

赛艇队长

<2018年6月>

日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

搜索

找找看

随笔分类

Android开发(21)

git(2)

Java基础(12)

Linux(2)

Python基础(3)

多线程技术(2)

计算机基础(1)

设计模式(2)

数据库学习(5)

算法与数据结构(1)

网络编程(5)

有趣的东西(1)

正则表达式(1)

最新评论

1. Re:http协议无状态中的 "状态" 到底指的是什么? !

学习了~

--本尼

2. Re:http协议无状态中的 "状态" 到底指的是什么? !

八错八错, 讲解的很清楚, 谢楼主, 我是一直存在疑惑, 但是一直没有理解清楚, 现在比较清楚了!

--LazyCat1990

3. Re:http协议无状态中的 "状态" 到底指的是什么? !

感谢博主, 对我这种小白很受用, 这样的博文通俗易懂, 却是融入了博主一定的时间精力, 谢谢您。

--期待某一天

4. Re:以神经网络使用为例的Matlab和Android混合编程

"总结一下上面的内容: 1.直接在android app上使用deploytool生成的jar包是不行的, 因为它不能独立运行, 还需要MCR的支持, 而由于CPU的原因MCR是不能运行在android环境的....."

--月破黄昏

5. Re:在Java中谈尾递归--尾递归和垃圾回收的比较

尾递归属于尾调用, 在讲Java呢, 怎么写到python的代码了, Java中好似没有尾递归一说? JVM缺乏尾调用指令

--Subfire

阅读排行榜

1. git log命令全解析, 打log还能

博客园 首页 新随笔 联系 管理

随笔- 55 文章- 0 评论- 38

git log命令全解析, 打log还能这么随心所欲!

git log命令非常强大而好用, 在复杂系统的版本管理中扮演着重要的角色, 但默认的git log命令显示出的东西实在太丑, 不好好打扮一下根本没法见人, 打扮好了用alias命令拍个照片, 就正式出道了!

下面先详细而系统地介绍git log的所有配置知识 (用我一向简洁清晰的表述方式), 熟悉了这些东西, 你就可以自由配置自己美丽的git log了~

最后上个干货, 直接给一个我打扮好的alias配置, 懒人直接跳到最后吧!

(转载请注明: 博客园-阁刚广志, 地址: http://www.cnblogs.com/bellkosmos/p/5923439.html)

git log用于查询版本的历史, 命令形式如下:

git log [<options>] [<since>..<until>] [--] <path>...

这条命令有很多参数选项

一、不带参数

1. 如果不带任何参数, 它会列出所有历史记录, 最近的排在最上方, 显示提交对象的哈希值, 作者、提交日期、和提交说明

2. 如果记录过多, 则按Page Up、Page Down、↑、↓来控制显示

3. 按q退出历史记录列表

二、显示参数

1. -p: 按补丁显示每个更新间的差异, 比下一条- -stat命令信息更全

2. --stat: 显示每次更新的修改文件的统计信息, 每个提交都列出了修改过的文件, 以及其中添加和移除的行数, 并在最后列出所有增减行数小计

3. --shortstat: 只显示--stat中最后的行数添加修改删除统计

4. --name-only: 尽在已修改的提交信息后显示文件清单

5. --name-status: 显示新增、修改和删除的文件清单

6. --abbrev-commit: 仅显示SHA-1的前几个字符, 而非所有的40个字符

7. --relative-date: 使用较短的相对时间显示 (例如: "two weeks ago")

8. --graph: 显示ASCII图形表示的分支合并历史

9. --pretty=: 使用其他格式显示历史提交信息, 可选项有: oneline,short,medium,full,fuller,email,raw以及format:<string>,默认为medium, 如:

1. --pretty=oneline: 一行显示, 只显示哈希值和提交说明 (--oneline本身也可以作为单独的属性)

2. --pretty=format:" ": 控制显示的记录格式, 如:

1. %H 提交对象 (commit) 的完整哈希字符串

2. %h 提交对象的简短哈希字符串

3. %T 树对象 (tree) 的完整哈希字符串

4. %t 树对象的简短哈希字符串

5. %P 父对象 (parent) 的完整哈希字符串

6. %p 父对象的简短哈希字符串

7. %an 作者 (author) 的名字

8. %ae 作者的电子邮件地址

9. %ad 作者修订日期 (可以用 -date= 选项定制格式)

10. %ar 作者修订日期, 按多久以前的方式显示

11. %cn 提交者(committer)的名字

1. 作者和提交者的区别不知道是啥?

