

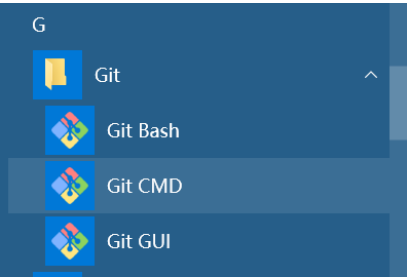
Git安装与学习报告

由于我使用的是windows平台，所以git的安装和学习也是基于windows下的。

Git安装

首先是官网下载git，链接是<https://git-scm.com/download/win>，下载完成之后默认安装就好了。

安装完成之后在开始菜单中找到Git文件夹，点击里面的Git Bash，



然后会弹出一个小窗口，那就说明安装成功了。



安装完成之后还需要最后一步设置，因为Git是分布式版本控制系统，所以需要填写用户名和邮箱作为一个标识，在刚刚弹出的小窗口命令行中输入如下：

```
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

这里git config命令的--global参数表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

这样Git就安装完成了。

Git基本使用学习

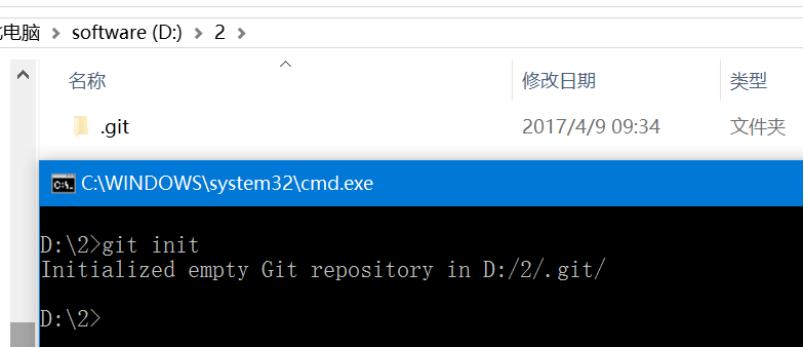
创建版本库

版本库又名仓库，英文名repository，可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

因为我使用的是windows系统，所以最简单的创建一个版本库的方式是：

- 先在你需要的位置建立一个文件夹；
- 在这个文件夹所在位置调出cmd命令行；
- 在命令行中输入: **git init**

git init可以把这个目录变成Git可管理的仓库。之后你会在这个文件夹下看到多了一个.git目录（这个目录是隐藏的），在命令行中提示这是一个空的仓库：



把文件添加到版本库

假如说要把一个readme.txt文件添加进去(这个文件所在目录必须是在我们刚刚创建的版本库中)，那么，

第一步：

```
git add readme.txt
```

第二步：

```
git commit -m "提交readme.txt"
```

如图:

```
D:\2>git add readme.txt

D:\2>git commit -m "提交readme.txt"
[master (root-commit) 7788a2d] 提交readme.txt
1 file changed, 1 insertion(+)
 create mode 100644 readme.txt

D:\2>
```

这样就添加成功了。

另外使用:

```
git status
```

可以查看当前仓库文件的状态, 比如说未添加的(红字显示), 未提交的(绿字显示)

git的一些常用的命令

```
git diff readme.txt // 查看readme.txt这个文件修改了什么内容
git log // 查看历史记录, 可以显示从最近到最远的日志信息
git reset --hard HEAD^ // 将当前版本回退到上一个版本
git reset --hard HEAD~10 // 将当前版本回退到前10个版本
git checkout -- readme.txt // 丢弃工作区的修改(工作区就是你在电脑上看到的目录)
```

远程仓库

首先去GitHub上注册一个账号, 这样自然就有了GitHub远程仓库。

第一步: 创建SSH Key. 打开Git Bash,

```
$ ssh-keygen -t rsa -C "youremail@example.com"
```

一路回车默认就好了(当然需要把右键地址换成你自己的右键地址)

可以在用户主目录里找到.ssh目录, 里面有id_rsa和idrsa.pub两个文件, 这两个就是SSH Key的密钥对, id_rsa是私钥, idrsa.pub是公钥。

第二步: 登录GitHub, setting -> SSH and GPG keys -> new SSH key. title任意填写, 在key文本框中粘贴id_rsa.pub文件中的内容, 点击add key就好了。

添加远程仓库


首先登录你的GitHub账号, 在右上方点击一个"+", 点击new repository,跳转到如下界面:

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

 SYSU-Winter ▾ /

Great repository names are short and memorable. Need inspiration? How about **musical-carnival**.

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾





Add a license: **None** ▾



Create repository

填写仓库名称点击创建就好了:

Quick setup — if you've done this kind of thing before

 Set up in Desktop or  HTTPS  SSH 

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

然后在本地的仓库下运行如下命令：


