WebTextEditor Editing Engine

This whitepaper describes the concept and features of editing engine introduced in WebTextEditor.

Table of Contents

Editing Architecture

Core Editing Features

WebTextEditor is using its own editing engine that developed to support various kinds of new scenarios and to provide richer user experience. In addition to common editing scenarios such as text editing, font formatting and paragraph editing, WebTextEditor covers more advanced scenarios such as discussed in the following points.

WebTextEditor includes key editing features such as:

- **Streamlined Cut, Copy and Paste Operation**
  WebTextEditor implements sophisticated clipboard management to address the limitations and different specifications of clipboard in various browsers. Based on how users invoke the clipboard saving operation such as cut and copy, WebTextEditor automatically uses the best approach to manage the clipboard data according to the browser type.

  This streamlined clipboard management is crucial to ensuring smooth clipboard operation in various browsers. As such, WebTextEditor manages the clipboard data and decides when it should get the clipboard data from browser, and when it should get from editor's internal cache.

  User can perform clipboard operations through the following ways:

  1. Toolbar command
     Operation invoked through toolbar commands for cut and copy operation will use internal cache clipboard management.

     The limitation from this approach is that user cannot paste the clipboard data from editor to other place except to the editor itself.

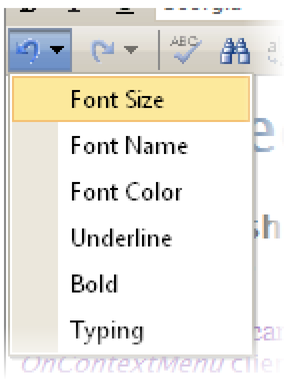
  2. Keyboard shortcut
     Operation invoked through keyboard shortcut for cut and copy operation will use browser default clipboard management.

     When using keyboard shortcut, WebTextEditor will perform both saving mechanism at the same time when cut or copy operation is invoked, this can be achieved by setting *EnableKeyboardShortcut* property to true.

  The purpose of this clipboard management is tightly integrated with paste operation. In paste operation, internal editor's cache of clipboard data will become the first priority to be pasted. If editor's internal cache is empty, WebTexteditor will obtain data from browser's default clipboard.

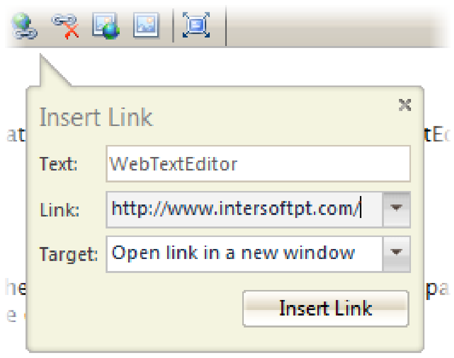
WebTextEditor also implements automatic sliding expiration for its internal clipboard cache to keep the clipboard data concurrent and latest.

- **Multiple Undo and Redo**
  WebTextEditor includes multiple undo and redo feature. It is unique in the way every action is logged with user-friendly message to a maximum of 20 stacks using FIFO (First-in-first-out) implementation. Each logged action can be found in Undo and Redo list, by clicking on the dropdown arrow of respective tool command.



WebTextEditor logs every action that cause changes to the editor, including typing. You can customize how fast WebTextEditor should log typing action. Simply set the provided *UndoLatency* property to an integer value measured in milliseconds. The default value is set to 1000(ms), which means every typing action will be logged on every 1000ms after the last stop.

- **Insert Link**
  WebTextEditor provides a new user interface to insert link, which is focused on simplicity and efficiency. Link insertion in WebTextEditor is displayed with lightweight, elegant callout. In the callout, user can easily specify the text of link, URL and other input.


Insert Link

Text: WebTextEditor

Link: http://www.intersoftpt.com/

Target: Open link in a new window

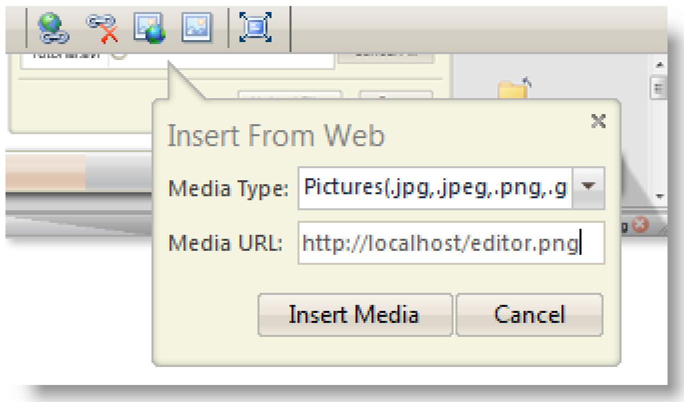
If there is a text selection when user clicked on “Insert Link” command, the selected text will be automatically set as the text of the link.

- **Insert Image**
  WebTextEditor introduces new ways to work with images and provides more flexibility for user to insert images. User can insert images from callout interface or Media Gallery pane.

  Task pane is an independent visual element on sidebar panel that displays related contents and editing actions which significantly improves overall user experience. It enables users to perform editing tasks more easily and conveniently in a single interface.

  Image insertion through callout can be done by clicking on “Insert from Web” command in the toolbar.

  For further information about how to insert image using Media Gallery pane, please refer to Task Pane topic in WebTextEditor User Interface whitepaper.



