

# Package ‘firesale’

December 11, 2025

**Title** Datastore for 'fiery' Web Servers

**Version** 0.1.1

**Description** Provides a persistent datastore for 'fiery' apps. The datastore is build on top of the 'storr' package and can thus be based on a variety of backends. The datastore contains both a global and session-scoped section.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** cli, R6, reqres, rlang, storr

**Suggests** fiery (>= 1.3.0), later, testthat (>= 3.0.0)

**URL** <https://github.com/thomasp85/firesale>

**BugReports** <https://github.com/thomasp85/firesale/issues>

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Thomas Lin Pedersen [aut, cre] (ORCID: <https://orcid.org/0000-0002-5147-4711>),  
Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

**Maintainer** Thomas Lin Pedersen <[thomas.pedersen@posit.co](mailto:thomas.pedersen@posit.co)>

**Repository** CRAN

**Date/Publication** 2025-12-11 06:10:25 UTC

## Contents

FireSale . . . . .	2
Index	5

---

FireSale

*A FireSale plugin*


---

### Description

A FireSale plugin

A FireSale plugin

### Details

The class encapsulates the firesale functionality into a fiery plugin. You use it by creating and attaching it to a fiery server object.

### Initialization

A new 'FireSale'-object is initialized using the `new()` method on the generator (shown here with the environment driver):

#### Usage

```
datastore <- FireSale$new(storr::driver_environment())
```

### Fiery plugin

This class is mainly intended to be used as a fiery plugin, by attaching it to a fiery server object. It works by providing a datastore element (name can be modified with the `arg_name` argument during initialization) in the `arg_list` argument to request handlers. The object contains two elements, `global` and `session`. The first contains data shared by all sessions, while the latter is scoped to the current session. Both of these elements are list-like, but in reality are interfaces to the underlying data store

### Active bindings

`name` The name of the plugin

`arg_name` The name of the argument that will contain the data store

### Methods

#### Public methods:

- `FireSale$new()`
- `FireSale$format()`
- `FireSale$get_mall()`
- `FireSale$shiny_mall()`
- `FireSale$on_attach()`
- `FireSale$clone()`

**Method** `new()`: Initializes a new FireSale object

*Usage:*

```
FireSale$new(
  driver,
  arg_name = "datastore",
  gc_interval = 3600,
  max_age = gc_interval
)
```

*Arguments:*

**driver** A storrr driver to use for the backend

**arg\_name** A string giving the name under which the data store should appear in the `arg_list` argument

**gc\_interval** The interval with which the backend should be garbage collected. The value is indicative and a garbage collection may happen at longer intervals

**max\_age** The maximum age in second an ID can be left unused before being purged. The value is indicative and a stale ID store may linger longer than this

**Method** `format()`: Textual representation of the plugin

*Usage:*

```
FireSale$format(...)
```

*Arguments:*

`...` ignored

**Method** `get_mall()`: Create a mall (a collection of storefronts) containing a global and a session-specific storefront

*Usage:*

```
FireSale$get_mall(id)
```

*Arguments:*

**id** The session id of the current session

**Method** `shiny_mall()`: Create a mall from a shiny session object. If the shiny app has been launched from a plumber2 server the session id is automatically resolved. If not, you must provide an id function that extracts the session id from a `reqres::Request` object.

*Usage:*

```
FireSale$shiny_mall(session, id_fun = NULL)
```

*Arguments:*

**session** A ShinySession object

**id\_fun** A function that can extract the session ID from a Request object. This is handled automatically for shiny apps launched from a plumber2 server. The default id function for fiery servers is constructed with `fiery::session_id_cookie()`

**Method** `on_attach()`: Method for use by fiery when attached as a plugin. Should not be called directly.

*Usage:*

```
FireSale$on_attach(app, ...)
```

*Arguments:*

app The fiery server object  
... Ignored

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

*Usage:*

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

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```
# Create a datastore object
ds <- FireSale$new(storr::driver_environment())

# Attach it to a fiery server
app <- fiery::Fire$new()

app$attach(ds)
```

# Index

FireSale, [2](#)

reqres::Request, [3](#)