TIB 自动化测试工作室
http://www.cnblogs.com/testware/

TestComplete Tutorials（Testcomplete 自学教程）

Testcomplete 包含了一系列的教程来帮助你熟悉 testcomplete 和学习相关的测试技术。每一个教程都详细的步骤来指引你轻松完成整个项目的创建，涵括了各种类型的测试场景。

Getting Started Tutorial（快速入门教程）

这份教程的对象是初学者。它会教你如何让在 testcomplete 首次创建简单的功能测试。在你学习完这节教程之后，你会掌握录制、修改、回放测试的技术。
这份教程包括如下两部分：
1.简要介绍自动化测试和 testcomplete
2.创建你的第一个自动化测试：按照教程的指引步骤在 testcomplete 中创建你的第一个测试项目。

Introducing Automated Testing and TestComplete（自动化测试与 Testcomplete 简介）

这一节的主题是给出自动化测试和 testcomplete 的一个概况，包括：
自动化测试；
测试类型；
Testcomplete 项目和项目内容（Projects and Project Items）；
Testcomplete 的用户界面（User Interface）；
Testcomplete 的测试对象模型（Test Object Model）；
检查点和数据存储（Checkpoints and Stores）；

Automated Testing（自动化测试）

软件测试是一个检查应用程序并从中发现错误的一个过程。测试中需要比较应用程序的实际输出与预期输出是否一致，才可以下结论说应用程序是否实现了它应有的功能——这就是软件测试和试用软件的根本区别。换句话说，测试员不仅仅要保证应用程序显示了一系列的值，还要证实显示的值是正确的。

所以，测试的基本步骤如下：
定义预期输出；
执行测试（输入合适的值）；
收集程序的输出并与预期输出（基线数据）作出比较；
如果比较失败，要告知开发人员或者经理。

自动化测试是一种用特定的程序（无人干预或者少量的人工干预）自动执行软件测试。自动化执行保证不会跳过每一个测试动作；它使测试人员从重复执行沉闷的测试步骤中解放出来。

Testcomplete 提供了一些专为自动化测试动作、定义基线数据、运行测试和记录测试结果日志的功能。它还提供了专门的对话框和提示来帮助你在测试中自动化比较命令（或者检查点）。

Test Types（测试的种类）

Testcomplete 支持多种测试类型和方法：单元测试、功能和图形界面测试、回归测试、分布式测试等。在这份教程里，我们将创建一个我们最常用到的测试——功能测试。功能测试会检查应用程序与其他系统、用户之间的接口。它们将验证应用程序的功能与预期是否相同。

一个典型的功能测试由一系列的测试指令组成，这些测试命令形如模拟用户的鼠标点击和键盘输入。循环运行这些测试指令来验证被测软件的功能。

在 Testcomplete 里，能够以关键字测试（Keyword tests）或者脚本（Script）来的形式创建功能测试。这两种类型的测试都可以用内置的编辑器来录制（recordered）或者从零开始创建。创建关键字测试是可视化的，非常简单且不需要任何编程基础。脚本方式创建的测试则需要理解相关脚本的命令，但你能够从中创建更强大、更灵活的测试。Testcomplete 支持的脚本包括 VBScript, JScript, DelphiScript, C++Script and C#Script，所以你可以从中挑选你最熟悉的脚本语言来使用。
在这份教程里，我们会使用关键字测试这一功能。

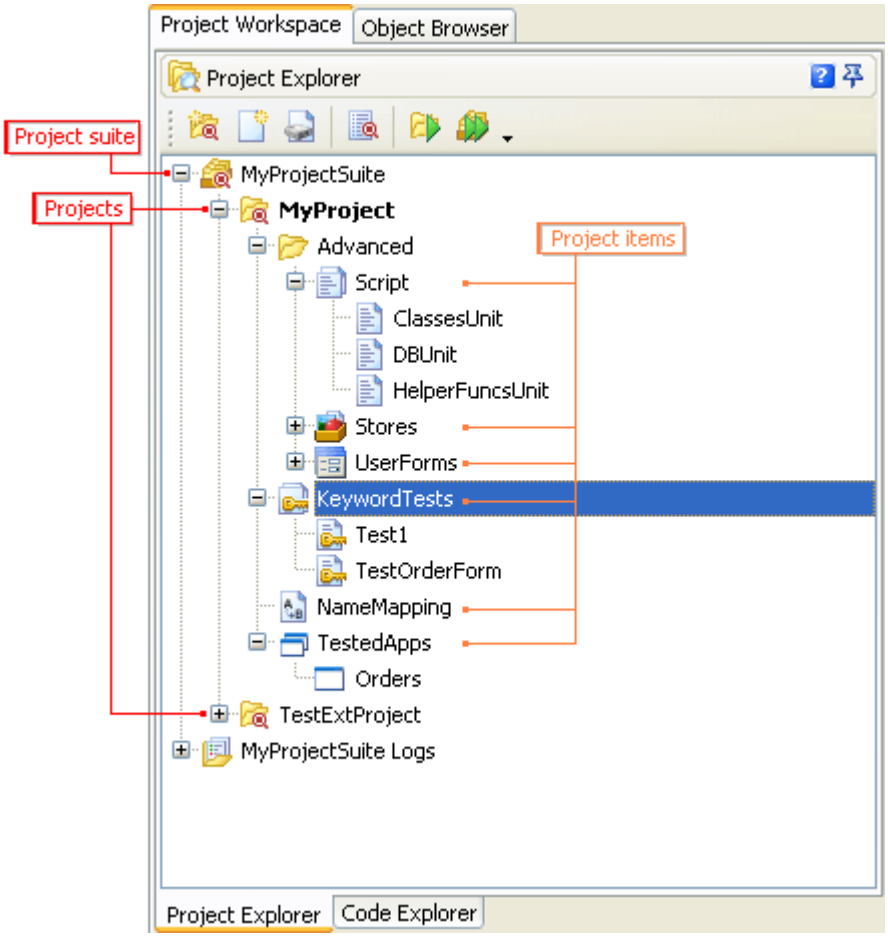
TestComplete Projects and Project Items（Testcomplete 的项目与项目组）

Testcomplete 用项目（Projects）和项目组（project suites）来管理。项目创建测试的开端。它包括了你的所有测试、检查点的基线数据、被测程序的相关信息和其在测试执行中需要用到的各项内容。项目中也定义了多个测试之间的执行次序，和项目累计执行的测试日志。

一个项目可以包括对被测程序的所有测试。在复杂的被测程序中，你可以仅将一个项目专注被测程序的一部分，而让其他项目关注被测程序的另外的部分（通常是各个不同的模块）。

相关的多个项目（Projects）可以组织成一个项目组（project suites）。Testcomplete 会在你创建一个新项目的时候自动创建一个项目组。你也可以创建一个空的项目组，然后利用 Testcomplete 的对话框来对项目组添加相应的项目。

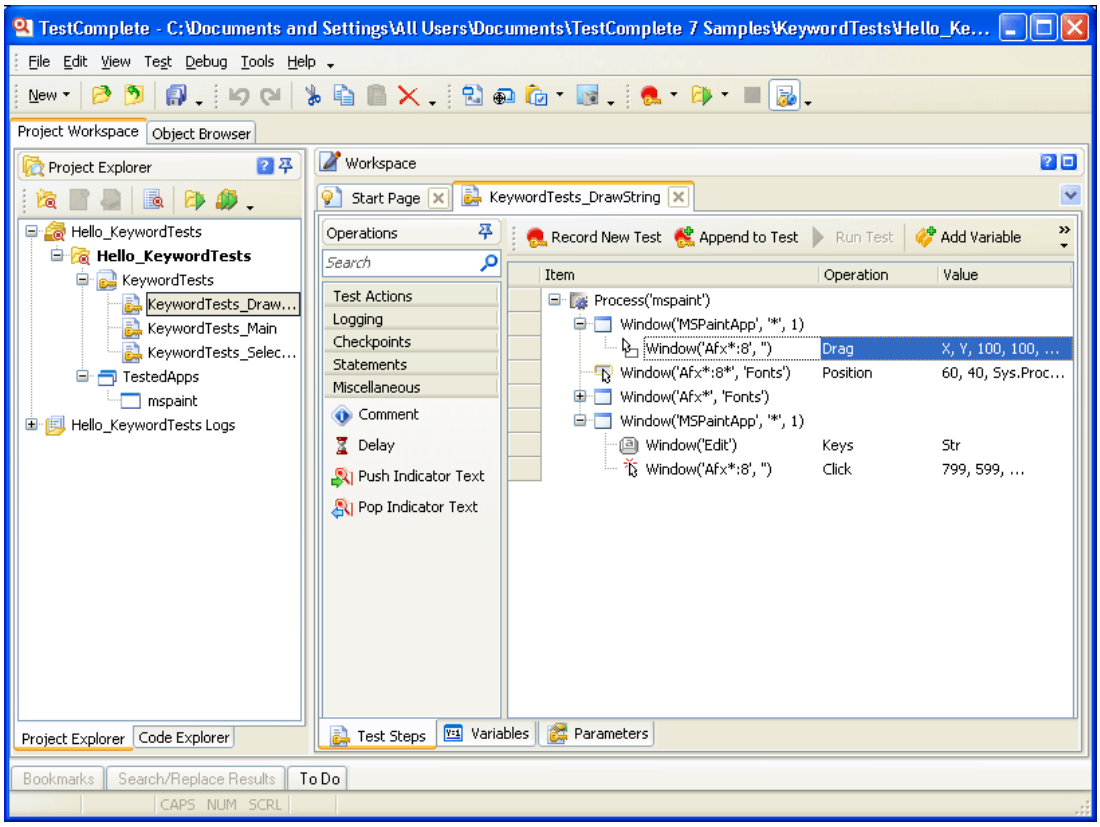
项目项（Project items）是用来执行或者支援测试的项目元素。你可以在Testcomplete里查看和管理项目、项目组、项目项，如下图所示：



如果项查看关于项目组的完整信息，请查看About Project Items.

