Git 和 BUG 管理系统 BugFree
安装和学习报告

前言

Git 是一个快速、可扩展的分布式版本控制系统，它具有极为丰富的命令集，对内部系统提供了高级操作和完全访问。Git 与你熟悉的大部分版本控制系统的差别是很大的。也许你熟悉 Subversion、CVS、Perforce、Mercurial 等等，他们使用“增量文件系统”（Delta Storage systems），就是说它们存储每次提交(commit)之间的差异。Git 正好与之相反，它会把你的每次提交的文件的全部内容（snapshot）都会记录下来。理论上，Git 可以保存任何文档，但是最善于保存文本文档，因为它本来就是为解决软件源代码（也是一种文本文档）版本管理问题而开发的，提供了许多有助于文本分析的工具。对于非文本文档，Git 只是简单地为其进行备份并实施版本管理。所以，通过使用 Git，不仅避免了版本回溯问题，还提高了团队的协作效率。

BugFree 是一款基于 WEB 的 Bug 管理系统，配置安装简单，只需到网上获取安装包，再配下 PHP 通用的环境即可；纯功能型的界面就无所谓美观；没有直接的截图功能但是可以以附件的形式存在；也有简单的报表统计功能；整体使用还是比较容易上手，而且是开源免费中文版的 BUG 管理系统。

1. Git 的安装和学习

1.1 Git 的安装

1.1.1 安装环境

Ubuntu Linux

1.1.2 安装命令

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

```
sql@ubuntu:~$ sudo apt-get install git
```

1.1.3 安装结果

```
sql@ubuntu:~$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
   tag        Create, list, delete or verify a tag object signed with GPG

'git help -a' and 'git help -g' lists available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

1.1.4 设置名字和 Email 地址

```
sql@ubuntu:~/Desktop/git_learning$ git config --global user.email suyj5@mail2.sy
su.edu.cn
sql@ubuntu:~/Desktop/git_learning$ git config --global user.name Eugene
```

注意 git config 命令的--global 参数，用了这个参数，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。

1.2 Git 的学习

1.2.1 创建版本库

什么是版本库呢？版本库又名仓库，英文名 **repository**，你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”。

首先，在本地创建一个空目录：

```
sql@ubuntu:~$ cd Desktop
sql@ubuntu:~/Desktop$ mkdir git_learning
sql@ubuntu:~/Desktop$ cd git_learning
sql@ubuntu:~/Desktop/git_learning$ pwd
/home/sql/Desktop/git_learning
sql@ubuntu:~/Desktop/git_learning$
```

然后，通过 `git init` 命令把这个目录变成 Git 可以管理的仓库：

```
sql@ubuntu:~/Desktop/git_learning$ git init
Initialized empty Git repository in /home/sql/Desktop/git_learning/.git/
sql@ubuntu:~/Desktop/git_learning$ ls
sql@ubuntu:~/Desktop/git_learning$ ls =ah
ls: cannot access =ah: No such file or directory
sql@ubuntu:~/Desktop/git_learning$ ls -ah
.  ..  .git
```

Git 把空仓库建好后，当前目录下会多了一个.git 的目录，这个目录是 Git 来跟踪管理版本库的，如果没有看到.git 目录，那是因为这个目录默认是隐藏的，用 `ls -ah` 命令就可以看见。

1.2.2 把文件添加到版本库

在新建的 Git 空仓库 git_learning 里新建一个 txt 文件，编辑好后，用命令 `git add` 告诉 Git，把文件添加到仓库：

example.py ×

```
#!/usr/bin/python

import sys

print 'Hello World!'
```

```
sql@ubuntu:~/Desktop/git_learning$ git add example.py
sql@ubuntu:~/Desktop/git_learning$ ls
example.py  example.py~
```

用命令 `git commit` 告诉 Git，把文件提交到仓库:

```
sql@ubuntu:~/Desktop/git_learning$ git commit -m "wrote an example.py"
[master (root-commit) 29e3628] wrote an example.py
 1 file changed, 5 insertions(+)
 create mode 100644 example.py
```

`git commit` 命令执行成功后会告诉你，1 个文件被改动（我们新添加的 example.py 文件），插入了 5 行内容（example.py 有 5 行内容）。

1.2.3 git status 和 git diff

当对 example.py 进行过修改后：

example.py ×

```
#!/usr/bin/python

import sys

print 'Hello!'

print 'Change the example.py'
```

使用 git status 查看可以知道 example.py 有没有修改过：

```
sql@ubuntu:~/Desktop/git_learning$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   example.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        example.py~

no changes added to commit (use "git add" and/or "git commit -a")
```

然后用 git diff 来查看哪里被修改了：

```
sql@ubuntu:~/Desktop/git_learning$ git diff
diff --git a/example.py b/example.py
index 2b4d475..9fe25c8 100644
--- a/example.py
+++ b/example.py
@@ -2,4 +2,6 @@

 import sys

-print 'Hello World!'
+print 'Hello!'
+
+print 'Change the example.py'
```

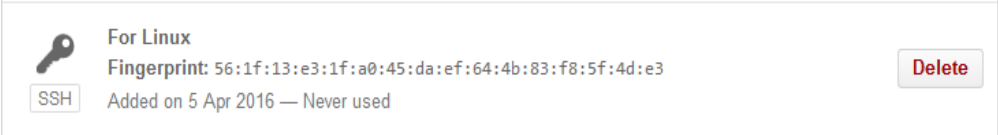
在查看过修改内容后就可以用 git add 和 git commit 命令将修改后的 example.py 提交到仓库里了。

```
sql@ubuntu:~/Desktop/git_learning$ git add example.py
sql@ubuntu:~/Desktop/git_learning$ git commit -m example.py
[master f135be6] example.py
 1 file changed, 3 insertions(+), 1 deletion(-)
sql@ubuntu:~/Desktop/git_learning$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        example.py~

nothing added to commit but untracked files present (use "git add" to track)
```

1.2.4 远程仓库

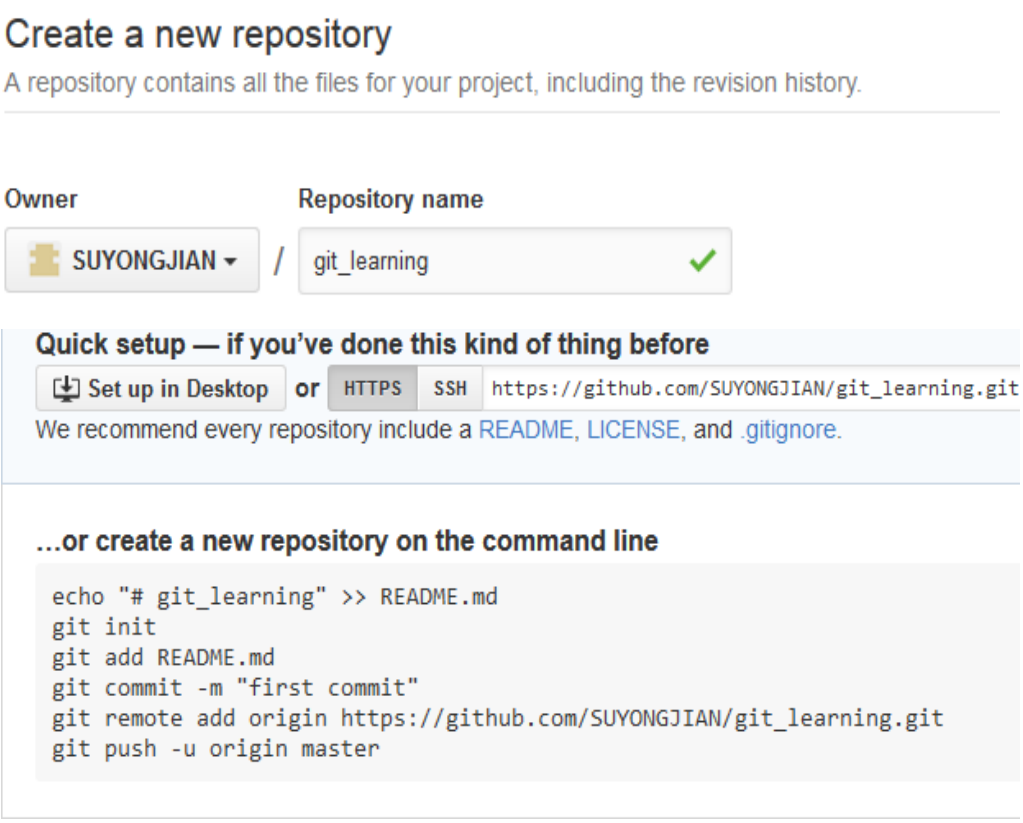
这个世界上有个叫 GitHub 的神奇的网站，从名字就可以看出，这个网站就是提供 Git 仓库托管服务的，所以，只要注册一个 GitHub 账号，就可以免费获得 Git 远程仓库。

连接到 GitHub：



a. 添加远程库

在 GitHub 上添加新的 repository：

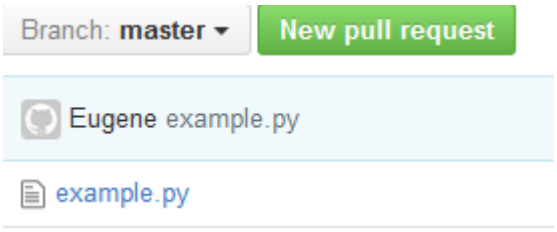


根据 GitHub 的提示，在本地的 learngit 仓库下运行命令：

