Package 'ottrpal'

May 9, 2026

Type Package

Title Companion Tools for Open-Source Tools for Training Resources (OTTR)

Version 2.0.0

Description Tools for converting Open-Source Tools for Training Resources (OTTR) courses into Leanpub or Coursera courses. 'ottrpal' is for use with the OTTR Template repository to create courses.

License GPL-3

URL https://github.com/jhudsl/ottrpal

BugReports https://github.com/jhudsl/ottrpal/issues

Imports dplyr, httr, readr, rvest, stringr, tidyr, gitcreds, knitr, purrr, rmarkdown, rprojroot, webshot2, openssl, spelling, xml2, R.utils, googledrive, jsonlite, yaml

Suggests testthat, bookdown, quarto

VignetteBuilder knitr

ByteCompile true

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Candace Savonen [aut, cre] (ORCID: <https://orcid.org/0000-0001-6331-7070>), Carrie Wright [ctb], Howard Baek [ctb], Kate Isaac [ctb]

Maintainer Candace Savonen <cansav09@gmail.com>

Repository CRAN

Date/Publication 2025-03-31 20:50:02 UTC

Contents

`app_set_up` *App Set Up*

Description

This is a function that sets up the app. It's generally called by another function

Usage

```
app_set_up(app_name = "google")
```

Arguments

`app_name`	app would you like to authorize? Supported apps are 'google' 'calendly' and 'github'

`authorize` *Authorize R package to access endpoints*

Description

This is a function to authorize the R package to access APIs interactively. To learn more about the privacy policy for ottrpal [read here](https://www.ottrproject.org/privacypolicy.html)

Usage

```
authorize(app_name = NULL, cache = FALSE, ...)
```

Arguments

`app_name`	app would you like to authorize? Supported apps are 'google' 'calendly' and 'github'
`cache`	Should the token be cached as an .httr-oauth file or API keys stored as global options?
`...`	Additional arguments to send to oauth2.0_token

Value

API token saved to the environment or the cache so it can be grabbed by functions

Examples

```
## Not run:

authorize()

authorize("github")

authorize("google")

## End(Not run)
```

`auth_from_secret` *Use secrets to authorize R package to access endpoints*

Description

This is a function to authorize ottrpal to access calendly, github or google noninteractively from passing in a keys or tokens.

Usage

```
auth_from_secret(
  app_name,
  token,
  access_token,
  refresh_token,
  cache = FALSE,
  in_test = FALSE
)
```

Arguments

`app_name`	Which app are you trying to authorize? 'google', 'calendly' or 'github'?
`token`	For calendly or github, pass in the API key or Personal Access Token that you have set up from going to https://github.com/settings/tokens/new or https://calendly.com/integrations/api_ respectively.
`access_token`	For Google, access token can be obtained from running authorize interactively: token <-authorize(); token$credentials$access_token
`refresh_token`	For Google, refresh token can be obtained from running authorize interactively: token <-authorize(); token$credentials$refresh_token
`cache`	Should the credentials be cached? TRUE or FALSE?
`in_test`	If setting up auth in a test, set to TRUE so that way the authorization doesn't stick

Value

OAuth token saved to the environment so the package access the API data

Examples

```
## Not run:

# Example for GitHub
# You go to https://github.com/settings/tokens/new to get a Personal Access Token
auth_from_secret("github", token = "ghp_a_github_pat_here")

# Example for authorizing for Google
token <- authorize("google")
auth_from_secret(
  app_name = "google",
  access_token = token$credentials$access_token,
  refresh_token = token$credentials$refresh_token
)

## End(Not run)
```

---

bad_quiz_path *Path to bad example quiz*

Description

Path to bad example quiz

Usage

```
bad_quiz_path()
```

Value

The file path to an example bad quiz included in the package that will fail the quiz checks.

Examples

```
quiz_path <- bad_quiz_path()
```

borrow_chapter *Borrow/link a chapter from another bookdown course*

Description

If you have two courses that the content and topics overlap, you may want to share written material between the two. But, if you copy and paste to share material this would create a maintenance problem because if you update one you will need to remember to copy over the other! To borrow a chapter from another course, create an .Rmd as you normally would, with an [‘H1‘ title](https://www.markdownguide.org/basic-syntax/) if you wish. Then in a code chunk, use cow::borrow_chapter() to have the content from an Rmd from another repository knitted into the Rmd.

Usage

```
borrow_chapter(
  doc_path,
  repo_name = NULL,
  remove_h1 = FALSE,
  tag_replacement = NULL,
  branch = "main",
  token = NULL,
  base_url = "https://raw.githubusercontent.com",
  dest_dir = file.path("resources", "other_chapters")
)
```

Arguments

doc_path	A file path of markdown or R Markdown document of the chapter in the repository you are retrieving it from that you would like to include in the current document. e.g "docs/intro.md" or "intro.md"
repo_name	A character vector indicating the repo name of where you are borrowing from. e.g. "jhudsl/OTTR_Template/". For a Wiki of a repo, use "wiki/jhudsl/OTTR_Template/" If nothing is provided, will look for local file.
remove_h1	If TRUE Remove all h1 headers.
tag_replacement	An optional list of tags that need to be replaced in the child document upon bringing it into the render.
branch	Default is to pull from main branch, but need to declare if other branch is needed.
token	A personal access token from GitHub. Only necessary if the repository being checked is a private repository.
base_url	it's assumed this is coming from github so it is by default 'https://raw.githubusercontent.com/'
dest_dir	A file path where the file should be stored upon arrival to the current repository.

