Package 'biodbKegg'

November 23, 2023

Title biodbKegg, a library for connecting to the KEGG Database

Version 1.8.0

Description The biodbKegg library is an extension of the biodb framework package that provides access to the KEGG databases Compound, Enzyme, Genes, Module, Orthology and Reaction. It allows to retrieve entries by their accession numbers. Web services like ``find'', ``list'' and ``findExactMass'' are also available. Some functions for navigating along the pathways have also been implemented.

URL https://github.com/pkrog/biodbKegg

BugReports https://github.com/pkrog/biodbKegg/issues

biocViews Software, Infrastructure, DataImport, Pathways, KEGG

Depends R (>= 4.1)

License AGPL-3

Encoding UTF-8

VignetteBuilder knitr

Suggests BiocStyle, roxygen2, devtools, testthat (>= 2.0.0), knitr, rmarkdown, igraph, magick, lgr

Imports R6, biodb (>= 1.4.2), chk, lifecycle

RoxygenNote 7.2.1

Collate 'KeggShape.R' 'KeggCircle.R' 'KeggConn.R' 'KeggCompoundConn.R' 'KeggEntry.R' 'KeggCompoundEntry.R' 'KeggEnzymeConn.R' 'KeggEnzymeEntry.R' 'KeggGenesConn.R' 'KeggGenesEntry.R' 'KeggGlycanConn.R' 'KeggGlycanEntry.R' 'KeggModuleConn.R' 'KeggModuleEntry.R' 'KeggOrthologyConn.R' 'KeggOrthologyEntry.R' 'KeggPathwayConn.R' 'KeggPathwayEntry.R' 'KeggReactionConn.R' 'KeggReactionEntry.R' 'KeggRect.R' 'package.R'

git_url https://git.bioconductor.org/packages/biodbKegg

git_branch RELEASE_3_18

git_last_commit 6c54c2d

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2023-11-23

Author Pierrick Roger [aut, cre] (<https://orcid.org/0000-0001-8177-4873>)

Maintainer Pierrick Roger <pierrick.roger@cea.fr>

R topics documented:

`KeggCircle` *A class for representing a circle.*

Description

A class for representing a circle.

A class for representing a circle.

Details

This class represents a rectangle, used for graphical representation. It is used by KeggPathwayConn::extractPathwayMapShapes() method.

Arguments to the constructor are:

x: X coordinate of center.

y: Y coordinate of center.

r: Radius.

Super class

`biodbKegg::KeggShape` -> `KeggCircle`

Methods

Public methods:

- `KeggCircle$new()`
- `KeggCircle$getX()`
- `KeggCircle$getY()`
- `KeggCircle$getRadius()`
- `KeggCircle$clone()`

Method `new()`**:** Initialize new instance.

Usage:

```
KeggCircle$new(x, y, r, ...)
```

Arguments:

`x` Abscissa of the circle's center.

`y` Ordinate of the circle's center.

`r` Radius of the circle.

`...` Additional parameters are passed to super class' initializer.

Returns: Nothing.

Method `getX()`**:** Get the X coordinate.

Usage:

```
KeggCircle$getX()
```

Returns: The X coordinate.

Method `getY()`**:** Get the Y coordinate.

Usage:

```
KeggCircle$getY()
```

Returns: The Y coordinate.

Method `getRadius()`**:** Get the radius.

Usage:

```
KeggCircle$getRadius()
```

Returns: The radius.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

KeggShape, KeggRect.

Examples

```
# Create an instance
c1 <- KeggCircle$new(x=12, y=5, r=3)

# Since it inherits from KeggShape, a color and a label can be set
c2 <- KeggCircle$new(x=12, y=5, r=3, color='blue', label='Circle 2')

# Getting center
c1$getX()
c1$getY()

# Getting radius
c1#getRadius()

# Draw a circle on the current image

c1$draw()
```

`KeggCompoundConn` *The connector class to KEGG Compound database.*

Description

The connector class to KEGG Compound database.

The connector class to KEGG Compound database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

biodb::BiodbConnBase -> biodb::BiodbConn -> biodbKegg::KeggConn -> KeggCompoundConn

Methods

Public methods:

- KeggCompoundConn$new()
- KeggCompoundConn$wsFindExactMass()
- KeggCompoundConn$wsFindMolecularWeight()
- KeggCompoundConn$getPathwayIdsPerCompound()
- KeggCompoundConn$getModuleIdsPerCompound()
- KeggCompoundConn$getPathwayIds()
- KeggCompoundConn$addInfo()
- KeggCompoundConn$clone()

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggCompoundConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

Method `wsFindExactMass()`**:** Searches for entries by mass. You must either provide a single mass through 'mass' parameter or provide a range through 'mass.min' and 'mass.max'.

