Package ‘resourcer’

October 14, 2022

Type Package
Title Resource Resolver
Version 1.3.0
Description A resource represents some data or a computation unit. It is described by a URL and credentials. This package proposes a Resource model with `resolver` and `client` classes to facilitate the access and the usage of the resources.
License LGPL (>= 2.1)
Depends R (>= 3.5.0), R6, httr
Suggests testthat, knitr, haven, readr, readxl, ssh, sys, mongolite, dplyr, dbplyr, DBI, RMariaDB, RPostgres, sparklyr, R Presto (>= 1.4.0), nodbi, rmarkdown
BugReports https://github.com/obiba/resourcer
RoxygenNote 7.2.1
VignetteBuilder knitr
Encoding UTF-8
NeedsCompilation no
Author Yannick Marcon [aut, cre] (<https://orcid.org/0000-0003-0138-2023>), OBiBa group [cph]
Maintainer Yannick Marcon <yannick.marcon@obiba.org>
Repository CRAN
Date/Publication 2022-09-30 15:20:06 UTC

R topics documented:

as.data.frame.resource .................. 3
as.data.frame.ResourceClient .......... 3
as.resource.data.frame ................ 4
as.resource.object .................... 4
as.resource.tbl ....................... 5
as.data.frame.resource

Coerce a resource to a data.frame

Description

Attempt to coerce a resource object to a data.frame: find a ResourceResolver and get the ResourceClient that will connect to the described dataset and make a data.frame of it.

Usage

## S3 method for class 'resource'
as.data.frame(x, ...)

Arguments

x a resource object.

... additional parameters, that may be used (or ignored) by the resource client.

Value

a data.frame (or a tibble)

as.data.frame.ResourceClient

Coerce a ResourceClient object to a data.frame

Description

Attempt to coerce a resource object to a data.frame: find a ResourceResolver and get the ResourceClient that will connect to the described dataset and make a data.frame of it.

Usage

## S3 method for class 'ResourceClient'
as.data.frame(x, ...)

Arguments

x a ResourceClient object

... additional parameters, that may be used (or ignored) by the resource client.

Value

a data.frame (or a tibble)
**as.resource.data.frame**

*Coerce resource client to a data.frame*

**Description**

Coerce a ResourceClient object to a data.frame.

**Usage**

```r
as.resource.data.frame(x, strict = FALSE, ...)
```

**Arguments**

- `x` : The ResourceClient object to coerce to a data.frame.
- `strict` : logical whether the resulting object must be strictly of class data.frame or if it can be a tibble.
- `...` : Additional parameters, that may be used (or ignored) by the resource client.

**Value**

a data.frame (or a tibble)

---

**as.resource.object**

*Coerce resource client to the internal data object*

**Description**

Coerce a ResourceClient object to internal data object: depending on the implementation of the ResourceClient, it can be a data connection object (like a DBI connection to a SQL database), or the actual data structure (when a resource is a R object extracted from a R data file for instance).

**Usage**

```r
as.resource.object(x, ...)
```

**Arguments**

- `x` : The ResourceClient object to coerce to a data.frame.
- `...` : Additional parameters, that may be used (or ignored) by the resource client.

**Value**

the internal data object.
Description

Coerce a ResourceClient object to a dplyr’s tbl.

Usage

\texttt{as.resource.tbl(x, ...)}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{2cm} The ResourceClient object to coerce to a data.frame
  \item \texttt{...} \hspace{2cm} Additional parameters, that may be used (or ignored) by the resource client.
\end{itemize}

Value

a dplyr's tbl

CommandResourceClient  \hspace{.5cm} \textit{Command resource client}

Description

Command resource client

Command resource client

Format

A R6 object of class CommandResourceClient

Details

Base class for resource clients issuing commands and get a result with the status of the execution, the output and the error messages.

Super class

\texttt{resourcer::ResourceClient} -> CommandResourceClient
Methods

Public methods:
• `CommandResourceClient$new()`
• `CommandResourceClient$clone()`

Method `new()`: Creates a new `CommandResourceClient` instance

Usage:
CommandResourceClient$new(resource)

Arguments:
resource  A valid resource object.

Returns:  A `CommandResourceClient` object.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
CommandResourceClient$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

---

**DBIResourceConnector**  **DBI resource connector**

Description

DBI resource connector
DBI resource connector

Format

A R6 object of class DBIResourceConnector

Details

Makes a DBI connection from a resource description, used in SQLResourceClient that is based on DBI.

Methods

Public methods:
• `DBIResourceConnector$new()`
• `DBIResourceConnector$isFor()`
• `DBIResourceConnector$createDBIConnection()`
• `DBIResourceConnector$getName()`
• `DBIResourceConnector$readDBTable()`
DBIResourceConnector

- `DBIResourceConnector$readDBTibble()`
- `DBIResourceConnector$closeDBIConnection()`
- `DBIResourceConnector$clone()`

**Method `new()`:** Creates a new DBIResourceConnector instance

*Usage:*

```
DBIResourceConnector$new()
```

*Returns:* A DBIResourceConnector object.

**Method `isFor()`:** Check that the provided parameter is of class "resource".

*Usage:*

```
DBIResourceConnector$isFor(resource)
```

*Arguments:*

- `resource` The resource object to validate.

*Returns:* A logical.

**Method `createDBIConnection()`:** Stub function which subclasses will implement to create a DBI connection object from a resource.

*Usage:*

```
DBIResourceConnector$createDBIConnection(resource)
```

*Arguments:*

- `resource` A valid resource object.

**Method `getTableName()`:** Get the SQL table name from the resource URL.

*Usage:*

```
DBIResourceConnector$getTableName(resource)
```

*Arguments:*

- `resource` A valid resource object.

*Returns:* The SQL table name.

**Method `readDBTable()`:** Read a table as a vanilla tibble using DBI connection object.

*Usage:*

```
DBIResourceConnector$readDBTable(conn, resource)
```

*Arguments:*

- `conn` A DBI connection object.
- `resource` A valid resource object.

*Returns:* A vanilla tibble.

**Method `readDBTibble()`:** Read a table as a SQL tibble using DBI connection object.

*Usage:*

```
DBIResourceConnector$readDBTibble(conn, resource)
```

*Arguments:*
conn  A DBI connection object.
resource  A valid resource object.

Returns: A SQL tibble.

Method closeDBIConnection(): Disconnect the DBI connection.

Usage:
DBIResourceConnector$closeDBIConnection(conn)

Arguments:
conn  A DBI connection object.

Method clone(): The objects of this class are cloneable with this method.

