Package 'onbrand'

September 2, 2021

Type Package

Title Templated Reporting Workflows in Word and PowerPoint

Version 1.0.1

Maintainer John Harrold <john.m.harrold@gmail.com>

Description Automated reporting in Word and PowerPoint can require customization for each organizational template. This package works around this by adding standard reporting functions and an abstraction layer to facilitate automated reporting workflows that can be replicated across different organizational templates.

URL https://onbrand.ubiquity.tools/

BugReports https://github.com/john-harrold/onbrand/issues

License BSD_2_clause + file LICENSE

Encoding UTF-8

LazyData FALSE

RoxygenNote 7.1.1

VignetteBuilder knitr

Imports dplyr, flextable, ggplot2, magrittr, officer (>= 0.3.7), stringr, rlang, yaml

Suggests knitr, knitrdata, markdown, rmarkdown, testthat

NeedsCompilation no

Author John Harrold [aut, cre] (<https://orcid.org/0000-0003-2052-4373>), Bryan Smith [aut]

Repository CRAN

Date/Publication 2021-09-02 05:10:02 UTC

R topics documented:

`add_pptx_ph_content` *Populate Placeholder In Officer Report*

Description

Places content in a PowerPoint placeholder for a given officer document.

Usage

```
add_pptx_ph_content(obnd, content_type, content, ph_label, verbose = TRUE)
```

Arguments

`obnd`	onbrand report object
`content_type`	string indicating the content type
`content`	content
`ph_label`	placeholder location (text)
`verbose`	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Details

For each content type listed below the following content is expected:

- `"text"` text string of information
- `"list"` vector of paired values (indent level and text), eg. c(1, "Main Bullet", 2 "Sub Bullet")
- `"imagefile"` string containing path to image file
- `"ggplot"` ggplot object, eg. p = ggplot() + ...

- "table" list containing the table content and other options with the following elements (defaults in parenthesis):
  - table Data frame containing the tabular data
  - header Boolean variable to control displaying the header (TRUE)
  - first_row Boolean variable to indicate that the first row contains header information (TRUE)
- "flextable" list containing flextable content and other options with the following elements (defaults in parenthesis):
  - table Data frame containing the tabular data
  - header_top, header_middle, header_bottom (NULL) a list with the same names as the data frame names containing the tabular data and values with the header text to show in the table
  - header_format string containing the format, either "text", or "md" (default NULL assumes "text" format)
  - merge_header (TRUE) Set to true to combine column headers with the same information
  - table_body_alignment, table_header_alignment ("center") Controls alignment
  - table_autofit (TRUE) Automatically fit content, or specify the cell width and height with cwidth (0.75) and cheight (0.25)
  - table_theme ("theme_vanilla") Table theme
- "flextable_object" user defined flextable object

Value

officer pptx object with the content added

See Also

view_layout

---

fetch_md_def *Fetch Markdown Default Format from onbrand Object*

Description

Used to extract the formatting elements for a given style from an onbrand object.

Usage

```
fetch_md_def(obnd, style = "default", verbose = TRUE)
```

Arguments

obnd	onbrand report object
style	name of style in md_def for the report type in obnd to fetch ("default")
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Value

list with the following elements

- isgood: Boolean variable indicating success or failure
- md_def: List with the default format for the specified style
- msgs: Vector of messages

Examples

```
obnd = read_template(
   template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
   mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))
obnd = fetch_md_def(obnd, style="default")
md_def =  obnd[["md_def"]]
```

---

fetch_officer_object *Extracts Officer Object From Onbrand Report Object*

Description

If you need modify the onbrand report object directly with officer functions you can use this function to extract the report object from the onbrand object.

