

---

# 基于 Hudson 的持续集成指南



创想空间商务通信服务有限公司  
Super Sales Team  
2009-05

## 修订表

| 日期         | 作者  | 版本   | 注释      |
|------------|-----|------|---------|
| 2009-04-27 | 朱建永 | V0.1 | 初始      |
| 2009-05-30 | 朱建永 | V0.9 | 完成第一版初稿 |
|            |     |      |         |
|            |     |      |         |
|            |     |      |         |
|            |     |      |         |
|            |     |      |         |



---

## 文档目录

|                          |    |
|--------------------------|----|
| 1 前言.....                | 6  |
| 1.1 引言.....              | 6  |
| 1.2 关于本指南.....           | 6  |
| 1.3 目标.....              | 6  |
| 1.4 前提条件.....            | 6  |
| 1.5 系统要求.....            | 7  |
| 1.6 术语和缩略语.....          | 7  |
| 2 基于 Hudson 的 CI 环境..... | 7  |
| 2.1 搭建 CI 环境.....        | 7  |
| 2.2 配置 Hudson.....       | 11 |
| 2.3 新建一个 Job.....        | 13 |
| 2.4 启动 Build.....        | 20 |
| 2.5 如何利用 CI.....         | 24 |
| 2.6 进阶.....              | 29 |
| 3 附录.....                | 35 |
| 3.1 相关资源.....            | 35 |
| 3.2 联系作者.....            | 35 |

## 图表索引

|  |    |
|--|----|
| 图表 1 Tomcat 启动 console.....            | 8  |
| 图表 2 Tomcat 的初始页面.....                 | 9  |
| 图表 3 Tomcat console – 自动部署 hudson..... | 9  |
| 图表 4 Hudson 系统首页.....                  | 10 |
| 图表 5 Hudson – 安装 Plug-Ins.....         | 10 |
| 图表 6 验证 Ant 是否就绪.....                  | 11 |
| 图表 7 Hudson 系统首页.....                  | 11 |
| 图表 8 Manage Hudson.....                | 12 |
| 图表 9 Hudson – 新建一个 Job 首页.....         | 14 |
| 图表 10 Hudson – Job 配置页面.....           | 15 |
| 图表 11 Hudson 系统首页.....                 | 16 |
| 图表 12 Hudson – Job 首页.....             | 16 |
| 图表 13 Job 首页- 完成配置.....                | 21 |
| 图表 14 Job 首页 – 手动启动 build.....         | 21 |
| 图表 15 Job Build 信息页.....               | 22 |
| 图表 16 Build 信息页 – Console Output.....  | 22 |
| 图表 17 Job 页面 – Workspace 信息.....       | 23 |
| 图表 18 Job 首页 – 完成一次 build.....         | 24 |
| 图表 19 Hudson Super Sales Job.....      | 25 |
| 图表 20 静态代码检查报告.....                    | 26 |
| 图表 21 静态代码检查 – 异常信息详情.....             | 26 |
| 图表 22 单元测试报告.....                      | 27 |
| 图表 23 单元测试报告 – 详情.....                 | 27 |
| 图表 24 单元测试覆盖率 1.....                   | 28 |
| 图表 25 单元测试覆盖率 2.....                   | 29 |
| 图表 26 用户信息.....                        | 30 |
| 图表 27 Hudson – 安全设置 1.....             | 31 |
| 图表 28 Hudson 首页 – 安全设置.....            | 32 |
| 图表 29 Hudson 注册管理员信息.....              | 32 |

---

|                                 |    |
|---------------------------------|----|
| 图表 30 Hudson – 安全设置 2.....      | 33 |
| 图表 31 Hudson 安全设置 – 系统首页.....   | 33 |
| 图表 32 Hudson 安全设置 – Job 首页..... | 34 |



# 1 前言

## 1.1 引言

在最近的一个项目中，为了保证产品质量，项目进行了持续集成（Continuous Integration，以下简称 CI）活动的实践，这之中遇到并解决过一些问题；目前，CI 真切的在我们项目中发挥了作用。

写这个指南，旨在和大家交流经验、分享成果。这里介绍了在这次 CI 项目实践中的所有细节，特别是对 Hudson 的使用，但并未涵盖 Hudson 的所有范畴。

## 1.2 关于本指南

本指南介绍了基于 Java 语言的 CI，重点关注如何建立起一个稳定、可重复的 CI 过程。同时简单说明了什么是 CI，以及 CI 对软件系统的意义。

通过本指南，你能学到如何正确的配置 CI 服务来定时轮询代码配置管理库，当发现源码更新时，自动对项目进行构建操作。你还能学到如何利用 CI 服务及第三方插件作静态代码检查、自动运行单元测试并生成用户界面友好的报告。最后，如果在这过程中有任何异常，CI 服务会及时通过邮件通知你。

基于上面的行为，你可以每天了解所开发软件系统的状态，并更快的构建出稳定的软件系统。

## 1.3 目标

这个指南讨论了 CI 的具体实践；通过与 Ant、Subversion, 和一些 plug-ins 的集成，你将可以建立起自己的基于的 Hudson 的持续集成环境。

## 1.4 前提条件

首先，你需要了解一般的 Java 应用开发，特别是应该比较熟悉 ant 角本的编写和 JUnit 的使用。

本指南假设你理解，在为确保一定质量的前提下而进行持续的软件构建的意义，认可为之付出一些努力是值得的，这也是编写本指南的目的。

## 1.5 系统要求

持续集成环境需要自动 Build 工具、代码管理工具和持续集成服务器，在这里我们分别通过 Ant、SubVersion 和 Hudson 实现这些功能。同时，我们需要一个运行有 Java 运行时环境的计算机系统；本指南涉及的所有操作及截图都是在 Windows XP 进行的。

部署运行 Hudson 所需的系统资源不应少于：

内存：512MB

硬盘：空间不小于 30MB（用于部署 Hudson 需要的空间，不包括运行时生成的过程文件占用的空间）

## 1.6 术语和缩略语

下面的表格描述了本指南中使用到的工具、术语和缩略语：

| 术语和缩略语      | 说明                                   |
|-------------|--------------------------------------|
| CI          | Continuous Integration, 持续集成         |
| Hudson      | 用于 CI 的开源项目                          |
| SVN         | SubVersion, 配置管理工具                   |
| Ant         | 用于 Java 系统编译的开源项目，类似于 C 的 make       |
| JUnit       | 用于 Java 单元测试的开发框架                    |
| JRE         | Java Runtime Environment, Java 运行时环境 |
| Check-Style | 开源项目，用于 Java 代码的静态检查                 |
| Cobertura   | 开源项目，用于统计单元测试对代码的覆盖率                 |
| SMTP        | 简单邮件发送协议                             |

# 2 基于 Hudson 的 CI 环境

简单的说，CI 就是关于在代码发现变化时，自动 build 软件系统的一个过程；它带来的好处很简单，你可以尽早的发现问题、处理问题。对于软件过程来说，问题暴露得越晚，处理起来的代价也越大。

下面提及的所有软件、相应的版本和获取资源的位置请参见[\[3.1 相关资源\]](#)。

## 2.1 搭建 CI 环境

这一节，我们重点关注的是 Tomcat、Hudson、Ant 和 SubVersion 集成环境的搭建。

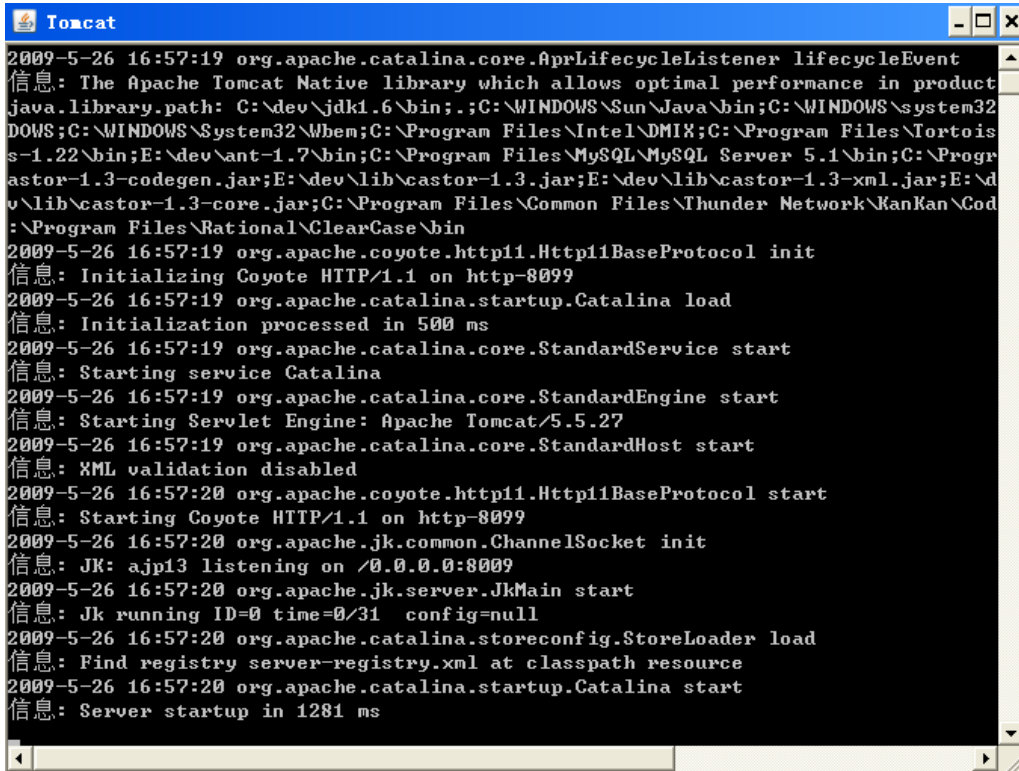
### [安装 Tomcat]

请先在你的计算机系统安装 Tomcat。下载 Tomcat 安装包，按照默认选项安装。完成安

装后，在 Tomcat 的安装路径下找到：

```
{tomcat_home}/bin/startup.bat
```

运行该批处理文件，如果弹出如下的 Console 窗口，则说明 Tomcat 已经可以正常启动了：



```
Tomcat
2009-5-26 16:57:19 org.apache.catalina.core.AprLifecycleListener lifecycleEvent
信息: The Apache Tomcat Native library which allows optimal performance in product
java.library.path: C:\dev\jdk1.6\bin;.;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32
DOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\Intel\DMIX;C:\Program Files\Tortois
s-1.22\bin;E:\dev\ant-1.7\bin;C:\Program Files\MySQL\MySQL Server 5.1\bin;C:\Progr
astor-1.3-codegen.jar;E:\dev\lib\castor-1.3.jar;E:\dev\lib\castor-1.3-xml.jar;E:\d
v\lib\castor-1.3-core.jar;C:\Program Files\Common Files\Thunder Network\KanKan\Cod
:\Program Files\Rational\ClearCase\bin
2009-5-26 16:57:19 org.apache.coyote.http11.Http11BaseProtocol init
信息: Initializing Coyote HTTP/1.1 on http-8099
2009-5-26 16:57:19 org.apache.catalina.startup.Catalina load
信息: Initialization processed in 500 ms
2009-5-26 16:57:19 org.apache.catalina.core.StandardService start
信息: Starting service Catalina
2009-5-26 16:57:19 org.apache.catalina.core.StandardEngine start
信息: Starting Servlet Engine: Apache Tomcat/5.5.27
2009-5-26 16:57:19 org.apache.catalina.core.StandardHost start
信息: XML validation disabled
2009-5-26 16:57:20 org.apache.coyote.http11.Http11BaseProtocol start
信息: Starting Coyote HTTP/1.1 on http-8099
2009-5-26 16:57:20 org.apache.jk.common.ChannelSocket init
信息: JK: ajp13 listening on /0.0.0.0:8009
2009-5-26 16:57:20 org.apache.jk.server.JkMain start
信息: Jk running ID=0 time=0/31 config=null
2009-5-26 16:57:20 org.apache.catalina.storeconfig.StoreLoader load
信息: Find registry server-registry.xml at classpath resource
2009-5-26 16:57:20 org.apache.catalina.startup.Catalina start
信息: Server startup in 1281 ms
```