Value

An Rmarkdown or markdown is knitted into the document from another repository

Examples

```
## Not run:

# In an Rmarkdown document:

# For a file in another repository:
# ```{r, echo=FALSE, results='asis'}
borrow_chapter(
  doc_path = "docs/02-chapter_of_course.md",
  repo_name = "jhudsl/OTTR_Template"
)
# ```

# For a local file:
# ```{r, echo=FALSE, results='asis'}
borrow_chapter(doc_path = "02-chapter_of_course.Rmd")
# ```

tag_replacement_list <- list(
  "{A_TAG}" = "replacement here",
  "{TEMPLATE_URL}" = "https://www.ottrproject.org/",
  "{SECOND_TAG}" = "some other replacement here"
)

# For replacing tags
# ```{r, echo=FALSE, results='asis'}
# borrow_chapter(
#  doc_path = "02-chapter_of_course.Rmd",
#  tag_replacement = tag_replacement_list,
# )
# )
# ```

## End(Not run)
```

---

cache_secrets_folder *See where your cached secrets are being stored*

Description

This is a function to retrieve the file path of where your cached secrets are stored

Usage

```
cache_secrets_folder()
```

Value

an file path that shows where your cached secrets are stored

Examples

```
## Not run:

# You can see where your cached secrets are being stored by running:
cached_secrets_folder()

## End(Not run)
```

`check_all_questions` *Check all quiz questions*

Description

Takes output from [ottrpal::parse_quiz] and runs checks on each question in a quiz by calling [ottrpal::check_question] for each question. First splits questions into their own data frame. Returns a list of messages/warnings about each question's set up.

Usage

```
check_all_questions(
  quiz_specs,
  quiz_name = NA,
  verbose = TRUE,
  ignore_coursera = TRUE
)
```

Arguments

`quiz_specs`	quiz_specs which is output from [ottrpal::parse_quiz].
`quiz_name`	The name of the quiz being checked.
`verbose`	Whether progress messages should be given.
`ignore_coursera`	Coursera doesn't like '¡ or ':' in the quizzes. Do not convert quizzes to coursera and ignore ! and : in question prompts that would not be allowed in Leanpub quizzes when converted to a Coursera quiz. Default is to ignore Coursera compatibility.

Value

A list of the output from [ottrpal::check_question] with messages/warnings regarding each question and each check.

Examples

```
## Not run:

# Using good quiz md example

quiz_path <- good_quiz_path()
good_quiz <- readLines(quiz_path)
good_quiz_specs <- parse_quiz(good_quiz)
good_quiz_checks <- check_all_questions(good_quiz_specs)

# Using bad quiz md example

bad_quiz <- readLines(bad_quiz_path())
bad_quiz_specs <- parse_quiz(bad_quiz)
bad_quiz_checks <- check_all_questions(bad_quiz_specs)

## End(Not run)
```

---

check_git_repo *Check if a repository exists on GitHub*

Description

Given a repository name, check with git ls-remote whether the repository exists and return a TRUE/FALSE

Usage

```
check_git_repo(
  repo_name,
  token = NULL,
  silent = TRUE,
  return_repo = FALSE,
  verbose = TRUE
)
```

Arguments

repo_name	the name of the repository, e.g. jhudsl/OTTR_Template
token	A personal access token from GitHub. Only necessary if the repository being checked is a private repository.
silent	TRUE/FALSE of whether the warning from the git ls-remote command should be echoed back if it does fail.
return_repo	TRUE/FALSE of whether or not the output from git ls-remote should be saved to a file (if the repo exists)
verbose	TRUE/FALSE do you want more progress messages?

Value

A TRUE/FALSE whether or not the repository exists. Optionally the output from git ls-remote if return_repo = TRUE.

Examples

```
## Not run:

authorize("github")
check_git_repo("jhudsl/OTTR_Template")

## End(Not run)
```

`check_question` *Check Quiz Question Set Up*

Description

Check quiz question set up to see if it is compliant with Leanpub and Coursera needs. Based off of [Markua guide](https://leanpub.com/markua/read#leanpub-auto-quizzes-and-exercises). Is called by [ottrpal::check_all_questions] and run for each question.

Usage

```
check_question(
  question_df,
  quiz_name = NA,
  verbose = TRUE,
  ignore_coursera = TRUE
)
```

Arguments

`question_df`	Which is an individual question's data frame after being parse from
`quiz_name`	The name of the quiz the question is from
`verbose`	Whether progress messages should be given
`ignore_coursera`	Coursera doesn't like '¡ or ':' in the quizzes. Do not convert quizzes to coursera and ignore ! and : in question prompts that would not be allowed in Leanpub quizzes when converted to a Coursera quiz. Default is to ignore Coursera compatibility

Value

A list of messages/warnings regarding each check for the given question.

Examples

```
## Not run:

# Use readLines to read in a quiz
quiz_path <- good_quiz_path()
quiz_lines <- readLines(quiz_path)

# Use group_split to get the questions
questions_df <- parse_quiz(quiz_lines)$data %>%
  dplyr::group_split(question)

good_quiz_checks <- check_question(questions_df[[2]])

## End(Not run)
```

`check_quiz` *Check Quiz*

Description

For a file path to a quiz, check whether it is properly formatted for Leanpub.

