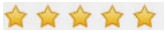
使用 XML 维护 Selenium 自动化测试脚本

祝 尚元, 测试架构设计师, 海信集团

简介： 软件自动化回归测试是软件工程实践中的重要发展趋势之一。目前业内主流的商业自动化测试工具和开放源代码测试工具，都需要自动化测试人员熟悉开发编程语言。但是过高的编程门槛让很大一部分优秀的业务测试人员只能对自动化测试驻足遥望。为解决该群体的困扰，本文将介绍如何让不熟悉编程的业务测试人员也能编写出类似传统功能测试用例的自动化测试用例，从而扫除自动化测试的技术障碍。

发布日期： 2013 年 4 月 25 日
级别： 中级
访问情况 ： 1347 次浏览
评论： 0 (查看 | 添加评论 - 登录)


平均分 (6个评分)
为本文评分

本文适用于需要为测试团队开发自动化回归测试框架的测试设计开发人员。目前业内主流的商业自动化测试工具和开放源代码测试工具，都需要测试人员熟悉开发编程语言。但是过高的编程门槛让很多测试人员只能对自动化测试驻足遥望。为解决该群体的困扰，本文将介绍一种基于 XML 维护自动化测试脚本的自动化测试框架，并详细描述了其中的技术实现细节。XML 具有编写简单，易于在任何程序中读写数据等特性，这使得 XML 成为了数据交换的唯一公共语言。使用基于 XML 的 Selenium 自动化测试框架，可以让不熟悉编程的测试人员快速编写出类似传统功能测试用例的自动化测试用例，从而扫除自动化测试的技术障碍。

本文首先会对比一下业内主流的自动化测试工具，简单分析测试人员使用这些工具时存在的问题。然后以测试人员编写的一份传统测试用例为例子，演示如何使用 Selenium 为其编写自动化测试脚本文件。接着，介绍了如何开发基于 XML 的 Selenium 自动化测试框架。最后，简单介绍了一下如何为基于 XML 的 Selenium 自动化测试框架编写更多的特性。通过比较两种自动化测试实现方式，使用 XML 维护 Selenium 自动化测试脚本的优势一览无余的展示出来。

Web 应用自动化回归测试工具一览

随着软件公司的发展，产品线不断扩大，软件新版本不断推出，支持的平台越来越丰富，如何高效、快速的对软件产品进行测试成为软件测试团队的新挑战。越来越多的项目组在考虑对软件软件产品进行自动化回归测试，考虑选取适合自己项目组的自动化工具。面对数目繁多、各式各样的自动化工具，选择也不是件很容易的事情。测试人员耳熟能详的测试工具如下表：

表格 1. 自动化测试工具

工具名称	发布公司
Rational Functional Tester	IBM
QuickTest Professional	HP
Rational Robot	IBM
SilkTest	Micro Focus
Selenium	开源软件
Watir	开源软件

如上表所示，开源的有 Selenium、Watir 等，商业的测试工具如 Rational Functional Tester、Rational Robot、QTP、SilkTest 等等。开源工具使用成本比较低，但功能不如商业工具丰富。商业工具可以提供强大的录制功能，为初学者所喜爱，但录制脚本并不是自动化测试最优方案。录制脚本中的测试数据和脚本程序混杂在一起，很难进行维护；此外，当测试用例发生较大的变更时，维护用例和重新录制用例所花费的精力不分上下。值得一提的是，RFT 作为 IBM Rational 产品家族的重要组成部分，吸引了大量的测试人员使用该工具。IBM Frame 框架，更是让 RFT 测试脚本开发效率倍增，您们可以再 developerWorks 上面发现很多关于这个框架的文章。以 Selenium 为代表的开源工具，也吸引了大量的测试人员，developerWorks 上也发表了很多文章，如《使用分层的 Selenium 框架进行复杂 Web 应用的自动测试》等等。但是，不管使用什么样的工具，都需要测试人员熟悉开发编程语言。编程门槛让很大一部分的熟悉业务的测试人员只能对自动化测试驻足遥望。那么，如何让不熟悉编程的测试人员也能编写自动化回归测试用例呢？

