JIRA JQL Function: InStr

As I have been quite annoyed with JIRAs JQL Capabilities, heres an JQL Search function (Requires Scriptrunner):

It gives ability to do searches like "*peter*" in known fields..

Usage:

issueFunction in inStr(JQL,Fieldname,String_to_search_for) - eg:

```
issueFunction in inStr("Project=TEST","Single Line Text","Phasellus")
```

This will find all issues in TEST, where "phasellus" is in the Customfield named "Single Line Text".
Its search is case insensitive...

```
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

class inStr extends AbstractScriptedJqlFunction implements JqlQueryFunction {

    @Override
    String getDescription() {
        "Function to search for a (sub)string in Summary, Description or a Customfield"
    }

    @Override
    List<Map> getArguments() {
        [
                [ "description": "Subquery", "optional": false],
                [ "fieldname": "Custofield name", "optional": false],
                [ "searchString": "Search string", "optional": false]
        ]
    }

    @Override
    String getFunctionName() {
        "inStr"
    }

    def getFieldValue(theIssue,fieldName)
    {
        CustomFieldManager customFieldManager = ComponentAccessor.getCustomFieldManager()

        if (fieldName.toLowerCase() == "summary")
            return theIssue.getSummary()?:"".toLowerCase()

        if (fieldName.toLowerCase() == "description")
            return theIssue.getDescription()?:"".toLowerCase()

        def theFields = customFieldManager.getCustomFieldObjectsByName(fieldName)
```

```
        if (theFields)
        {
            CustomField theField = theFields[0]
            return theIssue.getCustomFieldValue(theField)?:""
        }

        return ""
    }

    @Override
    Query getQuery(QueryCreationContext queryCreationContext, FunctionOperand operand, TerminalClause
terminalClause) {
        final JiraAuthenticationContext context = ComponentAccessor.getJiraAuthenticationContext()
        final ApplicationUser applicationUser = context.getLoggedInUser()
        final BooleanQuery.Builder boolQueryBuilder = new BooleanQuery.Builder()
        String theFieldValue = ""
        String jql = operand.args[0]
        String fieldName = operand.args[1]
        String searchString = operand.args[2].toLowerCase()
        String theFinalValue = ""

        if (fieldName != "" && searchString != "")
        {
            issues = getIssues(jql, applicationUser)
            issues.each { Issue issue ->

                //Compare with lower cases
                def theValue =  getFieldValue(issue, fieldName)
                if(theValue.class.isArray())
                {
                    theFieldValue.each { avalue ->
                        theFinalValue = theFinalValue + avalue.value.toString()
                    }
                }
                else
                    theFinalValue = theValue.toString()

                if (theFinalValue.toLowerCase().indexOf((String)searchString) > -1)
                {
                    //The string was found in theFieldValue, add the Issue
                    boolQueryBuilder.add(new TermQuery(new Term("issue_id", issue.id as String)), BooleanClause.
Occur.SHOULD)
                    theFinalValue = ""
                }
            }
        }
        return boolQueryBuilder.build()
    }
}
```