IBM Software Group

IBM 自动化测试框架

李剑波
IBM Rational技术顾问
lijianbo@cn.ibm.com

Rational. software

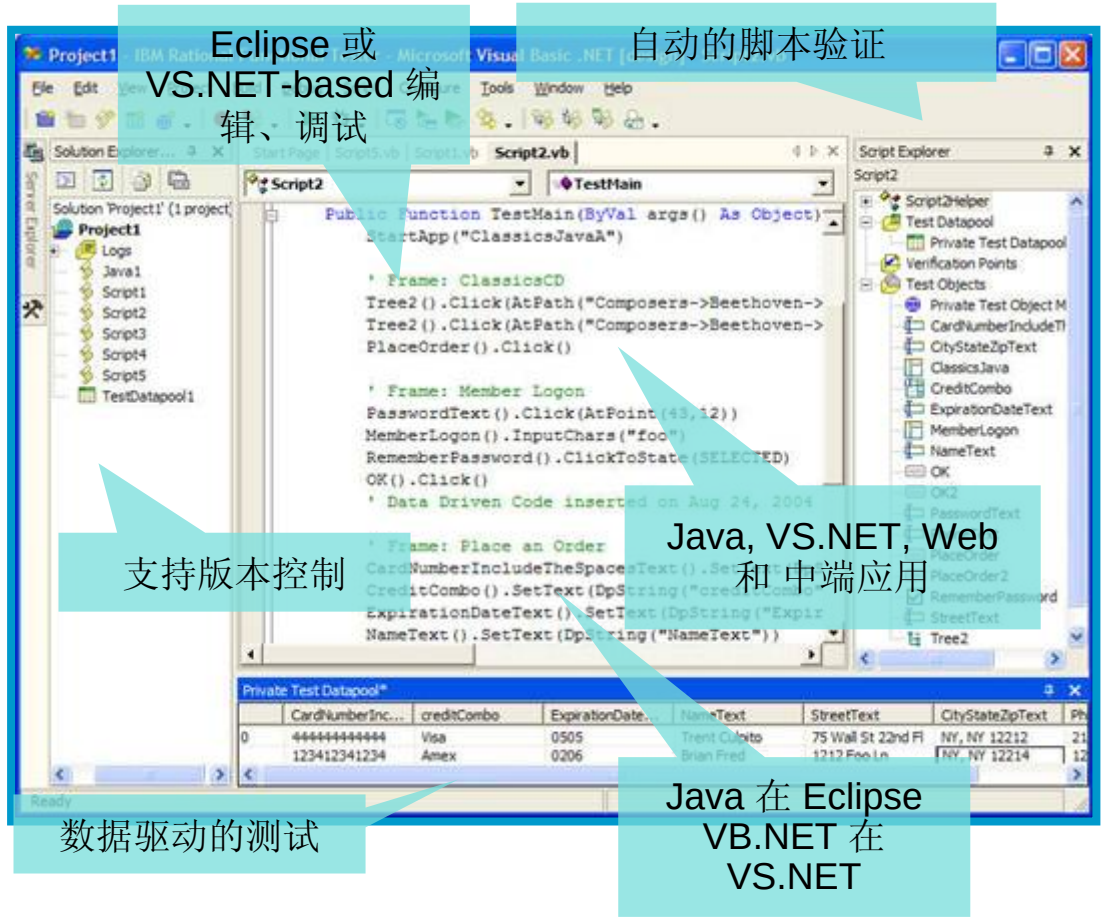
e business on demand software

议程

- 什么是IBM自动化测试框架
- IBM测试框架的架构和优点
- 利用ibm包改进自动化测试
- 如何应用IBM自动化测试框架

RFT Rational自动化测试平台

- 集成的功能测试和回归测试
- Objectmap保留测试对象
- 采用Eclipse的插件机制
- 测试Java, VS.NET WinForms, Web, Siebel 和终端应用
- 灵活的测试向导支持
- Java或者.net



用RFT进行自动化测试的通常途径

- 等待一个可用的构建……
- 耐心等待……
- 得到第一个构建，录制脚本，插入验证点(VPs)
- 回放脚本
- 如果新的版本有明显的改动，重新改进相关的脚本，也许要重新录制相关的脚本

