Package ‘wikkitidy’

August 17, 2024

Title Tidy Analysis of Wikipedia

Version 0.1.13

Description Access 'Wikipedia' through the several 'MediaWiki' APIs (<https://www.mediawiki.org/wiki/API>), as well as through the 'XTools' API (<https://www.mediawiki.org/wiki/XTools/API>). Ensure your API calls are correct, and receive results in tidy tibbles.

License MIT + file LICENSE

URL https://wikihistories.github.io/wikkitidy/, https://github.com/wikihistories/wikkitidy

BugReports https://github.com/wikihistories/wikkitidy/issues

Depends R (>= 2.10)

Imports cli, dplyr, glue, httr2, lubridate, magrittr, openssl, pillar, purrr, rlang (>= 0.4.11), stringr, tibble, vctrs, webfakes

Suggests covr, igraph, roxygen2, testthat (>= 3.0.0), tidyr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Michael Falk [aut, cre, cph] (<https://orcid.org/0000-0001-9261-8390>)

Maintainer Michael Falk <michaelgfalk@gmail.com>

Repository CRAN

Date/Publication 2024-08-17 10:30:06 UTC

Contents

get_diff *Search for insertions, deletions or relocations of text between two versions of a Wikipedia page*

Description

Any two revisions of a Wikipedia page can be compared using the 'diff' tool. The tool compares the 'from' revision to the 'to' revision, looking for insertions, deletions or relocations of text. This operation can be performed in any order, across any span of revisions.

Usage

```
get_diff(from, to, language = "en", simplify = TRUE)
```

Arguments

`from`	Vector of revision ids
`to`	Vector of revision ids
`language`	Vector of two-letter language codes (will be recycled if length==1)
`simplify`	logical: should R simplify the result (see return)

Value

The return value depends on the `simplify` parameter.

- If `simplify` == TRUE: A list of tibble::tbl_df objects the same length as `from` and `to`. Most of the response data is stripped away, leaving just the textual differences between the revisions, their location, type and 'highlightRanges' if the textual differences are complicated.
- If `simplify` == FALSE: A list the same length as `from` and `to` containing the full wikidiff2 response for each pair of revisions. This response includes additional data for displaying diffs onscreen.

Examples

```
# Compare revision 847170467 to 851733941 on English Wikipedia
get_diff(847170467, 851733941)

# The function is vectorised, so you can compare multiple pairs of revisions
# in a single call
# See diffs for the last two revisions of the Main Page
revisions <- wiki_action_request() %>%
  query_by_title("Main Page") %>%
  query_page_properties(
    "revisions",
    rvlimit = 2, rvprop = "ids", rvdir = "older"
  ) %>%
  gracefully(next_result)

if (tibble::is_tibble(revisions)) {
  revisions <- revisions %>%
    tidyr::unnest(cols = c(revisions)) %>%
    dplyr::mutate(diffs = get_diff(from = parentid, to = revid))

  print(revisions)
}
```

`get_history_count` *Count how many times Wikipedia articles have been edited*

Description

Count how many times Wikipedia articles have been edited

Usage

```
get_history_count(
  title,
  type = c("edits", "anonymous", "bot", "editors", "minor", "reverted"),
  from = NULL,
  to = NULL,
  language = "en"
)
```

Arguments

`title`	A vector of article titles
`type`	The type of edit to count
`from`	Optional: a vector of revision ids
`to`	Optional: a vector of revision ids
`language`	Vector of two-letter language codes for Wikipedia editions

Value

A tibble::tbl_df with two columns:

- 'count': integer, the number of edits of the given type
- 'limit': logical, whether the 'count' exceeds the API's limit. Each type of edit has a different limit. If the 'count' exceeds the limit, then the limit is returned as the count and 'limit' is set to TRUE

Examples

```
# Get the number of edits made by auto-confirmed editors to a page between
# revisions 384955912 and 406217369
get_history_count("Jupiter", "editors", 384955912, 406217369)

# Compare which authors have the most edit activity
authors <- tibble::tribble(
  ~author,
  "Jane Austen",
  "William Shakespeare",
  "Emily Dickinson"
) %>%
  dplyr::mutate(get_history_count(author))
authors
```

`get_query_results` *Perform a query using the Rhrefhttps://www.mediawiki.org/wiki/Special:MyLanguage/API:Main_ Action API*

Description

`next_result()` sends exactly one request to the server.

