## DevExpress DXCore framework services list

September 27th, 2010

Here is the list of **services** (and a few properties) **DevExpress** [DXCore](http://skorkin.com/2010/06/devexpress-dxcore-framework/) provides, which are accessible through the main [CodeRush object](http://skorkin.com/2010/11/coderush-object-for-accessing-to-dxcore-services/)**:**

|  |  |
| --- | --- |
| Name | Description |
| [ActionHint](http://skorkin.com/2010/11/dxcore-services-actionhint/) | Provides methods for manipulating of [action hints](http://skorkin.com/2010/11/dxcore-components-actionhint/) and tweaking its settings for the current Visual Studio session. |
| **Actions** | This is not a service, but a property to access the collection of [actions](http://skorkin.com/2010/09/dxcore-components-action/) registered in the **DXCore**. |
| [Adornments](http://www.skorkin.com/2012/06/dxcore-services-adornments/) | Provides methods and properties to the [graphic adornments architecture](http://skorkin.com/2010/10/dxcore-adornments-architecture/). |
| **ApplicationObject** | This is not a service, but a useful property that returns an **EnvDTE** object of the **DXCore** add-in instance. Note that it is hidden from Intellisense. |
| [Assembly](http://www.skorkin.com/2012/06/dxcore-services-assembly/) | Provides access to plug-in assemblies of the DXCore Framework. Note that it is hidden from Intellisense. |
| [BigHint](http://www.skorkin.com/2010/12/dxcore-services-bighint/) | Provides methods for manipulating of big hints such as creating new ones and showing at the specified position. |
| [Breakpoint](http://skorkin.com/2010/09/dxcore-services-breakpoint/) | Provides methods for retrieving and toggling breakpoints in the IDE code editor. |
| [Caret](http://www.skorkin.com/2011/08/dxcore-services-caret/) | Provides methods and properties for the code editor text caret (cursor). Allows moving, hiding, showing the caret and making text operations such as text insertion, deletion, and retrieving for the current caret position. |
| [Clipboard](http://www.skorkin.com/2011/05/dxcore-services-clipboard/) | Provides access to the Windows clipboard. Contains methods for checking, retrieving, copying, cutting and pasting clipboard contents into the code editor. |
| [CodeMod](http://www.skorkin.com/2012/02/dxcore-services-codemod/) | Code modification service provides methods for the code generation and code provider extensions. |
| [CodeStyle](http://www.skorkin.com/2011/11/dxcore-services-codestyle/) | Provides properties for the current code style settings, specified on the “**Editor | Code Style**” category option pages in the [Options Dialog](http://skorkin.com/2010/08/ide-tools-options-dialog/). |
| [Color](http://skorkin.com/2010/09/dxcore-services-color/) | Provides methods for operations with a color, such as blending, brightening, inverting, etc. |
| [Command](http://www.skorkin.com/2011/10/dxcore-services-command/) | Provides access to the [DXCore actions](http://skorkin.com/2010/09/dxcore-components-action/) and Visual Studio command services. |
| [Constants](http://skorkin.com/2010/10/dxcore-services-constants/) | Contains **DXCore** global constants for text commands. |
| [Content](http://www.skorkin.com/2011/11/dxcore-services-content/) | Provides services for content providers, such as refactoring provider, code provider, and navigation provider. |
| [Context](http://skorkin.com/2010/09/dxcore-services-context/) | Responsible for the **DXCore** contexts system. Provides access to context providers and methods that report whether specified context is satisfied. |
| [Contract](http://www.skorkin.com/2011/06/dxcore-services-contract/) | Provides access to the registered contract providers used in the [Add Contract](http://www.skorkin.com/2011/02/code-providers-add-contract/) code provider. |
| [Debugger](http://skorkin.com/2010/10/dxcore-services-debugger/) | Provides access to the Visual Studio debugger and its current mode properties. |
| [Designer](http://skorkin.com/2010/10/dxcore-services-designer/) | Provides methods for Windows Forms design-time support. |
| [Dialogs](http://skorkin.com/2010/10/dxcore-services-dialogs/) | Provides methods for a **DXCore** internal dialogs manipulation. |
| [Documents](http://www.skorkin.com/2011/11/dxcore-services-documents/) | Provides methods and properties for text documents manipulation. |
| [DynamicLists](http://skorkin.com/2010/10/dxcore-services-dynamiclists/) | Provides methods for working with dynamic lists. Note, that it is only used internally, and hidden from Intellisense. |
| [Editor](http://skorkin.com/2010/10/dxcore-services-editor/) | Provides methods and properties for the IDE code editor. |
| [EditPoints](http://www.skorkin.com/2011/11/dxcore-services-editpoints/) | Provides methods for manipulating **DXCore** EditPoints. Note that it is hidden from Intellisense. |
| [Feature](http://www.skorkin.com/2012/06/dxcore-services-feature/) | Provides methods for working with the [What Happened window](http://skorkin.com/2010/08/ide-tools-what-happened-feature-ui-window/). |
| [File](http://www.skorkin.com/2011/02/dxcore-services-file/) | Provides methods for file reading, writing, editing, etc. |
| [Graphics](http://www.skorkin.com/2011/10/dxcore-services-graphics/) | Provides methods for GDI graphic manipulation. |
| [GUIDs](http://skorkin.com/2010/11/dxcore-services-guids/) | Provides access to the window GUIDs. |
| [Hints](http://www.skorkin.com/2011/12/dxcore-services-hints/) | Provides access to hinting services for showing shortcut hints, big feedback, and other discoverability services. |
| [History](http://skorkin.com/2010/11/dxcore-services-history/) | Stores execution history lists of [text commands](http://www.skorkin.com/2011/01/dxcore-text-commands-list/), [string providers](http://www.skorkin.com/2011/08/dxcore-standard-string-providers-list/) and regex aliases. |
| [IDE](http://skorkin.com/2010/11/dxcore-services-ide/) | Provides access to the main Visual Studio IDE. |
| [Intellassist](http://www.skorkin.com/2011/10/dxcore-services-intellassist/) | Provides methods for [Intellassist](http://www.skorkin.com/2011/10/coderush-code-editor-auto-complete-features-intellassist/) manipulation (e.g. suspending, resuming, canceling) and corresponding properties. |
| [Issues](http://www.skorkin.com/2011/07/dxcore-services-issues/) | Provides methods and properties for the [Code Issues engine](http://www.skorkin.com/2011/02/coderush-code-issues-technology-overview/). |
| [Key](http://www.skorkin.com/2011/06/dxcore-services-key-keyboard/) | Provides access to keystrokes, key names, and key mapping methods and properties. |
| [Language](http://www.skorkin.com/2011/04/dxcore-services-language/) | Provides access to the programming language service, including access to **LanguageExtensions**. |
| [LinkedIdentifiers](http://www.skorkin.com/2010/12/dxcore-services-linkedidentifiers/) | Provides access to the [LinkedIdentifers](http://www.skorkin.com/2010/12/dxcore-linked-identifiers-feature/) engine manipulation. |
| [Markers](http://www.skorkin.com/2011/07/dxcore-services-markers/) | [Markers](http://skorkin.com/2010/09/navigation-markers/) services (e.g., dropping, swapping, and collecting markers in source code). |
| [Menus](http://www.skorkin.com/2011/05/dxcore-services-menus/) | Provides access to **DXCore** and Visual Studio menus. |
| [Navigation](http://www.skorkin.com/2011/05/dxcore-services-navigation/) | Provides access to the navigation providers’ engine. |
| [Options](http://www.skorkin.com/2011/02/dxcore-services-options/) | Provides methods and properties for manipulation of [options pages](http://www.skorkin.com/2010/12/dxcore-plug-in-option-pages-overview/) and [decoupled storages](http://www.skorkin.com/2010/12/dxcore-decoupledstorage-object-for-storing-data/). |
| [Outline](http://www.skorkin.com/2011/03/dxcore-services-outline/) | Provides access to the editor outlining allowing you to collapse, expand, and toggle outlining. |
| [PlugInExtensions](http://www.skorkin.com/2012/02/dxcore-services-plugin-extensions/) | Provides access to registered plug-in extensions. |
| [Profiling](http://www.skorkin.com/2012/06/dxcore-services-profiling/) | Provides access to profiling (internal diagnostic) tasks. |
| [Progress](http://www.skorkin.com/2012/06/dxcore-services-progress/) | Provides access to the **Progress Visualizer** engine. |
| [Project](http://www.skorkin.com/2010/12/dxcore-services-project/) | Provides access to the start-up project, active project, and count of opened projects. |
| [ProjectItems](http://www.skorkin.com/2011/03/dxcore-services-projectitems/) | Contains a single property returning the project item for the active document. |
| [Refactoring](http://www.skorkin.com/2011/12/dxcore-services-refactoring/) | Provides methods and properties for refactoring support. |
| [Reflection](http://www.skorkin.com/2011/09/dxcore-services-reflection/) | Provides access to useful .NET reflection methods. |
| [RegEx](http://www.skorkin.com/2011/07/dxcore-services-regex-regular-expressions/) | Provides access to the regular expression search engine. |
| [Resources](http://www.skorkin.com/2011/02/dxcore-services-resources/) | Contains shared resources for use in plug-ins. |
| [Searchers](http://www.skorkin.com/2011/06/dxcore-services-searchers/) | Provides access to **Rename** as well as **Navigation**[Search Providers](http://www.skorkin.com/2011/06/dxcore-components-searcherprovider/). |
| [Selection](http://www.skorkin.com/2011/08/dxcore-services-selection/) | Text selection and active edit point services. |
| [Serialization](http://www.skorkin.com/2011/04/dxcore-services-serialization/) | Provides methods for objects serialization (saving and loading). |
| [SmartTags](http://www.skorkin.com/2011/10/dxcore-services-smarttag/) | Provides access to the smart tags manipulation methods. |
| [Solution](http://skorkin.com/2010/10/dxcore-services-solution/) | Provides methods for solution items manipulation, such as adding a file, directory, etc. |
| [Source](http://www.skorkin.com/2011/04/dxcore-services-source-sourcemodel/) | Provides access to the source model services. |
| [Strings](http://www.skorkin.com/2011/11/dxcore-services-strings/) | Provides methods for manipulation of [string providers](http://www.skorkin.com/2011/08/dxcore-standard-string-providers-list/). |
| [StrUtil](http://www.skorkin.com/2011/09/dxcore-services-string-utilities/) | Provides strings utility methods. |
| [Synchronization](http://www.skorkin.com/2011/07/dxcore-services-synchronization/) | Provides methods for synchronizing thread code on Visual Studio’s foreground thread. |
| [Templates](http://www.skorkin.com/2011/04/dxcore-services-templates/) | Provides [CodeRush code template](http://www.skorkin.com/2012/04/coderush-code-templates-overview/) and text expansion services. |
| [Test](http://www.skorkin.com/2011/12/dxcore-services-test/) | Provides services for the functional tests runner. |
| [TextBuffers](http://www.skorkin.com/2012/06/dxcore-services-text-buffers/) | Provides low-level access to text buffers open (visibly or invisibly) in Visual Studio. |
| [TextClassification](http://www.skorkin.com/2012/02/dxcore-services-text-classification/) | Provides methods for Visual Studio 2010 text classification manipulation. |
| [TextCommands](http://www.skorkin.com/2011/03/dxcore-services-textcommands/) | **Text Commands** services (used internally by the **DXCore** to register and access [text commands](http://www.skorkin.com/2011/01/dxcore-text-commands-list/)). |
| [TextExpansions](http://www.skorkin.com/2011/01/dxcore-services-textexpansions/) | Provides methods and properties for expanding dynamic text (e.g., [CodeRush Templates](http://www.skorkin.com/2012/04/coderush-code-templates-overview/)). Note that it is hidden from Intellisense. |
| [TextFields](http://www.skorkin.com/2012/06/dxcore-services-text-fields/) | Provides access to the [Text Fields](http://www.skorkin.com/2011/01/ide-tools-terminologykey-concepts/) engine. |
| [TextViews](http://www.skorkin.com/2011/11/dxcore-services-text-views/) | Provides methods and properties for text views manipulation. |
| [ThirdPartyExtensions](http://www.skorkin.com/2010/11/dxcore-services-thirdpartyextensions/) | Provides access to the third-party extensions installed into Visual Studio 2010, e.g. Microsoft Productivity Power Tools. |
| [ToolWindows](http://www.skorkin.com/2011/11/dxcore-services-tool-windows/) | Provides methods for DXCore tool windows manipulation. |
| [UndoStack](http://www.skorkin.com/2011/09/dxcore-services-undostack/) | Provides access to the undo context and undo/redo stack services. |
| [UnitTests](http://www.skorkin.com/2011/07/dxcore-services-unittests/) | Provides methods for executing and debugging unit tests cases, and test collection. |
| [Version](http://www.skorkin.com/2011/11/dxcore-services-version/) | Provides DXCore version information services. |
| [VSSettings](http://www.skorkin.com/2011/12/dxcore-services-visual-studio-settings/) | Provides access to the Visual Studio settings (Visual Studio version, code editor font, tab size, syntax highlight colors, etc.). | |
| [Windows](http://www.skorkin.com/2011/12/dxcore-services-windows/) | Provides methods for windows manipulation. |
| [Wizards](http://www.skorkin.com/2011/07/dxcore-services-wizards/) | Provides access to the DXCore Wizards services. Note that it is hidden from Intellisense. |
| [XPO](http://skorkin.com/2010/11/dxcore-services-xpo/) | Provides services for the [Xpress Persistent Objects](http://www.devexpress.com/xpo)framework from [DevExpress](http://www.devexpress.com/). |

## DXCore Services – ActionHint

November 22nd, 2010

The **ActionHint** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for manipulating of [action hints](http://skorkin.com/2010/11/dxcore-components-action-hint/) and tweaking its settings for the current Visual Studio session.

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| DisableHint(HintBase) | Disables the specified hint. |
| DisableHintFeatureGroup(HintBase) | Disables the feature group of the specified hint. |
| GetFeatureGroup(HintBase) | Returns the FeatureGroup for the specified feature name. |
| GetHintStats(FeatureGroup, HintBase) | Returns the HintStats for the specified action hint text within the specified FeatureGroup. |
| HintIsEnabled(HintBase) | Returns true if it is OK to display the specified action hint. |
| PointTo(SourcePoint, ActionHintBase) | Displays the action hint pointing to the specified SourcePoint in the active text view. |
| PointTo(Point, ActionHintBase) | Displays the action hint pointing to the specified screen coordinates. |
| PointTo(Control, ActionHintBase) | Displays the action hint pointing at the center of the specified control (increase the action hint’s Distance property to move the arrow back a bit). |
| PointTo(SourcePoint, String, Color) | Displays the action hint pointing to the specified SourcePoint with specified text and color of the arrow. |
| PointTo(Point, String, Color) | Displays the action hint pointing to the specified screen coordinates with the specified text and color of the arrow. |
| PointTo(Int32, Int32, ActionHintBase) | Displays the action hint pointing to the specified line and column in the active text view. |
| PointTo(Control, String, Color) | Displays the action hint pointing at the center of the specified Control with the specified text and color of the arrow. |
| PointTo(Point, Int32, String, Color) | Displays the action hint pointing to the specified screen coordinates with the specified text and color of the arrow. |
| PointTo(Int32, Int32, String, Color) | Displays the action hint pointing to the specified line and column in the active view with the specified text and color of the arrow. |
| PointTo(Control, Int32, Sting, Color) | Displays the action hint pointing to the center of the specified Control with specified text, color and distance from the center of the arrow. |
| PointToCaret(ActionHintBase) | Displays the action hint pointing to the editor text caret in the code editor. |
| PointToCaret(String, Color) | Displays the action hint pointing to the editor text caret in the code editor with the specified text and color of the arrow. |
| PointToCaret(String, Int32, Color) | Displays the action hint pointing to the editor text caret in the code editor with specified text, color and distance from the center of the arrow. |
| PointToLineAndDisplayColumn(Int32, Int32, ActionHintBase) | Displays the action hint pointing to the specified line and display column in the active text view. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| DisplayTime | Determines how long (in milliseconds) the action hint will appear before closing. |
| Enabled | Gets or sets the enabled state of action hinting. |
| FadeIn | Determines whether the action hint will fade in (from transparent to the final Opacity setting) when it first appears. If false, action hints will simply appear. |
| FadeOut | Determines whether the action hint will fade out (from its Opacity setting to completely transparent) when it closes. If false, action hints will simply disappear. |
| Font | The font for the text of the action hint. |
| Opacity | Gets or sets the opacity of the action hint. |

## DXCore Services – Adornments

June 13th, 2012

The **Adornments** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the [graphic adornments architecture](http://www.skorkin.com/2010/10/dxcore-adornments-architecture/).

Methods:

|  |  |
| --- | --- |
| Name | Description |
| DecorateCodeIssue(TextDocument, CodeIssue) | Returns an adornment set that contains the list of adornments grouped by the adornments holder for the specified CodeRush code issue. |
| DecorateEditor(TextDocument) | Returns an adornment set that contains the list of adornments grouped by the adornments holder for the code editor. |
| DecorateLanguageElement (TextDocument, LanguageElement) | Returns an adornment set that contains the list of adornments grouped by the adornments holder for the language element. |
| RefreshAll | Refreshes all adornments in all opened text documents. |
| RegisterAdornmentDecoratorCreator (CreateAdornmentDecoratorHandler) | Registers the adornment decorator creator handler passed as a parameter to this method. |
| RegisterAdornmentHolderCreator (CreateDocument AdornmentHolderHandler) | Registers the adornment holder creator handler passed as a parameter to this method. |
| Update(IAdornmentsHolder) | Updates the specified adornments holder. |

Properties:

|  |  |
| --- | --- |
| Name | Description |
| BindingManager | Returns the binding manager instance. |
| Engine | Returns the adornments engine instance. |
| Factory | Return the adornments factory instance. |

## DXCore Services – Assembly

June 13th, 2012

The **Assembly** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the [plug-in assemblies](http://www.skorkin.com/2010/08/dxcore-plug-ins-overview/) of the [DXCore Framework](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/).

Methods of this service:

|  |  |
| --- | --- |
| Name | Description |
| Assemblies(DXCoreAssemblyType) | Returns an array of DXCore assemblies of the specified assembly type. |
| CoreAssemblies | Returns an array of the registered DXCore assemblies. |
| GetCustomAttribute(Assembly, Type) | Gets a custom attribute of the specified attribute type from the passed-in assembly. |
| PlugInAssemblies | Returns an array of the registered CodeRush non-system plug-ins. |
| RegisterAssembly(Assembly) | Registers a DXCore assembly. This method should not be called by plug-in developers. It is used by the DXCore loader. |
| SystemPlugInAssemblies | Returns an array of the registered DXCore system plug-ins. |

Properties:

|  |  |
| --- | --- |
| Name | Description |
| EvaluationAssemblies | Gets a collection of DXCore assemblies that will expire after the trial period ends. Note that it is hidden from Intellisense. |
| ExpiredAssemblies | Gets a collection of DXCore assemblies that have expired. Note that it is hidden from Intellisense. |
| FutureAssemblies | Gets a collection of DXCore assemblies whose run window has not yet opened (the system date/time is \*before\* this assembly’s start time). Note that it is hidden from Intellisense. |
| Item(String) | Gets a registered DXCore assembly by name. |

## DXCore Services – BigHint

December 14th, 2010

The **BigHint** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for manipulating of [big hints](http://www.skorkin.com/2010/12/dxcore-components-bighint/), such as creating new ones and showing at the specified position.

The following **methods** are available:

|  |  |
| --- | --- |
| **Name** | **Description** |
| DisableHint(HintBase) | Disables the specified hint. |
| DisableHintFeatureGroup(HintBase) | Disables the feature group of the specified hint. |
| GetFeatureGroup(HintBase) | Returns the FeatureGroup for the specified feature name. |
| GetHintStats(FeatureGroup, HintBase) | Returns the HintStats for the specified big hint text within the specified FeatureGroup. |
| HintIsEnabled(HintBase) | Returns true if it is OK to display the specified **big hint**. |
| RequestHint(BigHintBase) | Creates new hint if it is allowed. Returns null if the hint is not allowed to show. |
| ShowAt(SourcePoint, BigHintBase) | Displays the **big hint** at the specified SourcePoint. |
| ShowAt(Point, BigHintBase) | Displays the specified **big hint** at the given screen coordinates. |
| ShowAt(Control, BigHintBase) | Displays the specified **big hint** at the given control. |
| ShowAt(SourcePoint, String, String) | Displays the **big hint** at the specified SourcePoint with the specified title and text. |
| ShowAt(Point, String, String) | Displays the **big hint** at the specified screen coordinates with the specified title and text. |
| ShowAt(Int32, Int32, BigHintBase) | Displays the **big hint** at the specified line and column in the active view. |
| ShowAt(Control, String, String) | Displays the **big hint** at the specified control with the specified title and text. |
| ShowAt(SourcePoint, String, String, String) | Displays the **big hint** at the specified SourcePoint with the specified title, text and [User Guide](http://skorkin.com/2010/08/ide-tools-user-guide/) page. |
| ShowAt(Int32, Int32, String, String, String) | Displays the **big hint** at the specified line and column in the active view with the specified title, text and **User Guide** page. |
| ShowAtCaret(BigHintBase) | Displays the **big hint** at the editor caret with the specified title and text. |
| ShowAtCaret (String, String) | Displays the **big hint** at the editor caret with the specified title and text. |
| ShowAtCaret (String, String, String) | Displays the **big hint** at the editor caret with the specified title, text and **User Guide** page. |
| ShowAtLowerRight(Rectangle, Int32, BigHintBase) | Displays the **big hint** at the lower right angle of the specified rectangle. Takes the margin in pixels, to use as a parameter. |
| ShowUnderCaret(BigHintBase) | Displays the **big hint** at the editor caret. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Enabled | Gets or sets the enabled state of **big hinting**. |
| FadeIn | Determines whether the **big hint** will fade in when it first appears. If false, big hints will simply appear. |
| FadeOut | Determines whether the **big hint** will fade out when it closes. If false, big hints will simply disappear. |

## DXCore Services – Breakpoint

September 27th, 2010

The **Breakpoint** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for retrieving and toggling breakpoints in the IDE code editor.

Here are these methods:

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| Get(String, Int32) | Returns a breakpoint object for the specified file and line number if one exists. |
| Toggle(Int32) | Toggles the breakpoint at the specified line number in the active file. |

## DXCore Services – Caret

August 31st, 2011

The **Caret** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for the code editor text caret (cursor). It allows you to move, hide, show the caret and execute many text operations, such as text insertion, deletion, and retrieving for the current caret position.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| DeleteLeft(Int32) | Deletes the specified number of characters to the left of the editor caret. |
| DeleteLeftWhiteSpace | Deletes whitespace characters to the left of the editor caret and returns the number of characters deleted. |
| DeleteRight(Int32) | Deletes the specified number of characters to the right of the editor caret. |
| DeleteRightWhiteSpace | Deletes whitespace characters to the right of the editor caret and returns the number of characters deleted. |
| GetSurroundingCharacters( Char, Char) | Gets the character to the left and right of the editor caret (on the current line). |
| GetSurroundingText(String, String) | Gets the text to the left and right of the editor caret (on the current line) in the active text document. |
| GetSurroundingText( SourcePoint, SourcePoint, String, String) | Gets the text to the left and right of the specific source points in the active text document. |
| Hide | Hides the editor caret. You can restore the editor caret after calling Hide() through a call to CodeRush.Caret.Show(). Every call to Hide() must have a matching call to Show(). |
| InCodeSnippetField | Determines whether or not the editor caret is currently in a code snippet field. This will only return true in Visual Studio 2005 or newer. |
| InCurrentStatementMarker | Determines whether or not the editor caret is currently in a current statement marker. This will only return true in Visual Studio 2005 or newer. |
| Insert(Char) | Inserts a character at the editor caret position and moves the caret. |
| Insert(Char, Boolean) | Inserts a character at the editor caret position and optionally moves the caret. |
| Insert(String) | Inserts the given text at the editor caret position and optionally moves the caret. |
| Insert(String, Boolean) | Inserts the given text at the caret position and optionally moves the caret. |
| MoveLeft(Int32) | Moves the editor caret to the left a specified number of characters. If the caret is currently inside virtual space, it will moved to the end of the current line first. |
| MoveOutOfVirtualSpace | If the editor caret is in virtual space, it will be moved to the position of the end-of-line terminator on the line. |
| MoveRight(Int32) | Moves the editor caret to the right a specified number of characters. If the caret is currently inside virtual space, it will moved to the end of the current line first. |
| MoveTo(Int32) | Moves the editor caret to the location represented by the specified stream position. |
| MoveTo(Int32, Boolean) | Moves the editor caret to the location represented by the specified stream position, and optionally, extends the selection. |
| MoveTo(Int32, Int32) | Moves the active point to the specified line and column. |
| MoveTo(Int32, Int32, Boolean) | Moves the active point to the specified line and column. |
| MoveTo(SourcePoint) | Moves the editor caret to the given position. |
| MoveTo(SourcePoint, Boolean) | Moves the editor caret to the given position. |
| MoveToEndOfLine | Positions the editor caret at the end of the current line. |
| MoveToStartOfLine | Positions the editor caret at the start of the current line. |
| Show | Shows the editor caret. Call this after calling Hide(). |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| AtCompilerDirective | Returns true when the editor caret is at the line with a compiler directive (e.g. #warning, #error). |
| AtEndOfWord | Returns true if the editor caret is at the end of a word. |
| AtStartOfWord | Returns true if the editor caret is at the start of a word. |
| InsideCode | Returns true if the editor caret is inside a code block. |
| InsideComment | Returns true if the editor caret is inside of a comment. |
| InsideParens | Returns true if the editor caret is inside parentheses. |
| InsideString | Returns true if the editor caret is inside of a string. |
| InsideVirtualSpace | Returns true if the editor caret is inside virtual space. |
| InsideXmlDocComment | Returns true if the editor caret is inside an XML documentation comment. |
| LeftChar | Gets the character to the left of the editor caret. |
| LeftText | Gets the text to the left of the editor caret. |
| Line | Returns the line number of the active editor caret, or -1 if a text view is not active. |
| Location | Returns a CaretLocation instance that is the current location of the editor caret. |
| Offset | Returns the line character offset of the active editor caret, or -1 if a text view is not active. If the caret is in virtual space, this will be the position of the end-of-line terminator on the line. |
| OnDeclaration | Returns true if the editor caret is on a declaration. |
| OnDeclarationName | Returns true if the editor caret is on a declaration name. |
| OnDeclarationStartToken | Returns true if the editor caret is on a start token of the declaration. |
| OnDeclarationType | Returns true if the editor caret is on a declaration type. |
| OnDeclarationVisibility | Returns true if the editor caret is on a declaration visibility keyword. |
| OnEmptyLine | Returns true if the editor caret is on an empty line. |
| OnMethod | Returns true if the editor caret is on a method. |
| OnMethodName | Returns true if the editor caret is on a method name. |
| OnMethodType | Returns true if the editor caret is on a method type. |
| OnMethodVisibility | Returns true if the editor caret is on a visibility keyword. |
| OnProperty | Returns true if the editor caret is on a property. |
| OnPropertyName | Returns true if the editor caret is on a property name. |
| OnPropertyType | Returns true if the editor caret is on a property type. |
| OnPropertyVisibility | Returns true if the editor caret is on a property visibility keyword. |
| RightChar | Gets the character to the right of the editor caret. |
| RightText | Gets the text to the right of the editor caret. |
| ScreenPosition | Returns the screen position of the editor caret. |
| SourcePoint | Gets a source point of the current editor caret position, or an empty source point if a text view is not active. A source point can be located inside a virtual space. |
| ViewColumn | Returns the current view column of the active editor caret, or -1 if a text view is not active. A value can be inside a virtual space. |

## DXCore Services – Clipboard

May 21st, 2011

The **Clipboard** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the Windows clipboard. Contains methods for checking, retrieving, copying, cutting and pasting clipboard contents into the code editor.