这种简单的录制—回放模式的优点

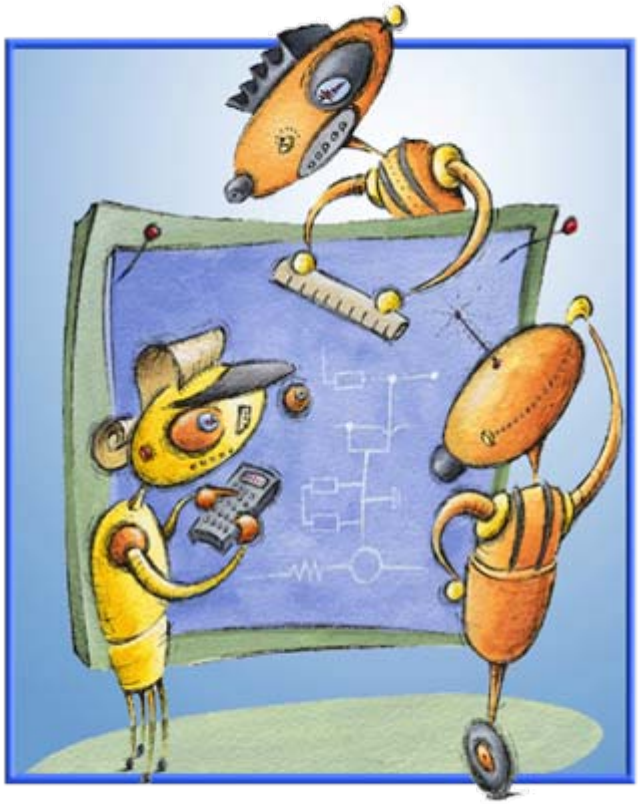
- 很容易使用... 不需要额外的编程经验
- 简单...需要设计的工作很少
- 直观的一步一步的执行方式

这种简单方式将会遇到的挑战

- 界面对象、执行步骤和验证点掺杂在一个脚本中
  → *代码不能很好重用！*
- 脚本很长并且按顺序录制
  → *很难调试!*
- 要录制脚本必须等到有可以执行的构建
  → *延迟产品可以发布的时间！*
- 测试脚本和被测系统紧密关联
  → *代码不容易维护!*

如何应对这种复杂的情况

IBM自动化测试框架 (由 QSE维护)!



首先，我们看看IBM自动化测试框架可以提供…

然后，我们看一下这个测试框架怎么帮助我们…

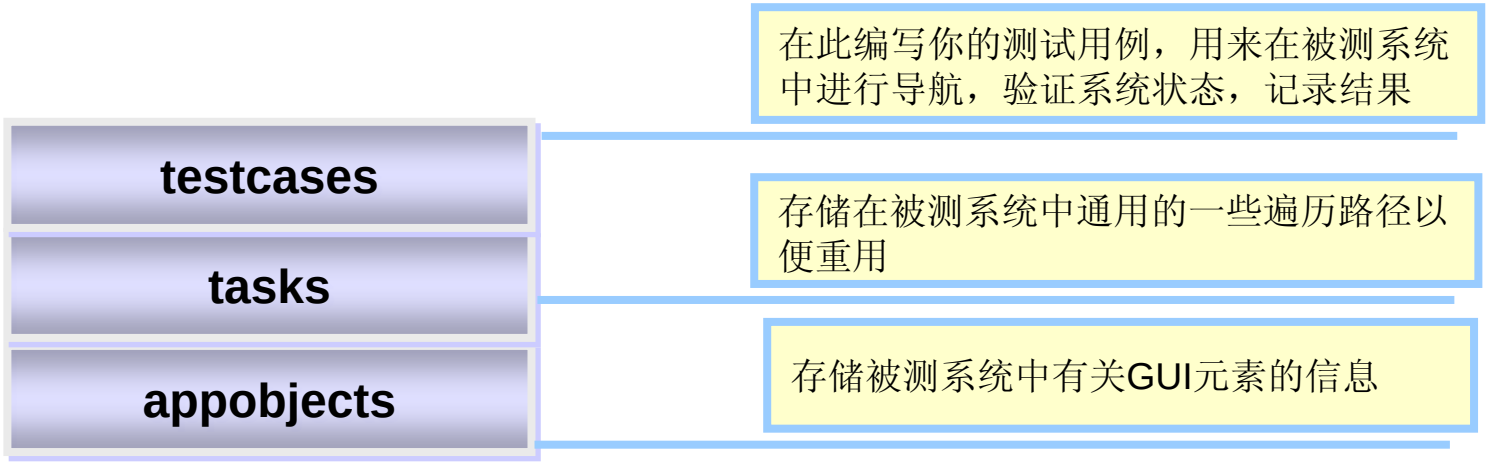
IBM自动化测试框架是?

