestSpec.structure(structureId))
def forestToAdd = forestBuilder.build()
forestSource.apply(new ForestAction.Add(forestToAdd, 0, 0, 0))
```

Below adds a JQL filter to a structure

```
package examples.docs.structure

// Structure imports
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.generator.CoreGeneratorParameters
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents

// Scriptrunner imports
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin

// Atlassian import (might be available without the import in some instances, but better safe than sorry)
import com.atlassian.jira.component.ComponentAccessor

// A Hashmap
import java.util.Map

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version = '16.9.0')

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

@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def generatorManager = structureComponents.getGeneratorManager()

// For brevity's sake we will have a permission variable that we pass when we make an admin level request
def permission = PermissionLevel.valueOf("ADMIN")

// the false flag is optional, it is a variable whether we want to search archived structures too
// we are getting the first (zeroth) element because this method returns a list, be mindful if you have multiple
structures with the same name!
def currentStructure = structureManager.getStructuresByName("testScript", permission, false)[0]


def jqltext = "assignee=admin";
Map<String, Object> jqlparams = new java.util.HashMap()
jqlparams.put(CoreGeneratorParameters.JQL, jqltext)

// We have now added our JQL text to the filter (creating a different kind of inserter works almost exactly the
same way, just build up the params variable differently)
def jqlfilterId = generatorManager.createGenerator("com.almworks.jira.structure:filter-jql", jqlparams,
currentStructure.getId())

def generatorItem = CoreIdentities.generator(jqlfilterId); // item to add to forest
def forestSource = forestService.getForestSource(ForestSpec.structure(currentStructure.getId())) //resolving
forest source for structure
forestSource.apply(new ForestAction.Add(generatorItem, 0, 0, 0))
```