Package 'factualR'

February 19, 2015

Type Package

Title thin wrapper for the Factual.com server API

Version 0.5

Date 2010-12-31

Author Ethan McCallum

Maintainer Ethan McCallum <factualr-package@exmachinatech.net>

Description Per the Factual.com website, ``Factual is a platform where anyone can share and mash open, living data on any subject." The data is in the form of tables and is accessible via REST API. The factualR package is a thin wrapper around the Factual.com API, to make it even easier for people working with R to explore Factual.com data sets.

License Apache License 2.0

LazyLoad yes

Depends methods, RJSONIO, RCurl

URL http://www.exmachinatech.net/factualR/

Repository CRAN

Date/Publication 2011-01-03 12:35:05

NeedsCompilation no

R topics documented:

`factualR-package` *thin wrapper for the Factual.com server API*

Description

Per the Factual.com website, "Factual is a platform where anyone can share and mash open, living data on any subject." The data is in the form of tables and is accessible via REST API. The factualR package is a thin wrapper around the Factual.com API, to make it even easier for people working with R to explore Factual.com data sets.

For now, this package supports read-only requests ("read" and "schema"); a future revision may support modification of Factual.com data ("input" API calls).

Details

Package:	factualR
Type:	Package
Version:	1.0
Date:	2010-12-26
License:	Apache 2.0
LazyLoad:	yes
Depends:	methods RJSONIO RCurl

To use this package, please refer to the functions `createFactualConnection`, `factualGetSchema`, and `factualRead`. That documentation includes examples.

Also, please refer to the developer documentation for the "Server API" at `http://wiki.developer.factual.com/`

Author(s)

Ethan McCallum

Maintainer: Ethan McCallum <factualr-package@exmachinatech.net>

References

http://wiki.developer.factual.com/

See Also

`createFactualConnection` `factualGetSchema` `factualRead`

`factualR` *common functions in the factualR package*

Description

This document describes the end-user functions for the factualR package.

Please note: as this package is a thin wrapper around the Factual.com API, you will need a Factual API Key. You can sign up for one at `http://www.factual.com/`.

For now, this package supports read-only requests ("read" and "schema"); a future revision may support modification of Factual.com data ("input" API calls).

Usage

```
createFactualConnection( apiKey = NULL , apiVersion = NULL , baseURL = NULL )

factualGetSchema( connection , tableID , verbose = FALSE )

factualRead(connection , tableID , sort = NULL , limit = NULL , offset = NULL , filters = NULL , subjec
```

Arguments

`apiKey`	your "Factual API key" as acquired from `http://www.factual.com/`.
`apiVersion`	the version of the Factual.com API. Unless you are a developer on factualR, you needn't specify this parameter.
`baseURL`	the base URL for calls to Factual.com. Unless you are a developer on factualR, you needn't specify this parameter.
`connection`	a connection as returned from `createFactualConnection()`.
`tableID`	the table's unique identifier, sometimes called "table reference," from Factual.com.
`sort`	sort field(s), using JSON syntax. factualR will URL-encode this string for you.
`limit`	maximum number of records (rows) to pull from this table.
`offset`	start pulling records (rows) from this numeric offset.
`filters`	limit records (rows) based on these criteria, in JSON format. factualR will URL-encode this string for you.
`subject_key`	unique identifier for a row. (If you're exporting several rows, e.g. for an analysis, you probably won't use this.)
`reformatNames`	whether to reformat column names to be more R-friendly (e.g., remove all but alphanumeric characters).
`verbose`	should we show debugging information, such as the URL, during processing? (Useful for debugging.)

Details

For a description of the `filters` and `sort` filters, please refer to the Factual.com developer documentation at: `http://wiki.developer.factual.com/w/page/29670788/Server-API`.

Note that factualR will URL-encode all parameters for you; please don't URL-encode any of the values passed to `factualGetSchema` or `factualRead`.

