Package: ProjectTemplate (via r-universe)

August 30, 2024

Type Package

Title Automates the Creation of New Statistical Analysis Projects

Version 0.11.0

Date 2024-07-01

Description Provides functions to automatically build a directory structure for a new R project. Using this structure, 'ProjectTemplate' automates data loading, preprocessing, library importing and unit testing.

License GPL-3 | file LICENSE

Language en-US

LazyLoad yes

Roxygen list()

Encoding UTF-8

Depends R (>= 2.7), digest, tibble

Imports methods

Suggests foreign, feather, reshape, plyr, formatR, qs, stringr, ggplot2, lubridate, log4r (>= 0.1-5), DBI, RMySQL, RSQLite, gdata, RODBC, RJDBC, readxl, xlsx, tuneR, pixmap, data.table, RPostgreSQL, GetoptLong, whisker, testthat (>= 3.0.0), reticulate

URL http://projecttemplate.net

BugReports https://github.com/KentonWhite/ProjectTemplate/issues

Collate 'ProjectTemplate-package.R' 'add.config.R' 'preinstalled.readers.R' 'add.extension.R' 'addins.R' 'arff.reader.R' 'get.project.R' 'cache.R' 'cache.name.R' 'cache.project.R' 'clean.variable.name.R' 'clear.R' 'clear.cache.R' 'translate.dcf.R' 'config.R' 'create.project.R' 'create.project.rstudio.R' 'create.template.R' 'csv.reader.R' 'csv2.reader.R' 'db.reader.R' 'dbf.reader.R' 'epiinfo.reader.R' 'feather.reader.R' 'file.reader.R' 'list.data.R' 'load.project.R' 'migrate.project.R' 'migrate.template.R'

'mp3.reader.R' 'mtp.reader.R' 'octave.reader.R' 'ppm.reader.R'
'project.config.R' 'r.reader.R' 'rdata.reader.R' 'rds.reader.R'
'reload.project.R' 'require.package.R' 'run.project.R'
'show.project.R' 'spss.reader.R' 'sql.reader.R'
'stata.reader.R' 'stopifnotproject.R' 'stub.tests.R'
'systat.reader.R' 'test.project.R' 'tsv.reader.R'
'url.reader.R' 'wsv.reader.R' 'xls.reader.R' 'xlsx.reader.R'
'xport.reader.R'

RoxygenNote 7.3.1

Config/testthat/edition 3

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

RemoteUrl https://github.com/kentonwhite/projecttemplate

RemoteRef HEAD

RemoteSha 8dcfdc3bff89ea758709242995df4416528b147c

Contents

`.add.extension` *Associate a reader function with an extension.*

Description

This function will associate an extension with a custom reader function.

Usage

```
.add.extension(extension, reader)
```

Arguments

`extension`	The extension of the new data file.
`reader`	The function to use when reading the data file. It should accept three arguments: `data.file`, `filename` and `variable.name` (in that order). The function should read the contents of the file `filename`, and save it into the workspace under the name `variable.name`. The `data.file` argument is just a relative file name and can be ignored.

Value

No value is returned; this function is called for its side effects.

Warning

This interface should not be considered as stable and is likely to be replaced by a different mechanism in a forthcoming version of this package.

See Also

preinstalled.readers

Examples

```
## Not run: .add.extension('foo', foo.reader)
```

add.config *Add project specific config to the global config*

Description

Enables project specific configuration to be added to the global config object. The allowable format is key value pairs which are appended to the end of the `config` object, which is accessible from the global environment.

Usage

```
add.config(..., apply.override = FALSE)
```

Arguments

`...`	A series of key-value pairs containing the configuration. The key is the name that gets added to the config object. These can be overridden at load time through the `...` argument to `load.project`.
`apply.override`	A boolean indicating whether overrides should be applied. This can be used to add a setting disregarding arguments to `load.project`

Details

Once defined, the value can be accessed from any `ProjectTemplate` script by referencing `config$my_project_var`.

