Package: Storage (via r-universe)

October 29, 2024

Title Data Access Layer

Version 0.0.1.0003

Description Test Data Access Layer Implementation for Standard Compliant Todo App.

License file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports Validate, Environment, Query, DBI, odbc, uuid

Remotes FlippieCoetser/Validate, FlippieCoetser/Environment, FlippieCoetser/Query

Suggests testthat (>= 3.0.0),

Config/testthat/edition 3

Depends R (>= 2.10)

Repository https://flippiecoetser.r-universe.dev

RemoteUrl https://github.com/FlippieCoetser/Storage

RemoteRef HEAD

RemoteSha e9c69e290a712dce1d77bae89afde14aa81a68e2

Contents

`ODBC.Configurator` *Configure Data Store*

Description

Different data store providers: ODBC, SQL, File System, requires different configuration parameters. `ODBC.Configurator()` helps retrieve and validate ODBC specific configuration sets.

`ODBC.Configurator()` returns a set of functions that can be used to define and retrieve configuration sets required by ODBC Data Sources:

- The first function: `open.config.file`, will open an existing or a new empty `.Renviron` configuration file in your IDE. User can then review or add environment variables for either a Preset Configuration: OS defined DSN or Manual Configuration.
- The second function: `Get.Config`, will return a Preset Configuration by default or if type = 'Manual' is passed a Manual Configuration. Independent of the type of configuration retrieved, extensive validation will be performed to ensure all required parameters are available. If any parameter is missing, an error message will be thrown with useful information to allows the user to fix the problem.

Value

A `list` of functions:

- `open.config.file()`
- `Get.Config()`

`Storage` *Storage*

Description

Provide an storage interface with basic CRUD Operations with ability to switch between MOCK or ODBC Storage Provider.

Value

A `list` of functions:

- `execute.query(query)`
- `add(entity, table)`
- `retrieve(table, fields)`
- `retrieve.where.id(id, table, fields)`
- `modify(entity, table)`
- `remove(id, table)`

`Todo.Broker` *Example Todo Broker*

Description

Provide a Broker which leverage the storage interface provided by this package. The broker excludes structural and logical validation of data.

Value

A `list` of functions:

- `insert(todo)`
- `select(...)`
- `select.where.Id(id)`
- `update(todo)`
- `delete(id)`

`Todo.Service` *Example Todo Service*

Description

Provide a Service which leverage the storage interface provided by this package. The service includes structural and logical validation of data.

Value

A `list` of functions:

- `add(todo)`
- `retrieve(...)`
- `retrieve.where.id(id)`
- `modify(todo)`
- `remove(id)`

`Todos` *Todos Dataset*

Description

This dataset contains todos for testing the package.

Usage

```
Todos
```

Format

A data frame with 3 rows and 5 columns:

id Unique identifier for the todo, stored as a UUID string.

task Description of the task.

status Status of the task.

Source

Generated synthetic data.

Index