

总复习

安排

- 周三: 总复习
- 周四/周五: 考试和项目答辩
 - 重点: 项目!!!
 - 项目要部署在自己的服务器上!!!
- **项目展示:部署在自己服务器!!!!**
 - <https://www.django.cn/article/show-4.html>
- 收集每个人项目--->能在我电脑上跑起来!!!
 - **项目: 完整的用户模块, 任务管理小系统**
 - **必备1: 项目源码**
 - **必备2: 项目依赖的所有第三方库(文档列表)**
 - 如何导出自己项目依赖的第三方库???
 - 建议:做项目时.一个项目单独独立一个工作空间!!!
 - **必备3: 数据库(mysql数据库倒出来)**

复习: git的工作流程/分支

- git工作流程

```
1 # 前提: 项目经理,把项目框架上传到服务器的远程库/共有库!
2 经理发git地址: https://github.com/经理名/1901two_03.git
3
4 # 普通员工
5 # 1. Fork 一份到自己的git仓库
6 git@github.com:自己名字/1901two_03.git
7 # 2. 克隆到本地
8 git clone git@github.com:自己名字/1901two_03.git
9 # 3. 同步分支(看是否有其他分支)
10 查看本地是否有dev分支: `git branch`
11 # 创建分支指令
12 git checkout -b dev # 创建并切换到dev分支! = 创建 + 切换
13 自己创建一个开发分支: 保证主分支上的代码都是完全正确的! 日常开发不操作主分支!
14 功能测试都正常, 才会合并dev代码到主分支上!
15
16 # 4. dev上修改项目!
17 add....commit, 测试正行 marge 指令合并! 提交到远程git库!!
18 # 5. 更新/解决冲突
19 先拉取: git pull
20 解决冲突
```

服务器项目应该也有两个分支: `master`(永远保证代码都是正确的可运行的!) / `dev` (项目都提交到`dev`上,)

完整工作流程

经理

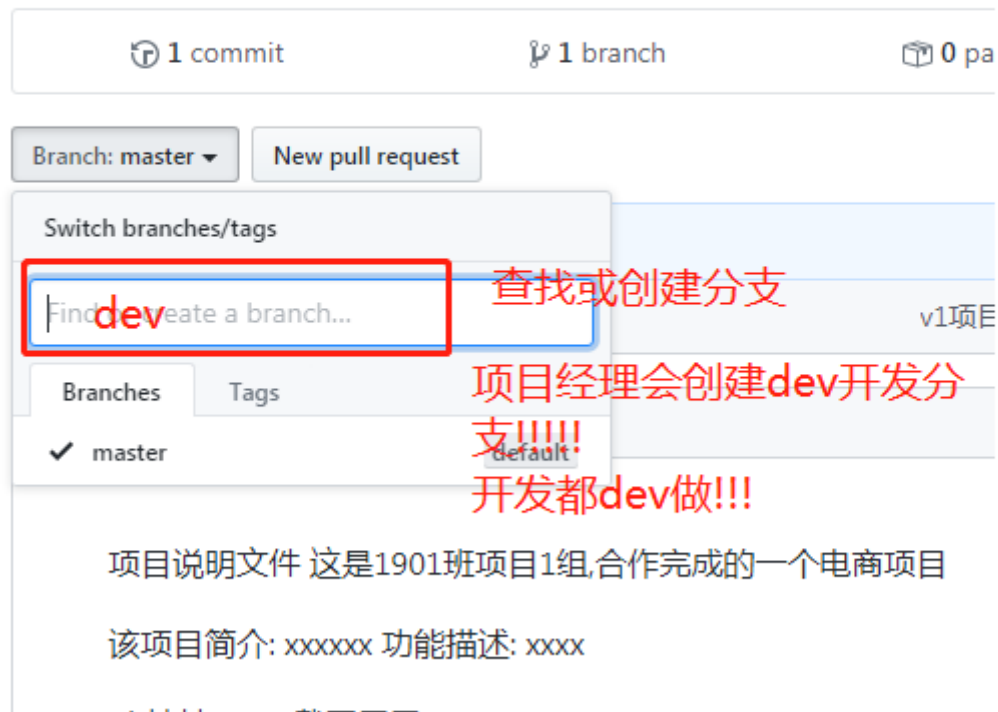
- 1. 新建远程仓库(空仓库) `shop1901`
- 2. 创建本地仓库/项目 `shop1901`

```
1 git init
2 git add README.md
3 git commit -m "first commit"
4 git remote add origin git@github.com:askyang/shop1901.git
5 git push -u origin master
```

- 3. 创建`dev`分支(不影响主分支!保证主分支永远是正确可上线运行的版本!)