Examples

```
library('ProjectTemplate')
## Not run:
add.config(
    keep_bigdata=TRUE,     # Whether to keep the big data file in memory
    parse=7                # number of fields to parse
)

if (config$keep_bigdata) ...

## End(Not run)
```

cache *Cache a data set for faster loading.*

Description

This function will store a copy of the named data set in the `cache` directory. This cached copy of the data set will then be given precedence at load time when calling `load.project`. Cached data sets are stored as `.RData` or optionally as `.qs` files.

Usage

```
cache(variable = NULL, CODE = NULL, depends = NULL, ...)
```

Arguments

variable	A character string containing the name of the variable to be saved. If the CODE parameter is defined, it is evaluated and saved, otherwise the variable with that name in the global environment is used.
CODE	A sequence of R statements enclosed in {..} which produce the object to be cached. Requires suggested package formatR
depends	A character vector of other global environment objects that the CODE depends upon. Caching will be forced if those objects have changed since last caching
...	Additional arguments passed on to save or optionally to qsave. See project.config for further information.

Details

Usually you will want to cache datasets during munging. This can be the raw data just loaded, or it can be the result of further processing during munge. Either way, it can take a while to cache large variables, so cache will only cache when it needs to. The `clear.cache("variable")` command can be run to flush individual items from the cache.

Calling `cache()` with no arguments returns the current status of the cache.

Value

No value is returned; this function is called for its side effects.

See Also

qsave, project.config

Examples

```
library('ProjectTemplate')
## Not run: create.project('tmp-project')

setwd('tmp-project')

dataset1 <- 1:5
cache('dataset1')

setwd('..')
unlink('tmp-project')
## End(Not run)
```

cache.project *Cache a project's data sets in binary format.*

Description

This function will cache all of the data sets that were loaded by the load.project function in a binary format that is easier to load quickly. This is particularly useful for data sets that you've modified during a slow munging process that does not need to be repeated.

Usage

```
cache.project()
```

Value

No value is returned; this function is called for its side effects.

See Also

create.project, load.project, get.project, show.project

Examples

```
library('ProjectTemplate')
## Not run: load.project()

cache.project()
## End(Not run)
```

clear *Clear objects from the global environment*

Description

This function removes specific (or all by default) named objects from the global environment. If used within a ProjectTemplate project, then any variables defined in the config$sticky_variables will remain.

Usage

```
clear(..., keep = c(), force = FALSE)
```

Arguments

...	A sequence of character strings of the objects to be removed from the global environment. If none given, then all items except those in keep will be deleted. This includes items beginning with .
keep	A character vector of variables that should remain in the global environment
force	If TRUE, then variables will be deleted even if specified in keep or config$sticky_variables

Value

The variables kept and removed are reported

Examples

```
library('ProjectTemplate')
## Not run:
clear("x", "y", "z")
clear(keep="a")
clear()

## End(Not run)
```

---

clear.cache *Clear data sets from the cache*

Description

This function remove specific (or all by default) named data sets from the cache directory. This will force that data to be read in from the data directory next time load.project is called.

Usage

```
clear.cache(...)
```

Arguments

...	A sequence of character strings of the variables to be removed from the cache. If none given, then all items in the cache will be removed.

Value

Success or failure is reported

Examples

```
library('ProjectTemplate')
## Not run:
clear.cache("x", "y", "z")

## End(Not run)
```

`create.project` *Create a new project.*

Description

This function will create all of the scaffolding for a new project. It will set up all of the relevant directories and their initial contents. For those who only want the minimal functionality, the `template` argument can be set to `minimal` to create a subset of ProjectTemplate's default directories. For those who want to dump all of ProjectTemplate's functionality into a directory for extensive customization, the dump argument can be set to `TRUE`.