```
D:\2>git remote add origin https://github.com/SYSU-Winter/testGitHub.git


D:\2>git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 270 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/SYSU-Winter/testGitHub.git
 * [new branch]      master -> master


D:\2>
```

然后就可以在远程仓库中看到和本地仓库一样的东西了：

Branch: master ▼ New pull request

 SYSU-Winter 提交readme.txt

 readme.txt 提交readme.txt

 readme.txt

12345

使用git clone命令可以将远程仓库克隆到本地：

```
git clone https://github.com/SYSU-Winter/testGitHub.git
```

创建和合并分支

```
git checkout -b dev // 创建并切换分支dev
git branch // 查看分支，会列出所有的分支，当前分支前带星号
git merge dev // 《首先要切换回master分支》，在master分支上合并dev分支内容
git branch -d dev // 删除dev分支
```

Git基本常用命令如下：

mkdir: XX(创建一个空目录 XX指目录名)

pwd: 显示当前目录的路径。

git init 把当前的目录变成可以管理的git仓库，生成隐藏.git文件。

git add XX把xx文件添加到暂存区去。

git commit -m "XX" 提交文件 -m 后面的是注释。

git status 查看仓库状态

git diff XX 查看XX文件修改了那些内容

git log 查看历史记录

git reset --hard HEAD^ 或者 git reset --hard HEAD~ 回退到上一个版本

(如果想回退到100个版本，使用git reset --hard HEAD~100)

cat XX 查看XX文件内容

git reflog 查看历史记录的版本号id

git checkout -- XX 把XX文件在工作区的修改全部撤销。

git rm XX 删除XX文件

git remote add origin https://github.com/tugenhua0707/testgit 关联一个远程库

git push -u(第一次要用-u 以后不需要) origin master 把当前master分支推送到远程库

git clone https://github.com/tugenhua0707/testgit 从远程库中克隆

git checkout -b dev 创建dev分支 并切换到dev分支上

git branch 查看当前所有的分支

git checkout master 切换回master分支

git merge dev 在当前的分支上合并dev分支

git branch -d dev 删除dev分支

git branch name 创建分支

git stash 把当前的工作隐藏起来 等以后恢复现场后继续工作

git stash list 查看所有被隐藏的文件列表

git stash apply 恢复被隐藏的文件，但是内容不删除

git stash drop 删除文件

git stash pop 恢复文件的同时 也删除文件

git remote 查看远程库的信息

git remote -v 查看远程库的详细信息

git push origin master Git会把master分支推送到远程库对应的远程分支上

BUG管理系统：Bugzilla

在Windows平台下安装使用Bugzilla。

安装bugzilla需要的软件有MySQL数据库软件，activeperl软件，bugzilla安装包，IIS组件

安装MySQL数据库

（由于我的电脑先前已经装好了MySql, 所以这里只简单介绍一下安装过程）

到官网<https://dev.mysql.com/downloads/installer/>下载MySQL installer. 下载完成之后，安装基本可以默认安装，当然你可以选择更改安装目录。安装完成之后会有配置界面，选择“configuration type”的时候选择“Standard Configuration”，然后在下一个页面选中“include BinDirectory in Windows PATH”自动配置环境变量，单击next，会出现一个页面在此页面中设置root用户密码，并选中enable root access from remote machines，并且选中create an anonymous account 选项创建一个匿名用户，最后点击execute就配置完成了。

之后到开始菜单中找到“MySQL Command line Client”，点击出现命令行窗口，输入之前配置时创建的密码，之后下面在MySQL服务器中创建一个bugs数据库，和一个bugs用户，以及为该用户授予相应的权限，命令如下：

```
create database bugs; //创建一个数据库bugs

create user bugs@localhost ; //创建一个用户bugs

grant all on bugs.* to bugs@'localhost'; //为用户bugs授权

flush privileges; //刷新用户权限
```

输入“quit”退出，这样MySQL就配置完成了。

安装activeperl

（perl之前已完成安装）

下载：<https://www.perl.org/get.html>

perl安装过程也是一路默认。

安装Bugzilla

下载：<http://www.bugzilla.org/>

将安装包解压到想要安装的位置，我这里是解压到D盘根目录

然后使用bugzilla自带的checksetup.pl来安装所需的perl模块，如图，红字包围的是必须安装的模块：

```

COMMANDS TO INSTALL OPTIONAL MODULES:

    Chart: ppm install Chart
    Template-GD: ppm install Template-GD
    MIME-tools: ppm install MIME-tools
    XML-Twig: ppm install XML-Twig
    PatchReader: ppm install PatchReader
    perl-ldap: ppm install perl-ldap
    Authen-SASL: ppm install Authen-SASL
    Net-SMTP-SSL: ppm install Net-SMTP-SSL
    RadiusPerl: ppm install RadiusPerl
    SOAP-Lite: ppm install SOAP-Lite
    XMLRPC-Lite: ppm install XMLRPC-Lite
    JSON-RPC: ppm install JSON-RPC
    Test-Taint: ppm install Test-Taint
    HTML-Scrubber: ppm install HTML-Scrubber
    Encode-Detect: ppm install Encode-Detect
    Email-Reply: ppm install Email-Reply
HTML-FormatText-WithLinks: ppm install HTML-FormatText-WithLinks
    TheSchwartz: ppm install TheSchwartz
    Daemon-Generic: ppm install Daemon-Generic
    mod_perl: ppm install mod_perl
Apache-SizeLimit: ppm install Apache-SizeLimit
    File-MimeInfo: ppm install File-MimeInfo
    IO-stringy: ppm install IO-stringy
Cache-Memcached: ppm install Cache-Memcached

COMMANDS TO INSTALL REQUIRED MODULES (You *must* run all these commands
and then re-run checksetup.pl):

