Package 'featureflag'

May 8, 2026

Title Turn Features On and Off using Feature Flags

Version 0.2.0

Description Feature flags allow developers to turn features of their software on and off in form of configuration. This package provides functions for creating feature flags in code. It exposes an interface for defining own feature flags which are enabled based on custom criteria.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), spelling, knitr, rmarkdown, shiny, withr

Language en-US

URL https://github.com/szymanskir/featureflag

BugReports https://github.com/szymanskir/featureflag/issues

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Ryszard Szymański [aut, cre]

Maintainer Ryszard Szymański <ryszard.szymanski@outlook.com>

Repository CRAN

Date/Publication 2025-03-22 22:30:01 UTC

Contents

create_bool_feature_flag 2
create_connect_group_feature_flag 3
create_connect_user_feature_flag 3
create_env_var_feature_flag 4
create_feature_flag 4
create_percentage_feature_flag 5
create_time_period_feature_flag 5

feature_if 6
feature_ifelse 7
is_enabled 8
is_enabled.bool_feature_flag 8
is_enabled.connect_group_feature_flag 9
is_enabled.connect_user_feature_flag 10
is_enabled.env_var_feature_flag 11
is_enabled.percentage_feature_flag 12
is_enabled.time_period_feature_flag 12

Index **14**

`create_bool_feature_flag` *Creates an instance of a bool feature flag with the specified bool value.*

Description

Creates an instance of a bool feature flag with the specified bool value.

Usage

```
create_bool_feature_flag(value)
```

Arguments

`value`	single logical determining whether the flag should be enabled

Value

feature flag object of the bool value

Examples

```
{
  enabled_flag <- create_bool_feature_flag(TRUE)
  disabled_flag <- create_bool_feature_flag(FALSE)
}
```

create_connect_group_feature_flag

Creates an instance of a connect feature flag that is enabled for specific groups

Description

Creates an instance of a connect feature flag that is enabled for specific groups

Usage

```
create_connect_group_feature_flag(groups)
```

Arguments

groups	groups for which the feature flag should be enabled

Value

feature flag that is enabled for specific groups

Examples

```
{
  connect_group_flag <- create_connect_group_feature_flag(groups = c("group1", "group2"))
}
```

create_connect_user_feature_flag

Creates an instance of a connect feature flag that is enabled for specific users

Description

Creates an instance of a connect feature flag that is enabled for specific users

Usage

```
create_connect_user_feature_flag(users)
```

Arguments

users	users for which the feature flag should be enabled

Value

feature flag that is enabled for specific users

Examples

```
{
  connect_user_flag <- create_connect_user_feature_flag(users = c("user1", "user2"))
}
```

`create_env_var_feature_flag` *Creates an instance of a feature flag that is enabled based on an environment variable*

Description

Creates an instance of a feature flag that is enabled based on an environment variable

Usage

```
create_env_var_feature_flag(env_var)
```

Arguments

`env_var`	Name of the environment variable

Value

Feature flag that is enabled based on the specified environment variable

Examples

```
{
  env_flag <- create_env_var_feature_flag(env_var = "FEATURE_X")
}
```

`create_feature_flag` *Creates the base of a feature flag.*

Description

It should not be used directly, but only as a prerequisite when creating concrete feature flag.

Usage

```
create_feature_flag()
```

Value

instance of a base feature flag.

create_percentage_feature_flag

Creates an instance of a percentage feature flag with a specified chance of being enabled

Description

Creates an instance of a percentage feature flag with a specified chance of being enabled

Usage

```
create_percentage_feature_flag(percentage)
```

Arguments

percentage	chance of being enabled e.g. 1 for always being enabled

Value

feature flag object of the percentage type

Examples

```
{
  always_enabled_flag <- create_percentage_feature_flag(percentage = 1)
  randomly_enabled_flag <- create_percentage_feature_flag(percentage = 0.5)
}
```

create_time_period_feature_flag

Creates an instance of a time period feature flag.

