Defining test cases using a DataTable

As mentioned earlier, a data-driven test is one that is designed to behave as required by different sets of parameter values. Basically, such sets of values actually represent different test cases. When executing a login test action, for example, valid or invalid values for the username and the password will trigger different application responses. Of course, the best is to have a single action (or function) that will handle all cases, with the flow branching according to the input data.

Getting ready

Ensure that you have the Flight Reservation sample application shipped with the installed UFT. You can check this by navigating to **Start** | **All Programs** | **HP Software** | **Unified Functional Testing** | **Sample Applications**. You should have a shortcut named **Flight GUI** that launches `flight4a.exe`. Create a new test by navigating to **File** | **New** | **Test** from the menu, or by using the *Ctrl* + *N* keyboard shortcut. Rename `Action1` to `FR_Login` (optional).

How to do it...

Proceed with the following steps:

1. In the DataTable, select the `FR_Login` (or `Action1` if you decided

tasheet. Create the following parameters in the

ibed in the *Creating a DataTable parameter*

   - `TC_ID`
   - `Agent`
   - `Password`
   - `Button`
   - `Message1`
   - `Message2`
   - `Description`

2. We will derive the test cases with reference to the system requirements, as we know (for this example, we will ignore the **Cancel** and **Help** buttons):

   - The correct login password is always `mercury`. A wrong password triggers an appropriate message.
   - The agent name must be at least four characters long. If shorter, the application prompts the user with an appropriate message.
   - An empty agent name triggers an appropriate message.
   - An empty password triggers an appropriate message.
   - After four consecutive failed login attempts with a wrong password, the application prompts the user with an appropriate message and then closes.

   Accordingly, we will enter the following data to represent the test cases:

#	TC_ID	Agent	Password	Button	Message1	Message2	Description

1	AgentEmpty		mercury	**OK**	Please enter agent name		Empty agent
2	AgentLT4	Mer	mercury	**OK**	Agent name must be at least 4 characters long		Agent with less than 4 characters
3	Agent4EmptyPass	Merc		**OK**	Please enter password		Wrong password #1 (empty)
4	Agent4WrongPass	Merc	Merc	**OK**	Incorrect password. Please try again		Wrong password #2
5	Agent4WrongPass	Merc	1234	**OK**	Incorrect password. Please try again		Wrong password #3
6	Agent4WrongPass	Merc	Gfrgfgh	**OK**	Incorrect password. Please try again	Login unsuccessful. Please try again later.	Wrong password #4; App closes
7	SuccessfulLogin	mercury	mercury	**OK**			Correct username and password

3. Apart from learning the TOs for the login and the message dialogs, create two checkpoints for the messages that appear after unsuccessful logins (one for the first and the other for the second type mentioned in the preceding table), and name them Message1 and Message2 respectively.

   OR should contain the following TOs (smart identification should be turned off):

   - Dialog: Login (parent: none, description: text=Login, nativeclass=#32770, is owned window=False, is child window=False)
   - WinEdit: Agent Name (parent: Dialog Login, description: nativeclass=Edit, attached text=Agent Name:)
   - WinEdit: Password (parent: Dialog Login, description: nativeclass=Edit, attached text=Password:)
   - WinButton: OK (parent: Dialog Login, description: text=OK, nativeclass=Button)
   - Dialog: Flight Reservations (parent: Dialog Login, description: text= Flight Reservations, nativeclass=#32770, is owned window=True, is child window=False)
   - Static: Message (parent: Dialog Flight Reservations, description: window id=65535, nativeclass=Static)
   - WinButton: OK (parent: Dialog Flight Reservations, description: text=OK, nativeclass=Button)
   - Window: Flight Reservation (parent: none, description: regexpwndtitle=Flight Reservation, regexpwndclas=Afx:, is owned window=False, is child window=False)
   - WinButton: Delete Order (parent: Window Flight Reservation, description: text=&Delete Order, nativeclass=Button)
   - WinButton: Insert Order (parent: Window Flight Reservation, description: text=&Insert Order, nativeclass=Button)
   - WinButton: Update Order (parent: Window Flight Reservation, description: text=&Update Order,

  nativeclass=Button)

- WinButton: FLIGHT (parent: Window Flight Reservation, description: text=FLIGHT, nativeclass=Button)
- WinRadioButton: First (parent: Window Flight Reservation, description: text=First, nativeclass=Button)

OR should contain the following Checkpoint objects:

- Message1 and Message2: These checkpoints identify the static text appearing in the message that opens after a failed attempt to log in. The checkpoints should verify the enabled=True and text=LocalSheet DataTable parameters for Message1 and Message2 respectively.
- Flight Reservation: This checkpoint verifies that the main window opens with the properties enabled=True and with text (title)=Flight Reservation.
- Delete Order, Insert Order, and Update Order: All three checkpoints should verify that the buttons have the enabled=False and text properties set while opening the main application window set as their learned text property with the ampersand character (&) in the beginning of the string.
- First: This checkpoint for the WinRadiobutton should verify that upon opening the main application window, the properties enabled=False and checked=OFF are set.

