Project Management

Matthew Leonawicz

December 24, 2014

1 Example usage

This is how **projman** functions can be used to create and manipulate a project, using the **projman** project itself as an example. The code below is not intended to be run directly, but serves as a guide.

2 R code

2.1 Create a project

Note that I use my own default path for storing a project when creating a new project. See the **projman** [default objects](objects.html "default objects") and [creating a new project](func_new.html "new project") for more details.

```
source("C:/github/ProjectManagement/code/projman.R")  # Eventually load projman package instead
proj.name <- "ProjectManagement"  # Project name
proj.location <- matt.proj.path  # Use default file location

docDir <- c("Rmd/include", "md", "html", "Rnw", "pdf", "timeline")
newProject(proj.name, proj.location, docs.dirs = docDir, overwrite = T)  # create a new project

rfile.path <- file.path(proj.location, proj.name, "code")  # path to R scripts
docs.path <- file.path(proj.location, proj.name, "docs")
rmd.path <- file.path(docs.path, "Rmd")

# generate Rmd files from existing R scripts using default yaml front-matter
genRmd(path = rfile.path, header = rmdHeader())
```

2.2 Update a project

Functions can be used to create, read, or update. See [Rmd-related functions](func_rmd.html "Rmd-related functions").

```
# update yaml front-matter only
genRmd(path = rfile.path, header = rmdHeader(), knitrSetupChunk = rmdknitrSetup(),
    update.header = TRUE)

# obtain knitr code chunk names in existing R scripts
chunkNames(path = file.path(proj.location, proj.name, "code"))

# append new knitr code chunk names found in existing R scripts to any Rmd
# files which are outdated
chunkNames(path = file.path(proj.location, proj.name, "code"), append.new = TRUE)
```

2.3 Prepare a project website

With some additional project-specific setup, files can be generated which will assist in creating a project website. See [website-related functions](func_website.html "website-related functions").

```
# Setup for generating a project website
index.url <- file.path(rmd.path, "proj_intro.html")  # temporary
file.copy(index.url, file.path(rmd.path, "index.html"))

proj.title <- "Project Management"
proj.menu <- c("projman", "R Code", "All Projects")

proj.submenu <- list(c("About projman", "Introduction", "Related items", "Example usage"),
    c("Default objects", "divider", "Functions", "Start a new project", "Working with Rmd files",
        "Make a project website", "Github user website"), c("Projects diagram",
        "divider", "About", "Other"))

proj.files <- list(c("header", "proj_intro.html", "code_sankey.html", "example.html"),
    c("objects.html", "divider", "header", "func_new.html", "func_rmd.html",
        "func_website.html", "func_user_website.html"), c("proj_sankey.html",
        "divider", "proj_intro.html", "proj_intro.html"))

proj.github <- file.path("https://github.com/leonawicz", proj.name)

# generate navigation bar html file common to all pages
genNavbar(htmlfile = file.path(proj.location, proj.name, "docs/Rmd/include/navbar.html"),
    title = proj.title, menu = proj.menu, submenus = proj.submenu, files = proj.files,
    title.url = "index.html", home.url = "index.html", site.url = proj.github,
    include.home = FALSE)

# generate _output.yaml file Note that external libraries are expected,
# stored in the 'libs' directory below
yaml.out <- file.path(proj.location, proj.name, "docs/Rmd/_output.yaml")
libs <- "libs"
common.header <- "include/in_header.html"
genOutyaml(file = yaml.out, lib = libs, header = common.header, before_body = "include/navbar.html")
```