Usage:
DBIResourceConnector$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.
FileResourceGetter

Usage:
FileResourceClient$new(resource, file.getter = NULL)

Arguments:
resource  A valid resource object.
file.getter A FileResourceGetter object, optional. If not provided, it will be looked up in
the FileResourceGetters registry. The operation will fail if none can be found.

Returns: A FileResourceClient object.

Method downloadFile(): Performs the file download, if it does not already exists locally.

Usage:
FileResourceClient$downloadFile()

Returns: The local path to the downloaded file.

Method close(): Removes the file if it was downloaded. A local file resource will remain
untouched.

Usage:
FileResourceClient$close()

Method clone(): The objects of this class are cloneable with this method.

Usage:
FileResourceClient$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

FileResourceGetter  File resource getter

Description
File resource getter
File resource getter

Format
A R6 object of class FileResourceGetter

Details
Helper base class for getting the file described by the resource object. ResourceClient class imple-
mentations can use this utility to retrieve files from any locations before processing them according
to the declared data format.
Methods

Public methods:

• FileResourceGetter$new()
• FileResourceGetter$isFor()
• FileResourceGetter$downloadFile()
• FileResourceGetter$extractFileName()
• FileResourceGetter$makeDownloadDir()
• FileResourceGetter$clone()


Usage:
FileResourceGetter$new()


Method isFor(): Check that the provided parameter is of class "resource" and has a format defined.

Usage:
FileResourceGetter$isFor(resource)

Arguments:
resource The resource object to validate.

Returns: A logical.

Method downloadFile(): Stub function which subclasses will implement to make a "resource.file" object from a resource.

Usage:
FileResourceGetter$downloadFile(resource, ...)

Arguments:
resource A valid resource object.
... Additional parameters that may be relevant for FileResourceGetter subclasses.

Returns: A "resource.file" object.

Method extractFileName(): Utility to get the base name from the file path.

Usage:
FileResourceGetter$extractFileName(resource)

Arguments:
resource A valid resource object.

Returns: The file base name.

Method makeDownloadDir(): Creates a directory where to download file in the R session’s temporary directory. This directory will be flushed when the R session ends.

Usage:
FileResourceGetter$makeDownloadDir()
findDBIResourceConnector

Find a DBI resource connector

Description

Find the DBI resource connector that will download the DBI from the provided resource object.

Usage

findDBIResourceConnector(x)

Arguments

x The resource object which corresponding DBI connector is to be found.

Value

The corresponding DBIResourceConnector object or NULL if none applies.

findFileResourceGetter

Find a file resource getter

Description

Find the file resource getter that will download the file from the provided resource object.

Usage

findFileResourceGetter(x)

Arguments

x The resource object which corresponding file getter is to be found.

Value

The corresponding FileResourceGetter object or NULL if none applies.
getDBIResourceConnectors

Get DBI resource connectors registry

Description
Get the DBIResourceConnectors registry, create it if it does not exist.

Usage
getDBIResourceConnectors()

getFileResourceGetters

Get file resource getters registry

Description
Get the FileResourceGetters registry, create it if it does not exist.

Usage
ggetFileResourceGetters()

getResourceResolvers
Get resource resolvers registry

Description
Get the resource resolvers registry, create it if it does not exist.

Usage
getResourceResolvers()

Examples
{
    resourcer::getResourceResolvers()
}
GridFsFileResourceGetter

*GridFS file resource getter*

**Description**

GridFS file resource getter

**Format**

A R6 object of class GridFsFileResourceGetter

**Details**

Access a file that is in the MongoDB file store (GridFS). Credentials may apply.

**Super class**

`resourcer::FileResourceGetter -> GridFsFileResourceGetter`

**Methods**

**Public methods:**

- `GridFsFileResourceGetter$new()`
- `GridFsFileResourceGetter$isFor()`
- `GridFsFileResourceGetter$downloadFile()`
- `GridFsFileResourceGetter$clone()`

**Method `new()`**: Creates a new GridFsFileResourceGetter instance.

*Usage:*

`GridFsFileResourceGetter$new()`


**Method `isFor()`**: Check that the provided resource has a URL that locates a GridFS object: either the URL scheme is "gridfs" or it is "mongodb"/"mongodb+srv" with a query parameter "prefix=fs" (that identifies GridFS collection names).

*Usage:*

`GridFsFileResourceGetter$isFor(resource)`

*Arguments:*

- `resource` The resource object to validate.

*Returns*: A logical.

**Method `downloadFile()`**: Download the file from the MongoDB file store in a temporary location.
HttpFileResourceGetter

Usage:
GridFsFileResourceGetter$downloadFile(resource, ...)

Arguments:
resource  A valid resource object.
...  Unused additional parameters.

Returns:  The "resource.file" object.

Method clone(): The objects of this class are cloneable with this method.

Usage:
GridFsFileResourceGetter$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

Description
HTTP file resource getter
HTTP file resource getter

Format
A R6 object of class HttpFileResourceGetter

Details
Access a file that is stored at a HTTP(S) address. Use Basic authentication header if both resource’s identity and secret are defined.

Super class
resourcer::FileResourceGetter -> HttpFileResourceGetter

Methods
Public methods:
• HttpFileResourceGetter$new()
• HttpFileResourceGetter$isFor()
• HttpFileResourceGetter$downloadFile()
• HttpFileResourceGetter$clone()

LocalFileResourceGetter

Usage:
HttpFileResourceGetter$new()


Method isFor(): Check that the provided resource has a URL that locates a file accessible through "http" or "https".

Usage:
HttpFileResourceGetter$isFor(resource)

Arguments:
resource The resource object to validate.

Returns: A logical.

Method downloadFile(): Download the file from the remote address in a temporary location. Applies Basic authentication if credentials are provided in the resource.

Usage:
HttpFileResourceGetter$downloadFile(resource, ...)

Arguments:
resource A valid resource object.
... Unused additional parameters.

Returns: The "resource.file" object.

Method clone(): The objects of this class are cloneable with this method.

Usage:
HttpFileResourceGetter$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

LocalFileResourceGetter

Local file resource getter

Description

Local file resource getter

Format

A R6 object of class LocalFileResourceGetter

Details

Access a file that is stored in the local file system. No credentials apply.
Super class

```
resourcer::FileResourceGetter -> LocalFileResourceGetter
```

Methods

**Public methods:**

- `LocalFileResourceGetter$new()`
- `LocalFileResourceGetter$isFor()`
- `LocalFileResourceGetter$downloadFile()`
- `LocalFileResourceGetter$clone()`

**Method new():** Creates a new LocalFileResourceGetter instance.

*Usage:*

```
LocalFileResourceGetter$new()
```

*Returns:* A LocalFileResourceGetter object.

**Method isFor():** Check that the provided resource has a URL that locates a file stored in the local file system.

*Usage:*

```
LocalFileResourceGetter$isFor(resource)
```

*Arguments:*

- `resource` The resource object to validate.

