

# Git

纪老师，第五阶段

联系方式： vx 18800108022

今日内容：Git

## Git是什么

Git是一款软件， 提供了 代码版本控制功能。

解决实际开发中的痛点：

- 新增代码 导致了 新的BUG， 然而你记不住改了什么
- 代码书写完毕后， 出现bug之后， 实在无法找到， 希望回退到 最近的 完美节点
- 代码的分支操作
- 代码的远程保存

## Git软件的安装

下载地址： <https://dl.softmgr.qq.com/original/Development/Git-2.29.2.3-64-bit.exe>

安装过程： 打开软件只有， 一路 **下一步** 到最后即可！

**查看已安装的git版本**

```
1 | git version
2 |
3 | 目前最新: git version 2.29.2.windows.3
```

## 初始化配置

git软件在使用之前， 必须配置 **用户名** 和 **邮箱**

两个配置命令： **最好是英文名， 中文有乱码风险**

- `git config --global user.name "用户名"`
- `git config --global user.email "邮箱"`

查看配置是否成功： 这是小写的L；      结果列表中有 `user.name` 和 `user.email` 就成功了

- `git config -l`

## 暂存功能

场景：

然然：在写代码... 本来好用的，然哥的媳妇过来玩.. 不小心碰到键盘了... 代码出错了..

此时 如何得知 哪些代码被修改了??!!

让项目被 `git` 工具管理：

在项目目录下运行cmd：执行 `git init`

快捷打开操作：在文件夹目录的 地址栏，输入 `cmd` ，回车 即可快速访问 `cmd`

VSCode会自动识别 `Git` 管理的程序，会为每个文件添加后缀：

- U: `unadd` 未暂存
- A: `add` 暂存
- C: `conflict` 冲突
- M: `modify` 修改的

## 什么是暂存

代表代码已经完成了一小部分，是一个完美的状态。 可以保存起来 类似：`ctrl+s`

命令：

- `git add 文件名`      暂存指定文件

- `git add .`          暂存所有文件

## 版本提交

类似于毕业论文的书写过程：

每当修改到一定阶段的时候，会形成一个版本：

- 基于java的物流管理系统 V1
- 基于java的物流管理系统 V2
- 基于java的物流管理系统 V3 完美版
- 基于java的物流管理系统 V3 完美版 --不再修改
- 基于java的物流管理系统 V3 完美版 --绝对不改...

版本的提交命令：

只有暂存的文件才能形成版本 -- 暂存属于一种 认可状态

即 从文件的标识来讲， U 不可以形成版本

把所有暂存的文件，形成版本：

```
git commit -m "版本说明"
```

注意：版本说明部分，cmd工具的中文有乱码可能，不允许写空格!!!

- commit: 提交

插件：Git History

可以查看版本历史~

## 版本回退

使用场景：

- 第一版本：OK
- 第二版本：OK
- 第三版本：客户拒绝.. 要求回归到 第二版本！

查看所有版本: `git log`

`commit` 后面的值是唯一标识

显示不全, 可以按方向键 ↓ 来访问更多

按键盘的 `q` 可以退出查看

回退到指定版本:

`git reset --hard` 版本的唯一标识

查看所有版本命令:

`git log` 只能查看当前版本之前的版本

`git reflog` 查看所有

## 分支功能

例如: 工作正在完成A功能... 临时来需求, 需要完成B功能

A还没做完..

