TMS FMX RichEditor
DEVELOPERS GUIDE

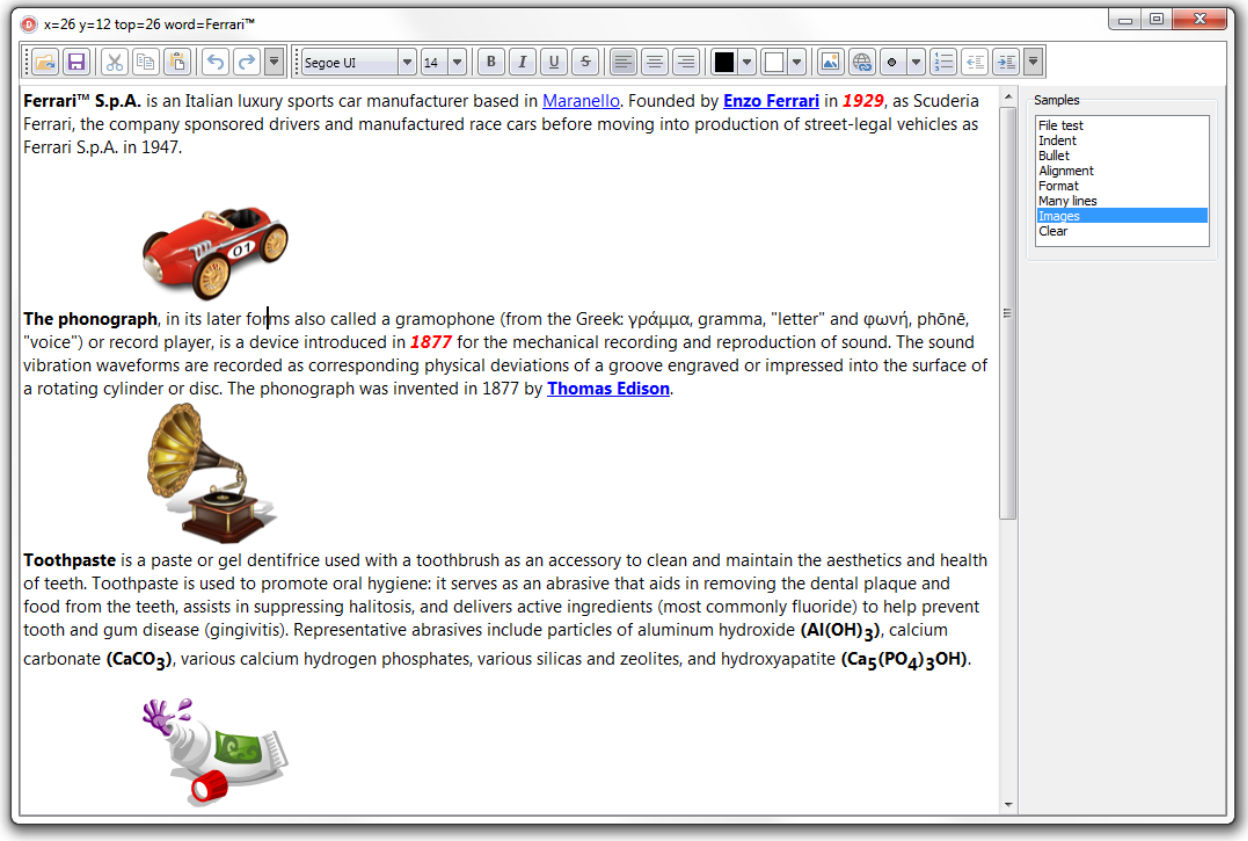
September 2019

Web: https://www.tmssoftware.com
Email: info@tmssoftware.com

Index

Description

TTMSFMXRichEditor is a compact light-weight WYSIWYG editor for formatted text. TTMSFMXRichEditor can include formatted text with bullets, hyperlinks, images, indenting, and aligned paragraphs. It offers functions for merging, highlighting text, find & replace, undo/redo, clipboard.
TTMSFMXRichEditor stores its text natively in the .RTE file format. It can load text from .TXT and .RTE files and can export to .TXT, .RTF, .HTML and .RTE files. Rich editing/formatting toolbars are included to perform clipboard functions, undo/redo, formatting, paragraph alignment, inserting bullets, pictures, hyperlinks, special characters.



Organization

The core component is TTMSFMXRichEditor. This is a standalone component that can be used as-is for WYSIWYG editing of formatted text. It comes with a formatting and editing toolbar that can be used to quickly setup a rich editor or its many predefined toolbar buttons/pickers

can be used to create a specific user interface around the TTMSFMXRichEditor according to your needs.

Internally the TTMSFMXRichEditor consists of a simple DOM. This DOM is a generic list of document elements. Different types of document elements exist such as a text element, image element, linebreak element, bullet element, … Each document element has several attributes that determine the appearance in the document. While the TTMSFMXRichEditor provides a large series of methods to add or remove elements from the DOM, it is also accessible via TTMSFMXRichEditor.Context.Content. It is recommended though that the API used instead of direct DOM manipulation.