Here are the **methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Copy | Copies the current selection to the clipboard. |
| Cut | Cuts the current selection to the clipboard. |
| Cycle | Cycles the clipboard ring and places the next available text on the clipboard. |
| FindData(string[]) | Returns an object supporting the clipboard format specified by formats. |
| FindData(Type[]) | Returns an object supporting the clipboard format specified by formats |
| GetData(String) | Returns an object supporting the clipboard format specified by format |
| GetData(Type) | Returns an object supporting the clipboard format specified by format |
| GetText | Returns the text on the clipboard. GetText tries to obtain the “UnicodeText” format first, and checks for the “Text” format if “UnicodeText” is not available. |
| HasDataFormat(String) | Returns true if the clipboard holds data that can be represented by the format specified. |
| HasDataFormat(Type) | Returns true if the clipboard holds data that can be represented by the format specified. |
| HasType(String) | Returns true if the clipboard has an identifier of the same type as that specified by type name. |
| Paste | Pastes the current contents of the clipboard. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| AsComponent | Returns the data on the clipboard as a deserialized Component. |
| AsHTML | Returns the data on the clipboard as HTML. |
| AsImage | Returns the data on the clipboard as an Image. |
| AsObject | Returns the data on the clipboard as a deserialized object. |
| AsRTF | Returns the data on the clipboard as RTF. |
| AsString | Returns the data on the clipboard as string. |
| AsText | Returns the data on the clipboard as text. |
| AsUnicodeText | Returns the data on the clipboard as Unicode text. |
| DataIsFresh | Returns true if the data in the clipboard originates from the current cursor position in the editor. |
| HasComponent | Returns true if the clipboard holds a Component. |
| HasHTML | Returns true if the clipboard holds data that can be represented as HTML. |
| HasImage | Returns true if the clipboard holds data that can be represented as an Image. |
| HasObject | Returns true if the clipboard holds a .NET object. |
| HasRTF | Returns true if the clipboard holds data that can be represented in RTF (Rich Text Format). |
| HasText | Returns true if the clipboard holds data that can be represented as text. |
| HasUnicodeText | Returns true if the clipboard holds data that can be represented in Unicode text. |
| LastEntry | Returns a ClipboardEntry instance of the last entry cut or copied to the clipboard. |
| LastOperation | Returns the last clipboard operation performed in the IDE. The operation shall be – ClipboardOperation.Cut, ClipboardOperation.Copy, or ClipboardOperation.Unknown. |
| LastType | Gets the TypeDeclaration instance for the identifier on the clipboard, if determinable at the time of the cut or copy operation. |
| MonitoringEnabled | Gets or sets the value which specifies whether**DXCore**should monitor clipboard  operations. Note that it is hidden from Intellisense. |

## DXCore Services – Code Modification

February 14th, 2012

The **Code Mod** (Modification) [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides APIs for code generation and code declaration, and access to the code provider extensions.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| CanShowSmartTagAsync( AsyncAvailabilityEventArgs) | Return true if there’s at least one code provider available. |
| GenerateCode( LanguageElement) | Generates the code of the given language element. |
| GenerateCode( LanguageElement, Boolean) | Generates the code of the given language element. Optionally removes the trailing line terminator. |
| GetArgumentsCollection( LanguageElement) | Returns arguments of the given language element. |
| GetAvailableCodeProviders | Returns all available code providers. |
| GetDeclaration(LanguageElement) | Returns the declaration of the given element. |
| GetElementType(LanguageElement) | Returns the type of the undeclared element. |
| GetElementTypes(LanguageElement) | Returns all possible types of the undeclared element. |
| HasDeclaration(LanguageElement) | Returns true if the given element is declared. |
| IsInsideMacro(SourceFile, SourcePoint) | Returns true if the there is a macro definition (C++) at the given source point in the specified source file. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| AvailableCodeProviders | Gets all available code providers. |
| CanShowSmartTag | Return true if there’s at least one code provider available. |
| CodeProviders | Gets all registered code providers. |
| Providers | Gets all registered code providers. |
| ProvidersCount | Gets the count of registered code |

## DXCore Services – CodeStyle

November 22nd, 2011

The **Code Style** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods and properties that specify user code style formatting settings, such as: prefixes and suffixes of locals, fields, properties and other declarations, a default visibility scope for types, members and others.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| HasFieldStyle(String) | Return true if the passed name has a field variable style based on user settings for the active language (on the Editor\Code Style\Identifiers options page of the [Options Dialog](http://www.skorkin.com/2010/08/ide-tools-options-dialog/)). |
| HasLocalStyle(String) | Return true if the passed name has a local variable style based on user settings for the active language (on the Editor\Code Style\Identifiers options page). |
| HasParamStyle(String) | Return true if the passed name has a parameter style based on user settings for the active language (on the Editor\Code Style\Identifiers options page). |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| FormatAutoDeclaration | Returns true if the [Auto Declare](http://www.skorkin.com/2011/02/coding-helpers-auto-declare/) code should be formatted according to the current IDE formatting style options. |
| FormatCodeLibraryClip | Returns true if the inserted code library clips should be formatted according to the current IDE formatting style options. |
| FormatDuplicateLine | Returns true if the [Duplicate Line](http://www.skorkin.com/2011/04/intelligent-code-lines-enhancement-and-modification-using-the-duplicate-line-feature/) actions should be formatted according to the current IDE formatting style options. |
| FormatMultiLineEmbedding | Returns true if the [multi-line Embeddings](http://www.skorkin.com/2011/09/wrapping-code-blocks-with-the-coderush-code-embeddings-feature/) should be formatted according to the current IDE formatting style options. |
| FormatRefactoring | Returns true if the [code refactorings](http://www.skorkin.com/2010/09/what-is-refactoring-benefits-of-a-code-refactoring/) should be formatted according to the current IDE formatting style options. |
| FormatSelectionInversion | Returns true if the [Selection Inversions](http://www.skorkin.com/2011/10/inverting-a-code-block-into-its-opposite-using-coderush-selection-inversion/) should be formatted according to the current IDE formatting style options. |
| FormatSingleLineEmbedding | Returns true if the single-line Embeddings should be formatted according to the current IDE formatting style options. |
| FormatSmartPaste | Returns true if the [Intelligent Pastes](http://www.skorkin.com/2011/03/clipboard-intelligent-paste/) should be formatted according to the current IDE formatting style options. |
| FormatTemplate | Returns true if the code template expansions should be formatted according to the current IDE formatting style options. |
| Formatting | Returns the IFormattingService implementer for accessing code formatting rules. |
| PrefixField | Returns the prefix associated with field variables for the active language based on user settings (on the Editor\Code Style\Identifiers options page). |
| PrefixLocal | Returns the prefix associated with local variables for the active language based on user settings (on the Editor\Code Style\Identifiers options page). |
| PrefixParam | Returns the prefix associated with parameters for the active language based on user settings (on the Editor\Code Style\Identifiers options page). |
| ScopeConstant | Returns the MemberVisibility value for constants specified on the Editor\Code Style\Scope options page. |
| ScopeEvent | Returns the MemberVisibility value for events specified on the Editor\Code Style\Scope options page. |
| ScopeField | Returns the MemberVisibility value for fields specified on the Editor\Code Style\Scope options page. |
| ScopeMethod | Returns the MemberVisibility value for methods specified on the Editor\Code Style\Scope options page. |
| ScopeNestedType | Returns the MemberVisibility value for nested types specified on the Editor\Code Style\Scope options page. |
| ScopeProperty | Returns the MemberVisibility value for properties specified on the Editor\Code Style\Scope options page. |
| ScopeType | Returns the MemberVisibility value for types specified on the Editor\Code Style\Scope options page. |
| SuffixField | Returns the suffix associated with field variables for the active language based on user settings (on the Editor\Code Style\Identifiers options page). |
| SuffixLocal | Returns the suffix associated with local variables for the active language based on user settings (on the Editor\Code Style\Identifiers options page). |
| SuffixParam | Returns the suffix associated with parameters for the active language based on user settings (on the Editor\Code Style\Identifiers options page). |
| SuppressDefaultAccessModifiers | Returns true if access modifiers for default visibility are omitted in source code, by default. |

## DXCore Services – Color

September 27th, 2010

The **Color** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for operations with a color, such as blending, brightening, inverting, etc.

Methods of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| AdjustBrightness(Color, Int32) | Adjusts the brightness of a specified color by the specified amount. |
| Blend(Color, Color) | Blends two colors to produce a third color. |
| Blend(Color, Color, Byte) | Blends two colors to produce a third color with a specified percent of the first color. |
| Invert(Color) | Inverts a color. |
| IsMatch(Color, Color) | Determines if two colors are equivalent. |
| IsMatch(Color, Color, Int32) | Determines if two colors are equivalent with the specified tolerance used in the comparison. |
| ReloadColors | Loads custom color settings from storage. |

Properties:

|  |  |
| --- | --- |
| **Name** | **Description** |
| SyntaxHighlighting | Colors for syntax highlighting of CodeRush elements such as StringProvider, TextCommand, and Alias. |
| VSDark | Returns a color one interval darker than the button shadow standard system color. |
| VSLight | Returns a color one interval lighter than the button face standard system color. |
| VSLightCaption | Returns a color one interval lighter than the active caption standard system color. |
| VSMedium | Returns a color one interval darker than the button face standard system color. |

## DXCore Services – Command

October 31st, 2011

The **Command** [DXCore Service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the [DXCore actions](http://www.skorkin.com/2010/09/dxcore-components-action/) and Visual Studio commands.

Here are methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| AddNamed(String, String, String) | Adds a named command and registers it exclusively with Visual Studio (and not with CodeRush). If you want to register a command with Visual Studio, it is recommended to drop an [Action DXCore control](http://www.skorkin.com/2010/09/dxcore-components-action/) on your plug-in form and set its properties as required. |
| Delete(String) | Removes the given named Visual Studio command. |
| DeleteAllStartingWith (String) | Deletes all Visual Studio commands starting with the specified prefix in the CodeRush category (e.g., “CodeRush.” + prefix). |
| DeleteAllStartingWith (String, String) | Deletes all Visual Studio commands starting with the specified prefix and category. |
| DeleteAllStartingWith (String, String, string[]) | Deletes all Visual Studio commands starting with the specified prefix and category, except for the commands listed in the given “exceptions” string array. |
| DeleteAllStartingWith (String, string[]) | Deletes all Visual Studio commands starting with the specified prefix in the CodeRush category (e.g., “CodeRush.” + aCommandPrefix). |
| Execute(Command) | Executes the specified Visual Studio command. If the command is null or cannot be executed, this method returns false. |
| Execute(Command, String) | Executes the specified Visual Studio command. If the command is null or cannot be executed, this method returns false. |
| Execute(Guid, Int32) | Executes the Visual Studio command represented by the specified command group and id. If the command does not exist or cannot be executed, this method returns false. |
| Execute(Guid, Int32, String) | Executes the Visual Studio command represented by the specified command group and id. If the command does not exist or cannot be executed, this method returns false. |
| Execute(String) | Executes an action or command registered with the DXCore and/or Visual Studio. If the action or command does not exist or cannot be executed, this method returns false. |
| Execute(String, Int32) | Executes the Visual Studio command represented by the specified command group and id. If the command does not exist or cannot be executed, this method returns false. |
| Execute(String, Int32, String) | Executes the Visual Studio command represented by the specified command group and id. If the command does not exist or cannot be executed, this method returns false. |
| Execute(String, String) | Executes an action or command registered with the DXCore and/or Visual Studio. If the action or command does not exist or cannot be executed, this method returns false. |
| Exists(String) | Returns true if the specified command exists and has been registered with Visual Studio. |
| FindPopup(MenuBar, String, Boolean) | Searches a MenuBar for a popup menu control whose Bar.Name matches name. Note that it is hidden from Intellisense. |
| Get(String) | Returns the Visual Studio registered command given by name. “CodeRush.” is used as the command category if a category is not specified. |
| Get(String, String) | Returns the Visual Studio command specified by the category and name. |
| GetFirstStartingWith (String, String) | Returns the first Visual Studio command starting with the specified prefix in the specified category. |
| Unregister(Action) | Unregisters an Action so that it is no longer available in DXCore and the IDE. Note that it is hidden from Intellisense. |

## DXCore Services – Constants

October 6th, 2010

The **Constants** [service](http://skorkin.com/2010/09/dxcore-services-list/) contains [DXCore](http://skorkin.com/2010/06/devexpress-dxcore-framework/) global constants for text commands. Unfortunately, it is not very useful at the moment.

The following fields are available:

|  |  |
| --- | --- |
| **Name** | **Description** |
| TextCommandBegin | The character constant for the text command begin. |
| TextCommandEnd | The character constant for the text command end. |

## DXCore Services – Content

November 22nd, 2011

The **Content** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for accessing content providers. Content providers are a common object for [refactoring](http://www.skorkin.com/2010/09/what-is-refactoring-benefits-of-a-code-refactoring/) operations and [code providers](http://www.skorkin.com/2011/01/ide-tools-terminologykey-concepts/) which modify source code.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| BeginUpdate | Suspends an availability check for content providers (refactoring or code operations). If you call this inside a plug-in, you must guarantee a call to EndUpdate (e.g., placed in a finally block). |
| EndUpdate | Marks the end of an update session. You must call this once for every call to BeginUpdate. |
| Execute(String) | Executes a content provider with the specified provider name. |
| Get(String) | Returns a content provider with the specified provider name. |
| GetAvailableContentProvidersAfterParse | Parses the active text document if its text has been changed. Then, returns all available content providers in the current context. |
| GetFixesForIssue(String) | Returns the list of all content providers that can fix the specified CodeRush code issue. |
| IsAvailable(ContentProvider, IElement) | Returns true if the given code provider is available in the specified context. |
| IsAvailable(String, IElement) | Returns true if the given code provider is available in the specified context. |
| IsAvailable(String, IElement, Boolean) | Returns true if the given code provider is available in the specified context. |
| IsAvailable(String, ScopeResolveResult, IElement, Boolean) | Returns true if the given code provider is available in the specified context. |
| RegisterExtension(String) | Registers a new content provider extension. |
| ResetIsUpdating | Resets the internal update counter, so the availability checks can be resumed. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| AvailableProviders | Gets all available content providers in the current context. |
| CanShowSmartTag | Returns true if there’s at least one content provider available and it’s possible to show the Smart Tag. |
| IsUpdating | Returns true if an availability check is being updated. |
| LastExecutedProvider | Returns the last applied content provider. |
| Providers | Gets all registered content providers. |
| VSFeatures | Returns the list of Visual Studio built-in refactorings wrappers. |

## DXCore Services – Context

September 27th, 2010

The **Context** [service](http://skorkin.com/2010/09/dxcore-services-list/) is responsible for the **DXCore** [contexts system](http://skorkin.com/2010/09/dxcore-contexts-overview/). It provides access to [context providers](http://skorkin.com/2010/09/dxcore-components-contextprovider/) and methods that report whether specified context is satisfied.

Here are its methods:

|  |  |
| --- | --- |
| **Name** | **Description** |
| BeginCheck | Marks the beginning of a sequence of context checks. A cache is maintained until the EndCheck method is called, so duplicate checks on a context will return the cached value. If you call this method, be sure to call EndCheck. |
| ClearCache | Clears the context cache (created inside calls toBeginCheck and EndCheck). |
| EndCheck | Marks the end of a sequence of context checks. |
| GetAllContextPaths(string) | Returns an array of all available context names for the specified language. |
| GetAllContexts(string) | Returns an array of all available **context providers** for the specified language. |
| GetContextProvider(String) | Returns the specified context provider. |
| PopulateContextPicker | Fills out a [context picker](http://skorkin.com/2010/09/dxcore-components-contextpicker/) with all the registered contexts. |
| Satisfied(String) | Checks a context to see if it is satisfied. |
| Satisfied(StringCollection, Boolean) | Checks a collection of contexts to see if they are satisfied or not. |
| Satisfied(String, Boolean) | Checks a context to see if it is satisfied. Will return the default value if the context is not found. |
| Satisfied(String, Match) | Checks a context to see if it is satisfied. |
| Satisfied(String[], Boolean) | Checks an array of contexts to see if they are satisfied or not. |
| Satisfied(StringCollection, Match, Boolean) | Checks a collection of contexts to see if they are satisfied or not. |
| Satisfied(String, Boolean, Match) | Checks a context to see if it is satisfied. Will return the default value if the context is not found. |
| Satisfied(String[], Match, Boolean) | Checks an array of contexts to see if they are satisfied or not. Checks for parameter replacement with named groups from the passed regular expression match. |

Properties:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Providers | Returns an array of all available **context providers**. |

## DXCore Services – Contract

June 18th, 2011

The **Contract** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the registered contract providers used in the [Add Contract](http://www.skorkin.com/2011/02/code-providers-add-contract/) code provider. It has only three properties:

|  |  |
| --- | --- |
| **Property name** | **Description** |
| AvailableContractProviders | Gets an IEnumerable<ContractProvider> instance of all available contract providers in the current context. |
| ContractProviders | Gets an IEnumerable<ContractProvider> instance of all registered contract providers. |
| Providers | Gets an IEnumerable<ContentProvider> instance of all registered contract providers. |

## DXCore Services – Debugger

October 6th, 2010

The **Debugger** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides access to the Visual Studio debugger and its current mode properties.

Methods:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Break | Causes the current process to pause its execution. This method waits for the debugger to enter break mode before returning. |
| Break(Boolean) | Causes the current process to pause its execution. Takes a boolean parameter which specifies whether this method should wait for the debugger to enter break mode before returning or return immediately. |
| DetachAll | Detaches the debugger from all attached programs. |

Properties:

|  |  |
| --- | --- |
| **Name** | **Description** |
| ActiveMode | Active debugger mode (break, design or run). |
| ActiveProcess | Active process object of EnvDTE.Process type. |
| ActiveProgram | Active program object of EnvDTE.Program type. |
| ActiveStackFrame | Active stack frame of EnvDTE.StackFrame type. |
| ActiveThread | Active thread of EnvDTE.Thread type. |
| IsBreakMode | Returns true if the debugger is in break mode. |
| InDesignMode | Returns true if the debugger is in design mode. |
| InRunMode | Returns true if the debugger is in run mode. |

## DXCore Services – Designer

October 11th, 2010

The **Designer** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for Windows Forms design-time support. It is similar to the **Editor** service.

Here they are:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Activate(Document) | Activates (gives focus to) or opens the form designer for the specified document. |
| Activate(Document, Boolean) | Activates (gives focus to) or opens the form designer for the specified document. Takes a boolean parameter that specifies whether to force opening a designer for the document if it is not found. |
| ShowCodeView | Shows the code view corresponding to the current designer. |
| ToggleCodeAndDesignerViews | Switches the view between code and designer views. |

Properties:

|  |  |
| --- | --- |
| **Name** | **Description** |
| IsActive | True if a form designer is active. |

## DXCore Services – Dialogs

October 6th, 2010

The **Dialogs** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for a [DXCore](http://skorkin.com/2010/06/devexpress-dxcore-framework/) internal dialogs manipulation. Currently it is used internally for selecting a text command or a string provider from the specified list, and is not very useful for plug-in developers at the moment.

There are two methods available:

|  |  |
| --- | --- |
| **Name** | **Description** |
| SelectFromList(ICollection, String, String) | Creates a new dialog with the specified title, prompt, and items list. Returns the object selected. |
| SelectFromList(ICollection, String, String, Boolean) | Creates a new dialog with the specified title, prompt, and items list. Allows returning an item missing from the list if specified by the boolean parameter. Returns the object selected. |

## DXCore Services – Documents

November 22nd, 2011

The **Documents** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides an API to control opening and closing documents inside the Visual Studio IDE. You can insert, remove and change any text at the specified coordinates of the given documents.

|  |  |
| --- | --- |
| Method name | Description |
| Activate(Document) | Activates (gives focus to) the specified document. Preference is given to the source code view. If a source code view is unavailable but an alternate view is available (e.g., a form designer for this document), it will be activated. See also: CodeRush.Designer.Activate and CodeRush.Editor.Activate. |
| Activate(Document, Boolean) | Activates (gives focus to) the specified document. See also: CodeRush.Designer.Activate and CodeRush.Editor.Activate. |
| Activate(Document, DocumentView) | Activates (gives focus to) or opens the specified view of the specified document. |
| Activate(Document, DocumentView, Boolean) | Activates (gives focus to) or opens the specified view of the specified document. |
| AttachObject(Document, String, Object) | Attaches a named object to the specified document. |
| AttachObject(String, Object) | Attaches a named object to the active document. |
| BelongsToValidProject(Document) | Checks if the specified document belongs to a valid project. |
| DeleteCharsLeft(EditPoint, Int32) | Deletes the specified number of characters to the left of the specified edit point in the active document. |
| DeleteCharsRight(EditPoint, Int32) | Deletes the specified number of characters to the right of the specified edit point in the active document. |
| Find(String) | Returns the document by the specified path. |
| Find(EndDTE.TextDocument) | Returns the document by the EnvDTE.TextDocument instance. |
| Format | Formats the active document. |
| Format(Int32, Int32, Int32, Int32) | Formats the specified range of the active document. |
| Format(SourceRange) | Formats the specified range of the active document. |
| GdiPaintOnTextViews( EditorPaintEventHandler) | Paints on all text views corresponding to the active document. |
| Get(String) | Returns the document specified by the given file name. |
| GetAllOpenedDocuments | Returns the designer document for the active file. |
| GetDesigner | Returns the designer document for the active file. |
| GetDesigner(String) | Returns the designer document specified by the given file name. |
| GetDocuments(GetDocumentFlags) | Returns a collection of all documents based on the specified criteria. For example, you can use this method to get a collection of all text documents in the current project. |
| GetEnumerator | Returns an enumerator for iterating documents. |
| GetIndentSize(String) | Returns the indent size for the specified document. |
| GetLeadingWhiteSpace(Int32) | Returns the leading white space (tabs and spaces) for the specified line. |
| GetLineAt(Int32) | Returns the code line at the specified line number for the active document. |
| GetLineAt(TextDocument, Int32) | Returns the code line at the specified line number for the specified document. |
| GetObject(Document, String) | Gets the object attached to the given document. |
| GetObject(String) | Gets the object attached to the active document. |
| GetProjectName(Document) | Returns the name of the project for the specified document. |
| GetText(String) | Retrieves the current text of the specified file. If the file is opened in Visual Studio in a document window or in-memory, the current contents of the buffer will be returned. If the file is not open, the contents of the file on disk will be returned. |
| GetText(String, Boolean) | Retrieves the current text of the specified file. If the file is open in Visual Studio in a document window or in-memory, the current contents of the buffer will be returned. If the file is not open, the contents of the file on disk will be returned. |
| GetTextDocument | Returns the non-designer source document for the active file. |
| GetTextDocument(String) | Returns the non-designer source document specified by the given file name. |
| InsertText(String) | Inserts the specified text at the editor caret in the active document. |
| LineStartsWith(Int32, String) | Returns true if the given line number in the current file starts with the specified prefix. Leading white space is ignored. |
| Parse(TextDocument) | Parses the specified text document and returns the language element representing it. |
| ParseActive | Parses the active text document and returns the language element representing it. |
| RefreshTextViews | Refreshes text views of the active document. |
| RemoveObject(Document, String) | Removes an attached object from the specified document. |
| RemoveObject(String) | Removes an attached object from the active document. |
| ShowCode | Opens the specified file and scrolls it to the specified source range. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| Active | Returns the active document. |
| ActiveFileExt | Returns the file extension (e.g., “.cs”, “.cpp”, “.h”, “.vb”, etc.) of the active document. |
| ActiveFileName | Returns the file name of the active or last active document. |
| ActiveLanguage | Returns the programming language used in the active document. |
| ActiveLine | Returns the line of source code at the cursor in the active document. |
| ActiveTextDocument | Returns the active text document or null if the active document is not a text document (e.g., a designer). |
| ActiveTextView | Gets the active text view for the active text document. Returns null if not available. |
| AllDocuments | Returns a collection of all documents. |
| AllProjectDocuments | Returns a collection of all documents in the current project. |
| AllProjectTextDocuments | Returns a collection of all text documents in the current project. |
| AllSolutionDocuments | Returns a collection of all documents in the current solution. |
| AllSolutionTextDocuments | Returns a collection of all text documents in the current solution. |
| AllTextDocuments | Returns a collection of all text documents. |
| CollapsibleRegionsProvider | Returns a collapsible regions implementer that allows you to iterate via collapsed or expanded regions of outlined text. |
| Count | Gets the number of documents available. |
| Formatting | Returns true if code in the active document is being formatted, through a call to CodeRush.Document.Format or TextDocument.Format. |
| Item | Return the document by the specified document path. |
| LastActive | Returns the last active document. |

## DXCore Services – DynamicLists

October 6th, 2010

The **DynamicLists** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for working with dynamic lists. Note, that it is only used internally and hidden from Intellisense but can be used for your purposes if needed.

Methods:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Get(String) | Returns a collection of DynamicLists that match the specified list name. |
| Get(String, String) | Returns a collection of DynamicLists that match the specified dynamicListName and languageID parameters. Throws an exception if the specified DynamicListProvider is not found. |
| Get(String, String, String) | Returns a collection of DynamicLists that match the specified dynamicListProviderName, dynamicListName, and languageID parameters. Throws an exception if the specified DynamicListProvider is not found. |
| GetLanguageDynamicLists | Returns a collection of DynamicLists for the specified language ID. |
| ProvidersByLanguage(String) | Returns an array of DynamicLists that match the specified languageID parameter. |

Properties:

|  |  |
| --- | --- |
| **Name** | **Description** |
| ProvidersList | Returns an array of all DynamicListProviders. |

## DXCore Services – Editor

October 27th, 2010

The **Editor** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for the IDE code editor. It is is similar to the [Designer service](http://skorkin.com/2010/10/dxcore-services-designer/).

Here are its methods:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Activate(Document) | Activates (gives focus to) or opens the form designer for the specified document. |
| Activate(Document, Boolean) | Activates (gives focus to) or opens the form designer for the specified document. Takes a boolean parameter that specifies whether to force opening a designer for the document if it is not found. |
| ShowDesigner | Shows the code view corresponding to the current designer. |
| ToggleCodeAndDesignerViews | Switches the view between code and designer views. |

Properties:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Active | Returns the active TextWindow object. |
| AllowEditsWhileDebugging | Indicates whether or not language service allows writing to the active text document during debugging. |
| HasFocus | True if an edit window has focus. |
| IncrementalSearchActive | Returns true if incremental searching is activated. |

## DXCore Services – EditPoints

November 22nd, 2011

The **EditPoints** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for creating edit points. Edit points allow you to manipulate text as data in text buffers. The service is hidden from Intellisense, and it contains only different overloads of the New method to create edit points, such as:

|  |  |
| --- | --- |
| Method name | Description |
| New(Text document, Int32) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, Int32, Boolean) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, Int32, Boolean, String) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, Int32, Int32) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, Int32, Int32, Boolean) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, Int32, Int32, Boolean, String) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, Int32, Int32, String) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, Int32, String) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, SourcePoint) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, SourcePoint, Boolean) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, SourcePoint, Boolean, String) | Initializes a new EditPoint in the given Text document using the specified coordinates. |
| New(Text document, SourcePoint, String) | Initializes a new EditPoint in the given Text document using the specified coordinates. |

## DXCore Services – Feature

June 13th, 2012

The **Feature** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the Feature engine (the [What Happened](http://www.skorkin.com/2010/08/ide-tools-what-happened-feature-ui-window/) window). The features list is available on the Core | Features options page in the [Options Dialog](http://www.skorkin.com/2010/08/ide-tools-options-dialog/). The engine also collects features execution count statistics on the Core | Features Statistics options page.

Methods of this service:

|  |  |
| --- | --- |
| Name | Description |
| AddFeature(String) | Add a new feature specified by its name into the list of features. |
| AddShortcut(String) | Add a new shortcut specified by its key combination into the list of features. |
| AddTemplate(String) | Add a new code template into the list of features. |
| CanExecuteFeature(IFeature) | Returns true if the specified feature is not disabled by the Engine and can be performed. |
| CanExecuteShortcut(ShortcutBase, ShortcutBase, String, String, String) | Returns true if the specified shortcut is not disabled by the Engine and can be handled. |
| CanExecuteTemplate(String, String, String, String) | Returns true if the specified template is not disabled by the Engine and can be expanded. |
| Disable(String) | Disables the specified feature. |
| Enable(String) | Enables the specified feature. |
| GetFeatureState(String) | Returns the state of the specified feature (enabled, disabled or missing). |
| GetRequestCount(String) | Returns the number of requests of the specified feature. |
| IncreaseQuery(ProviderBase) | Increases the execution count of the specified feature. |
| RemoveFeature(String) | Removes the feature from the Features engine list. |
| ShowHintWindow(ActionWindowHint) | Shows the **Feature UI** (**What Happened**) hint. |
| VSBindingMatchesFirstKey (ShortcutBase) | Returns true if the Visual Studio key binding exists with the same key combination as the passed in shortcut. |

## DXCore Services – File

February 19th, 2011

**File** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for file reading, writing, and editing.

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Activate | Activates the code view of the specified file name. If the code view is not available, it is opened. |
| ActivateHtmlView | Activates the HTML view of the specified file name. If the HTML view is not available, the code file is opened. |
| ApplyChanges | Applies the specified file changes collection, replaces the text in several files. |
| ChangeFile | Sets the text in the given ranges to the specified text. This can be used to delete, replace or insert text. |
| IsOpen | Returns true if the specified file is open in any kind of view (e.g., code view, design view). |
| ReadTextFile | Reads the contents of a text file and returns them in a StringCollection or as an array of strings (another overload). |
| WriteTextFile | Writes the contents of a string array or a StringCollection to a text file. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Active | Returns the file name of the active document. |

## DXCore Services – Graphics

October 31st, 2011

The **Graphics** [DXCore Service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for GDI graphic manipulation.