分支: 类似于把项目复制一份, 然后操作克隆体, 不会对本体产生影响

团队协作时:

- A要在基础版本上新增 a功能
- B要在基础版本上新增 b功能

互相之间不能影响

此时就需要创建两个分支, A 和 B 互相的操作 互补影响

创建分支命令:

`git checkout -b` 分支名

-b: 代表创建完分支, 直接切换到新分支

查看已有所有分支:

```
git branch
```

切换到指定分支:

```
git checkout 分支名
```

## 复习

Git就是一款软件, 提供了 版本控制 功能

- `git init`
  - 把一个项目用git软件初始化. 初始化之后才能使用git管理
- `git add .` 或 `git add 文件名`
  - 暂存: 暂时存储, 可以对比新的代码 和 暂存代码的差异. 可以快速恢复到暂存的版本
- `git commit -m "版本说明"`
  - 提交版本: 当项目每进入到一个阶段, 就可以形成一个版本. 可以通过历史查看每个版本都完成了什么任务
  - 回退版本: `git reset --hard` 版本唯一标识
  - `git log`: 查看当前版本 及 之前的记录
  - `git reflog`: 查看所有版本的记录
- 分支
  - 查看所有分支: `git branch`
  - 切换到指定分支: `git checkout 分支名`
  - 创建分支并跳转: `git checkout -b 分支名`

- 创建一个 demo2 文件夹
- 在 demo2 文件夹下打开cmd. 执行git的初始化操作

```
git init
```

- 使用 vscode 打开 demo2 文件夹; 创建 01.js 02.js 03.js 文件. 不用写内容
  - 后缀符号: U 代表没有暂存过
- 提交第一个版本: **必须在demo2目录下执行cmd**
  - 暂存所有: `git add .`
  - 提交版本: `git commit -m "版本1"`

### 分支操作的场景:

#### 多人合作场景:

- 然然在 版本1 的基础上, 开发 CSS
- 亮亮在 版本1 的基础上, 开发 HTML
- 东东在 版本1 的基础上, 开发 JS

在没有git的场景下: 把版本1的代码 复制给 3个人 3份; 各自开发完毕后, 再手动进行合并

#### 创建分支: 项目下进行

- `git checkout -b ran`
- `git checkout -b liang`
- `git checkout -b dong`

#### 查看已有所有分支:

- `git branch`

当分支操作完毕后 或者 想要切换到其它分支前: 必须**提交版本** 才能把**更改固定到当前分支**

- 暂存: `git add .`
- 版本: `git commit -m "新增js"`

切换到 ran分支

- `git checkout ran`
- 查看当前分支: `git branch`
- 创建 01.css 02.css 03.css
- 暂存+版本:
  - `git add .`
  - `git commit -m "新增css"`

切换 liang 分支

- `git checkout liang`
- 查看是否成功切换: `git branch`
- 新建: 01.html 02.html 03.html
- 稳固版本:
  - `git add .`
  - `git commit -m "新增html"`

合并操作:

场景:当开发完毕之后, 就可以合并代码; 我们仅支持 把**其它分支** 合并到 **当前分支**

主分支: master

如果想要把 dong 分支合并给 master 主分支, 则**必须确保** master 为**当前分支**

- 确保当前分支为 master 主分支: `git checkout master`
- 查看当前: `git branch`
- 合并 dong 分支的内容到**当前分支**: `git merge dong`
- 合并 liang : `git merge liang`
- 合并 ran: `git merge ran`

合并命令: `git merge 分支名`

可以把 分支 合并到 当前分支

## 冲突的解决

目前：

- 主分支有 01.js
- 其它分支 也有01.js

此时：

- 主分支修改了 01.js 的 第一行代码 并 提交版本
  - `git add .`
  - `git commit -m "修改01.js"`
- 其它分支 也修改了 01.js 的第一行代码 并提交版本
  - 切换到 ran: `git checkout ran`
  - 修改 01.js 的第一行代码： 写一串2 就可以，然后保存；
  - 提交版本：
    - `git add .`
    - `git commit -m "修改01.js"`
- 合并分支
  - 把当前分支切换成 主分支: `git checkout master`
  - 查看当前分支: `git branch`
  - 合并ran分支: `git merge ran`

```
1 | Auto-merging 01.js
2 | CONFLICT (content): Merge conflict in 01.js
3 | Automatic merge failed; fix conflicts and then commit the result.
4 |
5 | 自动合并失败，因为有冲突。 请 手动解决冲突之后 再提交版本！
```

