Submodules

Creating Submodules

This is the equivalent to SVN externals. Submodules are subdirectories of a Git project that point to another Git project.

Starting in an existing Git repository, add a submodule.

```
git submodule add git://somehost/someproject.git mysubproj
```

This will create a subfolder named `mysubproj` in the current directory and a new file called `.gitmodules` that contains mappings to the repositories.

Git tracks the submodule and the parent module separately. To change, update, or modify the submodule, `cd` into it's directory.

Git helps the parent project precisely track the commit (not a symbolic `master` or `HEAD`) hash that the submodule is at for this parent project. This commit point can be different than another parent project that points to this submodule at a different commit point.

Restoring Submodules

To check out a project and set up the submodules:

```
git clone <git://url/yourrepo.git>
cd <yourrepo>
# Setup the submodule .gitmodules file
git submodule init
# Retrieve the submodules' code
git submodule update
```

If another developer changes the submodule, then commits the parent project and you pull and retrieve that, you'll need to update the submodules again to stop your `git status` from telling you the tree is dirty.

```
git submodule update
```

The main concept is just to put the submodule in whatever state you want it to be in, then add and commit from the parent project. If a colleague does that and then you pull the parent project, you need to `update` to get the latest submodule code. Remember that submodule changes need to be pushed distinctly from their parent. If a developer fails to do that, her colleagues won't be able to retrieve the commits; the parent project will be pointing to a hash that isn't publicly available.