

Package ‘eventr’

October 13, 2022

Title Create Event Based Data Architectures

Version 1.0.0

Description Event-

driven programming is a programming paradigm where the flow of execution is defined by event. In this paradigm an event can be defined as ``a change in the state" of an object. This package offers a set of functions for creating event-based architectures using three basic functions: events, dispatchers, and handlers. The handlers manage the events, the dispatchers are in charge of redirecting the events to each of the handlers, finally the events are the objects that carry the information about the change of state.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports purrr, dplyr, magrittr

Suggests knitr, rmarkdown

NeedsCompilation no

Author Alvaro Franquet [aut, cre],
Maria Antonia Barceló [aut] (<<https://orcid.org/0000-0001-9720-690X>>),
Marc Saez [aut] (<<https://orcid.org/0000-0003-1882-0157>>),
Pere Plaja [aut]

Maintainer Alvaro Franquet <afranquet@salutemporda.cat>

Repository CRAN

Date/Publication 2020-05-26 14:40:02 UTC

R topics documented:

+ .event	2
+ .handler	3
add_event	4

add_handler	5
dispatcher	7
get_body	8
get_body_attr	9
get_dispatch	10
get_fun	11
get_handlers	12
get_header	13
get_id	14
get_time	14
get_type	15
handler	16
handlers_list	17
is_event	18
new_event	19
new_event_list	20

Index 21

+.event	<i>Add events to an event list</i>
---------	------------------------------------

Description

+ allows the user to add events or event_list object to an other event or event_list objects.

Usage

```
## S3 method for class 'event'
e1 + e2
```

Arguments

e1	An object of class event or event_list.
e2	An object of class event.

Value

Return an event_list object.

Examples

```
library(eventr)
library(magrittr)

birth_event <- event(
  id = 'first-id',
  type = 'BIRTH',
```

```
time = '1936-11-09',
birth_date = '1936-11-09'
)

measurement_event <- event(
  id = 'second-id',
  type = 'MEASUREMENT',
  time = '1937-11-09',
  weight = list(value = 10,2, unit = "kg"),
  height = list(value = 0,76, unit = "m")
)

death_event <- event(
  id = 'third-id',
  type = 'DEATH',
  time = '2019-05-22',
  death_date = '2019-05-22')

the_event_list <-
  birth_event +
  measurement_event +
  death_event
```

<code>+.handler</code>	<i>Add handlers to a handlers_list</i>
------------------------	--

Description

Merge two handler objects or add a handler in a handlers_list or dispatcher objects.

Usage

```
## S3 method for class 'handler'
e1 + e2
```

Arguments

e1 A handler, handler_list or dispatcher object.
e2 A handler or handlers_list object.

Value

Return a handlers_list object.

Examples

```
library(eventr)
library(magrittr)

birth_handler <- handler(type = 'BIRTH', FUN = function(obj, event){
  obj$birth_date <- get_body_attr(event, 'birth_date')
  return(obj)
})

measurement_handler <- handler(type = 'MEASUREMENT', FUN = function(obj, event){

  obj$weight <- get_body_attr(event, 'weight')
  obj$height <- get_body_attr(event, 'height')
  return(obj)

})

death_handler <- handler(type = 'DEATH', FUN = function(obj, event){
  obj$death_date <- get_body_attr(event, 'death_date')
  return(obj)
})

handlers <- birth_handler +
  measurement_handler +
  death_handler
```

add_event

Add Event

Description

this functions allows you to add events or event_list object to an other event or event_list objects.

Usage

```
add_event(e1, e2)
```

Arguments

e1 An object of class event or event_list.
e2 An object of class event.

Value

Return an event_list object.

