Package 'biocViews'

October 8, 2015

Version 1.36.2

Date 2014-02-11

Title Categorized views of R package repositories

Author VJ Carey <stvjc@channing.harvard.edu>,
BJ Harshfield <rebjh@channing.harvard.edu>,
S Falcon <sfalcon@fhcrc.org> ,
Sonali Arora <sarora@fredhutch.org>

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

Depends R (>= 2.4.0)

Imports Biobase, graph (>= 1.9.26), methods, RBGL (>= 1.13.5), tools, utils, XML, RCurl, RUnit, knitr

Suggests Biobase, BiocGenerics, RUnit

Description structures for vocabularies and narratives of views

License Artistic-2.0

URL http://www.bioconductor.org/packages/release/BiocViews.html

Collate AllClasses.R AllGenerics.R as-methods.R htmlDoc-methods.R htmlFilename-methods.R htmlValue-methods.R show-methods.R getPackNames.R packageDetails.R pump.R repository.R showvoc.R getPackageNEWS.R validation_tests.R recommendBiocViews.R

biocViews Infrastructure

NeedsCompilation no

R topics documented:

`biocViews-package` *Categorized views of R package repositories*

Description

Structures for vocabularies and narratives of views. This can be used to create HTML views of the package structure in a Bioconductor repository.

Details

Package:	biocViews
Version:	1.11.4
Depends:	R (>= 2.4.0), methods, utils
Imports:	tools, Biobase, graph (>= 1.9.26), RBGL (>= 1.13.5), XML
Suggests:	Biobase
License:	Artistic-2.0
URL:	http://www.bioconductor.org/packages/release/BiocViews.html
biocViews:	Infrastructure

Index:

The terms of the vocabulary are stored in a DAG, which can be loaded as the serialized data object biocViewsVocab. For listing of available terms use function getSubTerms.

Further information is available in the following two vignettes:

HOWTO-BCV	Basic package usage
createReposHtml	Further information for repository admins

Author(s)

VJ Carey <stvjc@channing.harvard.edu>, BJ Harshfield <rebjh@channing.harvard.edu>, S Falcon <sfalcon@fhcrc.org>

Maintainer: Biocore Team c/o BioC user list <bioconductor@stat.math.ethz.ch>

Examples

```
data(biocViewsVocab)
getSubTerms(biocViewsVocab, "Technology")
```

BiocView-class *Class "BiocView"*

Description

Representation of of Bioconductor "view".

Objects from the Class

Objects can be created by calls of the form `new("BiocView", ...)`.

Slots

`name`: Object of class `"character"` giving the name of the view.

`subViews`: Object of class `"character"` giving the names of the subviews of this view.

`parentViews`: Object of class `"character"` giving the names of the views that are this view's parents.

`Title`: Object of class `"character"` giving longer description of view?

`reposRoot`: Object of class `"character"` URL for repository

`homeUrl`: Object of class `"character"` ?

`htmlDir`: Object of class `"character"` ?

`packageList`: Object of class `"list"` consisting of `PackageDetail-class` objects

Extends

Class `"RepositoryDetail"`, directly. Class `"Htmlized"`, directly.

Methods

coerce `signature(from = "BiocView", to = "rdPackageTable")`: ...

htmlDoc `signature(object = "BiocView")`: ...

htmlFilename `signature(object = "BiocView")`: ...

htmlValue `signature(object = "BiocView")`: ...

show `signature(object = "BiocView")`: ...

Author(s)

Seth Falcon

`biocViewsVocab` *Bioconductor Task Views Vocabulary Data*

Description

A `graphNEL-class` instance representing the Bioconductor Task Views as a directed graph.

Usage

```
data(biocViewsVocab)
```

Format

The format is: graphNEL instance

Details

The source for the vocabulary data is in the dot directory of the package in file biocViewsVocab.dot. This is transformed to GXL using the dot2gxl command line utility from the graphviz package. Then the `fromGXL` function from the `graph` package is used to convert to `graphNEL-class`.

Examples

```
data(biocViewsVocab)
biocViewsVocab
## If you have Rgraphviz available, you can
## plot the vocabulary with plot(biocViewsVocab)
```

`extractManuals` *Extract Rd man pages and build pdf reference manuals from local package repository*

Description

This function extracts Rd man pages and builds pdf reference manuals from the `man` subdirectory of R source packages archives (`.tar.gz`) found in a local package repository.

