Package 'registry'

May 9, 2026

Type Package

Title Infrastructure for R Package Registries

Version 0.5-1

Author David Meyer

Maintainer David Meyer <David.Meyer@R-project.org>

Description Provides a generic infrastructure for creating and using registries.

Depends R (>= 2.6.0)

Imports utils

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2019-03-05 14:21:41 UTC

Contents

`matchfuns` *Matching functions*

Description

Functions used for lookups of search keys.

Usage

```
match_ignorecase(lookup, entry, ...)
match_exact(lookup, entry, ...)
match_partial(lookup, entry, ...)
match_partial_ignorecase(lookup, entry, ...)
match_regexp(lookup, entry, ...)
```

Arguments

lookup	Search value to look up (for some key field).
entry	Vector of key values where lookup is sought.
...	For match_ignorecase and match_exact: currently not used. For match_partial and match_partial_ignorecase: additional arguments passed to pmatch. For match_regexp: additional arguments passed to grep.

Details

These are matching functions to be specified for key fields, controlling how search values are looked up in the registry.

Author(s)

David Meyer <David.Meyer@R-project.org>

See Also

regobj

Examples

```
## use exact matching
R <- registry(stop_if_missing = FALSE)
R$set_field("Key", type = "character", is_key = TRUE, index_FUN = match_exact)
R$set_field("Value", type = "numeric")
R$set_entry("the key", 1)

R[["the key"]]
R[["the"]]

## use partial matching
R <- registry()
R$set_field("Key", type = "character", is_key = TRUE, index_FUN = match_partial)
R$set_field("Value", type = "numeric")
R$set_entry("the key", 1)

R[["the"]]

## use regular expressions
R <- registry()
R$set_field("Key", type = "character", is_key = TRUE, index_FUN = match_regexp)
```

```
R$set_field("Value", type = "numeric")
R$set_entry("the key", 1)
R$set_entry("key", 2)

R[["k.*"]]
R["k.*"]
```

`registry` *Registry creator*

Description

Function to create a registry object.

Usage

```
registry(entry_class = NULL, registry_class = NULL,
         validity_FUN = NULL, stop_if_missing = FALSE)
```

Arguments

`entry_class`	character string indicating a class the returned registry object will additionally inherit from (optional). Used for dispatching user-specified print and summary methods.
`registry_class`	character string indicating a class the registry entries will additionally inherit from (optional). Used for dispatching user-specified print and summary methods.
`validity_FUN`	a function accepting a new registry entry as argument for checking its validity and possibly aborting with an error message. The entry will be provided by the calling function as a list with named components (fields).
`stop_if_missing`	logical indicating whether the registry lookup functions should abort or just return `NULL` in case of no match.

Details

This is a generating function that will return a registry object whose components are accessor functions for the contained data. These are documented separately (`regobj`).

Author(s)

David Meyer <David.Meyer@R-project.org>

See Also

`regobj`

Examples

```
R <- registry()

R$set_field("X", type = TRUE)
R$set_field("Y", type = "character")
R$set_field("index", type = "character", is_key = TRUE,
            index_FUN = match_partial_ignorecase)
R$set_field("index2", type = "integer", is_key = TRUE)

R$set_entry(X = TRUE, Y = "bla", index = "test", index2 = 1L)
R$set_entry(X = FALSE, Y = "foo", index = c("test", "bar"), index2 = 2L)

R$get_entries("test")
R[["test", 1]]
R["test"]
R[["test"]]
```

---

regobj *Registry object*

Description

Registry object.