Usage

```
check_quiz(quiz_path, verbose = TRUE, ignore_coursera = TRUE)
```

Arguments

`quiz_path`	A file path to a quiz markdown file
`verbose`	print diagnostic messages? TRUE/FALSE
`ignore_coursera`	Coursera doesn't like '¡ or ':' in the quizzes. Do not convert quizzes to coursera and ignore ! and : in question prompts that would not be allowed in Leanpub quizzes when converted to a Coursera quiz. Default is to ignore Coursera compatibility

Value

A list of checks. "good" means the check passed. Failed checks will report where it failed.

Examples

```
## Not run:

# Take a look at a good quiz's checks:
quiz_path <- good_quiz_path()
good_checks <- check_quiz(quiz_path)
```

```
# Take a look at a failed quiz's checks:
quiz_path <- bad_quiz_path()
failed_checks <- check_quiz(quiz_path)

## End(Not run)
```

---

check_quizzes *Check all quizzes in a directory*

Description

Check the formatting of all quizzes in a given directory.

Usage

```
check_quizzes(
  path = ".",
  quiz_dir = "quizzes",
  write_report = TRUE,
  verbose = TRUE,
  ignore_coursera = TRUE
)
```

Arguments

path	path to the top of course repository (looks for .github folder)
quiz_dir	A path to a directory full of quizzes that should all be checked with [ottrpal::check_all_quizzes].
write_report	TRUE/FALSE save warning report to a CSV file?
verbose	print diagnostic messages
ignore_coursera	Coursera doesn't like '¡ or ':' in the quizzes. Do not convert quizzes to coursera and ignore ! and : in question prompts that would not be allowed in Leanpub quizzes when converted to a Coursera quiz. Default is to ignore Coursera compatibility

Value

A list checks performed on each quiz

Examples

```
## Not run:


## Make a temporary quiz directory
quiz_dir <- dirname(good_quiz_path())
```

```
## Now check the quizzes in that directory
all_quiz_results <- check_quizzes(quiz_dir = quiz_dir)

## End(Not run)
```

`check_quiz_attributes` *Check Quiz Attributes*

Description

Check Quiz Attributes

Usage

```
check_quiz_attributes(quiz_specs, quiz_name = NULL, verbose = TRUE)
```

Arguments

`quiz_specs`	The output from [ottrpal::parse_quiz].
`quiz_name`	A character string indicating the name of the quiz being checked.
`verbose`	Would you like progress messages? TRUE/FALSE

Value

A logical

`check_quiz_dir` *Check all quizzes' formatting for Leanpub*

Description

Check all quizzes' formatting for Leanpub

Usage

```
check_quiz_dir(
  path = ".",
  quiz_dir = "quizzes",
  output_dir = "check_reports",
  resources_dir = "resources",
  report_all = FALSE
)
```

Arguments

path	path to the bookdown or quarto course repository, must have a '.github' folder which will be used to establish the top of the repo.
quiz_dir	A relative file path to the folder (existing or not) that contains the quizzes in Leanpub format. Default is "quizzes".
output_dir	A relative file path to the folder (existing or not) that the output check file should be saved to. Default is "check_reports"
resources_dir	A relative file path to the folder (existing or not) that the ignore_urls.txt file and exclude_files.txt will be found. Default is "resources". If no ignore_urls.txt file and exclude_files.txt files are found, we will download one.
report_all	Should all URLs that were tested be returned? Default is FALSE meaning only broken URLs will be reported in the url_checks.tsv file.

Value

A file will be saved that lists the broken URLs will be saved to the specified output_dir.

Examples

```
## Not run:

rmd_dir <- setup_ottr_template(dir = ".", type = "rmd", render = FALSE)

check_quiz_dir(rmd_dir)

# If there are broken URLs they will be printed in a list at 'question_error_report.tsv'

qmd_dir <- setup_ottr_template(dir = ".", type = "quarto", render = FALSE)

check_quiz_dir(qmd_dir)

## End(Not run)
```

---

check_quiz_question_attributes

Check a question's attributes

Description

This is ran automatically by [ottrpal::check_all_questions] for all questions. It checks that the attributes specified are accepted ones by Leanpub.

Usage

```
check_quiz_question_attributes(question_df, quiz_name = NULL, verbose = TRUE)
```

Arguments

question_df	a data.frame obtained from [ottrpal::parse_quiz_df] and dplyr::group_split(question).
quiz_name	inherited from parse
verbose	print diagnostic messages

Value

Will return a warning for any quiz question attributes used that are not supported.

---

check_spelling *Check spelling of all md, rmd, and qmd files*

Description

Check spelling of all md, rmd, and qmd files

Usage

```
check_spelling(
  path = ".",
  output_dir = "check_reports",
  resources_dir = "resources",
  file_pattern = "md$"
)
```

Arguments

path	path to the bookdown or quarto course repository, must have a '.github' folder which will be used to establish the top of the repo.
output_dir	A relative file path to the folder (existing or not) that the output check file should be saved to. Default is "check_reports"
resources_dir	A relative file path to the folder (existing or not) that the dictionary.txt file and exclude_files.txt will be found. Default is "resources". If no dictionary.txt file and exclude_files.txt files are found, we will download one.
file_pattern	A file pattern should we be looking for for the files whose spelling should be tested. Default is "md$". Regex interpreted.

Value

A file will be saved that lists the broken URLs will be saved to the specified output_dir.