*Returns:* A logical.

**Method downloadFile():** Make a "resource.file" object from a local file resource.

*Usage:*

```
LocalFileResourceGetter$downloadFile(resource, ...)
```

*Arguments:*

- `resource` A valid resource object.
- `...` Unused additional parameters.

*Returns:* The "resource.file" object.

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

```
LocalFileResourceGetter$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.
MariaDBResourceConnector

MariaDB DBI resource connector

Description

MariaDB DBI resource connector

Format

A R6 object of class MariaDBResourceConnector

Details

Makes a MariaDB/MySQL DBI connection from a resource description.

Super class

resourcer::DBIResourceConnector -> MariaDBResourceConnector

Methods

Public methods:

• MariaDBResourceConnector$new()
• MariaDBResourceConnector$isFor()
• MariaDBResourceConnector$createDBIConnection()
• MariaDBResourceConnector$clone()

Method new(): Creates a new MariaDBResourceConnector instance.

Usage:

MariaDBResourceConnector$new()

Returns: A MariaDBResourceConnector object.

Method isFor(): Check that the provided resource has a URL that locates a MySQL or MariaDB object: the URL scheme must be "mysql" or "mariadb".

Usage:

MariaDBResourceConnector$isFor(resource)

Arguments:

resource The resource object to validate.

Returns: A logical.

Method createDBIConnection(): Creates a DBI connection object from a resource.

Usage:
MariaDBResourceConnector$createDBIConnection(resource)

Arguments:
resource A valid resource object.

Returns: A DBI connection object.

Method clone(): The objects of this class are cloneable with this method.

Usage:
MariaDBResourceConnector$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

---

### newResource

**Create a Resource**

**Description**

Creates a new Resource structure.

**Usage**

newResource(name = "", url, identity = NULL, secret = NULL, format = NULL)

**Arguments**

- **name**: Optional human friendly name that identifies the resource.
- **url**: URL to access the resource whether it is data or computation capability.
- **identity**: User name or account ID (if credentials are applicable).
- **secret**: User password or token (if credentials are applicable).
- **format**: Data format, to help resource resolver identification and coercing to other formats, optional.

**Examples**

```r
# make a SPSS file resource
res <- resourcer::newResource(
  name = "CNSIM1",
  url = "file:///data/CNSIM1.sav",
  format = "spss"
)
```
newResourceClient

Creates a resource client

Description

From a resource object, find the corresponding resolver in the resolver registry and create a new resource client.

Usage

newResourceClient(x)

Arguments

x

The resource object which corresponding resolver is to be found.

Value

The corresponding ResourceClient object or NULL if none applies.

Examples

```r
library(resourcer)
res <- newResource(
    name = "CNSIM1",
    url = "file:///data/CNSIM1.sav",
    format = "spss"
)
client <- newResourceClient(res)
```

NoSQLResourceClient

NoSQL database resource client

Description

NoSQL database resource client

NoSQL database resource client

Format

A R6 object of class NoSQLResourceClient
NoSQLResourceClient

Details
Resource client that connects to a NoSQL database supported by nodbi.

Super class
resourcer::ResourceClient -> NoSQLResourceClient

Methods

Public methods:

• NoSQLResourceClient$new()
• NoSQLResourceClient$getConnection()
• NoSQLResourceClient$asDataFrame()
• NoSQLResourceClient$getDatabaseName()
• NoSQLResourceClient$getTableName()
• NoSQLResourceClient$close()
• NoSQLResourceClient$clone()

Method `new()`: Creates a new NoSQLResourceClient instance.
Usage:
NoSQLResourceClient$new(resource)
Arguments:
resource  A valid resource object.
Returns:  A NoSQLResourceClient object.

Method `getConnection()`: Creates the nodbi connection object if it does not exist.
Usage:
NoSQLResourceClient$getConnection()
Returns:  The nodbi connection object.

Method `asDataFrame()`: Makes a data.frame from the remote database table.
Usage:
NoSQLResourceClient$asDataFrame()
Returns:  A tibble.

Method `getDatabaseName()`: Extract the database name from the resource URL.
Usage:
NoSQLResourceClient$getDatabaseName()
Returns:  The database name.

Method `getTableName()`: Extract the database table name from the resource URL.
Usage:
NoSQLResourceClient$getTableName()
Returns: The database table name.

Method close(): Close the nodbi connection.
Usage:
NoSQLResourceClient$close()

Method clone(): The objects of this class are cloneable with this method.
Usage:
NoSQLResourceClient$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

NoSQLResourceResolver NoSQL Database Resource resolver

Description
NoSQL Database Resource resolver
NoSQL Database Resource resolver

Format
A R6 object of class NoSQLResourceResolver

Details
The resource is NoSQL database such as mongodb (elasticsearch, redis, couchdb, sqlite are not supported yet).

Super class
resourcer::ResourceResolver -> NoSQLResourceResolver

Methods
Public methods:
• NoSQLResourceResolver$isFor()
• NoSQLResourceResolver$newClient()
• NoSQLResourceResolver$clone()

Method isFor(): Check that the provided resource has a URL that locates a nodbi object: the URL scheme must be one of "mongodb", "mongodb+srv". Other NoSQL databases "elasticsearch", "redis", "couchdb", "sqlite" are not supported yet.
Usage:
NoSQLResourceResolver$isFor(x)
**OpalFileResourceGetter**

*Arguments:*

- `x` The resource object to validate.

*Returns:* A logical.

**Method** `newClient()`: Creates a NoSQLResourceClient instance from provided resource.

*Usage:*