Under some circumstances, a call to `factualRead()` or `factualGetSchema()` will fail in a way that does not return an object. When that happens, please rerun the call using `verbose=TRUE` to print the target URL. If you paste that into a browser or a commandline tool such as `curl`, you can determine whether there is a connectivity problem to Factual.com.

Author(s)

Ethan McCallum

References

The main `http://www.factual.com/` website explains Factual's mission.

`http://www.factual.com/devtools/serverAPI` explains the REST API used by factualR, including the (optional) `filters` and `sort` parameters used by `factualRead`.

Finally, `http://wiki.developer.factual.com/w/page/12298839/Basics` explains more about the elements common to all responses from the API, such as `status` and `message`.

Examples

```
## Not run:
## make sure you have signed up for a developer API key at http://www.factual.com/

myAPIKey <- " ... YOUR Factual.com API KEY"

factual <- createFactualConnection( myAPIKey )

## Skim http://www.factual.com/topics for an interesting table.  Click the table's
## link and then click the "develop" tab.  Note the "table reference."
tableID <- "... table reference from the table's 'Develop' tab ..."

## now, let's get an idea of the table's schema and metadata:
table.meta <- factualGetSchema( factual , tableID )

str(table.meta)

## with that in mind, get the table's data
table.data <- factualRead( factual , tableID )

## get an idea of the result object
str(table.data)

## that's great, but we really want to play with the table data
## (it's a data frame)
table.data@results
```

```
## hm, let's get 60 rows so we can explore
table.data.small <- factualRead( factual , tableID , limit=60 )

str(table.data.small@results)


## let's use some filters to limit the data we pull.
## pretend the table of interest has colums named "state"
## and "city"
filters <- '{"city":"New York","state":"NY"}'

table.data.filtered <- factualRead( factual , tableID , filters = filters )
str(table.data.small@filtered)

## End(Not run)
```

FactualReadResult-class

Class "FactualReadResult"

Description

Encapsulates the result of the Factual.com "read" API call, which reads data from a table.

Objects from the Class

Objects of type `FactualReadResult` should not be created directly; instead, please invoke factualRead.

Slots

`url`: URL used to call the Factual.com API. Useful mainly for debugging.

`tableID`: unique identifier for the Factual.com table you queried.

`status`: request status as returned by Factual.com's API: one of "ok," "warn," or "error."

`message`: blank if the Factual handled the request without a problem; otherwise, a description of the problem.

`resultRows`: number of rows in this result.

`tableRows`: number of total rows in the table.

`results`: a `data.frame` of the table returned by Factual.com.

`processingTime`: time required on Factual's end to process the request; returned by Factual.com in verbose mode.

`runTime`: time (ms) required for round-trip call to Factual.com.

`fetchedAt`: date/time the data was requested from Factual.com.

Author(s)

Ethan McCallum

See Also

factualRead

`FactualSchemaResult-class`
Class "FactualSchemaResult"

Description

Encapsulates the result of the Factual.com "schema" API call, which describes a table's structure.

Objects from the Class

Objects of type `FactualSchemaResult` should not be created directly; instead, please invoke factualGetSchema.

Slots

`url`: URL used to call the Factual.com API. Useful mainly for debugging.

`tableID`: unique identifier for the Factual.com table you queried.

`status`: request status as returned by Factual.com's API: one of "ok," "warn," or "error."

`message`: blank if the Factual handled the request without a problem; otherwise, a description of the problem.

`resultRows`: number of rows in this result.

`tableRows`: number of total rows in the table.

`results`: a data frame of the table returned by Factual.com.

`name`: human-readable description of the table.

`table.meta`: a `data.frame` of table metadata, such as the creator, total number of rows, and description.

`fields`: a `data.frame` of the fields in this table. Each row describes one field's name, data type, and so on.

`processingTime`: time required on Factual's end to process the request; returned by Factual.com in verbose mode.

`runTime`: time (ms) required for round-trip call to Factual.com.

`fetchedAt`: date/time the data was requested from Factual.com.

Author(s)

Ethan McCallum

See Also

factualGetSchema

Index