Package: onetime (via r-universe)

October 14, 2024

Type Package

Title Run Code Only Once

Version 0.2.0

Author David Hugh-Jones [aut, cre]

Maintainer David Hugh-Jones <davidhughjones@gmail.com>

Description Allows code to be run only once on a given computer, using lockfiles. Typical use cases include startup messages shown only when a package is loaded for the very first time.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE, roclets = c(``collate", ``rd", ``namespace", ``doctest::dt_roclet"))

RoxygenNote 7.2.3

Imports rappdirs, filelock

URL https://github.com/hughjonesd/onetime, https://hughjonesd.github.io/onetime/

BugReports https://github.com/hughjonesd/onetime/issues

Suggests callr, covr, devtools, doctest (>= 0.1.0), knitr, lifecycle, mockr, rlang, rmarkdown, testthat (>= 2.1.0), withr

VignetteBuilder knitr

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

RemoteUrl https://github.com/hughjonesd/onetime

RemoteRef CRAN-0.2.0

RemoteSha e36294db9b20ec964487b251fd76306c890d3891

Contents

`check_ok_to_store` *Check if the package has permission to store files on the user's computer*

Description

The onetime package works by storing lockfiles in `rappdirs::user_config_dir()`. It won't do so unless permission has been granted. Before using `onetime` functions, package authors should call `check_ok_to_store(ask = TRUE)` in an interactive session, in functions which are called directly from the command line.

Usage

```
check_ok_to_store(
  ask = FALSE,
  message = "The onetime package requests to store files in '%s'.",
  confirm_prompt = "Is this OK? [Yn] ",
  confirm_answers = c("Y", "y", "Yes", "yes", "YES"),
  default_answer = "Y"
)
```

Arguments

`ask`	`TRUE` to ask the user for permission.
`message`	Message to display to the user.
`confirm_prompt`	Character string. Question to prompt the user to hide the message in future.
`confirm_answers`	Character vector. Answers which will cause the message to be hidden in future.
`default_answer`	Character string. Default answer if user simply presses return.

Details

If your package is not used interactively, a workaround is to call `set_ok_to_store()`. This grants permission and prints an informative message. Package owners should *only* call this if they cannot ask explicitly.

`onetime_message_confirm()` is an exception: by default it doesn't require global permission to store files, since the user accepting "Don't show this again" is considered sufficient.

`ask = TRUE` asks the user, if he or she has not already given permission, and if the session is `interactive()`.

Remaining parameters are passed to `onetime_message_confirm()` in this case, and ignored otherwise. A `"%s"` in `message` will be replaced by the onetime storage directory.

Value

`TRUE` if:

- We already have permission;
- `ask` is `TRUE`, we are in an interactive session and the user gives us permission;
- `options("onetime.dir")` is set to a non-`NULL` value.

Otherwise `FALSE`.

Examples

```
check_ok_to_store()
```

`onetime` *Run code only once*

Description

Onetime allows package authors to run code only once (ever) for a given user. It does so by writing a file, typically to a folder in the user's configuration directory as given by `rappdirs::user_config_dir()`. The user can set an alternative filepath using `options("onetime.dir")`.

Details

Core functions include:

- `onetime_do()` runs arbitrary code only once.
- `onetime_warning()` and friends print a warning or message only once.
- `onetime_message_confirm()` prints a message and asks "Show this message again?"
- `onetime_rlang_warn()` and `onetime_rlang_inform()` print messages using functions from the rlang package.
- `onetime_only()` returns a function that runs only once.

- check_ok_to_store() and set_ok_to_store() check for or grant permission to store lockfiles on the user's computer. It is package authors' responsibility to check for permission to store lockfiles. This may have been done already by another package if onetime was already installed. You can ask permission interactively on the command line by calling check_ok_to_store() with ask = TRUE.

For more information, see vignette("onetime").

