NAME

git-sh-setup − Common Git shell script setup code

SYNOPSIS

. "$(git −−exec−path)/git−sh−setup"

DESCRIPTION

This is not a command the end user would want to run. Ever. This documentation is meant for people who are studying the Porcelain−ish scripts and/or are writing new ones.

The *git sh−setup* scriptlet is designed to be sourced (using **.**) by other shell scripts to set up some variables pointing at the normal Git directories and a few helper shell functions.

Before sourcing it, your script should set up a few variables; **USAGE** (and **LONG_USAGE**, if any) is used to define message given by **usage()** shell function. **SUBDIRECTORY_OK** can be set if the script can run from a subdirectory of the working tree (some commands do not).

The scriptlet sets **GIT_DIR** and **GIT_OBJECT_DIRECTORY** shell variables, but does **not** export them to the environment.

FUNCTIONS

die
: exit after emitting the supplied error message to the standard error stream.

usage
: die with the usage message.

set_reflog_action
: Set **GIT_REFLOG_ACTION** environment to a given string (typically the name of the program) unless it is already set. Whenever the script runs a **git** command that updates refs, a reflog entry is created using the value of this string to leave the record of what command updated the ref.

git_editor
: runs an editor of user's choice (GIT_EDITOR, core.editor, VISUAL or EDITOR) on a given file, but error out if no editor is specified and the terminal is dumb.

is_bare_repository
: outputs **true** or **false** to the standard output stream to indicate if the repository is a bare repository (i.e. without an associated working tree).

cd_to_toplevel
: runs chdir to the toplevel of the working tree.

require_work_tree
: checks if the current directory is within the working tree of the repository, and otherwise dies.

require_work_tree_exists
: checks if the working tree associated with the repository exists, and otherwise dies. Often done before calling cd_to_toplevel, which is impossible to do if there is no working tree.

require_clean_work_tree <action> [<hint>]
: checks that the working tree and index associated with the repository have no uncommitted changes to tracked files. Otherwise it emits an error message of the form **Cannot <action>: <reason>. <hint>**, and dies. Example:

  require_clean_work_tree rebase "Please commit or stash them."

get_author_ident_from_commit
: outputs code for use with eval to set the GIT_AUTHOR_NAME, GIT_AUTHOR_EMAIL and GIT_AUTHOR_DATE variables for a given commit.

create_virtual_base
: modifies the first file so only lines in common with the second file remain. If there is insufficient common material, then the first file is left empty. The result is suitable as a virtual base input for a 3−way merge.

GIT

Part of the **git**(1) suite