

Search Jira fields for a Key=Value entry

I am always missing a Custom Field that can handle Key=Value entries; combined with multiple entries, so I use a multiline Text Field:

```
Service Level=AppDrift
CustomerNumber=1043
Othername=Myhost
```

Lets call the field "VC Tags". Issues with a Key=Value can be found with a JQL like:

```
"VC Tags"~"Service Level=AppDrift"
```

But is quite volatile and the fuzzy search ~ will also find:

```
Service Level=AppDrift+
CustomerNumber=1043
Othername=Myhost
```

So, this JQL Function (For Adaptavist ScriptRunner) can cope with that:



This is working sample code, there is room for better Error Handling 😊

```
package com.onresolve.jira.groovy.jql

import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.Issue
import com.atlassian.query.clause.TerminalClause
import com.atlassian.jira.jql.query.QueryCreationContext
import org.apache.lucene.index.Term
import com.onresolve.jira.groovy.jql.AbstractScriptedJqlFunction
import org.apache.lucene.search.BooleanClause
import org.apache.lucene.search.BooleanQuery
import org.apache.lucene.search.Query
import org.apache.lucene.search.TermQuery
import com.atlassian.jira.issue.fields.CustomField
import com.atlassian.jira.issue.CustomFieldManager
import com.atlassian.jira.user.ApplicationUser
import com.atlassian.jira.security.JiraAuthenticationContext
import com.atlassian.query.operand.FunctionOperand
import com.atlassian.jira.issue.Issue

class keyValueInField extends AbstractScriptedJqlFunction implements JqlQueryFunction {

    @Override
    String getDescription() {
        "Function to find a key value pair in a Custom Field"
    }

    @Override
    List<Map> getArguments() {
        [
            [ "subquery": "Subquery", "optional": false],
            [ "fieldname": "Custom Fieldname", "optional": false],
            [ "key": "", "The Key to find": false],
            [ "value": "", "The value to check for in the key": false],
            [ "usecontains": "", "Use contains instead of equal": false]
        ]
    }
}
```

```

@Override
String getFunctionName() {
    "keyValueInField"
}

@Override
Query getQuery(QueryCreationContext queryCreationContext, FunctionOperand operand, TerminalClause
terminalClause) {

    CustomFieldManager customFieldManager = ComponentAccessor.getCustomFieldManager()

    final JiraAuthenticationContext context = ComponentAccessor.getJiraAuthenticationContext()
    final ApplicationUser applicationUser = context.getLoggedInUser()
    final BooleanQueryBuilder boolQueryBuilder = new BooleanQueryBuilder()
    String jql = operand.args[0]
    String fieldname = operand.args[1]
    String key = operand.args[2].toLowerCase()
    String value = operand.args[3].toLowerCase()
    String usecontains = operand.args[4].toLowerCase()

    issues = getIssues(jql, applicationUser)
    issues.each { Issue issue ->

        //Get Field
        def field = customFieldManager.getCustomFieldObjectsByName(fieldname)[0]
        if (field)
        {
            //Get the Value
            def fieldValue = issue.getCustomFieldValue(field)?:""
            if (fieldValue)
            {
                //Loop each Value
                fieldValue.toString().split("\n").each { theLine ->
                    theLine = theLine.toLowerCase()
                    if (usecontains == "1" || usecontains == "true")
                    {
                        if (theLine.split("=")[0] == key && theLine.split("=")[1].contains(value))
                        {
                            boolQueryBuilder.add(new TermQuery(new Term("issue_id", issue.id as String)),
BooleanClause.Occur.SHOULD)
                        }
                    }
                    else
                    {
                        if (theLine.split("=")[0] == key && theLine.split("=")[1] == value)
                        {
                            boolQueryBuilder.add(new TermQuery(new Term("issue_id", issue.id as String)),
BooleanClause.Occur.SHOULD)
                        }
                    }
                }
            }
        }
    }
    return boolQueryBuilder.build()
}
}

```

Usage:

```
Issuefunction in keyValueInField("Project=TEST","VC Tags","Service Level","AppDrift",0)
```

for an equal search (0). For a contains (fuzzy ~) search:

```
ssuefunction in keyValueInField("Project=TEST and IssueType=Task","VC Tags","Service Level","AppDrift",1)
```