1901班电商框架

[Manage topics](#)



- 4. 发`git`地址给开发人员:

`git@github.com:askyang/shop1901.git` askyang是经理

普通员工

- 1. Fork 一份到自己github仓库

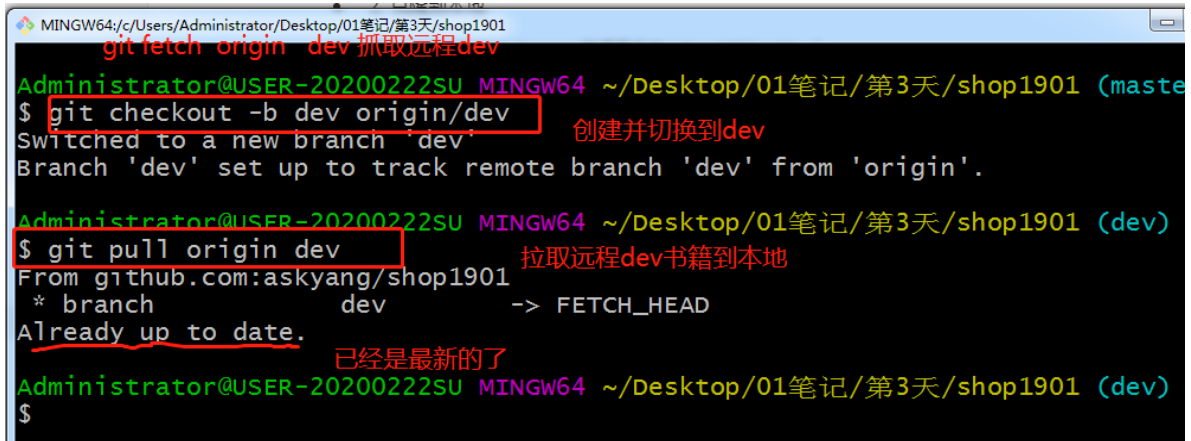
```
git@github.com:普通员工名/shop1901.git
```

- 2. 克隆到本地

```
git clone git@github.com:普通员工名/shop1901.git
```

- 3. 同步分支

```
1 # 查看本地是否有dev分支
2 git branch
3 # 没有就抓取 (fetch抓取)
4 git fetch origin dev
5 # 创建并切换到dev
6 git checkout -b dev origin/dev
7 # 拉取数据到本地
8 git pull origin dev
```



The screenshot shows a Windows command prompt window titled 'MINGW64: c:/Users/Administrator/Desktop/01笔记/第3天/shop1901'. The user is in the 'dev' branch. The following commands and their outputs are shown with red annotations:

```
Administrator@USER-20200222SU MINGW64 ~/Desktop/01笔记/第3天/shop1901 (master)
$ git checkout -b dev origin/dev
Switched to a new branch 'dev'
Branch 'dev' set up to track remote branch 'dev' from 'origin'.
Administrator@USER-20200222SU MINGW64 ~/Desktop/01笔记/第3天/shop1901 (dev)
$ git pull origin dev
From github.com:askyang/shop1901
* branch dev -> FETCH_HEAD
Already up to date.
```

Annotations in red:

- `git fetch origin dev` 抓取远程dev
- `git checkout -b dev origin/dev` 创建并切换到dev
- `git pull origin dev` 拉取远程dev书籍到本地
- `Already up to date.` 已经是最新了

- 4. 干活(修改项目)

创建自己的模块!!

- 5. 在dev上开发

增加到暂存区--->提交给git版本库-----> 测试正常----->合并到本地master主分支一份/ 提交到远程dev

```
git add .
```

```
git commit -m '提示'
```

测试没有问题--->提交到远程!!! master正确的! dev是最新的没有经过详细测试的!

- 6. 先更新

```
1 # 先拉取: git pull
2 # 解决冲突
3 # 再上传 git push origin 分支名!
```

我们可以直接查看readme.txt的内容：

```
Git is a distributed version control system.
Git is free software distributed under the GPL.
Git has a mutable index called stage.
Git tracks changes of files.
<<<<<< HEAD
Creating a new branch is quick & simple.
=====
Creating a new branch is quick AND simple.
>>>>>> feature1
```

如果大家改了同一个文件
会冲突!
删除了乱码,重新提交一份!

Git用<<<<<<, =====, >>>>>>标记出不同分支的内容,我们修改如下后保存:

```
Creating a new branch is quick and simple.
```


- 7. 提交


```
Administrator@USER-20200222SU MINGW64 ~/Desktop/01笔记/第3天/s
$ git push origin dev 开发分支提交代码
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 375 bytes | 375.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:askyang/shop1901.git
    50aa89f..a02e809 dev -> dev
```


🔗 dev (1 minute ago)


Branch: dev ▼ New pull request

This branch is 1 commit ahead of master.

 askyang 周正杨购物车模块完成

 zhouzhengyang 周正杨购物车模块

 README.md 正式版master中,测试通过后再提交!!!

 README.md

- 8. 发稳定版给项目经理

master- 分支中都是稳定版

dev-----分支中都是开发版!!!

自己的稳定版---->发到领导的dev上!!!!

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If

The screenshot shows the GitHub 'Comparing changes' interface. At the top, there are two dropdown menus: 'base: dev' and 'compare: master'. Both are highlighted with red boxes. A red arrow points from the text '领导的dev' (Leader's dev) to the 'base: dev' dropdown, and another red arrow points from the text '自己的稳定版!!' (My stable version!!) to the 'compare: master' dropdown. Below these is a modal window titled 'Choose a base ref'. It has a search bar with the placeholder text 'Find a branch'. Below the search bar are two tabs: 'Branches' and 'Tags'. Under the 'Branches' tab, there is a list of branches: 'master' (marked as 'default') and 'dev' (which has a checkmark and is circled in red). To the right of the modal, there is a text snippet that reads 'There isn't any dev and ma:'. At the bottom left of the modal, there is a '+ Show' button.

领导:项目稳定版---->开发版

自己: 稳定版---->开发版

自己: 稳定版---->提交---->领导的开发板!!

最稳定和最安全的方案!!!!!! 保证master的稳定!!!