```r
NoSQLResourceResolver$newClient(x)
```

*Arguments:*

- `x` A valid resource object.

*Returns:* A NoSQLResourceClient object.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```r
NoSQLResourceResolver$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.

---

**OpalFileResourceGetter**

*Opal file resource getter*

---

**Description**

Opal file resource getter

Opal file resource getter

**Format**

A R6 object of class `OpalFileResourceGetter`

**Details**

Access a file that is stored in an Opal server. Use Basic authentication header if both resource’s identity and secret are defined, token authentication if secret only is defined.

**Super class**

`resourcer::FileResourceGetter` -> `OpalFileResourceGetter`
Methods

Public methods:

- `OpalFileResourceGetter$new()`
- `OpalFileResourceGetter$isFor()`
- `OpalFileResourceGetter$downloadFile()`
- `OpalFileResourceGetter$clone()`

**Method new():** Creates a new `OpalFileResourceGetter` instance.

*Usage:*

`OpalFileResourceGetter$new()`


**Method isFor():** Check that the provided resource has a URL that locates a Opal file: the URL scheme must be "opal+http" or "opal+https" and the path must designate a file web service entry point (i.e. starts with "ws/files/").

*Usage:*

`OpalFileResourceGetter$isFor(resource)`

*Arguments:*

- `resource` The resource object to validate.

*Returns:* A logical.

**Method downloadFile():** Download the file from the Opal file system in a temporary location.

*Usage:*

`OpalFileResourceGetter$downloadFile(resource, ...)`

*Arguments:*

- `resource` A valid resource object.
- `...` Unused additional parameters.

*Returns:* The "resource.file" object.

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

`OpalFileResourceGetter$clone(deep = FALSE)`

*Arguments:*

- `deep` Whether to make a deep clone.
PostgresResourceConnector

Postgres DBI resource connector

Description
Postgres DBI resource connector
Postgres DBI resource connector

Format
A R6 object of class PostgresResourceConnector

Details
Makes a Postgres DBI connection from a resource description.

Super class
resourcer::DBIResourceConnector -> PostgresResourceConnector

Methods

Public methods:
• PostgresResourceConnector$new()
• PostgresResourceConnector$isValid()
• PostgresResourceConnector$createDBIConnection()
• PostgresResourceConnector$clone()

Method new(): Creates a new PostgresResourceConnector instance.

Usage:
PostgresResourceConnector$new()

Returns: A PostgresResourceConnector object.

Method isValid(): Check that the provided resource has a URL that locates a Postgres object: the URL scheme must be "postgres" or "postgresql".

Usage:
PostgresResourceConnector$isValid(resource)

Arguments:
resource  The resource object to validate.

Returns: A logical.

Method createDBIConnection(): Creates a DBI connection object from a resource.

Usage:
PrestoResourceConnector

PostgresResourceConnector$createDBICConnection(resource)

Arguments:
resource  A valid resource object.

Returns:  A DBI connection object.

Method clone(): The objects of this class are cloneable with this method.

Usage:
PostgresResourceConnector$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

---

PrestoResourceConnector

Presto DBI resource connector

Description

Presto DBI resource connector
Presto DBI resource connector

Format

A R6 object of class PrestoResourceConnector

Details

Makes a Presto DBI connection from a resource description.

Super class

class resourcer::DBIResourceConnector <- PrestoResourceConnector

Methods

Public methods:

• PrestoResourceConnector$new()
• PrestoResourceConnector$isNew()
• PrestoResourceConnector$createDBICConnection()
• PrestoResourceConnector$clone()

Method new(): Creates a new PrestoResourceConnector instance.

Usage:
PrestoResourceConnector$new()

Returns:  A PrestoResourceConnector object.
Method isFor(): Check that the provided resource has a URL that locates a Presto object: the URL scheme must be "presto", "presto+http" or "presto+https".

Usage:
PrestoResourceConnector$isFor(resource)

Arguments:
resource The resource object to validate.

Returns: A logical.

Method createDBIConnection(): Creates a DBI connection object from a resource.

Usage:
PrestoResourceConnector$createDBIConnection(resource)

Arguments:
resource A valid resource object.

Returns: A DBI connection object.

Method clone(): The objects of this class are cloneable with this method.

Usage:
PrestoResourceConnector$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

RDataFileResourceClient

R data file resource client

Description

R data file resource client
R data file resource client

Format

A R6 object of class RDSFileResourceClient

Details

Connects to a R data file and loads it in a controlled environment.

Super classes

resourcer::ResourceClient -> resourcer::FileResourceClient -> RDataFileResourceClient
Methods

Public methods:

• RDataFileResourceClient$new()
• RDataFileResourceClient$asDataFrame()
• RDataFileResourceClient$getValue()
• RDataFileResourceClient$clone()

Method new(): Creates a new RDataFileResourceClient instance.

Usage:
RDataFileResourceClient$new(resource)

Arguments:
resource  A valid resource object.

Returns: A RDataFileResourceClient object.

Method asDataFrame(): Coerce the resource value extracted from the R data file to a data.frame.

Usage:
RDataFileResourceClient$asDataFrame(…)

Arguments:
… Additional parameters to as.data.frame (not used yet).

Returns: A data.frame.

Method getValue(): Get the resource value extracted from the R data file.

Usage:
RDataFileResourceClient$getValue(…)

Arguments:
… Additional parameters to get the value object (not used yet).

Returns: The resource value.

Method clone(): The objects of this class are cloneable with this method.

Usage:
RDataFileResourceClient$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.
**RDataFileResourceResolver**

*R data file Resource resolver*

---

**Description**

R data file Resource resolver

**Format**

A R6 object of class RDataFileResourceResolver

**Details**

The resource is a R data file and data format is the class of the symbol that will be loaded.

**Super class**

`resourcer::ResourceResolver` -&gt; RDataFileResourceResolver

**Methods**

**Public methods:**

- `RDataFileResourceResolver$isFor()
- `RDataFileResourceResolver$newClient()
- `RDataFileResourceResolver$clone()

**Method `isFor()`:** Check that the provided resource has a URL that locates a R data file: the resource can be accessed as a file and the resource URL path ends with ".rda" or ".rdata" (case ignored), or the resource format is prefixed with "r:" or "rda:" (a kind of namespace to qualify the R object class).

*Usage:*

```r
RDataFileResourceResolver$isFor(x)
```

*Arguments:*

- `x` The resource object to validate.

*Returns:* A logical.

**Method `newClient()`:** Creates a RDataFileResourceClient instance from provided resource.

*Usage:*

```r
RDataFileResourceResolver$newClient(x)
```

*Arguments:*

- `x` A valid resource object.

*Returns:* A RDataFileResourceClient object.
**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```r
RDataFileResourceResolver$clone(deep = FALSE)
```

*Arguments:*

- **deep**: Whether to make a deep clone.

---

**Description**

R object file resource client

**Format**

A R6 object of class RDSFileResourceClient

**Details**

Connects to a RDS file and loads the serialized object. Similar to the R data file resource, except that the RDS format stores a single R object.

**Super classes**

resourcer::ResourceClient -> resourcer::FileResourceClient -> RDSFileResourceClient

**Methods**

**Public methods:**

- `RDSFileResourceClient$new()`
- `RDSFileResourceClient$asDataFrame()`
- `RDSFileResourceClient$getValue()`
- `RDSFileResourceClient$clone()`

**Method** `new()`: Creates a new RDSFileResourceClient instance.

*Usage:*

```r
RDSFileResourceClient$new(resource)
```

*Arguments:*

- **resource**: A valid resource object.

*Returns*: A RDSFileResourceClient object.

**Method** `asDataFrame()`: Coerce the resource value extracted from the R object file to a `data.frame`. 
Usage: RDSFileResourceClient$asDataFrame(...)  
Arguments:  
... Additional parameters to as.data.frame (not used yet).

