

Git 和 JIRA 环境配置

Git-环境配置

一 . Git 介绍

Git 作为一个版本控制系统，常用于多人协作开发过程中。它可以作为一个平台，使团队中每个开发人员获取到最新版本的项目，同时它可以记录所有版本，有效追踪文件的变化，当代码出现错误时，可以有效地恢复此前任意状态。此外，这个平台也可以进行“监控”，当出现代码冲突——即多人修改同一段代码时，可以提示上传人人员，并要求上传人员进行冲突选择。

由此，Git 作为时下最流行的版本控制系统，减少了多人协作的团队在开发过程中额外的沟通成本，提高协作效率。

二 . Git 安装和配置

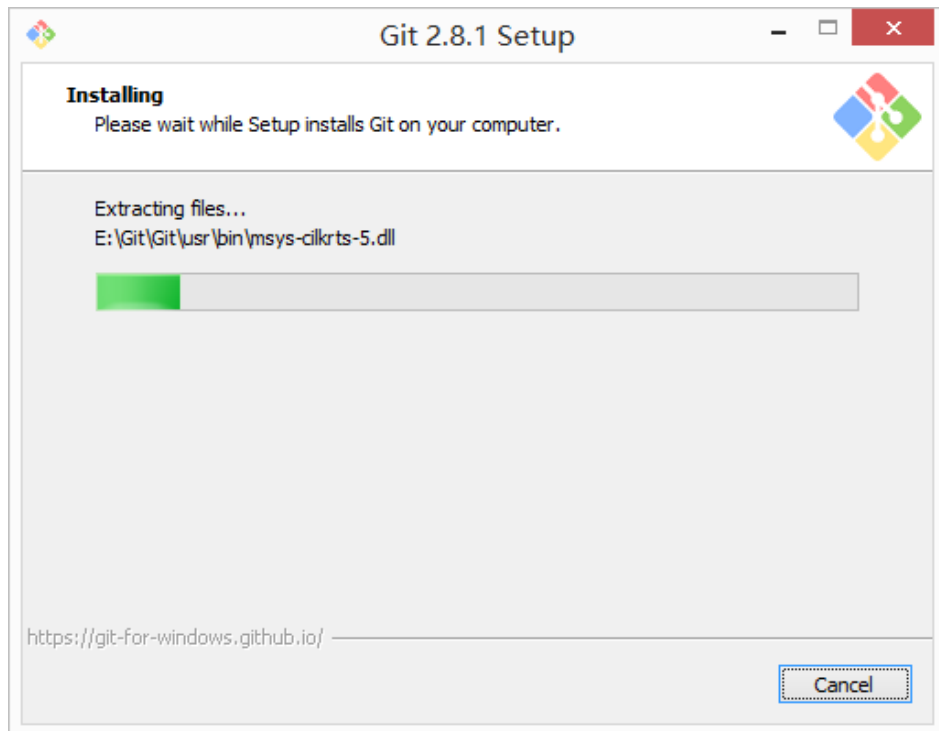
1. 环境： Win8 操作系统

2. Git 安装和配置过程

- 安装

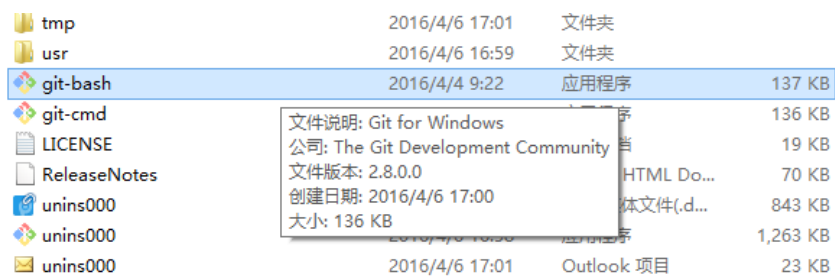
Msysgit 是 Windows 下版本的 Git。集成了 Git 复杂的依赖环境。

