**Git原理 - Git对象[tree对象]（一）**

**1、Tree对象介绍**

接下来要探讨的 Git 对象类型是树对象（tree object），它能解决文件名保存的问题。tree对象可以存储文件名，也允许我们将多个文件组织到一起。

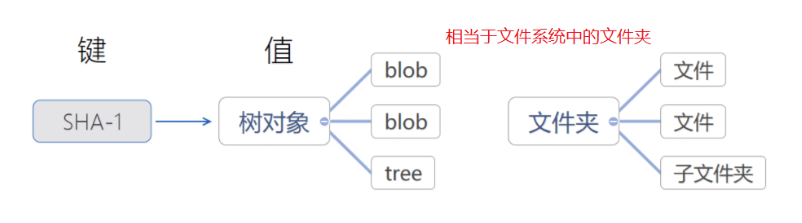
Git以一种类似于UNIX文件系统的方式存储内容，但做了一些简化。所有内容均以树（tree）对象和数据（blob）对象的形式存储，其中树对象对应了UNIX中的目录项，数据对象blob则大致上对应了文件中的内容。

一个树对象可以包含一条或多条记录（tree对象和blob对象），每条记录含有一个指向blob对象或者子tree对象的SHA-1指针，以及相应的模式、类型、文件名信息。

如下图：

|  |
| --- |
| Bash # 文件模式、对象类型、对象的SHA-1指针、文件名 100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt |

**Tree对象存储方式如下图所示**：



**2、Tree对象说明**

**（1）初始化一个新的本地版本库**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning $ git init Initialized empty Git repository in J:/git-repository/git\_learning/.git/  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll -a total 8 drwxr-xr-x 1 L 197121 0 4月 11 14:50 ./ drwxr-xr-x 1 L 197121 0 4月 10 20:23 ../ drwxr-xr-x 1 L 197121 0 4月 11 14:50 .git/ |

**（2）创建一个树对象（重点）**

**1）新建一个文件，然后把文件提交到本地版本库**

例如：新建文件test.txt，文件内容version 1

|  |
| --- |
| Bash # 创建文件 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ echo "version 1" >> test.txt  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll total 1 -rw-r--r-- 1 L 197121 10 4月 11 14:57 test.txt  # 查看文件内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ cat test.txt version 1 |

**2）把test.txt文件，提交到本地版本库**

|  |
| --- |
| Bash # 1.test.txt文件提交到本地版本库 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git hash-object -w ./test.txt 83baae61804e65cc73a7201a7252750c76066a30  # 2.查看Git数据库内容，可以看到新增了一个blob对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/83/baae61804e65cc73a7201a7252750c76066a30  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -t 83baae61804e65cc73a7201a7252750c76066a30 blob  # 3.查看blob对象对象内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -p 83baae61804e65cc73a7201a7252750c76066a30 version 1 |

以上就和我们讲blob对象的操作一样。

此时test.txt文件被管理在Git本地版本库中。

**3）创建一个树对象**

通常Git是**根据暂存区或者索引文件index来创建tree对象**，因此要把文件存储到暂存区进并建立index文件。

提示1：

|  |
| --- |
| index文件在.git目录中，最新初始化的Git本地仓库是没有index文件，只有添加过一次数据到暂存区之后，才会在.git目录中自动生成index文件。 |

新初始化的.git目录内容如下：是没有index文件的。

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll .git/ total 7 -rw-r--r-- 1 L 197121 130 4月 11 14:50 config -rw-r--r-- 1 L 197121 73 4月 11 14:50 description -rw-r--r-- 1 L 197121 23 4月 11 14:50 HEAD drwxr-xr-x 1 L 197121 0 4月 11 14:50 hooks/ drwxr-xr-x 1 L 197121 0 4月 11 14:50 info/ drwxr-xr-x 1 L 197121 0 4月 11 14:59 objects/ drwxr-xr-x 1 L 197121 0 4月 11 14:50 refs/ |