Returns: A data.frame.

Method `getValue()`: Get the resource value extracted from the R object file.

Usage: RDSFileResourceClient$getValue(...)  
Arguments:  
... Additional parameters to get the value object (not used yet).

Returns: The resource value.

Method `clone()`: The objects of this class are cloneable with this method.

Usage: RDSFileResourceClient$clone(deep = FALSE)  
Arguments:  
deep Whether to make a deep clone.

---

RDSFileResourceResolver

*R object file Resource resolver*

**Description**

R object file Resource resolver

R object file Resource resolver

**Format**

A R6 object of class RDSFileResourceResolver

**Details**

The resource is a RDS file.

**Super class**

`resourcer::ResourceResolver` -> `RDSFileResourceResolver`
**Methods**

**Public methods:**

- `RDSFileResourceResolver$isFor()`
- `RDSFileResourceResolver$newClient()`
- `RDSFileResourceResolver$clone()`

**Method isFor():** Check that the provided resource has a URL that locates a R object file: the resource can be accessed as a file and the resource URL path ends with ".rds" (case ignored), or the resource format is prefixed with "rds:" (a kind of namespace to qualify the R object class).

*Usage:*

`RDSFileResourceResolver$isFor(x)`

*Arguments:*

- `x` The resource object to validate.

*Returns:* A logical.

**Method newClient():** Creates a RDSFileResourceClient instance from provided resource.

*Usage:*

`RDSFileResourceResolver$newClient(x)`

*Arguments:*

- `x` A valid resource object.

*Returns:* A RDSFileResourceClient object.

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

`RDSFileResourceResolver$clone(deep = FALSE)`

*Arguments:*

- `deep` Whether to make a deep clone.

---

**registerDBIResourceConnector**

*Register a DBI resource connector*

**Description**

Maintain an list of DBIResourceConnectors that will be called when a new DBI resource connector is to be found.

**Usage**

`registerDBIResourceConnector(x)`

**Arguments**

- `x` The DBI resource connector object to register.
registerFileResourceGetter

*Register a file resource getter*

**Description**

Maintain an list of FileResourceGetters that will be called when a new file resource getter is to be found.

**Usage**

```r
registerFileResourceGetter(x)
```

**Arguments**

- `x`: The file resource getter object to register.

registerResourceResolver

*Register a resource resolver*

**Description**

Maintain an list of resource resolvers that will be called when a new resource is to be resolved.

**Usage**

```r
registerResourceResolver(x)
```

**Arguments**

- `x`: The resource resolver object to register.

**Examples**

```r
event::registerResourceResolver(MyFileFormatResourceResolver$new())
```
resolveResource

Find a resource resolver

Description
Find the resolver that will make a client from the provided resource object.

Usage
resolveResource(x)

Arguments
x
The resource object which corresponding resolver is to be found.

Value
The corresponding ResourceResolver object or NULL if none applies.

Examples

library(resourcer)
res <- newResource(
  name = "CNSIM1",
  url = "file:///data/CNSIM1.sav",
  format = "spss"
)
resolver <- resolveResource(res)

ResourceClient

Resource client

Description
Resource client
Resource client

Format
A R6 object of class ResourceClient

Details
Helper class for connecting to a resource data store or a computation unit.
Methods

Public methods:

• ResourceClient$new()
• ResourceClient$getResource()
• ResourceClient$getConnection()
• ResourceClient$downloadFile()
• ResourceClient$asDataFrame()
• ResourceClient$asTbl()
• ResourceClient$exec()
• ResourceClient$close()
• ResourceClient$clone()

Method new(): Creates a ResourceClient instance.

Usage:
ResourceClient$new(resource)

Arguments:
resource The resource object to be interpreted.

Returns: A ResourceClient object.

Method getResource(): Get the resource object.

Usage:
ResourceClient$getResource()

Returns: The resource object.

Method getConnection(): Get the implementation-specific object that connects to the resource

Usage:
ResourceClient$getConnection()

Returns: The connection object.

Method downloadFile(): Stub function to be implemented by subclasses if relevant. Get the resource as a local file.

Usage:
ResourceClient$downloadFile(...)

Arguments:
... Additional parameters.

Returns: The path to the local file.

Method asDataFrame(): Stub function to be implemented by subclasses if relevant. Coerce the resource as a data.frame.

Usage:
ResourceClient$asDataFrame(...)

Arguments:
... Additional parameters.

Returns: A data.frame object (can also be a tibble).

**Method** `asTbl()`: Stub function to be implemented by subclasses if relevant. Coerce the resource as a dplyr's tbl.

*Usage:*

```r
ResourceClient$asTbl(...) 
```

*Arguments:*

... Additional parameters.

*Returns: A dplyr's tbl object.*

**Method** `exec()`: Stub function to be implemented by subclasses if relevant. Executes a command on a computation resource.

*Usage:*

```r
ResourceClient$exec(...) 
```

*Arguments:*

... Additional parameters that will represent the command to execute.

*Returns: A command execution result object.*

**Method** `close()`: Silently closes the connection to the resource

*Usage:*

```r
ResourceClient$close() 
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```r
ResourceClient$clone(deep = FALSE) 
```

*Arguments:*

depth Whether to make a deep clone.

---

<table>
<thead>
<tr>
<th>ResourceResolver</th>
<th>Resource resolver</th>
</tr>
</thead>
</table>

**Description**

Resource resolver

**Format**

A R6 object of class ResourceResolver

**Details**

Helper class for building a Client that implements the access to the data or the computation unit.
Methods

Public methods:

- `ResourceResolver$new()`
- `ResourceResolver$isFor()`
- `ResourceResolver$newClient()`
- `ResourceResolver$clone()`

**Method new():** Creates a new `ResourceResolver` instance.

*Usage:*

`ResourceResolver$new()`

*Returns:* A `ResourceResolver` object.

**Method isFor():** Check that the provided object is of class "resource".

*Usage:*

`ResourceResolver$isFor(x)`

*Arguments:*

- `x` The resource object to evaluate.

*Returns:* A logical.

**Method newClient():** Stub function to be implemented by subclasses. Makes an object which class inherits from `ResourceClient`.

*Usage:*

`ResourceResolver$newClient(x)`

*Arguments:*

- `x` The resource object to evaluate.

*Returns:* The `ResourceClient` object that will access the provided resource.

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

`ResourceResolver$clone(deep = FALSE)`

*Arguments:*

- `deep` Whether to make a deep clone.
ScpFileResourceGetter  
SCP file resource getter

Description

SCP file resource getter
SCP file resource getter

Format

A R6 object of class ScpFileResourceGetter

Details

Access a file that is stored on a server accessible through SSH. Credentials apply.

