Tedu.cn 达内教育

项目综合

PROJECT DAY01

Tedu.cn 达内教育

目录 | CONTENTS

01

GIT概述和配置

GIT简介

GIT概述

Git 是一个开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。

Git 是Linus Torvalds 为了帮助管理Linux内核开发而开发的一个开放源码的版本控制软件。

你也许会想，为什么Linus不把Linux代码放到版本控制系统里呢？不是有CVS、SVN这些免费的版本控制系统吗？因为Linus坚定地反对CVS和SVN，这些集中式的版本控制系统不但速度慢，而且必须联网才能使用。有一些商用的版本控制系统，虽然比CVS、SVN好用，但那是付费的，和Linux的开源精神不符。

相关概念

- 什么是版本控制?
  - ✓ 版本控制（Revision control）是维护工程蓝图的标准作法，能追踪工程蓝图从诞生一直到定案的过程。此外，版本控制也是一种软件工程技巧，借此能在软件开发的过程中，确保由不同人所编辑的同一代码文件案都得到同步。

相关概念（续1）

- 集中式
  - ✓ 集中式版本控制系统，版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始操作，再把自己的活推送给中央服务器。集中式版本控制系统有一个缺点就是必须联网才能工作，遇到网速慢的话，可能提交就比较慢。

相关概念（续2）

- 分布式
  - ✓ 分布式版本控制系统没有“中央服务器”，每个人的电脑上都是一个完整的版本库，你工作的时候，就不需要联网了，因为版本库就在你自己的电脑上。开发者之间的协作方式因着 Git 的分布式特性而变得更为灵活多样。在集中式系统上，每个开发者就像是连接在集线器上的节点，彼此的工作方式大体相像。

GIT特点

GIT更加适合个人开发，管理开源代码。

GIT是分布式管理，不同于svn的集中式管理。这是GIT和其它非分布式的版本控制系统最核心的区别。

GIT支持更强大的分支功能

GIT的内容完整性要优于SVN：GIT的内容存储使用的是SHA-1哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。

GIT把内容按元数据方式存储，而SVN是按文件：所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn,.cvs等的文件夹里。

GIT安装配置

GIT安装

最早Git是在Linux上开发的，很长一段时间内，Git也只能在Linux和Unix系统上跑。不过，慢慢地有人把它移植到了Windows上。现在，Git可以在Linux、Unix、Mac和Windows这几大平台上正常运行了。（到那时windows 还是 SVN 使用比较广泛）

Linux 平台

```
sudo apt-get install git
```

mac os

```
brew install git
```

GIT配置

Git 提供了一个叫做 git config 的工具，专门用来配置或读取相应的工作环境变量。这些环境变量，决定了Git在各个环节的具体工作方式和行为。

在做基本操作之前必须要进行这些基本配置。这些配置变量可以存放在以下三个不同的地方：

1. /etc/gitconfig 文件：系统中对所有用户都普遍适用的配置。若使用 git config 时用 --system 选项，读写的就是这个文件。
2. ~/.gitconfig 文件：用户目录下的配置文件只适用于该用户。若使用 git config 时用 --global 选项，读写的就是这个文件。

Git配置(续1)

3. 当前项目的 Git 目录中的配置文件（也就是工作目录中的 `.git/config` 文件）：也就是git config 不加任何选项，这样的配置仅仅针对当前项目有效。

每一个级别的配置都会覆盖上层的相同配置，所以 `.git/config` 里的配置会覆盖 /etc/gitconfig 中的同名变量。

用户名

```
git config --global  user.name  [yourname]
```

邮箱

```
git config --global  user.email  [youremail]
```

GIT配置(续2)