All Rd files found in `man` will be extracted and used during the pdf construction process. Only source package archives will be processed. The constructed pdf files will be extracted under `destDir` and will be found in `PKGNAME/man/*.pdf`.

Prior to extraction, all Rd and pdf files in `destDir/PKGNAME/man` will be removed.

Usage

```
extractManuals(reposRoot, srcContrib, destDir)
```

Arguments

reposRoot	character vector giving the path to the root of the local CRAN-style package repository
srcContrib	character vector giving the relative path from the reposRoot to the source packages. In a standard CRAN-style repository, this will be src/contrib.
destDir	character vector specifying the directory in which the extracted files will be written. If missing, files will be written to <reposRoot>/manuals.

Author(s)

Patrick Aboyoun

---

extractNEWS *Extract NEWS files from source package tarballs*

Description

Extracts NEWS files from source tarballs of packages.

Usage

```
extractNEWS(reposRoot, srcContrib, destDir)
```

Arguments

reposRoot	Top level path for CRAN-style repos
srcContrib	Location of source packages
destDir	where to extract

extractTopLevelFiles *Extract files from the top level of source package tarballs*

Description

Extracts files from source tarballs of packages.

Usage

```
extractTopLevelFiles(reposRoot, srcContrib, destDir, fileName)
```

Arguments

reposRoot	Top level path for CRAN-style repos
srcContrib	Location of source packages
destDir	where to extract
fileName	name of file to extract

extractVignettes *Extract pdf vignettes from local package repository*

Description

These functions extract pdf or HTML files from the inst/doc subdirectory of R source packages archives (.tar.gz) found in a local package repository.

All pdf files found in inst/doc will be extracted. With extractHTMLDocuments, all HTML files except index.html will be extracted. Only source package archives will be processed. The extracted pdf or HTML files will be extracted under destDir and will be found in PKGNAME/inst/doc/.

Prior to extraction, all pdf files in destDir/PKGNAME/inst/doc will be removed.

Usage

```
extractVignettes(reposRoot, srcContrib, destDir)
extractHTMLDocuments(reposRoot, srcContrib, destDir)
```

Arguments

reposRoot	character vector giving the path to the root of the local CRAN-style package repository
srcContrib	character vector giving the relative path from the reposRoot to the source packages. In a standard CRAN-style repository, this will be src/contrib.
destDir	character vector specifying the directory in which the extracted files will be written. If missing, files will be written to <reposRoot>/vignettes.

Author(s)

Seth Falcon

---

genReposControlFiles *Generate CRAN-style repository control files*

Description

This function generates control files for CRAN-style repositories. For each path specified in `contribPaths` a `PACKAGES` file is written. In addition, two top-level control files are created:

`REPOSITORY` contains information about the specified contrib paths.

`VIEWS` contains metadata for all packages in the repository including the paths to any extracted vignettes, if found. This file is useful for generating HTML views of the repository.

Usage

```
genReposControlFiles(reposRoot, contribPaths)
```

Arguments

`reposRoot`	character vector containing the path to the CRAN-style repository root directory.
`contribPaths`	A named character vector. Valid names are `source`, `win.binary`, `win64.binary`, `mac.binary`, and `mac.binary.leopard`. Values indicate the paths to the package archives relative to the `reposRoot`.

Author(s)

Seth Falcon

See Also

`write_PACKAGES`, `extractVignettes`, `write_REPOSITORY`, `write_VIEWS`

---

getBiocSubViews *Build a list of BiocView objects from a package repository*

Description

This function returns a list of `BiocView-class` objects corresponding to the subgraph of the views DAG induced by `topTerm`. In short, this does the same thing as `getBiocViews`, but limits the vocabulary to `topTerm` and all of its decendents.

Usage

```
getBiocSubViews(reposUrl, vocab, topTerm, local = FALSE, htmlDir = "")
```

Arguments

reposUrl	URL for a CRAN-style repository that hosts a VIEWS file at the top-level.
vocab	A graph-class object representing the ontologyof views. This graph should be a directed acyclic graph (DAG).
topTerm	A string giving the name of the subview DAG. This view and all of its decendents will be included in the result.
local	logical indicating whether to assume a local package repository. The default is FALSE in which case absolute links to package detail pages are created.
htmlDir	if the local argument is TRUE, this will be used as the relative path for package HTML files.