Usage

```
create.project(
  project.name = "new-project",
  template = "full",
  dump = FALSE,
  merge.strategy = c("require.empty", "allow.non.conflict"),
  rstudio.project = FALSE
)
```

Arguments

`project.name`	A character vector containing the name for this new project. Must be a valid directory name for your file system.
`template`	A character vector containing the name of the template to use for this project. By default a `full` and `minimal` template are provided, but custom templates can be created using `create.template`.
`dump`	A boolean value indicating whether the entire functionality of ProjectTemplate should be written out to flat files in the current project.
`merge.strategy`	What should happen if the target directory exists and is not empty? If `"force.empty"`, the target directory must be empty; if `"allow.non.conflict"`, the method succeeds if no files or directories with the same name exist in the target directory.
`rstudio.project`	A boolean value indicating whether the project should also be an 'RStudio Project'. Defaults to `FALSE`. If `TRUE`, then a 'projectname.Rproj' with usable defaults is added to the ProjectTemplate directory.

Details

If the target directory does not exist, it is created. Otherwise, it can only contain files and directories allowed by the merge strategy.

Value

No value is returned; this function is called for its side effects.

See Also

load.project, get.project, cache.project, show.project

Examples

```
library('ProjectTemplate')

## Not run: create.project('MyProject')
```

---

create.template *Create a new template*

Description

This function writes a skeleton directory structure for creating your own custom templates.

Usage

```
create.template(target, source = "minimal")
```

Arguments

target	Name of the new template. It is created under the directory specified by options('ProjectTemplate.ter or, when missing, in the current directory.
source	Name of an existing template to copy, defaults to the built in 'minimal' template.

---

get.project *Show information about the current project.*

Description

This function will return all of the information that ProjectTemplate has about the current project. This information is gathered when load.project is called. At present, ProjectTemplate keeps a record of the project's configuration settings, all packages that were loaded automatically and all of the data sets that were loaded automatically. The information about autoloaded data sets is used by the cache.project function.

Usage

```
get.project()
```

Details

In previous releases this information has been available through the global variable project.info. Using this variable is now deprecated and will result in a warning.

Value

A named list.

See Also

create.project, load.project, cache.project, show.project

Examples

```
library('ProjectTemplate')

## Not run: load.project()

get.project()
## End(Not run)
```

---

list.data *Listing the data for the current project*

Description

This function produces a data.frame of all data files in the project, with meta data on if and how the file will be loaded by load.project.

Usage

```
list.data(...)
```

Arguments

...	Named arguments to override configuration from config/global.dcf and lib/global.R.

Details

The returned data.frame contains the following variables, with one observation per file in data/:

filename	Character variable containing the filename relative to data/ directory.
varname	Character variable containing the name of the variable into which the file will be imported. *
is_ignored	Logical variable that indicates whether the file. is ignored through the data_ignore option in the configurat
is_directory	Logical variable that indicates whether the file is a directory.
is_cached	Logical variable that indicates whether the file is already available in the cache/ directory.
cached_only	Logical variable that indicates whether the variable is only available in the cache/ directory. This occurs wh
reader	Character variable containing the name of the reader function that will be used to load the data. Contains a c

* Note that some readers return more than one variable, usually with the listed variable name as prefix. This is true for for example the xls.reader and xlsx.reader.

Value

A data.frame listing the available data, with relevant meta data

See Also

load.project, show.project, project.config

Examples

```
library('ProjectTemplate')

## Not run: list.data()
```

---

load.project *Automatically load data and packages for a project.*

Description

This function automatically load all of the data and packages used by the project from which it is called. The behavior can be controlled by adjusting the project.config configuration.

Usage

```
load.project(...)
```

Arguments

...	Named arguments to override configuration from config/global.dcf and lib/global.R.

Details

... can take an argument override.config or a single named list for backward compatibility. This cannot be mixed with the new style override. When a named argument override.config is present it takes precedence over the other options. If any of the provided arguments is unnamed an error is raised.

Value

No value is returned; this function is called for its side effects.

