

晶传美软件项目持续集成插件系统

CI-PLUGINS 使用说明 (V1.0)

北京晶传科技有限公司
2013 年 6 月

目录

1 编写目的.....	4
2 持续集成 (Continuous Integration)	4
2.1 为什么要持续集成.....	4
2.2 什么是持续集成.....	5
2.3 持续集成的价值.....	5
2.3.1 减少风险.....	5
2.3.2 减少重复过程.....	5
2.3.3 任何时间、任何地点生成可部署的软件.....	6
2.3.4 增强项目的可见性.....	6
2.3.5 建立团队对开发产品的信心.....	6
2.4 持续集成怎么做.....	6
2.5 持续集成的基本原则.....	7
3 软件构建生命周期 (Build Life cycles)	7
3.1 生命周期.....	7
3.1.1 构建 (Default)	7
3.1.2 清理 (Clean)	7
3.1.3 生成文档 (Site)	7
3.2 默认构建步骤 (phase)	8
3.2.1 验证 (Validate)	8
3.2.3 编译 (Compile)	8
3.2.4 单元测试 (Test)	8
3.2.5 打包 (Package)	8
3.2.6 集成测试 (Integration test)	8
3.2.7 质量检测 (Verify)	8
3.2.8 安装 (Install)	8
3.2.9 发布 (Deploy)	8
3.3 构建生命周期图示.....	8
4 相关工具介绍.....	8
4.1 Jenkins.....	8
4.2 Ant 和 Ivy	9
4.3 Maven.....	9
4.4 JUnit.....	9
4.5 Cobertura.....	9
4.6 JMeter.....	9
4.7 Capistrano	10
4.8 JIRA	10
5 集成插件使用介绍	10
5.1 插件特点.....	10
5.2 快速开始 (Quick Start)	11
5.2.1 下载.....	11
5.2.1.1 Windows 系统.....	11
5.2.1.2 Linux 系统.....	11
5.2.2 修改配置.....	12

5.2.2.1 修改 ivy.xml 文件	12
5.2.2.2 修改 build.xml 文件	13
5.2.3 运行 ant.....	13
5.2.4 与 Eclipse 集成	13
5.3 目录结构规范.....	14
5.4 集成插件配置详解.....	15
5.5 图解集成插件使用流程.....	16
5.5.1 从 http://192.168.7.244:8080/ 下载 create 脚本文件	16
5.5.2 修改脚本的权限.....	17
5.5.3 创建一个空的 JAVA 项目	18
5.5.4 通过 ant 运行单元测试.....	18
5.5.5 创建本地的 Git 仓库	20
5.5.6 创建团队共享的 Git 仓库（需要项目管理员来操作）	20
5.5.7 推送代码到共享的 Git 仓库中	20
5.5.8 创建一个自己的工作分支.....	21
5.5.9 配置 jenkins	21
5.5.10 编写你的第一个 Java 类.....	23
5.5.11 编写你的第一个单元测试类.....	25
5.5.12 执行代码覆盖率测试.....	26
5.5.13 源码规范性检查.....	27
5.5.14 提交代码到本地仓库.....	28
5.5.15 提交代码到共享 Git 仓库	29
6 Git 仓库使用说明	29
6.1 Git 快速入门	30
7 常见问题.....	30

1 编写目的

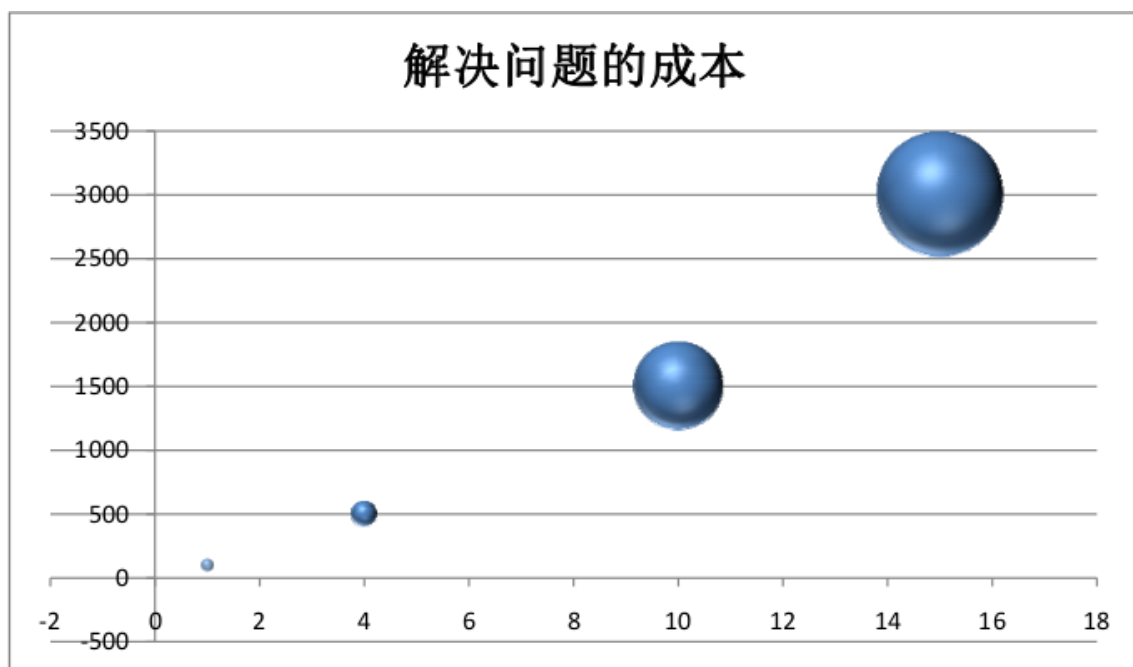
“晶传美软件项目持续信息插件系统”（以下简称集成插件），主要是为了在项目执行过程中减少开发风险、减少重复性工作过程、增强项目的可见性以及建立团队对开发产品的信心，而制定的一系列规范及按照这些规范执行项目构建过程和集成过程的工具集。这些工具集以插件的形式存在于各个软件研发项目的集成开发环境中，通过这些工具，可以使软件项目在任何时间、任何地点生成部署用的工件（artifact）和可执行的软件包。

本文档在后续篇幅中通过对软件的自动构建过程、持续集成过程和相关工具的介绍来帮助软件开发人员和其他项目成员理解持续集成的基本概念，以及如何正确的下载、安装、使用和扩展集成插件。

