
Sedge

# Sedge 开发规范

开源框架：作业提交平台的开发规范

## 1. Sedge 概述

Sedge 是“开源框架：作业提交平台”项目的别称，目的是开发一个开源的作业提交平台框架，以帮助更多的人快速建立自己的作业提交平台。

Sedge 使用 PHP5 + MySQL，采用三层架构，在 GitHub 开发，能够运行在 Windows 及 Linux 平台。

## 2. 加入 Sedge

只要你对 Sedge 感兴趣，我们欢迎你的加入！

开发方式非常简单：阅读本规范，按照要求开发即可。

不用担心没时间，Sedge 允许的最小开发量是 1 模块，也就是说，只写一个分分钟就搞定的模块， 同样是对 Sedge 的贡献。

## 3. 开发方式概述

首先，Sedge 被划分为三层架构，分别是表示层（UIL）、业务逻辑层（BLL）和数据访问层（DAL）。每层事先划分子模块，自下而上（DAL 到 UIL）进行开发。基本上，每个模块划分为一个单独的源文件。模块划分参见《Sedge 模块划分》。

开发者每次选择一个模块进行开发，完成后需提交审核，以验证是否符合本规范，审核通过后，开发者的贡献值+1；若审核不通过，开发者需修改后重新提交。

## 4. 源文件规范

### 4.1 文件名

最终提交的文件名必须与《Sedge 模块划分》中的**文件名**一致。一致是指**字符级别**上的相等。例如：Ascii 字母‘k’和全角字母‘ｋ’不相等、Ascii 字符‘?’和全角字符‘？’不相等。

默认地，DAL 层的文件名前缀为两个下划线，如“__insert.php”，BLL 层的文件名前缀为一个下划线，如“_login.php”，UIL 层的文件名无前缀，如“show.php”。

### 4.2 编码字符集

最终提交的源文件，必须以 UTF-8 格式进行编码。Windows 中默认的文本文件以 ANSI 格式编码，在其他软件中可能会出现乱码。

改变编码的方式有：Windows 下用记事本打开，点击“文件”→“另存为”，在“编码”下拉菜单中选择“UTF-8”，点击“保存”即可；如果你使用 Notepad++，点击“格式”→“转为 UTF-8 编码格式”。其它软件操作类似。

### 4.3 开发者信息

在源文件 PHP 代码的最开始，需要注上此文件的开发者信息，包括作者、日期和功能描述。具体格式如下：

```
<?php
/*
Author: Li Lei
Date: 2013-09-30
Description: 登录验证，传入$user、$pass，验证登录信息是否正确并返回结果。
*/
```

注意：信息必须用一对注释符号“/*  */”进行注释，并且写在<?php 之后的第一行。

Author：本文件的编写者，可以是姓名、昵称、邮箱等任何字符串，长度不超过 255；

Date：本文件开始编写的日期，格式为 yyyy-mm-dd

Description：简单描述本文件功能。

### 4.4 “库”规范

对于经常使用的常量、函数，可以将其放入称为“库”的源文件中。在 Sedge 的每层中/有一个库，命名方式是“**前缀 + main.php**”，例如：DLL 层的库文件名是“**_main.php**”。

库文件允许多个开发者进行修改，因此不保留 Author 信息，仅保留 Date 和 Description，同样，编写库文件也不计贡献值。

## 5. 代码规范

### 5.1 缩进

代码必须使用缩进格式。每级缩进为 1 个 Tab（4 个空格）。

例如：

```
function login($user, $pass)
{
    include('_main.php');
    $err = '';

    /*  判空  */
    if(is_none($user))
    {
        $err = '登录名不能为空';
        return $err;
    }
```

## 5.2 大括号

大括号必须写在单独的一行，并且与上一行的缩进保持一致。

例如：

```
if(is_none($user))
{
    $err = '登录名不能为空';
    return $err;
}
```

以下是两个**错误**的示例：

```
if(is_none($user)) {      // 此处错误
    $err = '登录名不能为空';
    return $err;
}
```

```
if(is_none($user))
    {                     // 此处错误
    $err = '登录名不能为空';
    return $err;
    }                     // 此处错误
```