Getting Started

Drop a TTMSFMXRichEditor on the form. The component with its default settings is ready for use. Entering of text can be done with default font & alignment. For ease of use, connect a TTMSFMXRichEditorEditToolBar or TTMSFMXRichEditorFormatToolBar, to apply all kinds of formatting to the text without writing any code or use its ribbon equivalents for a WYSIWYG editor with toolbar UI.

Properties & Events

Properties

Author	Sets the author of the document that will be persisted when saving to .RTE file format.
Color	Sets the default background of the TTMSFMXRichEditor
Comments	Sets comments for the document that will be persisted when saving to .RTE file format.
FontColor	Sets the default font color of the TTMSFMXRichEditor
GraphicSelection	Sets the appearance of the grips that appear when selecting graphics in the TTMSFMXRichEditor
GraphicSelection.BorderColor	Sets the border color of graphic item grips
GraphicSelection.Color	Sets the background color of graphic item grips
GraphicSelection.Style	Selects the style between rectangular or circular for the grips

HighlightColor	Sets the background color for highlighted text in the TTMSFMXRichEditor
HighlightTextColor	Sets the text color for highlighted text in the TTMSFMXRichEditor
LastModifiedBy	Sets the name of the person who last modified the content of the document and this name Is persisted in the .RTE file
ReadOnly	When true, the content of the document cannot be altered but selection is possible
SelectionColor	Sets the background color for selection in the TTMSFMXRichEditor
SelectionTextColor	Sets the text color for selection in the TTMSFMXRichEditor
Tags	Sets tags for the document that will be persisted when saving to .RTE file format.
URLColor	Sets the text color for hyperlinks in the TTMSFMXRichEditor
Version	Read-only property returning the version of the component

Events

OnCaretChanged	Event triggered whenever the caret changes in the TTMSFMXRichEditor
OnClick	Event triggered when the editor is clicked
OnClickHyperlink	Event triggered when a hyperlink is clicked in the editor. The URL for the hyperlink is returned as a parameter
OnDrawGraphic	Event triggered for drawing custom graphic elements in the TTMSFMXRichEditor. This event returns the canvas and rectangle where to draw the custom graphic and an ID for the graphic element
OnEnter	Event triggered when the TTMSFMXRIchEditor gets focus
OnEnterWord	Event triggered when one or more characters were entered before a word boundary. The event returns the word just entered
OnExit	Event triggered when the TTMSFMXRIchEditor looses focus

OnSelectionChanged	Event triggered whenever the selection changes in the TTMSFMXRichEditor

Methods

AddBullet(AType: TBulletType = btCircle);	Appends a bullet element to the TTMSFMXRichEditor and returns a bullet document element. The bullet types can be: - btSquare - btCircle - btArrow - btStar - btTick
AddGraphic(AWidth, AHeight: integer; AID: string);	Appends a graphical element with a specific ID to the TTMSFMXRichEditor and returns a graphic document element. This graphical element needs to be drawn via the OnDrawGraphic event
AddHyperlink(AValue, AURL: string);	Sets a hyperlink for the currently selected text in the TTMSFMXRichEditor
AddImage(FileName: string); overload;	Appends an image from file to the TTMSFMXRichEditor and returns a graphic document element
AddImage(FileName: string; AWidth, AHeight: integer); overload;	Appends an image from file with a specific width and height to the TTMSFMXRichEditor and returns a graphic document element
AddImage(Picture: TPicture); overload;	Appends an image to the TTMSFMXRichEditor and returns a graphic document element. Images of the type BMP,JPEG,GIF,PNG,ICO are supported.
AddImage(Picture: TPicture; AWidth, AHeight: integer); overload;	Appends an image with a specific width and height to the TTMSFMXRichEditor and returns a graphic document element. Images of the type BMP,JPEG,GIF,PNG,ICO are supported.
AddLineBreak: TREElement;	Appends a linebreak to the TTMSFMXRichEditor and returns a linebreak document element
AddMultiLineText(AValue: string);	Appends multiple lines of text as word-wrapped text in the TTMSFMXRichEditor
AddText(AValue: string): TTextElement;	Appends text to the TTMSFMXRichEditor and

