10.5 Git Internals - The Refspec

The Refspec

Throughout this book, we've used simple mappings from remote branches to local references, but they can be more complex. Suppose you were following along with the last couple sections and had created a small local Git repository, and now wanted to add a *remote* to it:

```
$ git remote add origin https://github.com/schacon/simplegit-progit
```

Running the command above adds a section to your repository's `.git/config` file, specifying the name of the remote (`origin`), the URL of the remote repository, and the *refspec* to be used for fetching:

```
[remote "origin"]
        url = https://github.com/schacon/simplegit-progit
        fetch = +refs/heads/*:refs/remotes/origin/*
```

The format of the refspec is, first, an optional `+`, followed by `<src>:<dst>`, where `<src>` is the pattern for references on the remote side and `<dst>` is where those references will be tracked locally. The `+` tells Git to update the reference even if it isn't a fast-forward.

In the default case that is automatically written by a `git remote add origin` command, Git fetches all the references under `refs/heads/` on the server and writes them to `refs/remotes/origin/` locally. So, if there is a `master` branch on the server, you can access the log of that branch locally via any of the following:

```
$ git log origin/master
$ git log remotes/origin/master
$ git log refs/remotes/origin/master
```

They're all equivalent, because Git expands each of them to `refs/remotes/origin/master`.

If you want Git instead to pull down only the `master` branch each time, and not every other branch on the remote server, you can change the fetch line to refer to that branch only:

```
fetch = +refs/heads/master:refs/remotes/origin/master
```

This is just the default refspec for `git fetch` for that remote. If you want to do a one-time only fetch, you can specify the specific refspec on the command line, too. To pull the `master` branch on the remote down to `origin/mymaster` locally, you can run:

```
$ git fetch origin master:refs/remotes/origin/mymaster
```

You can also specify multiple refspecs. On the command line, you can pull down several branches like so:

```
$ git fetch origin master:refs/remotes/origin/mymaster \
         topic:refs/remotes/origin/topic
From git@github.com:schacon/simplegit
 ! [rejected]        master     -> origin/mymaster  (non fast forward)
 * [new branch]      topic      -> origin/topic
```

In this case, the `master` branch pull was rejected because it wasn't listed as a fast-forward reference. You can override that by specifying the `+` in front of the refspec.

You can also specify multiple refspecs for fetching in your configuration file. If you want to always fetch the `master` and `experiment` branches from the `origin` remote, add two lines:

```
[remote "origin"]
        url = https://github.com/schacon/simplegit-progit
        fetch = +refs/heads/master:refs/remotes/origin/master
        fetch = +refs/heads/experiment:refs/remotes/origin/experiment
```

Since Git 2.6.0 you can use partial globs in the pattern to match multiple branches, so this works:

```
fetch = +refs/heads/qa*:refs/remotes/origin/qa*
```

Even better, you can use namespaces (or directories) to accomplish the same with more structure. If you have a QA team that pushes a series of branches, and you want to get the `master` branch and any of the QA team's branches but nothing else, you can use a config section like this:

```
[remote "origin"]
        url = https://github.com/schacon/simplegit-progit
        fetch = +refs/heads/master:refs/remotes/origin/master
        fetch = +refs/heads/qa/*:refs/remotes/origin/qa/*
```

If you have a complex workflow process that has a QA team pushing branches, developers pushing branches, and integration teams pushing and collaborating on remote branches, you can namespace them easily this way.

Pushing Refspecs

It's nice that you can fetch namespaced references that way, but how does the QA team get their branches into a `qa/` namespace in the first place? You accomplish that by using refspecs to push.

If the QA team wants to push their `master` branch to `qa/master` on the remote server, they can run:

```
$ git push origin master:refs/heads/qa/master
```

If they want Git to do that automatically each time they run `git push origin`, they can add a `push` value to their config file:

```
[remote "origin"]
        url = https://github.com/schacon/simplegit-progit
        fetch = +refs/heads/*:refs/remotes/origin/*
        push = refs/heads/master:refs/heads/qa/master
```

Again, this will cause a `git push origin` to push the local `master` branch to the remote `qa/master` branch by default.

Note You cannot use the refspec to fetch from one repository and push to another one. For an example to do so, refer to Keep your GitHub public repository up-to-date.

Deleting References

You can also use the refspec to delete references from the remote server by running something like this:

```
$ git push origin :topic
```

Because the refspec is `<src>:<dst>`, by leaving off the `<src>` part, this basically says to make the `topic`branch on the remote nothing, which deletes it.

Or you can use the newer syntax (available since Git v1.7.0):

```
$ git push origin --delete topic
```