Package 'srcr'

October 14, 2022

Type Package

Title Simplify Connections to Database Sources

Version 1.1.0

Maintainer Charles Bailey <baileyc@chop.edu>

Description Connecting to databases requires boilerplate code to specify
connection parameters and to set up sessions properly with the DBMS.
This package provides a simple tool to fill two purposes: abstracting
connection details, including secret credentials, out of your source
code and managing configuration for frequently-used database connections
in a persistent and flexible way, while minimizing requirements on the
runtime environment.

License Artistic-2.0

Encoding UTF-8

ByteCompile TRUE

Imports DBI, dplyr, jsonlite, utils

Suggests knitr, rmarkdown, RSQLite, withr

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Charles Bailey [aut, cre],
Hanieh Razzaghi [aut]

Repository CRAN

Date/Publication 2021-10-30 20:00:02 UTC

R topics documented:

---

find_config_files   *Locate candidate configuration files*

Description

Given vectors of directories, basenames, and suffices, combine them to find existing files.

Usage

```
find_config_files(
  basenames = .basename.defaults(),
  dirs = .dir.defaults(),
  suffices = .suffix.defaults()
)
```

Arguments

basenames	A vector of file names (without directory or file type) to use in searching for configuration files.
dirs	A vector of directory names to use in searching for configuration files.
suffices	A vector of suffices (file "type"s) to use in searching for the configuration file.

Details

This function is intended to support a variety of installation patterns, so it attempts to be flexible in looking for configuration files. First, environment variables of the form *basename*_CONFIG, where *basename* is the uppercase form of each candidate basename, are examined to see whether any translate to a file path.

Following this, the path name parts supplied as arguments are used to build potential file names. If dirs is not specified, the following directories are checked by default:

1. the user's $HOME directory
2. the directory named .srcr (no leading . on Windows) under $HOME
3. the directory in which the executing script is located
4. the directory in which the calling function's calling function's source file is located (typically an application-level library). For example, if the function my_setup() calls srcr(), which in turn calls find_config_files(), then the directory of the file containing my_setup() will be tried.
5. the directory in which the calling function's source file is located (typically a utility function, such as srcr())

Note that the current working directory is not part of the search by default. This is done to limit the potential for accidentally introducing (potentially harmful) configuration files by setting the working directory.

In each location, the file names given in basenames are checked; if none are specified, several default file names are tried:

1. the name of the calling function's source file

2. the name of the executing script

3. the directory in which the calling function's calling function's source file is located (typically an application-level library). For example, if the function my_setup() calls srcr(), which in turn calls find_config_files(), then the name of the file containing my_setup() will be tried.

The suffices (file "type"s) of .json, .conf, and nothing, are tried with each candidate path; you may override this default by using the suffices parameter. Finally, in order to accommodate the Unix tradition of "hidden" configuration files, each basename is prefixed with a period before trying the basename alone.

Value

A vector of path specifications, or an empty vector if none are found.

Examples

```
## Not run:
find_config_files() # All defaults
find_config_files(dirs = c(file.path(Sys.getenv('HOME'),'etc'),
                          '/usr/local/etc', '/etc'),
                  basenames = c('my_app'),
                  suffices = c('.conf', '.rc'))

## End(Not run)
```

---

srcr   *Connect to database using config file*

Description

Set up a or DBI or legacy dplyr database connection using information from a JSON configuration file, and return the connection.

Connecting to databases requires boilerplate code to specify connection parameters and to set up sessions properly with the DBMS. This package provides a simple tool to fill two purposes: abstracting connection details, including secret credentials, out of your source code and managing configuration for frequently-used database connections in a persistent and flexible way, while minimizing requirements on the runtime environment.

Usage

```
srcr(
  basenames = NA,
  dirs = NA,
  suffices = NA,
  paths = NA,
```

```
  config = NA,
  allow_post_connect = getOption("srcr.allow_post_connect", c())
)
```

Arguments

basenames	A vector of file names (without directory or file type) to use in searching for configuration files.
dirs	A vector of directory names to use in searching for configuration files.
suffices	A vector of suffices (file "type"s) to use in searching for the configuration file.
paths	A vector of full path names for the configuration file. If present, only these paths are checked; `find_config_files()` is not called.
config	A list containing the configuration data, to be used instead of reading a configuration file, should you wish to skip that step.
allow_post_connect	
	A vector specifying what session setup you will permit after the connection is established. If any element of the vector is sql, then the post_connect_sql section of the configuration file is executed. Similarly, if any element is fun, then the post_connect_fun section of the config file is executed (after post_connect_sql, if both are present and allowed).

Details

The configuration file must provide all of the information necessary to set up the DBI connection or dplyr src. Given the variety of ways a data source can be specified, the JSON must be a hash containing at least two elements:

- The src_name key points to a string containing name of a DBI driver method (e.g. SQLite), as one might pass to `DBI::dbDriver()`, or an old-style dplyr function that sets up the data source (e.g. `dplyr::src_postgres()`. If the value associated with src_name begins with 'src_', it is taken as the latter, otherwise it is taken as the former. In this case, an attempt will be made to load the appropriate DBI-compliant database library (e.g. RSQLite for the above example) if it hasn't already been loaded.

- The src_args key points to a nested hash, whose keys are the arguments to that function, and whose values are the argument values.

To locate the necessary configuration file, you can use all of the arguments taken by `find_config_files()`, but remember that the contents of the file must be JSON, regardless of the file's name. Alternatively, if paths is present, only the specified paths are checked. The first file that exists, is readable, and evaluates as legal JSON is used as the source of configuration data.

If your deployment strategy does not make use of configuration files (e.g. you access configuration data via a web service or similar API), you may also pass a list containing the configuration data directly via the config parameter. In this case, no configuration files are used.

Once the connection is established, the post_connect_sql and post_connect_fun elements of the configuration data can be used to perform additional processing to set session characteristics, roles, etc. However, because this entails the configuration file providing code that you won't see prior to runtime, you need to opt in to these features. You can make this choice globally by setting the srcr.allow_post_connect option via `base::options()`.

Value

A database connection. The specific class of the object is determined by the src_name in the configuration data.

Examples

```
## Not run:
# Search all the (filename-based) defaults
srcr()

# "The usual"
srcr('myproj_prod')

# Look around
srcr(dirs = c(Sys.getenv('PROJ_CONF_DIR'), 'var/lib', getwd()),
     basenames = c('myproj', Sys.getenv('PROJ_NAME')) )

# No defaults
srcr(paths = c('/path/to/known/config.json'))
srcr(config =
       list(src_name = 'Postgres',
            src_args = list(host = 'my.host', dbname = 'my_db', user = 'me'),
            post_connect_sql = 'set role project_role;'))

## End(Not run)
```

Index