Examples

```
library(eventr)
library(dplyr)
library(purrr)

birth_event <- event(
  id = 'first-id',
  type = 'BIRTH',
  time = '1936-11-09',
  birth_date = '1936-11-09'
)

measurement_event <- event(
  id = 'second-id',
  type = 'MEASUREMENT',
  time = '1937-11-09',
  weight = list(value = 10.2, unit = "kg"),
  height = list(value = 0.76, unit = "m")
)

death_event <- event(
  id = 'third-id',
  type = 'DEATH',
  time = '2019-05-22',
  death_date = '2019-05-22')

the_event_list <- birth_event %>%
  add_event(measurement_event) %>%
  add_event(death_event)

the_event_list <- birth_event +
  measurement_event +
  death_event

the_event_list

# transform the_event_list to data.frame

the_event_list %>%
  purrr::map(as.data.frame) %>%
  bind_rows()
```

add_handler

Add Handler

Description

Merge two handler objects or add a handler in a handlers_list or dispatcher objects.

Usage

```
add_handler(obj, .handler)
```

Arguments

obj	A handler, handler_list or dispatcher object.
.handler	A handler object.

Value

Return a handlers_list object.

Examples

```
library(eventr)
library(magrittr)

birth_handler <- handler(type = 'BIRTH', FUN = function(obj, event){
  obj$birth_date <- get_body_attr(event, 'birth_date')
  return(obj)
})

measurement_handler <- handler(type = 'MEASUREMENT', FUN = function(obj, event){

  obj$weight <- get_body_attr(event, 'weight')
  obj$height <- get_body_attr(event, 'height')
  return(obj)

})

death_handler <- handler(type = 'DEATH', FUN = function(obj, event){
  obj$death_date <- get_body_attr(event, 'death_date')
  return(obj)
})

handlers <- add_handler(birth_handler, death_handler)

handlers_01 <- birth_handler %>%
  add_handler(measurement_handler) %>%
  add_handler(death_handler)

handlers_02 <- birth_handler +
  measurement_handler +
  death_handler

identical(handlers_01, handlers_02)
```

dispatcher	<i>Dispatcher</i>
------------	-------------------

Description

Dispatcher Constructor.

Usage

```
dispatcher(handlers)
```

```
new_dispatcher(handlers)
```

```
validate_dispatcher(handlers)
```

Arguments

handlers A list of handlers build using handler() function.

Value

The dispatcher() function returns a dispatcher object. A dispatcher object contains two items: handlers and dispatch. handlers is a handlers_list object with all the handlers definitions. dispatch is a function with two params: obj and event. The dispatch() function allows the user to run a list of events using the handlers definitions.

Examples

```
library(eventr)
library(dplyr)

birth_event <- event(
  id = 'first-id',
  type = 'BIRTH',
  time = '1936-11-09',
  birth_date = '1936-11-09'
)

death_event <- event(
  id = 'second-id',
  type = 'DEATH',
  time = '2019-05-22',
  death_date = '2019-05-22'
)

set_birth_date <- function(obj, event){
  obj$birth_date <- get_body_attr(event, "birth_date")
  return(obj)
```

```
}

set_death_date <- function(obj, event){
  obj$death_date <- get_body_attr(event, "death_date")
  return(obj)
}

birth_handler <- handler(type = 'BIRTH', FUN = set_birth_date)
death_handler <- handler(type = 'DEATH', FUN = set_death_date)

handlers <- handlers_list(birth_handler, death_handler)

the_dispatcher <- dispatcher(handlers)

dispatch <- get_dispatch(the_dispatcher)

events <- event_list(birth_event, death_event)

the_obj <- dispatch(events = events, accumulate = FALSE)
the_obj

the_obj <- dispatch(events = events, accumulate = TRUE)
the_obj

# transform the_obj to data.frame
the_obj %>%
  purrr::map(as.data.frame) %>%
  bind_rows
```

get_body

Get Event Body

Description

This function returns the body of an event or list of events.

Usage

```
get_body(obj)
```

Arguments

obj An event or a list of events.

