Code Repository Exercises

State Management

Purpose: The purpose of this exercise is to familiarize you with various mechanisms for preserving state in ASP.NET applications. You will learn how state is preserved on a single page through *ViewState*, and you'll learn how to preserve state across pages using cross-page posting, the querystring, cookies and session variables.

Directory Name: State

Instructions:

Part 1: Maintaining state on a single page

1. Create a new .aspx page called *ViewState.aspx* in the directory created for this exercise. Add the following to this page:
    a. A Label, TextBox, and DropDownList Web Server control. The *EnableViewState* property of each of these controls should be set to *false.* Give the label and the textbox some default text in the content file (e.g. "Hello World"). Also, add two or three ListItems to the DropDownList
    b. An appropriately labeled Button that changes the text of the label to something else (e.g. "Goodbye Everyone")
    c. An appropriately labeled Button that changes the ForeColor property of the label to *System.Drawing.Color.Red*
    d. An appropriately labeled Button that changes the text of the TextBox to something else (e.g. "Goodbye Everyone")
    e. An appropriately labeled Button that changes the ReadOnly property of the TextBox to *true*
    f. An appropriately labeled Button that changes the SelectedIndex property of the DropDownList to the second item in the list.
    g. An appropriately labeled Button that changes the BackColor property of the DropDownList to *System.Drawing.Color.Yellow*
    h. An appropriately labeled Button that simply posts back the page (you just need to add the button to the page; you don't need to write any code for it)
2. Try clicking each button and then clicking the page postback button. Notice which changes are preserved and which changes are lost across page postbacks when Viewstate is disabled for a control. Try setting the *EnableViewState* property of each of the Label, TextBox, and DropDownList controls to *true* and note the difference in behavior.
3. Add a Label (id="lblCount") to your page. Create an event handler method for the click event of the button you created in step 1-h. In this method, implement a Viewstate counter as shown on

page 234-235 of the MacDonald book. This will keep a tally of the number of times the page is posted back in response to the button click.

Part 2: Maintaining state across pages

In this part of the exercise, you will use various ASP.NET state mechanisms, including cross-page posting, the querystring, cookies and session variables, to create a 'Madlib' based on user entries. Download the starting files for this exercise to the *State* directory created for this exercise and complete the following steps for each file:

1. *MadLib1.aspx*: (This page will forward its values to another page, *MadLib2.aspx*, using cross-page posting.)
    a. Add 9 public properties of type *TextBox* to the code-behind file. There should be one public property for each TextBox control on the page. Each property should have only a *get* accessor that simply returns the TextBox object that it corresponds to.
    b. Set the PostBackUrl property of the *btnNext* Button to "MadLib2.aspx". This will cause the page to redirect to *MadLib2.aspx* when the button is pressed.
2. *MadLib2.aspx:* (This page will retrieve the values from *MadLib1.aspx*, store them as session variables, and pass the values of its TextBoxes on to *MadLib3.aspx* in the QueryString)
    a. Add a *PreviousPageType* directive under the page directive. Set the *VirtualPath* attribute of this directive to point to *MadLib1.aspx*. This will enable *MadLib2.aspx* to retrieve values from the public properties of *MadLib1.aspx*.
    b. In the *Page_Load* method, check to see if the *PreviousPage* object is null. If it is **not** null, retrieve the values from the public properties of *MadLib1.aspx* and store each one in a session variable (you will access these later on).
    c. Add an event handler method for the click event of *btnNext*. This method will retrieve the values entered in the TextBoxes on this page (*MadLib2.aspx*), and create a string variable that contains the URL to *MadLib3.aspx* with the TextBox values appended in the querystring. The method should then redirect the browser to the URL stored in the string using Response.Redirect()
3. *MadLib3.aspx:* (This page will retrieve the querystring values from *MadLib2.aspx*, store them as session variables, and pass the values of its Textboxes on to *MadLibFinal.aspx* in a cookie)
    a. In the *Page_Load* method, retrieve the querystring values from *MadLib2.aspx* using the Request.QueryString object. Store each of these values in a session variable.
    b. Add an event handler method for the click event of *btnNext*. This method will retrieve the values entered in the TextBoxes on this page (*MadLib3.aspx*) and create a single cookie that contains each of these values. After the values have been added, the cookie should be added to the Cookies collection of the Response object. Then, send the browser to *MadLibFinal.aspx* using Response.Redirect()
4. *MadLibFinal.aspx:* (This page will retrieve the values from all previous pages and create the final Madlib)
    a. In the *Page_Load* method, do the following

i. Retrieve the cookie created on *MadLib3.aspx*. Assign each value stored in the cookie as the text of the appropriate Label control representing the verbs in the MadLib (e.g. lblVerb1, lblVerb2, lblVerb3, etc.)
ii. Retrieve the noun and adjective values from the session variables you created on *MadLib2.aspx* and *MadLib3.aspx*. Assign these values to the Text property of the appropriate labels on the page.

b. Add an event handler method for the click event of *btnStartOver*. This method should abandon the current session and redirect the browser to *MadLib1.aspx* using Response.Redirect();

Header on Default.aspx: State

Pages linked from Default.aspx: *ViewState.aspx, MadLib1.aspx*