Super class

resourcer::FileResourceGetter -> ScpFileResourceGetter

Methods

Public methods:

• ScpFileResourceGetter$new()
• ScpFileResourceGetter$isFor()
• ScpFileResourceGetter$downloadFile()
• ScpFileResourceGetter$clone()

Method $new(): Creates a ScpFileResourceGetter instance.

Usage:
ScpFileResourceGetter$new()


Method $isFor(): Check that the provided resource is a file accessible by SCP: the resource
URL scheme must be "scp".

Usage:
ScpFileResourceGetter$isFor(resource)

Arguments:
resource  The resource object to evaluate.

Returns: A logical.

Method $downloadFile(): Download the file described by the resource over an SSH connection.

Usage:
ScpFileResourceGetter$downloadFile(resource, ...)
Arguments:
resource  The resource that identifies the file.
...  Additional parameters (not used).
Returns:  The "resource.file" object.

Method clone(): The objects of this class are cloneable with this method.
Usage:
ScpFileResourceGetter$clone(deep = FALSE)
Arguments:
deep  Whether to make a deep clone.

ShellResourceClient  Shell resource client

Description
Shell resource client
Shell resource client

Format
A R6 object of class ShellResourceClient

Details
Executes local system shell commands.

Super classes
resourcer::ResourceClient -> resourcer::CommandResourceClient -> ShellResourceClient

Methods
Public methods:
• ShellResourceClient$new()
• ShellResourceClient$getAllowedCommands()
• ShellResourceClient$copyFile()
• ShellResourceClient$exec()
• ShellResourceClient$clone()

Method new(): Create a ShellResourceClient instance. This client will interact with a computation resource using shell commands.
Usage:
ShellResourceClient$new(resource)
Method `getAllowedCommands()`: Get the command names that can be executed.

Usage:
ShellResourceClient$getAllowedCommands()

Returns: A character vector

Method `copyFile()`: Copy one or more files (wildcard * is supported in the file name (which can be a directory))

Usage:
ShellResourceClient$copyFile(file, to = ".", verbose = FALSE)

Arguments:
file The file to copy.
to The copy destination.
verbose If TRUE, details the file operations on the console output.

Returns: The path to the files having been copied.

Method `exec()`: Executes a shell command in the working directory specified in the resource’s URL.

Usage:
ShellResourceClient$exec(command, params = NULL, test = FALSE)

Arguments:
command The command name.
params A character vector of arguments to pass.
test If TRUE, the command is printed but not executed (for debugging).

Returns: The command execution result object.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
ShellResourceClient$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.
ShellResourceResolver  Shell Resource resolver

Description

Shell Resource resolver
Shell Resource resolver

Format

A R6 object of class ShellResourceResolver

Details

The resource is a computation unit, accessible by issuing local system commands, i.e. which URL scheme is "sh".

Super class

resourcer::ResourceResolver -> ShellResourceResolver

Methods

Public methods:

•  ShellResourceResolver$isNewFor()
•  ShellResourceResolver$newClient()
•  ShellResourceResolver$clone()

Method isFor(): Check that the provided resource is a computation resource accessible by shell commands. The resource URL scheme must be "sh" or "shell".

Usage:
ShellResourceResolver$isNewFor(x)

Arguments:

x The resource object.

Returns: A logical.

Method newClient(): Create a ShellResourceClient instance from the provided resource.

Usage:
ShellResourceResolver$newClient(x)

Arguments:

x A valid resource object.

Returns: A ShellResourceClient object.

Method clone(): The objects of this class are cloneable with this method.
SparkResourceConnector

Usage:
ShellResourceResolver$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

Description
Apache Spark DBI resource connector

Format
A R6 object of class SparkResourceConnector

Details
Makes a Apache Spark connection object, that is also a DBI connection object, from a resource description.

Super class
resourcer::DBIResourceConnector -> SparkResourceConnector

Methods
Public methods:
• SparkResourceConnector$new()
• SparkResourceConnector$isFor()
• SparkResourceConnector$createDBIConnection()
• SparkResourceConnector$closeDBIConnection()
• SparkResourceConnector$clone()

Method new(): Create a SparkResourceConnector instance.
Usage:
SparkResourceConnector$new()

Returns: A SparkResourceConnector object.

Method isFor(): Check if the provided resource applies to a Apache Spark server. The resource URL scheme must be one of "spark", "spark+http" or "spark+https".
Usage:
SparkResourceConnector$isFor(resource)
Arguments:
resource  The resource object to validate.

Returns:  A logical.

Method `createDBIConnection()`: Creates a DBI connection object from a Apache Spark resource.

Usage:
`SparkResourceConnector$createDBIConnection(resource)`

Arguments:
resource  A valid resource object.

Returns:  A DBI connection object.

Method `closeDBIConnection()`: Close the DBI connection to Apache Spark.

Usage:
`SparkResourceConnector$closeDBIConnection(conn)`

Arguments:
conn  A DBI connection object.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
`SparkResourceConnector$clone(deep = FALSE)`

Arguments:
deep  Whether to make a deep clone.

---

**SQLResourceClient**  
**SQL database resource client**

**Description**
SQL database resource client  
SQL database resource client

**Format**
A R6 object of class SQLResourceClient

**Details**
Resource client that connects to a SQL database supported by DBI.

**Super class**
`resourcer::ResourceClient` -> SQLResourceClient
Methods

Public methods:

• `SQLResourceClient$new()`
• `SQLResourceClient$getConnection()`
• `SQLResourceClient$asDataFrame()`
• `SQLResourceClient$asTbl()`
• `SQLResourceClient$close()`
• `SQLResourceClient$clone()`

Method `new()`: Creates a SQLResourceClient from a resource.

Usage:
`SQLResourceClient$new(resource, dbi.connector = NULL)`

Arguments:
resource The resource object.
dbi.connector An optional DBIResourceConnector object. If not provided, it will be looked up in the DBIResourceConnector registry.

Returns: The SQLResourceClient object.

Method `getConnection()`: Get or create the DBI connection object that will access the resource.

Usage:
`SQLResourceClient$getConnection()`

Returns: The DBI connection object.

Method `asDataFrame()`: Coerce the SQL table to a data.frame.

Usage:
`SQLResourceClient$asDataFrame(...)`

Arguments:
... Additional parameters (not used).

Returns: A data.frame (more specifically a tibble).

Method `asTbl()`: Get the SQL table as a dplyr’s tbl.

Usage:
`SQLResourceClient$asTbl()`

Returns: A dplyr’s tbl object.

Method `close()`: Silently close the DBI connection.