Details

The root of the vocabulary DAG is implicitly included in the view creation process order to build views with a link back to the top. It is removed from the return list.

This function is tailored to generation of Bioconductor Task Views. With the current vocabulary, it probably only makes sense to call it with topView set to one of "Software", "AnnotationData", or "ExperimentData". This is a hack to allow the biocViews code to manage HTML views across more than one repository.

Value

A list of BiocView-class objects. The names of the list give the name of the corresponding view.

Author(s)

Seth Falcon

See Also

write_VIEWS, writeBiocViews

Examples

```
data(biocViewsVocab)
reposPath <- system.file("doc", package="biocViews")
reposUrl <- paste("file://", reposPath, sep="")
biocViews <- getBiocSubViews(reposUrl, biocViewsVocab, "Software")
print(biocViews[1:2])
```

getBiocViews *Build a list of BiocView objects from a package repository*

Description

Given the URL to a CRAN-style package repository containing a VIEWS file at the top-level and a graph-class object representing a DAG of views, this function returns a list of BiocView-class objects.

Usage

```
getBiocViews(reposUrl, vocab, defaultView, local = FALSE, htmlDir = "")
```

Arguments

reposUrl	URL for a CRAN-style repository that hosts a VIEWS file at the top-level.
vocab	A graph-class object representing the ontology of views. This graph should be a directed acyclic graph (DAG).
defaultView	A string giving the term to use for packages that do not list a term of their own via the biocViews field in the 'DESCRIPTION' file.
local	logical indicating whether to assume a local package repository. The default is FALSE in which case absolute links to package detail pages are created.
htmlDir	if the local argument is TRUE, this will be used as the relative path for package HTML files.

Value

A list of BiocView-class objects. The names of the list give the name of the corresponding view.

Author(s)

Seth Falcon

See Also

write_VIEWS, writeBiocViews

Examples

```
data(biocViewsVocab)
reposPath <- system.file("doc", package="biocViews")
reposUrl <- paste("file://", reposPath, sep="")
biocViews <- getBiocViews(reposUrl, biocViewsVocab, "NoViewProvided")
print(biocViews[1:2])
```

getCurrentbiocViews *Get a list of biocViews for each branch*

Description

This function looks returns a list containing all the biocViews that are present on the Bioconductor website.

Usage

```
getCurrentbiocViews()
```

Details

It parses the dot file present inside the biocViews package.

Value

It returns a named list with 3 components.

`Software`	biocViews from the software branch
`ExperimentData`	biocViews from the ExperimentData branch
`AnnotationData`	biocViews from the AnnotationData branch

Author(s)

Sonali Arora

Examples

```
ans <- getCurrentbiocViews()
## only the first 6 from each branch are shown here.
lapply(ans, head)
```

getPackageNEWS *Retrieve and print package NEWS*

Description

These functions visit two repository PACKAGE files, identifying packages that are present in the 'current' repository and have NEWS since the base version of the same package in the 'previous' repository. All NEWS is reported for packages only in the current repository.

Usage

```
getPackageNEWS(prevRepos="http://www.bioconductor.org/packages/2.10/bioc",
               currRepos="http://www.bioconductor.org/packages/2.11/bioc",
               srcDir)
printNEWS(dbs, destfile, overwrite = FALSE, width = 68,
          output=c("md", "text"), ...)
```

Arguments

prevRepos	character(1) repository from which NEWS starts.
currRepos	character(1) repository of current packages.
srcDir	character(1) directory containing the source all current pacakges
dbs	A list of news_db elements, as returned by getPackageNEWS.
destfile	character(1) file path to the location where NEWS will be printed.
overwrite	logical(1) indicating whether destfile can be over-written, if it exists.
width	numeric(1) number of characters news items are to be wrapped to, excluding indent.
output	character(1) output to text or markdown format.
...	additional arguments, unused.

Value

A list of news_db files, as returned by utils::news, for each package for which relevant NEWS is available.

Author(s)

Martin Morgan mtmorgan@fhcrc.org

---

getPacksAndViews *Parse VIEWS file for views and packages*

Description

Given a repository URL, download and parse the VIEWS file.

Usage

```
getPacksAndViews(reposURL, vocab, defaultView, local=FALSE)
```

Arguments

reposURL	character vector giving the URL of a CRAN-style repository containing a VIEWS file at the top-level.
vocab	A graph-class object representing the ontologyof views. This graph should be a directed acyclic graph (DAG).
defaultView	A string giving the term to use for packages that do not list a term of their own via the biocViews field in the 'DESCRIPTION' file.
local	logical indicating whether certain links should be absolute (using reposURL) or relative.

