

# Tools Solutions

陈磊

2017 年 5 月 18 日

## 目录

|          |                          |          |
|----------|--------------------------|----------|
| <b>1</b> | <b>代码质量</b>              | <b>1</b> |
| 1.1      | checkstyle . . . . .     | 1        |
| 1.1.1    | maven . . . . .          | 1        |
| 1.2      | findbugs . . . . .       | 2        |
| 1.2.1    | IDEA 插件 . . . . .        | 2        |
| 1.2.2    | maven . . . . .          | 2        |
| 1.3      | sonar . . . . .          | 3        |
| 1.3.1    | IDEA 插件 . . . . .        | 3        |
| 1.3.2    | maven . . . . .          | 3        |
| <b>2</b> | <b>持续集成</b>              | <b>3</b> |
| 2.1      | travis-ci . . . . .      | 3        |
| 2.1.1    | travis-ci 使用流程 . . . . . | 3        |
| 2.1.2    | 添加编译状态图标 . . . . .       | 3        |
| 2.1.3    | travis 配置文件样本 . . . . .  | 4        |
| <b>3</b> | <b>版本控制</b>              | <b>4</b> |
| 3.1      | git 多账户 . . . . .        | 4        |
| 3.1.1    | 生成公钥 . . . . .           | 4        |
| 3.1.2    | 添加 ssh 配置文件 . . . . .    | 5        |
| 3.1.3    | 为项目设置单独的账户配置 . . . . .   | 5        |

|       |   |   |
|-------|---|---|
| 1     | 代码质量                                      | 2 |
| 4     | 文档处理工具                                    | 5 |
| 4.1   | Pandoc                                    | 5 |
| 4.1.1 | markdown 多个文档转换                           | 5 |
| 4.1.2 | markdown 转 LaTeX, 用 listings 包替换 verbatim | 6 |
| 4.2   | LaTeX                                     | 6 |
| 4.2.1 | Windows 上编译缓慢, 经常卡在 eu1lmr.fd             | 6 |
| 4.2.2 | listings 代码换行                             | 6 |
| 5     | 编辑器及插件                                    | 6 |
| 5.1   | Visual Studio Code                        | 6 |
| 6     | 工具安装                                      | 7 |
| 6.1   | nodejs                                    | 7 |
| 6.2   | git                                       | 8 |
| 6.2.1 | Ubuntu                                    | 8 |
| 6.2.2 | CentOS 从源码安装 git                          | 8 |

## 1 代码质量

### 1.1 checkstyle

#### 1.1.1 maven

```

1 <reporting>
2   <plugins>
3     <plugin>
4       <groupId>org.apache.maven.plugins</groupId>
5       <artifactId>maven-checkstyle-plugin</artifactId>
6       <version>2.17</version>
7       <configuration>
8         <!--自定义 checkstyle 配置路径-->
9         <configLocation>${project.basedir}/checkstyle.xml</
            ↪ configLocation>
10      </configuration>
11    </plugin>
12
13    <!--使 checkstyle 结果可以直接跳转到代码行位置-->
14    <plugin>

```

```
15     <groupId>org.apache.maven.plugins</groupId>
16     <artifactId>maven-jxr-plugin</artifactId>
17     <version>2.3</version>
18   </plugin>
19 </plugins>
20 </reporting>
```

mvn site 执行后, 打开 target/site/index.html, 在 Project Reports -> Checkstyle 查看报告. 点击对应行数, 可查看具体代码位置.

**checkstyle 常见问题包括:**

1. 代码缩进不对.
2. 控制语句没加大括号.
3. equals 字符串应放在前面.
4. if-else 条件语句包含 return 的未简化 (可去除 if-else 直接 return).
5. 无用的包引入.
6. 命名问题: 包名、变量名、类名、函数名.
7. 字符串太长 (超出限制).
8. 缺注释.

**checkstyle 修改快捷方法 (eclipse 或者 IDEA):**

1. 变量名修改可用 Refactor 的快捷键. eclipse、IDEA 可用.
2. 缩进或控制语句缺大括号, 可配置格式化属性 (或导入模板), 快捷键针对工程中所有文件格式化, 比如 IDEA 的 Ctrl+Alt+L. eclipse, IDEA 可用.
3. eclipse 可安装 javadoc (不是这个名字), 对整个包快捷键添加注释.

## 1.2 findbugs

### 1.2.1 IDEA 插件

idea 编辑器可安装 FindBugs-IDEA 插件 (File->Settings->Plugins->Browse repo...). 安装重启后编辑器后, 底部会显示. 选择 src 文件夹, 点击 Analyze Module Files 可分析全部源文件.

