WinRunner 7.6 Tutorial

Oldsidney 学习笔记
http://home.kimo.com.tw/oldsidney

Table of Contents

Welcome to the WinRunner Tutorial 4

1. WinRunner 简介 5
   - 1.1 自动测试的好处 5
   - 1.2 了解 WinRunner 的测试流程 5
   - 1.3 熟悉 WinRunner 的使用者接口 6
     - 1.3.1 执行 WinRunner 6
     - 1.3.2 档案（File）工具列 7
     - 1.3.3 测试（Test）工具列 7
     - 1.3.4 除错（Debug）工具列 8
     - 1.3.5 使用者（User）工具列 8

2. 设定 GUI Map 10
   - 2.1 了解 WinRunner 如何识别应用程序中的 GUI 对象 10
   - 2.2 用 GUI Spy 查看 GUI 对象的属性 10
   - 2.3 选择 GUI Map 模式 13
     - 2.3.1 GUI Map File per Test 13
     - 2.3.2 Global GUI Map File 13
     - 2.3.3 设定要使用的 GUI Map File 模式 14
   - 2.4 使用 RapidTest Script Wizard 14

3. 录制测试脚本 20
   - 3.1 选择录制模式 20
     - 3.1.1 Context Sensitive 20
     - 3.1.2 Analog 20
   - 3.2 录制 Context Sensitive 模式的测试脚本 21
   - 3.3 了解测试脚本 22
   - 3.4 录制 Analog 模式的测试脚本 23
   - 3.5 执行测试脚本 25
   - 3.6 分析测试结果 26
   - 3.7 录制时的建议 28

4. 同步点（Synchronize） 30
   - 4.1 何时该使用同步点 30
   - 4.2 录制测试脚本 32
   - 4.3 变更预设等待时间的设定 34
   - 4.4 如何识别何种问题需要以同步点解决 34
   - 4.5 加入同步点 35
   - 4.6 执行测试脚本并检视结果 36

5. GUI 对象检查点（Checkpoint） 38
   - 5.1 如何检查 GUI 对象 38
   - 5.2 建立 GUI 对象检查点 39
   - 5.3 执行测试脚本 41
   - 5.4 在另一个版本的 Flight Reservation 执行测试脚本 44
   - 5.5 建立 GUI 对象检查点时的建议 46

6. 图像检查点 47
6.1 如何检查应用程序的图像 47
6.2 建立图像检查点 48
6.3 检视预期结果 49
6.4 在另一个版本的 Flight Reservation 执行测试脚本 50
6.5 建立图像检查点时的建议 51

7. 使用 TSL 撰写测试脚本 53
7.1 录制基本测试脚本 53
7.2 使用函数产生器（Function Generator）在测试脚本中插入函数 54
7.3 在测试脚本中加入判断式 55
7.4 了解 tl_step 函数 56
7.5 测试脚本的除错 56
7.6 在另一个版本的 Flight Reservation 执行测试脚本 57

8. 建立数据驱动（Data-Driven）测试脚本 59
8.1 如何建立数据驱动（Data-Driven）测试脚本 59
8.2 将测试脚本转成数据驱动（Data-Driven）测试脚本 59
8.3 将数据加入数据表 63
8.4 以 regular expression 调整测试脚本 64
8.5 修改结果信息 65
8.6 执行测试脚本并分析结果 66
8.7 建立数据驱动脚本时的建议 67

9. 文字检查点（Text checkpoint） 69
9.1 从应用程序读取文字 69
9.2 检查文字 73
9.3 除错 74
9.4 在另一个版本的 Flight Reservation 执行测试脚本 74
9.5 建立文字检查点时的建议 76

10. 建立批次（batch）测试 77
10.1 何谓批次（batch）测试 77
10.2 建立批次测试 77
10.3 在另一个版本的 Flight Reservation 执行批次测试 79
10.4 检视批次测试的结果 79
10.5 建立批次测试脚本时的建议 82

11. 维护你的测试脚本 83
11.1 当使用者接口改变时 83
11.2 在 GUI Map 中编辑 GUI 对象的属性 84
11.3 新增 GUI 物件到 GUI Map 87
11.4 使用执行精灵（Run wizard）自动更新 GUI Map 88

12. 从这里出发 92
12.1 获得更多信息 92

Welcome to the WinRunner Tutorial

欢迎使用 WinRunner 快速入门手册，本手册将引导你学习如何使用 WinRunner 建立自动化测试。

本手册分成 12 个课程，在每一个课程中你会以 WinRunner 内附的范例程序-Flight Reservation 作为练习的对象，建立并执行自动化测试脚本（script）。

Flight Reservation 范例程序有二种版本：Flight 4A、Flight 4B，Flight 4A 是可正常执行的版本，而 Flight 4B 则是内藏一些 defect。

最后希望你完成此手册教学之后，可以对 WinRunner 有基本的认识，并将你所学习到的技巧，应用到你的测试工作中。

1. WinRunner 简介

课程摘要：

- 说明自动测试的好处
- 介绍 WinRunner 的测试流程
- 介绍 WinRunner 的使用者接口

1.1 自动测试的好处

假如你执行过人工测试，你一定了解人工测试的缺点，人工测试非常无聊而且浪费时间与人力。使用人工测试的结果，往往是在应用程序交付前都无法对应用程序的所有功能都作过完整测试。

使用 WinRunner 可以加速整个测试的过程，并且在建置（build）新版本后，重复使用测试脚本进行测试。以 WinRunner 执行测试，就与人工测试一样，WinRunner 会仿真鼠标的动作与键盘的输入，不过 WinRunner 比人工测试快多了。

使用 WinRunner 你可以实现白天建置（Daily build）夜晚测试（Night test）。

自动测试的好处	
快速 （Fast）	WinRunner 执行测试比人工测试速度快多了。
可靠 （Reliable）	WinRunner 每一次的测试都可以正确的执行相同的动作，可以避免人工测试的错误。
可重复 （Repeatable）	WinRunner 可以重复执行相同的测试。
可程序化 （Programmable）	WinRunner 可以程序的方式，撰写复杂的测试脚本，以带出隐藏在应用程序中的信息。
广泛的 （Comprehensive）	WinRunner 可以建立广泛的测试脚本，涵盖应用程序的所有功能。
可再使用 （Reusable）	WinRunner 可以重复使用测试脚本，即使应用程序的使用接口已经改变。

1.2 了解 WinRunner 的测试流程

WinRunner 的测试流程包含六个阶段：

1. 识别应用程序的 GUI 对象
2. 建立测试脚本
3. 对测试脚本除错（debug）
4. 在新版应用程序执行测试脚本

5. 检视测试结果
6. 回报缺陷（defect）

1.3 熟悉 WinRunner 的使用者接口

在开始使用 WinRunner 建立测试脚本前，你应该先熟悉 WinRunner 的使用者接口。

在开始执行 WinRunner 之前，请先确认屏幕的设定至少要 256 色以上。

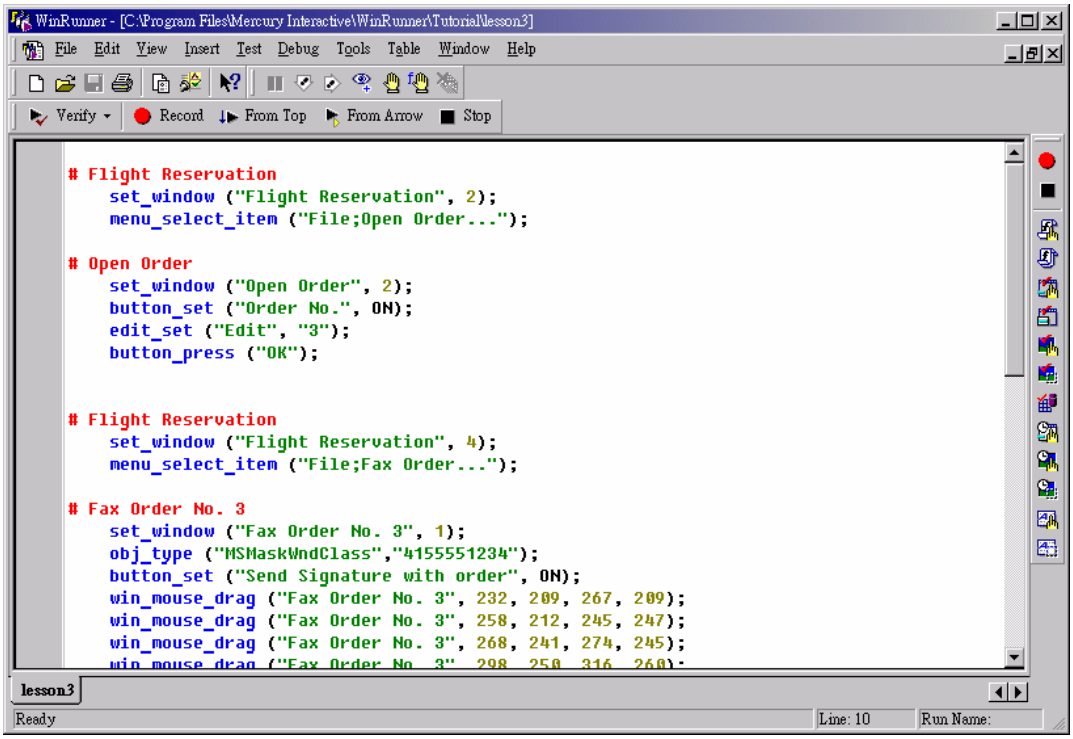
1.3.1 执行 WinRunner

执行【开始】->【程序集】->【WinRunner】->【WinRunner】，首先开启 WinRunner Add-in Manager 窗口。

如果是第一次执行 WinRunner，会开启欢迎窗口以及「What's New in WinRunner」的说明文件。

WinRunner Add-in Manager 显示你目前可以使用的 Add-in，本快速入门手册中的课程并没有使用任何 Add-in，所以请确认没有勾选任何 Add-in 后按下【OK】按钮。

以下是 WinRunner 的画面：



1.3.2 *档案（File）工具列*

提供常用的按钮，如新增、开启、储存等。

 ：New（Ctrl+N），建立新的测试脚本

 ：Open（Ctrl+O），开启旧的测试脚本

 ：Save（Ctrl+S），储存测试脚本

 ：Print（Ctrl+P），打印

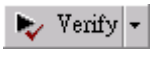
 ：Test Properties，设定测试脚本的属性

 ：Test Results，检视测试脚本的测试结果

 ：Help（Shift+F1），说明文件

1.3.3 *测试（Test）工具列*

提供常用的按钮，如录制、执行、停止等



：Run Mode，设定执行模式，有 Verify、Debug、Update 三种执行模式

：Click to Record in Context Sensitive，开始以 Context Sensitive 模式录制

：Run from Top，从头开始执行

：Run from Arrow，从黄色小箭头处开始执行

：Stop，停止录制或执行

1.3.4 *除错（Debug）工具列*

提供除错时常用的按钮，如逐步执行、暂停、设定断点等。


：Pause，暂停


：Step，逐步执行


：Step Into，逐步执行并进入


：Add Watch，新增监视变数


：Toggle Breakpoint（F9），设置断点


：Break in Function（Ctrl+B），设置函数中的专断点


：Delete All Breakpoints，清除所有断点

1.3.5 *使用者（User）工具列*

提供让使用者自订常用的按钮。


：Click to Record in Context Sensitive，开始以 Context Sensitive 模式录制


：Stop，停止录制或执行


：Insert Function for Object/Window


：Insert Function from Function Generator


：GUI Checkpoint for Object/Window


：GUI Checkpoint for Multiple Objects


：Bitmap Checkpoint for Object/Window


：Bitmap Checkpoint for Screen Area


：Default Database Checkpoint


：Synchronization Point for Object/Window Property

 ：Synchronization Point for Object/Window Bitmap

 ：Synchronization Point for Screen Area Bitmap

 ：Get Text from Object/Window

 ：Get Text from Screen Area

2. 设定 GUI Map

课程摘要：

- 解释 WinRunner 如何识别应用程序中的 GUI 对象
- 示范如何使用 GUI Spy 查看 GUI 对象的属性（properties）
- 说明二种 GUI Map 模式
- 说明如何使用 RapidTest Script Wizard 学习对象并产生测试脚本（script）

2.1 了解 WinRunner 如何识别应用程序中的 GUI 对象

一般的 Windows 应用程序，通常是由窗口、按钮、list、菜单等所组成，在 WinRunner 这些窗口、按钮等通称为 GUI 对象（GUI object）。

WinRunner 会透过这些 GUI 对象的属性（physical properties），如 class、label、width、height、handle 与 enabled 等，来识别 GUI 对象。WinRunner 只会纪录最少但可组合成唯一的属性来辨识 GUI 对象。

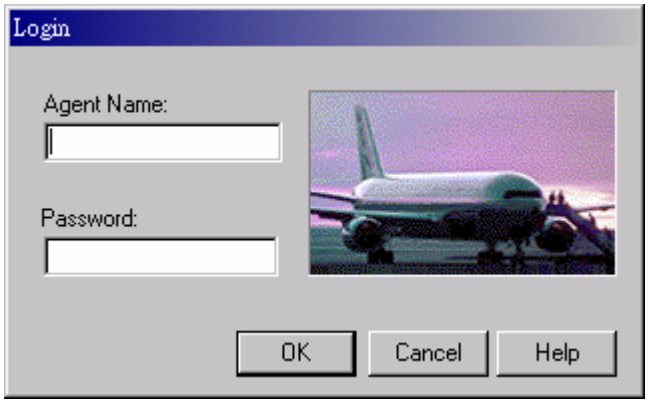
例如，当 WinRunner 识别一个 OK 按钮时，会记录这个按钮所属的窗口（如属于 Open 窗口中的 GUI 对象）、隶属的 class（如 push_button）、按钮的文字卷标（如 OK）来识别此按钮，至于如 width、height、handle 或其它属性就不会被使用到。

2.2 用 GUI Spy 查看 GUI 对象的属性

WinRunner 提供一个工具叫 GUI Spy，可以用来检视某个 GUI 对象有哪些属性以及 WinRunner 是以哪些属性来识别此 GUI 对象的。

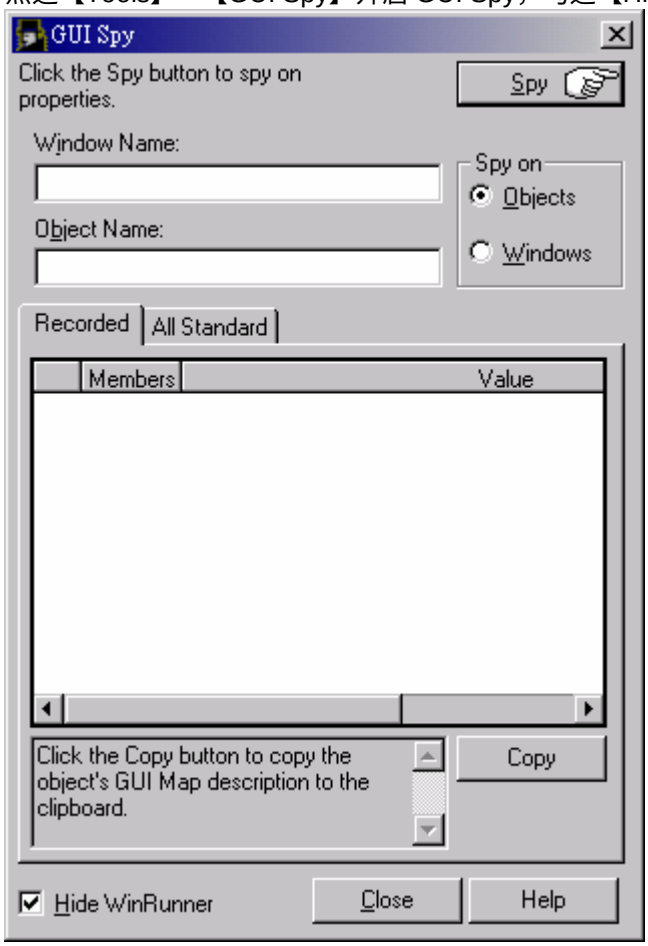
以下将示范以 GUI Spy 检视 Flight Reservation 范例程序登入窗口的 GUI 对象。

1. 开启 Flight Reservation 范例程序
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。

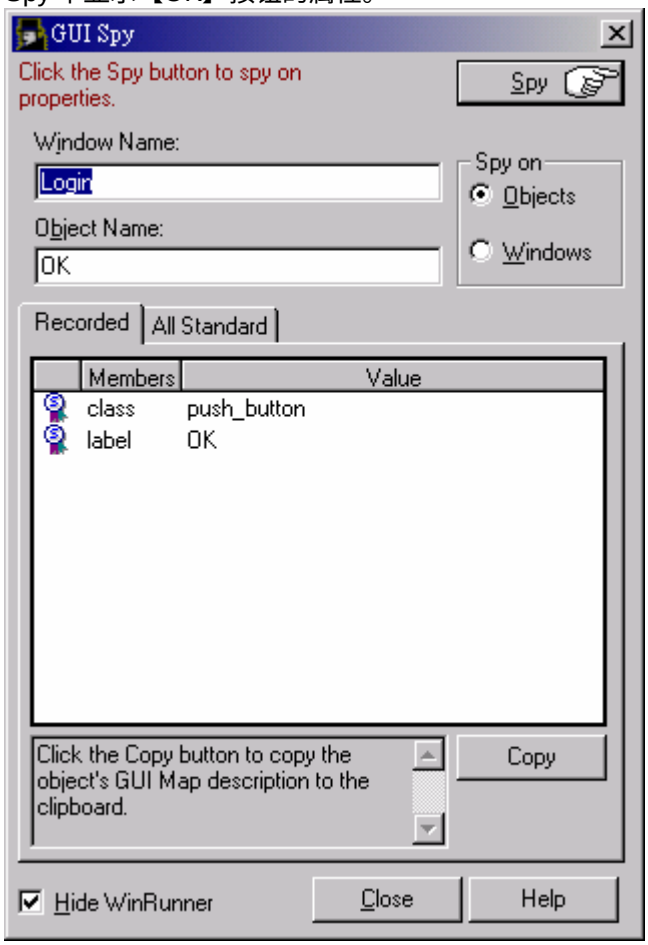


2. 开启 WinRunner
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。

3. 开启 GUI Spy
   点选【Tools】->【GUI Spy】开启 GUI Spy，勾选【Hide WinRunner】。



4. 检视 WinRunner 用来识别【OK】按钮的属性
   在 GUI Spy 按下【Spy】按钮，WinRunner 会缩到最小，这时你可以看到 Flight Reservation 的登入窗口，将鼠标移动到登入窗口上，这时你可以看到被鼠标指到的 GUI 对象会有个外框在闪动，同时 GUI Spy 也会显示此 GUI 对象的属性。
   将鼠标移到【OK】按钮上，然后按下左边的【Ctrl+F3】，会跳出 Spy 模式，你可以看到 GUI Spy 中显示【OK】按钮的属性。



5. 检视 GUI Spy 显示的信息
   在 GUI Spy 最上面显示了这个【OK】按钮所隶属的窗口是 Login 窗口，且此【OK】按钮的 logic name 为 OK。
   在【Recorded】页签，则是显示 WinRunner 用来识别【OK】按钮的属性，分别是 class：push_button 以及 label：Ok，表示这个 GUI 对象是个按钮，按钮上面的文字是 OK。
   在【All Standard】页签，则是显示【OK】按钮的所有属性。
   在这你发现到 WinRunner 只用最少的属性来识别 GUI 对象。

6. 检视 Login 窗口上其它 GUI 对象的属性
   花一点时间，用 GUI Spy 检视一下 Login 窗口上其它 GUI 对象的属性。

7. 关闭 GUI Spy
   按下【Close】关闭 GUI Spy。

2.3 选择 GUI Map 模式

当 WinRunner 识别完 GUI 对象后，会将 GUI 对象储存在 GUI Map File，WinRunner 提供二种 GUI Map File 模式：GUI Map File per Test 与 Global GUI Map File。

因此在开始使用 WinRunner 识别 GUI 对象并执行自动测试之前，你应该先考虑要使用哪种 GUI Map 模式，是 GUI Map File per Test 还是 Global GUI Map File？

2.3.1 GUI Map File per Test

在 GUI Map File per Test 模式，当你新建立一个测试脚本（test script），WinRunner 就会自动帮你建立此测试脚本的 GUI Map File，当你储存测试脚本时，WinRunner 也会自动储存 GUI Map File，而当你开启测试脚本时，其 WinRunner 也会自动加载其 GUI Map File，总之所有与 GUI Map File 有关的动作，都由 WinRunner 自动会你处理掉了。

如果你才刚开始接触 WinRunner，可以考虑使用 GUI Map File per Test 模式，如此一来你就不需要处理 GUI Map File 的相关动作，如建立、储存与加载。

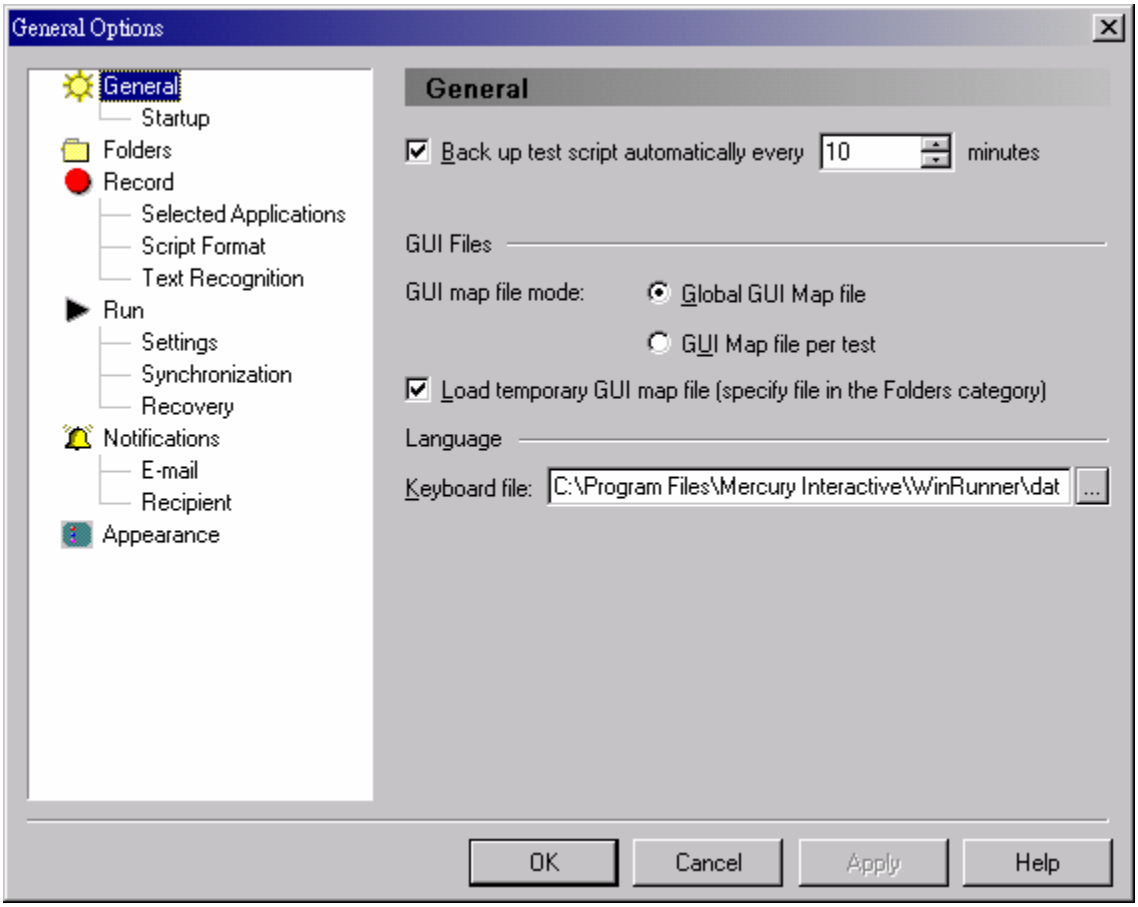
2.3.2 Global GUI Map File

在 Global GUI Map File 模式，你可以多个测试脚本共享一个 GUI Map File。另外，你还要记得储存 GUI Map File，并且在开启测试脚本时，也要同时加载使用的 GUI Map File。

如果你已经熟悉 WinRunner 的使用，可以考虑使用 Global GUI Map File 模式。

2.3.3 *设定要使用的 GUI Map File 模式*

WinRunner 默认值是使用 Global GUI Map File，要设定 GUI Map File 模式，点选【Tools】->【General Options…】->【General】->【GUI Files】，设定你要的 GUI Map File 模式，按下【OK】按钮就行了。



如果你重新设定 GUI Map File 模式后，记得要重新启动 WinRunner 让设定生效。

本课程与练习只适用于 Global GUI Map File 模式。

建议你以一种 GUI Map File 模式完成本课程，不要在练习的过程中，任意切换 GUI Map File 模式。

2.4 使用 RapidTest Script Wizard

当你选择 Global GUI Map File 模式时，可以使用 RapidTest Script Wizard 帮助你快速建立 GUI Map File。RapidTest Script Wizard 会有系统的开启应用程序中的窗口，并识别窗口中所有的 GUI 对象。接下来将利用 RapidTest Script Wizard 识别 Flight Reservation 的 GUI 对象。

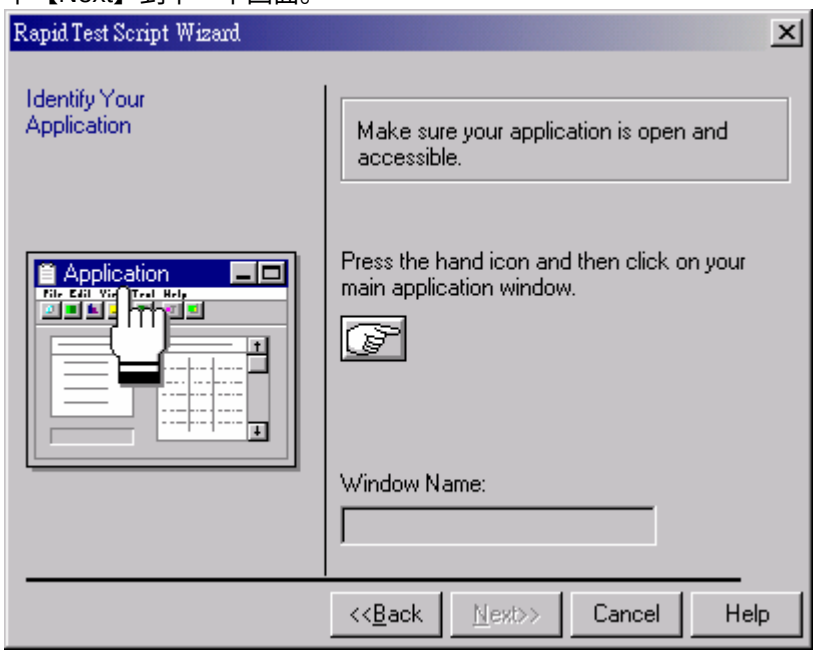
RapidTest Script Wizard 只能在 Global GUI Map File 模式下使用。