2 持续集成（Continuous Integration）

2.1 为什么要持续集成

集成软件的过程不是新问题，如果项目开发的规模比较小，比如一个人的项目，如果它对外部系统的依赖很小，那么软件集成不是问题，但是随着软件项目复杂度的增加（即使增加一个人），就会对集成和确保软件组件能够在一起工作提出了更多的要求-要早集成，常集成。早集成，频繁的集成帮助项目在早期发现项目风险和质量问题，如果到后期才发现这些问题，解决问题代价很大，很有可能导致项目延期或者项目失败。



2.2 什么是持续集成

大师 **Martin Fowler** 对持续集成是这样定义的：持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽快地发现集成错误。许多团队发现这个过程可以大大减少集成的问题，让团队能够更快的开发内聚的软件。

2.3 持续集成的价值

2.3.1 减少风险

一天中进行多次的集成，并做了相应的测试，这样有利于检查缺陷，了解软件的健康状况，减少假定。

2.3.2 减少重复过程

减少重复的过程可以节省时间、费用和工作量。说起来简单，做起来难。这些浪费时间的重复劳动可能在我们的项目活动的任何一个环节发生，包括代码编译、数据库集成、测试、审查、部署及反馈。通过自动化的持续集成可以将这些重复的动作都变成自动化的，无需太多人工干预，让人们的时间更多的投入到动

脑筋的、更高价值的事情上。

2.3.3 任何时间、任何地点生成可部署的软件

持续集成可以让您在任何时间发布可以部署的软件。从外界来看，这是持续集成最明显的好处，我们可以对改进软件品质和减少风险说起来滔滔不绝，但对于客户来说，可以部署的软件产品是最实际的资产。利用持续集成，您可以经常对源代码进行一些小改动，并将这些改动和其他的代码进行集成。如果出现问题，项目成员马上就会被通知到，问题会第一时间被修复。不采用持续集成的情况下，这些问题有可能到交付前的集成测试的时候才发现，有可能会延迟发布产品，而在急于修复这些缺陷的时候又有可能引入新的缺陷，最终可能导致项目失败。

2.3.4 增强项目的可见性

持续集成让我们能够注意到趋势并进行有效的决策。如果没有真实或最新的数据提供支持，项目就会遇到麻烦，每个人都会提出他最好的猜测。通常，项目成员通过手工收集这些信息，增加了负担，也很耗时。持续集成可以带来两点积极效果：

(1)有效决策：持续集成系统为项目构建状态和品质指标提供了及时的信息，有些持续集成系统可以报告功能完成度和缺陷率。

(2)注意到趋势：由于经常集成，我们可以看到一些趋势，如构建成功或失败、总体品质以及其它的项目信息。

2.3.5 建立团队对开发产品的信心

持续集成可以建立开发团队对开发产品的信心，因为他们清楚的知道每一次构建的结果，他们知道他们对软件的改动造成了哪些影响，结果怎么样。

2.4 持续集成怎么做

1) 构建统一的代码库

2) 实现自动构建

3) 实现自动测试

4) 每个人每天都要向代码库提交代码, 并由 **Leader** 合并到主干上

5) 每次代码递交后都会在持续集成服务器上触发一次构建

6) 保证快速构建

7) 模拟生产环境的自动测试

8) 每个人都可以很容易的获取最新可执行的应用程序

9) 每个人都清楚正在发生的状况

10) 实现自动化的部署

2.5 持续集成的基本原则

1) 所有的开发人员需要在本地机器上做本地构建, 然后再提交的版本控制库中, 从而确保他们的变更不会导致持续集成失败。

2) 开发人员每天至少向版本控制库中提交一次代码。

3) 开发人员每天至少需要从版本控制库中更新一次代码到本地机器。

4) 需要有专门的集成服务器来执行集成构建, 每天要执行多次构建。

5) 每次构建都要 100% 通过。

6) 每次构建都可以生成可发布的产品。

7) 修复失败的构建是优先级最高的事情。

3 软件构建生命周期 (Build Life cycles)

3.1 生命周期

3.1.1 构建 (Default)

用于生成项目的工作件。

3.1.2 清理 (Clean)

用于清理旧的构建文件和临时文件。

3.1.3 生成文档 (Site)

用于生成项目文档。

3.2 默认构建步骤（phase）

3.2.1 验证（Validate）

验证项目所有的必要的信息是否可用。

3.2.3 编译（Compile）

编译项目源码。

3.2.4 单元测试（Test）

使用一个合适的单元测试框架来测试编译好的源代码。这些测试执行时不用把源码打包，也不用部署。

3.2.5 打包（Package）

将编译好的代码打包成可以发布的格式，如 JAR。

3.2.6 集成测试（Integration test）

将打好的包部署到一个可以运行的集成环境中进行测试。

3.2.7 质量检测（Verify）

检查证实该发布包是有效的并符合质量标准的。

3.2.8 安装（Install）

将发布包安装到本地仓库中，以便用作其他本地项目中的一个依赖库。

3.2.9 发布（Deploy）

在一个集成或者发布环境中，为了与其他的开发人员和项目进行共享，将最终版的发布包复制到远程仓库中。

3.3 构建生命周期图示

4 相关工具介绍

4.1 Jenkins

Jenkins，之前叫做 Hudson，是基于 Java 开发的一种持续集成工具，用于监控有序且重复的工作，包括：

- 1) 持续的构建和测试软件项目。
- 2) 监控各种外部运行的执行器（插件）的工作。

详情可参考：<https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins>

4.2 Ant 和 Ivy

Apache Ant 是一种基于 Java 的 build 工具。理论上来说，它有些类似于（Unix）C 中的 make。Apache Ivy 是 Apache Ant 下的一个子项目，最新版本是 2009 年 1 月 20 日发布的 2.0.0 正式版。Apache Ivy 是一个优秀的管理（记录、跟踪、解析和报告）项目依赖的工具，提供了强大的依赖管理功能，可与 Apache Ant 紧密集成。

详情参考：<http://ant.apache.org>

4.3 Maven

Maven 是基于项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。

4.4 JUnit

JUnit 是一个 Java 语言的单元测试框架。它由 Kent Beck 和 Erich Gamma 建立，逐渐成为源于 Kent Beck 的 sUnit 的 xUnit 家族中为最成功的一个。JUnit 有它自己的 JUnit 扩展生态圈。

4.5 Cobertura

Cobertura 是一个免费的 Java 代码覆盖率测试工具。

4.6 JMeter

Apache JMeter 是 Apache 组织开发的基于 Java 的压力测试工具。用于对软件做压力测试，它最初被设计用于 Web 应用测试但后来扩展到其他测试领域。它可以用于测试静态和动态资源例如静态文件、Java 小服务程序、CGI 脚本、Java 对象、数据库，FTP 服务器，等等。JMeter 可以用于对服务器、网络或对象模拟巨大的负载，来在不同压力类别下测试它们的强度和分析整体性能。

