前言

众所周知，Hudson 一直以来都被认为是 JAVA 平台最流行，使用得最广泛的开源持续集成系统了，它以其人性化的界面功能，丰富的插件，高度的可扩展性，强大的分布式构建能力赢得了广泛称赞。

然而，正是因为 Hudson 在 JAVA 业界优秀表现，使得国内很少有人对它在**其他**语言领域的表现有所了解。其实 Hudson 本质上只是一个 CI（continuous-integration）系统，通过其丰富的插件和高度的扩展能力，它同样可以在其他众多的语言领域有着优异的表现，昨天正好为我们组的 **Ruby** 页面**自动化测试**脚本搭建了 Hudson 的持续集成环境，所以这里就简单谈一下我是如何做到的。

搭建 Hudson 的 windows client

要做 Ruby 页面自动化的测试，目前的解决方案是使用开源框架——Watir，而 Watir 只能运行在 windows 环境下，而我们的 Hudson 服务器是搭建在 **Linux** 环境的，难道要再部署一台 windows 环境下的 Hudson 服务么？不，再部署一台只会带来管理的混乱，我想到了使用 Hudson 的分布式构建的功能，

分布式构建是 Hudson 非常重要的特性之一，这一特性让 Hudson 有能力管理数量庞大的项目，这些项目在构建（包括编译，测试，打包，部署等）可以利用分布式的特性分别在不同的 client 上运行，同时 Hudson 还提供了丰富的对这些 client 进行管理的功能，包括资源监控，以及环境变量设置等等。

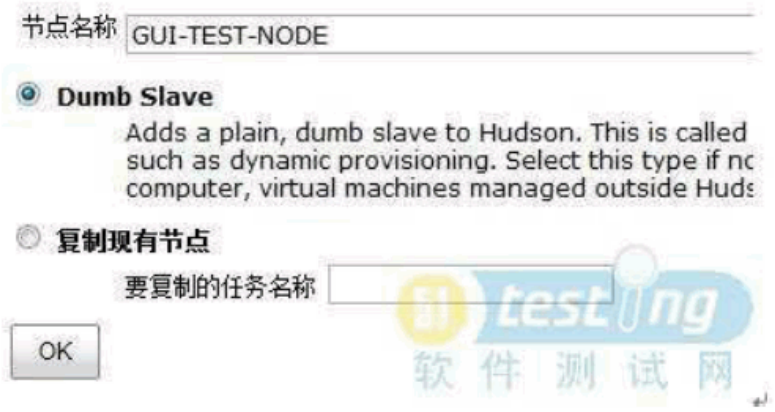
OK，说了很多废话，其实是想让大家了解一下为什么我要用这种方式。下面开始是方法：

前提：client 机器已经可以独立的运行 ruby 页面自动化测试脚本

步骤：

1. 进入页面：在要配置的 client 机器上，进入 Hudson 的 nodes 界面，一般连接为 xxx/hudson/computer/

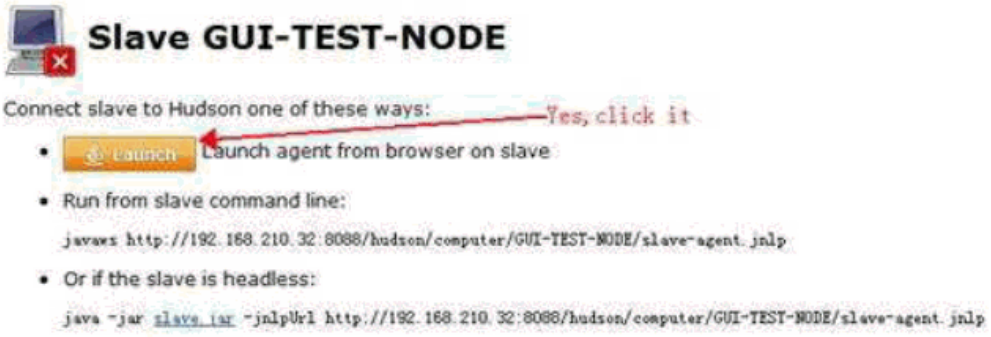
2. 新建节点：点击左侧"新建节点"，输入节点名称，选中 Dumb Slave，最后点击 OK，如下图所示：