Insert From Web

Media Type: Pictures(.jpg,.jpeg,.png,.g

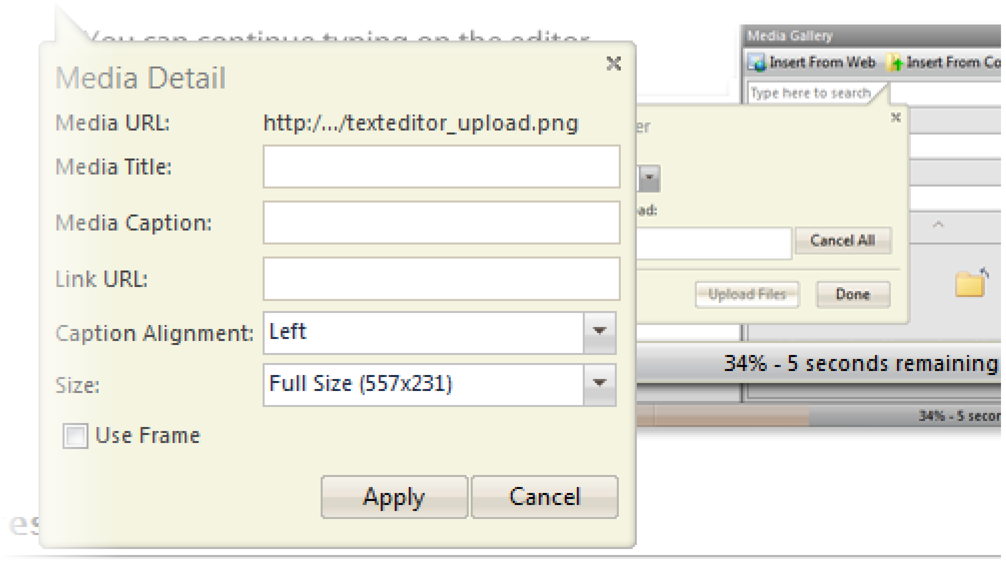
Media URL: http://localhost/editor.png

- **Edit Image**
  Every time an image is added to WebTextEditor, an image detail callout will appear near the added image. User can input the details of the image in this callout.

  There are several settings of image that can be customized:
  - Media Title
    Specify title to the image.
  - Media Caption
    Specify caption to the image. This will be reflected if the image is using frame.
  - Link URL
    Specify link URL of the image.
  - Caption Alignment

Specify caption alignment of the image. This will be reflected if the image is using frame.

- Size
  Specify the size of image.
- UseFrame
  Specify whether the image should be placed in a frame.

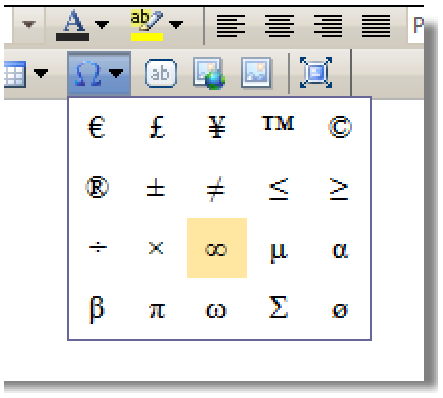


Media URL: http:/.../texteditor_upload.png
Media Title:
Media Caption:
Link URL:
Caption Alignment: Left
Size: Full Size (557x231)
Use Frame

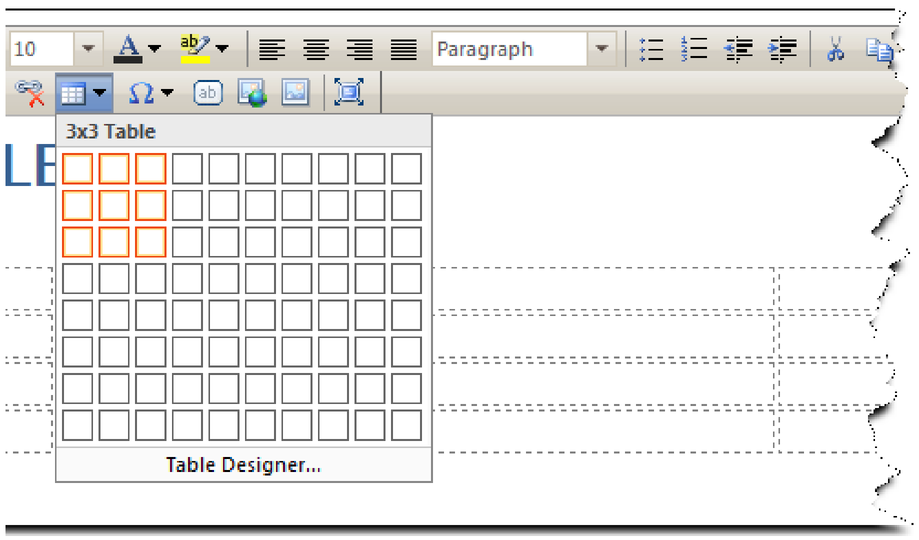
- **Insert Media**
  Beside images, user can also insert other media type in WebTextEditor such as audios, videos, flash video and even YouTube video. Several types of media file that supported by WebTextEditor are .wav, .wma, .wmv, .avi, .mov, .mpeg and .mpg.

  These media can also be inserted using callout and Media Gallery pane. Please refer to the Task Pane topic in WebTextEditor User Interface whitepaper for further information.

- **Insert Symbol**
  There are 20 predefined symbols in WebTextEditor; user can insert the symbol to editor with Symbol pop up. Simply click on the “Insert Symbol” command on the toolbar, and choose the symbol to be inserted from the symbol popup.



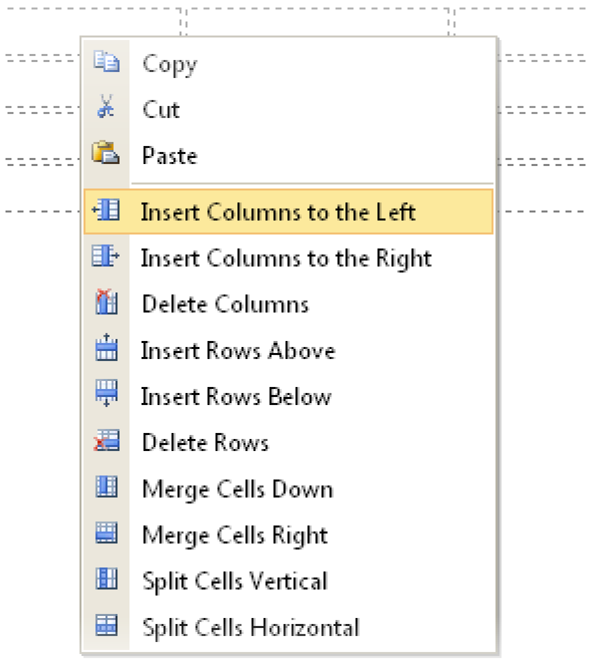
- **Insert Table**
  WebTextEditor introduces new ways to work with table. User can insert table through intuitive *Insert Table pop up* similar to Microsoft Word, or through *Insert Table task pane* for more advanced table design task.



