Package 'emayili'

October 8, 2021

Type Package

Title Send Email Messages

Version 0.6.1

Description A light, simple tool for sending emails with minimal dependencies.

URL https://datawookie.github.io/emayili/

BugReports https://github.com/datawookie/emayili/issues

License GPL-3

Language en-GB

Imports base64enc, commonmark, curl (>= 4.0), digest, dplyr, glue, htmltools, httr, logger, magrittr, mime, purrr, rmarkdown, stringr, tidyr, urltools, vctrs, xfun, xml2

Suggests here, memoise, testthat (>= 2.1.0), roxygen2, showtext

SystemRequirements The function render() requires Pandoc (http://pandoc.org).

Encoding UTF-8

RoxygenNote 7.1.1

KeepSource true

NeedsCompilation no

Author Andrew B. Collier [aut, cre, cph],
Matt Dennis [ctb],
Antoine Bichat [ctb] (<https://orcid.org/0000-0001-6599-7081>),
Daniel Fahey [ctb],
Johann R. Kleinbub [ctb],
Panagiotis Moulos [ctb]

Maintainer Andrew B. Collier <andrew@fathomdata.dev>

Repository CRAN

Date/Publication 2021-10-08 08:30:08 UTC

R topics documented:

`address` *Email Address*

Description

Create an `address` object which represents an email address.

Usage

```
address(email = NA, display = NA, local = NA, domain = NA, normalise = TRUE)
```

Arguments

email	Email address.
display	Display name.
local	Local part of email address.
domain	Domain part of email address.
normalise	Whether to try to normalise address to RFC-5321 requirements.

Details

Implemented as an S3 vector class.

Value

An address object, representing an email address.

Examples

```
address("gerry@gmail.com")
address("gerry@gmail.com", "Gerald")
address(
  c("gerry@gmail.com", "alice@yahoo.com", "jim@aol.com"),
  c("Gerald", "Alice", NA)
)
```

---

addresses *Add address fields to message*

Description

Add address fields to message

Usage

```
to(msg, ..., append = TRUE)

cc(msg, ..., append = TRUE)

bcc(msg, ..., append = TRUE)

from(msg, addr = NULL)

reply(msg, addr = NULL)

sender(msg, addr = NULL)
```

Arguments

msg	A message object.
...	Addresses.
append	Whether to append or replace addresses.
addr	Single address.

Value

A message object.

Examples

```
# Populating the To field.
msg <- envelope()
msg %>% to("bob@gmail.com, alice@yahoo.com")
msg %>% to("bob@gmail.com", "alice@yahoo.com")
msg %>% to(c("bob@gmail.com", "alice@yahoo.com"))

# Populating the Cc field.
msg <- envelope()
msg %>% cc("bob@gmail.com, alice@yahoo.com")
msg %>% cc("bob@gmail.com", "alice@yahoo.com")
msg %>% cc(c("bob@gmail.com", "alice@yahoo.com"))

# Populating the Bcc field.
msg <- envelope()
msg %>% bcc("bob@gmail.com, alice@yahoo.com")
msg %>% bcc("bob@gmail.com", "alice@yahoo.com")
msg %>% bcc(c("bob@gmail.com", "alice@yahoo.com"))

msg <- envelope()

# Populating the From field.
msg %>% from("craig@gmail.com")

# Populating the Reply-To field.
msg <- envelope()
msg %>% reply("gerry@gmail.com")

# Populating the Sender field.
msg <- envelope()
msg %>% sender("on_behalf_of@gmail.com")
```

---

append.envelope *Append children to message*

Description

Append children to message

Usage

```
## S3 method for class 'envelope'
append(x, child)
```

Arguments

x	Message object
child	A child to be appended

---

as.address *Create an address object*

Description

Create an address object

Usage

```
as.address(addr)
```

Arguments

addr	An email address.

Value

An address object.