3. 节点配置，如下图：

配置完成，点击“Save”。

4. 下载 JNLP 文件：做完上述操作后，会看见一个 JNLP 得 launch 图片，如下图



右键点击“Launch”图标，然后选择保存这个文件（在 windows 上直接点击会直接运行，为了后面的步骤，我们先保存这个文件），下载的文件名为：slave-agent.jnlp

5. 配置 client：这个是重点了，要使 Hudson 的 windows client 能够运行 GUI 测试，最好配置一个 Hudson 独有的用户账户，然后打开 windows 的自动登录功能，然后把刚刚保存的 JNLP 文件的快捷方式，放到这个账户的启动栏中。了解 Hudson 的人会知道 Hudson 可以将 client 注册为一个 windows 服务，但是这样的话就会导致 watir 无法调用 win32 的控件，导致不能完成页面测试，因此不选择注册为 windows 服务的方式（这里要非常感谢柱石同学帮助）。

6. 首次建立 Master 和 client 的连接：到目前为止我们的 Hudson 主机与客户端都是没有连接上的，现在我们点击“slave-agent.jnlp”文件，会出现弹出框：

选中"始终信任此发行者的内容"，如果不选中，以后 client 机器重启了，便不回自动连接上 master 了，到此为止就完成了 Windows client 的搭建工作了，以后如果 client 机器重启了，也会在重启完毕后自动连接 master 的。

配置 Ruby 脚本以产生测试报告

由于昨天接到的 Ruby 脚本是用 rake（据柱石说是类似 java 的 ant）来运行测试的，我在本地跑了很多次也没找到它在测试运行完成后，是怎样处理测试报告的……，因为我尝试过 N 多方式去让脚本产生测试报告。最后在 Hudson 的一个 mailing list 中找到的解决方案：使用 ci_reporter 自动将 Ruby 的测试报告转换成 Junit 风格的 xml 文件，然后 Hudson 可以直接读取 Junit 风格的 xml 文件，展示和记录测试结果，具体做法如下：

1. 安装 ci_reporter：gem install ci_reporter

2. 修改 Rakefile，增加以下几行

```
require 'rubygems'

gem 'ci_reporter'

require 'ci/reporter/rake/test_unit'
```

3. 接下来就可以通过 ci_reporter 来生成 Junit 风格的 xml 报告文件了，在运行 rake 命令的时候稍作修改即可：

```
rake ci:setup:testunit test
```

缺省情况下，会在 test/reports 目录下生成符合 Junit 风格的报告文件。

新建 Hudson job 运行脚本

上一节已经找到了合适的方式 Hudson 上是用 job 来管理要运行的项目（对于测试来说，可以理解为测试工程）的，同样，作为一个 Ruby 的自动化测试，同样也会有工程或者项目的的概念，我们需要在 Hudson 中新建一个 Job 来对应我们要运行的 Ruby 工程，配置很简单跟普通的 java 项目配置基本一样，但是有几点需要注意：

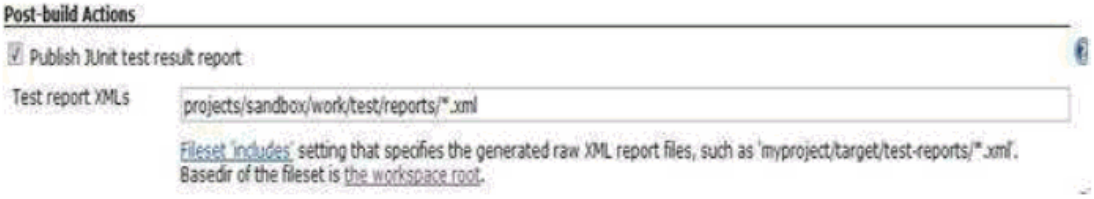
1. 需要绑定这个任务到我们刚刚配置的 client，因为如果不绑定的话，Hudson 会根据机器的负载情况自动分配当前的任务到某个 client 或者 master 上，如果被分配到 Linux 的机器上，则脚本不能调用 Win32 程序来完成界面测试，具体做法如下：