- IBM自动化测试框架是
  - IBM自动化测试框架，以前叫ITCL(IBM Test Community Leadership)框架，是由QSE (Quality Software Engineering)小组协同IBM内部资深的自动化测试专家开发并维护的
  - 它是用RFT进行GUI自动化测试的理论指导和最佳实践

- IBM自动化测试框架提供
  - 由appobjects, tasks和testcases组成的3层架构的实现
  - IBM类包
  - IBM最佳实践
  - 一个集成的IDE

3层架构

由appobjects, tasks和testcases组成的3层架构的实现



分层的架构可以分离“做什么”和“怎么做”

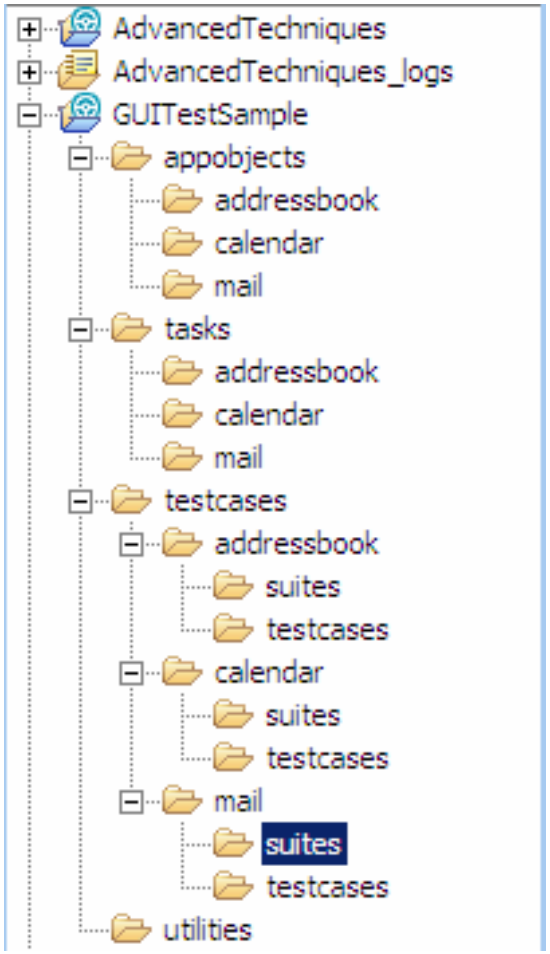
3层架构的好处？

- 代码重用…
- 调试…
- 测试脚本的维护 …
- 团队之间的协作…

如何使这个方法更可用？

- 如何基于这个3层架构来鉴别和组织工件
- Appobjects层中怎么组织GUI元素
- 找出并编写tasks
- 基于tasks编写系统特定的测试用例

如何组织类包



Testcases层可以看作是你要测试“做什么”的一个高层的视图

Tasks和appobjects层为testcases层提供必要的实现细节—测试“怎么做”