TestComplete User Interface（Testcomplete 的用户界面）

Testcomplete 的主界面如下图所示：



如你所见，Testcomplete的用户界面组织在一系列的控制面板上。位于左边的项目浏览器面板（Project Explorer）显示了项目和项目组的内容。它也提供了可链接到测试日志的节点。

工作区面板（The Workspace panel）是你的工作台：它显示了项目和项目项的编辑器，你可以在上面创建或者修改测试、查看测试结果。例如，在上图中你可以看到关键字测试编辑器在工作区中处于打开状态。

除了项目浏览器和工作区，Testcomplete还提供了其他控制面板。例如，Watch List, Locals, Breakpoints 和 Call Stack面板供调试所用。To Do面板把将要执行的测试管理起来，Code Explorer面板则提供了浏览脚本内容、导航脚本单元的的快捷方式。

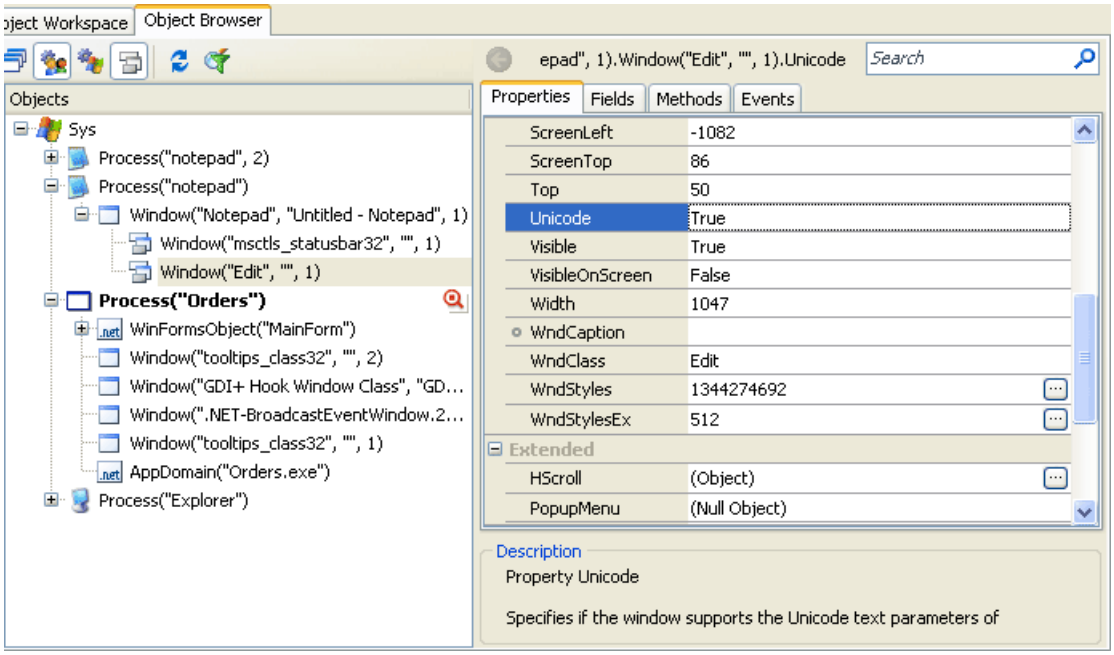
Object Browser 面板显示了 Testcomplete 中与项目无关的功能：它以列表的形式显示了当前机器上的所有进程和窗口。若对应进程和窗口的属性和方法能够被 Testcomplete 内部识别出来，Object Browser 面板就会显示。换句话说，Object Browser 能告诉你那些对象、方法和属性是可测的，和怎样取来测。

如果想跟深入了解面板的使用，在里面单击一下，然后点击 F1，就能打开对应面板的描述信息。

你可以使用菜单和工具栏来执行相应的功能。它的菜单、工具栏系统和 Microsoft Visual Studio、流行的 windows 应用程序非常类似。你可以改变工具栏的位置，把里面的内容转移到其他菜单或者工具栏中、隐藏内容、把隐藏的内容还原等等。请参见 Toolbars Customization。

TestComplete Test Object Model（Testcomplete 的测试对象模型）

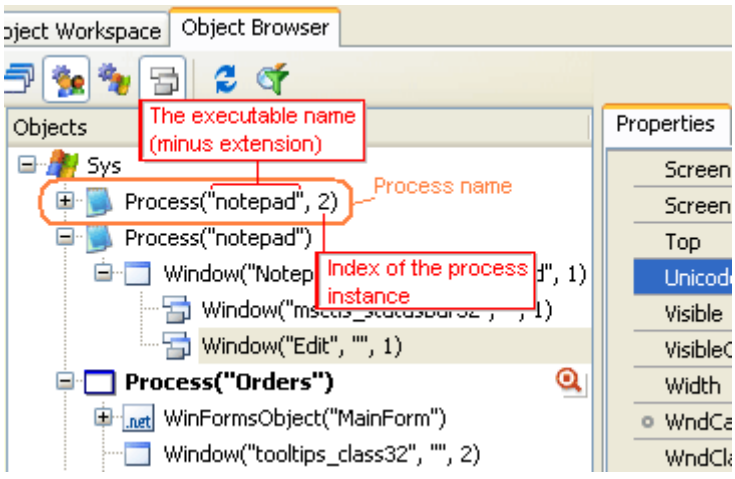
对象浏览器面板显示了对象的结构，如下图示：



Testcomplete 使用了树形模型来组织被测对象。根结点是 Sys（桌面应用程序和窗口）和 PDA（一些运行在连接你计算机的 WindowsMobile 设备的程序）

Process 对象相当于运行在操作系统上的应用程序。我们用术语 Process（进程）而不是 application（应用程序），这是因为这等价于 windows 文档里有关进程的概念。

进程对象的名称包括正在执行的进程的名称和它对应的索引（index）值。注：index 只使用在多个应用程序执行的环境下：

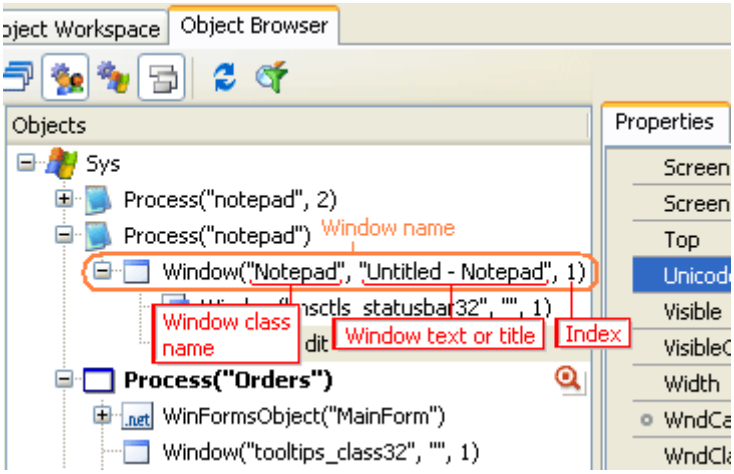


进程（Processes）有他们的子对象（窗口），也就说是（子对象的）顶层窗口。这些对象都拥有它们对应控件的子窗口。这些窗口和控件的命名取决与测试引擎是否能够识别出被测应

用程序的内部方法和属性。Testcomplete 支持上述的两种类型，但用不同的方法来命名它们的各总窗口的控件。

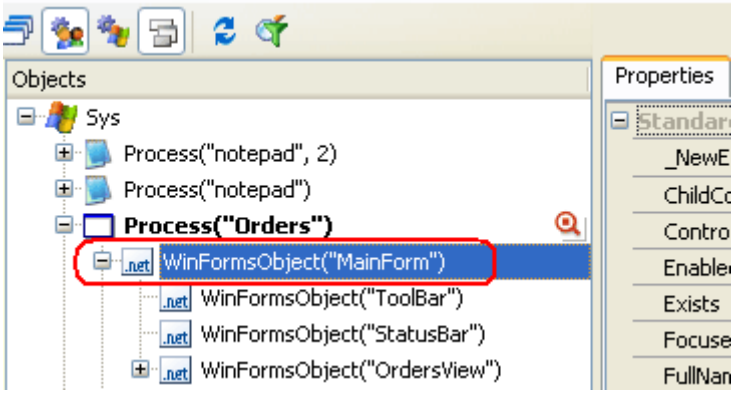
黑合应用程序：

黑合应用程序指的是不提供访问它们内部方法和属性的应用程序。它们的命名包括 window's class name, window's text 或 title (caption) ，和它的 index。控件的命名方式和窗口的命名方式类似，因为就操作系统而言，控件只是窗口的类型之一：



白盒应用程序：

那种向 Testcomplete 提供其内部属性和方法的应用程序叫做白盒应用程序或者开合应用程序。它们用作标示，显示在对象浏览器上。为了突出白盒应用程序的窗口和控件，Testcomplete 使用了特别的命名方式，可以反映出控件或窗体的类型、在源码中定义的名称。例如，你有个用 C#调用 Microsoft WinForms 库生成的名为 MainForm 的窗体，Testcomplete 将以这种形式来识别：`WinFormsObject("MainForm")`



注：强烈建议在有条件的情况下选择白盒应用程序来测试，而不是黑合应用程序。这使得测试引擎能够识别出被测程序的内部方法和属性，使你的测试更加有效灵活。

有些程序像.NET, WPF, Visual Basic, Java 或者 Web 对 Testcomplete 都是白盒应用程序，而其他的可能要通过一些特别的手段编译才行。

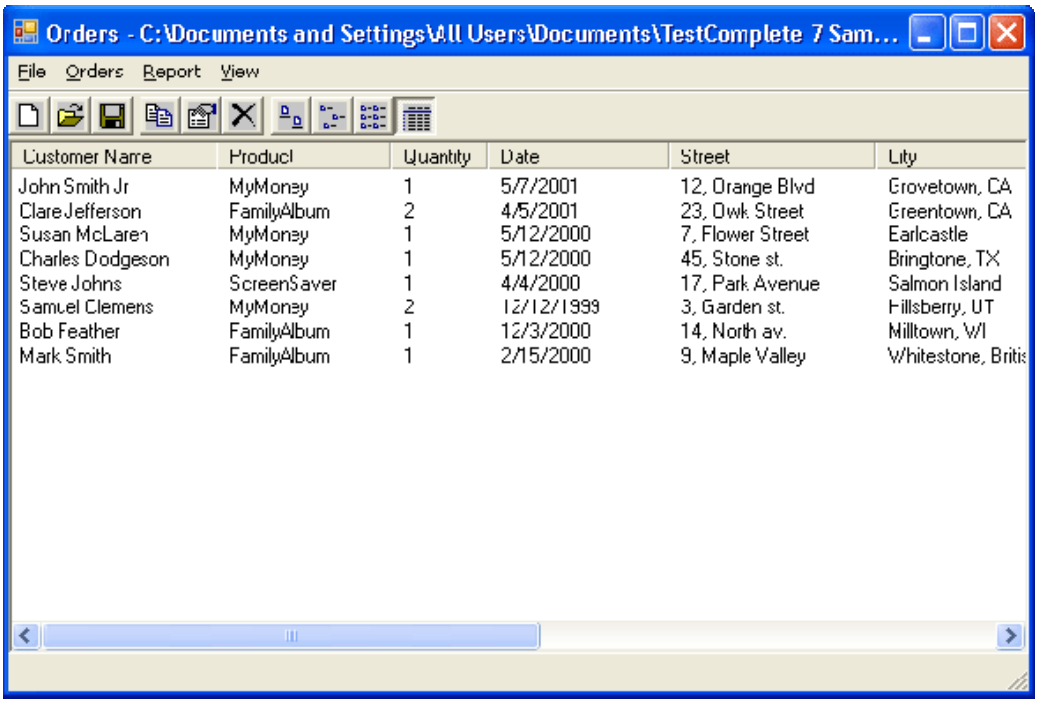
Checkpoints and Stores（检查点和数据存储）

一个典型的测试中会进行多次的比较。例如，一个模拟用户输出应用程序的文件的测试场景，你需要去验证输出的数据是否有效。为了完成这一测试，你需要将输出的文件和预期的文件相比较。这仅仅是一个例子，在真实的测试中，你要作成百上千次比较。各种类型的自动化测试（回归、单元、功能等等）都需要一个参照值来对比验证。

在 Testcomplete 下可以很轻松地在测试中加入比较命令（或者检查点）。你可以在录制测试和设计时加入检查点。Testcomplete 提供比较不同类型数据的检查点：图像、文件、对象文本和属性（Object text and properties）、xml 文档、数据库表（database tables）等。Testcomplete 包含了数据存储（**Stores**）这个项目项（project item），可用来存放检查点的基线数据。这些项目项是一个容器，存放了图像、文件和其他用于测试比较的元素。唯一例外的是用于验证对象属性的检查点：这**些基线数据是在测试中定义的。**

Creating Your First Test（创建你的第一个自动化测试）

这一小节的教程提供了详细的操作步骤，告诉你如何在 Testcomplete 下创建项目、录制和回放简单的测试、和分析测试结果。这个测试模拟了用户使用被测软件的行为，同时验证了一些数据。验证命令会在录制测试的过程中生成。
在我们的讲解里，我们使用一个名为 *Orders* 的应用程序来作演示（它是随着 Testcomplete 发布的）。这一应用程序显示了一个订单列表，还包含了一些特定的功能：增加、删除、修改、输出订单。



这个应用程序在以下路径可以找到：

- 在 Windows 7, Windows Vista 或者 Windows Server 2008:

C:\Users\Public\Documents\TestComplete 7 Samples\Open Apps\OrdersDemo

- 在 Windows XP, Windows Server 2003 或者 Windows 2000:

  C:\Documents and Settings\All Users\Documents\TestComplete 7 Samples\Open Apps\OrdersDemo

注意在 Windows Explorer 和打开、保存文件框下，*All Users\Documents* 文件夹可能会显示成 *All Users\Shared Documents*.