```
sql@ubuntu:~/Desktop/git_learning$ git remote add origin https://github.com/SUYO
NGJIAN/git_learning.git
```

将本地的内容 push 到远程库：git push

```
sql@ubuntu:~/Desktop/git_learning$ git push -u origin master
Username for 'https://github.com': SUYONGJIAN
Password for 'https://SUYONGJIAN@github.com':
Counting objects: 6, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 555 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/SUYONGJIAN/git_learning.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```



b. 从远程库克隆

在 GutHub 上新建 repository，名字为 clone；
在本地克隆远程库：git clone

```
sql@ubuntu:~/Desktop/git_learning$ git clone git@github.com:SUYONGJIAN/clone.git

Cloning into 'clone'...
The authenticity of host 'github.com (192.30.252.131)' can't be established.
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.252.131' (RSA) to the list of know
n hosts.
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
Checking connectivity... done.
sql@ubuntu:~/Desktop/git_learning$
```

1.2.5 分支管理

a. 创建与合并分支

首先，我们创建 dev 分支，然后切换到 dev 分支：

```
sql@ubuntu:~/Desktop/git_learning$ git checkout -b dev
Switched to a new branch 'dev'
```

```
sql@ubuntu:~/Desktop/git_learning$ git branch
* dev
  master
```

修改并提交 example.py 后，切换回 master 分支：

```
sql@ubuntu:~/Desktop/git_learning$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
```

查看 example.py 文件，刚才添加的内容不见了！因为那个提交是 dev 分支上，而 master 分支此刻的提交点并没有变；

example.py ×

```
#!/usr/bin/python

import sys

print 'Hello!'

print 'Change the example.py'
```

现在，我们把 dev 分支的工作成果合并到 master 分支上：

```
sql@ubuntu:~/Desktop/git_learning$ git merge dev
Updating f135be6..c732b5b
Fast-forward
 example.py | 2 ++
 1 file changed, 2 insertions(+)
```

