Package 'RObjectTables'

July 23, 2012

Version 0.3

Date 2012/07/20

Title User-level attach()'able table support

Author Duncan Temple Lang <duncan@research.bell-labs.com>

Depends R (>= 1.5.0), methods

Imports methods

Maintainer Duncan Temple Lang <duncan@research.bell-labs.com>

Description The C and S code allows one to define R objects
to be used as elements of the search path with their own
semantics and facilities for reading and writing variables.
The objects implement a simple interface via R functions (either methods or closures) and can access external data,e.g. in other applications, languages, formats, ...

License GPL

URL http://www.omegahat.org/RObjectTables, http://www.omegahat.org
http://www.omegahat.org/bugs

R topics documented:

dbexists.DirectoryTable

Methods for accessing DirectoryTable objects

Description

Methods for the DirectoryTable objects used for managing R variables (name-value bindings) in a directory rather than in memory.

Usage

```
dbread.DirectoryTable(database, name, na=1)
dbwrite.DirectoryTable(database, name, object)
dbexists.DirectoryTable(database, name)
dbobjects.DirectoryTable(database)
dbremove.DirectoryTable(database, name)
```

Arguments

database	the DirectoryTable object
name	the name of the variable of interest
na	a value which is to be returned if there is no variable in the table corresponding to the requested name. This differentiates a non-existent variable from one with a value of NULL, say.
object	the value to be assigned to the specified variable.

Details

See the corresponding generic functions.

Value

See the corresponding generic functions.

Author(s)

Duncan Temple Lang

See Also

dbread, dbobjects, dbexists, dbwrite, newRFunctionTable, attach, detach

dbmethods *Methods for user-defined tables*

Description

These are generic functions that are extended by different classes of user-level tables that can be attached to the search path. They are called when the corresponding user-level functions are called for that 'database'. A classes implementations of these methods must be globally accessible so that they can be called when needed. This differs from closure tables which pass functions, (rather than function names) to the C-level interface that implements the table's connection to the R engine.

Usage

```
dbobjects(database)
dbexists(database, name)
dbread(database, name, na=1)
dbwrite(database, name, object)
dbremove(database, name)
dbattach(database)
dbdetach(database)
dbcanCache(database, name)

dbobjects.default(database)
dbexists.default(database, name)
dbread.default(database, name, na=1)
dbremove.default(database, name)
dbwrite.default(database, name, object)
dbattach.default(database)
dbdetach.default(database)
dbcanCache.default(database, name)
```

Arguments

database	the database object which manages the name-value pairs.
name	the name of the symbol in the database.
object	an R object that is to be assigned to the specified symbol in the database.
na	a specific object that can be returned to indicate that the database does not contain an object of that name. This is similar to an "NA" while still allowing any value bound to a variable to be returned. This uses the uniqueness of the objects internal address. Its value is irrelevant, but the dbread method should not mdofy it in anyway.

Details

These methods are the S4-compatible accessors for user-level tables that can be attached to the search path. They correspond to the exists, get, remove, assign and objects that are used to access and operate on variables within elements of the search path. These are not typically called directly but by the R engine when accessing user-level tables that are implemented by particular methods for these generic functions.

These functions are compatible with the equivalent S4 functions.

Value

dbexists returns a logical value indicating whether the database has a variable by that name.

dbread is equivalent to get and returns the value in the database assigned to the specified name.

dbwrite is equivalent to assign and returns the value being assigned, i.e. object. This allows one to do chained assigments of the form x <- y <- 10.

dbremove is equivalent to remove and removes the binding for the specified name in the database, discarding the value.

dbobjects is equivalent to objects and returns a character vector containing the names of all the name-value bindings in the database.

dbcanCache returns a logical value indicating whether the value of the specified variable (given by name) cannot be changed except for in R (TRUE) or whether it might be changed externally (FALSE). This is used by the R engine to determine if it is entitle to cache the value associated with name. It does this to avoid searching through the list of elements in the search path each time it wants the value of a variable that it has already seen. This is useful when the data source can be modified externally by other applications such as another thread in a Java application, the CORBA naming service, etc.

dbattach and dbdetach have no (useful) return values and are invoked each time the user-level table is added and removed from the search path, respectively. These can be used to perform initialization and cleanup of values that the database uses to implement the other methods. For example, it might create a directory for caching values when it is attached and remove it on detach. Alternatively, it might open a connection to a database and close it when it is no longer needed.

Note

This is experimental. Make certain that important data is backed up before using this user-level table interface.