当 WinRunner 有加载 Terminal Emulator、WebTest、Java add-ins 时，RapidTest Script Wizard 也无法使用。

7. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

8. 开启 WinRunner
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。

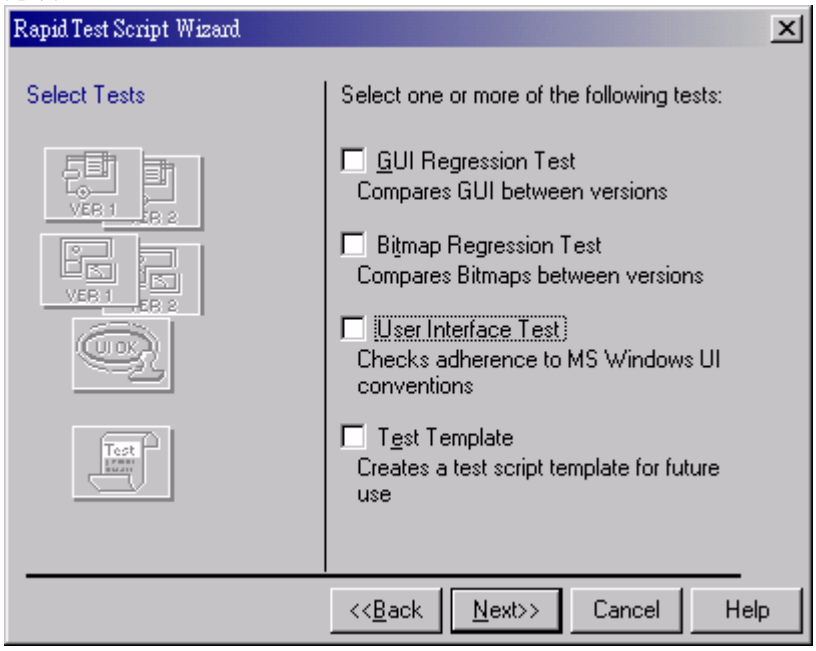
9. 开启 RapidTest Script Wizard
   点选【Insert】->【RapidTest Script Wizard…】开启 RapidTest Script Wizard 欢迎窗口，按下【Next】到下一个画面。



10. 指定要测试的应用程序
    点选 然后点选 Flight Reservation 任一位置，在【Window Name】会出现 Flight Reservation 的窗口名称，按下【Next】。

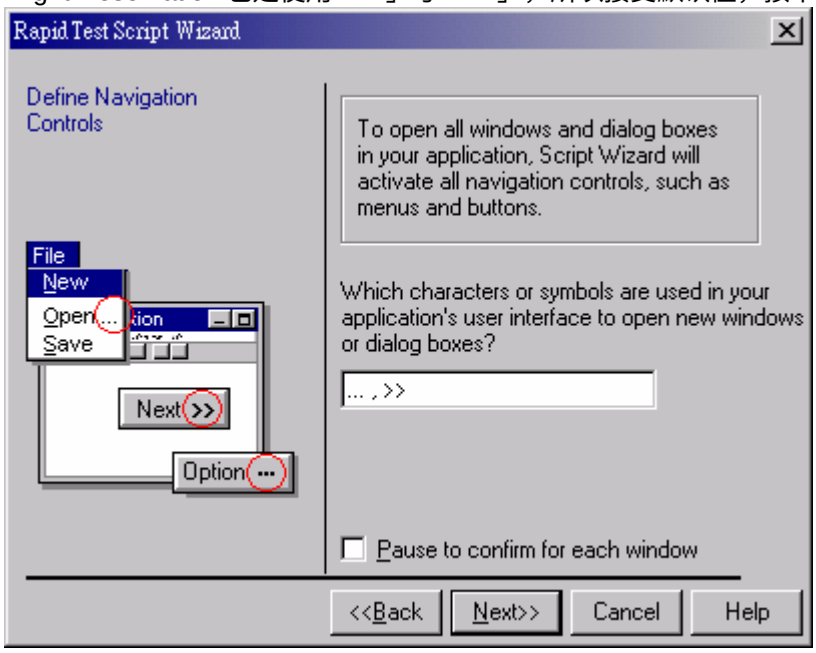
11. 清除所有设定

本练习只利用 RapidTest Script Wizard 识别 Flight Reservation 的 GUI 对象，所以要清除所有选项。



12. 接受 Navigation Controls 默认值

此窗口主要告诉 WinRunner 哪些 GUI 对象会开启一个新窗口，默认值为「...」与「>>」，而 Flight Reservation 也是使用「...」与「>>」，所以接受默认值，按下【Next】。

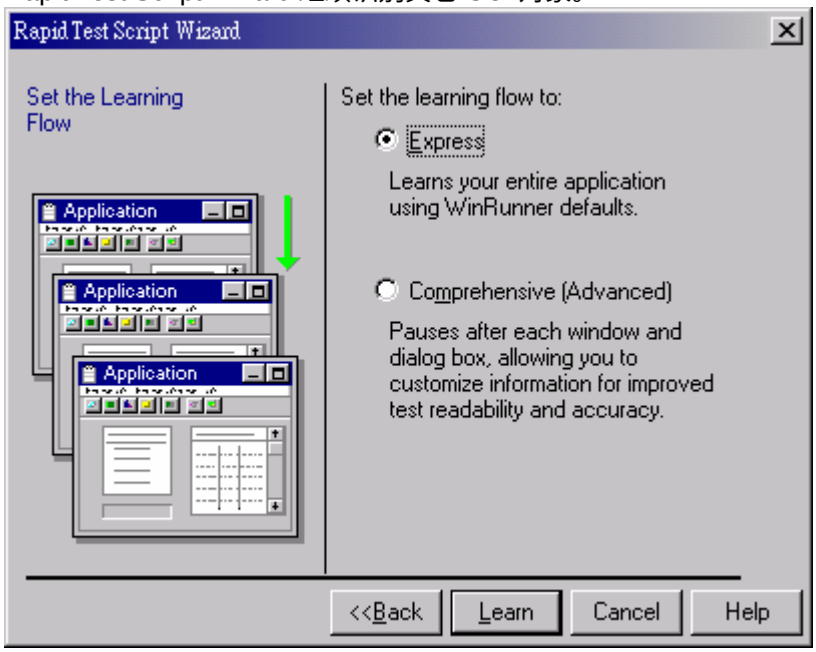


13. 设定 Learning Flow 为 Express

RapidTest Script Wizard 提供二种学习模式：Express 与 Comprehensive，本练习使用 Express 模式。

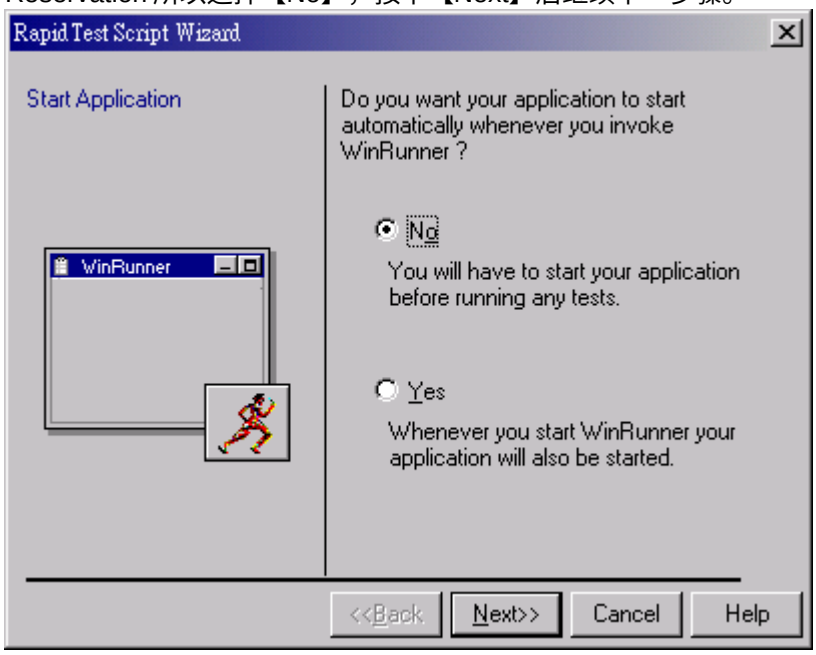
按下【Learn】按钮，你可以看到 RapidTest Script Wizard 开始识别 Flight Reservation 中所有 GUI 物件，包含下拉式菜单、开启窗口、识别窗口上所有 GUI 对象。此识别过程会花费几分钟的时间。

假如识别的过程中，跳出对话窗口通知你有 GUI 对象是 disalbed，按下【Continue】按钮让 RapidTest Script Wizard 继续识别其它 GUI 对象。



14. 在 Start Application 接受默认值 No

WinRunner 可以自动帮你执行 Flight Reservation 程序。在本课程中我们手动执行 Flight Reservation 所以选择【No】，按下【Next】后继续下一步骤。

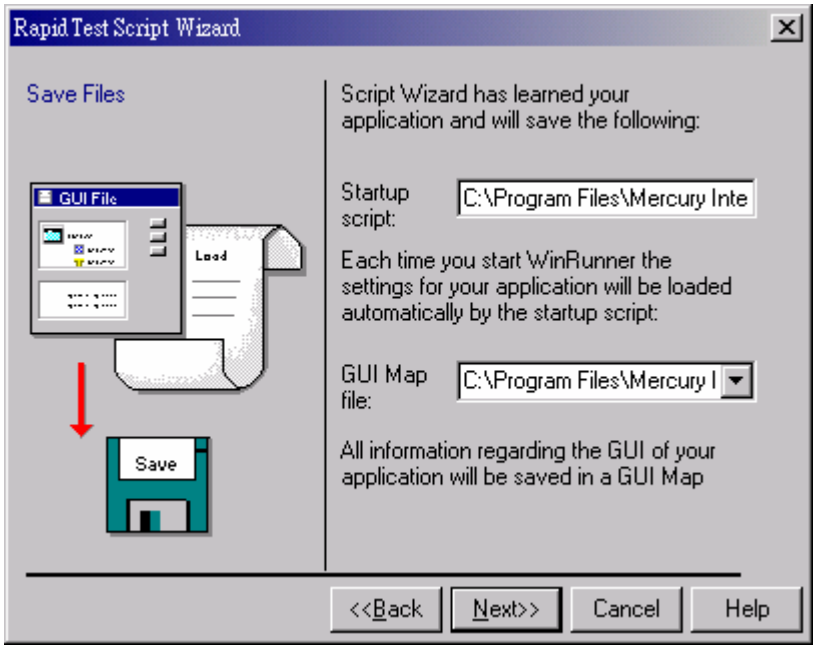


15. 储存 GUI Map File 并设定 Startup Script

在 Save Files 窗口，主要将 RapidTest Script Wizard 识别的所有 GUI 对象的信息储存在一个 GUI Map File 中。并且设定 Startup Script，则每次执行 WinRunner 时会自动执行此 Startup Script，而此 Startup Script 内只有一个指令，就是加载此 GUI Map File。

此 Startup Script 与 GUI Map File 预设储存路径在<WinRunner 安装目录>\dat\下，GUI Map File 名称为 flight4a.gui。

按下【Next】。



16. 出现 Congratulations 窗口，表示你已经完成建立 Flight Reservation 的 GUI Map File 的动作了！

3. 录制测试脚本

课程摘要：

- 说明二种录制模式：Context Sensitive 与 Analog
- 示范如何录制测试脚本
- 如何阅读测试脚本内容
- 执行测试脚本并分析其结果

3.1 选择录制模式

WinRunner 可以让你以录制的方式快速建立自动测试脚本。在录制时，使用者还是与平常一样操作应用程序，而 WinRunner 会将使用者的动作录制下来，如按下鼠标左键、键盘的输入等，并以 TSL（Test Script Language）产生测试脚本，TSL 会显示在 WinRunner 窗口中。

在录制之前，你要先选择录制的模式，WinRunner 提供二种录制模式：Context Sensitive 与 Analog。

3.1.1 Context Sensitive

Context Sensitive 录制模式主要是以 GUI 对象为基础，WinRunner 会识别使用者点选的 GUI 对象（如窗口、菜单、按钮等），以及执行的操作（如按下、移动、选取等）。

举例来说，假如你以 Context Sensitive 模式录制在 Flight Reservation 的登入窗口上按下【OK】按钮的动作，WinRunner 会产生以下的 TSL：

button_press(“OK”);

当你执行这段 TSL，WinRunner 会在应用程序上找寻【OK】按钮，然后按下它。

3.1.2 Analog

在 Analog 模式，WinRunner 主要录制鼠标移动的轨迹、鼠标的点选以及键盘的输入三种动作。

举例来说，假如你以 Analog 模式录制在 Flight Reservation 的登入窗口上按下【OK】按钮的动作，WinRunner 会产生以下的 TSL：

```
move_locator_track (1);          鼠标移动
mtype ("<T110><kLeft>-");        按下鼠标左键
mtype ("<kLeft>+");              放开鼠标左键
```

当你执行上面的 TSL，你会发现 WinRunner 会控制鼠标移动，此鼠标移动的轨迹是以屏幕的绝对坐标为基准，所以当应用程序的位置或是使用接口变动，则以 Analog 模式录制的测试脚本将会执行失败。

建议只有在测试需要记录鼠标移动的应用程序时，如绘图软件，才使用 Analog 录制模式。否则以使用 Context Sensitive 录制模式为优先。

以下的考虑可以帮助你决定使用哪种录制模式：

以 Context Sensitive 模式录制	以 Analog 模式录制
应用程序包含一般 GUI 对象	应用程序包含绘图区域
不需要录制鼠标移动的轨迹	需要录制鼠标移动的轨迹 （如绘图软件）
你打算将测试脚本运用在同一应用程序不同的版本上	

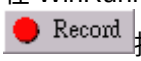
假如你测试的应用程序包含一般 GUI 对象，也包含绘图区域，你可以在录制的过程中，依需要随时切换录制模式，在后面的课程将会有详细的介绍。

3.2 录制 Context Sensitive 模式的测试脚本

接下来你会以 Context Sensitive 模式录制一段测试脚本，此测试脚本的操作流程为在 Flight Reservation 开启一笔订单。

1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。
   调整 WinRunner 与 Flight Reservation 的窗口大小与位置，让这二个窗口内容都可以清楚的倍看见。

3. 开始以 Context Sensitive 模式录制测试脚本
   在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

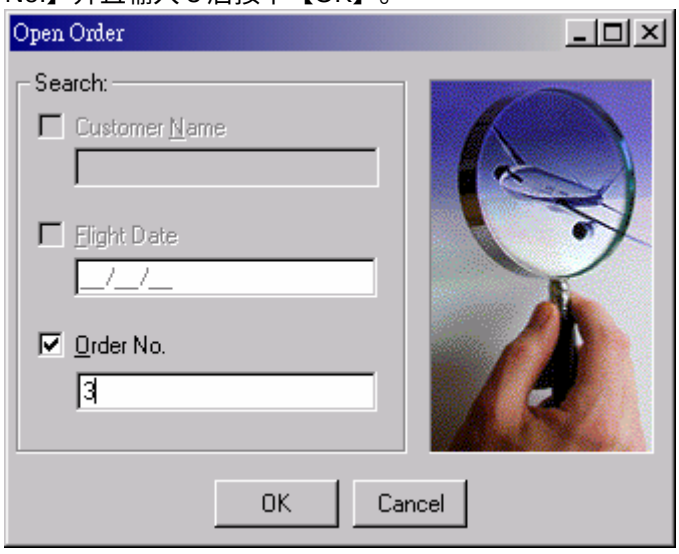
按钮，从现在开始 WinRunner 会录制所有鼠标的点选以及键盘的输入。请注意 

会变成 

，蓝色的 Rec 会出现在按钮下方，表示现在已经进入 Context Sensitive 录制模式了。在 WinRunner 下方的状态列同样也会有变化，表示现在已经在录制测试脚本了。

4. 开启 3 号订单

   在 Flight Reservation 中点选【File】->【Open Order】，在 Open Order 窗口中点选【Order No.】并且输入 3 后按下【OK】。



5. 停止录制

   在 WinRunner 中点选【Test】->【Stop Recording】，或是直接点选工具列上的 

按钮停止录制测试脚本。

6. 储存测试脚本

   点选【File】->【Save】或是直接点选工具列上的  按钮，将测试脚本储存成 lesson3。

注意 WinRunner 是以目录的方式而不是单一档案的方式储存 lesson3 测试脚本。此目录将会包含测试脚本以及测试执行的结果。

3.3 了解测试脚本

在之前的练习中，你录制了在 Flight Reservation 中开启订单的测试脚本，而 WinRunner 产生了以下的测试脚本：

```
# Flight Reservation
    set_window ("Flight Reservation", 2);
    menu_select_item ("File;Open Order...");

# Open Order
    set_window ("Open Order", 2);
    button_set ("Order No.", ON);
    edit_set ("Edit", "3");
    button_press ("OK");
```

就如你所见，WinRunner 产生的 TSL 描述了你选择的 GUI 对象以及互动的方式。例如当你点选下拉式菜单时，WinRunner 产生的 menu_select_item 的指令。

接下来将针对此测试脚本作进一步的详细说明：

- 当你点选一个 GUI 对象，WinRunner 会自动帮这个 GUI 对象取个名字，通常是以 GUI 对象上的文字做为名字，此名字称为在 WinRunner 称为 logic name。这个 logic name 可以让你更容易的阅读测试脚本。例如当你点选【Order No.】这个 check box 时，WinRunner 产生以下的指令：

```
button_set ("Order No.", ON);
```

  而 Order No. 就是这个【Order No.】check box 的 logic name。

- 当你换到另一个窗口上操作时，WinRunner 会自动在测试脚本上加上一行批注，帮助你更容易阅读测试脚本。例如当你点选 Flight Reservation 窗口时，WinRunner 会自动加上下面的注解：

```
# Flight Reservation
```

- 当你换到另一个窗口上操作时，WinRunner 会自动产生一行 set_window 指令，然后才是它操作的指令。例如当你开启 Open Order 窗口时，WinRunner 会先产生下面的指令：

```
set_window ("Open Order", 2);
```

- 当你以键盘输入时，WinRunner 会产生 type、obj_type、或是 edit_type 等指令。例如当你在 Order No. 中输入 3 时，WinRunner 会产生下面的指令：

```
edit_set ("Edit", "3");
```

3.4 录制 Analog 模式的测试脚本

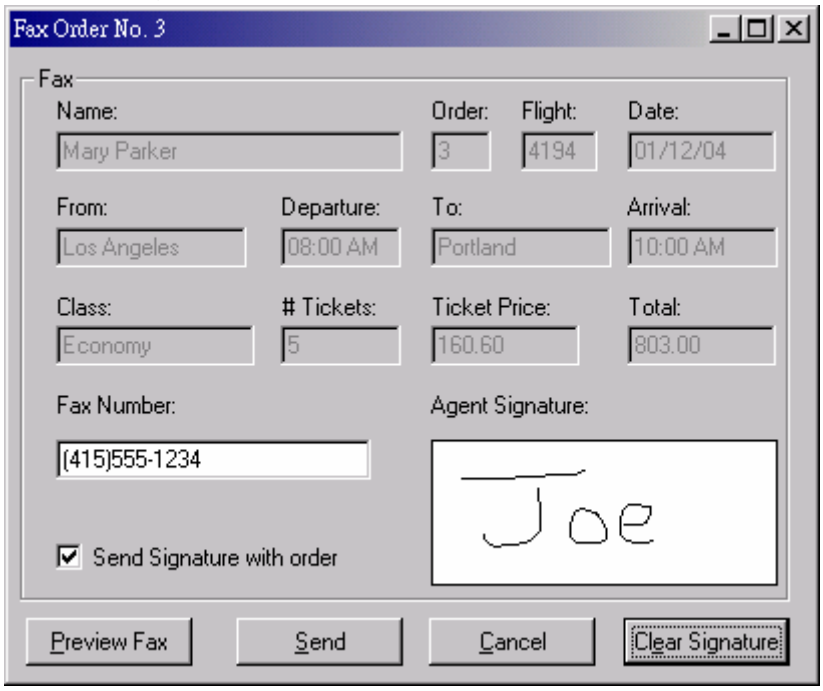
接下来你会以 Analog 模式录制一段测试脚本，此测试脚本的操作流程为在 Flight Reservation 传真一笔订单。一开始你会以 Context Sensitive 的模式录制，然后在签名的时候切换成为 Analog 的模式录制测试脚本，录制完签名的部份，再切换回 Context Sensitive 的模式。

1. 开启 lesson3 测试脚本，并将光标移到最后一行
   接下来的操作将以 lesson3 测试脚本继续录制下去，。先点选【File】->【Open】开启 lesson3 测试脚本，并且将光标移到最后一行。

2. 开始以 Context Sensitive 模式录制测试脚本
   在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

按钮。

3. 开启传真订单
   在 Flight Reservation 中点选【File】->【Fax Order】，在【Fax Number】中输入 4155551234。



4. 勾选【Send Signature with order】

5. 在 Context Sensitive 模式下录制签名动作
   以鼠标在【Agent Signature】空白区域中签名。
   这时请注意 WinRunner 如何录制你的签名动作。

6. 清除签名
   按下【Clear Signature】按钮。

7. 将 Fax Order 窗口移动到其它位置
   在切换到 Analog 模式之前，移动一下 Fax Order 窗口。

8. 在 Analog 模式下录制签名动作
   按下键盘上的【F2】或是再按一次工具列上的 Record 按钮，此时录制模式将从 Context Sensitive 切换到 Analog 模式，且 Record 会变成 Record 表示现在是 Analog 模式。
   以鼠标在【Agent Signature】空白区域中签名。
   这时请注意 WinRunner 如何录制你的签名动作。

9. 切换回 Context Sensitive 模式并将订单传真出去
   按下键盘上的【F2】或是再按一次工具列上的 Record 按钮，此时录制模式会从 Analog 模式切换回 Context Sensitive 模式。按下【Send】按钮后，Flight Reservation 会仿真将订单传真出去。

10. 停止录制

    在 WinRunner 中点选【Test】->【Stop Recording】，或是直接点选工具列上的 

按钮停止录制测试脚本。

11. 储存测试脚本

    点选【File】->【Save】或是直接点选工具列上的  按钮。

12. 如果在 Global GUI Map File 模式下，记得储存新的 GUI 对象
    在前一个课程中你已经以 RapidTest Script Wizard 识别过 Flight Reservation 的 GUI 对象，但是因为 Fax Order 这个窗口必须在开启一笔订单后才能再开启 Fax Order 窗口，因此 RapidTest Script Wizard 并未将 Fax Order 窗口的 GUI 对象也识别下来。所以当你录制到 Fax Order 窗口上的操作时，WinRunner 会识别到新的窗口与 GUI 对象，并且先放到 temporary GUI Map File 中。但是当你关闭 WinRunner 后，在 temporary GUI Map File 中的 GUI 对象将会被抛弃，所以你一定要将新识别的 GUI 对象储存下来，这个动作非常重要，一定要记得。点选【Tools】->【GUI Map Editor】，再点选【View】->【GUI Files】，你可以看到 Fax Order No. 3 窗口是放在 L0<temporary> GUI Map File。点选【File】->【Save】，选取 flight4a.gui，按下【OK】，则 Fax Order No. 3 窗口以及其 GUI 对象，将会被储存到 flight4a.gui 中。最后关闭 GUI Map Editor。

3.5 执行测试脚本

当你完成上面的练习之后，你已经准备好执行测试脚本并分析测试结果了。WinRunner 提供三种执行测试脚本的模式：Verify、Debug、Update。

- Verify：当你真正执行测试以检查应用程序的功能，并且要储存测试结果。
- Debug：当你想检查测试脚本执行是否流畅，没有错误时。
- Update：当你要更新检查点的预期值时。

执行

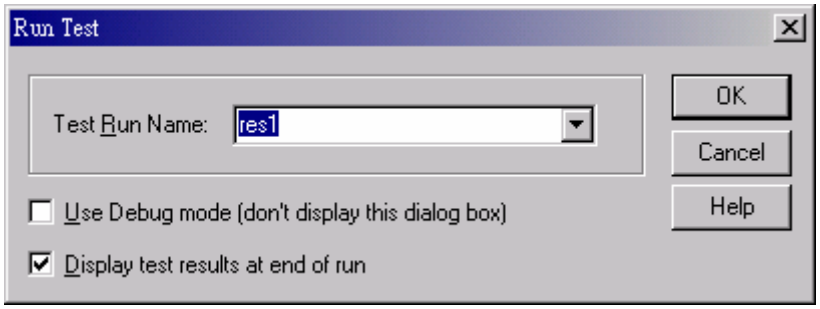
1. 确认 WinRunner 与 Flight Reservation 的主窗口都已经开启
2. 开启 lesson3 测试脚本
   先点选【File】->【Open】开启 lesson3 测试脚本。
3. 检查 Flight Reservation 在主窗口
   如果有其它对话窗口请先关闭。
4. 确认工具列上显示 

模式

5. 点选 Run From Top

   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则 Run Test 窗口将会开启，按下【OK】开始执行测试。



6. 输入 Test Run Name

   输入 Test Run Name，WinRunner 会将测试脚本执行的结果储存在 Test Run Name 的目录下，如 res1。而此测试结果将会储存在测试脚本目录下。

   请注意窗口下方【Display test results at end of run】，若勾选此选项，则当测试脚本执行完毕后，WinRunner 会自动开启测试执行结果的窗口。请勾选此选项。

