**Github团队开发示例（一）**

作者：[Grey](http://www.cnblogs.com/greyzeng/)

原文地址：http://www.cnblogs.com/greyzeng/p/6044837.html

我们可以在[Github](https://github.com/)上管理自己的团队项目。团队成员通过[Github](https://github.com/)进行协同工作，也方便管理，接下来我提供一种Github团队建立仓库的目录格式，并对每个目录进行必要说明，最后根据这个目录结构写了一个简单的例子。

**团队仓库目录结构**

* TestOrgName
  + xxxsystem
    - configs
    - dbscripts
    - docs
    - src

下面一一说一下这些目录的作用：

**TestOrgName**

**团队/组织名**，可以自定义，最好用英文。

**xxxsystem**

**团队要做的项目名称**，可以自定义，最好用英文。

**configs**

在开发的时候，项目是部署在开发环境中的，所以项目中的一些配置文件，比如：数据库连接配置文件，都是用的开发环境的数据库连接配置，项目要部署上线的时候，我们需要把开发环境的数据库连接配置替换成生产环境的数据库连接配置，所以这个目录可以用来存**生产环境的相关配置文件**。

之所以将测试环境/开发环境/生产环境分开是因为开发过程中，需要对数据库中的一些数据进行测试或者修改，如果不和生产环境分开，**会“污染”生产环境的数据**。

**dbscripts**

在开发过程中，**每次开发过程中涉及到要执行一些脚本**，比如：开发过程中，要新建一个表，建表语句就要保存在这个目录里面，待开发完毕上线的时候，就需要在生产环境中执行这个建表语句。

**docs**

这里存放**项目的相关文档**，比如：需求说明书，会议纪要，上线手册等。

**src**

这里存放**项目源码**。

**一个例子**

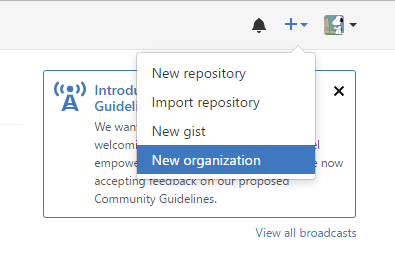
我们通过这个简单的例子来说明如何配置相关目录中的信息。

**团队基本信息**

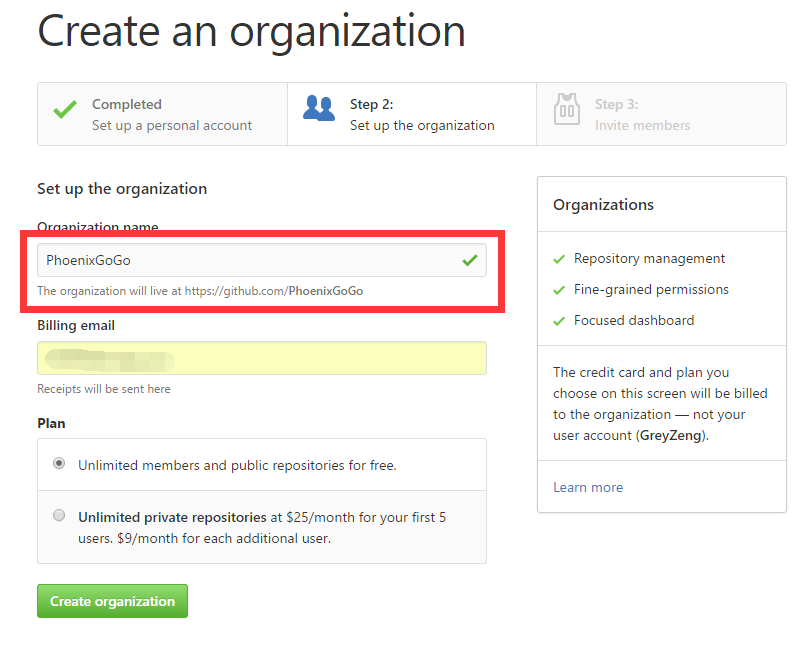
* 团队名称：**PhoenixGoGo**
* 团队要做的项目名称：**helloworld**