图表 1 Tomcat 启动 console

**Tips:**

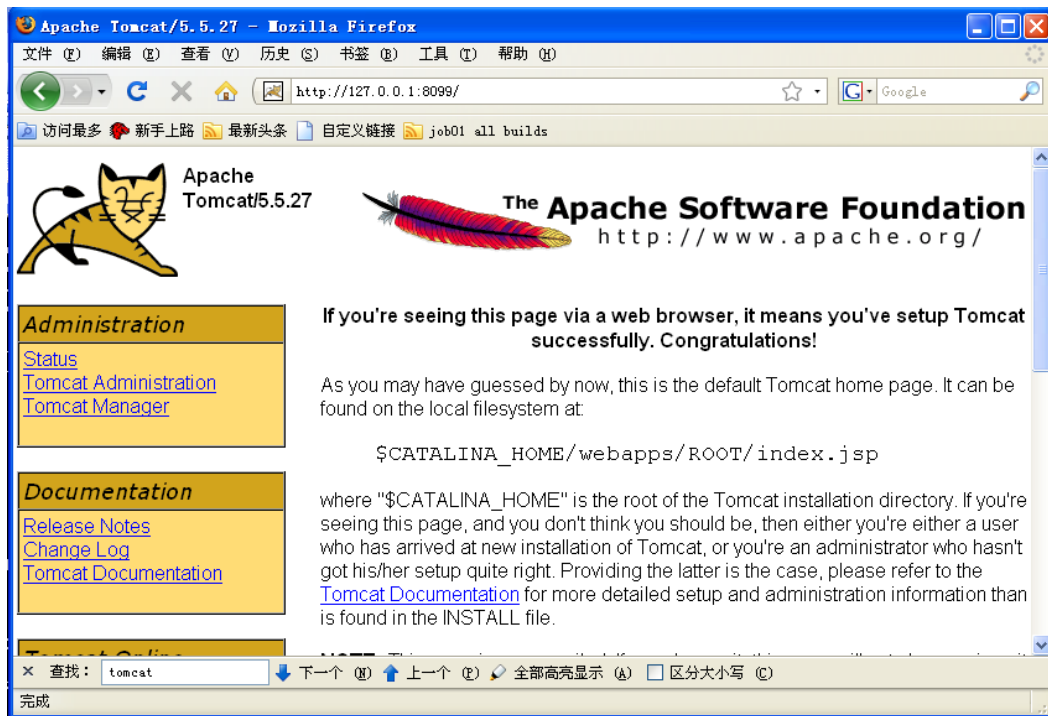
- 1、如果 Tomcat 不能正常启动，请检查你的计算机系统的环境变量中是不是已经正确设置了[java\_home]。
- 2、如果启动 Tomcat 发生了端口冲突提示，请关闭占用端口的应用；或是修改 Tomcat 的配置文件 [{tomcat\_home}/conf/server.xml]，按注释的说明，修改相应端口。

在你的浏览器地址栏上键入安装 Tomcat 计算机的 IP 地址，和 tomcat 提供 http 服务的端口，这里是安装在了本地环境上，端口使用的是 8099：

```
http://127.0.0.1:8099/
```

按下回车，如果能看到如下页面，则说明你的 Tomcat 已经安装成功了。





图表 2 Tomcat 的初始页面

### [安装 Hudson 及 Plug-Ins]

下载 Hudson 和我们用到的两个 Plug-ins；Hudson 是以 J2EE Web 应用包的形式提供下载的，是一个以 war 作为后缀的文件；Hudson 的 Plug-ins 以 hpi 为后缀的文件：

```
hudson.war
checkstyle.hpi
cobertura.hpi
```

把 hudson.war 拷贝到 tomcat\_home 的 webapps 目录下：

```
{tomcat_home}/webapps/
```

Tomcat 会自动部署 hudson，如下图所示：

```
2009-5-26 17:43:40 org.apache.catalina.startup.HostConfig deployWAR
信息: Deploying web application archive hudson.war
hudson home directory: C:\Documents and Settings\jianyong.zhu\.hudson
2009-5-26 17:43:42 hudson.ClassicPluginStrategy createPluginWrapper
信息: Loading plugin: C:\Documents and Settings\jianyong.zhu\.hudson\plugins\maven-plugin.hpi
2009-5-26 17:43:42 hudson.ClassicPluginStrategy explode
信息: Extracting C:\Documents and Settings\jianyong.zhu\.hudson\plugins\maven-plugin.hpi
```

图表 3 Tomcat console - 自动部署 hudson

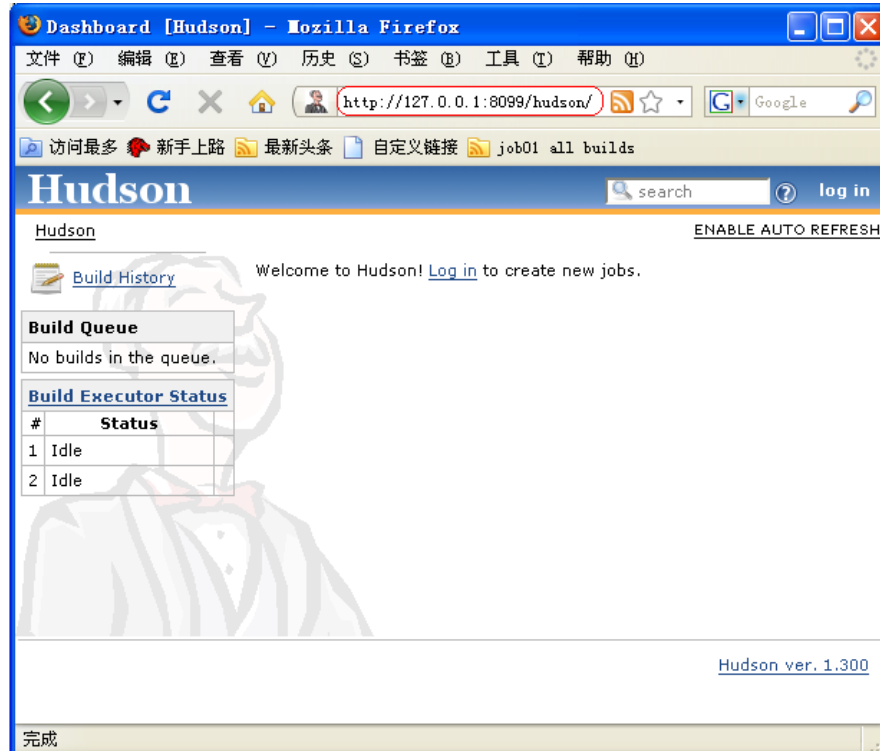
#### Tips:

- 1、Hudson 会被自动部署
- 2、Hudson 会为自己设置一个工作目录，这个目录默认值为系统用户目录下的.hudson 文件夹
- 3、插件 maven-plugin 会被作为一个默认的插件被安装（本指南并不会用到这个插件）

这个时候，在浏览器的地址栏键入：

<http://127.0.0.1:8099/hudson/>

按下回车，如果能看到如下页面，那么你的 Hudson 安装成功了：



图表 4 Hudson 系统首页

下一步是安装我们需要的两个 Plug-Ins。在 hudson 的工作目录下有一个[plugins]文件夹，把下载的两个 Plug-Ins (hpi 文件) 拷贝到这里，重新启动 Tomcat，如下图：

```

hudson home directory: C:\Documents and Settings\jianyong.zhu\hudson
2009-5-26 19:48:34 hudson.ClassicPluginStrategy createPluginWrapper
信息: Loading plugin: C:\Documents and Settings\jianyong.zhu\hudson\plugins\checkstyle.hpi
2009-5-26 19:48:34 hudson.ClassicPluginStrategy explode
信息: Extracting C:\Documents and Settings\jianyong.zhu\hudson\plugins\checkstyle.hpi
2009-5-26 19:48:34 org.apache.coyote.http11.Http11BaseProtocol start
信息: Starting Coyote HTTP/1.1 on http-8099
2009-5-26 19:48:34 org.apache.jk.common.ChannelSocket init
信息: JK: ajp13 listening on /0.0.0.0:8009
2009-5-26 19:48:34 org.apache.jk.server.JkMain start
信息: Jk running ID=0 time=0/32 config=null
2009-5-26 19:48:34 org.apache.catalina.storeconfig.StoreLoader load
信息: Find registry server-registry.xml at classpath resource
2009-5-26 19:48:34 org.apache.catalina.startup.Catalina start
信息: Server startup in 2485 ms
2009-5-26 19:48:35 hudson.ClassicPluginStrategy createPluginWrapper
信息: Loading plugin: C:\Documents and Settings\jianyong.zhu\hudson\plugins\cobertura.hpi
2009-5-26 19:48:35 hudson.ClassicPluginStrategy explode
信息: Extracting C:\Documents and Settings\jianyong.zhu\hudson\plugins\cobertura.hpi
2009-5-26 19:48:35 hudson.ClassicPluginStrategy createPluginWrapper
    
```

图表 5 Hudson - 安装 Plug-Ins

你应该能注意到，这两个插件被识别到并解包安装了。

**[Ant 和 SubVersion]**

请确认你的环境中已经能正常使用 Ant 和 SubVersion，具体的安装方法，这里不多做赘述。在 command line 下，键入 ant -version 并回车，如果出现下面的提示，说明在你的环境中 ant 已经就绪了：