overload;	returns a text document element containing this added text
AddText(AValue: string; AAlignment: TAlignment): TTextElement; overload;	Appends text with a specific alignment to the TTMSFMXRichEditor and returns a text document element containing this added text
AddText(AValue: string; AColor: TColor): TTextElement; overload;	Appends text with a specific text color to the TTMSFMXRichEditor and returns a text document element containing this added text
AddText(AValue: string; AColor: TColor; BkColor: TColor): TTextElement; overload;	Appends text with a specific text color and background color to the TTMSFMXRichEditor and returns a text document element containing this added text
AddText(AValue: string; AFont: TFont): TTextElement; overload;	Appends text with a specific font setting to the TTMSFMXRichEditor and returns a text document element containing this added text
AddText(AValue: string; AFontSize: integer; AFontName: string; AFontStyle: TFontStyles): TTextElement; overload;	Appends text with a specific font setting to the TTMSFMXRichEditor and returns a text document element containing this added text
AddText(AValue: string; AFontSize: integer; AFontName: string; AFontStyle: TFontStyles; AAlignment: TAlignment): TTextElement; overload;	Appends text with a specific font setting and alignment to the TTMSFMXRichEditor and returns a text document element containing this added text
BeginUpdate;	Use to block updates when doing many programmatic manipulations in the TTMSFMXRichEditor
CanRedo: boolean;	Returns true when a Redo operation is possible
CanUndo: boolean;	Returns true when an Undo operation is possible
CanUnindent: boolean;	Returns true when the selection in the document is indented (and thsu can be unindented)
Clear;	Removes all elements from the document
ClearSelection;	Clears the selection in the document
DeleteCaretElement;	Deletes the document element where the caret is
DeleteChar;	Deletes the character at caret position
DeleteSelected;	Deletes the selected element in case an image or graphical element is selected
DeleteSelection;	Deletes the selection in the TTMSFMXRichEditor

EndUpdate;	Use to block updates when doing many programmatic manipulations in the TTMSFMXRichEditor
FindFirst(AText: string; MatchCase: boolean = false): boolean;	Finds the first occurrence of text from the document origin
FindNext: boolean;	Finds the next occurrence of text from the position of the last find operation
GetSelectionBkColor: TColor;	Returns the background color for the selected text
GetSelectionBullet: TBulletType;	Returns the bullet type used for the selected text
GetSelectionFontName: string;	Returns the font face name for the selected text
GetSelectionFontSize: integer;	Returns the font size for the selected text
GetSelectionIndent: integer;	Returns the indent of the selected text
GetSelectionTextColor: TColor;	Returns the text color for the selected text
GetWordAndIndexAtCaret(var AValue: string; var AIndex: integer);	Returns the word at caret position and the index of the element containing the word
HasSelection: boolean;	Function returns true when there is a selection in the TTMSFMXRichEditor
Highlight(AText: string; MatchCase: boolean = false): boolean;	Highlight the text in the document with or without case sensitivity in the document
InsertBullet(AType: TBulletType = btCircle);	Inserts a bullet element at caret position in the TTMSFMXRichEditor and returns a bullet document element
InsertChar(ch: char);	Inserts a character at caret position
InsertFromStream(const AStream: TStream; f: double);	Inserts plain text from file at caret position
InsertGraphic(ID: string; AWidth, AHeight: integer);	Inserts a custom graphic element with a specific width and height at caret position in the TTMSFMXRichEditor and returns a graphic document element
InsertImage(FileName: string; AWidth: integer = 0; AHeight: integer = 0); overload;	Inserts an image with a specific width and height at caret position in the TTMSFMXRichEditor and returns an image document element
InsertImage(Picture: TPicture; AWidth: integer = 0; AHeight: integer = 0); overload;	Inserts an image with a specific width and height at caret position in the TTMSFMXRichEditor and returns an image document element

InsertMultiLineText(AValue: string);	Inserts text in the TTMSFMXRichEditor at caret position
InsertText(AValue: string): TTextElement; overload;	Inserts text in the TTMSFMXRichEditor at caret position and returns a text document element containing this added text
InsertText(Index: integer; AValue: string): TTextElement; overload;	Inserts text in the TTMSFMXRichEditor at document element Index and returns a text document element containing this added text
IsCaretInBulletList(var AType: TBulletType; var AIndex, AIndent: integer): boolean;	Returns true when the caret is within a list of bulleted items and when so, returns the bullet type, the index of the item in the list and the indent of the bulleted items
IsEmpty: boolean;	Returns true when the document is empty
IsSelectionBold: boolean;	Returns true when the selected text font style is bold
IsSelectionCenter: boolean;	Returns true when the selected text alignment is center aligned
IsSelectionItalic: boolean;	Returns true when the selected text font style is italic
IsSelectionLeft: boolean;	Returns true when the selected text alignment is left aligned
IsSelectionRight: boolean;	Returns true when the selected text alignment is right aligned
IsSelectionStrikeOut: boolean;	Returns true when the selected text font style is strikeout
IsSelectionSubscript: boolean;	Returns true when the selected text font style is subscript
IsSelectionSuperscript: boolean;	Returns true when the selected text font style is superscript
IsSelectionUnderline: boolean;	Returns true when the selected text font style is underline
LoadFromFile(const FileName: string);	Load a document from the .RTE file format
LoadFromStream(const AStream: TStream);	Load a document in the .RTE file format from stream
LoadFromTextFile(const FileName: string);	Loads the document from a plain text file
Merge(NamesAndValues: TStringList);	Performs merging of mergefields with merge values contained in the stringlist
PlainText: string;	Returns the text of the TTMSFMXRichEditor document as plaintext
property Caret: TCaret read FCaret write	Allows to get and set the caret based on

