

---

# Continuous Integration

-基于 Jenkins, Ant, Git 的实现

[www.51autotest.com](http://www.51autotest.com)

## 一. 前言

本文主要介绍在 Jenkins（就是以前的 hudson）平台上实现持续集成（CI），所使用的工具包括开源构建工具 Ant 和开源版本控制工具 Git，想了解更多这三个工具的资料可以查阅它们的官方网站：

Jenkins: <http://jenkins-ci.org/>

Ant: <http://ant.apache.org/>

Git: <http://git-scm.com/>

## 二. 基础环境搭建

首先我们需要在本地配置 tomcat 和 JDK 环境，我选择的是 tomcat-5.0.28 和 jdk1.6.0\_10，虽然官方说支持 JDK1.5 以上的版本，但是我用 JDK1.5 启动 Jenkins 会出现异常，所以这里建议使用 1.6 的作为 tomcat 的启动环境，如果项目打包需要 1.5 的环境，可以在启动以后将环境改成 1.6 即可。配置完成以后，去 Jenkins 的官方网站下载 Jenkins.war，放到 tomcat 的 webapp 目录下，启动 tomcat 后，在浏览器中输入：<http://localhost:8080/jenkins>，得到如下界面，说明启动成功：



Git 和 Ant 官方网站下载安装文件，这两个文件安装比较简单，这里不做赘述。我这里用的分别是：

- tomcat-5.0.28
- apache-ant-1.6.5
- JDK 1.6.0\_10
- Git-1.7.3.1
- checkstyle-5.4
- clover-ant-3.1.0

全部安装好并且配好环境变量以后，查看一下 ant 和 git 有没有安装成功，方法是在命令行中输入 ant -help, git --help 查看是否出现帮助文档即可。Checkstyle 和 clover 是后面会用到的 2 个工具，在 Jenkins 的 workspace 目录下找到当前项目名称，在 lib 目录中新建 clover 目录存放 clover.jar，另外 ant 的 lib 中还要保证有 checkstyle-\*.jar 和 checkstyle-\*.all.jar 这两个 jar 包。

## 三. 安装插件

Jenkins 有非常强大的插件库支持，可以满足我们的诸多需求，比如后面要用到的 checkstyle, Junit, Clover 等等，都有相应的插件提供我们安装使用。安装的方法是：系统管理->插件管理->可选插件，选择想要的插件，然后打勾，点最下面的安装按钮即可。安装好了以后建议重启 tomcat，可在已安装的目录里面查看安装的插件：

更新	可选插件	已安装	高级
启用			
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Maven 2 Project Plugin			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Static Code Analysis Plug-ins			
This plug-in provides utilities for the static code analysis plug-ins.			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Checkstyle Plugin			
This plug-in collects the <a href="#">Checkstyle</a> analysis results of the project modules and visualizes the warnings.			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Clover plugin			
This plugin integrates <a href="#">Clover code coverage reports</a> to Jenkins.			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Cobertura Plugin			
This plugin integrates <a href="#">Cobertura coverage reports</a> to Jenkins.			
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CVS Plugin			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Deploy to container Plugin			
This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Deploy to Websphere container Plugin			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Doxygen Plug-in			
This plugin publishes the reports generated by the <a href="#">Doxygen</a> tool.			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FindBugs Plugin			
This plug-in collects the <a href="#">FindBugs</a> analysis results of the project modules and visualizes the f			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins GIT plugin			
This plugin integrates <a href="#">GIT</a> with Jenkins.			
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Github Plugin			
This plugin integrates <a href="#">GitHub</a> to Jenkins.			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
JUnit Attachments Plugin			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PMD Plugin			
This plug-in collects the <a href="#">PMD</a> analysis results of the project modules and visualizes the found			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Publish Over SSH			
Send build artifacts over SSH			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Hudson Sonar Plugin			
Quickly benefit from <a href="#">Sonar</a> an open-source dashboard based on many analysis tools like Chec and Cobertura.			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SSH Slaves plugin			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins SSH plugin			
This plugin executes shell commands remotely using SSH protocol.			
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Subversion Plugin			
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jenkins Violations plugin			
This plugin does reports on checkstyle, csslint, pmd, cpd, fxcop, pylint, jcReport, and findbugs			

## 四. Jenkins 配置

打开 Jenkins 首页，点击页面左侧的“系统管理”，我们会进入到 Jenkins 的系统管理页面，首先我们配置“系统设置”。

1. 主目录：Jenkins 将所有的数据都存放在当前目录下面，如果不进行配置，默认的存放路径是：C:\Documents and Settings\Administrator\.jenkins，如果你想修改存放路径看，可以通过增加环境变量的方式来修改：