提示2：

可以通过git ls-files命令查看暂存区的文件信息。

参数信息如下，括号中简写：

* --cached(-c)： 查看暂存区中文件。git ls-files命令默认执行此选项。
* --midified(-m)：查看修改的文件。
* --delete(-d)：查看删除过的文件。
* --other(-o) ：查看没有被Git跟踪的文件。
* --stage(-s)：显示mode以及文件对应的Blob对象，进而我们可以获取暂存区中对应文件里面的内容。

例如：git ls-files -c或者git ls-files --cached （其他命令同理）

我们常用git ls-files -s命令查看暂存区的文件信息。

接下来，我们可以通过底层命令：update-index、write-tree、read-tree等命令，轻松创建自己的tree对象。

|  |
| --- |
| Bash # 1.查看暂存区当前状态，可以看到没有任何显示 # 说明暂存区没有存储任何文件 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s  # 2.把test.txt文件存入暂存区 # 通过git update-index命令实现 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git update-index --add --cacheinfo 100644 \ > 83baae61804e65cc73a7201a7252750c76066a30 test.txt  # 3.再次查看暂存区当前状态，可以看到暂存区中有一个文件了。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s 100644 83baae61804e65cc73a7201a7252750c76066a30 0 test.txt # 这里就说明了为什么之前要把test.txt文件先存入到本地版本库中了， # 因为需要文件的hash键，来添加到暂存区。 ### 这里也说明了文件名和文件hash键的结合。（重点） |

命令说明：

* 为创建一个树对象，首先需要通过暂存一些文件到暂存区。
* 通过底层命令 git update-index将一个单独文件存入暂存区中。
* --add 选项：因为此前该文件并不在暂存区中，一个文件首次添加到暂存区，需要使用--add 选项。
* --cacheinfo 选项：因为要添加的test.txt文件位于Git 数据库中（上一步的操作），而不是位于当前工作目录，所以需要--cacheinfo 选项。
* 最后需要指定文件模式、SHA-1 与文件名。

文件模式说明：

* 100644：表明这是一个普通文件。（blob对象的文件模式一般都为100644）
* 100755：表示一个可执行文件。
* 120000：表示一个符号链接。

继续，下面来观察生成的树对象：：

|  |
| --- |
| Bash # 4.完成上面步骤后，查看.git目录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll .git/ total 8 -rw-r--r-- 1 L 197121 130 4月 11 14:50 config -rw-r--r-- 1 L 197121 73 4月 11 14:50 description -rw-r--r-- 1 L 197121 23 4月 11 14:50 HEAD drwxr-xr-x 1 L 197121 0 4月 11 14:50 hooks/ -rw-r--r-- 1 L 197121 104 4月 11 15:39 index # 出现了index文件 drwxr-xr-x 1 L 197121 0 4月 11 14:50 info/ drwxr-xr-x 1 L 197121 0 4月 11 14:59 objects/ drwxr-xr-x 1 L 197121 0 4月 11 14:50 refs/ # 这里提示一下，暂存区Stage可以理解成是一个简单的索引文件。 # 指的就是.git/index文件。（重点）  # 5.现在先查看一下Git数据库内容，还是之前那一个blob对象。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/83/baae61804e65cc73a7201a7252750c76066a30  # 6.把暂存区中的内容提交到本地版本库 # 换句话说就是把暂存区中存放的文件索引（快照）提交到本地版本库。 # 使用write-tree命令实现 # 也就是通过write-tree命令生成树对像 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git write-tree d8329fc1cc938780ffdd9f94e0d364e0ea74f579  # 7.再次查看Git数据库内容，多了一个d8对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/83/baae61804e65cc73a7201a7252750c76066a30 .git/objects/d8/329fc1cc938780ffdd9f94e0d364e0ea74f579  # 8.查看d8对象的类型，可以看到是一个树对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -t d8329fc1cc938780ffdd9f94e0d364e0ea74f579 tree  # 9.再次查看暂存区当前状态，发现暂存区的内容没有清空 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s 100644 83baae61804e65cc73a7201a7252750c76066a30 0 test.txt # 说明：在查看暂存区，发现暂存区的内容没有清空，即：暂存区内容写到版本库，暂存区不清空。（重点） |