**在Github中新建团队**

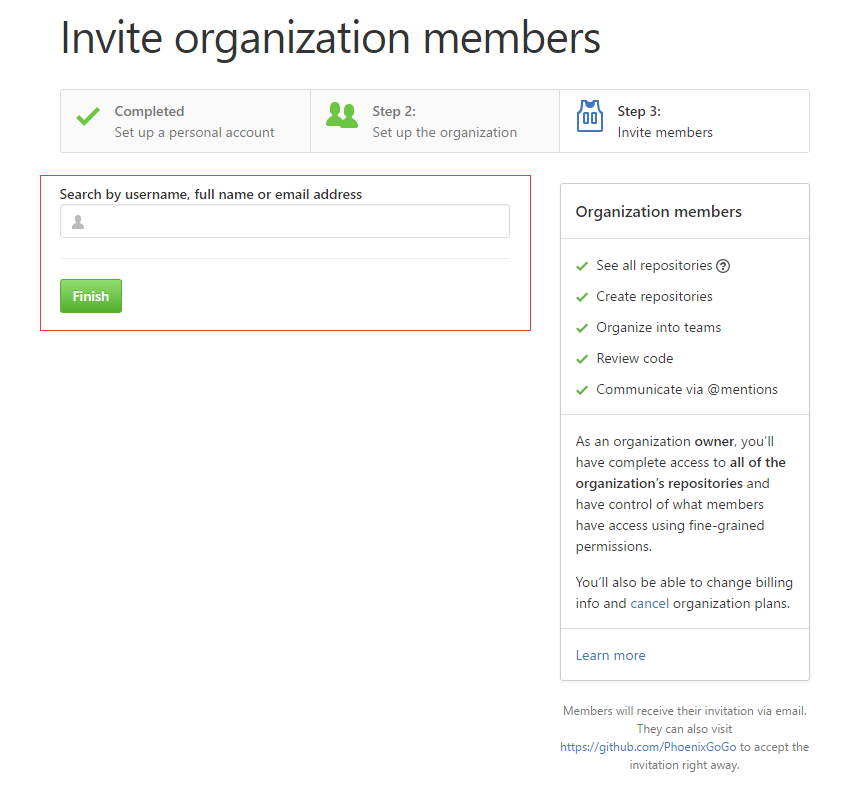
**新建organization**

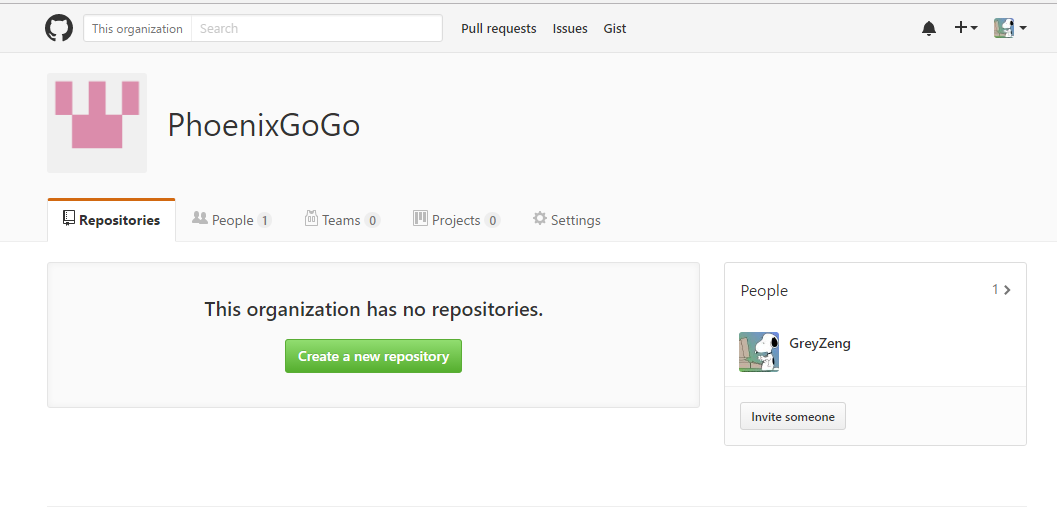


**输入Organization Name等相关信息，并点击Create organization**



**邀请成员，这步我们先略过，点击Finish**

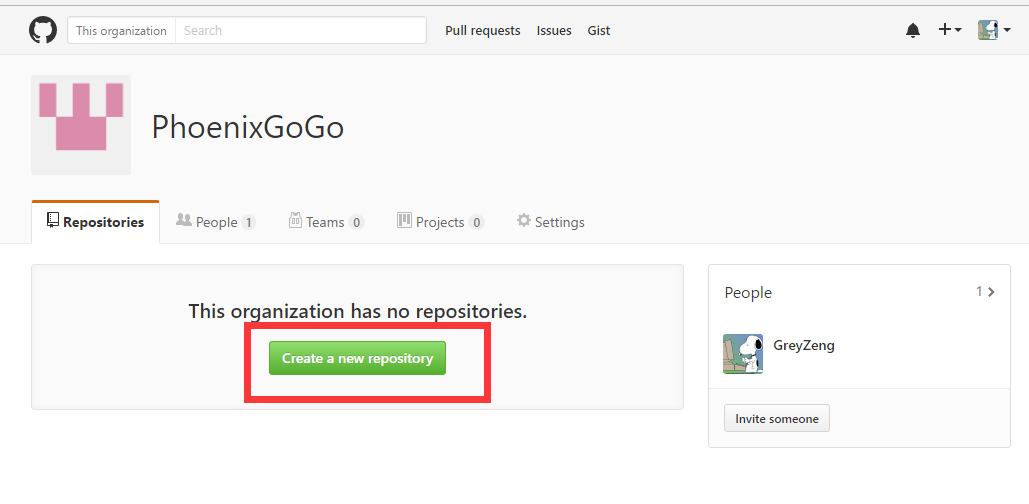




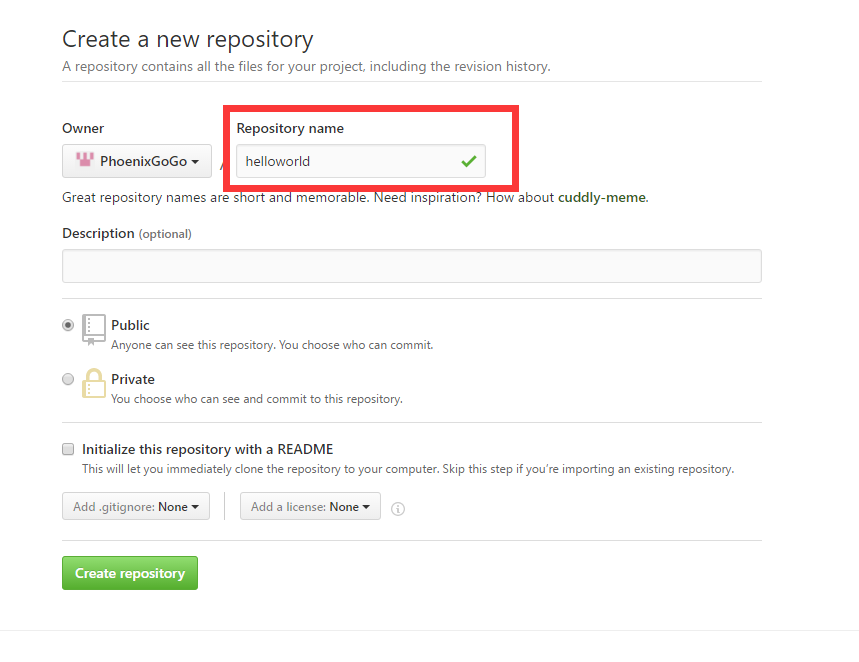
此时，我们就把团队仓库新建完毕。

**在团队仓库中新建项目**

**点击Create a new repository**



**输入项目相关信息, 并点击Create repository**



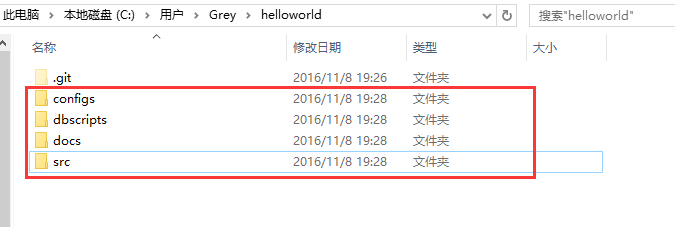
此时，我们就把项目建好了。

**初始化相关目录**

**先将项目clone至本地：**

git clone https://github.com/PhoenixGoGo/helloworld.git

**进入项目目录，并新建以下文件夹：**



**项目需求**

我们要将控制台输入的文本写入到C盘根目录中的file\_product文件夹中的ds.txt文件中。

**配置和初始化**

为了不影响生产环境file\_product文件夹中的ds.txt文件, 我们考虑在C盘根目录中新建一个file\_develop的文件夹，并在这个文件夹里面ds.txt文件，我们平时在开发的时候，就读取file\_develop中的ds.txt文件，等我们开发完毕测试通过以后，我们可以把相关的目录改为file\_product。

为了说明问题，我通过脚本来新建相关目录，以Windows为例，

* 新建一个名为init\_ds.bat文件,
* 用记事本打开这个文件写入以下内容并保存：

@echo off

md "C:\file\_develop"

C:

cd "C:\file\_develop"

cd.>ds.txt

双击这个bat文件，可以在C盘下生成了相关目录和文件。同时，我们把init\_ds.bat文件放到helloworld/dbscripts文件夹下面。

因为以后发布项目的时候要切换目录，所以，我们不能把要读取的目录写死在代码里面，我们可以将目录路径写在配置文件里面，通过读取配置文件来获得目录的路径，这样我们在部署的时候只需要用生产环境的配置文件覆盖开发环境的配置文件即可。

我们可以新建两个配置文件，一个用于开发环境，一个用于生产环境。

我们先新建一个开发环境中需要的配置文件：

* 新建一个名为config.properties的配置文件,
* 用记事本打开这个文件写入以下内容并保存：

# test environment

file.name=C:/file\_develop/ds.txt

