Synchronizing a test with a web page loading

An essential requirement from any automated test script is that it must synchronize with **Application Under Test** (**AUT**). This is especially true for GUI automation, because the test script actually attempts to emulate the actions that a human user would perform on the application's front end. Hence, it is of utmost importance to take care that operations on a page are carried out when the controls are fully loaded and ready to accept inputs.

How to do it...

All web classes, `Page`, `WebEdit`, `WebButton`, `Image`, and so on, provide an attribute that indicates the state of the element. This attribute is `readystate`, and its value ranges from 0 (uninitialized) to 4 (complete). So basically, to synchronize our script with the full loading of a web element, we need to wait until its `readystate` attribute reaches the value of 4.

The following generic function accepts an element and a timeout period, and it waits until the element reaches the load complete state or timeout:

```
Function WaitUntilComplete(o, timeout)
    Dim iElapsed

    WaitUntilComplete=true
    iElapsed=0
                                    eadystate <> 4
                                    AL_MSEC
                                    ed+C_INTERVAL_MSEC
                                    imeout Then
            reporter.ReportEvent micWarning, "WaitUntilComplete", "Elemen
            WaitUntilComplete=false
            Exit Do
        End If
    Loop
End Function
```

We will call the function in `Action1` as follows:

```
Dim bPageComplete

bPageComplete=WaitUntilComplete(Browser("title:=.*Advanced QTP.*").Page("
```

How it works...

The `WaitUntilComplete` function takes two arguments, the object to synchronize with and the maximum time we are prepared to wait in milliseconds. We initialize this function optimistically, and then loop while the native `readystate` attribute is not equal to 4 (complete), waiting for an interval of `200` milliseconds each time, as defined in the `C_INTERVAL_MSEC` constant. The `iElapsed` variable increases with each cycle, and a check is performed to see if it reaches the timeout. If it reaches timeout, we exit the `Do` loop after assigning the function to return the value `False`. If `true` is returned, then it means that the complete state was reached before the timeout; therefore, our test can continue as planned.

> **Note**
>
> The `readystate` attribute is accessed through the element's `Object` property, which provides access to the native properties

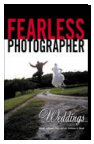
and methods of all TOs except standard Windows objects.

People who finished this

PREV
Identifying eleme…

NEXT
Accessing web ele…

BOOK SECTION



The Ceremony

from: Fearless Photographer Weddings by Joseph Salvatore Prezioso...

Released: August 2010

22 MINS

Photography

BOOK SECTION

Creating Sites and Pages with Dreamweaver CS6

from: Dreamweaver CS6 Mobile and Web Development with HTML5, CSS3, and jQuery Mobile by David Karlins

Released: May 2013

10 MINS

DreamWeaver

BOOK SECTION

Auto discovery

from: Oracle Enterprise Manager Cloud Control 12c: Managing Data Center Chaos by Porus Homi Havewala

Released: December 2012

5 MINS

Oracle

BOOK SECTION

Automation Life Cycle and Automation Goals

from: Designing and Implementing Test Automation Frameworks with QTP by Ashish Bhargava

Released: November 2013

14 MINS

Software Development