Vi,Java,Ant,Junit,SonarQube的自学报告
Vi/Vim
vi和vim的区别
vi的使用
不同模式下常用的快捷键和命令
Ant
JUnit
Sonar及Sonar Runner

Vi,Java,Ant,Junit,SonarQube的自学报告

Vi/Vim

参考资料：《鸟哥的Linux私房菜》

为什么在Linux的世界中选择vi，vim

1. 可以快速地在终端中编辑属于纯文本文件的配置文件，方便系统管理员的使用。
2. vim有代码高亮，方便程序设计。
3. 内置的文本编辑器

vi和vim的区别

vim是从 vi 发展出来的一个文本编辑器 。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。和Emacs 并列成为类Unix系统用户最喜欢的编辑器。

vi的使用

三种模式：

1. 一般模式（默认，esc）
2. 编辑模式（按i、o键）
3. 命令行模式（按‘：’、‘/’、‘？’）

注意：编辑模式和命令行模式之间不能切换，必须先转换到一般模式

不同模式下常用的快捷键和命令

命令模式：

- :ZQ 无条件退出
    - :q! 无条件退出
        - :ZZ 存盘并退出

- :wq 存盘并退出

一般模式：

移动光标： **移动单词**

- h 左移一字符
    - l 右移一字符
- w/W 移动到下一单词的开头
- b/B 移动到上一单词的开头
    - e/E 移动到光标所在单词的末尾
- W、B、E 命令操作的单词是以空白字符（空格、Tab）分隔的字串，比如字符串“str1-str2 str3-str4”，对 W、B、E 命令来说是两个单词，而对 w、b、e 命令来说则是四个单词。

移动行

- j 下移一行
- k 上移一行
- 0 移到当前行开头
- ^ 移到当前行的第一个非空字符
- $ 移到当前行末尾
-) 移动到当前句子的末尾
    - (移动到当前句子的开头段落
- } 移动当前段落的末尾
- { 移到当前段落的开头

移动屏幕

- H 移动到屏幕的第一行
- M 移动到屏幕的中间一行
- L 移动到屏幕的最后一行

移动页

- Ctrl-f 向前滚动一页
- Ctrl-b 向后滚动一页
- Ctrl-u 向前滚动半页
    - Ctrl-d 向后滚动半页
    - 文件G 移动到文件末尾
- gg 移动到文件开头
- :0 移动到文件第一行
    - :$ 移动到文件最后一行0 为数字零（zero）

文本编辑：

- yw 复制当前单词从光标开始的部分

- yy 复制光标所在行的所有字符
- p 将最后一个删除或复制文本放在当前字符
- u 撤消更改
- Ctrl-R 重做更改
    - x向后删除一个字符
- X 向前删除一个字符
    - dd 删除游标所在的那一整行(常用)
    - d0 删除游标所在处，到该行的最前面一个字符
    - d$ 删除游标所在处，到该行的最后一个字符
- d1G 删除光标所在到第一行的所有数据
- dG 删除光标所在到最后一行的所有数据

学习要点： 在进行vim ～/.bashrc 更该环境变量时非常高效。 **注意：/etc/profile和～/.bashrc的区别**

1. bashrc与profile都用于保存用户的环境信息，bashrc用于交互式non-loginshell，而profile用于交互式login shell。
2. /etc/profile，/etc/bashrc 是系统全局环境变量设定，~/.profile，~/.bashrc用户家目录下的私有环境变量设定。
3. 更改/etc/profile 需要sudo，～/.bashrc不用。

export完环境变量后注意source ～/.bashrc 来激活变化。如果只是在bash shell命令行中export环境变量，则再次打开终端时则不会保存。

Ant

> Ant 是一个 Apache 基金会下的跨平台的构件工具，它可以实现项目的自动构建和部署等功能。

Ant 的构件文件是基于 XML 编写的，默认名称为 build.xml。ant 命令默认同一目录下的的 build.xml 文件。

基本用法：

- `<?xml version="1.0" encoding="big5">` 定义xml版本,编码
- `<project name="projectA" default="clean" basedir="." >` 定义了工程名,默认方式为清除,如清楚某些文件,basedir为当前目录.
- 设定全局变量 如: `<property name="date" value="2007.03.22">` 这样下面的可以直接调用变量date.
- 定义文件编译所需要的jar档. `

```
    <fileset dir="路径" includes="*.jar"/>
```

`
- target 元素，它为 Ant 的基本执行单元，它可以包含一个或多个具体的任务。 `<target name="targetB" depends="targetA" unless="amigo">`

- depends 属性用于描述 target 之间的依赖关系，若与多个 target 存在依赖关系时，需要以“,”间隔。
- basedir：project 基目录的绝对路径。
- ant.file：buildfile 的绝对路径。