`next_batch()` requests results from the server until data is complete the latest batch of pages in the result.

`retrieve_all()` keeps requesting data until all the pages from the query have been returned.

Usage

```
next_result(x)

next_batch(x)

retrieve_all(x)
```

Arguments

x	The query. Either a wiki_action_request or a query_tbl.

Details

It is rare that a query can be fulfilled in a single request to the server. There are two ways a query can be incomplete. All queries return a list of pages as their result. The result may be incomplete because not all the data for each page has been returned. In this case the *batch* is incomplete. Or the data may be complete for all pages, but there are more pages available on the server. In this case the query can be *continued*. Thus the three functions for `next_result()`, `next_batch()` and `retrieve_all()`.

Value

A query_tbl containing results of the query. If `x` is a query_tbl, then the function will return a new data with the new data appended to it. If `x` is a wiki_action_request, then the returned query_tbl will contain the necessary data to supply future calls to `next_result()`, `next_batch()` or `retrieve_all()`.

Examples

```
# Try out a request using next_result(), then retrieve the rest of the
# results. The clllimt limits the first request to 40 results.
preview <- wiki_action_request() %>%
  query_by_title("Steve Wozniak") %>%
  query_page_properties("categories", cllimit = 40) %>%
  gracefully(next_result)
preview

all_results <- preview %>%
  gracefully(retrieve_all)
all_results

# tidyr is useful for list-columns.
if (tibble::is_tibble(all_results)) {
  all_results %>%
    tidyr::unnest(cols=c(categories), names_sep = "_")
}
```

get_rest_resource *Get resources from one of Wikipedia's* R*hrefhttps://www.mediawiki.org/wiki/APItwo REST APIs*

Description

This function is intended for developer use. It makes it easy to quickly generate vectorised calls to the different APIs.

Usage

```
get_rest_resource(
  ...,
  language = "en",
  api = c("core", "wikimedia", "wikimedia_org", "xtools"),
  response_format = c("json", "html"),
  response_type = NULL,
  failure_mode = c("error", "quiet")
)
```

Arguments

`...`	<`dynamic-dots`> The URL components and query parameters of the desired resources. Names of the arguments are ignored. The function follows the tidyverse vector recycling rules, so all vectors must have the same length or be of length one. Unnamed arguments will be appended to the URL path; named arguments will be added as query parameters
`language`	Character vector of two-letter language codes
`api`	The desired REST api: "core", "wikimedia", "wikimedia_org", or "xtools"
`response_format`	The expected Content-Type of the response. Currently "html" and "json" are supported.
`response_type`	The schema of the response. If supplied, the results will be parsed using the schema.
`failure_mode`	How to respond if a request fails "error", the default: raise an error "quiet", silently return NA

Value

A list of responses. If `response_format` == "json", then the responses will be simple R lists. If `response_format` == "html", then the responses will `xml_document` objects. If `response_type` is supplied, the response will be coerced into a tibble::tbl_df or vector using the relevant schema. If the response is a 'scalar list' (i.e. a list of length == 1), then it is silently unlisted, returning a simple list or vector.

gracefully — *Gracefully request a resource from Wikipedia*

Description

The main purpose of this function is to enable examples using live resources in the documentation. Examples must not throw errors, according to CRAN policy. If you wrap a requesting method in `gracefully`, then any errors of type `httr2_http` will be caught and no error will be thrown.

Usage

```
gracefully(request_object, request_method)
```

Arguments

`request_object`	A `httr2_request` object describing a query to a Wikimedia Action API
`request_method`	The desired function for performing the request, typically one of those in get_query_results

Value

The output of `request_method` called on `request_object`, if the request was successful. Otherwise a `httr2_response` object with details of the failed request.

Examples

```
# This fails without throwing an error
req <- httr2::request(httr2::example_url()) |>
  httr2::req_url_path("/status/404")

resp <- gracefully(req, httr2::req_perform)

print(resp)

# This request succeeds
req <- httr2::request(httr2::example_url())

resp <- gracefully(req, httr2::req_perform)

print(resp)
```

new_generator_query *Constructor for generator query type*

Description

Construct a new query to a generator module of the Action API. This low-level constructor only performs basic type-checking. It is your responsibility to ensure that the chosen `generator` is an existing API endpoint, and that you have composed the query correctly. For a more user-friendly interface, use query_generate_pages.