编辑器设置

```
git config --global  core.editor  [youreditor]
```

检查已有的配置信息

```
git config  -list
```

忽略文件

在git仓库中有时候不需要把所有文件都进行协同操作。创建.gitignore文件，把忽略的文件名添加进去，这样在同步的时候就不会自动上传。

小结

- 了解GIT的基本概念
- 掌握GIT的配置方法

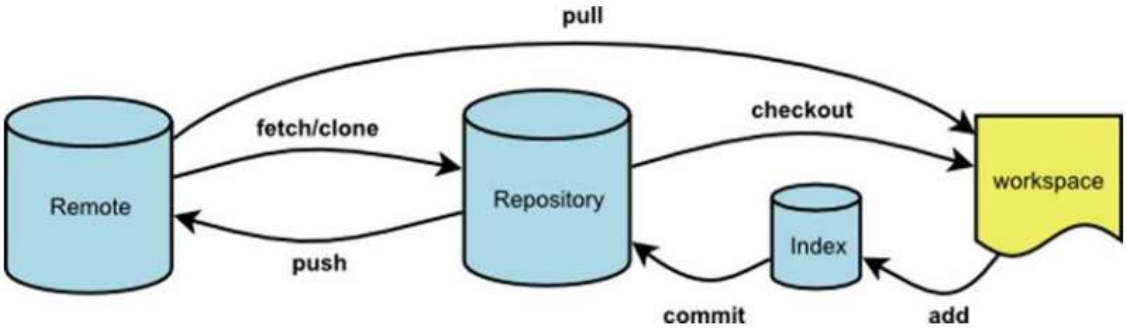
02

基本命令和版本控制

GIT管理结构

GIT结构

- 工作区：项目所在操作目录，实际操作项目的区域
- 暂存区：用于记录工作区的工作（修改）内容
- 仓库区：用于备份工作区的内容
- 远程仓库：远程主机上的GIT仓库



基本命令

基本命令

Tedu.cn 达内教育

- 初始化git仓库

```
git  init
```

- 查看当前分支状态

```
git  status
```

- 将文件提交到暂存区

```
git  add   file1 file2
```

- 删除暂存区中的文件

```
git rm --cached file
```

基本命令（续）

- 文件的移动删除

```
git mv test/file.config  .
```

- 将暂存区内容保存提交

```
git commit  -m  'some message'
```

- 查看提交日志

```
git log
```

```
git log --pretty=oneline  每个之日只显示一行
```

- 查看当前文件和已提交内容的差别

```
git  diff  file
```

版本控制

版本控制

回到上一个版本

```
git reset  --hard  HEAD^
```

回到上一个版本就一个 ^ 回到上两个版本就两个^^

超过10 就 HEAD~10

直接通过commit_id 前7位调到某个版本

先通过 git log 可以查看commit_id

```
git reset  --hard  7764c27
```

版本控制(续)

那么如何去往较新的版本

当退回到某个版本后，比其新的版本log就无法通过git log查看 此时使用git reflog查看，然后通过

```
git reset --hard  去往指定版本
```

标签操作

如果你达到一个重要的阶段，并希望永远记住那个特别的提交快照，你可以使用 git tag 给它打上标签。Git的标签虽然是版本库的快照，但其实它就是指向某个commit的指针，这跟分支很像，但是分支可以移动，标签不能移动，所以，创建和删除标签都是瞬间完成的。

选择要打标签的分支后进行如下操作：

```
git  tag  v1.0
```

打一个v1.0的标签，默认打在最新的commit上

```
git  tag  v0.9    6224637
```

打在对应的commit_id上

标签操作（续）

查看标签

```
git  tag  查看标签
git  show   v1.0  查看标签commit的具体内容
```

还可以创建带有说明的标签，用-a指定标签名，-m指定说明文字：

```
git tag -a v0.1 -m "version 0.1 released"3628164
git tag -d v0.1  删除标签
git  reset  --hard    v1.0  回复到v1.0标签状态
```

小结

- 掌握GIT的基本使用命令
- 掌握GIT版本控制方法

03

保存工作区和分支管理

保存工作区

保存工作区

- 临时保存工作区
  - git stash
- 查看现有保存的工作区
  - git stash list
- 恢复到某个工作区
  - git stash apply stash@{2}
- 恢复并删除上一个工作区
  - git stash pop

创建临时保存工作区（续）

- 删除指定的工作区
  - git stash drop stash@{0}
- 清除工作区
  - git stash clear

GIT分支命令

什么是分支

几乎每一种版本控制系统都以某种形式支持分支。使用分支意味着你可以从开发主线上分离开来，然后在不影响主线的同时继续工作。假设你准备开发一个新功能，但是需要两周才能完成，第一周你写了50%的代码，如果立刻提交，由于代码还没写完，不完整的代码库会导致别人不能干活了。如果等代码全部写完再一次提交，又存在丢失每天进度的巨大风险。现在有了分支，你创建了一个属于你自己的分支，别人看不到，还继续在原来的分支上正常工作，而你在自己的分支上干活，想提交就提交，直到开发完毕后，再一次性合并到原来的分支上，这样，既安全，又不影响别人工作。

分支命令

查看当前分支 有 `*` 的为当前作用分支

```
git branch
```

创建一个叫dev的分支

```
git branch dev
```

切换到 dev分支

```
git checkout dev
```

以上两个命令 可以用 `git checkout -b dev` 完成。

分支命令（续）

合并dev分支到当前分支　快速合并

```
git  merge dev
```

删除dev分支

```
git branch  -d  dev
```

强行删除分支

```
git branch -D dev
```

分支冲突

冲突问题是合并分支过程中最为棘手的问题：

- 当分支合并时，原分支和以前发生了变化就会产生冲突
- 当合并分支时添加新的模块（文件），这种冲突可以自动解决，只需自己决定commit操作即可。
- 当合并分支时两个分支修改了同一个文件，则需要手动解决冲突。