7. 执行

   按下【OK】后 WinRunner 会开始执行测试脚本。

   请注意观察 WinRunner 如何执行测试脚本。

8. 检视执行结果

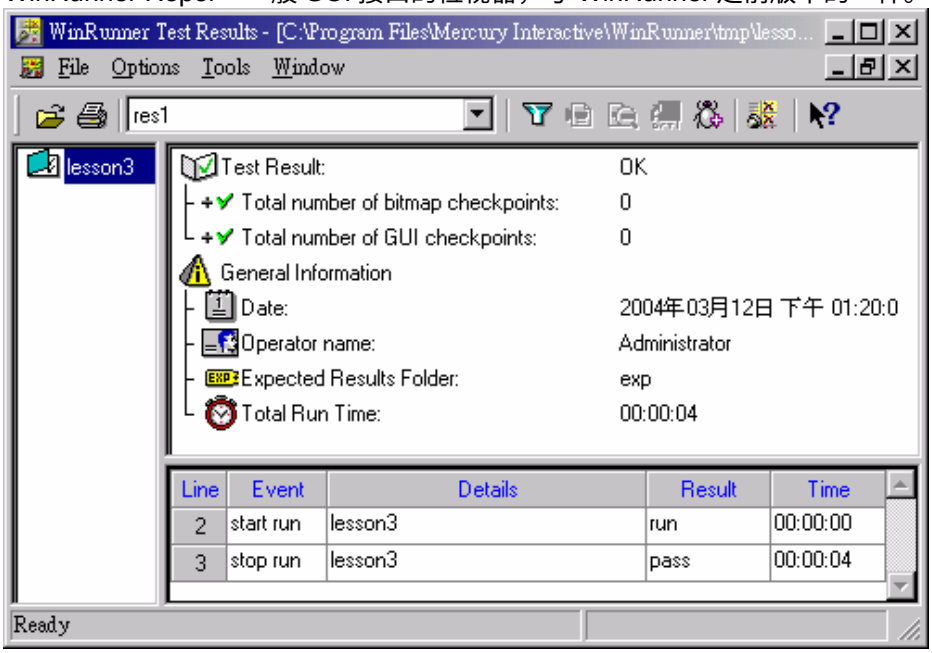
   当测试执行完毕后，WinRunner 会开启 Test Results 窗口，显示测试执行的结果。

3.6 分析测试结果

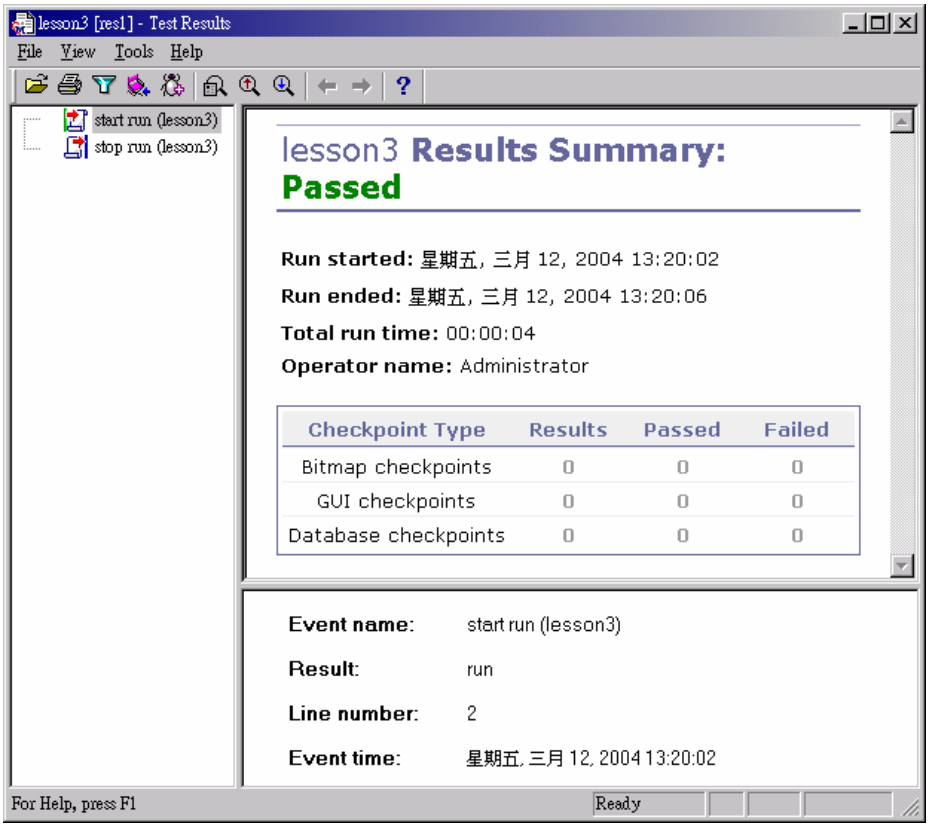
当执行完测试脚本，就可以检视测试结果。

WinRunner 提供二种类型的测试结果检视器：

- WinRunner Repor：一般 GUI 接口的检视器，与 WinRunner 之前版本的一样。



- Unified Repor：HTML 类型的检视器，与 QuickTest Professional 的测试结果检视器一样。

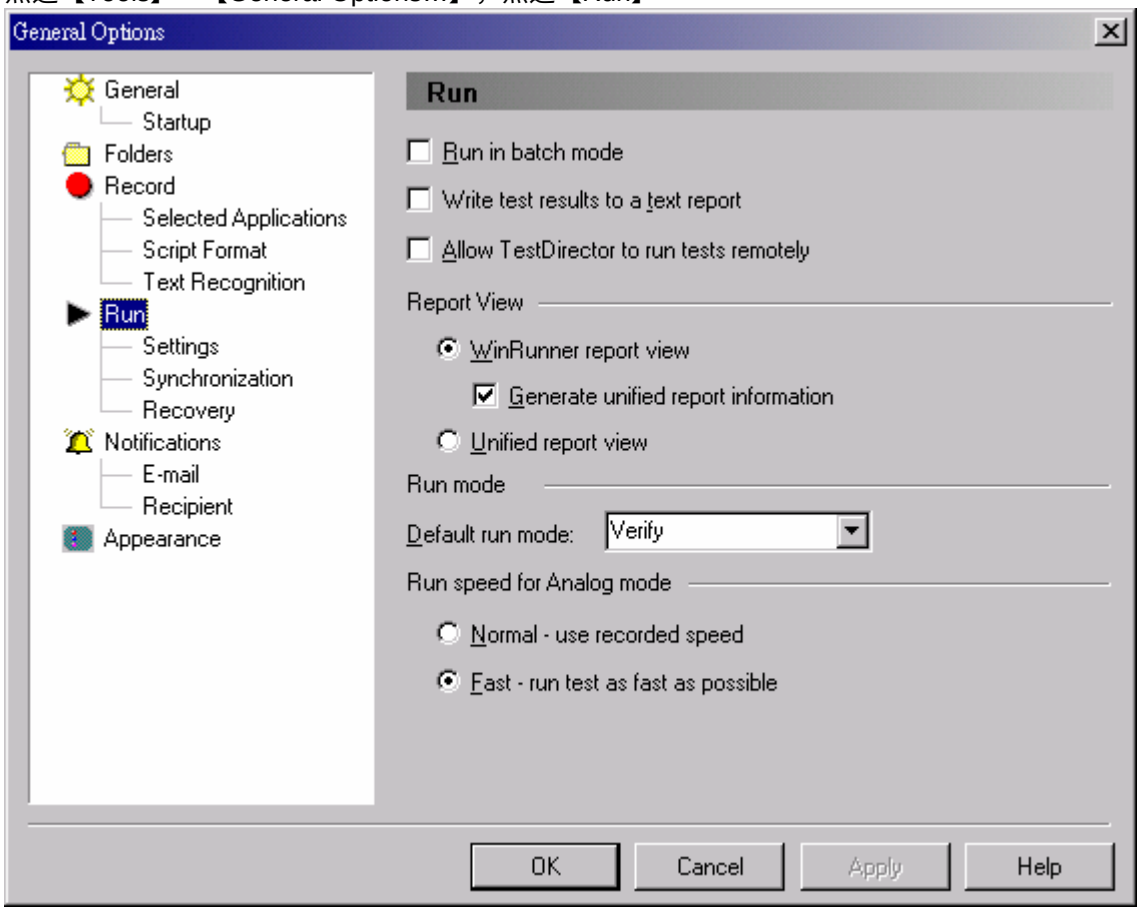


当测试执行结束时，预设 WinRunner 会以 WinRunner Report 检视器开启测试执行结果，同时也会产生供 Unified Report 检视用的测试结果。

本快速入门手册的画面以 WinRunner Report 为主，如果你的 WinRunner 设定为 Unified Report，则画面将与本快速入门手册不同。你可以依照下列步骤切换回 WinRunner Report 检视器。

切换测试结果检视器：

1. 关闭 Unified Report。
2. 点选【Tools】->【General Options…】，点选【Run】



3. 点选【WinRunner report view】，然后按下【OK】。
4. 点选【Tools】->【Test Results…】，或是按下工具列上的  按钮，就可以开启 WinRunner Report 了。

3.7 录制时的建议

1. 录制前请先关闭不必要的应用程序或窗口
2. 尽量在录制结束时，回到开始录制的画面，以便测试脚本可以重复执行测试。例如当你从主窗口开始录制测试脚本时，在测试脚本的最后，还是要回到主窗口画面。
3. 当以 Analog 模式录制时，尽量避免录制按住鼠标的动作。例如当要卷动窗口画面时，以 click 的方式卷动窗口，尽量不要以按住 scroll bar 拖曳的方式卷动窗口。

4. 当需要从 Context Sensitive 模式切换到 Analog 模式时，在切换前建议移动一下窗口，如此可确保以 Analog 模式录制完成后执行时，窗口位置为固定的。
5. 当录制点选非标准的 GUI 对象时，WinRunner 会产生 obj_mouse_click 的指令。例如当你点选一个图像对象时，WinRunner 可能会产生下列的指令：
   obj_mouse_click(GS_Drawing, 8, 53, LEFT);
   假如这个非标准 GUI 对象的行为与标准的 GUI 对象一样会类似，如其功能与按钮一样，则你可以透过对应（map）的方式，将其对应到标准的 GUI 对象，如此一来 WinRunner 便会以标准 GUI 对象的指令，如 button_press 来取代 obj_mouse_click 的指令了。
   关于如何将非标准的 GUI 对象对应到标准的 GUI 对象的详细说明，请参考 WinRunner User's Guide Configuring the GUI Map 章节。
6. 当你在 Global GUI Map File 模式下录制测试脚本时，录制的 GUI 对象之前并未录制过，则 WinRunner 会将其放在 temporary GUI Map File 中。
7. 在录制过程中可以利用【F2】切换 Context Sensitive 与 Analog 的录制模式。
8. 当你在 Global GUI Map File 模式下录制测试脚本时，记得经常检查新的 GUI 对象是否被新增到 temporary GUI Map File 中。当你离开 WinRunner 之前请记得将存放在 temporary GUI Map File 中的 GUI 对象存盘。

4. 同步点（Synchronize）

课程摘要：

- 说明何时该使用同步点
- 如何建立同步点
- 执行测试并分析结果

4.1 何时该使用同步点

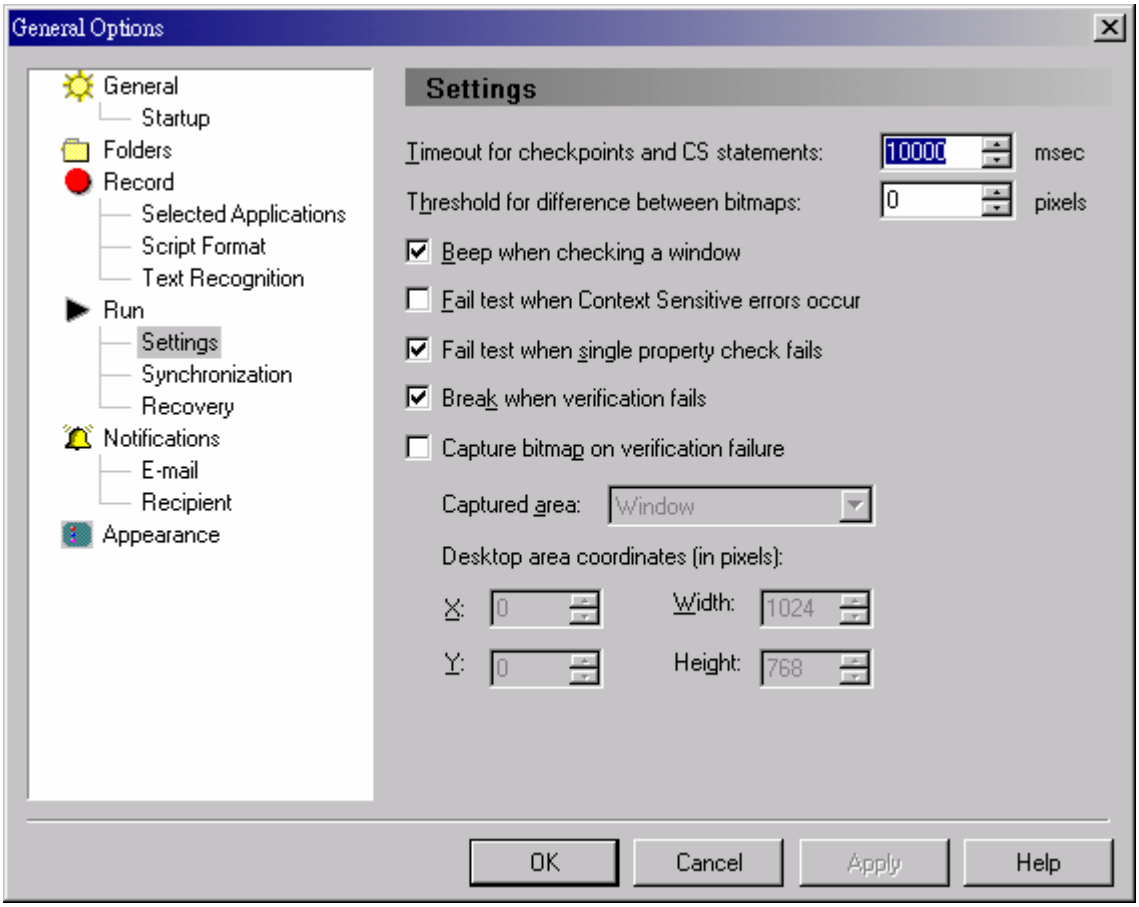
当你执行测试时，所测试的应用程序每次操作的响应时间并不一定，有时快，有时慢，导致执行输入动作的时间也需要等待。例如以下的动作常会花个几秒钟：

- 从数据库取得数据
- 等待一个窗口开启
- 等待状态列成为 100%
- 等待某个状态讯息出现

当遇到这类的情况，WinRunner 会等待一段固定的时间，直到应用程序接受输入的动作。这个等待时间的默认值为 10 秒钟。假如应用程序响应的时间超过 WinRunner 等待的时间，则测试执行就可能会失败。

假如在测试执行过程中遇到这样的情况，你可以以下列的方式解决：

- 增加 WinRunner 预设等待的时间
  点选【Tools】->【General Options…】->【Run】->【Settings】，将【Timeout for checkpoints and CS statements】的值加大，预设为 1000msec。加大这个设定可能会造成在 Context Sensitive 的动作变慢。



- 在测试脚本中插入同步点（synchronization point）
  当 WinRunner 执行到同步点时，会暂停执行以等待应用程序某些状态的改变后，再继续执行。这个方式也是较常被使用的方式。

接下来的练习你将会：

- 在 Flight Reservation 中建立一张新的订单，并新增到数据库中。
- 变更预设等待时间的设定
- 如何识别何种问题需要以同步点解决
- 加入同步点
- 执行测试脚本并检视结果

4.2 录制测试脚本

1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

3. 开始以 Context Sensitive 模式录制测试脚本
   在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

按钮。

4. 建立新的订单
   在 Flight Reservation 中点选【File】->【New Order】。

5. 填入航班与旅客资料

   请输入以下数据

   【Date of Flight】：02/01/04（日期格式为 MM/DD/YY，日期要大于今天的日期）

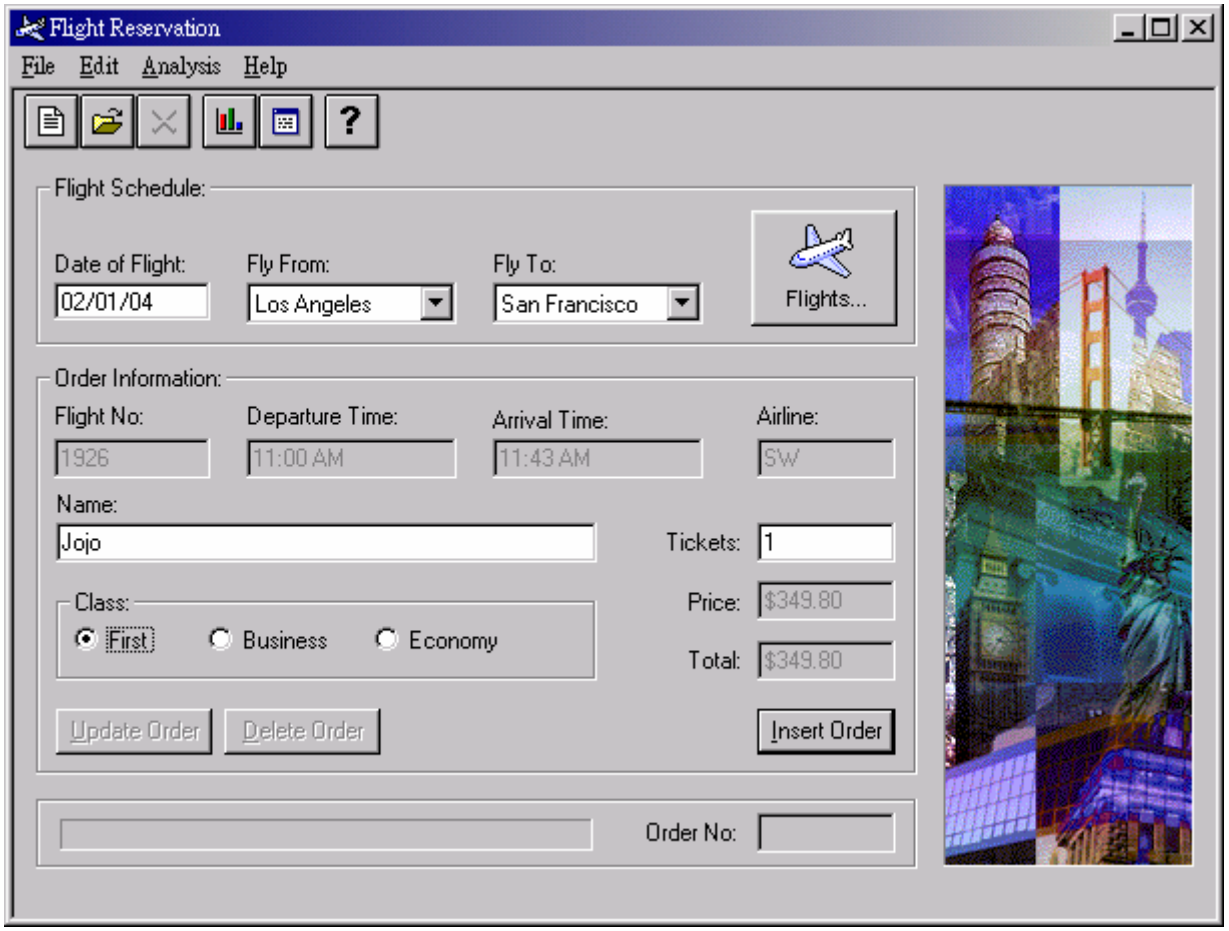
   【Fly From】： Los Angeles

   【Fly To】： San Francisco

   点选【Flights…】按钮，选取一个航班

   【Name】： Jojo

   【Class】： First



6. 点选【Insert Order】，当完成新增订单后，状态列会显示 Insert Done…的讯息。

7. 点选【Delete Order】删除刚刚新增的订单，并按下【Yes】确认。

8. 停止录制

   在 WinRunner 中点选【Test】->【Stop Recording】，或是直接点选工具列上的 Stop 按钮停止录制测试脚本。

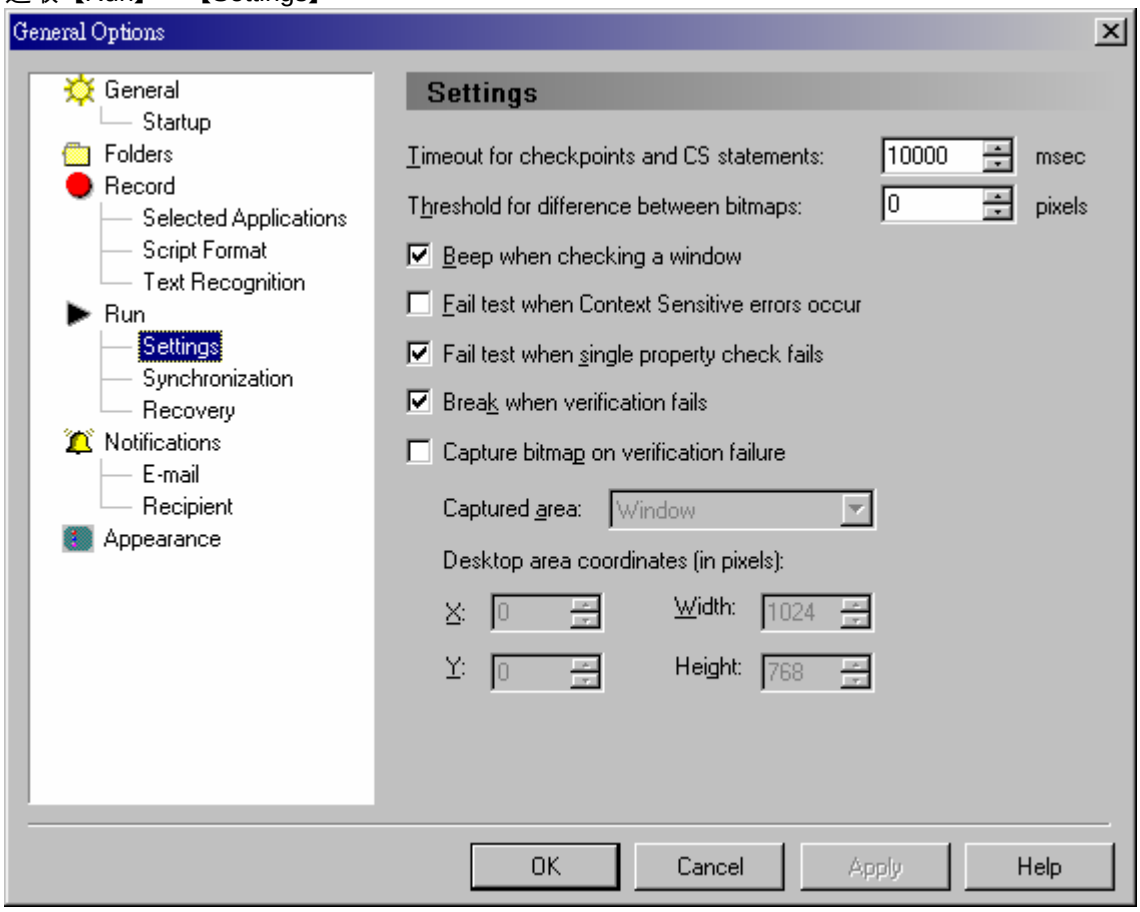
9. 储存测试脚本

   点选【File】->【Save】或是直接点选工具列上的按钮，将测试脚本储存成 lesson4。

4.3 变更预设等待时间的设定

WinRunner 预设等待时间为 10 秒钟。为了模拟出需要加入同步点的状况，接下来的练习将变更 WinRunner 预设等待时间的设定，缩短成为 1 秒钟。

1. 开启 General Optios 对话窗口
   点选【Tools】->【General Options…】，开启 General Optios 对话窗口。
2. 选取【Run】->【Settings】



3. 将 10000 msec 改成 1000 msec（1 秒）
   在【Timeout for checkpoints and CS statements】将 10000 改成 1000。
4. 按下【OK】关闭对话窗口。

4.4 如何识别何种问题需要以同步点解决

现在你已经准备好了，当你执行 lesson4 测试脚本时，将会出现同步点的问题。

1. 执行 WinRunner 并开启 lesson4

2. 点选 Run From Top

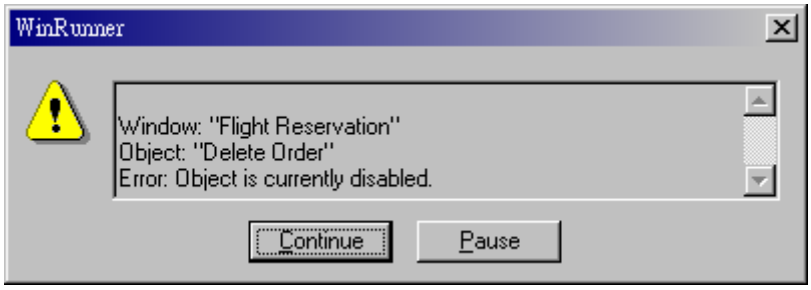
   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则 Run Test 窗口将会开启，按下【OK】开始执行测试。

   在测试脚本执行的过程中，请特别注意当 WinRunner 点选【Delete Order】按钮时发生什么事。

3. 暂停执行

   当 WinRunner 执行到点选【Delete Order】按钮时，由于 Insert Order 的动作尚未完成，而 WinRunner 最多只等待 1 秒钟，所以当 1 秒钟已经过去了，而【Delete Order】按钮还是 disabled 的状态，造成 WinRunner 无法点选【Delete Order】按钮，并跳出【Object is currently disabled】的对话窗口，表示 WinRunner 要操作的 GUI 对象是 disabled 的，所以无法执行。



4. 按下【Pause】

   这时你可以发现黄色小箭头停在点选【Delete Order】这行指令上。

```
# Flight Reservation
    set_window ("Flight Reservation", 1);
    edit_set ("Name:", "Jojo");
    button_set ("First", ON);
    button_press ("Insert Order");
    set_window ("Flight Reservation", 6);
    button_press ("Delete Order");

# Flight Reservations
    set_window ("Flight Reservations", 2);
    button_press ("是(Y)");
```

4.5 加入同步点

接下来你要在 lesson4 测试脚本中插入同步点，这个同步点会撷取状态列上 Insert Done…的图像，然后当你再次执行测试脚本时，WinRunner 会等到 Insert Done…的图像出现后，才执行点选【Delete Order】的动作。

1. 确认 Flight Reservation 已经开启
2. 确认 WinRunner 已经开启，并加载 lesson4 测试脚本与 GUI Map File

3. 将光标移动到要插入同步点的位置
   在 button_oress(“Delete Order”); 这一行上面插入一行空白行，并将光标移到这一行空白行的开头。

4. 插入同步点
   点选【Insert】->【Synchronization Point】->【For Object/Window Bitmap】，或是点选使用者自订工具列上的  按钮。
   将鼠标光标移动到 Insert Done…的状态列上并点选，WinRunner 会在测试脚本中插入一行 obj_wait_bitmap("Insert Done...", "Img1", 1); 的指令，这一行指令表示当 WinRunner 执行到这里时，会等待 Insert Done…的图像出现，等待时间为 1 秒钟，当图像出现了，才会继续往下执行。