## 5.3 if 语句 (*)

PS. 此标题中的“*”标记，代表本规范是可选的，你可以这样做，也可以当做没有这条规范。

若 if 语句后只有一句代码，则**可以**省略大括号和换行，直接写成如下格式。

```
if( $num > 2) return true;
```

同样，若 else 语句后只有一句代码，也可以写成如下格式

```
if( $num > 2) return true;
else return false;
```

## 5.4 易阅读的格式

在代码中加入适当的空格能使代码更易阅读。

双目运算符左右各加一个空格。（如“=”、“+”）

逗号后加一个空格。

例如：

```
$a = $b + $c;             // 正确
$a=$b+$c;                 // 错误
```

```
login($user, $pass);      // 正确
login($user,$pass);       // 错误
```

### 5.5 注释

注释分为 4 种，分别是文件注释、函数注释、代码块注释、语句注释。

注释建议用中文描述，以提高可读性。

文件注释：参见 4.3 节描述。

函数注释：每个函数都应有注释，描述此函数的功能。用一对“/**/”，并且注释符号占单独的一行。如：

```
/*
    插入一条记录
    $para 为 key-value 数组，key 属性名，value 为值
    $talbe 为表名
    返回执行结果
*/
function insert($para, $table)
{
```

代码块注释：在某个代码块前进行注释，表示该代码块的功能。用一对“/**/”注释，占一行。如：

```
/*  判空  */
if(is_none($user))
{
    $err = '登录名不能为空';
    return $err;
}
```

语句注释：在语句后的同一行用“//”进行注释，描述此语句的功能。如：

```
return $err; // 返回错误信息
```

### 5.6 命名

标识符、函数名的命名尽量选择简单易懂的名称。均以小写字母命名，**多个单词之间用一个下划线连接**，一般情况下，名称不以下划线开头。比如：

```
$login_type          // 正确
$login-type          // 错误
$LoginType           // 错误
$_login_type         // 错误
```