Here they are:

|  |  |
| --- | --- |
| Method name | Description |
| CorrectPoint(Point, Int32, Int32) | Translates the given point by the specified amount. |
| CorrectRectangle(Rectangle, Int32, Int32) | Translates the given rectangle by the specified amount. |
| CorrectRectangle(Rectangle, Point) | Translates the given rectangle by the specified point. |
| DrawArrow(Graphics, Point, Point, Int32, Color) | Draws an arrow at the specified location with the specified color. |
| DrawText(Graphics, StringFormat, String, Font, Brush, Int32, Rectangle) | Draws the specified text. |
| GetBitmapMask(Bitmap, Color) | Returns a bitmap that is the mask of the given Bitmap using the specified color. |
| GetBorderPoints(TextView, SourceRange, SourcePoint, SourcePoint) | Returns the left and right source points that are the border of the given source range. |
| GetImageFromWindow(HWND, Rectangle) | Returns a bitmap copied from the specified rectangle in the specified window. |
| GetPathForBezierCurve(PointF, PointF, PointF, PointF, Single) | Creates and returns a GraphicsPath that surrounds the bezier curve specified by the p1, c1, c2 and p2, widened by the specified radius. Returns null if the specified bezier curve has no length, or if radius is less than or equal to zero. |
| GetRectangleFromMaxRange (TextView, SourceRange) | Returns the rectangle coordinates of the given source range. |
| GetScreenAt(Int32, Int32) | Returns the screen at the specified coordinates. |
| GetScreenAt(Point) | Returns the screen at the specified point. |
| GetTextWidth(Graphics, Font, String) | Returns the exact width of the specified text. |
| IsValidBitmap(Bitmap) | Returns false if all pixels of the bitmap have same color. |
| LoadIndependentBitmap(String) | Loads a bitmap into memory from a filename. This method avoids the file lock that GDI+ would normally create. |
| MakeBitmapFromRange(TextView, SourceRange) | Returns a bitmap of the specified source range. |
| MakeBitmapFromString(String, Graphics, Boolean, Color, Color, Color, Int32) | Returns a bitmap representing the given string with the specified text and background colors. |
| MakeBitmapFromString(String, Graphics, Color, Color) | Returns a bitmap representing the given string with the specified text and background colors. |
| MeasureString(Graphics, String, Font) | Returns a bounding box for an unwrapped string. This method is more precise than Graphics.MeasureString, and will measure right to the end of the last character (including trailing spaces). |
| MeasureStringCollection (StringCollection, Graphics) | Returns a bounding box for a given string collection. |
| PaintPlus(Graphics, Pen, Int32, PointF) | Paints a “+” at the specified center point using the specified pen. |
| ReplaceBitmapColor(Bitmap, Color, Color) | Replaces a color in a System.Drawing.Bitmap with a new color. This method works on both indexed and non-indexed bitmaps. |
| SetImageOpacity(Image, Single) | Changes the opacity of the image. |
| StringRectF(Graphics, String, Font, Rectangle, StringFormat) | Returns a RectagleF instance for the given string and the specified parameters. |

## DXCore Services – Hints

December 14th, 2011

The **Hints** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to hinting services for showing/hiding Shortcut hints, BigFeedback hints, and other discoverability services.

Methods of this service:

|  |  |
| --- | --- |
| Method Name | Description |
| GetAvailableShortcuts | Collects all available shortcuts and descriptions for the current context for use in the Shortcuts hint. |
| HideFeatureActionsHint(HintWindowBase) | Hides the specified [Feature UI](http://www.skorkin.com/2010/08/ide-tools-what-happened-feature-ui-window/) hint. |
| HideShortcutsHint | Hides the active Shortcuts hint. |
| HideShortcutsHint(Boolean) | Hides the active Shortcuts hint. The boolean parameters specify whether the hint animates the close, so the hint appears to drop down into it’s bottom line. |
| HideShortcutsHint(ShortcutsHint) | Hides the specified Shortcuts hint. |
| HideShortcutsHint(ShortcutsHint, Boolean) | Hides the specified Shortcuts hint. The boolean parameters specify whether the hint animates the close, so the hint appears to drop down into it’s bottom line. |
| HideSolutionProcessHint(HintWindowBase) | Hides the SolutionProcess hint. |
| HideTooltip | Hides the Tooltip previously displayed through a call to ShowTooltip. |
| IsHintSuppressed(ShortcutsHint) | Returns true if the user has requested to suppress future appearances of the specified Shortcuts hint. Note that it is hidden from Intellisense. |
| IsHintSuppressed(String) | Returns true if the user has requested to suppress future appearances of the specified hint. Note that it is hidden from Intellisense. |
| IsShortcutsHintVisible(ShortcutsHint) | Return true if the specified Shortcuts hint is visible. |
| IsSolutionProcessHintVisible( HintWindowBase) | Returns true if the SolutionProcess hint is visible. |
| RefreshForm | Invalidates the Shortcuts hint form. |
| RefreshSolutionForm | Invalidates the SolutionProcess hint form. |
| SetLastTooltipControlUnderMouse (ITooltipProvider, Control) | Allows an ITooltipProvider to specify a control under the mouse. ITooltipProviders must call this method anytime the control under the mouse changes. When an ITooltipProvider is destroyed, it should call SetLastTooltipControlUnderMouse (null, null) if it previously called SetLastTooltipControlUnderMouse. This is to prevent an attempt to call back into it after the ITooltipProvider is destroyed. |
| ShowBigFeedback(HWND, String) | Shows a BigFeedback message, centered in the specified HWND. |
| ShowBigFeedback(Rectangle, String) | Shows a BigFeedback message, centered in the specified bounds. |
| ShowBigFeedback(String) | Shows a BigFeedback message, centered in the active text view. |
| ShowFeatureActionsHint(HintWindowBase) | Displays the Feature UI hint. |
| ShowShortcutsHint(ShortcutsHint) | Displays the specified Shortcuts hint. |
| ShowShortcutsHint(ShortcutsHint, Boolean) | Displays the specified Shortcuts hint.  If the boolean parameter is false, the Shortcuts hint will appear as long as it is not suppressed. If true, the Shortcuts hint will appear even if the user requested to suppress it in the past. |
| ShowShortcutsHint(ShortcutsHint, Boolean, Boolean) | Displays the specified Shortcuts hint.  If the first boolean parameter is false, the Shortcuts hint will appear as long as it is not suppressed. If true, the Shortcuts hint will appear even if the user requested to suppress it in the past. The second boolean parameter specifies whether this hint should be displayed as a design-time preview. If true, clicking the red “X” button will close the hint list, but will not suppress further showings. Note that it is hidden from Intellisense. |
| ShowShortcutsHint(ShortcutsHint, Boolean, Boolean, String, String, String, String) | Displays the specified Shortcuts hint with the specified UI text. Note that it is hidden from Intellisense. |
| ShowSolutionProcessHint(TextView, HintWindowBase) | Displays the specified SolutionProcess hint. |
| ShowSolutionProcessHint(TextView, HintWindowBase, Boolean) | Displays the specified SolutionProcess hint. Optionally specifies whether to locate the hint on the top of the text view. |
| ShowTooltip(Int32, Int32, String) | Shows the Tooltip containing the specified text at the specified coordinates. Caller is responsible for hiding the Tooltip, through a call to HideTooltip(). |
| SuppressFutureShortcutsHintAppearances( ShortcutsHint) | Suppresses the appearance of future Shortcuts hints. |
| SuppressFutureShortcutsHintAppearances( String) | Suppresses the appearance of future Shortcuts hints specified by the hint title. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| FeatureActionsHintVisible | Returns true if the Feature UI hint is visible. |
| Settings | Settings for discoverability features, used by the Hint service. |
| ShortcutsHintBounds | Returns the bounds of the currently active Shortcuts hint. |
| ShortcutsHintIsFocused | Returns true if the Shortcuts hint is visible and focused. |
| ShortcutsHintIsUp | Returns true if the Shortcuts hint has been created and displayed. Note that it is hidden from Intellisense. |
| SolutionProcessHintVisible | Returns true if the SolutionProcess hint is visible. |

## DXCore Services – IDE

November 8th, 2010

The **IDE** [service](http://skorkin.com/2010/09/dxcore-services-list/) contains properties of the main Visual Studio IDE window. The service reference itself (“**CodeRush.IDE**“) can be used as an instance of the ***IWin32Window*** type to the Visual Studio IDE window. So, for example, if you’d like to show a modal dialog and want to have the main IDE window as its owner, pass the **CodeRush.IDE** reference as a parameter to the “**ShowDialog**” method call of the dialog window.

Here are properties of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Handle | Handle of the main window in the IDE. Satisfies the IWin32Window interface. Forms displayed modally should pass CodeRush.IDE as a parameter to their ShowDialog method. |
| IsActive | Returns true if the IDE is the active (foreground) application. |
| IsMenuActive | Returns true if a main menu is active. |
| IsModal | Gets whether or not the IDE is a modal state. |
| IsSysMenuActive | Returns true if a system menu is active. |
| Mode | Returns an instance of the IDEMode enumeration type (Debug or Design) of the IDE. |
| ShuttingDown | Returns true if the IDE is currently shutting down. |
| WindowState | Returns the window state of the IDE. |

## DXCore Services – Intellassist

October 31st, 2011

The **Intellassist** [DXCore Service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for [CodeRush Intellassist](http://www.skorkin.com/2011/10/coderush-code-editor-auto-complete-features-intellassist/) feature manipulation (e.g. suspending, resuming, canceling) and additional properties.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| Cancel | Cancels Intellassist if it is active. |
| Resume | Resumes Intellassist once it was suspended. Call this after CodeRush.Intellassist.Suspend is called to end the suspended state. |
| Suspend | Puts Intellassist in a suspended state so that it will not activate. To end this suspended state, call Resume. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| Active | Determines whether Intellassist is displayed or not. |
| Suspended | Determines whether Intellassist is in a suspended state. |

## DXCore Services – Issues

July 4th, 2011

The **Issues** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for manipulating the **CodeRush** [Code Issues Analysis](http://www.skorkin.com/2011/02/coderush-code-issues-technology-overview/).

**Methods** of this service:

|  |  |
| --- | --- |
| **Method name** | **Description** |
| CanCheckCodeIssues | Returns true if code issues can be checked inside the given text document. |
| CaretInside(String) | Returns true if the caret is inside the specified issue. |
| CheckCodeIssues | Returns all code issues inside the given source file. |
| CreateTextMarker(TextMarkerType, TextDocument, SourceRange) | Creates a new text marker (code issue highlighting). |
| DrawIcon(Graphics, Rectangle, CodeIssueType) | Draws the icon of the specified code issue onto a graphics surface. |
| Get(String) | Returns the code issue provider with the specified name. |
| GetCodeSmellsInView(TextView) | Returns all code issues of the [code smell type](http://www.skorkin.com/2011/02/code-issues-types-overview/) inside the given text view. Note that it is hidden from Intellisense. |
| GetEnumerable | Returns an IEnumerableimplementer which enumerates a given element and all it’s children recursively. |
| GetFirstIssueAtCaret | Returns the first code issue found at the caret position. |
| GetFirstIssueAtCaret( Predicate<CodeIssue>) | Returns the first code issue found at the caret position. |
| GetIssueAfter( IEnumerable<CodeIssue>, SourcePoint) | Returns the closest code issue after the given source point. |
| GetIssueAfter(SourceFile, SourcePoint) | Returns the closest code issue after the given source point inside the specified source file. |
| GetIssueAfter(SourcePoint) | Returns the closest code issue after the given source point inside the active document. |
| GetIssueBefore( IEnumerable<CodeIssue>, SourcePoint) | Returns the closest code issue before the given source point. |
| GetIssueBefore(SourceFile, SourcePoint) | Returns the closest code issue before the given source point inside the specified source file. |
| GetIssueBefore(SourcePoint) | Returns the closest code issue before the given source point inside the active document. |
| GetIssuesAtPoint | Returns a list of code issues at the specified source point. |
| GetIssuesInDocument | Returns a list of code issues found in the given text document. |
| GetIssuesInView(TextView) | Returns a list of code issues found in the given text view. |
| GetIssuesStorage | Returns a [DecoupledStorage](http://www.skorkin.com/2010/12/dxcore-decoupledstorage-object-for-storing-data/)instance for this service. |
| GetLastIssueAtCaret | Returns the last code issue found at the caret position. |
| GetNextIssue(CodeIssue) | Returns a code issue followed by the specified issue in the active source file. |
| GetPreviousIssue(CodeIssue) | Returns a code issue followed before the specified issue in the active source file. |
| GetSuppressionScope(IElement) | Returns the suppression scope for the given element. |
| GetWideAnalysisProgressInfo | Returns the  current state of the [Solution Wide Analysis](http://www.skorkin.com/2011/05/code-issues-configuration-and-options/). |
| GetWorst( IEnumerable<CodeIssue>) | Returns the worst code issue type inside the given list of issues. |
| GetWorst(SourceFile) | Returns the worst code issue type inside the given source file. |
| GetWorst(TextDocument) | Returns the worst code issue type inside the given text document. |
| InvalidateCache(SourceFile) | Invalidates the code issues cache inside the given source file. Note that it is hidden from Intellisense. |
| InvalidateCache(TextDocument) | Invalidates the code issues cache inside the given text document. Note that it is hidden from Intellisense. |
| InvalidateCodeIssueMarkers | Invalidates the code issues markers inside the given text document. Note that it is hidden from Intellisense. |
| InvalidateSolutionUI | Invalidates the code issues solution UI. |
| InvalidateUI | Invalidates the code issues UI. |
| IsLanguageSupported(String) | Returns true if the code issues engine supports the specified language. |
| IsSuppressed(CodeIssueProvider, IElement) | Returns true if the specified code issue provider is suppressed for the given element. |
| IsSuppressed(CodeIssueProvider, ISuppressionScope) | Returns true if the specified code issue provider is suppressed for the given scope. |
| RefreshActiveDocumentIssues | Checks the active text document for code issues and updates them. |
| SortByRange( IEnumerable<CodeIssue>) | Sorts an array of code issues by their range. |
| StopCalculation | Stops the current code issues search if it is running. Note that it is hidden from Intellisense. |

**Properties**:

|  |  |
| --- | --- |
| **Property name** | **Description** |
| CalculationIsActive | Returns true if the code issues search is in progress. |
| CanCheckIssuesInXAML | Returns true if code issues can be checked in XAML. |
| CodeFixHintIsActive | Returns true if the [code fix hint](http://www.skorkin.com/2011/03/code-issues-visual-presentation/) is active. |
| IsEnabled | Returns true if the code issues analysis is enabled. |
| IsSolutionWide | Returns true if the [solution wide analysis](http://www.skorkin.com/2011/05/code-issues-configuration-and-options/) is turned on. |
| Providers | Returns a list of registered code issue providers. |
| ZoomWindowHintIsActive | Returns true if the [Code Issues Zoom Window](http://www.skorkin.com/2011/03/code-issues-visual-presentation/) hint is active. |

## DXCore Services – Key (Keyboard)

June 28th, 2011

The **Key** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to keystrokes, key names, and key mapping methods and properties. This service can be used for retrieving shortcuts [bound to the specific actions](http://www.skorkin.com/2010/09/how-to-assign-a-shortcut-key-to-a-particular-action/) or an entire shortcut binding scheme that you can see on the [Shortcuts](http://www.skorkin.com/2011/08/ide-tools-coderush-shortcuts-options-page/) options page.

**Methods** of the **Key** service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| ActionToKeyNames(String) | Returns a list of key names associated with the passed Action. |
| CodeToString(UInt32, UInt32) | Returns a string presentation of the specified virtual key and scan code. |
| GetAllBindings | Returns an array of all registered command key bindings. |
| GetAllBindings(String) | Returns a NameValueCollection of key names and bindings associated with the passed command. If the command specifies no parameters, and bindings exist to the command that accept parameters, all key bindings will be returned. The Name field holds the key name, and the Value field holds the command with any parameters (e.g., “Embed(try/finally)”). |
| GetAllContextMatchBindings | Returns a NameValueCollection of key names and context satisfied bindings associated with the passed command. If the command specifies no parameters, and bindings exist to the command that accept parameters, all key bindings will be returned. The Name field holds the key name, and the Value field holds the command with any parameters (e.g., “Embed(try/finally)”). |
| GetAsyncKeyState(Int32) | Determines whether a key is up or down at the time the function is called, and whether the key was pressed after a previous call to GetAsyncKeyState. |
| GetName | Returns a string that is the name of a key for the current locale. |
| HasEnabledShortcut | Determines if the specified command has at least one [DXCore](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/) shortcut that is enabled. |
| MapVirtual(Int32) | Translates (maps) a scan code into a virtual-key code that does not distinguish between left- and right-hand keys. If there is no translation, the function returns 0. |
| MapVirtualLeftRight(Int32) | Translates (maps) a scan code into a virtual-key code that distinguishes between left- and right-hand keys. If there is no translation, the function returns 0. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| AltKeyName | Returns the name of the Altkey. |
| CtrlKeyName | Returns the name of the Ctrl key. |
| SendShortcutToIDE | Command event handlers can change this property to determine whether the key bound to their **CodeRush** command should go on to the IDE for further processing. This property is initialized by **DXCore** before calling the CodeRushCommand’s DoExecute() method. The initialized value is determined by the user-specified binding option established in the Shortcuts option dialog. This property is checked immediately after calling DoExecute(), so if your event handler doesn’t touch it, it will respect the user’s settings. |
| ShiftKeyName | Returns the name of the Shift key. |
| WaitingForSecondKeyOfChord | Returns true if the Visual Studio IDE is waiting for a second key of chord (used for multiple key shortcuts, VS is waiting for a second part of a key combination). |

## DXCore Services – Language

April 11th, 2011

The **Language** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) is one of the most important services that provides access to the programming language fundamentals, including access to language extensions. Language extensions are **DXcore plug-in extensions**, which implement specific programming language support for Visual Studio, including parsers and code generators.

Lots of the methods of this service have two additional overloads, which take a string parameter identifying the programming language (e.g. “CSharp“, “Visual Basic“) or a LanguageExtensionBase instance of the particular language implementation. You are welcome to pass a specific value (a language you would like to work with) for these parameters, otherwise, the active language will be used, if they are not specified.

Here are the methods of this **DXCore** service:

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| AddTypeSuffix | Adds the type suffix (e.g. ‘\*’ or ‘^’ in C++) for the specified full type name and element. This method will try to obtain a simple type name with the namespace stripped off. |
| ChangeElementSectionVisibility( AccessSpecifiedElement, MemberVisibility) | Changes the visibility of the section of the specified element in the active text document. Note, visibility sections are used in C++ and F# languages. |
| CheckBaseMemberVirtualModifier | Returns true if the virtual modifier should be checked when searching for a base member (in other words, current language has a serial members order, e.g. C++). |
| CheckQualifiedName(String, String) | Returns true if the given name is qualified by the specified delimiter operators. |
| CheckSimpleName(String, String) | Returns a simple name for the given name qualified by the specified delimiter operators. |
| CreateLanguageElement( LanguageElementType) | Returns a new language element for the given language element type. |
| CreateParserFromLanguageID(String) | Creates a new parser that is capable of parsing text documents matching the specified language ID. |
| DeclaresTypeBeforeName | Returns true if the current language declares the type before the name in variable or parameter declarations (e.g. CSharp). |
| DisableLanguage(String) | Disables parsing for the given language in the current Visual Studio session. |
| EnableAllLanguages | Enables parsing for all languages. |
| EnableLanguage(String) | Enables parsing for the given language. |
| ExtractFirstIdentifier(String) | Removes the first identifier from the passed-in qualified name and returns that identifier. For example, if the qualified name is “System.Boolean“, this method will return “System” and will set the passed-in qualified name to “Boolean“. |
| GenerateElement( LanguageElement) | Generates code for the given language element for the active language. Note, this method always adds a carriage return line feed at the end. |
| GenerateExpressionCode( Expression) | Generates code for the given expression in the active language. Note, this method doesn’t add a carriage return line feed at the end. |
| GenerateXmlCode( XmlDocComment) | Generates code for the given XmlDocCommentfor the active language. |
| GetActiveCodeDomProvider | Returns a descendant of the CodeDomProvider type for the active language, if possible. |
| GetActiveCodeGenerator | Returns the active code generator. |
| GetActiveElementBuilder | Returns the active ElementBuilderinstance. |
| GetActiveEnvDTEProject( ProjectElement) | Returns an instance of the EnvDTE.Project for the given ProjectElementinstance. |
| GetActiveExpressionCodeGenerator | Returns the expression code generator for the active language. |
| GetActiveExpressionParser | Returns the expression parser for the active language. |
| GetActiveLanguageID(IElement) | Returns a language ID (as a string) for ASP.NET embedding, code nuggets, asp directives, scripts, or web event handlers. |
| GetBaseTypeName(String) | Returns a simple type name (e.g. ‘bool‘) for the given type full name (“System.Boolean“). |
| GetBinaryOperatorType( PrimitiveType, PrimitiveType) | Returns the resulting type after performing a binary operation on the two given types. |
| GetBlockNecessityInformation( ParentingStatement) | Returns information about the resulting block for the given statement after inserting a new language element. |
| GetBuiltInTypeNarrowingConversionTable | Returns an explicit conversion table for the built-in types. |
| GetBuiltInTypeWideningConversionTable | Returns an implicit conversion table for the built-in types. |
| GetCaretLocation | Return the current CaretLocationinstance for the active language. |
| GetCodeBlockStart( LanguageElement, SourcePoint) | Returns the first available insertion point for the given code block. |
| GetCodeDomProvider(String) | Returns a descendant of the CodeDomProviderclass for the specified language, if possible. |
| GetCodeGenerator( LanguageExtensionBase) | Returns code generator for the given language extension. |
| GetCodeGenerator(String) | Returns the code generator for the given language. |
| GetCollapsibleRange( LanguageElement) | Returns a SourceRangerepresenting the range of the specified [LanguageElement](http://www.skorkin.com/2011/01/dxcore-abstract-source-tree-structure/) that can be collapsed for the active language. |
| GetComment(String) | Returns a single-line comment for the specified text in the active language. |
| GetContinueStatement | Returns a continue statement for the given language, if supported. |
| GetDefaultAssemblyReferences( String) | Returns a list of default assembly references for the given language (e.g. “Microsoft.VisualBasic” for VB). |
| GetDefaultValueExpression( PrimitiveExpression) | Returns default value expression for the given primitive expression. |
| GetElementBuilder(String) | Returns an ElementBuilderinstance specific for the given language. |
| GetElementFullName( IElement) | Returns a language specific full name for the given element. |
| GetEscapedString( String, Boolean) | Returns an escaped version of the passed-in string. For example, in C# this would convert all \r’s to carriage-returns, all \n’s to line-feeds, etc. |
| GetExpressionCodeGenerator( String) | Returns an expression code generator for the given language. |
| GetExpressionParser(String) | Returns an expression parser for the given language. |
| GetFileLanguageID(SourceFile) | Returns the language ID for the given SourceFile instance. |
| GetFullTypeName | Returns a full type name for the given built-in type. |
| GetHexadecimalValuePresentation( Byte) | Returns a hexadecimal value presentation (e.g. “0xFF” in C#, “HFF” in VB). |
| GetHexadecimalValuePresentation( Byte, String) | Returns a hexadecimal value presentation (e.g. “0xFF” in C#, “HFF” in VB). |
| GetIdentifierFromKeyword(String) | Returns an identifier from the keyword for the active language. If the passed-in string is not a keyword, then method does not process it. |
| GetImplementsCode(Interface, Member, Boolean) | Returns the code for the given member implementation. |
| GetImportsStatement(String) | Returns a properly-formatted Imports statement (VB) or using directive (C#) for the specified namespace. Returns an empty string, if an error occurs. This method is identical to GetUsingDirective. |
| GetLanguageElementCodeGenerator( LanguageElement, String) | Returns the language element generator instance for the given language. |
| GetLanguageExtension(String) | Returns a LanguageExtensionBase instance for the given language ID. |
| GetLanguageExtension- FromDotNetLanguageType( DotNetLanguageType) | Returns a LanguageExtensionBaseinstance for the given DotNetLanguageTypeenumeration. |
| GetLanguageExtensionFromFileExt(String) | Returns a LanguageExtensionBaseinstance for the given file extension. |
| GetLanguageID(DotNetLanguageType) | Returns a language ID for the given DotNetLanguageType enumeration. |
| GetLanguageID(IElement) | Returns a language ID for the given element. |
| GetLanguageID(String) | Returns a language ID for the given file extension. |
| GetMacroExpansion(SourceFile, TextDocument, SourceRange) | Returns a string, containing a full expansion of the specified macro. |
| GetNullCheck(String) | Returns a language specific check for nullness. |
| GetNullReferenceExpression(String) | Returns a null reference expression for the given language (e.g. ‘null‘ for C#, ‘Nothing‘ for Visual Basic). |
| GetObjectTypeReferenceExpression | Returns a System.Object type reference expression. |
| GetOverloadableOperatorName( OperatorType, Int32) | Returns an operator name (e.g. “op\_Addition” for the given operator type. |
| GetParserFromFileExt(String) | Returns a parser that is capable of parsing files matching the specified file extension. |
| GetParserFromLanguageID(String) | Returns a parser that is capable of parsing text documents matching the specified language ID. |
| GetProjectLanguageID(Project) | Returns a language ID for the given project. |
| GetProjectLoader(String) | Returns a project loader for the given language. |
| GetRegionFooter | Returns a properly formatted region directive footer. |
| GetRegionHeader(String) | Returns a properly-formatted region directive header with the passed-in caption. |
| GetRegionHeaderAndFooter(String, String, String) | Returns properly formatted region directive header and footer with the passed-in caption. |
| GetSealedKeyword | Returns a sealed keyword (e.g. ‘sealed‘ in C#, ‘NotInheritable‘ in VB) for the active language, if supported. |
| GetSimpleName(String) | Returns a simple name for the given full type name. |
| GetSimpleTypeName(String) | Returns a simple type name for the given full type name. |
| GetSupportedFileExtensions(String) | Returns a list of supported file extensions for the given language. |
| GetTokenCategory(Token) | Returns a token category string. |
| GetTokens(String) | Returns a collection of tokens from the given text. |
| GetTypeKindKeyword(TypeDeclaration) | Returns a keyword that defines the given type declaration. |
| GetTypeParameterInitializer(String) | Returns a default initializer for the specified type parameter name. |
| GetTypeSuffix(String) | Returns a type suffix for the given type name. For example, ‘\*’ or ‘^’ for C++ language. |
| GetUsingDirective(String) | Returns a properly formatted ‘using’directive (C#) or Importsstatement (VB) for the specified namespace. Returns an empty string, if an error occurs. This method is identical to GetImportsStatement. |
| GetVariableModifier(String) | Returns a variable modifier for the given language. |
| GetVariableRemoveRanges(Variable) | Returns a list of SourceRangesfor the given Variable. Useful for a multi-variable declarations. |
| GetVisibility(MemberVisibility, String) | Returns a language specific ‘visibility’ keyword name. |
| GetXmlCodeGenerator(String) | Returns a XmlDocComments generator for the given language. |
| HasManagedPointers | Returns true if the current language uses managed pointers (e.g. C++). |
| HasSerialMembersOrder | Returns true if the current language has serial members order (e.g. C++). |
| IdentifiersMatch(String, String) | Returns true if the specified identifiers are equivalent. |
| ImplementInterface(TypeDeclaration, Interface) | Implements the given interface for the given type. |
| InheritType(TypeDeclaration, TypeDeclaration) | Inherits the specified type for the given type (makes the given type a descendant of the specified type). |
| IntroduceComments( LanguageElement, CommentCollection) | Adds comments from the given comment collection to the specified language element node. |
| IsBaseInstanceKeyword(String, String) | Returns true if the given string equals the ‘base’ instance keyword for the given language. |
| IsBasicLanguage(String) | Returns true if the specified language is Visual Basic. |
| IsBuiltInType(String) | Returns true if the given type name is a language built-in type name (e.g. “object” in C#, “Object” in VB). |
| IsCaseSensitiveLanguage(String) | Returns true if the language is case-sensitive. |
| IsClassInstanceKeyword(String, String) | Returns true if the given string equals a class instance keyword for the given language. |
| IsCode(CaretLocation) | Determines if a CaretLocationis inside the source code. |
| IsCollapsible(LanguageElement) | Determines whether or not the specified LanguageElement is collapsible for the active language. |
| IsComment(CaretLocation) | Determines if the specified CaretLocationis a comment. |
| IsCPlusPlusLanguage(String) | Returns true if specified language is C/C++. |
| IsCSharpLanguage(String) | Returns true if specified language is CSharp. |
| IsDefaultPrimitiveValue( ITypeElement, IExpression) | Returns true if the given expression is primitive and has a default value for the specified type. |
| IsExtensionMethod(IMethodElement) | Returns true if the given method is an extension method. |
| IsHTML\_XMLLanguage(String) | Returns true if the specified language is HTML/XML. |
| IsHtmlLanguage(String) | Returns true if the specified language is HTML. |
| IsHtmlOrXmlLanguage(String) | Returns true if the specified language is HTML/XML. |
| IsInCPlusPlus(IElement) | Returns true if the specified element is inside the C/C++ language. |
| IsInCSharp(IElement) | Returns true if the specified element is inside the CSharp language. |
| IsInVisualBasic(IElement) | Returns true if the specified element is inside the Visual Basic language. |
| IsJavaScriptLanguage(String) | Returns true if the specified language is JavaScript. |
| IsKeyword(String) | Returns true if the specified string is a legal keyword for the active language. |
| IsKeyword(String, String) | Returns true if the specified name is a keyword for the given language. |
| IsKeywordElement(IElement) | Returns true if the specified element is a keyword element for the active language. |
| IsLanguageCaseSensitive(IElement) | Returns true if the given element’s language is case-sensitive. |
| IsLanguageDisabled(String) | Returns true if parsing for the specified language is disabled. |
| IsLeadingIdentifierChar(Char) | Returns true if the given character qualifies as a character that can start an identifier. |
| IsNativeType(TypeDeclaration) | Returns true if the given type declaration is native (not managed). |
| IsNumber(String) | Returns true if the specified string is a legal number for the active language. |
| IsOptionExplicitOff(IElement) | Returns true if the Explicitoption is off for the source file of the specified element. |
| IsOptionInferOff(IElement) | Returns true if the Infer option is off for the source file of the specified element. |
| IsOptionStrictOff(IElement) | Returns true if the Strictoption is off for the source file of the specified element. |
| IsPlainTextLanguage(String) | Returns true if the specified language is plain text. |
| IsQualifiedName(String) | Returns true if the given name is qualified. |
| IsSimpleIdentifier(String) | Returns true if the specified string is a legal simple identifier for the active language. A simple identifier typically starts with an alpha character, and can include numbers and the underscore character. Parens, brackets, and internal whitespace around member access operators (e.g., “A . B . C” instead of “A.B.C”) are not included in this evaluation. |
| IsString(CaretLocation) | Returns true if the specified CaretLocationis a string. |
| IsTrailingIdentifierChar(Char) | Returns true if the given character qualifies as a character that can follow the first character of an identifier. |
| IsVisualJSharpLanguage(String) | Returns true if the specified language is Visual JSharp. |
| IsXAMLLanguage(String) | Returns true if the specified language is XAML. |
| IsXMLLanguage(String) | Returns true if the specified language is XML. |
| LoadProject(Project) | Loads a project and returns its instance (as a ProjectElement) from the given EnvDTE.Project reference. |
| NeedChangeLanguage | Returns true if the current language should be changed. For example, if the given element is an ASP code embedding, html script definition, or the element is inside a web handler file, web event handler, or inside a code nugget, this function will return true. |
| Parse(LanguageElement, RegionDirective, TextStringCollection, CompilerDirective, TextDocument, SourceRange) | Parses 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 BeforeParseor AfterParseevents, 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). |
| Parse(LanguageElement, TextDocument, SourceRange) | Parses the specified source range of the given text document with the given context and returns the LanguageElement that specifies it. |
| Parse(ParserContext) | Parses 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 BeforeParseor AfterParseevents, 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). |
| Parse(String) | Parses the specified source file on disk and returns the LanguageElementthat specifies it. |
| Parse(TextDocument) | Parses the specified text document and returns the LanguageElementthat specifies it. |
| ParseActiveDocument | Parses the active text document and returns the LanguageElementthat specifies it. |
| ParseDocument(TextDocument) | Parses the specified text document. |
| ParseExpression(String) | Parses an expression from the given string. |
| ParseString(String, String) | Parses the specified string and returns the LanguageElementthat specifies it. |
| ParseTypeReferenceExpression( String, Int32, Int32, String) | Parses an expression from the given string. |
| RemoveKeywordEscapeCharacters- FromIdentifier( String) | Removes keyword escape characters (like ‘@’ in C#, or ‘[]’   in VB) from an identifier name. |
| SupportsAutoImplementedProperty- AccessorVisibility | Returns true if the given language supports a visibility keyword for auto-implemented properties. |
| SupportsDecoupledForeachDeclaration( String) | Returns true if the given language supports a decoupled declaration of the ‘foreach’ iteration variable. |
| SupportsFileBuildAction(String) | Returns true if the build action for a source file can be checked for the given language. |
| SupportsImplicitPointers | Returns true if the active language supports implicit pointer declarations, e.g. return true for C# or VB languages. C++ language requires explicit pointers. |
| SupportsMacroses(String) | Returns true if the given language supports macros. |
| SupportsMultiLineComments(String) | Returns true if the given language supports multi-line comments. |
| SupportsNestedLanguages(String) | Returns true if the given language supports nested languages. |
| SupportsNestedNamespaceSearch(String) | Returns true if the given language supports searching of the nested namespaces. |
| SupportsObjectCreationFromMethod( String) | Returns true if the given language supports creation of new objects using a method declaration. |
| SupportsOutParameters(String) | Returns true if the given language supports out parameters. |
| SupportsParenslessCalls(String) | Returns true if the given language supports method calls without parens (e.g. Visual Basic). |
| SupportsPartialClasses | Returns true if the active language supports partial classes. |
| SupportsPopulatingAnArrayWith- InitialValues(IElement) | Returns true if the language of the given element supports populating an array with initial values. For example, Visual Basic version 10 supports it. |
| SupportsPrototypesAndImplementations( String) | Returns true if the given language supports prototypes and implementations (e.g. C++). |
| SupportsRootMembers(String) | Returns true if members can be placed directly to the file or namespace (e.g. C++). |
| SupportsRootNamespace(String) | Returns true if the given language supports the root namespace. |
| SupportsTypes(String) | Returns true if the given language supports type declarations. |
| ToFileExt(String) | Returns a file extension based on the specified language ID. Illegal characters are removed or converted to legal characters. If the specified language ID is empty, this method returns an empty string. |
| TypesMatch(String, String) | Returns true if the specified types are equivalent. This method converts the specified types to simple equivalents if they exist in the active language (e.g., “System.Boolean” becomes “bool” in C#). |