See Also

create.project, get.project, cache.project, show.project, project.config

Examples

```
library('ProjectTemplate')

## Not run: load.project()
```

migrate.project *Migrates a project from a previous version of ProjectTemplate*

Description

This function automatically performs all necessary steps to migrate an existing project so that it is compatible with this version of ProjectTemplate

Usage

```
migrate.project()
```

Value

No value is returned; this function is called for its side effects.

See Also

create.project

Examples

```
library('ProjectTemplate')

## Not run: migrate.project()
```

migrate.template *Migrate a template to a new version of ProjectTemplate*

Description

This function updates a skeleton project to the current version of ProjectTemplate.

Usage

```
migrate.template(template)
```

Arguments

template	Name of the template to upgrade.

project.config *ProjectTemplate Configuration file*

Description

Every ProjectTemplate project has a configuration file found at config/global.dcf that contains various options that can be tweaked to control runtime behavior. The valid options are shown below, and must be encoded using the DCF format.

Usage

```
project.config()
```

Details

Calling the project.config() function will display the current project configuration.

The options that can be configured in the config/global.dcf are shown below

data_loading	This can be set to TRUE or FALSE. If data_loading is on, the system will load data from bo
data_loading_header	This can be set to TRUE or FALSE. If data_loading_header is on, the system will load text d
data_ignore	A comma separated list of files to be ignored when importing from the data/ directory. Reg
cache_loading	This can be set to TRUE or FALSE. If cache_loading is on, the system will load data from th
recursive_loading	This can be set to TRUE or FALSE. If recursive_loading is on, the system will load data fro
munging	This can be set to TRUE or FALSE. If munging is on, the system will execute the files in the
logging	This can be set to TRUE or FALSE. If logging is on, a logger object using the log4r package
logging_level	The value of logging_level is passed to a logger object using the log4r package during loggi
load_libraries	This can be set to TRUE or FALSE. If load_libraries is on, the system will load all of the R p
libraries	This is a comma separated list of all the R packages that the user wants to automatically load
as_factors	This can be set to TRUE or FALSE. If as_factors is on, the system will convert every charac
tables_type	This is the format for default tables. Values can be 'tibble' (default), 'data_table', or 'data_fr
attach_internal_libraries	This can be set to TRUE or FALSE. If attach_internal_libraries is on, then every time a new
cache_loaded_data	This can be set to TRUE or FALSE. If cache_loaded_data is on, then data loaded from the d
sticky_variables	This is a comma separated list of any project-specific variables that should remain in the glo
underscore_variables	This can be set to TRUE to use underscores ('_') in variable names or FALSE to replace under
cache_file_format	The default file format for cached data is 'RData'. This can be set to 'qs' in order to benefit

If the config/globals.dcf is missing some items (for example because it was created under an old version of ProjectTemplate, then the following configuration is used for any missing items during load.project():

```
data_loading                TRUE
data_loading_header         TRUE
data_ignore
cache_loading               TRUE
recursive_loading           FALSE
munging                     TRUE
```

```
logging                   FALSE
logging_level             INFO
load_libraries            FALSE
libraries                 reshape2, plyr, tidyverse, stringr, lubridate
as_factors                FALSE
tables_type               tibble
attach_internal_libraries TRUE
cache_loaded_data         FALSE
sticky_variables          NONE
underscore_variables      FALSE
cache_file_format         RData
```

When a new project is created using `create.project()`, the following values are pre-populated:

```
version                   0.11.0
data_loading              TRUE
data_loading_header       TRUE
data_ignore
cache_loading             TRUE
recursive_loading         FALSE
munging                   TRUE
logging                   FALSE
logging_level             INFO
load_libraries            FALSE
libraries                 reshape2, plyr, tidyverse, stringr, lubridate
as_factors                FALSE
tables_type               tibble
attach_internal_libraries FALSE
cache_loaded_data         TRUE
sticky_variables          NONE
underscore_variables      TRUE
cache_file_format         RData
```

Value

The current project configuration is displayed.