Usage

```
fetch_officer_object(obnd, verbose = TRUE)
```

Arguments

obnd	onbrand report object
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Value

List with the following elements

- isgood: Boolean variable indicating success or failure
- rpt: Officer object
- msgs: Vector of messages

See Also

set_officer_object

Examples

```
obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

rpt = fetch_officer_object(obnd)$rpt
```

`fetch_report_format` *Fetch The Specified Report Formatting Information*

Description

Returns a list of the default font format for the report element

Usage

```
fetch_report_format(obnd, format_name = "default", verbose = TRUE)
```

Arguments

obnd	onbrand report object
format_name	Name of report format to fetch; this is defined in the md_def
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned list. section for the given report type ("default")

Value

list containing the following elements

- isgood: Boolean variable indicating success or failure
- msgs: Vector of messages
- format_details: List containing the format details for the specified format_name

Examples

```
obnd = read_template(
       template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
        mapping = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

fr = fetch_report_format(obnd)
```

fetch_rpttype *Determines Type of Report Template*

Description

Based on the file extension for a template

Usage

```
fetch_rpttype(template = NULL, verbose = TRUE)
```

Arguments

template	Name of PowerPoint or Word file
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned list.

Value

List with the following elements

- rpttype: Either Word, PowerPoint or Unknown
- rptext: Either docx, pptx, or Unknown
- rptobj: Either rdocx, rpptx, or Unknown
- isgood: Boolean variable indicating success or failure
- msgs: Vector of messages

Examples

```
rpttype = fetch_rpttype(template=
  file.path(system.file(package="onbrand"), "templates", "report.pptx"))
```

fph *Fetch PowerPoint Placeholder*

Description

Retrieves the placeholder name in PowerPoint for a specified layout element.

Usage

```
fph(obnd, template = NULL, pn = NULL, verbose = TRUE)
```

Arguments

obnd	onbrand report object
template	Name of slide template (name from templates in yaml mapping file)
pn	Placehodler name to fetch
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned list.

Value

List with the following elements

- ph: Placeholder label or NULL if failure
- type: Placeholder content type in PowerPoint or NULL if failure
- isgood: Boolean variable indicating success or failure
- msgs: Vector of messages

Examples

```
# Creating an onbrand object:
obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

# Pulling out the placeholder information:
ph = fph(obnd, "two_content_header_text", "content_left_header")
```

---

fst *Fetch Word Style*

Description

Retrieves the style name in Word for a specified onbrand style name.

Usage

```
fst(obnd, osn = NULL, verbose = TRUE)
```

Arguments

obnd	onbrand report object
osn	onbrand Word style name to fetch
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned list.

Value

List with the following elements

- wsn: Word style name that corresponds to the specified onbrand style name (osn)
- dff: Default font format for that style (the corresponding md_def section of the yaml file for that style)
- isgood: Boolean variable indicating success or failure
- msgs: Vector of messages

Examples

```
# Creating an onbrand object:
obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

# Pulling out the placeholder information:
st = fst(obnd, "Heading_3")
```

---

md_to_officer *Parse Markdown for Officer*

Description

Parses text in Markdown format and returns fpar and as_paragraph command strings to be used with officer

Usage

```
md_to_officer(
  str,
 default_format = list(color = "black", font.size = 12, bold = FALSE, italic = FALSE,
  underlined = FALSE, font.family = "Cambria (Body)", vertical.align = "baseline",
    shading.color = "transparent")
)
```

Arguments

str	string containing Markdown can contain the following elements: • paragraph: two or more new lines creates a paragraph • bold: can be either "**text in bold**" or "__text in bold__" • italics: can be either "*text in italics*" or "_text in italics_" • subscript: "Normal~subscript~" • superscript: "Normal^superscript^"

- color: "<color:red>red text</color>"
- shade: "<shade:#33ff33>shading</shade>"
- font family: "<ff:symbol>symbol</ff>"

default_format list containing the default format for elements not defined with markdown default values.

```
default_format = list(
   color          = "black",
   font.size      = 12,
   bold           = FALSE,
   italic         = FALSE,
   underlined     = FALSE,
   font.family    = "Cambria (Body)",
   vertical.align = "baseline",
   shading.color  = "transparent")
```

Value

list with parsed paragraph elements ubiquity system object with the content added to the body, each paragraph can be found in a numbered list element (e.g. pgraph_1, pgraph_2, etc) each with the following elements:

- locs: Dataframe showing the locations of markdown elements in the current paragraph
- pele: These are the individual parsed paragraph elements
- ftext_cmd: String containing the ftext commands.
- fpar_cmd: String containing the fpar commands that can be run using eval to return the output of fpar. For example:

```
myfpar = eval(parse(text=pgparse$pgraph_1$fpar_cmd))
```

- as_paragraph_cmd: String containing the as_paragraph_cmd that can be run using

```
myas_para = eval(parse(text=pgparse$pgraph_1$as_paragraph_cmd))
```

Examples

```
res             = md_to_officer("Be **bold**!")
fpar_obj        = eval(parse(text=res$pgraph_1$fpar_cmd))
as_paragraph_obj = eval(parse(text=res$pgraph_1$as_paragraph_cmd))
```

md_to_oo *Parse Markdown into Officer as_paragraph Result*

Description

Used to take small markdown chunks and return the as_paragraph results. This function will take the markdown specified in str, calls md_to_officer, evals the as_paragraph field from the first paragraph returned, evals that result and returns the object from the as_paragraph command.

Usage

```
md_to_oo(strs, default_format = NULL)
```

Arguments

strs	vector of strings containing Markdown can contain the following elements:
default_format	list containing the default format for elements not defined with markdown default values (format the same as md_to_officer, default is NULL)

Value

list with the following elements

- isgood: Boolean value indicating the result of the function call
- msgs: sequence of strings contianing a description of any problems
- as_par_cmd:as_paragraph generated code from md_to_officer
- oo: as_paragraph officer object resulting from running the as_par_cmd code

Examples

```
res = md_to_oo("Be **bold**")
```

onbrand *onbrand: officer Abstraction Layer for Organizational Templates*

Description

The onbrand package creates an abstraction layer that is easily configurable with a yaml file to allow for creation of reproducible reporting work flows across Word and PowerPoint templates.

Author(s)

Maintainer: John Harrold <john.m.harrold@gmail.com> (ORCID)

Authors:

- Bryan Smith <r.bryan.smith@gmail.com >

See Also

https://github.com/john-harrold/onbrand

`preview_template` *Generate Report Previewing the Locations From Mapping File*

Description

Takes an onbrand object with a loaded template and populates the template with the elements from the mapping file.

Usage

```
preview_template(obnd, verbose = TRUE)
```

Arguments

obnd	onbrand report object
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Value

onbrand object with template previews added and any messages passed along

Examples

```
obnd = read_template(
   template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
   mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))
obnd = preview_template(obnd)

obnd = read_template(
   template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
   mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))
obnd = preview_template(obnd)
```

`read_template` *Read Word or PowerPoint Templates*

Description

Takes a given template file/yaml mapping file combination, reads in that information, checks to make sure the mapping information is correct and then returns an onbrand object.

Usage

```
read_template(
 template = file.path(system.file(package = "onbrand"), "templates", "report.pptx"),
 mapping = file.path(system.file(package = "onbrand"), "templates", "report.yaml"),
  verbose = TRUE
)
```

Arguments

`template`	Name of PowerPoint or Word file to annotate (defaults to included PowerPoint template)
`mapping`	Name of yaml file with configuration information
`verbose`	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Value

onbrand object which is a list with the following elements:

- isgood: Boolean variable indicating the current state of the object
- rpt: Officer object containing the initialized report
- rpttype: Type of report (either PowerPoint or Word)
- key_table: Empty (NULL) mapping table for tracking cross referencing (Word only)
- placeholders: Empty list to hold placeholder substitution text (Word only)
- meta: Metadata read in from the yaml file
- mapping: Mapping yaml file
- msgs: Vector of messages indicating any errors that were encountered

Examples

```
obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))
```

`report_add_doc_content` *Add Content to Body of a Word Document Report*

Description

Appends content to the body of a Word document

Usage

```
report_add_doc_content(obnd, type = NULL, content = NULL, verbose = TRUE)
```

Arguments

obnd	onbrand report object
type	Type of content to add
content	Content to add
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Details

For each `content` type listed below the different `content` outlined is expected. Text can be specified in different formats: `"text"` indicates plain text, `"fpar"` is formatted text defined by the `fpar` command from the `officer` package, and `"md"` is text formatted in markdown format (`?md_to_officer` for markdown details).

