Customizing mouse operations (DeviceReplay)

The `DeviceReplay` object enables us to perform mouse and keyboard operations using code, for instance `MouseMove`, `MouseClick`, `PressKe` `SendString`, and `DragAndDrop`. Though in past years it was not so we documented in HP's materials, now, thanks to the work of some very dedicated people from the QTP (UFT) community, light has been shed upon the workings of this object.

The `DeviceReplay` object is very important for automation, one reason being that the common Test Object methods that UFT provides, such `Click`, do not always perform well. For example, quite often, one can encounter objects that do not respond to an `onclick` event unless the mouse is actually moved to a location within its bounding rectangle. I such a case, we may revert to the `DeviceReplay` object to override t `Click` method and perform the `MouseMove` operation before sending a `MouseClick` event.

This recipe describes how to use the `DeviceReplay` object to override the `Click` method and perform the `MouseMove` operation before sending a `MouseClick` event.

Getting ready

From the **File** menu, navigate to **New** | **Test**, or use the *Ctrl + N* shortcut, or open an existing function library.

How to do it...

In the function library, write the following code:

```
Public Function MouseClick_(ByRef obj)
    Dim dr
    Dim X, Y, W, H

    If Not obj.GetROProperty("visible") Then
      MouseClick_ = 1
      Reporter.ReportEvent micWarning, "MouseClick_", _
      "Object is not visible."
      Exit Function
    End If

    With obj
      H = .GetROProperty("height")
      W = .GetROProperty("width")
      X = .GetROProperty("abs_x")
      Y = .GetROProperty("abs_y")
    End With

    If X < 0 Or Y < 0 Then
      MouseClick_ = 1
    Else
      X = X+W\2
      Y = Y+H\2
      Set dr = CreateObject("Mercury.DeviceReplay")
      dr.MouseMove X, Y
      Wait 0, 50
      dr.MouseClick X, Y, 0
      Set dr = Nothing
        MouseClick_ = 0
    End If
  End Function
```

Register the function to the relevant Test Object classes, as shown in the *Registering a method to all classes* recipe (`RegisterUserFunc`) in Chapter 4, *Method Overriding*. For example, we could register the function to the `Click` method of the `Image` class (Web) as follows:

```
RegisterUserFunc "Image", "Click", "MouseClick_"
```

This is done so that, every time we invoke a click event for a Web image, it will first move the mouse to trigger the `onmouseenter` event.

How it works…

The function is generic as it takes an object as an argument. It first checks if the object is visible, which is obligatory to perform a `MouseMove` operation. Then it gets its position and dimensions, to check that it is not positioned out of the screen boundaries (this can happen sometimes with the top-left coordinates being less than zero). If these two checks are passed, then it calculates the middle point of the object, instantiates the `DeviceReplay` object, and moves the mouse to its center. A short delay is added, to ensure that the object reacts to the presence of the mouse within its boundaries, and only then, the `MouseClick` is performed.