**How to parse source code using the DXCore integrated code parsers**

April 11th, 2011

There are times when you need to parse specific source files or blocks of code. Obviously, the [DXCore Framework](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/" \o "DevExpress DXCore Framework" \t "_blank) has many built-in parsers for [various programming languages](http://www.skorkin.com/2010/06/ide-tools-supported-programming-languages/). They can be used inside the Visual Studio environment, or outside an IDE in any other application type, such as a *Console App*, for example. Later, this kind of app (a *Console App*) can be used in the project building process for code validation, code clean-up, automatic refactoring and any other task.

Let’s see how can we parse the code inside a [DXCore plug-in](http://www.skorkin.com/2010/08/dxcore-plug-ins-overview/" \t "_blank" \o "DXCore plug-ins overview) when it is loaded into the Visual Studio IDE. Here, we can use the [Language](http://www.skorkin.com/2011/04/dxcore-services-language/) and the [Source Model](http://www.skorkin.com/2011/04/dxcore-services-source-sourcemodel/)**DXCore** [services](http://www.skorkin.com/2010/09/dxcore-services-list/) with several APIs specific for the parsing of the code:

* Parsing of the files outside Visual Studio:

*CodeRush.Language.Parse(“FileNameGoesHere”)* – parses the specified source file on disk and returns the LanguageElement that specifies it (most likely, a *SourceFile* instance).

* Parsing of the opened text documents (opened source files):

*ParseActiveDocument*, *Parse(TextDocument)* or *ParseDocument(TextDocument)* (from the [Language](http://www.skorkin.com/2011/04/dxcore-services-language/)**DXCore**service) – parses the specified text document (an opened source file) and returns the [LanguageElement](http://www.skorkin.com/2011/01/dxcore-abstract-source-tree-structure/" \o "DXCore abstract source tree structure" \t "_blank) that specifies it (a *SourceFile* instance).

*ParseIfTextChanged* (from the [SourceModel](http://www.skorkin.com/2011/04/dxcore-services-source-sourcemodel/" \o "DXCore Services – Source (SourceModel)" \t "_blank) **DXCore** service) – parses the active or the specified document if the text has been changed (but not necessarily committed) since the last parse. Plug-in authors can call this method to ensure that the **DXCore** structural image is in sync with the file. For example, you might call this immediately after some text edits.

* Specific parsing methods from the Language service:

*ParseExpression* – parses an expression from the given string.

*ParseString* – parses the specified string and returns the LanguageElement that specifies it.

*ParseTypeReferenceExpression* – parses a type reference expression from the given string.

* Other *Parse* method overloads:

These take additional parameters, such as *ParserContext*, *RegionDirective*, *CompilerDirective*, *SourceRange*, *TextStringCollection*, etc. – these methods parse the specified source range of the given text document with the given context, and returns the *LanguageElement* that specifies it. Nodes parsed in the specified range will be appended to the end of the context’s nodes. Does not trigger the *BeforeParse* or *AfterParse* events, nor does this method call *BindToCode* – the calling client code must do that (this allows the calling code to bind only the nodes within the parse range, and also append any trailing nodes to the end of the newly-parsed nodes).

For example, let’s imagine we’re going to release an open source project, but it is required to clean-up the source files from the comments. Here’s a sample on how to achieve this for a specific file:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | string fileName = "FileNameGoesHere";  SourceFile parsedFile = CodeRush.Language.Parse(fileName) as SourceFile;  if (parsedFile != null)  {    CleanUpFromComments(parsedFile);  }    // ...    /// <summary>  /// Removes all comments from the given file. Uses DXCore element enumerations methods and the DXCore File service to change the file.  /// </summary>  /// <param name="parsedFile">The file to clean-up.</param>  private void CleanUpFromComments(SourceFile parsedFile)  {    LanguageElementType[] commentTypes = new LanguageElementType[] { LanguageElementType.Comment, LanguageElementType.XmlDocComment };    ElementEnumerable elementEnumerable = new ElementEnumerable(parsedFile, commentTypes, true);    FileChangeCollection fileChanges = new FileChangeCollection();    foreach (LanguageElement comment in elementEnumerable)      fileChanges.Add(new FileChange(parsedFile.FilePath, comment.Range, String.Empty));    CodeRush.File.ApplyChanges(fileChanges);  } |

[Show Visual Basic code… »](http://www.skorkin.com/2011/04/how-to-parse-source-code-using-the-dxcore-integrated-code-parsers/#SID3123_1_tgl)

Don’t forget, that you can also use the **DXCore standalone parser** assembly (*DevExpress.DXCore.Parser.dll*), to parse the source code outside Visual Studio. Here’s a sample above, corrected for using the **DXCore** parsers in any type of application:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | string fileName = "FileNameGoesHere";  string extension = Path.GetExtension(fileName);  ParserBase parser = ParserFactory.CreateParserForFileExtension(extension);  if (parser != null)  {    SourceFile parsedFile = parser.ParseFile(fileName) as SourceFile;    if (parsedFile != null)    {      CleanUpFromComments(parsedFile);    }  } |

[Show Visual Basic code… »](http://www.skorkin.com/2011/04/how-to-parse-source-code-using-the-dxcore-integrated-code-parsers/#SID3123_2_tgl)

However, we can’t use [DXCore services](http://www.skorkin.com/2010/09/dxcore-services-list/" \o "DXCore overall services list" \t "_blank) outside Visual Studio. So, in the sample above, the [File service](http://www.skorkin.com/2011/02/dxcore-services-file/) won’t be available for modification of files. We will use another technique in this case – for example, directly change the text of the file.

There are two **DXCore** plug-ins for the *CSharp* and *Visual Basic* languages, and two *Windows Forms Applications* for the same languages are [attached as a sample](http://www.skorkin.com/files/2011/04/ParseSourceCodeSample.zip) (47,780 bytes). The *Windows Forms Application* projects uses a different mechanism for changing files, you are welcome to use any algorithm by your preference.