## 6. 开始工作

### 6.1 注册 GitHub

说了这么多枯燥的规范，下面就来动手准备开发吧。

登录 www.github.com，注册一个帐号。注册时选择免费 plan，不要勾选创建组织。

### 6.2 找到 Sedge

在上方的搜索条中搜索 Sedge，找到/ng1091/Sedge。（可以顺便点一下右上角的 Star，标记这个项目。）

### 6.3 Fork

点击右上角的 Fork 按钮，将 Sedge forking 到你的仓库，即克隆 Sedge 的副本作为你的项目，完成后可以在你的主页中看到。

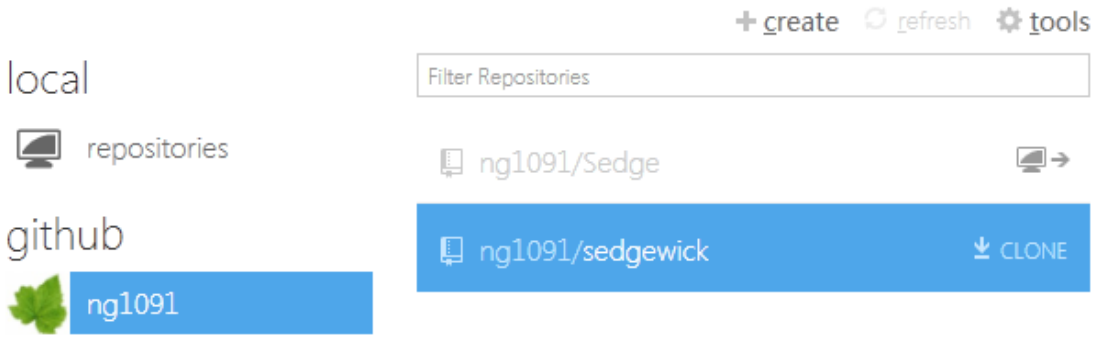
### 6.4 安装客户端

此时建议你安装 GitHub 客户端，将项目同步到本地进行修改。或者尝试更具挑战的 Git 命令行工具。下面以 Windows 客户端为例：

下载安装 GitHub Windows 客户端 http://windows.github.com/

### 6.5 同步至本地

登录之后点击你的用户名，会看到你的项目，点击 CLONE，将项目同步至本地。如下图所示：

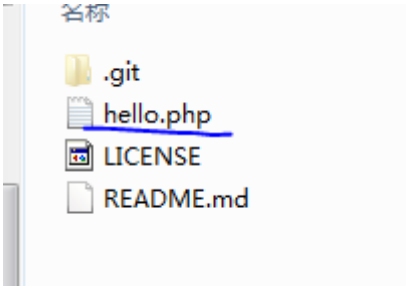



同步完成后，在该项目上点右键，选择“Open in Explorer”，会在资源管理器中打开。参照《Sedge 模块划分》，选择你想修改的模块，开始编码。

另外，为了防止其他开发者选择与你相同的模块，请首先在文件中写上自己的开发者信息，然后 Pull 回服务器，合并到 Sedge 主分支。当整个模块写完后，再 Pull 一次。具体操作流程如下：

### 6.6 写入开发者信息

首先选择想要修改的文件，这里以“hello.php”为例。



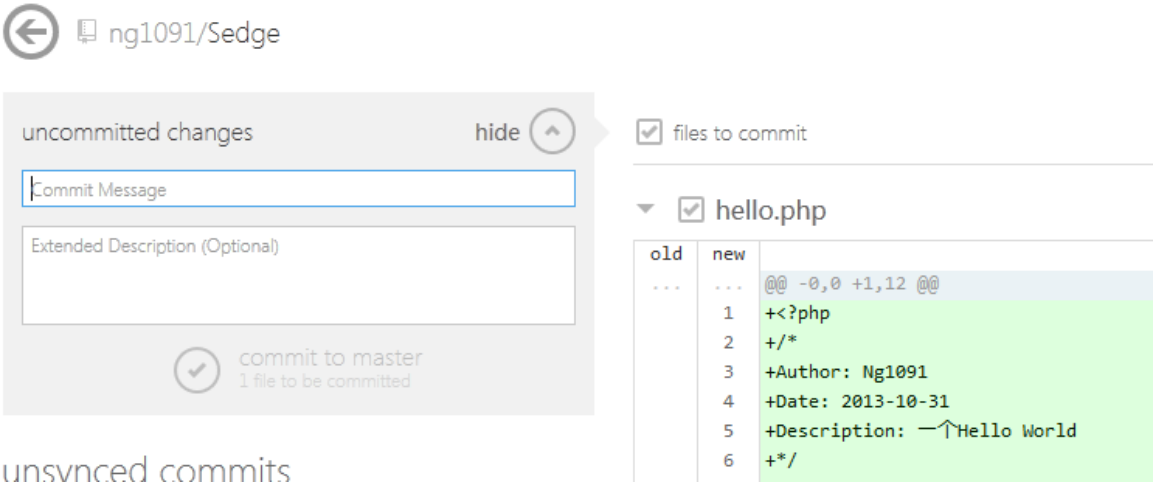

用你喜欢的编辑器打开，写入开发者信息。

```php
<?php
/*
Author: Ng1091
Date: 2013-10-31
Description: 一个Hello World
*/
```

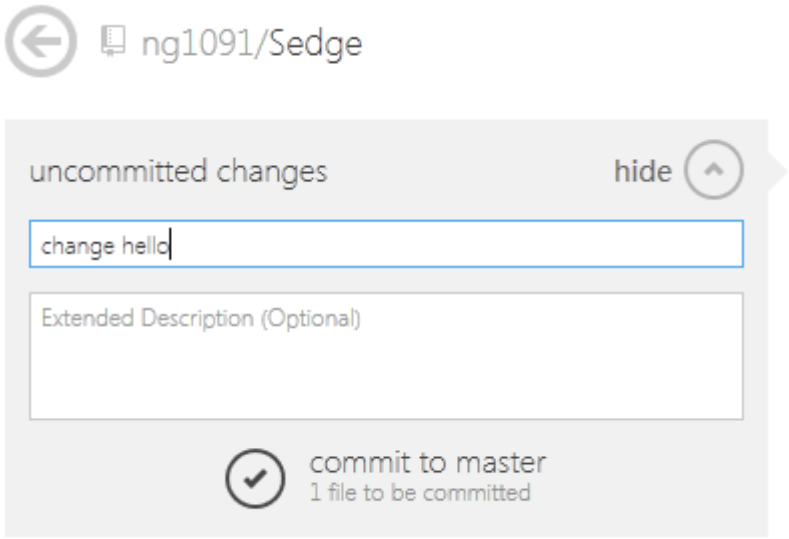
然后保存文件，当然，保存时别忘了必须是 UTF-8 编码格式，否则连 GitHub 他妈也不认识你。