下载地址：<http://msysgit.github.io/>



- 配置个人信息

在安装目录中找到 Git Bash.exe 点击进行 Git 操作。



配置个人名称和邮箱

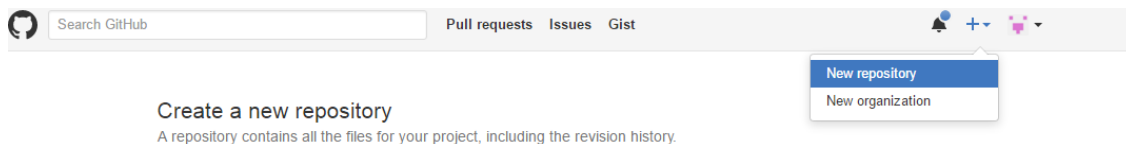
```
admin@lenovo MINGW64 /
$ git config --global user.name "guanlu"

admin@lenovo MINGW64 /
$ git config --global user.email 470616724@qq.com

admin@lenovo MINGW64 /
$
```

3. 创建 Github 远程仓库

- 点击图示，新建 repository



● 填写项目资料

Create a new repository

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

Owner **guan16** / Repository name

Great repository names are short and memorable. Need inspiration? How about [legendary-octo-memory](#).

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

● 创建完成

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

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

or **HTTPS** **SSH**

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

...or create a new repository on the command line

```
echo "# sekko" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/guan16/sekko.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/guan16/sekko.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

ProTip! Use the URL for this page when adding GitHub as a remote.

4. 本地仓库

● 在本地建立文件夹，右键 Git Bash Here



- 输入 `git init` 初始化，此后该文件夹文件加入到版本控制中

```
admin@lenovo MINGW64 /e/Git/home
$ git init
Initialized empty Git repository in E:/Git/home/.git/

admin@lenovo MINGW64 /e/Git/home (master)
$ |
```

- 拷贝远程仓库代码 `git clone <https>/<ssh>`

```
admin@lenovo MINGW64 /e/Git/home (master)
$ git clone https://github.com/Matthew1994/Sekko.git
Cloning into 'Sekko'...
remote: Counting objects: 6, done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 6
Unpacking objects: 100% (6/6), done.
Checking connectivity... done.

admin@lenovo MINGW64 /e/Git/home (master)
$
```

- 创建个人分支 `git branch <分支名>`

```
admin@lenovo MINGW64 /e/Git/home/Sekko (master)
$ git branch guanlu

admin@lenovo MINGW64 /e/Git/home/Sekko (master)
$ |
```

切换到个人分支 `git checkout <分支名>`

```
admin@lenovo MINGW64 /e/Git/home/Sekko (master)
$ git branch guanlu

admin@lenovo MINGW64 /e/Git/home/Sekko (master)
$ git checkout guanlu
Switched to branch 'guanlu'

admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ ls
README.md

admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$
```

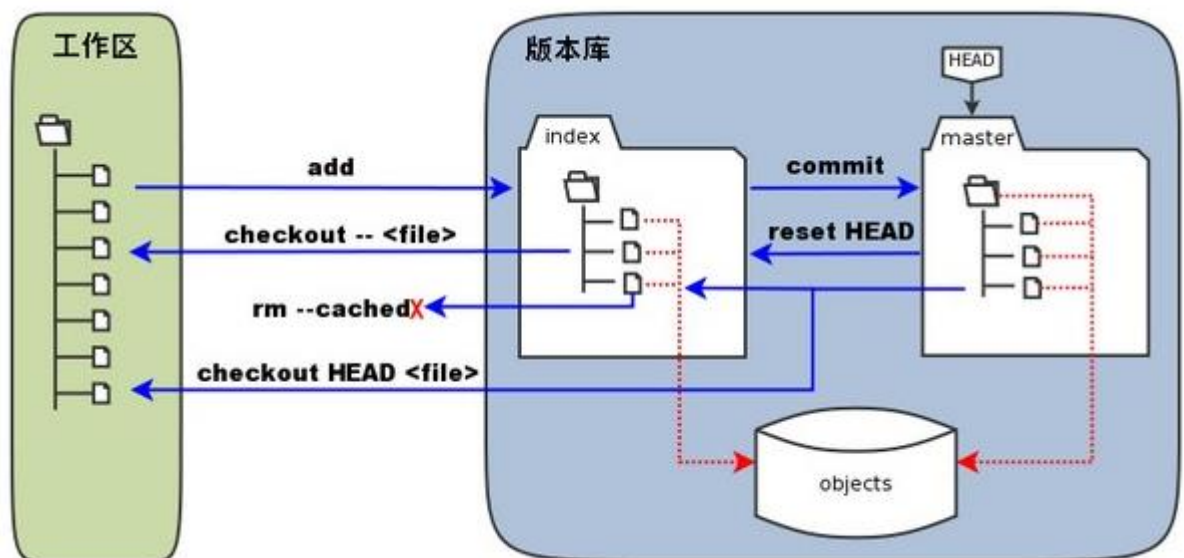
三 . Git 使用方法

1. Git 工作原理 (内容参考 <http://www.runoob.com/git/git-workflow.html>)

Git 分为三个区域

- 工作区：就是你在电脑里能看到的目录。
- 暂存区：英文叫 stage, 或 index。一般存放在 "git 目录" 下的 index 文件 (.git/index) 中，所以我们把暂存区有时也叫作索引 (index)。
- 版本库：工作区有一个隐藏目录 .git，这个不算工作区，而是 Git 的版本库

关系如图：



图中左侧为工作区 右侧为版本库。在版本库中标记为 "index" 的区域是暂存区 (stage, index)，标记为 "master" 的是 master 分支所代表的目录树。

图中我们可以看出此时 "HEAD" 实际是指向 master 分支的一个 "游标"。所以图示的命令中出现 HEAD 的地方可以用 master 来替换。

图中的 objects 标识的区域为 Git 的对象库，实际位于 ".git/objects" 目录下，里面包含了创建的各种对象及内容。

当对工作区修改（或新增）的文件执行 "git add" 命令时，暂存区的目录树被更新，同时工作区修改（或新增）的文件内容被写入到对象库中的一个新的对象中，而该对象的 ID 被记录在暂存区的文件索引中。

当执行提交操作（git commit）时，暂存区的目录树写到版本库（对象库）中，master 分支会做相应的更新。即 master 指向的目录树就是提交时暂存区的目录树。

当执行 "git reset HEAD" 命令时，暂存区的目录树会被重写，被 master 分支指向的目录树所替换，但是工作区不受影响。

当执行 "git rm --cached <file>" 命令时，会直接从暂存区删除文件，工作区则不做出改变。

当执行 "git checkout ." 或者 "git checkout -- <file>" 命令时，会用暂存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到暂存区的改动。

当执行 "git checkout HEAD ." 或者 "git checkout HEAD <file>" 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换暂存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除暂存区中未提交的改动。

2. 常用情景及操作

- 创建远程对应分支 git push origin <本地分支>:<远程分支>

```
admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ git push origin guanlu:sekko_guanlu
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 279 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Matthew1994/Sekko.git
 * [new branch]      guanlu -> sekko_guanlu
admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ |
```

- 本地开发

为了每次在最新版本上进行代码开发，首先需要拉取远程最新代码

git pull origin<分支名>

```
admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ git pull origin sekko_guanlu
From https://github.com/Matthew1994/Sekko
 * branch          sekko_guanlu -> FETCH_HEAD
Already up-to-date.

admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$
```

在本地个人分支修改之后，git add 将文件加入到缓存

之后 git commit 提交到本地仓库

```
admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ git commit
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory.
[guanlu warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory.
87abec3] modified: test.txt
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory.
1 file changed, 1 insertion(+), 1 deletion(-)

admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$
```

- 提交远程仓库

提交之前，首先拉取远程仓库，获取最新版本

```
admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ git pull origin sekko_guanlu
From https://github.com/Matthew1994/Sekko
 * branch          sekko_guanlu -> FETCH_HEAD
Already up-to-date.

admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ |
```


提交代码 git push origin <本地分支名> : <远程分支>

