Writing Your Own RTF Converter

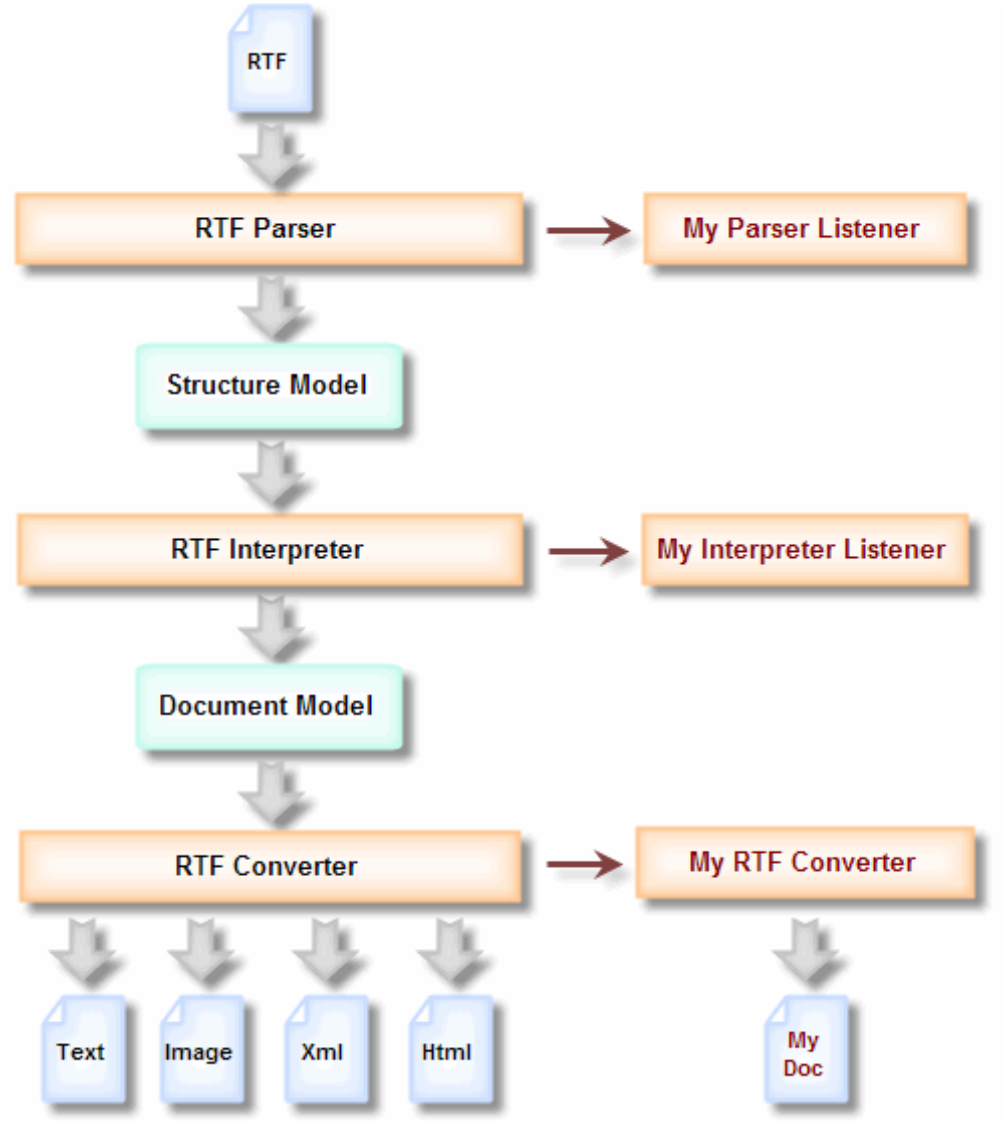
Jani Giannoudis,

1 Aug 2013 CPOL

An article on how to write a custom RTF parser and converter.

Download v1.7.0 - 2.5 MB

(Browse Code)



Introduction

In 1992, Microsoft introduced the Rich Text Format for specifying simple formatted text with embedded graphics. Initially intended to transfer such data between different applications on different Operating Systems (MS-DOS, Windows, OS/2, and Apple Macintosh), today this format is commonly used in Windows for enhanced editing capabilities (`RichTextBox`).

However, as soon as you have to work with such data in RTF format, additional wishes start to get strong:

- Simple extraction of text without consideration of any format information
- Extraction and conversion of embedded image information (scaled or unscaled)
- Conversion of the RTF layout and/or data into another format such as XML or HTML
- Transferring RTF data into a custom data model

The component introduced in this article has been designed with the following goals in mind:

- Support for the current RTF Specification 1.9.1
- Open Source C# code
- Unlimited usage in console, WinForms, WPF, and ASP.NET applications
- Independence of third party components
- Unicode support
- Possibility to analyze RTF data on various levels
- Separation of parsing and the actual interpretation of RTF data
- Extensibility of parser and interpreter
- Providing simple predefined conversion modules for text, images, XML, and HTML
- Ready-to-use RTF converter applications for text, images, XML, and HTML
- Open architecture for simple creation of custom RTF converters

Please keep the following shortcomings in mind:

- The component offers no high-level functionality to create RTF content
- The present RTF interpreter is restricted to content data and basic formatting options

