NAME

go – tool for managing Go source code

DESCRIPTION

An import path (see **go-importpath**(1)) denotes a package stored in the local file system. Certain import paths also describe how to obtain the source code for the package using a revision control system.

A few common code hosting sites have special syntax:

BitBucket (Mercurial)

```
import "bitbucket.org/user/project"
import "bitbucket.org/user/project/sub/directory"
```

GitHub (Git)

```
import "github.com/user/project"
import "github.com/user/project/sub/directory"
```

Google Code Project Hosting (Git, Mercurial, Subversion)

```
import "code.google.com/p/project"
import "code.google.com/p/project/sub/directory"

import "code.google.com/p/project.subrepository"
import "code.google.com/p/project.subrepository/sub/directory"
```

Launchpad (Bazaar)

```
import "launchpad.net/project"
import "launchpad.net/project/series"
import "launchpad.net/project/series/sub/directory"

import "launchpad.net/~user/project/branch"
import "launchpad.net/~user/project/branch/sub/directory"
```

For code hosted on other servers, import paths may either be qualified with the version control type, or the go tool can dynamically fetch the import path over https/http and discover where the code resides from a <meta> tag in the HTML.

To declare the code location, an import path of the form

```
repository.vcs/path
```

specifies the given repository, with or without the .vcs suffix, using the named version control system, and then the path inside that repository. The supported version control systems are:

Bazaar
.bzr

Git
.git

Mercurial
.hg

Subversion
.svn

For example,

```
import "example.org/user/foo.hg"
```

denotes the root directory of the Mercurial repository at example.org/user/foo or foo.hg, and

```
import "example.org/repo.git/foo/bar"
```

denotes the foo/bar directory of the Git repository at example.com/repo or repo.git.

When a version control system supports multiple protocols, each is tried in turn when downloading. For example, a Git download tries git://, then https://, then http://.

If the import path is not a known code hosting site and also lacks a version control qualifier, the go tool attempts to fetch the import over https/http and looks for a <meta> tag in the document's HTML <head>.

The meta tag has the form:

```
<meta name="go-import" content="import-prefix vcs repo-root">
```

The import-prefix is the import path corresponding to the repository root. It must be a prefix or an exact match of the package being fetched with "go get". If it's not an exact match, another http request is made at the prefix to verify the <meta> tags match.

The vcs is one of "git", "hg", "svn", etc,

The repo-root is the root of the version control system containing a scheme and not containing a .vcs qualifier.

For example,

```
import "example.org/pkg/foo"
```

will result in the following request(s):

```
https://example.org/pkg/foo?go-get=1 (preferred)
http://example.org/pkg/foo?go-get=1  (fallback)
```

If that page contains the meta tag

```
<meta name="go-import" content="example.org git https://code.org/r/p/exproj">
```

the go tool will verify that https://example.org/?go-get=1 contains the same meta tag and then git clone https://code.org/r/p/exproj into GOPATH/src/example.org.

New downloaded packages are written to the first directory listed in the GOPATH environment variable (see **go-path**(1)).

The go command attempts to download the version of the package appropriate for the Go release being used. See **go-install**(1) for more.

AUTHOR

This manual page was written by Michael Stapelberg <stapelberg@debian.org>, for the Debian project (and may be used by others).