  For further information about how to insert table using task pane, please refer to Task Pane topic in WebTextEditor User Interface whitepaper.

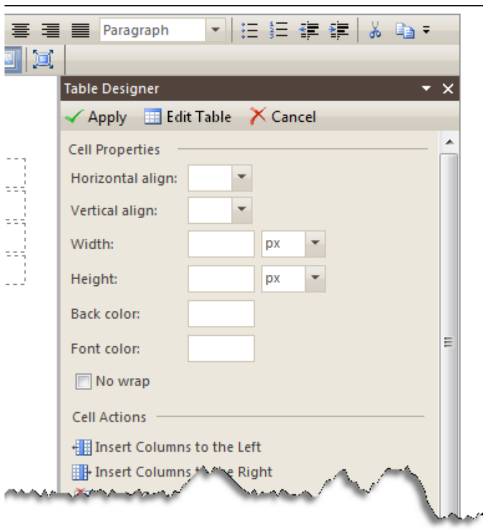
- **Edit Table**
  WebTextEditor also provides comprehensive table editing commands, such as listed below.
  1. Insert columns to the left
     Add new column to the left of selected cell.
  2. Insert columns to the right
     Add new column to the right of selected cell.
  3. Delete columns
     Delete selected column.
  4. Insert Rows Above
     Add new row to the above of selected cell.
  5. Insert Rows Below
     Add new row to the below of selected cell.
  6. Delete Rows
     Delete selected row.
  7. Merge Cells Down
     Merge selected cell with the cell positioned below it.
  8. Merge Cells Right
     Merge selected cell with the cell positioned at the right.

9. Split Cells Vertical
   Split selected cell vertically.
10. Split Cells Horizontal
    Split selected cell horizontally

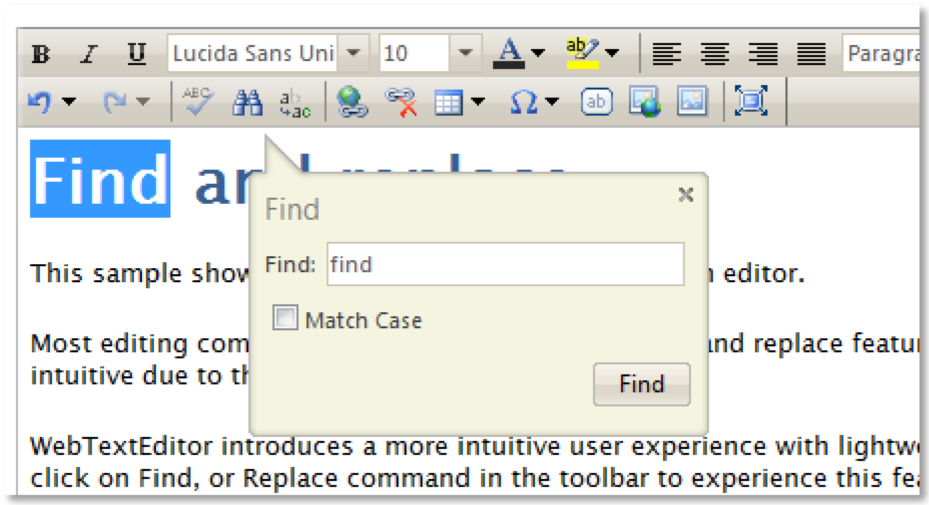


In order to perform table editing in WebTextEditor, user should select on a table's cell and open Table Designer task pane. Alternatively, context menu can also be used.

- **Edit Cell Properties**
  WebTextEditor provides more convenient way to edit cell properties in Table Designer Task Pane. The following are the cell properties that can be modified:
  1. Horizontal Align
  2. Vertical Align
  3. Width
  4. Height
  5. Back color
  6. Font color
  7. No wrap



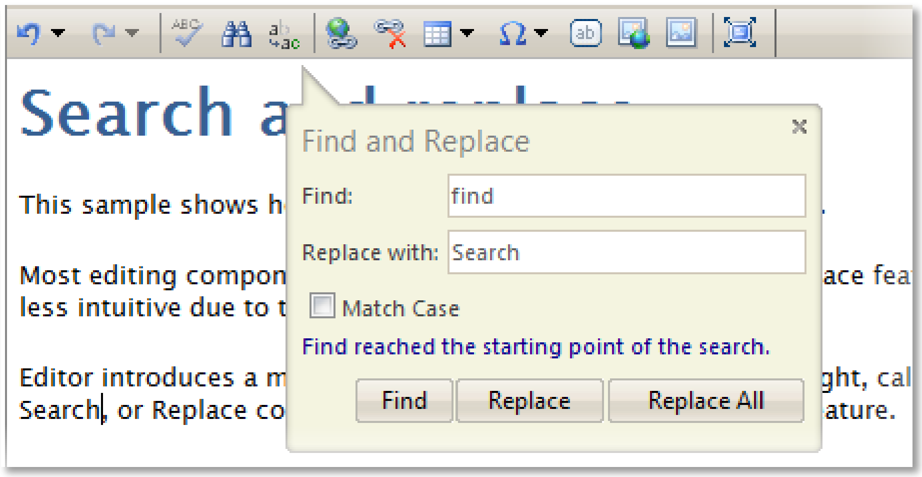
- **Find**
  WebTextEditor's unique search feature allows user to search in both design and html view. As in other commands, the Find command displays lightweight callout interface to provide a better searching experience. When a match is found, the text will be selected.