- `"break"` page break, content is (`NULL`) and a page break will be inserted here
- `"ph"` adds placeholder text substitution
  - `"name"` placeholder name (value in body of text surrounded by three equal signs, e.g. if you have `"===MYPH==="`. in the document the name is just `"MYPH"`)
  - `"value"` value to be substituted into the placeholder (`"my text"`)
  - `"location"` document location where the placeholder will be located (either `"header"`, `"footer"`, or `"body"`)
- `"toc"` generates the table of contents, and content is a list
  - `"level"` number indicating the depth of the contents to display (`3`)
  - `"style"` string containing the style to use (default `NULL` will use the `doc_def`, `Text` style)
- `"section"` formats the current document section
  - `"section_type"` type of section to apply, either `"columns"`, `"continuous"`, `"landscape"`, `"portrait"`, `"columns"`, or `"columns_landscape"`
  - `"width"` override the default page width with this value in inches (`NULL`)
  - `"height"` override the default page height with this value in inches (`NULL`)
  - `"widths"` column widths in inches, number of columns set by number of values (`NULL`)

  - "space" space in inches between columns (NULL)
  - "sep" Boolean value controlling line separating columns (FALSE)
- "text" content is a list containing a paragraph of text with the following elements
  - "text" string containing the text content either a string or the output of "fpar" for formatted text.
  - "style" string containing the style to use (default NULL will use the doc_def, Text style)
  - "format" string containing the format, either "text", "fpar", or "md" (default NULL assumes "text" format)
- "imagefile" content is a list containing information describing an image file with the following elements
  - image string containing path to image file
  - caption caption of the image (NULL)
  - caption_format string containing the format, either "text", "fpar", or "md" (default NULL assumes "text" format)
  - key unique key for cross referencing e.g. "FIG_DATA" (NULL)
  - height height of the image (NULL)
  - width width of the image (NULL)
- "ggplot" content is a list containing a ggplot object, (eg. p = ggplot() + ...) with the following elements
  - image ggplot object
  - caption caption of the image (NULL)
  - caption_format string containing the format, either "text", "fpar", or "md" (default NULL assumes "text" format)
  - key unique key for cross referencing e.g. "FIG_DATA" (NULL)
  - height height of the image (NULL)
  - width width of the image (NULL)
- "table" content is a list containing the table content and other options with the following elements:
  - table data frame containing the tabular data
  - "style" string containing the style to use (default NULL will use the doc_def, Table style)
  - caption caption of the table (NULL)
  - caption_format string containing the format, either "text", "fpar", or "md" (default NULL assumes "text" format)
  - key unique key for cross referencing e.g. "TAB_DATA" (NULL)
  - header Boolean variable to control displaying the header (TRUE)
  - first_row Boolean variable to indicate that the first row contains header information (TRUE)
- "flextable" content is a list containing flextable content and other options with the following elements (defaults in parenthesis):
  - table data frame containing the tabular data
  - caption caption of the table (NULL)

  - caption_format string containing the format, either "text", "fpar", or "md" (default NULL assumes "text" format)
  - key unique key for cross referencing e.g. "TAB_DATA" (NULL)
  - header_top, header_middle, header_bottom (NULL) a list with the same names as the data frame names containing the tabular data and values with the header text to show in the table
  - header_format string containing the format, either "text", or "md" (default NULL assumes "text" format)
  - merge_header (TRUE) Set to true to combine column headers with the same information
  - table_body_alignment, table_header_alignment ("center") Controls alignment
  - table_autofit (TRUE) Automatically fit content, or specify the cell width and height with cwidth (0.75) and cheight (0.25)
  - table_theme ("theme_vanilla") Table theme

- "flextable_object" content is a list specifying the a user defined flextable object with the following elements:
  - ft flextable object
  - caption caption of the table (NULL)
  - key unique key for cross referencing e.g. "TAB_DATA" (NULL)

Value

onbrand object with the content added to the body or isgood set to FALSE with any messages in the msgs field.