Examples

```
## Not run:

rmd_dir <- setup_ottr_template(dir = ".", type = "rmd", render = FALSE)

check_spelling(rmd_dir)

# If there are broken URLs they will be printed in a list at "check_reports/url_checks.tsv"

qmd_dir <- setup_ottr_template(dir = ".", type = "quarto", render = FALSE)

check_spelling(qmd_dir)

## End(Not run)
```

`check_urls` *Check URLs of all md,rmd,and qmd files*

Description

Check URLs of all md,rmd,and qmd files

Usage

```
check_urls(
  path = ".",
  output_dir = "check_reports",
  resources_dir = "resources",
  report_all = FALSE
)
```

Arguments

`path`	path to the bookdown or quarto course repository, must have a '.github' folder which will be used to establish the top of the repo.
`output_dir`	A relative file path to the folder (existing or not) that the output check file should be saved to. Default is "check_reports"
`resources_dir`	A relative file path to the folder (existing or not) that the ignore_urls.txt file and exclude_files.txt will be found. Default is "resources". If no ignore_urls.txt file and exclude_files.txt files are found, we will download one.
`report_all`	Should all URLs that were tested be returned? Default is FALSE meaning only broken URLs will be reported in the url_checks.tsv file.

Value

A file will be saved that lists the broken URLs will be saved to the specified output_dir.

Examples

```
## Not run:

rmd_dir <- setup_ottr_template(dir = ".", type = "rmd", render = FALSE)

check_urls(rmd_dir)

# If there are broken URLs they will be printed in a list at "check_reports/url_checks.tsv"

qmd_dir <- setup_ottr_template(dir = ".", type = "quarto", render = FALSE)

check_urls(qmd_dir)

## End(Not run)
```

---

clean_up *Clean up OTTR_Template files used for testing*

Description

Clean up OTTR_Template files used for testing

Usage

```
clean_up()
```

Value

Looks for dangling zips and directories downloaded for testing and removes them

---

convert_coursera_quizzes *Convert Leanpub md quiz to Coursera yaml quiz*

Description

Convert Leanpub md quiz to Coursera yaml quiz

Usage

```
convert_coursera_quizzes(
  input_quiz_dir = "quizzes",
  output_quiz_dir = "coursera_quizzes",
  verbose = TRUE
)
```

Arguments

input_quiz_dir	A path to a directory of leanpub formatted quiz md files. By default assumes "quizzes" and looks in current directory.
output_quiz_dir	A folder (existing or not) that the new coursera converted quizzes should be saved to. By default saves to "coursera_quizzes".
verbose	Would you like the progress messages: TRUE/FALSE?

Value

A folder of coursera ready quiz files saved to the output directory specified as a yamls.

Examples

```
# Set up a directory with a quiz in it for this example
tdir <- tempfile()
dir.create(tdir, showWarnings = FALSE, recursive = TRUE)

file.copy(
  from = good_quiz_path(),
  to = file.path(tdir, basename(good_quiz_path()))
)

# Provide path to directory of quizzes
convert_coursera_quizzes(tdir)

system("rm -r coursera_quizzes")
```

---

convert_quiz *Convert Leanpub md quiz to Coursera yaml quiz*

Description

Convert a Leanpub-formatted md quiz file to a Coursera-formatted yaml quiz file in preparation for uploading to Coursera.

Usage

```
convert_quiz(quiz_path, output_quiz_dir = dirname(quiz_path), verbose = TRUE)
```

Arguments

quiz_path	A path to a quiz .md file to be converted.
output_quiz_dir	An existing folder where you would like the new version of the quiz to be saved. Default is the directory of the quiz_path provided
verbose	Would you like the progress messages?

Value

A Coursera-ready quiz file saved to the output directory specified as a yaml.

Examples

```
## Not run:

quiz_path <- good_quiz_path()

# Provide path to quiz to convert
convert_quiz(quiz_path)

## End(Not run)
```

`course_path` *Find root of OTTR course provided*

Description

Find root of OTTR course provided

Usage

```
course_path(path = ".")
```

Arguments

`path`	Where should we be looking for a OTTR course. By default will look in current working directory.

Value

Absolute file path to the course pointed to

Returns a absolute file path to where the course is

`course_to_book_txt` *Create Book.txt file from files existing in quiz directory*

Description

Create Book.txt file from files existing in quiz directory

Usage

```
course_to_book_txt(
  path = ".",
  md_files = NULL,
  output_dir = "manuscript",
  quiz_dir = "quizzes",
  verbose = TRUE
)
```

Arguments

path	path to the bookdown or quarto course repository, must have a '_bookdown.yml' or '_quarto.yml' file
md_files	vector of file path of the md's to be included
output_dir	output directory to put files. It should likely be relative to path
quiz_dir	Where are the quizzes stored? Default looks for folder called "quizzes".
verbose	print diagnostic messages

Value

A list of quiz and chapter files in order in a file called Book.txt – How Leanpub wants it.

---

delete_creds *Delete cached ottrpal credentials*

Description

This is a function to delete cached creds and creds in the current environment that were set by ottrpal

Usage

```
delete_creds(app_name = "all")
```

Arguments

app_name	which app would you like to delete the creds for? Default is to delete the creds for all.