另外，JMeter 能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证你的程序返回了你期望的结果。为了最大限度的灵活性，JMeter 允许使用正则表达式创建断言。

4.7 Capistrano

Capistrano 是一种在多台服务器上运行脚本的开源工具，它主要用于部署 web 应用。它自动完成多台服务器上新版本的同步更新，包括数据库的改变。

Capistrano 最初由 Jamis Buck 用 Ruby 开发，并用 RubyGems 部署渠道部署。现在 Capistrano 不仅限于应用 Ruby on Rails 的 web 应用框架，而且可以用于部署用其他框架的 web 应用程序，比如用 PHP 开发的。Capistran 最初是用来应用于 bash 指令行。现在 Ruby on Rails 框架的用于也可以使用它的新特性，例如，对当前 web 应用部署改变使其更新版本，或者使其回滚到之前的旧版本。

Capistran 最初叫 SwitchTower，由于商标争端于 2009 年二月 24 日改为 Capistran。而且原始作者于当天宣布不再是该软件的维护者。

4.8 JIRA

JIRA 是 Atlassian 公司出品的项目与事务跟踪工具，被广泛应用于缺陷跟踪、客户服务、需求收集、流程审批、任务跟踪、项目跟踪和敏捷管理等工作领域。

JIRA 中配置灵活、功能全面、部署简单、扩展丰富，其超过 150 项特性得到了全球 115 个国家超过 19,000 家客户的认可。

5 集成插件使用介绍（CI-Plugins）

5.1 插件特点

1) 安装简单：只要将下载的 jpt-plugin-x.jar 解压到项目根目录，并修改名称为 plugins 即可。

2) 配置简单：拷贝 plugins/default/文件夹下的 build.xml/ivy.xml/project.xml 到项目根目录下，并修改项目名称和版本号。

3) 通用构建步骤整合：在 plugins/build-common.xml 中包含了常用的构建任

务，只要复合项目规范的文件目录结构，即可调用。

4) 支持生成 JUnit 测试报告：运行 ant 即可生成 HTML 格式的报告到指定的文件夹中。

5) 支持代码覆盖率测试：集成了 Cobertura 工具，可生成代码覆盖率报告。

6) 支持性能测试：集成了 JMeter 性能测试框架。

7) 自动打包发布：通过调用“ant install”构建任务可以将项目打包并发布到本地共享给本地的其他项目使用；通过调用“ant deploy”构建任务可以将项目打包并发布到指定的共享服务器上。

8) 自动更新：通过配置 project.properties 里的 plugins.auto.update 选项，可以将插件设置为自动更新模式，前提是不能改变 plugins 文件夹里的内容。

5.2 快速开始 (Quick Start)

5.2.1 下载

5.2.1.1 Windows 系统

下载地址：<http://192.168.7.244:8080/job/Jcm-Project-Template/lastSuccessfulBuild/artifact/dist/jpt-plugins-1.0.tar.gz>

下载后直接用 WinRAR 工具解压缩，然后将解压后的文件夹名称 jpt-plugins-x 修改为项目名称即可。

5.2.1.2 Linux 系统

方式一，执行以下代码：

```
$> mkdir -p path/to/projectname
```

```
$> wget http://192.168.7.244:8080/job/Jcm-Project-Template/lastSuccessfulBuild/artifact/dist/jpt-plugins.tar.gz
```

```
$> tar -vzxf jpt-plugins.tar.gz
```

```
$> mv jpt-plugins-1.0 path/to/your_project_name
```

方式二，执行以下代码：

```
$> wget http://192.168.7.244:8080/job/Jcm-Project-Template/ws/plugins/create
```

```
$> chmod 775 create
```

```
$> ./create projectName -t jar
```

5.2.2 修改配置

5.2.2.1 修改 ivy.xml 文件

在信息节点（`<info></info>`）里修改组织名称（`organisation`）、项目名称（`module`）、版本号（`revision`）和添加项目成员。

也可以在项目描述节点（`<description></description>`）下添加项目描述信息。依赖关系节点（`<dependencies></dependencies>`）用来添加该项目所依赖的其他项目模块。

```
<ivy-module version="2.0">
  <info organisation="com.jcm" module="Jcm-Project-Template" revision="1.0">
    <ivyauthor name="Zhong Lizhi" url="mailto:zhonglizhi@8chedao.com" />
    <ivyauthor name="Xu Huiyao" url="mailto:xuhuiyao@8chedao.com" />
    <description>
      Describes the current module. This tag is the only one which can contain
      free text, including html. It is used to describe the module itself,
      usually in a single short phrase(it is not meant to replace the module
      description on the corresponding web site), and then gives all information
      necessary to use the module, especially information about public
      configurations, how and when to use them.
    </description>
  </info>
  <publications>
    <artifact name="${ivy.module}" />
  </publications>

  <dependencies defaultconf="*->default">
    <dependency org="org.hamcrest" name="hamcrest-all" rev="1.3" />
    <dependency org="junit" name="junit" rev="4.8.2" transitive="false" />
    <!-- 在这里添加项目的依赖包 -->
```

```
<!-- globe exclude -->
<exclude org="*" ext="*" type="source" />
<exclude org="*" ext="*" type="javadoc" />
</dependencies>
</ivy-module>
```

5.2.2.2 修改 build.xml 文件

将项目节点（project）里的项目名称属性（name）的值修改为对应的项目名称。

```
<project name="Jcm-Project-Template" default="test" basedir=".">
  <description>
    project description
  </description>
  <property file="project.properties" />
  <import file="plugins/build.xml" as="phases" />
  <!-- 在这里可以添加自定义的构建任务，也可以覆盖已有的构建任务 -->
</project>
```

5.2.3 运行 ant

编译并测试

\$> ant

打包

\$> package

发布到本地

\$> install

发布到共享资源库

\$> deploy

5.2.4 与 Eclipse 集成

拷贝 `plugins/eclipse/jar` 目录下的 `.project` 和 `.classpath` 文件到项目根目录下即可。如果没有安装 Apache IvyDE 插件，则要手动将 `lib` 下的 `jar` 添加到项目构建路径中。