Value

A list with named elements:

views: Vector of view memberships. Names are package names.

pkgList: A list of PackageDetail-class objects.

Author(s)

Seth Falcon

---

getSubTerms *Retrieve a term and its children from a vocab DAG*

Description

Given a Directed Acyclic Graph (DAG) represented as a graphNEL instance, return a character vector consisting of the specified term and all of its descendants. That is, give the list of terms for which a path exists starting at term.

Usage

```
getSubTerms(dag, term)
```

Arguments

dag	A graphNEL representing a DAG
term	A string giving a term in the vocabulary (a node in dag)

Value

A character vector of term names.

Author(s)

S. Falcon

Examples

```
data(biocViewsVocab)
getSubTerms(biocViewsVocab, "Software")
```

`htmlDoc` *Create a complete HTML document representation of an object*

Description

This generic function should return an `XMLNode` instance representing the specified object in HTML as a complete HTML document.

Usage

```
htmlDoc(object, ...)
```

Arguments

`object`	An object
`...`	Not currently used.

Value

An instance of `XMLNode` from the `XML` package.

Author(s)

Seth Falcon

See Also

`htmlValue`, `htmlFilename`

`htmlFilename` *Return a filename for an object's HTML representation*

Description

This function returns a string containing an appropriate filename for storing the object's HTML representation.

Usage

```
htmlFilename(object, ...)
```

Arguments

`object`	An object.
`...`	Not currently used

Value

A character vector of length one containing the filename.

Author(s)

Seth Falcon

See Also

`htmlValue`, `htmlDoc`

`Htmlized-class` *Class "Htmlized"*

Description

A virtual class for HTML serialization method dispatch.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

htmlDoc `signature(object = "Htmlized")`: Return the html-ized representation of `object` as a complete HTML document.

Author(s)

Seth Falcon

htmlValue *HTML Representation of an Object*

Description

This generic function should return an XMLNode instance representing the specified object in HTML

Usage

```
htmlValue(object)
```

Arguments

object	An object

Value

An instance of XMLNode from the XML package.

Author(s)

Seth Falcon

See Also

htmlDoc, htmlFilename

PackageDetail-class *Class "PackageDetail"*

Description

Representation of R package metadata. Most slots correspond to fields in a package's DESCRIPTION file.

Objects from the Class

Objects can be created by calls of the form new("PackageDetail", ...).

Slots

`Package`: Object of class `"character"` see DESCRIPTION

`Version`: Object of class `"character"` see DESCRIPTION

`Title`: Object of class `"character"` see DESCRIPTION

`Description`: Object of class `"character"` see DESCRIPTION

`Author`: Object of class `"character"` see DESCRIPTION

`Maintainer`: Object of class `"character"` see DESCRIPTION

`Depends`: Object of class `"character"` see DESCRIPTION

`Imports`: Object of class `"character"` see DESCRIPTION

`Suggests`: Object of class `"character"` see DESCRIPTION

`SystemRequirements`: Object of class `"character"` see DESCRIPTION

`License`: Object of class `"character"` see DESCRIPTION

`URL`: Object of class `"character"` see DESCRIPTION

`biocViews`: Object of class `"character"` see DESCRIPTION

`vignettes`: Object of class `"character"` giving paths to vignette pdf files in the repository

`vignetteScripts`: Object of class `"character"` giving paths to vignette Stangled R files in the repository

`vignetteTitles`: Object of class `"character"` giving the titles of the vignette files in the repository

`source.ver`: Object of class `"character"` version string for the source package

`win.binary.ver`: Object of class `"character"` version string for the 32-bit Windows binary package

`win64.binary.ver`: Object of class `"character"` version string for the 64-bit Windows binary package

`mac.binary.leopard.ver`: Object of class `"character"` version string for the OS X Leopard binary package

`downloadStatsUrl`: Object of class `"character"` An optional URL for the download history statistics.

`manuals`: Object of class `"character"` giving paths to reference manual pdf files in the repository

`dependsOnMe`: Object of class `"character"` giving packages found in the repository that depend on this package

`importsMe`: Object of class `"character"` giving packages found in the repository that imports this package

`suggestsMe`: Object of class `"character"` giving packages found in the repository that suggest this package

`functionIndex`: Object of class `"character"` Not used. Intended to hold function index data.