5. 手动将 1 秒钟改成 10 秒钟
   由于等待 1 秒钟还是太短，所以手动将 obj_wait_bitmap("Insert Done...", "Img1", 1);指令改成 obj_wait_bitmap("Insert Done...", "Img1", 10);，等待 10 秒钟。

6. 储存测试脚本
   点选【File】->【Save】或是直接点选工具列上的  按钮。

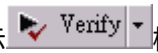
7. 如果在 Global GUI Map File 模式下，记得储存新的 GUI 对象
   由于 Insert Done…的图像为 WinRunner 新识别的 GUI 对象，所以要记得储存。
   点选【Tools】->【GUI Map Editor】，再点选【View】->【GUI Files】，你可以看到新识别的 GUI 对象是放在 L0<temporary> GUI Map File。点选【File】->【Save】，选取 flight4a.gui，按下【OK】，则新识别的 GUI 对象，将会被储存到 flight4a.gui 中。最后关闭 GUI Map Editor。

4.6 执行测试脚本并检视结果

接下来你将执行已加入同步点的测试脚本，并检视执行结果。

1. 确认 WinRunner 与 Flight Reservation 的主窗口都已经开启

2. 开启 loeeson5 测试脚本
   先点选【File】->【Open】开启 lesson4 测试脚本。

3. 确认工具列上显示 

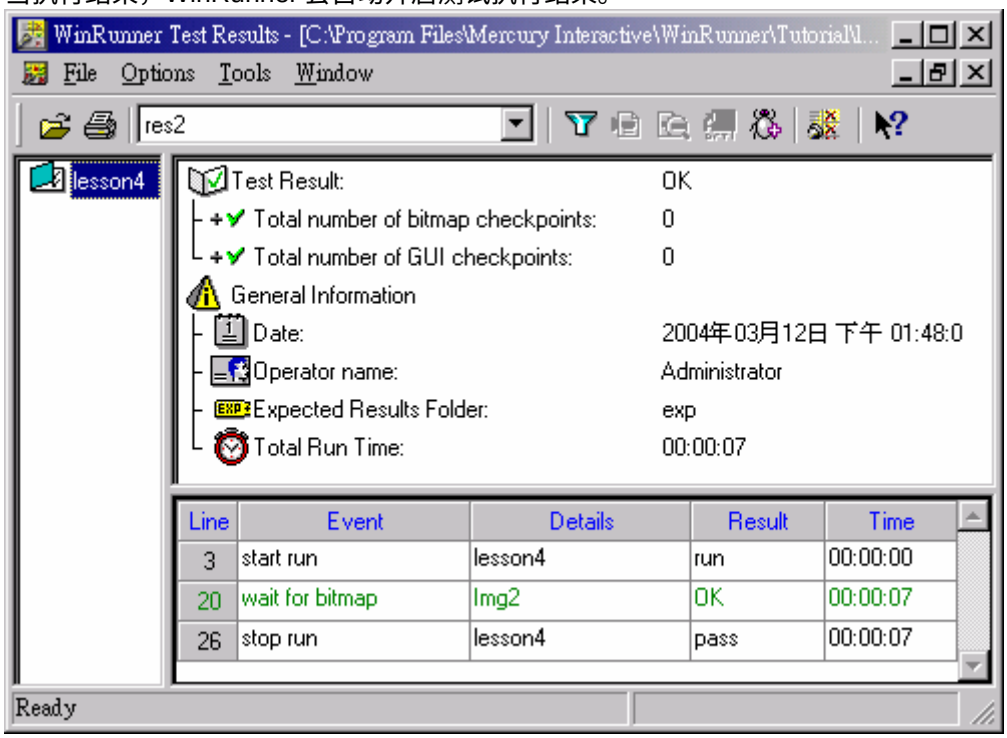
模式

4. 点选 Run From Top
   点选【Test】->【Run From Top】或是直接点选工具列上的 

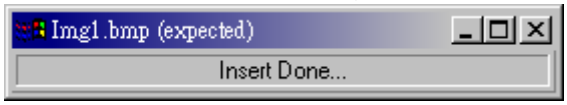
按钮，则 Run Test 窗口将会开启，接受预设 res2 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。

5. 检视测试结果

   当执行结束，WinRunner 会自动开启测试执行结果。



   你可以看到在测试结果下方的事件中，有一行绿色的 wait for bitmap 事件，表示同步点执行成功。你也可以对此事件点二下，检视此同步点的图像结果。



6. 关闭测试结果窗口

   点选【File】->【Exit】。

7. 关闭 lesson4 测试脚本

   点选【File】->【Close】。

8. 关闭 Flight Reservation

   点选【File】->【Exit】。

9. 将 WinRunner 预设等待时间改回 10 秒钟

   点选【Tools】->【General Options…】，开启 General Optios 对话窗口，选取【Run】->【Settings】。在【Timeout for checkpoints and CS statements】将 1000 改回 10000。

关于同步点的详细说明，请参考 WinRunner User's Guide Synchronizing the Test Run 章节。

5. GUI 对象检查点（Checkpoint）

课程摘要：

- 说明如何检查 GUI 对象
- 练习建立 GUI 对象检查点
- 用不同的应用程序验证 GUI 对象检查点

5.1 如何检查 GUI 对象

在测试应用程序时，通常是透过检查 GUI 对象的属性，来测试功能是否正常，当 GUI 对象的属性值与预期的值不符合时，也就表示可能有问题产生了。

在 WinRunner 中可以建立 GUI 检查点（checkpoint），检查 GUI 对象的属性，例如你可以检查：

- 

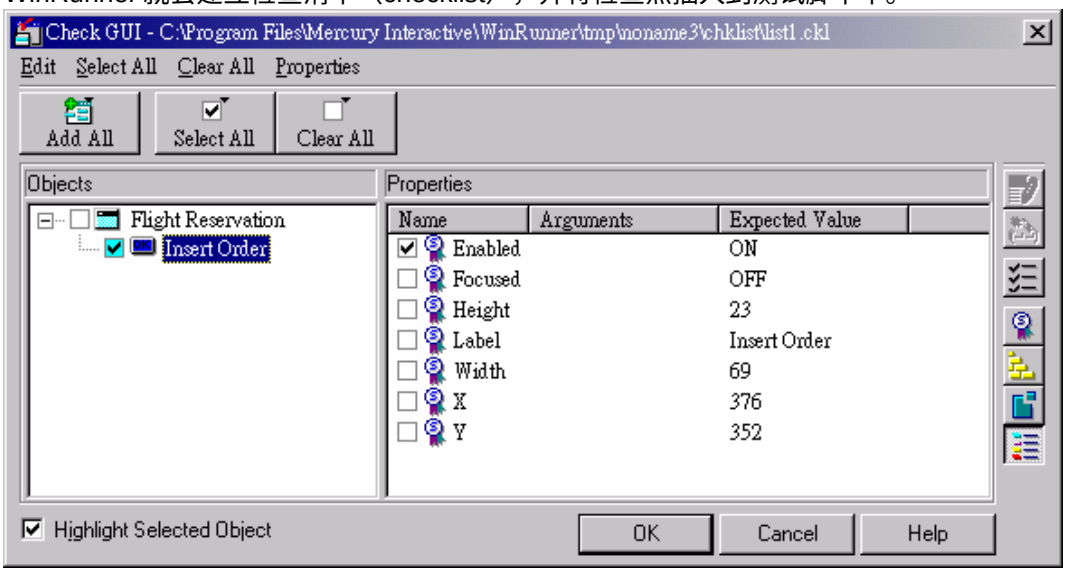
：输入字段的内容
- 

：Radio button 式否被选取
- 

：按钮是不是 enabled 的

要建立单一 GUI 对象的检查点，首先你要指到要建立检查点的 GUI 对象。

- 当你以鼠标点一下，WinRunner 会以预设检查的属性建立检查清单（checklist），并将检查点插入到测试脚本中。
  「检查清单」的内容纪录了要 WinRunner 检查的 GUI 对象与属性。
- 当你以鼠标点二下，【Check GUI】对话窗口会开启并显示你选取的 GUI 对象，以及此 GUI 对象可供检查的属性，你只要在【Check GUI】对话窗口上勾选你想检查的属性，WinRunner 就会建立检查清单（checklist），并将检查点插入到测试脚本中。



不管建立的检查点是检查预设的属性还是你选取的属性，WinRunner 会撷取建立检查点当时的属性值当作预期的值，并且在测试脚本中插入 obj_check_gui 或 win_check_gui（端看你所建立的检查点是针对 GUI 对象还是窗口对象）。

当你执行测试脚本时，WinRunner 会自动比对执行时的实际值与建立检查点时的预期值，如果一致，表示检查点检查通过；如果不一致，表示检查点检查失败。

5.2 建立 GUI 对象检查点

接下来的练习，你将对开启订单窗口建立检查点。

1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

3. 开始以 Context Sensitive 模式录制测试脚本
   在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

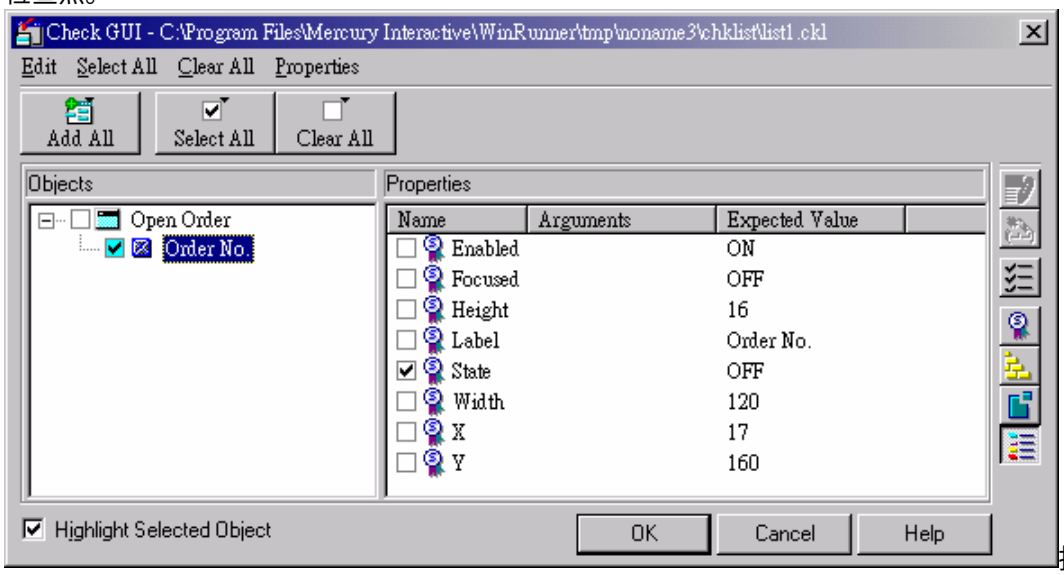
按钮。

4. 开启【Open Order】窗口
   在 Flight Reservation 点选【File】->【Open Order】，开启【Open Order】窗口。

5. 对【Order No.】check box 建立检查点

   在 WinRunner 点选【Insert】->【GUI Checkpoint】->【For Object/Window】，或是点选使用者自订工具列上的  按钮。

   以鼠标在【Order No.】check box 上点二下，则【Check GUI】对话窗口会开启并显示你选取的 GUI 对象，以及此 GUI 对象可供检查的属性。请注意如果你只点一下，则【Check GUI】对话窗口将不会开启，且 WinRunner 会直接以【State】属性当成检查点要检查的属性，并插入检查点。



   按下【OK】按钮，WinRunner 会在测试脚本中插入 obj_check_gui 检查点。

6. 输入订单编号 4

   在【Open Order】窗口中，勾选【Order No.】check box，并且在字段中输入 4。

7. 对【Order No.】check box 建立另一个检查点

   在 WinRunner 点选【Insert】->【GUI Checkpoint】->【For Object/Window】，或是点选使用者自订工具列上的  按钮。

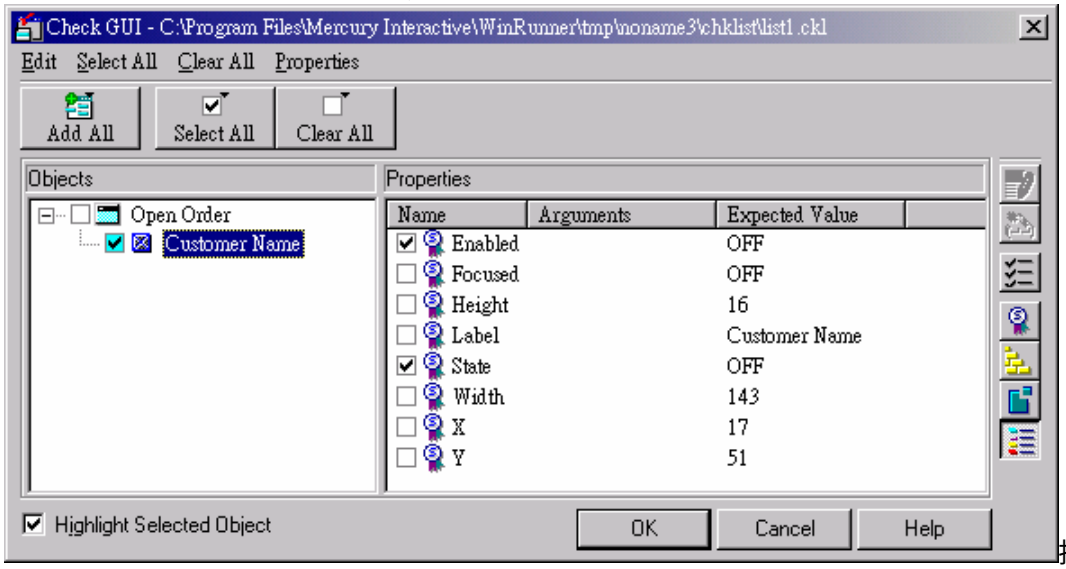
   以鼠标在【Order No.】check box 上点一下，WinRunner 会马上以预设的属性（status）在测试脚本中加上检查点（obj_check_gui），其预期值为 ON。

8. 对【Customer Name】check box 建立一个检查点

   在 WinRunner 点选【Insert】->【GUI Checkpoint】->【For Object/Window】，或是点选使用者自订工具列上的  按钮。

   以鼠标在【Customer Name】check box 上点二下，则【Check GUI】对话窗口会开启并显示你选取的 GUI 对象，以及此 GUI 对象可供检查的属性。请注意如果你只点一下，则【Check GUI】对话窗口将不会开启，且 WinRunner 会直接以【State】属性当成检查点要检查的属性，并插入检查点。

   勾选【State】与【Enabled】属性，其预期值分别为 OFF 与 OFF。



   按下【OK】按钮，WinRunner 会在测试脚本中插入 obj_check_gui 检查点。

9. 按下【OK】按钮开启订单

10. 停止录制

    在 WinRunner 中点选【Test】->【Stop Recording】，或是直接点选工具列上的 

按钮停止录制测试脚本。

11. 储存测试脚本

    点选【File】->【Save】或是直接点选工具列上的  按钮，将测试脚本储存成 lesson5。

5.3 执行测试脚本

接下来将执行 lesson5 测试脚本，以验证测试脚本可以正常执行。

1. 确认 WinRunner 与 Flight Reservation 的主窗口都已经开启

2. 开启 lesson5 测试脚本

   点选【File】->【Open】开启 lesson5 测试脚本。

3. 确认工具列上显示 

模式

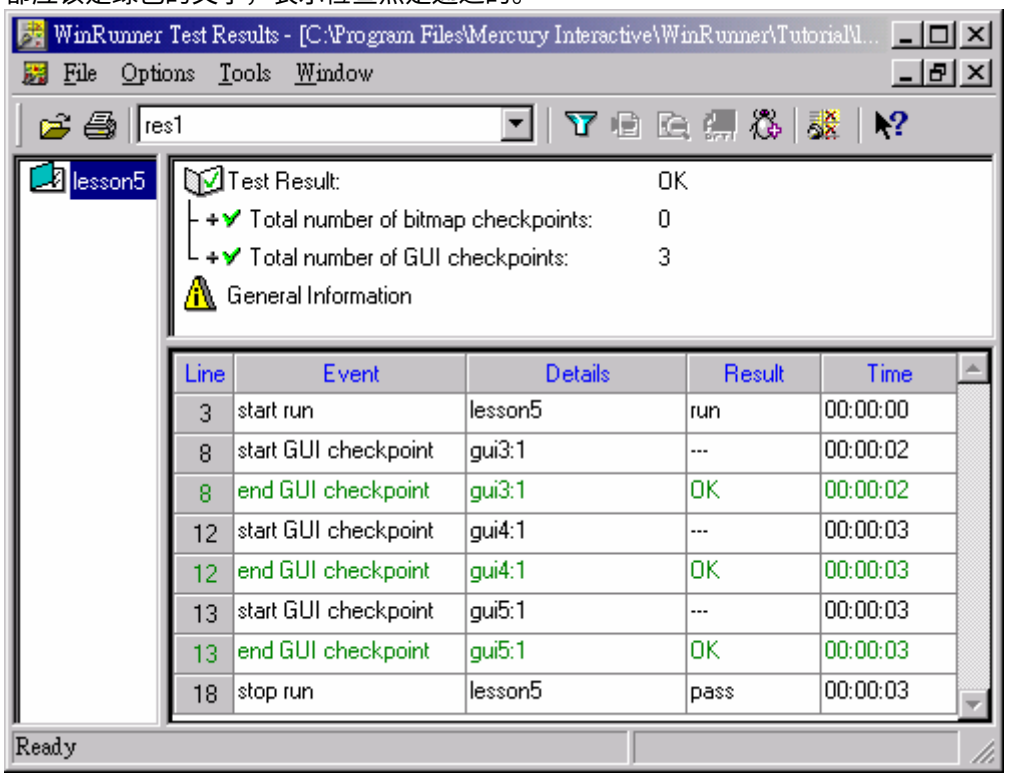
4. 点选 Run From Top

   点选【Test】->【Run From Top】或是直接点选工具列上的 

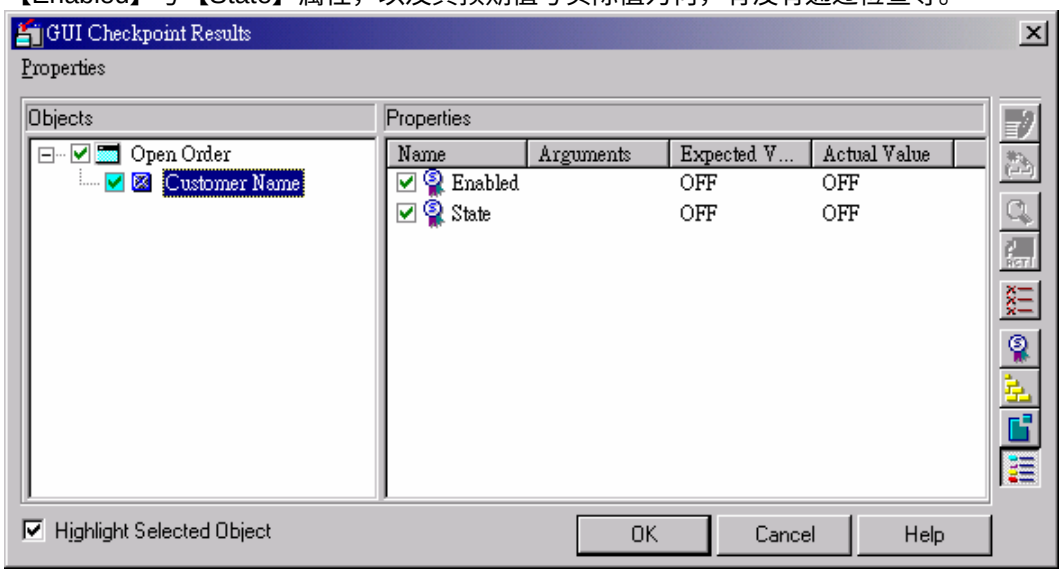
按钮，则 Run Test 窗口将会开启，接受预设 res1 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。

5. 检视测试结果

当执行结束，WinRunner 会自动开启测试执行结果。你可以看到每个【end GUI checkpoint】都应该是绿色的文字，表示检查点是通过的。



对最后一个【end GUI checkpoint】点二下，会开启【GUI Checkpoint Results】窗口，显示此检查点的测试结果。如此一检查点检查【Open Order】窗口的【Customer Name】的【Enabled】与【State】属性，以及其预期值与实际值为何，有没有通过检查等。



6. 关闭【Test Results】窗口

5.4 在另一个版本的 Flight Reservation 执行测试脚本

在接下来的练习，你会在另一个版本的 Flight Reservation 执行 lesson5 测试脚本。

1. 开启 Flight Reservation 4B 版
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4B】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

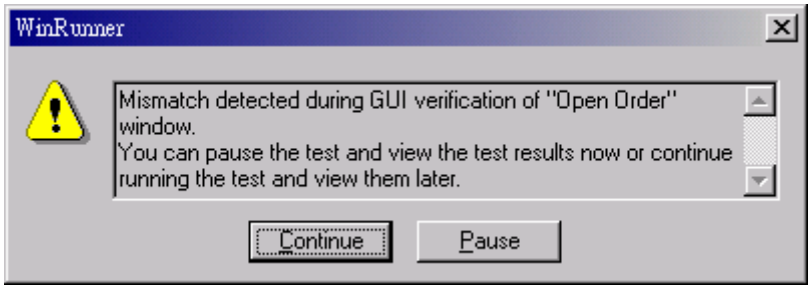
2. 开启 WinRunner 并加载 lesson5 测试脚本
   开启 WinRunner，点选【File】->【Open】开启 lesson5 测试脚本。

3. 确认工具列上显示 

模式

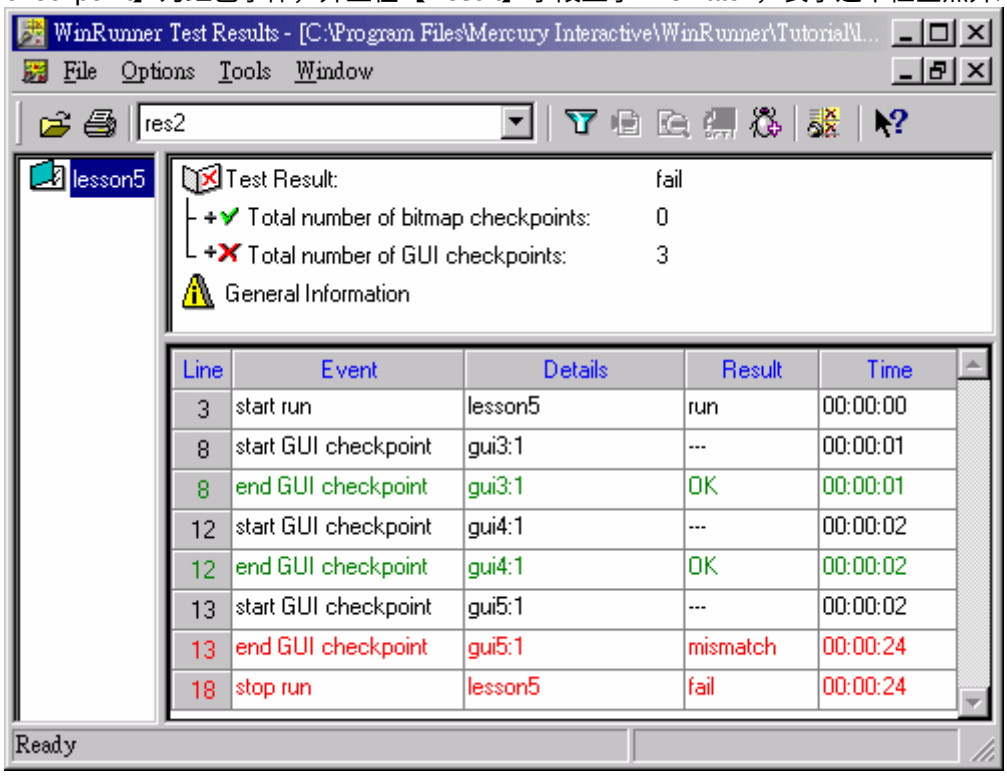
4. 点选 Run From Top
   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则 Run Test 窗口将会开启，接受预设 res2 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。
   在测试执行过程中，如果出现【Mismatch detected during GUI verification of …】讯息窗口，请按下【Continue】以便继续执行测试。

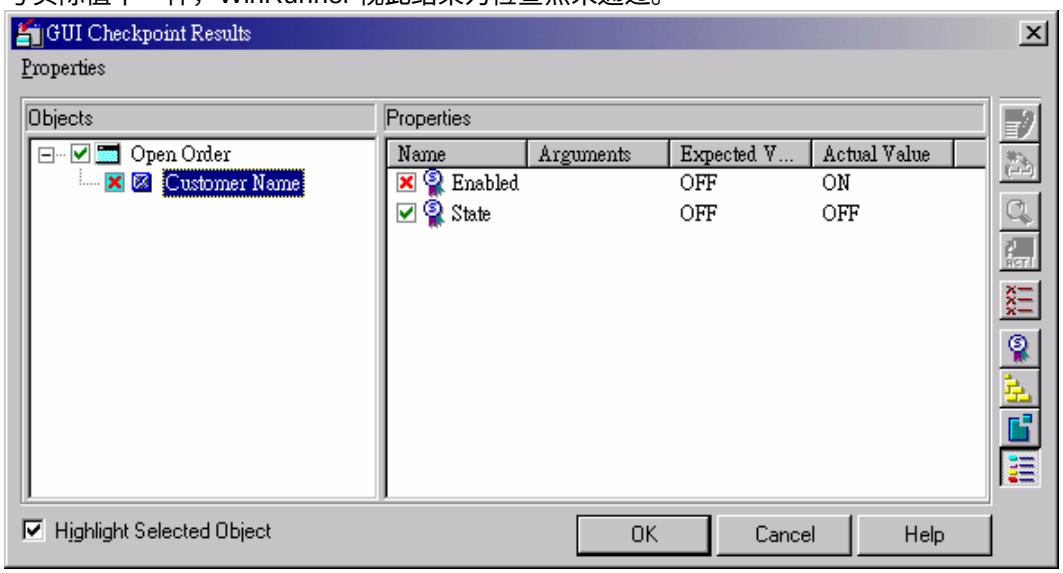


5. 检视测试结果

当执行结束，WinRunner 会自动开启测试执行结果。这次你会发现最后一次【end GUI checkpoint】为红色字体，并且在【Result】字段显示 mismatch，表示这个检查点并未通过。



点二下红色的【end GUI checkpoint】，会开启【GUI Checkpoint Results】窗口，显示此检查点的测试结果。这次你可以看到在检查【Customer Name】的【Enabled】属性时，预期值与实际值不一样，WinRunner 视此结果为检查点未通过。