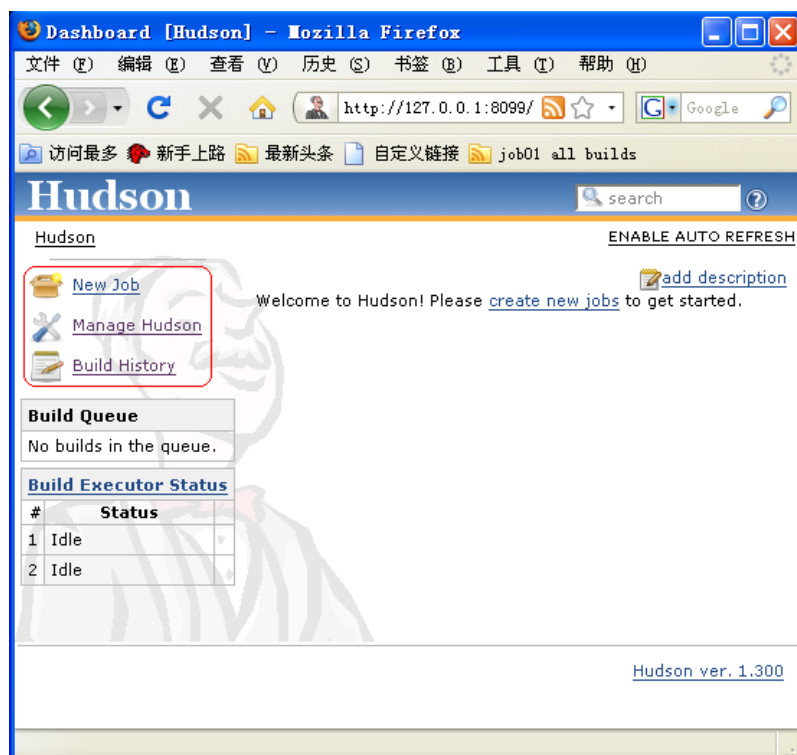
```
C:\Documents and Settings\jianyong.zhu>ant -version
Apache Ant version 1.7.1 compiled on June 27 2008
```

图表 6 验证 Ant 是否就绪

## 2.2 配置 Hudson

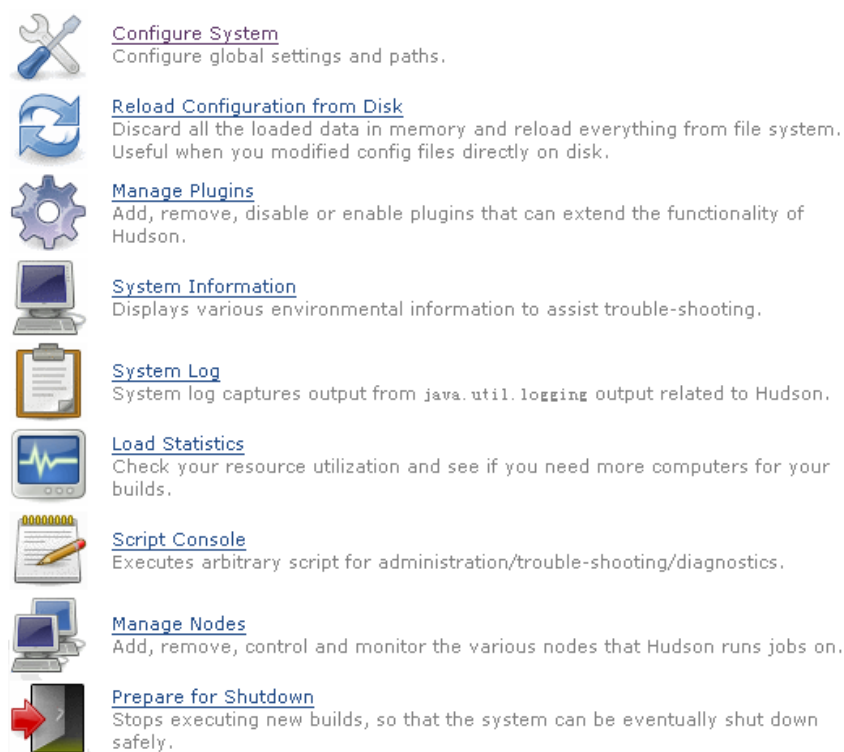
这一节会带大家了解 Hudson 基本环境的配置。

下图为 hudson 系统的首页，左侧的菜单栏有三个可选的项目 [New Job]、[Manage Hudson]、[Build History]：



图表 7 Hudson 系统首页

点击 [Manage Hudson]，进入 [Manage Hudson] 页面：



图表 8 Manage Hudson

在本节，我们主要关心的是[Config System]一项，点击该链接，进入系统设置页面。在这一页，有三个项目需要被设置，[JDKs]、[Ant]和[E-mail Notification]，请根据你个人系统环境的实际情况添写：

#### JDKs

|                   |           |  |
|-------------------|-----------|--|
| JDK installations | name      | <input type="text" value="java_home"/>     |
|                   | JAVA_HOME | <input type="text" value="C:\dev\jdk1.6"/> |

#### Ant

|                  |          |   |
|------------------|----------|---|
| Ant installation | name     | <input type="text" value="ant_home"/>       |
|                  | ANT_HOME | <input type="text" value="E:\dev\ant-1.7"/> |

需要重点说明的是[Default user e-mail suffix]一项。Hudson 在每次从 SVN 上取得源文件时，都会拿到对源文件有更新操作的用户名，如：[jianyong.zhu]，通过这种方式，Hudson 会维护一个用户列表，一旦在 CI 活动中有任何异常发生，Hudson 会自动匹配用户名加上电子邮件后缀（e-mail suffix），在当前场景下，这个邮件地址将会是：

**[jianyong.zhu@gnetis.com]**

**E-mail Notification**

[Test configuration by sending e-mail to System Admin Address](#)

|                             |   |
|-----------------------------|---|
| SMTP server                 | <input type="text" value="smtp.gnetis.com"/>                    |
| Default user e-mail suffix  | <input type="text" value="@gnetis.com"/>                        |
| System Admin E-mail Address | <input type="text" value="jianyong.zhu@gnetis.com"/>            |
| Hudson URL                  | <input type="text" value="http://192.168.12.126:8099/hudson/"/> |

点击[Manage Hudson]页面的[Manage Plugins]，可以查看到 Hudson 已经安装了的 Plug-Ins，如下图所示，当前已经安装了三个 Plug-Ins：

| Updates                             | Available  | Installed | Advanced |
|-------------------------------------|--|-----------|----------|
| Enabled                             | Name ↓   | Version   |          |
| <input checked="" type="checkbox"/> | <a href="#">Checkstyle Plugin</a><br>This plug-in collects the <a href="#">Checkstyle</a> analysis results of the project modules and visualizes the found warnings. | 2.7       |          |
| <input checked="" type="checkbox"/> | <a href="#">Cobertura Plugin</a><br>This plugin integrates <a href="#">Cobertura coverage reports</a> to Hudson.   | 0.8.5     |          |
| <input checked="" type="checkbox"/> | <a href="#">Maven Integration plugin</a>   | 1.300     |          |

其它条目暂时可以不用关心，现在，我们已经完成 Hudson 环境的基本配置了。

## 2.3 新建一个 Job

这一节，我们会新建一个 Hudson Job，并对其进行详细的配置。

### **[生成一个 Job]**

在 Hudson 系统首页，点击[New Job] 链接，可先看下图所示的页面：

Hudson

search ?

Hudson

New Job

Manage Hudson

Build History

Build Queue

No builds in the queue.

Build Executor Status

| # | Status |
|---|--------|
| 1 | Idle   |
| 2 | Idle   |

Job name

ss\_job

Monitor an external job

This type of job allows you to record the execution of a process run outside Hudson, even on a remote machine. This is designed so that you can use Hudson as a dashboard of your existing automation system. See [the documentation for more details](#).

Build a free-style software project

This is the central feature of Hudson. Hudson will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Build multi-configuration project (alpha)

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Build a maven2 project

Build a maven2 project. Hudson takes advantage of your POM files and drastically reduces the configuration.

OK

Hudson ver. 1.300

图表 9 Hudson – 新建一个 Job 首页

输入[Job name]，选择[build a free-style software project]，我们这次实践指南只用到了这种类型的 Job；点击[OK] 按钮，这时，你已经建立起一个新的 Job 了。这时，Hudson 会自动把页面跳转到这个 Job 的配置页面，如下图所示：

---

Project name

Description

?

☐ Discard Old Builds

?

☐ This build is parameterized

?

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

?

☒ Tie this project to a node

?

Node

▼

Advanced Project Options

Advanced...

Source Code Management

☒ None
☐ CVS
☐ Subversion

Build Triggers

☐ Build after other projects are built
☐ Build periodically
☐ Poll SCM

?

?

?

Build

Add build step ▼

Post-build Actions

☐ Publish Javadoc
☐ Archive the artifacts
☐ Aggregate downstream test results
☐ Publish JUnit test result report
☐ Build other projects
☐ Record fingerprints of files to track usage
☐ Publish Checkstyle analysis results
☐ Publish Cobertura Coverage Report
☐ E-mail Notification

?

?

?

?

?

?

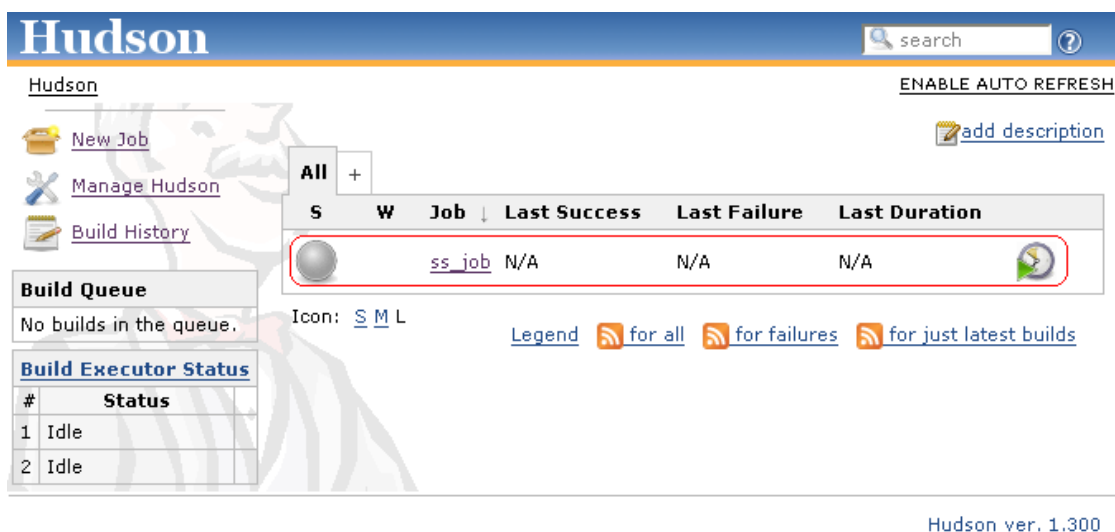
?

?