**Properties**:

|  |  |
| --- | --- |
| **Property Name** | **Description** |
| Active | Returns the language ID of the active source file. |
| ActiveExtension | Returns the active **DXCore**LanguageExtensionBaseinstance, or null if no language extension is available for the active programming language. |
| ActiveParser | Returns the active language parser, or null if no parser is available for the active programming language. |
| AllKnownLanguages | Returns a list of all known languages registered in the IDE. |
| ClassInstanceKeyword | Returns the keyword for accessing the current instance of the class in a method (e.g. ‘this’ for C#). |
| DisabledLanguages | Returns the list of language ID’s with disabled parsing. |
| IsBasic | Returns true if the active file contains Visual Basic source code. |
| IsCaseSensitive | Returns true if the active file contains a case-sensitive language. |
| IsCPlusPlus | Returns true if the active file contains C/C++ source code. |
| IsCSharp | Returns true if the active file contains CSharpsource code. |
| IsHtml | Returns true if the active file has an HTMLlanguage id. |
| IsHTML\_XML | Returns true if the active file has an HTML/XML language id. |
| IsHtmlOrXml | Returns true if the active file has an HTML/XML language id or HTMLlanguage id. |
| IsJavaScript | Returns true if the active file contains JavaScriptsource code. |
| IsNativeCPlusPlus | Returns true if the active file is inside a native C++ project. |
| IsPlainText | Returns true if the active file contains plain text. |
| IsVisualJSharp | Returns true if Visual JSharp is the language used in the active file. |
| IsXAML | Returns true if the active file has XAMLlanguage ID. |
| IsXML | Returns true if the active file has XMLlanguage ID. |
| LineContinuationCharacter | Gets line continuation character for the active language (e.g., “\_” in VB). Returns an empty string if the active language does not support the line continuation character. |
| MemberAccessOperator | Gets the member access operator (e.g., “.” in C# and VB) for the active language. Returns an empty string if the active language does not support a member access operator. |
| PointerAccessOperator | Gets the character for accessing the members of a struct, class or an interface via a pointer. |
| StatementTerminator | Gets the statement terminator for the active language. For example, in C#, statements are terminated with a semi-colon. |
| SupportedFileExtensions | Gets supported file extensions for the active language. |
| SupportsCLRSettings | Determines whether or not the active language supports CLR settings. |
| SupportsConstantDeclarations | Determines whether or not the active language supports constant declarations. |
| SupportsTernaryExpressions | Determines whether or not the active language supports ternary expressions. |
| SupportsVisibilitySections | Determines whether or not the active language supports visibility sections. |
| SupportsWithStatement | Determines whether or not the active language supports ‘with’ statements. |
| TypeAccessOperator | Returns the character for accessing the nested elements of a type. |

## DXCore Services – LinkedIdentifiers

December 27th, 2010

[Linked Identifiers](http://www.skorkin.com/2010/12/dxcore-linked-identifiers-feature/) are a built-in feature of [DXCore](http://skorkin.com/2010/06/devexpress-dxcore-framework/), which allows you to simultaneously change similar pieces of the text (code) located in different places. The [service](http://skorkin.com/2010/09/dxcore-services-list/) provides access to the Linked Identifiers engine.

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| AddMultiLink | Creates a new multi-document linked identifier. Note that it is hidden from Intellisense. |
| BreakAllLinks | Removes all linked identifiers. |
| BreakAllLinksInRange | Removes all linked identifiers in the specified source range. |
| CloseMultiLidContainerGroup | Closes IMultiLidContainerGroup opened by the OpenMultiLidContainerGroup method. |
| ConsolidateLinks | Consolidates linked identifiers in the specified array of source ranges. |
| Find | Returns an array of ILinkedIdentifiers from the specified TextDocument entirely contained inside the given range. |
| FindLinkRanges | Returns an array of ILinkedIdentifiersInfo from the specified TextDocument entirely contained inside the given range. |
| GetStorage | Returns the ILinkedIdentifierStorage instance for the specified TextDocument. |
| Invalidate | Refreshes the state of all ILinkedIdentifiers in the given TextDocument. This may be required after creating any new ILinkedIdentifiers, to ensure that they are properly highlighted. |
| LockDelete | Call this to keep linked identifiers from being deleted while making text changes. Generally, this will not be required. When finished, call UnlockDelete. |
| NewMultiDocumentContainer | Creates a new multi-document linked identifier container. |
| OpenMultiLidContainerGroup | Opens a new multi-document linked identifier container group. |
| UnlockDelete | Call this after LockDelete to allow linked identifiers to be deleted again. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Active | Returns true if there are any linked identifiers that are currently active (being edited). |
| ActiveLink | Gets the linked identifier that is currently active (being edited). |
| ActiveStorage | Gets the ILinkedIdentifierStorage instance for the active TextDocument. |
| Changing | Indicates if the text in the linked identifiers is changing. Note that it is hidden from Intellisense. |
| DeleteLocked | Determines whether or not linked identifiers will be deleted when making text changes. |
| LinkedIdentifierPropagation | Gets or sets the ILinkedIdentifierPropagation implementer, which specifies how propagation is applied for the linked identifiers. |

## DXCore Services – Markers

July 4th, 2011

The **Markers** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for dropping, swapping, collecting and managing [navigation markers](http://www.skorkin.com/2010/09/navigation-markers/) and [bookmarks](http://www.skorkin.com/2012/05/coderush-and-visual-studio-bookmarks/) in source code.

**Methods** of this service:

|  |  |
| --- | --- |
| **Method name** | **Description** |
| AddBookmark(TextDocument, Int32, Int32) | Adds a bookmark at the specified position. |
| Clear | Clears all markers and invalidates all text views of the active text document. |
| Collect | Moves the editor cursor to the topmost marker on the stack, and removes that marker from the stack. |
| Drop (52 overloads) | Drops a marker in the active document at the current caret position. |
| DropSelection (52 overloads) | Drops a selection marker in the active document at the current caret position. |
| GetEnumerator | Returns a markers’ enumerator. |
| GotoBookmark(String) | Moves the editor caret to the specified bookmark. |
| GotoNextBookmark | Moves the editor caret to the next bookmark. |
| GotoPreviousBookmark | Moves the editor caret to the previous bookmark. |
| Remove(IMarker) | Removes the specified marker. |
| RemoveBookmark(String) | Removes the specified bookmark marker. |
| Swap | Swaps the editor caret with the topmost marker and drops a new marker at the source position (initial caret position). |
| ToggleBookmark(String, TextDocument, Int32, Int32) | Toggles the bookmark at the specified position (adds or removes a bookmark). |

**Properties**:

|  |  |
| --- | --- |
| **Property name** | **Description** |
| Count | Returns the total number of markers in this session. |
| Item | Returns a marker by the specified index. |
| TemporalMarkerLifeInSeconds | The life of a temporal marker, in seconds. |
| Top | Returns the top marker on the marker stack. |

## DXCore Services – Menus

May 3rd, 2011

The **Menus** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to Visual Studio and [DXCore](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/) menus.

**Methods** of this service:

|  |  |
| --- | --- |
| GetMemberPopupMenuItems (LanguageElement) | Returns a list of MenuItemsappropriate for the specified [LanguageElement](http://www.skorkin.com/2011/01/dxcore-abstract-source-tree-structure/). Note that it is hidden from Intellisense. |
| GetMenuBar (String) | Returns a menu bar instance by the given name. |
| SetPopupMember (LanguageElement) | Sets the LanguageElement associated with the [Member popup menu](http://www.skorkin.com/2010/11/visualization-member-icons/). Note that it is hidden from Intellisense. |

**Properties**:

|  |  |
| --- | --- |
| Bars | Provides access to various menu bars in Visual Studio. |
| DXCore | Gets the instance of the **DXCore** top-level menu (in other words, the **DevExpress** menu). |
| DXCoreMenuName | Gets the name of the **DevExpress**menu. Note that it is hidden from Intellisense. |
| Main | Gets the instance of the **Main** Visual Studio top-level menu. |
| PopupMember | Gets the LanguageElementinstance associated with the active [Member popup menu](http://www.skorkin.com/2010/11/visualization-member-icons/). The Memberpopup menu is a context menu which appears when a member icon is clicked. |
| ToolWindows | Gets the Tool Windows sub-menu inside the **DXCore** top-level menu. |
| ToolWindowsMenuName | Gets the name of the **DevExpress**Tool Windows menu. Note that it is hidden from Intellisense. |
| VisualizeToolBar | Gets the instance of the [DXCore Visualize Toolbar](http://www.skorkin.com/2010/09/ide-tools-dxcore-visualize-toolbar/). Note that it is hidden from Intellisense. |

## DXCore Services – Navigation

May 22nd, 2011

The **Navigation** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the [navigation providers’ engine](http://www.skorkin.com/2011/03/coderush-navigation-providers-engine-overview/). This service is dedicated to control navigation providers and allows you to navigate inside your code structure programmatically.

Here are the **methods** of this **DXCore** service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Execute(String) | Executes the navigation provider with the given name. |
| ExecuteIfAvailable(String) | Executes the navigation provider with the given name. Performs an availability check before executing. |
| Get(String) | Returns an instance of the navigation provider by the specified name. |
| Navigate(IElement) | Navigates to the specified element. |
| Navigate(IElement, Boolean) | Navigates to the specified element and selects the name of the target element if specified. Note that it is hidden from Intellisense. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| AvailableProviders | Gets all available navigation providers for the current context. |
| CanShowSmartTag | Returns true if there is at least one navigation provider available; otherwise, false. |
| LastExecutedProvider | Returnы the last applied navigation provider. |
| Providers | Gets all registered navigation providers. |

## DXCore Services – Options

February 5th, 2011

This [DXCore service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for manipulation of [options pages](http://www.skorkin.com/2010/12/dxcore-plug-in-option-pages-overview/) and [decoupled storages](http://www.skorkin.com/2010/12/dxcore-decoupledstorage-object-for-storing-data/) for persisting plug-in settings.

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| CollectFullNames | Returns a string array containing the full names of all registered options pages. |
| CollectFullNames(UserLevel) | Returns a string array containing the full names of all registered options pages whose UserLevel is equal to or less than the specified level. |
| GetCategory | Returns the category of an options page. |
| GetFullName | Returns the full name of an options page. |
| GetLevel | Returns the user level for an options page. |
| GetPage | Returns the options page specified by a full name or its type instance. |
| GetPageName | Returns the name of an options page specified by its type instance. |
| GetRegisteredCategories | Returns a list of all registered options page categories. If this method is called when **DXCore**/**CodeRush** is not loaded, it uses the last known list of registered categories. |
| GetStorage | Creates and returns an instance of the [DecoupleStorage](http://www.skorkin.com/2010/12/dxcore-decoupledstorage-object-for-storing-data/) object specified by the page full name or its type instance. |
| GetStorageFile | Returns the file name used by the specified options page for a language-neutral storage. |
| IsRegistered | Determines whether an options page is registered. |
| Reload | Reloads options by triggering the OptionsChanged event. |
| ReloadAll | Reloads all DXCore options by triggering the OptionsChanged event, causing event listeners to read data from decoupled storage. You can set the directory where **DXCore** reads settings by changing the Paths property. |
| Show | Displays the [DXCore options dialog](http://skorkin.com/2010/08/ide-tools-options-dialog/). |
| Show(String) | Displays the **DXCore** options dialog and selects the specified page full name. |
| Show(Type) | Displays the **DXCore** options dialog and selects the specified page type instance. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Item[] | Gets an options page by an index or a full name. |
| PageCount | Gets the number of registered options pages. |
| Paths | Provides access to various **DXCore**/**CodeRush** paths, for example: setting file storage physical path, cache files path, install directory path, different registry paths, and others. | VisualStudioRegistryPath | Gets the registry root path for the current Visual Studio IDE. |

## DXCore Services – Outline

March 16th, 2011

The **Outline** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the editor outlining, allowing you to collapse, expand, and toggle outlining.

Here are its methods:

|  |  |
| --- | --- |
| **Name** | **Description** |
| CollapseSelection | Collapses the current selection if automatic outlining is turned off. Call ExpandCurrent() to restore the collapsed region. |
| CollapseToDefinitions | Collapses all outlines to their definitions. |
| ExpandCurrent | Expands the collapsed region at the cursor. |
| StartAutomatic | Starts automatic outlining. |
| Stop | Stops all outlining (expanding all collapsed text). |
| Toggle | Toggles the expansion on the outline at the cursor. |
| ToggleAll | Expands collapsed outlines, or collapses top-level outlines (if nothing is collapsed). |

## DXCore Services – PlugIn Extensions

February 14th, 2012

The **PlugIn Extensions** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to registered DXCore plug-in extensions.

The service contains several overloads of the Find method that searches for a plug-in extension based on a stored property value. Stored properties are used by the plug-in loader to retrieve information without demand-loading the plug-in in which this extension is contained. This should only be used in special circumstances.

There are also two indexed properties:

|  |  |
| --- | --- |
| Item(String) | Returns an array of plug-in extensions by category. NOTE: This call can be expensive because it demands loading all plug-ins containing the specified category of plug-in extension. |
| Item(String, String) | Returns an array of plug-in extensions by category and language. |

## DXCore Services – Profiling

June 12th, 2012

The **Profiling** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the internal performance diagnostic tasks. See the [How to speed up the issue investigation time and help to eliminate complex problems](http://www.skorkin.com/2012/05/how-to-speed-up-the-issue-investigation-time-and-help-to-eliminate-complex-problems/) topic to learn more on how this service can be used.

Methods of this service:

|  |  |
| --- | --- |
| Name | Description |
| EndTask(IProfilingTask) | Completes the specified profiling task. |
| NeedProfileEvent (CodeRushEventKey) | Returns true if the specified event will be profiled and logged. Note that it is hidden from Intellisense. |
| Task(String) | Creates and performs the profiling task. |

Properties:

|  |  |
| --- | --- |
| Name | Description |
| LogToDisk | Returns true if the logging to disk is enabled. |
| NeedProfileContentProviders | Returns true if the content providers should be profiled. Note that it is hidden from Intellisense. |
| NeedProfileDocumentClosing | Returns true if the document closing event should be profiled. Note that it is hidden from Intellisense. |
| NeedProfileParsing | Returns true if the source code parsing should be profiled. Note that it is hidden from Intellisense. |
| NeedProfileResolving | Returns true if the symbols resolving should be profiled. Note that it is hidden from Intellisense. |
| NeedProfileTemplates | Returns true if CodeRush Code Template expansions should be profiled. Note that it is hidden from Intellisense. |

## DXCore Services – Progress

June 13th, 2012

The **Progress** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the Progress Visualizer engine. The engine creates a visualizer hint that shows the progress of the time-consuming operations (e.g. parsing of a solution). The hint may contain a progress bar and the Abort button (link).

Methods of this service:

|  |  |
| --- | --- |
| Name | Description |
| GetProgressVisualizer | Returns a new instance of the progress visualizer of the specific type. |
| IsActive(IProgressVisualizer) | Returns true if the specified progress visualizer is active. |
| SetActive(IProgressVisualizer) | Sets the active progress visualizer. |

## DXCore Services – Project

December 22nd, 2010

The **Project** [service](http://skorkin.com/2010/09/dxcore-services-list/) has a pretty short list of functions. This service is intended to manipulate projects of an active solution, and provides access to the start-up project, active project, and count of opened projects. All methods and properties work with an instance of the **DevExpress.CodeRush.Core.Project** type.

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| GetEnumerator | Returns enumerator of all projects in the active solution. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Active | Returns an instance of the active project. |
| Count | Returns count of projects in the active solution. |
| Item[Int32] | Returns an instance of a project by its index |
| StartupProject | Gets or sets the startup project for the active solution. |

he **Refactoring** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for refactoring support such as finding all required elements, an element’s declaration, its references, type name, etc.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| CalculateRequiredParameters( LanguageElement, SourceRange) | Calculates parameters that should be passed into the selected range of the code if the code is extracted from the given method or property accessor. |
| CanShowSmartTagAsync( AsyncAvailabilityEventArgs) | Returns true if there’s at least one refactoring provider available. |
| ClearFromFakeElement(String) | Removes all injected fake elements in the given file specified by its path. |
| CollectAssignments(LanguageElement) | Collects all assignments to the given local variable. |
| CollectExpressions( LanguageElement) | Collects all expressions in the given scope. |
| CollectExpressions( LanguageElement, Boolean) | Collects all expressions in the given scope. The boolean parameter specifies whether this method should drill down into the expression nodes. |
| CollectExpressionsInNodes( NodeList, Boolean) | Collects all expressions in the given node list collection. |
| CollectLocalDeclarations( LanguageElement) | Collects all local declarations within the given language element. Local declaration may be a Const, a Variable, an Initialized Variable or a Param. |
| FindAllDefineDirectives( LanguageElement) | Finds all DefineDirectives within the given scope. |
| FindAllDelegateCreationExpressions( LanguageElement, String) | Finds all delegate initializations by the specified method name within the given scope. |
| FindAllFieldReferences( LanguageElement, LanguageElement) | Finds all references to a field declaration by the specified name within the given scope. The language elements passed to this method should be determined through a call that either guarantees a parse will happen if needed, or the element was acquired directly after calling [CodeRush. Language](http://www.skorkin.com/2011/04/dxcore-services-language/). ParseIfNeeded(). |
| FindAllLanguageElements( LanguageElement, LanguageElementType) | Finds all language elements of the specified type within the given scope. |
| FindAllLanguageElements( LanguageElement, LanguageElementType, Boolean) | Finds all language elements of the specified type within the given scope. The boolean parameter specifies whether this method should drill down into the language element that has been added into the result collection. |
| FindAllLocalDeclarations( LanguageElement) | Finds all local declarations within the given scope. |
| FindAllLocalReferences( LanguageElement, LanguageElement) | Finds all references to the local declaration specified by its name within the given scope. The language elements passed to this method should be determined through a call that either guarantees a parse will happen if needed, or the element was acquired directly after calling [CodeRush. Language](http://www.skorkin.com/2011/04/dxcore-services-language/). ParseIfNeeded(). |
| FindAllMethodReferences( LanguageElement, LanguageElement) | Finds all references to the method specified by its name within the given scope. |
| FindAllNamespaceReferences( LanguageElement) | Finds all NamespaceReferences (with nested NamespaceReferences) within the given scope. |
| FindAllNamespaceReferences( LanguageElement, Boolean) | Finds all NamespaceReferences within the given scope. |
| FindAllObjectCreationExpressions( LanguageElement, String, Int32) | Finds all ObjectCreationExpressions specified by its ObjectName propety and parameters count within the given scope. |
| FindAllPrimitiveExpressions( LanguageElement, String) | Finds all PrimitiveExpressions specified by its name within the given scope. |
| FindAllReferences(IElement, IElement) | Searches for references to the specifed declaration within the given scope. |
| FindAllRelationalOperations( LanguageElement) | Finds all RelationalOperations within the given scope. |
| FindAllReturnStatements( LanguageElement) | Finds all ReturnStatements within the given scope. |
| FindAllReturnStatements( LanguageElement, Boolean) | Finds all ReturnStatements within the given scope. |
| FindAllTypeDeclarations( LanguageElement) | Finds all TypeDeclarations within the given scope. |
| FindAllTypeDeclarations( LanguageElement, Boolean) | Finds all TypeDeclarations within the given scope. |
| FindAllTypeDeclarations( LanguageElement, Boolean, Boolean) | Finds all TypeDeclarations within the given scope. |
| FindAllTypeReferenceExpressions( LanguageElement) | Finds all TypeReferenceExpressions specified by its level within the given scope. |
| FindAllTypeReferenceExpressions( LanguageElement, Int32) | Finds all TypeReferenceExpressions specified by its level within the given scope. |
| FindAllTypeReferenceExpressions( LanguageElement, Int32, Boolean) | Finds all TypeReferenceExpressions specified by its level within the given scope. |
| FindFieldDeclaration(LanguageElement) | Finds field declaration for the specified element reference. |
| FindLocalDeclaration(LanguageElement) | Finds local declaration for the specified element reference. |
| FindLocalDeclaration(LanguageElement, String) | Finds local declaration with the given name, starting from the specified element reference. |
| Get(String) | Returns the refactoring provider with the given name. |
| GetAvailability(String) | Returns the availability of the specified refactoring. |
| GetElementExpression(LanguageElement) | Returns the expression of the given element. |
| GetElementExpressionTypeName( LanguageElement) | Returns the type name of the given element expression. |
| GetElementExpressionTypeName( LanguageElement, Boolean) | Returns the type name of the given element expression. The boolen parameter specifies whether the resulting type name should be simplified. |
| GetElementType(LanguageElement) | Return the type of the given element. Note that it is hidden from Intellisense. |
| GetElementTypeName(LanguageElement) | Returns the type name of the given element. Note that it is hidden from Intellisense. |
| GetElementTypeName(LanguageElement, Boolean) | Returns the type name of the given element. Note that it is hidden from Intellisense. |
| GetExpressionFullTypeName(Expression) | Returns the full type name of the given expression. Note that it is hidden from Intellisense. |
| GetExpressionType(Expression) | Returns the type of the given expression. Note that it is hidden from Intellisense. |
| GetExpressionTypeFromContext( Expression) | Returns the type name of the given expression by analyzing the context of the expression. |
| GetExpressionTypeFromContext( Expression, Boolean) | Returns the type name of the given expression by analyzing the context of the expression. |
| GetExpressionTypeFromContext( LanguageElement) | Returns the type name that the given element expression should match by analyzing expression context. |
| GetExpressionTypeFromContext( LanguageElement, Boolean) | Returns the type name that the given element expression should match by analyzing expression context. |
| GetExpressionTypeName(Expression) | Returns the type name of the given expression. Note that it is hidden from Intellisense. |
| GetLanguageElement(IElement) | Gets the LanguageElement instance from the given IElement instance. This method will return null, if the IElement instance doesn’t have corresponding LanguageElement available (e.g., when IElement instance is declared inside a referenced assembly). |
| GetLanguageElementCollection( IElementCollection) | Gets the LanguageElementCollection instance from the given IElementCollection instance. This method will return an empty collection, if the IElementCollection instance doesn’t have corresponding elements available. |
| GetUniqueName(LanguageElement, LanguageElement, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueName(LanguageElement, LanguageElementType, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueName(LanguageElement, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueName(LanguageElementType, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueNameInsideScope( LanguageElement, String) | Returns a unique element name. |
| GetUniqueNameInsideScope( LanguageElement, String, Boolean) | Returns a unique element name. |
| HasExpression(LanguageElement) | Checks if the passed element has an expression. This method will return true for the following elements: if, while, do and initialized variable element. |
| HideSmartTag | Hides the smart tag if it is active. |
| InjectFakeElement(String, SourcePoint, LanguageElement) | Injects the fake element into the source tree. |
| InjectFakeElement(String, SourcePoint, LanguageElement, LanguageElementType, LanguageElementType[]) | Injects the fake element into the source tree. |
| IsAvailable(String) | Returns true if the specified refactoring provider is available. |
| ProcessInnerTypes(LanguageElement, LanguageElement) | Converts simple type references to full type references taking into account nested types of the active type. Note that it is hidden from Intellisense. |
| ResolveShortTypesToFullTypesIfNeeded( LanguageElement, LanguageElement, LanguageElement) | Converts short type names to full type names in the target element that can be located in a different type. For example, “String” -> “System.String” if “System” namespace isn’t referenced. Note that it is hidden from Intellisense. |
| SimplifyExpression(Expression) | Simplifies the given expression. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| Active | Gets the currently active refactoring. |
| ActivePicker | Gets the currently active target picker. |
| AvailableRefactorings | Returns a list of all available refactorings in the current context. |
| CanShowSmartTag | Return true if there’s at least one refactoring provider available. |
| IsActive | Returns true if a refactoring is active. |
| IsMenuActive | Returns true if the refactoring menu is active. |
| IsSmartTagActive | Returns true if the smart tag is active. |
| PickerIsActive | Returns true if the current target picker is active. |
| Providers | Returns an array of all registered refactorings. |
| SuppressActionHints | Gets or sets the value that controls the refactoring action hints availability. |

