VisCis: Visual Studio "Code in Strings"

Paul Roper
School of Business and Computer Technology
Nelson Marlborough Institute of Technology
Nelson, NZ
proper@nmit.ac.nz

When developing web applications using Visual Studio .NET there are times when the programmer is coding either SQL or Javascript within text strings. Consequently such code is never checked for correct syntax before it is run. As far as Visual Studio is concerned, the contents of a string are irrelevant. As long as the code is inside matching quotes, it is simply a valid string.

There is an irony here in that Visual Studio is a sophisticated tool with powerful features for developers such as highlighting syntax errors as lines are typed. However, the SQL and Javascript code inside strings is just as prone to errors as any other code but these errors are only discovered at run time. In terms of time taken to find and fix, they probably consume relatively more time per lines of code than the VB or C# code that is thoroughly checked at edit-time. Both SQL and Javascript are necessary for any non-trivial web application although the use of stored procedures in MSSQL removes SQL code from program code. However, students often start out using an Access database without support for stored procedures. A facility to allow Visual Studio to syntax check "code in strings" is extremely useful as catching an error at edit-time is far more efficient than discovering the error at run-time.

The writer decided to investigate what is required to achieve this. Having developed a Javascript parser, and being aware of several third party SQL parsers (with source code), extracting the strings is the major challenge. This is relatively easy for "static" strings but more difficult for strings created dynamically. This paper describes the issues that evolved developing VisCis, Visual Studio Code in Strings. Ultimately, it is envisaged that it could be integrated into Visual Studio by way of an "add-in".

Keywords

ASP.NET, Visual Studio, SQL, Javascript, Parsing

1. INTRODUCTION

If we look at the code-behind editor in Visual Studio when developing an ASP.NET application, we see instructions written in the chosen language, usually VB or C#. Visual Studio is excellent at dealing with the programming language in terms of continuously syntax checking in the background and offering code completion (intelli-sense) at appropriate times. However, often string literals contain programming code in other languages such as SQL and Javascript. This code is ignored by Visual Studio, and consequently can be the source of run-time errors. The writer wanted to see if another program could extract this "code in strings" and syntax check it at edit-time, potentially saving developer time and frustration.

1.1 SQL in ASP.NET program code

SQL instructions have long been part of web application code (Java, ASP, PHP etc). Both students and experienced developers frequently experience problems with the syntax of embedded SQL particularly unmatched quotes, inadvertent use of a reserved word for a field name and the confusing line continuation mechanism of VB (_). To debug the SQL in isolation, it is often copied and pasted into a query (or query analyser) to check the result. Usually, the "copy" is complicated as the SQL is not a contiguous whole, but on multiple lines or segments appended into one string, so the copy becomes many copies. Another approach is to display the SQL string with a Response.Write before the error occurs, and if the error is still obscure, copying and pasting this into a query. Consequently, trying to find the source of an embedded SQL error can be frustrating and slow.

1.2 Javascript in ASP.NET program code

ASP.NET embraces Javascript to enable the sophisticated controls (such as the Datagrid) to function across browsers. In addition the ASP.NET languages provide two methods for the developer to emit Javascript into the HTML page. During Page_Load, RegisterClientScriptBlock() and/or RegisterStartupScript() would be called and the

Javascript is inserted into the page. These are not the only techniques for putting Javascript into a page, using a simple Response.Write() through to sophisticated developer generated classes and methods can achieve this exactly how the developer desires.

Regardless of how the Javascript is created, it starts life as string literals and if it is faulty, the browser will indicate an error at a particular line number. This line number pertains to the generated HTML that can be seen by viewing the source. Internet Explorer shows it in Notepad, a simple editor that does not provide line numbering, so the text then has to be copied and pasted into an editor with line numbering to locate the line producing the error. Non-obvious Javascript errors can be time consuming and frustrating, particularly for students.