删除 dev 分支：`git branch -d dev`

```
sql@ubuntu:~/Desktop/git_learning$ git branch -d dev
Deleted branch dev (was c732b5b).
sql@ubuntu:~/Desktop/git_learning$ git branch
* master
```

b. 解决冲突

当在分支上和主支上同样的地方同时修改或添加了内容后，合并分支可能会出现错误，要手动修正冲突后才能合并。

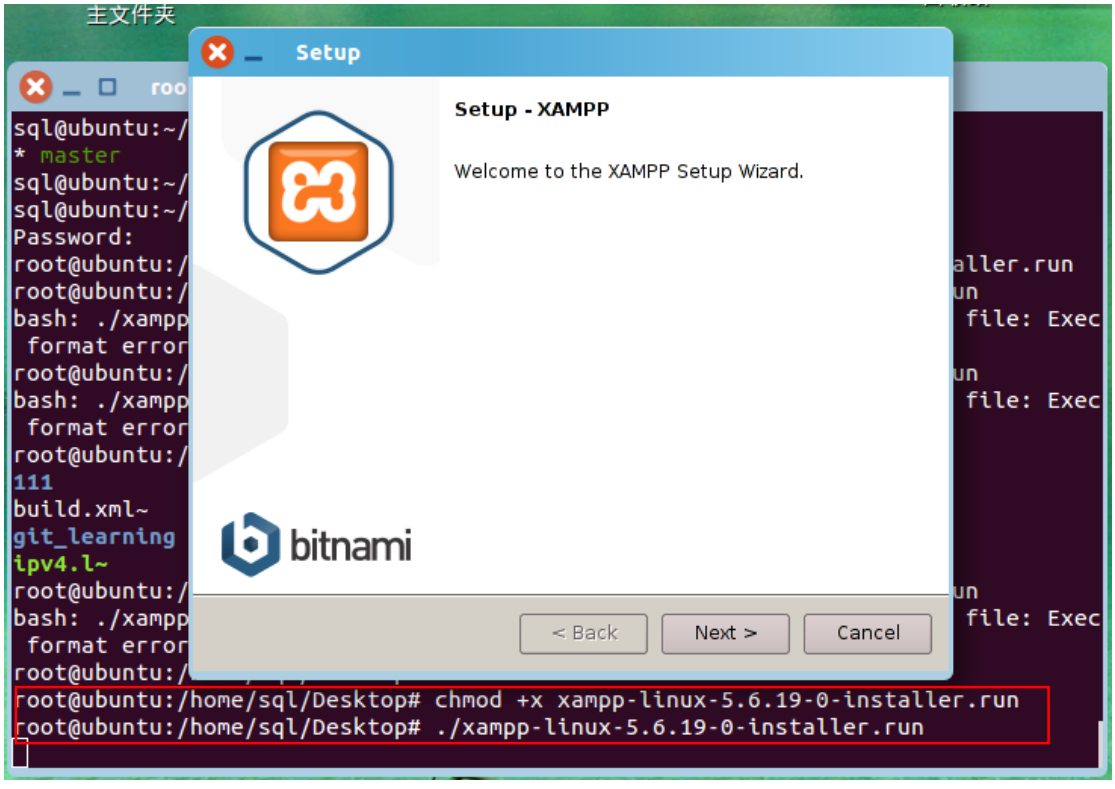
用 `git log --graph` 命令可以看到分支合并图。

2. BugFree 的安装和学习

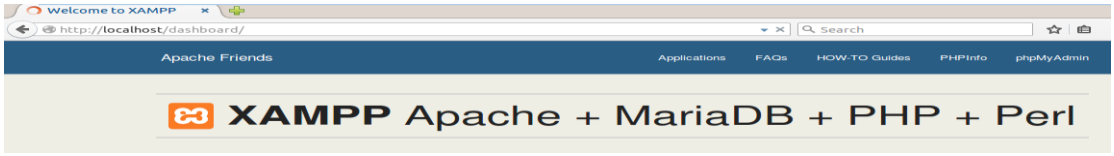
2.1 BugFree 的安装

a. 安装 BugFree 前需要部署配置 PHP, Apache Http Server, MySQL 环境。可以使用 XAMPP, EASYPHP 等集成环境快速部署。这里我们下载并安装 xampp：xampp-linux-1.8.1.tar.gz（下载链接 http://www.apachefriends.org/zh_cn/xampp-linux.html ）。