See Also

`load.project`

reload.project *Reload or reset a project*

Description

This function will clear the global environment and reload a project. This is useful when you've updated your data sets or changed your preprocessing scripts. Any `sticky_variables` configuration parameter in `project.config` will remain both in memory and (if present) in the cache by default. If the `reset` parameter is `TRUE`, then all variables are cleared from both the global environment and the cache.

Usage

```
reload.project(..., reset = FALSE)
```

Arguments

`...`	Optional parameters passed to `load.project`
`reset`	A boolean value, which if set `TRUE` clears the cache and everything in the global environment, including any `sticky_variables`

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: load.project()

reload.project()
## End(Not run)
```

require.package *Require a package for use in the project*

Description

This functions will require the given package. If the package is not installed it will stop execution and print a message to the user instructing them which package to install and which function caused the error.

Usage

```
require.package(package.name, attach = TRUE)
```

Arguments

package.name	A character vector containing the package name. Must be a valid package name installed on the system.
attach	Should the package be attached to the search path (as with library) or not (as with loadNamespace)? Defaults to TRUE. (Internal code will use FALSE by default unless a compatibility switch is set, see below.)

Details

The function .require.package is called by internal code. It will attach the package to the search path (with a warning) only if the compatibility configuration attach_internal_libraries is set to TRUE. Normally, packages used for loading data are not needed on the search path, but not loading them might break existing code. In a forthcoming version this compatibility setting will be removed, and no packages will be attached to the search path by internal code.

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: require.package('PackageName')
```

---

run.project *Run all of the analyses in the* src *directory.*

Description

This function will run each of the analyses in the src directory in separate processes. At present, this is done serially, but future versions of this function will provide a means of running the analyses in parallel.

Usage

```
run.project()
```

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: run.project()
```

show.project *Show information about the current project.*

Description

This function will show the user all of the information that ProjectTemplate has about the current project. This information is gathered when `load.project` is called. At present, ProjectTemplate keeps a record of the project's configuration settings, all packages that were loaded automatically and all of the data sets that were loaded automatically. The information about autoloaded data sets is used by the `cache.project` function.

Usage

```
show.project()
```

Value

No value is returned; this function is called for its side effects.

See Also

`create.project`, `load.project`, `get.project`, `cache.project`

Examples

```
library('ProjectTemplate')

## Not run: load.project()

show.project()
## End(Not run)
```

stub.tests *Generate unit tests for your helper functions.*

Description

This function will parse all of the functions defined in files inside of the `lib` directory and will generate a trivial unit test for each function. The resulting tests are stored in the file `tests/autogenerated.R`. Every test is excepted to fail by default, so you should edit them before calling `test.project`.

Usage

```
stub.tests()
```

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: stub.tests()
```

---

test.project *Run all unit tests for this project.*

Description

This function will run all of the `testthat` style unit tests for the current project that are defined inside of the `tests` directory. The tests will be run in the order defined by the filenames for the tests: it is recommend that each test begin with a number specifying its position in the sequence.

Usage

```
test.project()
```

Value

No value is returned; this function is called for its side effects.

Examples

```
library('ProjectTemplate')

## Not run: load.project()

test.project()
## End(Not run)
```

---

translate.dcf *Read a DCF file into an R list.*

Description

This function will read a DCF file and translate the resulting data frame into a list. The DCF format is used throughout ProjectTemplate for configuration settings and ad hoc file format specifications.

Usage

```
translate.dcf(filename)
```

Arguments

filename	A character vector specifying the DCF file to be translated.

Details

The content of the DCF file are stored as character strings. If the content is placed between the back tick character , then the content is evaluated as R code and the result returned in a string

Value

Returns a list containing the entries from the DCF file.

Examples

```
library('ProjectTemplate')

## Not run: translate.dcf(file.path('config', 'global.dcf'))
```

Index