Using Descriptive Programming inline

Apart from storing TOs in OR, UFT supports using **Inline Descriptio** (**ID**), a technique also generally known as **Descriptive Programming** (**DP**). As these descriptions are strings, the technique allows for parameterization, and runtime and dynamic identification of objects using data that was predefined or retrieved from AUT during the run session.

Getting ready

Please ensure that the Internet Explorer application is open on Google

How to do it...

Suppose that we need to identify an object like one of the links on the Google search engine, such as YouTube, then click on it, and navigat back to Google.

Proceed with the following steps:

1. If we know the values for identification properties (which we can obtain using the UFT Spy), then we can easily write code to accomplish this as follows:

```
with Browser("micclass:=Browser", "title:=Google").Page("title:=Goo
    If .exist(0) Then
        .highlight
        .click
    End If
    Browser("title:=YouTube").Back
End with
```

2. We may be required to parameterize the `innertext` variable of the `Link` function so that we can run the same test on several links. To achieve this, we will replace the first line with the following:

```
with Browser("micclass:=Browser", "title:=Google").Page("title:=Goo
```

Also, replace the last line with the following:

```
Browser("title:=.*"&DataTable("LinkText", dtLocalSheet)&".*").Back
```

3. We execute this code with a `DataTable` parameter called `LinkText` (refer to the *Creating a DataTable parameter* recipe in Chapter 1, *Data-driven Tests*) with three rows with the values `YouTube™`, `News`, and `Maps`. To set the Action to run on all rows, go to the **Test Flow** pane (from the menu navigate to **View** | **Test Flow**), and then right-click on `Action` and select **Action Call Properties...**, or navigate to **File** | **Settings**, select **Run** tab, and then select the **Run on all rows** option.

Note that in the last case, we concatenated an asterisk (`*`) before and after the `LinkText` parameter to match the `Browser` title, which adds `Google` before `News` and `Maps`. The inline description is treated by UFT as a regular expression by default.

How it works...

Instead of storing the TOs in OR, we use the `Browser`, `Page`, and `Link` objects and provide them either with a parameter string or a series of

strings separated by commas, as shown in the previous section. These strings are used to build the description with which the objects are identified. First, the top level parent object (`Browser`), then `Page`, and finally, `Link` are identified. Take note of the syntax for the comma-separated property-value pairs in some of the descriptions.

Instead of hardcoded strings, such as `YouTube` for `innertext`, we concatenated a value taken from an external data source, namely `DataTable`, ran the test for several iterations (one link per `DataTable` row), and hence, tested that each link leads to the correct target. In our example, we check that our definition actually works by asking if it exists, and only then performing the `Highlight` and `Click` operations.