## DXCore Services – Refactoring

December 20th, 2011

The **Refactoring** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for refactoring support such as finding all required elements, an element’s declaration, its references, type name, etc.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| CalculateRequiredParameters( LanguageElement, SourceRange) | Calculates parameters that should be passed into the selected range of the code if the code is extracted from the given method or property accessor. |
| CanShowSmartTagAsync( AsyncAvailabilityEventArgs) | Returns true if there’s at least one refactoring provider available. |
| ClearFromFakeElement(String) | Removes all injected fake elements in the given file specified by its path. |
| CollectAssignments(LanguageElement) | Collects all assignments to the given local variable. |
| CollectExpressions( LanguageElement) | Collects all expressions in the given scope. |
| CollectExpressions( LanguageElement, Boolean) | Collects all expressions in the given scope. The boolean parameter specifies whether this method should drill down into the expression nodes. |
| CollectExpressionsInNodes( NodeList, Boolean) | Collects all expressions in the given node list collection. |
| CollectLocalDeclarations( LanguageElement) | Collects all local declarations within the given language element. Local declaration may be a Const, a Variable, an Initialized Variable or a Param. |
| FindAllDefineDirectives( LanguageElement) | Finds all DefineDirectives within the given scope. |
| FindAllDelegateCreationExpressions( LanguageElement, String) | Finds all delegate initializations by the specified method name within the given scope. |
| FindAllFieldReferences( LanguageElement, LanguageElement) | Finds all references to a field declaration by the specified name within the given scope. The language elements passed to this method should be determined through a call that either guarantees a parse will happen if needed, or the element was acquired directly after calling [CodeRush. Language](http://www.skorkin.com/2011/04/dxcore-services-language/). ParseIfNeeded(). |
| FindAllLanguageElements( LanguageElement, LanguageElementType) | Finds all language elements of the specified type within the given scope. |
| FindAllLanguageElements( LanguageElement, LanguageElementType, Boolean) | Finds all language elements of the specified type within the given scope. The boolean parameter specifies whether this method should drill down into the language element that has been added into the result collection. |
| FindAllLocalDeclarations( LanguageElement) | Finds all local declarations within the given scope. |
| FindAllLocalReferences( LanguageElement, LanguageElement) | Finds all references to the local declaration specified by its name within the given scope. The language elements passed to this method should be determined through a call that either guarantees a parse will happen if needed, or the element was acquired directly after calling [CodeRush. Language](http://www.skorkin.com/2011/04/dxcore-services-language/). ParseIfNeeded(). |
| FindAllMethodReferences( LanguageElement, LanguageElement) | Finds all references to the method specified by its name within the given scope. |
| FindAllNamespaceReferences( LanguageElement) | Finds all NamespaceReferences (with nested NamespaceReferences) within the given scope. |
| FindAllNamespaceReferences( LanguageElement, Boolean) | Finds all NamespaceReferences within the given scope. |
| FindAllObjectCreationExpressions( LanguageElement, String, Int32) | Finds all ObjectCreationExpressions specified by its ObjectName propety and parameters count within the given scope. |
| FindAllPrimitiveExpressions( LanguageElement, String) | Finds all PrimitiveExpressions specified by its name within the given scope. |
| FindAllReferences(IElement, IElement) | Searches for references to the specifed declaration within the given scope. |
| FindAllRelationalOperations( LanguageElement) | Finds all RelationalOperations within the given scope. |
| FindAllReturnStatements( LanguageElement) | Finds all ReturnStatements within the given scope. |
| FindAllReturnStatements( LanguageElement, Boolean) | Finds all ReturnStatements within the given scope. |
| FindAllTypeDeclarations( LanguageElement) | Finds all TypeDeclarations within the given scope. |
| FindAllTypeDeclarations( LanguageElement, Boolean) | Finds all TypeDeclarations within the given scope. |
| FindAllTypeDeclarations( LanguageElement, Boolean, Boolean) | Finds all TypeDeclarations within the given scope. |
| FindAllTypeReferenceExpressions( LanguageElement) | Finds all TypeReferenceExpressions specified by its level within the given scope. |
| FindAllTypeReferenceExpressions( LanguageElement, Int32) | Finds all TypeReferenceExpressions specified by its level within the given scope. |
| FindAllTypeReferenceExpressions( LanguageElement, Int32, Boolean) | Finds all TypeReferenceExpressions specified by its level within the given scope. |
| FindFieldDeclaration(LanguageElement) | Finds field declaration for the specified element reference. |
| FindLocalDeclaration(LanguageElement) | Finds local declaration for the specified element reference. |
| FindLocalDeclaration(LanguageElement, String) | Finds local declaration with the given name, starting from the specified element reference. |
| Get(String) | Returns the refactoring provider with the given name. |
| GetAvailability(String) | Returns the availability of the specified refactoring. |
| GetElementExpression(LanguageElement) | Returns the expression of the given element. |
| GetElementExpressionTypeName( LanguageElement) | Returns the type name of the given element expression. |
| GetElementExpressionTypeName( LanguageElement, Boolean) | Returns the type name of the given element expression. The boolen parameter specifies whether the resulting type name should be simplified. |
| GetElementType(LanguageElement) | Return the type of the given element. Note that it is hidden from Intellisense. |
| GetElementTypeName(LanguageElement) | Returns the type name of the given element. Note that it is hidden from Intellisense. |
| GetElementTypeName(LanguageElement, Boolean) | Returns the type name of the given element. Note that it is hidden from Intellisense. |
| GetExpressionFullTypeName(Expression) | Returns the full type name of the given expression. Note that it is hidden from Intellisense. |
| GetExpressionType(Expression) | Returns the type of the given expression. Note that it is hidden from Intellisense. |
| GetExpressionTypeFromContext( Expression) | Returns the type name of the given expression by analyzing the context of the expression. |
| GetExpressionTypeFromContext( Expression, Boolean) | Returns the type name of the given expression by analyzing the context of the expression. |
| GetExpressionTypeFromContext( LanguageElement) | Returns the type name that the given element expression should match by analyzing expression context. |
| GetExpressionTypeFromContext( LanguageElement, Boolean) | Returns the type name that the given element expression should match by analyzing expression context. |
| GetExpressionTypeName(Expression) | Returns the type name of the given expression. Note that it is hidden from Intellisense. |
| GetLanguageElement(IElement) | Gets the LanguageElement instance from the given IElement instance. This method will return null, if the IElement instance doesn’t have corresponding LanguageElement available (e.g., when IElement instance is declared inside a referenced assembly). |
| GetLanguageElementCollection( IElementCollection) | Gets the LanguageElementCollection instance from the given IElementCollection instance. This method will return an empty collection, if the IElementCollection instance doesn’t have corresponding elements available. |
| GetUniqueName(LanguageElement, LanguageElement, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueName(LanguageElement, LanguageElementType, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueName(LanguageElement, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueName(LanguageElementType, String) | Returns a unique element name in the given scope. Note that it is hidden from Intellisense. |
| GetUniqueNameInsideScope( LanguageElement, String) | Returns a unique element name. |
| GetUniqueNameInsideScope( LanguageElement, String, Boolean) | Returns a unique element name. |
| HasExpression(LanguageElement) | Checks if the passed element has an expression. This method will return true for the following elements: if, while, do and initialized variable element. |
| HideSmartTag | Hides the smart tag if it is active. |
| InjectFakeElement(String, SourcePoint, LanguageElement) | Injects the fake element into the source tree. |
| InjectFakeElement(String, SourcePoint, LanguageElement, LanguageElementType, LanguageElementType[]) | Injects the fake element into the source tree. |
| IsAvailable(String) | Returns true if the specified refactoring provider is available. |
| ProcessInnerTypes(LanguageElement, LanguageElement) | Converts simple type references to full type references taking into account nested types of the active type. Note that it is hidden from Intellisense. |
| ResolveShortTypesToFullTypesIfNeeded( LanguageElement, LanguageElement, LanguageElement) | Converts short type names to full type names in the target element that can be located in a different type. For example, “String” -> “System.String” if “System” namespace isn’t referenced. Note that it is hidden from Intellisense. |
| SimplifyExpression(Expression) | Simplifies the given expression. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| Active | Gets the currently active refactoring. |
| ActivePicker | Gets the currently active target picker. |
| AvailableRefactorings | Returns a list of all available refactorings in the current context. |
| CanShowSmartTag | Return true if there’s at least one refactoring provider available. |
| IsActive | Returns true if a refactoring is active. |
| IsMenuActive | Returns true if the refactoring menu is active. |
| IsSmartTagActive | Returns true if the smart tag is active. |
| PickerIsActive | Returns true if the current target picker is active. |
| Providers | Returns an array of all registered refactorings. |
| SuppressActionHints | Gets or sets the value that controls the refactoring action hints availability. |

## DXCore Services – Reflection

September 19th, 2011

The **Reflection** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to useful .NET reflection methods.

Here are the methods of this service:

|  |  |
| --- | --- |
| Name | Description |
| CallInheritedStaticMethod (Type, String, object[]) | Calls a static method with the specified arguments inside an ancestor of the specified Type. |
| CallStaticMethod (MemberInfo) | Calls a static method. |
| CallStaticMethod (MemberInfo, object[]) | Calls a static method with the specified arguments. |
| CallStaticMethod(Type, String) | Calls a static method inside the given Type. |
| CallStaticMethod(Type, String, object[]) | Calls a static method inside the given Type with the specified arguments. |
| CallStaticVirtualMethod (Type, String) | Calls a static method inside the given Type (or an ancestor class). |
| CallStaticVirtualMethod (Type, String, object[]) | Calls a static method inside the given Type (or an ancestor class) with the specified arguments. |
| GetDefaultMembers (Object) | Returns an array of default members (which have the DefaultMember attribute set) of the given object. |
| GetDefaultProperty (Object) | Returns the first PropertyInfo instance of the default members (which have the DefaultMember attribute set) of the given object. |
| GetDefaultPropertyInfo (Object, String, Object) | Returns the first PropertyInfo instance of the default members (which have the DefaultMember attribute set) of the value’s object returned by the specified property of the given type. |
| GetFieldInfo(Object, String) | Returns a FieldInfo instance of the specified field from the given object. |
| GetFieldValue(Object, String) | Returns the value of the given field. |
| GetNonPublicFieldInfo (Object, String) | Returns a FieldInfo instance of the specified non-public field from the given object. |
| GetNonPublicFieldValue (Object, String) | Returns the value of the given non-public field. |
| GetNonPublicPropertyInfo (Object, String) | Returns a PropertyInfo instance of the specified non-public property from the given object. |
| GetNonPublicValue (Object, String) | Returns the value of the given non-public property. |
| GetPropertyDescriptor (Object, String) | Returns a PropertyDescriptor instance for the given property inside the specified object. |
| GetPropertyDescriptor (Object, String, Boolean) | Returns a PropertyDescriptor instance for the given property inside the specified object. Optionally specifies whether to ignore case when searching for the specified property name inside the given object. |
| GetPropertyInfo(Object, String) | Returns a PropertyInfo instance of the specified property from the given object. |
| GetPropertyValueList (Object) | Returns a list of properties and values for the given object. This method can be useful to debug objects. |
| GetValue(Object, String) | Returns the value of the given public property in the specified object. |
| GetValue(Object, String, object[]) | Returns the value of the given default indexed property in the specified object. |
| HasBrowsableProperty (Object, String) | Determines whether or not the given object has a specified property that is browsable, as specified in the System.ComponentModel.Browsable attribute. |
| HasProperty(Object, String, String) | Determines whether or not the given object has the specified property. |
| HasProperty(Object, String, String, Boolean) | Determines whether or not the given object has the specified property. Optionally specifies whether to search for a property in descendant types or not. |
| HasProperty(Object, String, String, Boolean, Boolean) | Determines whether or not the given object has the specified property. Optionally specifies whether to search for a property in descendant types and whether a property should be browsable, as specified in the System.ComponentModel.Browsable attribute. |
| InheritsFrom(Type, String) | Returns true if the given type name is inherited from the specified Type. |
| InheritsFrom(Type, Type) | Returns true if the given Type is inherited from the specified Type. |
| SetFieldValue (Object, String, Object) | Sets the value of the specified field in the given object. |
| SetNonPublicFieldValue (Object, String, Object) | Sets the value of the non-public field supported by the given object. |
| SetNonPublicValue (Object, String, Object) | Sets the value of the non-public property supported by the given object. |
| SetProperty(Object, String, Object) | Sets the value for the specified public property. Use this method to change the component properties at design-time. The document will be automatically marked as being modified, and undo will be automatically supported. |
| SetValue(Object, String, Object) | Sets the value for the specified property in the given object. |
| SetValue(Object, String, Object, object[]) | Sets the value for the specified default indexed property in the given object. |
| StringToEnum(Type, String) | Converts the string representation of the given name or numeric value of one or more enumerated constants to an equivalent enumerated object. |
| StringToEnum(Type, String, Boolean) | Converts the string representation of the given name or numeric value of one or more enumerated constants to an equivalent enumerated object. A boolean parameter specifies whether the operation is case-sensitive. |
| TypesMatch(Type, String, Boolean) | Determines whether the specified types are equivalent. Optionally specifies whether descendant types qualify as equivalent. |

## DXCore Services – RegEx (regular expressions)

July 4th, 2011

The **RegEx** [DXcore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the **DXCore** regular expressions and [aliases engine](http://www.skorkin.com/2011/01/ide-tools-terminologykey-concepts/).

**Methods** of the service:

|  |  |
| --- | --- |
| **Method name** | **Description** |
| GetAliases(String) | Returns a collection of aliases for the specified language. |
| GetSystemAliases(String) | Returns a collection of system aliases for the specified language. |
| IsMatch(String, String) | Indicates whether the regular expression finds a match in the input string using the regular expression specified in the pattern parameter. Replaces aliases in the pattern before calling the Regex object. |
| IsMatch(String, String, RegexOptions) | Indicates whether the regular expression finds a match in the input string, using the regular expression specified in the pattern parameter and the matching options supplied in the options parameter. |
| Match(String, String) | Searches the specified input string for an occurrence of the regular expression supplied in the pattern parameter. Replaces aliases in the pattern before calling the Regex object. |
| Match(String, String, RegexOptions) | Searches the input string for an occurrence of the regular expression supplied in a pattern parameter with matching options supplied in an options parameter. Replaces aliases in the pattern before calling the Regex object. |
| Matches(String, String) | Searches the specified input string for all occurrences of the regular expression specified in the pattern parameter. Replaces aliases in the pattern before calling the Regex object. |
| Matches(String, String, RegexOptions) | Searches the specified input string for all occurrences of the regular expression supplied in a pattern parameter with matching options supplied in an options parameter. Replaces aliases in the pattern before calling the Regex object. |
| Replace(String, String, MatchEvaluator) | Replaces all occurrences of a character pattern defined by a regular expression with a replacement character string starting at the first character. A MatchEvaluator delegate is called at each match to evaluate the replacement. Replaces aliases in the pattern before calling the Regex object. |
| Replace(String, String, MatchEvaluator, RegexOptions) | Replaces all occurrences of a character pattern defined by a specified regular expression with a replacement character string starting at the first character. Options can be specified to modify matching behavior and a MatchEvaluator delegate is called at each match to evaluate the replacement. Replaces aliases in the pattern before calling the Regex object. |
| Replace(String, String, String) | Replaces all occurrences of matches defined by the regular expression with a replacement string, starting at the first character in the input string. Replaces aliases in the pattern before calling the Regex object. |
| Replace(String, String, String, RegexOptions) | Replaces all occurrences of a pattern defined by a specified regular expression with a specified replacement character string, starting at the first character in the input string. Options can be specified to modify the matching behavior. Replaces aliases in the pattern before calling the Regex object. |
| ReplaceRegexAliases( String) | Replace aliases in the passed string, using Aliases first, then SystemAliases. Discards unrecognized aliases. |
| ReplaceRegexAliases( String, Boolean) | Replace aliases in the passed string, using Aliases first, then SystemAliases, and optionally throwing an exception if an unrecognized alias is found. |
| ReplaceRegexReplacement( String) | Replaces aliases in the passed string with the matching values from the collection, discarding any unrecognized aliases. |
| Split(String, String) | Splits the input string at the positions defined by a regular expression pattern. Replaces aliases in the pattern before calling the Regex object. |
| Aliases | Gets a collection of language neutral regular expression aliases. |
| AliasesLang | Gets a collection of language specific regular expression aliases. |
| SystemAliases | Gets a collection of language neutral system regular expression aliases. Note: for internal system use only. |
| SystemAliasesLang | Gets a collection of language specific system regular expression aliases. Note: for internal system use only. |

## DXCore Services – Resources

February 19th, 2011

**Resources** [service](http://skorkin.com/2010/09/dxcore-services-list/) of [DXCore](http://skorkin.com/2010/06/devexpress-dxcore-framework/) contains shared resource images and drawing methods for use in plug-ins.

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| DrawSourceModelImage | Draws the image at the specified index onto a Graphics location. Clipping rects are checked and the image is only drawn if the center of the graphic is inside the clipping rect/region. |
| GetSourceModelHighligtImage | Returns the highlighted source model image with the given index. |
| GetSourceModelImage | Returns the source model image with the given index. |
| HighlightSourceModelImage | Highlights the image at the specified index onto a Graphics location. Clipping rects are checked and the image is only drawn if the center of the graphic is inside the clipping rect/region. |
| UserColorImages | Forces plug-ins that draw images to use color source model images. |
| UserMonochromeImages | Forces plug-ins that draw images to use monochrome (blue and white) source model images. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| SourceModelHighlightImages | Returns an ImageList of shared images for representing highlighted elements (e.g., bright images to be painted on a dark background) of the source model. |
| SourceModelImages | Returns an ImageList of shared images for representing elements of the source model. |
| VisualStudio2008Icon | Returns the Visual Studio 2008 icon (16×16). |
| VisualStudio2010Icon | Returns the Visual Studio 2010 icon (16×16). |
| VisualStudioIcon | Returns the Visual Studio icon – 2008 or 2010 – depending on the current Visual Studio version loaded. |

## DXCore Services – Searchers

June 19th, 2011

The **Searchers** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the extension points (also known as **Search Providers**) to the [Rename](http://www.skorkin.com/2010/12/refactorings-rename/) refactoring and [Coderush Navigation features](http://www.skorkin.com/2010/08/coderush-pro-navigation-features/) such as [*Tab to Next Reference*](http://www.skorkin.com/2011/06/coderush-code-navigation-inside-visual-studio-tab-to-next-reference/).

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Get(String) | Returns the search provider by the specified name. |
| GetAllAvailableNavigationSearchers | Returns available navigation search providers in the current context. |
| GetFirstAvailableNavigationSearcher | Returns the first available search provider that can be used for source code navigation. |
| GetFirstAvailableRenamingSearcher | Returns the first available search provider that can be used for renaming. |
| GetProgressVisualizer | Returns an instance of the IProgressVisualizer which is indented to visually indicate the progress of an operation. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| ProgressVisualizerGetter | Gets or sets the ProgressVisualiser handler used for visual progress indication of an operation. |
| Providers | Gets an array of registered search providers. |
| RenamingProviders | Gets an array of search providers specifically used for extending the Rename refactoring |

## DXCore Services – Selection

August 30th, 2011

The **Selection** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for manipulating the selected text in the code editor.

**Methods** of this service:

|  |  |
| --- | --- |
| Method name | Description |
| Collapse | Collapses the current selection if automatic outlining is turned off. Call the CodeRush.Outline.ExpandCurrent() method to restore the collapsed region. |
| Delete | Deletes the selected text. |
| GetPoints(SourcePoint, SourcePoint) | Outputs the anchor and active points of the active selection via the anchorPoint and activePoint parameters. If no text is selected, activePoint will be the active caret position. If no document is open, activePoint and anchorPoint will be set to empty source points. This method returns true if a selection exists; otherwise, false. |
| SelectRange(Int32, Int32, Int32, Int32) | Selects the specified range in the active text document. |
| SelectRange(SourcePoint, SourcePoint) | Selects the specified range in the active text document. |
| SelectRange(SourceRange) | Selects the specified range in the active text document. |
| SelectRange(TextPoint, TextPoint) | Selects the range between the specified text points in the active document. |
| SelectToMarker | Selects the text from the editor caret to the topmost marker. |
| SelectWord | Selects the word at the editor caret. |

**Properties**:

|  |  |
| --- | --- |
| Property name | Description |
| Exists | Returns true if a text is selected in the code editor. |
| Height | Returns the number of lines in the current selection. |
| State | Gets the current selection state. A state is one of the following values: no selection, line fragment, whole line or multiple lines. |
| Text | Sets or gets the text selected in the code editor, as a string. |

## DXCore Services – Serialization

April 22nd, 2011

The **Serialization** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for objects serialization (saving and loading). It can be used to [load CodeRush Code tempalates](http://www.skorkin.com/2011/04/how-to-loadchange-coderush-templates-from-inside-your-dxcore-plug-in/).

**Methods** of this service:

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| LoadFromBinaryFile (String) | Deserializes the stream specified by the full path to the file into an object graph using the binary formatter. |
| LoadFromBinaryFile (String, SerializationBinder) | Deserializes the stream specified by the full path to the file into an object graph using the binary formatter. Takes the SerializationBinderobject instance that controls the binding of a serialized object to a type. |
| LoadFromXmlFile (String) | Deserializes the data on the stream specified by the full path to the file and reconstitutes the gpraph of objects using the SOAPformatter. |
| LoadFromXmlFile (String, SerializationBinder) | Deserializes the data on the stream specified by the full path to the file and reconstitutes the gpraph of objects using the SOAPformatter. Takes the SerializationBinderobject instance that controls the binding of a serialized object to a type. |
| SaveToBinaryFile (Object, String) | Serializes the given object into the stream specified by the full path to the file using the binary formatter. |
| SaveToXMLFile (Object, String) | Serializes the given object into the stream specified by the full path to the file using the SOAP formatter. |
| SmartLoad (String) | Deserializes the data on the stream specified by the relative or full path to the file and reconstitutes the graph of objects using the SOAPformatter first. If it fails, uses the binary formatter. |
| SmartLoad (String, SerializationBinder) | Deserializes the data on the stream specified by the relative or full path to the file and reconstitutes the graph of objects using the SOAPformatter first. If it fails, uses the binary formatter. Takes the SerializationBinderobject instance that controls the binding of a serialized object to a type. |
| SmartSave (Object, String) | Serializes the data on the stream specified by the relative or full path to the file using both SOAPand binary formatters. |

## DXCore Services – SmartTag

October 31st, 2011

The **SmartTag** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the [smart tags and popup menu](http://www.skorkin.com/2010/09/how-to-perform-a-code-refactoring-using-ide-tools/) manipulation APIs.

Here are methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| ClosePopupMenu | Closes the Popup menu. Note that it is hidden from Intellisense. |
| CreateManager(String) | Creates and returns ISmartTagManager of the specified type. |
| Get(String) | Returns a smart tag provider with the specified name. |
| GetBigHintLocation | Gets the best location of the SmartTag big hint. |
| GetPopupMenuPoint | Gets the best point for the popup menu location. |
| GetSmartTagItem(String) | Returns a smart tag menu item with the specified path. |
| HidePopupMenu | Hides the Popup menu. |
| HideSmartTag | Hides the Smart Tag. |
| ShowPopupMenu(Point, SmartTagProviderBase) | Displays a popup menu inside the currently active code editor. |
| ShowPopupMenu(Point, SmartTagProviderBase, Boolean) | Displays a popup menu inside the currently active code editor. |
| ShowPopupMenu(Point, SmartTagProviderBase[]) | Displays a popup menu inside the currently active code editor. |
| ShowPopupMenu(Point, SmartTagProviderBase[], Boolean) | Displays a popup menu inside the currently active code editor. |
| ShowPopupMenu(Point, SmartTagProviderBase[], Boolean, Boolean) | Displays a popup menu inside the currently active code editor. Note that it is hidden from Intellisense. |
| ShowSmartTag | Displays the Smart Tag inside the currently active code editor. |
| ShowSmartTag (SmartTagProviderBase) | Displays the Smart Tag inside the currently active code editor. |
| ShowSmartTag (SmartTagProviderBase[]) | Displays the Smart Tag inside the currently active code editor. |
| UpdateCodeFixContext (SourcePoint) | Updates the [Code Issues](http://www.skorkin.com/2011/02/coderush-code-issues-technology-overview/) Fix context with the given caret source point. |
| UpdateCodeFixContext (SourcePoint, SourceRange) | Updates the Code Issues Fix context with the given caret source point and selection source range. |
| UpdateCodeFixContext (TextView, SourcePoint, SourceRange) | Updates the Code Issues Fix context with the given caret source point, selection source range and a text view. |
| UpdateContext | Updates the SmartTag context. Note that it is hidden from Intellisense. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| AvailableProviders | Returns an array of all available providers. |
| CodeFixContext | Gets the context for the Code Issues code fixes. |
| Context | Gets the context for the Smart Tag. |
| IsPopupMenuVisible | Returns true if Popup menu is visible. |
| IsSmartTagVisible | Returns true if Smart Tag is visible. |
| Providers | Returns an array of all registered providers. |

## DXCore Services – Solution

October 21st, 2010

The **Solution** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods for adding, removing, and renaming project items of the current opened solution.

Here they are:

|  |  |
| --- | --- |
| **Name** | **Description** |
| AddDirectoryToProject | Adds the specified directory to the project with the given name. The project must be open in Visual Studio, and part of the active solution. |
| AddFileToProject | Removes the specified file with the given name from the project. The project must be open in Visual Studio, and part of the active solution. |
| AddProject(String, String) | Adds a new project with the given name of the specified template (e.g. “ClassLibrary.zip”). |
| AddProjectReference(String, String) | Adds the given project reference to the specified project. |
| ExcludeFileFromProject | Removes the specified file with the given name from the project. The project must be open in Visual Studio, and part of the active solution. |
| FindEnvDTEProject(String) | Locates the EnvDTE.Project with the specified name in the active solution. If not found, null is returned. |
| FindEnvDTEProjectByHashCode(Int32) | Locates the EnvDTE.Project with the specified hash code in the active solution. If not found, null is returned. Note that it is hidden from Intellisense. |
| FindProjectItemByName | Returns an instance of the EnvDTE.ProjectItem type by specified name. |
| FindProjectItemByPath | Returns an instance of the EnvDTE.ProjectItem type by specified file path. |
| FindProjectItemByPathEnding | Searches recursively among the items of the project and returns an item with a path, which ends with a given string. |
| ProjectDirectoryIsExist | Checks whether or not the specified directory exists in a project with a given name. The project must be opened in Visual Studio and be a part of the active solution. |
| RemoveFileFromProject | Removes the specified file with the given name from the project, and physically deletes it. The project must be open in Visual Studio, and part of the active solution. |
| RenameFileInProject | Renames the specified file from the project with the given path. The project must be open in Visual Studio and part of the active solution. |

Properties:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Active | Returns an instance of the active opened Solution of type EnvDTE.Solution. |
| AllProjects | Gets enumerator for all available projects. |

**DXCore Services – Source (SourceModel)**

April 7th, 2011

Provides access to the source model services. This [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) contains lots of useful APIs for working with the source code of the entire solution hierarchy.