Usage

```
new_generator_query(.req, generator, ...)
```

Arguments

`.req`	A `query/action_api/httr2_request` object, or a generator query as returned by this function.
`generator`	The generator to add to the query. If the generator is based on a property module, then `.req` must be a subtype of `prop/query/action_api/httr2_request`. If the generator is based on a list module, then `.req` must subclass `query/action_api/httr2_request` directly.
`...`	<`dynamic-dots`> Further parameters to the generator

Value

The output type depends on the input. If `.req` is a `query/action_api/httr2_request`, then the output will be a `generator/query/action_api/httr2_request`. If `.req` is a `prop/query/action_api/httr2_request`, then the return object will be a subclass of the passed request, with "generator" as the first term in the class vector, i.e. `generator/(titles|pageids|revids)/prop/query/action_api/httr2_request`.

Examples

```
# Build a generator query using a list module
# List all members of Category:Physics on English Wikipedia
physics <- wiki_action_request() %>%
  new_generator_query("categorymembers", gcmtitle = "Category:Physics")

# Build a generator query on a property module
# Generate the pages that are linked to Albert Einstein's page on English
# Wikipedia
einstein_categories <- wiki_action_request() %>%
  new_prop_query("titles", "Albert Einstein") %>%
  new_generator_query("iwlinks")
```

new_list_query *Constructor for Rhrefhttps://www.mediawiki.org/wiki/API:Listslist queries*

Description

This low-level constructor only performs basic type checking.

Usage

```
new_list_query(.req, list, ...)

## S3 method for class 'list'
new_list_query(.req, list, ...)

## S3 method for class 'generator'
new_list_query(.req, list, ...)

## S3 method for class 'prop'
new_list_query(.req, list, ...)

## S3 method for class 'query'
new_list_query(.req, list, ...)
```

Arguments

`.req`	A `query/action_api/httr2_request` object, or a `list/query/action_api/httr2_request` as returned by this function.
`list`	The list module to add to the query
`...`	<`dynamic-dots`> Parameters to the list module

Value

An object of type `list/query/action_api/httr2_request`.

Examples

```
# Create a query to list all members of Category:Physics
physics_query <- wiki_action_request() %>%
  new_list_query("categorymembers", cmtitle="Category:Physics")
```

new_prop_query *Constructor for the property query type*

Description

The intended use for this query is to set the 'titles', 'pageids' or 'revids' parameter, and enforce that only one of these is set. All property modules API in the Action API require this parameter to be set, or they require a generator parameter to be set instead. The prop/query type is an abstract type representing the three possible kinds of property query that do not rely on a generator (see below on the return value). A complication is that a prop/query can *itself* be used as the basis for a generator.

Usage

```
new_prop_query(.req, by, pages, ...)
```

Arguments

.req	A query/action_api/httr2_request object, or a prop query object as returned by this function. This parameter is covariant on the type, so you can also pass all subtypes of prop.
by	The type of page. Allowed values are: pageids, titles, revids
pages	A string, the pages to query by, corresponding to the 'by' parameter. Multiple values should be separated with "|"
...	<dynamic-dots> Further parameters to the query

Value

A properly qualified prop/query object. There are six possibilities:

- titles/prop/query
- pageids/prop/query
- revids/prop/query
- generator/titles/prop/query
- generator/pageids/prop/query
- generator/revids/prop/query

Examples

```
# Build a query on a set of pageids
# 963273 and 1159171 are Kate Bush albums
bush_albums_query <- wiki_action_request() %>%
  new_prop_query("pageids", "963273|1159171")
```

page_vector_functions *Get data about pages from their titles*

Description

`get_latest_revision()` returns metadata about the latest revision of each page.

`get_page_html()` returns the rendered html for each page.

`get_page_summary()` returns metadata about the latest revision, along with the page description and a summary extracted from the opening paragraph

`get_page_related()` returns summaries for 20 related pages for each passed page

`get_page_talk()` returns structured talk page content for each title. You must ensure to use the title for the Talk page itself, e.g. "Talk:Earth" rather than "Earth"

`get_page_langlinks()` returns interwiki links for each title

Usage

```
get_latest_revision(title, language = "en")

get_page_html(title, language = "en")

get_page_summary(title, language = "en")

get_page_related(title, language = "en")

get_page_talk(title, language = "en")

get_page_langlinks(title, language = "en")
```

Arguments

`title`	A character vector of page titles.
`language`	A character vector of two-letter language codes, either of length 1 or the same length as `title`

Value

A list, vector or tibble, the same length as `title`, with the desired data.