Description

Creates an instance of a time period feature flag.

Usage

```
create_time_period_feature_flag(from = NULL, to = NULL)
```

Arguments

from	date-time from which the feature flag should be enabled set as null if you want a one sided boundary.
to	date-time to which the feature flag should be enabled set as null if you want a one sided boundary

Details

Boundaries are set as inclusive

Examples

```
{
  two_sided_flag <- create_time_period_feature_flag(
    from = ISOdatetime(2020, 10, 10, 0, 0, 0, tz = "UTC"),
    to = ISOdatetime(2020, 11, 10, 0, 0, 0, tz = "UTC")
  )

  left_sided_flag <- create_time_period_feature_flag(
    from = ISOdatetime(2020, 10, 10, 0, 0, 0, tz = "UTC")
  )

  right_sided_flag <- create_time_period_feature_flag(
    to = ISOdatetime(2020, 10, 10, 0, 0, 0, tz = "UTC")
  )
}
```

---

feature_if *Evaluates the provided expression if the feature flag is enabled.*

Description

Evaluates the provided expression if the feature flag is enabled.

Usage

```
feature_if(feature_flag, expr)
```

Arguments

feature_flag	flag which defines whether the provided expression should be evaluated
expr	expression to evaluate when the feature_flag is enabled

Details

The passed expression is evaluated in the frame where `feature_if` is called.

Value

If the passed `feature_flag` is enabled, than the result of the evaluation of the passed expression is returned. Otherwise there is no return value.

Examples

```
{
  flag <- create_bool_feature_flag(TRUE)

  feature_if(flag, {
    2 + 7
  })
}
```

`feature_ifelse` *Evaluates one or the other expression based on whether the feature flag is enabled.*

Description

Evaluates one or the other expression based on whether the feature flag is enabled.

Usage

```
feature_ifelse(feature_flag, true_expr, false_expr)
```

Arguments

`feature_flag`	flag which defines which expression should be evaluated
`true_expr`	expression to evaluate when the feature_flag is enabled
`false_expr`	expression to evaluate when the feature_flag is disabled

Details

The passed expression is evaluated in the frame where `feature_ifelse` is called.

Value

The result of evaluating `true_expr` is returned if the passed feature_flag is enabled. Otherwise the result of evaluating `false_expr` is returned.

Examples

```
{
  flag <- create_bool_feature_flag(TRUE)

  feature_ifelse(
    flag,
    2 * 7,
    3 * 7
  )
}
```

`is_enabled` *Checks if the given feature flag is enabled.*

Description

Checks if the given feature flag is enabled.

Usage

```
is_enabled(feature_flag)
```

Arguments

`feature_flag`	feature flag to be tested whether it is enabled

Value

TRUE if the feature flag is enabled.

`is_enabled.bool_feature_flag` *Checks if the given bool feature flag is enabled*

Description

Checks if the given bool feature flag is enabled

Usage

```
## S3 method for class 'bool_feature_flag'
is_enabled(feature_flag)
```

Arguments

`feature_flag`	flag to be checked whether it is enabled

Value

TRUE if the feature flag is enabled.

Examples

```
{
  enabled_flag <- create_bool_feature_flag(TRUE)

  if (is_enabled(enabled_flag)) {
    print("The flag is enabled!")
  }
}
```

`is_enabled.connect_group_feature_flag` *Checks if the given connect group feature flag is enabled*

Description

Checks if the given connect group feature flag is enabled

Usage

```
## S3 method for class 'connect_group_feature_flag'
is_enabled(feature_flag)
```

Arguments

`feature_flag`	flag to be checked whether it is enabled

Details

The session$groups field is used for retrieving the information on the logged-in user groups