Usage:

```
KeggCompoundConn$wsFindExactMass(
  mass = NULL,
  mass.min = NULL,
  mass.max = NULL,
  ...
)
```

Arguments:

`mass` Single mass.

`mass.min` Minimal mass.

`mass.max` Maximal mass.

`...` parameters passed to KeggConn::wsFind().

`See` http //www.kegg.jp/kegg/docs/keggapi.html for details.

Returns: wsFind().

Method `wsFindMolecularWeight()`**:** Searches for entries by molecular mass. You must either provide a single mass through 'mass' parameter or provide a range through 'mass.min' and 'mass.max'. See http //www.kegg.jp/kegg/docs/keggapi.html for details.

Usage:

```
KeggCompoundConn$wsFindMolecularWeight(
  mass = NULL,
  mass.min = NULL,
  mass.max = NULL,
  ...
)
```

Arguments:

`mass` Single mass.

`mass.min` Minimal mass.

`mass.max` Maximal mass.

`...` Parameters passed to KeggConn::wsFind().

Returns: wsFind().

Method `getPathwayIdsPerCompound()`**:** Gets organism pathways for each compound. This method retrieves for each compound the KEGG pathways of the organism in which the compound is involved.

Usage:

```
KeggCompoundConn$getPathwayIdsPerCompound(id, org, limit = 3)
```

Arguments:

`id` A character vector of KEGG Compound IDs.

`org` The organism in which to search for pathways, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

`limit` The maximum number of modules IDs to retrieve for each compound. Set to 0 to disable.

Returns: A named list of KEGG pathway ID vectors, where the names of the list are the compound IDs."

Method `getModuleIdsPerCompound()`**:** Gets organism modules for each compound. This method retrieves for each compound the KEGG modules of the organism in which the compound is involved.

Usage:

```
KeggCompoundConn$getModuleIdsPerCompound(id, org, limit = 3)
```

Arguments:

`id` A character vector of KEGG Compound IDs.

`org` The organism in which to search for modules, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

`limit` The maximum number of modules IDs to retrieve for each compound. Set to 0 to disable.

Returns: A named list of KEGG module ID vectors, where the names of the list are the compound IDs."

Method `getPathwayIds()`**:** Gets organism pathways. This method retrieves KEGG pathways of the specified organism in which the compounds are involved.

Usage:

```
KeggCompoundConn$getPathwayIds(id, org)
```

Arguments:

`id` A character vector of KEGG Compound IDs.

`org` The organism in which to search for pathways, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

Returns: A vector of KEGG pathway IDs.

Method `addInfo()`**:** Add informations (as new column appended to the end) to an existing data frame containing a column of KEGG Compound IDs.

Usage:

```
KeggCompoundConn$addInfo(x, id.col, org, limit = 3, prefix = "")
```

Arguments:

`x` A data frame containing at least one column with Biodb entry IDs identified by the parameter 'id.col'.

`id.col` The name of the column containing IDs inside the input data frame.

`org` The organism in which to search for pathways, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

`limit` This is the maximum number of values obtained for each ID, for every column added, in case multiple values are obtained. Set to 0 to get all values.

`prefix` Insert a prefix at the start of name of all new columns.

Returns: A data frame containing 'x' and new columns appended with KEGG identifiers and data.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

KeggConn, KeggPathwayConn.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector to KEGG Compound
conn <- mybiodb$getFactory()$createConn('kegg.compound')

# Search for compounds by exact mass
conn$wsFindExactMass(mass=174.05, retfmt='parsed')
```

```
# Search for compounds by molecular weight
conn$wsFindMolecularWeight(mass=300, retfmt='parsed')

# Get pathway IDs related to compounds
pathway.ids=conn$getPathwayIds(c('C02648', 'C06144'), org='mmu')

# Terminate instance.
mybiodb$terminate()
```

KeggCompoundEntry *KEGG Compound entry class.*

Description

This is the entry class for the KEGG Compound database.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggCompoundEntry

Methods

Public methods:

- KeggCompoundEntry$clone()

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.compound')

# Get an entry
e <- conn$getEntry('C00133')

# Terminate instance.
mybiodb$terminate()
```

KeggConn *The connector abstract class to KEGG databases.*

Description

The connector abstract class to KEGG databases.