改过以后如图所示：

主目录 E:\jenkins\_workspace

系统消息

执行者数量 2

生成前等待时间 5

SCM 签出重试次数 0

☐ 启用安全

☐ 防止跨站点请求伪造

2. 系统消息：这里你可以写点随意的东西，比如 **welcome**，然后弄个 **Logo** 图片什么的，它也支持 **HTML** 语言，可以把这个欢迎界面弄的漂亮一些，这里不做赘述。
3. 执行者数量：能同时执行的任务数量，这里默认不改。
4. 生成前等待时间：一个计划中的构建在开始之前需要等待选项中设置的秒数，这里默认不改。
5. SCM 签出重试次数：如果从版本库签出代码失败，Jenkins 会按照这个指定的次数进行重试之后再放弃。
6. 启用安全和防止跨站点请求伪造暂时为默认设置。
7. 全局属性：默认不改。
8. JDK 设置：别名随便起，**JAVA\_HOME** 填写 **JDK** 本地的绝对路径。
9. Git：(安装了 **Git** 插件“**Jenkins Git plugin**”以后才能配置，如何安装详见下文)只需要正确配置路径，名字随意，见下图：

Git

Git 安装

Git Name Default

Path to Git executable D:\Program Files\Git\bin\git.exe

☐ 自动安装

新增 Git

删除 Git

系统下 Git 安装列表

10. Ant：只需要正确配置路径，名字随意，见下图：

**Ant**

Ant 安装

Ant Name

ANT\_HOME

☐ 自动安装

系统下 Ant 安装列表

11. Jenkins URL: 这里我默认是 <http://localhost:8080/jenkins>, 可以自行修改。

12. 邮件通知:

- 1) SMTP 服务器: 公司服务器的邮件地址, 比如: mail.gmail.com
- 2) 用户默认邮件后缀: @\*\*\*.com
- 3) 系统管理员邮件地址: \*\*\*@\*\*\*.com

到此, Jenkins 的系统环境变量暂时设置完成了。

## 五. 创建构建任务

点击 Jenkins 首页的 New Job 新建一个任务, 选择第一个, 点击 ok。如图:

**Jenkins**

Jenkins » All

**构建队列**

当前队列没有构建任务

**构建状态**

#	状态
1	空闲
2	空闲

Job name

☒ 构建一个自由风格的软件项目  
这是Jenkins的主要功能.Jenkins将会结合任何SCM和使用任何构建系统来构建你的项目, 甚至可以使

☐ 构建一个maven2/3项目  
构建一个maven2/3项目.Jenkins利用你的POM文件,这样可以大大减轻构建配置。

☐ 构建一个多配置项目  
适用于多配置项目,例如多环境测试,平台指定构建,等等。

☐ 监控一个外部的任务  
这个类型的任务允许你记录执行在外部Jenkins的任务, 任务甚至运行在远程机器上.这可以让Jenkins

☐ Copy existing Job  
要复制的任务名称

点击 OK 后进入项目的设置页面:

Project name

Description

☐ Discard Old Builds

☐ This build is parameterized

☐ Disable Build (No new builds will be executed until the project is re-enabled.)

☐ Execute concurrent builds if necessary (beta)

如有必要可以修改 project name 和 description, 其他默认不填。

**Source Code Management:** Jenkins 支持多种代码控制工具, 有 SVN, VSS, Git 等,

在安装了 Git 插件以后，我们这里可以选择 Git，然后进行如下的设置：

The screenshot shows the 'Source Code Management' configuration page in Jenkins. It is set to 'Git'. The 'URL of repository' is 'git@github.com:ariesliu/sgxfxfs'. There are buttons for 'Advanced...', 'Delete Repository', and 'Add'. The 'Branches to build' section has a 'Branch Specifier (blank for default):' set to 'master', with buttons for 'Delete Branch' and 'Add'. The 'Repository browser' is set to '(Auto)' with an 'Advanced...' button.

- 1) **Repositories:** 这里是我们 clone 代码的地址，因为我实验用到的项目是用 SSH 传输数据的，所以这里选择的地址的时候需要注意一点，就是 clone 代码的帐号不能有密码，否则会提示你无法获取代码，可讲生成的 key 放置在 git 安装目录下，比如我的是放在 D:\Program Files\Git\.ssh 中。
- 2) **Branches to build:** 这个是获取代码的 branch，我这里输入的是 master，大家可以根据自己的需要来填写。
- 3) **Repository browser:** 这里默认 auto。

**Build Triggers:** 该选项是用来配置进行自动构建的，比如我们想让项目中每天中午 12 点和晚上 6 点自动构建一次，只需要在 Build Triggers 中选择 Build periodically，并在 Schedule 中输入 0 12,18 \* \* \* 即可，这里要强调一下，星号之间有空格隔开，从左往右的含义是：分、时、天、月、年，这里表示任意年月日的 12 点和 18 点 0 分都运行构建任务。

The screenshot shows the 'Build Triggers' configuration page. It has three checkboxes: 'Build after other projects are built', 'Build periodically', and 'Poll SCM'. 'Poll SCM' is checked. Below is a 'Schedule' field with the text '0 12,18 \* \* \*'.