Examples

```
as.address("gerry@gmail.com")
as.address("Gerald <gerry@gmail.com>")
as.address(c("Gerald <gerry@gmail.com>", "alice@yahoo.com", "jim@aol.com"))
as.address("Gerald <gerry@gmail.com>, alice@yahoo.com, jim@aol.com")
as.address(c("Gerald <gerry@gmail.com>", "alice@yahoo.com, jim@aol.com"))
```

as.character.envelope *Create formatted message.*

Description

Accepts a message object and formats it as a MIME document.

Usage

```
## S3 method for class 'envelope'
as.character(x, ..., details = TRUE)
```

Arguments

x	A message object.
...	Further arguments passed to or from other methods.
details	Whether or not to display full message content.

Value

A formatted message object.

as.character.header *Create formatted header.*

Description

Accepts a header object and formats it as a header field.

Usage

```
## S3 method for class 'header'
as.character(x, ...)
```

Arguments

x	A header object.
...	Further arguments passed to or from other methods.

Value

A formatted header field.

as.character.MIME *Convert MIME object to character vector*

Description

Convert MIME object to character vector

Usage

```
## S3 method for class 'MIME'
as.character(x, ...)
```

Arguments

x	MIME object
...	Further arguments passed to or from other methods.

as.character.vctrs_address *Convert address object to character*

Description

Convert address object to character

Usage

```
## S3 method for class 'vctrs_address'
as.character(x, ...)
```

Arguments

x	A vector of `address` objects.
...	Further arguments passed to or from other methods.

Value

A character vector.

attachment *Add attachments to a message object*

Description

Add attachments to a message object

Usage

```
attachment(msg, path, name = NA, type = NA, cid = NA)
```

Arguments

msg	A message object.
path	Path to file.
name	Name to be used for attachment (defaults to base name of path).
type	MIME type or *NA*, which will result in a guess based on file extension.
cid	Content-ID or NA.

Value

A message object.

Examples

```
library(magrittr)

path_mtcars  <- tempfile(fileext = ".csv")
path_scatter <- tempfile(fileext = ".png")
path_cats    <- system.file("cats.jpg", package = "emayili")

write.csv(mtcars, path_mtcars)

png(path_scatter)
plot(1:10)
dev.off()

msg <- envelope() %>%
  attachment(path_mtcars) %>%
  # This attachment will have file name "cats.jpg".
  attachment(path_cats, name = "cats.jpg", type = "image/jpeg") %>%
  attachment(path_scatter, cid = "scatter")

file.remove(path_scatter, path_mtcars)
```

compare *Compare vectors*

Description

Returns TRUE wherever elements are the same (including NA), and FALSE everywhere else.

Usage

```
compare(lhs, rhs)
```

Arguments

lhs	LHS of operation.
rhs	RHS of operation.

Value

A Boolean value.

compliant *Tests whether an email address is syntactically correct*

Description

Checks whether an email address conforms to the syntax rules.

Usage

```
compliant(addr, error = FALSE)
```

Arguments

addr	An email address.
error	Whether to create an error if not compliant.

Details

An email address may take either of the following forms:

- local@domain or
- Display Name <local@domain>.

Value

A Boolean.

Examples

```
compliant("alice@example.com")
compliant("alice?example.com")
```

---

display *Extract display name*

Description

Extracts the display name from an email address.

Usage

```
display(addr)
```

Arguments

addr	An address object.

Value

The display name or NA.

Examples

```
gerry <- as.address("Gerald <gerry@gmail.com>")
display(gerry)
```

---

domain *Extract domain of email address*

Description

Extract domain of email address

Usage

```
domain(addr)
```

Arguments

addr	An address object.

Value

A character vector.

Examples

```
domain("alice@example.com")
```

`envelope` *Create a message.*

Description

Create a message.

Usage

```
envelope(
  to = NULL,
  from = NULL,
  cc = NULL,
  bcc = NULL,
  reply = NULL,
  subject = NULL,
  importance = NULL,
  priority = NULL,
  text = NULL,
  html = NULL
)
```

Arguments

`to`	See `to()`.
`from`	See `from()`.
`cc`	See `cc()`.
`bcc`	See `bcc()`.
`reply`	See `reply()`.
`subject`	See `subject()`.
`importance`	See `importance()`.
`priority`	See `priority()`.
`text`	See `text()`.
`html`	See `html()`.