**Methods** of this service:

|  |  |
| --- | --- |
| AddPrototype(Method, String) | Adds a prototype for the given *Method*[language element](http://www.skorkin.com/2011/01/ide-tools-terminologykey-concepts/) instance, if needed (e.g. in C++ language). |
| AssemblyReferenced(String) | Returns true, if the specified assembly is referenced in the active project. |
| BeginUpdate | This method suspends parsing and is useful if you plan to make several repeated changes to the code. If you call this inside plug-in, you must guarantee a call to *EndUpdate*() (e.g., placed in a finally block), so normal parsing can resume. |
| ChangeVisibility( AccessSpecifiedElement, MemberVisibility) | Changes the visibility of the specified element in the active text document. |
| ChangeVisibility( AccessSpecifiedElement[], MemberVisibility) | Changes the visibility of an array of the elements in the active text document. |
| ContainsAnonymousType(ICollection) | Returns true, if the given collection contains at least one anonymous type or anonymous type reference. |
| ContainsAnonymousType(IElement) | Returns true, if the given element contains at least one anonymous type or anonymous type reference inside its children. |
| CutNamespaceFromFullTypeName( String, String) | Removes namespace name from the full type name. Note that it is hidden from Intellisense. |
| DeclareNamespaceReference(String) | Declares the specified namespace reference (e.g., inside a using statement for C# or an imports statement for VB) if not yet declared. |
| DeclaresLocal(String) | Returns true, if the specified identifier is declared locally inside the active method or property. [Refactor! product](http://www.skorkin.com/2010/06/devexpress-refactor-pro/) must be installed for this method to work. |
| DeclaresLocalOfType(String) | Returns true, if a local variable of the specified type is declared inside the active method or property. [Refactor! product](http://www.skorkin.com/2010/06/devexpress-refactor-pro/) must be installed for this method to work. |
| DeclaresMember(String) | Returns true, if the active type declares a member with the specified name. |
| DescendsFrom(String) | Returns true, if the active class descends from the specified ancestor type. |
| ElementFromLocation(String) | Returns a language element corresponding to the specified location. Note that it is hidden from Intellisense. |
| EndUpdate | Marks the end of an update session. You must call this once for every call to *BeginUpdate*(). |
| EndUpdate(Boolean) | Marks the end of an update session. You must call this once for every call to *BeginUpdate*(). Takes a boolean parameter which specifies whether any changes were made during the update. |
| FindElement( LanguageElementCollection, String) | Finds an element with the given name inside the specified collection of language elements. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| FindElementByFullName( LanguageElementCollection, String) | Performs a recursive search for an element with the given name inside the specified collection of language elements. Note that it is hidden from Intellisense. |
| FindElementByLocation( LanguageElementCollection, String) | Finds an element with the given location inside the specified collection of language elements. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| FindElements( LanguageElementCollection, String) | Finds elements with the given name inside the specified collection of language elements. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| FindMember(LanguageElement, String) | Finds a member with the given name inside the specified language element. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| FindMembers(LanguageElement, String) | Finds members with the given name inside the specified language element. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| FindNamespaceByLocation( LanguageElement, String) | Recursively searches for a namespace with the given location inside the specified language element. Note that it is hidden from Intellisense. |
| FindType(String) | Finds the *ITypeElement* instance by the specified type name. The returned type will be in the scope of the active language element. |
| FindTypeOrNamespace( LanguageElement, String) | Finds a type or namespace with the given name inside the specified language element. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| FindTypeOrNamespace( LanguageElement, String, Boolean) | Recursively searches for a type or namespace with the given name inside the specified language element. Note that it is hidden from Intellisense. |
| FindTypeOrNamespaceByLocation( LanguageElement, String) | Recursively searches for a type or namespace with the given location inside the specified language element. Note that it is hidden from Intellisense. |
| FindTypes(String) | Returns a list of *ITypeElement* instances specified by the given name. |
| GetActiveClassInterfaceOrStruct( LanguageElement) | Gets the class, interface, struct or module that contains the language element. Returns null if the language element is not located inside a class, interface, struct or module. |
| GetActiveMethod( LanguageElement) | Gets the *Method* instance that contains the language element. Returns null if the language element is not located inside a method. |
| GetActiveMethodOrProperty( LanguageElement) | Gets the *Property*instance that contains the language element. Returns null if the language element is not located inside a property. |
| GetActiveMethodOrPropertyAccessor( LanguageElement) | Gets the method or property’s accessor instance that contains the language element. Returns null if the language element is not located inside a method or property’s accessor. |
| GetActiveOrNodeBefore(SourcePoint) | Returns the language element that ends before the specified point. Note that it is hidden from Intellisense. |
| GetActiveOrNodeBeforeCaret | Returns the language element that ends before the editor caret position. Note that it is hidden from Intellisense. |
| GetAjaxAssemblyName | Returns the name of the ASP.NET AJAX Extensions assembly. |
| GetAllBaseTypes(ITypeElement) | Returns all base types for the pointed type declaration. |
| GetAllCodeMetricProviders | Retrieves all available code metric providers. |
| GetAspImportedNamespaces( ProjectElement) | Returns a string collection of imported namespaces for the specified ASP.NET project. Note that it is hidden from Intellisense. |
| GetBaseTypes(ITypeElement) | Returns a list of all base types for the given type declaration instance. |
| GetBlockNecessityAt(SourcePoint) | Gets the necessity of the start and end blocks (e.g. “{” and “}”) after inserting a new text at the specified source point |
| GetBuildAction(SourceFile) | Gets the source file build action. |
| GetCodeActive( LanguageElement, SourcePoint) | Returns the active language element inside an ASP code embedding element. |
| GetCodeMetricProvider(String) | Retrieves a code metric provider by name. |
| GetCodeMetricValue(String, LanguageElement) | Retrieves a code metric value for the given LanguageElement instance using the specified code metric provider. |
| GetComments | Returns a collection of comments inside the given scope. |
| GetCommentsFromNodes | Returns a collection of comments inside the given nodes list. |
| GetCommonParent( LanguageElement, LanguageElement) | Gets the common parent of the two specified nodes. Returns null if the two nodes do not have a common parent. |
| GetCommonParent( LanguageElement, LanguageElement, Boolean) | Gets the common parent of the two specified nodes. Returns null if the two nodes do not have a common parent. If the ‘checkNodes’ boolean parameter is true, the method will check the language element nodes of the passed in first and second elements. If the first node parents the second one, the first node will be returned. If the second node parents the first one, then the first node will be returned. |
| GetDeclaration(IElement) | Returns declaration for the given language element. Language element is restored for the declaration. |
| GetDeclaration(IElement, Boolean) | Returns declaration for the given language element. |
| GetDeclaration(IElement, String) | Returns declaration for the given identifier starting from the given element. Language element is restored for the declaration. |
| GetDeclaration(IElement, String, Boolean) | Returns declaration for the given identifier starting from the given element. |
| GetDeclaration(SourcePoint, String) | Returns declaration for the given identifier starting from the editor caret position. |
| GetDeclarationAtCaret(String) | Returns declaration for the given identifier starting from the editor caret position. |
| GetDefineDirectiveDeclaration( SourceFile, SourcePoint) | Returns a declaration of the define directive at the specified location. Note that it is hidden from Intellisense. |
| GetExpressionInRange( LanguageElement, SourceRange) | Returns an expression inside of the specified language element and restricted within the given source range. |
| GetFileNode(Text document) | Returns the source file language element for the specified text document. |
| GetFirstStatement(LanguageElement) | Returns the first statement inside the given one. |
| GetIdentifierDeclaration(String) | Finds the declaration for the specified identifier. Note that it is hidden from Intellisense. |
| GetIdentifierDeclaration(String, IdentifierList) | Finds the declaration for the specified identifier. Note that it is hidden from Intellisense. |
| GetIdentifierDeclaration(String, LanguageElement, Int32, Int32) | Finds the declaration for the specified identifier. Note that it is hidden from Intellisense. |
| GetIdentifierList( LanguageElement, Int32, Int32) | Returns an IdentifierList suitable for iterating through nodes visible to the specified location. |
| GetLanguageElement(IElement) | Gets the *LanguageElement* instance from the given *IElement*instance. |
| GetMember(ITypeElement, String) | Finds the specified member in the specified type or an ancestor class. Returns null, if not found. |
| GetMember(ITypeElement, String, Boolean) | Finds the specified member in the specified type. Optionally searches inside ancestor classes. Returns null, if not found. |
| GetMember(ITypeElement, String, IElementFilter, Boolean) | Finds and returns the specified member in the specified type. Optionally searches ancestor classes, and also allows you to pass an [*IElementFilter*](http://www.skorkin.com/2011/04/how-to-enumerate-solution-and-source-code-items-using-dxcore/)descendant to filter out possible elements. Returns null, if not found. |
| GetMembers(LanguageElement) | Returns a collection of members defined inside the specified language element. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| GetMembers(LanguageElement, Boolean) | Returns a collection of members defined inside the specified language element. Note that it is hidden from Intellisense. |
| GetMethod(ITypeElement, String) | Finds the specified method in the specified type or inside an ancestor class. Returns null if not found. |
| GetMethod(ITypeElement, String, Boolean) | Finds the specified method in the specified type. Optionally searches ancestor classes. Returns null if not found. |
| GetNamespaces(LanguageElement) | Returns a collection of namespaces defined inside the specified language element. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| GetNamespaces(LanguageElement, Boolean) | Returns a collection of namespaces defined inside the specified language element. Note that it is hidden from Intellisense. |
| GetNamespaces(String) | Returns the collection of all namespace declarations specified by the given name. Note that it is hidden from Intellisense. |
| GetNearestParentingStatement( SourcePoint) | Returns the first *PartingStatement* instance that parents the element at the given source point. |
| GetNextStatementToExecute( LanguageElement) | Returns the next statement that will be executed. Returns null if the control flow will leave current method or property accessor. |
| GetNodeAt(EditPoint) | Returns the node at the specified edit point. |
| GetNodeAt(Int32, Int32) | Returns the node at the specified line and column position in the active document. |
| GetNodeAt(LanguageElement, Int32, Int32) | Returns the node at the specified line and column position in the specified file node. |
| GetNodeAt(SourcePoint) | Returns the node at the specified *SourcePoint*. |
| GetNodeAt(TextPoint) | Returns the node at the specified *TextPoint*. |
| GetNodeBefore(SourcePoint) | Gets the node that ends prior to the specified SourcePoint in the active text document. |
| GetNodeBefore(SourcePoint, LanguageElement) | Gets the node that ends prior to the specified SourcePoint in the active text document. |
| GetNonBorderingParent(SourceRange) | Gets a parent of the specified SourceRange that does not immediately border the *SourceRange’s*Start or End points. There must be at least one character between the parent’s boundaries and the specified *SourceRange*. |
| GetParentsThatAreSiblings( LanguageElement, LanguageElement, LanguageElement, LanguageElement) | Gets the parent nodes to the specified nodes that are siblings to each other. This method is useful when you need to select a range of code that spans two sibling nodes, and you have two non-sibling nodes that must be included in that selection. |
| GetProjectConfigurationName( ProjectElement) | Returns the name of the project configuration. |
| GetProperty(ITypeElement, String) | Finds a property in the specified type or inside an ancestor class. Returns null, if not found. |
| GetProperty(ITypeElement, String, Boolean) | Finds a property in the specified type. Optionally searches inside ancestor classes. Returns null, if not found. |
| GetReferencedDeclarationInRange( LanguageElement, SourceRange) | Returns the reference to the declaration mapping for all elements inside the specified scope at the given source range. |
| GetRegionAt(Int32, Int32) | Returns the region directive at the specified line and column position in the active Text document. |
| GetRegionAt(SourcePoint) | Returns the region directive at the specified SourcePoint in the active text document. |
| GetReturnTypeName | Returns the name of the type of the active method or property. |
| GetSelectedExpression( TextDocument, TextView, SourceRange) | Returns the currently selected expression. If there is no expression selected, then this property returns null. |
| GetSelectedNodes | Retrieves an array, containing the currently selected nodes of the active text document’s parse tree. |
| GetStartWordRange( LanguageElement) | Returns the source range of the language element’s starting word. |
| GetStorage(ProjectElement, String) | Returns the project specific decoupled storage for the given options page. |
| GetStorage(ProjectElement, String, String) | Returns the project specific decoupled storage for the given options page. |
| GetStorage(SolutionElement, String) | Returns the solution specific decoupled storage for the given options page. |
| GetStorage(SolutionElement, String, String) | Returns the solution specific decoupled storage for the given options page. |
| GetStringAt(EditPoint) | Returns the TextString at the specified EditPoint. |
| GetStringAt(Int32, Int32) | Returns the TextString at the specified line and column position. |
| GetStringAt(SourcePoint) | Returns the TextString at the specified SourcePoint. |
| GetStrings(LanguageElement) | Returns a collection of strings inside the language element. |
| GetSupportedCodeMetricProviders( String) | Retrieves all code metric providers that support the specified language. |
| GetTypeName(LanguageElement) | Returns the type (as a string) of the specified language element. |
| GetTypeName(String) | Returns the type (as a string) of the specified identifier. |
| GetTypes(LanguageElement) | Returns a collection of types defined inside the specified language element. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| GetTypes(LanguageElement, Boolean) | Returns a collection of types defined inside the specified language element. Note that it is hidden from Intellisense. |
| GetTypesOrNamespaces( LanguageElement) | Returns a collection of types or namespaces defined inside the specified language element. Only first level nodes are checked. Note that it is hidden from Intellisense. |
| GetTypesOrNamespaces( LanguageElement, Boolean) | Returns a collection of types or namespaces defined inside the specified language element. Note that it is hidden from Intellisense. |
| GetWordRangeAt(SourcePoint) | Returns the source range of the word at the given location. |
| GetXmlDocComments | Returns a collection of XML doc comments inside the given scope. |
| GetXmlDocCommentsFromNodes | Returns a collection of XML doc comments inside the given collection of nodes. |
| HasMember(String) | Returns true, if the active type or an ancestor class declares a member with the specified name. |
| HasMethod(String) | Returns true, if the active type or an ancestor class declares a method with the specified name. |
| HasProperty(IElement, String) | Returns true, if the specified IElement instance is a type that declares the specified property. |
| HasProperty(String, String) | Returns true, if the specified identifier (whose declaration is visible from the editor caret position) has the specified property. |
| HasStaticInitializer(String) | Returns true, if the specified type is a value type and has a static property of the same type (e.g., Empty, MinValue, etc.). |
| ImplementInterface( TypeDeclaration, Interface) | Implements the interface for the given type. |
| Implements(ITypeElement, String) | Returns true if the specified type implements the specified interface. |
| Implements(String, String) | Returns true if the specified type implements the specified interface. |
| IncludeDirectiveDeclared(String) | Returns true if the ‘include’ directive with the given name is declared. |
| InheritType(TypeDeclaration, TypeDeclaration) | Inherits the specified type for the given type (makes given type a descendant of the specified type). |
| InMacroCall(LanguageElement) | Returns true if the language element is located inside a macro call (C++). |
| Is(ITypeElement, String) | Returns true if the specified type is or descends from the specified ancestor class. |
| Is(String, String) | Returns true if the specified type is or descends from the specified ancestor class. |
| IsAnonymousType(IElement) | Returns true if the given element is an anonymous type or an anonymous type reference. |
| IsLocal(LanguageElement) | Returns true if the specified language element is a local variable. |
| IsStatement(LanguageElement) | Returns true if the given element is a statement and can be executed. |
| IsStatic(LanguageElement) | Returns true if the specified language element is a static method or property. |
| IsType(LanguageElement) | Returns true if the specified language element is a class, struct, or interface. |
| IsXmlDocNode(LanguageElement) | Returns true if the given element is an xml doc node (xml element, xml attribute or xml text). |
| IterateNodesInRange | Iterates through the nodes in the specified range. Parent nodes that overlap the range will be included in the iteration. |
| IterateVisibleNodesInRange | Iterates through the visible nodes in the specified range. Parent nodes that overlap the range will be included in the iteration, however, nodes contained inside collapsed regions will not. |
| Parse | Parses the active text document. Plug-in authors should never need to call this method, as parsing is handled automatically by [DXCore](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/). Note that it is hidden from Intellisense. |
| ParseIfNeeded | Parses the active text document if changes have been committed since the last parse of the active file. This is a low-level method that normally should not need to be called by plug-in writers. |
| ParseIfNeeded(TextDocument) | Parses the active text document if changes have been committed since the last parse of the active file. This is a low-level method that normally should not need to be called by plug-in writers. |
| ParseIfTextChanged | Parses the active 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. |
| ParseIfTextChanged(TextDocument) | Parses the specified text 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. |
| PromoteToSiblings( LanguageElement, LanguageElement) | Converts the first and the second passed in, non-sibling elements, to their closest parents which are siblings. Returns true, if successful, or false if the first element and the second element do not share a common parent. Note that if one of these elements parents the other, then both of these elements will point to the same parent. If the specified elements are already siblings, they will leave this method unchanged. If one element is a detail node and the other is a child node, each will be set to the common parent. |
| ReferenceDeclared(String) | Determines whether the specified namespace reference is declared (e.g., a using reference in C#, or an imports statement in VB). |
| RegionNodeIsValid( LanguageElement) | Returns true, if the specified node is a valid region node. This method is useful for iterating through parent nodes of regions. The topmost region node is used simply as a container for all other regions in the file, and if that node is passed, this method will return false. |
| ReplacePrototype(Method, String) | Replaces the prototype of the given method to the new one. |
| ResetSourceUpdate | Resets the internal source update counter, so parsing can be normally resumed. |
| StatementCount(LanguageElement) | Returns the number of statements inside the given element. |
| StripNamespace | Gets the simple type (as a string) of the specified type element. |

**Properties**:

|  |  |
| --- | --- |
| Active | Gets the language element at the editor caret. |
| ActiveAnonymousExpression | Gets the anonymous method expression, containing the editor caret. Returns null if the editor caret is not inside an *AnonymousMethodExpression*or *LambdaExpression*element instance. |
| ActiveClass | Gets the class, containing the editor caret. Returns null if the editor caret is not inside a class. |
| ActiveClassInterfaceOrStruct | Gets the class, interface, or struct that contains the editor caret. Returns null if the editor caret is not located inside a class, interface, or struct. |
| ActiveClassInterfaceStructOrModule | Gets the class, interface, struct or module that contains the editor caret. Returns null if the editor caret is not located inside a class, interface, struct or module. |
| ActiveClassName | Gets the name of the active class. Note that if the editor caret is inside a nested class (e.g., a class inside a class), only the name of the innermost class is returned. Returns an empty string, if the editor caret is not inside a class. |
| ActiveComment | Gets the *Comment*or *XMLDocComment*instance that contains the editor caret. |
| ActiveDeclaration | Gets the variable, initialized variable, implicit variable or constant declaration that contains the editor caret. Returns null if the editor caret is not contained inside a declaration. |
| ActiveDefineDirective | Gets the *DefineDirective* instance at the editor caret. Returns null if the editor caret is not contained inside a ‘define’ directive. |
| ActiveEvent | Gets the *Event*instance containing the editor caret. Returns null if the editor caret is not inside an event. |
| ActiveEventAccessor | Gets the *EventAccessor*instance containing the editor caret. Returns null if the editor caret is not inside an event add or event remove accessor. |
| ActiveFileNode | Gets the *SourceFile*instance that represents the active text document. |
| ActiveFileNodeWithoutParse | Gets the *SourceFile* instance that represents the active text document without parsing it before returning. |
| ActiveInterface | Gets the Interface containing the editor caret. Returns null if the editor caret is not inside an interface. |
| ActiveMacroInfo | Gets the macro info from the editor caret location. Returns null if the editor caret is not contained inside a macro reference. |
| ActiveMember | Gets the active *Method*, *Property*, *Event*, *Field*, etc., that is immediately parented by a *Class*, *Struct*, or *Interface*. |
| ActiveMemberName | Gets the name of the active member (*Method*, *Property*, *Event*, *Field*, etc.), that is immediately parented by a *Class*, *Struct*, or *Interface*. |
| ActiveMethod | Gets the *Method*containing the editor caret. Returns null if the editor caret is not inside a method. |
| ActiveMethodName | Returns the name of the active method. Returns an empty string if the editor caret is not inside a method. |
| ActiveMethodOrProperty | Gets the active method or property at the editor caret. Returns null if the editor caret is not contained inside a method or property. |
| ActiveMethodOrPropertyAccessor | Gets the active method, property get/set accessor or event add/remove block at the editor caret. Returns null if the editor caret is not contained inside a method or property accessor. |
| ActiveMethodOrPropertyAccessorName | Returns the name of the method or property accessor containing the editor caret. |
| ActiveMethodPropertyOrEvent | Gets the active method, property or event at the editor caret. Returns null if the editor caret is not contained inside a method, property or event. |
| ActiveNamespace | Gets the active namespace language element. |
| ActiveNamespaceName | Gets the name of the active namespace. Note that if the editor caret is inside an nested namespace (e.g., a namespace inside a namespace), only the name of the innermost namespace is returned. Returns an empty string if the editor caret is not inside a namespace. |
| ActiveProject | Gets the *ProjectElement*instance containing the active text document. |
| ActiveProjectWithoutParse | Gets the *ProjectElement*instance containing the active text document without parsing it before returning. |
| ActiveProperty | Gets the *Property* containing the editor caret. Returns null if the editor caret is not inside a property. |
| ActivePropertyAccessor | Gets the *PropertyAccessor*containing the editor caret. Returns null if the editor caret is not inside a property’s get or set accessor. |
| ActiveRefType | Gets the *RefType*instance (C++) containing the editor caret. Returns null if the editor caret is not inside a reference type. |
| ActiveRegion | Gets the region that contains the editor caret. |
| ActiveSolution | Gets the active Solution instance. |
| ActiveSourceFile | Gets the *SourceFile*instance that represents the active text document. |
| ActiveString | Gets the *TextString*instance containing the editor caret. Returns null if the editor caret is not inside a quoted text string. |
| ActiveStruct | Gets the *Struct*instance containing the editor caret. Returns null if the editor caret is not inside a struct. |
| ActiveType | Gets the class, interface, struct, enum, module, or delegate that contains the editor caret. Returns null if the editor caret is not contained inside a class, interface, struct, enum, module, or delegate. |
| ActiveTypeName | Gets the name of the active class, struct or interface. Note that if the editor caret is inside a nested type (e.g., a type inside a type), only the name of the innermost type is returned. Returns an empty string if the editor caret is outside a type. |
| ActiveValueType | Gets the *ValueType*instance containing the editor caret. Returns null if the editor caret is not inside a value type. |
| ActiveXMLDocComment | Returns the *XMLDocComment*instance for the active node (or for the first parent of the active node that an XML documentation comment can be bound to). Note, this method can return*XMLDocComment*instance that does not contain the editor caret. |
| Aliases | Returns the reference aliases for the current file, or null if the current language does not support using/import aliases. |
| AllNamespaces | Gets an array of all namespaces. Note that it is hidden from Intellisense. |
| CodeActive | Gets the active LanguageElement for the active IAspSourceFile instance. |
| CompilerDirectiveRootNode | Gets the compiler directive root node for the current file, or null, if the current language does not support compiler directives. The root node’s children (accessed through its Nodes property) are the CompilerDirectives for this file. This root node itself, however, is not an actual CompilerDirective for this file (even though its type is CompilerDirective, it does not represent an actual CompilerDirective – it is merely the parent of the CompilerDirectives in this file). |
| CompilerDirectives | Gets the compiler directives for the current file, or null if the current language does not support compiler directives. |
| FirstSelectedNode | Gets the first selected node. |
| Identifiers | Gets an *IEnumerable*member suitable for use in a ‘foreach’ statement. Identifier iterates through identifiers visible to the current node. |
| IncludeDirectives | Returns the list of the ‘include’ directives from the active source file. |
| InsideAttribute | Returns true, if the editor caret is inside an attribute or attribute section. |
| InsideClassInterfaceOrStruct | Returns true, if the editor caret is inside a class, interface, or struct. |
| InsideClassInterfaceStructOrModule | Returns true, if the editor caret is inside a class, interface, struct or module. |
| InsideComment | Returns true, if the editor caret is inside a comment or xml doc comment. This property reflects the current state as determined by the parse tree, but it may not match recent edits on the current line. If you need to know if immediate recent changes on the line have placed the editor caret inside a comment, use*CodeRush.Caret.InsideComment* call. |
| InsideInterface | Returns true, if the editor caret is inside an interface, interface class or interface struct. |
| InsideMethodOrPropertyAccessor | Returns true, if the editor caret is inside a method, property get/set accessor, or event add/remove block. |
| InsideString | Returns true, if the editor caret is inside a string. This property reflects the current state as determined by the parse tree, but it may not match recent edits on the current line. If you need to know if immediate recent changes on the line have placed the editor caret inside a string, use *CodeRush.Caret.InsideString* call. |
| InsideType | Returns true, if the editor caret is inside a class, interface, struct, enum, or delegate. |
| InsideXMLDocComment | Returns true, if caret is inside xml doc comment. |
| IsAjaxInstalled | Returns true, if Ajax is installed. |
| IsUpdating | If true, the source model cache will report that parsing is unnecessary. |
| LastSelectedNode | Gets the last selected node. |
| MiscFiles | Returns the list of opened miscellaneous files that do not belong to the current project. |
| NamespaceReferences | Returns the using/import references for the current file, or null, if the current language does not support using/import references. |
| RegionRootNode | Returns the region root node for the current file, or null, if the current language does not support regions. The root node’s children (accessed through its Nodes property) are the RegionDirectives for this file. This root node itself, however, is not an actual RegionDirectives for this file (even though its type is RegionDirectives, it does not represent an actual RegionDirectives – it is merely the parent of the regions in this file). |
| Regions | Returns the regions list for the current file, or null, if the current language does not support regions. |
| SelectedExpression | Returns the currently selected expression. If there is no expression selected, then this property returns null. |
| Strings | Returns the strings for the current file, or null, if the current language does not support strings. |

## DXCore Services – String Utilities

September 19th, 2011

The **String Utilities** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) (accessed via the [CodeRush.StrUtil](http://www.skorkin.com/2010/11/coderush-object-for-accessing-to-dxcore-services/) object) provides utility methods for manipulating single string, multi-line texts, and string arrays.

