Building a reusable component (action)

The controller was designed to load, configure, and execute **Action Based Testing** (**ABT**) or **Business Process Testing** (**BPT**), which are classes built as command wrappers with a common `Run` method in wh the main flow of the Action is implemented. In this section, we will se how to implement such an action.

Getting ready

From the **File** menu, navigate to **New** | **Function Library...**, or use *Alt* + *Shift* + *N* shortcut. Save the file as `cls.Actions.vbs` in the `C:\Automation\Lib` folder.

How to do it...

As mentioned before, a reusable component (action) is a class that implements the command wrapper design pattern. The following code shows the sample class `Login` as a typical example for an Action wit this framework:

```
Class [Login]
    ' ----------------------------------------------------------
    ' Reusable Action: Login
    ' Description: Login to the application
    ' ----------------------------------------------------------
    Public Status
    Public Iteration
    Public StepNum
    Public dt
    Public Details

    Public Function Run()
        me.Details = "Ended with "
        me.Status.[=]0
        '---
        me.Status.[+=]EnterUsername()
        me.Status.[+=]EnterPassword()
        me.Status.[+=]ClickOnLoginButton()

        '--- Report
        Call ReportActionStatus(me)
    End Function

    Function EnterUsername()
        EnterUsername = me.Parent.WebEdit("txtUsername").Set(dt.GetParame
    End Function

    Function EnterPassword()
        EnterPassword = me.Parent.WebEdit("txtPassword").Set(dt.GetParame
    End Function

    Function ClickOnLoginButton()
        ClickOnLoginButton = me.Parent.WebButton("btnLogin").Click
    End Function

    Property Get Parent()
        Set Parent = Browser("MyStore").Page("Main")
    End Property

    Private Sub Class_Initialize
        Call InfoClassInstance(me, C_OBJ_LOADED_MSG)
        Set me.Status = [As Num](0)
    End Sub
    Private Sub Class_Terminate
        Call InfoClassInstance(me, C_OBJ_UNLOADED_MSG)
        Set me.Status = Nothing
    End Sub
End Class
```

> **Note**
>
> The Test Objects referred to in the internal methods of the class are for illustration purposes only.

How it works...

The controller creates an instance of the `Login` class when it finds its name in the `Steps` datasheet and is planned to run (the `RUN` parameter equals `TRUE`).

It then invokes the object's `Run` method, in which the main flow of the Action is coded. Note that additional fields, functions, subroutines, and properties can be added to extend the basic pattern of an action; thus making it a very powerful and flexible tool to encapsulate basic blocks of code, which are usually business-oriented functions. The `Run` method finally invokes the `ReportActionStatus` function, which takes care to send the information accumulated during the process to the UFT reporter.

Note how data is referred to in the internal functions:

```
dt.GetParameter("USERNAME")
```

Through the `dt` field of the Action, which was set by the controller with a reference to the `LocalSheet` DataTable, we can retrieve the values of any of the required parameters.