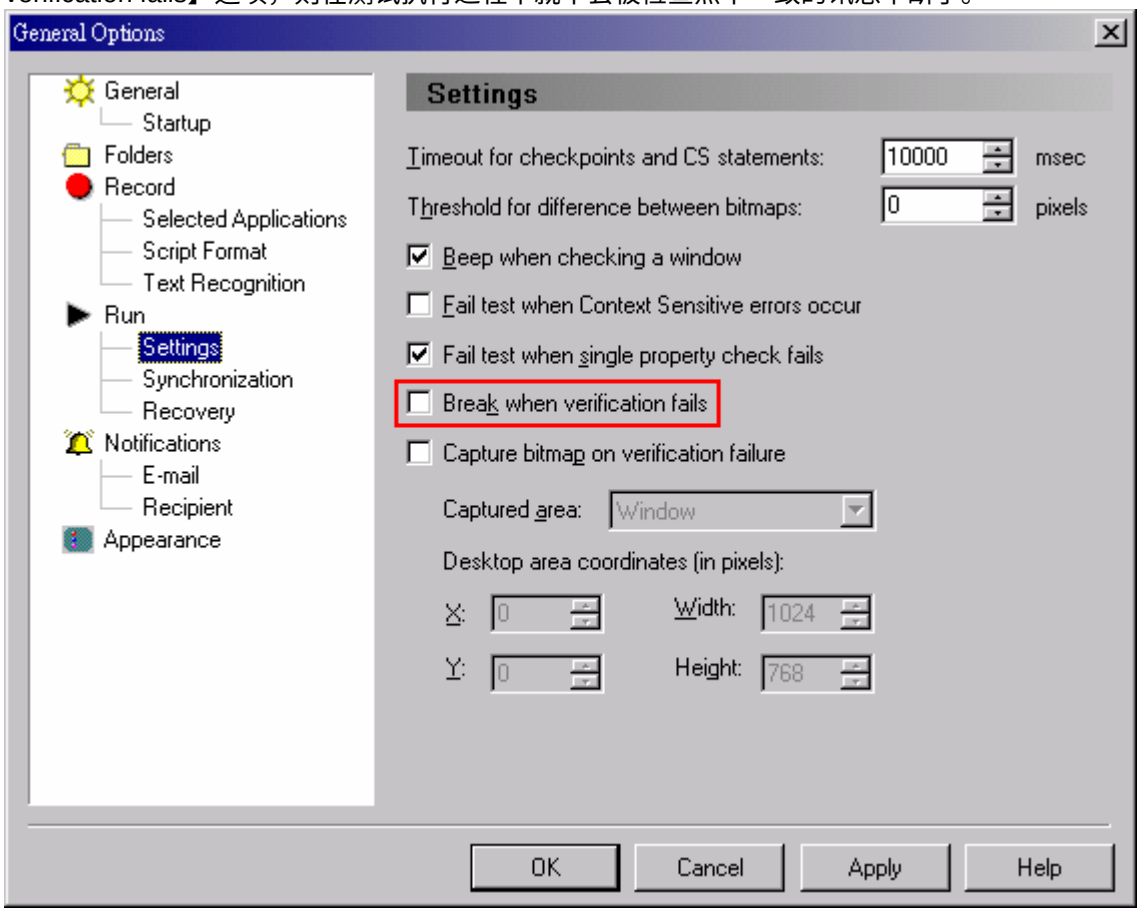


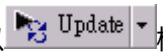
6. 关闭【Test Results】窗口

5.5 建立 GUI 对象检查点时的建议

1. 你可以使用【Insert】->【GUI Checkpoint】->【For Multiple Objects…】，一次检查窗口中多个或是全部的 GUI 对象。透过【Check GUI】对话窗口选取你要检查的 GUI 对象与其属性，以建立检查点。WinRunner 会在测试脚本中插入 win_check_gui 指令。

2. 如果你打算在深夜或无人时执行测试，你可以设定当检查点不一致时，WinRunner 不要显示讯息以免中断测试的执行。
   点选【Tools】->【General Options】->【Run】->【Settings】，清除【Break when verification fails】选项，则在测试执行过程中就不会被检查点不一致的讯息中断了。



3. 假如你想要更新检查点的预期值，请以 

模式执行一次测试脚本，则 WinRunner 会以执行当时撷取到的值，覆盖原本的预期值，成为新的预期值。

6. 图像检查点

课程摘要：

- 说明如何检查应用程序的图像
- 示范如何建立图像检查点
- 用不同的应用程序验证图像检查点

6.1 如何检查应用程序的图像

假如你想要检查应用程序中的图像，WinRunner 提供图像的检查点（bitmap checkpoint），以图素（pixel）的方式比对图像。

WinRunner 提供三种方式建立图像检查点：

- 屏幕区域（screen area）：以鼠标拖拉方式决定图像检查点的区域。
- 窗口：以整个窗口作为图像检查点的区域。
- GUI 物件：以整个 GUI 对象作为图像检查点的区域。

以下范例是以屏幕区域（screen area）方式建立图像检查点：

WinRunner 会直接撷取方框区域部分并储存成预期值，然后在测试脚本中插入 obj_check_bitmap 或是 win_check_bitmap 指令（端看你所建立的图像检查点是针对区域、GUI 对象还是窗口对象）。

当执行测试脚本时，WinRunner 会比对执行时的图像与预期的图像，将结果显示在【Test Results】窗口，如果有不一致，在【Test Results】窗口也提供检视差异的部份。

6.2 建立图像检查点

接下来的练习，你将透过以图像检查点方式，测试传真订单（Fax Order）对话窗口中的签名功能。

1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。
2. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。
3. 开始以 Context Sensitive 模式录制测试脚本
   在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

按钮。
4. 开启订单
   在 Flight Reservation 选取【File】->【Open Order】，勾选【Order No.】，输入 6 然后按下【OK】。
5. 传真订单
   在 Flight Reservation 选取【File】->【Fax Order】。
6. 输入传真号码
   在【Fax Number】中输入 10 位数字，不需要输入括号与横线。
7. 移动传真订单窗口
   将窗口移动到新的位置。
8. 切换到 Analog 录制模式
   按下键盘上的【F2】或是再按一次工具列上的 

按钮，此时录制模式将从 Context Sensitive 切换到 Analog 模式。
9. 在【Agent Signature】中签下你的名字。
10. 切换到 Context Sensitive 模式
    按下键盘上的【F2】或是再按一次工具列上的 

按钮，此时录制模式会从 Analog 模式切换回 Context Sensitive 模式。

11. 建立图像检查点检查你的签名
    选取【Insert】->【Bitmap Checkpoint】->【For Object/Window】，或是按下使用者工具列上的  按钮，以鼠标点选【Agent Signature】，WinRunner 会撷取【Agent Signature】的图像，并且在测试脚本中插入 obj_check_bitmap 指令。
12. 清除签名
    点选【Clear Signature】按钮，清除签名。
13. 再建立图像检查点
    选取【Insert】->【Bitmap Checkpoint】->【For Object/Window】，或是按下使用者工具列上的  按钮，以鼠标点选【Agent Signature】，WinRunner 会撷取【Agent Signature】的图像，并且在测试脚本中插入 obj_check_bitmap 指令。
14. 关闭传真订单窗口
    按下【Cancel】按钮关闭传真订单窗口。
15. 停止录制
    在 WinRunner 中点选【Test】->【Stop Recording】，或是直接点选工具列上的 

按钮停止录制测试脚本。
16. 储存测试脚本
    点选【File】->【Save】或是直接点选工具列上的  按钮，将测试脚本储存成 lesson6。
17. 如果你现在使用 Global GUI Map File 模式请记得将 GUI Map File 存档。
    在 WinRunner 点选【Tools】->【GUI Map Editor】。在 GUI Map Editor 点选【View】->【GUI Files】，然后选取【File】->【Save】。

6.3 检视预期结果

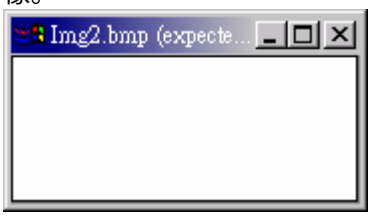
接下来你可以检视 lesson6 测试脚本的预期结果。

1. 开启 WinRunner 测试结果窗口
   选取【Tools】->【Test Results】或是直接点选工具列上的  按钮，开启测试结果窗口。

2. 检视 WinRunner 撷取的图像

   在第一个 capture bitmap 事件点二下，或直接点选工具列上的  按钮，开启第一个撷取的图像。

   在第二个 capture bitmap 事件点二下，或直接点选工具列上的  按钮，开启第二个撷取的图像。



3. 关闭测试结果窗口

   在测试结果窗口点选【File】->【Exit】关闭测试结果窗口。

6.4 在另一个版本的 Flight Reservation 执行测试脚本

接下来你将在另一个版本的 Flight Reservation 执行测试脚本，以比较何谓图像检查点未通过测试。

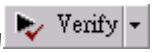
1. 关闭 Flight Reservation 4A

   选取【File】->【Close】。

2. 执行 Flight Reservation 4B

   点选【开始】->【程序集】->【WinRunner】->【Sample Application】->【Flight 4B】，，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

3. 确认目前的测试脚本为刚刚录制的 lesson6

4. 确认目前工具列上的执行模式为 

5. 点选 Run From Top

   点选【Test】->【Run From Top】或是直接点选工具列上的 

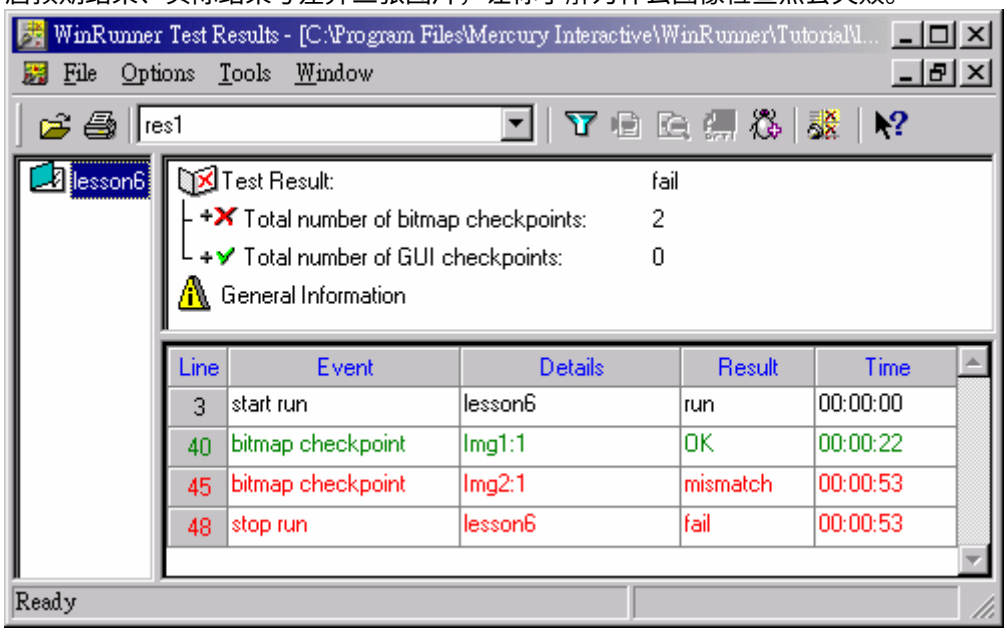
按钮，则 Run Test 窗口将会开启，接受预设 res1 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。

6. 执行时出现 mismatch 窗口

   当测试脚本执行时发现图像检查点实际结果与预期结果不一致，会出现 mismatch 窗口，这时只要按下【Continue】按钮，就可以让测试脚本继续执行下去。

7. 检视测试结果

   当测试脚本执行完毕，会自动开启测试结果窗口，你可以点选图像检查点失败的事件，将会开启预期结果、实际结果与差异三张图片，让你了解为什么图像检查点会失败。



8. 关闭测试结果窗口

   点选【File】->【Exit】。

9. 关闭 Flight Reservation 4B

   点选【File】->【Exit】。

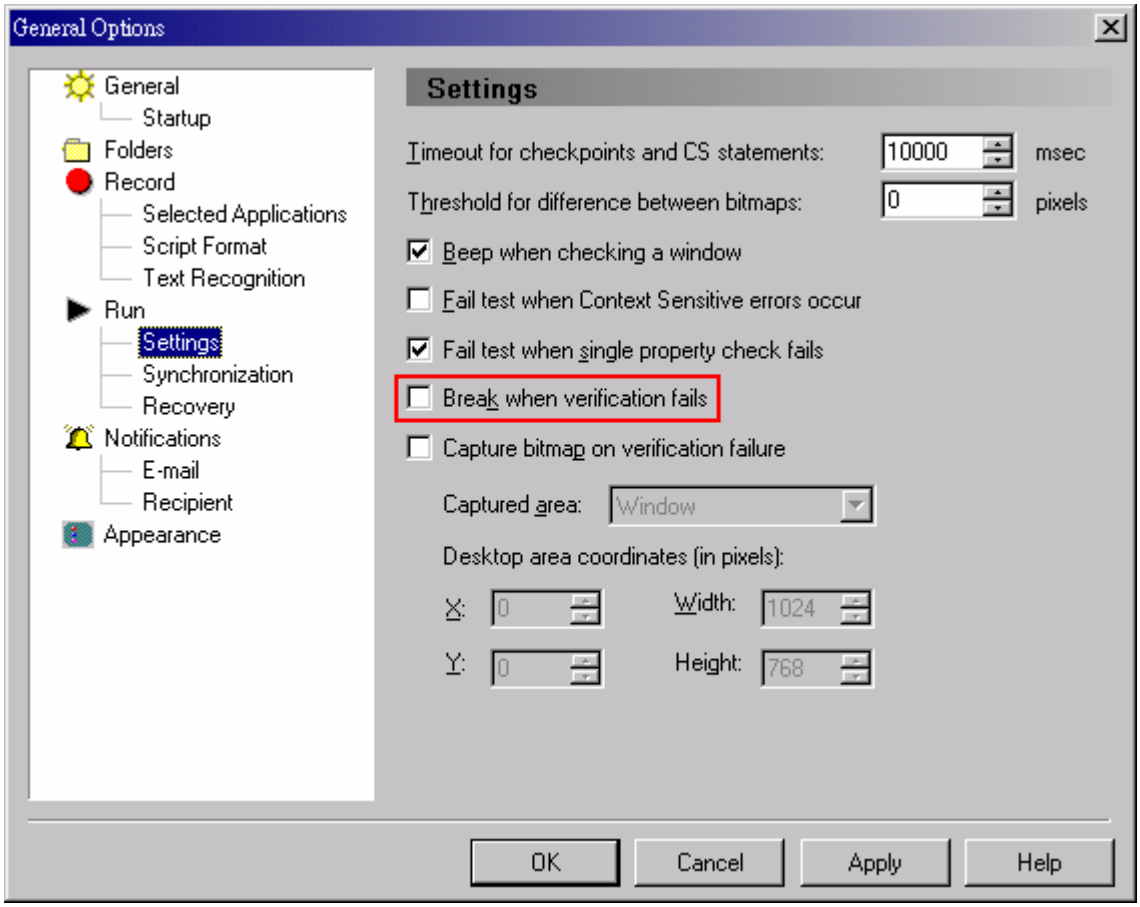
10. 关闭 lesson6 测试脚本

    点选【File】->【Close】。

6.5 建立图像检查点时的建议

1. 如果要以屏幕区域（screen area）建立图像检查点，点选【Insert】->【Bitmap Checkpoint】->【For Screen Area】，或是点选自订工具列  按钮，则鼠标会变成十字光标，然后再以拖拉方式框出要比对的区域。WinRunner 会插入 win_check_bitmap 指令，此指令参数包含屏幕区域的 x 坐标、y 坐标、宽度以及高度。

2. 如果你打算在深夜或无人时执行测试，你可以设定当检查点不一致时，WinRunner 不要显示讯息以免中断测试的执行。
   点选【Tools】->【General Options】->【Run】->【Settings】，清除【Break when verification fails】选项，则在测试执行过程中就不会被检查点不一致的讯息中断了。



3. 当执行含有图像检查点的测试脚本时，请确认屏幕显示设定与显示卡驱动程序，与当初测试脚本建立时一样。如果不一样，可能会影响图像检查点的正确性，如应该是通过的图像检查点，WinRunner 却判断为失败的图像检查点。

4. 假如你想要更新图像检查点的预期值，请以 

模式执行一次测试脚本，则 WinRunner 会以执行当时撷取到的图像，覆盖原本的预期值，成为新的预期值。

7. 使用 TSL 撰写测试脚本

课程摘要：

- 示范如何在录制好的测试脚本中以可视化的方式加入函数
- 示范如何在测试脚本中加入判断式
- 如何除错（debug）

当你在录制测试脚本时，你对应用程序的所有操作，不管是点选按钮或是键盘的输入，WinRunner 会产生一行一行的测试脚本，这每一行的测试脚本称为 TSL（Test Script Language）。除了以录制的方式产生测试脚本之外，TSL 还内建了许多函数，你可以依照需求很弹性的应用这些功能强大的函数。除此之外，WinRunner 还提供可视化工具「函数产生器（Function Generator）」，帮助你在测试脚本中快速插入函数。

函数产生器（Function Generator）提供二种使用方式：

- 你可以直接点选 GUI 对象，让 WinRunner 为你建议合适的函数，然后你再把函数加入测试脚本中。
- 你可以直接依照分类，从函数清单中挑选你要使用的函数。

除了使用函数外，TSL 也提供一般程序语言具备的元素，如条件判断（condition）、循环（loop）、表达式（arithmetic operator）。

再接下来的练习将建立测试脚本：

- 开启订单
- 开启传真订单窗口
- 检查总金额是否等于机票单价乘上机票张数
- 显示检查结果是否正确

7.1 录制基本测试脚本

从开启订单开始录制。

1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。
3. 开始以 Context Sensitive 模式录制测试脚本
   在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

按钮。
4. 开启订单
   在 Flight Reservation 选取【File】->【Open Order】，勾选【Order No.】，输入 3 然后按下【OK】。
5. 传真订单
   在 Flight Reservation 选取【File】->【Fax Order】。
6. 点选【Cancel】关闭传真订单窗口
7. 停止录制
   在 WinRunner 中点选【Test】->【Stop Recording】，或是直接点选工具列上的 

按钮停止录制测试脚本。
8. 储存测试脚本
   点选【File】->【Save】或是直接点选工具列上的  按钮，将测试脚本储存成 lesson7。

7.2 使用函数产生器（Function Generator）在测试脚本中插入函数

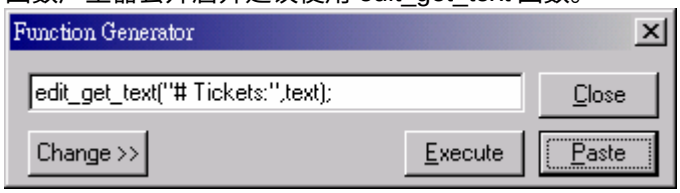
现在你已经准备好，透过加入函数的方式，取得传真订单窗口上的#Tickets、Ticket Price、Total 各字段的值。

1. 在 button_press(“Calcel”);脚本前插入一行空白
2. 开启传真订单窗口
   在 Flight Reservation 选取【File】->【Fax Order】。

3. 取得#Tickets 字段的值

   选取【Insert】->【Function】->【For Object/Window】，或是按下使用者工具列上的  按钮。

   函数产生器会开启并建议使用 edit_get_text 函数。



   这个 edit_get_text 函数会取得#Tickets 字段的值，并储存到变量中。变量的预设名称为 text。请直接将变量名称 text 改成 tickets，然后按下【Paste】按钮将函数插入测试脚本中。

```
edit_get_text("# Tickets:",tickets);
```

4. 取得 Ticket Price 字段的值

   选取【Insert】->【Function】->【For Object/Window】，或是按下使用者工具列上的  按钮。

   函数产生器会开启并建议使用 edit_get_text 函数。将变量名称 text 改成 price，然后按下【Paste】按钮将函数插入测试脚本中。

```
edit_get_text("Ticket Price:",price);
```

5. 取得 Total 字段的值

   选取【Insert】->【Function】->【For Object/Window】，或是按下使用者工具列上的  按钮。

   函数产生器会开启并建议使用 edit_get_text 函数。将变量名称 text 改成 total，然后按下【Paste】按钮将函数插入测试脚本中。

```
edit_get_text("Total:",total);
```

6. 点选【Cancel】关闭传真订单窗口

7. 储存测试脚本

   点选【File】->【Save】或是直接点选工具列上的  按钮。

7.3 在测试脚本中加入判断式

接下来你将在测试脚本中加上 if / else 的判断式，如此测试脚本便可以透过计算方式判断测试是否通过。

1. 将游标放在最后一个 edit_get_text 脚本的下一行

2. 加上下列的脚本

```
if(tickets*price == total)
    tl_step("total", 0, "Total is correct.");
else
    tl_step("total", 1, "Total is incorrect.");
```

3. 加上批注

在 if 脚本前加上一行空白，然后选取【Edit】->【Comment】，然后在 # 后加上批注。

```
#   check that the total ticket price is calculated correctly.
    if(tickets*price == total)
        tl_step("total", 0, "Total is correct.");
    else
        tl_step("total", 1, "Total is incorrect.");
```

4. 储存测试脚本

点选【File】->【Save】或是直接点选工具列上的  按钮。

7.4 了解 tl_step 函数

透过加上 tl_step 函数，你可以自行决定测试脚本中的某段动作是通过或是失败的，进而决定整个测试脚本的执行结果是通过或失败。

举例来说：

```
tl_step("total", 1, "Total is incorrect.");
```

第一个参数 total 代表这个动作的名称。

第二个参数 1 则 WinRunner 会判定此动作为失败，如果参数值为 0 则 WinRunner 会认定此动作为通过。

第三个参数 Total is incorrect 则是 WinRunner 针对此动作显示的讯息，透过有意义的描述，帮助你在检视最后测试结果时，更了解此动作代表的意义。

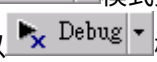
如果要更深入的了解 tl_step 函数的用法，请参考 WinRunner TSL Online Reference。

7.5 测试脚本的除错

在修改完测试脚本后，通常会执行看看是不是顺利，看看有没有语法或是逻辑上的错误。WinRunner 同时也提供了除错的工具。透过使用除错工具，你可以：

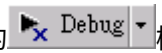
- 逐行执行测试脚本
- 设定断点
- 以 Watch List 检视变数的值

你可以以 

模式执行测试脚本以便除错，如此一来测试结果便会储存在 debug 目录下，且每次当你以 

模式执行测试脚本后，WinRunner 会覆写前一次 debug 的执行结果。

在接下来的练习你将透过逐行执行的方式，对测试脚本除错，并尝试修正错误。

1. 选取 Debug 模式
   选取工具列上的 

模式。

2. 将执行箭头放在测试脚本第一行
   用鼠标在测试脚本第一行左边灰色地方点一下，会出现一个黄色小箭头

```
# Flight Reservation
    set_window ("Flight Reservation", 3);
    menu_select_item ("File;Open Order...");
```

3. 逐行执行
   选取【Debug】->【Step】，或是点选工具列上  按钮，WinRunner 开始执行第一行测试脚本。

4. 逐行执行完整个测试脚本
   继续点选工具列上  按钮，一行一行执行完整个测试脚本。

5. 停止执行
   执行完最后一行后，点选工具列上 

按钮。

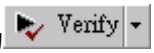
6. 检视测试结果
   当以 Debug 模式执行完测试脚本，执行结果窗口并不会自动开启。选取【Tools】->【Test Results】，或是点选工具列上的  按钮，将会开启测试结果窗口。

7. 关闭测试结果窗口
   在测试结果窗口选取【File】->【Exit】。

7.6 在另一个版本的 Flight Reservation 执行测试脚本

接下来你将在另一个版本的 Flight Reservation 执行测试脚本。

1. 执行 Flight Reservation 4B
   点选【开始】->【程序集】->【WinRunner】->【Sample Application】->【Flight 4B】，，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

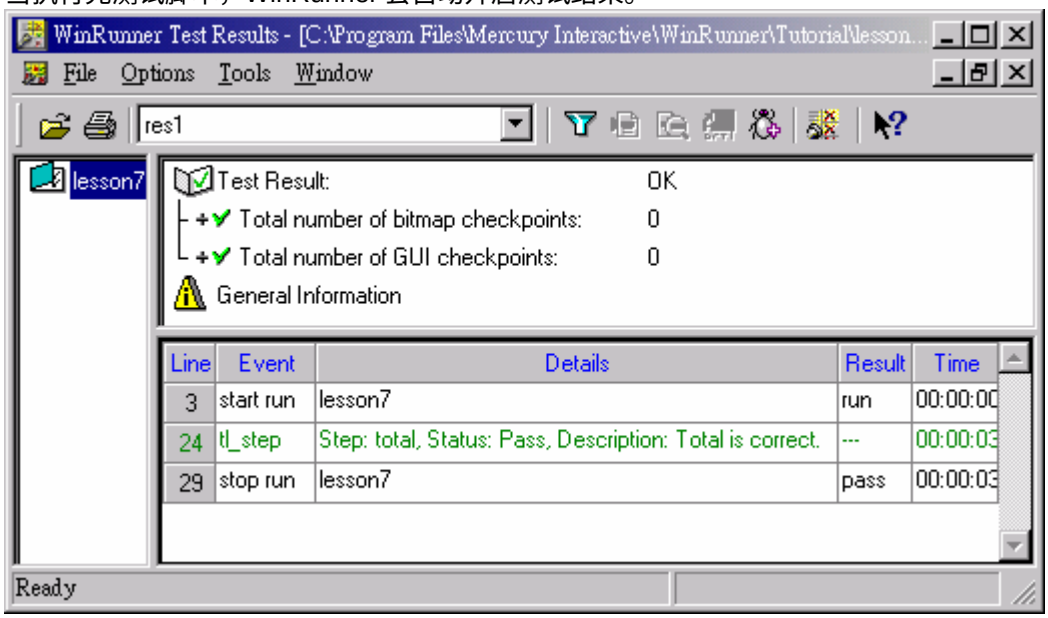
2. 点选工具列上的执行模式为 

3. 点选 Run From Top
   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则 Run Test 窗口将会开启，接受预设 res1 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。

4. 检视测试结果

   当执行完测试脚本，WinRunner 会自动开启测试结果。



   对 tl_step 点二下会显示完整个讯息，你可以看到 Description 显示的讯息就是你在测试脚本中所加入的字符串。

5. 关闭测试结果窗口

   在测试结果窗口选取【File】->【Exit】。

6. 关闭 Flight Reservation

   在 Flight Reservation 选取【File】->【Exit】。

7. 关闭测试脚本

   在 WinRunner 点选【File】->【Close】。

8. 建立数据驱动（Data-Driven）测试脚本

课程摘要：

- 示范如何使用数据驱动精灵（Data Driver Wizard）建立数据驱动测试脚本
- 解释如何以 regular expression 作为对象名称
- 如何让测试脚本重复执行