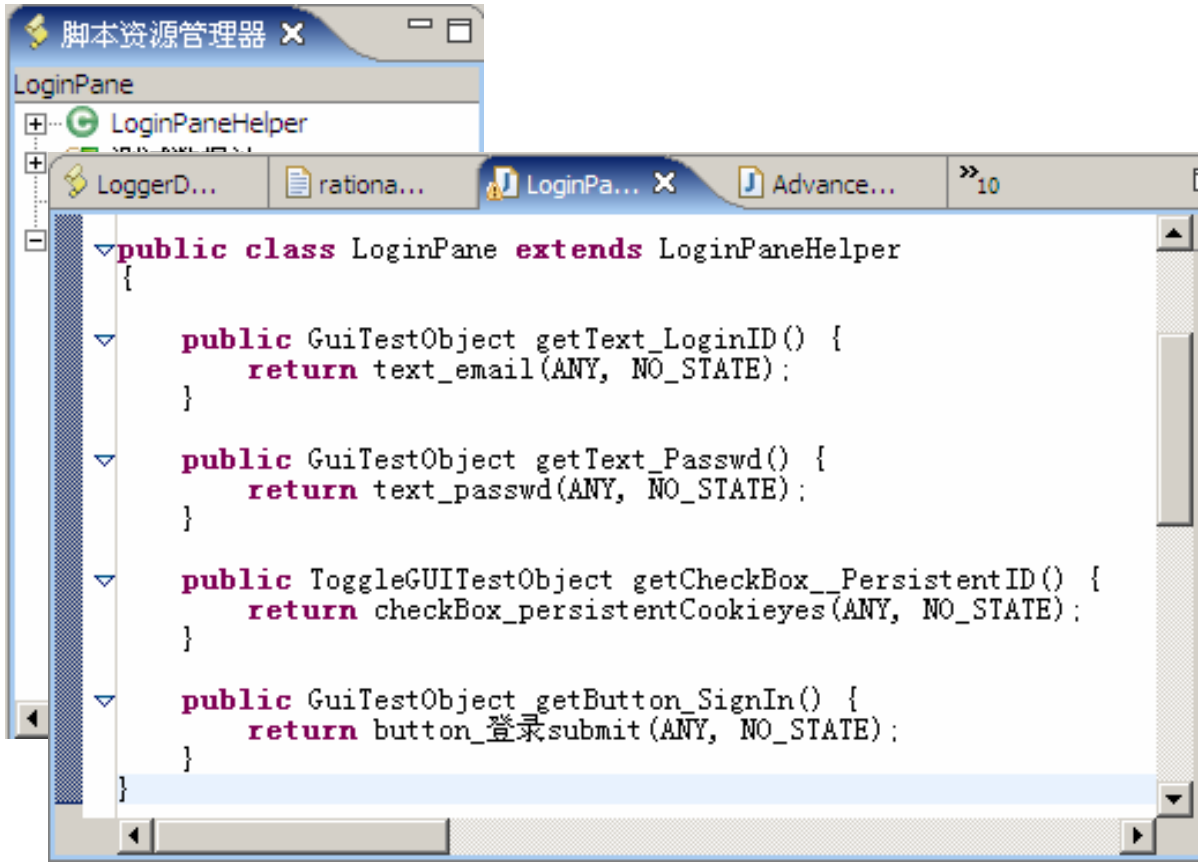
在appobjects 中组织GUI对象

- 大部分的GUI对象信息都存储在Object Map中
- Object maps 的主要目的是在被调用时返回控件对象，它被存储在脚本中(test script)。每一个脚本都包含一个特定的Object Map
- Appobjects包存储所有的脚本，这些脚本返回被测系统中的GUI对象

要点:

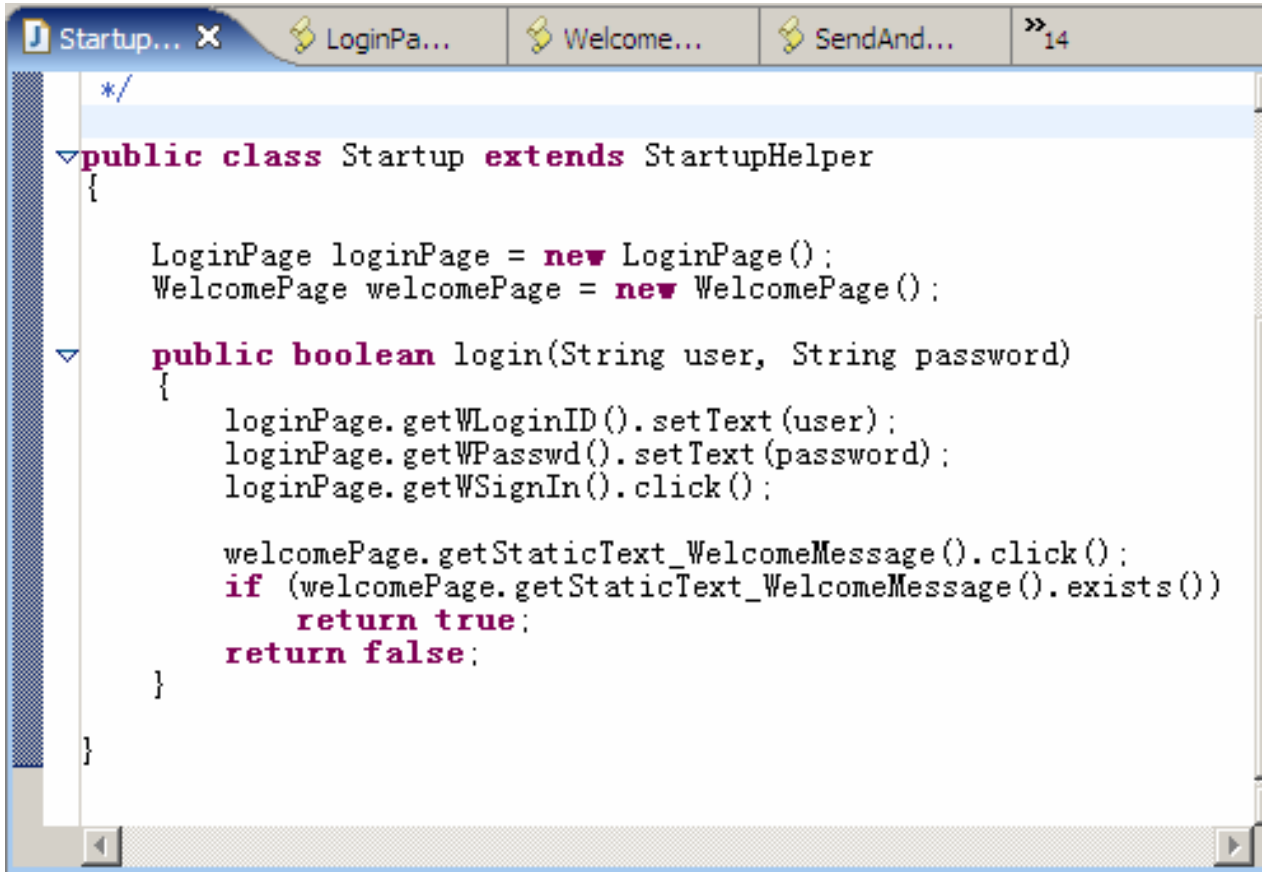
- 脚本关联GUI对象的策略
  - 对于简单的GUI界面, 一个页面或一个表单关联一个脚本
  - 对于复杂的界面，可以分解为几个部分以利于重用
- 在脚本中把对象映射到getter方法
  - 使用ibm类包中的方法可以简化对象的使用