Value

TRUE if the logged in user belongs to a group defined in the feature flag

Examples

```
{
  flag <- create_connect_group_feature_flag(c("group1"))

  # Returns TRUE when the logged-in user belongs to at least one of the specified groups
  mock_session <- shiny::MockShinySession$new()
  mock_session$groups <- "group1"
  shiny::withReactiveDomain(
    domain = mock_session,
    expr = is_enabled(flag)
  )

  # Returns FALSE if session$groups does not have any of the specified groups
  mock_session <- shiny::MockShinySession$new()
```

```
  mock_session$user <- "group2"
  shiny::withReactiveDomain(
    domain = mock_session,
    expr = is_enabled(flag)
  )
}
```

`is_enabled.connect_user_feature_flag` *Checks if the given connect user feature flag is enabled*

Description

Checks if the given connect user feature flag is enabled

Usage

```
## S3 method for class 'connect_user_feature_flag'
is_enabled(feature_flag)
```

Arguments

`feature_flag`	flag to be checked whether it is enabled

Details

The session$user field is used for retrieving the information on the logged-in user

Value

TRUE if the feature flag is enabled.

Examples

```
{
  flag <- create_connect_user_feature_flag(c("user1"))

  # Returns TRUE if the session$user matches the specified users
  mock_session <- shiny::MockShinySession$new()
  mock_session$user <- "user1"
  shiny::withReactiveDomain(
    domain = mock_session,
    expr = is_enabled(flag)
  )

  # Returns FALSE if the session$user does not match the specified users
  mock_session <- shiny::MockShinySession$new()
  mock_session$user <- "user2"
  shiny::withReactiveDomain(
```

```
    domain = mock_session,
    expr = is_enabled(flag)
  )
}
```

`is_enabled.env_var_feature_flag` *Checks if the given environment variable feature flag is enabled*

Description

Checks if the given environment variable feature flag is enabled

Usage

```
## S3 method for class 'env_var_feature_flag'
is_enabled(feature_flag)
```

Arguments

`feature_flag`	Flag to be checked whether it is enabled

Value

TRUE if the environment variable is set to 'true'

Examples

```
{
  flag <- create_env_var_feature_flag("FEATURE_X")

  withr::with_envvar(new = list(FEATURE_X = "true"), {
    is_enabled(flag) # Returns TRUE
  })

  is_enabled(flag) # Returns FALSE by default
}
```

is_enabled.percentage_feature_flag

Checks if the given percentage flag is enabled

Description

Checks if the given percentage flag is enabled

Usage

```
## S3 method for class 'percentage_feature_flag'
is_enabled(feature_flag)
```

Arguments

feature_flag	flag to be checked whether it is enabled

Value

TRUE if the feature flag is enabled.

Examples

```
{
  enabled_flag <- create_percentage_feature_flag(1)

  if (is_enabled(enabled_flag)) {
    print("The flag is enabled!")
  }
}
```

is_enabled.time_period_feature_flag

Checks if the given bool feature flag is enabled

Description

Checks if the given bool feature flag is enabled

Usage

```
## S3 method for class 'time_period_feature_flag'
is_enabled(feature_flag)
```

Arguments

feature_flag	flag to be checked whether it is enabled

Value

TRUE if the feature flag is enabled.

Examples

```
{
  feature_flag <- create_time_period_feature_flag(
    from = ISOdatetime(2020, 10, 10, 0, 0, 0, tz = "UTC")
  )

  if (is_enabled(feature_flag)) {
    print("The flag is enabled!")
  }
}
```

Index

create_bool_feature_flag, 2
create_connect_group_feature_flag, 3
create_connect_user_feature_flag, 3
create_env_var_feature_flag, 4
create_feature_flag, 4
create_percentage_feature_flag, 5
create_time_period_feature_flag, 5

feature_if, 6
feature_ifelse, 7

is_enabled, 8
is_enabled.bool_feature_flag, 8
is_enabled.connect_group_feature_flag, 9
is_enabled.connect_user_feature_flag, 10
is_enabled.env_var_feature_flag, 11
is_enabled.percentage_feature_flag, 12
is_enabled.time_period_feature_flag, 12