The connector abstract class to KEGG databases.

Details

This is the mother class of all KEGG connectors. It defines code common to all KEGG connectors.

The constructor accepts the following arguments:

db.name: The database name as defined in http://www.kegg.jp/kegg/docs/keggapi.html.

db.abbrev: The database abbreviated name, as defined in http://www.kegg.jp/kegg/docs/keggapi.html.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `KeggConn`

Methods

Public methods:

- `KeggConn$new()`
- `KeggConn$wsList()`
- `KeggConn$wsFind()`
- `KeggConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class. The parameters of this function are for the use of subclasses.

Usage:

```
KeggConn$new(
  db.name = NA_character_,
  db.abbrev = NA_character_,
  accession.prefix = NA_character_,
  ...
)
```

Arguments:

`db.name` The database name as defined in www.kegg.jp/kegg/docs/keggapi.html.

`db.abbrev` The database abbreviation as defined in www.kegg.jp/kegg/docs/keggapi.html.

`accession.prefix` The prefix used for accession identifiers.

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

Method `wsList()`**:** Gets the full list of entry IDs. See

Usage:

```
KeggConn$wsList(retfmt = c("plain", "request", "ids"))
```

Arguments:

`retfmt` Use to set the format of the returned value. 'plain' will return the raw result from the server, as a character value. 'request' will return the request as it would have been sent, as a BiodbRequest object. 'ids' will return a character vector containing entry IDs.

`http` //www.kegg.jp/kegg/docs/keggapi.html for details.

Returns: Depending on 'retfmt'.

Method `wsFind()`**:** Searches for entries. See http://www.kegg.jp/kegg/docs/keggapi.html for details.

Usage:

```
KeggConn$wsFind(
  query,
  option = c("NONE", "formula", "exact_mass", "mol_weight", "nop"),
  retfmt = c("plain", "request", "parsed", "ids", "ids.no.prefix")
)
```

Arguments:

`query` The query to send to the database web service. When searching by mass (i.e. 'option' parameter set to either 'exact_mass' or 'mol_weight'), this query field must be set to either an exact (i.e. 174.05) or a range (i.e. '250-260').

`option` Set this parameter to 'NONE' for querying on fields 'ENTRY', 'NAME', 'DESCRIPTION', 'COMPOSITION', 'DEFINITION' and 'ORTHOLOGY'. See http //www.kegg.jp/kegg/docs/keggapi.html for an exact list of fields that are searched for each database, and also for other possible values of this 'option' paramater.

`retfmt` Use to set the format of the returned value. 'plain' will return the raw result from the server, as a character value. 'request' will return the request as it would have been sent, as a BiodbRequest object. 'parsed' will return a data frame. 'ids' will return a character vector containing the IDs of the matching entries.

Returns: Depending on 'retfmt'.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector to a KEGG database
conn <- mybiodb$getFactory()$createConn('kegg.compound')
```

```
# Search for an entry
conn$wsFind('NADPH', retfmt='parsed')

# Terminate instance.
mybiodb$terminate()
```

KeggEntry *KEGG entry abstract class.*

Description

This is the abstract entry class for all KEGG entry classes.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> KeggEntry

Methods

Public methods:

- KeggEntry$clone()

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.compound')

# Get an entry
e <- conn$getEntry('C00133')

# Terminate instance.
mybiodb$terminate()
```

KeggEnzymeConn *The connector class to KEGG Enzyme database.*

Description

The connector class to KEGG Enzyme database.

The connector class to KEGG Enzyme database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `biodbKegg::KeggConn` -> `KeggEnzymeConn`

Methods

Public methods:

- `KeggEnzymeConn$new()`
- `KeggEnzymeConn$getPathwayIds()`
- `KeggEnzymeConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggEnzymeConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

Method `getPathwayIds()`**:** Gets organism pathways. This method retrieves KEGG pathways of the specified organism in which the enzymes are involved.

Usage:

```
KeggEnzymeConn$getPathwayIds(id, org)
```

Arguments:

`id` A character vector of KEGG Compound IDs.