再新建一个生产环境中需要的配置文件：

* 新建一个名为config.properties的配置文件,
* 用记事本打开这个文件写入以下内容并保存：

# product environment

file.name=C:/file\_product/ds.txt

我们把这个用于生产的配置文件放在helloworld/configs下面。

**项目源码(部分)**

为了方便，这里对代码没有做优化和检查。[完整源码](https://github.com/PhoenixGoGo/helloworld/tree/master/src/helloworld)

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Properties;

import java.util.Scanner;

public class App {

private static Scanner scanner;

public static void main(String[] args) {

Properties prop = new Properties();

try {

prop.load(new FileInputStream("config/config.properties"));

scanner = new Scanner(System.in);

String line = null;

System.out.print("please input:");

while (!"exit".equals(line = scanner.nextLine())) {

System.out.println("input context = " + line);

System.out.print("please input: ");

File writename = new File(prop.getProperty("file.name"));

writename.createNewFile();

BufferedWriter out = new BufferedWriter(new FileWriter(writename));

out.write(line);

out.flush();

out.close();

}

System.out.println("The program will exit");

} catch (IOException e) {

e.printStackTrace();

}

}

}

我们将源码放入helloworld/src目录下。

我们将helloworldpush到远程仓库：

控制台下分别执行以下命令：

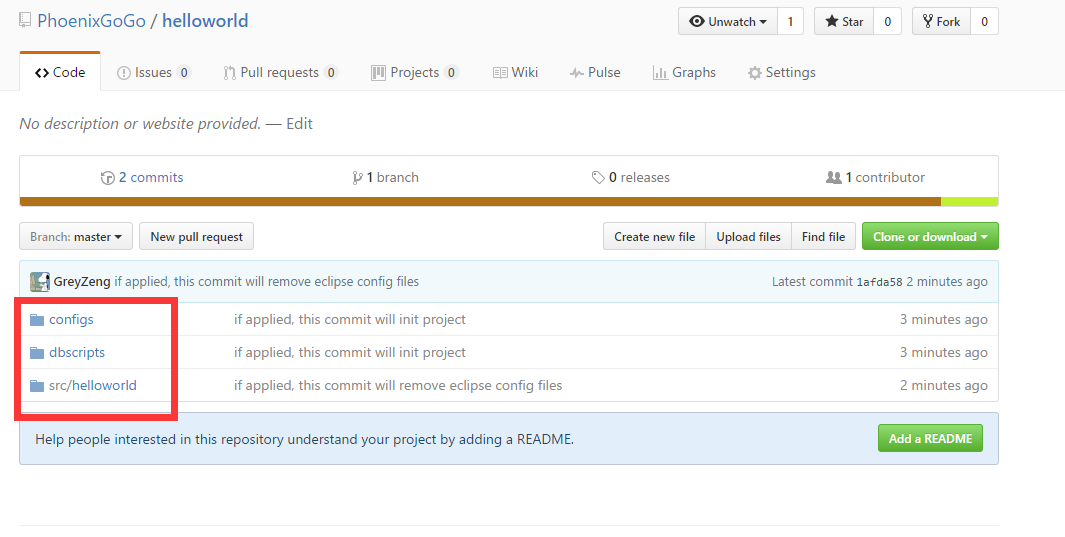
cd helloworld

git add \*

git commit -m "if applied, this commit will init project"