Value

The get_body function returns a list object

Examples

```
first_event <- event(  
  id = 'first-event',  
  type = 'FIRST_EVENT',  
  time = Sys.time(),  
  attr_01 = 'first-body-attr',  
  attr_02 = 'second-body-attr'  
)  
  
get_body(first_event)
```

get_body_attr

Get Event Body Attribute

Description

This function returns an attribute or list of attributes from the body of an event or list of events.

Usage

```
get_body_attr(obj, attr)
```

Arguments

obj An event or a list of events.
attr A character string indicating the attribute to return.

Value

get_body_attr() returns a single attribute from an event or an event_list object.

Examples

```
first_event <- event(  
  id = 'first-event',  
  type = 'FIRST_EVENT',  
  time = Sys.time(),  
  attr_01 = 'first-attribute-01'  
)  
  
second_event <- event(  
  id = 'second-event',  
  type = 'SECOND_EVENT',  
  time = Sys.time(),  
  attr_01 = 'first-attribute-02'  
)
```

```
the_event_list <- event_list(first_event, second_event)

get_type(the_event_list)
```

get_dispatch	<i>Get dispatch function</i>
--------------	------------------------------

Description

This function returns the dispatch function of a dispatcher type object.

Usage

```
get_dispatch(obj)
```

Arguments

obj An object of class dispatcher.

Value

This function returns a function with three parameters obj, events and accumulate. This function allows the user to evaluate a set of events.

Examples

```
library(eventr)
library(dplyr)

birth_event <- event(
  id = 'first-id',
  type = 'BIRTH',
  time = '1936-11-09',
  birth_date = '1936-11-09'
)

death_event <- event(
  id = 'second-id',
  type = 'DEATH',
  time = '2019-05-22',
  death_date = '2019-05-22'
)

set_birth_date <- function(obj, event){
  obj$birth_date <- get_body_attr(event, "birth_date")
  return(obj)
}
```

```
set_death_date <- function(obj, event){
  obj$death_date <- get_body_attr(event, "death_date")
  return(obj)
}

birth_handler <- handler(type = 'BIRTH', FUN = set_birth_date)
death_handler <- handler(type = 'DEATH', FUN = set_death_date)

handlers <- handlers_list(birth_handler, death_handler)

the_dispatcher <- dispatcher(handlers)

dispatch <- get_dispatch(the_dispatcher)

events <- event_list(birth_event, death_event)

the_obj <- dispatch(events = events, accumulate = FALSE)
the_obj

the_obj <- dispatch(events = events, accumulate = TRUE)
the_obj

# transform the_obj to data.frame
the_obj %>%
  purrr::map(as.data.frame) %>%
  bind_rows
```

get_fun

Get Function

Description

Get the function in a handler, handler_list or dispatcher objects.

Usage

```
get_fun(obj)
```

Arguments

obj An object of type handler, handler_list or dispatcher.

Value

Returns a function

Examples

```
set_birth_date <- function(obj, event){
  obj$birthDate <- event$body$birthDate
  return(obj)
}

set_death_date <- function(obj, event){
  obj$deathDate <- event$body$deathDate
  return(obj)
}

birth_handler <- handler(type = 'BIRTH', FUN = set_birth_date)
get_fun(birth_handler)

death_handler <- handler(type = 'DEATH', FUN = set_death_date)
get_fun(death_handler)

handlers <- handlers_list(birth_handler, death_handler)
get_fun(handlers)
```

get_handlers

Get handlers

Description

This function returns the list of handlers on a dispatcher type object.

Usage

```
get_handlers(obj)
```

Arguments

obj a dispatcher object.

Value

get_handlers() Returns an event object.