业务测试人员擅长什么

显然，测试人员十分熟悉业务，擅长编写传统的功能测试用例文档。本文会给出一份简化的测试用例：在搜索引擎谷歌站点查询 IBM 开发者社区，如下表：

表格 2. 测试用例

步骤编号	步骤名称	期望结果	实际结果	执行状态
1 .	打开浏览器，访问谷歌站点	谷歌首页被打开		
2 .	在谷歌搜索输入框输入“IBM Developerworks cn”	成功输入		
3 .	点击 “搜索”按钮	搜索成功，显示搜索结果		
4 .	截屏	成功截屏，图片文件保存成功		

该测试用例只包括测试步骤，每个测试步骤又分成若干组成部分：步骤编号、测试步骤、期望结果、实际结果、执行状态等。很显然，业务测试人员肯定会写出比这复杂得多的用例。我们用这份简单的测试做演示使用。下面就演示如何对这个用例做自动化测试。

使用 Selenium 为测试用例编写自动化脚本

我们先以开源的 Selenium 工具为例，开发上述用例的自动化测试脚本。Selenium 包含 Selenium-IDE、Selenium Remote Control、Selenium WebDriver 和 Selenium Grid，他们共同组成了强大的自动化测试工具。Selenium RC 和 Selenium WebDriver 都可以使用 Selenium Sever，Selenium Server 还包括内建的 Grid 功能特性。如果浏览器和测试用例运行在相同的机器上并且测试用例完全基于 WebDriver API 编写，可以直接使用 WebDriver 驱动浏览器，而不必使用 Selenium Server。在特定情况下，比如计划使用 Selenium Grid 创建分布式自动化测试环境、或需要连接

到安装特别版本浏览器的远程机器上执行测试用例脚本，必须使用 Selenium Server。下面分别简单介绍一下各个组件：

- Selenium 2（ 又名 Selenium WebDriver)：Selenium 2.0 的主要新特性就是集成了 Selenium WebDriver API。WebDriver 被设计用于提供简化精炼的编程接口，并解决了在 Selenium RC 中的一些限制。WebDriver 可以更好的支持使用 AJAX 技术的动态网页。总的来说，WebDriver 提供设计良好的面向对象的接口和改进的对动态网页支持，是 Selenium 的未来发展方向。
- Selenium 1(Selenium Remote Control)：曾是 Selenium 的主要组成部分，现已经正式被 Selenium 官方弃用。替代者为 Selenium 2（Selenium WebDriver）。更多详细内容，请参考 Selenium 项目简史。
- Selenium IDE：Selenium-IDE 是快速开发 Selenium 测试用例的 Firefox 插件工具。可以利用它录制用户的基本操作，生成测试用例脚本，还可以将这些测试用例转化为其他编程语言的自动化测试脚本。通过点击回放按钮，可以回放录制的测试用例脚本。该工具还包含一个上下文菜单，测试工程师可以使用它从浏览器的当前显示页面选择页面元素，然后可以选择适配已选取元素的带预定义参数的 Selenium 命令来快速完成脚本编写。这一特性为自动化测试人员节省了花费在脚本编写上的时间、也提供了学习 Selenium 命令的快捷方法。
- Selenium Grid：Selenium Grid 用于在不同的机器系统上、不同的浏览器上并行运行自动化测试。利用 Selenium Grid，可以轻松创建分布式自动化测试环境

下面，我们将会使用 JUnit4 和 Selenium Server 演示如何对前文提到的测试用例编写自动化测试脚本。准备项目依赖的 Jar 文件，如 Selenium、dom4j 等 Jar 包，启动 Eclipse 集成开发环境，创建项目名称为 SeleniumDemo 的 Java 项目，并创建一个 JUnit 4 单元测试类：dw.junit.DWloginJUnit.java，依次编写 @BeforeClass、@Test 、@AfterClass 方法。本文提供源代码下载，您可自行下载，导入到自己的 Eclipse 开发工作区。首先，创建 Selenium 实例，启动浏览器，使浏览器窗口最大化，打开要访问的谷歌网站。代码如下：

清单 1. 创建初始化 Selenium 实例

```
private static Selenium selenium;
@BeforeClass
public static void setUpBeforeClass() throws Exception {
    selenium = new DefaultSelenium("localhost", 4444, "*iexplore",
            "http://www.google.com/");
    System.out.println("正在启动 Selenium。。。");
    selenium.start();
    selenium.windowMaximize();
    selenium.open("/");
}
```