Example:

```
library(onetime)

ids  <- paste0("onetime-readme-", 1:3)

for (i in 1:5) {
  onetime_do(cat("This command will only be run once.\n"), id = ids[1])
  onetime_warning("This warning will only be shown once.", id = ids[2])
  onetime_message("This message will only be shown once.", id = ids[3])
}

## This command will only be run once.

## Warning: This warning will only be shown once.

## This message will only be shown once.
```

Author(s)

Maintainer: David Hugh-Jones <davidhughjones@gmail.com>

See Also

Useful links:

- https://github.com/hughjonesd/onetime
- https://hughjonesd.github.io/onetime/
- Report bugs at https://github.com/hughjonesd/onetime/issues

---

onetime-rlang *Print a warning or message only once using* rlang *functions*

Description

If you use these you will need to add "rlang" to your package dependencies.

Usage

```
onetime_rlang_warn(
  ...,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL,
  without_permission = "warn"
)

onetime_rlang_inform(
  ...,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL,
  without_permission = "warn"
)
```

Arguments

`...`	Passed to `rlang::warn()` or `rlang::inform()`.
`id`	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting `id` is deprecated.
`path`	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath `rappdirs::user_config_dir()`. Normally you should leave this as the default.
`expiry`	`difftime()` or e.g. `lubridate::duration()` object. After this length of time, code will be run again.
`without_permission`	Character string. What to do if the user hasn't given permission to store files? `"warn"` runs the action with an extra warning; `"run"` runs the action with no warning; `"pass"` does nothing and returns the default; `"stop"` throws an error; `"ask"` asks for permission using `check_ok_to_store()`, and returns the default if it is not granted.

Value

Invisibly: `TRUE` if the message/warning was shown, `FALSE` otherwise.

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1)

for (n in 1:3) {
  onetime_rlang_warn(c("rlang-style warning", i = "Extra info"), id = id)
}

onetime_reset(id = id)
options(oo)
```

onetime_been_done *Check if a onetime call has already been made*

Description

Check if a onetime call has already been made

Usage

```
onetime_been_done(
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL
)
```

Arguments

id	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting id is deprecated.
path	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath rappdirs::user_config_dir(). Normally you should leave this as the default.
expiry	difftime() or e.g. lubridate::duration() object. After this length of time, code will be run again.

Value

TRUE if the call has been recorded (within the expiry time, if given).

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1)
onetime_been_done(id = id)
onetime_message("Creating an ID",  id = id)
onetime_been_done(id = id)

onetime_reset(id = id)
options(oo)
```

onetime_dir *Return a path to a directory beneath the onetime base directory*

Description

By default lockfiles are stored beneath the onetime base directory, in a directory named after the calling package. You can use a different subdirectory by setting `path = onetime_dir("dirname")` in calls to onetime functions.

Usage

```
onetime_dir(dir)
```

Arguments

`dir`	String. Name of a single directory.

Details

`onetime_dir()` does not autocreate the directory (but it will get created during the call to `onetime_do()`).

Value

The path.

Examples

```
onetime_dir("my-folder")
oo <- options(onetime.dir = tempdir(check = TRUE))
onetime_dir("my-folder")
options(oo)
```

onetime_do *Run code only once*

Description

When first called, `onetime_do()` evaluates an expression. It then creates a lockfile recording a unique ID which will prevent the expression being run on subsequent calls.

Usage

```
onetime_do(
  expr,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL,
  default = NULL,
  without_permission = c("warn", "run", "stop", "pass", "ask")
)
```

Arguments

`expr`	The code to evaluate. An R statement or `expression()` object.
`id`	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting `id` is deprecated.
`path`	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath `rappdirs::user_config_dir()`. Normally you should leave this as the default.
`expiry`	`difftime()` or e.g. `lubridate::duration()` object. After this length of time, code will be run again.
`default`	Value to return if `expr` was not executed.
`without_permission`	Character string. What to do if the user hasn't given permission to store files? `"warn"` runs the action with an extra warning; `"run"` runs the action with no warning; `"pass"` does nothing and returns the default; `"stop"` throws an error; `"ask"` asks for permission using `check_ok_to_store()`, and returns the default if it is not granted.

Details

`onetime_do()` is the engine used by other onetime functions.

Calls are identified by `id`. If you use the same value of `id` across different calls to onetime functions, only the first call will get made.

The default `path`, where lockfiles are stored, is in a per-package directory beneath `rappdirs::user_config_dir()`. To use a different subdirectory within the onetime base directory, set `path = onetime_dir("dirname")`.

End users can also set `options(onetime.dir)` to change the base directory. Package authors should only set this option locally within package functions, if at all.