Save

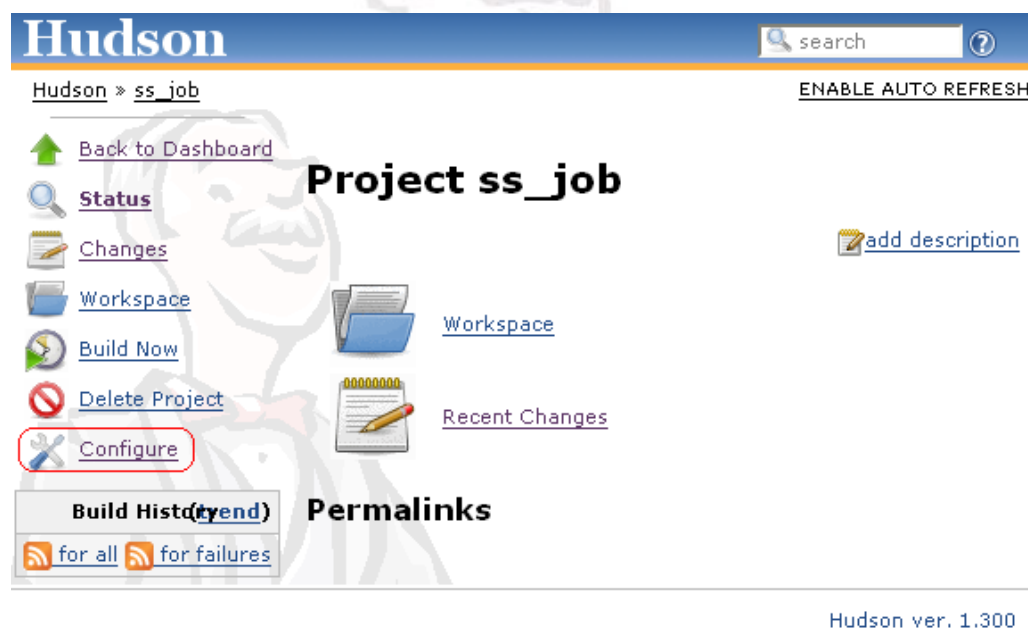
图表 10 Hudson – Job 配置页面

你可以直接开始对 Job 的配置项进行设置，或者放弃当前操作，点击页面左上方的 Link[hudson]退回到系统首页；由于当前已经有一个 Job 了，hudson 的首页将和你这前看到的有所区别：



图表 11 Hudson 系统首页

如上图所示，这个时候，你能看到在 Job 列表里多了一个 Job；同时，针对未来 Job 的运行情况，Job 列表下有三个 RSS 订阅项可以选择。点击 Job 的名字（当前实例使用的 Job 名是 ss\_job），可以进入 Job 首页，如下图所示：



图表 12 Hudson – Job 首页

点击页面左侧菜单栏的[configure]一项，同样可以进入[图 2-10 Job 配置页面]。

### [配置 Job]

在[图 2-10]中可以看到，每一个配置项后都[?]这样一个标记，点击后就能看到真对该输入项的说明，对我个人来说，这些提示在配置 Job 时带来了很大帮助。



### 1. Project name:

默认会把 Job 名称作为 Project 的名称，如果你改写了他的默认值并提交保存，相当于对该 Job 做了重命名操作。

Project name

### 2. Description:

Job 的描述信息，在 Job 首页说明性文字，以增强用户体验，如 [Super Sales Job]。

Description

### 3. Tie this project to a node:

把当前 Job 绑定一个 Hudson 节点，当前只有一个节点可选。

☒ Tie this project to a node

Node

### 4. Source Code Management:

我们使用的是 SVN，所以选择[Subversion] radio 按钮，根据项目的实际情况，[Repository URL]和[Local module directory] 的值是：

`svn://192.168.11.216/supersales/code/trunk/supersales`  
`supersalesjob`

#### Source Code Management

- ☐ None  
☐ CVS  
☒ Subversion

Modules

Repository URL  ?

Local module directory (optional)  ?

### 5. Use update:

选择了这一项，Hudson 将会尝试每次只从 SVN 取更新过的源文件，这可以使 build 变得更快。

Use update ☒

If checked, Hudson will use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

### 6. Build periodically:

这一项添写 build 动作被触发的时间，你可以通过帮助信息来了解表达式的编写规

则；我们项目要求的是在每天上午 10:05 和晚上 22:05 这两个时间点自动执行 **build** 操作，所以可以写成下图所示的形式：

☒ Build periodically

Schedule

```
5 12 * * *
5 22 * * *
```

## 7. Build:

配置 **build** 的具体步骤，点击[Add build step] 下拉框，由于我们使用的是 **Ant** 做 **build** 工具，选择[Invoke Ant]这一项：

**Build**

Add build step ▼

- Execute shell
- Invoke top-level Maven targets
- Execute Windows batch command
- Invoke Ant**

这时，会看到下图所示的输入框：[Ant Version] 选择[ant\_home]，[Targets]输入[jar]，这里的[jar]是我们项目中 **Ant Build** 角本中的一个任务。

**Build**

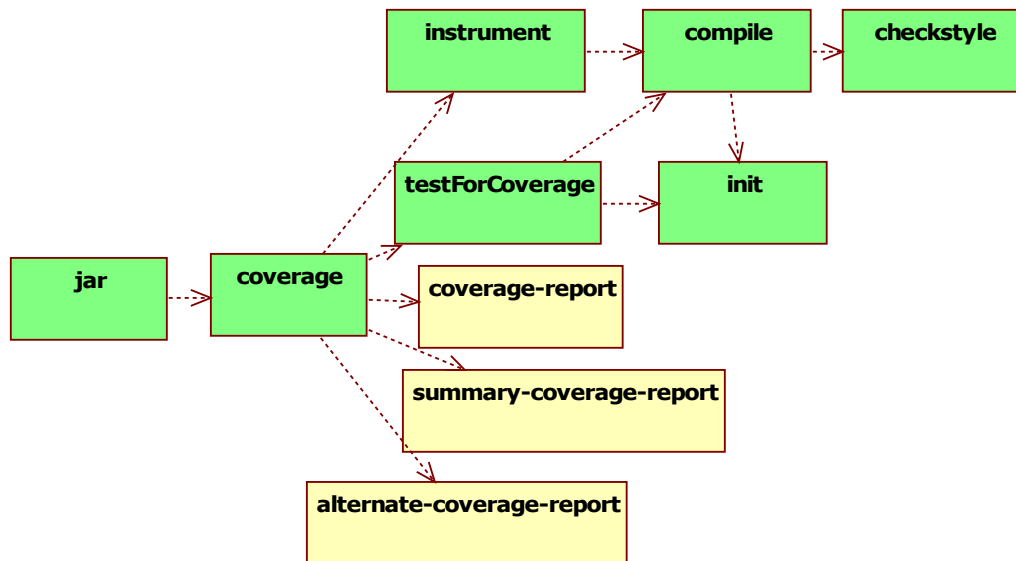
**Invoke Ant**

Ant Version ant\_home ▼

Targets jar ▼

为了让大家对我们系统的 **build** 角本有一个初步了解，下面给出了一个概览图，描述了各个 **Task** 之间的依赖关系，绿色标识的为主 **Task** 流，如图所示，**build** 的过程中将依次进行如下任务：

[代码检查]→[初始化]→[编译]→[单元测试]→[单元测试覆盖率报告]→[打包部署]



其中：

**代码检查：**依赖 CheckStyle，需要下载专用的 Hudson 插件对该项目支持；

**单元测试：**依赖 JUnit，Hudson 已经内置了对该项目的支持；

**单元测试覆盖率报告：**依赖 Corbertura，需下载专用的 Hudson 插件对该项目支持。

关于 ant 角本的编写方法，请查阅相关资料：

## 8. Publish JUnit test result report:

设置开放 JUnit 测试结果报告。如下图，选中选择框，并正确设置报告生成的目的路径，该路径对应 build 角本中的单元测试报告生成的路径：

☒ 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](#).

## 9. Public Checkstyle analysis results:

设置开放静态代码检查报告。如下图，选中选择框，并正确设置报告生成的目的路径，该路径对应 build 角本中的代码检查报告生成的目的路径：

☒ 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.

## 10. Publish Cobertura Coverage Report:

设置开放单元测试覆盖率报告。如下图，选中选择框，并正确设置报告生成的目的路径，该路径对应 build 角本中的单元测试覆盖绿报告的生成路径：

☒ Publish Cobertura Coverage Report

Cobertura xml report pattern




This is a file name pattern that can be used to locate the cobertura xml report files (for example with Maven2 use `**/target/site/cobertura/coverage.xml`). The path is relative to the module root unless you are using Subversion as SCM and have configured multiple modules, in which case it is relative to the workspace root. Cobertura must be configured to generate XML reports for this plugin to function.

Consider only stable builds ☒

Include only stable builds, i.e. exclude unstable and failed ones.

Coverage Metric Targets

| Metric       | Threshold | Unstable | Failed |
|--------------|-----------|----------|--------|
| Methods      | 80        | 0        | 0      |
| Lines        | 80        | 0        | 0      |
| Conditionals | 70        | 0        | 0      |

Configure health reporting thresholds.  
For the  row, leave blank to use the default value (i.e. 80).  
For the  and  rows, leave blank to use the default values (i.e. 0).

## 11. E-mail Notification:

设置 email 提醒选项，如下图所示，选中复选框并输入收件人地址，这样一来，在该 Job 的 CI 活动时发生异常时，就可以给相关人员发送 email 了，该 email 将会通过在系统设置时配置的 SMTP 服务器发送：

☒ E-mail Notification

Recipients

Whitespace-separated list of recipient addresses. E-mail will be sent when a build fails.

☒ Send e-mail for every unstable build

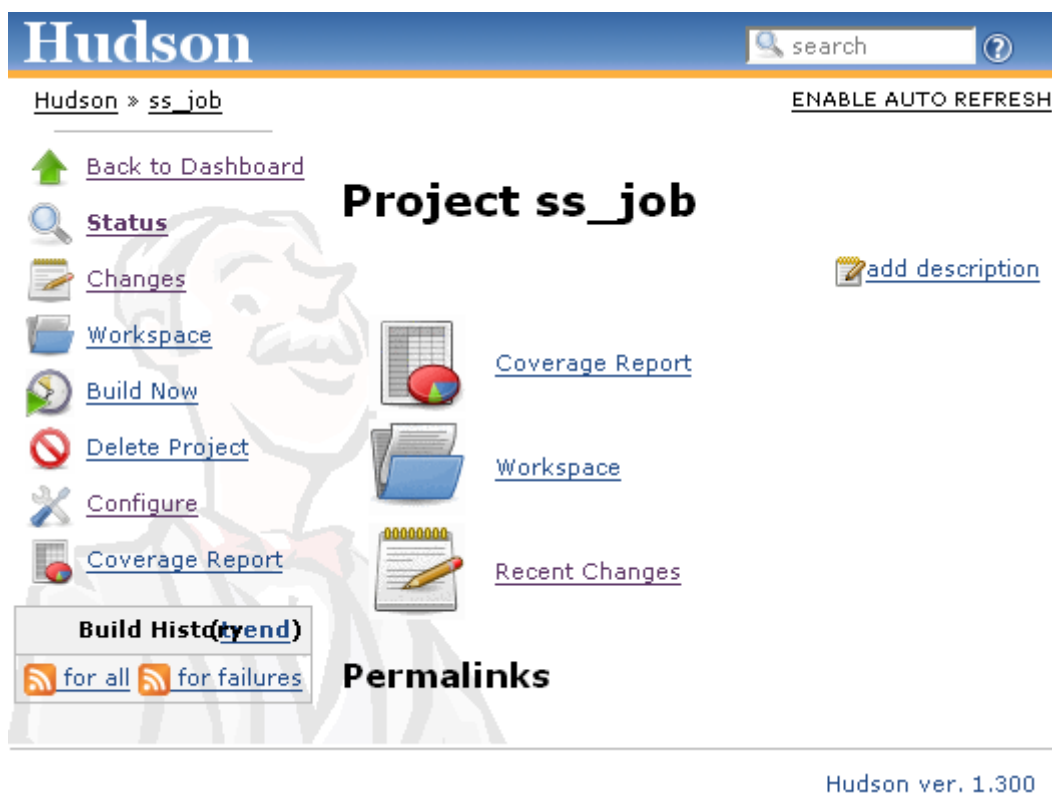
☒ Send separate e-mails to individuals who broke the build