小结

- 掌握工作区的用途和保存方法
- 掌握分支的使用方法
- 理解什么是冲突

04

远程仓库和GitHub使用

GitHub介绍

GitHub介绍

GitHub是一个面向开开源及私有软件项目的托管平台，因为只支持git作为唯一的版本库格式进行托管，故名gitHub。

实际上github就可以看做是一个大的开源仓库。相对于我们的个人主机而言可以视作一个远程仓库。

地址：www.github.com

远程仓库

添加远程仓库

添加远程仓库(这是一条指令)

```
git remote add origin
tarena@127.0.0.1:/home/tarena/gitrepo/project.git
```

clone远程仓库

将项目克隆到本地

```
git clone tarena@127.0.0.1:/home/tarena/gitrepo/project.git
```

推送代码

向远程仓库推送代码

```
git push -u origin master
```

注：第一次推送代码时加 -u 选项

推送标签到远程仓库

```
git push origin --tags  推送所有标签
git push origin v1.0  推送指定标签
```

推送代码（续）

删除远程分支

```
git push  -u  origin :dev_Jame
```

删除标签

```
git push  origin --delete tag v1.0
```

拉取代码

获取新的分支和标签

```
git fetch origin
```

获取更新的代码

```
git pull
```

与git pull相比git fetch相当于是从远程获取最新版本到本地，但不会自动merge。如果需要有选择的合并git fetch是更好的选择。效果相同时git pull将更为快捷。

小结

- 掌握远程仓库操作命令
- 掌握GitHub的用法

05

软件项目开发

项目开发流程

需求分析

Tedu.cn 达内教育

1. 确定用户的真实需求，项目的基本功能
2. 确定项目的整体难度和可行性分析
3. 需求分析文档，用户确认

概要设计

1. 确定功能模块
2. 进行可行性分析，搭建整体架构图
3. 确定技术思路和使用框架
4. 形成概要文档指导开发流程

项目计划

1. 确定开发工作的先后顺序
2. 确定时间轴 ， 事件里程碑
3. 人员分工
4. 形成甘特图和思维导图等辅助内容

详细设计

1. 形成详细设计文档 ： 思路，逻辑流程，
2. 功能说明，技术点说明，数据结构说明
3. 重点代码说明

编码测试

1. 编写代码
2. 技术攻关
3. 代码基本测试

项目测试

1. 根据业务逻辑组织项目测试
2. 完成测试报告，提交研发修改
3. 完善项目细节

项目发布和后期维护

1. 项目交付用户使用
2. 编写项目说明文档
3. 对项目进行后期维护
4. 项目的迭代升级

项目开发注意事项

注意事项

Tedu.cn 达内教育

1. 按时完成项目和项目时间不足之间的冲突
2. 项目实施人员之间的冲突
3. 抓大放小，不要过于追求完美

小结

- 了解项目的基本开发流程
- 了解项目开发过程中的一些主要矛盾

06

在线词典(一)

功能说明

功能说明

Tedu.cn 达内教育

用户可以登录和注册

- 登录凭借用户名和密码登录
- 注册要求用户必须填写用户名，密码，其他内容自定
- 用户名要求不能重复
- 要求用户信息能够长期保存

功能说明（续1）

可以通过基本的图形界面print以提示客户端输入。

- 程序分为服务端和客户端两部分
- 客户端通过print打印简单界面输入命令发起请求
- 服务端主要负责逻辑数据处理
- 启动服务端后应该能满足多个客户端同时操作

功能说明（续2）

客户端启动后即进入一级界面，包含如下功能：登录 注册 退出

- 退出后即退出该软件
- 登录成功即进入二级界面，失败回到一级界面
- 注册成功可以回到一级界面继续登录，也可以直接用注册用户进入二级界面

功能说明（续3）

用户登录后进入二级界面，功能如下：查单词、历史记录、注销

- 选择注销则回到一级界面
- 查单词：循环输入单词，得到单词解释，输入特殊符号退出单词查询状态
- 历史记录：查询当前用户的查词记录，要求记录包含name word time。可以查看前10条均即可。

小结

- 运用所学分析方法对项目分析
- 完成数据库搭建工作

总结 | SUMMARY

- **掌握git的使用方法**
  - ✓ 什么是git，什么是GitHub，如何使用git工作
- **掌握软件项目的基本开发流程**
  - ✓ 软件项目开发流程，软件项目开发注意事项
- **分析在线词典项目**
  - ✓ 进行项目分析，搭建数据库