Usage:
`SQLResourceClient$close()`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
`SQLResourceClient$clone(deep = FALSE)`

Arguments:
deep Whether to make a deep clone.
**SQLResourceResolver**  
*SQL Database Resource resolver*

**Description**

SQL Database Resource resolver

**Format**

A R6 object of class SQLResourceResolver

**Details**

The resource is SQL database.

**Super class**

```
resourcer::ResourceResolver -> SQLResourceResolver
```

**Methods**

**Public methods:**

- SQLResourceResolver$isNewFor()
- SQLResourceResolver$newClient()
- SQLResourceResolver$clone()

**Method newClient():** Creates a SQLResourceClient instance from provided resource.

*Usage:*

```
SQLResourceResolver$newClient(x)
```

*Arguments:*

- `x` A valid resource object.

*Returns:* A SQLResourceClient object.

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

```
SQLResourceResolver$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.
**SshResourceClient**

**SSH resource client**

---

**Description**

SSH resource client

**Format**

A R6 object of class SshResourceClient

**Details**

Connects to a SSH server.

**Super classes**

`resourcer::ResourceClient -> resourcer::CommandResourceClient -> SshResourceClient`

**Methods**

**Public methods:**

- `SshResourceClient$new()`
- `SshResourceClient$getAllowedCommands()`
- `SshResourceClient$getConnection()`
- `SshResourceClient$downloadFile()`
- `SshResourceClient$uploadFile()`
- `SshResourceClient$tempDir()`
- `SshResourceClient$removeTempDir()`
- `SshResourceClient$exec()`
- `SshResourceClient$close()`
- `SshResourceClient$clone()`

**Method new():** Create a SshResourceClient instance. This client will interact with a computation resource using ssh commands.

*Usage:*

```
SshResourceClient$new(resource)
```

*Arguments:*

- `resource` The computation resource.

*Returns:* The SshResourceClient object.

**Method getAllowedCommands():** Get the command names that can be executed.

*Usage:*