Find: find

☐ Match Case

- **Replace**
  Similar to Find feature, user can perform replace operation in both design view and html view. The replace function includes match case option, and also provides action to perform find, replace and replace all.



Find: find

Replace with: Search

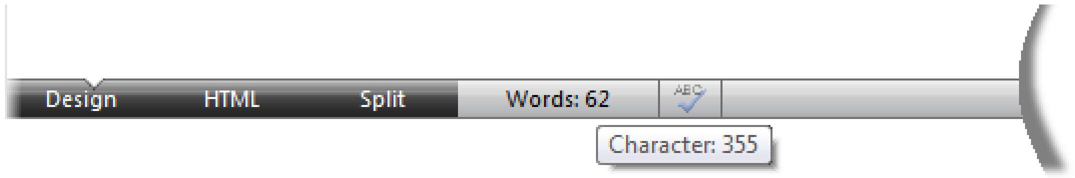
☐ Match Case

Find reached the starting point of the search.

- **Word and Character Count**
  Word count is enabled by default, while character count is disabled. To enable both word and character count, just set *ShowWordInformation* and *ShowCharacterInformation* properties to

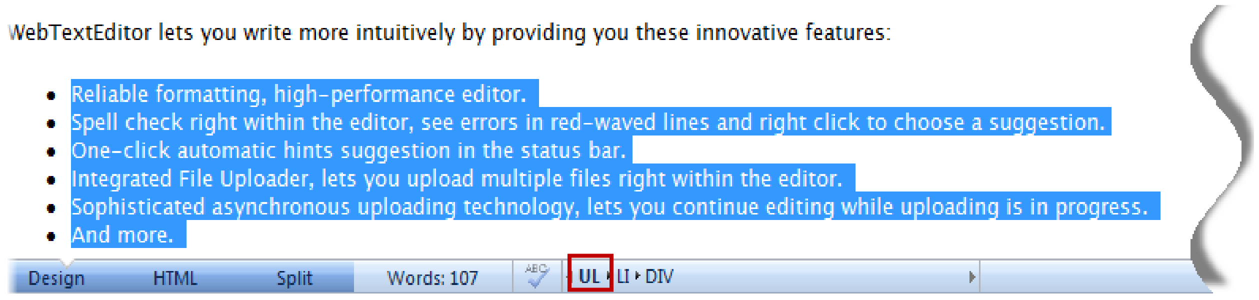
true.

Character count information and other details are available in the word information tooltip, thus making efficient use of screen real estate.



- **Html Inspector**
  WebTextEditor provides a capability to recognize selected text's html tag and its complete path. This information is displayed in the footer row. User can enable this feature by setting *EnableHTMLInspector* property to true.

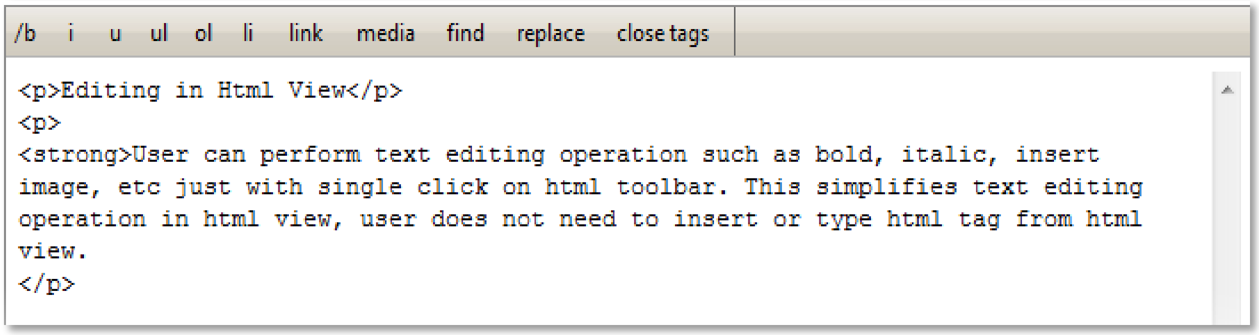
  Each html tag that displayed in HTML Inspector is clickable. When clicked, WebTextEditor will create a selection based on the clicked html tag.



WebTextEditor lets you write more intuitively by providing you these innovative features:

- Reliable formatting, high-performance editor.
- Spell check right within the editor, see errors in red-waved lines and right click to choose a suggestion.
- One-click automatic hints suggestion in the status bar.
- Integrated File Uploader, lets you upload multiple files right within the editor.
- Sophisticated asynchronous uploading technology, lets you continue editing while uploading is in progress.
- And more.

- **Editing in html view**
  User can perform basic text editing operation such as bold, italic, insert image, etc with just a single click on html toolbar. This simplifies text editing operation in html view as the html command automatically insert and close the tags, minimizing required typing effort.