FCaret;	document elements and character position within the selected document element
property Selected: TREElement read FSelected write FSelected;	Get or set the selected (graphical) document element
property Selection: TSelection read FSelection write FSelection;	Allows to get and set the selection in the TTMSFMXRichEditor based on document elements for the selection start and selection end and character positions within the selections
Redo;	Performs Redo
ReplaceFirst(AText, AReplacement: string; MatchCase: boolean = false): boolean;	Replaces the first occurrence of text from the document origin
ReplaceNext: boolean;	Replaces the next occurrence of text from the position of the last find operation
SaveSelectionToStream(const AStream: TStream);	Saves the current selected document elements in .RTE file format to stream
SaveToFile(const FileName: string);	Save a document to the .RTE file format
SaveToStream(const AStream: TStream);	Save a document in the .RTE file format to stream
SaveToText(AFileName: string);	Saves the document in TTMSFMXRichEditor as plain text
ScrollToCaret;	Vertically scroll the TTMSFMXRIchEditor to make the caret visible
SelectAll;	Selects all document elements in TTMSFMXRichEditor
SelectedText: string;	Returns the selected text
SelectText(FromChar, ALength: integer);	Selects text in the TTMSFMXRichEditor based on character position of the text and length in characters
SelectWordAtCaret: string;	Selects the word in the TTMSFMXRichEditor document at caret position
SelectWordAtXY(X,Y: integer): string;	Selects the word in the TTMSFMXRichEditor document at mouse coordinates X,Y
SetSelectionAttribute(AAlignment: TAlignment); overload;	Sets the alignment of the selected text
SetSelectionAttribute(AError: boolean); overload;	Sets the selected text with red error underlining or remove error underlining
SetSelectionAttribute(AFont: TFont; AColor: TColor); overload;	Sets the font and color attribute of the seleted text
SetSelectionAttribute(AFont: TFont;	Sets the font, text color and background

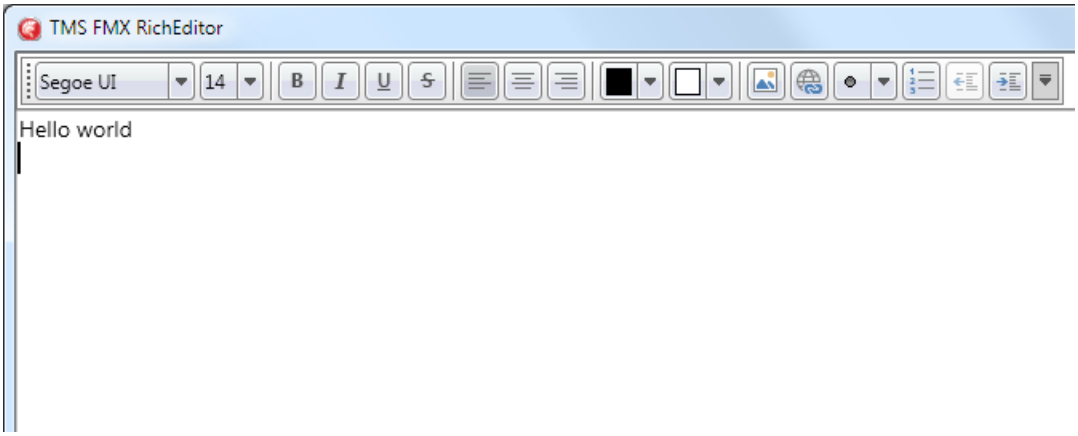
AColor: TColor; BkColor: TColor); overload;	color attribute of the seleted text
SetSelectionAttribute(AFontName: string; AFontSize: integer; AFontStyle: TFontStyles; AColor: TColor); overload;	Sets the font and color attribute of the seleted text
SetSelectionAttribute(AFontName: string; AFontSize: integer; AFontStyle: TFontStyles; AColor, BkColor: TColor); overload;	Sets the font, text color and background color attribute of the seleted text
SetSelectionBkColor(AColor: TColor);	Sets the background color of the selected text
SetSelectionBold(DoBold: boolean);	Sets the selected text bold or remove bold
SetSelectionBullets(AType: TBulletType); overload;	Sets bullets for the selected text. Each line separated by a linebreak gets a bullet. AType sets the bullet type
SetSelectionColor(AColor: TColor);	Sets the text color of the selected text
SetSelectionError(DoError: boolean);	Sets the selected text with red error underlining or remove error underlining
SetSelectionFontName(AName: string);	Sets the font face name for the selected text
SetSelectionFontSize(ASize: integer);	Sets the font size for the selected text
SetSelectionHighlight;	Sets the selected text in highlight text / background colors
SetSelectionHyperlink(AURL: string);	Sets a hyperlink for the text selected element in the document
SetSelectionIndent(AIndent: integer);	Sets the indent on the selected text
SetSelectionItalic(DoItalic: boolean);	Sets the selected text italic or remove italic
SetSelectionMergeField(AMergeName: string);	Defines a mergefield value for the selected text
SetSelectionStrikeOut(DoStrikeOut: boolean);	Sets the selected text strikeout or remove strikeout
SetSelectionSubscript(DoSubScript: boolean);	Sets the selected text subscript or remove subscript
SetSelectionSuperscript(DoSuperScript: boolean);	Sets the selected text superscript or remove superscript
SetSelectionUnderline(DoUnderline: boolean);	Sets the selected text underlined or remove underlined
Undo;	Performs Undo
UnHighlight;	Undo any previous highlight
UnSelect;	Undo any selection in the document
UpdateWordAndIndexAtCaret(AValue:	Replaces the word at document element at