`reposFullUrl`: Object of class `"character"` The URL for the full URL of the root of the repository.

`reposRoot`: Object of class `"character"` The URL for the root of the repository.

`viewRoot`: Object of class `"character"` The URL for the view of the repository.

`devHistoryUrl`: Object of class `"character"` The URL for the development changelog.

Extends

Class "Htmlized", directly.

Methods

htmlDoc signature(object = "PackageDetail"): Return an XMLNode instance containg a complete HTML document representation of the package.

htmlFilename signature(object = "PackageDetail"): Return a filename appropriate for the HTML document representation.

htmlValue signature(object = "PackageDetail"): Return XMLNode instance containing an HTML representation of the package.

Details

pdAuthorMaintainerInfo-class pdVignetteInfo-class pdDownloadInfo-class pdDetailsInfo-class pdDescriptionInfo-class pdVigsAndDownloads-class

Dummy classes for HTML generation. Each dummy class is a simple extension (it does not add any slots). The purpose of each dummy class is to allow for method dispatch to generate HTML via the htmlValue method.

You can convert convert a PackageDetail instance to one of the dummy classes like this: descInfo <- as(pdObj, "pdDe

Author(s)

Seth Falcon

Examples

```
pd <- new("PackageDetail",
          Package="MyFancyPackage",
          Version="1.2.3",
          Title="A Fancy Package",
          Description="This package does fancy things",
          Author="A. Coder",
          Maintainer="A. Coder <acoder@foo.bar.net>",
          Depends="methods",
          Imports="ASimplePackage",
          Suggests="MyDataPackage",
          biocViews="Infrastructure",
        vignettes="vignettes/MyFancyPackage/inst/doc/MFP1.pdf,\nvignettes/MyFancyPackage/inst/doc/MFP2.pdf",
        vignetteScripts="vignettes/MyFancyPackage/inst/doc/MFP1.R\nvignettes/MyFancyPackage/inst/doc/MFP2.R",
          vignetteTitles="MFP1 Document,\nMFP2 Document",
          source.ver="src/contrib/MyFancyPackage_1.2.3.tar.gz",
          win.binary.ver="bin/windows/contrib/2.6/MyFancyPackage_1.2.2.zip",
          win64.binary.ver="bin/windows64/contrib/2.6/MyFancyPackage_1.2.2.zip",
        mac.binary.leopard.ver="bin/macosx/leopard/contrib/2.6/MyFancyPackage_1.2.3.tgz",
          dependsOnMe=c("PackageThatExposesMe"),
          importsMe=c("AnEvenFancierPackage","AMuchFancierPackage"),
          suggestsMe="PackageThatUsesMeInVignette",
          reposRoot="http://foo.bar.org")
```

```
html <- htmlValue(pd)
pd
```

---

recommendBiocViews *Recommend biocViews for an existing Package.*

Description

Packages being added to the Bioconductor Project require biocViews in their DESCRIPTION file.(Note that the field name "biocViews" is case-sensitive and must begin with a lower-case 'b'.)biocViews are "keywords" which are used to describe a given package. They are broadly divided into three categories, representing the type of packages present in the Bioconductor Project - Software, Annotation Data and Experiment Data.

Usage

```
recommendBiocViews(pkgdir, branch)
```

Arguments

pkgdir	The path of the package Directory.
branch	The branch which your package will belong to. It can be either 'Software', 'AnnotationData' or 'ExperimentData'.

Details

This function parses the package directory provided by the user to recommend biocViews to the user. The output is a suggested list - the user of this function is expected to go through this list and find which biocViews best describe his or her package. It uses the following strategies.

- It parses the "Description", "Title", "Package" of the DESCRIPTION page to find biocViews.
- It looks up the biocViews of the packages in the "Depends" field of the given package to recommend biocViews
- It parses the text from the man pages and the vignettes to suggest biocViews.

Please note the following:

- Do not make up your own biocViews.
- Double check the spelling and case of the biocViews added.
- Please add biocViews only from the appropriate branch. eg: Software packages should have only Software biocViews.

Value

A list is returned with 3 characters - current , recommended and remove.