5.3 目录结构规范

- + `ProjectName` 项目根目录
 - + `src` 项目源码
 - + `conf` 配置文件
 - + `test` 测试源码
 - + `unit` 单元测试的源代码
 - + `verify` 质量测试的源代码
 - + `integration` 集成测试源代码
 - + `plugins` 集成插件
 - + `lib` 项目依赖的 Jars
 - + `build` 编译后的 class 文件
 - + `main` 项目源码的 class 文件
 - + `test` 测试源码的 class 文件
 - + `dist` 项目发布路径
 - + `publish` 项目部署脚本
 - + `WebContent` Web 项目根目录（非 Web 项目没有该文件夹）
- + `.git` Git 版本控制文件
- `.gitignore` 版本控制需要忽略的文件配置
- `build.xml` Ant 构建用文件
- `ivy.xml` 项目信息及依赖关系管理
- `project.properties` 项目构建用配置文件
- `.project` Eclipse 集成开发环境的项目配置文件

- .classpath Eclipse 集成开发环境的 CLASSPATH 环境配置
- pom.xml Maven 配置

5.4 集成插件配置详解

```
## 全局变量
## 包括项目文件目录结构的设置和对系统环境变量的引用。
ECLIPSE_HOME=d:/eclipse
debuglevel=source,lines,vars
target=1.6
source=1.6
encoding=UTF-8

## 项目环境设置
publish.dir=publish
project.name=${ ant.project.name}

## jar or war
package.type=jar
version=0.1.0
web.root=WebContent

## source code directories
src.dir=${ basedir}/src
src.core=${ basedir}/src
src.extends=${ basedir}/extends
src.test=${ basedir}/test
lib.dir=${ basedir}/lib

## test code directories
test.dir=${ src.test}
test.unit.dir=${ test.dir}/unit
test.verify.dir=${ test.dir}/verify
test.integration.dir=${ test.dir}/integration

##
```

```
build.dir=${basedir}/build
build.src.dir=${build.dir}/main
build.test.dir=${build.dir}/test
dist.dir=${basedir}/dist
dist.lib=${dist.dir}/${project.name}/lib
```

各种报告输出设置(Junit/Jmeter/Cobertura)

```
target.report.dir=${dist.dir}/${project.name}/report
junit.report.xml.dir=${target.report.dir}/junit-xml
junit.report.html.dir=${target.report.dir}/junit-html
```

Ivy

```
#ivy.cache.dir=${basedir}/.cache
ivy.settings.dir=${basedir}/plugins/ivy/settings
ivy.settings.file=${ivy.settings.dir}/ivysettings.xml
```

Cobertura

```
cobertura.home=/home/kfs/sourceforge/cobertura-1.9.4.1
```

Junit

```
# Print one-line statistics for each testcase. Can take the values on, off,
# and withOutAndErr. withOutAndErr is the same as on but also includes the
# output of the test as written to System.out and System.err.
junit.printsummary=on
```

Shared Resolve url

```
resolve.shared.url = http://192.168.7.244:8080/job/Jcm-Project-Template/ws/plugins/ivy/settings/ivysettings-shared.xml
```

Auto Update Plugins

```
plugins.auto.update = false
plugins.checksum.url = http://192.168.7.244:8080/job/Jcm-Project-Template/lastSuccessfulBuild/artifact/dist/jpt-plugins.MD5
```

5.5 图解集成插件使用流程

5.5.1 从 <http://192.168.7.244:8080/> 下载 create 脚本文件

\$> wget <http://192.168.7.244:8080/job/Jcm-Project-Template/ws/plugins/create>

```
zhong@X201: ~/projects
zhong@X201:~/projects$ wget $> wget http://192.168.7.244:8080/job/Jcm-Project-Template/ws/plugins
/create
--2013-06-05 16:13:44-- http://$/
Resolving $ ($)... failed: No such file or directory.
wget: unable to resolve host address '$'
--2013-06-05 16:13:44-- http://192.168.7.244:8080/job/Jcm-Project-Template/ws/plugins/create
Connecting to 192.168.7.244:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1424 (1.4K) [application/octet-stream]
Saving to: 'create'

100%[=====>] 1,424      --.-K/s   in 0.005s

2013-06-05 16:13:44 (279 KB/s) - 'create' saved [1424/1424]

FINISHED --2013-06-05 16:13:44--
Total wall clock time: 0.04s
Downloaded: 1 files, 1.4K in 0.005s (279 KB/s)
zhong@X201:~/projects$ ls
2013                highjs-ex          monitor-ui-n       web-template
8chedao-nodejs     ivy-deploy-tool   monitor-ui-n.tar.gz wget
auto-management-sys ivytest           my_app            ztools-company
build.xml.template javabroker        python-test       z-xml
create             Jcm-Project-Template RemoteSystemsTempFiles
hellojava          monitor_service   robot
hellojava.tar.gz   monitor-ui        Servers
zhong@X201:~/projects$
```

5.5.2 修改脚本的权限

\$> chmod 775 create

```

zhong@X201: ~/projects
Length: 1424 (1.4K) [application/octet-stream]
Saving to: 'create'

100%[=====>] 1,424      --.-K/s   in 0.005s

2013-06-05 16:13:44 (279 KB/s) - 'create' saved [1424/1424]

FINISHED --2013-06-05 16:13:44--
Total wall clock time: 0.04s
Downloaded: 1 files, 1.4K in 0.005s (279 KB/s)
zhong@X201:~/projects$ ls
2013                                highjs-ex                          monitor-ui-n                       web-template
8chedao-nodejs                     ivy-deploy-tool                   monitor-ui-n.tar.gz               wget
auto-management-sys                 ivytest                           my_app                             ztools-company
build.xml.template                  javabroker                        python-test                        z-xml
create                              Jcm-Project-Template              RemoteSystemsTempFiles
hellojava                           monitor_service                   robot
hellojava.tar.gz                   monitor-ui                         Servers
zhong@X201:~/projects$ chmod 775 create
zhong@X201:~/projects$ ls
2013                                highjs-ex                          monitor-ui-n                       web-template
8chedao-nodejs                     ivy-deploy-tool                   monitor-ui-n.tar.gz               wget
auto-management-sys                 ivytest                           my_app                             ztools-company
build.xml.template                  javabroker                        python-test                        z-xml
create                              Jcm-Project-Template              RemoteSystemsTempFiles
hellojava                           monitor_service                   robot
hellojava.tar.gz                   monitor-ui                         Servers
zhong@X201:~/projects$