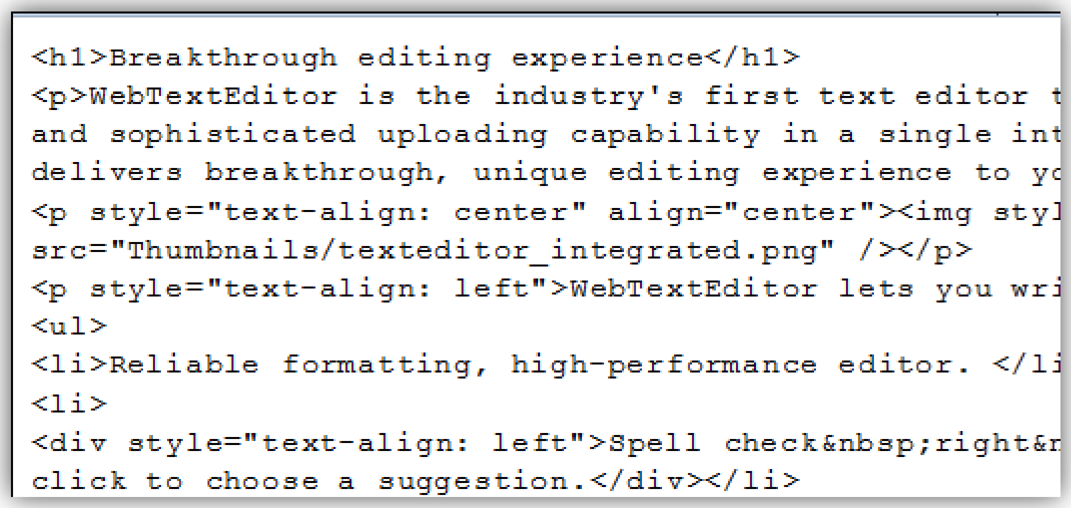


```
<p>Editing in Html View</p>
<p>
<strong>User can perform text editing operation such as bold, italic, insert
image, etc just with single click on html toolbar. This simplifies text editing
operation in html view, user does not need to insert or type html tag from html
view.
</p>
```

XHTML 1.1 Compliance Output

WebTextEditor produces only XHTML 1.1 compliance content regardless of browsers. This is made possible with WebTextEditor's editing architecture that seamlessly translates invalid tags and obsolete attributes into valid XHTML 1.1 markup.

WebTextEditor's XHTML engine is enabled by default and thus doesn't require additional configuration.



```
<h1>Breakthrough editing experience</h1>
<p>WebTextEditor is the industry's first text editor t
and sophisticated uploading capability in a single int
delivers breakthrough, unique editing experience to yo
<p style="text-align: center" align="center"><img styl
src="Thumbnails/texteditor_integrated.png" /></p>
<p style="text-align: left">WebTextEditor lets you wri
<ul>
<li>Reliable formatting, high-performance editor. </li
<li>
<div style="text-align: left">Spell check&nbsp;right&r
click to choose a suggestion.</div></li>
```

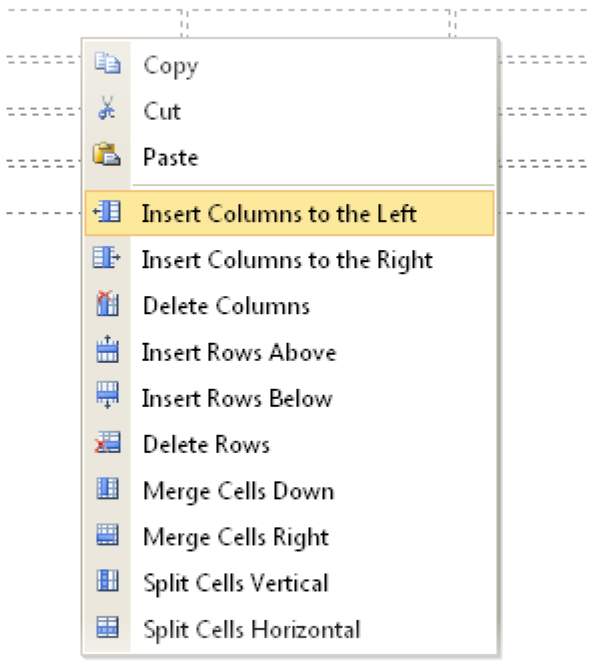
Enable Keyboard Support

WebTextEditor is designed to provide intuitive editing environment and rich editing experience which produces consistent results on all browsers. Since each browser has their own keyboard shortcut and unfortunately some of the editing operations do not generate the same results, WebTextEditor addresses this limitation with its own keyboard support.

The keyboard support is enabled by default in WebTextEditor. To disable keyboard support, simply set *EnableKeyboardSupport* property to false.

Enable Context Menu

Context menu gives users a more intuitive and natural way to work with editing and authoring tasks. Cut, copy, paste and table editing operation are available based on the selected object. Context menu is not enabled by default. User can enable context menu by setting *EnableContextMenu* property to true.



User can also customize context menu based on specific scenarios. Please refer to Context Menu topic in WebTextEditor User Interface whitepaper.

Auto Save

WebTextEditor introduces innovative feature which performs auto saving based on configured interval. Auto save operation will be invoked through AJAX callback and will be performed only when the document is modified. Developers can write their own codes to handle the auto saving operation in *Save* server-side event. For instance, the content can be saved to a file, database, or other storage.

Here are the steps to implement auto save in WebTextEditor:

1. Set *EnableAutoSave* property to true.
2. Set *AutoSaveInterval* property, default is 15 minutes.
3. Handle *OnSave* server side event.
4. Implement save mechanism in *OnSave* server side event.

```
protected void WebTextEditor1_Save(object sender, WebTextEditorSaveArgs e)
   {
       if (e.SaveAction == SaveAction.AutoSave)
       {
           StreamWriter sw = new
                 StreamWriter(HttpContext.Current.Server.MapPath("./SampleH
                 tml/AutoSave.html"));
           sw.Write(e.Content);
           sw.Flush();
           sw.Close();
       }
   }
```

Web.

