Kepler - Bug #2963

Add data structure for tabular data and associated metadata

09/11/2007 05:45 PM - ben leinfelder

Status:	New	**Start date:**	09/11/2007
Priority:	Normal	**Due date:**	
Assignee:	ben leinfelder	**% Done:**	0%
Category:	data access	**Estimated time:**	0.00 hour
Target version:	3.X.Y	**Spent time:**	0.00 hour
Bugzilla-Id:	2963		

Description

Essentially add support for R dataframe type to be passed around the workflow.
Most likely to be used for EMLDatasource -> RExpression transfer, but having the ability to transfer it among many different actors would be ideal.

History

#1 - 09/12/2007 10:29 AM - Dan Higgins

There is already support for handing an R dataframe from one RExpression to another. If RExpression has an output port assigned to a dataframe, it writes the dataframe to a file and passes a string starting with 'dataframe:' followed by the file path. Another RExpression actor can access this as an input.

The rough equivalent Kepler data structure for a dataframe is a Kepler/PTII record. Each record has a name (the column name) and an array of values (the column vector). See $KEPLER/demos/R/emlToRecord_R.xml for an example of creating a record/dataframe and importing it into the RExpression actor.

There is one shortcoming with representing a dataframe with a record of arrays - namely, the Kepler record is not ordered; i.e. record are always referenced by name, not by location in an array. Thus, the nth item in creating a record may not be in the nth location when accessing. R dateframe columns can be referenced by name or index.

I suggest that we consider an extension of the R dataframe concept. This would include not only the list of named column arrays of the dataframe, but also additional associated metadata (i.e. all the eml attribute info) and possibly sematic information. (Dan Higgins)

#2 - 08/17/2009 02:30 PM - ben leinfelder

I've introduced an OrderedRecordToken to ptolemy - it preserves the original order of the RecordToken.
Unfortunately this can be blown away when you serialize it as a string {a={1,2,3, b={4,5,6}} and then deserialize it back to a RecordToken.
In the mean time, the 'wrp' module overrides the ptolemy RecordToken to use the correct storage class to preserve the original order. I'm not sure how we can have both implementations without changing the serialization syntax for the ordered version. Plausible?

Or just change RecordToken in Ptolemy to keep it's order and give people a method to sort them if they want that behavior

#3 - 03/27/2013 02:21 PM - Redmine Admin

Original Bugzilla ID was 2963