Value

Cached credentials are deleted and report is given back

Examples

```
## Not run:

delete_creds("google")

## End(Not run)
```

`encrypt_creds_path` *Get file path to an encrypted credentials RDS*

Description

Get file path to an encrypted credentials RDS

Usage

```
encrypt_creds_path()
```

`encrypt_creds_user_path` *Get file path to an default credentials RDS*

Description

Get file path to an default credentials RDS

Usage

```
encrypt_creds_user_path()
```

`extract_meta` *Extract meta fields from a tag*

Description

Extract meta fields from a tag

Usage

```
extract_meta(tags)
```

Arguments

`tags`	A single tag or vector of tags to extract the fields from.

Value

A named vector indicating the field and entry associated with it.

Examples

```
### Simple example
tag <- "{quiz, id: quiz_name_here, attempts: 10}"

# Extract metadata tags
meta <- extract_meta(tag)

### Example using a file
quiz_path <- good_quiz_path()
quiz_lines <- readLines(quiz_path)

# Put this in a data.frame so we can identify the content
quiz_df <- parse_quiz_df(quiz_lines)

# Extract the tags
tags <- quiz_df %>%
  dplyr::filter(type == "tag") %>%
  dplyr::pull("original")

# Extract metadata tags
meta <- extract_meta(tags)
```

---

extract_object_id *Extract Object IDs using Google Slides API*

Description

Performs a HTTP GET method to request the IDs of every slide in a Google Slides presentation. The ID of the first slide is always 'p'.

Usage

```
extract_object_id(
  slide_url,
  token = NULL,
  access_token = NULL,
  refresh_token = NULL
)
```

Arguments

slide_url	URL whose 'General access' is set to 'Anyone with the link'
token	OAuth 2.0 Access Token. If you don't have a token, use [authorize()] to obtain an access token from Google's OAuth 2.0 server.

access_token	Access token can be obtained from running authorize() interactively (token <- authorize(); token$credentials$access_token). This allows it to be passed in using two secrets.
refresh_token	Refresh token can be obtained from running authorize() interactively (token <- authorize(); token$credentials$refresh_token). This allows it to be passed in using two secrets.

Value

Character vector of object ID(s)

Examples

```
## Not run:
# First, obtain access token and store token for extract_object_id() to use
authorize(client_id = "MY_CLIENT_ID", client_secret = "MY_CLIENT_SECRET")
# Use stored token to talk to Google Slides API
extract_object_id(slide_url = "https://docs.google.com/presentation/d/1H5aF_ROKVxE-H
                               FHhoOy9vU2Y-y2M_PiV0q-JBL17Gss/edit?usp=sharing")

## End(Not run)
```

---

find_issue *Find an issue on GitHub with a particular title*

Description

Given text and repository name, find if an issue exists.

Usage

```
find_issue(text, repo_name, token = NULL)
```

Arguments

text	What text to be searched for in the GitHub issues. Can be regex.
repo_name	the name of the repository, e.g. jhudsl/OTTR_Template
token	A personal access token from GitHub. Only necessary if the repository being checked is a private repository.

Value

A TRUE/FALSE whether or not the issue with this text on this repository exists.

Examples

```
## Not run:

authorize("github")

find_issue(text = "TEST", repo_name = "jhudsl/ottrpal")

## End(Not run)
```

`get_chapters` *Make Leanpub file that has embed webpage of a chapter*

Description

Make Leanpub file that has embed webpage of a chapter

Usage

```
get_chapters(
  path = ".",
  html_page = file.path("docs", "index.html"),
  base_url = "."
)
```

Arguments

`path`	path to the bookdown or quarto course repository, must have a '_bookdown.yml' or '_quarto.yml' file
`html_page`	The file path of the rendered index.html file. It can be a url
`base_url`	The base url of where the chapters are published – the url to provide to the iframe in Leanpub e.g. https://jhudatascience.org/OTTR_Template/coursera

Value

A data.frame of the chapter urls and their titles that are to be ported to Leanpub. This can be passed to

get_github *Handler function for GET requests from GitHub*

Description

This is a function to get the GitHub user's info

Usage

```
get_github(token = NULL, url)
```

Arguments

token	You can provide the Personal Access Token key directly or this function will attempt to grab a PAT that was stored using the 'authorize("github")' function
url	What is the URL endpoint we are attempting to grab here?

Value

Information regarding a Github account

get_gs_pptx *Download Google Slides pptx file*

Description

Download Google Slides pptx file

Usage

```
get_gs_pptx(id)
```

Arguments

id	Identifier of Google slides presentation, passed to get_slide_id

Value

Downloaded file (in temporary directory)

Note

This downloads presentations if they are public and also try to make sure it does not fail on large files

get_object_id_notes *Retrieve Speaker Notes and their corresponding Object (Slide) IDs from a Google Slides presentation*

Description

Google Slides API calls a presentation slide ID as an 'object ID'.

Usage

```
get_object_id_notes(slide_url)
```

Arguments

slide_url	URL whose 'General access' is set to 'Anyone with the link'

Value

Data frame of Object IDs and Speaker notes.

Examples

```
## Not run:
get_object_id_notes("https://docs.google.com/presentation/d/
                     1H5aF_ROKVxE-HFHhoOy9vU2Y-y2M_PiV0q-JBL17Gss/edit?usp=sharing")

## End(Not run)
```

get_pages_url *Retrieve pages url for a repo*

Description

Given an repository on GitHub, retrieve the pages URL for it.