4. In FR_Login (Action1), write the following code:

```
'Checks if either the Login or the Main window is already open
Function appNotOpen()
    appNotOpen = true
    If Dialog("Login").Exist(0) or Window("Flight Reservation").Ex
        appNotOpen = false
    End If
End Function

'Opens the application if not already open
Function openApp()
    If appNotOpen() Then
        SystemUtil.Run "C:\Program Files\HP\Unified Functional Tes
        openApp = Dialog("Login").WaitProperty("enabled", 1, 5000)
    else
        openApp = true
    End If
End function

'Handles the Login dialog: Enters the Agent Name and the Password 
Function login(agentName, password, button)
    with Dialog("Login")
        .WinEdit("Agent Name").Set agentName
        .WinEdit("Password").SetSecure password
        .WinButton(button).Click
        If .exist(0) Then
            login = false
        else
            login = true
        End If
    end with
End Function

'Performs a standard checkpoint on a message open by the FR applic
Function checkMessage(id)
    If Dialog("Login").Dialog("Flight Reservations").Exist(0) Then
        checkMessage = Dialog("Login").Dialog("Flight Reservations
        Dialog("Login").Dialog("Flight Reservations").WinButton("O
    else
        checkMessage = false
    End if
End Function

'Performs several standard checkpoints on the Main window and on s
'to verify its initial state
function verifyMainWndInitialState()
    with Window("Flight Reservation")
        if .Check(CheckPoint("Flight Reservation")) then
            .WinButton("FLIGHT").Check CheckPoint("FLIGHT")
            .WinRadioButton("First").Check CheckPoint("First")
            .WinButton("Update Order").Check CheckPoint("Update Or
            .WinButton("Delete Order").Check CheckPoint("Delete Or
            .WinButton("Insert Order").Check CheckPoint("Insert Or
        End if
    end with
End function

'Variables declaration and initialization
Dim agentName, password, button

agentName = DataTable("AgentName", dtLocalSheet)
password = DataTable("Password", dtLocalSheet)
button = DataTable("Button", dtLocalSheet)

'Tries to open the application
If not openApp() Then
    ExitTest
End If

'Tries to login with the input data
if not login(agentName, password, button) Then
    'Checks if a warning/error message opened, if it's correct in 
    if checkMessage("1") then
        'Checks if a second warning/error message opened, if it's 
        if checkMessage("2") then
            If not Dialog("Login").Exist(0) Then
                reporter.ReportEvent micPass, "Login", "Maximum nu
                'If a second message opened, then the number of lo
                call openApp()
            End If
        End if
    End If
else
    call verifyMainWndInitialState()
End if 'Tries to login
```

How it works...

Now, we will explain the flow of the `FR_Login` action and the local functions.

We declare the variables that we need for the `Login` operation, namely, `AgentName`, `Password`, and `Button`. We then initialize them by retrieving their values from the local sheet in the DataTable. The `button` value is parameterized to enable further elaboration of the code to incorporate the cases of clicking on the `Cancel` and `Help` buttons.

Next, we call the `openApp()` function and check the returned value. If it is `False`, then the Flight Reservation application did not open, and therefore we exit the test.

We attempt to log in and pass the `AgentName`, `Password`, and `Button` parameters to the function. If it returns `true`, then login was successful and the `else` block of code is executed where we call the `verifyMainWndInitialState()` function to assert that the main window opened as expected.

If the login did not succeed, we check the first message with a checkpoint that compares the actual text with the text recorded in the DataTable, which is correct in the context of the planned flow.

If the first message check passes, then we check to see if there is another message. Of course, we could have used a counter for the actual password failures to see if the second message is shown exactly by the fourth attempt. However, as we set the input data, the flow is planned such that it must appear at the right time. This is the true sense of defining test cases with input data. If a message appears, then the `checkMessage(id)` function closes the message box. We then check if the login dialog box is closed with the code `If not Dialog("Login").Exist(0) Then`, and it then calls `openApp()` to begin again for the last iteration.

In the last iteration, with the input data on the seventh row (refer to the table in the previous section), the script performs a successful login, and then calls the function `verifyMainWndInitialState()`, as mentioned in the previous section.