利用commitizen来规范化你的commit-message

我们在每次提交代码时，都需要编写Commit Message，否则是不允许提交的。 git commit -m "first commit with userInfo service"
编写Commit Message需要遵循一定的范式，内容应该清晰明了，指明本次提交的目的，便于日后追踪问题。
`commitizen` 就是这么样一款工具,他用来规范化我们的commit消息。

安装指南

1. 安装commitizen

```
sudo npm install -g commitizen
```

1. 配置

cd到 `.git` 所在目录

```
commitizen init cz-conventional-changelog --save-dev --save-exact
```

1. 使用

用 `git cz` 命令来取代 `git commit`

异常情况

有可能提示你缺少package.json

package.json的下载地址 (http://hangzhou.qixing-group.com/download/package.json)

配置第二个项目

当我们配置第二个项目的时候我们只需要进入到对应的项目目录下，输入如下的命令

```
commitizen init cz-conventional-changelog --force
```

我们就可以使用 `git cz` 命令了

commitizen详解

Message 格式

一般来说，Commit Message 应包含三部分内容：Header、Body、Footer

```
<type>(<scope>): <subject>
// 空一行
<body>
// 空一行
<footer>
```

Header

Header部分应只包含一行，包括三个字段：type、scope和subject

- type type用于说明Commit的类型，包含一下7种类型

```
feat：新功能（feature）
fix：修补bug
docs：文档（documentation）
style： 格式（不影响代码运行的变动）
refactor：重构（即不是新增功能，也不是修改bug的代码变动）
test：增加测试
chore：构建过程或辅助工具的变动
```

- scope

```
scope用于说明本次Commit所影响的范围，比如controller、user或者README，视项目的不同而不同
```

- subject

  subject是本次Commit目的的简短描述，一般不要超过50个字符

```
以动词开头，使用第一人称现在时，比如change，而不是changed或changes
第一个字母小写
结尾不加句号（.）
```

Body

Body是对本地提交的一个详细描述，下面是一个示例

```
More detailed explanatory text, if necessary.  Wrap it to
about 72 characters or so.

Further paragraphs come after blank lines.

- Bullet points are okay, too
- Use a hanging indent
```

Footer

Footer只用于两种情况

- ***不兼容改动***

如果当前代码与上一个版本不兼容，则 Footer 部分以BREAKING CHANGE开头，后面是对变动的描述、以及变动理由和迁移方法。

- ***关闭Issue***

如果当前Commit是针对某个Issue的提交，那么久可以在Footer中关闭这个Issue：
Closes #234