Usage

```
get_pages_url(repo_name, token = NULL, verbose = FALSE, keep_json = FALSE)
```

Arguments

repo_name	The full name of the repo to get chapters from. e.g. 'jhudsl/OTTR_Template'
token	If private repositories are to be retrieved, a github personal access token needs to be supplied. Run 'authorize("github")' to set this.
verbose	TRUE/FALSE do you want more progress messages?
keep_json	verbose TRUE/FALSE keep the json file locally?

Value

a data frame with the repository with the following columns: data_level, data_path, chapt_name, url, repository name

Examples

```
## Not run:

usethis::create_github_token()

get_pages_url("jhudsl/Documentation_and_Usability")

## End(Not run)
```

`get_repo_info` *Retrieve information about a github repo*

Description

Given an repository on GitHub, retrieve the information about it from the GitHub API and read it into R.

Usage

```
get_repo_info(repo_name, token = NULL, verbose = FALSE)
```

Arguments

`repo_name`	The full name of the repo to get bookdown chapters from. e.g. "jhudsl/OTTR_Template"
`token`	If private repositories are to be retrieved, a github personal access token needs to be supplied. If none is supplied, then this will attempt to grab from a git pat set in the environment with usethis::create_github_token().
`verbose`	TRUE/FALSE do you want more progress messages?

Value

a data frame with the repository with the following columns: data_level, data_path, chapt_name, url, repository name

Examples

```
## Not run:

repo_info <- get_repo_info("jhudsl/Documentation_and_Usability")

## End(Not run)
```

get_slide_id *Get Slide ID from URL*

Description

Get Slide ID from URL

Usage

```
get_slide_id(x)
```

Arguments

x	URL of slide

Value

A character vector

Examples

```
x <- paste0(
  "https://docs.google.com/presentation/d/",
  "1Tg-GTGnUPduOtZKYuMoelqUNZnUp3vvg_7TtpUPL7e8",
  "/edit#slide=id.g154aa4fae2_0_58"
)
get_slide_id(x)
```

get_urls *Identify and collect URLs in a single md,rmd, or qmd file*

Description

Identify and collect URLs in a single md,rmd, or qmd file

Usage

```
get_urls(file, ignore_urls = "")
```

Arguments

file	A file path to a md,rmd, or qmd file that contains URLs to be check
ignore_urls	A vector of URLs which to ignore.

Value

a data.frame of all the URLs identified in the given md,rmd, or qmd file

Examples

```
## Not run:

# Add in a URL error
# writeLines("A URL error: https://notawebsiteaaaaaaa.com", "url_test_error.md")

get_urls("url_test_error.md")

## End(Not run)
```

`good_quiz_path` *Path to good example quiz*

Description

Path to good example quiz

Usage

```
good_quiz_path()
```

Value

The file path to an example good quiz included in the package that should pass the quiz checks.

`gs_id_from_slide` *Google Slides Helper Functions*

Description

Google Slides Helper Functions

Usage

```
gs_id_from_slide(file)

get_image_link_from_slide(file)

get_image_from_slide(file)
```

Arguments

`file`	markdown file for manuscript

Value

A scalar character vector

gs_png_url *Get Google Slide PNG URL*

Description

Get Google Slide PNG URL

Usage

```
gs_png_url(url)

get_slide_page(url)

gs_png_download(url, output_dir = ".", overwrite = TRUE)

include_slide(
  url,
  output_dir = knitr::opts_chunk$get("fig.path"),
  overwrite = TRUE,
  ...
)
```

Arguments

url	URL to Google Slide
output_dir	path to output png
overwrite	should the slide PNG be overwritten?
...	for include_slide, options passed to [knitr::include_graphics()]

Value

A character vector of URLs

Examples

```
url <- paste0(
  "https://docs.google.com/presentation/d/",
  "12DPZgPteQBwgal6kSPP58zhPhjZ7QSPZLe3NkA8M3eo/edit",
  "#slide=id.gc8648f14c3_0_397&t=4"
)
id <- get_slide_id(url)
gs_png_url(url)
```

key_encrypt_creds_path *Get file path to an key encryption RDS*

Description

Get file path to an key encryption RDS

Usage

```
key_encrypt_creds_path()
```

make_embed_markdown *Make Leanpub file that has embed webpage of a chapter*

Description

Make Leanpub file that has embed webpage of a chapter

Usage

```
make_embed_markdown(
  path = ".",
  url,
  chapt_title,
  img_path,
  output_dir = "manuscript",
  footer_text = "",
  width_spec = 800,
  height_spec = 600,
  verbose = TRUE
)
```

Arguments

path	path to the bookdown or quarto course repository, must have a '_bookdown.yml' or '_quarto.yml' file
url	The url to the chapter that is to be embed
chapt_title	Title of chapter to be used as file name and printed on iframe
img_path	File path to image to use for poster
output_dir	output directory to put files. It should likely be relative to path
footer_text	Optionally can add a bit of text that will be added to the end of each file before the references section.
width_spec	How wide should the iframe be in pixels?
height_spec	How high should the iframe be in pixels?
verbose	print diagnostic messages

Value

A markdown file with an iframe of the provided chapter

`make_screenshots` *A function to make screenshots from an OTTR bookdown website*

Description

This function creates screenshots of course chapters that are stored in a created output directory

Usage

```
make_screenshots(
  path = ".",
  token,
  repo,
  output_dir = file.path(path, "resources", "chapt_screen_images"),
  base_url = NULL
)
```