`org` The organism in which to search for pathways, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

Returns: A vector of KEGG pathway IDs.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

KeggConn.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Get connector
conn <- mybiodb$getFactory()$createConn('kegg.enzyme')

# Get pathway IDs related to enzymes
pathway.ids=conn$getPathwayIds(c('1.2.1.3', '3.7.1.3'), org='mmu')

# Terminate instance.
mybiodb$terminate()
```

KeggEnzymeEntry *KEGG Enzyme entry class.*

Description

This is the entry class for the KEGG Enzyme class.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggEnzymeEntry

Methods

Public methods:

- KeggEnzymeEntry$clone()

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.enzyme')

# Get an entry
e <- conn$getEntry('1.1.1.54')

# Terminate instance.
mybiodb$terminate()
```

KeggGenesConn *The connector class to KEGG Pathway database.*

Description

The connector class to KEGG Pathway database.

The connector class to KEGG Pathway database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `biodbKegg::KeggConn` -> KeggGenesConn

Methods

Public methods:

- `KeggGenesConn$new()`
- `KeggGenesConn$getPathwayIdsPerGene()`
- `KeggGenesConn$getPathwayIds()`
- `KeggGenesConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggGenesConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

Method `getPathwayIdsPerGene()`**:** Gets organism pathways for each gene. This method retrieves for each gene the KEGG pathways of the organism in which the gene is involved.

Usage:

```
KeggGenesConn$getPathwayIdsPerGene(id, org, limit = 3)
```

Arguments:

`id` A character vector of KEGG Gene IDs.

`org` The organism in which to search for pathways, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

`limit` The maximum number of modules IDs to retrieve for each gene. Set to 0 to disable.

Returns: A named list of KEGG pathway ID vectors, where the names of the list are the gene IDs."

Method `getPathwayIds()`**:** Gets organism pathways. This method retrieves KEGG pathways of the specified organism in which the genes are involved.

Usage:

```
KeggGenesConn$getPathwayIds(id, org)
```

Arguments:

`id` A character vector of KEGG Genes IDs.

`org` The organism in which to search for pathways, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

Returns: A vector of KEGG pathway IDs.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

KeggConn.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.genes')

# Get an entry
e <- conn$getEntry('mmu:14635')
```

```
# Terminate instance.
mybiodb$terminate()
```

KeggGenesEntry *KEGG Genes entry class.*

Description

This is the entry class for the KEGG Genes database.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggGenesEntry

Methods

Public methods:

- KeggGenesEntry$clone()

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.genes')

# Get an entry
e <- conn$getEntry('mmu:14635')

# Terminate instance.
mybiodb$terminate()
```

KeggGlycanConn *The connector class to KEGG Glycan database.*

Description

The connector class to KEGG Glycan database.

The connector class to KEGG Glycan database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `biodbKegg::KeggConn` -> `KeggGlycanConn`

Methods

Public methods:

- `KeggGlycanConn$new()`
- `KeggGlycanConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggGlycanConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

`KeggConn`.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector to KEGG Glycan
conn <- mybiodb$getFactory()$createConn('kegg.glycan')

# Terminate instance.
mybiodb$terminate()
```

KeggGlycanEntry *KEGG Glycan entry class.*

Description

This is the entry class for the KEGG Glycan database.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggGlycanEntry

Methods

Public methods:

- KeggGlycanEntry$clone()

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.glycan')

# Get an entry
e <- conn$getEntry('G00018')

# Terminate instance.
mybiodb$terminate()
```

KeggModuleConn *The connector class to KEGG Pathway database.*

Description

The connector class to KEGG Pathway database.

The connector class to KEGG Pathway database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `biodbKegg::KeggConn` -> `KeggModuleConn`

Methods

Public methods:

- `KeggModuleConn$new()`
- `KeggModuleConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggModuleConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

`KeggConn`.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.module')

# Get an entry
e <- conn$getEntry('M00009')

# Terminate instance.
mybiodb$terminate()
```

KeggModuleEntry *KEGG Module entry class.*

Description

This is the entry class for KEGG Module database

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggModuleEntry

Methods

Public methods:

- KeggModuleEntry$clone()

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.module')

# Get an entry
e <- conn$getEntry('M00009')

# Terminate instance.
mybiodb$terminate()
```

KeggOrthologyConn *The connector class to KEGG Orthology database.*

Description

The connector class to KEGG Orthology database.