然后在测试方法中实现演示用例，即在谷歌站点搜索输入框输入“ibm developerworks cn”，然后点击搜索按钮，最后截图。Pause 和 captureScreenshot（）方法在本文附带的源代码文件中可查。执行用例部分的代码清单如下：

清单 2. 执行用例的程序代码

```
 @Test
 public void test() {
 // 在 Google 查询输入框，输入 ibm developerworks cn
 String queryString="ibm developerworks cn";
 selenium.type("//input[@name='q']", queryString);
                 pause(1000);
 // 单击查询按钮，执行查询
 selenium.click("//input[@name'btnK']");
                 pause(1000);
 System.out.println("获取的页面标题："+selenium.getTitle());
                 pause(2000);
 SeleneseTestBase.assertTrue(selenium.getTitle().contains(
 queryString));
 // 测试截图函数
 captureScreenshot("截图测试 JUnit");
         }
```

最后，执行用例完毕后调用 tearDownAfterClass() 方法销毁 Selenium 实例。

清单 3. 销毁 Selenium 实例

```
@AfterClass
public static void tearDownAfterClass() throws Exception {
            if(selenium != null){
            System.out.println("停止 Selenium ！");
            selenium.stop();
        }
         }
```

我们来演示如何执行这个单元测试类：先打开一个命令行窗口，并切换到 Selenium Server Jar 文件所在的目录，运行 java -jar selenium-server-standalone-2.{ 版本号 }.{ 修订版本号 }.jar 来启动 Selenium Server。在 Eclipse 工作区，右键选择 dw.junit.DWloginJUnit.java，选择 Run as — JUnit Test 来运行单元测试类。可以看到，浏览器自动启动起来，访问谷歌查询引擎主页，在搜索框自动输入了“ibm developerworks cn”，并点击查询按钮进行了查询。在 Eclipse 的 JUnit 视图，可以看到显示代表执行成功的绿色图标，在 SeleniumDemo 项目的根目录下生成了一个 png 格式的谷歌查询结果页面的截图文件。

自动化测试框架开发人员如何做

我们上文实现了演示用例的自动化回归测试，很多公司的测试工程师也是用这样的方法执行自动化测试的，但这不是最好的方法。理由如下：

1. 如果用例发生变更，比如界面的美化重构导致页面的某一控件元素信息变化，就不得不修改 Java 脚本代码，重新编译打包。
2. 擅长业务的测试人员很大一部分还是不熟悉 Java 编程的，他们期望有一种不需要编程的测试框架，这样他们就可以根据对业务的把握，快速编写自动化测试用例脚本。

目前市场上还没有自动化测试工具可以直接识别传统的测试用例，继而直接执行自动化测试。也就是说，现在还没有工具可以读取像“用户名输入框”这样的纯文字性描述内容，然后映射到页面上的输入框控件。那么，如何添加一些额外的信息到传统的用例里面，从而使自动化工具能执行业务测试人员写的这些用例呢？

先分析一下 Web 应用的测试用例及其测试步骤，显而易见，都包含下述几部分：Web 页面控件部分、用户执行动作部分、测试数据部分等。比如，演示用例中提到的用户输入框、密码输入框和登录按钮属于 Web 页面控件，动词输入、点击等属于用户执行动作，用户名和密码数据属于测试数据。我们需要把这几部分都添加额外信息、转化为测试工具能识别的形式。测试工具都提供定位 Web 页面控件的方式，以 Selenium 为例，它可以通过 CSS、JS 和 XPath 定位控件，方式非常丰富。用户执行动作对应着测试工具的应用编程接口 API，测试工具都提供类似单击 Click、双击 DoubleClick、拖拽 DragAndDown、输入数据 Type 等动作的 API。页面控件信息和测试数据对应着 API 的参数列表。比如，selenium.type(“//input[@id= ’ username ’]”,”我的用户名”)。

下面怎么做呢？传统的测试用例文档存储方式在不同的公司里差异很大，有的是 Word 文档，有的是 Excel 文档，也有维护在配置管理站点的，比如 IBM 的 Rational ClearQuest、开源的 TestLink 等等。Word 和 Excel 在维护用例额外的步骤信息方面都不够灵活简便，我们决定使用 XML 文档。下面是使用 XML 维护本文中演示用例。

