Dave's Development Blog

Software Development using Borland / Codegear / Embarcadero RAD Studio

Go straight to JAIL! DO NOT Pass GO! DO NOT Collect £200

By David | December 13, 2017 0 Commen

I've been lazy and as a consequence my parsers are not Unicode compliant and there's no real reason for not doing a proper job. When I was originally converting my code to Unicode I didn't spend much time understanding Unicode, I just did the bare minimum.

Stefan contacted me about my Browse and Doc It plug-in which he is evaluating and pointed out that he could not use umlauts in identifiers.

There are two parts to the solution; the first containing the lazy bit. When I defined my Open Tools API function for extracting the source code from an `IOTAModule` (via `IOTASourceEditor`) I wrote the following:

```
Function EditorAsString(Const SourceEditor : IOTASourceEditor) : String;

Var
  Reader : IOTAEditReader;
  iRead : Integer;
  iPosition : Integer;
  strBuffer : AnsiString;

Begin
  Result := '';
  Reader := SourceEditor.CreateReader;
  Try
    iPosition := 0;
    Repeat
      SetLength(strBuffer, iBufferSize);
      iRead := Reader.GetText(iPosition, PAnsiChar(strBuffer), iBufferSize);
      SetLength(strBuffer, iRead);
      Result := Result + String(strBuffer);
      Inc(iPosition, iRead);
    Until iRead < iBufferSize;
  Finally
    Reader := Nil;
  End;
End;
```

Can you see the problem? I used a cast to get the `AnsiString` to a `UnicodeString`. For plain English and American this didn't present a problem but for other languages (like German) the accented characters would come through as unreadable. So the first thing was to fix this as follows:

```
Function EditorAsString(Const SourceEditor : IOTASourceEditor) : String;

Var
  Reader : IOTAEditReader;
  iRead : Integer;
  iPosition : Integer;
  strBuffer : AnsiString;

Begin
```

```
  Result := '';
  Reader := SourceEditor.CreateReader;
  Try
    iPosition := 0;
    Repeat
      SetLength(strBuffer, iBufferSize);
      iRead := Reader.GetText(iPosition, PAnsiChar(strBuffer), iBufferSize);
      SetLength(strBuffer, iRead);
      Result := Result + UTF8ToUnicodeString(strBuffer);
      Inc(iPosition, iRead);
    Until iRead < iBufferSize;
  Finally
    Reader := Nil;
  End;
End;
```

And now the accented characters appear correctly in the error messages as the characters that are written in the IDE editor. This is used by all the parsers when in the IDE so this fix affects them all.

So the next problem was… How on earth to I identify accented characters as valid letters while parsing the code? To date I have been using a character set construct as follows:

```
Const
  strTokenChars : Set Of AnsiChar = ['#', '_', 'a'..'z', 'A'..'Z'];
```

Notice there are Ansi characters not Unicode characters. To check these against a Unicode string you need to use the function `CharInSet()` which I've wrapped in an `inline` function as follows (was for supporting different IDEs pre and post Unicode):

```
Function IsInSet(Const C : Char; Const strCharSet : TSetOfAnsiChar) : Boolean; InLine;

Begin
  Result := CharInSet(C, strCharSet);
End;
```

Thus the parser calls the following:

```
      Else If IsInSet(ch, strTokenChars) Then
```

To fix the Unicode compliance all I had to do was add a second condition as follows:

```
      Else If IsInSet(ch, strTokenChars) Or IsLetter(ch) Then
```

Suddenly all is fixed.

I've done this for the Object Pascal parser but now need to do this for the others as well. If anyone wants an early fix I can provide a beta version of the appropriate DLL but there is still some work to be done before I can release the current version (finish implementing metrics and checks).

regards
Dave

Related posts:

1. Chapter 5: Useful Open Tools Utility Functions (17.4)
2. Chapter 10: Reading editor code (17.1)
3. Chapter 11: Writing editor code (8.4)
4. RAD Studio Custom Editor Sub-views (6.2)
5. Chapter 16: Getting help when there's no help (5.7)

Category: Browse and Doc It Open Tools API RAD Studio Tags: IOTAEditReader, IOTASourceEditor