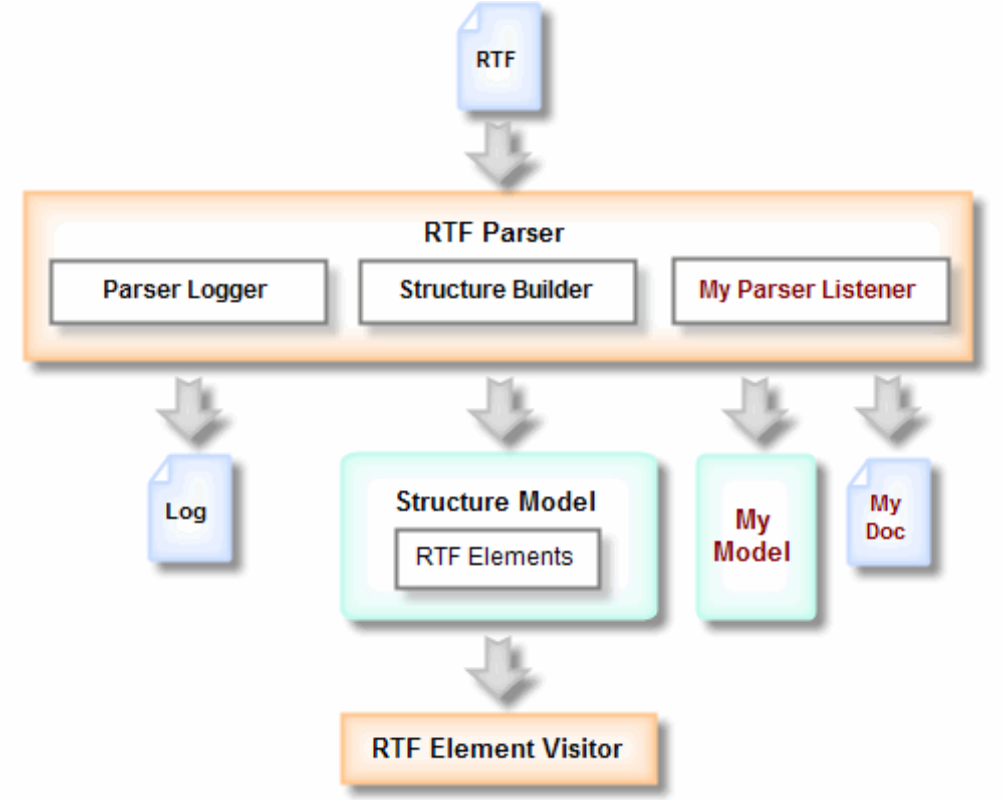
  There is no special support for the following RTF layout elements:

  - Tables
  - Lists
  - Automatic numbering
  - All features which require knowledge of how Microsoft Word might mean it ...

In general, this should not pose a big problem for many areas of use. A conforming RTF writer should always write content with readers in mind that they do not know about tags and features which were introduced later in the standards history. As a consequence, a lot of the content in an RTF document is stored several times (at least if the writer cares about other applications). This is taken advantage of by the interpreter here, which just simply focuses on the visual content. Some writers in common use, however, improperly support this alternate representation, which will result in differences in the resulting output.

Thanks to its open architecture, the RTF parser is a solid base for development of an RTF converter which focuses on layout.

RTF Parser



The actual parsing of the data is being done by the class `RtfParser`. Apart from the tag recognition, it also handles (a first level of) character encoding and Unicode support. The RTF parser classifies the RTF data into the following basic elements:

- RTF Group: A group of RTF elements
- RTF Tag: The name and value of an RTF tag
- RTF Text: Arbitrary text content (not necessarily visible!)

The actual parsing process can be monitored by `ParserListener`s (Observer Pattern), which offers an opportunity to react on specific events and perform corresponding actions.

The integrated parser listener `RtfParserListenerFileLogger` can be used to write the structure of the RTF elements into a log file (mainly intended for use during development). The produced output can be customized using its `RtfParserLoggerSettings`. The additional `RtfParserListenerLogger` parser listener can be used to log the parsing process to any `ILogger` implementation (see System functions).

The parser listener `RtfParserListenerStructureBuilder` generates the Structure Model from the RTF elements encountered during parsing. That model represents the basic elements as instances of `IRtfGroup`, `IRtfTag`, and `IRtfText`. Access to the hierarchical structure can be gained through the RTF group available in

`RtfParserListenerStructureBuilder.StructureRoot`. Based on the Visitor Pattern, it is easily possible to examine the structure model via any `IRtfElementVisitor` implementation:

Hide Shrink Copy Code

```
// ------------------------------------------------------------------------
public class MyVisitor : IRtfElementVisitor
{
    void RtfWriteStructureModel()
    {
      RtfParserListenerFileLogger logger =
         new RtfParserListenerFileLogger( @"c:\temp\RtfParser.log" );
      IRTFGroup structureRoot =
         RtfParserTool.Parse( @"{\rtf1foobar}", logger );
      structureRoot.Visit( this );
    } // RtfWriteStructureModel

    // ------------------------------------------------------------------------
    void IRtfElementVisitor.VisitTag( IRtfTag tag )
    {
      Console.WriteLine( "Tag: " + tag.FullName );
    } // IRtfElementVisitor.VisitTag

    // ------------------------------------------------------------------------
    void IRtfElementVisitor.VisitGroup( IRtfGroup group )
    {
      Console.WriteLine( "Group: " + group.Destination );
      foreach ( IRtfElement child in group.Contents )
      {
        child.Visit( this ); // recursive
      }
    } // IRtfElementVisitor.VisitGroup

    // ------------------------------------------------------------------------
    void IRtfElementVisitor.VisitText( IRtfText text )
    {
      Console.WriteLine( "Text: " + text.Text );
    } // IRtfElementVisitor.VisitText

} // MyVisitor
```

Note, however, that the same result for such simple functionality could be achieved by writing a custom `IRtfParserListener` (see below). This can, in some cases, be useful to avoid the overhead of creating the structure model in memory.

The utility class `RtfParserTool` offers the possibility to receive RTF data from a multitude of sources, such as `string`, `TextReader`, and `Stream`. It also allows, via its `IRtfSource` interface, to handle all these (and even other) scenarios in a uniform way.

The interface `IRtfParserListener`, with its base utility implementation `RtfParserListenerBase`, offers a way to react in custom ways to specific events during the parsing process:

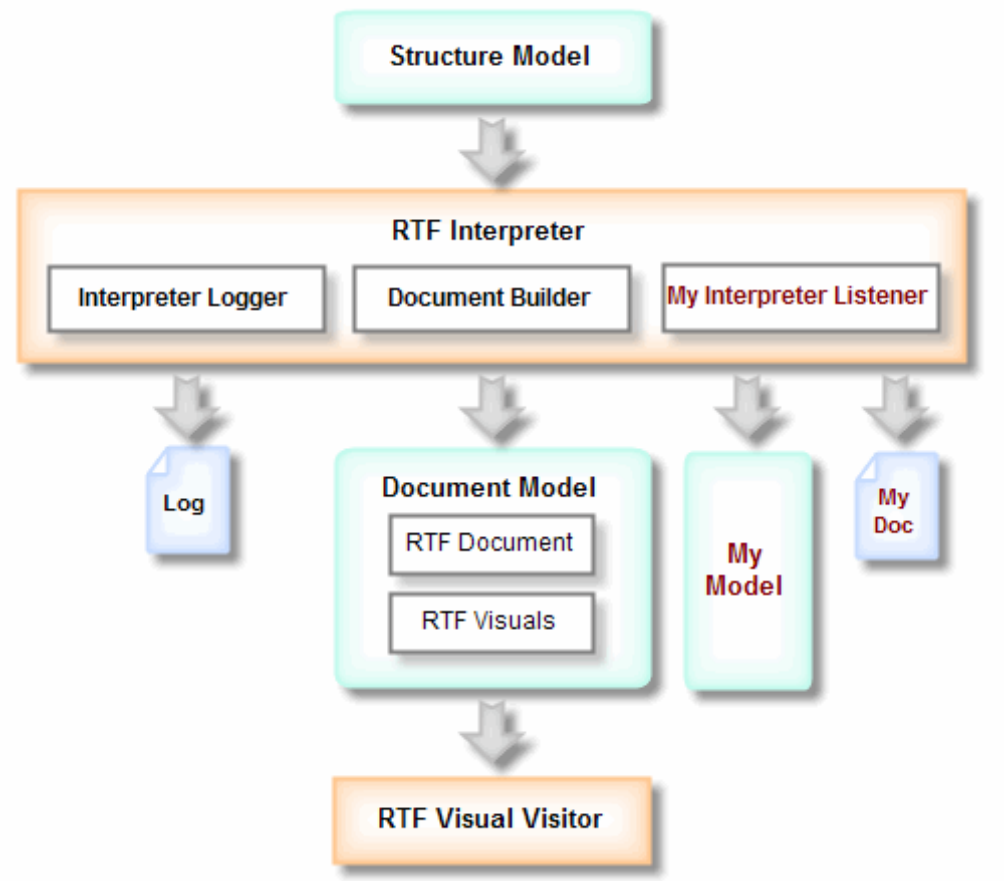
Hide Shrink Copy Code

```
// ------------------------------------------------------------------------
public class MyParserListener : RtfParserListenerBase
{
    // ------------------------------------------------------------------------
    protected override void DoParseBegin()
    {
      Console.WriteLine( "parse begin" );
    } // DoParseBegin

    // ------------------------------------------------------------------------
    protected override void DoGroupBegin()
    {
      Console.WriteLine( "group begin - level " + Level.ToString() );
    } // DoGroupBegin
```

```
    // ------------------------------------------------------------------------
    protected override void DoTagFound( IRtfTag tag )
    {
      Console.WriteLine( "tag " + tag.FullName );
    } // DoTagFound

    // ------------------------------------------------------------------------
    protected override void DoTextFound( IRtfText text )
    {
      Console.WriteLine( "text " + text.Text );
    } // DoTextFound

    // ------------------------------------------------------------------------
    protected override void DoGroupEnd()
    {
      Console.WriteLine( "group end - level " + Level.ToString() );
    } // DoGroupEnd

    // ------------------------------------------------------------------------
    protected override void DoParseSuccess()
    {
      Console.WriteLine( "parse success" );
    } // DoParseSuccess

    // ------------------------------------------------------------------------
    protected override void DoParseFail( RtfException reason )
    {
      Console.WriteLine( "parse failed: " + reason.Message );
    } // DoParseFail

    // ------------------------------------------------------------------------
    protected override void DoParseEnd()
    {
      Console.WriteLine( "parse end" );
    } // DoParseEnd

} // MyParserListener
```

Note that the used base class already provides (empty) implementations for all the interface methods, so only the ones which are required for a specific purpose need to be overridden.

RTF Interpreter



Once an RTF document has been parsed into a structure model, it is subject to interpretation through the RTF interpreter. One obvious way to interpret the structure is to build a **Document Model** which provides high-level access to the meaning of the document's contents. A very simple document model is part of this component, and consists of the following building blocks:

- Document info: title, subject, author, etc.
- User properties
- Color information
- Font information
- Text formats
- Visuals
    - Text with associated formatting information
    - Breaks: line, paragraph, section, page
    - Special characters: tabulator, paragraph begin/end, dash, space, bullet, quote, hyphen
    - Images

The various **Visuals** represent the recognized visible RTF elements, and can be examined with any `IRtfVisualVisitor` implementation.

Analogous to the possibilities of the RTF parser, the provided `RtfInterpreter` supports monitoring the interpretation process with `InterpreterListener`s for specific purposes.

Analyzing documents might be simplified by using the `RtfInterpreterListenerFileLogger` interpreter listener, which writes the recognized RTF elements into a log file. Its output can be customized through its `RtfInterpreterLoggerSettings`. The additional `RtfInterpreterListenerLogger` interpreter listener can be used to log the interpretation process to any `ILogger` implementation (see System functions).

Construction of the document model is also achieved through such an interpreter listener (`RtfInterpreterListenerDocumentBuilder`) which, in the end, delivers an instance of an `IRtfDocument`.

The following example shows how to make use of the high-level API of the document model:

Hide Shrink ▲ Copy Code

```
// ---------------------------------------------------------------------
void RtfWriteDocumentModel( Stream rtfStream )
{
    RtfInterpreterListenerFileLogger logger =
      new RtfInterpreterListenerFileLogger( @"c:\temp\RtfInterpreter.log" );
```

```
    IRtfDocument document = RtfInterpreterTool.BuildDoc( rtfStream, logger );
    RtfWriteDocument( document );
} // RtfWriteDocumentModel

// ------------------------------------------------------------------------
void RtfWriteDocument( IRtfDocument document )
{
    Console.WriteLine( "RTF Version: " + document.RtfVersion.ToString() );

    // document info
    Console.WriteLine( "Title: " + document.DocumentInfo.Title );
    Console.WriteLine( "Subject: " + document.DocumentInfo.Subject );
    Console.WriteLine( "Author: " + document.DocumentInfo.Author );
    // ...

    // fonts
    foreach ( IRtfFont font in document.FontTable )
    {
      Console.WriteLine( "Font: " + font.Name );
    }

    // colors
    foreach ( IRtfColor color in document.ColorTable )
    {
      Console.WriteLine( "Color: " + color.AsDrawingColor.ToString() );
    }

    // user properties
    foreach ( IRtfDocumentProperty documentProperty in document.UserProperties )
    {
      Console.WriteLine( "User property: " + documentProperty.Name );
    }

    // visuals (preferably handled through an according visitor)
    foreach ( IRtfVisual visual in document.VisualContent )
    {
      switch ( visual.Kind )
      {
        case RtfVisualKind.Text:
          Console.WriteLine( "Text: " + ((IRtfVisualText)visual).Text );
          break;
        case RtfVisualKind.Break:
          Console.WriteLine( "Tag: " +
                  ((IRtfVisualBreak)visual).BreakKind.ToString() );
          break;
        case RtfVisualKind.Special:
          Console.WriteLine( "Text: " +
                 ((IRtfVisualSpecialChar)visual).CharKind.ToString() );
          break;
        case RtfVisualKind.Image:
          IRtfVisualImage image = (IRtfVisualImage)visual;
          Console.WriteLine( "Image: " + image.Format.ToString() +
           " " + image.Width.ToString() + "x" + image.Height.ToString() );
          break;
      }
    }
} // RtfWriteDocument
```