**Build:** Jenkins 支持多种构建工具，比如 ant，maven 等，这里以 ant 为例，设置如下图：

The screenshot shows the 'Build' configuration page for the 'Invoke Ant' step. It has a dropdown for 'Ant Version' set to 'ant165' and a text field for 'Targets' set to 'clean test-all'. There are buttons for 'Advanced...', 'Delete', and 'Add build step'.

图中的 ant 版本是从下拉列表选择的，列表中的内容来源于之前系统设置中配的 ant 版本，而 targets 是我们想执行的 ant 命令。

**Post-build Actions:** 这里是 Jenkins 多种分析工具集中的地方，比如代码式样检查工具 checkstyle，代码覆盖率检查工具 Clover，还有 Junit 执行和发送 report 的工具都在这里设置。

Publish Checkstyle analysis results 设置如下图所示：

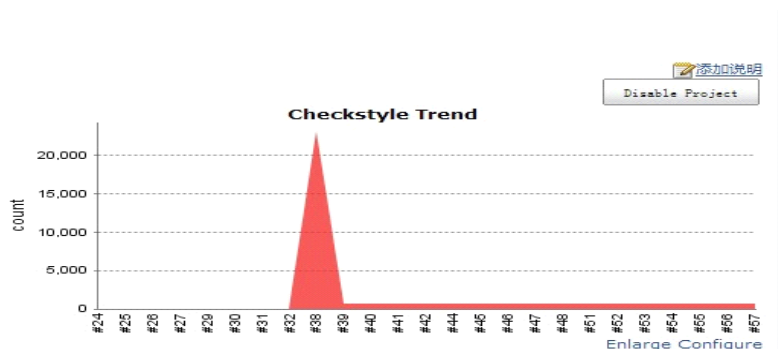
☒ Publish Checkstyle analysis results ?

Checkstyle results

Fileset includes setting that specifies the generated raw CheckStyle XML report files, such as \*\*/checkstyle-result.xml. Basedir of the fileset is the workspace root. If no value is set, then the default \*\*/checkstyle-result.xml is used. Be sure not to include any non-report files into this pattern.

[Advanced...](#)

这里的 checkstyle results 中输入的是 ant 调用 checkstyle 工具运行后产生的 xml report 的存放位置，图中表示在项目根目录下寻找 checkstyle\_report.xml 这个文件并进行分析，然后会在项目首页以线形图状态显示出来。效果如下图：



红色部分表示本次构建有多少 warning，文章后面部分会详细介绍 checkstyle 运行结果在 Jenkins 中的详细查看方法。

**Publish JUnit test result report:** 这里是 Jenkins 设置导入 Junit 运行结果 xml 文件的位置，设置如下图：

☒ Publish JUnit test result report ?

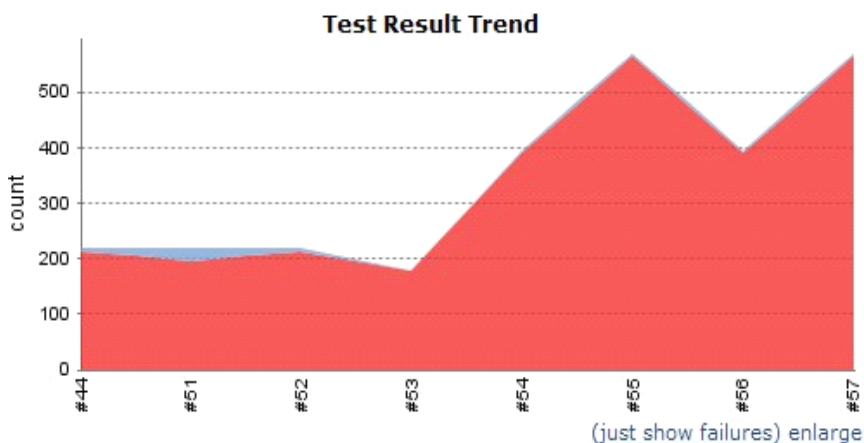
Test report XMLs

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/\*.xml'. Basedir of the fileset is the workspace root.

☐ Retain long standard output/error ?

Additional test report features ☐ Publish test attachments

设置以后进行构建时，会在项目首页显示线形图：



其中红色和蓝色分别表示每次执行构建后失败和成功的文件数量。

**Publish Clover Coverage Report:** 在这里设置 clover 工具进行代码覆盖率测试后产生的报告的位置，设置如下图：



☒ Publish Clover Coverage Report

Clover report directory

Clover report file name

Coverage Metric Targets

	% Methods	% Conditionals	% Statements
	<input type="text" value="70"/>	<input type="text" value="80"/>	<input type="text" value="80"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>

Configure health reporting thresholds.  
 For the row, leave blank to use the default values (i.e. 70, 80, and 80 for methods, conditionals and statements respectively).  
 For the and rows, leave blank to use the default values (i.e. 0).