git push -u origin master

此时，项目的结构是这样的：



**相关文档**

项目组经常会有新来的成员 ，他们如何在没有老成员带的情况下，通过文档就可以在一台全新的机器上部署项目呢？

这时候我们就需要一个文档，来说明项目如何部署运行，下面是这个helloword项目的**部署文档**，仅供参考:

**开发环境**

* [JDK1.8+](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)
* [Eclipse Neon](https://www.eclipse.org/downloads/)
* [Maven-3.3.9](http://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.zip)

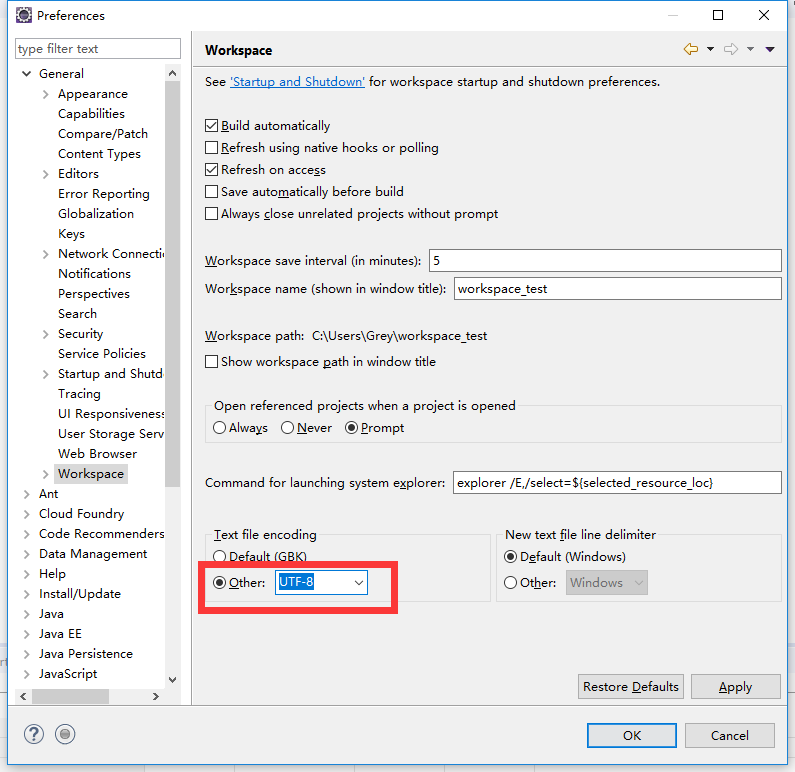
**安装**

* [JDK1.8+](https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html)
* [Eclipse Neon](https://wiki.eclipse.org/Eclipse/Installation)
* [Maven-3.3.9](http://maven.apache.org/install.html)