Examples

```
library(eventr)
library(dplyr)

birth_event <- event(
  id = 'first-id',
```

```
    type = 'BIRTH',
    time = '1936-11-09',
    birth_date = '1936-11-09'
  )

  death_event <- event(
    id = 'second-id',
    type = 'DEATH',
    time = '2019-05-22',
    death_date = '2019-05-22'
  )

  set_birth_date <- function(obj, event){
    obj$birth_date <- get_body_attr(event, "birth_date")
    return(obj)
  }

  set_death_date <- function(obj, event){
    obj$death_date <- get_body_attr(event, "death_date")
    return(obj)
  }

  birth_handler <- handler(type = 'BIRTH', FUN = set_birth_date)
  death_handler <- handler(type = 'DEATH', FUN = set_death_date)

  handlers <- handlers_list(birth_handler, death_handler)

  the_dispatcher <- dispatcher(handlers)

  the_handlers <- get_handlers(the_dispatcher)
```

get_header

Get Event Header

Description

This function returns the header of an event or list of events.

Usage

```
get_header(obj)
```

Arguments

obj An event or a list of events.

Value

Return a list with three elements: id, type and time.

Examples

```
first_event <- event(id = 'first-event', type = 'FIRST_EVENT', time = Sys.time())  
get_header(first_event)
```

get_id	<i>Get Event Identifier</i>
--------	-----------------------------

Description

This function returns the identifier of an event or list of events.

Usage

```
get_id(obj)
```

Arguments

obj An event or a list of events.

Value

get_id returns a character string.

Examples

```
first_event <- event(id = 'first-event', type = 'FIRST_EVENT', time = Sys.time())  
get_id(first_event)
```

get_time	<i>Get Event Time</i>
----------	-----------------------

Description

This function returns the time of an event or list of events.

Usage

```
get_time(obj)
```

Arguments

obj An event or a list of events.

Value

get_time() return a POSIXct object

Examples

```
first_event <- event(  
  id = 'first-event',  
  type = 'FIRST_EVENT',  
  time = Sys.time()  
)  
  
get_time(first_event)
```

get_type

Get Event Type

Description

This function returns the type of an event or list of events.

Usage

```
get_type(obj)
```

Arguments

obj An event or a list of events.

Value

get_type() return a character string

Examples

```
first_event <- event(id = 'first-event', type = 'FIRST_EVENT', time = Sys.time())  
second_event <- event(id = 'second-event', type = 'SECOND_EVENT', time = Sys.time())  
  
the_event_list <- event_list(first_event, second_event)  
  
get_type(the_event_list)
```

handler

Handler Constructor

Description

Construct a handler object.

Usage

```
handler(type, FUN)
```

```
new_handler(type, FUN)
```

```
validate_handler(type, FUN)
```

Arguments

type A character string indicating the event type to handle.

FUN A function which use an obj and an event as input parameters and returns an object as output.

Value

The handler() function returns an object of type handler. A handler object contains two elements: type and FUN. Type is a string indicating the type of event and FUN is a R function with to params: obj and event. obj is any R object and event is an event object returned by the event() function.

Examples

```
set_birth_date <- function(obj, event){  
  obj$birthDate <- event$body$birthDate  
  return(obj)  
}
```

```
set_death_date <- function(obj, event){  
  obj$deathDate <- event$body$deathDate  
  return(obj)  
}
```

```
birth_handler <- handler(type = 'BIRTH', FUN = set_birth_date)  
birth_handler
```

```
death_handler <- handler(type = 'DEATH', FUN = set_death_date)  
death_handler
```

handlers_list	<i>Handler List Constructor</i>
---------------	---------------------------------

Description

A handler list constructor. The function validates the structure of the input params and creates a handlers_list object.

Usage

```
handlers_list(...)
```

```
validate_handlers_list(...)
```

```
new_handlers_list(...)
```

Arguments

... A set of handler objects.

Value

The handlers_list() function returns a list of handlers objects.

Examples

```
measurement_handler <- handler(  
  type = 'MEASUREMENT',  
  FUN = function(obj, event) {  
    obj$measurement = get_attr(event, 'measurement')  
    return(obj)  
  }  
)  
  
bmi_handler <- handler(  
  type = 'BMI',  
  FUN = function(obj, event) {  
  
    obj$imc <- get_attr(event, 'mass') / get_attr(event, 'height')^2  
    return(obj)  
  
  }  
)  
  
handlers <- handlers_list(measurement_handler, bmi_handler)
```

is_event	<i>Is an Object from a class?</i>
----------	-----------------------------------

Description

Test if an object is some of the classes implemented in ‘eventr’ package.