string; AIndex: integer);	caret position at character index AIndex by AValue
WordAtCaret: string;	Returns the word at caret position
WordAtXY(X,Y: integer): string;	Returns the word at X,Y mouse coordinates
XYToCaret(X,Y: integer); overload;	Sets the caret at mouse X,Y coordinates
XYToCaret(X,Y: single); overload;	
XYToChar(X,Y: integer; el: TREElement; var CX,CY: integer): integer;	Converts the X,Y mouse coordinates to character position in the document text
XYToElement(X,Y: integer; var el: TREElement): boolean;	Retrieves the document element at mouse X,Y coordinates
XYToWord(X,Y: integer): string; overload;	Returns the word at mouse coordinates X,Y
XYToWord(X,Y: integer; el: TREElement): string; overload;	Returns the word and document element at mouse coordinates X,Y

Programmatic access to the document

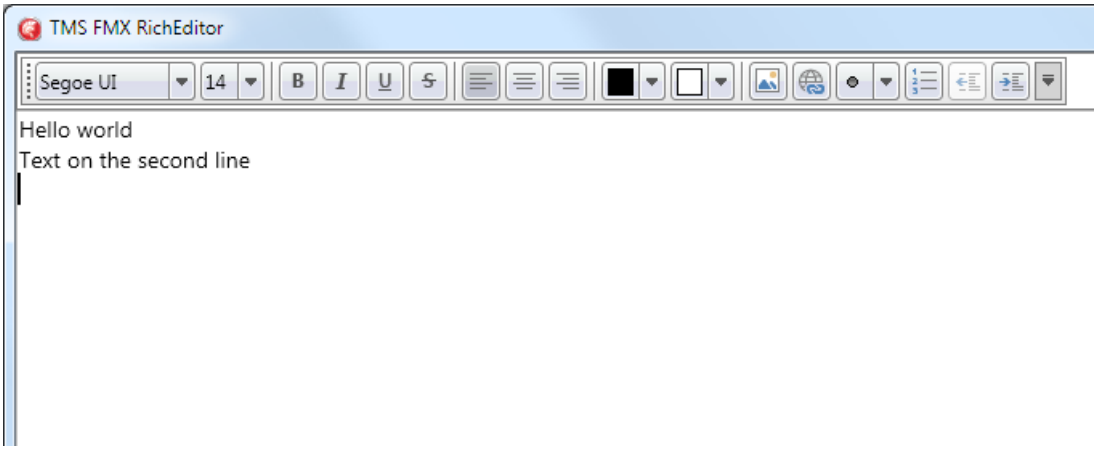
Text can be inserted in TTMSFMXRichEditor in various ways. To start with call:

```
TMSFMXRichEditor1.AddText('Hello world');
```



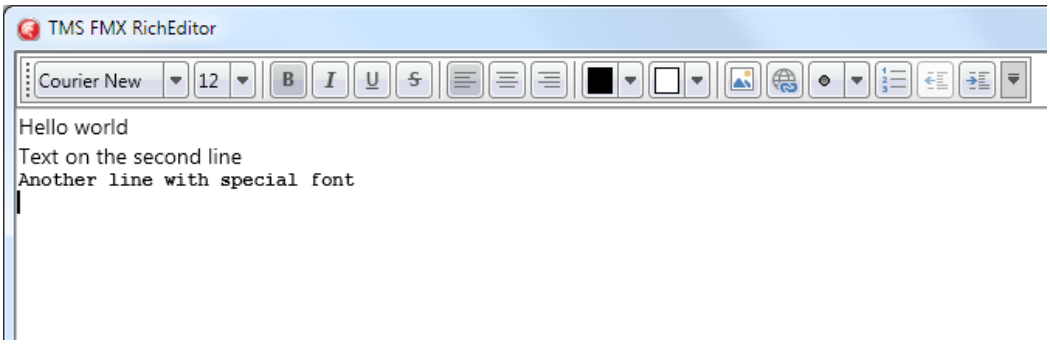
Add text on the next line with:

```
TMSFMXRichEditor1.AddLineBreak;
TMSFMXRichEditor1.AddText('Text on the second line');
```