**相关配置**

**打开Eclipse，设置编码格式UTF-8：**

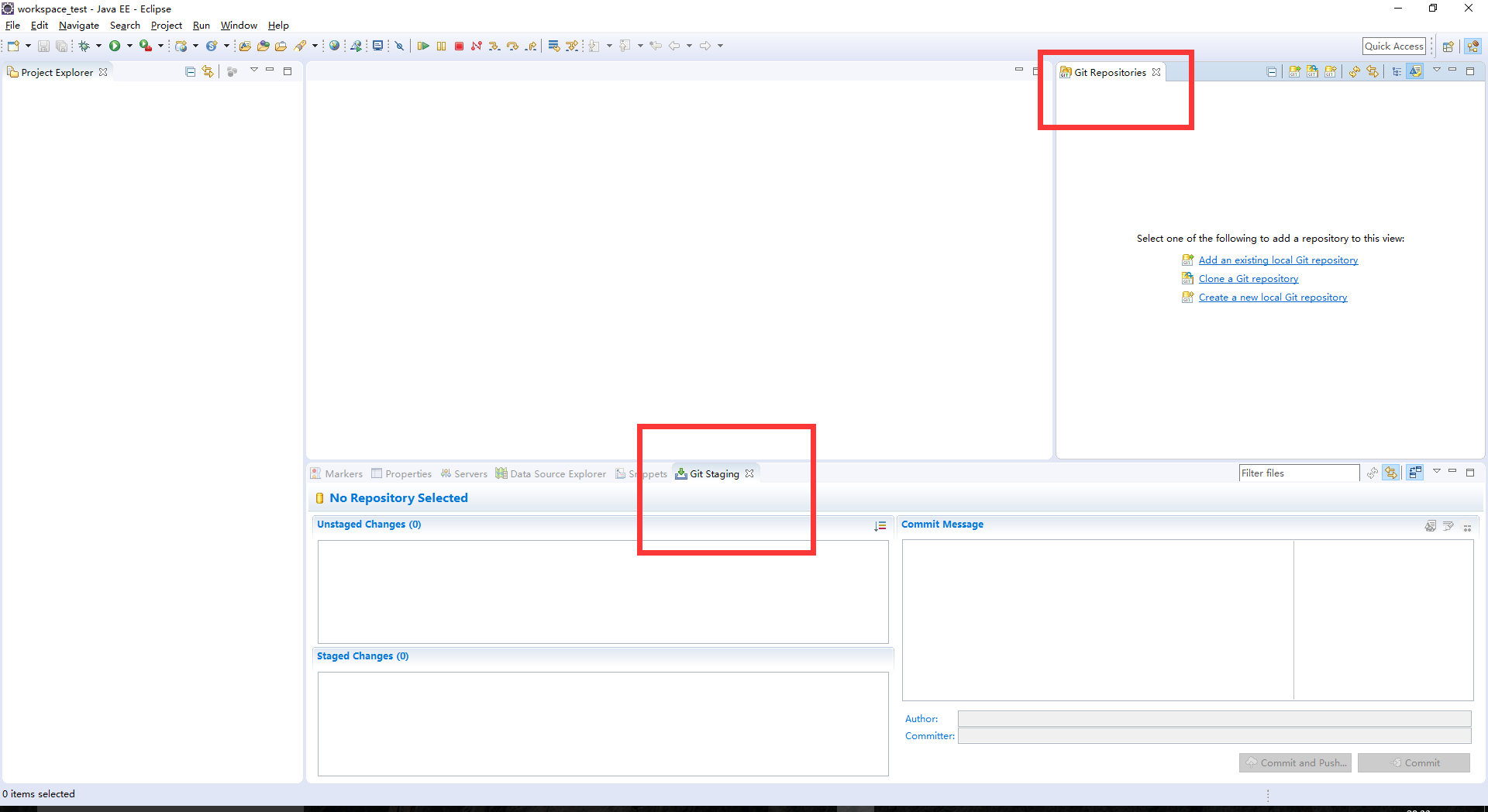
Window->Preferences->General->Workspace->Text file encoding->UTF-8



**在Eclipse中打开：**

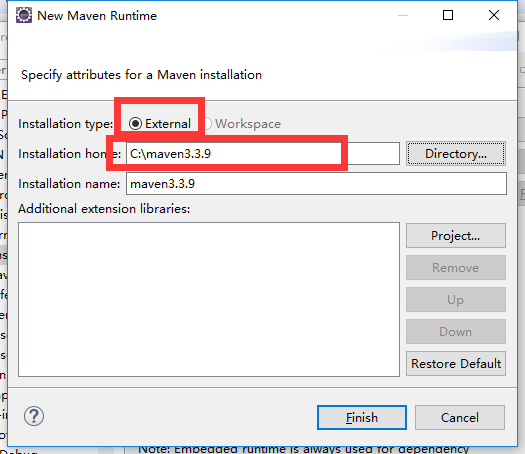
Window->Show View->Other...->找到Git Repositories并双击

Window->Show View->Other...->找到Git Staging并双击



**配置Maven到Eclipse中：**

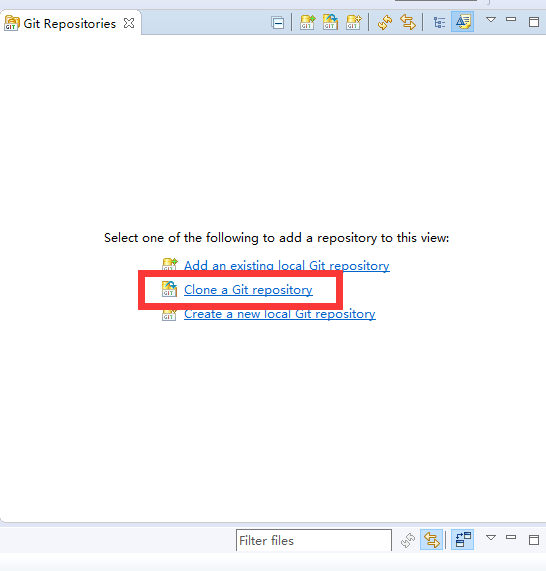
Window->Preferences->Maven->Installations->Add...->选择前面安装的Maven的目录