做完上述设置，点下页面下方的[save]按钮保存，你就完成对一个 Job 的配置了。

## 2.4 启动 Build

这一节，让我们手动启动一个 Build，看看她是怎么工作的。

回到[ss\_job]的首页。一个完成配置 Job 将如下图所示，由于还没有进行过任何 build 操作，[build history]列表为空，也没有任何可利用的统计数据信息：



图表 13 Job 首页- 完成配置

如果你还记得，我们设置的 build 时间是每天的上午 9:05 和晚上 22:05，为了尽快看一下 Hudson build 起来是什么样的，我们可以点击[Build Now] 链接，立即开始一个 build 操作，如下图所示：



图表 14 Job 首页 - 手动启动 build

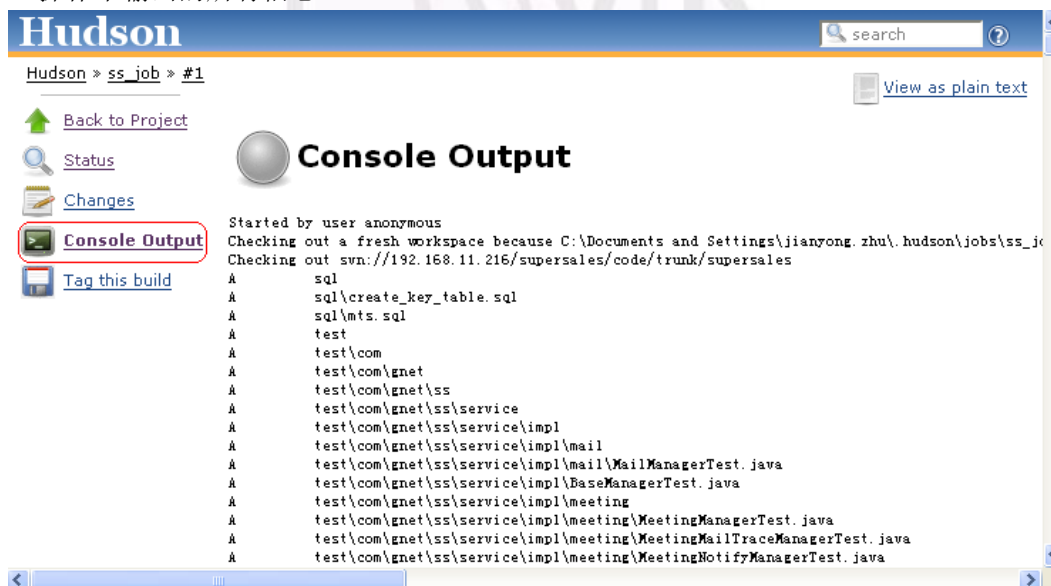
这个时候 Build History 会显现一个正在进行 build 的进度条及 build 的开始时间，你也可以点击进度右侧的红色按钮中止这次 build。

第一次 build 会一次取下 SVN 上的所有相关源文件，所以时间可能会比较长，在等待她完成这段时间里，让我们看看 Hudson 提供给我们的一些有用信息，当前的作业序号[#1]是一个可以点击的 link，点击进入 build 信息页，如下图所示：



图表 15 Job Build 信息页

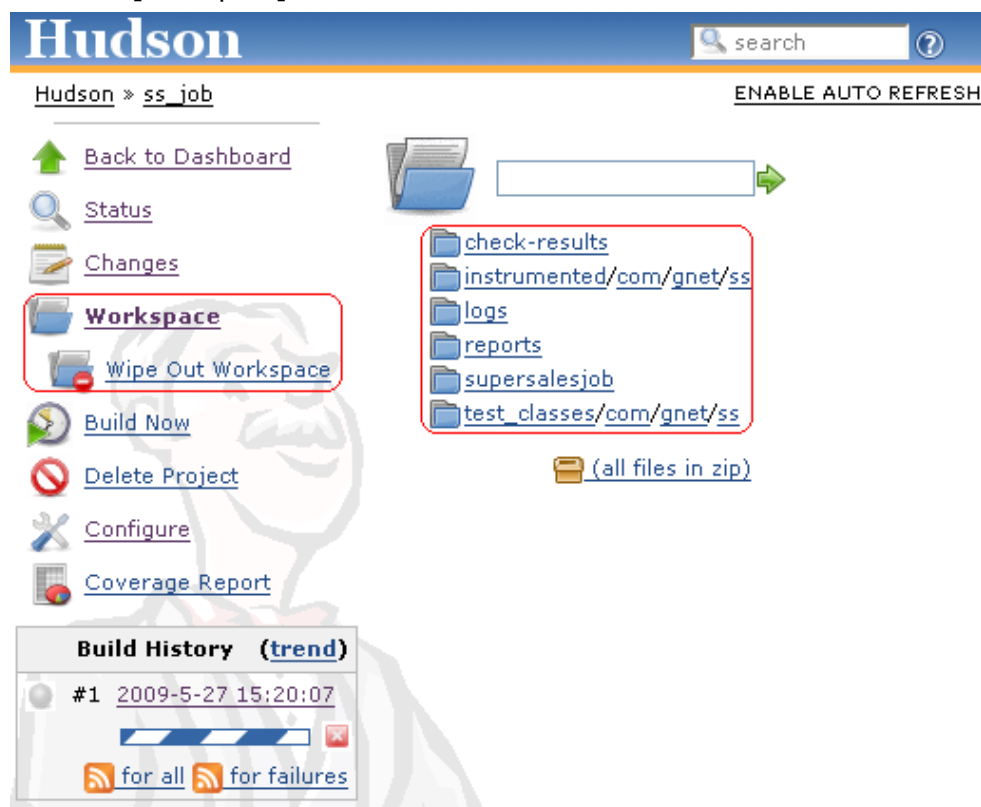
该页显示了当前 build 的一些基本信息，由于现在进行的是第一次 build 操作，Changes 信息没有内容；点击[Console Output] 链接，你能看到 Hudson 在当前 build 操作中输出的所有信息：



图表 16 Build 信息页 - Console Output

这个时候，已经有源文件从 SVN 上下载的到本地的 Hudson 工作目录下了，点击

Job 首页的[Workspace]链接，Hudson 会显示如下页面：



Hudson ver. 1.300

图表 17 Job 页面 - Workspace 信息

在这你能看到工程的源文件、生成的报告文件、日志文件及在 **build** 的过程中生成的临时文件。点击那些 **link**，可以查看文件的详情，如果文件是文本类型的，则文件内容可以直接在浏览器上显示。

视你的源文件多少及工程的复杂程度，经过一段或长或短的时间，这次 **build** 将会完成，如下图所示：



图表 18 Job 首页 - 完成一次 build

由于这一次有 13 个失败的测试用例，这里显示[黄灯]，表示这是一次不稳定的 build,同时，[静态代码检查]、[单元测试]及[单元测试覆盖率]这些信息也可以使用了，点击相应的 link 可以看到详情。

**Tips:**

1. 由于当前只有一次 build，一些插件还不能产生数据分析图表；
2. 某些插件，如 Corbertura，可以通过设置，只对稳定的 build 产生分析数据；如果是这种情况，当前这次 build 将不会产生相应的分析数据。