    ppm install TimeDate
    ppm install DateTime
    ppm install DateTime-TimeZone
    ppm install Template-Toolkit
    ppm install Email-Sender
    ppm install Email-MIME
    ppm install File-Slurp
搜狗拼音输入法 全 :N-XS
    ppm install DateTime-TimeZone-Local-Win32
*** Installation aborted. Read the messages above. ***

```

在当前窗口一条条复制以上命令即可。

安装完成之后在bugzilla目录下会生成一个localconfig文件，打开它，将其中的\$dbport = 0;改为\$dbport = 3306; \$indexhtml = 0;改为\$indexhtml = 1;

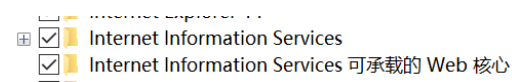
在命令行下再次运行checksetup.pl将会生成和数据库有关的数据表，

生成数据表后会要求填入主机的地址服务器地址，

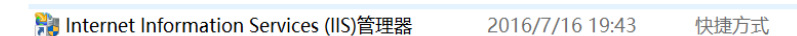
管理员名字和账号（该账号是一个email地址）以及管理员登陆的密码.和确认密码

配置IIS

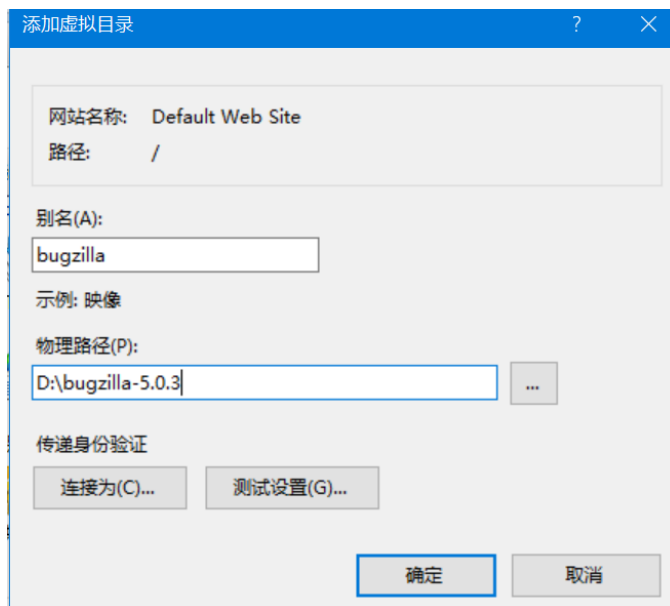
win+R下输入OptionalFeatures, 对下图中两个选项全部打钩:



之后在控制面板中的管理工具中找到IIS:



之后在Default Web Site下新建虚拟目录bugzilla,其中的物理路径选择bugzilla的安装目录:



添加虚拟目录

网站名称: Default Web Site
路径: /

别名(A):
bugzilla

示例: 映像

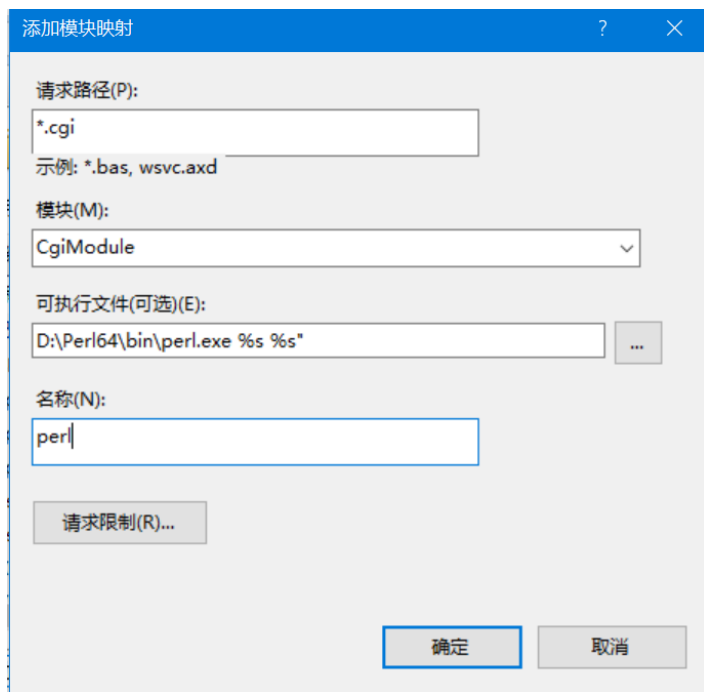
物理路径(P):
D:\bugzilla-5.0.3

传递身份验证

连接为(C)... 测试设置(G)...

确定 取消

然后先选中Default Web Site, 选中模块, 右边配置本机模块, 选中CgiModule,点击确定, 然后回到bugzilla, 双击处理程序映射, 添加模块映射:



添加模块映射

请求路径(P):
*.cgi

示例: *.bas, wsvc.axd

模块(M):
CgiModule

可执行文件(可选)(E):
D:\Perl64\bin\perl.exe %s %s

名称(N):
perl

请求限制(R)...

确定 取消

点击确认。

然后回去选择默认文档, 添加index.cgi.

基于Jenkins和Github的持续集成环境安装与学习

持续集成（Continuous integration）简称CI，是一种软件开发的实践，可以让团队在持续集成的基础上收到反馈并加以改进，不必等到开发的后期才寻找和修复缺陷。当然要明白的是持续集成环境的搭建也不是一劳永逸的，随着软件项目复杂度的增加，持续集成的环境同样要加以维护以确保集成环境的可靠性。

持续集成的重要要素：

1. 统一的代码库；
2. CI服务器；
3. 自动化测试和构建的脚本；
4. Slaves

持续集成的流程：CI服务器控制持续集成的整个过程，轮询代码库更新，根据预定义脚本进行项目的构建，服务器将任务分配到Slave端，这就是整个持续集成的过程。我们需要根据我们的项目需求，制定好一个完善的持续集成方案，然后根据方案选择CI服务器和版本管理软件。这里选择Jenkins+Github的持续集成环境。

在官网下在Jenkins安装包：<https://jenkins.io/download/>

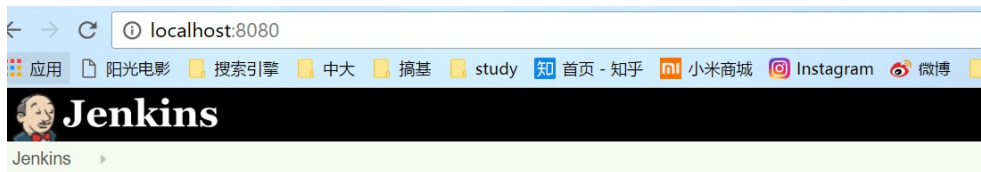
安装过程默认安装即可。

启动方式其实有好几种，我这里由于以前安装了java SDK，所以直接启动，也可以通过tomcat启动。

安装完成之后，在浏览器中输入

localhost:8080

可以显示Jenkins页面实现管理和配置



New Item

People

Build History

Manage Jenkins

My Views

Credentials

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

32484af88bd483ef8eadbd211ca248887ac18790

学习使用

配置Git

一般来说git插件都是有自动安装好的，在安装好插件之后，依次点击Manage Jenkins -> Global Tool Configuration;

Git

Git installations

Git

Name

Default

Path to Git executable

D:\Program Files\Git\bin\git.exe

☐ Install automatically

Delete Git

Add Git

新建job

Enter an item name

test666

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system or something other than software build.

就选“freestyle”，点击ok

配置

勾选github project

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Project name

test3

Description

[Plain text] [Preview](#)

☐ Discard old builds

☒ GitHub project

Project url

<https://github.com/SYSU-Winter/testGitHub/>

Advanced...

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

<https://github.com/SYSU-Winter/testGitHub.git>

Credentials

Winter@me

Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

Add Branch

构建

构建可以选择bat批处理或者shell脚本处理，配置好之后就可以点击Build now进行构建了