在任务配置界面中，在如上图所示的下拉框中，选中刚刚已经配置好的 client 名称“GUI-TEST-NODE”

2. 要运行 ruby 脚本，使用的 rake 命令，因此这里我们需要将这个命令配置好，同时由于我们使用了 ci_reporter，所以如上一节一样，我们稍微修改一下命令

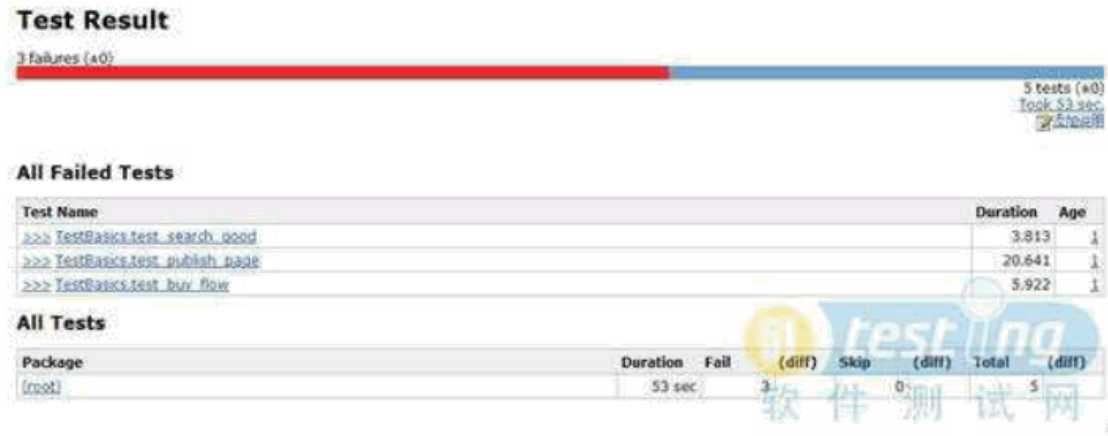
```
rake ci:setup:testunit test -trace
```

3. 最后，因为测试的最终结果是需要得到测试报告，而测试报告我们已经通过 ci_reporter 转换成了 Junit 风格的 xml 文件了，因此这里需要像传统的 Junit 测试一样，指定 Junit 报告位置，像上一节描述的一样，ci_reporter 的报告默认会在放在 test/reports 目录下。



运行并且展示结果

最后在 Job 页面点击"立即生成"开始运行这个任务，运行过程中就会将这个任务绑定到刚才已经配置好的 client 上，调用配置好的 rake ci:setup:testunit test –trace 命令，最终运行完成后将测试报告保存在 test/reports 目录下，并且在 Hudson 页面进行展示。然后通过旺旺消息发送给相关人员，点击消息中 URL 便可直接查看测试结果，如下图所示：



后记

完成这样一个尝试要感谢柱石同学提供的无私帮助，柱石同学陪我加了两天班，非常感谢！另外还要感谢 Hudson 的创始人 Kohsuke，它开创了这样好的一个产品，让我们现在 2 个人维护了接近 4000*3 这样一个数量级的用例，让我们放飞我们的思想，做任何我们想做的事情，如同 Java 之父 James gosling，Hudson 之父 Kohsuke 也与近日离开了 Oracle，近日他也宣布了 Hudson 2.0 的 road map，让我们祝福他一路走好！