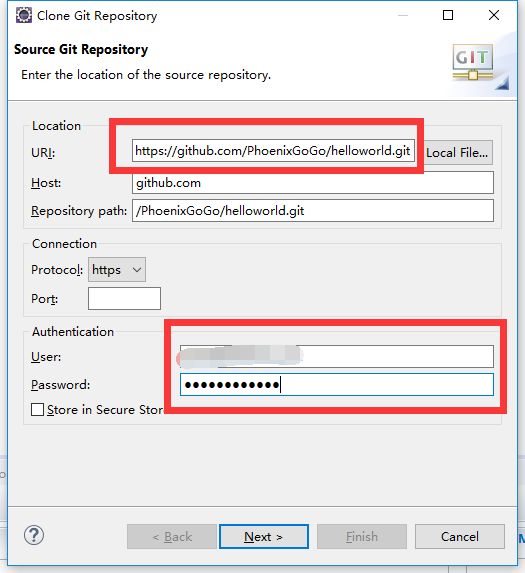
点击Finish。

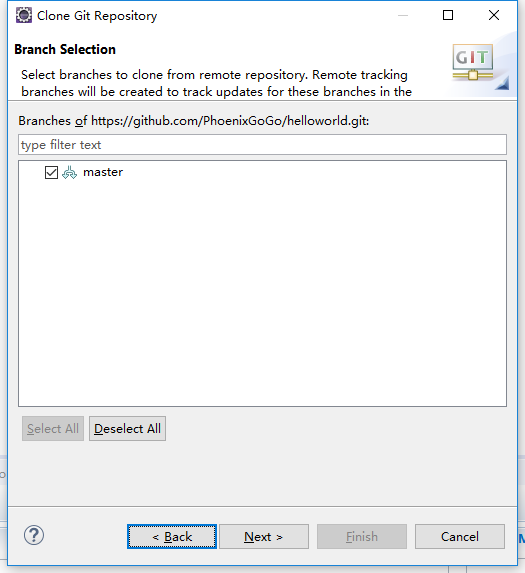
**导入仓库**

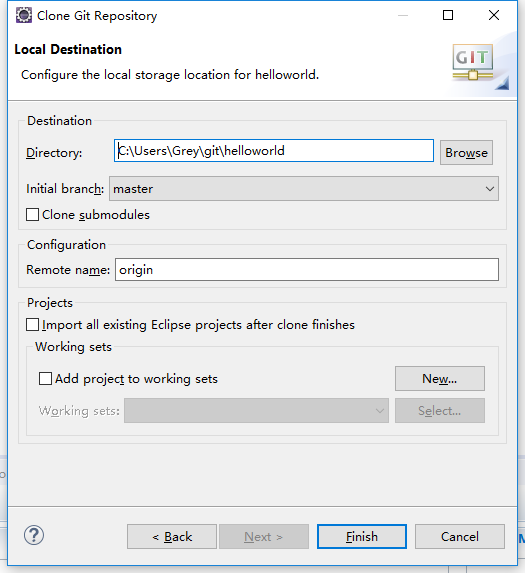
**在Git Repositories下点击clone git repository**