2. 作者与提交者的关系: 作者是程序的修改者, 提交者是代码提交人 (自己的修改不提交是怎么能让别人拉下来再提交的?)

3. 其实作者指的是实际作出修改的人, 提交者指的是最后将此工作成果提交到仓库的人。所以, 当你为某个项目发布补丁, 然后某个核心成员将你的补丁并入项目时, 你就是作者, 而那个核心成员就是提交者 (soga)

12. %ce 提交者的电子邮件地址

13. %cd 提交日期 (可以用 -date= 选项定制格式)

14. %cr 提交日期, 按多久以前的方式显示

15. %s 提交说明

3. 带颜色的--pretty=format:" ", 这个另外写出来分析

1 of 5

6/7/18, 10:22 AM

这么随心所欲! (14415)

2. http协议无状态中的 "状态" 到底指的是什么? ! (12396)

3. Android中的依赖问题 (五种依赖、eclipse、AS、添加第三方库、jar) (7412)

4. 在Java中谈尾递归--尾递归和垃圾回收的比较(5425)

5. 由SOAP说开去 - - 谈谈WebServices、RMI、RPC、SOA、REST、XML、JSON(4569)

评论排行榜

1. http协议无状态中的 "状态" 到底指的是什么? ! (26)

2. 华为手机EditText光标 (cursor) 颜色修改(6)

3. Android中的依赖问题 (五种依赖、eclipse、AS、添加第三方库、jar) (3)

4. 在Java中谈尾递归--尾递归和垃圾回收的比较(3)

5. Charles maplocal 时中文显示乱码问题(3)

推荐排行榜

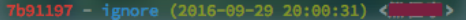
1. http协议无状态中的 "状态" 到底指的是什么? ! (26)

2. Linux文件系统详解(9)

3. git log命令全解析, 打log还能这么随心所欲! (7)

4. 由SOAP说开去 - - 谈谈WebServices、RMI、RPC、SOA、REST、XML、JSON(5)

5. 以神经网络使用为例的Matlab和Android混合编程(4)

1. 以这句为例: %Cred%h%Creset -%C(yellow)%d%Cblue %s %Cgreen(%cd) %C(bold blue)<%an>
2. 它的效果是: 
3. 先断句: [%Cred%h] [%Creset -] [%C(yellow)%d] [%Cblue%s] [%Cgreen(%cd)] [%C(bold blue)<%an>]
4. 然后就是很明显能得到的规律了
1. 一个颜色 + 一个内容

2. 颜色以%C开头, 后边接几种颜色, 还可以设置字体, 如果要设置字体的话, 要一块加个括号

1. 能设置的颜色值包括: reset (默认的灰色), normal, black, red, green, yellow, blue, magenta, cyan, white.

2. 字体属性则有bold, dim, ul, blink, reverse.

3. 内容可以是占位元字符, 也可以是直接显示的普通字符
10. --date= (relative|local|default|iso|rfc|short|raw): 定制后边如果出现%ad或%cd时的日期格式
1. 有几个默认选项

1. --date=relative: shows dates relative to the current time, e.g. "2 hours ago".

2. --date=local: shows timestamps in user's local timezone.

3. --date=iso (or --date=iso8601): shows timestamps in ISO 8601 format.

4. --date=rfc (or --date=rfc2822): shows timestamps in RFC 2822 format, often found in E-mail messages.

5. --date=short: shows only date but not time, in YYYY-MM-DD format. 这个挺好用

6. --date=raw: shows the date in the internal raw git format %s %z format.

7. --date=default: shows timestamps in the original timezone (either committer's or author's).