上图表示在根目录的 reports 目录里面，将所有 xml 文件作为 report file 来解析，在首页显示位置如下图红色圈圈出来的地方：

**E-mail Notification:** 这里设置每次构建后发送结果到指定邮箱，多人邮箱用空格隔开，设置如下图：

☒ E-mail Notification

Recipients

Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

☒ Send e-mail for every unstable build

☒ Send separate e-mails to individuals who broke the build

☐ Send build artifacts over SSH

点击 save 保存设置，到此 project 的设置基本完成，文中没提到的都作为默认处理。

## 六. 运行构建任务

构建任务配置好了以后，点击页面左侧的“立即构建”，出现如图的状态：

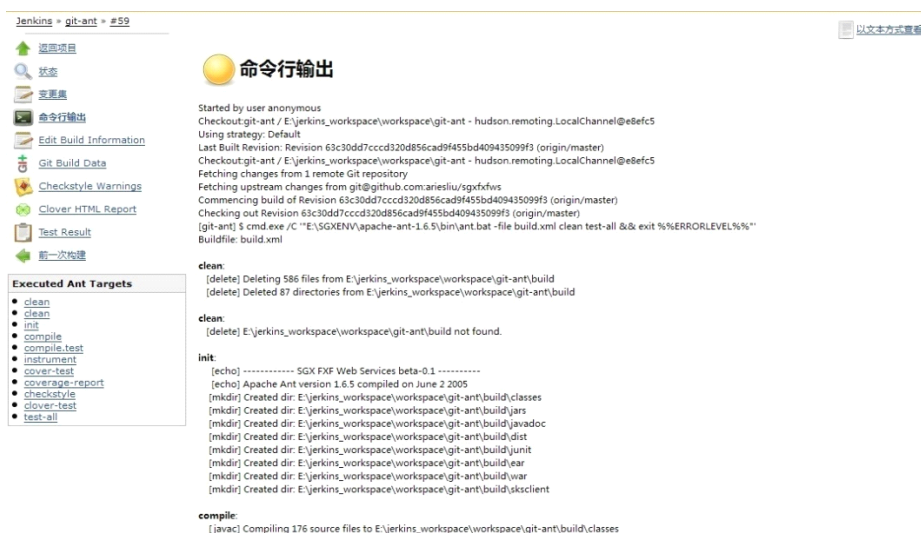




你可以点击下方的#59 旁边的时间进入此次构建任务，查看详细的运行情况：



每次构建的时候会查看 **git** 中的代码有没有提交更新，如果更新的话会在页面中间红框位置显示，如果没有提交更新则显示“no changes.”。右上角的红框显示任务进度，如果点击左侧的“命令行输出”，可以查看命令行执行的详细信息：



这里输出的信息和我们在命令行执行 **ant** 命令的时候 **cmd** 输出的信息是一样的，有助于我们观察构建时的运行情况，分析出错的原因。

运行完成以后，我们在此构建任务的页面可以查看之前在项目设置中需要查看的插件所反馈的信息，比如代码覆盖率，代码式样以及单元测试：

Jenkins » git-ant » #59

构建 #59 (2011-9-5 13:48:23)

无更改。

启动用户 anonymous

Revision: 63c30dd7cccd320d856cad9f455bd409435099f3

- origin/master

Checkstyle: 614 warnings from one Checkstyle file.

Test Result (564 failures)

- <init>.com.sgxfws.bean.TradeAndTransactionStatus
- <init>.com.sgxfws.client.pushnotification.Constant
- <init>.com.sgxfws.client.pushnotification.LogUtil
- <init>.com.sgxfws.client.pushnotification.PushHttpProxySelector
- <init>.com.sgxfws.client.pushnotification.PushNotification
- <init>.com.sgxfws.client.pushnotification.PushNotificationClient
- <init>.com.sgxfws.dao.AbstractDao
- <init>.com.sgxfws.dao.ClientConfigDao
- <init>.com.sgxfws.dao.ClientRequestDao
- <init>.com.sgxfws.dao.FXFWSDaoException

Show all failed tests >>>

我们先点击左边的 checkstyle warnings 查看一下这次检查有多少个 warning:

#### Details

Package	Files	Categories	Types	Warnings	Details	New
Package		Total	Distribution			
com.sgxfws.client.pushnotification		1				
com.sgxfws.dao		28				
com.sgxfws.dao.domain		4				
com.sgxfws.manager		2				
com.sgxfws.sks		1				
com.sgxfws.socketserver		17				
com.sgxfws.socketserver.bean		43				
com.sgxfws.socketserver.manager		17				
com.sgxfws.socketserver.spooler		17				
com.sgxfws.socketserver.ssl		13				
com.sgxfws.util		53				
com.sgxfws.webservice		11				
com.sgxfws.security.ws		30				
com.sgxfws.tdc.ws		9				
com.sgxfws.tdc.ws.fxfs		33				
com.sgxfws.tdc.ws.fxfs_push		25				
com.sgxfws.tdc.ws.sgxfxf.calypsodocument		290				
com.sgxfws.tdc.ws.sgxfxf.fxfs		20				
Total		614				