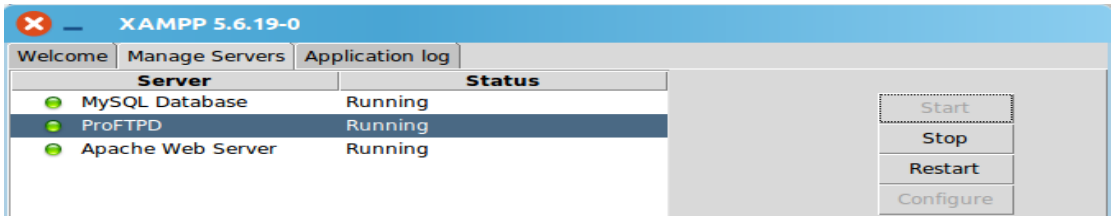
下载 xampp-linux-5.6.19-0-installer.run



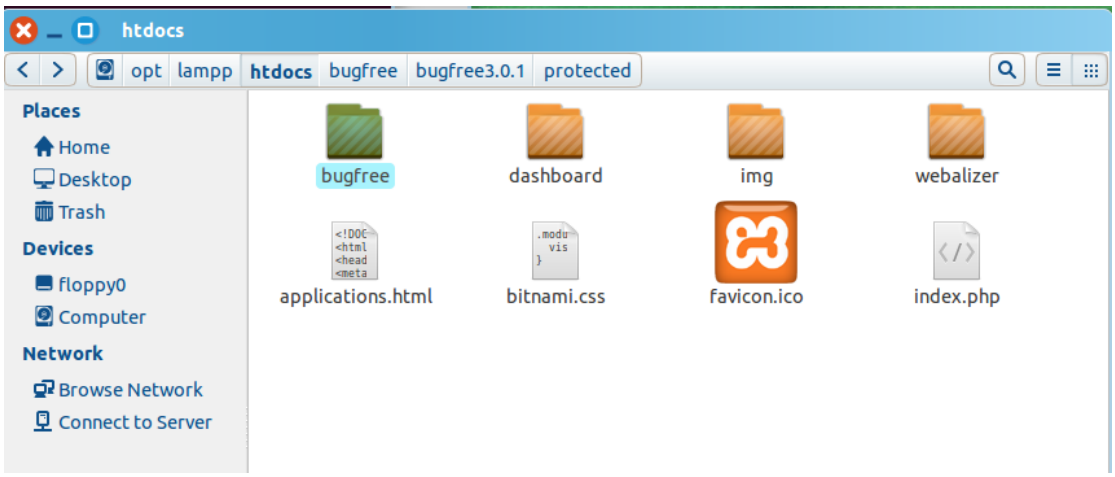
安装完后在浏览器上打开 http://localhost 验证是否成功



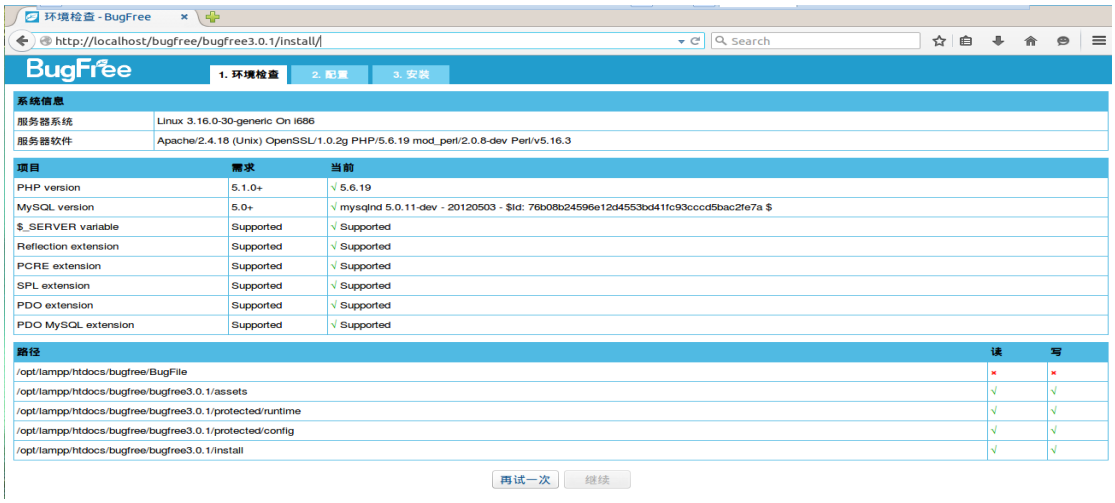
开启 Apache 等服务：



下载 bugfree 压缩包，解压后放在 xampp 的子目录 htdocs 下：



在浏览器上打开 http://localhost/bugfree/bugfree3.0.1/install (与你的文件路径有关)



要创建 BugFile 文件，所有路径需要系统给予权限 （chmod）：

路径	读	写
/opt/lampp/htdocs/bugfree/BugFile	√	√
/opt/lampp/htdocs/bugfree/bugfree3.0.1/assets	√	√
/opt/lampp/htdocs/bugfree/bugfree3.0.1/protected/runtime	√	√
/opt/lampp/htdocs/bugfree/bugfree3.0.1/protected/config	√	√
/opt/lampp/htdocs/bugfree/bugfree3.0.1/install	√	√