The connector class to KEGG Orthology database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `biodbKegg::KeggConn` -> `KeggOrthologyConn`

Methods

Public methods:

- `KeggOrthologyConn$new()`
- `KeggOrthologyConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggOrthologyConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

`KeggConn`.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.orthology')

# Get an entry
e <- conn$getEntry('K12668')

# Terminate instance.
mybiodb$terminate()
```

KeggOrthologyEntry *KEGG Orthology entry class.*

Description

This is the class entry for KEGG Orthology database.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggOrthologyEntry

Methods

Public methods:

- KeggOrthologyEntry$clone()

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.orthology')

# Get an entry
e <- conn$getEntry('K12668')

# Terminate instance.
mybiodb$terminate()
```

KeggPathwayConn *The connector class to KEGG Pathway database.*

Description

The connector class to KEGG Pathway database.

The connector class to KEGG Pathway database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `biodbKegg::KeggConn` -> `KeggPathwayConn`

Methods

Public methods:

- `KeggPathwayConn$new()`
- `KeggPathwayConn$getReactions()`
- `KeggPathwayConn$convertToOrgPathways()`
- `KeggPathwayConn$buildPathwayGraph()`
- `KeggPathwayConn$getPathwayIgraph()`
- `KeggPathwayConn$getDecoratedGraphPicture()`
- `KeggPathwayConn$extractPathwayMapShapes()`
- `KeggPathwayConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggPathwayConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

Method `getReactions()`**:** Retrieves all reactions part of a KEGG pathway. Connects to KEGG databases, and walk through all pathways submitted, and their modules, to find all reactions they are composed of.

Usage:

```
KeggPathwayConn$getReactions(id, drop = TRUE)
```

Arguments:

`id` A character vector of entry IDs.

`drop` If set to TRUE, returns a single KEGG reaction object instead of a list, if the list contains only one element.

Returns: A list of KEGG reaction objects.

Method `convertToOrgPathways()`**:** Takes a list of pathways IDs and converts them to the specified organism, filtering out the ones that do not exist in KEGG.

Usage:

```
KeggPathwayConn$convertToOrgPathways(id, org)
```

Arguments:

`id` A character vector of entry IDs.

`org` The organism in which to search for pathways, as a KEGG organism code (3-4 letters code, like 'hsa', 'mmu', ...). See https //www.genome.jp/kegg/catalog/org_list.html for a complete list of KEGG organism codes.

Returns: A character vector, the same length as 'id', containing the converted IDs.

Method `buildPathwayGraph()`**:** Builds a pathway graph in the form of two tables of vertices and edges, using KEGG database.

Usage:

```
KeggPathwayConn$buildPathwayGraph(id, directed = FALSE, drop = TRUE)
```

Arguments:

`id` A character vector of KEGG pathway entry IDs.

`directed` If set to TRUE, use available direction information to create directed edges, duplicating if necessary the vertices.

`drop` If set to TRUE and the output list contains only one element, then the returned value is a single list of two data frames.

Returns: A named list whose names are the pathway IDs, and values are lists containing two data frames named vertices and edges.

Method `getPathwayIgraph()`**:** Builds a pathway graph, as an igraph object, using KEGG database.

Usage:

```
KeggPathwayConn$getPathwayIgraph(id, directed = FALSE, drop = TRUE)
```

Arguments:

`id` A character vector of KEGG pathway entry IDs.

`directed` If set to TRUE, use available direction information to create directed edges, duplicating if necessary the vertices.

`drop` If set to TRUE and the output list contains only one element, then the returned value is a single igraph object.

Returns: A list of igraph objects, or an empty list if the igraph library is not available.

Method `getDecoratedGraphPicture()`**:** Create a pathway graph picture, with some of its elements colorized.

Usage:

```
KeggPathwayConn$getDecoratedGraphPicture(id, color2ids)
```

Arguments:

`id` A KEGG pathway ID.

`color2ids` A named list defining colors for entry IDs that are present on the graph. The names of the list are standard color names. The values are character vector of entry IDs.

Returns: An image object or NULL if the package magick is not available.

Method `extractPathwayMapShapes()`**:** Extracts shapes from a pathway map image.

Usage:

```
KeggPathwayConn$extractPathwayMapShapes(id, color2ids)
```

Arguments:

`id` A KEGG pathway ID.

`color2ids` A named list defining colors for entry IDs that are present on the graph. The names of the list are standard color names. The values are character vector of entry IDs.

Returns: A list of BiodbShape objects.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

KeggConn.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Get connector
conn=mybiodb$getFactory()$createConn('kegg.pathway')

# Retrieve all reactions related to a mouse pathway:
reactions=conn$getReactions('mmu00260')

# Get a pathway graph
graph=conn$buildPathwayGraph('mmu00260')

# Terminate instance.
mybiodb$terminate()
```

KeggPathwayEntry *KEGG Pathway entry class.*

Description

This is the class entry for KEGG Pathway database.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggPathwayEntry

Methods

Public methods:

- KeggPathwayEntry$clone()

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.pathway')

# Get an entry
e <- conn$getEntry('map00053')

# Terminate instance.
mybiodb$terminate()
```

KeggReactionConn *The connector class to KEGG Reaction database.*

Description

The connector class to KEGG Reaction database.