If the call gives an error, the lockfile is still written.

`expiry` is backward-looking. That is, `expiry` is used at check time to see if the lockfile was written after `Sys.time() - expiry`. It is not used when the lockfile is created. So, you should set `expiry` to the same value whenever you call `onetime_do()`. See the example.

Value

`onetime_do()` invisibly returns the value of `expr`, or `default` if `expr` was not run because it had been run already.

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1L)

for (n in 1:3) {
  onetime_do(print("printed once"), id = id)
}

# expiry is "backward-looking":
id2 <- sample(10000L, 1L)
expiry <- as.difftime(1, units = "secs")
onetime_do(print("Expires quickly, right?"), id = id2, expiry = expiry)
Sys.sleep(2)
onetime_do(print("This won't be shown..."), id = id2)
onetime_do(print("... but this will"), id = id2, expiry = expiry)


onetime_reset(id = id)
onetime_reset(id = id2)
options(oo)
```

`onetime_mark_as_done` *Mark an action as done*

Description

This manually marks an action as done.

Usage

```
onetime_mark_as_done(
  id = deprecate_calling_package(),
  path = default_lockfile_dir()
)
```

Arguments

`id`	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting `id` is deprecated.
`path`	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath `rappdirs::user_config_dir()`. Normally you should leave this as the default.

Details

Note that no `expiry` parameter is available, because `expiry` is backward-looking. See `onetime_do()` for more information.

Marking an action done requires permission to store files on the user's computer, just like other onetime actions.

Value

Invisible `TRUE` if the action represented by `id` had not been done before, and has now been explicitly marked as done. Invisible `FALSE` if it was already marked as done (and still is).

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1)
onetime_mark_as_done(id = id)
onetime_message("Won't be shown", id = id)

onetime_reset(id = id)
options(oo)
```

`onetime_message_confirm` *Print a message, and ask for confirmation to hide it in future*

Description

This uses `readline()` to ask the user if the message should be shown again in future.

Usage

```
onetime_message_confirm(
  ...,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL,
  confirm_prompt = "Show this message again? [yN] ",
  confirm_answers = c("N", "n", "No", "no"),
  default_answer = "N",
  require_permission = FALSE,
  without_permission = "warn",
  noninteractive = paste0("To hide this message in future, run:\n",
    "  onetime::onetime_mark_as_done(id = \"", id, "\")"),
  message = .Deprecated()
)
```

Arguments

`...`	Passed to `message()`.
`id`	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting `id` is deprecated.
`path`	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath `rappdirs::user_config_dir()`. Normally you should leave this as the default.

`expiry` `difftime()` or e.g. `lubridate::duration()` object. After this length of time, code will be run again.

`confirm_prompt` Character string. Question to prompt the user to hide the message in future.

`confirm_answers` Character vector. Answers which will cause the message to be hidden in future.

`default_answer` Character string. Default answer if user simply presses return.

`require_permission` Logical. Ask permission to store files on the user's computer, if this hasn't been granted? Setting this to `FALSE` overrides `without_permission`.

`without_permission` Character string. What to do if the user hasn't given permission to store files? `"warn"` runs the action with an extra warning; `"run"` runs the action with no warning; `"pass"` does nothing and returns the default; `"stop"` throws an error; `"ask"` asks for permission using `check_ok_to_store()`, and returns the default if it is not granted.

`noninteractive` String. Additional message to send in non-interactive sessions. Set to `NULL` to do nothing in non-interactive sessions. The default tells the user how to manually mark the message as done.

`message` Deprecated. Use unnamed arguments `...` instead.

Details

By default, the message will be hidden if the user answers "n", "No", or "N", or just presses return to the prompt question.

Unlike other `onetime` functions, `onetime_message_confirm()` doesn't by default require permission to store files on the user's computer. The assumption is that saying "Don't show this message again" counts as granting permission (just for this one message). You can ask for broader permission by setting `require_permission = TRUE` and `without_permission = "ask"`.

Value

- `NULL` if the message was not shown (shown already or non-interactive session and `noninteractive` was `NULL`).
- `TRUE` if the user confirmed, i.e. chose to hide the message.
- `FALSE` if the message was shown but the user did not confirm (did not choose to hide the message, or non-interactive session and `noninteractive` was not `NULL`).