Here are the methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| AddExpressionTagsToText (String) | Adds expression tags to the given string. You can use the RemoveExpressionTags method to delete expression tags from the text. |
| AddLineTerminatorIfNecessery (String) | Verifies if the given string ends with a line terminator, and then adds it to the end of the string if it does not exist. |
| AddLinkTagsToText(String) | Adds link tags to the given string. You can use the RemoveLinkTags method to delete link tags from the text. |
| AddMultiLinkTagsToText (String, string[]) | Adds multi-link tags to the given string. You can use the RemoveMultiLinkTags method to delete multi-link tags from the text. |
| AddNameTagsToText(String) | Adds name tags to the given string. You can use the RemoveNameTags method to delete name tags from the text. |
| AddQuotes(String) | Adds double quotes (“) to the given string. |
| AddTagsToText(String) | Adds default tags to the given string. You may use the RemoveTextTags(ref string) method to delete tags from the text. |
| AddTagsToText(String, String, String) | Adds specified tags to the text. You can use the RemoveTextTags method to delete tags from the text. |
| AddTypeTagsToText(String) | Adds type tags to the given string. You can use the RemoveTypeTags method to delete name tags from the text. |
| BuildCSV(string[]) | Creates a comma-separated string from the given string array. |
| BuildParamString(string[]) | Creates a parameter with a comma-separated string from the given string array. The result is a comma-separated string wrapped by parens. |
| CharIn(Char, Char, Char) | Verifies if the given character is in the specified range. |
| CharIn(Char, char[]) | Verifies if the given character is in the specified array of characters. |
| CharIn(Char, String) | Verifies if the given character is in the specified string. |
| CloneStringCollection (StringCollection) | Creates a copy of a string collection instance. |
| CollectionFromString(String) | Creates a string collection from the given string separated by line terminators. |
| ConvertToCSV(string[]) | Converts the given string array to a comma-separated string. |
| ConvertToCSV(string[], Boolean) | Converts the given string array to a comma-separated string. If the quoteCommas parameter is true, any values in the given string array containing commas are added inside double-quotes. |
| CountLines(String) | Returns the number of lines in the given text. (Note: an empty string counts as one line). This function uses Regex to find line terminators. |
| CountLines(String, LineTerminator) | Returns the number of lines in the given text using the specified line terminator. (Note: an empty string counts as one line). |
| DecodeName(String) | Converts the special character-codes to the original characters (‘\’, ‘/’, ‘:’, ‘\*’, ‘?’, ‘”‘, ‘<‘, ‘>’, ‘|’, ‘\_’). Use with the EncodeName method. |
| DecodeText(String) | Converts the special line break tags ($LineBreak$) into a cariage-return/line-break characters (\r\n). Used internally. |
| DuplicateArray(string[]) | Creates a copy of the given string array. |
| EncodeName(String) | Converts the special characters (‘\’, ‘/’, ‘:’, ‘\*’, ‘?’, ‘”‘, ‘<‘, ‘>’, ‘|’, ‘\_’) to the special character-codes. Use with the DecodeName method. |
| EncodeText(String) | Converts different types of a line terminators to the special line break tag ($LineBreak  $). Used internally. |
| EndsWithLineTerminator (String) | Verifies if the given string ends with a line terminator. |
| ExtractStartingWord(String) | Extracts the first word from the given string. |
| FillArray(String, Int32) | Creates an array of strings from the specified text. |
| FirstChar(String) | Returns the first character of the given text, if any. |
| FirstChar(String, Boolean) | Returns the first character of the given text, if any. Optionally, throws an exception if the given string is null or empty. |
| FormatForInsertText(String) | Converts a string with carriage-return/line-feeds to line feeds only, preventing the EditPoint’s InsertText method from doubling-up the inserted lines. Note that it is hidden from Intellisense. |
| FormatForTextBox(String) | Converts a string with line feeds to a string with carriage-return/line feeds. Note that it is hidden from Intellisense. |
| GetClosingWhiteSpace(String) | Returns the closing white space (tabs and spaces) for the specified text. |
| GetClosingWhiteSpace(String, Int32) | Returns the closing white space characters (tabs and spaces) on the given text. |
| GetClosingWhiteSpaceCharCount (String) | Returns the number of closing white space characters (tabs and spaces) on the given text. |
| GetFirstLineTerminatorIndex (String) | Returns the index of the first line terminator in the given string or -1 if the given string does not have a line terminator. |
| GetLastLineTerminator(String) | Returns the last line terminator in the given string or empty TagText if the given string does not end with a line terminator. |
| GetLastLineTerminatorIndex (String) | Returns the index of the last line terminator in the given string or -1 if the given string does not end with the line terminator. |
| GetLeadingWhiteSpace(String) | Returns the leading white space (tabs and spaces) for the given text. |
| GetLeadingWhiteSpace(String, Int32) | Returns the leading white space characters (tabs and spaces) on the given text. |
| GetLeadingWhiteSpaceCharCount (String) | Returns the number of the leading white space characters (tabs and spaces) on the given text. |
| GetLineEndIndex(ArrayList, String, Int32) | Returns the end index of the specified line number in the given text separated by the specified line terminators passed as an array list. Note that it is hidden from Intellisense. |
| GetLineEndIndex(String, Int32) | Returns the end index of the specified line number in the given text. |
| GetLineEndIndex(String, Int32, LineTerminator) | Returns the end index of the specified line number in the given text using the specified line terminator. |
| GetLineEndPoint(String, Int32) | Returns the source point indicating the end of the given string. |
| GetLineEndPoint(String, Int32, LineTerminator) | Returns the source point indicating the end of the given text ending with the specified line terminator. |
| GetLineLength(String, Int32) | Returns the length of the specified line in the given text. |
| GetLineLength(String, Int32, LineTerminator) | Returns the length of the specified line in the given text separated by the specified line terminator. |
| GetLineStartIndex(String, Int32) | Returns the starting index of the specified line in the given text. |
| GetLineStartIndex(String, Int32, LineTerminator) | Returns the starting index of the specified line in the given text using the passed-in line terminator. |
| GetLineTerminator(String) | Returns the line terminator used in the given text. |
| GetLineTerminator(String, Int32) | Returns the line terminator used at the specified index in the given text. |
| GetLineTerminatorChar(Char) | Returns a line terminator for the given character. |
| GetLineTerminatorChar (LineTerminatorChar) | Returns a character for the given line terminator. |
| GetLineTerminatorLength (LineTerminator) | Returns the length (in characters) of the given line terminator. |
| GetLineTerminatorRegex | Returns the regular expression that matches all line terminators. |
| GetLineTerminators(String) | Return the list of line terminators used in the given text. |
| GetLineTerminators(String, Int32) | Return the list of line terminators used in the given text starting from the specified index. |
| GetLineTerminatorText (LineTerminator) | Returns the string that represents the given line terminator. |
| GetLineText(String, Int32) | Returns the line of text from the given text at the specified line index. |
| GetLineText(String, Int32, LineTerminator) | Returns the line of text from the given text separated by the specified line terminator at the specified line index. |
| GetParameters(String) | Returns the text between open and close parens, and removes the parameter block from the command. For example, given the command of “Say(Hello)”, this method returns “Hello” string and adjusts the command, so it becomes “Say”. |
| GetParametersFromText(String) | Returns the text of parameters (inside parens). |
| GetSharedStringBuilder | Returns a SharedStringBuilder from an internal pool. |
| GetSubstring(String, Int32, Int32) | Returns the substring for the given text, using the given start and end indices. Returns String.Empty if the passed arguments are invalid. |
| GetSubstring(String, Int32, Int32, Boolean) | Returns the substring for the given text, using the given start and end indices. Returns String.Empty if the passed arguments are invalid. The throwOnError parameter determines if an exception should be thrown in case invalid arguments. |
| GetText(ArrayList, String, Int32, Int32) | Returns the text separated by the specified line terminators at the given start and end lines. |
| GetText(String, Int32, Int32) | Returns the text at the given start and end lines. |
| GetText(String, Int32, Int32, Int32, Int32) | Returns the text at the given coordinates (start line, start offset, end line, end offset). |
| GetText(String, SourcePoint, SourcePoint) | Returns the text at the given coordinates (start source point, end source point). |
| GetText(String, SourceRange) | Returns the text at the given source range coordinates. |
| GetTextLength(String, Int32, Int32, Int32, Int32) | Returns the length of text at the given coordinates (start line, start offset, end line, end offset). |
| GetTextLength(String, SourcePoint, SourcePoint) | Returns the length of text at the given coordinates (start source point, end source point). |
| GetTextTags(String, String, String) | Returns the tags from the given text. This method allows you to get the text that is marked with the tags using the AddTagsToText method. The method does not remove the tags from the given text. |
| GetUniqueName(String, IList) | Returns the string based on the passed base name which is unique in the given list. The returned string is the passed base name with a trailing number, starting with 1 and incrementing until the return value is unique in the given list. |
| GetWhiteSpaceCount(String) | Return the count of all white space characters in the given string. |
| GuessLineTerminator(String) | Tries to guess what line terminator is used in the given text. If there are mixed line terminators then this function returns only the first found line terminator. |
| InsideParens(String, String) | Returns true if the given leftText contains more open parens than close parens, or if the given rightText contains more close parens than open parens. |
| IsEndLineTerminatorNeeded (String, SourcePoint) | Returns true if an end line terminator is required to be added to the text that is going to be inserted at the specified source point. |
| IsEndLineTerminatorNeeded (TextDocument, SourcePoint) | Returns true if an end line terminator is required to be added to the text that is going to be inserted at the specified source point in the specified text document. |
| IsNullOrEmpty(String) | Determines if the given string is null or an empty string, containing no text. |
| IsNumber(String) | Determines if the given string represents an integer. |
| IsStartLineTerminatorNeeded (String, SourcePoint) | Returns true if a start line terminator is required to be added to the text that is going to be inserted at the specified source point. |
| IsStartLineTerminatorNeeded (TextDocument, SourcePoint) | Returns true if a start line terminator is required to be added to the text that is going to be inserted at the specified source point in the specified text document. |
| JoinLines(string[]) | Combines the given string array into a single string separated by an MS-DOS (carriage-return\line-feed) line separator. A line terminator will not be added to the resulting string. |
| JoinLines(string[], Boolean) | Combines the given string array into a single string separated by an MS-DOS (carriage-return\line-feed) line separator. A boolean parameter specifies whether a line terminator should be added at the end of the resulting string. |
| JoinLines(string[], LineTerminator) | Combines the given string array into a single string separated by the specified line terminator. A line terminator will not be added to the resulting string. |
| JoinLines(string[], LineTerminator, Boolean) | Combines the given string array into a single string separated by the specified line terminator. A boolean parameter specifies whether a line terminator should be added at the end of the resulting string. |
| JoinLines(StringCollection) | Combines the given string collection into a single string separated by an MS-DOS (carriage-return\line-feed) line separator. A line terminator will not be added to the resulting string. |
| JoinLines(StringCollection, Boolean) | Combines the given string collection into a single string separated by an MS-DOS (carriage-return\line-feed) line separator. A boolean parameter specifies whether a line terminator should be added at the end of the resulting string. |
| JoinLines(StringCollection, LineTerminator) | Combines the given string collection into a single string separated by the specified line terminator. A line terminator will not be added to the resulting string. |
| JoinLines(StringCollection, LineTerminator, Boolean) | Combines the given string collection into a single string separated by the specified line terminator. A boolean parameter specifies whether a line terminator should be added at the end of the resulting string. |
| LastChar(String) | Returns the last character from the given string. |
| LastChar(String, Boolean) | Returns the last character from the given string. Optionally throws an exception if the given string is null or empty. |
| LeftString(String, Int32) | Returns a part of the given string of the specified length from the left. |
| LeftString(String, Int32, Boolean) | Returns a part of the given string of the specified length from the left. Optionally throws an exception if the given string is null or empty, or an incorrect count is specified. |
| MustEndWith(String, String) | Ensures that the given text ends with the specified suffix. |
| MustStartWith(String, String) | Ensures that the given text starts with the specified suffix. |
| OffsetToTabColumn(String, Int32, Int32) | Calculates the column position in the given text for the specified character offset. |
| ParenScore(String) | Returns a negative number if there are more open parens than close parens. Returns a positive number if there are more close parens than open parens. |
| ParseCSV(String) | Parses the given comma separated string into a string array. |
| ParseCSV(String, Boolean) | Parses the given comma separated string into a string array. Optionally handles text command quotes (“” and “”) as well as regular double quotes. |
| PasteText(String, Int32, Int32, String) | Inserts the specified string into the given text at the specified line and offset. |
| PasteText(String, Int32, String) | Inserts the specified string into the given text at the specified index. |
| PasteText(String, SourcePoint, String) | Inserts the specified string into the given text at the specified source point. |
| PasteText(String, String) | Inserts the specified string before the given text. |
| RangeOf(String, Int32, String) | Returns the source range of the specified string in the given text starting from the specified index. |
| RangeOf(String, SourcePoint, String) | Returns the source range of the specified string in the given text at the specified source point. |
| RemoveExpressionTags(String) | Removes the expression tags from the given text that were added by the AddExpressionTagsToText method. |
| RemoveLastLineTerminator (String) | Removes the last line terminator from the given string. |
| RemoveLeadingDigits(String) | Removes any digits from the start of the given string. |
| RemoveLeadingLineTerminator (String) | Removes the leading line terminator from the given string. |
| RemoveLinkTags(String) | Removes the link tags from the given text that were added by the AddLinkTagsToText method. |
| RemoveMultiLinkTags(String) | Removes the multi-link tags from the given text that were added by the AddMultiLinkTagsToText method. |
| RemoveNameTags(String) | Removes the name tags from the given text that were added by the AddNameTagsToText method. |
| RemoveQuotes(String) | Remove the double quotes (“) from the given string. |
| RemoveTagsForPreview(String) | Removes all redundant text tags and text commands from the given text. |
| RemoveText(String, Int32, Int32) | Removes the text at the specified start and end index. |
| RemoveText(String, Int32, Int32, Int32) | Removes the text starting from the specified line and offset of the specified length. |
| RemoveText(String, SourcePoint, SourcePoint) | Removes the text at the specified start and end source point coordinates. |
| RemoveTextTags(String) | Removes default tags from the given text. This method allows deleting tags that were added by the AddTagsToText method. |
| RemoveTextTags(String, String, String) | Removes the specified tags from the given text. This method allows deleting tags that were added by the AddTagsToText method. |
| RemoveTextTags(String, String, String, Boolean) | Removes the specified tags from the given text. This method allows deleting tags that were added by the AddTagsToText method. Optionally checks for parameters inside parens. |
| RemoveTrailingDigits(String) | Removes digits from the end of the given string. |
| RemoveTrailingLineTerminator (String) | Removes the trailing line terminator from the given string. |
| RemoveTypeTags(String) | Removes the type tags from the given text that were added by the AddTypeTagsToText method. |
| ReplaceInnerText(String, String, String, ReplaceTextDelegate) | Searches for all occurrences of the given text which starts with the specified start substring and ends with the specified end substring, then replaces each found occurrence using the text replacer delegate. |
| RightString(String, Int32) | Returns a part of the given string of the specified length from the right. |
| RightString(String, Int32, Boolean) | Returns a part of the given string of the specified length from the right. Optionally throws an exception if the given string is null or empty, or a wrong count is specified. |
| SeparateParams(String, String) | Splits the given string into a command and its parameters. For example, given the string of “Say(Hello)”, this method returns the “Hello” string as an out parameter and adjusts the command, so it becomes “Say”. |
| SeparateParams(String, string[]) | Splits the given string into a command and its parameters. Parameters are returned as a string array if they are separated by commas. |
| SeparateValues(string[], String) | Converts the given string array into a single string. Strings from the given array are separated by the specified separator. |
| SeparateValues(string[], String, Boolean) | Converts the given string array into a single string. Strings from the given array are separated by the specified separator. Optionally adds the double quotes around the items that contain the separator. |
| SeparateValues(StringCollection, String) | Converts the given string collection into a string. Strings from the given collection are separated by the specified separator. |
| SeparateValues(StringCollection, String, Boolean) | Converts the given string collection into a string. Strings from the given collection are separated by the specified separator. Optionally adds the double quotes around the items that contain the separator. |
| SetText(String, SourceRange[], String) | Changes the text inside the given string within the specified source ranges. |
| SetText(String, SourceRange[], String, SourceRange[]) | Changes the text inside the given string within the specified source ranges. |
| SetText(String, SourceRange[], string[]) | Changes the text inside the given string within the specified source ranges. |
| SetText(String, SourceRange[], string[], SourceRange[]) | Changes the text inside the given string within the specified source ranges. |
| SetText(String, SourceRangeCollection, String) | Changes the text inside the given string within the specified source ranges. |
| SetVariables(String) | Calls the Set string provider once for each comma-delimited assignment in the form of “variable=value” inside the specified string. White space, if specified, will also be assigned to the variable. |
| ShowWhitespace(String, Int32) | Converts the whitespace characters in the given string into the characters that represent whitespace. |
| ShowWhitespace(string[], Int32) | Converts the whitespace characters in the given string array into the characters that represent whitespace. |
| SplitCodeLine(String, Int32, String, String) | Splits the given code into the left and right segments at the specified character offset. |
| SplitLines(String) | Splits the given string containing line terminators into a string array. If, after splitting, the last entry is empty, it is removed from the result. |
| SplitLines(String, Boolean) | Splits the given string containing line terminators into a string array. Optionally removes the last entry from the result if it is empty. |
| StartsWithLineTerminator(String) | Verifies if the given string starts with a line terminator. |
| SubstringToLower(String, Int32, Int32) | Converts the specified part of the given string to the lower case string. |
| SubstringToUpper(String, Int32, Int32) | Converts the specified part of the given string to the upper case string. |
| TabColumnToOffset(String, Int32, Int32) | Converts the column in the given text line into a character offset. |
| TabifyLine(String, Int32) | Replaces the space characters in the given text line with the tab characters using the given tab size. |
| TabifyLine(String, Int32, Int32) | Replaces the space characters in the given text line with the tab characters using the given tab size starting from the specified index. |
| TabifyLine(String, Int32, Int32, Int32) | Replaces the space characters in the given text line with the tab characters using the given tab size starting from the specified index with the specified length. |
| TabifyLines(string[], Int32) | Replaces the space characters in the given string array with the tab characters using the given tab size. |
| TabifyLines(string[], Int32, Int32, Int32) | Replaces the space characters in the given string array with the tab characters using the given tab size starting from the specified index with the specified length. |
| TextToXML(String) | Converts the given text into XML (replacing special characters with XML analogs (quot; amp; lt; gt;), returning the converted value as a string. |
| ToIndex(Int32, String, SourcePoint) | Returns the index of the specified source point in the given text starting from the specified line number. |
| ToIndex(String, SourcePoint) | Returns the index of the specified source point in the given text. |
| ToSourcePoint(ArrayList, String, Int32) | Returns the source point of the specified index in the given text separated by the specified line terminators. |
| ToSourcePoint(String, Int32) | Returns the source point of the specified index in the given text. |
| ToSourceRange(ArrayList, String, Int32, Int32) | Returns the source range of the specified start and end indices in the given text separated by the specified line terminators. |
| ToSourceRange(String, Int32, Int32) | Returns the source range of the specified start and end indices in the given text. |
| TrimLinesEnd(String) | Trims the end of each line inside the given text. |
| UntabifyLine(String, Int32) | Replaces the tab characters in the given string with the space characters using the given tab size. |
| UntabifyLine(String, Int32, Int32) | Replaces the tab characters in the given string with the space characters using the given tab size starting from the specified index. |
| UntabifyLine(String, Int32, Int32, Int32) | Replaces the tab characters in the given string with the space characters using the given tab size starting from the specified index with the specified length. |
| UntabifyLines(string[], Int32) | Replaces the tab characters in the given string array with the space characters using the given tab size. |
| UntabifyLines(string[], Int32, Int32, Int32) | Replaces the tab characters in the given string array with the space characters using the given tab size starting from the specified index with the specified length. |

## DXCore Services – Strings

November 22nd, 2011

The **Strings** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) manipulates [DXCore string providers](http://www.skorkin.com/2011/08/dxcore-standard-string-providers-list/). It has methods that expand and format string providers for further expansion inside the code editor using the [DXCore plug-ins](http://www.skorkin.com/2010/08/dxcore-plug-ins-overview/).

Members of this service:

|  |  |
| --- | --- |
| Name | Description |
| Expand(String) | Returns a string with all string providers expanded that are found in the given text. |
| Format(String) | Formats a string provider for further expansion using the given name. |
| Format(String, string[]) | Formats a string provider for further expansion using the given name and parameters list. |
| Format(StringProviderBase) | Formats a string provider for further expansion using the given string provider. |
| Format(StringProviderBase, string[]) | Formats a string provider for further expansion using the given string provider and parameters list. |
| Get(String, String) | Returns a string supplied by the specified string provider name and parameters. Throws an exception if the specified string provider is not found. |
| ProvidersByLanguage(String) | Returns an array of all string providers for the given language ID. |
| Providers | Returns an array of all string providers. |

## DXCore Services – Synchronization

July 29th, 2011

The **Synchronization** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for synchronizing thread code on Visual Studio’s foreground thread. The service is used by the [CodeRush Code Issues technology](http://www.skorkin.com/2011/02/coderush-code-issues-technology-overview/) for example.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| BeginInvoke(Delegate) | Executes the specified delegate asynchronously on the thread that owns the IDE’s underlying window handle (the UI thread). |
| BeginInvoke(Delegate, object[]) | Executes the specified delegate asynchronously with the specified arguments, on the thread that the IDE’s underlying handle was created on (the UI thread). |
| EndInvoke(IAsyncResult) | Retrieves the return value of the asynchronous operation represented by the IAsyncResult passed. |
| Invoke(Delegate) | Executes the specified delegate on the thread that owns the IDE’s underlying window handle (the UI thread). |
| Invoke(Delegate, object[]) | Executes the specified delegate, with the specified list of arguments, on the thread that owns the IDE’s underlying window handle (the UI thread). |
| InvokeRequired | Gets a value indicating whether the caller must call an invoke method when making method calls because the caller is on a different thread than the one that the IDE was created on (the UI thread). |

## XCore Services – Templates

April 22nd, 2011

The **Templates** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides [CodeRush Code Templates](http://www.skorkin.com/2012/04/coderush-code-templates-overview/) APIs and some additional text expansion services. Using this service, you can [programmatically](http://www.skorkin.com/2011/04/how-to-loadchange-coderush-templates-from-inside-your-dxcore-plug-in/) create your own templates and/or modify the existing ones.

**Methods** of this service:

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| AddCategory (String, String) | Adds new templates category with the given name. |
| AddTemplate (String, String, String) | Adds new template to the specified category for the given language. |
| ExpandAtCursor(Boolean) | Expands the template at the cursor. Returns true if expansion was successful. Returns false if the template wasn’t found, if code is selected, or if the expansion was cancelled by a text command or a plug-in. |
| ExpandTemplate(String) | Expands the template specified by its name. Returns true if expansion was successful. Returns false if the template wasn’t found, if code is selected, or if the expansion was cancelled by a text command or a plug-in. |
| FillCommandMenu (MenuItem, EventHandler, Hashtable) | Fills a menu item’s submenu with all [TextCommands](http://www.skorkin.com/2011/01/dxcore-text-commands-list/). Sorted by the command name. Used by **Editor** | **Templates** options page. |
| FillStringProviderMenu (MenuItem, EventHandler, Hashtable, String) | Fills a menu item’s submenu with all StringProvidersfor the specified language. Sorted by the provider name. Used by **Editor**| **Templates**options page. |
| FilterTemplates(String) | Filters the Primary and Secondary template lists based on the specified programming language. |
| Find(String, Boolean) | Finds a specified template in the list. |
| FindCategory(String) | Returns a template category by its path. |
| FindTemplate (String, String, String) | Searches for the specified template inside the given category for the given language. |
| GetCategories(String) | Returns all root template categories for the specified language. |
| GetExpansion (TemplateData, String, Boolean) | Gets the template expansion text based on the specified TemplateDatainstance. Nested templates are recursively expanded, while StringProviders and TextCommands are left in place. This method is ideal for documenting template expansions. |
| GetPossibleTemplateNames (String) | Returns a collection of possible template names matching the given text. This method doesn’t check any context or if a template exists. |
| GetRegExAliasArray(String) | Retrieve an ArrayListcontaining all [RegEx](http://www.skorkin.com/2011/01/ide-tools-terminologykey-concepts/) aliases (system and regular) for the specified language. Sorted by the RegEx name. |
| GetStringProviders(String) | Retrieve an array containing all StringProvidersfor the specified language. Sorted by the provider name. |
| GetTemplateAtCursor (Boolean, Template, TemplateData) | Gets the Templateand TemplateItemcorresponding to the text to the left of the cursor. Multiple template possibilities are evaluated. The best (longest) template with matching context wins. |
| GetTextCommands | Retrieve an array containing all TextCommands. Sorted by the command name. |
| Reload | Reloads the language-neutral and language-specific template lists and divides them into two categories; one that responds to the primary template expansion key, and the other which responds to the secondary expansion key (available through the Primary and Secondary TemplateListproperties). |
| RemoveCategory (String, String, String) | Removes the category from the specified parent for the given language. |
| RemoveTemplate (String, String, String) | Removes the template from the specified category for the given language. |
| RenameCategory (String, String, String, String) | Renames the category in the specified parent for the given language. |
| RenameTemplate (String, String, String, String) | Renames the template in the specified category for the given language. |
| Save | Saves all templates. |
| SelectRegExAlias(String) | Creates a dialog which allows you to select a particular RegEx alias and return its instance. |
| SelectStringProvider(String) | Creates a dialog which allows you to select a particular [string provider](http://www.skorkin.com/2011/08/dxcore-standard-string-providers-list/) and return its instance. |
| SelectTextCommand | Creates a dialog which allows you to select a particular [text commands](http://www.skorkin.com/2011/01/dxcore-text-commands-list/) and return its instance. |

**Properties**:

|  |  |
| --- | --- |
| **Property Name** | **Description** |
| CheckingAvailability | Returns true if we’re in the process of checking template availability. Some contexts (e.g., OnEmptyLine) are satisfied differently if this is true. |
| ExpandingOnEmptyLine | Returns true if the line is empty when stripped of the template text. |
| ForEachElement | The IElementassociated with a ForEach template expansion. As the ForEach TextCommandexpands, for each IElement this property will be set. |
| LeftText | Returns the document text to the left of the currently expanding template. |
| PossibleTemplateAtCursor | Returns true if there is a possible template at the cursor. |
| Primary | Returns a list of templates triggered by the primary key (typically the Space key). |
| RightText | Returns the document text to the right of the currently expanding template. |
| Secondary | Returns a list of templates triggered by the secondary key (typically the Shift+Space key). |
| TemplateStoragePath | Gets the path (rooted from the [CodeRush settings path](http://www.skorkin.com/2010/08/where-ide-tools-coderush-andor-refactor-settings-are-stored/)) where templates are stored. |

## DXCore Services – Test

December 20th, 2011

The **Test** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides services for the functional tests runner.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| Delay(Int32) | Suspends the current thread for a specified time (in milliseconds). |
| DeleteTestFile(String) | Deletes the specified temporary test file. |
| DeleteTestFile(String, String) | Deletes the specified temporary test file. |
| ExtractTestFile(Assembly, String, String, String) | Extracts the test file from resources embedded to the running assembly and creates a temporary copy at the specified path. |
| GetAllTests | Returns the collection of all test cases. Note that it is hidden from Intellisense. |
| MouseClick | Emulates the mouse click at the current mouse cursor position. |
| MouseClick(Control, Point) | Emulates the mouse click at the specified point of the given control. |
| MouseClick(Control, Point, MouseButtons) | Emulates the mouse click with the specified mouse buttons at the specified point of the given control. |
| MouseClick(MouseButtons) | Emulates the mouse click with the specified mouse buttons at the current mouse cursor position. |
| MouseClick(Point) | Emulates the mouse click at the specified point. |
| MouseClick(Point, MouseButtons) | Emulates the mouse click at the specified point with the specified mouse buttons. |
| MouseDblClick | Emulates the mouse double click at the current mouse cursor position. |
| MouseDblClick(Control, Point) | Emulates the mouse double click at the specified point of the given control. |
| MouseDblClick(Control, Point, MouseButtons) | Emulates the mouse double click with the specified mouse buttons at the specified point of the given control. |
| MouseDblClick(MouseButtons) | Emulates the mouse double click with the specified mouse buttons at the current mouse cursor position. |
| MouseDblClick(Point) | Emulates the mouse double click at the specified point. |
| MouseDblClick(Point, MouseButtons) | Emulates the mouse double click at the specified point with the specified mouse buttons. |
| MouseDown | Emulates the mouse down at the current mouse cursor position. |
| MouseDown(Control, Point) | Emulates the mouse down at the specified point of the given control. |
| MouseDown(Control, Point, MouseButtons) | Emulates the mouse down with the specified mouse buttons at the specified point of the given control. |
| MouseDown(MouseButtons) | Emulates the mouse down with the specified mouse buttons at the current mouse cursor position. |
| MouseDown(Point) | Emulates the mouse down at the specified point. |
| MouseDown(Point, MouseButtons) | Emulates the mouse down at the specified point with the specified mouse buttons. |
| MouseUp | Emulates the mouse up at the current mouse cursor position. |
| MouseUp(Control, Point) | Emulates the mouse up at the specified point of the given control. |
| MouseUp(Control, Point, MouseButtons) | Emulates the mouse up with the specified mouse buttons at the specified point of the given control. |
| MouseUp(MouseButtons) | Emulates the mouse up with the specified mouse buttons at the current mouse cursor position. |
| MouseUp(Point) | Emulates the mouse up at the specified point. |
| MouseUp(Point, MouseButtons) | Emulates the mouse up at the specified point with the specified mouse buttons. |
| MoveMousePointTo(Control, Point) | Moves the mouse cursor at the specified point of the given control. |
| MoveMousePointTo(Point) | Moves the mouse cursor at the specified point. |
| OpenTestFile(String) | Opens the specified test file and parses its source code. Returns null if the file does not exist or could not be parsed. |
| SendKey(Control, Char) | Sends the specified character key code to the specified control. |
| SendKey(IntPtr, Char) | Sends the specified character key code to the specified handle. |
| SendString(Control, String) | Sends the specified keys sequence to the specified control. |
| SendString(IntPtr, String) | Sends the specified keys sequence to the specified handle. |
| StartRunner(String, String, Int64, String) | Starts the test runner. Note that it is hidden from Intellisense. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| KeyboardDelay | Gets or sets the delay in milliseconds when sending keys to a keyboard. Note that it is hidden from Intellisense. |
| MouseMoveDelay | Gets or sets the delay in milliseconds when moving the mouse cursor. Note that it is hidden from Intellisense. |
| Providers | Returns an array of all registered test providers. |
| RunningTest | Gets the test currently running, if any. |

## XCore Services – Text Buffers

June 12th, 2012

The **Text Buffers** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides low-level access to the text buffers in Visual Studio. Text buffers manage the stream of Unicode text that is entered into the IDE text editor and file persistence, including saving text files. A text buffer can use either a one- or two-dimensional coordinate system to identify character positions in the buffer.

Methods of this service:

|  |  |
| --- | --- |
| Name | Description |
| FromDocCookie(Int32) | Returns the text buffer associated with the specified doc cookie passed as an argument to this method. |
| FromTextDocument (TextDocument) | Returns the text buffer associated with the specified text document passed as an argument to this method. |
| GetEnumerator | Returns the Enumerator for opened text buffers. |
| NewMultiFileCompoundAction (String) | Creates and returns a multi-file compound action required to perform the undo/redo action of the multi-file editing operations. |
| NewMultiFileCompoundAction (String, Boolean) | Creates and returns a multi-file compound action required to perform the undo/redo action of the multi-file editing operations. Takes a boolean parameter which specifies whether the text selection should be restored on undo. |
| NewMultiFileCompoundAction (String, Int32, SourceRange) | Creates and returns a multi-file compound action required to perform the undo/redo action of the multi-file editing operations. Takes the top line and the source range of the selection as parameters to restore them on undo. |
| NewMultiFileCompoundAction (String, Int32, SourceRange, String) | Creates and returns a multi-file compound action required to perform the undo/redo action of the multi-file editing operations. Takes the top line, the source range of the selection, and the file name as parameters to restore them on undo. |
| Open(String) | Opens a text buffer specified by its full name. |

Properties:

|  |  |
| --- | --- |
| Name | Description |
| Active | Returns the active opened text buffer. |
| Count | Returns the count of opened text buffers. |
| Item(Int32) | Returns the text buffer by its index. |
| Item(String) | Returns the text buffer by its full name. |

## DXCore Services – Text Classification

February 14th, 2012

The **Text Classification** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for Visual Studio 2010+ text classification manipulation. A text classification logically classifies the content of the code editor (e.g., the text behind the code editor) and highlights elements of the editor content matching a classification. A classifiers main task is to recognize elements of the underlying text buffer and highlight those that match a classification type.

Members of this service:

|  |  |
| --- | --- |
| Name | Description |
| ClearTextClassification(TextView, SourceRange) | Clears the text classification in the given source range of the specified text view. |
| RemoveTextClassification(TextView, String, SourceRange) | Removes the text classification in the given source range of the specified text view. |
| SetTextClassification(TextView, String, SourceRange, ClassificationProperties) | Sets a text classification in the given source range of the specified text view with the specified classification properties. |
| UpdateMarkerClassification( ITextMarker) | Updates the specified line marker. |
| Engine | Gets the instance of the ITextClassificationEngine. |

## DXCore Services – TextCommands

March 17th, 2011

The **TextCommands** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to registered text commands inside [DXCore](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/) and allows you to format a [text command](http://www.skorkin.com/2011/01/ide-tools-terminologykey-concepts/) to prepare it for expansion. Formatting of a text command means adding special characters defined in the [Constants DXCore service](http://www.skorkin.com/2010/10/dxcore-services-constants/) to be able to expand it using the **TextExpansions** service. For example, to format the Caret text command, pass its name to the Format method:

|  |  |
| --- | --- |
| 1 | CodeRush.TextCommands.Format("Caret") |

and as a result, it will return: «Caret».

After the string, containing this formatted text command, is expanded, the text command will be executed, e.g. expanding this text inside code editor:

|  |  |
| --- | --- |
| 1 | int count = «Caret»100; |

will produce the following result:

|  |  |
| --- | --- |
| 1 | int count = 100; |

with the editor caret positioned right at the beginning of the “100” digit expression.

Methods of the service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| Format(TextCommand) | Formats the text command using its name. |
| Format(String) | Formats the text command using its name passed as string. |
| Format(TextCommand, String[]) | Formats the text command using its name and a given list of parameters. |
| Format(String, String[]) | Formats the text command using its name and a given list of parameters. |
| GetAll | Returns an array of all registered TextCommands in DXCore. |
| GetAll(Boolean) | Returns an array of all registered TextCommands in DXCore. The boolean parameter specifies whether the result array is sorted. |
| GetEnumerator | Returns   an enumerator of all registered TextCommands in DXCore. |

Also, there’s an indexed property, returning the TextCommand object by its name.

## DXCore Services – TextExpansions

January 14th, 2011

**Text Expansions DXCore** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for expanding dynamic text containing encoded [text commands](http://community.devexpress.com/blogs/rorybecker/archive/2010/11/23/coderush-common-textcommands.aspx) and/or [string providers](http://community.devexpress.com/blogs/rorybecker/archive/2010/11/19/coderush-common-stringproviders.aspx). **CodeRush** [Templates](http://community.devexpress.com/blogs/rorybecker/archive/2010/09/14/coderush-templates-101.aspx) is an example of such a feature. Note that the service is hidden from Intellisense.

One of the most important methods is “**Insert**” (with various overloads), which allows you to expand the text you prepared. All **Insert** methods return the **SourceRange** of the expanded text. For example, when you expand the “ai” template in CSharp (to create a new auto-implemented property), the following text is being expanded:

«?ScopeProperty(true)»«TypeLink(“«?Get(Type)»”)» «BlockAnchor»«FieldStart(Property name)»«Link(PropertyName)»«FieldEnd»«Caret» {get; set;}«Target»

You can use this service when developing [DXCore plug-ins](http://skorkin.com/2010/08/dxcore-plug-ins-overview/) with different refactoring/code providers, which need insertion of complex text expansion.