Value

A message object.

See Also

`subject()`, `from()`, `to()`, `cc()`, `bcc()` and `reply()`.

Examples

```
# Create an (empty) message object.
#
msg <- envelope()

# Create a complete message object, specifying all available fields.
#
envelope(
  to = "bob@gmail.com",
  from = "craig@gmail.com",
  cc = "alex@gmail.com",
  bcc = "shannon@gmail.com",
  reply = "craig@yahoo.com",
  importance = "high",
  priority = "urgent",
  subject = "Hiya!",
  text = "Hi Bob, how are you?"
)
```

---

format.vctrs_address *Encode email addresses in a common format*

Description

Encode email addresses in a common format

Usage

```
## S3 method for class 'vctrs_address'
format(x, ...)
```

Arguments

x	A vector of address objects.
...	Further arguments passed to or from other methods.

Value

A character vector.

`html` *Add an HTML body to a message object.*

Description

Add an HTML body to a message object.

Usage

```
html(
  msg,
  content,
  disposition = "inline",
  charset = "utf-8",
  encoding = "quoted-printable",
  css_files = c(),
  interpolate = TRUE,
  .open = "{{",
  .close = "}}",
  .envir = NULL
)
```

Arguments

`msg`	A message object.
`content`	A string of message content.
`disposition`	How content is presented (Content-Disposition).
`charset`	How content is encoded.
`encoding`	How content is transformed to ASCII (Content-Transfer-Encoding).
`css_files`	Extra CSS files.
`interpolate`	Whether or not to interpolate into input using glue.
`.open`	The opening delimiter.
`.close`	The closing delimiter.
`.envir`	Environment used for `glue` interpolation. Defaults to `parent.frame()`.

Value

A message object.

See Also

`text`

Examples

```
library(magrittr)

# Inline HTML message.
envelope() %>% html("<b>Hello!</b>")

# Read HTML message from a file.
htmlfile <- tempfile(fileext = ".html")
cat("<p>Hello!</p>\n", file = htmlfile)
envelope() %>% html(htmlfile)
```

---

local *Extract local part of email address*

Description

Extract local part of email address

Usage

```
local(addr)
```

Arguments

addr	An address object.

Value

A character vector.

Examples

```
local("alice@example.com")
```

---

new_address *Helper function for creating address objects*

Description

Helper function for creating address objects

Usage

```
new_address(
  email = character(),
  display = character(),
  local = character(),
  domain = character(),
  normalise = TRUE
)
```

Arguments

email	Email address.
display	Display name.
local	Local part of email address.
domain	Domain part of email address.
normalise	Whether to try to normalise address to RFC-5321 requirements.

Value

An address object, representing an email address.

---

parties *Extract sender and recipient(s)*

Description

Extract sender and recipient(s)

Usage

```
parties(msg)
```

Arguments

msg	A message object.

Value

A tibble.

Examples

```
email <- envelope() %>%
  from("Gerald <gerald@gmail.com>") %>%
  to(c("bob@gmail.com", "alice@yahoo.com")) %>%
  cc("Craig     < craig@gmail.com>") %>%
  bcc("  Erin   <erin@yahoo.co.uk    >")

parties(email)
```

---

precedence *Add fields for message importance and priority*

Description

A hint to influence transmission speed and delivery.

A hint to the message recipient about how important the message is.

Usage

```
priority(msg, priority = NULL)

importance(msg, importance = NULL)
```

Arguments

msg	A message object.
priority	Priority level. One of "non-urgent", "normal", or "urgent".
importance	Importance level. One of "low", "normal", or "high".

Details

Does not influence transmission speed or delivery.

Value

A message object.