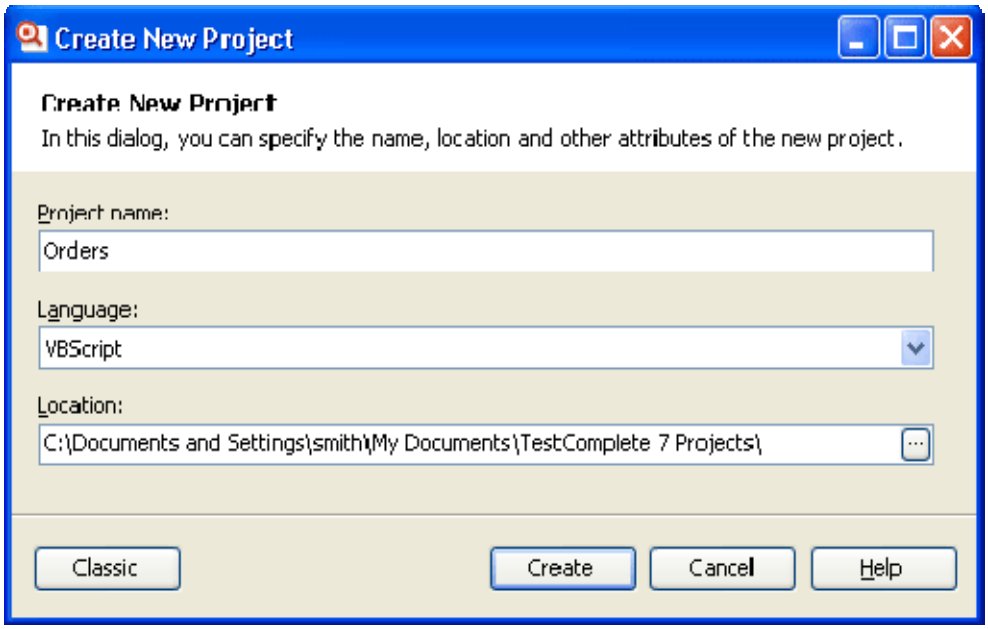
这个文件夹下存放和很多订单项目，它们由不同的编译器创建的：C#, Visual C++, Visual Basic, Delphi, C++Builder, Swing 等等。我们采用由 C#创建的 *Orders* 应用程序。

1. Creating a Test Project（创建一个测试项目）

让我们来创建一个新项目：

1. 如果你当前在 Testcomplete 下打开了项目组或者项目，先关闭它们。在菜单栏点击 **File | Close** 能完成这一步骤。
2. 在菜单栏选择 **File | New | New Project**。这会调用创建新项目的对话框。这个对话框由两种工作模式：简单（Simple）和典型（Classic）。在简单模式是默认的模式，对话框包括了 3 个输入文本框，你可以填上项目名、路径和脚本语言。在典型模式下，你也可以定义项目组的名称（project suite name）、选择项目模板（project's template）和项目项（project items）。

这份教材用的是简单模式，它比典型模式更常用。



3. 我们现在输入项目名、路径和脚本语言
   - 在 project name 输入 *Orders*：Testcomplete 会自动创建项目的路径并显示在 **Location** 这一栏上。项目文件家是用来存放项目生成或者用到的信息：关键字测试、脚本、测试日志、数据等等。你可以在 **Location 文本框**中改变项目的路径。在我们的例子里我们用默认值。

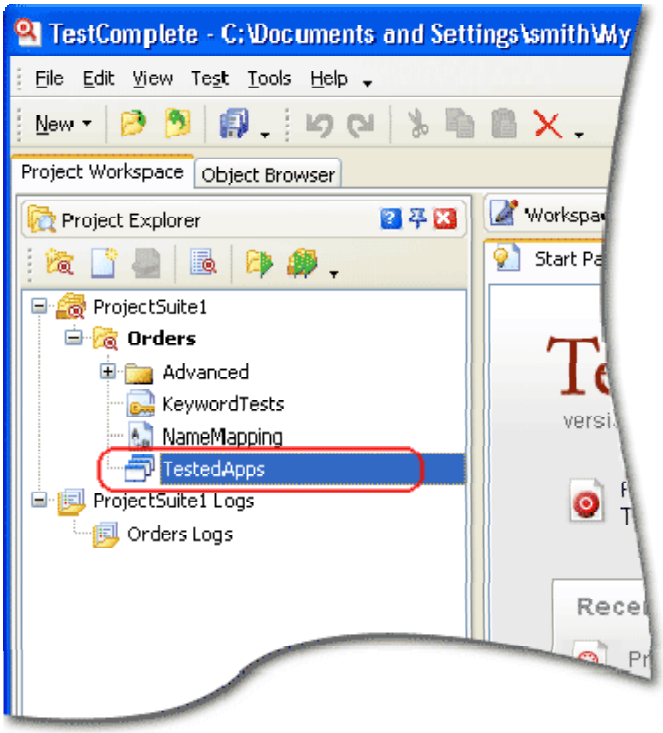
- 在 **Language** 文本框我们定义在测试中将要用到的脚本语言。在这里我们选择了 **VBScript**。即使你不使用脚本单元，脚本语言也是非常重要的。即使你准备使用关键字测试，你可能也会调用一小段的代码或者使用脚本来定义执行参数（operation parameters）。
  说脚本语言重要，是因为它定义了对象名（object names）的格式，这是测试总要用到的，不管你是用脚本还是关键字测试。名称的格式是由脚本语言的语法来定义的。例如，在 VBScript 和 Jscript，Notepad 显示成 `Process("Notepad")`，而在 DelphiScript 你要把双引号变成单引号，如 `Process ('Notepad')`；在 C++Script 和 C#Script 下"Process"必须在方括号里：`["Process"]("Notepad")`。
  Testcomplete对VBScript, JScript, DelphiScript, C++Script 和 C#Script都支持，因此你可以选择你最熟悉的语言。无论你选择哪种语言，你都能够使用Testcomplete的所有特性。然而，也难为不同的语言有不同的语法，它们的脚本解释器是由不同的脚本引擎来执行的，它们会有一些例外（见*Supported Scripting Languages – Peculiarities of Usage*）。想获得更过关于语言选择的信息，请看（*Selecting the Scripting Language*）。请注意到，脚本语言和被测程序使用什么语言是毫无关系的。例如，你用 Jscript 来测试 Visual C++写的程序或者用 VBScript 测试 Delphi 写的程序。然而，如果你想创建一个连接（**Connected**）或者自动自测（**Self-Testing Application**）的程序，我们建议你选择与被测程序的开发工具关联最密切的语言。例如，如果你使用 Visual C++ 或 C++Builder，你应该选择 C++Script；如果你使用的是 Visual Basic，你应该选择 VBScript，以此类推。这会使录制脚本更容易导入连接（**Connected**）或者自动自测（**Self-Testing Application**）程序。

4. 在你定义了项目名称、脚本语言和路径之后，点击 **Create** 按钮。Testcomplete 将会创建一个新项目，*Orders.mds*，和一个对应的项目组。它会接着在项目浏览器面板显示项目组和项目内容。

2. Defining Applications to Test（定义被测程序）

每个项目或许会有被测程序的列表。这是让你跟踪哪个项目对应哪个程序和它们在测试里如何配置。这允许 Testcomplete 去执行列表里面的每一个引用程序或者在内容菜单或测试里手工执行。当然，因为项目之间是相对独立的，不同的被测程序会分布在不止一个的项目里面。你可以手工向列表添加被测应用程序，或者让 Testcomplete 在录制测试的过程中自动完成。录制器十分聪明，能够通过命令行、windows Explorer 或者其他方式检测到应用程序开始执行。在录制结束之后，Testcomplete 会把被测程序加载到列表中，同时加载"Run Tested Application"到录制完的测试里面。
在这一份教程里，为了让你更熟悉列表、演示 Testcomplete 中管理被测程序这一特性，我们手工来向列表添加被测程序。

让我们添加一个简单的被测程序到列表里：
1. 在项目浏览面板里右键单击 **TestedApps** 节点



2. 在内容菜单里选择 **Add | New Item**，这会弹出一个标准的 **Open File** dialog。
3. 找到使用这个 **Open File** dialog 的 *Orders.exe* 可执行文件，然后点击 **Open**。
   回忆一下，我们将使用 Orders 这个用 C#编写、随着 Testcomplete 发布的简单程序。它的路径像是这样的：

   C:\Documents and Settings\All Users\Documents\TestComplete 7 Samples\Open Apps\OrdersDemo\C#\bin\Debug\Orders.exe (Windows XP, Windows 2000 or Windows Server 2003), 或 *C:\Users\Public\Documents\TestComplete 7 Samples\Open Apps\OrdersDemo\C#\bin\Debug\Orders.exe* (Windows 7, Vista or Windows Server 2008).

4. 在你向列表添加完被测程序后，你可以定义它的运行模式和运行参数。要获取更多的信息，请查看*Run Modes*这份帮助文档。在这份教材里，我们会使用默认设置。
5. 通过在主菜单点击**File | Save**来保存你的更改。

3. Planning Your Test（制定测试计划）

General Notes on Planning Tests（制定测试计划的注意事项）

通常，做个测试计划是个很不错的实践：

- **定义测试目标**和制定哪些功能需要被测试。目标越清晰，测试越简单、越有效。大型的、要处理大量程序行为的测试非常的难以创建和维护。创建一个目标明确的、简单的测试更有好处。一旦你建立了很多简单的测试，你可以将它们组织到一个更大的测试里面。
- **计划测试步骤**和决定运行哪些测试场景。测试步骤由测试的目的和被测程序的特性

来共同决定。测试步骤可能包括测试程序的前期准备（比如被测程序的初始化）。测试步骤也可以是给应用程序一些初始的输入值。

- **计划检查点的动作**。通常，在应用程序执行了一些操作之后，应用程序将会发生一些改变：在程序窗口的数据可能会被改变或者重新分配、一个新的窗口被创建、一个文件在硬盘里创建或者删除。你必须定义测试通过与否的标准和哪些检查点来验证这些标准。
- **记录测试结果**。测试结果能用不同的方式记录下来，例如，你的测试可以把完整的测试结果保存在一个文件里，或者弹出一个对话框来提醒你的测试已经结束了。Testcomplete记录了测试中模拟的所有行为同时在日志里保存了这些行为的信息。你也可以记录定制信息、图像、文件或者指向测试日志的链接。这些信息可以组织成文件夹，且每一份信息都可以用特定的字体和背景色来显示。测试结果可以输出到文件、压缩和通过电子邮件发送到你同事。你甚至可以直接通过测试结果的日志，在缺陷跟踪系统创建缺陷报告。请查阅*Test Log*来获取更多信息。