一个例子



tasks

- Tasks 的任务是调用appobjects提供的方法来操纵GUI元素以完成一些通用的Task。反过来，tasks的方法会被testcases调用
- Tasks的力度要把握提高代码重用性以及对testcases屏蔽下层的实现细节

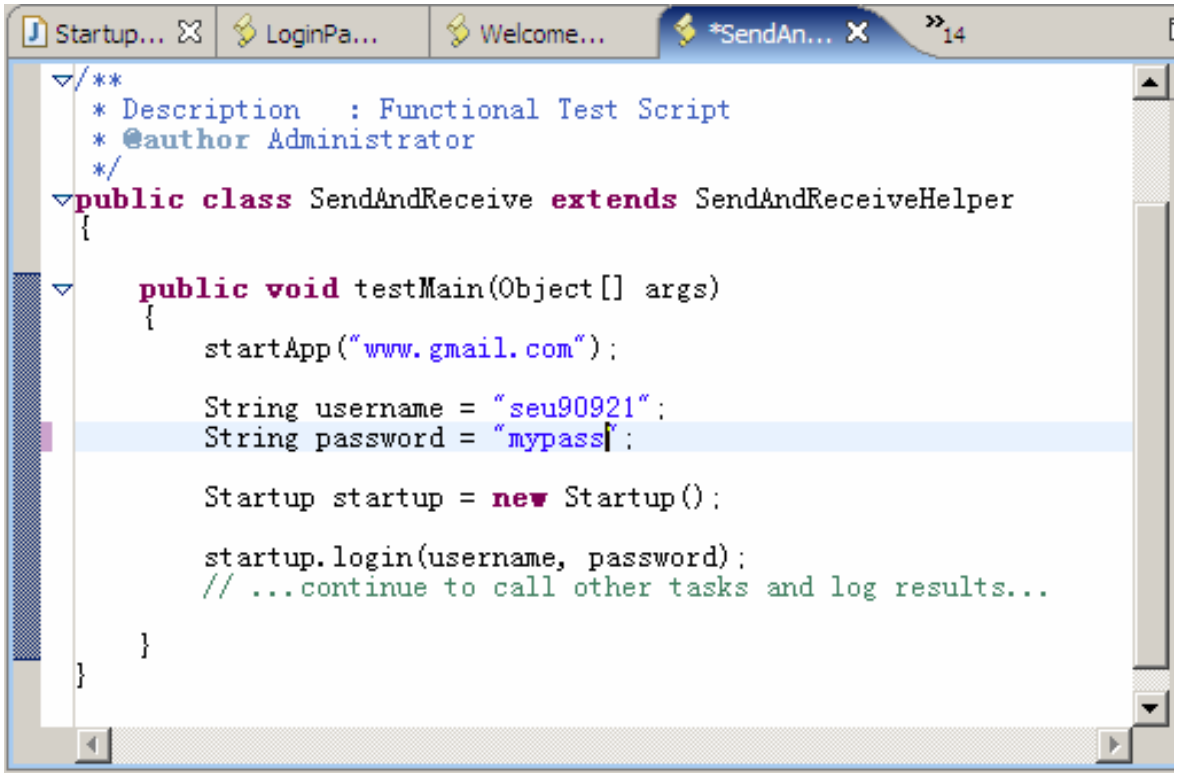


tasks (续.)