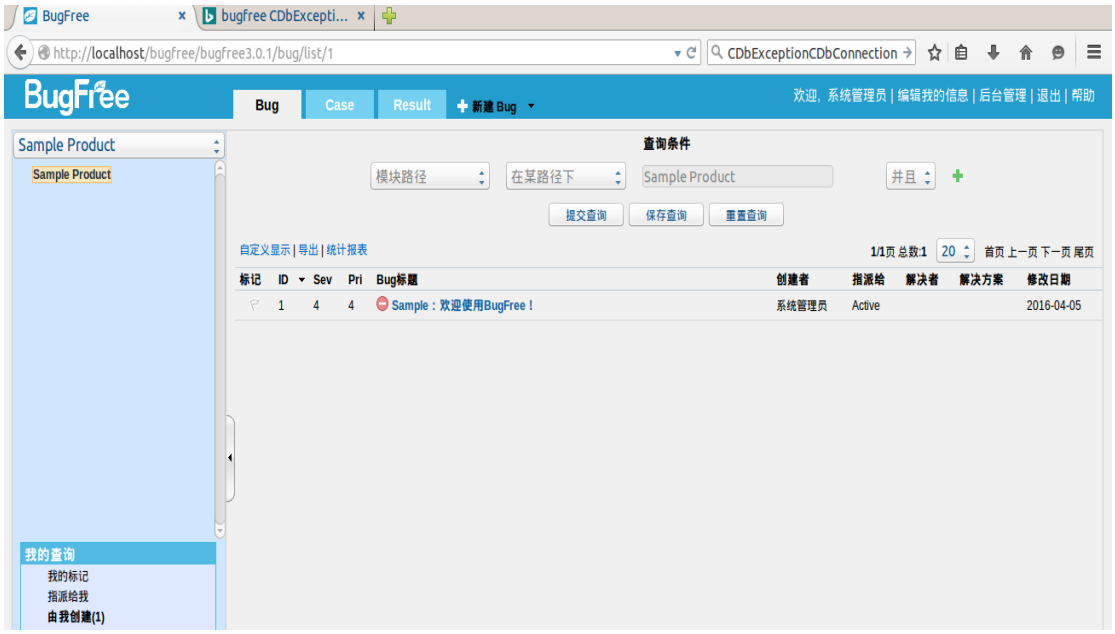
配置数据库：

安装成功

2.2 BugFree 的学习

2.2.1 主界面

进入 Bugfree （使用 admin 和 123456 登录）主界面



① 项目选择框：可快速切换当前项目，项目模块框和查询结果框显示相应的模块结构和记录

② 项目模块框：显示当前项目的模块结构。点击某一模块，查询结果框会显示所选模块的所有记录
③ 个性显示框：
1. 我的标记：显示我标记的记录
2. 指派给我：显示指派给我的记录
3. 由我创建：显示由我创建的记录
4. 抄送给我：显示抄送给我的记录
④ 模式切换标签：切换 Bug，Case，Result 模式。默认登陆为 Bug 模式
⑤ 新建：Bug，Case 新建
⑥ 查询框：设置查询条件
⑦ 查询结果框：显示当前查询的结果
1. 自定义显示：设置查询结果的显示字段
2. 导出：将当前查询结果记录导出到网页
3. 导入：导入测试测试结果
4. 统计报表：显示当前查询结果的统计信息
⑧ 导航栏：显示当前登录用户名、编辑我的信息、后台管理（管理员登录显示）、退出等

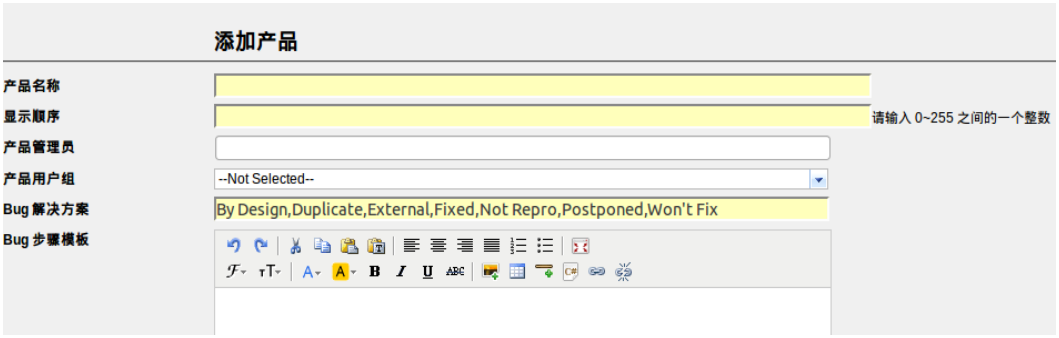
2.2.2 后台管理