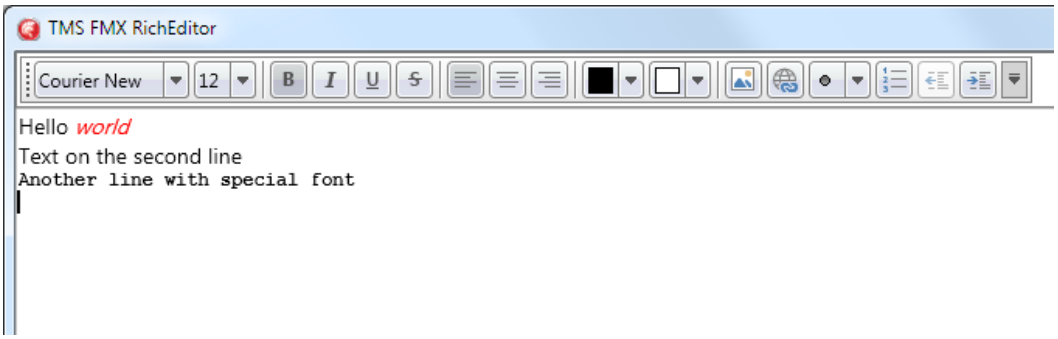
To add text with a different font than default font, use:

```
TMSFMXRichEditor1.AddLineBreak;
TMSFMXRichEditor1.AddText('Another line with special
font',12,'Courier',[fsBold]);
```



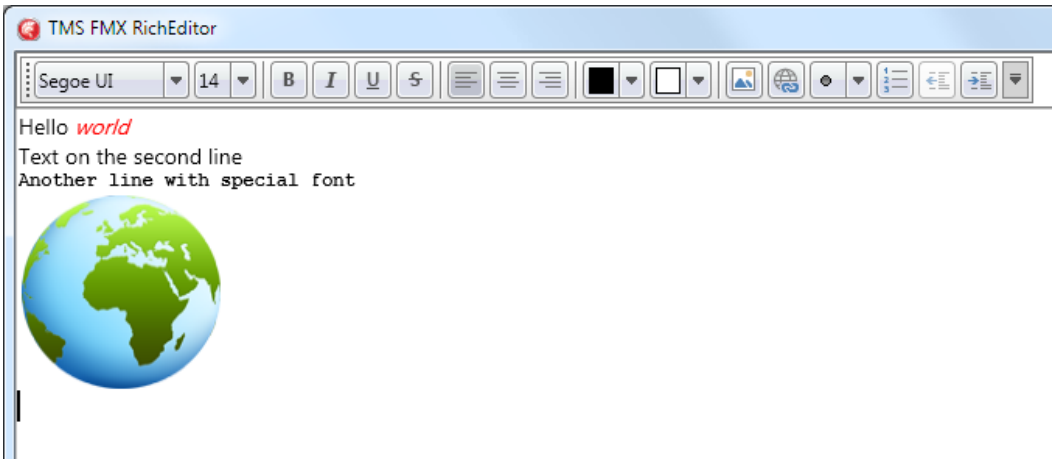
To change attributes of text in the TTMSFMXRichEditor, perform a selection based on index of the text and length. For example, to change the color of “world” on the first line, set a selection from character 6 for 5 characters (character index starts at zero) and set an attribute for the selection followed by remove the selection itself:

```
TMSFMXRichEditor1.SelectText(6,5);
TMSFMXRichEditor1.SetSelectionColor(claRed);
TMSFMXRichEditor1.SetSelectionItalic(True);
TMSFMXRichEditor1.ClearSelection;
```



To add images to the TTMSFMXRichEditor, use:

```
TMSFMXRichEditor1.AddImage('.\sample.png');
```


TMS FMX RichEditor
Segoe UI
14
Hello world
Text on the second line
Another line with special font

