Kepler - Bug #4270

RExpression exports special characters in strings as the 2 character escaped sequence

07/27/2009 01:03 PM - Oliver Soong

Status:	Resolved	**Start date:**	07/27/2009
Priority:	Normal	**Due date:**	
Assignee:	ben leinfelder	**% Done:**	0%
Category:	actors	**Estimated time:**	0.00 hour
Target version:	Unspecified	**Spent time:**	0.00 hour
Bugzilla-Id:	4270		

Description

Let's say I want to export the string "alpha", including the double quotes, on the port a from an RExpression actor. The R code is:
a <- "\"alpha\""
nchar(a) (string length) indicates 7 characters as expected.
cat(a, "\n", sep = "") displays as expected.
Exporting strings is filtered through dput, which escapes special characters, so the output string contains the 2 character escape sequence \". If I pass the output port to an Expression actor with an input port named input and evaluate the expression input.length, I get 9. Similar problems occur for \n and \t and presumably other characters I'm not thinking of at the moment.

History

#1 - 07/28/2009 01:22 PM - ben leinfelder

It's easy enough to strip out the backslashes - this makes sense in the case of \" where we are just escaping the "
But for \n and \t I don't think we're ever getting an actual newline or tab character so we'd end up with a random-looking n or t in the string if we just removed the backslash.
Thoughts?

#2 - 07/28/2009 04:25 PM - Oliver Soong

I think it'd be better to handle this more rigorously by searching specifically for special escape sequences (\", \\, \n, and \t at least) and converting them. There's the java.lang.string.replaceAll function which would probably work. Even better would be to actually figure out what all the sequences converted by R's dput are.

I haven't studied the RExpression.java code about this, but I think Kepler's catching the R stdout stream and parsing it. In that case, I think another alternative might be to replace the dput with a cat. I believe it would replace the escape sequences with the appropriate characters in stdout. Consider these R commands:

```
a <- c("a\\b", "c\"d", "e\nf", "g\th")
dput(a)
cat("c(\"", paste(a, collapse = "\", \""), "\")\n", sep = "")
cat("{\"", paste(a, collapse = "\", \""), "\"}\n", sep = "")
```

The only problem is what exactly happens on the RExpression side when it encounters one of these special characters. I think it might be easier to just continue with the dput and careful use of replaceAll.

#3 - 07/29/2009 06:34 AM - ben leinfelder

Thanks to Apache Commons, I think this is the answer:
http://commons.apache.org/lang/api-2.4/org/apache/commons/lang/StringEscapeUtils.html#unescapeJava(java.lang.String)

It will convert the "special" sequences into the actual characters we want: \t will be a tab, \n a newline, \" a quote....etc.

#4 - 07/29/2009 06:36 AM - ben leinfelder

this change is committed to trunk

#5 - 07/30/2009 09:37 AM - Oliver Soong

I did a little digging, and the escape sequences R uses in deparsing (core of dput) seems to be handled in the EncodeString function in printutils.c. There's a very minor discrepancy with the escape sequences used by Java (see http://java.sun.com/docs/books/jls/second_edition/html/lexical.doc.html#101089).

The discrepancies seem to be \a, \v, and \0 (audible bell, vertical tab, and null). A fix would be to use replaceAll after the unescapeJava, but it's complicated because these three aren't Java escape sequences, so we would have to look up the appropriate octal or unicode escape in Java.

I doubt anybody will actually use these in practice, so it's unlikely to cause any real problems. I'm deferring judgment and leaving this as closed.

#6 - 03/27/2013 02:26 PM - Redmine Admin

Original Bugzilla ID was 4270