As with the parser, the class `RtfInterpreterTool` offers convenience functionality for easy interpretation of RTF data and creation of a corresponding `IRtfDocument`. In case no `IRtfGroup` is yet available, it also provides for passing any source to the `RtfParserTool` for automatic on-the-fly parsing.

The interface `IRtfInterpreterListener`, with its base utility implementation `RtfInterpreterListenerBase`, offers the necessary foundation for a custom interpreter listener:

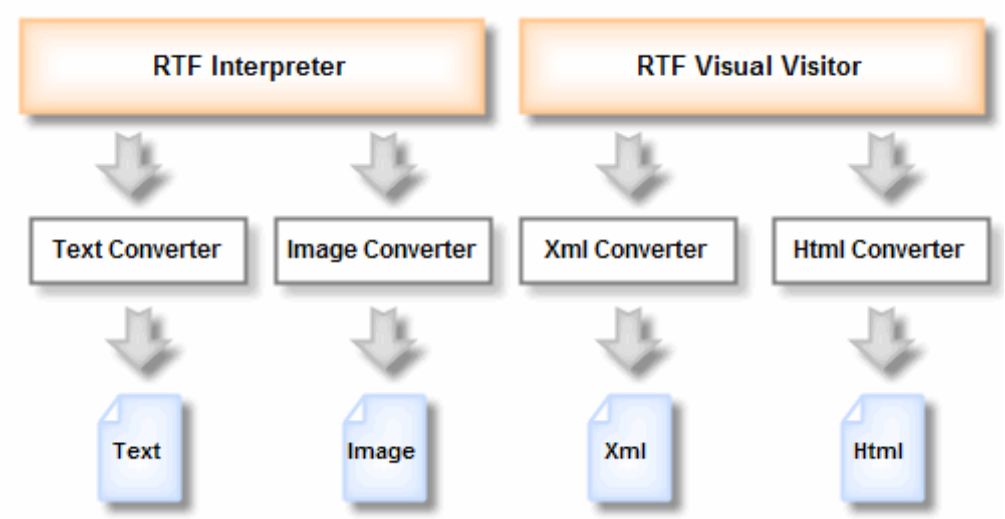

```
// ------------------------------------------------------------------------
public class MyInterpreterListener : RtfInterpreterListenerBase
```

```
{
    // ------------------------------------------------------------------------
    protected override void DoBeginDocument( IRtfInterpreterContext context )
    {
      // custom action
    } // DoBeginDocument

    // ------------------------------------------------------------------------
    protected override void DoInsertText( IRtfInterpreterContext context,
                                          string text )
    {
      // custom action
    } // DoInsertText

    // ------------------------------------------------------------------------
    protected override void DoInsertSpecialChar( IRtfInterpreterContext context,
                                                 RtfVisualSpecialCharKind kind )
    {
      // custom action
    } // DoInsertSpecialChar

    // ------------------------------------------------------------------------
    protected override void DoInsertBreak( IRtfInterpreterContext context,
                                           RtfVisualBreakKind kind )
    {
      // custom action
    } // DoInsertBreak

    // ------------------------------------------------------------------------
    protected override void DoInsertImage( IRtfInterpreterContext context,
      RtfVisualImageFormat format,
      int width, int height, int desiredWidth, int desiredHeight,
      int scaleWidthPercent, int scaleHeightPercent,
      string imageDataHex
    )
    {
      // custom action
    } // DoInsertImage

    // ------------------------------------------------------------------------
    protected override void DoEndDocument( IRtfInterpreterContext context )
    {
      // custom action
    } // DoEndDocument

} // MyInterpreterListener
```

The `IRtfInterpreterContext` passed to all of these methods contains the document information which is available at the very moment (colors, fonts, formats, etc.) as well as information about the state of the interpretation.

RTF Base Converters

As a foundation for the development of more complex converters, there are four base converters available for **text**, **images**, **XML**, and **HTML**. They are designed to be extended by inheritance.



Text Converter

The `RtfTextConverter` can be used to extract plain text from an RTF document. Its `RtfTextConvertSettings` determines how to represent special characters, tabulators, white space, breaks (line, page, etc.), and what to do with them.

Hide Copy Code

```
// ------------------------------------------------------------------------
void ConvertRtf2Text( Stream rtfStream )
{
    // logger
    RtfInterpreterListenerFileLogger logger =
      new RtfInterpreterListenerFileLogger( @"c:\temp\RtfInterpreter.log" );

    // text converter
    RtfTextConvertSettings textConvertSettings = new RtfTextConvertSettings();
    textConvertSettings.BulletText = "-"; // replace default bullet text 'Â°'

    RtfTextConverter textConverter = new RtfTextConverter( textConvertSettings );

    // interpreter
    RtfInterpreterTool.Interpret( rtfStream, logger, textConverter );
    Console.WriteLine( textConverter.PlainText );
} // ConvertRtf2Text
```

Image Converter

The `RtfImageConverter` offers a way to extract images from an RTF document. The size of the images can remain unscaled or as they appear in the RTF document. Optionally, the format of the image can be converted to another `ImageFormat`. File name, type, and size can be controlled by an `IRtfVisualImageAdapter`. The `RtfImageConvertSettings` determines the storage location as well as any scaling.