Examples

```
# How rapidly does the message need to be delivered?
#
envelope() %>%
  subject("Deliver this immediately!") %>%
  priority("urgent")

envelope(priority = "non-urgent") %>%
  subject("No rush with this.")
```

```
# How much attention should be paid by recipient?
#
envelope() %>%
  subject("Read this immediately!") %>%
  importance("high")

envelope(importance = "low") %>%
  subject("Not important at all. Just delete.")
```

`print.envelope` *Print a message object*

Description

The message body will be printed if `details` is `TRUE` or if the `envelope_details` option is `TRUE`.

Usage

```
## S3 method for class 'envelope'
print(x, details = NA, ...)
```

Arguments

`x`	A message object.
`details`	Whether or not to display full message content.
`...`	Further arguments passed to or from other methods.

Examples

```
msg <- envelope() %>% text("Hello, World!")

print(msg)
print(msg, details = TRUE)

options(envelope_details = TRUE)
print(msg)
```

`print.vctrs_address` *Print an address object*

Description

Print an address object

Usage

```
## S3 method for class 'vctrs_address'
print(x, ...)
```

Arguments

x	An address object.
...	Further arguments passed to or from other methods.

Examples

```
gerry <- as.address("gerry@gmail.com")
print(gerry)
```

---

qp_decode *Decode a quoted-printable string.*

Description

Decode a quoted-printable string.

Usage

```
qp_decode(x)
```

Arguments

x	A vector of strings.

Value

A vector of decoded strings.

---

qp_encode *Encode a string to quoted-printable.*

Description

Encode a string to quoted-printable.

Usage

```
qp_encode(x)
```

Arguments

x	A vector of strings.

Value

A vector of encoded strings.

---

raw *Extract raw email address*

Description

Strips the display name off an email address (if present).

Usage

```
raw(addr)
```

Arguments

addr	An address object.

Value

A raw email address.

Examples

```
gerry <- as.address("Gerald <gerry@gmail.com>")
raw(gerry)
```

---

render *Render Markdown into email*

Description

Render either Plain Markdown or R Markdown directly into the body of an email.

If input is a file then it will be interpreted as R Markdown it its extension is either "Rmd" or "Rmarkdown". Otherwise it will be processed as Plain Markdown.

Usage

```
render(
  msg,
  input,
  params = NULL,
  squish = TRUE,
  css_files = c(),
  include_css = c("rmd", "bootstrap", "highlight"),
  interpolate = TRUE,
  .open = "{{",
  .close = "}}",
  .envir = NULL
)
```

Arguments

msg	A message object.
input	The input Markdown file to be rendered or a character vector of Markdown text.
params	A list of named parameters that override custom parameters specified in the YAML front-matter.
squish	Whether to clean up whitespace in rendered document.
css_files	Extra CSS files.
include_css	Whether to include rendered CSS from various sources ("rmd" — native R Markdown CSS; "bootstrap" — Bootstrap CSS; "highlight" — highlight.js CSS).
interpolate	Whether or not to interpolate into input using glue.
.open	The opening delimiter.
.close	The closing delimiter.
.envir	Environment used for glue interpolation. Defaults to parent.frame().

Value

A message object.

Plain Markdown

Plain Markdown is processed with commonmark::markdown_html().

R Markdown

R Markdown is processed with rmarkdown::render().

Regardless of what output type is specified in the input file, render() will always use the "html_document" output format.

Examples

```
# Plain Markdown

markdown <- "[This](https://www.google.com) is a link."
filename <- "message.md"

# Render from Markdown in character vector.
msg <- envelope() %>% render(markdown)

# Create a file containing Markdown
cat(markdown, file = filename)

# Render from Markdown in file.
msg <- envelope() %>% render(filename)

# Cleanup.
file.remove(filename)

# R Markdown

filename <- "gh-doc.Rmd"

# Create an Rmd document from template.
rmarkdown::draft(
  filename,
  template = "github_document",
  package = "rmarkdown",
  edit = FALSE
)

# Check for suitable version of Pandoc (https://pandoc.org/).
#
# Need to have version 2.0 or greater to support required --quiet option.
#
pandoc <- rmarkdown::find_pandoc()
suitable_pandoc <- !is.null(pandoc$dir) && grepl("^2", pandoc$version)

# Render from Rmd file.
if (suitable_pandoc) {
  msg <- envelope() %>%
    render(filename, include_css = c("rmd", "highlight"))
}

# Cleanup.
file.remove(filename)
```

`server` *Create a SMTP server object.*

Description

Create a SMTP server object.