```
admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$ git push origin guanlu:sekko_guanlu
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 286 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Matthew1994/Sekko.git
 a3900ca..87abec3  guanlu -> sekko_guanlu

admin@lenovo MINGW64 /e/Git/home/Sekko (guanlu)
$
```




此时去 github 验证，可以看到自己的远程分支上的提交文件


Your recently pushed branches:

 **sekho_guanlu** (2 minutes ago) [Compare & pull request](#)

Branch: **sekho_...** [New pull request](#) [New file](#) [Upload files](#) [Find file](#) [HTTPS](#) <https://github.com/Matthe> [Download ZIP](#)

This branch is 2 commits ahead of master. [Pull request](#) [Compare](#)

 guanl6 modified: test.txt	Latest commit 87abec3 9 minutes ago
 README.md	update README.md 22 days ago
 test.txt	modified: test.txt 9 minutes ago

 **README.md**

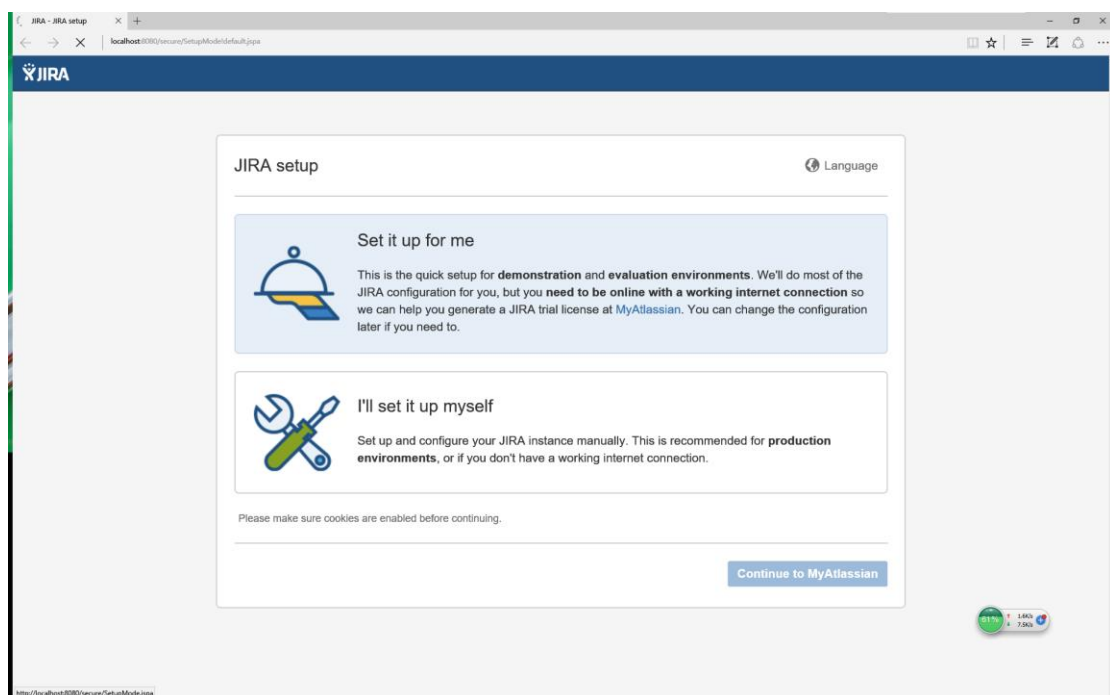
- 提交远程仓库出现冲突

冲突表现为，两个分支下同时修改了同一段代码。解决方式为手动选择更改，使代码统一后提交。注意，如果不确认的情况下，应该与相应的提交人员进行沟通后再进行代码选择。

JIRA 简易使用教程

1. 安装官网地址：<https://www.atlassian.com/software/jira/>
2. 在 window 下在安装是傻瓜式的，按照它的引导一步一步来就可以了。
3. 使用

3.1 安装成功后会在浏览器中出现以下界面



3.2 创建项目

YOUR JIRA

Your first project

Now that you know the basics, let's get started on some work. What is the name of a project you are currently working on?

Project name ×
Max. 80 characters.

Key ?