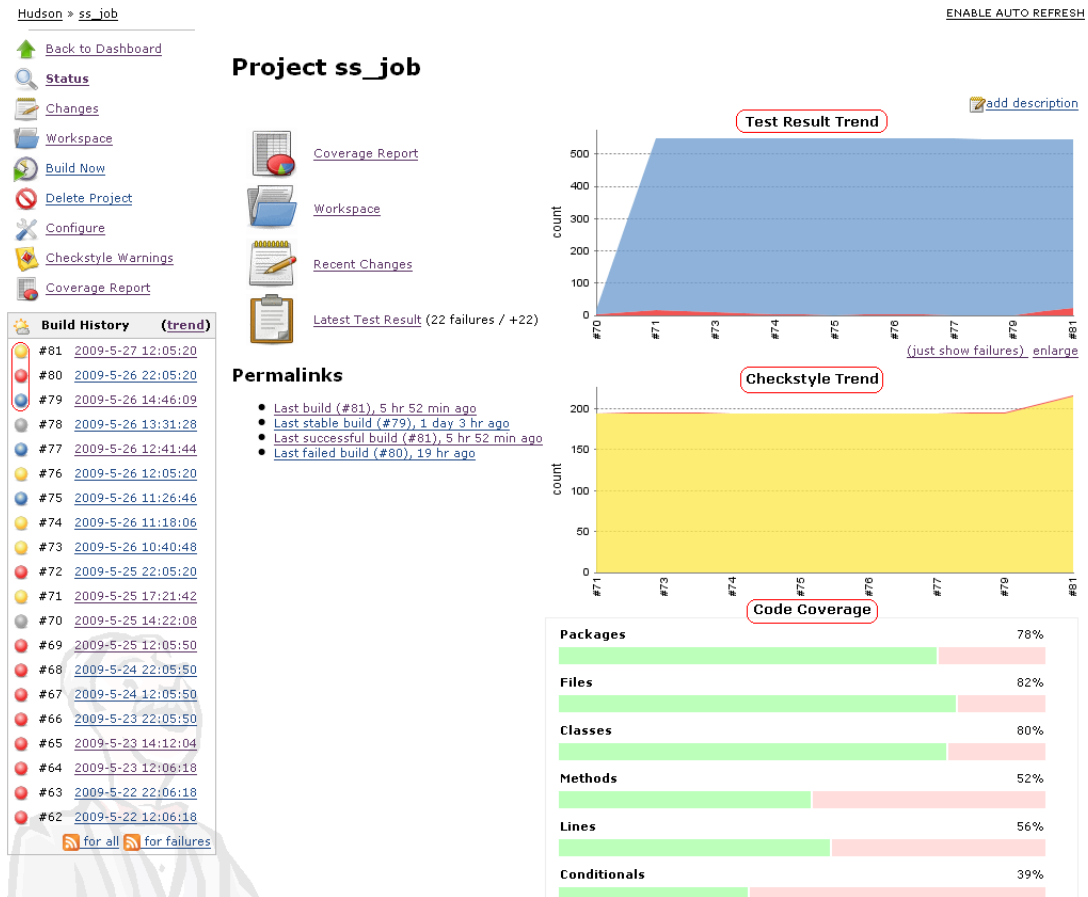
下一节，让我们看看如何利用产生的这些分析数据和报告。

## 2.5 如何利用 CI

如何利用 CI 产生的数据呢？

首先，让我看一个真实场景下 Hudson 的 Job，如下图：





图表 19 Hudson Super Sales Job

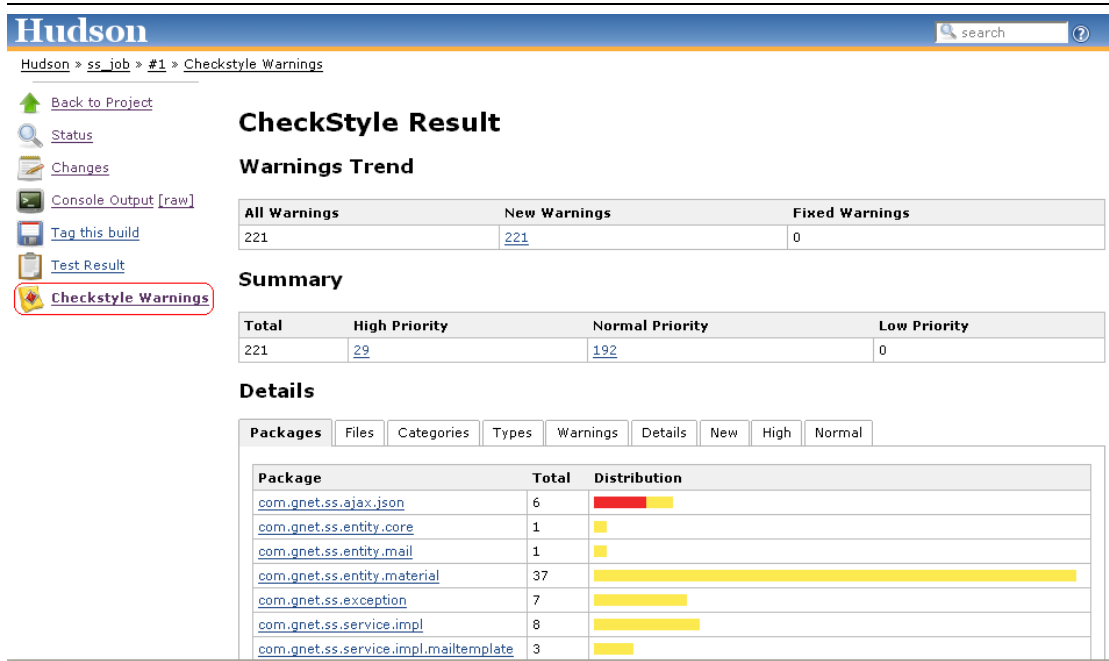
这时已经有 81 次 build 活动了，每一次 build 序号前都有一个带有颜色的灯作为标识，其中：

|   |                         |
|---|-------------------------|
| 蓝 | 成功的 build               |
| 红 | 失败的 build，如编译不能通过       |
| 黄 | 成功、但不稳定的 build，如有测试用例失败 |
| 灰 | 被中止的 build              |

在页面的右侧，依次显示了[单元测试结果趋势图]、[静态代码检查趋势图]和[单元测试覆盖率状态图]，每一个图表都可以点击进入并查看详情。

### [静态代码检查]

Hudson 可以通过友好的用户界面显示静态代码检查报告，如下图所示：

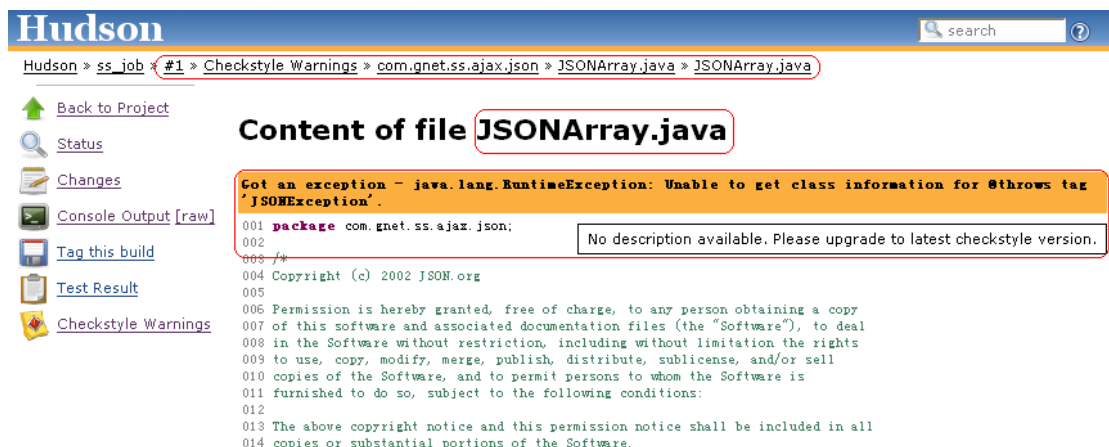


图表 20 静态代码检查报告

除了历次 build 产生的趋势以外，还有如下信息可以利用：

|                 |                   |
|-----------------|-------------------|
| All Warnings    | 异常的总数             |
| New Warnings    | 本次 build 新增的异常数   |
| Fixed Warning   | 本次 build 处理掉了的异常数 |
| Total           | 异常的总数             |
| High Priority   | 高优先级的异常           |
| Normal Priority | 一般优先级的异常          |
| Low Priority    | 低优先级的异常           |
| Details         | 分栏显示异常信息          |

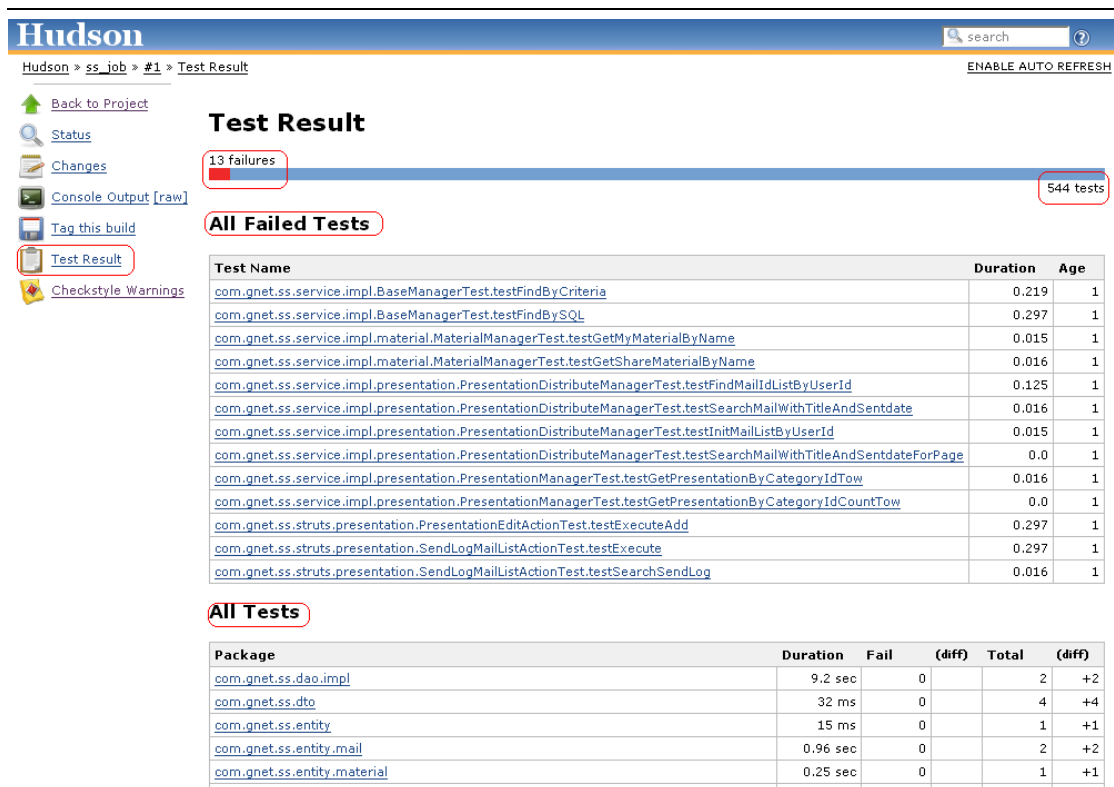
对于[Details]一项，我们可以通过[按包分类]、[按文件分类]、[按类型分类]等方式显示异常信息，通过点击相应的链接，可以反异常定位到具体的文件，并得到详细的异常信息，如下图所示：



图表 21 静态代码检查 - 异常信息详情

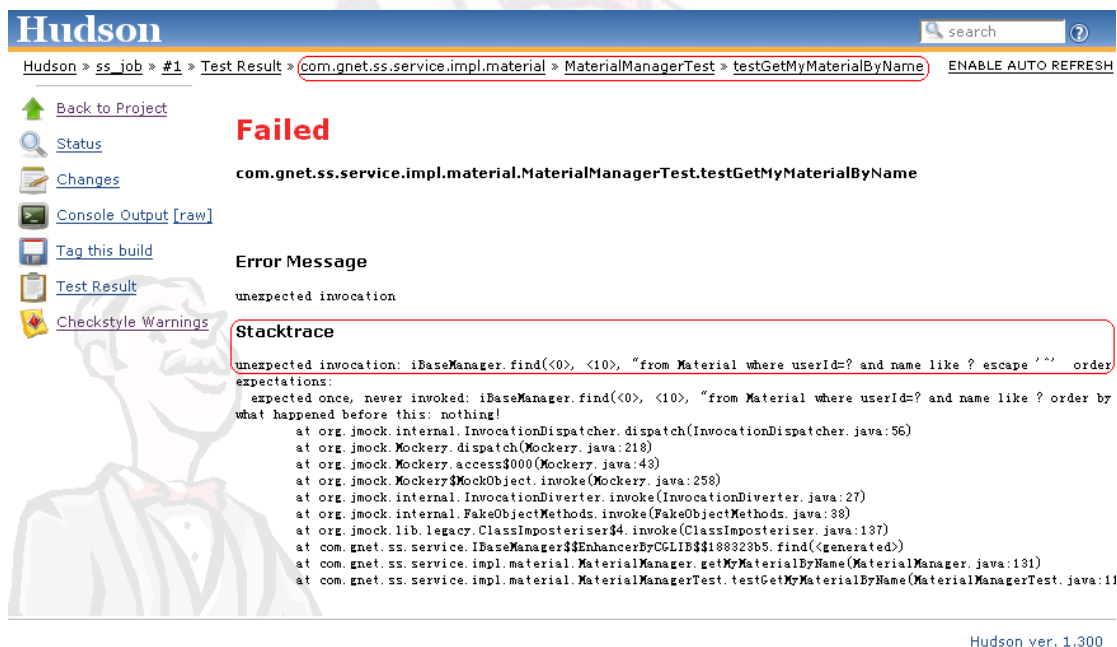
## [单元测试]

除了 Job 首页的趋势图外，详情页里有更多可利用的信息，如下图：



图表 22 单元测试报告

在这能看到你的工程里共有多少测试用例，在本次 build 中有多少个是失败的，并且会“高亮”的显示出那些失败的用例，通过点击相应的 link，你可以很方便的查看出错的详细原因：