Usage

```
server(
  host,
  port = 25,
  username = NULL,
  password = NULL,
  insecure = FALSE,
  reuse = TRUE,
  helo = NA,
  pause_base = 1,
  max_times = 5,
  ...
)
```

Arguments

host	DNS name or IP address of the SMTP server.
port	Port that the SMTP server is listening on.
username	Username for SMTP server.
password	Password for SMTP server.
insecure	Whether to ignore SSL issues.
reuse	Whether the connection to the SMTP server should be left open for reuse.
helo	The HELO domain name of the sending host. If left as NA then will use local host name.
pause_base	Base delay (in seconds) for exponential backoff. See rate_backoff.
max_times	Maximum number of times to retry.
...	Additional curl options. See curl::curl_options() for a list of supported options.

Value

A function which is used to send messages to the server.

Examples

```
library(magrittr)

# Set parameters for SMTP server (with username and password)
smtp <- server(
  host = "smtp.gmail.com",
  port = 465,
  username = "bob@gmail.com",
  password = "bd40ef6d4a9413de9c1318a65cbae5d7"
```

```
)

# Set parameters for a (fake) testing SMTP server.
#
# More information about this service can be found at https://www.smtpbucket.com/.
#
smtp <- server(
  host = "mail.smtpbucket.com",
  port = 8025
)

# Create a message
msg <- envelope() %>%
  from("bob@gmail.com") %>%
  to("alice@yahoo.com")

# Send message (verbose output from interactions with server)
## Not run:
smtp(msg, verbose = TRUE)

## End(Not run)

# To confirm that the message was sent, go to https://www.smtpbucket.com/ then:
#
# - fill in "bob@gmail.com" for the Sender field and
# - fill in "alice@yahoo.com" for the Recipient field then
# - press the Search button.

# With explicit HELO domain.
#
smtp <- server(host = "mail.example.com",
               helo = "client.example.com")
```

`subject` *Add or query subject of message.*

Description

Add or query subject of message.

Usage

```
subject(
  msg,
  subject = NULL,
  interpolate = TRUE,
  .open = "{{",
  .close = "}}",
  .envir = NULL
)
```

Arguments

msg	A message object.
subject	A subject for the message.
interpolate	Whether or not to interpolate into input using glue.
.open	The opening delimiter.
.close	The closing delimiter.
.envir	Environment used for glue interpolation. Defaults to parent.frame().

Value

A message object or the subject of the message object (if subject is NULL).

See Also

to, from, cc, bcc and reply

Examples

```
library(magrittr)

# Create a message and set the subject
msg <- envelope() %>% subject("Updated report")

# Retrieve the subject for a message
subject(msg)
```

---

text *Add a text body to a message.*

Description

Uses glue::glue() to evaluate expressions enclosed in brackets as R code.

Usage

```
text(
  msg,
  content,
  disposition = "inline",
  charset = "utf-8",
  encoding = "7bit",
  interpolate = TRUE,
  .open = "{{",
  .close = "}}",
  .envir = NULL
)
```

Arguments

msg	A message object.
content	A string of message content.
disposition	How content is presented (Content-Disposition).
charset	How content is encoded.
encoding	How content is transformed to ASCII (Content-Transfer-Encoding).
interpolate	Whether or not to interpolate into input using glue.
.open	The opening delimiter.
.close	The closing delimiter.
.envir	Environment used for glue interpolation. Defaults to parent.frame().

Value

A message object.

See Also

html

Examples

```
library(magrittr)

msg <- envelope() %>% text("Hello!")

# Using {glue} interpolation.
#
name <- "Alice"
msg <- envelope() %>% text("Hello {name}.")

print(msg, details = TRUE)

# Disable {glue} interpolation.
#
msg <- envelope() %>% text("This is a set: {1, 2, 3}.", interpolate = FALSE)
```

Index