  学习要点：
- 注意各命令之间的依赖关系以及文件构建的目录。
- 在完成了build 和 run的基础上，加入junit命令，首先在编译时导入junit包，然后同时编译Test类和被测试的类，同样，在run时也需要先导入junit包。
- 和标签中，都可以使用fileset, dirset, pathelement等子标签来指明jar/zip, classes目录等。它们之间的区别如下：
- 可以独立定义，并赋予id供其它或引用。不能独立定义，但可以通过ID引用。classpath主要用于编译和运行时，指定具体的类路径。path则类似于一个的路径变量。
- 在和标签中都可以使用子标签来指明要包含的类文件目录和jar/zip文件。有两个属性：location, path，这两个属性在一个pathelement标签中只能出现一个。它们的区别为：location是用于指明一个包含类文件的目录，或者一个jar/zip文件(注意：重点强调是一个！)；而path则能指定多个目录或jar/zip文件，多个目录或jar/zip文件之间用":",或";"。

```
<?xml version="1.0" encoding="UTF-8" ?>
<project name="helloworld" default="test" basedir="..">
    <property name="lib.path" value="lib"/>
    <path id="compile.path">
        <fileset dir="${lib.path}">
            <include name="**/*.jar"/>
        </fileset>
        <pathelement path="bin/classes"/>
    </path>
    <path id="run.path">
        <path refid="compile.path" />
        <pathelement location="org" />
    </path>
    <target name="clean">
        <delete dir="bin"/>
    </target>
    <target name="compile" depends="clean">
        <mkdir dir="bin/classes"/>
        <javac srcdir="src" destdir="bin/classes" classpathref="compile.path"
includeantruntime="false" />

    </target>
    <target name="run" depends="compile">
        <java classname="helloworld">
            <classpath refid="run.path" />
        </java>
    </target>
    <target name="test" depends="clean, compile">
        <junit printsummary="true">
            <classpath refid="compile.path"/>
            <test name="helloworldTest"/>
        </junit>
    </target>
</project>
```

JUnit

> 单元测试的目的: 测试当前所写的代码是否是正确的, 例如输入一组数据, 会输出期望的数据; 输入错误数据, 会产生错误异常等. 在单元测试中, 我们需要保证被测系统是独立的, 即当被测系统通过测试时, 那么它在任何环境下都是能够正常工作的. 编写单元测试时, 仅仅需要关注单个类就可以了.

用法：

1. @Test (expected = Exception.class) 表示预期会抛出Exception.class 的异常
2. @Ignore 含义是“某些方法尚未完成，暂不参与此次测试”。这样的话测试结果就会提示你有几个测试被忽略，而不是失败。一旦你完成了相应函数，只需要把@Ignore注解删去，就可以进行正

常的测试。
3. @Test(timeout=100) 表示预期方法执行不会超过 100 毫秒，控制死循环
4. @Before 表示该方法在每一个测试方法之前运行，可以使用该方法进行初始化之类的操作
5. @After 表示该方法在每一个测试方法之后运行，可以使用该方法进行释放资源，回收内存之类的操作
6. @BeforeClass 表示该方法只执行一次，并且在所有方法之前执行。一般可以使用该方法进行数据库连接操作，注意该注解运用在静态方法。
7. @AfterClass 表示该方法只执行一次，并且在所有方法之后执行。一般可以使用该方法进行数据库连接关闭操作，注意该注解运用在静态方法。

```
public class HelloWorldTest {
     public HelloWorld helloworld = new HelloWorld( );
     @Test
     Public void testHello() {
     helloworld.hello();
     assertEquals( "Hello World!" , helloworld.getStr() );
     }
}
```

注意： 如果没有在测试类中加入main函数，则需要在运行时加入 `-ea org.junit.runner.JUnitCore` 参数。

或者在测试类加入

```
public static void main(String[] args) {
         org.junit.runner.JUnitCore.main("helloworldTest");
     }
```

设置环境变量：

```
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_91
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:/opt/resources/junit-
4.9.jar:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH:$HOME/bin
```

JUnit FAQ

Sonar及Sonar Runner

> Sonar 是一个用于代码质量管理的开源平台，用于管理源代码的质量，可以从七个维 度检测代码质量。通过插件形式，可以支持包括java,C#,C/C++,PL/SQL,Cobol,JavaScrip,Groovy 等等二十几种编程语言的代码质量管理与检测。

检测内容：

1. 糟糕的复杂度分布

2. 重复
3. 缺乏单元测试
4. 没有代码标准
5. 没有代码标准
6. 潜在的 bug
7. 糟糕的设计（原文 Spaghetti Design，意大利面式设计）

Spaghetti Design

用法：

添加环境变量

```
export SONAR_HOME=/home/administrator/Downloads/sonar-3.7.4/bin/linux-x86-64
export SONAR_RUNNER_HOME=/home/administrator/Downloads/sonar-runner-2.4
export PATH=$SONAR_RUNNER_HOME/bin:$PATH
```

```
cd SONAR_HOME
./sonar.sh start   //启动服务
./sonar.sh stop    //停止服务
./sonar.sh restart //重启服务
```

访问http:\localhost:9000，如果显示 SonarQube 的项目管理界面，表示安装成功。

在项目源码的根目录下创建 sonar-project.properties 配置文件，其中 projectKey 是项目的唯一标识，不能重复；要修改的内容包括 sonar.projectKey，sonar.projectName，java-module.sonar.projectBaseDir 三项；

编写好sonar-project.properties 文件后，在 shell 里面进入含有 sonar-project.properties文件的目录，输入 sonar-runner，运行测试；

出现以上信息时，证明运行成功，打开http:\localhost:9000 查看对应项目的情况即可。 注意：每次使用完 Sonar，记得关闭，进入启动目录，./sonar.sh stop

[官方文档](http://docs.codehaus.org/display/SONAR/Analysis+Parameters)

总结：

1. 用vi配置环境变量文件的方法。
2. ant和junit是用java语言进行项目设计的高效工具，类似于c++语言的makefile和debug工具。
3. SonarQube很强大可以对源码的质量进行检测，方便日后的开发和维护。