**AcuGitLibrary Documentation**

*RepoMaker* (Class)

Public Variables

string *Root* : (Default is null) the directory that will be made into a repository, or where the repositories will be made.

bool *IncludeSubs* : (Default is false) if true it will create repositories in all folders under the root path.

bool *SingleDirectory* : (Default is false) make the root directory into a repository

bool *Overwrite* : (Default is true) will delete pre-existing Git repositories

bool *SetDefaultIgnore* : (Default is true) will create a default .gitignore file that ignores (.db, .pdf, .doc, .docx, .xls, .xlsx, .tmp)

Constructors







Methods

void *Run()*

*Description:* Simply executes the RepoMaker

*e.g.*

**

*ChangeRequest* (Class)

Public Variables

enum *RequestType* : Two types

* Job\_Based : The ChageRequest will use the job number to submit the change
* Precise : Requires the user to enter the path of the master repository and the location of the submitted file

Constructor



string *subPath* : The path to the folder where the files that are being submitted are stored

string *repoPath* : (RequestType.Precise)The path to the master repository

(RequestType.Job\_Based) The path to where the jobs folders are stored

RequestType *reqType* : (Default is Precise) The RequestType that you are submitting

bool *deleteSub* : (Default is false) Delete the submitted files

*Important Note!: This feature (Delete sub) has not been implemented in the most recent version of the library*

string *jobNumb* : The job number of the files being submitted

Methods

void *Submit()*

*Description:* Executes the ChangeRequest*.*

*Email* (Class)

Public Variables

enum *Smtp* : Two types

* AcutecPA : Acutec's noreply email set up for PA, administrator access will need to be used for this I believe
* AcutecSc : Acutec's llcit@acutecind.com address set up for use in Acutec's South Carolina location

Constructors

Smtp *SmtpServer* : (Default is AcutecPA) The email account you want to use

string[] *Recievers* : (Default is null) An array of the email addresses you wish to send the notifications to

**or**

string *MailingList* : (Default is null) The path to a .csv file that has a list of the email addresses you wish to send the notifications to

string *Subject* : (Default is “Automated Email Response as Part of AcuGit System”) The subject of the email

string *Body* : (Default is “This email has been sent … ”) The body of the email

Methods

void *Send()*

*Description:* Sends out the email.

*Git* (Class)

Public Variables

enum *ResetMode* : Three types

* Soft: Will reverse the working directory to the indicated commit leaving any staged or changed files alone
* Mixed: (Set as the Default) Will reverse the working directory to the indicated commit while also removing all files from staging
* Hard: Will not only reverse the working directory to the indicated commit it will also undo any possible changes to any files within the repository

List<CommitInfo> *CommitList* : (Default is empty) the list of all commits of this repository. See CommitInfo below.

Constructor

string *repoPath* : (Default is null)

Methods

void *Stage(string filename = “\*”)*

*Description:* Works the same as *git add*, and by default it is set to *(git add –all)*

*e.g.***

void *Reset(ResetMode resetMode = Mixed, int stepBack = 0)*

*Description:* Resets commits or changes made in the master repository (Untested at this time)

e.g.



void *Commit(string message, string username = “(Automated)”, string email\_Address = “--”)*

*Description:* Commit the files that are staged to the currently checked out branch

e.g.



*Git* (Class) cont.

Methods

void *Commit()*

*Description:* Quick Commit, useful for unimportant commits such as those used when initializing a repository

e.g.

void *Init()*

*Description:* Initializes (Creates) the Git repository

e.g.

void *Branch(string branchName)*

*Description:* Creates a new branch “branchName” on the given Git repository

e.g.

void *Checkout(string branchName)*

*Description:* Checks out the branch given by “branchName” on the given Git repository

Note: When a Branch is checked out any files that have been commited to it will now be visible, so it may be important to lock out folders when the master branch is not checked out

e.g.

void *SetPath(string \_repoPath)*

*Description:* Sets a new path for the Git object to point to, this can be used if one didn’t want to create a new Git object.

e.g.

string *GetPath()*

*Returns:* The current working path of the repository the Git object is pointing to.

e.g.

*Git* (Class) cont.

Methods

void *CreateIgnore()*

*Description:* Creates the .gitignore file in the root of the repositories working directory

e.g.

void *AddIgnore(string[] Ignoreables)*

*Description:* Appends an array of ignore statements to the .gitignore file

e.g.

void *AddIgnore(string Ignoreables)*

*Description:* Appends a single ignore statement to the .gitignore file

e.g.

void *ResetIgnore()*

*Description:* Deletes and recreates the .gitignore file

e.g.

string *GetIgnore()*

*Returns:* A comma separated list of all ignore statements

e.g.

bool *Ignored(string ignore\_argument)*

*Returns:* true if an ignore argument is within the .gitignore and false if it is not

e.g.

void *Remove(string filename)*

*Description:* Deletes a single file from a

e.g.

*Git* (Class) cont.

Methods

void *Untrack(string filename)*

*Description:* Removes a file from being tracked by the Git repository

e.g.

void *CommitList()*

*Description:* Builds/Rebuilds the Git objects public List of CommitInfo variables which represent all the commits in the repository. This method at this point is the only way to cherry-pick a commit, and revert (unless one uses a Libgit2sharp.Commit variable for Revert)

e.g.

void *CommitPick(CommitInfo info)*

*Description:* Adds a particular commit to the master branch of the repository

e.g.

bool *Revert(LibGit2Sharp.Commit commit)*

*Description:* Reverts a given commit, and will return a bool for whether or not the revert succeeds.

e.g.

bool *Revert(CommitInfo info)*

*Description:* Reverts a given commit, and will return a bool for whether or not the revert succeeds.

e.g.

*CommitInfo* (Class)

Public Variables

string *BranchName* : (Default is empty) The user friendly name of the Branch this commit resides in.

string *CommitShortMessage* : (Default is empty) The first line of the commit.

string *ID* : (Default is empty) The SHA id of the commit.

*LibGit2Sharp.Commit commit* : (Default is null) The commit that the CommitInfo object will be referring to

*LibGit2Sharp.Branch branch* : (Default is null) The branch that the CommitInfo object is within

Constructors

