hom.Dm.inp.db

February 3, 2010

`hom.Dm.inp.db` *Bioconductor annotation data package*

Description

Welcome to the hom.Dm.inp.db annotation Package. The purpose of this package is to provide detailed information about the hom.Dm.inp platform. This package is updated biannually.

You can learn what objects this package supports with the following command:

```
ls("package:hom.Dm.inp.db")
```

Each of these objects has their own manual page detailing where relevant data was obtained along with some examples of how to use it.

Examples

```
ls("package:hom.Dm.inp.db")
```

`hom.Dm.inp_dbconn` *Collect information about the package annotation DB*

Description

Some convenience functions for getting a connection object to (or collecting information about) the package annotation DB.

Usage

```
hom.Dm.inp_dbconn()
hom.Dm.inp_dbfile()
hom.Dm.inp_dbschema(file="", show.indices=FALSE)
hom.Dm.inp_dbInfo()
```

Arguments

`file`	A connection, or a character string naming the file to print to (see the `file` argument of the `cat` function for the details).
`show.indices`	The CREATE INDEX statements are not shown by default. Use `show.indices=TRUE` to get them.

Details

hom.Dm.inp_dbconn returns a connection object to the package annotation DB. IMPORTANT: Don't call dbDisconnect on the connection object returned by hom.Dm.inp_dbconn or you will break all the AnnDbObj objects defined in this package!

hom.Dm.inp_dbfile returns the path (character string) to the package annotation DB (this is an SQLite file).

hom.Dm.inp_dbschema prints the schema definition of the package annotation DB.

hom.Dm.inp_dbInfo prints other information about the package annotation DB.

Value

hom.Dm.inp_dbconn: a DBIConnection object representing an open connection to the package annotation DB.

hom.Dm.inp_dbfile: a character string with the path to the package annotation DB.

hom.Dm.inp_dbschema: none (invisible NULL).

hom.Dm.inp_dbInfo: none (invisible NULL).

See Also

dbGetQuery, dbConnect, dbconn, dbfile, dbschema, dbInfo

Examples

```
## Count the number of rows in the "metadata" table:
dbGetQuery(hom.Dm.inp_dbconn(), "SELECT COUNT(*) FROM metadata")

## The connection object returned by hom.Dm.inp_dbconn() was created with:
dbConnect(SQLite(), dbname=hom.Dm.inp_dbfile(), cache_size=64000, synchronous=0)

hom.Dm.inp_dbschema()

hom.Dm.inp_dbInfo()
```

---

hom.Dm.inpMAPCOUNTS

Number of mapped keys for the maps in package hom.Dm.inp.db

Description

hom.Dm.inpMAPCOUNTS provides the "map count" (i.e. the count of mapped keys) for each map in package hom.Dm.inp.db.

Details

This "map count" information is precalculated and stored in the package annotation DB. This allows some quality control and is used by the checkMAPCOUNTS function defined in AnnotationDbi to compare and validate different methods (like count.mappedkeys(x) or sum(!is.na(as.list(x)))) for getting the "map count" of a given map.

See Also

mappedkeys, count.mappedkeys, checkMAPCOUNTS

Examples

```
hom.Dm.inpMAPCOUNTS
mapnames <- names(hom.Dm.inpMAPCOUNTS)
hom.Dm.inpMAPCOUNTS[mapnames[1]]
x <- get(mapnames[1])
sum(!is.na(as.list(x)))
count.mappedkeys(x)    # much faster!

## Check the "map count" of all the maps in package hom.Dm.inp.db
checkMAPCOUNTS("hom.Dm.inp.db")
```

---

hom.Dm.inpORGANISM *The Organism for hom.Dm.inp*

Description

hom.Dm.inpORGANISM is an R object that contains a single item: a character string that names the organism for which hom.Dm.inp was built.

Details

Although the package name is suggestive of the organism for which it was built, hom.Dm.inpORGANISM provides a simple way to programmatically extract the organism name.

Examples

```
hom.Dm.inpORGANISM
```

---

hom.Dm.inpHOMSA *Map between IDs for genes in one organism to their predicted paralogs in another*

Description

A map of this type is an R object that provides mappings between identifiers for genes in the package organism and their predicted paralogs in the map that the organism is named after. So for example, if the inparanoid package is the human package, then the hom.Dm.inpRATNO map would provide mappings between human and rat.

Details

Mappings between gene identifiers and their paralogs as predicted by the Inparanoid algorithm. The map filters out paralogs that have an Inparanoid score less than 100

Mappings are normally given from the ID of the organism in the package to the IDs of the organism listed in the map name.

Reversal can be made of ANY map by using the function revmap (see examples below).

Names for these maps are done in the "INPARANOID style" which means that they are normally the 1st three letters of the genus followed by the 1st two letters of the species. For example: "Mus musculus" becomes "MUSMU", "Homo sapiens" becomes "HOMSA", "Monodelphis domestica" becomes "MONDO" etc. This means that for most of these organisms it will be possible to easily guess the abbreviations used. An exception may occur in the future if a new model organism has a very similar genus and species name to an existing one.

References

http://inparanoid.sbc.su.se/download/current/sqltables

Examples

```
x <- hom.Dm.inpAPIME
# Get honeybee IDs that are paralogous to the pkg IDs
mapped_IDs <- mappedkeys(x)
# Convert to a list
xx <- as.list(x[mapped_IDs])
if(length(xx) > 0) {
  # Get the paralogs for the first five genes
  xx[1:5]
  # Get the first one
  xx[[1]]
}

#Now for the reverse map (honeybee IDs back to pkg paralog)
x <- revmap(hom.Dm.inpAPIME)
mapped_IDs <- mappedkeys(x)
# Convert to a list
xx <- as.list(x[mapped_IDs])
if(length(xx) > 0) {
  # Get the paralogs for the first five IDs
  xx[1:5]
  # Get the first one
  xx[[1]]
}

## Not run:
    #For the most common organisms, we try to ensure that you can
    #map back to an Entrez Gene ID by providing you with necessary
    #maps in the related organism based annotation packages.  The
    #following example shows how to get from an Entrez Gene ID for
    #Human to Entrez Gene IDs for Mouse even though inparanoid does
    #not map to Entrez Gene IDs for either of these species.

    #You will have to include the appropriate packages for
    #humans:
    library("org.Hs.eg.db")
```

```
#and for mouse:
library("org.Mm.eg.db")
#And of course you will need the inparanoid package:
library("hom.Hs.inp.db")

#Start with some Human Entrez Gene IDs
humanEGIds <- c("4488","4487")

#Inparanoid uses ensembl protein IDs so start with
#those. Notice that there will be many protein IDs returned for
#a typical gene since there are many possible translations.
humanProtIds <- mget(humanEGIds,org.Hs.egENSEMBLPROT)

#Map the IDs that we can from inparanoid. Notice that by design,
#inparanoid only represents each gene product with a single
#translation product. Therefore your list could slim down a lot
#during this step.  Also, if the thing you are trying to match
#up at this step has less than 100% seed status, you will not
#find it in this step.
rawMouseProtIds <- mget(unlist(humanProtIds),hom.Hs.inpMUSMU,ifnotfound=NA)
#This also means that we need to clean up the NAs from our result
mouseProtIds <- rawMouseProtIds[!is.na(rawMouseProtIds)]

#Then use the mouse organism based packages to convert these IDs
#back to an Entrez Gene ID again (this time for mouse).
mouseEGIds <- mget(unlist(mouseProtIds),org.Mm.egMGI2EG,ifnotfound=NA)

#Now go ahead and have a look at the output
mouseEGIds

## End(Not run)
```

Index