2. 也可以自定义格式 (需要git版本2.6.0以上), 比如--date=format:'%Y-%m-%d %H:%M:%S' 会格式化成为: 2016-01-13 11:32:13, 其他的格式化占位符如下:

1. %a: Abbreviated weekday name

2. %A: Full weekday name

3. %b: Abbreviated month name

4. %B: Full month name

5. %c: Date and time representation appropriate for locale

6. %d: Day of month as decimal number (01 - 31)

7. %H: Hour in 24-hour format (00 - 23)

8. %I: Hour in 12-hour format (01 - 12)

9. %j: Day of year as decimal number (001 - 366)

10. %m: Month as decimal number (01 - 12)

11. %M: Minute as decimal number (00 - 59)

12. %p: Current locale's A.M./P.M. indicator for 12-hour clock

13. %S: Second as decimal number (00 - 59)

14. %U: Week of year as decimal number, with Sunday as first day of week (00 - 53)

15. %w: Weekday as decimal number (0 - 6; Sunday is 0)

16. %W: Week of year as decimal number, with Monday as first day of week (00 - 53)

17. %x: Date representation for current locale

18. %X: Time representation for current locale

19. %y: Year without century, as decimal number (00 - 99)

20. %Y: Year with century, as decimal number

21. %Z, %z: Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown

22. %%: Percent sign

三、筛选参数:

1. 按数量
1. -n: 显示前n条log
2. 按日期
1. --after=

1. 比如git log --after="2014-7-1", 显示2014年7月1号之后的commit(包含7月1号)

2. 后边的日期还可以用相对时间表示, 比如"1 week ago"和"yesterday", 比如git log --after="yesterday"

3. 这里的格式可以是什么?

2. --before=

1. 同上

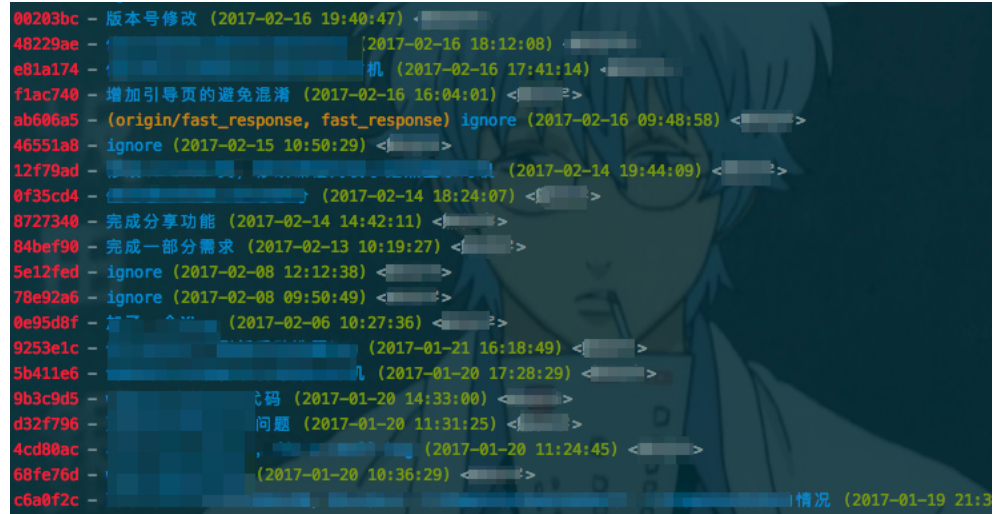
2. 另外这两条命令可以同时使用表示时间段, 比如git log --after="2014-7-1" --before="2014-7-4"