Planning a Test for the Orders Application（为订单程序制定测试计划）

这个 Orders 应用程序的例子维护了一个订单列表。假定我们要测试这个系统编辑订单的功能是否正确、是否成功修改了订单的数据，在这个例子里：

- **测试目的**：这个测试应该检查**编辑订单界面**是否保存了修改后的数据、改变后的数据是否能够显示出来。
- **测试步骤**：我们的测试必须模拟出修改订单的细节，同时验证订单列表里面的数据。我们将录制模拟用户操作的测试。为了简单起见，我们的测试只改变了一个订单的一个属性。
- **检查和验证测试结果**：对订单的修改能正确保存，那它应该显示在订单列表里。为了作出检查，我们的测试将列表里的数据和预期数据作出比较。我们在测试里加入特殊的指令来完成这一任务。这个指令会把对比的结果记录到测试日志里，从而我们可以验证测试通过与否。

4. Recording a Test（录制测试）

Recording Tests in TestComplete （在 Testcomplete 里录制测试脚本）

录制测试由一下三个步骤组成：

1. 在Testcomplete的主菜单或者测试引擎工具栏点击**Test | Record | Record Keyword Test** 或 **Test | Record | Record Script**。你也可以在开始页面上点击**Record a New Test**。在Testcomplete下面，你可以选择一下类型的测试录制：keyword tests, scripts, low-level procedures 和 HTTP load testing tasks。录制用的菜单项定义了主要的测试录制方式：keyword test 或 script code。其他的测试类型将会在录制开始之后启动。主要的录制方式会包含特别的指令来完成这些测试。
   在你命令 Testcomplete 启动测试之后，它会切换测试类型和在屏幕上显示 **Recording toolbar**：

这个工具栏包括了一些附加功能，你可以在录制中使用，暂停或停止录制，改变录制类型(keyword test, script code, Windows or PDA low-level procedure，或 HTTP traffic).

2. 在你启动测试之后，执行以下的测试步骤：启动被测程序（有必要的话）、在上面点击命令按钮、选择菜单项、键盘输入文本等等。
3. 在测试动作结束后，在Recording toolbar点击 **Stop**来停止录制。

要获取更多的信息，请查阅帮助文档*Recording in TestComplete*；

Recording Test for the Orders Application（为订单应用程序录制脚本）

现在，我们为 Orders 这个应用程序录制一个关键字脚本。这个测试会运行应用程序、往里面加载数据、模拟程序界面内的鼠标点击和键盘输入、验证程序的数据。

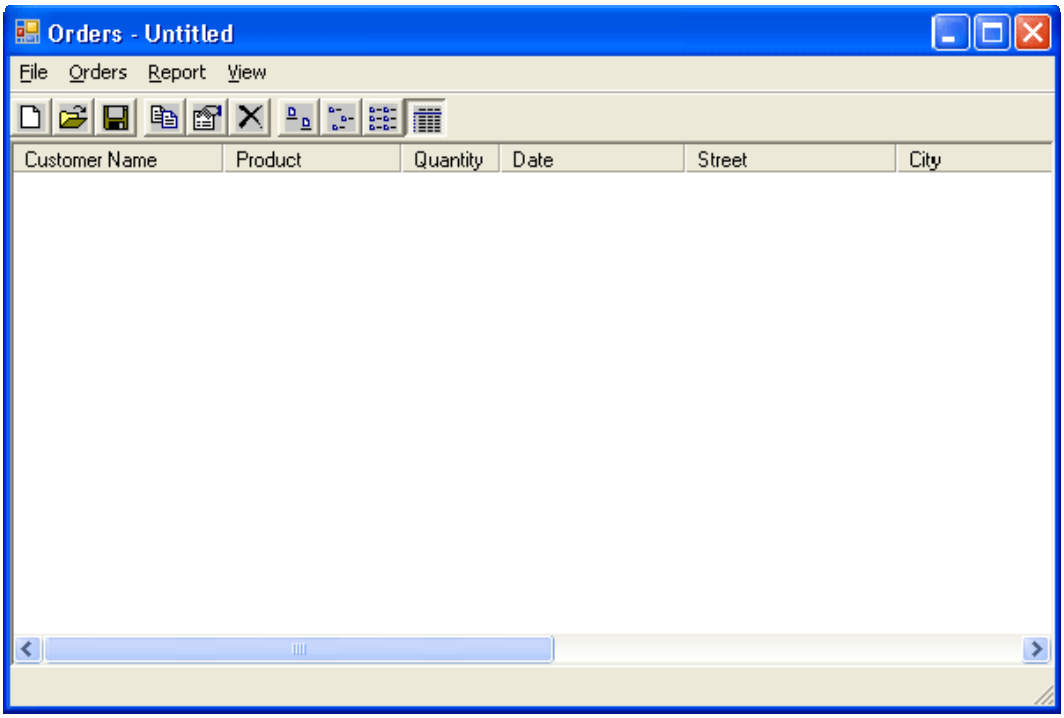
注：如果你在屏幕里阅读这份文档，请不要在录制脚本的时候进行切换。因为录制引擎会跟踪和录制用户的所有操作，切换文档的指令也会录制到的测试脚本里。
如果真的想看这些操作的话，你可以事先打印出栏。或者你用两台显示器，这样你就可以把这份文档放在另一个屏幕里。

开始录制：

1. 从主菜单选择 **Test | Record | Record Keyword Test**
2. 点击展开 **Run Tested Applications** 按钮旁边的小箭头，在下拉列表选择 Orders：

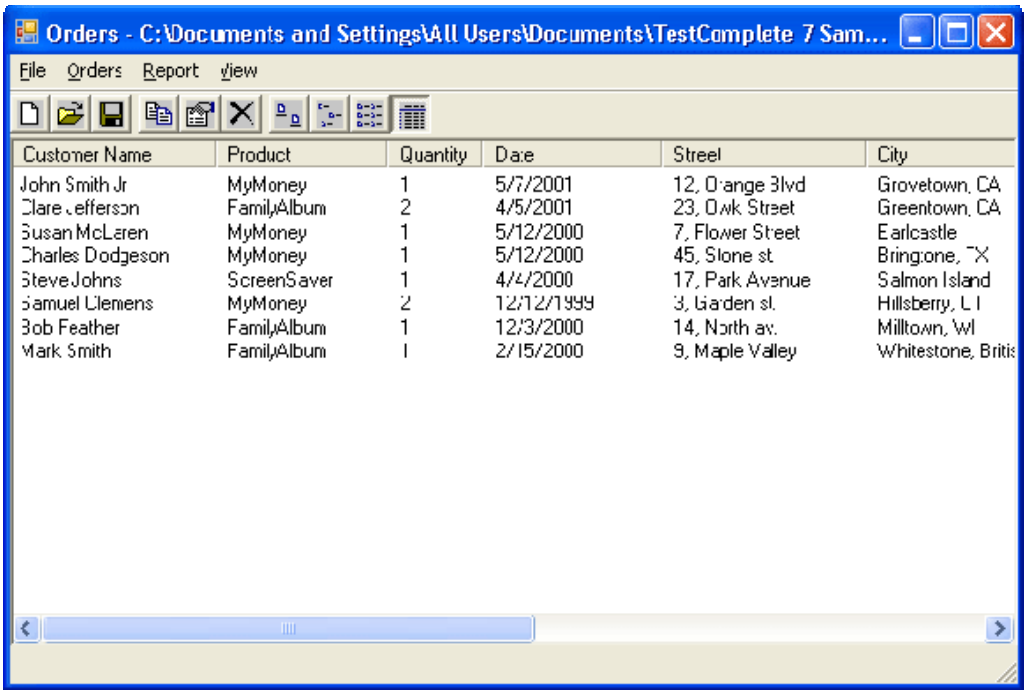
Testcomplete 会自动把启动应用程序的命令加载到录制的测试里。在后面我们分析录制测试的时候你会看到这条指令。

3. 稍等片刻，直到应用程序的主窗口启动，如下图示：

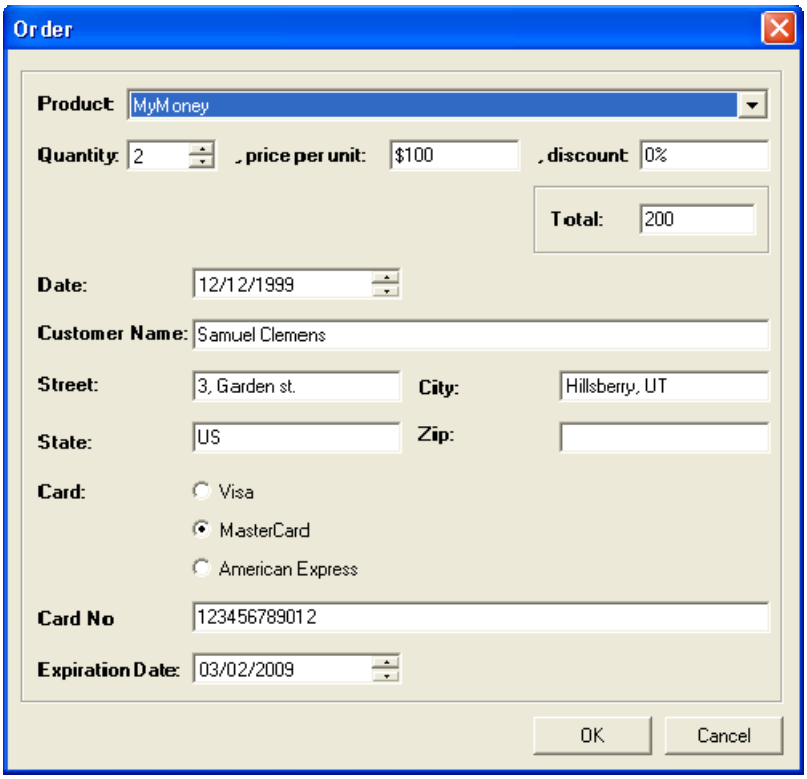


如果交互式帮助文档的界面可见的话，请调整它的大小或者移开它，一面挡住了应用程序的窗口。 否则你在界面上的操作是无法录制的。

4. 切换到 Orders 程序，点击它的主菜单 **File | Open**。这会打开一个标准的打开文件对话框（Open File dialog）。
5. 在这个对话框里，打开文件 *MyTable.tbl*。它的路径取决与你安装的操作系统。在 Windows 7, Windows Vista and Windows Server 2008 下的路径是 *C:\Users\Public\Documents\TestComplete 7 Samples\Open Apps\OrdersDemo*。在其他操作系统，它的路径在 *C:\Documents and Settings\All Users\Documents\TestComplete 7 Samples\Open Apps\OrdersDemo*。建议你在 Open File dialog 的 **File name** 文本框用键盘输入完整的文件名。用键盘输入而不是鼠标点击，目的是防止由于 Open File dialog 显示不同的初始路径而造成回放问题。
6. 在 **File name** 文本框输入文件名后，点击 **Open**。Orders 应用程序会从文件加载数据，将其显示在主界面上：



7. 在订单列表点击 *Samuel Clemens* 这一行。
8. 移动鼠标到 Orders 的工具栏上，并点击 **Edit order** 调用 **Order** 对话框：