图 1. 浏览器打开测试用例

可以看出来，这和传统的用例看起来很像。这样的用例编写方式，阅读起来非常容易。同事在会议室一起评审这个用例时，很容易就能看明白这个测试用例的执行行为。但是又与传统测试用例不同的是，它包含了额外的能被自动化工具识别的步骤信息。我们使用 XML 编辑器打开这一个测试用例文件，如下：

清单 4. XML 格式的自动化测试脚本

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="./SeleniumUseCase.xsl"?>
<SUITE name="IBM 开发者社区 演示测试用例">
<SETTINGS browser='*firefox' seleniumServer='127.0.0.1'
                seleniumPort='4444' application_url='http://www.google.com/'>
        </SETTINGS>
<TEST name="在谷歌中查询 IBM 开发者网络社区" timeout='1'>

                <STEPS>
<STEP index="1" type="Type.INPUT" name="在谷歌查询输入框，
输入 ibm developerworks cn">
<XPATH>//input[@name='q']</XPATH>
                                <VALUE>ibm developerworks cn</VALUE>
                        </STEP>
<STEP index="2" type="Type.CLICK" name="单击查询按钮，执行查询">
                                <XPATH>//input[@name='btnK']</XPATH>
                        </STEP>
<STEP index='3' type='Type.CLASS' name="测试截图函数">
<METHOD>captureScreenshotRemote</METHOD>
<ARGUMENT> 查询后截图测试 Remote</ARGUMENT>
                        </STEP>
                </STEPS>
        </TEST>
</SUITE>
```

下面对上述 xml 进行解释：

第一行是 XML 声明。它定义 XML 的版本 (1.0) 和所使用的编码 (UTF-8) 。紧接着是指定的处理指令。设置的 XLST 转换，可以使测试用例 XML 文件能被浏览器打开并显示为 HTML 形式。XSLT 是一个 W3C 标准，可将 XML 转换为其他的格式，比如 HTML 格式。本文提供的下载附件里包括引用的 XSL，这里不再赘述。每个 SUITE 元素可以指定用例执行的配置信息，比如浏览器和 Selenium 信息等。每个 TEST 元素对应着一个测试用例，每个 TEST 包含若干 STEP 元素，对应着不同的测试步骤。每个 STEP 步骤包括步骤编号 index、 步骤类型 type、步骤名称 name，也包含页面控件定位信息 XPATH 和测试数据 VALUE 等。

下文来演示，如何使用自动化测试工具开发自定义的测试框架来执行上述 XML 用例。本中定位控件元素的使用的是 XPath，我们就以 Selenium Server 为例进行自动化框架的开发。IBM 公司的 Rational Functional Tester 工具也是支持 XPATH 定位控件元素的，请参看 developerWorks 文章《使用 XPath 在 Rational Functional Tester 中动态识别对象》。读者如果感兴趣，可自行基于 RFT 开发自己的自动化测试框架。

XML 用例驱动 Selenium 框架开发

我们创建另一个 JUnit 4 单元测试类：dw.xml.DWloginXML.java，依次编写 @BeforeClass、@Test、@AfterClass 方法。读者可自行下载本文附带的源代码项目导入到自己的 Eclipse 开发工作区。下面对代码进行讲解：

1. 解析测试用例 XML，读取执行设置信息

   使用 Dom4J 来解析 XML 用例脚本文件，ucXMLFile 变量存放测试用例文件路径。首先解析 SETTINGS 元素，读取其中的属性设置。可以看出，使用 XML 维护测试用例，能轻松维护 Selenium Server 的相关信息。如果想换其他浏览器执行这个用例，只需修改 XML 文件中 SETTINGS 元素的 browser 属性，比如把“*iexplore”改为“*firefox”，运行的时候就会启动 firefox 浏览器来执行该用例，而不需要改动任何 Java 程序代码。同样，如果喜欢使用谷歌 Chrome 浏览器，则需要设置为“*googlechrome”。当然，需要预先在启动 Selenium Server 的机器上安装相应的浏览器软件。

   清单 5. 使用 DOM4J 解析 XML 脚本文件

```
Document document = null;
try {
System.out.println("测试用例 XML 文件位置 ::" + ucXMLFile);
document = new SAXReader().read(new File(ucXMLFile));
} catch (DocumentException e) {
```