- 对抽象对象（业务对象）和具体对象（复杂的界面控件）都要遵循面向对象的思想
- Task的颗粒度的划分是确保代码重用的关键
- 通常，task会返回一个boolean值给testcase以标识task是否执行成功
- 基本上在task中不会有和应用系统相关的验证点。当一个异常出现时，task应该抛出这个异常并且由testcase这一层捕获并处理

testcases

- Testcases 调用 tasks (如果需要的话传入需要的数据), 验证条件和记录结果
- Testcase应该仅仅包含简单的逻辑和控制流，其他的都应该在tasks这个包中实现

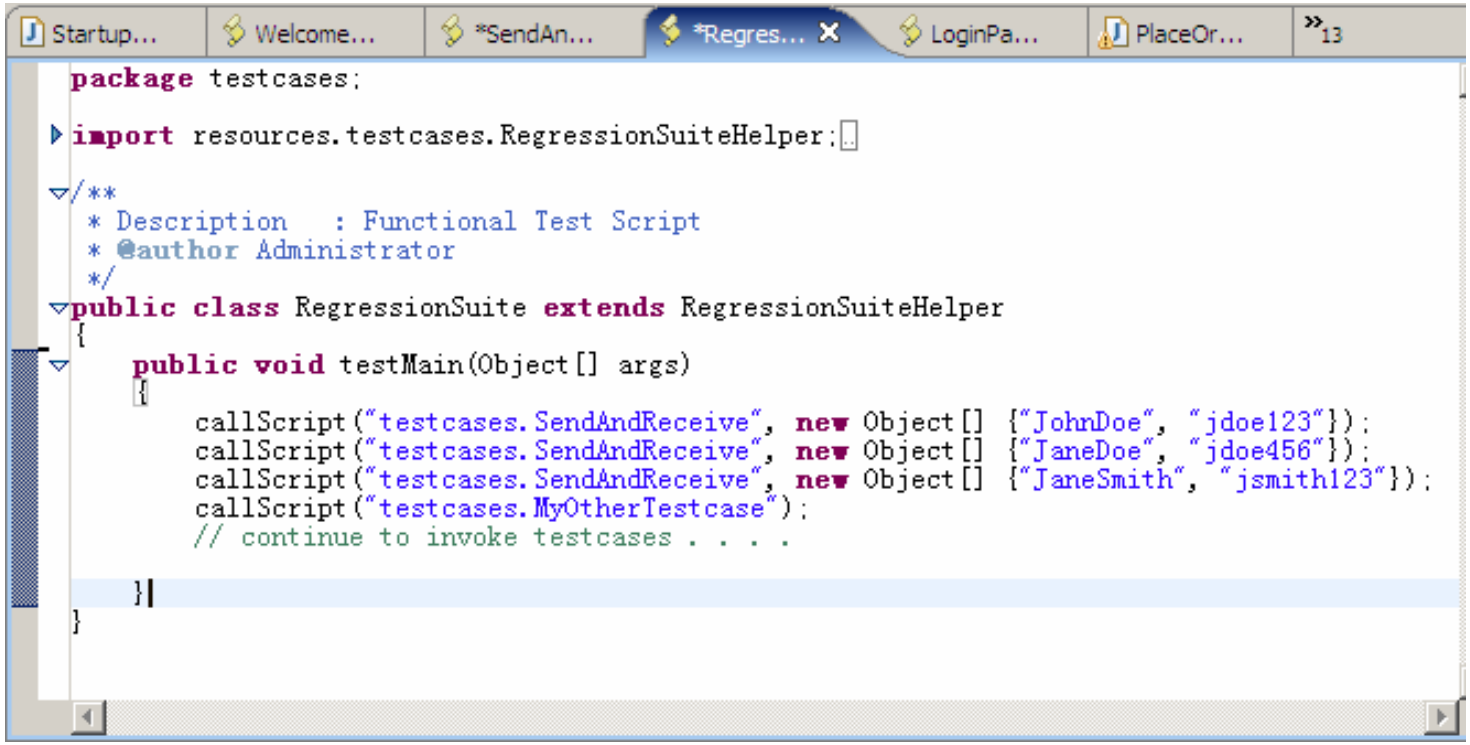


testcases(续.)