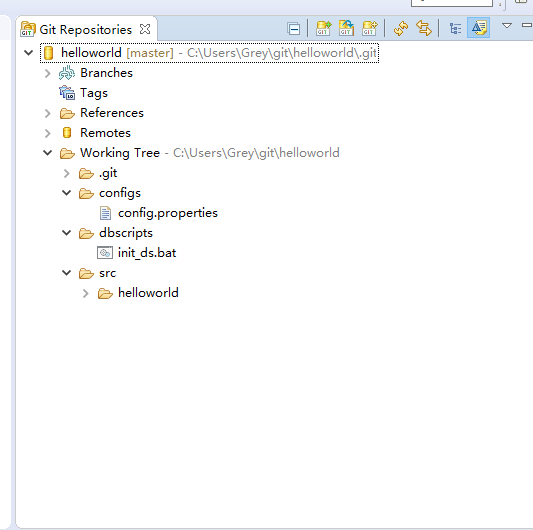
**输入URI以及Github用户名和密码**







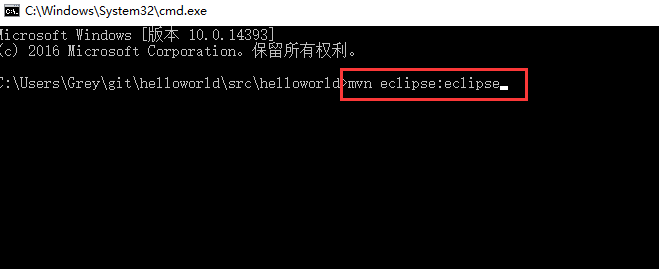
点击finsh

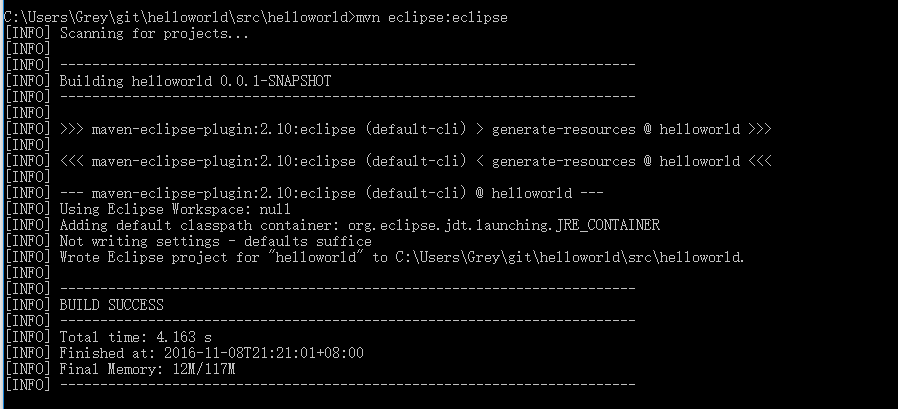


**导入项目**

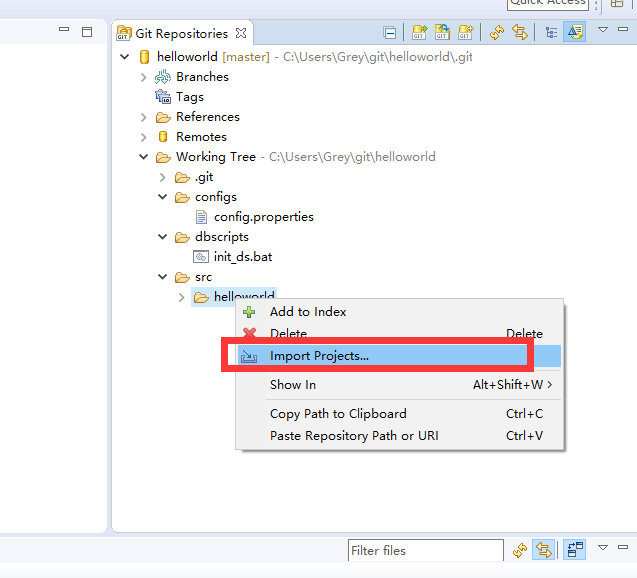
**在控制台下，我们进入源码的目录执行：**

mvn eclipse:eclipse

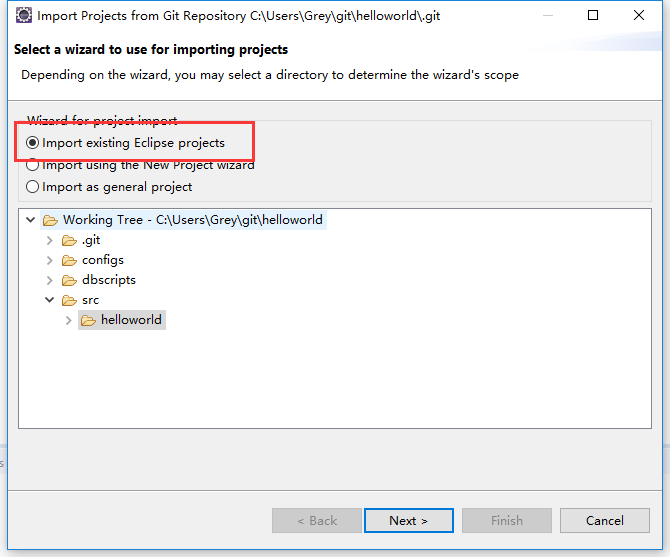


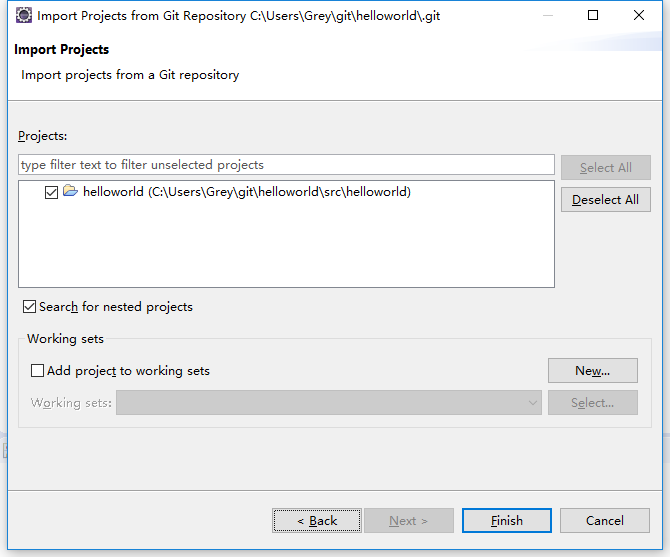


**在src目录下右键，选择Import Projects：**

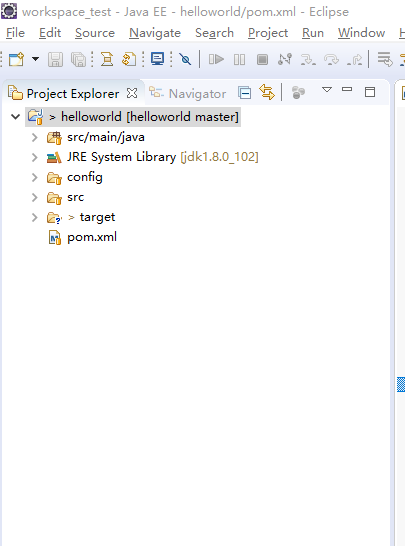


**选择import existing Eclipse Projects:**



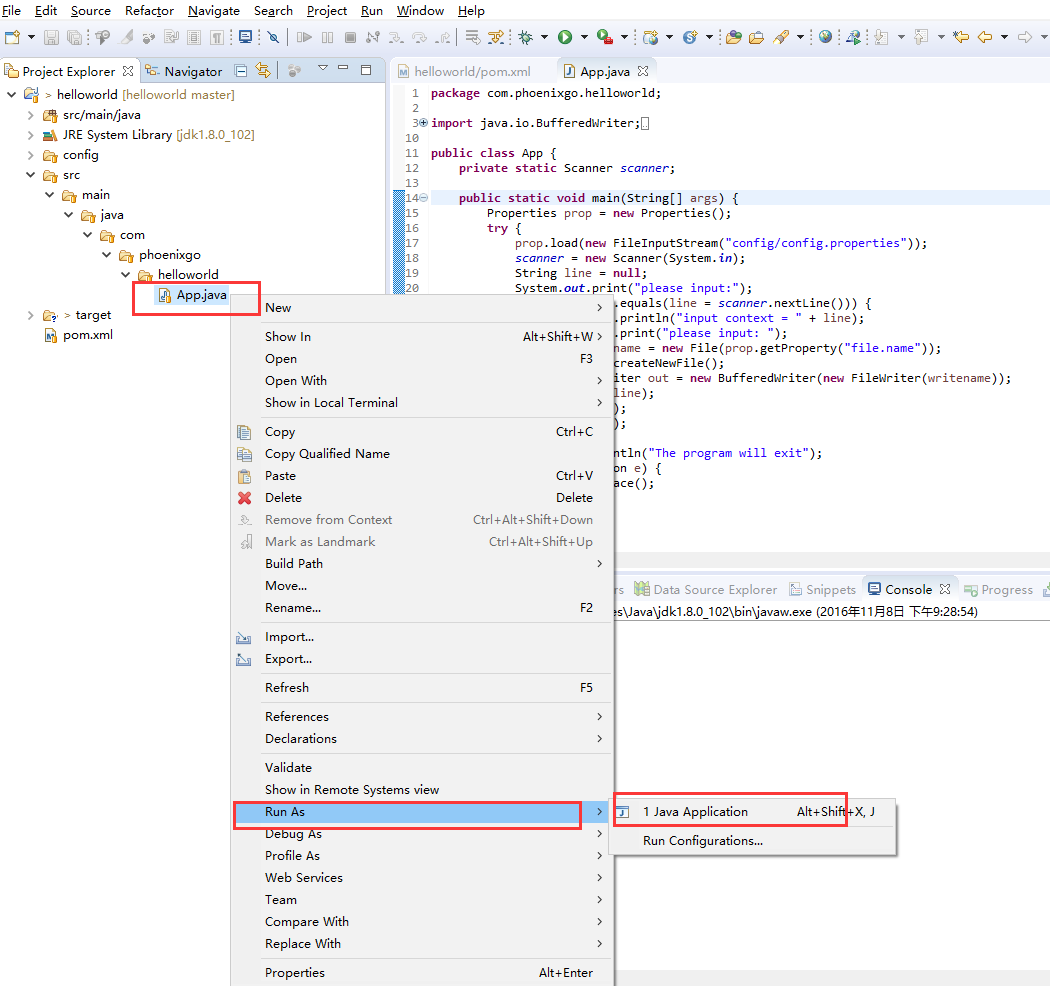