Usage

```
regobj$get_field(name)
regobj$get_fields()
regobj$get_field_names()
regobj$set_field(name,
                 type = NA, alternatives = NA, default = NA,
                 is_mandatory = FALSE, is_modifiable = TRUE,
                 is_key = FALSE, validity_FUN = NULL,
                 index_FUN = match_ignorecase, ...)

regobj$has_entry(key)
regobj$get_entry(...)
regobj$get_entries(...)
regobj$grep_entries(pattern, ...)
regobj$get_entry_names()
regobj$set_entry(...)
regobj$modify_entry(...)
regobj$delete_entry(...)
regobj$n_of_entries(name)
regobj$get_field_entries(field, unlist = TRUE)

regobj$get_permissions()
regobj$restrict_permissions(set_entries = TRUE,
```

```
          modify_entries = TRUE, delete_entries = TRUE, set_fields = TRUE)
regobj$seal_entries()
regobj$get_sealed_field_names()


## S3 method for class 'registry'
print(x, ...)
## S3 method for class 'registry'
summary(object, ...)

## S3 method for class 'registry'
x[[...]]
## S3 method for class 'registry'
x[...]
```

Arguments

name	character string representing the name of an entry (case-insensitive).
pattern	regular expression to be matched to all fields of class "character" in all entries.
type	character vector specifying accepted classes for this field. If NA (default), any class will be accepted. If type is not a character vector, the class will be inferred from the argument given.
alternatives	vector of alternatives accepted for this field.
default	optional default value for the field.
is_mandatory	logical specifying whether new entries are required to have a value for this field.
is_modifiable	logical specifying whether entries can be changed with respect to that field.
is_key	logical indicating whether the field is (part of) an index.
validity_FUN	optional function or character string with the name of a function that checks the validity of a field entry. Such a function gets the value to be investigated as argument, and should stop with an error message if the value is not correct.
index_FUN	vectorized predicate function matching an index value to a vector (of existing field entries). See matchfuns.
x, object	a registry object.
...	for regobj$set_entry and regobj$modify_entry: named list of fields to be modified in or added to the registry, including the index field(s) (see details). For grep_entries: additional parameters passed to grep. For set_field: additional parameters passed to the specified match function. For get_entry, get_entries and the indexing functions: key values for the entry (entries) to be looked up.

Details

regobj represents a registry object returned by registry whose elements can be processed using the following accessor functions:

get_field_names() returns a character vector with all field names. get_field() returns the information for a specific field as a list with components named as described above. get_fields()

returns a list with all field entries. `set_field()` is used to create new fields in the repository (the default value will be set in all entries).

`get_entry_names()` returns a character vector with (the first alias of) all entries. `entry_exists()` is a predicate checking if an entry with the specified alias exists in the registry. `get_entry()` returns the first specified entry if at least one exists (and, by default, gives an error if it does not). `get_entries()` is used to query more than one entry matching the index (named argument list) exactly. `grep_entries()` returns those entries where the regular expression in `pattern` matches *any* character field in an entry. By default, all values are returned. `delete_entry` removes an existing entry from the registry.

`set_entry`, `delete_entry` and `modify_entry` require a named list of arguments used as field entries. At least the index fields are required.

`set_entry` will check for all other mandatory fields. If specified in the field meta data, each field entry and the entry as a whole is checked for validity. Note that it is possible to specify a vector of values for an index field, treated as alternative keys for this entry.

It is possible to *restrict* permissions (for setting, getting, deleting and modifying entries) using `restrict_permissions`. Further, a user can *seal* the current registry state (fields, entries) so that *existing* structure and information becomes immutable. Additional fields and entries can be changed according to the permissions set. Permissions and sealing are useful for exported registry objects to control the users' capabilities of modifying/extending them.

Author(s)

David Meyer <David.Meyer@R-project.org>

See Also

`dist`, matchfuns

Examples

```
regobj <- registry()
regobj$set_field("X", type = TRUE)
regobj$set_field("Y", type = "character")
regobj$set_field("index", type = "character", is_key = TRUE,
            index_FUN = match_partial_ignorecase)
regobj$set_field("index2", type = "integer", is_key = TRUE)
regobj$set_entry(X = TRUE, Y = "bla", index = "test", index2 = 1L)
regobj$set_entry(X = FALSE, Y = "foo", index = c("test", "bar"), index2 = 2L)
regobj$get_entries("test")
regobj[["test", 1]]
regobj["test"]
regobj[["test"]]
```

Index