- 在testcases目录中，每一个类都应该代表某个功能区域或其他的逻辑的、直观的分组；每一个方法都应该代表一个测试用例(testcase)
- 偶尔，testcase也会绕过tasks层直接调用appobjects中的方法，但是当这种越层调用发生的太过频繁时就会使testcases变得过于复杂而需要额外的维护
- 在建立tasks这一层时多花点时间考虑和设计，将来生成testcases这层就会很快而且很容易

testsuites

- Test suite类可以通过从RationalTestScript继承过来的callScript()方法定义。Testsuite类可以放在testcases包中或在testcases中新建一个包来存放。你可以定义好testcase，然后用很多不同的数据作为输入以达到数据驱动测试的方法。



IBM 类包

- Ibm类包提供了很多类库能帮助你实现测试的自动化
- Ibm类包包含了：
  - Ibm.loggers: 提供了测试脚本可以直接调用的loggers
  - Ibm.recovery: 提供记录一个特定的状态并在需要时可以恢复到需要的位置
  - Ibm.tools: 包含了ClassGenerator类，可以用来从脚本的object map中生成所有的getter方法
  - Ibm.util: 包含了一些常用的方法方便使用RFT
  - Ibm.widgets: 扩展了TestObject功能 -- ***Most valuable!***

Ibm.widgets

- 通常，测试对象的操纵如下(例如：在一个输入框中进行输入)：

```
Text_name().click(atPoint(35,10));
Browser_htmlBrowser(document_C(),DEFAULT_FLAGS).inputKeys
("{ExtHome}+{ExtEnd}{ExtDelete}");
Browser_htmlBrowser(document_C(),DEFAULT_FLAGS).inputKeys("test");
```

如果有成千上万的地方需要这么写…

你会喜欢吗?

神啊，救救我吧！

ibm.widgets

- 为什么不把这些公用的步骤封装在一个方法中呢？

对！就是这样。但是你不必自己做，**ibm.widgets** 已经将大部分都做好了！

ibm.widgets

Class WTextField

```
java.lang.Object
  └com.rational.test.ft.object.interfaces.TestObject
      └com.rational.test.ft.object.interfaces.GuiTestObject
          └ibm.widgets.ancestors.Widget
              └ibm.widgets.WTextField
```

```
Text_name().setText("Test");
```

ibm.widget – 怎么用?

- 在appoject包中那些类提供的getter方法里构建widget对象

```
public WTextField getPasswordText()
{
    TestObject to = passwordText(ANY, NO_STATE);
    return new WTextField(to);
}
```

- 从 appobjects包的类中的getter方法得到这些widget对象以便操作:

```
WTextField passwdField = loginForm.getPasswordText();
passwdField.setText("Test");
```

ibm.widgets(续.)

- 对Java界面元素和Web界面，常用的GUI对象都已经有相应的widgets
- 对Java的swt GUI 对象, 最新的ibm包中也已经有子包中包含了响应的支持

Ibm.util

- Util这个包提供了对图象，浏览器，文件的一些常用操作。BitmapOps, BrowserOps, FileOps…etc.
- 最新的ibm包也提供了对基于终端的应用系统的支持

Ibm.util

- BitmapOps:
  - 截屏
  - 捕获一个特定的测试对象
  - 比较两个图片并找出区别
- BrowserOps:
  - 启动一个浏览器窗口
  - 关闭特定的或所有活动的浏览器窗口
  - 找到一个特定的浏览器窗口
  - 确定一个浏览器窗口是否已经完成
  - 在页面中找到一个特定的字符串
  - …

Ibm.loggers

- 如果利用ibm包中提供的日志支持，实现如下需求将会非常容易：
  - 确定输出目标：你可能想将日志写到HTML文件、文本文件、控制台或同时输出到文件和控制台
  - 自定义输出格式：你可能需要清除日志中一些额外的信息或添加一些自己的信息
  - 保存截屏：每次当自动化测试记录到一个错误时捕获并保存截屏
  - 实现一个被动日志(passive logger)：记录所有的动作和过程以减少调试时间
  - 实现测试用例追踪：查看当前多少测试用例通过了；多少失败了

Ibm.recovery

- 想象一下这个场景:
  - 100个测试用例需要一个一个的执行
  - 执行到第三个时，意外发生...这个测试用例执行失败
  - 由于没有必要的清除步骤，第四个测试用例由于运行环境问题而失败
  - 同样的原因，的五个测试用例失败....

- 必须要有步骤来确保在执行下一个测试用例的开始环境是“干净”的

Ibm.recovery (续.)

- 首先，根据应用系统特定的环境要求，定义并实现一个基状态：
  - 我需要多少个不同的基状态？
  - 基状态需要一些输入参数吗？
  - 我要接受不通的输入参数来重载基状态方法吗？

Ibm.recovery (续.)