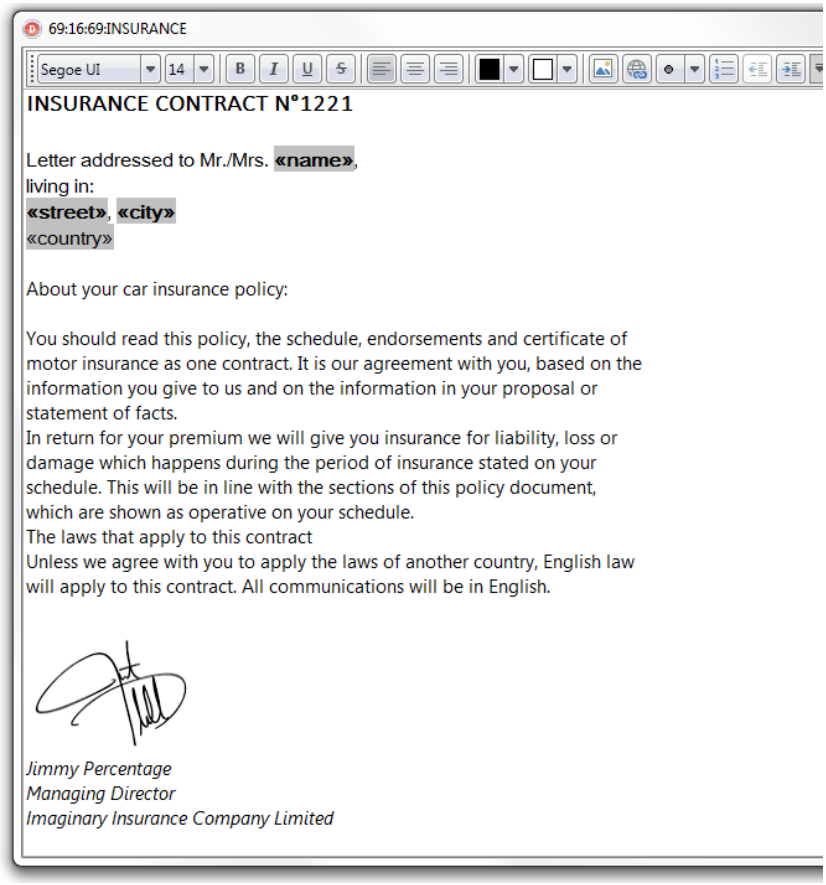
Using merge fields

Via merge fields, specific places in the document can be quickly replaced during a merge operation. To perform merging, first insert merge fields in the document. Merge fields are pieces of text that get a merge field name. These pieces of text are displayed between brackets «» and with a gray background. To set a piece of text as merge field, select the text and call

```
TMSFMXRichEditor1.SetSelectionMergeField('MergeFieldName');
```

Assume that following merge field names exist in the TTMSFMXRichEditor document:

'Name'
'Street'
'City'
'Country'



then a merge operation can be done in the following way:

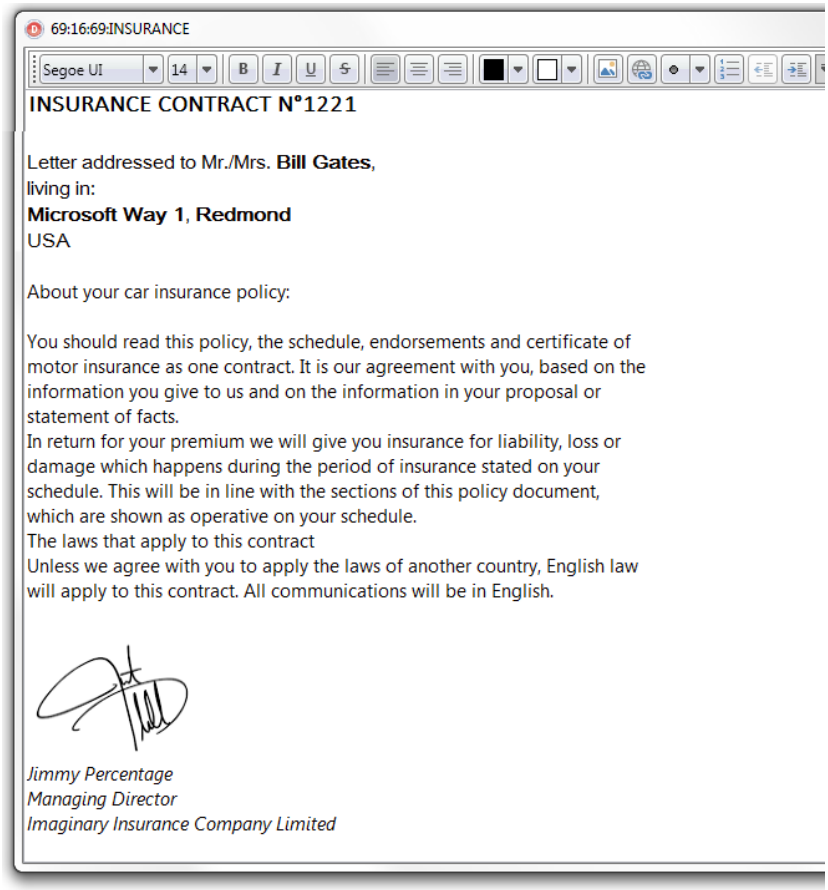
```
var
  sl: TStringList;

sl := TStringList.Create;
sl.Add('Name=Bill Gates');
sl.Add('Street=Microsoft Way 1');
sl.Add('City=Redmond');
sl.Add('Country=USA');

TMSFMXRichEditor1.Merge(sl);

sl.Free;
```

This will replace the merge fields Name, Street, City, Country with the values 'Bill Gates', 'Microsoft Way 1', 'Redmond', 'USA' specifically.