### 1.2.2 maven

```
1 <reporting>
2   <plugins>
```

```
3      <!-- https://mvnrepository.com/artifact/org.codehaus.mojo/  
      ↪ findbugs-maven-plugin -->  
4      <plugin>  
5          <groupId>org.codehaus.mojo</groupId>  
6          <artifactId>findbugs-maven-plugin</artifactId>  
7          <version>3.0.4</version>  
8      </plugin>  
9  </plugins>  
10 </reporting>
```

mvn site 执行后在 Project Reports -> FindBugs 查看报告. mvn site 执行下载 FindBugs 相关依赖错误时, 可尝试用 VPN 解决.

## 1.3 sonar

### 1.3.1 IDEA 插件

idea 编辑器可安装 SonarLint 插件 (File->Settings->Plugins->Browse repo...). 安装重启后, 点击底部显示的 SonarLint, 并点击此 tab 里面的 Project Files, Scope 项选择 All projects files, 点击绿色运行图标进行分析. 在编写代码过程中, 可切换到 Current file 实时查看当前编写代码的问题.

### 1.3.2 maven

需配置数据库.

## 2 持续集成

### 2.1 travis-ci

#### 2.1.1 travis-ci 使用流程

travis-ci 结合 github 使用, 每次提交自动执行编译或测试任务.

1. <travis-ci.org> 用 github 账户授权登录
  2. travis-ci 上添加需要持续集成的 github repo
  3. github repo 中添加 .travis.yml 配置, 用以配置 travis 所要执行的任务和环境
  4. repo 更改完成, 提交, 可在 travis-ci 上实时查看任务执行过程
- 备注:

### 2.1.2 添加编译状态图标

```
1 [[Build Status](https://travis-ci.org/HereChen/Tools-Solutions.svg?  
  ↳ branch=master)](https://travis-ci.org/HereChen/Tools-Solutions)
```

### 2.1.3 travis 配置文件样本

此库的 .travis.yml 样本

```
1 sudo: required  
2 dist: trusty # ubuntu 14.04  
3  
4 # install  
5 before_install:  
6   - sudo apt-get update  
7   - sudo apt-get -y install texlive-full  
8   - sudo apt-get -y install pandoc  
9  
10 before_script:  
11   - chmod +x build.sh  
12  
13 script:  
14   - ./build.sh
```

## 3 版本控制

### 3.1 git 多账户

应用于多个代码托管站点多账户场景, 比如 github、gitlab、码云等<sup>12</sup>. 配置完成后方便多个账户同时使用. 以 github 和 gitlab 为例, 按照以下步骤完成配置. 除了以下提供的配置方法, Windows 10 上可以用 Bash On Ubuntu On Windows 额外添加一个 git 账户, 适用于只有两个账户的情况.

#### 3.1.1 生成公钥

生成公钥并添加到代码托管站点. 生成的时候, 输入不同的文件名保存.

```
1 ssh-keygen -t rsa -C "chenlei.here@qq.com"
```

<sup>1</sup>Multiple SSH Keys settings for different github account

<sup>2</sup>ssh hostname returns “Bad owner or permissions on ~/.ssh/config”

```
1 chmod 600 ~/.ssh/config
2 ssh-agent bash
3 ssh-add ~/.ssh/id_rsa_gitlab
4 ssh-add ~/.ssh/id_rsa_github
```

### 3.1.2 添加 ssh 配置文件

```
1 touch ~/.ssh/config
```

创建 config 文件, 并添加以下内容.

```
1 # gitlab
2 Host gitlab
3     HostName gitlab.com
4     User HereChen
5     IdentityFile ~/.ssh/id_rsa_gitlab
6
7 # github
8 Host github
9     HostName github.com
10    User HereChen
11    IdentityFile ~/.ssh/id_rsa_github
```

### 3.1.3 为项目设置单独的账户配置

```
1 git clone git@github.com:HereChen/Tools-Solutions.git
2 cd Tools-Solutions
3 git config user.name "HereChen"
4 git config user.email "chenlei.here@qq.com"
```

## 4 文档处理工具

### 4.1 Pandoc

Pandoc 主页 <http://pandoc.org>. Pandoc 可以实现不同格式文档的相互转换, 支持的格式包括 markdown、HTML、LaTeX、docx、EPUB ODT 等等.

