exportPublishedToPdf

Export Matlab generated HTML documentation (publish) to pdf-files and combine them into a LaTeX index file ready compile to pdf manual. This script works on unix sytems only or needs to be adjusted for windows systems for library path and wkhtmltopdf binary path.

Runs on Unix systems only!

Requirements

- Other m-files required: src/util/removeFilesFromDir.m
- Subfunctions: wkhtmltopdf (shell), pdflatex (shell)
- MAT-files required: data/config.mat

See Also

- generateConfigMat
- system
- wkhtmltopdf
- publishProjectFilesToHTML
- Documentation Workflow

Created on December 10. 2020 by Tobias Wulf. Copyright Tobias Wulf 2020.

Start Exporting Script, Clean Up and Load Config

At first clean up junk from workspace and clear prompt for new output. Set project root path to create absolute file path with fullfile function. Load absolute path variables and publishing options from config.mat

```
disp('Workspace cleaned up ...');
clearvars;
clc;
disp('Load configuration ...');
try
    load('config.mat', 'PathVariables');
catch ME
    rethrow(ME);
end
```

Define Manual TOC

The maual toc must be in the same order as in helptoc.xml in the publish html folder. The toc is used to generate a latex file to include for appendices.

```
toc = ["section",       "GaussianProcessDipoleSimulation.pdf";
       "section"        "Workflows.pdf";
       "subsection", "Project_Preparation.pdf";
       "subsection", "Project_Structure.pdf";
       "subsection", "Git_Feature_Branch_Workflow.pdf";
       "subsection", "Documentation_Workflow.pdf";
       "subsection", "Simulation_Workflow.pdf";
       "section",    "Executable_Scripts.pdf";
       "subsection", "publishProjectFilesToHTML.pdf";
       "subsection", "generateConfigMat.pdf";
       "subsection", "generateSimulationDatasets.pdf";
```

```
        "subsection", "deleteSimulationDatasets.pdf";
        "subsection", "deleteSimulationPlots.pdf";
        "subsection", "exportPublishedToPdf";
        "section",    "Source_Code.pdf";
        "subsection", "sensorArraySimulation.pdf";
        "subsubsection",     "rotate3DVector.pdf";
        "subsubsection",     "generateDipoleRotationMoments.pdf";
        "subsubsection",     "generateSensorArraySquareGrid.pdf";
        "subsubsection",     "computeDipoleH0Norm.pdf";
        "subsubsection",     "computeDipoleHField.pdf";
        "subsubsection",     "simulateDipoleSquareSensorArray.pdf";
        "subsection", "util.pdf";
        "subsubsection",     "removeFilesFromDir.pdf";
        "subsubsection",     "publishFilesFromDir.pdf";
        "subsubsection",     "plotFunctions.pdf";
        "paragraph",  "plotTDKCharDataset.pdf";
        "paragraph",  "plotTDKCharField.pdf";
        "paragraph",  "plotTDKTransferCurves.pdf";
        "paragraph",  "plotKMZ60CharDataset.pdf";
        "paragraph",  "plotKMZ60CharField.pdf";
        "paragraph",  "plotKMZ60TransferCurves.pdf";
        "paragraph",  "plotDipoleMagnet.pdf";
        "paragraph",  "plotSimulationDataset.pdf";
        "paragraph",  "plotSingleSimulationAngle.pdf";
        "paragraph",  "plotSimulationSubset.pdf";
        "paragraph",  "plotSimulationCosSinStats.pdf"
        "paragraph",  "plotSimulationDatasetCircle.pdf";
        "section",    "Datasets.pdf";
        "subsection", "TDK_TAS2141_Characterization.pdf";
        "subsection", "NXP_KMZ60_Characterization.pdf";
        "subsection", "Config_Mat.pdf";
        "subsection", "Training_and_Test_Datasets.pdf";
        "section",    "Unit_Tests.pdf";
        "subsection", "runTests.pdf";
        "subsection", "removeFilesFromDirTest.pdf";
        "subsection", "rotate3DVectorTest.pdf";
        "subsection", "generateDipoleRotationMomentsTest.pdf";
        "subsection", "generateSensorArraySquareGridTest.pdf";
        "subsection", "computeDipoleH0NormTest.pdf";
        "subsection", "computeDipoleHFieldTest.pdf";
        "subsection", "tiltRotationTest.pdf";];

nToc = length(toc);
fprintf("%d toc entries remarked ...\n", nToc);
```