Max. 10 characters.

[Create Project](#) [Skip](#)

3.3 项目成功后会出现一下界面 ,这是 JIRA 管理项目的主界面 ,主要分为两部分 sprint 和 Backlog。

YOUR JIRA Dashboards Projects Issues Boards Create Search ? ⚙️ 👤

SEK board Board ⌵ ⌵

Backlog

QUICK FILTERS: Only My Issues Recently Updated

FILL YOUR BACKLOG WITH ISSUES
This is your team backlog. Create and estimate new issues, and prioritize the backlog using drag and drop.

Backlog 0 issues [Create Sprint](#)

What needs to be done?
New Story in Backlog

Cancel ...

88% 1,200 2,200

3.4 创建事件。先输入事件名字 , 然后打开右边的指示标志可以得到以下界面 , 可以编辑相关内容。其中 issue Type 有四个选项:story, task, bug, Erp。可以编辑 sprint 的事件 , 或者编辑 backlog 的故事 , 或者作为 bug 跟踪。

You have temporary access to administrative functions. Drop access if you no longer require it. For more information, refer to the documentation.

Create Issue

Project: SEKKO (SEK)

Issue Type: Story

Summary: 登录注册

Reporter: 103774566@qq.com

Component/s: None

Description:

1. 点击登录，出现登录界面，登录界面有注册的链接。
2. 点击注册，出现注册界面，注册基本信息包括手机号码，昵称，邮箱。
3. 成功登录后2小时之内关闭网页再重新访问，可以保持原来的登录状态。
4. 注销后清除登录状态。

☐ Create another **Create** Cancel

然后创建事件的时候可以选择时间的优先级，标签等详细项。

Create Issue

Fix Version/s: None

Priority: Medium

Labels:

Linked Issues: blocks

Issue:

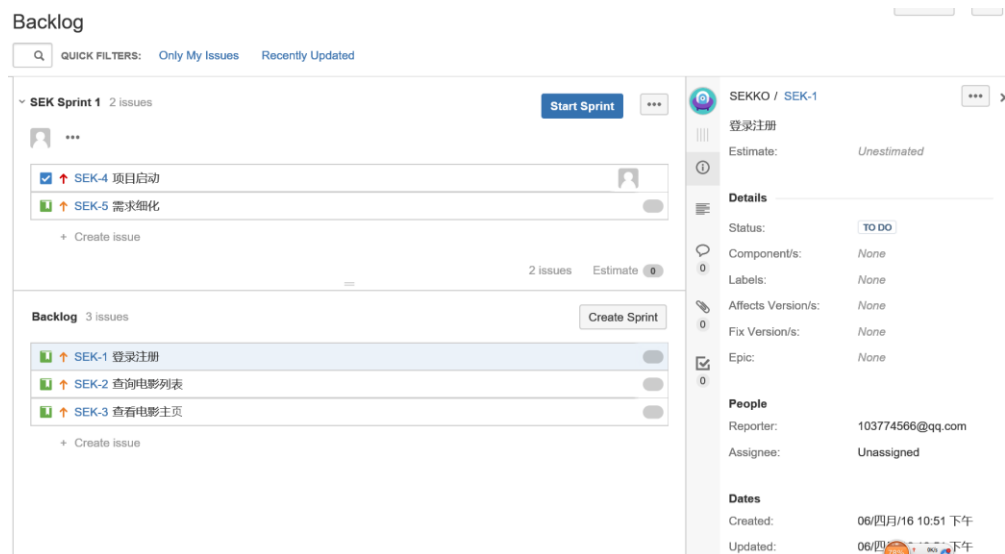
Assignee: Automatic

Epic Link:

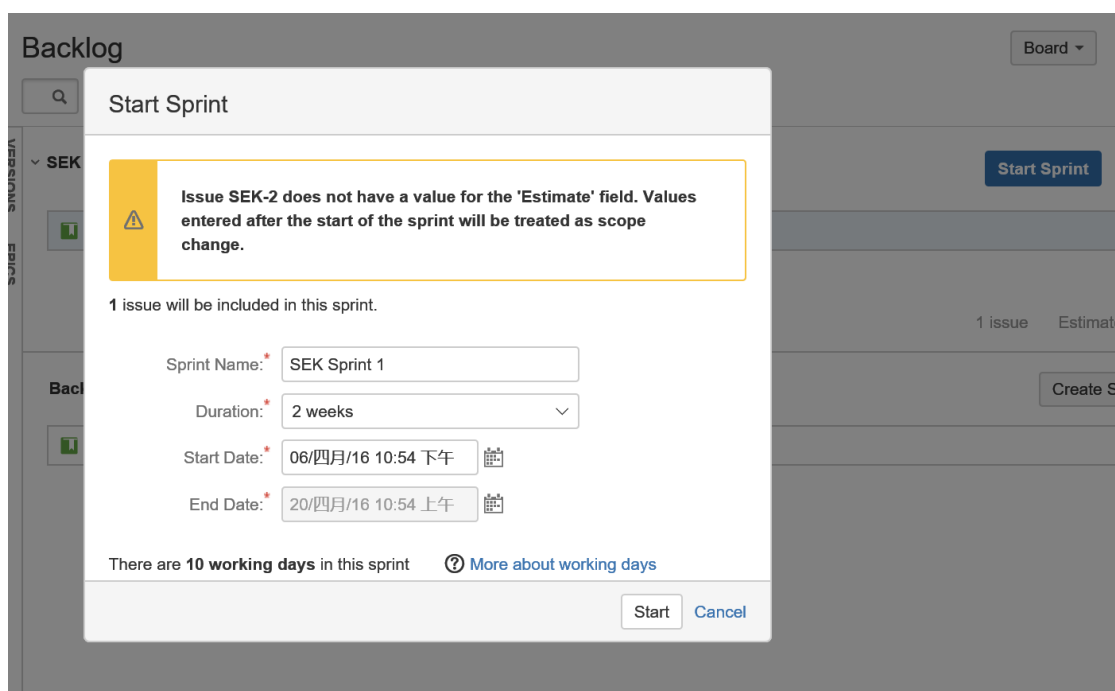
Sprint: SEK Sprint 1