Results are returned invisibly.

Note that by default, `TRUE` is returned when the user answers "no" to "Show this message again?" and `FALSE` is returned when the user answers "yes".

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1L)
onetime_message_confirm("A message to show one or more times", id = id)
```

```
onetime_reset(id = id)
options(oo)
```

`onetime_only` *Wrap a function to be called only once*

Description

This takes a function and returns the same function wrapped by `onetime_do()`. Use it for code which should run only once, but which may be called from multiple locations. This frees you from having to use the same `id` multiple times.

Usage

```
onetime_only(
  .f,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  default = NULL,
  without_permission = "warn"
)
```

Arguments

`.f`	A function
`id`	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting `id` is deprecated.
`path`	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath `rappdirs::user_config_dir()`. Normally you should leave this as the default.
`default`	Value to return from `.f` if function was not executed.
`without_permission`	Character string. What to do if the user hasn't given permission to store files? `"warn"` runs the action with an extra warning; `"run"` runs the action with no warning; `"pass"` does nothing and returns the default; `"stop"` throws an error; `"ask"` asks for permission using `check_ok_to_store()`, and returns the default if it is not granted.

Value

A wrapped function. The function itself returns the result of `.f`, or `default` if the inner function was not called.

See Also

`onetime_do()`

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1)

sample_once <- onetime_only(sample, id = id)
sample_once(1:10)
sample_once(1:10)

onetime_reset(id)
options(oo)
```

`onetime_reset` *Reset a onetime call by ID*

Description

Reset a onetime call by ID

Usage

```
onetime_reset(id = deprecate_calling_package(), path = default_lockfile_dir())
```

Arguments

`id`	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting `id` is deprecated.
`path`	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath `rappdirs::user_config_dir()`. Normally you should leave this as the default.

Value

The result of `file.remove()`, invisibly.

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1)
onetime_message("will be shown",  id = id)
onetime_message("won't be shown", id = id)
onetime_reset(id = id)
onetime_message("will be shown",  id = id)

onetime_reset(id = id)
options(oo)
```

onetime_warning *Print a warning or message only once*

Description

These functions use `onetime_do()` to print a warning or message just once.

Usage

```
onetime_warning(
  ...,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL,
  without_permission = "warn"
)

onetime_message(
  ...,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL,
  without_permission = "warn"
)

onetime_startup_message(
  ...,
  id = deprecate_calling_package(),
  path = default_lockfile_dir(),
  expiry = NULL,
  without_permission = "warn"
)
```

Arguments

`...`	Passed to `warning()`, `message()` or `packageStartupMessage()`.
`id`	Unique ID string. If this is unset, the name of the calling package will be used. Since onetime 0.2.0, not setting `id` is deprecated.
`path`	Directory to store lockfiles. The default uses a unique directory corresponding to the calling package, beneath `rappdirs::user_config_dir()`. Normally you should leave this as the default.
`expiry`	`difftime()` or e.g. `lubridate::duration()` object. After this length of time, code will be run again.
`without_permission`	Character string. What to do if the user hasn't given permission to store files? `"warn"` runs the action with an extra warning; `"run"` runs the action with no

warning; "pass" does nothing and returns the default; "stop" throws an error; "ask" asks for permission using check_ok_to_store(), and returns the default if it is not granted.

Value

Invisible TRUE if the message/warning was shown, invisible FALSE otherwise.

See Also

onetime_do()

Examples

```
oo <- options(onetime.dir = tempdir(check = TRUE))
id <- sample(10000L, 1)

for (n in 1:3) {
  onetime_warning("will be shown once", id = id)
}

onetime_reset(id = id)
options(oo)
```

---

set_ok_to_store *Grant or revoke permission to store lockfiles on the user's computer*

Description

End users may use this from the command line. Package authors should *only* call it if they cannot ask for permission interactively using check_ok_to_store(ask = TRUE).

Usage

```
set_ok_to_store(ok = TRUE)
```

Arguments

ok	TRUE to grant permission to store lockfiles, FALSE to revoke it and unset options("onetime.dir").

Value

Invisible NULL.

Examples

```
## Not run:
set_ok_to_store()

## End(Not run)
```

Index