**Git原理 - Git对象[blob对象]（二）**

**（4）查看blob对象内容**

我们先用cat命令直接读取上面文件，看看是什么情况，如下图：



可以看到显示的内容是一片乱码。

**我们需要根据Hash键读取数据，使用命令git cat-file -p 键**。

-p选项可指示该命令自动判断内容的类型，并为我们使用友好的格式显示内容。

如下：

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -p cb2eb834126f53952590c448f14fece6cbb1bff3 git object test content |

|  |
| --- |
| 提示：  用cat命令直接读取Git对象文件，为什么是乱码信息？  文件内容是先通过 zlib 压缩，然后将 zlib 压缩后的内容写入磁盘文件(SHA-1 前两个字符作为子目录名称，后 38 个字符作为子目录文件的名称) |

**（5）查看Git对象的类型**

通过git cat-file -t 键命令，可以查看.git/objects 目录中Git对象的类型

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -t cb2eb834126f53952590c448f14fece6cbb1bff3 blob |

这里也说明，我们之前存储的Git对象是一个blob对象。

**（6）Git管理文件**

至此，你已经掌握了如何向 Git 中存入内容，以及如何将它们取出。

我们同样可以将这些操作应用于文件中的内容。 例如，可以对一个文件进行简单的版本控制。

**1）首先，创建一个文件**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ echo "hello-git.txt v1" > hello-git.txt  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll total 1 -rw-r--r-- 1 L 197121 17 4月 10 23:17 hello-git.txt  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ cat hello-git.txt hello-git.txt v1 |

此时文件还有被Git管理。

**2）将hello-git.txt文件存入Git数据库**

|  |
| --- |
| Bash # 加入版本库，生成blob对象 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git hash-object -w ./hello-git.txt a620c95d3001e1f64cecfc6715f9750cc7bbbf98 |

**3）查看Git数据库内容**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/a6/20c95d3001e1f64cecfc6715f9750cc7bbbf98 .git/objects/cb/2eb834126f53952590c448f14fece6cbb1bff3 |

可以看到有多了一个a6子目录，就说明有新增了一个对象。

**4）查看a6对象的内容**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -p a620c95d3001e1f64cecfc6715f9750cc7bbbf98 hello-git.txt v1 |

**5）查看a6对象的类型**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -t a620c95d3001e1f64cecfc6715f9750cc7bbbf98 blob |

可以看到，不管是你存储一个文件还是存储控制台内容到Git中，最终存储到Git数据库中的都是一个blob类型的Git对象。（即：blob对象是存储数据内容的）

**（7）Git管理修改过的文件**

**1）我们继续向hello-git.txt文件中添加内容**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ echo "hello-git.txt v2" >> hello-git.txt  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ cat hello-git.txt hello-git.txt v1 hello-git.txt v2 |

**2）查看Git数据库中的对象**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/a6/20c95d3001e1f64cecfc6715f9750cc7bbbf98 .git/objects/cb/2eb834126f53952590c448f14fece6cbb1bff3 |

可以看到还是之间的两个对象cb和a6，说明我们修改过的文件不会自动的存储到Git数据库中。

我们还需要手动的把修改后的hello-git.txt文件，存储到Git数据库中。

**3）把修改后的hello-git.txt文件添加到Git数据库中**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git hash-object -w ./hello-git.txt 7c320a2d671f2ff177063f98343a0123432521dd |

**4）再次查看Git数据库中的对象**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/7c/320a2d671f2ff177063f98343a0123432521dd .git/objects/a6/20c95d3001e1f64cecfc6715f9750cc7bbbf98 .git/objects/cb/2eb834126f53952590c448f14fece6cbb1bff3 |

我们可以看到Git数据库中多了一个7c对象。

**5）查看7c对象存储的内容**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git cat-file -p 7c320a2d671f2ff177063f98343a0123432521dd hello-git.txt v1 hello-git.txt v2 |

如上所示，我们可以看到7c对象存储了v1和v2的内容，v1内容即在a6对象中，也在7c对象中。所以对于Git来说，存储的不是文件内容的增量。

**2、blob对象总结**

* Git的核心部分是一个简单的键值数据库（key-value data store），键就是文本内容的hash，值就是文本内容。
* blob对象都存储在.git/objects 目录中，子目录+目录中的文件名，就是40位Hash值，也就是对象的键值。
* 通过这个键就能找到对应的内容。
* 每个文本内容存储到Git数据库的时候，内容都会进行zlib 压缩再存储。
* blob对象存储的是文件的内容，相同的内容不产生新的blob对象。
* blob对象并没有存储文件名。

|  |
| --- |
| 提示：Git对象的hash键，我们截取前几位就行，练习时对象不那么对，就不用全部都写，能够表示唯一对象就行。 |

**3、问题**

我们对文件做一次修改，存储到Git数据库中，都会在Git数据库中创建一个新的blob对象。而在实际的工作中，我们需要做很多的改动，才提交一个版本，我们是否可以用一个blob对象代表整个项目的一次快照。

**不能**，只能代表一次存储时，一个文件中的内容，与之前数据内容相同时不新增Git对象，数据内容不同时再次新增blob对象。即：只要有新的内容被Git纳入管理，必定有一个blob对象与之对应。

那么还有如下问题：

1. 记住文件的每一个版本所对应的SHA-1值并不现实。
2. 在blob对象中，文件名并没有被保存，仅保存了文件的内容。

所以，没有文件名就没有办法通过文件名来读取数据，只能用40位Hash值读取，非常的不现实。

解决方案：树对象。

|  |
| --- |
| 提示：以上的操作都是在工作区和本地版本库之间进行操，不涉及暂存区，因为我们直接存储到了本地版本库中。 |

**4、本文用到的命令总结**

Git底层命令：

* git hash-object -w 文件路径：把工作区的一个文件提交到本地版本库中。
* find .git/objects -type f：查看Git数据库中的对象。（Linux命令）
* git cat-file -p 键：查看该Git对象的内容。
* git cat-file -t 键：查看该Git对象的类型。

|  |
| --- |
| 参考：   * [https://www.cnblogs.com/jasongrass/p/10582465.html](https://links.jianshu.com/go?to=https%3A%2F%2Fwww.cnblogs.com%2Fjasongrass%2Fp%2F10582465.html) * [https://blog.csdn.net/songyunzu/article/details/92023816](https://links.jianshu.com/go?to=https%3A%2F%2Fblog.csdn.net%2Fsongyunzu%2Farticle%2Fdetails%2F92023816) * [https://blog.csdn.net/weixin\_39881387/article/details/111199788](https://links.jianshu.com/go?to=https%3A%2F%2Fblog.csdn.net%2Fweixin_39881387%2Farticle%2Fdetails%2F111199788) |

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/bdb75e45ea3d 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。