Examples

```
# Get language links for a known page on English Wikipedia
get_page_langlinks("Charles Harpur")

# Many of these functions return a list of data frames. Tidyr can be useful.
# Get 20 related pages for German City
cities <- tibble::tribble(
  ~city,
  "Berlin",
  "Darmstadt",
) %>%
  dplyr::mutate(related = get_page_related(city))
cities

# Unest to get one row per related page:
tidyr::unnest(cities, "related")

# The functions are vectorised over title and language
# Find all articles about Joanna Baillie, and retrieve summary data for
# the first two.
baillie <- get_page_langlinks("Joanna Baillie") %>%
  dplyr::slice(1:2) %>%
  dplyr::mutate(get_page_summary(title = title, language = code))
baillie
```

`query_by_` *Query the Rhrefhttps://www.mediawiki.org/wiki/API:Main_pageMediaWiki Action API using a vector of Wikipedia pages*

Description

These functions help you to build a query for the MediaWiki Action API if you already have a set of pages that you wish to investigate. These functions can be combined with query_page_properties to choose which properties to return for the passed pages.

Usage

```
query_by_title(.req, title)

query_by_pageid(.req, pageid)

query_by_revid(.req, revid)
```

Arguments

`.req`	A wiki_action_request query to modify
`title`	A character vector of page titles
`pageid`	A character or numeric vector of page ids
`revid`	A character or numeric vector of revision ids

Details

If you don't already know which pages you wish to examine, you can build a query to find pages that meet certain criteria using query_list_pages or query_generate_pages.

Value

A request object of type `pages/query/action_api/httr2_request`. To perform the query, pass the object to next_batch or retrieve_all

See Also

`gracefully()`

Examples

```
# Retrieve the categories for Charles Harpur's Wikipedia page
 resp <- wiki_action_request() %>%
  query_by_title("Charles Harpur") %>%
  query_page_properties("categories") %>%
  gracefully(next_batch)
```

---

query_category_members — *Explore Wikipedia's category system*

Description

These functions provide access to the CategoryMembers endpoint of the Action API.

`query_category_members()` builds a generator query to return the members of a given category.

`build_category_tree()` finds all the pages and subcategories beneath the passed category, then recursively finds all the pages and subcategories beneath them, until it can find no more subcategories.

Usage

```
query_category_members(
  .req,
  category,
  namespace = NULL,
  type = c("file", "page", "subcat"),
  limit = 10,
  sort = c("sortkey", "timestamp"),
  dir = c("ascending", "descending", "newer", "older"),
  start = NULL,
  end = NULL,
  language = "en"
)

build_category_tree(category, language = "en")
```

Arguments

`.req` A query request object

`category` The category to start from. `query_category_members()` accepts either a numeric pageid or the page title. `build_category_tree()` accepts a vector of page titles.

`namespace` Only return category members from the provided namespace

`type` Alternative to `namespace`: the type of category member to return. Multiple types can be requested using a character vector. Defaults to all.

`limit` The number to return each batch. Max 500.

`sort` How to sort the returned category members. 'timestamp' sorts them by the date they were included in the category; 'sortkey' by the category member's unique hexadecimal code

`dir` The direction in which to sort them

`start` If `sort` == 'timestamp', only return category members from after this date. The argument is parsed by `lubridate::as_date()`

`end` If `sort` == 'timestamp', only return category members included in the category from before this date. The argument is parsed by `lubridate::as_date()`

`language` The language edition of Wikipedia to query

Value

`query_category_members()`: A request object of type `generator/query/action_api/httr2_request`, which can be passed to `next_batch()` or `retrieve_all()`. You can specify which properties to retrieve for each page using `query_page_properties()`.

`build_category_tree()`: A list containing two dataframes. `nodes` lists all the subcategories and pages found underneath the passed categories. `edges` records the connections between them. The `source` column gives the pageid of the parent category, while the `target` column gives the pageid of any categories, pages or files contained within the `source` category. The `timestamp` records the moment when the `target` page or subcategory was included in the `source` category. The two dataframes in the list can be passed to igraph::graph_from_data_frame for network analysis.

