Package ‘registry’

March 5, 2019

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

R topics documented:

matchfuns ......................................................... 1
registry .......................................................... 3
regobj ........................................................... 4

Index

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_partialignorecase(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_partialignorecase: 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)
registry

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

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

---

**Description**

Function to create a registry object.

**Usage**

```r
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
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**

```r
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, ...)
```

```r
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)
```

```r
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.

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. \texttt{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 \textit{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 \textit{seal} the current registry state (fields, entries) so that \textit{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/extendng them.

\textbf{Author(s)}

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

\textbf{See Also}

dist, matchfuns

\textbf{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

*Topic data
  matchfuns, 1
  registry, 3
  regobj, 4
  [.registry (regobj), 4
  [[.[registry (regobj), 4

dist, 6

grep, 2, 5

match_exact (matchfuns), 1
match_ignorecase (matchfuns), 1
match_partial (matchfuns), 1
match_partial_ignorecase (matchfuns), 1
match_regexp (matchfuns), 1
matchfuns, 1, 5, 6

pmatch, 2
print.registry (regobj), 4

registry, 3, 5
regobj, 2, 3, 4

summary.registry (regobj), 4