```

5.5.3 创建一个空的 JAVA 项目

\$> ./create my_app -t jar

```

zhong@X201:~/projects$ ./create my_app -t jar
--2013-06-05 16:45:28-- http://192.168.7.244:8080/job/Jcm-Project-Template/lastSuccessfulBuild/artifact/dist/jpt-plugins.tar.gz
Connecting to 192.168.7.244:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1907795 (1.8M) [application/x-gzip]
Saving to: 'jpt-plugins.tar.gz'

100%[=====>] 1,907,795    2.14MB/s   in 0.9s

2013-06-05 16:45:29 (2.14 MB/s) - 'jpt-plugins.tar.gz' saved [1907795/1907795]

$download_file is downloaded!
jpt-plugins/plugins/
jpt-plugins/plugins/checkstyle/
jpt-plugins/plugins/cobertura/
jpt-plugins/plugins/eclipse/
jpt-plugins/plugins/eclipse/jar/
jpt-plugins/plugins/eclipse/jar/.externalToolBuilders/
jpt-plugins/plugins/jmeter/schematic.xml
jpt-plugins/plugins/jmeter/startup.bsh
jpt-plugins/plugins/jpt-plugins.MD5
jpt-plugins/plugins/project.properties
jpt-plugins/plugins/update
temp file is deleted!
initialized project configure!
my_app is create!
zhong@X201:~/projects$

```

5.5.4 通过 ant 运行单元测试

\$> cd my_app && ant

```
zhong@X201:~/projects$ cd my_app/ && ant
Buildfile: /home/lizhi/workspace/2013/my_app/build.xml

buildclean:

distclean:

common.cleanall:

_cobertura_clean:

common.init:
[mkdir] Created dir: /home/lizhi/workspace/2013/my_app/build/main
[mkdir] Created dir: /home/lizhi/workspace/2013/my_app/build/test
[mkdir] Created dir: /home/lizhi/workspace/2013/my_app/dist
[mkdir] Created dir: /home/lizhi/workspace/2013/my_app/publish
[echo] Auto Update Plugins Option is: true
[echo] Is Update: true
[echo] Import Warning: No found cobertura home!, isRunCoverage is ${isRunCoverage}

common.resolve:
[echo] ivysettings-shared.xml
[ivy:retrieve] :: Apache Ivy 2.3.0 - 20130110142753 :: http://ant.apache.org/ivy/ ::
[ivy:retrieve] :: loading settings :: file = /home/lizhi/workspace/2013/my_app/plugins/ivy/setting
gs/ivysettings.xml
[ivy:retrieve] :: resolving dependencies :: com.jcm#my_app;1.0
[ivy:retrieve]   confs: [default]

[ivy:retrieve]   confs: [default]
[ivy:retrieve]   found org.hamcrest#hamcrest-all;1.3 in public
[ivy:retrieve]   found junit#junit;4.8.2 in public
[ivy:retrieve] :: resolution report :: resolve 128ms :: artifacts dl 3ms
-----
|                  | modules                               || artifacts |
|      conf        | number| search|dwnlded|evicted|| number|dwnlded|
-----+-----+-----+-----+-----+-----+-----+-----+
|      default     |      2 |      0 |      0 |      0 ||      2 |      0 |
-----+-----+-----+-----+-----+-----+-----+
[ivy:retrieve] :: retrieving :: com.jcm#my_app
[ivy:retrieve]   confs: [default]
[ivy:retrieve] 2 artifacts copied, 0 already retrieved (531kB/5ms)

validate:

common.compile:

compile:

common.compile-test:
[echo] my_app: /home/lizhi/workspace/2013/my_app/build.xml

common.unit-test:

coverage:

junit-test:

_run-test:
[mkdir] Created dir: /home/lizhi/workspace/2013/my_app/dist/my_app/report/junit-xml

test:

BUILD SUCCESSFUL
```

5.5.5 创建本地的 Git 仓库

\$> git init

```
zhong@X201:~/projects/my_app$ git init
Initialized empty Git repository in /home/lizhi/workspace/2013/my_app/.git/
```

\$> git add .

\$> git commit -m 'Initialized my_app'

```
zhong@X201:~/projects/my_app$ git commit -m 'Initialized my_app'
[master (root-commit) ad48e1e] Initialized my_app
68 files changed, 5338 insertions(+)
create mode 100644 .gitignore
create mode 100644 build.xml
create mode 100644 ivy.xml
create mode 100644 plugins/ant2ide.jar
create mode 100644 plugins/build-common.xml
create mode 100644 plugins/build.xml
create mode 100644 plugins/checkstyle/checkstyle-5.6-all.jar
create mode 100644 plugins/checkstyle/checkstyle-simple.xsl
create mode 100644 plugins/checkstyle/sun_checks.xml
create mode 100644 plugins/cobertura/README
create mode 100644 plugins/cobertura/build.properties
create mode 100644 plugins/cobertura/build.xml
create mode 100644 plugins/cobertura/build.xml.bak
create mode 100644 plugins/cobertura/build_with_specific_data_file.xml
create mode 100644 plugins/cobertura/coverage.properties
create mode 100644 plugins/create
create mode 100644 plugins/eclipse/ivy.properties
create mode 100644 plugins/eclipse/jar/.classpath
create mode 100644 plugins/eclipse/jar/.externalToolBuilders/AntBuilder.launch
```

\$> git status

```
zhong@X201:~/projects/my_app$ git status
# On branch master
nothing to commit, working directory clean
zhong@X201:~/projects/my_app$
```

5.5.6 创建团队共享的 Git 仓库（需要项目管理员来操作）

\$> ssh [kfs@192.168.7.242](ssh://kfs@192.168.7.242)

\$> cd code-library

\$> mkdir my_app.git && cd my_app.git

\$> git init --bare

\$> exit

5.5.7 推送代码到共享的 Git 仓库中

\$> git remote add origin ssh://kfs@192.168.7.242/home/kfs/code-library/my_app

\$> git status

\$> git push origin master

```
zhong@X201:~/projects/my_app$ git remote add origin ssh://kfs@192.168.7.242/home/kfs/code-library/my_app
zhong@X201:~/projects/my_app$ git status
# On branch master
nothing to commit, working directory clean
zhong@X201:~/projects/my_app$ git push origin master
kfs@192.168.7.242's password:
Counting objects: 79, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (77/77), done.
Writing objects: 100% (79/79), 1.83 MiB, done.
Total 79 (delta 16), reused 0 (delta 0)
To ssh://kfs@192.168.7.242/home/kfs/code-library/my_app
 * [new branch]      master -> master
zhong@X201:~/projects/my_app$
```

5.5.8 创建一个自己的工作分支

为了更好的防止代码冲突，可以创建一个自己的工作分支，将 master 分支成为一个远程仓库的镜像。

\$> git branch zlz master

\$> git branch -a

\$> git checkout zlz

\$> git status

```
zhong@X201:~/projects/my_app$ git branch zlz master
zhong@X201:~/projects/my_app$ git status
# On branch master
nothing to commit, working directory clean
zhong@X201:~/projects/my_app$ git branch -a
* master
  zlz
remotes/origin/master
zhong@X201:~/projects/my_app$ git checkout zlz
Switched to branch 'z lz'
zhong@X201:~/projects/my_app$ git status
# On branch z lz
nothing to commit, working directory clean
zhong@X201:~/projects/my_app$
```