See Also

`gracefully()`

Examples

```
# Get the first 10 pages in 'Category:Physics' on English Wikipedia
physics_members <- wiki_action_request() %>%
  query_category_members("Physics") %>%
  gracefully(next_batch)
physics_members


# Build the tree of all albums for the Melbourne band Custard
tree <- build_category_tree("Category:Custard_(band)_albums")
```

```
tree

# For network analysis and visualisation, you can pass the category tree
# to igraph
tree_graph <- igraph::graph_from_data_frame(tree$edges, vertices = tree$nodes)
tree_graph
```

query_generate_pages *Generate pages that meet certain criteria, or which are related to a set of known pages by certain properties*

Description

Many of the endpoints on the Action API can be used as generators. Use list_all_generators() to see a complete list. The main advantage of using a generator is that you can chain it with calls to query_page_properties() to find out specific information about the pages. This is not possible for queries constructed using query_list_pages().

Usage

```
query_generate_pages(.req, generator, ...)

list_all_generators()
```

Arguments

.req	A httr2_request, e.g. generated by wiki_action_request
generator	The generator module you wish to use. Most list and property modules can be used, though not all.
...	<dynamic-dots> Additional parameters to the generator

Details

There are two kinds of generator: list-generators and prop-generators. If using a prop-generator, then you need to use a query_by_() function to tell the API where to start from, as shown in the examples.

To set additional parameters to a generator, prepend the parameter with "g". For instance, to set a limit of 10 to the number of pages returned by the categorymembers generator, set the parameter gcmlimit = 10.

Value

query_generate_pages: The modified request, which can be passed to next_batch or retrieve_all as appropriate.

list_all_generators: a tibble of all the available generator modules. The name column gives the name of the generator, while the group column indicates whether the generator is based on a list module or a property module. Generators based on property modules can only be added to a query if you have already used query_by_ to specify which pages' properties should be generated.

See Also

`gracefully()`

Examples

```
# Search for articles about seagulls
seagulls <- wiki_action_request() %>%
  query_generate_pages("search", gsrsearch = "seagull") %>%
  gracefully(next_batch)

seagulls
```

`query_list_pages` *List pages that meet certain criteria*

Description

See API:Lists for available list actions. Each list action returns a list of pages, typically including their pageid, namespace and title. Individual lists have particular properties that can be requested, which are usually prefaced with a two-word code based on the name of the list (e.g. specific properties for the `categorymembers` list action are prefixed with `cm`).

Usage

```
query_list_pages(.req, list, ...)

list_all_list_modules()
```

Arguments

`.req`	A httr2_request, e.g. generated by `wiki_action_request`
`list`	The type of list to return
`...`	<`dynamic-dots`> Additional parameters to the query, e.g. to set configure list

Details

When the request is performed, the data is returned in the body of the request under the `query` object, labeled by the chosen list action.

If you want to study the actual pages listed, it is advisable to retrieve the pages directly using a generator, rather than listing their IDs using a list action. When using a list action, a second request is required to get further information about each page. Using a generator, you can query pages and retrieve their relevant properties in a single API call.

Value

An HTTP response: an S3 list with class httr2_request

See Also

`gracefully()`

Examples

```
# Get the ten most recently added pages in Category:Physics
physics_pages <- wiki_action_request() %>%
  query_list_pages("categorymembers",
    cmsort = "timestamp",
    cmdir = "desc", cmtitle = "Category:Physics"
  ) %>%
  gracefully(next_batch)

physics_pages
```

`query_page_properties` *Choose properties to return for pages from the action API*

Description

See API:Properties for a list of available properties. Many have additional parameters to control their behavior, which can be passed to this function as named arguments.

Usage

```
query_page_properties(.req, property, ...)

list_all_property_modules()
```

Arguments

`.req`	A httr2_request, e.g. generated by `wiki_action_request`
`property`	The property to request
`...`	<`dynamic-dots`> Additional parameters to pass, e.g. to modify what is returned by the property request

Details

query_page_properties is not useful on its own. It must be combined with a query_by_ function or query_generate_pages to specify which pages properties are to be returned. It should be noted that many of the API:Properties modules can themselves be used as generators. If you wish to use a property module in this way, then you must use query_generate_pages, passing the name of the property module as the `genenerator`.