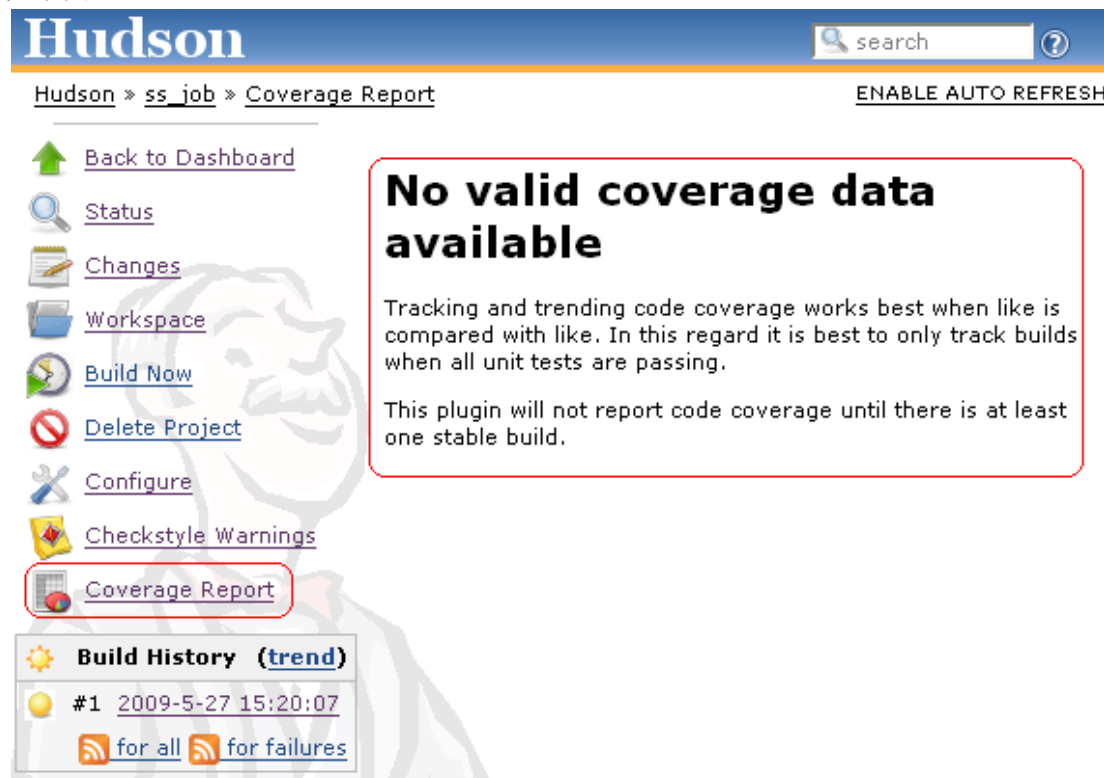


图表 23 单元测试报告 - 详情

### [单元测试覆盖率]

通过查看单元测试覆盖率报告，你可以进一步了解你的单元测试用例对项目的价值；

我们当前只有一次不稳定的 build，如果你在这个时候去看覆盖率报告，那一定会很失望，如下图：



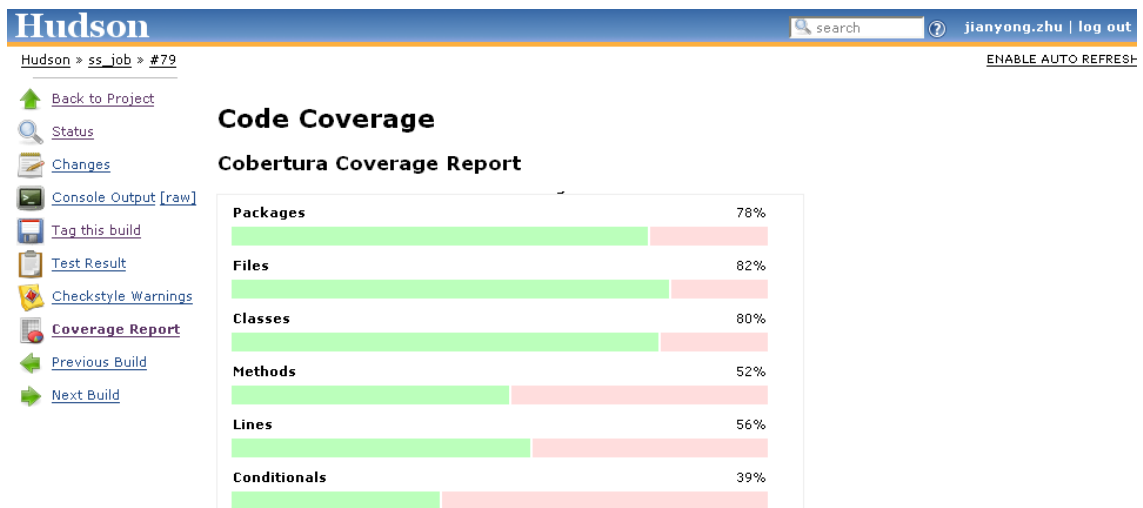
Hudson ver. 1.300

图表 24 单元测试覆盖率 1

正像提示告诉我们的，必须至少有一次稳定的 build，这个 Plug-In 才会显示报告；那还是让我看一个项目中的真实场景吧，如下图 25 所示，这是这个项目的第 79 次 build；利用这个插件，Hudson 会分类显示单元测试的覆盖情况：

|              |              |
|--------------|--------------|
| Packages     | 按包统计的代码覆盖率   |
| Files        | 按源文件统计的代码覆盖率 |
| Classes      | 按类统计的代码覆盖率   |
| Methods      | 按方法统计的代码覆盖率  |
| Lines        | 按代码行统计的代码覆盖率 |
| Conditionals | 按条件统计的代码覆盖率  |

这些数据对于分析单元测试的质量是很有参考意义的。



#### Project Coverage Summary

| Name                      | Packages    | Files         | Classes       | Methods         | Lines            | Conditionals    |
|---------------------------|-------------|---------------|---------------|-----------------|------------------|-----------------|
| Cobertura Coverage Report | 78% (25/32) | 82% (195/238) | 80% (200/250) | 52% (1734/3360) | 56% (7282/12989) | 39% (1105/2811) |

#### Coverage Breakdown by Package

| Name  | Files        | Classes      | Methods       | Lines         | Conditionals |
|---|--------------|--------------|---------------|---------------|--------------|
| <a href="#">com.gnet.ss.struts.mailtemplate</a>     | 82% (9/11)   | 82% (9/11)   | 61% (75/123)  | 66% (211/320) | 70% (28/40)  |
| <a href="#">com.gnet.ss.struts.material</a>         | 100% (15/15) | 100% (15/15) | 78% (98/125)  | 91% (303/333) | 78% (14/18)  |
| <a href="#">com.gnet.ss.service.impl</a>            | 100% (3/3)   | 100% (3/3)   | 68% (21/31)   | 47% (63/134)  | 22% (8/36)   |
| <a href="#">com.gnet.ss.entity</a>                  | 100% (1/1)   | 100% (1/1)   | 100% (5/5)    | 100% (17/17)  | 92% (11/12)  |
| <a href="#">com.gnet.ss.entity.meeting</a>          | 100% (9/9)   | 100% (9/9)   | 67% (78/117)  | 72% (123/171) | 100% (0/0)   |
| <a href="#">com.gnet.ss.struts</a>                  | 0% (0/4)     | 0% (0/4)     | 0% (0/23)     | 0% (0/90)     | 0% (0/24)    |
| <a href="#">com.gnet.ss.struts.material.servlet</a> | 0% (0/2)     | 0% (0/3)     | 0% (0/9)      | 0% (0/85)     | 0% (0/16)    |
| <a href="#">com.gnet.ss.struts.mail</a>             | 100% (25/25) | 100% (26/26) | 60% (194/325) | 67% (568/842) | 86% (96/112) |
| <a href="#">com.gnet.ss.dao.impl</a>                | 100% (1/1)   | 100% (1/1)   | 100% (4/4)    | 100% (6/6)    | 100% (0/0)   |

图表 25 单元测试覆盖率 2

## 2.6 进阶

在这一节，我会和大家分享一些使用 Hudson 的心得。

### [保留最新的 Build 记录]

在每次 build 活动中，Hudson 都会在工作目录下保留所有的工程文件，经过一段时间以后，这将占用大量的硬盘空间，这有可能是不必要的；Hudson 提供设置项，让你有选择的保留最近 Build 记录，在 Job 的配置信息页，如下图所示：

☒ Discard Old Builds

Days to keep builds

if not empty, build records are only kept up to this number of days

Max # of builds to keep

if not empty, only up to this number of build records are kept

|                         |                |
|-------------------------|----------------|
| Days to keep builds     | 保留最近 build 的天数 |
| Max # of builds to keep | 保留最近 build 的次数 |

这时你可能会提一个问题，如果你准备把某一次 build 永远保存下去该怎么办？回去看一看图 15 吧，注意到图右角的那个按钮[keep this build forever]了吗？我猜它就是起这个作用的。

### 【用户管理】

正像之前提到过的 Hudson 通过 SVN 取得源文件的同时，也会拿到对 SVN 上的文件有过操作的用户信息，如下图所示：



| Name                          | Last Active ↑ | On                     |
|-------------------------------|---------------|------------------------|
| <a href="#">bing.wen</a>      | 6 hr 0 min    | <a href="#">ss_job</a> |
| <a href="#">zhonghai.wang</a> | 6 hr 0 min    | <a href="#">ss_job</a> |
| <a href="#">weiwei</a>        | 6 hr 0 min    | <a href="#">ss_job</a> |
| <a href="#">rigen.mo</a>      | 20 hr         | <a href="#">ss_job</a> |
| <a href="#">jiawu.liu</a>     | 20 hr         | <a href="#">ss_job</a> |
| <a href="#">li meng</a>       | 20 hr         | <a href="#">ss_job</a> |
| <a href="#">xiaoyu.wang</a>   | 2 days 0 hr   | <a href="#">ss_job</a> |
| <a href="#">lin.ge</a>        | 21 days       | <a href="#">try</a>    |
| <a href="#">Zhu Jianyong</a>  | 21 days       | <a href="#">try</a>    |
| <a href="#">Ren Dong</a>      | N/A           |                        |

Hudson ver. 1.300

图表 26 用户信息

Hudson 维护着这些用户信息，当 build 操作有异常发生时，就会给相关人员发送 email 通知。

### 【权限管理】

请参见图 18，任何能访问 Hudson 的用户都能操作右侧所有的菜单项，包括[Build Now]、[Delete Project]和[Configure]，而这些操作是不应该被所有的人执行的，或者说，这些操作只应该被 Hudson 的系统管理员执行。

Hudson 也提供了安全管理的选项，打开 Hudson 的系统管理页，如下图：

Hudson

search

Hudson

[New Job](#)
[Manage Hudson](#)
[People](#)
[Build History](#)

Build Queue

No builds in the queue.

