

15 Seconds : User-Friendly Errors

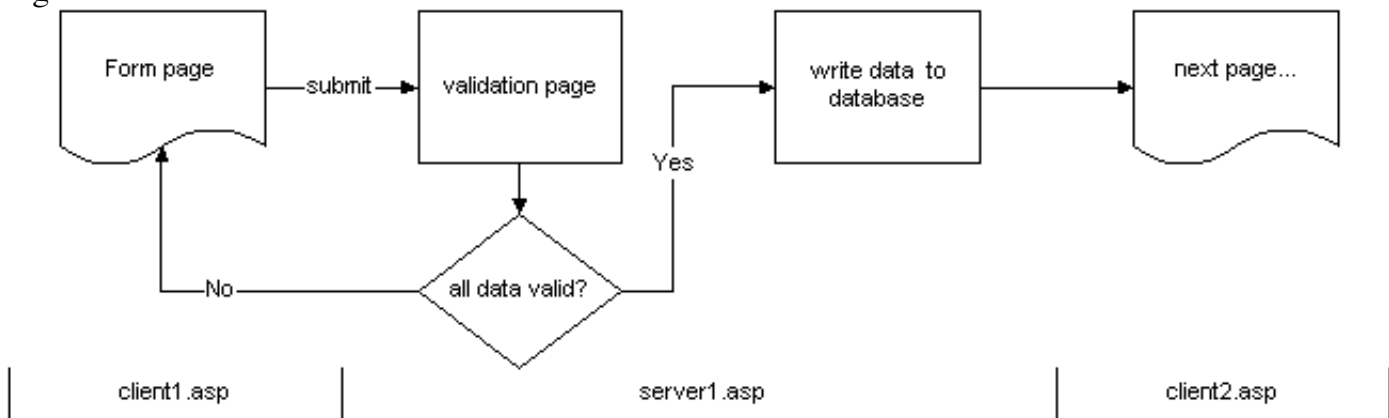
Don Franke
05/20 / 1999

When creating Web-based data-entry forms, data validation is a must. But even with many sites that employ exhaustive data validation techniques, often little time is spent developing exactly how to report validation errors to the user. At these sites, a validation error often means getting the following generic statement on a white page: "One or more errors were found . . . please click your browser's 'Back' button and check the form again." This is hardly an intuitive response, and I'm less than enthusiastic, in this day of fourth- and soon fifth-generation browsers, to click "Back" and visually check each field. I'd rather spend the time with a search engine looking for a competitor's more user-friendly site.

So, when client-side JavaScript form validation won't cut it (i.e. for validating against a database whose data you don't necessarily want accessible in the HTML code), I use the following solution. All code examples, by the way, are for VBScript.

The following is the logical flow:

Figure 1



I break up the data-entry pages into page pairs, one page for the form and the other page for validating and writing to the database. Respectively, we will call these "client1.asp" for client-side and "server1.asp" for server-side.

Client-Side

Client1.asp contains the form, as well as any JavaScript validation. I use JavaScript to check for proper numeric values, date values, or if the field is blank. I enclose the form within a table (<TABLE CELLBORDER=0>) and enclose each form control in its own cell (<TD>.)

Server-Side

Server1.asp receives the values from client1.asp and performs the rest of the validation. The typical validation page has the following top-down structure:

1. Receive values (request.querystring)
2. Validate values
3. If invalid values are found, redirect to previous page.
4. If no invalid values are found:

1. Write values to database
2. Redirect to next page

Step 1 and 2

I dimension an ERROR string, which I build with every invalid value encountered. Example:

```
<%F1 = Request.QueryString("F1")
  If Len(F1)=0 or IsNull(F1) Then ERROR=ERROR & "__F1"%>
```

I delimit the field names in the error string with a double-underscore to distinguish between fields with similar names (i.e., "F1" and "F10").

Step 3

After validating all the received values, the length of the ERROR string is checked. If the length is 0, no errors have been found, and all data is valid. If greater than zero, however, the page is redirected to the previous page. The following is included in the redirection URL:

1. ERROR=error
2. All field values

Example:

```
If Len(ERROR) > 0 Then
    response.redirect("client1.asp?ERROR=" & ERROR & _
        "&F1=" & Server.URLEncode(F1) & "&F2=" & _
        Server.URLEncode (F2) & "&F3=" & _
        Server.URLEncode (F3))
```

Or in URL-speak:

```
Client1.asp?F1=&F2=abc&F3=def&ERROR=__F1
```

Be sure to use the Server.URLEncode function for text field values since these values will be passed in the URL field.

The form page (client1.asp) receives ERROR and checks the length of its value:

```
<%
    If Len(ERROR) > 0 Then
        response.write("Errors have been found . . . " & _
            "please checked the highlighted sections.")%>
```

Since Len(ERROR) is greater than zero, the message "Errors have been found . . . please check the highlighted sections" is drawn at the top of the page.

And now the form. The form elements are all preceded with an IF...THEN statement. The following is an

example for form control "F1."

```
<%If Instr(ERROR, "F1__") > 0 Then
    BG = " BGCOLOR='RED' "
Else
    BG = ""
End If%>
<TD <%=BG%>><INPUT TYPE=TEXT NAME="F1" VALUE="<%=F1%>"></TD>
```

Note that the value of this control matches what was sent to it by the redirect (<INPUT TYPE=TEXT NAME="F1" VALUE="<%=F1%>">). If F1 has a value, it will be shown in the field. This avoids the user having to re-enter all the data. (How many users would stick around for this?) Also note the Instr command. If the field names in the ERROR string weren't delimited by the double-underscore (or something similar), the example F1 Instr would be true if ERROR contained "F1" or "F10." Of course, these examples only apply to text controls; fancier measures have been taken for other types of HTML form controls, such as checkboxes. Lastly, notice that the control is placed in its own table cell (<TD>). This allows the actual control to be highlighted with a <TD BGCOLOR=RED>, showing the user exactly what form controls need correction.

Step 4

When data from client1.asp passes through the server1.asp validation routine without any errors, the database section of server1.asp is executed, updating the database with the data from the form. Then the user is redirected to the next page (i.e., client2.asp.)

Conclusion

And that is basically it. This solution should also work well with server-side ActiveX validation components that return validation errors.

About the Author

Don Franke is a Web developer and programmer at CoAMS, Inc., in Chicago, where he is currently working on the migration from UNIX Informix to Microsoft SQL Server. He was previously a software engineer at 3Com, where he developed an R&D test group intranet using ASP and Informix for Windows NT. His main development tool is Microsoft's Development Studio 6.0, which he uses for ASP and Active X DLL development and soon object-oriented programming in Visual Basic. You can find him in the January 1999 issue of Windows NT magazine, in which he won an award as one of 1998's top Windows NT innovators of the year. He can be contacted at donfranke@yahoo.com.

[Back to article](#)