Value

An HTTP response: an S3 list with class httr2_request

See Also

gracefully()

Examples

```
# Search for articles about seagulls and retrieve their number of
# watchers

resp <- wiki_action_request() %>%
  query_generate_pages("search", gsrsearch = "seagull") %>%
  query_page_properties("info", inprop = "watchers") %>%
  gracefully(next_batch) %>%
  dplyr::select(pageid, ns, title, watchers)
resp
```

query_tbl *Representation of Wikipedia data returned from an Rhrefhttps://www.mediawiki.org/wiki/API:QueryAction API Query module as tibble, with request metadata stored as attributes.*

Description

Representation of Wikipedia data returned from an Action API Query module as tibble, with request metadata stored as attributes.

Usage

```
query_tbl(x, request, continue, batchcomplete)
```

Arguments

x	A tibble
request	The httr2_request object used to generate the tibble
continue	The continue parameter returned by the API
batchcomplete	The batchcomplete parameter returned by the API

Value

A tibble: an S3 data.frame with class `query_tbl`.

verify_xml_integrity *Check that a Wikimedia XML file has not been corrupted*

Description

The Wikimedia Foundation publishes MD5 checksums for all its database dumps. This function looks up the published sha1 checksums based on the file name, then compares them to the locally calcualte has using the openssl package.

Usage

```
verify_xml_integrity(path)
```

Arguments

path	The path to the file

Value

True (invisibly) if successful, otherwise error

wikimedia_rest_apis *Build a REST request to one of the Wikimedia Foundation's central APIs*

Description

wikimedia_org_rest_request() builds a request for the wikimedia.org REST API, which provides statistical data about Wikimedia Foundation projects

xtools_rest_request() builds a request to the XTools API, which provides additional statistical data about Wikimedia foundation projects

Usage

```
wikimedia_org_rest_request(endpoint, ..., language = "en")

xtools_rest_request(endpoint, ..., language = "en")
```

Arguments

endpoint	The endpoint for the specific kind of request; for wikimedia apis, this comprises the path components in between the general API endpoint and the component specifying the project to query
...	<dynamic-dots> Components to add to the URL. Unnamed arguments are added to the path of the request, while named arguments are added as query parameters.
language	Two-letter language code for the desired Wikipedia edition.

Value

A `wikimedia_org/rest` or `xtools/rest` object, an S3 vector that subclasses httr2::request.

Examples

```
# Build request for articleinfo about Kate Bush's page on English Wikipedia
request <- xtools_rest_request("page/articleinfo", "Kate_Bush")

# Build request for most-viewed pages on German Wikipedia in July 2020
request <- wikimedia_org_rest_request(
    "metrics/pageviews/top",
    "all-access", "2020", "07", "all-days",
    language = "de"
    )
```

`wikipedia_rest_apis` *Build a REST request to one of Wikipedia's specific REST APIs*

Description

`core_request_request()` builds a request for the MediaWiki Core REST API, the basic REST API available on all MediaWiki wikis.

`wikimedia_rest_request()` builds a request for the Wikimedia REST API, an additional api just for Wikipedia and other wikis managed by the Wikimedia Foundation

Usage

```
core_rest_request(..., language = "en")

wikimedia_rest_request(..., language = "en")
```

Arguments

`...`	<`dynamic-dots`> Components to add to the URL. Unnamed arguments are added to the path of the request, while named arguments are added as query parameters.
`language`	The two-letter language code for the Wikipedia edition

Value

A `core/rest`, `wikimedia/rest`, object, an S3 vector that subclasses `httr2_request` (see httr2::request). The request needs to be passed to httr2::req_perform to retrieve data from the API.

Examples

```
# Get the html of the 'Earth' article on English Wikipedia
response <- core_rest_request("page", "Earth", "html") %>%
  httr2::req_perform()

response <- wikimedia_rest_request("page", "html", "Earth") %>%
  httr2::req_perform()

# Some REST requests take query parameters. Pass these as named arguments.
# To search German Wikipedia for articles about Goethe
response <- core_rest_request("search/page", q = "Goethe", limit = 2, language = "de") %>%
  httr2::req_perform() %>%
  httr2::resp_body_json()
```

wiki_action_request *Query Wikipedia using the Rhrefhttps://www.mediawiki.org/wiki/API:Main_pageMediaWiki Action API*

Description

Wikipedia exposes a To build up a query, you first call `wiki_action_request()` to create the basic request object, then use the helper functions `query_page_properties()`, `query_list_pages()` and `query_generate_pages()` to modify the request, before calling `next_batch()` or `retrieve_all()` to perform the query and download results from the server.