8.1 如何建立数据驱动（Data-Driven）测试脚本

当你建立好测试脚本后，你可能会想要用多组不同的数据去执行测试脚本。为了达到此目的，你可以将测试脚本转换成数据驱动测试脚本，并建立一个数据表提供测试所需的多组数据。

你可以透过下列步骤将测试脚本转换成数据驱动测试脚本：

- 加上开启及关闭数据表的指令
- 加上循环并读取数据表的每一笔数据
- 将录制的固定值与检查点的值参数化为数据表的字段值

你可以使用数据驱动精灵（Data Driver Wizard）或是自己手动修改测试脚本，将测试脚本转成数据驱动测试脚本。

当你执行数据驱动测试脚本时，WinRunner 会读取数据表的每一笔数据，并放入被参数化的地方，然后执行一次。每执行一次称为一个反复（iteration），数据表有几笔数据，WinRunner 就会执行几次反复（iteration）。并在最后的测试结果中显示每一次反复的测试结果。

在上一个课程中，你已经建立了一个测试脚本，开启一笔订单数据，并检查单价乘上票数是否等于总金额。在接下来的课程中，你会建立一个相同检查的测试脚本，并且开启多笔订单，以验证不同的票数与单价的计算也是正确的。

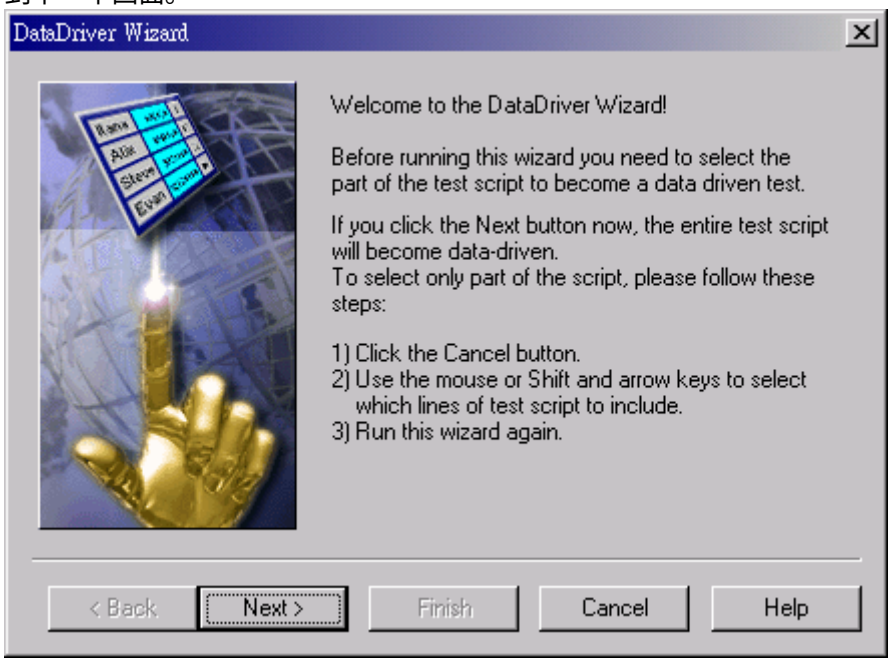
8.2 将测试脚本转成数据驱动（Data-Driven）测试脚本

接下来的练习将把上一课程录制的测试脚本转成数据驱动测试脚本。

1. 开启 lesson7 测试脚本
   如果 WinRunner 尚未开启，执行【开始】->【程序集】->【WinRunner】->【WinRunner】，选取【File】->【Open】开启 lesson7 测试脚本。
   选取【File】->【Save As】将 lesson7 另外储存成 lesson8。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

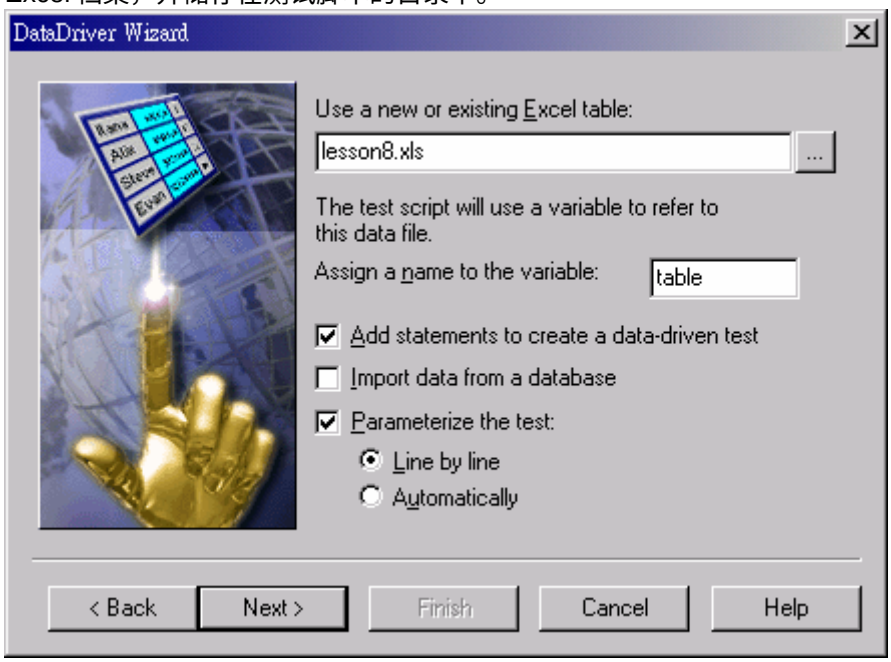
2. 执行数据驱动精灵

选取【Table】->【Data Driver Wizard】数据驱动精灵的欢迎窗口会开启。按下【Next】按钮到下一个画面。



3. 建立数据表

在【Use a new or existing Excel table】输入 lesson8.xls，数据驱动精灵会自动建立一个 Excel 档案，并储存在测试脚本的目录下。



4. 指定数据表的变量名称

   【Assign a name to the variable】使用默认值 table 为数据表的变量名称。

   在测试脚本的开头，会以数据表的变量来取代数据表的完整路径与文件名，如此一来，当你想要用其它的数据表来取代原本的测试数据时，只要修改此变量的值就可以了。

5. 设定参数化选项

   【Add statements to create a data-driven test】此选项表示由数据驱动精灵自动将转成数据驱动测试脚本的指令加到测试脚本中，预设是勾选的。

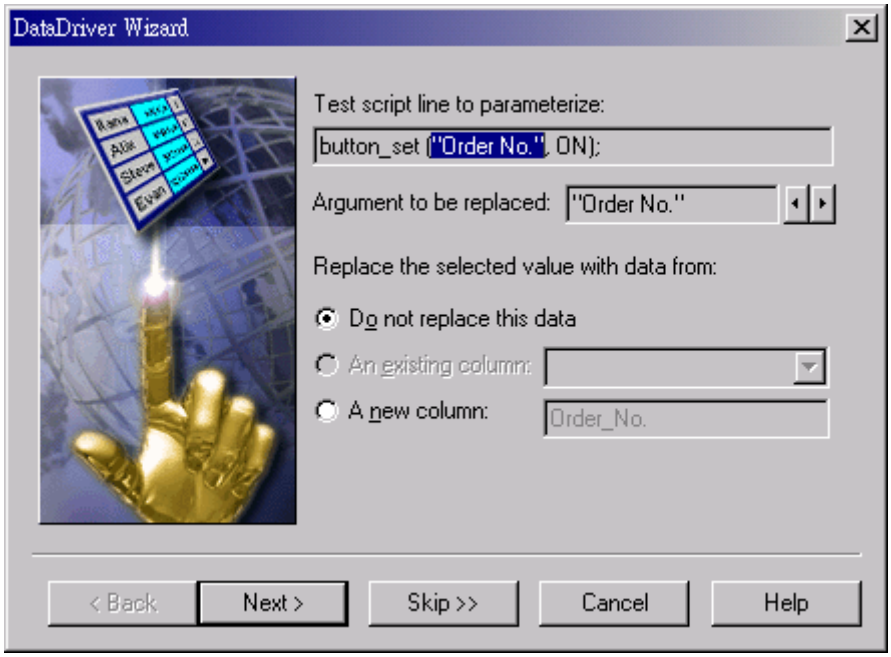
   【Parameterize the test】此选项表示要做参数化，预设是勾选的。

   【Line by line】WinRunner 会显示可以做参数化的脚本，并让你决定真正要做参数化的值为何，预设也是勾选的。

   按下【Next】按钮。

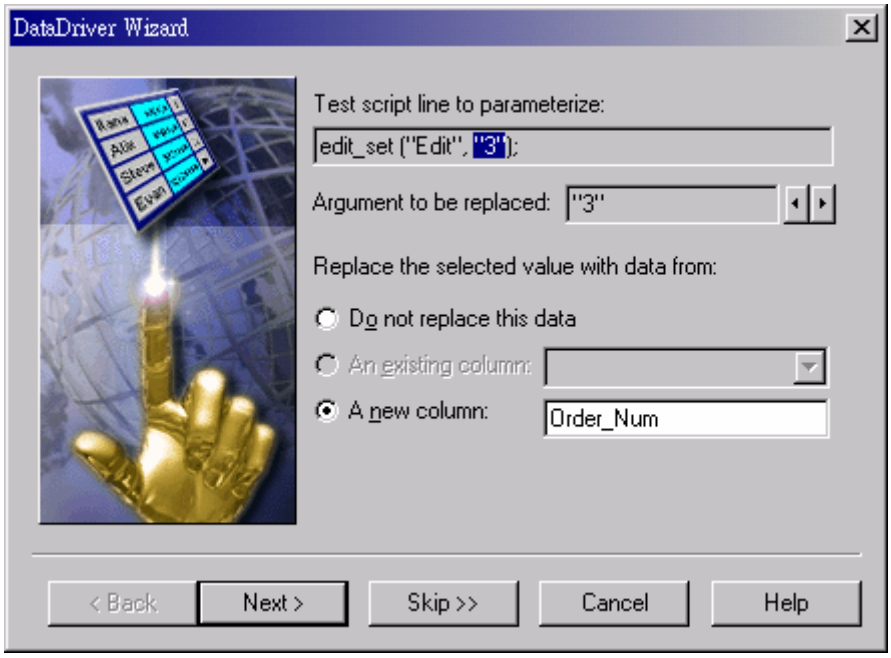
6. 选择要被参数化的值

第一个显示要参数化的测试脚本为 button_set("Order No.", ON);，这行脚本是勾选【Order No.】radio button，不是我们要作参数化的测试脚本，勾选【Do not replace this data】，按下【Next】按钮。



第二个显示要参数化的测试脚本为 edit_set("Edit", "3");，这行脚本是在【Order No.】字段中输入 3，就是我们要做参数化的脚本，此时可以看到在【Argument to be replaced】字段中显示要被参数化的资料为 3。

在【Replace the selected value with data from:】下选取【A new column】，并在字段中输入 Order_Num，则数据驱动精灵会在 lesson8.xls 中新增一栏 Order_Num 字段，且第一笔数据为被参数化的资料：3。按下【Next】按钮。



7. 完成

按下【Finish】按钮，数据驱动精灵将测试脚本转成数据驱动测试脚本，如下：

```
table = "lesson8.xls";
rc = ddt_open(table, DDT_MODE_READ);
if (rc!= E_OK && rc != E_FILE_OPEN)
    pause("Cannot open table.");
ddt_get_row_count(table,table_RowCount);
for(table_Row = 1; table_Row <= table_RowCount; table_Row ++)
{
    ddt_set_row(table,table_Row);

    # Flight Reservation
        set_window ("Flight Reservation", 4);
        menu_select_item ("File;Open Order...");

    # Open Order
        set_window ("Open Order", 1);
        button_set ("Order No.", ON);
        edit_set ("Edit", ddt_val(table,"Order_Num"));
        button_press ("OK");

    # Flight Reservation
        set_window ("Flight Reservation", 2);
        menu_select_item ("File;Fax Order...");

    # Fax Order No. 3
        set_window ("Fax Order No. 3", 1);
        edit_get_text("# Tickets:",tickets);
        edit_get_text("Ticket Price:",price);
        edit_get_text("Total:",total);

    #   check that the total ticket price is calculated correctly.
        if(tickets*price == total)
            tl_step("total", 0, "Total is correct.");
        else
            tl_step("total", 1, "Total is incorrect.");

        button_press ("Cancel");
}
ddt_close(table);
```

8.3 將数据加入数据表

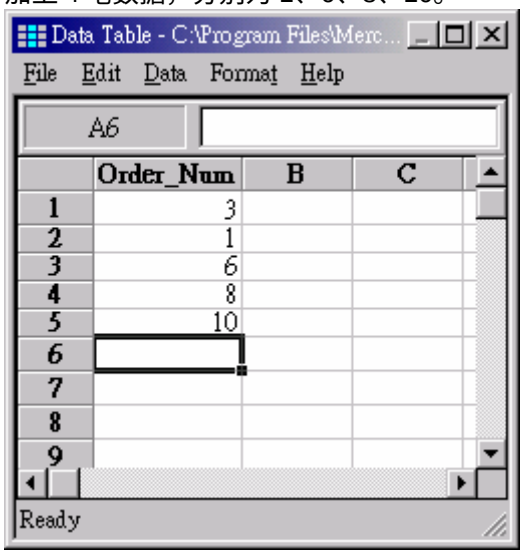
接下来将数据加入数据表中。

1. 开启数据表

选取【Table】->【Data Table】开启数据表，可以看到第一栏为 Order_Num，且其第一笔资料为 3。

2. 加上数据

   加上 4 笔数据，分别为 1、6、8、10。



3. 储存数据表

   选取【File】->【Save】将数据表存盘，选取【File】->【Close】关闭数据表。

4. 储存测试脚本

   点选【File】->【Save】或是直接点选工具列上的  按钮。

8.4 以 regular expression 调整测试脚本

你的测试脚本已经接近完成了，不过在执行测试脚本之前，还是要先检查一下测试脚本是否有冲突的地方。虽然数据驱动精灵已经帮你将测试脚本中需要作参数化的值，以参数取代掉了，但是数据驱动精灵并没有帮你取代像对象 label 的值，这些固定的值可能会导致数据驱动测试脚本执行失败。

在 Flight Reservation 这支范例程序中，传真窗口的 label 会随着开启的订单编号而改变，所以如果你执行刚刚转换成数据驱动的测试脚本，在第二次反复（iteration）时，就会出现找不到窗口的错误讯息。

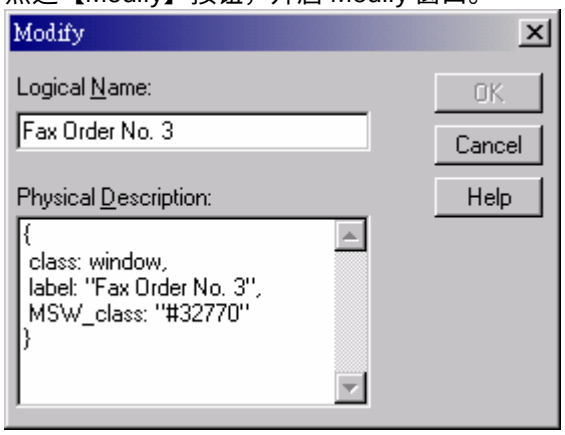
要解决这个问题，可以透过 regular expression。所谓 regular expression 就是利用某些字符，来表示特定的字符，例如用*来表示所有字符。接下来你会将传真窗口的 label 属性修改成 regular expression，以解决找不到窗口的问题。

1. 在 flight4a.GUI 找到 Fax Order 窗口

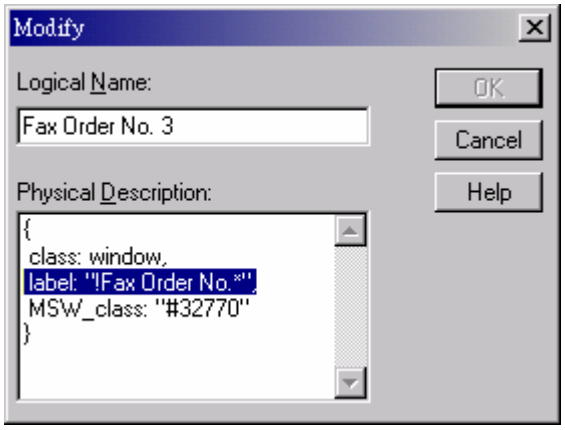
   选取【Tools】->【GUI Map Editor】。选取【View】->【GUI Files】。选择 flight4a.gui。选取 Fax Order No. 3 窗口。

2. 修改窗口 label 属性

   点选【Modify】按钮，开启 Modify 窗口。



   在【Physical Description】字段中，将 label 这一行第一个双引号后加上!，然后将 3 与前面的空白删除改成*号。



3. 关闭 Modify 窗口

   按下 OK 按钮关闭 Modify 窗口。

4. 如果你现在使用 Global GUI Map File 模式请记得将 GUI Map File 存盘。

   在 WinRunner 点选【Tools】->【GUI Map Editor】。在 GUI Map Editor 点选【View】->【GUI Files】，然后选取【File】->【Save】。

8.5 修改结果信息

现在你已经可以执行这个测试脚本了。只不过在显示测试结果时的信息都是一样的。为了让测试结果也能更有意义，接下来将修改测试脚本的 tl_step，使其显示的信息更有意义。

1. 修改 tl_step

   找到第一个 tl_step 脚本：

```
tl_step("total", 0, "Total is correct.");
```

   并改成以下的脚本：

```
tl_step("total", 0, "Correct. "tickets" tickets at $"price" cost $"total".");
```

   同样找到第二个 tl_step 脚本：

```
tl_step("total", 1, "Total is incorrect.");
```

   修改成以下的脚本：

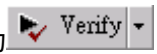
```
tl_step("total", 1, "Error. "tickets" tickets at $"price" does not equal $"total".");
```

2. 储存测试脚本

   点选【File】->【Save】或是直接点选工具列上的  按钮。

8.6 执行测试脚本并分析结果

接下来执行此测试脚本，并于测试脚本执行完成后，检视测试结果。

1. 确认 Flight 4A 已经开启在桌面上
2. 点选工具列上的执行模式为 

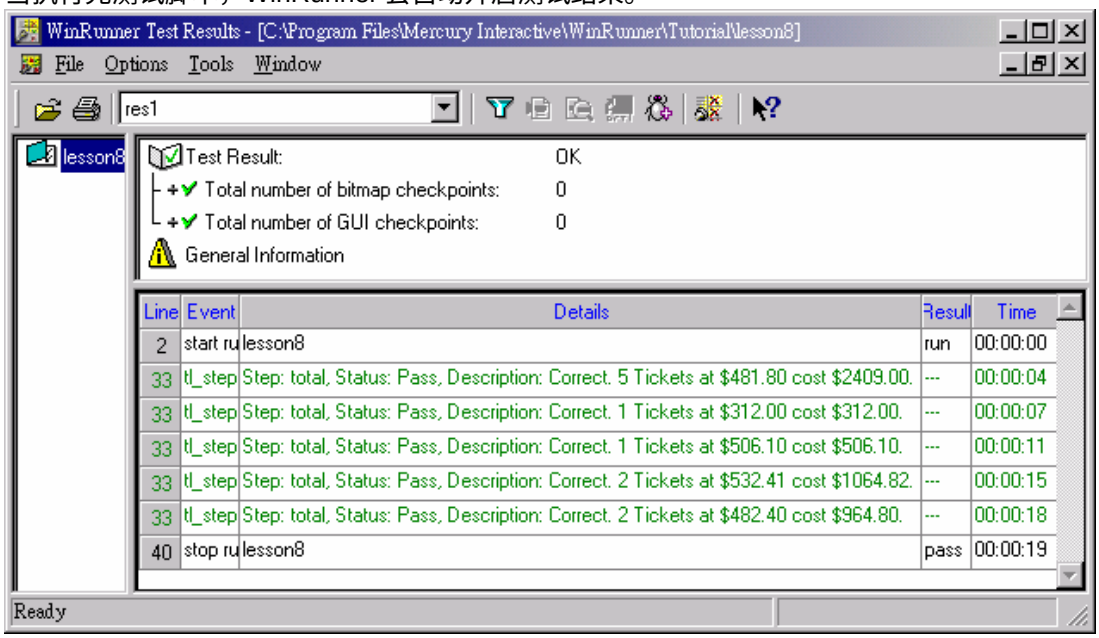
3. 点选 Run From Top

   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则 Run Test 窗口将会开启，接受预设 res1 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。

4. 检视测试结果

   当执行完测试脚本，WinRunner 会自动开启测试结果。



   测试结果显示了 5 笔 tl_step 纪录，而且每一笔纪录都显示了票数、单价、总金额的值。

5. 关闭测试结果

   选取【File】->【Exit】关闭测试结果窗口。

6. 关闭 Flight Reservation

   选取【File】->【Exit】关闭 Flight Reservation 范例程序。

7. 关闭 lesson8 测试脚本

   选取【File】->【Close】关闭测试脚本。

8.7 建立数据驱动脚本时的建议

1. 你可以只将测试脚本的一部份转成数据驱动测试脚本，只要在开启数据驱动精灵前，先选取要转成数据驱动的测试脚本即可。同一测试脚本中也可以包含多个数据驱动测试脚本。
2. 你可以开启 default.xls 然后储存成其它档名，以便使用多个测试数据表。
3. GUI 检查点、图像检查点、图像同步点、常数都可以参数化。
4. 数据表的使用方式与 Excel 工作表相同，你也可以在储存格中使用公式。
5. 在执行数据驱动测试脚本之前，你应该先检查整个测试脚本以及其它部份，如 GUI 对象的属性等，看看是否有冲突的部份。以下是解决冲突的二种解决方案：
   - 使用 regular expression 将属性变动的部份以特殊字符取代。
   - 重新设定 GUI Map Configuration，将会变动的属性排除掉。
6. 测试执行时并不需要开启数据表检视器（data table viewer）。

9. 文字检查点（Text checkpoint）

课程摘要：

- 如何读取图像或非标准 GUI 对象上的文字
- 建立一个读取并验证文字的测试脚本
- 执行测试脚本并分析结果

WinRunner 提供读取图像或非标准 GUI 对象上的文字的功能，并手动撰写测试脚本判断，以检查文字是否正确。

举例来说，你可以透过文字检查点达到下列的目的：

- 验证某个值是否在一定范围之内
- 计算数值是否正确
- 当某个指定的文字出现在画面上时，就执行某些动作

你只要指定要读取文字的区域、对象或窗口，就可以建立文字检查点。

WinRunner 会以 win_get_text 或是 obj_get_text 读取文字，并将读取到的文字储存到变量中。然后你再以手动撰写测试脚本的方式，检查变量中的文字是否为预期的文字。

另外要提醒你，当你要验证标准的 GUI 对象（按钮、功能选单、list、edit box 等）上的文字，建议只要使用 GUI 检查点，以省去手动撰写测试脚本的不便。

接下来的练习，你将会建立测试脚本：

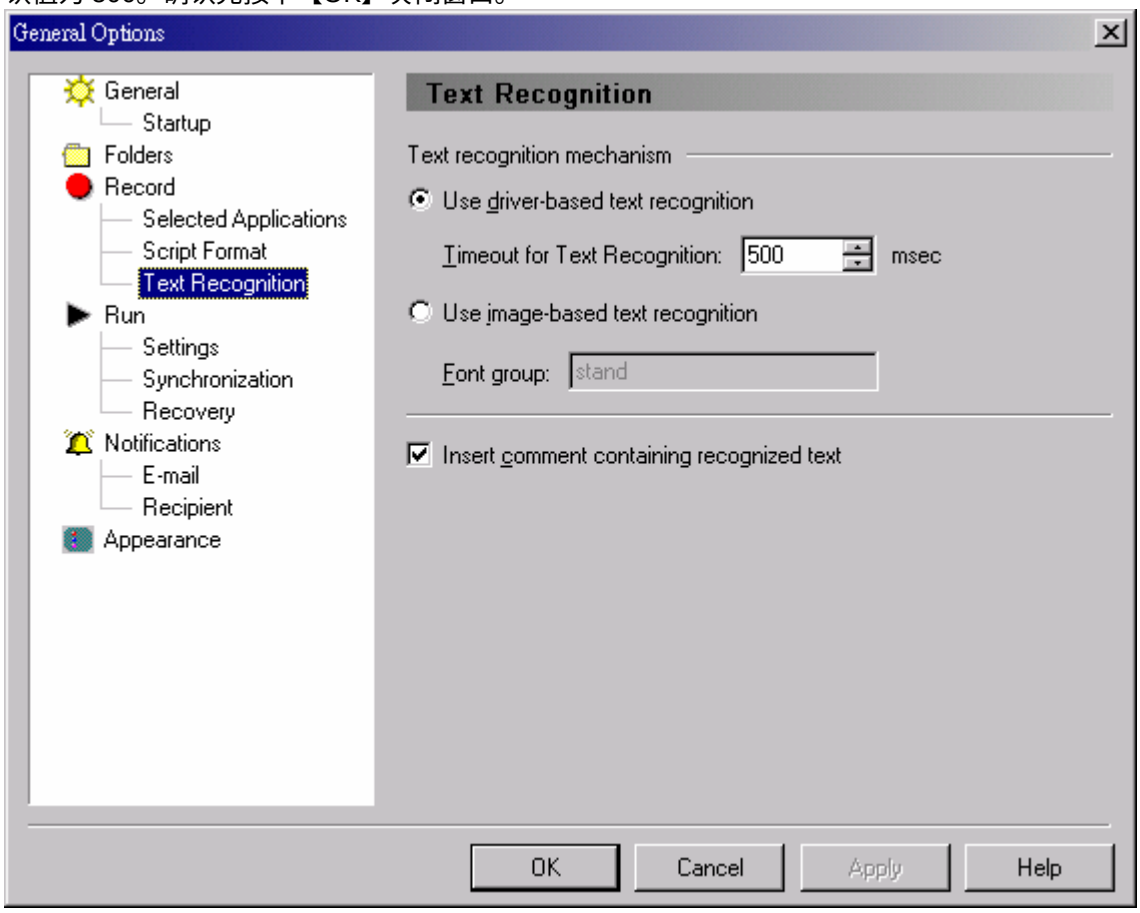
- 开启图表并读取卖出的票数
- 新增一笔订单
- 再开启图表检查卖出的票数是否被更新
- 回报数值是否正确