```
SshResourceClient$getAllowedCommands()
```
SshResourceClient$getAllowedCommands()

*Returns:* A character vector

**Method getConnection():** Get or create the SSH connection object, for raw interaction.

*Usage:*
SshResourceClient$getConnection()

*Returns:* The SSH connection object.

**Method downloadFile():** Download one or more files (wildcard * is supported in the file name (which can be a directory))

*Usage:*
SshResourceClient$downloadFile(file, to = ".", verbose = FALSE)

*Arguments:*
- file: The file path(s) to download, either absolute or relative to the working directory.
- to: The download destination.
- verbose: If TRUE, details the file operations on the console output.

*Returns:* The paths of the files having been downloaded.

**Method uploadFile():** Upload one or more files

*Usage:*
SshResourceClient$uploadFile(file, to = ".", verbose = FALSE)

*Arguments:*
- file: The file or vector of files to upload.
- to: The upload destination, either absolute or relative to working directory.
- verbose: If TRUE, details the file operations on the console output.

*Returns:* The paths of the files having been uploaded.

**Method tempDir():** Get connection's temporary directory in the remote server, create it if it does not exists.

*Usage:*
SshResourceClient$tempDir()

*Returns:* The path to the temporary directory.

**Method removeTempDir():** Remove the connection's temporary directory from the remote server, if it was defined.

*Usage:*
SshResourceClient$removeTempDir()

**Method exec():** Executes a ssh command.

*Usage:*
SshResourceClient$exec(command, params = NULL, test = FALSE)

*Arguments:*
- command: The command name.
params A character vector of arguments to pass.
test If TRUE, the command is printed but not executed (for debugging).

*Returns:* The command execution result object.

**Method** `close()`: Close the SSH connection.

*Usage:*
`SshResourceClient$close()`

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*
`SshResourceClient$clone(deep = FALSE)`

*Arguments:*
  *deep* Whether to make a deep clone.

---

### Description

**SSH Resource resolver**

**Format**

A R6 object of class `SshResourceResolver`

**Details**

The resource is a computation unit, accessible through SSH, i.e. which URL scheme is "ssh".

**Super class**

`resourcer::ResourceResolver` -> `SshResourceResolver`

**Methods**

**Public methods:**

- `SshResourceResolver$isFor()`
- `SshResourceResolver$newClient()`
- `SshResourceResolver$clone()`

**Method** `isFor()`: Check that the provided resource is a computation resource accessible by ssh commands. The resource URL scheme is expected to be "ssh".

*Usage:*
`SshResourceResolver$isFor(x)`
TidyFileResourceClient

Arguments:
  x  The resource object.

Returns: A logical.

Method newClient(): Create a SshResourceClient instance from the provided resource.

Usage:
  SshResourceResolver$newClient(x)

Arguments:
  x  A valid resource object.

Returns: A SshResourceClient object.

Method clone(): The objects of this class are cloneable with this method.

Usage:
  SshResourceResolver$clone(deep = FALSE)

Arguments:
  deep  Whether to make a deep clone.

Description

Tidy file resource client
Tidy file resource client

Format

A R6 object of class TidyFileResourceClient

Details

Connects to a file and use one of the tidyverse reader.

Super classes

resourcer::ResourceClient -> resourcer::FileResourceClient -> TidyFileResourceClient
Methods

Public methods:

• TidyFileResourceClient$new()
• TidyFileResourceClient$asDataFrame()
• TidyFileResourceClient$clone()

Method new(): Create a TidyFileResourceClient instance.

Usage:
TidyFileResourceClient$new(resource)

Arguments:
resource  A valid resource object.

Returns: A TidyFileResourceClient object.

Method asDataFrame(): Coerce the resource value extracted from the file in tidy format to a data.frame.

Usage:
TidyFileResourceClient$asDataFrame(...)

Arguments:
... Additional parameters to as.data.frame (not used yet).

Returns: A data.frame (more specifically a tibble).

Method clone(): The objects of this class are cloneable with this method.

Usage:
TidyFileResourceClient$clone(deep = FALSE)

Arguments:
depth  Whether to make a deep clone.

TidyFileResourceResolver

Tidy file Resource resolver

Description

Tidy file Resource resolver
Tidy file Resource resolver

Format

A R6 object of class TidyFileResourceResolver
Details

The resource is a file and data format is handled by a reader from tidyverse. The data format is one of: csv (comma delimiter), csv2 (semicolon delimiter), tsv (tab delimiter), ssv (space delimiter), delim (delim parameter to be specified in the URL, default is space char), spss, sav, por, stata, dta, sas, xpt, excel, xls, xlsx.

Methods

$new() Create new TidyFileResourceResolver instance. $isFor(x) Get a logical that indicates that the resolver is applicable to the provided resource object. $newClient() Make a client for the provided resource.

Super class

resourcer::ResourceResolver -> TidyFileResourceResolver

Methods

Public methods:

- TidyFileResourceResolver$isFor()
- TidyFileResourceResolver$newClient()
- TidyFileResourceResolver$clone()

Method isFor(): Check that the provided resource has a URL that locates a tidy data file: the resource can be accessed as a file and the resource format is one of "csv", "csv2", "tsv", "delim", "ssv", "spss", "sav", "por", "stata", "dta", "sas", "xpt", "excel", "xls" or "xlsx" (case is ignored).

Usage:

TidyFileResourceResolver$isFor(x)

Arguments:

- x The resource object to validate.

Returns: A logical.

Method newClient(): Creates a TidyFileResourceClient instance from provided resource.

Usage:

TidyFileResourceResolver$newClient(x)

Arguments:

- x A valid resource object.

Returns: A TidyFileResourceClient object.

Method clone(): The objects of this class are cloneable with this method.

Usage:

TidyFileResourceResolver$clone(deep = FALSE)

Arguments:

- deep Whether to make a deep clone.
unregisterDBIResourceConnector

Remove a DBI resource connector from the registry

Description

Remove any DBI resource connectors with the provided class name.

Usage

unregisterDBIResourceConnector(x)

Arguments

x The DBI resource connector class name to unregister.

unregisterFileResourceGetter

Remove a file resource getter from the registry

Description

Remove any file resource getters with the provided class name.

Usage

unregisterFileResourceGetter(x)

Arguments

x The file resource getter class name to unregister.
unregisterResourceResolver

Remove a resource resolver from the registry

Description

Remove any resolvers with the provided class name.

Usage

unregisterResourceResolver(x)

Arguments

x The resource resolver class name to unregister.

Examples

## Not run:
resourcer::unregisterResourceResolver("MyFileFormatResourceResolver")

## End(Not run)
<table>
<thead>
<tr>
<th>Function/Class</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.data.frame.resource</td>
<td>3</td>
</tr>
<tr>
<td>as.data.frame.ResourceClient</td>
<td>3</td>
</tr>
<tr>
<td>as.resource.data.frame</td>
<td>4</td>
</tr>
<tr>
<td>as.resource.object</td>
<td>4</td>
</tr>
<tr>
<td>as.resource.tbl</td>
<td>5</td>
</tr>
<tr>
<td>CommandResourceClient</td>
<td>5</td>
</tr>
<tr>
<td>DBIResourceConnector</td>
<td>6</td>
</tr>
<tr>
<td>FileResourceClient</td>
<td>8</td>
</tr>
<tr>
<td>FileResourceGetter</td>
<td>9</td>
</tr>
<tr>
<td>findDBIResourceConnector</td>
<td>11</td>
</tr>
<tr>
<td>findFileResourceGetter</td>
<td>11</td>
</tr>
<tr>
<td>getDBIResourceConnectors</td>
<td>12</td>
</tr>
<tr>
<td>getFileResourceGetters</td>
<td>12</td>
</tr>
<tr>
<td>getResourceResolvers</td>
<td>12</td>
</tr>
<tr>
<td>GridFsFileResourceGetter</td>
<td>13</td>
</tr>
<tr>
<td>HttpFileResourceGetter</td>
<td>14</td>
</tr>
<tr>
<td>LocalFileResourceGetter</td>
<td>15</td>
</tr>
<tr>
<td>MariaDBResourceConnector</td>
<td>17</td>
</tr>
<tr>
<td>newResource</td>
<td>18</td>
</tr>
<tr>
<td>newNoSQLResourceClient</td>
<td>19</td>
</tr>
<tr>
<td>NoSQLResourceClient</td>
<td>19</td>
</tr>
<tr>
<td>NoSQLResourceResolver</td>
<td>21</td>
</tr>
<tr>
<td>OpalFileResourceGetter</td>
<td>22</td>
</tr>
<tr>
<td>PostgresResourceConnector</td>
<td>24</td>
</tr>
<tr>
<td>PrestoResourceConnector</td>
<td>25</td>
</tr>
<tr>
<td>RDataFileResourceClient</td>
<td>26</td>
</tr>
<tr>
<td>RDataFileResourceResolver</td>
<td>28</td>
</tr>
<tr>
<td>RDSFileResourceClient</td>
<td>29</td>
</tr>
<tr>
<td>RDSFileResourceResolver</td>
<td>30</td>
</tr>
<tr>
<td>registerDBIResourceConnector</td>
<td>31</td>
</tr>
<tr>
<td>registerDBIResourceConnector</td>
<td>32</td>
</tr>
<tr>
<td>registerResourceResolver</td>
<td>32</td>
</tr>
<tr>
<td>resolveResource</td>
<td>33</td>
</tr>
<tr>
<td>ResourceClient</td>
<td>33</td>
</tr>
<tr>
<td>resourcer::CommandResourceClient</td>
<td>38, 45</td>
</tr>
<tr>
<td>resourcer::DBIResourceConnector</td>
<td>17, 24, 25, 41</td>
</tr>
<tr>
<td>resourcer::FileResourceClient</td>
<td>26, 29, 48</td>
</tr>
<tr>
<td>resourcer::FileResourceGetter</td>
<td>13, 14, 16, 22, 37</td>
</tr>
<tr>
<td>resourcer::ResourceClient</td>
<td>5, 8, 20, 26, 29, 38, 42, 45, 48</td>
</tr>
<tr>
<td>resourcer::ResourceResolver</td>
<td>21, 28, 30, 40, 44, 47, 50</td>
</tr>
<tr>
<td>ResourceResolver</td>
<td>35</td>
</tr>
<tr>
<td>ScpFileResourceGetter</td>
<td>37</td>
</tr>
<tr>
<td>ShellResourceClient</td>
<td>38</td>
</tr>
<tr>
<td>ShellResourceResolver</td>
<td>40</td>
</tr>
<tr>
<td>SparkResourceConnector</td>
<td>41</td>
</tr>
<tr>
<td>SQLResourceClient</td>
<td>42</td>
</tr>
<tr>
<td>SQLResourceResolver</td>
<td>44</td>
</tr>
<tr>
<td>SshResourceClient</td>
<td>45</td>
</tr>
<tr>
<td>SshResourceResolver</td>
<td>47</td>
</tr>
<tr>
<td>TidyFileResourceClient</td>
<td>48</td>
</tr>
<tr>
<td>TidyFileResourceResolver</td>
<td>49</td>
</tr>
<tr>
<td>unregisterDBIResourceConnector</td>
<td>51</td>
</tr>
<tr>
<td>unregisterFileResourceGetter</td>
<td>51</td>
</tr>
<tr>
<td>unregisterResourceResolver</td>
<td>52</td>
</tr>
</tbody>
</table>