☐ Create another **Create** Cancel

3.5 创建 sprint 和 backlog 的细项后，可以通过双击查看该事件的详情，如下图。



3.6 创建了 sprint 的内容后，就可以按 start sprint 来启动 sprint，然后可以选择每个 sprint 的开始时间和持续时间。



3.7 用户的每次修改和添加行为都会被系统记录下来，然后其他人可以对该行为进行评论。



Project Lead

 103774566@q

Key

SEK

Today

 103774566@qq.com created **SEK-3 - 查看电影主页**

1. 点击电影列表中的电影条目可以进入该电影的主页
2. 主页中有该电影的基本信息（同ID2），以及提供上映该电影的电影院链接
3. 主页还应该影迷的评论区，显示用户的评论和允许已登录的用户评论

 Just now [Comment](#)

 103774566@qq.com removed the Sprint of **SEK-2 - 查询电影列表**

 2 minutes ago [Comment](#)

 103774566@qq.com created **SEK-2 - 查询电影列表**

1. 首页显示最新电影列表
2. 列表中的电影信息包括：片名，海报，类型，评分，时长，主演
3. 可以通过热度，评分，上映时间给列表排序
4. 可以通过类型来进行筛选影片

 4 minutes ago [Comment](#)

 103774566@qq.com created **SEK-1 - 登录注册**


1. 点击登录，出现登录界面，登录界面有注册的链接。
2. 点击注册，出现注册界面，注册基本信息包括手机号码，昵称，邮箱。
3. 成功登录后2小时之内关闭网页再重新访问，可以保持原来的登录状态。
4. 注销后清除登录状态。

 6 minutes ago [Comment](#)

3.8 sprint 启动后可以通过 sprint report 查看项目进度

SEK board

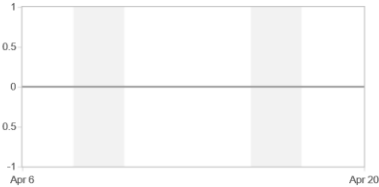
Sprint Report [Switch report](#)

Board 

Hide this information





SEK Sprint 1




Active Sprint 06/四月/16 11:19 下午 - 20/四月/16 11:19 上午 [Linked pages](#) [View SEK Sprint 1 in Issue Navigator](#)



Status Report

Issues Not Completed [View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (-)
SEK-4	项目启动	 Task	 Highest	TO DO	-
SEK-5	需求细化	 Story	 Medium	TO DO	

 85%  1 0.00  1 1.00