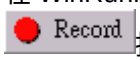
9.1 从应用程序读取文字

1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

3. 确认文字识别的设定
   选取【Tools】->【General Options】开启 General Option 窗口，点选【Record】->【Text Recognition】，确认一下【Timeout for Text Recognition】设定为合理的值（如不为 0），默认值为 500。确认完按下【OK】关闭窗口。



4. 开始以 Context Sensitive 模式录制测试脚本
   在 WinRunner 点选【Test】->【Record – Context Sensitive】或是直接点选工具列上的 

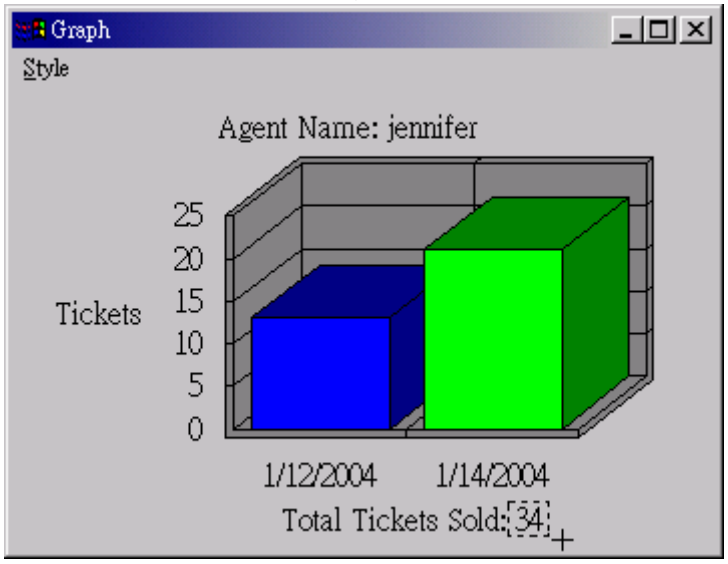
按钮。

5. 开启图表
   在 Flight Reservation 中点选【Analysis】->【Graphs】。

6. 读取图表上的票数

   在 WinRunner 点选【Insert】->【Get Text】->【From Screen Area】或是直接点选工具列上的  按钮。

   此时鼠标光标会变成十字光标，以左键拖拉的方式框住票数后，再以鼠标右键结束操作。



   WinRunner 会插入 obj_get_text 指令，并且在后面加上批注文字「# 34」，表示目前读取到的文字为 34。

```
obj_get_text("GS_Drawing", text, 247, 202, 266, 222);    # 34
```

7. 关闭图表窗口

8. 建立新订单

   在 Flight Reservation 中选取【File】->【New Order】。

9. 填入航班与旅客资料

请输入以下数据

【Date of Flight】：03/10/04（日期格式为 MM/DD/YY，日期要大于今天的日期）

【Fly From】： Denver

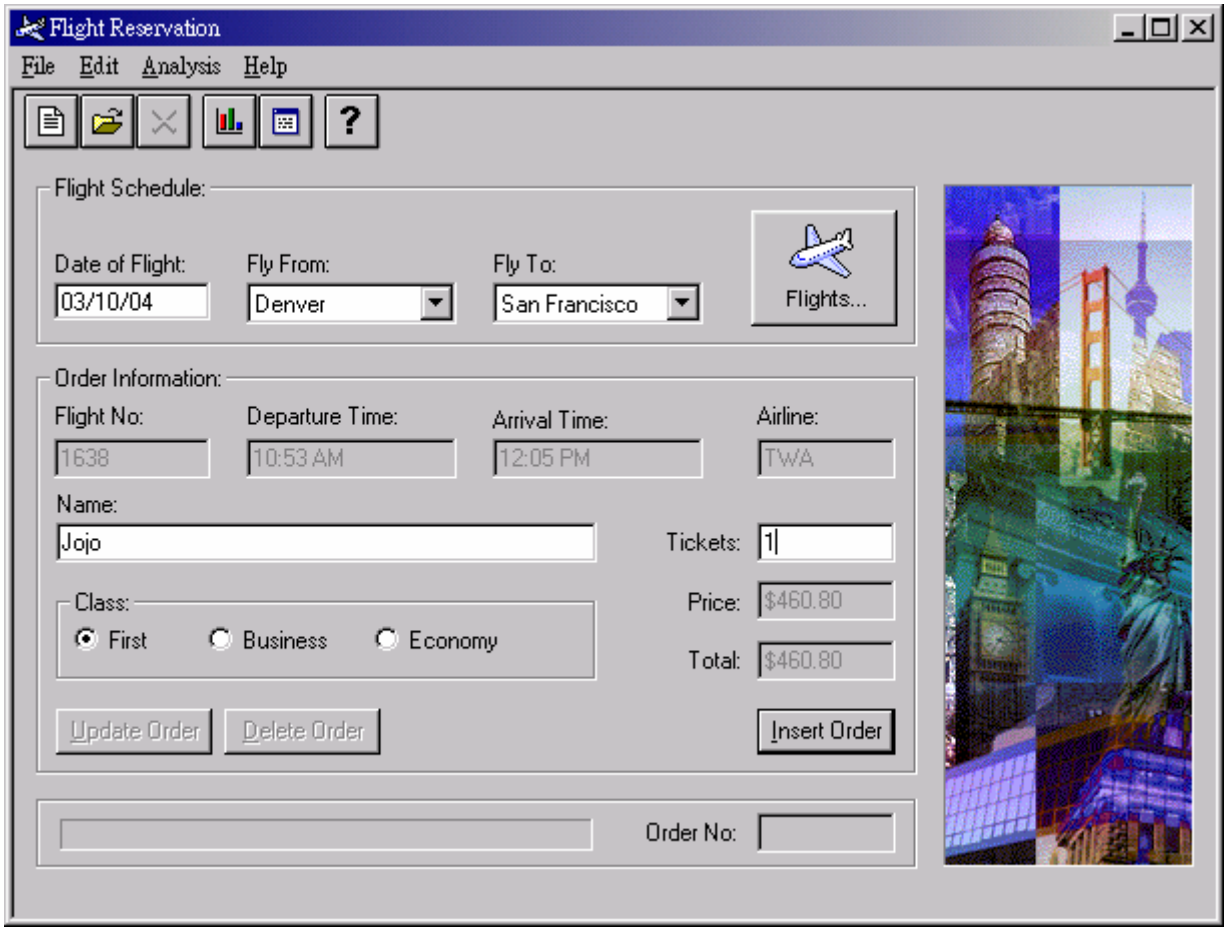
【Fly To】： San Francisco

点选【Flights…】按钮，选取一个航班

【Name】： Jojo

【Class】： First

【Tickets】： 1



10. 新增订单

点选【Insert Order】，当完成新增订单后，状态列会显示 Insert Done…的讯息。

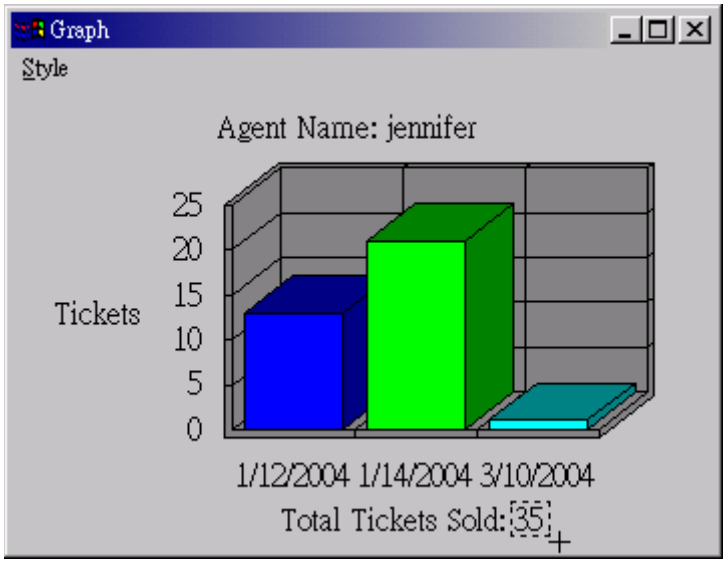
11. 插入同步点

点选【Insert】->【Synchronization Point】->【For Object/Window Bitmap】，或是点选使用者自订工具列上的  按钮。

将鼠标光标移动到 Insert Done…的状态列上并点选，WinRunner 会在测试脚本中插入一行 obj_wait_bitmap("Insert Done...", "Img1", 1); 的指令

12. 再开启图表
    在 Flight Reservation 中点选【Analysis】->【Graphs】。

13. 读取图表上的票数
    在 WinRunner 点选【Insert】->【Get Text】->【From Screen Area】或是直接点选工具列上的  按钮。
    此时鼠标光标会变成十字光标，以左键拖拉的方式框住票数后，再以鼠标右键结束操作。



    WinRunner 会插入 obj_get_text 指令，并且在后面加上批注文字「# 35」，表示目前读取到的文字为 35。

14. 关闭图表窗口

15. 停止录制
    在 WinRunner 中点选【Test】->【Stop Recording】，或是直接点选工具列上的 

按钮停止录制测试脚本。

16. 储存测试脚本
    点选【File】->【Save】或是直接点选工具列上的  按钮，将测试脚本储存成 lesson9。

17. 如果在 Global GUI Map File 模式下，记得储存新的 GUI 对象
    由于 Insert Done…的图像为 WinRunner 新识别的 GUI 对象，所以要记得储存。
    点选【Tools】->【GUI Map Editor】，再点选【View】->【GUI Files】，你可以看到新识别的 GUI 对象是放在 L0<temporary> GUI Map File。点选【File】->【Save】，选取 flight4a.gui，按下【OK】，则新识别的 GUI 对象，将会被储存到 flight4a.gui 中。最后关闭 GUI Map Editor。

9.2 检查文字

接下来你将透过 if/else 验证当新增一笔机票订单后，图表上的票数是否有更新。

1. 在第一行 obj_get_text 指令将 text 变量名称改成 first_total。

2. 在第一行 obj_get_text 指令将 text 变量名称改成 new_total。
3. 将光标移到测试脚本最后一行。
4. 加入以下的测试脚本
   当 new_total=first_total+1 则回报检查点通过，反之则回报检查点失败。

```
if(new_total == first_total + 1)
{
    tl_step("graph total", 0, "Total is correct.");
}
else
{
    tl_step("graph total", 1, "Total is incorrect.");
}
```

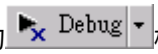
5. 加上批注
   在 if 前加上以下批注：

```
# check that graph total increments by one.
```

6. 储存测试脚本
   点选【File】->【Save】或是直接点选工具列上的  按钮。

9.3 除错

接下来你应该以除错（debug）模式执行测试脚本，检查是否有语法或逻辑上的错误。如果有任何错误讯息，试着去修正问题。

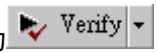
1. 选取 Debug 模式
   选取工具列上的 

模式。
2. 执行测试脚本
   看你的习惯，点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，或是点选工具列上  按钮，一行一行执行完整个测试脚本。
3. 检视测试结果
   当以 Debug 模式执行完测试脚本，执行结果窗口并不会自动开启。选取【Tools】->【Test Results】，或是点选工具列上的  按钮，将会开启测试结果窗口。
4. 关闭测试结果窗口
   在测试结果窗口选取【File】->【Exit】。
5. 关闭 Flight Reservation
   在 Flight Reservation 选取【File】->【Exit】。

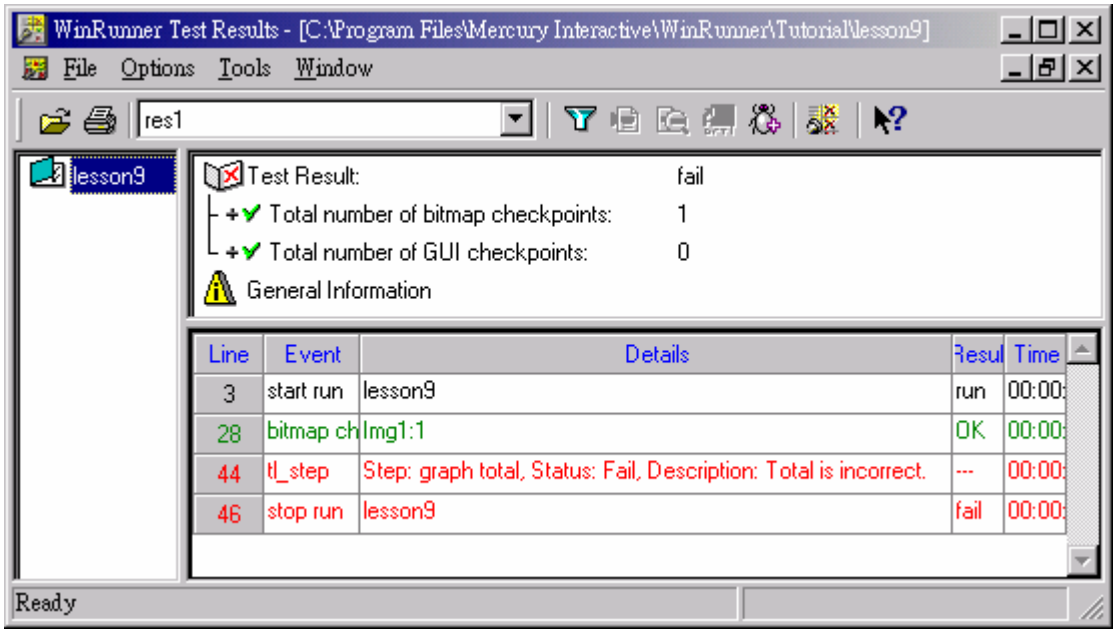
9.4 在另一个版本的 Flight Reservation 执行测试脚本

接下来你将在另一个版本的 Flight Reservation 执行测试脚本。

1. 执行 Flight Reservation 4B
   点选【开始】->【程序集】->【WinRunner】->【Sample Application】->【Flight 4B】，，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。
2. 点选工具列上的执行模式为 

3. 点选 Run From Top
   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则 Run Test 窗口将会开启，接受预设 res1 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。
4. 检视测试结果
   当执行完测试脚本，WinRunner 会自动开启测试结果。
   这时可以看到测试结果是失败的，因为图表上的票数并没有更新。



5. 关闭测试结果窗口
   在测试结果窗口选取【File】->【Exit】。
6. 关闭 Flight Reservation
   在 Flight Reservation 选取【File】->【Exit】。
7. 关闭测试脚本
   在 WinRunner 点选【File】->【Close】。

9.5 建立文字检查点时的建议

1. 在建立文字检查点之前请先确认文字的位置。假如文字是属于标准 GUI 对象的一部分，请使用 GUI 检查点，或是使用 GUI 对象专属的文字函数如 edit_get_text 或 button_get_text 等。假如文字是属于非标准 GUI 对象的一部分，请使用【Insert】->【Get Text】->【From Object/Window】建立文字检查点。假如文字是属于图像的一部分，请使用【Insert】->【Get Text】->【From Screen Area】建立文字检查点。
2. 当建立文字检查点时，WinRunner 会将当时撷取到的文字以批注放在测试脚本的后方，假如批注出现 `#no text was found`，表示 WinRunner 撷取不到应用程序上面的字型，此时可能需要使用 Font Expert 教 WinRunner 识别应用程序上的字型。
3. WinRunner 测试脚本语言（TSL）另外提供一些文字相关的函数，如 win_find_text、obj_find_text、compare_text，详细使用说明请参考 *WinRunner User's Guide*。

10. 建立批次（batch）测试

课程摘要：

- 说明如何使用批次测试执行一整组的测试脚本
- 如何建立批次测试
- 执行批次测试并检视结果

10.1 何谓批次（batch）测试

想象一下这样的情境，你刚刚变更了你的应用程序，然后你想要在新版的应用程序上执行所有的测试脚本。你不需要一个一个单独的执行测试脚本，你只需要执行一个批次测试，然后去吃午餐，吃完午餐回来，屏幕上已经显示所有测试脚本的测试结果。

批次测试脚本看起来与一般的测试脚本没什么不一样，不过批次测试脚本还是与一般测试脚本有二个不同的地方：

- 批次测试脚本含有 call 指令，用来开启其它测试脚本，例如：

```
call "c:\\qa\flights\\lesson9"();
```

  当批次测试执行时，WinRunner 一执行到 call 指令，便会开启并执行指定的测试脚本，当被呼叫的测试脚本执行完毕，WinRunner 便会回到批次测试继续执行下去。

- 在执行批次测试之前，你会先选取 Tools->General Options，在点选 Run 后勾选 Run in batch mode 选项。这个选项会让 WinRunner 不再跳出讯息对话窗口而中断测试的执行。例如当一个图像检查点失败时，WinRunner 不会再暂停测试执行并显示 mismatch 的讯息了。

当你检视测试结果时，你可以看到整个批次测试的测试结果是通过或失败，也可以看到所有被批次测试呼叫的测试，其结果是通过或失败。

10.2 建立批次测试

接下来你会建立一个批次测试：

- 呼叫你之前建立的测试脚本（lesson5、lesson6、lesson7）
- 执行每个被呼叫的测试脚本三次

1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 加上 call 指令呼叫其它测试脚本

   在新开启的测试脚本中输入以下的脚本：

```
call "c:\\qa\flights\\lesson5"();
call "c:\\qa\flights\\lesson6"();
call "c:\\qa\flights\\lesson7"();
```

   在你的测试脚本中，请将 c:\\qa\\flights 换成你的测试脚本存放的路径。

   请注意，在 WinRunner 的测试脚本中用双斜线「\\」取代一般档案路径的斜线「\」。

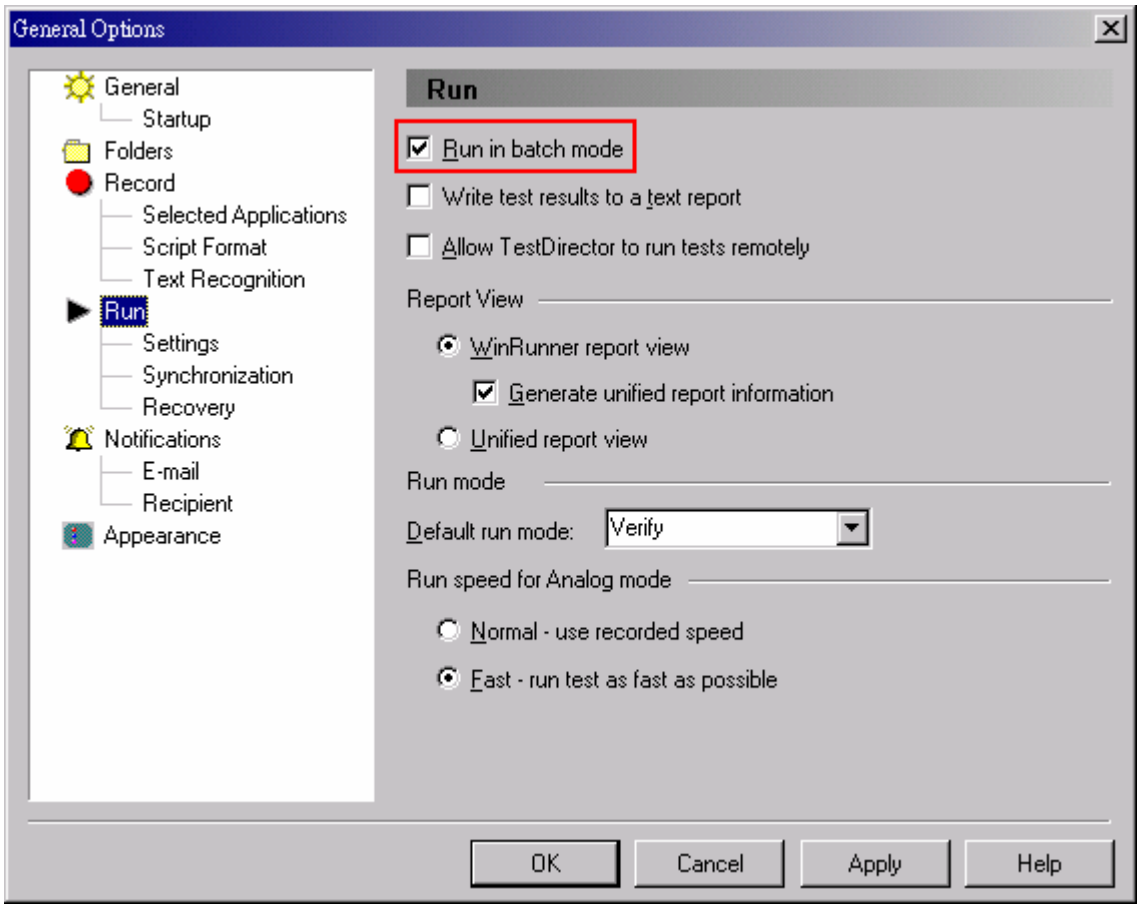
3. 加上 loop 循环

   为了执行三次所有被呼叫的测试脚本，请加上以下的 loop 循环：

```
for(i=0; i<3; i++)
{
    call "c:\\qa\flights\\lesson5"();
    call "c:\\qa\flights\\lesson6"();
    call "c:\\qa\flights\\lesson7"();
}
```

4. 设定在以批次模式执行

   点选【Tools】->【General Options】->【Run】，勾选【Run in batch mode】选项，然后按下【OK】按钮。



5. 储存批次测试脚本

   点选【File】->【Save】或是直接点选工具列上的  按钮，将测试脚本储存成 batch。

10.3 在另一个版本的 Flight Reservation 执行批次测试

接下来你将在另一个版本的 Flight Reservation 执行批次测试脚本。

1. 开启 Flight Reservation 4B 版

   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4B】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

2. 开启 WinRunner 并加载 lesson6 测试脚本

   开启 WinRunner，点选【File】->【Open】开启 lesson6 测试脚本。

3. 确认工具列上显示 

模式

4. 点选 Run From Top

   点选【Test】->【Run From Top】或是直接点选工具列上的 

按钮，则 Run Test 窗口将会开启，接受预设 res1 的执行名称，确认已勾选【Display test results at the end of run】，按下【OK】开始执行测试。