每个 package 下面有多少个 warning 都会显示出来，点击 Details 标签，我们会看到：

www.51autotest.com

## Details

Package	Files	Categories	Types	Warnings	Details	New
<b>PushNotificationClient.java:57, StaticVariableNameCheck, Priority: High</b>						
'instance' static Final变量的名字应该都大写，并且指出完整含义。						
No description available. Please upgrade to latest checkstyle version.						
<b>TradeStatusCacheDao.java:73, JavadocMethodCheck, Priority: High</b>						
<b>Missing a Javadoc comment.</b>						
Checks the Javadoc of a method or constructor. By default, does not check for unused throws. To allow documented java.lang.RuntimeExceptions that are not declared, set property allowUndeclaredRTE to true. The scope to verify is specified using the Scope class and defaults to Scope.PRIVATE. To verify another scope, set property scope to a different <a href="#">scope</a> .						
Error messages about parameters and type parameters for which no param tags are present can be suppressed by defining property allowMissingParamTags. Error messages about exceptions which are declared to be thrown, but for which no throws tag is present can be suppressed by defining property allowMissingThrowsTags. Error messages about methods which return non-void but for which no return tag is present can be suppressed by defining property allowMissingReturnTag.						
Javadoc is not required on a method that is tagged with the @Override annotation. However under Java 5 it is not possible to mark a method required for an interface (this was corrected under Java 6). Hence Checkstyle supports using the convention of using a single @inheritDoc tag instead of all the other tags.						
Note that only inheritable items will allow the @inheritDoc tag to be used in place of comments. Static methods at all visibilities, private non-static methods and constructors are not inheritable.						
For example, if the following method is implementing a method required by an interface, then the Javadoc could be done as:						
<pre>/** @inheritDoc */ public int checkReturnTag(final int aTagIndex,     JavadocTag[] aTags,     int aLineNo)</pre>						
The classpath may need to be configured to locate the class information. The classpath configuration is dependent on the mechanism used to invoke Checkstyle.						
<b>TransactionDeclareSpoolDao.java:64, JavadocMethodCheck, Priority: High</b>						
<b>Missing a Javadoc comment.</b>						

Details 中会列出每个 warning 产生的原因以及具体的文件的位置，行数。如果点击文件名，会直接定位到该文件，显示出错的地方：

```
8080/jenkins/job/git-ant/59/checkstyleResult/source.0/#57
050 * @author $Author: wcao@cn.unity.com $
051 * @version $Revision: 1 $
052 * @since $Date: 2011-06-22 10:45:41 +0800 (Wen, 22 Jun 2011) $
053 */
054 public class PushNotificationClient extends LogUtil
055 {
056     /** PushNotificationClient instance */
057     private static PushNotificationClient instance;
058
059     /** 'instance' static Final变量的名字应该都大写，并且指出完整含义。No description available. Please upgrad
060     * Constructor for PushNotificationClient
061     */
062     private PushNotificationClient()
063     {
064     }
```

点击左侧的“Clover Html Report”查看代码覆盖率的结果：

<

点击左侧的“Test Result”我们可以查看单元测试的运行结果：

## Test Result

564 failures

568 tests

Took 38 秒.

 添加说明

### All Failed Tests

Test Name	Duration	Age
>>> <init>.com.sgxfxfs.bean.TradeAndTransactionStatus	0.0	<u>1</u>
>>> <init>.com.sgxfxfs.client.pushnotification.Constant	0.0	<u>1</u>
>>> <init>.com.sgxfxfs.client.pushnotification.LogUtil	0.0	<u>1</u>
>>> <init>.com.sgxfxfs.client.pushnotification.PushHttpProxySelector	0.0	<u>1</u>
>>> <init>.com.sgxfxfs.client.pushnotification.PushNotification	0.0	<u>1</u>
>>> <init>.com.sgxfxfs.client.pushnotification.PushNotificationClient	0.0	<u>1</u>
>>> <init>.com.sgxfxfs.dao.AbstractDao	0.0	<u>1</u>

### All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
(root)	0 毫秒	352	+352	0		352	+352
com.sgxfxfs.client.test.ws.pushnotification	22 秒	2	+2	0		2	+2
com.sgxfxfs.dao	12 秒	38	+38	0		38	+38
com.sgxfxfs.manager	1.7 秒	22	+22	0		22	+22
com.sgxfxfs.webservice	1 秒	0		0		4	+4
junit.framework	1 秒	150	+150	0		150	+150

点击上面的一个失败的 test，我们可以查看到该测试失败的具体原因：

### Failed

<init>.com.sgxfxfs.bean.TradeAndTransactionStatus (from com.sgxfxfs.bean.TradeAndTransactionStatus)