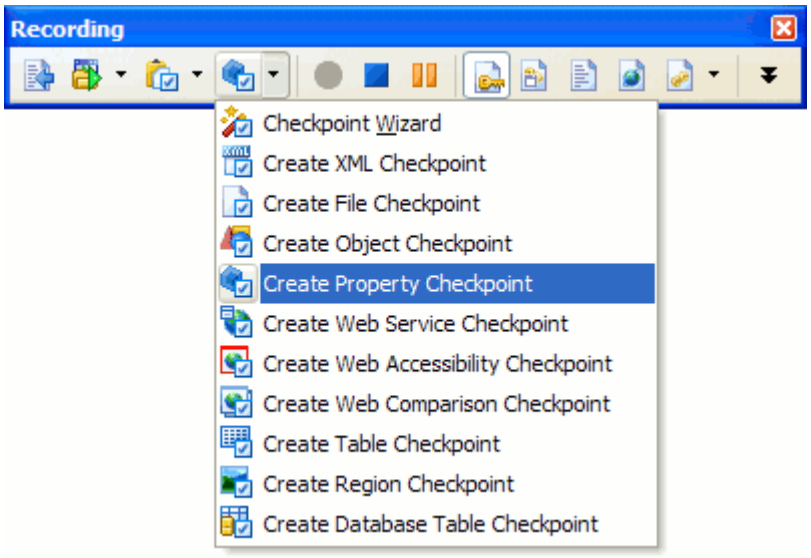
9. 在对话框里，点击一下 **Customer Name** 文本框把光标插入点（insertion point）移进去。在 **Customer Name** 文本框右击鼠标，从内容菜单里选择 **Select All**，然后输入 *Mark Twain* 作为客户名称。

   注： 你可以通过鼠标拖动文本框的文本来选择，但这种情况下的测试就没那么通用了，因为如果文本长度是一个变量，在测试里我们不得不拖动长一点或者短一点，

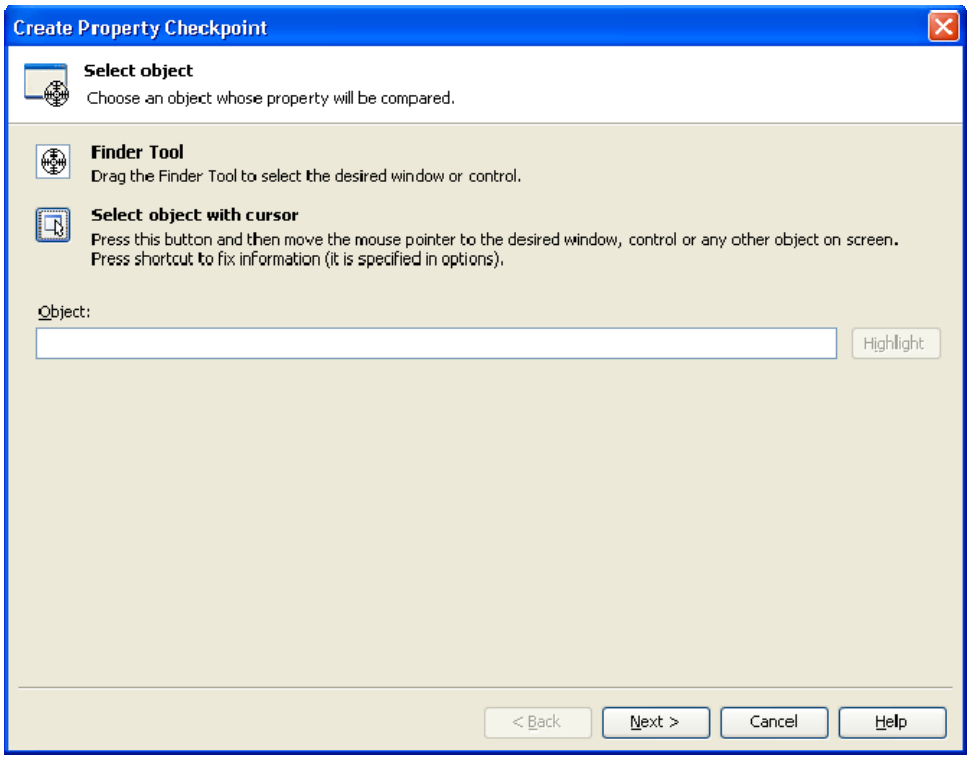
换句话说，你不得不在测试中计算你拖动的长度。使用 Select All 菜单项这个快捷方式，你可以免去计算的问题，同时使你的测试更加稳定。另一种可选的途径也可以实现 Select All 菜单项这个功能：使用快捷键（通常是 CTRL+A)来选择所有文本。

10. 点击 **OK** 来关闭对话框。Testcomplete 会更新应用程序主窗口里面的客户列表（customer list）。
11. 现在我们在测试中插入一个比较指令。它将会验证显示在应用程序客户列表中被修改过的名字——*Mark Twain*.
    我们可以把这类比较指令称作检查点（**checkpoints**） 。Testcomplete 提供验证不同类型数据的各类检查点。其中用得最多的检查点是属性检查点（Property checkpoint）。它用来验证应用程序的控件数据。我们在这份教材里使用这个检查点。

    - 从录制工具栏（Recording toolbar） 的 **Checkpoint** 下拉列表选择 **Create Property Checkpoint**:

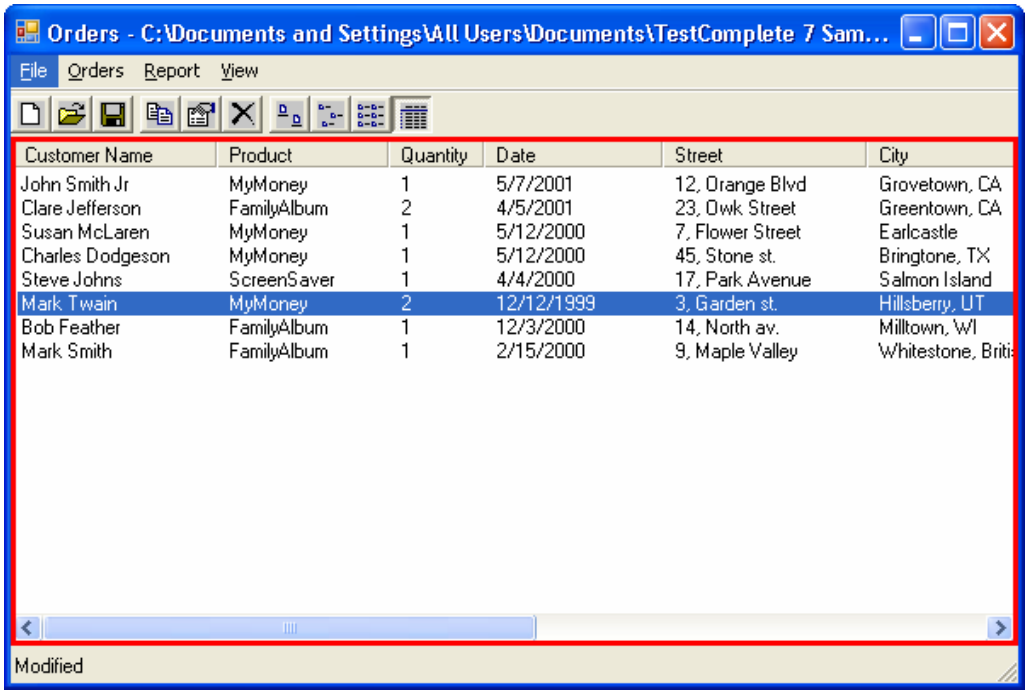


    - 这会调用 Create Property Checkpoint 这一向导。它会指导完成创建检查点的过程：

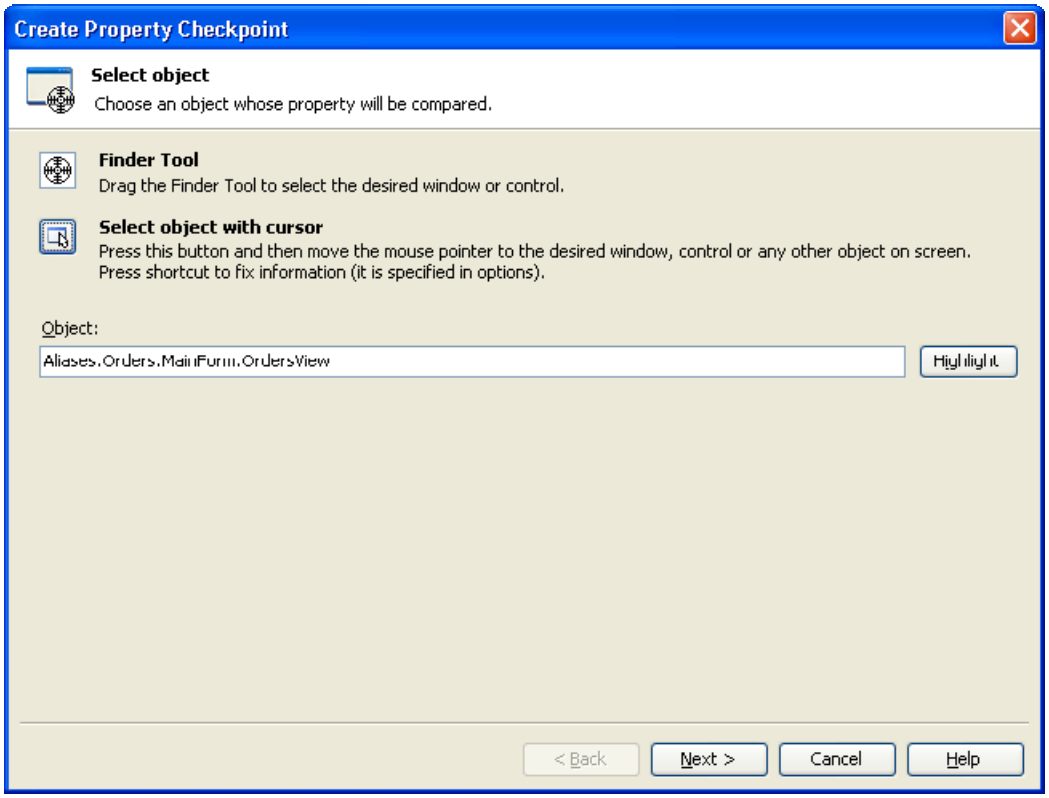


- 在向导的第一页，用鼠标左键单击 Finder tool 这个图标 ( )，同时不要松开鼠标左键。

  等待向导页面最小化，然拖动图标到 Orders 应用程序的客户列表 。在你拖动的过程中，Testcomplete 会在红框里突出光标下面的控件和窗口。
- 当 Finder tool 图标放在客户列表上面、同时客户列表在红框里突出显示时，松开鼠标键：



