Package 'ids'

November 4, 2016

Title Generate Random Identifiers

Version 1.0.0

Description Generate random or human readable and pronounceable identifiers.

License MIT + file LICENSE

URL https://github.com/richfitz/ids

BugReports https://github.com/richfitz/ids/issues

Imports openssl, uuid

Suggests knitr, rcorpora, rmarkdown, testthat

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Rich FitzJohn [aut, cre]

Maintainer Rich FitzJohn <rich.fitzjohn@gmail.com>

Repository CRAN

Date/Publication 2016-11-04 23:51:27

R topics documented:

adjective_animal 2
ids 3
random_id 4
sentence 5
uuid 6

Index **7**

`adjective_animal` *Ids based on a number of adjectives and an animal*

Description

Ids based on a number of adjectives and an animal

Usage

```
adjective_animal(n = 1, n_adjectives = 1, style = "snake",
  max_len = Inf)
```

Arguments

`n`	number of ids to return. If `NULL`, it instead returns the generating function
`n_adjectives`	Number of adjectives to prefix the anmial with
`style`	Style to join words with. Can be one of "Pascal", "camel", "snake", "kebab", "dot", "title", "sentence", "lower", "upper", and "constant".
`max_len`	The maximum length of a word part to include (this may be useful because some of the names are rather long. This stops you generating a `hexakosioihexekontahexaphobic_queenalex` A vector of length 2 can be passed in here in which case the first element will apply to the adjectives (all of them) and the second element will apply to the animals.

Details

The list of adjectives and animals comes from https://github.com/a-type/adjective-adjective-animal, and in turn from gfycat.com

Author(s)

Rich FitzJohn

Examples

```
# Generate a random identifier:
adjective_animal()

# Generate a bunch all at once:
adjective_animal(5)

# Control the style of punctuation with the style argument:
adjective_animal(style = "lower")
adjective_animal(style = "CONSTANT")
adjective_animal(style = "camel")
adjective_animal(style = "kebab")

# Control the number of adjectives used
```

```
adjective_animal(n_adjectives = 3)

# This can get out of hand quickly though:
adjective_animal(n_adjectives = 7)

# Limit the length of adjectives and animals used:
adjective_animal(10, max_len = 6)

# The lengths can be controlled for adjectives and animals
# separately, with Inf meaning no limit:
adjective_animal(10, max_len = c(6, Inf), n_adjectives = 2)

# Pass n = NULL to bind arguments to a function
id <- adjective_animal(NULL, n_adjectives = 2, style = "dot", max_len = 6)
id()
id(10)
```

`ids` *Generic id generating function*

Description

Generic id generating function

Usage

```
ids(n, ..., vals = list(...), style = "snake")
```

Arguments

`n`	number of ids to return. If `NULL`, it instead returns the generating function
`...`	A number of character vectors
`vals`	A list of character vectors, *instead* of `...`
`style`	Style to join words with. Can be one of "Pascal", "camel", "snake", "kebab", "dot", "title", "sentence", "lower", "upper", and "constant".

Value

Either a character vector of length `n`, or a function of one argument if `n` is `NULL`

Author(s)

Rich FitzJohn

Examples

```
# For an example, please see the vignette
```

random_id *Cryptographically generated random identifiers*

Description

Random identifiers. By default this uses the openssl package to produce a random set of bytes, and expresses that as a hex character string. This does not affect R's random number stream.

Usage

```
random_id(n = 1, bytes = 16, use_openssl = TRUE)
```

Arguments

n	number of ids to return. If NULL, it instead returns the generating function
bytes	The number of bytes to include for each identifier. The length of the returned identifiers will be twice this long with each pair of characters representing a single byte.
use_openssl	A logical, indicating if we should use the openssl for generating the random identifiers. The openssl random bytestream is not affected by the state of the R random number generator (e.g., via set.seed) so may not be suitable for use where reproducibility is important. The speed should be very similar for both approaches.

Author(s)

Rich FitzJohn

Examples

```
# Generate a random id:
random_id()

# Generate 10 of them!
random_id(10)

# Different length ids
random_id(bytes = 8)
# (note that the number of characters is twice the number of bytes)

# The ids are not affected by R's RNG state:
set.seed(1)
(id1 <- random_id())
set.seed(1)
(id2 <- random_id())
# The generated identifiers are different, despite the seed being the same:
id1 == id2
```

```
# If you need these identifiers to be reproducible, pass use_openssl = FALSE
set.seed(1)
(id1 <- random_id(use_openssl = FALSE))
set.seed(1)
(id2 <- random_id(use_openssl = FALSE))
# This time they are the same:
id1 == id2

# Pass \code{n = NULL} to generate a function that binds your arguments:
id8 <- random_id(NULL, bytes = 8)
id8(10)
```

`sentence` *Sentence style identifiers*

Description

Create a sentence style identifier. This uses the approach described by Asana on their blog https://blog.asana.com/2011/09/6-sad-squid-snuggle-softly/. This approach encodes 32 bits of information (so 2^32 ~= 4 billion possibilities) and in theory can be remapped to an integer if you really wanted to.

Usage

```
sentence(n = 1, style = "snake", past = FALSE)
```

Arguments

`n`	number of ids to return. If `NULL`, it instead returns the generating function
`style`	Style to join words with. Can be one of "Pascal", "camel", "snake", "kebab", "dot", "title", "sentence", "lower", "upper", and "constant".
`past`	Use the past tense for verbs (e.g., slurped or jogged rather than slurping or jogging)

Author(s)

Rich FitzJohn

Examples

```
# Generate an identifier
sentence()

# Generate a bunch
sentence(10)

# As with adjective_animal, use "style" to control punctuation
sentence(style = "Camel")
sentence(style = "dot")
```

```
sentence(style = "Title")

# Change the tense of the verb:
set.seed(1)
sentence()
set.seed(1)
sentence(past = TRUE)

# Pass n = NULL to bind arguments to a function
id <- sentence(NULL, past = TRUE, style = "dot")
id()
id(10)
```

---

uuid *Generate UUIDs*

Description

Generate UUIDs using the uuid package. This is simply a thin wrapper around `uuid::UUIDgenerate` that matches the interface in the rest of the ids package.

Usage

```
uuid(n = 1, drop_hyphens = FALSE, use_time = NA)
```

Arguments

`n`	number of ids to return. If `NULL`, it instead returns the generating function
`drop_hyphens`	Drop the hyphens from the UUID?
`use_time`	Passed through to `UUIDgenerate` as `use.time`.

Author(s)

Rich FitzJohn

Examples

```
# Generate one id
uuid()

# Or a bunch
uuid(10)

# More in the style of random_id()
uuid(drop_hyphens = TRUE)
```

Index

adjective_animal, 2

ids, 3

random_id, 4

sentence, 5
set.seed, *4*

uuid, 6