Failing for the past

#### Error Message

com.sgxfxfs.bean.TradeAndTransactionStatus

#### Stacktrace

```
java.lang.ClassNotFoundException: com.sgxfxfs.bean.TradeAndTransactionStatus
    at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:268)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:251)
    at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:319)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:164)
```

## 七. 远程部署

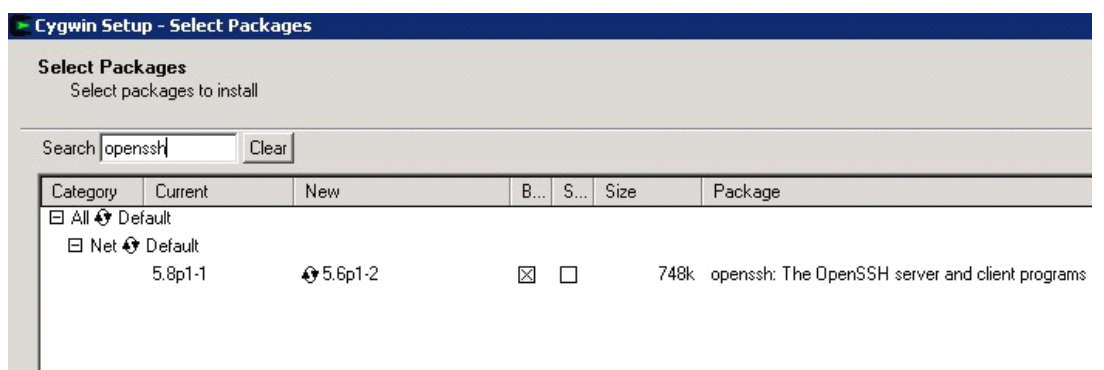
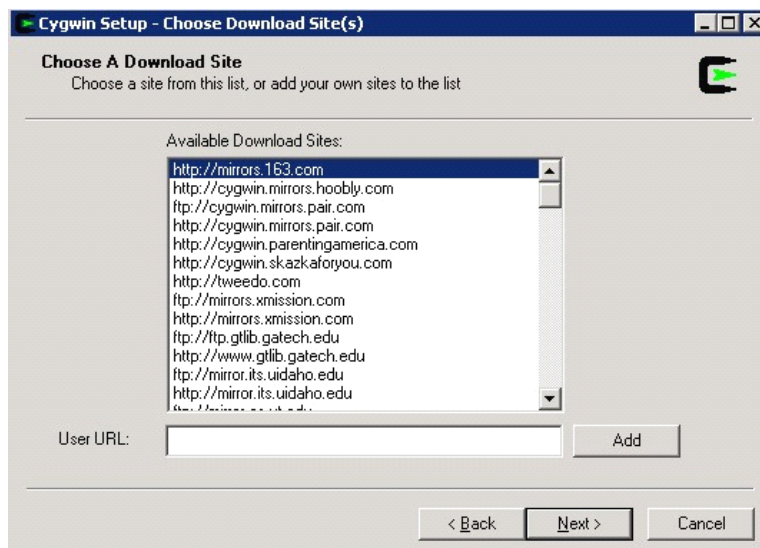
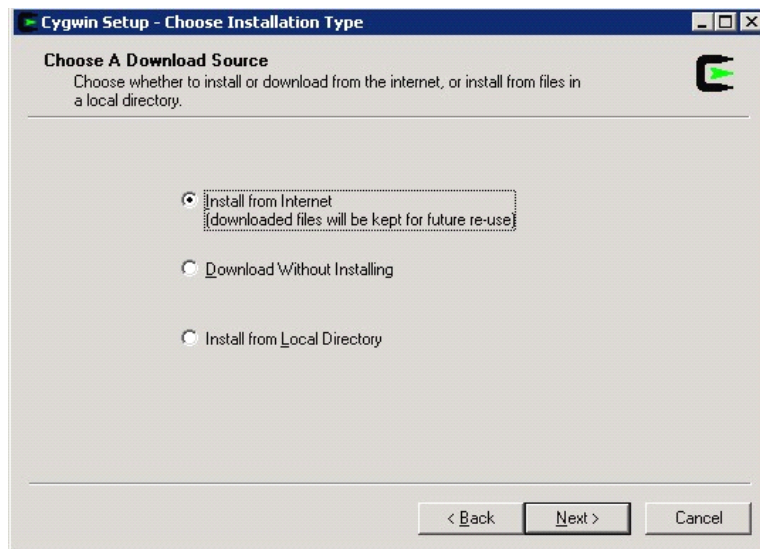
平台搭建完成以后，我们和实际项目结合起来使用来看一下。下面例子中提到的项目所使用的 web 容器是 **websphere6.1**，安装在另外一台机器上，我们要做的是利用上面说到的平台，从 **git** 中获取最新的代码打包，然后发送到远程服务器的 **websphere** 中进行部署。

由于 **Ant** 自带 **ftp**, **ssh**, **telnet** 三种远程连接方式供我们使用，因此我们可以直接通过在 **build.xml** 文件中配置 **scp** 和 **sshexec** 任务来实现远程连接，本文选择的是 **ssh** 服务来实现远程部署任务。