**4）总结**

以上就是在Git中，使用底层命令手动创建一个树对象的过程。

* 创建一个文件，把该文件通过git hash-object命令存储到本地版本库中。
* 通过git update-index命令，把文件存储到暂存区中。
* 通过git write-tree命令，把暂存区中的文件索引信息提交到本地版本库，生成了一个树对象。

**（3）创建第二个文件（重点）**

**1）新增new.txt文件，并修改test.txt文件内容**

|  |
| --- |
| Bash # 1.创建new.txt文件 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ echo "new file" > new.txt  # 2.修改test.txt文件内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ echo "version 2" >> test.txt  # 3.查看两个文件的内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ cat new.txt new file  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ cat test.txt version 1 version 2  # 4.查看工作目录中的文件 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll total 2 -rw-r--r-- 1 L 197121 9 4月 11 16:25 new.txt -rw-r--r-- 1 L 197121 20 4月 11 16:25 test.txt |

**2）将new.txt文件和test.txt文件的第二个版本添加到暂存区**

将test.txt文件添加到暂存区

|  |
| --- |
| Bash # 1.查看暂存区当前文件信息 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s 100644 83baae61804e65cc73a7201a7252750c76066a30 0 test.txt  # 2.把test.txt文件提交到本地版本库 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git hash-object -w ./test.txt 0c1e7391ca4e59584f8b773ecdbbb9467eba1547  # 3.查看Git数据库内容，可以看到又多出一个0c对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/0c/1e7391ca4e59584f8b773ecdbbb9467eba1547 .git/objects/83/baae61804e65cc73a7201a7252750c76066a30 .git/objects/d8/329fc1cc938780ffdd9f94e0d364e0ea74f579 # 提示：因上面修改了test.txt文件内容，在提交到版本库，内容不一样了hash就变了。  # 4.把修改后的test.txt文件加入暂存区 # 因为之前提交过test.txt文件到暂存区，所以不用加--add选项 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git update-index --cacheinfo 100644 \ > 0c1e7391ca4e59584f8b773ecdbbb9467eba1547 test.txt  # 5.查看暂存区当前文件信息 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s 100644 0c1e7391ca4e59584f8b773ecdbbb9467eba1547 0 test.txt # 我们可以看到，暂存区的test.txt文件被最新修改的版本覆盖了， # 之前是83开头的 # 提示：暂存区是按对应文件覆盖的，新修改的文件，覆盖之前的原文件， # 不会覆盖其他文件，即暂存区不是整体覆盖的。（重点） |

将new.txt文件添加到暂存区

|  |
| --- |
| Bash # 1.new.txt文件添加到暂存区 # 这次我们直接用一个命令把new.txt文件从工作区直接添加到暂存区 # 说明： # 因为是new.txt文件是第一次添加到暂存区，所以需要--add选项 # 因为是从new.txt文件是在工作区，所以不需要--cacheinfo选项 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git update-index --add new.txt  # 2.查看Git数据库内容，可以看到对了一个fa对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/0c/1e7391ca4e59584f8b773ecdbbb9467eba1547 .git/objects/83/baae61804e65cc73a7201a7252750c76066a30 .git/objects/d8/329fc1cc938780ffdd9f94e0d364e0ea74f579 .git/objects/fa/49b077972391ad58037050f2a75f74e3671e92  # 3.查看暂存区当前文件信息，new.txt文件已经添加到暂存区中 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s 100644 fa49b077972391ad58037050f2a75f74e3671e92 0 new.txt 100644 0c1e7391ca4e59584f8b773ecdbbb9467eba1547 0 test.txt |

