Dave's Development Blog

Software Development using Borland / Codegear / Embarcadero RAD Studio



# Searching Libraries for Units

By David | August 26, 2019 0 Comment

Yesterday I eventually got around to implementing a function of one of my IDE plug-ins that was missing. In the ~~Debug with CodeSite~~ plug-in, one of the missing checks was to ensure that `CodeSiteLogging` was on the search path either for the project or the IDE. This is similar to a question that was asked on DelphiPraxis recently.

The routine worked out to be not too complicated as follows (this is test code not final code and probably needs refactoring into smaller chunks):

```
Class Procedure TDDTOpenToolsAPIFunctions.CheckLibraryPath;
ResourceString
  strCodeSiteLoggingNotFound = '''CodeSiteLogging.dcu'' not found in your library
paths!';
Const
  strProjectSrcDir = 'SrcDir';
  strDCCLibraryPath = 'LibraryPath';
  strCodeSiteLoggingDcu = 'CodeSiteLogging.dcu';
Var
  slSearchLibrary: TStringList;
  PO: IOTAProjectOptions;
  S : IOTAServices;
  BDSMacros: TRegEx;
  M: TMatch;
  slMacros: TStringList;
  iMacro: Integer;
Begin
  slSearchLibrary := TStringList.Create;
  Try
    If Supports(ActiveProject.ProjectOptions, IOTAProjectOptions, PO) Then
      slSearchLibrary.Add(StringReplace(VarToStr(PO.Values[strProjectSrcDir]), ';',
```

```
#13#10,
        [rfReplaceAll]));
    If Supports(BorlandIDEServices, IOTAServices, S) Then

slSearchLibrary.Add(StringReplace(VarToStr(S.GetEnvironmentOptions.Values[strDCCLibrary
Path]), ';',
        #13#10, [rfReplaceAll]));
    slMacros := TStringList.Create;
    Try
      BDSMacros := TRegEx.Create('\$\((\w+)\)', [roIgnoreCase, roMultiLine,
roCompiled]);
      For M In BDSMacros.Matches(slSearchLibrary.Text) Do
        Begin
          If slMacros.IndexOfName(M.Groups.Item[1].Value) = -1 Then
            slMacros.AddPair(M.Groups.Item[1].Value, StringReplace(
              GetEnvironmentVariable(M.Groups.Item[1].Value), '\', '\\',
[rfReplaceAll]));
        End;
      For iMacro := 0 To slMacros.Count - 1 Do
        Begin
          BDSMacros := TRegEx.Create('\$\(' + slMacros.Names[iMacro] + '\)',
            [roIgnoreCase, roMultiLine, roCompiled]);
          slSearchLibrary.Text := BDSMacros.Replace(slSearchLibrary.Text,
slMacros.ValueFromIndex[iMacro]);
        End;
    Finally
      slMacros.Free;
    End;
    If FileSearch(strCodeSiteLoggingDcu, StringReplace(slSearchLibrary.Text, #13#10,
';',
      [rfReplaceAll])).Length = 0 Then
      OutputMsg(strCodeSiteLoggingNotFound);
  Finally
    slSearchLibrary.Free;
  End;
End;
```

In the DelphiPraxis discussion it was recommended to use the `IOTABuildConfigurations` interface to get the projects options however in my tests I found that the above interface would not return the information I wanted, in fact, it returned very little of use (something to check another time as this is supposed to be the new way to get project options). So I decided to use the older technical of using the `IOTAProjectOptions` interface and this allowed me to find the project's search path using the `SrcDir` value name.

I take this information and add it to a string list and changes the `;` characters for line-feed and carriage returns so each directory is on a separate line.

I then repeat this with the `IOTAEnviromentOptions` to get the Win32 library path using the `Library` value name and again add this to the string list in the same manner. In both circumstances above I was not able to use the string constants in either `DCCStrs` or `CommonOptionsStrs` so more investigation is required as I'm sure I've done this before and it worked (note: I'm using 10.3.2).

Now the directories usually contain various IDE macros ( `$(BDSPROJECTSDIR)` for instance) that correspond to environment variables and these need to be resolved before we can search.

I ended up using a regular expression to find all the macros and add them to another string list along with the resolved paths. Now I used the `GetEnivironmentVariable()` function of the `SysUtils` unit however Uwe pointed out that I could have done this using the `IOTAServices.ExpandRootMacro()` method from the Open Tools API. You learn something new everyday 🙂

Once I'd got all the macros resolved I replace all the occurrences of the macros in the original string list. Then I get a `;` separated list of the resolved search paths and pass that to the `FileSearch()` function to see if the file I want ( `CodeSiteLogging.dcu` ) exists.

regards
Dave.

Category: Debug with CodeSite Delphi Open Tools API RAD Studio Tags: IOTABuildConfiguration, IOTAEnvironmentOptions, IOTAProjectOptions, IOTAProjectOptionsConfiguration, IOTAServices