Build Executor Status

| # | Status |
|---|--------|
| 1 | Idle   |
| 2 | Idle   |

Home directory

C:\Documents and Settings\jianyong.zhu\hudson

System Message

# of executors

2

Quiet period

5

☒ Enable security

TCP port for JNLP slave agents

☐ Fixed :
 ☐ Random
 ☐ Disable

Access Control

Security Realm

☐ Hudson's own user database
 ☐ LDAP
 ☐ Delegate to servlet container

Authorization

☐ Matrix-based security
 ☐ Logged-in users can do anything
 ☒ Anyone can do anything
 ☐ Project-based Matrix Authorization Strategy
 ☐ Legacy mode

图表 27 Hudson – 安全设置 1

首先，选中[Enable security]前的选择框，在下面的[Access Control]中有[Security Realm]一项，如下图所示：

Security Realm

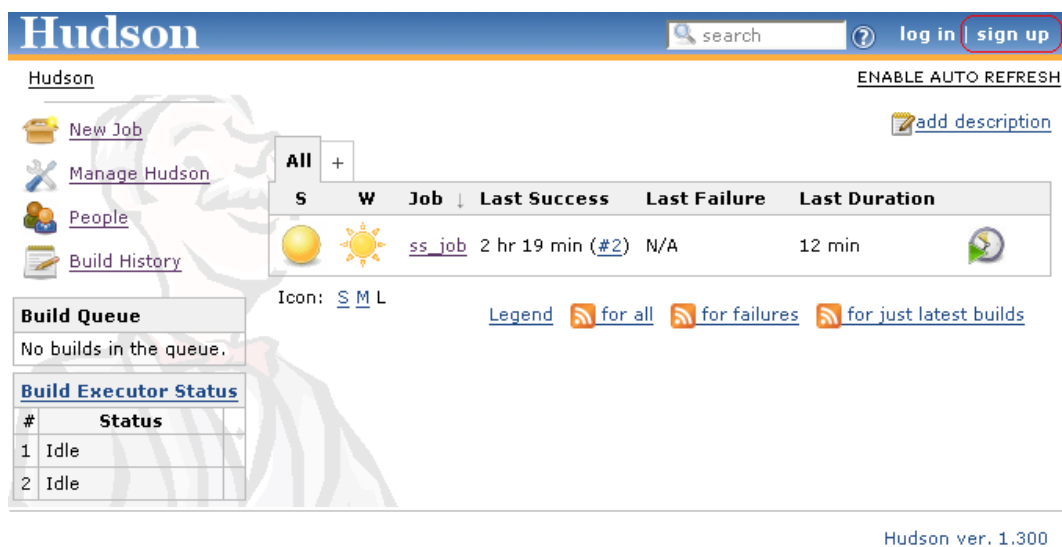
☒ Hudson's own user database

☒ Allow users to sign up

☐ LDAP

☐ Delegate to servlet container

我们可以使用 Hudson 自己的用户数据库，选中[Hudson's own user database]一项，同时选中下面自动展开的附选框[Allow users to sign up]，确认保存，并返回 Hudson 系统的首页，如下图所示：



Hudson

search log in **sign up**

Hudson ENABLE AUTO REFRESH

New Job Manage Hudson People Build History

Build Queue  
No builds in the queue.

Build Executor Status

| # | Status |
|---|--------|
| 1 | Idle   |
| 2 | Idle   |

Icon: S M L Legend for all for failures for just latest builds

Hudson ver. 1.300

图表 28 Hudson 首页 - 安全设置

这时，页面的右上会出现一个[sign up]的链接，点击进入 Sign up 页面，如下图所示：



Hudson

search log in | sign up

Hudson

New Job Manage Hudson People Build History

Build Queue  
No builds in the queue.

Build Executor Status

| # | Status |
|---|--------|
| 1 | Idle   |
| 2 | Idle   |

## Sign up

Username: jianyong.zhu

Password: .....

Confirm password:

Full name:

E-mail address:

Enter text as shown:

Sign up

Hudson ver. 1.300

图表 29 Hudson 注册管理员信息

按提示输入必要信息，并按下[Sign up]保存；重新返回 Hudson 的系统设置页面，如下图所示：



☒ Enable security

TCP port for JNLP slave agents ☐ Fixed :  ☒ Random ☐ Disable

Access Control

**Security Realm**

☒ Hudson's own user database

☐ Allow users to sign up

☐ LDAP

☐ Delegate to servlet container

**Authorization**

☐ Matrix-based security

☒ Logged-in users can do anything

☐ Anyone can do anything

☐ Project-based Matrix Authorization Strategy

☐ Legacy mode

图表 30 Hudson – 安全设置 2

选中[Authorization]下的 [Logged-in users can do anything]一项并保存。返回 Hudson 的系统首页。



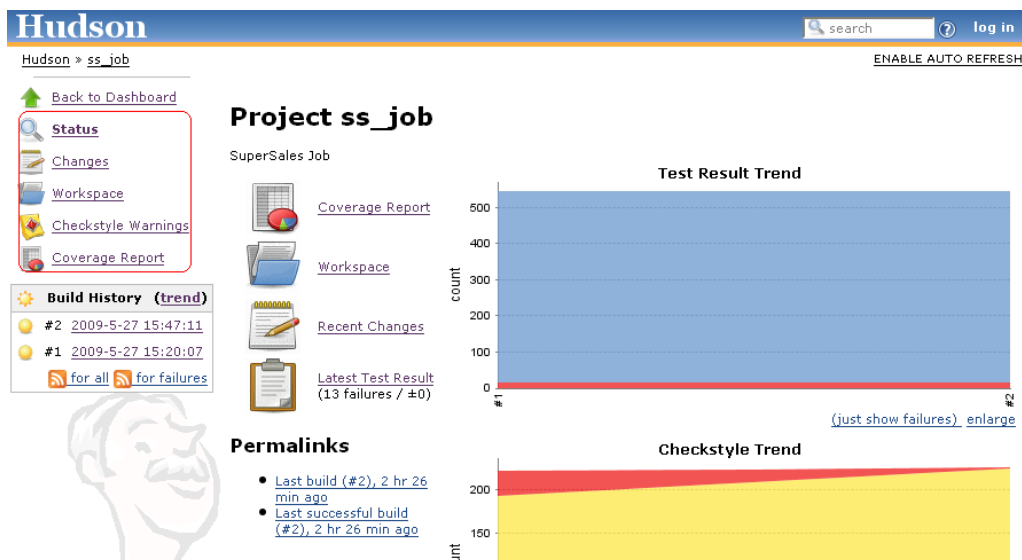
The screenshot shows the Hudson system homepage. At the top, there's a blue header with the 'Hudson' logo, a search bar, and a 'log in' button. Below the header, there's a navigation bar with links for 'People', 'Build History', 'Build Queue', and 'Build Executor Status'. The 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows a table with two executors, both in 'Idle' status. The main content area displays a table of build jobs. The first job is 'ss\_job', which is currently 'S' (Success) and 'W' (Waiting). It shows the last success time as '2 hr 25 min (#2)' and the last failure as 'N/A'. The last duration is '12 min'. There are also links for 'Icon: S M L' and a 'Legend' section with links for 'for all', 'for failures', and 'for just latest builds'.

| S | W | Job                    | Last Success                       | Last Failure | Last Duration |
|---|---|------------------------|------------------------------------|--------------|---------------|
|   |   | <a href="#">ss_job</a> | 2 hr 25 min ( <a href="#">#2</a> ) | N/A          | 12 min        |

| # | Status |
|---|--------|
| 1 | Idle   |
| 2 | Idle   |

图表 31 Hudson 安全设置 – 系统首页

这时，页面的右上会显示一个[log in]的链接；试着点击 Job 的链接查看他的详细信息，如下图所示：



图表 32 Hudson 安全设置 - Job 首页

你会发现左侧的菜单少了几项，现在能做的就只是查看该 Job 的状态信息，而无法做出任何更新操作了。如果有必要进行这些操作，就必须先通过刚才建立的管理员帐号登录 Hudson 系统。

### [其它]

这里还有一些建议，其中很多并不只是针对基于 Hudson 的 CI，它们同样也是日常开发过程中要注意的一些问题：

#### Tips:

1. 开发人员需确保代码在本地可以成功编译后再提交至配置库；
2. 开发人员需确保提交至配置库代码的完整性；
3. 更新配置库中的源文件时，为之添加有意义的说明性文字；
4. Hudson 服务由专人维护；
5. Schedule 可以根据实际情况调整，必要时可以手动执行；
6. CI 过程中发现的问题，相关责任人需及时做出响应；
7. 每次编译部署生成的 web 工程包以时间戳命名，以便于追溯问题。

## 3 附录

### 3.1 相关资源

下表是这个指南提及的所有软件、相应的版本，和你能找到这些资源的位置：

| 资源                     | 版本              | 说明及资源   |
|------------------------|-----------------|---|
| <b>SVN</b>             | <b>1.5</b>      | 配置管理工具<br><a href="http://subversion.tigris.org/">http://subversion.tigris.org/</a>   |
| <b>Hudson</b>          | <b>1.300</b>    | CI Server 的一个解决方案<br><a href="https://hudson.dev.java.net/servlets/ProjectDocumentList">https://hudson.dev.java.net/servlets/ProjectDocumentList</a>  |
| <b>Hudson Plug-ins</b> | <b>N/A</b>      | 在这次实践中，我们用到了两个 <b>Hudson Plug-in</b> <ul style="list-style-type: none"><li>● Checkstyle</li><li>● Cobertura</li></ul> 你可以通过下面的链接找到它们<br><a href="https://hudson.dev.java.net/servlets/ProjectDocumentList">https://hudson.dev.java.net/servlets/ProjectDocumentList</a> |
| <b>JRE</b>             | <b>1.6.0_12</b> | Java 运行时环境<br><a href="http://java.sun.com/javase/downloads/index.jsp">http://java.sun.com/javase/downloads/index.jsp</a>   |
| <b>Tomcat</b>          | <b>5.5.27</b>   | J2EE 应用服务器<br><a href="http://tomcat.apache.org/">http://tomcat.apache.org/</a>   |
| <b>JUnit</b>           | <b>3.8.1</b>    | 用于 Java 应用单元测试的开源项目<br><a href="http://www.junit.org/">http://www.junit.org/</a>  |
| <b>Ant</b>             | <b>1.7.1</b>    | 用于 Java 语言的 Build 工具<br><a href="http://ant.apache.org/">http://ant.apache.org/</a>   |
| <b>CheckStyle</b>      | <b>5.0</b>      | 用于 Java 语言的静态代码检查工具<br><a href="http://checkstyle.sourceforge.net/">http://checkstyle.sourceforge.net/</a>  |
| <b>Corbertura</b>      | <b>1.9.1</b>    | 自由软件，可配合 JUnit 使用计算单元测试覆盖率<br><a href="http://cobertura.sourceforge.net/">http://cobertura.sourceforge.net/</a>   |
| <b>SMTP</b>            | <b>N/A</b>      | 对应本指南实践，使用了公司的邮件服务器<br><a href="mailto:smtp.gnetis.com">smtp.gnetis.com</a>   |

### 3.2 联系作者

由于时间仓促，本文难免存在问题，请读者见谅；对基于 Hudson 的 CI 还有什么问题，欢迎通过下面的 email 地址联系作者共同讨论：

**Email: [jianyong.zhu@gnetis.com](mailto:jianyong.zhu@gnetis.com)**