### 6.7 同步至服务器

打开 GitHub 客户端，点击 Sedge 项目，可以看到刚才所做的修改。



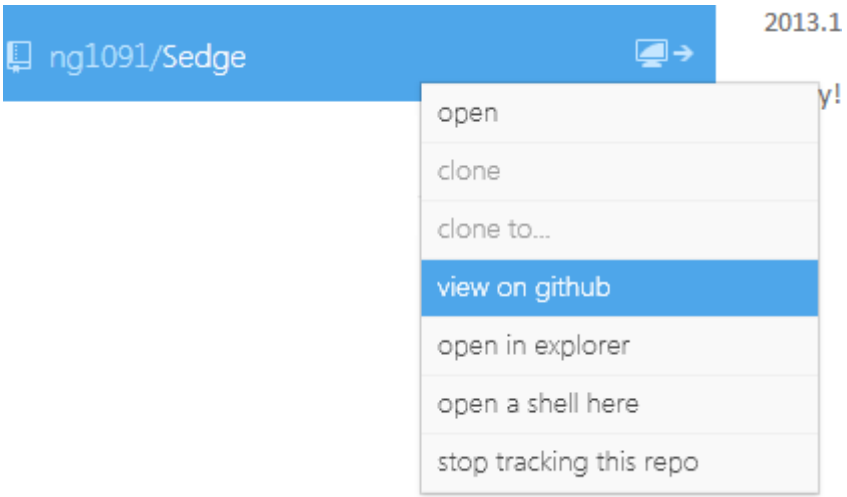
在左侧的 uncommitted changes 中填入 commit 消息，点击 commit to master。




注意，此时变化并未推送至服务器，还需要点击右上角的 Sync 按钮进行同步。

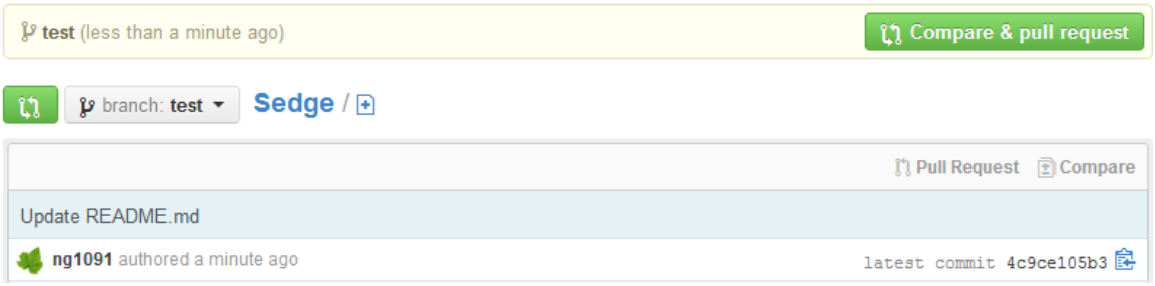
### 6.8 Pull 回主分支

同步完成后，在项目上点右键，选择“open in github”。如图。




它会自动在浏览器中打开你的项目，因为**你的项目是主分支的一个副本**，所以需要发送 Pull Request，申请将你所做的修改合并到主分支。

在网页中点击绿色的“Compare & pull request”按钮，如图




GitHub 比较副本和主分支的差异，列出修改过的文件，点击“Send pull request”按钮，将请求发送给主分支。（当然，如果有什么想说明的，可以在旁边留言）