Hide Copy Code

```
// ------------------------------------------------------------------------
void ConvertRtf2Image( Stream rtfStream )
{
    // logger
    RtfInterpreterListenerFileLogger logger =
       new RtfInterpreterListenerFileLogger( @"c:\temp\RtfInterpreter.log" );

    // image converter
    // convert all images to JPG
    RtfVisualImageAdapter imageAdapter = new RtfVisualImageAdapter( ImageFormat.Jpeg );
    RtfImageConvertSettings imageConvertSettings =
            new RtfImageConvertSettings( imageAdapter );
    imageConvertSettings.ImagesPath = @"c:\temp\images\";
    imageConvertSettings.ScaleImage = true; // scale images
```

```
    RtfImageConverter imageConverter = new RtfImageConverter( imageConvertSettings );

    // interpreter
    RtfInterpreterTool.Interpret( rtfStream, logger, imageConverter );
    // all images are saved to the path 'c:\temp\images\'
} // ConvertRtf2Image
```

XML Converter

The `RtfXmlConverter` converts the recognized RTF visuals into an XML document. Its `RtfXmlConvertSettings` allows to specify the used XML namespace and the corresponding prefix.

Hide Copy Code

```
// ------------------------------------------------------------------------
void ConvertRtf2Xml( Stream rtfStream )
{
    // logger
    RtfInterpreterListenerFileLogger logger =
      new RtfInterpreterListenerFileLogger( @"c:\temp\RtfInterpreter.log" );

    // interpreter
    IRtfDocument rtfDocument = RtfInterpreterTool.BuildDoc( rtfStream, logger );

    // XML convert
    XmlWriterSettings xmlWriterSettings = new XmlWriterSettings();
    xmlWriterSettings.Indent = true;
    xmlWriterSettings.IndentChars = ( "  " );
    string fileName = @"c:\temp\Rtf.xml";
    using ( XmlWriter writer = XmlWriter.Create( fileName, xmlWriterSettings ) )
    {
      RtfXmlConverter xmlConverter = new RtfXmlConverter( rtfDocument, writer );
      xmlConverter.Convert();
      writer.Flush();
    }
} // ConvertRtf2Xml
```

HTML Converter

The `RtfHtmlConverter` converts the recognized RTF visuals into an HTML document. File names, type, and size of any encountered images can be controlled through an `IRtfVisualImageAdapter`, while the `RtfHtmlConvertSettings` determines storage location, stylesheets, and other HTML document information.

Hide Shrink Copy Code

```
// ------------------------------------------------------------------------
void ConvertRtf2Html( Stream rtfStream )
{
    // logger
    RtfInterpreterListenerFileLogger logger =
      new RtfInterpreterListenerFileLogger( @"c:\temp\RtfInterpreter.log" );

    // image converter
    // convert all images to JPG
    RtfVisualImageAdapter imageAdapter =
       new RtfVisualImageAdapter( ImageFormat.Jpeg );
    RtfImageConvertSettings imageConvertSettings =
                   new RtfImageConvertSettings( imageAdapter );
    imageConvertSettings.ScaleImage = true; // scale images
    RtfImageConverter imageConverter =
            new RtfImageConverter( imageConvertSettings );

    // interpreter
    IRtfDocument rtfDocument = RtfInterpreterTool.Interpret( rtfStream,
                                          logger, imageConverter );
```

```
    // html converter
    RtfHtmlConvertSettings htmlConvertSettings =
           new RtfHtmlConvertSettings( imageAdapter );
    htmlConvertSettings.StyleSheetLinks.Add( "default.css" );
    RtfHtmlConverter htmlConverter = new RtfHtmlConverter( rtfDocument,
                                                htmlConvertSettings );
    Console.WriteLine( htmlConverter.Convert() );
} // ConvertRtf2Html
```

HTML `Style`s can be integrated in two ways:

- Inline through `RtfHtmlCssStyle` in `RtfHtmlConvertSettings.Styles`
- Link through `RtfHtmlConvertSettings.StyleSheetLinks`

`RtfHtmlConvertScope` allows to restrict the target range:

- `RtfHtmlConvertScope.All`: complete HTML document (=Default)
- ...
- `RtfHtmlConvertScope.Content`: only paragraphs

RTF Converter Applications

The console applications Rtf2Raw, Rtf2Xml, and Rtf2Html demonstrate the range of functionality of the corresponding base converters, and offer a starting point for the development of our own RTF converter.

Rtf2Raw

The command line application Rtf2Raw converts an RTF document into plain text and images:

Hide Shrink ▲ Copy Code

```
Rtf2Raw source-file [destination] [/IT:format] [/BC:color] [/CE:encoding]
                    [/IS] [/XS] [/UI] [/ST] [/SI] [/LD:path] [/IDF] [/IUF] [/LP]
                    [/LI] [/D] [/O] [/HT] [/?]

      source-file       source rtf file
      destination       destination directory (default=source-file directory)
      /IT:format        images type format: bmp, emf, exif, gif, icon, jpg,
                              png, tiff or wmf (default=original)
      /BC:color         image background color name (default=none)
      /CE:encoding      character encoding: ASCII, UTF7, UTF8, Unicode,
                              BigEndianUnicode, UTF32, OperatingSystem (default=UTF8)
      /IS               image scale (default=off)
      /XS               extended image scale - border fix (default=off)
      /UI               enforce unscaled images (default=off)
      /ST               don't save text to the destination (default=on)
      /SI               don't save images to the destination (default=on)
      /LD:path          log file directory (default=destination directory)
      /IDF              ignore duplicated fonts (default=off)
      /IUF              ignore unknown fonts (default=off)
      /LP               write rtf parser log (default=off)
      /LI               write rtf interpreter log (default=off)
      /D                write text to screen (default=off)
      /O                open text in associated application (default=off)
      /HT               show hidden text (default=off)
      /?                this help

Samples:
     Rtf2Raw MyText.rtf
     Rtf2Raw MyText.rtf c:\temp
     Rtf2Raw MyText.rtf c:\temp /CSS:MyCompany.css
     Rtf2Raw MyText.rtf c:\temp /CSS:MyCompany.css,ThisProject.css
     Rtf2Raw MyText.rtf c:\temp /CSS:MyCompany.css,ThisProject.css /IT:png /BC:white
     Rtf2Raw MyText.rtf c:\temp /CSS:MyCompany.css,ThisProject.css /IT:png /BC:white
                                /LD:log /LP /LI
```

Rtf2Xml

The command line application Rtf2Xml converts an RTF document into an XML document:

Hide Copy Code

```
Rtf2Xml source-file [destination] [/CE:encoding] [/P:prefix]
                    [/NS:namespace] [/LD:path] [/IDF] [/IUF] [/LP] [/LI] [/HT] [/?]

      source-file       source rtf file
      destination       destination directory (default=source-file directory)
      /CE:encoding      character encoding: ASCII, UTF7, UTF8, Unicode,
                              BigEndianUnicode, UTF32, OperatingSystem (default=UTF8)
      /P:prefix         xml prefix (default=none)
      /NS:namespace     xml namespace (default=none)
      /LD:path          log file directory (default=destination directory)
      /IDF              ignore duplicated fonts (default=off)
      /IUF              ignore unknown fonts (default=off)
      /LP               write rtf parser log (default=off)
      /LI               write rtf interpreter log (default=off)
      /HT               show hidden text (default=off)
      /?                this help

Samples:
     Rtf2Xml MyText.rtf
     Rtf2Xml MyText.rtf c:\temp
     Rtf2Xml MyText.rtf c:\temp /NS:MyNs
     Rtf2Xml MyText.rtf c:\temp /LD:log /LP /LI
```

Rtf2Html

The command line application Rtf2Html converts an RTF document into an HTML document:

Hide Shrink ▲ Copy Code

```
Rtf2Html source-file [destination] [/CSS:names] [/IS] [/ID:path]
                     [/IT:format] [/BC:color] [/XS] [/CE:encoding] [/CS:charset]
                     [CS:mappings] [/DS:scope] [/IDF] [/IUF] [/SH] [/UI] [/SI] [/LD:path]
                     [/LP] [/LI] [/D] [/O] [/HT] [/NBS] [/CH] [/HP:pattern] [/?]

      source-file             source rtf file
      destination             destination directory (default=source-file directory)
      /CSS:name1,name2        style sheet names (default=none)
      /IS                     use inline styles (default=on)
      /ID:path                images directory (default=destination directory)
      /IT:format              images type format: jpg, gif or png (default=jpg)
      /BC:color               image background color name (default=none)
      /XS                     extended image scale - border fix (default=off)
      /CE:encoding            character encoding: ASCII, UTF7, UTF8, Unicode,
                                    BigEndianUnicode, UTF32, OperatingSystem (default=UTF8)
      /CS:charset             document character set used for the HTML header meta-tag
                                    'content' (default=UTF-8)
      /SC:mapping1,mapping2   special character mapping (default=none)
                                    mapping: special-character=replacement
                                    special characters: Tabulator, NonBreakingSpace,
                                    EmDash, EnDash, EmSpace, EnSpace, QmSpace,
                                    Bullet, LeftSingleQuote,
                                    RightSingleQuote, LeftDoubleQuote,
                                    RightDoubleQuote, OptionalHyphen, NonBreakingHyphen
      /DS:scope               document scope, comma separated list of document sections:
                                    doc, html, head, body, content, all (default=all)
      /IDF                    ignore duplicated fonts (default=off)
      /IUF                    ignore unknown fonts (default=off)
      /SH                     don't save HTML to the destination (default=on)
      /SI                     don't save images to the destination (default=on)
      /UI                     enforce unscaled images (default=off)
      /LD:path                log file directory (default=destination directory)
      /LP                     write rtf parser log file (default=off)
```

```
      /LI                       write rtf interpreter log file (default=off)
      /D                        display HTML text on screen (default=off)
      /O                        open HTML in associated application (default=off)
      /NBS                      use non-breaking spaces (default=off)
      /HT                       show hidden text (default=off)
      /CH                       convert visual hyperlinks (default=off)
      /HP:pattern               regular expression pattern to recognize visual hyperlinks
      /?                        this help

Samples:
     Rtf2Html MyText.rtf
     Rtf2Html MyText.rtf /DS:body,content
     Rtf2Html MyText.rtf c:\temp
     Rtf2Html MyText.rtf c:\temp /CSS:MyCompany.css
     Rtf2Html MyText.rtf c:\temp /CSS:MyCompany.css,ThisProject.css
     Rtf2Html MyText.rtf c:\temp /CSS:MyCompany.css,ThisProject.css
                                 /ID:images /IT:png /BC:white
     Rtf2Html MyText.rtf c:\temp /CSS:MyCompany.css,ThisProject.css
                                 /ID:images /IT:png /BC:white /LD:log /LP /LI
     Rtf2Html MyText.rtf c:\temp /SC:Tabulator=&gt;,Bullet=:
```

RTF to HTML Example

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.

Inline, font type and weight, and a List

- ListItem 1
- ListItem 2
- ListItem 3

Bolded

Underlined

Bolded and Underlined

Italic

left aligned

centered

right aligned

Sample RTF to HTML Converter - RTF Input

RTF to HTML Example

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.

Inline, font type and weight, and a List

- ListItem 1
- ListItem 2
- ListItem 3

Bolded
Underlined
Bolded and Underlined
Italic
left aligned
centered
right aligned

Sample RTF to HTML Converter - HTML Output

Projects

The following projects are provided in the RTF converter component:

Sys	System functions. See below for a short description.
Parser	Parsing of RTF data.
ParserTests	Unit tests for parser.
Interpreter	Interpretation of parsed RTF data. Functionality for conversion of RTF to plain text and images.
InterpreterTests	Unit tests for interpreter.
ConverterXml	Functionality for conversion of RTF to XML.
ConverterHtml	Functionality for conversion of RTF to HTML.
Rtf2Raw	Command line application to convert from RTF to text- and image data.
Rtf2Xml	Command line application to convert from RTF to XML.
Rtf2Html	Command line application to convert from RTF to HTML.
RtfWinForms	Windows Forms sample application which demonstrates a simple conversion from RTF to Text/XML/HTML.
RtfWindows	WPF sample application which demonstrates a simple conversion from RTF to Text/XML/HTML.

System Functions

The project *Sys* contains some base functionality for use in all the other applications/projects:

HashTool	Functions to simplify implementing overrides of the `object.GetHashCode()` method.
StringTool	Functions for `String` formatting.
CollectionTool	Functions to simplify handling of collections.
Application Arguments	Functions to interpret command line arguments. Offers support for the argument types `ToggleArgument`, `ValueArgument`, and `NamedValueArgument`.
Logging	Functionality to abstract the embedding of a Logger facility. Supports the logger types `LoggerNone`, `LoggerTrace`, and `LoggerLog4net`.
Test	Functionality to build a unit-based test application.