```
  e.printStackTrace();
                }
Element suiteElm = document.getRootElement();
// settings element
Element settings = suiteElm.element("SETTINGS");
if (settings != null) {
   browser = settings.attributeValue("browser");
   seleniumServer = settings.attributeValue("seleniumServer");
   seleniumPort = settings.attributeValue("seleniumPort");
   application_url = settings.attributeValue("application_url");
   }
```

2. 读取 STEP 测试步骤信息

获取用例级别的延时设置，在每一个测试步骤执行之前，暂停 timeout 指定的秒数。通过指定延时，可以让程序执行的慢一些，符合用户真实操作，也方便演示。获取的测试步骤信息保存在列表实例 List<Element> stepNodes。

清单 6. 获取测试步骤信息

```
// TEST element
Element testElm = suiteElm.element("TEST");
System.out.println("测试用例名称为 ::" + testElm.attributeValue("name"));
if (testElm.attributeValue("timeout") != null) {
// 用例级别的 timeout 设置 , 覆盖了默认设置!
timeout = new Integer(testElm.attributeValue("timeout")).intValue();
System.out.println("延时设置, timeout=" + testElm.attributeValue("timeout")+"秒");
        }
stepNodes = testElm.element("STEPS").elements();
}
```

3. 创建 Selenium 实例，启动浏览器，最大化浏览器窗口，打开要访问的网站

清单 7. 创建初始化 Selenium 实例

```
selenium = new DefaultSelenium(seleniumServer,
new Integer(seleniumPort).intValue(), browser, application_url);
System.out.println("正在启动 Selenium。。。");
selenium.start();
selenium.windowMaximize();
selenium.open(application_url);
```

4. 迭代解析 STEP 元素执行各个测试步骤

创建一个迭代器实例 iterator，使用 while 循环，依次执行各个测试步骤。在执行的时候，打印详细的步骤信息，方便调试。根据 type 属性指定的步骤类型，分别调用不同的 Selenium 接口。作为演示，本文提供的代码例子，仅仅包括鼠标单击、输入框输入和截图功能。通过丰富处理 SETP 步骤元素的代码，可以支持更多的功能。

清单 8. 迭代依次解析测试步骤

```
Element elm = null;
// 循环解析测试步骤
Iterator<Element> iterator = null;
iterator = stepNodes.iterator();
while (iterator != null && iterator.hasNext()) {// while 循环开始
elm = iterator.next();
System.out.println("------------------------------>");
System.out.println("step index=" + elm.attributeValue("index"));
System.out.println("step name=" + elm.attributeValue("name"));
System.out.println("step type=" + elm.attributeValue("type"));
String type = elm.attributeValue("type");
if (type == null) {
SeleneseTestBase.fail("必须定义 type 属性，请检查 XML 测试用例。");
                }
if (type.equals("Type.CLICK")) {
// 处理 Click 单击操作
pause(timeout*1000);
try {
String xpValue = elm.element("XPATH").getText();
selenium.click(xpValue);