OK，请求已发送，剩下的就等管理员审核了，审核通过后，所做的修改就会被合并到主分支啦！

### 6.9 完成模块

推送完开发者信息，就可以开始着手写这个模块了。写完之后，用同样的方法，将请求推送至服务器。

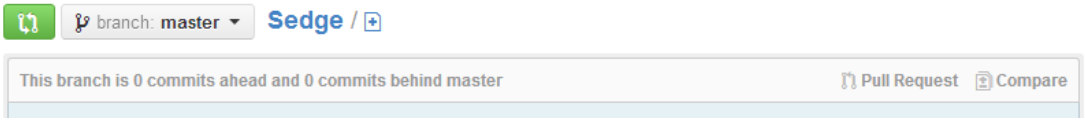
至此，已经成功开发了 Sedge 的一个模块，功德圆满！

### 6.10 再次工作

N 多天之后，想要再写一个模块的时候，也许主分支已经发生了翻天覆地的变化，那么如何才能同步到自己项目中呢？删除之前的副本，重新 forking 一次吗？也行，但是这是最笨的方法，没有之一。

### 6.11 同步主分支

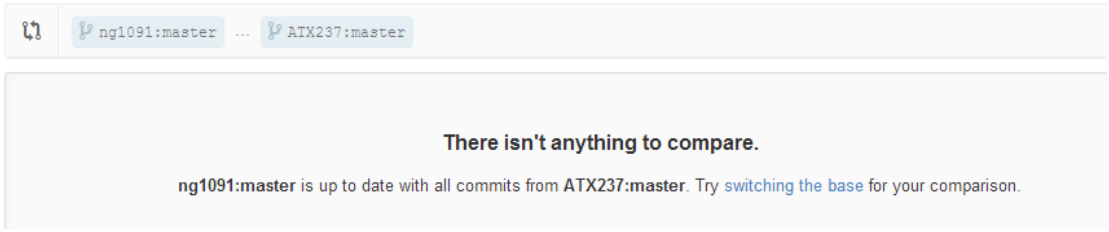
在网页中打开自己的项目，点击灰色的 Compare 链接，或者那个绿色的小图标（Compare & Review）也行。如图所示。



### 6.11.1 切换比较的方向

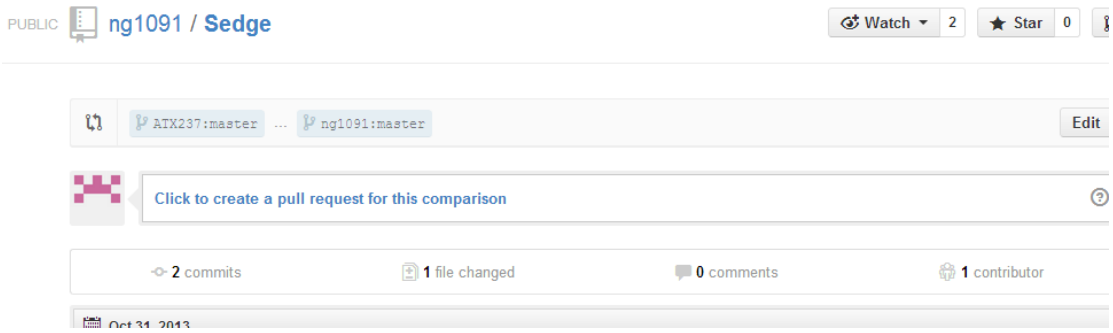
点击“switching the base”链接，或者“Edit”按钮，来改变同步方向。