The connector class to KEGG Reaction database.

Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory `BiodbFactory`. Only specific methods are described here. See super classes for the description of inherited methods.

Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `biodbKegg::KeggConn` -> `KeggReactionConn`

Methods

Public methods:

- `KeggReactionConn$new()`
- `KeggReactionConn$clone()`

Method `new()`**:** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

Usage:

```
KeggReactionConn$new(...)
```

Arguments:

`...` All parameters are passed to the super class initializer.

Returns: Nothing.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

`KeggConn`.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.reaction')

# Get an entry
e <- conn$getEntry('R00105')

# Terminate instance.
mybiodb$terminate()
```

KeggReactionEntry *KEGG Reaction entry class.*

Description

This is the entry class for KEGG Reation database.

Super classes

biodb::BiodbEntry -> biodb::BiodbTxtEntry -> biodbKegg::KeggEntry -> KeggReactionEntry

Methods

Public methods:

- KeggReactionEntry$clone()

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('kegg.reaction')

# Get an entry
e <- conn$getEntry('R00105')

# Terminate instance.
mybiodb$terminate()
```

KeggRect *A class for representing a rectangle.*

Description

A class for representing a rectangle.

A class for representing a rectangle.

Details

This class represents a rectangle, used for graphical representation.

Arguments to the constructor are:

left: Coordinate of left border.

right: Coordinate of right border.

top: Coordinate of top border.

bottom: Coordinate of bottom border.

Super class

biodbKegg::KeggShape -> KeggRect

Methods

Public methods:

- KeggRect$new()
- KeggRect$clone()

Method `new()`**:** Initialize new instance.

Usage:

```
KeggRect$new(left, top, bottom, right, ...)
```

Arguments:

`left` Coordinate of rectangle's left side.

`top` Coordinate of rectangle's top side.

`bottom` Coordinate of rectangle's bottom side.

`right` Coordinate of rectangle's right side.

`...` Additional parameters are passed to super class' initializer.

Returns: Nothing.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

KeggShape, KeggCircle.

Examples

```
# Create a rectangle instance
r <- KeggRect$new(left=10, top=10, bottom=20, right=30, color='yellow')

# Draw a rectangle on current image

r$draw()
```

KeggShape *A class for representing a shape.*

Description

A class for representing a shape.

A class for representing a shape.

Details

This abstract class represents a shape, used for graphical representation.

Arguments to the constructor are:

label: A text label to associate with the shape.

color: A color, as a character string.

Methods

Public methods:

- KeggShape$new()
- KeggShape$equals()
- KeggShape$getLabel()
- KeggShape$getColor()
- KeggShape$getRgbColor()
- KeggShape$draw()
- KeggShape$clone()

Method `new()`**:** Initialize new instance.

Usage:

```
KeggShape$new(label = NA_character_, color = NA_character_)
```

Arguments:

`label` The text label to display.
`color` The color to use.

Returns: Nothing.

Method `equals()`**:** Test if this shape is the same as another.

Usage:

```
KeggShape$equals(other)
```

Arguments:

`other` The other shape to compare with.

Returns: TRUE or FALSE.

Method `getLabel()`**:** Gets the label associated with this shape.

Usage:

```
KeggShape$getLabel()
```

Returns: The label.

Method `getColor()`**:** Gets the color associated with this shape.

Usage:

```
KeggShape$getColor()
```

Returns: The color name as a string.

Method `getRgbColor()`**:** Gets the RGB color associated with this shape.

Usage:

```
KeggShape$getRgbColor(alpha = 255)
```

Arguments:

`alpha` The value to use for the alpha channel when building the RGB color object.

Returns: The color as an RGB color object.

Method `draw()`**:** Draw the shape on the current image.

Usage:

```
KeggShape$draw()
```

Returns: None.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

`KeggRect`, `KeggCircle`.

Examples

```
# Create a circle instance
c <- KeggCircle$new(x=12, y=5, r=3, label='MyCircle')

# Create a rectangle instance
r <- KeggRect$new(left=10, top=10, bottom=20, right=30, color='yellow')
```

Index