publishProjectFilesToHTML

The is script is used to publish all toolbox included files to HTML documentation folder docs/html. The script runs a section with certain options for each project part and uses the built-in function to generate the documentation files. For a complete documentation support each generated html document needs to get listed in the project helptoc file with toc entry.

Contents

Requirements

- Other m-files required: src/util/removeFilesFromDir.m
- Subfunctions: None
- MAT-files required: data/config.mat

See Also

- generateConfigMat
- publishFilesFromDir
- builddocsearchdb
- removeFilesFromDir
- Documentation Workflow

Start Publishing 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', 'PublishOptions');
catch ME
    rethrow(ME);
end
```

Remove Equation PNG Files

Remove equation png file from HTML output folder before create or recreate publishing files. To prevent the directory expanse of old or edited equation files.

```
yesno = input('Renew eqautions in docs [y/n]: ', 's');
if strcmp(yesno, 'y')
    removeFilesFromDir(PublishOptions.outputDir, '*_eq*.png');
end
```

Project Documentation Files

In this section of the publish script every bare documentation script should be handled and executed to publish. These are m-files without any executeable code so they exist just to transport the documentation content into html output. Dir all m-files from docs path. Not recursively but verbose. No expected directory tree search for m-files.

```
disp('Publish project documentation files ...');
publishFilesFromDir(PathVariables.docsPath, PublishOptions, false, true);
```

Executable Script Files

The section collects all ready to execute scripts from project scripts folder and publish them to html documentation folder. Every script must be notice in in Executable_Scripts.m file with one line description. That is very important to not execute the scripts during publishing. If a script contains critical or loop gaining code. In example the publishProjectFilesToHTML.m script such loop gaining code. If eval code during publishing is enabled the script starts publishing itself over and over again because it contains the loop entry via the publish function. So routine is minmal adjusted by evalCode parameter in PublishOptions struct. No expected directory to search for m-files so no recursively but verbose

```
disp('Publish executable scripts ...');
PublishOptions.evalCode = false;
publishFilesFromDir(PathVariables.scriptsPath, PublishOptions, false, true);
```

Source Code Functions and Classes

That part of the publish script collects function and class m-files from the util section of the source code located in src/. Introcude every new m-file to the source code related documentation m-file and add a description. In general functions and class files are not executed on publishing execution so set evalCode option to false in PublishOptions struct. In addition to that the source code itself should not be in the published document, so the showCode option is switched to false. Publish recursively from underlaying directory tree, verbose.

```
disp('Publish source code functions and classes ...');
PublishOptions.evalCode = false;
publishFilesFromDir(PathVariables.srcPath, PublishOptions, true, true);
```

Unit Test Scripts

Publish unit tests scripts and run test suite to include unit test results. It is the only section of whole publish process which executes script code. The test files are closed loop and do not harm other sections of source code.

```
disp('Publish unit tests scripts ...');
PublishOptions.evalCode = true;
publishFilesFromDir(PathVariables.unittestPath, ...
```

```
    PublishOptions, true, true);
```

Build Documentation Database for Matlab Help Browser

To support Matlabs help browser it is needed build searchable help browser entries including a searchable database backend. Matlabs built-in function builddocsearchdb does the trick. The function just needs the output directory of builded html documentation and it creates a subfolder which includes the database. About the info.xml from the project root and the helptoc.xml file the html documentation folder all listet documentation is accessable. At first remove old database before build the new reference database. Remove autogenerated directory helpsearch-v3. At first get folder content and remove first two relative directory entries from struct. Then delete files and check if files do not exist any more. At least build up new search database entries to Matlab help.

```
disp('Remove old search entries ...');
clearvars;
close all;
clc;
disp('Reload configuration after unit test execution ...');
try
    load('config.mat', 'PathVariables', 'PublishOptions');
catch ME
    rethrow(ME);
end
if removeFilesFromDir(PathVariables.helpsearchPath)
    builddocsearchdb(PublishOptions.outputDir);
else
    disp('Could not remove old search entries ...');
end
```

Open Generated Documentation.

Open generated HTML documentation from documentation root HTML file which should be a project introduction or project roadmap page. Comment out if this script is added to project shutdown tasks.

```
open(fullfile(PublishOptions.outputDir, ...
    'GaussianProcessDipoleSimulation.html'));
disp('Done ...');
```

Published with MATLAB® R2020b