Acknowledgement

Special thanks to Leon Poyyayil for the design and his support and contribution in the development of this component.

History

- 1th August, 2013 - v1.7.0.0
  - `RtfHtmlConverter`: Replaced align attributes with style attributes
- 20th January, 2013 - v1.6.0.0
  - `RtfHtmlConverter`: Added support for HTML tags superscript and subscript
  - `RtfImageBuilder.VisitTag`: Fixed negative image dimensions by enforcing the absolute value
  - `RtfFontTableBuilder`: Added option to ignore duplicated fonts
  - `RtfInterpreter`: Added option to ignore unknown fonts
  - `RtfInterpreterSettings`: New class to customize the interpreter
  - `RtfInterpreter`/`RtfInterpreterTool`: Added support fo the interpreter settings
  - `Rtf2Xml`/`Rtf2Raw`/`Rtf2Html`: New command line argument */IDF* to ignore duplicated fonts
  - `Rtf2Xml`/`Rtf2Raw`/`Rtf2Html`: New command line argument */IUF* to ignore unknown fonts
  - `Rtf2Raw`/`Rtf2Html`: New command line argument */UI* to enforce unscaled images
- 12th September, 2012 - v1.5.0.0
  - `ToggleArgument`: Fixed parsing of the off-toggle
  - `RtfInterpreter`: Allowed font with zero size
  - `RtfEmptyHtmlStyleConverter`: New class with empty style converter
  - `Rtf2Html`: New command line argument */IS* to suppress inline styles using `RtfEmptyHtmlStyleConverter`
- 3rd March, 2011 - v1.4.0.0
  - `RtfTextFormat`: Fixed reset of super script in `DeriveWithSuperScript`
- 26th April, 2011 - v1.3.0.0
  - `RtfVisualImageAdapter`: Relaxed handling for non-hex image data
- 8th April, 2011 - v1.2.0.0
  - `RtfFontBuilder`: Relaxed handling for missing font names, generating font name '*UnnamedFont_{fond-id}*'

- 14th February, 2011
  - Replaced `RtfHmltCpecialCharConverter` with `RtfHtmlSpecialCharCollection`
  - `RtfHtmlConverter`: New property `SpecialCharacters`
- 25th January, 2011
  - RTF Interpreter: Fixed retroactive paragraph changes
- 1st December, 2010
  - `RtfHtmlSpecialCharConverter`: New class to handle special character conversion
  - `RtfHtmlConverterSettings`: New property `SpecialCharsRepresentation`
  - `RtfHtmlConverter`: Added support for special character conversion
  - `Rtf2Html`: New command line argument */SC* to control the special character conversion
  - Removed projects and solutions for Visual Studio 2005
  - Added projects and solutions for Visual Studio 2010
- 20th August, 2009
  - `RtfHtmlConverterSettings`: New property `UseNonBreakingSpaces`
  - `Rtf2Html`: New command line argument */NBS* to replace spaces with non-breaking spaces (default=off)
- 18th August, 2009
  - Signed assemblies
  - `RtfImageConverterSettings`: New property `BackgroundColor`
  - `Rtf2Raw`, `Rtf2Html`: New command line argument */BC* for the image background color
- 3rd August, 2009
  - `RtfHtmlConverter`: Replacing text spaces with non-breaking-spaces
  - `RtfImageConverter`: Fixed missing converted image info in case of undefined target format
- 20th May, 2009
  - `RtfHtmlConverter`: Added support for `Justify` alignment
  - `RtfHtmlConverter`: Fixed missing closing tag `</ul>` for bulleted lists, in cases when `ConvertScope` is set to `Content`
- 5th May, 2009
  - `RtfSpec`: New tag `highlight` for highlighted text
  - `RtfInterpreter`: Added support for text highlighting
  - `Rtf2Raw`, `Rtf2Xml`, `Rtf2Html`: Enumeration `ProgramExitCode` contains all program exit codes
- 20th February, 2009
  - `RtfParser`: Various new specialized exceptions based on `RtfParserException`
  - `RtfInterpreter`: Various new specialized exceptions based on `RtfInterpreterException`
  - Projects *Sys*, *Parser*, and *Interpreter*: Extracted localizable strings to *Strings.cs* and *Strings.resx*
  - `Rtf2Html`: New command line argument */DS* to control the conversion scope
- 18th February, 2009
  - `RtfSpec`: New tag `nonshppict` for alternative images
  - `RtfInterpreter`: Ignoring alternative images
  - `RtfInterpreterTest`: New unit-test for alternative images
- 16th February, 2009
  - `RtfImageConvertSettings`: New properties `ScaleOffset` and `ScaleExtension` to control image scaling
  - `Rtf2Raw` and `Rtf2Html`: New command line argument */XS* to fix the BorderBug
  - Changed binaries from debug to release (slightly better performance)