Here are **methods**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| BeginExpansion | Marks the beginning of a dynamic text insertion session. This call is needed to correctly process all encoded text commands. If you call this method, be sure to call EndExpansion. Note that Insert methods do not need to call this method, because Insert methods call it automatically. |
| CancelExpansion | Cancels the text expansion session if one is currently active. |
| EndExpansion | Marks the end of a dynamic text insertion action. |
| Insert(String) | Inserts text containing encoded text commands at the active editor caret point. |
| Insert(String, Boolean) | Inserts text containing encoded text commands at the active editor caret point. Specifies whether to format the expanded text. |
| Insert(TextDocument, SourcePoint, String) | Inserts text containing encoded text commands at the specified point. Specifies whether to format the expanded text. |
| Insert(String, Boolean, Boolean) | Inserts text containing encoded text commands at the active editor caret point. Specifies whether to format the expanded text and whether the expansion can be cancelled. |
| Insert(TextDocument, SourcePoint, String, Boolean) | Inserts text containing encoded text commands at the specified point and text document. Specifies whether to format the expanded text. |
| Insert(TextDocument, SourcePoint, String, Boolean, Boolean) | Inserts text containing encoded text commands at the specified point and text document. Specifies whether to format the expanded text and whether the expansion can be cancelled. |
| Insert(TextDocument, SourcePoint[], String[], Boolean, Boolean) | Inserts multiple text expansions at the specified text coordinates in the specified text document. Specifies whether to format the expanded text and whether the expansion can be cancelled. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| IsExpanding | Returns true if the text expansion is currently being expanding. |
| Variables | Provides get/set access to string variables used during expansion. String variables can be added by text expansions using the Set StringProvider, and queried using the Get StringProvider (as well as being set and retrieved programmatically through this property). |

## DXCore Services – Text Fields

June 12th, 2012

The **TextFields** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the [Text Fields](http://www.skorkin.com/2011/01/ide-tools-terminologykey-concepts/) engine. Text fields allow you to enter data and then press Num Enter or Enter to quickly move to the next text field. Text fields are highlighted in the editor and usually have a tool tip below describing the data expected.

The service contains a single method called GetDocumentFieldAdornment. The method creates a new TextDocumentAdornment object which represents a text field. The method takes three arguments:

* start point (EditPoint)
* end point (EditPoint)
* a boolean parameter which specifies whether the new text field is active

## DXCore Services – Text Views

November 28th, 2011

The **TextViews** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods and properties for text view manipulation. A text view is a window in Visual Studio IDE that lets you type, edit and view source code text.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| CreateNewTextView(IVsTextView) | Creates and registers a new text view from the specified IVsTextView and VsTextEditPane sub-class. |
| FindByHandle(HWND) | Returns an IVsTextView text view instance by the given HWND. |
| Focus(TextView) | Focuses the specified text view. |
| GdiPaint(EditorPaintEventHandler) | Iterates through all text views for the active document, calling the specified event handler. |
| Get | Returns an array of text views for the active text document. |
| Get(Boolean) | Returns an array of text views for the active text document. A boolean parameter specifies whether errors are suppressed (a method will return null) or not (an exception will be thrown). |
| Get(TextDocument) | Returns an array of text views for the specified text document. |
| Get(TextDocument, Boolean) | Returns an array of text views for the specified text document. A boolean parameter specifies whether errors are suppressed (a method will return null) or not (an exception will be thrown). |
| GetEnumerator | Returns an enumerator for active text views. |
| Refresh | Invalidates all text views. |
| RequestTextView(IVsTextView) | Returns an existing text view by the given IVsTextView instance or creates a new one. |
| RestorePosition(ViewPosition) | Restores the specified view. If the file associated with the view has been closed, it will be reopened. The restored text view is returned. NOTE: This method sets focus to the restored by posting a message to the TextView’s message queue. So, calling CodeRush.TextViews.Active or TextView.Active is not guaranteed to return the newly restored text view until that message is handled. If any other operations need to be performed on the restored view, use the text view returned by this method. |
| SavePosition | Creates and returns a new ViewPosition object for the active text view. |
| SavePosition(TextView) | Creates and returns a new ViewPosition object for the specified text view. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| Active | Gets the active text view. |
| Count | Get the count of active text views. |
| ImeCompositionActive | Determines whether an IME composition is in an active state, and we’re waiting for the end of the composition. IME stands for Input Method Editor. IMEs are typically used to input unicode characters such as Kanji. |
| Item | An indexed property that returns a text view by the given index. |
| LineHeight | Returns the line height of the active text view. If there is no active text view, zero is returned. |

## DXCore Services – ThirdPartyExtensions

November 22nd, 2010

The **ThirdPartyExtensions** [service](http://skorkin.com/2010/09/dxcore-services-list/) provides methods to check whether or not specified Visual Studio 2010 extensions are installed and/or enabled. This service also contains a few useful properties for the ***Microsoft Power Tools*** extension.

**Methods** of this service:

|  |  |
| --- | --- |
| **Name** | **Description** |
| IsExtensionEnabled | Returns true if the extension with the specified name, identifier and author is installed and enabled. |
| IsExtensionInstalled | Returns true if the extension with the specified name, identifier and author is installed. |

**Properties**:

|  |  |
| --- | --- |
| **Name** | **Description** |
| IsAutoBraceCompleteOn | Returns true if Microsoft Productivity Power Tools extension is installed and the AutoBraceComplete feature is enabled. |
| IsPowerToolsEnabled | Returns true if Microsoft Productivity Power Tools extension is installed and enabled. |
| IsPowerToolsInstalled | Returns true if Microsoft Productivity Power Tools extension is installed. |
| IsQuickFindActive | Returns true if Microsoft Productivity Power Tools QuickFind feature is active. |

## DXCore Services – Tool Windows

November 28th, 2011

The **ToolWindows** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for [DXCore](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/) and Visual Studio tool windows manipulation.

|  |  |
| --- | --- |
| Method name | Description |
| GetEnumerator | Returns an enumerator for available tool windows inside Visual Studio IDE. |
| GetPlugIn | Returns the [DXCore plug-in](http://www.skorkin.com/2010/08/dxcore-plug-ins-overview/) instance containing  the specified **DXCore** tool window. Note that it is hidden from Intellisense. |
| GetPlugIn(Guid) | Returns the **DXCore plug-in** instance containing  the DXCore tool window specified by its GUID. Note that it is hidden from Intellisense. |
| GetPlugIn(String) | Returns the DXCore plug-in instance containing the DXCore tool window specified by its title. Note that it is hidden from Intellisense. |
| GetPlugIn(Type) | Returns the DXCore plug-in instance containing  the DXCore tool window specified by its type. Note that it is hidden from Intellisense. |
| GetPlugInControl | Returns the base UserControl of a DXCore tool window plug-in for the specified tool window. |
| GetPlugInControl(Guid) | Returns the base UserControl of a DXCore tool window plug-in for the specified tool window by its GUID. |
| GetPlugInControl(String) | Returns the base UserControl of a DXCore tool window plug-in for the specified tool window by its title. |
| GetPlugInControl(Type) | Returns the base UserControl of a DXCore tool window plug-in for the specified tool window by its type. |
| Hide | Hides the specified tool window. |
| Hide(Guid) | Hides the tool window specified by its GUID. |
| Hide(String) | Hides the tool window specified by its title. |
| Hide(Type) | Hides the tool window specified by its type. |
| Show | Shows the specified tool window. |
| Show(Guid) | Shows the specified tool window by its GUID. |
| Show(String) | Shows the specified tool window by its title. |
| Show(Type) | Shows the specified tool window by its GUID. |

Indexed properties:

|  |  |
| --- | --- |
| Item(Guid) | Returns the tool window by the specified GUID. |
| Item(Type) | Returns the tool window by the specified Type. |

## DXCore Services – UndoStack

September 19th, 2011

The **Undo Stack** [DXCore Service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the Undo Manager and undo/redo stack of the Visual Studio IDE.

Here are the methods of this service:

|  |  |
| --- | --- |
| Name | Description |
| Abort | Aborts the previously open undo context (canceling all edits made since the context was open). You can open an undo context by calling Undo.BeginUpdate(); |
| Add(IUndoUnit) | Adds a custom undo unit to the top of the active undo stack. Note: depending on the state of the stack this may be added to the redo stack. (E.g., if the stack is currently undoing.) Note that it is hidden from Intellisense. |
| Add(TextDocument, IUndoUnit) | Adds a custom undo unit to the top of the specified text document’s undo stack. Note: depending on the state of the stack this may be added to the redo stack. (E.g., if the stack is currently undoing.) Note that it is hidden from Intellisense. |
| BeginUpdate(String) | Marks the start of a sequence of changes that will be treated as a single undo operation (also known as an “undo context”). |
| BeginUpdate(String, Boolean) | Marks the start of a sequence of changes that will be treated as a single undo operation (also known as an “undo context”). You can nest multiple paired calls to BeginUpdate and EndUpdate within each other. |
| EndUpdate | Marks the end of a sequence of changes that will be treated as a single undo operation. If this is the outermost EndUpdate for a particular sequence of actions, then the previously-open undo context will be closed and the edit actions will be committed. You can nest multiple paired calls to BeginUpdate and EndUpdate within each other. |
| GetRedoDescriptions | Returns an array of redo descriptions from the redo stack for the active text document. |
| GetRedoDescriptions (TextDocument) | Returns an array of redo descriptions from the redo stack for the given text document. |
| GetUndoDescriptions | Returns an array of undo descriptions from the undo stack for the active text document. |
| GetUndoDescriptions (TextDocument) | Returns an array of undo descriptions from the undo stack for the given text document. |
| InsertMarker(String) | Inserts an empty undo unit (or “undo marker”) onto the undo stack of the specified text document. This is useful for situations in which the undo stack needs to be restored to a specific state. A System.Int32 cookie value is returned, which can be used with RollbackToMarker to restore the undo stack state. |
| InsertMarker (TextDocument, String) | Inserts an empty undo unit (or “undo marker”) onto the undo stack of the specified text document. This is useful in situations in which the undo stack needs to be restored to a specific state. A System.Int32 cookie value is returned, which can be used with RollbackToMarker to restore the undo stack state. |
| MarkerIsLast (TextDocument, Int32) | Defines if the specified undo marker is the last in the undo stack. |
| Redo | Performs a redo operation. |
| RedoExecuted | Returns true if a redo operation is executed. |
| RedoTo(String) | Executes a redo up to and including the unit with the specified description on the active redo stack. |
| RedoTo(TextDocument, String) | Executes a redo up to and including the unit with the specified description on the redo stack of the specified text document. |
| RollbackToMarker (Int32) | Returns the state of the undo stack of the active text document to its position before the specified undo marker. |
| RollbackToMarker (TextDocument, Int32) | Returns the state of the undo stack of the specified text document to its position before the specified undo marker. |
| Undo | Performs an undo operation. |
| UndoTo(String) | Executes an undo up to and including the unit with the specified description on the active undo stack. |
| UndoTo(String, String) | Executes an undo up to and including the unit with the specified description on the active undo stack of the given text buffer. |
| UndoTo(TextDocument, String) | Executes an undo up to and including the unit with the specified description on the undo stack of the specified text document. |

Properties:

|  |  |
| --- | --- |
| Name | Description |
| IsRedoing | Returns true if a redo operation is executing. |
| IsUndoing | Returns true if an undo operation is executing. |
| IsValidUndoManager | Checks if the undo manager is valid. |
| LastRedoDescription | Returns the description of the last unit on the redo stack or an empty string if the redo stack is empty. |
| LastUndoDescription | Returns the description of the last unit on the undo stack or an empty string if the undo stack is empty. |

## DXCore Services – UnitTests

July 4th, 2011

The **UnitTests** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for executing and debugging unit test cases and test collections in the specified scopes. Allows to add custom context menu items into the drop-down menu of a [Unit Test Runner](http://www.skorkin.com/2011/07/coderush-unit-testing-service-overview/) Tile in the code editor.

**Methods** of the service:

|  |  |
| --- | --- |
| **Method name** | **Description** |
| AbortTestRun | Aborts the currently unit test running session. |
| CreateContextMenuItem | Creates a custom context menu item into the drop-down menu of a Test Runner Tile in the code editor. |
| Debug | Starts debugging all test cases in the current solution. |
| Debug(IMemberElement) | Starts debugging all test cases in the specified member. |
| Debug(INamespaceElement) | Starts debugging all test cases in the specified namespace. |
| Debug(IProjectElement) | Starts debugging all test cases in the specified project. |
| Debug(ITypeElement) | Starts debugging all test cases in the specified type declaration. |
| Debug(TestMethod) | Starts debugging the specified test method. |
| Debug(TestMethodCollection) | Starts debugging the specified test method collection. |
| Execute | Executes all test cases in the current solution. |
| Execute(IMemberElement) | Executes all test cases in the specified member. |
| Execute(INamespaceElement) | Executes all test cases in the specified namespace. |
| Execute(IProjectElement) | Executes all test cases in the specified project. |
| Execute(ITypeElement) | Executes all test cases in the specified type declaration. |
| Execute(TestMethod) | Executes the specified test method. |
| Execute(TestMethodCollection) | Executes the specified test method collection. |

**Properties**:

|  |  |
| --- | --- |
| **Property name** | **Description** |
| ServiceState | Returns a state of the testing service (testing started, searching of test cases, building projects, pending). |
| Tests | Returns a test method collection containing all found test cases in the current solution. |

## DXCore Services – Version

November 28th, 2011

The **Version** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides [DXCore](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/)/[IDE Tools](http://www.skorkin.com/2010/06/devexpress-visual-studio-productivity-add-ins/) version information.

|  |  |
| --- | --- |
| Property name | Description |
| Build | Returns the build number as a string. |
| Classification | Returns the release state (e.g. “Beta“, “Early Experience“, “Release Candidate“, etc). |
| FullName | Returns the full version number of this product (e.g. “11.2 (Beta)“). |
| Major | Returns the major version number as a string. |
| MajorDotMinor | Returns the major version number and minor version numbers separated by a dot “.” (e.g. “1.0”). |
| Minor | Returns the minor version number as a string. |
| Revision | Returns the revision number as a string. |
| Version | Returns the Version (DevExpress.CodeRush.Common) instance for the current product. |
| VERSION\_Build | Returns the build number. |
| VERSION\_Major | Returns the major version number. |
| VERSION\_Minor | Returns the minor version number. |
| VERSION\_Revision | Returns the revision number. |

## DXCore Services – Visual Studio Settings

December 14th, 2011

The **VSSettings** (Visual Studio Settings) [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the Visual Studio IDE settings (Visual Studio version, environment settings, fonts and colors settings, etc).

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| EngageVBPrettyListing | Engages VB’s pretty listing feature. This method does not check to see if VB is the active language, so you might want to perform that test before calling. After calling this method, you must call RestoreVBPrettyListing to ensure that pretty listing features are properly restored. |
| GetAtStartUpValue | Gets the value of the “At startup” Visual Studio property. NOTE: If the value cannot be retrieved for any reason, AtStartUp.Unknown will be returned. |
| GetFontAndColorCategory( String) | Retrieves the FontAndColorCategory object for the specified category name. |
| GetFontAndColorSetting( String) | Returns a FontAndColorSetting object for the specified name from the “Text Editor” category. |
| GetFontAndColorSetting( String, String) | Returns a FontAndColorSetting object for the specified name from the given category. |
| GetLanguageSettings( String) | Returns the LanguageSettings object for the specified language. |
| GetTabSettings( ParserLanguageID) | Returns the ITabSettings instance containing the tab size in characters and the value indicating whether to insert spaces instead of tabs. |
| GetTabSize(Guid) | Returns the tab characters size for the specified language service ID. |
| GetVBProperty(String) | Gets the EnvDTE.Property associated with the “Basic-Specific” page of the TextEditor options category. |
| GetVersion | Returns the version of Visual Studio. Note that it is hidden from Intellisense. |
| GetVirtualSize(Guid) | Returns the size of the virtual space. |
| GetWordWrap(Guid) | Returns the value of the word wrap option. |
| GetXMLTagName | Returns the name of the XML tag settings category. |
| ReloadEditorFont | Reloads the editor font. This method is useful if the editor font is changed in a manner other than through the VS Fonts and Color settings page in the VS Options dialog. |
| RestoreVBPrettyListing | Restores VB’s pretty listing feature to the state it was in before calling SuppressVBPrettyListing or EngageVBPrettyListing. You must call this method after calling SuppressVBPrettyListing or EngageVBPrettyListing to ensure that pretty listing features are properly restored. |
| SetAtStartUpValue( AtStartUp) | Sets the value of the “At startup” Visual Studio property. |
| SuppressVBPrettyListing | Suppresses VB’s pretty listing feature. This method does not check to see if VB is the active language, so you might want to perform that test before calling. After calling this method you must call RestoreVBPrettyListing to ensure that pretty listing features are properly restored. |
| VersionAtLeast( VisualStudioVersion) | Determines whether the currently running version of Visual Studio is the same or greater than the specified version. |
| VersionAtMost( VisualStudioVersion) | Determines whether the currently running version of Visual Studio is the same or less than the specified version. |
| VersionGreaterThan( VisualStudioVersion) | Determines whether the currently running version of Visual Studio is greater than the specified version. |
| VersionIs( VisualStudioVersion) | Determines whether the currently running version of Visual Studio is the same as the specified version. |
| VersionLessThan( VisualStudioVersion) | Determines whether the currently running version of Visual Studio is less than the specified version. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| BackgroundColor | Gets the background color of the code editor. |
| BoldFontHandle | Gets the handle of the bold fonts used by text views. |
| Bookmark | Gets the “Bookmark” FontAndColorSetting instance. |
| BraceMatching | Gets the “Brace Matching ” FontAndColorSetting instance. |
| BreakpointDisabled | Gets the “Breakpoint Disabled” FontAndColorSetting instance. |
| BreakpointEnabled | Gets the “Breakpoint Enabled” FontAndColorSetting instance. |
| BreakpointError | Gets the “Breakpoint Error” FontAndColorSetting instance. |
| BreakpointWarning | Gets the “Breakpoint Warning” FontAndColorSetting instance. |
| CallReturn | Gets the “Call Return” FontAndColorSetting instance. |
| CollapsibleText | Gets the “Collapsible Text” FontAndColorSetting instance. |
| Comment | Gets the “Comment” FontAndColorSetting instance. |
| CompilerError | Gets the “Compiler Error” FontAndColorSetting instance. |
| CSSComment | Gets the “CSS Comment” FontAndColorSetting instance. |
| CSSKeyword | Gets the “CSS Keyword” FontAndColorSetting instance. |
| CSSPropertyName | Gets the “CSS Property Name” FontAndColorSetting instance. |
| CSSPropertyValue | Gets the “CSS Property Value” FontAndColorSetting instance. |
| CSSSelector | Gets the “CSS Selector” FontAndColorSetting instance. |
| CSSStringValue | Gets the “CSS String Value” FontAndColorSetting instance. |
| CurrentListLocation | Gets the “Current list location” FontAndColorSetting instance. |
| CurrentStatement | Gets the “Current Statement” FontAndColorSetting instance. |
| DebuggerDataChanged | Gets the “Debugger Data Changed” FontAndColorSetting instance. |
| DisassemblyFileName | Gets the “Disassembly File Name” FontAndColorSetting instance. |
| DisassemblySource | Gets the “Disassembly Source” FontAndColorSetting instance. |
| DisassemblySymbol | Gets the “Disassembly Symbol ” FontAndColorSetting instance. |
| DisassemblyText | Gets the “Disassembly Text” FontAndColorSetting instance. |
| Environment | Gets the EnvironmentSettings instance. |
| EnvironmentFontCategory | Gets the FontAndColorCategory object for the “Environment Font” category. NOTE: This is only valid in Visual Studio 2008 or later. In earlier versions of Visual Studio, this will return null. |
| ExcludedCode | Gets the “Excluded Code” FontAndColorSetting instance. |
| Font | Gets the font used in the code editor. |
| FontCharSet | Gets the CharSet of the font used in the code editor. |
| FontName | Gets the name of the font used in the code editor. |
| FontSize | Gets the size of the font used in the code editor. |
| FunctionBlockStart | Gets the “Function Block Start” FontAndColorSetting instance. |
| HasBreakingFormatting | Gets the value indicating whether the current Visual Studio has breaking formatting. |
| HasInstantLineChanges CommittedEvent | Gets the value indicating whether the current Visual Studio has the instant line changes committed event. |
| HasRestorableLineMarkers | Gets the value indicating whether the current Visual Studio has restorable line markers. |
| HasTextChangedEventsQueue | Gets the value indicating whether the current Visual Studio has the text changed events queue. |
| HTMLAttributeName | Gets the “HTML Attribute Name” FontAndColorSetting instance. |
| HTMLAttributeValue | Gets the “HTML Attribute Value ” FontAndColorSetting instance. |
| HTMLComment | Gets the “HTML Comment” FontAndColorSetting instance. |
| HTMLElementName | Gets the “HTML Element Name” FontAndColorSetting instance. |
| HTMLEntity | Gets the “HTML Entity” FontAndColorSetting instance. |
| HTMLOperator | Gets the “HTML Operator” FontAndColorSetting instance. |
| HTMLServerSideScript | Gets the “HTML Server-Side Script” FontAndColorSetting instance. |
| HTMLString | Gets the “HTML String” FontAndColorSetting instance. |
| HTMLTagDelimiter | Gets the “HTML Tag Delimiter” FontAndColorSetting instance. |
| HTMLTagText | Gets the “HTML Tag Text” FontAndColorSetting instance. |
| Identifier | Gets the “Identifier” FontAndColorSetting instance. |
| InactiveSelectedText | Gets the “Inactive Selected Text” FontAndColorSetting instance. |
| IndicatorMargin | Gets the “Indicator Margin” FontAndColorSetting instance. |
| IndicatorMarginVisible | Gets the “Indicator Margin Visible” FontAndColorSetting instance. |
| Keyword | Gets the “Keyword” FontAndColorSetting instance. |
| LineNumbers | Gets the “Line Numbers” FontAndColorSetting instance. |
| MemoryAddress | Gets the “Memory Address” FontAndColorSetting instance. |
| MemoryChanged | Gets the “Memory Changed” FontAndColorSetting instance. |
| MemoryData | Gets the “Memory Data” FontAndColorSetting instance. |
| MemoryUnreadable | Gets the “Memory Unreadable” FontAndColorSetting instance. |
| Number | Gets the “Number” FontAndColorSetting instance. |
| Operator | Gets the “Operator” FontAndColorSetting instance. |
| OtherError | Gets the “Other Error” FontAndColorSetting instance. |
| OverType | Gets the “Over Type” FontAndColorSetting instance. |
| PlainText | Gets the “Plain Text” FontAndColorSetting instance. |
| PreprocessorKeyword | Gets the “Preprocessor Keyword” FontAndColorSetting instance. |
| ReadOnlyRegion | Gets the “Read-Only Region” FontAndColorSetting instance. |
| RegisterData | Gets the “Register Data” FontAndColorSetting instance. |
| RegisterNAT | Gets the “Register NAT” FontAndColorSetting instance. |
| RegularFontHandle | Gets the handle of the regular font used in the code editor. |
| SelectedText | Gets the “Selected Text” FontAndColorSetting instance. |
| SelectionMarginVisible | Gets the “Selection Margin Visible” FontAndColorSetting instance. |
| Shortcut | Gets the “Shortcut” FontAndColorSetting instance. |
| SmartEditorError | Gets the “Smart Editor Error” FontAndColorSetting instance. |
| StaleCode | Gets the “Stale Code” FontAndColorSetting instance. |
| String | Gets the “String” FontAndColorSetting instance. |
| TextEditor | Gets the TextEditorSettings instance. |
| UndoCaretMovements | Gets the value indicating whether caret movements and text selections are included in the undo list. |
| UserKeyword | Gets the “User Keyword” FontAndColorSetting instance. |
| UserTypes | Gets the “User Types” FontAndColorSetting instance. |
| UserTypesDelegates | Gets the “User Types (Delegates)” FontAndColorSetting instance. |
| UserTypesEnums | Gets the “User Types (Enums)” FontAndColorSetting instance. |
| UserTypesInterfaces | Gets the “User Types (Interfaces)” FontAndColorSetting instance. |
| UserTypesValueTypes | Gets the “User Types (Value types)” FontAndColorSetting instance. |
| Version | Gets the Visual Studio version string, suitable for use in registry key names. |
| VisibleWhiteSpace | Gets the “Visible White Space” FontAndColorSetting instance. |
| VisualStudioVersion | Gets a VisualStudioVersion enumeration value. |
| WizardCode | Gets the “Wizard Code” FontAndColorSetting instance. |
| XMLDocComment | Gets the “XML Doc Comment” FontAndColorSetting instance. |
| XMLTag | Gets the “XML Tag” FontAndColorSetting instance. |

## DXCore Services – Windows

December 14th, 2011

The **Windows** [DXCore service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides methods for Visual Studio windows manipulation.

Methods of this service:

|  |  |
| --- | --- |
| Method name | Description |
| CloseIntellisense(Boolean) | Closes the Intellisense window if active. |
| ConvertWindowType( vsWindowType) | Converts an EnvDTE.vsWindowType enum to a WindowType enum. |
| ConvertWindowType( WindowType) | Converts a WindowType enum to an EnvDTE.vsWindowType enum. |
| DisplayIntellisense | Shows the Intellisense window if appropriate. |
| ForceCloseTipWindow | Closes the tip window if active. |
| GetCodeWindowFromView( IVsTextView) | Returns the corresponding IVsCodeWindow from an IVsTextView interface. |
| GetWindow(Guid) | Returns the EnvDTE.Window instance for the specified Guid. |
| GetWindow(String) | Returns the EnvDTE.Window instance for the specified Guid as a string. |
| GetWindow(Type) | Returns the EnvDTE.Window instance for the specified Type instance. |
| IsCodeWindow(Window) | Determines if the given window is a text window. |
| IsDesigner(Window) | Determines if the given window is a form designer. |
| IsHTMLCodeWindow( Window) | Determines if the given window is an HTML window in HTML (Source) view. |
| IsHTMLDesigner(Window) | Determines if the given window is an HTML window opened in Design view. |
| IsHTMLWindow(Window) | Determines if the given window is an HTML window. |
| ShowStartPage | Displays the Visual Studio start page. |
| ShowWindow(Guid) | Shows the window specified by Guid. |
| ShowWindow(String) | Shows the window specified by Guid as a string. |
| ShowWindow(Type) | Shows the window specified by Type instance. |

Properties:

|  |  |
| --- | --- |
| Property name | Description |
| Active | Gets the active window. |
| ActiveDocument | Gets the active document window. |
| ActiveTipWindowRect | Gets the TextView-relative coordinates of the active Tip window. Returns the Rect.Empty instance if no tip window is active. |
| ActiveType | Returns the window type of the active window. |
| CompletorPaneHandle | Gets the handle of the completer pane. Returns HWND.Empty if no completer pane is active. |
| IntellisenseEngine | Returns the IIntellisenseEngine instance with the Intellisense related APIs. |
| IsIntellisenseActive | Returns true if the Intellisense window is visible. |
| IsTipWindowActive | Returns true if the Tip window is visible. |
| LastActive | Returns the last active window. |
| LastActiveType | Returns the window type of the last active window. |
| TipWindowHandle | Gets the handle of the most recently displayed tool tip window. Returns HWND.Empty if no tool tip windows are active. |
| Toolbox | Returns the instance of the Toolbox window. |
| VisibleTipWindows | Returns an array of the currently visible tip windows. |

## DXCore Services – Wizards

July 4th, 2011

The **Wizards** [DXCore Service](http://www.skorkin.com/2010/09/dxcore-services-list/) provides access to the [DevExpress DXCore Framework](http://www.skorkin.com/2010/06/devexpress-dxcore-framework/) and Visual Studio wizards. The service is hidden from Intellisense when accessing it though the [CodeRush](http://www.skorkin.com/2010/11/coderush-object-for-accessing-to-dxcore-services/) object. Methods of this service have several overloads where you can specify additional parameters for launching a particular wizard.

|  |  |
| --- | --- |
| **Method name** | **Description** |
| LaunchAddItemWizard | Starts the Add New Item Visual Studio wizard for the specified project. |
| LaunchCodeRushPlugInWizard | Starts the [CodeRush/DXCore New Plug-In](http://www.skorkin.com/2010/08/how-to-create-a-new-dxcore-plug-in/) wizard. |
| LaunchNewProjectWizard | Starts the Add New Project Visual Studio wizard. |
| LaunchStandardPlugInWizard | Starts the Add New Item Visual Studio wizard with the [DXCore Standard plug-in](http://www.skorkin.com/2010/08/dxcore-plug-ins-overview/) items for CSharp selected. |

## DXCore Services – XPO

November 11th, 2010

The **XPO** [service](http://skorkin.com/2010/09/dxcore-services-list/) is dedicated for the [eXpress Persistent Object](http://www.devexpress.com/xpo) object-relational mapping tool from [DevExpress](http://www.devexpress.com/). At the moment, it contains only a single method ***IsPersistent***(**TypeDeclaration** type), which returns true if the passed type has a Persistent attribute and/or descends from XPObject, XPBaseObject or XPCustomObject, and the class does not include a NonPersistent attribute.

In the future this service is going to be extended with additional useful methods and properties.