It is also possible to replace merge fields by pictures, i.e. insert pictures dynamically during a merge operation.

To do this, set a merge fieldname just like for text but using following construct for the mergelist:

Assume that in the previous example we want to add a picture of the person in the document, this would become:

'Photo'
'Name'
'Street'
'City'
'Country'

A merge operation is done the following way:

```
var
  sl: TStringList;
  pic: TBitmap;

pic := TBitmap.Create;
pic.LoadFromFile('billgates.jpg');

sl := TStringList.Create;
sl.AddObject('Photo=',pic);
sl.Add('Name=Bill Gates');
sl.Add('Street=Microsoft Way 1');
sl.Add('City=Redmond');
sl.Add('Country=USA');

TMSFMXRichEditor1.Merge(sl);

sl.Free;
pic.Free;
```

To undo the merge operation (and have the document ready for a new merge operation), simply call TMSFMXRichEditor1.UnMerge; after the merge operation.

Using accompanying toolbars

TTMSFMXRichEditor comes with 2 ready-to-use toolbars that enable to quickly create user-interfaces for manipulating the formatted text without writing code. To start using the toolbars, simple drop one of the toolbars on either a TTMSFMXDockPanel or directly on the form.

TTMSFMXRichEditorEditToolBar, TTMSFMXRichEditorFormatToolBar

These are 2 toolbars designed to be used in combination with a TTMSFMXDockPanel. The toolbars are divided in functions for Open/Save/Clipboard/Undo/Redo with the TTMSFMXRichEditorEditToolBar, changing font characteristics, alignment, bullets, indents, colors and inserting images, hyperlinks, special characters with the TTMSFMXRichEditorFormatToolbar.

Importing & exporting in rich text

TTMSFMXRichEditor comes with a component to allow to import or export its content in rich text (.RTF) files.
Performing such export or import is easy. Drop a TTMSFMXRichEditorRTFIO component on the form and connect the TTMSFMXRichEditor to this non-visual component's RichEditor property.

Export

Simply call:

```
TMSFMXRichEditorRTFIO.Save(FileName);
```

Import

Simply call:

```
TMSFMXRichEditorRTFIO.Load(FileName);
```

Importing & exporting in HTML format

TTMSFMXRichEditor comes with a component to allow exporting its content in HTML (.HTML) files. It is also possible to import from files that use a HTML subset (mini HTML) described here: http://www.tmssoftware.com/site/minihtml.asp

Performing such export or import is easy. Drop a TTMSFMXRichEditorHTMLIO component on the form and connect the TTMSFMXRichEditor to this non-visual component's RichEditor property.

Export

Simply call:

```
TMSFMXRichEditorHTMLIO.Save(FileName);
```

Notice that for HTML export, the default behaviour is that all images used in the document are exported as separate linked image files in the same folder where the .HTML file is generated. If it is preferred that images are generated in a different folder, use the 2nd default parameter ImagePath:

```
TMSFMXRichEditorHTMLIO.Save(FileName, ImagePath);
```

Import

This is limited to mini HTML files and import is done via the non-visual component TTMSFMXRichEditorMiniHTMLIO. In the same way as TTMSFMXRichEditorHTMLIO, assign the TTMSFMXRichEditor instance via TTMSFMXRichEditorMiniHTMLIO.RichEditor. The component provides the following overloads to import from HTML:

```
procedure Load(HtmlValue: string); overload;
procedure Load(FileName: string; Encoding: TEncoding = nil); overload;
procedure Load(AStream: TStream; Encoding: TEncoding = nil); overload;
```

This way, it can import from a simple HTML formatted string, a file with HTML formatted text or a stream. In the case of loading from a HTML formatting string, 1 extra parameter Pictures can be used as a container for referenced images in the HTML formatted string.

Finally, one more helper method is available in TTMSFMXRichEditorHTMLIO:

```
procedure Insert(AHtmlValue: string);
```

This inserts the formatted text from a HTML formatted string at caret position in the TTMSFMXRichEditor.

Exporting in PDF format

TTMSFMXRichEditor comes with a component to allow exporting its content in PDF (.PDF) files.

Performing such export is easy. Drop a TTMSFMXRichEditorPDFIO component on the form and connect the TTMSFMXRichEditor to this non-visual component's RichEditor property.

Export

Simply call:

```
TMSFMXRichEditorPDFIO.Save(FileName);
```

Alternatively the TMSFMXRichEditorPDFIO component is able to save to a stream. Simply call the Save method with a TStream instance.

Additionally, the TMSFMXRichEditorPDFIO component is capable of configuring the margins, header, footer as well as PDF meta-data such as the creator, author title and keywords. These properties are found under Options and Information.