Arguments

`path`	default is to look for OTTR files in current directory based on existence of .github. But if you'd like to run this in a different path, you can point to that file path.
`token`	required argument; a Git secret – see https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens for more info
`repo`	required argument; GitHub repository name, e.g., jhudsl/OTTR_Template
`output_dir`	default is "resources/chapt_screen_images"; Output directory where the chapter's screen images should be stored. For OTTR courses, don't change this unless you've changed the downstream functions accordingly.
`base_url`	default is NULL; rendered bookdown URL where screenshots are taken from, if NULL, the function will use the repo_name and and token to find the base_url

Value

the file path for file where chapter urls are saved

Author(s)

Candace Savonen

Examples

```
## Not run:

make_screenshots(
  token = Sys.getenv("secrets.GH_PAT"),
  repo = "jhudsl/OTTR_Template"
)

## End(Not run)
```

`ottrfy` *OTTRfy your repository*

Description

This script downloads all the files and sets up the folders you need to OTTR-fy a repository that has markdown or R Markdown files

Usage

```
ottrfy(path = ".", type = "rmd", overwrite = FALSE)
```

Arguments

`path`	What's the file path we are making an OTTR. Needs to be a directory
`type`	Can be "rmd", "quarto" "rmd_web", or "quarto_web" depending on what kind of OTTR site you'd like to make
`overwrite`	TRUE or FALSE existing files should be overwritten?

Value

Information regarding a Github account

Examples

```
## Not run:

ottrfy(type = "rmd")

ottrfy(type = "quarto")

ottrfy(type = "rmd_web")

ottrfy(type = "quarto_web")


## End(Not run)
```

ottr_check *Ottrpal Checks*

Description

Ottrpal Checks

Usage

```
ottr_check(
  check_type,
  path = ".",
  output_dir = "check_reports",
  resources_dir = "resources",
  ...
)
```

Arguments

check_type	What check would you like to run "spelling", "urls" or "quiz_format".
path	path to the bookdown or quarto course repository, must have a ‘.github‘ folder which will be used to establish the top of the repo.
output_dir	A relative file path to the folder (existing or not) that the output check file should be saved to. Default is "check_reports"
resources_dir	A relative file path to the folder (existing or not) that the dictionary.txt file and exclude_files.txt will be found. Default is "resources". If no dictionary.txt file and exclude_files.txt files are found, we will download one.
...	All additional arguments passed to the respective check

Value

The result of the check being called

parse_quiz *Parse Quiz and Other Checking Functions*

Description

Parse Quiz and Other Checking Functions

Extract lines of the quiz

Usage

```
parse_quiz(quiz_lines, quiz_name = NULL, verbose = FALSE)

extract_quiz(quiz_lines)
```

Arguments

quiz_lines	A quiz's contents read in with readLines()
quiz_name	A character vector indicating the name of the quiz.
verbose	Would you like progress messages? TRUE/FALSE

Value

A list of elements, including a 'data.frame' and metadata for questions

the lines of the quiz that actually contain of the content of the quiz.

Examples

```
quiz_lines <- c(
  "{quiz, id: quiz_00_filename}",
  "### Lesson Name quiz",
  "{choose-answers: 4}",
  "? What do you think?",
  "",
  "C) The answer to this one",
  "o) Not the answer",
  "o) Not the answer either",
  "C) Another correct answer",
  "m) Mandatory different answer",
  "",
  "{/quiz}"
)
quiz_specs <- parse_quiz(quiz_lines)
check_quiz_attributes(quiz_specs)
```

---

parse_quiz_df *Parse quiz into a data.frame*

Description

Parse quiz into a data.frame

Usage

```
parse_quiz_df(quiz_lines, remove_tags = FALSE)
```

Arguments

quiz_lines	A character vector of the contents of the markdown file obtained from readLines()
remove_tags	TRUE/FALSE remove tags and empty lines?

Value

A data frame containing a type column which indicates what type of line each is.

Examples

```
## Not run:

# Use readLines() to read in a quiz
quiz_path <- good_quiz_path()
quiz_lines <- readLines(quiz_path)

# Can use this to parse the quiz into a data.frame
quiz_df <- parse_quiz_df(quiz_lines)

## End(Not run)
```

---

parse_q_tag *Parse apart a tag*

Description

Parse apart a tag

Usage

```
parse_q_tag(tag)
```

Arguments

tag	A single tag to extract from

Value

A named vector indicating the field and entry associated with it.

Examples

```
tag <- "{quiz, id: quiz_name_here, attempts: 10}"
parse_q_tag(tag)
```

pptx_notes *Get Notes from a PowerPoint (usually from Google Slides)*

Description

Get Notes from a PowerPoint (usually from Google Slides)

Usage

```
pptx_notes(file, ...)

pptx_slide_text_df(file, ...)

pptx_slide_note_df(file, ...)

unzip_pptx(file)
```

Arguments

file	Character. Path for 'PPTX' file
...	additional arguments to pass to xml_notes, particularly xpath

Value

Either a character vector or 'NULL'

Examples

```
## Not run:
pptx_notes(ex_file)
pptx_slide_note_df(ex_file)
pptx_slide_text_df(ex_file)

## End(Not run)
```

qrmd_files *Get file paths to all qmds or rmds in the course website directory*

Description

Get file paths to all qmds or rmds in the course website directory

Usage

```
qrmd_files(path = ".")
```

Arguments

path	Where to look for the _bookdown.yml or _quarto.yml file. Passes to get_yaml_spec() function. By default looks in current directory

Value

The file paths to rmds or wmds listed in the _bookdown.yml or _quarto.yml file.