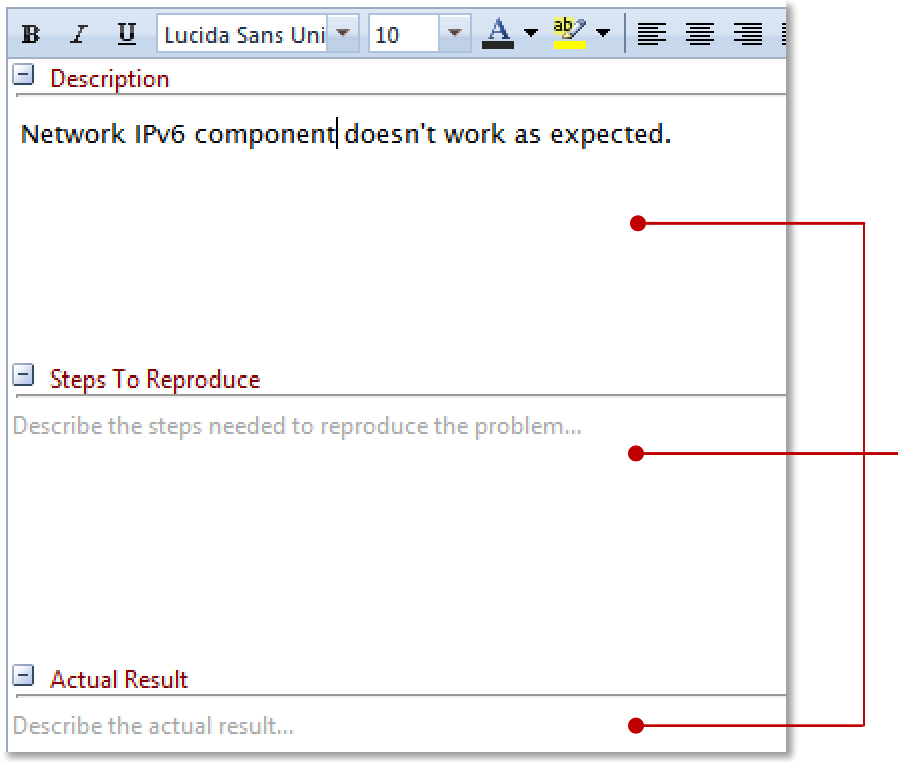
The above sample will perform auto save every 15 minutes and the content will be saved to file "AutoSave.html".

Sections

Unique to WebTextEditor is the capability to have more than one section in a single editor. This time-saving feature creates new possibilities and richer editing experiences that are previously difficult to achieve.

By default, WebTextEditor uses single section instead of multiple sections. Single section settings can be found in *RootTextEditor* property while multiple sections settings can be found in each *TextEditorSection* object.

To enable multiple sections, simple add the *TextEditorSection* object into *TextEditorSections* collection. When multiple sections are activated, the single section configuration such as in *RootTextEditor* will be automatically ignored.


B
I
U
Lucida Sans Uni
10
Description
Steps To Reproduce
Describe the steps needed to reproduce the problem...
Actual Result
Describe the actual result...

Network IPv6 component doesn't work as expected.

WebTextEditor makes editing more intuitive with multiple sections in a single editor instance. It makes efficient use of screen real estate which ultimately improves overall user experience.

There are several properties inside WebTextEditorSection such as:

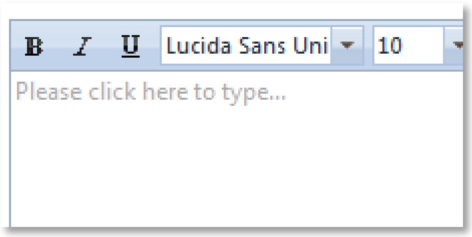
- **Content**
  Gets or sets the section's content.
- **Cssfiles**

This is a collection property; user can specify the custom CSS files to be loaded into each section. Please refer to Load Custom CSS File topic in WebTextEditor User Interface whitepaper.

- **Height**
  Specify the minimum height of section. This property is applicable only in multiple sections. Leave it empty for single section.
- **Name**
  Specify the name of section. This property is applicable only in multiple sections. Leave it empty for single section.
- **ReadOnly**
  Specify whether the section should be read only.
- **Text**
  Specify the caption text of the section. This property is applicable only in multiple sections. Leave it empty for single section.
- **WatermarkSettings**
  This is a property that handle watermark of sections.
  - **EnableWatermark**
    Specify whether or not watermark should be enabled. By default, watermark is disabled.
  - **WatermarkText**
    Specify the watermark text for the section.

Watermark

With watermark feature, WebTextEditor can display a default text before any user interaction occurs on an empty editor. The text can be customized for more user-friendly message such as “Type here for subject…”



Watermark can be customized in *WatermarkSettings* property in *RootTextEditor* or *WebTextEditorSection* object.

ReadOnly

WebTextEditor also can be set to non editable (read-only mode). Simply set the section’s *ReadOnly* property to true from server-side.

Read-only configuration can also be set from client-side using *SetReadOnly* client-side function. There are two parameters for this function: *isReadOnly(boolean)* and *editorSectionName(string).*

isReadOnly parameter determines whether or not the section should be read-only, while *editorSectionName* determines the section name where read-only configuration should be applied. The second parameter is needed when multiple sections is enabled only.

Save to HTML

WebTextEditor enables developers to easily save editor's content to HTML file through server side or client side method.

Server side

In the server side, WebTextEditor provides **SaveContentToFile** method which includes a handful of save options. The save options are provided to handle numerous common scenarios in saving operation.

By default, WebTextEditor will save only editor's content to the file. To save the content with full html declaration, please specify *saveFullHTML* parameter to true. Optionally, you can also specify the HTML document type declaration (i.e, XHTML DocType) by specifying the *docType* parameter. For further information, please refer to the following parameter information.

The following table lists the parameters of **SaveContentToFile** method:

Parameter name	Data Type	Description
filename	String	Specifies the complete file path and name of the saved file
sectionName	String	Determines which section's content of WebTextEditor to be saved. This parameter is used in multiple sections mode.
docType	WebTextEditorHtmlDocType	Specifies whether the content of WebTextEditor should be saved with HTML or XHTML doctype. If docType is specified, WebTextEditor will automatically add the selected doctype declaration to the saved file.
title	String	Determines the title of the saved file. This parameter can be used if docType is specified.
saveFullHTML	Boolean	Determines whether WebTextEditor should produce complete HTML content, which includes valid doctype declaration, opening <html>, and <body> and so on.

Client side

In client side, WebTextEditor provides **SaveContent** method which has a parameter to invoke save operation. The parameter is **sectionName(string)**.