Scan for HTML Files

Scan for all published HTML files in the project publish directory.

```
disp('Scan for published files ...');
HTML = dir(fullfile(PathVariables.publishHtmlPath, '*.html'));
if nToc ~= length(HTML)
    warning(...
        'TOC (%d) length and found HTML (%d) files are diverging.', ...
        nToc, length(HTML));
end
```

Export HTML to Pdf

Export found HTML files to Pdf files. Each file gets its own Pdf represenstation. Filename is kept with pdf extension. Write files into

Manual folder under LaTeX subdirectory in docs path. Using wkhtmltopdf shell application. Get filename, add pdf extension new path to file. Create shell string to execute with system command. Get current library path (Matlab) and change it to system library path to execute wkhtmltopdf after that restor library back to Matlab.

```
disp('Change local library path to system path ...');
matlabLibPath = getenv('LD_LIBRARY_PATH');
systemLibPath = '/usr/lib/x86_64-linux-gnu';
setenv('LD_LIBRARY_PATH', systemLibPath);

disp('Export published HTML to Pdf ...');
fprintf('Source: %s\n', HTML(1).folder);
fprintf('Destination: %s\n', PathVariables.exportPublishPath);
for fhtml = HTML'
    disp(fhtml.name);
    [~, fName, ~] = fileparts(fhtml.name);
    sourcePath = fullfile(fhtml.folder, fhtml.name);
    destinationPath = fullfile(...
        PathVariables.exportPublishPath, [fName '.pdf']);

    cmdStr = join(["wkhtmltopdf", ...
        "-B 47mm", ...
        "-L 27mm", ...
        "-R 27mm", ...
        "-T 37mm", ...
        "--minimum-font-size 12", ...
        "--enable-local-file-access", ...
        "--disable-external-links", ...
        "--disable-internal-links", ...
        ... "--disable-smart-shrinking", ...
        "--window-status finished", ...
        "--no-stop-slow-scripts", ...
        "--javascript-delay 2000", ...
        "%s %s"]);
    shellStr = sprintf(cmdStr, sourcePath, destinationPath);

    try
        [status, cmdout] = system(shellStr);
        % disp(cmdout);
        if status ~= 0
            error('Export failure.');
        end
    catch ME
        setenv('LD_LIBRARY_PATH', matlabLibPath);
        disp(cmdout);
        rethrow(ME)
    end
end

disp('Restore local library path ...');
setenv('LD_LIBRARY_PATH', matlabLibPath);
```

Write TOC to LaTeX File

Wirete TOC to LaTeX file and generate for each pdf to include a toc content line with marked toc depth. Get the number of pages and add only page title first pdf page.

```
disp('Write TOC to Manual.tex ...');
addFirstPage = "\\addtocounter{%s}{1}\n" + ...
    "\\includepdf[page=1," + ...
    "pagecommand={\\phantomsection\\" + ...
    "addcontentsline{toc}{%s}" + ...
    "{\\protect\\numberline{\\the%s}%s}\\label{%s}}]{%s}\n";
addRestPages = "\\includepdf[page=2-, pagecommand={\\phantomsection}]{%s}\n";

fileID = fopen(fullfile(...
    PathVariables.exportPublishPath, 'Manual.tex'), 'w');
% fprintf(fileID, "%% !TEX root = ../thesis.tex\n");
fprintf(fileID, "%% appendix software documentation\n");
fprintf(fileID, "%% @author Tobias Wulf\n");
fprintf(fileID, ...
    "%% Autogenerated LaTeX file. Generated by exportPublishedToPdf.\n");
fprintf(fileID, ...
    "%% Software manual with TOC generated in the same script.\n");
fprintf(fileID, "%% Generated on %s.\n\n", datestr(datetime('now')));

pat = regexpPattern("\d+");
shellStr = "pdfinfo %s | grep Pages";

for i = 1:nToc
    level = toc(i);
    fName = toc(i,2);
    [~, titleStr, ~] = fileparts(fName);
    titleStr = strrep(titleStr, '_', ' ');
    try
        [status, cmdout] = system(sprintf(shellStr, ...
            fullfile(PathVariables.exportPublishPath, fName)));
        pages = double(extract(string(cmdout), pat));

        fprintf(fileID, addFirstPage, level, level, level, ...
            titleStr, titleStr, fName);
        if pages > 1, fprintf(fileID, addRestPages, fName); end

    catch ME
        setenv('LD_LIBRARY_PATH', matlabLibPath);
        fclose(fileID);
        disp(cmdout);
        rethrow(ME)
    end
end
fclose(fileID);
```