- 打开vscode 查看 01.js 的样子
  - 文件右侧有小图标： C 代表 conflict， 是冲突的意思
- 手动解决冲突之后： 暂存+提交版本



- `git add .`
- `git commit -m "解决01.js的冲突"`

## 删除分支

分支用完之后，就可以删除

```
git branch -d 分支名
```

-d: delete删除

- 查看所有: `git branch`
- 删除 ran 分支: `git branch -d ran`
- 查看所有: `git branch`
- 如上操作: 删除 liang 和 dong 分支

## 远程仓库

相当于是网盘

- 可以保存代码到 网络仓库
  - 便于 在 家里 加班!
  - 团队合作

两个非常有名的 Git 免费仓库网站: **两个都要申请!**

- 码云 : 国内的 <https://gitee.com/>
  - 优点: 速度快, 全中文界面
- Github: 世界的 <https://github.com/> 注册按钮: Sign Up
  - 优点: 知名度高. 对于国内来讲!
  - 缺点: 速度慢, 全英文

对于简历书写: Github 显得高大上... 实际应用推荐 **码云**

仓库修改权限分两种：

- 个人仓库：只有你自己能修改代码
- 组织仓库：
  - 免费：允许5个人 修改代码
  - 付费：不限制

仓库的查看权限有两种：

- 私有：只有自己能看
- 公开：大家都能看

## 新建仓库

登录之后：右上角 + 新建仓库 .， 仓库名随便起

- 是否开源： 私有 只有自己看， 公开 所有人都能看
- 其它选项：随意
- 创建即可

远程仓库 需要下载到本地才能使用，下载方式有两种

- https：需要账号和密码 才能下载 私有仓库
  - 公开仓库 不需要账号密码
- ssh：需要设定密钥，以后就再也不需要账号密码

克隆远程仓库的命令：

`git clone 仓库远程地址`

大概格式如下： 具体地址要参考 你自己的

`git clone https://gitee.com/736907613/web2009.git`

此代码在哪个文件夹下执行，就会下载远程仓库 到哪个文件夹下

```
D:\Web2009\Git>git clone https://gitee.com/736907613/web2009.git
Cloning into 'web2009'...
warning: You appear to have cloned an empty repository.
D:\Web2009\Git>
```

你好像克隆了一个空的仓库： 正常！ 因为仓库是刚刚创建的

注意事项：git有保存账号密码的功能，如果第一次输入账号密码之后，下一次就不用输入；

如果你修改了 码云的账号密码，然而本地保存的是之前的；此时就会报错！



## 上传操作

- 用vscode 打开 clone 到本地的文件夹
- 创建 01.js 里面随意写字， 保存即可
- 暂存+提交版本： **注意， 命令行操作的是克隆的文件夹**
  - `git add .`
  - `git commit -m "新增01.js"`
- 把版本上传到远程仓库： `git push`

```
D:\Web2009\Git\web2009>git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 219 bytes | 219.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Powered by GITEE.COM [GNK-5.0]
To https://gitee.com/736907613/web2009.git
 * [new branch]      master -> master

D:\Web2009\Git\web2009>
```

- 到码云，刷新你的仓库， 看是不是有了！

## 更新远程到本地

在网上操作远程仓库 新增文件：

实际的使用场景：

在公司 push 了代码； 回家以后需要把网上的代码 更新到家里的电脑上



在项目下，执行：

```
git pull
```

此方式 已经可以团队合作了；但是下载仓库时 需要公用同一个账号密码！

## 团队方式 组织

允许5个人 修改同一个项目：使用各自的账号密码



新建仓库:



Github

操作方式与码云完全一致!

## 学习的网站

---

**哔哩哔哩：**非常有名的免费技术分享网站

更多git知识点 都可以在这个网站搜索