Package 'SchemaOnRead'

August 29, 2016

Type Package

Title Automated Schema on Read

Version 1.0.2

Date 2015-12-22

Description Provides schema-on-read tools including a single function call (e.g., schemaOnRead('filename')) that reads text ('TXT'), comma separated value ('CSV'), raster image ('BMP', 'PNG', 'GIF', 'TIFF', and 'JPG'), R data ('RDS'), HDF5 ('H5'), NetCDF ('CS'), spreadsheet ('XLS', 'XLSX', 'ODS', and 'DIF'), Weka Attribute-Relation File Format ('ARFF'), Epi Info ('REC'), SPSS ('SAV'), Systat ('SYS'), and Stata ('DTA') files. It also recursively reads folders (e.g., schemaOnRead('folder')), returning a nested list of the contained elements.

Copyright Argonne National Laboratory

License GPL-3

Depends R (>= 3.2.1)

Imports caTools (>= 1.17.1), foreign (>= 0.8-66), ncdf4 (>= 1.14), network (>= 1.12.0), readbitmap (>= 0.1-4), readODS (>= 1.4), tiff (>= 0.1-5), xml2 (>= 0.1.2), haven (>= 0.2.0), readxl (>= 0.1.0)

URL https://github.com/drmichaelnorth/SchemaOnRead

LazyData TRUE

Suggests knitr (>= 1.11), rmarkdown (>= 0.8.1), testthat (>= 0.10.0)

VignetteBuilder knitr

NeedsCompilation no

Author Michael North [aut, cre]

Maintainer Michael North <north@anl.gov>

Repository CRAN

Date/Publication 2015-12-24 08:18:59

R topics documented:

checkExtensions 2
defaultProcessors 3
schemaOnRead 3
simpleProcessors 4

Index **6**

`checkExtensions` *Automated Schema on Read File Extensions Checker*

Description

Checks to see if the given file exists and has one of the provided extensions.

Usage

```
checkExtensions(path = ".", extensions = NULL)
```

Arguments

`path`	a file to load or folder to recursively load.
`extensions`	a list of file extensions to check provided as lower case strings.

See Also

`checkExtensions`.

Examples

```
## Load the needed library.
library(SchemaOnRead)

## Define a new processor.
newProcessor <- function(path, ...) {

  # Check the file existance and extensions.
  if (!SchemaOnRead::checkExtensions(path, c("xyz"))) return(NULL)

  ## As an example, attempt to read an XYZ file as a CSV file.
  read.csv(path, header = FALSE)

}

## Define a new processors list.
newProcessors <- c(newProcessor, SchemaOnRead::simpleProcessors())

# Use the new processors list.
schemaOnRead(path = "../inst/extdata", processors = newProcessors)
```

defaultProcessors *Automated Schema on Read Default Processors*

Description

Provides a default list of extensions for custom processing.

Usage

```
defaultProcessors()
```

See Also

defaultProcessors.

Examples

```
## Load the needed library.
library(SchemaOnRead)

## Define a new processor.
newProcessor <- function(path, ...) {

  # Check the file existance and extensions.
  if (!SchemaOnRead::checkExtensions(path, c("xyz"))) return(NULL)

  ## As an example, attempt to read an XYZ file as a CSV file.
  read.csv(path, header = FALSE)

}

## Define a new processors list.
newProcessors <- c(newProcessor, SchemaOnRead::defaultProcessors())

# Use the new processors list.
schemaOnRead(path = "../inst/extdata", processors = newProcessors)
```

schemaOnRead *Automated Schema on Read*

Description

Provides schema-on-read tools including a single function call (e.g., schemaOnRead("filename")) that reads text (TXT), comma separated value (CSV), raster image (BMP, PNG, GIF, TIFF, and JPG), R data (RDS), HDF5 ('H5'), NetCDF ('CS'), spreadsheet (XLS, XLSX, ODS, and DIF), Weka Attribute-Relation File Format (ARFF), Epi Info (REC), Pajek network (PAJ), R network (NET), Hypertext Markup Language (HTML), SPSS (SAV), Systat (SYS), and Stata (DTA) files.

It also recursively reads folders (e.g., schemaOnRead("folder")), returning a nested list of the contained elements.

The standard use of SchemaOnRead is to recursively load a folder. The result is a named list of elements for each entry in the folder's tree. Sub-elements (e.g., files or subfolders) of a folder can be accessed using the R named list ("\$") operator followed by the sub-element name. Files or folders with names that do not conform to standard R variable naming requirements can be accessed using single quote notation (e.g., "results$'Nonconforming Name!'"). `schemaOnRead(path)`.

Usage

```
schemaOnRead(path = ".", processors = SchemaOnRead::defaultProcessors(), verbose = FALSE)
```

Arguments

`path`	a file to load or folder to recursively load.
`processors`	a list of processors to use in the order in the list.
`verbose`	a flag specifying whether or not to show progress feedback, warnings, and errors.

See Also

`schemaOnRead`.

Examples

```
## Load a file to a variable.
file <- SchemaOnRead::schemaOnRead("file.txt")

## Load a directory to a list.
folder <- SchemaOnRead::schemaOnRead("folder")
```

`simpleProcessors` *Automated Schema on Read Simple Processors*

Description

Provides a simple list of extensions for custom processing.

Usage

```
simpleProcessors()
```

See Also

`simpleProcessors`.

Examples

```
## Load the needed library.
library(SchemaOnRead)

## Define a new processor.
newProcessor <- function(path, ...) {

  # Check the file existance and extensions.
  if (!SchemaOnRead::checkExtensions(path, c("xyz"))) return(NULL)

  ## As an example, attempt to read an XYZ file as a CSV file.
  read.csv(path, header = FALSE)

}

## Define a new processors list.
newProcessors <- c(newProcessor, SchemaOnRead::simpleProcessors())

# Use the new processors list.
schemaOnRead(path = "../inst/extdata", processors = newProcessors)
```

Index

∗Topic **checkExtensions**
checkExtensions, 2
∗Topic **defaultProcessors**
defaultProcessors, 3
∗Topic **schemaOnRead**
schemaOnRead, 3
∗Topic **simpleProcessors**
simpleProcessors, 4

checkExtensions, *2*, 2

defaultProcessors, *3*, 3

schemaOnRead, 3, *4*
simpleProcessors, *4*, 4