3. 另外--since --until和 --after --before是一个意思, 都可以用
3. 按作者
 1. --author=
 1. 比如git log --author="John", 显示John贡献的commit
 2. 注意: 作者名不需要精确匹配, 只需要包含就行了
 3. 而且: 可以使用正则表达式, 比如git log --author="John|Mary", 搜索Marry和John贡献的commit
 4. 而且: 这个--author不仅包含名还包含email, 所以你可以用这个搜索email
4. 按commit描述
 1. --grep=
 1. 比如: git log --grep="JRA-224"
 2. 而且: 可以传入-i用来忽略大小写
 3. 注意: 如果想同时使用--grep和--author, 必须在附加一个--all-match参数
5. 按文件
 1. -- (空格) 或 [没有]
 1. 有时你可能只对某个文件的修改感兴趣, 你只想查看跟某个文件相关的历史信息, 你只需要插入你感兴趣文件的路径 [对, 是路径, 所以经常是不太好用] 就可以了
 2. 比如: git log -- foo.py bar.py, 只返回和foo.py或bar.py相关的commit
 3. 这里的--是告诉Git后面的参数是文件路径而不是branch的名字. 如果后面的文件路径不会和某个branch产生混淆, 你可以省略--, 比如git log foo.py
 4. 另外, 后边的路径还支持正则, 比如: git log *install.md 是, 指定项目路径下的所有以install.md结尾的文件的提交历史
 5. 另外, 文件名应该放到参数的最后位置, 通常在前面加上--并用空格隔开表示是文件
 6. 另外, git log file/ 查看file文件夹下所有文件的提交记录
6. 按分支
 1. --
 1. --branchName branchName为任意一个分支名字, 查看某个分支上的提交记录
 2. 需要放到参数中的最后位置处
 3. 如果分支名与文件名相同, 系统会提示错误, 可通过--选项来指定给定的参数是分支名还是文件名
 1. 比如: 在当前分支中有一个名为v1的文件, 同时还存在一个名为v1的分支
 2. git log v1 -- 此时的v1代表的是分支名字 (—后边是空的)
 3. git log -- v1 此时的v1代表的是名为v1的文件
 4. git log v1 — v1 代表v1分支下的v1文件
7. 按内容
 1. -S"<string>"、-G"<string>"
 1. 有时你想搜索和新增或删除某行代码相关的commit. 可以使用这条命令
 2. 假设你想知道Hello, World!这句话是什么时候加入到项目里去的, 可以用: git log -S"Hello,World!"
 3. 另外: 如果你想使用正则表达式去匹配而不是字符串, 那么你可以使用-G代替-S.
 4. 这是一个非常有用的debug工具, 使用他你可以定位所有跟某行代码相关的commit. 甚至可以查看某行是什么时候被copy的, 什么时候移到另外一个文件中去的
 5. 注: -S后没有 "=", 与查询内容之间也没有空格符
8. 按范围
 1. git log <since>..<until>
 1. 这个命令可以查看某个范围的commit
 2. 这个命令非常有用当你使用branch做为range参数的时候. 能很方便的显示2个branch之间的不同
 3. 比如: git log master..feature, master..feature这个range包含了在feature有而在master没有的所有commit, 同样, 如果是feature..master包含所有master有但是feature没有的commit
 4. 另外, 如果是三个点, 表示或的意思: git log master...test 查询master或test分支中的提交记录
9. 过滤掉merge commit
 1. --no-merges
 1. 默认情况下git log会输出merge commit. 你可以通过--no-merges标记来过滤掉merge commit, git log --no-merges
 2. 另外, 如果你只对merge commit感兴趣可以使用--merges, git log --merges
10. 按标签tag
 1. git log v1.0
 1. 直接这样是查询标签之前的commit
 2. 加两个点git log v1.0.. 查询从v1.0以后的提交历史记录(不包含v1.0)
11. 按commit
 1. git log commit : 查询commit之前的记录, 包含commit
 2. git log commit1 commit2: 查询commit1与commit2之间的记录, 包括commit1和commit2
 3. git log commit1..commit2: 同上, 但是不包括commit1
 1. 其中, commit可以是提交哈希值的简写模式, 也可以使用HEAD代替