- "current" contains the biocViews from the package's DESCRIPTION file.
- "recommended" are the recommended biocViews - This is a suggested list which the user can add in addition to "current" biocViews - the user is expected to go through this list and find which biocViews best describe their package.
- "remove" are those biocViews which are inconsistent with the Bioconductor biocViews. (Hint - check for spelling, cases and plural)

Author(s)

Sonali Arora.

RepositoryDetail-class

Class "RepositoryDetail"

Description

Representation of R package repository index

Objects from the Class

Objects can be created by calls of the form `new("RepositoryDetail", ...)`.

Slots

`Title`**:** Object of class `"character"` giving the title for the repository.

`reposRoot`**:** Object of class `"character"` giving the root URL of the repository

`homeUrl`**:** Object of class `"character"` ?

`htmlDir`**:** Object of class `"character"` ?

`packageList`**:** Object of class `"list"` consisting of objects of class `PackageDetail-class`

Extends

Class `"Htmlized"`, directly.

Methods

htmlDoc `signature(object = "RepositoryDetail")`: ...

htmlFilename `signature(object = "RepositoryDetail")`: ...

htmlValue `signature(object = "RepositoryDetail")`: ...

Author(s)

Seth Falcon

validate_bioc_views *Validate a package's biocViews.*

Description

Ensures that a package has biocViews and that they are valid. Function is designed to be called from the unit tests of another package.

Usage

```
validate_bioc_views(pkg)
```

Arguments

pkg	character(1) Name of package to validate.

Value

invisible(NULL) if tests pass.

Author(s)

Dan Tenenbaum dtenenba@fhcrc.org

Examples

```
validate_bioc_views("biocViews")
```

writeBiocViews *Write a list of BiocView objects to HTML*

Description

This function serializes a list of BiocView-class objects to a series of HTML files.

Usage

```
writeBiocViews(bvList, dir, backgroundColor="transparent")
```

Arguments

bvList	A list of BiocView-class objects
dir	A character vector giving the directory where the HTML files will be written.
backgroundColor	A character vector giving the background color for the body in the CSS file.

Author(s)

Seth Falcon

See Also

getBiocViews, genReposControlFiles, write_VIEWS

---

writeHtmlDoc *Write an XML DOM containing HTML to a file*

Description

Given a DOM tree from the XML package and a filename, write the DOM to disk creating an HTML file.

Usage

```
writeHtmlDoc(html, file)
```

Arguments

html	A DOM object from the XML package
file	A string giving the filename

Author(s)

S. Falcon

---

writePackageDetailHtml *Write HTML files for packages in a CRAN-style repository*

Description

This function creates package "homepages" that describe the package and provide links to download package artifacts in the repository.

Usage

```
writePackageDetailHtml(pkgList, htmlDir = "html", backgroundColor="transparent")
```

Arguments

pkgList	A list of PackageDescription objects.
htmlDir	The files will be written to this directory.
backgroundColor	A character vector giving the background color for the body in the CSS file.

Author(s)

Seth Falcon

See Also

writeRepositoryHtml

---

writeRepositoryHtml *Write package descriptions and a repository index as HTML*

Description

This function generates an HTML file for each package in a repository and generates an index.html file that provides an alphabetized listing of the packages.

Usage

```
writeRepositoryHtml(reposRoot, title, reposUrl = "..", viewUrl = "../..",
                    reposFullUrl=reposUrl, downloadStatsUrl="",
                    devHistoryUrl="", link.rel = TRUE,
                    backgroundColor="transparent")
```

Arguments

reposRoot	string specifying the path to the root of the CRAN-style package repository.
title	string giving the title for the repository
reposUrl	string giving the prefix for URL in links generated on the package description pages. The default is "..." which works well if the package description HTML files are written to an html subdirectory under the root of the repository.
viewUrl	string giving the prefix for the URL in links to the view pages. The biocViews terms will be linked to views summary pages with this prefix.
reposFullUrl	string giving the full prefix for URL in links generated on the package description pages. The default is reposUrl.
downloadStatsUrl	string giving the prefix for the URL in links to the download history statistics pages.
devHistoryUrl	string giving the prefix for the URL in links to the development changelog.

link.rel	logical indicating whether the index page should generate relative URL links. The default is TRUE. If you are generating HTML for a remote repository, you will want to set this to FALSE.
backgroundColor	A character vector giving the background color for the body in the CSS file.

Author(s)

Seth Falcon

`writeRFilesFromVignettes` *Write R files from vignettes*

Description

Ensures that .R files from vignette code chunks are written out.