2.2.2.1 Bug 管理员

	系统管理员	项目管理员	用户组管理员
项目管理	1.可添加产品 2.可查看和编辑所有产品 3.可修改产品名称和显示顺序 4.可指派产品用户组 5.可指派产品管理员 6.可编辑产品模块	1.不可添加产品 2.仅可查看和编辑自己是产品管理员的产品 3.不可修改产品名称和显示顺序 4.可指派产品用户组 5.不可指派产品管理员 6.可编辑产品模块	无权限
用户管理	1.可查看所有用户 2.可添加用户 3.可编辑、禁用或激活所有用户	1.可查看所有用户 2.可添加用户 3.可编辑、禁用或激活所有用户 或本人	1.可查看所有用户 2.可添加用户 3.可编辑、禁用或激活所有用户 或本人
用户组管理	1.可查看所有用户组 2.可添加用户组 3.可编辑或删除所有用户组	1.可查看所有用户组 2.可添加用户组 3.可编辑或删除自己添加的用户组	1.可查看所有用户组 2.可添加用户组 3.可编辑或删除自己添加的用户组或自己是用户组管理员的组

2.2.2.2 项目管理

a. 添加产品



b. 点击”模块”可添加产品模块

c.d.e.f.....

总之，待 Bugfree 安装好后，所有的操作都可以在主界面上完成，还是很简单的。其他深入的学习就不再这里赘述。