**Git原理 - Git对象[blob对象]（一）**

Git 是一套内容寻址文件系统。什么意思呢？

就是Git的核心部分是一个简单的键值数据库（key-value data store）。你可以向该数据库插入任意类型的内容，并会返回一个键值，通过该键值可以在任何时候再取出该内容。

**（一）Git对象的存放目录**

Git中对象都保存在本地版本库的.git/objects 目录（即：**对象数据库**）中。

首先初使化一个赶紧的Git仓库：

|  |
| --- |
| Bash # 创建一个本地的git仓库git\_learning L@DESKTOP-T2AI2SU MINGW64 /j/git-repository $ mkdir git\_learning  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository $ cd git\_learning/  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 10 20:24 ./ drwxr-xr-x 1 L 197121 0 4月 10 20:23 ../ drwxr-xr-x 1 L 197121 0 4月 10 20:24 .git/ |

确认 objects 目录是默认初始状态：

|  |
| --- |
| Bash # 查看.git/objects/目录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll -a .git/objects/ total 4 drwxr-xr-x 1 L 197121 0 4月 10 20:24 ./ drwxr-xr-x 1 L 197121 0 4月 10 20:24 ../ drwxr-xr-x 1 L 197121 0 4月 10 20:24 info/ drwxr-xr-x 1 L 197121 0 4月 10 20:24 pack/  # 查看info目录和pack目录中的内容 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll -a .git/objects/info/ total 0 drwxr-xr-x 1 L 197121 0 4月 10 20:24 ./ drwxr-xr-x 1 L 197121 0 4月 10 20:24 ../  L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll -a .git/objects/pack/ total 0 drwxr-xr-x 1 L 197121 0 4月 10 20:24 ./ drwxr-xr-x 1 L 197121 0 4月 10 20:24 ../ |

从上可以看到Git初始化一个本地版本库的时候，就已经初始化了objects目录，并在其中创建了pack和info子目录，但是没有其他常规文件，pack和info子目录中也没有文件。我们只关注objects目录下除了info和pack目录之外的变化。

**（二）Git中对象类型**

Git中对象类型有四种：blob（块）对象，tree（目录树）对象，commit（提交）对象和tag（标签）对象，这四种原子对象构成了Git高层数据结构的基础。

**（三）blob对象**

**1、blob对象说明**

**（1）blob对象定义**

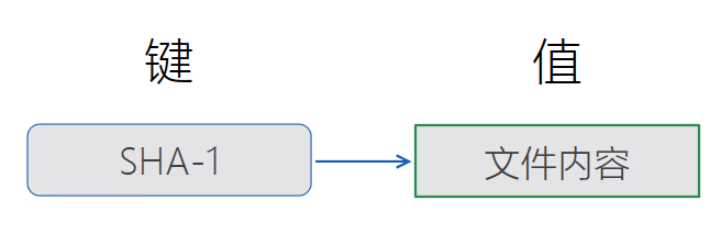
blob对象有叫数据对象。

blob对象是用来存储文本内容的。即把一个文本文件的内容，作为一个blob对象存储在Git系统中。

翻译：

* Git中blob对象就是对应文件系统中的文件，确切的说是文件的内容，包含
* 键：一个hash值和校验值的组合，
* 值：文件的内容。
* 比较特殊的是：blob对象只存内容，不存文件名，文件名在tree对象中保存。

**blob对象存储方式如下图**：



**（2）blob对象说明**

通过底层命令 git hash-object 来演示，该命令可将任意数据保存于 .git/objects 目录（即 **对象数据库**）中，并返回指向该数据对象的唯一的键。

**1）创建一个新的数据对象，并将它手动存入你的新Git数据库中：**

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ echo 'git object test content' | git hash-object -w --stdin cb2eb834126f53952590c448f14fece6cbb1bff3 |

**说明**：这是在Git中以最简单的形式存储数据到Git版本库中，git hash-object命令会接受你传给它的东西，而它只会返回，可以存储在Git仓库中数据对象的唯一的键。

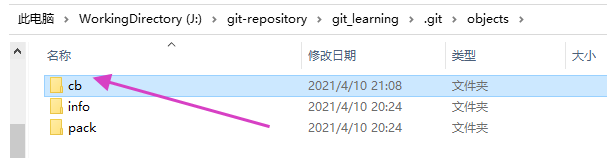
命令含义如下：

* git hash-object：Git底层命令，可以根据传入的文本内容返回表示这些内容的键值。
* git object test content：为文本文件中的内容。
* -w选项：表示hash-object命令将数据对象存储到Git数据库中；若不指定此选项，则该命令仅返回对应的键（也就是那串Hash值）
* --stdin 选项：表示该命令从标准输入（比如键盘）读取内容，若不指定此选项，则须在命令尾部给出待存储文件的路径。例如：git hash-object -w 文件路径。

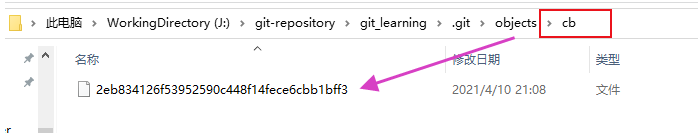
此命令输出一个长度为40个字符的校验和，这是一个SHA-1哈希值，如上cb2eb834126f53952590c448f14fece6cbb1bff3。该值是文件原内容加上特定头部信息拼接起来，做散列计算得到的数值。（只要文本内容相同，计算出的结果都是一样的）

**2）在计算机中查看本地版本库.git/objects 目录中的变化**

可以看到.git/objects 目录中多了一个cb文件夹，如下：



进入cb文件夹，可以看到有一个文件，如下：



好，我们就看到这里就行，这样看比较复杂，还是回到Git Bash中查看。

**3）看本地版本库.git/objects 目录中的变化**

|  |
| --- |
| Bash # 只看文件可以加 -type f选项参数 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects .git/objects .git/objects/cb .git/objects/cb/2eb834126f53952590c448f14fece6cbb1bff3 .git/objects/info .git/objects/pack |

我们可以看到cb目录和cb目录中的2eb834126f53952590c448f14fece6cbb1bff3文件。

这就是Git中一个blob对象的存储。

**（3）blob对象存储的方式**

Git对象的寻址使用40位的16进制数表示，也就是SHA-1散列码，例如cb2eb834126f53952590c448f14fece6cbb1bff3。

**为了管理方便，在文件系统中前两位作为.git/objects/ 子目录的名字，后38为作为文件名字。**

如下：

|  |
| --- |
| Bash # 查看objects目录中的文件，一个文件对应一个Git对象（数据内容） L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ find .git/objects -type f .git/objects/cb/2eb834126f53952590c448f14fece6cbb1bff3  # 输出数据对象的hash键，不保存该数据对象。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ echo 'git object test content' | git hash-object --stdin cb2eb834126f53952590c448f14fece6cbb1bff3 |

我们可以看到两个Hash串是一样的。

|  |
| --- |
| 提示：你可能感觉用40位作为Git对象的寻址ID，可能会存在不同的内容但是散列码相同的情况，你的感觉是正确的，但是这种情况出现的概率肯定可以忽略不计了。 |

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/6974dab17e40 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。