项目导入完成



**运行项目**

选择App.java  
右键-> Run As->Java Application



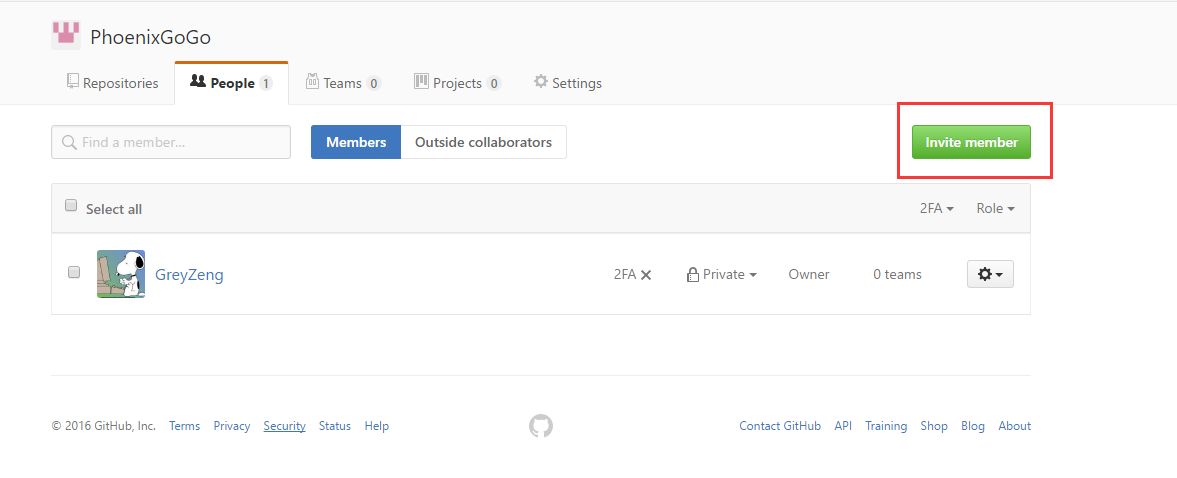
以上为helloworld这个项目的开发部署文档, 我们可以将其保存为：develop\_guide.md 文件，  
并且放到hellworld/docs目录下

最后，把docs目录也提交到团队仓库中去。

**增加团队成员**

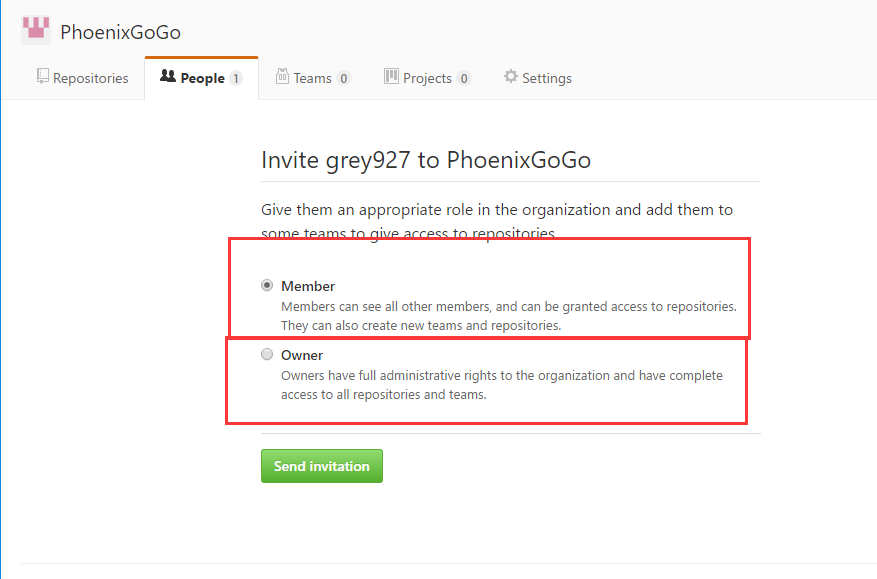
团队仓库初始化好以后，我们可以增加团队成员进行协作开发：

点击Invite member



输入你的团队成员的Github账户即可对相应的成员发出邀请，接收到邀请的成员要进Github注册邮箱进行确认方可加入：

这里包含两类人：Member/Owner ：



他们拥有的权限可以自定义,管理员可根据需要配置不同团队成员的权限：