说明：git update-index --add 文件名完成了之前的两步操作。

1. 把new.txt文件内容存入了Git版本库。
2. 把new.txt文件添加到了暂存区中。

**3）把暂存区的内容提交的本地版本库**

此时工作目录和暂存区中的文件状态是一样的， 可以通过git write-tree命令提交到本地版本库，生成树对像了。

|  |
| --- |
| Bash # 1.提交暂存区内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git write-tree 163b45f0a0925b0655da232ea8a4188ccec615f5  # 2.查看Git数据库内容，可以看到又多了一个名为16的tree对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/0c/1e7391ca4e59584f8b773ecdbbb9467eba1547 .git/objects/16/3b45f0a0925b0655da232ea8a4188ccec615f5 .git/objects/83/baae61804e65cc73a7201a7252750c76066a30 .git/objects/d8/329fc1cc938780ffdd9f94e0d364e0ea74f579 .git/objects/fa/49b077972391ad58037050f2a75f74e3671e92  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -t 163b45f0a0925b0655da232ea8a4188ccec615f5 tree |

此时Git版本库中的5个对象，即表示了项目的2个版本。（不明白这句话？继续往下看）

**（4）将第一个树对象加入暂存区，使其成为新的树对**

|  |
| --- |
| Bash # 1.查看暂存区当前文件信息 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s 100644 fa49b077972391ad58037050f2a75f74e3671e92 0 new.txt 100644 0c1e7391ca4e59584f8b773ecdbbb9467eba1547 0 test.txt  # 2.将第一个树对象加入暂存区 # 第一个树对象hash:d8329fc1cc938780ffdd9f94e0d364e0ea74f579 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git read-tree --prefix=bak d8329fc1cc938780ffdd9f94e0d364e0ea74f579  # 3.再次查看暂存区当前文件信息，有多了一个bak/test.txt文件 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git ls-files -s 100644 83baae61804e65cc73a7201a7252750c76066a30 0 bak/test.txt 100644 fa49b077972391ad58037050f2a75f74e3671e92 0 new.txt 100644 0c1e7391ca4e59584f8b773ecdbbb9467eba1547 0 test.txt |

说明：

* read-tree命令：可以把树对象读入暂存区。
* --prefix=bak选项：将一个已有的树对象作为子树读入暂存区。

接下来继续，再提交暂存区的内容，会继续生成一个新的tree对象在Git仓库中。

|  |
| --- |
| Bash # 把暂存区的内容生成新的tree对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git write-tree 01ab2a43b1eb150bcf00f375800727df240cf653  # 查看新生成的对象 # 查看tree对象的类型 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -t 01ab2a43b1eb150bcf00f375800727df240cf653 tree  # 查看tree对象的内容，也就是记录暂存区的内容。 # 可以看到该tree对象，包含了两个blob对象和一个tree对象。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -p 01ab2a43b1eb150bcf00f375800727df240cf653 040000 tree d8329fc1cc938780ffdd9f94e0d364e0ea74f579 bak 100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt 100644 blob 0c1e7391ca4e59584f8b773ecdbbb9467eba1547 test.txt   # 查看当前Git仓库中的对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/01/ab2a43b1eb150bcf00f375800727df240cf653 .git/objects/0c/1e7391ca4e59584f8b773ecdbbb9467eba1547 .git/objects/16/3b45f0a0925b0655da232ea8a4188ccec615f5 .git/objects/83/baae61804e65cc73a7201a7252750c76066a30 .git/objects/d8/329fc1cc938780ffdd9f94e0d364e0ea74f579 .git/objects/fa/49b077972391ad58037050f2a75f74e3671e92 |

到这里我们的演示就完成了，请看下面的总结。

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/81d9e0caeea2 来源：简书 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。