### 4.1.1 markdown 多个文档转换

```
1 pandoc src/markdown/*.md -o build/latex/content.tex
```

### 4.1.2 markdown 转 LaTeX, 用 listings 包替换 verbatim

```
1 pandoc --listings src/markdown/*.md -o build/latex/content.tex
```

## 4.2 LaTeX

### 4.2.1 Windows 上编译缓慢, 经常卡在 eullmr.fd

删除 `texlive\2016\texmf-var\fonts\cache` 文件夹下字体缓存, 重新生成字体缓存, 执行 `texlive\2016\bin\win32\fc-cache.exe`<sup>3</sup>. 可更改示例代码的路径, 编译前执行一次.

```
1 del /q D:\applications\texlive\2016\texmf-var\fonts\cache\*
2 D:\applications\texlive\2016\bin\win32\fc-cache.exe
```

### 4.2.2 listings 代码换行

添加 `breaklines=true` 后, 还是存在没有空格的 url 无法换行, 需设置 `breakatwhitespace=false`. 另外, 可以在代码换行的地方添加箭头标识换行<sup>4</sup>.

```
1 \lstset{
2   breaklines=true,
3   % 箭头标识换行
4   postbreak=\raisebox{0ex}[0ex][0ex]{\ensuremath{\color{red}\hookrightarrow hookrightarrow\space}},
5   breakatwhitespace=false
6 }
```

## 5 编辑器及插件

### 5.1 Visual Studio Code

#### 插件

<sup>3</sup>LaTeX 编译卡顿怎么办?

<sup>4</sup>Pandoc: Markdown to PDF, without cutting off code block lines that are too long, Code not wrapping in listings even though `breaklines=true`, `lstlisting` line wrapping

- **Markdown TOC**, markdown 文本的目录生成, 可应用于 wiki 或博客目录生成.
- **REST Client**, 在文本编辑器中测试 API
- **Project Manager**, 类似于 Sublime Text 快捷切换项目目录

## 6 工具安装

### 6.1 nodejs

安装的方式包括: 从源码编译安装, 直接下载二进制, 依靠工具直接从库下载安装.

#### 从源码安装

```
1 curl -o node-v6.10.0.tar.gz https://nodejs.org/dist/v6.10.0/node-v6
   ↳ .10.0.tar.gz
2 tar -xvzf node-v6.10.0.tar.gz
3 cd node-v6.10.0
4 ./configure
5 make
6 make install
```

#### CentOS 下载二进制安装

```
1 # 64bit node install
2 curl -o node-v6.10.0-linux-x64.tar.xz https://nodejs.org/dist/v6.10.0/
   ↳ node-v6.10.0-linux-x64.tar.xz
3 mv /opt/
4 cd /opt/
5 tar -xvJf node-v6.10.0-linux-x64.tar.xz
6 mv node-v6.10.0-linux-x64 node
7
8 vi ~/.bash_profile
9 # 添加以下内容到 .bash_profile
10 # export NODE_HOME=/opt/node
11 # export PATH=$NODE_HOME/bin:$PATH
12 source ~/.bash_profile
13
14 node -v
```

#### Ubuntu 从库中下载安装

```
1 curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
```



```
2 sudo apt-get install -y nodejs
```

## 6.2 git

### 6.2.1 Ubuntu

安装最新版的 git<sup>5</sup>.

```
1 sudo add-apt-repository ppa:git-core/ppa
2 sudo apt-get update
3 sudo apt-get install git
```

如果上面执行报错 `Errors were encountered while processing: runit git`  
↪ `-daemon-run`, 尝试执行下面命令.

```
1 sudo apt-get purge runit
2 sudo apt-get purge git-all
3 sudo apt-get purge git
4 sudo apt-get autoremove
5 sudo apt update
6 sudo apt install git
```

### 6.2.2 CentOS 从源码安装 git

执行 `yum install git` 安装的版本比较老, 安装最新版可从源码安装<sup>6</sup>.

```
1 # 依赖工具安装
2 yum install curl-devel expat-devel gettext-devel openssl-devel zlib-
   ↪ devel
3 yum install gcc perl-ExtUtils-MakeMaker
4
5 # git 源码下载
6 # https://www.kernel.org/pub/software/scm/git/
7 curl -o git-2.9.3.tar.xz https://www.kernel.org/pub/software/scm/git/git
   ↪ -2.9.3.tar.xz
8 tar -xvJf git-2.9.3.tar.xz
9
10 # 编译安装 git
11 cd git-2.9.3
```

---

<sup>5</sup>Installing Latest version of git in ubuntu, How to fix processing with runit and git-daemon-run [duplicate]

<sup>6</sup>How to install latest version of git on CentOS 6.x/7.x , 起步 - 安装 Git

```
12 make prefix=/usr/local all
13 sudo make prefix=/usr/local install
```