Package: eburones (via r-universe)

March 13, 2025

Title User Session for 'Ambiorix'

Version 0.0.0.9000

Description Track user sessions in 'Ambiorix' applications.

License GPL (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports DBI, methods, fastmap, ambiorix (>= 2.0.0.9000)

Suggests scilis, mongolite

Remotes devOpifex/scilis

Config/pak/sysreqs make libssl-dev zlib1g-dev

Repository https://ambiorix-web.r-universe.dev

RemoteUrl https://github.com/ambiorix-web/eburones

RemoteRef HEAD

RemoteSha 47ab2673251c4ce680482f2bc8baf358f3f22f53

Contents

DBI *DBI Backend*

Description

DBI backend for sessions.

Methods

Public methods:

- DBI$new()
- DBI$set()
- DBI$has()
- DBI$get()
- DBI$clone()

Method new():

Usage:

```
DBI$new(con, table)
```

Arguments:

`con` Live DBI connection.

`table` Name of table to store sessions. The table must exists, it must also contain the expected columns, as well as a `key` column where the user token will be stored.

Details: Constructor

Method set():

Usage:

```
DBI$set(key, value)
```

Arguments:

`key` Key of the `value`.

`value` Value to store.

Details: Add a key-value pair to the map.

Method has():

Usage:

```
DBI$has(key = NULL)
```

Arguments:

`key` Key of the `value`.

Details: Check whether a key is present in the map.

Method get():

Usage:

```
DBI$get(key)
```

Arguments:

`key` Key of the `value`.

Details: Retrieve a value given its key.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`eburones` *Eburones*

Description

Session middleware.

Usage

```
eburones(
  ...,
  name = "session",
  backend = Local$new(),
  callback = function(req, res) list(),
  identifier = token_create
)
```

Arguments

`...`	Passed to the `cookie` method of the ambiorix::Response class.
`name`	Cookie name.
`backend`	Storage class to keep track of callbacks.
`callback`	Function to run when a callback is created or retrieved.
`identifier`	Function that creates a unique token. This is run when creating a new session.

`Local` *Local backend*

Description

Local backend for sessions. Do not use this in production, it will likely leka memory and will not track sessions on load balancers.

Methods

Public methods:

- `Local$new()`
- `Local$set()`
- `Local$has()`
- `Local$get()`
- `Local$clone()`

Method `new()`:

Usage:

```
Local$new()
```

Details: Constructor

Method `set()`:

Usage:

```
Local$set(key, value)
```

Arguments:

`key` Key of the `value`.

`value` Value to store.

Details: Add a key-value pair to the map.

Method `has()`:

Usage:

```
Local$has(key)
```

Arguments:

`key` Key of the `value`.

Details: Check whether a key is present in the map.

Method `get()`:

Usage:

```
Local$get(key)
```

Arguments:

`key` Key of the `value`.

Details: Retrieve a value given its key.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Mongo` *Mongo backend*

Description

Mongo db backend for sessions.

Methods

Public methods:

- `Mongo$new()`
- `Mongo$set()`
- `Mongo$has()`
- `Mongo$get()`
- `Mongo$clone()`

Method `new()`**:**

Usage:

```
Mongo$new(con, key)
```

Arguments:

`con` Live DBI connection.

`key` Name of key to use internally.

Details: Constructor

Method `set()`**:**

Usage:

```
Mongo$set(key, value)
```

Arguments:

`key` Key of the `value`.

`value` Value to store.

Details: Add a key-value pair to the map.

Method `has()`**:**

Usage:

```
Mongo$has(key = NULL)
```

Arguments:

`key` Key of the `value`.

Details: Check whether a key is present in the map.

Method `get()`**:**

Usage:

```
Mongo$get(key)
```

Arguments:

`key` Key of the `value`.

Details: Retrieve a value given its key.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Index