---

render_without_toc *Create TOC-less course website for use in Coursera or Leanpub*

Description

Create a version of the course that does not have a TOC and has quizzes in the Coursera yaml format. This is only needed to be used on Bookdown courses. Quarto has a simple command for this.

Usage

```
render_without_toc(
  path = ".",
  output_dir = file.path("docs", "no_toc"),
  output_yaml = "_output.yml",
  convert_quizzes = FALSE,
  input_quiz_dir = "quizzes",
  output_quiz_dir = "coursera_quizzes",
  verbose = TRUE
)
```

Arguments

path	path to the bookdown or quarto course repository, must have a '_bookdown.yml' or '_quarto.yml' file
output_dir	A folder (existing or not) that the TOC-less Bookdown for Coursera files should be saved. By default is file.path("docs", "coursera")
output_yaml	A output.yml file to be provided to bookdown. By default is "_output.yml"
convert_quizzes	TRUE/FALSE whether or not to convert quizzes. Default is TRUE
input_quiz_dir	A path to a directory of Leanpub-formatted quiz md files. By default assumes "quizzes" and looks in current directory.
output_quiz_dir	A folder (existing or not) where the coursera quizzes should be saved. By default is "coursera_quizzes".
verbose	Would you like the progress messages? TRUE/FALSE

Value

A folder of coursera ready quiz files and html chapter files saved to output directories specified.

---

setup_ottr_template *Download and render files from main OTTR_Template to test*

Description

Download and render files from main OTTR_Template to test

Usage

```
setup_ottr_template(dir = ".", type, render = TRUE)
```

Arguments

dir	What relative file path should the files be downloaded
type	Which OTTR repo are we downloading? Options are "rmd", "quarto", "rmd_website", "quarto_website"
render	Should the OTTR repo be rendered after downloading? Default is TRUE

Value

This downloads the main branch repo files from the respective repo for testing purposes

---

set_knitr_image_path *set_knitr_image_path is being deprecated*

Description

set_knitr_image_path is being deprecated

Usage

```
set_knitr_image_path()
```

Value

Message that says to no longer use this function

supported_endpoints *Supported endpoints*

Description

This is function stores endpoints and supported app names

Usage

```
supported_endpoints()
```

test_url *Test a URL*

Description

Test a URL

Usage

```
test_url(url, ignore_urls = "")
```

Arguments

url	A single URL that will be checked whether it is real.
ignore_urls	A vector of URLs which to ignore.

Value

a logical TRUE/FALSE for whether the URL is legitimate.

Examples

```
## Not run:

# This should print out "failed"
test_url("https://notawebsiteaaaaaaa.com")


# This should print out "success"
test_url("https://github.com")

## End(Not run)
```

website_to_embed_leanpub *Convert Website Course to Leanpub*

Description

Convert Website Course to Leanpub

Usage

```
website_to_embed_leanpub(
  path = ".",
  chapt_img_key = NULL,
  render = NULL,
  html_page = file.path(base_url, "index.html"),
  base_url = NULL,
  clean_up = TRUE,
  default_img = NULL,
  output_dir = "manuscript",
  make_book_txt = FALSE,
  quiz_dir = "quizzes",
  run_quiz_checks = FALSE,
  remove_resources_start = FALSE,
  verbose = TRUE,
  footer_text = ""
)
```

Arguments

path	path to the bookdown or quarto course repository, must have a '_bookdown.yml' or '_quarto.yml' file
chapt_img_key	File path to a TSV whose contents are the chapter urls ('url'), the chapter titles ('chapt_title'), the file path to the image to be used for the chapter ('img_path'). Column names 'url', 'chapt_title', and 'img_path' must be used. If no chapter title column supplied, the basename of the url will be used, If no image column supplied, default image used.
render	if 'TRUE', then [bookdown::render_book()] will be run on each Rmd.
html_page	The file path of the rendered index.html file
base_url	The base url of where the chapters are published – the url to provide to the iframe in Leanpub e.g. https://jhudatascience.org/OTTR_Template/coursera
clean_up	Should the previous docs and manuscript folder be cleaned up?
default_img	A google slide link to the default image to be used for all chapters
output_dir	output directory to put files. It should likely be relative to path
make_book_txt	Should [ottrpal::course_to_book_txt()] be run to create a 'Book.txt' in the output directory?

quiz_dir	directory that contains the quiz .md files that should be checked and incorporated into the Book.txt file. If you don't have quizzes, set this to NULL
run_quiz_checks	TRUE/FALSE run quiz checks
remove_resources_start	remove the word 'resources/' at the front of any image path.
verbose	print diagnostic messages
footer_text	Optionally can add a bit of text that will be added to the end of each file before the references section.

Value

A directory of output files in a folder 'manuscript' for publishing on Leanpub.

Examples

```
## Not run:

ottrpal::website_to_embed_leanpub(
  base_url = "https://jhudatascience.org/OTTR_Template/",
  make_book_txt = TRUE,
  quiz_dir = NULL
)

## End(Not run)
```

---

xml_notes *Get Notes from XML*

Description

Get Notes from XML

Usage

```
xml_notes(file, collapse_text = TRUE, xpath = "//a:r//a:t")
```

Arguments

file	XML file from a PPTX
collapse_text	should text be collapsed by spaces?
xpath	xpath to pass to [xml2::xml_find_all()]

Value

A character vector

Index