- 在你松开鼠标键后，Testcomplete 会保存这个向导并把所选的对象显示在 **Object** 文本框下：



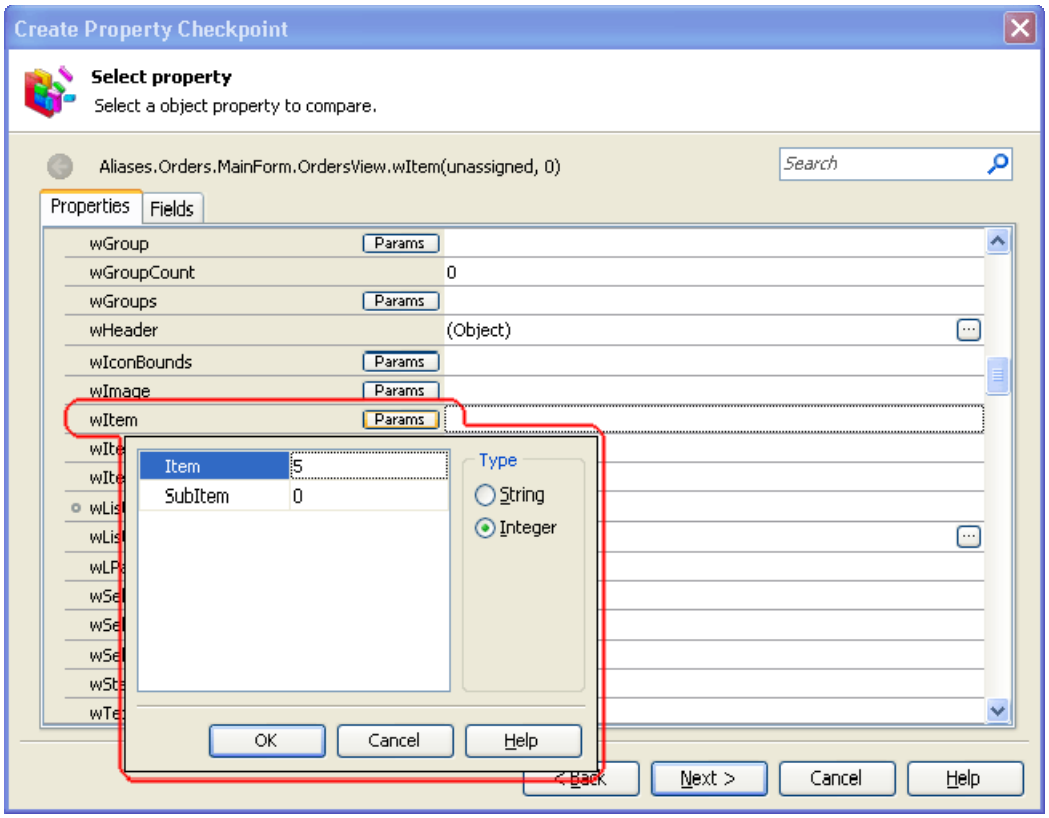
点击 **Next** 进行后续操作。

- 向导的下一页显示了被选对象的属性列表。这个列表包括了 Testcomplete 提供的属性和被测程序自己定义的属性。例如，我们的

被测程序是用 C#创建的，所以列表上显示的是对应.Net 类库的属性。你可以在**.NET** 节点下看到它们。

Testcomplete 提供了两组对象的属性：一组包括了被测窗口或控件的公共属性。大部分可以在 **Standard** 节点那里查看。另一组包括了 tree-view 控件特有的属性（因为我们选择的是 treeview 控件）。这些属性名称的首字母是 **w.** 你可以在 **Extended** 节点下看到它们。为了验证数据，我们使用 `wItem` 属性。可以通过来它访问 tree view 控件里每一项。

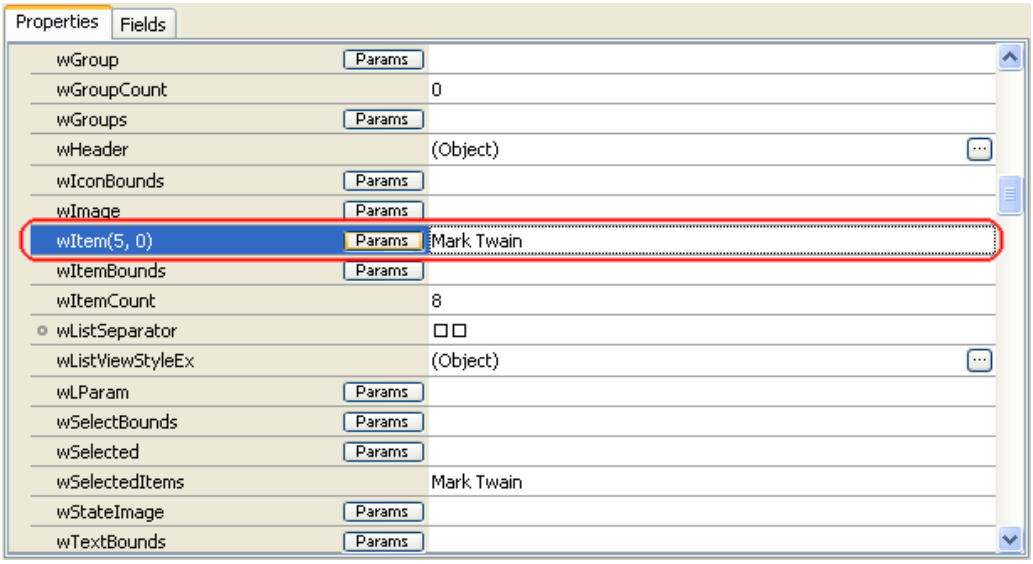
- 遭到列表里面的 `wItem` 属性（在 **Extended** 的节点下面）。单击 **Params** 按钮。弹出如下窗口：



在这个窗口里面，定位到持有 *Mark Twain* 这个字符串的单元格：

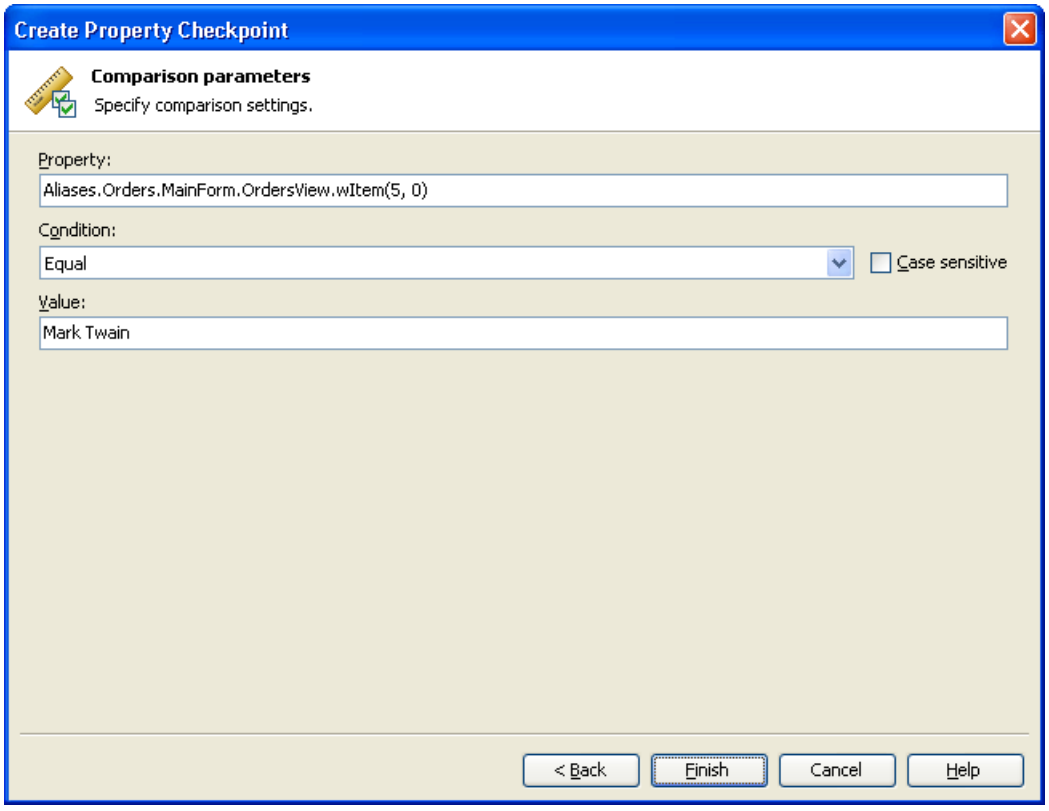
- 在 **Type** 区域选择 **Integer**。
- 在 **Item** 文本框键入 **5**。（5 是 *Mark Twain* 在 treeview 控件里的索引。索引的值是从 0 开始递增的）
- 点击 **OK**.

- 测试引擎会检索到这一项的数据并显示在属性列表中：



点击 next 继续。

- 在向导的下一页，你会看到属性的名称（将要验证的值）。比较的条件和基线数据在 **Value** 文本框定义：

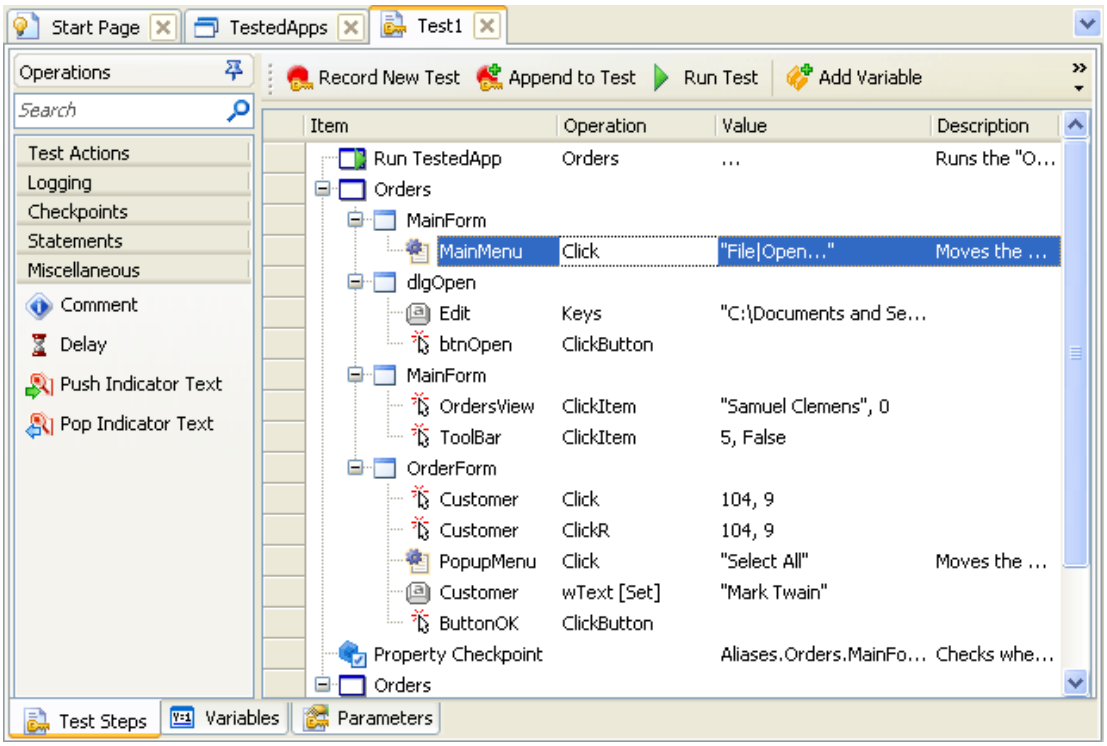


点击 **Finish** 来完成检查点的创建。Testcomplete 会将检查点的指令附加到录制的测试中。

12. 点击 window 标题栏的 X 按钮来关闭 Orders 程序的窗口。这会弹出一个对话框来问你是否要保存变更。点击 **No**。Orders 程序关闭了。
13. 点击录制工具栏的 **Stop** 按钮来停止录制。Testcomplete 会处理录制的测试指令并将测试保存下来。