1. HEAD代表最后一次提交, HEAD^为最后一个提交的父提交, 等同于HEAD~1
2. HEAD~2代表倒数第二次提交

最后干货, 你会喜欢的~

下面第一条的效果是这样:



```
00203bc - 版本号修改 (2017-02-16 19:40:47) <[redacted]>
48229ae - [redacted] (2017-02-16 18:12:08) <[redacted]>
e81a174 - [redacted] 机 (2017-02-16 17:41:14) <[redacted]>
f1ac740 - 增加引导页的避免混淆 (2017-02-16 16:04:01) <[redacted]>
ab606a5 - (origin/fast_response, fast_response) ignore (2017-02-16 09:48:58) <[redacted]>
46551a8 - ignore (2017-02-15 10:50:29) <[redacted]>
12f79ad - [redacted] (2017-02-14 19:44:09) <[redacted]>
0f35cd4 - [redacted] (2017-02-14 18:24:07) <[redacted]>
8727340 - 完成分享功能 (2017-02-14 14:42:11) <[redacted]>
84bef90 - 完成一部分需求 (2017-02-13 10:19:27) <[redacted]>
5e12fed - ignore (2017-02-08 12:12:38) <[redacted]>
78e92a6 - ignore (2017-02-08 09:50:49) <[redacted]>
0e95d8f - [redacted] (2017-02-06 10:27:36) <[redacted]>
9253e1c - [redacted] (2017-01-21 16:18:49) <[redacted]>
5b411e6 - [redacted] (2017-01-20 17:28:29) <[redacted]>
9b3c9d5 - [redacted] 码 (2017-01-20 14:33:00) <[redacted]>
d32f796 - [redacted] 问题 (2017-01-20 11:31:25) <[redacted]>
4cd80ac - [redacted] (2017-01-20 11:24:45) <[redacted]>
68fe76d - [redacted] (2017-01-20 10:36:29) <[redacted]>
c5a0f2c - [redacted] 情况 (2017-01-19 21:3[redacted]) <[redacted]>
```



```
git config --global alias.lm "log --no-merges --color --date=format:'%Y-%m-%d %H:%M:%S' --author='
你的名字! 自己修改!' --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Cblue %s %Cgreen(%cd) %C(bold
blue)<an>%Creset' --abbrev-commit"
```

```
git config --global alias.lms "log --no-merges --color --stat --date=format:'%Y-%m-%d %H:%M:%S'
--author='你的名字! 自己修改!' --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Cblue %s %Cgreen(%cd)
%C(bold blue)<an>%Creset' --abbrev-commit"
```

```
git config --global alias.ls "log --no-merges --color --graph --date=format:'%Y-%m-%d %H:%M:%S'
--pretty=format:'%Cred%h%Creset -%C(yellow)%d%Cblue %s %Cgreen(%cd) %C(bold blue)<an>%Creset'
--abbrev-commit"
```

```
git config --global alias.lss "log --no-merges --color --stat --graph --date=format:'%Y-%m-%d
%H:%M:%S' --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Cblue %s %Cgreen(%cd) %C(bold
blue)<an>%Creset' --abbrev-commit"
```



参考资料:

<http://www.cnblogs.com/irocket/p/advanced-git-log.html>
<http://stackoverflow.com/questions/7853332/git-log-date-formats>
<https://git-scm.com/docs/git-log>

分类: git

标签: git, tag, alias

好文要顶

关注我

收藏该文



赛艇队长

关注 - 7

粉丝 - 20

+加关注

« 上一篇: Android布局尺寸思考

» 下一篇: View绘制过程理解

posted @ 2016-09-30 12:09 赛艇队长 阅读(14414) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问网站首页](#)。

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库!

【推荐】华为云7大明星产品0元免费使用

【大赛】2018首届“顶天立地”AI开发者大赛



最新IT新闻:

- 为什么一定要考大学? 马云、俞敏洪、雷军这样说
 - 通过手术“换个性格”, 了解一下?
 - Twitter拟发行六年期10亿美元可转换债券
 - Facebook股东想要废除扎克伯格超级投票权 但以失败告终
 - 路透: 腾讯与政府开发通关系统 用微信取代港澳通行证
- » 更多新闻...



最新知识库文章:

- 程序员的宇宙时间线
 - 突破程序员思维
 - 云、雾和霭计算如何一起工作
 - 你可以把编程当一项托付终身的职业
 - 评审的艺术——谈谈现实中的代码评审
- » 更多知识库文章...