} catch (Exception e) {
SeleneseTestBase.fail(e.getMessage()+ "\n 步骤执行失败，测试执行被中止 , \
      测试元素为 :\n" + elm.asXML());
                }
} else if (type.equals("Type.INPUT")) {
// 处理 Input 在输入框执行输入操作
pause(timeout*1000);
try {
   String inValue = elm.element("VALUE").getText();selenium.type(\
      elm.element("XPATH").getText(), inValue);
} catch (Exception e) {
SeleneseTestBase.fail(e.getMessage()
+ "\n 步骤执行失败，测试执行被中止 , 测试元素为 :\n" + elm.asXML());
                }
            }
else if(type.equals("Type.CaptureScreenshot")){
pause(timeout*1000);
String fileName = elm.element("VALUE").getText();
                captureScreenshot(fileName);
            }
```

完整的框架项目代码请参考附件进行下载。对这个单元测试类的运行和前文描述的是一致的，不再赘述。当然，现在的框架支持的功能是非常简单的（ 仅仅支持单击、表单输入和截图功能 ），框架开发人员可以添加更多的功能特性，然后就可以把该自动化测试框架交付给业务测试人员，他们就再不会为编写测试代码发愁了。

Selenium 框架的更多特性开发

通过上面的演示，我们了解了如何开发一套不需要掌握编码知识就能编写自动化测试用例的测试框架。因为篇幅的原因，本框架支持的特性比较少，框架的功能非常简单，只支持鼠标单击操作、数据输入操作和截图功能，远远不能满足自动化测试的需求。本文的最主要目的在于让读者理解这种自动化框架的理念。读者可以根据自己项目组的实际情况进行自定义开发更多的功能特性。

结束语

通过上面的介绍，我们了解到了如何向传统的功能测试用例里面添加额外的控件信息，从而形成能被自动化测试工具识别的自动化测试用例；以及如何使用 XLST 转换 XML 格式的测试用例文件成 HTML 格式，增加了用例的可读性；并通过详细的例子探讨了如何定制属于自己的基于 Selenium 二次开发的自动化测试框架。最后再次总结一下该框架的让人心动之处，整理一下思路 ， 规划属于您自己的自动化测试框架吧。

- 业务测试人员不需要编程知识，熟悉业务和浏览器，即可开发自动化测试脚本业务测试
- 人员不需要编程知识，熟悉业务和浏览器，即可开发自动化测试脚本
- 自动化测试用例和传统的功能测试用例看起来很像，阅读方便，利于评审
- 扩展性比较强大，可以开发出更多的功能特性。XML 测试用例脚本通用性强，可以适用不同的测试工具

下载

描述	名字	大小	下载方法
示例代码	SeleniumDemo.zip	9KB	HTTP

关于下载方法的信息

参考资料

学习

- 参考 IBM RFT 站点，查看更多面向测试人员的自动化测试技术参考资源。
- 访问站点 http://code.google.com/p/selenium/，了解更多 Selenium 的内容。
- 查看文章“使用 XPath 在 Rational Functional Tester 中动态识别对象”，了解如何使用 RFT 与 XPath 开发自动化测试框架
- 查看文章“使用分层的Selenium框架进行复杂Web应用的自动测试”，了解如何使用 Selenium 进行自动化回归测试。
- developerWorks Web development 专区：通过专门关于 Web 技术的文章和教程，扩展您在网站开发方面的技能。
- developerWorks Ajax 资源中心：这是有关 Ajax 编程模型信息的一站式中心，包括很多文档、教程、论坛、blog、wiki 和新闻。任何 Ajax 的新信息都能在这里找到。
- developerWorks Web 2.0 资源中心，这是有关 Web 2.0 相关信息的一站式中心，包括大量 Web 2.0 技术文章、教程、下载和相关技术资源。您还可以通过 Web 2.0 新手入门 栏目，迅速了解 Web 2.0 的相关概念。
- 查看 HTML5 专题，了解更多和 HTML5 相关的知识和动向。

讨论

- 加入 developerWorks 中文社区。查看开发人员推动的博客、论坛、组和维基，并与其他 developerWorks 用户交流。

关于作者

祝尚元，自 2006 年参加工作，先后在浪潮软件、IBM 中国研发中心和海信传媒从事软件的性能测试框架、自动化测试平台的设计与架构工作。

关闭 [x]

developerWorks：登录

IBM ID：
需要一个 IBM ID?
忘记 IBM ID?

密码：
忘记密码?
更改您的密码

保持登录。

单击提交则表示您同意developerWorks 的条款和条件。 使用条款

提交 取消

当您初次登录到 developerWorks 时，将会为您创建一份概要信息。您在 developerWorks 概要信息中选择公开的信息将公开显示给其他人，但您可以随时修改这些信息的显示状态。您的姓名（除非选择隐藏）和昵称将和您在 developerWorks 发布的内容一同显示。

所有提交的信息确保安全。

关闭 [x]

请选择您的昵称：

当您初次登录到 developerWorks 时，将会为您创建一份概要信息，您需要指定一个昵称。您的昵称将和您在 developerWorks 发布的内容显示在一起。

昵称长度在 3 至 31 个字符之间。 您的昵称在 developerWorks 社区中必须是唯一的，并且出于隐私保护的原因，不能是您的电子邮件地址。

昵称： （长度在 3 至 31 个字符之间）

单击提交则表示您同意developerWorks 的条款和条件。 使用条款.

提交 取消

所有提交的信息确保安全。

平均分（6个评分）

1 星

1 星
2 星

2 星
3 星

3 星
4 星

4 星
5 星

5 星

提交

添加评论：

请 登录 或 注册 后发表评论。

注意：评论中不支持 HTML 语法

有新评论时提醒我剩余 1000 字符

发布

在检索评论时出错，请稍后刷新。