See Also

newRFunctionTable newRClosureTable attach detach http://developer.r-project.org/RObjectTables.pdf

Examples

```
fixedTable <- list(x=1, y = "abc",
                   z = list(a= rnorm(3), b = c(TRUE, FALSE, TRUE)),
                   cube = function(x) x^3)

   dbread.FixedTable <- function(database, name) {
     database[[name]]
             }
   dbremove.FixedTable <- function(database, name) {
     stop("This is a read-only table")
   }
   dbexists.FixedTable <- function(database, name) {
     any(name == names(database))
                   }

   dbobjects.FixedTable <- function(database) {
                     names(database)
             }
```

```
class(fixedTable) <- c("FixedTable")
attach(newRFunctionTable(fixedTable), name = "my fixed list")

search()
objects(2)
objects("my fixed list")

exists("x", where = 2)
find(x)
x
get("x")
get("x", pos = 2)
get("x", pos = "my fixed list")


try(assign("myVar", 10, pos = 2))
try(remove("x", pos = "my fixed list"))
try(rm(x, pos = "my fixed list"))
try(rm(x, inherits = TRUE))

detach("my fixed list")

# now the table has gone from the search list.
# It is still available as 'fixedTable'
search()
```

`DirectoryObjectTable` *Create an R variable table using a directory*

Description

This creates a `DirectoryTable` object which is used to manage R variables by reading and writing them to disk in the directory associated with the table. The variables can be accessed in the usual manner i.e. by name without the need for an explicit `get`. This gives S-like storage by writing the objects to disk when they are assigned rather than at the end if the session.

Usage

```
DirectoryObjectTable(directory, create = TRUE)
```

Arguments

`directory`	a string giving the name of the directory in which the variables will be stored.
`create`	a logical value indicating whether the directory should be created if it does not exist.

Details

This creates an object of class `DirectoryTable` that stores the name of the directory. This is passed to the different `db*` methods (`dbread`, `dbobjects`, `dbexists`, `dbwrite`, ... as the first argument).

Value

An object of class DirectoryTable that also inherits from UserDefinedDatabase. This is a list containing a single element.

dir	the fully expanded name of the directory associated with table.

Author(s)

Duncan Temple Lang

References

http://developer.r-project.org/RObjectTables.pdf http://www.omegahat.org/RinS for storing R and S-Plus objects in a common format.

See Also

dbread, dbobjects, dbexists, dbwrite, newRFunctionTable, attach, detach

Examples

```
db <- DirectoryObjectTable("/tmp/myRData")
dbwrite(db, "x", 1:10)
dbwrite(db, "y", letters[1:3])
dbobjects(db)
dbread(db, "x")

attach(newRFunctionTable(db), name = "myRData")

assign("z", c(TRUE, FALSE), pos = "myRData")
get("z", pos = 2)
get("z")
z

detach("myRData")
```

---

unbound *Get the object representing an unbound variable*

Description

This retrieves the built-in R object representing the value of an unbound variable, i.e. an undefined value. This can be returned by the dbread method of an object table if it does not have a definition for the requested variable. This makes it easy to handle such cases in the R code rather than in the underlying C-level interface that glues the internal engine and the user-level object table implementations.

Usage

```
unbound()
```

Value

Returns the internal C-level value of R_UnboundValue.

Author(s)

Duncan Temple Lang <duncan@research.bell-labs.com>

See Also

dbread

Examples

```
## Not run:
  dbread.FixedList <-
  function(database, name) {
     if(is.na(name, match(names(database$elements))))
       return(unbound())

     return(database$elements[[name]])
  }

## End(Not run)
```

UserDatabase *Create user-defined attach()able table*

Description

These functions convert a user-level object into an R object that can be attached as an element of the R search path. The newRFunctionTable works on objects that have methods for the dbexists, dbread, ... functions.

newRClosureTable works on a collection of functions (usually sharing state with a common environment) that are called directly by the C-level interface between the R engine and the user-level table.

Usage

```
newRFunctionTable(db)
newRClosureTable(db)
```

Arguments

db	a user-defined database object. When passed to newRClosureTable, this is a list of functions that implement the assign, get, exists, remove, objects, canCache, attach and detach. When passed to newRFunctionTable, this is an object with methods for the dbread, dbwrite, dbexists, dbremove, dbcanCache, dbattach, dbdetach functions.

Value

An object of class UserDefinedTable that is an external pointer to a C-level object that represents the R table. This object can then be used in a call to attach.

Note

This interface is experimental. Please ensure that important data is saved before using this.

See Also

attach http://developer.r-project.org/RTableObjects.pdf

Examples

```
source(system.file("examples", "ListTable.S", package ="RObjectTables"))
attach(newRClosureTable(createListHandlers(x=1:3, y = letters[1:4])), name ="my list table")
assign("x", 1, pos = "my list table")
objects(pos = 2)
exists("x", pos = 2)
remove("x", 1, pos = "my list table")
exists("x", pos = 2)
```

Index