2. VISCIS

As a developer and tutor, I would rather spend time on more interesting problems than looking for missing commas or unmatched quotes. As a result, I wrote VisCis in Delphi 5, basing it on an existing Javascript authoring tool (ScrypTik) and incorporating two parsers, attempting to find errors within "code in strings". For SQL I tried gaSQLParser, open source components from Gert Kello and for Javascript, my own previously developed units/classes. The SQL component gave memory Access Violations and necessitated a look for alternatives. A completely satisfactory SQL parser is still to be found.

In order to extract "code in strings", I needed to parse the primary language to locate the strings. I started with C# and will deal with VB if I develop further. This is not a full parser, it only looks for the obvious clues to SQL and Javascript strings. Further development will make this more sophisticated so that dynamically constructed strings can be identified. Initially, the intention is for VisCis to execute alongside Visual Studio, both IDE's would have the same .cs file open. Both applications detect if a file has been changed by the other and prompt for a reload, so keeping synchronised should not be a problem.

3. SQL EXAMPLE

SQL code is inserted into a new editor with SQL syntax highlighting and comments to describe the file it is from and the line number. So C# code such as this:

```
    OleDbCommand aCommand = new
OleDbCommand("select * from
emp_test", aConnection);
```

appears in an editor looking like this:

```
# SQL extracted from csharpsql.cs
# From Line number 12:
select * from emp_test
```

This code is passed to the SQL parser, which locates any syntax errors.

The key to finding the above SQL is the declaration and creation of an OleDbCommand object. This applies to SqlCommand as well.

The first parameter string is the SQL code. This is usually a variable set up on preceding lines, not a literal, so there is further work to determine the contents of this variable.

4. JAVASCRIPT EXAMPLE

C# code such as this:

```
RegisterClientScriptBlock("clientScript",
"<script language=Javascript> func-
tion DoClick()
{Form1.show.value='Welcome to
Microsoft .NET'}</script>");
```

appears in the form .aspx file inserted before the </head> tag, opened in a new editor with both HTML and Javascript syntax highlighting thus:

```
<HTML>
     <HEAD>
          <title>WebForm1</title>
```

```
          <meta content="Microsoft Visual Studio .NET 7.1" name="GENERATOR">
          <meta content="C#" name="CODE_LANGUAGE">
          <meta content="JavaScript" name="vs_defaultClientScript">
          <meta content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
<script language=Javascript>
 function DoClick() {Form1.show.value='Welcome to Microsoft .NET'}
</script>
     </HEAD>
     <body onload="DoClick();">
          <form id="Form1" method="post" runat="server">
               <asp:label id="Label1" runat="server">Test for VisCis locating
Javascript</asp:label><br>
               <asp:TextBox id="show" runat="server" Width="498px"></
asp:TextBox></form>
     </body>
</HTML>
```

This is parsed, including matching up the external reference to a HTML element, "Form1.show.value" with the asp:TextBox with an id of "show" on the form with an id of "Form1". The key to finding the Javascript is the RegisterClientScriptBlock () and/or RegisterStartupScript () methods. The second parameter string is the Javascript code. As with SQL, this is usually a variable set up on the preceding lines. Perhaps a better way to find Javascript code in strings will be to examine every string literal for <script language="javascript"> tags and work from there.

5. CONCLUSION

While it is impossible to achieve 100% code perfection, even in a modern IDE like Visual Studio that is extremely good at catching edit-time errors, every bit of early detection does save time. I was curious as to what could be achieved and although VisCis is a "work in progress" results so far look promising. Feedback from other developers and tutors will be useful for determining what to do next. If further development seems warranted, it would eventually be ideal to take the parsers out of VisCis and make them part of an add-in for Visual Studio thereby enabling the extra checking functionality within the primary IDE.

The alpha version will be made available to those interested. It may be downloaded from a website with a forum for discussion. Email the writer if this interests you.

REFERENCES

Birdwell at al (2002) Beginning ASP.NET 1.0 with Visual Basic .Net, Wrox Birmingham

Duthie G A (2003) Microsoft ASP.NET Programming with Visual C# .NET Step by step, Microsoft Redmond