5.5.9 配置 jenkins

1 在浏览器中输入jenkins服务器地址

2 点击“New Job”创建一个新的项目

S	W	Name	Last Success	Last Failure	Last Duration
●	☁	IndexManage	23 days - #1	21 hr - #25	26 sec
●	☁	Java程序监控插件	2 days 5 hr - #38	5 hr 35 min - #43	1 min 35 sec
●	☁	ROBOT	16 hr - #8	6 days 7 hr - #1	53 sec
●	☁	信息流、消息队列、用户关系	18 hr - #11	5 days 1 hr - #5	46 sec
●	☁	数据去重代理服务	8 days 23 hr - #33	9 days 1 hr - #31	1 min 35 sec
●	☁	数据去重客户端	7 days 0 hr - #5	7 days 1 hr - #3	1 min 25 sec
●	☁	数据去重索引服务	9 days 7 hr - #6	13 days - #1	43 sec
●	☁	晶传美Java项目通用模板	8 min 57 sec - #65	6 days 23 hr - #46	34 sec
●	☁	汽车数据管理系统	13 days - #31	6 hr 48 min - #45	1 min 21 sec

Page generated: Jun 5, 2013 5:36:59 PM

192.168.7.244:8080/job/dataSource/lastBuild

3 填写项目名称

4 选择从已有的项目中拷贝

5 在文本框中填写“Jcm-Project-Template”

6 点击“OK”

Job name: my_app

Build a free-style software project

Build a maven2/3 project

Build multi-configuration project

Monitor an external job

☒ Copy existing Job

Copy from: Jcm-Project-Template

OK

Jenkins

my_app > configuration

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Coverage Report
Performance Trend

Build History (trend)
RSS for all
RSS for failures

Project name: my_app
Description: 我的一个应用
[Raw HTML] Preview

7 修改项目描述

☐ Discard Old Builds
☐ Enable project-based security
☐ This build is parameterized
☐ Disable Build (No new builds will be executed until the project is re-enabled.)
☐ Execute concurrent builds if necessary

Advanced Project Options
Advanced...


Source Code Management
☐ CVS
☐ CVS Projectset
☒ Git
Repositories
Repository URL: ssh://kfs@192.168.7.242/home/kfs/code-library/my_app
8 填写正确的Git仓库地址
Advanced...
Delete Repository
Add

Branches to build
Branch Specifier (blank for default): master
9 选择master分支
Delete Branch
Advanced...

Repository browser: FishEye
URL: http://192.168.6.191:8060/browse/my_app/
9 添加fisheye服务器地址, 如果没有则选择AUTO
Unable to connect http://192.168.6.191:8060/browse/my_app/

Publish Coverage / Complexity Scatter Plot
Code coverage plugin used: Cobertura(Statement)
Assign code coverage plugin from which Cov/Compl scatter plot gets data.
Exclude getter/setter: ☒
Exclude methods which are only 1 line, 1 complexity, and get~/set~/ method name
Verbose: ☐
Check if you want to see detailed messages for diagnosis.
Locate the graph at the topmost of Jenkins project page: ☒
check if you want to locate the graph at the topmost side of Jenkins project page.
Delete

Add post-build action
Save Apply
10 点击应用后点击保存



5.5.10 编写你的第一个 Java 类

```
$> mkdir -p src/com/example
```

```
$> nano src/com/example/Hello.java
```

```

package com.example;

/**
 * Describe class <code>Hello</code> here.
 *
 * @author <a href="mailto:zlj.3907@gmail.com">Zhong Lizhi</a>
 * @version 1.0
 */
public class Hello {

    /**
     * The name will be print.
     *
     */
    private String name = "Lion";

    /**
     * Describe <code>getName</code> method here.
     *
     * @return a <code>String</code> value
     */
    public final String getName() {
        return this.name;
    }

    /**
     * main.
     *
     * @param args String[] input args.
     */
    public static void main(final String[] args) {
        String name = new Hello().getName();
        System.out.println("Hello! " + name + " " + args);
    }
}

```

按 ctrl+o 保存文件，按 ctrl+x 退出。

\$> ant


```
[ivy:retrieve] :: resolution report :: resolve 167ms :: artifacts dl 5ms
-----
|               |          modules          || artifacts |
|      conf     | number| search|dwnlded|evicted|| number|dwnlded|
|-----|-----|-----|-----|-----|
|      default  |     2 |    0  |    0  |    0  ||     2  |    0  |
|-----|-----|-----|-----|-----|
[ivy:retrieve] :: retrieving :: com.jcm#my_app
[ivy:retrieve]  confs: [default]
[ivy:retrieve]  0 artifacts copied, 2 already retrieved (0kB/6ms)

validate:

common.compile:
[javac] Compiling 2 source files to /home/lizhi/workspace/2013/my_app/build/main
[javac] warning: [options] bootstrap class path not set in conjunction with -source 1.6
[javac] 1 warning
[javac] Creating empty /home/lizhi/workspace/2013/my_app/build/main/com/example/package-info.class

compile:

common.compile-test:
[echo] my_app: /home/lizhi/workspace/2013/my_app/build.xml

common.unit-test:

coverage:

junit-test:

_run-test:
[mkdir] Created dir: /home/lizhi/workspace/2013/my_app/dist/my_app/report/junit-xml

test:

BUILD SUCCESSFUL
```

5.5.11 编写你的第一个单元测试类

```
$> mkdir -p test/unit/com/example/
```

```
$> nano test/unit/com/example/HelloTest.java
```

```
package com.example;

import org.junit.Test;
import com.example.Hello;

public class HelloTest {
    @Test
    public void testMain() {
        Hello.main(new String[] {"This is a test!"});
    }
}
```

按 `ctrl+o` 保存文件，按 `ctrl+x` 退出。

```
$> ant
```

```

validate:

common.compile:
[javac] Compiling 2 source files to /home/lizhi/workspace/2013/my_app/build/main
[javac] warning: [options] bootstrap class path not set in conjunction with -source 1.6
[javac] 1 warning
[javac] Creating empty /home/lizhi/workspace/2013/my_app/build/main/com/example/package-info.class

compile:

common.compile-test:
[echo] my_app: /home/lizhi/workspace/2013/my_app/build.xml
[javac] Compiling 1 source file to /home/lizhi/workspace/2013/my_app/build/test
[javac] warning: [options] bootstrap class path not set in conjunction with -source 1.6
[javac] 1 warning

common.unit-test:

coverage:

junit-test:
|
|_run-test:
| [mkdir] Created dir: /home/lizhi/workspace/2013/my_app/dist/my_app/report/junit-xml
| [junit] Running com.example.HelloTest
| [junit] Testsuite: com.example.HelloTest
| [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.086 sec
| [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.086 sec
| [junit] ----- Standard Output -----
| [junit] Hello! Lion [Ljava.lang.String;@1083964f
| [junit] -----
| [junit]
| [junit] Testcase: testMain took 0.004 sec
|

test:

BUILD SUCCESSFUL
Total time: 3 seconds