Examples

```
# Read  Word template into an onbrand object
obnd = read_template(
 template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
 mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

# The examples below use the following packages
library(ggplot2)
library(flextable)
library(officer)

# Adding text
obnd = report_add_doc_content(obnd,
  type     = "text",
  content  = list(text="Text with no style specified will use the doc_def text format."))


# Text formatted with fpar
fpartext = fpar(
ftext("Formatted text can be created using the ", prop=NULL),
ftext("fpar ", prop=fp_text(color="green")),
ftext("command from the officer package.", prop=NULL))
```

```
obnd = report_add_doc_content(obnd,
 type     = "text",
 content  = list(text   = fpartext,
                 format = "fpar",
                 style  = "Normal"))

# Text formatted with markdown
mdtext = "Formatted text can be created using
**<color:green>markdown</color>** formatting"
obnd = report_add_doc_content(obnd,
 type     = "text",
 content  = list(text   = mdtext,
                 format = "md",
                 style  = "Normal"))


# Adding figures
p = ggplot() + annotate("text", x=0, y=0, label = "picture example")
imgfile = tempfile(pattern="image", fileext=".png")
ggsave(filename=imgfile, plot=p, height=5.15, width=9, units="in")

# From an image file:
obnd = report_add_doc_content(obnd,
type     = "imagefile",
content  = list(image   = imgfile,
              caption = "This is an example of an image from a file."))

# From a ggplot object
obnd = report_add_doc_content(obnd,
type     = "imagefile",
content  = list(image   = imgfile,
              caption = "This is an example of an image from a file."))


#Adding tables
tdf =    data.frame(Parameters = c("Length", "Width", "Height"),
                Values     = 1:3,
                Units      = c("m", "m", "m") )

# Word table
tab_cont = list(table   = tdf,
              caption = "Word Table.")
obnd = report_add_doc_content(obnd,
 type     = "table",
 content  = tab_cont)

# onbrand flextable abstraction:
tab_cont = list(table   = tdf,
              caption = "Word Table.")
obnd = report_add_doc_content(obnd,
type     = "table",
content  = tab_cont)
```

```
# flextable object
tab_fto = flextable(tdf)
obnd = report_add_doc_content(obnd,
 type     = "flextable_object",
 content  = list(ft=tab_fto,
                 caption  = "Flextable object created by the user."))

# Saving the report output
save_report(obnd, tempfile(fileext = ".docx"))
```

`report_add_slide` *Add Slide and Content*

Description

Creates a report slide and populates the content

Usage

```
report_add_slide(obnd, template = NULL, elements = NULL, verbose = TRUE)
```

Arguments

obnd	onbrand report object
template	Name of slide template to use (name from templates in yaml mapping file)
elements	Content and type for each placeholder you wish to fill for this slide: This is a list with names set to palceholders for the specified tempalte. Each placeholder is a list and should have a content element and a type element (see Details below).
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Details

For example consider the mapping information for the slide template `title_slide` with the two place holders `title` and `subtitle`.

```
rpptx:
 master: Office Theme
 templates:
   title_slide:
     title:
       type:         ctrTitle
       index:        1
       ph_label:     Title 1
       content_type: text
     subtitle:
```

```
type:         subTitle
index:        1
ph_label:     Subtitle 2
content_type: text
```

This shows how to populate a title slide with text:

```
obnd = report_add_slide(obnd,
 template = "title_slide",
 elements = list(
    title     = list( content      = "Slide Title",
                      type         = "text"),
    subtitle  = list( content      = "Subtitle",
                      type         = "text")))
```

See the function add_pptx_ph_content for a list of allowed values for type. Note that if mapping defines the content_type as text, you cannot use a list type. Similarly, if the content_type is defined as list, you cannot use a text type.

