ASP.NET Built-in Objects

Outline of the Lecture

- Introduction
- Response object
- Implementation of C# foreach loop
- Programming Example

Introduction

- Using ASP.NET built-in objects, you can access to information regarding the Web server, the client who is accessing a Web page, the Web application that contains the Web page, and the fields in the HTTP request and response streams.
- The Request, Response, Server, Application, and Session objects are part of ASP.NET and are used in much the same way as they are in ASP. However, in ASP.NET these objects are defined in new classes in the System.Web namespace.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Describes the methods, properties, and collections of the object that stores information related to the entire Web application, including variables and objects that exist for the lifetime of the application.</td>
</tr>
<tr>
<td>Request</td>
<td>Describes the methods, properties, and collections of the object that stores information related to the HTTP request. This includes forms, cookies, etc.</td>
</tr>
<tr>
<td>Response</td>
<td>Describes the methods, properties, and collections of the object that stores information related to the server's response. This includes displaying content, manipulating headers, etc.</td>
</tr>
<tr>
<td>Server</td>
<td>Describes the methods and properties of the object that provides methods for various server tasks. With these methods you can execute</td>
</tr>
</tbody>
</table>
code, get error conditions, encode text strings, create objects for use by the Web page, and map physical paths.

| Session | Describes the methods, properties, and collections of the object that stores information related to the user's session, including variables and objects that exist for the lifetime of the session. |

Response object

Response.Write Method

- Writes a variable or text to the current HTTP output as a string.

Examples:

Example 1 (string)
```<% Response.Write "Hello World" %>```
Output:
Hello World

Example 2.a (string with HTML Tags)
```<% Response.Write("Hello<br/>World") %>``
Output:
Hello World

Example 2.a (string with HTML Tags)
```<% Response.Write "<TABLE WIDTH = 100\%>" %>``

Example 3 (variables and concatenation operators)
```Int myNum = 25;
String myString = "Hello";
Response.Write("myNum = " + myNum + "<br/>");
Response.Write("myString = " + myString + this.TextBox1.Text + "<br/>");```
Implementation of C# foreach loop

- **foreach** loop in C# help us to iterate through elements on a given collection on the simplest possible way so that we can easily access the elements of the collection and do some operations inside the loop.

- **Syntax**

```csharp
foreach (Element_of_Collection Variable_Name in Collection_Name )
{
   //Code
}
```

**Example 4 (using foreach structure with Response.Write)**
```
int[] array = { 5, 10, 15, 20, 25 };
foreach (int number in array) {
   Response.Write(number + "","");
}
```

**Example 5 (using foreach structure with Response.Write and ArrayList object)**
```
ArrayList namesArrayList = new ArrayList();
namesArrayList.Add("Mohammed");
namesArrayList.Add("Khaled");
namesArrayList.Add("Badr");
foreach (string name in namesArrayList) {
   Response.Write(name + "...<br>");
}
```

**Example 6 (Hide all mobile controls using foreach mechanism)**

**Homework!!!!!!!!!!!!!!!!!!!!!!!!!!!!!**

- There are a few things that should be pointed out about this code as compared with the **for** syntax.
  - You don't have to know the bounds/dimensions of the array.
  - It automatically gives you an instance (of the right type) of whatever is collected by the array.
  - The syntax (and therefore the resulting code) is just so much cleaner.
  - This same syntax will work whether you're iterating over an array, ArrayList, Hashtable or any other type of collection.
Programming Example

Example 19-1

```csharp
<script language="C#" runat="server">

private class Accounts
{
    long LUserID;
    string SEmail, SFirst, SLast;
    bool Bstatus;
    public Accounts ( long id, string em, string f, string l, bool s)
    {
        LUserID = id;   SEmail = em;   SFirst = f;
        SLast = l;    Bstatus = s;
    }
    public long UserID
    { get { return LUserID; } }
    public string Email
    { get { return SEmail; } }
    public string First
    { get { return SFirst; } }
    public string Last
    { get { return SLast; } }
    public bool Status
    { get { return Bstatus; } }
}

private void Page_Load ( object sender, EventArgs e)
{
    ArrayList AccountList = new ArrayList ();
    if (!IsPostBack)
    {
        AccountList.Add(new Accounts (1, "User1@yahoo.com", "Sami", "Issa", true));
        AccountList.Add(new Accounts (2, "User2@yahoo.com", "Naji", "Ali", false));
        AccountList.Add(new Accounts (3, "User3@Hotmail.com", "Najeeb", "Issa", true));
        AccountList.Add(new Accounts (4, "User4@yahoo.com", "Fadi", "Fawzi", true));
    }
</script>
Accounts FirstObj = (Accounts)AccountList[0];
Response.Write("<b>First Account Information</b></br>");
Response.Write("<table border='3'>");
Response.Write("<tr ><th>UserID</th><th>Email</th><th>Status</th></tr>");
Response.Write("<tr><td>" + FirstObj.UserID + "</td>");
Response.Write("<td>" + FirstObj.Email + "</td>");
Response.Write("<td>" + FirstObj.Status + "</td></tr>");
Response.Write("</table>");
Response.Write("<ol>");
Response.Write("<b>Account List Information</b></br>");
foreach (Accounts AccouObj in AccountList)
{
    Response.Write("<li>Full Name(First and Last): " + AccouObj.First + "--
                    " + AccouObj.Last + "</li>");
    Response.Write("<ul>");
    Response.Write("<li>Email: " + AccouObj.Email + "</li>");
    Response.Write("</ul>");
}
Response.Write("</ol>");
}