```

5.5.12 执行代码覆盖率测试

\$> ant -Dcobertura.home=/home/zhong/sourceforge/cobertura-1.9.4.1

Cobertura.home 指向 cobertura 插件的安装目录。

```

[copy] Copying 1 file to /home/lizhi/workspace/2013/my_app/build/instrumented
[junit] Running com.example.HelloTest
[junit] Testsuite: com.example.HelloTest
[junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.096 sec
[junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.096 sec
[junit] ----- Standard Output -----
[junit] Hello! Lion [Ljava.lang.String;@4cdc5b6a
[junit] -----
[junit]
[junit] Testcase: testMain took 0.012 sec
[junit] Flushing results...
[junit] Flushing results done
[junit] Cobertura: Loaded information on 2 classes.
[junit] Cobertura: Saved information on 2 classes.
[junitreport] Processing /home/lizhi/workspace/2013/my_app/dist/my_app/report/junit-xml/TESTS-TestSuites.xml to /tmp/null1333492001
[junitreport] Loading stylesheet jar:file:/usr/share/ant/lib/ant-junit.jar!/org/apache/tools/ant/taskdefs/optional/junit/xsl/junit-frame
s.xsl
[junitreport] Transform time: 650ms
[junitreport] Deleting: /tmp/null1333492001

coverage-report:
[cobertura-report] Cobertura 1.9.4.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
[cobertura-report] Cobertura: Loaded information on 2 classes.
[cobertura-report] Report time: 66ms

summary-report:
[cobertura-report] Cobertura 1.9.4.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
[cobertura-report] Cobertura: Loaded information on 2 classes.
[cobertura-report] Report time: 49ms

alternate-report:
[cobertura-report] Cobertura 1.9.4.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
[cobertura-report] Cobertura: Loaded information on 2 classes.
[cobertura-report] Report time: 144ms

cobertura.coverage:

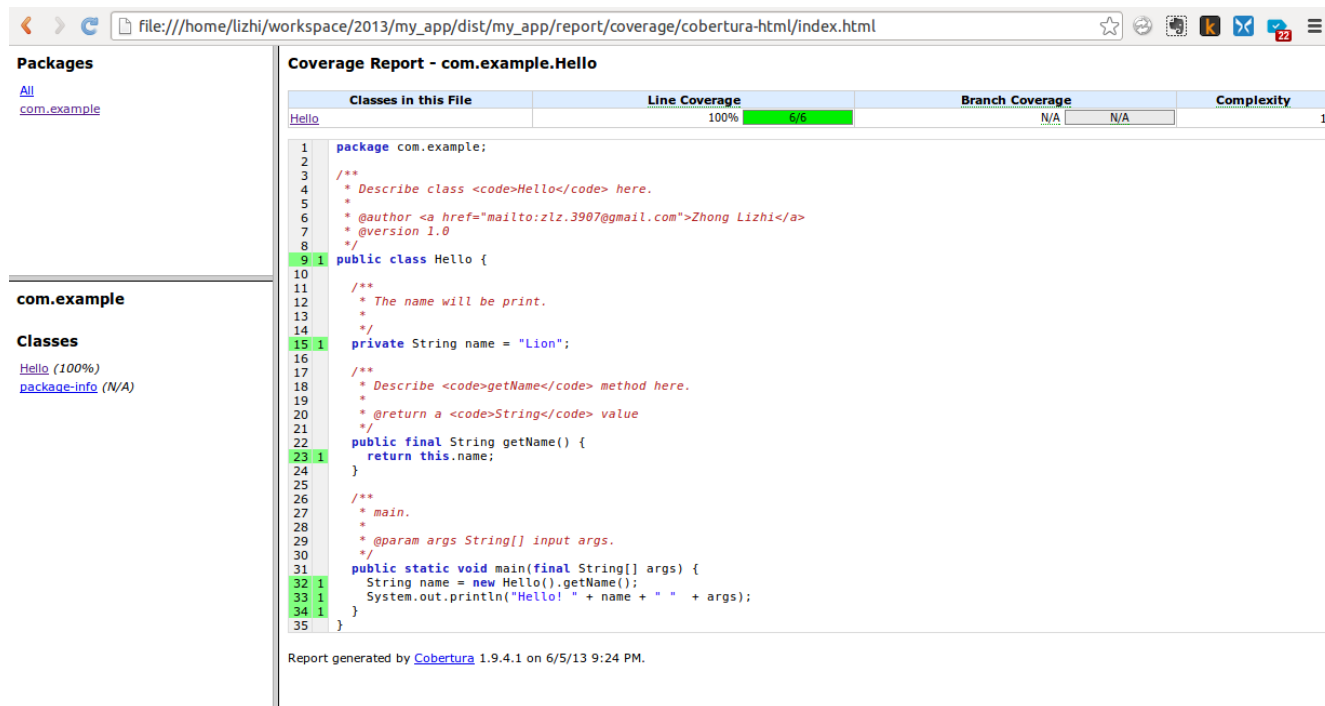
junit-test:

test:

BUILD SUCCESSFUL
Total time: 8 seconds

```

在浏览器里打开 `dist/my_app/report/coverage/cobertura-html/index.html` 页面，即可查看测试结果。



file:///home/lizhi/workspace/2013/my_app/dist/my_app/report/coverage/cobertura-html/index.html

Coverage Report - com.example.Hello

Classes in this File	Line Coverage	Branch Coverage	Complexity
Hello	100% 6/6	N/A	N/A

```

1 package com.example;
2
3 /**
4  * Describe class <code>Hello</code> here.
5  *
6  * @author <a href="mailto:ziz.3907@gmail.com">Zhong Lizhi</a>
7  * @version 1.0
8  */
9 public class Hello {
10
11     /**
12      * The name will be print.
13      */
14     private String name = "Lion";
15
16     /**
17      * Describe <code>getName</code> method here.
18      *
19      * @return a <code>String</code> value
20      */
21     public final String getName() {
22         return this.name;
23     }
24
25     /**
26      * main.
27      *
28      * @param args String[] input args.
29      */
30     public static void main(final String[] args) {
31         String name = new Hello().getName();
32         System.out.println("Hello! " + name + " " + args);
33     }
34 }
35

```

Report generated by Cobertura 1.9.4.1 on 6/5/13 9:24 PM.