Value

onbrand report object with either the content added or isgood set to FALSE with any messages in the msgs field.

See Also

add_pptx_ph_content

Examples

```
obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

obnd = report_add_slide(obnd,
 template = "content_text",
 elements = list(
    title         = list( content      = "Text Example",
                          type         = "text"),
    sub_title     = list( content      = "Adding a slide with a block of text",
                          type         = "text"),
    content_body  = list( content      = "A block of text",
                          type         = "text")))
```

save_report *Save Onbrand Report to a File*

Description

Saves report in onbrand object to the specified file.

Usage

```
save_report(obnd, output_file = NULL, verbose = TRUE)
```

Arguments

obnd	onbrand report object
output_file	File name to save the report.
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Value

List with the following elements

- isgood: Boolean variable indicating success or failure
- msgs: Vector of messages

Examples

```
obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

save_report(obnd, tempfile(fileext = ".pptx"))

obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

save_report(obnd, tempfile(fileext = ".docx"))
```

`set_officer_object` *Places Officer Object Into Onbrand Report Object*

Description

After modifying a report object manually, you can return it to the onbrand object using this function.

Usage

```
set_officer_object(obnd, rpt = NULL, verbose = TRUE)
```

Arguments

obnd	onbrand report object
rpt	officer object
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned onbrand object.

Value

onbrand object with the report replaced

See Also

`fetch_officer_object`

Examples

```
obnd = read_template(
      template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
      mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))

# pulling out the report
rpt = fetch_officer_object(obnd)$rpt

# Modifications would be made here with officer directly

# Replacing the report into the onbrand object
obnd = set_officer_object(obnd, rpt)
```

`template_details` *Show Template Details for 'onbrand' Object*

Description

Takes an onbrand object with a loaded template and displays details about the template. For PowerPoint this contains the template names and elements present for that template. For Word it will contain defined text and table styles. This information can be displayed in the console, returned as text or formatted for use in RMarkdown documentation.

Usage

```
template_details(obnd, verbose = TRUE)
```

Arguments

obnd	onbrand report object
verbose	Boolean variable when set to TRUE (default) messages will be displayed on the terminal; Messages will be included in the returned results object.

Value

list with the following elements:

- rpttype: Type of report (either PowerPoint or Word)
- msgs: Vector of messages with details or any errors that were encountered
- txt: Vector of template details in text format
- df: Vector of template details in a dataframe
- ft: Vector of template details in flextable format
- isgood: Boolean variable indicating the current state of the object

Examples

```
obnd = read_template(
   template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
   mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))
details = template_details(obnd)

obnd = read_template(
   template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
   mapping  = file.path(system.file(package="onbrand"), "templates", "report.yaml"))
details = template_details(obnd)
```

`view_layout` *Generate Annotated Layout for Report Templates*

Description

Elements of slide masters are identified by placeholder labels. As PowerPoint masters are created the labels can be difficult to predict. Word documents are identified by style names. This function will create a layout file identifying all of the elements of each slide master for a PowerPoint template or each paragraph and table style for a Word template.

Usage

```
view_layout(
 template = file.path(system.file(package = "onbrand"), "templates", "report.pptx"),
  output_file = NULL,
  verbose = TRUE
)
```

Arguments

`template`	Name of PowerPoint or Word file to annotate (defaults to included PoerPoint template)
`output_file`	name of file to place the annotated layout information, set to `NULL` and it will generate a file named layout with the appropriate extension
`verbose`	Boolean variable when set to TRUE (default) messages will be

Value

List with the following elements

- isgood: Boolean variable indicating success or failure
- rpt: Officer with the annotated layout
- msgs: Vector of messages

Examples

```
lpptx = view_layout(
     template = file.path(system.file(package="onbrand"), "templates", "report.pptx"),
     output_file   = file.path(tempdir(), "layout.pptx"))

ldocx = view_layout(
     template = file.path(system.file(package="onbrand"), "templates", "report.docx"),
     output_file   = file.path(tempdir(), "layout.docx"))
```

Index