- 调用callBaseState方法转到定义好的基状态。根据你的被测系统，可以在一个测试用例开始时调用，也可以在结束时调用

```
public boolean login(String sUserName, String sPassword) {

    //set the default user name and password here,
    gsUserName = sUserName;
    gsPassword = sPassword;

    new WebBaseState("login.yahoo.com").callBaseState();

    LoginPage loginPage = new LoginPage();

    WTextField tfPassword = loginPage.getText_password();

    if (!tfPassword.exists()) {
        logError("Could not Login! Cannot continue.");
        return false;
     }
```

Ibm.tool

- 目前, 在这个包中只有一个类 (ClassGenerator),它可以用来帮助在appobjects中生成getter方法

```
Vector v = new Vector();

v.add(new PlaceOrderForm());
v.add(new OrderCfmBox());
v.add(new FailOrderBox());
v.add(new AppForm());
v.add(new LoginForm());

new ClassGenerator().updateScripts(v);
```

如果按照IBM自动化测试框架开始一个项目？

- 回答以下问题以便找出答案：
  - 谁来实现自动化? *- 人员*
  - 什么需要自动化? *- 目标*
  - 何时进行自动化? *- 开始点*
  - 如何实现自动化? *- 方法*

谁来实现自动化？

- Java经验
  - 必须 了解基本的JAVA技术: class, object, method, package, static.
  - 必须 了解异常处理以及在车是中如何使用异常处理
  - 如果知道 interface, 会更好.

- RFT经验
  - 每个人 必须了解

什么需要自动化?

- 不是所有的功能都必须自动化
- GUI测试自动化的秘密
  - 执行自动化测试只会暴露很少的bugs
    - 执行自动化回归测试来验证系统状态，不是用来找出很多的bugs
    - 执行自动化测试可以让我们抽出更多的时间来做手工测试并发现缺陷
  - 编写自动化测试过程会帮助找到大部分的bugs
    - 会发现通过手工测试没有发现的bugs
    - 记录下每一个找到的bug — 不要等以后再记录

- 哪些需要被自动化?
  - 最佳标准：可以节省时间
  - 回归测试是最需要被自动化的

何时进行自动化?

- 最有效的方法:
  - 在第一个构建版本出来前
    - 为整个团队创建包结构
    - 实现日志策略以及编写通用的异常处理策略
    - Tasks: 根据功能和界面说明(Spec)说明(spec)并在有条件时编写Task的伪代码
    - 常用方法: 为团队指出、说明常用方法
    - Testcases:说明(spec)并在有条件时编写伪代码
  - 当第一个构建版本出来时
    - 编写appobjects方法，并查看其完备性
    - 完成tasks的编写，然后编写测试用例 ***测试用例应该很快完成!***

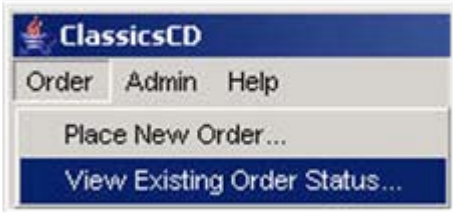
如何实现自动化?

- 没有清晰定义的手工测试用例时不要编写代码
  - 一步一步的做
  - 不要含糊不清
  - 区分开导航步骤和验证点
- 用如下注释的伪代码编写tasks和testcases，然后一行一行的填写代码

```
public void hideShowFoldersLink () {
                        // Delete all folders
                        // add folders a, b, c (3-level nested)
                        // open folder c
                        // verify clicking hide folders causes show folders link to appear
                        // verify that the tree view control is not visible
                        // verify clicking show folders causes hide folders link to appear
                        // click hide folders
            (etc.)
```

如何实现自动化?(续.)

- 在编写代码时可以利用录制－回放的功能
- 通过这个方法可以找到自动化一个控件的替代方法




```
jmb().click(atPath("Order->View Existing Order Status"));
```

如何实现自动化?(续.)

- 对appobjects和tasks包中的类进行单元测试可以在整体上节省很多时间
- 在一个新的构建版本出来时，对appobjects进行单元测试可以节省更多的时间
- 单元测试可以很简单，只要在每个类的下面添加testMain()方法，然后给每个对象传入一个消息

```
public void testMain (Object[] args) {
        // To invoke this page: start the app and make sure the default
        // Album tab is selected.
        try {
                getButton_PlaceOrder().getParent();
        } catch(Exception e) {
                System.out.println("getButton_PlaceOrder(): "+e.getClass());
        }
        try {
                getLabel_AlbumComposer().getParent();
        } catch(Exception e) {
                System.out.println("getLabel_AlbumComposer(): "+e.getClass());
        }
        (etc.)
```

问题?

.....

...

..

.