因为默认主分支为 Base，我们需要把自己的项目设为 Base，以实现将主分支同步给自己。



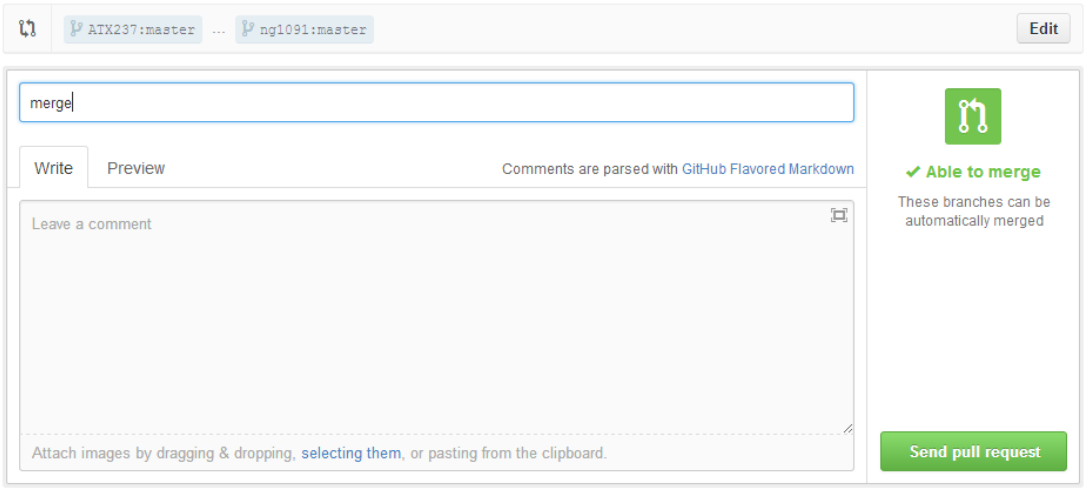

### 6.11.2 发送 Pull Request

页面列出发生变化的文件。点击 Click to create a pull request…，创建一个 Pull 请求。



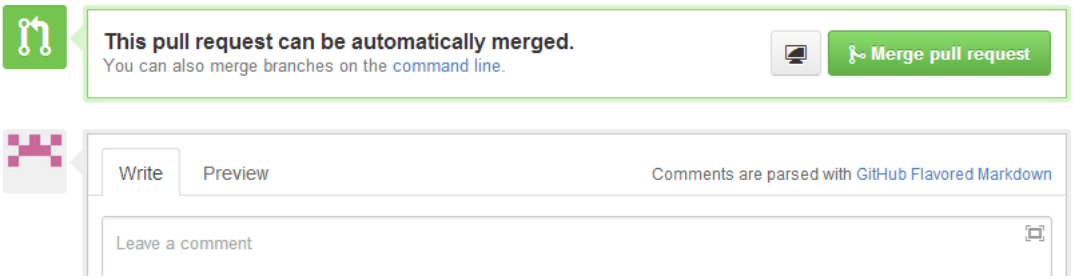

### 6.11.3 填写评论

写上评论，点击 Send pull request 按钮。评论是个很好的东西，可以记录当时的想法，或者与其他开发者进行交流。

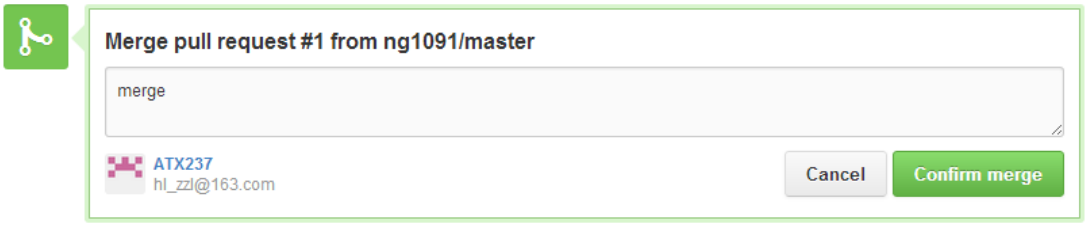


### 6.11.4 合并

此时提示这个 pull request 可以自动合并，点击“Merge pull request”进行合并。




最后还要确认一下，点击“Comfirm merge”按钮。




好了，又一次功德圆满，主分支已加入肯德基豪华午餐！可以开始新一轮的工作了！

### 6.12 更多参考

想对 GitHub 了解更多吗？下面提供两个入门的参考。

《如何在 GitHub 上协作开发开源项目？》

《Github for Windows 使用图文教程》

## 7. 版本与声明

### 7.1 版本

本文档名称：《Sedge 开发规范》

版本：2013.11.1

### 7.2 声明

加入 Sedge 开发，须遵守以上规范。

实行规范以**最新版**为准，最新版发布在 GitHub.com/ng1091/Sedge 及 Sedge 开发群，群号 185288022。

规范发布新版本时，适用文件以开发者信息中的日期为准，在新版本之前的代码，**不适用**新规范。