Usage

```
wiki_action_request(..., action = "query", language = "en")
```

Arguments

`...`	<`dynamic-dots`> Parameters for the request
`action`	The action to perform, typically 'query'
`language`	The language edition of Wikipedia to request, e.g. 'en' or 'fr'

Details

wikkitidy provides an ergonomic API for the Action API's Query modules. These modules are most useful for researchers, because they allow you to explore the structure of Wikipedia and its back pages. You can obtain a list of available modules in your R console using `list_all_property_modules()`, `list_all_list_modules()` and `list_all_generators()`,

Value

An `action_api` object, an S3 list that subclasses httr2::request. The dependencies between different aspects of the Action API are complex. At the time of writing, there are five major subclasses of `action_api/httr2_request`:

- `generator/action_api/httr2_request`, returned (sometimes) by query_generate_pages
- `list/action_api/httr2_request`, returned by query_list_pages
- `titles`, `pageids` and `revids/action_api/httr2_request`, returned by the various query_by_ functions

  You can use query_page_properties to modify any kind of query *except* for `list` queries: indeed, the central limitation of the `list` queries is that you cannot choose what properties to return for the pages the meet the given criterion. The concept of a `generator` is complex. If the `generator` is based on a property module, then it must be combined with a query_by_ function to produce a valid query. If the generator is based on a list module, then it *cannot* be combined with a query_by_ query.

See Also

`gracefully()`

Examples

```
# List the first 10 pages in the category 'Australian historians'
historians <- wiki_action_request() %>%
  query_list_pages(
    "categorymembers",
    cmtitle = "Category:Australian_historians",
    cmlimit = 10
  ) %>%
  gracefully(next_batch)
historians
```

`wikkitidy_example` *Get path to wikkitidy example*

Description

wikkitidy comes bundled with a number of sample files in its `inst/extdata` directory. This function make them easy to access

Usage

```
wikkitidy_example(file = NULL)
```

Arguments

`file`	Name of file. If `NULL`, the example files will be listed.

Value

A character vector, containing either the path of the chosen file, or the nicknames of all available example files.

Examples

```
wikkitidy_example()
wikkitidy_example("akan_wiki")
```

---

xtools_page *Access page-level statistics from the Rhrefhttps://www.mediawiki.org/wiki/XTools/API/PageXTools Page API endpoint*

Description

get_xtools_page_info() returns basic statistics about articles' history and quality, including their total edits, creation date, and assessment value (good, featured etc.)

get_xtools_page_prose() returns statistics about the word counts and referencing of articles

get_xtools_page_links() returns the number of ingoing and outgoing links to articles, including redirects

get_xtools_page_top_editors() returns the list of top editors for articles, with optional filters by date range and non-bot status

get_xtools_page_assessment() returns more detailed statistics about articles' assessment status and Wikiproject importance levels

Usage

```
get_xtools_page_info(
  title,
  language = "en",
  failure_mode = c("error", "quiet")
)

get_xtools_page_prose(
  title,
  language = "en",
  failure_mode = c("error", "quiet")
)

get_xtools_page_links(
  title,
  language = "en",
  failure_mode = c("error", "quiet")
)

get_xtools_page_top_editors(
  title,
  start = NULL,
  end = NULL,
  limit = 1000,
  nobots = FALSE,
  language = "en",
  failure_mode = c("error", "quiet")
)

get_xtools_page_assessment(
  title,
  classonly = FALSE,
  language = "en",
  failure_mode = c("error", "quiet")
)
```

Arguments

`title` Character vector of page titles

`language` Language code for the version of Wikipedia to query

`failure_mode` What to do if no data is found. See `get_rest_resource()`

`start` A character vector or date object (optional): the start date for calculating top editors

`end` A character vector or date object (optional): the end date for calculating top editors

`limit` An integer: the maximum number of top editors to return

`nobots` TRUE or FALSE: if TRUE, bots are excluded from the top editor calculation

`classonly` TRUE or FALSE: if TRUE, only return the article's assessment status, without Wikiproject information

Value

A list or tbl of results, the same length as `title`. **NB:** The results for `get_xtools_page_assessment` are still not parsed properly.

Examples

```
# Get basic statistics about Erich Auerbach on German Wikipedia
auerbach <- get_xtools_page_info("Erich Auerbach", language = "de")
auerbach
```

Index