5. Analyzing the Recorded Test（分析录制的测试）

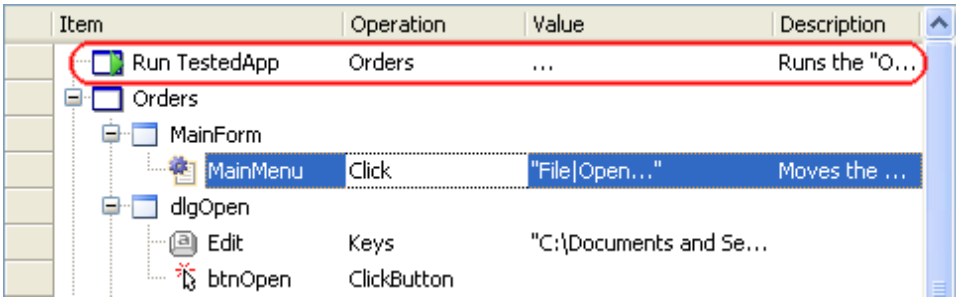
在录制完成之后，Testcomplete 会打开录制完成的关键字测试（可供编辑）并把测试的内容显示在 KeywordTest 编辑器上：



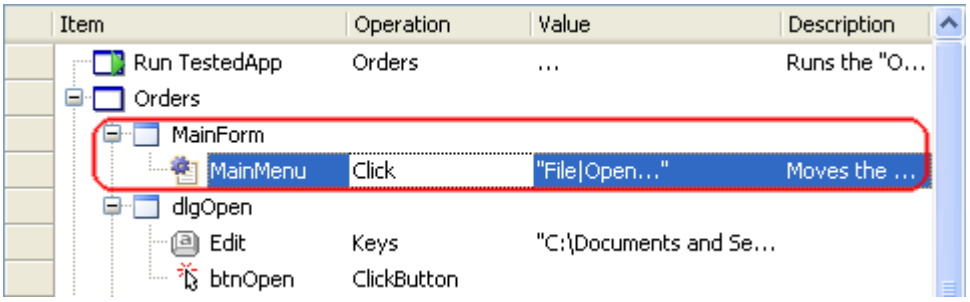
录制完成的测试与上图显示的测试非常类似。在你实际的测试中和会和这个例子会稍有出入。例如，如果你录制的是 C++或者 Delphi 写的应用程序，它可能会包括其他的对象名或者窗口索引。

测试指令取决与你在录制过程中怎样操作 Orders 应用程序。我们把测试指令称作**操作(operations)。**

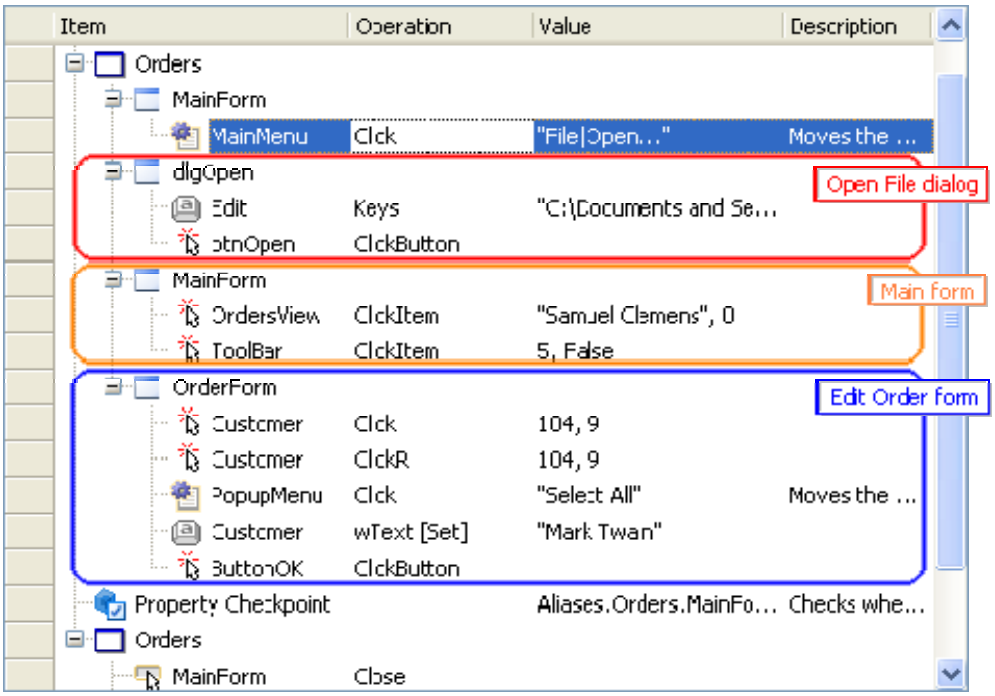
- 在测试里执行的第一个操作是运行被测程序（Run TestedApp）。它在关键字测试中启动被测程序。（在我们的例子里，是 *Orders* 这个应用程序。）当 Testcomplete 检测到用户在 Testcomplete 用户界面或者在操作系统界面的其他地方运行被测程序，它会自动录制这个操作。



- 下一个操作与选择 File | Open 这个菜单有关。

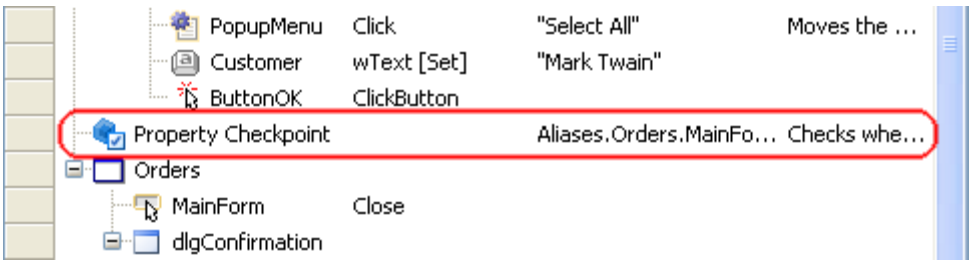


- 接下来是一连串模拟你在打开文件对话框、应用程序主窗口、订单界面的操作：

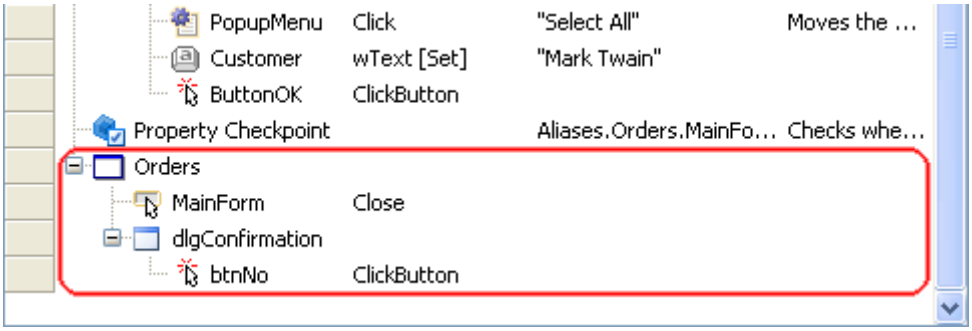


请查看*Simulating User Actions*这一帮助文档来获取更多关于模拟鼠标事件、键盘输入及其他动作的信息。

- 然后是我们在测试录制是添加进去的比较操作：

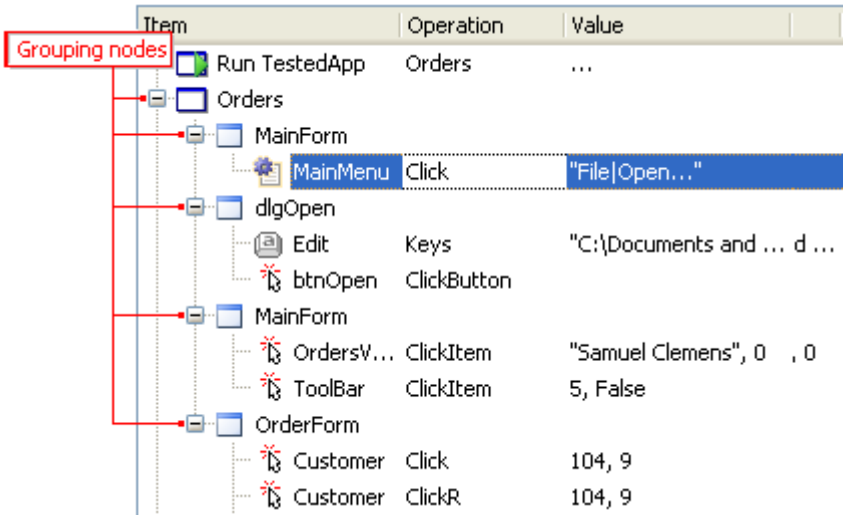


- 最后，是关闭 Orders 应用程序和模拟点击对话框上“No”按钮的操作：



如你所见，Testcomplete 自动把你操作进程和窗体的步骤分组组织起来。分组的测试体系更容易理解，同时提供了一些有关被测对象层次结构的信息 。

- 我们录制了在一个进程（*Orders*）的用户操作。所以，我们只有一个进程组节点，你在进程窗口和控件上的所有模拟操作都包含进去。我们在 Orders 进程的窗口和控件上执行的操作都被组织成一系列的“window”节点：



你可能会注意到被测进程及其窗口、控件的名称与我们之前在对象浏览器（Object Browser）看到的名称不一样。例如，在对象浏览器里我们看到的被测进程名称是 *Process("Orders")*，但在测试中它的名称是 *Orders*;主窗口的名称是 *WinFormsObject("MainForm")*，但在测试里它叫 *MainForm*，依此类推。

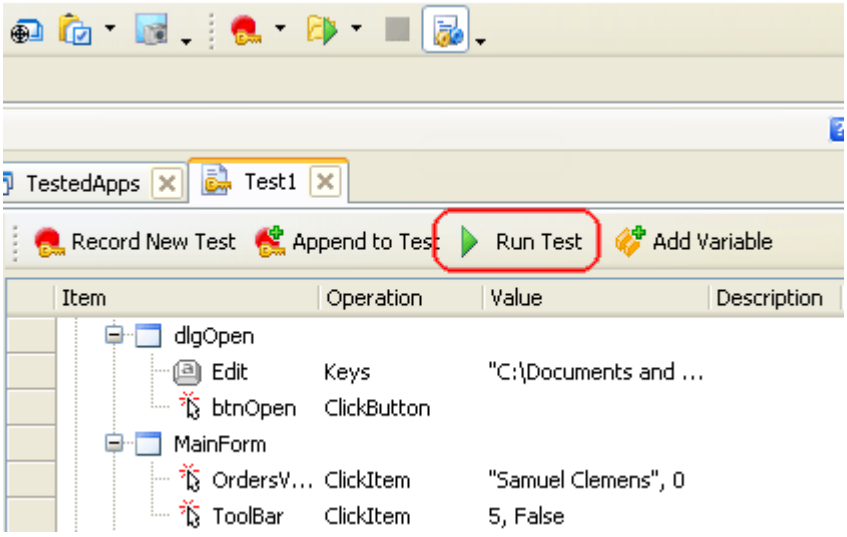
这个逻辑背后的原理是：Testcomplete默认在录制测试脚本的时候会为你的被测对象生成和使用自定义名称。生成和分配自定义名称，这个动作称为命名映射（*name mapping*）.Testcomplete之所以这么做，是因为默认的名称可能会难以理解。要确定哪个窗体和控件对应哪个名称可能会比较困难。使用命名映射使得测试更容易理解和更加可靠。请参考*Name Mapping*帮助文档来获取更多信息。

6. Running the Recorded Test（执行测试脚本）

现在我们可以运行这个测试例子，看看一下 Testcomplete 怎样去模拟用户操作。**在回放录制的测试之前，请确保当前启动程序的初始条件和录制测试的初始条件是相同的**。例如，测试几乎都需要被测程序处于运行状态。所以，在模拟用户的操作之前，你必须启动被测程序。在我们的例子里，测试开始前，我们用 Run TestedApp