- 5th February, 2009
  - `RtfInterpreter`: Extended group handling to recognize state transition from header to document in case no other mechanism detects it and the content starts with a group with such a 'destination'
- 3rd February, 2009
  - `Rtf2Html`: Added support to convert visual hyperlinks
  - New command line argument */CH* to convert visual hyperlinks (default is off)
  - New command line argument */HP:pattern* with the Regular Expression pattern to recognize visual hyperlinks (optional)
  - Programmatic control with `RtfHtmlConvertSettings.ConvertVisualHyperlinks` and `RtfHtmlConvertSettings.VisualHyperlinkPattern`
  - Only visible hyperlinks will be converted; does not support hyperlinks which are represented by a display text and the actual hyperlink stored in a (hidden) field content
  - Refactored code - or rather 'ReSharped' Smile |
- 22nd October, 2008
  - `RtfParser`: Fixed to properly handle skipping of Unicode alternative representation in case these are written in hex-encoded form
  - `RtfHtmlConverter`: New property `DocumentImages` which provides information about the converted images using `IRtfConvertedImageInfo`
  - Added *ChangeHistory.txt*
- 15th October, 2008
  - Added support for tags
    - *sub*: Changes font size to 2/3 of the current font size and moves 'down' by half the current font size
    - *super*: Changes font size to 2/3 of the current font size and moves 'up' by half the current font size
    - *nosupersub*: Resets the 'up'/'down' baseline alignment to zero; **Attention**: this leaves the font size unchanged as it is not known by the current implementation what the 'previous' font size was; hence, depending on the RTF-writer, this might lead to content that is displayed with a smaller font size than intended
    - *v**: Toggles the new `IsHidden` property of `IRtfTextFormat`; *v* and *v1* turn it on while *v0* turns it off (according to the behavior or 'boolean tags')
    - *viewkind*: Triggers the transition from interpreter state `InHeader` to `InDocument` (but only if the font table is already defined); this supports documents without color table and prevents formatting or content at the beginning from being ignored
  - Extended/fixed support for tags
    - *dn* and *up*: will use the specified default value of '6' if none is given in the RTF (instead of resetting to zero)
  - `RtfTextConverterSettings`/ `RtfXmlConverterSettings`/ `RtfHtmlConverterSettings`: have a new flag `IsShowHiddenText` which defaults to `false`
  - `RtfTextConverter`/ `RtfXmlConverter`/ `RtfHtmlConverter`: will only append found text to the plain text buffer if it is not marked hidden in its text format or if the new setting `IsShowHiddenText` is explicitly set to `true`
  - `Rtf2Raw/Rtf2Xml/Rtf2Html`: New command-line argument */HT* to convert hidden text
  - `RtfWinForms/RtfWindows`: Conversion is considering the text selection
  - Added projects and solutions for Visual Studio 2005
  - Download now contains the binaries
- 13th October, 2008
  - `RtfImageConverter`: New property `ConvertedImages` which provides information about the converted images using `IRtfConvertedImageInfo`
- 3rd October, 2008
  - `RtfHtmlConverter`: Fixed encoding of image file names
- 26th September, 2008

    - Added support for *\pict* tags wrapped in a **\shppict* group
    - `RtfGroup`: Extended debugging visualization

- 23rd September, 2008
    - `RtfParser`: Fixed local group fonts

- 18th September, 2008
    - `RtfSpec`: New tag constants for the theme fonts and stylesheet
    - `RtfParser`: Added support for dealing with theme fonts during decoding (ugly but necessary when such fonts are used for hex encoded content)
    - `RtfFontBuilder`: Added support for theme fonts
    - `RtfFontBuilder`: Added support for font names with an alternative representation
    - `RtfFontTableBuilder`: Added support for theme fonts
    - `RtfInterpeter`: Added special support for stylesheets by ignoring their content (to prevent the style names from appearing in the document content)
    - Added the test document (in two variants) as unit test input

- 31st July, 2008
    - New Windows Forms sample application which demonstrates a simple conversion from RTF to Text/XML/HTML
    - New WPF sample application which demonstrates a simple conversion from RTF to Text/XML/HTML
    - `IRtfInterpreterContext`: Replaced `int DefaultFontIndex` by `string DefaultFontId` to support WPF `RichTextBox` font indexing

- 14th July, 2008
    - `RtfHtmlConverter`: New class `IRtfHtmlStyleConverter` which provides external style conversion
    - `RtfHtmlConverter`: Changed default image type from GIF to JPEG
    - `RtfHtmlConverter`: Fixed alignment of images

- 11th July, 2008
    - Support for character set decoding per font
    - Support for decoding multi-byte hex-encoded characters (this handles East Asian fonts commonly encoded in this way instead of using Unicode)
    - Special treatment for the Windows legacy pseudo codepage 42 (mapped to the ANSI codepage)
    - All command line applications: new command line parameter */CE* for specifying the output encoding (such as UTF-8 or UTF-16 a.k.a. Unicode)
    - `Rtf2Html`: Fixed HTML-encoded string
    - Command line application *Rtf2Html*: New command line parameter */CS* for the HTML character set
    - `RtfParserTest` and `RtfInterpreterTest`: New unit-tests for multi-byte character set decoding
    - Minor bug fixes

- 3rd July, 2008
    - Command line applications: Fixed exception in cases when log parser is not used

- 1st July, 2008
    - Initial public release

License

Share

About the Author

Comments and Discussions

Add a Comment or Question

Email Alerts | Search Comments

First Prev Next

RtfUndefinedFontException
Member 14476201 31-May-19 10:37

Three problems converting RTF to text, duplicate font id, Invalid Font size, undefined font
Member 13848428 27-Feb-19 13:33

Rtf2Mht
yetibrain 11-Feb-19 21:04

Rtf2Html error (Itenso.Rtf.RtfStructureException: 'a text cannot appear on root level, must be child of a group: '')
eyanson 10-Jan-19 21:21

License Details
Thamotharan G 26-Nov-18 21:59

Convert rtf with table
stefano1856 8-Oct-17 23:37

rtf to xml converter
Jayesh Sapariya 14-Aug-17 12:39

Embedded Image body is not working here
Member 11398072 12-Jul-17 18:42

RichTextBox edited RTF file not converting to HTML
Member 12827530 13-Jun-17 3:54

Html@Rtf
Gurgen Chlingaryan 23-May-17 14:43

Re: Html@Rtf
Jani Giannoudis 30-May-17 18:09

Gtihub Up to date?
martinrj30 17-May-17 15:43

Re: Gtihub Up to date?
Jani Giannoudis 30-May-17 18:07

Is there a setting to do before running the project?
Member 13124909 8-May-17 13:14

Re: Is there a setting to do before running the project?
Jani Giannoudis 30-May-17 18:04

rtf2html command line
YOUSFI 18-Apr-17 7:34

Re: rtf2html command line
Jani Giannoudis 20-Apr-17 0:09

Great job!
stefano1856 17-Apr-17 16:23

Re: Great job!
Jani Giannoudis 30-May-17 18:19

Great tool, saved me hours!
Martin Hart Turner 7-Oct-16 19:50

Tables plus line formatting
warpengine 2-Dec-15 6:40

Re: Tables plus line formatting
martinrj30 3-Dec-15 12:03

ConvertRtf2Image is not creating an image
Luzzifuge 23-Oct-15 2:33

Header
vladipron 19-Mar-15 23:18

Does it support parsing of RTF file containing table ?
Member 11437933 9-Feb-15 15:25

Refresh

1 2 3 4 5 6 7 8 9 10 11 Next »

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

Permalink
Advertise
Privacy
Cookies
Terms of Use

Layout: fixed | fluid

Server Web01
Version 2.8.190619.2