   注意观察 WinRunner 如何执行被叫的测试脚本。

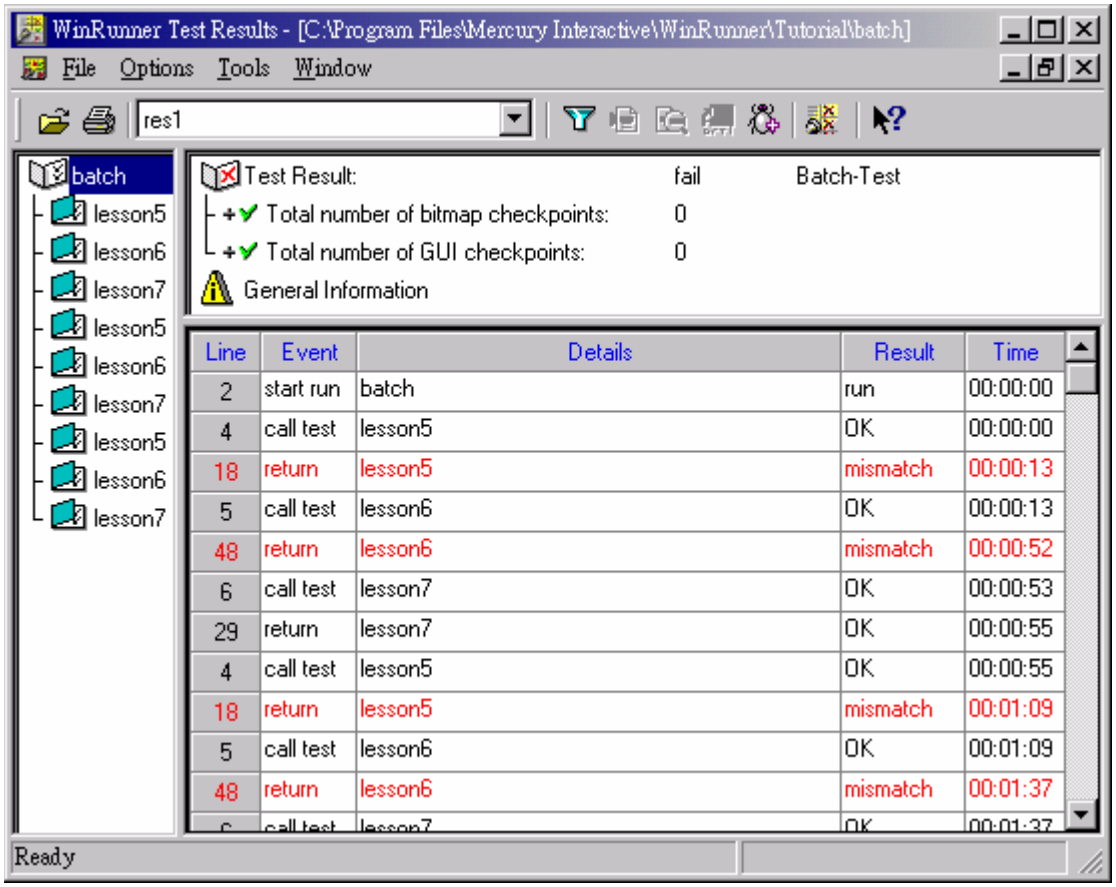
10.4 检视批次测试的结果

接下来你将检视批次测试的结果。

1. 开启测试结果窗口

   假如 WinRunner 没有自动开启测试结果，选取【Tools】->【Test Results】，或是点选工具列上的  按钮，将会开启测试结果窗口。

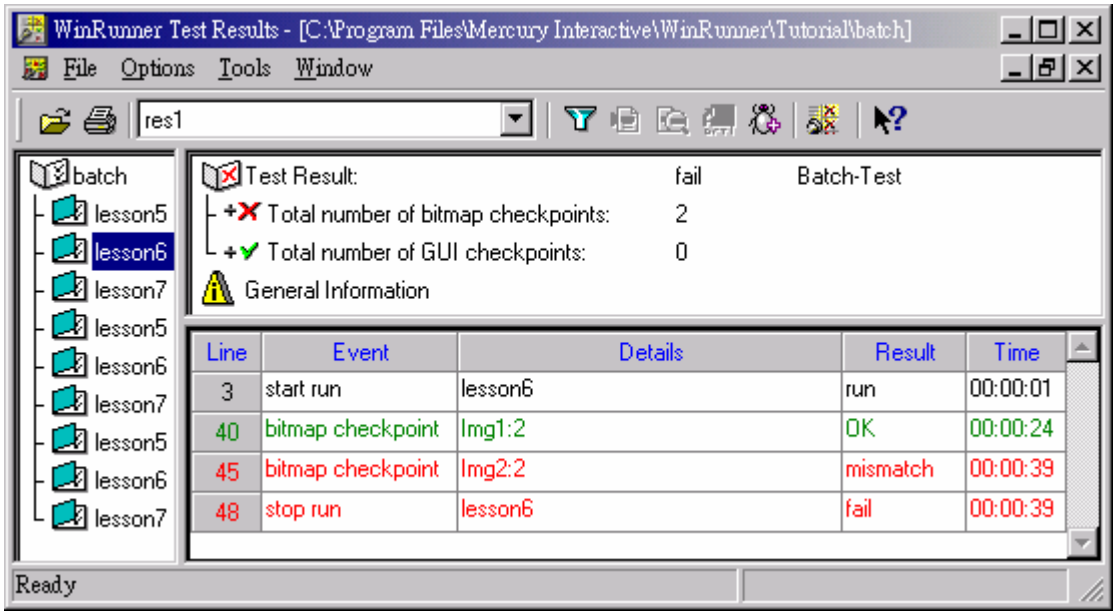
2. 检视批次测试的结果



由于是以 Flight 4B 执行测试，所以测试结果会失败。

3. 检视被呼叫测试脚本的测试结果

你可以点选一个测试脚本，以便进一步检视其测试结果，你可以清楚看出 lesson6 测试失败是因为图像检查点不一致。

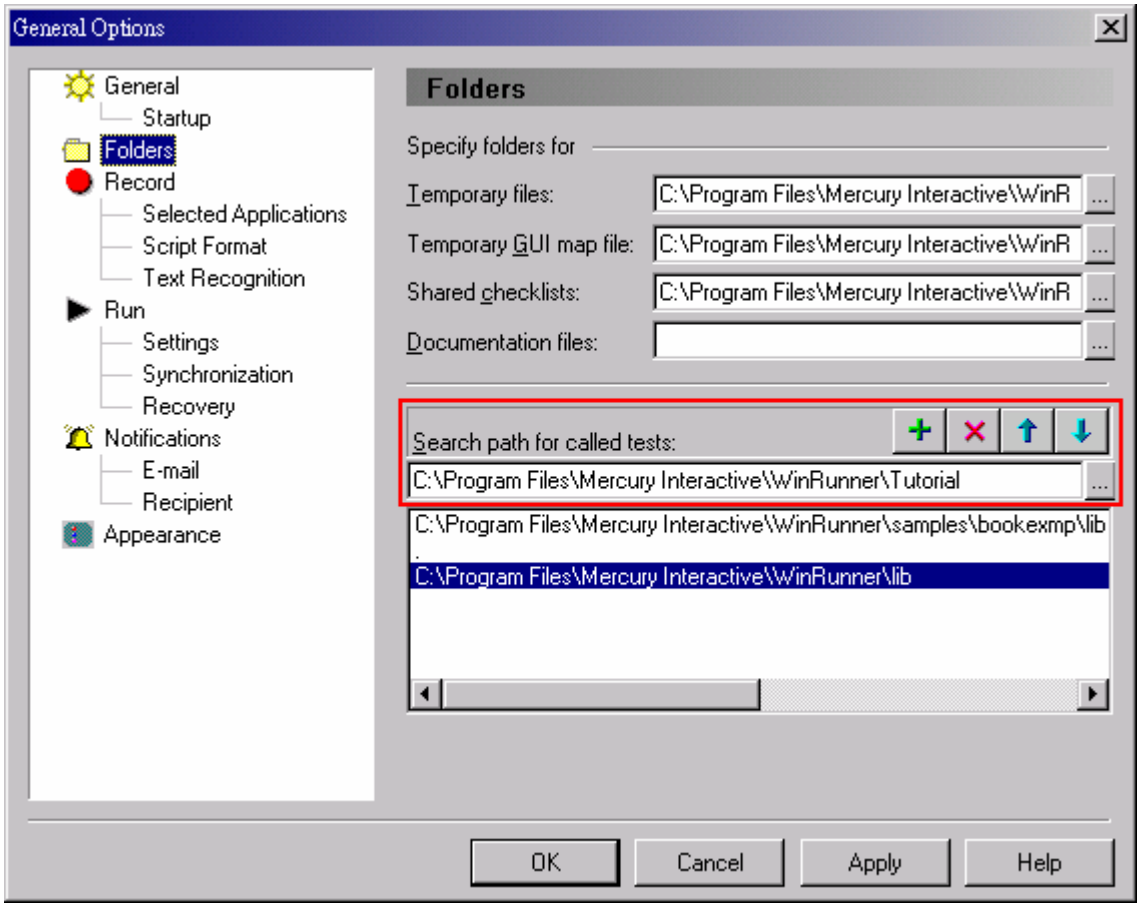


4. 关闭测试结果
   选取【File】->【Exit】关闭测试结果窗口。
5. 关闭 Flight 4B
   选取【File】->【Exit】关闭 Flight Reservation 范例程序。
6. 关闭 batch 测试脚本
   选取【File】->【Close】关闭测试脚本，或是选取【File】->【Close All】。
7. 清除以批次模式执行的设定
   点选【Tools】->【General Options】->【Run】，清除勾选【Run in batch mode】选项，然后按下【OK】按钮。

10.5 建立批次测试脚本时的建议

1. 你可以设定测试脚本的搜寻路径，则 WinRunner 会自动到设定的路径下搜寻被呼叫的测试脚本，如此一来你在批次测试中呼叫其它测试脚本时，就不需要输入完整的测试脚本的路径，而只要输入测试脚本名称就可以了。

   选取【Tools】->【General Options】->【Folders】，在【Search path for called tests:】中输入测试脚本的搜寻路径，然后按下  按钮，即完成设定测试脚本的搜寻路径。



   设定完成后，你在批次测试脚本中呼叫其它测试脚本时，就不需要输入完整的路径，而只要输入测试脚本名称就可以了。

```
call "lesson6"();
```

2. 请记得在【Tools】->【General Options】->【Run】，勾选【Run in batch mode】选项，否则当批次脚本执行过程中，假如有任何错误发生，将导致测试执行中断。

11. 维护你的测试脚本

课程摘要：

- 说明当应用程序的使用者接口改变时，如何透过 GUI Map 让测试脚本可以继续使用
- 如何些改 GUI Map
- 如何使用执行精灵（Run wizard）自动更新 GUI Map

11.1 当使用者接口改变时

想象一下这样的情境：你过去花了几个礼拜的时间，建立一整套涵盖应用程序所有功能的测试脚本。然后开发团队为了改善使用者接口，所以修改了一些 GUI 对象，也新增了一些 GUI 对象，甚至删除了一些 GUI 对象，并且发行了新版本。而你要如何用既有的测试脚本来测试新版的应用程序呢？

面对这样的情境，WinRunner 提供了一个非常方便的解决方案：GUI Map。透过更新 GUI Map，WinRunner 就可以识别这些被新增、修改的 GUI 对象，你就不需要手动去修改既有的测试脚本了。

在 GUI Map 中记录了应用程序中 GUI 对象的描述（descriptions），其内容由以下二部份所组成：

- 逻辑名称（logic name）：一个简短且直觉的名称，用来代表 GUI 对象，你可以看到这个名称出现在测试脚本中，例如

  ```
  button_press("Insert Order");
  ```

  「Insert Order」就是某个 GUI 对象的逻辑名称。

- 实体描述（phsical description）：一组可用来唯一识别 GUI 对象的属性，例如

  ```
  {
   class: push_button,
   label: "Insert Order"
  }
  ```

  表示这个 GUI 对象是属于「push_button」类别，也就是一个按钮，且按钮上的卷标（label）为「Insert Order」

在执行测试脚本时，当 WinRunner 读取到一个 GUI 对象的逻辑名称后，WinRunner 会到 GUI Map 中寻找这个 GUI 对象实体描述，然后以这些属性，在应用程序上找到拥有这些属性的 GUI 对象。

所以当应用程序上的 GUI 对象有变更，你就必须在 GUI Map 中修改此 GUI 对象的实体描述，如此一来，WinRunner 就可以识别此 GUI 对象了。

接下来的课程你会

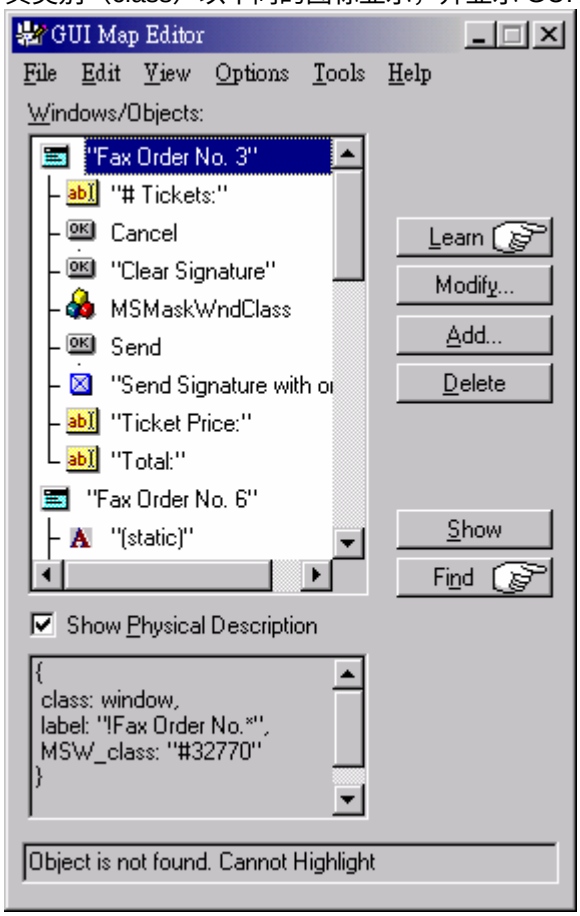
- 在 GUI Map 中编辑 GUI 对象的属性
- 新增一个 GUI 物件到 GUI Map 中
- 使用执行精灵（Run wizard）自动侦测使用者界面的变动，并自动更新 GUI Map

11.2 在 GUI Map 中编辑 GUI 对象的属性

假设在新版本的 Flight Reservation 应用程序中，原本的「Insert Order」按钮已经修改成「Insert」按钮，为了要让有用到「Insert Order」按钮的测试脚本可以继续被使用，你必须在 GUI Map 中修改「Insert Order」按钮的卷标。

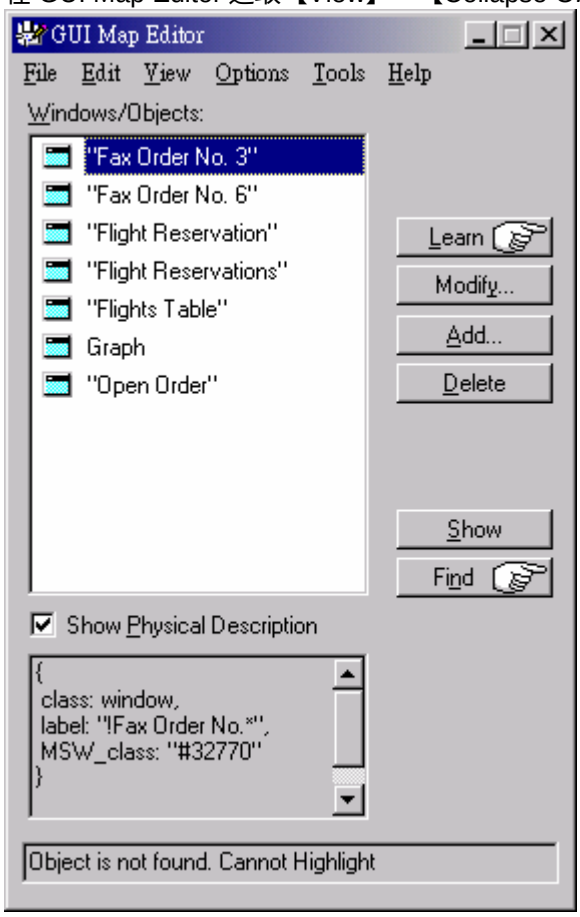
1. 开启 WinRunner 并加载 GUI Map File
   执行【开始】->【程序集】->【WinRunner】->【WinRunner】，如果是第一次执行 WinRunner，会开启欢迎窗口，则点选【New Test】；如果没有开启欢迎窗口，则点选【File】->【New】。
   检查 GUI Map File 是否已经加载，点选【Tools】->【GUI Map Editor】开启 GUI Map Editor，再点选【View】->【GUI Files】检查是否加载 flight4a.gui。如果 flight4a.gui 没有加载，点选【File】->【Open】然后选取 flight4a.gui 后，按下【Open】将其载入。

2. 开启 GUI Map Editor
   选取【Tools】->【GUI Map Editor】，开启 GUI Map Editor。
   在 GUI Map Editor 选取【View】->【GUI Map】，则【Windows/Objects】清单会以阶层方式列出目前 GUI Map 的内容，每个 GUI 对象都在其所隶属的窗口之下，且每个 GUI 对象会根据其类别（class）以不同的图标显示，并显示 GUI 对象的逻辑名称。



3. 找到「Insert Order」按钮

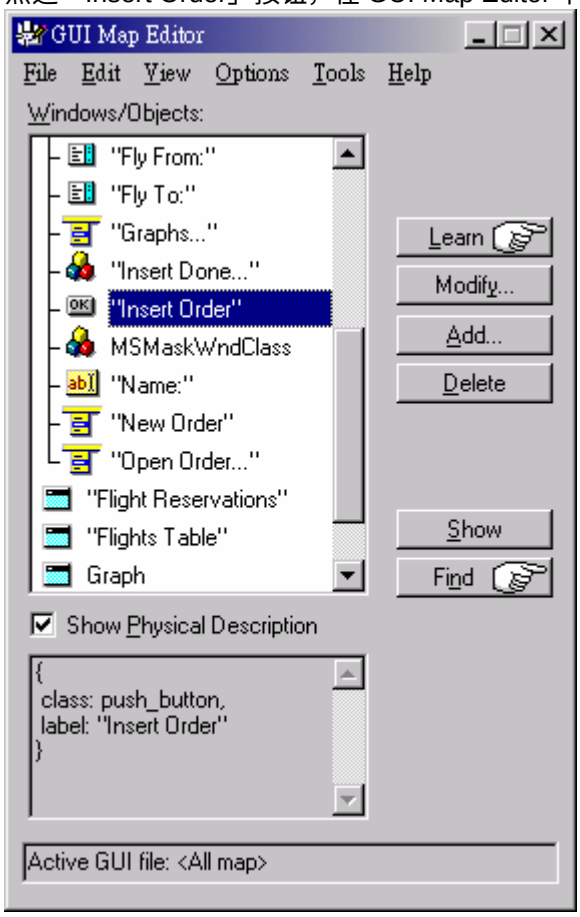
在 GUI Map Editor 选取【View】->【Collapse Objects Tree】，以便只检视窗口。



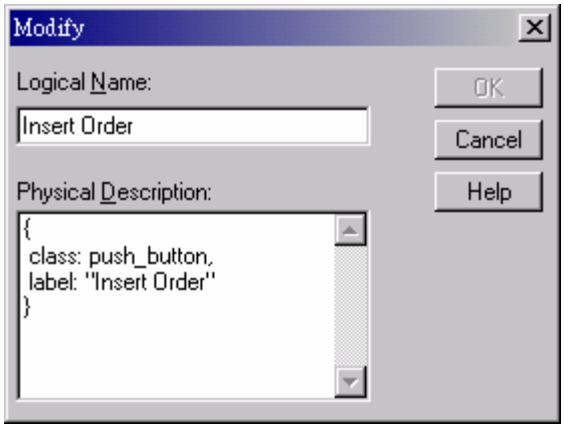
对「Flight Reservation」窗口点二下，「Flight Reservation」窗口会展开并显示属于「Flight Reservation」窗口的所有 GUI 对象。找到「Insert Order」按钮。

4. 检视「Insert Order」按钮的实体描述

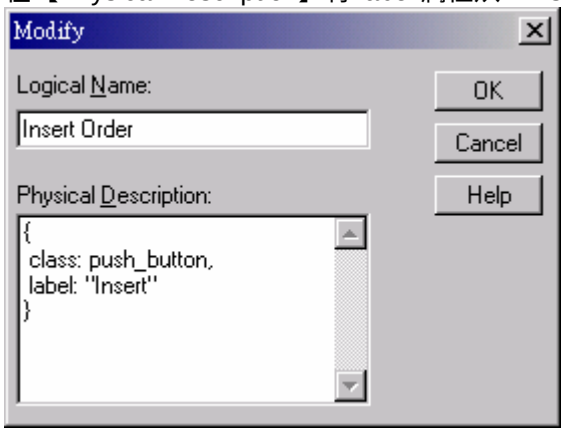
   点选「Insert Order」按钮，在 GUI Map Editor 下方会显示「Insert Order」按钮的实体描述。



5. 修改「Insert Order」按钮的实体描述
   点选【Modify】按钮或试在「Insert Order」按钮点二下，会开启 Modify 窗口，并显示「Insert Order」按钮的逻辑名称与实体描述。



   在【Physical Description】将 label 属性从「Insert Order」改成「Insert」。



   按下【OK】按钮，储存修改。

6. 关闭 GUI Map Editor
   先选取【File】->【Save】储存 GUI Map。然后点选【File】->【Exit】关闭 GUI Map Editor。

11.3 新增 GUI 物件到 GUI Map

当应用程序新增 GUI 对象时，你只要以 GUI Map Editor 的学习（learn）功能，就可以将新增的 GUI 对象加到 GUI Map 中，不需要再执行 RapidTest Wizard。GUI Map Editor 可以一次学习一个 GUI 对象或是个窗口中的所有 GUI 对象。

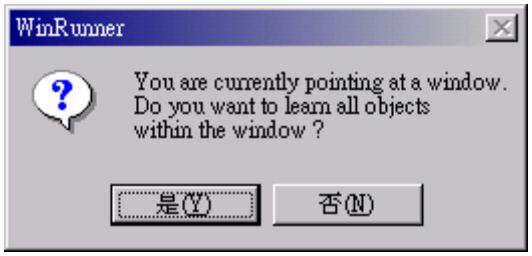
接下来你将把 Flight Reservation 登入窗口学习到 GUI Map 中。

1. 开启 Flight Reservation 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】。

2. 开启 GUI Map Editor
   在 WinRunner 选取【Tools】->【GUI Map Editor】，当 GUI Map Editor 开启后选取【View】->【GUI Files】。

3. 学习登入窗口的所有 GUI 对象

   点选【Learn】按钮，此时鼠标光标会变成，点选登入窗口的标题列，则 WinRunner 会跳出一个讯息窗口，询问你是否要学习窗口中所有的 GUI 对象。



   点选【Yes】后，注意看 WinRunner 如何将窗口中的 GUI 对象一个一个学习下来。

4. 储存 GUI Map

   选取【File】->【Save】储存 GUI Map，点选【OK】将新的窗口与 GUI 对象储存到 flight4a.gui 中。

5. 关闭 GUI Map Editor

   点选【File】->【Exit】关闭 GUI Map Editor。

6. 关闭登入窗口

   在登入窗口中点选【Cancel】按钮。

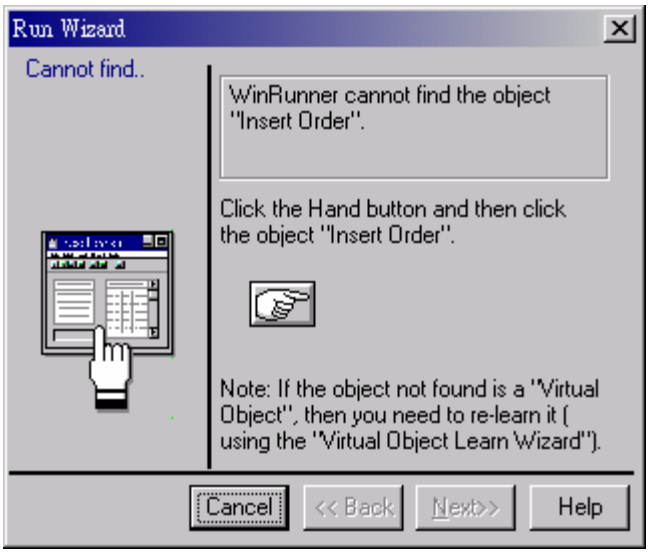
11.4 使用执行精灵（Run wizard）自动更新 GUI Map

在测试执行过程中，假如 WinRunner 在应用程序上无法找到测试脚本中所使用的 GUI 对象，就会自动开启执行精灵（Run wizard）。透过执行精灵（Run wizard），你可以让 WinRunne 自动重新识别找不到的 GUI 对象或是新增 GUI 对象。

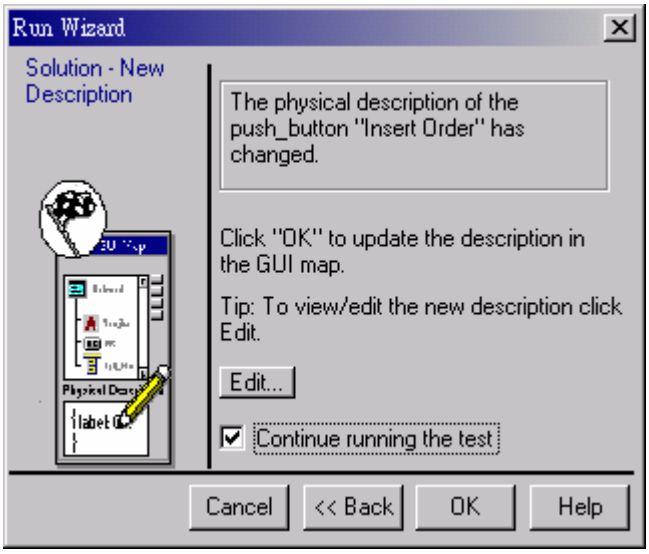
举例来说，当 WinRunner 在 Flight Reservation 执行到点选【Insert Order】按钮的测试脚本，

```
button_press("Insert Order");
```

假设这个按钮的卷标（label）已经从【Insert Order】改成【Insert】。这时执行精灵（Run wizard）会自动开启，并提醒你 WinRunner 找不到【Insert Order】按钮。



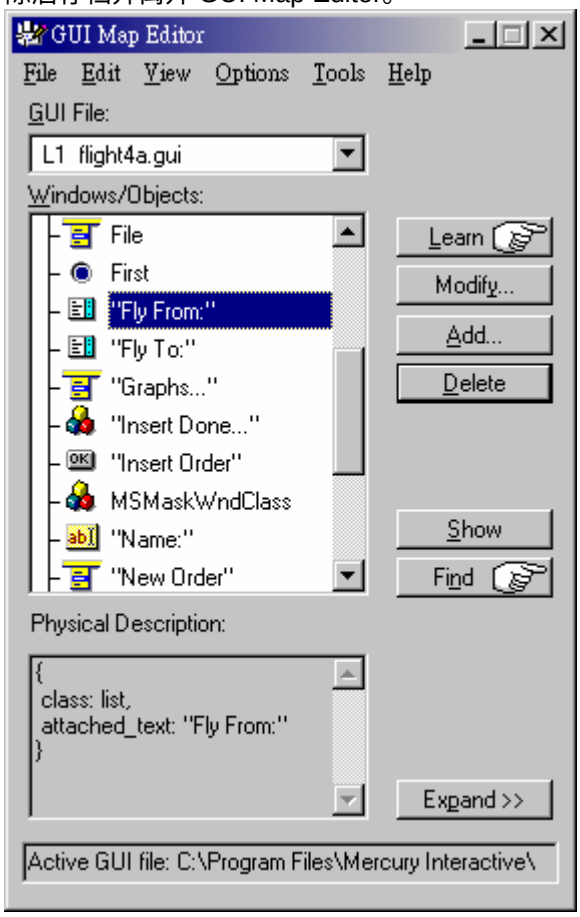
然后你将点选  按钮，并重新点选【Insert】按钮。



这时执行精灵（Run wizard）会建议解决方案，按下【OK】按钮，则执行精灵（Run wizard）会自动更新【Insert Order】按钮的实体描述，并且从中断的地方继续执行下去。

1. 开启 GUI Map Editor
   在 WinRunner 选取【Tools】->【GUI Map Editor】，当 GUI Map Editor 开启后选取【View】->【GUI Files】，并在【GUI Files】中选取 flight4a.gui。

2. 从 GUI Map Editor 中删除【Fly From】清单对象
   选取位于【Flight Reservation】窗口下的【Fly From】清单对象，并按下【Delete】按钮。删除后存档并离开 GUI Map Editor。



3. 开启 Flight 4A 并登入
   执行【开始】->【程序集】->【WinRunner】->【Sample Applications】->【Flight 4A】，登入窗口会开启。在【Agent Name】输入名字，至少四个英文字母，【Password】输入 mercury，按下【OK】按钮登入 Flight Reservation。

4. 开启 lesson4 测试脚本并执行
   注意当 WinRunner 执行到下面这一行测试脚本时会发生什么事。

```
list_select_item ("Fly From:", "Los Angeles");
```

5. 依照执行精灵（Run wizard）的指示将【Fly From】清单对象加到 GUI Map
   由于【Fly From】清单对象已经被删除，所以执行精灵（Run wizard）会开启，同样点选  按钮，并重新点选【Fly From】清单对象，然后按下【OK】按钮，则执行精灵（Run wizard）会自动将【Fly From】清单对象加到 GUI Map 中。
   另外由于之前变更了【Insert Order】按钮的实体描述，同样也使用执行精灵（Run wizard）将【Insert Order】按钮的实体描述改回来，则 WinRunner 继续完成测试脚本的执行。

6. 储存 GUI Map
   选取【File】->【Save】储存 GUI Map，点选【OK】将新的窗口与 GUI 对象储存到 flight4a.gui 中。
7. 关闭 GUI Map Editor
   点选【File】->【Exit】关闭 GUI Map Editor。
8. 关闭 Flight Reservation
   选取【File】->【Exit】关闭 Flight Reservation。

12. 从这里出发

恭喜！你已经完成以上的 11 个课程，现在你已经准备好运用你学习到的知识与技巧，测试你的应用程序了！

12.1 获得更多信息

你可以从以下地方获取更多关于使用 WinRunner 与 TSL 得信息：

手册

- WinRunner User's Guide
- WinRunner Installation Guide
- WinRunner Customization Guide
- TSL Reference Guide

在线说明

- Read Me：【开始】->【程序集】->【WinRunner】->【Read Me】
- What's New in WinRunner：在 WinRunner 中选取【Help】->【What's New in WinRunner】
- Books Online：在 WinRunner 中选取【Help】->【Books Online】（PDF 格式）
- WinRunner Context Sensitive Help：
- TSL Online Reference：在 WinRunner 中选取【Help】->【TSL Online Reference】
- Sample Tests：Mercury Interactive Customer Support Web site
- Technical Support Online：Mercury Interactive Customer Support Web site
- Mercury Interactive Web Site：http://www.mercuryinteractive.com/