Note that **SaveContent** does not save the content to a file in the client-side due to security permission. Instead, it invokes **Save** server-side event through FlyPostback (AJAX) where developers can implement their own saving logic in the server-side. It's essential to implement **Save** server-side event properly as the event is fired consistently in other scenarios, such as auto save, server-side direct save method call, and more.

The *sectionName* parameter of **SaveContent** method is applicable only in multiple sections mode. To save all sections, simply specify the *sectionName* parameter to **All** value.

Load from HTML

WebTextEditor supports loading content from an HTML file in the server-side through **LoadContentFromFile** method.

The **LoadContentFromFile** method provides several options such as listed in the table below.

Parameter name	Data Type	Description
filename	String	The HTML file to be loaded
sectionName	String	Determines the WebTextEditor's section which the content will be loaded to. This parameter is used in multiple sections mode.
isCompleteHTML	Boolean	Determines whether the loaded file is a complete valid HTML file.

TextSettings

All textual settings in WebTextEditor are customizable and can be found in *TextSettings* property. The following table lists all the customizable text settings:

Category	Text Setting Key	Description	Default Value
Callout	CallOutButtonApplyText	Specifies the text of Apply button in callout.	Apply
	CallOutButtonCancelText	Specifies the text of Cancel button in callout.	Cancel
	CallOutButtonFindText	Specifies the text of Find button in callout.	Find
	CallOutButtonInsertFromWebText	Specifies the text of Insert Media button in callout.	Insert Media
	CallOutButtonInsertLinkText	Specifies the text of Insert Link button in callout.	InsertLink
	CallOutButtonPasteText	Specifies the text of Paste button in callout.	Paste
	CallOutButtonReplaceAllText	Specifies the text of Replace All button in callout.	Replace All
	CallOutButtonReplaceText	Specifies the text of Replace button in callout.	Replace
	CallOutCheckBoxMatchCaseText	Specifies the text of Match Case	Match Case

		checkbox in callout.	
	CallOutFindText	Specifies the text of Find field name in callout.	Find:
	CallOutHyperlinkLinkText	Specifies the text of Link field name in callout.	Link:
	CallOutHyperlinkTargetText	Specifies the text of Target field name in callout.	Target:
	CallOutHyperlinkText	Specifies the text of Text field name in callout.	Text:
	CallOutMediaAlignmentText	Specifies the text of Caption Alignment field name in callout.	Alignment:
	CallOutMediaCaptionText	Specifies the text of Media Caption field name in callout.	Media Caption:
	CallOutMediaLinkURLText	Specifies the text of Link URL field name in callout.	Link URL:
	CallOutMediaSizeText	Specifies the text of Size field name in callout.	Size:
	CallOutMediaTitleText	Specifies the text of Media Title field name in callout.	Media Title:
	CallOutMediaTypeText	Specifies the text of Media Type field name in callout.	Media Type:
	CallOutMediaURLText	Specifies the text of Media URL field name in callout.	Media URL:
	CallOutMediaUseFrameText	Specifies the text of Use Frame checkbox in callout.	Use Frame
	CallOutPasteTitleText	Specifies the text of Paste Information title in callout.	Paste Information
	CallOutReplaceText	Specifies the text of Replace field name in callout.	Replace with:
Context menu	CopyText	Specifies the text of Copy menu item in context menu.	Copy
	CutText	Specifies the text of Cut menu item in context menu.	Cut
	PasteText	Specifies the text of Paste menu item in context menu.	Paste
Footer	DesignViewText	Specifies the text of Design view tab in footer row.	Design
	HtmlViewText	Specifies the text of HTML view tab in footer row.	HTML
	SplitViewText	Specifies the text of Split view tab in footer row.	Split
	PreviewText	Specifies the text of Preview view tab in footer row.	Preview
	ReadyText	Specifies the text displayed in status	Ready.

		when the editor is in Ready state.	
	CharacterInformationText	Specifies the text of character information in tooltip.	Character: {0}
	WordsInformationText	Specifies the text of word information in footer row.	Words: {0}
	SaveStatus	Specifies the text when auto save has been performed.	Saved at {0:g}.
TaskPane	TaskPaneHTMLViewText	Specifies the text of pane content when viewing in HTML view.	No action is available in this view.
Media Gallery	NoResultFoundText	Specifies the text when no result is found when searching the media in Media Gallery pane.	No results found.
	SearchInText	Specifies the text of Search In field name in Media Gallery pane.	Search in:
	ResultTypeText	Specifies the text of Result Type field name in Media Gallery pane.	Results should be:
	SearchingMediaText	Specifies the text displayed in status bar when media search action is being performed.	Searching medias…
Table Designer	SplitCellHorizontal	Specifies the text of Split Cell Horizontal item in Table Designer pane and context menu.	Split Cells Horizontal
	SplitCellVertical	Specifies the text of Split Cell Vertical item in Table Designer pane and context menu.	Split Cells Vertical
	TableAlignmentText	Specifies the text of Table Alignment field name in Table Designer pane.	Alignment:
	TableBackColorText	Specifies the text of Table Back Color field name in Table Designer pane.	Back color:
	TableBorderColorText	Specifies the text of Table Border Color field name in Table Designer pane.	Border color:
	TableBorderWidthText	Specifies the text of Table Border Width field name in Table Designer pane.	Border width:
	TableCaptionText	Specifies the text of Table Caption field name in Table Designer pane.	Caption:
	TableCellActionsText	Specifies the text of Cell Actions title in Table Designer pane.	Cell Actions
	TableCellPaddingText	Specifies the text of Cell Padding field name in Table Designer pane.	Cellpadding:
	TableCellPropertiesText	Specifies the text of Cell Properties title in Table Designer pane.	Cell Properties
	TableCellSpacingText	Specifies the text of Cell Spacing field name in Table Designer pane.	Cellspacing:

	TableColumnsText	Specifies the text of Columns field name in Table Designer pane.	Columns:
	TableCustomText	Specifies the text of Custom radio button in Table Designer pane.	Custom
	TableDesignerText	Specifies the text of Table Designer title.	Table Designer
	TableFontColorText	Specifies the text of Table Font Color field name in Table Designer pane.	Font color:
	TableHeightText	Specifies the text of Table Height field name in Table Designer pane.	Height:
	TableHorizontalAlignText	Specifies the text of Table Horizontal Align field name in Table Designer pane.	Horizontal align:
	TableNoWrapText	Specifies the text of No Wrap checkbox in Table Designer pane.	No wrap
	TablePropertiesText	Specifies the text of Table Properties title in Table Designer pane.	Table Properties
	TableRowsText	Specifies the text of Rows field name in Table Designer pane.	Rows:
	TableSummaryText	Specifies the text of Summary field name in Table Designer pane.	Summary:
	TableTemplateText	Specifies the text of Template radio button in Table Designer pane.	Template
	TableVerticalAlignText	Specifies the text of Vertical Align field name in Table Designer pane.	Vertical align:
	TableWidthText	Specifies the text of Table Width field name in Table Designer pane.	Width:
	EditTableLinkText	Specifies the text of Edit Table command in Table Designer toolbar.	Edit Table
	DeleteColumnText	Specifies the text of Delete Column item in Table Designer pane and context menu.	Delete Columns
	DeleteRowText	Specifies the text of Delete Row item in Table Designer pane and context menu.	Delete Rows
	InsertLeftColumnsText	Specifies the text of Insert Columns to the Left item in Table Designer pane and context menu.	Insert Columns to the Left
	InsertRightColumnsText	Specifies the text of Insert Columns to the Right item in Table Designer pane and context menu.	Insert Columns to the Right
	InsertRowAboveText	Specifies the text of Insert Rows Above item in Table Designer pane and context menu.	Insert Rows Above
	InsertRowsBelowText	Specifies the text of Insert Rows Below	Insert Rows Below

		item in Table Designer pane and context menu.	
	InsertTableText	Specifies the text of Insert Table command in Table Designer toolbar.	Insert Table
	MergeCellDown	Specifies the text of Merge Cell Down item in Table Designer pane and context menu.	Merge Cells Down
	MergeCellRight	Specifies the text of Merge Cell Right item in Table Designer pane and context menu.	Merge Cells Right
Form Control	InsertButtonText	Specifies the text of Insert Button item in Form Control pane.	Insert Button
	InsertFieldsetText	Specifies the text of Insert Fieldset item in Form Control pane.	Insert Fieldset
	InsertHorizontalRuleText	Specifies the text of Insert Horizontal Rule item in Form Control pane.	Insert Horizontal Rule
	InsertIFrameText	Specifies the text of Insert IFrame item in Form Control pane.	Insert IFrame
	InsertInputButtonText	Specifies the text of Insert Input Button item in Form Control pane.	Insert Input Button
	InsertInputCheckBoxText	Specifies the text of Insert Input CheckBox item in Form Control pane.	Insert Input CheckBox
	InsertInputFileUploadText	Specifies the text of Insert Input File Upload item in Form Control pane.	Insert Input File Upload
	InsertInputHiddenText	Specifies the text of Insert Input Hidden item in Form Control pane.	Insert Input Hidden
	InsertInputPasswordText	Specifies the text of Insert Password item in Form Control pane.	Insert Input Password
	InsertInputRadioText	Specifies the text of Insert Radio item in Form Control pane.	Insert Input Radio
	InsertInputResetText	Specifies the text of Insert Reset item in Form Control pane.	Insert Input Reset
	InsertInputSubmitText	Specifies the text of Insert Submit item in Form Control pane.	Insert Input Submit
	InsertInputText	Specifies the text of Insert Text item in Form Control pane.	Insert Input Text
	InsertMarqueeText	Specifies the text of Insert Marquee item in Form Control pane.	Insert Marquee
	InsertSelectDropdownText	Specifies the text of Insert Select Dropdown item in Form Control pane.	Insert Select Dropdown
	InsertSelectListboxText	Specifies the text of Insert Select Listbox item in Form Control pane.	Insert Select Listbox
	InsertTextAreaText	Specifies the text of Insert TextArea item in Form Control pane.	Insert TextArea

Mail Merge	PreviewMailMergeText	Specifies the text that will be displayed in status bar when the preview data is being fetched.	Preview Mail Merge…
	SendMailMergeText	Specifies the text that will be displayed in status bar when the emails are being sent.	Sending mail…
	FailedSendMailText	Specifies the text that will be displayed in status bar when the emails failed to be sent.	Failed sending to {0} recipient(s)
	SuccessSendMailText	Specifies the text that will be displayed in status bar when the emails have been successfully sent.	Send mail completed.

Client-side events

OnAfterExecCommand(`controlId, action`)
Gets or sets the event handler after execCommand is executed.

OnBeforeExecCommand(`controlId, action`)
Gets or sets the event handler before execCommand is executed.

OnEditorContextMenu(`controlId, menuObject`)
Gets or sets the event handler that will be invoked when the editor's context menu is about to be displayed.

OnInitialize(`controlId`)
Gets or sets the event handler when WebTextEditor is initialized.

OnSave(`controlId`)
Gets or sets the event handler that will be invoked the content of WebTextEditor content is being saved.

OnToolBarClick(`controlId, command, commandSection`)
Gets or sets the event handler that will be invoked when a command in toolbar is clicked.

OnResponse(`controlId, status`)
Gets or sets the event handler that will be invoked when a response from server is returned.

OnSendMailCompleted(`controlId, failed, failedEmails`)
Gets or sets the event handler that will be invoked when send mail action is completed.

Server-side events

OnMailMerge
Fires when mail merge is being processed in each data context.

OnMailMergeCompleted
Fires when mail merge process is completed.

OnBeforeSendMail
Fires before send mail action is performed.

OnSendMail
Fires when each email is being sent.

OnSendMailCompleted
Fires when sending email action is completed.

OnInitializeToolBar
Fires when the toolbar is initialized.

OnSave
Fires when the content is being saved.