在操作之前，我们先要下载两个 **jar** 包，分别是 **oro-2.0.8.jar** 和 **commons-net.jar**，将两个 **jar** 包放到 **ant** 安装路径下的 **lib** 中。然后检查 **websphere** 所在的服务器有没有开通 **ssh** 服务，查看方式很简单，看看进程中有没有 **sshd.exe** 即可，如果没有，建议安装一下 **openssh**，官方推荐几款工具，具体方法如下：

1. 到 <http://www.openssh.com/windows.html> 下载工具，本文选择 **cygwin**。

2. 安装 cygwin 的时候基本上都是点 next 安装，有以下几处需要注意一下：



安装好了以后，打开 cygwin，在里面输入：**net start sshd**，看到提示“启动成功”即可：

```
Administrator@junliu ~
$ net start sshd
CYGWIN sshd 服务正在启动。
CYGWIN sshd 服务已经启动成功。

Administrator@junliu ~
$
```



3. 可以利用远程访问工具，如 **putty** 来访问一下，看看是否成功。
4. 远程机器的 **ssh** 服务启动好以后，我们在本地用 **ant** 尝试连接一下，**build.xml** 中的代码如下：

```
<target name="test" >
    <sshexec host="192.168.0.19"
        port="22"
        username="Administrator"
        password="123456"
        trust="true"
        command="d:/tagCollection/copy.bat"/>
</target>
```

为了更容易看懂，这里的参数都没有用变量替换，路径都是绝对路径，在写的时候可以替换一下。这里的默认访问端口是 **22**，如果没有特殊情况可以不修改，**Command** 里面是要执行的命令。

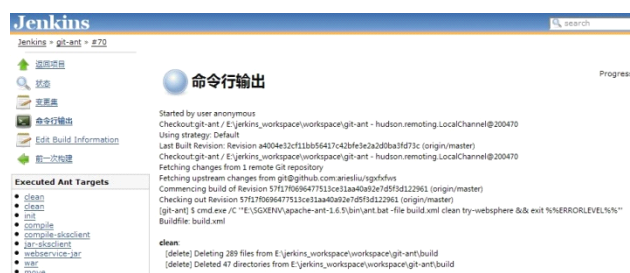
5. 利用 **Ant** 的 **scp** 任务将本地的包发送到远程服务器中的指定位置，代码如下：

```
<property name="address" value="192.168.0.19:d:/tagCollection" />
<property name="user" value="Administrator" />
<property name="password" value="123456" />

<target name="move" >
    <echo message="uploading....." />
    <scp file="E:\test.war" todir="${user}:${password}@${address}"
        trust="true" verbose="true"/>
    <echo message="done....." />
</target>
```

例子中是将生成的 **war** 包放到远程服务器 **D** 盘的 **tagCollection** 目录下，位置任意指定。

6. 待 **war** 包放好以后，可以利用 **Ant** 的 **command** 参数来执行服务器上的 **bat** 文件来做你想做的事情了。
7. 最后额外说一下，由于我所用的项目是在 **Windows server 2008** 上进行部署的，另外 **websphere** 不支持热拷贝，因此需要用到解压命令，而 **Windows2008** 不支持 **unzip** 命令，我使用的是 **7z** 工具来解压 **war** 包，然后将解压好的文件放到 **websphere** 服务器的 **war** 包目录下。如果你们也要用到 **7z**，需要将 **7z.exe** 放到 **cygwin** 的 **bin** 目录下。
8. 当 **ant** 远程部署配置完成以后，我们到 **Jenkins** 中运行构建任务，调用 **ant** 命令完成一系列的任务，执行结果如下所示：