Usage

```
is_event(x)
is_event_list(x)
is_handlers_list(x)
is_handler(x)
is_dispatcher(x)
```

Arguments

x any ‘R’ object.

Value

Returns a logical value

Examples

```
birth_event <- event(
  id = 'first-id',
  type = 'BIRTH',
  time = '1936-11-09',
  birth_date = '1936-11-09'
)

is_event(birth_event)
```

new_event	<i>Event Constructor</i>
-----------	--------------------------

Description

Event constructor

Usage

```
new_event(id = uuid::UUIDgenerate(), type, time = Sys.time(), ...)
```

```
validate_event(id, type, time, ...)
```

```
event(id = uuid::UUIDgenerate(), type, time, ...)
```

Arguments

id	A character identifier
type	A character indicating the event type
time	A character string indicating the event timestamp
...	A list with the event attributes.

Value

The function returns an object of class `event`. `event` objects are implemented as a list of two main elements: A head and a body. The head contains an identifier (`id`), a string indicating the event type (`type`) and a `POSIXct` object indicating when the event occurs (`time`). The body contains the event attributes defined by the user. By default `id` is generated using **uuid** and `time` value is the result of `Sys.time()` function by default.

Examples

```
birth_event <- event(  
  id = 'first-id',  
  type = 'BIRTH',  
  time = '1936-11-09',  
  birth_date = '1936-11-09'  
)  
  
death_event <- event(  
  id = 'second-id',  
  type = 'DEATH',  
  time = '2019-05-22',  
  death_date = '2019-05-22'  
)
```

new_event_list	<i>Event list</i>
----------------	-------------------

Description

Event list constructor. The function validates the structure of the input params and creates an event_list object.

Usage

```
new_event_list(...)
```

```
validate_event_list(...)
```

```
event_list(...)
```

Arguments

... A set of 'event' type objects.

Value

The event_list() function returns a list of event objects.

Examples

```
birth_event <- event(  
  id = 'first-id',  
  type = 'BIRTH',  
  time = '1936-11-09',  
  birth_date = '1936-11-09'  
)  
  
death_event <- event(  
  id = 'second-id',  
  type = 'DEATH',  
  time = '2019-05-22',  
  death_date = '2019-05-22'  
)  
  
the_event_list <- event_list(birth_event, death_event)
```

Index

`+.event`, [2](#)
`+.handler`, [3](#)

`add_event`, [4](#)
`add_handler`, [5](#)

`dispatcher`, [7](#)

`event (new_event)`, [19](#)
`event_list (new_event_list)`, [20](#)

`get_body`, [8](#)
`get_body_attr`, [9](#)
`get_dispatch`, [10](#)
`get_fun`, [11](#)
`get_handlers`, [12](#)
`get_header`, [13](#)
`get_id`, [14](#)
`get_time`, [14](#)
`get_type`, [15](#)

`handler`, [16](#)
`handlers_list`, [17](#)

`is_dispatcher (is_event)`, [18](#)
`is_event`, [18](#)
`is_event_list (is_event)`, [18](#)
`is_handler (is_event)`, [18](#)
`is_handlers_list (is_event)`, [18](#)

`new_dispatcher (dispatcher)`, [7](#)
`new_event`, [19](#)
`new_event_list`, [20](#)
`new_handler (handler)`, [16](#)
`new_handlers_list (handlers_list)`, [17](#)

`validate_dispatcher (dispatcher)`, [7](#)
`validate_event (new_event)`, [19](#)
`validate_event_list (new_event_list)`, [20](#)
`validate_handler (handler)`, [16](#)
`validate_handlers_list (handlers_list)`,
[17](#)