NAME

gitmodules - defining submodule properties

SYNOPSIS

$GIT_WORK_DIR/.gitmodules

DESCRIPTION

The .gitmodules file, located in the top-level directory of a Git working tree, is a text file with a syntax matching the requirements of **git-config(1)**.

The file contains one subsection per submodule, and the subsection value is the name of the submodule. The name is set to the path where the submodule has been added unless it was customized with the *--name* option of *git submodule add*. Each submodule section also contains the following required keys:

submodule.<name>.path
: Defines the path, relative to the top-level directory of the Git working tree, where the submodule is expected to be checked out. The path name must not end with a /. All submodule paths must be unique within the .gitmodules file.

submodule.<name>.url
: Defines a URL from which the submodule repository can be cloned. This may be either an absolute URL ready to be passed to **git-clone(1)** or (if it begins with ./ or ../) a location relative to the superproject's origin repository.

In addition, there are a number of optional keys:

submodule.<name>.update
: Defines what to do when the submodule is updated by the superproject. If *checkout* (the default), the new commit specified in the superproject will be checked out in the submodule on a detached HEAD. If *rebase*, the current branch of the submodule will be rebased onto the commit specified in the superproject. If *merge*, the commit specified in the superproject will be merged into the current branch in the submodule. If *none*, the submodule with name $name will not be updated by default.

  This config option is overridden if git submodule update is given the --merge, --rebase or --checkout options.

submodule.<name>.branch
: A remote branch name for tracking updates in the upstream submodule. If the option is not specified, it defaults to *master*. See the --remote documentation in **git-submodule(1)** for details.

submodule.<name>.fetchRecurseSubmodules
: This option can be used to control recursive fetching of this submodule. If this option is also present in the submodules entry in .git/config of the superproject, the setting there will override the one found in .gitmodules. Both settings can be overridden on the command line by using the --[no-]recurse-submodules option to git fetch and git pull.

submodule.<name>.ignore
: Defines under what circumstances git status and the diff family show a submodule as modified. When set to all, it will never be considered modified (but will nonetheless show up in the output of status and commit when it has been staged), dirty will ignore all changes to the submodules work tree and takes only differences between the HEAD of the submodule and the commit recorded in the superproject into account. untracked will additionally let submodules with modified tracked files in their work tree show up. Using none (the default when this option is not set) also shows submodules that have untracked files in their work tree as changed. If this option is also present in the submodules entry in .git/config of the superproject, the setting there will override the one found in .gitmodules. Both settings can be overridden on the command line by using the --ignore-submodule option. The *git submodule* commands are not affected by this setting.

EXAMPLES

Consider the following .gitmodules file:

```
[submodule libfoo]
path = include/foo
url = git://foo.com/git/lib.git

[submodule libbar]
path = include/bar
url = git://bar.com/git/lib.git
```

This defines two submodules, libfoo and libbar. These are expected to be checked out in the paths *include/foo* and *include/bar*, and for both submodules a URL is specified which can be used for cloning the submodules.

SEE ALSO

git-submodule(1) git-config(1)

GIT

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