Usage

```
writeRFilesFromVignettes(reposRoot, reposUrl="..",
                                    viewUrl="../..", reposFullUrl=reposUrl,
                                    downloadStatsUrl="", devHistoryUrl="")
```

Arguments

reposRoot	Root directory of a CRAN-style repository
reposUrl	URL of repository
viewUrl	url of VIEWS file
reposFullUrl	Full URL of VIEWS file
downloadStatsUrl	URL to download stats page
devHistoryUrl	Dev history URL

writeTopLevelView *Write the view for the root of a vocabulary to disk*

Description

Given a directory and a vocabulary represented as a graphNEL containing a DAG of terms, write the top-level term to disk as HTML.

This assumes your vocabulary has a single term with no parents.

Usage

```
writeTopLevelView(dir, vocab)
```

Arguments

dir	A string giving a directory in which to write the HTML file
vocab	A graphNEL instance giving the DAG of terms. It should have a root node. That is, there should be exactly one node with no incoming edges.

Author(s)

S. Falcon

write_REPOSITORY *Write a REPOSITORY control file for a CRAN-style package repository*

Description

This function writes a REPOSITORY file at the top-level of a CRAN-style repository. This file is DCF formatted and describes the location of packages available in the repository. Here is an example for a repository containing only source and Windows binary packages:

```
source: src/contrib
win.binary: bin/windows/contrib/2.6
win64.binary: bin/windows64/contrib/2.6
mac.binary.leopard: bin/mac/leopard/contrib/2.6
provides: source, win.binary, win64.binary, mac.binary.leopard
```

Usage

```
write_REPOSITORY(reposRootPath, contribPaths)
```

Arguments

reposRootPath	character vector containing the path to the CRAN-style repository root directory.
contribPaths	A named character vector. Valid names are source, win.binary, win64.binary, mac.binary, and mac.binary.leopard. Values indicate the paths to the package archives relative to the reposRoot.

Author(s)

Seth Falcon

See Also

write_PACKAGES, extractVignettes, genReposControlFiles, write_VIEWS

---

write_SYMBOLS *Write a SYMBOLS file*

Description

Writes a DCF formatted file, SYMBOLS, containing the symbols exported by each package in a directory containg R package source directories.

Usage

```
write_SYMBOLS(dir, verbose = FALSE, source.dirs=FALSE)
```

Arguments

dir	The root of a CRAN-style package repository containing source packages. When source.dirs is TRUE, dir should be a directory containing R package source directories
verbose	Logical. When TRUE, progress is printed to the standard output.
source.dirs	Logical. When TRUE, interpret dir as a directory containing source package directories. When FALSE, the default, dir is assumed to be the root of a CRAN-style package repository and the function will operate on the source package tarballs in dir/src/contrib.

Value

Returns NULL. Called for the side-effect of creating a SYMBOLS file in dir.

Author(s)

S. Falcon

See Also

write_PACKAGES write_VIEWS

write_VIEWS *Write a VIEWS control file for a CRAN-style package repository*

Description

This function writes a VIEWS file to the top-level of a CRAN-style package repository. The VIEWS file is in DCF format and describes all packages found in the repository.

The VIEWS file contains the complete DESCRIPTION file for each source package in the repository. In addition, metadata for available binary packages and vignettes is centralized here.

Usage

```
write_VIEWS(reposRootPath, fields = NULL,
            type = c("source", "win.binary", "win64.binary",
                    "mac.binary", "mac.binary.leopard","mac.binary.mavericks"),
            verbose = FALSE, vignette.dir = "vignettes")
```

Arguments

reposRootPath	character vector containing the path to the CRAN-style repository root directory.
fields	Any additional fields to include. You shouldn't need this, but if you have added fields to the DESCRIPTION files of the packages in the repository, you may want it.
type	One of source, win.binary, win64.binary, mac.binary, or mac.binary.leopard indicating which set of packages should be used to build up the "shared" information. Since a repository can contain different versions of the same package (source vs binary) the shared information may not be reliable.
verbose	logical, if TRUE, print progress messages.
vignette.dir	character specifying where to look for vignettes.

Warning

This function uses a private function from the tools package: tools:::.build_repository_package_db.

Author(s)

Seth Falcon

See Also

write_PACKAGES, extractVignettes, genReposControlFiles, write_REPOSITORY

Index