---

```
init:
[echo] ----- SGX FXF Web Services beta-0.1 -----
[echo] Apache Ant version 1.6.5 compiled on June 2 2005
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\classes
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\jars
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\javadoc
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\dist
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\junit
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\year
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\war
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\skscient

compile:
[javac] Compiling 176 source files to E:\jerkins_workspace\workspace\git-ant\build\classes

compile-skscient:
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\skscient\classes
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\skscient\src
[copy] Copying 12 files to E:\jerkins_workspace\workspace\git-ant\build\skscient\src
[javac] Compiling 12 source files to E:\jerkins_workspace\workspace\git-ant\build\skscient\classes

jar-skscient:
[jar] Building jar: E:\jerkins_workspace\workspace\git-ant\build\skscient\sgxsksservice.jar

webservice-jar:
[jar] Building jar: E:\jerkins_workspace\workspace\git-ant\build\jars\sgx-fxfs.jar

war:
[mkdir] Created dir: E:\jerkins_workspace\workspace\git-ant\build\webinf
[copy] Copying 16 files to E:\jerkins_workspace\workspace\git-ant\build\webinf
[war] Building war: E:\jerkins_workspace\workspace\git-ant\build\war\sgxfxfs.war

move:
[echo] uploading.....
[scp] Connecting to 192.168.0.19:22
[scp] Sending: sgxfxfs.war : 35184113
[scp] ..... 50%
[scp] ..... 100%
[scp] File transfer time: 4.47 Average Rate: 7,872,927.5 B/s
[scp] done.
[echo] done.....
```

```
copy-ear:
[sshexec] Connecting to 192.168.0.19:22

[sshexec] D:\cygwin\Install\home\Administrator>echo "start to extract the war package...."
[sshexec] "start to extract the war package...."

[sshexec] D:\cygwin\Install\home\Administrator>cd D:\tagCollection

[sshexec] D:\tagCollection>rd /s /q D:\tagCollection\META-INF

[sshexec] D:\tagCollection>rd /s /q D:\tagCollection\WEB-INF

[sshexec] D:\tagCollection>7z x sgxfxfs.war

[sshexec] 7-Zip 9.20 Copyright (c) 1999-2010 Igor Pavlov 2010-11-18

[sshexec] Processing archive: sgxfxfs.war

[sshexec] Extracting META-INF
[sshexec] Extracting META-INF\MANIFEST.MF
[sshexec] Extracting WEB-INF
[sshexec] Extracting WEB-INF\application.xml
[sshexec] Extracting WEB-INF\hibernate.reveng.xml
[sshexec] Extracting WEB-INF\sun-jaxws.xml
[sshexec] Extracting WEB-INF\wsdl
[sshexec] Extracting WEB-INF\wsdl\FXFNotificationService.wsdl
[sshexec] Extracting WEB-INF\wsdl\FXFTradeService.wsdl
[sshexec] Extracting WEB-INF\wsdl\fpml-asset-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-bond-option-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-business-events-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-cd-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-com-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-confirmation-processes-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-correlation-swaps-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-dividend-swaps-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-doc-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-enum-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-eq-shared-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-eqd-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-fx-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-ird-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-main-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-msg-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-option-shared-5-1.xsd
[sshexec] Extracting WEB-INF\wsdl\fpml-return-swaps-5-1.xsd
```



```
[sshexec] D:\tagCollection\WEB-INF\wsdl\fpml-shared-5-1.xsd
[sshexec] D:\tagCollection\WEB-INF\wsdl\fpml-variance-swaps-5-1.xsd
[sshexec] D:\tagCollection\WEB-INF\wsdl\FXFNotificationService.wsdl
[sshexec] D:\tagCollection\WEB-INF\wsdl\FXFTradeService.wsdl
[sshexec] D:\tagCollection\WEB-INF\wsdl\fxfxws-schema.xsd
[sshexec] D:\tagCollection\WEB-INF\wsdl\push-schema.xsd
[sshexec] D:\tagCollection\WEB-INF\wsdl\trsws-schema.xsd
[sshexec] D:\tagCollection\WEB-INF\wsdl\xmldsig-core-schema.xsd
[sshexec] 90 File(s) copied

[sshexec] D:\Program Files (x86)\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\WIN-FNHJWJ9651KNode01Cell\sgxfxfs.ear
\sgxfxfs.war>xcopy D:\tagCollection\META-INF\META-INF /e /y
[sshexec] D:\tagCollection\META-INF\MANIFEST.MF
[sshexec] 1 File(s) copied
```

try-websphere:

```
BUILD SUCCESSFUL
Total time: 55 seconds
[CHECKSTYLE] Collecting checkstyle analysis files...
[CHECKSTYLE] Successfully parsed file E:\jerkins_workspace\workspace\git-ant\checkstyle_report.xml of module with 615 warnings.
Finished: SUCCESS
```

## 八. 其他事项

在使用 Ant 进行构建的时候，配置 build.xml 文件也是一个很重要的环节。这里对如何编写 build.xml 不做讨论，这里贴一下我 build.xml 文件的部分代码给大家参考：

```
<target name="clover-test" depends="compile.test" description="Runs the tests">
    <mkdir dir="${test.reports}/clover"/>
    <junit fork="yes" printsummary="true" showoutput="true">
        <classpath refid="clover.classpath"/>
        <classpath refid="project.class.path" />
        <formatter type="xml"/>
        <batchtest fork="yes" todir="${test.reports}" >
            <fileset dir="src" includes="**/*.java"/>
        </batchtest>
    </junit>
</target>

<target name="checkstyle" depends="compile.test" >
    <mkdir dir="${test.reports}/checkstyle"/>
    <taskdef resource="checkstyletask.properties" />
    <checkstyle config="docs\sun_checks.xml" failureProperty="checkstyle.failure"
failOnViolation="false">
        <formatter type="xml" tofile="checkstyle_report.xml"/>
        <fileset dir="src" includes="**/*.java"/>
    </checkstyle>
    <style in="checkstyle_report.xml"
        out="${test.reports}/checkstyle/checkstyle_report.html"
        style="checkstyle-frames.xsl"/>
</target>
```