5.5.13 源码规范性检查

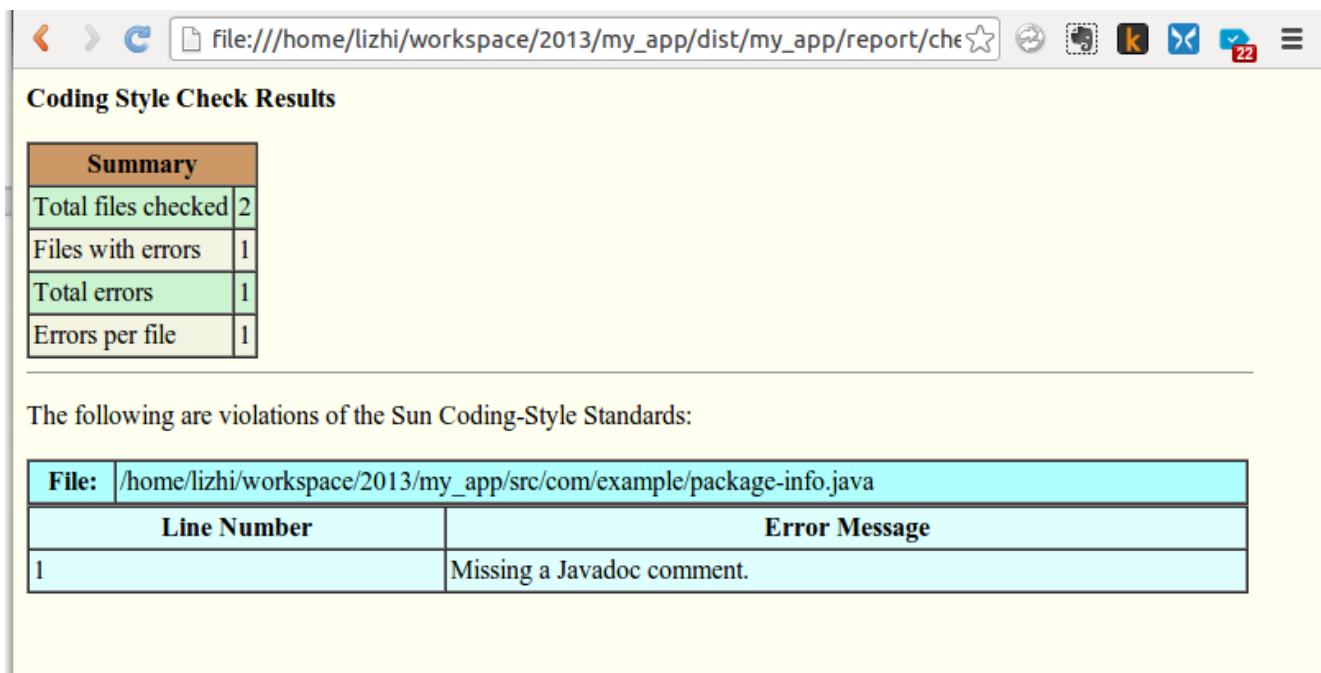
\$> ant checkstyle

```
zhong@X201:~/projects/my_app$ ant checkstyle
Buildfile: /home/lizhi/workspace/2013/my_app/build.xml

checkstyle:
[checkstyle] Running Checkstyle 5.6 on 2 files
[checkstyle] /home/lizhi/workspace/2013/my_app/src/com/example/package-info.java:1: warning: Missing a Javadoc comment.
[style] Warning: the task name <style> is deprecated. Use <xslt> instead.
[style] Processing /home/lizhi/workspace/2013/my_app/dist/my_app/report/checkstyle/checkstyle_report.xml to /home/lizhi/workspace/2013/my_app/dist/my_app/report/checkstyle/checkstyle_report.html
[style] Loading stylesheet /home/lizhi/workspace/2013/my_app/dist/my_app/report/checkstyle/checkstyle.xsl

BUILD SUCCESSFUL
Total time: 2 seconds
```

在浏览器里打开 `dist/my_app/report/checkstyle/checkstyle_report.html` 页面，即可看代码规范性检查的结果。



Coding Style Check Results

Summary	
Total files checked	2
Files with errors	1
Total errors	1
Errors per file	1

The following are violations of the Sun Coding-Style Standards:

File:	/home/lizhi/workspace/2013/my_app/src/com/example/package-info.java	
	Line Number	Error Message
	1	Missing a Javadoc comment.

5.5.14 提交代码到本地仓库

\$> git status

\$> git add .

\$> git status

\$> git commit git commit -m '添加 Hello.java 类和 HelloTest.java 测试类'

\$> git status

```

zhong@X201: ~/projects/my_app
dist/my_app/report/checkstyle/checkstyle_report.html
dist/my_app/report/junit-xml
dist/my_app/report/junit-xml/TESTS-TestSuites.xml
dist/my_app/report/junit-xml/TEST-com.example.HelloTest.xml
zhong@X201:~/projects/my_app$ find dist/^C
zhong@X201:~/projects/my_app$ git status
# On branch zlz
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       src/
#       test/
nothing added to commit but untracked files present (use "git add" to track)
zhong@X201:~/projects/my_app$ git add .
zhong@X201:~/projects/my_app$ git status
# On branch zlz
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   src/com/example/Hello.java
#       new file:   src/com/example/package-info.java
#       new file:   src/conf/example.properties
#       new file:   test/unit/com/example/HelloTest.java
#
zhong@X201:~/projects/my_app$ git commit -m '添加Hello.java类和HelloTest.java测试类'
[zhz bea4f4b] 添加Hello.java类和HelloTest.java测试类
4 files changed, 48 insertions(+)
create mode 100644 src/com/example/Hello.java
create mode 100644 src/com/example/package-info.java
create mode 100644 src/conf/example.properties
create mode 100644 test/unit/com/example/HelloTest.java
zhong@X201:~/projects/my_app$ git status
# On branch zlz
nothing to commit, working directory clean
zhong@X201:~/projects/my_app$

```

5.5.15 提交代码到共享 Git 仓库

\$> git push origin zlz

\$> git checkout master

\$> git pull origin master

\$> git merge zlz

\$> ant

\$> git push origin master

\$> git checkout zlz

\$> git merge master

6 Git 仓库使用说明

6.1 Git 快速入门

参考: <http://progit.org/book/zh/>

7 常见问题