这个操作来启动我们的被测程序，所以我们的测试会自动为我们启动被测程序。或者，你可以手动从 Testcomplete 的集成开发环境（IDE）里面启动被测程序。

▪ 点击测试编辑器工具栏的 **Run Test** 来回放录制的测试：



测试引擎会自动把 Testcomplete 的窗口最小化，并启动测试指令。在我们的例子里，这个测试会简单的重复你之前录制的操作。

注：不要在测试过程中移动鼠标或者按下键盘。你这些行为可能会干涉到 Testcomplete 模拟的测试，导致这个测试出现问题。

在测试执行结束之后，Testcomplete 会还原它的窗口并显示测试结果。在下一节里我们会分析它们。

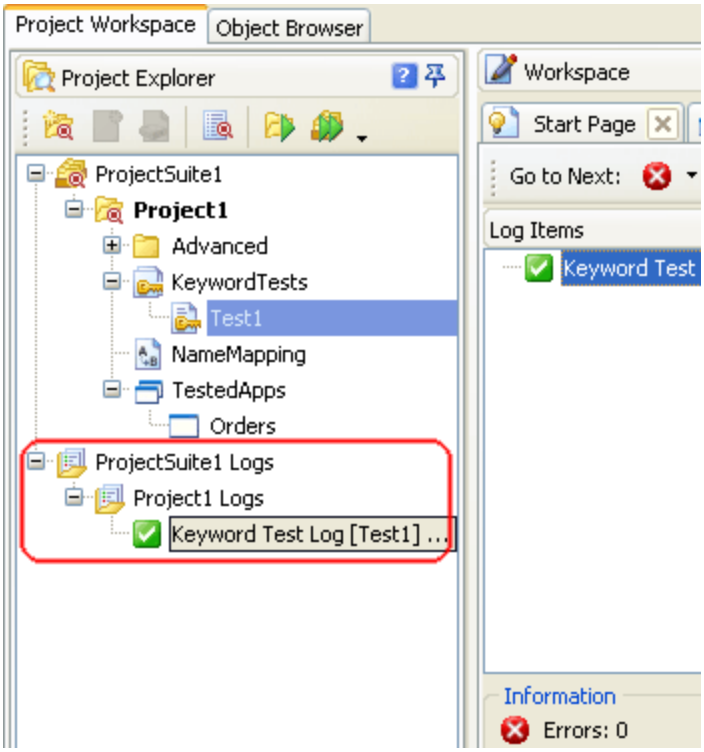
运行测试的一些注意事项：

- 之前创建的测试并非经过编译而用于测试的可执行文件。你直接从 Testcomplete里运行测试。如果想在没有安装Testcomplete的计算机上执行测试，你可以使用名为TestExecute的共享工具。请看*Connected and Self-Testing Applications*这节帮助文档以获取更多信息。
- 在你开始测试之后，Testcomplete会一直运行测试命令直到测试停止。你可以点击测试引擎工具栏上的 **Stop**按钮或者点击主菜单上的**Test | Stop**。
- 你可以点击 **Pause**来暂停测试。在暂停中，你可以按需做任何操作。例如，你可以浏览测试日志或者通过Testcomplete的Watch List 、Locals panel 或Evaluate dialog检查测试变量和对象。（看*Debugging Tests*这一节帮助文档）
- 我们使用测试编辑器工具栏上的 Run Test 按钮来启动测试。这只是其中一种执行测试的方法（还有很多种其他方法）。你也可以在项目浏览器或其他测试里运行测试。你也可以使用项目编辑器下的 Test Items 页来执行批量的测试。

  请查看*Running Tests*文档获取更多的信息。

7. Analyzing Test Results（分析测试结果）

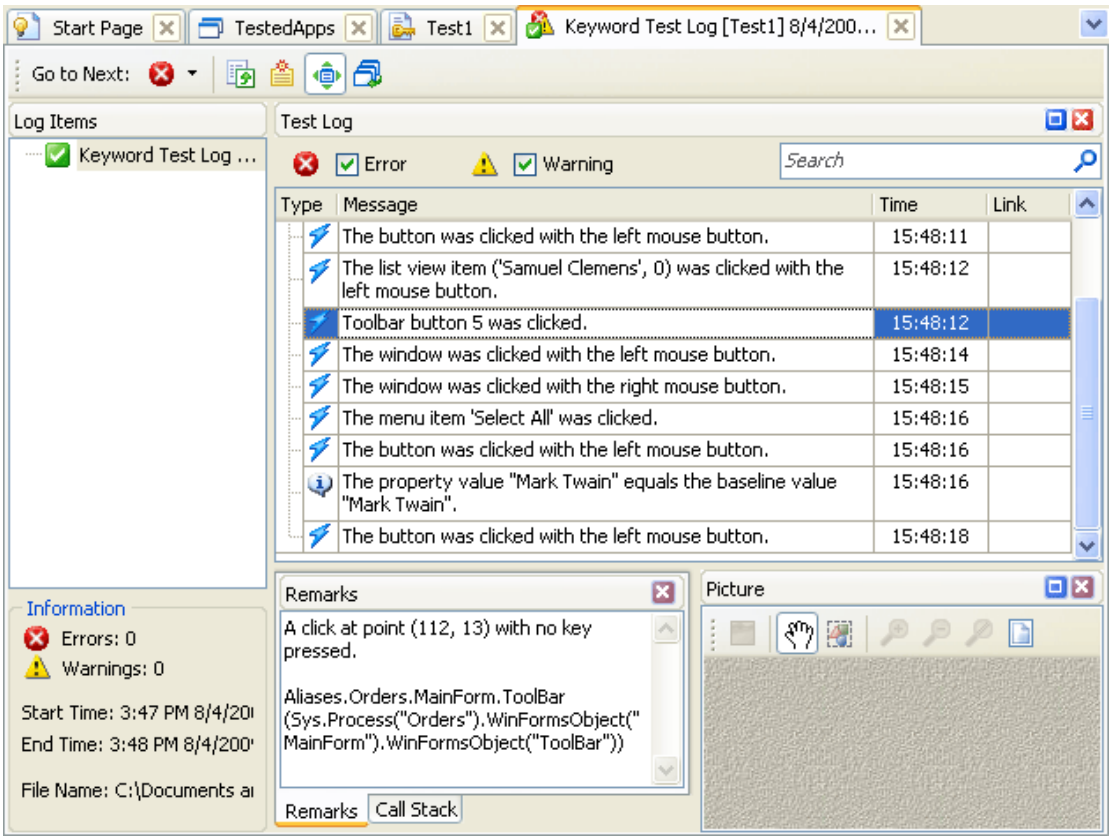
Testcomplete 的测试日志保存了测试中的所有操作。指向测试结果的链接显示在项目浏览器界面的 **ProjectSuite1 Log | Orders Log** 节点下。这是查看项目或则项目组测试历史的主要工作区。每一个节点都与对应的测试相关。节点左边的图标指明了它对应的测试通过与否：



注意到 Testcomplete 会在测试结束之后自动添加节点。也就是，测试在执行的时候测试结果是不会显示的（你可以通过暂停测试来查看中间结果）。

因为到目前为止我们值执行了一次测试，我们在项目浏览器里只有一个日志节点。默认下，Testcomplete 会在工作区自动打开这个节点的内容。你也可以随时查看日志。在项目浏览器右击所选的测试结果，选择菜单里的 **Open**。

- 在我们的例子里，日志是这样的：

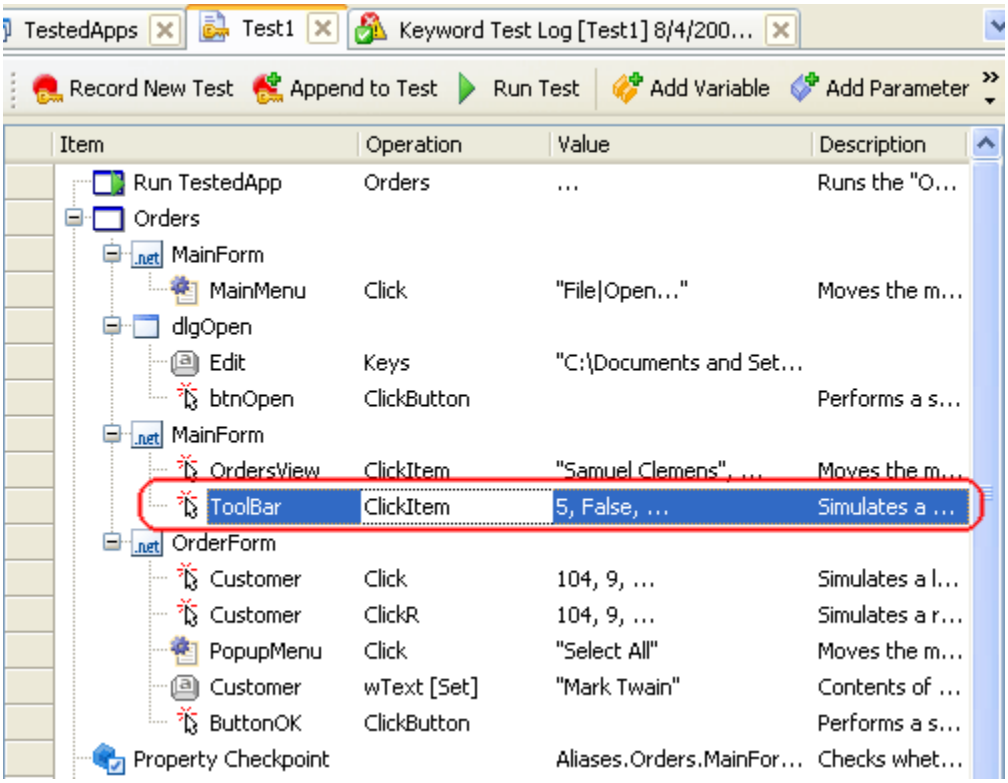


日志窗口显示了一次测试的结果。左边的窗口是一个已运行测试的树形结构 ；每个测试的节点都可以选择，以查看它们的结果。在我们的例子里，我们只运行了一个测试，所以这颗树只有一个节点。节点的图标显示了测试是否通过。

测试日志包括错误、警告、提示信息和其他类型的信息。左边的图标显示了信息的类型。使用信息列表顶上的单选框你可以按类隐藏或者显示信息。Testcomplete 可能会提供附加的文本或者图像信息。点击对应的信息，并查看 **Remarks** 和 **Picture** 子窗口，你就能找到它们。例如，在上图 Remarks 子窗口里显示了如下附加信息“Toolbar button 5 was clicked”。

日志的 **Call Stack** 子窗口显示了（能触发写日志行为的）测试调用的层次。

- 双击对应的日志信息来查看（触发写日志的）测试操作。Testcomplete 会在编辑器里打开相应的关键字测试，并把对应的操作高亮显示。例如，如果你在日志里双击“Toolbar button 5 was clicked”这条信息，Testcomplete 会把执行这个测试操作高亮显示出来：



请看*Test Log* 以获取更多信息。

注：这里我们讨论的日志是 Testcomplete 里的关键字测试和脚本所生成的。其他类型的测试可能在结构上会有所不同。例如，在 HTTP 压力测试，生成的测试日志里会包含一些有关虚用户、连接、模拟请求和响应的表格。如需获取这些日志的更多细节信息，请查看相关项目项（project item）的描述，或者单击日志页面并按下 F1 。