Package 'SSEparser'

July 21, 2025

Title Parse Server-Sent Events

Version 0.1.0

Description Functionality to parse server-sent events with a high-level interface that can be extended for custom applications.

License MIT + file LICENSE

URL https://github.com/calderonsamuel/SSEparser, https://calderonsamuel.github.io/SSEparser/

BugReports https://github.com/calderonsamuel/SSEparser/issues

Imports magrittr, purrr, R6, rlang, stringr

Suggests jsonlite, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Samuel Calderon [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-6847-1210>)

Maintainer Samuel Calderon <samuel.calderon@uarm.pe>

Repository CRAN

Date/Publication 2023-12-14 16:30:06 UTC

Contents

parse_sse *Parse Server-Sent Events*

Description

This functions converts Server-Sent Events to a R list. This a wrapper function for the lower level SSEparser R6 class. A single string can contain multiple SSEs.

Usage

```
parse_sse(event)
```

Arguments

event	A length 1 string containing a server sent event as specified in the HTML spec.

Value

An R list on which each element is an event

Examples

```
event <- "data: test\nevent: message\nid: 123\n\n"
parse_sse(event)

with_comment <- "data: test\n: comment\nevent: example\n\n"
parse_sse(with_comment)
```

SSEparser *Parse a Server Sent Event*

Description

This class can help you parse a single server sent event or a stream of them. You can inherit the class for a custom application. The `parse_sse()` function wraps this class for a more *functional* approach.

Details

The HTML specification tells us that event streams are composed by chunks (also called *blocks*, or *messages*) and lines. A single new line character (`\n`) states the end of a line, and two consecutive new line characters (`\n\n`) state the end of a chunk.

This means that, in practice, an event can be composed of one or more chunks, and a chunk can be composed of one or more lines.

```
data: This is the first chunk, it has one line

data: This is the second chunk
extra: It has two lines

data: This is the third chunk, it has an id field. This is common.
id: 123

: Lines that start with a colon are comments, they will be ignored
data: This is the forth chunk, it has a comment

data: This is the fifth chunk. Normally you will receive a data field
custom: But the server can send custom field names. SSEparser parses them too.
```

Typically, an event stream will send a single chunk for event, but it is important to understand that event != chunk because `SSEparser$events` will be a list of all the chunks received as it makes a more consistent output.

Value

An object with R6 class `SSEparser`

Public fields

`events` List that contains all the events parsed. When the class is initialized, is just an empty list.

Methods

Public methods:

- `SSEparser$append_parsed_sse()`
- `SSEparser$parse_sse()`
- `SSEparser$new()`
- `SSEparser$clone()`

Method `append_parsed_sse()`**:** Takes a parsed event and appends it to the `events` field. You can overwrite this method if you decide to extend this class.

Usage:

```
SSEparser$append_parsed_sse(parsed_event)
```

Arguments:

`parsed_event` Event to append to the `events` field.

Method `parse_sse()`**:** Takes a string that comes from a server sent event and parses it to an R list. You should never overwrite this method.

Usage:

```
SSEparser$parse_sse(event)
```

Arguments:

`event` A length 1 string containing a server sent event as specified in the HTML spec.

Method `new()`**:** Create a new SSE parser

Usage:

```
SSEparser$new()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SSEparser$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
example_event <-
"data: This is the first chunk, it has one line

data: This is the second chunk
extra: It has two lines

data: This is the third chunk, it has an id field. This is common.
id: 123

: Lines that start with a colon are comments, they will be ignored
data: This is the fourth chunk, it has a comment

data: This is the fifth chunk. Normally you will receive a data field
custom: But the server can send custom field names. SSEparser parses them too."

parser <- SSEparser$new()
parser$parse_sse(example_event)

str(parser$events)
```

Index